Performance Analysis and Optimization of Asynchronous Circuits Produced by Martin Synthesis

Steven M. Burns

Computer Science Department
California Institute of Technology

Caltech-CS-TR-90-12

Performance Analysis and Optimization of Asynchronous Circuits Produced by Martin Synthesis[1]

Steven M. Burns

Computer Science Department
California Institute of Technology
Pasadena, CA 91125 USA

Abstract

We present a method for analyzing the timing performance of asynchronous circuits, in particular, those derived by program transformation from concurrent programs using the synthesis approach developed by Martin. The analysis method produces a performance metric (related to the time needed to perform an operation) in terms of the primitive gate delays of the circuit. Because the gate delays are functions of transistor sizes, the performance metric can be optimized with respect to these sizes. For a large class of asynchronous circuits — including those produced by Martin synthesis — these techniques produce the global optimum of the performance metric. A CAD tool has been implemented to perform this optimization.

1 Introduction

Performance analysis of a synchronous computer system is simplified by an external clock that partitions the events in the system into discrete segments. In asynchronous systems, no such quantization exists. Instead, the operation of the system proceeds at a rate determined by: the speed of its individual components, and sequencing of the operation of the components. Unlike the synchronous case, the time needed to perform an asynchronous computation cannot be determined by merely counting the number of clock cycles required and multiplying by the clock period. Instead, to determine the time required to perform the computation as a whole, the times of those individual components of the computation that must occur sequentially are summed.

[1]Presented at *TAU '90*, the *1990 ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, August 14–17, 1990, Vancouver, BC, Canada

The techniques required to analyze asynchronous systems resemble those used to determine the clock period of a synchronous system; that of summing the delays along the longest path through the combinational logic connecting adjacent latches. In the clocked case, the critical path has a clear beginning and a clear end because all paths are broken by latches. No clear separation is available in asynchronous systems. Analysis procedures must deal directly with cyclic critical paths, and thus existing critical path analysis tools such as CRYSTAL[9] cannot be easily applied to this problem.

This paper discusses a framework for determining the time needed to perform computations using asynchronous systems, and applies especially to repetitive computations. Previous work in the area of timed Petri nets [10, 5] applies to this problem as well. The results we describe here are based on event-rule systems, a different formalism that is more closely connected to the methods we use to synthesize the asynchronous systems. Furthermore, we use our formalism to model the performance of asynchronous circuits and provide a method for optimizing such circuits for performance.

Martin ([6] and elsewhere) has developed a synthesis method whereby asynchronous circuits are produced from concurrent program descriptions. By applying a systematic series of semantics-preserving transformations, a high-level description (CSP program) is refined, using the intermediate forms of handshaking expansions and production rules, until a provably correct asynchronous CMOS circuit is constructed.

At each stage of the synthesis procedure, a variety of transformations can potentially be applied. In the automated compiler of [1], these choices are made so that the same subcircuit template can be used to implement each instance of the same CSP language construct. Instances of these small templates are composed together to form a correct circuit implementing the original CSP program. However, in order to produce high-performance circuits, these choices must be directed by performance concerns. We observed this potential benefit of performance-directed transformations during the design of the Caltech Asynchronous Microprocessor[7]. The decisions of what transformation to apply were based on performance goals and this accounts for its high-performance.

Event-rule (*ER*) systems can be used at each stage of the synthesis procedure to analyze the potential performance of the current refinement. Given a trace of the execution of a complete, closed program (environment included), an *ER* system can be generated from any of the intermediate forms: CSP

programs, handshaking expansions, production rules, or CMOS circuits. The trace of execution is used to unroll each process that contains guarded commands into a straight-line process. In the cases where the trace of execution repeats, a repetitive *ER* system can be generated. The cycle period (the time between repeated events) can be determined using the techniques explained in Section 2.

These techniques provide an expression for the cycle period in terms of maximums and sums of individual component delays. At the circuit level, the component delays are functions of transistor widths and, as such, the cycle period can be optimized with respect to these widths. Nonlinear optimization methods (such as those used in TILOS[3] and EPOXY[8]) can be used to perform the optimization of this expression for the cycle period. Our approach differs from those used for synchronous systems because we optimize all critical paths simultaneously.

2 Event-Rule Systems

An *event-rule (ER) system*, is a pair $\langle E, R \rangle$, where:

E is a set of events, and

R is a set of rules defining the timed causal dependencies between the events. Each $r \in R$ is written $e \overset{\alpha}{\mapsto} f$, where

$e \in E$ is the *source* of r,

$f \in E$ is the *target* of r, and

$\alpha \in [0, +\infty)$ is the *delay* of r.

