An Interconnect-Driven Low Power Design Methodology

Shih-Hsu Huang

Department of Electronic Engineering,
Chung Yuan Christian University,
Chung Li, Taiwan
E-mail: shhuang@cycu.edu.tw

Hsu-Ming Hsiao

SoC Technology Center,
Industrial Technology Research Institutes,
Hsinchu, Taiwan
E-mail: 870855@ccl.itri.org.tw

ABSTRACT

Low power is a significant concern for the today's ASIC designs. To shorten the design time, it is very important to correctly supply the design environment all the power related information that is necessary. However, the interconnect capacitance estimation is a difficult task during the synthesis stage due to the lack of place and route information. In this paper, we will present an interconnect-driven design methodology. To minimize the iterations between synthesis and layout, the proposed approach is distinctive in that it constructs physical hierarchy during the synthesis stage. Our optimization goal is to minimize the power dissipation of the chip, especially when the system is at the standby mode. Experimental data shows that this design methodology achieved very good results.

1. INTRODUCTION

The dominant source of power dissipation in CMOS circuits is the charging and discharging of the node capacitances [1,2,3], and is given by the following equation:

$$P_i = 1/2 * V^2 * C_i * f * E_i$$

where V is the supply voltage, f is the clock frequency, C_i is the node capacitance, and E_i (referred as the switching activity) is the expected number of transitions per clock cycle. The product of C_i and E_i is referred to as the switched capacitance. An estimate for the power consumption of a logic network is obtained by summing the power consumption over all the nodes. At the logic level, it is assumed that V and f are fixed, and thus, the total switched capacitance of the circuit is assumed to be the cost measure that is optimized.

Power optimization of VLSI circuits can be accomplished at the various levels of design abstraction from algorithmic and system levels down to layout levels. The optimization at higher level can exploit a larger degree of freedom and then have a larger impact in the power dissipation reduction. However, the power estimation is a difficult task at the higher level because interconnect plays a role in determining the total chip power dissipation. The power estimation may be inaccurate due to the lack of physical place and route information. Since time-to-market is one of the most important factors in the ASIC design, in order to reduce the design time, the number of design iterations should be minimized. To shorten the design time, physical place and route information must be accounted for as early as possible.

Many design automation tools have been used extensively in the industry and are an integral part of typical design flow. These tools achieve good results in their own design stages. Our goal is to provide necessary and accurate information at each design stage. As a result, by taking advantage of existing design automation tools and combining their features, the proposed design methodology is able to develop low power digital systems while speeding up the design process. In other words, our approach is to develop a methodology for effective and efficient use of existing design automation tools.

This proposed design methodology has been applied to a wireless communication chip to verify its correctness and effectiveness. If compared with the typical ASIC design flow, the main distinctions of the proposed approach are elaborated as the below:

- *It tries to minimize the power dissipation when the system is at stand-by mode.* For most of usage time (at least, above 95% of usage time), this wireless communication system stays at the standby mode to detect and be ready to receive new message. In order to extend the battery life, the power dissipation when the system is stand-by should be as low as possible.

- *It tries to minimize the capacitances of high switching frequency nets during the stage of logic synthesis.* By applying the procedure of quick synthesis and practical wire load models, the approach may accurately estimate the minimum areas of logic modules and the corresponding capacitances of high switching frequency nets. Power optimization tool, like Synopsys Power Compiler, may optimize the logic design based on the estimated capacitances.

- *It tries to let the final capacitances in the layout stage as close to the estimated value in the synthesis stage.* During the stage of placement and routing, the logic modules of high frequency nets are group together to form a minimum physical block (i.e., the same size as predicted in the synthesis stage). As a result, the final capacitance will be as close to the estimated values. And, hence, the number of design iterations between synthesis and layout can also be reduced.

The rest of the paper is organized as follows. Section 2 presents the motivation of interconnect-driven design flow.

