UNIVERSITY OF CINCINNATI

Date: November 16, 2007

I, Balasubramanian Sethuraman,

hereby submit this work as part of the requirements for the degree of:

Doctor of Philosophy

in:

Computer Science and Engineering

It is entitled:

Novel Methodologies for Efficient Networks-on-Chip Implementation on Reconfigurable Devices

This work and its defense approved by:

Chair: Dr. Ranga Vemuri

Dr. Harold W. Carter

Dr. Wen-Ben Jone

Dr. Karen Tomko

Dr. Karam Chatha

Novel Methodologies for Efficient Networks-on-Chip implementation on Reconfigurable Devices

A Dissertation submitted to the

Division of Research and Advanced Studies
of the University of Cincinnati

in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in the Department of
Electrical and Computer Engineering
of the College of Engineering

November 2007

by

Balasubramanian Sethuraman

Bachelor of Engineering (B.E.) in Computer Science & Engineering
Sri Venkateswara College of Engineering
University of Madras, Chennai, Tamil Nadu, India
2002

Thesis Advisor and Committee Chair: Dr. Ranga Vemuri

Abstract

International Technology Roadmap for Semiconductors (ITRS) project the latest trend moving towards a system-level and platform-based design, involving large percentage of design reuse. Different Intellectual Property (IP) cores, including processor and memory, are interconnected to build a typical System-on-Chip (SoC) architectures. Larger SoC designs dictate the data communication to happen over the global interconnects. At 45 nm and below, interconnects have profound impact on the performance (and power), due to increased delays and cross-coupling from multiple sources. Hence, attaining timing closure with reasonable performance and low power is increasingly becoming impractical. Also, the traditional bus based interconnection architectures present synchronization nightmare in a heterogenous System-on-Chip environment. At system level, the performance of the shared-bus start to deteriorate with increased number of cores.

Networks-on-chip (NoC) has been proposed as a new design paradigm to solve the communication and performance bottlenecks in the modern System-on-Chip designs. Unlike the shared-bus approach, the central idea in an NoC is to implement interconnection of various IP cores using on-chip packet-switched networks.

Due to reduced development costs and shorter design cycles and Time-To-Market, reconfigurable devices, especially, the FPGAs are increasingly being used in low/medium volume applications in place of their ASIC counterparts. Due to the scalability issues present in the use of shared-bus, NoC is gaining attention in the latest FPGA-based SoCs. In spite of the advantages, being a typical shared network, an NoC suffers from bottlenecks involving hop latency, congestion, bandwidth violations and increased area. .

In this thesis, we innovate and implement novelty to realize efficient Networks-on-Chip using commercial Xilinx FPGAs. We present tangible solutions for the issues that plague the efficient Networks-on-Chip implementation on the reconfigurable fabric.

First, we concentrate this area overhead issue, the solutions to which actually resulted in many-fold advantages. Area is at a premium on an FPGA and therefore, the communication network should be as small as possible. The on-chip micro network area can be reduced by: (1) Using a simple router without sacrificing on the performance, and (2) Reducing the number of routers. Implementing the first idea, we develop a light weight parallel router (LiPaR), with multiple optimizations that resulted in a significant reduction logic area usage.

The highlight of this dissertation remains in the translation of the second idea with the proposition of a novel router design that can handle multiple logic cores simultaneously, without any performance penalty. The new Multi Local Port Router (MLPR) provided many-fold advantages including reduction in area, power, transit time $\&$ congestion, and most importantly, bandwidth optimization, resulting in an efficient and high performance NoC design. Essentially, the MLPR is a marriage between switch-based and router-based interconnection network.

A NoC system comprising MLPRs represents a complex design environment and hence, generation of an efficient Network-on-Chip configuration is a great challenge. We present an exhaustive-search based optiMap algorithm (finding optimal solutions) and a heuristic based fast mapping cMap algorithm. The results portray a dramatic reduction in the latency as well as the number of packets flowing in the NoC mesh.

All the ports of an MLPR have the same mesh co-ordinate, thus, providing an opportunity to multicast to all the cores attached to the same MLPR, exploiting which we present a modified router architecture called Multi^2 Router. Utilizing the multicast capability, we present an energy-efficient NoC configuration generation approach (μMap), targeting data packet traffic reduction in the network.

Optimization for performance, latency or area constraints favor addition of more ports onto a single router. But, after extensive experimentation, we find a point of diminishing returns with regard to the power efficiency in using larger MLPRs. In addition to the average power increase, we observe the occurrences of several IR drop violations with increased port count, thus presenting a tradeoff between performance and power efficiency.

Task graphs in modern SoCs are not static in nature and the variations in the intercommunication patterns and bandwidth requirements need to taken into consideration during NoC architecture generation. Hence, we present technique to estimate the Minimum BandWidth Guarantee (MBWG) required for a given topology and extend to find the NoC architecture minimizing the MBWG, thereby, helping prevent surprises in terms of bandwidth violations.

In summary, the dissertation presents novel and efficient router designs, supported by the NoC architecture generation algorithms. Despite having an FPGA bias, the ideas proposed in this research are equally applicable to ASICs, thus, improving and taking forward an efficient NoC design flow.

To

My Dearest Parents

&

My Mentors

Acknowledgements

"If we knew what it was we were doing, it would not be called research, would it?"

"We can't solve problems by using the same kind of thinking we used when we created them."

"Imagination is more important than knowledge. For knowledge is limited to all we now know and understand, while imagination embraces the entire world, and all there ever will be to know and understand."

– Albert Einstein (1879-1955)

First and foremost, I thank the God Almighty for instilling the necessary strength and confidence in me to face the strides in life. The various stages that transpired in my life, helping me what I am today, make me truly believe in the existence of the supreme being. I am grateful and indebted to my parents for their support and encouragement, assisting my way up in the intellectual ladder.

I wish to thank the members of my dissertation committee Dr. Ranga Vemuri, Dr. Harold Carter, Dr. Wen-Ben Jone, Dr. Karen Tomko and Dr. Karam Chatha (Arizona State University). I would like to acknowledge the support of Ohio Board of Regents PhD Enhancement Program.

My heartfelt and sincere thanks to my advisor Dr. Ranga Vemuri, whose constant support and guidance gave a proper shape to my education at University of Cincinnati. He is truly an exceptional teacher in the field of Computer Engineering, with a unsurpassing teaching style that always kept together the attention of all his students. I really cherish the various moments of his courses, including the late-nights for the projects of Physical VLSI Design and VLSI Design Automation courses. I got to experience first-hand how it feels like to complete a project under tight time constraints under him. Above all, I greatly admire his stress on the academic integrity and the perfection of work. I really enjoyed the many lighter moments that I shared with him both in and around UC.

I had never been so open and candid with a professor as I had been with Dr. Wen-Ben Jone, who was more of a friend to me! The fun-filled interactions that I had with Dr. Jone throughout my stay at UC will always be close to my heart. During the various quarters that I served as his teaching assistant, apart from his encouraging comments, the level of importance that he gave to my thoughts with regard to the course conduct as well as the grading, really inspired me a lot! I wish to express my wholehearted gratitude towards Dr. Jone for nominating and assisting me in winning the Outstanding Student Services

Award at ECECS department, which eventually helped me in receiving the Outstanding International Student Award. I am thankful to Dr. Carter for his unflinching support during my tenure as the president of the Graduate Student Association of the ECECS department, which greatly helped to fetch the award for the ECECS GSA.

Being part of and working at DDEL (Digital Design Environments Laboratory) was a pleasant, a fun-filled and a memorable experience, transforming DDEL my second home! I fondly recall the discussions and interactions that I had over the years with my peers at DDEL. The rapport that I had with my fellow DDELites (Shubhankar, Vijay, Angan, Prasun, Priyanka, Srividhya, Almitra, Balaji, Suman, Jawad, Harish, . . .) will be unforgettable! I am much obliged to Shubhankar, who had been a significant influence in my life at UC - a friend & a mentor, always engaging me in many thought-provoking discussions (technical and otherwise) and being readily available for anything and everything! The banters that I shared with Vijay were delightful and my thanks to him for being approachable at all times. Not to mention the cozy discussions and arguments that I had with Angan, watched amusingly by Almitra! Special thanks to Prasun for his collaborative efforts during the initial stages of my research, which gave the much needed push to an otherwise fledgling topic. I am thankful to Balaji, Angan and Suman for helping me during my final dissertation defense.

All together, it had been truly a great and a fulfilling experience here at University of Cincinnati, not only in terms of my educational advances, but also in terms of the different roles that I got an opportunity to play, making it the most memorable five years of my life! The roles of teaching assistant and adjunct faculty had been extremely rewarding to me, refining my understanding of the concepts. I am thankful to all my friends (Parineeta, Jothiram, Lakshmi, Arun, Anand, Julie, . . .), whom I enjoyed working with towards the events of the various UC student organizations - GSA, SABHA & Cultural Connections. Thanks to all my VLSI batchmates and the tamil junta (Sathish, Sridhar, Karthik, Ali, Jayaram, Prasanna, Kutty, Krishna, Hari, Diva, Sriram, . . .) for all the memorable and enjoyable moments at UC.

Last but not the least, I wish to gratefully remember and thank the mentors whom I had association with during my high school (Brinda & Jayashree) and undergrad college (Prof. Venkateswaran) for their constant motivation and guidance, which helped me define the direction and the quality of my academic career. Thanks to all who have positively influenced me directly or indirectly, thus helping me achieve laurels and become the person whom I am today!

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The Semiconductor industry has shown rapid advancements in various facets, including design and fabrication techniques, while obeying the Moore's law and in some cases pushing over the limits so as to keep the productivity and eventually the profitability with growing competition in the market. In the current nanometer era of design, more challenges are inevitable both in design and fabrication compared against the sub-micron and the Deep Sub Micron (DSM) period. Continued Technology scaling helps integration of large number of transistors on the same chip area. According to the International Technology Roadmap for Semiconductors (ITRS), we are looking into at a multi-million transistor chips, with feature sizes at 50nm and below, operating at clock frequencies over 10GHz. The tremendous increase in the transistor count results in the effective gate count available, thus, paving a path towards implementing large designs, which were otherwise realized as the interconnection of designs spanning across multiple chips. System-on-Chip is a new buzz word and represents a complex integration of various heterogenous designs called Design Cores (refer Figure 1.1).

1.1 Interconnects in the Nano-meter Era

1.1.1 Impact of Technology Scaling

In the billion-transistor era, designers can pack large number of modules on a single chip. But, technology scaling does not present a rosy picture in terms of the interconnects. The assumption of the interconnect delay being negligible compared to the gate delay is no longer valid. The actual scenario is that the technology scaling is better for transistors

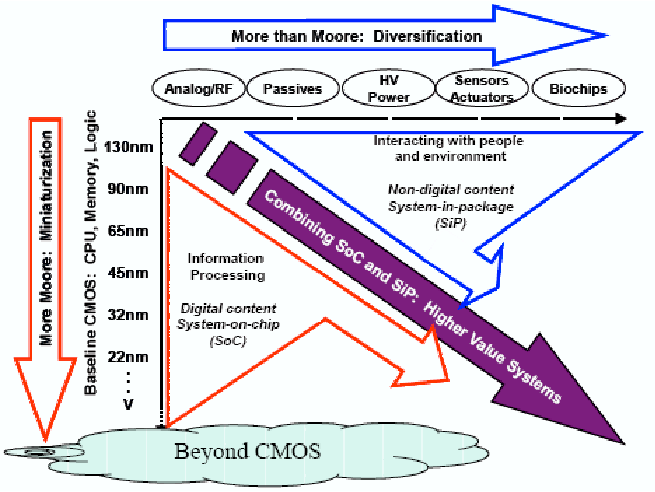


Figure 1.1: Chip Design under Moore's law [Sem06]

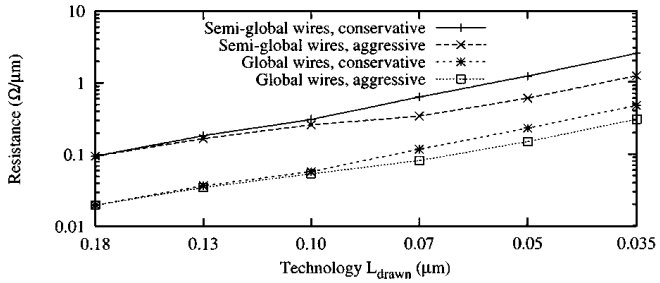


Figure 1.2: Effect of scaling of wires on the resistance [HMH01]

compared to the interconnect wires. As designs scale to newer technologies, the active device parameters including the gate length and the speed of operation of the transistor get improved. But, as the wires get shorter, they do not scale uniformly with length and the delays over the global communication wires start to increase.

Scaling of wire includes prediction various physical parameters that affect the electrical properties in an circuit. Scaling is performed in using a set of both conservative and aggressive parameters to observe the overall trend when modelling interconnects. Figure 1.2, as shown in [HMH01], shows the drastic increase of resistance with technology scaling. The effect of technology scaling is very moderate for capacitance and inductance [Sem06, HMH01]. It can be seen that the conservative and aggressive scalings estimates are very close (Figure 1.3).

To summarize, the relative change between the speed of wires and the speed of gates has been modest. But, at 50nm and below, the performance gap between the wire

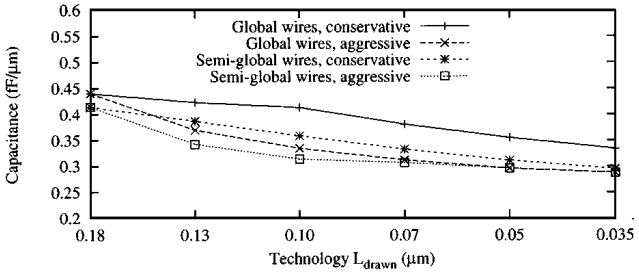


Figure 1.3: Effect of scaling of wires on the capacitance [HMH01]

and the gate widen significantly. With advancements in process technology, the delay and the crosstalk happening over the interconnects start to dominate the gate delay, thus affecting the performance of the system. In addition to the performance, the wires create considerable overheads with regard to the power consumption and the occupied area [Sem06, RSV97, KMN+00, SSM+01]. With addition of large number of modules per chip, the accumulation of wire problems become unmanageable.

1.1.2 Effects in Nanometer Design Regime

In the nanometer regime, there is a drastic rise in cross-coupling effects, noise and transient errors, and start affecting the circuit operation. With decreased feature size and reduced operating voltage circuits are more susceptible to noise sources and hence are vulnerable to failures. Some of the sources that affect the timing closure are the parasitic capacitance, inductance & resistance, signal delay, simultaneous switching noise (SSN), Electro-Magnetic-Inductive effects, crosstalk noise, substrate coupling noise, and leakage currents.

Coupling noise and the resultant crosstalk is a complex and serious problem during chip design, since both mutual capacitance and inductance effects come into picture. Modelling the effect of mutual inductance is much more difficult. Unlike the capacitive coupling, inductive coupling affects the victim wire(s) in an opposite fashion. Also, while capacitive coupling mostly affects the neighboring interconnects, inductive coupling happens over a bigger range. Switching of large number of wires that are present in a bus-like fashion compound the problem of inductive coupling. The modern System-on-Chips with thicker Wires that span long lengths and clock frequency in the GHz range will require interconnection analysis considering inductance and inductive coupling effects on interconnect delay and noise. Designers employ techniques to ameliorate this situation by using

shielding lines using power/ground rails or by inserting buffers at regular intervals.

In addition, effects like instantaneous voltage drop and ground bounce exacerbate the problem with signal integrity. Arriving at effective analytical models that predict the overall effect of the various noise sources is extremely difficult. Uncertainties and inaccuracies in the estimation of the cumulative effect of these device effects results in a circuit having critical failures.

1.1.3 Low Power operation

In addition to high performance chip designs, the current design trend is towards building designs that consume *low power*. To achieve this goal, the noise margin is kept low for operating the circuit with low voltage swing. In addition to this, the supply and threshold voltages are scaled to minimize the power consumption of the system. For the nodes that are weakly driven and the nodes that operate with low voltage swing, coupling due to mutual capacitance and inductance result in illegal or indeterminate states causing circuit failures. In this context, the global interconnects result in the switching of very high capacitance (quadratic) and hence lead to high power consumption.

1.1.4 Impact on circuit delay and Bandwidth

As projected by ITRS, the rising impedance of the interconnects has resulted in the interconnect delay getting multiplied by a large factor. The situation is worsened with added problems caused by skew and jitter affecting the global signals, especially, the clock.

To tackle the growing interconnect delay effectively, designers break the long interconnects by inserting repeaters at regular intervals (Figure 1.4) . Also, the repeaters help to reduce the inductive current paths effectively, thereby, restricting the inductive noise at low levels. But, the repeaters are not available for free and have associated penalties. The repeaters that are realized as inverters, consume a significant area and add to the floorplan and power/ground rail routing woes. Also, they result in a transfer involving multiple clock cycles, though providing an opportunity to increase the bandwidth. Optimistic predictions estimate the propagation delays for highly optimized global wires to be between 6-10 clock cycles at 50nm [BdM02].

Thus, we observe that transmission of signal on a global basis will only create more problems. Due to skew and jitter effects, global signal synchronization is becoming

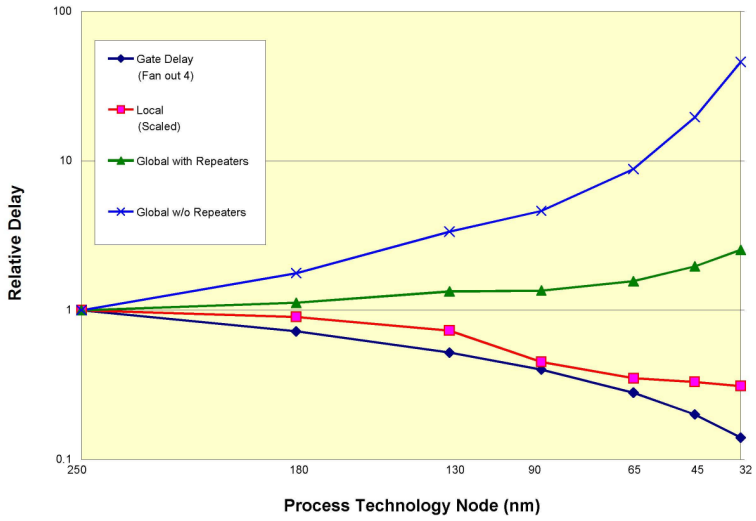


Figure 1.4: Need for repeaters for global interconnects [Sem06]

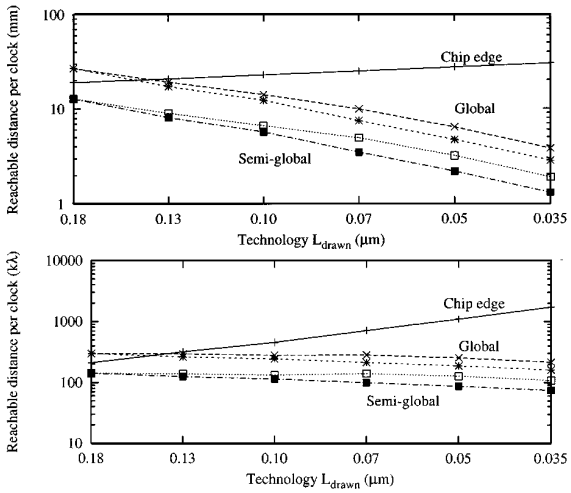


Figure 1.5: Distance reachable in one clock cycle [HMH01]

highly impractical. According to the best estimates, the realistic clock frequency at 50nm is less than 4GHz (taking the *Time of Flight* into account), whereas, the active devices have the ability of operating at 10GHz. Figure 1.5 shows how far a signal can reach within a single clock. Wires affect both circuit delay and robustness and are turning out to be show-stoppers in terms of performance and power. Realizing design closure under such a complex design environment having many noise sources is becoming exceedingly difficult, under tight productivity constraints.

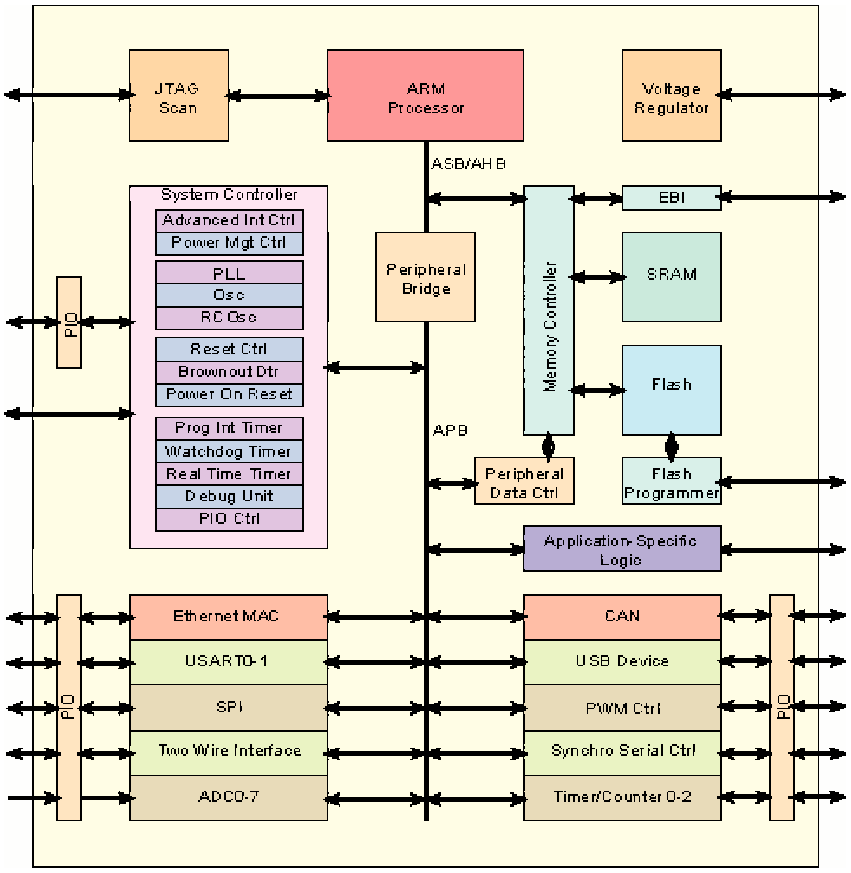


Figure 1.6: Microcontroller-based System-on-a-Chip [Wik06]

1.2 System-on-Chip & Platform-based Design

System-on-a-chip (SoC or SOC) a complex interconnection of various functional modules or components of a computer or other electronic system into a single chip (Figure 1.6). The modules can be digital, analog, mixed-signal and even radio-frequency components, all brought inside a single chip. The typical components of a System-on-Chip is summarized as follows [Wik06].

- One or more microcontroller, microprocessor or DSP core(s)
- Memory blocks including a selection of ROM, RAM, EEPROM and Flash
- Timing sources including oscillators and phase-locked loops
- Peripherals including counter-timers, real-time timers and power-on reset generators

- External interfaces including industry standards such as USB, FireWire, Ethernet, USART, SPI

- Analog interfaces including ADCs and DACs
- Voltage regulators & Power management circuits

These blocks are connected by means of interconnection networks which includes industry-standard shared-bus cores like AMBA bus from ARM or STbus from STMicro [AMB06, STB06]. Other than the hardware resources summarized above, a System-on-Chip design flow cycle also involves design of software to control the various hardware components like the microprocessors, controllers and other IP cores/interfaces. As illustrated in 1.7, the design flow (refer Figure) for an SoC involves development of both the hardware and the software in a concurrent fashion.

The functional modules themselves can support different features, with varied performance and power levels. In a heterogenous environment, we can have a System-on-Chip wherein the different islands or modules operating at varied clock frequencies. With such a complex environment having various interconnected modules, design and full synthesis of System-on-Chip designs from scratch is highly impractical. More than design, verification of such an unified system is a nightmare for the engineers. Also, it is hard to meet the Time-To-Market (TTM) demands with a fine grained approach.

Answer to the above issue lies in the traditional approach of divide-and-conquer. Design Cores are getting prominence for building System-on-Chip designs, which are basically the (parameterized) functional blocks or hardware elements that are pre-verified and pre-qualified for functional correctness, power, performance (frequency of operation, throughput & bandwidth) and other constraints of merit. As we look into future, the design cores will encompass a variety of heterogenous cores operating at various voltage levels and frequencies.

Building a System-on-Chip is essentially comprised of assembling a set of pre-existing and pre-verified components (Design cores), according to a complex & application-specific architecture, while satisfying the performance and power constraints. According to ITRS 2005 edition, future designs are projected to employ more design reuse so as to increase the productivity. According to optimistic estimates, the gains in terms of productivity that can be achieved as high as 200% and is projected to be the key element for system level design [Sem06].

A typical application is in the area of platform-based or embedded systems. A platform is a specific combination of system components that support specific application areas (e.g,. automotive, consumer electronics/multimedia, wireless, communication

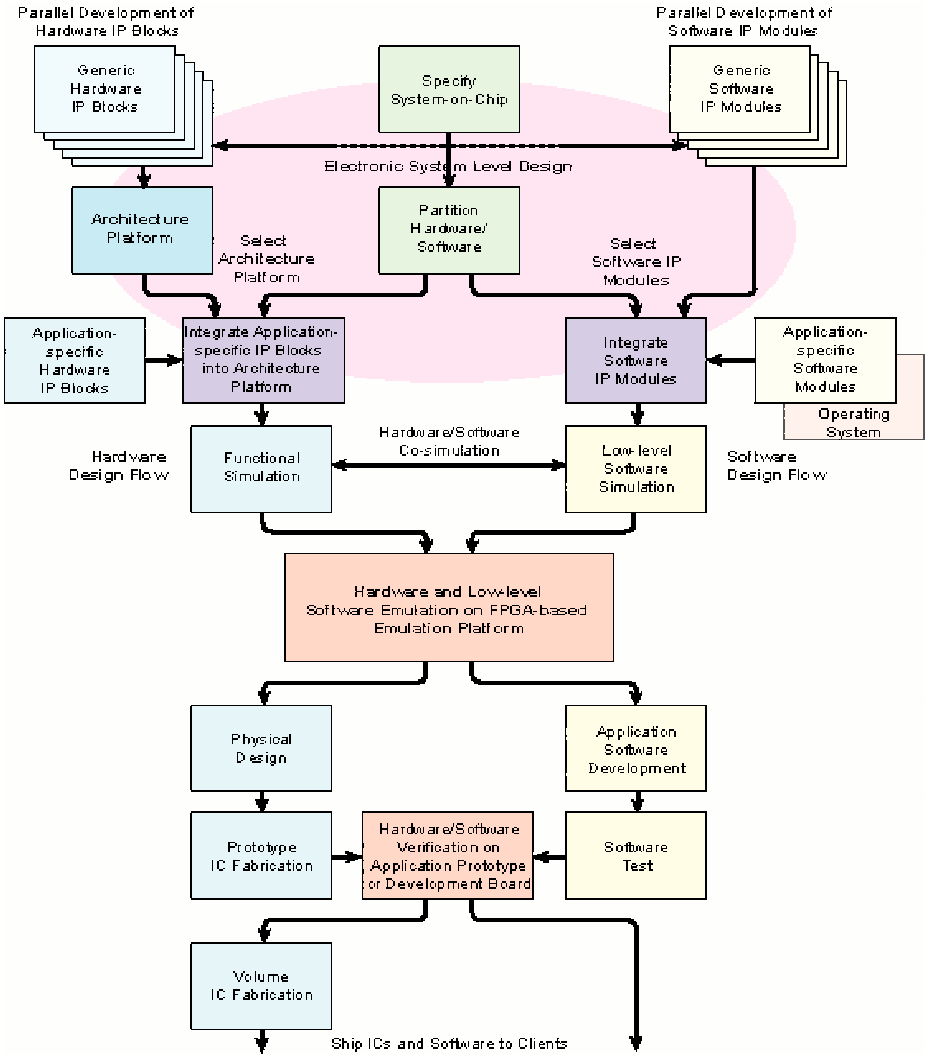


Figure 1.7: System-on-a-Chip Design Flow [Wik06]

infrastructure, memory, customizable analog and digital log, and virtual sockets for new logic [Sem06]. Functional requirements for the given application area is satisfied by integrating a number of components, which are basically the design cores or Intellectual Property (IP) cores. Customization for a particular platform to arrive at a System-on-Chip design boils down to an exploration of a design space, subject to various constraints. The communication backbone and the processor/memory choices are pre-determined for a particular platform and the designers have freedom in choosing certain customization parameters and optional IP from a library.

Platform-based design also involves Hardware-Software partitioning, wherein the delineation between what goes in as hardware components and what is translated to software is drawn, so that the various system figures of merit (system performance, energy consumption, on-chip communications, bandwidth) are optimized. As aptly described by ITRS [Sem06], platform based design is an important driver for design productivity as it greatly promotes design reuse, under ever increasing TTM constraints. Different platforms are expected to converge in future, thanks to the great advancements being achieved in fabrication and manufacturing processes. As seen from the Table 1.1, we are observing a full-support for 97% of the platforms available by the year 2020 and more than 55% of the design being reused.*

1.3 System-on-Chip & Reconfigurability

Another dimension of the future system level design of SoCs is the percentage of reconfigurability that is available. The complexity of the systems follow an upscale trend and errors will be part and parcel of future System-on-Chips. Hence, in order to be able to fix the errors after fabrication and exploit the concept of design re-use (reprogramming the existing hardware blocks to realize new tasks), reconfigurability is an essential component of system design. We expect upto 62% of the design functionality (either in hardware or software) to be reconfigurable, by the year 2020 (refer the last row in Table 1.1).

*Full Support for a particular platform means an integrated development environment that supports and automates architectural exploration, HW/SW partitioning, architectural/platform mapping, HW/SW co-verification, performance/area/power/costs trade-offs, HW and SW synthesis and HW/SW interface synthesis for that platform [Sem06]

Year of Production	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
Design Reuse																
Design block reuse % to all logic size	32%	33%	35%	36%	38%	40%	41%	42%	44%	46%	48%	49%	51%	52%	54%	55%
Platform Based Design																
Available platforms Normalized to 100% in the start year	96%	88%	83%	83%	75%	67%	60%	55%	50%	46%	43%	42%	39%	36%	33%	32%
Platforms supported % of platforms fully supported by tools	3%	6%	10%	25%	35%	50%	57%	64%	75%	80%	85%	90%	92%	94%	95%	97%
Reconfigurability																
SOC reconfigurability % of SOC functionality reconfigurable	23%	26%	28%	28%	30%	35%	38%	40%	42%	45%	48%	50%	53%	56%	60%	62%

Table 1.1: System Level Design Requirements - Near-term years [Sem06]

1.4 Future System-on-Chip: Summary of the real picture

Attaining design closure for timing with better signal integrity and reduced crosstalk & glitch is increasingly becoming important. Signal communication across the global interconnects do not present a rosy picture for the future System-on-Chips [Kon04]. As we look at the future System-on-Chips, integration of various digital, analog and mixed-signal designs represents a very complex environment. This prompts us to adopt a platform-based design involving larger percentage of design reuse in terms of the Intellectual Property Cores. In this context, providing point-to-point interconnection by means of long wires spanning the length and breadth of the chip is out of question. The major challenges include irregular topology, wires with unpredictable electrical parameters as it is difficult to model beforehand and nodes with variable in and out degrees and most importantly, the aforementioned nanometer era design challenges. The growing performance issues created by the conventional wires or wiring methodologies (global interconnects) demand for new interconnection methods. The use of interconnection networks provides an excellent opportunity to mitigate the challenges of implementing interconnects in the nanometer regime [SSM+01].

1.5 Interconnection Networks

According to [DT04], the term *Interconnection network* refers to a *programmable system that can transport data between terminals*. In simple words, it is a shared medium of communication which is *time-multiplexed* while establishing the communications. The most important application of the interconnection in the mainstream digital systems is in the intercommunication between processor, memory, and input/output (I/O) devices & controllers. The importance of an interconnection network is highlighted in a multiprocessor environment involving large scale communication between the processor and memory (shared/local). Broadly, the interconnection networks can be classified into two distinct

types.

- Bus-based interconnection networks (or) Shared-Medium systems
- Distributed Point-to-Point interconnection networks

1.5.1 Shared-Medium Systems

Traditionally, bus-based systems were the interconnection medium of choice because of the various offered advantages. Shared-bus approach defines standard-interfaces, that make the applicability of the bus universal and modular. There are no routing protocols as the data present on the bus wires is accessible to all the cores connected to it. The advantages and disadvantages of a shared-bus scheme are summarized as follows [GG02],

Advantages

- Bus is a well researched and well studied interconnection medium with standard interfaces [AMB06, STB06].
- The process of connection initiation and establishment is simple and straightforward.

- By virtue of being a standard, it is compatible with almost all Intellectual Property cores, microprocessor cores, memory controllers, micro-controllers and input/output device controllers.
- The area overhead of the bus-based scheme is very low as it is basically a set of wires interconnecting the the various cores attached plus a small control (or) arbitration logic to grant access of the bus.
- Bus supports various levels of protocols including priority based access and broadcasting.

Disadvantages

- Every node or cores that gets attached to the common bus adds to the total load of the bus. This translates to large and unpredictable switching delays, thus resulting in degraded performance.
- The direct impact of the increased capacitive load seen by the set of wires forming the bus, has a profound impact on the total power consumption.

- In the Nanometer era, shared-bus system will necessitate the use of the dedicated wiring approach, which in turn forces the use of long wires (global interconnects). Also, we are looking at an operating data width of 64 bits and beyond. As explained in the previous sections, this global interconnect environment having longer and wider bunch of wires present insurmountable challenges in the pursuit of design closure. The problems raised of inductive and capacitive coupling get compounded as it is very difficult to model them and accurately predict in the early phase of the chip design cycle.
- The bus has a fixed static bandwidth allocated to it which can be efficiently used. But, since it is a common shared medium, there is no possibility of having simultaneous parallel communications. The absence of system-level communication parallelism is a major drawback [DT01].
- Above all, the shared-bus system presents a great deal of scalability issues to contend with. Since, the cores operate via a common communication medium, the throughput of the system takes a hit with increased number of cores attached to the bus. The arbitration process becomes longer and the control logic has to be designed from scratch making it application-specific, thus adding to the design effort required to realize the system.

There are new and advanced bus communication techniques proposed including multi-bus and split-transaction bus. A *Multi-bus* is a collection of independent local buses which are integrated by means of *bridges*. The key idea here is to increase the parallel operation possible in the overall shared-bus and to reduce the level of capacitance switched as the component bus lengths are comparatively very small [HP00]. The authors in [HP00] claim that the power savings using the multi-bus approach vary between 16% and 50%, compared to the traditional global bus.

Normally, the bus master does not relinquish the bus unless the transaction is completely finished. The variable response time increases the time during which the bus is idle and hence the bandwidth is wasted significantly. *Split-Transaction bus* is an approach to reduce the amount of time the bus remains idle in between transactions. Here, the bus is released after the request and hence, the response transaction has to obtain access of the bus to complete the data transfer process. Also, the commercial bus designs like IBM CoreConnect bus [Arc05], the ARM AMBA bus [AMB06] and the STMicro's STBus [STB06] incorporate advanced techniques like having bus hierarchy with varied levels of operating frequency and performance.

In spite of all these improvements, the shared-bus has inherent scalability issues with large number of operating cores. Thus, at system level, the increasing performance gap between the processor and the interconnects necessitate the use of point-to-point dedicated and distributed interconnection networks in place of the shared-bus approach, in order to satisfy the communication needs an high-performance System-on-Chip.

1.5.2 Distributed Point-to-Point Interconnection Networks

The distributed point-to-point switched network (*direct network*) is the new paradigmatic alternate to the common shared-bus architecture. The notion of long, common and shared communication fabric is replaced by the short, point-to-point, distributed and independent switching communication fabric. Here, the data transfer is effected between the co-operating pairs of switching elements or routers. There is no central arbitration as the control of the communication fabric is distributed. The idea of gaining access of the communication medium by arbitration or polling is done away with. Instead, the data transfers happen between and within the routers and as long as the destination point (can be a logic core or another router) is free and available to participate in the data transfer process.

A *Direct Network* is made of the communication backbone comprising of the switching elements called routers. The cores including the processors, memory or any IP, get attached to the router elements. The cores initiate data transfers to a destination core (using proper addressing mechanisms) by pushing the data to be communicated to the corresponding router element. The responsibility of transporting the data properly to the destination core is offloaded to the routers of the direct network and the sender/receiver cores are totally abstracted from the actual communication that is happening over the network backbone. The data is made to hop between the neighboring routers until the data reaches the router to which the destination logic core is attached. At this point, the router pushes the data into the destination logic core. We see that the cores can be involved either in sending or receipt of data, and hence, this necessitates having two independent channels of communication. Namely, the input channel to receive the data and the output channel to send the data. A bi-directional channel is possible in place the two uni-directional channel, but, this will result in severe performance degradation as the possibility of simultaneous sending and receipt of data is removed. That is the birds eye view of a direct network from which is derived the new communication paradigm called *Networks-on-Chip*.

As we can clearly see, because of the distributed communication nature, the direct

networks scales nicely with the addition of large number of cores (processors, memory, input/output devices, microcontrollers or IPs). The concurrent operation and communication results in increased bandwidth and processing power of the system. The same trend can be observed in the computer bus implementing for attaching peripheral devices to a computer motherboard. The Peripheral Component Interconnect standard (PCI) is a popular bus-standard that follows a shared-bus topology for attaching different (PCI) devices onto a common bus (e.g., sound card, USB card, network card, etc.) for communicating with the CPU. PCI-Express (PCIe) standard is introduced as an alternative to PCI bus standard to overcome the severe performance bottlenecks and poor scalability encountered with PCI standard. PCIe standard presents a drastic improvement by adopting a point-to-point bus topology effected by means of a shared switch in place of the shared-bus. The switch routes the traffic over the bus by establishing point-to-point connections between the pairs of communicating devices present in the system. Thus, *off-chip or on-chip, use of interconnection networks is the effective answer to the constrained shared-bus approach*.

Apart from the classification of the interconnection networks discussed above, there exists a type referred as indirect network, which are predominantly switch-based networks. As against a direct network, the node of the communication backplane can operate as a switching element (forward packets to-and-fro between routers) or a terminal (interface to the logic core for sending and receiving data), but not both. Examples of such interconnection networks include crossbar networks and the multi-stage interconnection networks like butterfly networks and clos networks (non-blocking). Also, hybrid networks like hyperbuses, hypermeshes and hierarchical cluster-based networks are available in the literature [DYN98, DT04].

The point to be noted here is that a *Networks-on-Chip* is a generic term and can include both direct and indirect networks. In this thesis, we concentrate on the widely popular direct networks which are essentially router-based networks involving switching mechanism ranging between circuit switching and packet switching. The following chapter describes in detail the various parameters involved in a Networks-on-Chip (NoC).

1.6 System-on-Chip & FPGAs

ASICs are increasingly being replaced by FPGAs for applications with low to medium volume, due to longer design cycles and high cost. Decreasing feature size and

improved fabrication techniques have enabled the current FPGAs to provide larger gate count with increased performance and reduced power dissipation [Xil06a]. Modern FPGAs [Xil06b] provide greater scope for realizing System-on-Chip designs, because of the availability of several Intellectual Property (IP) cores, including embedded hard & soft processors, memory, Digital Signal Processing (DSP) and Input/Output (I/O) Transceiver cores [Xil05]. For instance, the latest Xilinx Virtex 4 devices offer a variety of capabilities that can be summarized as follows [Xil06a].

- Advanced system clock management with operating frequency over 500MHz
- Large and versatile memory resources (SmartRAM)
- Parallel connectivity including differential I/O (1+ Gbps) and single-ended I/O (600 Mbps)
- RocketIO transceivers (622 Mbps 10.3125 Gbps)
- Ethernet Media Access Controller (MAC)
- Embedded IBM PowerPC 405 Processor hard cores (450 MHz)
- Auxiliary Processor Unit (APU) controller
- Dedicated Ultra-high-performance DSP slices (XtremeDSP)

In addition, the Embedded Development Kit (EDK 6.3i) has wide range of IP cores for realizing embedded applications using the Xilinx FPGAs [Xil06a]. To cite some,

- Soft processing cores in the form of MicroBlaze (32 bit RISC) & PicoBlaze core (8 bit microcontroller), along with low-latency IEEE-754 compatible Floating Point Unit (FPU).
- IBM CoreConnect Bus Technology infrastructure cores, including the OPB (On-chip Peripheral Bus), PLB (Processor Local Bus) and DCR (Device Control Register).
- High value CoreConnect cores like EMAC 10/100/1000, PCI, UART 16450/550 etc.

- Wide range of interfaces and controllers for memories (BRAM, SDRAM and external memory), timers, counters, UART and controllers for the OPB.
- Bus Functional Model Simulation (BFM).
- Base System Builder Wizard - generates system IP, example software application code and simplifies creation of HW for any board.
- XMD - Xilinx Microprocessor Debug engine for MicroBlaze and PowerPC Advanced system clock management with operating frequency over 500MHz

- Graphical memory map manager
- 'Platform Debug' support via integrated software (GNU gdb) and hardware debuggers (ChipScope Pro)
- Instruction set simulator for PowerPC and MicroBlaze
- Data2MEM - tool for loading on-chip memories

This presents an ideal environment for implementing embedded and System-on-Chip designs including the multi-processors. Also, as discussed in the earlier subsection, one of the important dimension of the future System-on-Chip design the ability to provide reconfigurability. With the System-on-Chip capabilities, the latest FPGAs help to achieve this end in a desired and satisfactory manner. FPGAs can be reconfigured according to the changing needs of the system even at runtime (dynamic & partial reconfiguration), and thus provide a versatile implementation platform. Tile based execution is one of the popular approaches for task execution. The size and the granularity of the tiles can be defined by the user according to the requirements.

Moreover, we have to contend poor scalability issue that haunts a shared-bus style of system design.

1.7 Motivation & Overview of the Thesis

All of the factors explained in previous motivate us to adopt the Networks-on-Chip of design for implementing System-on-Chip applications using the modern FPGAs. The dissertation is developed, organized and discussed related to the issues that primarily come into picture when SoC designs are implemented on FPGAs. Ideas explained in this research, though being restricted to FPGAs, are equally pertinent and applicable for ASICs.

In an FPGA, area is available at a premium, and hence the on-chip communication network should be as small as possible. This ensures that the maximum area can be utilized by the user logic while maintaining the performance of the on-chip network. Also, reduction in the logic blocks used in FPGAs has a direct impact on the power consumption and the timing [SBKV05]. The central component of an NoC architecture is a router or the switch element. Hence, it is prudent to make its area smaller. The on-chip network area can be reduced in the following ways.

1. Using a simple router supporting complete functionality, without sacrificing the performance [SBKV05].

2. Reducing the number of routers, without reducing the number of communicating logic cores [SV06b].

1.7.1 Light Weight Router for FPGAs

As a first step, we implement a light weight router design customized for the Xilinx FPGAs, which is capable of establishing multiple simultaneous connections between any two channels of the various ports of the router. The router occupied only 352 Xilinx Virtex-II Pro FPGA slices (2.57% of XC2VP30).

1.7.2 Multi Local Port Routers

With n available cores, the second strategy cannot be used, unless the cores are grouped together and made to share the same Network Interface (NI). This will complicate the system integration process and create severe performance bottlenecks.

To solve this *critical issue*, we propose to use the second strategy by introducing an architectural change to the router element to handle multiple design cores, simultaneously. The modified router architecture has more than one Local Port (LP), defined as Multi Local Port Router (MLPR) [SV06b]. The Network Interface is same as that of a traditional NoC architecture, with the change made only to the header packet and the decoding logic. In essence, we propose the idea of having a *Network within a Network* (explained in detail in Chapter 4).

We exhaustively analyze the merits of Multi Local Port Routers, including area $\&$ power reduction, decrease in transit time $\&$ congestion, and most importantly, optimization of the bandwidth, resulting in an efficient NoC design. We highlight certain issues that play a role in the use of the proposed approach for NoC design in FPGAs, which are equally pertinent and applicable to Networks-on-Chips realized on ASICs. Overall, the Multi Local Port Routers are observed to improve the performance of the NoC system [SV06b].

1.7.3 Efficient NoC architectures having MLPRs

A Network-on-Chip system comprising Multi Local Port Routers represents a complex design environment. Hence, generation of an efficient Network-on-Chip configuration is a great challenge. Further, obtaining the optimal mapping of cores is no longer a simple

nearest neighbor or shortest path finding algorithm. As a proof-of-concept, we present an exhaustive search algorithm (optiMap) that maps the input task graph optimally, minimizing the overall execution time. This is an exhaustive search algorithm and is guaranteed to find the optimal NoC configuration, and hence a formal proof is redundant. We test the algorithm on a wide variety of benchmarks and report the results. For a given set of constraints and objectives, the algorithm finds the optimum number of routers, the configuration of each router, the optimum mesh topology and the the best possible mapping of cores onto the NoC architecture. optiMap effectively automates the NoC design cycle by finding the optimum mesh topology and the final mapping for the given task communication graph.

1.7.4 Heuristic Fast Mapping Algorithm

The search space of the optiMap algorithm exploded for Network-on-Chip systems with larger number of communicating cores. This is because of the expensive configuration generation step (discussed in detail in later sections). The huge memory requirement (for generating the configurations) and large CPU runtime (several hours) render the use of optiMap infeasible for larger System-on-Chips.

Hence, we present a fast mapping algorithm (cMap) for generating NoC architectures using Multi Local Port Routers. The algorithm exploits the advantages offered by an Multi Local Port Router and starts by defining a minimum-dimension mesh. After the initial bandwidth-communication-cost based nearest-neighbor placement, the algorithm uses a force-directed approach to expand the mesh iteratively, as the cost gets reduced. The algorithm introduces the idea of *Folding* to improve the NoC design (Chapter 7).

Unlike the optiMap algorithm, the cMap algorithm is capable of handling task graphs of any size, giving near-optimal results (NoC architecture & the best possible mapping) within a couple of seconds. We experiment the algorithms on wide variety of synthetic & practical (including *MPEG4*, *MWD*, *FFT* and *VOPD*) benchmarks and exhaustively analyze the results. The results indicate significant gains on multiple parameters including area, performance and power. cMap effectively automates the Network-on-Chip design cycle by finding the best MLPR-based mesh topology and the final mapping of the given task communication graph.

1.7.5 Multi^2 Router

Given the fact that multiple cores can be attached to a single router, as a natural extension, we introduce capability to multi cast data packets in a traditional mesh based Networks-on-Chip design. We present the architectural modifications necessary to incorporate the multicasting feature.

1.7.6 Energy Efficient Networks-on-Chip

The ability to multicast helps to reduce the number of data packets that flow in the Networks-on-Chip mesh. The scenario of application mapping on to the NoC backbone changes when multicasting is considered. We modify the optiMap algorithm and experiment with a set of benchmarks. In addition to the performance gains, the results show a drastic reduction in the overall power consumed.

1.7.7 Power Issues in Larger Multiport Routers

With respect the latency and area overhead, larger multi port routers remain the undisputed choice and when dealing with larger bandwidth flow in the design of application-specific NoCs, it is intuitive to initiate transfers as intra-port rather than inter-port. But, there is absence of a clear picture with regard to the power efficiency of the larger multi port routers. Hence, we perform extensive power and IR analysis on the multi port routers and find the shortcomings in using larger multi port routers from an power efficiency angle. There is an increase in the average power and also more IR drop violations are observed with increased port count.

1.7.8 Handling Dynamic Task Structure

Modern System-on-Chips comprise of versatile and varied functional modules which inter-communicate in a highly dynamic fashion, thereby, introducing varying bandwidth constraints between various edges in the task graph. In such a scenario, an (near) optimal topology and mapping generated based on the static task graph structure may introduce severe bandwidth violations in a dynamic task environment. In order to better estimate the bandwidth requirements and hence detect any possible violations, we present a heuristic technique called dynaMap to find the NoC architecture requiring minimal bandwidth

guarantee (MBWG). Also, we analyze the impact of maximum port count on the MBWG required. The results demonstrate the need for a highly application-specific NoC architecture generation, catered to the task structure(s) at hand. The knowledge of the maximal bandwidth variation across links is highly beneficial during the NoC design, as it can help a great deal in preventing many unwanted surprises in terms of the bandwidth violations created.

1.7.9 Extension to Multi-FPGAs

Finally, we explore and discuss the applicability of the router designs and NoC architecture generation algorithms in an multi FPGA environment, having a set of devices interconnected in a mesh topology. The topology generation and mapping algorithms require minor modifications compared to ideas proposed related to MLPRs, whereas significant architectural changes are necessary so as to reduce the latency, without any sacrifice in the addressability of the entire System-on-Chip.

Despite the results and issues being based on FPGAs, the ideas proposed in this research can be easily extended without loss of generality and applied to ASICs.

1.8 Research Summary

Emerging Platform-FPGAs with embedded soft and hard processors cores can be used for System-on-Chip (SoC) designs. SoC systems represent a complex interconnection of various functional elements. Exploiting the advantages of NoC in FPGAs for implementing SoC designs is an active area of research. Despite the presence of the hard core blocks in the form of multipliers and processors, the amount of configurable logic is highly limited, when compared to the design sizes that get implemented on FPGAs and hence the logic area is at a premium in FPGAs.

In order to implement an efficient NoC architecture in FPGAs, the area occupied by the network logic should be kept to a minimum. This ensures maximum area utilization by the logic while maintaining the performance of the on-chip network. Also, reduction in the logic blocks used in FPGAs has a direct impact on the power consumption and the timing. Firstly, we achieve area reduction by designing a light weight router for FPGAs. A router is the central component of a NoC and hence one way of achieving low network area

overhead is to have a light-weight router without sacrificing the performance. We design a light-weight router (LiPaR) for NoCs implemented on FPGAs [SBKV05]. LiPaR is a parallel router that can service five simultaneous routing requests at the same time (Figure 3.1). The router uses store-and-forward type of flow control and XY deterministic routing. Two important optimizations are introduced to improve the router design. First important optimization is achieved by performing a simple logical OR (instead of the expensive Multiplexer-based implementation) of the control/select lines, thereby, gaining in area and performance. The fact that at a given instant an input channel will request only one output channel is exploited here. In the second optimization, the inter channel data transfer is made to be governed by the empty status of the FIFO, thereby, removing complex decoding logic. Thus, the empty condition is used to automatically trigger the end of transfer. This optimization significantly reduced the size of the Finite State Machine (FSM) for XY routing. Both the optimizations result in significant reduction of the number of slices used. Also, the performance (frequency of operation) was improved and the power consumption was reduced. The area overhead of LiPaR is only 352 Xilinx Virtex-II Pro FPGA slices (2.57% of XC2VP30).

Then, we propose novel router architecture designs (Multi Local Port Routers and Multi^2 Routers) that provide ample opportunity to optimize the data traffic, thereby achieving improvement in both the power and the performance. This is primarily because of the reduction in the number of packets flowing in the main Networks-on-Chip mesh. Also, in this research work, we present efficient NoC configuration generation strategies.

Having achieved the objective of a light-weight and efficient router [SBKV05], we reduce the network area overhead by reducing the number of routers. But, if n cores are to be mapped, this strategy cannot be applied effectively, unless the cores are constrained to share the same Network Interface (NI). But, this will only result in deterioration the performance of the overall NoC system, due to bandwidth limitations. Hence, we propose an innovative architectural modification to the single local port router (LiPaR) to handle multiple logic cores, at the same time, without any performance or clock penalty. We implement an improved Multi Local Port Router (MLPR) design having variable number of local ports (Figure 1.8) [SV06b]. Advantages of such an approach are plenty, including bandwidth optimization, area reduction, ease of congestion, transit time reduction and power reduction. We observe a 36% average area savings (maximum of 47.5%) on XC2VP30 FPGA and a 30% average performance gain by using MLPRS in place of traditional single Local Port version.

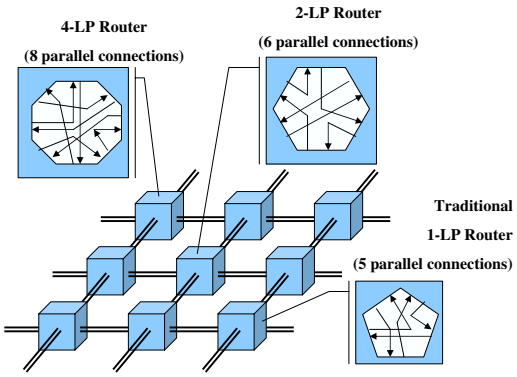


Figure 1.8: Multi Local Port Router based NoC mesh

Mapping of cores onto such a non-traditional NoC architecture is a complex task and hence we present a proof-of-concept algorithm (optiMap) for optimally mapping the cores [SV06b]. OptiMap is an exhaustive search algorithm and finds the optimum NoC architecture (& hence the formal proof is redundant). For the given system task graph and the set of constraints & objectives, the algorithm finds the optimal number of routers, configuration of each router, optimal mesh topology and the final mapping. The cost function includes the overall execution time (taking queuing due to congestion in to consideration). The algorithm is tested on a wide variety of benchmarks.

Because of the exhaustive search nature, optiMap requires large execution time and memory (for generating all possible configurations). Thus, scalability issues make the use of optiMap infeasible for larger systems. Hence, based on the insight and results from optiMap algorithm, we develop a heuristic based mapping algorithm (cMap) [SV07a]. It is a force-directed mapping strategy, which starts from the smallest possible mesh size and expands along all four directions as the cost gets improved. In this approach, there is no expensive configuration generation step. Also, the execution time is reduced to a couple of seconds compared to the optiMap that took several hours.

Unlike the bus-based systems, the NoCs are handicapped in terms of versatile features like broadcast (present in bus). This is due to the pair-wise data packet transfer that happen in the traditional NoC having one core per router, thus, offering no scope for multicasting. In contrast, all the ports of an MLPR have the same mesh XY co-ordinate, thus, providing an excellent opportunity to multicast to all the cores attached to the same MLPR. Exploiting this fact, we present a novel & efficient MLPR design (Multi^2 Router) that is capable of multicasting packets to multiple nodes, without any performance penalty [SV06a]. We modify the decoding logic to send the data packet only once, if the receiving cores are

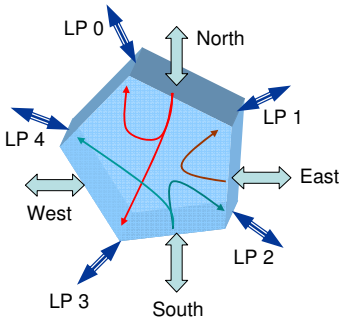


Figure 1.9: Multicasting Illustration in a 5 port Multi^2 Router

mapped to the same Multi Local Port Router (Figure 1.9). In addition to the performance gain, this approach significantly decreases the data traffic in the NoC backbone and provide ample opportunity to reduce the power consumption considerably. In this context, NoC system design can be used in conjunction with the tiled execution of cores in FPGAs [SKV04]. Utilizing the multicast capability, we present an energy-efficient NoC configuration generation approach (μMap), targeting data packet traffic reduction in the network [SV07b].

The NoC architecture generation algorithms that optimized different metrics including latency and power consumption favor clustering of cores aimed to fewer number of routers forming the NoC backbone. All the optimizations primarily target reduction in overall packet count, which in turn results in lesser time and effort spent towards the communication (packet switching) across the on-chip network. This scenario is achieved by way of reduction in the number of routers (the best case being a single router), thereby, resulting in fewer links forming the NoC topology. This presents a scenario of having routers with increased port count. We attempt to identify the shortcomings in the use of large multi port router from a power angle. The experiments show that when using large MLPRs, the average power of the router increases dramatically, eventually diminishing the gains made through overall packet reduction. In addition to the average power increase, larger port count create several IR drop violations because of concentration of large switching activity across the crossbar of the router [SV07c].

Architecture generation and core mapping assume a static level of data flow between the nodes of the task graphs and hence, an optimal configuration is generated for the static data traffic pattern at hand. But, the modern SoCs are a heterogenous composition of cores, wherein traffic is generated between various cores at runtime, thus, making the communication dynamic in nature. Taking these variations in the inter-communication patterns and bandwidth requirements into account, we present technique to estimate the

Minimum BandWidth Guarantee (MBWG) required for a given topology. We show that calculation of MBWG is essential in avoiding bandwidth violations across the various links due the varying communication patterns. In the end, we elaborate on the applicability of an Networks-on-Chip based system when extended to a Multi-FPGA based system design. We identify the commonality in the ideas on an Multi-FPGA approach and also identify the key issues to be tackled in order to seamlessly integrate multiple FPGAs. We leave the issue open-ended while detailing the necessary architectural modifications and the changes to the topology generation algorithms.

To summarize, the dissertation offers novelties in terms of the router architecture as well as the topology generation and mapping strategies. Exploiting the inherent capabilities and features of a multi port (multi cast) router, we present a set of algorithms targeting various optimization metrics. Through extensive experimentation, we observe the possibility of degradation in the power efficiency and creation of bandwidth violations, when increasing the port count of a router. The proposed novel architectural methodologies will serve as a step forward in the pursuit of an efficient Networks-on-Chip design, in terms of area, performance and power.

1.9 Organization of the Dissertation

The rest of the dissertation is organized as follows:

Chapter 2 presents a short survey detailing the terminology and approaches involved in a Networks-on-Chip design. The readers who are familiar with Networks-on-Chip design can skip this chapter to the following chapter.

