A Self-Adjusting Clock Tree Architecture to Cope with Temperature Variations

Jieyi Long, Ja Chun Ku†, Seda Ogrenci Memik, and Yehea Ismail

Department of EECS, Northwestern University, Evanston, IL 60208, U.S.A.

{jlo198, seda, ismail}@ece.northwestern.edu, j-ku@northwestern.edu

***Abstract*-Ensuring resilience against environmental variations is becoming one of the great challenges of chip design. In this paper, we propose a self adjusting clock tree architecture, SACTA, to improve chip performance and reliability in the presence of on-chip temperature variations. SACTA performs temperature dependent dynamic clock skew scheduling to prevent timing violations in a pipelined circuit. We present an automatic temperature adjustable skew buffer design, which enables the adaptive feature of SACTA. Furthermore, we propose an efficient and general optimization framework to determine the configuration of these special delay elements. Experimental results show that a pipeline supported by SACTA is able to prevent thermal induced timing violations within a significantly larger range of operating temperatures (enhancing the violation-free range by as much as 45°C).**

I. INTRODUCTION

Steady miniaturization and large-scale integration lead to increasing power densities. As a result, on-chip temperatures are rising steadily as technology is scaling down. Making matters worse, power management techniques such as clock gating, voltage islands, and power gating may lead to drastic temporal and spatial variations of chip temperatures. In addition, a chip may be deployed in diverse environments where the level of cooling support as well as the nominal temperature could not be accurately predicted at the design time. Finally, power consumption of a chip can be highly input dependent, leading to variations in chip temperature throughout the execution of an application.

Temperature variation affects timing since interconnect resistance and cell delay are dependent on temperature. Both temporal and spatial temperature variations may cause significant changes in switching speed of gates and result in timing violations. This phenomenon further amplifies the challenges of chip design considering thermal effects. The idea of designing circuits with guaranteed performance bounds while exhibiting resilience against environmental variations arises as an attractive option.

In this paper, we introduce SACTA, a Self-Adjusting Clock Tree Architecture to address this problem. SACTA guarantees correct timing behavior in pipelined circuits within a large range of thermal conditions through a self-adjusting, temperature-sensitive skew distribution mechanism. SACTA exploits clock skews to maintain the performance in the presence of delay variations within pipeline stages. Adaptability is achieved by utilizing a set of special skew buffers. These elements are designed to exhibit carefully tuned temperature dependent delay behavior in synchronization with the temperature levels prevalent in the logic of the pipeline. Thereby, they generate a self-adjusting skew tailored to the temperature dependent timing behavior of each pipeline stage. We also developed a systematic design method to determine the physical specifications of these skew buffers (i.e. parameters that define their delay behavior) for a given pipelined circuit and a range of operating temperatures.

Our approach aims to provide a solution that can effectively avoid thermal induced delay violations considering the impact of temperature on both the clock tree and the datapath. We have evaluated the effectiveness of SACTA using a set of synchronous pipelined circuits. SACTA is able to prevent thermal induced timing violations within a significantly larger range of operating temperatures (enhancing the violation-free range by as much as 45°C). SACTA effectively enhances reliability while introducing negligible circuit level overhead.

The remainder of this paper is organized as follows. Section II provides an overview of related work. In Section III, our model of temperature dependent delay variation is presented, followed by the detailed discussion of the self-adjusting clock tree architecture. Design of our automatic temperature adjustable (ATA) skew buffers is described in Section IV. We introduce our clock tree design framework in Section V. Our experimental evaluation is presented in Section VI. We conclude with a summary of our findings in Section VII.

II. RELATED WORK

The increasing impact of temperature variation on circuit timing has motivated several techniques. Some effort has been devoted to developing temperature-insensitive zero/bounded skew clock trees. Dual-supply-voltage clock tree [1] utilizes the fact that CMOS gate delay is insensitive to temperature at a specific supply voltage level V_{ZTC}. V_{ZTC} is technology dependent and is at around 0.83V for 90nm. By supplying the on-tree buffers with this voltage, the clock tree is made to have zero skew for any given thermal profile. Although this technique can effectively eliminate delay fluctuations on the clock tree, *considering the clock tree alone is insufficient.* In a circuit where a zero-skew tree is used, to achieve maximal performance, T_{cp} is roughly set to the delay of the critical path. Since the delay of the critical path has positive dependence on temperature, as temperature increases, the delay of the critical path may exceed T_{cp} at a certain point. A zero-skew design will fail beyond this temperature. Also, it might at first appear

This work was supported by the National Science Foundation CAREER Award CNS-0546305 and NSF grant CCF-0541337.

†Now with Samsung Electronics Co., Ltd., Korea.

as a viable approach to supply the entire design with V_{ZTC} to achieve a system with absolute temperature independence. However, technology trends, particularly the relationship between supply level and the necessary threshold voltage levels for transistors, indicate that supply levels are unlikely to scaled beyond a certain level [2].

An alternative is to use clock skew to enhance immunity to temperature variation. An integer linear programming formulation for clock skew scheduling in the presence of process and environment variations has been proposed [3]. However, improvement in reliability was achieved at the expense of performance, making this approach less attractive to high performance circuit designers.

Lee et al. proposed a circuit-level timing error detection/ correction scheme [4]. The idea is to create clock skews dynamically such that those pipeline stages that require longer execution times due to environmental fluctuations are assigned longer intervals. However, compared to our scheme, this requires significantly more hardware resources (such as shadow registers and a central control unit for error monitoring and skew creation) and is able to provide only a discrete set of skew values (versus the continous scale generated by our scheme).

Finally, a large body of work addressed static clock skew scheduling [5-7]. These techniques do not attempt to solve the problem of delay variability, hence, our problem is fundamentally different.

III. The Self-Adjusting Clock Tree Architecture

Consider a local pipeline stage between two registers R_i and R_{i+1}. The following two constraints should be met to preserve correct circuit functionality [7]:

$$x_i + D_{i,i+1} \le x_{i+1} + T_{cp} \quad \text{and} \quad x_i + d_{i,i+1} \ge x_{i+1}$$

Here x_i and x_{i+1} are defined as the clock signal delays from the clock source to R_i and R_{i+1}. The difference between the arrival times of the clock signal at two successive registers is defined as the *clock skew*. It could be expressed as $(x_i - x_{i+1})$. $D_{i,i+1} = T_{c\text{-}q} + T_{logic(max)} + T_{setup}$, $d_{i,i+1} = T_{c\text{-}q} + T_{logic(min)} + T_{hold}$ signify the largest and shortest expected latencies for the pipeline stage located between these registers.

