HARDWARE DESIGN AND CERTIFICATION ASPECTS OF A FIELD PROGRAMMABLE GATE ARRAY-BASED TERRAIN DATABASE INTEGRITY MONITOR FOR A SYNTHETIC VISION SYSTEM

A thesis presented to the faculty of the Fritz J. and Dolores H. Russ College of Engineering and Technology of Ohio University

In partial fulfillment of the requirement for the degree Master of Science

Anupriya Kakkeroda August 2004
This thesis entitled
HARDWARE DESIGN AND CERTIFICATION ASPECTS OF A FIELD
PROGRAMMABLE GATE ARRAY-BASED TERRAIN DATABASE INTEGRITY
MONITOR FOR A SYNTHETIC VISION SYSTEM

by

Anupriya Kakkeroda

has been approved
for the School of Electrical Engineering and Computer Science
and the Fritz J. and Dolores H. Russ College of Engineering and Technology

by

Maarten Uijt de Haag
Assistant Professor of School of Electrical Engineering and Computer Science

R. Dennis Irwin
Dean, Fritz J. and Dolores H. Russ College of Engineering and Technology
This thesis describes the concept of integrity monitoring for a safety-related application such as SVS, and the certification aspects of avionics systems in general, using a SVS as a case study. Based on the functionality as defined by a Terrain Database Integrity Monitor (TDIM) for a SVS, functional requirements are generated to be implemented in software (SW) and hardware (HW). Accordingly, HW and SW components are designed to meet these functional requirements. A Field Programmable Gate Array (FPGA) device is chosen as the HW platform. HW/SW co-design is employed to optimize the performance and increase efficiency of the design. The certification concerns associated with Complex Electronic Hardware (CEH), especially the verification and validation of integrated HW/SW avionics systems are addressed. Additional certification considerations for Programmable Logic Devices (PLDs) in avionics systems, such as usage of Commercial-Off-The-Shelf (COTS) components, Intellectual Property (IP) cores, and tool qualification are enumerated.

Approved:

Maarten Uijt de Haag
Assistant Professor of Electrical Engineering and Computer Science
Dedicated to my parents
Acknowledgments

First of all, my sincere thanks to my advisor, Dr. Maarten Uijt de Haag, for being an excellent motivator and believing in me. I’m thankful for the opportunity to have assisted him in his research- a very valuable experience that I will always cherish.

I thank all my committee members- Dr. Frank van Graas, Dr. Jeffrey Dill and Dr. Gene Kaufman, for giving their valuable time and effort to be on my thesis committee.

I want to especially thank Sanjeev G. for helping me with my thesis on so many occasions and giving me very useful tips in my thesis.

How can I forget Sharon and Louise opposite my room in AEC? Without them, I would probably not even get a copier machine to work. Thank you, you both and everyone for all your help in the smallest way, and I wish good luck to each one of you in all your endeavors!

Lastly, I want to thank my parents for all their love and support without which this thesis would not have been possible.
TABLE OF CONTENTS

Abstract........................................................................................................................................... 3
Acknowledgments .......................................................................................................................... 5
Table of Contents .......................................................................................................................... 6
List of Tables .................................................................................................................................. 9
List of Figures ............................................................................................................................... 10
List of Abbreviations .................................................................................................................. 12
Chapter 1  Outline of Thesis ......................................................................................................... 15
Chapter 2  Overview of Synthetic Vision Systems, Integrity Monitor and Certification of Avionics Systems .......................................................... 18

  2.1 Terrain Awareness and Warning Systems (TAWS) and Synthetic Vision Systems (SVS) ........................................................................................................ 18

  2.2 Integrity Monitoring ............................................................................................................. 20

  2.3 Certification of Avionics Systems ....................................................................................... 25

Chapter 3  Certification of Avionics Systems: Software and Hardware Design Assurance Guidance .................................................................................. 29

  3.1 Certification Stages of a Typical Life Cycle Process .......................................................... 29

    3.1.1 Concept Development .................................................................................................. 29

    3.1.2 Safety Assessment ....................................................................................................... 29

    3.1.3 Preliminary Design ...................................................................................................... 30

    3.1.4 Detailed Design .......................................................................................................... 30

    3.1.5 Implementation ........................................................................................................... 31

    3.1.6 Validation and Verification ......................................................................................... 31

    3.1.7 Configuration Management ....................................................................................... 31

    3.1.8 Quality Assurance ...................................................................................................... 32

  3.2 SW and HW Considerations ................................................................................................. 32

    3.2.1 SW Certification based on RTCA/DO-178B (Also EUROCAE ED-12) ............... 33

      3.2.1.1 SW Planning Process ......................................................................................... 35

      3.2.1.2 SW Development Processes ............................................................................. 35

      3.2.1.3 SW Integral Processes ....................................................................................... 35

      3.2.1.4 Additional Considerations for SW .................................................................... 36

    3.2.2 HW Certification based on RTCA/DO-254 .............................................................. 37
3.2.2.1 HW Safety Assessment ......................................................................... 38
3.2.2.2 HW Design Life Cycle Processes ......................................................... 39
  3.2.2.2.1 HW Planning Process ..................................................................... 39
  3.2.2.2.2 HW Design Processes ................................................................... 40
  3.2.2.2.3 Supporting Processes .................................................................... 42
  3.2.2.2.4 Additional Considerations ............................................................ 43

Chapter 4 Functionality and Implementation Platforms of Terrain Database
  Integrity Monitor for a SVS ........................................................................ 44
  4.1 Choice of Implementation Platform .......................................................... 44
    4.1.1 HW/SW Co-Design .......................................................................... 44
  4.2 Terrain Database .................................................................................... 49
  4.3 Input Block ........................................................................................... 52
  4.4 Computation Block ................................................................................ 53
  4.5 Output Block .......................................................................................... 58
  4.6 Choice of HW Platform for TDIM (System Requirements) ...................... 59
  4.7 About PLDs and FPGAs ........................................................................ 61

Chapter 5 Design Process for SW and HW Design Components of TDIM ...... 68
  5.1 Scope of HW Design ............................................................................... 68
  5.2 Design Process ........................................................................................ 71
  5.3 High-level Description of Algorithm ....................................................... 73
  5.4 Low-level Description of Algorithm ........................................................ 74
    5.4.1 Terrain Database .............................................................................. 74
      5.4.1.1 Direct Memory Access (DMA) of terrain database elevations ....... 76
      5.4.1.2 Modified Terrain Database ......................................................... 77
      5.4.1.3 Bi-linear Interpolation Offsets in Terrain Database ...................... 80
    5.4.2 Input Block ....................................................................................... 85
      5.4.2.1 Memory addressing scheme ....................................................... 86
      5.4.2.2 Calculation of Interpolation offsets of input latitudes and longitudes.. 89
    5.4.3 Computation Block .......................................................................... 91
      5.4.3.1 Dataflow ..................................................................................... 92
      5.4.3.2 Control Unit ............................................................................... 96
        5.4.3.2.1 Scheme to Generate Translation Offsets ............................... 97
        5.4.3.2.2 Control Logic for Generation of Translation Offsets ............. 99
      5.4.3.3 Functional Components in VHDL .............................................. 101
        5.4.3.3.1 How to describe a functional requirement in VHDL? ........... 101
        5.4.3.3.2 Hierarchical Design of HW Component for TDIM of SVS .... 104
    5.4.4 Output Block .................................................................................... 106
Chapter 6 Verification and Validation of SW/HW Integrated Systems: FPGA-based TDIM for a SVS, a Case Study

6.1 Concerns for Certification of Complex Electronic Hardware (CEH) .................................................. 107
6.2 RTCA/DO-254-Based Structured Design Approach for PLDs ..................................................... 108
6.3 Verification and Validation of PLDs ............................................................................................... 110
   6.3.1 Validation .......................................................................................................................... 110
   6.3.2 Verification ....................................................................................................................... 111
6.4 Additional Considerations ............................................................................................................. 112
   6.4.1 Tool Qualification ............................................................................................................. 112
   6.4.2 IP Cores ............................................................................................................................ 113
   6.4.3 COTS Components ......................................................................................................... 114
6.5 TAD PLD Generic Issue Paper ..................................................................................................... 115

Chapter 7 Results and Proposed RTL Model .................................................................................... 117

7.1 Simulation Details .......................................................................................................................... 117
7.2 Results ........................................................................................................................................ 119
   7.2.1 Case 1 ................................................................................................................................ 119
   7.2.2 Case 2 ................................................................................................................................ 121
   7.2.3 Case 3 ................................................................................................................................ 124
   7.2.4 Case 4 ................................................................................................................................ 125
   7.2.5 Final Results ....................................................................................................................... 126
7.3 Proposed RTL Model .................................................................................................................... 126

Chapter 8 Summary, Conclusions and Recommendations ............................................................ 131

8.1 Summary and Conclusions ......................................................................................................... 131
8.2 Recommendations for Future Research .................................................................................... 132

References .......................................................................................................................................... 135

Appendix A ....................................................................................................................................... 140
Appendix B ....................................................................................................................................... 141
Appendix C ....................................................................................................................................... 142
Appendix D ....................................................................................................................................... 144
Appendix E ....................................................................................................................................... 148
LIST OF TABLES

Table 1: Original Terrain Database and modified version................................................ 78
Table 2: Mapping a Latitude/Longitude pair into memory address ................................. 86
Table 3: Generation of control logic using arrays and indices ....................................... 100
Table 4: Typical ASIC/PLD Process mapping [12] ....................................................... 109
Table 5: Simulation Details for all Cases ....................................................................... 117
LIST OF FIGURES

- Figure 1: A typical block diagram of a typical SVS [5] ................................................... 20
- Figure 2: A simplified block diagram of a Terrain Database Integrity Monitor .............. 22
- Figure 3: Document Layout of RTCA/DO-178B ............................................................. 34
- Figure 4: Effort Vs Life Cycle Stages [15]....................................................................... 36
- Figure 5: Decomposition of FFPs ..................................................................................... 39
- Figure 6: Hardware Design Life Cycle [12] ..................................................................... 41
- Figure 7: Components of a TDIM .................................................................................... 45
- Figure 8: HW/SW co-design............................................................................................. 46
- Figure 9: Flowchart of MPDLIM functionality.................................................................. 48
- Figure 10: Terrain database elevation profile (meters) with flight path ....................... 49
- Figure 11: Resolution of terrain database (not to scale) ................................................... 51
- Figure 12: Input Block ...................................................................................................... 53
- Figure 13: Checking for linear bias in terrain ................................................................... 54
- Figure 14: T-value and threshold plots ............................................................................. 56
- Figure 15: Computation Block ........................................................................................ 57
- Figure 16: Output Block ................................................................................................... 58
- Figure 17: Logic Device Family ....................................................................................... 62
- Figure 18: Xilinx™ FPGA Structure [27] ........................................................................ 62
- Figure 19: CLB of Xilinx™ XCV3000 FPGA [28] ......................................................... 64
- Figure 20: 2-Slice Virtex© CLB ....................................................................................... 66
- Figure 21: Typical FPGA Design Process [31] ................................................................. 69
- Figure 22: Design process of TDIM HW and SW Components ...................................... 72
- Figure 23: Mapping terrain elevation profile into memory .............................................. 75
- Figure 24: Representation of the Modified Terrain Database (Not to scale) ............... 79
- Figure 25: Bi-linear interpolation ..................................................................................... 81
- Figure 26: Interpolation offsets for a memory address ..................................................... 84
- Figure 27: Design components of Computation Block ..................................................... 95
- Figure 28: Swapping Latitude and Longitude Translation Offsets .................................. 98
Figure 29: Hierarchical HW Design ................................................................. 105
Figure 30: Plots for High-Level Description of Algorithm ...................... 119
Figure 31: Terrain Database Plots for Low-Level Description ............... 121
Figure 32: Plots for Low-Level Description for Case 2 ......................... 123
Figure 33: Plots for Low-Level Description of Algorithm for Case 3 ....... 124
Figure 34: Plots for Low-Level Description of Algorithm for Case 4 ...... 125
Figure 35: Proposed RTL Model of HW component ............................... 128
Figure 36: Pipelined stages of RTL Model .................................................... 128
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC</td>
<td>Advisory Circular</td>
</tr>
<tr>
<td>AD</td>
<td>Absolute Disparity</td>
</tr>
<tr>
<td>AGL</td>
<td>Above Ground Level</td>
</tr>
<tr>
<td>arcs</td>
<td>arc seconds</td>
</tr>
<tr>
<td>ARINC</td>
<td>Aeronautical Radio Inc.</td>
</tr>
<tr>
<td>ARP</td>
<td>Aerospace Recommended Practice</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application Specific Integrated Circuit</td>
</tr>
<tr>
<td>CEH</td>
<td>Complex Electronic Hardware</td>
</tr>
<tr>
<td>CFIT</td>
<td>Controlled Flight Into Terrain</td>
</tr>
<tr>
<td>CFR</td>
<td>Code of Federal Regulations</td>
</tr>
<tr>
<td>CLB</td>
<td>Configuration Logic Block</td>
</tr>
<tr>
<td>COTS</td>
<td>Commercial-Off-The-Shelf</td>
</tr>
<tr>
<td>deg</td>
<td>degrees</td>
</tr>
<tr>
<td>DEM</td>
<td>Digital Elevation Model</td>
</tr>
<tr>
<td>DGPS</td>
<td>Differential Global Positioning System</td>
</tr>
<tr>
<td>DMA</td>
<td>Direct Memory Access</td>
</tr>
<tr>
<td>DOT</td>
<td>Department of Transportation</td>
</tr>
<tr>
<td>DPRAM</td>
<td>Dual Port Random Access Memory</td>
</tr>
<tr>
<td>DTED</td>
<td>Digital Terrain Elevation Data</td>
</tr>
<tr>
<td>DWL</td>
<td>Downward Looking</td>
</tr>
<tr>
<td>ECM</td>
<td>Electronic Component Management</td>
</tr>
<tr>
<td>EUROCAE</td>
<td>European Organization for Civil Aviation Equipment</td>
</tr>
<tr>
<td>FAA</td>
<td>Federal Aviation Administration</td>
</tr>
<tr>
<td>FAR</td>
<td>Federal Aviation Regulations</td>
</tr>
<tr>
<td>FFP</td>
<td>Functional Failure Path</td>
</tr>
<tr>
<td>FHA</td>
<td>Fault Hazard Analysis</td>
</tr>
<tr>
<td>FMS</td>
<td>Flight Management System</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>GNSS</td>
<td>Global Navigation Satellite System</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>HDD</td>
<td>Head Down Display</td>
</tr>
<tr>
<td>HDL</td>
<td>Hardware Description Language</td>
</tr>
<tr>
<td>HUD</td>
<td>Head Up Display</td>
</tr>
<tr>
<td>HW</td>
<td>Hardware</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>INS</td>
<td>Inertial Navigation System</td>
</tr>
<tr>
<td>IOB</td>
<td>Input Output Block</td>
</tr>
<tr>
<td>JAA</td>
<td>Joint Aviation Authorities</td>
</tr>
<tr>
<td>LC</td>
<td>Logic Cell</td>
</tr>
<tr>
<td>LCD</td>
<td>Liquid Crystal Display</td>
</tr>
<tr>
<td>LRU</td>
<td>Line Replaceable Unit</td>
</tr>
<tr>
<td>LUT</td>
<td>Look Up Table</td>
</tr>
<tr>
<td>m</td>
<td>meters</td>
</tr>
<tr>
<td>MPDLIM</td>
<td>Multiple Path Downward Looking terrain database</td>
</tr>
<tr>
<td>MSDAD</td>
<td>Mean Square Difference of Absolute Disparities</td>
</tr>
<tr>
<td>MSL</td>
<td>Mean Sea Level</td>
</tr>
<tr>
<td>MTI</td>
<td>Misleading Terrain Information</td>
</tr>
<tr>
<td>MUX</td>
<td>Multiplexer</td>
</tr>
<tr>
<td>PDS</td>
<td>Previously Developed Software</td>
</tr>
<tr>
<td>PHAC</td>
<td>Plan for Hardware Aspects of Certification</td>
</tr>
<tr>
<td>PROM</td>
<td>Programmable Read Only Memory</td>
</tr>
<tr>
<td>PSAC</td>
<td>Plan for Software Aspects of Certification</td>
</tr>
<tr>
<td>PSSA</td>
<td>Preliminary System Safety Assessment</td>
</tr>
<tr>
<td>RADALT</td>
<td>Radar Altimeter</td>
</tr>
<tr>
<td>RTL</td>
<td>Register Transfer Level</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>SAE</td>
<td>Society of Automotive Engineers</td>
</tr>
<tr>
<td>SRAM</td>
<td>Static Random Access Memory</td>
</tr>
<tr>
<td>SSA</td>
<td>System Safety Assessment</td>
</tr>
<tr>
<td>STC</td>
<td>Supplemental Type Certificate</td>
</tr>
<tr>
<td>SVS</td>
<td>Synthetic Vision System</td>
</tr>
<tr>
<td>SW</td>
<td>Software</td>
</tr>
<tr>
<td>TAD</td>
<td>Transport Airplane Directorate</td>
</tr>
<tr>
<td>TAWS</td>
<td>Terrain Awareness and Warning System</td>
</tr>
<tr>
<td>TC</td>
<td>Type Certificate</td>
</tr>
<tr>
<td>TDIM</td>
<td>Terrain Database Integrity Monitor</td>
</tr>
<tr>
<td>TSO</td>
<td>Technical Standard Order</td>
</tr>
<tr>
<td>US</td>
<td>United States</td>
</tr>
<tr>
<td>VHDL</td>
<td>VHSIC Hardware Description Language</td>
</tr>
<tr>
<td>VHSIC</td>
<td>Very High Speed Integrated Circuit</td>
</tr>
</tbody>
</table>
Chapter 1  Outline of Thesis

The focus of this thesis is in two areas, namely, Field Programmable Gate Array (FPGA)-based hardware (HW) design of a Synthetic Vision System (SVS) Terrain Database Integrity Monitor (TDIM) and the certification aspects of the TDIM with the designed HW component as pertinent to SVS. The HW design of the TDIM is accomplished using a combination of software (SW) and HW components, i.e. a HW/SW co-design methodology is used. The HW component is described using Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (HDL) (VHDL), which is used to model and simulate the behavior of the HW component. To improve performance of the TDIM, a pipeline-based HW component is proposed and analyzed.