Neither E nor R need be finite. When R is infinite, we require that no event is the target of an infinite number of rules. Sometimes it is convenient to view $\langle E, R \rangle$ as a directed graph (multiple arcs and self-loops allowed); this graph will be referred to as the *constraint graph* G. For a given $\langle E, R \rangle$, there is a (possibly empty) set of functions T, that satisfies:

T is a subset of the functions from E to $[0, +\infty)$;

$t \in T$ if and only if

$$t(f) \geq t(e) + \alpha \text{ for every } e \overset{\alpha}{\mapsto} f \in R \,. \tag{1}$$

We call a function t in the set T a *timing function* of $\langle E, R\rangle$. Each t represents a possible or consistent timing specification for the events of the system. If the set T is empty, the constraints (1) cannot be satisfied by any such function t. In this case, the $\langle E, R\rangle$ is called *infeasible*; otherwise, it is called *feasible.*

Example 2.1 Consider the $\langle E, R\rangle$ with:

$$\begin{aligned} E &= \{a, b, c\} \\ R &= \{a \overset{\alpha_b}{\mapsto} b, b \overset{\alpha_a}{\mapsto} a, b \overset{\alpha_c}{\mapsto} c\} \end{aligned}$$

This ER system is feasible if and only if $\alpha_a = 0$ and $\alpha_b = 0$.

The smallest timing function denotes the earliest time at which the events of E can execute. Any feasible ER system with cyclic constraints or zero delay rules can be transformed into an equivalent one that satisfies the hypotheses of Lemma 2.1.

Lemma 2.1 If $\langle E, R\rangle$ is feasible, the constraint graph G is acyclic, and $\alpha > 0$ for every rule in R, then there exists a unique function $\hat{t} \in T$ such that for every $t \in T$,

$$\hat{t}(e) \leq t(e) \text{ for every } e \in E. \tag{2}$$

We call $\hat{t}$ the *timing simulation* of $\langle E, R\rangle$.

Proof: We propose the following recursive definition for $\hat{t}$:

$$\hat{t}(f) = \begin{cases} 0 & \text{if } \{e \mid e \overset{\alpha}{\mapsto} f \in R\} = \emptyset \\ \max\{\hat{t}(e) + \alpha \mid e \overset{\alpha}{\mapsto} f \in R\} & \text{otherwise} \end{cases} \tag{3}$$

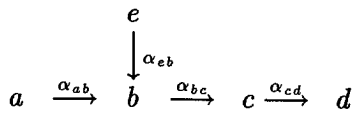
Such a function is well-defined, because G is acyclic and thus there are no circular dependencies between the events in E.

We show, by contradiction, that this $\hat{t}$ satisfies (2). Pick a t such that the set F of events e that satisfy $t(e) < \hat{t}(e)$ is non-empty. Let $f \in F$ have the smallest $\hat{t}(f)$. Then for some $e \overset{\alpha}{\mapsto} f \in R$,

$$t(f) < \hat{t}(f) = \hat{t}(e) + \alpha \leq t(e) + \alpha \leq t(f) \quad .$$

The equality of the previous line follows by choosing $e \overset{\alpha}{\mapsto} f$ as the rule that achieves the maximum in (3). The inequality $\hat{t}(e) + \alpha \leq t(e) + \alpha$ follows, since $e \notin F$; that is, $\hat{t}(e)$ is strictly less than $\hat{t}(f)$. The last inequality holds by (1). Thus, F is empty and $\hat{t}$ satisfies (2). ∎

Example 2.2 The *ER* system defined by the constraint graph:



has the timing simulation:

$$\begin{aligned}
\hat{t}(a) &= 0 \\
\hat{t}(e) &= 0 \\
\hat{t}(b) &= \max(\alpha_{ab}, \alpha_{eb}) \\
\hat{t}(c) &= \max(\alpha_{ab}, \alpha_{eb}) + \alpha_{bc} \\
\hat{t}(d) &= \max(\alpha_{ab}, \alpha_{eb}) + \alpha_{bc} + \alpha_{cd}
\end{aligned}$$

2.1 Repetitive Systems

ER systems of unbounded size constructed from finite circuits can be represented by a finite set of events that are repeated infinitely often. Let the event set E be generated from the finite set E' by

$$E = E' \times \mathbb{N}.$$

The elements of the finite set R' are quadruples:

$$\langle u, v, \alpha, \varepsilon \rangle \in R', \text{ where } R' \subseteq E' \times E' \times [0, +\infty) \times \mathbb{Z},$$

which we will write as

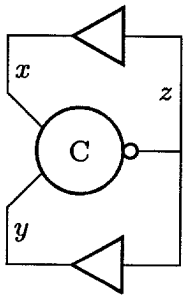
$$\langle u, i - \varepsilon \rangle \overset{\alpha}{\mapsto} \langle v, i \rangle.$$

The set R is the set of all instantiations of the rules $r \in R'$ with $i \geq \max\{0, \varepsilon\}$.

