Relative Timing

Ken Stevens[1], Ran Ginosar[1,2], Shai Rotem[1]
[1]Strategic CAD Labs, Intel Corporation, Hillsboro, OR
[2]VLSI Systems Research Center, Technion, Haifa, Israel

Abstract

Relative Timing is introduced as an informal method for aggressive asynchronous design. It is demonstrated on three example circuits (C-Element, FIFO, and RAPPID Tag Unit), facilitating transformations from speed-independent circuits to burst-mode, relative timed, and pulse-mode circuits. Relative timing enables improved performance, area, power and testability in all three cases.

1. Introduction

The design of RAPPID, the asynchronous instruction length decoder, took more than two years to complete [13]. Beyond investigating whether asynchronous design could improve performance, we also wanted to find out which design styles and circuit families are most suitable for aggressive circuit design.

We started with Speed Independent (SI) and Extended Burst Mode (XBM) specifications. However, existing synthesis tools [5, 17] yielded results that were less than satisfactory for critical paths. Next, we turned to timed design and employed a metric timing synthesis tool [9]. The resulting circuits demonstrated improved performance but were still below our expectations. Therefore, we turned to aggressive manual design for the critical paths and managed to obtain the results reported in [13]. Now we face the question of how our method of semi-manual design can be turned into an effective CAD methodology and tools.

In retrospect, one approach stands out as the most successful method in that process. We employed *Relative Timing* (RT) assumptions to specify and argue about our circuits, applied certain transformations that preserved relative timing, and validated that the relative timing assumptions held in the final circuits. This approach turned out to be a very effective method to semi-formalize the substitution of aggressive pulse-mode, self-resetting circuits for the original full-handshake speed-independent ones.

We propose that a new formal methodology and tools be developed to support this method. In the absence of such CAD tools, the method is quite inefficient for the design of large systems. This paper presents our lessons in order to motivate such an effort. We start with simple, contrived examples that demonstrate basic principles, and move to a RAPPID circuit which has been improved substantially with relative timing.

2. 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 [12]. Timed asynchronous circuits can have significantly enhanced performance, but require better understanding and modeling of circuit performance and delay variation.

Metric timing requires the specification of propagation times or ranges thereof [16, 9]. Unfortunately metric timing analysis can explode in complexity to the extent that the synthesis and verification of even moderately sized timed circuits can become intractable [1]. Metric timing typically needs complete characterization of all device and environment delays to achieve improvements over unbounded delay models. Complete characterization of environment delays as well as estimation of the latencies of the circuits to be synthesized seem awkward.

An alternative to metric timing allows the designer or CAD algorithms to specify 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 already used by most asynchronous synthesis and verification tools. SI or XBM specifications are easily restricted based on designer specified assumptions of relative signal orderings of the environment. The circuits are then designed to meet the relative orderings, or verified that the restrictions are already part of the delays in the system.

Many timing CAD tools and methodologies exist; asynchronous design itself is a timing methodology. Ordering signals temporally is not novel. Metric and non-metric timed automata has been considered by [1, 9, 6, 11, 2, 4]. Component databooks include waveforms showing relative signal orderings. However, we do feel that the RT methodology used in RAPPID applies timing top-down in a novel way that is intuitive, flexible, creates high performance small low power testable circuits, and is easily supported by CAD.

3. RAPPID relative timing design

Once the RAPPID architecture was complete the challenge of circuit mapping began. Initial specifications were synthesized using full-handshake circuits. We began studying the environment of many of the critical circuits to see if timing could be employed to reduce the number of logic levels in each controller. The system architecture created environmental signal relations where the fastest arrival delays are large compared to the local controllers (as in the ring example in Sections 4.3 and 4.4). Signal orderings were also enforced by design. The latency of many circuits in RAPPID was reduced by a factor of as much as 3× through such timing transformations. These transformations modified many behavioral aspects of the specifications, concurrency in particularly. However, the essential functionality of the controllers – synchronization and ordering – remained.

Most of the RT circuits in RAPPID were designed by hand. This effort, while time consuming, helped us better understand timing, timed technology mapping, and what types of transformations appeared most beneficial. We investigated various forms of handshaking, including protocols without direct handshaking. These pulse-based protocols can at times significantly improve the simplicity and latency of asynchronous circuits.

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

A key aspect to the correct operation of the silicon was the verification of these timed circuits. The timing verification tool Analyze [15] was enhanced to support relative timing verification. The verifier was also used to generate a complete set of RT constraints from the critical races in a circuit. These constraints enforce a particular resolution of the races that guarantee correct operation of a circuit. This is shown in Section 4.2.1, where the hidden timing assumptions of burst-mode are explicitly derived. Timing assumptions in this paper are labeled RTA, whereas critical races that are discovered through verification and must be ordered for the circuit to operate correctly are labeled RTC.

Some of the hand designed RT circuits were checked for validity through ATACS. However, the environmental and local path delays in the RT assumptions were typically validated with SPICE simulations.

We feel that relative timing had significant impact on the throughput (3× improvement), latency (2× improvement), and area (15% bloat) over similar logic in a commercial synchronous implementation. Although harder to quantify, we feel that relative timing was key in achieving the 95% stuck-at testability in RAPPID through removing redundancies that naturally result through fixed signal orderings induced by timing.

The lack of synthesis support was a serious productivity limitation once our methodology was in place. Part of this aspect has been successfully addressed in joint research with the Petrify team by creating integrated algorithms that support RAPPID-style automatic RT synthesis. Many of the key RT controllers, including the one presented in Section 4.4, can now be directly synthesized in Petrify.

