An Automatic Flow for Timing and Static Power Cell Characterization

Ingrid C. Machado, Rafael B. Schivittz, Cristina Meinhardt, Paulo F. Butzen
Universidade Federal do Rio Grande – FURG
Computer Science Center – C3
Rio Grande, Brazil
{ingridmachado, rafaelschivittz, cristinameinhardt, paulofbutzen}@furg.br

ABSTRACT

Characterize a logic gate is the process of establishing reference values about timing and power to the project designers in a Standard Cell project. In this paper, we present an automatic flow to characterize standard cells about timing characteristics and static power consumption. This flow is composed of two main steps: timing characterization and static power characterization. The tool is designed to deal with different technologies, power supplies, transistors arrangements and others boundary conditions.

Categories and Subject Descriptors

B.6.3 [**Hardware]:** Logical Design - *Design Aids* - *Automatic synthesis*

General Terms

Algorithms, Measurement, Design, Verification.

Keywords

Standard Cell Flow, Logic Gates Timing analysis, Static Power, CAD tool.

1. INTRODUCTION

Integrated circuits (ICs) are increasingly present in daily life serving the diverse personal needs. Therefore, different types of devices are required, each one serving a particular design feature. For many projects, the performance is the main characteristic being measured by the frequency of operation of the device. In other projects, the power consumption is a major constraint. And in many cases, you want to find the best possible relation between power consumption and performance that meets the characteristics of the device. These design conditions are called design constraints and define the set of design specifications of an integrated circuit.

There are two main methodologies for design integrated circuit design [1]: full custom design or standard cell design. In full custom design, all circuit is specially designed for the project, optimizing area, power and performance to the maximum for the specified function. All steps, including the layout are customized for a single project, i.e., it is necessary to specify the layout of each individual transistor and the interconnections between them. Differently, standard cell methodology is an example of design abstraction, whereby a low-level very-large-scale integration (VLSI) layout is encapsulated into an abstract logic representation. Along with semiconductor manufacturing advances, standard cell methodology has helped designers scale application-specific integrated circuits (ASICs) from comparatively simple single-function ICs with several thousand gates, to complex multi-million gate system-on-a-chip (SoC) devices. At the low level of design, standard cell libraries are themselves designed using full-custom design techniques[1].

In the design of standard cells is necessary to establishing information about timing, power and noise for each cell. This information set characterize the standard cell. To determine the cell characteristics for a large set of logic functions is a hard task. Even the simplest functions present a lot of different behavior according to, for example, the internal design of the transistors in the circuits, technology, transistor sizing, and the boundary constraints.

In this context, a flow to automatic characterizes logic functions is an important computer aid design (CAD) tool. In this work, an automatic flow to determine timing and static power characteristics of standard cells. This tool is divided in two main blocks: timing characterization and static power characterization.

Next Section introduces some background concepts about timing and power extraction. Section 3 presents the automatic flow proposed and the timing characterization tool and static power characterization tool are described in the sub-sections 3.1 and 3.2 respectively. Finally, Section 4 discusses some conclusions about the tool development and the future works.

2. BACKGROUND

This section presents a collection of concepts and definitions regarding to the tool functionality described in Section 3.

A standard cell is a group of transistors and interconnection structures that provides a Boolean logic function or a storage function. The functional behavior of a combinational logic cell is commonly captured in the form of a truth table or Boolean algebra equation.

A way to represent the Truth table as input for a program is the hexadecimal format. This format is defined by the conversion of the truth table output to the hexadecimal representation, where the I_0 entry of the truth table represents the most significant digit and the I_{2^k-1}, entry represents the least significant digit, where *k* represents the number of entries in the truth table. For instance, for the NOR2 logic gate, the truth table is showed in Figure 1(a). The input combination I_0 has the output O_0 = 1, in other words, when A = 0 and B = 0, F = 1. Ordering the output values, and considering the I_0 entry as the most significant bit, the NOR2 gate can be described as the sequence of outputs O: (1,0,0,0). Reading the digits, the NOR2 gate is represented in hexadecimal format by the value $8_{(h)}$.

Usually, the initial design of a standard cell is developed at the transistor level, in the form of transistor netlist or schematic view. The netlist is a nodal description of transistors, of their connections to each other, and of their terminals to the external environment.

Characterize a logic gate is the process of establishing reference values about timing and power to the project designers. This values need to be realistic and reflects all electrical behaviors of the circuit. This work address two important steps in the cell characterization: determine the delays and static power consumption.

