Timing Analysis of Cyclic Combinational Circuits*

Marc D. Riedel and Jehoshua Bruck
California Institute of Technology
Mail Code 136-93, Pasadena, CA 91125
E-mail: {riedel, bruck}@paradise.caltech.edu

Abstract— **The accepted wisdom is that combinational circuits must have *acyclic* (i.e., loop-free or feed-forward) topologies. And yet simple examples suggest that this need not be so. In previous work, we advocated the design of *cyclic* combinational circuits (i.e., circuits with loops or feedback paths). We proposed a methodology for analyzing and synthesizing such circuits, with an emphasis on the optimization of area.**

In this paper, we extend our methodology into the temporal realm. We characterize the true delay of cyclic circuits through symbolic event propagation in the *floating mode* of operation, according to the *up-bounded inertial delay* model. We present analysis results for circuits optimized with our program CYCLIFY**. Some benchmark circuits were optimized significantly, with simultaneous improvements of up to 10% in the area and 25% in the delay.**

I. Introduction

A collection of logic gates forms a *combinational* circuit if the outputs can be described as boolean functions of the current input values only. A common misconception is that combinational circuits must have acyclic topologies; that is to say, they must be designed without any loops or feedback paths. In fact, the idea that "combinational" and "acyclic" are synonymous terms is so thoroughly ingrained that many textbooks provide the latter as a definition of the former.

And yet, circuits with cyclic topologies can be combinational. Examples are shown in Figures 1 and 5. How can we characterize the temporal behavior of such circuits?

In the conventional view, timing analysis is predicated on a *topological ordering*. The computation of arrival times proceeds from the gates at the top of the ordering – those connected only to primary inputs – down to those at the bottom – those producing the primary outputs.

*This work is supported in part by a grant from the National Human Genome Research Institute (Grant no. P50 HG02370).

With a cyclic circuit, where do we begin?

Example 1

The circuit in Figure 1 consists of six AND and OR gates, with two primary outputs, f_1 and f_2, and five primary inputs a, b, c, d and x (note that the input x is repeated).

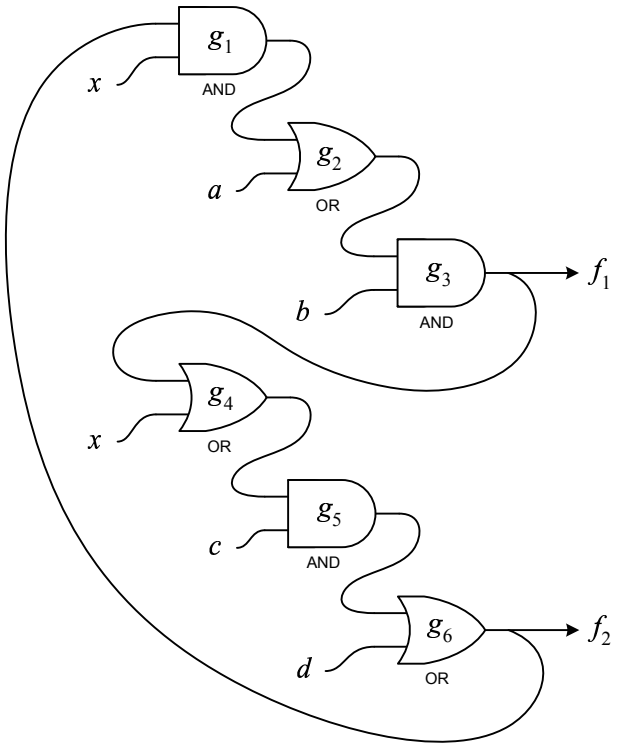


Fig. 1. A cyclic combinational circuit.

Since the gates in this example are connected in a cycle, we cannot establish an ordering *a priori*. And yet, consider what happens when we apply specific input values. (Assume that the gates each have a delay bound of 1 time unit.)

