

Rating the Efficiency of VHDL Behavioral Models

John A. Wicks, Jr. and James R. Armstrong
The Bradley Department of Electrical Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA, USA 24061-0111
E-mail: wicks@birch.ee.vt.edu, jra@vt.edu

II. VHDL Modeling Styles

Abstract

Due to the great complexity of VHDL models that are created today, the amount of CPU time required to simulate these models and the amount of labor required to develop these models have become critical issues. The amount of CPU time required to simulate a model can be directly influenced by the efficient use of VHDL concepts in creating the model. Research in the determination of what VHDL concepts and modeling styles are most efficient will be discussed in this paper. The development of tests that can be run on VHDL models to reveal the efficiency of the code in the form of a numerical efficiency rating will also be discussed.

I. Introduction

When modeling in VHDL, under many circumstances, there are many different ways to accomplish the same goal. For example, in many cases either signals or variables can be used to represent the same object. In this case, the use of a variable would be more efficient because all signals require scheduling which causes the process of assigning new values to be slower, but variables require no scheduling and when a variable is assigned a new value the change happens immediately [1]. The purpose of this paper is to discuss the results of the comparisons of the simulation performances of different modeling styles that are semantically equivalent (Section II). Additionally, a system that determines a numerical efficiency rating (in terms of simulation performance) for VHDL models will be discussed (Section III).

As stated earlier, VHDL is a complex language that allows for the use of many different coding styles to accomplish the same goal. However, as systems that are to be modeled become increasingly larger, it is very important that modelers use the most efficient modeling styles. One of the most important reasons that efficient modeling styles must be used is to optimize simulation performance. Numerous experiments were conducted to determine what some of the most efficient VHDL modeling styles are. In each experiment, a VHDL model was created to perform a certain function using one particular style. Another VHDL model was created to perform the same function using an alternate style. The same test bench was used to drive both models. The simulation performance of both models would then be measured in terms of CPU time using a program that was written in C. Two different simulators were used in these comparisons. The simulators used are Vantage and Synopsys. Vantage is a compiled code simulator while Synopsys is an interpreted code simulator. Generally, compiled code simulators analyze slower and simulate faster, and interpreted code simulators analyze faster and simulate slower. It is very important to consider more than one kind of simulator due to the fact that some modeling styles are more efficient in one simulator than in another simulator due to the manner in which the various simulators are coded.

Some descriptions of the more interesting and useful experimental results are as follows [2]:

- (1) Variables perform better than signals in both Vantage and Synopsys, however, the percent difference was much greater using Vantage. As stated earlier, this is because signals require scheduling which causes the process of assigning new values to be

slower, while variables require no scheduling and a value change for a variable happens immediately.

(2) Case statements were found to be slightly (less than 5% difference) more efficient than an equivalent series of if-endif statements.

(3) Integer types perform much better than logic types with a greater than 20% difference.

(4) The attribute 'EVENT performs better than the attribute 'STABLE. Signal'EVENT is a function call that returns true if the signal has changed during the current delta cycle, but Signal'Stable is a Boolean signal whose value is true if there has been no event on the signal during the current cycle [1]. The fact that signals require scheduling, as discussed above, causes the attribute 'STABLE to be less efficient.

(5) A process with a static sensitivity list at the beginning (i.e. process (x, y, z)) is slightly more efficient than a process with a dynamic sensitivity list at the end (i.e. wait on x, y, z).

(6) The use of arrays in the form of bit_vectors was found to be more efficient than the use of an equivalent set of scalars in the form of individual bits.

(7) The use of resolved signals is much less efficient (greater than 25% difference) than an alternative method of creating an extra process to resolve signal assignment conflicts.

III. Efficiency Ratings

While there has been some work done in the determination of which VHDL modeling styles are most efficient, there has been little work done towards developing systems that analyze VHDL models and produce a numerical efficiency rating for the entire model [1,3]. Therefore an efficiency rating system has been developed to accurately measure the efficiency of a VHDL model in terms of simulation performance.

In this system the contents of a process in a VHDL model is represented by a control flow graph. Each node of the control flow graph represents a VHDL statement that is terminated by a semicolon. The arcs between nodes of the control flow graph represent the transfer of control from the statement represented by the source node to the statement represented by the destination node.