Section 3 addresses the creation of practical wire load models. Section 4 presents the proposed design methodology. Section 5 shows the experimental results on the implementation of the wireless communication chip. Finally, some concluding remarks are given in Section 6.

2. MOTIVATION

Power optimization of VLSI circuits can be accomplished at the various levels of design abstraction. The optimization at higher level can exploit a larger degree of freedom and then have a larger impact in the reduction of power consumption. Clock gating is a very common power optimization technique at register transfer level design. Furthermore, due to clock gating, we can distinguish the registers, design instances, and logic modules according to their clocks.

2.1 Clock Gating

A very common power optimization technique is clock gating. By turning off the clock when its circuitry is not required, designers can reduce significantly the switching activity within the design. This technique can be applied at the module level as well as internal to modules around specific circuitry. While the power saved may be considerable, the designer must be cautious in the design of combinational logic between registers of different clock systems.

Figure 1 shows the clock enabling circuit of the wireless communication chip designed in the ITRI. The clock source is divided into 6 clocks because of the low power requirement. The *clock0* is the clock source. The *clock1*, *clock2*, *clock3*, *clock4*, and *clock5* are gated clocks. Each gated clock is enabled at different condition. Because these 6 clocks are originated from the same clock source, they are required to remain synchronized.

2.2 Typical ASIC Design Flow

There are many opportunities to reduce power at the register transfer level, including clock gating and other architecture trade-offs. Once the architecture has been decided upon, the designer moves into the implementation phase. Here, gate level synthesis is required.

Timing driven synthesis is a common practice in today's ASIC implementation. As the processing technology shrinks down to the deep sub-micron arena, the interconnect delay on a chip dominates the gate delay and the gate count on a chip may be as large as millions of gates. Because the gate count is huge, in order to synthesize the design effectively, each design must be divided into many hierarchical blocks and synthesized separately. Due to clock gating, we can divide the logic modules according to their clocks. After synthesis, in the placement & route stage, many hierarchical blocks are flattened and grouped together to form a big physical block. Here, back annotation of parasitic values is required to verify that design constraints have been met. Since time-to-market is one of the most important factors in the ASIC implementation, the number of loops between the synthesis and layout should be minimized.

Many design automation tools have been used extensively in the industry and are an integral part of typical ASIC design flow. In the following, we briefly describe the design automation tools [4,5,6,7,8] used in our design environment to implement an ASIC design.

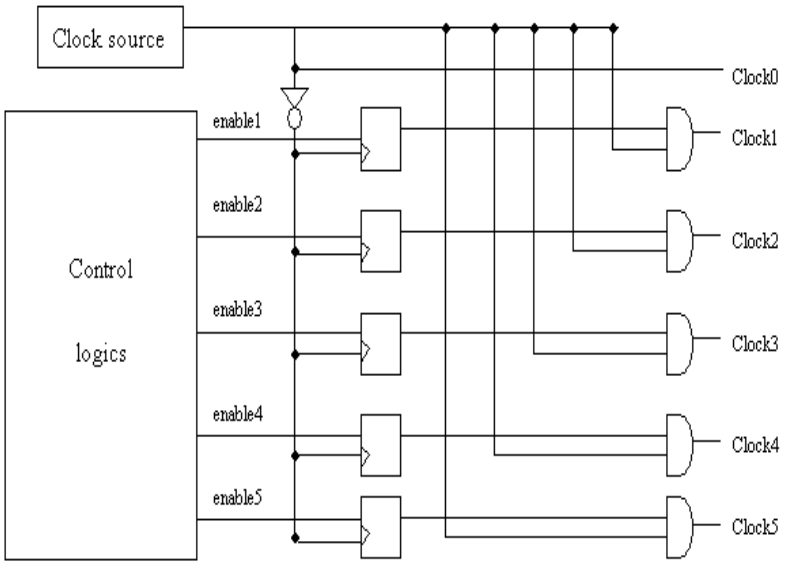

Figure 1: The Clock Enabling Circuit.

