

## Synthesis of Anti-Aging Gated Clock Designs\*

SHIH-HSU HUANG, CHUN-HUA CHENG AND SONG-BIN PAN

*Department of Electronic Engineering*

*Chung Yuan Christian University*

*Chungli, 320 Taiwan*

*E-mail: {shhuang; g9402601; g9576042}@cycu.edu.tw*

As device dimensions shrink to deep sub-micron ranges, the hot-carrier effect is a main concern for the long-term reliability. It is known that the delay degradation (induced by the hot-carrier effect) of a logic gate is linearly proportional to the number of output switching activities. Due to the clock gating design, the clock gates often have different active probabilities, which lead to different delay degradations. The difference among the delay degradations of clock paths results in an additional clock skew, called aging skew. Based on that observation, in this paper, we present the first attempt, called anti-aging clock gating (AACG), for the synthesis of gated clock designs with the aging skew considered. Given a constraint on the power consumption, the objective of our AACG is to minimize the aging skew (by equalizing the delay degradations of clock paths as possible). Two integer linear programs (ILP) are proposed to formally draw up our AACG problem in the register-transfer-level (RTL) synthesis stage and the high-level synthesis stage, respectively. Compared with previous works, benchmark data show that our approach can greatly reduce the aging skew without any penalty on the power consumption.

**Keywords:** electronic design automation, high-level synthesis, reliability, clock gating, clock skew, integer linear programming

### 1. INTRODUCTION

In a synchronous sequential circuit, the clock signal defines a relative time reference for the movement of data. The most common strategy for clock distribution is to insert a number of buffers along the clock paths (*i.e.*, the paths from clock source to flip-flops), forming a clock tree structure. The clock skew is the maximum difference among the delays of clock paths. Since the clock skew has a negative impact on the circuit performance, the minimization of clock skew is always a very important topic in the design of synchronous sequential circuit.

Conventionally, it is thought that the clock skew includes two components: the design skew and the process skew. The design skew results from a clock tree design that is not optimal (with respect to a target manufacturing process), while the process skew results from the variation of manufacturing process. Earlier research efforts [1] only focus on the elimination of design skew. In recent years, several research efforts [2] have been paid to construct a robust clock tree (*i.e.*, to minimize the worst-case clock skew) by considering both the design skew and the process skew.

In addition, it is noteworthy to mention that the clock signal is the most active sig-

---

Received August 18, 2008; revised December 3, 2008 & March 9, 2009; accepted April 16, 2009.

Communicated by Yao-Wen Chang.

\* This work was supported in part by the National Science Council of Taiwan, R.O.C., under contract No. NSC 97-2221-E-033-053-MY3.

nal in the circuit. Therefore, it is also important to distribute the clock signal with low power. In fact, it is very often that only a portion of the circuit is active. Thus, clock gating [3-8] has been recognized as one of the most effective techniques to reduce the power consumption. By shutting down the idle modules, clock gating can prevent the circuit from consuming unnecessary power. Further, clock gating can also prevent wasteful switching of the clock tree by masking off the clock at the internal nodes of the clock tree.

Note the synthesis of gated clock designs is dependent on the results of high-level synthesis, including which modules are idle, when, and for how long. Based on a scheduled data flow graph (DFG) and a module binding solution, Tellez *et al.* [3] and Farrahi *et al.* [4] derive the clock control logics, called activity driven clock tree, for low power. Recently, Cheng *et al.* [5] study the synthesis of clock gating in the high-level synthesis stage. Based on a DFG, they [5] derive the module binding solution for low power clock gating. As a result, the power consumption of activity driven clock tree can be further minimized.

On the other hand, as device dimensions shrink to deep sub-micron ranges, the hot-carrier effect [9-11] has become an important concern for the long-term reliability of a circuit. The hot-carrier-induced damage in metal-oxide-semiconductor (MOS) transistors is caused by the injection of high-energy electrons and holes in the gate oxide near the drain region. Those injected carriers may be trapped in the oxide, which causes the degradation of MOS transistors. As a result, the delay characteristics of MOS transistors will change over time. To ensure the long-term reliability, many research efforts [12-17] have been paid to reduce the hot-carrier effect in different design stages. However, to the best of our knowledge, the hot-carrier effect has not been considered in the synthesis of gated clock designs.

In fact, the delay degradation (induced by the hot-carrier effect) of a logic gate is linearly proportional to the number of output switching activities [16, 17]. Due to the clock gating design, the clock gates often have different active probabilities, which lead to different delay degradations. Note that the difference in the delay degradations of clock paths results in an additional clock skew, called aging skew. Obviously, if the aging skew is too large, the circuit will fail to function at some point later in time. Thus, there is a demand to properly control the aging skew.<sup>1</sup>

### 1.1 Comments on Previous Work

To the best of our knowledge, the invention of Cohn *et al.* [18] is the only work that considers the aging skew. By adjusting the design skew, their approach [18] can reduce the effect of aging skew at most 50%. In the following, we use a simple example for illustration.

Consider two clock paths: clock path  $p1$  and clock path  $p2$ . Suppose:

- (1) The delays of clock path  $p1$  and clock path  $p2$  are 50 ps and 50 ps, respectively (*i.e.*, the design skew is 0).
- (2) After ten years, the delay degradations of clock path  $p1$  and clock path  $p2$  are 15 ps and 5 ps, respectively (*i.e.*, the aging skew is 10 ps). Therefore, after ten years, the delays of clock path  $p1$  and clock path  $p2$  become 65 ps and 55 ps, respectively. Ob-

<sup>1</sup> If the clock skew (including the aging skew) is zero, the clock period is the maximum aged combinational delay. Note previous works [16, 17] have tried to reduce the maximum aged combinational delay at the gate-level design stage.

viously, within the ten-year period, the worst-case clock skew is 10 ps (due to the effect of aging skew).

To reduce the effect of aging skew, Cohn *et al.* [18] adjust the design skew (*i.e.*, adjust the delay of clock path  $p1$ ) as below: the delay of clock path  $p1$  is decreased by half of the aging skew. For instance, in this example, the delay of clock path  $p1$  is decreased by 5 ps. As a result, the delay of clock path  $p1$  is adjusted to be 45 ps. Since the delay of clock path  $p2$  is still 50 ps, the design skew becomes 5 ps. After ten years, the delays of clock path  $p1$  and clock path  $p2$  become 60 ps and 55 ps, respectively. Obviously, within the ten-year period, the worst-case clock skew is only 5 ps.

It should be mentioned that Cohn *et al.* [18] do *not* reduce the aging skew. In fact, they [18] only reduce the effect of aging skew by adjusting the design skew. Moreover, the reduction to the effect of aging skew is at most 50%. As a result, if the aging skew is too large, the long-term reliability still cannot be ensured. For example, suppose the aging skew is 10 ps. By applying the invention of Cohn *et al.* [18], the worst-case clock skew is at least 5 ps. Therefore, if the clock skew constraint is 4 ps, the circuit still will fail to function within the ten-year period.