A system of penalties was developed for certain VHDL modeling styles based upon an exhaustive set of experiments some of which are discussed in Section II. For example, using objects of type bit_vector was found to be 20% slower than using equivalent objects of type integer with Vantage. Therefore the penalty for using any object of type bit_vector that could have been represented as an integer is 20.

Each node of the control flow graph of a model is initially given a numerical weight of 100. Then each node is evaluated in terms of any inefficient characteristics it may contain. The necessary penalties are deducted from the initial weight of 100. The overall static efficiency rating of the model is then calculated by taking the average of the final weights of each individual node.

An example of this system is given in Figures 1-4. The model to be evaluated is a simple RAM[3]. The efficient model of the RAM (RAME) is given in Figure 1. The inefficient model of the RAM (RAMi) is given in Figure 2 with the node numbers from the corresponding CFG (Fig. 3) placed at the left of each VHDL statement. In this inefficient model a signal is used to represent memory (MEM), but in this case a variable could have been used. With Vantage, the penalty for using signals instead of variables is 20, therefore each signal assignment statement that has signal MEM as its destination will be penalized 20 points (Fig. 3, nodes 1 - 16, 24). Another inefficient construct in the model is the use of the attribute 'STABLE with the signals INIT, RD, and WR. As discussed earlier, the attribute signal'EVENT is more efficient than not signal'STABLE therefore each node that contains not signal'STABLE is penalized (Vantage - 10 points). Figure 3 is a control flow graph of the inefficient model with the final weight of each node. The overall static efficiency rating based upon the node weights is 86.

The static rating discussed above does not take into account the fact that certain paths of the CFG will be traversed many more times than other paths. Therefore a more accurate dynamic rating system has been developed that considers the frequency of path traversals as a factor. To obtain this rating the test bench of the model is needed so that the coverage of each node can be retrieved using Synopsys. The node coverage is multiplied by the node weight to form a modified node weight. The dynamic rating is then calculated by taking the sum of all modified node weights and dividing by the sum of the node coverages. The dynamic rating is very useful because it allows the modeler to pinpoint inefficient nodes in the model that are being traversed a high percentage of time and are therefore hindering simulation performance.

Figure 4 is a chart demonstrating the accuracy of the rating schemes. Using test_bench1, the time to simulate the efficient RAM model (RAME) was 15.93 s. The time to simulate RAMi (inefficient) was 16.68 s. These times yield a 4% difference in simulation time. This means that the actual rating of RAMi is 96 (100 - 4). The dynamic rating given by the rating system is 93

```

use work.all, work.USER_TYPES.all;
entity RAME is
  generic (RDEL, DISDEL: TIME);
  port (DATA: inout MVL4_VECTOR(7 downto 0) := "ZZZZZZZZ";
        ADDR: in BIT_VECTOR(3 downto 0);
        RD, WR, NCS, INIT: in BIT);
end RAME;

architecture BEHAVIORAL of RAME is
  type MEMORY is array (0 to 15) of MVL4_VECTOR(7 downto 0);
begin
  RAM_E: process(NCS, RD, WR, INIT)
    variable MEM: MEMORY;
  begin
    if (INITEVENT and INIT = '1') then
      MEM(0) := "00000000";
      MEM(1) := "00000001";
      MEM(2) := "00000010";
      MEM(3) := "00000011";
      MEM(4) := "00000100";
      MEM(5) := "00000101";
      MEM(6) := "00000110";
      MEM(7) := "00000111";
      MEM(8) := "00001000";
      MEM(9) := "00001001";
      MEM(10) := "00001010";
      MEM(11) := "00001011";
      MEM(12) := "00001100";
      MEM(13) := "00001101";
      MEM(14) := "00001110";
      MEM(15) := "00001111";
    end if;
    if (NCS = '0') then
      if (RD'EVENT) then
        if (RD = '1') then
          DATA <= MEM(INTVAL(ADDR)) after RDEL;
        else
          DATA <= "ZZZZZZZZ" after DISDEL;
        end if;
      elsif (WR'EVENT and WR = '1') then
        MEM(INTVAL(ADDR)) := DATA;
      end if;
    else
      DATA <= "ZZZZZZZZ" after DISDEL;
    end if;
  end process;
end BEHAVIORAL;