The general certification aspects of safety-related avionics systems are discussed, and special attention is given to HW/SW co-design. The HW component of TDIM is used as a case study. The obstacles associated with the verification and certification processes of HW/SW integrated systems for safety-related avionics applications are enumerated. Possible solutions and current practice to deal with these problems are discussed.

Chapter 2 provides an overview of SVS and introduces the concept of terrain database integrity monitoring. It also explains the need of a TDIM for a SVS from a safety-related perspective. The certification process for avionics systems is summarized to explain the steps involved.
Chapter 3 continues with the description of the certification process, but specifically explains the SW certification and HW certification processes. These processes are explained in general, but their relevance to the SW and HW components for a SVS TDIM are pointed out.

Based on a Multiple Path Downward Looking terrain database Integrity Monitor (MPDLIM) scheme, Chapter 4 explains the detailed functionality of the SVS TDIM. It also introduces the concept of HW/SW co-design for a TDIM in a SVS and enlists the advantages of this type of design. Based on the MP-DLIM scheme, each individual functional block of the SVS TDIM is described. Accordingly, the SW and HW requirements are listed as per the HW/SW co-design methodology. The choice of FPGA as the HW design platform is justified and a background on FPGAs is provided.

Chapter 5 describes the scope of the HW design and the assumptions for the design process. It explains, in detail, the design process for the SW components and the design techniques used to enable the HW/SW co-design. The VHDL design and simulation process of the HW component for the SVS TDIM, as targeted for an FPGA device, is explained in detail.

Chapter 6 describes the verification and validation processes from the certification perspective. The significance of each of these processes for SW/HW integrated systems is enlisted, while stating the additional considerations that are required for the HW/SW co-design of an FPGA-based SVS TDIM. The design and verification processes for the FPGA-based SVS TDIM are put in perspective of the
processes required for certification and attention is drawn to the areas where the HW/SW interface is created.

Chapter 7 discusses the results of the VHDL simulation of the designed HW component and compares these simulation results with the expected results from SW. Based on the designed component, a Register Transfer Level (RTL) model is proposed and analyzed for a pipelined design.

Chapter 8 concludes the thesis and summarizes the results and observations made from each chapter. It also includes recommendations for future research.
Chapter 2  Overview of Synthetic Vision Systems, Integrity Monitor and Certification of Avionics Systems

2.1 Terrain Awareness and Warning Systems (TAWS) and Synthetic Vision Systems (SVS)

Broadly speaking, avionics systems can be classified into application-based categories such as navigation, control, display, and communication systems, or according to their level of required safety, as follows [1]:

- Safety-critical systems are systems where a system failure or error results in (at least) a specified level of loss, e.g. the landing gear of an aircraft.
- Safety-related systems are systems that are activated to prevent or minimize the effect of a failure of a safety-critical system, e.g. a ‘fasten seat belt’ sign in an aircraft.
- Non-safety-related systems are systems that perform functions that are not related to safety, e.g., an entertainment television screen in an aircraft.

Safety-related avionics systems include TAWS and collision avoidance systems, which help enhance the safety-related features of an aircraft. In recent years, Terrain Awareness and Warning Systems (TAWS) have been the focus of significant research and development in order to help increase the situational awareness of a pilot and reduce the likelihood of Controlled Flight Into Terrain (CFIT). "CFIT is an accident or incident in which the airplane, under the flight-crew's control, is inadvertently flown into terrain, obstacles or water without either sufficient or timely flight-crew awareness to prevent the event or both"[2].
SVS go beyond TAWS and are intended to make the pilots aware of the terrain conditions, traffic and obstacles around the aircraft. It is important to note that so far the intended function of SVS is advisory only [3]. Most SVS use Digital Terrain Elevation Data (DTED) to generate the terrain imagery on the flight displays. Figure 1 shows a simplified block diagram of a typical SVS. All information available from the navigation sensors (such as the Global Positioning System, GPS), attitude sensors (such as Inertial Navigation Systems, INS) and terrain data is combined with the commanded position of the aircraft obtained from the Flight Management System (FMS) and displayed on a Liquid Crystal Display (LCD) monitor in the cockpit of the aircraft. This display could either be a Head Up Display (HUD) or a Head Down Display (HDD) [4].
2.2 Integrity Monitoring

Integrity can be described as the attribute of a system, indicating when it can be relied upon to perform its intended function and when it cannot, accompanied with appropriate warning under the circumstances governing it. One source of misleading terrain information (MTI) in an SVS display is the DTED in the SVS. Any error in the DTED could potentially degrade the performance of that SVS by providing the pilot with misinformation on the surrounding terrain or obstacles. One possible solution to
alleviate this problem is the addition of a terrain database integrity monitor (TDIM) [3].

A terrain database integrity monitor is a system that monitors the integrity of the data obtained from the terrain database and generates an alert when a failure occurs in this data. A SVS TDIM not only checks the DTED, but also provides the user with an indication when the SVS should be used with caution or not to be used at all.

One example integrity monitor concept for an SVS synthesizes the terrain elevation profile from GPS and radar altimeter data, and compares this “synthesized” profile to the profile retrieved from the terrain database. If an inconsistency exists between the synthesized terrain profile and the profile obtained from the DTED, the pilot is alerted in some fashion. Figure 2 shows the basic, functional blocks of a TDIM and its application in a SVS.

---

1 For a detailed description of GPS and radar altimetry in SVS, see [4]
Figure 2: A simplified block diagram of a Terrain Database Integrity Monitor
The integrity monitor function depicted in Figure 2 is carried out using a radar altimeter in conjunction with the navigation and attitude sensors in the SVS shown earlier in Figure 1. SVS prototype hardware and SVS prototype software flowcharts are employed for implementation for this purpose\(^2\). The input, computation, output and decision blocks constitute the TDIM. The terrain elevations of a region are stored in a Digital Elevation Model (DEM) [6] and provide the terrain database height \(h_{DTED}\) for the flight path under consideration. Differential GPS (DGPS) measurements provide the DGPS height \(h_{DGPS}\) of the aircraft with respect to Mean Sea Level (MSL). A Radar altimeter (RADALT), using a downward-looking (DWL) sensor [6] on the aircraft is employed to sense the height Above Ground Level (AGL) denoted by \(h_{RADALT}\). The synthesized height \(h_{SYNT}\) of the terrain surrounding aircraft is computed as follows [3]:

\[
h_{SYNT} = h_{DGPS} - h_{RADALT}
\]

\[\text{------ (1.1)}\]

As shown in Figure 2, Absolute Disparity (AD) is required to measure the bias between the sets of \(h_{SYNT}\) and \(h_{DTED}\). The AD \((p)\) between \(h_{SYNT}\) and \(h_{DTED}\) is computed as follows:

\[
p = h_{SYNT} - h_{DTED}
\]

\[\text{------ (1.2)}\]

\(^2\) For a more detailed description, refer to [4]
For each set of ADs, a test statistic called Mean Square Difference of Absolute Disparities (MSDAD), denoted by $MSD_{AD}$, can be computed as follows:

$$MSD_{AD} = \sum_{i=1}^{N} (p_i)^2$$

where,

$N$ = degrees of freedom

$MSD_{AD}$ can be physically interpreted to be a measure of the similarity between $h_{SYNT}$ and $h_{DTED}$. Based on the $MSD_{AD}$, a test-statistic, $T$, is given by [6],

$$T = \left( \frac{1}{\sigma^2} \right) \times MSD_{AD}$$

where,

$\sigma$ = Standard deviation of the AD

Thus, by observing the $T$-value, it is possible to determine a measure of how much bias $h_{BIAS}$ exists between $h_{SYNT}$ and $h_{DTED}$. To compute the $T$-value, a Multiple Path Downward Looking terrain database Integrity Monitor (MPDLIM) [7] (described in detail in Chapter 4) scheme is employed, which creates multiple terrain profiles by shifting the path of an aircraft horizontally in the terrain database. In the
MPDLIM scheme, horizontal offsets are introduced to the “synthesized” terrain profiles and multiple translations are performed in the terrain database to compute the MSDAD and $T$-values. Based on these computed values of $T$ and a chosen threshold value of $T$ (denoted by $T_{th}$), a decision to generate an alert is contemplated by the integrity monitor. As shown in Figure 2, the input, computation and output blocks are the three blocks that provide the required $T$-value. Based on a preset threshold value, the decision block decides whether or not to generate an alarm. Factors that influence this alert decision include target environmental conditions of the aircraft, visibility, safety-critical situations, etc. Thus, depending on the need of the situation, the integrity monitor should be capable of deciding when to generate an alert or not.

2.3 Certification of Avionics Systems

Every avionics system has aspects related to design, safety, engineering, certification, production and marketing. All these aspects are inter-dependent and play a significant role in ensuring the system's long-term performance and reliability. In the rush of technological advancements, especially in the HW and SW areas, many avionics systems are being designed using a hierarchy of many HW and SW components. Hence, it becomes necessary to evaluate the impact of these advancements on safety-related avionics systems as well. Consequently, the design and certification aspects emerge to be paramount, because they greatly impact the safety of any avionics system. This is, perhaps, one of the reasons why the certification of any airborne equipment, as mandated by the civil aviation authority in the United States (U.S.),
namely, the Federal Aviation Administration (FAA), is a very stringent, planned and iterative process. In order to support and guide the avionics systems' manufacturers and others through this process, the FAA provides very specific guidance, standards and design assurance methodologies to be followed. While the involvement of the FAA is not mandated at every developmental stage of a project under construction, it is always advantageous to keep the FAA informed of all developments in the design process. This aids in reducing the possibility of miscommunication, original design re-modifications at later stages, and ultimately, in saving time and money. A few other organizations such RTCA, Inc., the European Organization for Civil Aviation Equipment (EUROCAE), the Joint Aviation Authorities (JAA) in Europe, and the Society of Automotive Engineers (SAE), provide additional advice and recommendations to promote civil aviation safety and public trust [8].

Under the jurisdiction of Title 49, U.S. Code (Subtitle I: Department of Transportation (DOT), Subtitle VII: Aviation Programs),"The FAA acts primarily through the publication and enforcement of the Federal Aviation Regulations, (FARs), which are Chapter 1 of Title 14, Code of Federal Regulations (C.F.R). "FARs are organized by sections known as Part numbers"[9]. The FAR Parts covering most avionics-related activity are listed in Appendix A. Part 21 forms the basic regulatory document for all certification work and issues. The other types of guidance and policy sources available are FAA Advisory Circulars (AC) and FAA Orders, respectively. The former is of non-regulatory type and the latter prescribe the responsibilities and procedures for FAA personnel [8]. For the benefit of the public, RTCA, Inc., has
developed consensus-based recommendations on contemporary avionics issues, such as SW, electronic HW, GPS and environmental test [10].

The general avionics systems certification process can be summarized as follows:

i. The applicant turns in an application for certification of proposed product or system.

ii. The FAA establishes the certification basis consisting of all policies and procedures to be applied.

iii. The Technical Standard Order (TSO) Authorization Approval Process is initiated (This process is voluntary).

iv. The Type Certificate (TC) or Supplemental Type Certificate (STC) Approval process is initiated.

v. The Installation Approval process (such as Field Approval Process) is initiated if necessary.

vi. Additional Approval processes (such as FAA Form 337 Process) are initiated if necessary.

vii. The FAA issues the certificate and defines the certificate limitations.

As per the requirements of the product or system, the optimum path should be chosen for obtaining the certification, i.e. first a TSO, followed by a TC/STC and an installation approval or only a TC/STC followed by an installation approval. It may be noted that a TSO is voluntary but restrictive and will eventually require a TC/STC, whereas a TC/STC is mandatory but more flexible to allow room for innovations in
avionics systems. Generally, a product which is TSO’d is more receptive in the industry because it is applicable to an aircraft of any type and it is also easier to obtain a TC/STC for a TSO’d product since a prior standard has already been complied with for certification [8][11].

The next chapter will enumerate the certification issues concerning avionics systems, focusing on HW and SW design assurance guidelines, using the SVS integrity monitoring function as a case study.
Chapter 3  
Certification of Avionics Systems: Software and Hardware Design  
Assurance Guidance

3.1  Certification Stages of a Typical Life Cycle Process

Chapter 2 provided a brief overview of the general FAA certification process. For a prospective applicant who wants to have his/her product certified by FAA, the following stages of a typical product life-cycle process form the basis for certification—ensuring communication with the FAA after every stage would only help speed up the certification process of the product once it has been developed.

3.1.1 Concept Development

The initial idea of the functionality and design is described and all other aspects such as design components, environment, costs, application, etc. are evaluated. The purpose of this stage is to provide a big picture of the concept in mind, with respect to the need of the concept, what the intended function is and to check the feasibility of the concept for practical purposes.

3.1.2 Safety Assessment

Once the design concept has been established, it is necessary to assess the safety of the system with respect to human life, property and the environment. It is necessary to analyze all possible failure conditions and effects on the environment of the system and determine the implications and severity of each of the system, subsystem, and component failure conditions. The results of the safety assessment process are used to
quantify and evaluate the safety of the overall system, and show how the system will satisfy the safety objectives established by its intended function. When it becomes necessary to certify a safety-related avionics system, such as TAWS, the entire certification process is more rigorous and iterative and the ability to demonstrate that safety of the system is assured is emphasized.

3.1.3 Preliminary Design

In this stage, a high-level design description is created for the concept developed and system requirements are written accordingly. All requirements are written to describe the system functionality: the requirements are written such that they define what functionality the system should implement, rather than how it should be implemented. For all kinds of requirements, requirement standards can be adhered to, in order to provide a clear and unambiguous description of the system’s functionality.

3.1.4 Detailed Design

High-level and low-level requirements provide the means of describing the detailed design of a system, which are based on the system requirements developed earlier. High-level and low-level requirements enlist all the functionality and components required at the sub-system and sub-component levels. During the detailed design process it is important to make sure that the high-level and low-level requirements can be traced to and comply with the system requirements.
3.1.5 Implementation

Once the low-level requirements have been developed, it is possible to implement the design directly from them, to achieve the functionality of the system as described in the system requirements earlier.

3.1.6 Validation and Verification

After the design has been implemented, it is almost useless if it cannot be verified that it satisfies all design requirements. Therefore, the verification and validation stages are an integral part of the certification process, especially in a safety-related avionics system. These stages are usually also the most involved tasks. Validation is an evaluation of the requirements to check that they describe the functionality of the system correctly, whereas verification is an evaluation of the implementation of the requirements to determine that they have been met [12]. The verification and validation results are submitted to the FAA as evidence to show that the system performs as described by the system and safety requirements in the previous development stages.

3.1.7 Configuration Management

After the completion of each stage, any design additions and changes must be taken in account during the next design stage and feedback occurs to the previous design stages. Therefore, it becomes necessary to track and document all change history created during the design life cycle. Configuration management refers to the process of
keeping track of all change history of the configuration of a system and documenting it to maintain the integrity and traceability of the configuration throughout the systems’ life-cycle.

3.1.8 Quality Assurance

Quality assurance is required to monitor the quality of the system or product being developed. Generally, the quality of a system or a product depends mostly on the people involved in the development and verification processes of the system. Hence, quality assurance monitors the way these groups of people perform their responsibilities and ensures that the system has been developed while adhering to the appropriate standards and procedures and that any inadequacies in the system or standards are brought to light so that they can be remedied.

3.2 SW and HW Considerations

The several stages discussed in the basis for certification are common to any avionics system to be certified. At the same time, additional and detailed considerations are required for SW and HW components, individually. Since many avionics systems today, are built using a large number of HW components and complex SW components, there is a need to have specific guidance that caters to the certification aspects of these components alone. This is especially significant in case of a safety-related application, since it becomes necessary to ascertain how the SW or HW components affect the overall safety of the system. FAA and other organizations have developed standards
that help the general public and manufacturers ascertain and demonstrate that their
products meet the safety and quality requirements for avionics systems with respect to
HW and SW. Although adherence to these standards does not guarantee certification
from the FAA, it definitely establishes a basis for certification and demonstrates that
the system has been designed, implemented and verified as per the system
requirements. Some of the standards and guidance available for SW and HW in
avionics systems are:

- RTCA/DO-178B *Software Considerations in Airborne Systems and Equipment
  Certification*, December 1, 1992
- RTCA/DO-254 *Design Assurance Guidance for Airborne Electronic Hardware*,
  April 19, 2000
- DOT/FAA/AR-95/31 *Design, Test, and Certification Issues for Complex
  Integrated Circuits*, Final Report, August 1996
- Part 25 Generic Programmed Logic Device Generic Issue Paper issued by
  Transport Airplane Directorate (TAD), July 28, 1999[13]

### 3.2.1 SW Certification based on RTCA/DO-178B (Also EUROCAE ED-12)

The purpose of RTCA/DO-178B is to provide guidelines for the production of
SW for an airborne system that performs its intended function with a level of
confidence in safety that complies with airworthiness requirements [14]. These
guidelines are in the form of objectives for SW life cycle processes, descriptions of
activities and design considerations for achieving those objectives, and evidence that
indicates that the objectives have been satisfied. Figure 3 shows the document layout of RTCA/DO-178B. The SW life cycle processes are given below.

