Probabilistic Retiming: A Circuit Optimization Technique*

Sissades Tongsima Chantana Chantrapornchai
Nelson Luiz Passos and Edwin H.-M. Sha
Department of Computer Science and Engineering
University of Notre Dame, Notre Dame, IN 46556

Abstract

VLSI circuit manufacturing may result in devices with different propagation delays. Hence, the estimation of such delays during the design procedure may not prove totally accurate due to the fabrication process. This paper presents a new optimization methodology, called probabilistic retiming, which transforms a circuit based on statistical data gathered from the production history. Such circuits are modeled as graphs where each vertex represents a combinational element that has a probabilistic timing characteristic. A polynomial-time algorithm, applicable to such a graph, is developed which retimes a circuit in order to produce a design operating on a specified cycle time c within a given confidence level θ. In other words, the clock cycle of the retimed circuit is guaranteed to be less than or equal to c with at least probability θ. Experiments show the effectiveness of the algorithm, subject to the designer requirements and to the manufacturing information, which is able to significantly improve the confidence in the fabrication yielding factors.

1 Introduction

In VLSI design, engineers are normally facing the problem of designing a circuit able to achieve a given time constraint. The circuit transformation technique called *retiming* is an optimization tool commonly used in achieving such a goal [8]. Nevertheless, the information about the execution time[1] of each circuit component is uncertain since, in general, two equal elements may have different running time due to the fabrication process. The use of estimated values of the execution time in the synthesis process, to guide the design, may not always be correct and, therefore, the system may need to be redesigned. This paper presents a polynomial-time circuit transformation algorithm, called *probabilistic retiming*, which considers the probabilistic nature of the timing characteristic of the basic elements of the circuitry, producing a circuit able to satisfy the time constraint specification within a given confidence level. As traditional retiming has great impact to the optimization problem of various applications, this new technique may give similar effectiveness to various applications with uncertain characteristics.

*This work was partially supported by the Royal Thai Government Scholarship, Mensch Fellowship, and the NFS grant MIP 95-01006.

[1] Note that we will use the term execution time and propagation delay interchangeably.

In the traditional retiming, Leiserson and Saxe [8] presented a polynomial-time algorithm that transforms a circuit into an equivalent one within a given and feasible clock period, preserving its functional behavior. Malik et al. [12] generalized the idea of retiming and re-synthesis by allowing negative registers during the optimization phase so that a larger portion of a circuit could be viewed as a single block. Such an operation is legal as long as those negative registers were returned to the environment after re-synthesizing the circuit. In [3, 10], retiming algorithms were also developed to address the problem of multi-phase, level-clocked circuitry. However, all of these contributions considered the propagation delay of the elements in the circuit as a fixed value, ignoring the uncertainty of the manufacturing process.

Many other researchers implicitly adopted the worst case timing information as one of their design assumptions. In particular, Ishii modified the retiming technique in such a way that it can also handle precharged structures and gated clock signal circuitry [2]. Dey, Potkonjak and Rothweiler proposed a method to enhance retiming by transforming sequential circuits [1]. A large number of applications of retiming were explored, always making the use of the same assumption, such as the work of Shenoy and Brayton [16], Lockyear and Ebeling [11], Liu et al. [9]. Lower bounds were computed, such as the work by Papaefthymiou, based on the minimum clock period presented in terms of the delay-to-register ratio, which can be yielded by retiming a circuit without considering the probabilistic behavior [14].

More comprehensive delay models were proposed by Soyata, Friedman and Mulligan [17,18], while Lalgudi and Papaefthymiou [7] presented an efficient retiming algorithm based on monotonic programming. In their studies, a more realistic timing behavior of reading from and writing to registers was presented. It does not, however, demonstrate the variance of circuit delays due to the manufacturing process. Recently, Karkowski and Otten introduced a model to handle the imprecise propagation delay of events [4]. In their approach, the fuzzy set theory [19] was employed to model the imprecise delays with only three possible values. Multiple objective linear programming [6] was chosen to reduce the uncertainty of such delays. Nonetheless, this model is restrict to a simple triangular fuzzy distribution and does not consider probability values, usually easy to obtain from the quality control of the fabrication process.

In order to generalize the idea of having imprecise delays, the probability theory must be applied to model the variance of such delays. In this paper, propagation delays are assumed to be random variables that are associated with their probability distributions. The retiming technique is, then, extended to optimize circuits under probabilistic environments. In order to establish this methodology, a circuit is represented as a graph where a vertex set V is a collection of combinational logic components, associated with random variables representing the propagation delays. Edges in the graph represent the interconnections between components which may route through zero or more registers. The probability distribution of the execution time is denoted by $\mathcal{P}(\mathrm{X} = \mathrm{x})$, where X represents the propagation delay for a component and x is one of the possible values [5, 13].

For example, Figure 1(a) illustrates a graph G in which the set of vertices has the nodes $\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}$, and E. Assume such nodes have the following timing characteristics:

$$\text{Node A}: \mathcal{P}(X = x) = \begin{cases} .1 & \text{if } x = 1, \\ .9 & \text{if } x = 3. \end{cases} \qquad \text{Node B}: \mathcal{P}(X = x) = \begin{cases} .3 & \text{if } x = 1, \\ .7 & \text{if } x = 2. \end{cases}$$

$$\text{Node C}: \mathcal{P}(X = x) = \begin{cases} .4 & \text{if } x = 1, \\ .6 & \text{if } x = 2. \end{cases} \qquad \text{Node D}: \mathcal{P}(X = x) = \begin{cases} .3 & \text{if } x = 1, \\ .7 & \text{if } x = 2. \end{cases}$$

and

$$\text{Node E}: \mathcal{P}(X = x) = \begin{cases} .9 & \text{if } x = 1, \\ .1 & \text{if } x = 4. \end{cases}$$

