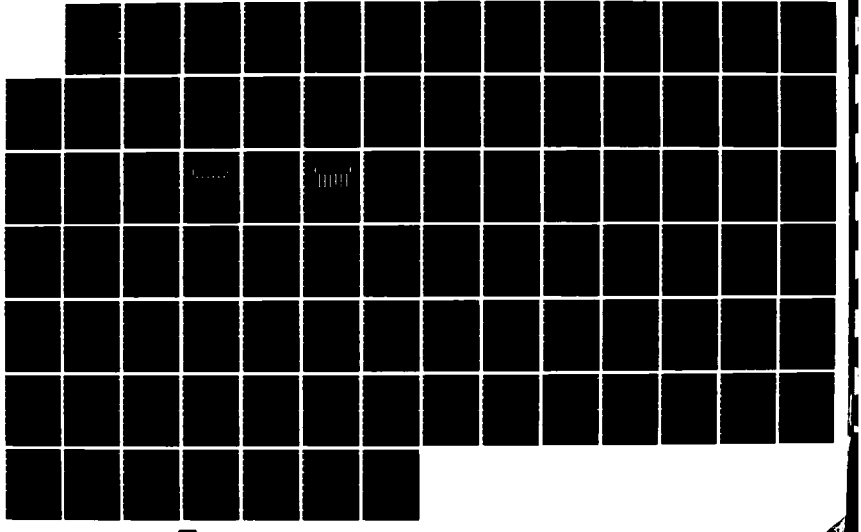
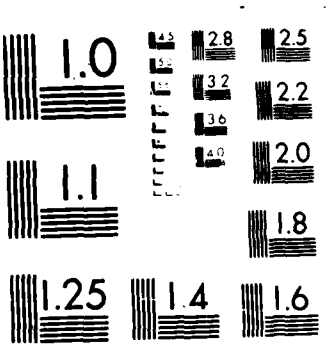
AD-A179 344 MODELING AND SIMULATION OF THE WFTA (WINOGRAD FOURIER TRANSFORM ALGORITHM (U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI... C H COOPER 1/1

UNCLASSIFIED DEC 86 AFIT/GE/ENG/86D-44-VOL-1 F/G 12/1 NL





MODELING AND SIMULATION OF THE WFTA 16 PFA PROCESSOR USING THE VHSIC HARDWARE DESCRIPTION LANGUAGE VOLUME I

THESIS

Charles H. Cooper
Captain, USAF

AFIT/GE/ENG/86D-44

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/86D-44

MODELING AND SIMULATION OF THE WFTA 16 PFA PROCESSOR USING THE VHSIC HARDWARE DESCRIPTION LANGUAGE VOLUME I

THESIS

Charles H. Cooper
Captain, USAF

AFIT/GE/ENG/86D-44

Approved for public release; distribution unlimited

AFIT/GE/ENG/86D-44

MODELING AND SIMULATION OF THE WFTA 16 PFA PROCESSOR USING THE VHSIC HARDWARE DESCRIPTION LANGUAGE

VOLUME I

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

Charles H. Cooper, B.S.

Captain, USAF

A-1

December 1986

Approved for public release; distribution unlimited

Acknowledgement

I would like to thank my thesis advisor, Captain Richard Linderman, for the guidance and timely remotivation needed to ensure the successful completion of this research effort. I would especially like to thank Lieutenant Colonel (Retired) Harold Carter for his repeated assistance in getting the VHDL support environment installed on the AFIT computer resources.

Most importantly, I would like to thank my wife, Jennifer Leigh, for all the assistance and support she provided during the months of research. Without her patience and understanding, this thesis effort would not have been completed.

Chuck Cooper

TABLE OF CONTENTS

VOLUME I

Acknowledgements ii

List of Figures vi

Abstract viii

Chapter 1 Introduction

1.1 Overview 1

1.1.1 VHDL 2

1.1.2 WFTA Digital Signal Processing 6

1.2 Problem 7

1.3 Scope 7

1.4 Assumptions 8

1.5 Standards 8

1.6 Summary of Current Knowledge 8

1.6.1 1985 WFTA/VHDL Thesis Work at AFIT 8

1.6.1.1 Capt. James M. Collins' Thesis 8

1.6.1.2 Capt. Paul C. Rossbach's Thesis 9

1.6.1.3 Capt. Paul W. Coutee's Thesis 9

1.6.1.4 Capt. Kent Taylor's Thesis 9

1.6.2 1986 WFTA/VHDL Thesis Work at AFIT 10

1.6.2.1 Capt. Carl Shephard's Thesis 10

1.6.2.2 Capt. Gary Hedrick's Thesis 10

1.6.3 N.2 Simulation System 11

1.7 Approach 12

1.8 Sequence of Presentation 13

Chapter 2 Development of WFTA Architecture

2.1 Overview 15

2.2 The Winograd Fourier Transform Algorithm 15

2.3 WFTA Pipeline Processor Architecture 18

2.4 VHDL Support Environment 23

2.4.1 VHDL Language Analyzer 26

2.4.2 VHDL Design Library Mangager 27

2.4.3 VHDL Simulator 31

2.4.4 VHDL Design Library 32

2.4.5 VHDL Reverse Analyzer 34

2.4.6 VHDL Simplifier 35

Chapter 3 VHDL Modeling of the WFTA 16 PFA Processor

3.1 Overview 36

3.2 Top-down Decomposition of the WFTA 16 PFA Processor . 38

3.2.1 Signal Flow and Operation of the WFTA 16 PFA Processor 39

3.2.2 Decomposition of the PISO and SIPO Registers . 46

3.2.3 Decomposition of the Arithmetic Circuitry 49

3.3 VHDL Format for the WFTA 16 PFA Processor 53

3.4 Bottom-up Composition of the VHDL Design Library 57

Chapter 4 Simulation of Simple WFTA 16 PFA Processor Components

4.1 Overview 60

4.2 Development of the VHDL Test Bench 60

4.3 Model Generation from the VHDL IVAN Representations . 68

4.4 Compiling and Linking of the VHDL Ada Models 69

4.5 Running the Simulator Kernel 71

4.6 Execution of the Report Generator 71

Chapter 5 Conclusions and Recommendations

5.1 Conclusions 73

5.2 Recommendations 74

VOLUME II

APPENDIX A VHDL Department of Defense Objectives A-1

APPENDIX B VHDL Modeling Descriptions B-1

APPENDIX C VHDL Analyzer Listings C-1

APPENDIX D VHDL Simulation Reports D-1

APPENDIX E VHDL Error Report E-1

LIST OF FIGURES

Chapter 2

Figure 2-1. Cubic Data Structure of the 4080-point Good-Thomas PFA Implementation 19

Figure 2-2. 4080-Point WFTA PFA Processor 20

Figure 2-3. Winograd Processor Architecture 21

Figure 2-4. Active/Watchdog Processors 22

Figure 2-5. The VHDL Support Environment 25

Figure 2-6. The VHDL Language Analyzer 28

Figure 2-7. Role of the Design Library Manager 30

Figure 2-8. VHDL Supports Hierarchical Descriptions 33

Figure 2-9. Example of the Design Library Organization 34

Chapter 3

Figure 3-1. Winograd Processor Architecture 37

Figure 3-2. Decomposition and Signal Flow of the WFTA 16 Processsor 40

Figure 3-3. Pre-addition Pipeline 44

Figure 3-4. Post-addition Pipeline 45

Figure 3-5. PISO Cell 47

Figure 3-6. SIPO Cell 47

Figure 3-7. Clocked CMOS Latch 49

Figure 3-8. Column Form of the WFTA 16 PFA Processor 50

Figure 3-9. Resettable CMOS Latch 52

Figure 3-10. Interface Declaration Format 54

Figure 3-11. Body Declaration Format 55

Chapter 4

Figure 4-1. PISO Cell 61

Figure 4-2. PISO Test Bench Interface Declaration 62

Figure 4-3. PISO Test Bench Body Declaration o3

Figure 4-4. Timing Diagram for Two Phased Clock 66

Figure 4-5. Dynamic MSFF 67

Abstract

The VHSIC Hardware Description Language (VHDL) is applied to the problem of modeling and simulating VLSI CMOS components of the WFTA 16 PFA processor. The 16-point PFA processor is one of three PFA processors under design and development for the implementation of the 4080-point PFA pipeline processor by the VLSI design group at the Air Force Institute of Technology. The PFA processor is modeled by applying the hierarchical facilities of the VHDL language to form the top level register component descriptions from combinations of the primary building block hardware element descriptions. Two simple VHDL simulations are performed using the beta test versions of the VHDL simulator and support environment. The simple component simulations are performed on a VHDL behavioral description of the Parallel-In, Serial-Out register cell and a VHDL structural description of a dynamic MSFF.

MODELING AND SIMULATION
OF THE
WFTA 16 PFA PROCESSOR
USING THE
VHSIC HARDWARE DESCRIPTION LANGUAGE
VOLUME I

Chapter 1

Introduction

1.1 Overview

The development of technology critical Department of Defense (DOD) programs, such as the Strategic Defense Initiative (SDI), continually drive the state-of-the-art in silicon fabrication technology. The actual state-of-the-art in silicon fabrication technology doubles approximately every three to five years, and currently includes the development of the VHSIC Phase I (Very High Speed Integrated Circuit) class of silicon chips. The VHSIC class of integrated circuit (IC) chips is ideally suited for critical military system applications due to the extremely high operating speeds, the tremendous rate of data throughput, and the minute size of the individual IC's.

The rate of change in the state-of-the-art in silicon fabrication technology results in a tremendous increase in both the performance and function achieved by a single IC. With the increase in the performance and function of an IC comes an increase in the design complexity of the same IC. As the number of functions per IC chip is increased the design

complexity increases. In the silicon industry, recent advances in the development and application of VHSIC chips require functionally complex chip designs. This no longer affords a company the luxury of allowing the complete development of a single IC to be the sole responsibility of an individual design engineer. Instead, VHSIC chips are developed by design teams that require concise and accurate communication of design information between team members and possibly even between design teams composed of more than one silicon manufacturing company [3].

In the past, the communication of critical design information for the function of small and medium scale ICs was accomplished through the formal language approach, using hardware description languages (HDLs). Unfortunately, these HDLs were not flexible or universal enough to allow modifications in the language to keep pace with the advances and changes in the technology of the silicon industry. As the silicon fabrication industry advances into the VLSI (Very Large Scale Integration) era there exists a need to develop portable computer aided engineering tools that can both model and simulate the VLSI class of IC chips in a concise and timely manner [8].

1.1.1 VHDL

In March 1980, the Department of Defense (DOD) established the VHSIC program office to drive the state-of-the-art in IC technology toward the needs of the DOD. In the early stages of development of the VLSI program, the program office recognized that a single standard HDL acceptable to the nation's defense contractors and major universities

was not available. However, the lack of a single standard language for hardware design was very similar to the problem that the Ada program office had encountered a few years earlier when the Department of Defense set out to establish the Ada programming language as the single standard software programming language for all the defense contractors. To solve the problem, the VHSIC program office decided to use a similar approach to that of the Ada program office and set out to establish a DOD standard HDL for the defense contractors and major universities. The result was the development of the VHSIC Hardware Description Language (VHDL) [18].

In the summer of 1981, the Institute for Defense Analyses set up the Orono workshop to define the system requirements for a standard HDL. The results of this workshop were used as the basis for establishing the set of language requirements for VHDL and for defining the system requirements for the VHDL support environment [18]. In addition, the VHSIC program office levied the requirement that the VHDL language be based on Ada constructs wherever possible [14].

The primary objective of the VHDL language is to support technology insertion. Technology insertion is defined in the "VHDL Design Analysis and Justification" report as "the utilization of the latest technology in the development of new systems, as well as in existing ones, in order to relax their environmental requirements, enhance their capabilities, or improve their performance" [9]. When the VHDL language has been fully developed and tested there should be a significant reduction in both the lag time and total system costs required to insert the most current VLSI state-of-the-art technology into DOD systems [9].

The team of Intermetrics, IBM, and Texas Instruments was awarded the DOD contract for the design, development, and implementation of VHDL and its support environment by the VHSIC program office. The preliminary design phase of the VHDL contract began in July 1983 and concluded in July 1984 with the delivery of the language design. During a two month review period the language design was reviewed extensively by the DOD user community, the nation's defense contractors, and the major colleges and universities conducting VLSI research and development. In October 1984 the VHDL development contractors began the implementation phase by first modifying the existing VHDL design language (Version 5.0) based on the recommendations agreed upon during the design reviews [17]. This resulted in the release of Version 7.2 of the VHDL language. The VHDL implementation phase was originally scheduled to end in December 1985; however, this phase is now scheduled to end in early 1987 and will culminate with the delivery of the final implementation of the design tools for the Version 7.2 of the VHDL support environment [2].

The design, development, and implementation of VHDL as a standard HDL has made significant progress in the past two years and has caught the interest of segments of the DOD user community. However, the VHSIC program office must still convince the majority of the defense community that VHDL is capable of satisfying all the requirements for a complete HDL at all the levels of digital system description. Furthermore, the program office must also convince the defense community that the language will have the universality and flexibility to adapt to the technology changes in silicon manufacturing and remain a viable design tool well

into the 1990's. The VHSIC program office has recruited the assistance of the Air Force Institute of Technology (AFIT) School of Engineering to help pioneer the beta test and use of the VHDL language in the research and development of VLSI technologies at major universities [2].

The AFIT School of Engineering is conducting thesis research in four major areas of VHDL development. The first area of research involves the development of an UNIX-based VHDL compiler for the modeling, simulation, and design of VLSI chips at major universities using the UNIX operating system. The second area of research is being done in the definition of design tools to be used in conjunction with the VHDL development of VLSI IC's. The third area of research requires the definition and development of the interface requirements between the UNIX-based VHDL compiler and the individual VHDL design tools. Finally, a portion of the VLSI beta test research conducted at AFIT will include the design, development, fabrication, modeling, and simulation of a pipeline of VLSI Prime Factor Algorithm (PFA) signal processors which implement the Winograd Fourier Transform Algorithm (WFTA). The VHSIC program office is funding the VLSI research and VHDL education conducted at the AFIT School of Engineering in order to help develop a foundation for the application of the VHDL language at major universities involved in VLSI research [2].

The VHSIC program office is continuing to make every effort possible to ensure that the VHDL language captures all the requirements of the potential user groups and develops a large, widespread base among DOD agencies, defense contractors, and major universities. This includes the submission of the VHDL language to the Institute of Electrical and

Electronic Engineers (IEEE) for approval as the standard HDL for VLSI class ICs. The IEEE held reviews on the VHDL language this year in an effort to release the approved VHDL language as an industry standard at the earliest possible date [2].

1.1.2 WFTA Digital Signal Processing

As the state-of-the-art in VLSI technology increases a systems ability to receive information or collect data, it also increases the burden on the system to process or obtain useful information from the data. In fact, in most of the advanced military systems in development and/or use today, the most limiting factor in system performance is the ability of the system to process the information received in a real-time or near real-time mode of operation. However, great strides have been made toward reducing the total amount of time required to accomplish the digital signal processing requirements of a system. With many of the most important advancements coming in the application of state-of-the-art VLSI technologies to provide the ability to implement very complex signal processing functions on a single silicon substrate. Additionally, the application of innovative algorithm designs can physically reduce the total number of numerical computations required in most system signal processing [3].

The Air Force Institute of Technology is currently developing a PFA pipeline signal processor which combines innovative techniques for the development of processor algorithms with the advanced silicon properties of the VLSI technology. The result is the implementation of the design, development, and test of the 4080-point Winograd Fourier Transform

Algorithm (WFTA) method of implementing a Discrete Fourier Transform (DFT). The WFTA PFA pipeline processor should produce about a tenfold increase in the signal processing throughput over the current use of existing signal processing algorithms [3].

1.2 Problem

The problem addressed in this thesis is to model and simulate the large CMOS WFTA 16 PFA processor using the VHSIC Hardware Description Language (VHDL).

