PELib: A VHDL Library for Power Dissipation Estimation of CMOS Digital Circuits

Verginia-Iulia-Maria CHEREJA, Adriana-Ioana POTARNICHE, Sergiu-Alex RANGA,   
Botond Sandor KIREI, Marina Dana TOPA

*Technical University of Cluj Napoca, 410011, Romania*

*botond.kirei@bel.utcluj.ro*

*[[1]](#footnote-1)Abstract*— In this paper, the PElib VHDL library is conceived (and described), which can monitor the switching activity of several digital circuits (logic gates, encoders, multiplexers, counters) and to generate pin toggle information to estimate the power consumption of circuits described at the gate level. The advantage of this library is that of obtaining a power dissipation estimate using a logic simulator, rather than a specialized power simulation and estimation software. To validate the library two test circuits were implemented with standard logic gates and their power dissipation was measured. The first circuit is a 5 stage ring oscillator implemented with logic gates from the CD4000 series and the power estimation error is XXX%. The second circuit is a finite state machine implemented with circuits from the 74HC series, and the estimated power dissipation error is XXX percent. The library can be extended to estimate circuits implemented with the use of CMOS standard cell libraries. Also, TTL logic gates may be included into the PELib library.

*Index Terms*—VHDL, power dissipation estimation, switching activity, gate level description.

# Introduction

Power estimation in early design stages of VLSI circuits is a must for the low-power applications of current day: mobiles, handheld devices and battery powered gadgets []. Also, power estimation is carried out in different design levels: system, transaction, RTL, gate and transistor level [].

The power density of integrated circuits is increasing with the reduction of transistor size [1]. This hardens the thermal dissipation requirements of the packaging, but also inserts new requirements for digital designers, conducing to the appearance of new digital circuit design paradigms at many levels: dynamic adaptive control of the power supply and the threshold voltage at transistor level [4]; clock gating, partitioning and precomputation at the circuit level [5]; power-down and stand-by operating states at the system level. In this environment, the estimation of power dissipation in early design stages is becoming important. There are many commercial tools to estimate power consumption at different stages in the design flow [PowerSim, XPA]. Often, these tools are not available for a designer, but good, royalty free HDL simulation tools for both Verilog and VHDL (ModelSim, Active HDL, Xsim, Ghdl, Icarus) are. Thus, the idea of using a simulation tool for power estimation is born.

A possible solution for power estimation in CMOS is the monitoring of switching activity in the HDL code. Such a power estimation method was implemented in SystemC [6]. Ref. [6] presents a dynamic power estimation approach for a circuit described at the gate level. The shortcoming of the method is that it neglects the static power dissipation. A VHDL power dissipation estimation method is presented in [7]. This approach considers the input slope, output capacitive loading () and glitches in the modelling of power/energy consumption. Of course, the above solutions have their limitations, for example in the VHDL design phase, the layout of the circuit is unknown, thus the parasitic capacitance of the connecting wires are unknown, the best one can do is to estimate them, too.

In this paper, a static and dynamic power estimation methodology in VHDL is given. The static power of a CMOS gate is estimated by the product of the subthreshold leakage current and the power supply voltage, while the dynamic power is estimated by the accounting of parasitic capacitor charge and discharge in the CMOS gate. To carry out these computations a VHDL library was created, which comprise of data types, components and functions used in the power/energy estimation. The library also contains descriptions of logic gates from several CMOS logic family (CD4000, 74HC, 74AC to name a few). These gates can be used in the structural description of the desired circuit. As the shortcoming of the method we mention that a gate level description does not give any information about the layout of the circuit, thus power consumption due to wire parasitic capacitance are neglected. No timing is assumed in the gate descriptions, thus glitching power dissipation is also neglected. Still, the power estimation in this early design phase can be an aid for the designer, to find the power-hungry circuits/subsystems. Furthermore, power saving methods, such as clock gating, functional partitioning and precomputation may be verified, thus quantitative estimations may be obtained.