The switching activities of nodes in a circuit are reported by the logic simulator (*Cadence/Verilog PLI*) and fed into the synthesis tool. Then, we use *Synopsys* synthesis tool, including *HDL Compiler*, *Design Compiler*, *Power Compiler*, and *Design Power*, for gate level timing, area, and power optimization. Then, *Avant!/Apollo* is used for placement & route. After layout is done, the distributed resistance/capacitance (RC) is extracted by a distributed RC extractor (*Avant!/StarRC*). Then, we use a delay calculator (*Ultima/MDC*) to calculate timing delay and generate a Standard Delay Format (SDF) file which stores the timing delay data of the cells and the nets. The SDF file is fed to a timing simulator (*Cadence/Verilog*) for post-layout simulation.

Since low power is an important requirement for this wireless communication chip, the circuit level netlist with distributed RC is also fed to a power simulator (*EPIC/PowerMill*) to verify that the power constraint is met or not.

2.3 Basic Ideas

There are many techniques, which may decrease the switching activity and/or the capacitance, to reduce the switching power of a logic design. Logic structuring, gate sizing, pin swapping and technology mapping are examples of techniques that allow reduction of switching power by manipulating the gate level netlist. Furthermore, these techniques also have been implemented into logic synthesis tools, e.g., Synopsys/Power Compiler. As a result, during the synthesis stage, we may optimize simultaneously for timing, area, and power. However, the major challenge for us is that the power estimation may be inaccurate due to the lack of physical place and route information. In order to minimize the iterations between synthesis and physical design, it is important that accurate physical hierarchy can be predicted during the synthesis stage.

The proposed design methodology tries to minimize the power dissipation of the chip and the design time (i.e., the number of iterations between synthesis and layout) at the same time. The basic ideas behind the proposed methodology are elaborated as the below:

- The mismatch of the logic hierarchy and the physical hierarchy may cause the estimated wire load far away from the correct value. Using the "wire load model" to provide an estimate of interconnect loading is a practical method to solve the problem. A "wire load model" describes wire load value according to the fan-out number of a net and the physical size of the block that encloses the net. The design methodology includes the method to create wire load models and the design procedure to select the appropriate wire load model during synthesis.

- The reduction of power is larger, if the capacitances of higher switching frequency nets are reduced. Therefore, we will group high switching frequency nets into physical blocks as small as possible. In other words, we will enclose the high switching frequency nets within smaller physical blocks so that the interconnect capacitances may be as small as possible. Furthermore, by using wire load models, logic synthesis tool may apply power optimization techniques, such as logic structuring and pin swapping, based on accurately estimated wire capacitances.

3. PRACTICAL WIRE LOAD MODEL

The dominant source of power dissipation in CMOS circuits is the charging and discharging of the node capacitances. A node capacitance includes the pin capacitances and the interconnect capacitance. Accurate pin capacitance can be obtained from the standard cell library. The interconnect capacitance estimation is, however, a difficult task during synthesis due to the lack of place and route information.

To shorten the design time, it is very important to correctly supply the synthesis environment all the power related information that is necessary during the synthesis procedure. In the following, a procedure to create wire load models for a specific processing technology is presented. The created wire load model is a statistical result of previous layouts of production chips. It is included in the synthesis library. Then, for a brand new design, the logic designer can use the wire load model for optimization during the synthesis stage. According to our experience, the difference between the wire load model and the real layout will be confined within a small value.

3.1 Wire Load Model Definition

A wire load model [9,10,11] describes wire load value according to the fan-out number of a net and the physical size of the block encloses the net. It is created for specific processing technology and layout tool. The wire load model consists of a set of wire load tables. Each table contains wire load data such as capacitance, resistance, and the average wire length for a series number of fan-out.

3.2 Wire Load Model Creation Procedure

To create wire load model, we need to collect physical blocks from many layouts of the same processing technology and the same place & route tool. These collected layouts are from previous production chips.

