1995 High Level Synthesis Design Repository

Preeti R. Panda and Nikil D. Dutt
Department of Information and Computer Science
University of California, Irvine, CA 92717-3425, USA

Abstract

In this paper we briefly describe a set of designs that can serve as examples for High Level Synthesis (HLS) systems. The designs vary in complexity from simple behavioral finite state machines to more complex designs such as microprocessors and floating point units. Most of the designs are described in the VHDL language at the behavioral level. We divide the designs into two categories. The first category contains designs that have documentation on the specifications of the designs along with the strategy used to test the individual design models. The second category contains examples used in many HLS papers, but lack comprehensive documentation and/or test vectors.

1 Introduction

The effort at creating a repository of High Level Synthesis (HLS) benchmark designs has been under way since the 24th Design Automation Conference in 1987. An informal set of HLS benchmarks was created after the High Level Synthesis Workshop in 1988 and maintained at the SIGDA repository at *mcnc.mcnc.org*. The purpose of maintaining this repository was to serve as a basis of comparison of various approaches to High Level Synthesis and to provide a means for researchers and developers to exercise their synthesis systems on a wide range of digital circuits.

This informal set was consolidated [DuRa92] into nine benchmark designs whose functionality and verification schemes were well documented. These benchmarks, which include some simple controllers, digital filters, a microprocessor slice, and a USART design among others, have been widely used as examples by HLS researchers. In [DuRa92], guidelines for developing and submitting new benchmarks were also formulated so as to make the benchmark collection an ongoing process. With this paper we release another set of designs that were submitted in accordance with these guidelines, as well as a set of designs that have complete HDL descriptions but lack comprehensive documentation and/or test vectors.

We present a new set of design examples that augment the existing benchmark suite. Most of these are fairly large designs with reasonably complex data and control structures. In most cases, the models are accompanied by test patterns that were used to verify the correctness of the descriptions. The tests are, of course, not exhaustive but are intended to check for typical behaviors. In some cases, boundary conditions have been extensively checked, while in others (like the microprocessor examples) a few cases have been tested for every instruction. We must mention that while the models have all been subjected to simulation checks, they may not all be directly synthesizable, and might require modification when used as an example for a synthesis tool. Some of the constructs used might make sense only for simulation. In such cases, however, the test vector set should be useful in determining the correctness of any modifications to the design descriptions.

In Section 2, we outline the guidelines for submission of new designs for incorporation into the repository. In Section 3, we present an overview of the designs in the repository. We conclude with a summary, where we also indicate the publicly accessible location of the repository.

2 Guidelines for Submission of Designs

In this section, we mention the guidelines for the submission of new designs to be incorporated into the repository, in an effort towards introducing more rigor into the benchmarking process, and towards the creation of a robust set of design examples for testing High-Level Synthesis tools and systems.

2.1 "Well-Known" HDL Description

The design must be described using a "well-known" HDL which has a publicly available Language Reference Manual (LRM), and which has a publicly available simulator. Sample HDLs that fit this criterion include VHDL and Verilog. The HDL description must be liberally commented to allow readability.

2.2 Documentation, Assumptions and Simplifications

The source of the design information should be specified (e.g., data sheet, initial design spec., etc.). A description of the design's functionality (using English, flowcharts, block diagrams, etc.) must accompany the HDL description. All assumptions and simplifications made in writing the HDL model must be clearly stated.

2.3 Simulation Vectors

A set of input and expected output functional test vectors must accompany the HDL description for simulating typical operational behaviors of the design. These test vectors need not be designed to exhaustively test the design. Instead, they can give some level of confidence in the behavioral HDL model, and allow translation and validation of the model into another HDL or description style.

The test vectors must also be accompanied by a (English) description of what functionality is being tested. The input and expected output vectors should be described in a generic format that allows ease of use in different simulation environments. A brief description of the test vector format must accompany the test vector set.

2.4 Simulator Details

Each benchmark design must indicate the name, version, and availability (where appropriate) of the simulator used to test the design.

