[13] C. J. Tseng and D. P. Siewiorek, "Automated Synthesis of Data Paths in Digital Systems", IEEE Trans. CAD, vol. CAD-5, no.3, March 1986.

[14] F. Prosser and D. Winkel, "The Art of Digital Design", Prentice-Hall, 1987.

[15] D. Siewiorek, C. Bell and A. Newell, "Computer Structures: Principles and Examples", McGraw-Hill, 1971.

[16] L. J. Hafer and A. C. Parker, "Automated Synthesis of Digital Hardware", IEEE Tran. Computers, Feb 1982.

[17] Jayanta Roy and Ranga Vemuri, "Appropriate Usage of VHDL : The Synthesis Point of View", 53pp, TM-DDE-89-08, December 1989 (Revised, April 1990, May 1990).

[18] Jay Roy, "Guide to the VSS System and Its Input VHDL Subset", TM-ECE-DDE-91-21, June 1991.

[19] Ranga Vemuri, "Distributed Synthesis Systems Research : Goals, Methods and Status", TM-ECE-DDE-91-18, June 1991.

[20] Jay Roy and Ranga Vemuri, "DSS : A Distributed Synthesis System for VHDL Specifications" VHDL Conf., April 1991.

[21] Jay Roy, Nand Kumar, R. Dutta, P. Mamtora, Ranga Vemuri, "VSS : Its Organization and Structure", TM-ECE-DDE-91-15, June 1991.

References

[1] M. R. Barbacci and D. P. Siewiorek, "The Design and Analysis of Instruction Set Processors", McGraw-Hill, 1982.

[2] Jayanta Roy and Ranga Vemuri, "Appropriate Usage of VHDL : The Synthesis Point of View", 53pp, TM-DDE-89-08, December 1989 (Revised, April 1990, May 1990).

[3] Kapila Udawatta and Ranga Vemuri, "A VHDL Subset for Synthesis : Reference Manual", 34pp, TM-DDE-89-09, December 1989.

[4] P. J. Drongowski, "An Organization-Level Story Board for Agent - A VLSI Designer's Assistant", Internal Report, DSRG, CES Dept, Case Western Reserve University, Jan 1987.

[5] W. J. Cullyer, "Implementing Safety Critical Systems: The VIPER Microprocessor", pp. 1-26, in [6].

[6] G. Birtwistle and P. A. Subrahmanyam (eds.), "VLSI Specification, Verification, and Synthesis", Kluwer Academic Publishers, 1988.

[7] W. A. Hunt Jr., "FM8501: A Verified Microprocessor", Tech Rep 47, Institute for Computing Science, Univ of Texas at Austin, Feb 1986.

[8] S. Johnson, R. Wihrmeister, and B. Bose, "On the Interplay of Syntheis and Verification", pp. 385-404, in [9].

[9] L. Claesen (ed.), **Applied Formal Methods for Correct VLSI Design**, Proceedings of the IMEC-IFIP International Workshop, November 1989.

[10] J. R. Southard, "MacPitts: An Approach to Silicon Compilation", IEEE Computer, pp. 74-82, 1983.

[11] P. G. Paulin, J. P. Knight, and E. F. Girczyc, "HAL : A multi-paradigm approach to automatic data path synthesis", 23rd DA Conf, pp. 263-270, July 1986.

[12] D. Brewer and D. Gajski, "Chippe: A System for Constraint-Driven Behavioral Synthesis", in IEEE Trans. CAD, vol. 9, no. 7, July 1990.

7 Status and Availability of the Benchmarks

As mentioned before, all of these benchmarks have been processed by the front-end of our synthesis system DSS. All are analyzed and most are also simulated using the Intermetrics VHDL simulator. DSS has been used to produce fabricatable designs for the Viper, Move Machine, Traffic Light Controller, Intel 8251, TMS-1000 and other smaller examples. The Viper description, the largest of these three, is included in Appendix A as an example to illustrate the constructs in the synthesizable subset we used in our benchmark suite. The traffic light controller chip has actually been fabricated recently. In the next several months, we expect to use some of the larger examples to capacity-test the DSS.