## 1.2 Our Contributions

In this paper, we present the first attempt for the synthesis of gated clock designs with the aging skew considered. We say our design style is anti-aging clock gating (AACG). The framework of our AACG is based on the design methodology of activity driven clock tree [3, 4]. Given a constraint on the power consumption, the objective of our AACG is to equalize the delay degradations of clock paths as possible. As a result, the aging skew can be minimized. It is also noteworthy to mention that our AACG has no penalty on both the design skew and the process skew.

Our work includes the following two aspects.

- (1) *RTL-AACG approach.* We propose an integer linear programming (ILP) approach, called RTL-AACG, to formally draw up our AACG problem in the RTL synthesis stage. Given a scheduled DFG, a module binding solution, and a constraint on the power consumption, our RTL-AACG is to find a gated clock tree construction so that the aging skew can be minimized.
- (2) *HLS-AACG approach.* We propose an ILP approach, called HLS-AACG, to formally draw up the AACG problem in the high-level synthesis stage. Given a scheduled DFG, a set of modules, and a constraint on the power consumption, our HLS-AACG is to find a module binding solution and a gated clock tree construction so that the aging skew can be minimized. Note, compared with our RTL-AACG, our HLS-AACG has a larger solution space. As a result, our HLS-AACG can achieve a smaller aging skew.

In fact, our approach has no conflict with the invention of Cohn *et al.* [18]. The designer can combine our approach with the invention of Cohn *et al.* [18] as below. First, our approach is applied to minimize the aging skew. Then, the invention of Cohn *et al.* [18] is applied to further control the effect of aging skew.

Note, as device dimensions shrink to nanometer ranges, the negative-bias-temperature-instability (NBTI) effect [19] has become another important concern for the long-term reliability.<sup>2</sup> The readers may raise a question: is our approach applicable to the NBTI effect? Actually, from [19], we can derive: in a gated clock tree, the delay degradation (induced by the NBTI effect) of logic gate is also linearly proportional to the active probability. Therefore, our work is also applicable to the NBTI effect. For brevity sake, in this paper, we focus on the hot-carrier effect. However, the extension of our work to the NBTI effect is straightforward.

## 2. ACTIVITY DRIVEN CLOCK TREE

The framework of our AACG is based on the design methodology of activity driven clock tree [3, 4]. In this section, we borrow the materials from [3, 4] to demonstrate the framework.

### 2.1 The Inputs

The design methodology of activity driven clock tree requires high-level synthesis information: a scheduled DFG and a module binding solution. We describe the two inputs as below.

In high-level synthesis, a behavior-level description is represented by a DFG, in which each node corresponds to an operation, and each directed edge corresponds to a dependency relationship. A scheduled DFG is a DFG in which each operation is scheduled into a proper control step to start its execution. Take the scheduled DFG, called *ex*, shown in Fig. 1 for illustration. This scheduled DFG *ex* has eight operations. Operations  $o_1$ ,  $o_2$ , and  $o_3$  are scheduled into control step 1, operation  $o_4$  is scheduled into control step 2, and so on.

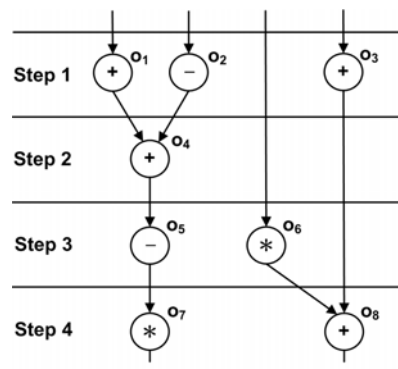


Fig. 1. Scheduled DFG *ex*.

Module binding is to assign each operation in the scheduled DFG to a module that can execute it. Two operations can share the same module if they are not executed in the

<sup>2</sup> To the best of our knowledge, only the hot carrier effect and the NBTI effect may produce the aging skew on gated clock designs.

same control step. Take the scheduled DFG *ex* as an example. Suppose that we are given two adders, called *A1* and *A2*, one subtractor, called *S1*, and one multiplier, called *M1*. Operations  $o_1$ ,  $o_4$  and  $o_8$  can share adder *A1*, operation  $o_3$  can be assigned to adder *A2*, operations  $o_2$  and  $o_5$  can share subtractor *S1*, and operations  $o_6$  and  $o_7$  can share multiplier *M1*. We use the following form to describe this module binding solution:  $A1 = \{o_1, o_4, o_8\}$ ,  $A2 = \{o_3\}$ ,  $S1 = \{o_2, o_5\}$ , and  $M1 = \{o_6, o_7\}$ . For brevity sake, in the following, we say this module binding solution is *sol\_1*.

If a module executes an operation at a control step, then the module is active during that control step, otherwise the module is idle. Based on a scheduled DFG and a module binding solution, we can derive the activity patterns of modules. Consider the scheduled DFG *ex*. Suppose the module binding solution is *sol\_1*. Then, we can derive the activity pattern of each module as displayed in Table 1.

**Table 1. Activity patterns of modules (for binding solution *sol\_1*).**

Module	Control Step				Activity Pattern
	1	2	3	4	
<i>A1</i>	$o_1$	$o_4$		$o_8$	1101
<i>A2</i>	$o_3$				1000
<i>S1</i>	$o_2$		$o_5$		1010
<i>M1</i>			$o_6$	$o_7$	0011

Note, from the activity pattern, we can easily derive the active probability. For example, the activity pattern of module *A1* is 1101. Since module *A1* is active in three control steps and there are totally four control steps, the active probability of module *A1* is 3/4.

## 2.2 Clock Tree Layout Estimation

To estimate the delays and power consumptions of the clock tree, we require detailed information about the parasitic capacitances of the clock tree. Therefore, some assumptions are made for the clock tree:

- (1) The layout of the clock tree is an *H*-tree structure.
- (2) The topology of the clock tree is a binary tree structure, and each node in the binary tree structure corresponds to a clock gate.
- (3) Each clock gate is a two-input AND gate.
- (4) Each clock gate is pre-placed.
- (5) The modules have registers on their inputs that are to be clocked. Based on these assumptions, the delays and power consumptions of the clock tree can be estimated. Further, both the design skew and the process skew can also be controlled.<sup>3</sup>

Fig. 2 (a) gives an example. The layout of the clock tree is an *H*-tree structure. Clock gates  $v_1, v_2, v_3, v_4, v_5, v_6, v_7$  are pre-placed. In Fig. 2 (a), the signals *En1*, *En2*, *En3*, *En4*, *En5*, *En6*, and *En7* represent the clock control logics of clock gates  $v_1, v_2, v_3, v_4, v_5, v_6$ ,

<sup>3</sup> Note that our approaches, including our RTL-AACG and our HLS-AACG, have no penalty on both the design skew and the process skew.

