

Automatic Generation of Timing Annotated VHDL Models of VLSI Circuits

Ramesh Krishnamurthy, *

Design Automation Research Laboratory,
Department of Electrical Engineering,
The Ohio State University,
517 Dreese, 2015, Neil Avenue,
Columbus, OH 43210
email : ramesh@ee.eng.ohio-state.edu

Joanne DeGroat, Ph. D.

Design Automation Research Laboratory,
Department of Electrical Engineering,
The Ohio State University,
656 Dreese, 2015, Neil Avenue,
Columbus, OH 43210
email : degroat@ee.eng.ohio-state.edu

Abstract

This paper presents a system that automatically generates VHDL Models from VLSI layouts provided in transistor netlist format. The generated models can be simulated in available VHDL Simulation Environments and can thus provide an analysis of both logic and temporal behavior. Parameterized behavioral VHDL models have been developed for switch level and gate level components using simple but effective timing models. Program modules for extraction of hierarchical components and for generating from the extracted components, a VHDL structural model with instantiated components have been implemented in *PROLOG*. Comprehensive results and considerable speedups have been obtained for appreciable performance as compared to general purpose simulation and verification systems like *spice* and *irsim*.

* Current Address : Intel Corporation, 2111, NE 25th Avenue, JF1-71, Hillsboro, OR 97124
email : ramesh@ichips.intel.com

Section 1. Introduction

The Design Automation community has been addressing the very important issue of reducing the turnaround time from concept to implementation of a design. Several approaches are being developed by research and academic groups in this regard. Primarily, the development of tools for automating and enhancing the facilities at each phase of the development life cycle has been the major thrust. If one looks closely at the design cycle, it becomes clear that verification is the phase where most of the effort and resources are being spent before the design becomes a reality in silicon. This phase, where the performance of the design is verified against the specifications, becomes all the more important.

Simulation is one of the most common and extensive methods that can verify a design to that of the specifications. Simulation, however, validates the design only over a part of the specification domain, thus limiting its application for a robust and complete validation. Various approaches and techniques have been developed for extraction of characteristics from the final design, which can then be used in a variety of ways to verify the design over the specifications. Extraction of the circuit description, from the mask layout description, to a level of higher abstraction, with performance characteristics annotated into the extracted model, is seen as a very promising approach. If verification at this level reveals that the design complies with the specifications, the design need not even be simulated, it may simply be released for fabrication.

A design system based on these premises is the eventual goal of this research group. Towards this objective, several projects have been undertaken and the results have shown that there is ample scope for further work and research, to be able to define and identify techniques and methodologies for formal design verification.

Section 2. System Requirements

Tools and techniques for extraction of the circuit components into transistor netlists and simulation at the switch level are available and are being used both in the public domain and in industry. Simulation is available either as a general purpose circuit simulator, like *spice* or a switch level simulator with limited capabilities, like *crystal* or *irsim*. *Spice* has proven to be very inefficient in spite of its detail and accuracy for circuits of moderate complexity where hundreds of computations have to be made. Systems like *irsim* and *crystal*, by design, provide an optimistic estimate of the worst case behavior or provide timing information as an aside to logic simulation and verification. In-house and proprietary tools are neither available for discussion nor published in literature. Research efforts that have been reported vary from verification by default systems, to extensive switch-level VHDL model generation.

The expectations from this system in broad terms were as follows.

- be able to extract components at higher levels of abstraction than the switch level
- be as versatile as possible to be able to interface with a variety of design automation tools
- be testable/simulatable to completely verify the logic that the circuit implements and perform timing analysis on the design
- be as accurate as possible in timing analysis, sufficient to validate and verify the temporal behavior of the circuit.

VHDL [5] models were the best suited for the purpose as stated above. State-of-art simulators are available for directly simulating VHDL models. VHDL is also fast becoming the industry accepted standard format for all forms of information interchange about circuit specification, design, verification and production. Most VHDL platforms available also provide a wide repertoire of features including graphic output, interfaces with other programming and CAD environments.

