ENERGY EFFICIENT COMPUTING IN FPGA THROUGH EMBEDDED RAM BLOCKS

by

ANANDAROOP GHOSH

Submitted in partial fulfillment of the requirements

for the degree of Master of Science

Thesis Advisor: Dr. Swarup Bhunia

Department of Electrical Engineering and Computer Science

CASE WESTERN RESERVE UNIVERSITY

May, 2013

CASE WESTERN RESERVE UNIVERSITY
SCHOOL OF GRADUATE STUDIES

We hereby approve the thesis / dissertation of

ANANDAROOP GHOSH

MASTER OF SCIENCE

candidate for the _____degree*

SWARUP BHUNIA

signed by _____

(Research Advisor)

CHRISTOS PAPACHRISTOU

(Committee Member)

FRANCIS MERAT

(Committee Member)

Date: **03/07/2013**

*We also certifiy that written approval has been obtained for any proprietary material contained therein.

Contents

List of Tables

List of Figures

Acknowledgements

I have countless people to acknowledge for whom I have been able to complete my graduate studies. First I want to thank my parents for their help and mental support throughout the thick and thins of my graduate life. My mother has always been a constant source of encouragement for me. My friends at Case were also a great source of knowledge and encouragement throughout my tenure at Case. Secondly I want to thank my advisor Professor Swarup Bhunia for his technical and financial support he has provided me during my grad studies. Starting from an idea to maturing it to an attractive research, to writing a technical article, I have learnt loads from his expertise. I also want to thank my labmates, especially my seniors who guided me during the initial days of my grad research. Starting from technical discussions to how to balance grad life, their inputs have always been extremely helpful for me. The courses I have taken at Case Western have been an integral part of my academic development and I would sincerely like to thank all the course instructors. Finally I want to thank Professor Chris Papachristou and Professor Frank Merat for acting as my thesis committee members.

List of Abbreviations

ASIC	Application Specific Integrated Circuit
RAM	Random Access Memory
EMB	Embedded Memory Block
CLB	Configurable Logic Block
CPU	Central Processing Unit
DFG	Data Flow Graph
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
IC	Integrated Circuit
ITRS	International Technology Roadmap for Semiconductors
LUT	Look-Up Table
MSB	Most Significant Bit
LSB	Least Significant Bit

Energy Efficient Computing in FPGA through Embedded RAM Blocks

Abstract

by

ANANDAROOP GHOSH

FPGAs have emerged as the preferred prototyping and accelerator platform for diverse application domains like digital signal processing (DSP), security, and multimedia having real time performance requirements. Applications in these domains are often dominated by complex compute-intensive operations requiring implementation of complex datapaths or functions e.g. transcendental functions. Conventional spatial mapping of these operations to the configurable logic blocks (CLBs) or embedded DSP blocks of a FPGA device imposes a major bottleneck in energy efficiency. In this thesis, we propose to use embedded memory blocks (EMBs) in FPGA for energy-efficient mapping of these operations. We select appropriate parts of an application for mapping into embedded memory blocks in a heterogeneous mapping framework that aims at maximizing energy efficiency. Complex operations are decomposed / fused into large multi-input multi-output look-up tables, mapped into EMBs and evaluated through sequential access of them. Optimal energy configuration of the embedded memory blocks are determined and effectiveness of the proposed methodology is evaluated for a set of common applications using a commercial state-of-the-art FPGA system (Altera Stratix IV). The proposed work also builds a strategy for energy-accuracy trade-off for multimedia applications and leverages the effectiveness of memory based computing in FPGA for such approximate computations.

Chapter 1

Introduction

Chiefly the research objectives, the already existing works, the outline and the contribution of the thesis have been stated in this chapter. The increasing use of reconfigurable platforms like Field Programmable Gate Array (FPGA) have motivated research in energy efficient application mapping in FPGA which is the main objective of this work. FPGAs are increasingly used in embedded applications (such as digital signal processing, multimedia, security and graphics) due to the flexibility in application mapping, reduced design cost and improved time-to-volume. FPGA has also emerged as a preferred coprocessor platform providing higher performance while working in conjunction with a CPU for a variety of applications with real-time processing requirements [34]. However, these platforms are well-known to suffer from poor energy efficiency, primarily due to large overhead of their elaborate programmable interconnect (PI) fabric. Currently the PI accounts for 80% of power and 60% of delay in FPGAs at scaled process technologies [4]. For resource-constrained embedded systems, it is extremely important to minimize the energy requirement and hence, there is a growing need to address the energy issues in reconfigurable platforms while retaining their performance and flexibility advantages [2].

FPGA vendors such as Xilinx and Altera, as well as academic researchers have in-

vestigated various device engineering options (such as low-k dielectric, multiple device thresholds) [23] as well as architecture-level techniques (e.g. clustered architecture) [24] to improve the energy consumption of the FPGA devices. These optimization approaches however, cannot provide adequate solution to reduce the energy requirement for many compute-intensive applications. The energy consumption is dominated by the routing energy for compute intensive applications, which result in comparatively lesser improvements at the application level. Moreover, as the interconnect delay does not scale as well as logic delay, fine-grained architectures of FPGAs suffer from poor technological scalability of performance and energy than custom implementation of a design.

An efficient application mapping methodology that can drastically reduce the need of PIs for an application while using conventional FPGA architecture, can be extremely effective in reducing energy consumption. Such an approach can be attractive to both FPGA vendors and users alike, since it does not require modifications in FPGA hardware and thus saves device fabrication cost. Furthermore, it can work on legacy hardware. In this thesis, we propose a novel application mapping methodology for FPGA that aims at achieving these objectives. We propose using the embedded memory arrays in FPGA for mapping compute-intensive parts of an application. We note that modern FPGAs come with large number of embedded memory blocks (EMBs). For example, Altera Stratix-IV devices are equipped with 17-33 Megabits (Mb) of block random access memory (RAM), in addition to the one-dimensional lookup tables (LUT) for CLBs. As shown in Fig. 1.1, driven by aggressive technology scaling, FPGA devices from different vendors are integrating larger amount of RAM with faster access speed in each technology generation. There has also been a significant reduction in the access energy of an embedded block RAM across technology generations (e.g. 53% from Stratix II to Stratix III and 48% from Stratix III to Stratix IV). While many applications use the EMBs for storing input

and intermediate data during processing, large part of memory may remain unused for many compute-intensive applications [3]. Hence, if these memory blocks can opportunistically be used for computation - in particular, to realize complex datapaths or functions, one can significantly improve both energy consumption and resource utilization by reducing the combinational elements and routing power required for purely logic based implementation.

We observe that fine-grained architecture of conventional FPGA devices cannot efficiently map complex coarse-grained operations, like complex datapath (like filtering) operations and complex transcendental functions, which demand large amount of reconfigurable logic and PI resources. At the same time, the large number of DSP blocks available in an FPGA device can efficiently map coarse-grained functions for higher bitwidth. But at lower bitwidth, energy improvement is small primarily due to poor resource usage at smaller bitwidth [33]. For example, when a functional unit in DSP block operates as a multiplier, the minimum configurable multiplier width is 9×9 resulting in unused logical resources for smaller bitwidth operations. On the other hand, judicious mapping of complex operations in EMBs as large two-dimensional LUTs can be highly effective to reduce the energy requirement. Earlier works have considered mapping fine-grained applications like ISCAS and MCNC benchmarks in EMBs inside a FPGA [14], [16]. These investigations lay the foundation of using EMBs for computation. However, they primarily focus on mapping small Boolean functions in fine-grained applications (e.g. control logic). Furthermore, use of EMBs for mapping logic operations in earlier work is driven by performance or throughput improvement. The authors in [17] show the power implications of mapping logic in embedded RAM for Stratix-1 series of devices. Due to power-hungry nature of EMBs in previous generation FPGA devices, mapping logic in embedded memory resulted in considerable increase in dynamic power compared to the implementation in CLBs. Several other works have also been reported on mapping transcendental functions like

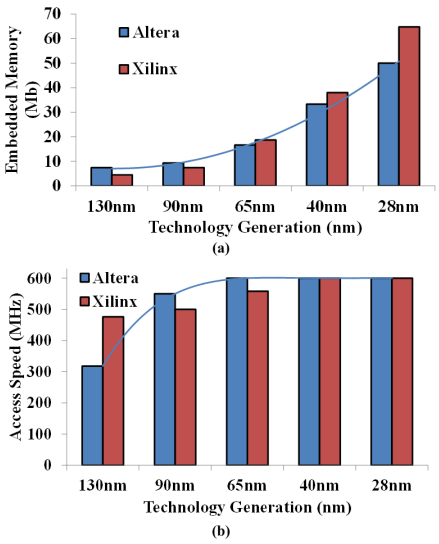

Figure 1.1: The trend in embedded memory in FPGA: a) size (Mb), and b) access speed (MHz) for Altera Stratix [11] and Xilinx Virtex [12] series of FPGA devices across different technology generations.

arctan in an embedded RAM of a FPGA [35]. However, existing works in this area do not consider optimization of energy exploiting the properties of nanoscale embedded memory blocks. They also do not provide a comprehensive heterogeneous mapping flow. Finally, existing works also do not consider dynamic trade-off between energy and output quality in the memory based computing framework.

Using a commercial FPGA platform, we have shown that compared to conventional mapping approaches, the proposed mapping approach achieves significant improvement in energy requirement for a set of representative applications. This is due to the fact that typical implementation of complex operations require multiple logic levels connected through PIs, thus incurring large overhead in performance and energy. On the contrary, a memory based mapping approach where a large multiple input/output function is computed in one or few lookup table (LUT) accesses significantly improves the latency of operation and energy due to large reduction in PI overhead.

1.1 Research objectives

FPGA have traditionally been used as a prototyping platform for new design methodologies and ASIC implementations. However due to the increasing resources inside a FPGA like embedded multipliers and embedded memory as well as logic blocks at low power consumption and lower cost coupled with lower time to market, current commercial FPGAs have been found to be extremely helpful for several application domains and have been able to successfully replace ASIC in several scenarios. However still the power consumption is a huge issue in computing with FPGA as increased programmability leads to higher energy consumption. This thesis is aimed at exploring energy efficiency in FPGA in terms of mapping applications more efficiently by exploiting the heterogeneity in current commercial FPGAs. Such a heterogeneous application mapping strategy, especially the opportunistic use of embedded memory blocks have been able to provide significant energy efficiency in DSP/ multimedia applications. Finally the thesis also investigates the role of memory based computing in approximate computation for several applications and shows the advantages obtained compared to conventional logic / DSP based computations.

1.2 Thesis Outline

From inception to completion, this thesis is dedicated in analyzing and developing a memory based heterogeneous computing strategy inside a FPGA and the energy consumption savings with respect to normal logic based computing as well as logic + DSP based computing for several application scenarios. Secondly it also shows the advantage of such a framework for performing energy accuracy trade-off at run-time for several DSP and multimedia applications.

