

On the Tradeoff between Productivity and Area Optimization in VLSI Design

Sreesa Akella, *Student Member, IEEE*, James P. Davis, *Member, IEEE*, and Hideaki Kobayashi, *Member, IEEE*.

Abstract— *In this paper, we examine, through direct experimentation, the tradeoff between design quality and performance, in terms of optimized design area, and the design time required to achieve this desired level of optimization. It is accepted that the costs associated with executing a VLSI design project must be managed. This includes finding an appropriate (and profitable) balance between area optimizations (affecting production costs) and design time (affecting NRE costs). We explore this tradeoff through direct experimentation as a precursor to identifying an appropriate set of design utility/cost functions to allow this design decision-making to be automated.*

I. INTRODUCTION

Design time, in this paper, is scoped to mean that time in a VLSI design project required for completing the design, from a given specification through generating an optimized gate-level netlist. Here, design time, and its associated Non-Recurring Engineering (NRE) costs, is directly affected by the amount of time spent by VLSI project team members on optimizing the design across a number of parameters—area being one of the most important. For high volume applications a considerable reduction in gate count would affect the per-unit cost, thus minimizing the overall cost. Employing the right optimization strategies, one can reduce the gate count considerably, but this effort has a cost in additional design time, thus increasing the time to market—affecting downstream product revenues [1].

Thus, a tradeoff and set of decision criteria—in the form of a cost/utility function—between this design time and the gate count reduction is to be obtained. This would give a design team an idea to what extent time spent on area optimization can be pursued without affecting the design time and thus the NRE cost.

Our experiments in exploring the space of tradeoffs between optimization of design area and the associated cost in design time try to obtain a shape of the tradeoff. Design benchmarks have been selected, designs executed, and optimization aiming at reducing the gate count has been performed. The time taken for each step in a stepwise optimization of the design has been recorded for the designs executed by a study group. This data is presented in this

paper, analyzed and we discuss a tradeoff point form which we can derive appropriate utility function candidates.

II. GATE COUNT AND OPTIMIZATION STRATEGY

Logic synthesis is a central design step for most ASIC and FPGA design projects today. The objective of obtaining an optimal gate-level netlist using an automated synthesis tool has been discussed as being central to an iterative high-level design-for-synthesis methodology [3], [6]. Removal of timing information and non-synthesizable constructs, technology mapping and defining area and timing constraints are involved in this activity [7]. The end result for the scope of our analysis is generating a gate-level netlist from RTL code, apart from achieving area, timing, and power optimization through suitable constraint inputs.

The gate count of a design is an accepted metric for size of a VLSI design, and thus, reducing gate count, or optimizing against it, reduces design size and thus per-unit cost. Optimization is generally achieved through applying optimization strategies, well understood for many VHDL and Verilog design styles [8], [9]. Adopted strategies vary according to design complexity, target technology and application. Design tradeoffs are made against these optimization strategies in order to maximize meeting as many design constraints as possible, based on objective criteria against which the design project itself is being gauged (such as time to market, design size for purposes of packaging requirements, design speed for application conformance, etc.).

The problem of optimizing gate count (as a function of product cost) and design time (as a function of NRE cost) could be considered as a specialized problem of the Discrete Time-Cost Tradeoff class [13]. In this class of problem, the objective is to model projects with discrete resources, such as manpower, that have to be allocated to specific project activities, where cost functions are calculated according to project tasks and applied resources. Of interest to our work is the formulation of this tradeoff as it pertains to VLSI design activities. Bouldin [15], van der Hamer [14] and others have investigated formulations of the VLSI design space in an attempt to give designers the means to evaluate and manage tradeoffs among criteria efficiently, to control complexity, and thus design cost, through use of automated tools.

Our work attempts to investigate one set of design criteria, namely gate-level optimization, against design time, so as to identify general characteristic functions that can be used in a similar decision-support context. The relationship of this tradeoff to project and product development costs has been discussed elsewhere [16].