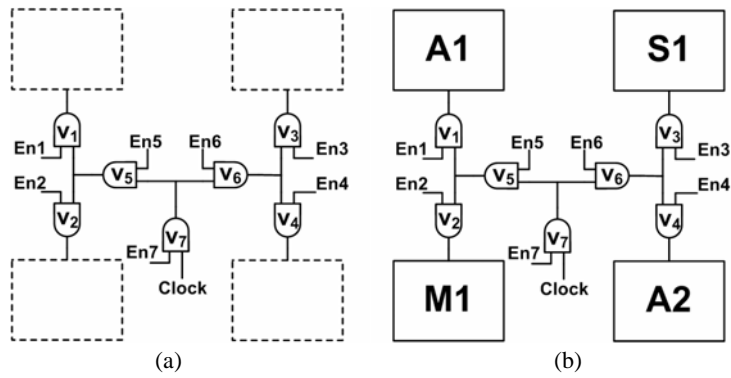


Fig. 2. Clock tree layout style.

and  $v_7$ , respectively. Note, now, the modules have not been placed. The method to derive the placement of modules (e.g., the placement displayed in Fig. 2 (b)) is discussed in section 2.3.

### 2.3 Derivation of Clock Control Logics

Given a clock tree, the activity pattern of each clock gate is defined as below. For a leaf node, its activity pattern is the activity pattern of the module that it controls. For an internal node, its activity pattern can be calculated by OR-ing (bitwise OR operation) the activity patterns of its two children. Obviously, for each clock gate, its activity pattern corresponds to its control logic.

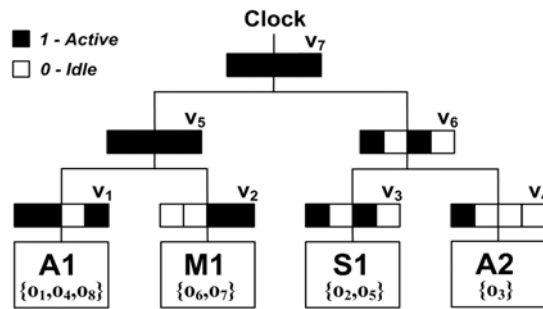


Fig. 3. Clock tree example 1 – CT1.

Take the scheduled DFG  $ex$  for example. Suppose that the module binding solution is  $sol\_1$ , i.e.,  $A1 = \{o_1, o_4, o_8\}$ ,  $A2 = \{o_3\}$ ,  $S1 = \{o_2, o_5\}$ , and  $M1 = \{o_6, o_7\}$ . Fig. 3 gives a possible clock tree, called CT1. Clock gates  $v_1, v_2, v_3$ , and  $v_4$  control modules  $A1, M1, S1$ , and  $A2$ , respectively. From Table 1, the activity patterns of clock gates  $v_1, v_2, v_3$ , and  $v_4$  are 1101, 0011, 1010, and 1000, respectively. By OR-ing the activity patterns of clock gates  $v_1$  and  $v_2$ , that the activity pattern of clock gate  $v_5$  is 1111. Similarly, the activity patterns of clock gates  $v_6$  and  $v_7$  are 1010 and 1111, respectively.

Based on a scheduled DFG and a module binding solution, the objective of [3, 4] is

to build a clock tree (*i.e.*, derive the clock control logics) so that the power consumption is minimized. Note the power consumption of the clock tree includes the power consumption of modules, the power consumption of clock gates, and the power consumption of clock control logics. We use the clock tree *CT1* (as shown in Fig. 3) to illustrate the power calculation as below.

- (1) *The power consumption of modules.* Suppose the power consumptions of adder, subtractor, and multiplier at an active control step are 50, 50, and 100, respectively; the power consumptions of adder, subtractor, and multiplier at an idle control step are 2, 2 and 5, respectively. Thus, the power consumption of module *A1*, *A2*, *S1*, and *M1* are 152, 56, 104, and 210, respectively. The power consumption of modules is  $152 + 56 + 104 + 210 = 522$ .
- (2) *The power consumption of clock gates.* Suppose the power consumption of each clock gate is 10. Thus, the power consumption of clock gates  $v_1, v_2, v_3, v_4, v_5, v_6$ , and  $v_7$  are 30, 20, 20, 10, 40, 20, and 40, respectively. The power consumption of clock gates is  $30 + 20 + 20 + 10 + 40 + 20 + 40 = 180$ .
- (3) *The power consumption of clock control logics.* Suppose the power consumption of each transition in the clock control logic is 1. Since the activity pattern of clock gate  $v_1$  is 1101, its clock control logic *En1* has two transitions: one transition from control step 2 to control step 3, and the other one transition from control step 3 to control step 4. Thus, the power consumption of clock control logic *En1* is  $2 \times 1 = 2$ . Similarly, the power consumption of clock control logic *En2*, *En3*, *En4*, *En5*, *En6*, and *En7* are 2, 4, 2, 0, 4, and 0, respectively. Thus, the power consumption of clock control logics is  $2 + 2 + 4 + 2 + 0 + 4 + 0 = 14$ .

From the above discussions, we conclude: the power consumption of this clock tree is  $522 + 180 + 14 = 716$ .

Finally, it is noteworthy to mention that, for the design convergence, the module placement should follow the built clock tree. Suppose the clock tree layout is estimated as shown in Fig. 2 (a) and the clock control logics are derived as shown in Fig. 3. Then, for the design convergence, we should have the placement of modules as shown in Fig. 2 (b).

### 3. AGING DUE TO HOT-CARRIER EFFECT

The thin oxide damage by the hot-carrier effect impacts the delay of an MOS transistor because it shifts the threshold voltage and decreases the drain current driving capability. In the following, we borrow the materials from [16, 17] to estimate the delay degradations of logic gates due to the hot-carrier effect.

From [16, 17], we know the delay degradation of logic gate  $v$  is equal to  $\delta_v \times sw_v \times CL_v \times FD_v$ , where  $\delta_v$ ,  $sw_v$ ,  $CL_v$ , and  $FD_v$  denote the aging factor<sup>4</sup>, the number of output switching activities, the output load capacitance, and the fresh gate delay, respectively. Further, for a clock gate  $v$ , it is obvious that  $sw_v = TT \times AP_v$ , where the notation *TT* denotes the duration of circuit operation (*e.g.*, ten years) and the notation  $AP_v$  denotes the active probability of clock gate  $v$ . Note that  $AP_v = AS_v/TC$ , where  $AS_v$  is the number of

<sup>4</sup> Note the aging factor is in fact inversely proportional to the driving strength of the logic gate.

active control steps of clock gate  $v$  and  $TC$  is the number of total control steps. Therefore, we can also represent the delay degradation of clock gate  $v$  as  $\delta_v \times TT \times AS_v \times CL_v \times FD_v / TC$ .

In fact, for each clock gate  $v$ , when we derive its clock control logic, the values of  $\delta_v$ ,  $TT$ ,  $CL_v$ ,  $FD_v$ , and  $TC$  are already constants. The reasons are elaborated as below.