```

Figure 1. Efficient VHDL Model of the RAM

```

use work.all, work.USER_TYPES.all;
entity RAMi is
  generic (RDEL, DISDEL: TIME);
  port (DATA: inout MVL4_VECTOR(7 downto 0) := "ZZZZZZZZ";
        ADDR: in BIT_VECTOR(3 downto 0);
        RD, WR, NCS, INIT: in BIT);
end RAMi;

architecture BEHAVIORAL of RAMi is
  type MEMORY is array (0 to 15) of MVL4_VECTOR(7 downto 0);
  signal MEM: MEMORY;
begin
  RAM_I: process(NCS, RD, WR, INIT)
  begin
    0   if (not INIT'STABLE and INIT = '1') then
    1     MEM(0) <= "00000000";
    2     MEM(1) <= "00000001";
    3     MEM(2) <= "00000010";
    4     MEM(3) <= "00000011";
    5     MEM(4) <= "00000100";
    6     MEM(5) <= "00000101";
    7     MEM(6) <= "00000110";
    8     MEM(7) <= "00000111";
    9     MEM(8) <= "00001000";
    10    MEM(9) <= "00001001";
    11    MEM(10) <= "00001010";
    12    MEM(11) <= "00001011";
    13    MEM(12) <= "00001100";
    14    MEM(13) <= "00001101";
    15    MEM(14) <= "00001110";
    16    MEM(15) <= "00001111";
    end if;
    17   if (NCS = '0') then
    18     if (not RD'STABLE) then
    19       if (RD = '1') then
    20         DATA <= MEM(INTVAL(ADDR)) after RDEL;
    21       else
    22         DATA <= "ZZZZZZZZ" after DISDEL;
    23       end if;
    24     elsif (not WR'STABLE and WR = '1') then
    25       MEM(INTVAL(ADDR)) <= DATA;
    26     end if;
    27   else
    28     DATA <= "ZZZZZZZZ" after DISDEL;
    29   end if;
    end process;
end BEHAVIORAL;

```

Figure 2. Inefficient VHDL Model of the RAM

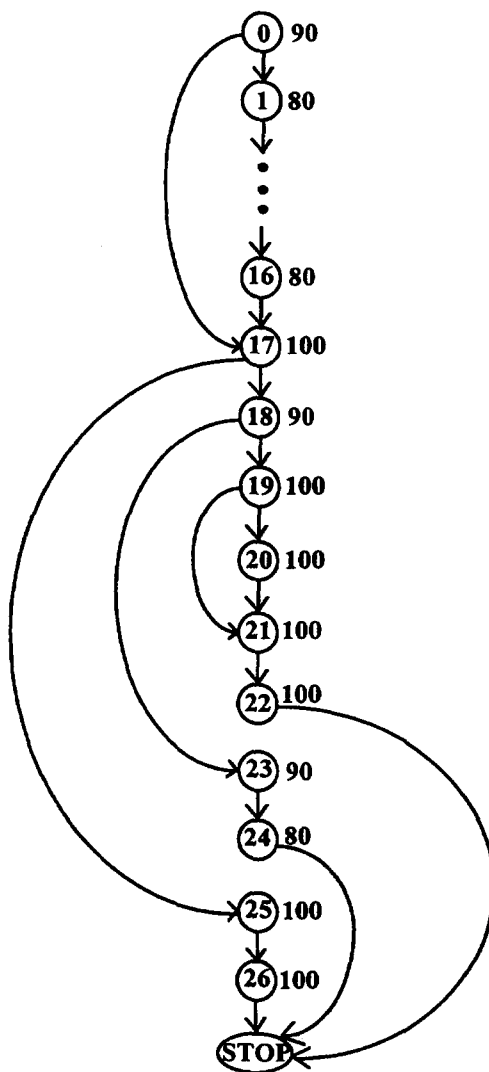


Figure 3. CFG of the RAM Model with Node Weights

RAM MODEL		Simulation Time	Actual Rating	Static Rating	Dynamic Rating
Test_bench1	RAMe	15.93 sec	100	100	100
	RAMi	16.68 sec	96	86	93
Test_bench2	RAMe	410 msec	100	100	100
	RAMi	470 msec	87	86	88