A significant weakness in RAPPID validation was timing analysis support in the back-end. Henrik Hulgaard verified the timing of the RAPPID FIFO. A relative timing flow that automatically generates all essential RT constraints, calculates the best and worst case paths necessary for the constraint to hold, and completes the timing analysis for these paths is research yet to be completed.

We encourage researchers to further develop CAD for RT design.

4. Examples

4.1. Notation and terminology

Table 1 shows some notations used in this paper. For CCS [7], '.' is the sequential operator, '$+$' is the nondeterministic choice operator, '$|$' is parallel composition, and '$\backslash\{\underline{a}\}$' is the restriction operator applied to signal $\underline{a}$.

All simulations have been made using standard library cell device sizes driving six standard inverters as a load. They were simulated in SPICE using the MOSIS 0.5μ process parameters. A more complete modeling of some of these circuits and parameters can be found in [14].

The circuit examples in this paper are all based on nonclocked domino gates employing a single pMOS device.

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

Table 1. Notation conventions

Asynchronous tools such as 3D [17], ATACS [8] and Petrify [5] can typically synthesize set-reset flops and the appropriate functions (Figure 1(a)). We apply technology mapping into single-variable reset (equivalently set) functions, and implement them using standard footed domino gates as in Figure 1(b). When the reset variable is not used in the set function, an unfooted domino gate is used instead (Figure 1(c)).

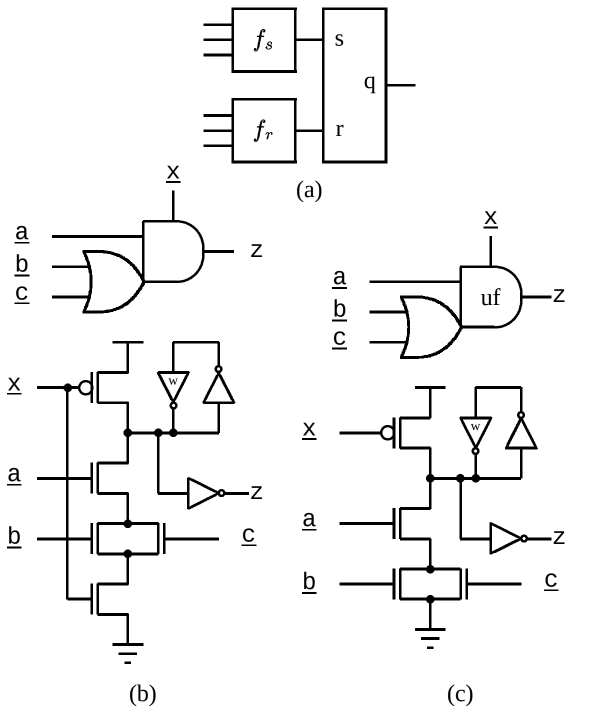


Figure 1. (a) Set-Reset flop and functions. (b) Footed domino gate (symbol and circuit) implementing a Set-Reset flop with $f_r = \overline{\underline{\mathbf{x}}}$, $f_s = \underline{\mathbf{x}} \times \underline{\mathbf{a}} \times (\underline{\mathbf{b}} + \underline{\mathbf{c}})$. (c) Unfooted domino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

4.2. C-Element

A simple two-input generalized C-Element $\mathrm{C} = (\underline{\mathsf{a}} \mid \underline{\mathsf{b}}).\mathsf{z}.\mathrm{C}$ (as defined in CCS [7]) and its CMOS implementation are shown in Figure 2(a). Let's assume that we know that the environment always produces transitions on $\underline{\mathsf{a}}$ before transitions on $\underline{\mathsf{b}}$, and we feel this knowledge might simplify our circuit. This relative timing assumption is expressed as a follows:

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

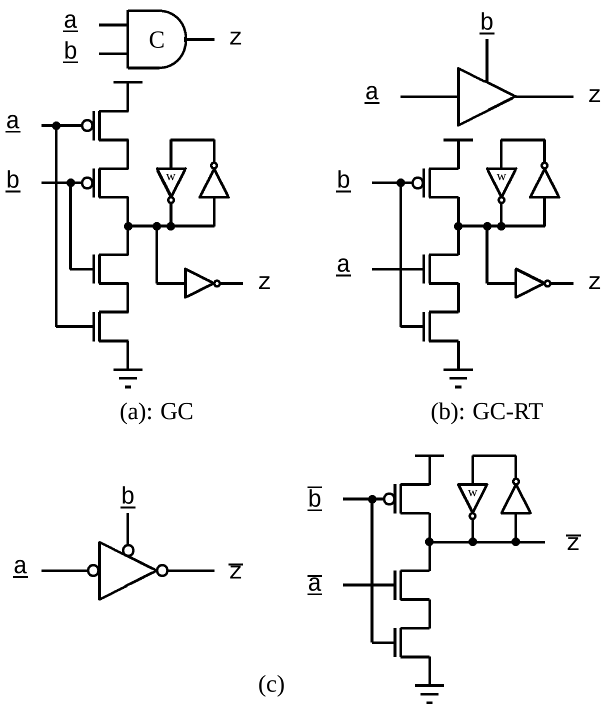
The C-Element is reduced to a buffer: $\mathrm{C} = \underline{\mathsf{b}}.\mathsf{z}.\mathrm{C}$ using this assumption. If the assumption is limited to the falling edges,

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

the reset function contains only $\underline{\mathsf{b}}\downarrow$, and the C-Element can be implemented as a footed domino gate (Figure 2(b)): $\mathrm{C} = (\underline{\mathsf{a}}\uparrow \mid \underline{\mathsf{b}}\uparrow).\mathsf{z}\uparrow.\underline{\mathsf{a}}\downarrow.\underline{\mathsf{b}}\downarrow.\mathsf{z}\downarrow.\mathrm{C}$. With a similar assumption on the positive edges,