1.3 Scope

The VHDL modeling and simulation research addressed in this thesis is one of three concurrent 1986 efforts, under the guidance and direction of Dr. Linderman, implementing the description, design, development, and fabrication of the VHSIC 4080-point WFTA PFA pipeline signal processor. Capt. Carl Shephard has designed, developed, and fabricated the VLSI architecture for the 16-point WFTA DFT from the original WFTA algorithm concepts developed by Dr. Linderman and advanced by four 1985 graduate electrical engineering thesis students. Capt. Gary Hedrick has designed, developed, and fabricated the PFA pipeline interface processor and error correcting memory chips for the 4080-point WFTA pipeline PFA signal processor.

The major portion of the beta test research in this thesis will be directed toward the analysis of VHDL as a design tool for the modeling and simulation of the 16-point DFT developed for the 4080-point WFTA PFA pipeline signal processor. The research analysis performed in this thesis will examine and evaluate the ability of the VHDL language and

VHDL support environment to support the modeling and simulation task. There will be no hardware development in this thesis; however, any design errors identified in the modeling and simulation process will be brought to the attention of the actual processor chip design engineer.

1.4 Assumptions

All VHDL models will be developed upon the completion of the VLSI component design and development process. This thesis assumes complete verification of the arithmetic and control functions for each of the DFT components which form the VHSIC 4080-point WFTA pipeline processor. In addition, to accomplish the research required of this thesis effort the VHDL software required for VHDL source code analysis and simulation must be installed and available for use.

1.5 Standards

The overall objective of the VHDL program is to develop a standard hardware description language for VLSI class components. This thesis will evaluate the proposed VHDL standard with respect to ability to model and simulate the WFTA 16 PFA processor.

1.6 Summary of Current Knowledge

1.6.1 1985 WFTA/VHDL Thesis Work at AFIT

1.6.2.1 Capt. James M. Collins' Thesis

Capt. Collins' thesis provides the basic foundation upon which this thesis is built. The thesis applied the preliminary version (Version 5.0) of the VHDL language to the problem of modeling a CMOS VLSI class

circuit. In the thesis Capt. Collins applies the hierarchical facilities of the VHDL language to the modeling of the 16-point DFT processor, which is a major component of the 4080-point WFTA PFA pipeline processor. In addition, Capt. Collins' thesis used the C programming language to verify timing, control, and hardware macrocells used to implement the WFTA PFA pipeline processor architecture [3].

1.6.1.2 Capt. Paul C. Rossbach's Thesis

Capt. Rossbach's thesis addresses the initial design, simulation, implementation, and testing of the control circuitry for the VLSI linear 4080-point WFTA pipeline processor. The WFTA processor is designed to compute a 4080-point DFT of complex data points approximately every 120 microseconds, using a 70 MHZ clock signals [16].

1.6.1.3 Capt. Paul W. Coutee's Thesis

Capt. Coutee's thesis addresses the initial design, development, and implementation of the arithmetic circuitry for the VLSI 4080-point WFTA PFA pipeline processor. The thesis includes a detailed analysis of the serial pipeline architecture of the arithmetic circuitry sections of the WFTA PFA processors [4].

1.6.1.4 Capt. Kent Taylor's Thesis

Capt. Taylor's thesis addresses the design validity and VLSI circuit implementation of the Winograd Fourier Transform Algorithm and the Good-Thomas Prime Factor Algorithm. The two algorithms are used to compute the linear 4080-point DFT by creating an array of DFTs with blocklengths of 15-points, 16-points, and 17-points. In addition, Capt. Taylor's

thesis investigates the numerical accuracy of the algorithms through a software simulation that uses the signal-to-noise ratio as the accuracy metric. The arithmetic circuitry of the WFTA PFA processor is designed to compute over eight thousand 4080-point DFT problems per second, with a numerical accuracy of over 10 dB [18].

1.6.2 1986 WFTA PFA Thesis Work at AFIT

1.6.2.1 Capt. Carl Shephard's Thesis

As previously stated, Capt Shephard is a member of the 1986 WFTA design/test team directed by Dr. Richard Linderman. Capt. Shephard's thesis addresses the design, development, and fabrication of the VLSI architecture for the 16-point, 15-point, and 17-point WFTA processors. The processors will be designed and fabricated using a 1.25 micron VLSI technology. The 1986 thesis research conducted by Capt. Shephard is a continuation of the preliminary design and development efforts of the 1985 WFTA design/test team, which were described in the previous section.

1.6.2.2 Capt. Gary Hedrick's Thesis

Capt. Hedrick is also a member of the 1986 WFTA design/test team. Capt. Hedrick's thesis addresses the design, development, and fabrication of the pipeline interface processor and the processor error correcting memory for the 4080-point WFTA pipeline processor. Capt. Hedrick's chip designs will also be implemented using 1.25 micron VLSI technology. All of Capt. Hedrick's thesis efforts are new thesis research work, and are not a continuation of the 1985 WFTA design efforts.

1.6.4 N.2 Simulation System

The N.2 simulation system is a collection of computer aided design (CAD) software tools assembled to assist system architects, digital design engineers, and software programmers in the development, test, and evaluation of potential hardware and software designs prior to the actual system implementation. N.2 is a register transfer level (RTL) simulation system which utilizes the Instruction Set Processor (ISP) language to describe digital hardware components. The N.2 simulation system was identified as a potential design tool for the modeling and simulation of the WFTA pipeline processor, due to the N.2 systems ability to describe hardware components. However, after extensive research and analysis, a major limitation in the N.2 simulation system prevented the use of the system as a modeling and simulation tool for the evaluation of the WFTA pipeline processor. The current AFIT CAD version of the N.2 simulation system is limited to the description of hardware components at the RTL level, and does not support hardware descriptions at the logical design level (gates, flip-flops, etc.) or the circuit design level (transistors, resistors, etc.). Therefore, if future thesis research work at AFIT is to include N.2 simulation of hardware designs at the logical design level an updated version of the N.2 simulation system must be purchased. An N.2 description of the 4080-point WFTA pipeline processor would require logical design level descriptions and will not be addressed in this thesis effort due to inability of the software [1].

1.7 Approach

The primary emphasis of this thesis effort was to model and simulate the VLSI large CMOS WFTA 16 PFA processor using Version 7.2 of the VHDL language and UTMC Release 2.0 of the VHDL support environment. As previously stated, the VHDL source code descriptions developed by Capt Collins formed the baseline source code descriptions used to model and simulate the WFTA 16 PFA processor. The 1985 VHDL source code modeling descriptions developed by Capt. Collins were rewritten and modified to conform to Version 7.2 of the VHDL language, and then run on the VHDL language analyzer to verify conformity with the VHDL language syntax, semantic, and lexical construct requirements. The WFTA 16 PFA processor modeling and simulation results provided the much needed inputs to the understanding of the relative ease of operation and application of the VHDL language and VHDL language support environment. The results also provided insight into the effectiveness of the VHDL language to model VLSI ICs.

The VHDL source code developed by Capt. Collins was accomplished through the structural decomposition of the 16-point WFTA processor. The VHDL source code descriptions of the processor architecture were modeled by decomposing the WFTA processor into subsystems, major components of those subsystems, macrocells of the major components, and finally the microcells that make up the macrocells. Decomposition of the 16-point PFA processor architecture led to a top-down approach to VHDL modeling, and provides a definition of the hardware and signal interfaces at each of the levels of decomposition. Once the lowest level of decomposition

is complete, the smallest individual logic components were modeled using a behavioral architecture description of the VLSI processor hardware. The 16-point WFTA PFA processor was then modeled at successively higher levels of decomposition, by using the previously analyzed VHDL library descriptions of the lower level components to create structural, and mixed structural and behavioral architecture descriptions through the predefined hardware and signal interfaces. Once the 16-point WFTA PFA processor is modeled and the VHDL library contains all the hardware modeling descriptions at each of the decomposition levels, then the simulation of the processor or component of the processor is performed to verify the dynamic behavior.

1.8 Sequence of Presentation

Chapter 2 begins by presenting a brief description of the Winograd Fourier Transform Algorithm and the development of the WFTA PFA pipeline processor architecture being developed at AFIT. The chapter concludes with a brief description of the VHDL language support environment.

Chapter 3 reports on the steps involved with the modeling of the WFTA 16 PFA processor and the creation of the VHDL design library to support the modeling descriptions of the PFA processor. The VHDL source code modeling descriptions and the VHDL language analyzer listings for the WFTA 16 processor will be provided in the appendix.

Chapter 4 reports on the steps involved with the simple simulation of a single WFTA 16 PFA processor component. The chapter describes the steps involved with the compilation, linking, and running of Ada programs created from the VHDL source code modeling descriptions contained in the

VHDL design library. The Ada programs created perform the actual dynamic analysis and simulation of the hardware component. The simulation also generates a report of the time history of the simulation. The reports from simulation runs will be provided in Appendix D.

Chapter 5 provides an analysis of the utility of the VHDL language as a VLSI design tool for the modeling and simulation of large CMOS ICs. Recommendations and conclusions based on the research performed in this thesis will be presented. An error report on problems encountered in the modeling and simulation efforts will be provided in Appendix E.

Chapter 2

VHDL Development of WFTA Architecture

2.1 Overview

This chapter presents the reader with the basic WFTA concepts and VHDL support system background required to gain a complete understanding of the modeling and simulation results presented in Chapters 3 and 4. This thesis assumes that the reader is familiar with the basic constructs of the VHDL language. If the reader is not familiar with the use of the VHDL language, Volumes I (The Tutorial) & II of the VHDL User's Reference Manual [11,12], the VHDL Language Reference Manual [10], and Chapter 3 of Capt. Collins' thesis [3] all provide excellent references for learning the basic concepts of the VHDL language.

The chapter begins with a brief description of the development of the Good-Thomas Prime Factor Algorithm (PFA) and the Winograd Fourier Transform Algorithm. Once an understanding of these two algorithms is established, their specific application to the AFIT 4080-point WFTA PFA pipeline processor is presented in the next section. The final section of the chapter is dedicated to the understanding of the application of the VHDL language support environment. Appendix A of the this thesis provides the reader with additional VHDL information on the definition of the seven DOD objectives for the VHDL language and VHDL language support environment.

2.2 The Winograd Fourier Transform Algorithm

The primary objective of VLSI DFT design engineers is to directly and efficiently compute very large DFTs by minimizing the total number of

arithmetic operations required. Graduate students at AFIT, under the direct supervision of Dr. Richard Linderman, have implemented in VLSI a DFT algorithm technique designed to achieve a significant reduction in the number of arithmetic operations required to calculate a 4080-point DFT. In fact, the current VLSI design of the WFTA 4080-point processor will reduce the total number of arithmetic additions and multiplications by a factor greater than 500 over the direct implementation of the DFT algorithm [3]. The dramatic reductions in the total number of arithmetic operations required by the processor were achieved through the use of the following two computationally efficient algorithms: 1) the Good-Thomas Prime Factor Algorithm (PFA), and 2) the Winograd Fourier Transform Algorithm [4].

The Good-Thomas PFA is applied to a one-dimensional linear DFT with an array of n data points, and transforms the linear array into m DFTs with a m-dimensional array of n data points. As a result, the direct implementation of the Good-Thomas PFA allows the arithmetic computations of a large one-dimensional DFT to be obtained through the calculation of a serial sequence of m fourier transforms. However, the application of the Good-Thomas PFA requires the concurrent application of the Chinese Remainder Theorm (CRT), to properly map the linear sequential data points into the m-dimensional data structure. The implementation of the CRT requires that the m-dimensions of the data structure be relatively prime from one another; therefore, the m-dimensional data structure should not contain any dimension lengths with the same common factors. Furthermore, to achieve the maximum efficiency for a sequential PFA pipeline processor

network, each of the DFT processors should take approximately the same amount of time to solve their respective DFT arithmetic operations. This restricts the lengths of the m dimensions of the data structure to m prime lengths of the same relative size [3,4,18].

Once the Good-Thomas PFA has been applied to a large DFT to obtain a sequence of m smaller DFTs, the Winograd Fourier Transform Algorithm is applied to each of the m smaller DFTs to minimize the total number of arithmetic operations required to compute each of the DFTs. The WFTA, developed by Dr. Shmuel Winograd, represents a significant improvement in the standard algorithm techniques used by the direct implementation of many of the existing Fast Fourier Transform (FFT) algorithms. In fact, Dr. Winograd's algorithm reduces the total number of multiplications in the arithmetic operations required to compute a DFT from O(nlogn) to O(n), while at the same time leaving the number of additions constant. Therefore, the Winograd processing algorithm significantly improves the computational efficiency of the PFA pipeline processor over the direct implementation of existing FFT algorithms [3,4].

As a result, the 4080-point DFT was chosen as the best possible demonstration case for the design, development, and fabrication of a WFTA PFA pipeline processor network. The 4080-point DFT also has many practical applications because it closely approximates the 4096-point scans used in current radar systems [3]. The design of the 4080-point WFTA pipeline processor was based on the application of the Good-Thomas PFA, which transforms the linear 4080-point DFT into a sequence of three relatively prime DFTs (16-point, 15-point, and 17-point DFTs). The three

prime DFTs were chosen to help achieve the maximum PFA pipeline processor efficiency by requiring the total computation times of the DFT processors to be approximately equal to each other. The 3-dimensional data structure created by the application of the Good-Thomas PFA is demonstrated in Figure 2-1. Additional information on the application of the Good-Thomas PFA, the CRT, and the WFTA to the development of the 4080-point WFTA PFA pipeline processor can be obtained by referencing the 1985 WFTA theses [3],[4],[16], and [18].

2.3 WFTA Pipeline Processor Architecture

As previously stated, AFIT is developing a 4080-point WFTA pipeline processor. The PFA processor is constructed from three smaller prime DFT blocks determined by the application of the principles of the Good-Thomas PFA. The WFTA pipeline processor solves a 4080-point DFT through the sequential calculation of 255 (15*17) 16-point DFTs, 272 (16*17) 15-point DFTs, and 240 (15*16) 17-point DFTs. The VLSI WFTA pipeline architecture designed to implement the sequential calculations is shown in Figure 2-2. As seen in the figure, the 4080-point WFTA pipeline processor is composed of the three WFTA processors (16,15,17), eight individual memory blocks, four memory controllers, and an interface control processor.

Each of the three WFTA processors can be viewed as a sequence of pre-additions, multiplications, and post-additions to a given input vector. Therefore, the WFTA process can be written as the equation:

$$\underline{X} = CDA\underline{x} \qquad (1)$$

where D is defined as the diagonal matrix composed of the multiplication

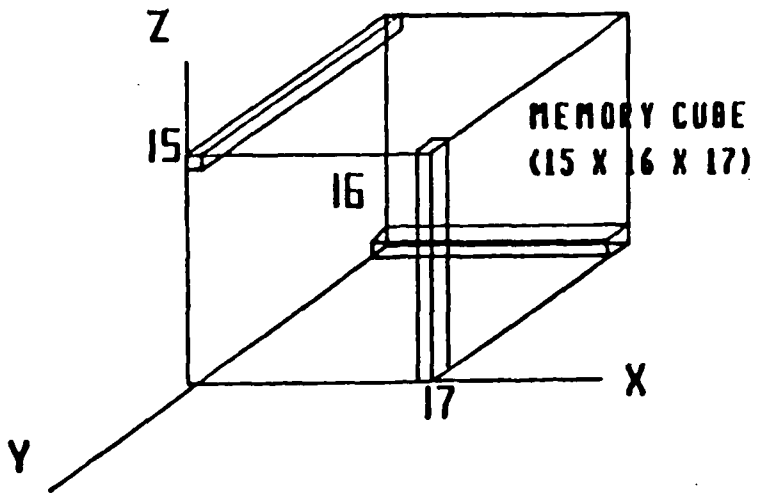


FIGURE 2-1. Cubic Data Structure of the 4080-point Good-Thomas PFA Implementation [3].

coefficients, C is defined as the incidence matrix of pre-additions, and A is defined as the incidence matrix of post-additions [3,16]. The hardware used to implement the above equation for each of the WFTA PFA processors is shown in Figure 2-3. However, one should note that the design structure in Figure 2-3 performs separate pre-additions and multiplications for both the real and imaginary parts of the data stream. Therefore, the data only takes on a complex format in the post-addition operations of the WFTA PFA processor.