We define the *collapsed constraint graph* G' of $\langle E', R' \rangle$ as the directed graph with nodes from E' and arcs from R'.

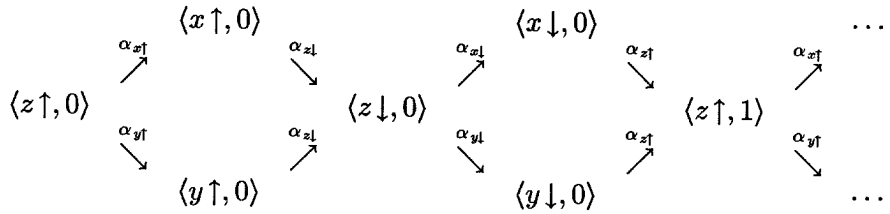
Example 2.3 Consider the repetitive *ER* system constructed from a circuit con-

taining a single Muller C-element:



$$
\begin{aligned}
E' &= \{x\uparrow, y\uparrow, z\uparrow, x\downarrow, y\downarrow, z\downarrow\} \\
R' &= \{\ \langle x\downarrow, i-1\rangle \overset{\alpha_{x\uparrow}}{\longmapsto} \langle z\uparrow, i\rangle, \\
&\qquad \langle y\downarrow, i-1\rangle \overset{\alpha_{x\uparrow}}{\longmapsto} \langle z\uparrow, i\rangle, \\
&\qquad \langle z\uparrow, i\rangle \overset{\alpha_{x\uparrow}}{\longmapsto} \langle x\uparrow, i\rangle, \\
&\qquad \langle z\uparrow, i\rangle \overset{\alpha_{y\uparrow}}{\longmapsto} \langle y\uparrow, i\rangle, \\
&\qquad \langle x\uparrow, i\rangle \overset{\alpha_{z\downarrow}}{\longmapsto} \langle z\downarrow, i\rangle, \\
&\qquad \langle y\uparrow, i\rangle \overset{\alpha_{z\downarrow}}{\longmapsto} \langle z\downarrow, i\rangle, \\
&\qquad \langle z\downarrow, i\rangle \overset{\alpha_{x\downarrow}}{\longmapsto} \langle x\downarrow, i\rangle, \\
&\qquad \langle z\downarrow, i\rangle \overset{\alpha_{y\downarrow}}{\longmapsto} \langle y\downarrow, i\rangle \\
&\ \}
\end{aligned}
$$

The repeated events (those events generated from E') represent the occurrence of transitions of circuit variables. The event $\langle x\uparrow, i\rangle$ represents the i^{th} repetition (or occurrence) of a transition from $x =$ **false** to $x =$ **true**. Similarly, $\langle x\downarrow, i\rangle$ represents the i^{th} repetition of a transition from $x =$ **true** to $x =$ **false**. The repeated rules correspond to dependencies introduced by the inverters and the C-element that make up the circuit. We can represent the infinite sets E and R graphically:



Notice that event $\langle z\uparrow, 0\rangle$ has no predecessors. In the timing simulation, $\hat{t}(\langle z\uparrow, 0\rangle)$ is set to 0. (For ease of notation, $\hat{t}(\langle z\uparrow, 0\rangle)$ will sometimes be written as $\hat{t}(z\uparrow, 0)$.) The entire timing simulation $\hat{t}$, which can be constructed by inspection from the constraint graph, is:

$$
\begin{aligned}
\hat{t}(z\uparrow, i) &= pi \\
\hat{t}(x\uparrow, i) &= \alpha_{x\uparrow} + pi \\
\hat{t}(y\uparrow, i) &= \alpha_{y\uparrow} + pi \\
\hat{t}(z\downarrow, i) &= \max(\alpha_{x\uparrow}, \alpha_{y\uparrow}) + \alpha_{z\downarrow} + pi \\
\hat{t}(x\downarrow, i) &= \max(\alpha_{x\uparrow}, \alpha_{y\uparrow}) + \alpha_{z\downarrow} + \alpha_{x\downarrow} + pi \\
\hat{t}(y\downarrow, i) &= \max(\alpha_{x\uparrow}, \alpha_{y\uparrow}) + \alpha_{z\downarrow} + \alpha_{y\downarrow} + pi
\end{aligned}
$$

where $p = \max(\alpha_{x\uparrow}, \alpha_{y\uparrow}) + \alpha_{z\downarrow} + \max(\alpha_{x\downarrow}, \alpha_{y\downarrow}) + \alpha_{z\uparrow}$.