A *clock skew schedule* for a given pipeline is a set of delay values $\{x_i\}$ satisfying these constraints. Existing techniques only address *static clock skew schedule*. However, these constraints are in fact temperature dependent. Therefore, a given static clock skew schedule satisfying the constraints for some temperature profiles may fail for others, even if for these profiles, a static clock skew schedule does exist. This has motivated us to develop a dynamic clock skew scheduling scheme and the self-adjusting clock tree architecture.

On circuit layout, the logic gates and the registers in the same pipeline stage are normally placed in close proximity, since this will help shorten the critical path. Due to this spatial correlation, the temperature of the combinational logic and the associated pipeline registers are approximately the same. We use $\theta_{i,i+1}$ to denote this *local temperature* for the pipeline stage between registers R_i and R_{i+1}. Then, the maximum and minimum stage latencies $D_{i,i+1}$ and $d_{i,i+1}$ can be expressed as functions of the local temperature as $D_{i,i+1}(\theta_{i,i+1})$ and $d_{i,i+1}(\theta_{i,i+1})$.

Our goal is to design a self-adjusting clock tree that is able to adapt to different thermal profiles. In other words, for a given thermal profile, if there exists a static clock skew schedule to guarantee correct operation, our clock tree should be able to configure itself to provide this clock skew. To solve this problem we need to first understand the relationship between temperature and delay for circuit elements. In the following we present the temperature dependent delay model we have utilized to describe the behavior of logic.

A. Temperature Dependent Delay Model and Its Validation

Circuit delay of a CMOS gate can be written as

$$\tau \propto \frac{CV_{dd}}{I_d} \tag{1}$$

where C is the load capacitance driven by the gate, V_{dd} is the supply voltage (voltage swing), and I_d is the drain current of the transistors. The drain current stays mostly in the saturation region in deep submicron technologies due to velocity saturation [8]. Using the alpha-power law [8], the drain current in the saturation region is expressed as

$$I_d \propto \mu(\theta)\frac{W}{L}\left(V_{gs} - V_{th}(\theta)\right)^{\alpha} \tag{2}$$

θ denotes the gate temperature, μ is the carrier mobility, W and L are the channel width and length, respectively, V_{gs} is the gate-to-source voltage, V_{th} is the threshold voltage, and α is the velocity saturation index whose value is between 1 and 2 (closer to 1 in deep submicron technologies) [8]. Note that μ decreases as the temperature is raised, but its temperature dependence weakens to a linear relationship in the saturation region due to velocity saturation [9]. The temperature dependence of μ in the saturation region can be written as

$$\mu = \mu_0 - \eta(\theta - \theta_0) \tag{3}$$

where μ_0 is the mobility at the nominal temperature θ_0, which is typically 25ºC, and η is the temperature coefficient. V_{th} also decreases linearly as the temperature is raised, and is given by

$$V_{th} = V_{th0} - \kappa(\theta - \theta_0) \tag{4}$$

where V_{th0} is the threshold voltage at the nominal temperature, and κ is the temperature coefficient. Substituting (3) and (4) into (2) and (1), and taking a first order approximation, the temperature dependence of a CMOS gate delay can be simplified to a linear function of temperature [9]:

$$\tau = \tau_0 + k(\theta - \theta_0), \tag{5}$$

where τ_0 is the delay at the nominal temperature.

This linear model was validated against HSPICE using a chain of several different gates such as NAND, NOR, and XOR in the 90nm PTM technology [10]. The result is shown in Figure 1. It can be seen that there is an excellent agreement between the linear model and HSPICE.

Note that this linear temperature dependence of the delay is equally applicable to the combinational logic and the skew

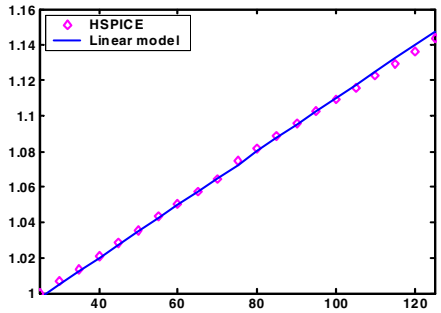

Figure 1. Validation of the linear model for the temperature dependence of gate delay against HSPICE.

buffers. In the local datapath of pipeline stages, the *RC* delay of the signal is dominated by the gate capacitances. We can safely ignore the share of the interconnect delay. As a result, we can assume the same temperature dependence for the combinational logic and the skew buffers.

B. Temperature Dependent Dynamic Clock Skew Scheduling

Our methodology to design the self-adjusting clock tree utilizes the linear model described in the previous section. Given this model, we formulate our problem as follows:

Problem 1. Thermal-aware dynamic clock skew scheduling: Given a pipeline driven by a clock with period time T_{cp} and the temperatures of the pipeline stages $\{\theta_{i,i+1} \mid \theta_{min} \le \theta_{i,i+1} \le \theta_{max}\}$, if the set of inequalities

$$-d_{i,i+1}(\theta_{i,i+1}) \le x_i - x_{i+1} \le T_{cp} - D_{i,i+1}(\theta_{i,i+1}) \qquad (6)$$

have a static solution $\{x_i\}_{expect}$, then, the self-adjusting clock tree should be able to adjust the actual arrival time $\{x_i\}$ to $\{x_i\}_{expect}$ in order to avoid circuit malfunction.

