Quantifying Design Productivity: An Effort Distribution Analysis

Makarand Joshi and Hideaki Kobayashi
Department of Electrical and Computer Engineering
University of South Carolina
Columbia, SC 29208
E-mail : makarand@ece.scarolina.edu

Abstract: *This paper presents basic process models for textual and graphical HDL-based design. The models are used to measure effort (time) required for various design activities, with an aim to quantify design productivity. Effort-distribution in man-minutes is used as a parameter to evaluate design productivity. Design quality, defined as a probability that the design meets its specifications, is plotted for various design activities. We also discuss the resources that are essential to perform each design activity. These experiments demonstrate that "effort-distribution analysis" is useful for real life HDL-based design projects.*

Keywords: *Design productivity, design quality, design resources, effort-distribution, graphical HDL, human interaction, ISO 9000, textual HDL.*

1. Introduction

Hardware description languages (HDLs) have enabled designers to represent complex digital systems at an increasingly higher level of abstraction. The bulk of the cost associated with a design project lies in the design environment comprising of computer resources such as software tools, HDL compilers, workstations, PCs, etc. and human resources [8]. Most cost models represent human resources in terms of person-months (PM) or man-hours (MH).

A unit activity model [2,3] has been used to analyze the effort-distribution over various design activities in two process models for top-down HDL-based design. Each design activity in a design process model provides an output in response to an input from its previous activity along with design effort and resources. The resource input includes designer's experience with the design language, tools and working environment. In the models discussed here, a synthesis activity at each level of abstraction is followed by a verification activity to ensure adherence to the design specifications. The *synthesis-verification* cycle is an essential measure to achieve an improved design quality.

Experiments were performed by two designers using a benchmark for HDL-based design, from a problem statement to working netlists in an educational environment.

2. Design process models

An HDL-based design process typically includes activities such as planning, system design and logic design [9]. Each of these activities contributes to the turn around time (TAT) and plays a significant role in determining the productivity of a design project.

2.1 HDL-based design process models

A typical textual HDL-based design process model is shown in Figure 1. The system design process extends from a problem statement to working netlists and involves a series of design activities [1].

A set of specifications (hereinafter referred to as "*spec*") describing system features is provided to the designer. A clear understanding of the *spec* should be made prior to starting the design. This activity is referred to as *conceptualization*.

A system can be hierarchically decomposed into subsystems considering the availability of design resources, while adhering to timing constraints. *System partitioning* provides a set of concurrently executing subsystems, each of which may be represented internally in an hierarchical fashion.

Conceptual model validation ensures the closeness of the partitioned system to the *spec*. Any deviation from the *spec* leads to an

erroneous system model to start with, which in turn adds to inaccuracy as the design process continues.

HDL modeling refers to representing the system behavior using a textual HDL. The model should handle timing constraints and manage data dependencies between subsystems using delay specifications and synchronization signals.

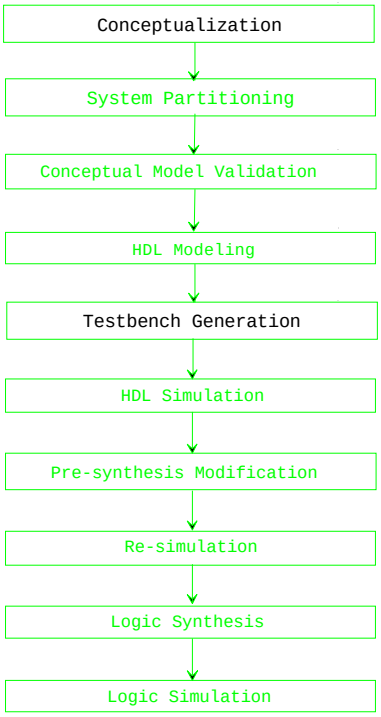

Figure 1. Textual HDL-based design process model.

Testbench generation produces a test suite for simulation of operating conditions based on the *spec* and constraints. The accuracy of the system design bears a direct relationship with the test coverage. The lost design quality (deviation from the *spec*) can be recovered through wide coverage of test cases. An exhaustive testing is highly desirable, however, it may be impractical due to the cost factor associated with testing.

HDL simulation refers to verifying the functionality of HDL description using testbenches. Simulation results in the form of waveforms or output tables are often used for verification of system behavior.

Pre-synthesis modification converts simulated constructs in an HDL description into synthesizable constructs supported by a specific library for logic synthesis. Non-synthesizable constructs are eliminated from the HDL description. The time required for *pre-synthesis modification* largely depends on the designer's experience.

