Relative Timing

Kenneth S. Stevens, *Senior Member, IEEE*, Ran Ginosar, *Member, IEEE*, Shai Rotem, *Member, IEEE*

Abstract—
Relative Timing (RT) is introduced as a method for asynchronous design. Timing requirements of a circuit are made explicit using relative timing. Timing can be directly added, removed, and optimized using this style. RT synthesis and verification is demonstrated on three example circuits, facilitating transformations from speed-independent circuits to burst-mode and pulse-mode circuits. Relative timing enables improved performance, area, power, and functional testability of up to a factor of 3× in all three cases. This method is the foundation of optimized timed circuit designs used in an industrial test chip, and may be formalized and automated.

Keywords— **Asynchronous-design, Performance-tradeoffs, Dynamic-logic-circuit High-performance, Low-power-design**

I. Introduction

THE design of RAPPID, the asynchronous instruction length decoder, took more than two years to complete [1]. The primary goal was to investigate whether asynchronous design could improve performance in high-end microprocessors. This naturally led to the effort, reported in this paper, to study and develop circuits, CAD, and methodology most suitable for aggressive timed asynchronous circuit design.

Initial designs and methods were based on the CAD available at that time. The circuits were specified and synthesized using speed-independent (SI) or burst-mode (BM/XBM) methodologies [2][3][4], as well as metric timed circuit design [5]. We quickly discovered that many of the circuits that achieved our performance goals contained some form of timing assumptions - either the fundamental mode assumption of burst-mode or gate-level metric timing. The performance was improved by studying the natural delays of the circuits to employ timing that simplified the designs by reducing series transistors and logic levels.

Unfortunately, all the asynchronous methodologies at that time had what we considered an impediment to conceptualizing, optimizing, validating, and interfacing timed circuits. The timing assumptions were all *implicit*. We felt that in many cases the key performance was achieved through careful management and design of the *timing* of the circuits as much as the behavior. We therefore studied ways to make the timing of circuits explicit. This effort resulted in the *relative timing* (RT) style reported here.

Relative timing proved to be a very effective method of substituting aggressive pulse-mode, self-resetting circuits for the original full-handshake speed-independent designs in RAPPID. This novel method also allowed us to design and verify speculative asynchronous state machines. However, This effort required a new way of thinking about asynchronous designs and required a new set of tools.

In the absence of RT CAD tools, the manual flow is quite inefficient for the design of large systems. Now we face the question of how our manual method can be formalized into an effective CAD methodology and tools. We propose that new formal methodologies and tools be developed to support this method. This paper presents our methodology and lessons in order to motivate further CAD development. We start with simple, contrived examples that demonstrate basic principles, and move to a key RAPPID circuit which has been improved substantially with relative timing.

K. S. Stevens and S. Rotem are with Strategic CAD Labs, Intel Corporation, Hillsboro, OR.

R. Ginosar is with the VLSI Systems Research Center, Technion, Haifa, Israel

II. Motivation and description

The design of timing in digital circuits is an extremely difficult challenge. The conventional clocked digital design methodology solves this problem by decomposing the circuit into cycle-free combinational logic (CL) stages and interstage clocked latches; the clock cycle is simply tuned to accommodate the worst-case propagation delay in the CL stages. The behavior of the combinational logic can be specified and synthesized without considering timing. Delay Insensitive (DI) asynchronous circuits are analogous to clocked CL design in the sense that both types are independent of time – the behavior will be correct for arbitrary gate and wire delay.

High-performance circuits, both clocked and asynchronous, benefit from more aggressive timing methodologies. Clocked circuits can be considerably enhanced using local self-timing [6][7][8]. Timed asynchronous circuits can also have significantly enhanced performance.

Asynchronous design consists of handshake protocols that ensure validity of data [9][10]. Asynchronous design methodologies, apart from DI, make timing assumptions in the protocols, function logic, or data transmission [11]. If the assumptions are invalid in the physical implementation, the circuits can glitch and fail to operate correctly. SI circuits assume indistinguishable skew on wire forks, burst-mode assumes fundamental-mode (the circuit will stabilize internally before new inputs arrive), and bundled-data assumes that all data is stable before the handshake signal arrives. Ensuring that the timing assumptions hold in timed design, such as burst-mode, can be challenging [12].

The design style we investigated explicitly specifies the *effect* of delays in a circuit in terms of assertions on relative ordering of events (e.g. `a` goes high before `b` goes low). Our application of relative timing is based on the unbounded delay model commonly used by many asynchronous synthesis and verification tools. The circuits are then designed to meet the relative orderings, and validated that the constraints are part of the natural delays in the system.

A number of benefits emerged from making RT constraints explicit in our designs. Timing relationships are no longer hidden by a design style or tool. RT can unify the asynchronous methodologies as well as support for ad hoc manual designs. The bundled and burst-mode assumptions, for example, can usually be made explicit with a small number of RT constraints as

shown in Section IV-B.4. The explicit nature of the constraints can simplify interfacing, synthesis, and performance verification. RT is not restricted to any particular specification style and supports arbitrary designs. Since timing can directly effect the quality and robustness of the circuits, each assumption can be individually evaluated, and their application can be aggressive or conservative.

Many timing CAD tools and methodologies exist; asynchronous design itself is a timing methodology. Ordering signals temporally is not novel. This ordering can be achieved through graph transformations that reduce concurrency similar to the theory developed by Vanbekbergen [13]. Timed Petri nets, timed finite state machines, and other bounded-delay formalisms have been used to reason about timed circuits by [14][15][16][17] [18][19][20]. Component databooks include waveforms showing relative signal orderings, and orderings have been applied to micropipeline latches and controllers [21][22][23]. These methodologies can achieve extremely efficient circuits; indeed the tag unit in RAPPID, used as the primary example in this text, was first specified, synthesized, and validated using the metric tool ATACS [24].

However, we do feel that the RT methodology used in RAPPID applies timing top-down in a novel way that is intuitive and flexible, creating compact, testable, high-performance, low-power circuits in a style that can be automated by CAD. Further, this methodology supports both automatic and user-specified timing transformations. Initial RT solutions based on this work applied to synthesis [20] and verification [25] show remarkable results and potential for an automated RT design flow.

III. RAPPID RELATIVE TIMING DESIGN

Relative timing had significant impact on the RAPPID results. The timed asynchronous circuits, when compared to similar clocked logic in a commercial synchronous implementation, showed $3\times$ improvement in throughput, a $2\times$ improvement in latency, and half the energy per operation, at a 20% area penalty [1]. Although harder to quantify, we feel that relative timing was also key in achieving the 95% stuck-at testability in RAPPID with our functional BIST method through removing redundancies that naturally result through fixed signal orderings induced by timing.

Most of the RT circuits in RAPPID were designed by hand. The RT transformations modified many behavioral aspects of the specifications, concurrency in particular. However, the essential functionality of the controllers – synchronization and ordering – remained. This effort, while time consuming, helped us better understand timing, timed technology mapping, and what types of transformations appeared most beneficial. Various forms of handshaking were investigated, including protocols without direct acknowledgment. These pulse-based protocols can at times significantly improve the simplicity and latency of asynchronous circuits.

Most of our implementations were mapped onto standard static and domino library cells. Domino circuits are a restricted class of generalized C-Elements [26] where only a single term exists in the reset function. The combination of state-holding, low transition latency and the low activity factor of the domino gates made them the best circuit alternative we investigated.

The following sections describe the method we developed for designing and optimizing relative-timed circuits. What started out as a number of circuit experiments evolved into a manual flow. Automated tool support for these flows was painfully lacking, so we began mentoring development of RT CAD. Early engagement with the Petrify team led to automate synthesis using relative timing [20]. Verification using RT constraints was added to the verification tool Analyze [27] in-house. This tool was used to optimize the constraints in a slow, error-prone manual loop. Theory automating the verification and RT constraint optimization is under development [25]. We encourage researchers to further formalize and develop new CAD for automating RT design.

Signal	Description	Example
input signal	underline	$\underline{\texttt{input}}$
output signal		output
inverted (asserted low)	over-bar	$\overline{\texttt{z}}$
rising transition	up arrow	$\underline{\texttt{a}}\uparrow$
falling transition	down arrow	$\texttt{b}\downarrow$
timing arc	dashed arc	$- - \rightarrow$
behavioral arc	solid arc	$\longrightarrow$

TABLE I
NOTATION CONVENTIONS

IV. EXAMPLES

A. Notation and terminology

Table I shows some notations used in this paper. For CCS [28], '.' is the sequential operator, '+' is the nondeterministic choice operator, '|' is parallel composition, and '$\backslash\{\underline{\texttt{a}}\}$' is the restriction operator applied to signal $\underline{\texttt{a}}$ which disallows independent $\underline{\texttt{a}}$ and a transitions. Restricting signal a only permits the internal τ synchronization of the "handshake" between $\underline{\texttt{a}}$ and a.

All simulations have been made using synchronous standard library cells in a 0.18μ process. The output of each circuit drives a $0.18\mu \times 25\mu$ gate load. The circuits are simulated using SPICE and the values are normalized against one of the circuits in terms of area and energy. A more complete modeling of some of these circuits and parameters can be found in [29].

The circuit examples in this paper contain static and domino gates normally employing a single pMOS device. Asynchronous tools such as 3D [4][30], ATACS [5] and Petrify [2] can typically synthesize set-reset flops and the appropriate functions (Fig. 1(a)). We can often apply technology mapping into single-variable reset (equivalently set) functions, and implement them using standard footed domino gates as in Fig. 1(b). When the reset variable is not used in the set function, an unfooted domino gate is used instead (Fig. 1(c)).