In chapter one, we have described the requirement of energy efficient computing in FPGA and the research objectives coupled with the contribution of our work. The background and motivation will be described in chapter two. It also describes

the motivation of the proposed approach and the different function / application types which would gain maximum energy efficiency through the proposed approach. Chapter three describes in detail with the application mapping methodology for the proposed framework, starting from the decomposition of functions, to fusion of individual operations to packing of fused operations in the available resources and finally placement and routing of the synthesized hardware. It also describes the memory logic interface and the timing strategy of the proposed heterogeneous framework. Next it describes the optimal mapping of RAM blocks in FPGA to achieve energy efficiency followed by the energy consumption comparison of using embedded memory for mapping a unit coarse grained datapath compared to conventional logic based and DSP based mapping over different technology generations and input bitwidth. Secondly it also shows the use of EMBs for mapping compute intensive transcendental functions and the energy savings achieved in the corresponding cases over different input bitwidth of operand. The energy savings obtained in these unit functions using the proposed mapping approach is chiefly exploited at the application level in order to achieve lower energy consumption. Chapter four deals with the performance of the proposed application mapping procedure for several DSP and multimedia applications over different bitwidths of input operands and the energy savings compared to conventional computing in Stratix IV FPGA. Applications mapped are mainly of two types-applications dominated by complex datapaths and applications dominated by transcendental functions. In chapter five, finally the usefulness of the proposed framework for performing dynamic energy accuracy trade-off is described and compared with conventional approaches for the same set of applications to showcase the advantages of the proposed framework at lower precision scenarios. Specifically the application of a 32-tap FIR filter is taken and judicious truncation techniques are applied in order to achieve graceful degradation of output quality at iso-energy consumption. For the filtering example, both conventional and distributed arithmetic

implementations are studied and the energy savings of the proposed approach are shown compared to conventional mapping approaches. Finally in Chapter six, we describe the conclusions and the future work which can potentially improve the already proposed work.

1.3 Contributions

The chief contributions of this work are as listed below:

1. It analyzes the effectiveness of mapping coarse-grained compute-intensive operations in an application to EMBs in FPGA. It also explores the most energy-efficient memory configuration in a FPGA for mapping operations with varying memory requirements.

2. It then develops a hybrid application mapping framework, which combines conventional application mapping in logic and DSP blocks (for DSP-enhanced FPGA devices) with judicious mapping of specific computations in memory. It determines the complete mapping methodology including functional decomposition, which partitions complex functions into multi-input/multi-output LUTs of manageable size, fusion of small operations into a large one, and finally optimal packing of operations into a combination of EMBs and logic array. It considers applications from diverse domains with varying datapath width.

3. It validates the effectiveness of the proposed methodology by mapping a number of common signal processing, scientific and graphics applications on a commercial state-of-the-art FPGA platform (Altera Stratix IV, 40nm process).

4. It shows that computation with memory can be an effective vehicle for energy-accuracy tradeoff at run time. In this case, operand bitwidth for complex operation can be dynamically truncated to achieve exponential reduction in memory

space leading to large saving in computation energy. We demonstrate the effectiveness of such dynamic trade-off for a common filtering application using conventional arithmetic (CA). Furthermore, noting that some realizations of DSP applications resort to distributed arithmetic (DA) to improve area and performance, we extend the approach to DA based implementation.

Chapter 2

Background And Motivation

In this chapter, we propose the already existing works for improving energy efficiency in FPGA. Researchers from industry as well as academia have proposed several techniques at the circuit level, logic level, architecture level as well as software level in order to improve the usability of FPGA for computing. We describe the circuit/architecture level approaches previously proposed in the following sections. Then the motivation of the proposed approach is described. Finally we describe the concept of computation with memory and the specific functions or operation types which is amenable for mapping in memory.

2.1 Energy-Efficient Design Techniques in FPGA

Several system and device-circuit-architecture level low-power design techniques have been applied to FPGA in order to improve its power consumption. A comprehensive description of the power optimization techniques have been provided in [31]. The techniques can be broadly classified into three categories:

Device/Circuit Level Techniques: a) Clock gating can be effectively used to turn off the unused portions of the FPGA resources in order to minimize propagation of undesired signal values [28]. b) Dynamic voltage scaling can be used to adapt the

supply voltage of a FPGA according to the temperature of the device during operation [29]. Efficient circuit implementation techniques have been employed and improved across generations in order to address the power/performance issues of logic blocks, routing resources, and I/O resources. c) Moreover, the logic array blocks (LABs) are made with a mix of high threshold (Vt) and Low Vt transistors to achieve lower power while maintaining the performance target in a particular technology node [30].

Architecture Level Techniques: a) Sizes of the LUTs present in the configurable logic blocks have been increased from 4 to 7-input in order to map larger functions in a single LUT, thereby significantly improving on routing power consumption [6]. b) Generally for embedded RAM, coarse grained block based architecture gives better power/energy results compared to fine-grained RAM architecture [31]. c) The addition of more specialized units like DSP blocks and more embedded memories have helped improve energy consumption for specific functions which can otherwise be mapped using a large number of logic blocks with significant combinational cell power and routing power consumption. d) Finally, most vendors have also improved the routing distance between the neighboring logic blocks that can be reached using 1-hop or 2-hop paths in order to reduce the average capacitance of the routes [5].

Software Techniques: Energy-efficient mapping algorithms for legacy FPGA hardware have been investigated earlier e.g. [36]. However, existing software techniques primarily focus on reducing PI overhead in logic and DSP based implementations. They use embedded RAMs only for storing large input data or intermediate values and do not leverage on the prospect of mapping complex operations in memory.

2.2 Computation with Memory

Computation with memory is a LUT based computation approach, where the function response is stored in a 2D memory array. The same memory array may also

store multiple LUTs and computing is done by accessing the LUT with the right address values. The table values are pre-computed and used during application mapping process. LUT based realization has been traditionally used as a method for computing complex functions to avoid logic based computation, which may be very expensive in resource requirements. However, the downside of the LUT based computation is the exponential growth of the memory size with the increase in resolution of the input operands. As a result, without the presence of effective decomposition techniques or approximate computing techniques as proposed in [9] and [10], LUT based computation can only be possible in FPGA up to a certain LUT input size. Altera Stratix IV FPGA devices support a maximum LUT input size of 16 in a single EMB. In general application mapping scenarios, a large part of the existing embedded RAM in a FPGA remain unutilized. Authors in [14] and [16] have proposed using the idle memory blocks for fine-grained logic computation. It has also been shown that performance or throughput of an application can be improved through effective computation in memory [14], [16].

2.3 Motivation for the Proposed Approach

Due to the power hungry nature of the interconnects, if for a given complex function, the maximum LUT input size is not violated then it is highly beneficial from an energy perspective to map the function on to the embedded memory. For DSP applications, which lend themselves to energy-accuracy trade-off, dynamic truncation of primary input operands can often provide significant improvement in energy efficiency at graceful degradation of output quality when the energy budget is limited. At the same time, computation at consistently lower input resolutions often provide sufficient output quality for many DSP applications [25]. The proposed mapping strategy achieves significant improvement in energy-efficiency using a heterogeneous

mapping procedure. The main advantages achieved is chiefly due to the following reasons: a) improvement in PI requirement; b) improvement in access energy for nanoscale memories; c) for DSP applications which are amenable to energy-accuracy trade-off, memory provides a better computing fabric compared to fine-grained logic or DSP blocks which either have high PI overhead or the hardware far exceeds the requirement.

2.4 Use of Embedded Memory Blocks for Computation

It is well-known that a majority of scientific and graphics applications include a set of common compute-intensive kernels, which essentially constitute of basic mathematical operations such as addition, multiplication as well as evaluation of many complex functions such as sine, cosine, reciprocal, arctan, square root, exponentiation, and logarithm [1]. Traditionally, in an FPGA framework the transcendental functions are mapped using CORDIC approach [7] or Taylor series expansion [8], which either requires large number of computing resources or suffers from large latency. Evaluation of these functions by holding the output response of a function as LUT in the embedded memory array is an attractive solution. Besides, use of memory blocks in computing offers enormous parallel computing resources due to the presence of large (hundreds to thousands) number of embedded RAM blocks in a modern FPGA device. For complex transcendental functions, efficient decomposition techniques proposed in [9] and [10] have been employed in order to achieve improvement for larger bitwidth computations. The focus of our work is to achieve energy efficient mapping of the complex functions instead of performing memory-latency trade-off as illustrated in [15].

Memory based computing can also be very effective in case of complex datapaths

having multiple combinational levels, just as in the case of two constant coefficient multiplications followed by an addition. This kind of datapath is very common in many multimedia and signal processing (e.g. discrete cosine transform, filtering) applications. Investigations done in this thesis point to the fact that for complex datapaths, memory based computation can give improvement for up to 8-bit of input resolution in many cases. We note that an effective mapping methodology, which can leverage such decomposition algorithms and large parallel computing resources offered by embedded high-density RAM blocks in FPGA, can provide significant improvement in performance and energy-efficiency using the memory based computing model. The following computations have been identified to be amenable for mapping to memory:

1. Any function satisfying the maximum LUT input size (I). The function can be a regular transcendental one (such as $Sine(x)$) where x has a resolution of $\leq I$) or any arbitrary function with input size of $\leq I$, obtained by fusing many simple ones.

2. Complex datapath having constant coefficients like two constant coefficient multiplications followed by an addition, with total input size $\leq I$.

3. Any function that is easily bit-sliceable such as logic and Galois field operations and some datapath operations like addition, multiplication, etc.

4. Functions with arbitrary input width which can be realized by cascading LUTs, each of input size $\leq I$, e.g. transcendental functions with fixed and floating point operands.

5. Functions which are amenable to decomposition into acceptable input size in other ways. For example, a multiplication realized using logarithmic number system is converted from a two operand function to a single operand function.

The following chapter describes in detail the proposed application mapping method-

ology which will be finally used to map applications which are dominated by one or more above mentioned functions so as to understand the impact of such a heterogeneous application mapping process.

Chapter 3

Application Mapping Methodology

In this chapter, we describe the application mapping process using a combination of EMBs and other FPGA resources to maximize energy efficiency for a given application. Secondly we also describe the energy efficient configuration of RAM blocks which can contribute maximum energy efficiency at the application level. Finally we also describe the mapping of a unit complex datapath and complex transcendental function using the proposed approach across different input bitwidth to understand the energy savings achieved compared to normal logic / DSP based implementation.

