Vertical Benchmarks for CAD

Christopher Inacio, Herman Schmit, David Nagle, Andrew Ryan, Donald E. Thomas,
Yingfai Tong[1], Ben Klass[2]
Dept. of Electrical and Computer Engineering
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213, USA
{inacio, herman, dnagle, ar39, thomas}@ece.cmu.edu

1. ABSTRACT

Vertical benchmarks are complex system designs represented at multiple levels of abstraction. More effective than component-based CAD benchmarks, vertical benchmarks enable quantitative comparison of CAD techniques within or across design flows. This work describes the notion of vertical benchmarks and presents our benchmark, which is based on a commercial DSP, by comparing two alternative design flows.

2. INTRODUCTION

Quality benchmarks are essential to the continuing progress of electronic design automation because they provide the only way to qualitatively judge the effectiveness of CAD techniques. Benchmarks should, therefore, be representative of the size and diversity of designs encountered in industry. They should also provide for the evaluation of the effects of a particular CAD technique on the entire design flow. For example, if a logic synthesis technique reduces gate counts but makes a design harder to place and route, it may not be worthwhile. Finally, benchmarks should include some standard testing methodology, to provide some level of assurance CAD techniques construct functionally correct designs.

Table 1 lists some the popular CAD benchmark suites, and compares some of their characteristics. All of these suites are compilations of small, difficult or esoteric modules encountered in industry and academia. Within a suite, all benchmarks are specified at the same level of abstraction. Only the PREP benchmark suite offers any testing or functional verification capability. Clearly, these benchmarks suites do not meet the expectations outlined above; a fact long recognized [3].

1. Presently with Motorola, Austin, TX.
2. Presently with Cognex, Cambridge, MA.

DAC 99, New Orleans, Louisiana

This paper proposes the concept of a *vertical benchmark* and describes an example of a vertical benchmark, the CMU-DSP, which is being made available to the CAD community via the World-Wide Web. A vertical benchmark is a system-level design that includes multiple representations of the design at different levels of abstraction, as illustrated in Figure 1. A vertical benchmark should also include a complete design flow that can be used to create each of the lower-level design representations from a higher-level representation as well as a testing methodology for each of the design representations. As shown in Table 1, the CMU-DSP is the only benchmark that includes multiple levels of design representation. Furthermore, in terms of gate count, the CMU-DSP is 4.6 times larger than the average design in the listed benchmarks.

Vertical benchmarks satisfy all the expectations previously enumerated, and address many of the shortcomings of current CAD benchmarks. Because they are complete system-level designs, vertical benchmarks represent the size and diversity of components in commercial ASIC design. Using a vertical benchmark, the impact of CAD techniques can be quantified in terms of system-level performance and