Manuscript received April 12, 2002.

J. P. Davis, H. Kobayashi and S. Akella are with the Department of Computer Science and Engineering, Swearingen Engineering Center, University of South Carolina, Columbia, SC 29208 USA. (J. P. Davis phone: 803-413-3484; fax: 803-777-3767; e-mail: jimdavis@cse.sc.edu).

III. AN ACTIVITY-BASED ANALYSIS MODEL

A. Experimental Framework

An activity-centered model has been in use for some time, as shown in Figure 1, and can be used for measuring productivity in HDL-based design [2], [3], [4], [5]. Each activity requires a certain effort to produce an output according to input. This effort is quantified by measuring the effort required for each activity. Two types of resources are required for each activity; these are *human resource* and *computer resource*. Time required for each activity depends on factors such as size and complexity of design, designer experience in design methodologies and tools, and availability of reusable IP component libraries.

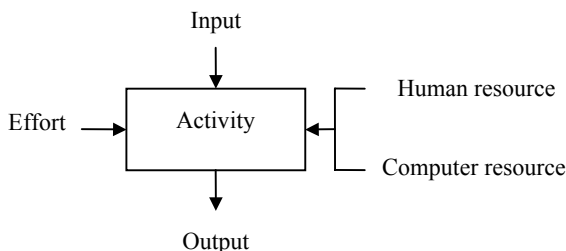


Fig. 1. An activity-centered model.

B. Experiment Resources

We subjected the design exemplars to graduate students with appropriate VLSI design backgrounds. These were 2nd-year Masters students with more than 5 courses in HDL-based design—roughly corresponding to the skills of a 2-3 year junior-level design engineer. In addition, students have performed design internships in industry, working on VHDL design projects, and thus have experience in working on design teams. These criteria were defined in order to identify qualified candidates to participate in this study.

The subjects worked in a UNIX environment for VHDL-based design activities such as Design entry (textual), simulation, and logic synthesis. Synopsys VSS was used for simulation and the Synopsys Design Compiler/Analyzer tool set was used for synthesis.

IV. THE DESIGN ACTIVITIES BEING OBSERVED

A. Overview of the Experiment Exemplars

In the experiment, we chose two design benchmarks, both in the range of 5K-15K gates in size, and employed HDL-based design methods as recommended by Bhatnagar [7], Lee [11], Chang [12] and others. For each project, the time taken for each design activity was measured and recorded in an Effort Distribution Table (data not included in this paper, but available by request). Once the design tasks were completed, the total design time (as defined earlier for purposes of this paper) was measured and tabulated.

The designs created by the study subjects were then optimized for reduction in gate count with the objective of reaching an overall 15% reduction in gate count without loss

of design functionality. The strategy was to apply incremental design optimizations to the VHDL code, such that reductions could be realized in four steps; say 2.5%, 5%, 10%, and 15%. These intermediate reduction targets were selected as initial test points in order to capture design-time measurements.

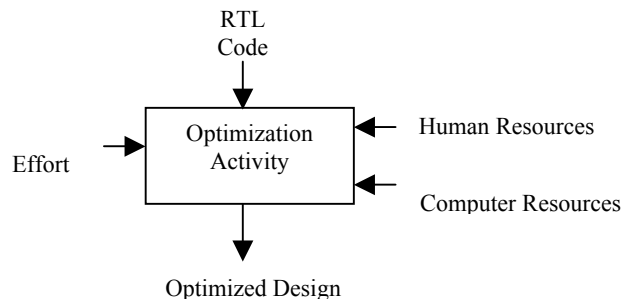


Fig. 2. An activity-centered model for the Optimization step.

The time taken for achieving each of these reductions is measured and recorded. As part of the design activity measured, in terms of design time, was the selection of suitable optimization techniques by the designer subjects. The design activity model can be depicted in Fig. 2.