2.2 Linear Timing Functions

In the previous example, we saw that the timing simulation of a repetitive *ER* system took on a simple form that is linear in the occurrence index i. This is not the case for all repetitive *ER* systems. However, as we now show, a *linear timing function* exists whenever the timing simulation exists, and the "best" such function will be a good approximation of the timing simulation.

We call $\bar{t} \in T$ a *linear timing function* of $\langle E', R' \rangle$, if

$$\bar{t}(v,i) = x_v + p_v i \text{ for every } v \in E' \text{ and } i \in \mathbb{N}. \tag{4}$$

Each x_v and p_v are independent of i. For each $v \in E'$, x_v and p_v are called, respectively, the *offset* and *cycle period* of the repeated event v.

Because of the linear form of $\bar{t}$, the timing function constraints, (1), reduce to linear inequalities in the offsets and cycle periods of the events. All dependence on the occurrence index i can be eliminated. For each rule $r = \langle u, i-\varepsilon \rangle \overset{\alpha}{\mapsto} \langle v, i \rangle \in R'$, we have the infinite set of constraints:

$$\bar{t}(v,i) \geq \bar{t}(u, i-\varepsilon) + \alpha \ , \text{ for each } i \geq \max(0, \varepsilon)$$

Replacing $\bar{t}$ by its definition (4), we get

$$\begin{aligned} x_v + p_v i &\geq x_u + p_u(i-\varepsilon) + \alpha \\ x_v &\geq x_u - p_u\varepsilon + \alpha + (p_u - p_v)i \ . \end{aligned}$$

The preceding equations can never be satisfied for all i when $p_u > p_v$. Thus, the infinite set of constraints generated by r can be replaced by the two inequalities,

$$x_v \geq x_u - p_u\varepsilon + \alpha \ , \text{ and} \tag{5}$$

$$p_v \geq p_u \ . \tag{6}$$

From (6) we see that for a feasible solution to exist, a partial ordering between the p_v's must be satisfied. If two nodes, u and v, are in the same cycle of the collapsed constraint graph G', then p_u must equal p_v. All events in the same strongly connected component of G' have the same cycle period. In the following, we consider only those repetitive *ER* systems in which G' is strongly connected, and p is used to denote the cycle period of every element in E'.

2.3 Linear Programming

Among the possible linear timing functions, there are those that minimize the cycle period p. The techniques of linear programming[4] can be used to find such a minimum-period, linear timing function.

The constraints of a linear timing function, (5), are simple linear inequalities in the x_v's and p. By ordering the sets E' and R', we can construct a linear program in matrix form:

$$\left.\begin{array}{rcl} \min 0^T x + 1^T p & = & z \\ A'x + \varepsilon p & \geq & \alpha \\ x, p & \geq & 0 \end{array}\right\} \tag{7}$$

The matrix A' is the edge-vertex incidence matrix of the collapsed constraint graph G'. If row j of A' represents the constraint $r_j = \langle u, i-\varepsilon\rangle \overset{\alpha}{\mapsto} \langle v, i\rangle \in R'$, and column k of A' represents the event $u_k \in E'$, then

$$a'_{jk} = \begin{cases} -1 & \text{if } u_k = u \\ 1 & \text{if } u_k = v \\ 0 & \text{otherwise} \end{cases} .$$

The jth elements of the (column) vectors ε and α are the scalar quantities ε and α of the constraint r_j, respectively.

Example 2.4 Consider the repetitive *ER* system:

$$\begin{array}{rcl} E' & = & \{lo\uparrow, li\uparrow, ro\uparrow, ri\uparrow, lo\downarrow, li\downarrow, ro\downarrow, ri\downarrow\} \\ R' & = & \{\ \langle li\downarrow, i-1\rangle \overset{\alpha_{lo\uparrow}}{\mapsto} \langle lo\uparrow, i\rangle, \\ & & \ \ \langle ro\downarrow, i-1\rangle \overset{\alpha_{lo\uparrow}}{\mapsto} \langle lo\uparrow, i\rangle, \\ & & \ \ \langle li\uparrow, i\rangle \overset{\alpha_{lo\downarrow}}{\mapsto} \langle lo\downarrow, i\rangle, \\ & & \ \ \langle ri\uparrow, i\rangle \overset{\alpha_{ro\uparrow}}{\mapsto} \langle ro\uparrow, i\rangle, \\ & & \ \ \langle lo\downarrow, i\rangle \overset{\alpha_{ro\uparrow}}{\mapsto} \langle ro\uparrow, i\rangle, \\ & & \ \ \langle ri\downarrow, i\rangle \overset{\alpha_{ro\downarrow}}{\mapsto} \langle ro\downarrow, i\rangle, \\ & & \ \ \langle lo\uparrow, i\rangle \overset{\alpha_{li\uparrow}}{\mapsto} \langle li\uparrow, i\rangle, \\ & & \ \ \langle lo\downarrow, i\rangle \overset{\alpha_{li\downarrow}}{\mapsto} \langle li\downarrow, i\rangle, \\ & & \ \ \langle ro\downarrow, i-1\rangle \overset{\alpha_{ri\uparrow}}{\mapsto} \langle ri\uparrow, i\rangle, \\ & & \ \ \langle ro\uparrow, i\rangle \overset{\alpha_{ri\downarrow}}{\mapsto} \langle ri\downarrow, i\rangle \\ & & \} \end{array}$$

