

A Power Simulator for VHDL Structural Descriptions

Srinivas Katkooori and Ranga Vemuri
ECE&CS Department
University of Cincinnati
Cincinnati, OHIO, OH 45221-0030
ranga@thor.ece.uc.edu

Abstract

In this work, VASP – Power Simulator is presented which estimates the total power consumed by a design represented at RT Level (Architectural) in VHDL. Power simulation consists of the following tasks: (a) characterization of the module library, (b) simulation of the VHDL structural description. The module library consists of register level modules such as adders, registers, multiplexors. In the VHDL Library, for each module two architectures exists namely, one-bit and n-bit. The one-bit architecture has capacitance measuring code embedded in it. The n-bit architecture is an instantiation of 1-bit architecture using `generate` statement. Thus the module library characterization is very fast as it is done only for 1-bit architecture. For a given VHDL structural description, it is simulated using a VHDL simulator and a power profile is obtained.

Section 1. Introduction

Power consumption is becoming one of the key constraints that the designer has to address as early as possible in the design process, as the complexity of VLSI systems is increasing rapidly [1]. To estimate power consumed in a design, a *power simulator* would be of great help to the designer. It estimates power by simulating the hardware description using user specified input vectors. In this work, we present a power simulator known as VASP (VHDL Architectural Simulation for Power).

Power Estimation techniques exist in literature [2]. A general technique to estimate power at circuit level is by simulation : perform a simulation of the design and monitor the power supply current waveform [3]. Although, this technique gives an accurate power consumption, it is very time-consuming. To improve computational efficiency, other simulation based techniques were proposed using various kinds of timing, switch-level, and logic simulation [4, 5] Hence, efforts are on way to estimate power at a higher level of abstraction, so as to gain in time and at the same time not lose the accuracy. The above simulation technique could be extended to a RT level of abstraction. This work exploits the hierarchical nature of a VHDL structural description to estimate the power.

The power consumed by a digital system consists of three components namely (a) switching (b) short-circuit and (c) leakage components. The components (b) and (c) are highly dependent on the actual physical implementation of the system which is not known at the architectural level. Hence, VASP estimates only the first component of the power. For certain technologies like CMOS, the switching component is dominant and hence VASP gives accurate results for such a technology. In this work, CMOS technology is assumed. The power consumed by a system for a set of input vectors is given by the equation :

$$P_{consumed} = \frac{1}{2} * C_{switched} * V_{supply}^2$$

where $C_{switched}$ is the total capacitance switched in the system to pro-

cess the entire set of input vectors and V_{supply} is the supply voltage. Assuming fixed supply voltage, it is obvious that switched capacitance is a direct measure of the power consumed. Thus, in this work we use power and switched capacitance synonymously.

Typically, a VHDL structural description of a system contains various instantiation of components from a module library and their interconnection by signals. The module library consists of modules such as adders, registers, multiplexors etc. Each of the module has generic attributes like width, delay, capacitance etc., to which specific values are provided in each instantiation. For each kind of module, two architectures reside in the library. The first architecture (one-bit architecture) is a one-bit implementation of the module and is a purely behavioral description. The second one (n-bit architecture) is an n-bit structural implementation using `generate` statement instantiating modules of the one-bit architecture.

Power simulation consists of the following tasks : (a) characterization of the module library, (b) simulation of the VHDL structural description. The module library is characterized as follows: for each module and for its one-bit architecture, the corresponding layout implementation in the target technology (in this work, CMOS technology), is simulated to obtain the actual switched capacitance values for all possible input transitions. This data is embedded in the behavioral description of the one-bit architecture as a look-up table. Hence, during the simulation, whenever there is an `event` in one or more inputs of the module, the switched capacitance value for the corresponding transition, is accumulated. As the n-bit architecture is composed of modules of one-bit architecture, the power for the module is the sum of that of n number of modules of one-bit architecture. The VHDL structural description which consists of modules of n-bit architecture is simulated using user specified set of input vectors. At the end of simulation, a power profile of the various modules is obtained.