Two prototype systems were examined in detail relating to this project. Dukes et. al., [1] at the Air Force Institute of Technology, developed a system for extraction of basic logical components from a transistor netlist description of the VLSI circuit. It was also capable of producing structural VHDL models consisting of instantiations of known components, that could be used to verify the logic behavior of the circuit. This system was later enhanced to extract the critical path and some trivial timing information. Navabi and Dube [2], at the Northeastern University have developed an effective parametrized switch-level VHDL model extractor from `magic` files. This system used a table driven approach to delay calculation by parametrizing the transistor geometries into the extracted switch-level components.

Section 3. Overview of The System

The approach taken in this system [3] is to extract components at the primitive gate level, and generate a structural VHDL model with instantiations of parameterized *level-n* circuit components. Each of these components are mapped to prewritten generic behavioral models. With a user supplied test bench, this complete VHDL description can be simulated to provide a good analysis of the temporal behavior of the circuit. Using first-order timing models of the p- and n-switches, generic models were designed to take into account the dynamic behavior of the circuit components under various conditions of the inputs switching and circuit configurations.

The flow-diagram in Figure 1 depicts the broad structure of the system and that of the generated VHDL model.

A flattened transistor netlist file, `.sim`, is obtained from the mask layout, `.magic`, file.

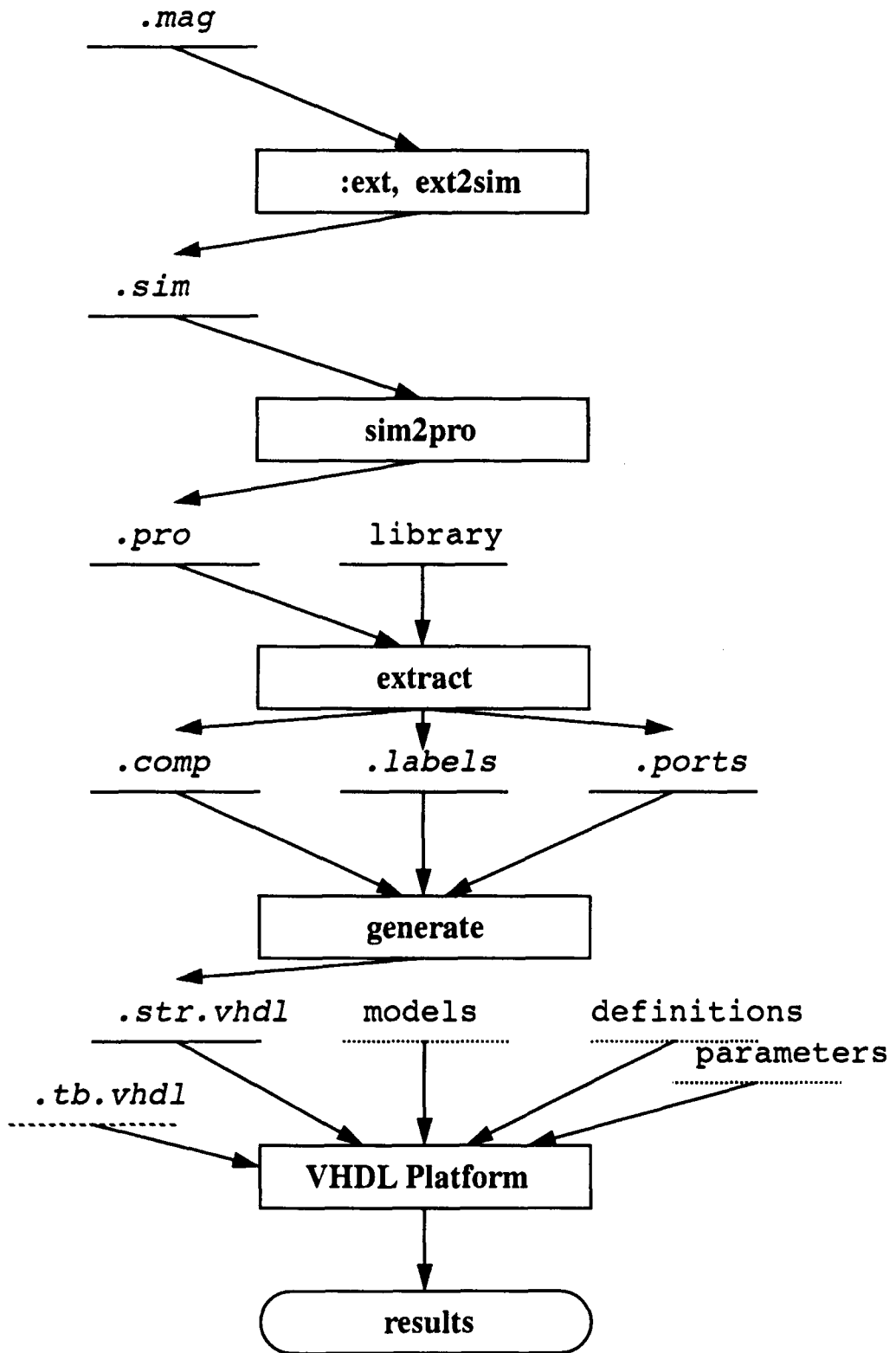


Figure 1 : System Overview

This is then filtered and converted into a *prolog* database file, `.pl`, by the program module `sim2pro`, implemented in C programming language. This *prolog* database representation is then processed by the program module `extract`, implemented in *prolog*, to generate several intermediate files, all in *prolog* database format. One of the files, `.comp`, is the list of extracted components, tagged with labels, transistor geometries and aspect ratios. Of the other files, `.labels`, is a list of intermediate node names that have been used in the component netlist, which will eventually be the signals in VHDL model. These files are then used by the `generate` module, implemented in *prolog*, to generate a structural VHDL model consisting of the components that were extracted in the previous stage. This files uses instantiations of generic models that are defined in the `models.vhdl` file, and most of the other process parameters and the calculation models are defined conveniently in the `defs.vhdl` and `params.vhdl` files. These VHDL modules along with a user supplied test-bench in `tb.vhdl` can be analysed and simulated on a VHDL Platform. The generated structural model can itself be edited to provide stimuli and simulate the model.