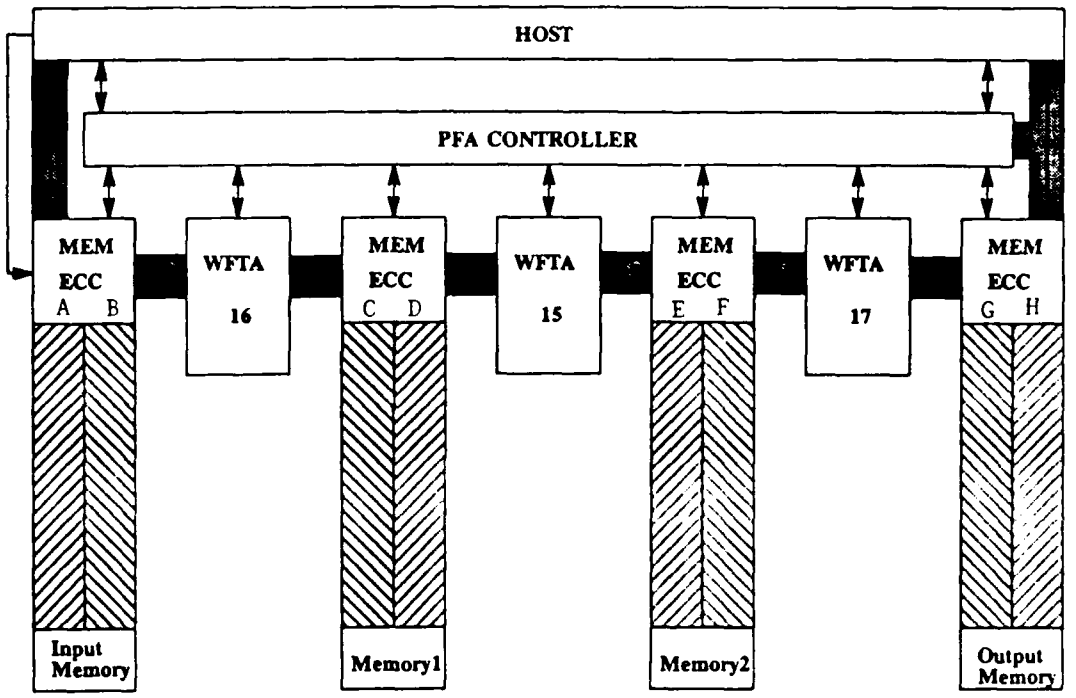


FIGURE 2-2. 4080-Point WFTA PFA Processor.

Each of the three processors in the 4080-point WFTA PFA pipeline processor are designed with a fault tolerance capability. The fault tolerance is incorporated into the basic architectural design of each WFTA PFA processor through the use of redundant processors and a "watch dog" monitoring of the functional operation of the processors for parity

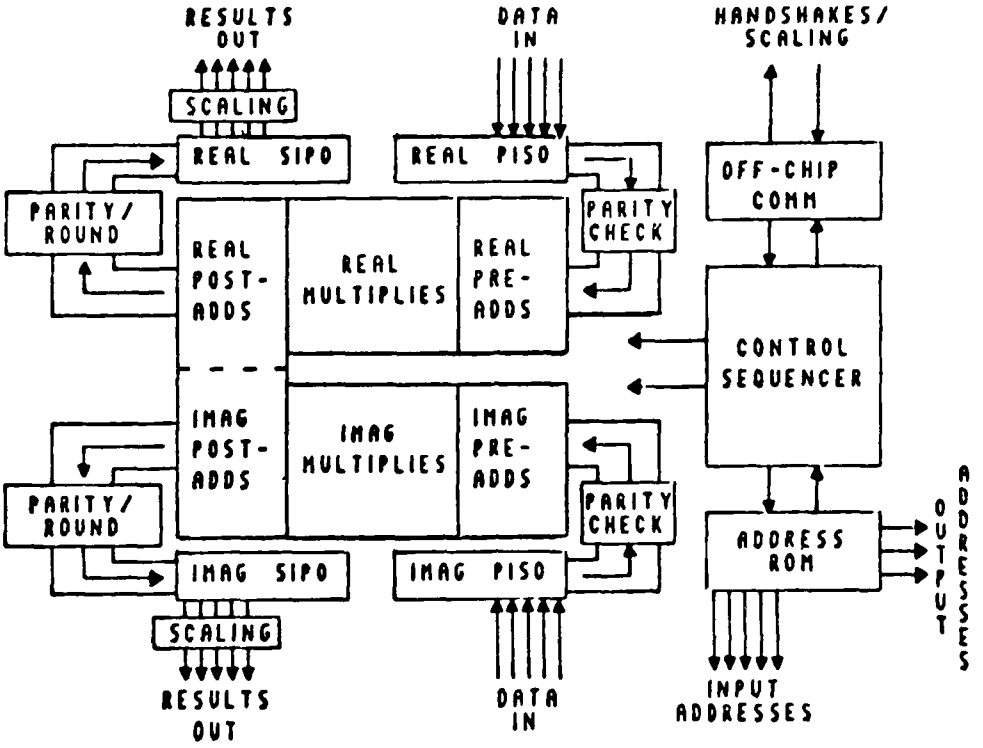


FIGURE 2-3. Winograd Processor Architecture [3].

errors. The redundant design architecture of the WFTA processors is shown in Figure 2-4. The control of each of the redundant processors is resident in the PFA interface control processor. The interface control processor will only allow one of the redundant WFTA processors to be "active" and transferring the computed data set information to memory. Additional information on fault tolerance in the WFTA pipeline processor can be obtained by referencing Capt. Coutee's thesis [4].

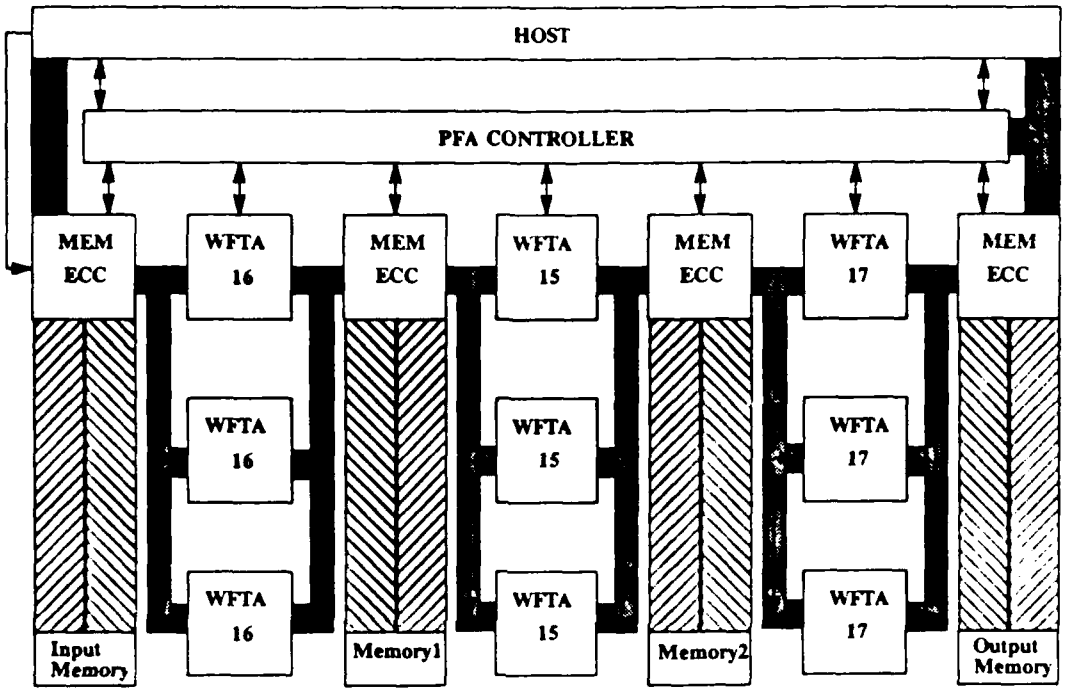


FIGURE 2-4. Active/Watchdog Processors [3].

Each of the eight separate memory banks has the capacity to store 4080 complex data points with a word length of 48 bits. The control of each memory bank is accomplished through a memory controller and the PFA interface processor (See Figure 2-2). The controllers alternate the input and output banks of XROM addressed memory for each of the WFTA PFA processor operations. For example, the initial input bank of memory for the 16-point WFTA processor is the A bank of memory. However, during the

first WFTA 16 operation on the input data in bank A, the host processor is loading the data for the second WFTA 16 operation into bank B. The first WFTA 16 operation puts the data results from the execution of the 16-point DFT into the C bank of memory. Upon completion of the first WFTA 16 PFA processor operation, the memory controllers immediately (when directed by the PFA interface control processor) alternate memory banks A & B and C & D. Now the PFA interface controller for the PFA pipeline processors begins the execution of the first WFTA 15 operation and the second WFTA 16 operation. The WFTA 16 PFA processor is now receiving input data from the B bank of memory and transmitting the results to bank D. While at the same time, the WFTA 15 PFA processor is receiving input data from the C bank of memory (the results from the first WFTA 16 PFA operation) and putting the results from the execution of the 15-point DFT into the E bank of memory, and the host processor once again transmits input data for the third WFTA 16 operation into the A bank of memory. Therefore, the PFA interface control processor and the memory controllers continue to alternate the input and output data banks of memory for each execution of the DFTs in the WFTA PFA processors. For a more complete and detailed description of the WFTA PFA processor, memory controller operations, and DFT execution reference Capt. Taylor's thesis [18].

2.4 VHDL Support Environment

The VHDL support environment is composed of an assorted collection of user support software and system data designed to assist VLSI design managers and engineers in making the most efficient use of the design capabilities of the VHDL language. The contracting team of Intermetrics,

Texas Instruments, and IBM have developed the initial beta releases of the VHDL support environment for implementation on a standard VAX/VMS operating system and computer hardware configuration. The software configuration of the VAX must include Version 4.2 (or later) of the VMS operating system and the Ada software support required to create an Ada runtime support library for the VHDL language analyzer. The design tools and system data in the VHDL support environment provide four basic system capabilities: 1) the VHDL design verification of the static and dynamic language semantics, lexical constructs, and language syntax of a VHDL source code modeling description for a hardware component; 2) the design library support management for the revision and cataloguing of segments (design units) of a hardware component design; 3) the simultaneous access of multiple design managers and engineers to VHDL design support data, design tools, and design libraries; and 4) the system capability to provide a basic foundation for the future addition of VHDL design automation tools [5].

The VHDL support environment is constructed of the six major design components illustrated in Figure 2-5. The figure shows the VHDL system configuration and data flow between the major components. Five of the major components of the VHDL support environment are VHDL design tools, these include the VHDL language analyzer, the VHDL reverse analyzer, the VHDL simplifier, the VHDL simulator, and the VHDL design library manager. The sixth component is the designated repository for the design data, the VHDL design library.

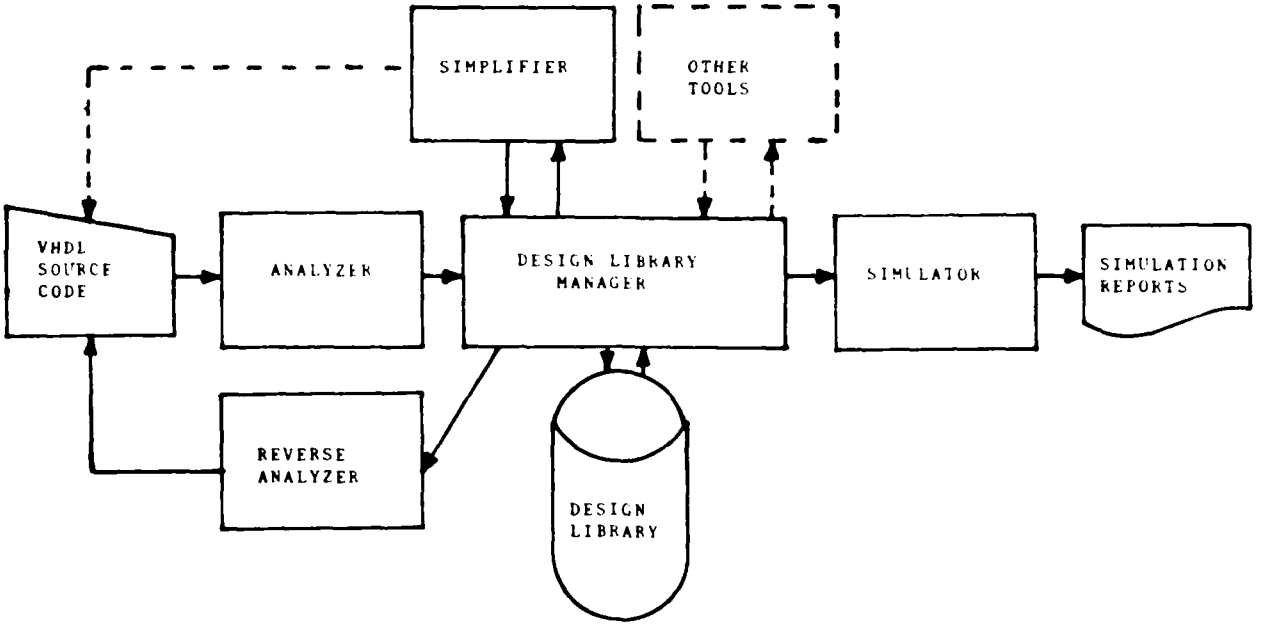


FIGURE 2-5. The VHDL Support Environment [5].

The initial beta releases of the VHDL support environment have been delivered to AFIT for installation, test, and evaluation. The design tools provided in the initial beta releases of the support environment include; the VHDL language analyzer, the VHDL reverse analyzer, the VHDL simulator, and the VHDL design library manager. However, due to language restrictions and time constraints on the amount of thesis research that can be accomplished, the VHDL reverse analyzer will not be evaluated in this thesis. The following subsections will describe the functions of all six of the major components of the VHDL support environment, and will describe in detail the VHDL language analyzer, the VHDL design library manager, the VHDL design library, and VHDL simulator tested at AFIT.

2.4.1 VHDL Language Analyzer

The VHDL language analyzer is one of the four primary components of the VHDL support environment to be tested and evaluated by this thesis. The analyzer performs a static error check of the modeled VHDL hardware descriptions. The static error check includes a lexical check of the VHDL source code modeling descriptions, a language syntax check of the VHDL source code against the construct rules of the VHDL language, and a static analysis of the language semantics in the VHDL source code. The semantic static analysis of the VHDL source code modeling descriptions is based upon the previously analyzed design units called in the code [5].

If the analyzer static error analysis yields no errors in the VHDL source code, then the language analyzer translates the VHDL source code modeling descriptions of the hardware component into the Intermediate VHDL Attributed Notation (IVAN) form and places the results into the VHDL design library as a new design unit. The VHDL analyzer also produces a detailed listing that documents the static error analysis of the specific hardware component being analyzed.

The language analyzer listing provides the user with information about the errors detected in the VHDL source code being analyzed (if there are any), an echo of the actual VHDL source code, a mapping of what design units were created in the design library, a mapping of what design units the analysis depends on, information on the processing and runtime statistics of the language analyzer, and a listing of what symbols and parameters that were used by the language analyzer. The actual analyzer listings for the VHDL source code modeling descriptions of the WFTA 16

PFA processor are available in Appendix C, while the actual VHDL source code modeling descriptions are available in Appendix B. The structure of the analyzer listing can also be altered to provide only the specific information desired by the user. The restructuring of the VHDL analyzer listing is accomplished through the use of the VHDL library system, which is a set of functions available to the design library manager. For a more detailed description on the use of the VHDL language analyzer and analyzer listing options, reference the VHDL Analyzer User's Manual [5].

If the analyzer static error analysis detects errors in the VHDL source code descriptions, then the analyzer creates a detailed analyzer listing showing the location and type of errors detected in the source code. In addition, the design unit in which the errors were detected is not placed into the VHDL design library. The errors should then be corrected by using the text editor of the host system, such as the VAX EDT editor. The errors detected during the static analysis of the VHDL source code descriptions initiate error messages at the user's terminal that inform the user of the errors detected during the execution of the language analyzer. The functions of the analyzer are demonstrated in Figure 2-6. A detailed listing of the analyzer errors is presented in the VHDL Analyzer User's Manual and can be referenced during the actual execution of the language analyzer to give a user a better understanding of the type of errors encountered [5].