Then, we can analyze the average wire load for each physical block. Firstly, we use *Avant!/Apollo* as the placement and route tool, use *Cadence/Dracula* to extract interconnect capacitance, and *lpe2syn* that is an in-house program to translate the output capacitance file of *Dracula* to the input format of *Synopsys/Floorplan Manager*. Then the *Floorplan Manager* will create average wire load for different fan-out number of every physical block.

After applying the analysis procedure to every physical block, we further analyze all the wire load data. The wire load increases with area and for the same area the wire load increases with the number of fan-outs. We applied the linear regression technique to the data and get a family of

slopes of different wire load tables.

4. LOW POWER DESIGN METHODOLOGY

To reduce iterations between logical and physical design, wire load of nets should be estimated as close to the actual value as possible. The wire load model creation procedure, as described in Section 3, is a practical method to provide an accurate estimate during the synthesis stage. By supplying the wire load model, the synthesis tool may use the precise wire loads to optimize the netlist for timing, area, and power simultaneously. As a result, the number of iterations can be minimized and design convergence can be easily achieved.

4.1 Front-End Design Procedures

By constructing physical hierarchy during the synthesis stage, we can effectively make use of the practical wire load model. Figure 2 shows the procedure of front-end design. The distinctions of our approach are elaborated in the followings.

4.1.1 Map Logic Hierarchy to Physical Hierarchy

The reduction of power is larger, if the capacitances of higher switching frequency nets are reduced. For the illustrated wireless communication chip, most of its usage time stays at the standby mode to detect and be ready to receive new message. When the chip is at standby mode, only original clock source *clock0* is active. Therefore, we prefer to enclose the registers and the modules, which are driven by original clock source, within a smaller physical block. As a result, a smaller interconnect capacitance can be obtained.

Selecting a wire load model for a logic module is a key factor in correctly predicting the wire capacitances during the synthesis stage. In order to select correct wire load model, accurately estimating the physical block size for a logic module is very important. The correspondence may be derived from past experience. According to previous size for a logic module M, which has total cell area $Cell\_Area(M)$, is equal to $f_b*Cell\_Area(M)$. Furthermore, the estimated physical block size for the logic modules M_i $(1 \leq i \leq n)$, which have total cell area $\sum Cell\_Area(M_i)$, is equal to $f_b*\sum Cell\_Area(M_i)$.

Let's use Figure 3 (a) as an example to describe the procedure. Figure 3 (a) is the logic hierarchy of a design. Suppose leaf modules A and B are driven by original clock source. In order to correctly select the wire load model, designers should plan ahead to map the logic hierarchy to a physical hierarchy. The procedure of logic synthesis is described as below:

Step 1: Perform a quick synthesis without setting any constraint on the leaf modules from A to E in order to get the total area of standard cells inside each leaf module.

Step 2: Since leaf modules A and B are driven by original clock source, we prefer the physical block enclose the two modules as small as possible. The estimated area of the physical block is $f_b*(Cell\_Area(A)+Cell\_Area(B))$. As a result, we can select a wire load model according to the estimated area. Then, we re-synthesize leaf modules A and B with the selected wire load model to optimize timing, area, and power simultaneously.