Chapter 3 describes our initial effort in the pursuit of an FPGA-based Networks-on-Chip. We propose an architecture design for a light weight router that can establish five parallel connections simultaneously. The router occupies 2.57% of the Xilinx Virtex II Pro (XC2VP30) FPGA and 0.4% in the Virtex 4 FPGA. The resource usage provides more area for the user application logic.

Chapter 4 proposes the novel design architecture for the Networks-on-Chip design. The key idea is to have have a router with more than one local port, giving rise to the *Multi Local Port Routers (MLPR)*. An MLPR is a marriage between router-based interconnection networks and the switch-based interconnection networks. The proposed approach provides many-fold advantages in terms of area, power, transit time, congestion and optimal mesh

design. Also, the design issues involved in the use of Multi Local Port Routers are discussed in detail.

Chapter 5 presents the details of the experimental platform followed throughout the thesis. We develop a set of eighteen synthetic benchmarks for experimenting with the exhaustive-search based algorithms presented in later chapters. The need for synthetic benchmarks is because of the scalability issues faced in doing an exhaustive search of the configuration space.

Chapter 6 describes the structured approach to perform an exhaustive search for finding the optimal NoC mesh having Multi Local Port Routers. The steps of the optiMap algorithm are discussed in detail. The results clearly indicate the performance gains obtained by the novel MLPRs.

Chapter 7 present a heuristic-based fast mapping algorithm (cMap) for finding the near-optimal NoC mesh in a fast manner. The algorithm scales nicely with the increased number of cores in the task graph and finds the NoC configuration within a couple of seconds.

Chapter 8 extends and exploits the features of the Multi Local Port Router by introducing the capability of *Multicasting*. We are motivated to introduce this modification by the presence of high level of out-degree prevalent in the system task graphs. We discuss the necessary architectural modifications in detail. The proposed design bridges the gap between the shared-medium and router-based networks in terms of versatile feature availability.

Chapter 9 presents the design methodology to realize an energy-efficient Networks-on-Chip. The optiMap algorithm is modified taking into account the multicast capability. We observe a drastic reduction in the total number of packets, leading to the ease in congestion in the data traffic. Moreover, the reduction in the packet hops translated to significant improvement in terms of the performance and power consumption.

Chapter 10 presents the elaborate analysis in terms of the power efficiency in the multi port routers. We experiment and present the intra-port power savings possible, when choosing a particular port for mapping in a multi port router. Also, in spite of the improved performance and area requirements, it is observed that larger multi port routers may not be ideal when targeting a reliable (reduced IR drop violations) and low power Networks-on-Chip.

Chapter 11 describes the need for handling the dynamic nature of task graphs present in the modern System-on-Chips. We point out the need to consider the bandwidth variation along various links in order to prevent violations, during the phases of topology generation and mapping in an NoC. Towards this end, we present a heuristic algorithm (which is an extended and modified version of cMap) to estimate the Minimum BandWidth Guarantee (MBWG) required for a particular NoC architecture, thereby, helping the designers to better predict the violations that may arise because of dynamic inter-communication.

Chapter 12 proposes design modifications necessary for seamless integration of cores that are spread across multiple FPGAs. We discuss the features available in latest Xilinx Virtex 5 FPGAs that help to realize efficient inter-FPGA transfers, optimizing performance, power and effective bandwidth. The actual implementation details are suggested as a possible future direction of work.

Chapter 13 summarizes the contributions of the dissertation, throwing highlight on the novelty and usefulness of the various router designs and algorithms presented in this dissertation. A balance between the RTL design and the mapping algorithms optimizing various metrics, coupled with the critical analysis on the pros and cons of the multi port routers, provide well-rounded coverage to the dissertation topic.

Chapter 14 presents a discussion about the possible directions in terms of the future work. It is shown that there is enough scope for improvement both in the design of routers as well as improved Networks-on-Chip architecture generation algorithms.

Chapter 2

Networks-on-Chip Background

Traditionally, in a System-on-Chip, interconnection between different Intellectual Property cores is achieved by means of a shared-bus. In the GigaBits era of System-on-Chip design, with increasing communication demands and constraints, interconnection of cores using shared-bus architectures present communication bottlenecks [Sem06], affecting the overall system performance. Also, they do not present a scalable solution to existing problems in the communication [Axe03].

2.1 Networks-on-Chip

On-chip point-to-point distributed interconnection networks or Networks-on-Chip (NoC) have been proposed as a new design paradigm to solve the communication bottlenecks in modern day System-on-Chip designs [BdM02, DT01]. Unlike the shared-bus approach, the key idea in an NoC design is to implement interconnection of various Intellectual Property (IP) cores using on-chip packet-switched networks [KJS+02].

The concept of on-chip interconnection networks is actually borrowed from the well established field of computer networks, where the data flow takes place in terms of packets between the various nodes (computers). The interconnection between the computers are effected by means of router nodes. The source node or computer packs the data to to communicated into large chunks of packets and addresses it to the destination computer. The router has all the necessary intelligence and information to route the data packets properly. These are also referred as *macro networks*.

Using the same analogy, the on-chip interconnection networks or the *micro-networks*

realize the communication between the various communicating modules by initiating packet transfer through the switching elements or routers. In spite of the similarity, there are key differences between the micro and the macro networks. They can be summarized as follows,

- Unlike the macro-networks, the micro-networks are greatly resource-limited. The silicon real estate is costly and hence the associated area overheads cannot be ignored. Also, they do not have the comfort of large buffer size and routing tables (like a macro-network) for the network backbone. But, the wires available to interconnect modules are abundant in an micro-network [DT01, VDC03].
- The micro-network design involves a number of parameters like performance, power, least transit time, signal integrity, etc. This is in contrast to macro-networks where the goal is to achieve reliable packet delivery with guaranteed performance and hence the means of achieving them is assumed to be available without any resource restriction.

- Further, the data communication in an NoC is expected to be lossless. Hence, the Networks-on-Chip needs to have both Guaranteed Throughput (GT) and Best Effort (BE) for the data traffic involved.
- The micro-network design is highly application-specific, against the macro-networks which are generic in nature.
- The micro-network topology is static as they are the networks implemented and fabricated on silicon.

2.1.1 Components of a micro-network

An on-chip interconnection network is made of the following components that are interconnected to form a System-on-Chip.

- Routers
- Network Interface (NI)
- Design Core or Processing Element (PE)

Routers

Routers are the switching elements that are responsible for forwarding of data packets such that they reach the destination. In the traditional mesh-based NoC, a router has four directional ports (North, East, West and South) and a local port to which the design cores is attached. For communication topology other than the regular mesh, the number of directional ports can vary. The source design core forms a data packet which is addressed to the router to which the destination core is attached. The data packet is made to hop across the routers using proper routing mechanisms [GG02].

Network Interface

The Network Interface (NI) is a wrapper or interface between the router and the design core. It supports communication along two directions, performing two distinct set of operations. First, it collects the data from the design core, packetizes, adds the header (having the destination address) & an optional tail information, and pushes the packet into the attached router, thereby, inserting the packet into the network. Second, it receives the packet from the attached router (when the packet is actually addressed to it),

Design Core

The design core can be any heterogeneous core including a microprocessor, DSP core, microcontroller, memory, input/output device controllers. These are often referred as the Processing Element (PE). The design cores are totally abstracted from the actual process of communication that is happening at the System-on-Chip. This property promotes independent design of the design cores as Intellectual Property (IP) cores that are integrated to support a platform-based system functionality [HJK+00, Axe03].

2.2 Summary of the Benefits & the Issues Involved

2.2.1 Advantages

The advantages offered by a Networks-on-Chip communication architecture are summarized as follows,

- **Reusability:**

As discussed earlier, the Processing Elements are completely abstracted from the fine details of the communication mechanisms. The sending and receiving of data with reasonable quality of service is ensured by the communication backbone comprising of the routers. The designers are offloaded the burden of initiating different communication that is necessary for the system operation. Moreover, the Processing Elements, by themselves, are reusable in nature, provided they conform to a common interface and synchronization mechanisms with the router network. Also, the switching elements are generic in nature and the communication network can be used with any conforming design core. Hence, NoC greatly promotes design reuse and platform-based design that is necessary for the future system-on-chip.

- **Scalability:**

  Networks-on-Chip communication backbone is made of point-to-point communication links that are basically distributed and independent in nature. New design cores are added into the network along with a dedicated router having a unique address or coordinate in the network. Now, the communication is just hopping of packets from the source router to the neighboring router. There is no central arbitration mechanism of the communication backbone. This results in a communication architecture whose performance is not constrained or degraded by the adding of Processing Elements. This is the essential characteristic of a scalable and modular architecture [HJK+00, Axe03].

- **Predictability:**

  The architecture is highly modular and has regular geometry. Except for the clock, power and ground wires, there are no global interconnects that span across the chip. This leads to better electrical and physical properties of the network structure. Accurate analysis and modelling of the system is possible, overcoming many of the uncertainties that exist in the nanometer regime. Predictability is essential for reusing the NoC platform for supporting various System-on-Chip applications.

- **Heterogenous System:**

  If we view the arguments presented in reusability from a different angle, an NoC system supports a heterogeneous system. Each Processing Element or the design core is isolated from the rest of the System-on-Chip. Hence, it is great possible to realize the individual components that can operate at varied frequencies and voltage levels (for power-performance tradeoffs). This property will benefit and support the idea of Globally Asynchronous Locally Synchronous (GALS) style of system design

[Cha84].

- **Parallel Communication & Aggregated Bandwidth:**

  Since, the arbitration of the data flow is not centrally managed, but actually distributed among the network routers, an NoC structure is inherently parallel in terms of operation. For example, in a $n \times n$ 2D mesh NoC, the total number of parallel communications that can happen is $\frac{n \times n}{2}$. This increases with different network topologies. This simultaneous data traffic results in an aggregated bandwidth that is capable of satisfying the communication requirements of the future System-on-Chip.

2.2.2 Issues Involved

There are few issues that are to be contended with, when using a Networks-on-Chip.

- **Area Overhead:**

  The additional of logic in terms of routers and the network interface adds to the area requirements, which is quite significant. With more complex and advanced routing techniques, the area of the router network increases. Any addition in logic directly results in increased power consumption.

- **Transit Time:**

  Data communication in terms of packet hops across the network routers takes considerable number of clock cycles to accomplish data transfer. The number of hops varies depending on the topology and the mapping of cores, which needs to be highly efficient to optimize for performance.

- **Network Contention:**

  Though the data transfer is distributed in nature, multiple data packets can contend for a common link in the NoC topology. With increased data traffic, the network contention problem only compounds. If the network design is inefficient and the data traffic is not properly managed, in the worst case, the on-chip network can suffer from problems including deadlocks and livelocks.

- **Sub-optimal Resource Usage:** In spite of the parallel communication and aggregated bandwidth available, a proper mapping and design is necessary to exploit the gains that is available. Hence, the Networks-on-Chip design tends to be highly application-specific in nature in order to have high network resource utilization.

- **Additional Design Effort:**

  The communication architecture, because of its application-specific nature, requires additional design effort and time, compared to the well-established communication backplanes like AMBA, STBus and Silicon Backplane. Additional wrappers are necessary to make the existing shared-bus based IP cores to resolve the compatibility issues. All these factors require re-education of NoC design concepts.

2.3 Description of a micro-network (Networks-on-Chip)

An On-chip interconnection network is described in terms of the interconnection network topology, switching mechanism, flow control, routing, queuing (buffering) and scheduling [DYN98, DT04].

2.3.1 Network Topology

Network topology is the arrangement and type of interconnection of the nodes in the network. In simple words, it defines the various channels that are available for the data transfer across the network. The topology is described by various parameters like node degree, maximum hop distance, network diameter, channel width, bisection density, and node distance. The various network topologies include 2D mesh, 2D torus, folded torus, hypercube and fat-tree [DT04].

Each topology has its own pros and cons. In a fat tree, the complexity of the router increases as we proceed down the levels and the irregular structure makes it layout incompatible. A 2D mesh is highly regular, but, there exists high traffic density near the center of the mesh compared to the periphery and this creates problems like hot-spot creation and network contention. But, overall it is highly energy-efficient [BJM+05]. The other topologies like hypercubes and k-ary n-cubes suffer and become impractical for a 2D layout realization.

2.3.2 Switching Mechanism

Switching denotes the mechanism of moving data from a source to a destination node. Circuit switching and packet switching form the two extremes of switching mechanisms [DT04].

Circuit Switching

In circuit switching, for every communication between a pair of nodes, a connection is made which stays active for a fixed time. During this process, the channels remain occupied for the entire duration and no other communication can take place along the occupied channels. Also, all the packets reach the destination in order and hence results in the reduction of the complexity associated with the re-ordering of the packets at the destination node. This kind of system has low delay and guaranteed bandwidths, but, suffers from channel under-utilization, inefficient usage of bandwidth available and long time to setup connections.

Packet Switching

In packet switching, data communication takes place in form of packets of variable length. The packets have control information built into them in the form of header along with data to be transported. Here, the packet transfer happens between the cooperating routers, with no channel reservation along the path and independent routing decisions being made along the entire path. Hence, the overall packet transfers happen asynchronously over the network. There exists a possibility of the packets arriving out of order (depending on the routing mechanisms) and hence may require extra effort to re-order them at the destination. Packet switching provides the best effort service by efficient utilization of the resources. The issues involved in this type of switching are congestion, increased transit time of the data packets and the non-guaranteed bandwidth.

In short, achieving both Guaranteed Throughput (GT) and Best Effort (BE) traffic is highly application-specific and requires considerable design effort to achieve optimal results. Flow control helps to achieve a better GT/BE tradeoff.

2.3.3 Flow Control Mechanism

Flow control deals with the allocation of channel and buffers for the data packets as it travels from source to destination. Though, there are bufferless flow control and circuit-switching variants, the buffered type of flow control are observed to be very efficient [DT04]. The popular buffered flow control mechanisms are [Moh98] listed as follows. The former two are packet-buffer flow control, while the latter two are flit-buffer flow control.

- Store and forward
- Cut-Through
- Wormhole
- Virtual Channel

Store and Forward flow control

Store and forward is the simplest flow control mechanism. Here, each node along the path receives the packet in its entirety, stores (buffers) it locally and forwards it to the neighboring router along the path. Although there are certain buffer requirements depending on specific applications, store and forward does not reserve channels, and hence it does not lead to idle physical channels [KS03]. The major drawback of the Store-and-Forward flow control is the high latency that is proportional to packet size. The overall latency of a packet is $T_0 = H(t_r + \frac{L}{b})$, where H is the channel cycle time, b is the channel width and L is the message length and t_r is the cycle latency for a flit.

Cut-through flow control

Cut-through flow control is a slight variant of the store-and-forward flow control, that overcomes the latency penalty by forwarding the packet as soon as the header is received and resources (buffer and channel) are acquired, without waiting for the entire packet to be received [DT04]. Hence, the latency reduces to $T_0 = H.t_r + \frac{L}{b}$

Worm-hole flow control

Worm-hole flow control is similar to the cut-through method, but here the channel and buffer allocation is done on a flit-basis rather than packet-basis. Here, the packet is divided into basic units called the flow control digits (flits). The header flit acquires the virtual channel and buffer. The data portion or the body of the packet simply follow the virtual channel already acquired by the header flit. Thus, The packets flow in a pipelined fashion. The tail flit releases the acquired virtual channel. Wormhole routing demands increased decoding logic in each router [Moh98]. Also, there is a great chance of blocking in the network as the packet reserves the channel along the path, which are prevented from being used by (other) following data packets. If the leading packet is blocked for some reason, it blocks the path and this cascades to the successive packets.

For the set of applications we experimented with, we found that with heavy random blockages due to shared paths, the performance of the wormhole routing is comparable to the store-and-forward case.

Virtual Channel

Virtual channel overcomes the problem of blockages associated with the worm-hole flow control. Here, there are several virtual channels (channel + flit buffers) for each physical channel. In simple words, there are multiple paths available along each of the physical channel. Hence, even if a particular virtual channel is blocked, alternate paths are available for the following packets to flow without any blocks occurring. Unlike the worm-hole flow control, there is no guaranteed bandwidth, as there exists a competition between several virtual channels in acquiring the access of a particular link. Virtual channel flow control can be costly in terms of the buffer numbers and the associated decoding & arbitration logic.

2.3.4 Routing Mechanism

Routing refers to the decisions that are taken to forward the packets along appropriate directions. The underlying requirement here is that every node in the network is capable of communicating or sending data packets every other node in the network. Efficient routing techniques are necessary as they not only affect the transit time but also after the power consumption and the congestion in the network.

Source vs Distributed Routing

Based on the routing decision taken, there are two distinct types of routing.

- **Source Routing:**

  Here, the path to be taken is decided by the source node, the information about which is made part of the packet. The path of the packet is hence statically allocated.

- **Distributed Routing:**

  This is also referred as *Table-Based routing*. In this approach, the routing decision are take by the individual routers depending on various parameters. In order to maintain the performance, the routing algorithm needs to take the decision in a very fast

manner. In addition, there are overheads associated with storing the global route information at every node.

Deterministic vs Oblivious vs Adaptive Routing

The following is the classification based on the adaptability of the routing decision. The mode of routing can be deterministic or adaptive or oblivious.

- **Deterministic Routing:**

  Deterministic routing follows a fixed path based on the source and destination locations, but the decisions are independent. For example, in deterministic XY routing, starting from the source node, the packet travels along the X direction (left or right) until the packet reaches the column same as that of the destination node. Next, it proceeds along the Y direction (top or bottom) till reaches the destination router node.

- **Oblivious Routing:**

  This is a variant of deterministic routing. In short words, the packet is first routed to a random node in the network, from where it is routed to the destination node. Here, the packets are routed with no regard to the network condition and represents a tradeoff between locality and load balance.

- **Adaptive Routing:**

  In this type of routing, the path taken is decided dynamically based on the network conditions including congestion, wait time, and presence of faulty channels along the path. Because of this, the path taken can be *non-minimal*. Adaptive routing is more flexible, but requires global knowledge of the system for making dynamic routing decisions. Also, the packets can arrive out of order and hence huge buffering space at the receiver end is required for reassembly.

Minimal vs Non-Minimal Routing

This is a redundant classification based on the actual path chosen.

- **Minimal Routing:**

  The actual path taken is the shortest between the source and the destination nodes in terms of the number of hops involved.

- **Non-Minimal Routing:**

  Here, the path is not minimal always and this is a direct consequence of the adaptive type of routing. If not properly managed, this could result in live-locks in the network.

Critical problems: Deadlocks & Livelocks

- **Deadlock**

  This represents the network state when packets are blocked because a leading packet is waiting indefinitely for an event to happen, which in turn is blocked because of another wait happening in the network. In short words, it is a *cyclic-wait* state occurring in the network.

- **Livelock**

  This is the result of a non-minimal and adaptive type of routing mechanisms. Here the packets are made to flow (take different paths) based on the network conditions. The data packet keeps flowing in the network actively, *without reaching the destination*.

- **Indefinite Postponement**

  This is caused due to the fairness criteria. Here the packet waits for an event to happen, which never does [KS03].

Some of the popular methods of routing that guarantee dead-lock and live-lock free routing include XY, West-First, North-Last, and Negative-First routing [KS03]. XY-routing is more popular due to its simplicity, low area overhead and *guaranteed minimal route*.

2.3.5 Buffering or Queuing

Queuing refers to the strategy of storing the packets received in case of output contention. Different queuing strategies include output buffering, input buffering or both, and central buffering. Usually, the type of buffering is a tradeoff between the latency, area and the performance.

We use both input and output buffering, in LiPaR router (Chapter 3), to decrease congestion. This is later converted to input buffering in the Multi^2 Router (Chapter 8).

2.3.6 Scheduling

Scheduling refers to the arbitration algorithm used to service the requests. The types include static and dynamic.

Static

In static scheduling, a fixed time is reserved for each input to communicate with output. It has predictable latency and fixed bandwidth, but, it is inflexible.

Dynamic

Dynamic scheduling makes the arbitration decision at run-time and is more flexible in the service of requests. Dynamic scheduling makes the decision at run-time. Invariably there is request-grant kind of service mechanism in this type of scheduling. Dynamic scheduling is a of fair scheduling scheme, with high flexibility, but with variable bandwidths and latency.

Scheduling algorithms include Least Recently Used (LRU), Maximum size Matching, Maximum Weight Matching, Parallel Iterative Matching (PIM) and Iterative Round Robin with Slip (SLIP) [MA98].

Detailed information on the various parameters related to a Networks-on-Chip is available in [DT04, DYN98, KS03, Axe03].

2.4 Design of a Networks-on-Chip

2.4.1 Design Flow

A typical Networks-on-Chip design flow comprises of three main phases as follows [Axe03].

- Network Backbone Design
- Communication Architecture Design
- Application Mapping Phase

Network Backbone Design

This phase is basically the infrastructure design stage, wherein various generic architectural decisions are made. This is the preliminary step wherein the infrastructure of the on-chip communication network desired is sketched in terms of the network topology, the switches & buffers required, the channels of communication & related protocols and the network interfaces. This is one of the critical phase as the decisions directly affect the cost and the eventual performance of the system. Though regular grid-based topologies are preferred for their simplicity and controlled electrical parameters, at times, custom topology generation is desired because of the varying Processing Element sizes. In this context, floorplanning phase crops into the standard Networks-on-Chip design flow. The width of the channels for effecting data communication greatly affects the bandwidth and the performance of the system. This is because the packet latency in terms of the number flits required is dependent on the channel width chosen. Further, when using the buffered flow control strategies, the choice of the buffer size/depth has impact on many parameters including the area, latency and ultimately on the system performance. The application mapping phase is tightly coupled with buffer sizing problem as the traffic pattern or the workload dictates the optimal buffer size.

Communication Architecture Design

This is the resource allocation phase exclusively catering the workload of the application in hand. The decisions taken during this phases are majorly affected by the various technological and implementational constraints. The dimension of the on-chip communication network and the types & sizes of the resources to be allocated and the contents of the resources are primarily fixed in this phase. All the decisions are dictated by the feasibility estimations based on the characteristics of the application and the resource, and also on the function/architecture mappings [Axe03].

Once the communication architecture is in place, the eventual performance of the on-chip network is affected by the routing (Section 2.3.4), switching (Section 2.3.2) and flow control (Section 2.3.3) methodologies adopted in the routing elements. Adaptiveness in the routing process creates the issue of out-of-order delivery, the re-assembly process of which will degrade the performance of the system. Further, proper router design is necessary to prevent deadlocks and livelocks.

Application Mapping Phase

Even after the choice of NoC communication architecture and the associated routing mechanism, the eventual performance of the overall micro-network is dependant on the application mapping process. In this phase, the application, that is typically represented as an annotated system task graph, is mapped to the resources supported by the Network-on-Chip backbone system. In other words, the different Intellectual Property (IP) cores of a System-on-Chip are assigned to an appropriate router node of the on-chip network in hand. This is a very important phase as the effectiveness of the application mapping will determine the overall performance of the system. The primary goal here is to achieve maximum utilization of the bandwidth that is available across the network, while decreasing the communication latency involved in the packet hops and reducing the congestion/contention problems that is prevalent in a shared network.

Apart from the above mentioned phases, validation and verification of the system needs to be done. The key idea here is to use the existing communication of the Networks-on-Chip to feed test input to the Processing Elements at large. Since, we are looking a platform-based integration of the system, the test methodologies must be sufficiently adaptable for the heterogenous System-on-Chip environment consisting of microprocessors, storage elements and compute/data intensive DSP cores, etc.

2.5 Research in Networks-on-Chip

The purpose of this section is to present a holistic picture of the leading research groups and works in the field of Networks-on-Chip design. The works that are pertinent to the issues addressed are given in the respective chapters.

The issues related to the on-chip communication infrastructures like clock skew, synchronization issues and related power and performance bottlenecks were brought to limelight by the ITRS in its 2000 edition report [Sem06]. The problems with the global interconnects and the shared-bus approach were highlighted by Dally et al [DT01]. They presented arguments discussing the need to adopt a radical approach in the way data communication is effected in a System-on-Chip environment. They floated the idea of routing the packet, in place of the wires [DT01]. Hemani et al initiated the idea and need for research on Networks-on-Chip [HJK+00]. At the same time, a high-level design point of view of the new paradigm of Networks-on-Chip was presented by Benini and De-Micheli, wherein

they delineated the various details involved in an Networks-on-Chip design [BdM02]. As one of the frontrunners, Sashi Kumar/ Axel Jantsch et al presented a Networks-on-Chip architecture and design methodology [KJS+02]. They also present different approaches for application specific mapping in Networks-on-Chip [LK03b, LK03a]. Some of the important works related to the interconnection topologies include SPIN [SPI06, GG02], CLICHE [KJS+02], SoCIN [ZS03] and OCTAGON [KNDR01, KND02]. SPIN has a fat-tree topology, while CLICHE uses a 2D mesh topology. SoCIN uses a 2D torus topology, while the Octagon [KNDR01, KND02] has an an eight router chordal ring interconnection architecture intended to replaced the shared-bus. A honeycomb type of interconnection topology is presented in [HJK+00].

There are a number of Networks-on-Chip prototypes from a number of research groups involved in the NoC domain. These include the Æthereal project from Philips [GDR05, ARG05], aSoC from University of Massachusetts, NoC design from Arteris Corporation [Art06], Nostrum project from Royal Institute of Technology [Pro06] and ×Pipes project [JMBM04].

Nostrum research project [Pro06] aims in developing a Network-on-Chip (NoC) architecture and a corresponding design methodology, focussing on various communication issues from the physical to the application levels. It is a packet-switched network with regular 2D mesh topology and adaptive & flexible routing, and offers best effort and guaranteed latency traffic [MNTJ04].

The SPIN (Scalable Programmable Integrated Network) micro-network is a generic, scalable interconnect architecture for System-on-Chip. Following is a short overview of the SPIN project [SPI06]. A SPIN micro-network consists of three VLSI macro cells, which included a router (routing packets) and two VCI compliant wrappers (interfacing the SPIN network with the subscribers like processors, coprocessors, memories, etc).

The adaptive System-on-Chip (aSoC) research project at University of Massachusetts, presents a single-chip interconnect architecture that provides scalable data transfer along with capability to reconfigure with the change in application-level communication pattern. An important aspect of the architecture is the support for compile-time schedule communication.

The Raw Architecture Workstation (RAW) is a simple, wire-efficient multicore architecture that scales with increasing VLSI gate densities. The Raw architecture's goal is to provide performance that is comparable to that provided by scaling an existing architec-

ture, but that can achieve orders of magnitude more performance for applications in which the compiler can discover and statically schedule fine-grain parallelism. RAW prototype is composed of a set of interconnected tiles, each of which comprises of instruction memory, a switch-instruction memory, data memory, ALU, FPU, registers and a programmable switch.The tiles are interconnected by two static (compile time route specification) and two dynamic (run-time routes) 32 bit fully-duplex on-chip networks. The data transfer is effected by means of regular packet hops [Raw06].

The Silicon Networks (SlicNets) [Sli06] at Carnegie Mellon University aims to provide a communication-centric SOC design and the related support for analysis and optimization of novel on-chip communication architectures. Essentially, the research explores the possibility of having a scalable and flexible communication schemes via the Network-on-Chip (NoC) approach, that includes the entire spectrum of the NoC design (communication infrastructure design, communication paradigm and application mapping).

×pipesCompiler project aims at generating application specific NoC for heterogenous multiprocessor SoCs. A library comprising of parameterizable soft macros of the various building blocks is available to quickly realize NoC designs. The authors show a custom-defined (non-tile) approach to realize Networks-on-Chip architectures on ASICs. The non-tiled approach is justified by the presence of non-uniform sized cores. The Paris and SoCIN project present a parametric and scalable Networks-on-Chip design [ZSS04, ZS03]. Also, the Hermes project and RaSoC router present methodologies for developing efficient soft core routers [ZKS04].

The Æthereal NoC project presents a Networks-on-Chip architecture that provides guaranteed services (such as uncorrupted, lossless, & ordered data delivery, with guaranteed throughput and bounded latency) in order to realize robust SoCs [GDR05]. A commercial implementation of the Æthereal NOC is presented in [SDN+06], wherein the dedicated interconnect of the companion chip of the high-performance Philips PNX8550 (Viper2) SOC, is replaced by an Æthereal NOC [GDR05, ARG05]. The PNX8550 Viper platform is a television System-on-Chip based on the Philips's Nexperia Home platform [Phi03, Phi04, Phi02], that is used by a variety of TV vendors.

Apart from the architectures and design methodologies, there are several simulators which are required to evaluate various network architectures and application mappings to realize optimal performance. Some of the generic simulators include OPNET [OPN06], OMNET++ [Var06], Network Simulator (NS-2) [FV00, ns2] and RSIM [HPRA02]. The

simulators that are targeted towards a Network-on-Chip environment include Orion [WZPM02], NoCSim [Whe06], PoPNet [Sha06], retargettable simulation platform [ZM04] using Liberty Simulation Environment [VVP+02]. The emerging hardware modelling language SystemC [Sys06] and SystemVerilog support both modelling and simulation of the hardware designs, thus providing an ideal co-design environment. A performance analysis model using a over-simplified router of an NoC is presented in [Har05], wherein various parameters are measured by event-based simulation for a poisson distributed random traffic, using queuing-theory. It must be noted that there is comparison with any available bus implementations. Simulators for modelling shared-bus and NoC using the concept of Operation State Machines is presented in the PhD thesis [Zhu05]. The collaborative work of this thesis [Bha06] presents a comparison of single-port and multi-port NoCs with contemporary buses implemented on FPGAs, wherein event-driven simulator is used to determine the performance parameters of both NoC and bus.

The discussion presented so far is by no means a complete research overview of the field of Networks-on-Chip design. In the interest of space, we do not present the specific research papers dealing with router design and routing strategies, testing methodologies and various application-specific synthesis techniques. Research in this particular problem area is still in its nascency, wherein researchers are trying to apply the established test methodologies to suit the Networks-on-Chip design environment and hence, a great deal of work is required in the research area of Networks-on-Chip. The specific research papers that have a direct relation to this thesis are discussed in the respective chapters.

Chapter 3

Light Weight Parallel Router (LiPaR)

Present day technology for ASICs supports Networks-on-Chip designs which can have 100 million gates on a single chip. The latest FPGAs can support only about 10 million gates to accommodate all logic and the associated routing. In order to implement a competitive NoC architecture in FPGAs, the area occupied by the network should be kept to a minimum. This ensures that the maximum area can be utilized by the logic while maintaining the performance of the router network. Reducing area also reduces the power consumption. In this chapter, we present a parallel router which can service five simultaneous routing requests (the maximum possible) with minimum area overhead.

3.1 Related Work

The proposed router is designed to be used in a reconfigurable computing platform. Area limitations of an FPGA based device demand that the router be of small size. Although there are a number of ASIC based router implementations, we restrict our discussion of the related work to FPGA-based implementations only. Marescaux et al. [MBV+02] present the first working implementation of a NoC based system on FPGAs. Their system has scalable 2D-Torus, XY blocking, hop-based and deterministic routing. The packet size has 16 data bits and 3 control bits. It has two virtual channels each operating at 40MHz. Their switch takes 446 slices on a Xilinx Virtex $XCV800$. The main drawback is that it is a 2D torus formed using 1D-routers. This creates a serious bottleneck in the traffic as it has to go through a linear structure of 1D-routers. They also extended their work to a highly scalable network on chip for reconfigurable systems later in [BMN+03].

In [KS04] the authors present an energy-efficient NoC for heterogeneously tiled reconfigurable SoC. Their system was based on the Chameleon System, which is a tiled organization, composed of heterogenous components. FPGA is on of the heterogenous component. It has a 5×5 wormhole router with virtual channels along each direction. There is a central arbiter which establishes necessary connections between various channels. The bottleneck occurs at the central arbiter, which follows a round-robin approach of service. Theoretically, there are five possible parallel connections out of the total of twenty five combinations in a five port router. If the arbitration scheme is central, it leads to performance loss because of queuing of the requests from all ports at one central place. Also, their switch uses 1832 slices in a Virtex-II FPGA [Xil06a], which is more than 10% of the total FPGA area. This large area usage results in two drawbacks: First, it is not energy efficient and second, it will put severe constraints on the total available area for the user logic.

Zerferino et al. [ZKS04] present a soft-core router called RASoC for NoC. The VHDL core of RASoC is synthesized in the Altera family of FPGAs [Alt07] using the Embedded Array Blocks (EABs) or the FlipFlops to build the FIFOs. They report that it consumes 486 Logic Cells (slices in a Xilinx family FPGAs [Xil06a]) for a 4-flit buffer having an 8-bit implementation, which is quite high. Each of their input and output channels has four distinct blocks and uses a large area decoding logic. Zerferino et al. [ZSS04] also presented a parameterizable interconnect switch for NoC. They also present a highly modular and parameterizable VHDL core, which is based on SoCIN [ZS03]. Zerferino et al. also present a highly modular and parameterizable VHDL core for switch [ZS03]. They report the area overheads based on Altera FPGAs. We obtain the beta version of their core and synthesize it on Xilinx ML310 platform with a Xilinx Virtex-II Pro FPGA [Xil06a]. We observe that this implementation consumes approximately 11% of the target FPGA area, which is quite high.

As explained earlier, such a high area penalty puts severe constraints on the available area for logic in a practically large systems, and the power consumption also increases. The only comparable work to ours is reported by Moraes et al [MdMM+03]. Some drawbacks of their work are the following. First, their packet implementation has two header flits which is quite expensive. Although they claim simultaneous connection can be achieved, it cannot happen in parallel, as all the requests go to a central arbitration logic and a central XY-Routing logic, which oversees the establishment of connections. With increased amount of traffic and sequential service of the requests by the arbitration logic (and associ-

ated XY routing), the performance of the system may be severely degraded.

Also, the buffers are present only at the input channels. If the router wants to send the data out, then receiving buffer is available only at the input of neighboring router. So, if that receiving router's input buffer is blocked/occupied, then the current router's input channel is also blocked. If there is an output buffer present, the input channel can empty the contents into the corresponding output buffer, thereby, making the input channel available to communicate with the other output channels, without any blocks. The absence of output buffer creates a handicap in terms of the performance of the router. So, such an optimization is not profitable, as it hampers performance. Also, their arbitration logic is quite large and goes through several states plus there is a four clock cycle delay before a new routing request can be entertained. This also affects the performance of the router. Their router has a counter running at each of the channels to keep track of the packet length, further aggravating the problem. They report consuming 316 Virtex-II FPGA slices for a router with flit width of 8 bits and buffer size of 8. Projecting the number for a buffer depth of 16 at each input channel would increase the number of slices. Our paper removes most of the handicaps cited above with a low area overhead and guaranteed performance without any deadlocks or livelocks.

3.2 Router Architecture

Area is at a premium on an FPGA and therefore, the communication network should be as small as possible. Hence, a router, which is a central component of any NoC, must also be small. In this Chapter, we present a light-weight router for an NoC implemented on FPGAs, which can support five parallel connections simultaneously.

A router or a switch is an key component in any NoC based design. A router has a set of ports, namely, Local(L), North(N), East(E), South(S) and West(W), to communicate with the local logic element and the neighboring routers. It receives the incoming packets and forwards them to the appropriate port. Buffers are present at various ports to store the packets temporarily. A control logic is be present to take routing decisions and arbitration decisions. The router uses store-and-forward type of flow control and XY deterministic routing. We implement two important optimizations: we reduce the size of the Finite State Machine (FSM) for XY routing and we perform a simple logical OR of the *Select/Gnt* lines, which significantly reduces the number of slices. The area savings have significant impact

on the performance and the power consumption of the router.

In this work, we design a light weight, energy efficient, parallel router. The motivation is to reduce the area which also reduces the power consumed. We choose one of the popular methods of buffering called store and forward. The motivation behind choosing such a scheme is to have the simplest possible decoding logic, thereby, reducing both area and power. Establishment of connections is made automatically without any complex decoding logic. The router switches with a set of inter-communicating ports, define the physical layer of the NoC system. There are two types of ports to establish communication, namely, the Input and output port. Communication is done by use of two handshake signals (*Req/Ack*) between the co-operating ports (output and input). This forms the data link layer. Dynamic establishment of connections and routing of packets constitutes the network layer. Here, the cross-point matrix is a very important component, the controls of which are maintained by the output channels. The transport layer is taken care by the Network Interface of a NoC system. Inside the router, the *Gnt/Ack* signals are used to access the FIFO, without the need of any explicit signals. The empty status signal of the FIFO is used to indicate the end of communication. A detailed explanation is given below.

3.2.1 Packet Description

Packet specification is very simple in our router. In this work, we have used the FIFO available in Xilinx LogiCORE [Xil06a]. The depth of FIFO is 16. Since we have a store-and-forward scheme, it makes sense to have a larger buffer size. The flit size is fixed at 8 bits. The first flit is always the header having the coordinates of the destination router. With 8 available bits, we can support a maximum of 16×16 NOC system. The flit size has to be increased if we want to build a bigger system. With the available FPGAs, building a 16×16 system is itself impractical because the NoC system would occupy more than 50% of the total FPGA area (even if a single router takes only 0.2% of the total FPGA area) and there will be less area available for the user logic.

In this work, we fix the number of X and Y bits at 2 each. Hence, it can support a maximum of 4×4 NOC system. The remaining 4 bits are reserved to implement High Level Protocols (HLP). We are building an advanced router prototype, incorporating HLP, as a part of our future work. There is no trailer flit and hence the maximum data size is 120 bits per packet (for the FIFO of depth of 16), which in practical terms, would suffice communication between cores. If there is a requirement for bigger packet size, we can

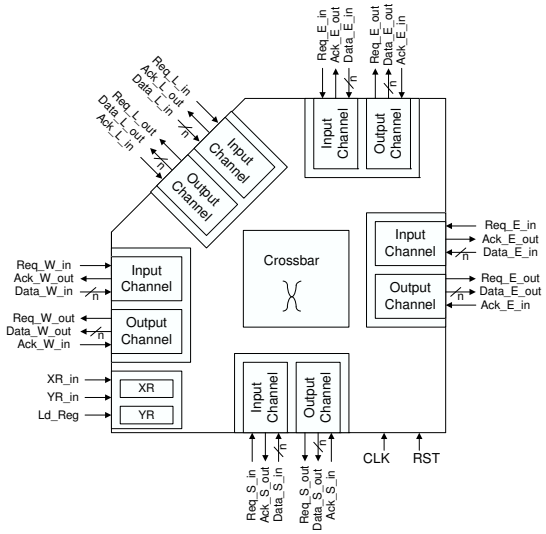


Figure 3.1: LiPaR - Router Architecture

easily build one by increasing the FIFO buffer provided by Xilinx LogiCORE [Xil06a].

3.2.2 Implementation of the Router

Figure 3.1 shows the block-level diagram of the proposed router. The router has three main blocks, namely the Input Channel, Crosspoint Matrix and Output Channels (Figures 3.2, 3.3 and 3.4). The input and output channels are the two unidirectional channels of communication to buffer the data when sending and receiving the data, respectively. The crossbar switch is a full crossbar that can establish connection between the any input-output channel combination by setting the select/control signals of the Multiplexers (`MUX`s) and Demultiplexers (`DEMUX`s) that form the crossbar. More than these components, we have two registers inside every router to store the X and Y coordinate information, which is used to identify the router in an NoC mesh.

Input channel

There is one input channel at each port, each running its own control logic. Each Input Channel has a FIFO of depth 16 and data width of 8 bits and a Control Logic which has been implemented as a FSM. The input channel (refer 3.2) accepts request from other neighboring router. On receiving the request, if it is free, it will acknowledge the request. The first flit is the header and following flits constitute the data. It will accept data as long

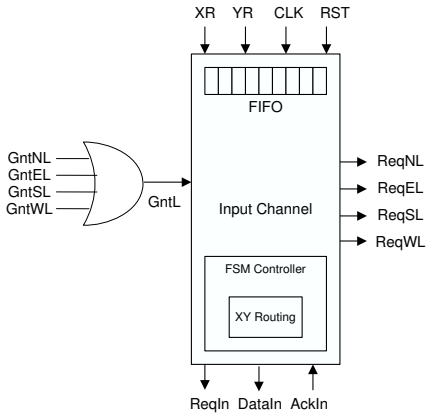


Figure 3.2: Input Channel of the proposed router

as the request signal is held high. The previous router's output channel ensures that the request line is held high until it empties the packet of data, being accepted by the input channel. The input channel sets the acknowledge line high, as long as there is a transfer taking place (indicated by request line). When the transfer is complete, the request and acknowledge lines go low in sequence. The packet of data received from the previous router is stored locally in the FIFO thereby implementing a store-and-forward dataflow. Next the control logic reads the header of the packet and using XY Routing (described later) decides which output channel is to be requested for sending the packet out of the router and sends the request to that output channel. It is to be noted that each of the input channel is running an independent FSM and hence can initiate five possible parallel connections at the same time. Once the input channel gets a grant from the requested output channel, the control bits of the crossbar switch are set appropriately by the granting output channel. An important optimization is done here by performing a simple logical OR (instead of the costly Multiplexer-based implementation) of the *Gnt* and `DEMUX` select lines, thereby, gaining in area and performance. The fact that at a time only one output channel will be requested by an input channel is exploited here. The inter channel data transfer is also governed by the empty status of the FIFO, thereby, removing complex decoding logic. Empty condition automatically triggers the next transfer. The pseudo code of the FSM at the output channel is given in Algorithm 1.

Crossbar switch

The Crossbar switch (refer 3.3) is a made of a set of Multiplexers (`MUX`) and De-multiplexers (`DEMUX`) having an interconnection allowing all possible connection between

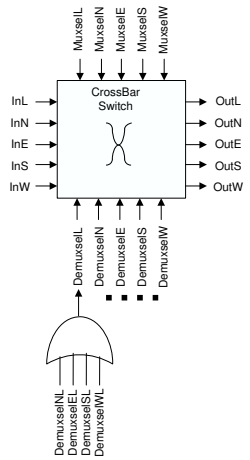


Figure 3.3: Crossbar switch of the router

the 5 input and 5 output channels. The output channel while granting the request of an input channel configures the MUXs and DEMUXs of the cooperating input and output channels thereby establishing the connection between them for the transfer of the packet.

The select signals are driven by the output channel FSM to setup the link with there respective input channel which has requested the transfer. The MUXs are placed at the input channel end of the crossbar to enable them to send the data to any of the output channels. But in cases of the north and south port input channels, connections to the east and west port output channels are not required as we are using XY type of routing. Hence optimized MUXs with fewer combinations were used. At the output channel side the DEMUXs are placed to enable them to receive data from any of the input channels. Again since we are using XY type of routing the DEMUXs towards the east and west output channels can be optimized similarly. Here also since at a time there can be communication between one input and one output channel only we can use logical OR function to implement the select signals of the crossbar MUXs and DEMUXs to save area. Once the communication has been completed the output channel releases the MUXs and DEMUXs by resetting their select signals.

Output channel

There is one output channel (refer 3.4) at each port which has an 8-bit FIFO of depth 16 and an control logic (FSM) making arbitration decisions. The output channel gets requests from the different input channels and and grants one using Round Robin Arbiter

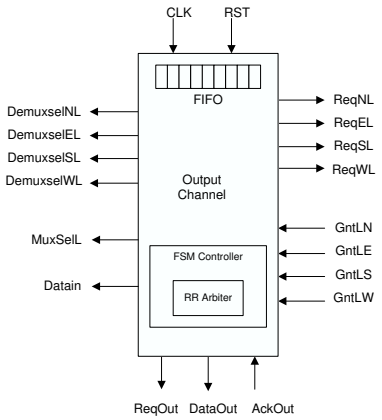


Figure 3.4: Output Channel of the router

(RRA) (explained in following paragraphs) and sets the control bit lines of crossbar switch. It accepts the packet into its FIFO as long as the sending input FIFO is not empty thereby providing a simple decoding logic. When transfer is complete the crossbar switch controls are reset. FSM then initiates the process to send the data into neighboring router using handshake mechanism. Empty status of its FIFO triggers the next inter-channel transfer. The pseudo code of the FSM at the output channel is given in Algorithm 2.

Two important algorithms are part of the FSM of each router. One is the XY Routing algorithm which is used to route the packets so that they hop and reach the destination router. Second is the Round-Robin Arbiter (RRA) used in the output channels to arbitrate and grant access to the input channel.

3.2.3 XY Routing

In the Input Channel, once the FIFO is filled, the X-coordinate of the destination router(say H_x) is compared with the locally stored X coordinate of the Router first to decide on the horizontal displacement. If $H_x > X$ then the packet is forwarded to the East port of the Router, and if $H_x < X$ then the packet goes out through the West port of the router. If H_x is equal to X then the Y-coordinate of the destination router(H_y) is compared with local Y coordinate of the Router to decide on the vertical displacement. If $H_y > Y$ then the packet is forwarded to the North port and if $H_y < Y$ the packet is forwarded to the South port. When H_y equals Y it indicates that the packet is at the destination router and so the packet is forwarded to the local port.

An important optimization decision has helped save a significant number of slices

in the router. In XY routing, a packet is forwarded horizontally till the target column is reached and is then forwarded vertically to the destination router. This means that there are no request for the East or West output ports by the North or South input ports. This fact is exploited and the FSMs of the mentioned output channels are simplified, as they need not service the mentioned input ports. This translates to significant area saving and reduction in number of clock cycles in servicing requests. This helps in the implementation of a light-weight router, having area overheads at the minimum with acceptable level of performance.

3.2.4 Round-Robin Arbiter (RRA)

Round Robin Arbiter is implemented as FSM at each output channel. RRA arbitrates and decides which input channel is to be given access to that output channel when many channels are requesting the same output. Generally, the output channel must follow a priority based arbitration. If a fixed priority scheme is followed, the same input channel may get access repeatedly. Hence in our arbiter, the priorities of the input ports are changed dynamically taking the last input port serviced into account. The priorities are implemented in a clockwise fashion i.e., if the last input port serviced was North, then during next service, the priorities will be in the order of East, South, West, Local and North. It should be noted that no clock cycles are wasted in our scheme as the grant is issued only if there is a request from corresponding input channel.

Since each input channel has its own XY Routing FSM and each output channel has its own RRA FSM, there is no latency in establishing the connections. This allows five different requests to be granted simultaneously at the same time, when five requests come for different output channels. This provides a significant improvement in the performance of our router. It is to be noted that the router coordinates are stored in two registers inside each of the router, which can be accessed from primary inputs. This facilitates easy reconfiguration of the router coordinates in case of system change, compared to the hard-coded coordinates, where one has to re-synthesize with new coordinates. We extend our work to build a 1×2 and a 3×3 mesh-type router network (Fig. 3.5).

3.3 Synthesis Platform

We use the Xilinx $ML310$ board, which has a XC2VP30 FPGA [Xil06a], to functionally verify the stand alone router and the NoC system. We use Xilinx ISE $6.2i$ [Xil06a]

Reset State:
Initialize the signals
Go to `Wait-For-Request State`
Wait-For-Request State:
if *Current port is a* Local Port **then**
Wait for data transfer request (REQ) from the Network Interface (NI)
else if *Current port is a* Directional Port **then**
Wait for data transfer request (REQ) from the neighbor router (NI)
end
On receiving request, go to `Grant-Request State`
Grant-Request State:
if *Channel Buffer is Free* **then**
Send GNT to the connected Output Channel that requested
Data transfer begins from the next clock cycle
Go to `Receive-Data State`
end
Receive-Data State:
Use REQ as the Write Enable (WR_{en}) for the FIFO buffer
Receive data into FIFO till REQ becomes low
Go to `Initiate-Data-Out State`
Initiate-Data-Out State:
Based on header flit, initiate data transfer request to respective output channel
On receiving GNT, go to `Push-Data-Out State`
Push-Data-Out State:
Use GNT as the Read Enable (RD_{en}) for the FIFO buffer
Push the data out of the FIFO till $FIFO_{EMPTY}$ becomes high
Use $FIFO_{EMPTY}$ high signal to indicate end of transfer
On completion, go to `Reset State`

Algorithm 1: LiPaR: Pseudo Code of the Input Channel FSM

to synthesize the system and Modelsim $5.8c$ [Men07] to simulate the model and generate activity data of the Placed-And-Routed (PAR) model and the FloorPlanner tool of the Xilinx ISE 6.2i to implement placement constraints on the NoC system. The XPower tool of the Xilinx ISE 6.2i is used to get the power estimate values of the designs.

The router core is implemented in VHDL in a modular fashion. The data width and the FIFO depth are parameterizable. In this work, the data width is fixed at 8 bits (flit size). The coordinates of the router are designed to be fed from primary I/O. Hence, it is necessary to initialize the routers with their coordinate values at the start of simulation. Alternately, it is also possible to hardcode the coordinate values. But, the former approach gives more flexibility and is more suited in dynamic reconfiguration environment.

We use the Synchronous FIFO $v4.0$ from Xilinx LogiCORE. The parameters of the FIFO are customizable and can be appropriately set to meet the system requirements. The FIFO can be implemented as a Block-RAM (BRAM) or a distributed-RAM (dRAM).

The flow control mechanism is handshake based with minimal decoding logic. Both

Reset State:

Initialize the signals

Go to `Decide-Priority-Of-Current-Data-Transfer State`

Decide-Priority-Of-Current-Data-Transfer State:

(Dynamic Priority Scheme, that is statically decided) Based on the input channel that was serviced in the last transfer, Decide the Round Robin Priority Scheme for the current data transfer

Go to `Wait-For-Request State`

Wait-For-Request-&Grant State:

if *Multiple requests are received simultaneously* **then**
    Set the GNT for the input channel, according to the dynamic-fixed priority scheme decided in previous step

else if *Single request from an input channel* **then**
    Set the GNT for the requested input channel

end

Wait for acknowledgement (ACK) from the granted input channel

Go to `Setup-Crossbar State`

Setup-Crossbar State:

Set the select signals of the appropriate `MUX` and `DEMUX` to establish appropriate connection

Go to `Data-Transfer State`

Data-Transfer State:

Use ACK as the Read Enable (RD_{en}) for the FIFO buffer

Use $FIFO_{EMPTY}$ high signal from the sending input channel to indicate end of transfer

When data transfer is complete ($FIFO_{EMPTY}$ =1), reset the crossbar connections and disable FIFO and go to `Initiate-Push-Data-Out State`

Initiate-Push-Data-Out State:

if *Current port is Local Port* **then**
    Request the Network Interface of the Local Port

else
    Request the input channel of the neighbor router

end

Go to `Push-Data-Out State`

Push-Data-Out State:

Wait for ACK from the requested channel/ NI

Use ACK as the Read Enable (RD_{en}) for the FIFO buffer

Push the data out of the FIFO till $FIFO_{EMPTY}$ becomes high

Use $FIFO_{EMPTY}$ high signal to indicate end of transfer

On completion, go to `Reset State`

Algorithm 2: LiPaR: Pseudo Code of the Output Channel FSM

the input and output channels are buffered, so as to minimize the blockages in a store-and-forward buffering scheme. We employ XY routing, and the FSMs and decoding logic has been optimized accordingly. The arbitration scheme is dynamic, as there is round-robin arbiter implemented with a dynamic priority scheme.

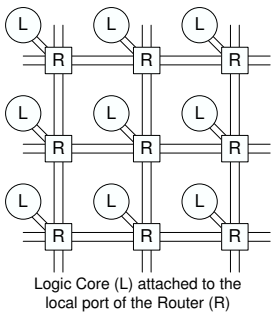


Figure 3.5: A typical 3×3 Mesh Network

3.4 Simulation and Results

We experiment the router design with various set of test cases, which are explained as follows.

3.4.1 Best Case: Single Router without blocking

We initially test a single router by feeding random inputs in a fashion such that there are no blocking taking place at any of the output channels. It is to be noted that establishing five simultaneous connections in parallel is possible in this router. Figure 3.6 shows that case where five simultaneous requests are being serviced by the router. As the inputs coming to the five input channels of the router, request five different outputs thus the router is able to service them at the same time and send them out through the output channels at the same time. As per the request the data from the local port (*rin_l*) is being routed to the west port (*r_out_w*), data from north (*rin_n*) is sent to the local port (*r_out_l*), data from east (*rin_e*) goes to south port (*r_out_s*) and the data from west(*rin_w*) is sent out via the north port (*r_out_n*) simultaneously. Figure 3.6 five simultaneous connections are established and serviced, when non-blocking inputs are present.

3.4.2 Worst Case: Single Router with blocking

Figure 3.7 shows the worst case when all the packets arriving at different input channels simultaneously request the same output channel This leads to blocking as one output channel cannot service all of them at the same time. Here the packets coming in via the north, east, south and west channels request the same local output channel of the router. So the Round Robin Arbiter at the local output port decides the order in which the input

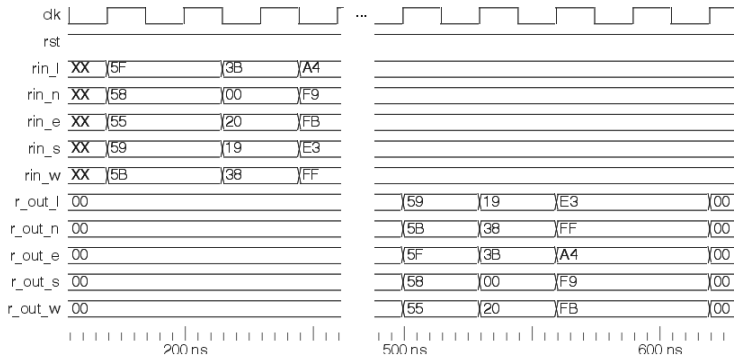


Figure 3.6: Simulation of stand alone router (non-blocking inputs)

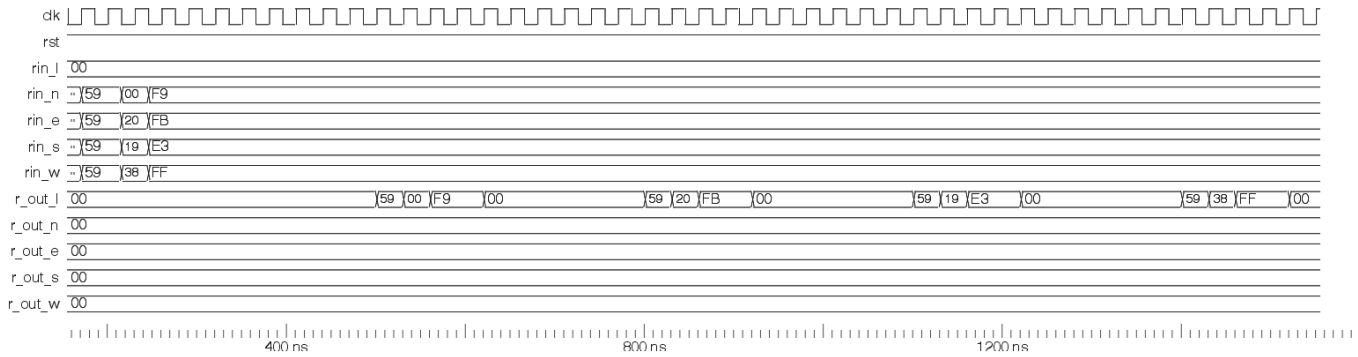


Figure 3.7: Simulation of stand alone router (blocking inputs)

requests will be serviced. In this case for the local output port the order of service is north, east, south followed by west according to the priorities given in the Arbiter.

3.4.3 3×3 Mesh network

Figure 3.8 shows the routing of two packets of data, one going from the local port of router 0 (*l00*) to the local port of router 8 (*l22_o*) and the other going from the local port of router 8 (*l22*)to the local port of router 0 (*l00_o*). These are the two of the four longest paths possible in the 3×3 matrix of routers. The other two longest paths are the cases where there is communication along other diagonal.

3.4.4 Timing Analysis

The timing analysis for both the non-blocking and blocking types of inputs are presented below. The frequency of operation of the router is found to be close to 95 MHz.

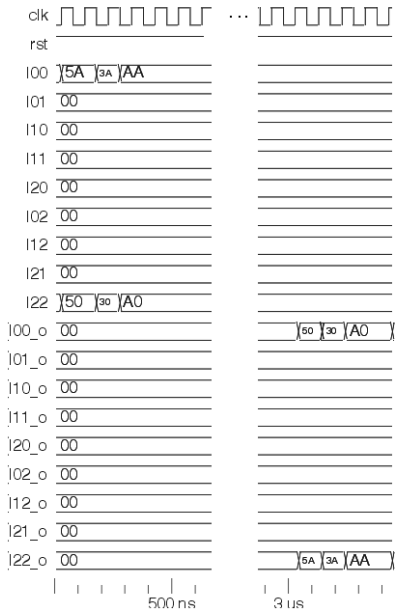


Figure 3.8: Simulation of the 3×3 mesh network

Packet Size (bits)	single router (# clocks)	$1 \times 2 mesh$ (# clocks)	$3 \times 3 mesh$ (# clocks)
16	10	20	50
32	14	28	70
64	22	44	110
128	38	76	190

Table 3.1: Timing report (stand alone router) - simultaneous non-blocking inputs

Best Case: Non-Blocking Style

In the case of the stand-alone router, each input channel requests a different output channel. In the case of 1×2 mesh network, the time taken for the first flit to reach from the input of one router to the output of the other router is given in Table 3.1. In the case of 3×3 mesh network, the time taken for the communication between the local ports of the two farthest routers is given in Table 3.1.

