Bayesian Theory based Switching Probability Calculation Method of Critical Timing Path for On-Chip Timing Slack Monitoring

Byung Su Kim
Design Technology Team, Foundry,
Samsung Electronics, Korea
bs2014.kim@samsung.com

Joon-Sung Yang
Department of Semiconductor Systems Engineering,
Sungkyunkwan University, Suwon, Korea
js.yang@skku.edu

***Abstract*— Accurate in-situ monitoring is urgently required for an adaptive performance control system and post silicon validation. For accurate in-situ monitoring, a direct probing method is presented in which monitors directly measure a path delay from real critical timing paths. However, we may not be able to predict when the timing slack monitors would activate since the activation depends on a design structure and input patterns. If a timing slack monitor is rarely activated by timing critical paths, the observability from this monitor would be low and the monitor possibly can be discarded. For this reason, we propose a novel timing slack monitoring methodology based on switching probability of timing critical paths. Switching probability and correlation on critical timing paths are formulated, and the proposed method finds a list of critical path endpoints for the timing slack monitor insertion under given power and area constraints. Experimental results with ISCAS'89 circuits show that, compared to the method which places monitors for all worst critical paths, 16.67 ~ 97.2% of timing slack monitors are removed and 32.56 ~ 96.88% of dynamic power reduction from the monitors is achieved by the proposed method.**

Keywords*— *Timing Slack Monitor, Adaptive clocking

I. INTRODUCTION

Due to a process technology shrinkage, increasing transistor counts and ultra-low power design, an integrated circuit (IC) suffers from huge process, voltage and temperature variations and shows nonlinear operations. This makes a prediction of a worst chip performance very difficult. Therefore, wider design margins are required to assure reliable circuit operations. To avoid a design margin overhead, various adaptive performance control methods such as adaptive clocking [3] and resilient circuit method [8] are introduced. In the circuits, the information such as supply voltage fluctuations, process/ temperature variation and aging effect is collected by various in-situ monitors. The information is used to adaptively control the clock speed or voltage for avoiding timing failures [3-8]. To apply these methods, an accurate on-chip In-situ monitoring methodology is needed.

Various In-situ monitoring methods [1-8] are introduced. The delay monitor based methods [4-7] periodically measure a delay of internal delay chains (i.e., ring oscillator) or critical path replica circuits which are popular approaches since they are easy to use. However, because they are an indirect monitoring approach measuring delays from replica circuits or ring oscillators, it would be difficult to represent a real circuit performance [3] and to consider local process, voltage and temperature variations in the circuit. To overcome the problems, other approaches [1, 2] are proposed which place timing slack monitors directly on functional critical timing paths. Slackprobe [1] presents a modified min-cut algorithm that finds minimum uncorrelated intermediate nets for timing slack monitor insertion. Xie and Davoodi [2] find representative critical paths considering various variations. Since the timing slack monitor in [1, 2] is directly inserted on functional critical timing paths, they can accurately observe a path delay than indirectly monitoring the performance like [4-7]. Although these methods can accurately measure the circuit performance with a reduced number of monitors, the timing slack monitors on functional critical timing paths can detect the critical path delay only if a timing critical signal transition is activated. The monitors on critical paths with low switching probability can be considered as idle monitors since they are rarely or never activated. This means that the observability from the monitors with a low switching probability is poor. Hence, to efficiently capture the real critical path delay for adaptive performance control system [3,7], a timing slack monitor needs to be inserted where a switching probability of a critical path is high.

To measure critical path delay with a high observability, this paper proposes a novel timing slack monitoring methodology. The key contributions of the proposed method are summarized as follows.

1) A novel switching probability and correlation coefficient calculation method for timing critical paths is proposed as a timing arc based probability model.

2) A mathematical formation is given to determine a list of endpoints for a timing slack monitor insertion considering the observability (i.e., switching probability) and area/power budget. A solution which is a mixed integer quadratic constrained problem is converted to a dynamic programming based solution with Markov clustering.

Experimental results show that, compared to the method which places monitors for all worst critical paths, the proposed method significantly reduces the number of timing slack monitors (16.67~97.2% of monitors) by removing the monitors on critical paths with a low signal probability. This also achieves a considerable a dynamic power reduction (32.56 ~ 96.88% dynamic power reduction from monitors). The rest of the paper is organized as follows. Sec. II provides motivation. Sec. III explains a calculation of switching probability and correlation for critical path and endpoint. Sec. IV shows how an optimized set of monitors is selected. Sec. V gives the experimental evaluation and the conclusion is given in Sec. VI.