B. Optimizing Designs

An acceptably optimized design is one having met timing requirements and occupying the smallest area. The optimization can be done in two stages: one at the code level, the other during successive passes through logic synthesis. The optimization at the code level involved modifying RTL-level VHDL code that had already been debugged and subjected to functional verification testing through VHDL simulation. As observed in [7], this level of modification to the RTL code is generally avoided because it can lead to inconsistencies between simulation results before and after such modifications. Therefore, we included re-simulation time in the data collection to account for this need to verify model consistency between optimization steps.

C. Optimizations Employed

Model optimizations are important to a certain level, as the logic that is generated by the synthesis tool is sensitive to the RTL code that is provided as input. Different RTL codes generate different logic. Minor changes in the model might result in an increase or decrease in the number of synthesized gates.

A logic optimizer reaches different endpoints for best area and best speed, depending on the starting point provided by a synthesized netlist from the RTL code. The different starting points are obtained by rewriting the same HDL model using different constructs. The strategies used and measured included: Resource Allocation, common sub-expression reuse, common factoring, flip-flop and latch elimination, code rearranging, which were observed to be effective in obtaining better quality designs (i.e., resulting in smaller gate counts without loss or variation in functionality).

For additional design optimization passes, considerable experimentation and iteration through the steps of design analysis and logic synthesis were performed to achieve

minimum area. In addition, the designers attempted to maximize design speed within the optimized area “envelope”. The process of analyzing the design for speed and area to achieve the fastest logic with minimum area involves exploration of the design space by attempting different techniques and observing the movement of the optimized design through the search space—i.e., was the design moving towards the desired reduction targets fast enough.

Various optimization techniques were employed using the logic synthesis tools, such as flattening and structuring, removing hierarchy, and area optimization using tool constraints.

V. THE DESIGN BENCHMARKS EMPLOYED

A. Network Repeater

An 802.3 Ethernet network repeater circuit was chosen as the first benchmark, based on an architecture derived from [17]. The function of the repeater is to create the logical equivalent of a shared medium, over which network nodes contend for use as a shared, multiplexed resource. Each node on the network connects to a common repeater through a segment of twisted pair coaxial cable. The repeater logically joins cable segments to create a larger network. This scheme improves reliability and performance, as it isolates network nodes with faulty connections thus preventing them from disrupting the rest of the network.

A network repeater's basic function is to retransmit data that is sent from one port to all other ports. Transceivers perform the electrical functions needed for interfacing the host ports to the repeater core logic. The basic objectives for our repeater design are:

- Detect carrier activity on ports and receive Ethernet frames on active ports;
- Forward the Ethernet frame to each of the active ports;
- Detect and signal a collision event throughout the network;
- Support interoperability of various physical layers (10BASE2, 10BASE-T, etc.);
- Provide centralized management of network operations and statistics; and,
- Partition bad segments

To summarize the functions of the repeater: 1) In general, the repeater receives data from one port and retransmits it to the other ports; 2) it detects collisions, activity, and errors, generating and transmitting the appropriate symbols under these conditions; and 3) it detects jabbering and partition conditions, asserting **tx_en**, **rx_en** as appropriate. To accomplish the functions required of the network repeater, the incoming data must be buffered and the correct data generated. The buffered data must be multiplexed with other data streams, depending on which should be transmitted to the active ports. Given the complexity of the problem at hand, we

partition and abstract the design into manageable units as follows:

- Port controller
- Arbiter
- Clock Multiplexer
- Symbol Generator and Output Multiplexer
- Core Controller

