A VHDL Based Approach for Fast and Accurate Energy Consumption Estimations

César A. M. Marcon, Sérgio Johann Filho, Fabiano P. Hessel
PPGCC / FACIN / PUCRS – Av. Ipiranga, 6681, Porto Alegre, RS – Brazil
cesar.marcon@pucrs.br

Abstract

Efficient energy consumption became an important requirement and constraint to be considered in many systems implementations, mainly to the embedded ones. Accurate and efficient power estimation during the design phase is required, in order to meet the power specifications without a costly redesign. High abstraction levels descriptions enable fast energy consumption estimations, but hardly enable accurate estimations. It normally requires evaluations at low abstraction levels, such as electric ones. On the other hand, low abstraction levels require too much design effort and design time. In this sense, this work presents an approach for energy consumption estimation for systems written in synthesizable VHDLs. A VHDL cell library is the base of the methodology, which is characterized with some relevant energy consumption information according to foundry parameters. The use of this approach leads to high-quality energy consumption estimations and design time saving.

1. Introduction

The increases in chip density, operating frequency, and demand for portable applications have made power consumption a VLSI design major concern. Excessive power dissipation in integrated circuits not only discourages their use in a mobile environment, but also causes overheating, which degrades performance and reduces chip lifetime [1]. Therefore, designers need methods and tools to minimize not only the energy consumption, but also the high power density, avoiding reliability problems and expensive cooling systems.

It is imperative the use of tools for power consumption estimation, in order to guide the designer in solving these kind of problems. Unfortunately, accurate estimations can be achieved only with low description levels, which demand too much design and processing time. In addition, low abstraction level descriptions are more error prone, difficult to understand, to maintain, and to document the systems, when compared to high abstraction levels.

These drawbacks encourage researchers in finding new methods and/or models that leads to accurate estimations that overcome the problems. This paper proposes an approach for energy consumption estimation, named **EngyLib** (which stands for Energy Library), which allows the designer describing systems in high abstraction levels, with accuracy estimations of low abstraction levels.

EngyLib highlight is a VHDL cell library, which is characterized according to static power dissipation, dynamic energy consumption, and number of transistors switching. The approach consists of converting a high-level system description to an equivalent low-level one, which is a VHDL netlist composed by the characterized cell. Next, through a simulation step, the energy consumption is estimated according to the input stimuli.

Many levels and description languages may be used to describe the system under energy consumption estimation, like behavioral VHDL and structural Verilog. However, the high-level description has to be synthesizable to a VHDL composed by the characterized cells.

EngyLib is part of CAFES [2], which is a Java open-source framework developed targeting different sort of analyses and syntheses of embedded systems.

Results achieved by comparing electric simulation of small processors and embedded applications with the proposed approach attest the efficiency and efficacy of EngyLib. This approach reduces in average 38 times the estimation time, paying the cost of only 13 percent of energy consumption imprecision, when compared to energy consumption reference, accomplished by electric simulation.

2. EngyLib, a VHDL Based Approach for Energy Consumption Estimation

EngyLib is the approach proposed here, which uses the precision of electric levels to build a VHDL cell library. This one is timing and energy consumption characterized, enabling to estimate through VHDL simulation the static and dynamic energy consumption of circuits. EngyLib provides many benefits, as for instance, VHDL simulation is much faster than an electric one, speeding up the energy consumption estimation of the system under evaluation. In addition, although the approach proposed here synthesizes the system description to a low-level one, the designer may only work at high abstraction levels, since low-level description are seen as an intermediary language of the estimation flow.

Figure 1 illustrates that starting from a system description the designer may estimate the energy consumption, number of switching and execution timing, of the entire system or part of it by performing three steps: (*i*) *logic synthesis*, (*ii*) *connection analysis,* and (*iii*) *system simulation*.

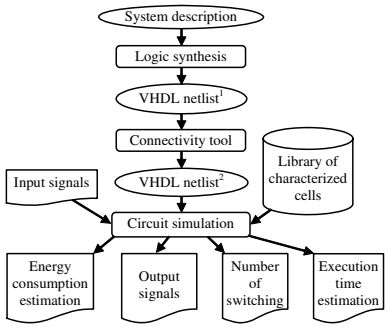

Figure 1 – Workflow for energy consumption estimation

978-1-4244-1710-0/07/$25.00 © 2007 IEEE

Any abstraction level, which reduces the designer description effort and the error prone, may be used to perform the *system description*. The only restriction here is that this description has to be synthesizable, so that a *logic synthesis* tool may convert it to a netlist of VHDL cells. This synthesis is constrained to only the set of cells that are pre-characterized according to energy consumption parameters. This work uses Leonardo Spectrum [3] as the synthesis tool.