B. C-Element

We use a simple C-Element example to demonstrate the concepts and methods of applying Relative Timing to synthesis and verification. A simple two-input generalized C-Element and its

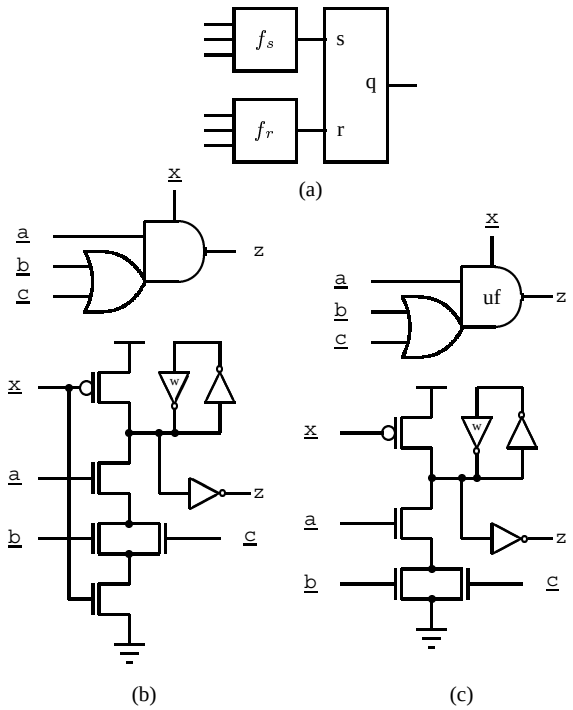


Fig. 1. (a) Set-Reset flop and functions. (b) Footed domino gate (symbol and circuit) implementing a Set-Reset flop with $f_r = \overline{\mathbf{x}}$, $f_s = \mathbf{x} \times \mathbf{a} \times (\mathbf{b} + \mathbf{c})$. (c) Unfooted domino gate implementing $f_r = \overline{\mathbf{x}}$, $f_s = \mathbf{a} \times (\mathbf{b} + \mathbf{c})$.

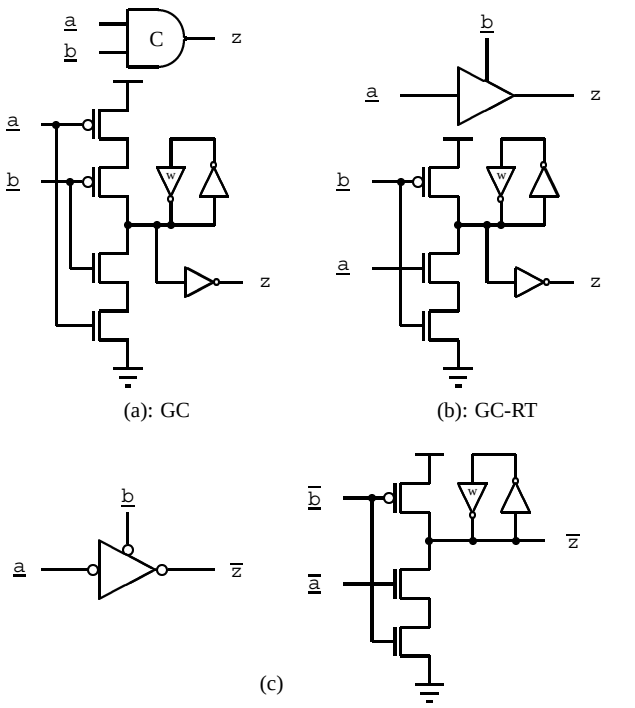


Fig. 2. Generalized C-Elements: (a) GC, (b) GC-RT for $\underline{\mathbf{a}}\downarrow \prec \underline{\mathbf{b}}\downarrow$ (c) for $\underline{\mathbf{a}}\uparrow \prec \underline{\mathbf{b}}\uparrow$

CMOS implementation are shown in Fig. 2(a). The formal definition in CCS is $\mathrm{C} = (\underline{\mathtt{a}} \mid \underline{\mathtt{b}}).\mathtt{z}.\mathrm{C}$, which reads "C is defined as single transitions showing on inputs $\underline{\mathtt{a}}$, $\underline{\mathtt{b}}$ in parallel (at any order), followed by a transition on the output $\mathtt{z}$, then followed recursively by C again" [28]. An equivalent STG representation of the specification is shown in Fig. 3(a) [31][32].

B.1 Relative Timing Synthesis

RT synthesis optimizes a circuit by adding timing arcs to a behavioral specification. Both timing and causality affect the behavior of a RT circuit. Behavioral arcs must be synthesized into gates, and timing relations enforce a specific ordering between concurrent events, resulting in concurrency reduction in the specification.

Relative timing assumptions come in two forms: local and global. Local timing constraints can automatically be generated by moving behavioral arcs based on various assumptions such as *lazy transition systems* [20]. Global assumptions are dictated by the response of the environment. These assumptions can be applied manually, as in Section V-C, or automatically, as in the burst-mode assumption that a circuit will stabilize before a new input burst arrives [4][30][33].

RT synthesis supports the creation and strengthening of timing assumptions by moving the relative positions of the heads and tails of arcs in a specification. If timing arcs are restricted to relative translations of behavioral arcs, aggressive timing optimizations can be performed on a circuit while ensuring a consistent, compatible result. The new specification can now be synthesized, and timing assumptions and requirements can be back-annotated. In this section we show some simple, intuitive transformations on C-elements. In Section V we show aggressive application of relative timing in a large circuit.

B.2 Synthesis Examples

Assume that the environment always produces transitions on $\underline{\mathtt{a}}$ before transitions on $\underline{\mathtt{b}}$. This relative timing assumption is expressed as a follows:

$$\text{RTA1:} \quad \underline{\mathtt{a}} \prec \underline{\mathtt{b}}$$

The C-Element can be reduced to a buffer $\mathrm{C} = \underline{\mathtt{b}}.\mathtt{z}.\mathrm{C}$ using this assumption. Fig. 3(b) shows the STG when the assumption is limited to the falling edges:

$$\text{RTA2:} \quad \underline{\mathtt{a}}\downarrow \prec \underline{\mathtt{b}}\downarrow$$

The dashed arc represents the timing assumption RTA2. Note that the timing arc supersedes the behavioral arc from $\underline{\mathtt{a}}\downarrow$ to $\mathtt{z}\downarrow$. Relative timing effectively moves the tail of this arc from one event ($\mathtt{z}\downarrow$) to a predecessor of the event ($\underline{\mathtt{b}}\downarrow$), as indicated by the double arrow in Fig. 3(b). The new timing arc makes the behavioral arc redundant, as shown in Fig. 3(c). In the corresponding circuit, the reset function contains only $\underline{\mathtt{b}}\downarrow$, and the C-Element can be implemented as the Fig. 2(b) footed domino gate GC-RT: $\mathrm{C} = (\underline{\mathtt{a}}\uparrow \mid \underline{\mathtt{b}}\uparrow).\mathtt{z}\uparrow.\underline{\mathtt{a}}\downarrow.\underline{\mathtt{b}}\downarrow.\mathtt{z}\downarrow.\mathrm{C}$. Given a similar assumption on the positive edges,

$$\text{RTA3:} \quad \underline{\mathtt{a}}\uparrow \prec \underline{\mathtt{b}}\uparrow$$

the circuit can be mapped to the domino gate in Fig. 2(c) by inverting the inputs and employing the non-buffered $\overline{\mathtt{z}}$ output.

Static C-element implementations can be synthesized with Petrify. The STG of Fig. 3(a) produces the circuit SIC shown in Fig. 4(a). Timing assumptions RTA2 and RTA3 lead to the

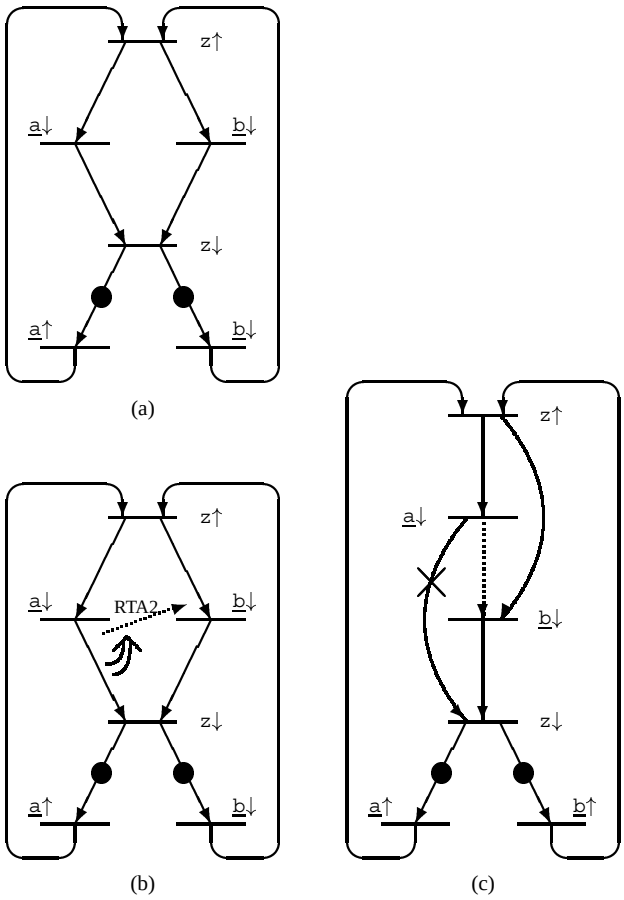


Fig. 3. Relative timing transformations on the petri-net of a C-element: (a) initial spec (b) relative timing arc RTA2 $\underline{a}\downarrow \prec \underline{b}\downarrow$ effectively "translates" arc $(\underline{\mathbf{a}}\downarrow, \mathbf{z}\downarrow)$ to $(\underline{\mathbf{a}}\downarrow, \underline{\mathbf{b}}\downarrow)$ (c) new spec, $\times$'ed arc is redundant.

simpler static circuits of Fig. 4(b) and 4(c) respectively. Note that these two circuits are actually subcircuits of the speed-independent one. 3D synthesizes the circuit of Fig. 4(d).