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

the circuit can be mapped to the domino gate in Figure 2(c) by inverting the inputs and employing the non-buffered $\overline{\mathsf{z}}$ output. Alternatively, the output can be buffered for high loads. A "wobbly" C-Element $\mathrm{C} = \underline{\mathsf{a}}.\underline{\mathsf{b}}.\mathsf{z}.\mathrm{C} + \underline{\mathsf{b}}.(\underline{\mathsf{a}}.\mathsf{z}.\mathrm{C} + \underline{\mathsf{b}}.\mathrm{C})$, that is unsafe because input $\underline{\mathsf{b}}$ may toggle and withdraw, can also be verified and synthesized as above.



a
b
C
z
w
(a): GC
(b): GC-RT
$\overline{\mathsf{z}}$
$\overline{\underline{\mathsf{b}}}$
$\overline{\underline{\mathsf{a}}}$
(c)

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

Let's consider the static C-Element (SC) in Figure 3(a). This circuit is not speed-independent, but is safe provided the environment is sufficiently slow. Alternatively, Petrify [5] synthesizes the static complex gate circuit shown in Figure 3(c). Timing assumptions RTA2 or RTA3 lead to the simpler static circuits of Figure 3(d) and 3(e), respectively. Note that these two circuits are actually subcircuits of the speed-independent one.

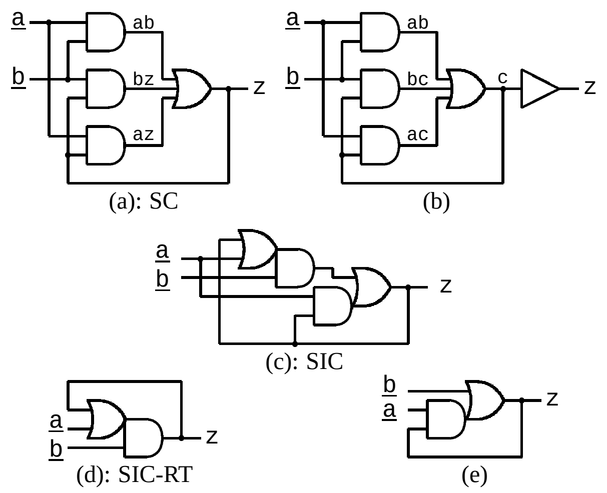


Figure 3. Static C-Elements: (a) C-Element with hazards, (b) locally timed, (c) Speed-Independent, (d) with RT assumption $\underline{a}\downarrow \prec \underline{b}\downarrow$, (e) with RT assumption $\underline{a}\uparrow \prec \underline{b}\uparrow$

4.2.1. Relative timing verification

The SC circuit in Figure 3(a) is implicitly hazard-free under the fundamental mode assumption. Relative timing allows this assumption to be made explicitly. If, for instance, the environment responds quickly, $\underline{\texttt{b}}\downarrow$ may immediately follow $\texttt{z}\uparrow$, before node $\underline{\texttt{az}}$ rises. This race shows up as a failure when verifying the circuit against the specification. Verification engines can be enhanced to support relative timing by generating a set of RT constraints from these verification failure states. The following two explicit relative timing constraints on the burst-mode implementation were generated by an enhanced version of Analyze[1] [15]:

$$\text{RTC4:}\quad \texttt{bz}\uparrow \prec \underline{\texttt{a}}\downarrow$$
$$\text{RTC5:}\quad \texttt{az}\uparrow \prec \underline{\texttt{b}}\downarrow$$

Valid sets of RT constraints are not necessarily unique. The following is another set of RT constraints that are less restrictive because they do not require circuit stability:

$$\text{RTC6:}\quad \texttt{bz}\uparrow \prec \texttt{ab}\downarrow$$
$$\text{RTC7:}\quad \texttt{az}\uparrow \prec \texttt{ab}\downarrow$$

[1]Analyze is a bisimulation verifier. Only hazards that affect the outputs are reported.

These sets of RT constraints rely on delay paths through the environment because $\texttt{az}\uparrow$, $\texttt{bz}\uparrow$, $\underline{\texttt{a}}\downarrow$, and $\underline{\texttt{b}}\downarrow$ are all enabled from $\texttt{z}\uparrow$. One possible implementation that can guarantee that these constraints hold independent of environment delays is shown in Figure 3(b), where a buffer is added at the output. All constraints can be made local to the circuit because the AND gates and the buffer are enabled by signal $\texttt{c}$. Constraints RTC4 and RTC5 can be modified to $\texttt{bc}\uparrow \prec \texttt{z}\uparrow$ and $\texttt{ac}\uparrow \prec \texttt{z}\uparrow$, which hold if the delay through the buffer is larger than through the AND gates.

4.2.2. C-Element summary

Table 2 summarizes the five alternative designs. Except for the static C-Element (SC), all implementations are hazard-free in their respective environments. The speed-independent circuit (SIC) is slower than all others. The relative timing assumption (SIC-RT), which leads to a half size circuit, also enhances performance by 30%. The static SC requires the largest circuit but it is also relatively fast. The reduced domino C-Element (GC-RT) is 15% faster to rise (having only a single pull-up transistor), but is actually slower than the gC on the falling edge. The speed-independent circuits require considerably higher switching energy even when applying RT assumptions. The static implementation without relative timing shows comparative power to the simpler GC and GC-RT circuits largely due to the short circuit current through the keepers as the GC circuits switch. The GC-RT circuit shows higher power consumption than the GC circuit because the removal of the pMOS device results in an additional short-circuit current when $\underline{\texttt{b}}\uparrow$ follows $\underline{\texttt{a}}\uparrow$. The table shows that the static and SI circuits are fully testable for exhaustive patterns, but not when timing reduces signal interleavings (in column RTA2). The RT optimized versions of these circuits are fully testable.