- (1) The duration of circuit operation depends on the design specification. Thus, when we derive the clock control logics, we can treat  $TT$  as a constant.
- (2) In our design methodology, the clock gates are placed before their clock control logics are derived. Thus, when we derive the clock control logics, we already know the aging factor, the output load capacitance, and the fresh gate delay. In other words, for each clock gate  $v$ , when we derive its clock control logic, we can treat  $CL_v$ ,  $FD_v$ , and  $\delta_v$  as constants.
- (3) For a scheduled DFG, the number of total control steps is fixed. Thus, when we derive the clock control logics, we can treat  $TC$  as a constant.

Based on the above discussions, for brevity sake, we define a degradation factor  $\alpha_v$ , which is a constant and equals to  $\delta_v \times TT \times CL_v \times FD_v / TC$ . Then, we can re-write the delay degradation of clock gate  $v$  as  $\alpha_v \times AS_v$ . Note that, when we derive the clock control logic of clock gate  $v$ , since the degradation factor  $\alpha_v$  is a constant, the number of active control steps  $AS_v$  becomes the only one variant to affect the delay degradation of clock gate  $v$ . Further, from the formula, we know: the delay degradation of clock gate  $v$  is linearly proportional to the number of active control steps  $AS_v$ .

## 4. MOTIVATIONAL EXAMPLE

In this section, we demonstrate our motivations. Section 4.1 presents the motivation to study the AACG problem in the RTL synthesis stage. Section 4.2 presents the motivation to study the AACG problem in the high-level synthesis stage.

### 4.1 AACG in RTL Synthesis Stage

Previous works [3, 4] minimize the power consumption of the clock tree without considering the aging skew. As a result, although the power consumption of the clock tree is minimized, the aging skew may be very large.

Take the scheduled DFG shown in *ex* for example. Suppose that the module binding solution is *sol\_1*. As studied in section 2.3, the power consumptions of modules, clock gates, and clock control logics are 522, 180, and 14, respectively. Therefore, the power consumption of the clock tree *CT1* is 716.

Suppose that, for each clock gate  $v$ , its degradation factor  $\alpha_v$  is 1. We analyze the delay degradation of each clock gate in the clock tree *CT1* as below. Since the number of active control steps of clock gate  $v_1$  is 3 (*i.e.*,  $AS_{v_1} = 3$ ), the delay degradation of clock gate  $v_1$  is  $\alpha_{v_1} \times AS_{v_1} = 1 \times 3 = 3$ . Similarly, the delay degradations of clock gates  $v_2$ ,  $v_3$ ,  $v_4$ ,  $v_5$ ,  $v_6$ , and  $v_7$  are 2, 2, 1, 4, 2, and 4, respectively.

Then, we analyze the aging skew of the clock tree *CT1*. The clock path from clock source to module *A1* includes clock gates  $v_7$ ,  $v_5$ , and  $v_1$ . Thus, the delay degradation of

the clock path from clock source to module  $A1$  is  $4 + 4 + 3 = 11$ . Similarly, the delay degradation of the clock path from clock source to module  $M1$  is 10; the delay degradation of the clock path from clock source to module  $S1$  is 8; and the delay degradation of the clock path from clock source to module  $A2$  is 7. Therefore, the aging skew of the clock tree  $CT1$  is  $11 - 7 = 4$ .

However, in fact, based on the same scheduled DFG (*i.e.*, the scheduled DFG  $ex$ ), the same module binding solution (*i.e.*, the module binding solution  $sol\_1$ ), and the same power consumption (*i.e.*, 716), we can derive a clock tree which has a smaller aging skew. Consider another possible clock tree – the clock tree  $CT2$  shown in Fig. 4. With an analysis, we find: the power consumptions of modules, clock gates, and clock control logics are 522, 180, and 14, respectively. Thus, the power consumption of the clock tree  $CT2$  is also 716. In other words, this clock tree  $CT2$  has the same power consumption as the clock tree  $CT1$ .

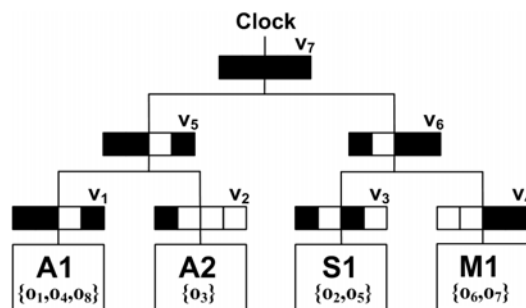


Fig. 4. Clock tree example 2 –  $CT2$ .

Then, we analyze the aging skew of the clock tree  $CT2$ . The delay degradation of the clock path from clock source to module  $A1$  is 10; the delay degradation of the clock path from clock source to module  $A2$  is 8; the delay degradation of the clock path from clock source to module  $S1$  is 9; and the delay degradation of the clock path from clock source to module  $M1$  is 9. Therefore, the aging skew of the clock tree  $CT2$  is  $10 - 8 = 2$ . Compared with the clock tree  $CT1$ , we find: the clock tree  $CT2$  reduces the aging skew from 4 to 2 without any penalty on the power consumption.

From the above observation, we know: even if the clock trees have the same power consumption, they still have different aging skews. Since previous works [3, 4] do not take the aging skew into account, they may obtain a minimum-power activity driven clock tree with a very large aging skew. Therefore, we are motivated to overcome the drawback of previous works [3, 4]. In section 5, we propose our RTL-AACG approach. Based on a scheduled DFG, a module binding solution, and a constraint on the power consumption, our RTL-AACG can construct a clock tree in which the aging skew is minimized.

#### 4.2 AACG in High-Level Synthesis

In fact, the activity patterns of clock gates are dependent on the activity patterns of modules, while the activity patterns of modules are dependent on the module binding

solution. Therefore, if we can consider the AACG problem when we derive the module binding solution, the aging skew can be further reduced.

Take the scheduled DFG  $ex$  for illustration. If the module binding solution is  $sol_1$ , the aging skew is at least 2.<sup>5</sup> Therefore, if the module binding solution is  $sol_1$ , the aging skew cannot be completely eliminated.

Let's consider another possible module binding solution:  $A1 = \{o_1, o_8\}$ ,  $A2 = \{o_3, o_4\}$ ,  $S1 = \{o_2, o_5\}$ , and  $M1 = \{o_6, o_7\}$ . For brevity sake, in the following, we say this module binding solution is  $sol_2$ . Based on the module binding solution  $sol_2$ , Table 2 tabulates the activity pattern of each module.

**Table 2. Activity patterns of modules (for binding solution  $sol_2$ ).**

Module	Control Step				Activity Pattern
	1	2	3	4	
$A1$	$o_1$			$o_8$	1001
$A2$	$o_3$	$o_4$			1100
$S1$	$o_2$		$o_5$		1010
$M1$			$o_6$	$o_7$	0011

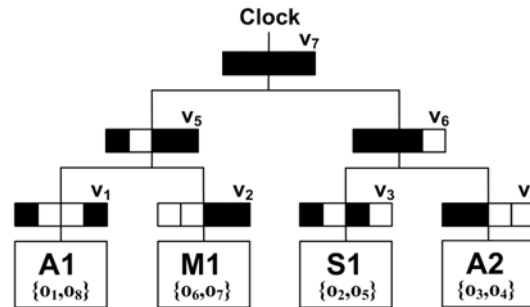


Fig. 5. Clock tree example 3 –  $CT3$ .