978-3-9819263-0-9/DATE18/©2018 EDAA

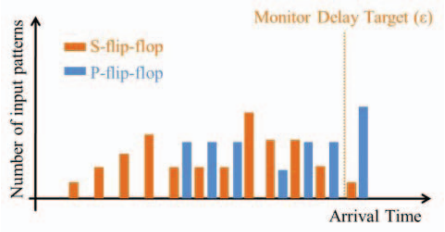

Figure 1.An arrival time histogram of two flip-flops, S and P

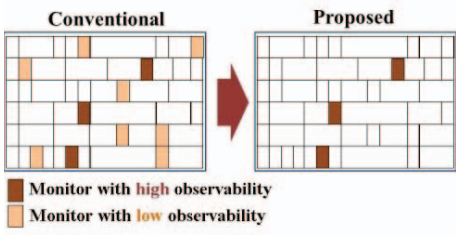

Figure 2.Objective of the proposed methodology

II. Motivation

The timing slack monitors on functional timing critical paths can only be activated when there is a signal transition. A cell on a clock network is switching every clock cycle while some cells related to chip reset or turn-on would hardly toggle. Hence, the monitors on functional critical paths with high switching rate would give more monitoring opportunities and this can be considered as highly efficient and observable timing slack monitoring. Fig. 1 shows arrival time measurements for two flip-flops, *S* and *P*, when inputs are applied. A *Y*-axis shows the number of input patterns and an *X*-axis denotes an arrival time corresponding to the input. A monitoring delay target (ε) is set and the inputs give longer arrival time than ε can activate the monitor. Assume that timing slack monitors are inserted for *S* and *P*. For flip-flop *P*, there are more inputs exercising the monitor than *S*. Since *S* has a lower switching probability than *P*, the timing slack monitor for *S* needs to spend more time for activation. Hence, we can consider that *P* has higher observability than *S*. With a given area and power budget, it is critical to decide a set of end points to insert timing slack monitors that give high observability and low switching activity of each end point. As Fig. 2 shows, this paper tries to reduce the number of monitors considering switching probabilities of critical timing paths. Hence, we propose a novel method to calculate switching probability and correlation of critical paths for efficient timing slack monitoring methodology.

III. Switching Probability Calculation Method

In this section, a method to calculate switching probability and correlation calculation is discussed. We assume that each endpoint of critical path is a flip-flop or latch, and a timing slack monitor can be inserted at the endpoints. Sec. III.A describes a switching probability calculation of critical paths at each endpoint and Sec. III.B describes a correlation calculation among each endpoints.

A. Timing Arc Based Switching Probability Model

For a given timing path, a net is denoted as $n_{i,s}$ where *i* is a net number and *s* is a state of the net. The net state can transit from *0* to *1*, from *1* to *0* or keep its current state. This can be expressed as

$$n_{i,s} = \{n_{i,0\to0}, n_{i,0\to1}, n_{i,1\to1}, n_{i,1\to0}\} \quad 1)$$

, when don't care (*X*) is not considered. $n_{i,s}$ can be one of four events and its switching probability is denoted as $\Pr(n_{i,s})$. N_i is defined as an input net of an endpoint such as flip-flop or latch. In this paper, N_i is an endpoint and it can be expresses as follows.

$$N_i = n_{i,0\to1} \cup n_{i,1\to0}$$
$$\Pr(N_i) = \Pr(n_{i,1\to0} \cup n_{i,0\to1}) \quad 2)$$

A switching probability of each timing path from $n_{1,s}$ to $n_{i,s}$ can be represented as (3) by net denotation.

$$\Pr(\text{Timing path from } n_{1,s_1} \text{ to } n_{i,s_i}) = \Pr(n_{i,s_i}, n_{i-1,s_{i-1}}, ..., n_{1,s_1}) \quad 3)$$

For power consideration, path balancing approaches are generally applied and this helps to reduce a glitch occurrence [13]. Hence, as a monitoring delay target (ε) becomes narrower, critical paths for one endpoint can be considered as mutually independent. This means that if one critical path is sensitized, other critical paths of this endpoint is not activated within our monitoring delay target. The sum of critical paths switching probability of N_i is denoted as follows.

$$\Pr(N_i, Delay(N_i) > \varepsilon) = \sum \Pr(n_{i,s_i}, n_{i-1,s_{i-1}}, ..., n_{1,s_1}) \quad 4)$$

In this paper, we omit *Delay* $(N_i) > \varepsilon$ since a worst speed (ε) is only considered. We can calculate this switching probability by checking all conditions of each timing path. However, to calculate $\Pr(n_{i,si}, n_{i-1,si-1}, \ldots, n_{1,s1})$, events from all nets need to be considered as a single event. As the number of critical paths grows, this becomes inefficient since the duplicated net events (i.e., a net event from one path occurs at other paths) have to be calculated repetitively. That is, the same net event occurring at many critical paths has to be calculated repetitively for each path calculation.

To overcome this problem, this paper proposes a novel switching probability calculation method which uses a timing arc based switching probability model. Firstly, the switching probability of (3) can be written by conditional probability as follows.

$$\Pr(n_{i,s_i}, n_{i-1,s_{i-1}}, ..., n_{1,s_1})$$
$$= \Pr(n_{i,s_i} \mid n_{i-1,s_{i-1}}, ..., n_{1,s_{i-1}}) \Pr(n_{i-1,s_{i-1}}, ..., n_{1,s_{i-1}}) \quad 5)$$

A signal transition of n_i is determined only by a directly connected input net. This means that, assuming a two input gate, the output net n_i is only dependent on its input nets, n_{i-1} and n_{i-2}, not on n_{i-3}~n_1 since a logic operation is only influenced by its relative inputs. Under this rule, (5) can be re-written as follows.