Worst Case: Blocking Style

The packets coming in through the west, north, east and south requesting the same local output port. So the inputs are serviced one by one in the order of their priorities as decided by the Round-Robin Arbiter of the local output port. In this case the order of service is north, east, south followed by west. Table 3.2 shows the number of clock cycles before the first flit of data coming in at each input channel arrives at the output of the router.

Packet Size (bits)	North clock cycles	East clock cycles	South clock cycles	West clock cycles
16	10	20	30	40
32	14	28	42	56
64	22	44	66	88
128	38	76	114	152

Table 3.2: Timing report (stand alone router) - simultaneous blocking inputs

The latency, L (in number of clock cycles) of our router is given in Equation. 3.1

$$L = \sum_{i=0}^{n} (6 + 2b + (S_i - 1)(6 + b)) \tag{3.1}$$

Here, n is the number of routers in the path, b is the number of bytes in the packet and S_i represents the service order number inside each router according to the arbitration scheme. Note that the S_i can vary between 1 and 4.

3.4.5 Synthesis Report

The results of synthesis of our router are given in Table 3.3. The BRAM-based implementation of the router consumes 437 slices. After putting an area constraint, the number of slices decreases to 352 (352 slices , 478 FF, 772 4-input LUTs, 10 BRAMs), thereby, consuming only 2.57 % of the total Xilinx XC2VP30 FPGA area. This is smallest possible parallel router with all the standard features. The synthesis report is presented in Table 3.3. We also synthesize the 1×2, 2×2 and 3×3 network and they take $874(6.38\%)$, $1748(12.76\%)$ and $3934(28.72\%)$ slices of Xilinx XC2VP30 FPGA, respectively, which is reasonable and we have a lot of area available for the user logic.

3.4.6 Power Analysis

Initially, we experiment to calculate the power consumed by a stand alone router. We feed random input vectors following uniform distribution. We simulate the post placed-and-routed model using ModelSim $5.8c$ [Men07] and generate an activity file of design. Then, we use Xpower [Xil06a] to calculate the average power taken by the router design. It is noted that the router consumes 824.25 mW, out of which 797.5 mW is the quiescent power. We then build 1×2, 2×2 and 3×3 NoC systems. The aim is to obtain power data for random input vectors. Care is taken to place each router module in the Xilinx Virtex-II

Component	BRAM based (# Slices)	dRAM based (# Slices)
Router	437	489
Crosspoint Matrix	98	98
Input Channel	34	35
Output channel (L, N, S)	55	60
Output channel (E, W)	36	41
Input Channel Controller	13	13
Output channel controller (L, N, S)	34	34
Output channel controller (E, W)	15	15

Table 3.3: Synthesis report for a stand-alone router

Pro FPGA (XC2VP30) [Xil06a]. This is required as we have to ensure that routers are optimally placed near the BRAM modules, while giving ample amount of space for the logic to fit in, without any area wastage. This is achieved using the FloorPlanner tool of the Xilinx ISE 6.2 tool set [Xil06a]. It is seen that the average power consumed by 1×2, 2×2 and 3×3 are 828.8mW, 844.58mW and 873.36mW respectively.

3.5 Conclusion

We present a light weight parallel router architecture for implementing Networks-on-Chip on FPGAs. We introduce optimizations in XY routing and decoding logic thereby gaining in area and performance. The header overhead is 8 bits per packet and the packet size can vary between 16 and 128 bits. Each router consumes only 352 Xilinx Virtex-II Pro FPGA slices (2.57% of XC2VP30). We also implement a 3×3 mesh network with a total area overhead of 28% leaving 72% of the area available for the logic in a Virtex-II Pro XC2VP30 device. We characterize the router and several mesh networks for power and performance parameters. We show the functional validation of the stand alone router and a 3×3 mesh network. We also present the timing data for a stand alone router, 1×2 network and 3×3 network. We obtain the area and power values of the design implemented on Xilinx XC2VP30.

Chapter 4

Multi Local Port Router

In a Networks-on-Chip style of design, systems are built by integrating different design cores in a preset fashion. *Design Cores* are the Intellectual Property (IP) cores that are pre-built and verified for functionality, performance and other constraints. On-chip communication networks offer several advantages including increased performance, modular & structured design, and reuse of cores [BdM02, DT01, KS03, ALMM05]. At the same time, there can be increased network area overheads, congestion, latency and bandwidth limitations. So far, researchers have primarily concentrated in developing an Networks-on-Chip (NoC) configuration and mapping that optimized a specific objective [SC05, HM03a, BJM+05, KPN+05]. But, the inherent limitations of a shared network prevent further optimization of the performance/power.

4.1 FPGAs & NoCs: Improving Area overhead

Exploiting the NoC style of system design for FPGAs in an active area of research [NMAM05, MBD+05, NMV+04, BDM+04, MNM+04, MMB+03]. Though system design using an FPGA is popular, area limitation is one of the major constraints that dictates the choice of designs. Because of this premium availability of the user logic area, the on-chip communication network should be as small as possible. This ensures that the maximum area can be utilized by the logic while maintaining the performance of the on-chip network. Also, reduction in the logic blocks used in FPGAs has a direct impact on the power consumption and the timing [SBKV05].

The on-chip network area can be reduced by means following approaches.

- The router is the key component of an NoC architecture and is the major source of the network area overhead. Hence, it is prudent to use *a small and simple router* supporting complete functionality, without sacrificing the performance [SBKV05].
- The *router count can be reduced* while maintaining the communication backbone between the existing communicating logic cores.

With n available cores, the latter strategy of area reduction cannot be exploited unless the cores are grouped together and made to share the same Network Interface (NI). Combining of core interfaces will further complicate the system integration process and cause severe performance bottlenecks due to increased congestion.

To solve this critical issue, we propose to use the second strategy by introducing an innovative architectural change to the router design to handle multiple design cores, simultaneously. The modified router architecture has more than one Local Port (LP), thus, capable of servicing multiple cores at the same time, without taking a hit on the performance.

4.2 Related Work

Premium availability of logic area in an FPGA based device demand that the router of the on-chip interconnection network be of small size. Researchers in the reconfigurable computing domain have tried to attack the area overhead issue of Networks-on-Chip implemented on FPGAs, by coming up with small and efficient router designs [ZSS04, SBKV05, ZKS04, ZSS04, ZS03, MCM+04, MdMM+03]. Sethuraman et al present the smallest router called LiPaR for FPGAs [SBKV05]. LiPaR is a light-weight router (2.57% in $XC2VP30$) capable of establishing parallel connections between various ports, simultaneously.

Apart from a prudential router design, an efficient NoC configuration and a proper mapping of design cores are important. In the literature, we have different mapping strategies for mesh networks [BJM+05, HM03a, LK03a]. An NoC synthesis flow is presented by Bertozzi et al [BJM+05]. They present algorithms for mesh NoC architectures under different routing functions and delay/bandwidth constraints. Hu and Marculesu present a branch-and-bound algorithm to get a power efficient mapping of cores onto tile-based mesh architectures, while satisfying the bandwidth constraints of the Networks-on-Chip [HM03a]. Ascia et al use evolutionary computing techniques to implement multi-objective explo-

ration mapping in mesh based Networks-on-Chip, to obtain pareto mappings optimizing performance and power [ACP04]. Lei and Kumar present a two-step genetic algorithm to map the task graph minimizing the execution time [LK03a]. Chan and Parameshwaran give a template based NoC generation methodology [CP04]. In many of the above works, the authors use non-deterministic routing schemes to minimize cost. In this context, the authors make an over-simplistic assumption about the router design & its capability and do not exhaustively discuss the overheads involved in the design. Overheads in implementing such complex routers and latency of the logic (that implements dynamic routing decisions) cannot be taken for granted, as it affects the final system performance. Deterministic XY routing is a simple routing scheme and yields a simpler decoding logic. Oblivious or adaptive routing schemes are observed to increase the overhead of the on-chip network as well as the latency of connection establishment [DT04, SBKV05].

The only conceptually comparable work is reported in [ARG05]. Æthereal architecture supports multiple cores to be attached onto the Network Interface (NI) kernel ports, but, not directly onto the router node. In spite of having multiple NI kernel ports, the single link (which is time-multiplexed between different channels) between the router and NI creates a large bottleneck, preventing simultaneous parallel connections from the router. Further, the connections between NI kernel ports suffer from a lengthy transaction going through a scheduler. Using Æthereal as the base architecture, Hansson et al present a scheme for combined mapping, routing and slot allocation [HGR05]. But, the fundamental differences in the underlying architecture and the non-availability of router metrics make it hard to achieve an apples-to-apples comparison.

To the best of our knowledge, this is the first work to propose the innovative concept of using Multi Local Port Router design to improve the Networks-on-Chip design. As a proof-of-concept, we present an optimal NoC configuration generation algorithm, for regular mesh architectures having Multi Local Port Routers (refer Chapter 6. Though an exhaustive search algorithm guarantees to find the optimal NoC architecture, it presents scalability and runtime issues. Hence, in this research, we present a heuristic mapping methodology for handling system task graphs of any size and generating efficient Networks-on-Chip configurations in a fast and efficient manner (refer Chapter 7).

4.3 MLPR Design

Multi Local Port Router is a modified router architecture where a router in the mesh can handle more than one logic core at the same time. The MLPR is based on the LiPaR design [SBKV05] (Figure 3.1), wherein the router header and the decoding scheme are modified so as to serve multiple cores simultaneously, without any performance penalty. The LiPaR is a parallel router that is capable of establishing between any channel-pair, simultaneously, without any additional latency. LiPaR is reported to be the smallest router design for Xilinx FPGAs [SBKV05]. The customization for Xilinx devices is because of the use of the efficient FIFO design cores (Synchronous FIFO v4.0) provided by Xilinx LogiCORE™ [Xil05]. Except for the FIFO part, the rest of design is universal in terms of its applicability. Figure 3.1 shows the block diagram of the LiPaR router which is traditional single local port router. We refer the readers to [SBKV05] for a detailed architectural description of the LiPaR design. Before we proceed with the modified architecture details, we discuss other NoC parameters of the designs that were used in this research.

4.3.1 Topology

Design cores can be mapped onto an on-chip network following various network topologies. Network topology refers to the arrangement and type of interconnection of the nodes. Various network topologies include mesh, torus, hypercube and fat-tree [DT04, DYN98, KS03]. Figure 3.5 shows a typical 3×3 mesh NoC. In an FPGA, the three-dimensional form of network topologies will increase the routing complexity, using the expensive global wires. Hence, two dimensional form of network topologies are preferred. We choose the mesh network topology because of the following advantages,

- The two-dimensional mesh topology is reported to be efficient in terms of area and power [BJM+05].
- Mesh topology uses less routing overheads and fewer # expensive global wires in FPGAs [SBKV05].
- Pin limitations and memory requirements (for logic & buffer) in an FPGA cause the IP cores to be spread across the FPGA. In a mesh style, cores can be distributed effectively complementing the above requirement. Also, the block-RAMS (Xilinx FPGA) used in the MLPR are distributed across the FPGA, and hence a mesh topology is preferred.

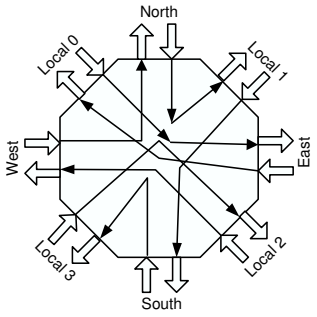


Figure 4.1: 4 LP Router *(having 8 Parallel connections)*

4.3.2 Routing & Flow Control

We use the deterministic XY routing and store-and-forward type of flow control*. XY routing is one of the efficient forms of routing for a mesh based NoC [KS03]. It imposes lesser overhead on the part of a router, as the decoding logic for routing the packets is simple. The popular flow control mechanisms like store-and-forward, wormhole or virtual cut-through have their own pros and cons. Wormhole routing reserves channels with the reduced buffer overhead for the routers, used in the on-chip network. But, there is a high possibility of under utilized channels and wasted bandwidth in certain applications [KS03]. Virtual channel approach removes the limitations of the wormhole approach to some extent. Store-and-forward, though having some buffer requirements, can result in increased channel utilization and a simple router [KS03, SBKV05].

4.3.3 Modified Architecture

For the proposed Multi Local Port Router architecture, we use LiPaR (having 8 bit header with the X and Y coordinates) as the base system. We modify the header (Figure 4.3) to have two parts, namely, the Address co-ordinate (A) and the Local Port ID (LID). The X and Y co-ordinates of the destination router are described in the address part (A). The number of bits of LID (and hence A) varies with the maximum number of Local Ports (LP) present in the router. There is no additional overhead on part of the header. Also, there is no reduction in the total number of addressable cores. This scheme primarily aims to replace the inter-router-channel communication with intra-channel communication. Please

*The MLPR architecture is a proof-of-concept router design. The methodologies and algorithms presented are applicable & extensible to other popular schemes like virtual channel, the use of which will further enhance the quality of results

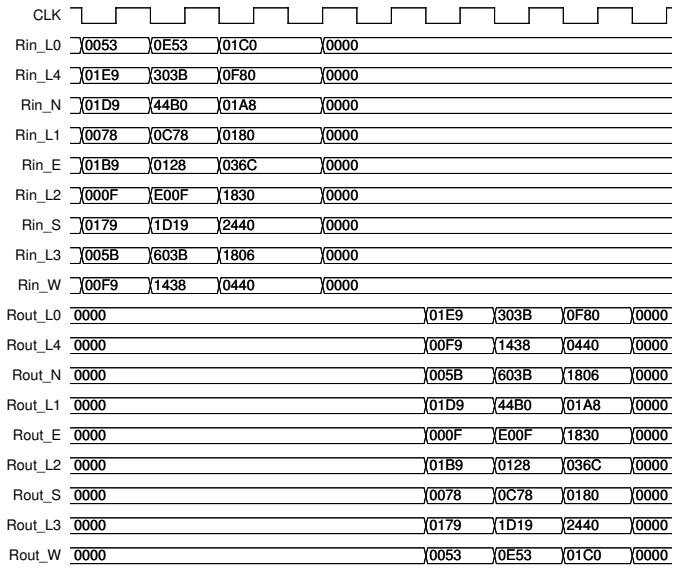


Figure 4.2: Simulation of a 5 LP Router, showing 9 Parallel Connections

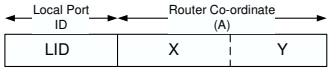


Figure 4.3: Modified Header Flit

note that there is no tail flit in the LiPaR design [SBKV05]. As a general case, an n-LP router having 4 directional ports is capable of establishing $n+4$ parallel connections simultaneously. Figure 4.2 shows the simulation of a 5 LP router where 9 parallel connections are established simultaneously.

Input Channel

The input channel is similar to the LiPaR, except for the additional decoding logic to operate on the upper 4 bits to identify the local port and issue appropriate request signals to a particular local port.

Crosspoint Matrix

The structure of the crosspoint matrix also remains similar to LiPaR, with changes happening with regard to the sizes of the `MUX`s and `DEMUX`s used to establish proper connections. There is a quadratic increase in the number of the interconnections, as from a particular port (directional or local) connection can be established to any of the remaining ports (directional or local). This is required to maintain the capability of initiating simulta-

neous parallel connections between any two ports.

Output Channel

Similar to the input channel, the output channel is very similar except for the fact that the round-robin arbitration FSM operating at each of the output channel will have to handle requests from larger number of ports.

4.3.4 Adapted Decoding Logic

The deterministic XY routing scheme is followed till data packet reaches the destination router. On reaching the destination router, the LID part of the header is used to identify the local port to which the packet is addressed. Based on the LID, the cross point matrix (made of Multiplexer (`MUX`) and Demultiplexer (`DEMUX`)) select signals are appropriately set. There is no central arbiter and because of the distributed nature of channel access, every channel-pair connection can be established in an independent and concurrent fashion. Thus, transfer is initiated to the appropriate local port, when more than one local ports are available. But for the extra decoding logic, to correctly select one among many logic (referred interchangeably as *local*, hereon) ports, the rest are same as the traditional mesh based NoC system. Since there is no clock penalty while establishing connections, this approach does not degrade the performance.

Figure 4.1 shows the block diagram of a 4 Local Port (LP) router. As an example case, we see the 8 connections between different input and output channels that can be established simultaneously. In general, in an n LP router with 4 directional ports, $n + 4$ connections can be established simultaneously. This is possible because of the presence of an arbiter at each of the output channels (no central arbiter) in LiPaR [SBKV05]. Because of the distributed nature of arbitration of channel access, the connections can be established independent of other channels. Please refer to [SBKV05] for an more details including the dynamic priority scheme.

In addition, we implement an important design optimization on part of Network Interface (NI) design. The Network Interface has a lookup table having the address coordinate information of the destination core (which is mapped to a (the) local port of the destination router). Assume that the task graph is mapped in a particular fashion on an Network-on-Chip architecture. If a source node communicates to multiple destination nodes (out-degree > 1), we give precedence in sending, based on the distance of the re-

ceiving node from the sender node (farthest first) in the mapped NoC architecture. This increases the data pipelining rate in the network and serves to reduce the overall time by overlapping the multiple transfers at the same time [SV06b]. There is no additional overhead on part of the header of each packet and there is no reduction in the total number of addressable logic cores. This scheme mainly aims to replace the inter-router-channel communication with intra-router-channel communication. This scheme is applicable to other types of flow control including the virtual cut-through and wormhole. Also, this scheme provides an opportunity to optimize the buffer size of the channels in each router [OHM05].

4.4 Architectural Advantages

A Multi Local Port Router provide several advantages in an Network-on-Chip design, bolstering the merit of this research. We summarize the salient merits of an MLPR below.

4.4.1 Bandwidth Optimization

An important advantage of an Multi Local Port Router is that it helps to optimize the data traffic by adjusting the BandWidth (BW) along different links (paths). This results in the improvement of the overall performance of the NoC system. Consider the task graph and an example mapping (Figure 4.5(a)). If the BW limit along each link is ≤ 300, the condition is violated at 3 different links (1-2, 2-3, 3-6). By using two 2-LP routers (Figure 4.5(b)), we are able to resolve all the bandwidth violations.

This is because the transfer between the Local Ports in an MLPR occur directly on a point-to-point basis, and hence does not affect the bandwidth of the main mesh of the NoC. Highly communicating nodes can be mapped onto the same MLPR, and hence reduces the traffic of main mesh. This results in increased performance and reduced energy consumption, in addition to other benefits. Thus, using this efficient architectural transformation, we can optimize the NoC design according to the bandwidth requirements.

In short, an MLPR transforms the traditional design into an NoC with 2 levels of network hierarchy, the normal mesh, and the point-to-point network (crossbar/pseudo-circuit-switched-network) within a single router. As shown in Figure 4.4, each router node in the NoC mesh can have different number of Local Ports (including the traditional single

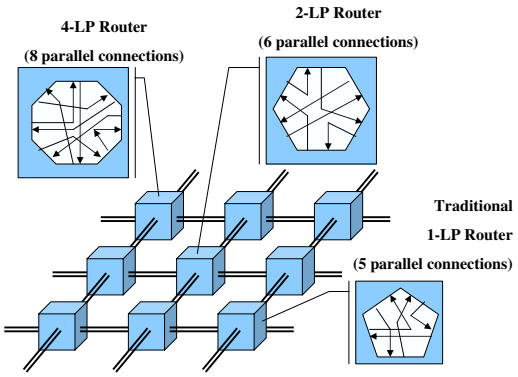


Figure 4.4: Multi Local Port Router based NoC mesh

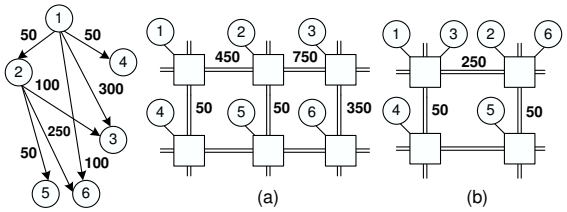


Figure 4.5: BandWidth Optimization (in data units/s)

LP router). For illustration, the polygons shown are indicative of the number of the Local Ports (= # polygon edges - 4 directional ports), and hence the number of parallel connections (= # LPs + 4) present in a router node. In Figure 4.4, the pentagon, the hexagon and the octagon represent, respectively, the single, double and four local port routers.

4.4.2 Area Reduction

Figure 4.6 shows the Xilinx ISE synthesis results for the router versions having 1, 2, 3, 4 and 8 local ports. Multi Local Port Routers provide an average area savings of 36% (maximum savings of 47.5%) in Xilinx $XC2VP30$ Virtex II Pro FPGA (upto 9 LPs) (Figure 4.6). The large area savings are due to the fact that for every single LP router that is removed (and the corresponding logic core added to an MLPR), we save upon 8 channel buffers (of 4 directional ports) and the associated decoding & routing logic. Furthermore, we save on the routing area (interconnections) between single LP routers, which was not directly available from Xilinx ISE™ Place-And-Route (PAR) tool [Xil06a].

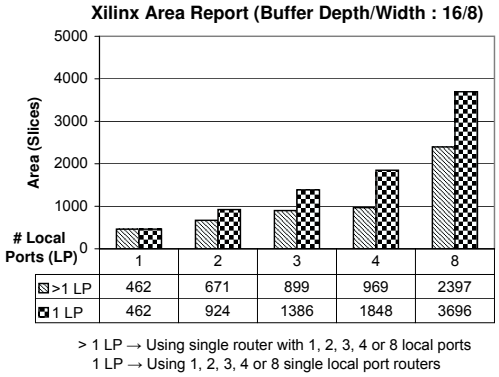


Figure 4.6: Synthesis Results

4.4.3 Power Savings

Quiescent power occupies close to 94% of the total power consumption in LiPaR [SBKV05]. Hence, area reduction in terms of the number of routers effectively reduces the usage of logic slices, thereby, reducing the power consumption substantially. Also, the power savings by optimizing data traffic is enormous [BdM01, SC05, HM03b, KS04].

4.4.4 Congestion Reduction

Networks-on-chip design using traditional approach can suffer in terms of performance, if the inter-communication of cores is not streamlined and managed properly. Several techniques have been proposed to map the cores on the NoC architecture [BJM+05] [HM03a] [LK03a] [ACP04]. In the traditional NoC design, there exists serious bottlenecks because of the sharing of the paths by different logic communications at the same time.

Congestion is an inherent problem in a shared medium [DYN98]. An efficient mapping algorithm can improve the performance by reducing congestion, following the intuitive way of nearest-neighbor first. But, the algorithms are handicapped in terms of mapping as they have limited positions available, when mapping cores. As the data traffic between the cores increase, there can be *heavy congestion due to shared paths*. Further, congestion creates and compounds the problems of deadlocks/livelocks [DT04]. Since Multi Local Port Router replaces an inter-router-channel transfer with an intra-channel transfer, the number of shared path(s) is reduced, resulting in ease of congestion in the network.

It is to be noted that if the task graph has bottlenecks at the receiving end (desti-

nation), even the multi-local port router will not be of much use. But, in any case, the performance will be better than a single-local port design.

4.4.5 Transit Time Reduction

The transit time of the packets is one of the performance bottlenecks in an NoC. If a proper network design and an efficient core mapping are absent, the performance of the system can be very poor. A mapping algorithm will try to place cores so as to reduce the number of hops. But, the final mapping may not be optimum. For example, consider a case where a core is communicating with more than four cores (out-degree > 4). In such a case, there are only four positions that can be reached in one router hop, in a $n \times n$ mesh. The fifth receiving core, at the best, can only be reached in 2 router hops.

In a large System-on-Chip having several high-communicating cores, with limited number of the nearby- locations available for the placement of cores, separation of cores to farther locations is a possibility. In such a situation, the cores can be placed in the opposite ends of the mesh in the worst case. This increases the hop count of the packets (transit time), thereby, affecting the overall system performance. With use of Multi Local Port Routers, the total number of hops can be significantly reduced. For every Local Port combined, we do away with two complete channel hops, which reduces the number of clock cycles by $2 \times \#flits$ [SBKV05]. This is because the inter-router-channel transfers are replaced by intra-router-channel transfers and this effect is more pronounced in routers having store-and-forward type of flow control.

4.4.6 Better Mesh Design

Multi-local port routers can be used to improve the NoC design by avoiding certain unnecessary routers. In a mesh-based NoC, when there are odd number of logic cores (hence odd number of routers), it creates a linear chain of routers giving rise to maximum router hops. But, with use of Multi Local Port Routers, we can transform the NoC design to use even number of routers, thereby, providing varied mesh topologies to experiment with (Eg., Figure. 7.1).

Consider the case where 5 cores are communicating in a particular fashion (Figure 4.7). With traditional mesh design, we have either a 5×1 or a 1×5 mesh. In this scenario, even with the best mapping, the communication between core 1 and 5 needs four router

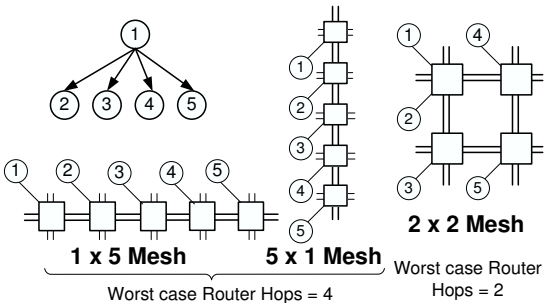


Figure 4.7: Mapping of Five cores

hops (assuming that the cores 2, 3 and 4 are placed by nearest-neighbor strategy). In other words, the worst case consists of 4 router hops. But, with the use of just one 2 LP router, we can implement a 2×2 design, reducing the worst-case router hops to 2. This is a very effective strategy and with large number of cores and complex interconnection patterns, the savings will be much larger.

4.5 Design Issues

The proposed approach provides gain in terms of area, power and performance. Intuitively, a single router with n local ports seems to be the best option. With a single n Local Port router, we can do away with the directional ports, going away from traditional NoC design. But, there are certain factors that have to be weighed carefully when choosing Multi Local Port Routers in an NoC design. The issues that may limit the maximum number of Local Ports in an MLPR based NoC design are discussed below.

4.5.1 Critical Path

Addition of more Local Ports to a single router results in increased interconnect count/length and larger decoding logic. The cross-point matrix (having the `MUX`s and `DEMUX`s) must handle increased number of interconnects. Further, the variation in the size of `MUX`s/`DEMUX`s is highly non-uniform. For example, an 8-LP or 6-LP router can be implemented in an efficient manner (8×1 `MUX` or a $4 \times 1 \& 2 \times 1$ `MUX`) than a 7-LP router which has to either use a 8×1 `MUX` or a combination of larger number of smaller `MUX`s. The mapping results observed of optiMap [SV06b] justifies this point. Thus, factors including the `MUX` size $\&$ availability of `MUX`s of required size in CLBs (FPGA device specific parameter) make the use of the larger MLPR designs inefficient compared smaller ones.

Also, this impacts the critical path of the router design, thereby, affecting the operating frequency.

In FPGAs, because of the organization of Configurable Logic Blocks (CLBs), the channels will be spaced and placed apart. This may increase the interconnect length of the intra-channel communication, thereby, increasing the critical path of the router. Thus, the performance of the router network may get impacted, because the operating frequency is reduced, compared to the smaller design versions. The variation of operating frequency between a 1-LP and 9-LP router is close to $15MHz$ which is significant [SV06b]. The router designs are reported to be operating close to $90MHz$. With increased local port count, we can expect a larger variation.

4.5.2 Buffer Requirements

Buffers at various channels of the router use the block-RAMs (bRAMs) available in the Xilinx $XC2VP30$ Virtex II Pro FPGA. The bRAMs are distributed across the FPGA fabric. Consider the case where a set of bRAMs are sufficient to implement a k-local port router and there are no contiguous bRAMs available. In such a situation, any additional port will mean that the bRAM from the next available set will have to be used. This increases the interconnect length.

Also, there is a limit up till which one FIFO can be synthesized using only a single bRAM available in Xilinx FPGA devices. That is, we can get a FIFO version upto certain depth and width. If we try to increase the depth and/or width of the FIFO, then that version of the FIFO has to make use of another bRAM from the FPGA. Since, there are limited number of bRAM resources available in an FPGA device,

Hence, one has to be very judicious in the FIFO selection for the final router(s), therefore, the NoC design. Also, cores that require high memory transfer are better placed near the bRAMs available in the FPGA. This will form a placement constraint and hence limit the use of higher local port routers. Hence, the buffer requirements and its availability, places an upper bound on the number of local ports that can be added to a single router, without affecting the performance.

4.5.3 Input-Output (I/O) Constraints

There are limited number of I/O pins in an FPGA. External I/O requirements of the cores will dictate the physical placement of the core. This in turn affects the placement of the router and its configuration. If a core is placed in one corner of the FPGA due to the I/O constraint and the associated router for the core is a multi-local port router present at a farther place in the FPGA, it leads to unnecessary increase in the interconnect length. This has a negative impact on the performance. This impact is more pronounced, if this path has high bandwidth requirements. Also, because of the use of long/global wires, the power consumption increases (due to higher capacitance switching). This effectively places an upper bound on the number of local ports that can be added.

4.5.4 Routing Resources Congestion

A larger Local Port router requires larger number and complex form of interconnects (quadratic increase in the number of interconnects inside the crossbar). With the area/placement constraints, this may create problems for the FPGA Place-And-Route (PAR) tool. For a 9-LP router, PAR was successful by the Xilinx ISE [Xil06a], with increased synthesis time. FPGA's have limited routing resources. A router with numerous local ports places a big constraint on the synthesis tool. A cross-point matrix (with `MUX`s and `DEMUX`s) implements many-to-many interconnections. There is a limit on the number of lines that can be drawn from a component (say, a multiplexer to the channel in all the ports) in an FPGA. Hence, the synthesis may fail for a router design with numerous local ports, if an area constraint for the router is placed. This forms another constraint on the number of local ports in a router.

4.5.5 Logic Requirements

Area is at a premium in an FPGA device. We have to accommodate as big a logic core as possible inside the reconfigurable fabric. IP cores are made available with certain dimensions and hence has to be effectively placed in the available CLBs of a reconfigurable device. Alternatively, the logic cores can be forced to fit in one of the few available spots adjacent to a router, using placement constraints. Arbitrary addition of local ports to a router may cause the placement of the logic cores infeasible, which would have been possible otherwise with smaller router designs. Hence, the aspect ratio requirements of the

logic cores put a constraint on the NoC design, at large. This, in turn, affects the available choice of routers.

4.5.6 Arbitration

Our modified router design has an arbiter at each of the output channels and there are no extra cycles wasted for arbitration. The access grant of an output channel is given inside a single FSM state (`if-then-else` construct) and hence has a fixed round-robin service scheme. To be fair, this round-robin structure is changed in the current arbitration cycle, based on the port granted access in the previous cycle. For example, if Local Port 1 was granted access in the previous transfer cycle, it (LP1) will get the *least priority* (i.e., polled last) in the current transfer cycle. Thus, the service order is changed (coded as nested `FSM`s), giving rise to a *revolving round-robin* structure. Even this new scheme may not be totally fair, if multiple input channels are requesting the same output channel. Here, since the amount of wait time of each request is not taken into account, the (fixed) service order, in the current arbitration cycle, will not preferentially grant access to the input channel that has been waiting for a longest period of time. Alternately, a scheme addressing the wait time of the requests will result in a more complex and larger router.

4.5.7 Address Utilization Factor

LiPaR uses just a 8 bit header for a maximum payload of 120 bytes. With the 8 bit header (in case of single local port routers), we can address a maximum of 16×16 NoC system and can get the desired type of mesh (square or rectangle). But, consider the case where there are a maximum of 8 local ports in a router. This will use 3 bits of the header. We have 5 bits remaining, thus, removing the possibility of a square mesh. Thus, for a square mesh, the $\#$ bits of A must be even, in order to get equal $\#$ bits for X and Y. In other words, the $\#$ bits of LID must be a multiple of two ($\#$ Local Ports must be a multiple of four). Hence, for a desired mesh dimension, a constraint on the number of local ports is created. Also, the $\#$ bits of A will determine the maximum dimension of the NoC system. Theoretically, we can map equal number of cores similar to a system having only 1-LP routers. In order to achieve this, same size MLPRs must be used in every node in the mesh, i.e., LID address space in each router must be utilized fully.

Based on the above factors, we investigate the alternate router architecture with

multiple local ports against the traditional NoC implementation. Apart from critical path constraint (that directly affects the operating frequency), others dictate the upper bound on the number of local ports that can be used. We form a cost function and the set of constraint files incorporating all the above parameters and experiment with a variety of benchmarks (discussed in Chapter 5).

4.6 Scope for Reducing the Latency

The idea behind buffering at both input and output channels is to avoid a packet waiting at an input channel from preventing the following data packets that share the path through the same input channel. The solution here is to push the data packet to the output channel (from where it leaves the router), by which any future *waiting* will not prevent the input channels of the router being blocked (through which there are other potential data transfers possible).

It is evident that with the use MLPRs, the size of the NoC mesh is considerably reduced. To restate what was mentioned earlier, the connections happen more frequently as intra-router-channel connections in place of the traditional inter-router channel connections. Hence, it is intuitive to avoid buffering at both input and output channels. Hence, a low latency router is possible by having just the input channels buffered. The key point to be noted here is that now the connections are established between the various input channels, in place of the earlier MLPR version where the connections happened between an input-output channel pair. Removal of output buffers leads to the change in both the input and output FSMs (refer Algorithm and). This is explained in detail in []. The latency of the new low-latency router is described by the equation

$$L = \sum_{i=0}^{n} (3 + b + (S_i - 1)(3 + b)) \tag{4.1}$$

Here, *n* is the number of routers through which the packet hops, b is the number of flits per packet, the term 3 refers to the constant latency that is caused because of the arbitration algorithm, and S_i is the service order number defined by the arbitration algorithm. In an *n* Local Port router, for a given ports, requests can come from the remaining *(n+3)* ports (including directional and local) and hence, S_i can vary between 1 and (n+3). Note that because of output buffering the latency according to number of flits is now *b* instead of *2b*. Also, the constant latency factor has come down from 6 to 3.

Reset State:

Initialize the signals

Go to `Wait-For-Request State`

Wait-For-Request State:

if *Current port is a* Local Port **then**

Wait for data transfer request (REQ) from the Network Interface (NI)

else if *Current port is a* Directional Port **then**

Wait for data transfer request (REQ) from the neighbor router (NI)

end

On receiving request, go to `Receive-Data State`

Receive-Data State:

Use REQ as the Write Enable (WR_{en}) for the FIFO buffer

Receive data into FIFO till REQ becomes low

While data transfer is in progress, based on header flit, initiate data transfer request to respective output channel

On completion of transfer, wait for grant (GNT) from requested output channel

On receiving GNT, go to `Push-Data-Out State`

Push-Data-Out State:

Use GNT as the Read Enable (RD_{en}) for the FIFO buffer

Push the data out of the FIFO till $FIFO_{EMPTY}$ becomes high

Use $FIFO_{EMPTY}$ high signal to indicate end of transfer

Go to `Wait-For-Request State`

Algorithm 3: Pseudo Code of low latency router Input Channel FSM

4.7 Conclusion

We present a novel approach of incorporating routers with multiple local ports in a regular mesh based Networks-on-Chip design in FPGAs. We find that MLPRs provide a host of advantages, especially reduction in area and transit time of packets in the network. We analyze the merits and the constraints involved in using such a design methodology. As evident from the discussion, the advantages of an Multi Local Port Router can be exploited in order to create a high-performance router design and hence, more efficient Networks-on-Chip architectures.

`Reset State:`

Initialize the signals

Go to `Wait-For-Request State`

`Wait-For-Request-&Grant State:`

if *Multiple requests are received simultaneously* **then**
Set the GNT for the input channel, according to the dynamic-fixed priority scheme

else if *Single request from an input channel* **then**
Set the GNT for the requested input channel

end

Wait for acknowledgement (ACK) from the granted input channel

if *Current router is Destination Router* **then**
Request the Network Interface of the destination Local Port

else
Request the input channel of the neighbor router

end

On receiving ACK, go to `Setup-Crossbar State`

`Setup-Crossbar State:`

Set the select signals of the appropriate `MUX` and `DEMUX` to establish appropriate connection

Go to `Data-Transfer State`

`Data-Transfer State:`

Use ACK as the Read Enable (RD_{en}) for the FIFO buffer

Use $FIFO_{EMPTY}$ high signal from the sending input channel to indicate end of transfer

When $FIFO_{EMPTY}$ of the sender input channel becomes high, reset the crossbar connections and disable FIFO and go to `Wait-For-Request-&Grant State`

Algorithm 4: Pseudo Code of O/P Channel FSM

Chapter 5

Experiment Setup

A typical System-on-Chip (SoC) design is described as a task communication graph. The application is described as a Directed Acyclic Graph $G(T, E)$ (referred to as *Task Graph* hereon), where T represents the vertices (tasks) and E is the set of directed edges describing the precedence, the dependence, the timing and the bandwidth constraints in the task graph. Nodes in a task graph represent independent units of computation. Parallel branches represent parallel code sequence of the algorithm/system. All parallel computation structures can be represented as task graphs. Task graphs cannot represent control flow information such as loops. In fact, all the control and loop structures are abstracted inside the node in the form of IP core. If a loop structure cannot be implemented within a node, then the loop is unrolled in space using redundant copies. However, Most applications, including Fast Fourier Transform, Discrete Cosine Transform (DCT) and Auto Regressive Filter, can be described in the form of task graphs.

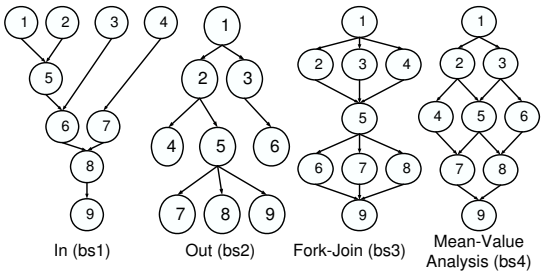


Figure 5.1: Basic Task Graphs [KA96]

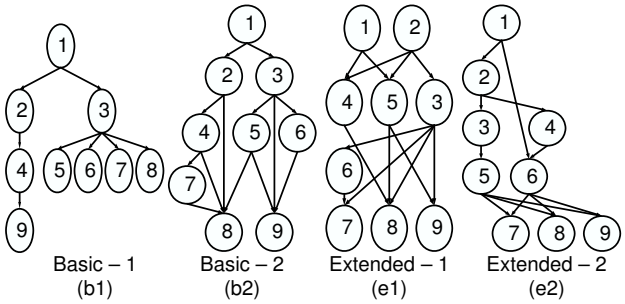


Figure 5.2: Benchmark Set 1

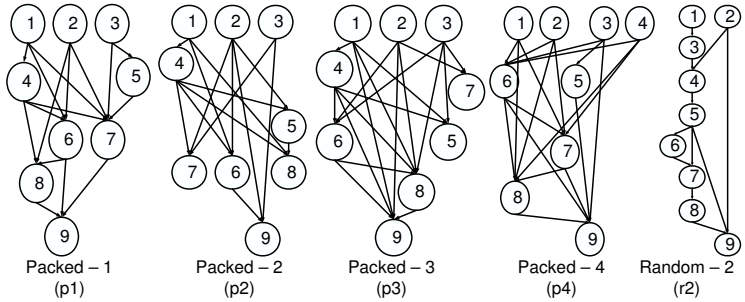


Figure 5.3: Benchmark Set 3

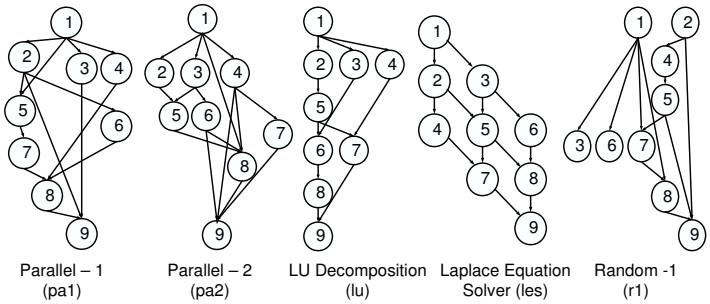


Figure 5.4: Benchmark Set 3 (lu,les [KA96])

5.1 Benchmarks

Kwok and Ahmad [KA96] use a set of different task graph types for studying various scheduling algorithms for multiprocessors, including Out-Tree, In-Tree, Fork-Join and Mean-Value-Analysis. Figure 5.1 shows the basic graph structures [KA96]. These basic task graph types represent high level task structures that are commonly encountered in parallel applications. We use a tool called Task Graphs for Free (TGFF) [DRW98] to generate the basic task graphs having nine nodes ($bs1, bs2$) by fixing the in-degree, out-degree, and dependence width/depth. The fork-join ($bs3$) and mean-value-analysis ($bs4$) graph structures are manually created, as TGFF was not capable of generating these graph structures.

In order to generate the rest of the benchmarks, we write a C++ program that takes in the set of basic task graph structures that were already generated and outputs an application task graph, G. The application task graph G is formed by a random combination of the basic graph structures, by varying the dependence degree, the width and the depth across different levels of nodes. It is to be noted that most of the application task structures can be generated using the combination of the basic graph structures and hence the program generates a variety of the system task graphs. We set the upper limit on the number of tasks in the graphs to nine and develop a set of eighteen synthetic benchmarks (including the 4 basic graphs). We fix the number of nodes at 9 because it represents a 3×3 NoC system, a typical case. But, most importantly, the scalability and runtime issues of the optiMap algorithm make the use of benchmarks with more than nine nodes infeasible (discussed in detail in Chapter 7). We analyze this system with 9 single local port routers, against the new scheme.

The optiMap algorithm presented in Chapter 6 explains the strategy to find an optimum mapping in the new scenario involving Multi Local Port Routers (discussed in previous Chapter). Figures 5.2, 5.3 and 5.4 shows the benchmark set that was generated using our tool. We omit the specific details of the nodes and edges of each of the benchmarks due to space limitations. The benchmarks $b1$ and $b2$ represent the simple graph structures, whereas $e1$ and $e2$ have large out-degree. Benchmarks $p1$ - $p4$ represent packed structures [DRW98], while $r1$ and $r2$ are two random cases. Benchmarks $pa1$ and $pa2$ have high degree of parallelism. LU Decomposition (lu) and Laplace Equation Solver(les) represent the practical examples [KA96]. The set of eighteen synthetic benchmarks cover a wide variety of graph types encountered in multiprocessor and System-on-Chip applications. For experimentation purposes, we assume equal execution times (defined as *k* clock

cycles) for all nodes in the graph (communication time from Network Interface to logic is also abstracted into this lumped time) and equal bandwidth constraints (50 data units/s) for all the edges in the task graph. Additionally, we test the cMap algorithm on the 4 widely experimented practical benchmarks (*MPEG4*, *VOPD*, *MWD*, *FFT*). The ***FFT*** benchmark has 15 nodes and the other 3 benchmarks have 12 nodes. We refer the readers to [KA96] and [JMBM04] for the details (task structure, bandwidth) on these 4 benchmarks.

5.2 Experiment Platform

The $ML310$ board from Xilinx® Inc. is used to functionally verify the various designs of the stand alone router and the NoC system. The board has Virtex II Pro™ FPGAs ($XC2VP30$) [Xil06a]. The buffers for the router are implemented using the efficient FIFO design cores (Synchronous FIFO v4.0) provided by Xilinx LogiCORE™ [Xil05]. Xilinx ISE™ $6.2i$ is used to synthesize the designs and Modelsim™ $5.8c$ [Men07] is used to simulate the model and generate activity data of the Placed-And-Routed (PAR) models. Placement/ Timing constraints are input using the FloorPlanner™ tool of the Xilinx ISE™ $6.2i$. Power estimates are obtained from the XPower™ tool of the Xilinx ISE™ $6.2i$.

The mapping algorithms (discussed in detail in following Chapters) were coded in C++/STL and were executed on a SunBlade™ 1000 workstation having dual processors operating at $750MHz$ and $2GB$ RAM. The average execution time of the optiMap and μMap algorithm varied between 5 and 6 hours. The large execution time is due to the fact that the algorithm does an exhaustive search of a very large design space (refer Chapter 6), to find the optimum NoC configuration. The cMap algorithm took a couple of seconds (4 seconds for 15 node *FFT*) to arrive at the near-optimal NoC configuration. This small runtime is highly suited for a dynamic reconfiguration environment involving online placement of tasks.

5.3 Conclusion

We discuss the experimental setup and framework adopted throughout the thesis. Because of scalability issues and lack of standard benchmarks for System-on-Chip design, we present a method to generate synthetic benchmarks that is described by a task graph. The node in a task graph represents an IP present in a System-on-Chip and the edges determine the data transfer happening between the nodes (IPs), described by the bandwidth of data transfer.

Chapter 6

Optimal NoC Configuration Generation

Generation of a Networks-on-Chip configuration, described by the topology and the mapping is a great challenge. In this chapter, we present an algorithm which maps the input task graph optimally, minimizing the overall execution time. This is an *exhaustive search* algorithm and is guaranteed to find the optimal configuration and hence a formal proof is redundant. For a given set of constraints and objectives, the algorithm finds the optimum number of routers, the configuration of each router, the optimum mesh topology and the the best possible mapping of cores onto the NoC architecture. The algorithm effectively automates the NoC design cycle by finding the optimum mesh topology and the final mapping for the given task communication graph. We test the algorithm on a wide variety of benchmarks presented in Chapter 5 and discuss the results.

6.1 optiMap: The Mapping Algorithm

6.1.1 Mapping in an MLPR-based NoC

Generating efficient NoC architectures and mapping of cores using Multi Local Port Routers is not a simple process and presents a complex design environment. It is to be noted that increasing the number of local ports will not always guarantee better performance. In certain cases, the traditional NoC architecture may be better under some constraints. Even in that situation, there can be a better mapping compared to a naive nearest-neighbor based mapping.

Let us consider the example shown in Figure 6.1, to give a flavor of mapping on

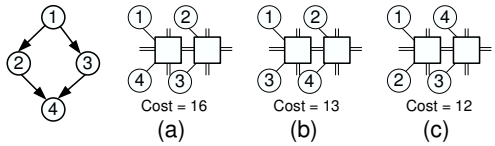


Figure 6.1: Mapping using with 2 Local Port routers

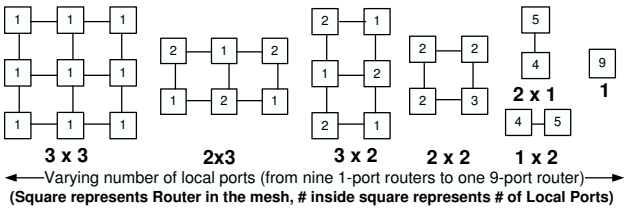


Figure 6.2: Mapping Search Space for Cores

multi local port routers. Here we choose a 1×2 mesh each having a two local port router. Let us consider a simple cost function where we sum number of hops. We increment the cost for each channel access, to get the final cost. We observe from Figure 6.1 that the cost is reduced by 25%, with proper mapping. This represents the simplest of cases, where the queuing effect at channels and other cost parameters are not considered.

In such a complex NoC system, a mapper that finds an optimum NoC configuration is required. Hence, as a proof-of-concept, we present a mapping algorithm, capturing the effects explained in Section 4.4 and 4.5. The algorithm does an exhaustive search and hence guarantees to find the optimal configuration and hence a formal proof is redundant. The mapping algorithm can be easily extended to incorporate additional cost parameters, thereby, giving a multi-objective based mapping algorithm. The algorithm efficiently maps the cores of the given task graph for various objectives and constraints. Also, the algorithm finds the optimum number of routers, the configuration of each router and an optimum mesh topology for a given task graph. We test the algorithm on a wide variety of benchmarks and report the results.

6.1.2 Problem Definition

Given a system level task graph, $G(T, E)$ and the set of constraints, find the optimum NoC configuration, that is, find the number of routers, the mesh topology configuration of each router and the final mapping of logic cores, reducing the cost function.

Input: Given a system level acyclic task graph, $G(T, E)$ with *n* logic cores
Input: Input the placement constraint, routing constraint, I/O constraint and buffer constraint
Output: The optimum NoC configuration
1 *Analyze the constraint file and set the upper bound of the router configuration, thereby, defining the search space of the algorithm*
2 **repeat**
3 γ: Generate all partition configurations for *n* for the defined search space
4 Γ: Permute and generate all possible combinations of γ
5 **foreach** *Partition configuration in* Γ **do**
6 Define the # of routers for the present configuration
7 Define the configuration of each router
8 Υ: Generate all possible mesh topologies for the partition configuration
9 **foreach** *Mesh and Partition configuration in* Γ, Υ **do**
10 φ: Generate all possible ways of mapping of cores
11 **foreach** *Mapping of given Mesh and Partition configuration in* $\Gamma, \Upsilon, \varphi$ **do**
12 **foreach** *Edge in the Task graph* **do**
13 Identify the source (i_1, j_1) and the destination routers (i_2, j_2)
14 κ: Decompose in terms of the intra/inter-channel communications between (i_1, j_1) and (i_2, j_2) using XY routing scheme
15 Update the communication times of all channels
16 Calculate the Queue times (Q_t) at all channels and update the transit/arrival times appropriately
17 C_f = $\alpha\times$ Total Execution Time of cores (transit, core execution and arbitration times) $+\beta \times Q_t$
18 **if** $C_f <$ *Best.Config.Cost* **then**
19 Best.Config $\leftarrow$ Current.Config
20 **end**
21 **end**
22 **end**
23 **end**
24 **end**
25 **until** *All the configurations are evaluated*

Algorithm 5: optiMap Algorithm

6.1.3 Description of optiMap Algorithm

Algorithm 5 presents the pseudo code of the optiMap Algorithm. Before the start of the NoC configuration generation, the algorithm performs a pre-processing step. In this step, the various constraint files (area, placement, buffer & operating frequency) are parsed to determine the upper bound (UB) on the number of local ports that can be used during the course of the algorithm. Based on UB, the next phase involves generating different router counts (partitions). For the given task graph , the problem of finding the different partition configurations (γ) is translated into the *partitioning of an integer* problem. That is, for a given value (n), find the combination of numbers (which are $\leq n$) such that the sum of the numbers is *equal* to n. We permute the above configurations and find all possible ways of partitioning the value n (which is the number of nodes in the graph) to get the set Γ. Using the set Γ and based on the constraints, we describe the number of router(s) and the

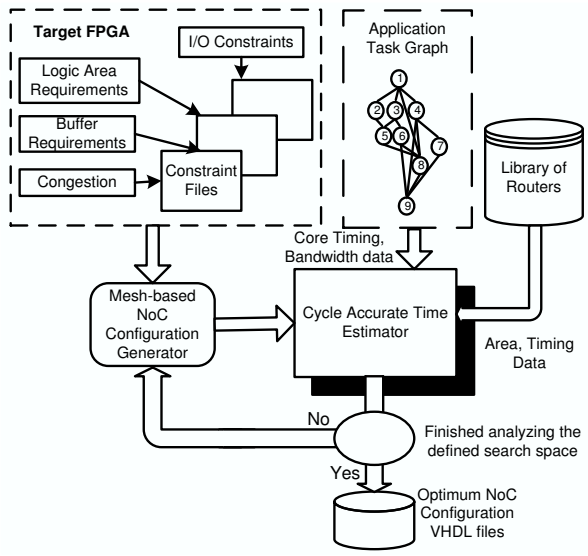


Figure 6.3: Algorithm Flow of optiMap

configuration of each router. For each partition configuration, we generate all possible mesh topologies to get the set Υ. After this step, we map the cores onto each of the configuration in all possible ways (φ) and evaluate the cost function.

Figure 6.2 shows the search space of the mapping algorithm for a specific case of a system having nine cores. The cost function is the weighted sum of total execution time (which includes transit time, core execution time and arbitration time) and queue delay (due to simultaneous access of paths/channels). The factors explained in the Section 4.5 like routing density, placement constraints, input/output constraints, buffer requirements, etc. dictate the upper bound of the search space, that is, the maximum number of local ports that can be added to a router. We build a cycle accurate (C++ based) simulator to calculate the execution and queue times, for the overall data transfer. In other words, we find the overall execution time of the given task (application). In the end, the algorithm outputs the best NoC configuration (including the best partition, configuration of each router, the mesh topology, and the optimum mapping of the cores).

In short terms, we are doing an exhaustive search of all possible NoC configurations (Figure 6.3). Analytically, for a task graph with n nodes, the total number of configurations analyzed is (2^{n-1} partitions) $\times$ (# mesh configurations) $\times$ ($n!$ ways of mapping n cores). The # of mesh configurations (k) for a given value of partition (p) is k += t (where t=1, if ($\frac{p}{i}$)=0; else t=0;), where i = 1,...,p.

6.2 Experiment Results

In this research, we make the optiMap algorithm to search the entire search space, while caching the results under different constraints. We collect the results of all the benchmarks and form a database. Based on the specific constraints, we can derive the results for a specific case from this database. We perform this to present a broad picture of the effectiveness of proposed architecture and the algorithm. For experimentation purposes, we assume equal execution times (defined as *k* clock cycles) for all nodes in the graph (communication time from Network Interface to logic is also abstracted into this lumped time) and equal bandwidth constraints for the edges in the task graph. The equal execution time/BW assumption in the synthetic benchmarks is primarily to find the effect of the task graph structure on the final optimal mapping. The operating frequency of the largest Multi Local Port Router in the final NoC architecture is taken as the overall operating frequency. The frequency value is back-annotated to get the absolute time[†] from the number of clock cycles taken at each step (node) of the task graph. The time unit of the synthetic benchmarks reported in *ns* is for illustrative and intuitive purposes only. Without any loss of generality, the methodology and algorithm is applicable to any system at hand, provided the clock cycle latencies between the various nodes in the task graph and the operating frequency of the system are known. We discuss the results of selected benchmarks in each of the cases below.

6.2.1 Optimization Cases

Upper Bound on # Routers

Figure 6.7 shows the execution times[*] of selected benchmarks where there exist an upper bound on the number of routers used. It is to be noted that for a case where the upper bound on the number of routers is k, the optimum configuration can contain combination of router(s) with $\leq k$ local ports. The timing of the router with maximum number of local ports is taken as the timing of NoC configuration and back-annotated in the simulator. It is seen in all of the benchmarks that reduction in the number of routers increases performance. Interestingly, from the Figure 6.7, we see that having 8 routers is more beneficial than having 7 routers. This is due to the fact the maximum number of hops (diagonal length) is reduced from 6 to 4. Overall, we infer that it is better to reduce the

[*]Refers to the completion (end) time of the last node in the task graph

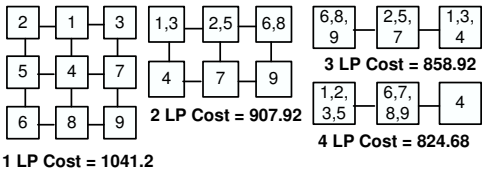


Figure 6.4: LU Decomposition *(lu)* - Optimum NoC configuration with varying upper bound on # local ports(LP) - the mapped cores are inside square (router) *(shown till # LP = 4)*

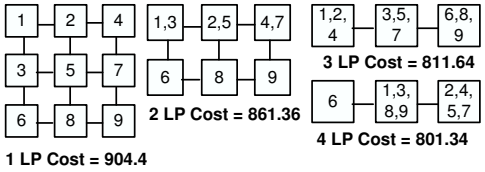


Figure 6.5: Laplace Equation Solver *(les)* - Optimum NoC configuration with varying upper bound on # local ports(LP) - the mapped cores are inside square (router) *(shown till # LP = 4)*

router count, by introducing multi LP routers.

Upper Bound on # LP

In Figure 6.8, we fix an upper bound on the number of local ports (timing data in Table 6.1). We see that increasing the number of local ports increases performance. But, it is seen in most of the cases that the 7 LP version is not better than the 6 LP version. Also, for the specific case of $p4$, the 8 LP version is better than the 9 LP version. The best NoC configurations obtained by the optiMap algorithm for lu, les and $p4$ (for different upper bounds on number of Local Ports) are shown in Figures 6.4, 6.6 and 6.5, along with the single LP versions. It is to be noted that the algorithm finds the optimum NoC configuration even for the single LP router version. Interestingly, in Figure 6.5, for a 3 LP upper bound case, the optimum NoC configuration uses 4 routers (instead of three 3 LP routers). This is due to the fact that the algorithm tries to reduce the overall hop count (thereby reducing overall time) and the inter-communication pattern of $p4$ dictates this configuration.

We observe an average performance improvement of 30% across the set of benchmarks, compared to a single LP design. To summarize, the optimum number of routers and the router configuration (and the final mapping) depends on the given application task graph and the maximum frequency of operation of the NoC system. Across all the benchmarks,

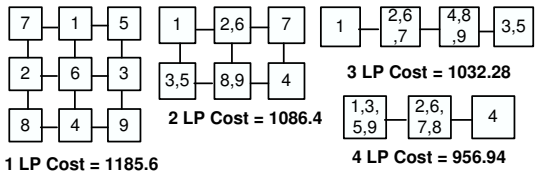


Figure 6.6: Packed-4 *(p4)* - Optimum NoC configuration with varying upper bound on # local ports(LP) - the mapped cores are inside square (router) *(shown till # LP = 4)*

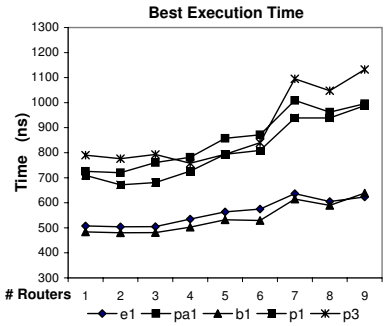


Figure 6.7: optiMap Experimental Results I

it is observed that choosing a single LP router design is never optimal, thus, validating our proposed approach.