2.5 Synthesis Outputs

The outputs of synthesis tools must be simulated using the same simulator and test vectors used to check the behavior of the input description.

3 Overview of the Designs

This section presents a brief overview of the designs. Figures 1 and 2 summarize some of the important aspects related to the functionality and verification of these designs, such as typical control features present, style of description, major data types used and the extent to which the design example has been tested. The number of lines of code (LOC) is mentioned to give a rough idea of the design's size. The LOC includes lines with comments. (The lines of *executable* VHDL code is typically 50 % of the total lines of code. *The LOC figures must be used with caution, for writing styles vary and the sizes of the models are small enough to permit erroneous estimations about design size based on LOC alone.*)

The first column gives the name of the design and the second mentions the type of design. In the *Design Level* column, *Algorithmic Behavior* refers to the types of designs which are described in an algorithmic fashion in a HDL.

The examples have been classified into two groups: those for which a substantial amount of documentation is available, and those for which the documentation is insufficient. Some of the models are quite thoroughly tested for errors while others have not been exercised very much.

Three examples in the repository are arithmetic circuits (*FP_Adder*, *FP_Mult* and *FP_Divider*), all based on the IEEE Floating Point Standard. In the case of *FP_Adder* and *FP_Mult*, the operands and result are represented by a sign bit, a 127-biased integer exponent in the range 0..255, and a 23-bit vector mantissa with a *hidden 1* [HePa90]. In *FP_Divider*, the Digit-Recurrence Algorithm [LaEr94] is employed for the division using radix 512, the operands and the result being represented by a sign bit, a 10-bit 511-biased integer exponent in the range 0..1023, and a 53-bit vector mantissa with a hidden 1. All three models are described in VHDL language.

Two examples (*Prawn* and *RT-PC*) are models of CPUs, both described in VHDL. *RT-PC* [ThDu94] is a VHDL description of the IBM RT PC processor, which has an 8 bit data-bus and a 24 bit address-bus. *Prawn* is an enhanced version of the *Parwan* RISC processor described in [Nava93], with the instruction set having been enhanced to include interrupt handling and conditional branches.

Two examples (*Volume* and *Answer*) are modeled as extended Finite State Machines, and are described in SpecCharts [GVNG94] language. *Volume* models an instrument for measurement of the volume of the human bladder and *Answer* models the controller of a Telephone Answering Machine.

Filter, *FFT* and *Beamformer* are three DSP-related applications, the first described in Verilog language, and the next two in VHDL. *Filter* [Ugur95] describes an *Interpolating Switchable 3rd Order FIR Filter* that samples its input either every two or four clock cycles depending on a switch input. *FFT* models the Fast Fourier Transform algorithm [CaSv93]. *Beamformer* describes the behavior of a Beamformer system, an example of a typical DSP application that involves the temporal alignment and summation of digitized signals from an N-element antenna array [BaGa93].

The set of examples titled *Memory* are C routines related to image processing [PTVF92] that were used in [KoND94] to validate some algorithms on memory synthesis.

The remaining examples include a variety of applications, like a barcode reader (*Barcode* - [BhBD93]) in VHDL; an E.C.G. Application chip (*QRS* - [BhBD93]) in VHDL; an algorithm for adaptive interpolation for digital audio signals (*Adaptive* - VeJaVr86) described in Silage; a *Period Counter* in Verilog, that counts the length of a complete cycle of an input signal in terms of the number of clock cycles it takes [Ugur95]; a Robotics application that describes in C, an algorithm for the computation of the Jacobian [PaMu93] of an open kinematic chain and a Differential Heat Release Computation algorithm, described in VHDL, that models the heat release within a combustion engine [CaSv93].

4 Summary

In this paper we presented an overview of the 1995 High Level Synthesis Design Repository. The designs are available from the design repository at U.C Irvine (anon ftp; site: *ics.uci.edu*; location: *pub/HLSynth95*). [PaDu95] gives a more detailed overview about the functionality and testing strategy of the individual designs in the repository.