3.1 Mapping Flow

Figure 3.1 shows the application mapping process for the proposed framework. The input to our application mapping flow is a control data flow graph (CDFG) containing a hypergraph representation (G(V,E)) of the input application. The operations in the input application constitute the set of vertices (V) and the dataflow between them is represented by the edges (E) in the hypergraph. Types and subtypes of the operation nodes currently supported in the CDFG representation are summarized in Table 3.1. Each operation node is characterized by the following fields: (i) *name*, (ii) *type*, (iii)

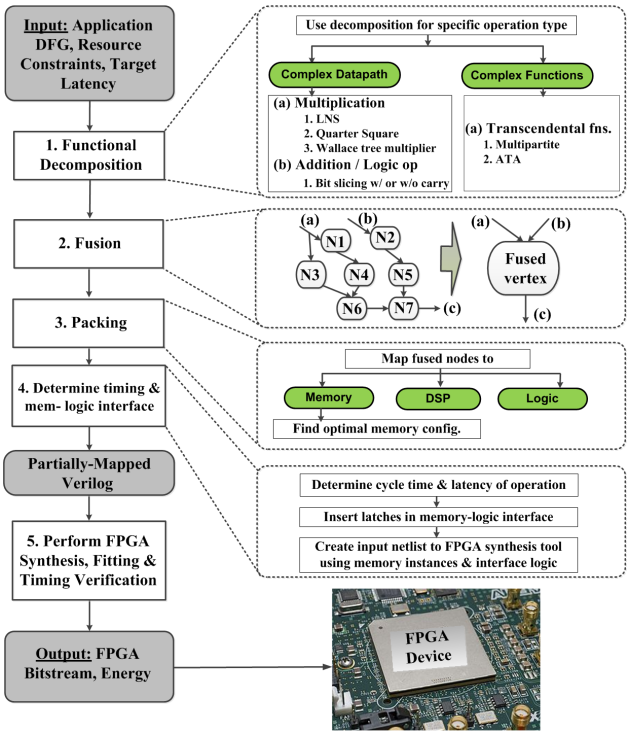

Figure 3.1: Application mapping steps using EMBs for computation.

subtype, (iv) *inputs*, (v) *outputs*, and (vi) *bitwidth*. '*name*' field specifies the name of the operation and is unique. Each '*type*' can have multiple *subtypes*. For example addition and subtraction both fall under the greater type of operations that can be bit-sliced into sub-operations with a smaller bitwidth and with a carry function (denoted as bitswC). '*bitwidth*' denotes the bitwidth of the input operands. '*bits*' represents operations which are bit-sliceable without carry. '*mult*' denotes conventional two-input multiplication. '$shift$' and '*rotate*' has one operand and the shift / rotate direction as subtype. '*complex*' denotes a general class of LUT operations. Maximum number of inputs for each type of operation is fixed, except for select type, which represents N to 1 selection, where N is a variable. The proposed application mapping flow is implemented in C language and is here after referred to as the FPGA mapper tool. Major steps of the application mapping flow are described below.

Table 3.1: Operation types supported in the FPGA mapper tool

Instr Type	Instr Subtype	Inputs	Outputs	Bitwidth	Comment
bitswC	add	a, b, cin	d, e	$x_1, x_2, 1$	bit-sliceable w/ carry
bits	xor	a, b,	c	$x_1 x_2$	bit-sliceable w/o carry
mult	mult	a, b	prod	x_1, x_2	two-in mult
shift	left/right	a,	a_s shftamt	x_1, x_2	shift op w/ shftamt
rotate	left/right	a,	a_r rotamt	x_1, x_2	rotate op w/ rotamt
complex	rand	a,..,sel	out	$x_0, .., x_n$	select one from n
const multadd	rand	x_0, x_1, C_0, C_1	out	x_0, x_1, C_0, C_1	const coeff mult add
sel	rand	a,..	out	$x_0, .., x_n$	select one from n

3.1.1 Functional Decomposition

Decomposition is the process of replacing a vertex in the hypergraph with large input/output count into multiple vertices satisfying the input-output constraint for the LUTs. For example, operations which are bit-sliceable, transcendental functions (e.g. sine, arctan, etc.) and other complex functions, which can be appropriately decomposed are broken down in this step into vertices satisfying the maximum LUT input/output count for a target FPGA device. The resource constraints of the FPGA along with the maximum available RAM input size are read from a parameter file, which acts as an input to the FPGA mapper tool. Transcendental functions and other complex functions with fixed-point operands can be decomposed using either (i) bipartite / multipartite [9] or (ii) ATA (Add-Table lookup-Add) [10] based methods in order to address the exponential increase in memory requirement with the increase in LUT input size.

The idea behind bipartite and multipartite decomposition is to divide the input

space (2^{α} segments) into 2^{γ} larger intervals (where $\gamma < \alpha$) so that the slope of the function is considered a constant in the larger interval. Hence it contains 2^{γ} tables of offsets, each with 2^{β} tables of offsets. This allows storing a total of $2^{\alpha} + 2^{\gamma+\beta}$ values instead of $2^{\alpha+\beta}$. For transcendental functions (such as sine, cosine, $\log_2$x, 2^x etc), further reduction in memory requirement at the cost of increased latency can be achieved through multipartite methods. Another method for reducing the memory requirement is the Addition Table Addition or ATA method. The main premise is to approximate a function $f(x)$ using Taylor series central difference method and the errors induced in the approximation are calculated in each stage to ensure that the error is less than the unit in the last place (ULP). The basic method involves parallel additions, followed by parallel lookups and finally multi-stage additions. This method can be used to approximate all polynomial and transcendental functions which can be represented in a given fixed point resolution. For functions which are decomposable with both multipartite and ATA, multipartite based decomposition is selected as it requires smaller memory at iso-output accuracy. A list of decomposition routines, which are included in the proposed mapping flow is shown in Fig. 3.1.

3.1.2 Fusion

Fusion involves opportunistic reduction of the total number of operations after decomposition by combining multiple operations into a single operation, which can be suitably mapped into a LUT. It incorporates two routines: (i) fusion of random LUT based operations, and (ii) fusion of bit-sliceable operations. Figure 3.2 shows an example of fusion of multiple nodes into a single node satisfying the input / output constraint of the available RAM blocks in the selected FPGA device. We have developed a heuristic for partitioning a target application into multi-input multi-output partitions. The partitioning is based on an input granularity, which indicates how fine-grained decomposition of an input operand can be done. The vertices inside each

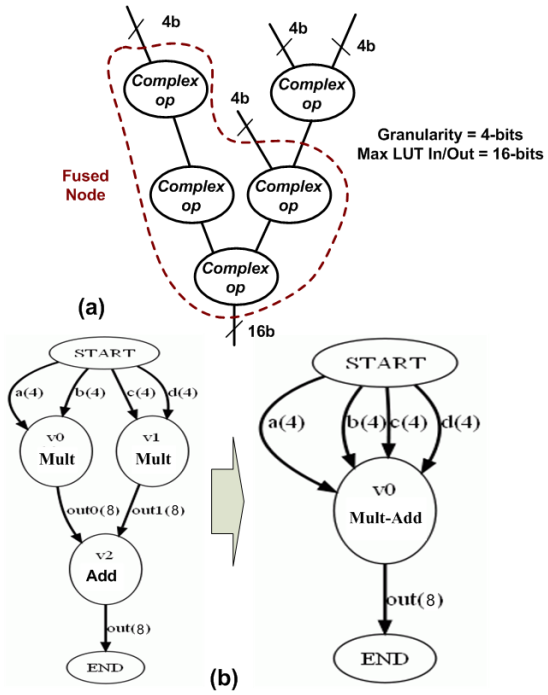

Figure 3.2: (a) Example of fusion of multiple nodes into a single node satisfying the LUT input/output count; (b) Specific example of mult-add fusion for 4-input operands.

partition are then fused to form a single vertex to be mapped as a LUT operation. This heuristic is inspired from the Maximum Fanout Free Cone (MFFC) and Maximum Fanout Free Subgraph (MFFS) approach as outlined in [14]. Through off-line analysis using FPGA synthesis tool, we evaluate the effectiveness of mapping each fused node of specific input/output width to embedded memory. This is done by mapping the function realized in the fused node using three alternative implementations (memory, logic and DSP-based) and comparing their energy behaviors.

The datapath operation in a DFG, as described in Fig. 3.2(a), provide opportunities for fusion of number of operations into a more complex one. For the datapath illustrated in Fig. 3.2(a), following vertex types can be obtained after fusion: 1) multadds: stands for the function $(a*b+c) << shiftamt$; 2) multaddsel: stands for the function $sel?(a*b+c) : c$; 3) adds: stands for the function $(a+b) << shiftamt$; 4) addsel: stands for the function $sel?(a+b) : a$; 5) adds: stands for the function

$(b + c) << shiftamt$; 6) multadd: stands for the function $a * b + c * d$; 7) mults: stands for the function $(a * b) << shiftamt$; 8) multsel: stands for the function $sel?(a * b) : c$. Note that the constraint for fusion of the datapath operations is that they must share one or more input operands. The fusion process is constrained by the number of input/output bits and granularity. Figure 3.2(b) shows an example fusion of two multiplications followed by an addition (multadd) into a single node where the total input and output operand bitwidth of the resultant fused node is 16 and 8, respectively.

3.1.3 Packing

The decomposed and fused nodes in the DFG act as the input to the packing stage. The sequence of packing steps are illustrated in Fig. 3.3. As shown in the figure, we use a mapping database in this stage which contains a-priori knowledge about energy-efficient mapping of different operations and input bitwidth. Such a database can be created through off-line analysis of different operation types/sizes for a target FPGA device. For mapping an application to memory, we derive energy-optimal configuration for the required LUT size. During the mapping step, we also consider specific resource constraints of the FPGA device. For example, in case of mapping to memory, if the number of particular memory block types (say M9K or M144K for Stratix IV) present in the selected device is insufficient, then the next best energy efficient configuration is attempted to fit in the device. On successful determination of the final implementation strategy for each individual fused node, we create resource-binding information for all nodes. For the nodes which are mapped to memory, we store the optimal memory configuration. For these nodes, we also create LUT contents at this stage.

3.1.4 Memory-Logic Interface and Timing Strategy

For a heterogeneous mapping procedure, where the computational blocks can be mapped either to fine-grained distributed logic, DSP datapaths or embedded RAMs, there are three mapping scenarios, which need to be considered for timing assignment: (i) memory access followed by another memory access, (ii) memory access followed by logic/DSP operation, and (iii) logic/DSP operation followed by memory access. Figure 3.7 shows these scenarios. Embedded RAMs in FPGA are typically synchronous, which requires latching address at a clock edge. The output data from memory can optionally be latched. The Trimatrix embedded memory blocks in Altera Stratix family of devices follow such a timing strategy [11]. From the DFG representation of the packed nodes, we derive a multicycle timing assignment, as described in Algorithm 1, which tries to optimize the total latency of an application. The algorithm considers a step size, which represents the timing increment on clock period while searching the optimal clock cycle. It depends on how fine-grained division can be done on the logic/DSP operations. The algorithm assigns a clock boundary to each memory operation. It starts with a minimum clock period (T) determined by the access time of maximum memory block and then increments T by the step size up to a limit (determined by maximum logic delay between two memory operations or one memory and primary input/output). At each step, it tries to balance the latency of each stages in a multicycle timing assignment. The best clock cycle and timing assignment are chosen through this process.