Delay can be divided in propagation delay and transition delay. Propagation delay is the time required for a digital signal to travel from the input(s) of a logic gate to the output. Propagation delay is the time for a gate output to arrive at 50% of its final value. The propagation delay from a transition where the output is high and goes to low is named *TpHL* and the propagation delay from the output going high is named *TpLH*. Terminology *TpHL* and *TpLH* always refers to the transition on the output [1].

Transition time is generally used to mean the time it takes for a gate to arrive at 10% (for a logic 0) or 90% (for a logic 1) of its final value. The terminology for transition time to output fall is *Tfall* and for the output rises is *Trise.*

To determine the transition and propagation delays is necessary evaluate all the arches for the function. Arch of a function is a pair of input combinations that produces a change in the logical state of the output. A delay arch corresponds to the delay measured from a change in gate input to the change in gate output for a specific steady state combination of other input signal values.

For a given logical operation, each arch A_{ij} represents combining pairs of I inputs, which one transition *i* to *j* causes a change in the output state, allowing to measure the delay and propagation times.

The power dissipation in digital CMOS circuits can be decomposed in two parts, which are summarized in the following equation (1). The first term represents the dynamic component of power. This portion is also composed by two parts as presented in equation (2).

$$P_{total} = P_{dynamic} + P_{static} \quad (1)$$

$$P_{dynamic} = P_{switching} + P_{short\text{-}circuit} \quad (2)$$

The switching power $P_{switching}$ is due to the charge and discharge of the capacitors driven by the circuit. The short-circuit power $P_{short\text{-}circuit}$ is caused by the short circuit currents that arise when pairs of PMOS/NMOS transistors are conducting simultaneously in a CMOS gate. It is well explored in [2].

The second term in equation (1) represents the static power component of total power. It is also called leakage power. It is due to the leakage current that flows in the circuit such as subthreshold, gate tunneling or reverse-biased PN junction leakages for instance [3]. For nanometer processes, P_{static} becomes more important and should be considered in the total power dissipation analysis [4].

The output capacitance *CL* of a CMOS gate represents the total capacitance associated with the gate output. This includes the internal capacitances of the MOSFET devices, the wiring capacitance, and the capacitance of the device that the output is connected to. Others examples of boundary parameters are temperature and slope of the input signals (*in slope*).

3. Automatic flow for timing and static power cell characterization

This work presents an automatic flow to determine timing and static power characteristics of standard cells. This tool is divided in two main blocks: timing characterization and static power characterization. The tool is developed in C++ with an optional graphic interface developed with QT Creator IDE [5]. Currently the tool allows the configuration of 13 parameters. The input parameters are divided in three classes of parameters:

- Logic function parameters: function name, truth table in hexadecimal format*.
- Circuit parameters: netlist, Supply source name, Ground name, number of entries, technology.
- Boundary Parameters: inslope*, transition step*, output capacitance*, temperature.

It is also possible to determine the name of the output files. Parameters marked with * are only necessary for timing characterization. Also, this flow allows the user to call each one of the characterization tool individually. All the electrical simulations are executed with NGSPICE [6].

3.1 Temporal Characterization

The temporal characterization of logic gates is a task that can be very laborious, according to the logical operation and the number of entries. The temporal characterization tool receives as entry the logical function parameters (*f*), circuit parameters (*c*) and boundary parameters (*b*) and process all this information to automatic generate the complete simulation file necessary to measure the timing characteristics of a logic gate. The implementation is divided in five steps, as showed in Algorithm 1.

According to the (*f,c,b*) parameters, the temporal characterization starts with the generation of the arches of the function (*l.2*) that receives as input the number of entries of the logical function (*k*) and the logical function represented by its output (*O*) and returns an adjacent matrix (*M*). Next step (*l.3*) is the input waveform generation for make possible determines all the propagation delays for the function. This step receives the adjacent matrix generated in the previous step and returns the transitions path generated from the matrix (P_A). Step 3 generate the source signals for all the entries of the circuit obeying the P_A and returns the set of waveforms for the inputs (I_A). Step 4 generate the correct description of the measurement commands in SPICE language receiving the set of waveforms (I_A) and a list of the falls and rises in the output for the transition path (P_A) generated. Finally, step 5 composes all information generated in the previous steps with the circuit and boundary parameters to generate the complete file description to the simulation.

```
1. Temporal Characterization (f, c, b)
2. M = generate_arches(k, O)
3. P_A = generate_waveform(M)
4. I_A = generate_source_signals(P_A)
5. Measure_A = generate_measure (I_A, VR_A, VF_A)
6. Results = Simulate (Generate_file (I_A, Measure_A, netlist) )
```

Algorithm 1 – Temporal Characterization Algorithm high level description

The first step obtains the delay arches, found through the analysis of the truth table of the logic gate being simulated. For this tool, the logic function is described in the hexadecimal format. The set of all arches of a logical operation is named S_A and it is used to fill an adjacency matrix M of size (2^k-1) x (2^k-1), in other words, with one line and one column for each input combination. The arches are inserted as Boolean values into the adjacency matrix. Cells filled with the value 1 represents arches and cells filled with the value 0 represents that the transitions between the entries combinations do not change the state of the output.