• If $x = 0$, then gate g_1 produces a value of 0 after one time unit, since 0 is a controlling value for an AND gate. In this case, the gates may be ordered

$$g_1 \rightarrow g_2 \rightarrow g_3 \rightarrow g_4 \rightarrow g_5 \rightarrow g_6.$$

Outputs arrive at f_1 and f_2 after at most 3 and at most 6 time units, respectively.

• If $x = 1$, then gate g_4 produces a value of 1 after one time unit, since 1 is a controlling value for an OR gate. In this case, the gates may be ordered

$$g_4 \rightarrow g_5 \rightarrow g_6 \rightarrow g_1 \rightarrow g_2 \rightarrow g_3.$$

Outputs arrive at f_1 and f_2 after at most 6 and at most 3 time units, respectively.

In both cases the outputs may arrive earlier, depending on the values of a, b, c and d.

We conclude that the circuit is combinational – x must assume one of these two values – and that the maximum delay is 6 time units. It may be shown that the circuit implements the functions [1]

$$\begin{aligned} f_1 &= b(a + x(d + c)), \\ f_2 &= d + c(x + b\,a). \end{aligned}$$

Note that both functions depend on all 5 variables. It may be shown that if we implement these functions with an acyclic circuit, at least 8 fan-in two gates are required.

A. Related Work

In an earlier era, theoreticians commented on the possibility of having cycles in combinational logic, and conjectured that this might be a useful property [6], [7], [19]. Both McCaw and Rivest presented examples of cyclic circuits with provably fewer gates than is possible with equivalent acyclic circuits [11], [16]. (We have extended and generalized these theoretical results. Most notably, we have constructed a family of circuits with cyclic topologies having half as many gates as is possible with acyclic topologies [15]).

In a later era, practitioners observed that cycles sometimes appear in combinational circuits synthesized from high-level descriptions. Stok noted that cycles are occasionally introduced during resource-sharing optimizations at the level of functional units [20]. However, most synthesis and verification tools balk when given combinational logic with cycles. The accepted strategy has been to simply disallow cycles in the high-level phases.

Motivated by Stok's observation, Malik discussed analysis techniques for cyclic circuits [10]. He formulated a symbolic analysis algorithm based on ternary-valued simulation. Shiple refined and formalized Malik's results, and extended the concepts to combinational logic embedded in sequential circuits [18].

Malik discussed timing analysis of cyclic circuits, but concluded that the problem of computing the true delay of sensitizable paths is unmanageable. Instead, he proposed a topological approach, beginning with a transformation from a cyclic specification to an equivalent acyclic one. Recently, Edwards presented work in a similar vein [5]. However, as Malik and Edwards admitted, unraveling cyclic circuits this way is an inherently complex problem. It does not scale well, and is particularly ill-suited for circuits with deeply-nested cyclic topologies.

B. Contributions

In previous work, we advocated the design of combinational circuits with cycles, and demonstrated that such an approach permits significant optimizations of area. We discussed analysis techniques for validating cyclic circuits [13], and suggested synthesis strategies [14]. In this work, we extend our methodology into the temporal realm.

We characterize the timing of circuits according to a standard model, described in Section I-C: the gates operate in the so-called *floating mode*, with *up-bounded inertial* delays [4].

Instead of a path-based approach, we compute arrival times through event propagation [1], [21]. Of course, we do not apply this process exhaustively for all possible input assignments. Rather we perform the computation symbolically, with decision diagrams [2], [9]. In Section II, we describe the algorithm and illustrate its application on examples.

Our approach is not a radical departure from existing practice. In fact, the salient message of this paper is that well-developed techniques for timing analysis can readily be applied to cyclic circuits.

We have incorporated timing analysis into our cyclic optimization program, called CYCLIFY, built within the Berkeley SIS environment [17]. Although not the primary focus of this paper, in Section III we discuss synthesis strategies targeting delay. Our previous work was limited to cyclic optimizations in the re-structuring and minimization phases of synthesis. In our current work, we extend these optimizations to the decomposition and technology mapping phases.