4.3. Timing evolution in a ring

In this section we trace the development of a simple FIFO cell, a simplified abstraction of a part of the RAPPID design [13], following closely the actual steps we have made. We begin with a speed-independent design, and review a succession of progressively simpler circuits, enabled through careful application of relative timing assumptions.

4.3.1. Speed-independent FIFO cell

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

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

Circuit	Has HF Circuit	Fall Delay	Rise Delay	Switching Energy	Area # Transistors	Exhaustive Testability	RTA2 Environment Testability
SIC	Yes	1170pS	1190pS	20.2pJ	16	100%	90%
SIC-RT	Yes	735pS	785pS	14.0pJ	8	n/a	100%
SC	No	700pS	545pS	11.6pJ	18	100%	92%
GC	Yes	640pS	585pS	11.1pJ	10	100%	100%
GC-RT	Yes	530pS	600pS	11.6pJ	9	n/a	100%

Table 2. Comparison of C-Element implementations. Energy is for a complete cycle (rise and fall). Test columns show COSMOS stuck-at fault coverage, with reduced patterns in RTA2 column due to environment restrictions.

The specification in Equation (1) consists of two handshake processes, LEFT and RIGHT. The $\underline{\texttt{c}}$ signal synchronizes the two processes so that $\underline{\texttt{ri}}$ must go low and $\underline{\texttt{li}}$ must rise before both processes may proceed. This process-based specification can easily be mapped to the equivalent Petri-net of Figure 4.

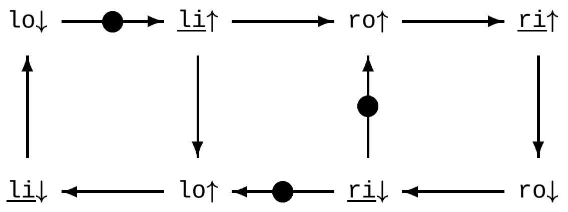


Figure 4. FIFO specification Petri-net

The circuit definition, shown in Figure 5, can be synthesized from this specification using Petrify [5]. This circuit definition uses the complex gate assumptions where the inverters are zero-delay or are combined with the complex gates. This definition, as well as a physical circuit implementation that includes discrete inverters, can be proven to conform to the specification of the FIFO in Equation (1).

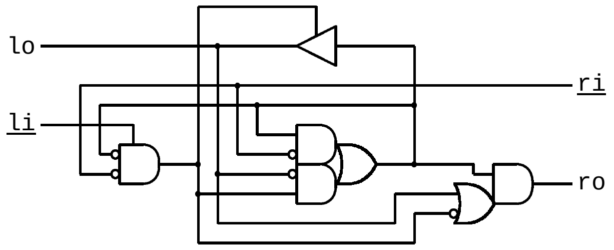


Figure 5. Speed-independent FIFO cell

4.3.2. Burst-mode FIFO cell

The circuit definition of Figure 5 pays a considerable delay penalty to achieve speed independence. Note that $\texttt{lo}\uparrow$ is produced after three complex gate delays, and $\texttt{ro}\uparrow$ in 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:

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

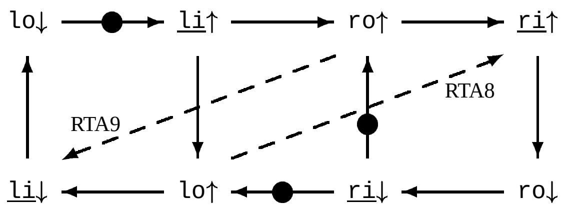


Figure 6. FIFO specification Petri-net with RT constraints RTA8 and RTA9 represented as dashed arcs

A new specification is generated by adding these two relative timing assumptions to the specification. The specification can be represented as

$$\text{FIFO} \wedge \texttt{lo}\uparrow \prec \underline{\texttt{ri}}\uparrow \wedge \texttt{ro}\uparrow \prec \underline{\texttt{li}}\downarrow \qquad (2)$$

where FIFO is the specification from Equation (1). This can be represented in the Petri-net of Figure 6 where the dashed arrows are relative timing constraints.

Note that the two relative timing constraints in RTA8 and RTA9 are in a form where outputs precede inputs. You can also note from the specification that the outputs are enabled concurrently from a pair of inputs. This is exactly a burst-mode constraint [3] 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, shown in Figure 7, assumes that the variance in the generation of the concurrent outputs is always less than the response time of the environment[2].

Incorporating the RT assumptions RTA8 and RTA9 directly into Specification (1) produce the Mealy state machine of Figure 8. This new form is suitable for synthesis:

[2]Applying burst-mode constraints on the signals $\{\underline{\texttt{li}}\downarrow\ \underline{\texttt{ri}}\uparrow\}$ as well results in a C-Element – the micropipelines implementation.