Figure 3: Document Layout of RTCA/DO-178B
3.2.1.1 SW Planning Process

The SW Planning Process produces the SW plans and standards that direct the SW development processes and the integral processes. It mainly consists of the Plan for Software Aspects of Certification (PSAC), SW Development Plan, SW Verification Plan, SW Configuration Management Plan, and the SW Quality Assurance Plan. The PSAC is a document that tells FAA of the plan of the proposed development methods and how the applicant intends to comply with RTCA/DO-178B. From this plan, FAA can determine if the proposed life cycle is rigorous enough for the target SW level [14]. Please refer to Appendix B for definitions of SW levels.

3.2.1.2 SW Development Processes

The SW Development Processes include requirements definition, design, coding and integration. It identifies the development standards, development environment and transition criteria.

3.2.1.3 SW Integral Processes

The SW Integral Processes comprise of SW Verification, SW Configuration Management, SW Quality Assurance and Certification Liaison activities. This set of processes, undoubtedly, is the most time-consuming and tedious of all the certification processes, because every requirement needs to be validated, and every design component needs to be tested, verified and all the test results must be documented as evidence. Figure 4 illustrates this fact for all SW levels [14].
3.2.1.4 Additional Considerations for SW

Additional considerations for SW include any SW components involving Previously Developed Software (PDS) and tool qualifications. PDS includes Commercial off-the-shelf (COTS) SW (e.g. compressing SW such as Winzip™), airborne SW developed to other standards, airborne SW that predates RTCA DO-178B and airborne SW developed at a lower SW level. These considerations shall be revisited later in this document.
3.2.2 HW Certification based on RTCA/DO-254

The purpose of RTCA/DO-254 is to provide HW design assurance objectives for the development of airborne electronic HW such that it safely performs its intended function, in its specified environments [12]. RTCA/DO-254 uses a top-down approach, based on the system functions being performed by electronic HW and not a bottom-up perspective or an approach based solely on the specific hardware component used to implement the function. This approach is more effective for recognizing and isolating safety-design errors and facilitates the verification processes. The guidance in the document is applicable, but not limited, to Line Replaceable Units (LRUs), circuit board assemblies, custom micro-coded components such as Application Specific Integrated Circuits (ASICs), PLDs and COTS HW components. Unlike SW, HW is classified as follows in the context of certification:

- **Simple HW Item:**

  A HW item is considered simple if a comprehensive combination of deterministic tests and analyses can ensure correct functional performance under all foreseeable operating conditions with no anomalous behavior. Examples of simple HW items are diodes, resistors, and logic gates [12].

- **Complex HW Item (Complex Electronic Hardware (CEH)):**

  A HW item is considered complex if it cannot be classified as a simple HW item. Such an item is known as Complex Electronic Hardware (CEH). This means that
CEH cannot be tested 100% and exhaustively and that it is very difficult to prove unintended function or malfunction because of this complexity. Examples of CEH include ASICs, PLDs, and FPGAs [13].

Depending on the type of HW components involved, the design assurance procedures vary as a function of the complexity and the safety-related aspects involved. The HW Life Cycle process consists of the following processes.

### 3.2.2.1 HW Safety Assessment

The HW safety assessment is conducted to support the system safety assessment processes, such as the functional hazard analysis (FHA), the preliminary system safety assessment (PSSA) and the system safety assessment (SSA) [16]. The intent of this safety process is to demonstrate that the applicable systems, including the HW subsystems, satisfied the safety requirements of applicable aircraft certification requirements. A Functional Failure Path (FFP) Analysis is a method of safety assessment that is structured, top-down, iterative analysis that can be used to identify individual FFPs to be addressed by an appropriate design assurance method. Decomposition of FFPs progresses as shown in Figure 5.
3.2.2.2 HW Design Life Cycle Processes

Figure 6 shows the details of the HW design life cycle processes, which produce a HW item that fulfills the requirements allocated to HW from the systems requirements [12]. The HW design life cycle processes comprise of the following processes:

3.2.2.2.1 HW Planning Process

The purpose of the HW planning process is to define the means by which the functional and airworthiness requirements are converted into a hardware item with an acceptable amount of evidence of assurance that the item will safely perform its intended functions. This involves the generation of a Plan for Hardware Aspects of Certification (PHAC), a proposed method of design, the selection and definition of design and development standards, among other objectives.
3.2.2.2 HW Design Processes

The HW Design processes are shown in Figure 6 and they may be applied to any hierarchical level of the HW item, such as ASICs and PLDs. These processes are:

A) **Requirements Capture** - This process identifies and records the HW item requirements. Process activities may include imposition of design assurance levels as identified from the safety assessment process, probabilistic requirements, derived requirements, such as timing relationships, supply voltage demands and signal noise.

B) **Conceptual Design** - This process produces a high-level design concept that is anticipated to result in an implementation of the desired design. Conceptual design can be accomplished using functional block diagrams, design and architecture descriptions, etc. It should be noted that, regardless of the design description, the evidence to support the design assurance level must be provided. Additionally, HDL design representations used coded text-based techniques that seem to resemble those used for SW representations. This similarity in appearance can mislead one to attempt to use SW verification methods directly on a HDL design representation. The guidance of RTCA/DO-254 is applicable for design assurance for designs using an HDL representation [12].
Figure 6: Hardware Design Life Cycle [12]
C) & D) **Detailed Design and Implementation**- Detailed design data should be generated based on the requirements and conceptual design data. Architectural design techniques, constraints and test features should be incorporated wherever possible to allow verification of safety requirements. Implementation process uses the detailed design data to produce the HW item that is an input to the testing activity.

E) **Production Transition**- This process uses the outputs from the implementation and verification processes to move the product into production. Manufacturing data, baseline data required duplicating a HW item on a large scale and other objectives are outputs of this process.

### 3.2.2.2.3 Supporting Processes

These processes are shown in Figure 6 and have the same interpretation as defined for the RTCA/DO-178B-based SW certification. However, the implications for HW items involve some different considerations. “Experience indicates that attention to the development and validation of requirements can identify subtle errors or omissions early in the development cycle and reduce exposure to subsequent redesign or inadequate HW performance”[12]. Verification, as mentioned before, is as important as the design process; in the case of HW items and SW/HW integrated items, it becomes more complicated and must be addressed separately. These issues will be discussed in detail in Chapter 6.
3.2.2.2.4 Additional Considerations

Additional considerations for HW items include use of Previously Developed HW, including the use of COTS components, design upgrades, especially in hierarchical digital designs, tool assessment and qualification for development and verification tools. These and other additional considerations will be enumerated in Chapter 6 with respect to FPGA-based avionics systems, such as SVS.
Chapter 4  Functionality and Implementation Platforms of Terrain Database Integrity Monitor for a SVS

4.1  Choice of Implementation Platform

Chapter 2 described the basic functionality of a TDIM as part of an SVS. The basic components of the TDIM that are required to compute the $T$-values are shown in Figure 2. For purposes of this thesis, only the input, computation and output blocks are considered to constitute the TDIM, the decision block is not included in the implementation. Based on this assumption, the functionality for the input, computation and output blocks is discussed in this chapter.

The values of $N$, $\sigma$, $T_{th}$ and the basic TDIM algorithm, which are used for the proposed implementation in the following sections, are obtained from the results of extensive research conducted in the past on terrain database integrity monitoring for SVS [18][7][19]. The research results described in these papers show that the chosen parameter values and test-statistics are the suitable choice for the improved performance of a TDIM for a SVS. In this thesis, these results are extended to a HW/SW safety-related design of a component for TDIM, which will be applicable in a SVS.

4.1.1  HW/SW Co-Design

In the TDIM shown in Figure 7, the input block pre-processes the input data whereas the computation block handles the computation-intensive operations. The key point is that if the computation block could be designed such that it processes its inputs
at a high speed, then the performance of TDIM can be enhanced greatly. This is possible if both HW and SW are used to build the TDIM- i.e. HW/SW co-design is employed. “Hardware/software co-design means meeting system-level objectives by exploiting the synergism of hardware and software through their concurrent design” [20]. Rather than targeting all the blocks of the SVS TDIM for a HW device, only the functionality of the computation block is designed in HW whereas the input and output blocks are implemented in SW on the host PC.

Figure 8 shows the interface of the HW and SW components. Depending on the size of the data that needs to be stored, the terrain elevations can be stored in an external storage element, typically a Programmable Read Only Memory (PROM).

As described in the MP-DLIM scheme in Section 2.2 of Chapter 2, $N \times N$ horizontal offsets are introduced to the “synthesized” terrain profiles and multiple

![Figure 7: Components of a TDIM](image)

Figure 7: Components of a TDIM
translations are performed in the terrain database to compute the MSDAD and $T$-values.
After the input block has processed the input data in SW, the computationally intensive
operations required for computing the $N \times N$ MSDAD and $T$-values can be offloaded to
the HW component and sent to the output block in SW. Thus, HW/SW co-design helps
implement the functionality of the SVS TDIM with faster processing of input data and
better utilization of HW resources.

![Figure 8: HW/SW co-design](image-url)
Figure 9 shows the functionality required for the MP-DLIM scheme, which is described in detail for each block in the following sections. It also shows those portions of the functionality that correspond to SW and HW components.
Figure 9: Flowchart of MPDLIM functionality
4.2 Terrain Database

A terrain database stores elevations for the region under consideration. Data is typically obtained from a DEM, which contains elevations at a fixed, lateral resolution, for the set of longitudes and latitudes corresponding to that region. Figure 10 shows the MATLAB™ elevation plot for a region of 603 arc-seconds (approximately 1/6 of a degree) of latitude and longitude each. The color bar in the plot indicates the color scaling for the elevations in meters.

Figure 10: Terrain database elevation profile (meters) with flight path
An example of an aircraft flight path is also depicted (in red) which will be explained shortly. The data depicted in Figure 10 represents an area in the vicinity of Eagle/Vail, CO [21]. The Eagle/Vail airport or EGE is located right above the middle of the picture. The spatial resolution, defined as the difference between each latitude step and longitude step, is 3 arc seconds (arcs), which corresponds to approximately 92 meters (m) at the equator.

\[ 3 \text{ arcs} = \frac{3}{3600} \text{ degree} = \frac{1}{1200} \text{ degree} = 8.333 \times 10^{-4} \text{ degree} \qquad \text{(4.1)} \]

\[ 3 \text{ arcs} = 8.333 \times 10^{-4} \text{ degree} \times 60 \text{ nautical miles} \times 1852 \text{ meters} \approx 92 \text{m} \]

Distance in longitude direction of terrain database = 92 \times \cos(40) \approx 70 \text{m}

It is important to note that, at higher latitudes, the lines of longitude are closer; therefore, the same 3-arcs terrain database would represent a distance of less than 92m in the longitude direction at higher latitudes. Distances in the latitude direction are independent of the latitude (Conceptually and practically, latitude is the same no matter where you go on earth; however, in reality it varies from the poles to the equator due to the earth bulging slightly from its rotational spin). The distance in the longitude direction varies as the cosine of the latitude. The terrain database shown in Figure 10 is at latitude of approximately 40° N, hence the distance in the longitude direction of the terrain database step is approximately 70 meters, which is illustrated in Figure 11.
Figure 11: Resolution of terrain database (not to scale)

Figure 11 illustrates the resolution of the terrain database. It is evident from Figure 11 that elevations would be available only for the positions marked for every intersection of latitude and longitude and the elevations at the other positions must be computed from its four closest neighbors through a bi-linear interpolation [22]. The elevation for a particular position is extracted from the terrain database by the computation block to compute the desired outputs.
4.3 Input Block

Figure 10 shows a flight path (in red, with black arrows showing the direction of flight) superimposed on the terrain elevation data. For each flight path point, the GPS MSL (Mean Sea Level) height and radar altimeter AGL (Above Ground Level) height are measured. Both height measurements and the GPS latitude and longitude estimates form the input block data. For the computation block, a set of $N$ consecutive points of the flight path segment are selected indicated by the yellow portion of the red flight path, as shown in Figure 10. All computations, henceforth, are performed with respect to this flight path segment alone.

For the SVS TDIM the values of $N$ and $\sigma$ from Equations 1.3 and 1.4 that will be used in the example computations are 50 and 18.9 m, respectively [6]. In this case, $N$ refers to the number of past samples that is used to determine the $T$-value at a time. The threshold value, $T_{th}$ is the threshold against which the computed $T$-value is checked. For the example computations, $T_{th}$ is chosen to be 96 [6].

As shown in Figure 12, the input block accepts the test data and initialization parameters and computes the synthesized elevations for the chosen path segment. These elevations along with the latitudes and longitudes of the corresponding path segment are pre-processed. The pre-processing step consists of changing the numerical format of the input parameters to all decimal (fixed-point) values. Thus, the preprocessing steps include scaling integers, precision rounding, etc.
4.4 Computation Block

The idea of terrain database integrity monitoring is to check the presence of failure modes in the terrain elevation database. For a SVS, the most significant factor would be to check if each elevation stored in the database is the correct elevation for that position of terrain.

This concept is illustrated in Figure 13. The elevations for positions (1,1), (2,2) and (3,3) (line AB) in the terrain database are 5, 7 and 2 meters, respectively and the elevations for positions (2,0), (3,1) and (4,2) (line CD) are 29, 25 and 47 meters, respectively. However, if the measured elevations for positions (1,1), (2,2) and (3,3) (line AB) differ from the true elevations (say, 30, 25 and 48 meters respectively), then this could imply that there is either a fault present in the terrain elevations (vertical direction) or that the elevation values are accurate, but refer to erroneous reference points (horizontal faults).
Assuming that the terrain database has accurate values of elevations, this would imply that a spatial bias exists in elevations stored in the terrain database. Considering a linear bias alone, in Figure 13 this would imply that the elevations for positions (1,1), (2,2) and (3,3) (line AB) in the terrain database actually correspond to positions (2,0), (3,1) and (4,2) (line CD) on the real terrain.

Such a linear bias can be detected if the true elevations measured for positions on line AB (30, 25 and 48 meters) are compared with the stored elevations for positions on line CD (29, 25 and 47 meters) and their MSDAD is computed. Consequently, the MSDAD of measured elevations on line AB after translation to positions on line CD

Figure 13: Checking for linear bias in terrain
would be less than the MSDAD of the same elevations without the intentionally introduced translation.

Thus, to detect the presence of a linear bias in the terrain database, the MP-DLIM scheme, mentioned in Section 2.2 of Chapter 2 is used. Linear two-dimensional translations of the path segment (consisting of a set of positions of a flight path) are introduced in the terrain database and the MSDAD values of the set of elevations of the path segment are computed for each translation. The MSDAD values are scaled by $\sigma^2$ to compute the $T$-values. Mathematically, this can be expressed as follows [7].

$$
T(m, n) = \left[ \frac{1}{\sigma^2} \sum_{i=1}^{N} (h_{SYNT}(t_i) - h_{DTED}(x, y))^2 \right]^{\frac{1}{2}}
$$

where,

$$
x = \text{lon}(t_i) + \text{lon}_{off_m} \quad \text{for the index } m = 1 \text{ to } M
$$

$$
y = \text{lat}(t_i) + \text{lat}_{off_n} \quad \text{for the index } n = 1 \text{ to } M
$$

The translation offset can be set as an initialization parameter or a range of values, which would determine the translation area (shown by black grid at the start of the yellow path segment in Figure 10). The location of the minimum value of $T$ indicates the set of positions in the terrain database that are the most similar to the measured set of elevations of that path segment. Figure 10 shows a small (black) translation grid at the start of the (yellow) path segment, which is enlarged in Figure 14.
The left plot in Figure 14 shows the minimum of the $T$-value (which is shown by the red dot) in the translation grid, which is located on a translation offset of 0 in latitude and -1 in longitude from the center of the translation grid. This implies that at location (-1,0) in the translation grid, the terrain profile of the chosen path segment is most similar to that of the terrain database. Sorting all the $T$-values based on a threshold value generates the right-hand side plot in Figure 14. All values below the set threshold are shown in blue and those above are shown in brown.

From an SVS perspective, the right-hand side threshold plot is more significant. Whenever a discrepancy in the terrain profile is detected, the decision block of the

**Figure 14: T-value and threshold plots**
integrity monitor should generate an alarm informing the pilot of the discrepancy.
However, in a radar altimeter-based SVS TDIM, an alarm will not be generated since
the blue areas indicate those regions where a lateral error exists in the terrain database
and the consistency metric is below the selected threshold value. Ideally, these blue
regions should be as small as possible to improve the accuracy of the SVS TDIM [23].

![Computation Block](image)

**Figure 15: Computation Block**

Figure 15 shows the inputs and outputs for the computation block. This block
accepts the measured elevations for the path segment, compares them with the stored
terrain database elevations over a translation area to find the $T$-values of the set of
elevations in the path segment. It also finds the minimum $T$-value, its location and the
$T$-values below the threshold.
4.5 Output Block

The computation block provides the $T$-values of the set of elevations in the path segment, the minimum $T$-value, its location and the $T$-values below the threshold. The output block is used to store the latitude and longitude profile of the terrain database.