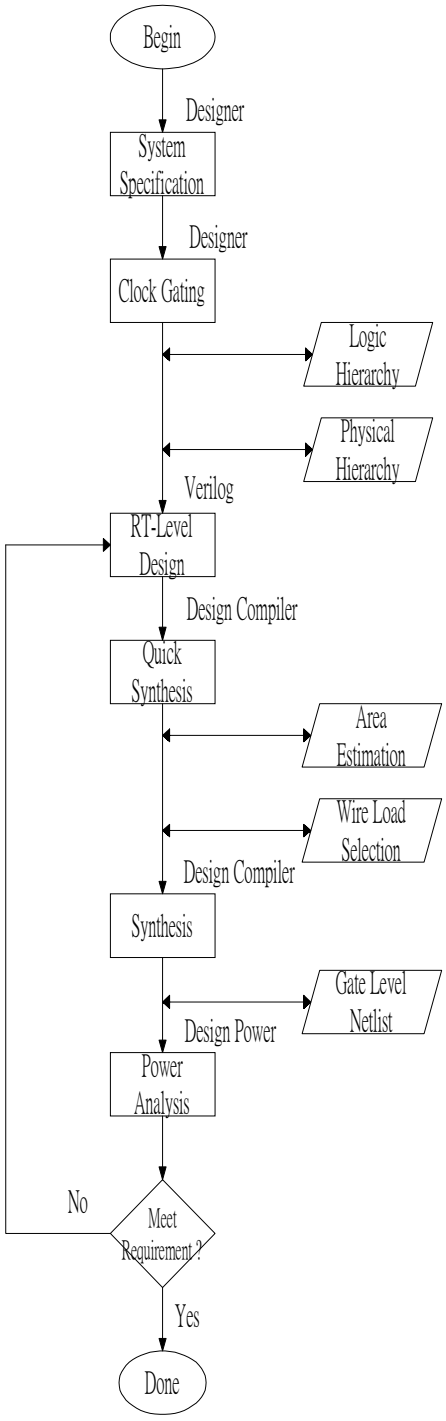

Figure 2: The Front-End Design Procedures.

Step 3: Tackle the synthesis of logic module F. The module F is the parent module of leaf modules A, B, and C. Therefore, the estimated area of the physical block, which

encloses module *F*, is as below:

f_b*(Cell_Area(A)+Cell_Area(B)+Cell_Area(C)).

We can select the wire load model according to the estimated area. Then, we re-synthesize leaf module *C* with the selected wire load model and link *A*, *B*, and *C* to form *F* also using the selected wire load model.

Step 4: Tackle the synthesis of logic module G. The module G is the parent module of leaf modules *D* and *E*. Therefore, the estimated area of the physical block, which encloses module G, is as below:

f_b*(Cell_Area(D)+Cell_Area(E)).

We can select the wire load model according to the estimated area. Then, we re-synthesize leaf module D and E with the selected wire load model and link D and E to form G also using the selected wire load model.

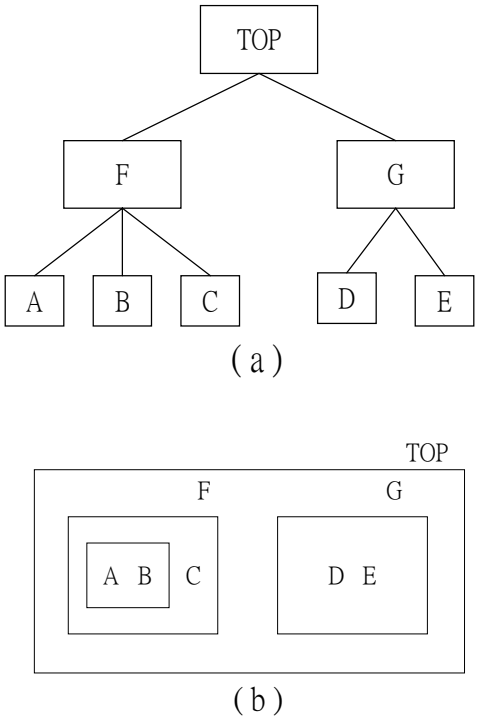

Figure 3: An Example. (a) The Logic Hierarchy. (b) The Physical Hierarchy.

Therefore, the estimated area of the physical block, which encloses module G, is as below:

f_b*(Cell_Area(D)+Cell_Area(E)).

We can select the wire load model according to the estimated area. Then, we re-synthesize leaf module D and E with the selected wire load model and link D and E to form G also using the selected wire load model.