$$= \Pr(n_{i,s_i} \mid n_{i-1,s_{i-1}}, n_{i-2,s_{i-1}}) \Pr(n_{i-1,s_{i-1}}, n_{i-2,s_{i-2}}, n_{i-3,s_{i-3}}, ..., n_{1,s_1}) \quad 6)$$

In (6), the second term is changed by concepts of covariance and correlation as (7). This correlation ρ can be extracted by the methods in the next section. However, it has a high calculation complexity. We only calculate the correlation when a timing path has a self-looping net, otherwise, it is set to 1.

$$\Pr(n_{i-1,s_{i-1}}, n_{i-2,s_{i-2}}, n_{i-3,s_{i-3}}, ..., n_{1,s_1})$$
$$= \rho \cdot \Pr(n_{i-2,s_{i-2}}) \Pr(n_{i-1,s_{i-1}}, n_{i-3,s_{i-3}}, ..., n_{1,s_1}) \quad 7)$$

For simplicity, we assume that an input n_{i-2} is independent with other nets. The correlation between n_{i-2} and others is 1.

The intersection with n_{i-2} is can be changed as follows.

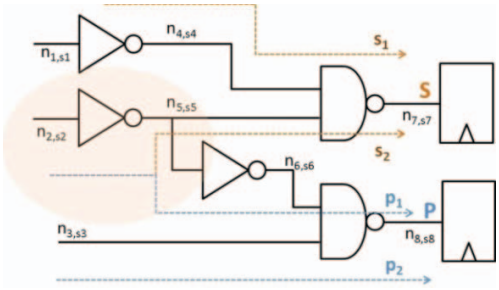

Figure 3.Example of a simple circuit with two monitors, S and P

$$= \Pr\left(n_{i,s_i} \mid n_{i-1,s_{i-1}}, n_{i-2,s_{i-1}}\right)\Pr\left(n_{i-2,s_{i-2}}\right)\Pr\left(n_{i-1,s_{i-1}}, n_{i-3,s_{i-3}}, \ldots, n_{1,s_1}\right) \quad 8)$$

(8) can be updated by Bayesian theory as,

$$\begin{aligned}
&\Pr(n_{i,s_i} \mid n_{i-1,s_{i-1}}, n_{i-2,s_{i-1}})\Pr(n_{i-2,s_{i-2}}) * \psi \\
&= \Pr(n_{i-1,s_{i-1}}, n_{i-2,s_{i-1}} \mid n_{i,0\to 1})\frac{\Pr(n_{i,s_i})}{\Pr(n_{i-1,s_{i-1}}, n_{i-2,s_{i-1}})}\Pr(n_{i-2,s_{i-2}}) * \psi \\
&= \Pr(n_{i-1,s_{i-1}}, n_{i-2,s_{i-1}} \mid n_{i,0\to 1})\frac{\Pr(n_{i,s_i})}{\Pr(n_{i-1,s_{i-1}})\Pr(n_{i-2,s_{i-1}})}\Pr(n_{i-2,s_{i-2}}) * \psi \quad 9) \\
&= \Pr(n_{i-1,s_{i-1}}, n_{i-2,s_{i-1}} \mid n_{i,0\to 1})\frac{\Pr(n_{i,s_i})}{\Pr(n_{i-1,s_{i-1}})} * \psi \\
&, \text{such that } \Pr(n_{i-1,s_{i-1}}, \ldots, n_{1,s_1}) = \psi
\end{aligned}$$

Finally, by chain rule, we can change an insertion form of timing path switching probability to a conditional probability based model.

$$\begin{aligned}
&\Pr\left(n_{i-1,s_{i-1}}, n_{i-2,s_{i-1}} \mid n_{i,s_i}\right)\frac{\Pr(n_{i,s_i})}{\Pr(n_{i-1,s_{i-1}})}\Pr\left(n_{i-1,s_{i-1}}, \ldots, n_{1,s_1}\right) = \\
&\Pr\left(n_{i,s_i}\right)\Pr\left(n_{i-1,s_{i-1}}, n_{i-2,s_{i-1}} \mid n_{i,s_i}\right)..\Pr\left(n_{1,s_1} \mid n_{2,s_2}\right)
\end{aligned} \quad 10)$$

As shown in (10), (3) is updated with conditional probabilities of a list of output nets and their input nets. Timing arc is defined as a signal transition from input net to output net. Hence, the switching probability calculation can be expressed with a timing arc based switching probability model. The following describes a generalized form of the timing arc based switching probability.

$$\begin{aligned}
&\Pr(n_{i,s_i}, n_{i-1,s_{i-1}}, \ldots, n_{1,s_1}) \\
&= \Pr(n_{i,s_i})\prod\Pr(\text{input nets of } n_{j,s_j} \mid n_{j,s_j})
\end{aligned} \quad 11)$$

This model can be calculated by a switching activity of endpoint and all conditional probability of each timing arc. We first calculate the probability of each timing arc, and then the switching probability of critical path can be calculated by multiplying timing arc probabilities. To obtain each timing arc probability, in this work, we use samples from Verilog simulation data.