This paper is organized as follows: in Section II. the theoretical bases of power dissipation are given, briefly presenting the main sources of power dissipation in CMOS digital circuits. The Section III. presents the VHDL implementation of the static and dynamic power dissipation estimation. Section IV, compares the power estimation results obtained in ModelSim with measured power consumption of two digital circuits: a ring oscillator and a finite-state-machine. Finally, conclusions and further work are drawn.

# Power Dissipation in CMOS Digital circuits

The power dissipation in CMOS digital circuits are discussed in many scholarly work [8], and it is briefly reviewed in this paper, too. The total power dissipation in digital circuits have two components. The first one is a static component, which is consumed whenever the circuits are turned on and it is caused by the leakage current (also refereed as quiescent current in logic gates datasheet). The second is the dynamic power consumption, which is cause by switching activity in the circuit. Dynamic power consumption is due to capacitive switching (the charge and discharge of parasitic capacitors in the circuit), short circuit power (for a short time both transistors may conduct, thus a short is proceeded between the supply and ground rails) and glitch power (occurring when the inputs of a gate is not changing in the same moment).

A particular problem of CMOS gate power estimation is that power dissipation depends on switching activity in the circuit (depends on the application), contrary to the transistor-transistor-logic gates power consumption, which is more or less constant (indifferent to application). Moreover, static power dissipation is getting more accentuated in the power budget with the reduction of CMOS transistor feature size, thus not only dynamic power dissipation reduction but also static power dissipation reduction is a necessity. Traditionally, static power dissipation was neglected (less than 10% of the total power use in the case on TTL and CMOS circuits) and dynamic power dissipation was considered the upholder for power dissipation

## Static power consumption

The static power computation for CMOS logic is

 

where static power consumption *Pstatic* is the product of the device leakage current *ICC* (this parameter is listed in CMOS gate datasheets as the quiescent current of the device) and the supply voltage *VCC*.

## Dynamic power consumption

The main component of the dynamic power consumption is due to capacitive switching (the energy consumed over time for the charge and discharge of parasitic capacitances in the CMOS transistors). The energy accumulated in a capacitance C is:



where *C* is the capacity and *V* is the voltage it is charged to.

The CMOS logic gate power models are comprised of three parasitic capacitances: *Cin* the input capacitance of a pin, *Cpd* power dissipation capacity and *Cload* the parasitic capacity of the connecting wires. Let us consider the most elementary CMOS gate, the inverter, and its parasitic capacitances (Fig. 1). When a transition on the input signal occur, the *Cin* is charged/discharged, *Cload* is discharged/charged and *Cpd*, an equivalent capacitance of parasitic ones of CMOS transistors, that is charged/discharged. Thus, the energy consumed for a single transition *Etransition* is:



where *Vsupply* is the supply voltage.



1. Power dissipation model of a CMOS inverter

The total energy consumption is the sum of energies consumed at each transition. As the gates are used in sequential circuits clocked at a given frequency it is costumery to express the power in terms of energy / time, in this case ending in energy \* frequency:

 

where *f* is the frequency of the input signal and *Nswitching* is the activity factor. In the case of an inverter *Nswitching* = 1, but the 2 or more input logic gates has s switching activity less than 1. The actual challenge in power estimation is to find the switching activity of each gate.

## Power dissipation measurement

In Fig. 1 also the measurement of the dissipated power is depicted. The current flowing to the inverter is measured with the use of a current sensing resistor *Rsense* (high side shunt resistor) and a voltmeter (V). The dissipated power is computed as:

 

where *VRsense* is the voltage across *Rsense*. The power dissipation of *Rsense* is neglected.

# VHDL Library for Power Estimation

## Library structure

The proposed VHDL library for power estimation is composed of three packages: PElib, PEGates and Nbits. The PElib package contains the data types and utility functions used for power estimation. PEGates comprises of logic gates with consumption monitoring. Nbits package contains configurable length components such as registers, counters etc, that are build using the components form PEGates package. Also the components in Nbits has the consumption monitoring activated.