2.4.2 VHDL Design Library Manager

The VHDL Design Library Manager (DLM) is the second of the four primary components of the VHDL support environment tested and evaluated by this thesis. The DLM is the backbone of the VHDL support environment,

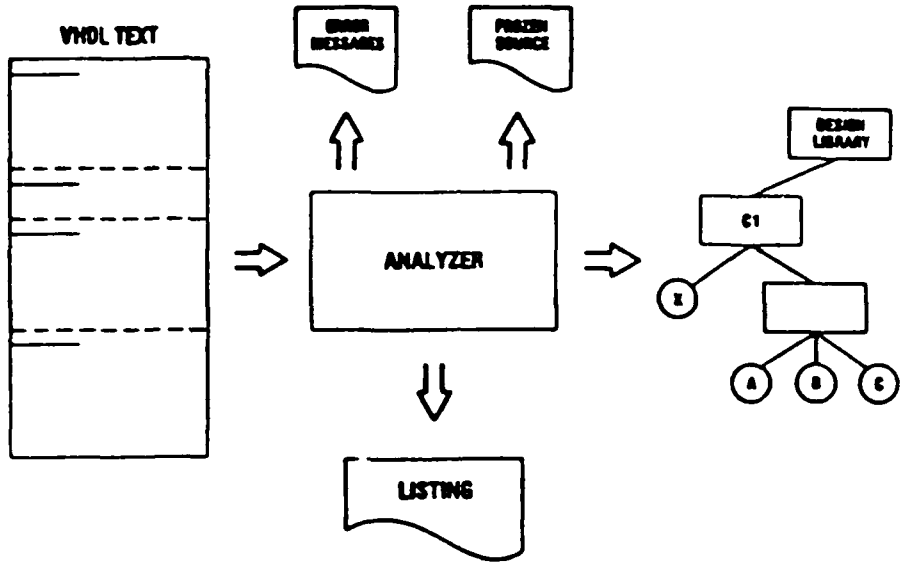


FIGURE 2-6. The VHDL Language Analyzer [5].

and defines the integration framework for communication of VLSI design information between the VHDL design tools and the VHDL design library. The DLM also provides the VHDL user software support required to access, manipulate, and manage the IVAN design units created and stored in the VHDL design library. A collection of the four VHDL software packages described below and illustrated in Figure 2-7 combine to form the VHDL DLM [9].

1. VHDL Library System (VLS) -- The VLS is a set of user oriented functions designed to allow the VHDL user to interactively examine and manipulate the IVAN design units in the VHDL design library. The VLS commands are available to the user at the host command level and indirectly access the VHDL design library through the utility operations software package. A detailed description of the VLS commands available to the user is presented in the VHDL Design Library User's Manual/Implementor's Guide [9].

2. Utility Operations -- The utility operations package of the DLM provides the software support of the interface between the VHDL design tools and the VHDL design library. The utility operations software is designed and defined by an Ada package constructed of numerous subprograms. Each subprogram implements a common VHDL user oriented function required for the general use and manipulation of the design library by the VHDL support environment design tools.

3. IVAN Operations -- The IVAN operations package provides the software support required for the access of the abstract form of the data contained in the design library nodes, see Figure 2-7. The IVAN operations software is defined by an Ada software package that communicates with the VHDL design tools through the interface with the DLM utility operations package.

4. Library Structure Operations (HIF) -- The library structure operations software is a set of Ada packages used to establish the hierarchical structure of the VHDL design library. The DLM library operations package is accessed by the VHDL design tools in the VHDL support environment through the DLM utility operations package. The

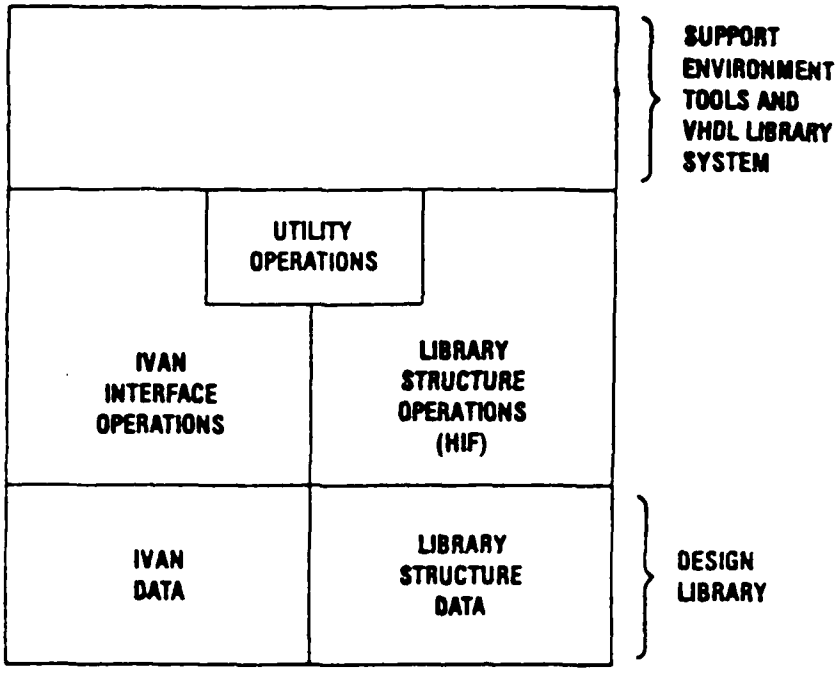


FIGURE 2-7. Role of the Design Library Manager [9].

VHDL design tools use the DLM library operations package to build, access, and manage the high-level organization of the design library nodes [9].

The interface between the DLM and the other design tools in the VHDL support environment is not tool dependent. The DLM interface with the VHDL design tools remains constant and is not modified as new VHDL design

tools are added to the VHDL support environment. Furthermore, as shown in Figure 2-7, no VHDL design tool has direct access to the VHDL design library. All design tool access to the design library occurs through the utility operations package, that has direct access to the IVAN operations package and the library structure operations package. The design library is then directly accessed by the IVAN operations package or DLM library structure operations package. Specific details on the DLM are available in the VHDL Design Library User's Manual/Implementor's Guide [9].

2.4.3 VHDL Simulator

The VHDL simulator is the third of the four primary components of the VHDL support environment tested and evaluated by this thesis. The simulator computes the dynamic behavior of a hardware component modeling description from the IVAN representations in the VHDL project design library nodes. The VHDL simulator is composed of three software programs provided to the user, the model generator program, the report generator program, and the Simulator core Ada program library. The model generator extracts the IVAN representations of the hardware components in the VHDL design library and constructs Ada modules to be used in the creation of a simulator kernel, without destroying the IVAN representations in the VHDL design library. The report generator generates user formatted simulation reports on the signal history of those signals specified by the user. The simulator core contains software packages which implement the dynamic event driven simulation capabilities of the VHDL support environment. The Ada hardware component modules created by the model generator are compiled and linked using the simulator core library for

the simulation kernels. The simulation kernel is an executable Ada program which generates the signal mapping and signal trace files used by the report generator. The report generator then creates a simulation report based on the user parameters defined in the report control file by using the signal mapping and signal trace files [6].

2.4.4 VHDL Design Library

The VHDL design library is the last of the four primary components of the VHDL support environment evaluated by this thesis. The design library is the system design data repository, shared by all design tools, for the IVAN representations of the analyzed VHDL design units. The design library supports the hierarchical structure of the VHDL language and the incremental analysis of a large-scale component partitioned into smaller design subcomponents.

The hierarchical structure of the VHDL language enables a user to break a hardware component into a combination of subcomponents, and then break the subcomponents into a combination of sub-subcomponents, etc. Continuing to break the components into smaller subcomponents, creates a tree type structure of design library nodes. Figures 2-8 and 2-9 present a user's view of the hierarchical structure of the design units in the VHDL design library. As seen from these illustrations, the structure of the design library supports more than one description of a particular design entity. Additional information on the structure and building of the VHDL design library is available in the VHDL Design Library User's Manual/Implementor's Guide [9].

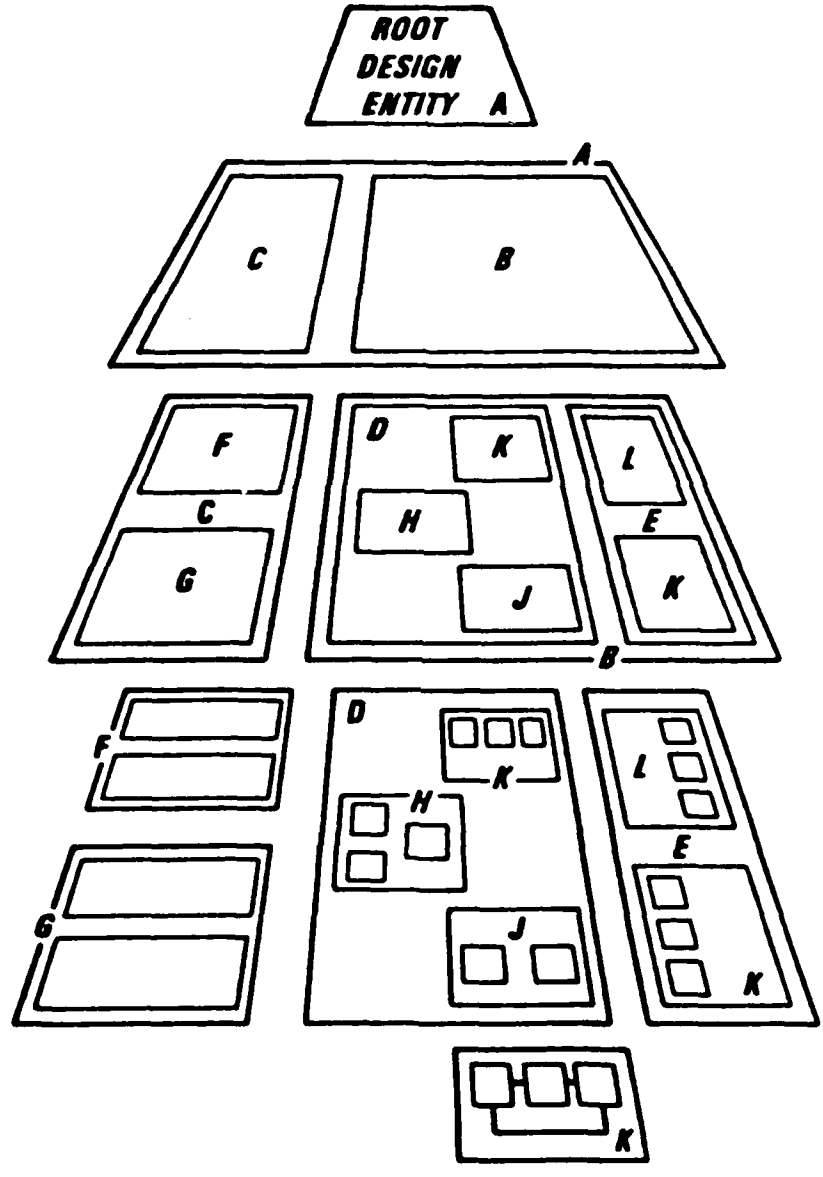


FIGURE 2-8. VHDL Supports Hierarchical Descriptions [9].

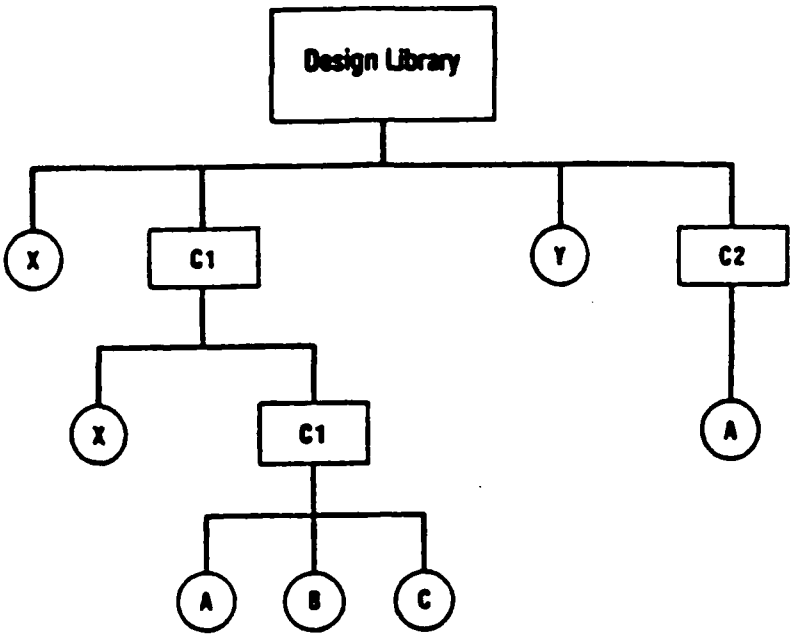


FIGURE 2-9. Example of the Design Library Organization [9].

2.4.5 VHDL Reverse Analyzer

The VHDL reverse analyzer is not tested or evaluated in this thesis; however, a brief description of the purpose of the reverse analyzer is provided for completeness of the VHDL support environment. The VHDL reverse analyzer constructs a VHDL source code modeling description from the IVAN representations in the design library. The reverse analyzer supports the analysis, modification, and redesign of VLSI class hardware components [5,9].

2.4.6 VHDL Simplifier

The VHDL simplifier is not tested or evaluated in this thesis. The simplifier merges design modules of a modeled hardware component into a single hardware description module. The new module then forms a complete modeling description of the entire hardware component and replaces the the previous hierarchical description in the design library with a single module [5,9].

Chapter 3

VHDL Modeling of the WFTA 16 PFA Processor

3.1 Overview

As previously stated, the structural architecture of the WFTA 16 PFA pipeline processor is built around the sequential application of the three PFA processors to compute the 16, 15, and 17-point DFTs. Each of the three WFTA PFA processors used to implement the 4080-point WFTA PFA pipeline processor architecture is constructed of the same basic hardware components. The major components of the WFTA processors at the register level include the input and output registers (PISO and SIPO) and the arithmetic circuitry cells (pre-addition array, multiplier array, post-addition array, parity check/zero fill cells, and parity/round cells). The major register level components of the WFTA 16 PFA processor are illustrated in Figure 3-1. The primary differences between the 16, 15, and 17-point PFA processors is in the number of arithmetic operations and the height of the multiplication array. The basic architectural structure of the major register level components is the same [3]. As a result of the similar processor architectures, only the WFTA 16 PFA processor will be modeled. This chapter presents the basic steps applied to accomplish the modeling of the hardware components and the building of the VHDL design library to support the modeling and simulation of the WFTA 16 PFA processor.

The chapter begins with a section describing the successive steps required to perform a top-down decomposition of the WFTA 16 PFA processor into its basic structural elements. The actual modeling of the hardware

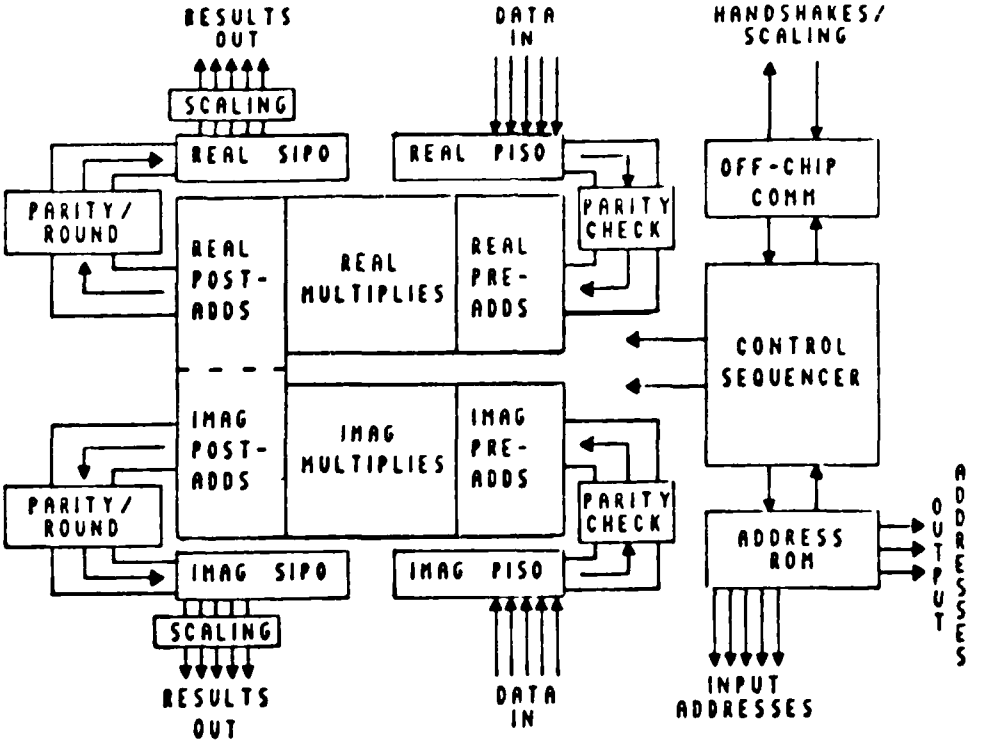


FIGURE 3-1. Winograd Processor Architecture [3].

components with the VHDL language will be accomplished by creating VHDL source code descriptions of the basic structural elements and then combining these descriptions at successively higher levels of abstraction to eventually form the VHDL descriptions of the major register level components. The second section of the chapter will be dedicated to the

definition of the format used to create the VHDL source code descriptions of the WFTA 16 PFA processor hardware components. While, the last section of the chapter describes the direct application of the bottom-up concepts of design to the analysis of the VHDL source code descriptions and the development of the VHDL design library to support the simulation of the WFTA 16 processor.