B. Correlation Calculation Method by Timing Arc Based Probability Model

In the previous section, we assume that each critical path for the same endpoint is mutually independent within a narrow monitoring delay target. However, this may not hold true for timing paths between two different endpoints. Depending on circuit characteristics, there could be high dependence among paths. Therefore, it is important to find correlation coefficients for switching probability calculation of entire monitoring endpoints. Assume that there are two different endpoints, *S* and *P*, and timing paths for S and P are denoted as s_i and p_j where $i \in \{1, n\}$ and $j \in \{1, m\}$. The intersection of *S* and *P* can be expressed using a correlation coefficient, $\rho_{S,P}$.

$$\Pr(S,P) = \Pr(S)\Pr(P) + \rho_{S,P}\sqrt{\Pr(S)(1-\Pr(S))\Pr(P)(1-\Pr(P))} \quad 12)$$

The intersection of *S* and *P* can also be denoted by all timing paths in *S* and *P* as

$$\begin{aligned}
&\Pr(S,P) = \Pr(s_1,p_1) + \Pr(s_1,p_2) + \ldots + \Pr(s_n,p_m) = \\
&\sum_{i=1}^{n}\sum_{j=1}^{m}\Pr(s_i)\Pr(p_j) + \sum_{i=1}^{n}\sum_{j=1}^{m}\rho_{s_i,p_j}\zeta \\
&, \text{such that } \zeta = \sqrt{\Pr(s_i)(1-\Pr(s_i))\Pr(p_j)(1-\Pr(p_j))}
\end{aligned} \quad 13)$$

where $\rho_{si,pj}$ is a correlation coefficient of s_i and p_j. By combining (12) and (13), $\rho_{S,P}$ can be found.

$$\begin{aligned}
&\rho_{s,p} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\rho_{s_i,p_j}\sqrt{\Pr(s_i)(1-\Pr(s_i))\Pr(p_j)(1-\Pr(p_j))}}{\sqrt{\Pr(S)(1-\Pr(S))\Pr(P)(1-\Pr(P))}} \\
&, \text{such that } \Pr(S)\Pr(P) = \sum_{i=1}^{n}\sum_{j=1}^{m}\Pr(s_i)\Pr(p_j)
\end{aligned} \quad 14)$$

To calculate $\rho_{S,P}$, it needs the correlation coefficient of two timing paths, $\rho_{si,pj}$. By (12), $\rho_{si,pj}$ can be expressed as (15).

$$\rho_{s_i,p_j} = \frac{\Pr(s_i,p_j) - \Pr(s_i)\Pr(p_j)}{\sqrt{\Pr(s_i)(1-\Pr(s_i))\Pr(p_j)(1-\Pr(p_j))}} \quad 15)$$

To solve (15), an intersection of two different paths needs to be found. *Algorithm 1* describes how the intersection of two different paths can be calculated. It multiplies a switching signal probability of each timing arc in s_i and p_i. In this manner, $\rho_{si,pj}$ can be found. It is used to prune high correlated timing paths.

Algorithm 1 : Intersection(s_i, p_i) //s_i, p_i are timing paths

```
1: N ∈ {timing arc in s_i}; M∈ {timing arc in p_j}
2: O = Pr(s_1)
3:
4: For each c ∈ M do
5:     For each k ∈ N do
6:         if c != k
7:             O = O*Pr(c)
8:     End
9: End
10:
11: return O
```

Fig. 3 shows an example with two monitoring endpoints, *S* and *P*. There are two timings paths s_1, s_2 and p_1, p_2 for *S* and *P* respectively. The following illustrates how the correlation coefficient of two timing paths, $\rho_{s2,p1}$ is found.

$$\begin{aligned}
&\Pr(s_2) = \Pr(n_{7,1\to 0})(n_{5,0\to 1}, n_{4,1\to 1} \mid n_{7,1\to 0})\Pr(n_{2,1\to 0} \mid n_{5,0\to 1}) \\
&\Pr(p_1) = \Pr(n_{8,0\to 1})(n_{6,1\to 0}, n_{3,1\to 1} \mid n_{8,0\to 1}) \\
&\qquad * \Pr(n_{5,0\to 1} \mid n_{6,1\to 0})\Pr(n_{2,1\to 0} \mid n_{5,0\to 1}) \\
&\Pr(s_2,p_1) = \Pr(n_{7,1\to 0})(n_{5,0\to 1}, n_{4,1\to 1} \mid n_{7,1\to 0})\Pr(n_{2,1\to 0} \mid n_{5,0\to 1}) \\
&\qquad * \Pr(n_{8,0\to 1})(n_{6,1\to 0}, n_{3,1\to 1} \mid n_{8,0\to 1})\Pr(n_{5,0\to 1} \mid n_{6,1\to 0}) \\
&\rho_{s_2,p_1} = \frac{\Pr(s_2,p_1) - \Pr(s_2)\Pr(p_1)}{\sqrt{\Pr(s_2)(1-\Pr(s_2))\Pr(p_1)(1-\Pr(p_1))}}
\end{aligned} \quad 16)$$

IV. Selection of Endpoints

A. Problem Formulatin

To develop an efficient and observability timing slack monitoring system, the selection of monitoring endpoints is important. It influences power consumption, area overhead and observability. Therefore, we introduce object functions to select the monitoring endpoints. $x = [x_1, x_2, \ldots]$ is defined as a decision vector where x_i is 1 when a monitor is inserted at N_i and becomes 0 otherwise.