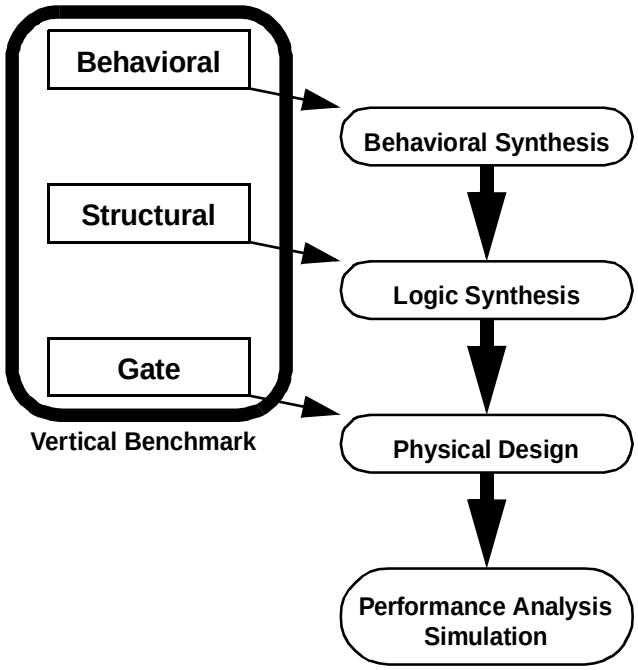

Figure 1. Vertical benchmarks

Table 1: CAD Benchmark Comparisons

	PREP[6]	ISCAS[1]	LGSynth[8]	HLSynth[3]	CMU-DSP
Behavioral	YES	NO	NO	YES	YES
Structural	NO	NO	NO	NO	YES
Gate	NO	YES	YES	NO	YES
Testing	Inconsistent[a]	None	Test vector	Hand crafted	Multi-level functional
# of Benchmarks	13	31	202	9	1
Avg. Gates[b]	112 (of 8)[c]	3149	1055	N/A	14,550

a. The PREP benchmark is composed of submissions from various sources. Each source for a benchmark develops their own test for their submitted circuit.

b. The average number of gates was obtained by compiling the source files for the benchmark with default options in Synopsys DesignCompiler version 1997.08 to the Duet HP14B cell library and reporting the number of cells.

c. Of the PREP benchmarks, we were only able to compile 8 of the 13 Verilog tests using Synopsys DesignCompiler.

area, rather than abstract notions of gate count and sub-system clock rate. Vertical benchmarks also allow for the evaluation of a CAD technique's effect on downstream tool performance, ensuring that a reduction in gate count, for instance, is not offset by bad routing performance.

Because vertical benchmarks include multiple levels of representation, they can be used to evaluate many different types of CAD techniques. In the CMU-DSP, the design is presented in behavioral, structural and gate-level representations. For the purposes of this paper, a behavioral representation is one in which some of the scheduling or allocation decisions have not yet been made. Structural descriptions are fully scheduled and allocated, but include abstract definitions of the contents of the functional blocks. For example, a structural description might have a multiplier, but there is no specification of whether that multiplier is implemented using a Wallace tree or an array. A gate level specification is one in which the architecture of all components is specified at a low level, typically in terms of members of a standard cell library. The CMU-DSP is therefore particularly useful to CAD researchers in the fields of synthesis (behavioral and logic), physical design, module generation, timing analysis, and circuit simulation. It could also be used to evaluate test synthesis and simulation techniques. Finally, because our benchmark is based on a DSP core, it may prove useful to evaluate techniques in IP-oriented and systems-on-a-chip CAD.

There have been a number of efforts to develop synthetic circuit benchmarks, primarily to assist design of FPGAs.[2] The disadvantage of these tools is that they provide no system context, because the circuits themselves do not perform a meaningful function. In addition, the CIRC and GEN tools [4] characterize real benchmark circuits and then create random circuits of approximately the same size with similar characteristics. Designs such as the CMU-DSP can serve as seeds for generating these designs.

Full applications have been used for a number of years to measure computer systems performance. The SPEC benchmark suites [7] use portable source code to measure combined compiler and computer performance. These benchmarks are not "vertical" in the sense that they do not consist of multiple levels of representation. Indeed, common lower-level representation of applications would not even be possible across different computer systems.

The remainder of this paper is structured as follows: Section 3 describes the architecture of the CMU-DSP. Section 4 describes the default commercial design flow and testing methodology used to construct the design. Section 5 describes two possible variations on this design flow and the results of using each flow in a system context. Section 6 describes the structure of the benchmark distribution and instructions on how to obtain the design. Finally, Section 7 concludes with a discussion of future work and other vertical benchmarks which are in development at Carnegie Mellon.

3. CMU-DSP CORE

The CMU-DSP is a 24-bit digital signal processor. Its core is modeled after the Motorola 56002 processor, although it is not binary compatible. The CMU-DSP, like the 56002, has a Harvard architecture, and can access two data memories and one program memory per cycle. As shown in Figure 2, the core of the CMU-DSP has four functional units: an address generator unit, a data arithmetic unit, a program control unit, and a bus switch. For further architectural reference see the Motorola manual [5].