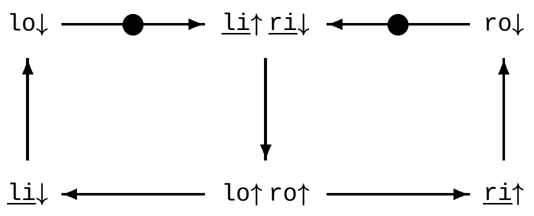


Figure 7. Petri-net with partial burst-mode RT

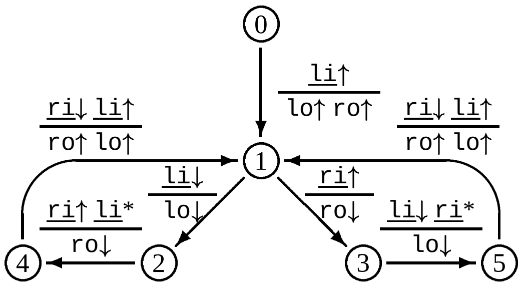


Figure 8. 3D AFSM specification 2

0	1	$\underline{li}\uparrow$	|	$lo\uparrow ro\uparrow$	2	4	$\underline{ri}\uparrow\underline{li}*$	|	$ro\downarrow$
1	2	$\underline{li}\downarrow$	|	$lo\downarrow$	3	5	$\underline{li}\downarrow\underline{ri}*$	|	$lo\downarrow$
1	3	$\underline{ri}\uparrow$	|	$ro\downarrow$	4	1	$\underline{ri}\downarrow\underline{li}\uparrow$	|	$lo\uparrow ro\uparrow$
					5	1	$\underline{ri}\downarrow\underline{li}\uparrow$	|	$lo\uparrow ro\uparrow$

The circuit of Figure 9 was synthesized by 3D. The 3D specification is not identical to Figure 7 due to the implied mutex transitions between $\underline{li}\downarrow$ and $\underline{ri}\uparrow$. However, the synthesized circuit does not require mutual exclusion and implements the Petri-net behavior.

Unbounded delays in the inverters result in critical races which can cause the physical implementation to fail to conform to the specification. However, this circuit can still be a valid implementation for some actual device delays. RT verification by Analyze extracts the critical races in the physical circuit and creates an ordering that must hold for the circuit to operate correctly:

$$\begin{aligned} &\text{RTC10:} \quad \overline{y}\uparrow \prec \underline{li}\uparrow \\ &\text{RTC11:} \quad \underline{y} \prec \overline{ri} \\ &\text{RTC12:} \quad \overline{li}\downarrow \prec ro\uparrow \end{aligned}$$

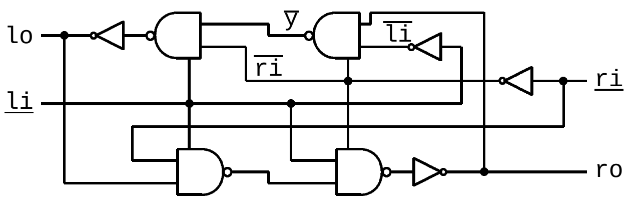


Figure 9. Relative timed burst-mode FIFO

The burst-mode implementation achieves a 2.8× average speedup over the SI circuit. Constraints RTC10–RTC12 apply only to the physical implementation and must be validated by a timing verifier.

4.3.3. 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 BM circuits can safely be used in a large ring. However, if one takes advantage of the timing of the system, an improved circuit (in terms of power, performance, area and testability) can be derived. RTA13 expresses ordering due to timing in a large ring:

$$\text{RTA13:} \quad \underline{ri}\downarrow \prec \underline{li}\uparrow$$

This assumption can be graphically represented as shown in Figure 10, where the dashed arc is the relative timing relation RTA13.

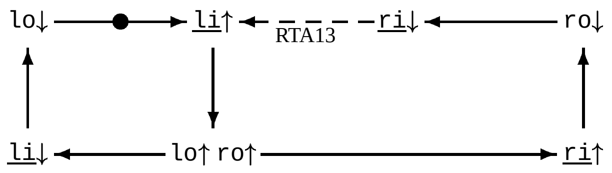


Figure 10. Net representing addition of RT assumptions $\underline{ri}\downarrow \prec \underline{li}\uparrow$

The dashed arc is not a *causal* arc; $\underline{ri}$ must go low before the $\underline{li}$ can rise but $\underline{ri}$ cannot delay $\underline{li}$. 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 $\underline{lo}\downarrow$.

The circuit in Figure 11 can be synthesized with 3D from Specification (2) using assumption RTA13. The rising edge of signal $\underline{li}$ must be delayed sufficiently through lo and the buffer to ensure that the domino AND gate is not disabled before it is fully set. This results in a number of RT constraints on critical races in the circuit that can be derived as was done for RTC4–RTC7 in the SC circuit. This circuit shows 1.7× and 3.6× improvement in worst case performance over the burst-mode and SI circuits respectively, and energy is also improved by 1.8× and 2.1×.

4.3.4. Pulse-mode FIFO cell

RTA13 now constrains the specification sufficiently to derive a pulse-mode circuit. Note that through transitivity,

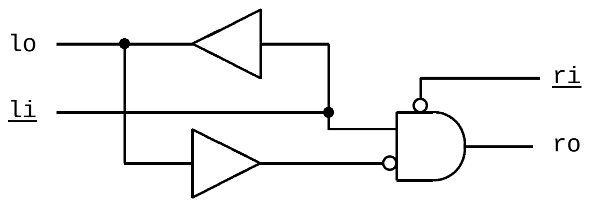


Figure 11. Aggressive relative timed FIFO

ro↓ must also 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 Figure 11.

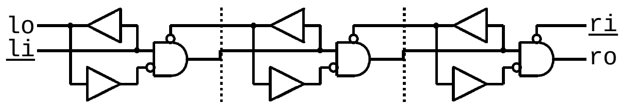


Figure 12. Aggressive relative timed FIFOs

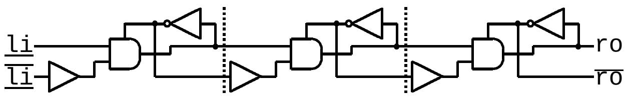


Figure 13. Shuffled aggressive relative timed FIFO cell

Three elements of the ring are shown in Figure 12. 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 Figure 13, that has only forward-moving signals without any inter-cellular feedback. The shuffling that removes acknowledgment is directly based on RTA13 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 Figure 13 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 Figure 14.