Figure 1(b) shows a graph which considers the execution time of each vertex based on the worst case analysis of the propagation delays. The maximum propagation delay of the paths that contains no registers, termed clock or cycle period of the graph G, $\Phi(G)$, is *8*. Based on the worst case scenario, the traditional retiming gives the "best" circuit shown in figure Figure 1(c). But due to the variance of manufacturing, the best retiming obtained by the worst-case scenario may produce less optimal circuits. One might wish to retime G in order to obtain $\Phi(G) = 3$ with the consideration of the manufacturing uncertainty. It is obvious that the desired cycle time cannot be achieved from the worst case analysis. However, since the propagation delay associated with each component is a random variable, a designer might wish to produce a circuit in such a way that, with at least 70% confidence, the final produced systems could operate at a cycle time less than or equal to *3*.

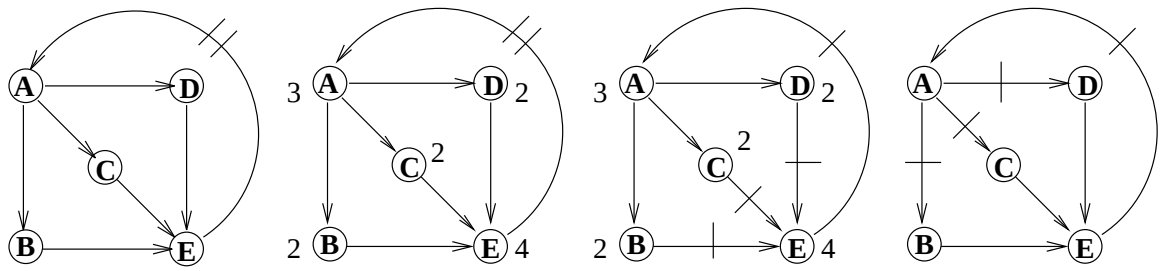


Figure 1: An example of 5-node graph

Let us investigate the retimed graph presented in Figure 1(d). By using the probability theory, we can calculate the probability distribution of the total execution time of the paths that go through zero registers and compute the possible maximum values from them. If Y is the random variable representing the *maximum* of the cumulative propagation delays between any path that has no registers on it, then $\mathcal{P}(Y = 2) = 0.026, \mathcal{P}(Y = 3) = 0.703$, and $\mathcal{P}(Y > 4) = 0.271$. Now after manufacturing, it is most likely (approximately 80%) that the propagation delay of node A will be 3 and about 90% that node E will take one time unit. Consider that the execution time of nodes B, C, and D is 2. Therefore, the configuration in Figure 1(c) will have the clock period $\Phi(G) = 5$ while the clock period of the graph in Figure 1(d) will be 3.

Notice that the probability of the random variable to Y assume a value $y > 3$ is less than *0.3*. Consequently, the retimed graph in Figure 1(d) has satisfied the designer requirements that the probability of

$\Phi(G) \leq 3$ is greater than or equal to 70%, i.e., the probability of $\Phi(G) > 3$ is less than *0.3*. The algorithm able to achieve such results is described in the remainder of this paper which is organized as follows: Section 2 presents some basic fundamentals of the model. The probabilistic retiming is discussed in Section 3. Some experimental results are presented in Section 4. Finally, Section 5 concludes the contribution of this research.

2 Preliminaries

In this section we introduce the model used in the probabilistic retiming problem. The terminology and some notations relevant to this work are also discussed. We begin by presenting the graph model which is the representation of the circuit.

2.1 Graph model

A circuit that contains functional elements associated with probability distribution of propagation delays can be modeled as a *probabilistic graph* (PG). The following gives the formal definition for a PG.

Definition 2.1 *A probabilistic graph (PG)* $G = \langle V, E, d, t \rangle$ *is a vertex-weighted, edge-weighted, directed graph where* V *is a set of nodes and each node represents one of the circuit functional elements,* E *is a set of edges representing the circuit interconnections,* d *is a function* $d : E \mapsto \mathbf{Z}$ *representing the number of register counts on an edge, and* t *is a function* $t : V \mapsto \mathcal{R}$ *where* $\mathcal{R}$ *is a set of discrete random variables of propagation delay.*

The notation $\mathcal{P}(T = x)$ is read "the probability that the random variable T assumes the value x". Each vertex $v \in V$ is weighted with a *probability distribution function* (pdf) of execution time, given by $t(v) = \mathcal{P}(T_v = x)$, where T_v is a discrete random variable associated with the set of possible propagation delays of the vertex v such that $\sum_{\forall x} \mathcal{P}(T_v = x) = 1$. As an example, the set of vertices $V = \{A, B, C, D, E\}$ in Figure 1(a) have their pdfs presented in Table 1.

$v \in V$	$T_v = x$		$\mathcal{P}(T_v = x)$	
	x_1	x_2	$\mathcal{P}(T_v = x_1)$	$\mathcal{P}(T_v = x_2)$
A	2	3	0.8	0.2
B	1	2	0.5	0.5
C	1	2	0.7	0.3
D	1	2	0.9	0.1
E	1	3	0.9	0.1

Table 1: pdf of the vertices in Figure 1(a)

An edge $e \in E$ from vertices u to v is denoted by $u \xrightarrow{e} v$ and a path p starting from a node u and ending at a node v is indicated by the notation $u \overset{p}{\rightsquigarrow} v$. The register count of a path $p = v_0 @ > e_0 >> v_1 @ > e_1 >> \cdots @ > e_{k-1} >> v_k$ is $d(p) = \sum_{i=0}^{k-1} d(e_i)$. As an example, Figure 1(a) has the set of edges

$E = A \xrightarrow{e_1} B, A \xrightarrow{e_2} C, A \xrightarrow{e_3} D, B \xrightarrow{e_4} E, C \xrightarrow{e_5} E, D \xrightarrow{e_6} E,$ and $E \xrightarrow{e_7} A$. The register counts of each edge $e \in E$ is $d(e_7) = 2$, and $d(e_i) = 0$, for $i = 1 \ldots 6$.