6.3 Conclusion

We present a proof-of-concept algorithm to find the optimal Networks-on-Chip configuration describing the optimal number of routers, the optimal mesh topology and the optimal mapping of cores onto the mesh. We experiment with a wide set of benchmarks and report the results. The results show significant area savings and improvement in performance, thereby, validating the merit of an MLPR-based Networks-on-Chip.

Max local port count	Benchmark Execution Time (ns) †				
	$p4$	$pa2$	les	lu	$r2$
1	1185.60	1086.8	904.40	1041.20	1451.60
2	1086.40	985.52	861.36	907.92	1241.60
3	1032.28	914.08	811.64	858.92	1158.36
4	956.94	894.70	801.34	824.68	1089.20
5	889.28	849.58	786.06	817.82	1071.90
6	829.92	821.94	758.10	805.98	1077.30
7	817.02	825.03	776.97	809.01	1081.35
8	808.00	792.00	760.00	776.00	1024.00
9	814.06	773.76	717.34	749.58	1023.62

Table 6.1: Execution Time - Upper bound on # LP

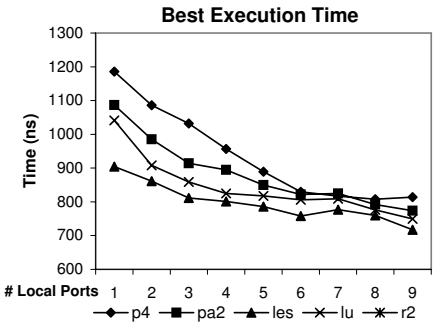


Figure 6.8: optiMap Experimental Results II

Chapter 7

Heuristic Fast Mapping Algorithm

The optiMap is an exhaustive-search algorithm that is capable of finding the optimum NoC configuration. But, because of the expensive configuration-generation step, the search space of the algorithm exploded for larger designs. optiMap requires generation of $n!$ combinations, each for creating partitions (outer loop) and creating mapping (inner loop). The memory requirement of the process (for the configurations) is exorbitant as n increases. For instance, the size of the configuration file for a design with 9 nodes is $70MB$. For every node added, the size jumps by the factor of 10 (eg., approx $700MB$ for 10 node graph). The factorial size of the configuration and mapping generation ($O(n^n)$) make the problem NP hard. Also, the runtime is in the order of several hours because of larger search space. All these factors make the use of optiMap impractical for larger systems.

7.1 cMap: The Fast Mapping Algorithm

In this section, we present a heuristic fast mapping algorithm (cMap) for generating NoC architectures using Multi Local Port Routers [SV07a]. The cMap algorithm exploits the advantages offered by a Multi Local Port Router and arrives at a near-optimal NoC configuration. Let us assume that the cost calculation of a given NoC configuration as the basic operation. Then, the average order of complexity of cMap algorithm is $O(n^2)$, as against optiMap that analyzed $O(n^n)$ NoC configurations. In the worst case, when maximum expansion occurs (explained in detail below), the cMap complexity increases to $O(n^3)$. Unlike optiMap algorithm, cMap can handle task graphs of any size. The NoC architecture generation process is reduced to a couple of seconds. In addition to the eighteen synthetic

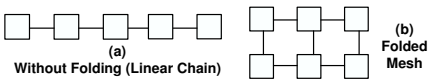


Figure 7.1: cMap: Folding Example

benchmarks explained earlier, we test the cMap algorithm on the four widely experimented practical benchmarks and analyze the results in Section 7.2.

7.1.1 Problem Definition

Given a system level task graph, $G(T, E)$ and a set of constraints/objectives, find an NoC configuration (number of routers, mesh topology, configuration of each router and the final mapping) with minimum cost* [SV07a].

7.1.2 cMap Algorithm Description

The cMap Algorithm (refer Algorithm 6) has five phases as follows.

- Mesh Definition (steps 6-7)
- Folding (step 8)
- Core Placement (steps 10-22)
- Design Evolve (steps 23-34)
- Design Perturb (steps 35-41)

Mesh Definition

The results of optiMap demonstrate the merits of the Multi Local Port Routers, favoring lesser router count and hence, routers with more number of Local Ports. Using this inference, we start with a mesh with the minimum router count (k). This is obtained by defining routers (k) having maximum number of Local Ports (m).

*For a fast-mapping algorithm intended for an arbitrary task graph structure/ size, it is not feasible to perform a (time-consuming) cycle-accurate simulation as in optiMap and hence the new bandwidth based cost function. Also, area is not part of the cost, since, the addition of more LPs always decreases the network area (as evident from the Xilinx synthesis results in Figure 4.6)

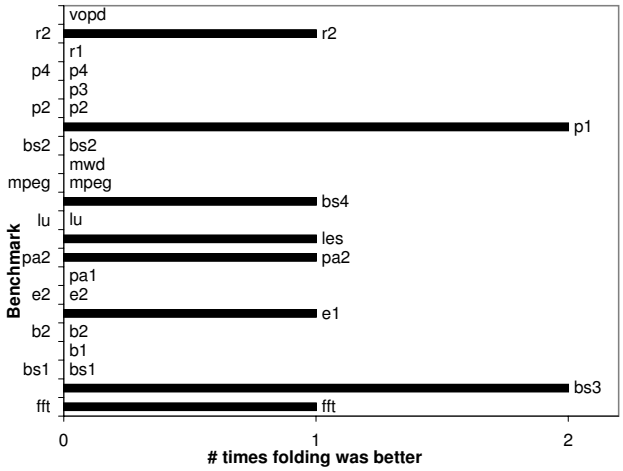


Figure 7.2: cMap: Results of Folding Phase

Folding

If the number of routers (k) is an odd number, then we are forced to choose a linear chain of routers (Figure 7.1 (a)). Hence, we introduce a phase called *Folding*. Here, we add an additional 1-LP router to mesh (*k+1 routers*), thereby, giving an opportunity for a folded mesh. For example, in Figure 7.1(b), the mesh is folded by half (having two rows). Then, we define all possible mesh configurations for both of the above cases (γ_1, γ_2) and repeat the following three phases. Figure 7.2 shows the number of times the folding was better, across all configurations ($1 \leq m \leq \#$ *cores*), thereby, validating the usefulness of *Folding*.

Core Placement

In a nearest-neighbor strategy, the start mesh location (where the first core is placed) determines the best possible mapping that can be obtained and this varies with the input task graph structure. Hence, to amortize the effect of the start location choice, we analyze the NoC configurations starting from each location of the current mesh configuration at hand. For each mesh configuration in the lists γ_1 and γ_2, starting from each location of the mesh configuration, we do a novel cost-based nearest-neighbor placement to achieve minimum traffic in the mesh (refer Steps 12-19, with lists L1 & L2 generated in Steps 2-5, in Algorithm 6).

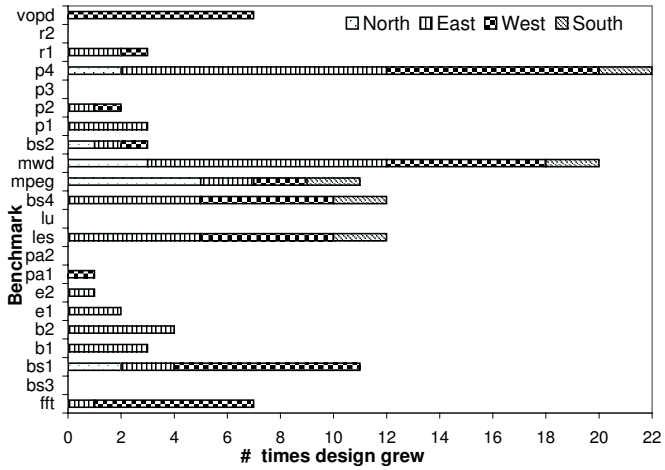


Figure 7.3: cMap: Design Evolve (Grow) Phase

Design Evolve

The results from optiMap demonstrate that the best NoC configuration need not necessarily have minimum number of routers. A best NoC configuration can have increased router count, provided it significantly reduced the overall traffic. Using this insight, we define a phase called *Design Evolve*, where the mesh size is allowed to grow. After obtaining the initial mesh and a placement, we employ a force-directed approach to expand the mesh along all four directions, as the cost gets reduced. Along each direction, if there are peripheral routers with $\leq m$ cores mapped, we move only one core (d=1) having the highest cost to the best possible location. Otherwise, we expand the mesh by adding routers (d = # columns for North/South expansion; d = # rows for East/West expansion) and move top d costly cores to the best possible location (in the d added routers). The best possible location is the router node where the cost is maximally reduced. Movement of the cores is subject to the global constraint, $1 \leq \#$ *Local Ports in any router* $\leq m$. Hence, if a chosen core (being moved) violates this condition, the next costlier core is chosen for movement.

Figure 7.3 shows the number of times the design grew for the 22 benchmarks, across all configurations ($1 \leq m \leq \#$ *cores*). We observe that for certain cases (*r2, p3, lu, pa2, bs3*), there is no evolution of the mesh. At the same time, we have benchmarks (*p4, mwd, mpeg, bs4, les, bs1*) evolving at least 10 times during the course of cMap algorithm, in finding the NoC configuration. Design growth occurs more along East/West than North/South. Apparently, from the minimum-sized starting mesh, East/West expansion needs addition of lesser number of routers than a North/South expansion and hence this behavior is justified.

Input: Given a system level task graph, $G(T, E)$ with *n* cores
Input: The placement constraint, routing constraint, I/O constraint and buffer constraint
Output: A near-optimal NoC configuration, such that $1 \leq$ *# Max. # LPs in any router* $\leq m$

*Analyze the constraint files and set Max # LP value (*m*)*
L1: List of nodes in descending order of *In-degree + Out-degree*
foreach *Node-L1: Node in the List L1* **do**
    L2: List of nodes that are connected to Node-L1, sorted in the descending order of the *Bandwidth (BW)*
end
Routers, $k = \frac{n}{m} \diamond \gamma_2 = \gamma_2 = \{\}$
γ_1: Generate all possible mesh connection topologies for the *k* routers
γ_2: If *k* is odd, increment the router count by one and create a folded mesh and generate all possible mesh connection topologies for *k+1* routers
repeat
    foreach *Loc: Location in the current mesh configuration* **do**
        Start.Mesh.Location $\leftarrow$ *Loc*
        repeat
            foreach Node-L1*: Unplaced node in List* L_1 **do**
                Perform a nearest-neighbor placement of *Node-L1*
                foreach *Node-L2: Unplaced node that is connected to* Node-L1 *in List* L_2 **do**
                    Perform a nearest-neighbor placement of *Node-L2*
                end
            end
        until *All cores are Placed*
        forall *Edges in Task Graph* **do**
            Best.Config.Cost += $\alpha\times$ Distance between the cores $\times$ BandWidth between the cores
        end
        repeat
            foreach *Peripheral Direction* (North/East/West/South) **do**
                if *Peripheral Routers have* $\#cores < m$ **then** d = 1
                else Add one row/column to the mesh along the respective direction (*Hence, d = 1 row (or) column length*)
                Move top *d* costly nodes to best available location along respective periphery
                $Current.Config.Cost \leftarrow$ *Cost(Current.Config)*
                if Current.Config.Cost $<$ Best.Config.Cost **then**
                    Best.Config $\leftarrow$ *Current.Config*
                    Best.Config.Cost $\leftarrow$ Current.Config.Cost
                end
            end
        until *Cost NOT reduced by Mesh Expansion*
        for *i=1* **to** *iter-max* **do**
            Current.Config: Initiate a random pair-wise swap of mapped cores
            if Current.Config.Cost $<$ Best.Config.Cost **then**
                Best.Config$\leftarrow$ *Current.Config*
                Best.Config.Cost $\leftarrow$ *Current.Config.Cost*
            end
        end
    end
until *All the mesh configurations in* γ_1 *and* γ_2*are evaluated*

Algorithm 6: cMap Algorithm

Design Perturb

In this final phase, we contemplate pairwise swaps for *iter-max* times so as to reduce the overall cost. This is a fine tuning phase performed to escape from the local maxima. At the end of this phase, the best Networks-on-Chip configuration found is returned.

7.2 Experiment Results

The cMap algorithm is a heuristic-based NoC configuration finder, that uses a force-directed approach to arrive at near-optimal NoC configuration [SV07a]. In most of the benchmarks, the result was obtained within a couple of seconds.

Table 7.1 shows the comparison of the costs (optiMap vs. cMap) of the NoC configuration of three selected benchmarks[†].

The NoC configuration produced by optiMap and cMap were back-annotated into optiMap framework to obtain the cost numbers. This was done because of the difference in cost function between the two algorithms. The average cost difference[‡] varied between 3% and 10%. For the 9-LP case, there is no difference in the results produced, since all the cores are mapped to a 9-LP router (hence, the cost difference = 0). If we are to ignore this redundant case, the average cost difference will increase slightly. Interestingly, cMap also found the *optimal NoC configuration* for the 7-LP and 8-LP case of packed-4. Note that the worst case cost difference[§] is not related to number of Local Ports and varies with the benchmark at hand.

[†]We compare the effectiveness of the proposed cMap algorithm with the exhaustive-search based optiMap algorithm. Note that this can be done only for task graphs with 9 nodes (optiMap limitation). For larger systems, there are no existing algorithms (heuristic, ILP, branch-and-bound or evolutionary strategies) for comparison in the literature that generate an MLPR-based NoC configuration. In fact, an ILP formulation of such an NP-hard problem (as explained in Section 7.1) is unrealistic, as scalability issues crop up and hence will run forever.

[‡]The cost of optiMap algorithm is clock latency and the cost of cMap algorithm is bandwidth. Hence, in order to compare the quality of results, the NoC configurations produced by optiMap and cMap are back-annotated into optiMap framework, thus comparing the clock latencies (Table 7.1)

[§]The percentage deviation in the results must be appreciated in conjunction with the fact that we are not performing an exhaustive-search that requires exorbitant amount of run-time/resources.

Max LP Count	pa2			p3			p4			r2			e1		
	optiMap	cMap	% Diff.	optiMap	cMap	% Diff.	optiMap	cMap	% Diff.	optiMap	cMap	% Diff.	optiMap	cMap	% Diff.
1	138	162	17.39	125	141	12.80	137	149	8.76	188	216	14.89	71	83	16.90
2	122	138	13.11	113	117	3.54	113	117	3.54	152	172	13.16	63	71	12.70
3	110	122	10.91	89	101	13.48	105	111	5.71	144	162	12.50	59	67	13.56
4	110	110	0.00	85	85	0.00	97	99	2.06	132	148	12.12	59	67	13.56
5	102	110	7.84	81	85	4.94	81	86	6.17	132	148	12.12	55	59	7.27
6	98	113	15.31	77	83	7.79	81	82	1.23	132	132	0.00	55	61	10.91
7	98	98	0.00	77	81	5.19	81	81	0.00	132	132	0.00	51	55	7.84
8	94	94	0.00	77	77	0.00	77	77	0.00	124	124	0.00	51	55	7.84
9	90	90	0.00	77	77	0.00	77	77	0.00	124	124	0.00	51	51	0.00
	Average % Diff.		7.17	Average % Diff.		5.31	Average % Diff.		3.05	Average % Diff.		7.20	Average % Diff.		10.07

Table 7.1: cMap vs optiMap: Cost Difference (in # clock cycles)

7.2.1 Effect of # LP

We present the variation of cost with the upper bound on the number of Local Ports in Figures 7.4 and 7.5 (for the sake of clarity). Overwhelmingly, the results favor the use of more # LPs. At the same time, increased LP count is not necessarily better always. For instance, in *MPEG4*, there is negligible variation from 6-LP to 9-LP case. We observe the same trend in MWD, between 8-LP and 10-LP case. Interestingly, in *VOPD*, the 4-LP cost is significantly more than the 3-LP case (mapping shown in Figure 7.12). A similar variation is observed for the 5-LP case in *MWD*. This behavior is highly dependent on the bandwidth and the intercommunication pattern of the benchmark. Hence, given an upper bound on the number of Local Ports, it is necessary to analyze all design possibilities with $\leq$ Maximum # LP, in order to arrive at the best possible NoC configuration. Overall, the cost is reduced by an average 74% across the benchmarks (Figures 7.4 & 7.5). This is the percentage reduction in the amount of traffic in the *main mesh*. In *mpeg*, with all 12 cores mapped to a single router, the cost is 0 (representing no traffic in *main mesh*, actually speaking, the mesh is absent!), thereby, indicating a 100% reduction. But, actually, there is point-to-point traffic inside the single router. If we are to take that traffic into account in the cost calculation, we get a 30.17% net reduction in total traffic (12 1-LPs vs. a 12-LP). Thus, to find the actual reduction in the overall traffic, one has to specifically calculate the cost of the actual mapping (NoC configuration) that was obtained for the given benchmark.

7.2.2 Mapping Results

Figures 7.6, 7.7, 7.8, 7.9, 7.10, 7.11, 7.12, 7.13, 7.14, and 7.15 present the mapping results of selected benchmarks. The usefulness of the *Design Evolve* phase is seen in both *les* and *MPEG4* (Figures 7.7 and 7.8). A closer look on the evolved design shows that a hand manipulation can provide a better design, with reduced router count. For example, in the 2-LP (non-folded) case of *les*, having cores 1 and 4 in the same router is obviously

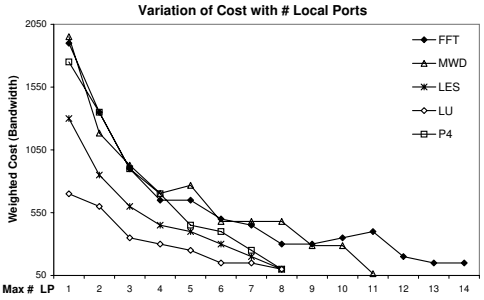


Figure 7.4: cMap Experimental Results I

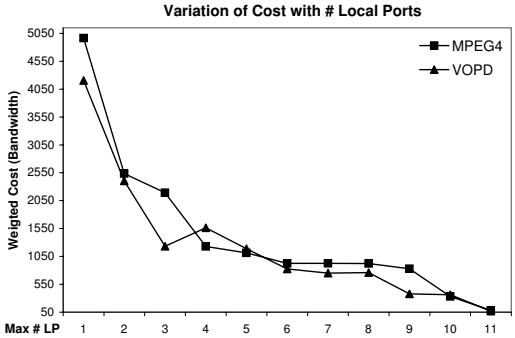


Figure 7.5: cMap Experimental Results II

better than the configuration shown. The absence of knowledge about the actual direction of growth during the initial placement is the reason. In this case (*les*), the core 1 was initially mapped to a separate router due to the global constraint, $1 \leq \#$ *Local Ports in any router* $\leq m$. During the course of cMap, the mesh grew along west direction, moving the core 6 to the left periphery (earlier it was with core 4) and finally, stopped at this configuration. The effect of folding on the final cost is seen in *lu* and *les* (Figures 7.7 and 7.7). We observe that folded design is not always better than an unfolded (achieving comparable costs in many cases). In any case, *folding* is a good design alternative to explore, in the pursuit of the best Network-on-Chip configuration.

Overall, the results of the cMap algorithm validate the merits of Multi Local Port Routers (MLPRs) in a Networks-on-Chip design, producing the near-optimal configurations within a couple of seconds [SV07a].

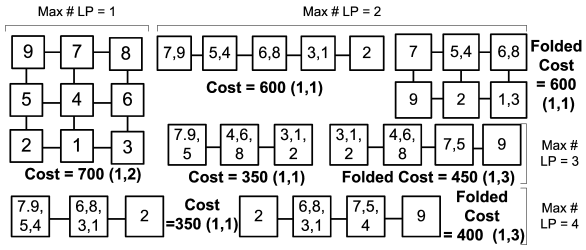


Figure 7.6: LU Decomposition *(lu)* NoC configuration with varying upper bound on # local ports(LP) - the mapped cores are inside square (router) *(shown till # LP = 4)*

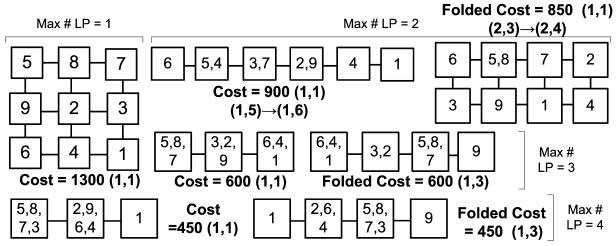


Figure 7.7: Laplace Equation Solver *(les)* NoC configuration with varying upper bound on # local ports(LP) - the mapped cores are inside square (router) *(shown till # LP = 4)*

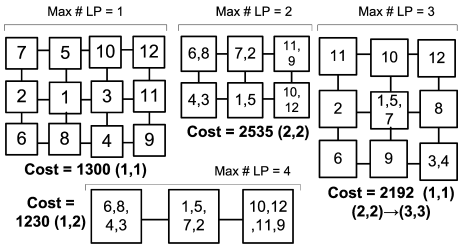


Figure 7.8: MPEG4 *(mpeg)* NoC configuration with varying upper bound on # local ports(LP) - the mapped cores are inside square (router) *(shown till # LP = 4)*

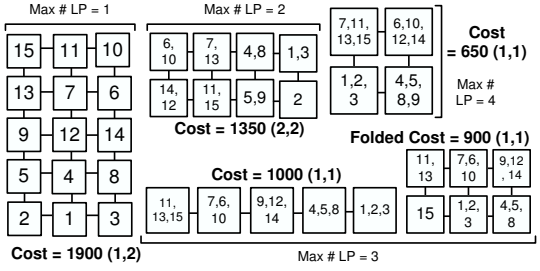


Figure 7.9: cMap: NoC configuration for FFT

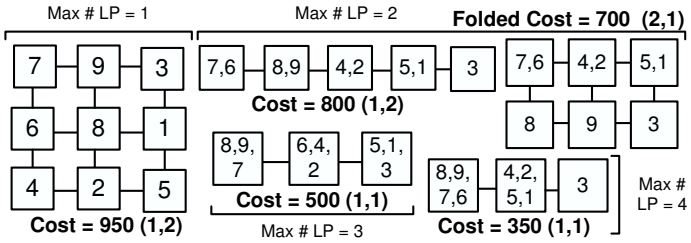


Figure 7.10: Parallel2 *(pa2)* NoC configuration with varying upper bound on # local ports(LP) - the mapped cores are inside square (router) *(shown till # LP = 4)*

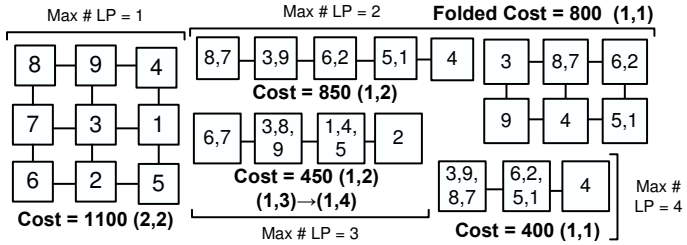


Figure 7.11: Extended1 *(e1)* NoC configuration with varying upper bound on # local ports(LP) - the mapped cores are inside square (router) *(shown till # LP = 4)*

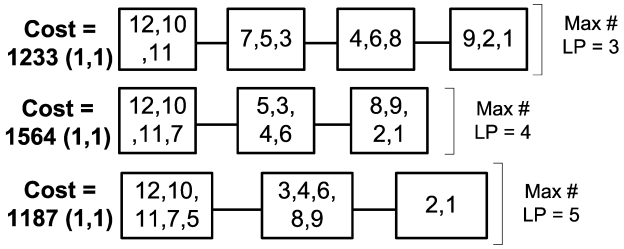


Figure 7.12: VOPD *(vopd)* NoC configuration with varying upper bound on # local ports(LP) - the mapped cores are inside square (router) *(shown till # LP = 4)*

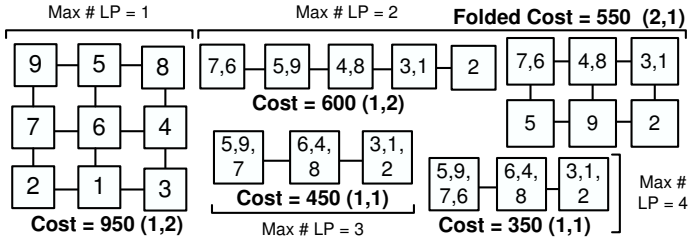


Figure 7.13: Random2 *(r2)* NoC configuration with varying upper bound on # local ports(LP) - the mapped cores are inside square (router) *(shown till # LP = 4)*

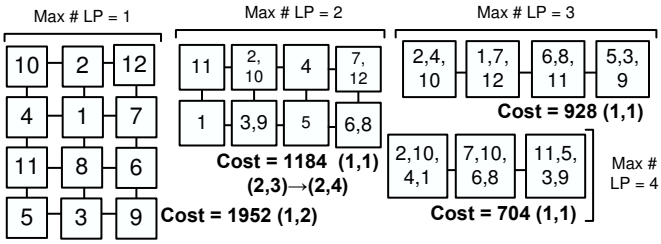


Figure 7.14: MWD *(mwd)* NoC configuration with varying upper bound on # local ports(LP) - the mapped cores are inside square (router) *(shown till # LP = 4)*

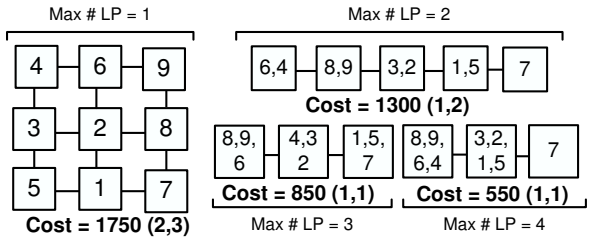


Figure 7.15: Packed3 *(p3)* NoC configuration with varying upper bound on # local ports(LP) - the mapped cores are inside square (router) *(shown till # LP = 4)*

7.3 Conclusion

To resolve the scalability issues encountered in the optiMap algorithm and to obtain the results in a fast manner, we present a fast mapping algorithm (cMap) for generating near-optimal Networks-on-Chip configurations using the Multi Local Port Routers. The cMap algorithm follows an heuristic approach to arrive at the results within a couple of seconds. We experiment with a wide set of benchmarks and report the results. The results obtained from the cMap algorithm come within the acceptable range compared to the optimal results obtained from the optiMap algorithm. There is a scope for improvement of the heuristic so as to reduce the average percentage cost difference from an optimum Networks-on-Chip design.

Chapter 8

Multi^2 Router

Multi Local Port Router (MLPR) is a design alternative for the traditional NoC design and offers various gains including bandwidth optimization and reduced network area & power consumption, resulting eventually in an efficient Networks-on-Chip. Till date, communication in router-based NoCs have no scope of multicasting, an effective feature available in bus-based systems. In this chapter, we present a novel approach to incorporate the multicast feature into the routers in an MLPR-based Networks-on-Chip. Ideas explained in this research are equally pertinent and applicable to NoCs realized on ASICs.

8.1 Multicast Feature

Most practical applications including image processing, multimedia and Digital Signal Processing (DSP) have high degree of parallelism and involve communicating data to multiple nodes in the system [Sod03]. An in-depth look at the different benchmarks presented in [SV06b] provide a good idea of the level of out-degree prevalent in the task graphs. Further applications like DCT & Fast Fourier Transform [LSEV99] and FIR filters [DKSL04, YN01, CC05, Eva94] have multicast nature. Multicasting is a feature by which data is transferred to multiple nodes, in one-shot. A neural network is a classic example having a complex interconnection structure [LTXW05, YP03] and has received considerable attention [CLP00, TLV+04, GCBM00, SHSM02].

Bus is a well-researched interconnection architecture having a variety of data transfer capabilities [HK03, Cor99, DSY96, MT99, CCCS90, THT+97, GC98, KDHDS]. They

are very efficient for point-to-point communication and transferring multicast messages (with/without priority) [EM96,KCPP00,KR87,Sto83,CKPP99,DEW99,BOSW94]. But, it suffers from the under-utilization of bandwidth because of static allocation nature. Networks-on-Chip (NoC) help in the efficient usage of the bandwidth available across the network. In an NoC, multicasting is an entirely novel feature. This is because the traditional single local port router based NoCs can provide only point-to-point packet transfers, thus, removing any possibility of multicasting in NoCs.

Multi Local Port Routers (MLPRs) have been introduced as a architectural alternative in place of the traditional NoC design, with the aim of improving the overall performance of the system. MLPRs provide a host of advantages including reduction in area, power, transit time & congestion, and most importantly, the optimization of the bandwidth, thus, resulting in an efficient and high performance NoC design [SV06b]. With the novel MLPR architecture, multiple cores can be mapped to the same router, thus, providing an opportunity to multicast to all the cores attached to the same MLPR. This is because the packets reaching a particular router node have the same [X,Y] mesh co-ordinate value, irrespective of the destination local port. The proposed architectural modification is illustrated for a five port Multi Local Port Router in Figure 8.1. Since the data transfers are distributed in nature, multicast can happen only between co-operating router pairs. In an NoC mesh, *universal broadcasting will translate to flooding of packets* in all possible directions, thus, requiring more complex routing algorithms and flow control. Moreover, the problem of congestion is compounded, and hence is highly impractical in NoCs.

In this chapter, we propose an architectural modification for incorporating the multicast feature in the routers of a Networks-on-Chip implemented on FPGAs. We discuss in detail the modified decoding scheme & the crosspoint matrix that result in reduced logic usage.

8.2 Related Work

In the Reconfigurable Computing platform, researchers have tried to realize efficient router designs [ZSS04, SBKV05, ZKS04, ZSS04, MdMM+03]. LiPaR is an efficient router design fine-tuned towards Xilinx based FPGAs [SBKV05] and can establish connections between different channel pairs simultaneously. The authors in [SV06b] introduce an innovative concept of using Multi Local Port Routers (MLPRs) to improve the NoC design.

For System-on-Chip designs, bus based interconnect architectures [AMB06,STB06, Arc05, wis06, Son06] are popular and well-studied. In the literature, we have several efficient designs and protocols for bus-based design. A mesh network containing array processors with separate buses is presented in [KR87]. [KCPP00] introduces the concept of Virtual Bus (VB) that is dynamically built on the communication links of the mesh network. Various point-to-point links of the mesh (row/column) are used to setup real bus. The authors extend their work to present a bandwidth efficient implementation of mesh with multiple multicasting [CKPP99]. [Cor99] presents a bus architecture with enhanced reusability and customizability for System-on-Chip (SoC) designs. Efficient multicasting techniques for constrained reconfigurable meshes are presented in [DEW99]. There are efficient protocols for multicasting in a bus based design [CCCS90, BOSW94].

To to the best of our knowledge, this is the first work to introduce the multicast feature into the routers that form the backbone of the Networks-on-Chip for implementing SoC designs on FPGAs.

8.3 Multi^2 Router Architecture

Multi Local Port Router is an innovative architecture design, wherein a router in an NoC mesh, handles more than one logic core simultaneously, without any performance penalty [SV06b]. The MLPR NoC architecture has deterministic XY routing and store-and-forward flow control in a mesh structure. MLPRs primarily aim to replace the inter-router-channel communication with the intra-router-channel communication. As a general case, an n-LP router having 4 directional ports is capable of establishing n+4 parallel connections simultaneously (Fig. 8.2). MLPRs offer a host of advantages including reduction in area (maximum of 47.5% on $XC2VP30$ FPGA), power, congestion & transit time, improved mesh design and most importantly, optimization of bandwidth usage. The issues that may limit the maximum number of Local Ports (LPs) include the critical path, routing resource availability, buffer, logic & I/O requirements and the effective address space [SV06b].

An important observation to be noted in case of the novel MLPR architecture is that the packets reaching a particular router node have the same [X,Y] mesh co-ordinate value, irrespective of the destination local port. Thus, the various cores mapped to the same router provide an excellent opportunity to multicast to all the cores attached to the same MLPR. The proposed architectural modification is illustrated for a five port Multi Local Port Router

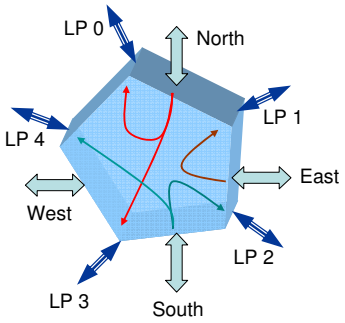


Figure 8.1: Multicasting Illustration in a 5 port Multi^2 Router

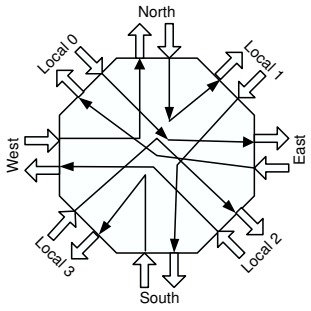


Figure 8.2: Multi^2 Router having 8 parallel connections

in Figure 8.1.

In this section, we present an innovative router design with multicast capability, called as Multi^2 Router (*Multi*cast *Multi* Local Port Router). We improve upon the existing MLPR design and implement major modifications to the header flit and the corresponding decoding logic.

8.3.1 Addressing

Figure 8.3 shows the header packet in the various schemes. Below, we discuss the two addressing schemes that were contemplated and justify the choice of a particular scheme over the other.

Contiguous Scheme

Figure 8.3(a) shows the header flit of an MLPR, where LID uniquely identifies a *single* destination Local Port (LP). In order to achieve multicasting, the header flit is modified as shown in Figure 8.3(b). Here, $sLID$ identifies the start Local Port ID and

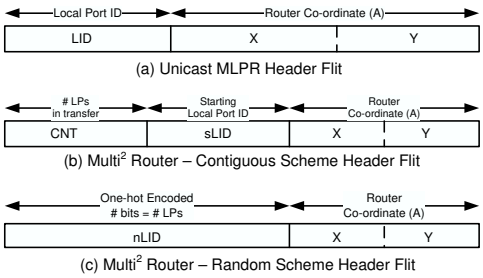


Figure 8.3: Header Flit

CNT specifies the # LPs (from the start Local Port, LID) that receive the multicast data. Theoretically, # bits of $sLID$ must be equal to # bits of CNT, in order to be able to multicast to all the Local Ports present in an MLPR. This scheme is capable of multicasting to CNT contiguous Local Ports, starting from any desired Local Port. For example, if $sLID = 1$ and $CNT = 2$, the data will be multicast to 2 contiguous LPs, starting from $LP1$ (i.e., $LP1$ and $LP2$). The architecture is also capable of *wrapping around* while multicasting. For example, in a 4-LP router, if $sLID = 3$ and $CNT = 3$, LP3, LP0 & LP1 will receive the data in a multicast fashion, simultaneously. The major handicap of this scheme is the contiguity constraint. Let us assume a case where the data is to be multicasted to LP0 and LP2. Here, multicasting is not possible because of the non-contiguity of the participating Local Ports. Hence, a data transfer in terms of multiple point-to-point communication is forced. Also, this scheme requires implementation of costly comparators for identifying the destination Local Port(s). Comparators are required to check if a particular LP is within the range of $\{sLID, sLID + CNT\}$. Above all, the *contiguous* constraint creates a severe handicap on the mapping algorithm by eliminating many potentially optimal combinations [SV06b].

Random Scheme

We overcome the contiguous constraint in this scheme, by employing one-hot encoding for the identification of the Local Port IDs that are to multicasted. Figure 8.3(c) shows the modified header flit for proposed architecture. Here, each bit of $nLID$ identifies a unique Local Port and hence presence of 1 will indicate if the data packet is destined for a particular Local Port. Summary of the merits of the scheme are,

- This is a simple scheme wherein any combination of Local Ports can receive the multicast data.

# LPs	Scheme - # Bits required	
	Contiguous	Random
n	$2 \times \lceil \log_2(n) \rceil$	n
2	2	2
4	4	4
6	6	6
8	6	8
10	8	10
12	8	12
14	8	14
16	8	16
32	10	32
64	12	64

Table 8.1: Comparison of # bits required

- The expensive comparison operation is removed and the decoding logic inside the router is greatly simplified. This is due to the fact that the destination Local Port(s) identification information is readily available from the header flit itself. This results in the reduced number of CLB usage and reduced latency in connection establishment.
- The mapping tool is provided with a large configuration space for choosing the optimal mapping, thanks to the random multicast capability [SV06b].

The flip side of this scheme is the considerable increase in the # of bits for LID identification, with the addition of more number of Local Ports (refer Table 8.1). The LID representation format (binary for *contiguous* scheme & one-hot for *random* scheme) is the reason behind this disparity. Hence, the random addressing is preferred for a typical system having an average of 10 communicating cores [SV06b]. For larger systems, the use of the contiguous scheme is to be contemplated because of the aforementioned reasons.

8.3.2 Modified Architecture & Decoding Logic

We use LiPaR [SBKV05] as the base system and implement changes to the architecture. The inter-router-channel transfers and the transfers between the Local Port channels & the Network Interface (NI) remain similar to LiPaR. The new decoding scheme is implemented after the data packet arrives at the input channel of various ports (directional & local). The Request signal (REQ) is increased to be 2 bits wide (in place of a 1 bit line in LiPaR), the combinations of which indicate the type of transfer (Table 8.2).

Figure 8.4 shows the architecture of the modified input channel of various ports. The $START_{TRANSFER}$ signal is the `AND` of several 2-input `OR` logical operations. One input of the `OR` function is the GNT signal, set (granted) by the output channel of other

```
if Multicast Transfer (≥ 1 Destination LP) then
    Send REQ = 10 to the destination Local Ports
    Set the CTRL lines of non-participating LPs to 1
    if START_TRANSFER = 1 then
        Set REQ = 11 indicating start of transfer from next cycle
        On completion, Reset REQ = 00 indicating end of operation
        Wait for data for new data packet
    else
        Wait for GNT from the output channels of the requested LPs

else if UniCast Transfer (inside/outside router) then
    Send REQ = 01 to the destination output channel
    Set the CTRL lines of other (non-requested) output channels to 1
    if START_TRANSFER = 1 then
        Set REQ = 11 indicating start of transfer from next cycle
        On completion, Reset REQ = 00 indicating end of operation
        Wait for data for new data packet
    else
        Wait for GNT from the requested output channels

end
```

Algorithm 7: Pseudo Code of Input Channel FSM

ports. The second input is the locally generated $CTRL$ signal, which is set high if a particular output channel was *not* requested for channel-access. This is due to the obvious fact that a non-requested output channel will not set the GNT signal for a given input channel. The logic function, AND{ORs($CTRL$, GNT) signals} indicates the actual start of the transfer ($START_{TRANSFER}$). When $START_{TRANSFER}$ = 1, the input channel sets REQ = 11, thus, indicating to all the (requested) output channels that the data transfer is set to start from the following cycle. The transfer continues until REQ becomes 00.

The pseudo code of the new FSMs at the input and output channels are given, respectively, in Algorithms 7 and 8. During the channel transfers, the multicast transfers are given precedence over the unicast transfers. And, there is no distinction in the precedence when multiple multicast happen simultaneously inside a single router. In other words, the multicast requests are granted access on a first-come first-serve basis.

Additionally, we remove the entire block containing the DEMUXs from crosspoint matrix present in an MLPR [SV06b] and directly connect the output of the various input channels to the MUXs of the crosspoint matrix. Figure 8.5 shows *Connection Matrix* in the modified architecture. The connection to various MUXs is governed by positional association. That is, the output of a particular input channel is port-mapped to the same input position (1st, 2nd,...) of the various MUXs, thus, maintaining the uniformity across all the channels. These architectural optimizations help to reduce the CLB usage and the latency

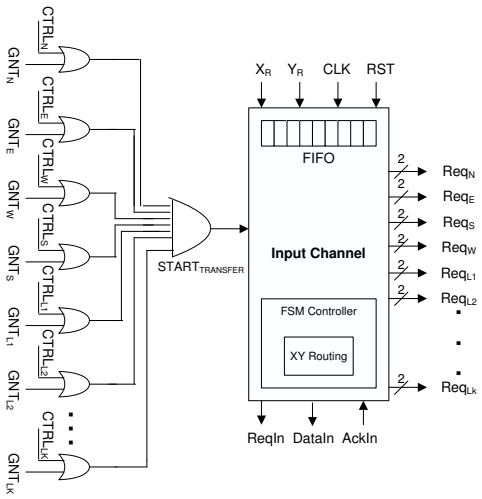


Figure 8.4: Modified Input Channel

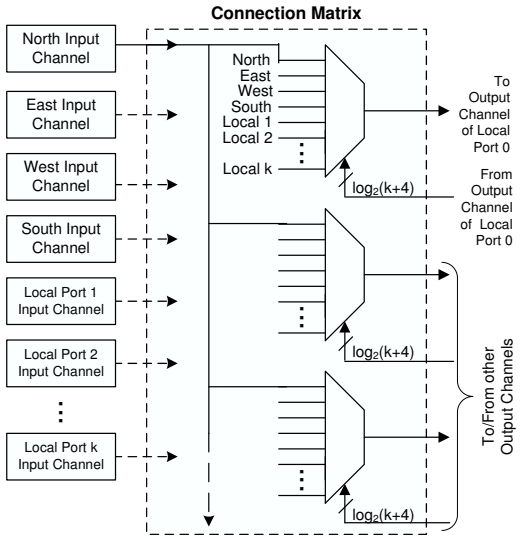


Figure 8.5: Modified Crosspoint Matrix

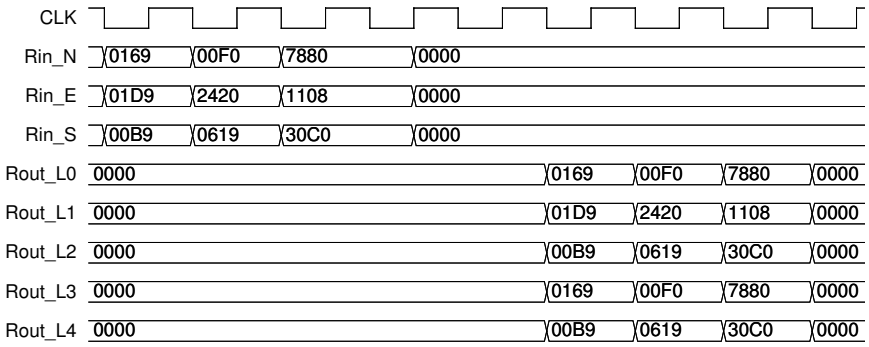


Figure 8.6: Simulation with simultaneous multicasts

if *Free to receive data* **then**
  if REQ_i = 10 *(Multicast Transfer)* **then**
    Follow the dynamic-fixed priority scheme [SBKV05]
    Set GNT_i = 1 and wait for REQ to become 11
    Set the select signals of the MUXs appropriately
    Enable the FIFO to receive the data
    On completion of transfer (indicated by REQ = 00), Disable the FIFO & Reset the select signals of MUX
  else if REQ_i = 01 *(Unicast Transfer)* **then**
    Follow the dynamic-fixed priority scheme [SBKV05]
    Set GNT_i = 1 and wait for REQ to becomes 11
    Set the select lines of the MUX appropriately
    Enable the FIFO to receive the data
    On completion of transfer (indicated by REQ = 00), Disable the FIFO & Reset the select lines of MUX;

end
else
  Wait for the destination output channel to become available

Algorithm 8: Pseudo Code of O/P Channel FSM

REQ	Type of operation
00	End-of-transfer/No Operation
01	Channel Request (unicast)
10	Channel Request (multicast)
11	Start Transfer (beginning next cycle)

Table 8.2: Modified Request (REQ) Signal

of connection establishment, thus, improving the overall performance of the system.

On completion of the data transfer, the input channel sets the REQ signal to 00 and the FIFO output lines to high impedance state. Also, the output channel stops the FIFO transfer by appropriately tristating the *Select/Enable* lines, thus removing any possibility of corrupt data being sent/received. The architecture of the output channel is similar to LiPaR [SBKV05], except for the removal of output buffer and the new priority-based control logic to set MUX lines.

Multicast can happen from one local port to multiple local ports present in the same router node, thus, giving a *versatile* multicast capability. Since, a multicast represents the data transfer to output channels of the local ports (inside a single router), the directional ports do not participate in the multicast. This is because a data packet arriving at the output channel of a directional port (actually, passing through) indicates that the packet is addressed to a different router node. Hence, multicast is restricted between router pairs or within a router. Universal broadcasting will result in flooding of the network with packets

# LPs	# slices (% in XC2VP30)	
	Multi^2 Router	Unicast MLPR
1	446 (3%)	461(3%)
3	1202(8%)	1015(7%)
4	1838(13%)	1419(10%)
5	2370(17%)	1759(12%)
7	4004(29%)	2866(20%)
9	5354(39%)	3753(27%)

Table 8.3: Xilinx ISE synthesis - Area results in $XC2VP30$

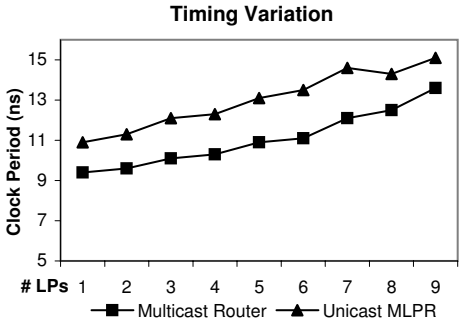


Figure 8.7: Variation in clock period

and hence is impractical for mesh-based NoCs.

8.4 Synthesis & Simulation Results

8.4.1 Synthesis Platform

The Multi^2 Router designs with varied number of Local Ports are coded in VHDL and synthesized using Xilinx ISE 6.3i [Xil06a]. Modelsim $5.8c$ [Men07] is used to simulate the model and generate activity data (Value Change Dump) of the Placed-And-Routed (PAR) models. The FloorPlanner tool of the Xilinx ISE $6.3i$ is used to implement placement constraints on the NoC system. Xpower tool of the Xilinx ISE $6.3i$ is used to obtain the power estimates. The $ML310$ board provided by Xilinx is used to functionally verify the various versions of the stand alone router and the NoC system. The test board has Virtex II Pro family ($XC2VP30$) of FPGA [Xil06a].

The synthesis results of the various Placed-And-Routed (PAR) designs are given in Table 8.3. Figure 8.7 shows the variation in the clock period for both the (original) unicast router designs and new Multi^2 Router designs. Though the overall variation in the clock period is linear, an interesting point to be noted is the bump in the clock period of the router designs with odd number of local ports. Especially, the 7 LP version is worse compared

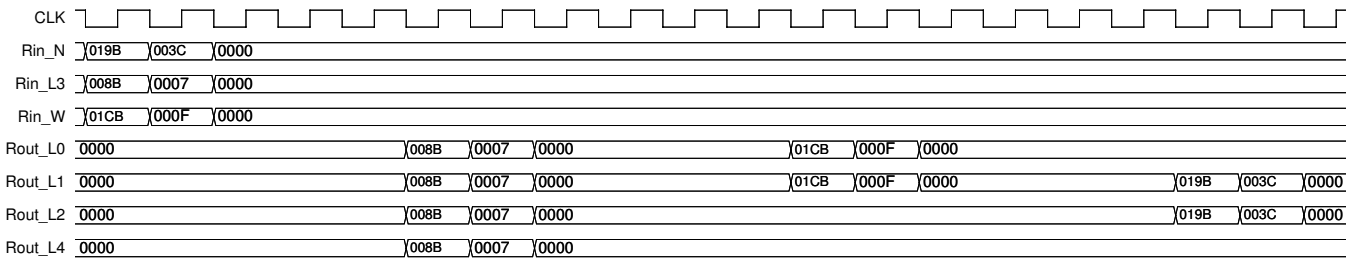


Figure 8.8: Simulation with Input Channel(s) waiting for channel-access to multicast

to the 8 LP router. It is to be noted that the clock period of the Multi^2 Router is better than original MLPR, but, consumes more area. The variations in the logic area usage are attributed to the following reasons.

- The width of data packet is increased to 16 bits (from 8), in order to accommodate the extra bit requirements for implementing the multicast feature. This translates to accretion in additional logic & routing overheads.
- The synchronization signal REQ is now a 2 bit signal.
- The added decoding logic & signals($START_{TRANSFER}$, $CTRL$) to achieve multicasting.
- Modification of the output channel FSM to implement the prioritized multicast, in addition to the unicast transfers.

Simulation: Figure 8.8 shows the simulation wherein the input channel of LP3 multicasts to LP0, LP1, LP2 & LP4, the input channel of West directional port multicasts to LP0 & LP1, and the input channel of North directional port multicasts to LP1 & LP2. In this case, not all the requested (required) local ports are free to receive the multicast data. Hence, the requesting input channel has to wait till all the requested output channels become available to participate in the multicast transfer. Here, initially, the local port LP3 multicasts to local ports, LP0, LP1, LP2 & LP4. When the local ports LP0 & LP1 become free, the West directional port multicasts to LP0 & LP1. Later, when LP1 is ready to receive data, the North directional port multicasts to LP1 & LP2.

Figure 8.6 presents a case showing simultaneous multicast between different channels. Here, the North directional ports multicasts to local ports, LP0 & LP3, the South directional port multicasts to LP2 & LP4 local ports, and the East directional port performs a single transfer to LP1, all happening simultaneously. The ability to multicast to any

combination (non-contiguous random scheme) of local ports can be clearly observed here. Thus, we verify the functional correctness of the proposed design.

8.5 Conclusion

We present a novel approach of introducing multicast capability into the router elements in a Networks-on-Chip design, thus, bridging the gap between NoC and shared-bus in terms versatile feature availability. We present the modified architecture for the new router design and report the synthesis and simulation results.

Chapter 9

Energy Efficient NoC Configuration

A Networks-on-Chip having Multi^2 Routers provides a many-fold advantage, eventually improving the power and the performance of the system. The scheme significantly reduces the data traffic in the network by exploiting the multicast capability in place of the traditional unicast transfers. A complex environment having MLPRs with multicast ability exacerbates the process of finding the efficient mapping of design cores in a System-on-Chip and hence demands an efficient NoC architecture generation algorithm. Based on modified optiMap algorithm [SV06b], we experiment with a variety of benchmarks and exhaustively analyze the performance and power gains obtained by exploiting the multicast feature.

9.1 Advantages of the Multicast Router

The salient advantages of the proposed approach are summarized as follows.

- There is a marked reduction in the total number of clock cycles taken for data transfer across the system task graph, due to reduced packet flow.
- Reduction in the # of packets flowing across the network will help ease congestion and hence helps to remove the contentions for channel access to a great extent.
- The bandwidth across the links is not wasted for doing multiple unicasts, thereby, improving the effective bandwidth available for the (other) data transfers.
- Every channel hop of the data packet adds to the dynamic power consumption of the system. Thus, reduced number of packet transfer between channels provides

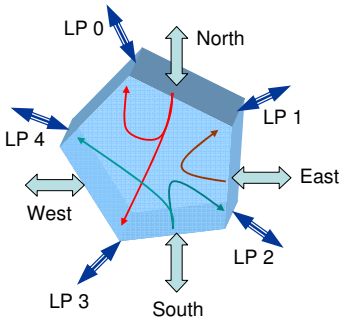


Figure 9.1: Multicasting Illustration in a 5 port Multi^2 Router

an excellent opportunity to reduce the overall power consumption in the network, drastically.

9.2 μMap Algorithm

The factors discussed above provide the basis for an energy-efficient NoC architecture generation. We envisage to arrive at an Network-on-Chip topology and a corresponding mapping that is optimum in terms of the energy-efficiency and performance. We use the optiMap [SV06b] NoC configuration generator as the base system for implementing the μMap Algorithm. We implement the changes and develop the modified algorithm called μMap (**Mu**lti^2 router **Map**) to generate the various energy-efficient Networks-on-Chip configurations. We modify the cycle-accurate simulator for the new scenario involving multicast routers. We explore all possible combinations of multicast based transfer in order to generate the various power-efficient Networks-on-Chip configurations.

In a system task graph, when a set of cores are the children of the same parent and if those the (child) cores are mapped to a single router in the current NoC configuration, there exists a possibility of receiving the same data simultaneously. Accordingly, we modify the cost calculation so as to send the data packet once, thus, effectively exploiting the multicasting to the maximum. The above idea is illustrated in Figure 9.1, wherein simultaneous multicast are shown in a five port Multi^2 Router.

The μMap algorithm is shown in detail in Algorithm 9. It is an exhaustive search algorithm analyzing all possible configurations in forming an NoC mesh (Lines 3-27). A Multi^2 Router can have varying number of ports ranging from a single port (in addition to the North/East/West/South ports) to a maximum port count equal to the number of nodes

(n) in the task graph. Hence, the topology can have varying partition count In effect, the different partition count reflects on the number of routers for the current iteration (Line 4-5). For a given partition count, different mesh combinations are possible, ranging from a linear chain (eg. $1 \times n$ or $n \times 1$ mesh) to a square mesh ($m \times m$), with the other rectangular combinations forming the rest (Line 6). Once the topology is in place, the n cores can be mapped onto the n ports (distributed based on the partition count and mesh topology) in $n!$ way (Line 7).

After the formation of mesh topology and the corresponding mapping (for the current iteration), we find the top hop count based on the mapping of the task graph nodes. In a system task graph, when a set of cores are the children of the same parent and if those the (child) cores are mapped to a single router in the current NoC configuration, there exists a possibility of receiving the same data simultaneously. Accordingly, we modify the cost calculation so as to send the data packet once, thus, effectively exploiting the multicasting to the maximum (Lines, 9-16). The hop count (estimated by the manhattan distance because of XY routing scheme) is translated into the clock cycles needed which in turn is dictated by the flits per packet (assumed to be 5 in this work). In Line 13, the last term of 1 is necessary because even in case of the source and destination ports present within the same router, at least once hop is necessary to complete the transfer. The value is weighted to account for the difference in the arbitration cycles at various ports, which is dictated based on the NoC configuration at hand. We implement a simple C++ program to perform cycle accurate simulation for the new scenario involving multicast routers. This step is necessary to find the queue times that may be arise because of the blocking at the various channels and due the different arbitration times due the round-robin policy (Lines 17-19). The configurations are stored in a greedy fashion (Lines 20-22). This process is repeated for all possible configurations formed using various Multi^2 Routers.

We experiment with a set of eighteen benchmarks presented in [SV06b]. The benchmarks cover a variety of task structures commonly found in a multiprocessor environment. The modified mapping algorithm (μMap) is written in C++ using the Standard Template Library (STL) vectors and dynamic arrays. The μMap algorithm is executed on a SunBlade 1000 workstation having dual processors operating at $750MHz$ and $2GB$ RAM. Because of the exhaustive search of the complete search space (factorial order, O(n^n)), the average execution time varied between 5 and 6 hours for the various benchmarks.

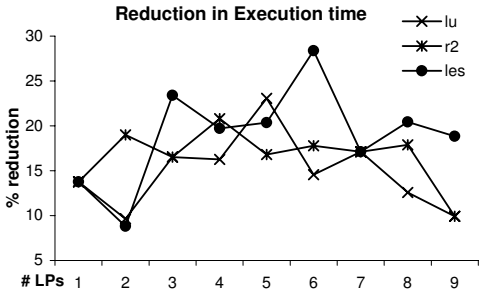


Figure 9.2: Results 1 - Reduction in the overall execution time

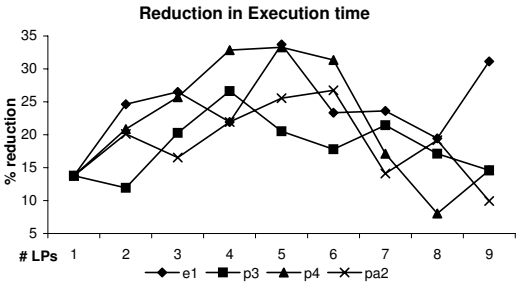


Figure 9.3: Results 2 - Reduction in the overall execution time

9.3 Experimental Results

The NoC configuration generation process involves an exhaustive search of the entire search space for each of the benchmark in the set. By virtue of being exhaustive in nature, the optimal solution is guaranteed. Hence, understandably, the μMap algorithm caches the optimal results in a greedy fashion. We present and exhaustively discuss the results of selected benchmarks below.