Figure 2 illustrates the timing constraints as functions of temperature. According to the analysis in Section IV.A, both $d_{i,i+1}(\theta_{i,i+1})$ and $D_{i,i+1}(\theta_{i,i+1})$ are linear functions of $\theta_{i,i+1}$. Therefore, the timing constraints can be represented with two lines as shown in Figure 2(a). Furthermore, the temperature range, where correct operation must be guaranteed, can be represented with two vertical lines. If static clock skew scheduling is used, the best we can do to cope with temperature variation is to set $(x_i - x_{i+1})$ equal to $-d_{i,i+1}(\theta_{min})$. However, as shown in Figure 2(a), timing violations can still occur even below the maximum operating temperature θ_{max}. If we can couple the value of $(x_i - x_{i+1})$ with $\theta_{i,i+1}$, with a linear function, then Constraint (6) may never be violated as long as the local temperature $\theta_{i,i+1}$ remains between θ_{min} and θ_{max}. An example of such a linear function is depicted in Figure 2(b). Now our problem becomes designing a clock tree that can supply dynamically changing skew values to pipeline registers, where the relationship between the skew value and the temperature in the region of interest should be in the form of a linear function. Note that we have established a linear relationship between the delay of a logic gate and temperature. We will take advantage of this result and employ special skew buffers in our clock tree architecture, which will render the temperature dependent behavior we desire.

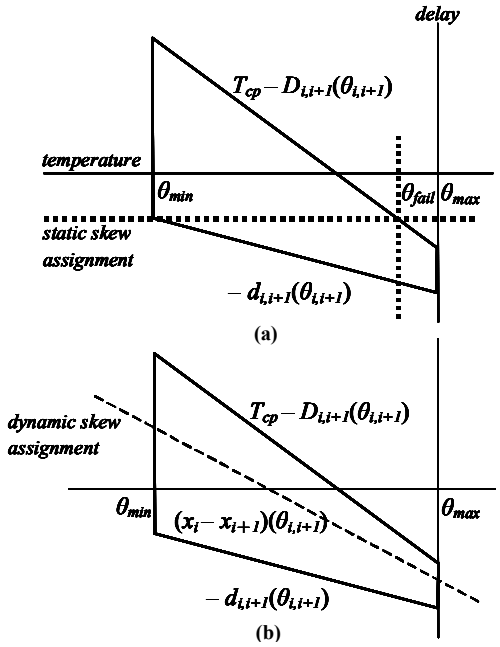

Figure 2. (a) Failing Temperature θ_{fail}; Timing violations may occur if θ_{fail} is exceeded. (b) Utilizing temperature-sensitive skew to improve system immunity against temperature variation.

C. Self-Adjusting Clock Tree Architecture

Figure 3 depicts a pipeline with our proposed Self-Adjusting Clock Tree Architecture (SACTA). The white triangles represent the Automatic Temperature Adjustable (ATA) skew buffers. The relationship between their delay and temperature is expressed as $s_i - k_i\Delta\theta$. s_i is the delay of the skew buffer at the worst case temperature θ_{max}. We refer to this delay value as the *base delay* of the ATA skew buffer. k_i is the temperature sensitivity coefficient. Here, $\Delta\theta$ is defined as $\theta_{max} - \theta$, i.e., the difference between θ_{max} and actual operating temperature in the vicinity of the skew buffer. The gray triangles represent the temperature-insensitive skew buffers (which will be refered to as fixed buffers) with *base delay* f_i. In this architecture, the skew of the i^{th} pipeline stage will be

$$x_i - x_{i+1} = f_i - f_{i+1} - s_i + k_i(\theta_{max} - \theta) \qquad (7)$$

In order to ensure that this function is linearly dependent on the local temperature $\theta_{i,i+1}$ of the pipeline stage i, all we need to do is to place the i^{th} ATA skew buffer close to the logic of the i^{th} pipeline stage on the circuit layout. Spatial correlation will enable the coupling between the temperature variable θ in Equation (7) and the local temperature $\theta_{i,i+1}$ of pipeline stage i. It can be easily seen that the purpose of the ATA buffers is to

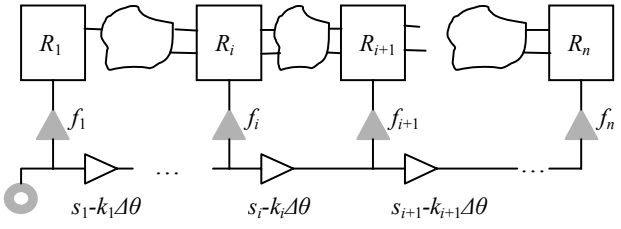

Figure 3. SACTA: Self-Adjusting Clock Tree Architecture.

generate the temperature-dependent coupling between pipeline latencies and the skew needed to ensure timing correctness. However, by employing ATA buffers alone we can only create negative skew between consecutive registers. Positive skew might also be needed. The purpose of the fixed buffers is to provide such positive skews.

D. Applications of SACTA and Its Limitations

With different temperature profiles the skew distribution of SACTA will clearly be different. As a result the total skew created within the pipeline (i.e., the sum of all skews across the stages) will be variable with temperature. This means, that there is not a single fixed total skew value between any two registers. In fact, this is the special property of SACTA that enables us to achieve adaptability. At the boundary case, this can be interpreted as having a variable non-zero skew between the input registers and the output registers.

For a one-dimensional linear pipeline this does not pose any limitations. On the other hand, if the pipeline is expected to communicate synchronously with another entity, then this might become an issue. In such a case, one possibility is to isolate the last pipeline stage from SACTA, thereby confining the end-to-end skew. In systems such as the Globally Asynchronous Locally Synchronous (GALS) system, this would not be needed. SACTA can be safely applied to the entire pipeline. Since for GALS inter-module communication is asynchronous, zero off-module clock skew is not required.

For those pipeline stages lying on the feedback loops, the ATA skew buffers of SACTA cannot completely cover the associated combinational logic along this loop. However, if those uncovered pipeline stages do not lie on the critical path, then this will not be crucial.

In the remainder of our discussions, each synchronous pipeline will be modeled as a one-dimensional pipeline corresponding to a submodule. This model is representative for a large number of synchronized circuits, such as various ASICs for signal processing and applications [11-13].

IV. SKEW BUFFER DESIGN

In this section, the designs of the fixed skew buffer and the ATA skew buffer are presented.