Using this profile, the latitude and longitude position of the minimum $T$-value that is sent by the computation block is computed. Figure 16 shows the inputs and outputs of this block. It can also be used to modify its outputs to make them compatible for another block, e.g. sent to an image-processing block or changed to binary format.
4.6 Choice of HW Platform for TDIM (System Requirements)

The concept of a TDIM for a SVS was developed for a real-time implementation, using online GPS and Radar Altimeter inputs. But, for purposes of this thesis, a TDIM for an SVS is implemented using offline data only. Data obtained from flight tests [24][25] is used instead of real receiver inputs and the DTED data surrounding the flight test region is used to form the terrain database. Although the design is implemented using offline data, the fastest rate of data input, processing and output are considered, while keeping the future, real-time implementation in mind. Based on the MP-DLIM algorithm, the following HW and SW requirements are defined:

1. A non-volatile memory shall be used to store the elevations of the terrain under consideration (HW requirement).

2. The spatial resolution of the terrain database shall be a value that corresponds to an integral power of 2 in terrain database steps per degree of latitude and longitude each. If necessary, a new terrain database with such a spatial resolution shall be constructed from the existing terrain database by bi-linearly interpolating from the terrain database elevations and by re-sampling the latitude and longitude profiles with the required spatial resolution (SW requirement).

3. The input data shall be read in from the input source, which are the input latitudes, input longitudes, GPS MSL heights and RADALT AGL heights (SW requirement).
4. The input data shall be pre-processed conform the fixed-point formats used in
the computation block.

5. Linear two-dimensional translations, as defined by the MP-DLIM scheme (in
Equations 4.2 to 4.4) of the path segment (consisting of a set of positions of a
flight path) shall be introduced in the terrain database and the MSDAD values
of the set of elevations of the path segment shall be computed for each
translation (HW requirement).

6. The corresponding elevation should be extracted from the terrain database (HW
+ SW requirement).

7. If the point corresponds to an unknown location in the terrain database, bi-linear
interpolation shall be used to compute the elevation of the unknown location.
The elevations of the closest surrounding four points of the unknown location
shall be used to interpolate and compute the elevation of the unknown location
(HW requirement).

8. The $T$-value shall be computed for each translation (HW requirement).

9. If all the translations for that point are complete, then the next point of the path
segment shall be read in (HW requirement).

10. If all the points of path segment are complete, then the minimum $T$-value, its
location (in terms of latitude and longitude offset from the center point) and the
$T$-values below $T_{th}$ shall be computed (HW requirement).

11. The output block shall find the location of the minimum $T$-value in the latitude
and longitude profile of the terrain database (SW requirement).
Assuming an input rate of 1 point per 1 second (it is the rate at which an Aeronautical Radio Inc. (ARINC) receiver provides GPS MSL heights), the outputs of the output block need to be ready within 1 second. This requires high speed, a large amount of memory for storing the $T$-values and also re-initialization of the HW components for every new input position or new terrain region being considered.

Traditionally, a combination of ASICs could be built to implement the functionality of a SVS TDIM as described above. But ASICs do not provide the architectural flexibility offered by FPGAs. Also, performing the position translations in a sequential fashion would introduce a large processing delay, which would make it impractical for real-time applications.

4.7 About PLDs and FPGAs

Keeping the HW requirements stated in Section 4.6 in mind, an FPGA is chosen as the HW platform for implementing the functionality of the computation block in the SVS TDIM. FPGAs are high gate-density, re-configurable PLDs, which can be programmed repeatedly to implement the desired functionality by the end-user. Figure 17 [26] shows the family of logic devices. Since a device from the Xilinx™ FPGA device family will be considered for the HW implementation, a basic Xilinx™ FPGA structure is shown in Figure 18 [27].
Figure 17: Logic Device Family

Figure 18: Xilinx™ FPGA Structure [27]
An FPGA consists of an array of Configurable Logic Blocks (CLBs), which are arranged in a matrix within the perimeter of Input/Output Blocks (IOBs) and an internally distributed array of Static Random Access Memory (SRAM) cells, and interconnected via a routing network called Programmable Interconnect. Programmable-interconnection resources in the FPGA provide routing paths to connect inputs and outputs of the IOBs and CLBs into logic networks [28]. Each IOB shown in Figure 18 is user-configurable and provides an interface between the external package pin of the FPGA chip and the internal user logic.

The array of CLBs provides the functional elements from which the user’s logic is constructed. Figure 19 [28] shows the CLB structure of a Xilinx™ XCV3000 FPGA device. Each CLB includes a combinatorial logic section, multiplexer (MUX) selections of functions and two flip-flops (which act as output registers). Each combinatorial section consists of an SRAM cell, which is the configuration program memory and controls the MUXs selection functions and the configuration of the entire CLB. Consequently, each SRAM cell holds one configuration bit and all the bits in each SRAM cell inside the FPGA constitute the bit-file that is used to configure the CLBs inside the FPGA. These SRAM cells allow the user to implement look-up tables (LUTs) and are therefore called function generators. The outputs of these LUTs are inputs to the MUXs, which are controlled by the SRAM cells of the configuration program memory. The two flip-flops can be used as registers to store the output of the function generators.
It is this combination of SRAM-based LUT function generator logic, configuration of the CLB through the configuration program memory controlled-MUXs and output register logic that makes the FPGA a very high-performance, flexible device that can be exploited to implement high-speed, hierarchical and pipelined structure-based designs. Newer versions of Xilinx™ FPGA device families, such Virtex-E ©, have many advanced features such as:
- SRAM cells that can be configured as RAM, Dual-Port RAM or high speed adders
- Partitioned IOBs to group and optimize I/O data based on voltage ranges
- Carry-ahead logic to minimize propagation delay of carry-bit in cascaded designs
- VersaRing©; better I/O routability and use of routing resources to reduce path delay
- Block-Select©RAMS: to implement deeper, customized Dual-port RAM (DPRAM)
- Delay-locked loops: to eliminate the internal clock skew
- IEEE Standard 1149.1 Boundary Scan logic: to test the device before initial use
- Dedicated multipliers that use shift-register logic and help reduce carry-delay
- and many more [29]

Figure 20 shows a 2-Slice Virtex© CLB with an extra carry-control block in each Logic Cell (LC) (Four Virtex© LCs or slices make up one CLB in the Virtex© Architecture)[30]
In general, FPGAs offers the following advantages over ASICs:

1. Architectural flexibility
2. Higher processing speed (Parallel processing capabilities)
3. Re-configurability using HDLs
4. Reliability
5. Less time-to-market, no minimum order quantities
6. Other ingenious advantages e.g. programming FPGAs remotely via Internet!

Figure 20: 2-Slice Virtex© CLB
However, FPGAs also have some disadvantages:

1. FPGAs are not as easy to use as PLDs.

2. Timing is difficult to handle at lower-level HW design.

3. FPGAs mostly require use of HDLs to manage complexity.

4. Higher development costs as compared to ASICs (depends on the quantity—large volumes, ASICs are cheaper, whereas for small volumes, they are still relatively expensive to develop. In aviation applications, where generally low quantities of ASICs are required, development costs of FPGAs might turn out to be lower).

5. High per unit/cost because of customization facilities.

From the SVS perspective, the use of the FPGA as a HW platform is a good choice because the high-speed and parallel processing features of the FPGA can handle multiple simultaneous two-dimensional position translations. On-chip distributed memory can be used to store MSDAD values. Even if the input rate is higher, techniques such as pipelining and HW/SW co-design can be employed to meet the performance requirement. Chapter 6 will enumerate on the certification aspects FPGAs, including verification and validation. The next chapter will discuss the HW design of each component of the SVS TDIM in detail.
Chapter 5  

Design Process for SW and HW Design Components of TDIM

5.1  

Scope of HW Design

Chapter 4 described the functional block details of the SVS TDIM. The next step is the design strategy for implementing the required functionality. Before proceeding with the detailed design of the SW and HW design components, it would be worthwhile to assess the complexity of the design involved. The SW design for the input and output blocks is not very complicated and can be implemented using Matlab™ as a design tool. On the other hand, the HW design is not quite as simple due to the design process of the FPGA device. Figure 21 shows the typical FPGA design process [31].

The first step in FPGA design is the high-level description of the algorithm that depicts the desired functionality. This description is typically a SW code, e.g. Matlab™ or C language. The results of this high-level description are used as the cross-reference results to verify the results of HW simulations. The next step is the design of the behavioral model of the HW components for the high-level functionality. HDLs such as, VHDL and Verilog, are typically used for describing the functionality of a design. Using the outputs of the input block to model real receiver inputs, the VHDL description is simulated to check if the correct functionality is being implemented or not. At this stage of design, no timing constraints are considered. It is important that after design stage, the results are verified with those of the previous stage and, if necessary, the design stages are re-implemented.
This structured and iterative nature of the FPGA design process makes it a good choice for certifiable HW to ensure that verification and design requirements have been complied with.
The next step is to generate a Register Transfer Level (RTL) model that follows the data path of the behavioral model. In this stage, registers are used to capture the data between partitioned design blocks, known as logic levels. The data is transferred through the entire data path from one logic level to the next. After this RTL model has been simulated, a synthesis tool is used to carry out logic synthesis. Logic synthesis is a process that starts from a high level of logic abstraction (typically Verilog or VHDL behavioral model) and automatically creates a lower level of logic abstraction using a library containing primitives [32]. Timing and any other implementation constraints can be specified during synthesis. Synthesis of the RTL model helps gain an estimate of the device resources that will be used, the path delay associated with each logic level and the maximum delay for the entire data path. As shown in Figure 21, the synthesized model goes through mapping, place and route procedures, which produce a configuration file. A configuration file is a file that contains the bitstream that is used to program an FPGA device. The FPGA device is connected to the host PC and is loaded with the configuration file, which configures the FPGA with the designed functionality. Thus, each time the FPGA is loaded with a new configuration file, a new functionality is implemented, making an FPGA a very flexible, re-usable and application-specific device.

For purposes of this thesis, the scope of the HW design of the SVS TDIM component is limited in the FPGA design process as shown in Figure 21. This scope includes the simulation of the VHDL behavioral model, a proposed RTL model based on this behavioral model and an analysis of the RTL-based pipelined model for
implementations of the remaining stages of the FPGA design process. The goals of this thesis include the functional simulation of the HW design methodology, verification of the simulation results, a proposal of the RTL-based pipelined model for synthesis and last, but not the least, the SW and HW certification aspects of the HW design and verification process, as seen from the SVS perspective. The following sections of this chapter describe the design process and methodology used to implement the aforementioned goals.

5.2 Design Process

Figure 22 illustrates the design process used to design the HW component for TDIM of a SVS. Matlab™ (v6.1) is used to create a high-level description of the functionality for a SVS TDIM and the input and output blocks, as shown in Figure 22. The input data consists of the DTED, initialization parameters, flight test data and a modified terrain elevation profile. As the first step, the high-level description is simulated in Matlab™. Then, using the design methodology for the HW component, a low-level description is generated in Matlab™ to model the VHDL implementation in terms of precision and assumptions made for the VHDL simulation. This low-level description is necessary for two reasons. First, it allows better understanding of the HW that will be involved in executing a particular operation.
Figure 22: Design process of TDIM HW and SW Components
Second, the results of the low-level description serve as a good reference to crosscheck the accuracy of the output of the VHDL simulation. The low-level description consists of the input, computation and output blocks, all implemented in Matlab™. In the next step, only the computation block is off-loaded for a behavioral description in VHDL. Aldec Active-HDL™ (v6.1) is used as the VHDL simulation tool to simulate the behavior of the computation block. Text Input/Output (Text I/O) is used to transfer data between the Matlab™ and Aldec Active-HDL™ environments. Based on the VHDL behavioral model, an RTL model is proposed, that can be used for synthesis in the FPGA design process. For improved performance, an analysis of a pipelined RTL model is carried out. The following sections of this chapter describe the design process in detail.

5.3 High-level Description of Algorithm

A Matlab™ code [33] describes the MP-DLIM algorithm defined by Equations 4.2 through 4.4, as depicted in Figure 9 of Chapter 4. The purpose of this code is mainly to illustrate what functionality needs to be implemented rather than how it should be implemented. From a certification perspective, this is analogous to the system requirements design stage. After examining the inputs and outputs of this design stage, all the parameters necessary to generate the low-level description are calculated. These parameters include ranges and precisions of the input and output data, looping parameters, etc. The high-level description also gives an idea of how the results of this design stage can be interpreted for a SVS.
5.4 Low-level Description of Algorithm

The low-level description is also implemented as Matlab™ code, but it implements the functionality described by the high-level description by following the design process shown in Figure 22. It consists of the input, computation and output blocks, all of which have been designed as if they were to be implemented in HW. For example, all multi-dimensional arrays, which are indexed using row and column indices in the high-level description, are replaced by either row, or column arrays, which are indexed using a single index. The reason for this is that memory is simply a one-dimensional stack of values stored at their corresponding addresses. Care is taken to avoid using built-in Matlab™ functions, e.g. ‘find’ (value in an array) and the functionality of these functions is enumerated in low-level Matlab™ code instead. Other design modifications include the modification of the terrain database structure for HW design, implementation of a memory-addressing scheme and the choice of the precisions of the input values to match those of the VHDL implementation. The following sections describe the design process for the low-level description in detail.

5.4.1 Terrain Database

For purposes of this thesis, the terrain database depicted in Figure 10 of Chapter 4, is a “.mat” (Matlab™ format) file containing the elevations of the region under consideration. For simulating the VHDL description, the elevations in this “.mat” file are stored in a text file on the PC and read in by Aldec Active-HDL™. Although no physical memory is involved, the elevations must be organized in the text file such that
they represent physical addressable memory. Typically, a terrain database contains elevations for a grid of positions specified by latitudes and longitudes. From a HW perspective, these elevations must be mapped into a 1-dimensional memory stack with addresses that correspond to the positions specified in the terrain database. Figure 23 illustrates the mapping of a terrain elevation profile into memory.

Each combination of latitude and longitude corresponds to a single address in memory. The number of points in the terrain database determines the number of
required memory locations. All the values that correspond to one latitude of the longitudes of the terrain database elevation profile constitute a page in memory. Thus, every latitude page contains $M$ longitude values and the entire terrain database contains $M$ pages of latitudes. Therefore, for a terrain profile having $M$ latitudes and $M$ longitudes, the number of addresses will be $M^2$, using zero as the starting address. Since the elevations in memory are only identified by their addresses and not by the latitude and longitude values of their positions, a memory-addressing scheme is needed to map the incoming latitude and longitude of a position (say, in a flight path) to the memory address where the elevation for that point is stored. This task is performed in the pre-processing stage.

5.4.1.1 Direct Memory Access (DMA) of terrain database elevations

In order to map the coordinates of a point to its address in memory, the longitude and latitude values of that point and its address should be correlated in such a way that these values directly point to the corresponding address without having to search through the entire terrain database. This type of memory access is called direct memory access (DMA) of data. This type of memory access is possible only if the addresses in memory follow a mathematical expression, in this case, an integral power of $2^n$ an integer. This stems from the fact that the integral power of 2 is the number of bits required to represent a number in binary. Therefore, it follows that every degree of longitude and latitude should be subdivided into $2^k$ steps. The resulting step size corresponds to the new spatial resolution of the database. Note that $k$ is a positive
integer. Consequently, every terrain database step, i.e. page, in latitude and longitude each should have $2^k$ addresses as well. In other words, the lateral resolution of the terrain database must be such that there are $2^k$ terrain database steps in each page of latitude and longitude.

### 5.4.1.2 Modified Terrain Database

Figure 10 of Chapter 4 shows the plot of the terrain elevation profile of flight test data that was obtained from Eagle/Vail, Colorado [24]. This terrain database has a lateral resolution of 3 arc seconds, which implies that there are 1200 terrain database steps in every 1-degree step of latitude and longitude each. In a binary system,

$$3 \text{ arc-s} = \frac{1}{1200 \text{ deg}} = 2^0/2^{10.2281} \Rightarrow 1200 \text{ is not an integral power of 2} \quad (5.1)$$

Thus it is not possible to use this database for implementing the DMA indexing scheme. In order to do so, the original database was modified (as per system requirement no.2 in Section 4.6 of Chapter 4) to have a 3.515625 arc-s resolution such that,

$$3.515625 \text{ arc-s} = \frac{1}{1024 \text{ deg}} = 2^0/2^{10} \Rightarrow 1024 \text{ is an integral power of 2} \quad (5.2)$$

Table 1 shows the difference between the original terrain database and its modified version for two sample pages.
Table 1: Original Terrain Database and modified version

<table>
<thead>
<tr>
<th>Lat/Long (degrees)</th>
<th>Terrain database steps</th>
<th>3 arc-s resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1°</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1200</td>
</tr>
<tr>
<td>2°</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lat/Long (degrees)</th>
<th>Terrain database steps</th>
<th>3.515625 arc-s resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1°</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1024 = 2^10</td>
</tr>
<tr>
<td>2°</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1024 = 2^10</td>
</tr>
</tbody>
</table>

The original terrain database has the following parameters:

1. \( N = \) number of points around the center point in 4 directions = 100
2. \( M = [(2^N) +1] [(2^N)+1] = \) region of interest with center point = 201x 201 points
3. Size of Longitude vector = Size of Latitude vector = 201 points
4. Lateral resolution = 3 arc-s = 1/1200 deg = 8.333e-4 deg, total area is 600x600 arc-s

Scaling the latitude and longitude profiles in the old terrain database with the new lateral resolution i.e. 3.515625 arc-s creates the new latitude and longitude profiles for the new terrain database. The elevations for the new terrain database are obtained by interpolating between the elevations of the old terrain database. It is important to note that for a real-time implementation, a terrain database containing measured elevations for valid latitude and longitude positions should be used, because interpolation introduces a filtering effect in the resulting terrain database elevations.
Since $M = 201$ and the length of latitude/longitude profile for the original database = 600arc-s, the new value of $M$ is the value closest to $600/3.515625$ and which is also an integral power of 2. Since $600/3.515625 = 170.66$, $M$ is chosen to be equal to 256. There is no need to have all 1024 values since only 171 elevations are available and 256 points serve the purpose of DMA indexing in this case. Since elevations have been computed only for the 171 points from the old database, the remaining of the 256 points are stored with zero elevations. This is illustrated in Figure 24, with the zero elevations shown with the shaded region. Figure 24 also shows that the terrain database is asymmetric around the center point because the number of points on each latitude and longitude is even.