For the synthesis results in Section III, we applied the standard "*script.delay*" optimization sequence, followed by mapping to NAND2/NOR2 gates. In trials

[1] We use the standard notation: addition (+) denotes disjunction (OR), multiplication (·) denotes conjunction (AND), and an overbar ($\bar{x}$) denotes negation (NOT).

with benchmarks, we achieved simultaneous improvements of up to 10% in the area and 25% in the delay, as compared to the standard SIS optimizations.

C. Circuit Model

The concepts discussed in this paper are not tied to any particular physical model or computing substrate. In our discussion of timing analysis, the exposition is at a *symbolic* level, that is to say, in terms of boolean expressions. However, we first discuss the circuit model in an explicit sense – in terms of signal values.

We work with the digital abstraction of zeros and ones. Nevertheless, our model recognizes that the underlying signals are, in fact, analog: each signal is a continuous real-valued function of time, corresponding to a voltage level. For analysis, we adopt a *ternary* framework, extending the set of *boolean* values $\mathbb{B} = \{0, 1\}$ to the set of ternary values $\mathbb{T} = \{0, 1, \perp\}$. Here $\perp$ represents either an *ambiguous* value, e.g., a voltage value between logical 0 and logical 1, or else an *uncertain* value, i.e., a signal that might be 0 or 1 – but we do not know which.

The idea of three-valued logic for circuit analysis is well established. It was originally proposed for the analysis of *hazards* in combinational logic [22]. Bryant popularized its use for verification [3], and it has been widely adopted for the analysis of asynchronous circuits [4]. For a theoretical treatment, see [12]. Malik and Shiple discuss the analysis of cyclic circuits in this framework [10], [18].

Central to timing analysis is the concept of *controlling* values. In Riedel's Ph.D. dissertation, a formalism is presented for computing the controlling values of arbitrary logic functions using the so-called *marginal* operator [15]. For simplicity, in this paper we assume that the network has been decomposed into primitive gates, namely AND/OR/NAND/NOR gates and inverters. Recall that 0 is the controlling value for an AND gate, as shown in Figure 2. Similarly, 1 is the controlling value for an OR gate.

Our analysis characterizes the functional and temporal behavior of circuits according to the so-called *floating-mode* assumption [4]: at the outset, all wires in a circuit are assumed to have unknown or possibly undefined values, and so assigned the value $\perp$. Although conservative, this assumption ensures that the analysis does not infer stability in cases where ambiguous or unstable signals might persist.

Consider the circuit fragment in Figure 3. One might be tempted to reason as follows: The output of the AND gate g_1 is fed in complemented and uncomplemented form into the OR gate g_2. Thus, one of the inputs to the OR gate must be 1, and so its output must be 1.

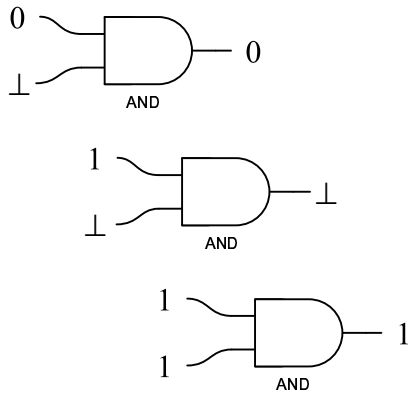


Fig. 2. An AND gate with 0, 1, and $\perp$ inputs.

And yet, by definition, $\perp$ designates an *undefined* value. For instance, it could indicate a voltage value exactly half way between logical 0 and logical 1. Within the floating-mode framework, we remain agnostic: the output of the OR gate is $\perp$.

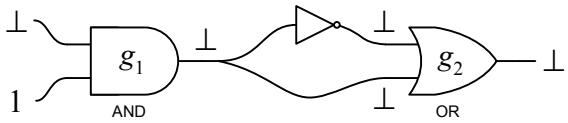


Fig. 3. An illustration of the floating mode.