2.2 Two-dimensional random variables

Since the propagation delay of a vertex is a random variable, an operation between two random variables of two dependent vertices involves a function of two-dimensional random variables, i.e., two or more numerical characteristics must be observed simultaneously. In this section, some notions about two-dimensional random variables [5], which will be used extensively in probabilistic retiming, are discussed.

For a sample space S associated with an experiment $\mathcal{E}$, and $X = X(s)$ and $Y = Y(s)$ being two one-dimensional random variables that assign a real number to each outcome $s \in S$, a *two-dimensional discrete random variable* is denoted by (X, Y) if the possible values of (X, Y) are finite or countably infinite. Notice that for an *n-dimensional* random variable, the n-tuple $(X_1, \ldots, X_n)$ is applied where $X_i = X_i(s), i = 1 \ldots n$, is a function mapping a real number to every outcome $s \in S$.

If (x_i, y_j) is a possible outcome of a two-dimensional discrete random variable (X, Y), then $p(x_i, y_j) = \mathcal{P}(X = x_i, Y = y_j)$ satisfies the following conditions:

$$(1) \quad p(x_i, y_j) \geq 0 \qquad \forall (x, y),$$

$$(2) \quad \sum_{j=1}^{\infty} \sum_{i=1}^{\infty} p(x_i, y_j) = 1.$$

We assume that the execution time T_v associated with any vertex $v \in V$ is independent of the other. Therefore, if X and Y are independent random variables, any possible outcome of X does not influence the outcome of Y, i.e., the random variables X and Y are independent if and only if, for all i, j,

$$\mathcal{P}(X = x_i, Y = y_j) = \mathcal{P}(X = x_i) \times \mathcal{P}(Y = y_j).$$

For instance, consider two independent combinational elements α and β. Let T_α and T_β be two random variables describing the set of possible propagation delays for the elements α and β respectively. For the circuit α, let $\mathcal{P}(T_\alpha = 1) = 0.3, \mathcal{P}(T_\alpha = 4) = 0.7$ while $\mathcal{P}(T_\beta = 2) = 0.9, \mathcal{P}(T_\beta = 5) = 0.1$ are the pdf of circuit β. Thus $\mathcal{P}(T_\alpha = 1, T_\beta = 2) = \mathcal{P}(T_\alpha = 1) \times \mathcal{P}(T_\beta = 2) = 0.27$.

Consider $\Theta = H(X, Y)$, a function of two random variables X and Y. Hence, $\Theta = \Theta(s)$ is also a random variable. In order to compute a function of two random variables, $H(X, Y)$, the following sequence of steps must be considered: first, evaluate the possible outcome s of the experiment $\mathcal{E}$. Then compute the values for $X(s)$ and $Y(s)$. Finally, we can compute the number $\Theta = \Theta(s) = H(X, Y)$.

Note that Θ is now a one-dimensional random variable. If X and Y are both independent of each other, we can simply calculate the pdf of Θ. As a demonstration, consider $A = X+Y$, a one-dimensional discrete random variable that is computed by the addition of the discrete random variables X and Y, and $M = \max(X, Y)$, obtained from finding the greatest value among all pairs of items of X and Y. As an example, consider the

		X			
	$\sigma(\pi)$	1	2	7	$p(y)$
Y	2	3(0.16)	4(0.40)	9(0.24)	0.8
	3	4(0.02)	5(0.05)	10(0.03)	0.1
	4	5(0.02)	6(0.05)	11(0.03)	0.1
	$p(x)$	0.2	0.5	0.3	1.0

(a) Example of $\mathcal{P}(A = X + Y)$

		X			
	$\mu(\pi)$	1	2	7	$p(y)$
Y	2	2(0.16)	2(0.40)	7(0.24)	0.8
	3	3(0.02)	3(0.05)	7(0.03)	0.1
	4	4(0.02)	4(0.05)	7(0.03)	0.1
	$p(x)$	0.2	0.5	0.3	1.0

(b) Example of $\mathcal{P}(M = \max(X, Y))$

Table 2: Results from computing A and M

assignments for X and Y as following:

$$p(x) = \mathcal{P}(X = x) = \begin{cases} 0.2 & \text{if } x = 1, \\ 0.5 & \text{if } x = 2, \\ 0.3 & \text{if } x = 7. \end{cases} \qquad p(y) = \mathcal{P}(Y = y) = \begin{cases} 0.8 & \text{if } y = 2, \\ 0.1 & \text{if } y = 3, \\ 0.1 & \text{if } y = 4. \end{cases}$$

We demonstrate the calculation of the discrete random variable A by using Table 2(a). Note that $\sigma(\pi)$, in each entry represents a possible addition output σ and its probability π. Since both X and Y are independent, the probability π, then, is calculated from $\mathcal{P}(X = x) \times \mathcal{P}(Y = y)$. Table 2(b) presents the results of the maximum of the two random variables X and Y. The pair of numbers $\mu(\pi)$ in each entry represents the possible maximum result μ, and π is the probability associated with μ. The pdf associated with A and M are presented in Table 3.