The following specification removes the direct handshake signals lo and ri of Specification (1) and adds RTA13:

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

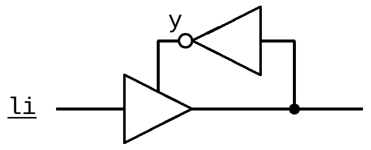


Figure 14. Relative timed pulse-mode FIFO

Designing reliable pulse-mode circuits is very difficult [10]. We can observe some of the constraints of pulse circuits by understanding how we have derived the pulse-mode circuit in this example. Figure 15 shows a four-phase request-acknowledge handshake. Constraints 1 through 4 are causal with speed-independent signaling. By removing the acknowledgment signal (lo and ri in this case), 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. Interestingly, these arcs correspond to a protocol very similar to the standard request acknowledge handshaking.

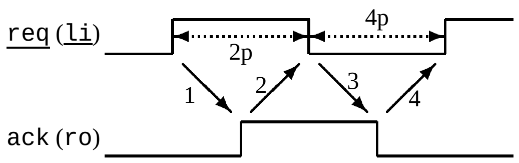


Figure 15. Four cycle and pulse handshake protocol constraints

The pulse on li of Figure 14 causes the output pulse ro, as required by specification (3). If we map req to li and ack to ro in Figure 15, 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 Figure 15 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 Figure 15 is a necessary RT constraint for the circuit to work. Finally, arc 4 is assumed to hold from RTA13 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.

Circuit	Has HF Circuit	Worst Delay	Average Delay	Switching Energy	Area # Trans.	SI Env. Testability	RTA13 Environment Testability	Pulse-Mode Testability
SI	Yes	2160pS	1560pS	37.6pJ	39	98%	91%	n/a
RT-BM	No	1020pS	550pS	32.2pJ	40	95%	74%	n/a
RT-Agr	No	595pS	390pS	18.2pJ	20	n/a	100%	n/a
Pulse	No	350pS	350pS	16.2pJ	17	n/a	n/a	100%

Table 3. Comparison of FIFO implementations. Energy accounts for a complete four-phase cycle. Synchronous testing in COSMOS required extra test gate for pulse circuit.

4.3.5. Ring summary

Some consequences of evolving a simple FIFO-like controller from a speed-independent to a pulse-mode circuit are summarized in Table 3. The different circuits are characterized in terms of robustness, performance, power, area, and testability. The latency of the SI circuit is from three to five times longer than the circuits that use timing. The 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 generally increase the testability. Note that the most significant improvements in performance, area and power have all been achieved by the burst-mode and aggressive RT transformations. The additional savings awarded by going to pulse mode are much less pronounced. Indeed, the 'aggressive' RT controller may already be considered a pulse mode circuit. We feel that testability is increased using relative timing because many of the redundant coverings are removed when the circuits are optimized for time.

4.4. 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 [13] 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 [13]. A critical component in RAPPID is the Tag Unit, which synchronizes the serial ordering of instructions. The tagging control signals interconnect the Tag Units to form a 4×16 torus.

Assume that the simplified interfaces of Figure 16 are all speed-independent interfaces. This requires request/acknowledge handshakes; a four-phase protocol is used. The three behaviors in the boxes are specified as follows:

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

The two PA active processes synchronize the four-phase handshake after $\underline{\texttt{r}}$ requests are received, while the two PB processes are passive and synchronize before handshaking. Therefore, when the $\underline{\texttt{irdy}}$ and $\underline{\texttt{ti}}$ 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:

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

The implementation of these processes using ATACS is shown in Figure 17. 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.

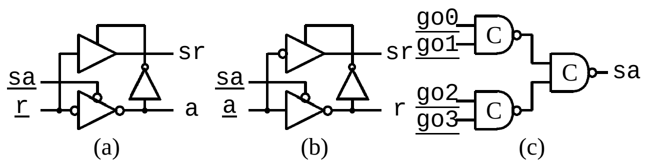


Figure 17. Speed-independent Tag Unit circuits: (a) PA (b) PB (c) C4

The circuit used in RAPPID is shown in Figure 18. This efficient circuit is very similar to the simplified FIFO derived in Section 4.3, with the extra gates being used to steer

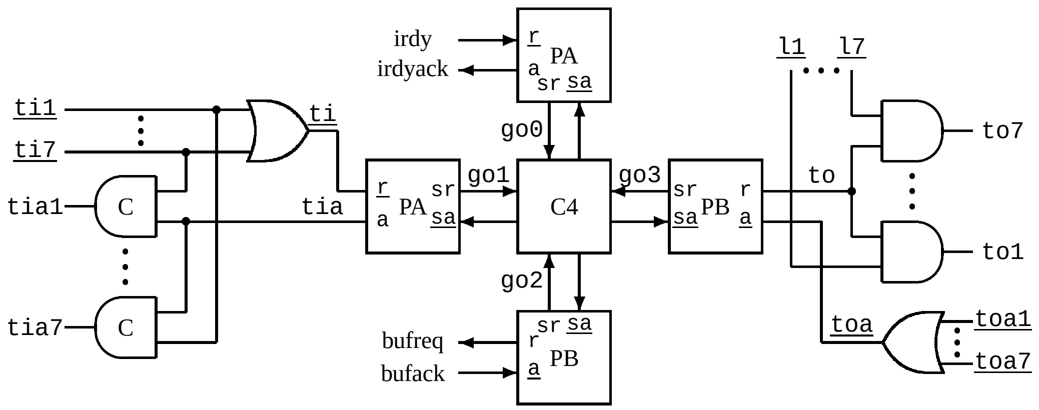


Figure 16. SI Tag Unit. Assumes tagin (ti) handshakes are mutex.

the tag paths based on the instruction length. 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 $\underline{\texttt{irdy}}$ and `bufreq` paths are combinations of four-phase and pulse-mode signaling – `irdyack` and `bufreq` being pulses.