We welcome any feedback on the design examples and their accompanying documentation. We also welcome the submission of more designs for future inclu-

Design Name	Design Description	Design Level	Description Style	Control features	Data Types	Test Vectors	Lines Of Code
FP_Adder	Floating Point Adder	Algorithmic Behavior	1 VHDL Process	Nested Ifs For Loops Proc/Func	Bit Vector Integer Enum	417 Cycles	640 (VHDL)
FP_Mult	Floating Point Multiplier	Algorithmic Behavior	1 VHDL Process	Nested Ifs For Loops Proc/Func	Bit Vector Integer Enum	169 Cycles	425 (VHDL)
FP_Divider	Floating Point Divider	Algorithmic Behavior	1 VHDL Process. Func in separate package	For Loop Case Stmt	Integer StdLogicVector	10 Cycles	410 (VHDL)
Prawn	CPU 8-bit, 40 instructions	Instruction Set Behavior	1 VHDL Process	Nested Ifs Case Stmt	Bit Vector	1600 Cycles	700 (VHDL)
RT-PC	CPU 8-bit, 119 instructions	Instruction Set Behavior	Multiple Entities. Functions in sep. pack.	For Loop While Loop Case Stmt	Subtypes Bit, int Array Overloaded op	900 Cycles	3000 (VHDL)
Barcode	Barcode Reader	Algorithmic/ High Level FSM	1 VHDL Process	Nested Loops	Subtypes Integer	1 Test Suite	110 (VHDL)
QRS	E.C.G Application Chip	Algorithmic/ High Level FSM	1 VHDL Process	Loops Nested Ifs	Subtypes Integer	4300 Cycles	280 (VHDL)
Adaptive	Adaptive Interpolation Algorithm	Algorithmic Behavior	Set of Silage Functions	Func Calls, Nested Loop	Multi Dimensional Integer Arrays	6 Test Suites	810 (Silage)
Volume	Bladder Volume Computation	FSM with Datapath	Set of Seq/Conc SpecCharts Behaviors	Transition Arcs, For/ While Loops	Bit Vector Integer Array	20 Test Cases	220 (SpecCharts)
Answer	Telephone Answering Machine	FSM with Datapath	Set of Seq/Conc SpecCharts Behaviors	Transition Arcs, For/ While Loops	Integer Bit Vector	23 Test Sequences	640 (SpecCharts)

Figure 1: Features (in brief) of Designs with Complete Information

Design Name	Design Description	Design Level	Description Style	Control features	Data Types	Test Vectors	Lines Of Code
Memory (7 models)	Image Processing Applications	Algorithmic Behavior	1 C function for each example	Nested Loops	2-Dimen-sional float Arrays	No Test Suite Available	Each ~20 Lines (C)
Filter	"Switchable" 3rd order FIR Filter	Algorithmic Behavior	1 Verilog module	If Stmt	Bit Vector	No Test Suite Available	35 (Verilog)
Period Counter	Period Counter	Algorithmic Behavior	1 Verilog module	While Loop If Stmt	Bit Vector	No Test Suite Available	90 (Verilog)
Beamformer	Filter	Vector Product/ Summation	1 VHDL Process	Nested For Loops (4 levels)	3-dim array of Integer	No Test Suite Available	100 (VHDL)
Jacobian	Robot Motion Computation	Algorithmic Behavior	Set of C Functions	For Loops	Struct Pointers 2-dim array of 'double' trigon. func	No Test Suite Available	450 (C)
FFT	Fast Fourier Transform	Algorithmic Behavior	1 VHDL Process	Nested While Loops	Array of Bit Vector	No Test Suite Available	145 (VHDL)
DHRC	Differential Heat Computation	Algorithmic Behavior	1 VHDL Process	While Loops	Array of Bit Vector	No Test Suite Available	100 (VHDL)

Figure 2: Features (in brief) of Designs with Incomplete Information

sion - preferably, those whose functionality does not overlap with that of the existing designs. This variety in the design examples is important and is in accordance with our goal of making realistic design examples available to the HLS community as well as to serve as a reliable basis for stimulating new research, as well as for meaningful comparison of HLS systems and algorithms.