Outside of the core there are significant differences between the commercial DSP and CMU-DSP. The memory and external interface of the CMU-DSP is primarily intended to make testing of the core as easy as possible. It does not support most of the functionality provided by the 56002. The memory can operate in two modes: an upload/download

mode, where package pins have exclusive access to the memory, and an execution mode, where the core has exclusive access to the memory. The CMU-DSP does not support interrupts. These two major simplifications make the core impractical for purposes other than research and testing. Furthermore, not all of the instructions supported by the Motorola core have equivalents in the CMU-DSP. All atomic bus instructions were discarded, along with some other general instructions. CMU-DSP does, however, contain the bulk of the arithmetic instructions along with a large degree of the addressing modes supporting those instructions. We can successfully run many common DSP application kernels such as FIR, LMS and FFT.

4. DESIGN AND TESTING FLOW

In the past, the largest impediment to creating vertical benchmarks has been the lack of standardization of CAD flows. Thanks to the university programs of many commercial CAD vendors, defacto standard design flows currently exist at most locations performing CAD research. While it is not strictly necessary that users of vertical benchmarks have the same commercial tools that were used to create our benchmark, it will ease the design process. At all levels of abstraction, Verilog is used to describe the design. The particular dialect of Verilog used at each level is the only tools-specific aspect of the benchmark.

4.1 Design Flow

Building the CMU-DSP requires using a number of tools throughout the design flow to accomplish different tasks. Currently, the core is represented as a combination of structural and behavioral level Verilog code. There also exists a behavioral instruction set architecture (ISA) representation of the core, but at present it is not completely compatible with the structural version and a behavioral compiler flow is not fully realized. In the future we will include a behavioral ISA version of the core which can be compiled into a structural representation that can flow through the rest of the design flow.

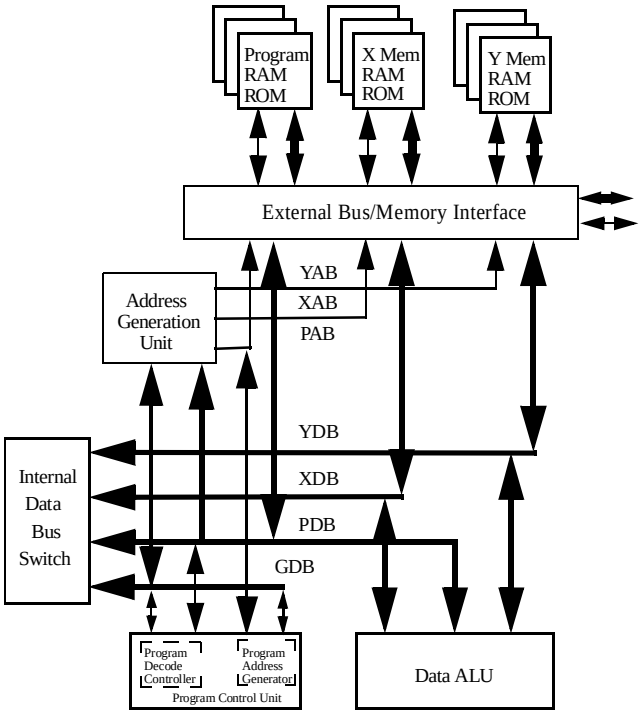

Figure 2. CMU-DSP Core Architecture

The CMU-DSP Verilog source can be divided into two sets of files; the datapath components which represent the multiplier, adders, registers, etc. within the computational core of the CMU-DSP and the control logic components which mostly represent control points for the datapath and instruction decode. The control logic is written in synthesizable behavioral Verilog which is compiled with Synopsys DesignCompiler version 1997.08. DesignCompiler maps the behavioral Verilog into structural Verilog based on the Duet HP14B version 4.0 cell library. The resulting structural Verilog is a standard cell based implementation of the control logic. The datapath components are then compiled using Duet Epoch version 4.0 and its module generators. The compiled control logic and datapath files are then placed and routed using Duet Epoch to generate a gate-level representation. Epoch can model the gate-level representation in various file formats useful for testing, simulation, or manufacturing.