In general, applying relative timing for synthesis means that new (timing) arcs are inserted, rendering other arcs redundant. This could also be considered as moving either the head, tail, or both ends of behavioral arcs to predecessors. This effectively reduces concurrency in the specification, allowing a simpler implementation by removing transistors and gates.

B.3 Relative timing verification

This section introduces the method developed to verify a large, relative-timed asynchronous circuit called RAPPID [1]. An implementation I *conforms* to a specification S ($I \succeq_c S$) when an implementation is an acceptable construction of the specification [34][16][27]. In this section, implementations can be assumed to be parallel compositions of the untimed behavioral specifications of the gates. Relative timing predicates can be added to implementations and specifications to reduce their concurrency by pruning states in a state graph (SG) that are unreachable due to timing. Thus, a specification S conforms to an implementation I with RT predicate R when $I \wedge R \succeq_c S$.

Early in this effort the Analyze verifier was enhanced to support RT predicates on both implementations and specifications. Circuits can then be verified using SI and DI unbounded delays

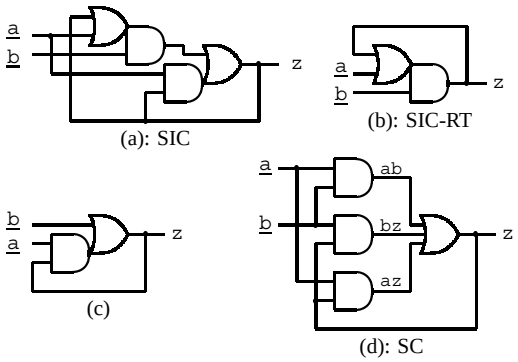


Fig. 4. Static C-Elements: (a) Speed-Independent (b) with RT assumption $\underline{\mathbf{a}}\downarrow \prec \underline{\mathbf{b}}\downarrow$ (c) with RT assumption $\underline{\mathbf{a}}\uparrow \prec \underline{\mathbf{b}}\uparrow$ (d) Burst-mode C-Element with hazards

with RT constraints.

RT verification has two aspects. First, RT constraints reduce concurrency in the implementation by disallowing transitions to failure states. Second, the set of RT constraints are optimized and merged through a set of transformations.

The following algorithm was applied to generate RT constraints and verify RAPPID and the circuit examples in this paper. Step 1 generates RT constraints that remove a single failure state as will be shown in the following example. This capability was added to our verifier. Step 2 optimizes the constraint by reducing additional concurrency beyond the single failure state. Step 3 adds the new optimized RT constraint to the set, removing any constraints covered by the new constraint. Steps 2 and 3 were done manually.

1. Verify conformance using current RT predicates.
   - If failure free, report RT constraints.
   - If failure cannot be fixed through timing, quit.
   - If failure exists, create RT constraint(s) that remove this failure.
2. Optimize new constraint(s):
   - Remove concurrency by increasing coverage of the SG by the RT constraint.
   - Iterate optimization, terminating when:
   – Further concurrency reduction would remove states required by the specification.
   – Slack in constraint is no longer positive.
   – An arc edge touches a primary input or output.
3. Add optimized constraint to RT constraint set, remove covered constraints, and iterate.

The following section illustrates the procedure used in RAPPID to determine how and when to increase coverage of a RT constraint.

B.4 Verification Example

Consider the static C-Element (SC) in Fig. 4(d). This circuit is implicitly hazard-free under burst-mode or fundamental mode assumptions. However, it is not hazard-free in a speed-independent environment. If the environment responds quickly, $\underline{\mathtt{b}}\downarrow$ may immediately follow $\mathtt{z}\uparrow$ before node $\underline{\mathtt{az}}$ rises, resulting in a hazard.

Fig. 5 shows a state graph of the SC C-Element circuit. The "bottom" symbols in the left and right corner of the diamonds label error states. Transitions $\mathtt{ab}\downarrow_1$ and $\mathtt{bz}\downarrow_1$ lead to the error

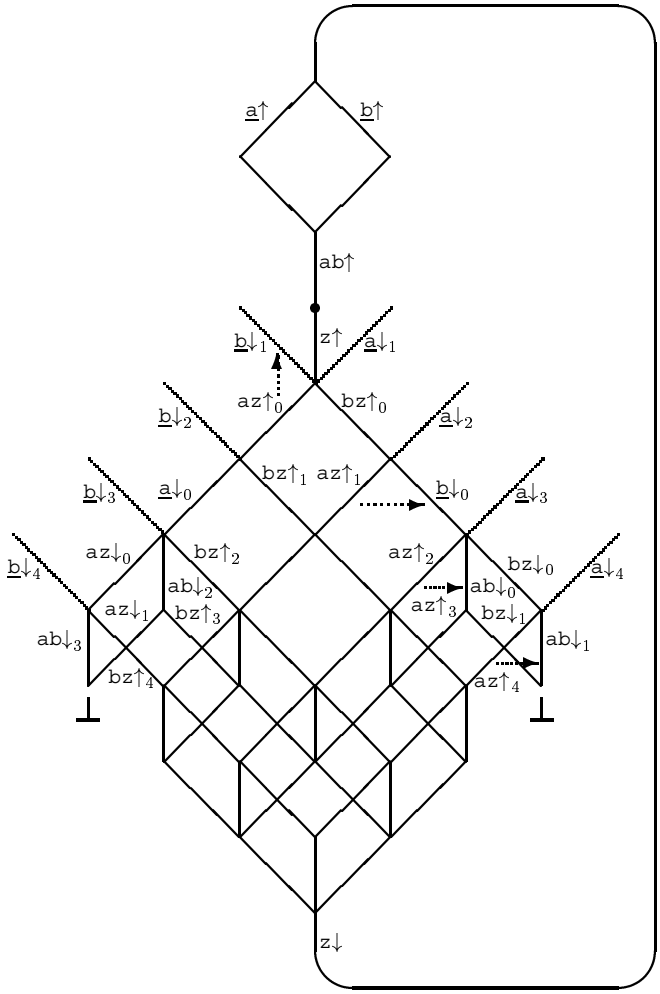


Fig. 5. Relative Timed burst-mode SC State Graph

state on the right.

Using the method described in Section IV-B.3, we first try to eliminate the right error. Verification will identify any arcs that lead to error states. The following two constraints eliminate the right error state:

$$\text{RTC4:}\quad \mathtt{az}\uparrow_4 \prec \mathtt{ab}\downarrow_1$$
$$\text{RTC5:}\quad \mathtt{az}\uparrow_3 \prec \mathtt{bz}\downarrow_1$$

If one signal must precede another and both exit from a single state, then the later arc will never be taken. (e.g. $\mathtt{ab}\downarrow_1$ in RTC4). RTC4 and RTC5 therefore disallow entrance to the right error state.

One representation of the timed precedence of RTC4 is the dashed arc between $\mathtt{az}\uparrow_4$ and $\mathtt{ab}\downarrow_1$ in the SG of Fig. 5. We now try to strengthen this constraint to cover more of the graph. While there may be many methods to optimize the instance-based constraints, our hand methodology used two main iterations.

First, instance information is removed from the constraints when possible. The generalized constraint $\mathtt{az}\uparrow \prec \mathtt{ab}\downarrow_1$ is equivalent to RTC4 as it adds no new timing arcs to the SG. Generalizing the right side as well results in the constraint $\mathtt{az}\uparrow \prec \mathtt{ab}\downarrow$ effectively adding a second timing arc $\mathtt{az}\uparrow_3 \prec \mathtt{ab}\downarrow_0$ to the SG. This constraint now removes two states from the graph: the error state and the state of RTC5. Hence RTC5 is covered by the optimized RTC4 constraint.

Second, the generalized RT constraint can be strengthened based on slack calculations[1]. The constraint is strengthened by moving the left and/or right transition to earlier transitions in the SG. Hence the right side of the constraint can be strengthened to transition $\underline{\mathtt{b}}\downarrow$ and $\mathtt{bz}\uparrow$. A simple unit delay model can be used to calculate slack, where local gates are assigned a single delay, and input transitions are assigned a value k where $k \geq 1$. The following example illustrates the strengthening of $\mathtt{az}\uparrow \prec \mathtt{ab}\downarrow$ starting from the common signal $\mathtt{z}\uparrow$.

$$\begin{array}{lcll} \mathtt{z}\uparrow\mathtt{az}\uparrow & \prec & \mathtt{z}\uparrow\{\mathtt{bz}\uparrow\underline{\mathtt{b}}\downarrow\}\mathtt{ab}\downarrow & k \\ \mathtt{z}\uparrow\mathtt{az}\uparrow & \prec & \mathtt{z}\uparrow\underline{\mathtt{b}}\downarrow & k-1 \\ \mathtt{z}\uparrow\mathtt{az}\uparrow & \prec & \mathtt{z}\uparrow\mathtt{bz}\uparrow & 0 \end{array}$$

This indicates that if $k > 1$ the best strengthening is RTC6, otherwise the weaker $\mathtt{az}\uparrow \prec \mathtt{ab}\downarrow$ should be used. Applying the same method to the left error state generates RTC7.

$$\text{RTC6:}\quad \mathtt{az}\uparrow \prec \underline{\mathtt{b}}\downarrow$$
$$\text{RTC7:}\quad \mathtt{bz}\uparrow \prec \underline{\mathtt{a}}\downarrow$$

The RT implementation now conforms to the specification. Precisely, SC $\wedge$ RTC6 $\wedge$ RTC7 $\succeq_c$ C $= (\underline{\mathtt{a}} \mid \underline{\mathtt{b}}).\mathtt{z}.$C. All signals in these constraints are either primary inputs or directly enabled by the primary output, simplifying hierarchical validation.