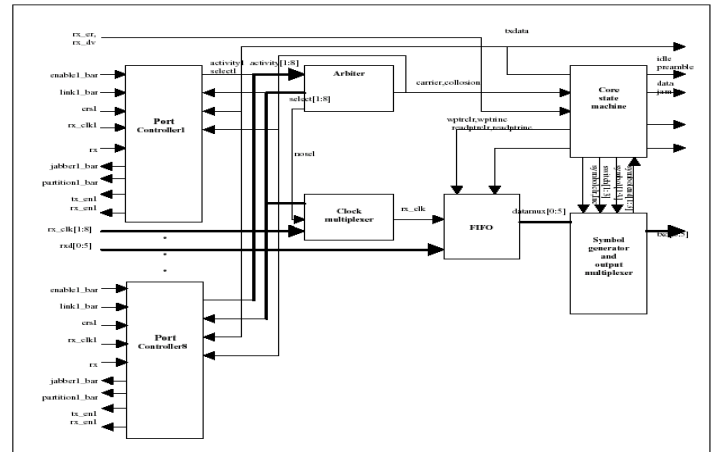


Fig. 3. Block diagram of Network Repeater

B. Datapath of MIPS R2000/3000

The second benchmark was a larger design and thus was considered as the suitable second choice when considering the impact of design scale on the experiment. The design description and subcomponent information is given below.

The objective of this benchmark is to design the data path of the MIPS processor for an instruction subset [18]. The control unit design is not part of the design benchmark. The main functional unit takes control signals as inputs along with clock and reset signals. It has ALU results, output of the Mux6, and a second output of the registers as the output ports. The external Reset is used to reset the general-purpose registers.

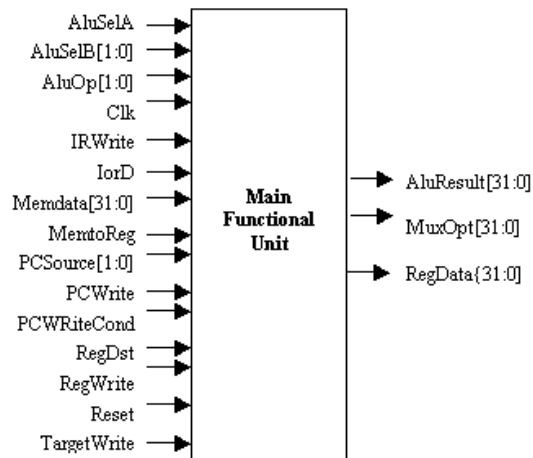
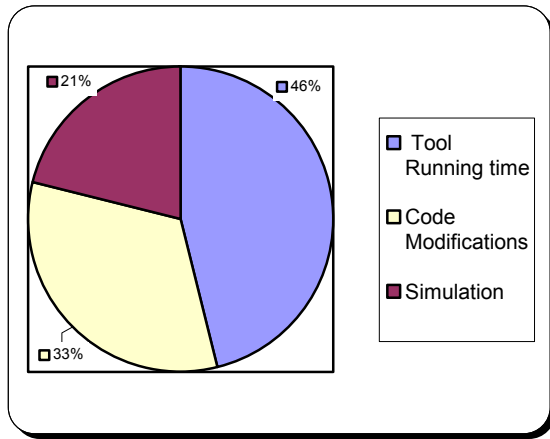


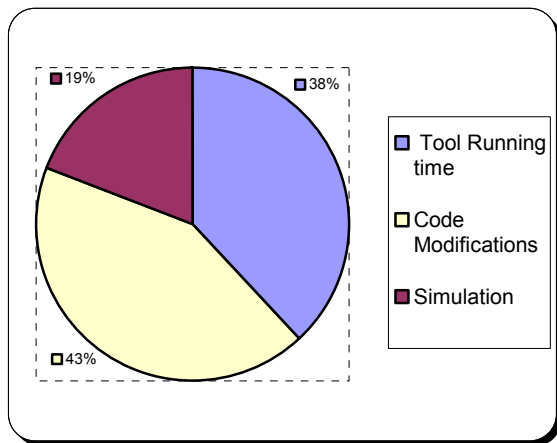
Fig. 4. Block diagram of MIPS R2000 circuit.

The top-level block diagram, generated as an output from logic synthesis, details the architectural sub-components and

The total time taken for each activity in the whole optimization experiment is quantified and presented in Fig. 7. The graph indicates a major portion of effort was put into VHDL code modification and the tool running (i.e., synthesis) activities.