**Figure 24: Representation of the Modified Terrain Database (Not to scale)**
The parameters for the new terrain database are:

1. \( N \) = number of points around the center point in 4 directions = 127 (excluding the region of points of the asymmetry as shown in Figure 24)

2. \( M = [(2\times N) +1][2\times N] +1 \) = region of interest with center point = 256x 256 points

3. Size of Longitude vector = Size of Latitude vector = 256 points

4. Lateral resolution = 3.515625 arc-s = 1/1024 deg = 9.765625e-4 deg, total area is 597x597 arc-s

5. All elevations are rounded to integers and their units are meters.

For purposes of this thesis, this modified terrain database is used to test the functionality of the TDIM as described earlier. Thus, for \( M = 256 \), the modified terrain database memory has addresses ranging from 0 to \( (256\times 256)-1 = 0 \) to 65535. Each block of \( M = 256 \) addresses starting from zero corresponds to one page of latitude and all longitudes for that particular latitude. The memory-addressing scheme to read out elevations from these pages of memory is described in Section 5.4.2.1.

### 5.4.1.3 Bi-linear Interpolation Offsets in Terrain Database

When the input block receives the input latitude and longitude from the input device (GPS receiver and RADALT), the computation block must access the terrain database and retrieve the elevation stored for that position. If the latitude and longitude belong to a position that is explicitly specified in the terrain database retrieval of the stored elevation is straightforward. But if the incoming latitude and longitude specify a position unknown in the terrain database, i.e. either the input position is out of the
region specified by the terrain database, or the position is specified with a resolution higher than that of the terrain database, then the elevation for that position cannot be retrieved directly from the terrain database. The former situation can be avoided by performing a simple range check before accepting the input data. In the latter case, bi-linear interpolation is employed to obtain the elevation for the unknown input position in the terrain database [22]. Figure 25 shows point E surrounded by four points A, B, C, and D.
Point E is the point whose elevation is unknown and points A, B, C, and D are the four posts, whose elevations are used to compute the elevation of point E, using the following expression [33]:

\[ Elv(E) = elv(A) + [elv(B) - elv(A)]dx + [elv(D) - elv(A)]dy + [elv(A) + elv(C) - elv(D) - elv(B)]dxdy \]

\[ \textbf{(5.3)} \]

where,

- \( elv(point) \) represents the elevation of a point
- \( dx = (\text{longitude of point (E) - longitude of point (A)})/\text{lateral resolution of database} \)
- \( dy = (\text{latitude of point (E) - latitude of point (A)})/\text{lateral resolution of database} \)

In terms of the memory, the quantities \( dx \) and \( dy \) represent the offsets from the terrain database posts. In the case of TDIM, such sub-resolution values are encountered in the terrain database because of the difference in the low resolutions of the terrain database latitudes, longitudes and the higher resolutions of the input data. This implies that the addressing scheme of having \( M \times M \) addresses to represent the \( M \times M \) points of a region must be augmented with a sub-address or sub-database step resolution offset as well. This means that it is required to break down each terrain database step in longitude and latitude into a combination of a valid memory address plus an interpolation offset, together which correspond to the unknown position of latitude and
longitude each. Such a combination will allow the implementation of the DMA of elevations, since the input position will simply be a valid memory address plus an interpolation offset.

Referring to Appendix E, the input latitudes and longitudes are each 31 bits wide and have an input resolution of $8.38e-8 = 180/2^{31}$ degrees. Compared to the lateral resolution of $1/1024$ degree of the terrain database, we get,

$$\text{Number of sub-database step levels } L = \left( \frac{1/1024}{8.38e-8} \right) = 11650.8 \approx 11651$$

$2^{13} = 8192$ and $2^{14} = 16,384 \Rightarrow 14$-bits are required for $L$

Therefore, $L=11651$ levels exist between each consecutive address in the terrain database having $(M \times M) = (256*256)$ addresses. At least 14 bits are required to represent $L=11651$ in binary. Figure 26 illustrates the interpolation offsets in one memory page and also shows the length of each data-word (Assuming an additional 1-bit for the two’s complement values of negative latitude and longitude inputs).
To ensure that the interpolation offset is always positive, the terrain database is organized such that the addresses, which increment from zero onwards, always correspond to an ascending order of latitudes and longitudes. This always makes the interpolation offset an increment when measured with respect to the address of the known position in the terrain database (shown in Figure 23 with arrow near Latitude column).
5.4.2 Input Block

As described earlier, the input block accepts the input data and pre-processes them in preparation for the computation block. The following functions are implemented in the input block:

1. Set all initialization parameters and test statistic values and initialize terrain database (SW requirement)

2. Accept input latitude, input longitude of point, GPS MSL height and radar altimeter AGL height (SW requirement)

3. Calculate Synthesized elevation = GPS MSL height - radar altimeter AGL height and round to integer values. (SW requirement)

4. Use a memory-addressing scheme to split the input latitude and longitude into valid memory addresses and their corresponding interpolation offsets. (SW requirement)
5.4.2.1 Memory addressing scheme

Table 2: Mapping a Latitude/Longitude pair into memory address

<table>
<thead>
<tr>
<th>Zero offset</th>
<th>+veLat (deg)</th>
<th>-veLon (deg)</th>
<th>Zero offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-180</td>
<td>0</td>
</tr>
<tr>
<td>.</td>
<td>1</td>
<td>-179</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>2</td>
<td>-178</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>40443</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>180*1024</td>
<td>180</td>
<td>0</td>
<td>-180*1024</td>
</tr>
</tbody>
</table>

Table 2A

<table>
<thead>
<tr>
<th>Lat_index</th>
<th>Lon_index</th>
<th>Address</th>
<th>Elevation (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1121</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>7822</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>2</td>
<td>4123</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>256-1</td>
<td>256-1</td>
<td>65535</td>
<td>8786</td>
</tr>
</tbody>
</table>

The memory address shown in Table 2B can be written as the following expression:

\[ \text{Mem\_address} = 256(\text{lat\_index}) + (\text{lon\_index}) \]  
\[ \text{------(5.4)} \]

where,

\( \text{lat\_index} \) = page index of latitude, \( \text{lon\_index} \) = page index of longitude

\( \text{mem\_address} \) = the address of the stored elevation for that latitude and longitude
In order for any memory address to correspond to a position of latitude and longitude, the equivalent of \textit{lat\_index} and \textit{lon\_index} should be calculated for the point in question. In this case, \textit{lat\_index} and \textit{lon\_index} simply indicate the latitude and longitude count of terrain database steps from zero reference. But since the first latitude and longitude stored in the terrain database in the first location are not zero latitude and zero longitude, a zero offset needs to subtracted from the memory address as shown in Eqn 5.5.

\[
\text{Mem\_address} = 256 \left( \text{Latitude in deg} / (1/1024) - \text{lat\_zero\_offset} \right) + \left[ \text{Longitude in deg} / (1/1024) - \text{lon\_zero\_offset} \right] \tag{5.5}
\]

Table 2A shows that the zero offset is nothing but the number of terrain database steps from zero reference. For longitude, the zero reference is taken as -180 deg, since all the input longitudes are negative. Since the first address in memory corresponds to the lowest value of latitude and longitude in the terrain database, this zero offset can be easily determined as:

\[
\text{Longitude zero offset} = \text{Lowest value of Longitude in terrain profile/ resolution}
\]

\[= -107.0556640625 * 1024 = -109625 \quad -----(5.6)\]

\[
\text{Latitude zero offset} = \text{Lowest value of Latitude in terrain profile/ resolution}
\]

\[= 39.4951171875 * 1024 = 40443 \quad ------(5.7)\]
Therefore, Eqn 5.5 can be rewritten as,

\[
    \text{Mem}_{-}\text{address} = 256[(\text{Latitude in deg} \times 1024) - 40443] + \frac{1}{\text{[(Longitude in deg} \times 1024) + 109625]}\]  

Also,

\[
    \text{lat}_{-}\text{index} = [(\text{Latitude in deg} \times 1024) - 40443] \] \quad \text{---------(5.9)}
\]

\[
    \text{lon}_{-}\text{index} = [(\text{Longitude in deg} \times 1024) + 109625] \] \quad \text{---------(5.9)}
\]

Thus, to create the modified terrain database, the terrain database latitudes and longitudes are converted using Equations 5.8 and Equations 5.9 to generate their corresponding memory addresses and the elevations of these points are stored in their corresponding addresses.

The assumptions for Equations 5.8 and 5.9 are:

1. All input values of latitude are positive & greater than or equal to 39.4951171875 deg.
2. All input values of longitude are negative & greater than or equal to -107.0556640625 deg.
3. Latitudes and longitudes (of input and database) are expressed in the range of:
   -180 to +180 deg, and NOT as 0 to +360 deg (if they are, then they must be changed to the former format)
5.4.2.2 Calculation of Interpolation offsets of input latitudes and longitudes

As far as the terrain database latitudes and longitudes are considered, there is no need to consider the 14-bit interpolation offset. But when the input data is received, this factor has to be taken into consideration. The memory addressing scheme makes it very easy for the input block to split the 32-bit latitude and longitude words of the input data into combinations known terrain database positions and interpolation offsets. From Eqn. 5.9,

\[
lat\_index = integer + fraction \rightarrow \text{for input latitude from flight path segment}
\]

\[
lon\_index = integer + fraction \rightarrow \text{for input latitude from flight path segment}
\]

The fractional parts represent the interpolation offset required. Therefore, for an input position, we have,

\[
lat\_page = \text{integer part}(lat\_index)
\]

\[
= \left[ \text{integer} \left( \text{Latitude in deg} \times 1024 \right) - 40443 \right]
\]  
------(5.10)

\[
lon\_page = \text{integer part}(lon\_index)
\]

\[
= \left[ \text{integer} \left( \text{Longitude in deg} \times 1024 \right) + 109625 \right]
\]

where,

\[
lat\_page \text{ = the page index for an input latitude,}
\]

\[
lon\_page \text{ = the page index for an input longitude}
\]
From Equations 5.9 through 5.10, the memory address can be written as,

\[
\text{mem\_address\_inp} = 256(\text{lat\_page}) + \text{lon\_page}
\]  

-----(5.11)

Hence, the memory address can be calculated for the input latitude and longitude by the input block, in SW, and sent to the computation block (FPGA).

The interpolation offsets, \(dx\) and \(dy\) are given by,

\[
dx = \text{lat\_index} - \text{lat\_page}
\]  

\[
dy = \text{lat\_index} - \text{lon\_page}
\]  

where,

\(dx = \text{interpolation offset, } dx, \text{ as described in Eqn 5.3 (Figure 25)}\)

\(dy = \text{interpolation offset, } dy, \text{ as described in Eqn 5.3 (Figure 25)}\)

In order to reduce the number of bits required to transmit \(dx\) and \(dy\) from the input block (PC) to the computation block (FPGA), \(dx\) and \(dy\) can be represented as integers by multiplying them with the database offset, L (11651) and rounding the result.

\[
dxsc = \text{rounded value (} dx \ast 11651\text{)}
\]  

\[
dysc = \text{rounded value (} dy \ast 11651\text{)}
\]  

-----(5.13)
where,

\[ dxsc = \text{scaled and rounded value of interpolation offset, } dx \]
\[ dysc = \text{scaled and rounded value of interpolation offset, } dy \]

When the values of \( dxsc \) and \( dysc \) will be required in the computation block, they are scaled back to the fractional representation by dividing \( dxsc \) and \( dysc \) by \( L \).

The next section describes the design of the computation block in detail.

### 5.4.3 Computation Block

The pre-processed inputs obtained from the input block are written to text files and saved on the PC. After the computation block completes its job, its outputs are also written and saved to text files on the PC. Based on the MPDLIM algorithm shown in Figure 9 of Chapter 4, and the low-level description of the algorithm, the behavioral description of the functionality is described using VHDL. A top-down design approached is used to translate the low-level description of the algorithm to a behavioral description. Each step of the algorithm is considered as a functional component that takes an input and gives an output. So, if each component is designed first and then all the components are connected together to build the complete functional design component in a bottom-up fashion, then the functionality defined by the sequence of operations as shown in the algorithm can be implemented. Figure 7 shows all the design components that are used to build the required logic block. (The legend for the components and other details shown in Figure 7 are explained in the
following sections of this chapter). This type of design is hierarchical and has some advantages over non-hierarchical design. A very significant advantage is that it is much easier to recognize and isolate bugs in the design, which facilitates the verification process of the design. In the presence of real receiver inputs, dotted block arrows in Figure 27 show the interface between the HW component for SVS TDIM and such inputs.

5.4.3.1 Dataflow

The description for each of the functional components shown in Figure 27 is as follows:

1. \( A1A2 \): Adds the translation offsets to the input memory address, using Equation 5.15 and generates the two new addresses of the translated points.

2. \( B1B2 \): Accepts the two new addresses from component \( A1A2 \). Increments the latitude and longitude page indices to obtain the memory addresses of the four closest neighbors of each of the two translated points in the terrain database.

3. \( C \): Accepts the input memory addresses from component \( B1B2 \) and reads the corresponding elevations from memory. In this case, for every translation, component \( C \) reads the text file containing the terrain database elevations into the Aldec Active-HDL™ environment, to obtain the required elevations.

4. \( D1D2 \): Accepts four input memory addresses and two interpolation offsets for each point from component \( C \) and interpolates among the four memory addresses to find the terrain database elevations. All outputs are rounded to integers for simplicity.
5. **E1E2**: Accepts the synthesized elevations from input data and the terrain database elevations from component D1D2 to compute the ADs for each point. Since the terrain database elevations are integers, the synthesized elevations are also integers, which have been pre-processed by the input stage.

6. **F1F2**: Scales each of the computed AD obtained from E1E2 by the Standard Deviation of AD which is 18.9, but a value of 19 is assumed for simplicity. Only the integer values of the results are considered (this is not equal to rounding to integers), i.e. in this case, the integer values are always rounded towards zero and not towards $+\infty$ or $-\infty$.

7. **G1G2**: Squares each of the scaled AD value to obtain the corresponding $T$-values (Threshold value).

8. **H**: Accepts the latest $T$-value and the corresponding array indices for the first point. It reads the $T$-value last recorded at the location in the $T$-value array specified by the array index, adds it to the latest $T$-value and stores it back at that location in the array. The control signals are used to determine the correct array indices, since the translations performed correspond to the specific locations in the array. After the first $T$-value is recorded, this procedure is repeated for the second $T$-value obtained from component G1G2 as well. After the translations for all the points in the path segment are complete, all the latest $T$-values from the $T$-value array are written to a text file on the host PC.

9. **I**: Accepts the latest $T$-value for each point and updates a copy of the $T$-value array obtained from component I. Only when the $T$-values for all the points in the chosen
path segment of the input data have been computed, this functional component is designed to find the minimum of the \( T \)-value array. It also finds the location of the minimum \( T \)-value, in terms of an offset measured in terrain database steps. Values below the threshold value (which is 96) are considered to be zero and an array is created to store these below \( T \)-values as well. After the translations for all the points in the path segment are complete, all below \( T \)-values, the value and the location of the minimum \( T \)-value are written to a text file on the host PC.

Although requirements on the ranges of the inputs and outputs for every functional block have not been specified, assumed ranges are still used during simulation. This can help test for any out of range values and also in synthesis, because only the required number of bits would be assigned for each parameter.
Figure 27: Design components of Computation Block
5.4.3.2 Control Unit

The “looping” requirement that is derived from the position translations can be modeled in VHDL by introducing a separate control unit. As shown in Figure 27, the control unit consists of a master control unit and a set of three counters. The master control unit generates the master control signals namely, clock pulse, reset, enable and any other derived control signals. The counters, CNTR1, CNTR2 and CNTR3, are used to generate the latitude, longitude translation offsets and the number of points in path segment respectively. All these control signals are supplied to each functional component in the design.

The segregation of functional components and the control unit allows either of them to be changed independently. Since the control unit acts as the feedback component between the first step and the last step of the algorithm, it represents sequential logic. Sequential logic is logic in which the output of the current state is determined by the input of the current state and the output of the previous state. The functional components are referred to as combinational logic; the output of the current state is determined only by the input of the current state. In an RTL model, the functional components do not have any control logic embedded inside them and the control unit is wholly responsible for the entire control of all the functional components. Since the VHDL behavioral description, that is implemented based on the low-level description, is not an RTL model, each of the functional components has some common, control logic embedded inside them. Nevertheless, the idea is to restrict
as much control logic as possible to the control unit alone- the control unit itself and each functional component have been designed keeping this idea in mind.