If the module binding solution is  $sol_2$ , there exists a clock tree whose aging skew is 0. Fig. 5 gives a possible clock tree, called  $CT3$ , based on the scheduled DFG  $ex$  and the module binding solution  $sol_2$ . We analyze the aging skew of the clock tree  $CT3$ . We find the delay degradation of each clock path is 9. Therefore, the aging skew is 0. Obviously, in the clock tree  $CT3$ , the aging skew is completely eliminated.

From the above observation, we know: the module binding solution has a significant impact on the aging skew. Therefore, we are motivated to consider the AACG problem in the high-level synthesis stage. In section 6, we propose our HLS-AACG approach. Based on a scheduled DFG, a set of modules, and a constraint on the power consumption, our HLS-AACG can further minimize the aging skew by performing the simultaneous application of module binding and clock tree construction.

## 5. RTL-AACG APPROACH

In this section, we use an ILP to formally draw up our RTL-AACG approach. Given

<sup>5</sup> In section 5, we show: if the module binding solution is  $sol_1$ , the minimum value of the aging skew is 2.

a scheduled DFG, a module binding solution and a constraint on the power consumption, our objective is to construct a minimum-aging-skew clock tree. Note, under the given constraints, our RTL-AACG guarantees obtaining the optimal solution.

First, we define the notations used in our ILP as below.

- (1) The notation  $V$  denotes the set that includes all the clock gates in the clock tree. The notation  $B$  denotes the set that includes all the clock gates at the bottom level (*i.e.*, the level nearest to the modules) of the clock tree. The notation  $I$  denotes the set that includes all the modules. The notation  $C$  denotes the set that includes all the control steps.
- (2) The notation  $H$  denotes the set that includes all the clock paths in the clock tree. Note, in the set  $H$ , each clock path is represented by a set that includes all the clock gates in this clock path.
- (3) The notation  $Pow$  is a constant that denotes the constraint on the power consumption of the clock tree. The notation  $PA_i$  (the notation  $PI_i$ ) is a constant that denotes the power consumption of module  $i$  at an active (idle) control step. The notation  $PG_v$  is a constant that denotes the power consumption of clock gate  $v$  at an active control step. The notation  $PT_v$  is a constant that denotes the power consumption of each transition in the clock control logic of clock gate  $v$ .
- (4) The notation  $\alpha_v$  is a constant that denotes the degradation factor of clock gate  $v$ .
- (5) The notation  $parent(v)$  denotes the parent of clock gate  $v$ .
- (6) The notation  $\beta_{i,c}$  is a constant. The value of  $\beta_{i,c}$  is determined according to the scheduled DFG and the module binding solution: if module  $i$  is active at control step  $c$ , then  $\beta_{i,c} = 1$ ; otherwise,  $\beta_{i,c} = 0$ .
- (7) The notation  $A_{v,c}$  is a binary variable. If clock gate  $v$  is active at control step  $c$ , then  $A_{v,c} = 1$ ; otherwise,  $A_{v,c} = 0$ .
- (8) The notation  $T_{v,c}$  is a binary variable. If the clock control logic of clock gate  $v$  has a transition from control step  $c$  to control step  $[(c + 1) \text{ modulo } TC]$ , where  $TC$  is a constant that represents the number of total control steps, then  $T_{v,c} = 1$ ; otherwise,  $T_{v,c} = 0$ .
- (9) The notation  $Y_{i,v}$  is a binary variable. If module  $i$  is the child of clock gate  $v$ , then  $Y_{i,v} = 1$ ; otherwise,  $Y_{i,v} = 0$ .
- (10) The notation  $max\_deg$  (the notation  $min\_deg$ ) is a real-value variable that denotes the maximum delay degradation (the minimum delay degradation) among all the clock paths. Then, obviously, the aging skew is equal to  $(max\_deg - min\_deg)$ .

Next, we introduce our objective function and the constraints. Our objective function is to minimize the aging skew. Thus, we describe our objective function as below: minimize  $(max\_deg - min\_deg)$ .

If clock gate  $v$  is active at control step  $c$ , then its parent is also active at control step  $c$ . Thus, for each clock gate  $v$  and each control step  $c$ , we have the following constraint:

$$A_{v,c} \leq A_{parent(v),c}. \quad (1)$$

At the bottom level of the clock tree, each clock gate  $v$  is the parent of one module. Thus, for each clock gate  $v$  at the bottom level of the clock tree, we have the following constraint:

$$\sum_{i \in I} Y_{i,v} = 1. \quad (2)$$

Each module  $i$  is the child of one clock gate at the bottom-level. Thus, for each module  $i$ , we have the following constraint:

$$\sum_{v \in B} Y_{i,v} = 1. \quad (3)$$

Suppose that clock gate  $v$  (at the bottom level of the clock tree) is the parent of module  $i$ . If module  $i$  is active at control step  $c$ , clock gate  $v$  is also active at control step  $c$ . Thus, for each clock gate  $v$  (at the bottom level of the clock tree), each module  $i$ , and each control step  $c$ , we have the following constraint:

$$Y_{i,v} + \beta_{i,c} \leq A_{v,c} + 1. \quad (4)$$

The variable  $max\_deg$  denotes the maximum delay degradation among all the clock paths. Thus, for each clock path  $h \in H$ , we have the following constraint:

$$\sum_{v \in h} \sum_{c \in C} \alpha_v \times A_{v,c} \leq max\_deg. \quad (5)$$

The variable  $min\_deg$  denotes the minimum delay degradation among all the clock paths. Thus, for each clock path  $h \in H$ , we have the following constraint:

$$\sum_{v \in h} \sum_{c \in C} \alpha_v \times A_{v,c} \geq min\_deg. \quad (6)$$

If the value of  $A_{v,c}$  and the value of  $A_{v,c'}$  are different, where  $c' = [(c + 1) \text{ modulo } TC]$ , then the clock control logic of clock gate  $v$  has a transition from control step  $c$  to control step  $c'$ . Thus, for each clock gate  $v$  at control step  $c$ , we have the following two constraints:

$$A_{v,c} - A_{v,c'} \leq T_{v,c}. \quad (7)$$

$$A_{v,c'} - A_{v,c} \leq T_{v,c}. \quad (8)$$

The constant  $Pow$  gives an upper bound on the power consumption. Thus, we have the following constraint:

$$\sum_{i \in I} \sum_{c \in C} [\beta_{i,c} \times PA_i + (1 - \beta_{i,c}) \times PI_i] + \sum_{v \in V'} \sum_{c \in C} A_{v,c} \times PG_v + \sum_{v \in V'} \sum_{c \in C} T_{v,c} \times PT_v \leq Pow. \quad (9)$$

## 6. HLS-AACG APPROACH

In this section, we use an ILP to formally draw up our HLS-AACG approach. Given a scheduled DFG, a set of modules, and a constraint on the power consumption, our objective is to minimize the aging skew by performing the simultaneous application of mod-

ule binding and clock tree construction. Note, under the given constraints, our HLS-AACG guarantees obtaining the optimal solution.