$$N_i \cdot x_i = \begin{cases} \text{if } x_i = 0, null \\ \text{if } x_i = 1, N_i \end{cases} \quad 17)$$

Our object function can be described as follow.

$$\begin{aligned} &\min \sum x_i \\ &\min \sum sw_i \cdot vdd_i^2 \cdot freq_i \cdot x_i \ , sw_i = \text{Switching Activity} \\ &\max \Pr\left(\sum N_i \cdot x_i\right) \end{aligned} \quad 18)$$

The first cost function is to obtain a minimum number of delay monitors to reduce area and routing congestions. Power optimization is found from the second function and the last maximization function finds optimal observability. Since the power consumption optimization indirectly helps to reduce the number of monitors, the objective functions can be updated only by focusing on the power minimization function. In addition, within a power budget, the monitors need to maximize observability. Hence, (18) can be updated as follows.

$$\begin{aligned} &\max \Pr\left(\sum N_i \cdot x_i\right) \\ &\text{subject to} \\ &\quad \sum sw_i \cdot vdd_i^2 \cdot freq_i \cdot x_i \le \text{Power budget} \\ &\quad x_i = \{0,1\} \end{aligned} \quad 19)$$

The observability function can be calculated with a switching probability of each endpoint and intersection. Since higher order intersections are rapidly decreasing, the 1st order intersection is only considered. Therefore, the observability function can be described as follows.

$$\begin{aligned} \Pr\left(\sum N_i \cdot x_i\right) &= \sum \Pr(N_i) \cdot x_i - \sum\sum \Pr(N_i N_j) \cdot x_i x_j \\ &\quad + \sum\sum\sum \Pr(N_i N_j N_k) \cdot x_i x_j x_k + \ldots \\ &\quad + (-1)^{m-1} \Pr\left(\prod N_i\right) \cdot \prod x_i \\ \Leftrightarrow \Pr\left(\sum N_i \cdot x_i\right) &\propto \sum \Pr(N_i) \cdot x_i - \sum\sum \Pr(N_i N_j) \cdot x_i x_j \end{aligned} \quad 20)$$

By combining (19) and (20), (21) can be presented as follows.

$$\begin{aligned} &\max \sum \Pr(N_i) \cdot x_i - \sum\sum \Pr(N_i N_j) \cdot x_i x_j \\ &\text{subject to} \\ &\quad \sum sw_i \cdot vdd_i^2 \cdot freq_i \cdot x_i \le \text{Power budget} \\ &\quad x_i = \{0,1\} \end{aligned} \quad 21)$$

(21) is one of mixed integer quadratic problems with constraints (MIQCP) [10]. It can be solved with mixed integer linear problem (MILP) based solvers by changing variables. Because MIQCP is an *NP-hard* problem, this requires a huge runtime overhead with a large number of variables. To solve the runtime problem, this paper proposes a dynamic programming based solution with fewer constraints. We can claim that, only under some conditions, if $\Pr(N_i,N_j)$ is sufficiently smaller than $\Pr(N_i)$, a solution maximizing $\sum\Pr(N_i)$ maximizes (21). These conditions are that all endpoints are independent and $\sum\Pr(N_i)$ is less *1*. To prove this, we present lemma1. Sec. IV.B. explains how this is used.

Lemma 1: $\Pr(X_i)$ is a probability variable which is all independent and $\sum\Pr(X_i)$ is less than *1*. The solution of maximizing $\sum\Pr(X_i)$ is a solution of maximizing $\sum\Pr(X_i)$ - $\sum\sum\Pr(X_i,X_j)$.

Proof 1: $\Pr(X_i)$ and $\Pr(X_j)$ are non-negative real numbers and less than *1*. $\Pr(X_i, X_j)$ can be expressed as

$$\begin{aligned} &\Pr(X_i)\Pr(X_j) \le \Pr(X_i X_j) \le \Pr(X_i), \text{if } \Pr(X_i) \le \Pr(X_j) \\ &\Pr(X_i X_j) = \alpha_{i,j} \cdot \Pr(X_i)\Pr(X_j) \ , \text{such that } \alpha_{i,j} \ge 1 \end{aligned} \quad 22)$$

By (22), $\sum\Pr(X_i)$ - $\sum\sum\Pr(X_i,X_j)$ can be written as:

$$\begin{aligned} &\sum \Pr(X_i) - \sum\sum \Pr(X_i X_j) = \\ &\Pr(X_1) + \Pr(X_2) + \ldots + \Pr(X_k) + \ldots + \Pr(X_n) - \alpha_{1,2} \cdot \Pr(X_1)\Pr(X_2) \\ &- \alpha_{1,3} \cdot \Pr(X_1)\Pr(X_3) \ldots - \alpha_{n-1,n} \cdot \Pr(X_{n-1})\Pr(X_n) \end{aligned} \quad 23)$$

(24) shows an equation that $\Pr(X_k)$ is replaced by $\Pr'(X_k)$ whose value is smaller than $\Pr(X_k)$ where $k \in [1, N]$.

$$\begin{aligned} &\Pr(X_1) + \Pr(X_2) + \ldots + \Pr'(X_k) + \ldots + \Pr(X_n) - \alpha_{1,2} \cdot \Pr(X_1)\Pr(X_2) \\ &- \alpha_{1,3} \cdot \Pr(X_1)\Pr(X_3) \ldots - \alpha_{n-1,n} \cdot \Pr(X_{n-1})\Pr(X_n) \end{aligned} \quad 24)$$