Re-simulation ensures functionality of the HDL description, which may have been altered due to the removal of non-synthesizable constructs.

Logic synthesis is an automatic activity for generating a gate-level description in the form of netlists from an HDL description.

Logic simulation, an optional activity after the logic synthesis, verifies the functional and timing correctness of generated netlists.

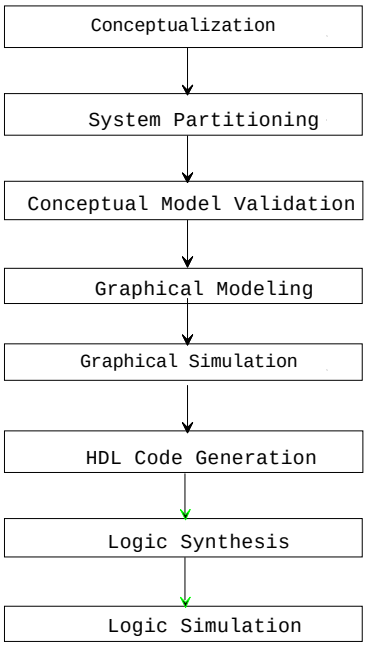

Figure 2. Graphical HDL-based design process model.

A graphical HDL-based design process model is shown in Figure 2. The three new activities included in the graphical HDL-based design process are described as follows :

Graphical modeling refers to representation of system behavior with the aid of a graphical utility. The graphical representation may be a flow chart or a state diagram in the case of a finite state machine.

Graphical simulation entails verification of the functionality of the system, by traversing through the graphical representation using a front-end tool. The flow of control in the design is illustrated during *graphical simulation* as test

inputs are applied interactively or via a test batch file.

HDL c*ode generation* refers to automatic generation of an HDL description and a testbench after *graphical modeling* and *simulation* are done.

In graphical HDL-based design, design entry may partly be done using a textual HDL for design components which may be better represented using textual languages.

2.2 Design resources

Designer's knowledge is an important resource, essential in order to perform various HDL-based design activities. *System partitioning* requires the designer to have a knowledge of system design in order to decompose a problem statement into subsystems. *HDL modeling* and *simulation* require the user to have an HDL knowledge, design experience and design tool skills. Tools may include an HDL simulator and a logic synthesis tool. In order to perform *pre-synthesis modification*, a designer should know synthesizable constructs for logic synthesis. A designer is required to know finite state machine modeling and flowcharting, in order to perform *graphical modeling*.

These resources comprise the resource input to the respective activities, as shown in the activity unit model [2].

3. Case study

As a case study, design experiments were performed by two designers using a benchmark. Designers A and B involved in the case study were engineering graduate students who had an HDL-based design experience for approximately two years. Designer A used the textual HDL-based design process, while Designer B used the graphical HDL-based design process. The effort (in man-minutes) required to perform each design activity was recorded and effort distribution graphs for each design process were plotted.

3.1. The benchmark

A programmable interrupt controller (PIC) was used as a benchmark to experience various activities shown in the design process models. A PIC acts as an interface between peripheral interrupting devices and a CPU. The PIC can handle upto eight vectored priority interrupts for the CPU. It accepts requests from the peripheral devices and determines which of the interrupt requests should be forwarded to the CPU. The PIC is set up for operation through a series of initialization command words (ICWs) timed with write (WR) pulses. Followed by ICWs, a sequence of byte long commands is provided to the PIC for specifying interrupt masks and modes of operation. These operational command words (OCWs) may be received more than once by the PIC, in order to reset or change the mode of operation.

In response to an interrupt on one or more interrupt lines, the PIC executes a typical interrupt sequence and transfers program control to a vectored memory location based on the type of interrupt(s) received.

The benchmark included control logic, datapaths and a memory interface, which are typically found to be elements of real life design projects.

3.2. Textual HDL-based design experiment

Figure 3 shows an effort distribution over the various activities in textual HDL-based design and a quality curve illustrating the accuracy of the design at every activity in the design process.

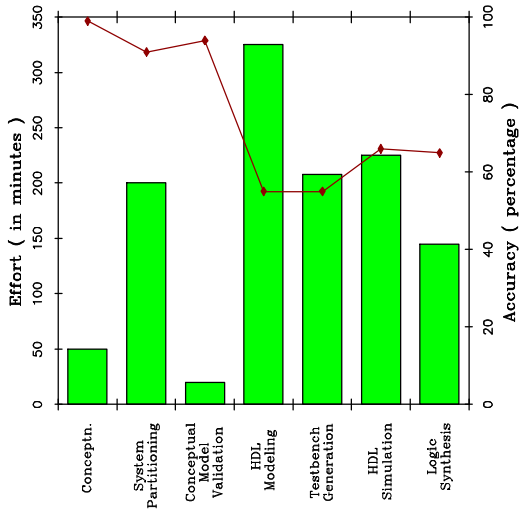

Figure 3. Design effort-distribution and quality curve for textual HDL-based design process.