Step 5: At the top level, when the modules F and G are linked together, the global wire load model of the top level is used.

4.1.2 Low Power Clock Tree Design

Since the original clock source is the net of highest switching activity, we need to construct a low power clock network. As a result, the chip may have lower power dissipation, especially when the chip is at the standby mode.

We use clock tree structure to implement the clock network. The low power clock tree is constructed that minimize the load on the clock drivers, subject to meeting a tolerable clock skew. The tree is built from the leaves (i.e., receivers) to the root (i.e., source). The procedure is described below.

Step 1: In order to minimize power dissipation, we select drivers with lower driving capability to be used at the bottom level. The number of receivers may be driven by each driver is calculated by the equation:

maximum number of receivers = (maximum driving capacitance – wire load capacitance) / (input pin capacitance of receiver)

The maximum driving capacitance and input pin capacitance are defined in standard cell library. The wire load capacitance is an estimated value as described in Section 4.1.1

Step 2: After the bottom level drivers are defined, then the input capacitance of these drivers are used as the capacitance load for the selection of drivers of one level above it. To reduce the power dissipation, the smallest drivers that have enough driving capability are chosen. Repeating this process until the clock source is reached.

4.2 Back-End Design Procedures

In the previous section, we have discussed how to control the effects of physical layout during the synthesis stage. In this section, we will describe how to realize the physical layout.

4.2.1 Floorplan

The mismatch in physical and logical hierarchy may cause many number of iterations between synthesis and layout. In order to shorten the design time, the logic hierarchy and the physical hierarchy must be consistent.

Let's use the logic hierarchy in Figure 3 (a) as an example. In order to minimize the lengths of interconnections, the logic modules that have strongly relation had better been placed in the same physical block. For example, logic module *A*, *B* and *C* are placed in the same physical block, and logic modules *D* and *E* are placed in the same physical block. Furthermore, since *A* and *B* are driven by original clock source, we prefer the physical block enclose them as small as possible. Figure 3 (b) shows the physical hierarchy of the logic hierarchy in Figure 3 (a).

After the physical floorplan is done, we can start the

placement & route process. The command "group" of placement & route tool may be used to instantiate logic modules or instances in a physical block.

4.2.2 The Synthesis of Clock Trees

A procedure to construct a low power clock tree for original clock source is described in Section 4.1.2. We use *Avant!/Apollo* to implement the clock tree during the layout stage. After cell placement, we supply the number of levels, the number of fan-outs at each level, and the driver type at each level to the *Avant!/Apollo*. According to the given clock tree structure, it will perform the clock routing, and update the netlist.

Since the gated clocks originated from the same clock source, they need to synchronize with each other. In order to minimize the skew among the gated clocks, their propagation delays must be as close to the propagation delay of original clock source. Therefore, we supply the propagation delay of original clock source to the layout tool to implement the other clock trees (i.e., the clock trees of gated clocks).

5. EXPERIMENTAL RESULTS

We used a wireless communication chip as an example to test the effectiveness of the proposed design methodology. This test chip is implemented using 0.35μm CMOS technology [12].

Because of the low power requirement, the system clock is divided into 6 clocks. Each clock is enabled at different condition for power saving. The architecture of the clock enabling circuit is the same as the concept of gated-clock for low power. Because these 6 clocks are originated from the same clock source, they are required to remain synchronized. When the system is at the standby mode, all the gated clocks in the chip are disabled (i.e., only system clock is active). On the other hand, the chip will have peak power consumption when all 6 clocks are active.

Condition	chip1	chip2	power reduction
standby_mode	0.162 mA	0.173 mA	6.3%
peak	20.76 mA	20.92 mA	0.8%

Table 1： Power Dissipation Reduction Percentage Using The Proposed Design Methodology.