5.4.3.2.1 Scheme to Generate Translation Offsets

As described in chapter 2, the input latitudes and longitudes need to be translated from the center point (that is simply the first of the points in the chosen path segment) and MSDAD values calculated for each of these translations for each point of the path segment. In order to generate the translation offsets for this offset square, one method would be to generate each offset sequentially, using a loop construction, and perform a translation for that value of offset. A better alternative is to generate more than one translation offset per iteration of the looping construct and perform more than one translation in parallel for every run of the loop or counter. Figure 28 illustrates the concept used to implement alternative scheme.

It is observed that all the points in opposite, diagonal halves of the MSDAD sample grid shown in Figure 28, correspond to complimentary latitude and longitude values. This means that a point B (3,0) is simply A (0,3) with the latitude and longitude values swapped for each other. Therefore, if all the offset values for one diagonal half of the offset square are generated, then the translations for the other diagonal half of the offset square can be performed simultaneously, simply by swapping the latitude and longitude offsets.
\( A(\text{Latitude}) = A(\text{Latitude}) + \text{latitude offset} \)

\( A(\text{Longitude}) = A(\text{longitude}) + \text{longitude offset} \)

\( B(\text{Latitude}) = B(\text{latitude}) + \text{longitude offset} \)

\( B(\text{Longitude}) = B(\text{longitude}) + \text{latitude offset} \)

In terms of the addresses of the elevations stored in memory, this implies that,

\( A(\text{address}) = A(\text{address}) + [256 \times \text{latitude offset}] + [\text{longitude offset}] \)

\( B(\text{address}) = B(\text{address}) + [256 \times \text{longitude offset}] + [\text{latitude offset}] \)
Thus, two translations are performed simultaneously, using only one set of latitude and longitude offset values. Figure 27 shows the different sets of inputs as A1, A2, B1, B2, C1, C2 and so on. In terms of HW, the same functionality is implemented, but in the scheme mentioned here, there are two distinct advantages gained. First, the number of iterations required is reduced by a factor of approximately two, which means that the simulation time is reduced as well. Second, although the device resources that would be used in HW to implement such a scheme are doubled, they would still be used simultaneously. This means that the parallel processing capabilities of the HW device are exploited and higher processing speed can be achieved.

5.4.3.2.2 Control Logic for Generation of Translation Offsets

After designing the translation offset generation sequence this sequence must be generated using control logic. In HW, this means that a counter is required to loop through the values of the offsets required for each translation. Since the values for the latitude and longitude offsets and the number of points are known (say, -20 to +20 and 50 respectively), these values of can be stored as array constants in VHDL. The latitude (say, \textit{lat}), longitude (say, \textit{lon}) and the number of points in the input path segment (say, \textit{nsize}) are constants (arrays whose values are constant in a simulation cycle) with individual array indices (say \textit{mlat}, \textit{mlon} and \textit{mnsize} respectively).

Table 3 shows the array indices for \textit{nsize, lat} and lon. For each run of the loop, the corresponding indices for the required values of the offsets are picked. Therefore, the combination of array indices for the first run of the counter will be \textit{mnsize, mlat} and
Table 3: Generation of control logic using arrays and indices

<table>
<thead>
<tr>
<th>Array indices</th>
<th>nsize</th>
<th>lat</th>
<th>lon</th>
<th>For every run of loop</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>msize</td>
<td>mlat</td>
<td>mlon</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>-20</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>-19</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>18</td>
<td>18</td>
<td>19</td>
<td>-18</td>
</tr>
<tr>
<td>...</td>
<td>19</td>
<td>19</td>
<td>20</td>
<td>-18</td>
</tr>
<tr>
<td>...</td>
<td>19</td>
<td>19</td>
<td>20</td>
<td>19</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>49</td>
<td></td>
<td>50</td>
<td>49</td>
<td>19</td>
</tr>
</tbody>
</table>

\( mlon \) equivalent to 0, 0 and 0 respectively. For the next run, \( msize \) will be 0, \( mlat \) will be 0 and \( mlon \) will be 1. In this way, only the array indices have to be manipulated and the actual values of the offsets of need not be changed in the simulation. Also, declaring more constants in VHDL significantly reduces the simulation time of the design [34].

The sequence of operations of the functional components is designed such that only one component is enabled at one time, for a particular translation. This means that component \( B1B2 \) is not enabled for a particular translation until \( A1A2 \) has finished its computation, and enables \( B1B2 \). This is shown by the control signals between each component in Figure 27. Also, once \( A1A2 \) has finished its computation for a translation, it is not enabled until the next translation is performed. Thus, when component \( I \) has finished its computation, it informs the control unit. The control unit then initiates a
reset-like action to disable all the components before the next translation is performed. This enabling and disabling of components is implemented inside each individual component. The functional description for that component is embedded within some of this control logic, whereas, the reset after each translation is monitored and initiated by the control unit alone.

5.4.3.3 Functional Components in VHDL

5.4.3.3.1 How to describe a functional requirement in VHDL?

Now that the functional requirements, inputs and outputs for each component have been identified, the behavior of the HW that would implement these requirements needs to be described. The very first step is to be able to identify the flow of data between components. Section 5.4.3.1 gave an idea of this dataflow. Based on this dataflow, the transformations that take place from the input to the output of a component define the logic circuit required to make those transformations. For example, component $A_1A_2$ accepts inputs, and adds them using Eqn 5.15, therefore, we can conclude that at least one adder and multiplier is required. It is important to think about the HW that will be synthesized based on the VHDL description. In this respect, HDLs are very different from SW languages, in the sense, that HDLs describe the behavior of the HW that will be used for implementing the required functionality. For e.g. an expression $(a+b) + c$ would give the same results as $(a)+(b+c)$ in SW. But from a synthesis perspective, the former expression would result in $(a \text{ OR } b)$ first and then $(a$
OR b) OR c, as compared to (b OR c) and then (b OR c) OR (a) for the latter expression.

Each functional component is described as an entity-architecture pair. Entity refers to the interface between the actual HW and its environment [34]. It defines all the ports (signals) and generics (static parameters) that interface with its environment. Architecture refers to the description of the behavior or the structure of the HW component to be designed. An entity can have multiple architectures, but architecture can have only one entity. Based on the architecture, the VHDL description can be structural or behavioral. A behavioral architecture can be specified using the process statement to model real HW. Thus, each functional component described in the dataflow can be considered a process with embedded processes. A process always waits for a specified event to occur in time. Once that event occurs, the process is resumed and it executes all the statements that are inside that process and gets suspended again. All outputs are updated once the process is suspended. Whenever the specified event occurs again, the process resumes again. A good analogy is a vending machine or a washing machine. The event that resumes the process is specified in a sensitivity list for the process. All processes that are sensitive to a common event will be resumed when that event occurs. Shown below is an example of VHDL code for an AND operation:
library IEEE;            \Comment{standard library}
use IEEE.STD_LOGIC_1164.ALL; \Comment{entity}
entity andgate_ent is    \Comment{constant parameter}
generic(
    cin: bit := '0');
port(
    RST : in bit;
    CLK : in bit;
    ain : in bit;
    aout : out bit);
end entity andgate_ent;
architecture andgate_beh of andgate_ent is    \Comment{architecture}
signal en: bit := '1';    \Comment{internal signal}
begin
P1:process(CLK,RST)            \Comment{sensitivity list}
variable ainv, aoutv: bit;
begin
    if RST = '1' then
        aoutv:='0';
        aout <= aoutv;
    elsif CLK = '1' then
        aoutv:= ((cin) and(ain)) and(en);
        aout<= aoutv;
    end if;
end process P1;        \Comment{return to start of process & wait for event}
end architecture andgate_beh;

The most important concept associated with processes in real-life and in VHDL too, is concurrency. In a system with multiple processes, like the functional components shown in Figure 27, all the processes are executed concurrently. For example, a computer can have multiple processes running on it at a given time, such as playing music files, writing to a disk, and performing a simulation simultaneously. This means that the VHDL description should also take this concurrency of processes into account.
With respect to the functional components described for the TDIM of SVS, if we consider all the components to be sensitive to the control signals, then it means that all of these components are resumed for every change in say, the clock pulse or a reset signal from the master control. Therefore, if only one component needs to be enabled at a time, then additional control logic is necessary inside each component to ensure that only the desired component is completely enabled. In the RTL model, however, the control unit handles all the control and registers in between consecutive functional components are used to register the inputs and the outputs. Thus, RTL model is the basis for a pipelined implementation of the behavioral VHDL description. Based on the VHDL behavioral description of each functional component, an RTL model is proposed in Chapter 7.

5.4.3.3.2 Hierarchical Design of HW Component for TDIM of SVS

Each component that was designed as per the individual functional requirements is now ready to be joined with all the other components to build the logic circuit that will implement the functionality of the HW component for TDIM of SVS. It is important to simulate and test each individual component before proceeding with the other components. If this verification step is not performed it will become very difficult to pinpoint the faulty component or any other source of an erroneous output during simulation of the final design. Thus, each component is first simulated for functional behavior, then it is simulated with the control logic for a single iteration to verify the results and finally it is simulated for all the iterations as specified in the control logic.
This procedure is followed for all the components from component $A_{1A2}$ through component $I$. Thereafter, all these entity-architecture pairs (functional components) are instantiated as components (VHDL construct) and declared in a package in a library. The advantage of declaring components in a package is that the entities described in the package are visible to all designs using this entity.

A high-level VHDL description is generated to specify the how all the components should be connected, using port-maps and generic maps (VHDL constructs). This type of bottom-up, VHDL structural description is called hierarchical HW design, which is illustrated in Figure 29.

![Figure 29: Hierarchical HW Design](image-url)
The structural description of all functional components is used to create a component that represents the HW component for TDIM in SVS, without any control. In order to generate the control signals for these components, a control unit is designed using the methodologies described in Section 5.4.3.2. It is simulated as a standalone unit to check if the correct control signals are generated or not. The final step is the structural description that joins the control unit and all the functional components. This is the HW component required for TDIM for a SVS. Again, it is simulated to test the outputs and check for any errors.

5.4.4 Output Block

After the computation block has performed all its computations, the minimum $T$-value, location of this minimum $T$-value in the translation offset square and the below threshold values are stored in text files on the PC. These files are read back into Matlab™ to generate plots to compare the outputs from the VHDL simulation with the values of the expected results in SW. This block is used to compute the location of the minimum $T$-value in the latitude and longitude profiles of the terrain database that are stored in SW.

The following chapters discuss the simulation results and discuss the verification and validation processes for SW/HW integrated avionics systems, using the SVS TDIM design as a case study.
Chapter 6 Verification and Validation of SW/HW Integrated Systems: FPGA-based TDIM for a SVS, a Case Study

Chapter 3 described the SW and HW certification processes based on RTCA/DO-178B and RTCA/DO-254 respectively. Since HW/SW co-design was employed additional considerations are required for both SW and HW components are applicable to the HW component designed for of the SVS TDIM.

6.1 Concerns for Certification of Complex Electronic Hardware (CEH)

Since the HW design for the SVS TDIM is FPGA-based, it is necessary to investigate the concerns for CEH in the certification process. For a system consisting of only SW or HW components operating independently, the guidance for design assurance of that system can be obtained directly from RTCA/DO-178B and RTCA/DO-254. However, for SW/HW integrated systems, there is no specific guidance available from the FAA for the certification of such systems.

The major concern in CEH is that a HW component is programmed using SW, thus creating an interface that allows the HW/SW functional failure paths to overlap. This makes it very difficult to test, verify and validate the SW and HW components independently. This means that the verification and validation requirements for the system cannot be met sufficiently. In the case of an FPGA, the HDL is used to model the behavior of the FPGA for a given functionality. Due to the nature of FPGA design process, any malfunction in the HDL, the design tool, the development tool, the synthesis tool or the FPGA device itself will translate into an incorrect configuration of
the FGPA. The difficult part is to determine where exactly the malfunction originated from and how the failure propagated through the complex electronic circuit and affected the system’s behavior. For a safety-related avionics application, like SVS, such uncertainty of system behavior is not acceptable. This is one of the reasons why even today, simple electronic HW may be preferred over CEH in safety-related avionics systems, in spite of the performance enhancements that CEH components offer. Nevertheless, due to the increasing complexity of electronic circuits and dependency on CAD of electronic HW today, it is inevitable that this issue will need to be addressed sooner or later. Hence, it becomes necessary to address the concerns and possible solutions to verify CEH used in safety-related systems in avionics.

6.2 RTCA/DO-254-Based Structured Design Approach for PLDs

RTCA/DO-254 provides some recommendations for design assurance guidance for PLDs, such as FPGAs. It states that a basis for certification can be established, provided a structured design approach was used for designing the PLD. Table 4 maps typical ASIC/PLD processes to the processes stated in Chapter 3 for certification of HW components [12]. As long as the PLD has been designed by using a structured design approach, which allows the PLD processes to be mapped to their corresponding certification processes, it is possible to establish a basis for certifying that PLD. Thus the design process of the HW component for an FPGA-based TDIM should also follow the sequence of the certification processes shown in Table 4.
In this thesis, the HW/SW component that was designed for a SVS TDIM went through design processes that can be considered analogous to the requirements capture, conceptual design, detailed design and implementation stages. Although the FPGA
design process was implemented only to the extent of functional VHDL simulation, the same processes can be extended for the rest of the FPGA design process as well.

6.3 Verification and Validation of PLDs

6.3.1 Validation

HW validation is required to ensure that all the requirements allotted to the HW component are correct and complete with respect to the system requirements. For a safety-related design such as SVS, the derived requirements should be validated especially with respect to the safety requirements. For an FPGA used in a hierarchical design, as described in Section 5.4.3.3.2 of Chapter 5, it implies that the requirements for each functional component specified in VHDL must be traceable and compliant with the system requirements as stated in Section 4.6 of Chapter 4. To ensure this, each requirement must be validated at each hierarchical level by review, analysis or test. In the design for the HW component for TDIM for an SVS, a similar procedure was followed.

First, the requirements for each functional block were derived from the system requirements and accordingly, the VHDL code was generated. For the next hierarchical level (say the structural VHDL description for components $A1A2$ through component $I$), the requirements for this level were derived with respect to the system requirements. It is important to note, “Validation completion criteria may be based on requirements, safety considerations, operational mode or implementation [12]”. HW validation objectives may be satisfied through a combination of activities such as simulation,
reviews, modeling and tests. Although validation may be carried out before or after the design process, it is generally carried out throughout the entire design life cycle.

### 6.3.2 Verification

Verification is an evaluation of the implementation of the requirements to determine that they have been met [12]. Verification is an integral process of the design life cycle process. This means that no design can be considered complete unless the design is verified. For safety requirements, especially for safety-related designs, it is advantageous to apply the verification process at various stages of the design process to increase the probability that design errors have been eliminated. Figure 4 of Chapter 3 shows that verification is done after every step in the FPGA design process to ensure the same. In the hierarchical HW design, each functional component is simulated individually before proceeding to the next level.

Verification can be done using tests, analysis (including simulation analysis), and reviews. “As the complexity of the HW design increases, it is advantageous to make use of computerized tools, such as simulation to verify requirements and implementation of the design [12]”.

According to RTCA/DO-254, “HDL design representations use coded text based techniques that are similar in appearance to those used for SW representations. This similarity in appearance can mislead one to use SW verification methods directly on the design representation of the HDL.” Again, it is important to remember that SW code and HDL represent different quantities, so all the SW verification methods may
not be applicable directly to HDLs. One possible solution is to verify the HDL using formal methods, which is a mathematical approach to verifying HDLs. Based on the requirement specification; a formal model of the component to be analyzed is constructed using a formal language [12]. This model is analyzed using formal logic to either check that no errors in the design exist or that certain classes of design errors are non-existent in the design. This model is treated like a theorem that needs to be proved to satisfy a certain condition [35]. Prototype Verification System™ (PVS) is a formal methods- based verification system that can be applied to verification of HDLs.

Most SW and HW component vendors are aware of the problems associated with verifying HW/SW integrated systems. As a result, design tools, which have HW and SW verification environments in the same tool, are available today. These tools allow for the simultaneous verification of HW and SW components. For e.g., Aldec Active-HDL™ has a Matlab™ interface available for performing HW and SW simulations of the VHDL description simultaneously. Other popular FPGA vendors such as Synopsys™ and Xilinx™ also support HW/SW co-verification tools for simultaneous debugging of the HW/SW interfaces.

6.4 Additional Considerations

6.4.1 Tool Qualification

Tool qualification is a very important factor that needs to be considered for certification of CEH. This is because the design of FPGA components is very much dependent on the design and development tools that are used in the design process. The
development tool needs to be qualified for the HW level [12] or SW level [14] as specified by the system requirements. RTCA/DO-178B and RTCA/DO-254 provides detailed guidance for tool qualification. For the HW component that was designed for TDIM for a SVS, Aldec Active-HDL™ and Matlab™ can be considered to be design and verification tools, since they are mainly used to describe and simulate the design, and need not be qualified to any level. But a synthesis tool is a development tool, which can introduce errors in the system design, and needs to be qualified as per the system requirements.

6.4.2 IP Cores

IP cores refer to previously developed SW or HDLs that can be used for a specific functionality in any design. They are available in the vendor libraries that are installed along with the design or development tool. In Matlab™, this would represent specialized functions or using a library of user-defined functions. In HW synthesis, IP cores usually refer to components that are instantiated in the design that requires the functionality defined by the core. E.g., if a counter is required in the synthesis tool, the core that contains the behavioral description for it can be selected and this core can be used in the design.