We assume an idealized model for timing analysis: each gate is characterized by a single parameter, a bound on its delay t_d. If a gate's inputs assume controlling values by time t, then the gate's output assumes a definite value between time t (i.e., immediately) and time $t + t_d$. This is called the *up-bounded inertial delay* model. We assume that the wires have zero propagation delay.

For the examples in this paper, we assign a delay bound of 1 time unit to each gate. (This is *not* the so-called *unit-delay* model, in which gate delays are assumed to be *exactly* 1; rather, it is the up-bounded inertial delay model with an upper bound of 1 for each gate.) More sophisticated models of temporal behavior could readily be incorporated into our algorithms; we neglect such details here in order to focus on the core issues.

II. Analysis

Conceptually, the analysis is just an algorithmic implementation of the idea illustrated in Example 1. We apply definite values to the primary inputs, and

track the propagation of signal values. Once we have established that a definite value has appeared on a gate output, this value persists for the duration of the analysis. The arrival time of a well-defined value at a gate output is determined either:

- by the arrival time of the earliest *controlling* input value;
- or by the arrival time of the latest *non-controlling* input value.

The analysis proceeds in time intervals. If the gates have fixed delay bounds, we can choose the interval length to match the shortest delay bound. In each time interval, we evaluate all the gates that received new input values in the previous interval. In this manner, we are assured that we know the earliest time that signal values becomes known. If definite boolean values never arrive at one or more of the primary outputs, then we conclude that the circuit is not combinational.

For most of the circuits encountered in practice, explicit analysis is not a viable option. With n inputs, there would be 2^n input assignments to consider. Instead, we tackle the problem with symbolic techniques, in which we manipulate *sets* of input assignments.[2] In our current implementation, we use binary decision diagrams [2], [9]. However, we note that the use of boolean satisfiability (SAT)-based techniques [8] would probably be more computationally efficient.

Algorithm 1: Symbolic Timing Analysis

Let $X = (x_1, \ldots, x_n)$ be the primary inputs. We maintain a pair of *characteristic sets* for the output of each gate g_i. The first

$$C_i^{(0)}(X),$$

consists of the set of input assignments for which the gate evaluates to 0; the second,

$$C_i^{(1)}(X),$$

the set for which it evaluates to 1. Implicitly, the complement of the union of these two sets is the set of assignments for which the gate evaluates to $\perp$.

[2]In a symbolic formulation, a set of input assignments is characterized by a boolean function: the function evaluates to 1 for those assignments *in* the set, and to 0 for those *not* in the set. Thus, OR corresponds to the union, AND to the intersection, and NOT to the complement of sets.

At the outset, all wires are assumed to have undefined values, so the characteristic sets are empty,

$$C_i^{(0)} := C_i^{(1)} := 0.$$

As the analysis proceeds, input assignments that induce gates to produce definite output values are added to these sets. Call the addition of input assignments to the set $C_i^{(v)}$, for some $v \in \{0, 1\}$, an *arrival event* valued v at gate g_i.

Initialization

The initial arrival events occur at gates controlled by primary inputs. For instance, suppose that an AND gate g_i is connected to the primary input x. We have an initial arrival event

$$C_i^{(0)} := \bar{x}.$$

Similarly, suppose that an OR gate g_j is connected to the primary input y. We have an initial arrival event

$$C_j^{(1)} := y.$$

We compute such arrival events for all gates attached to the primary inputs.

Propagation

In each interval, we compute new arrival events for gates based on the antecedent arrival events on their inputs.

1. Suppose that in the previous interval there was an arrival event valued v at gate g_i; that g_i is a fan-in to gate g_j; and that v is a controlling input value for g_j, producing an output value w. We compute

$$C_j^{(w)} := C_j^{(w)} + C_i^{(v)}.$$

If $C_j^{(w)}$ changes as a result (i.e., $C_i^{(v)}$ was not contained in $C_j^{(w)}$), then we have a new arrival event valued w at g_j.