If we assume $\alpha_{1,2} = \alpha_{1,3} = \ldots = \alpha_{n-1,n} = \alpha$ to simplify this problem, the difference between (23) and (24) can be expressed as in (25). If this becomes negative, the solutions of $\max(\sum\Pr(X_i))$ and $\max(\sum\Pr(X_i)$ - $\sum\sum\Pr(X_i,X_j))$ must be different since $\sum\Pr(X_i)$ - $\sum\sum\Pr(X_i,X_j)$ is maximized by $\Pr'(X_k)$ while $\sum\Pr(X_i)$ is not.

$$\begin{aligned} &\Pr(X_k) - \Pr'(X_k) \\ &- \alpha \cdot \Pr(X_k)\Pr(X_1) - \ldots - \alpha \cdot \Pr(X_k)\Pr(X_n) \\ &+ \alpha \cdot \Pr'(X_k)\Pr(X_1) + \ldots + \alpha \cdot \Pr'(X_k)\Pr(X_n) \\ &= (\Pr(X_k) - \Pr'(X_k))(1 - \alpha \cdot (\Pr(X_1) + \ldots \\ &\quad + \Pr(X_{k-1}) + \Pr(X_{k+1}) + \ldots + \Pr(X_n)) \end{aligned} \quad 25)$$

However, to make (25) positive, $\sum\Pr(X_i)$ needs to be smaller than *1* and α has to be *1* which is the smallest value.

$$\begin{aligned} &\alpha \cdot (\Pr(X_1) + \ldots + \Pr(X_{k-1}) + \Pr(X_{k+1}) + \ldots + \Pr(X_n) < 1 \\ &\Leftrightarrow \alpha = 1, \sum \Pr(X_i) < 1 \end{aligned} \quad 26)$$

Under this condition in (26), (25) is always positive and the solution of $\max(\sum\Pr(X_i))$ is a solution of $\max(\sum\Pr(X_i)$ - $\sum\sum\Pr(X_i,X_j))$ since higher $\Pr(X_i)$ is always selected than smaller $\Pr'(X_i)$.

In (21), the number of x_i is determined by power constraints. x_i would try different $\Pr(N_i)$ to find an optimal solution. This is the same as a replacement of $\Pr(X_k)$ by $\Pr'(X_k)$ in *lemma 1*. Thereby, if all endpoints are independent and $\sum\Pr(N_i)$ is less *1*, we can only focus on the maximization of $\sum\Pr(N_i)x_i$.

Satisfying the first condition that all endpoints are independent is difficult since all independent endpoints need to be found which is an *NP-hard* problem. This will be discussed in the following section. However, the second condition, $\sum\Pr(N_i) < 1$ is not a difficult condition to meet. We can satisfy it by normalization that adds more idle states to Verilog functional simulation vectors. In another way, we can add a new constraint function, $\sum\Pr(N_i) < 1$. Practically, if a chip designer can measure on-chip worst speed for few clock cycles, this condition can be simply satisfied. Therefore, (21) can be updated as an integer linear problem as follows.

$$\max \sum \Pr(N_i) \cdot x_i$$
$$\text{subject to} \quad \sum sw_i \cdot vdd_i^2 \cdot freq_i \cdot x_i \le \text{Power budget} \quad 27)$$
$$\sum \Pr(N_i) \cdot x_i < 1 \ , \ x_i = \{0,1\}$$

This problem can be dealt with 0/1 knapsack problem [11] by changing type of parameters that is well known in combinatorial optimization. The 0-1 Knapsack problem is proven to be *NP-hard* in the ordinary sense. It can be solved by a dynamic program in pseudo-polynomial time. Therefore, we can use a dynamic program approach to find an optimal solution with constraints.

B. Finding Independent Endpoints by Markov

As addressed in Sec. IV.A, *lemma 1* is satisfied under two conditions. To use the first condition, the proposed method requires finding all independent endpoints. For this, the correlation efficient can be expressed as a graph manner. The correlation coefficient between two different endpoints can be modeled as an indirect graph where its vertex, edge and weight are defined as follows.

V_i = vertex is each endpoint, N_i
$E_{i,j}$ =edge is connection between $V_i(N_i)$ and $V_j(N_j)$
ρ = weight is correlation coefficients of $V_i(N_i)$ and $V_j(N_j)$

This graph $G(V, E, \rho)$ is represent by NxN adjacency matrix A as in (27).

$$A = \begin{vmatrix} \rho_{1,1} & \rho_{1,2} & \cdots & \rho_{1,N} \\ \rho_{2,1} & \rho_{2,1} & \cdots & \rho_{2,1} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{N,1} & \rho_{N,2} & \cdots & \rho_{N,N} \end{vmatrix} \quad 28)$$

In the graph model, finding a set of independent endpoints is considered as a maximum clique problem in graph theory, however, this is a well-known *NP-complete* problem. Therefore, the proposed method uses an indirect approach such as a graph clustering algorithm to find independent endpoint sets. Markov Clustering (MCL), a graph clustering algorithm based on stochastic flow simulation, has proved to be highly effective for several networks [9]. The key intuition behind MCL is that a random walk that visits a dense cluster will likely not leave the cluster until many of its vertices have been visited. A column stochastic matrix M is NxN matrix which can be interpreted as the transition probabilities of a random walk defined on the network. Given an adjacency matrix A, a corresponding canonical flow matrix M can be denoted as,

$$M_{i,j} = \begin{cases} \dfrac{A_{i,j}}{\sum_{x=1}^{n} A_{x,j}} & \text{, if } \sum_{x=1}^{n} A_{x,j} > 0 \\ 0 & \text{, otherwise} \end{cases} \quad 29)$$

MCL involves iterative two operations, expansion and inflation.