These benchmarks are available to any one interested on an 'as is', 'no commitments or guarantees attached' basis. One can ftp them or receive them free-of-charge by electronic mail or on a floppy disk for the cost of the disk and postage. [3]

8 A Note to the Serious User

If you plan to use these benchmarks seriously with your synthesis system, then you most certainly will have to change them to suite your subset and the idiosyncrasies of your synthesis system. The same applies if you wish to simulate them. View these VHDL programs as something you can start with, modify and prepare a test case processable by your system rather than starting from scratch. These may not be the best possible benchmark suite for high-level synthesis but as a consequence of your using these if you could say something about the characteristics a good high-level synthesis benchmark suite must possess, we will be interested to listen; so will be the high-level synthesis community.

Please do let us know, if you have successfully simulated or synthesized from any of these benchmarks. If you make useful or interesting changes to these benchmarks do let us know as well. We do wish to upkeep the benchmark suite and make them available to the public and where appropriate incorporate your changes in the process if its okay with you.

[3] Ranga Vemuri can be reached at 513-556-4784 and ranga@uceng.uc.edu; J. Roy can be reached at 513-556-3025 and jay@babbage.ece.uc.edu.

5.3 Error Correction System

An error correction consisting of an encoder, a noise generator and a decoder is available in the HLSW Clearinghouse in the form a HardwareC program. Paul Joslin translated this for our benchmark suite into 3 separate VHDL programs (the encoder, the decoder and the noise generator). Together, these programs consists of about 300 lines.

5.4 Elliptic Wave Filter

This is another widely known example in the high-level synthesis community and has been used as a benchmark at high level synthesis workshops. Rajiv Dutta wrote the VHDL version of this benchmark. It consists of about 60 lines of VHDL code. It has been simulated and synthesized.

6 Group M: Micellaneous

This group has following 2 benchmarks. All can be used as small examples to test synthesis systems while under development.

6.1 Blackjack Machine

Winkle and Processor described a blackjack machine in [14]. This machine is later synthesized by S. Johnson's system for formal design derivation. For our benchmark set, Mich Horton wrote a VHDL specification of this blackjack machine. The code is less than 90 lines long.

6.2 Elevator Controller

This is a translation into VHDL by Jay Roy of the elevator controller described by Hafer and Parker [16]. This VHDL description is about 60 lines long and uses a two process architecture. This description has been simulated.

4.10 Motorola 68000

This is the well-known 16-bit microprocessor from Motorola. An instruction set description (without exception handling) of this processor was written in ISPS by Michaael Norton at CMU in 1982 and was subsequently made available through the High-Level Synthesis Workshop Clearinghouse. A team consisting of Ricci Heishman, David Hollinden, John Krautheim, Tim McBrayer and Praveen Sinha translated most of this description into VHDL. Currently, this VHDL benchmark is incomplete and has no-ops in place of several instructions. This VHDL code is currently about 1270 lines long; it contains one process and a large number of procedures.

5 Group D: Arithmetic-Dominated Machines

This group has the 4 benchmarks described below.

5.1 Tseng's Example

Probably the most quoted and used 'benchmark' in high-level synthesis research, Tseng's example [13] is used to test mostly the scheduling phase of the synthesis systems. It is rather small consisting of straight-line code. A HardwareC version of this example is available in the HLSW Clearinghouse. It is has been converted into a 100-line VHDL program by S. Panchumarti for our benchmark suite. It has one process statement with virtually no data-dependent control constructs. This description has been simulated and synthesized.

5.2 Differential Equation Solver

This rather simple-minded differential equation solver was used by Brewer and Gajski [12] as well as Paulin et. al. [11] to illustrate their respective synthesis algorithms. A HardwareC version was prepared by Rajesh Gupta. It is translated into VHDL by S. Panchumarti for our benchmark set. This code about 70 lines long and has just one process. It has mostly straight-line code with very few control constructs. This description has been simulated.