## PElib package

As the nature of the static and dynamic power dissipation is different, from the programming point of view, it makes sense, the declare a new record data type for the consumption, with one field for the static and another field for dynamic consumption. In the VHDL implementation the new data type is called consumption\_type. The code sequence in Fig. 2 illustrates the definition.

**type** consumption\_type **is record**

dynamic : **real**; *-- meant to represent dynamic consumption*

static : **real**; *-- meant to represent static consumption*

**end record** consumption\_type;

1. New data type definition for consumption

An important utility function defined for the new type is the addition of two *consumption\_type* values. This is realized with the overloading of the “+” operator and the corresponding VHDL sequence is depicted in Fig. 3.

**function** "+" (a, b : consumption\_type) **return** consumption\_type **is**

**variable** sum : consumption\_type;

**begin**

sum.dynamic := a.dynamic + b.dynamic;

sum.static := a.static + b.static;

**return** sum;

**end** **function**;

1. Overloading of “+” operator

**library** IEEE;

**use** IEEE.std\_logic\_1164.all;

**entity** activity\_monitor **is**

**port** ( signal\_in : **in** **std\_logic**;

activity : **out** **natural** := 0);

**end** activity\_monitor;

**architecture** behavioral **of** activity\_monitor **is**

**signal** nr\_trans: **natural** := 0;

**begin**

transition\_counter : **process**(signal\_in)

**begin**

nr\_trans <= nr\_trans + 1;

**end** process;

activity <= nr\_trans;

**end** behavioral;

1. Activity monitoring (counting the number of transitions) of signals.

In Fig. 4 the *activity\_monitor* component implementation is depicted. The *transition\_counter* process is invited whenever the value of *signal\_in* is changing. When the process is executed a counter (*nr\_trans*) is incremented. This component should be connected to each node in the circuit. Whenever a node in the circuit is changing its logic value, a process is invited and a counter is incremented. Thus, each charge and discharge, also glitches, will be accounted for in the circuit. The component returns just the number of transitions, the related energy or power is not computed in this module.

The PElib package also contains the *consumption\_monitor* component, which is instantiated in the modules where the user wants to monitor the power consumption. The component has two inputs: *sin* - to these ports are connected the input signals of the module (these signals will charge/discharge the *Cin* input capacitance); *sout* – to this port shall be connected the output signals of the module (the activity on these signals charges/discharges the load capacitance *Cload*). This component makes use of the *activity\_monitor* component, to monitor the activity on the input and output nodes. As the number of transitions on a node and the parasitic capacities connected to a node are known, the dynamic energy due to the charge and discharge of the capacities can be computed as:

 

where *N* is the number of input ports *IN* is the number of logic level transitions (equivalent to the number of capacitor charge/discharge) on the *Nth* input, *M* is the number of output signals, and *OM* is the number of logic level transitions on the *Mth* output. Not that, the numbers *IN and OM*, are obtained from *activity\_monitor* components.

Dynamic energy estimation is carried out as given in eq. (6) in *consumption\_monitor* component. Also, the static power is computed as indicated by Eq. (1) is computed in this component. The output of *consumption\_monitor* is the compound value of the static power dissipation and the dynamic energy consumption. Note that the dynamic consumption is expressed in energy, while the static power consumption is expressed in power. This also justifies the definition of the *consumption\_type* user defined type.

The PElib includes a component named *power\_estimator*. This component is used to derivate the dynamic energy over time and to add the result to the static power consumption, thus resulting in the total power dissipation estimate:

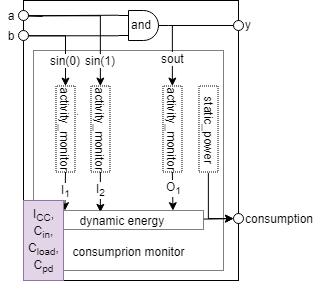
  

where *ΔEdynamic* is the variation of the dynamic energy consumption during the period *ΔT*.