4.2 Alternative Design Flow

Using the vertical structure of our CMU-DSP design, it is possible to explore different design flows and evaluate their merits. Our demonstration changes the method in which the datapath portion of the CMU-DSP is generated. In Section 4.1 we discussed the default design flow. We have developed an alternative in which the Epoch module generators for the datapath elements are replaced with using Synopsys tools and standard cells. The results of the tool replacement can be seen in Section 5. The flexibility of CMU-DSP to be adapted to new and different CAD tools and the new design flow, such as various datapath compilation methods is of enormous benefit to the CAD research community.

We developed the datapath Verilog files to be portable between various design flows. Currently, using a preprocessor we developed for the purpose, the same Verilog source files can be used with Synopsys DesignWare Foundation Libraries version 1997.08 with the Duet HP14B version 4.0 cell library to compile the datapath with using only standard cells instead of using Epoch module generators. The only change in this alternative flow is in the compilation of the datapath. See Figure 3 for a diagram of the design flows.

4.3 Software Test and Development

Previously, CAD benchmarks did not include the most robust test environments. Previous benchmarks often were built using a static array of test vectors hand-crafted to test the design at a specific level of abstraction. Most CAD benchmarks did not include more than a set of stuck-at test

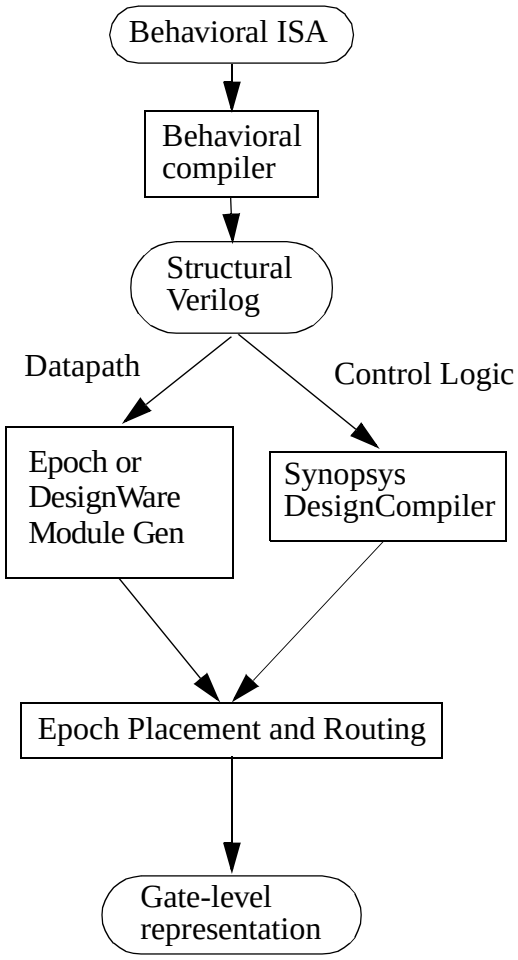

Figure 3. Design flow for CMU-DSP

vectors for each circuit. Vertically integrated benchmarks increase the value of time invested in test development since we propose fewer benchmarks used for more purposes. The CMU-DSP test environment is designed to be very extensible. The model is based on running programs on a simulator used as the reference, in our case the Motorola DSP simulator, and comparing the memory results from running the design on another simulator, typically Cadence Verilog-XL version 2.2.1. This testing model provides for efficient functional testing, but without complete validation. There is ongoing research on CMU-DSP using IBM's TestBench software and formal verification testing. A large part of the improved test framework is the ability to use high level languages and tools to develop test vectors and expected outputs for the CMU-DSP. This allows users to create their own specific tests with relative ease to increase the value of the CMU-DSP benchmark as a research vehicle. Some users, however, may have no interest in developing their own tests, and to this end, four tests are included with CMU-DSP. The following paragraphs discuss in more detail the test development flow and the four tests included with the suite.