(a)



(b)

Fig. 7. Effort distribution for each activity (a) DS1 (b) DS2

The total time taken for each optimization step, and the amount of optimization achieved, is shown in Table 2 for both the designs DS1 and DS2. Each row in Table 2 represents each of the 4 optimization steps presented earlier.

Time(mins)		Gate Count		%Reduction	
DS1	DS2	DS1	DS2	DS1	DS2
136	239	5388	10641	4.72	5.82
586	888	5028	10405	7.24	7.91
1192	1540	4989	10207	9.95	9.66
1856	2218	4773	10163	10.82	10.05

Table 2. Total Time for each optimization level in both designs.

The reduction in gate count as a percentage of the original gate count is used to quantify the amount of optimization achieved and is represented as %Reduction. Fig. 8 indicates

the optimization achieved corresponding to the effort (time in minutes).

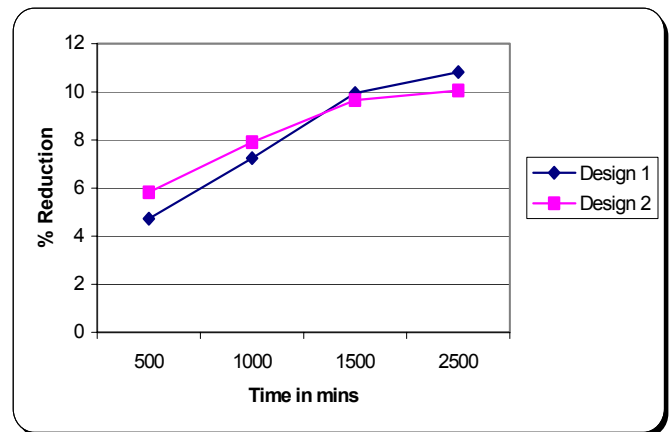


Fig. 8. Percentage reduction in circuit size versus design time.

It is evident from the graph plot that the amount of optimization achieved increases considerably during the initial optimization steps, but levels off in the later stages, even as the effort put forth by the designers increase—indicating a diminishing return on effort expended. The curve looks to be similar for both the designs indicating the closeness in the data obtained in two different design samples.

Fig. 9 illustrates the fact that the effort put forth increases drastically from Step 1 to Step 2 and by a considerable margin from Step 2 to Step 3. It then shows a gradual increase from Step 3 to Step 4. Thus there is reduction in the amount of optimization achieved for each step corresponding to the effort put in.

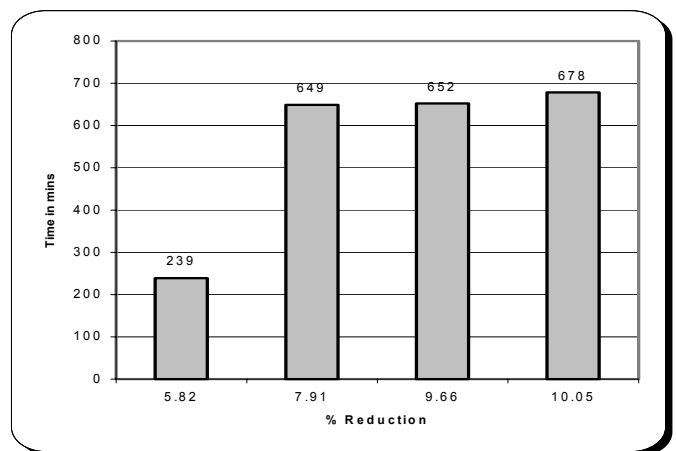


Fig. 9a.

Fig. 10 points out this fact in a much better way. In this graph, the total time taken for the optimization experiment is divided into two equal halves and the %Reduction achieved in each is quantized and compared. It is evident that the amount of optimization achieved in the first half is almost twice that achieved in the second half.