A. Temperature-Insensitive (Fixed) Skew Buffer

Temperature insensitive skew buffers can be designed by biasing the gate to the ZTC (Zero-Temperature-Coefficient) point. The ZTC point is a gate bias where the effect of change in the threshold voltage cancels out that of the mobility, making the drain current independent of temperature [14]. Thus, fixed skew buffers can be designed by using inverters whose gate-to-source voltage corresponds to the ZTC point. In order to bias the gate to the ZTC point, a voltage reference circuit that generates the ZTC voltage is needed. Figure 4(a) shows the voltage reference circuit using a transisor (*M*4) whose drain is connected to its gate and a current source. The connection between the drain and the gate ensure its operation to take place in the saturation region. The current equation in the saturation region in (2) can be rearranged as

$$V_{gs} = \sqrt[\alpha]{\frac{LI_d}{KW\mu}} + V_{th} \tag{8}$$

where K is a constant that is specific to a given technology. Hence, the current source can be used to provide the necessary amount of I_d to bias $M4$ to the ZTC point. The current source shown in Figure 4(a) consists of an inverter whose input and output are tied together and $M3$. By careful sizing of $M1$ and $M2$, the output of the inverter can be controlled to generate the ZTC voltage which, in turn, produces the required I_d for ZTC point through $M3$. Our HSPICE simulation showed that the ZTC voltage in the 90nm technology is 0.83V which places the PMOS transistor in the linear region, and the NMOS in the saturation region. Using the alpha-power law [8], the output voltage of the inverter is related to the sizing of $M1$ and $M2$ by

$$\frac{W_{M1}}{L_{M1}} K_{M1}\left(V_{out} - V_{thp}\right)^{\frac{\alpha}{2}} V_{out} = \frac{W_{M2}}{L_{M2}} K_{M2}\left(V_{out} - V_{thn}\right)^{\alpha} \tag{9}$$

where K_{M1} and K_{M2} are technology-specific constants. Note that using the inverter in Figure 4(a) alone to generate the ZTC voltage is unstable since any current flow from this source will result in fluctuations in the reference voltage. By using a chain of inverters whose supply rail is the ZTC voltage generated from the circuit as shown in Figure 4(b), the drain current of transistors becomes temperature-insensitive, thereby making the delay of the skew buffer independent of temperature variations as well.

In order to create a desired amount of delay for each buffer, a mixture of sizing and cascading of inverters is used. Larger L increases the delay, but inverters with L that is too large may not be able to generate full voltage swing at the output in a given clock cycle, possibly leading to functional errors especially after a chain of several inverters. Hence, $5L_{min}$ is used as the upper limit to avoid such errors for clock frequency of 3.0GHz in the 90nm technology, as determined from HSPICE simulation. To create more delay in a buffer, one can cascade more inverters as long as the correct logic level is preserved (i.e. even number of inverters). L of the cascaded inverters is varied between L_{min} and $5L_{min}$ except for the first and the last inverter, while W is kept at minimum size for all. Minimum size is used for the last inverter to recover fast transition time and full voltage swing of the signal before it reaches the register. Futhermore, the use of minimum size for the first inverter synchronizes and minimizes the load driven by the last inverter in the ATA skew buffer which will

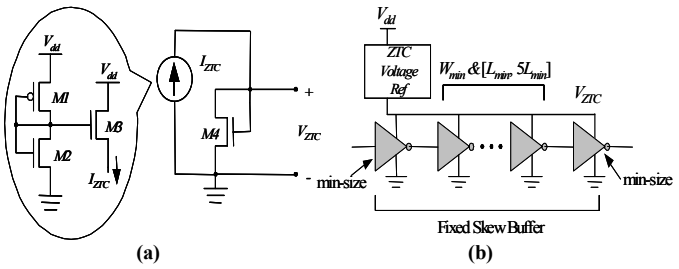

Figure 4. (a) Schematic of the ZTC voltage reference circuit (b) Design of the fixed skew buffer using the ZTC voltage reference.

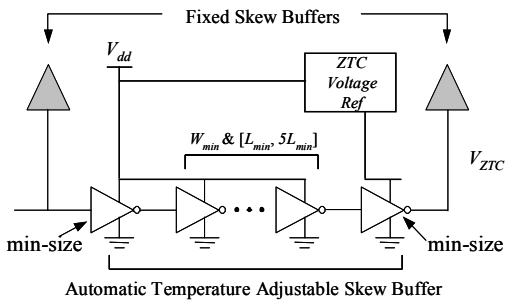

Figure 5. Design of the ATA skew buffer.

be discussed in the subsequent section. Besides the first and the last inverters, there is a degree of freedom for the number of inverters and their channel lengths.

B. Automatic Temperature Adjustable (ATA) Skew Buffer

The temperature-sensitivity of skew buffers can also be adjusted by a mixture of sizing and cascading of inverters. From (1) through (4), the temperature sensitivity of the gate delay (i.e. the temperature coefficient k in (5)) is related to gate sizing by

$$k = \left.\frac{\partial \tau}{\partial \theta}\right|_{\theta=\theta_0} \propto \left.\frac{CV_{dd}L\left[\eta\left(V_{gs}-V_{th}(\theta)\right)^{\alpha}-\alpha\kappa\mu(\theta)\left(V_{gs}-V_{th}(\theta)\right)^{\alpha-1}\right]}{W\mu(\theta)^2\left(V_{gs}-V_{th}(\theta)\right)^{2\alpha}}\right|_{\theta=\theta_0} \quad (10)$$

As indicated by (10), increasing L amplifies the temperature sensitivity, which results in a longer delay as well. Furthermore, when several inverters are cascaded together, increasing L of an inverter also increases its gate capacitance, which increases the delay and the temperature sensitivity of the previous inverter (increase in C in (10)). The increase in the absolute value of the delay is not a problem in the clock distribution network as long as the relative arrival times of the clock signals are well controlled.

Figure 5 shows the design of the ATA skew buffer, connected to the fixed skew buffers described in the preceding section. The ATA skew buffer consists of a chain of cascaded inverters. The supply rail of the last inverter is connected to the ZTC voltage instead of V_{dd} (1.3V) because its output is connected to the gate of the fixed skew buffer, which need to be biased to the ZTC voltage. The first inverter in the next ATA skew buffer then converts the voltage swing back to 1.3V. Minimum W is used for all the inverters in the chain since increased width reduces the temperature sensitivity. As for L, a range between L_{min} and $5L_{min}$ is used for all the inverters in the chain except for the first and the last inverter (L_{min} is used for the first and the last inverter for the same reasons explained in the preceding section).