Section 2. VASP - VHDL Architectural Simulation for Power

The VASP Environment is as shown in Figure 1. As shown in the figure, the inputs to the power simulator are :

1. An RT Level design in VHDL

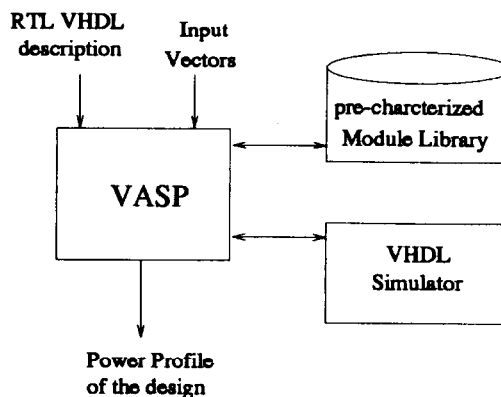


Figure 1: VASP Environment

2. A Set of Input Vectors
3. A pre-characterized Module Library, and

The RTL description is a VHDL structural description containing instantiations of components from the given module library. The module library contains execution units such as adders, multipliers etc., storage units such as registers, latches etc., and interconnection units such as multiplexors, buses etc. Each component is written in behavioral VHDL. The module library is discussed later in more detail. A portion of a RTL description is as shown below.

The user given set of input vectors is typical of the inputs used after synthesizing the RTL description to the gate level or circuit level of abstraction. The user is interested in the total capacitance switched by the design in processing the given set of vectors.

The module library is pre characterized for the average capacitance switched by the module of 1-bit architecture. The characterization procedure is discussed in detail in the next section.

```

Library Lib;
use Lib.all;

entity stack_dp is
  port (push : IN Bit_vector (0 downto 0);
        pop  : IN Bit_vector (0 downto 0);
        data_in : IN Bit_vector (3 downto 0);
        stack_full : OUT Bit_vector (0 downto 0);
        stack_empty : OUT Bit_vector (0 downto 0);
        :
        : );
end stack_dp;

architecture Structure of stack_dp is

  -- component declarations

  component C_Comparator
    generic (
      width : integer;
      delay : time);
    port(
      input1 : Bit_vector ((width - 1) downto 0);
      input2 : Bit_vector ((width - 1) downto 0);
      output : out Bit_vector (2 downto 0));
  end component;
  for all : C_Comparator use entity Lib.C_Comparator(Behavior);
  :

  -- signals for interconnecting the instances

  signal nmcir1_output : Bit_vector (2 downto 0);
  :

  signal nmcir33_output : Bit_vector (2 downto 0);

  signal mcir1_output : Bit_vector (0 downto 0);
  :

  signal mcir25_output : Bit_vector (0 downto 0);

begin -- the body of the architecture

  nmcir1 : C_Signal
    generic map (4, 7 ns)
    port map ( ....);
  :

  nmcir10 : Shift_Reg
    generic map (4, 7 ns)

```

```

        port map ( ....);
        :
nmcir33 : C_Comparator
        generic map (1, 7 ns)
        port map ( ....);
        :
mcir1 : C_Multiplexer
        generic map (1, 2, 1, 2 ns)
        port map ( ....);
        :
mcir25 : C_Multiplexer
        generic map (1, 2, 1, 2 ns)
        port map ( ....);

        stack_full(0 downto 0) <= nmcir4_output(0 downto 0);
        stack_empty(0 downto 0) <= nmcir5_output(0 downto 0);
        :
end Structure;

```

Section 3. Module Library

The module library contains register-level modules such as registers, adders and multiplexors. For each component there exists two architectures namely 1-bit and n-bit architectures. The 1-bit architecture is a module with input size equal to one. For example, the 1-bit architecture of an adder module contains the full-adder description as shown in figure 2 and the n-bit architecture is as shown in figure 3.