In general, RT verification allows one to manipulate the initial constraints to arrive at a minimal set of constraints that are easiest to verify in a hierarchical system. Constraints that have over aggressively reduced the slack can be weakened back to the original failure state. If any initial constraint contains unachievable timing, then the circuit is an invalid implementation of the specification.

RTC6 and RTC7 implement a "weak" form of the fundamental-mode requirement of burst-mode. Because Analyze uses bisimulation semantics, hazardous behavior inside a circuit that does not propagate to the outputs is permissible due to the *observational equivalence* property [28][27]. (This is not the case when using verifiers based on weaker formalisms.) RTC6 and RTC7 prune arcs $\underline{\mathtt{a}}\downarrow_{1,3,4}$ and $\underline{\mathtt{b}}\downarrow_{1,3,4}$ but transitions $\underline{\mathtt{a}}\downarrow_2$ and $\underline{\mathtt{b}}\downarrow_2$ of Fig. 5 remain. Given RTC6 and RTC7, if $\underline{\mathtt{a}}\downarrow_2$ occurs, $\mathtt{az}\uparrow$ will either glitch or not fire. This does not create an observable failure because signals bz and ab are asserted holding z high, and the output will not lower until $\underline{\mathtt{b}}\downarrow$ and $\mathtt{bz}\downarrow$ occur. Hence the additional "strong" burst-mode RT constraints $\mathtt{az}\uparrow \prec \underline{\mathtt{a}}\downarrow$ and $\mathtt{bz}\uparrow \prec \underline{\mathtt{b}}\downarrow$ are unnecessary.

B.5 C-Element summary

Table II summarizes the five alternative designs from Fig. 2 and 4. The circuits are all sized near the optimal power/performance point. All designs were simulated to drive the same load. If a circuit is hazard-free in an SI environment then no timing is required for correct operation. The speed-independent circuit (SIC) is slower than all others. Applying the RTA2 assumption to this design leads to a circuit (SIC-RT)

[1] Slack is the difference between the latest arrival of the first signal and the earliest arrival of the second.

Circuit	HF in SI Env.	Fall Delay	Rise Delay	Switching Energy	Area # Transistors	Exhaustive Testability	RTA2 Env. Testability
SIC	Yes	1.00	0.98	1.00	16	100%	90%
SIC-RT	No	0.59	0.52	0.74	8	n/a	100%
SC	No	0.56	0.51	2.03	18	100%	92%
GC	Yes	0.77	0.55	0.86	10	100%	100%
GC-RT	No	0.52	0.52	0.72	9	n/a	100%

TABLE II

COMPARISON OF C-ELEMENT IMPLEMENTATIONS. FALL, RISE, AND ENERGY COLUMNS USE WORST SIC PERFORMANCE AS BASE, WITH ALL OTHER NUMBERS A MULTIPLE OF THAT DELAY OR ENERGY. ENERGY IS AVERAGE FOR A COMPLETE CYCLE (RISE AND FALL). TEST COLUMNS SHOW COSMOS STUCK-AT FAULT COVERAGE ON ALL FANOUTS, WITH REDUCED PATTERNS IN RTA2 COLUMN DUE TO ENVIRONMENT RESTRICTIONS.

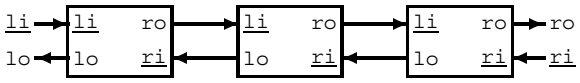


Fig. 6. FIFO block diagram containing three cells

that is half the size and enhances performance by almost a factor of two. The static SC requires the largest circuit and is fast, but doubles the power. The reduced domino C-Element (GC-RT) improves fall times over the GC circuit by 50% (due to simplification of the pull-up stack) and rise times by 5%.

Static circuits tend to expend more energy than domino circuits. This is largely due to the extra switching activity in the static designs as can be observed by the SC circuit which expends twice the power of the SIC circuit because all four gates toggle for every output transition. When activity factors are similar, the domino circuits have a slight edge. The GC-RT circuit uses only 3% less energy than the static SIC-RT circuit because the reduced device sizes in the domino gates are offset by the short circuit current when the gates switch. Testability was measured in COSMOS using a functional test methodology, where only valid timed signal orderings allowed by the environment can be supplied to the circuit. The table shows that the static and SI circuits are fully testable for complete patterns, but not when timing constraints reduce signal interleavings (in column RTA2). The RT optimized versions of these circuits are fully testable.

V. TIMING EVOLUTION IN A RING

In this section we trace the development of a simple FIFO controller, similar to a micropipeline [35]. These controllers can be connected in series as shown in Fig. 6. This circuit is a simplified abstraction of a part of the RAPPID design [1], and closely follows the actual steps used to derive the final circuit. We begin with a speed-independent design, and review a succession of progressively simpler circuits, enabled through careful application of relative timing assumptions.

A. Speed-independent FIFO cell

A simple FIFO cell can be specified in CCS as follows.

$$\begin{aligned} \text{LEFT} &= \underline{\texttt{li}}\uparrow.\underline{\texttt{c}}.\texttt{lo}\uparrow.\underline{\texttt{li}}\downarrow.\texttt{lo}\downarrow.\text{LEFT} \\ \text{RIGHT} &= \underline{\texttt{c}}.\texttt{ro}\uparrow.\underline{\texttt{ri}}\uparrow.\texttt{ro}\downarrow.\underline{\texttt{ri}}\downarrow.\text{RIGHT} \\ \text{FIFO} &= (\text{LEFT} \mid \text{RIGHT})\backslash\{\underline{\texttt{c}}\} \end{aligned} \quad (1)$$

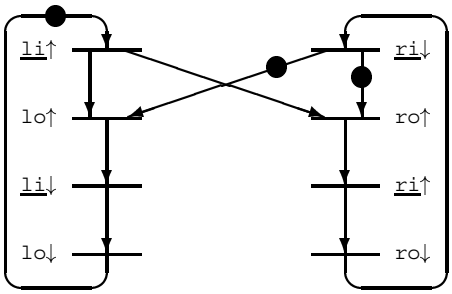


Fig. 7. FIFO specification Petri-net

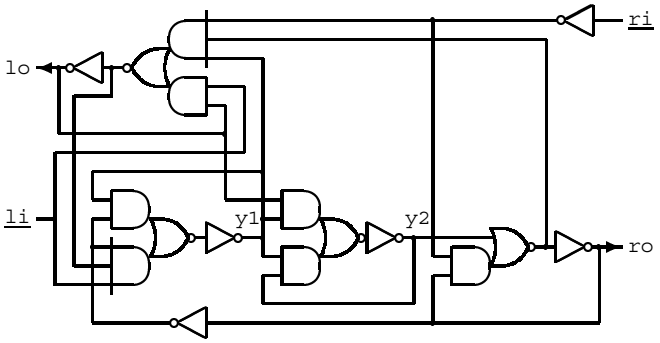


Fig. 8. Speed-independent FIFO cell (SI)

The specification in Equation 1 consists of two handshake processes, LEFT and RIGHT. The c event synchronizes the two processes so that ri must go low and li must rise before both processes may proceed. This process-based specification is equivalent to the Petri-net of Fig. 7.

The SI circuit in Fig. 8 was synthesized using Petrify [2] and is a hazard-free implementation of Equation 1.

B. Burst-mode FIFO cell

The SI FIFO pays a considerable delay penalty to achieve speed independence. The trace li↑, y1↑, lo↑, y2↑, ro↑ shows that li↑ produces lo↑ after two complex gate and inverter delays, and ro↑ after four. Perhaps the performance can be improved if the circuit can ensure that concurrent outputs are generated faster than they can be acknowledged by the environment. This assumption can be formulated as follows:

$$\begin{aligned} \text{RTA8:} &\quad \texttt{lo}\uparrow \prec \underline{\texttt{ri}}\uparrow \\ \text{RTA9:} &\quad \texttt{ro}\uparrow \prec \underline{\texttt{li}}\downarrow \end{aligned}$$

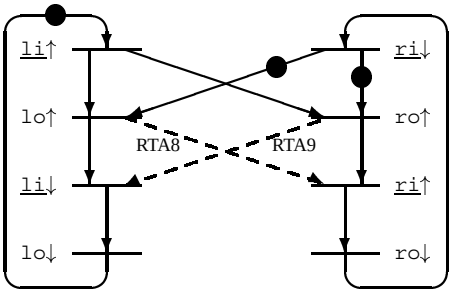


Fig. 9. FIFO specification petri-net with RT assumptions RTA8 and RTA9 represented as dashed arcs

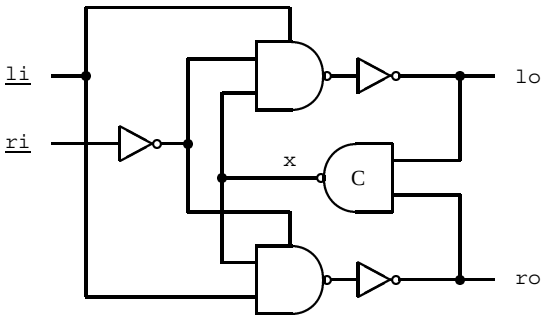


Fig. 10. Relative timed burst-mode FIFO (RT-BM)

A new specification is generated by adding these two relative timing assumptions to Equation 1:

$$\text{FIFO} \wedge \text{RTA8} \wedge \text{RTA9} \tag{2}$$

where FIFO is the specification from Equation 1. This is equivalent to the Petri-net of Fig. 9 where the dashed arrows represent relative timing constraints.

Note that the two constraints RTA8 and RTA9 are in a form where outputs precede inputs and these outputs are concurrently enabled from the same pair of inputs. This is a burst-mode constraint where the input burst is $\{\underline{\texttt{li}}\uparrow\ \underline{\texttt{ri}}\downarrow\}$ and the output burst is $\{\texttt{lo}\uparrow\ \texttt{ro}\uparrow\}$. This burst-mode timing assumes that the variance in the generation of the concurrent outputs is always less than the response time of the environment[2].