2. Suppose that in the previous interval there was an arrival event valued v at gate g_i; that g_i is a fan-in to gate g_j; and that v is a non-controlling input value for g_j. Let $g_{i_1}, \ldots, g_{i_k}$ be *all* the gates that fan-in to g_j, and let $v_{i_1}, \ldots, v_{i_k}$ be the non-controlling values for these fan-in gates. Suppose that these non-controlling inputs produce an output value w for g_j. We compute

$$C_j^{(w)} := C_j^{(w)} + \left[C_{i_1}^{(v_{i_1})} \cdots C_{i_k}^{(v_{i_k})} \right].$$

Again, if $C_j^{(w)}$ changes as a result, then we have a new arrival event valued w at g_j.

To illustrate these propagation conditions, suppose that we have an AND gate g_3 with fan-in gates g_1 and g_2, as shown in Figure 4.

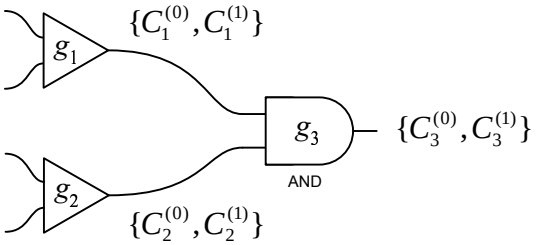


Fig. 4. An illustration of the propagation conditions.

Suppose that the characteristic sets are

$$\begin{aligned} C_1^{(0)} &= x_1, & C_1^{(1)} &= x_2, \\ C_2^{(0)} &= x_3, & C_2^{(1)} &= x_4, \\ C_3^{(0)} &= x_1 + x_3, & C_3^{(1)} &= x_2\, x_4. \end{aligned}$$

Now suppose that there is an arrival event valued 0 at g_1 setting

$$C_1^{(0)} = x_1 + x_5.$$

In the next interval, we compute

$$C_3^{(0)} := C_3^{(0)} + C_1^{(0)} = x_1 + x_3 + x_5.$$

Now suppose that there is an arrival event valued 1 at g_1 setting

$$C_1^{(1)} = x_2 + x_6.$$

In the next interval, we compute

$$C_3^{(1)} := C_3^{(1)} + \left[C_1^{(1)}\, C_2^{(1)}\right] = (x_2 + x_6)x_4.$$

Termination

Termination is guaranteed since the cardinality of the characteristic sets either increases or remains unchanged with arrival events. A characteristic set cannot grow beyond the size of the full set of input assignments.

When the algorithm terminates, the union of the characteristic sets

$$C_i^{(0)} + C_i^{(1)}$$

for each gate g_i specifies the input assignments for which g_i produces definite values. If the complement of this union includes input assignments not in the "don't care" set for any gate producing a primary output, then we conclude that the circuit is *not* combinational. In particular, if there are no "don't care" input assignments, then the circuit is combinational if and only if the union consists of *all* input assignments for every gate producing a primary output.

Also, when the algorithm terminates, the time that has lapsed – the number of intervals times the interval length – gives a bound on the circuit delay.

Analysis of Example 1

We step through a symbolic analysis of the circuit in Figure 1 of the Introduction. We assume that each gate has a delay bound of 1 time unit, and that the primary inputs arrive at time 0.

Time 1

For the AND gates, controlling values of 0 on the primary inputs result in

$$C_1^{(0)} = \bar{x}, \quad C_3^{(0)} = \bar{b}, \quad C_5^{(0)} = \bar{c}.$$

For the OR gates, controlling values of 1 on the primary inputs result in

$$C_2^{(1)} = a, \quad C_4^{(1)} = x, \quad C_6^{(1)} = d.$$

Time 2

For the AND gates, non-controlling values of 1 from the preceding OR gates result in

$$C_1^{(1)} = x\, d, \quad C_3^{(1)} = b\, a, \quad C_5^{(1)} = c\, x.$$

For the OR gates, non-controlling values of 0 from the preceding AND gates result in

$$C_2^{(0)} = \bar{a}\, \bar{x}, \quad C_4^{(0)} = \bar{x}\, \bar{b}, \quad C_6^{(0)} = \bar{d}\, \bar{c}.$$