Our HLS-AACG is developed from our RTL-AACG. The objective function of our HLS-AACG is the same as that of our RTL-AACG. Also, all the constraints in our RTL-AACG still exist in our HLS-AACG. However, it should be mentioned that: since the module binding has not been performed, our HLS-AACG does not know the activity patterns of modules in advance. Thus, in our HLS-AACG, the notation  $\beta_{i,c}$  is a binary variable instead of a constant.

Further, we define the following additional notations for HLS-AACG.

- (1) The notation  $FU(n)$  denotes the set that includes all the modules can execute operation  $n$ .
- (2) The notation  $N(c)$  denotes the set that includes all the operations scheduled at control step  $c$ .
- (3) The notation  $s(n)$  is a constant that denotes the control step that operation  $n$  is scheduled into. Note, for each operation  $n$ , its  $s(n)$  can be obtained directly from the given scheduled DFG.
- (4) The notation  $X_{n,i}$  is a binary variable. If operation  $n$  is assigned to module  $i$ , then  $X_{n,i} = 1$ ; otherwise,  $X_{n,i} = 0$ .

Compared with our RTL-AACG, our HLS-AACG need to have three extra constraints as below. Since each operation must be assigned to one module, for each operation  $n$ , we have the following constraint:

$$\sum_{i \in FU(n)} X_{n,i} = 1. \quad (10)$$

Due to lifetime constraint, two operations cannot share the same module at the same control step. Thus, for each module  $i$  at control step  $c$ , we have the following constraint:

$$\sum_{n \in N(c)} X_{n,i} \leq 1. \quad (11)$$

If operation  $n$  is assigned to module  $i$ , then module  $i$  is active at control step  $s(n)$ . Thus, for each operation  $n$  and each module  $i$ , we have the following constraint:

$$X_{n,i} \leq \beta_{i,s(n)}. \quad (12)$$

## 7. EXPERIMENTAL RESULTS

Section 7.1 describes the setup of our experimental environment. Section 7.2 compares our RTL-AACG and our HLS-AACG with the approach proposed in [3, 4]. Section 7.3 compares our HLS-AACG with the approach proposed in [5]. Section 7.4 makes comparisons between our RTL-AACG and our HLS-AACG. Section 7.5 demonstrates the CPU times of different approaches.

### 7.1 Setup of Experimental Environment

We use Extended LINGO Release 10.0 as the ILP solver. The platform is Windows 2003 x64 running on Intel Xeon E5355 CPU with 8GB RAM. Six benchmark circuits are used to test the effectiveness of our approach. Benchmark circuits HAL, BF, EF, and DWT are popular DSP applications, while benchmark circuits IDCT2 and DIST2 are representative functions adopted from the MediaBench suite [20]. The scheduled DFGs of these benchmark circuits are derived by the list scheduling approach [21].

Table 3 tabulates the characteristics of benchmark circuits. The column *Operations* gives 6-tuple ( $\#ad$ ,  $\#su$ ,  $\#mu\#$ ,  $\#di$ ,  $\#se$ ,  $\#co$ ), where  $\#ad$ ,  $\#su$ ,  $\#mu\#$ ,  $\#di$ ,  $\#se$ , and  $\#co$  are the numbers of addition operations, subtraction operations, multiplication operations, selection operations, and comparison operations, respectively. For example, benchmark circuit HAL has 2 addition operations, 2 subtraction operations, 6 multiplication operations, and 1 comparison operations. The column *Steps* gives the number of control steps. The column *Modules* gives 6-tuple ( $\#add$ ,  $\#sub$ ,  $\#mul\#$ ,  $\#div$ ,  $\#sel$ ,  $\#com$ ), where  $\#add$ ,  $\#sub$ ,  $\#mul\#$ ,  $\#div$ ,  $\#sel$ , and  $\#com$  are the numbers of adders, subtractors, multipliers, divisors, selectors, and comparators, respectively.

**Table 3. Characteristics of benchmark circuits.**

Circuit	Operations	Steps	Modules
HAL	(2, 2, 6, 0, 0, 1)	4	(2, 1, 4, 0, 0, 1)
BF	(12, 6, 11, 0, 0, 0)	8	(2, 2, 3, 0, 0, 0)
EF	(26, 0, 8, 0, 0, 0)	14	(4, 0, 2, 0, 0, 0)
DWT	(12, 0, 18, 0, 0, 0)	14	(2, 0, 2, 0, 0, 0)
IDCT2	(32, 20, 40, 4, 0, 0)	24	(6, 3, 7, 1, 0, 0)
DIST2	(256, 0, 64, 0, 3, 3)	86	(9, 0, 3, 0, 1, 1)

These benchmark circuits are targeted to TSMC 0.18 $\mu$ m process technology. Our experimental environment is established as below.

- (1) Each module is assumed to be 16-bit design. We use Synopsys DesignWare to implement the modules. For each module, its active power consumption and its idle power consumption (at a control step) are measured through Synopsys Hspice simulation.
- (2) The input transition time of the clock source is assumed to be 100 ps. The clock rate is assumed to be 500 MHz.
- (3) We use the cell AND2X1 to implement the clock gates. According to the *H*-tree structure, we can estimate the wire capacitances of the clock gates. Then, for each clock gate, based on the wire capacitance, we use Synopsys Hspice to derive its fresh gate delay and its power consumption at an active control step. Further, we use Mentor Eldo to measure the aged gate delays of clock gates (by considering the hot carrier effect). The duration of circuit operation is assumed to be ten years.
- (4) We use Synopsys Hspice to measure the power consumptions of transitions in the clock control logics of a real design. Then, we derive the average value. We use this average value to estimate the power consumption of a transition in the clock control logic.

For comparisons, we implement the following four methods.

- (1) *Method-A* (i.e., the approach proposed in [3, 4]). We use the left edge algorithm [21] to derive the module binding solution. Then, a post-processing is performed to balance the active probabilities of modules.<sup>6</sup> Finally, we use the approach proposed in [3, 4] to derive the gated clock tree construction in RTL synthesis stage. Note the approach proposed in [3, 4] does not take the aging skew into account.
- (2) *Method-B* (i.e., our RTL-AACG). We use the left edge algorithm [21] to derive the module binding solution. Then, a post-processing is performed to balance the active probabilities of modules. Finally, we use our RTL-AACG approach to derive the gated clock tree construction.
- (3) *Method-C* (i.e., our HLS-AACG). We use our HLS-AACG approach to derive the module binding solution and the gated clock tree construction.
- (4) *Method-D* (i.e., the approach proposed in [5]). We use the approach proposed in [5] to derive the module binding solution and the gated clock tree construction in the high-level synthesis stage. Note the approach proposed in [5] does not take the aging skew into account.