V. Systematic Design Framework for the Clock Tree

So far, we have established the models governing the relationship between temperature and delay. Furthermore, we demonstrated that we can design delay elements to achieve the delay adjustment. Now, we need a systematic approach to determine the physical specifications of the buffer elements in SACTA in order to be able to design the clock tree for a given circuit. In this section, we present a network flow based method to determine the skew buffer configurations.

A. Linking Circuit-Level Aspects with the Design Framework

Various physical aspects of the skew buffers have a direct impact on the formulation of the optimization framework. The first significant phenomenon is that the base delay of a ATA skew buffer is proportional to its temperature sensitivity coefficient. In fact, according to (1~2), for a skew buffer consisting of m inverters,

$$\begin{aligned} k &= \left.\frac{\partial \tau}{\partial \theta}\right|_{\theta=\theta_0} = \sum_{i=1}^{m}\left.\frac{\partial \tau_i}{\partial \theta}\right|_{\theta=\theta_0} \\ &\propto \sum_{i=1}^{m}\frac{C_i V_{dd}}{\frac{W_i}{L_i}\mu(\theta_{\max})\left(V_{gs}-V_{th}(\theta_{\max})\right)^{\alpha}}\frac{\partial}{\partial\theta}\left.\frac{\mu(\theta_{\max})\left(V_{gs}-V_{th}(\theta_{\max})\right)^{\alpha}}{\mu(\theta)\left(V_{gs}-V_{th}(\theta)\right)^{\alpha}}\right|_{\theta=\theta_0} \\ &= \frac{\partial}{\partial\theta}\left.\frac{\mu(\theta_{\max})\left(V_{gs}-V_{th}(\theta_{\max})\right)^{\alpha}}{\mu(\theta)\left(V_{gs}-V_{th}(\theta)\right)^{\alpha}}\right|_{\theta=\theta_0}\tau(\theta_{\max}) \end{aligned}$$

which means k is proportional to $\tau(\theta_{max})$. Hence, we can relate the temperature sensitivity coefficient and the base delay by

$$k = \lambda\tau(\theta_{\max}) \quad (11)$$

Therefore, our optimization scheme cannot assume these parameters as independent. As a result, this relationship must be accounted for in the form of a constraint in our optimization framework (see Constraint (18)).

Another observation is that the base delays of both types of skew buffers cannot be made arbitrarily small. As discussed in Section IV, both types of skew buffers contain at least a head and a tail minimum sized inverters. Therefore, their minimal achievable base delays will be bounded by the sum of the delays of these two inverters. These constraints can be expressed with the following:

$$s \geq s_{min},\ f \geq f_{min} \quad (12)$$

Our optimization objective will be a linear combination of the skew buffer base delays. Various properties of SACTA can be represented with this generalized function. Particularly, the overhead of SACTA, i.e., the number of inverters used to implement the skew buffers can be represented with this function. According to the discussion in Section IV, for both types of buffers, the only way to obtain large base delays is to size up L. However, there is an upper bound on this sizing ($5L_{min}$). Hence, to achieve a given base delay value, there is a minimal number of inverters required. Therefore, a large value of f_i or s_i corresponds to a higher number of inverters. Thus, we can use the summation of the base delays of all buffers as a metric for our optimization framework.

This objective function can be used to represent other design metrics as well. For example, the base delay is related to power. A skew buffer with longer base delay consumes larger amount of power, since sizing up L increases the active area, resulting in larger dynamic power consumption. The base delay values f_i and s_i can also be related to the susceptibility of the skew buffers towards process variation. The magnitude of f_i and s_i are positively related with the length L of the

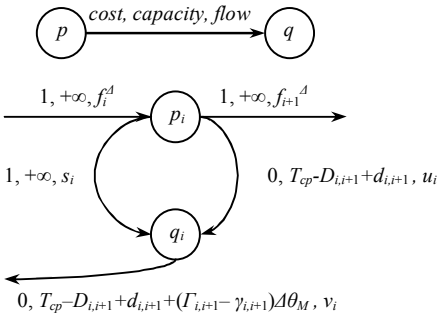

Figure 6. Graph based depiction of the constraints (29~33).

transistors in the skew buffers. Larger values of L indicate less sensitivity towards lithography induced variations.

B. Clock Tree Optimization Framework

We formulate the problem of determining the physical delay parameters s_i and f_i as a generalized min-cost flow problem.

Problem 2. **Base delay calculation for skew buffers:** Given a pipeline driven by a clock with period time T_{cp}, determine the base delays of the skew buffers, such that their sum is minimum, while satisfying the setup and hold time constraints. This can be formulated as a linear programming problem:

$$\textit{Minimize} \quad \sum_{i=1}^{n} f_i + \sum_{i=1}^{n-1} s_i \tag{13}$$

$$s.t. \qquad f_i - s_i - f_{i+1} \le T_{cp} - D_{i,i+1} \tag{14}$$

$$f_i - s_i - f_{i+1} \ge - d_{i,i+1} \tag{15}$$

$$f_i - s_i + k_i \Delta\theta_M - f_{i+1} \le T_{cp} - D_{i,i+1} + \Gamma_{i,i+1}\Delta\theta_M \tag{16}$$

$$f_i - s_i + k_i \Delta\theta_M - f_{i+1} \ge - d_{i,i+1} + \gamma_{i,i+1}\Delta\theta_M \tag{17}$$

$$k_i - \lambda\, s_i = 0 \tag{18}$$

$$s_i \ge s_{min},\ f_i, f_{i+1} \ge f_{min} \tag{19}$$

$$i = 1, 2, \ldots, n\text{-}1 \tag{20}$$

Here f_i, s_i, $D_{i,i+1}$, $d_{i,i+1}$ are the delay values at θ_{max}. $\Delta\theta_M$ denotes $\theta_{max} - \theta_{min}$, the gap between the worst-case temperature and minimum temperature. $\Gamma_{i,i+1}$ and $\gamma_{i,i+1}$ denote the temperature sensitivity coefficient of the longest and shortest combinational paths of the i^{th} pipeline stage. Constraints (14~17) are derived from (6) and (7). They guarantee that the line $(x_i - x_{i+1})(\theta_{i,i+1})$ will conform strictly to the timing constraints (as depicted in Figure 2).