9.3.1 Packet Reduction

From Table 9.1, across the benchmarks, the average packet reduction using Multi^2 Routers is 30%, compared to the *respective* unicast MLPR designs. In benchmark *p3*, the packet reduction is as high as 51.3% for the 8 & 9 LP routers (average of 44.6% across all nine router versions). From Table 9.2, we observe that, compared to the traditional 1 LP router, the average packet reduction is close to 50% (across all benchmarks) by using larger LP routers. The maximum reduction is to the tune of 74% in benchmark *p3*, which is quite significant. The drastic reduction in the total packet count exemplifies the merit of the proposed approach and provides a firm basis for further exploration in the pursuit of an

# LPs	e1			p3			p4			pa2			r2			lu			les		
	u	m	% Diff	u	m	% Diff	u	m	% Diff	u	m	% Diff	u	m	% Diff	u	m	% Diff	u	m	% Diff
1	656	656	-	1040	1040	-	1072	1072	-	704	704	-	544	544	-	560	560	-	576	576	-
2	560	376	32.9	896	648	27.7	928	712	23.3	592	504	14.9	480	432	10.0	496	448	9.7	528	472	10.6
3	496	336	32.3	800	512	36.0	800	528	34.0	544	440	19.1	432	368	14.8	448	376	16.1	480	368	23.3
4	464	280	39.7	752	432	42.6	768	448	41.7	512	400	21.9	416	352	15.4	432	344	20.4	480	376	21.7
5	432	264	38.9	720	384	46.7	720	408	43.3	480	384	20.0	384	336	12.5	400	328	18.0	448	352	21.4
6	432	272	37.0	688	344	50.0	672	392	41.7	480	368	23.3	384	312	18.8	400	328	18.0	432	320	25.9
7	416	248	40.4	672	328	51.2	656	376	42.7	464	344	25.9	368	296	19.6	384	312	18.8	432	336	22.2
8	400	232	42.0	640	312	51.3	640	360	43.8	448	312	30.4	368	296	19.6	384	296	22.9	416	304	26.9
9	384	216	43.8	608	296	51.3	608	344	43.4	416	296	28.8	352	280	20.5	352	280	20.5	384	288	25.0
	Avg. % Diff. = 38.4			Avg. % Diff. = 44.6			Avg. % Diff. = 39.2			Avg. % Diff. = 23.0			Avg. % Diff. = 16.4			Avg. % Diff. = 18.0			Avg. % Diff. = 22.1		

Table 9.1: Packet count in multicast (m) and unicast (u) transfers (optimal result chosen from respective cases)

energy-efficient Networks-on-Chip.

9.3.2 Performance Gain

Multi^2 router also provides a significant reduction in the overall execution time*. Figures 9.2 and 9.3 show the percentage reduction in the execution time (in # clock cycles) in the Multi^2 Router compared to the respective unicast MLPR. The gain in the execution time varies over a range (minimum gain of 7% and a maximum gain of 34%). Across most of the selected benchmarks, large gains are observed for the router designs having 4 to 7 local ports. The gains of the 9 LP routers match the gains of the traditional 1 LP routers, indicating the point of diminishing returns in terms of performance, as we increase the LP count of a single router. But, in any case, the reduction in area is significantly large for a router with larger LP count [SV06b].

9.3.3 Optimization Cases

We define two optimization cases for the μMap algorithm. In the first case (EXE), the total execution time of the task graph is optimized, while, in the second (PKT), the total packet count (including both inter-router-channel and intra-router-channel) is optimized. Table 9.2 lists the results of both optimization cases, for each of the seven selected benchmarks.

*Refers to the completion (end) time of the last node in the task graph

```
Input: Given a system level task graph, G(T, E) with n logic cores
Output: Power Efficient NoC configuration
Best.Config.Cost ← ∞
Best.Config ← NULL
repeat
  forall Partition configurations with Partition Count ≤ max{ Local Port Count} do
    # Router ← Current Partition Count
    forall Mesh topologies for the the given Partition configuration do
      forall Combinations of mapping of cores onto the given Mesh topology do
        Total.Hop.Count ← 0
        foreach Node (n1) in the Task graph do
          (i1, j1) ← Router Coordinate of Node n1
          repeat
            □: Set of child nodes with common router co-ordinate (i2, j2)
            Total.Hop.Count ← Total.Hop.Count + i1 ∼ i2 + j1 ∼ j2 + 1
            Mark the child nodes in □ as visited
          until All the child nodes of the given node are analyzed
        end
        Perform a cycle-accurate simulation of the task graph
        Wait.Time ← Sum of all queue times at various ports
        Current.Cost ← α× Total Hop Count ×# Flits per packet + Wait.Time
        if Current.Cost < Best.Config.Cost then
          Best.Config ← Current.Config
          Best.Config.Cost ← Current.Cost
        end
      end
    end
  end
until All the configurations are evaluated
```

Algorithm 9: μMap Algorithm

Case 1: EXE

In the EXE case, against the usual trend, there is a jump in the total number of packets as we increase the LP count, which is quite noteworthy. This is particularly evident in the benchmarks *lu* (4LP $>$ 3LP, 7LP $>$ 6LP)[†] and les (6LP,7LP,8LP, all three $>$ 5LP). This is because of the lesser out-degree level prevalent in those benchmarks. Further, the EXE optimization case searches for the NoC configuration with the minimum execution time, which need not necessarily have the least packet count for the complete transfer.

Case 1: PKT

The PKT optimization case involves a completely different optimization scenario. An important point to be noted here is that a Network-on-Chip configuration that is opti-

[†] *n*LP refers to a *n* LP router.

# LPs	p3				p4				pa2				lu				les				e1				r2			
	EXE		PKT		EXE		PKT		EXE		PKT		EXE		PKT		EXE		PKT		EXE		PKT		EXE		PKT	
	exe time	pkt count	exe time	pkt count	exe time	pkt count	exe time	pkt count	exe time	pkt count	exe time	pkt count	exe time	pkt count	exe time	pkt count	exe time	pkt count	exe time	pkt count	exe time	pkt count	exe time	pkt count	exe time	pkt count	exe time	pkt count
1	125	1136	129	1040	137	1120	133	1072	138	752	150	704	134	592	138	560	113	576	113	576	71	656	71	656	188	736	188	544
2	101	760	113	648	101	776	109	712	118	576	126	504	114	488	134	448	105	528	117	472	59	416	63	376	152	472	164	432
3	85	528	85	512	97	528	97	528	110	512	118	440	105	440	114	376	89	368	89	368	55	344	59	336	144	408	148	368
4	85	464	85	432	81	448	81	448	106	440	110	400	102	384	106	344	93	376	93	376	51	312	55	280	132	368	140	352
5	81	424	85	384	81	432	81	408	102	480	102	384	98	328	98	328	85	360	89	352	47	264	47	264	132	336	132	336
6	73	408	81	344	81	392	81	392	98	424	98	368	98	352	106	328	81	320	81	320	47	272	55	272	132	312	132	312
7	73	344	73	328	81	376	81	376	94	368	114	344	94	352	98	312	85	344	89	336	47	248	47	248	132	296	132	296
8	73	312	73	312	77	400	81	360	94	352	98	312	94	352	98	296	81	304	81	304	47	232	47	232	124	296	124	296
9	73	296	73	296	73	344	73	344	90	296	90	296	90	280	90	280	73	288	73	288	39	216	39	216	124	280	124	280

Table 9.2: Comparison of overall execution time (in # clock cycles) and packet count for two cost functions (EXE & PKT)

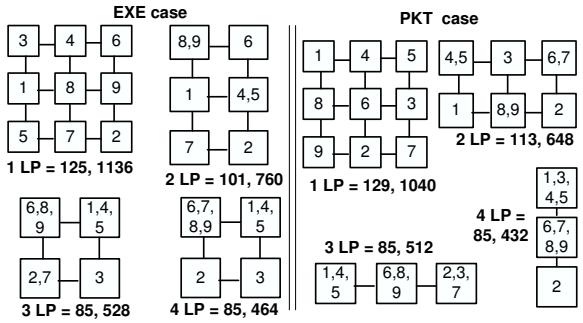


Figure 9.4: packed3 *(p3)* - NoC configuration with two optimization cases (EXE & PKT) - Cost represented as *execution time, packet count* - the mapped cores are shown inside the square (router) *(shown till # LP = 4, due to space constraints)*

mized for the total packet count need not be the configuration with the least total execution time. This is observed in almost all of the benchmarks (compare the packet count of EXE & PKT cases, in the respective benchmarks in Table 9.2). Alternately, the design that completes early can have increased packet count, thus, a tradeoff between the execution time and the packet count needs to be established. Apart from this trend, in benchmark *p4*, we observe that the execution time remains same from 4LP to 8LP router, but, the respective packet counts decrease steadily. Since, the total area will decrease with increased LP count [SV06b], the choice of a particular MLPR is now based on other constraints including the absolute power consumption, floorplan constraints, etc. This behavior stresses the need for application-specific NoC configuration generation.

In both the cases, the drastic reduction in the number of packets can be clearly observed. Against the traditional single port transfers, the μMap based transfers reduce the packet count by more than 3 folds (refer the packet count between 1LP and 9LP in PKT case(s), in Table 9.2). The effect of optimization case on the final NoC configuration is clearly seen from Figures 9.4 & 9.5.

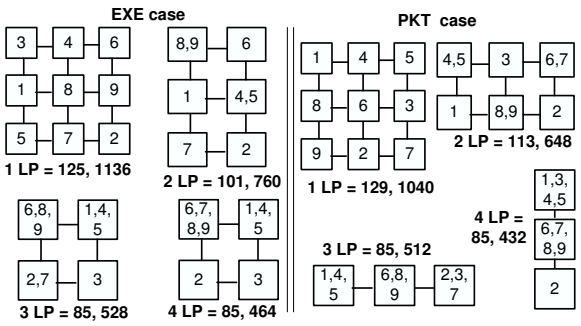


Figure 9.5: parallel2 *(pa2)* - NoC configuration with two optimization cases (EXE & PKT) - Cost represented as *execution time, packet count* - the mapped cores are shown inside the square (router) *(shown till # LP = 4, due to space constraints)*

9.4 Power Results

To get a real picture of the actual power gains obtained in absolute terms, we need to estimate the power consumed by Multi^2 router and unicast MLPR designs. For each router design, we simulate the Placed-And-Routed (PAR) model in order to generate the activity information of the PAR design. We use the XPower tool of the Xilinx ISE $6.3i$ [Xil06a] to get the power estimate values of the designs. XPower takes in the PAR design and VCD file (containing the activity data) and provides an estimate of various power parameters. We use the `ff896` package of *XC2VP30* for the purposes of power estimation. All the temperatures including the ambient and junction temperatures are set at 25 degree centigrade.

9.4.1 Power Per Flit

We define PPF (Power Per Flit) as the average dynamic power consumed in transferring a flit between two given channels. The interconnect usage information is not directly available from the Xilinx ISE synthesis tool and hence it is difficult to model them analytically in terms of the capacitances switched. Hence, we write multiple test cases and simulate wherein a fixed number of flits are transferred between various pairs of channels. For a given router design, the total average dynamic power consumption is divided by the total flit count to obtain the PPF. This is a coarse level of power estimation, but, is reasonably (relatively) accurate. Since, the aim of this research is to obtain an idea about the level of power gains achieved using Multi^2 routers, this kind of power analysis serves the purpose well.

9.4.2 Analysis of Power data

Table 9.3 presents the power estimates (of selected cases, owing to verbosity) obtained from the XPower tool of the Xilinx ISE [Xil06a]. In case of unicast MLPRs, we note that the PPF is relatively constant across the various LP routers. But, the PPF is highly non-uniform in Multi^2 routers and is observed to be increasing for larger LP routers. This increase is attributed to increased load seen at each input channel because of the direct port mapping of various crosspoint matrix lines (refer Chapter 8.3). At the same time, the PPF of the 9 LP router is significantly low and matches that of 1 LP router. To understand this behavior, we find the number of flits that are actually saved by multicasting, compared to the unicast transfers. From Table 9.3, we infer that the number of flits saved must be large in order to keep the PPF to a low value. In other words, a Multi^2 router must have a large degree of multicast transfers happening, which is highly application-specific.

We annotate the PPF & the timing information (from Figure 8.7) and recalculate the absolute values as shown in Table 9.4. For comparison, the corresponding timing & power estimates of the unicast MLPR is given in Table 9.5. The execution time (in terms of ns) is for the illustrative purposes of the synthetic benchmarks that are experimented with [SV06b]. On an average, the execution time of Multi^2 routers is found to be reduced by 20% across the benchmarks. But, the picture is entirely different in terms of the power numbers, having low power gains. We see that the PPF value of the unicast MLPRs are very low, compared to the respective Multi^2 routers. Though there is a significant reduction in packet count using Multi^2 routers, the high PPF overshadows that effect and results in low power gains. As seen from Tables 9.4 & 9.5, the 9 LP router is the best for a Multi^2 router implementation and is seen to have a very low PPF value. Thus, obtaining a low PPF value is of great importance.

Further, compared to single LP routers (Table 9.4), we observe a 16% reduction in execution time and 25% reduction in power in Multi^2 routers. For the benchmarks *r2*, *lu* and *les*, the 4, 5 & 7 LP versions fare poorly compared to the 1 LP router, leading to the negative gains. Again, the reason is the high PPF values of those router versions compared to the 1 LP case. Also, the low degree of multicast (leading to smaller gains in terms of the packet count) in those cases, aggravates the problem further. Hence, a large percentage reduction in the packet count is required to offset the high PPF values. The average power reduction raises to 35% if we ignore those odd cases involving negative gains.

To summarize, we observe significant gains in terms of both the power and per-

# LPs	Multi^2 Router		Unicast MLPR
	power/flit (mW)	# flits saved	power/flit (mW)
1	3.465	0	2.18
3	4.20	6	2.90
4	6.17	8	2.24
5	6.01	10	2.90
7	6.14	8	2.90
9	3.60	28	2.80

Table 9.3: Power Estimates from XPower [Xil06a]

# LPs	e1				p3				p4				pa2				r2				lu				les			
	Exe. Time	% Diff.	Pwr.	% Diff.	Exe. Time	% Diff.	Pwr.	% Diff.	Exe. Time	% Diff.	Pwr.	% Diff.	Exe. Time	% Diff.	Pwr.	% Diff.	Exe. Time	% Diff.	Pwr.	% Diff.	Exe. Time	% Diff.	Pwr.	% Diff.	Exe. Time	% Diff.	Pwr.	% Diff.
1	667	-	2273	-	1213	-	3604	-	1250	-	3714	-	1410	-	2439	-	1767	-	1885	-	1297	-	1940	-	1062	-	1996	-
3	596	11	1411	38	859	29	2150	40	980	22	2218	40	1192	15	1848	24	1495	15	1546	18	1151	11	1579	21	899	15	1546	23
4	567	15	1730	24	876	28	2670	26	834	33	2769	25	1133	20	2472	-1	1442	18	2175	-15	1092	16	2126	-14	958	10	2324	-16
5	512	23	1587	30	927	24	2308	36	883	29	2452	34	1112	21	2308	5	1439	19	2019	-7	1068	18	1971	-4	970	9	2116	-6
7	569	15	1523	33	883	27	2014	44	980	22	2309	38	1379	2	2112	13	1597	10	1817	4	1186	9	1916	0	1077	-1	2063	-3
9	530	20	778	66	993	18	1066	70	993	21	1238	67	1224	13	1066	56	1686	5	1008	47	1224	6	1008	57	993	7	1037	48
	Avg. Red.	17	Avg. Red.	38	Avg. Red.	25	Avg. Red.	43	Avg. Red.	25	Avg. Red.	41	Avg. Red.	14	Avg. Red.	20	Avg. Red.	13	Avg. Red.	9	Avg. Red.	12	Avg. Red.	12	Avg. Red.	8	Avg. Red.	9

Table 9.4: Multi^2 Router - Execution time (in ns) and power consumption (in mW)

formance, thus, validating the proposed approach. The ideas and the results presented in this research are just the tip of the iceberg and provides a great opportunity for realizing energy-efficient Networks-on-Chip.

9.5 Conclusion

We experiment the modified optiMap algorithm (called μMap) with the set of eighteen synthetic benchmarks presented in Chapter 5. Compared to the unicast transfers, we observe an average of 50% packet reduction (maximum of 74% using 9 Local Port (LP) router, in benchmark *p3*), across a set of benchmarks. On an average, when compared to the traditional 1 LP unicast router, there is a 16% reduction in the execution time and 35% reduction (maximum of 67% in benchmark p4) in total power consumption. In addition to significant performance benefits, the results clearly indicate a drastic reduction in the packet count of the network, thus, providing a way for an energy-efficient Networks-on-Chip. With improved multicast router designs, there is a larger possibility of further improving the energy efficiency of the overall NoC system.

# LPs	e1		p3		p4		pa2		r2		lu		les	
	Exe. Time	Pwr.	Exe. Time	Pwr.	Exe. Time	Pwr.	Exe. Time	Pwr.	Exe. Time	Pwr.	Exe. Time	Pwr.	Exe. Time	Pwr.
1	774	1430	1406	2267	1450	2337	1635	1535	2049	1186	1504	1221	1232	1256
3	811	1438	1077	2320	1319	2320	1428	1578	1791	1253	1379	1299	1174	1392
4	726	1039	1193	1684	1242	1720	1451	1147	1820	932	1304	968	1193	1075
5	773	1253	1166	2088	1323	2088	1493	1392	1729	1114	1389	1160	1218	1299
7	745	1206	1124	1949	1183	1902	1606	1346	1927	1067	1431	1114	1299	1253
9	770	1075	1163	1702	1163	1702	1359	1165	1872	986	1359	986	1223	1075

Table 9.5: Unicast MLPR - Execution time (in ns) and power consumption (in mW)

Chapter 10

Power Efficiency of Multi-Port Routers

Application-specific NoCs are preferred in place of a standard topology, so as to compete with the shared-bus in terms of performance. In order to reduce the communication overheads in traditional routers (key elements of any NoC), varying the port count of the routers are found to be beneficial. In this chapter, we critically analyze the power variations present in a router having varied number of ports, in a Networks-on-Chip. The work is divided into two major sections, projecting the merits and shortcomings of a multi-port router from the aspect of power consumption [SV07c]. First, we evaluate the power variations present during the transfers between various port pairs in a multi-port router. The power gains achieved through careful port selection during the mapping phase of the NoC design are shown. Secondly, through exhaustive experimentation, we discuss the IR-drop related issues that arise (due to large current drawn), when using large multi-port routers [Kon04].

10.1 Motivation

10.1.1 Port Level Power Savings

Being a shared network, a Networks-on-Chip suffers from several overheads including additional area, hop-based communication adding to the overall delay, congestion and tighter bandwidth constraints. Competent designs to improve the area overhead and performance are available in the literature [SBKV05, RDG+04]. An important highlight among these works is the stress on the application-specific topology generation and core mapping. The router nodes are custom-tailored in order to satisfy the performance

and bandwidth constraints, ignoring the ill-effects on the power front. NoCs consume a significant percentage of the total system power, thereby requiring power-efficient techniques [Vas04, Han03, RSG03, KYL+03, JKY05]. On the shared-bus front, we have different power optimization techniques that predominantly involve isolation of segments of bus to prevent switching-related overheads [HP00].

Objective 1: In this chapter, firstly, we attempt to improve the power efficiency by exploiting the intra-port variations in power present in a multiport router (the ad-hoc switch catered to the application at hand [SV06b, BJM+05]). Owning to space constraints, we present the results from a five port router and highlight the power savings that can be obtained by careful selection of ports during mapping of cores.

10.1.2 Impact of port count on IR drop

In addition to the router architecture, topology generation and mapping form an important phase of an NoC design, having a direct impact on the final System-on-Chip performance [OHM05]. Though it is possible to achieve a mapping that is efficient in terms of performance and power [KPN+05, SC05, YBK99], ad-hoc router design and topology generation will give rise to a larger crossbar (the key element inside router), possibly resulting in larger IR drops [ABPvG01]. Violations in terms of a larger current drawn will lead to timing issues and electromigration, eventually resulting in chip failures [CLRK99, You02]. Thus, in the nanometer regime of system design, ensuring power integrity is of utmost importance, due to the widespread appearance of IR drop and ground bounce [Str06, McC07, BPA01, BMSVH99].

Objective 2: We experiment exhaustively to observe the effect of adding ports to a router (in other words, increasing the complexity of the router) in terms of the average power variation and IR violations created. The results indicate that a large multi-port router is not beneficial from the viewpoint of power integrity.

10.2 Related Work

Different router designs aimed at improving the performance and power are available in the domain of Networks-on-Chip. A gradual shift towards the ad-hoc design of routers, tailored to the application(s) to be interconnected through the NoC backbone, is

gaining prominence. In contrast to total ad-hoc designs that route packets along any desired path [BJM+05], we have structured implementations that introduce a heterogenous composition of routers having multiple ports [SV06b]. Apart from the efficient router designs, topology generation and final mapping determine the efficiency of the NoC in terms of both power and performance [KPN+05, SC05].

With larger System-on-Chips, NoCs are found to be consuming a significant percentage of total system power, being as high as 40% [Vas04, Han03]. Several techniques that target power reduction by optimizing the packet transfer are available [MCM+05]. Hu and Marculesu [HM03b] present a branch-and-bound algorithm to get a power efficient mapping of cores onto tile-based mesh architectures, while satisfying the bandwidth constraints of the NoC. An ILP formulation for low energy mapping and routing is presented in [SC05]. An energy model and a buffer-reduction based energy-efficient NoC is given in [YMB02, YMB04]. Several techniques like wire-style and topology optimization [HZC+06, WPM05], selective long-link insertion [OM05, OMLC06, HCZ+05] and voltage-scaled links [CLKI06, SK04] aim to improve the energy efficiency of the NoC.

All of these works target efficient packet transfer between the various router nodes of an NoC. To the best of our knowledge, this is the first work to investigate the power variations at an individual router level using multiport routers [SV07c]. Multiport routers have been found to be efficient in terms of overall performance and power at the system-level. It is to be noted though that presently there is insufficient work looking on the IR drop effects of using ad-hoc router nodes. In this work, with suitable experimentation, we discuss the IR drop effects that arise by using larger multiport routers.

10.3 Experiment Platform

In contrast to ASICs, modern FPGAs come with numerous capabilities such as embedded hard/soft processor cores, DSP and transceiver blocks with an operating frequency crossing 500MHz, all of which provide an excellent opportunity for realizing platform-based and System-on-Chip (SoC) designs on reconfigurable fabrics [Xil06a]. Exploiting the advantages of a NoC style of interconnection is gaining popularity in FPGA-based SoC designs [SBKV05, SSA04]. In this work, we use the router designs that were originally developed for an FPGA-based NoC. Hence, in our experimental flow*, we carry out the

*An exhaustive discussion of the two flows is given in Appendix A

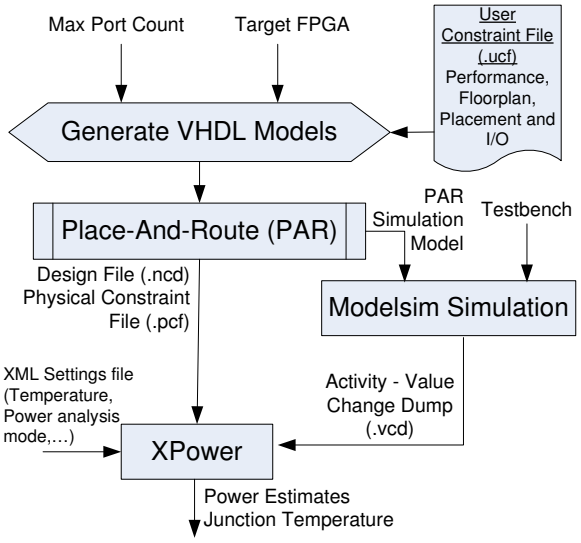


Figure 10.1: Xilinx FPGA flow

prototyping and characterization of router modules on an Xilinx-based FPGAs to meet the first objective.

10.3.1 Xilinx flow

In this flow, the power differences present between various pairs of ports in a multiport router are captured. We obtain the primary multi port router designs from [SV06a], which are based on Xilinx FPGAs and make intelligent use of the block RAMs available across the FPGA. As shown in Figure 10.1, given the port count and the target FPGA device, we synthesize the various multi port designs using the Xilinx ISE synthesis tool. After the Place-And-Route phase of synthesis, we simulate the Placed-And-Routed (PAR) router simulation model using ModelSim 6.3i [Men07] and generate the Value Change Dump (VCD) file. Throughout the simulation, the switching activity of all nets and logic in the PAR design at every clock step are stored in the VCD file. Next, XPower tool of the Xilinx ISE 6.3i [Xil06a] is used to obtain the power estimate values of the design, for the vectors that were provided as input for the current run. XPower takes in the PAR design file (.ncd), the physical constraint file (.pcf), the user settings file (.xml) and VCD file (containing the activity data) and provides an estimate of various power parameters. We use the ff896 package of XC2VP30 Xilinx Virtex II Pro FPGA for the purposes of power estimation. For experimentation purposes, the temperatures including the ambient and junction temperatures are set at 25 degree celsius during all simulation runs.

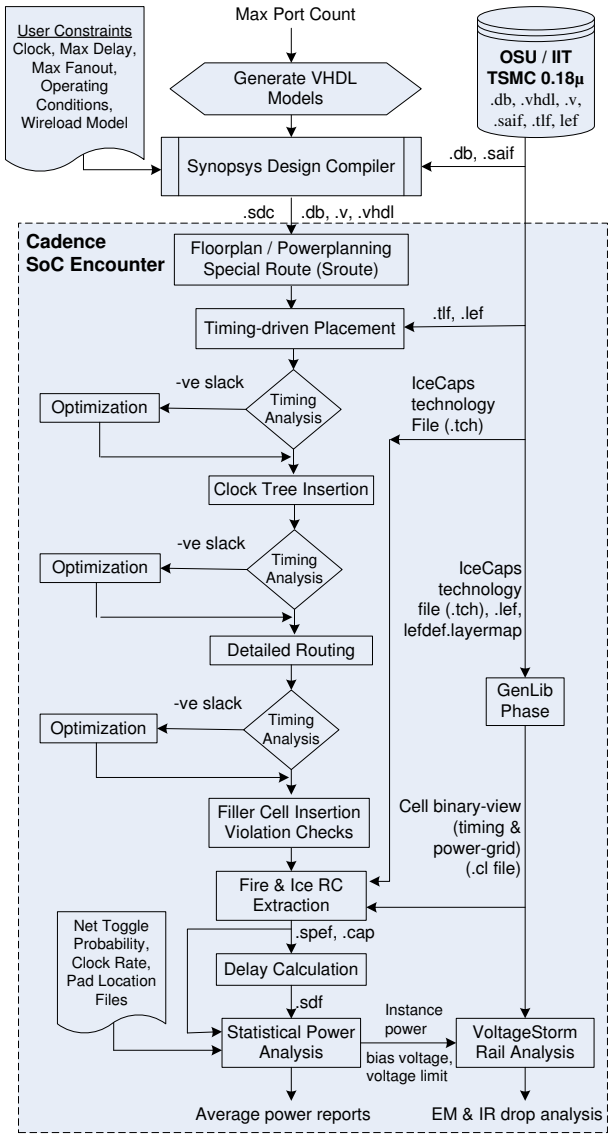


Figure 10.2: Synopsys-Cadence Flow

10.3.2 Synopsys-Cadence Flow

A vector set is used to observe the power variations between various ports in the Xilinx FPGA based flow. Though the power estimation tool (XPower) of the Xilinx ISE synthesis platform is able to report the average and peak powers, it is not comprehensive in terms of the temperature and IR drop analysis. Many of the device level details are abstracted away and the user has to remain contended with the summary reports and files generated by Xilinx ISE. Due to limited leeway available for an extensive IR drop analysis, we port the designs onto a ASIC based flow, making use of the Synopsys and Cadence

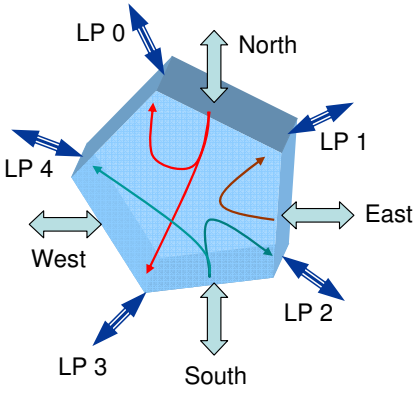


Figure 10.3: A Five Port Router with multicast capability

CAD tool set. During porting, the only required change was to replace the Xilinx BRAM based FIFO (the buffer elements that store the packets in an NoC) association into a user defined FIFO . This is because the Synchronous FIFO implementation using BRAM is only available as a black-box implementation using Xilinx LogiCORE tool [Xil05] and the corresponding reference is replaced with a new RTL implementation of the FIFO.

Figure 10.2 shows the complete flow for layout synthesis, followed by power and rail analysis. TSMC 0.18μ library available from OSU (formerly from IIT) [osu07] is used at various stages of the flow. First, the VHDL router designs are input into Synopsys Design Compiler and a gate level net list along with the timing constraint file (.sdc) are obtained. We port them into Cadence SoC Encounter in addition to the timing library (.tlf) and LEF (Library Exchange Format) files of the standard cells from IIT TSMC 0.18μ library. After initial floorplan and power rail (vdd/gnd) definition, the power track routing (special route) and via-insertion are performed. Timing-driven placement, clock tree insertion and detailed routing constitute the next phase of tasks, with the intermediate timing violations removed through an optimization phase. This is then followed by filler-cell insertion and verification, in order to check for various issues including a check for complete connectivity.

The next sequence of steps constitute the power and IR drop analysis using Cadence SoC Encounter [Cad07a]. Using the Layer Map file and IceCaps technology file (.tch file, having models for resistance and capacitance extraction in various layers) as input, the GenLib routine is invoked to create a binary-view (.cl library) of the LEF cells (TSMC 0.18μ). The binary view has two key data, namely, the graycell data (for extraction-for-timing flows) and power-grid view of all cells. Fire & Ice RC extractor [fir07] is used to generate the Standard Parasitic Exchange File (.spef) followed by the delay estimation

Source Port	# Destination Ports receiving the same data from Source port											
	1 (1-to-1 transfer)			2 (1-to-2 transfer)			3 (1-to-3 transfer)			4 (1-to-4 transfer)		
	Average Power (mW)			Average Power (mW)			Average Power (mW)			Average Power (mW)		
	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
N	173	180	177.75	186	190	187.70	193	197	195.20	201	204	202.20
E	190	194	192.13	200	204	203.30	210	216	212.70	220	224	222.00
W	162	165	163.75	172	177	174.90	182	188	184.80	193	197	194.80
S	153	163	159.63	170	174	172.10	180	184	182.00	190	193	191.80
L0	171	175	172.50	178	181	179.50	184	187	185.75	192	192	192.00
L1	176	179	177.63	184	187	185.67	192	194	193.00	200	200	200.00
L2	176	182	179.38	185	190	187.50	193	198	194.75	202	202	202.00
L3	172	175	173.50	179	174	179.83	193	197	195.20	193	193	193.00
L4	180	183	181.38	188	192	189.83	196	199	197.50	205	205	205.00

Table 10.1: Five port router - Average Power consumption between different of ports (L0-L4 represent logic port)

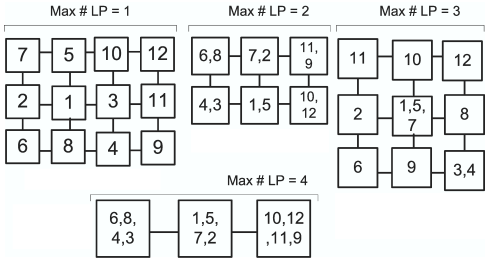


Figure 10.4: MPEG4 - Mapping of cores with different multiport routers

(SDF file generation). Since, the worst case for IR drop is hard to construct using vector based power analysis, we make use of the statistical power analysis with a net toggle probability of 0.5 and clock rate of 100MHz (typical). A report for average/peak power is generated along with detailed instance power files, which are input to the VoltageStorm tool. VoltageStorm is sign-off tool for detailed rail analysis to find IR drop violations in the layout and the profile is displayed as a power graph [Cad07b].

10.4 Intra-port Power Savings in Multi Port Routers

First, we describe some of the terminology related to a multiport router. In a traditional Networks-on-Chip design, a mesh based topology comprises of a router that has four directional ports (North/East/West/South) and one logic port to which the IP core gets attached. This is referred as single (logic) port router. Hence, not counting the directional ports, multiple number of logic ports constitutes a multiport router. Figure 10.3 shows a five (logic) port router having three simultaneous transfers happening in a multicast fashion. A detailed discussion of the router design and capabilities is present in Chapter 4, Chapter 6, Chapter 7, Chapter 8, and Chapter 9 [SV06b, SV07a, SV06a, SV07b]. Figure 10.4 shows a particular case wherein the MPEG4 application (refer [BJM+05] for bench-

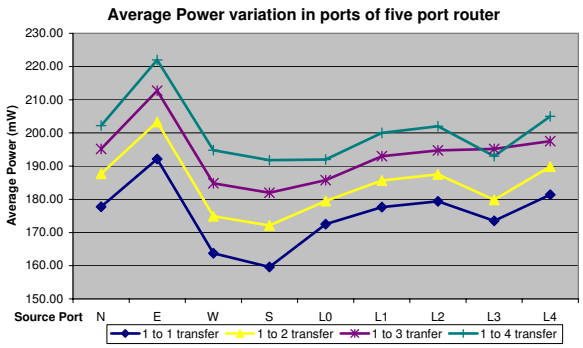


Figure 10.5: Average power variation in different source ports

mark) is mapped with various multiport routers[†]. The mapping algorithm(s) will attempt to cluster the cores optimizing the performance and the packet count, while satisfying the bandwidth constraints at the system level [SV06b,SV07a,SV07b]. The underlying assumption across the work presented so far is that when NoC architecture generation (topology + mapping) is complete, the amount of power for intra-router switching (between ports of the same router) is constant. Hence, on an individual router basis, the logic port to which core gets attached is considered to be having no effect on the overall power consumption.

In this work, we concentrate on the Multi^2 Router[‡] and analyze the power differences in switching packets between various ports (intra-port transfers) [SV07c]. We simulate the router designs by switching a *fixed set* 10000 packets between various router ports. Table 10.1 summarizes the average power estimated by the Xilinx XPower tool for different combinations of packet transfers between various ports. Each row corresponds to a source port from which the packets are switched to multiple ports in a multicast fashion [§] [SV06a]. Each source port can switch to any combination of the rest of the ports. Eg., in first case, N can transfer to {E, W, S, L0, L1, L2, L3, L4} and in 2nd case N can transfer to {L0&L1, L0&L3, L0&L4, L1&L2, L1&L3, L1&L4, L2&L3, L2&L4, L3&L4, . . .}. Similarly, combinations for rest of the cases and for rest of the source ports can be obtained. Owing to verbosity, for each of the four cases & for each of the source port, we present the *minimum*, *maximum* and *average* of all the combinations of transfers.

For a given source port, the *minimum* and *maximum* values clearly indicate the spread of the average power consumed based on the destination port(s) at hand. Contrast-

[†]Max # LP in the figure denotes the maximum number of logic ports, with the mapped cores inside the box.

[‡]Owing to verbosity, we present only results from a 5 port router

[§]Four cases indicate # of participating ports which receive the same data from the single source port (multicast) [SV06a].

ingly, the choice of source port also affects the average power, as shown in Figure 10.5. For instance, in case I (1-to-1 transfer), the difference in average power based on the source port is as high as 20% (refer 4th column of Table 10.1). Assuming the degenerate case wherein same amount of data is switched between all ports, the minimum savings of 20% is obtained. Depending on the amount of data that is switched between ports by an application in hand, the savings can grow to a large value. It is seen that irrespective of the type of transfer (1-to-1/1-to-2/1-to-3/1-to-4), W & S & L0 are better candidates as source ports for a high bandwidth transfer, with E & L4 being worse. For example, if there are two transfers, wherein one is switching 4$\times$ amount of data when compared to the other transfer, by careful port selection, the power savings gets improved by 4$\times$20% (due to 20% difference in average power between two extreme cases in mapping).

After developing a database of power values for various ports of a multiport router, a power-efficient mapping on a per router node basis is possible by careful selection of ports during the mapping phase of NoC. Thus, it is clearly seen that the choice of the source port and the destination port is of prime importance for improving the power efficiency of an NoC having multiport routers [SV07c].

10.5 Power related Issues in Multi port Routers

In this section, we analyze the power-related issues that arise owing to a complex/larger multiport router [SV07c]. We adopt the Synopsys-Cadence flow (refer Section 10.3.2) for the various simulation runs. In addition to statistical power analysis, an exhaustive rail analysis is performed to observe the negative effects in terms of IR drop increase [ABPvG01, Str06, You02].

10.5.1 Average Power Increase

Addition of ports to a router reduces the dimension of the topology of the Networks-on-Chip, since the routers with smaller port count are replaced with a larger multiport router. Albeit a reduction in the operating frequency, the overall system performance and power improves as lesser hops take place [SV06b]. But, a power perspective of larger multiport router needs to be established to find the point of diminishing returns for average power.

Figure 10.7 shows an increase in the average power that results due to addition of ports forming a larger multiport router. The flatness of the curve with changing toggle probabilities for the *same router* design is along the expected lines. With respect to the average power of a single port router, we observe an increase as high as 5× in case of nine port router (refer br9inc in the figure). For an overall power efficiency, this increase in average power must be less than the power gains obtained by hop reduction using a multiport router in place of multiple smaller routers.

A 3×3 mesh (with 9 single port routers) and different multiport routers are shown in Figure 10.6. Using data from Fig. 10.7, the increase in power with respect to the single port router is indicated inside the boxes of all the router versions (Figure 10.6). For example, the increase in the average power of a three port router is 1.3×. It takes approximately 20% more power to switch a packet between two routers compared to the packet switching within a router (between various ports) and hence the links connecting the various single port routers in the 3×3mesh are annotated with a 1.2× increase [SCK06].

Let us assume a simple case where there is a single source and other routers receive from the source (marked as D1-D8 in 3×3 mesh). The total power is the summation of the power at various links and ports of the routers. For instance, in a 3×3 mesh having 9 nodes (1 source and 8 destinations), D1/D2 consume 2.2× (1.2+1), D3/D4/D5 consume 3.4×(1.2∗2 +1) , D6/D7 consume 4.6 (1.2∗3 +1) and D8 consumes 5.4× (1.2∗4+1). Note that all of them have a constant single additive term of 1 (the average power of a single port router) representing the power required to switch to the logic module, on reaching the destination router. In total, it takes 29.6× to send one packet from the source to all of the 8 destinations. In contrast, if we have a single nine port router, the total power required is estimated as 40× (8 packets ∗ 5×), since the average power is 5× compared to a single port router.

Table 10.2 summarizes the results for other router versions using the above method. When using only the single port routers, prime & odd number of routers introduces a linear chain of routers. For example, 1×3 mesh for 3 routers, 1×5 mesh for 5 routers and 1×7 mesh for 7 routers, the diagonal length of which are abominably high. Case I in Table 10.2 represents the power increase tolerating this linear chain. This linear chain can be broken, provided the router count is even. We can render it possible by using a two port router in place of two single port routers. Thus, we can transform 1×3, 1×5 and 1×7 linear chains into 1×2, 2×2 (grey-shaded inside dotted box) and 2×3 meshes, thereby, reducing the diagonal length by half. Taking this new scenario into account, Case II of Table 10.2

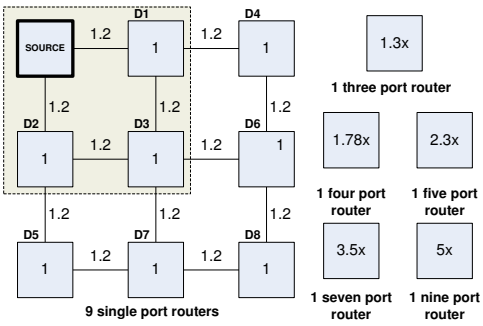


Figure 10.6: Example NoC mesh - Normalized power

# nodes	using single port routers		using one multiport router
	Case I	Case II	
3	5.6×	4.2×	2.6×
4	7.8×	-	5.34×
5	16×	11×	9.2×
7	31.2×	17×	21×
9	29.6×	-	40×

Table 10.2: Increase in Average Power normalized w.r.to a single port router

represents the effective power increase. We can observe that a single nine port router is inferior and a seven port router is bad compared to the Case I (linear chain) in terms of the power. Even though a combination of smaller multiport routers (eg., 2-5 port routers) is beneficial, it is inferred that a larger multiport router (eg., 9 port router) must be sparingly used from a power angle.

10.5.2 Rail Analysis

A router is a combination of various elements including buffers, I/O channels and a crossbar based interconnection of various ports. The switching activity is distributed in a non-uniform fashion across the router with the crossbar bearing the brunt. This is particularly true inside a crossbar where a complex interconnection of multiplexers and demultiplexers between various ports increases the interconnect density and hence the switching activity, within a small area. Concentration of large switching activity on the crossbar results in large current being drawn from the nearby rails. Such non-uniform IR drops introduce large variations in temperature gradient, with the creation of hotspots in extreme cases [BPA01,Bil74,BMSVH99]. The non-uniformities of temperature across the substrate is shown to degrade the interconnect performance [ABPvG01, ATT05]. Hence, timing issues are created due to excessive IR drop and ground bounce effects. Recent articles throw more light on the importance of doing an extensive IR drop analysis at a post-layout stage for maintaining system reliability [Str06, McC07, You02]. Thus, ensuring the power in-

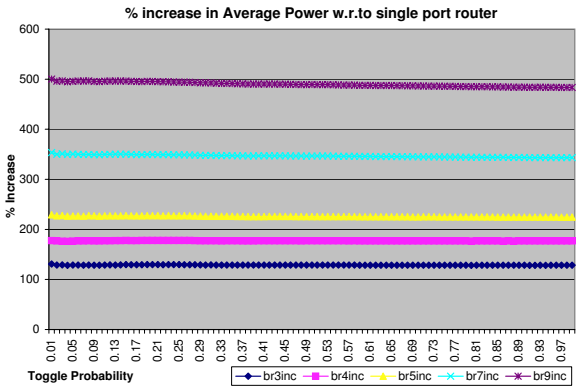


Figure 10.7: Average Power increase

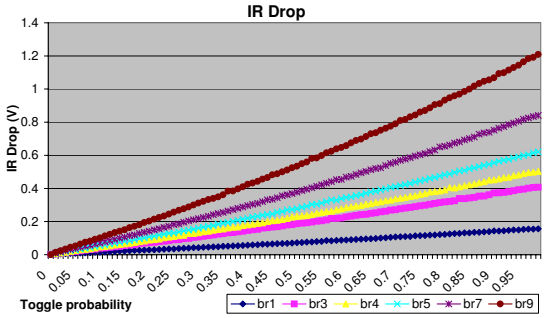
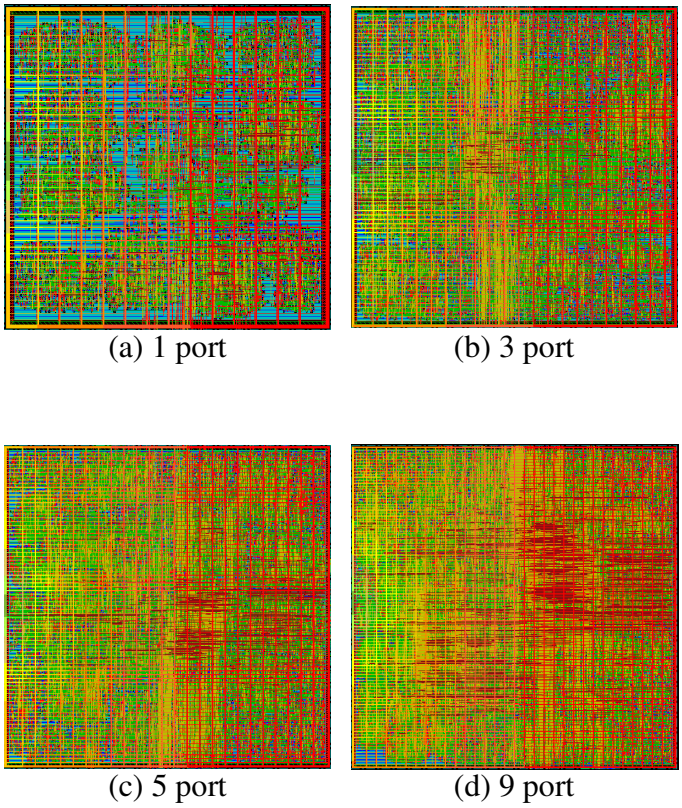


Figure 10.8: VoltageStorm Rail Analysis - IR drop in various routers

tegrity is of utmost importance in order to prevent chip failures that may result due to thermal and electromigration effects [ATT05].

The issues discussed above are exacerbated by the arbitrary addition of ports in order to realize a larger multiport router. In a large System-on-Chip, concentration of high switching logic design is detrimental due to various thermal-related issues. Hence, we do an extensive rail analysis of the various multiport router designs [SV07c], using the sign-off rail analysis tool, VoltageStorm of Cadence SoC Encounter [Cad07b, Cad07a].

Figure 10.8 shows that the rate of increase in IR drop with increasing activity inside the router is very large for the seven port router (br7) & nine port router (br9). A similar effect is evident in Figure 10.10, wherein the % increase in IR drop of various multiport routers is shown with respect to the IR drop of the single port router for various toggle probabilities. In Figure 10.11, we compare the percentage increase in IR drop of various router designs with respect to the corresponding base design. Here, for a given router design, a base design is the one having the typical minimal value of 0.1 as the toggle probability. For example, for a five port router (br5), the corresponding base design is the



(a) 1 port (b) 3 port

(c) 5 port (d) 9 port

Figure 10.9: VoltageStorm Rail Analysis (IR drop) power graphs - *Color illustration*

IR drop estimated for the same five port router, with 0.1 toggle probability. Till the port count of five, the % increases are uniform with the curves overlapping. But, the deviation is distinct for the seven and nine port cases.

Next, we use the VoltageStorm signoff tool to perform a detailed rail analysis for violations and display the results as a color-coded power graph of the router designs. As suggested in the industry flow [Str06], we fix the net toggle probability at 0.5 and a clock frequency of 100MHz (typical). In Figure 10.9, we present the power graphs of selected router versions, showing the violations occurring at various points of the router designs (as determined by VoltageStorm). As we move from left to right, we notice a marked increase in the amount of violations, the worst being indicated by dark red color[¶]. As pointed earlier, the worst violator regions happen to the ones having the crossbar connections. This distribution of IR drops is a good indicator of the temperature gradient profile that will result across the substrate [Str06, You02, Cad07b].

In summary, we can clearly infer that larger multiport routers result in the creation

[¶]Figure 10.9 is a colored illustration and the differences may not be observable in a B/W printout

of worst IR drops and will eventually lead to a host of critical issues including hotspots, electromigration and timing errors [ABPvG01, BPA01, Str06, McC07, You02].

10.6 Conclusion

Application-specific NoCs make use of multi port routers. We analyze the merits and shortcomings of multi port routers from a power perspective. We show that it is possible to achieve a power-efficient mapping, by exploiting the power differences in switching among various ports. Through exhaustive rail analysis, we infer that large multi port routers introduce greater amount of IR violations. Hence, a heterogenous mix of smaller multi port routers will provide a better tradeoff in terms of performance and power.

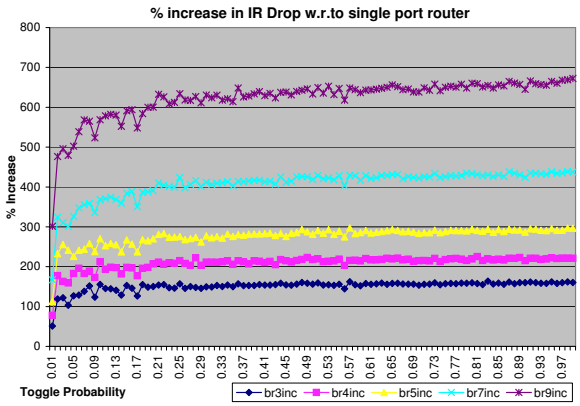


Figure 10.10: VoltageStorm Rail Analysis - % increase in IR drop w.r.t single port router

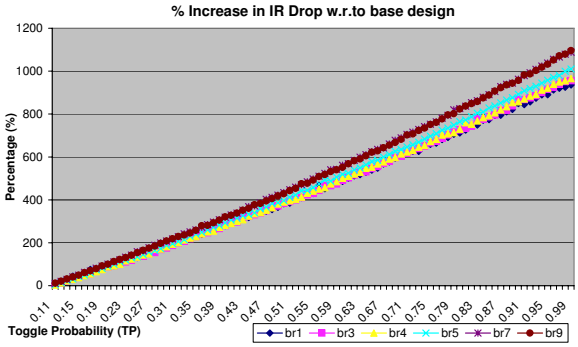


Figure 10.11: VoltageStorm Rail Analysis - % increase in IR drop w.r.t base design (having 0.1 toggle probability)

Chapter 11

Bandwidth Variations in a Dynamic Task Structure Environment

Modern System-on-Chips (SoCs) integrate a large number and variety of Intellectual Property (IP) cores involving high bandwidth transfers. Networks-on-Chip (NoC) is projected to be a scalable communication backbone in place of a shared-bus and an efficient Networks-on-Chip design will improve the performance and throughput of the overall system. Multitude of applications in a modern SoC dictate varying communication patterns between the mapped IP cores, thereby, creating a dynamic nature in the intercommunication [MKSC05]. In such circumstances, the NoC topology generation and mapping phases must consider the bandwidth variation along various links in order to prevent violations.

In this chapter, we investigate the impact of the dynamic nature of the task graphs that may lead to bandwidth violations. We present an algorithm to find the Minimum BandWidth Guarantee (MBWG) that can be satisfied without any violation, along the various links of the NoC architecture. In addition, we vary the port count of the various router nodes in the NoC architecture and experiment to find its effect on the MBWG required. The results demonstrate the promise of the proposed approach in aiding an efficient application-specific NoC design.

11.1 Motivation & Introduction

According to ITRS [Sem06], the trend in the semiconductor industry is towards a larger percentage of design reuse, in order to increase the productivity amidst stringent

Time-To-Market constraints. A platform based design methodology is followed to build a System-on-Chip, wherein several processor/memory cores along with the Intellectual Property (IP) cores are integrated [LZT04]. In such circumstances, we have high bandwidth communication traffic flowing between the various cores of a System-on-Chip. The traditional and popular shared-bus based interconnection of cores is predominantly a global interconnect scheme. In addition to the chip level synchronization issues (creating reliability problems), it creates a bottleneck in terms of the performance, as it is incapable of supporting hundreds of cores present in a future System-on-Chip [DT01].

Networks-on-Chip (NoC) is emerging as a scalable alternative, capable of integrating hundreds of cores while maintaining high throughput [BdM02, MMCM04]. NoC architecture is tailored in an application-specific fashion to improve the performance with least overhead [GDG+05, BJM+05, ASTBN04, HM04, CGMP99]. Different NoC topology generation and mapping strategies are available in the literature that optimize different metrics based on the given application [BJM+05, SV06b, HM03a, PRR+04, RGR+03, WKL+04].

Predominantly, the task structure has been assumed to be static (using a single trace) with negligible percentage variation in the traffic patterns. Such an assumption is no longer valid in modern multimedia SoCs [SDN+06, Phi03, Phi02], wherein there is a significant variation in the bandwidth of data flow between various cores at runtime [VM02, Pau04, SMCRT05, TST03, ZHM07, MKSC05]. For example, consider the snapshot of the traffic patterns found in Viper set-top box (Figure 11.1 [DJR01, MCR+06a]) at two different instances, portraying varying bandwidth flow. This is particularly true considering the fact that varied functional capabilities of the SoC make use of different combinations of the functional modules. For instance, the Viper SoC supports multiple resolution modes (high/standard definition), picture modes (Picture-In-Picture (PIP)/split-screen), Digital Video Recording (DVR) and internet capabilities. Such versatile feature capabilities are becoming a norm in many other electronic devices including mobile video/audio players and personal digital assistants [DJR01, MCR+06a].

Generation of an NoC architecture based on an overpessimistic bandwidth assumption is not the correct solution and will result in an oversized NoC having large overheads [MCR+06a]. Hence, it is of paramount importance that both the topology definition as well as the core mapping take the varying traffic patterns into consideration and tailor an architecture that will not cause any bandwidth violation along any of the links of the NoC. But, the number of such varying instances in a modern SoC run into several hundreds and hence, it is not possible to iteratively perform resource allocation and topology gen-

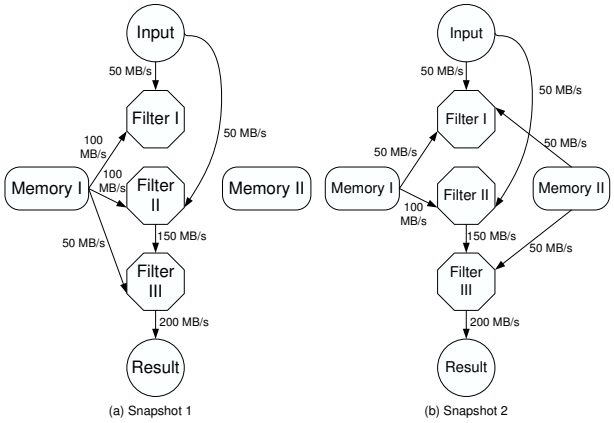


Figure 11.1: Viper communication snapshots

eration for each of the instances [MCR+06a]. Such a process is not only time-consuming (due to # configurations analyzed), but also sub-optimal, as it is hard to find a single NoC architecture meeting all the constraints, in an iterative search mechanism.

11.1.1 dynaMap Algorithm

In this chapter, we investigate the impact of the dynamic nature of the task graphs that may lead to bandwidth violations. We present an algorithm to find the Minimum BandWidth Guarantee along the various links of an NoC mesh. In addition to the traditional one-core-per-router NoC design, we analyze the effect of using multi-port routers in an NoC design flow [SV06b]. We define multiple combinations of the eighteen task graphs presented in [KA96, SV06b] along with the task graph structures of the widely experimented benchmarks including MPEG4, MWD and VOPD [JMBM04]. In this work, we restrict to the mesh-based NoCs and experiment exhaustively to find the NoC topology and mapping requiring the MBWG. The proposed methodology is generic and scalable to handle arbitrary core count as well as varied number/degree of the communication patterns.

11.2 Related Work

Topology generation and mapping are the two important steps in the Networks-on-Chip design. Several strategies are available that targeted different optimization metrics, while meeting the constraints of the application in hand [BJM+05, HM03a, LK03a]. Hu and Marculesu [HM03a] present a branch-and-bound algorithm to get a power efficient

mapping of cores onto tile-based mesh architectures, while satisfying the bandwidth constraints of the NoC. Ascia et al [ACP04] use evolutionary computing techniques to implement multi-objective exploration mapping in mesh based NoC, to obtain pareto mappings optimizing performance and power. Lei and Kumar [LK03a] present a two-step genetic algorithm to map the task graph minimizing the execution time. An NoC synthesis flow is presented by Bertozzi et al [BJM+05]. They present algorithms for mesh NoC architectures under different routing functions and delay/bandwidth constraints.

The underlying assumption of the above strategies is that the task structure is predominantly static (using a single trace) with negligible percentage variation in the traffic patterns. Modern SoCs support a wide variety of functional capabilities and the traffic patterns and conditions change at runtime [DJR01, SDN+06, Phi03]. Hence, an NoC architecture based on static assumptions will only result in violations along several links of the NoC. In the literature, we have two works that try to address this scenario. In [MCR+06b], a synthetic worst case is defined based on the constraints of all the communication patterns (instances) and an NoC architecture is generated. As admitted by the authors themselves, the strategy is based on over-specified constraints and hence, will not address the true dynamic nature of the application and will also lead to an oversized NoC. The authors extend their work to realize a scalable solution in [MCR+06a], wherein they couple the process of mapping and slot allocation to address the multiple instance scenario.

In such a dynamic task structure environment, this paper further complements and extends the work by presenting a strategy to find the Minimum BandWidth Guarantee (MBWG) required along the various links of the NoC mesh, accommodating all the communication patterns. Also, we vary the port count to observe the effect of the port count on the MBWG required. The dynaMap algorithm is scalable to support applications of any size (described by # nodes in a task graph), with any number of communication instances. Such a solution will aid the designer in observing the tradeoffs possible for the varying bandwidth constraints of the given design.

11.3 Dynamic Task Structure

In an NoC, systems are built by integrating different design cores in a defined fashion (topology + mapping). *Design Cores* are the Intellectual Property (IP) cores that are pre-built and verified for functionality, performance and other constraints. Modern System-

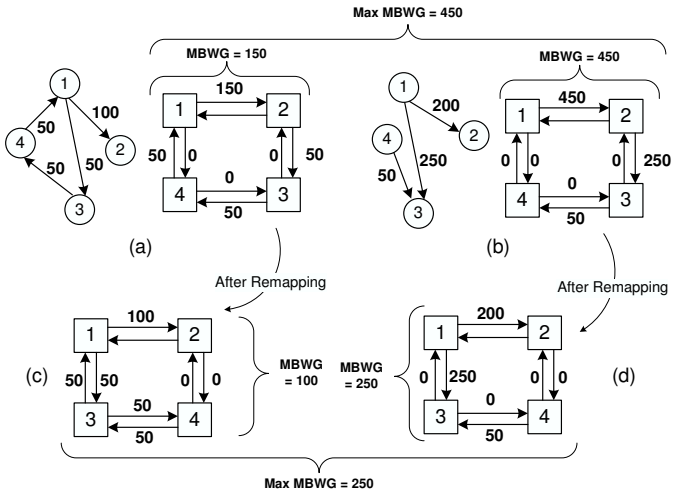


Figure 11.2: Dynamic Task Structure Illustration (units in MB/s)

on-Chips (SoCs) are a package of multiple functional modules that communicate with each other in a more dynamic fashion [DJR01, SDN+06, Phi03]. The dynamic nature of the task graphs (that define the intercommunication patterns between the Intellectual Property (IP) cores) will not greatly affect the NoC architecture, if the communication patterns remain fixed between nodes of a task graph, with only the change happening in the amount of data traffic (bandwidth) [MNTJ04, VM02, Pau04]. Such an environment will represent the simplest of the dynamic task graphs, and it is possible to only consider the worst case traffic between any pair of cores [MCR+06b]. But, in reality, there are new communication patterns (paths) created in the task graph structure (in addition to the increase in the amount of traffic), resulting in the formation of new bandwidth constraints in the system. This scenario will complicate the communication backbone design, as it becomes necessary to foresee the worst cast conditions in terms of bandwidth violations created in the NoC mesh. In short, for avoiding the bandwidth violations, an NoC architecture generation algorithm must consider the variation in the bandwidth requirement along various links and produce a tolerant & efficient design.