	Possible values (σ, μ)									
	2	3	4	5	6	7	8	9	10	11
$\mathcal{P}(A = \sigma)$	0.00	0.16	0.42	0.07	0.05	0.00	0.00	0.24	0.03	0.03
$\mathcal{P}(M = \mu)$	0.46	0.07	0.07	0.00	0.00	0.30	0.00	0.00	0.00	0.00

Table 3: The pdf of A and M

3 Probabilistic Retiming

Since the propagation delay of each vertex is now a random variable, the traditional notion of a global clock period, $\Phi(G)$, for a graph G is no longer valid. Therefore, we define a new concept, maximum reaching time (mrt) which represents the maximum possible clock period[2] for a node in the probabilistic graph (PG). This notion is essential in deciding the retiming function r. The algorithm for computing mrt is presented as a basic tool for the probabilistic retiming algorithm.

3.1 Retiming with maximum reaching time

In the traditional retiming, the clock period is given by $\Phi(G) = \max\{t(p) : d(p) = 0\}$, where $t(p)$ is the total execution time of the path p, and $d(p)$ is the number of registers along that path. In this paper, since the

[2] We can call mrt "clock period" if all random variables are replaced by some exact values.

execution time of each vertex along the path is a discrete random variable, the sum of propagation delays of all dependent vertices that are connected by a path $u \overset{p}{\rightsquigarrow} v$ which contains no registers establishes a *varying* clock period due to the several possible propagation delays.

First of all, let us introduce two basic functions of random variables, summation and maximum, which are needed in computing the maximum reaching time (mrt).

Definition 3.1 *Let T_1 and T_2 be two random variables of propagation delay associated with two vertices v_1 and v_2 where $v_1 \overset{e}{\rightarrow} v_2$. The summation function is defined as $A_e = T_1 + T_2$.*

Note that the summation function can be extended to handle n random variables of vertices along a path p, A_p, by applying the function successively to each pair of the random variables, i.e., the summation is closed under associativity property. The probability associated with the summation function can be computed by the following lemma:

Lemma 3.1 *Let T_u and T_v be two independent discrete random variables representing the propagation delays of nodes u and v, and $A = T_u + T_v$. Then the pdf of A is computed by $\mathcal{P}(A = x) = \sum_{\forall\, x_m, x_n} \mathcal{P}(T_u = x_m) \times \mathcal{P}(T_v = x_n)$, where $x_m + x_n = x$.*

Proof: Immediate from the definition of function of two independent random variables. □

As an example, let $x_1 = 1, x_2 = 2$ be the possible values of T_u and $y_1 = 2, y_2 = 3$ be the possible numbers of T_v with the pdfs, $\mathcal{P}(T_u = 1) = 0.5$, $\mathcal{P}(T_u = 2) = 0.5$, $\mathcal{P}(T_v = 2) = 0.3$, and $\mathcal{P}(T_v = 3) = 0.7$. If $A = T_u + T_v$, then $\mathcal{P}(A = 3) = 0.15, \mathcal{P}(A = 4) = 0.5$, and $\mathcal{P}(A = 5) = 0.35$. The following function determines the maximum possible values between two random variables computed from the summation function.

Definition 3.2 *Let A_1 and A_2 be two random variables. The maximum function is defined as $M = \max(A_1, A_2)$, where M is the set of possible greatest values produced from A_1 and A_2.*

The definition for the maximum function can be easily generalized for computing the maximum among n random variables, $A_1, \ldots, A_n$ by computing the function in pairs, repeatedly. Consider the case where edges $a \overset{e_1}{\rightarrow} d$, $b \overset{e_2}{\rightarrow} d$, and $c \overset{e_3}{\rightarrow} d$ are associated with the random variables A_1, A_2, and A_3, computed according to Definition 3.1. Let $h_i(x) = \mathcal{P}(A_i = x)$ such that

$$h_1(x) = \begin{cases} 0.5 & \text{if } x = 1, \\ 0.5 & \text{if } x = 2 \end{cases} \qquad h_2(x) = \begin{cases} 0.4 & \text{if } x = 2, \\ 0.6 & \text{if } x = 5 \end{cases} \qquad h_3(x) = \begin{cases} 0.2 & \text{if } x = 3, \\ 0.8 & \text{if } x = 4. \end{cases}$$

The possible greatest values $M = \max(A_1, A_2, A_3)$ are $3, 4$, or 5. The probability associated to the maximum function can be computed by the lemma below:

Lemma 3.2 *Let A_1 and A_2 be two indepedent discrete random variables and $M = \max(A_1, A_2)$. Then the pdf of M is given by $\mathcal{P}(M = x) = \sum_{\forall\, x_m, x_n} \mathcal{P}(A_1 = x_m) \times \mathcal{P}(A_2 = x_n)$, where $\max(x_m, x_n) = x$.*

Proof: Immediate from the definition of function of two independent random variables. □

To illustrate this lemma, consider the previous example, if $p(x) = \mathcal{P}(M = x)$, then, $p(3) = 1 \times 0.4 \times 0.2 = 0.08$, $p(4) = 1 \times 0.4 \times 0.8 = 0.32$, and $p(5) = 1 \times 0.6 \times 1 = 0.6$. Having defined the basic operations above, we can now state the definition of mrt as following:

Definition 3.3 *Let* $u \overset{p_i}{\rightsquigarrow} v$, $i = 1 \ldots n$, *be the possible paths between two nodes* u *and* v *containing zero register count, and* A_{p_i} *be the summation of propagation delays of the vertices along the path* p_i. *The maximum reaching time (mrt) among the paths* p_i *from* u *to* v *is* $\Psi(v) = \mathsf{mrt}(u,v) = \max_{\forall i}(A_{p_i})$.