Developing new tests for the DSP is done using Motorola's DSP 56K development tools. A test program is written in either assembly language or C language and compiled using either the assembler or C compiler into a Motorola DSP COFF file. This COFF file can then be simulated using the Motorola simulator to generate expected results, and converted into an input stimulus using the binary translator included with CMU-DSP. The CMU-DSP is then simulated using the test program as a stimulus and its memory is captured after simulation. The contents of the memories in the simulated CMU-DSP are compared with the memory contents from the Motorola DSP simulator. The advantage of comparing memory states in this way is that the test can be run using various simulators at different levels of representation of CMU-DSP. This multi-level testing ability lets the user see the effect of a change on any level of the design flow. The full test flow is shown in Figure 4.

For users who do not want to develop any tests for the CMU-DSP, we have developed a set of four tests. These tests include a 4-tap finite impulse response (FIR) filter, a 64-tap FIR filter, a least mean squares adaptive FIR filter, and a discrete fast Fourier transform (FFT). Harnessing the included tests is simplified by the included script `run-test`. The `run-test` script will use the included input memory files, run the Cadence Verilog-XL simulator and compare the memory output results telling the user if the system passed or failed. The 4-tap FIR test is not a representative example of a real application, but does exercise the core using a standard digital signal processing algorithm and can be run relatively quickly. The other three tests all take substantially longer to run, but do represent more typical DSP applications.

5. RESULTS

In Section 4 we have described two possible flows using the Synopsys and Duet tools. One flow uses Synopsys DesignWare components, and the second uses Duet data path module generators. Both use the Duet standard cell libraries, although when using the Duet module generators data path components are also used. Using Synopsys DesignWare components and Duet Epoch for placement and routing, the size of the core measures 11 mm^2 and runs at approximately 22 MHz. Using Duet Epoch for both structural compiling, placement and routing, and Synopsys DesignCompiler for the synthesis of control logic, the size of the core measures 20 mm^2 and runs at approximately 30 MHz. These two designs dissipate approximately 280 mW and 500 mW respectively. The results demonstrate that an interesting trade-off in size, speed, and power can be measured and quantitatively analyzed by adapting a portion of the vertical benchmark to a slightly varied design flow. The physical designs of both implementations are shown in Figure 5 and Figure 6.