Figure 2 depicts a partial VHDL description of an accumulator system, which is a synthesizable one.

```
architecture accumulator of accumulator is
                              ...
begin
                              ...
      soma <= opA + opB;
      process(clock, reset)
      begin
            if reset = '1' then
                  opA <= (others=>'0');
            elsif clock'event and clock = '1' then
                  opA <= opA + 1;
            end if;
      end process;
                              ...
end accumulator;
```

Figure 2 – Partial description of a synthesizable VHDL

The synthesis of the VHDL behavioral of Figure 2 generates a VHDL netlist description, which is partially illustrated in Figure 3, and named *VHDL netlist*[1] in Figure 1.

```
architecture accumulator of accumulator is
                              ...
begin
                              ...
opA_5: dffr port map(Q=>opA_5, QB=>nx210, D=>nx140, CLK=>nx248, R=>nx258);
ix216: nand02 port map(Y=>nx215, A0=>opA_3, A1=>nx50);
ix177: xor2 port map(Y=>saida_6, A0=>nx156, A1=>nx174);
ix157: mux21 port map(Y=>nx156, A0=>nx210, A1=>nx204, S0=>nx146);
                              ...
end accumulator;
```

Figure 3 – Partial VHDL netlist of Figure 2 system

The *connection analysis* step enhances the synthesized VHDL netlist by adding the fan-out estimation of each output signal of each cell. A tool that searches all connectivity of output signals and knows the fan-in of each type of cell does it. This tool is implemented inside CAFES framework [2]. In addition, this tool may help the designer inserting special cells that represents metal lines. This is a partial manual step, which is normally performed when the system under evaluation have large transmission wires.

Figure 4 shows the *VHDL netlist*[2] generated by connection analysis tools. Here, all output signals have annotated a fan-out according to the associated load capacitance.

```
architecture accumulator of accumulator is
                              ...
begin
                              ...
opA_5: Entity work.DffReset port map(Q=>opA_5, FanOut_Q=>2,
         QB=>nx210, FanOut_QB=>3, D=>nx140, CLK=>nx248, R=>nx258);
ix216: Entity work.Nand2 port map(Y=>nx215, FanOut_Y=>2, A0=>opA_3,
         A1=>nx50);
ix177: Entity work.Xor2 port map(Y=>saida_6, FanOut_Y=>1, A0=>nx156,
         A1=>nx174);
ix157: Entity work.Mux2to1 port map(Y=>nx156, FanOut_Y=>2,
         A0=>nx210, A1=>nx204, S0=>nx146);
                              ...
end accumulator;
```

Figure 4 – Partial VHDL netlist with fan-out annotation

```
architecture accumulatorTestBench of accumulatorTestBench is
                              ...
begin
      PowerEstimation1: Entity work.PowerEstimation;
      UUT: Entity work.accumulator port map (
            clock => clock,
            reset => reset, ...)
                              ...
end accumulatorTestBench;
```

Figure 5 – Partial test bench description

Any kind of commercial VHDL simulator can perform the *circuit simulation* step. The designer has just to insert the appropriated input stimuli to the inputs of the circuit under analysis and include the **PowerEstimation** entity inside the VHDL source file. **PowerEstimation** is an EngyLib entity that contains static and dynamic energy consumption estimations, the sum of transistors switching of each transition, timing, and other information. Figure 5 depicts a partial VHDL containing the input file for accumulator system simulation with energy consumption estimation.

3. VHDL Library Characterization

The first step in VHDL library characterization is choosing an appropriate set of cells. If this propose, it was selected a large number of embedded applications to verify the most used cells, and the effect of discard the less used one. This process carried out to a basic library with a set of 15 cells composed by logic gates and flip-flops. Once the set was select, the cells were geometrically and electrically described according to a target technology (for instance, CMOS TSMC 0.35μm). A stimuli generation tool, which was implemented inside CAFES, generates all possible stimulus combinations of each cell. Figure 6 illustrates that having as inputs the target technological library, the input signals, and the electric library, containing the cell behavior and its electric characteristics, an electric simulator generates a set of outputs, reflecting logic, timing, and electric behaviors.