The RT-BM circuit of Fig. 10 is derived in [20] using the new RT synthesis capabilities of Petrify, and implements Equation 2. (The C-Element here is synthesized as an OR gate in [20].) RT verification by Analyze extracts the timing in the physical circuit and creates additional orderings that must hold for the circuit to operate correctly:

$$\text{RTC10:}\quad \texttt{x}\downarrow \prec \texttt{lo}\downarrow$$
$$\text{RTC11:}\quad \texttt{x}\downarrow \prec \texttt{ro}\downarrow$$

These constraints, as well as the state variable x, are shown graphically in Fig. 11. The burst-mode implementation achieves a 2.6× average speedup over the SI circuit. Constraints RTC10–RTC11 apply only to the physical implementation and must be validated given actual circuit delays.

[2]Also applying burst-mode constraints on input set $\{\underline{\texttt{li}}\downarrow\ \underline{\texttt{ri}}\uparrow\}$ results in a C-Element – the micropipelines implementation.

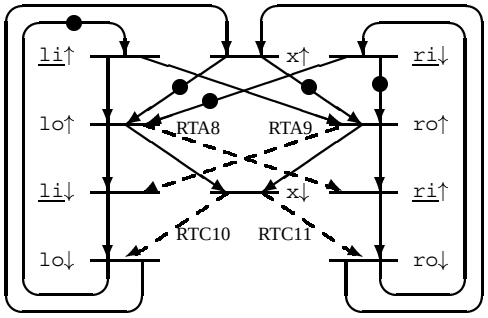


Fig. 11. Petri-net for circuit of Fig. 10 and constraints RTC10–RTC11.

C. Right before left

Assume that we connect the circuit of Specification (2) into a ring with a single token. The token will always arrive at an idle cell due to circuit delays if the ring is sufficiently large. Hence the handshake in process RIGHT will always complete before a new handshake in process LEFT. The SI or RT-BM circuits can safely be used in a large ring. However, the global timing of RTA12 can improve the circuit in terms of power, performance, area and testability.

$$\text{RTA12:}\quad \underline{\texttt{ri}}\downarrow \prec \underline{\texttt{li}}\uparrow$$

This assumption can be graphically represented as shown in Fig. 12. The arcs from $\underline{\texttt{ri}}\downarrow$ to $\texttt{ro}\uparrow$ and $\texttt{lo}\uparrow$ are now redundant and have been removed from the figure.

The dashed arcs are not *causal* arcs; $\underline{\texttt{ri}}$ must go low before $\underline{\texttt{li}}$ can rise and $\underline{\texttt{ri}}$ cannot delay $\underline{\texttt{li}}$. This represents a major change in the operation of the circuit; the LEFT process is no longer synchronized directly with the RIGHT process except through system timing. The design must guarantee that the token appears on the dashed arc before $\underline{\texttt{li}}\uparrow$.

The circuit in Fig. 13 can be synthesized with 3D and Petrify from Specification (2) adding assumption RTA12. The rising edge of signal $\underline{\texttt{li}}$ must be delayed sufficiently through lo and the buffer to ensure that the domino AND gate is not disabled

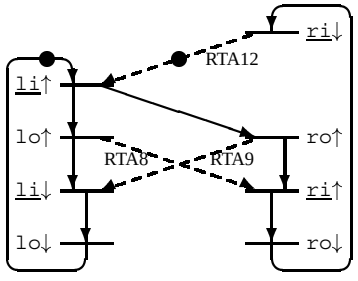


Fig. 12. Net representing addition of RT assumptions $\underline{\textbf{ri}}\downarrow \prec \underline{\textbf{li}}\uparrow$

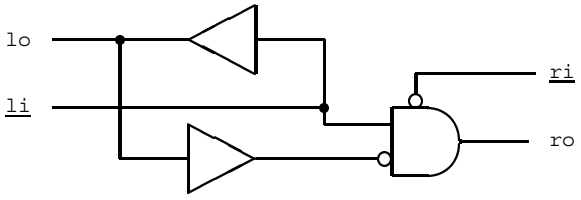


Fig. 13. Aggressive relative timed FIFO (RT-Agr)

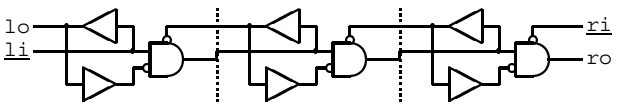


Fig. 14. Aggressive relative timed FIFOs

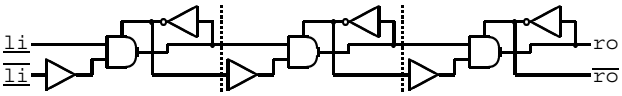


Fig. 15. Shuffled aggressive relative timed FIFO cells

before it is fully set. This results in a number of RT constraints on races in the circuit that can be derived as was done for RTC4–RTC7 in the SC circuit. This circuit shows 15% and 3× improvement in average case performance over the RT-BM and SI circuits respectively, and energy is also improved by factors of 26% and 1.9×.

D. Pulse-mode FIFO cell

RTA12 now constrains the specification sufficiently to derive a pulse-mode circuit. Through transitivity, ro↓ must precede li↑. We can use this weaker constraint to discard ri, the backward handshake signal, altogether. We show how this can be accomplished through transformations on the circuit of Fig. 13.

Three elements of the ring are shown in Fig. 14. Observe that the lo signal is nothing more than a delayed version of the li signal. Shuffling the lo devices and bubbles results in the circuit of Fig. 15, that has only forward-moving signals without any inter-cellular feedback. The shuffling that removes acknowledgment is directly based on RTA12 that dissociates the LEFT process from the RIGHT. This shuffling turns output lo and input ri into local signals.

Note that signal $\overline{\texttt{li}}$ in Fig. 15 is just li inverted. A transition li↑ creates a short period when both li and $\overline{\texttt{li}}$ are high, which will set the output of the domino AND gate. The duration of both inputs to the domino AND gate being high depends on the delay in the $\overline{\texttt{li}}$ path. This signal pair can be combined into a single wire li if the signal on this wire operates as a pulse. The final circuit derivation can be seen in Fig. 16.

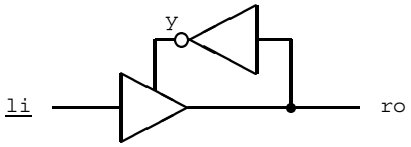


Fig. 16. Relative timed pulse-mode FIFO (Pulse)

The following specification removes the direct handshake signals lo and ri of Specification (1) and adds RTA12:

$$\begin{array}{lcl} \text{LEFTP} & = & \texttt{li}\uparrow.\texttt{c}.\texttt{li}\downarrow.\text{LEFTP} \\ \text{RIGHTP} & = & \texttt{c}.\texttt{ro}\uparrow.\texttt{ro}\downarrow.\text{RIGHTP} \\ \text{PULSE} & = & (\text{LEFTP} \mid \text{RIGHTP})\backslash\{\texttt{c}\} \\ & & \wedge\ \texttt{ro}\downarrow \prec \texttt{li}\uparrow \end{array} \tag{3}$$

Designing reliable pulse-mode circuits is very difficult [36]. We can observe some of the constraints of pulse circuits by understanding how we have derived the pulse-mode circuit in this example. Fig. 17 shows a four-phase request-acknowledge handshake. Constraints 1 through 4 are causal with speed-independent signaling. By removing the ack signal (lo and ri in Fig. 14), we are left with only the request signal that requires constraints 2p and 4p. These constraints contain both minimum and maximum metric bounds. However, the actual requirements for the size of these bounds can be represented with relative timing arcs between the inputs and outputs of a pulse-mode circuit (li and ro in Fig. 16). Interestingly, these arcs correspond to a protocol very similar to the standard request acknowledge handshaking.

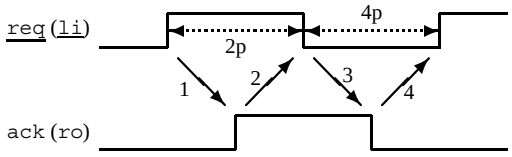


Fig. 17. Four cycle and pulse handshake protocol constraints

The pulse on li of Fig. 16 causes the output pulse ro, as required by Equation 3. If we map req to li and ack to ro in Fig. 17, we see that arc 1 is causal. However, this circuit can fail if the pulse is so short that the ro (ack) pulse does not occur. We can therefore impose an RT constraint that requires ro↑ (ack↑) before li↓ (req↓). This makes arc 2 in Fig. 17 an RT constraint, and slightly restricts the specification. (It may be possible to not restrict the specification if an internal signal toggles which ensures the domino gate has changed state.) The circuit will also fail if the li (req) pulse is too long. If ro↓ (ack↓) and y↑ have occurred before li↓ (req↓) then an additional pulse on ro might be generated. Therefore, arc 3 in Fig. 17 is a necessary RT constraint for the circuit to work. Finally, arc 4 is assumed to hold from RTA12 which drove this example. We therefore have a system of causal and relative timing relations that must hold in the pulse-mode circuit which directly mimic a four-phase handshake.