4.6 Intel 8251

The Intel 8251 is a serial communication interface adapter which acts as an interface between an 8080-type microprocessor bus, a serial transmitter and a serial receiver. John Nestor's ISPS description of the I8251 is in the HLSW clearinghouse. It is rewritten in VHDL by S. Datta and Guruprasad for our benchmark suite. This description is about 770 lines long and consists of many processes communicating through signals. This description has been synthesized.

4.7 TMS-1000

The TMS-1000 is a single chip microcomputer from Texas Instruments. It is described in [15] in detail using an ISPS description. Raghu Vutukuru and Srini Rangarajan used this as the basis to write a VHDL specification of the TMS-1000. This VHDL code is about 870 lines long and consists of many procedures inside a single VHDL process. This description has been synthesized.

4.8 ISPS Version of the MOS Technology MCS 6502

The 6502 processor was described in ISPS by G. W. Leive in 1978. This description is available through the HLSW Clearinghouse. Based on this, a VHDL description of the 6502 architecture is written by Ayman Abdo and Nael Abu-Ghazaleh. This VHDL description is about 800 lines long and consists of a large number of procedures called from a single process.

4.9 Intel 8080

An ISPS specification of the Intel 8080 processor appeared in [15]. Based upon this description, Sandeep Aji wrote a VHDL description of the processor for our benchmark set. This benchmark implements only 56 instructions out of the 128 instructions of the 8080 processor. This benchmark is about 1000 lines long and consists of one process and a large number of procedures.

4.3 Frisc Processor

Frisc microprocessor is used as an example by J. Southard to describe the operation of MacPitts, one of the early silicon compilers [10]. A HardwareC description of this simple microprocessor was written by Rajesh Gupta and is provided in the HLSW Clearinghouse. Using this as the basis, Paul Joslin prepared a VHDL specification for our benchmark suite. This code about 300 lines long; it uses few procedures inside a single process statement.

4.4 FM-8501 Processor

The 16-bit FM-8501 processor was described by W. Hunt who designed and used it for his formal verification research using the Boyer-Moore theorem prover. Since then, the FM-8501 has been implemented in VLSI in slightly different forms. It has been used by S. Johnson et. al. to investigate the interplay between synthesis and verification in the design process. Our FM-8501 benchmark is based on the textual description given by W. Hunt in [7]. This benchmark, written by Nand Kumar and Ranjan Prasad, is about 320 lines long and uses several processes communicating through signals. It has no procedures.

4.5 Viper Microprocessor

The Viper microprocessor is also well-known in the formal verification community. It was the first relatively large-scale processor to be specified, designed and implemented with formal verification in mind. Several companies have implemented the Viper microprocessor. Its design has been partially verified by A. Cohn. W. J. Cullyer [5] provided a detailed textual description of the Viper. Using this description, P. Subba Rao and S. Bangolore have written a VHDL specification of the Viper for our synthesis benchmark suite. This Viper description is about 480 lines long and consists of a large number of procedures used within a single VHDL process. This description has been synthesized.

4 Group P: Control-Dominated Processors

This group consists of 10 processors. Except one, all are 'real'; Some are 'very real' and are/were commercially available.

4.1 Mark-1

Mark-1 is an early digital computer built in late 40's and early 50's at the Manchester University. An ISPS (Instruction Set Processor Specification) of Mark-1 is given in [1]. This description is translated into ISPS by Jay Roy and is incorporated in our benchmark set. Mark-1 has a 32-bit, 8K memory. It has a 13-bit control register and a 32-bit accumulator. The instructions consist of a 13-bit address field and a 3-bit opcode field. Mark-1 has a repertoire of eight instructions. The Mark-1 specification has one VHDL process, several variables and no internal signals. The VHDL code for Mark-1 is about 40 lines long. This description is simulated.

4.2 Move Machine

The Move Machine concept was first suggested by Ivan Sutherland based on the observation that conventional processing units spend much of their time moving arguments from memory to the CPU and moving results from the CPU to the memory. As the name implies, the instruction set of the Move Machine merely controls instruction and data flow; it does not actually compute (manipulate) any data values. Instead, certain memory locations are (assumed to be) connected to external computational units which perform the actual computations. Paul Drongowski provided the ISPS description of a move machine in [4]. Jay Roy translated this description into VHDL for our benchmark set. This benchmark consists of about 80 lines of VHDL code. The description consists of three VHDL processes (fetch, decode and execute) and several internal variables, signals and 'wait' statements. This description has been simulated and synthesized.