3.2 Top-down Decomposition of the WFTA 16 PFA Processor

The first step in the VHDL modeling of the WFTA 16 PFA processor is to perform a top-down decomposition of the processor into the major register level components that perform the computation of the 16-point DFT. The top-down decomposition of the processor at the register level actually defines the signal flow characteristics between the major system components at this level. Once the signal flow characteristics have been determined, the interface specifications for the major components are defined and the VHDL design entity interface requirements for the VHDL source code descriptions are extracted from the interface specifications. Therefore, the VHDL design entities for the major components at the register level are actually pre-defined by the architectural structure and basic signal flow characteristics of the processor. Before beginning the top-down decomposition of the major register level components shown in Figure 3-1, it is important to describe the actual signal flow and operation of the WFTA 16 PFA processor. The description begins with the input of the data into the PISO register and culminates with the output of the data from the SIPO register.

3.2.1 Signal Flow and Operation of the WFTA 16 PFA Processor

At the register level the WFTA PFA processor can be described as a bit-serial machine. As presented in Figure 3-1, the major processing components include the input and output registers and the arithmetic circuitry. However, as illustrated in Figure 3-1 the WFTA processor is divided into two separate, but complete and identical computational sections. The real and imaginary sections of the WFTA processor are independent mirror images of one another through the last column of the post-addition array. In the last column of the post-addition array the computations on the real and imaginary data sets are combined to form the complex outputs of the 16-points DFT. As a result of the mirror image structural architecture of the two computational sections, only one of the two sections will be modeled and described in this thesis. However, it must be emphasized that the real and imaginary sections of the WFTA processor perform exactly the same operations, at the same time, in the same sequence, using only different sets of data from the input banks of the off-chip memory [3]. Figure 3-2 presents a register level view of a single section (real or imaginary) of the WFTA 16 PFA processor and best illustrates the processor hardware which will be modeled with the VHDL language in this thesis.

The parallel input register in Figure 3-2 is a Parallel-In, Serial-Out (PISO) register 24 bits wide and 16 words deep (24 x 16). The input data word is provided by an off-chip input memory bank and provides a data word 24 bits wide with 23 data bits and 1 parity bit. Every other clock cycle the PISO receives a new data word from one of the two input

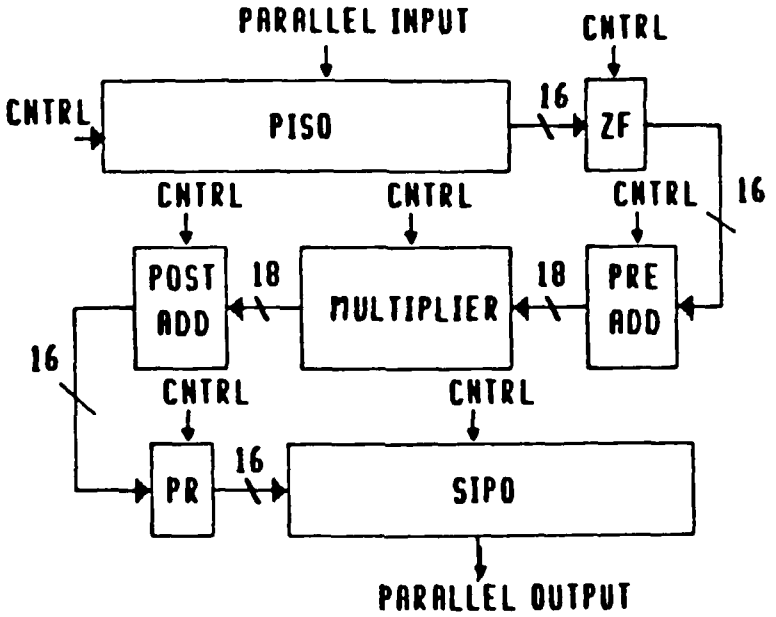


FIGURE 3-2. Decomposition and Signal Flow of the WFTA 16 Processor [3].

banks of memory addressed by the XROM. The control signal SD_PISO is used to shift the 24 bit data word input at the top row of the PISO down by one row. Therefore, after the input of 16 data words (or 32 clock cycles) the PISO is completely filled with input data and must now be shifted into the serial registers of the PISO. The control signal LATCH_PISO is forced high and the transfer of the word-parallel data into the serial shift registers of the PISO occurs at this time. Once the transfer of the data from the parallel registers of the PISO is

complete, the parallel registers are now free to receive the next 16 data words. The PISO will continue to input the data in blocks of 16 data words into the parallel registers and shift the data into the serial registers and out into the pipeline as long as the OPERATE control signal remains high [3].

The 16 data words in the serial registers of the PISO are now ready to be shifted into the arithmetic circuitry cells of the pipeline. As long as the SR_PISO control signal of the PISO stays high, the 16 data words are shifted one bit at a time, starting with the least significant bit (LSB) first, into the processor pipeline. To allow for the numerical growth required in the arithmetic circuitry of the processor, the data must be extended to a 32-bit word length in the Parity Check/Zero Fill (PC/ZF) cell. The PC/ZF cell also performs an odd-parity error check on the data and strips the parity bit from the data words. The zero fill section of the PC/ZF cell inserts zeros prior to the LSB to scale up the data and enhance the signal to noise ratio. The sign extensions required for a specific data set are appended after the most significant bit to prevent an arithmetic overflow problem in the computational array sections of the processor pipeline. For a more detailed register level description of the operations performed by the PFA processor the reader is referred to Capt. Coutee's [4], Capt. Rossbach's [16], and Capt. Taylor's [18] 1985 theses.

The actual number of zero fills and sign extensions required for a specific data set is determined by the adaptive scaling algorithm. The algorithm considers the relative magnitude of the input data set

and determines the necessary adjustments required to prevent arithmetic overflow in the computational sections of the processor. Each and every 4080-point data set inputted by the Host computer into the WFTA PFA pipeline processor has an associated scale factor. The scale factor reflects the magnitude of the largest number in the entire data set. The scale factor is also the smallest number of sign extensions that must be applied to any number in the data set. Thus, to avoid the possibility of arithmetic overflow, the largest number (a scale factor 0) requires a minimum of five sign extensions. If a 4080-point data set contains only very small numbers, the processor can replace sign extensions with zero fills to enhance the overall numerical accuracy of the WFTA PFA pipeline process [3].

The arithmetic circuitry cells that form the pre-addition array, the multiplier array, and the post-addition array actually implement the 16-point DFT. The multiplicand for the multiplier array is generated from the output of the PC/ZF cells by the pre-addition array. The cells of the pre-addition array require a maximum of four sequential addition/subtraction operations. All multiplicands generated in less than four addition/subtraction operations must remain aligned with the other data elements in the processor pipeline. Therefore, those positions in the pre-addition array, and also the post-addition array, that do not require an adder/subtractor (A/S) cell must be replaced with a one-delay wide Master-Slave Flip-Flops (MSFFs). Most of the circuit elements in the WFTA 16 PFA processor require an input Phi2 latch and an output Phi1 latch; however, the PC/ZF and A/S cells are exceptions to this rule. The PC/ZF cell requires only a Phi2 latch that is preceded by some

additional combinational logic. While the A/S cells are exactly the reverse, they require data that enters on the Phi1 latch and leaves on the Phi2 latch [3].

A pipeline view of the pre-addition array for the WFTA 16 PFA processsor is presented in Figure 3-3. As seen in Figure 3-3, the pre-addition array is only constructed from three columns of A/S cells and MSFFs. As previously stated, some multiplicands require a minimum of four addition/subtraction operations; however, only two of the multiplicands (data words) require four A/S cell columns. But these two data words do not have any other arithmetic operations performed on their bit streams, they travel through the remainder of the pipeline by passing through only single-delay MSFFs in the post-addition array. Furthermore, the sum and difference of these two bit streams only pass through trivial multipliers (x1) in the multiplier array. Since the two data streams do not perform any arithmetic operations in the post-addition array and the multiplier array equates to a multiplication by one, it is possible to delay the fourth column addition/subtraction operations required in the pre-addition array until the first column of the post-addition array. This eliminates the entire fourth column of the pre-addition array and reduces the total pipeline latency by one clock cycle and also eliminates 35 MSFFs. As illustrated in Figure 3-3, the PC/ZF cell and pre-addition array introduce four cycles of latency into the processor pipeline [3].

The multiplier array of the WFTA 16 PFA processor consists of a [18 X 14] array of multiplier cells. The 28 bit Winograd coefficients are encoded into fourteen multiplier cells using Booth's quaternary

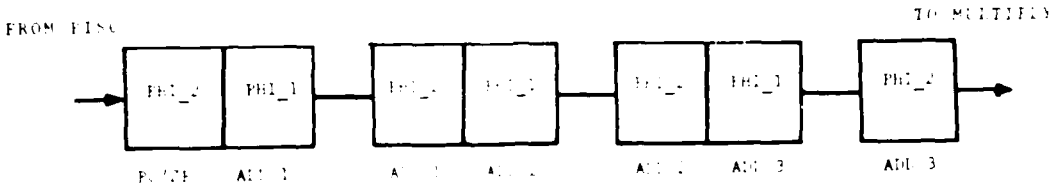


FIGURE 3-3. Pre-addition Pipeline [3].

encoding algorithm. Therefore, each bit of the reduced coefficient represents a single bit multiplier cell of the serial multiplier array. Each of the multiplier cells requires a latency of 3 clock cycles and the entire multiplier array requires a total latency of 41 clock cycles (3 X 14 - 1) for the pipeline processor [3].

The post-addition array of the WFTA 16 PFA processor arithmetic circuitry requires three columns of A/S cells. The pipeline view of the post-addition array and parity/round cells is presented in Figure 3-4. The first column of the post-addition array includes the deferred fourth column operations for the two bit streams from the pre-addition array. The first two columns of the post-addition array perform the independent operations on either the real or imaginary data streams. The third column combines the two independent data streams and produces the complex outputs of the 16-point DFT operation. The post-addition pipeline view also contains the parity generation and arithmetic rounding (PR) cell. The PR cell rounds the 32 bit results from the arithmetic arrays to 23

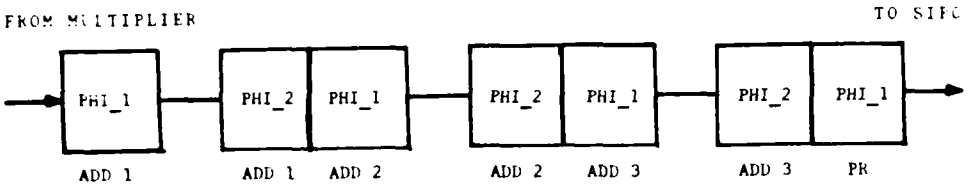


FIGURE 3-4. Post-addition Pipeline [3].

data bits and calculates the odd parity bit requirement for these 23 bits and appends this to the 23 bits to create a 24 bit data word. Thus, the post-addition array and PR cell introduce a latency of 3 1/2 clock cycles to the total latency for the WFTA pipeline processor [3].

The output of the post-addition array is input into the Serial-In-Parallel-Out (SIPO) register. The SIPO has the same basic structure as the PISO register. The data for the SIPO enters bit serial and is output word (24 bit) parallel. Once the MSB (parity bit) has been shifted into the SIPO, the control signal LATCH_SIPO is forced high and the data is shifted from the serial registers into the parallel registers of the SIPO. Every other clock cycle produces an output parallel data word that is sent to the off-chip memory location addressed by the XROM. The XROM is designed for the processor memory address generation and is an element of the control circuitry of the WFTA 16 PFA processor, which will not be modeled by this thesis effort [3].

3.2.2 Decomposition of the PISO and SIPO Registers

As previously stated, the basic structural architecture of the PISO and SIPO registers is the same. The registers form input and output data registers with a word length of 24 bits wide and 16 words deep. The PISO register has 24 parallel input data ports and 16 serial output data ports. While the SIPO register reverses the input and output functions and creates a register that has 24 parallel output data ports and 16 serial input data ports. The PISO and SIPO registers are controlled by a two phase clock and three control signals. When the individual PISO and SIPO registers are decomposed into the next lower level of hardware structural elements, each of the registers is viewed as an array (24 X 16) of PISO and SIPO cells. Both the PISO and SIPO cells of the PFA processor are constructed of the same basic hardware elements as shown in Figures 3-5 and 3-6.

The PISO and SIPO cells each have a parallel and serial input data port, a parallel and serial output data port, a two phased clock signal, and three control signals. The three control signals in the PISO cell control the parallel shift down of the data word to the next lower word level, the latching of the data into the serial registers in the cell, and the serial shift right of the data by one bit into the arithmetic circuitry pipeline. While, the three control signals in the SIPO cell control the serial shift right of the data by one bit out of the pipeline, the latching of the data into the parallel registers in the cell, and the parallel shift up of the data word to the next higher word level. As illustrated in Figures 3-5 and 3-6, the individual PISO and SIPO cells are constructed from three uni-directional transmission gates and two

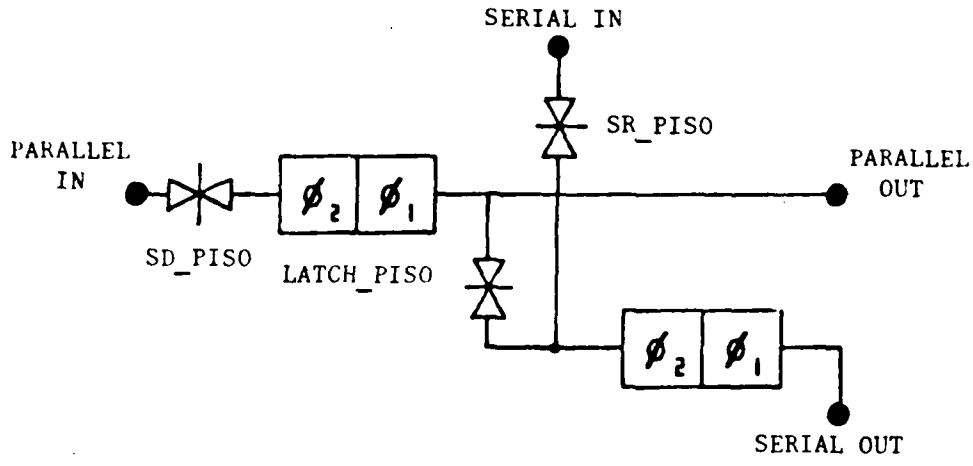


FIGURE 3-5. PISO Cell.