As in the original retiming, we want to optimize the clock period of PG. Since the propagation delay of each vertex is uncertain, the goal of retiming the graph must be extended in such a way that the probability of $\Psi(v) < c$, where c is the desired clock period, is larger than the confidence level θ, i.e., the probability of $\Psi(v) < c$ is strictly less than some acceptable probability value $\delta = 1 - \theta$ for all nodes v in the graph. In other words, the probability of the greatest possible value of mrt with respect to each path will be minimized. The following theorem expresses such a condition.

Theorem 3.3 *Given a PG* $G = \langle V, E, d, t\rangle$, $u \overset{p}{\rightsquigarrow} v$, *and a desired clock period* c *within a confidence probability* $\theta = 1 - \delta$, *i.e.,* $\mathcal{P}(\Psi(v) > c) \leq \delta$, *is equivalent to having at least one register in the path* $u \overset{p}{\rightsquigarrow} v$ *if* $\mathcal{P}(\Psi(v) > c) > \delta$.

Proof: Consider the case of a path $u \overset{p}{\rightsquigarrow} v$ such that $\mathcal{P}(\Psi(v) > c) \leq \delta$. Therefore, there is no need to insert any register into the path p. If $\mathcal{P}(\Psi(v) > c) > \delta$, then at least one register must be moved into the path p in order to relax the timing constraint such that $\mathcal{P}(\Psi(v) > c) \leq \delta$. □

3.2 Implementation of the algorithm

In this section, we present an algorithm implementing the concept described in Theorem 3.3. We begin by presenting the algorithm that computes the mrt. Since the mrt is defined as the maximum of the total execution time of any path routing through zero registers to the same destination node, one can calculate the mrt by operating on a graph that has only no-register edges, i.e., a directed acyclic graph (DAG). Assume that a dummy node v_0 which has zero execution time is connected by no-register edges to every other nodes in the graph. The following algorithm computes the $\mathsf{mrt}(v_0, v)$, or $\Psi(v)$ for short, with respect to the dummy node v_0, and any node $v \in V$.

Algorithm 3.1 (Maximum reaching time)
Input : PG $G = \langle V, E, d, t\rangle$
Output: $\Psi(v)$

```
1  begin
2    G0 = ⟨V0, E0, d, t⟩ such that
3    V0 = V + {v0}, E0 = E − {e ∈ E : d(e) ≠ 0} + {v0 --e--> v, v ∈ V, d(e) = 0}
4    ∀u ∈ V0, Ψ(u) = 0, Queue = v0
5    while Queue ≠ ∅ do
6          get(u, Queue)
7          foreach u --e--> v do
8            indegree(v) = indegree(v) − 1
```

```
 9            Ψ(v) = max(Ψ(u) + T_v, Ψ(v))                    /* using Lemma 3.1, 3.2 */
10            if indegree(v) = 0 then put(v, Queue) fi
11          od
12    od
13 end
```

Line 2 produces a DAG G_0 from G containing only edges $e \in E$, with $d(e) = 0$, and the additional edges connecting v_0 to every other node $v \in V$. Line 4 initializes the $\Psi(v)$ value for each vertex v. Lines 5–12 compute the mrt by calculating the summation and the maximum operations. The time complexity of calculating the sum, and max in Line 9 is $\mathcal{O}(n^2)$ where n is the maximum number of values in the random variable representing distribution of the execution time T_v. Therefore, the running time of Algorithm 3.1 is $\mathcal{O}(n^2|V||E|)$.

Since we want to reduce the chance that the maximum reaching time of a vertex is greater than a desired clock period c, i.e., to maximize the probability that the maximum reaching time of a vertex is less than c, we implement Algorithm 3.2 to determine whether there exists such a retiming that yields an equivalent circuit with the desired clock period c within an acceptable confidence level θ.

Algorithm 3.2 (Probabilistic retiming)

Input: PG $G = \langle V, E, d, t\rangle$, a desired clock period c, and probability δ.
Output: Retiming function r if one exists.

```
 1 begin
 2   foreach vertex v ∈ V do r(v) = 0 od                      /* initialize retiming r */
 3   for i = 1 to |V| do
 4       G_r = Retime(G, r); flag = true                      /* retime graph G with the value r */
 5       G_0 = ⟨V_0, E_0, d, t⟩ such that V_0 = V + {v_0, v_d}; t(v_0) = t(v_d) = 0
 6       E_0 = E − {e ∈ E : d(e) ≠ 0} + {v_0 -e1-> v, v -e2-> v_d, v ∈ V, d(e1) = d(e2) = 0}
 7       ∀u ∈ V_0, Ψ(u) = 0, Queue = v_0
 8       while Queue ≠ ∅ do
 9             get(u, Queue)
10             foreach u -e-> v do
11               indegree(v) = indegree(v) − 1
12               Ψ(v) = max(Ψ(u) + T_v, Ψ(v))                /* using Lemma 3.1, 3.2 */
13               if P(Ψ(v) > c) > δ and v = v_d              /* user-defined condition */
14                  then r(u) = r(u) − 1; flag = false
15               elsif indegree(v) = 0 then put(v, Queue) fi od
16             if flag = true then break fi
17       od
18       if flag = true then Report r else no feasible solution fi
19   end
```

Algorithm 3.2 retimes a vertex, whose probability of propagation delay greater than c, is larger than the acceptable probability value. The algorithm adds a dummy node v_d that works as a joining point for all possible paths in the graph which guarantees that the computed mrt is the combination of the mrt of all those paths. Line 14 updates the retiming[3] function of such vertices. This process is equivalent to deleting

[3]Note that the retiming operation on edge $u \xrightarrow{e} v$ is given by $d_r(e) = d(e) + r(u) - r(v)$, where $d_r(e)$ is the number of registers on edge e after retiming.

a register from the outgoing edge(s) of vertex v and inserting it into the incoming edge(s) of the vertex v. The algorithm stops when there exists a retiming function satisfying the requirement, i.e., $\mathcal{P}(\Psi(v) > c) < \delta$. Otherwise, the algorithm repeats for at most $|V|$ times. The time complexity of this algorithm is $\mathcal{O}(n^2|V|^2|E|)$. The following theorem presents the correctness of Algorithm 3.2.