Note that, *HDL modeling* contributes 28% of the total design effort. The modeling time can be reduced by utilizing a library of reusable generic components [5]. Also, 18% of the total design time is spent for *testbench generation*. An intuitive solution is the use of automated testbench generation (ATG) for *HDL simulation*. The use of ATG implies excellent repeatability and predictable time for simulation. *Pre-synthesis modification* can be eliminated if non-synthesizable constructs are avoided during *HDL modeling*. This was demonstrated by Designer A, who had a strong HDL design experience.

3.3. Graphical HDL-based design experiment

Designer B performed the design of the benchmark in a graphical HDL-based design environment. Figure 4 shows an effort distribution over the various activities in graphical HDL-based design and a quality curve illustrating the accuracy of the design at every activity in the design process.

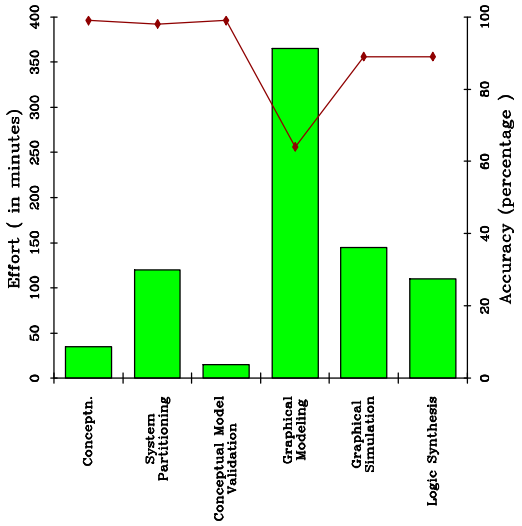

Figure 4. Design effort-distribution and quality curve for graphical HDL-based design process.

In graphical HDL-based design, 47% of the total design time is used for *graphical modeling*. The absence of *testbench generation* and *pre-synthesis modification* results in an increased percentage of the total design time being used for modeling. Intuitively, if the design process provides for modeling at a higher level of abstraction, a major portion of the total design time can be eliminated [8]. *Graphical modeling* also, creates a design documentation in the form of flow diagrams or state diagrams. This considerably reduces the time for design documentation. Note that *HDL code generation*, an automated activity, requires no design effort.

3.4. Design quality curve

In Figures 3 and 4, design quality (or accuracy) is plotted over the various activities involved in HDL-based design. The quality of a design is defined as the probability that the design satisfies its specifications [4]. The deviation from the *spec* results in a decline in the accuracy or design quality. Hence, every synthesis activity should be followed by a verification activity to ensure a recovery in terms of design quality, by maintaining closeness to the *spec*.

Design activities may be classified into two different types of activities, viz. *interactive* activities and *automated* activities. Interactive activities involve human interaction and introduce errors. An interactive activity such as modeling results in a decline in the design quality, while simulation helps in improving design quality. Consequently, *HDL modeling* and *HDL simulation* have a negative and a positive slope respectively, on the design quality curve. We assume that *logic synthesis*, an automated activity, does not affect design quality.

An accuracy of 100% is assumed at the start of a design process. It is a natural observation that the quality of a design tends to deteriorate during activities involving *human interaction*. The decline in design quality bears a direct relation with the amount of time required to perform that activity. This is indicated on the *design quality curve* with a negative slope proportional to the time involved in that activity. In Figures 3 and 4, the effort (time) required for *system partitioning* is greater for textual HDL-based design. As such, the decline in design quality during *system partitioning* is more for textual HDL-based design as compared to graphical HDL-based design.

System partitioning is performed at a higher level of abstraction in the design process. Hence, a deviation from the design specification in this

activity reflects as a relatively larger amount of decline in the design quality. This has been represented by having a steeper slope for activities involving human interaction at an abstract level in the design process. Similarly, the design quality recovery in *conceptual model validation* can be assumed to be faster than in *logic simulation.*

A state represented at an abstract level in the graphical description (e.g. a state diagram) corresponds to a section of the HDL description generated. Since *graphical modeling* is done at a higher level of abstraction as compared to *HDL modeling*, an improper assignment in a graphical description may affect the design quality much more than an improper assignment in an HDL description. Hence, the design quality curve has a steeper slope for *graphical modeling* as compared to *HDL modeling*.