E. Ring summary

Some consequences of evolving a simple FIFO-like controller from a speed-independent to a pulse-mode circuit are summarized in Table III. The different circuits are characterized in terms of performance, power, area, and testability. The worst case latency of the SI circuit is from three to five times longer than the circuits that use timing. The SI circuit is not fully testable, and the testability degrades as the circuit is placed in an environment where concurrency is restricted. The more aggressive timing assumptions tend to increase the performance of the circuits, reduce the area and power, and increase functional testability. Note that the bulk of the improvement in performance has been achieved with the simple burst-mode transformation; simple timing assumptions can often have significant impact on the quality of the circuit. The additional savings awarded by going to pulse mode are much less pronounced, except that the variation is eliminated. Indeed, the 'aggressive' RT controller may already be considered a pulse mode circuit. Power is improved for each transformation, as the pulse circuit shows a 40% reduction over RT-Agr. We feel that functional testability is increased using relative timing because many of

Circuit	HF in SI Env.	Worst Delay	Average Delay	Switching Energy	Area # Trans.	SI Env. Testability	RTA14 Env. Testability	Pulse-Mode Testability
SI	Yes	1.00	0.67	1.00	42	88%	79%	n/a
RT-BM	No	0.34	0.26	0.81	32	80%	77%	n/a
RT-Agr	No	0.34	0.22	0.53	18	n/a	100%	n/a
Pulse	No	0.22	0.22	0.32	15	n/a	n/a	100%

TABLE III

COMPARISON OF FIFO IMPLEMENTATIONS. ALL DELAYS ARE IN TERMS OF SI CIRCUIT WORST-CASE DELAY, ENERGY IN TERMS OF SI CIRCUIT. ENERGY ACCOUNTS FOR A COMPLETE FOUR-PHASE CYCLE. SYNCHRONOUS TESTING IN COSMOS REQUIRED EXTRA TEST GATE FOR PULSE CIRCUIT.

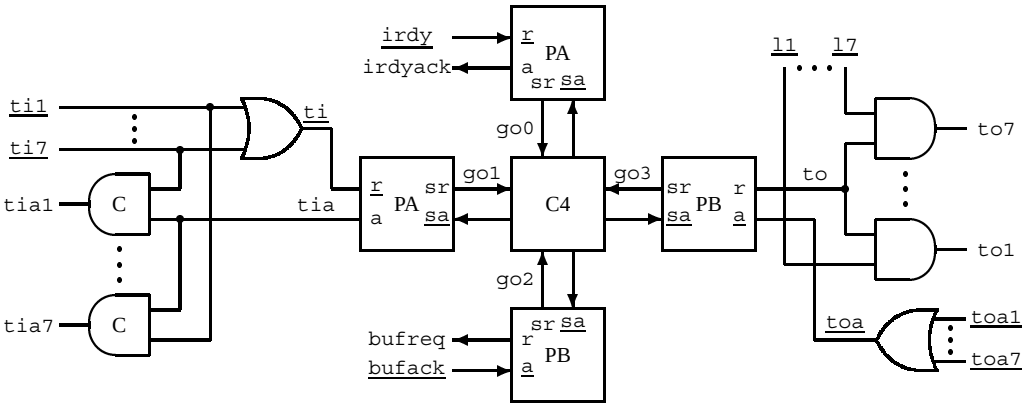


Fig. 18. SI Tag Unit. Assumes tagin (ti) handshakes are mutex.

the redundant coverings are removed when the circuits are optimized for time.

VI. TAG UNIT EXAMPLE

The FIFO ring is a simplified example used for illustration. Typically, such an application would have synchronizations coming from multiple paths. The Tag Unit example from RAPPID [1] shows how relative timing can be employed to generate extremely high-performance pulse-mode implementations.

Decoding of variable length instructions is inherently a serial process, since the length of any instruction directly depends on the lengths of all previous instructions since the last branch. The performance of decoding variable length instructions directly depends on how fast this serial process operates [1]. A critical component in RAPPID is the Tag Unit. The tagging control signals interconnect the Tag Units to form a 4×16 torus, synchronizing the serial ordering of instructions by passing a tag along the toroidal rings.

Fig. 18 shows a single speed-independent Tag Unit. An input tag arrives on at most one of the inputs $\underline{\texttt{ti1}} - \underline{\texttt{ti7}}$. The tag is synchronized with $\underline{\texttt{irdy}}$ and steered to one of $\texttt{to1} - \texttt{to7}$ depending on instruction length $\underline{\texttt{l1}} - \underline{\texttt{l7}}$. A `bufreq`, `irdyack`, and the corresponding `tia` are also issued concurrently. The four-input C4 allows four processes to complete their four-cycle handshake concurrently and begin a new transfer when all interfaces are synchronized. The three behaviors in the boxes are specified as follows:

$$\begin{aligned}
\text{PA} &= \underline{\texttt{r}}\uparrow.\texttt{sr}\uparrow.\underline{\texttt{sa}}\uparrow.(\texttt{sr}\downarrow.\underline{\texttt{sa}}\downarrow \mid \texttt{a}\uparrow.\underline{\texttt{r}}\downarrow).\texttt{a}\downarrow.\text{PA}\\
\text{PB} &= \texttt{sr}\uparrow.\underline{\texttt{sa}}\uparrow.(\texttt{sr}\downarrow.\underline{\texttt{sa}}\downarrow \mid \texttt{r}\uparrow.\underline{\texttt{a}}\uparrow).\texttt{r}\downarrow.\underline{\texttt{a}}\downarrow.\text{PB}\\
\text{C4} &= (\underline{\texttt{go0}} \mid \underline{\texttt{go1}} \mid \underline{\texttt{go2}} \mid \underline{\texttt{go3}}).\texttt{sa}.\text{C4}
\end{aligned}$$

The two passive PA processes synchronize the four-phase handshake after $\underline{\texttt{r}}$ requests are received, while the two PB processes are active and synchronize before handshaking. Therefore, when the $\underline{\texttt{ti}}$ and $\underline{\texttt{irdy}}$ requests arrive and the `bufreq` and `to` cycles have completed, the $\underline{\texttt{ti}}$ and $\underline{\texttt{irdy}}$ signals will be acknowledged and the `to` and `bufreq` cycles will start. This is accomplished in the specification by renaming the signals and composing the processes as follows:

$$\begin{aligned}
\text{IRDY} &= \text{PA}[\underline{\texttt{irdy}}/\underline{\texttt{r}},\ \underline{\texttt{irdyack}}/\underline{\texttt{a}},\ \underline{\texttt{go0}}/\underline{\texttt{sr}}]\\
\text{TAGIN} &= \text{PA}[\underline{\texttt{ti}}/\underline{\texttt{r}},\ \underline{\texttt{tia}}/\underline{\texttt{a}},\ \underline{\texttt{go1}}/\underline{\texttt{sr}}]\\
\text{TAGOUT} &= \text{PB}[\underline{\texttt{to}}/\underline{\texttt{r}},\ \underline{\texttt{toa}}/\underline{\texttt{a}},\ \underline{\texttt{go3}}/\underline{\texttt{sr}}]\\
\text{BUFREQ} &= \text{PB}[\underline{\texttt{bufreq}}/\underline{\texttt{r}},\ \underline{\texttt{bufack}}/\underline{\texttt{a}},\ \underline{\texttt{go2}}/\underline{\texttt{sr}}]\\
\text{TAGUNIT} &= (\text{IRDY} \mid \text{TAGIN} \mid \text{TAGOUT} \mid \text{BUFREQ}\\
&\quad \mid \text{C4})\backslash\{\underline{\texttt{go0}},\ \underline{\texttt{go1}},\ \underline{\texttt{go2}},\ \underline{\texttt{go3}},\ \underline{\texttt{sa}}\}
\end{aligned} \tag{4}$$

The SI implementation of these processes using ATACS is shown in Fig. 19. Processes PA and PB result in very efficient implementations. However, the large OR gates, C-Elements, and the necessity of passing through three state machines from the input to output of the tag path create significant latency in this implementation.

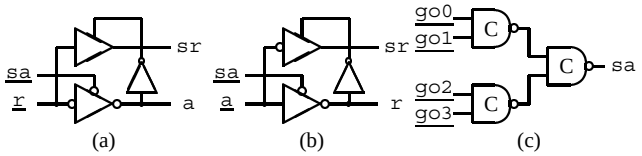


Fig. 19. Speed-independent Tag Unit circuits: (a) PA (b) PB (c) C4

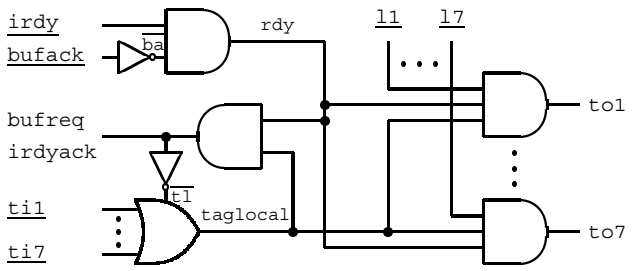


Fig. 20. Simplified RAPPID Tag Unit

The circuit used in RAPPID is shown in Fig. 20. This efficient circuit is very similar to the pulse FIFO (Fig. 16) derived in Section V. The extra gates are used to steer the tag paths (ti to to) based on the instruction length, and to synchronize with the instruction issue buffers. The backward handshake signals in the tag path have been removed, and the forward-going signals are pulses. The request and acknowledge protocols on the irdy and bufreq paths are combinations of four-phase and pulse-mode signaling – irdyack and bufreq being pulses.