Theorem 3.4 *Given a PG* $G = \langle V, E, d, t\rangle$, *a desired clock period* c, *and a confidence level* θ, *if the retiming* r *exists such that the desired clock period* c *occurs with the probability* $\delta = 1 - \theta$ *then the Algorithm 3.2 computes such a retiming on at most* $|V|$ *iterations.*

Proof: Recall that Theorem 3.3 imposes the constraint that for every zero-register path $u \overset{p}{\rightsquigarrow} v$ satisfying $\mathcal{P}(\Psi(v) > c) > \delta$, at least one register must be inserted. Notice again that, in Line 13, the computation of $\Psi(v)$ is equivalent to testing $\Psi(v_d)$ for all paths from a dummy node v_0 to the node v_d assuming that the node v_0 has $T_{v_0} = 0$ and zero-register paths connect v_0 to v_d. For every iteration, in Line 13, Algorithm 3.2 attempts to preserve the condition $\mathcal{P}(\Psi(v_d) > c) > \delta$ by retiming the vertex u once $(r(u) = r(u) - 1)$. This is equivalent to satisfying the constraint in Theorem 3.3. Since, for every iteration, Algorithm 3.2 always reallocates one register with respect to a node in such a way that one register is deleted from all outgoing edges and added to all incoming edges of that node, and the longest path in the graph has a maximum of $|V| - 1$ edges, Algorithm 3.2 needs at most $|V|$ iterations to satisfy the constraint imposed by Theorem 3.3 before repeating the same register configuration. □

Choosing the desired clock period

The algorithm presented previously is neccessary to tell us whether a selected desired clock period is feasible. When a desired clock period is not specified, one might wish to know the approximate value that we should begin with. In order to establish a possible target c, some criterion must be considered. The *shape* of the pdf is an important factor that should be investigated. Figure 2 show some possible distribution shapes. Figures 2(b) and 2(c) present two *skewed* distributions. Figure 2(b) represents higher probability for most of the possible lower values. On the other hand, Figure 2(c) illustrates the mirror of the previous one. These two shapes should be carefully handled.

In the literature, the lower bound on clock period $\Phi(G)$ was presented by Renfors and Neuvo [15]. Such a bound is simply obtained from calculating the maximum ratio of execution time to register count for all loops in a circuit. The lower bound is given by

$$B(G) = \max_{\forall \text{ cycle } l \in G} \frac{\sum t(v \in l)}{\sum d(e \in l)},$$

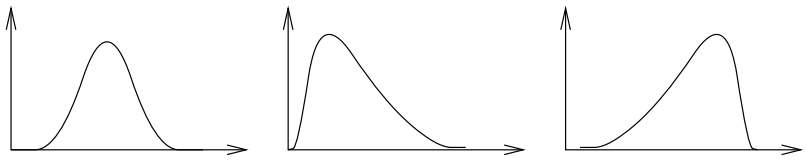
where $\sum t(v \in l)$ is the sum of propagation delays in loop l, and $\sum d(e \in l)$ is the sum of register counts in loop l. In order to use this bound to calculate the desired clock period, we need to transform our probabilistic propagation delays to some exact values.

Since the distribution in Figure 2(c) has higher probabilities for the higher possible value range, these numbers should be initialized to the maximum possible number of the distribution, i.e., worst case analysis. For the distributions in Figures 2(b) and 2(a), one might wish to initialize the propagation delays by using

the *expected value*,

$$E(T_v) = \sum_{i=1}^{\infty} \mathcal{P}(T_v = x_i) \times x_i.$$

After obtaining the graph with exact values of propagation delays, the approximate desired clock period can be computed. However, it is the designer decision to determine a proper acceptable probability δ to restrict the constraint, $\mathcal{P}(\Psi(v) > c) < \delta$, of Algorithm 3.2.



(a) normal (b) p-skewed (c) n-skewed

Figure 2: Distribution shapes of pdf

4 Experimental Results

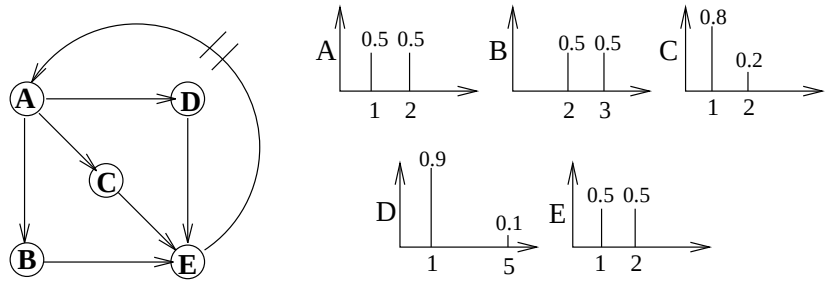
In this section we discuss some experimental results obtained from the algorithm proposed in Section 3.2. In order to illustrate the effectiveness of our algorithm, the results from the traditional retiming and the probabilistic retiming techniques are compared. First of all, let us demonstrate how the probabilistic retiming algorithm work.

4.1 Demonstration of the algorithm

Let us revisit the example presented in Section 1 with different propagation delay configurations. Figure 3 illustrates the *5*-node PG to be optimized. For simplicity, the dummy nodes v_0 and v_d are not shown. Since the shapes of the pdf of these vertices are either normal or p-skewed, we can choose the desired clock period c by initializing those random variables using an expected value method. For this experiment, we obtain $c = 3$. Let $\delta = 0.3$ be an acceptable probability. Algorithm 3.2 works by first checking the mrt $\Psi(v)$, for each vertex. Four iterations of the results of the maximum reaching time are tabulated in Figure 4.