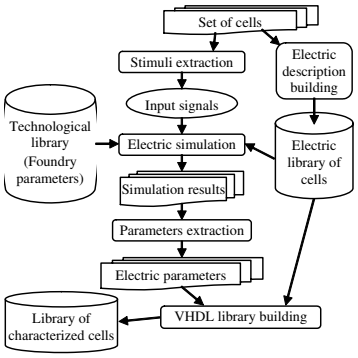

Figure 6 – VHDL library characterization

Each set of input stimulus produces different energy consumptions. Table 1(a) shows the energy consumption of all input combinations of a not gate that generate transistors switching. It considers a fan-out of one gate. Table 1(b) shows the static power dissipation, when inputs do not change.

While fan-out does not affect the static power dissipation, the load capacitance influences directly the dynamic energy consumption. To represent the fan-out influence, two solutions were evaluated: (i) to construct functions that allows the knowledge of energy consumption of a given fan-out by interpolation, and (ii) to annotate explicitly all fan-out possibilities. This work chose the last one, since this last leads to more accurate estimations.

Table 1 – Dynamic energy consumption and static power dissipation for all input combinations of a not gate

(a)

Input transition	Energy consumption (J)
0 ⇒ 1	2.2993 E-13
1 ⇒ 0	2.2258 E-13

(b)

Input transition	Power dissipation (W)
0 ⇒ 0	3.6825 E-10
1 ⇒ 1	7.1430 E-14

Figure 7 depicts a partial VHDL of the characterized not gate, where some energy and power parameters listed in Table 1 may be seen in lines 21, 22, 34 and 35.

```
1   use work.POWER_PCK.all;
2   entity Inv is
3     port(Y: out STD_LOGIC; FanOut_Y: in NATURAL := 3;
4          A: in STD_LOGIC);
5   end Inv;
6   architecture Inv of Inv is begin
7     process(A)
8       variable PreviousA: STD_LOGIC := '0';
9       variable staticPower, cellDynamicEnergy: REAL := 0.0;
10      variable dynamicEnergy: REAL;
11      variable switching: NATURAL := 0;
12      variable transition: STD_LOGIC_VECTOR(1 downto 0);
13    begin
14      transition := PreviousA & A;
15      dynamicEnergy := 0.0;
16      switching := switching + 2;
17      totalSwitching := totalSwitching + 2;
18      case transition is
19        when "01" =>
20          case FanOut_Y is
21            when 1 => dynamicEnergy := 2.2993E-13;
22            when 2 => dynamicEnergy := 3.1251E-13;
23                              ...
24        when "10" =>
25          case FanOut_Y is
26            when 1 => dynamicEnergy := 2.2258E-13;
27                              ...
28      end case;
29      cellDynamicEnergy := cellDynamicEnergy + dynamicEnergy;
30      totalDynamicEnergy := totalDynamicEnergy + dynamicEnergy;
31      totalStaticPower := totalStaticPower - staticPower;
32      transition := (A & A);
33      case transition is
34        when "00" => staticPower := 3.6825E-10;
35        when "11" => staticPower := 7.1430E-14;
36      end case;
37      totalStaticPower := totalStaticPower + staticPower;
38      Y <= not A;
39      PreviousA := A;
40    end process;
41  end Inv;
```

Figure 7 – Partial VHDL description of a not gate

In addition, Figure 7 shows some local and global variables used to compute static and dynamic energy consumption, and the number of transistors switching. For instance, **cellDynamicEnergy** is a local variable containing the total dynamic energy consumed by the cell during the simulation. The global variable **totalDynamicEnergy** contains the sum of all **cellDynamicEnergy** of the system. The local vector **transition** stores the previous input value and the new one. It is used to evaluate if occurred or not a transition in any input signal enabling to estimate both static and dynamic energy consumption. The global variable **totalStaticPower** contains the static energy consumption of all system cells. Each transition the static power value is recomputed to its new value, according to the latest input value. It preserves the consistence of the static energy consumption of the entire system, which is computed each time step of simulation.