PElib also contains constant definitions for the parameters such as leakage current *ICC*, input capacitance *Cin*, power dissipation capacitance *Cpd* for logic gates (inverter, tristate buffer, and gate with 2/3/4 input, or gate with 2/3/4 input, xor gate, 2:1 multiplexer, 4:1 multiplexer, 4 bit decade counter, and so on) from several CMOS logic families (4000 series, 74HC series, etc.). Once the user selects the desired logic family. This data can be extracted from the datasheets of the logic gates.

## PEGates package

This package contains primitives (logic gates, multiplexers, counters, etc.) with the *consumption\_monitor* component instantiated. Let us present the conceptual module of an *and* gate in Fig. 5, to illustrate the use of the package.



1. Consumption monitoring in an and gate

The module contains an and gate with two inputs and one output. Beside the gate a *consumption\_monitor* component is instantiated. This is configured to have inside 3 *activity\_monitor* components, which are connected to each node of the circuit, in this case the inouts of the and gate. Any transition on the nodes are accounted for in these components. The output of the *activity\_monitor* components is an integer number, equal to the transitions that occurred. The number of transitions are used to compute the dynamic energy consumed in each node and their sum is also computed. The static power is also computed in the *consumption\_monitor* component. The consumption is returned on a port named the same and the data type of the port is *consumption\_type*.

In Fig. 6 the VHDL description of the conceptual module in Fig. 5 is given. The entity of the and gate is parametrized, to pass the parasitic capacitance and leakage current values (these are considered parameters, as they are not excepted to change in time). The architecture contains two parts: the first describes the functionality of the circuit, the second is used to estimate the consumption. The estimation is carried in the *consumption\_monitor* component. This is instantiated with the next parameters: the parasitic capacitance and leakage current values are passed further to the *consumption\_monitor* component; as the inverter has one input and one output signal, parameters *N* and *M* (which are the number of input, respectively output, signals) are tied to 2, respectively 1.

**library** IEEE;

**use** IEEE.std\_logic\_1164.all;

**use** PElib.all;

**entity** and\_gate **is**

**generic** (delay : **time** :=1 ns;

Cpd, Cin, Cload : **real** := 20.0e-12; *--parasitic capacities*

Icc : **real** := 2.0e-6); *-- quiescent current at room temperature*

**port** ( a, b : **in** std\_logic;

y : **out** std\_logic;

consumption: out **consumption\_type** := (0.0,0.0));

**end** and\_gate;

**architecture** primitive **of** and\_gate **is**

**signal** internal : std\_logic;

**begin**

internal <= a and b after delay; *-- behavior*

y<=internal;

*-- consumption monitoring*

*-- pragma synthesis\_off*

cm\_i : consumption\_monitor generic map ( N=>2, M=>1, Cpd =>Cpd, Cin => Cin, Cload => Cload, Icc=>Icc) port map (sin(0) => a, sin(1) => b, sout(0) => internal, consumption => consumption);

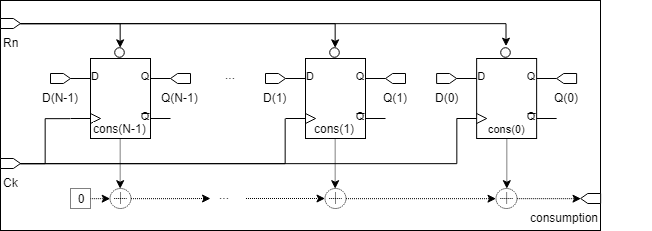
*-- pragma synthesis\_on*

**end** primitive;

1. The VHDL description of an and gate with consumption monitoring

## Nbits package

In this package, some basic circuits, such as 1 bit full adder, SR type latch, D type flip-flop, and configurable zise/width circuits, such as N bit adder, N bit register, N bit ripple counter, etc. are built using the gates form the PEGates package. The consumption of a circuit is obtained by summing all the consumptions of internal gates. In Fig. 7 the conceptual design of an N bit register with power monitoring is considered. A D type flip-flop is described using the PEgates package. This flip flop has a consumption port, as its consumption is the sum of the contained logic gate consumptions. The D flip-flop is instantiated N times in the circuit. The total consumption of the register is obtained by adding up all the DFF consumptions.