Algorithm 3.2 reduces $\Psi(v)$ of the vertex that has the possible maximum reaching time greater than the desired cycle period. For example, in Figure 4(a), the mrt of vertices B and E have the value $p(> c) > 0.3$. Therefore, according to Line 14 in the algorithm, these vertices are retimed, $r(B) = r(E) = -1$. The retimed graphs corresponding to the following iterations are presented in Figure 5. Figures 5(a), 5(b), and 5(c) result in the maximum reaching time shown in Figures 4(b), 4(c), and 4(d) respectively.

As shown in Figure 4(d), all vertices that have an mrt value greater than 3 have a probability less than 0.3 for such an occurrence, which is satisfying the initial specification.



(a) PG (b) pdf

Figure 3: Revisited example

$v \in V$	$p(\psi) = \mathcal{P}(\Psi(v) = \psi)$				$r(v)$
	$p(1)$	$p(2)$	$p(3)$	$p(> c)$	
A	0.5	0.5	0	0	0
B	0	0	0.25	0.75	−1
C	0	0.4	0.5	0.1	0
D	0	0.45	0.45	0.1	0
E	0	0	0	1.0	−1

(a) first

$v \in V$	$p(\psi) = \mathcal{P}(\Psi(v) = \psi)$				$r(v)$
	$p(1)$	$p(2)$	$p(3)$	$p(> c)$	
A	0.5	0.5	0	0	0
B	0	0.5	0.5	0	−1
C	0	0.4	0.5	0.1	0
D	0	0.45	0.45	0.1	0
E	0	0	0.25	0.75	−2

(b) second

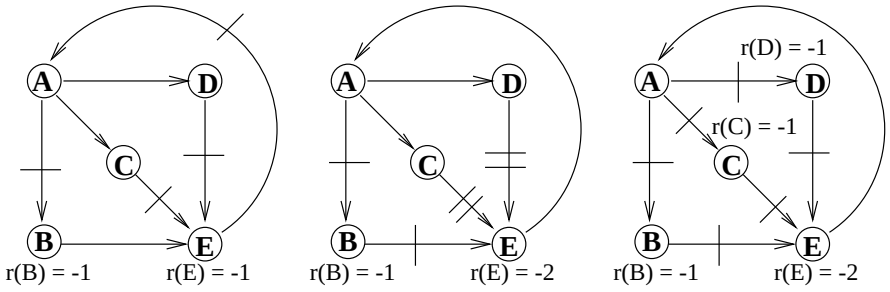
$v \in V$	$p(\psi) = \mathcal{P}(\Psi(v) = \psi)$				$r(v)$
	$p(1)$	$p(2)$	$p(3)$	$p(> c)$	
A	0	0.25	0.5	0.25	0
B	0	0.5	0.5	0	−1
C	0	0	0.2	0.8	−1
D	0	0	.225	.775	−1
E	0.5	0.5	0	0	−2

(c) third

$v \in V$	$p(\psi) = \mathcal{P}(\Psi(v) = \psi)$			
	$p(1)$	$p(2)$	$p(3)$	$p(> c)$
A	0	0.25	0.5	0.25
B	0	0.5	0.5	0
C	0.8	0.2	0	0
D	0.9	0	0	0.1
E	0.5	0.5	0	0

(d) forth

Figure 4: pdf of $\Psi(v)$, $v \in V$



(a) (b) (c)

Figure 5: Retimed graph corresponding to Figure 4

4.2 Benchmarks

In this section we presents the experimental results obtained from using our algorithm to optimize some well-known benchmarks. These benchmarks include the Biquadratic IIR filter, 3-stage direct from IIR filter, 4^{th}-order Jaunmann wave digital filter, 5^{th}-elliptic filter[4], All-pole lattice filter, Differential equation solver and Volterra filter. In order to perform the simulation, the distributions of the execution time for the basic components in those circuits were randomly selected. Tables 4 and 5 present the possible pdf of execution time for the adders and multipliers found in those filters.

	$\mathcal{P}(T_\Sigma = x)$						
	1	2	3	4	5	6	7
T_{Σ_1}	0.4	0.3	0	0.1	0.1	0	0.1
T_{Σ_2}	0.1	0.1	0.2	0.2	0.2	0.1	0.1
T_{Σ_3}	0.1	0	0.1	0	0.4	0	0.4

Table 4: Possible pdfs of execution time for addition

	$\mathcal{P}(T_\Pi = y)$								
	1	2	3	4	5	6	7	8	9
T_{Π_1}	0.5	0	0	0.2	0	0	0.1	0.1	0.1
T_{Π_2}	0.1	0.1	0.1	0.1	0.2	0.2	0.1	0	0.1
T_{Π_3}	0.1	0.1	0	0	0	0	0.3	0	0.5

Table 5: Possible pdfs of execution time for multiplication

We used our technique to optimize those benchmarks. The results shown in Table 6 discuss the difference between considering the worst case of the propagation delays (column worst) and considering the probabilistic model. Column worst in the table presents the optimized clock period obtained from applying the traditional retiming to those filters while considering the possible worst execution time of each adder (7) and multiplier (9). Columns 4–8 show the *feasible* desired clock period c with respect to the confidence level $1 - \delta = \theta$.