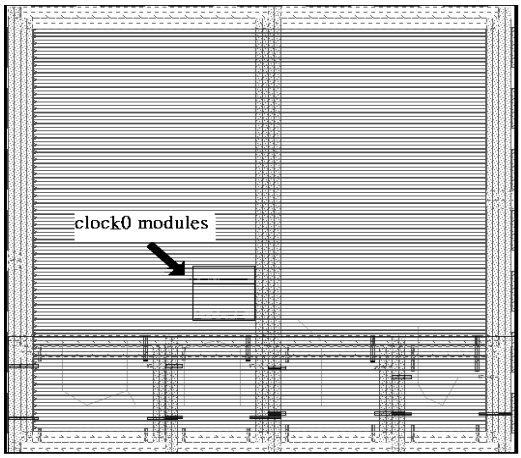

Figure 4： The Final Layout of the Test Chip.

Figure 4 shows the final layout of our experiment. The rectangle pointed by an arrow is the region of logic modules clocked by *clock0*. Table 1 presents power dissipation reduction with the proposed design methodology. The column *chip1* is the chip implemented with the proposed design methodology. The column *chip2* is the chip implemented without the proposed methodology (i.e., only with typical timing-driven ASIC design flow). The row *standby_mode* shows the power dissipation at the standby mode, and the row *peak* shows the peak power dissipation. The results of power dissipation are simulated and reported by *EPIC/PowerMill* under the conditions of 3.3 volts of voltage and the typical timing corner. Experimental data shows that, with the proposed design methodology, the standby mode power dissipation is 0.162 mA and the peak power dissipation is 20.76 mA; while without the proposed design methodology, the standby mode power dissipation is 0.173 mA and the peak power dissipation is 20.92 mA. In other words, the proposed design methodology have 6.3% power dissipation reduction in the standby mode and have 0.8% power dissipation reduction in the peak power dissipation, respectively.

6. CONCLUSIONS

In this paper, we presented an interconnect-driven low power design methodology for a wireless communication chip. This approach may be easily incorporated into existing ASIC design flow. Our goal is to minimize the power dissipation and shorten the design time at the same time. The goal is achieved by providing necessary and precise power information at each design stage. The main distinction of the proposed approach is that the physical hierarchy will be constructed during the synthesis stage. Experimental data shows that it achieved very good results, especially in the standby mode power dissipation reduction.

7. ACKNOWLEDGMENT

This work was supported in part by the National Science Council of Republic of China under contract number NSC 89-2218-E-033-023.

8. REFERENCES

[1] S.M. King, “Accurate simulation of power dissipation in VLSI circuits”, in IEEE Journal of Solid State Circuits, 21(5):889-891, 1986.

[2] S. Rajgopal and G. Mehta, “Experiences with simulation-based schematic level current estimation”, in Proc. of International Workshop on Low Power Design, pages 129-134, 1995.

[3] Synopsys Inc., “Power management in high-level design”, 1996.

[4] Synopsys Inc., “Deep submicron Technology Design Methodology”, 1995.

[5] Cadence Inc., “SDF toolkit reference manual”, 1991.

[6] Avant! Inc., “Apollo user guide”, 1998.

[7] Avant! Inc., “IC Layout Getting Started Guide”, 1995.

[8] C.H. Chien, “CCL 0.35um standard cell library”, in Technical Journal of Computer & Communication Research Laboratories, pages 32-40, Volume 72, 1998.

[9] Synopsys Inc., “Synopsys Link-to-Layout Methodology Flow Application Note”, 1996.

[10] S.H. Huang, K.C. Jung, et. al., “Links-to-Layout design environment”, in Technical Journal of Computer & Communication Research Laboratories, Volume 52, 1996.

[11] J.M. Tseng, S.H. Huang, et. al., “The creation and application of wire load model”, in Technical Journal of Computer & Communication Research Laboratories, pages 21-27, Volume 62, 1997.

[12] S.H. Huang, “A design methodology for a 0.35μm IC project”, in Technical Journal of Computer & Communication Research Laboratories, pages 42-47, Volume 82, 1999.