1. The conceptual module of an N bit regiter with consumption monitoring.

**library** IEEE;

**use** IEEE.std\_logic\_1164.all;

**use** xil\_defaultlib.PElib.all;

**use** xil\_defaultlib.PEGates.all;

**use** xil\_defaultlib.Nbits.all;

**entity** reg\_Nbits **is**

**generic** ( delay: time := 0 ns;

N: natural := 8);

**port** ( D : **in** std\_logic\_vector (N-1 downto 0);

Ck : **in** std\_logic;

Rn : **in** std\_logic;

Q : **out** std\_logic\_vector (N-1 downto 0);

Qn : **out** std\_logic\_vector (N-1 downto 0);

consumption : **out** consumption\_type := (0.0,0.0));

**end** reg\_Nbits;

**architecture** Structural **of** reg\_Nbits **is**

**type** cons\_t is **array** (**integer** **range** <> ) **of** consumption\_type;

**signal** cons : cons\_t(0 to N-1) := (**others** => (0.0,0.0));

**signal** sum : sum\_t (-1 to N-1) := (**others** => (0.0,0.0));

**begin**

registers: **for** i **in** 0 to N-1 **generate**

dffi : dff **port** **map** (D => D(i), Ck => Ck, Rn => Rn, Q => Q(i),  
Qn => open, consumption => cons(i));

**end** generate registers;

*-- pragma synthesis\_off*

sum(-1) <= (0.0, 0.0);

sum\_up\_energy : **for** I **in** 0 to N - 1 **generate**

sum\_i: sum(I) <= sum(I-1) + cons(I);

**end** **generate** sum\_up\_energy;

consumption <= sum(N - 1);

-- *pragma synthesis\_on*

**end** Structural;

1. The VHDL model of am N bit regiter with consumption monitoring

In Fig. 8, the VHDL description of an N bit adder is given. In the architecture a generate statement is used for repeated instantiations of D type flip flops, which are connected correspondingly. Each flip-flop has a consumption output, which are summed up, thus resulting in the total consumption of the circuit. If only the N bit register power dissipation was to be measured, then the *power\_estimation* component could be instantiated in the module, in order to obtain the total power dissipation, by adding the static and dynamic consumptions.

# Use Cases of Power Estimation Library

To verify the use of the library, we proposed to implement simple circuits and measure their power consumption, while the library is used to model the same circuit and to estimate the its power dissipation. The first circuit, where the use of this VHDL library is demonstrated is a 5-stage ring oscillator.

## 5-stage ring oscillator

## Finite-state-machine

The verification of the power estimation library was carried out by comparing the power dissipation of a real circuit with the estimated power delivered by the library. The test circuit used for this purpose was a finite-state-machine implemented with logic gates of the 74HCT series. We chose this circuit as it contains both combinational and sequential components. The state diagram of the FSM, the circuit design process for a 74xx163 type counter and the gate level schematic is done in Appendix A. The power measurement was carried out as indicated in Eq. (5).

In Table I. the measured and

1. Measured vs. Estimated Power of the FSM

| Control signal combination (a,b) | Power (mW) | | |
| --- | --- | --- | --- |
| Measured | Estimimated | Error |
| a=1, b=1 | More table copya |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Further work

Extending to CMOS standard logic cells

Including TTL libraries

Increase the quaraci, by including time delay

Power supply scaling

Verification of the libraru agains banchmarks

Further work: the gate descriptions can be extended with a power supply voltage port; thus voltage scaling approaches can be verified with the VHDL library; the library contains only device form logic gates families, but it can be extended for standard logic cells, only the extraction of some parameters are required (input capacitance, power dissipation capacitance, subthreshold leakage current).

# Conclusions

.