Benchmark	num. nodes	c worst	$\mathcal{P}(\Psi(v) \le c) \ge 1 - \delta = \theta$									
			$\theta = 0.5$		$\theta = 0.6$		$\theta = 0.7$		$\theta = 0.8$		$\theta = 0.9$	
			c	%	c	%	c	%	c	%	c	%
Biquad IIR	8	23	11	52	12	48	13	43	14	39	16	30
Diff. Equation	11	32	18	44	19	41	20	37	22	31	24	25
3-stage direct IIR	12	16	8	50	9	44	10	37	11	31	13	19
All-pole Lattice	15	46	28	39	30	35	31	33	32	30	35	24
4^{th} order WDF	17	46	23	50	25	46	26	43	28	39	30	35
Volterra	27	81	44	46	46	43	49	40	50	38	53	35
5^{th} Elliptic	34	97	60	38	62	36	63	35	66	32	69	29
5^{th} Elliptic (uf=4)	170	485	322	34	323	33	324	33	325	33	331	32
5^{th} Elliptic (uf=9)	340	970	644	34	646	33	648	33	650	33	662	32

Table 6: Probabilistic retiming versus worst case analysis

Notice that, for all benchmarks, the smallest feasible clock periods with $\theta = 0.9$ are still smaller than the number listed in Column 3. In order to illustrate the effectiveness of the algorithm, the column "%"

[4]We experimented Algorithm 3.2 on the original graph and the unfolded version by using unfolding factor (uf) 4 and 9.

presents the percent of the feasible clock period reduction with respect to the worst case analysis of c. With $\theta = 0.5$, all benchmarks have the percent reduction greater than 34% and larger than 19% for $\theta = 0.9$.

5 Conclusion

VLSI circuit manufacturing may result in devices with different propagation delays. Hence, the estimation of such delays during the design procedure may not prove totally accurate due to the fabrication process. We have presented the theoretical foundation and experimental results for a new transformation technique, called probabilistic retiming, which can be computed in $\mathcal{O}(n^2|V|^2|E|)$. Considering the realistic case, our algorithm can effectively optimize those circuits while utilizing the manufacturing probability information and the designer requirements which are a desired clock period c and the confidence level θ. This methodology are expected to have similar impact to many areas, e.g., high-level synthesis, loop scheduling, testing, etc., as the traditional retiming does.

References

[1] S. Dey, M. Potkonjak, and S. G. Rothweiler. Performance optimization of sequential circuits by eliminating retiming bottlenecks. In *Proceedings of the 1992 IEEE/ACM International Conference on Computer Aided Design*, pages 504–509, 1992.

[2] A. T. Ishii. Retiming gated-clocks and precharged circuit structures. In *Proceedings of the 1993 IEEE/ACM International Conference on Computer Aided Design*, pages 300–307, 1993.

[3] A. T. Ishii, C. E. Leiserson, and M. C. Papaefthymiou. Optimizing two-phase, level-clocked circuitry. In T. Knight and J. Savage, editors, *Advanced Research in VLSI and Parallel Systems: Proceedings of the 1992 Brown/MIT Conference*, pages 245–264, Cambridge, MA, 1992.

[4] I. Karkowski and R. H. J. M. Otten. Retiming synchronous circuitry with imprecise delays. In *Proceedings of the 32nd Design Automation Conference*, pages 322–326, San Francisco, CA, 1995.

[5] E. Kreyszig. *Advanced Engineering Mathematics*, chapter 23. John Wiley and Sons, New York, 6th edition, 1988.

[6] Y.-J. Lai and C.-L. Hwang. A new approach to some possibilistic linear programming problems. *Fuzzy Sets and Systems*, 49:121–133, 1992.

[7] K. N. Lalgudi and M. C. Papaefthymiou. DELAY: An efficient tool for retiming with realistic delay modeling. In *Proceedings of the 32nd Design Automation Conference*, pages 304–309, San Francisco, CA, 1995.

[8] C. E. Leiserson and J. B. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6:5–35, 1991.

[9] L.-T. Liu et al. Performance-driven partitioning using retiming and replication. In *Proceedings of the 1993 IEEE/ACM International Conference on Computer Aided Design*, pages 296–299, 1993.

[10] B. Lockyear and C. Ebeling. Optimal retiming of multi-phase, level-clocked circuits. In T. Knight and J. Savage, editors, *Advanced Research in VLSI and Parallel Systems: Proceedings of the 1992 Brown/MIT Conference*, pages 265–280, Cambridge, MA, 1992.

[11] B. Lockyear and C. Ebeling. The practical application of retiming to the design of high-performance systems. In *Proceedings of the 1993 IEEE/ACM International Conference on Computer Aided Design*, pages 288–295, 1993.

[12] S. Malik et al. Retiming and resynthesis: Optimizing sequential networks with combinational techniques. *IEEE Transactions on Computer-Aided Design*, 10(1):74–84, January 1991.

[13] P. L. Meyer. *Introductory Probability and Statistical Applications.* Addison-Wesley, Reading, MA, 2nd edition, 1979.

[14] M. C. Papaefthymiou. Understanding retiming through maximum average-delay cycles. *Mathematical Systems Theory*, 27:65–84, 1994.

[15] M. Renfors and Y. Neuvo. The maximum sampling rate of digital filters under hardware speed constraints. *IEEE Transactions on Circuits and Systems*, CAS-28:196–202, 1981.

[16] N. Shenoy and R. K. Brayton. Retiming of circuits with single phase transparent latches. In *Proceedings of the 1991 International Conference on Computer Design*, pages 86–89, 1991.

[17] T. Soyata and E. Friedman. Retiming with non-zero clock skew, variable register and interconnect delay. In *Proceedings of the 1994 IEEE/ACM International Conference on Computer Aided Design*, November 1994.

[18] T. Soyata, E. Friedman, and J. Mulligan. Integration of clock skew and register delays into a retiming algorithm. In *Proceedings of the International Symposium on Circuits and Systems*, pages 1483–1486, May 1993.

[19] L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.