In order to better appreciate the dynamic task structure environment, let us consider an example representing two communication patterns. Figure 11.2 (a) represents the first snapshot of the task graph, along with a sample NoC topology and mapping. The bandwidth of traffic flowing in the link from router R1 (having core 1) to router R2 (having core 2) is 150MB/s. Hence, we note that the minimum bandwidth that needs to be guaranteed, covering all the links, is 150MB/s. If the link bandwidth constraint for the NoC design is any value less than the 150MB/s, it will result in violations. Now, consider the second

snapshot of the task graph represented in Figure 11.2 (b), having the same NoC architecture (topology & mapping) as in Figure 11.2 (a). The bandwidth of data traffic across the link from Router R1 to Router R2 has increased to 450MB/s. Hence, in order to avoid bandwidth violations, this NoC configuration requires an MBWG of 450MB/s to accommodate the two communication patterns, a significant increase.

Figures 11.2 (c) & 11.2 (d) represent a changed NoC architecture (mapping of cores 3 and 4 have been interchanged) for the task graph structure snapshots in Figure 11.2 (a) & 11.2 (b), respectively. In both the cases in Figures 11.2 (c) & 11.2 (d), we see that the $\texttt{Max}\{\text{MBWG}\}$ across the various links (accommodating the two communication patterns) has come down to 250MB/s. This example represents the simplest of the scenarios, wherein a small change in the mapping has resulted in the reduction of the MBWG requirement.

The topology and mapping of the NoC architecture can be optimized by using a router with varied port count [SV06b]. Multi Local Port Router (MLPR) is a non-traditional packed switched router architecture design, which can handle more than one logic core simultaneously, without any performance penalty [SV06b]. In an NoC mesh, MLPRs primarily aim to replace the inter-router-channel communication (between router pairs) with the intra-router-channel communication (within a single router). MLPRs are reported to provide a host of advantages including reduction in area, power, transit time & congestion, and most importantly, the optimization of the bandwidth, thus, resulting in an efficient and high performance NoC design [SV06b]. It is shown that the diameter of the mesh can be reduced by using MLPRs, transforming an NoC mesh into a heterogenous structure having a network-within-a-network (as against the traditional homogeneous NoC mesh having single port routers). Thus, it is evident that, in addition to a prudent mapping, proper topology definition must have a significant impact. Hence, we vary the maximum port count of the routers used in the NoC mesh and observe the effect on the MBWG requirment.

11.4 dynaMap: Fast Mapping Heuristic Algorithm

Summarizing the discussions in the previous sections, we infer that an efficient NoC architecture should be tolerant to the task graph structure variations and be able to avoid any bandwidth violation. In other words, the Minimum BandWidth Guarantee (MBWG) required by the NoC architecture must be estimated, along with a proper design effort to reduce the MBWG, for realizing an efficient NoC architecture. We translate the above

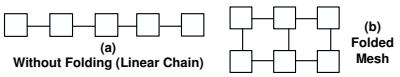


Figure 11.3: dynaMap: Folding Example

ideas into an algorithm called dynaMap*, with an objective to find the MBWG for a given combination(s) of task graph.

Problem Definition: Given a combination of communication patterns (Υ) of a system level task graph, $G(T, E)$ and a constraint on maximum port count, find an NoC configuration (# of routers, mesh topology, configuration of each router and the final mapping) with Minimum BandWidth Guarantee.

The dynaMap algorithm is described in Algorithm 10. First, we define the minimum dimension mesh for the cores in the task graph. If the number of nodes in the task graph is *n* and maximum port count is *m*, then we need at least $k = \frac{n}{m}$ routers (considering only the logic ports [SV06b]), using which we build all possible mesh configurations (ρ_1). Here, if the number of routers (k) is prime and odd, it forces a linear chain of routers (Figure 11.3(a)). Linear chain is not efficient as the diameter of the mesh is increased significantly. Further, a single row creates a large bottleneck in terms of the bandwidth usage of the links and any variation in the amount of data traffic or the communication patterns will result in violations. To resolve this situation, we accommodate one more router (k+1 routers) so that linear chain can be broken (E.g., folded mesh in Figure 11.3(b)) and build all possible mesh configurations (ρ_2).

Next, we construct a composite graph ($G_{comp}(T, E)$) by combining all the instances of the task graph, making sure that the total number of edges is sum of all the edges of different instances. For better understanding, this step is illustrated in Figure 11.4 . It is obvious that there will be multiple edges between a pair of nodes (from various instances). The idea of allowing this redundancy (in terms of the # edges between a pair of nodes) is to find the frequently communicating pairs of nodes (across all the instances). In order to achieve this, we build a list Ψ_1 comprising of the nodes in the task graph, sorted in the descending order of the *In-degree* + *Out-degree*. Thus, the frequently communicating nodes (Ψ_1) are given more priority during mapping in the mesh. For each node in the list Ψ_1, we build a corresponding list Ψ_2 comprising of nodes connected to it, sorted in the descending order of the amount of data traffic. Thus, large bandwidth transfers are given

*based on the cMap algorithm introduced earlier

Input: Given a system level task graph, $G(T, E)$ with *n* cores

Input: Υ: Communication patterns in $G(T, E)$

Output: An efficient NoC configuration representing the topology and mapping such that Max{MBWG} across various links is minimized, subject to the constraint that maximum number of port count is *m*

Routers, $k = \frac{n}{m}$

ρ_1: Generate all possible mesh topologies for the *k* routers

ρ_2: If *k* is prime & odd, increment the router count by one and generate all possible mesh topologies for *k+1* routers

$G_{comp}(T, E)$: Construct a composite graph representing all the communication patterns between all the nodes in the combinations (Υ)

Ψ_1: Build a list of nodes in the descending order of *In-degree + Out-degree* of nodes in $G_{comp}(T, E)$

foreach $\Psi_1(i)$*: Node in the List* Ψ_1 **do**

Ψ_2: Build a list of nodes connected to $\Psi_1(i)$, sorted in the descending order of the *Bandwidth (BW)*

end

$BW_{Global.Min} = \infty$

repeat

foreach τ*: Location in the current mesh configuration* **do**

repeat

foreach $\Psi_1(i)$*: Unplaced node in List* Ψ_1 **do**

Place the node $\Psi_1(i)$ in the nearest free location from τ

foreach $\Psi_2(i)$*: Unplaced node that is connected to* $\Psi_1(i)$ *in List* Ψ_2 **do**

Place the node $\Psi_2(i)$ in the nearest free location from τ

end

end

until *All cores are Placed*

Best.Cost $\leftarrow \sum_{i=1}^{\#\ edges\ in\ graph}$ Link BandWidth of mesh

for *i=1* **to** *iter-max* **do**

Current.Config: Initiate a random pair-wise swap of mapped cores

if Current.Config.Cost $<$ Best.Cost **then**

Best.Config$\leftarrow$ *Current.Config*

Best.Cost $\leftarrow$ *Current.Config.Cost*

end

end

foreach *Communication pattern in* Υ **do**

forall *Edges in a combination (task graph)* **do**

Identify the source (i_1, j_1) and destination (i_2, j_2) nodes in the mesh

Update the bandwidth of all the links between (i_1, j_1) and (i_2, j_2) in the mesh (XY routing)

end

if $BW_{Min} <$ Max{*Bandwidth of links in mesh*} **then**

$BW_{Min} \leftarrow$ Max{Bandwidth of links in the mesh}

Best.Config $\leftarrow$ *Current.Config*

end

end

if $BW_{Min} < BW_{Global.Min}$ **then**

Global.Min.BW.Config$\leftarrow$ *Best.Config*

end

end

until *All the mesh configurations in* ρ_1 *and* ρ_2*are evaluated*

Algorithm 10: dynaMap Algorithm

more priority during mapping (placed close to each other), so that the total bandwidth usage of links in the mesh can be minimized. It is quite obvious that this step will directly affect

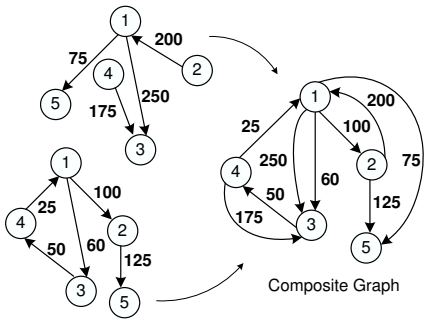


Figure 11.4: Composite graph formation (units in MB/s)

the MBWG of the mesh.

For each of the mesh configuration in ρ_1 (& ρ_2) and starting from each location of the mesh at hand, we perform the following steps in our pursuit of an NoC configuration, requiring an MBWG. The intuition behind this strategy to start from each location of the mesh is to amortize the effect of the choice of start location (which limits the # of nearest neighbors available). Using the ordered list Ψ_1 and a corresponding list Ψ_2, we perform a nearest-neighbor placement (Steps 12-19). The cost function is the sum of bandwidths in all the links - lesser the cost, the more tightly are the cores packed. It is not always necessary that the design having the cores placed with smallest sum of distances (between the cores) is optimal in terms of the global bandwidth usage. Hence, after the placement of all cores, we contemplate pair-wise swapping to see if the overall cost (bandwidth usage across all links) can be minimized (steps 21-27).

Next, we find the Minimum BandWidth Guarantee required for each of the communication pattern stored in the list Υ (steps 28-39). In order to achieve this, for each edge in the communication pattern at hand, we identify the links that will be used in the mesh (dictated by the co-ordinates of the source and destination nodes) and update them with the corresponding bandwidth. This step is similar to the illustration shown in Figure 11.2, wherein the bandwidth requirement is estimated for each of the links in the mesh for a given communication pattern. Overall, the effective MBWG for the current mesh configuration and placement is the `Max`{Bandwidth of all the links in the mesh}.

Across all configurations and mappings, we iterate to find the NoC configuration that will guarantee a minimal requirement of the MBWG. The complexity of the algorithm is linear in n. This is because we iterate only for constant factor of n, which is dictated by the number of meshes possible for the given router count (k), which in turn is dictated by the maximum port count constraint. During our experimentation with the benchmark

designs, the algorithm completed within a couple of seconds.

11.5 Experiment Results

A typical SoC design is represented as a task communication graph, which is described as a directed graph $G(T, E)$ (referred to as *Task Graph* interchangeably), where T represents the vertices (tasks) and E is the set of directed edges describing the precedence, the dependence, the timing and bandwidth constraints in the task graph. Nodes in a task graph represent independent units of computation, denoting the Intellectual Property (IP) cores of the system. The dynamacity of the task intercommunication is created by the formation of new directed edges between the various nodes having finite bandwidth requirements.

We experiment with the set of twenty two benchmarks described in 5. The set of eighteen synthetic benchmarks cover a wide spectrum of the communication patterns, with a node count of nine, having equal bandwidth of 50MB/s across all edges and equal execution time of the nodes. In this chapter, we make use of all the eighteen benchmarks and after careful analysis and experimentation, we define ten combinations or *Runs*[†] (Runs 1-10 in Table 11.1). The idea behind these Run definitions is to incorporate a widely varying dynamic task graph structure environment. In addition to the eighteen benchmarks, we include the three widely experimented benchmarks (*MPEG4, VOPD, MWD*) and define four additional runs (Runs 11-14 in Table 11.3). The node count in these three benchmarks is 12, with the bandwidth flow across the various edges varying significantly.

For the purposes of this research work, the focus is on the different communication patterns of the fourteen runs (Table 11.1) and their relation with the MBWG of the final NoC architecture. Further, in this work, we assume that the dynamic nature created in the task graphs is due to the varying communication patterns and data traffic between the nodes present. The case of having a new task graph reconfigured at runtime is not in the scope of this work. Given such a scenario, the objective of the work is to observe the variation in MBWG required for varying maximum port count constraint, across different runs.

[†]Each Run represents an unique combination of the chosen benchmarks representing varying degree of communication patterns (i.e., dynamic nature of task graph).

Run #	Combinations of benchmark set
1	bs1, bs2, bs3, bs4, r1
2	p1, p2, p3, p4
3	bs3, bs4, b1, b2, e1, e2
4	b1, b2, e1, e2, r1, r2, lu
5	bs1, bs2, bs3, bs4, lu, les, b1, b2, r2
6	e1, e2, p1, p3, lu, les, bs1, bs2, bs3, bs4
7	b1, b2, e1, e2, p1, p2, p3, p4, r1, r2, pa1, pa2
8	lu, les, r1, r2, bs1, bs2, bs3, bs4, p1, p2, p3, p4, pa1, pa2
9	bs1, bs2, bs3, bs4, b1, b2, r1, r2, e1, e2, p2, p3, les, lu, pa1, pa2
10	bs1, bs2, bs3, bs4, b1, b2, r1, r2, e1, e2, p1, p2, p3, p4, les, lu, pa1, pa2
11	mpeg4, vopd
12	mpeg4, mwd
13	vopd, mwd
14	mpeg4, vopd, mwd

Table 11.1: Runs : Combinations from 22 benchmarks

Maximum	Run 1		Run 2		Run 3		Run 4		Run 5		Run 6		Run 7		Run 8		Run 9		Run 10	
Port count	Mesh I	Mesh II	Mesh I	Mesh II	Mesh I	Mesh II	Mesh I	Mesh II	Mesh I	Mesh II	Mesh I	Mesh II	Mesh I	Mesh II	Mesh I	Mesh II	Mesh I	Mesh II	Mesh I	Mesh II
1	150	-	200	-	150	-	250	-	150	-	250	-	250	-	300	-	250	-	300	-
2	250	200	450	300	300	200	300	200	200	200	450	350	550	300	500	350	450	350	500	350
3	300	200	450	350	400	350	400	350	300	350	600	500	600	500	600	500	600	500	600	500
4	300	300	450	500	300	300	300	300	300	300	700	500	600	500	700	500	700	350	700	500
5	300	-	250	-	350	-	350	-	300	-	750	-	750	-	500	-	750	-	750	-
6	300	-	300	-	400	-	400	-	250	-	600	-	600	-	600	-	600	-	600	-
7	300	-	200	-	250	-	300	-	150	-	400	-	400	-	400	-	400	-	400	-
8	100	-	150	-	150	-	150	-	100	-	200	-	200	-	200	-	200	-	200	-

Table 11.2: Runs 1-10 : Minimum BandWidth Guarantee (MBWG) required in MB/s

11.5.1 Analysis of the Results

The algorithm was coded in C++/STL and was executed on a SunBlade 1000 workstation having dual processors operating at $750MHz$ and $2GB$ RAM. The average execution time to find the NoC configuration requiring MBGW for a given maximum port count was a couple of seconds. The algorithm uses the heuristic technique as described in Algorithm 10 to arrive the final NoC configuration. The results from various runs are given in Tables 11.2 & 11.3 for the sake of clarity. Here, Mesh I represents the case ρ_1 having k routers and Mesh II represents the case ρ_2 having k+1 routers (refer to Algorithm 10). Mesh II case is possible only when $\frac{n}{m}$ is a prime and odd number (where *n* is node count and *m* is maximum port count) and therefore, we see many *dashes (-)* in the Mesh II column of Tables 11.2 & 11.3. For instance, in Runs 1-10 (Table 11.2), Mesh II is possible only for the maximum port count of 2, 3 & 4 (as the node count is 9). Similarly, Mesh II is possible for Runs 11-14 for the maximum port count of 4 & 5 (as the node count is 12). Also, in Tables 11.2 & 11.3, the condition representing the maximum port count equal to the node count is not present because it represents the case of a single router, wherein all the transfers happen as direct intra-port transfers, and per se, there is no link present in the mesh. The results shown are obtained after back-annotating the operating frequency numbers obtained from [SV06b].

Across all the Runs[‡] in Table 11.2, the single port router outperformed most of the other cases (having maximum port count between 2 & 7) in terms of the MBWG required. Further, the MBWG for the cases having the maximum port count between 3 & 6 have less variation of MBWG. This is due to the fact that the number of routers required is either 3 (when maximum port count is 3 or 4) or 2 (when maximum port count is 5, 6 or 7) and the mappings across these various cases are similar. Hence, the links of the mesh having either 2 or 3 routers do not show a significant MBWG variation, because of the fact that less transfer happens across the shared links in the mesh. Further, Mesh II case in Runs 1-10 show better or comparable number in terms of the MBWG required, as the bottleneck due to the linear chain is avoided.

An important observation from Table 11.2 is that the cases with maximum port count of 3, 4 & 5 are worse across all the Runs 1-10, compared to the other cases. This is due to the fact that with a router count of 3, we have only two links (forward/reverse direction) available which are occupied or used by most of the transfers in the task graph. It is not so for the other case with lesser maximum port count (1 or 2). This is because the number of routers increase and so are the number of links available in the mesh. These links are used in a distributed manner with lesser bottlenecks. Similarly, in the cases with larger maximum port count (>5), despite the router count being the same, they have better MBWG. This is because more cores are mapped onto a single router and hence lesser number of transfers require the use of the link(s) of the mesh.

Runs 11-14 in Table 11.3 show a different trend, in contrast to the results in Table 11.2. We observe that there is a significant decrease in the MBWG required with every increase in the maximum port count constraint. Also, unlike the trend in Runs 1-10, the single port case is faring very poorly. This effect is attributed to the fact the Runs 11-14 involve varying bandwidth flow across different edges of the task graph (unlike Runs 1-10 having equal bandwidth across all the edges), creating larger bottlenecks in the shared links. Further, similar MBWG numbers in Run 12 & Run 14 for the maximum port count of 2, 4, 5 & 6 show the dominance of a particular task graph(s) structure (*mpeg4 & mwd*) on the final NoC design.

In order to better appreciate the above results, we present the selected NoC configurations of Runs 11-14 in Figure 11.5. As evident, the varying communication patterns in different runs (dictated by the frequency of communication through the number of edges and the bandwidth of data flow between the cores) create differences in the final mapping.

[‡]The results are multiples of 50 because the benchmarks used in Runs 1-10 have equal bandwidth of 50MB/s [SV06b]

Maximum Port Count	Run 11		Run 12		Run 13		Run 14	
	Mesh I	Mesh II	Mesh I	Mesh II	Mesh I	Mesh II	Mesh I	Mesh II
1	1843	-	1792	-	419	-	1702	-
2	1102	-	1770	-	392	-	1770	-
3	2999	-	1740	-	383	-	1813	-
4	963	1083	2451	2451	705	407	2451	2451
5	764	1250	843	1330	732	407	843	1330
6	925	-	783	-	702	-	783	-
7	924	-	845	-	389	-	924	-
8	923	-	844	-	389	-	923	-
9	830	-	751	-	357	-	830	-
10	407	-	251	-	353	-	407	-
11	80	-	128	-	64	-	80	-

Table 11.3: Runs 11-14 : Minimum BandWidth Guarantee (MBWG) required in MB/s

This observation is true even in cases where there is a fixed router count (due to the fact $\frac{n}{m}$ can be same for multiple maximum port count constraint, as it is an integral division representing the integer number of routers).

In summary, we infer that an efficient NoC configuration in terms of the MBGW required is possible, if the bottlenecks in the shared links of the mesh can be reduced, either by distributing the flow across many links (single port router) or by reducing the usage of the links (larger port router). In any case, there is no consensus winner in terms of the maximum port count constraint and an application-specific NoC architecture generation becomes necessary. Towards this objective, the proposed heuristic technique is found to be successful in finding the NoC configuration with Minimum BandWidth Guarantee (MBWG) required.

11.6 Conclusion

Modern System-on-Chips are a package of multiple functional modules that communicate with each other in a more dynamic fashion [DJR01, SDN+06, Phi03]. Here, new communication patterns (paths) created in the task graph structure (in addition to the traffic increase), resulting in the formation of new bandwidth constraints in the system. Hence, an efficient NoC configuration (topology & mapping) generation needs to take the variation in the bandwidth requirements along various links into account or alternately, must be tolerant to the task graph structure variations and be able to avoid bandwidth violations. Towards this end, we present a heuristic technique called dynaMap to find the NoC architecture requiring minimal bandwidth guarantee. We analyze the impact of maximum port count on the MBWG required. The results demonstrate the need for a highly application-specific NoC architecture generation, catered to the task structure(s) at hand. The knowledge of the

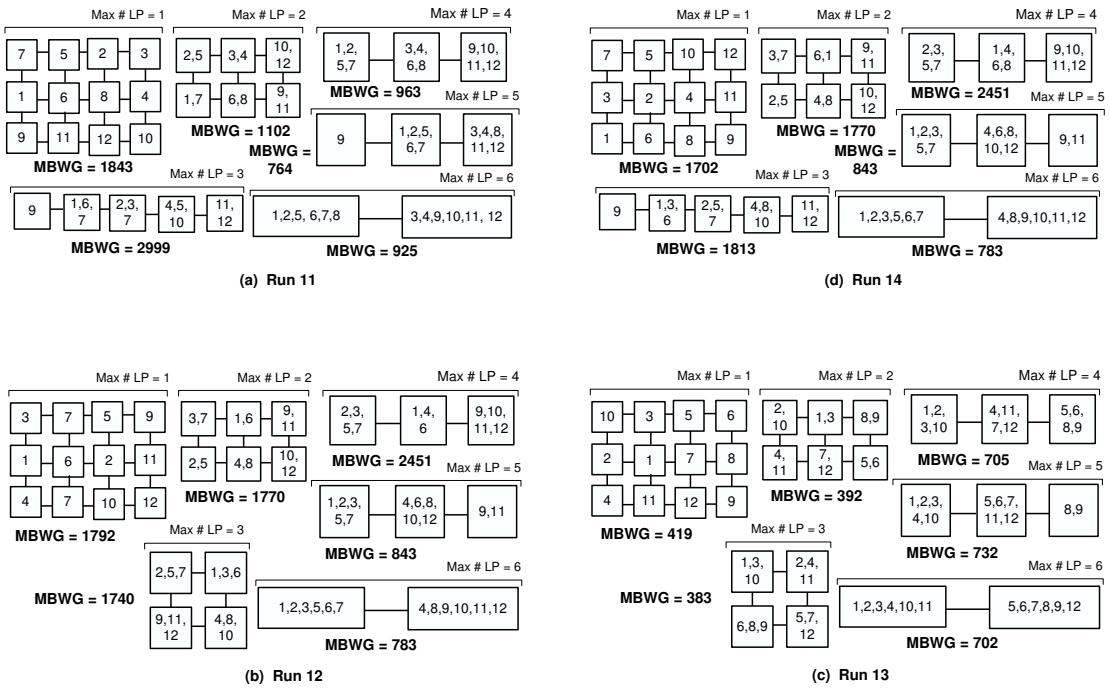


Figure 11.5: NoC configuration in Runs 11-14 (with Maximum Port Count between 1 and 6) - Mapped cores are inside square/router *(not to scale - for illustration of topology & mapping only)*

maximal bandwidth variation across links is highly inevitable during the NoC design, as it can help a great deal in preventing many unwanted surprises in terms of the bandwidth violations created.

Chapter 12

Towards Multi-FPGA Systems with Networks-on-Chip

Traditionally the reconfigurable devices including the CPLDs and the FPGAs have been limited predominantly in terms of the effective logic capacity available. The area limitation is the primary motivation for this dissertation because of which we attempt to optimize the individual router design(s) as well perform selective grouping of routers (thus, clustering of cores) so as to reduce the overall area occupied by the on-chip network. In this chapter, we explore the applicability of the designs and methodologies proposed in earlier chapters in relation to System-on-Chip design spanning across multiple FPGAs. We concentrate to find the changes that are necessary to fit the proposed methodologies seamlessly onto a Multi-FPGA based System-on-Chip design. The discussions given below captures the framework comprising of the architectural and algorithmic design modifications, and the actual implementation of the proposed ideas are suggested as a possible future direction of work. The primary objective of this chapter is to present a motivation for an multi-chip (multi-FPGA) Networks-on-Chip, and present the pertinent issues that need to be addressed in a Multi-FPGA scenario, in line with ideas proposed in this research.

12.1 Introduction

Field Programmable Gate Arrays have been the first choice in the context of logic emulation [Hau98]. Though there has been a significant jump in the number of configurable logic elements (eg., CLBs in Xilinx), FPGAs have been always limited in terms of the total

usable logic area that is available compared to the contemporary ASIC counterparts. Hence, a full-fledged logic emulation of a large design requires the use of multiple FPGAs in order to accommodate the entire system design. This gives rise to the new scenario involving a structured interconnection of multiple FPGAs called *Multi-FPGAs*. The chosen set of FPGAs are placed on an Printed Circuit Board and are inter connected through the communication links following a defined topology. Given the dynamic reconfiguration capability, the significance of a Multi FPGA systems are elevated to build scalable System-on-Chip designs controlled by the host computer or a dedicated on-chip configuration controller. In the literature, we have several multi FPGA based system designs that have demonstrated acceleration several orders higher than a comparable software based simulation and prototyping systems [GHK+90, HTA94, SSS95, SADB95, VBR+, CM94, Dzu93, CR93, MC93, HKL+95, MVB95, ST95]. Detailed account on the research and products based on the multi FPGA systems is present in [Hau95, HBE98, HB97b, HB97a, Hau98, HB95].

In Multi-FPGA based systems, the given application (described at the algorithmic/behavioral or the Register Transfer Level) is synthesized and mapped across the FPGA boundaries optimizing the overall performance and power. In this context, a mesh type of interconnection topology is found to be an ideal choice in optimizing the external interconnect resource usage. It must be understood that multi FPGA systems are realized in a two dimensional planar printed circuit boards, and the luxury of having several layers of interconnects running in three dimensions is very restricted in multi FPGA boards. Hence, the experimentation of the interconnect schemes are restricted to mesh in most of the works available in literature, though there are schemes including hierarchical crossbars and custom-tailored mesh-based schemes like one-hop, four-way or eight-way topologies [Hau98, Hau95]. The limitations present in earlier generation of FPGA in terms of the total available pin count is overcome by the adoption of time-based multiplexing schemes. One such scheme is the popular *Virtual Wires* project [BTA93], wherein a pipeline is defined internally to an FPGA and the Input/Output pins are time-shared to make transfers to and from the FPGAs. In other words, virtual wires represent a uncommitted logical interconnection between the output from one FPGA to the input of another FPGA. It is contended that this approach will result in the increase of the available off-chip bandwidth, in addition to overcoming the pin-count limitations that is prevalent in the previous generation FPGAs [BTA93]. The latest FPGAs (eg., Xilinx Virtex 5) offer increased pin count (reaching close to four hundred pins) [Xil07b], in addition to the improved logic area. Hence, in contrast to earlier generations, the multi FPGA designs are no longer restricted by the pin count, but rather by the effective logic area available inside each FPGA. This is highly

significant as optimal packing of FPGA can lead to the minimization of the out-of-chip interconnections.

12.2 Extension of Networks-on-Chip for Multi FPGAs

In this section, we attempt to throw light on the issues that need to be addressed when several FPGAs or for that matter several chips (ASICs) are to be interconnected, which is a commonality in a platform-based System-on-Chip designs & embedded systems. The design environment comprising of IP cores spread across several FPGAs, with Networks-on-Chip being the preferred medium for the communication backbone design, represents altogether a different scenario. On the positive side, the complex issues related to the routing resource optimization and the custom-tailoring of various of mesh-topologies [Hau98, Hau95] are totally abstracted away, thus, helping the System-on-Chip integration possible in a fast and efficient manner. At the same time, there exists some basic requirements in terms of the capability offered by the different routers and the flow control & routing schemes that are followed. We discuss the most pertinent issues in line with the other works presented in this dissertation below.

12.2.1 Modified Design Framework

Architecture

For a seamless integration in a multi FPGA environment, the changes made to a router must be kept minimal. This will allow the designers to operate at the highest level of abstraction. In this section, we describe the changes are necessary with regard to the router architecture, the flow control and the routing algorithm.

Predominantly, the strategy adopted in the topology definition of Multi FPGA systems has been to have a minimal usage of the out-of-chip interconnections running across the printed circuit board. This is intuitive considering the fact the impedance seen by the out-of-chip interconnections is tens of hundred times greater when compared to the on-chip interconnections. Hence, the inter-FPGA (chip) communication will result in larger currents being sinked during data transfers, confirming to the voltage standard of our choice (eg., LVTTL, LVCMOS33, LVCMOS18, LVDC33, LVDC15, PCI33, PCI66, PCI-X, GTLP, GTL, HSTL, DIFF HSTL, SSTL, DIFF SSTL [Xil07a]). In the moti-

vating scenario of versatile Input/Output capabilities present the latest FPGAs [Xil07b], tThe exact architecture at the FPGA I/O boundary is a matter of designer's choice depending on the tradeoffs required. For instance, the SelectIO technology (with rates upto 1.25 Gb/s LVDS using differential I/O pairs) provides a rich set of features like Input Serial-to-Parallel Logic Resources (ISERDES), Output Parallel-to-Serial Logic Resources (OSERDES), Input/Output Delay Element (IODELAY), Digitally Controlled Impedance (DCI), and Buffers with dual-ports, multi-rate & dual-edge-clockable capabality [Xil07b]. Also, advanced deskew capability (ChipSync Source-Synchronous Interfacing Logic) along with dedicated I/O & clocking resources, and built-in serializer/deserializers provide various means for efficient and reliable inter-FPGA transfers. In addition, programmable 8-24 channel RocketIO GTP transceivers (supporting different DC levels of operation) help to achieve transfer rates as high as 3.2Gb/s. Also, based on the amount of traffic flowing across the FPGA boudary, the communication width and number of such parallel links can be varied (Figure 12.1).

Apart from the implementation-specific architectural modifications, the following changes are required for the extension and application of the Networks-on-Chip communication architecture across multiple FPGAs. The ideas discussed below are the most pertinent ones related to the research in this disseratation and are described in an abstract fashion, and more research is required on the exact implementation based on the tradeoffs desired.

(i) Packet Control Information: In a Multi Local Port Router, the packets are routed in an XY fashion, and after reaching the destination router module, the packets are switched to the appropriate logic port [SV06b]. On similar lines, in case of Multi-FPGA systems, the packets follow the XY routing scheme when traversing the FPGA boundaries and reach the destination FPGA module. In order to provide this capability, modifications are necessary with regard the control information present in the header flit. There are two scenarios that are possible in a multi FPGA design environment. Packets can be generated (source core) and consumed (destination core) inside the same FPGA. Alternately, packets need to be crossing FPGA boundaries, in which case, there are two levels of XY routing happening. First, from source FPGA to destination FPGA and second, at source (destination) FPGA the normal XY routing inside the mesh within the source (destination) FPGA.

In order to support the two levels of hierarchies during routing of packets, we require a combination of a single-flit header and two-flit header packets. The case when the

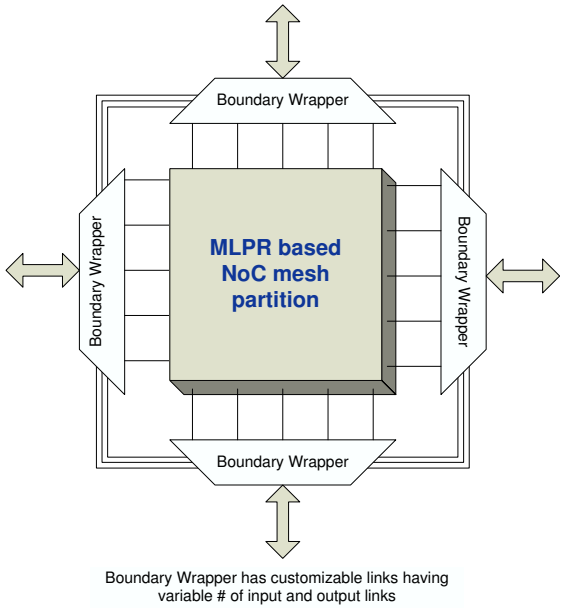


Figure 12.1: Multi-FPGA based Networks-on-Chip

transfers happen within a single FPGA resembles the normal scenario (traditional on-chip network) and hence, the packets in such a case need only a single header flit. On the other hand, Inter-FPGA transfers require the ability to route packets in two stages - first, between the FPGAs (from source FPGA to destination FPGA) and second, between the routers inside a single FPGA. In order to achieve, we require control information with two header flits - the first header flit will represent the mesh co-ordinate of the destination FPGA, while the second header flit will represent the (normal case) header flit having the co-ordinate information of the destination router/port in the destination FPGA. While effecting the inter-FPGA transfers, the packets are routed to the FPGA corner (boundary) in an XY fashion dictated by the destination FPGA co-ordinate.

(ii) Boundary Wrapper: The block diagram of an Multi-FPGA based NoC mesh is shown in Figure 12.1. Apart from packet level and routing modifications, the mesh inside each FPGA is surrounded by a wrapper present in all four corners of the FPGA. The wrapper provides multiple features based on the latency-bandwidth-power tradeoffs that are desired. All the four wrappers in the four corners of an FPGA are interconnected completing a pipelined circular links in both directions (clockwise/anti-clockwise) (Figure 12.1). Each of the link between the adjacent boundary wrappers is customizable according to the application at hand, wherein the traffic pattern will determine the size as well as the presence of the link.

Figure 12.2 shows a block-level description when implmenting an interconnection of two FPGAs. In order to appreciate the effectiveness of the circular pipelined path, consider a packet arriving at the eastern corner of an FPGA and the local NoC inside the FPGA is a n×n mesh. If the packet has to eject out the FPGA through the western corner (the opposite end of the FPGA)*, instead of making the packets follow the normal path inside the n×n (clock latency gets increased by n times), the pipelined circular link forms a kind of expressway. Thus, the packets bypass the n×n network and reach the other corner of the FPGA via the circular link. The path taken is indicated as a dotted red line in Figure 12.2. Note that the path shown is for illustration only and it is also possible to use the top (north) boundary wrapper in place of the bottom (south) boundary wrapper. This is valid because the circular pipelined links are available in both directions and hence, it is possible to have both clockwise and anti-clockwise paths established dynamically, with the only restriction in the form of the availability of buffers, which again is traffic dependent.

It must be noted that the corner-to-corner FPGA links have longer delay and hence, the wrapper must operate in an asynchronous manner so as to tolerate the worst case delay of all the links inside the circularly-interconnected wrapper, while being independent of the n×n mesh. Also, a Multi-FPGA system can have a heterogeneous mix of FPGAs (generation/size/speed grade) and the IO transceivers of each FPGA can operate at varied data rates. This necessitates the use of asynchronous type of buffers in the wrappers for interconnection between various FPGAs, while not sacrificing the throughput and the performance. In addition, various architecural improvements are possible in a boundary wrapper - (a) buffering the intermittent packets and initiating burst mode of inter-FPGA transfers. This results in the creation of macro packets so that the handshakes happening for each packet across FPGAs can be minimized. At the same time, the overheads for macro-packet creation and dis-assembly must be adequately addressed and taken into account in the cost function(s). (b) incorporating various packet encoding schemes in order to minimize the number of bit-changes when traversing FPGA boundaries, which can have significant impact on the overall power consumption [BV06]..

Also, each corner-wrapper can be made to have multiple channels (wherein, the upper-bound is dictated by the number of columns/rows of the mesh present in the respective corner), and again, the width of each channel can be made variable. The

*Without loss of generality, similar arguments are applicable in the cases, when the packet needs to pass through the northern or the southern corner of the FPGA.

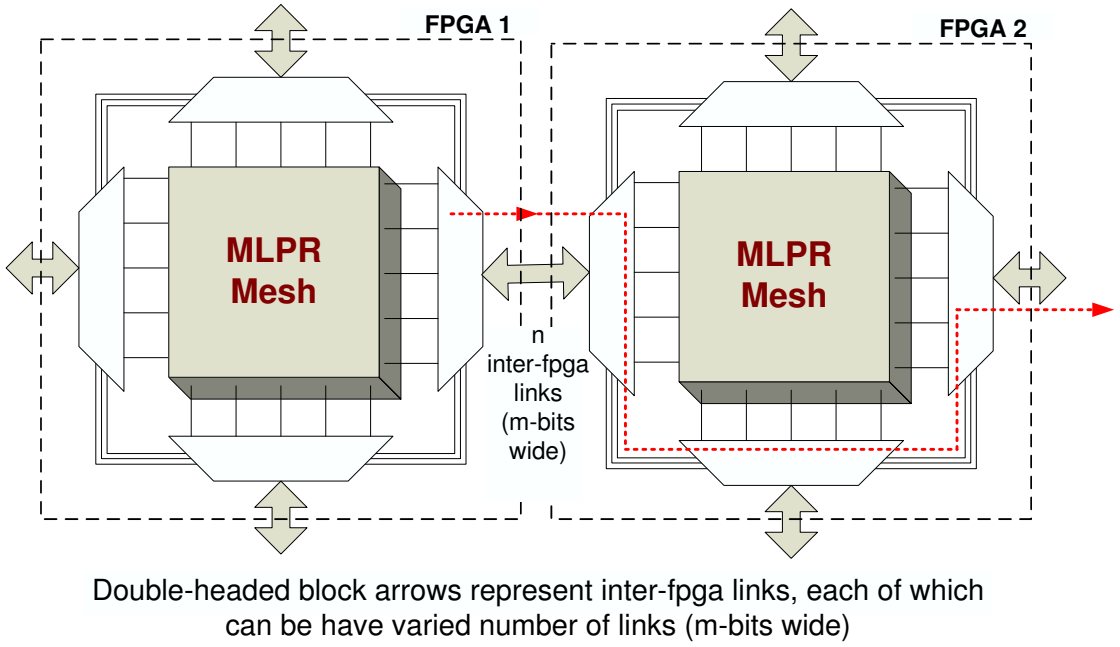


Figure 12.2: Illustration of an NoC on a Multi-FPGA having customizable inter-FPGA links

particular choice on the number of channel(s) and the width of the channel(s) is implementation specific and the decision are to be made based on the latency-power-bandwidth tradeoffs possible/desired. In this context, it is also possible to have dedicated links between various FPGAs to bypass the mesh-interconnected FPGAs similar to the seletive long-link insertion ideas found in [OM05, OMLC06, HCZ+05, HZC+06, WPM05]. Though this idea may find use in the form of selective insertion to facilitate high bandwidth traffic, this idea breaks the common standard interface available in the form of an mesh-inteconnected NoC and also, it is greatly dependant on the routing constraints in the PCB layout.

Mapping Algorithm

The mesh form of interconnection topology is preferred in FPGAs and the latest FPGAs (from Xilinx & Altera) offer exceedingly larger pin count. Hence, there is no restriction in realizing mesh-based SoC topologies, thus helping us to achieve a more efficient communication (bandwidth flow) between multiple FPGAs. Also, throughout the dissertation, we restrict our experimentation and discussion to mesh-based topologies, which again is the preferred choice to interconnect routers in an on-chip network implemented inside

FPGAs. The mesh-favored scenario greatly simplifies the Networks-on-Chip architecture generation phase. This is because of the fact that the FPGA count and the size of each FPGA is a specification at hand, and hence an exhaustive topology generation phase (eg., similar MLPR topology generation of optiMap [SV06b]) is absent at FPGA level. Based on the total number of FPGAs that are to be interconnected following an NoC type of communication backbone, the effort is now required largely during the mapping phase. Even the process of mapping gets greatly simplified to a procedure of the identification of the clusters[†] that will go inside each of the mesh-interconnected FPGAs (already defined in terms of logic capacity and topology), while optimizing the intended cost metric. This is not an restriction per se and the algorithm can be extended to explore various combinations taking the FPGA count and the size of FPGAs into equation. At the same time, the choice in terms of the combination of FPGAs (both count and capacity) is very restricted when realizing a Printed Circuit Board design having set of FPGAs (Multi FPGAs), due to the direct influence on the cost.

Changes Required: In the new multi FPGA environment and in the context of a Networks-on-Chip, an FPGA is equivalent to a Multi Local Port Router proposed in Chapter 4. In other words, inside each FPGA, multiple cores can be synthesized and mapped, thus, representing an analogous condition of having an MLPR having varied port count. The only constraint present here is in the form of the total logic capacity of the individual FPGA, which restricts the number of cores that can go onto that FPGA. Hence, with regard to the mapping algorithm(s) proposed earlier in Chapters 6, 7, 9 & 11, the actual change that is required is minimal. Given the set of constraints with respect to the individual FPGA size and the total FPGA count that are mesh-interconnected, the mapping algorithm(s) get simplified and will now populate the cores into each FPGA, while optimizing the latency/power metric. There is no change necessary with respect to the cost function, as in the new multi FPGA scenario, the intended mapping is to cluster the cores together locally inside an FPGA, while minimizing inter-FPGA transfers. At the same time, the availability and proximity of the Input-Output Pads (Pins) must be considered when clustering cores across multiple FPGAs. Also, topology generation inside an individual FPGA must take the IO pad/pin characteristics (which have direct impact on the latency/performance/power) into account when generating an appropriate MLPR-based mesh-topology. In this context, different packet encoding schemes (for the boundary wrapper) can be explored targetting the inter FPGA transfers in order to improve the overall power efficiency and performance [BV06].

[†] A cluster represents the group of IP cores populated inside a single FPGA.

As discussed in Chapter 9, different cluster configurations may provide different, but appreciable tradeoffs in terms of the overall latency, bandwidth or the total power consumption. But, with the inter FPGA transfers, the impedance seen is hundred times larger [Xil07a] and hence must be kept to the bare minimum possible for an efficient multi FPGA SoC.

Extensions with respect to the other topologies are not complex and will require appropriate changes to the latency and the packet count estimation metrics. At the same time, the instances wherein an FPGA is not large enough to accommodate even a single IP core is not in the scope of the Networks-on-Chip research. Such a scenario require hardware partitioning techniques [JKK00, Hau98, Hau95, JKK03, JKK02], which may not be efficient or adaptable in the System-on-Chips having on chip interconnection networks. This is particularly true because the Network Interfaces are defined conforming to a commonly-agreed packet standard and hence, in the worst case, the desired partitioning of an IP core may not be even possible[‡]. Also, when a core gets partitioned, there will be a host of problems created with respect to the addressability of cores and more importantly with respect to the buffering strategies required, amidst stringent throughput and latency constraints. All these aspects will force the final System-on-Chip to be sub-optimal with respect to both the resource usage and the system performance. In such cases, a complete redesign of the IP core (taking the logic capacity constraints of the FPGAs into account) or the use of an FPGA with larger logic capacity may be the only effective solutions possible.

12.3 Conclusion

We present a holistic and a bird's eye view of the issues to be tackled in the extension and application of Networks-on-Chip kind of system interconnection in a Multi FPGA environment. The topology exploration phase gets simplified due the adoption of the mesh-based interconnection topology as the preferred choice and only a minimal change in the cost function(s) is required with respect the mapping algorithms. Communication inside a single FPGA does not require any change, while the inter FPGA transfer requires a wrapper and associated & suitable changes to the control information of the packet(s). The proposed ideas provide a good scope for a possible direction of work in the future, in terms of the implementation of the framework.

[‡]This is because the core boundary requires a Network Interface to achieve effective switching of packets

Chapter 13

Conclusions

Future System-on-Chips (SoCs) will have billions of transistors integrated on a single die. In order to increase the productivity, decrease costs and meet stringent Time-to-Market constraints, a platform-based design methodology is preferred having a larger percentage of design re-use. In such cases, modern SoCs will be realized as a complex integration of various Intellectual Property cores.

In addition to the global interconnect related issues, the traditional and popular shared-bus approach is losing its steam on the scalability aspect. Networks-on-Chips (NoCs) are the modern day answer to address the compounded problems in the existing SoCs, with the data transfer happening in a distributed packet-switched fashion, using smaller and better predictable interconnects.

In addition to ASICs, modern FPGAs [Xil06b] provide greater scope for realizing System-on-Chip designs, because of the availability of several Intellectual Property (IP) cores, including embedded hard & soft processors, memory, Digital Signal Processing (DSP) and Input/Output (I/O) Transceiver cores [Xil05]. Due to the scalability issues present in the use of shared-bus, NoC is being preferred as the communication backbone for integration of IP modules. But, despite the several advantages, being a typical shared network, an NoC throws challenges on multiple fronts including area overheads, hop-based transfer increasing the latency, creation of congestion and deadlocks, and most importantly bandwidth violations across several link of an NoC.

In this thesis, we present novel designs that target area reduction and improvement in both power and performance. An important contribution of the thesis is the concept of a heterogenous Networks-on-Chip mesh, having routers with varied port count. We extend

this novel design to included the ability to multicast. In addition, use the proposed router designs as the communication backbone of the intended NoC mesh, we present several NoC topology generation and mapping algorithms that optimized for various metrics including latency, bandwidth and power. In addition, algorithms exploited the the heterogenous nature of multi port routers and minimized the overall packet count, resulting in improved performance and reduction in power.

13.1 Contributions

The specific contributions of the research in the dissertation are summarized as follows:

1. An area-efficient router design for implementing Networks-on-Chip (NoC) on reconfigurable devices like FPGAs. We present the architectural optimizations that resulted in the reduction in the logic area usage by the router. The router design is capable of establishing and servicing multiple requests simultaneously without any performance penalty.

2. We target area reduction by reduction of number of routers in an NoC mesh. An novel architectural modification to the traditional Networks-on-Chip architecture, giving rise to a Multi Local Port Router (MLPR) that is capable of handling multiple logic cores simultaneously, without compromising the performance of the system. The new Multi Local Port Router (MLPR) provided many-fold advantages including reduction in area, power, transit time $\&$ congestion, and most importantly, bandwidth optimization, resulting in an efficient and high performance NoC design. Essentially, the MLPR is a marriage between switch-based and router-based interconnection network. We exhaustively analyze the merits and issues involved in adopting MLPR into the NoC design flow.

3. We present the complexity involved in generating efficient NoC configurations in an MLPR-based NoC. As a proof-of-concept, we present an exhaustive search algorithm to find the optimum mesh based Network-on-Chip configuration (using Multi Local Port Routers) and the final mapping, reducing the overall execution time. The results indicate the significant performance gain that is obtained.

4. Since an exhaustive search is infeasible for larger NoC systems, we present a fast

mapping algorithm (cMap) based on the heuristics observed from the optiMap algorithm. The algorithm uses an forced-directed approach to iteratively expand the mesh design, as the cost is reduced. The algorithm finds the near-optimal results within a couple of seconds.

5. We extend the Multi Local Port Router to incorporate the multicast ability into them. We present the architectural modifications necessary to accomplish this facility. This extension bridges one of the feature-gap that exists between an NoC and a shared-bus.

6. Using the modified optiMap algorithm, we find energy-efficient NoC configuration exploiting the available multicast feature. There is a significant reduction in the amount of data traffic across the network, which directly result in a drastic reduction in the power consumption of the design. In addition, we observe significant performance gains due to reduction in the overall transit time.

7. In addition to the power reduction achieved by the direct reduction of packet flowing in the NoC mesh, we experiment and present the intra-port power savings possible. We observe significant variation in the amount of power required to switch packets between different ports, thereby, giving an opportunity to achieve more power savings by careful selection of port when mapping a design core to an MLPR.

8. We perform extensive power and IR analysis on the multi port routers and find the shortcomings in using larger multi port routers from an power efficiency angle. We observe that there is an increase in the average power with increased port count. Through exhaustive rail analysis, we infer that large multi port routers introduce greater amount of IR violations. Hence, a heterogenous mix of smaller multi port routers will provide a better tradeoff in terms of performance and power.

9. We highlight the necessity to handle the varying nature of inter-communication and bandwidth flow between the cores of a System-on-Chip, thereby, requiring a better approach to avoid possible bandwidth violations. For a given topology, we present technique to estimate the Minimum BandWidth Guarantee (MBWG) required for a given topology and , We analyze the impact of maximum port count on the MBWG required and extend the algorithm to find the NoC architecture minimizing the MBWG. The results demonstrate the need for a highly application-specific NoC architecture generation, catered to the task structure(s) at hand. The knowledge of the maximal

bandwidth variation across links is highly inevitable during the NoC design, helping the designers to better predict the possible violations that may arise because of dynamic inter-communication.

10. We propose and discuss the necessary design changes that may be required for adopting an on-chip network kind of core integration in a multi chip based System-on-Chip design environment, wherein a set of FPGAs are interconnected following a mesh-based topology.

13.2 Salient Inferences

In this section, we summarize the notable inferences that can be drawn out of the dissertation.

- Networks-on-Chip implementation on Reconfigurable Devices requires an efficient implementation of the communication backbone comprising of the routers or the switch elements.
  - This is more so because of the balanced and predictable (global) interconnects available across the reconfigurable fabric, along with a better managed clock management mechanisms, which is not the case in an ASIC based NoC design.
  - Also, premium availability of logic area in FPGAs motivates (forces) the logic area reduction has to be a prime target for optimization, even at the expense of performance due to the use of longer interconnects.
  - Field Programmable Gate Arrays are inherently suited for a mesh based interconnection of routers that forms the communication backbone of the Networks-on-Chip.
- Selective clustering or grouping of cores is more efficient through the careful selection of the multi port routers during topology generation and mapping, thus, enabling us to realize a heterogenous mix of routers.
- Multi-port router based NoC architecture generation provides an opportunity to achieve tradeoffs with respect to various metrics including area, latency, power and bandwidth.

- Heuristic and fast mapping techniques are necessary to handle the complexity present in the System-on-Chips involving hundreds of cores, and hence, requires cost functions targeting various optimization metrics.
- Router level multicasting is possible with least overheads in terms of the area and the operating frequency, helping a great deal in the avoidance of switching of same data packet multiple times (to the ports in a single multi port router).
- Reduction in the overall packet count in the Networks-on-Chip mesh may not necessarily result in the reduction in the overall latency, but has an direct and significant impact on the packet-switching power required by the communication network.
- Variations in the amount of power required to switch packets between various ports of a multi port router are significant. Hence, depending the amount of data transfer happening across the various port of a multi port router, a careful selection of the port during IP mapping phase will result in a pronounced reduction in the total power.
- Large multi port routers, despite being efficient in terms of area usage and latency, are poor candidates for power efficiency. In addition to the average power increase, there are large IR drop violations, requiring a concerted effort (eg., widening of supply rail stripes, . . .) and thus, making the NoC design flow more complex (in ASICs).
- In order to avoid bandwidth violations, the bottlenecks in the shared links of the mesh must be reduced, either by distributing the flow across many links (single port router) or by reducing the usage of the links (larger port router).
- Thus, we have contrasting requirements and possibilities in terms of the port count, and hence, an application-specific tailoring of the Networks-on-Chip architecture is required, based on the given application and the target implementation technology & platform.

In summary, the dissertation presents novel and efficient router designs, supported by the NoC architecture generation algorithms. Despite having an FPGA bias, the ideas proposed in this research are equally applicable to ASICs, thus providing a small but significant leap in the quest of an efficient NoC architecture design.

Chapter 14

Future Directions

Being an emerging domain of research, Networks-on-Chip offers larger scope of original research. As explained in Section 2.4, better techniques, designs and algorithms are necessary for all phases of design starting from communication architecture design (involving nittygritties with regard to buffer selection, flow control and routing) to application mapping [OHM05]. In this chapter, we present some of the possible directions of work that can be pursued in line with the research presented in this dissertation. We also cite some of the recent and relevant works along with the discussion [BM06].

Floorplan-inclusive NoC Architecture Generation: In this dissertation, the topology generation and mapping are done in isolation, without taking the effect of floorplan and placement of cores in FPGAs (& ASICs) into consideration. The relative positioning of the cores has a direct impact and affects the available choice of the multi-port routers. Floorplan information is necessary to account for variations created, covering the various metrics including the performance and the power. The longer the length of the interconnect, larger will the delay and the amount of wire capacitance switched, thus resulting in a reduction in the operating frequency, while increasing the power. In Chapter 10, we approximate and normalize the amount of link power consumed in relation to the intra-port packet transfers (20% increase) [SCK06]. This is valid only in the context of having an uniform and predictable value for the interconnect length.

In this work, we restrict our experimentation to a mesh type of interconnection topology that is observed to be highly efficient for an FPGA-based implementation. Also, topologies like the torus, the folded torus and the octagon can be explored to make

the application-specific generation of the Networks-on-Chip architecture more efficient. It is argued that power optimizations in significant proportions are possible with torus topology [HCZ+05]. Localized customization of channels and buffers along with optimal slot allocation are possible only if the slack possible can be estimated accurately, which in turn is highly floorplan and interconnect (link) specific. Also, the bandwidth constraint is dictated by the router-to-router interconnects. Better estimation and optimization of the above parameters are possible only through accurate modeling and data from the floorplan [MMA+06,SC06b,AMC+06,SC06a].

Globally Asynchronous Locally Synchronous Design: In Chapter 10, we show the presence of variations up to 25% in the amount of power that is used in switching packets from different source ports to the various combinations of other ports. In a similar manner, due to the relative placement of the input and output ports, the critical path between the various ports (through the central cross point matrix) varies over an appreciable range. This variation can be exploited with a provision and capability of having multiple operating frequencies for the Networks-on-Chip communication backbone. In order to achieve this, a router node must be capable of accepting and sending data at multiple frequencies, thereby, requiring the buffer or the FIFO elements to handle multiple frequencies. With regard to the reconfigurable devices, the Xilinx Virtex 4 FPGAs have the provision to derive as much as sixteen independent clocks [Xil06a]. But, this comes at a price in term of extra routing and logic overheads, but, more importantly, it results in a larger capacitance switched and hence larger average power consumption. Further, multiple clocks will complicate and throw more issues to tackle with regard to data synchronization [DGS04]. Also, routing multiple clocks to different modules is more complex involving increased associated overheads (area/power), along with a large number of reliability issues and meta-stability problems to contend with [DGS06].

Broadcast Router: In this dissertation, we have present the router design in which it is possible to multicast to various combinations of ports present in a multi port router, thereby, making is a broadcast-capable router at a node (router) level. In addition to such a node-restricted multicast capability, designs and strategies are possible and necessary to extend this versatile capability and make the Networks-on-Chip design to support multicast to various combinations of routers present in the communication backbone. In other words, router decoding logic and the control information embedded in a packet can be modified to incorporate the capability to multicast/broadcast

to multiple routers (ports).

Enforcing Coherency in Multi-processor Networks-on-Chip: An important component of realizing multi-processor system-on-Chips is maintaining a consistent and coherent memory model. In addition to the microprocessor cores, the dedicated/shared memory and its associated controllers increase the complexity of the memory architecture. Also, provisions of out-of-order delivery of packets and the associated re-ordering of the packets complicate the situation further. With the use of multi-port routers, preferences given to intra-port transfers will reduce the need for out-of-order packet delivery. Hence, this underlying capability can be exploited to enforce architecture generation that provides an effective tradeoff in terms of observable metrics like performance and power against reduction of situations requiring buffering and processing of packets that are received out of order [MABM06].

Architectural & Algorithmic Improvements: The switch or the router elements represent the key elements in the communication backbone defined in an Networks-on-Chip. There is a larger scope for improvement at every aspect of the router design including the flow control, routing and buffering strategies. It is argued in [AMC+06] that unlike the earlier assumptions, a majority of the power is utilized by the sequential elements that act as intermediate storage points (buffers). A small variation in the size of the flit-width has an enormous impact on the power. Here, the tradeoff is present in the form of the latency as well - larger the flit width, lesser is the number of flits (i.e., reduced latency). In the same context, the NoC design flow must take into consideration the possibility of allowing stream or burst transfers. Hence, if the overheads caused due to the packetization and depacketization steps are to be minimized and overshadowed, then the size of the payload of the packet must increase [AMC+06, Bha06]. There exists a plethora of possibilities in terms of the customization of the buffers and flow control for the application at hand [CMR+06, VN06a, TMG+06, KPKJ07].

In addition to the packet-switched architecture, the circuit-switched architecture can be exploited locally to improve the performance, by avoiding the latency created due to the packetization-depacketization process. The heavy data traffic between intra-port-mapped cores can be made to go through the circuit-switched layer, avoiding the packet creation and the successive joining of the payload of the packets. But, such an approach will throw more overhead in terms of the logic area and the operating power, thus, forcing yet another tradeoff point. Also, such an architecture is better for

a highly intermittent traffic with a large aggregated bandwidth [CSC06, HN06, GJ05].

Also, improved and adaptive routing schemes are necessary to offset the compounded queuing problems created due to congestions. But, care must be taken to avoid the situations resulting in the creation of deadlocks and livelocks. Also, the revised schemes must be as distributed as possible in nature, in terms of the routing decision making that need to taken based on congestion spots which are created in a more dynamic fashion [TMG+06, OM06, XWHC06]. Also, there is a greater stress for the system design, taking the link capacity and Quality-of-Service (Best Effort/ Guaranteed Throughput) into consideration [LB06, NPK+06, KNP06, PHKC06, DKS+06, MABM06, GWB+06, LT06].