##### Appendix A.

In Fig. 6, is the state diagram of the FSM implemented with 74HCT163 (counter), 74HCT253 (multiplexer 4:1) and 74HCT04 (inverting gate) is given.



1. The state diagramm of the test FSM

This FSM goes through six states, so the number of states can be represented on 3 bits, just as the diagram suggests. These 3 bits will be taken from the QCQBQA outputs of the counter. To aid the FSM synthesis, the actions performed by counter is listed (reset, count up, load, hold, etc) in a table (see Table I). The actions are determined by the control signals of the counter: load (LD) and hold (P&T), etc. In Table II, the truth table of the FSM is listed. Based on this table the logic functions for control sigbal LD, P&T, ABCD are derived, ultimately the logic functions are implemented using 4:1 multiplexers and inverters (see Fig. 7).

1. Operation of a 74XX163 Counter

|  |  |  |
| --- | --- | --- |
| Control signals | | Action |
| Ld | P&T |
| 0 | x | load |
| 1 | 0 | maintain |
| 1 | 1 | count |
| Values for combined actions | | |
| var-m | 0 | load+maintain |
| 1 | var-c | count+maintain |
| var-c | 1 | coad+count |
| 0 | x | load+load (paralel load) |

1. The Truth Table of Finite State Machine

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Current state QCQBQA | Operation | Control signals | | | | | |
| Cl | Ld | P,T | C | B | A |
| 000 | count | 1 | 1 | 1 | x | x | x |
| 001 | count | 1 | 1 | 1 | x | x | x |
| 010 | load+keep | 1 | b | 0 | 1 | 0 | 1 |
| 011 | x | 1 | x | x | x | x | x |
| 100 | x | 1 | x | x | x | x | x |
| 101 | count+keep | 1 | 1 | a | x | x | x |
| 110 | count | 1 | 1 | 1 | x | x | x |
| 111 | load | 1 | 0 | x | 0 | 0 | 0 |



1. FSM implementation with off the shelf logic components

The functional verification of the FSM was carried out manually by inspecting the waveforms obtained in the Modelsim logic simulator. The results of the simulation is depicted in In Fig. 8. One can see the state transitions that are conditioned by the control signals a and b corresponds to the specifications in the FSM diagram.

1. a )Modelsim simulation result b) Waveforms captured with a logic analyser

References

2. M. Pedram, “Power Simulation and Estimation in VLSI Circuits”, In The “VLSI Handbook”, Edited by W-K. Chen, The CRC Press and the IEEE Press, 1999.
3. A. Vassighi, M. Sachdev, Thermal and Power Management of Integrated Circuits. Series on Integrated Circuits and Systems. Springer-Verlag Berlin, Heidelberg, 2006
4. T. Sakurai, "Low power digital circuit design," Proceedings of the 30th European Solid-State Circuits Conference 2004, Leuven, Belgium, 23-23 Sept. 2004
5. S. Zhou and J. Bian, "Partition-based Retiming and Precomputation for Dynamic Power Reduction," 2006 International Conference on Communications, Circuits and Systems, Guilin, China, 2006
6. S. Xanthos, A. Chatzigeorgiou, G. Stephanides, "Energy Estimation with SystemC: A Programmer's Perspective", 7th WSEAS International Conference on CIRCUITS, July 2003, Corfu,
7. R. Bevaart, N. Dumitru, R. Nouta, “VHDL Power estimation of CMOS Logic Cells”, Proceedings - ProRISC CSSP97, Workshop on Circuits, Systems and Signal Processing, November, 27 - 28, 1997, Mierlo, the Netherlands
8. A. Chandrakasan, R. Brodersen, “Low Power Digital CMOS Design”, Kluwer Academic Publishers Norwell, MA, USA,1995
9. **At least 20-25 quality references should be cited**

1. This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS/CCCDI – UEFISCDI, project number PN-III-P2-2.1-BG-2016-0268, within PNCDI III. [↑](#footnote-ref-1)