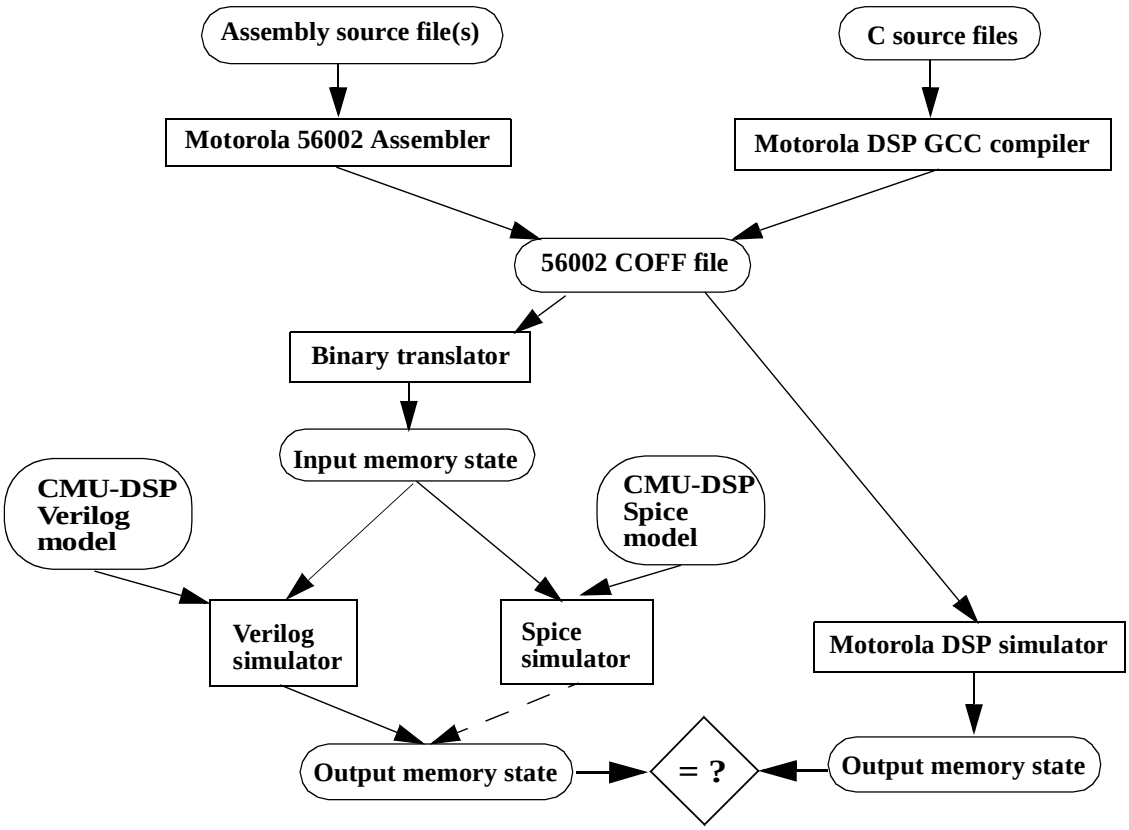

Figure 4. Functional Testing Flow

6. AVAILABILITY AND STATISTICS

6.1 Web Site Availability

These core designs are available over the World-Wide Web at http://www.ece.cmu.edu/~ceda. The DSP core package includes the Verilog source files, CMU-built additional programs, and scripts necessary to automate building and testing. Further, more detailed documentation on the design and construction of the DSP core is available from the web site. The documentation includes the source file layout and information on adding your own tests to the currently available test suite.

6.2 Source Repository

The source code for the CMU-DSP core is kept in a structural level description that can easily be mapped to transistor cells from an appropriate library to build the DSP core. The control logic source Verilog is at the behavioral

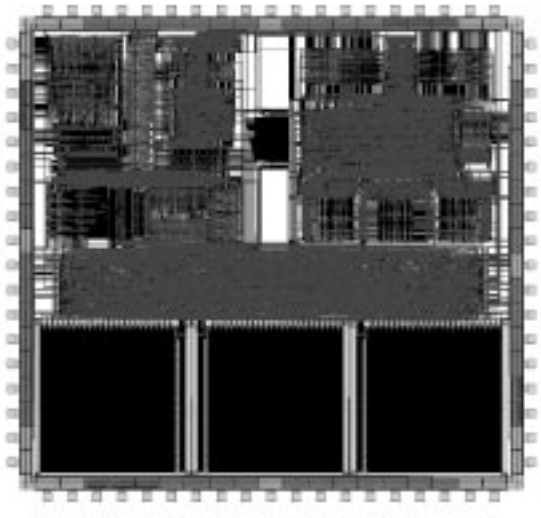

Figure 5. CMU-DSP with three 24-bit 1024 word RAMs using Duet Epoch design flow

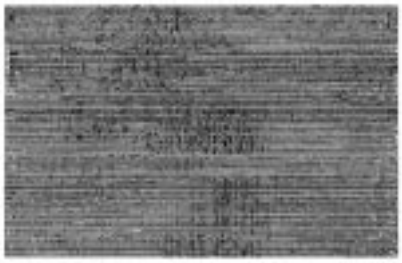

Figure 6. CMU-DSP (core-only, no RAM) using the Synopsys DesignWare Generators and Duet Epoch Cell Library

level, but we keep, and will distribute, the compiled versions of the synthesizable descriptions in the repository. The synthesizable descriptions are compiled with Synopsys's DesignCompiler and the results are kept in the repository. The advantage of this policy is that it allows us to keep a uniform compiled version of the synthesizable modules which everyone can use, we simply can copy and compile the DSP core, and it allows us to fix any errors generated from the DesignCompiler.