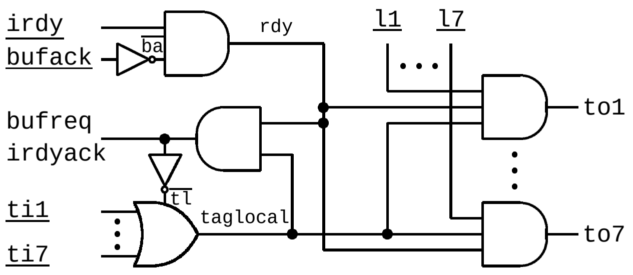


Figure 18. Simplified RAPPID Tag Unit

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` are concurrent. The timing assumptions necessary to create the circuit can be classified by type according to Figure 15. The type 4 assumptions on the $\underline{\texttt{ti}}$ and `to` signals are encoded into the specification since the TAGIN and TAGOUT processes have been combined. The synchronization signals `c1` and `c2` in the specification encode the causal transitions of type 1. RTA14–RTA16 encode the type 2p transitions – minimum pulse-widths constraints on `to`, `bufreq`, and `irdyack`. (When multiple signals precede another we can include them as a set in one constraint.) Assumptions RTA17–RTA19 are type 3 constraints, ensuring that the input pulse lowers before the output pulse. RTA20 and RTA21 are type 4 assumptions which require the pulses return to the stable state before the next tagin arrives. Assumptions RTA22 and RTA23 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.)

2p	RTA14:	$\{\texttt{bufreq}\uparrow, \texttt{irdyack}\uparrow\} \prec \texttt{to}\downarrow$
2p	RTA15:	$\{\texttt{to}\uparrow, \texttt{irdyack}\uparrow\} \prec \texttt{bufreq}\downarrow$
2p	RTA16:	$\{\texttt{to}\uparrow, \texttt{bufreq}\uparrow\} \prec \texttt{irdyack}\downarrow$
3	RTA17:	$\underline{\texttt{ti}}\downarrow \prec \texttt{to}\downarrow$
3	RTA18:	$\underline{\texttt{ti}}\downarrow \prec \texttt{bufreq}\downarrow$
3	RTA19:	$\underline{\texttt{ti}}\downarrow \prec \texttt{irdyack}\downarrow$
4	RTA20:	$\{\texttt{bufreq}, \underline{\texttt{bufack}}\uparrow, \texttt{irdyack}, \underline{\texttt{irdy}}\downarrow\} \prec \underline{\texttt{ti}}\uparrow$
4	RTA21:	$\{\texttt{to}, \texttt{bufreq}, \underline{\texttt{bufack}}\uparrow, \overline{\texttt{ba}}\downarrow\} \prec \underline{\texttt{irdy}}\uparrow$
	RTA22:	$\texttt{irdyack}\downarrow \prec \underline{\texttt{irdy}}\downarrow$
	RTA23:	$\texttt{bufreq}\downarrow \prec \underline{\texttt{bufack}}\downarrow$

$$
\begin{aligned}
\text{TAGS} &= \underline{\texttt{b1}}.\underline{\texttt{ti}}\uparrow.\texttt{c1}.(\underline{\texttt{ti}}\downarrow \mid \underline{\texttt{c2}}.\texttt{to}\uparrow.\texttt{to}\downarrow).\text{TAGS}\\
\text{BUF} &= \underline{\texttt{c1}}.\underline{\texttt{c2}}.\texttt{bufreq}\\
&\quad .(\texttt{bufreq} \mid \underline{\texttt{bufack}}.\underline{\texttt{bufack}}).\text{BUF}\\
\text{IRDY} &= \underline{\texttt{irdy}}.(\underline{\texttt{b2}}.\texttt{c2}.\texttt{irdyack}\\
&\qquad .(\texttt{irdyack}. \mid \underline{\texttt{irdy}}).\text{IRDY}\\
&\qquad + \underline{\texttt{nott}}.\underline{\texttt{irdy}}.\underline{\texttt{nott}}.\text{IRDY})\\
\text{MUTEX} &= (\texttt{b1}.\texttt{b2} + \texttt{nott}.\texttt{nott}).\text{MUTEX}\\
\text{TAG} &= (\text{TAGS} \mid \text{BUF} \mid \text{IRDY} \mid \text{MUTEX})\\
&\quad \backslash\{\underline{\texttt{c1}}, \underline{\texttt{c2}}, \underline{\texttt{b1}}, \underline{\texttt{b2}}, \underline{\texttt{nott}}\}\\
&\quad \wedge \text{RTA14} - \text{RTA23}
\end{aligned}
\tag{5}
$$

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 [15]. RTC24 and RTC25 are the type 2 constraints, RTC26–RTC28 are type 3 (the same as RTA17–RTA19 in the specification), RTC29–RTC32 the type 4 constraints, and type 4p RTC33–RTC34 constraints. Note that a single delay path constraint may include several RT constraints as we have used them here.