In this case, equation (7) becomes:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{lo\uparrow} \\ x_{li\uparrow} \\ x_{ro\uparrow} \\ x_{ri\uparrow} \\ x_{lo\downarrow} \\ x_{li\downarrow} \\ x_{ro\downarrow} \\ x_{ri\downarrow} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} p \geq \begin{pmatrix} \alpha_{lo\uparrow} \\ \alpha_{lo\uparrow} \\ \alpha_{lo\downarrow} \\ \alpha_{ro\uparrow} \\ \alpha_{ro\uparrow} \\ \alpha_{ro\downarrow} \\ \alpha_{li\uparrow} \\ \alpha_{li\downarrow} \\ \alpha_{ri\uparrow} \\ \alpha_{ri\downarrow} \end{pmatrix}$$

The duality theorem of Linear Programming relates the primal program (7) to the dual program:

$$\left.\begin{array}{rcl} \max y^T\alpha & = & w \\ y^T A' & \leq & 0^T \\ y^T\varepsilon & \leq & 1^T \\ y & \geq & 0 \end{array}\right\} \tag{8}$$

If both the primal and the dual programs have optimal solutions, then the optimal value z of the primal equals the optimal value w of the dual. We solve this dual program (8) in order to determine the cycle period.

2.4 Cycle Vectors of a Graph

A *cycle* of length ℓ in a directed graph, $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ (multiple edges and self-loops allowed), is an ordered subset $C = (c_0, c_1, \ldots, c_{\ell-1})$ of the edges $\mathcal{E}$ such that $target(c_{k-1}) = source(c_k)$ for all $0 < k < \ell$ and $target(c_{\ell-1}) = source(c_0)$. The cycle C can be represented by a *cycle* vector u, a $\{0,1\}$-vector of length $|\mathcal{E}|$, where $u_j = 1$ if and only if the j^{th} edge of $\mathcal{E}$ is in the set C. For each cycle vector u, $u^T A' = 0^T$, where A' is the edge-vertex incidence matrix of the graph $\mathcal{G}$. The following lemma relates the cycle vectors to an arbitrary vector y satisfying $y^T A' \leq 0^T$.

Lemma 2.2 Let U_i, $0 \leq i < q$ denote the cycle vectors of a graph with edge-vertex incidence matrix A'. Then, if $y \geq 0$ is such that $y^T A' \leq 0^T$, there exist scalars $\theta_i \geq 0$, $0 \leq i < q$ such that

$$y = \theta_0 U_0 + \theta_1 U_1 + \ldots + \theta_{q-1} U_{q-1} \quad . \tag{9}$$

Proof: See [2] for complete proof. Follows by induction on the number of cycles in the graph of A'. ∎

This lemma provides a straightforward means of determining the minimum cycle period p. By enumerating every cycle in G, and computing the delay around that cycle, we can find p.

Theorem 2.3 The minimum cycle period p, or equivalently, the optimal value w of the dual program (8), is

$$\max\left\{\left.\frac{U_k^T\alpha}{U_k^T\varepsilon}\right| \text{ for all cycle vectors } U_k\right\} . \tag{10}$$

Proof: Let U be the cycle matrix constructed by concatenating the (column) cycle vectors $U_0, U_1, \ldots, U_{q-1}$. By construction, $U^TA' \leq 0$ (actually equality). By lemma 2.2, any $y \geq 0$ with $y^TA' \leq 0^T$ can be represented as the product $U\Theta$, where the vector Θ has non-negative elements. The dual program (8) reduces to:

$$\begin{aligned} z &= \max \Theta^T(U^T\alpha) \\ \Theta^T(U^T\varepsilon) &\leq 1 \\ \Theta &\geq 0 \end{aligned}$$

The dual of the reduced dual is easily solved:

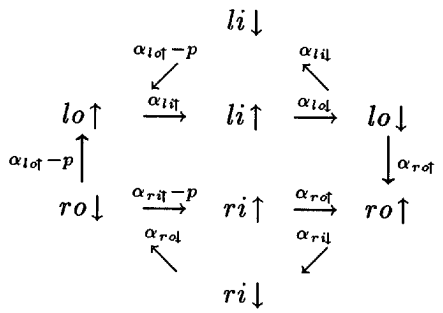
$$z = \min p \tag{11}$$

$$(U^T\varepsilon)p \geq (U^T\alpha) \tag{12}$$

$$p \geq 0 \tag{13}$$

The smallest scalar p, which satisfies the vector inequality (12) yields the desired minimum cycle period. ∎