3.1.5 Energy-Efficient Configuration of RAM Blocks

In this section, the packing stage is elaborated in more details. An important part of the packing is to derive optimal energy configuration of the EMBs for all memory blocks instantiated in any application. For energy-efficient mapping of RAM blocks in an FPGA, there are mainly two algorithms which the designer can target [13]:

Table 3.2: Energy improvements with the proposed mapping approach compared to mapping in logic and DSP blocks for Stratix II, Stratix III and Stratix IV FPGA families

Input bitwidth	Stratix II		Stratix III		Stratix IV	
	% impr. logic	% impr. DSP	% impr. logic	% impr. DSP	% impr. logic	% impr. DSP
4	40.59	56.30	16.11	63.29	11.72	52.88
5	36.82	30.86	51.91	64.61	51.68	56.11
6	9.67	-16.85	28.28	43.14	46.01	39.90
7	-40.66	-139.38	-21.95	-15.78	17.15	-26.38
8	*	*	-28.18	-65.07	-6.36	-89.85

*cannot place due to lack of memory resources

- The dynamic power of a SRAM EMB is dominated by dynamic power due to bitline precharging and the designer can use a clock enable signal which can be connected to the clk enable input port of the memory as shown in Fig.

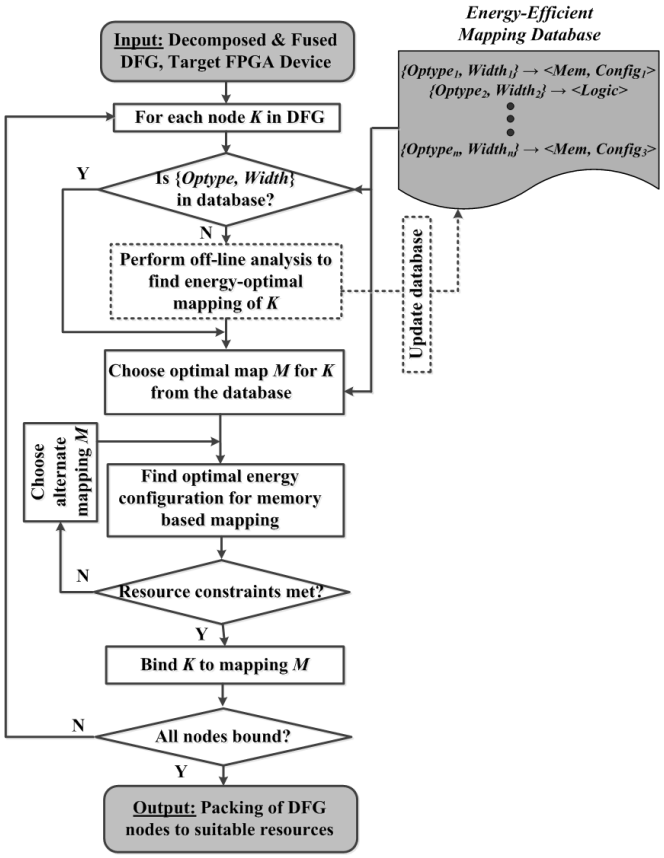

Figure 3.3: Packing algorithm for energy-efficient mapping in a FPGA device.

Table 3.3: Comparison of operational latency among the three mapping approaches for Stratix II, Stratix III and Stratix IV FPGA families

Input bitwidth	Stratix II			Stratix III			Stratix IV		
	Memory (ns)	Logic (ns)	DSP (ns)	Memory (ns)	Logic (ns)	DSP (ns)	Memory (ns)	Logic (ns)	DSP (ns)
4	1.8	3.3	3.6	1.6	3.4	2.8	1.7	3.1	2.2
5	1.8	3.7	3.6	1.6	3.7	2.8	1.7	3.6	2.2
6	1.8	4.0	3.6	1.6	3.8	2.8	1.7	3.7	2.2
7	1.8	4.1	3.6	1.6	4.6	2.8	1.7	4.1	2.2
8	2.1	4.2	3.6	2.8	4.8	2.8	2.8	4.2	2.2

3.4(a), thus avoiding the precharging energy of the unused memory blocks. Such optimizations can be done for a large logical RAM placed in a design and can switch off a large memory block as a whole. Typically such optimizations are

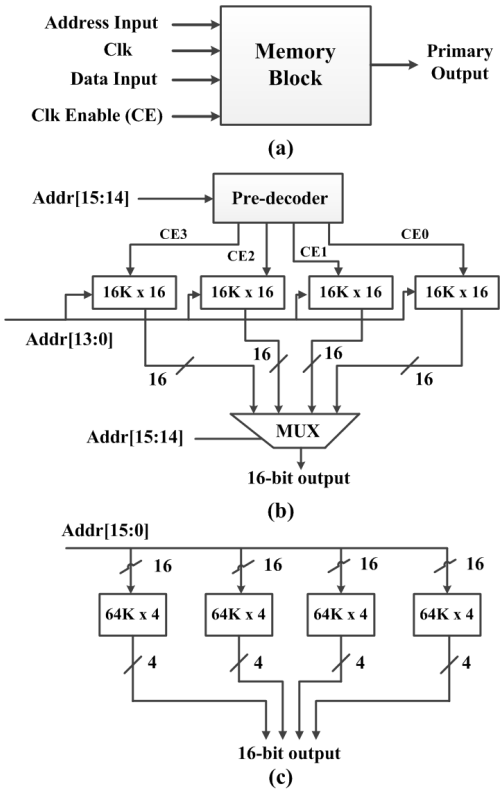

Figure 3.4: (a) Trimatrix memory block with one address, one clk and one combined clk enable port; (b) Implementation of a large memory block (16K x 16) using smaller RAM blocks with additional pre-decoding to save access energy; (c) Alternative implementation of the same memory, which incurs more access energy.

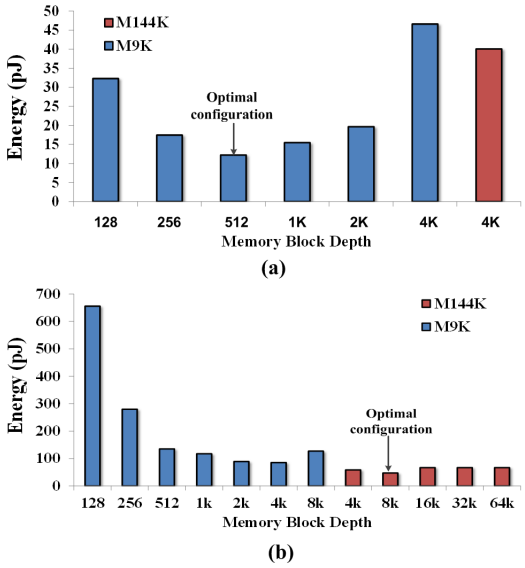

Figure 3.5: Variation in energy consumption of a memory block with varying memory type and block depth for: a) 12x12 memory; and b) 16x16 memory.

extremely helpful in applications where a particular RAM block in accessed once in many cycles.

- A single large memory block is broken into multiple small sub-banks with necessary additional pre-decoding so as to access only a single or a few smaller sub-banks for a particular memory access, keeping the unaccessed memory blocks to be off. In general an embedded memory block provided in Stratix family can be configured for a number of memory block depth and width configurations. For

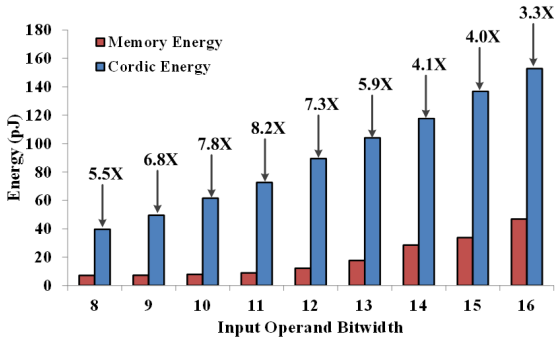

Figure 3.6: Variation in energy consumption with varying input resolution for CORDIC algorithm.

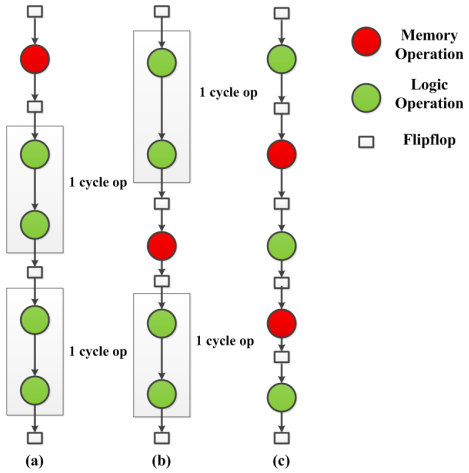

Figure 3.7: Interfacing and timing strategies for three different cases: (a) memory access followed by logic; (b) logic followed by memory access; and (c) memory access followed by another memory access.

example an m9k block available in Stratix IV can be configured as a 8192×1, 4096×2, 2048×4, 1024×8, 1024×9, 512×16, 512×18, 256×32 and 256×36. However this strategy introduces some additional logic and designers have to find a minima for energy consumption for different sizes of memory as larger number of sub-banks result in significant amount of routing energy of the pre-decoding logic. Fig. 3.4(b) and Fig. 3.4 (c) show two possible implementations of a $64k \times 16$ embedded RAM block. The implementation shown in Fig. 3.4(b) decomposes the $64k \times 16$ logical RAM block into 4 physical RAM blocks, each of output size 16. As a result, for each memory access, only one of the memory blocks are precharged and the rest remains off which results in significant amount of energy-efficiency. In Fig. 3.4(c), the physical RAM blocks chosen have an output size of 4 each and as a result for each memory access, all the blocks need to be turned on and hence requires much higher precharge energy. An energy optimal mapping result for a $4k \times 12$ memory block is shown in Fig. 3.5(a) and that for a $64k \times 16$ memory block is shown in Fig. 3.5(b) by varying the memory block depth and type of memory used.

3.1.6 Mapping Complex Datapath in Memory

For smaller bitwidths, datapaths with more number of combinational levels have significantly better energy consumption while mapping in memory compared to normal logic or DSP based mapping. The datapath for two constant coefficient multiplications followed by an addition (constmult add) have been optimally mapped in memory across different technology generations like Stratix II, Stratix III and Stratix IV and compared with the corresponding mapping results in logic and DSP blocks in the same technology nodes. Figure 3.9 shows the energy of the respective mapping schemes and Table 3.2 shows the corresponding energy improvement compared to mapping only in logic and DSP. The primary input for all the three implementation have been flopped and so also the primary output. The clk period for the respective implementations is the minimum clk period which the design can meet from the primary input flop to the primary output flop. For Stratix IV, energy consumption improvement is obtained up to 6-bit input resolution compared DSP based implementation and up to 7-bit compared to logic based implementation as shown in Table 3.2. Latency results for the different implementations across different technology generations are shown in Table 3.3. All implementations in memory, logic and DSP have been done with area optimal synthesis constraint with special emphasis on reducing routing energy for conventional logic based implementation.