Figure 1 (a) shows the truth table for the NOR2 logic gate It is represented as $8_{(h)}$ in the hexadecimal format Fig. 1 (b). The set of delay arches (*A*) for the function are shown in Fig.1 (c) and the adjacent matrix (*M*) generated for the NOR2 logical function is described in Figure 1 (d). **Generate_Arches***(k, O)* function, showed in Algorithm 2, describes how to obtain the arches and the construction of the adjacency matrix.

	A	B	F
I_0	0	0	1
I_1	0	1	0
I_2	1	0	0
I_3	1	1	0

(a)

F = (1,0,0,0) = $8_{(h)}$

(b)

Set of Arches (**S_A**):

A[0,1], A[1,0], A[0,2], A[2,0]

(c)

$$M = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

(d)

Figure 1 – Truth table (a), hexadecimal format representation, the set of Arches (c) and adjacent matrix (d) for NOR2 logical function

```
1:   Generate_Arches (k, O)
2:   Initialization
         $S_A = \emptyset$, i = 0, j = 0
4:   for each i ≤ ( $2^k$-1)
5:       for each j ≤ ( $2^k$-1)
6:           if O[i] ≠ O[j] then
7:               $S_A$ = A [i,j]
8:               M[i,j] = 1
9:           else
10:              M[i,j] = 0
11:  return M;
```

Algorithm 2 – Generate_Arches (*k, O*)

Generate_waveform (*M*) function is a greed algorithm that uses the adjacency matrix (*M*) to define how the waveforms should be described for a simulation capable to describe all the transitions optimizing the execution time. P_A represents the path build from the adjacency matrix. Breadth –first and Depth- First search algorithms [7] are evaluated to generate the best path optimizing the number of simulation steps (execution time) however they fails to find one reasonable solution. The greed solution for this function is showed in Algorithm 3 and presents good results and smalls modifications are currently being explored to find the best path minimizing the execution time.

```
1:   Generate_waveform(M)
2:   Initialization
         $P_A = \emptyset$, i = 0, j = 0
3:   for each i ≤ ( $2^k$-1)
4:       for each j ≤ ( $2^k$-1)
5:           if $M_{ij}$ = 1 then
6:               PA = (i, j)
7:               M[i,j] = 0
8:               i = j
9:   return $P_A$
```

Algorithm 3 - Generate_waveform(*M*) function

The third step uses the description of the waveforms to generate the source signals (I_A), used as entries in the electrical simulator. The fourth step uses the description of the waveforms to generate the measures of the delay and propagation times. The fifth step gathers the outputs of the previous steps to generate the full file for simulation, which will be used by the electrical simulator. With this file, the electrical simulator will generate the data for the temporal characterization of the logical gate.

Table 1 complete the example for a NOR2 gate showing the timing characteristics generate for a classical static CMOS NOR2 gate in the 32nm predictive technology [8] with *CL* = 1f, inslope = 10ps and supply voltage = 1V. The table describes the delay arc adopted to generate the transition delay and the propagation delay respectively.

Table 1 - NOR2 gate temporal characterization

Arch	Transition time (ps)		Propagation time (ps)	
A[0,2]	tfall_a_b0	12.06	tphl_a_b0	6.99
A[2,0]	trise_a_b0	38.78	tplh_a_b0	19.72
A[0,1]	tfall_b_a0	11.11	tphl_b_a0	6,52
A[1,0]	trise_b_a0	38.81	tplh_b_a0	17.50

3.2 Static power characterization

The static power characterization of a circuit evaluates all possibilities of combination that the circuit can persists static, i.e., with no changes in its entries. To one circuit with k entries, we will have 2^k-1 possibilities to be evaluated, resulting in 2^k-1 values of static power consumption. This means that all entries combinations have to be analyzed to be generated the complete e set of files *F* capable to obtain all the static powers for a logic gate, and also added it in the set of entries *E*.

For that, the static power characterization tool was developed in 4 steps, described in Algorithm 4.