Time 3

For the AND gates, controlling values of 0 from the preceding OR gates result in

$$C_1^{(0)} = \bar{x} + \bar{d}\, \bar{c}, \quad C_3^{(0)} = \bar{b} + \bar{a}\, \bar{x}, \quad C_5^{(0)} = \bar{c} + \bar{x}\, \bar{b}.$$

For the OR gates, controlling values of 1 from the preceding AND gates result in

$$C_2^{(1)} = a + x\,d, \quad C_4^{(1)} = x + b\,a, \quad C_6^{(1)} = d + c\,x.$$

Controlling and non-controlling values continue to propagate forward, in alternate fashion.

Time 6

Skipping forward, after six intervals we have:

$$\begin{aligned}
C_1^{(0)} &= \bar{x} + \bar{d}\,\bar{c}, & C_1^{(1)} &= x(d+c),\\
C_2^{(0)} &= \bar{a}(\bar{x} + \bar{d}\,\bar{c}), & C_2^{(1)} &= a + x(d+c),\\
C_3^{(0)} &= \bar{b} + \bar{a}(\bar{x} + \bar{d}\,\bar{c}), & C_3^{(1)} &= b(a + x(d+c)),\\
C_4^{(0)} &= \bar{x}(\bar{b} + \bar{a}), & C_4^{(1)} &= x + b\,a,\\
C_5^{(0)} &= \bar{c} + \bar{x}(\bar{b} + \bar{a}), & C_5^{(1)} &= c(x + ba),\\
C_6^{(0)} &= \bar{d}(\bar{c} + \bar{x}(\bar{b} + \bar{a})), & C_6^{(1)} &= d + c(x + b\,a).
\end{aligned}$$

At this point, there are no new arrival events. Note that for each $i = 1, \ldots, 6$

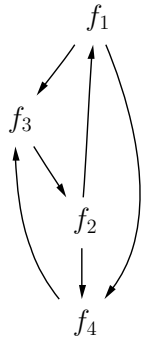
$$C_i^{(0)} + C_i^{(1)} = 1.$$

Hence, all input assignments produce definite values at the outputs, and so we conclude that the circuit is combinational. Since we propagated events for 6 time units, we conclude that the circuit has delay 6. □

Example 2

Consider the circuit shown in Figure 5. It computes four output functions, f_1, f_2, f_3, and f_4 of three input variables a, b, and c. The corresponding equations are:

$$\begin{aligned}
f_1 &= bc + \bar{b}\,\bar{f}_2\\
f_2 &= ac + b\bar{f}_3\\
f_3 &= a\bar{f}_1 + bf_1 + \bar{c}\bar{f}_4\\
f_4 &= a\bar{b} + \bar{f}_1 + b\bar{f}_2
\end{aligned}$$



Note that there are cyclic dependencies: f_1 depends on f_2; f_2 depends on f_3; f_3 depends on f_1 and f_4; and f_4 depends on f_1 and f_2. Nevertheless, this circuit is combinational with delay 8.

We do not trace through the analysis this time. The table in Figure 6 summarizes the results. It gives the characteristic sets $C_i^{(0)}$ and $C_i^{(1)}$ for the output gates. □