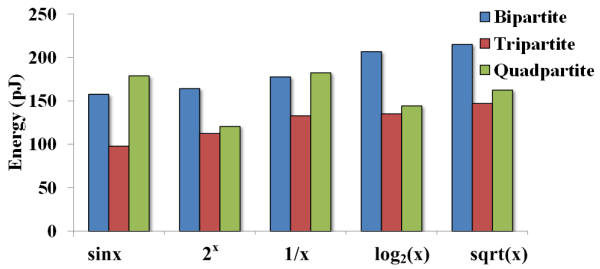

Figure 3.8: Energy trends with bipartite, tripartite and quadpartite decompositions for different transcendental functions.

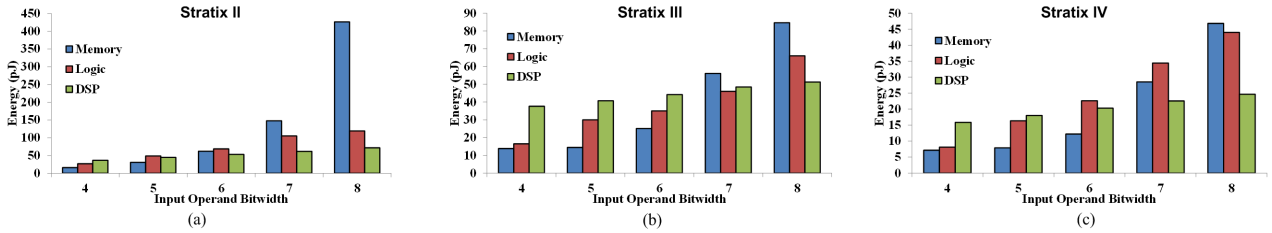

Figure 3.9: Comparison of Energy with varying input resolution for constmultadd in Altera (a) Stratix II, (b) Stratix III and (c) Stratix IV series of devices.

3.1.7 Mapping Complex Functions in Memory

Conventionally complex transcendental functions are computed in a FPGA using CORDIC based approaches where the latency of the computation increases linearly with the increase in the resolution of the input operands. On the other hand a LUT storing the output response of a function in a single lookup table gives tremendous improvement in terms of energy consumption and latency. In a Stratix IV FPGA, the maximum input size of a lookup table permissible is 16 bits. As a result, upto 16-bit resolution, any function can be mapped in a single lookup table. Figure 3.6 shows the improvement in energy consumption with respect to normal CORDIC based uni-variable function computation in Stratix IV. The energy improvement is minimum (3.3X) in case of 16-bit operands and maximum (8.2X) in case of 11-bit input operands. However, if the size of the input bitwidth does not match with the size of the lookup, then efficient decomposition techniques like multipartite and ATA (addition-table lookup-addition) have to be employed in order to minimize the memory requirement at the cost of some increase in latency. These decomposition techniques have been optimized so as to achieve energy-efficient mapping of complex transcendental functions. Figure 3.8 shows the optimization results for energy consumption in Stratix IV for bipartite, tripartite and quadpartite decompositions [9] of transcendental functions for 24 bit input fixed points. In general tripartite decompositions give the most energy optimal mapping for 24 bit input resolution uni-variable functions as shown in Fig. 3.8. Further decompositions have been avoided because

not all 24-bit input transcendental functions can be efficiently decomposed into more than 4 lookup operations.

Chapter 4

Application Mapping Results

This chapter summarizes the application mapping results of several real DSP / multimedia applications using the proposed approach and compares the energy savings achieved compared to normal logic based / DSP based implementation in Stratix IV device. In this chapter we evaluate the effectiveness of the proposed mapping approach in improving energy consumption for several common applications, which require complex datapaths and/or complex functions. We consider a commercial FPGA platform from Altera (Stratix IV) and compare with alternative implementations using both logic and DSP elements. For applications requiring no complex datapath, comparisons are made only with a logic based implementation. We have considered three common complex datapath dominated applications, namely finite impulse response (FIR) filtering, coherence calculation in a cluster, and discrete wavelet transform (DWT). We have also considered two complex function dominated applications, namely 3rd Order Polynomial Evaluation and Solution to Schrodinger Equation.

For all simulations we have used Altera Quartus II 11.0 tool suite. Altera's Design Space Explorer Tool in this tool suite is used to find out the energy consumption for the optimal memory configuration for varying memory sizes. For power simulations, the Quartus Powerplay analyzer tool is used with clock period set to a relaxed value,

which is satisfied by all three implementations (heterogeneous, logic and DSP). This is done to achieve iso-delay energy comparison between alternative mapping approaches. We used a vector set of 1000 vectors with uniform random input at each clock cycle to generate internal node activity. The same vector set is used for all three mapping scenarios. Energy comparison is made with respect to the compute energy, which is derived from the core dynamic power excluding IO and clock power, as reported by the power analyzer. In all three implementations, the mapping of logic functions is optimized for power under iso-delay target using the Quartus Design Space Explorer tool. For logic-based implementation, we have considered the constant coefficients as programmable input and hence not optimized the logic for a specific constant value. Many DSP applications (e.g. filtering) require the coefficients to be programmable. It also helps to make a fair comparison of energy through uniform mapping to memory and DSP, which provide no scope of such optimization.

Note that the FPGA synthesis tools typically provides an option for *logic to memory mapping* during physical synthesis optimizations. However, this built-in option is provided for better area optimization instead of energy. It tries to map some of the sequential elements in a design to small memory blocks (e.g. MLAB in Stratix IV) in order to reduce area or resource requirements. It does not consider mapping multi-output complex datapath/functions to large embedded memory arrays (e.g. M144K block in Stratix IV). Hence, such an option does not affect the energy behavior of the applications considered here.

4.1 8-tap FIR Filter

Due to the speed ups achieved in complex datapaths like constmult add, memory based mapping of FIR filter gives significant advantage over conventional FPGA based mapping [18]. The energy trends for an 8-tap FIR filter is shown in Fig. 4.1(a) with

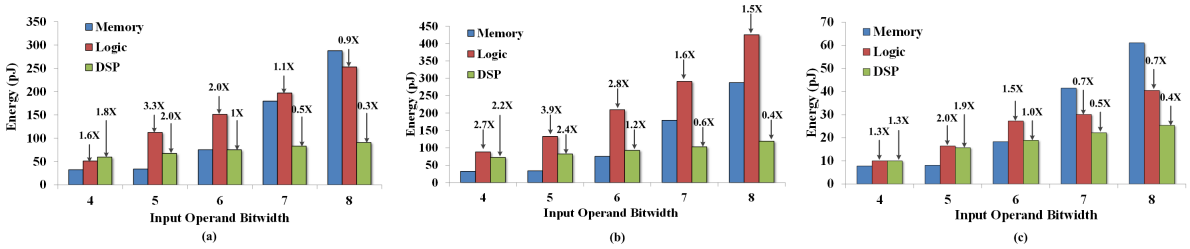

Figure 4.1: Variation in energy with varying input resolution for (a) 8-tap FIR filter; (b) coherence calculation; and (c) calculation of approximation coefficient in DWT.

variation in input bitwidth in Stratix IV. For a heterogeneous mapping of memory and logic, significant energy improvement is achieved up to 7 bit inputs (ranging from 37.5% improvement for 4 bit input to 9.1% improvement for 7 bit input) when compared to logic only implementation and up to 6 bit input resolution (ranging from 44.4% improvement at 4 bit input resolution to 0.08% improvement at 6 bit input resolution) when compared to a heterogeneous implementation of logic and DSP blocks.The implementation for all the complex datapath have been done in a similar fashion. The primary inputs and the primary outputs are flopped, however there are no flops in between the datapath. So essentially, it is a single cycle implementation, assuming that all the data are already available at the tap inputs. The cycle time of the clk is essentially the latency of the application. The resource usage for the 8-tap filter is shown below in Table 4.1 for the logic based mapping, DSP based mapping and memory based mapping process with variation in input bitwidth.

4.2 Coherence Calculation in a Cluster

During the iterative procedure of k-means clustering, the coherence in a cluster has to be calculated in order to measure the quality of clustering [19]. First the absolute distance between the different points allocated to a particular cluster from its centroid is computed and then the distances for all the points from their corresponding cluster centroid are squared and added which define the coherence of the cluster. Memory

Table 4.1: Comparison of Resource Usage for 3 complex datapath applications

Application		Logic based mapping	DSP based mapping		Memory based mapping	
	Input Bitwidth	Comb. ALUTs	DSP Blocks	Comb. ALUTs	Memory (KB)	Comb. ALUTs
FIR	4	192	8	8	1.02	48
	5	276	8	10	5.12	58
	6	344	8	12	24.57	220
	7	500	8	14	114.68	78
	8	584	8	16	524.28	272
Coherence Calculation	4	296	8	48	1.02	48
	5	428	8	58	5.12	58
	6	584	8	67	24.57	220
	7	764	8	78	114.68	78
	8	968	8	88	524.28	272
DWT	4	46	0	46	0.28	12
	5	66	2	12	1.40	14
	6	82	2	14	6.56	56
	7	120	2	16	30.72	18
	8	140	2	18	139.26	68

based computation in coherence calculation is extremely helpful and gives significant advantage till 6 bit input over a DSP based implementation (ranging from 54.5% improvement for 4 bit input resolution to 16.7% improvement for 6 bit input resolution) and till 8 bit input resolution over a logic only based implementation (ranging from 63.0% improvement for 4 bit input resolution to 33.3% improvement for 8 bit input resolution). The advantages in energy over different bitwidth of inputs are shown in Fig. 4.1(b) for a Stratix IV platform. The resource usage for different input bitwidths for coherence calculation are shown in Table 4.1 for different mapping methodologies.

4.3 Calculation of Approximation Coefficient in DWT

The datapath for the approximation computation in a DWT consists of 2 constant coefficient multiplications of the odd samples followed by one addition [20]. After a

single level of truncation, the intermediate output is added to the even sample of the current level. The first level of multiplications and the first addition is mapped in memory whereas the rest of the datapath is mapped spatially in FPGA logic. The energy results with the variations in input operands bitwidth is shown in Fig. 4.1(c) and compared with that of logic and DSP in Stratix IV. Energy improvements are obtained till 6 bit inputs compared to logic only implementation (ranging from 23.1% improvement for 4 bit input resolution to 33.3% improvement for 6 bit input resolution). Similarly improvements are obtained till 6 bit input resolution for a heterogeneous implementation of logic and DSP (ranging from 23.1% for 4 bit input resolution to a marginal 0.09% improvement for 6 bit input resolution). The resource usage for different input bitwidths are shown in Table 4.1 for different mapping methodologies.