Section 4. Generic VHDL Models For Extracted Components.

Parametrized generic models have been developed for the following primitive gates in addition to unidirectional n- and p- switches under certain limitations and restrictions.

1. Invertor
2. Transmission gate
3. Controlled Invertor
4. 2-input NAND gate
5. 3-input NAND gate
6. 2-input NOR gate
7. 3-input NOR gate

The Generated VHDL model is a structural model in which the instantiated components are interconnected. The input and output ports of all components are modeled using two attributes, namely the logic value at that node and the total capacitive load at that node. The declaration of type `node_port` is shown in Figure 2.

```

TYPE node_port IS RECORD
    logic : std_logic;
    load  : total_cap;
END RECORD;

```

Figure 2. VHDL Type Declaration For *node_port*.

Since all ports and connections will report their internal gate capacitance or the internode

capacitance, a resolution function sums up all the reported capacitance. At any node or port in the circuit, at the very beginning of the simulation cycle, the total capacitance is calculated. This capacitance does not change as the simulation progresses, and thus provides an effective mechanism to parametrize the generic models to the capacitive loads that each output drives.

A First-Order timing model was developed for generic modeling of the extracted components, which provided an as accurate an estimate of the delay as possible in switching the output relative to the time when any of its inputs switch to a different value. Using a standard CMOS inverter model driving an approximated capacitive load, parametrized expressions for the propagation delay in switching to 1 (or 0, as the case may be) can be shown to be in the form

$$t_f = (\text{factor}_n)C_L Z_n$$

and

$$t_r = (\text{factor}_p)C_L Z_p$$

where C_L is the capacitive load that the inverter is driving, and Z_n and Z_p are the aspect ratios of the n- and p-transistors in the inverter circuit. factor_n and factor_p are scaled constants for the entire design that depend only on the process parameters. Since C_L , Z_p and Z_n are characteristic to individual components, the model is parameterized on these parameters.

Using the above parametrized model, the standard inverter model shown in Figure 3 exemplifies the calculation of propagation delay at simulation time and annotation into the results.

Eventhough the `std_logic` type used corresponds to the 9-level logic as defined in the IEEE 1164 standards, this system has been implemented for a 3-level subset inclusive of the '0', '1' and 'Z' values.