- Expansion is a simple matrix multiplication, in which M is taken to the power $e>1$ thus simulating e steps of a random walk.
- Inflation is an operation, in which M is re-normalized after taking every entry to its r^{th} power to maximize large weights and to minimize small weights.

The iteration is halted upon reaching a recurrent state or a fixpoint [9]. The clustering is induced by connected components of the graph underlying the final matrix. This algorithm has a complexity of $O(NK^2)$ where N is the number of nodes, and K is the average of neighbors of nodes in the graph. To reduce runtime, if some correlation coefficients are less than a certain threshold, this edge can be removed. Finally, we select an independent endpoint from each cluster by MCL, which has high switching probability in the cluster.

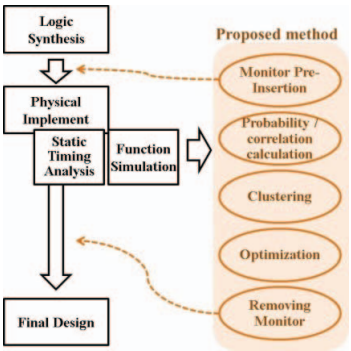

Figure 4. Overall flow of the proposed on-chip timing slack monitoring methodology.

C. Overall Flow of The Proposed On-Chip Timing Slack Monitoring Methodology

Fig.4 shows an overall flow of the proposed method. For on-chip timing slack monitor, we use a special flip-flop combining a normal flip-flop and a timing slack monitor [1]. Before P&R (place and route), this flip-flop can marginally be inserted considering timing analysis in the synthesis stage. After P&R, unnecessary timing slack monitoring flip-flops are discarded. Since the proposed method inserts the monitor before P&R, it would be much convenient and practical than the conventional methods which add the monitor after P&R.

After P&R, critical path reports are extracted by static timing analysis. For a switching probability and correlation coefficients calculation of critical paths, we use Verilog functional simulation data from value change dump (VCD) files. After a switching probability and correlation coefficients calculation, the independent endpoint candidates can be found by MCL. Moreover, the final endpoints are found among candidates considering observability probability and power constraints. It needs to be highlighted that the proposed method can easily integrated with conventional on-chip monitoring methods [1,2].

V. Experimental Results

For experiments, *ISCAS'89* benchmark circuits are synthesized by *FreePDK 45nm* library [12] and *Synopsys Design Compiler*. *Cadence* verilog simulator and *Synopsys PrimeTime* are used for functional simulation and static timing analysis. Since *ISCAS* circuits are benchmark logic whose function is not generally considered, random functional input vectors are used for simulations. The proposed method is implemented with *C/C++*.

- Observability and efficiency of the proposed method

Table 1 presents the experimental results with various *ISCAS'89* benchmark circuits. The proposed method is compared with a worst critical path based method which inserts timing slack monitor to endpoints with the worst timing slack. Design information is provided from the first to fourth columns.

TABLE 1. COMPARISON WITH WORST CRITICAL PATH BASED APPROACH AND THE PROPOSED METHOD

Design information				Worst Critical Path based method			Proposed method			Difference (ratio)		
Name	# of Cell	# of FF	Critical paths	S.P.	Power (S.A.)	# of Monitors	S.P.	Power (S.A.)	# of Monitors	S.P.	Power (S.A.)	# of Monitors
s953	388	29	416	5.54E-02	3.72E-01	18	4.98E-02	1.86E-01	9	10.05%	50.08%	50.00%
s1238	578	18	74	1.91E-02	2.88E-01	6	1.69E-02	1.94E-01	5	11.62%	32.56%	16.67%
s5378	1470	179	1942	1.71E-01	4.05E+00	51	1.53E-01	6.22E-01	14	10.88%	84.64%	72.55%
s9234	1134	211	5336	6.18E-04	2.12E-01	29	5.45E-04	4.91E-02	5	11.89%	76.78%	82.76%
s35932	11719	1728	10418	1.00E+00	2.08E+01	288	9.05E-01	2.85E+00	32	9.49%	86.28%	88.89%
s38584	12378	1426	18276	1.00E+00	3.40E+01	858	9.08E-01	1.06E+00	24	9.22%	96.88%	97.20%

**S.P. is sum of all switching probability; S.A. is sum of all switching activity for analysis a dynamic power consumption.*