HDL simulation requires a designer to generate a testbench for testing all the possible combinations based on the *spec*. It may require a considerable amount of time for generating an exhaustive testbench. This results in a decline in the overall design productivity and calls for a tradeoff between design quality and design productivity. In contrast, *graphical simulation* features an automated testbench generation based on the graphical model and test batch file, as the designer interactively simulates the design. An exhaustive testing results in a higher error recovery rate and better design quality.

4. Analysis of results

The two process models for HDL-based design exhibit a distinct difference in the effort distribution patterns. Results recorded from experiments indicate that activities involving human interaction, for example, modeling and simulation contribute 65% of the total design time in both textual and graphical HDL-based design. However, textual HDL-based design required 758 minutes for modeling, testbench generation and simulation and 145 minutes for synthesis related activities, whereas, graphical HDL-based design required 510 minutes for modeling and simulation and 110 minutes for synthesis related activities.

A large system may require large amount of testing and thus implies a requirement of some structure to organize and keep track of test cases. Such an organized collection of test cases is called a test suite. A good test suite includes test cases that maximize the likelihood of revealing design errors. *HDL simulation* requires generation of an HDL code as a testbench for evaluating the functionality of the HDL model for the system. A designer may have difficulty in accepting that his design has errors. Hence, it may be difficult for the designer himself to generate successful (error revealing) test patterns. In contrast, *graphical simulation* involves interactive usage of graphical tools to verify functionality and allows automatic testbench generation (ATG). ATG exhibits high repeatability and reliability in covering all possible test cases. Thus, ATG can be useful to obtain a good test suite and ensure better recovery of the lost design accuracy in modeling. Also, modeling in graphical HDL-based design is at a much higher level of abstraction. Hence, during simulation, a change can be modeled easily, in contrast to rewriting a section of the code in textual HDL-based design.

Pre-synthesis modification is not required in graphical HDL-based design because, HDL code generated for a graphical description includes only synthesizable constructs. This eliminates the "*synthesis bottleneck*".

Another difference between the two approaches is in *design documentation*. In textual HDL-based design, only an HDL description is available as design documentation. In contrast, graphical HDL-based design produces various design documents such as flowcharts, block diagrams and state diagrams, in addition to the HDL description. Quality management standards such as ISO 9000 require an extensive design documentation in order to prove the consistency and reliability of any given process [6]. If ISO 9000 is truly going to be a license for companies to do business, the design process needs to incorporate means to adhere to these standards.

5. Conclusion

We have presented two basic process models for HDL-based design. Textual HDL-based design uses a textual language for modeling, whereas graphical HDL-based design uses a graphical user interface (GUI).

It has been shown that design productivity can be measured in terms of effort (time) required to carry out all activities necessary to

complete the design. As modeling and simulation are done at a higher level of abstraction with the aid of automated tools, an increased design productivity can be achieved. The choice of an appropriate design process model can result in significant “time-to-market savings”.

Design quality is also an important factor for consideration in the choice of a design process. The deterioration in the design quality bears a direct relationship with the amount of human interaction and the time involved with interactive design activities. Automating design activities and allowing minimum human interaction proposes an enhancement in design productivity without compromising design quality.

References

[1] P. Kisson, H. Ding and A. A. Jarraya, "Structured Design Methodology for High-Level Design", *31st ACM/IEEE Design Automation Conference*, 1994.

[2] He and H. Kobayashi, "Process Management for Top-down HDL-Based Design", *APCHDL'94*, Japan, Oct. 1994.

[3] M. Joshi, H. Kim and H. Kobayashi, "Measuring HDL-Based Design Productivity: An Experimental Comparison", IEEE Int'l Verilog HDL Conf., Santa Clara, CA, Mar. 1995.

[4] E. J. Aas, T. Steen, and K. Klingsheim, "Quantifying Design Quality Through Design Experiments", *IEEE Design & Test of Computers*, pp. 22-37, Spring 1994.

[5] R. Goering, "HDLs get graphical tool", *Electronic Engineering Times*, May 24, 1993.

[6] W. Parzybok Jr., "As I see it", *Industry Week*, Jun. 6, 1994.

[7] M. Donlin, "ASIC complexity fuels drive to HDL design", *Computer Design*, May 1991.

[8] R. Moritz, "Graphical Entry Is a Must for HDLs", *ASIC & EDA*, Aug. 1993.

[9] P. Jalote , S. S. Muchnick, and P. Schnupp, "*An Integrated Approach to Software Engineering*", Springer-Verlag, 1991.