Æthereal architecture supports multiple cores to be attached onto the Network Interface (NI) kernel ports, but, not directly onto the router node. In spite of having multiple NI kernel ports, the single link (which is time-multiplexed between different channels) between the router and NI creates a large bottleneck, preventing simultaneous parallel connections from the router. Further, the connections between NI kernel ports suffer from a lengthy transaction going through a scheduler. Using Æthereal as the base architecture, Hansson et al present a scheme for combined mapping, routing and slot allocation [HGR05]. Thus, apart from the multi-port router architectures, designs can be custom-tailored so as to share the network interfaces [RJH04], thereby, still optimizing the Networks-on-Chip architecture for logic area. But, it must be understood that sharing of network interface is beneficial only in scenarios wherein multiple logic modules mapped (thus sharing the network interfaces) are not be competing for channel access simultaneously, in which case, the advantage of parallel communication present in a Networks-on-Chip kind of architecture may be lost.

The unused links, channels and ports can be removed, which will result in substantial savings in terms of the area and power. So far in this dissertation, when optimizing for latency, we have concentrated on the global latency metric, without taking the local latency into consideration. In other words, we did not bring the pair-wise latency constraints into the equation, when generating the topology and mapping. In this process, it is possible for latency violations to occur during packet transfers between pair(s) of cores. If this situation is not taken care and resolved during the Networks-on-Chip architecture generation, it will result in reliability issues, nullifying any gains that may be expected. Similar arguments will hold true when there exists a hotspot kind of traffic, thereby, resulting in a host of issues related to the

power and the performance. In this context, the highly localized activity and the resulting IR drops must be foreseen and addressed properly and adequately.

Hence, a comprehensive NoC architecture generation algorithm (in addition to addressing the above issues), must incorporate capabilities for custom tailoring (eg., removal of unused or least used logic consumed by various channels/ports). By this way, it can provide NoC architecture generation with varying tradeoffs, and when coupled with a *what-if analysis*, will provide the system designers with the invaluable insight regarding the various options available when creating a Networks-on-Chip based System-on-Chip. On top of the architectural changes suggested above, it is necessary to make the designs complaint with the various standards including OCP/VCI to provide a capability for seamless IP integration.

System Level Optimizations: Apart from the router design, and task graph based topology generation & mapping, latest configurable-logic based platform/embedded systems require an efficient hardware-software co-design approach. With greater stress on the design re-use and with larger usage of microprocessor cores, efficient usage of the limited logic available is necessary to increase the design productivity, while keeping the costs down. Networks-on-Chip is gaining much attention in this aspect, wherein the application at hand is partitioned into the synthesized-hardware and software components targeting improvement in both power and performance [VN06b]. The primary advantages offered by a Networks-on-Chip in terms of the large aggregate bandwidth and improved computational power provide a larger design space to explore in a hardware-software co-design methodology. In this context, efficient task allocation and scheduling techniques are required to maximize the resource utilization with improved performance [HCY+07]. Moreover, with improved dynamic reconfiguration capabilities offered by the latex Xilinx Virtex 4/5 FPGAs, there exists larger scope for improved system designs. Added to all these features/capabilities, dynamic voltage scaling and frequency scaling, along with the compiler-directed design optimizations, will help in the implementation of an highly optimal System-on-Chip design [CLK06, CLKI06].

.

Bibliography

[ABPvG01] Amir H. Ajami, Kaustav Banerjee, Massoud Pedram, and Lukas P. P. P. van Ginneken. Analysis of Non-Uniform Temperature-Dependent Interconnect Performance in High Performance ICs. In *DAC '01: Proceedings of the 38th conference on Design automation*, pages 567–572, 2001.

[ACP04] G. Ascia, V. Catania, and M. Palesi. Multi-objective Mapping for Mesh-based NoC Architectures. In *Intl. Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS'04)*, pages 182–187, 2004.

[ALMM05] Alexandre M. Amory, Marcelo Lubaszewski, Fernando Gehm Moraes, and Edson I. Moreno. Test Time Reduction Reusing Multiple Processors in a Network-on-Chip Based Architecture. In *DATE*, pages 62–63, 2005.

[Alt07] Altera Inc. http://www.altera.com, 2007.

[AMB06] AMBA Specification (ver 2.0). ARM Inc. In *http://www.arm.com/products/solutions/AMBAHomePage.html*, 2006.

[AMC+06] Federico Angiolini, Paolo Meloni, Salvatore Carta, Luca Benini, and Luigi Raffo. Contrasting a NoC and a Traditional Interconnect Fabric With Layout Awareness. In *DATE '06: Proceedings of the conference on Design, Automation and Test in Europe*, pages 124–129, 2006.

[Arc05] The Coreconnect Bus Architecture. IBM Corporation. In *http://www.chips.ibm.com*, 2005.

[ARG05] Santiago González Pestana Om Prakash Gangwal Edwin Rijpkema Paul Wielage Andrei Radulescu, John Dielissen and Kees Gooses. An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Programming. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 24(1):4–17, Jan 2005.

[Art06] Arteris NoC Solution 1.4. http://www.arteris.com, May 2006.

[ASTBN04] Tapani Ahonen, David A. Sigüenza-Tortosa, Hong Bin, and Jari Nurmi. Topology Optimization for Application-Specific Networks-on-Chip. In *SLIP '04: Proceedings of the 2004 international workshop on System level interconnect prediction*, pages 53–60, 2004.

[ATT05] Syed M. Alam, Donald E. Troxel, and Carl V. Thompson. Thermal Aware Cell-Based Full-Chip Electromigration Reliability Analysis. In *GLSVLSI '05: Proceedings of the 15th ACM Great Lakes symposium on VLSI*, pages 26–31, 2005.

[Axe03] Axel Jantsch and Hannu Tenhunen (Eds.). *Networks on Chip*. Kluwer Academic, 2003.

[BdM01] Luca Benini and Giovanni de Micheli. Powering Networks on Chips: Energy-efficient and Reliable design for SOCs. In *Design Automation Conference*, pages 33–37, 2001.

[BdM02] Luca Benini and Giovanni de Micheli. Networks on Chips: A New SOC Paradigm. In *IEEE Computer*, pages 70–78, Jan 2002.

[BDM+04] Andrei Bartic, Dirk Desmet, Jean-Yves Mignolet, Théodore Marescaux, Diederik Verkest, Serge Vernalde, Rudy Lauwereins, J. Miller, and Frédéric Robert. Network-on-Chip for Reconfigurable Systems: From High-Level Design Down to Implementation. In *FPL*, pages 637–647, 2004.

[Bha06] Prasun Bhattacharya. Comparison of single-port and multi-port nocs with contemporary buses on fpgas. Master of science thesis, University of Cincinnati, Cincinnati, Ohio, United States, March 2006.

[Bil74] Alberto A. Bilotti. Static Temperature Distribution in IC Chips with Isothermal Heat Source. *IEEE Transaction on Electron Device*, ED-21 (3):217–226, Feb 1974.

[BJM+05] Davide Bertozzi, Antoine Jalabert, Srinivasan Murali, Rutuparna Tamhankar, Stergios Stergiou, Luca Benini, and Giovanni De Micheli. NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip. *IEEE Trans. Parallel Distrib. Syst.*, 16(2):113–129, 2005.

[BM06] Tobias Bjerregaard and Shankar Mahadevan. A Survey of Research and Practices of Network-on-Chip. *ACM Computing Survey*, 38(1):1, 2006.

[BMN+03] T.A. Bartic, J.-Y Mignolet, V. Nollet, T. Marescaux, D. Verkest, S. Vernalde, and R. Lauwereins. Highly Scalable Network on Chip for Reconfigurable Systems. In *Proceedings of the International Conference on System-On-Chip 2003*, pages 79–82, Nov. 2003.

[BMSVH99] Kaustav Banerjee, Amit Mehrotra, Alberto Sangiovanni-Vincentelli, and Chenming Hu. On Thermal Effects in Deep Sub-Micron VLSI Interconnects. In *DAC '99: Proceedings of the 36th ACM/IEEE conference on Design automation*, pages 885–891, 1999.

[BOSW94] D. Bhagavathi, S. Olariu, W. Shen, and L. Wilson. A Unifying Look at Semigroup Computations on Meshes with Multiple Broadcasting. *Parallel Processing Letters*, 4(1), 1994.

[BPA01] Kaustav Banerjee, Massoud Pedram, and Amir H. Ajami. Analysis and Optimization of Thermal Issues in High-Performance VLSI. In *ISPD '01: Proceedings of the 2001 international symposium on Physical design*, 2001.

[BTA93] Jonathan Babb, Russ Tessier, , and Anant Agarwal. Virtual Wires: Overcoming Pin Limitations in FPGA-based Logic Emulators. In *IEEE Workshop on FPGAs for Custom Computing Machines*, pages 142–151, 1993.

[BV06] E. Beigne and P. Vivet. Design of On-chip and Off-chip Interfaces for a GALS NoC Architecture. In *ASYNC '06: Proceedings of the 12th IEEE International Symposium on Asynchronous Circuits and Systems*, page 172, 2006.

[Cad07a] Cadence® Encounter® digital IC design platform. SoC Encounter™: RTL-to-GDSII system. http://www.cadence.com/products/digital_ic/soc_encounter/index.aspx, 2007. Data Sheet.

[Cad07b] Cadence® Encounter® digital IC design platform. VoltageStorm® Power and Power Rail Verification. http://www.cadence.com/products/dfm/fireandice/index.aspx, 2007. Data Sheet.

[CC05] Li-Hsun Chen and O.T.-C. Chen. A Hardware-Efficient FIR Architecture with Input-Data and Tap Folding. In *IEEE International Symposium on Circuits and Systems (ISCAS '05)*, 2005.

[CCCS90] Y. C. Chen, W. T. Chen, G. H. Chen, and J. P. Sheu. Designing Efficient Parallel Algorithms on Mesh-Connected Computers with Multiple Broadcasting. *IEEE Transactions on Parallel and Distributed Systems*, 1990.

[CGMP99] Shang-Tse Chuang, Ashish Goel, Nick McKeown, and Balaji Prabhakar. Matching Output Queueing with a Combined Input Output Queued Switch. In *INFOCOM*, pages 1169–1178, 1999.

[Cha84] Daniel M. Chapiro. *Globally Asynchronous Locally Synchronous Systems*. PhD thesis, Stanford University, 1984.

[CKPP99] Jong Hyuk Choi, Bong Wan Kim, Kyu Ho Park, and Kwang-Il Park. A Bandwidth-Efficient Implementation of Mesh with Multiple Broadcasting. In *International Parallel Processing*, 1999.

[CLK06] Guangyu Chen, Feihui Li, and Mahmut Kandemir. Compiler-Directed Channel Allocation for Saving Power in On-Chip Networks. In *POPL '06: Conference record of the 33rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 194–205, 2006.

[CLKI06] Guangyu Chen, Feihui Li, Mahmut Kandemir, and Mary Jane Irwin. Reducing NoC Energy Consumption Through Compiler-Directed Channel Voltage Scaling. In *PLDI '06: Proceedings of the 2006 ACM SIGPLAN conference on Programming language design and implementation*, pages 193–203, 2006.

[CLP00] S. Coric, I. Latinovic, and A. Pavasovic. A Neural Network FPGA Implementation. In *5th Seminar on Neural Network Applications in Electrical Engineering (NEUREL 2000)*, pages 117–120, Belgrade, 2000.

[CLRK99] Danqing Chen, Erhong Li, Elyse Rosenbaum, and Sung-Mo (Steve) Kang. Interconnect Thermal Modeling for Determining Design Limits on Current Density. In *ISPD '99: Proceedings of the 1999 international symposium on Physical design*, pages 172–178, 1999.

[CM94] C. P. Cowen and S. Monaghan. A Reconfigurable Monte-Carlo Clustering Processor (MCCP). In *IEEE Workshop on FPGAs for Custom Computing Machines*, pages 59–65, 1994.

[CMR+06] Martijn Coenen, Srinivasan Murali, Andrei Ruadulescu, Kees Goossens, and Giovanni De Micheli. A Buffer-Sizing Algorithm for Networks on Chip using TDMA And Credit-Based End-To-End Flow Control. In *CODES+ISSS '06: Proceedings of the 4th international conference on Hardware/software codesign and system synthesis*, pages 130–135, 2006.

[Cor99] B. Cordan. An Efficient Bus Architecture for System-On-Chip Design. In *IEEE 1999 Custom Integrated Circuits*, 1999.

[CP04] Jeremy Chan and Sri Parameshwaran. NoCGEN: A Template Based Resuse Methodology for Networks on Chip Architecture. In *17th International Conference on VLSI Design (VLSID'04)*, pages 717–720, 2004.

[CR93] Steven A. Cuccaro and Craig F. Reese. The CM-2X: A Hybrid CM-2 / Xilinx Prototype. In *IEEE Workshop on FPGAs for Custom Computing Machines*, pages 121–130, 1993.

[CSC06] Kuei-Chung Chang, Jih-Sheng Shen, and Tien-Fu Chen. Evaluation and design trade-offs between circuit-switched and packet-switched NOCs for application-specific SOCs. In *DAC '06: Proceedings of the 43rd annual conference on Design automation*, pages 143–148, 2006.

[DEW99] O. Diessel, H. ElGindy, and L. Wetherall. Efficient Broadcasting Procedures for Constrained Reconfigurable Meshes. In *Third Australasian Conference on Parallel and Real-Time Systems*, pages 85–88, Brisbane, Australia, Sep 1999.

[DGS04] Rostislav (Reuven) Dobkin, Ran Ginosar, and Christos P. Sotiriou. Data Synchronization Issues in GALS SoCs. In *10th International Symposium on Asynchronous Circuits and Systems*, pages 170– 179, April 2004.

[DGS06] Rostislav (Reuven) Dobkin, Ran Ginosar, and Christos P. Sotiriou. High Rate Data Synchronization in GALS SoCs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(10):1063–1074, 2006.

[DJR01] Santanu Dutta, Rune Jensen, and Alf Rieckmann. Viper: A Multiprocessor SOC for Advanced Set-Top Box and Digital TV Systems. In *IEEE Design and Test of Computers*, pages 21–31, Sept.-Oct. 2001.

[DKS+06] Masoud Daneshtalab, Ali Afzali Kusha, Ashkan Sobhani, Zainanabedin Navabi, Mohammad D. Mottaghi, and Omid Fatemi. Ant Colony Based Routing Architecture for Minimizing Hot Spots in NOCs. In *SBCCI '06: Proceedings of the 19th annual symposium on Integrated circuits and systems design*, pages 56–61, 2006.

[DKSL04] Sarang Dharmapurikar, Praveen Krishnamurthy, T.S. Sproull, and J.W. Lockwood. Deep Packet Inspection using Parallel Bloom Filters. *Micro, IEEE*, 24(1):52–61, 2004.

[DRW98] R.P. Dick, D.L. Rhodes, and W. Wolf. Tgff: Task graphs for free. In *6th Intl Workshop on Hardware/Software Codesign (CODES/CASHE '98)*, pages 97–101, 1998.

[DSY96] J. Duato, F. Silla, and S. Yalamanchili. A High Performance Router Architecture for Interconnection Networks. In *International Conference on Parallel Processing*, 1996.

[DT01] W.J. Dally and B. Towles. Route Packets, Not Wires: On-Chip Interconnection Networks. In *Design Automation Conference*, pages 684–689, 2001.

[DT04] W.J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. San Francisco: Morgan Kaufmann, 2004.

[DYN98] J. Duato, S. Yalamanchili, and L. Ni. *Interconnect Networks: An Engineering Approach*. IEEE CS Press, 1998.

[Dzu93] Dzung T. Hoang. Searching Genetic Databases on Splash 2. In *FPGAs for Custom Computing Machines, 1993. Proceedings. IEEE Workshop on*, pages 185–191, 1993.

[EM96] K. Echtle and A. Masum. A Multiple Bus Broadcast Protocol Resilient to Non-Cooperative Byzantine Faults. In *Annual Symposium on Fault Tolerant Computing*, pages 158 – 167, 1996.

[Eva94] J.B. Evans. Efficient FIR Filter Architectures Suitable for FPGA Implementation. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 41(7), 1994.

[fir07] Fire & Ice® QXC 3-D Extractor. http://www.cadence.com/products/dfm/fireandice/index.aspx, 2007. Data Sheet.

[FV00] K. Fall and K. Varadhan. The `ns` Manual. In *The VINT Project, UC Berkeley, LBL, USC/ISI, Xerox PARC*, 2000.

[GC98] Ming-Huang Guo and Ruay-Shiung Chang. A Simple Multicast ATM Switches Based on Broadcast Buses. In *IEEE Global Telecommunications Conference (GLOBECOM 98)*, volume 4, pages 2369–2374, Sydney, NSW, Australia, 1998.

[GCBM00] R. Gadea, J. Cerda, F. Ballester, and A. Macholi. Artificial Neural Network Implementation on a Single FPGA of a Pipelined On-Line Backpropagation. In *13th International Symposium on System Synthesis*, pages 225–230, 2000.

[GDG+05] Kees Goossens, John Dielissen, Om Prakash Gangwal, Santiago González Pestana, Andrei Radulescu, and Edwin Rijpkema. A Design Flow for Application-Specific Networks on Chip with Guaranteed Performance to Accelerate SOC Design and Verification. In *DATE*, pages 1182–1187, 2005.

[GDR05] K. Goossens, J. Dielissen, and A. Radulescu. The Æthereal Network on Chip: Concepts, Architectures, And Implementations. *IEEE Design and Test of Computers*, 22(5):21–31, Sept-Oct 2005.

[GG02] P. Guerrier and A. Greiner. A Generic Architecture for on-chip Packet-Switched Interconnections. In *DATE'02*, pages 250–256, 2002.

[GHK+90] Maya Gokhale, William Holmes, Andrew Kopser, Dic Kunze, Daniel Lopresti, Sara Lucas, Ronald Minnich, and Peter Olsen. Splash: A Reconfigurable Linear Logic Array. In *International Conference on Parallel Processing*, pages 526–532, 1990.

[GJ05] Cristian Grecu and Michael Jones. Performance Evaluation and Design Trade-Offs for Network-on-Chip Interconnect Architectures. *IEEE Trans. Comput.*, 54(8):1025–1040, 2005. Student Member-Partha Pratim Pande and Senior Member-Andre Ivanov and Senior Member-Resve Saleh.

[GWB+06] Zvika Guz, Isask'har Walter, Evgeny Bolotin, Israel Cidon, Ran Ginosar, and Avinoam Kolodny. Efficient Link Capacity and QoS Design for Network-on-Chip. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 9–14, 2006.

[Han03] Hangsheng Wang and Li-Shiuan Peh and Sharad Malik. Power-Driven Design of Router Microarchitectures in On-Chip Networks. In *Proc. International Symposium on Microarchitecture*, page 105116, 2003.

[Har05] Sriram Hariharan. Performance evaluation of on-chip communications in a network-on-chip system. Master of science thesis, University of Cincinnati, Cincinnati, Ohio, United States, February 2005.

[Hau95] Scott Hauck. *Multi-FPGA Systems*. PhD thesis, University of Washington, 1995.

[Hau98] Scott Hauck. The Roles of FPGAs in Reprogrammable Systems. *Proceedings of the IEEE*, 86(4):615–638, April 1998.

[HB95] Scott Hauck and Gaetano Borriello. Logic Partition Orderings For Multi-FPGA Systems. In *FPGA '95: Proceedings of the 1995 ACM third international symposium on Field-programmable gate arrays*, pages 32–38, New York, NY, USA, 1995.

[HB97a] Scott Hauck and Gaetano Borriello. An Evaluation of Bipartitioning Techniques. *IEEE Transactions on Computer-Aided Design of Integrated Circuits & Systems*, 16(8):849–866, August 1997.

[HB97b] Scott Hauck and Gaetano Borriello. Pin Assignment for Multi-FPGA Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits & Systems*, 16(9):956–964, September 1997.

[HBE98] Scott Hauck, Gaetano Borriello, and Carl Ebeling. Mesh Routing Topologies for Multi-FPGA Systems. *IEEE Transactions on VLSI Systems*, 6(3):400–408, September 1998.

[HCY+07] Wei-Hsuan Hung, Yi-Jung Chen, Chia-Lin Yang, Yen-Sheng Chang, and Alan P. Su. An Architectural Co-Synthesis Algorithm for Energy-Aware Network-on-Chip Design. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pages 680–684, 2007.

[HCZ+05] Yuanfang Hu, Hongyu Chen, Yi Zhu, Andrew A. Chien, and Chung-Kuan Cheng. Physical Synthesis of Energy-Efficient Networks-on-Chip Through Topology Exploration and Wire Style Optimizations. In *ICCD '05: Proceedings of the 2005 International Conference on Computer Design*, pages 111–118, 2005.

[HGR05] Andreas Hansson, Kees Goossens, and Andrei Rădulescu. A Unified Approach to Constrained Mapping and Routing on Network-on-Chip Architectures. In *International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, sep 2005.

[HJK+00] Ahmed Hemani, Axel Jantsch, Shashi Kumar, Adam Postula, Johnny Öberg, Mikael Millberg, and Dan Lindqvist. Network on Chip: An Architecture for Billion Transistor Era. In *IEEE NorChip Conference*, Nov 2000.

[HK03] D. Hecht and C. Katsinis. Performance Analysis of a Fault-Tolerant Distributed-Shared Memory Protocol on the SOME-bus Multiprocessor Architecture. In *International Parallel and Distributed Processing Symposium*, 2003.

[HKL+95] H. Högl, A. Kugel, J. Ludvig, R. Männer, K. H. Noffz, and R. Zoz. Enable++: A Second Generation FPGA Processor. In *FCCM '95 : Proceedings of IEEE Symposium FPGAs for Custom Computing Machines*, pages 45–53, 1995.

[HM03a] Jingcao Hu and Radu Marculescu. Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures. In *DATE'03*, pages 688–693, 2003.

[HM03b] Jingcao Hu and Radu Marculesu. Energy-Aware Mapping for Tile-Based NoC Architectures under Performance Constraints. In *ASP-DAC '03*, 2003.

[HM04] Jingcao Hu and Radu Marculescu. Application-Specific Buffer Space Allocation for Networks-On-Chip Router Design. In *ICCAD '04: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pages 354–361, 2004.

[HMH01] R. Ho, K.W. Mai, and M.A. Horowitz. The Future of Wires. *Proc. of IEEE*, 89 (4):490504, 2001.

[HN06] Clint Hilton and Brent Nelson. PNoC: A Flexible Circuit-Switched NoC for FPGA-Based Systems. *IEE Proceedings Computers and Digital Techniques*, 153(3):181– 188, May 2006.

[HP00] Cheng-Ta Hsieh and Massoud Pedram. Architectural Power Optimization by Bus Splitting. In *Design, Automation and Test in Europe*, pages 612–616, March 2000.

[HPRA02] C.J. Hughes, V.S. Pai, P. Ranganathan, and S.V. Adve. RSIM: Simulating Shared-Memory Multiprocessor with ILP Processors. *IEEE Computer*, 35(2), February 2002.

[HTA94] Neil Howard, Andrew Tyrrell, and Nigel Allinson. *FPGA Acceleration of Electronic Design Automation Tasks*. Abingdon EE&CS Books, Oxford, UK, UK, 1994.

[HZC+06] Yuanfang Hu, Yi Zhu, Hongyu Chen, Ronald Graham, and Chung-Kuan Cheng. Communication Latency Aware Low Power NoC Synthesis. In *DAC '06: Proceedings of the 43rd annual conference on Design automation*, pages 574–579, 2006.

[JKK00] Sushil Chandra Jain, Anshul Kumar, and Shashi Kumar. Efficient Embedding of Partitioned Circuits onto Multi-FPGA Boards. In *FPL '00: Proceedings of the The Roadmap to Reconfigurable Computing, 10th International Workshop on Field-Programmable Logic and Applications*, pages 201–210, London, UK, 2000. Springer-Verlag.

[JKK02] Sushil Chandra Jain, Anshul Kumar, and Shashi Kumar. Hybrid Multi-FPGA Board Evaluation by Limiting Multi-Hop Routing. In *RSP '02: Proceedings of the 13th IEEE International Workshop on Rapid System Prototyping (RSP'02)*, 2002.

[JKK03] Sushil Chandra Jain, Anshul Kumar, and Shashi Kumar. Hybrid Multi-FPGA Board Evaluation by Permitting Limited Multi-Hop Routing. In *Design Automation of Embedded Systems*, pages 309–326, London, UK, 2003. Kluwer Academic Publishers.

[JKY05] Y. Jin, E.J. Kim, and K.H. Yum. Peak Power Control for a QoS Capable On-Chip Network. In *ICPP '05: International Conference on Parallel Processing*, pages 585–592, June 2005.

[JMBM04] Antoine Jalabert, Srinivasan Murali, Lica Benini, and Giovanni De Micheli. ×pipesCompiler: A Tool for Instantiating Application Specific Networks on Chip. In *DATE'04*, volume 2, pages 884–889, 2004.

[Joh05] Johannes Grad. First Encounter. http://www.chiptalk.org/modules/wfsection/article.php?articleid=1, Oct 14 2005. Cadence Encounter Tutorial.

[KA96] Yu-Kwong Kwok and Ishfaq Ahmad. Dynamic Critical-Path Scheduling: An Effective Technique for Allocating Task Graphs to Multiprocessors. *IEEE Trans. Parallel Distrib. Syst.*, 7(5):506–521, 1996.

[KCPP00] Bong Wan Kim, Jong Hyuk Choi, Kwang-Il Park, and Kyu Ho Park. A Wormhole Router with Embedded Broadcasting Virtual Bus for Mesh Computers. *Annual Symposium on Fault Tolerant Computing*, 10(1):29–38, 2000.

[KDHDS] C. Katsinis, booktitle="Parallel D. Hecht", title = "Fault-Tolerant Distributed Shared Memory on a Broadcast-Based Architecture", and volume=15(12) pages =1082-1092 month=December year=2004 Distributed Systems, IEEE Transactions on".

[KJS+02] Shashi Kumar, Axel Jantsch, Juha-Pekka Soininen, Martti Forsell, Mikael Millberg, Johny Öberg, Kari Tiensyrjä, and Ahmed Hemani. A Network on Chip Architecture and Design Methodology. In *Annual Symposium on VLSI'2002, IEEE CS Press*, pages 105–112, 2002.

[KMN+00] Kurt Keutzer, Sharad Malik, Richard Newton, Jan Rabaey, and Alberto Sangiovanni-Vincentelli. System-Level Design: Orthogonalization of Concerns and Platform-Based Design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19 (12):1523–1543, December 2000.

[KND02] A.N.F. Karim, A Nguyen, and S. Dey. An Interconnect Architecture for Networking Systems on Chip. *IEEE Micro*, 22(5):36–45, Sept-Oct 2002.

[KNDR01] F. Karim, A. Nguyen, S. Dey, and R.D. Rao. On-chip Communication Architecture for Oc-768 Network Processors. In *DAC*, pages 678–683, 2001.

[KNP06] Jongman Kim, Chrysostomos Nicopoulos, and Dongkook Park. A Gracefully Degrading and Energy-Efficient Modular Router Architecture for On-Chip Networks. In *ISCA '06: Proceedings of the 33rd annual international symposium on Computer Architecture*, pages 4–15, 2006.

[Kon04] Jeong-Taek Kong. CAD for Nanometer Silicon Design Challenges and Success. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 12 (11):1132–1147, Nov. 2004.

[KPKJ07] Amit Kumar, Li-Shiuan Peh, Partha Kunduz, and Niraj Jha. Express Virtual Channels: Towards the Ideal Interconnection Fabric. In *ISCA'07*, 2007.

[KPN+05] H. Jin Kim, David Park, Chrysostomos Nicopoulos, Vijaykrishnan Narayanan, and C. Das. Design and Analysis of an NoC Architecture from Performance, Reliability and Energy Perspective. In *Symposium On Architecture For Networking And Communications Systems*, pages 173 – 182, 2005.

[KR87] V.K.P. Kumar and C.S. Raghavendra. Array Processor with Multiple Broadcasting. *Parallel and Distributed Computing*, 4:173–190, 1987.

[KS03] Nikolay Kavaldjiev and Gerard J.M. Smit. A Survey of Efficient On-Chip Communications for SoC. In *PROGRESS 2003 Embedded Systems Symposium*, October 2003.

[KS04] Nikolay Kavaldjiev and Gerard J.M. Smit. An Energy-Efficient Network-on-Chip for a Heterogeneous Tiled Reconfigurable Systems-on-Chip. In *EUROMICRO Symposium on Digital System Design*, pages 492–498, Sep 2004.

[KYL+03] E.J. Kim, K.H. Yum, G.M. Link, N. Vijaykrishnan, M. Kandemir, M.J. Irwin, M. Yousif, and C.R. Das. Energy Optimization Techniques in Cluster Interconnects. In *ISLPED '03: International Symposium on Low Power Electronics and Design*, pages 459–464, 2003.

[LB06] Seung Eun Lee and Nader Bagherzadeh. Increasing the Throughput of an Adaptive Router in Network-On-Chip (NoC). In *CODES+ISSS '06: Proceedings of the 4th international conference on Hardware/software codesign and system synthesis*, pages 82–87, 2006.

[LK03a] Tang Lei and Shashi Kumar. A Two-step Genetic Algorithm for Mapping Task Graphs to a Network on Chip Architecture. In *Euromicro Symposium on Digital System Design (DSD'03)*, 2003.

[LK03b] Tang Lei and Shashi Kumar. Algorithms and Tools for Network on Chip Based System Design. In *16th Symposium on Integrated Circuits and Systems Design (SBCCI'03)*, 2003.

[LSEV99] D. Lau, A. Schneider, M.D. Ercegovac, and J. Villasenor. FPGA-Based Structures for On-Line FFT and DCT. In *Field-Programmable Custom Computing Machines (FCCM '99)*, pages 310–311, 1999.

[LT06] Lap-Fai Leung and Chi-Ying Tsui. Optimal Link Scheduling on Improving Best-Effort and Guaranteed Services Performance in Network-On-Chip Systems. In *DAC '06: Proceedings of the 43rd annual conference on Design automation*, pages 833–838, 2006.

[LTXW05] Yong Li, Zheng Tang, GuangPu Xia, and RongLong Wang. A Positively Self-feedbacked Hopfield Neural Network Architecture for Crossbar Switching. *Circuits and Systems I, IEEE Transactions on*, 52(1), 2005.

[LZT04] Jian Liu, Li-Rong Zheng, and Hannu Tenhunen. Interconnect Intellectual Property for Network-on-Chip (NoC). *J. Syst. Archit.*, 50(2-3):65–79, 2004.

[MA98] N. McKeown and T.E. Anderson. A Quantitaive Comparison of Scheduling Algorithms for Input-Queued Switches. In *Computer Networks & ISDN Systems, vol. 30, n. 24*, pages 2309–2326, December 1998.

[MABM06] Srinivasan Murali, David Atienz, Luca Benini, and Giovanni De Michel. A Multi-Path Routing Strategy with Guaranteed In-Order Packet Delivery and Fault-Tolerance for Networks on Chip. In *DAC '06: Proceedings of the 43rd annual conference on Design automation*, pages 845–848, 2006.

[MBD+05] Théodore Marescaux, B. Bricke, P. Debacker, Vincent Nollet, and Henk Corporaal. Dynamic Time-Slot Allocation for QoS Enabled Networks on Chip. In *ESTImedia*, pages 47–52, 2005.

[MBV+02] Theodore Marescaux, Andrei Bartic, Diderick Verkest, Serge Vernalde, and Rudy Lauwereins. Interconnection Networks Enable Fine-Grain Dynamic Multi-Tasking on FPGAs. In *FPL'2002*, pages 795–805, September 2002.

[MC93] S. Monaghan and C. P. Cowen. Reconfigurable Multi-Bit Processor for DSP Applications in Statistical Physics. In *IEEE Workshop on FPGAs for Custom Computing Machines*, pages 103–110, 1993.

[McC07] Peter McCrorie. Using Dynamic and Static Power Rail Analysis to Maximize Results with Minimum Effort. http://www.soccentral.com/results.asp?entryID=19453, 2007. Contributor: Cadence Design Systems, Inc.

[MCM+04] Fernando Gehm Moraes, Ney Calazans, Aline Mello, Leandro Möller, and Luciano Ost. HERMES: An Infrastructure for Low Area Overhead Packet-Switching Networks on Chip. *Integration*, 38(1):69–93, 2004.

[MCM+05] César Marcon, Ney Calazans, Fernando Moraes, Altamiro Susin, Igor Reis, and Fabiano Hessel. Exploring NoC Mapping Strategies: An Energy and Timing Aware Technique. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 502–507, Washington, DC, USA, 2005.

[MCR+06a] Srinivasan Murali, Martijn Coenen, Andrei Radulescu, Kees Goossens, and Giovanni De Micheli. A Methodology for Mapping Multiple Use-Cases onto Networks on Chips. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 118–123, 2006.

[MCR+06b] Srinivasan Murali, Martijn Coenen, Andrei Radulescu, Kees Goossens, and Giovanni De Micheli. Mapping and Configuration Methods for Multi-Use-Case Networks on Chips. In *ASP-DAC '06: Proceedings of the 2006 conference on Asia South Pacific design automation*, pages 146–151, 2006.

[MdMM+03] Fernando Gehm Moraes, Aline Vieira de Mello, Leandro Heleno Möller, Luciano Copello Ost, and Ney Laert Vilar Calazans. A Low Area Overhead Packet-switched Network On Chip: Architecture and Prototyping. In *IFIP VLSI-SOC 2003*, pages 318–323, 2003.

[Men07] Mentor Graphics Inc. ModelSim Simulator. http://www.model.com, 2007.

[MKSC05] César A. M. Marcon, Márcio Eduardo Kreutz, Altamiro Amadeu Susin, and Ney Laert Vilar Calazans. Models for Embedded Application Mapping onto NoCs: Timing Analysis. In *IEEE International Workshop on Rapid System Prototyping*, pages 17–23, 2005.

[MMA+06] Srinivasan Murali, Paolo Meloni, Federico Angiolini, David Atienza, Salvatore Carta, Luca Benini, Giovanni De Micheli, and Luigi Raffo. Designing Application-Specific Networks on Chips with Floorplan Information. In *ICCAD '06: Proceedings of the 2006 IEEE/ACM International Conference on Computer-Aided Design*, pages 355–362, 2006.

[MMB+03] Théodore Marescaux, Jean-Yves Mignolet, Andrei Bartic, W. Moffat, Diederik Verkest, Serge Vernalde, and Rudy Lauwereins. Networks on Chip as Hardware Components of an OS for Reconfigurable Systems. In *FPL*, pages 595–605, 2003.

[MMCM04] Aline Mello, Leandro Möller, Ney Calazans, and Fernando Gehm Moraes. MultiNoC: A Multiprocessing System Enabled by a Network on Chip. In *DATE*, pages 234–239, 2004.

[MNM+04] Théodore Marescaux, Vincent Nollet, Jean-Yves Mignolet, Andrei Bartic, W. Moffat, Prabhat Avasare, Paul Coene, Diederik Verkest, Serge Vernalde, and Rudy Lauwereins. Run-time Support for Heterogeneous Multitasking on Reconfigurable SoCs. *Integration*, 38(1):107–130, 2004.

[MNTJ04] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch. Guaranteed Bandwidth Using Looped Containers in Temporally Disjoint Networks within the Nostrum Network on Chip. In *Design, Automation and Test in Europe Conference and Exhibition*, pages 890–895, 2004.

[Moh98] Prasant Mohapatra. Wormhole Routing Techniques for Directly Connected Multicomputer Systems. In *ACM Computing Surveys*, volume 30(3), Sep 1998.

[MT99] S. Matsumae and N. Tokura. Simulating a Mesh with Separable Buses by a Mesh with Partitioned Buses. In *Fourth International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN '99)*, pages 198–203, 1999.

[MVB95] Laurent Moll, Jean Vuillemin, and Philippe Boucard. High-Energy Physics on DECPeRLe-1 Programmable Active Memory. In *FPGA '95: Proceedings of the 1995 ACM third International Symposium on Field-programmable Gate Arrays*, pages 47–52, 1995.

[NMAM05] Vincent Nollet, Théodore Marescaux, Prabhat Avasare, and Jean-Yves Mignolet. Centralized Run-Time Resource Management in a Network-on-Chip Containing Reconfigurable Hardware Tiles. In *DATE*, pages 234–239, 2005.

[NMV+04] Vincent Nollet, Théodore Marescaux, Diederik Verkest, Jean-Yves Mignolet, and Serge Vernalde. Operating-System Controlled Network on Chip. In *DAC*, pages 256–259, 2004.

[NPK+06] Chrysostomos A. Nicopoulos, Dongkook Park, Jongman Kim, N. Vijaykrishnan, Mazin S. Yousif, and Chita R. Das. ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers. In *MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 333–346, 2006.

[ns2] The Network Simulator : `ns-2`. http://www.isi.edu/nsnam/ns/.

[OHM05] Umit Y. Ogras, Jingcao Hu, and Radu Marculescu. Key Research Problems in NoC Design: A Holistic Perspective. In *CODES+ISSS '05: Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 69–74, New York, NY, USA, 2005.

[OM05] U. Y. Ogras and R. Marculescu. Application-Specific Network-on-Chip Architecture Customization Via Long-Range Link Insertion. In *ICCAD '05: Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design*, pages 246–253, 2005.

[OM06] Umit Y. Ogras and Radu Marculescu. Prediction-Based Flow Control for Network-on-Chip Traffic. In *DAC '06: Proceedings of the 43rd annual conference on Design automation*, pages 839–844, 2006.

[OMLC06] Umit Y. Ogras, Radu Marculescu, Hyung Gyu Lee, and Naehyuck Chang. Communication Architecture Optimization: Making the Shortest Path Shorter in Regular Networks-on-Chip. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 712–717, 2006.

[OPN06] OPNET Technologies Inc. OPNET Modeler. In *http://www.opnet.com/products/modeler/opnet_modeler.pdf*, May 2006.

[osu07] Oklahoma State University System on Chip (SoC) Design Flows. http://avatar.ecen.okstate.edu/projects/scells/, 2007.

[Pau04] P.G. Paulin. Automatic mapping of parallel applications onto multi-processor platforms: A multimedia application. In *Euromicro Symposium on Digital System Design (DSD '04)*, 2004.

[Phi02] Philips Semiconductors. Device Transaction Level (DTL) Protocol Specification. Version 2.2, July 2002.

[Phi03] Philips. Nexperia PNX8550 Home Entertainment Engine, Dec 2003.

[Phi04] Philips. Nexperia PNX15xx Series Data Book, December 2004.

[PHKC06] Maurizio Palesi, Rickard Holsmark, Shashi Kumar, and Vincenzo Catania. A Methodology for Design of Application Specific Deadlock-Free Routing Algorithms for NoC Systems. In *CODES+ISSS '06: Proceedings of the 4th international conference on Hardware/software codesign and system synthesis*, pages 142–147, 2006.

[Pro06] Nostrum Research Project. http://www.imit.kth.se/info/FOFU/Nostrum, May 2006.

[PRR+04] Santiago González Pestana, Edwin Rijpkema, Andrei Radulescu, Kees G. W. Goossens, and Om Prakash Gangwal. Cost-Performance Trade-Offs in Networks on Chip: A Simulation-Based Approach. In *DATE*, pages 764–769, 2004.

[Raw06] Raw Architecture Workstation (RAW) research project. http://cag-www.lcs.mit.edu/raw/, May 2006.

[RDG+04] Andrei Radulescu, John Dielissen, Kees G. W. Goossens, Edwin Rijpkema, and Paul Wielage. An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration. In *DATE*, pages 878–883, 2004.

[RGR+03] Edwin Rijpkema, Kees G. W. Goossens, Andrei Radulescu, John Dielissen, Jef L. van Meerbergen, Paul Wielage, and E. Waterlander. Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip. In *DATE*, pages 10350–10355, 2003.

[RJH04] Chae-Eun Rhee, Han-You Jeong, and Soonhoi Ha. Many-To-Many Core-Switch Mapping in 2-D Mesh NoC Architectures. In *ICCD '04: Proceedings of the 2004 International Conference on Computer Design*, pages 438–443, 2004.

[RSG03] V. Raghunathan, M.B. Srivastava, and R.K. Gupta. A Survey of Techniques for Energy Efficient On-Chip Communication. In *Design Automation Conference*, pages 900–905, 2003.

[RSV97] James A. Rowson and Alberto Sangiovanni-Vincentelli. Interface-Based Design. In *34th Design Automation Conference*, 1997.

[SADB95] Charles Selvidge, Anant Agarwal, Matt Dahl, and Jonathan Babb. TIERS: Topology Independent Pipelined Routing and Scheduling for VirtualWire Compilation. In *FPGA '95: Proceedings of the 1995 ACM third International Symposium on Field-programmable Gate Arrays*, pages 25–31, 1995.

[SBKV05] Balasubramanian Sethuraman, Prasun Bhattacharya, Jawad Khan, and Ranga Vemuri. LiPaR: A Light-Weight Parallel Router for FPGA-based Networks-on-Chip. In *15th Great Lakes Symposium on VLSI (GLSVLSI'05)*, Chicago, IL, USA, 2005.

[SC05] Krishnan Srinivasan and Karam S. Chatha. A Technique for Low Energy Mapping and Routing in Network-on-Chip Architectures. In *International Symposium on Low Power Electronics and Design (ISLPED'05)*, pages 387 – 392, 2005.

[SC06a] Krishnan Srinivasan and Karam S. Chatha. A Methodology for Layout Aware Design and Optimization of Custom Network-on-Chip Architectures. In *ISQED*, pages 352–357, 2006.

[SC06b] Krishnan Srinivasan and Karam S. Chatha. Layout aware design of mesh based NoC architectures. In *CODES+ISSS '06: Proceedings of the 4th international conference on Hardware/software codesign and system synthesis*, pages 136–141, 2006.

[SCK06] Krishnan Srinivasan, Karam S. Chatha, and Goran Konjevod. Linear-Programming-Based Techniques for Synthesis of Network-on-Chip Architectures. *IEEE Transactions on Very Large Scale Integration Systems*, 14(4):407–420, 2006.

[SDN+06] Frits Steenhof, Harry Duque, Björn Nilsson, Kees Goossens, and Rafael Peset Llopis. Networks on Chips for High-End Consumer-Electronics TV System Architectures. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 148–153, 2006.

[Sem06] International Sematech. International Technology Roadmap for Semiconductors, 2006 Update. http://public.itrs.net, 2006.

[Sha06] L. Shang. PoPNet Project. In *http://www.princeton.edu/ lshang/popnet.html*, 2006.

[SHSM02] F. Schurmann, S. Hohmann, J. Schemmel, and K. Meier. Towards an Artificial Neural Network Framework. In *NASA/DoD Conference on Evolvable Hardware*, pages 266–273, 2002.

[SK04] Dongkun Shin and Jihong Kim. Power-aware Communication Optimization for Networks-on-Chips with Voltage Scalable Links. In *CODES+ISSS '04: Proceedings of the 2nd IEEE/ACM/IFIP International Conference on Hardware/software Codesign and System Synthesis*, pages 170–175, 2004.

[SKV04] Balasubramanian Sethuraman, Jawad Khan, and Ranga Vemuri. Battery-Efficient Task Execution on Portable Reconfigurable Computing Platforms. In *IEEE International System-on-Chip Conference (IEEE-SOCC 2004)*, pages 237–240, April 2004.

[Sli06] SlicNets. Silicon Networks System-on-Chip Project. In *http://www.ece.cmu.edu/s̃ld/research/soc.php*, May 2006.

[SMCRT05] P. Schumacher, M. Mattavelli, A. Chirila-Rus, and R. Turney. A Software/Hardware Platform For Rapid Prototyping Of Video And Multimedia Designs. In *Fifth International Workshop on System-on-Chip for Real-Time Applications*, pages 30–33, 2005.

[Sod03] M.A Soderstrand. CSD multipliers for FPGA DSP applications. In *International Symposium on Circuits and Systems (ISCAS '03)*, 2003.

[Son06] Sonics Inc. SiliconBackplane^{TM} III. In *http://www.sonicsinc.com/sonics/products/siliconbackplaneIII/*, 2006.

[SPI06] SPIN. Scalable Programmable Integrated Network. In *http://www-asim.lip6.fr/recherche/spin/*, May 2006.

[SSA04] R. Soares, I.S. Silva, and A. Azevedo. When reconfigurable architecture meets network-on-chip. In *17th Symposium on Integrated Circuits and Systems Design (SBCCI'04)*, pages 216–221, 2004.

[SSM+01] Marco Sgroi, Michael Sheets, Andrew Mihal, Kurt Keutzer, Sharad Malik, Jan M. Rabaey, and Alberto L. Sangiovanni-Vincentelli. Addressing the System-on-a-Chip Interconnect Woes Through Communication-Based Design. In *DAC*, pages 667–672, 2001.

[SSS95] Stephen D. Scott, Ashok Samal, and Shared Seth. HGA: A Hardware-Based Genetic Algorithm. In *FPGA '95: Proceedings of the 1995 ACM third International Symposium on Field-programmable Gate Arrays*, pages 53–59, 1995.

[ST95] Herman Schmit and Don Thomas. Implementing Hidden Markov Modelling and Fuzzy Controllers in FPGAs. In *FCCM '95 : Proceedings of IEEE Symposium FPGAs for Custom Computing Machines*, pages 214–221, 1995.

[STB06] STBus Interconnect. STMicroelectronics Inc. In *http://www.st.com/stonline/prodpres/dedicate/soc/cores/stbus.htm*, 2006.

[Sto83] Q. F. Stout. Mesh-Connected Computers with Broadcasting. *IEEE Transactions on Computers*, C-32(9):826830, September 1983.

[Str06] David Stringfellow. Rail-Signoff Analysis Ensures SoC Power Integrity. http://www.elecdesign.com/Articles/ArticleID/11883/11883.html, Jan 19 2006. Article.

[SV06a] Balasubramanian Sethuraman and Ranga Vemuri. Multi^2 Router: A Novel Multi Local Port Router Architecture With Broadcast Facility For FPGA-Based Networks-On-Chip. In *International Conference on Field Programmable Lofic & Applications*, pages 543–546, 2006.

[SV06b] Balasubramanian Sethuraman and Ranga Vemuri. optiMap: A Tool for generating efficient NoC Architectures using Multi-Port Routers for FPGAs. In *Design Automation and Test in Europe (DATE '06)*, Munich, Germany, 2006.

[SV07a] Balasubramanian Sethuraman and Ranga Vemuri. A Force-directed Approach for Fast Generation of Efficient Multi-Port NoC Architectures. In *20th International Conference on VLSI Design + 6th International Conference on Embedded Systems*, pages 419–426, Bangalore, India, 2007.

[SV07b] Balasubramanian Sethuraman and Ranga Vemuri. Multicasting based topology generation and core mapping for a power efficient networks-on-chip. In *ISLPED '07 : IEEE/ACM SIGDA International Symposium on Low Power Electronics and Design*, Portland, OR, USA, August 2007.

[SV07c] Balasubramanian Sethuraman and Ranga Vemuri. Power Variations of MultiPort Routers in an Application-Specifi NoC Design : A Case Study. In *ICCD '07 : XXV IEEE International Conference on Computer Design*, Lake Tahoe, CA, USA, October 2007.

[Sys06] SystemC Language Specification ver 2.0. Open SystemC Initiative. In *http://www.systemc.org/*, May 2006.

[THT+97] Horng-Ren Tsai, Shi-Jinn Horng, Shun-Shan Tsai, Tzong-Wann Kao, and Shung-Shing Lee. Solving An Algebraic Path Problem and Some Related Graph Problems on a Hyper-Bus Broadcast Network. 8(12):1226–1235, December 1997.

[TLV+04] T. Theocharides, G. Link, N. Vijaykrishnan, M.J. Invin, and V. Srikantam. A Generic Reconfigurable Neural Network Architecture as a Network on Chip. In *IEEE International SOC Conference*, pages 191–194, 2004.

[TMG+06] Leonel Tedesco, Aline Mello, Leonardo Giacomet, Ney Calazans, and Fernando Moraes. Application Driven Traffic Modeling for NoCs. In *SBCCI '06: Proceedings of the 19th annual symposium on Integrated circuits and systems design*, pages 62–67, 2006.

[TST03] K. Tatas, D. Soudris Siozios, and A. Thanailakis. Power-Efficient Implementations of Multimedia Applications on Reconfigurable Platforms. In *Field-Programmable Logic and Applications*, volume 2778, pages 1032–1035. Springer-Verlag GmbH, 2003.

[Var06] A. Varga. OMNET++ User Manual Version 2.3. In *http://www.omnetpp.org/*, May 2006.

[Vas04] Vass Soteriou and Li-Shiuan Peh. Design space Exploration of Power-Aware On/Off Interconnection Networks. In *ICCD '04: Proceedings of International Conference on Computer Design*, Oct 2004.

[VBR+] Jean Vuillemin, Patrice Bertin, Didier Roncin, Mark Shand, Hervé Touati, and Philippe Boucard. Programmable Active Memories: Reconfigurable Systems Come of Age. *IEEE Transactions on VLSI Systems*, 4.

[VDC03] Bart Vermeulen, John Dielissen, and Calin Ciordas. Bringing Communication Networks on a Chip: Test and Verification Implications. *Communications Magazine, IEEE*, 41 (9):74–81, September. 2003.

[VM02] Girish Varatkar and Radu Marculescu. Traffic Analysis For On-Chip Networks Design Of Multimedia Applications. In *DAC '02: Proceedings of the 39th conference on Design automation*, pages 795–800, 2002.

[VN06a] Mário Véstias and Horácio Neto. Area and Performance Optimization of a Generic Network-On-Chip Architecture. In *SBCCI '06: Proceedings of the 19th annual symposium on Integrated circuits and systems design*, pages 68–73, 2006.

[VN06b] Mário Véstias and Horácio Neto. Co-Synthesis of a Configurable SoC Platform Based on a Network on Chip Architecture. In *ASP-DAC '06: Proceedings of the 2006 conference on Asia South Pacific design automation*, pages 48–53, 2006.

[VVP+02] M Vaccharajini, N. Vachharajani, D.A. Penry, J.A. Blome, and D.I. August. Micro-Architectural Exploration with Liberty. In *Proceedings of International Symposium on Microarchitecture*, pages 271–282, November 2002.

[Whe06] D. Whelihan. The CMU NOCSim Simulator. In *http://www.ece.cmu.edu/ djw2/NOCSim/*, 2006.

[Wik06] Wikipedia. System-on-Chip. In *http://en.wikipedia.org/wiki/System-on-a-chip*, 2006.

[wis06] Wishbone system-on-chip (soc) interconnect architecture. In *http://www.opencores.org/projects.cgi/web/wishbone/wishbone*, 2006.

[WKL+04] Andreas Wieferink, Tim Kogel, Rainer Leupers, Gerd Ascheid, Heinrich Meyr, Gunnar Braun, and Achim Nohl. A System Level Processor/Communication Co-Exploration Methodology for Multi-Processor System-on-Chip Platforms. In *Design, Automation and Test in Europe Conference and Exhibition*, volume 2, pages 1256–1261, 2004.

[WPM05] Hangsheng Wang, Li-Shiuan Peh, and Sharad Malik. A Technology-Aware and Energy-Oriented Topology Exploration for On-Chip Networks. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 1238–1243, Washington, DC, USA, 2005.

[WZPM02] H. Wang, X. Zhu, L. Peh, and S. Malik. ORION: A Power-Performance Simulator for Interconnection Networks. In *Proceedings of International Symposium on Microarchitecture*, pages 294–305, November 2002.

[Xil05] Xilinx Inc. logiCORE. In *http://www.xilinx.com/ipcenter/*, 2005.

[Xil06a] Xilinx Inc. http://www.xilinx.com, 2006.

[Xil06b] Xilinx Inc. Virtex IV Platform FPGA User Guide. In *http://direct.xilinx.com/bvdocs/userguides/ug070.pdf*, 2006.

[Xil07a] Xilinx Inc. Virtex-5 Data Sheet: DC and Switching Characteristics. In *http://direct.xilinx.com/bvdocs/userguides/ug100.pdf*, 2007.

[Xil07b] Xilinx Inc. Virtex V Platform FPGA User Guide. In *http://direct.xilinx.com/bvdocs/userguides/ug190.pdf*, 2007.

[XWHC06] Jiang Xu, Wayne Wolf, Joerg Henkel, and Srimat Chakradhar. A Design Methodology for Application-Specific Networks-on-Chip. *Trans. on Embedded Computing Sys.*, 5(2):263–280, 2006.

[YBK99] Joon-Seo Yim, Seong-Ok Bae, and Chong-Min Kyung. A Floorplan-Based Planning Methodology for Power and Clock Distribution in ASICs. In *DAC '99: Proceedings of the 36th ACM/IEEE conference on Design automation*, pages 766–771, 1999.

[YMB02] Terry Tao Ye, Giovanni De Micheli, and Luca Benini. Analysis of Power Consumption on Switch Fabrics in Network Routers. In *DAC '02: Proceedings of the 39th conference on Design automation*, pages 524–529, 2002.

[YMB04] Terry Tao Ye, Giovanni De Micheli, and Luca Benini. Packetization and Routing Analysis of On-Chip Multiprocessor Networks. *JSA*, 50(2-3):81–104, Feb 2004.

[YN01] M. Yamada and A. Nishihara. High-Speed FIR Digital Filter with CSD Coefficients Implemented on FPGA. In *Asia and South Pacific Design Automation Conference*, 2001.

[You02] Simon Young. Neglecting IR Drop In Nanometer Designs Leads To Silicon Failure. http://www.elecdesign.com/Articles/ArticleID/2594/2594.html, Aug 5 2002. Article.

[YP03] Fan Yang and M. Paindavoine. Implementation of an RBF Neural Network on Embedded Systems: Real-Time Face Tracking and Identity Verification. *Neural Networks, IEEE Transactions on*, 14(5):1162–1175, September 2003.

[ZHM07] Nicholas H. Zamora, Xiaoping Hu, and Radu Marculescu. System-Level Performance/Power Analysis for Platform-Based Design of Multimedia Applications. *ACM Transactions Design Automation of Electronic Systems*, 12(1):2, 2007.

[Zhu05] Xinping Zhu. *Sofware Tools for Modeling and Simulation of On-Chip Communication Architectures*. PhD thesis, Princeton University, Princeton, New Jersey, United States, June 2005.

[ZKS04] C.A. Zerferino, M.E. Kreutz, and A.A. Susin. RASoC: A Router Soft-Core for Networks-on-Chip. In *DATE'2004-Designer's Forum*. IEEE CS Press, 2004.

[ZM04] Xinping Zhu and Sharad Malik. Using A Communication Architecture Specification in an Application-Driven Retargetable Prototyping Platform for Multiprocessing. In *Proceedings of Design Automation and Test in Europe (DATE 04)*, volume 2, pages 1244–1249, February 2004.

[ZS03] C.A. Zerferino and A.A. Susin. SoCIN: A Parametric and Scalable Network on Chip. In *SBCCI'2003*, pages 169–174. IEEE CS Press, 2003.

[ZSS04] C.A. Zerferino, F.G.M.E. Santo, and A.A. Susin. ParIS: A Parametric and Scalable Network on Chip. In *SBCCI'2004*, 2004.

Appendix A

Demonstration of the Xilinx & Synopsys-Cadence Flow

In this appendix, we present a detailed discussion of the flows mentioned in Chapter 10.3. In both the flows, the discussion presented in this chapter is based on the reference design representing a five port Multi^2 Router (in both the flows). Note that this appendix is specific to Section 10.3 and changes may be necessary (Xilinx flow in particular), when operating with the designs from the other Chapters 3, 4, 5, 6, 8, & 9.

A.1 Xilinx Flow

Initially, a project is created in the Xilinx Integrated Synthesis Environment (ISE), populated with all the VHDL files in the design hierarchy. In the project properties window, the target device is chosen as Xilinx Virtex II Pro FPGA (XC2VP30, ff896package). At this point, the black box references to the synchronous FIFO (using BRAM/DRAMs) must be resolved by generating FIFO definitions using FIFO Generator v2.3 (year:2006), which is one of the many core generation utility in the logiCORE tool of Xilinx ISE. Any constraints for synthesis including the placement and the pin constraints have to be specified in the User Constraint File (.ucf), that is available as a design utility in the Process menu/window. Next, the design is compiled and checked for errors, followed a behavioral simulation (before mapping and Place-And-Route steps). In this work, we make use of the Synplicity Synplify Pro for RTL synthesis in place of the standard synthesis engine supported by the Xilinx ISE. Next, we invoke the scripts for Mapping and Placement from the process window. It

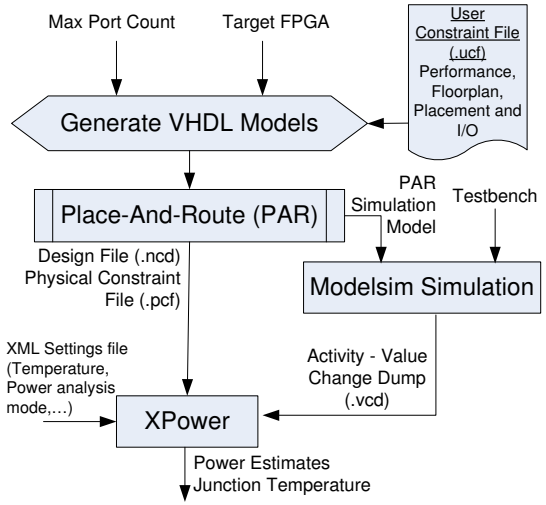


Figure A.1: Xilinx FPGA flow

is necessary to set the option to generate the Post PAR simulation model, before the routing script is initiated. This is the technology specific PAR model which can be simulate using the Modelsim simulator using a testbench and the activity information of various nodes and wires can be captured as a Value Change Dump (VCD) file.