The 1-bit architecture, besides having the functionality code, has VHDL code to accumulate the capacitance. Intuitively, when there is an **eventon** one of more inputs the module does some computation and thus switches some amount of capacitance. Equivalently, at the behavioral level, the module accumulates an average capacitance value which is determined as follows. The n-bit architecture of a module is structural description which instantiates the 1-bit architecture module in a **generate** statement. For example, look at the n-bit architecture of the adder module.

We define the *average intrinsic switching capacitance* (ISC) of a 1-bit module as the average capacitance that is expected to switch when an input event (change of logic values on the input lines) takes place. ISC of a 1-bit module instance is determined by extracting a switch level model of from its layout, simulating the switching level module using a very long stream of randomly generated input patterns and monitoring the capacitance switched per pattern, until the convergence occurs as discussed below. The capacitance measurements have been done by IRSIM-CAP [6], which is a modified version of IRSIM switch level simulator for better capacitance measurements.

Let S_k be the total capacitance charged after applying k random input patterns without reinitialization between successive patterns. $Z_k = \frac{S_k}{k}$ denotes the average capacitance per input pattern after applying k patterns. $\delta_k = \frac{|Z_k - Z_{k-1}|}{Z_{k-1}}$ denotes variation in the average capacitance between the $k-1$ th and k th patterns. We continue to apply random input patterns until δ_k remains less than 0.001 over 1000 consecutive input pattern applications. At this point we say that the average switching capacitance estimation converged and accept the value of Z_k after the last input pattern is applied. This value is the ISC of that instance of the module.

Figure 4 show ISC characteristics of a library module. Table 1 shows the ISC characteristics for some of the library modules.

The above approach is suitable for modules which are purely combinational in nature. For sequential cir-

	Module	ISC (pF)
1.	Adder	0.45
2.	Subtractor	0.97
3.	Comparator (>)	0.44
5.	Multiplier	2.27
6.	Multiplexor	2-inputs: 0.45 4-inputs: 1.41 6-inputs: 2.46 8-inputs: 3.29
7.	Register	3.77
8.	Signal Register (Register + Glue Logic)	10.90
9.	AND	0.17
10.	OR	0.18
11.	NOT	0.04
12.	NAND	0.06

Table 1: ISC Data for Some Parameterized Library Modules

uits, not only inputs but also the current state of the circuit is important. Moreover, we cannot accumulate the average capacitance for an **eventon** inputs. For example, the library consists of a latch. A layout instance of the latch is simulated from all possible transitions and the amount of capacitance switched is recored in a table. A portion of the look up table for a negative-edge triggered latch is as shown in the table 2. Each row in the table shows a transition from a set of inputs (represented by a 5-tuple (Input, Enable, Clear, Phase1, Output)) to another set of inputs. The units for the capacitance is pico farad. Note the wide variation for the capacitance switched for different transitions. Figure 5 shows the VHDL code for a D latch.

Section 4. Power Simulation

The procedure for the power simulation is as follows: the RTL description is modified such that all the n-bit instances of a module in the description are bound to the corresponding n-bit architecture of the module. A test bench is generated automatically to simulate the RTL description using the user specified set of input vectors. The VHDL simulator is used to simulate the modified design and a power profile of the design is generated. At the end of the simulation, each mod-

```

entity C_adder is
  generic (delay : TIME := 0 ns; isc : REAL := 0.0 );
  port (A,B,C_in : in Bit;
        Sum,C_out : out Bit);
end C_adder;

architecture 1_bit of C_adder is
begin
  P0 : process(A,B,C_in)
    -- cap accumulator
    variable cap : real := 0.0;
    begin

      Sum <= A xor (B xor C_in );
      C_out <= (A AND B) + (B AND C_in) + (C_in AND A);

      -- code to accumulate capacitance
      if(A'EVENT OR B'EVENT OR C_in'EVENT) then
        cap := cap + isc ;
      end if;

    end process P0;
end 1_bit;

```

Figure 2: 1-bit architecture of an Adder Module

ule prints out the switched capacitance accumulated is printed onto the screen or written into a file.