$$
\begin{array}{lll}
\text{2p} & \text{RTA13:} & \{\texttt{bufreq}\uparrow, \texttt{irdyack}\uparrow\} \prec \texttt{to}\downarrow \\
\text{2p} & \text{RTA14:} & \{\texttt{to}\uparrow, \texttt{irdyack}\uparrow\} \prec \texttt{bufreq}\downarrow \\
\text{2p} & \text{RTA15:} & \{\texttt{to}\uparrow, \texttt{bufreq}\uparrow\} \prec \texttt{irdyack}\downarrow \\
3 & \text{RTA16:} & \underline{\texttt{ti}}\downarrow \prec \texttt{to}\downarrow \\
3 & \text{RTA17:} & \underline{\texttt{ti}}\downarrow \prec \texttt{bufreq}\downarrow \\
3 & \text{RTA18:} & \underline{\texttt{ti}}\downarrow \prec \texttt{irdyack}\downarrow \\
4 & \text{RTA19:} & \{\texttt{bufreq}, \underline{\texttt{bufack}}\uparrow, \texttt{irdyack}, \underline{\texttt{irdy}}\downarrow\} \\
 & & \quad \prec \underline{\texttt{ti}}\uparrow \\
4 & \text{RTA20:} & \{\texttt{to}, \texttt{bufreq}, \underline{\texttt{bufack}}\uparrow, \overline{\texttt{ba}}\downarrow\} \prec \underline{\texttt{irdy}}\uparrow \\
 & \text{RTA21:} & \texttt{irdyack}\downarrow \prec \underline{\texttt{irdy}}\downarrow \\
 & \text{RTA22:} & \texttt{bufreq}\downarrow \prec \underline{\texttt{bufack}}\downarrow
\end{array}
$$

$$
\begin{array}{lcl}
\text{TAGS} & = & \underline{\texttt{b1}}.\underline{\texttt{ti}}\uparrow.\texttt{c1}.(\underline{\texttt{ti}}\downarrow \mid \underline{\texttt{c2}}.\texttt{to}\uparrow.\texttt{to}\downarrow).\text{TAGS} \\
\text{BUF} & = & \underline{\texttt{c1}}.\underline{\texttt{c2}}.\texttt{bufreq} \\
 & & \quad .(\texttt{bufreq} \mid \underline{\texttt{bufack}}.\underline{\texttt{bufack}}).\text{BUF} \\
\text{IRDY} & = & \underline{\texttt{irdy}}.(\underline{\texttt{b2}}.\texttt{c2}.\texttt{irdyack} \\
 & & \quad .(\texttt{irdyack}. \mid \underline{\texttt{irdy}}).\text{IRDY} \\
 & & \quad + \underline{\texttt{nott}}.\underline{\texttt{irdy}}.\underline{\texttt{nott}}.\text{IRDY}) \\
\text{MUTEX} & = & (\texttt{b1}.\texttt{b2} + \texttt{nott}.\texttt{nott}).\text{MUTEX} \\
\text{TAG} & = & (\text{TAGS} \mid \text{BUF} \mid \text{IRDY} \mid \text{MUTEX}) \\
 & & \backslash \{\underline{\texttt{c1}}, \underline{\texttt{c2}}, \underline{\texttt{b1}}, \underline{\texttt{b2}}, \underline{\texttt{nott}}\} \\
 & & \wedge \text{RTA13} - \text{RTA22}
\end{array}
\tag{5}
$$

The specification for the RAPPID tag circuitry is shown in Equation 5. The processes are behavioral pulse-based specifications without timing. For example, the lowering edge of the pulse signal $\underline{\texttt{ti}}\downarrow$ and the output pulse to in process TAGS are concurrent. The timing assumptions necessary to create a failure-free circuit can be classified by type according to Fig. 17. Type 4 assumptions on the ti and to signals are encoded into the specification since the TAGIN and TAGOUT processes have been combined. The synchronizations c1 and c2 encode causal transitions of type 1. RTA13–RTA15 encode type 2p transitions – minimum pulse-widths constraints on to, bufreq, and irdyack. Assumptions RTA16–RTA18 are type 3, ensuring that the input pulse lowers before the output pulse. RTA19 and RTA20 are type 4 assumptions which require the logic to stabilize before the next tagin arrives. Assumptions RTA21 and RTA22 simply constrain the ordering of the pulsed handshake signals. (Such constraints could have easily been placed in the specification, but have been included as RT assumptions because they are guaranteed by timing rather than by a causal relation.)

Equation 6 shows the complete set of RT constraints placed on the circuit and system for the simplified RAPPID implementation to be valid. These constraints were generated and verified through Analyze [27]. RTC23 and RTC24 are the type 2 constraints, RTC25–RTC27 are type 3 (the same as RTA16–RTA18 in the specification), RTC28–RTC31 the type 4 constraints, and type 4p RTC32–RTC33 constraints. Note that a single delay path constraint may include several RT constraints as we have used them here.

$$
\begin{array}{lll}
2 & \text{RTC23:} & \texttt{to}\uparrow \prec \texttt{taglocal}\downarrow \\
2 & \text{RTC24:} & \{\texttt{irdyack}\uparrow, \texttt{to}\uparrow, \overline{\texttt{tl}}\downarrow\} \prec \texttt{rdy}\downarrow \\
3 & \text{RTC25:} & \underline{\texttt{ti}}\downarrow \prec \texttt{to}\downarrow \\
3 & \text{RTC26:} & \underline{\texttt{ti}}\downarrow \prec \texttt{br}\downarrow \\
3 & \text{RTC27:} & \underline{\texttt{ti}}\downarrow \prec \texttt{irdyack}\downarrow \\
4 & \text{RTC28:} & \texttt{rdy}\downarrow \prec \texttt{taglocal}\uparrow \\
4 & \text{RTC29:} & \texttt{rdy}\downarrow \prec \overline{\texttt{ba}}\uparrow \\
4 & \text{RTC30:} & \{\texttt{taglocal}\downarrow, \overline{\texttt{tl}}\uparrow\} \prec \underline{\texttt{ti}}\uparrow \\
4 & \text{RTC31:} & \texttt{taglocal}\downarrow \prec \texttt{rdy}\uparrow \\
4\text{p} & \text{RTC32:} & \{\underline{\texttt{ba}}\uparrow, \overline{\texttt{ba}}\downarrow\} \prec \underline{\texttt{irdy}}\uparrow \\
4\text{p} & \text{RTC33:} & \texttt{taglocal}\downarrow \prec \overline{\texttt{tl}}\uparrow
\end{array}
\tag{6}
$$

While the circuit of Fig. 20 may be easier to verify using the metric timing of ATACS, we feel that explicitly attaching many, if not all, of the timing constraints as RT predicates make the specification and circuit timing requirements more perspicuous. Each interface has a simple behavioral definition, which is refined by timing assumptions as predicates. Incorporating the assumptions into the specification removes much of the clarity of the resulting synchronizations and orderings. Representing the complete behavior constraints or timing constraints as a Petri-net, as was shown in Section V, can be elucidating for understanding small examples, but can be confusing and impractical for larger, real-world examples such as the Tag Unit in RAPPID. This is particularly the case for pulse-based implementations where the set of timing constraints can be quite large.

Table IV compares the two implementations. The RT circuit has a 3.1× area, 1.9× power, and 2.5× improvement in latency and throughput over the speed-independent circuit. Since this circuitry is in the critical path of the RAPPID length decoder, the improvements in this example directly resulted in improvements to RAPPID [1]. The area impact on RAPPID from the RT circuit is arguably much higher than the transistor count comparison since this circuit is wire-limited and can be scaled. If slow parts are used, higher scaling factors must be employed to meet the target performance. If the slower SI tag unit had been used in RAPPID, the area would have ballooned significantly to meet the performance goals. The area savings in terms of the 50% reduction in wire count from removing the backward handshake is also significant. Since RAPPID tagging uses point-to-point signaling connected in a torus, removing the backward acknowledgment path resulted in a savings of 14 wires per tag unit. This reduced the network bisection of the tag logic by a total of 224 tag wires.

Circuit	Tag Latency	Cycle Time	Cycle Energy	Area # Trans.	RAPPID Testability
SI	0.53	1.00	1.00	297	n/a
RAPPID	0.21	0.39	0.54	97	98.6%

TABLE IV

COMPARISON OF RAPPID TAG UNIT WITH THE SI VERSION. CYCLE TIME OF SI CIRCUIT IS BASE CASE FOR DELAYS. AREA IS THE NUMBER OF TRANSISTORS, TESTABILITY REFERS TO THE COMPLETE RAPPID TAG UNIT AND STEERING LOGIC.

VII. CONCLUSION

The development of circuits requires correct operation in two domains - behavioral and temporal. Our experiments indicate that the design, synthesis, and verification of circuits can be significantly enhanced if both temporal and behavioral domains can be explicitly represented and merged. Relative timing is a means of combining behavioral and temporal information. The statespace of the untimed circuit is reduced by removing unreachable relative signal orderings that are induced through time constraints.

Relative timing is a useful way of reasoning about designs. The waveforms in databooks are presented in such a way as to highlight the relation between signals and transitions. One can use relative timing to architect systems, as well as synthesize controllers and verify the correctness of systems. Synthesis and verification algorithms can be designed to directly support this concept where time is represented as a relationship similar to a behavioral or causal relation.

RT can be applied as aggressively or conservatively as desired. Races due to the environment in burst-mode and in speed-independent implementations due to inverter delays can be discovered and explicitly listed with the circuit. Indeed, relative timing is a superset of asynchronous methodologies such as DI, SI, and burst-mode.

Relative timing does not preclude metric or absolute timing. Metric timing must eventually be applied in the implementation against the RT constraints to prove that they hold. Further, many of the RT constraints require a certain amount of slack, or setup and hold times, in the precedence relations. The robustness and reliability of the circuits can depend directly on the amount of slack on the RT constraints.

The quality of the RAPPID results in terms of throughput, power, area, testability, and latency was largely due to the timing employed in the circuits [1]. This benefit is shown through applying relative timing to the examples in this text, and in the early tools that have formalized some of these translations.

Acknowledgments

We are grateful for the helpful and constructive comments from the referees. Henrik Hulgaard and Steve Burns participated in timing verifications. Jordi Cortadella and Mike Kishinevsky were the first to introduce automatic RT into the CAD tool Petrify. Peter Beerel and Hoshik Kim have been key contributors to RT verification and optimization.

REFERENCES

[1] Ken Stevens, Shai Rotem, Ran Ginosar, Peter Beerel, Chris Myers, Kenneth Yun, Rakefet Kol, Charles Dike, and Marly Roncken, "An Asynchronous Instruction Length Decoder," *IEEE Journal of Solid State Circuits*, vol. 36, no. 2, pp. 217–228, Feb. 2001.