VHDL allows global variable through declaring them as *shared* inside a global package. This is the way that each instance of a cell can add its influence in global estimation. Figure 8 illustrates some of the main *shared* variables inside POWER_PCK, which is the power package of EngyLib.

```
package POWER_PCK is
      shared variable totalDynamicEnergy: REAL := 0.0;
      shared variable totalStaticPower: REAL := 0.0;
      shared variable totalStaticEnergy: REAL := 0.0;
      shared variable totalSwitching: NATURAL := 0;
                        ...
end POWER_PCK;
```

Figure 8 –Partial description of the main EngyLib package

Although, this Section depicts a flow to characterize a VHDL library of a particular technology, a set of tools built inside CAFES framework allows to building new VHDL libraries according to others technologies, as long as it will be used the same set of selected cells.

4. Library Refinements and Verification

Estimations achieved with a netlist simulation, based on a cell library, may be imprecise due to many issues, like an inappropriate extraction of electric parameters and an unfitting abstraction level modeling. To overcome it Figure 9 shows the main steps applied to cell library refinement.

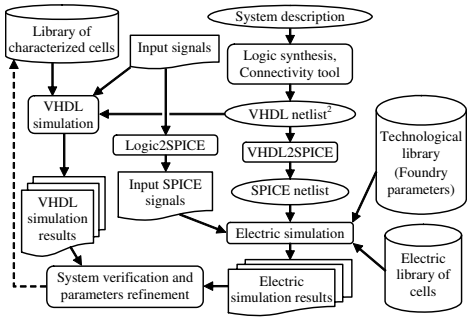

Figure 9 – Cell library refinements and verification

Innumerous circuit descriptions with different complexities may be evaluated with the depicted flow, including the precision of EngyLib. If the acquired precision is not satisfactory, a new energy model may be planned for a new evaluation.

The system description is converted by logic synthesis and the connectivity tool on the same VHDL netlist used in VHDL simulation of previous Section. Then, the VHDL2SPICE tool converts the VHDL netlist onto a SPICE one, in way of performing two equivalent estimations flows for future comparison. The values used as input of VHDL simulation are converted to electric ones with Logic2SPICE tool, according to some electric parameters, such as power supply, slope up, and slope down. Then, simulation results are compared in manner of verifying the accuracy of the proposed method and check the system behavior.

Table 2 – Energy consumption comparison between SPICE simulation and EngyLib of a chain of not gates

Interval (ns)	Energy consumption (J)		Imprecision (%)
	SPICE simulation	EngyLib approach	
2000	8.2274 E-11	8.2206 E-11	0.0827%

Table 2 illustrates the energy consumption achieved by SPICE simulation and EngyLib of a chain of not gates. A simulation during 2000 ns shows an imprecision of less than 0.1%. This imprecision practically maintains for larger evaluation.

Figure 10 illustrates **PowerEstimation,** which is the simulation core architecture. It performs the integration of all cells computation and some global calculi.

```
use work.POWER_PCK.all;
entity PowerEstimation is
end PowerEstimation;
architecture PowerEstimation of PowerEstimation is
      signal step: STD_LOGIC;
                              ...
begin
      process
      begin
            step <= '0', '1' after TREF / 2;
            wait for TREF;
      end process;
                              ...
      process(step)
            variable cycles: NATURAL := 0;
            variable totalEnergy: REAL;
      begin
            cycles:= cycles + 1;
            totalStaticEnergy := totalStaticEnergy +
                                 totalStaticPower * TREF;
            totalEnergy := totalDynamicEnergy + totalStaticEnergy;
      end process;
                              ...
end PowerEstimation;
```

Figure 10 – PowerEstimation architecture, which is the core of EngyLib approach

The parameter **TREF** of Figure 10 is a time reference, implying the precision of VHDL simulator, and affecting the total static energy computation.

5. Experimental Results

It was performed energy consumption estimation on a benchmark using electric simulation and EngyLib approach. Both were performed on a PC with Pentium 4, 3.2 GHz processors and 2 GB RAM. A router, two processors, and three embedded applications compose the benchmark. The selected router is a subcircuit of mesh NoC Hermes [4]. Plasma and 8051 processors are the ones selected. As embedded applications, it is evaluated a Fast Fourier Transform (FFT) and two image applications, one for object recognition and another for image encoding.