Different of the initial steps of the electric power characterization, for the static characterization the electrical circuit diagram is essential to determine the static power consumption. Different transistor arrangement, sizing and other boundary conditions are fundamental since the first steps of the static power characterization. After receive the circuit and boundary parameters, the execution takes few seconds, reducing all the work to analyze the circuit, having, in an automatic and fast manner, all information to improve the development of a circuit with specific characteristics.

```
1. **Static Power Characterization (f, c, b)**
2. Process_input_parameters();
3. E = Generate_static_states( )
4. F = Generate_files (I,E, netlist)
5. Results = Simulate (F)
6. Final_results = Format_results (Results)
```

Algorithm 4 – Static Power Characterization Algorithm high level description

In the first stage of implementation (*l.2*) of the tool is held data entry tool, such as the technology used, the file with the circuit, the number of inputs of the circuit, power supply, among others.

When the tool detects the number of entries, it adopts a hash table to store the information in their respective places, for a better search of values within the tool. Each piece of information inserted into the initialization of the tool is stored and prepared for be used in the future, for example, information about the number of inputs of the circuit. The second step is to generate all the static states combinations (*l.3*). To perform the combination of input voltages, in order to satisfy all states, the algorithm adopts a masking bit procedure that completes the combination of inputs. Such that, each simulation file has its own state to be analyzed, thus avoiding a repetition of information or a simulation unnecessary.

The third step is the creation of the files made with combinations of input voltages from all sources involved, so as to generate all simulation files. This step is described in Algorithm 5. The simulation file is done inserting all parameters needed to generate the input file for the electrical simulation. This step accomplishes two procedures in order to optimize the runtime of the tool. Thus the analysis is done in the best way possible.

```
1. Generate_files (I, E, netlist)
2. Initialize
3. k= Ø; i= Ø;
4. for each  i < (2^k-1)
5.       create F[i];
6.       for each E in (E[0] ... E[n])
7.                 F[i] = E[j] + I[i , j]
8.        i++
9. return F
```

Algorithm 5 – Generate_files (*I, E, netlist*) function

The fourth step treats the output files of the program, filtering only the information you need, making information visualization much faster and more organized. The relevant information is returned to the user.

Table 2 represents one example for a NOR2 gate. For a function with two entries, we will have 4 different combinations of entries that should be analyzed. In the end of the static characterization, we have the results of the static current for the respective values in the entries of the function. For this experiment, was used a classical static CMOS NOR2 gate described in the predictive 32nm technology [8].

Table 2 - Static Current results for a NOR2 gate

Entries (*E*)	I_{static} (nA)
I[0,0]	37.65
I[0,1]	9.770
I[1,0]	1.736
I[1,1]	553.26

4. CONCLUSIONS AND FUTURE WORKS

Characterize a logic gate is an important task in a standard cell design flow. This work developed and presented an automatic flow to determine two important steps of the cell characterization: timing and static power characterization. Timing constrains are one of the most relevant characteristics of a circuit for the strict relation on the final frequency of operation of a project. Static power becomes more important in nanometer technologies and the designers need to take into account project approaches to deal with the increase in the static consumption. The proposed flow addresses these two characteristics and the validation tests proved that this flow is perfectly able to deal with complex functions.

Future works include extend the flow to incorporate dynamic power consumption characterization and detailed information about static power components. Also is considered start to characterize cells about noise and variability.

5. ACKNOWLEDGMENTS

This work has been supported in part by the Brazilian National Research Council (CNPq - Brazil) and by scientific initiation program of the Universidade Federal do Rio Grande – Furg.

6. REFERENCES

[1]Rabaey, Jan M. et al. Digital Integrated Circuits. Prentice-Hall. Ed. 2., 2003.

[2] Veendrick, H. J. M., “Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits,” Journal of Solid-State Circuits, vol. 19, no. 4, pp. 468–473, Aug. 1984.

[3] ROY, K. et al. “Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits,” Proceedings of IEEE, vol. 91, no. 2, Feb. 2003, pp. 302-327.

[4] International Technology Roadmap for Semiconductors, 2007 Edition. Available at http://public.itrs.net

[5] QT Creator IDE. Available: http://qt.digia.com

[6] NGspice. [Online]. Available: http://ngspice.sourceforge.net/.

[7]Cormen, Thomas H.; et al. Introduction to Algorithms. MIT Press, 2001

[8] Y. Cao, T. Sato, D. Sylvester, M.Orshansky, C. Hu., "New paradigm ofpredictive MOSFET and interconnect modeling for early circuit design," *Proc.IEEE Custom Integrated Circuits Conference*, pp. 201-204, 2000