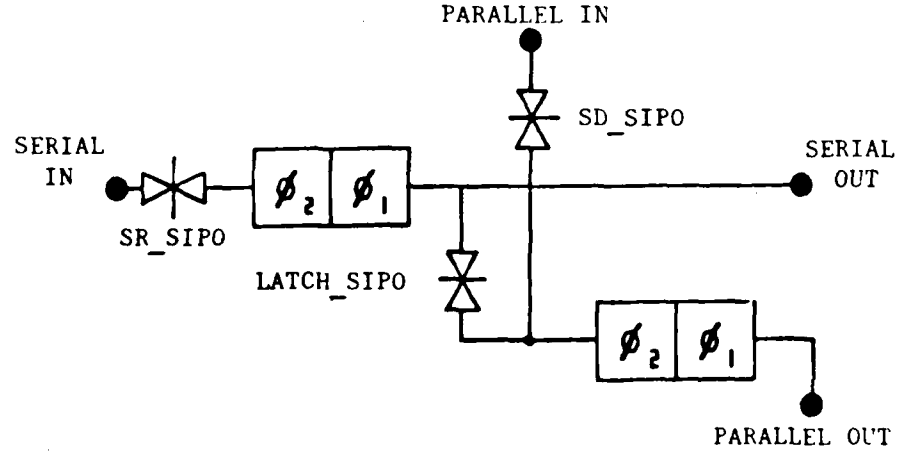


FIGURE 3-6. SIPO Cell.

Master-Slave Flip-Flops (MSFFs). The transmission gates are modeled as single direction gates because the data flow directions are pre-defined by the architectural design structure. The transmission gate is also defined as a basic structural hardware element of the PFA processor and is one of the primary building block elements. As one of the lowest level components modeled, the VHDL source code modeling descriptions be implemented using a VHDL behavioral architecture description.

The MSFFs can be further decomposed into two serial CMOS latches clocked by the Phi1 and Phi2 inputs from the two phase clock. The CMOS latch is illustrated in Figure 3-7 and is constructed from a transmission gate, a tri-state inverter, and a clocked inverter. The transmission gate and tri-state inverter are basic structural hardware elements and are modeled using a VHDL behavioral architecture description. While the clocked inverter is further decomposed into a transmission gate and inverter. The inverter will also be defined as a primary building block for the modeling of the WFTA 16 PFA processor and as a basic structural hardware element will be modeled using a VHDL behavioral architecture description. Thus, the entire PISO and SIPO registers are described by a combination of only three basic structural hardware elements; the uni-directional transmission gate, the tri-state inverter, and the inverter. VHDL modeling descriptions of these three primary building blocks will be written using a behavioral architecture description. The behavioral descriptions will then be structurally combined at successively higher levels of abstraction to form the structure of the individual PISO and SIPO cells. The PISO and SIPO cells are then structurally combined to form the [24 X 16] array that defines the PISO and SIPO register.

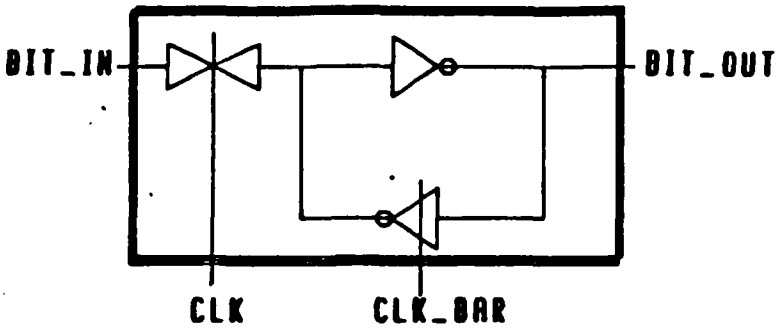


FIGURE 3-7. Clocked CMOS Latch [3].

Additional information and details on the modeling of the clocked CMOS latch and MSFF is contained in Capt. Collins' [3] and Capt. Coutee's [4] theses.

3.2.3 Decomposition of the Arithmetic Circuitry

The arithmetic circuitry is composed of the pre-addition array, the multiplier array, the post-addition array, the parity check/zero fill cells, and the parity/round cells. Any single section of the processor arithmetic circuitry can be viewed as a continuous operation on a single bit slice of a 32 bit vector. Once a data word enters the PISO, each bit of the word is associated with 15 other data bits from the same bit position in the other 15 data words of the PISO. This bit slice alignment must be maintained throughout the entire latency period of 119 clock cycles required for the processor to complete the pipeline signal processing.

The arithmetic circuitry for the PFA processor can be structurally decomposed into 23 parallel columns of functional computational elements, as illustrated in Figure 3-8. The height of each column represents the number of bit streams crossing the interface. Each of the columns is structurally decomposed into a single bit wide array of computational elements. In the pre-addition array the columns are composed of A/S cells and single delay MSFFs. The multiplier array is constructed of five multiplier elements (+2, +1, 0, -1, -2). While, the post-addition array is constructed of A/S cells and MSFFs like the pre-addition array.

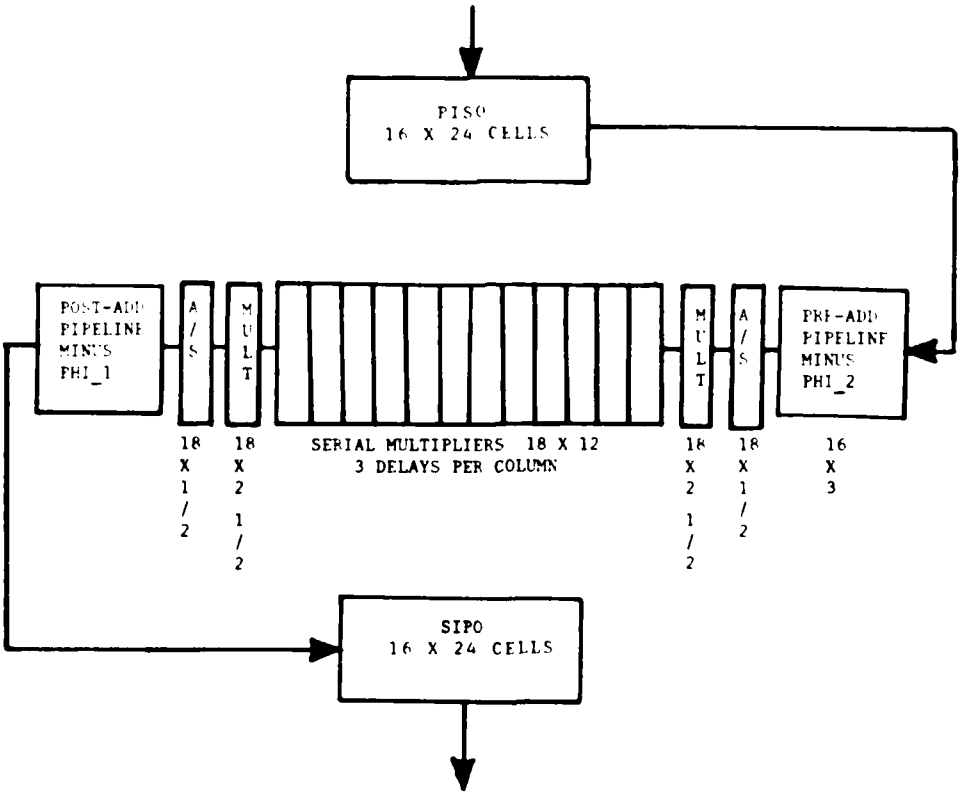


FIGURE 3-8. Column Form of the WFTA 16 PFA Processor [3].

The A/S cells and MSFFs of the pre- and post-addition arrays are decomposed into primary building block elements like the PISO and SIPO cells. As previously stated, the MSFFs and resettable MSFFs (RMSFFs) are constructed from combinations of three basic hardware elements; the uni-directional transmission gate, the tri-state inverter, and the inverter. The RMSFFs have a similar structural architecture to the MSFFs. The only difference is that the RMSFFs are constructed from the resettable CMOS latch shown in Figure 3-9. Therefore, the VHDL modeling descriptions are slightly different, but are still composed of the same basic structural hardware elements. The A/S cells are decomposed into 2 RMSFFs, 5 CMOS latches, an inverter, and a full adder/subtractor. The RMSFFs, CMOS latches, and inverter are constructed of the three basic structural hardware elements previously described (transmission gates, tri-state inverters, and inverters). However, the full adder/subtractor must be defined as a new basic structural hardware element and primary building block. The full adder/subtractor will be modeled by using a VHDL behavioral architecture description. Therefore, the pre- and post-addition arrays are structurally decomposed into four primary building block elements; the uni-directional transmission gate, the tri-state inverter, the inverter, and the full adder/subtractor. The exact combinations of the A/S cells and MSFFs in the pre- and post-addition array columns is described in detail in Capt. Coutee's thesis [4].

The [14 X 18] multiplier array is structurally decomposed into a combination of five different multiplier cells. The multiplier cells include the positive two multiplier cell, the positive one multiplier cell, the zero multiplier cell, the negative one multiplier cell, and

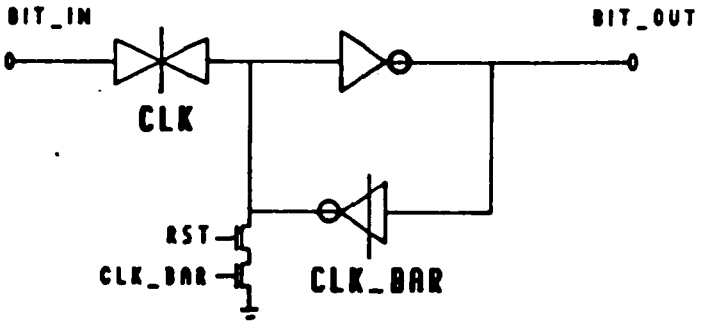


FIGURE 3-9. Resettable CMOS latch [3].

the negative two multiplier cell. The zero multiplier cell is decomposed into 4 MSFFs, which as previously described are constructed from three basic structural elements. The remaining multiplier cells (+2, +1, -1, and -2) are each decomposed into different structural combinations of 4 MSFFs, a RMSFF, an inverter, and a full adder/subtractor. Therefore, just as in the pre- and post-addition arrays, the multiplier cells are constructed from the four basic structural elements previously described; the uni-directional transmission gate, the tri-state inverter, the inverter, and the full adder/subtractor. The exact combination of multiplier cells required to form the Booth's quaternary representation of the multiplier array for the WFTA 16 PFA processor is described in detail in Capt. Coutee's thesis [4].

The parity/round and parity check/zero fill columns of the processor arithmetic circuitry are decomposed into parity/round (PARRND) cells and parity check/zero fill (PARZER) cells. The PARRND cell is structurally decomposed into a combination of 2 CMOS latches and 2 MSFFs. While the PARZER cell is decomposed into a combination of 2 transmission gates, a MSFF, and a Set-Reset Flip-Flop (SRFF). Each of these elements, except the SRFF, has previously been described in terms of its basic structural hardware elements. The SRFF is structurally decomposed into 3 CMOS latches. Therefore, the PARRND and PARZER cells can also be decomposed into three primary building block hardware elements; the uni-directional transmission gate, the tri-state inverter, and the inverter. Again, the specific details on the actual cell structure and columns structures are described in detail in Capt. Coutee's thesis [4].

Thus, the PISO/SIPO registers and the processor arithmetic circuitry are modeled in the VHDL language by using structural combinations of the four primary building block elements; the uni-directional transmission gate, the tri-state inverter, the inverter, and the full adder/subtractor. The VHDL source code modeling descriptions of these basic elements are contained in Appendix B, along with the structural combinations of these elements used to describe and model the major register level components of the WFTA 16 PFA processor.

3.3 VHDL Format for the WFTA 16 PFA Processor

The format used to document the VHDL source code descriptions for the modeling of the WFTA 16 PFA processor is shown in Figures 3-10 and 3-11. As shown in the uni-directional transmission gate example, the

```
--**************************************************************
--
--
--      DATE:  9 September 1986
--      VERSION: 2.0
--      TITLE: Transmission Gate
--      FILENAME: T_GATE.VHD
--      COORDINATOR: Capt. Charles H. Cooper
--      OPERATING SYSTEM: VMS
--      LANGUAGE: VHDL
--
--      ENTITY:
            with package WFTA_DECLARATIONS;
            use WFTA_DECLARATIONS;

            entity T_GATE
             (bit_in: in Z_BIT;
              control: in CNTRL;
              bit_out: buffer Z_BIT) is

            generic (tdelay: TIME := 0ns)

            end T_GATE;
--
--      FUNCTION:
--              This is a behavioral description of a
--      transmission gate. It actually needs both senses
--      of the control signal to drive the CMOS 'P' and
--      'N' transistors but since the inverted signal
--      does not perform any separate function it is not
--      included in the port list or architectural
--      description.  The T_GATE is sensitive to both the
--      input and control signals.  However if control =
--      '0', then the output will be not change regardless
--      of the value of the input.  The process statement
--      reflects this consideration.  If the control has
--      not just changed to '1' the output will not
--      reflect the input.  As soon as control switches
--      to '1' then the input will be enabled.  As long
--      as the control remains high the output will
--      reflect the input, when it falls the input signal
--      will be disabled and not be allowed to cause
--      events in the Transaction queue.
--
--**************************************************************
```

FIGURE 3-10. Interface Declaration Format.

```
architecture BEHAVIOR of T_GATE is

   BEHAVIOR_BLK:

 block

   begin

     process (bit_in , control)


     begin


--    if (control = '1' and not control'stable) then
--      enable bit_in;
--    end if;
--
--    if (control = '0') then
--     disable bit_in;
--    end if;

      if (control = '1') then
          bit_out <= bit_in;
      end if;

   end process;

 end block;

end BEHAVIOR;
```

FIGURE 3-11. Body Declaration Format.

format of the VHDL source code modeling description is divided into two sections; the interface declaration section and the body declaration section(s). The interface declaration section contains specific file information on a hardware element, the design entity description of the hardware element, and a functional description of the hardware element. The specific file information for the hardware element includes the origination date, version number, title, filename, coordinator, project name, operating system, and description language. The VHDL design entity description of a hardware element defines the element interfaces, the use of additional packages, and the use of any special assertion statements required for the element. The functional description provides the user with a top level functional description of a hardware element and provides an insight into the operation of the element.

The body declaration section(s) contains the actual VHDL source code for the specific hardware element defined in the design entity. The body declaration section is defined by the VHDL language documentation as either an architectural declaration or a configuration declaration. The WFTA 16 modeling descriptions all use an architectural declaration to define the behavior, architecture, or structure of a specific design entity. A VHDL modeling description can contain more than one type of architectural declaration, but must contain only one design entity description as illustrated in the SRFF VHDL source code description in Appendix B. Therefore, all VHDL source code modeling descriptions for the WFTA 16 PFA processor will use the interface and body declaration format described.

3.4 Bottom-up Composition of the VHDL Design Library

The VHDL design library for the WFTA 16 PFA processor is created by the VHDL language analyzer through the bottom-up analysis of the VHDL source code modeling descriptions. At AFIT, the VHDL support environment software required to perform the VHDL source code analysis is installed on the CSC VAX/VMS operating system under the DATAMGR VMS user name. The DATAMGR user directory is divided into two subdirectories for the VHDL system installation. The DESIGN_LIBRARY subdirectory is created to hold the data repositories required for the development of the multiple VHDL project design libraries. While the VHDL2_SYSTEM subdirectory is created to hold the VHDL system software required for the analysis and simulation of the modeling descriptions. In addition, the VHDL support environment requires the addition of an *execution command* call in the user LOGIN.COM file for the execution of the VHDL system VHDL_UTMC5_LOGIN.COM file. The VHDL support environment is designed to allow the user to create multiple design libraries. The individual design libraries are created to reduce the effect of possible design library corruption. Therefore, the AFIT VHDL support environment was designed to support three VMS project user accounts (USER1, USER2, and USER3). Each of the three user accounts has its own individual project design library data repositories located in the DATAMGR.DESIGN_LIBRARY subdirectory. The individual project design libraries hold the IVAN representations for each design project and are created by executing the VHDL design library manager ADD_USER command of the VHDL library system (VLS) software. The source code descriptions for the WFTA 16 PFA processor are located in the <<USER1>> design library.