The Verilog source files are maintained with conditional compilation definitions around all of the structural components. The conditional compilation allows us define all of the structural components for both the DesignWare and Epoch libraries in a single source file for easing maintenance. Unfortunately, DesignCompiler does not understand Verilog conditional compilation statements. We have written and include a simple Verilog preprocessor in order to preprocess the Verilog source files for DesignCompiler.

6.3 File Outputs

The results of building the core are representations of the final product at various levels of representation. The testing flow can test the design at many of these levels. The outputs of the placement and routing tool are a Spice model, a standard delay format (SDF) back-annotated Verilog file containing resistor and capacitor delay information, and a CIF/GDSII layout description. The SDF Verilog files and the Spice model can be simulated and tested with the appropriate tools.

6.4 Lines of Code and Gate Count

The number of lines of Verilog code used to build the CMU-DSP along with a gate count obtained using Synopsys tools are listed in Table 2. The numbers presented represent the core only, applicable to both the default design flow presented in Section 4.1 and the alternate design described in Section 4.2. The default design flow also includes, as can be seen by examining Figure 5 three memories, an external memory interface, and the pad frame and pads. The transistor count for the Epoch module generator design flow, including memories and peripheral components is 640,815. The transistor count using the alternate design flow is 150,514.

7. Conclusions and Future Work

CAD research needs better benchmarks in order to remain relevant and productive. These benchmarks should have size and diversity similar to ASICs designed in industry. Vertical benchmarks, which include multiple levels of representation, allows benchmarks to be used by more CAD researchers and allow researchers at a particular level to evaluate the effects of innovations on the entire design flow. Finally, research into IP-based design and systems-on-a-chip require large, well-documented vertical benchmarks.

Development of two more vertical benchmarks is currently ongoing. These circuits are an ARM-like processor core and an FIR filter engine ASIC from MITRE Corporation. All designs will have a parallel structure to the CMU-DSP, including multiple levels of representations, system level performance metrics, and a testing methodology. All benchmarks will be available on the WWW. We are also developing a public domain standard cell library so that all physical design information can be made public.

Our long-term plan is to maintain a bibliography of usage of our design referenced in other papers. Our intention is to let people in the EDA community have a central resource for the use of our designs and the ability to easily find and reference future works based on it or referenced to it. This bibliography will also be available from the CMU web site.

Table 2: Design Information

Component	Lines of Verilog	Number of Gates
AGU	8672	5011
ALU	4434	6395
PCU	2226	2711
Bus Switch	868	433
Total	16200	14550

8. Acknowledgments

This work was supported by DARPA under order number A564 and National Science Foundation Grant Number MIP90408457. The authors would like to thank Nitzan Weinberg for his work on the CMU-DSP.

9. References

[1] F. Brglez, D. Bryan, K. Kozminski. "Combinational Profiles of Sequential Benchmark Circuits", ISCAS '89, pp. 1929-1934, 1989.

[2] J. Darnauer and W. Dai, "A Method for Generating Random CIrcuits and Its Application to Routability Measurement", in *4th ACM/SIGDA Int'l Symp. on FPGAs*, pp. 66-72, Feb. 1996.

[3] N. Dutt. "Current Status of HLSW Benchmarks and Guidelines for Benchmark Submission", *HLSynth '92 Benchmark*, Sept. 1992.

[4] M. D. Hutton, J. P. Grossman, J. S. Rose, and D. G. Corneil, "Characterization and Parameterized Random Generation of Digital Circuits," in *33rd ACM/SIGDA Design Automation Conference (DAC)*, pp. 94-99, June, 1996.

[5] Motorola Corporation, DSP56000 Digital Signal Processor Family Manual. 1995.

[6] Programmable Electronics Performance Corporation, URL: http://www.prep.org/synth.htm.

[7] System Performance Evaluation Corporation (SPEC), *SPEC CPU95 Version 1.1*, URL: http://www.spec.org, August 21, 1995.

[8] S. Yang. "Logic Synthesis and Optimization Benchmarks User Guide, Version 3.0", Microelectronics Center of North Carolina, Research Triangle Park, NC, Jan. 1991.