	g_3	
$[b\bar{c},$	bc	$]_2$
$[a\bar{b}c + b\bar{c},$	$c(b + \bar{a}) + \bar{b}\bar{c}$	$]_4$
	g_6	
$[\bar{b}(\bar{c} + \bar{a}),$	ac	$]_2$
$[\bar{a}(c + \bar{b}) + a\bar{c},$	ac	$]_6$
$[\bar{a}(c + \bar{b}) + a\bar{c},$	$\bar{a}b\bar{c} + ac$	$]_7$
	g_{10}	
$[\bar{a}\bar{b}c,$	0	$]_2$
$[\bar{a}\bar{b}c,$	$b(c + a)$	$]_4$
$[\bar{a}(b\bar{c} + \bar{b}c),$	$b(c + a)$	$]_5$
$[\bar{a}\bar{b}c + \bar{c}(a\bar{b} + \bar{a}b),$	$c(b + a) + ab$	$]_6$
$[\bar{a}\bar{b}c + \bar{c}(a\bar{b} + \bar{a}b),$	$\bar{a}\bar{b}\bar{c} + c(b + a) + ab$	$]_7$
	g_{13}	
$[0,$	$a\bar{b}$	$]_2$
$[0,$	$b\bar{c} + a\bar{b}$	$]_3$
$[abc,$	$b\bar{c} + a\bar{b}$	$]_4$
$[abc + \bar{a}\bar{b},$	$b\bar{c} + a\bar{b}$	$]_5$
$[abc + \bar{a}\bar{b},$	$b(\bar{c} + \bar{a}) + a\bar{b}$	$]_8$

Fig. 6. Characteristic sets $[C_i^{(0)}, C_i^{(1)}]_j$ for the circuit of Figure 5, for gates g_i, $i = 3, 6, 10, 13$, at time intervals $j = 2, \ldots, 8$.

Timing analysis with such an idealized model is transparent. However, the devil is in the details – and with realistic timing models there are *many* detailed aspects to consider. Nevertheless, we conclude that, at least in a conceptual sense, the analysis of cyclic circuits is no more complicated than that of acyclic circuits. We can perform this task efficiently through symbolic event propagation, within the ternary framework.

III. Results

In previous work, we discussed design strategies for cyclic circuits and described our synthesis program, called CYCLIFY [14], built within the Berkeley SIS environment [17]. In our methodology, cycles are introduced in the restructuring and minimization phases, at the level of functional dependencies. Synthesis is performed through a branch-and-bound search, with analysis used to validated and rank potential solutions. In our current work, we have ex-