Example 2.5 The minimum cycle period of the previous example can be constructed as follows: The collapsed constraint graph G' is:



In the graphical representation, each constraint r is labeled with $\alpha_r - \varepsilon_r p$. Summing the delays along the three cycles through G',

$$\begin{aligned} C_0 &= (lo\uparrow, li\uparrow, lo\downarrow, ro\uparrow, ri\downarrow, ro\downarrow), \\ C_1 &= (lo\uparrow, li\uparrow, lo\downarrow, li\downarrow) \text{ and} \\ C_2 &= (ro\uparrow, ri\downarrow, ro\downarrow, ri\uparrow), \end{aligned}$$

yields:

$$\begin{aligned} p_0 &= \alpha_{lo\uparrow} + \alpha_{li\uparrow} + \alpha_{lo\downarrow} + \alpha_{ro\uparrow} + \alpha_{ri\downarrow} + \alpha_{ro\downarrow} \\ p_1 &= \alpha_{lo\uparrow} + \alpha_{li\uparrow} + \alpha_{lo\downarrow} + \alpha_{li\downarrow} \\ p_2 &= \alpha_{ro\uparrow} + \alpha_{ri\downarrow} + \alpha_{ro\downarrow} + \alpha_{ri\uparrow} \end{aligned}$$

By Theorem 2.3, $p = \max(p_0, p_1, p_2)$.

2.5 Approximating the Timing Simulation

We now show that a minimum-period linear timing function provides an accurate approximation to the timing simulation.

Theorem 2.4 Let $\bar{t}$ and $\hat{t}$ be a minimum-period linear timing function and the timing simulation, respectively, of a connected repetitive ER system. There exists a finite B such that for all $u \in E'$ and all $i \geq 0$

$$s_{u,i} = \bar{t}(u,i) - \hat{t}(u,i) \leq B \quad .$$

Proof: By definition for each u and i

$$\begin{aligned} \bar{t}(u,i) &= x_u + pi \\ \hat{t}(u,i) &= x_u + pi - s_{u,i} \quad . \end{aligned}$$

Each $s_{u,i}$ is nonnegative because $\hat{t}$ is the smallest timing function. For the constraints generated from $r = \langle u, i-\varepsilon \rangle \overset{\alpha}{\mapsto} \langle v, i \rangle \in R'$, we define the nonnegative slack variables $\hat{z}_{r,i}$ and $\bar{z}_r$, thus transforming inequalities into equalities:

$$x_u - p\varepsilon + \alpha + \bar{z}_r = x_v \tag{14}$$

$$x_u + p(i-\varepsilon) - s_{u,i-\varepsilon} + \alpha + \hat{z}_{r,i} = x_v + pi - s_{v,i} \tag{15}$$

By subtracting these equations and simplifying, we get

$$\bar{z}_r - \hat{z}_{r,i} = s_{v,i} - s_{u,i-\varepsilon} \quad . \tag{16}$$

From Theorem 2.3, $p\sum_{r\in C}\varepsilon_r = \sum_{r\in C}\alpha_r$ for at least one cycle C. Adding the constraints on $\bar{t}$, (14), for each $r \in C$, we see that

$$\sum_{r\in C} x_{u_r} - p\sum_{r\in C}\varepsilon_r + \sum_{r\in C}\alpha_r + \sum_{r\in C}\bar{z}_r = \sum_{r\in C} x_{v_r} \quad .$$

Since along any cycle $\sum_{r\in C} x_{u_r} = \sum_{r\in C} x_{v_r}$, we have for all $r \in C$,

$$\bar{z}_r = 0 \quad .$$

By (16), $s_{u,i-\varepsilon} \geq s_{v,i}$ for all $i \geq \max(0,\varepsilon)$ and all u, v on cycle C. By summing along the cycle C, we see that for each $u \in C$ and $i' \geq 0$

$$s_{u,i'} \geq s_{u,i} \text{ where } i = i' + \sum_{r\in C}\varepsilon_r \quad .$$

Therefore, we can bound $s_{u,i}$, for every $u \in C$, by

$$B' = \max\left\{ s_{u,i'} \,\middle|\, u \in C \wedge i' < \sum_{r\in C}\varepsilon_r \right\} \quad .$$