Next, we will show that this constraint set has a special structure, which enables us to use a generalized min-cost flow based algorithm to solve it optimally in polynomial time. Simple transformations on the constraints will help reveal this special structure. First, substituting (18) into (16~17) yields

$$f_i - s_i\,(1-\lambda\,\Delta\theta_M) - f_{i+1} \le T_{cp} - D_{i,i+1} + \Gamma_{i,i+1}\Delta\theta_M \tag{21}$$

$$f_i - s_i\,(1-\lambda\,\Delta\theta_M) - f_{i+1} \ge - d_{i,i+1} + \gamma_{i,i+1}\Delta\theta_M \tag{22}$$

Defining new variables $f_i^{\Delta} = f_i - f_{min}$, $s_i^{\Delta} = s_i - s_{min}$, $u_i = f_i - s_i - f_{i+1} + d_{i,i+1}$, and $v_i = f_i - s_i\,(1-\lambda\,\Delta\theta_M) - f_{i+1} + d_{i,i+1} - \gamma_{i,i+1}\Delta\theta_M$, constraints (14~15) and (21~22) can be rewritten as

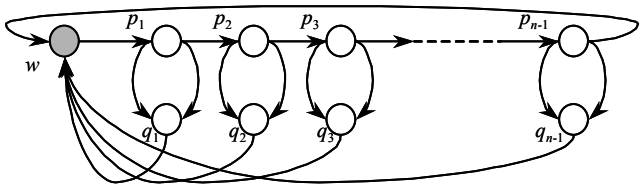

Figure 7. Graph based depiction of the constraints (29~34).

$$- f_i^{\Delta} + s_i^{\Delta} + f_{i+1}^{\Delta} + u_i = d_{i,i+1} + s_{min} \tag{23}$$

$$-f_i^{\Delta} + (1-\lambda\Delta\theta_M)s_i^{\Delta} + f_{i+1}^{\Delta} + v_i = d_{i,i+1} - \gamma_{i,i+1}\Delta\theta_M + (1-\lambda\Delta\theta_M)s_{min} \tag{24}$$

$$0 \le u_i \le T_{cp} - D_{i,i+1} + d_{i,i+1} \tag{25}$$

$$0 \le v_i \le T_{cp} - D_{i,i+1} + d_{i,i+1} + (\Gamma_{i,i+1} - \gamma_{i,i+1})\Delta\theta_M \tag{26}$$

Constraints (23) and (24) give us

$$-(\lambda\,\Delta\theta_M)\,s_i^{\Delta} - u_i + v_i = -\gamma_{i,i+1}\Delta\theta_M - (\lambda\,\Delta\theta_M)\,s_{min} \tag{27}$$

Equivalently, constraint (24) can be replaced by constraint (27). The above transformation yields a new formulation:

$$\textit{Minimize} \quad \sum_{i=1}^{n} f_i^{\Delta} + \sum_{i=1}^{n-1} s_i^{\Delta} \tag{28}$$

$$s.t. \qquad - f_i^{\Delta} + s_i^{\Delta} + f_{i+1}^{\Delta} + u_i = d_{i,i+1} + s_{min} \tag{29}$$

$$-(\lambda\,\Delta\theta_M)\,s_i^{\Delta} - u_i + v_i = -\gamma_{i,i+1}\Delta\theta_M - (\lambda\,\Delta\theta_M)\,s_{min} \tag{30}$$

$$0 \le u_i \le T_{cp} - D_{i,i+1} + d_{i,i+1} \tag{31}$$

$$0 \le v_i \le T_{cp} - D_{i,i+1} + d_{i,i+1} + (\Gamma_{i,i+1} - \gamma_{i,i+1})\Delta\theta_M \tag{32}$$

$$s_i^{\Delta}, f_i^{\Delta}, f_{i+1}^{\Delta} \ge 0 \tag{33}$$

$$i = 1, 2, \ldots, n\text{-}1 \tag{34}$$

Note that the cost functions (13) and (28) only differ by a constant. Therefore the new optimization problem (28~34) is equivalent to the original one. This formulation is a generalized min-cost flow formulation. Variables s_i^{Δ} and f_i^{Δ} can be any real number between 0 and +∞. They can be viewed as flows on directed edges, each having a capacity of +∞. Also, according to the objective function given in Expression (28), each of these edges should be associated with cost 1. Likewise, each u_i (or v_i) can be modeled as a flow on a directed edge with cost 0 and having capacity of $T_{cp} - D_{i,i+1} + d_{i,i+1}$ (or $T_{cp} - D_{i,i+1} + d_{i,i+1} + (\Gamma_{i,i+1} - \gamma_{i,i+1})\Delta\theta_M$). Equation (29) (or (30)), which has the form of flow conservation condition, can be modeled as four (or three) directed edges intersecting at a node with the balance expressed as

$$p_i = d_{i,i+1} + s_{min} \ \ (\text{or } q_i = -\gamma_{i,i+1}\Delta\theta_M - (\lambda\,\Delta\theta_M)\,s_{min}) \tag{35}$$

This is also depicted in Figure 6. Note that in constraint (30), the coefficient of s_i^{Δ} is not 1. However, this kind of constraints can still be handled using generalized min-cost flow algorithms [15].

Based on the graph representation of the constraints (29~33), we can model constraints (29~34) as shown in Figure 7. In Figure 7, $p_1, p_2, \ldots, p_{n-1}$ and $q_1, q_2, \ldots, q_{n-1}$ (the white vertices) are the nodes representing the constraints (29~30) for all $i = 1, 2, \ldots, n\text{-}1$. Note that we add a gray node w whose balance is expressed as

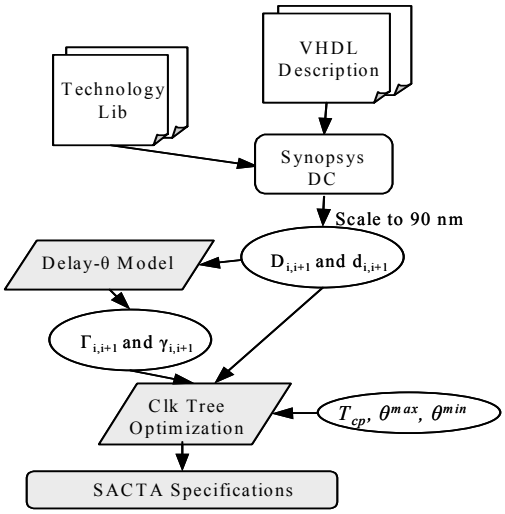

Figure 8. Experimental flow.

$$-\sum_{i=1}^{n-1}\left[\left(d_{i,i+1}-\gamma_{i,i+1}\Delta\theta_M\right)+\left(1-\lambda\Delta\theta_M\right)s_{min}\right]$$

ensuring the sum of the balances of all the vertices is zero. It is easy to prove that the constraints (29~34) have a feasible solution if and only if there is a feasible flow on this graph.