Transition	Capacitance switched (pF)
(0,0,1,0,0) → (0,1,0,0,0)	2.12
(0,1,0,0,0) → (0,0,1,0,1)	0.15
(0,0,1,0,1) → (0,1,0,0,1)	2.97
(0,1,0,0,1) → (0,0,1,1,0)	2.01
(0,0,1,1,0) → (0,1,0,1,0)	2.42
(0,1,0,1,0) → (1,0,1,0,0)	1.22
(1,0,1,0,0) → (1,1,0,0,0)	1.12
(1,1,0,0,0) → (1,0,1,0,1)	0.15
(1,0,1,0,1) → (1,1,0,0,1)	3.26
(1,1,0,0,1) → (1,0,1,1,0)	2.23
(1,0,1,1,0) → (1,1,0,1,0)	3.01
(1,1,0,1,0) → (1,1,0,1,0)	1.24

Table 2: A portion of the capacitance look up table of a negative-edge triggered flipflop

Section 5. Conclusions

In this work, we presented a Power Simulator for VHDL Structural Descriptions, which exploits the hierarchical nature of VHDL descriptions.

Acknowledgements

This work is done at the University of Cincinnati and is supported in part by the Solid State Electronics Directorate of the Wright Laboratory of the US Air Force under contract number F33615-91-C-1811 and by the Advanced Research Projects Agency under order no. 7056 monitored by the Federal Bureau of In-

```

library Lib;

entity C_adder is
  generic ( width : integer; delay : time);
  port (Input1, Input2 : in Bit_Vector((width - 1) downto 0);
        Output : out Bit_Vector(width downto 0));
end C_adder;

architecture n_bit of C_adder is

  component C_adder_1
    generic (delay : TIME := 0 ns);
    port (A, B, C_in : in Bit;
          Sum, C_out : out Bit);
  end component;

  for all: C_Adder_1 use entity Lib.C_Adder(1_bit);

  signal C_out : bit_vector((width-2)downto 0);
  signal zero_sig : bit := '0';

  begin

  C_addern : for i in 0 to (width-1) generate
    ls_bit : if i = 0 generate
      ls_cell: C_Adder_1
        generic map(delay)
        port map( Input1(0),Input2(0),zero_sig,Output(0),C_out(0));
    end generate ls_bit;

    middle_bit : if i > 0 and i < (width-1) generate
      middle_cell: C_Adder_1
        generic map(delay)
        port map( Input1(i),Input2(i),C_out(i-1),Output(i),C_out(i));
    end generate middle_bit;

    ms_bit : if i = (width-1) generate
      ms_cell: C_Adder_1
        generic map(delay)
        port map(Input1(width-1),Input2(width-1),
                  C_out(width-2),Output(width-1),Output(width));
    end generate ms_bit;

  end generate C_addern;

  end n_bit;

```

Figure 3: n-bit architecture of an Adder Module

```

entity D_Latch is
  generic (delay : TIME := 0 ns; cap := REAL := 0.0);
  port (Input : in Bit;
        Enable, Clear, Phase1 : in Bit;
        Output : out Bit);
end D_Latch;

architecture 1_bit of D_Latch is
begin
  latch_process : process(Phase1,Clear,Input,Enable)
    variable out_var : Bit := 0;
    variable cap : REAL := 0.0;
  begin
    out_var := '0';

    if (Clear = '1' and not Clear'STABLE) then
      Output <= out_var after delay;
    end if;

    if (Phase1 = '0' and not Phase1'STABLE and Enable = '1') then
      Output <= Input;
    end if;

    -- code to accumulate capacitance

    if(Input = '0' AND
       (Enable = '1' AND Enable'EVENT) AND
       (Clear = 0) AND
       (Phase = '1' AND Phase'EVENT) AND
       (Output = '0')) then
      cap := cap + 1.22 ;
    end if

      :
      :
      :

    end process latch_process;
end 1_bit;