3 Grouping of the Benchmarks

For convenience of presentation, we have divided the benchmarks into three groups:

1. **Group P: Control-Dominated Processors**

   These are microprocessors such as M 68000, TMS 1000, FM 8501 etc. and are typically control-dominated.

2. **Group D: Arithmetic-Dominated Machines**

   These circuits are typical of data-dominated machines. Many of these have applications in digital signal processing. Examples include Kalman Filter, Differential equation solver, Fifth-order elliptic wave filter etc.

3. **Group M: Micellaneous**

   These are small specifications which happened to be around. We used some of these for early testing and debugging of the our synthesis software.

In the subsequent sections, we describe each of these benchmarks. Within each group the benchmarks are arranged in the increasing order of their raw size. For each benchmark we include its origin where known and the specific authors of these VHDL implementations. All the benchmarks are analyzed using the Intermetrics VHDL Development System (version 3.0) and all the benchmarks are parsed and translated into the intermediate format by the front-end of our DSS synthesis system. In the following text, when we say that a benchmark is 'simulated' we mean that the VHDL description is simulated using the Intermetrics VHDL simulator (version 3.0). While we haven't attempted exhaustive simulation in any case, we are reasonably sure that the description is working as it is supposed to be. Other benchmarks (which we don't claim to have simulated) have actually been partially simulated. But our confidence level is not so relatively high that we don't claim that we have done any simulation. When we say a benchmark has been 'synthesized' we mean that our high-level synthesis system (DSS) can produce a register-level design (also in VHDL for that specification. In somes cases that register-level VHDL output has actually been compiled into CMOS layouts. We are in the process of gathering quality measures of the designs produced by our synthesis system for some of these benchmarks.

We use the word 'benchmark' to merely mean 'a test case' for the high-level synthesis software. Specifically, we *do not* intend to imply any qualitative attributes of these benchmarks nor of the synthesis systems that may or may not process these benchmarks. Development of such measures of quality either of the synthesis systems or of their benchmark tests is, in our opinion, still an open question which we hope will receive some attention soon, perhaps spurred by the 'dubious benchmarks' such as the ones described in this document.

2 What's Special About Our Benchmarks

It is important to understand the biases of the benchmarks described in this document. The prominent biases are,

1. These benchmarks are written using a specific VHDL subset described in [17, 3]. We believe that programs written in this subset are synthesizable. Our synthesis system, DSS, is being developed to synthesize hardware designs implementing the specifications written in this subset. We also believe that this subset syntactically subsumes many other VHDL subsets for synthesis proposed by other researchers. Therefore, we believe that these benchmarks may be of general interest outside our research group.

2. Many of these benchmarks, particularly the larger ones, specify existing real-world processors. Others, particularly the smaller ones, specify pedagogic test cases. We have chosen many larger examples, because we believe that most existing synthesis systems have considerable trouble in dealing with large examples, specifically while attempting to optimize large-scale designs. With one or two exceptions, all the benchmarks describe control-dominated machines. We believe that the optimization of large, control-dominated designs is an open problem.

3. Each benchmark has its origin, in some form or other, outside our research group. Some benchmarks existed in another language; we merely translated them into VHDL. Other benchmarks describe processors which due to various reasons became popular in the synthesis and/or verification community.