It should be mentioned: for the same benchmark circuit, these four methods have the same design skew and the same process skew, since they are based on the same  $H$ -tree structure.

## 7.2 Comparisons with Method-A

In this subsection, we compare Method-B (i.e., our RTL-AACG) and Method-C (i.e., our HLS-AACG) with Method-A (i.e., the approach proposed in [3, 4]). Table 4 tabulates the aging skews of these three methods. The column  $PB$  denotes the power consumption obtained by Method-A. Under the constraint that the power consumption does not exceed the value of  $PB$ , Method-B and Method-C are applied. The columns  $M-A$ ,  $M-B$ , and  $M-C$  denote Method-A, Method-B, and Method-C, respectively. The column *Improvement* denotes the relative improvement over Method-A. We find: the average improvement of Method-B over Method-A is 18.1%, and the average improvement of Method-C over Method-A is 25.9%.

**Table 4. Comparisons on the aging skews among Method-A, Method-B, and Method-C.**

Circuit	PB (mW)	Aging Skew (ps)			Improvement (%)	
		M-A	M-B	M-C	M-B	M-C
HAL	16.824	9.69	8.31	7.96	14.2	17.9
BF	16.351	39.86	31.59	29.40	20.7	26.2
EF	7.923	41.03	36.11	31.62	12.0	22.9
DWT	13.809	24.39	17.46	17.37	28.41	28.8
IDCT2	21.831	71.14	58.22	49.78	18.16	30.0
DIST2	11.541	56.79	48.08	39.96	15.3	29.6

<sup>6</sup> Intuitively, if the active probabilities of modules are close, the aging skew are easier to control. However, the left edge algorithm [21] may result in a large difference in the active probabilities of modules. Therefore, here, we perform a post-processing in order to reduce the difference. Our post-processing is a greedy algorithm as below. Initially, we unmark each operation. Then, we enter an iterative process. In each loop of the iterative process, we perform the following two actions. First, we find the module  $i$  that has the largest active probability; Second, we try to move an unmarked operation from the module  $i$  to another module that can also execute it but has a smaller active probability. Note, whenever an operation is moved, it is marked (in other words, each operation can be moved only one time). The iterative process repeats until no unmarked operation can be moved.

### 7.3 Comparisons with Method-D

Since both Method-C (*i.e.*, our HLS-AACG) and Method-D (*i.e.*, the approach proposed in [5]) synthesize the clock gating in the high-level synthesis stage, this subsection compares these two methods. Table 5 tabulates the aging skews of these two methods. The column *PB* denotes the power consumption obtained by Method-D. Under the constraint that the power consumption does not exceed the value of *PB*, Method-C is applied. The column *M-D* denotes Method-D. The column *Improvement* denotes the relative improvement of Method-C over Method-D. We find the average improvement is 23.7%.

**Table 5. Comparisons on the aging skews of Method-C and Method-D.**

Circuit	PB (mW)	Aging Skew (ps)		Improvement (%)
		M-C	M-D	
HAL	15.224	7.84	9.82	20.2
BF	15.708	29.02	37.04	21.7
EF	7.440	32.04	36.84	13.0
DWT	12.053	17.62	27.03	34.8
IDCT2	19.828	49.23	69.12	28.8
DIST2	10.624	40.25	52.68	23.6

### 7.4 Comparisons between Method-B and Method-C

Under our clock tree model and delay degradation model, if there is no clock gating (*i.e.*, the active probability of each clock gate is 100%), the aging skew is definitely to be zero. Therefore, each benchmark circuit exists at least one zero-aging-skew activity driven clock tree solution. Based on that observation, we can apply Method-B (*i.e.*, our RTL-AACG) and Method-C (*i.e.*, our HLS-AACG) to minimize the power consumption under the constraint that the aging skew is zero. In this subsection, we make comparisons between Method-B and Method-C.

Table 6 tabulates the power consumptions of Method-B and Method-C. In the second column, we report the power consumption of the clock tree (*i.e.*, the summation of the power consumption of modules, the power consumption of clock gates, and the power consumption of clock control logics). The column *Imp* denotes the relative improvement of Method-C over Method-B. It should also be mentioned that: for the same benchmark circuit, no matter which method is applied, the power consumption of modules is almost the same value. Therefore, to effectively judge the improvement on the power consumption, in the third column, we report the value that excludes the power consumption of modules. With an analysis, we find: if the power consumption of modules is included, the average improvement of Method-C over Method-B is 5.4%; if the power consumption of modules is excluded, the average improvement of Method-C over Method-B is 16.0%.

### 7.5 Comparisons on the CPU Times

Table 7 tabulates the CPU times of these four methods. Note, under different constraints on the power consumptions, the CPU times of the same circuit may be slightly

**Table 6. Comparisons on the power consumptions of Method-B and Method-C.**

Circuit	Power Consumption (Modules + Clock Gates + Clock Control Logics)			Power Consumption (Clock Gates + Clock Control Logics)		
	M-B (mW)	M-C (mW)	Imp (%)	M-B (mW)	M-C (mW)	Imp (%)
HAL	17.271	17.266	0.0	2.567	2.563	0.2
BF	17.093	16.929	0.9	3.093	2.929	5.3
EF	8.953	8.221	8.2	2.775	2.042	26.4
DWT	14.107	13.945	1.1	1.622	1.460	10.0
IDCT2	26.101	24.235	7.1	8.043	6.177	23.2
DIST2	16.088	13.689	14.9	7.787	5.388	30.8

**Table 7. Comparisons on the CPU times of these four methods.**

Circuit	M-A (sec)	M-B (sec)	M-C (sec)	M-D (sec)
HAL	2	5	156	72
BF	21	31	1781	398
EF	38	52	1045	214
DWT	7	9	697	151
IDCT2	330	551	15647	3645
DIST2	442	890	26324	7069

different. Thus, in Table 7, we report the average values. We find: since Method-A and Method-B synthesize the clock gating in the RTL synthesis stage, their CPU times are smaller; on the other hand, since Method-C and Method-D synthesizes the clock gating in the high-level synthesis stage, their CPU times are larger (due to a larger solution spaces). For Method-A and Method-B, even in large circuits (*e.g.*, IDCT2 and DIST2), the CPU times are still within fifteen minutes; on the other hand, for Method-C and Method-D, in large circuits, the CPU times are hours.

## 8. CONCLUSIONS AND FUTURE WORKS

This paper presents the first attempt for the synthesis of gated clock designs with the aging skew considered. Our work includes two aspects: RTL-AACG and HLS-AACG. Compared with the approach proposed in [3, 4], our RTL-AACG and HLS-AACG can reduce 18.1% and 25.9% of the aging skew, respectively, without any penalty on the power consumption. Compared with the approach proposed in [5], our HLS-AACG can reduce 23.7% of the aging skew without any penalty on the power consumption.

We leave the following two topics as our future work.