Figure 4. Table of Efficiency Predictions and Actual Results

and the static rating is 86. The dynamic rating is much closer to the actual rating because the node coverages are taken into account and test_bench1 causes the initialization of memory (nodes 1 - 16), which are penalized for inefficiency, to be traversed a low percentage of the time. In test_bench2, the dynamic and static ratings are equally good because nodes 1 - 16 are traversed at a higher percentage of time than in test_bench1.

The major tools that are used to develop this rating system are VTIP and Synopsys. VTIP (VHDL Tool Integration Platform) is used to evaluate the structure and characteristics of VHDL models. Synopsys is used to retrieve the node coverages of the VHDL model. The C programming language is also used in the rating tool development. Figure 4 is a

diagram of the structure of the efficiency rating system. The static and dynamic efficiency ratings can obviously be very useful in aiding designers in the development of highly efficient VHDL models.

IV. Conclusion

As stated earlier, the objective of this research is to aid the VHDL modeler in the development of highly efficient models. The methods of determining efficiency ratings described in this paper can certainly serve this purpose. Commercially there is a tremendous need for highly accurate efficiency rating systems. It appears that the rating systems discussed in this paper will help in the attainment of this goal.

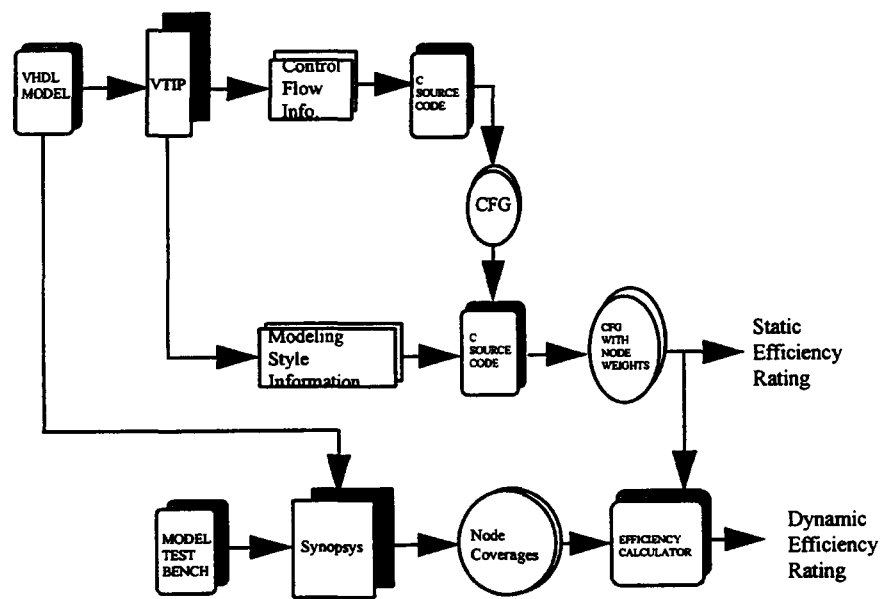


Figure 5. Block Diagram of Efficiency Rating System

V. References

1. Paulsen, B. and Levia, O., "Techniques for Writing High Performance and High Quality VHDL Models", *EURO VHDL 1992*.
2. Wicks, J. A., Jr. and Armstrong, James R., "VHDL Model Efficiency", Asian Pacific Conference on Hardware Description Languages, 1996.
2. Balboni, A., Mastretti, and Stefanoni, M., "Static Analysis for VHDL model Evaluation", *EURO VHDL 1994*.
3. Armstrong, James R., *Chip-Level Modeling With VHDL*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
4. Mazor, S. and Langstraat, P., *A Guide To VHDL*, Kluwer Academic Publishers, Boston, MA, 1992.
5. Armstrong, J. R., and Gray, F. G., *Structured Logic Design With VHDL*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
6. Bondy, J. A., and Murty, U. S. R., *Graph Theory With Applications*, Elsevier Science Publishing Co., Inc., New York, NY, 1976.