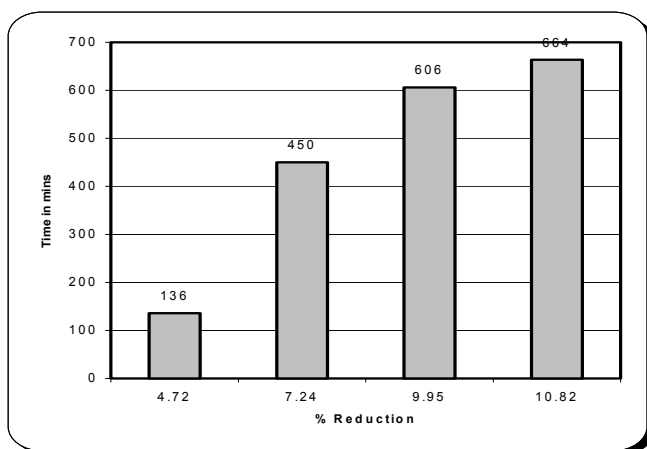
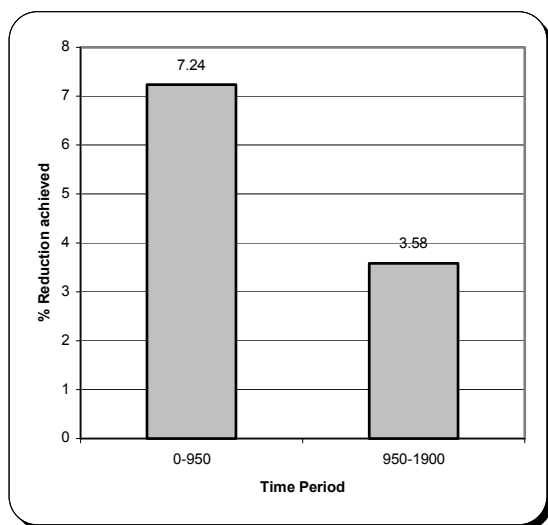
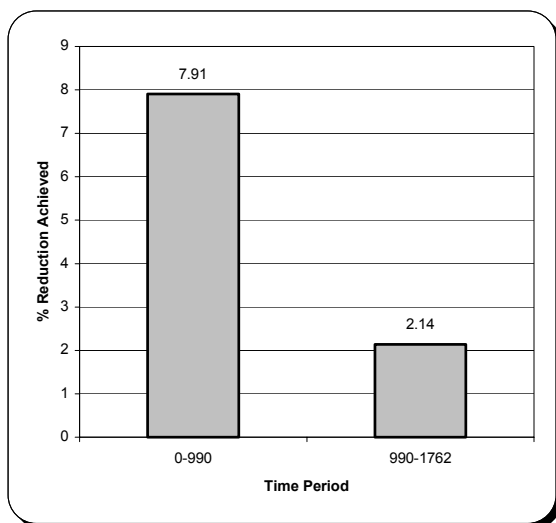


Fig. 9b.

Fig. 9. Step-wise effort distribution (a) DS1 (b) DS2



(a)



(b)

Fig. 10. The percentage reduction achieved in each half of the optimization period. (a) DS1 (b) DS2

From the above data, the obvious conclusion is that we see the clear decrease in designer productivity as a function of increased benefit of optimization versus design time. Furthermore, we see that there is some consistency of observed results across the two design benchmarks.

VII. CONCLUSIONS

The two factors affecting the cost in ASIC design are the design time impacting the NRE cost and the gate count impacting the per-unit cost. The shorter the design time, the better we can manage the non-recurring engineering (NRE) costs associated with making initial design turns, particularly for smaller applications in which a 10% area reduction would not have a great impact on the per unit cost. However, for larger designs, a 10% reduction in area due to optimization effort put forth prior to the point of diminishing return on effort would have a more significant impact. If this is achieved in a reasonable time frame, we can effectively balance our NRE and per unit cost, thus minimizing the overall design cost (taken to be the sum of NRE and per-unit costs [16]).