Circuit	Tag Latency	Cycle Time	Cycle Energy	Area # Trans.	RAPPID Testability
SI	4.75nS	9.68nS	255pJ	294	n/a
RAPPID	1.27nS	2.61nS	63pJ	85	98.6%

Table 4. Comparison of RAPPID Tag Unit with the SI version. Area is the number of transistors, testability refers to the complete RAPPID Tag Unit and steering logic.

$$\begin{array}{lll}
2 & \text{RTC24:} & \texttt{to}\uparrow \prec \texttt{taglocal}\downarrow \\
2 & \text{RTC25:} & \{\texttt{irdyack}\uparrow, \texttt{to}\uparrow, \overline{\texttt{tl}}\downarrow\} \prec \texttt{rdy}\downarrow \\
3 & \text{RTC26:} & \underline{\texttt{ti}}\downarrow \prec \texttt{to}\downarrow \\
3 & \text{RTC27:} & \underline{\texttt{ti}}\downarrow \prec \texttt{br}\downarrow \\
3 & \text{RTC28:} & \underline{\texttt{ti}}\downarrow \prec \texttt{irdyack}\downarrow \\
4 & \text{RTC29:} & \texttt{rdy}\downarrow \prec \texttt{taglocal}\uparrow \\
4 & \text{RTC30:} & \texttt{rdy}\downarrow \prec \overline{\texttt{ba}}\uparrow \\
4 & \text{RTC31:} & \{\texttt{taglocal}\downarrow, \overline{\texttt{tl}}\uparrow\} \prec \underline{\texttt{ti}}\uparrow \\
4 & \text{RTC32:} & \texttt{taglocal}\downarrow \prec \texttt{rdy}\uparrow \\
4\text{p} & \text{RTC33:} & \{\underline{\texttt{ba}}\uparrow, \overline{\texttt{ba}}\downarrow\} \prec \underline{\texttt{irdy}}\uparrow \\
4\text{p} & \text{RTC34:} & \texttt{taglocal}\downarrow \prec \overline{\texttt{tl}}\uparrow
\end{array} \qquad (6)$$

We feel that attaching many, if not all, of the timing constraints as RT predicates make the specification more perspicuous as well as explicitly annotating the timing requirements. Each process represents an interface with a simple definition, which is refined by timing assumptions as predicates. Incorporating the assumptions into the specification removes much of the clarity of the required synchronizations and orderings. Representing the complete behavior constraints or timing constraints as a Petri-net, as was shown in Section 4.3, can be illucidating 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.

A comparison of the two implementations is made in Table 4. The RT circuit shows a 3.5× area, 4× power, and 3.7× 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 can fairly directly map to improvements in RAPPID [13]. While the area of this controller is a fraction of RAPPID, the area impact on RAPPID from the RT circuit is arguably much higher than the size of the controller. The RAPPID architecture can be scaled to reach a higher performance. 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 through scaling if the performance goals were to be met. The area savings in terms of the 50% reduction in wire count 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.

5. 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 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. In a restricted form races in speed-independent implementations due to inverter delays can be discovered, and shown to not be critical, through relative timing. Burst-mode constraints are an example of conservative implicit application of RT. 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.

Relative timing was a large factor in the quality of the RAPPID results in terms of throughput, power, area, testability, and latency [13]. The benefit is shown through applying relative timing to the examples in this text.

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.

References

[1] R. Alur and D. L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[2] S. Chakraborty, K. Y. Yun, and D. L. Dill. Practical timing analysis of asynchronous systems using time separation of events. In *Proc. IEEE Custom Integrated Circuits Conference*, May 1998.

[3] 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*, pages 193–208, April 1993.

[4] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, A. Taubin, and A. Yakovlev. Lazy transition systems: application to timing optimization of asynchronous circuits. In *Proc. International Conf. Computer-Aided Design (ICCAD)*, pages 324–331, November 1998.

[5] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. Petrify: a tool for manipulating concurrent specifi cations and synthesis of asynchronous controllers. *IEICE Transactions on Information and Systems*, E80-D(3):315–325, 1997.

[6] H. Hulgaard. *Timing Analysis and Verification of Timed Asynchronous Circuits*. PhD thesis, Department of Computer Science, University of Washington, 1995.

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

[8] C. J. Myers. *Computer-Aided Synthesis and Verification of Gate-Level Timed Circuits*. PhD thesis, Dept. of Elec. Eng., Stanford University, October 1995.

[9] C. J. Myers, T. G. Rokicki, and T. H.-Y. Meng. Automatic synthesis and verifi cation of gate-level timed circuits. Technical Report CSL-TR-94-652, Stanford University, January 1995.

[10] V. Narayanan, B. A. Chappell, and B. M. Fleischer. Static Timing Analysis For Self Resetting Circuits. In *International Conference on Computer-Aided Design (ICCAD-96)*. IEEE Computer Society, November 1996.

[11] R. Negulescu and A. Peeters. Verifi cation of speed-dependences in single-rail handshake circuits. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 159–170, 1998.

[12] K. J. Nowka and T. Galambos. Circuit Design Techniques for a Gigahertz Integer Microprocessor. In *1998 IEEE International Converence on Computer Design: VLSI in Computers & Processors (ICCD98)*, pages 11–16. IEEE Computer Society, October 1998.

[13] S. Rotem, K. Stevens, R. Ginosar, P. Beerel, C. Myers, K. Yun, R. Kol, C. Dike, M. Roncken, and B. Agapiev. RAPPID: An Asynchronous Instruction Length Decoder. In *5th International Symposium on Advanced Research in Asynchronous Circuits and Systems*. IEEE, April 1999.

[14] M. Shams, J. C. Ebergen, and M. I. Elmasry. Modeling and Comparing CMOS Implementations of the C-Element. *IEEE Transactions on VLSI Systems*, 6(4):563–567, December 1998.

[15] K. S. Stevens. *Practical Verification and Synthesis of Low Latency Asynchronous Systems*. PhD thesis, University of Calgary, Calgary, Alberta, September 1994.

[16] F. C. D. Young, K. S. Stevens, and R. P. Graham Jr. Timed Logic Conformance and its Application. In *1999 International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU99)*. ACM/IEEE, March 1999.

[17] K. Y. Yun. *Synthesis of Asynchronous Controllers for Heterogeneous Systems*. PhD thesis, Stanford University, Aug. 1994.