- (1) To estimate the delays and power consumptions of the clock tree in the high-level synthesis stage, our design methodology has made many assumptions: the layout of the clock tree must be an *H*-tree structure; the topology of the clock tree must be a binary clock tree structure; and so on. These assumptions greatly limit the practical use of our design methodology. Thus, there is a demand to develop a more practical design methodology.

- (2) Instead of eliminating the clock skew, the clock skew can be utilized as a manageable resource to improve the circuit performance [22, 23]. However, the synthesis of non-zero clock skew circuits requires not only the aged clock path delay but also the aged combinational path delay. So far, our design methodology only can estimate the aged clock path delay. There is a demand to study the estimation of the aged combinational path delay.

## REFERENCES

1. A. D. Mehta, Y. P. Chen, N. Menezes, D. F. Wong, and L. T. Pileggi, "Clustering and load balancing for buffered clock tree synthesis," in *Proceedings of IEEE International Conference on Computer Design*, 1997, pp. 217-223.
2. G. Venkataraman, N. Jayakumar, J. Hu, P. Li, and S. Khatri, "Practical techniques to reduce skew and its variations in buffered clock networks," in *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, 2005, pp. 591-595.
3. G. E. Tellez, A. Farrahi, and M. Sarrafzadeh, "Activity driven clock design for low power circuits," in *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, 1995, pp. 62-65.
4. A. Farrahi, C. Chen, A. Srivastava, G. Tellez, and M. Sarrafzadeh, "Activity driven clock design," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Vol. 20, 2001, pp. 705-714.
5. C. H. Cheng, S. H. Huang, and W. P. Tu, "Module binding for low power clock gating," *IEICE Electronics Express*, Vol. 5, 2008, pp. 762-768.
6. Q. Wu, M. Pedram, and X. Wu, "Clock gating and its application to low power design of sequential circuits," *IEEE Transactions on Circuits and Systems – I: Fundamental, Theory, and Applications*, Vol. 47, 2000, pp. 415-420.
7. D. Garret, M. Stan, and A. Dean "Challenges in clock gating for a low power ASIC methodology," in *Proceedings of IEEE International Symposium on Low Power Electronics and Design*, 2002, pp. 176-181.
8. W. C. Chao and W. K. Mak, "Low power gated and buffer clock network construction," *ACM Transactions on Design Automation of Electronic Systems*, Vol. 13, 2008, article 20.
9. P. Yang and J. H. Chern, "Design for reliability: The major challenge for VLSI," *Proceedings of IEEE*, Vol. 81, 1993, pp. 730-744.
10. Y. Leblebici, "Design considerations of CMOS digital circuits with improved hot-carrier reliability," *IEEE Journal of Solid-State Circuits*, Vol. 31, 1996, pp. 1014-1024.
11. H. Yonezawa, J. Fang, Y. Kawakami, N. Iwanishi, L. Wu, A. H. Chen, N. Koike, P. Chen, C. S. Yeh, and Z. Liu, "Ratio based hot-carrier degradation modeling for aged timing simulation of millions of transistors digital circuits," *IEEE International Electron Devices Meeting Technical Digest*, 1998, pp. 93-96.
12. K. Roy and S. Prasad, "Logic synthesis for reliability: An early start to controlling electromigration and hot-carrier effects," *IEEE Transactions on Reliability*, Vol. 44, 1995, pp. 251-255.
13. T. Karnik, C. C. Teng, and S. M. Kang, "High-level hot carrier reliability-driven

- synthesis using macro-models,” in *Proceedings of IEEE Custom Integrated Circuits Conference*, 1995, pp. 65-68.
14. A. Dasgupta and R. Karri, “Hot-carrier reliability enhancement via input reordering and transistor sizing,” in *Proceedings of IEEE/ACM Design Automation Conference*, 1996, pp. 819-824.
  15. C. W. Chang, K. Wang, and M. Marek-Sadowska, “Layout-driven hot-carrier degradation minimization using logic restructuring techniques,” in *Proceedings of IEEE/ACM Design Automation Conference*, 2001, pp. 97-102.
  16. C. W. Kang and M. Pedram, “Technology mapping for low leakage power and high speed with hot-carrier effect consideration,” in *Proceedings of IEEE/ACM Asia and South Pacific Design Automation Conference*, 2003, pp. 203-208.
  17. C. W. Kang and M. Pedram, “A leakage-aware low power technology mapping algorithm considering the hot-carrier effect,” *International Journal of Low Power Electronics*, Vol. 1, 2005, pp. 133-144.
  18. J. M. Cohn, J. L. P. C. Neves, and P. S. Zuchowski, “Method for reducing design effect of wearout mechanisms on signal skew in integrated circuit design,” United States Patent, Patent No. 6651230, 2003.
  19. S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, “NBTI-aware synthesis of digital circuits,” in *Proceedings of IEEE/ACM Design Automation Conference*, 2007, pp. 370-375.
  20. C. Lee, M. Potkonjak, and W. H. Maggione-Smith, “MediaBench: A tool for evaluating and synthesizing multimedia and communications systems,” in *Proceedings of IEEE International Symposium on Microarchitecture*, 1997, pp. 330-335.
  21. G. D. Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw Hill, New York, U.S.A., 1994.
  22. J. P. Fishburn, “Clock skew optimization,” *IEEE Transactions on Computers*, Vol. 39, 1990, pp. 945-951.
  23. S. H. Huang and Y. T. Nieh, “Synthesis of nonzero clock skew circuits,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 25, 2006, pp. 961-976.



**Shih-Hsu Huang (黃世旭)** received the B.S. degree in Computer Science and Information Engineering from National Chiao Tung University, Hsinchu, Taiwan, R.O.C., in 1989, the M.S. degree in Computer Science from National Tsing Hua University, Hsinchu, in 1991, and the Ph.D. degree in Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan, in 1995. From 1995 to 2000, he was with Computer and Communications Research Laboratories, Industrial Technology Research Institute, Hsinchu, rising to the position of deputy manager of IC design department, responsible for the design of high performance IC's. In 2000, he joined the department of Electronic Engineering, Chung Yuan Christian University, Chungli, Taiwan, as a faculty member, where he is currently a full Professor. His research interests include high-level synthesis, timing optimization, and physical design.



**Chun-Hua Cheng (程駿華)** received the B.S. degree in Electronic Engineering from Chun Yuan Christian University, Chungli, Taiwan, R.O.C., in 2003, and the M.S. degree in Electronic Engineering from Chung Yuan Christian University, Chungli, Taiwan, in 2005. He is presently working toward the Ph.D. degree in Electronic Engineering at Chung Yuan Christian University, Chungli, Taiwan. His research interests include timing optimization and high-level synthesis.



**Song-Bin Pan (潘松濱)** received the B.S. degree in Electronic Engineering from Chun Yuan Christian University, Chungli, Taiwan, in 2006, and the M.S. degree in Electronic Engineering from Chung Yuan Christian University, Chungli, Taiwan, in 2008. His research interests include power minimization and high-level synthesis.