4.4 3rd Order Polynomial Evaluation

Newton Raphson's (NR) method is widely used for the evaluation of roots for polynomial functions. Consider a 3rd order polynomial given by [21]

$$f(x) = a_3x^3 + a_2x^2 + a_1x + a_0 \tag{4.1}$$

NR method employs the following iteration formula in order to move closer to the actual root:

$$X_{n+1} = X_n - \frac{f(x)}{f'(x)} = X_n - \frac{a_3X_n^3 + a_2X_n^2 + a_1X_n + a_0}{3a_3X_n^2 + 2a_2X_n + a_1} \tag{4.2}$$

For a 24 bit input fixed point, implementing the Addition Table Addition (ATA) method in the proposed framework takes 106 KB of memory through [10]. The ATA method takes an average energy consumption of 0.2 nJ with a latency of 8.2 ns, implemented in a single cycle. For the DSP based implementation, the divider macro

is employed which is available in the Quartus Altera suite in Stratix IV. The divider hardware customarily flops the input, and the rest of the division is done in a single cycle, that is the primary output of the division is also flopped. The rest of the logic is optimized and pipelined so as to meet the cycle time of the design. A cycle period of 5ns is employed in the design with a 11 cycle latency. So the DSP based computation takes a latency of 55ns and consumes 1 nJ of energy. The critical path is provided by the divider. Logic based implementation takes similar latency of 55 ns but consumes 2.4 nJ of energy. The logic based implementation is achieved by converting all the half DSP blocks used in the DSP based implementation into corresponding logic, however keeping the pipelining in the circuit the same. The energy improvements achieved with respect to conventional mapping in FPGA is shown in Table 4.2. The better result in between logic based and DSP based implementation is taken as the conventional implementation in FPGA for all the applications shown in Table 4.2.

4.5 Solution to Schrodinger Equation (1-D)

A very common scientific application dominated by transcendental functions is finding the solution of a time-independent Schrodinger wave equation for arbitrary periodic potentials. For single dimension, the function is given by [37]:

$$\psi_n(x) = \sqrt{\frac{2}{L}} sin(\frac{n\pi x}{L}) \tag{4.3}$$

An FPGA based evaluation of this function using the proposed mapping approach for 24-bit fixed-point input operand requires 193.96 KB. The decomposition of the complex functions involved in the application have been done using the energy efficient tripartite decompositions as obtained in Chapter 3. The memory based implementation has a latency of 25.8 ns with a 3 cycle latency and an average energy consumption of 0.7 nJ, where individual tripartite decompositions have been implemented in a sin-

Table 4.2: Comparison of Energy, Latency and EDP for Several common Applications

Application	Conventional Mapping			Proposed Mapping				
	Latency (ns)	Energy (nJ)	EDP ($aJ-s$)	Latency (ns)	Energy (nJ)	EDP ($aJ-s$)	Memory (KB)	% impr Energy
8-tap FIR (6-bit)	6.4	0.15	0.98	5.6	0.08	0.42	24.6	50.2
Coherence (6-bit)	8.3	0.21	1.7	5.6	0.08	0.42	24.6	64.1
DWT (6-bit)	4.6	0.027	0.13	4.4	0.018	0.08	6.1	33.0
3rd order poly. eval.	55.0	1.0	55.0	8.2	0.2	1.6	106.0	79.1
Schrodinger eqn.	308.7	5.9	1821.3	25.8	0.7	18.1	193.9	88.3

gle cycle. Conventional FPGA has a latency of 308.7 ns in logic-only implementation whereas DSP-based heterogeneous implementation is not possible for the datapath to be mapped. The conventional logic based implementation involves computing all the transcendental functions in CORDIC, that is consuming 24 cycles for individual 24-bit transcendental functions. The latency of the application is 49 cycles with the critical path determined by the 24 bit multiplication involved in several parts of the application. The logic based framework has an average energy consumption of 5.9 nJ. Due the increased latency of CORDIC based computation, the proposed framework has significant improvement in energy over conventional FPGA based mapping as shown in Table 4.2. The mapping time for the different applications are as shown in Table 4.3 for different mapping methodologies.

aJ-s= atto J-s

Through the investigations done in this chapter, we can evaluate the effectiveness of such a heterogeneous application mapping procedure at the application level over different bitwidths of input operands. Such improvements at lower bitwidths can be exploited in a dynamic framework to achieve energy efficiency at approximate computing scenarios with judicious strategies for achieving graceful degradation of

Table 4.3: Mapping Time in seconds for Applications Using Different mapping approaches for Stratix IV using Quartus 11.0

Applications	Input Bitwidth	Heterogeneous Mapping(secs)	Logic based Mapping(secs)	DSP based Mapping(secs)
FIR	8	140	146	143
Coherence Calculation	8	140	88	79
DWT	8	81	80	75
3rd order poly. eval.	24	90	97	87
Schrodinger eqn.	24	95	85	78

output quality which is the focus of the next chapter.

Chapter 5

Energy Accuracy Tradeoff

As demonstrated in the previous chapters, heterogeneous mapping can provide significant improvement in energy consumption for specific datapath bitwidth in case of applications dominated by complex datapaths or functions. As a result such a heterogeneous mapping methodology can be utilized to trade-off energy requirement and accuracy of computations for DSP/multimedia applications both statically and dynamically. Such a trade-off will allow FPGA based embedded systems in handheld and portable devices to run for longer times at lower battery levels with acceptable performance degradation at the output. The trade-off can be achieved by reducing operand bitwidth through truncation such that the memory requirement in mapping the complex datapaths/functions can be exponentially reduced leading to large saving in energy, as shown in Fig. 3.9. Clearly, operand truncation would have impact on the output quality. However, judicious choice of bit values assigned at truncated bit position and a non-uniform truncation approach that aggressively truncates inputs of only the less critical components can lead to modest impact on output.

DSP applications can be computed using two forms of arithmetic (a) *Conventional Arithmetic* (b) *Distributed Arithmetic*. The chief difference between both forms of arithmetic is in the multiplication stage. In conventional arithmetic coarse grained

multiplication is done in between the individual inputs and the constants whereas in the case of distributed arithmetic, bits at the same positions of multiple input instances are multiplied with the same constants. Heterogeneous computing can potentially provide improvement in energy consumption in both forms of arithmetic as illustrated in the following sections.

In most DSP/multimedia applications, the final output is interpreted by human senses. This fact gives the flexibility of producing approximate output for DSP applications to the extent of acceptable quality degradation to the human senses [25]. The scope of approximate output computation gives great opportunities to the designer for truncation of datapath in order to reduce the power/energy consumption. Earlier works have proposed the advantages of intermediate bit truncation in adders for DSP applications in order to reduce the process variation induced failures in the context of post-silicon calibration or repair approaches [27]. Standard techniques like Level Constrained Common Subexpression Elimination (LCCSE) have been used in order to optimize the number of adder stages in order to reduce the critical path of the filter for specific filter coefficients in order to avoid process variation induced delay failures[22].

The proposed methodology differs from the previous investigations in the following manner: i) previous investigations have looked into truncation of the constant coefficient multiplication output and adder outputs. In contrast we are investigating the effect of input truncation on the output performance. ii) the focus of this work is to improve the energy-accuracy trade-off in a FPGA platform using embedded memory based computing. The previous investigations have mostly focused into energy savings in DSP blocks with custom adders and multipliers. A FIR filter is implemented here using the Conventional Arithmetic (CA) and Distributed Arithmetic (DA) in the proposed work.

5.1 Energy Accuracy Trade-Off for Conventional Arithmetic

In the case of applications requiring complex datapaths e.g. FIR and DWT, results shown in previous sections clearly suggest improvements up to 6 or 7 bit input depending on the application using the proposed heterogeneous mapping approach.

5.1.1 Uniform Truncation

In this work, the effect in the primary output of the filter is studied by truncating the primary input operands or in effect by reducing the input operand resolution. During blind truncation, i.e. truncating inputs at all taps equally, the lookup tables are generated for the FIR filter by optimal value allocation at the truncated bits in order to see the impact on the output. The designed filter is essentially a low pass equiripple 32-tap FIR filter with a passband frequency of 9.6 kHz and a stopband frequency of 12 kHz. The original filter input operand resolution is 8 bits. 3 bits of the inputs, starting from the LSB have been allocated different values and the impact in the output mean square error (MSE) for such value allocation at the primary inputs is shown in Fig. 5.1. The inputs applied to the filter is an impulse input of magnitude 8′b11111111. It can be inferred that for 3 bit truncation, generating lookup tables assuming 3′b011 or 3′b100 values at the truncated positions gives minimum error at the output.

5.1.2 Preferential Truncation

Specific inputs are truncated more aggressively or less aggressively depending on the significance of the corresponding coefficients at the output.

The output impulse response magnitude of the designed 32-tap FIR filter when all the tap inputs are kept intact at 8 bit input resolution is 82.56. The variation in

Table 5.1: Resource Usage and Energy Consumption for a 32-tap FIR filter using heterogenous, logic based and DSP based mapping Using Conventional Arithmetic

Trunc. Method	Proposed mapping approach			Logic based mapping		DSP based mapping		
	Comb. ALUTs	Memory (KB)	Energy (pJ)	Comb. ALUTs	Energy (pJ)	Comb. ALUTs	DSP	Energy (pJ)
1-bit trunc	1279	458.7	637.3	2074	592.9	175	32	263.9
2-bit trunc	685	98.3	252.9	1454	483.1	159	32	215.4
3-bit trunc	242	20.5	88.1	1196	311.0	153	32	200.5
4-bit trunc	197	4.1	71.8	859	180.3	240	26	172.5
Config.1	587	138.5	246.9	1572	407.0	177	32	223.0
Config.2	279	29.5	108.6	1507	330.0	225	30	220.8
Config.3	127	6.1	63.7	1284	250.5	288	28	210.2
Config.4	1013 (1.7X*)	174.1 (1.3X)	247.0	4383 (2.8X)	407.1	410 (2.3X)	90 (2.8X)	223.1

*Times improvement in the last row (within braces) is with respect to $Config.1$.

the stopband ripple due to zeroing different tap coefficients are recorded in Fig. 5.2. The taps corresponding to the coefficients having the greatest impact in the stopband ripple is defined as the most significant tap or the corresponding coefficients are called the most significant coefficients. Similarly the next level important coefficients are called the 1st order coefficients and so on. Lesser truncation is done at the more significant taps and more aggressive truncation is done at the lesser significant taps in order to achieve minimum accuracy penalty at iso-energy consumption. As shown in Fig. 5.2, the middle most significant inputs are truncated by 1 bit, the inputs corresponding to the 1st order coefficients are truncated by 2 bits, followed by 3

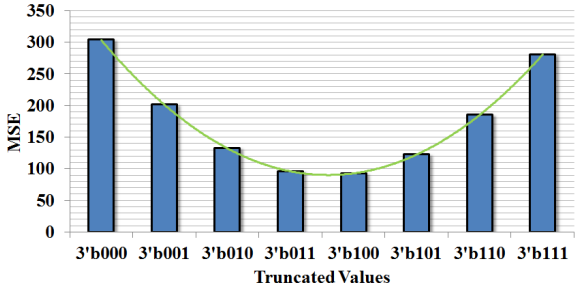

Figure 5.1: Variation of mean square error (MSE) with different bit allocation at the truncated bits for a 32-tap FIR filter.