Problem 2 is then reduced to the problem of finding a feasible flow that minimizes cost function (28). There are several efficient generalized min-cost flow algorithms in the literature that guarantee polynomial running time [16].

The generalized min-cost flow based algorithm determines the base delay value for each skew buffer while minimizing the total number of inverters used to construct the skew buffers. An inverter chain for each skew buffer is then constructed according to the base delay value for that buffer. In each chain, we set the length of all inverters in the chain except the last one to be $5L_{min}$, and finally, we scale the length of the last inverter to the proper value to provide the required base delay.

VI. Experimental Results

In this section we first describe our experimental flow. Next, we present our results demonstrating the effectiveness of our proposed self-adjusting clock tree architecture.

A. *Experimental Setup*

Figure 8 illustrates our experimental flow. We use Synopsys Design Compiler to synthesize the benchmarks onto the TSMC 180-nm technology library. Design Compiler reports the longest and shortest paths for each pipeline stage in the benchmarks. The delay values are then scaled for 90-nm technology. Also, based on our validated Delay-θ model at 90 nm, the temperature sensitivity of the longest/shortest paths, i.e. the Γ and γ values for each pipeline stage can be determined. Then, we feed the $D_{i,\,i+1}$, $d_{i,\,i+1}$, $\Gamma_{i,\,i+1}$ and $\gamma_{i,\,i+1}$ values, as well as the given T_{cp} into our clock tree optimization framework. The other two parameters for the optimization algorithm, i.e., the boundaries of the operating temperature range, are set to 25°C and 125°C. The optimization subroutine determines the appropriate buffer parameters, which are further used as the guidelines for finalizing the skew buffer design.

TABLE I. Pipeline Partition of the Benchmarks.

Circuit \ Partition	Balanced					Unbalanced				
	s1	s2	s3	s4	s5	s1	s2	s3	s4	s5
PolyEval	2	2	2	2	2	1	3	1	3	2
RSDecoder	2	2	2	2	2	2	3	1	1	3
FiniteFieldMult	2	2	2	2	2	1	3	1	2	3

TABLE II. Clock Period for the Pipelines.

	PB	PU	RB	RU	FB	FU
T_{cp}/ps	221	318	1331	1983	210	301

We have used a benchmark set consisting of systolic array circuits. This set includes a polynomial expression evaluator [11], the Reed-Solomon decoder [13], and a fast digital-serial multiplier for finite field [12]. These circuits share a common structure. A set of Processing Elements (PEs) are connected in series. In order to improve system throughput, the circuits are pipelined by inserting registers between the PEs. In our experiments, we divide each benchmark into five pipeline stages. To better evaluate our technique, we allow both balanced and unbalanced pipeline partitions. The pipeline partitions are summarized in Table I. The columns s1-s5 denote the number of PEs in each pipeline stage. For example, the unbalanced-partitioned benchmark PolyEval, contains 1, 3, 1, 3, and 2 PEs, in pipeline stages s1 through s5, respectively.

B. *Experimental Results*

Our first set of results, depicted in Figure 9, presents the maximum operating temperature with guaranteed timing correctness for all the pipelines under spatially uniform temperature distribution. The required clock periods for the pipelines are given in Table II. PB (PU) denotes balanced (unbalanced) pipeline of benchmark PolyEval; RB (RU) represents balanced (unbalanced) pipeline of benchmark RSDecoder; and FB (FU) stands for balanced (unbalanced) pipeline of benchmark FiniteFiledMult.

First, we evaluate SACTA's resilience against temperature variations. We compare two cases: SACTA versus utilizing static clock skew scheduling in the pipeline designs. For each pipeline, we keep increasing its operating temperature until the timing constraint is violated. This temperature is recorded as the maximum tolerable temperature for that pipeline under the given clock period constraint.

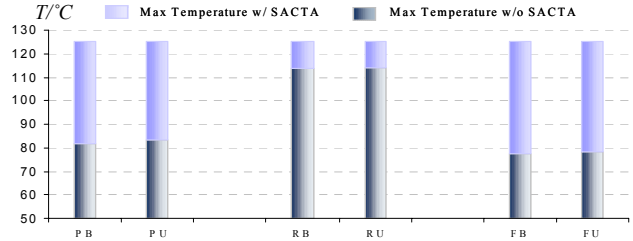

Figure 9. Maximum permissible temperature.

TABLE III. EXPERIMENTAL RESULTS WITH DIFFERENT THERMAL PROFILES.

Thermal Profiles/°C					Pipelines w/o SACTA						Pipelines w/ SACTA					
s1	*s2*	*s3*	*s4*	*s5*	*PB*	*PU*	*RB*	*RU*	*FB*	*FU*	*PB*	*PU*	*RB*	*RU*	*FB*	*FU*
125	115	110	107	105	X	X	X	X	X	X	√	√	√	√	√	√
105	107	110	115	125	X	X	X	X	X	X	√	√	√	√	√	√
100	105	110	105	100	X	X	√	√	X	X	√	√	√	√	√	√
110	105	100	105	110	X	X	√	√	X	X	√	√	√	√	√	√
135	125	120	117	115	X	X	X	X	X	X	X	√	X	√	X	√
115	117	120	125	135	X	X	X	X	X	X	X	√	X	X	X	X

We observe that using SACTA, the maximum tolerable temperature can be dramatically increased. All pipelines can function correctly until chip temperature attains the designed maximal value, 125°C. As a comparison, both balanced/unbalanced FiniteFieldMult pipelines without SACTA fail below 80°C, indicating an over 45°C increase in maximal tolerable temperature. Approximately the same improvement is observed for the benchmark PolyEval.