For any event, h, not on cycle, C, we find a path, P_h, to this event from an event g on C. Because G' is strongly connected, such a path must exist and be independent of i. Then, by summing (16) along that path, we get for all $i, i' \geq 0$

$$s_{g,i'} + \sum_{r\in P_h}\bar{z}_r \geq s_{h,i} \quad ,$$

where $i = i' + \sum_{r\in P_h}\varepsilon_r$. But $\sum_{r\in P_h}\bar{z}_r$ is independent of i; thus, $s_{h,i}$ is bounded by a quantity that does not increase with successive occurrences. Thus, every $s_{h,i}$ with $h \notin C$ is bounded by B where

$$B \geq \max\left\{ s_{h,i} \,\middle|\, h \notin C \wedge i < \sum_{r\in P_h}\varepsilon_r \right\} \text{, and}$$

$$B \geq B' + \max\left\{ \sum_{r\in P_h}\bar{z}_r \,\middle|\, h \notin C \right\} \quad . \blacksquare$$

3 Performance Optimization

Using the above analysis method, a performance metric (the minimum cycle period p) can be expressed in terms of the primitive delays of an *ER* system. If the *ER* system is modeling a CMOS circuit, these primitive delays are determined by transistor widths. Adjusting the transistor widths affects the performance metric, but the nature of the dependence is completely encapsulated by the expression for the minimum cycle period. Minimizing the expression for p in terms of the transistor widths yields an optimally sized circuit.

3.1 Tau Model

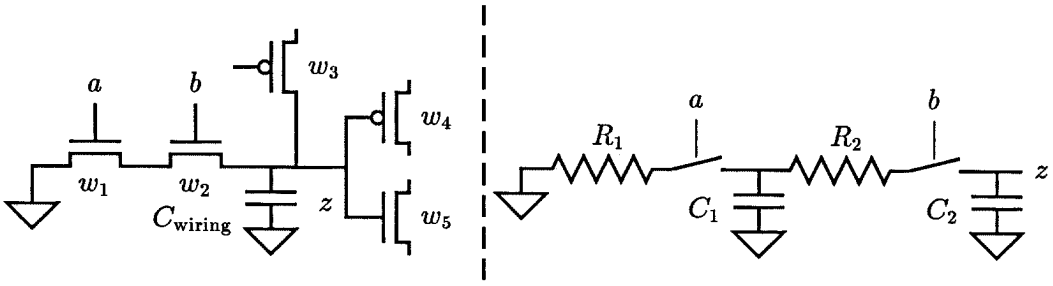


Figure 1: Linear approximation of a CMOS pulldown.

A simple RC switch model is used to relate each individual delay α, to the widths of circuit's transistors (w's). Each transistor is modeled as a switch with a resistance inversely proportional to its width. The gate of a transistor has a capacitance to ground proportional to its width. Source and drain capacitances are also proportional to transistor widths. Thus, the delays between $a\uparrow$ and $z\downarrow$, and $b\uparrow$ and $z\downarrow$, of the circuit shown in Figure 1 are modeled as:

$$\begin{aligned}
\alpha_{a\uparrow z\downarrow} &= R_1C_1 + (R_1 + R_2)C_2 \\
\alpha_{b\uparrow z\downarrow} &= (R_1 + R_2)C_2 \\
R_1 &= \mu/w_1 \\
R_2 &= \mu/w_2 \\
C_1 &= K_{\mathrm{ds}}(w_1 + w_2)
\end{aligned}$$

$$C_2 \;=\; K_{\rm ds}(w_2 + w_3) + C_{\rm wiring} + K_{\rm g}(w_4 + w_5) \;\;,$$

where μ is a constant that describes the differing per-unit-width strengths of the n- and p-channel transistors, $K_{\rm ds}$ is the per-unit-width capacitance contributed by the drain or source terminals, $K_{\rm g}$ is the per-unit-width gate capacitance and $C_{\rm wiring}$ is the capacitance contributed by wiring. All capacitances are expressed in terms of transistor width and thus $K_{\rm g} = 1$. The delays (α's) are in units of τ, the time needed for a unit-width n-channel transistor to switch a unit-width load. (Thus, $\mu_{\rm n} = 1$, $\mu_{\rm p} > 1$.) The values of $K_{\rm ds}$ and $C_{\rm wiring}$ are not constant, but depend on the final circuit layout that depends weakly on the transistor widths. This dependence is normally small and is ignored in the optimization problem.

3.2 Convex Objective Function

Every α derived using this simple model (and also many more accurate ones) is a posynomial functions (polynomial with positive coefficients and positive variables) of the transistor widths w's, and thus a convex function of the $\log w$'s[4]. Because both the sum and the maximum of two convex functions are convex functions, the resulting expression for p is a convex function of the $\log w$'s; and, thus, each minimum of p is global. The addition of convex constraints, for example, to limit energy usage or to bound transistor sizes, does not alter the unique minimum property.