The PFA processor VHDL source code modeling descriptions are created and stored in the USER1 user account using any system editor. The source code is then analyzed by executing the VHDL language analyzer. The exact order of analysis for the VHDL source code descriptions is critical in the creation of the PFA processor design library. The first VHDL source code descriptions analyzed must be the VHDL packages created to support the hardware descriptions. For the WFTA processor the WFTA_DECLARATIONS package is analyzed first, this creates the IVAN representations for the new signal types and functions required for the successful description of the processor hardware. Once the VHDL packages have been analyzed the lower level basic hardware structural element descriptions are analyzed. The basic hardware descriptions include the transmission gate, inverter, tri-state inverter, AND gate, OR gate, XOR gate, and XNOR gate.

Once the basic structural hardware element descriptions have been analyzed the bottom-up concepts of VLSI design are applied to create the the structural combination of hardware elements required to create the successively higher level hardware descriptions. The bottom-up concepts of design as applied to the creation of the PISO/SIPO register will be used as an example to illustrate the order required in the analysis of the successively higher level hardware descriptions. As previously stated, the PISO/SIPO register is created from only three basic hardware elements; the transmission gate, tri-state inverter, and inverter. Once these three elements are analyzed and their IVAN representations stored in the <<USER1>> default context design library, the next higher level of hardware elements are formed. For the PISO/SIPO the next higher level

of hardware elements only consists of the CMOS latch, the latch is built with a combination of the three basic hardware elements. When the CMOS latch IVAN representation is inserted into the design library by the VHDL language analyzer the next higher level of PISO/SIPO hardware elements can be analyzed. Again the next higher level component of the PISO/SIPO register consists of only one element, the MSFF. The MSFF is composed of a combination of CMOS latches. When the analysis of the MSFF is complete the PISO/SIPO cell source code modeling description is the next higher level of hardware element analyzed. This is followed by the analysis of the PISO/SIPO row and then the final analysis of the PISO/SIPO register itself. The most important concept to remember in the construction of the design library through the analysis of the source code descriptions, is that any component called in a description must already be analyzed by the VHDL language analyzer. The only exception to this concept is the analysis of a design entity can be accomplished and used for higher level analysis of the hardware elements without the actual definition of the architecture of the element. However, if the design entity is redefined the analysis of all higher elements is voided. Therefore, as seen from the PISO/SIPO cell example the design library is created through the successive analysis of lower level elements to form the highest level elements. All other major register level hardware elements of the WFTA PFA processor are analyzed in the same manner, by applying the bottom-up concepts of design.

Chapter 4

Simulation of Simple WFTA 16 PFA Processor Components

4.1 Overview

The simulation of a hardware element using the VHDL language and VHDL language support environment is accomplished through a sequence of of five major serial steps. The five major steps include the development of a VHDL test bench for the hardware element to be simulated, the model generation of the Ada source code for each of the hardware components of of the hardware element, the compiling and linking of these Ada files into a sublibrary of the simulator core, the execution of the simulation kernel created by the compiling and linking of the Ada files, and the report generation of the user specified signals created in the execution of the simulation kernel. This chapter presents the implementation of this sequence of steps in the simulation of a VHDL behavioral modeling description of the PISO cell and the simulation of a VHDL structural modeling description of the dynamic MSFF composed of two transmission gates and two tri-state inverters.

4.2 Development of the VHDL Test Bench

The behavioral modeling description of the PISO cell is dependent on the WFTA_DECLARATIONS package due to the use of tri-state signals. A structural view of the PISO cell is illustrated in Figure 4-1. As shown in the figure and specified in the design entity of the VHDL source code modeling description in Appendix D, the PISO cell has two inputs, two outputs, three control signals, and a two phase clock. Therefore, the VHDL test bench for the PISO cell must define these signals and the VHDL

predefined component variables required by the VHDL test bench, which is located in the VHDL simulator core. The VHDL source code for the PISO cell test bench is shown in Figures 4-2 and 4-3. As seen in the source code, the test bench is the top level unit of a hardware element modeling description and defines the unique dynamic characteristics for the VHDL simulation of the element. Therefore, the test bench must not depend on

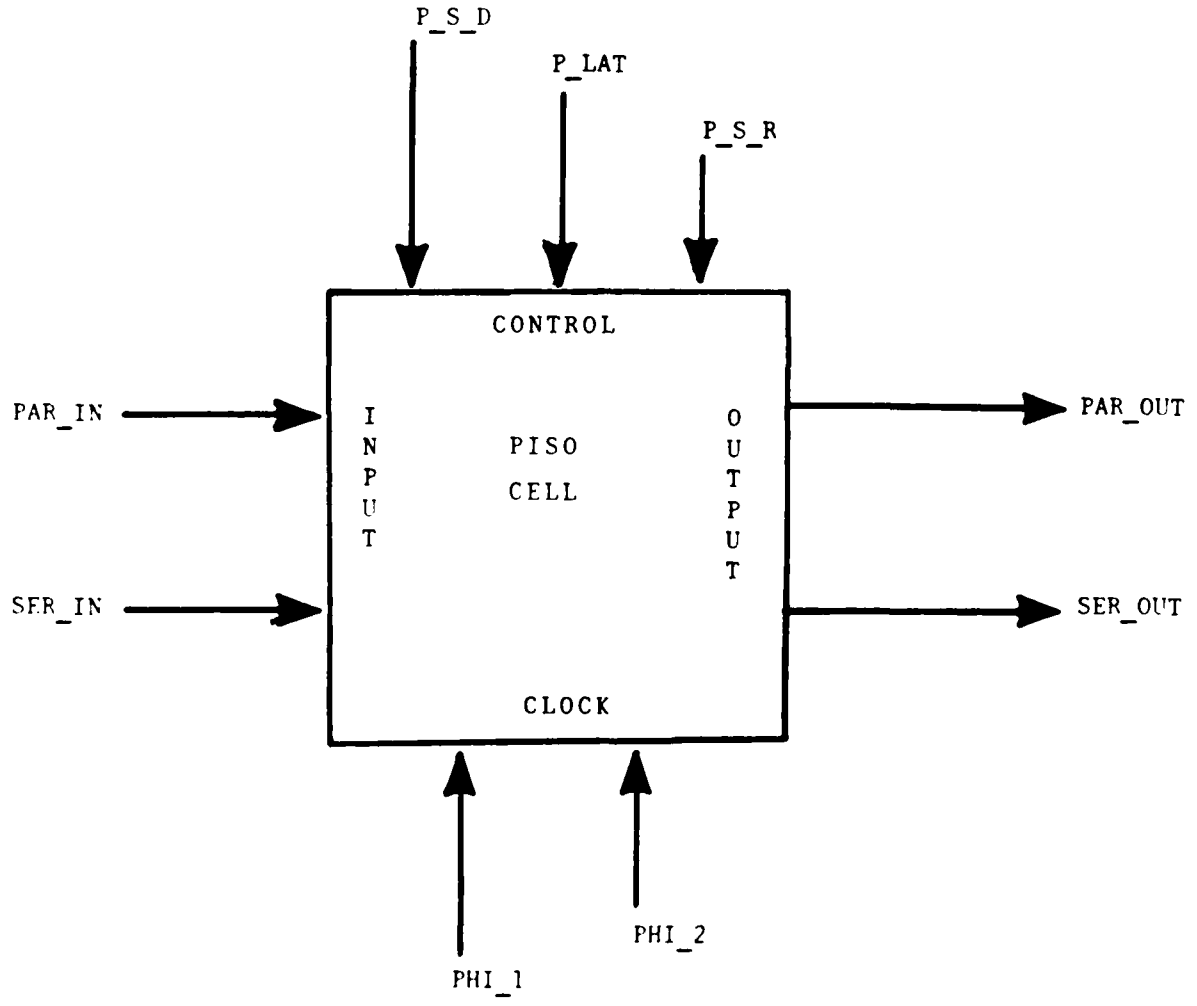


FIGURE 4-1. PISO Cell.

```
--**************************************************
--
--      DATE:  25 NOVEMBER 1986
--      VERSION:  1.0
--
--      TITLE:  Test Bench for Simple Piso Cell
--      FILENAME:  test_bench.vhd
--      COORDINATOR:  Capt. Charles H. Cooper
--      PROJECT:  THESIS
--      OPERATING SYSTEM:  VMS
--      LANGUAGE:  VHDL
--
--      ENTITY:

          with package <<VHDL>>SIMULATOR_STANDARD;
          use SIMULATOR_STANDARD;
          with package WFTA_DECLARATIONS;
          use WFTA_DECLARATIONS;


          entity TEST_BENCH is

          end TEST_BENCH;

--      FUNCTION:
--              This is the test bench for the simple
--      Piso cell.
--
--**************************************************
```

FIGURE 4-2. PISO Test Bench Interface Declaration.

any external influences and will not include any ports or generics in the interface declaration [6].

The control of the simulation conditions are accomplished through the virtual test equipment predefined by the package SIMULATOR_STANDARD in the <<VHDL>> library created during the installation of the simulator. The test bench accomplishes this through the instantiation of the virtual

```
architecture PISO of TEST_BENCH is

  PISO_BLK:

block

  signal stop, st_enable, overflow, bn_enable : BIT;
  signal quiescence, trans_overflow, delta_overflow : BIT;
  signal activity, cum_activity :
            SIMULATOR_STANDARD.TRANSACTION_NUMBER_TYPE;
  signal CLK_GO, SPAC_1, SPAC_2, PHI_2, PHI_1: BIT;
  signal DATA, P_S_D, P_S_R, P_LAT: BIT;
  signal P_I, S_I: Z_BIT;
  signal P_O, S_O: BIT;

  initialize BIT to '0';

  component SIMPLE_PISO_CELL port
                ( P_IN, S_IN: in Z_BIT;
                  P_SHIFT_DOWN,
                  P_SHIFT_RIGHT,
                  P_LATCH,
                  CLK2, CLK1:  in BIT;
                  P_OUT, S_OUT: buffer BIT);

  for all : SIMPLE_PISO_CELL use
          entity (SIMPLE_PISO_CELL)
          architecture (BEHAVIOR);
  end for;

begin

  L1 : signal_trace_data_recorder
       port(st_enable, overflow)
       generic(simulator_standard.trace_on);

  L2 : test_bench_control
       port (stop)
       generic (100, 1000, 1ns);

  L3 : bed_of_nails
       port (bn_enable, quiescence, trans_overflow,
             delta_overflow, activity, cum_activity)
       generic (10000, 100);
```

FIGURE 4-3. PISO Test Bench Body Declaration.

```
    st_enable <= '1';

    bn_enable <= '1';

    P_I <= convb_z(DATA and
                   (not (convz_b(S_I))));
    S_I <= P_I after 90 ns;

    DATA <= '1';

    P_S_D <= (DATA and (not P_S_R) and
              (not P_LAT));
    P_LAT <= P_S_D and (not P_S_R)
             after 30 ns;
    P_S_R <= P_LAT after 30 ns;

    CLK_GO <= '1';

    PHI_1 <= (CLK_GO and (not SPAC_1) and
              (not PHI_2) and (not SPAC_2))
              after 1 ns;
    SPAC_1 <= (PHI_1 and (not PHI_2) and
               (not SPAC_2))
               after 3 ns;
    PHI_2 <= (SPAC_1 and (not SPAC_2))
              after 3 ns;
    SPAC_2 <= PHI_2 after 3 ns;

    STOP <= '1' after 800 ns;

--PISO
      PISO1 : SIMPLE_PISO_CELL port
                (P_IN => P_I,
                 S_IN => S_I,
                 P_SHIFT_DOWN => P_S_D,
                 P_SHIFT_RIGHT => P_S_R,
                 P_LATCH => P_LAT,
                 CLK2 => PHI_2,
                 CLK1 => PHI_1,
                 P_OUT => P_O,
                 S_OUT => S_O);
   end block;
end PISO;
```

FIGURE 4-3 (CONT). PISO Test Bench Body Declaration.

test equipment components that include the SIGNAL_TRACE_DATA_RECORDER, TEST_BENCH_CONTROL, and BED_OF_NAILS. The VHDL test equipment components are described in more detail in the VHDL Build 2 Simulator User's Manual [6].

The input, control, and clock signals are defined in the test bench. The simulator restriction of no multiple inputs requires the use of the logical timing statements defined in Figure 4-3 to create the PHI_1 and PHI_2 phased clock signals. The timing diagram for the two phase clock signals is shown in Figure 4-4. The control signals are defined using a similar logical timing statement and controls the movement of data through the PISO. The data is first shifted down with the P_S_D (parallel shift down) control signal, then latched into the serial registers with the P_LAC (parallel latch) control signal, and finally the data is shifted right with the P_S_R (parallel shift right) control signal. The input data is also alternated between '1' and '0' values using logical timing statements. The time delay for the change of signals is defined using the reserved word after in each of the signal definition statements. The values for the variables are then assigned to the PISO cell in the component instantiations. Once the signal variables have been defined the test bench is analyzed with the VHDL language analyzer and the IVAN representation stored in the default design library (<<USER1>>). The commands used to analyze the VHDL source code modeling descriptions for the PISO cell simulation are as follows:

```
$ vhdl WFTAPAC.VHD
$ vhdl SIMPLE_PISO_CELL.VHD
$ vhdl PISO_TEST_BENCH.VHD
```

The errors identified in the analysis of the VHDL source code hardware descriptions are described in detail in the VHDL Analyzer User's Manual [5]. This completes the development of the VHDL test bench step of the simulation process.

The development of the dynamic MSFF test bench is similar to the development of the test bench for the PISO cell. The structural block diagram for the dynamic MSFF is shown in Figure 4-5 and is composed of two uni-directional transmission gates and two tri-state inverters. The test equipment variables are basically the same and only differ with

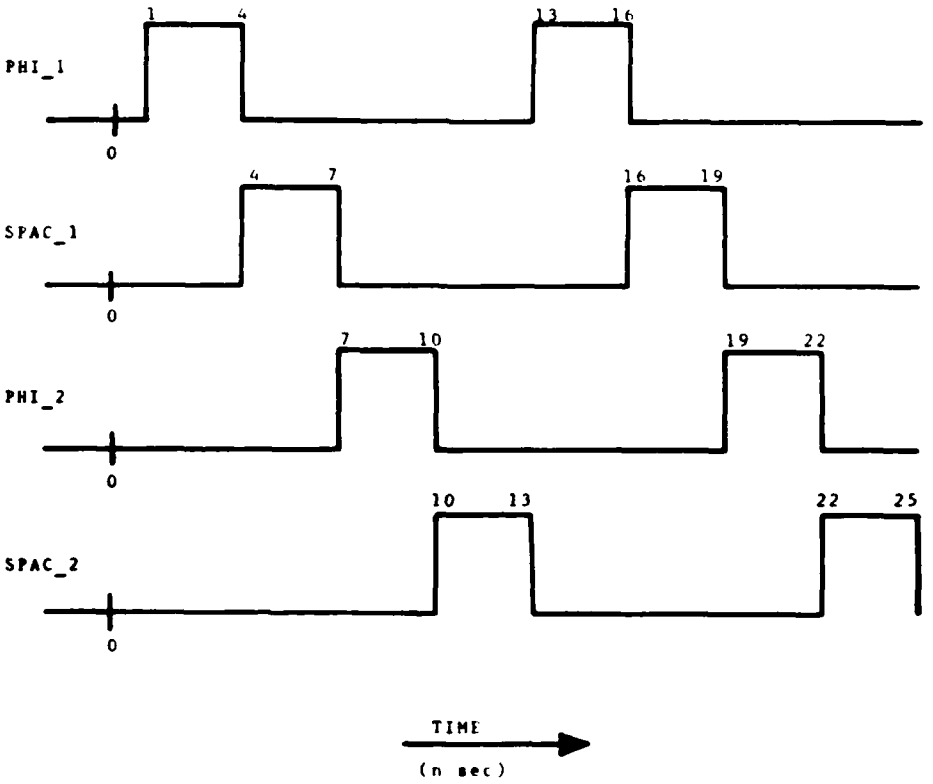


FIGURE 4-4. Timing Diagram for Two Phased Clock.

the simulation run times defined by the stop variable. The input, clock, and control signal assignments are done in a similar manner to the PISO cell; in fact, the two phased clock signals are identical. The VHDL source code modeling descriptions for all the MSFF components and the test bench are contained in Appendix D. Again, the test bench must be analyzed with the VHDL language analyzer and the IVAN representations inserted into the default design library with the following commands:

```
$ vhdl WFTAPAC.VHD
$ vhdl SIMPLE_T_GATE.VHD
$ vhdl Z_INVERT.VHD
$ vhdl MSFF_TEST_BENCH.VHD
```

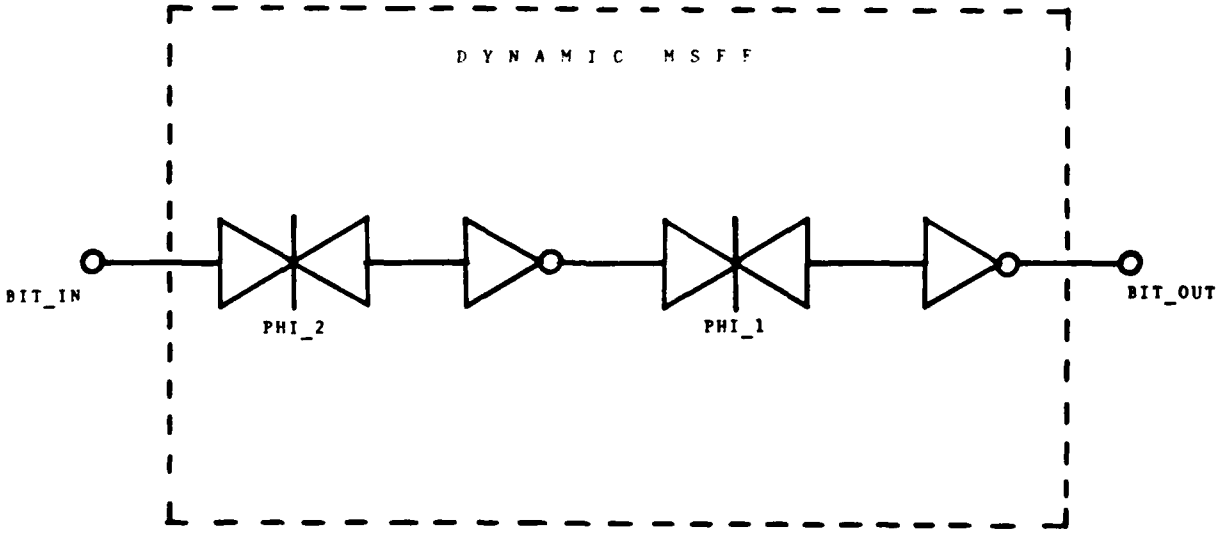


FIGURE 4-5. Dynamic MSFF.