Table 3 – Benchmark characteristics

Characteristics		VHDL Code size (lines)	
Circuit	**# of transistors**	**high-level**	**Synthesized**
Router	52,376	630	2,985
8051	148,040	2057	8,967
Plasma	255,240	3089	16,124
FFT	66,550	697	3,699
Object recognition	54,774	612	2,578
Image encoding	55,732	734	2,903

Table 3 reports some characteristics of the benchmark. Column 2 reports the number of transistors for the technology mapped net lists (CMOS TSMC 0.35μm), while Columns 3 and 4 indicate the number of code lines in the original VHDL code and the one achieved by the synthesis tool.

For each circuit, it was firstly employed SPICE simulations to obtain accurate reports of the reference energy consumption. After that, with equivalent input stimuli, all benchmarks were simulated applying EngyLib approach. A random traffic provides input signals to energy consumption estimation of the router; two small programs were used as input of the processors, while embedded applications were stimulated by typical input traces. After that, the corresponding results generated by EngyLib were compared with the SPICE reference report. It produces absolute and relative comparison values depicted in Table 4 and Table 5.

Table 4 shows the elapsed time achieved by electric simulation and EngyLib for each circuit. As it can be observed, the approach proposed here reduces around of 34 times the simulation average time.

Table 4 – Time comparison between electric simulation and EngyLib approach

Simulation time (seconds)			
Circuit	**SPICE**	**EngyLib**	**Reduction**
Router	73,983	2,521	29.35
8051	235,473	5,778	40.75
Plasma	211,850	6,710	31.57
FFT	100,113	2,953	33.90
Object recognition	180,224	5,880	30.65
Image encoding	47,187	1,157	40.78
Average			34.50

Column 2 of Table 5 depicts the energy consumption achieve by each circuit during an electric simulation. Having these energy consumptions as references, Column 3 to Column 5 show the module percentage deviation of three energy models variations. Column 4 presents the results of the most accurate model, since it considerer static power dissipation and the fan-out of each cell. It is also important to observe that energy consumption of each kind of circuit cannot be compared one with each other, since they are completely applications running for different times.

Table 5 – Accuracy of energy consumption estimation, regarding to different energy models

Circuit	SPICE energy consumption reference (mJ)	Energy consumption deviation from EngyLib to SPICE reference (%)		
		With static power dissipation		All fan-outs without static power dissipation
		Single fan-out	All fan-outs	
Router	1.13	5.07	3.11	3.12
8051	5.67	14.78	11.51	11.51
MIPS-like	11.19	31.23	25.12	25.13
FFT	1.47	9.35	8.49	8.49
Object recognition	0.90	8.42	5.03	5.04
Image encoding	1.02	21.05	16.94	16.95
Average		14.98	11.70	11.71

EngyLib approach has, in average, only 11.7% of deviation from SPICE energy consumption reference. It is well accepted when considering the benefits achieved by EngyLib approach, such as simulation time reduction. More deviation is achieved if the model considers only a single fan-out, even if it is achieved as an average fan-out.

Considering this technology the static energy consumption may be neglected when compared to the dynamic energy. On the other hand, the static energy has an important role in the estimation process of deep submicron technology, since it has the same magnitude order of the dynamic energy consumption.

6. Conclusions

Power estimation tools are required to manage the energy consumption of modern VLSI designs during the design phase, to avoid a costly redesign process. In this scenery, this paper proposed an approach based on a pre-characterized cell library, named EngyLib. It enables fast and accurate energy consumption estimations of systems described in high-level abstraction levels. The approach and auxiliary tools make part of a Java open-source framework, enabling the designer make any modification or enhancement into the library, tools, or the proposed approach. Experimental results show that EngyLib achieves high quality estimations, with expressive reduction of the computation time.

References

[1] Farid Najm. A Survey of Power Estimation Techniques in VLSI Circuits. **IEEE Transactions on VLSI Systems**, v. 2, n. 4, p. 446-455, Dec. 1994.

[2] C. Marcon et al. Modeling the Traffic Effect for the Application Cores Mapping Problem onto NoCs. **IFIP VLSI-SOC**, v. 1, p. 391-396, 2005.

[3] Mentor Graphics. Leonardo Spectrum Datasheet. Available at www.mentor.com/products/fpga_pld /synthesis/leonardo_spectrum, 2007.

[4] F. Moraes et al. HERMES: an infrastructure for low area overhead packet-switching networks on chip. **VLSI the Integration Journal**, v. 38, n. 1, p. 69-93, Oct. 2004.