Next, we experiment with different thermal profiles. Table III presents our results, where "√" signifies that circuit functions correctly, and "X" indicates timing constraits are violated. Obviously examining all possible thermal profiles is impractical. Here, we only consider some representative profiles. The first (second) one represents a monotonically decreasing (increasing) profile across the pipeline stages. The third (fourth) profile exhibits a profile that is first increasing (decreasing) and then decreasing (increasing). Finally, the last two profiles represent monotonically decreasing and increasing profiles respectively, however, the maximum temperatures are above 125°C, the expected worst case operating temperature. The clock period times for the pipelines are set according to Table II. We observe that for the thermal profiles with peak temperature less than 125°C, no timing violation occurs with SACTA. Even for some profiles with higher maximal temperature, pipelines with SACTA can still work correctly. The pipelines without SACTA fail in most cases.

Our experimental results also indicate that SACTA can enhance system performance. In this experiment, the chip is assumed to uniformly execute at the worst case operating temperature 125°C. For the pipelines without SACTA, as mentioned in Section III.B, the best we can do to prevent thermal induced timing violation is to set the skew of the i^{th} pipeline stage to $-d_{i,i+1}(\theta_{min})$. For the pipelines with SACTA, we determine f_i and s_i using the clock tree optimization algorithm. We then keep increasing the clock frequency until a timing violation occurs. Figure 10 plots the relative performance gain. For each pipeline, the maximal achivable frequency without SACTA is normalized to 1. By employing SACTA, the maximum achievable clock frequency can be increased. For PolyEval and FiniteFieldMult, the relative improvements range from 7% to 9%.

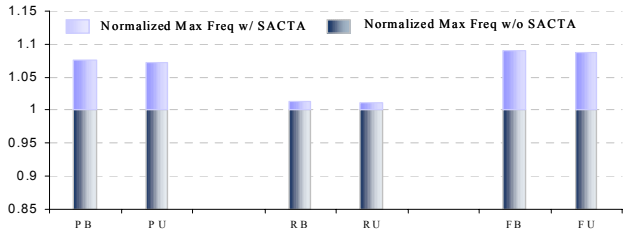

Figure 10. Relative performance improvement.

TABLE IV. OVERHEAD OF SACTA.

	PB	PU	RB	RU	FB	FU
On-Tree Inv Num	67	51	67	50	67	53
Pipeline Cell Num	2082	2082	1572	1572	498	498

Table IV reports the hardware overhead of SACTA, i.e., the number of inverters on the clock tree. We also report the number of standard cells in the original pipeline circuit generated by Design Compiler as a comparison. It is clear that the hardware overhead of SACTA is quite small.

VII. CONCLUSIONS

We have proposed SACTA, a self-adjusting clock tree architecture and a dynamic clock scheduling scheme to improve performance and reliability of pipelined circuits. We designed temperature adjustable skew buffers to create useful temperature dependent clock skews. We developed a two-step technique for design of a power optimized clock tree. Experimental results show that our scheme can dramatically improve the temperature tolerance. The increase in maximal tolerable temperature is by as much as 45°C.

REFERENCES

1. Bota, S.A., et al. *Within Die Thermal Gradient Impact on Clock Skew: A New Type of Delay-Fault Mechanism*. in *Int. Test Conf.* 2004.
2. Horowitz, M., et al. *Scaling, Power and the Future of CMOS*. in *Int. Electron Devices Meeting*. 2005.
3. Nawale, V., et al. *Optimal Useful Clock Skew Scheduling in the Presence of Variations Using Robust ILP Formulations*. in *Int. Conf. on Computer Aided Design*. 2006.
4. Lee, S., et al. *Reducing Pipeline Energy Demands with Local DVS and Dynamic Retiming*. in *Int. Symp. on Low Power Electronics and Design*. 2004.
5. Fishburn, J.P., *Clock Skew Optimization*. IEEE Trans. on Computers, 1990. **39**(7): p. 945-951.
6. Deoka, R.B., et al. *A Graph-Theoretic Approach to Clock Skew Optimization*. in *Int. Symp. on Circuits and Systems*. 1994.
7. Friedman, E.G., *Clock Distribution Networks in Synchronous Digital Integrated Circuits*. Proc. of the IEEE, 2001. **89**(5): p. 665-692.
8. Sakurai, T., et al., *Alpha-Power Law MOSFET Model and Its Application to CMOS Inverter Delay and Other Formulas*. IEEE Journal of Solid-State Circuits, 1990. **25**: p. 584-593.
9. Ku, J., et al., *Thermal-Aware Methodology for Repeater Insertion in Low-Power VLSI Circuits*. IEEE Trans. Very Large Scale Integration Systems, 2007. **15**(8).
10. Cao, Y., et al. *New Paradigm of Predictive MOSFET and Interconnect Modeling for Early Circuit Design*. in *Custom Integrated Circuit Conf.* 2000.
11. Mathias, P.C., et al., *Systolic Evaluation of Polynomial Expressions*. IEEE Trans. on Computers, 1990. **39**(5): p. 653-665.
12. Kim, C., et al. *A Fast Digit-Serial Systolic Multiplier for Finite Field GF(2m)*. in *Asia and South Pacific Design Automation Conf.* 2005.
13. Iwamura, K., et al., *A Design of Reed-Solomon Decoder with Systolic-Array Structure*. IEEE Trans. on Computers, 1995. **44**(1): p. 118-122.
14. Filanovsky, I.M., et al., *Compensation of Mobility and Threshold Voltage Temperature Effects with Application in CMOS Circuits*. IEEE Trans. on Circuit and Systems I, 2001. **48**(7): p. 876-884.
15. Ahuja, R.K., et al., *Network Flows: Theory, Algorithms, and Applications*. 1993: Prentice Hall, New Jersey.
16. Wayne, K.D. *A Polynomial Combinatorial Algorithm for Generalized Minimum Cost Flow*. in *Symp. on Theory of Computing*. 1999.