1 Introduction

The need for a common set of benchmarks to drive the high-level synthesis research has been emphasized on many occasions, most notably at the annual High Level Synthesis Workshops. In response to this need, a benchmark clearing house has been set up at the University of Washington and has been organized by Bob Mayo and others. [1] This clearing house currently has several benchmarks (behavioral specifications of various circuits) written in ISPS, HardwareC and some in VHDL. The multilingual benchmark repository is a consequence of the fact that until a few years ago, high-level synthesis systems have used various hardware specification languages, including ISPS, MIMOLA, ADA, Pascal, FORTRAN and even C, depending upon the history and taste of the particular research teams. More recently, driven by the growing acceptance of VHDL, many researchers have started providing VHDL front-ends to their synthesis systems (for example, CMU's System Architects Workbench) and several others have developed synthesis systems specifically designed to be driven by specifications written in VHDL (for example, Honeywell's VSYNTH system, UC Irvine's VSS and our own DSS). In spite of this growing acceptance of VHDL by the high-level synthesis research community, to our knowledge there is no collection of benchmarks in VHDL suitable for high-level synthesis.

At the University of Cincinnati, we have been developing a distributed high-level synthesis system called DSS [19, 20, 21, 18]. DSS executes on a multiple processor machine and is capable of producing CMOS layouts from behavioral VHDL specifications. The syntax and synthesis-semantics of the VHDL subset used to drive DSS are discussed in detail in [17, 3]. Over time, to test the DSS software, we have developed a collection of benchmarks written in that synthesizable VHDL subset. This document is brief description of these benchmarks. [2]

[1] Currently, this clearing house is located at the University of Irvine and is managed by Nikhil Dutt.

[2] The authors of this document have not actually written all the benchmarks. Most of them were written by students in a course on Computer Hardware Description Languages taught by Ranga Vemuri at UC during the Spring of 1991. P. Mamtora collected the ideas for these benchmarks from various sources. Jay Roy and Ranga Vemuri defined the VHDL subset used in these benchmarks. Jay Roy wrote a parser that parses this subset and produces a synthesis-oriented intermediate format file. Proper acknowledgements to the actual authors of the benchmarks appear in the subsequent sections.

8 A Note to the Serious User **10**

Contents

1 Introduction **2**

2 What's Special About Our Benchmarks **3**

3 Grouping of the Benchmarks **4**

4 Group P: Control-Dominated Processors **5**

4.1 Mark-1 5

4.2 Move Machine 5

4.3 Frisc Processor 6

4.4 FM-8501 Processor 6

4.5 Viper Microprocessor 6

4.6 Intel 8251 7

4.7 TMS-1000 7

4.8 ISPS Version of the MOS Technology MCS 6502 7

4.9 Intel 8080 7

4.10 Motorola 68000 8

5 Group D: Arithmetic-Dominated Machines **8**

5.1 Tseng's Example 8

5.2 Differential Equation Solver 8

5.3 Error Correction System 9

5.4 Elliptic Wave Filter 9

6 Group M: Micellaneous **9**

6.1 Blackjack Machine 9

6.2 Elevator Controller 9

7 Status and Availability of the Benchmarks **10**

Benchmarks for High Level Synthesis

Abstract

We introduce a collection of 'benchmarks' for high-level synthesis systems. These benchmarks are VHDL programs with mostly behavioral constructs. While these programs are written for a particular high-level synthesis system, called the Distributed Synthesis System (DSS) we have been developing, we believe that, with minor modifications, they are suitable for use by the high level synthesis community. In particular, the architects of high-level synthesis systems must aim to produce 'efficient' structures given such behavioral specifications as exemplified by these benchmarks. The collection benchmarks presented here includes the Viper microprocessor, the FM8501 microprocessor, the Intel 8251 and several others.

Benchmarks for High Level Synthesis *

Ranga Vemuri, Jay Roy, Paddy Mamtora and Nand Kumar
Laboratory for Digital Design Environments
Electrical and Computer Engineering Department
University of Cincinnati, Cincinnati, Ohio 45221-0030

June 1991
Revised November 1991

Technical Memo—ECE-DDE-91-11

LIMITED DISTRIBUTION NOTICE

This report has been, or will be, submitted for publication outside of UC, and will probably be copyrighted if accepted for publication. It has been issued as a Technical Memo for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of UC prior to publication is limited to peer communications and specific requests. Prepublication courtesies are requested.

*This work is partly supported by the Defense Advanced Research Projects Agency under order no. 7056 monitored by the Federal Bureau of Investigation under contract no. J-FBI-89-094 and by the University of Cincinnati Research Council.