We have implemented a program for solving the resulting nonlinear, non-differentiable, convex optimization problems based on the *subgradient* techniques described by Shor[11]. Table 1 lists the results of this program when applied to a variety of circuits. The column $n_{\rm trans}$ denotes the number of transistors in the circuit, and thus the number of free variables in the optimization problem. The columns $p_{\rm unsized}$ and $p_{\rm sized}$ show the cycle period in units of τ of the circuit before and after optimization. In the unsized case, all transistors have equal sizes. The CPU column denotes the number of CPU seconds needed to compute the optimum value on a SUN/Sparcstation 1. The performance metric of the sized circuit is generally 30 percent faster than the unsized circuit. A direct implementation of the optimization algorithm requires $O(n_{\rm trans}^2 + n_{\rm cycles}\ell_{\rm max})$ arithmetic operations per iteration, where $n_{\rm cycles}$ is the number of cycles used to form the cycle period function and $\ell_{\rm max}$ is the maximum number of edges per cycle. A more sophisticated

implementation that does not require enumeration of all cycles is described in [2] and requires only $O(n_{\text{trans}}^2)$ arithmetic operations per iteration. In this case, the linear program for the cycle period is solved directly, at each iteration, by a special-purpose algorithm.

4 Summary

We have demonstrated a method for determining the performance of circuits described by event-rule systems. Furthermore, we have shown how to optimally size transistors in such circuits. What we have not shown, due to lack of space, is how to transform the specifications of asynchronous circuits that we use for synthesis into *ER* systems. With the addition of these techniques, we have a complete method combining synthesis and performance analysis. The performance analysis can be done early and at each level of the synthesis procedure and can be used to guide the synthesis of efficient circuits. A complete description is given in [2].

5 Acknowledgments

I wish to thank Alain Martin, Pieter Hazewindus, Marcel Van der Goot, Drazen Brokovic, Tony Lee, Jose Tierno and Mass Sivilotti for their comments on this manuscript. The research described in this paper is sponsored by an IBM Graduate Fellowship and by the Defense Advanced Research Projects Agency, DARPA Order number 6202, monitored by the Office of Naval Research under contract number N00014–87–K–0745.

	n_{trans}	p_{unsized}	p_{sized}	CPU (s)
Three stage pipeline control	59	189	143	42
Ten stage pipeline control	192	189	151	190
Ten stage pipeline control*	192	189	151	95
Simple microprocessor control*	285	646	430	369
* indicates results generated by the special-purpose algorithm				

Table 1: Performance of optimization tool.

References

[1] Burns, Steven M., and Martin, Alain J., "Syntax-directed Translation of Concurrent Programs into Self-timed Circuits," in J. Allen and F. Leighton (eds), *Fifth MIT Conference on Advanced Research in VLSI*, pp. 35–50, MIT Press, Cambridge, MA, 1988.

[2] Burns, Steven M., *Synthesis and Analysis of Efficient Asynchronous Circuits*, Caltech PhD Thesis, 1990.

[3] Fishburn, J.P., Dunlop, A.E., "TILOS: A Posynomial Programming Approach to Transistor Sizing," *IEEE ICCAD*, pp. 326–328, Nov. 1985.

[4] Franklin, Joel, *Methods of Mathematical Economics*, Springer-Verlag, Berlin, 1980.

[5] Magott, Jan, "Performance Evaluation of Concurrent Systems Using Petri Nets," *Information Processing Letters*, 18, pp. 7–13, 1984.

[6] Martin, Alain J., "Programming in VLSI: From Communicating Processes to Delay-insensitive Circuits," in C.A.R. Hoare (ed), *UT Year of Programming Institute on Concurrent Programming*, Addison-Wesley, Reading, MA, 1989.

[7] Martin, A.J., Burns, S.M., Lee, T.K., Borkovic, D. and Hazewindus, P.J., "The Design of an Asynchronous Microprocessor," in C.L. Seitz (ed), *Advanced Research in VLSI: Proceedings of the Decennial Caltech Conference on VLSI*, pp. 351–373, MIT Press, Cambridge, MA, 1989.

[8] Obermeier, Fred W., *An Open Architecture for Improving VLSI Circuit Performance*, UC Berkeley PhD Thesis, 1989.

[9] Ousterhout, J.K., "A Switch-Level Timing Verifier for Digital MOS VLSI," *IEEE Transactions on CAD*, CAD–4(3), July 1985.

[10] Ramamoorthy, C.V., and Ho, Gary S., "Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets," *IEEE Transactions on Software Engineering*, SE–6(5), pp. 440–449, Sept. 1980.

[11] Shor, N.Z., *Minimization Methods for Non-Differentiable Functions*, translated from Russian, Springer-Verlag, Berlin, 1985.