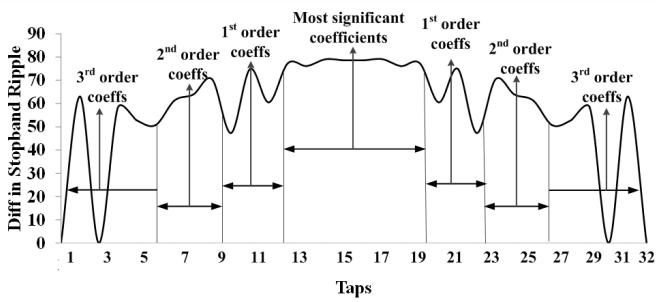

Figure 5.2: Variation of stopband ripple magnitude by zeroing different coefficients of a 32-tap FIR filter.

bits and so on ($Config.1$). In the second ($Config.2$) and third truncation modes ($Config.3$), 2 and 3 bits are truncated respectively at the middle with increasing truncation at the sides.

Figure 5.3 shows the effect in output mean square error at different energy consumption configurations due to varying levels of truncation for a 32-tap FIR filter. Zero value allocation denotes blind truncation with 1′b0 at the truncated bits. The three design points in case of zero value allocation are 1′b0, 2′b0 and 3′b0 truncations at primary inputs; i.e. truncating the inputs at the corresponding bits with zero value allocation. Similarly by allocating the optimal bits at the truncated inputs (1′b1, 2′b01, 3′b011, 4′b1000), the error due to uniform truncation through opti-

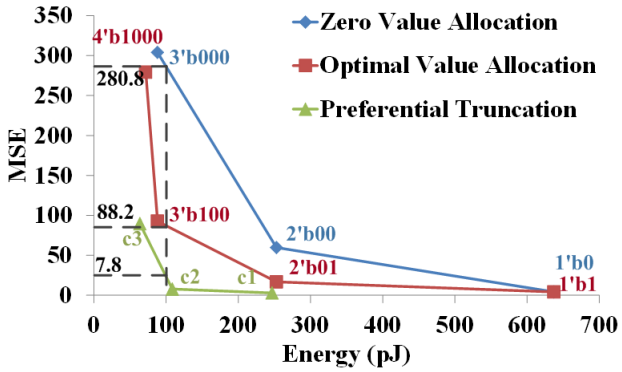

Figure 5.3: MSE versus energy consumption for a 32-tap FIR filter using heterogeneous mapping approach.

mal bit allocation is obtained at the output. Finally the 3 design points described as $Config.1$, $Config.2$ and $Config.3$ have been plotted and the mean square error (MSE) at the output is compared at similar energy consumption levels. Clearly when preferential truncation methodology is properly employed along with optimal value allocation, the energy savings is significant with respect to only optimal value allocation or any other form of blind truncation across the board. As shown in Fig. 5.3, for an energy consumption of 100 pJ, the MSE for preferential truncation is 7.8, for optimal value allocation is 88.2 and for zero value allocation is 280.8 which results in 91.1% improvement in output MSE over optimal value allocation and 97.2% improvement in output MSE over zero value allocation at iso-energy consumption. So in effect, optimal value allocation gives lesser MSE compared to blind truncation but consumes the same amount of energy. Preferential truncation on the other hand consumes lesser amount of energy consumption and also has lesser MSE compared to both blind truncation and optimal value truncation.

Heterogeneous mapping also significantly helps in reducing the resource requirements compared to conventional mapping techniques. Table 5.1 shows the resource requirements for uniform truncation (1b trunc, 2b trunc, 3b trunc and 4b trunc) and preferential truncation methodologies through heterogeneous mapping, logic based mapping and DSP based mapping techniques. For the 4 modes of uniform truncation described in Table 5.1, heterogeneous mapping provides 43.0% improvement in energy consumption on an average compared to logic based mapping. However it provides slight degradation in average energy consumption (nearly -11%) compared to DSP based implementation due to the significant difference in energy results at 7-bit input resolution. Similarly for preferential truncation, heterogeneous mapping provides 60.3% improvement in energy consumption over logic based mapping on an average and 36.6% improvement in energy consumption over DSP based mapping. Secondly at the cost of few hundred kilobytes of memory (which is around $5-10\%$ of the total

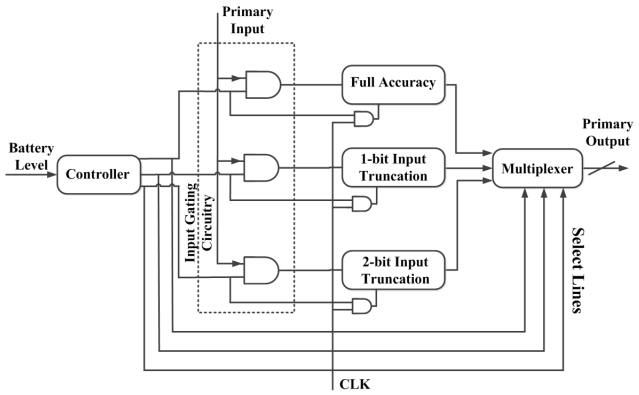

Figure 5.4: Functional block diagram of the computation unit to realize dynamic truncation for energy-accuracy trade-off.

embedded RAM available in high end FPGA devices in Stratix IV), the logic requirement or the combinational ALUT requirement goes down significantly compared to logic based mapping. So the proposed methodology can also be potentially employed in increasing the overall throughput of the device.

Configuration4 (as mentioned in Table 5.1) denotes a dynamic truncation framework which can operate in configuration1, configuration2 and configuration3 and can switch the configurations dynamically depending on the available energy budget. Figure 5.4 shows the methodology of dynamic truncation for trading off energy consumption and output quality. The overhead due to the controller and the multiplexer is negligible compared to the resource requirement of the actual mapped design as shown in Table 5.1. Due to the exponential reduction in memory space with the truncation at the inputs, the effective memory requirement for the dynamic framework is not significantly more than configuration1 which computes with minimum truncation of primary input operands. However the number of DSP blocks increases 2.3X-3X as also the logic requirements for DSP based mapping or purely logic based mapping respectively.

5.1.3 Energy Accuracy Trade-Off for Distributed Arithmetic

Distributed Arithmetic (DA) along with modulo-arithmetic has a very significant role in embedded DSP applications as mentioned in [26]. DA specifically targets the sum of products (SOP) computation that is common in most DSP applications. The arithmetic sum of products that defines the response of a linear time-invariant network can be expressed as

$$y(n) = \sum_{k=1}^{K} A_k * x_k(n) \tag{5.1}$$

where $y(n)$ is the response of the filter at time n, $x_k(n)$ is the kth input variable at time n, and A_k is the weighting factor of the kth input variable that is constant for all n. The variable x_k can be expressed in 2's complement fractional form as

$$x_k = -x_{k0} + \sum_{b=1}^{B-1} x_{kb} 2^{-b} \tag{5.2}$$

where x_{kb} is a binary variable and can assume only values of 0 and 1. The derivation can be extended to show that

$$y = \sum_{k=1}^{K} A_k[-x_{k0} + \sum_{b=1}^{B-1} x_{kb} * 2^{-b}] \tag{5.3}$$

Expanding equation 5.3 we get

$$\begin{aligned} y \quad = \quad & -[x_{10} \quad * A_1 + x_{20} * A_2 + x_{30} * A_3 + ... + x_{k0} * A_K] \\ & + [x_{11} \quad * A_1 + x_{21} \quad * A_2 + x_{31} \quad * A_3 + ... + x_{k1} \quad * A_K] 2^{-1} \\ & + [x_{12} \quad * A_1 + x_{22} \quad * A_2 + x_{32} \quad * A_3 + ... + x_{k2} \quad * A_K] 2^{-2} \\ & \qquad\qquad\qquad\qquad ... \\ & + [x_{1(B-1)} * A_1 \quad + x_{2(B-1)} * A_2 \quad + ... + x_{K(B-1)} * A_K] 2^{-(B-1)} \end{aligned} \tag{5.4}$$

Compute terms in brackets in equation 5.4 are computed in memory ALUTs or embedded memory lookups and is termed as the Distributed Arithmetic look-up table (DALUT). The rest of the additions are computed in combinational logic. In case of mapping filtering applications using DA based approach in the proposed heterogeneous mapping framework, the DALUT operation for all the inputs are done in memory and the final addition is done in logic to get the final output.

5.1.3.1 Uniform Truncation

While computing with memory, instead of decreasing memory size for each and every input in case of conventional arithmetic, the number of memory blocks gets diminished by 1 with each bit truncation at the primary input. This decreases the total latency of the circuit and the decrease in energy consumption with variation in input bitwidth also occurs at almost a linear rate for DA based approach compared to exponential energy reduction for CA based approach.

Zero Value Allocation: Uniform or blind truncation across the board is the simplest methodology of truncating inputs at the cost of graceful degradation in PSNR. For an input bitwidth of 8, and a memory wordsize of 16, for truncating one input bit at the primary input, the memory requirement for a 32-tap FIR filter goes down from 16KB to 8KB. Further truncation can be done at the primary input till 7 bits so as to have a meaningful output.

Optimal Value Allocation: Initially the LUT contents are generated assuming the truncated bits to be zero. However as in the case of conventional arithmetic, computing with distributed arithmetic also has similar observations about optimal bit allocation in order to reduce the MSE at the output.

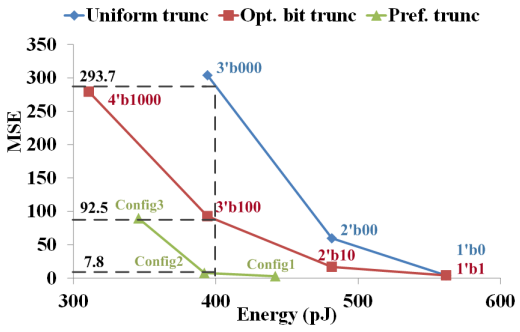

Figure 5.5: Variation in output MSE versus energy consumption for a DA based 32-tap FIR filter using heterogeneous mapping.

5.1.3.2 Preferential Truncation

The preferential truncation is implemented in exactly the same way as has been proposed for conventional arithmetic. For higher input resolution, preferential truncation methodology can be even more effective in applications like CA and DA based filtering.

As shown in Fig. 5.5, at iso-energy consumption of 400pJ, the output MSE through zero value allocation is 293.7, through optimal bit allocation is 92.5 and through the proposed preferential truncation technique is 7.75 as shown in Fig. 5.5. Thus clearly the preferential truncation strategy gives significantly less output error at iso-energy or conversely consumes significantly less energy at the iso-output quality.