IP cores are good for making HW or SW design efficient, but from a certification perspective, they are not always the best choice. Although IP cores are tested and made as error-free as possible by the vendor, their description is not provided to the user for copyright reasons. This means that only the behavior can be simulated,
but the core cannot be tested individually. For a safety-related design like SVS, it is worthwhile to compromise on efficiency as compared to safety. Keeping this factor in mind, the entire HW component for TDIM for a SVS has been designed without using any IP cores in VHDL or previously developed SW code.

### 6.4.3 COTS Components

This factor is probably most significant for the certification of the TDIM for a SVS. The FPGA device is a COTS component that is used in the HW design of a TDIM for a SVS. As is the case with most COTS components, the design data of the FPGA device is not available for review from the vendor. This means that the COTS component needs to be verified along with the system, through the overall design process. The basis of certification of COTS components can be established by the use of an Electronic Component Management (ECM) process in conjunction with the design process [36]. In order to gain certification credit for the COTS component, the manufacturer participates in an interactive process with the designer to provide the basis for certification. If the component manufacturer is able to demonstrate that the COTS component has high reliability, assured by quality control and good service experience supporting its successful operation, then certification credit can be granted to the designer using the COTS component.

In a safety-related design such as TDIM for SVS, these components should be kept to a minimum or purchased from the same vendor. This way, the ECM process for
certifying the COTS component can be initiated and a basis for the certification of the COTS component can be established.

6.5 TAD PLD Generic Issue Paper

Due to the growing number of vendors and verification standards of each vendor, the Transport Airplane Directorate (TAD) has issued a PLD Generic Issue paper to deal with the certification of ASICs and PLDs. Appendix D shows the Generic Issue Paper [13].

This issue paper states separate requirements for simple and complex electronic HW devices. The reason this issue paper is called generic is because it has can be used by any model and any vendor in question. Each vendor supplies the details of the PLD and states the avionics application that it will be used in. If the HW device is simple, then the device can be verified by tested to the appropriate level (gate or pin). If the device is complex (like an FPGA), a rigorous, structured development process commensurate with the risk should be followed. The following guidance is provided:

“Programmed Logic Devices shall be developed using a structured development approach approved by the FAA. The structured approach should provide design verification which achieves the same result as that provided for software development by RTCA document DO-178B”[36].
When applied to VHDL this guidance means that HDLs can be verified and tested using SW design assurance guidance, provided that a structured development process is followed. Thus, for a SVS TDIM, certification basis can be formed if the HW component for the SVS TDIM is developed using a structured and rigorous approach, commensurate with the safety requirements of the system.

The next chapter discusses the results of the HW design and provides discussions for these results.
Chapter 7  Results and Proposed RTL Model

7.1  Simulation Details

The SW and HW simulations were computed for the following cases, as shown in Table 1:

Table 5: Simulation Details for all Cases

<table>
<thead>
<tr>
<th>Case for simulation</th>
<th>Description of Algorithm</th>
<th>Resolution of Terrain Database used</th>
<th>N</th>
<th>σ  (meters)</th>
<th>T_m</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High-level Matlab™</td>
<td>3-arcs</td>
<td>50</td>
<td>18.9</td>
<td>96</td>
<td>Basic reference</td>
</tr>
<tr>
<td>2</td>
<td>Low-level Matlab™ and VHDL</td>
<td>3.515625 arcs</td>
<td>50</td>
<td>19</td>
<td>96</td>
<td>Equivalent</td>
</tr>
<tr>
<td>3</td>
<td>Low-level Matlab™ and VHDL</td>
<td>3.515625 arcs</td>
<td>4</td>
<td>19</td>
<td>96</td>
<td>Equivalent</td>
</tr>
<tr>
<td>4</td>
<td>Low-level Matlab™ and VHDL</td>
<td>3.515625 arcs</td>
<td>1</td>
<td>19</td>
<td>96</td>
<td>Equivalent</td>
</tr>
</tbody>
</table>

For the VHDL simulations, a clock pulse frequency of 1 GHz was chosen. Since a low-level VHDL description is simulated, the frequency of the clock pulse does not matter to the functionality because no timing constraints are being simulated. The purpose of the clock is only to model the asynchronous reset and rising clock edges in the circuit. The corresponding values for N, σ and T are reinitialized for every new case. After each simulation, the values of T, the location of the minimum value and below T-values are written to text files on the PC.
Matlab™ plots of terrain database elevation profile, $T$-values showing the minimum $T$-value and translation offsets, and threshold value plots were generated for all the cases mentioned above. All the outputs of the VHDL simulations were saved as text files on the PC and read into Matlab™ to compare the VHDL outputs and to generate the Matlab™ plots. Another way of checking that the output of the Matlab simulation and VHDL one is to use SW that compares text files, such as Windiff™. Since all the Matlab™ simulations exactly matched those obtained from VHDL, a single Matlab™ plot of the outputs for each case is shown. All latitude and longitude offsets are in steps of terrain database resolution i.e. 3 arc-s steps for the Eagle/Vail, CO terrain database and 3.515625 arc-s steps for the modified terrain database. On each terrain database plot, the flight path and the path segment are shown. On the $T$-value plot, the minimum $T$-value is indicated by a red dot at the location of the minimum.

The latitude and longitude values are displayed in Matlab™ using 10 digits of decimal precision to represent the location of the minimum $T$-value in the modified terrain database.
7.2 Results

7.2.1 Case 1

Figure 30: Plots for High-Level Description of Algorithm
Figure 30 shows the results for Case 1 (The plot of the terrain database elevation profile and the $T$-value plots are the same as those in Figure 10 and Figure 14 in Chapter 4 The data depicted in Figure 10 represents an area in the vicinity of Eagle-Vail, CO). The Eagle-Vail airport or EGE is located right above the middle of the plot showing the terrain database profile with flight path in Figure 30. The minimum $T$-value plot in Figure 1 shows the plot of the $T$-values and the location of the minimum $T$-value. The results for Case 1 are:

1. Minimum $T$-value = 15.968
2. Latitude offset = 0 and Longitude offset = -1

Again, the blue regions in the threshold plot in Figure 1 indicate the area where a bias would not be detected by the radar altimeter-based TDIM. Ideally, these blue regions should be made as small as possible around the center point. The units for the elevations are in meters. (Note that the scaling of color in every individual plot is not always the same and varies in every plot, depending on the minimum and maximum plotted values).
7.2.2 Case 2

Figure 31: Terrain Database Plots for Low-Level Description
Figure 31 shows the elevation of the modified terrain database, which was created by interpolating from the 3-arch second database. This region is shown by yellow and red background in the Modified terrain database plot in Figure 31. This database has 256 points latitude and longitude each, out of which 171 are elevations of points obtained by interpolation from the 3 arcs terrain database. This is illustrated in Figure 31 with the modified terrain database containing dark blue regions that correspond to the zero values. The interpolated elevation region is shown along with the flight path in green and the chosen path segment in red in the lower plot in Figure 31.

One important point is that the search of the below-$T$ values is achieved by sorting between the current and previous values of the $T$-value array. This means that if the $T$-value arrays were initialized with zero and their values never changed, and then the minimum that will be found will always be zero. Also, the location of the minimum of the $T$-value is simply the first zero that is encountered - even if there are more than one elements of $T$-array with zero values.
Figure 33 shows the plot of the $T$-values and the location of the minimum value of $T$. The results for Case 2 are:

1. Minimum $T$-value = 8.000
2. Latitude = 39.6132812500 deg, Latitude offset = 1
3. Longitude = -106.8818359375 deg, Longitude offset = -1
7.2.3 Case 3

Figure 34 shows the plot of $T$-values and the location of the minimum value of $T$, for 4 points the path segment, along with its location. The results for Case 3 are:

1. Minimum $T$-value = 0.000
2. Latitude = 39.5996093750 deg, Latitude offset = 15
3. Longitude = -106.8828125000 deg, Longitude offset = -2

Figure 33: Plots for Low-Level Description of Algorithm for Case 3
7.2.4 Case 4

Figure 34: Plots for Low-Level Description of Algorithm for Case 4

Figure 35 shows the plot of $T$-values and the location of the minimum value of $T$, for 1 point the path segment, along with its location. The results for Case 4 are:

1. Minimum $T$-value = 0.000
2. Latitude = 39.5947265625 deg, Latitude offset = 20
3. Longitude = -106.8632812500 deg, Longitude offset = 18
7.2.5 Final Results

Thus, by comparing the output of the HW and expected results, we can conclude that the design techniques and schemes employed in SW were successfully simulated in HW. The assumptions made in the process, i.e. using only integers, etc. were incorporated for simplicity and to check the functionality of the HW component of a TDIM for a SVS alone.

7.3 Proposed RTL Model

In the HW simulations performed, the path delay (time taken for an input to be processed to output) from Component \( A1A2 \) to Component \( I \) as shown in Figure 27 of Chapter 5 was found to be 12 clock pulses, i.e. 4 for the translation reset and 8 for the path delay, counting 1 clock pulse for each of the 9 functional components. Since only one component is enabled at one time in a translation, it takes at least 12 clock pulses before the next input can be processed. This estimate of path delay is only in simulation time; in real HW, there would be a path delay associated with each functional block as well. Therefore, the speed of such a design is greatly restricted. For a high input data rate application of a TDIM for a SVS, the designed HW component for a TDIM for a SVS would not be able to meet the timing requirements of the design and the very purpose of HW/SW co-design would not be served. In order to overcome this restriction, a pipeline-based RTL model for TDIM is proposed.

Pipelining is an implementation technique in which multiple instructions are overlapped in execution [37]. In the HW component designed for a SVS TDIM, the
RTL model is required to implement pipelining. As stated earlier in Chapter 5, RTL logic consists of registers and functional blocks that are controlled by a control unit. Figure 35 shows the proposed RTL model of the HW component that was designed for a SVS TDIM. Input and Output registers are inserted between each functional component. An Input and Output Buffer is added at the input and output stage of the pipeline respectively. These I/O buffers accept the data and store it till it needs to be sent out in a burst.
Figure 35: Proposed RTL Model of HW component

Figure 36: Pipelined stages of RTL Model
Once data is written to a register, it stores this data until new data is re-written to it. The control unit generates the control signals that enable the registers to be written to or read. Therefore, all the functional blocks are able to operate simultaneously. In the non-pipelined design, each functional component had some control embedded inside it, which is not the case in a pipelined implementation.

As far as the performance improvement with respect to execution time is concerned, we can write,

If, \( P \) is the execution time for each pipeline,

\[
E_{np} = \text{Execution time for non-pipelined design of } N \text{ points} = P \times N \quad \text{(7.1)}
\]

Therefore, we have,

\[
E_p = \text{Execution time for pipelined design} = P + (N - 1) \quad \text{---(7.2)}
\]

\[
E_{diff} = \text{Difference in execution time} = P \times N - [P + (N - 1)] \quad \text{---(7.3)}
\]

\[
= (P - 1)(N - 1)
\]

If, \( F \) is the performance factor, we have

\[
F = \frac{P \times N}{P + (N - 1)} = \frac{1}{\left[ \frac{1}{N} + \frac{1}{P} - \frac{1}{NP} \right]} \quad \text{---(7.4)}
\]

\( (N \text{ and } P \text{ are always }>0) \)
Thus, $F$ is high, when both $P$ and $N$ are high. This means that for a large number of inputs $N$ (it can also be considered to be $N$ from the TDIM perspective) and a long execution time, a higher performance gain can be achieved.
Chapter 8  Summary, Conclusions and Recommendations

8.1  Summary and Conclusions

This thesis described the concept of terrain database integrity monitoring for a SVS followed by the required functionality to implement a TDIM for a SVS. Based on this system functionality, the design of HW and SW components for an FPGA-based TDIM for a SVS was contemplated. Employing a HW/SW co-design methodology improves performance and provides a better utilization of the resources of the FPGA device. This methodology can be used to enable parallel processing of the input data and therefore, increase the processing speed.

After the summary of the certification process for safety-related avionics systems in general, the SW and HW certification processes based on RTCA/DO-178B and RTCA/DO-254 respectively were summarized. This summary helped to gain an insight into the certification aspects, including the additional considerations that applied to the HW and SW components designed to implement an FPGA-based TDIM for a SVS.

A memory-addressing scheme for the terrain database was designed that allows input latitudes and longitudes to be mapped into memory addresses. The SW component on the input side was used to implement this memory addressing scheme, which greatly reduced the number of bits transmitted to the FPGA device, thus making for a very efficient system design. Based on the functional requirements for the HW component, individual functional components and control logic were designed to optimize the parallel processing of data. A pipeline-based HW design, which helps to
better exploit the parallel processing capabilities of an FPGA device, was proposed for improved performance.

The certification concerns for CEH and integrated HW/SW avionics systems in general, were listed with emphasis on the verification and validation processes. The FPGA-based TDIM for a SVS was used as a case study to examine these concerns and suggest possible remedies, for e.g., using a structured design approach to facilitate the certification process.

8.2 Recommendations for Future Research

In light of the fact that the FAA has mandated the inclusion of TAWS on all turbine-powered aircraft carrying six or more passengers by 2005 [39], this thesis is significant for HW/SW integrated avionics systems such as Synthetic Vision Systems that serve as TAWS. Given the increasing dependence on CAD of electronic HW in avionics systems, the issues of design assurance guidance for CEH, especially for HW/SW co-verification must be addressed in detail. SAE standard Aerospace Recommended Practice (ARP) 4754 [40] may be investigated to obtain additional guidance for complex aircraft systems such as an FPGA-based TDIM for a SVS as described in this thesis.

From a certification perspective, the definitions of HW and SW need to be re-visited for avionics systems with CEH. Although CEH, such as an FPGA is basically a HW component, a HDL -a SW-like description- is used to configure it. Thus, it
becomes necessary to re-define HW and SW clearly, so that CEH can be addressed unambiguously for certification in avionics systems.

The HW design of an FPGA-based TDIM for a SVS as described in this thesis is limited to satisfying the functional requirements of the system. This design needs to be re-evaluated for a real-time implementation using finite precision of data. This HW design also needs to be assessed for meeting the safety requirements of the SVS. The safety requirements depend on the intended function of the SVS in an aircraft. The intended function of a SVS from a certification perspective is currently limited to advisory applications alone. However, if SVS-related research progresses to extent where a SVS can be designed for safety-critical applications and still be certifiable, the safety requirements of the system and the overall HW/SW design will need to be re-evaluated accordingly. In other words, the level to which the SVS would need to be certified would change - e.g. from level C to level A. Methodologies for safety assessment processes outlined in SAE standard ARP 4761 [41] may be referred for guidance on Aircraft Level Safety Assessment.

The legal aspects of a SVS with an FPGA-based TDIM, which will be used in a safety-critical application in an aircraft, also need to be considered while performing the SSA of the SVS.

The HW/SW design techniques, such as efficient memory addressing and optimization of control logic, and the terrain database integrity-monitoring concept itself may be extended to other non-SVS applications that utilize terrain databases. To
name a few, such applications may include cartography, geographical surveying and Geographical Information System (GIS) applications.
REFERENCES


    <http://www.rtca.org/aboutrtca.asp>

    Federal Aviation Administration, April 23, 1997.

    254, April 19, 2000.


    Standards, Software Design and Certification, Ohio University, Athens, OH, Handout #6,
    Spring 2002, pp.3.

    Design and Certification, Ohio University, Athens, OH, Handout #4, Spring 2002, pp.2.

    Airplanes,” Advisory Circular AC23.1309-1C, December 3, 1999,

    Accuracy Improvement using Digital Terrain Elevation Data With an Inertial Navigation


[24] Flight test data for Figure 3 and Figure 7 obtained from Runway 07 Eagle/Vail, Colorado by Ohio University, August 21- September 7, 2001.

   <http://avsp.larc.nasa.gov/pdfs/csrp7.pdf>

[26] Chang, M., Programmable Logic, February 3, 2004,
   <http://avsp.larc.nasa.gov/pdfs/csrp7.pdf>

   <http://www.associatedpro.com/aps/x84lab/FPGA.html>

[28] Xilinx©, XC3000 Series Field Programmable Gate Arrays, (XC 300A/L, XC3100A/L),


[33] MATLAB™ code, ‘fpga.m’, ‘databaseheight.m’, provided by Uijt de Haag, M., Ohio University, Athens, OH, July 7, 2002.