The number of critical paths in the fourth column includes not only worst critical paths but also sub critical paths within monitoring delay target in Fig. 1. In the fifth column, the sum of switching probability for all monitors is given for the worst critical path based method. The sixth column provides the sum of switching activity, however, this can be considered as power assuming *vdd* and *freq* are the same in the experiments. The number of monitors by the worst critical path based method is given in the seventh column. The sum of signal probability, the sum of switching activity and the number of monitors by the proposed method are respectively given in the eighth, ninth and tenth column. As can be seen, the proposed method achieves a close switching probability to the worst critical path based method with a smaller number of monitors. This tells that the proposed method provides high observability with a small number of monitors. The monitor reduction significantly reduces a dynamic power consumption of monitors by 32.56 ~ 96.88%. The proposed method degrades the sum of switching probability by 10%, however, it should be noted that the number of monitors for the proposed method is significantly small. In addition, processor implementation results of Slcakprobe [1] show that it introduces an additional power overhead of 0.12% per monitor. Therefore, we can expect that reduction of the number of monitor is very important.

- Accuracy of timing critical path switching probability

To compare the critical path switching probability calculation with Verilog simulation results, Verilog functional simulations are run with 30M input patterns. For each simulation, we calculate the switching probability of 200 endpoints in *s38584*. As in Fig. 1, a monitor delay target is set and we count the switching probability for each monitor when its arrival time exceeds the monitor delay target. Fig. 5 shows the switching probability for each endpoint and the reference data captured from the Verilog simulation as described above. It shows some difference between the reference data from Verilog simulations and the calculated signal probability by the proposed method. However, considering the difference of the delay calculation methods between STA and simulation, it should be noted that the results from the proposed method shows a good correlation with reference data.

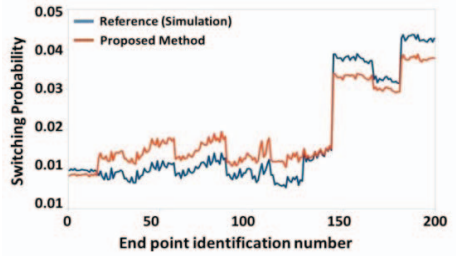

Figure 5. Switching probability comparison of proposed method vs. simulation.

VI. CONCLUSIONS

This paper proposes a novel on-chip timing slack monitoring methodology. The proposed method includes 1) switching probability of critical path and endpoint calculation by timing arc based probability model, 2) a new correlation coefficient calculation method, and 3) endpoints selection method for locating on-chip timing slack monitor. Experimental results show that the proposed method finds a set of monitor to minimize the power and area. With a small number of monitors, they still have a high switching probability. This means that they would be activated more often than larger number of monitors with a low switching probability. Therefore, the proposed on-chip timing slack monitoring methodology is highly efficient and observable. We will expand the work for improving accuracy of our model and adopting vectoreless approach.

ACKNOWLEDGEMENT

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea by the Ministry of Education under Grant NRF-2015R1D1A1A01058856, and in part by the MOTIE(Ministry of Trade, Industry & Energy (10080594) and KSRC(Korea Semiconductor Research Consortium) support program for the development of the future semiconductor device.

REFERENCES

[1] L. Lai, V. Chandra, R. Aitken, and P. Gupta, "SlackProbe: A low overhead in situ on-line timing slack monitoring methodology," in Proc. DATE, Mar. 2013, pp. 282–287.
[2] L. Xie and A. Davoodi, "Representative path selection for post-silicon timing prediction under variability," in Proc. ACM/IEEE DAC, Jun. 2010.
[3] Michael S Floyd et al., "Adaptive Clocking in the POWER9TM Processor for Voltage Droop Protection," IEEE ISSCC 2017
[4] T.-B. Chan et al., "DDRO: A novel performance monitoring methodology based on design-dependent ring oscillators", in Int'l Symp. on Quality Electronic Design, 2012, pp. 633–640
[5] A. Drake et al., "A distributed critical-path timing monitor for a 65 nm high-performance microprocessor," in 2007 IEEE ISSCC Dig., Feb. 2007, pp. 398–399.
[6] K.K. Kim, W. Wang, and K. Choi, "On-chip aging sensor circuits for reliable nanometer MOSFET digital circuits," T-CAS-II, vol.57, no. 10, pp.798–802, 2010.
[7] X. Wang et al., "Silicon odometers: Compact in situ aging sensors for robust system design", IEEE Micro, vol. 34, 2014.
[8] S. Das et al., "Razor II: In situ error detection and correction for PVT and SER tolerance," IEEE J. Solid-State Circuits, vol. 44, no. 1, pp. 32–48, Jan. 2009.
[9] Enright A.J., Van Dongen S., Ouzounis C.A., "An efficient algorithm for large-scale detection of protein families," Nucleic Acids Research 30(7):1575-1584 (2002).
[10] R. Lazimy, "Mixed–integer quadratic programming", Mathematical Programming 22 (1982)
[11] Oscar H. Ibarra and Chul E. Kim, "Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems," Journal of the ACM (JACM), vol. 22, no. 4, 1975
[12] Oklahoma State University System on Chip (SoC) Design Flows,http://vlsiarch.ecen.okstate.edu/flows
[13] Tezaswi Raja et al., "Variable Input Delay CMOS Logic for Low Power Design", IEEE Transactions on Very Large Scale Integration (VLSI) System, Vol. 17, Issue: 10, pp. 1534- 1545, 2009