Thus, we observe an invariant tradeoff between the gate count optimization and the design time expended. In our method, we define metrics and collect measurement data for a series of experimental design benchmarks that seem to indicate that this is the case. We believe there is some predictive capability in this approach—that will require further study and scaling to larger designs. However, at this point in our analysis, we believe this evaluation method can provide a design team or design manager with the capability to perform impact analysis of planned efforts on area reduction activities, showing how these activities will affect the design time and, therefore, the NRE cost associated with the project.

The data obtained from the design experiment shows that percentage reduction obtained during the first half of the optimization activity has far greater impact than the second half, giving approximately a 10% reduction in gate count. The second half of the optimization time yields decreasing productivity results by comparison. This gives us a clear idea that a goal with respect to reduction in gate count (through judicious use of optimization techniques) is to be realized, then a goal of achieving a 10% reduction in circuit gate count can be considered in the range most likely to be obtained with minimal impact to NRE.

Future studies will attempt to scale the results and see whether we observe similar patterns in productivity and ratios of optimization to design time for much larger designs, and across a wider range of design types.

REFERENCES

- [1] J.F. Wakerly, "Digital Design: Principles and Practices", 2nd Edition, Prentice Hall, 1994.
- [2] Y. Jawchinda and H. Kobayashi, "Quantifying IP Resuse for Increasing Design Productivity: A Case Study," HDLCON '99.
- [3] R. Prasad and H. Kobayashi, "Multi-Methodology design: An Experimental Comparison", in Proceedings IVC-96, 5th

- International Verilog HDL Conference, Santa Clara, CA, Open Verilog International, 1996, pp. 45-49.
- [4] Makarand Joshi, "Analysis of HDL-Based Design Productivity," Thesis, 1996, University of South Carolina.
 - [5] Debashish Bose, "Increasing HDL-based Design Productivity through Design Reuse: A Case Study," Thesis 1996, University of South Carolina.
 - [6] J. P. Davis, "High-level Design-for-Synthesis: the Next Step", in *EDA&T 95: Conference on Electronic Design Automation and Test*, Asian Sources Media Group, 1995, pp. 378-391.
 - [7] H. Bhatnagar, *Advanced ASIC chip synthesis: Using Synopsys Design Compiler and Primetime*, Kluwer Academic Publishers, 1999.
 - [8] J. Bhasker, *Verilog HDL Synthesis, A Practical Primer*, Star Galaxy Publishing, 1998.
 - [9] J. Bhasker, *A VHDL Synthesis Primer, 2nd Ed.*, Star Galaxy Publishing, 1998.
 - [10] D.L. Perry, *VHDL, 2nd Ed.*, McGraw-Hill, 1998.
 - [11] W. F. Lee, *VHDL Coding and Logic Synthesis with Synopsys*, Academic Press, Inc., 2000.
 - [12] K.C. Chang, *Digital Systems Design with VHDL and Synthesis, An Integrated Approach*, IEEE Computer Society, 1999.
 - [13] Martin Skutella, "Approximations for the Discrete Time-Cost Tradeoff Problem," *Proceedings of the Eighth Annual ACM-SLAM Symposium on Discrete Algorithms*, Society of Industrial and Applied Mathematics, 1997, pg 501-508.
 - [14] P. Van der Hamer, "A System Simulation Framework", *Proceedings of Electronic Design Process 2000 Workshop*, <http://www.eda.org/edps/edp00.html>.
 - [15] D. Bouldin, "MODA: The multi-objective Design Advisor", <http://microsys6.engr.utk.edu/ece/research/microsys/tolnas/section.html>.
 - [16] M. J. S. Smith, *Application-Specific Integrated Circuits, 2nd Ed.*, Addison Wesley Publishing, 1997.
 - [17] AMD, *Ethernet/IEEE 802.3 Family: 1992 World Network Data Book/Handbook*, Advanced Micro Devices, 1992.
 - [18] J. L. Hennessy and D. A. Patterson, *Computer Architecture A Quantitative Approach, 2nd. Ed.*, Morgan Kaufmann Publishers, 1996.