```

Figure 5: 1-bit architecture of a D-latch

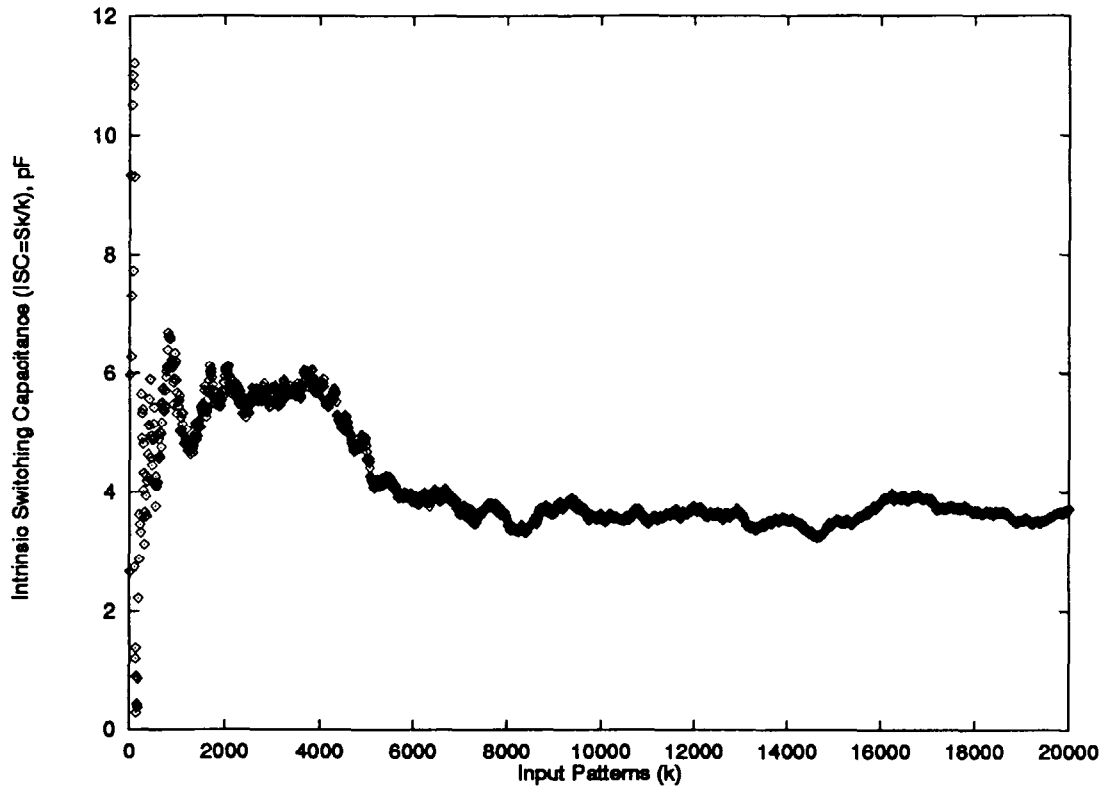


Figure 4: ISC Characteristic of a 1-bit Register

vestigation under contract no. J-FBI-89-094.

References

- [1] T. Bell, "Incredible Shrinking Computer", *IEEE Spectrum*, Vol. 28, No. 5, pp. 37-43, May, 1991.
- [2] F.N.Najm, "A Survey of Power Estimation Techniques in VLSI circuits (Invited Paper)" *IEEE Transactions VLSI Systems*, vol. 2, no. 4, pp. 446-455, January 1995.
- [3] S.M. Kang, "Accurate simulation of power dissipation in VLSI circuits," *IEEE Journal of Solid-State Circuits*, vol SC-21, no. 5, pp 889-891, Oct. 1986.
- [4] R. Tjarnstrom, "Power dissipation estimate by switch level simulation," *IEEE International Symposium on Circuits and Systems*, pp. 881-884, may 1989.
- [5] T. H. Krodel, "PowerPlay - fast dynamic power estimation based on logic simulation," *IEEE International Conference on Computer Design*, pp. 96-100, October 1991.
- [6] P. Landman, "IRSIM-CAP : A Version of the Event Driven Logic Level Simulator IRSIM-9.0 With Extensions for Improved Capacitance Measurements," Univ. of California, Berkeley, 1993.