4.3 Model Generation from the VHDL IVAN Representations

The VHDL model generator software extracts the hardware description modeling information from the design library (IVAN representations) and then constructs the Ada modules used in the generation of the simulator kernel. The model generator uses a two step process. First, the model generator transforms the IVAN representations from the design library into an internal Ada representation. The internal representation is then converted into the standard Ada text files. The commands used to execute the model generator for the PISO cell simulation are as follows:

```
$ mg package=WFTA_DECLARATIONS/srcf=WFTAPAC
$ mg architecture=BEHAVIOR[SIMPLE_PISO_CELL]/srcf=SIMPLE_PISO_CELL
$ mg architecture=PISO[TEST_BENCH]/TOP
```

The commands used to execute the model generator for the dynamic MSFF simulation are as follows:

```
$ mg package=WFTA_DECLARATIONS/srcf=WFTAPAC
$ mg architecture=BEHAVIOR[SIMPLE_T_GATE]/srcf=SIMPLE_T_GATE
$ mg architecture=BEHAVIOR[Z_INVERTER]/srcf=Z_INVERT
$ mg architecture=STRUCTURE[DYNAMIC_MSFF]/srcf=DYNAMIC_MSFF
$ mg architecture=D_MSFF[TEST_BENCH]/TOP
```

The detailed information on the execution of the model generator and the possible errors detected during the model generation step of the VHDL simulation process are contained in the VHDL Build 2 Simulator User's Manual [6].

4.4 Compiling and Linking of the VHDL Ada Models

The VAX Ada support software is executed to compile the standard Ada text files created by the model generator and then links the compiled Ada modules to form the SIM_KERNEL_MAIN program used to execute the actual simulation of the hardware component. However, to compile the Ada source modules requires the designation of one or more Ada program libraries for the storage of the compiled Ada objects. The libraries are created by the execution of the following commands for the PISO cell and dynamic MSFF simulations.

```
--PISO Cell
     $  acs create sublibrary [DATAMGR.VHDL2_SYSTEM.ADALIB_SIM_CORE.PISO]

--Dynamic MSFF
     $  acs create sublibrary [DATAMGR.VHDL2_SYSTEM.ADALIB_SIM_CORE.DMSFF]
```

Once the sublibraries have been created the Ada libraries must be set for the compilation and creation of the Ada object files. The library is set by the following commands for the PISO cell and dynamic MSFF simulations.

```
--PISO Cell
     $  acs set library [DATAMGR.VHDL2_SYSTEM.ADALIB_SIM_CORE.PISO]

--Dynamic MSFF
     $  acs set library [DATAMGR.VHDL2_SYSTEM.ADALIB_SIM_CORE.DMSFF]
```

The Ada object files are then created by the execution of the following commands:

--PISO Cell

```
$ ada/opt WFTAPAC.ADA
$ ada/opt SIMPLE_PISO_CELL
$ ada/opt PISO.ADA
```

--Dynamic MSFF

```
$ ada/opt WFTAPAC.ADA
$ ada/opt SIMPLE_T_GATE.ADA
$ ada/opt Z_INVERT.ADA
$ ada/opt DYNAMIC_MSFF.ADA
$ ada/opt D_MSFF.ADA
```

When the standard Ada text files have been compiled in the designated ACS sublibrary the models are linked with the test bench through the command:

--PISO Cell and Dynamic MSFF

```
$ acs link SIM_KERNEL_MAIN
```

The linking of the Ada models creates the SIM_KERNEL_MAIN.EXE file, this file is an executable Ada file that will run to perform the specified simulation experiment for a hardware component. Execution and error details on the compiling and linking step of the VHDL simulation process are contained in the VHDL Build 2 Simulator User's Manual [6].

4.5 Running the Simulator Kernel

The simulation of the hardware component is performed through the execution of the SIM_KERNEL_MAIN.EXE file. This process creates a Signal Map File (TEST_BENCH.SMF) and a Signal Trace File (TEST_BENCH.STF) that are used by the VHDL report generator. The Signal Map File is a text file that contains the documentation of the signal numbers used in the Signal Trace File, the Signal Map files for the PISO cell and dynamic MSFF are contained in Appendix D. The Signal Trace File contains a signal history record of the simulated hardware component. The hardware component simulation is run by executing the following VMS command:

```
--PISO Cell and Dynamic MSFF
    $  run SIM_KERNEL_MAIN
```

The runtime execution and error details for the VHDL simulator are also documented in the VHDL Build 2 Simulator User's Manual [6].

4.6 Execution of the Report Generator

The VHDL report generator documents the time history of the signals specified by the user. The report generator uses the Signal Map File and Signal Trace files as inputs for the documentation of the selected signal histories. The signals desired and the format for the report to be generated is specified in the INPUT.RCL file. This file must be created prior to the execution of the report generator and is created by using the host system editor. The INPUT.RCL file for both the PISO cell and dynamic MSFF simulation reports are documented on the first page of each report contained in Appendix D. The VMS command used to execute the

report generator for both simulations is as follows:

--PISO Cell and Dynamic MSFF

```
$  rg INPUT.RCL TEST_BENCH
```

The execution of this command creates a TEST_BENCH.RPT file that contains the user specified signal time history information. The specific details on the execution and error messages created by the VHDL report generator are specified in the VHDL Build 2 Simulator User's Manual [6]. The actual reports for the simulation of the PISO cell and dynamic MSFF are contained in Appendix D. The results in the signal history reports for the simple simulations of the PISO cell and dynamic MSFF show that both hardware elements performed as specified.

Chapter 5

Conclusions and Recommendations

5.1 Conclusions

This thesis addressed the problem of modeling and simulating the arithmetic processing hardware of the WFTA 16 PFA processor with the VHSIC Hardware Description Language (VHDL). The modeling of the WFTA 16 PFA processor hardware components was done at all the levels of component abstraction from the gate level to the register level. The VHDL language supported the description of hardware models at all these levels and therefore provides and excellent design tool for the modeling of VLSI hardware components. The VHDL support environment provides an excellent VLSI design tool for the successful modeling and simulation of the VHDL source code hardware modeling descriptions. The AFIT beta testing of the UTMC developmental releases of the VHDL language support environment were successful and provided much needed interface information on the actual development, modeling, execution, and simulation with the VHDL support environment. However, the Build 2 restrictions on the VHDL simulator did require limited modeling and simulation application of the VHDL language used in the source code descriptions. Therefore, the simulations run in this thesis are not optimal VHDL hardware simulations due to the fact that Build 2 release of the VHDL simulator software used was not a final version of the VHDL simulator, but was only a limited developmental beta release of the VHDL simulator modules completed to date.

5.2 Recommendations

The application of the VHDL language and VHDL support environment to the modeling and simulation of the 4080-point WFTA PFA pipeline signal processor should be continued with 1987 AFIT graduate students. The VHDL simulations completed in this thesis were done on very simple hardware elements of the WFTA 16 PFA processor, and were by no means a complete and comprehensive simulation of the processor. The simulations completed were limited by the current developmental restrictions of the Build 2 Simulator and were limited by disk memory on the AFIT CSC VAX/VMS system. Therefore, as additional releases of the VHDL simulator are received for beta testing the simulations on WFTA PFA processor elements should become more complete and should eventually provide a system simulation of the entire PFA pipeline processor.

The current disk memory space on the AFIT CSC VAX/VMS should be expanded to accommodate the VHDL support environment. In addition, the VHDL Data Manager should set up the required number of user accounts needed to support the use of the VHDL support environment by students and faculty. Overall the VHDL language and support environment provide an excellent design tool for the implementation, modeling, and simulation of VLSI class hardware components.

Bibliography

1. Air Force Institute of Technology. N.2 Manual. March 1985. Department of Electrical and Computer Engineering, School of Engineering, AFIT (AU), Wright-Patterson AFB OH.

2. Carter, Dr (Lt Col) Harold W., Associate Professor, Department of Electrical Engineering. Personal Interviews. Air Force Institute of Technology, Wright-Patterson AFB OH, 10 January through 7 February 1986.

3. Collins, Capt James M. Modeling and Simulation of a Signal Processor Implementing the Winograd Fourier Transform. MS thesis, AFIT/GE/ENG/85D-9. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1985.

4. Coutee, Paul W. Arithmetic Circuitry for High Speed VLSI Winograd Fourier Transform Processors. MS Thesis, GE/ENG/85D-11. School of Engineering, Air Force Institute of Technology (AU) Wright-Patterson AFB OH, December 1985.

5. Intermetrics Corporation. VHDL Analyzer User's Manual. 30 April 1986, ASD (AFSC), Wright-Patterson AFB OH. Contract F33615-83-C-1003.

6. Intermetrics Corporation. VHDL Build 2 Simulator User's Manual. 31 MAY 1986, ASD (AFSC), Wright-Patterson AFB OH. Contract F33615-83-C-1003.

7. Intermetrics Corporation. VHDL Design Analysis. 16 February 1984, ASD (AFSC), Wright-Patterson AFB OH. Contract F33615-83-C-1003.

8. Intermetrics Corporation. VHDL Design Analysis and Justification. Version 5.0. 30 July 1984, ASD (AFSC), Wright-Patterson AFB OH. Contract F33615-83-C-1003.

9. Intermetrics Corporation. VHDL Design Library User's Manual/ Implementor's Guide, Volume I -- Body. 15 March 1986, ASD (AFSC), Wright-Patterson AFB OH. Contract F33615-83-C-1003.

10. Intermetrics Corporation. VHDL Language Reference Manual, Version 7.2. 1 August 1985, ASD (AFSC), Wright-Patterson AFB OH. Contract F33615-83-C-1003.

11. Intermetrics Corporation. VHDL User's Manual, Volume I -- VHDL Tutorial. 1 August 1985, ASD (AFSC), Wright-Patterson AFB OH. Contract F33615-83-C-1003.

12. Intermetrics Corporation. VHDL User's Manual, Volume II -- VHDL User's Reference Guide. 1 August 1985, ASD (AFSC), Wright-Patterson AFB OH. Contract F33615-83-C-1003.

13. International Business Machines Corporation. Preliminary Language Requirements for VHDL. Version 2.0. 1 February 1984, ASD (AFSC), Wright-Patterson AFB OH. Contract F33615-83-C-1003.

14. Linderman, Dr (Capt) Richard W., Assistant Professor, Department of Electrical Engineering. Personal Interviews. Air Force Institute of Technology, Wright-Patterson AFB OH, 10 January through 7 February 1986.

15. Lipsett, Roger et al. "VHDL - The Language." Rough Draft of Report to be published in IEEEDT. Intermetrics Corporation, Bethesda MD, 7 November 1985.

16. Rossbach, Paul C. Control Circuitry for High Speed VLSI Winograd Fourier Transform Processors. MS Thesis, GE/ENG/85D-35. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1985.

17. Shahdad, Moe et al. "VHSIC Hardware Description Language." Computer,18: 94-102 (February 1985).

18. Taylor, Kent. Architecture and Numerical Accuracy of High Speed DFT Processors. MS Thesis, GE/ENG/85D-47. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1985.

Vita

Captain Charles H. Cooper was born on 18 December, 1956 at March Air Force Base, Riverside, California. He attended Kaiserslautern American High School in Kaiserslautern, West Germany, and was appointed to the United States Air Force Academy in June 1975. He graduated from the Air Force Academy with a Bachelor of Science Degree in Electrical Engineering in May 1979 and was commissioned as a regular officer in the United States Air Force. He was assigned to Space Division, Deputy for Technology, Los Angeles Air Force Station, California where he served as the mission planning project officer for the Teal Ruby Experiment. In November 1981, he was assigned to the Eastern Space and Missile Center, 6555th Aerospace Test Group, Satellite Integration Division, Cape Canaveral Air Force Station/Kennedy Space Center, Florida, where he served as the Air Force Launch Controller for orbiter/payload integration operations on the January 1985 launch of STS Mission 51-C. In May 1985 he entered the Air Force Institute of Technology to pursue a Masters degree in Electrical Engineering. He is married to the former Jennifer Leigh Barron of Colorado Springs, Colorado and has two daughters Stephanie Leigh and Jessica Anne.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/ENG/86D-44	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION School of Engineering 6b OFFICE SYMBOL (If applicable) AFIT/ENG	7a NAME OF MONITORING ORGANIZATION
6c ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, OH 45433-6583	7b ADDRESS (City, State, and ZIP Code)
8a NAME OF FUNDING / SPONSORING ORGANIZATION AFWAL 8b OFFICE SYMBOL (If applicable) AADE	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB, OH 45433-6583	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
Modeling and Simulation of the WFTA 16 PFA Processor using the VHSIC Hardware Description Language, Volume I.

12 PERSONAL AUTHOR(S)
Charles H. Cooper, Capt., USAF

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
MS Thesis	FROM _____ TO _____	1986 December	86

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
09	02		VHSIC Hardware Description Language Digital Simulation
			Computer Architecture Signal Processing

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The VHSIC Hardware Description Language (VHDL) is applied to the problem of modeling and simulating VLSI CMOS components of the WFTA 16 PFA processor. The 16-point PFA processor is one of three PFA processors under design and development for the implementation of the 4080-point PFA pipeline processor by the VLSI design group at the Air Force Institute of Technology. The PFA processor is modeled by applying the hierarchical facilities of the VHDL language to form the top level register component descriptions from combinations of the primary buildig block hardware element descriptions. Two simple VHDL simulations are performed using the beta test versions of the VHDL simulator and support environment. The simple component simulations are performed on a VHDL behavioral description of the Parallel-In, Serial-Out (PISO) cell and a VHDL structural description of a dynamic Master-Slave Flip-Flop (MSFF).

Approved for public release: IAW AFR 190-1.
[illegible] 5 March 87
[illegible] Development
[illegible]

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Richard W. Linderman	22b TELEPHONE (Include Area Code) 513-255-6913 / 22c OFFICE SYMBOL AFIT/ENG

DD Form 1473, JUN 86 Previous editions are obsolete

END

5-87

DTIC