5 Acknowledgments

We would to thank the following people who have contributed towards the development of the designs in the repository: Jesse Pan and Bob McIlhenny (FPAdder and FPMult), Alberto Nannarelli (FPDivide), Alfred Thordarson (RT-PC), Tadatoshi Ishii (Prawn), Franc Brglez (Barcode and QRS), David Kolson (Image Processing Applications), Lode Nachtergaele (Adaptive Interpolator), Fatih Ugurdag (Filter and Period Counter), Jie Gong (Answering Machine and Volume System) and Smita Bakshi (Beamformer, FFT, DHRC and Jacobian.)

We also thank Manu Gulati for his useful comments and suggested improvements on the Barcode and QRS designs.

This work was supported in part by SRC Grant 94-DJ-146. We are grateful for their support.

References

[BaGa93] S. Bakshi and D. D. Gajski, "Design Space Exploration for The Beamformer System," Technical Report 93-34, University of California, Irvine, 1993.

[BhBD93] S. Bhattacharya, F. Brglez and S. Dey, "Transformations and Resynthesis for Testability of RT-Level Control-Data Path Specifications," IEEE Transactions on VLSI Systems, September 1993.

[CaSv93] F. Catthoor and L. Svensson, "Application-Driven Architecture Synthesis," Kluwer Academic Publishers, 1993.

[DuRa92] N. D. Dutt and C. Ramchandran, "Benchmarks for the 1992 High Level Synthesis Workshop," Technical Report 92-107, University of California, Irvine.

[GVNG94] D. D. Gajski, F. Vahid, S. Narayan and J. Gong, "Specification and Design of Embedded Systems," Prentice-Hall, 1994.

[HePa90] J. L. Hennessy and D. A. Patterson, "Computer Architecture - a quantitative approach," Morgan Kaufman Publishers 1990.

[KoND94] D. J. Kolson, A. Nicolau and N. D. Dutt, "Integrating Program Transformations in the Memory-Based Synthesis of Image and Video Algorithms," Proceedings, ICCAD'94, pp 27-30, San Jose, CA, Nov. 1994.

[LaEr94] M. D. Ergegovac and T. Lang, "Division and Square Root - Digit-Recurrence Algorithms and Implementations," Kluwer Academic Publishers, 1994.

[Nava93] Z. Navabi, "VHDL : analysis and modeling of digital systems," McGraw-Hill 1993.

[PaDu95] P. R. Panda and N. D. Dutt, "1995 High Level Synthesis Design Repository," Technical Report 95-04, University of California, Irvine, 1995.

[PaMu93] F. C. Park and A. P. Murray, "Computational and Modeling Aspects of the Products-of-Exponentials Formula for Robot Kinematics," IEEE Transactions on Automatic Control, 1993.

[PTVF92] W. H. Press, et. al., "Numerical Recipes in C: The Art of Scientific Computing," Cambridge University Press, 1992.

[RTPC85] IBM RT PC Hardware Technical Reference (C), 1985.

[ThDu94] A. B. Thordarson and N. D. Dutt, "A VHDL Model and Testbench for the IBM RT-PC Risc Processor," CADLAB document, University of California, Irvine, 1994.

[Ugur95] H. F. Ugurdag, Personal Communication, 1995.

[VaGN94] F. Vahid, J. Gong and S. Narayan, "The SpecCharts/SpecSyn User's Manual," University of California, Irvine, 1994.

[VaNG91] F. Vahid, S. Narayan and D. D. Gajski, "SpecCharts: A language for system level synthesis," Proceedings of the International Symposium on Computer Hardware Description Languages and their Applications, 1991.

[VeJV86] R.N.J. Veldhuis, A.J.E.M. Janssen and L. B. Vries, "Adaptive Interpolation of Discrete-Time Signals That Can Be Modeled as Autoregressive Processes," IEEE Trans. Acoustics, Speech and Signal Processing, Vol. 34, No. 2, 1986.