All gates implemented in this system behave like a standard inverter with parallel and series transistors depending on the input combinations. Equivalence relations for the effective aspect ratios of the composite transistor models for series and parallel transistor configurations have been used in combination with a state machine approach to implement the behavior of multiple input gates for all combinations and sequences of the input signal changes. The gates are characterized by the geometries of identified transistors and these geometries are passed to the generic models as the area and aspect ratios of the transistors.

The example of a two input NOR gate is given in Figure 4.

```

ENTITY invertor IS
  GENERIC (APU,ZPU,APD,ZPD : real);
  PORT    (a,abar : INOUT node_port);
END invertor ;

ARCHITECTURE behavioral OF invertor IS
BEGIN -- behavioral OF invertor

  a.load      <= capacitance((C0_p*APU + C0_n*APD));
  a.logic     <= transport 'Z';
  abar.logic  <= transport 'Z';

  set_output : PROCESS

  VARIABLE t_r    : time := 0 fs;
  VARIABLE t_f    : time := 0 fs;
  VARIABLE C_load : real := 0.0;

  BEGIN
    C_load := REAL(abar.load);
    t_r := (Fac_p * (C_load*ZPU))* 1 fs;
    t_f := (Fac_n * (C_load*ZPD))* 1 fs;

  WAIT on a.logic;
  IF (a.logic'event AND (a.logic = '1' OR a.logic = 'H'))
  THEN IF (abar.load > 0.0)
    THEN abar.logic <= transport '0' AFTER t_f ;
    ELSE abar.logic <= transport '0' AFTER t_d ;
    END IF;
  ELSE
  IF (a.logic'event AND (a.logic = '0' OR a.logic = 'L'))
  THEN IF (abar.load > 0.0)
    THEN abar.logic <= transport '1' AFTER t_r ;
    ELSE abar.logic <= transport '1' AFTER t_d ;
    END IF;
  END IF;
  END IF;

  END process ;
END behavioral;

```

Figure 3. VHDL Code For The Standard Invertor Model.

```

ENTITY nor2 IS
  GENERIC (
    APUa,ZPUa,
    APUb,ZPUB,
    APDa,ZPDa,
    APDb,ZPDb : real
  );
  PORT (a,b,anorb : INOUT node_port);
END nor2;

```

Figure 4. Entity Declaration of a 2-input NOR Gate

In the generated model, instantiations of this gate will be in the form shown in Figure 5.

```

n2r1 : nor2 GENERIC MAP( 12.0, 0.333333,
                          12.0, 0.333333,
                          6.0, 0.666667,
                          6.0, 0.666667 )
  PORT MAP (ain, bin,
            anrb );

```

Figure 5. Instantiation of a 2-input NOR Gate

The state machine model implements a Moore Machine. At each state, on a transition on any of the inputs, the machine transits to a new state and the output is scheduled to be driven to a new value (if required), and this output is held for the period when the gate stays in that new state. A *State* for a gate is a unique identification about the inputs and their current values. Thus for the 2-input gate there are at least four states, *a0b0*, *a0b1*, *a1b0* and *a1b1*. There is a need for one state at the beginning of the simulation when the inputs are all still unknown. State *azbz* is used to identify such a state where one or more of the inputs is in a undefined state. Since this implementation assumes that there are no bus nodes in the circuit, note that all signals will be in one of two logic states, '0' or '1' only as the simulation proceeds. In case of a NAND gate, when in a state where inputs are '1' and '1', either of the inputs switching to a '0' will drive the output high. But since the switching of the second input will result in a different delay in driving the output, an intermediate state is needed where a check is made if the second input also switched before the output could be switched. Two intermediate states, *ta0b1* and *ta1b0*, for A switching first and for B switching first are also included. The model for the 2-input gates is a state machine with the following seven states : *azbz*, *a0b0*, *ta0b1*, *ta1b0*, *a0b1*, *a1b0* and *a1b1*. The model for the 3-input gates has the following fifteen states : *azbzc*, *a0b0c0*, *a0b0c1*, *ta0b0c1*, *a0b1c0*, *a0b1c1*, *ta0b1c0*, *ta0b1c1*, *a1b0c0*, *a1b0c1*, *ta1b0c0*, *ta1b0c1*, *a1b1c0*, *a1b1c1* and *ta1b1c0*.