[40] Society of Automotive Engineers, “ Certification Considerations for Highly-Integrated Or Complex Aircraft Systems,” SAE ARP 4754, November 1996,

APPENDIX A

Avionics-related FAR Parts with their names and numbers [9]

<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part 1</td>
<td>Definitions and Abbreviations</td>
</tr>
<tr>
<td>Part 21</td>
<td>Certification Procedures for Products and Parts</td>
</tr>
<tr>
<td>Part 23</td>
<td>Airworthiness Standards: Normal, Utility, Acrobatic, and Commuter Category Airplanes</td>
</tr>
<tr>
<td>Part 25</td>
<td>Airworthiness Standards: Transport Category Airplanes</td>
</tr>
<tr>
<td>Part 27</td>
<td>Airworthiness Standards: Normal Category Rotorcraft</td>
</tr>
<tr>
<td>Part 29</td>
<td>Airworthiness Standards: Transport Category Rotorcraft</td>
</tr>
<tr>
<td>Part 33</td>
<td>Airworthiness Standards: Aircraft Engines</td>
</tr>
<tr>
<td>Part 34</td>
<td>Fuel Venting and Exhaust Emission Requirements for Turbine Engine-Powered Airplanes</td>
</tr>
<tr>
<td>Part 39</td>
<td>Airworthiness Directives</td>
</tr>
<tr>
<td>Part 91</td>
<td>General Operating and Flight Rules</td>
</tr>
<tr>
<td>Part 121</td>
<td>Operating Requirements: Domestic, Flag, and Supplemental Operations</td>
</tr>
<tr>
<td>Part 183</td>
<td>Representatives of the Administrator</td>
</tr>
</tbody>
</table>
APPENDIX B

Relationship Among Airplane Classes, Probabilities, Severity of Failure Conditions, and SW Development Assurance Levels [17]

<table>
<thead>
<tr>
<th>Classification of Failure Conditions</th>
<th>No Safety Effect</th>
<th>&lt;---Minor---&gt;</th>
<th>&lt;---Major---&gt;</th>
<th>&lt;---Hazardous---&gt;</th>
<th>&lt; Catastrophic&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effect on Airplane</td>
<td>No effect on operational capabilities or safety</td>
<td>Slight reduction in functional capabilities or safety margins</td>
<td>Significant reduction in functional capabilities or safety margins</td>
<td>Large reduction in functional capabilities or safety margins</td>
<td>Normally with hull loss</td>
</tr>
<tr>
<td>Effect on Occupants</td>
<td>Inconvenience for passengers</td>
<td>Physical discomfort for passengers</td>
<td>Physical distress to passengers, possibly including injuries</td>
<td>Serious or fatal injury to an occupant</td>
<td>Multiple fatalities</td>
</tr>
<tr>
<td>Effect on Flight Crew</td>
<td>No effect on flight crew</td>
<td>Slight increase in workload or use of emergency procedures</td>
<td>Physical discomfort or a significant increase in workload</td>
<td>Physical distress or excessive workload impairs ability to perform tasks</td>
<td>Fatal injury or incapacitation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes of Airplanes</th>
<th>Allowable Quantitative Probabilities and Software (SW) Development Assurance Levels (Note 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class IV (Typically Commuter Category)</td>
<td>No Probability or SW Development Assurance Levels Requirement</td>
</tr>
<tr>
<td>Note 1</td>
<td>P=D</td>
</tr>
<tr>
<td>Note 2</td>
<td>P=C, S=D</td>
</tr>
<tr>
<td>Note 3</td>
<td>P=A, S=B</td>
</tr>
</tbody>
</table>

Note 1: A numerical probability range is provided here as a reference. The applicant is usually not required to perform a quantitative analysis for Minor and Major Failure Conditions. See Figure 3.

Note 2: The alphabets denote the typical Software (SW) Development Assurance Levels as described in ARPA 4754 for Primary System (P) and Secondary System (S). For example, SW Development Assurance Level A on Primary System is noted by P=A.

Note 3: At airplane function level, no single failure will result in a Catastrophic Failure Condition.

Note 4: At airplane function level, no single failure will result in the loss of a function that causes a Hazardous Failure Condition.

Note 5: Secondary System (S) may not be required to meet probability goals. If installed, S must meet stated criteria.

Note 6: A reduction of Software Development Assurance Levels applies only for Navigation, Communication, and Surveillance Systems if an altitude encoding altimeter transponder is installed. This option does not apply to CAT II/III operations.
SW/HW level Definitions [12,14]

SW and HW Levels [12,14]

<table>
<thead>
<tr>
<th>Failure Condition</th>
<th>RTCA/DO 178B SW Level</th>
<th>RTCA/DO 254 HW Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catastrophic</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>Hazardous</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>Major</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Minor</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>No effect</td>
<td>E</td>
<td>E</td>
</tr>
</tbody>
</table>

SW/HW level is based upon the contribution of SW/HW to potential failure conditions as determined by the system safety assessment/ hardware safety assessment process. The SW/HW level implies that the level of effort required showing compliance with certification requirements varies with the failure condition category. The SW/HW level definitions are:

1. **Level A**: SW/HW function whose anomalous behavior, as shown by the system safety assessment/ hardware safety assessment process, would cause or contribute to a failure of system function resulting in a catastrophic failure condition for the aircraft.

2. **Level B**: SW/HW function whose anomalous behavior, as shown by the system safety assessment/ hardware safety assessment process, would cause or contribute to a failure of system function resulting in a hazardous/sever-major failure condition for the aircraft.

3. **Level C**: SW/HW function whose anomalous behavior, as shown by the system safety assessment/ hardware safety assessment process, would cause or contribute to a failure of system function resulting in a major failure condition for the aircraft.

4. **Level D**: SW/HW function whose anomalous behavior, as shown by the system safety assessment/ hardware safety assessment process, would cause or contribute to a failure of system function resulting in a minor failure condition for the aircraft.

5. **Level E**: SW/HW function whose anomalous behavior, as shown by the system safety assessment/ hardware safety assessment process, would cause or contribute to a failure of system function resulting to a failure of system function with no effect on aircraft operational capability or pilot workload. Once SW/HW function has been...
confirmed as level E by the certification authority, no further guidelines of RTCA/DO-178B/RTCA/DO-254 apply.
APPENDIX D

STATEMENT OF ISSUE:
The <COMPANY NAME> <PRODUCT NAME & MODEL> proposes to use Programmed Logic Devices in airborne systems and equipment. At present there is no specific FAA policy or guidance for certification of airborne systems containing Programmed Logic Devices. The purpose of this Issue Paper is to define the specific aspects of certification associated with PLDs for systems containing such devices on the <COMPANY NAME> <PRODUCT NAME & MODEL> program.

BACKGROUND:
Systems used on the <Aircraft Model> will include Programmed Logic Devices. For clarification the following terminology applies:

**Programmed Logic Devices:**
Programmed Logic Devices include Application Specific Integrated Circuits (ASIC) and Programmable Logic Devices (PLDs).

**ASIC:**
An ASIC is defined as any unaltered programmed integrated circuit that is developed by or for the <COMPANY NAME> <Product Name & Model> that requires physical customizaton of the device by an ASIC vendor. Gate array, cell based and custom designs are included as they involve some level of customization of the mask sets used in the fabrication of the device.
PLD
A PLD is defined as any device that is purchased as an electronic part and altered to perform an application specific function. PLDs include, but are not limited to, Programmable Array Logic (PAL) devices, Programmable Logic Array (PLA) devices, General Array Logic (GAL) devices, Field Programmable Gate Array (FPGA) devices, and Erasable Programmable Logic Devices (EPLD). Programmable Logic Devices typically require programming via software which is done in-house by the equipment manufacturer.

These devices will be used in systems which have functions that can affect the safety of the airplane. These devices are often as complex as software controlled microprocessor based systems. Because of the nature and complexity of systems containing digital logic, the FAA has determined that adherence to a structured approach may be used to show compliance with FAR 25.1309 for complex, programmable logic devices. One means of showing such compliance for complex, programmable logic devices is adherence to the guidelines of RTCA document DO-178B, "Software Considerations in Airborne Systems and Equipment", as if these devices were software programs. Although systems containing Programmed Logic Devices can perform functions of the same complexity as software based systems, the FAA has no policy or guidelines for certification of systems containing Programmed Logic Devices. However, the problems are essentially the same as for software. This issue paper is concerned with the assurance of the encoded logic embedded in these devices.

FAA POSITION:
There is no existing FAA policy or guidance for showing compliance to the existing rules for those aspects of certification associated with Programmed Logic Devices. Accordingly, certification of systems on the <COMPANY NAME> <PRODUCT NAME & MODEL> which contain such devices will require the following:

Programmed Logic Devices associated with functions whose failure or malfunction could cause or contribute to a catastrophic failure condition for the aircraft as defined in Advisory Circular 25.1309-1A or to a hazardous/severe-major failure condition as defined in RTCA document DO-178B, shall undergo testing which demonstrates correct operation under all combinations and permutations of conditions of the gates within the device, or analysis which can show analogous results.

Programmed Logic Devices associated with functions whose failure or malfunction could cause or contribute to a major or a minor failure condition for the aircraft as defined in Advisory Circular 25-1309-1A shall undergo testing which demonstrates correct operation under all combinations and permutations of conditions at the pins of the device, or analysis which can show analogous results.

In the event that the complexity of the device makes the testing and analysis requirements outlined above uneconomical, the following shall apply:

IVT/Self-Study Course
Federal Aviation Administration
July, 1999
Programmed Logic Devices shall be developed using a structured development approach approved by the FAA. The structured approach should provide design verification which achieves the same result as that provided for software development by RTCA document DO-178B. The rigor of the structured approach should be commensurate with the hazard associated with failure or malfunction of the system in which the Programmed Logic Device is located. Guidance in this area can be found in the sections of DO-178B which describe the requirements for the software levels associated with software development and assurance. Furthermore, the applicant should ensure that: 1) Programmed Logic Devices are identified in the certification plan, 2) the development approach and rigor of the approach for each device is acceptable to the FAA, and 3) accomplishment summaries describe the means and level of design assurance achieved.

Information on how the applicant intends to present certification data for Programmed Logic Devices can be included in current certification plan documents or as stand-alone plans for Programmed Logic Devices.

Requirements identified in this issue paper do not in any way alleviate the need for traditional methods for hardware design and assurance.

**FCAA POSITION:**

**APPLICANT POSITION:**

**CONCLUSION:**

Manager, Transport Airplane Directorate
Airplane Certification

[Signature]

[Date]
## APPENDIX E

ARINC 429 Outputs For Global Navigation Satellite System (GNSS) Sensor

**Attachment 4-2**

**ARINC 429 Outputs For GNSS Sensor**

<table>
<thead>
<tr>
<th>OCT</th>
<th>PARAMETER</th>
<th>SIGNAL</th>
<th>UNITS</th>
<th>POS</th>
<th>RANGE</th>
<th>SIG</th>
<th>BITS</th>
<th>RES</th>
<th>MAX</th>
<th>MAX*</th>
<th>MAX**</th>
</tr>
</thead>
<tbody>
<tr>
<td>060</td>
<td>MEASUREMENT STATUS</td>
<td>PACKED</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>061</td>
<td>PSEUDO RANGE</td>
<td>MBR</td>
<td>METERS</td>
<td>*</td>
<td>268,435,456</td>
<td>20</td>
<td>256</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>062</td>
<td>PSEUDO RANGE FINE</td>
<td>MBR</td>
<td>METERS</td>
<td>*</td>
<td>4935</td>
<td>11</td>
<td>0.125</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>063</td>
<td>RANGE RATE</td>
<td>MBR</td>
<td>M/S</td>
<td>*</td>
<td>4935</td>
<td>20</td>
<td>0.0039</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>064</td>
<td>DELTA RANGE</td>
<td>MBR</td>
<td>METERS</td>
<td>*</td>
<td>4935</td>
<td>20</td>
<td>0.0039</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>065</td>
<td>SV POSITION X</td>
<td>MBR</td>
<td>Meters</td>
<td>ECEF</td>
<td>4935</td>
<td>20</td>
<td>64</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>066</td>
<td>SV POSITION Y</td>
<td>MBR</td>
<td>Meters</td>
<td>ECEF</td>
<td>4935</td>
<td>14</td>
<td>64</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>070</td>
<td>SV POSITION Z</td>
<td>MBR</td>
<td>Meters</td>
<td>ECEF</td>
<td>4935</td>
<td>14</td>
<td>64</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>071</td>
<td>SV POSITION P</td>
<td>MBR</td>
<td>Meters</td>
<td>ECEF</td>
<td>4935</td>
<td>14</td>
<td>64</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>072</td>
<td>SV POSITION Q</td>
<td>MBR</td>
<td>Meters</td>
<td>ECEF</td>
<td>4935</td>
<td>14</td>
<td>64</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>073</td>
<td>SV POSITION R</td>
<td>MBR</td>
<td>Meters</td>
<td>ECEF</td>
<td>4935</td>
<td>14</td>
<td>64</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>074</td>
<td>UTC MEASURED</td>
<td>MBR</td>
<td>SECONDS</td>
<td>*</td>
<td>10</td>
<td>20</td>
<td>9.33667354</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>076</td>
<td>锗 ALTITUDE (REL)</td>
<td>MBR</td>
<td>FEET</td>
<td>*</td>
<td>121,072</td>
<td>20</td>
<td>125</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>IPD</td>
<td>MBR</td>
<td>N/A</td>
<td>*</td>
<td>1024</td>
<td>15</td>
<td>0.031</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>IPD</td>
<td>MBR</td>
<td>N/A</td>
<td>*</td>
<td>1024</td>
<td>15</td>
<td>0.031</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>GNS TRACK ANGLE</td>
<td>MBR</td>
<td>DEGREES</td>
<td>CH^2</td>
<td>*</td>
<td>180</td>
<td>15</td>
<td>0.0005</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
</tr>
<tr>
<td>110</td>
<td>GNS LATITUDE</td>
<td>MBR</td>
<td>DEGREES</td>
<td>N</td>
<td>*</td>
<td>180</td>
<td>20</td>
<td>0.000172</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
</tr>
<tr>
<td>111</td>
<td>GNS LONGITUDE</td>
<td>MBR</td>
<td>DEGREES</td>
<td>E</td>
<td>*</td>
<td>180</td>
<td>20</td>
<td>0.000172</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
</tr>
<tr>
<td>112</td>
<td>GNS GROUND SPEED</td>
<td>MBR</td>
<td>KNOTS</td>
<td>*</td>
<td>4096</td>
<td>15</td>
<td>0.125</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>GNS LATITUDE FINE</td>
<td>MBR</td>
<td>DEGREES</td>
<td>CH^2</td>
<td>*</td>
<td>180</td>
<td>10</td>
<td>0.000172</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
</tr>
<tr>
<td>121</td>
<td>GNS LONGITUDE FINE</td>
<td>MBR</td>
<td>DEGREES</td>
<td>E</td>
<td>*</td>
<td>180</td>
<td>10</td>
<td>0.000172</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
</tr>
<tr>
<td>125</td>
<td>UTC</td>
<td>BCD</td>
<td>HIRMIN</td>
<td>*</td>
<td>23,59,9</td>
<td>5</td>
<td>0.1</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>HORIZONTAL INTEGRITY LIMIT</td>
<td>MBR</td>
<td>MN</td>
<td>*</td>
<td>16</td>
<td>17</td>
<td>0.000122</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>135</td>
<td>VERTICAL INTEGRITY LIMIT</td>
<td>MBR</td>
<td>FEET</td>
<td>*</td>
<td>32,768</td>
<td>18</td>
<td>125</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>136</td>
<td>UTC FRICTION</td>
<td>MBR</td>
<td>SECONDS</td>
<td>*</td>
<td>1,0</td>
<td>20</td>
<td>0.95567354</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>UTC FRICTION</td>
<td>MBR</td>
<td>SECONDS</td>
<td>*</td>
<td>1,0</td>
<td>20</td>
<td>0.95567354</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>141</td>
<td>UTC FRICTION</td>
<td>MBR</td>
<td>SECONDS</td>
<td>*</td>
<td>1,0</td>
<td>20</td>
<td>0.95567354</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>145</td>
<td>UTCB</td>
<td>BCD</td>
<td>HIRMIN</td>
<td>*</td>
<td>23,59,9</td>
<td>17</td>
<td>1.0</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>146</td>
<td>NORTH/SOUTH VELOCITY</td>
<td>MBR</td>
<td>KNOTS</td>
<td>*</td>
<td>4096</td>
<td>15</td>
<td>0.125</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>EAST/WEST VELOCITY</td>
<td>MBR</td>
<td>KNOTS</td>
<td>*</td>
<td>4096</td>
<td>15</td>
<td>0.125</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>260</td>
<td>DATE</td>
<td>BCD</td>
<td>STNDY</td>
<td>*</td>
<td>6</td>
<td>1</td>
<td>66</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>271</td>
<td>GNS SENSOR STATUS</td>
<td>DIS</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>335</td>
<td>MAINTENANCE WORD</td>
<td>BCD</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
</tbody>
</table>

---

* Time is in milliseconds. Nominal transmit interval is expected to be 1000 milliseconds.

---

**Output rates per satellite**

- The satellite measurement block should begin with measurement status and end with UTC measurement time. The maximum number of satellite measurement blocks which can be transmitted in one second is 32. The maximum transmit delay of the SV measurements is approximately 1000 milliseconds.

---

**Attained Positive**

**Fine data words contain the truncated position of the original data word. This information is unsigned although the sign bit is reserved. The two labels are concatenated (or combined) in the receiver.**

---

**The following information should be used in the generation of the raw data measurement blocks:**

1. The measurement status should indicate the SVID, measurement validity, USER clock update, rate measurement type and the CNM as defined in Attachment 4-3.
2. When not corrected within the differential mode, the pseudo range and pseudo range rate or delta range should be corrected for the satellite clock, ionospheric and tropospheric errors (defined in CDGPS-200 and GLOMARS IG). The clock correction should be applied to the satellite clock using a satellite clock update rate and measurement type of 0. The measurement status should be set to 2 when the sky is clear of satellites and the receiver has a valid satellite signal and is set to 0 whenever the receiver detects a satellite failure.
3. The ionospheric and tropospheric delay should be based on the ARINC 429 position input when the GPS position cannot be determined.
4. The integration time for the integrated delta range should be 1 second.
5. The integration time for the integrated delta range should be 1 second.
6. The integration time for the integrated delta range should be 1 second.
7. The satellite position from each SV should be corrected by the GNSS sensor for the rotation of the earth during the time of the signal transmission to earths.
8. The integration time for the integrated delta range should be 1 second.
9. The receiver identification being used, the raw data measurement block parameters should be corrected with differential corrections. The method of correction should be based on the differential GNSS method being used.