[2] Jordi Cortadella, Michael Kishinevsky, Alex Kondratyev, Luciano Lavagno, and Alex Yakovlev, "Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers," *IEICE Transactions on Information and Systems*, vol. E80-D, no. 3, pp. 315–325, 1997.

[3] Steven M. Nowick, *Automatic Synthesis of Burst-Mode Asynchronous Controllers*, Ph.D. thesis, Stanford University, Department of Computer Science, 1993.

[4] Kenneth Yi Yun, *Synthesis of Asynchronous Controllers for Heterogeneous Systems*, Ph.D. thesis, Stanford University, Aug. 1994.

[5] Chris J. Myers, *Computer-Aided Synthesis and Verification of Gate-Level Timed Circuits*, Ph.D. thesis, Dept. of Elec. Eng., Stanford University, October 1995.

[6] Kevin J. Nowka and Tibi Galambos, "Circuit Design Techniques for a Gigahertz Integer Microprocessor," in *1998 IEEE International Conference on Computer Design: VLSI in Computers & Processors (ICCD98)*. IEEE Computer Society, October 1998, pp. 11–16.

[7] David Sager, Glen Hinton, Michael Upton, Terry Chappell, Thomas D. Fletcher, Samie Samaan, and Robert Murray, "A 0.18μm CMOS IA32 Microprocessor with a 4GHz Integer Execution Unit," in *International Solid State Circuits Conference*, Feb. 2001, pp. 324–325, 461.

[8] Stanley Schuster, William Reohr, Peter Cook, David Heidel, Michael Immediato, and Keith Jenkins, "Asynchronous interlocked pipelined CMOS circuits operating at 3.3–4.5GHz," in *International Solid State Circuits Conference*, 2000, pp. 292–293.

[9] Charles L. Seitz, "System timing," in *Introduction to VLSI Systems*, Carver A. Mead and Lynn A. Conway, Eds., chapter 7. Addison Wesley, 1980.

[10] David E. Muller and W. S. Bartky, "A theory of asynchronous circuits," in *Proceedings of an International Symposium on the Theory of Switching*. Apr. 1959, pp. 204–243, Harvard University Press.

[11] Scott Hauck, "Asynchronous design methodologies: An overview," *Proceedings of the IEEE*, vol. 83, no. 1, pp. 69–93, Jan. 1995.

[12] Supratik Chakraborty, *Polynomial-Time Techniques for Approximate Timing Analysis of Asynchronous Systems*, Ph.D. thesis, Stanford University, Aug. 1998.

[13] Peter Vanbekbergen, Gert Goossens, Francky Catthoor, and Hugo J. De Man, "Optimized synthesis of asynchronous control circuits from graph-theoretic specifications," *IEEE Transactions on Computer-Aided Design*, vol. 11, no. 11, pp. 1426–1438, Nov. 1992.

[14] C. J. Myers, T. G. Rokicki, and T. H.-Y. Meng, "POSET timing and its application to the synthesis and verification of gate-level timed circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, vol. 18, no. 6, pp. 769–786, June 1999.

[15] Wendy Belluomini and Chris J. Myers, "Timed circuit verification using TEL structures," *IEEE Transactions on Computer-Aided Design*, vol. 20, no. 1, January 2001.

[16] Rajeev Alur and David L. Dill, "A Theory of Timed Automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[17] Henrik Hulgaard, *Timing Analysis and Verification of Timed Asynchronous Circuits*, Ph.D. thesis, Department of Computer Science, University of Washington, 1995.

[18] Radu Negulescu and Ad Peeters, "Verification of speed-dependences in single-rail handshake circuits," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, 1998, pp. 159–170.

[19] Supratik Chakraborty, Kenneth Y. Yun, and David L. Dill, "Practical timing analysis of asynchronous systems using time separation of events," in *Proc. IEEE Custom Integrated Circuits Conference*, May 1998.

[20] J. Cortadella, M. Kishinevsky, S. M. Burns, A. Kondratyev, L. Lavagno, K. S. Stevens, A. Taubin, and A. Yakovlev, "Lazy Transition Systems and Asynchronous Circuit Synthesis with Relative Timing Assumptions,"

IEEE Transactions on Computer-Aided Design, vol. 21, no. 2, pp. 109–130, Feb 2002.

[21] Paul Day and J. Viv Woods, "Investigation into micropipeline latch design styles," *IEEE Transactions on VLSI Systems*, vol. 3, no. 2, pp. 264–272, June 1995.

[22] Stephen B. Furber and Paul Day, "Four-phase micropipeline latch control circuits," *IEEE Transactions on VLSI Systems*, vol. 4, no. 2, pp. 247–253, June 1996.

[23] Sam S. Appleton, Shannon V. Morton, and Michael J. Liebelt, "Two-phase asynchronous pipeline control," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*. Apr. 1997, pp. 12–21, IEEE Computer Society Press.

[24] C. Myers, "Timed circuits: A new paradigm for high-speed design," in *Proc. of Asia and South Pacific Design Automation Conference*, Feb. 2001.

[25] Hoshik Kim, Peter A. Beerel, and Kenneth S. Stevens, "Relative timing based verification of timed circuits and systems," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, Apr. 2002.

[26] Alain J. Martin, "Programming in VLSI: From communicating processes to delay-insensitive circuits," in *Developments in Concurrency and Communication*, C. A. R. Hoare, Ed. 1990, UT Year of Programming Series, pp. 1–64, Addison-Wesley.

[27] Kenneth S. Stevens, *Practical Verification and Synthesis of Low Latency Asynchronous Systems*, Ph.D. thesis, University of Calgary, Calgary, Alberta, September 1994.

[28] Robin Milner, *Communication and Concurrency*, Computer Science. Prentice Hall International, London, 1989.

[29] Maitham Shams, Jo C. Ebergen, and Mohamed I. Elmasry, "Modeling and Comparing CMOS Implementations of the C-Element," *IEEE Transactions on VLSI Systems*, vol. 6, no. 4, pp. 563–567, December 1998.

[30] Steven M. Nowick and David L. Dill, "Exact two-level minimization of hazard-free logic with multiple-input changes," *IEEE Transactions on Computer-Aided Design*, vol. 14, no. 8, pp. 986–997, Aug. 1995.

[31] Tam-Anh Chu, *Synthesis of Self-Timed VLSI Circuits From Graph-Theoretic Specifications*, Ph.D. thesis, Massachusetts Institute of Technology, September 1987.

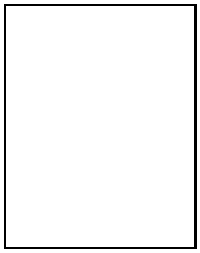
[32] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proceedings of the IEEE*, pp. 541–580, Apr. 1989.

[33] W. S. Coates, A. L. Davis, and K. S. Stevens, "Automatic Synthesis of Fast Compact Self-Timed Control Circuits," in *IFIP Working Conference on Design Methodologies*, April 1993, pp. 193–208.

[34] David L. Dill, *Theory for Automatic Hierarchical Verification of Speed-Independent Circuits*, ACM Distinguished Dissertations. MIT Press, 1989.

[35] Ivan E. Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, no. 6, pp. 720–738, June 1989, Turing Award Lecture.

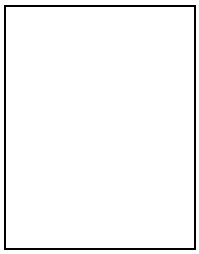
[36] Vinod Narayanan, Barbara A. Chappell, and Bruce M. Fleischer, "Static Timing Analysis For Self Resetting Circuits," in *International Conference on Computer-Aided Design (ICCAD-96)*. IEEE Computer Society, November 1996, pp. 119–126.



Kenneth S. Stevens received a B.A. degree in Biology in 1982 and the B.S. and M.S. degrees in Computer Science in 1982 and 1984 from the University of Utah. He received a Ph.D. in Computer Science from the University of Calgary, Alberta Canada, in 1994.

From 1984 through 1991 he held research positions at the Fairchild/Schlumberger Laboratory for AI Research, the Schlumberger Palo Alto Research Laboratory, and Hewlett Packard Laboratories, in Palo Alto, CA. Dr. Stevens became an Assistant Professor at the Air Force Institute of Technology in Dayton, OH in 1994, and since 1996 he has been an Adjunct Professor. Since 1996 he has been employed at Intel Corporation's Strategic CAD Labs in Hillsboro OR.

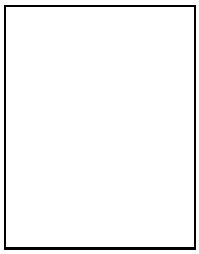
His primary expertise includes asynchronous circuits, VLSI, architecture, hardware synthesis and verification, and timing analysis. He holds seven patents and has been the principal author for three papers which received the best paper award and has served on technical program committees for conferences and workshops. He is a Senior Member of the IEEE.



Ran Ginosar received his B.Sc. in Electrical Engineering and Computer Engineering Summa cum Laude from the Technion in 1978, and his Ph.D. in Electrical Engineering and Computer Science from Princeton University in 1982.

After working with AT&T Bell Laboratories for one year, he joined the Technion faculty in 1983. He was a visiting Associate Professor with the University of Utah in 1989-1990 and a visiting faculty with the Strategic CAD Lab at Intel in 1997-1999. Ran Ginosar serves as the head of the VLSI Systems Research Center at the Technion.

His research interests include asynchronous systems and electronic imaging.



Shai Rotem was born in Haifa, Israel in 1954. He holds a B.Sc. degree from the Technion - Israel Institute of Technology, Haifa, Israel, from 1980.

Shai has been with Intel since 1980, in positions of VLSI design and architecture of data communication controllers and microprocessors, and CAD design and research in formal verification and asynchronous design. He is currently a Principal Engineer in the Mobile Processor Group's architecture team, responsible for platform architecture definition.

Shai is a member of the IEEE Computer society.