The following table shows a comparative analysis of the results obtained from simulating a standard inverter model under the implemented system and spice (version 3.e).

Table 1 : Comparative Analysis of Simulation Results of standard Inverter in *spice* and the Implemented VHDL Models

Load Capacitance fF	Aspect Ratio	Spice Model		VHDL Model	
		RiseTime 10e-8s	FallTime 10e-8s	RiseTime 10e-8s	FallTime 10e-8s
50	0.66667	0.42205	0.21335	0.36584	0.17849
80	0.66667	0.61387	0.31994	0.58535	0.28559
100	0.66667	0.81863	0.38771	0.73169	0.35679
150	0.66667	1.15062	0.56616	1.09753	0.53549
200	0.66667	1.51744	0.73233	1.46338	0.71399
300	0.66667	2.09560	1.07353	2.19507	1.07079
400	0.66667	2.75854	1.40803	2.92677	1.42798
500	0.66667	3.41109	1.74236	3.65844	1.78499
100	1.00000	1.09787	0.55441	1.09753	0.53549
100	0.66667	0.81863	0.38776	0.73169	0.35699
100	0.33333	0.42385	0.22769	0.36584	0.17849
100	0.20000	0.30591	0.16553	0.21950	0.10709
100	0.10000	0.22061	0.12988	0.10975	0.05354

Section 5. Conclusions

This system has successfully demonstrated the feasibility and practicality of automatically generating a timing annotated, testable, VHDL model from hierarchically extracted components from a lower level of design. Algorithms and methodologies developed for extracting components at the gate level from transistor netlists have been successfully implemented in *prolog*. Techniques for design error identification at different levels of extraction have been conveniently incorporated into the extraction process. The implementation itself has been done with an expert system approach, conducive to a design-style oriented environment. Simplistic and easily implemented timing models, designed to be tunable to desired process parameters and accuracy, were used in implementing generic parameterized behavioral models for the extracted components. By using VHDL as the intermediate format for representing the extracted model, it was also demonstrated that simulation can be performed independent of the platform used.

Section 6. Directions For Further Research

The experience gained with the development of the above system, motivates further work into extraction of components at higher levels of abstraction. One of the approaches that is being examined is to be able to use the extracted components in the definition and modeling of higher level components. The mechanism with which timing annotation is achieved needs to be examined using other approaches. For example, another approach is to be able to extract characteristic timing annotation, in terms of maxima, minima and nominal delay values. The goal is to be able to extract a model using multi-input, multi-output RTL like components, predefined and specified by the user, and be able to simulate it at that level of abstraction.

One of the issues that is intended to be examined in due course is the possibility that at some level of hierarchy, the process of validation of a design is performed at a level of abstraction that is same as that of the specification itself. This would be the ultimate tool that would enhance the productivity of a designer, and in more ways than one let him concentrate his efforts on the design activity than on the verification of the design iteratively.

Acknowledgments:

The authors gratefully acknowledge the Ohio State University Research Foundation, and the Department of Electrical Engineering, The Ohio State University for the financial support and facilities which made this research project a successful effort.

Bibliography

1. DeGroat, Joanne E, M. Dukes, and F. Brown, 'Generalized Extraction System for VHDL', July 1987-August 1990, WRDC Technical Report WRDC-TC-90-5021, Wright Research and Development Center, Wright-Patterson AFB, OH, August 1990.
2. Dube, J, *Switch Level VHDL Model Extraction For VLSI Circuits*, MS Thesis, Northeastern University, MA 1991.
3. Ramesh Krishnamurthy, '*Automatic Generation of Timing Annotated VHDL Models of VLSI Circuits*', MS Thesis, Department of Electrical Engineering, The Ohio State University, Columbus, OH, Summer 1992.
4. Berkeley Distribution of VLSI Design Tools, Computer Science Division, EECS Department, University of California at Berkeley, 1986.
5. IEEE Standard VHDL Language Reference Manual, ANSI/IEEE Std 1076-1987, New York, IEEE Press, 1987.