5.1.4 Comparison between CA and DA based Implementation

Both uniform and preferential truncation methodologies have been proposed in this work for conventional as well as distributed arithmetic. The memory requirement for heterogeneous mapping is much less for DA compared to CA based implementation. As a a result, memory based computing using DA is more scalable compared

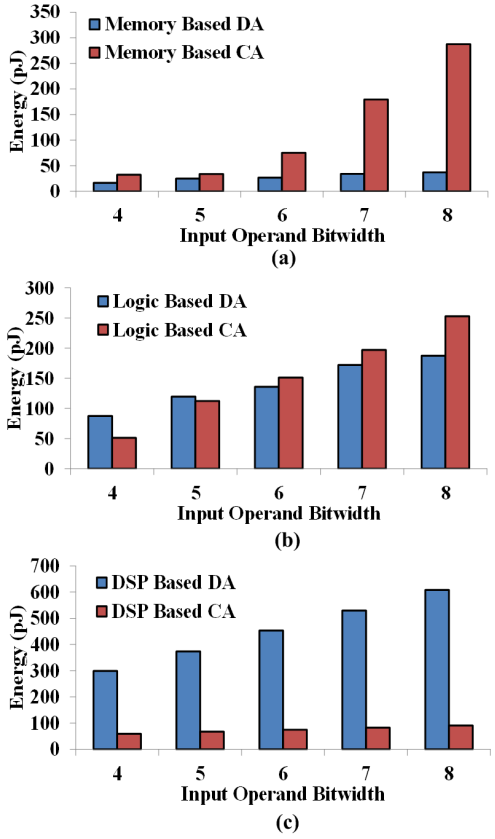

Figure 5.6: Comparison of energy consumption for a CA vs DA based implementation of FIR filter for: (a) memory based; (b) logic based; and (c) DSP based implementation.

to its CA-based counterpart. Figure 5.6 shows that when computing in a heterogeneous framework, the energy consumption increases exponentially with input operand bitwidth in conventional arithmetic unlike in distributed arithmetic where the energy consumption increases linearly. As a result the energy consumption for DA based computation is much less compared to CA based computation. For logic based implementation, at smaller input resolution the multiplier size requirement is smaller in CA compared to DA and a result, the energy consumption in CA is lesser. However with the increase in input resolution, DA based computation is more energy-efficient as shown in Fig. 5.6. For DSP based computation, due to the fine-grained nature of the computation, the number of DSP blocks in DA is more compared to CA and as result the power/energy consumption is much more significant in DA. In sum-

mary the proposed methodology gives exponential savings in energy for conventional arithmetic and linear savings in energy for distributed arithmetic with input bitwidth truncation.

5.1.5 Integration with Existing EMB-based Mapping Approaches

The proposed heterogeneous mapping approach, which focuses on efficient mapping of coarse-grained functions to EMBs, can be integrated with existing fine-grained approaches of mapping random logic to EMBs [14], [16]. Such an unified approach can maximize the energy saving. In the fine-grained mapping approach, the random logic functions or applications that require fine-grained control of data can benefit from memory based mapping. Hence, the proposed approach is complementary to the existing fine-grained mapping approaches. However, for seamless integration of both approaches in a unified mapping framework, we need to carefully adjust the timing of memory and logic operations. A multicycle timing approach as described in Algorithm 1 can be extended for use in such a framework.

Chapter 6

Conclusion and Future Work

We have presented an application mapping process for FPGA that exploits the embedded memory blocks for computation to minimize the overhead of programmable interconnects. We show that the proposed mapping process can significantly improve energy efficiency for many applications, which require complex datapath, complex functions or both. The software architecture to perform functional decomposition, fusion of operations, packing of individual coarse-grained operations and timing assignment is presented in this thesis. The effectiveness of the proposed mapping technique has been validated with a commercial FPGA platform, namely Altera Stratix IV FPGA. Finally, computation in embedded memory in FPGA is extended to dynamically trade-off energy versus accuracy through use of judicious bitwidth truncation at run time for both conventional as well as distributed arithmetic based implementations. To minimize the impact on output quality, we have presented a preferential truncation methodology that exploits the nature of DSP applications to minimize impact on output quality with operand truncation. With continued scaling of memory and emergence of novel high-density memory technologies - both volatile and non-volatile - FPGA devices are expected to integrate more memory with improved performance, which can significantly benefit the proposed approach. Future work

will include extending the framework to emerging FPGA platforms, other domains of applications, support for different precision and data formats, and more efficient decomposition/fusion methodologies.

Bibliography

[1] "The landscape of parallel computing research: A view from Berkeley [Online]" *http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html.*

[2] "Energy Efficient Hardware-Software Co-Synthesis Using Reconfigurable Hardware," *Chapman and Hall, CRC Computer and Information Science Series.*

[3] T. Good and M. Benaissa, "AES on FPGA from thr Fastest to the Smallest," *Cryptographic Hardware and Embedded Systems*, 2005.

[4] A. Rahman, S. Das, A. P. Chandrakasan and R. Reif, "Wiring Requirement and Three-Dimensional Integration Technology for Field Programmable Gate Arrays," *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 11, No. 1, 2003.

[5] "FPGA Architecture [Online]" *http://www.altera.com/literature/wp/wp-01003.pdf*

[6] "FPGA Logic Cells Comparison [Online]" *http://www.1-core.com/library/digital/fpga-logic-cells/*

[7] W.B. Ligon III, G. Monn, D. Stanzione, F. Stivers and K.D. Underwood, "Implementation and Analysis of Numerical Components for Reconfigurable Computing," *Aerospace Applications Conference*,1999.

[8] C. Brunelli, H. Berg and D. Guevorkian, "Approximating sine functions Using Variable Precision Taylor Polynomials," *IEEE Workshop on Signal Processing Systems*, 2009.

[9] F. D. Dinechin and A. Tisserand, "Multipartite Table Methods," *IEEE Transactions on Computers*, Vol. 54, No. 3, 2005.

[10] W.F. Wong and E. Goto, "Fast Evaluation of Elementary Functions in Single Precision," *IEEE Transactions on Computers*, Vol. 44, No. 3, 1995.

[11] "Stratix FPGA: Low Power, High Performance [Online]" *http://www.altera.com/devices/fpga/stratix-fpgas/stratix/stratix/stx-index.jsp*

[12] "Xilinx Virtex series of FPGAs [Online]" *http://www.xilinx.com/products/index.htm*

[13] R. Tessier, V. Betz, D. Neto, A. Egier and T. Gopalsamy, "Power-Efficient RAM Mapping Algorithms for FPGA Embedded Memory Blocks," *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, Vol.26, No. 2, 2007.

[14] J. Cong and S. Xu, "Technology Mapping for FPGAs with Embedded Memory Blocks," *International Symposium on Field Programmable Gate Arrays*, 1998.

[15] A. Ghosh, S. Paul and S. Bhunia, "Energy Efficient Application Mapping in FPGA through Computation in Embedded Memory Blocks," *VLSI Design*, 2012.

[16] S. Wilton, "SMAP: Heterogeneous Technology Mapping for Area Reduction in FPGAs with Embedded Memory Arrays," *International Symposium on Field Programmable Gate Arrays*, 1998.

[17] S. Chin, C. Lee and S. Wilton, "Power Implications of Implementing Logic Using FPGA Embedded Memory Arrays," *Field Programmable Logic and Applications*, 2006.

[18] N. Sankarayya, K. Roy and D. Bhattacharya, "Algorithms for low power and high speed FIR filter realization using differential coefficients" *IEEE Transactions on Circuits and Systems*, Vol. 44, No. 6, 1997.

[19] C. Ding and X. He, "K-means Clustering via Principal Component Analysis," *International Conference on Machine Learning*, 2004.

[20] S. Narasimhan, H.J. Chiel and S. Bhunia, "Ultra-Low Power and Robust Digital Signal Processing Hardware for Implantable Neural Interface Microsystems," *IEEE Transactions on Biomedical Circuits and Systems*,Vol. 5, No. 2, 2011.

[21] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery, "Fortran Numerical Recipes," *Cambridge University Press*, 1992.

[22] N. Banerjee, J.H. Choi and K. Roy, "A Process Variation Aware Low Power Synthesis Methodology for fixed point FIR filters," *International Symposium on Low Power Electronics and Design*, 2007.

[23] "Achieving low Power in 65-nm Cyclone-III FPGAs," *Altera White Paper*, 2007.

[24] F. Lee, D. Chen, L. He and J. Cong, "Architecture Evaluation for Power-Efficient FPGAs," *International Symposium on Field Programmable Gate Arrays*, 2003.

[25] V. Gupta, D. Mohapatra, S.P. Park, A. Raghunathan and K. Roy, "IMPACT: IMPrecise adders for low-power Approximate CompuTing," *International Symposium on Low Power Electronics and Design*, 2011.

[26] "The Role of Distributed Arithmetic in FPGA-Based Signal Processing" *http://www.xilinx.com/appnotes/theory1.pdf*

[27] K. Kunaparaju, S. Narasimhan and S. Bhunia, "VaROT: Methodology for Variation-Tolerant DSP Hardware Design Using Post-Silicon Truncation of Operand Width," *VLSI Design*, 2011.

[28] W.G. Osborne, J.G.F. Coutinho, W. Luk and O. Mencer, "Power-Aware and Branch-Aware Word-Length Optimization," *Field-Programmable Custom Computing Machines*, 2008.

[29] C.T. Chow, L.S.M. Tsui, P.H.W. Leong, W. Luk and S. Wilton, "Dynamic Voltage Scaling for Commercial FPGAs," *Field Programmable Technology*, 2005.

[30] M. Klein, "Power Consumption at 40 and 45 nm," *Xilinx White Paper*, 2009.

[31] J. Lamoureux and W. Luk, "An Overview of Low Power Techniques for Field Programmable Gate Arrays," *NASA/ESA Conference on Adaptive Hardware and Systems*, 2008.

[32] "Lookup table [Online]," *http://en.wikipedia.org/wiki/Lookuptable*

[33] "Soft Multipliers for DSP Applications," *Altera*.

[34] "Embedded Intel Solutions [Online]" *http://www.embeddedintel.com/news.php?article=276*

[35] R. Gutierrez, V. Torrez and J. Valls, "FPGA-implementation of atan(Y/X) based on logarithmic transformation and LUT-based techniques," *Journal of Systems Architecture*, Vol. 56, No. 11, 2010.

[36] S. Choi, R. Schrofano, V.K. Prasanna and J.W. Wang, "Energy Efficient Signal Processing Using FPGAs," *International Symposium on Field Programmable Gate Arrays*, 2003.

[37] "Time Dependent Schrodinger Equation [Online]" *http://hyperphysics.phy-astr.gsu.edu/hbase/quantum/scheq.html*