As explained in Chapter 10.3, randomly-generated but a fixed set of data is switched from each of the nine source ports(4 directional + 5 logic ports) available. In other words, except for the header flit that identifies the destination port, the payload of the packets switched from different source ports to the rest of the destination ports remains same. Note, this step covers all possible combinations in terms of the destination port(s) choice, thereby, exploiting the multi cast capabilities of the router. In order to achieve this, when generating a (fixed) set of vectors, the perl script is hardcoded with values (in the port_str variable), representing the header flit that is necessary to effect a particular transfer to one or more destination ports (covering all combinations in a multi cast type of transfer) as shown below.

```
#!/usr/local/bin/perl
#assuming the current router coordinate is 2,2

@vec = (5, 10, 15, 20, 25, 50, 100, 250, 500, 1000);
%port_str = (                                                    5
'n'    => "1111111111111011",
's'    => "1111111111111001",
'e'    => "1111111111111110",
'w'    => "1111111111110110",
'l-0'  => "1111111111101010",                                    10
```

```
'1-1' => "1111111111011010",
'1-2' => "1111111110111010",
'1-3' => "1111111101111010",
'1-4' => "1111111011111010",
'1-01' => "1111111111001010",
'1-02' => "1111111110101010",
'1-03' => "1111111101101010",
'1-04' => "1111111011101010",
'1-12' => "1111111110011010",
'1-13' => "1111111101011010",
'1-14' => "1111111011011010",
'1-23' => "1111111100111010",
'1-24' => "1111111010111010",
'1-34' => "1111111001111010",
'1-012' => "1111111110001010",
'1-013' => "1111111101001010",
'1-014' => "1111111011001010",
'1-023' => "1111111100101010",
'1-024' => "1111111010101010",
'1-034' => "1111111001101010",
'1-123' => "1111111100011010",
'1-124' => "1111111010011010",
'1-134' => "1111111001011010",
'1-234' => "1111111000111010",
'1-0123' => "1111111100001010",
'1-0124' => "1111111010001010",
'1-0134' => "1111111001001010",
'1-0234' => "1111111000101010",
'1-1234' => "1111111000011010"
);
foreach $co (keys (%port_str))
{
   print $co," => ", $port_str{$co}, "\n";
}
srand;
foreach $c2 (keys (%port_str))
  $file_name_str = "p1-$c2";
  foreach $c3 (@vec)
  {
   $t_name ="$file_name_str-$c3.vec";
   open DATA, ">$t_name" or die "Cannot open file: $t_name";
   for( $c4=0; $c4<$c3; $c4++)
   {
     $rand_str ='';
     print DATA "$port_str{$c2}\n";
```

```
    for($j=0; $j<4; $j++)
    {
      $rand_str ='';
      for($i=0; $i<16; $i++)                                    60
      {
        $randno = int(rand(2));
        $rand_str .= "$randno";
      } #i loop
      print DATA "$rand_str\n";                                 65
    }
   } #c4
   close (DATA);
  } #c3
} #c2                                                           70
```

Using the different vector sets generated using the above perl script, the power consumption is estimated for various cases and the iterative process involved in the average power analysis of various cases is explained as follows:

After the Place-And-Route phase of synthesis, we simulate the Placed-And-Routed (PAR) router simulation model using ModelSim 6.3i [Men07] and generate the Value Change Dump (VCD) file. During the simulation, the switching activity of all nets and logic in the PAR design is stored at every clock cycle in the VCD file. Next, XPower tool of the Xilinx ISE 6.3i [Xil06a] is used to obtain the power estimate values of the design, for the vectors that were provided as input for the current simulation run. XPower takes in the PAR design file (.ncd), the physical constraint file (.pcf), the user settings file (.xml) and the VCD file (containing the activity data), and provides an estimate of various power parameters. For experimentation purposes, the temperatures including the ambient and junction temperatures are set at 25 degree Celsius during all simulation runs.

The above iterative process is automated using a perl script (given below). At every iteration, the script chooses each of the different vector sets that was generated earlier and simulates the Post Placed-And-Routed simulation model that was synthesized earlier.

```
#!/usr/local/bin/perl
#assuming the current router coordinate is 2,2

@vec = (5, 10, 15, 20, 25, 50, 100, 250, 500, 1000);
@vec1 = (5, 10, 15, 20, 25, 50, 100);                           5

%src_str = (
```

```
'n'  => "file N_IN: TEXT open read_mode is \"../vec/",
'e'  => "file E_IN: TEXT open read_mode is \"../vec/",
'w'  => "file W_IN: TEXT open read_mode is \"../vec/",
's'  => "file S_IN: TEXT open read_mode is \"../vec/",
'l0'  => "file L0_IN: TEXT open read_mode is \"../vec/",
'l1'  => "file L1_IN: TEXT open read_mode is \"../vec/",
'l2'  => "file L2_IN: TEXT open read_mode is \"../vec/",
'l3'  => "file L3_IN: TEXT open read_mode is \"../vec/",
'l4'  => "file L4_IN: TEXT open read_mode is \"../vec/"
);

$ttt="./tbtemplate.vhd";

open TBTEMPLATE, "$ttt" or die "Cannot open file : tbtemplate.vhd";
open DOTEMPLATE, "br5do-template.fdo" or
  die "Cannot open file : br5do-template.fdo";
@dodata = <DOTEMPLATE>;
@tbdata = <TBTEMPLATE>;
close TBTEMPLATE;
close DOTEMPLATE;
$cnt =0;
$linecnt=@tbdata;
open RPFILE, ">>RandomPowerResults.txt" or
  die "Cannot open file: RandomPowerResults.txt";
foreach $c1 (@vec)
{
  foreach $c2 (keys (%src_str))
  {
        open TBFILE,">LWRL4_testb.vhd" or
          die "Cannot open file: LWRL4_testb.vhd";
        $cnt = 0;
        while ($cnt <= $linecnt)
        {
          if    ($tbdata[$cnt] !~ $src_str{$c2})
          {
            print TBFILE $tbdata[$cnt];
          }
          else
          {
            $prnt_str    = "$src_str{$c2}"."r-$c2-$c1.vec\"\;\n";
            $pwrrpt_str = "$c1 $c2 R ";
            print TBFILE "$prnt_str";
          }
          $cnt++;
        }#end of while cnt <= linecnt
         close TBFILE;
```

```
# Call the Modelsim and Xpwr
open DOFILE, ">br5do.fdo" or die "Cannot open file: br5do.fdo";
print DOFILE "set i $c1;\n";
$cnt2 = 1;
$linecnt2 = @dodata;

while($cnt2 <= $linecnt2)
{
        print DOFILE "$dodata[$cnt2]";
        $cnt2++;
}#end of while cnt2 <= linecnt2
close DOFILE;

#print "Calling System(\"sh x.bat\")\n";
system("sh simulate.sh");

# Read in the power report and print data into the PowerResults file
open PWRFILE, "lwrouter_pwr_rpt.pwr" or
 die "Cannot open file: lwrouter_pwr_rpt.pwr";
push(@select_str, $pwrrpt_str);
while(<PWRFILE>)
{
       @comp_str=split;
       if ($. == 19)
       {
         push(@select_str, $comp_str[4]);
         push(@select_str, ($comp_str[4]/$c1));
       }
       elsif ($. == 21)
       {
         push(@select_str, $comp_str[3]);
       }
       elsif ($. == 39)
       {
         push(@select_str, $comp_str[5]);
       }
       elsif ($. == 40)
       {
         push(@select_str, $comp_str[2]);
       }
       elsif ($. == 41)
       {
         push(@select_str, $comp_str[2]);
       }
       elsif ($. == 42)
```

```
                {
                  push(@select_str, $comp_str[2]);
                }
                elsif ($. == 46)
                {
                  push(@select_str, $comp_str[3]);
                }
                elsif ($. == 47)
                {
                  push(@select_str, $comp_str[2]);
                }
                elsif ($. == 48)
                {
                  push(@select_str, $comp_str[2]);
                }
                elsif ($. == 49)
                {
                  push(@select_str, $comp_str[2]);
                }
                elsif ($. == 51)
                {
                  push(@select_str, $comp_str[2]);
                }
                elsif ($. == 52)
                {
                  push(@select_str, $comp_str[2]);
                }
                elsif ($. == 53) { last;}
            }#end of while <pwrfile>

            $pwr_prnt_str = join(" ",@select_str);
            $#select_str=-1;#reset the array
            #Total Pwr, power/packet, vccint 1.50v, package power limit  25C, 250LFM, 500LFM, 750LFM,
            # Estimated Junction Temp, 250LFM, 500LFM, 750LFM, Case Temp, Theta J-A
            print RPFILE "$pwr_prnt_str\n";
            $pwr_prnt_str="";
            close PWRFILE;
    }
}
close RPFILE;
```

Broadly, the above perl script carries out three major steps during every iteration as follows:

(1) Testbench Generation: The template of the testbench in stored in the `tbtemplate.vhd` file. Depending on the source port and the type of transfer (one of the many combinations

possible in a multi cast) in the current iteration, the vector set of the different ports are modified. If a particular port is not switching any packets, a black vector file is attached to that port.

(2) Design Simulation & Power Estimation: Next in line is the process of simulation and power estimation. Before all, the actual simulation time for the current simulation must be fixed, which in turn is dictated by the number of packets that are switched in the current iteration. Hence, the simulation time is modified inside the TCL file (br5do.fdo), which will be used when invoking the Modelsim simulator. The template of the br5do.fdo TCL file is given below,

```
set i 1000;
set a 400;
set b [expr $a + ($i*400)];
set c [expr $b+50];
puts stdout "c value is $c";
vlib work
vcom -93 -explicit  LWRL4_testb.vhd
vsim -quiet +no_tchk_msg -c -lib work -t 1ps
  -sdfmax "/UUT=netgen/par/LWRouter_timesim.sdf" lwrouter_lwrl4_test_vhd_tb
vcd file lwrouter.vcd
vcd add -r /lwrouter_LWRL4_test_vhd_tb/uut/*
#report simulator control
run ${c}ns
vcd off lwrouter.vcd
echo "The time now is $now ps."
#report simulator state
quit -f
```

Here, the variable `$i` is set a value equal to the number of packets switched in the current iteration. Except for this modification, the rest of the TCL code remains same across all the iterations. Note that the values represented by the variables $a and $c represent, respectively, the fixed time overhead present during the start and end of the simulation. At the end of simulation, the Value Change Dump (VCD) file having the activity information is created, which is fed to the XPower tool of Xilinx ISE, along with the PAR design file. At the end, the power estimates from XPower tool are available in the file `lwrouter_pwr_rpt.pwr`.

The above two steps involving PAR design simulation followed by power estimation are performed using the shell script file (`simulate.sh` in the perl code presented earlier) as follows,

```
/opt/CAD/MentorGraphics/Modelsim/bin/vsim -c -do br5do.fdo
xpwr LWRouter.ncd LWRlwrouter.pcf -v -a -s lwrouter.vcd -o lwrouter_pwr_rpt.pwr
```

(3) Storing the Essential Power Values: Next, the power estimate values present in the `lwrouter_pwr_rpt.pwr` file are parsed and the required values are appended to the `PowerResults.dat` file, as shown in the perl script.

At the end of all iterations (controlled by the perl script), the power estimates for different cases are available in the `PowerResults.dat` file, which is used for the analysis present in Section 10.4.

A.2 Synopsys-Cadence Flow

A vector set is used to observe the power variations between various ports in the Xilinx FPGA based flow. Though the power estimation tool (XPower) of the Xilinx ISE synthesis platform is able to report the average and peak powers, it is not comprehensive in terms of the temperature and IR drop analysis. Many of the device level details are abstracted away and the user has to remain contended with the summary reports and files generated by Xilinx ISE. Due to limited leeway available for an extensive IR drop analysis, the router designs are ported into an ASIC flow, making use of the Synopsys and Cadence synthesis tool set. In the process of porting, the only required change is to replace the Xilinx BRAM based FIFO (the buffer elements that store the packets in an NoC) references and association with a user defined FIFO. This is because the Synchronous FIFO implementation using BRAM is only available as a black-box implementation using Xilinx LogiCORE tool [Xil05] and the corresponding reference is replaced with a new RTL implementation of the FIFO. Figure 10.2 shows the complete ASIC design flow for layout synthesis.

A.2.1 Logic Synthesis : Synopsys Design Compiler

The router designs available as VHDL source files are input to the industry-standard logic synthesis tool from Synopsys (Design Compiler). TSMC 0.18μ library available from OSU (formerly from IIT) [osu07] is used for technology mapping. Broadly, the steps involved are listed as follows:

Library Path Specification: Paths for the *Target Library* (having the technology logic gates to be used during logic synthesis), *Link Library* (similar to target library, but,

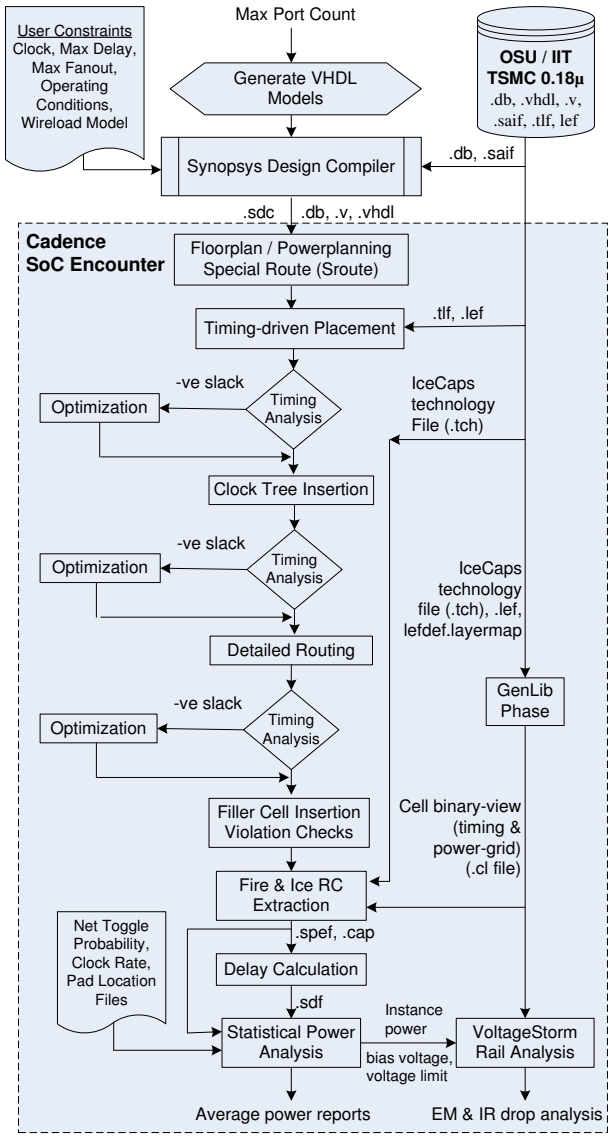


Figure A.2: Synopsys-Cadence Flow

used only for reference to obtain the information about the logic cells in the synthesis technology library and not for mapping) and *Symbol Library* (optional library required for symbol (visual) specification - needed when used with Design Analyzer). In this work, the library paths correspond to the TSMC 0.18μ library available from OSU (formerly from IIT) [osu07] and the *DesignWare* components of generic *GTECH* library available from Synopsys.

Analyze, Elaboration, Link & Uniquify Phases: Each of the VHDL design files must be analyzed and elaborated. *Analyze* command checks the design for syntax errors and performs RTL translation before building the generic logic for the design (GTECH

components). Also, the *analyze* command stores the result of translation as an intermediate design library, which can be used later without the need to be analyzed again. During *elaboration*, the generic parameters (if any) are passed down the hierarchy and multiple references are created, in terms of the generic technology independent DesignWare components from GTECH library. Alternately, the *read* command performs the combined operations of analyze and elaborate commands, but, does not store the analyzed results. Also, parameterized designs (generic valued) statements need to be elaborated, which is not possible using the read command.

Next, the intermediate representation is *linked* and *uniquified* (resolve multiple instances of a block and create unique definition and representation of each instance of a block, allowing individual and independent optimization). At the end of this step, a RTL net list in terms of technology independent generic components (from GTECH library) is available.

Constraints & Operating Conditions A clock must to be created with the target operating frequency. This will provide the constraints required for the maximum register-to-register delay, based on which there can be a positive or a negative slack. Global nets such as clock and reset must be attributed as *don't touch*, to prevent the Design Compiler from optimizing them and the resultant complications. The timing constraints including the skew, rise/fall times and slew rate, fan out constraints, false path specifications must be included at this step. Fault coverage, if necessary, can be specified, subject to various parameters that need must be critical (time/area/power).

In the Static Timing Analysis, the slack calculation is based on the .LIB file from the target library. The Synopsys .LIB technology library has the Wire-load models , operating conditions along with scaling k-factors for different delay components (to estimate the delay number based on the effects of temperature, process, and voltage), and different delay models (eg., piece-wise linear, non-linear, cmos2, etc.,). Note that the actual delay estimation is based on the set of parameters including the pin (name & direction), functional description (combinational/ sequential), pin capacitance & drive capabilities, pin-to-pin timing and area. Most importantly, the delay is greatly affected by the input slew and the output load. In addition, operating conditions including the supply voltage and temperature are specified.

Optimization Constraints & Logic Synthesis: With regard to the optimization constraints, we can turn on the other optimization options like *flattening* of the design (by default false) and structuring of the design (by default true). *Flattening* is the process of

converting the multi-level logic functions into a two-level boolean equations, removing all intermediate variable and parenthesis. This optimization is recommended for unstructured design with random logic and results in a fewer and more balanced logic levels, and may help in the avoidance of false paths. *Structuring* involves modification of the boolean expressions by factoring and representing the factored expressions using intermediate variables. Structuring can be done for either timing (default) or boolean optimization and may sometimes result in increased area. Next, the design is compiled at specified effort levels, with varied optimization metric (time/area/power).

In addition, the most useful optimization is the process of *ungrouping*, wherein the hierarchy of the design is smashed. This allows the possibility of cross boundary logic optimization and may result in better synthesized logic net list compared to the synthesis that is done in isolation with respect to blocks in the hierarchy. It should be noted that removal hierarchy may result in abominable increase in runtime, due the need to optimize the entire design for the constraints specified.

Netlist Save & Report Generation: After the slack is met and the design is optimized, the design can be saved in multiple format including the Synopsys proprietary .DB format, VHDL or verilog netlist. The timing constraints of various ports are available as .SDC file, which along with the synthesized technology-mapped netlist form the input for the next phase involving physical synthesis. In addition, different reports are available in varying levels of detail.

Note that there exists incompatibility with regard to the SDC file format produced by Synopsys Design Compiler and the input for Cadence SoC Encounter. The differences exist mainly in the syntax for the ports having a width greater than one (bus-like). Following is the modified SDC file (five port Multi^2 router) that is passed into Cadence SoC Encounter along with the technology-mapped gate-level net list.

```
##################################################

#  Created by Design Compiler write_sdc on Thu Dec 21 09:23:12 2006

##################################################                    5
set sdc_version 1.4

create_clock -name "clock" -period 15 -waveform {0 7.5} [get_ports {CLK}]
set_clock_uncertainty       0.15 [get_ports {CLK}]
```

```
set_max_fanout 10 [current_design]
set_wire_load_mode "top"
set_max_fanout 10 [get_ports {CLK}]
set_max_fanout 10 [get_ports {RST}]
set_max_fanout 10 [get_ports {Rin_L0(15)}]
set_max_fanout 10 [get_ports {Rin_L0(14)}]
set_max_fanout 10 [get_ports {Rin_L0(13)}]
set_max_fanout 10 [get_ports {Rin_L0(12)}]
set_max_fanout 10 [get_ports {Rin_L0(11)}]
set_max_fanout 10 [get_ports {Rin_L0(10)}]
set_max_fanout 10 [get_ports {Rin_L0(9)}]
set_max_fanout 10 [get_ports {Rin_L0(8)}]
set_max_fanout 10 [get_ports {Rin_L0(7)}]
set_max_fanout 10 [get_ports {Rin_L0(6)}]
set_max_fanout 10 [get_ports {Rin_L0(5)}]
set_max_fanout 10 [get_ports {Rin_L0(4)}]
set_max_fanout 10 [get_ports {Rin_L0(3)}]
set_max_fanout 10 [get_ports {Rin_L0(2)}]
set_max_fanout 10 [get_ports {Rin_L0(1)}]
set_max_fanout 10 [get_ports {Rin_L0(0)}]
set_max_fanout 10 [get_ports {Rin_L4(15)}]
set_max_fanout 10 [get_ports {Rin_L4(14)}]
set_max_fanout 10 [get_ports {Rin_L4(13)}]
set_max_fanout 10 [get_ports {Rin_L4(12)}]
set_max_fanout 10 [get_ports {Rin_L4(11)}]
set_max_fanout 10 [get_ports {Rin_L4(10)}]
set_max_fanout 10 [get_ports {Rin_L4(9)}]
set_max_fanout 10 [get_ports {Rin_L4(8)}]
set_max_fanout 10 [get_ports {Rin_L4(7)}]
set_max_fanout 10 [get_ports {Rin_L4(6)}]
set_max_fanout 10 [get_ports {Rin_L4(5)}]
set_max_fanout 10 [get_ports {Rin_L4(4)}]
set_max_fanout 10 [get_ports {Rin_L4(3)}]
set_max_fanout 10 [get_ports {Rin_L4(2)}]
set_max_fanout 10 [get_ports {Rin_L4(1)}]
set_max_fanout 10 [get_ports {Rin_L4(0)}]
set_max_fanout 10 [get_ports {Rin_N(15)}]
set_max_fanout 10 [get_ports {Rin_N(14)}]
set_max_fanout 10 [get_ports {Rin_N(13)}]
set_max_fanout 10 [get_ports {Rin_N(12)}]
set_max_fanout 10 [get_ports {Rin_N(11)}]
set_max_fanout 10 [get_ports {Rin_N(10)}]
set_max_fanout 10 [get_ports {Rin_N(9)}]
set_max_fanout 10 [get_ports {Rin_N(8)}]
set_max_fanout 10 [get_ports {Rin_N(7)}]
set_max_fanout 10 [get_ports {Rin_N(6)}]
```

```
set_max_fanout 10 [get_ports {Rin_N(5)}]
set_max_fanout 10 [get_ports {Rin_N(4)}]
set_max_fanout 10 [get_ports {Rin_N(3)}]
set_max_fanout 10 [get_ports {Rin_N(2)}]
set_max_fanout 10 [get_ports {Rin_N(1)}]
set_max_fanout 10 [get_ports {Rin_N(0)}]
set_max_fanout 10 [get_ports {Rin_L1(15)}]
set_max_fanout 10 [get_ports {Rin_L1(14)}]
set_max_fanout 10 [get_ports {Rin_L1(13)}]
set_max_fanout 10 [get_ports {Rin_L1(12)}]
set_max_fanout 10 [get_ports {Rin_L1(11)}]
set_max_fanout 10 [get_ports {Rin_L1(10)}]
set_max_fanout 10 [get_ports {Rin_L1(9)}]
set_max_fanout 10 [get_ports {Rin_L1(8)}]
set_max_fanout 10 [get_ports {Rin_L1(7)}]
set_max_fanout 10 [get_ports {Rin_L1(6)}]
set_max_fanout 10 [get_ports {Rin_L1(5)}]
set_max_fanout 10 [get_ports {Rin_L1(4)}]
set_max_fanout 10 [get_ports {Rin_L1(3)}]
set_max_fanout 10 [get_ports {Rin_L1(2)}]
set_max_fanout 10 [get_ports {Rin_L1(1)}]
set_max_fanout 10 [get_ports {Rin_L1(0)}]
set_max_fanout 10 [get_ports {Rin_E(15)}]
set_max_fanout 10 [get_ports {Rin_E(14)}]
set_max_fanout 10 [get_ports {Rin_E(13)}]
set_max_fanout 10 [get_ports {Rin_E(12)}]
set_max_fanout 10 [get_ports {Rin_E(11)}]
set_max_fanout 10 [get_ports {Rin_E(10)}]
set_max_fanout 10 [get_ports {Rin_E(9)}]
set_max_fanout 10 [get_ports {Rin_E(8)}]
set_max_fanout 10 [get_ports {Rin_E(7)}]
set_max_fanout 10 [get_ports {Rin_E(6)}]
set_max_fanout 10 [get_ports {Rin_E(5)}]
set_max_fanout 10 [get_ports {Rin_E(4)}]
set_max_fanout 10 [get_ports {Rin_E(3)}]
set_max_fanout 10 [get_ports {Rin_E(2)}]
set_max_fanout 10 [get_ports {Rin_E(1)}]
set_max_fanout 10 [get_ports {Rin_E(0)}]
set_max_fanout 10 [get_ports {Rin_L2(15)}]
set_max_fanout 10 [get_ports {Rin_L2(14)}]
set_max_fanout 10 [get_ports {Rin_L2(13)}]
set_max_fanout 10 [get_ports {Rin_L2(12)}]
set_max_fanout 10 [get_ports {Rin_L2(11)}]
set_max_fanout 10 [get_ports {Rin_L2(10)}]
set_max_fanout 10 [get_ports {Rin_L2(9)}]
set_max_fanout 10 [get_ports {Rin_L2(8)}]
```

```
set_max_fanout 10 [get_ports {Rin_L2(7)}]
set_max_fanout 10 [get_ports {Rin_L2(6)}]
set_max_fanout 10 [get_ports {Rin_L2(5)}]
set_max_fanout 10 [get_ports {Rin_L2(4)}]
set_max_fanout 10 [get_ports {Rin_L2(3)}]
set_max_fanout 10 [get_ports {Rin_L2(2)}]
set_max_fanout 10 [get_ports {Rin_L2(1)}]
set_max_fanout 10 [get_ports {Rin_L2(0)}]
set_max_fanout 10 [get_ports {Rin_S(15)}]
set_max_fanout 10 [get_ports {Rin_S(14)}]
set_max_fanout 10 [get_ports {Rin_S(13)}]
set_max_fanout 10 [get_ports {Rin_S(12)}]
set_max_fanout 10 [get_ports {Rin_S(11)}]
set_max_fanout 10 [get_ports {Rin_S(10)}]
set_max_fanout 10 [get_ports {Rin_S(9)}]
set_max_fanout 10 [get_ports {Rin_S(8)}]
set_max_fanout 10 [get_ports {Rin_S(7)}]
set_max_fanout 10 [get_ports {Rin_S(6)}]
set_max_fanout 10 [get_ports {Rin_S(5)}]
set_max_fanout 10 [get_ports {Rin_S(4)}]
set_max_fanout 10 [get_ports {Rin_S(3)}]
set_max_fanout 10 [get_ports {Rin_S(2)}]
set_max_fanout 10 [get_ports {Rin_S(1)}]
set_max_fanout 10 [get_ports {Rin_S(0)}]
set_max_fanout 10 [get_ports {Rin_L3(15)}]
set_max_fanout 10 [get_ports {Rin_L3(14)}]
set_max_fanout 10 [get_ports {Rin_L3(13)}]
set_max_fanout 10 [get_ports {Rin_L3(12)}]
set_max_fanout 10 [get_ports {Rin_L3(11)}]
set_max_fanout 10 [get_ports {Rin_L3(10)}]
set_max_fanout 10 [get_ports {Rin_L3(9)}]
set_max_fanout 10 [get_ports {Rin_L3(8)}]
set_max_fanout 10 [get_ports {Rin_L3(7)}]
set_max_fanout 10 [get_ports {Rin_L3(6)}]
set_max_fanout 10 [get_ports {Rin_L3(5)}]
set_max_fanout 10 [get_ports {Rin_L3(4)}]
set_max_fanout 10 [get_ports {Rin_L3(3)}]
set_max_fanout 10 [get_ports {Rin_L3(2)}]
set_max_fanout 10 [get_ports {Rin_L3(1)}]
set_max_fanout 10 [get_ports {Rin_L3(0)}]
set_max_fanout 10 [get_ports {Rin_W(15)}]
set_max_fanout 10 [get_ports {Rin_W(14)}]
set_max_fanout 10 [get_ports {Rin_W(13)}]
set_max_fanout 10 [get_ports {Rin_W(12)}]
set_max_fanout 10 [get_ports {Rin_W(11)}]
set_max_fanout 10 [get_ports {Rin_W(10)}]
```

```
set_max_fanout 10 [get_ports {Rin_W(9)}]
set_max_fanout 10 [get_ports {Rin_W(8)}]
set_max_fanout 10 [get_ports {Rin_W(7)}]
set_max_fanout 10 [get_ports {Rin_W(6)}]
set_max_fanout 10 [get_ports {Rin_W(5)}]
set_max_fanout 10 [get_ports {Rin_W(4)}]
set_max_fanout 10 [get_ports {Rin_W(3)}]
set_max_fanout 10 [get_ports {Rin_W(2)}]
set_max_fanout 10 [get_ports {Rin_W(1)}]
set_max_fanout 10 [get_ports {Rin_W(0)}]
set_max_fanout 10 [get_ports {Req_i_L0}]
set_max_fanout 10 [get_ports {Req_i_L4}]
set_max_fanout 10 [get_ports {Req_i_N}]
set_max_fanout 10 [get_ports {Req_i_L1}]
set_max_fanout 10 [get_ports {Req_i_E}]
set_max_fanout 10 [get_ports {Req_i_L2}]
set_max_fanout 10 [get_ports {Req_i_S}]
set_max_fanout 10 [get_ports {Req_i_L3}]
set_max_fanout 10 [get_ports {Req_i_W}]
set_max_fanout 10 [get_ports {Ack_o_L0}]
set_max_fanout 10 [get_ports {Ack_o_L4}]
set_max_fanout 10 [get_ports {Ack_o_N}]
set_max_fanout 10 [get_ports {Ack_o_L1}]
set_max_fanout 10 [get_ports {Ack_o_E}]
set_max_fanout 10 [get_ports {Ack_o_L2}]
set_max_fanout 10 [get_ports {Ack_o_S}]
set_max_fanout 10 [get_ports {Ack_o_L3}]
set_max_fanout 10 [get_ports {Ack_o_W}]
set_max_fanout 10 [get_ports {XR_in(1)}]
set_max_fanout 10 [get_ports {XR_in(0)}]
set_max_fanout 10 [get_ports {YR_in(1)}]
set_max_fanout 10 [get_ports {YR_in(0)}]
set_max_fanout 10 [get_ports {Ld_reg}]
```

A.2.2 Physical Synthesis : Cadence SoC Encounter

In order to do Physical Synthesis for performing Power & IR analysis, we port technology-mapped gate-level netlist along with the .SDC into Cadence SoC Encounter. In addition, the timing library (.tlf) and LEF (Library Exchange Format) files of the standard cells from IIT TSMC 0.18μ library are also input to SoC Encounter.

Encounter Configuration Setup: Initially, the configuration file (*encounter.conf*) must edited to have the path to the cell library properly set, wherein the paths to the Tim-

ing Library File (.tlf) and the LEF (Library Exchange Format) cell library are set. In this work, the path is set to the TLF and LEF of TSMC 0.18 library available from OSU (formerly available from IIT). In addition, the paths for the verilog netlist and the SDC file (outputted by Synopsys Design Compiler) must be set, along with the topmost design entity name. Note that at the time of this experimentation, Cadence SoC Encounter supported only the verilog gate-level netlist as design input. The buffer and inverter cell footprint must be properly set, to be used during optimization. Note that this step is only necessary only incase of a footprint-based physical synthesis and is optional. Detailed information regarding the footprintless is available in Cadence SoC Encounter User Manual. Also, the values for pin names must be set in the pin list. The encounter.conf file used for the five port Multi^2 Router is given below,

```
##############################################
#                                            #
#  FirstEncounter Input configuration file   #
#                                            #
##############################################

global rda_Input
set rda_Input(ui_netlist) "LWRouter_gate.v"
set rda_Input(ui_timingcon_file) "LWRouter_sdc.sdc"
set rda_Input(ui_topcell) "LWRouter"

set rda_Input(ui_netlisttype) {Verilog}
set rda_Input(ui_ilmlist) {}
set rda_Input(ui_settop) {1}
set rda_Input(ui_celllib) {}
set rda_Input(ui_iolib) {}
set rda_Input(ui_areaiolib) {}
set rda_Input(ui_blklib) {}
set rda_Input(ui_kboxlib) ""
set rda_Input(ui_timelib) "/home/sethurb/noc/iit_stdcells/\
    tsmc018/main/iit018_stdcells.tlf"
set rda_Input(ui_smodDef) {}
set rda_Input(ui_smodData) {}
set rda_Input(ui_dpath) {}
set rda_Input(ui_tech_file) {}
#set rda_Input(ui_buf_footprint) {BUFX2}
#set rda_Input(ui_delay_footprint) {BUFX2}
#set rda_Input(ui_inv_footprint) {INVX1}
#set rda_Input(ui_buf_footprint) {buf}
```

```
#set rda_Input(ui_delay_footprint) {buf}
#set rda_Input(ui_inv_footprint) {inv}
set rda_Input(ui_leffile) "/home/sethurb/noc/iit_stdcells/\
    tsmc018/main/iit018_stdcells.lef"
set rda_Input(ui_core_cntl) {aspect}
set rda_Input(ui_aspect_ratio) {1.0}
set rda_Input(ui_core_util) {0.7}
set rda_Input(ui_core_height) {}
set rda_Input(ui_core_width) {}
set rda_Input(ui_core_to_left) {}
set rda_Input(ui_core_to_right) {}
set rda_Input(ui_core_to_top) {}
set rda_Input(ui_core_to_bottom) {}
set rda_Input(ui_max_io_height) {0}
set rda_Input(ui_row_height) {}
set rda_Input(ui_isHorTrackHalfPitch) {0}
set rda_Input(ui_isVerTrackHalfPitch) {1}
set rda_Input(ui_ioOri) {R180}
set rda_Input(ui_isOrigCenter) {0}
set rda_Input(ui_exc_net) {}
set rda_Input(ui_delay_limit) {1000}
set rda_Input(ui_net_delay) {1000.0ps}
set rda_Input(ui_net_load) {0.5pf}
set rda_Input(ui_in_tran_delay) {120.0ps}
set rda_Input(ui_captbl_file) {}
set rda_Input(ui_cap_scale) {1.0}
set rda_Input(ui_xcap_scale) {1.0}
set rda_Input(ui_res_scale) {1.0}
set rda_Input(ui_shr_scale) {1.0}
set rda_Input(ui_time_unit) {none}
set rda_Input(ui_cap_unit) {}
set rda_Input(ui_sigstormlib) {}
set rda_Input(ui_cdb_file) {}
set rda_Input(ui_echo_file) {}
set rda_Input(ui_qxtech_file) {}
set rda_Input(ui_qxlib_file) {}
set rda_Input(ui_qxconf_file) {}
set rda_Input(ui_pwrnet) {vdd}
set rda_Input(ui_gndnet) {gnd}
set rda_Input(flip_first) {1}
set rda_Input(double_back) {1}
set rda_Input(assign_buffer) {0}
set rda_Input(ui_pg_connections) [list {PIN:vdd:} {PIN:gnd:} ]
set rda_Input(PIN:vdd:) {vdd}
        set rda_Input(PIN:gnd:) {gnd}
```

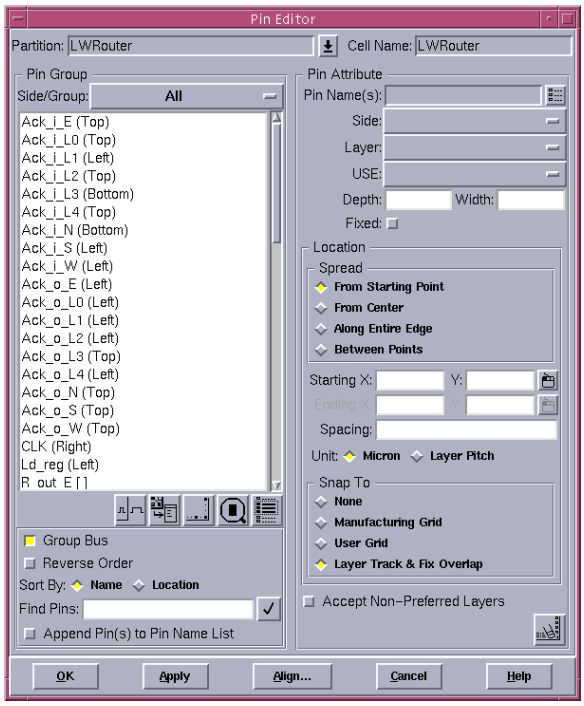


Figure A.3: Pin Editor

Encounter - From Floorplan till Detailed Routing & Verification: At the start, Cadence SoC encounter is invoked as follows,

encounter −config encounter.conf −init enc.tcl −log myrouter.log−overwrite

This will bring up the Encounter GUI with an empty design area. Note that the encounter must not be invoked as a background process (i.e., with an & at the end). The first step is to specify the floorplan, which is done by invoking, "Floorplan → Specify Floorplan". The core utilization is set at 50% (value = 0.5) and the space between the core and the boundary is set to 30μm on all sides (used for supply ring insertion - VDD, GND). The specified values and some of the following steps are loosely based on the First Encounter tutorial found at [Joh05]. It is argued that a 50% core utilization is ideal in terms of availability of space for buffer insertion during optimization. The location of the different ports are set (Top/Bottom/Left/Right corresponding to North/South/West/East) using the Pin Editor ("Edit → Pin Editor"). This is necessary in order to maintain the relative position for ease in identification and the integration of router modules when forming an NoC mesh (Figure A.3).

After initial floorplan and power rail (VDD/GND) definition, the power-track routing (special route command) and via-insertion are performed. Timing-driven placement,

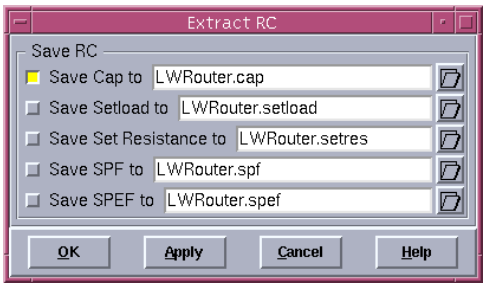


Figure A.4: Extract RC

clock tree insertion and detailed routing constitute the next phase of tasks, with the intermediate timing violations removed through an optimization phase. It is then followed by filler-cell insertion and verification, in order to check for various issues including a check for completeness in connectivity.

A step by step tutorial showing snapshots of the intermediate points are available at [Joh05] and is recommended to develop an initial understanding and develop a comfort level for the use of Cadence SoC Encounter. The purpose of this tutorial is to show the steps that are specific for performing a statistical power estimation and IR analysis, which require additional technology-specific files. The next sequence of steps constitute the power and IR drop analysis using Cadence SoC Encounter [Cad07a].

Parasitics Extraction: RC extraction must be completed before a power estimation can be be performed. This can be done in two ways. One, a simple RC extraction can be done (”Timing → Extract RC) as shown in Figure A.4.

Alternately, the *Fire & Ice* RC extractor [fir07] can be used for a more accurate parasitic extraction. Using the *Layer Map* file and *IceCaps* technology file (.tch file - having models for resistance and capacitance extraction in various layers) as input, the *GenLib* routine (a button inside the Fire & Ice RC Extractor window) is invoked to create a binary-view (.cl library) of the LEF cells (TSMC 0.18μ). The binary view has two key data, namely, the graycell data (for extraction-for-timing flows) and power-grid view of all cells. Also, Fire & Ice RC extractor [fir07] is used to generate the *Standard Parasitic Exchange File* (.spef). This RC extraction approach is recommended, because in addition to better accuracy, the binary-view cell library generation is necessary before doing an IR analysis using the VoltageStorm tool.

The snapshots of the major steps in the above process, namely, the binary-view cell generation (*GenLib* Routine) and *Fire & Ice* RC extraction (”Timing → Fire & Ice

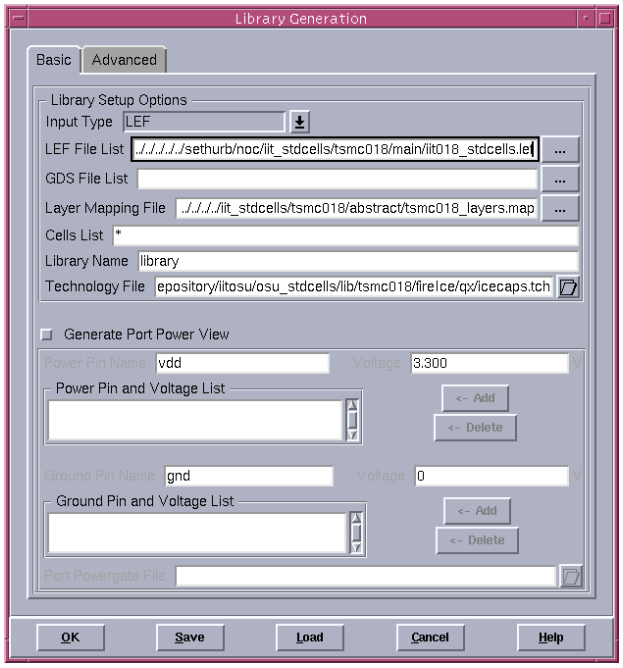


Figure A.5: GenLib Routine - Binary-view Cell Library generation

Extract RC) are shown in Figures A.5 and A.6, respectively. It is then followed by the parasitics-inclusive delay estimation (*SDF* file generation). At this point, the design must be saved ("Design → Save Design") so that the design can be restored (instead of performing synthesis from the scratch, starting from floorplan), when performing power and IR analysis for multiple times

Power Estimation & IR Drop Analysis: A corner case representing the worst case IR drop is very difficult to construct using vector based power analysis, and hence, we make use of the statistical power analysis ("Power → Analysis → Power Analysis → Statistical) using the typical parameter values having a net toggle probability of 0.5 and clock rate of 100MHz. As shown in Figure A.7, it is necessary to specify the appropriate options to generate the *Instance Power*, which is required for performing an IR drop analysis. At the end of statistical power estimation (Figure A.7), a report for average/peak power* is generated along with detailed instance power files, which are input to the VoltageStorm tool.

VoltageStorm is sign-off tool for performing a detailed rail analysis to find IR drop violations in the layout and the profile is displayed as a power graph [Cad07b]. VoltageStorm GUI ("Power → Analysis → Run Voltage Storm) and the Power Graph

*This step forms the basis for the analysis of average power variation discussed in Chapter 10.5.1.

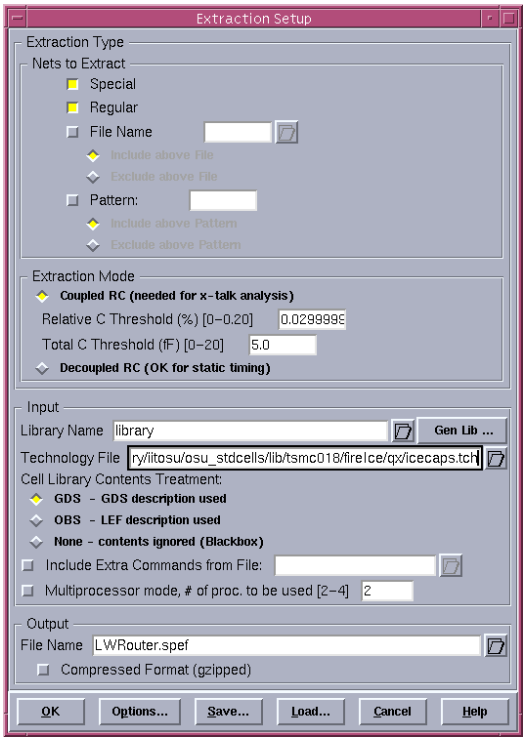


Figure A.6: Fire & Ice RC extractor

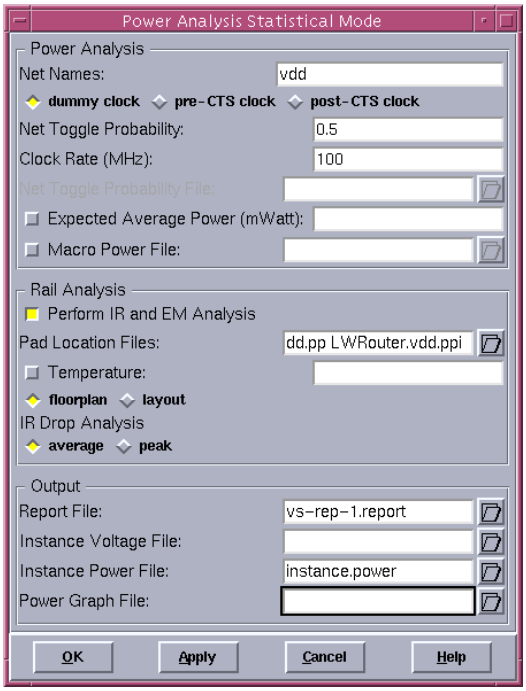


Figure A.7: Statistical Power Estimation

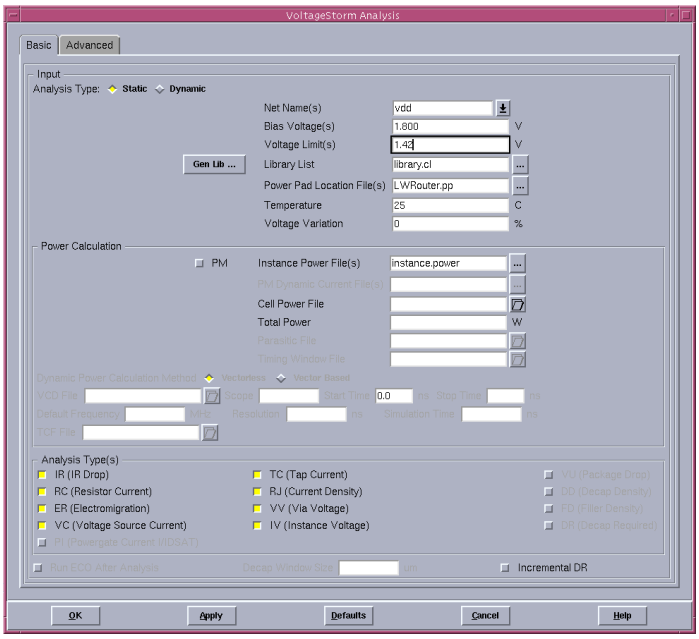


Figure A.8: VoltageStorm GUI

Display Browser (”Power → Analysis → Display → Display Rail Analysis Results) are shown in Figures A.8 and A.9, respectively.

A.2.3 Average/Peak Power Estimation - Iteration Procedure

The process of average/peak power estimation is needed to be automated, as it involves iteration covering hundreds of cases with various multi-port routers having various toggle probabilities. The iteration is controlled by a perl script as shown below.

```
#!/usr/local/bin/perl
%rep_file_name=(
        'average' => "br5-statistical-avg-results.txt",
        'peak'  => "br5-statistical-peak-results.txt"
);                                                                    5
#open in read only mode
open PWRTCLTEMPLATE, "<encouter-statistical-power-template.tcl"
  or die "Cannot open file : encouter-statistical-power-template.tcl";
@pwrtcldata = <PWRTCLTEMPLATE>;
close PWRTCLTEMPLATE;                                                 10
$ℓinecnt=@pwrtcldata;
foreach $c1 (keys (%rep_file_name))
{
  #open in append mode
```

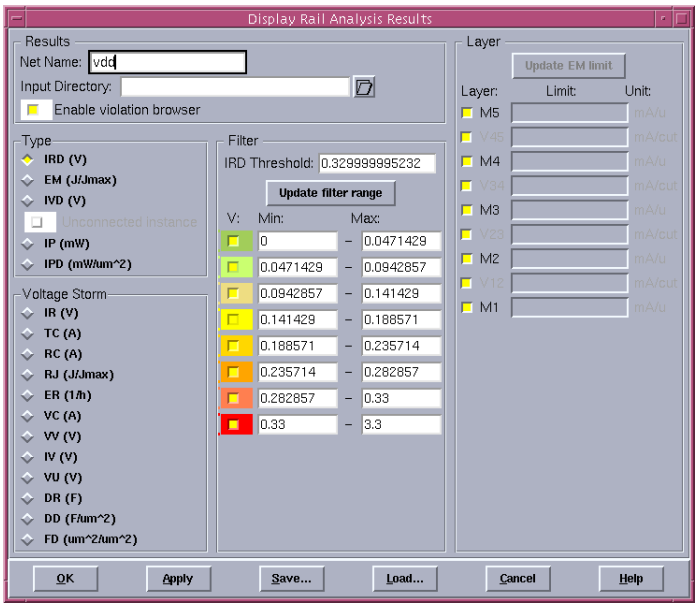


Figure A.9: Display Browser

```
open RPFILE, ">>$rep_file_name{$c1}" or die "Cannot open file: $rep_file_name{$c1}"; 15
print RPFILE "Format:\n
 avg/peak, toggle prob, temp, avg power, switching power, internal power,
 leakage power, avg power clocked, unclocked domain pwr, core power, block pwr,
 io pwr, worst IR drop, nodes in rail nw, worst EM m1, m2, m3, m4, m5, v12, v23,
 v34, v45, biggest toggled net, terminals, total cap\n";                    20
for($c3=20; $c3 <= 50; $c3++)
{
 for($c2=0.00; $c2 <= 1.00; ($c2=$c2+0.01))
 {
       #print "c1, c2, c3 = $c1, $c2, $c3\n";                               25
       #open in write/truncate mode
       open PWRTCLFILE,">encounter_statistical_power.tcl" or
            die "Cannot open file: encounter_statistical_power.tcl";
       $cnt = 0;
       while ($cnt <= $ℓinecnt)                                             30
       {
          if ($cnt == 12) # n
          {
            @encntr_str=split(/ /, $pwrtcℓdata[$cnt]);
            $encntr_str[2]    = $c1;                                        35
            $encntr_str[4]    = $c2;
            $encntr_str[10] = $c3;
            $pwr_tcl_prnt_str = join(" ",@encntr_str);
            $#encntr_str=-1;#reset the array
            print PWRTCLFILE "$pwr_tcl_prnt_str\n";                         40
```

```
        $pwr_tcl_prnt_str="";
      }
      else
      {
        print PWRTCLFILE $pwrtcℓdata[$cnt];
      }
      $cnt++;
}#end of while cnt <= linecnt
close PWRTCLFILE;

#Invoking Encounter
system("sh encntr_statistical.bat");

# Read in the power report and print data into the PowerResults file
#print string format: avg/peak, toggle prob, temp, avg power, switching power,
#internal power, leakage power,avg power clocked, unclocked domain pwr,
#core power, block pwr, io pwr, worst IR drop, #nodes in rail nw, worst EM m1,
#m2, m3, m4, m5, v12, v23, v34, v45, biggest toggled net, terminals, total cap
open PWRFILE, "LWRouter_Statistical.rep" or
  die "Cannot open file: LWRouter_Statistical.rep";
$pwrrpt_str = "$c1 $c2 $c3 ";
push(@select_str, $pwrrpt_str);
while(<PWRFILE>)
{
 @comp_str=split;
 if ($. == 5)
 {
   push(@select_str, $comp_str[2]);
 }
 elsif ($. == 6)
 {
   push(@select_str, $comp_str[3]);
 }
 elsif ($. == 7)
 {
   push(@select_str, $comp_str[3]);
 }
 elsif ($. == 8)
 {
   push(@select_str, $comp_str[3]);
 }
 elsif ($. == 11)
 {
   push(@select_str, $comp_str[4]);
 }
 elsif ($. == 12)
```

```
{
  push(@select_str, $comp_str[3]);
}
elsif ($. == 14)
{
  push(@select_str, $comp_str[1]);
}
elsif ($. == 15)
{
  push(@select_str, $comp_str[1]);
}
elsif ($. == 16)
{
  push(@select_str, $comp_str[1]);
}
elsif ($. == 18)
{
  push(@select_str, $comp_str[5]);
}
elsif ($. == 19)
{
  push(@select_str, $comp_str[6]);
}
elsif ($. == 21)
{
  push(@select_str, $comp_str[1]);
}
elsif ($. == 22)
{
  push(@select_str, $comp_str[1]);
}
elsif ($. == 23)
{
  push(@select_str, $comp_str[1]);
}
elsif ($. == 24)
{
  push(@select_str, $comp_str[1]);
}
elsif ($. == 25)
{
  push(@select_str, $comp_str[1]);
}
elsif ($. == 26)
{
  push(@select_str, $comp_str[1]);
```

```
        }
        elsif ($. == 27)
        {
          push(@select_str, $comp_str[1]);
        }
        elsif ($. == 28)
        {
          push(@select_str, $comp_str[1]);
        }
        elsif ($. == 29)
        {
          push(@select_str, $comp_str[1]);
        }
        elsif ($. == 30)
        {
          push(@select_str, $comp_str[3]);
        }
        elsif ($. == 31)
        {
          push(@select_str, $comp_str[3]);
        }
        elsif ($. == 32)
        {
          push(@select_str, $comp_str[2]);
        }
      }
      $pwr_prnt_str = join(" ",@select_str);
      $#select_str=-1;#reset the array
      print RPFILE "$pwr_prnt_str\n";
      $pwr_prnt_str="";
      close PWRFILE;
    }#c2
  }#c3
  close RPFILE;
}#c1
```

After necessary modifications to the *encounter_statistical_power.tcl*, SoC Encounter (`sh encntr_statistical.bat`) is invoked to do the power and IR drop analysis (command line mode) as shown below.

```
encounter -init encounter-statistical-power.tcl -log power-myrouter.log -overwrite
```

Here, Encounter is invoked using the *Restore Design* option, using the options present in *encounter_statistical_power.tcl* as follows.

```
# Restore the database
restoreDesign LWRouter_final.enc.dat LWRouter

# Extract for power Analysis
isExtractRCModeDefault
isExtractRCModeSignoff
setExtractRCMode -detail -noReduce
isExtractRCModeSignoff
autoFetchDCSources vdd
extractRC
delayCal -sdf LWRouter.sdf -idealclock

#Statistical Power Estimation
updatePower -irDropAnalysis peak -toggleProb 0.5 -clockRate 100 -pad LWRouter.pp
    -temperature 25 -report LWRouter_Statistical.rep -mode layout vdd

exit
```

Appendix B

List of Publications

1. Jawad Khan, Balasubramanian Sethuraman and Ranga Vemuri. "A Power-Performance Tradeoff Methodology for Portable Reconfigurable Platforms". In *ERSA'04*, pages 33-37, Las Vegas, Nevada, C.S.R.E.A. Press, June 2004.

2. Balasubramanian Sethuraman, Jawad Khan and Ranga Vemuri. "Battery-Efficient Task Execution on Portable Reconfigurable Computing Platforms". In *IEEE International SOC Conference: SOCC 2004*, pages 237-240, April 2004.

3. Balasubramanian Sethuraman, Prasun Bhattacharya, Jawad Khan and Ranga Vemuri. "LiPaR: A Light-Weight Parallel Router for FPGA-based Networks-on-Chip". In *15th Great Lakes Symposium on VLSI (GLSVLSI 2005)* pages 452-457, Chicago, 2005.

4. Balasubramanian Sethuraman and Ranga Vemuri, "optiMap: A Tool for Automated Generation of NoC Architectures using Multi-Port Routers for FPGAs", In *Design Automation and Test Europe, (DATE 2006)*, Munich, Germany, 2006.

5. Balasubramanian Sethuraman and Ranga Vemuri, "Multi^2 Router: A Novel Multi Local Port Router Architecture with broadcast facility for FPGA-based Networks-on-Chip", In *Field Programmable Logic & Applications conference (FPL 2006)*, Madrid, Spain, August 2006.

6. Balasubramanian Sethuraman, "Novel Methodologies for Performance & Power Efficient Reconfigurable Networks-on-Chip", In *PhD Forum - Field Programmable Logic & Applications conference (FPL 2006)*, Madrid, Spain, August 2006.

7. Balasubramanian Sethuraman and Ranga Vemuri, “A Force-directed Approach for Fast Generation of Efficient Multi-port Architectures”, In *20th International Conference on VLSI Design*, Bangalore, India, January 2007.

8. Balasubramanian Sethuraman and Ranga Vemuri, “Multicasting based Topology Generation and Core Mapping for a Power Efficient Networks-on-Chip”, In *IEEE/ACM-SIGDA International Symposium on Low Power Electronics and Design (ISLPED 2007)*, Portland, Oregon, USA, August 2007.

9. Balasubramanian Sethuraman and Ranga Vemuri, “Power Variations of MultiPort Routers in an Application-Specific NoC Design : A Case Study”, In *XXV International Conference on Computer Design (ICCD 2007)*, Lake Tahoe, California, USA, October 2007.

10. Balasubramanian Sethuraman and Ranga Vemuri, “A Methodology for Application-Specific NoC Architecture Generation in a Dynamic Task Structure Environment” (under review).