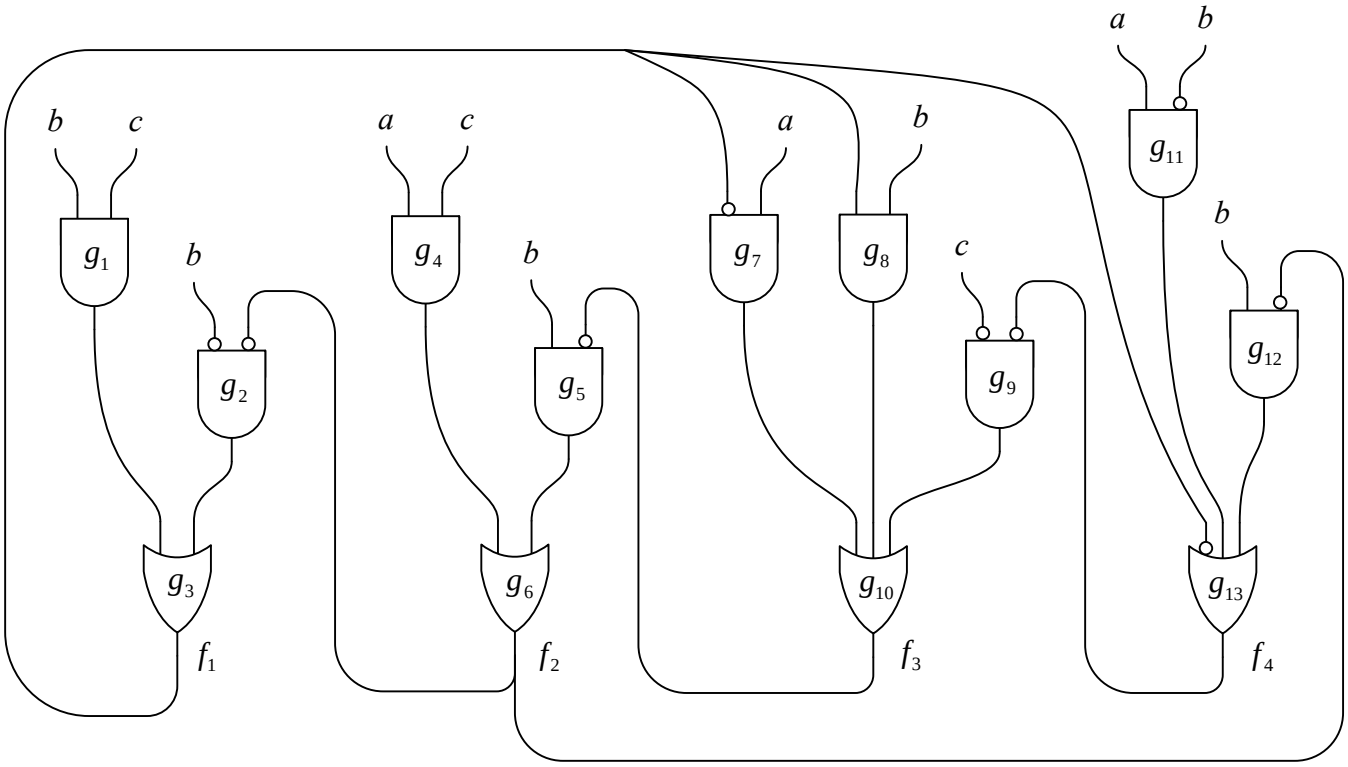


Fig. 5. A cyclic combinational circuit.

tended the methodology to the decomposition and mapping phases.

The case for using cycles to optimize area seems to be the most compelling. However, we have also investigated cyclic optimizations jointly targeting area and delay. In the branch-and-bound search, we use a sliding scale for the relative weight of area vs. delay when ranking solutions. The timing information is provided by the algorithm described in Section II. While this is a topic of ongoing research, we present some results.

For benchmark circuits, we used the usual suspects, namely the Espresso and LGSynth93 collections. Examples were selected based on size and suitability (generally, circuits with fewer than 30 inputs and fewer than 30 outputs). For circuits with latches, we extracted the combinational part.

Beginning from a collapsed specification, we applied the sequence of optimizations called "*script-.delay*" and then mapped to a library of two-input NAND/NOR gates and inverters. In the library

- NAND2/NOR2 gates have area 2, and inverters have area 1;
- NAND2/NOR2 gates have delay bounds 1, and inverters have delay bounds 0.5.

We compare the results obtained using the standard routines in SIS to those obtained with the corresponding routines from CYCLIFY. We chose a weighting of one-third for area and two-thirds for delay in the cyclic optimizations. Accordingly, the relative improvements in delay are more significant than those in area.

Figure 7 lists some of the benchmark circuits for which cyclic solutions were found. The area and delay of the SIS solutions are given in columns 2 and 3, respectively. The area and delay of the CYCLIFY solutions are given in columns 4 and 6, respectively. The improvements in area and delay, as percentages of the SIS solutions, are given in Columns 5 and 7, respectively.

We note that the improvements in delay were often greater than 10%. In some cases, there were simultaneous improvements in the vicinity of 10% for area and 25% for delay.

IV. Discussion

Early work in the 1960's and 70's established the premise of combinational circuits with cycles, and suggested the possible benefits. And yet, combinational circuits are not designed with cycles in practice. Perhaps designers have eschewed feedback due to the apparent complexity of reasoning about cyclic structures. Malik's work on the topic provided a solid foundation for analysis; however, he approached the

Espresso Benchmarks						
	SIS		CYCLIFY			
	Area	Delay	Area		Delay	
p82	175	19.0	167	4.6 %	15.0	21.1 %
t1	343	17.0	327	4.6 %	14.0	17.6 %
b4	474	30.0	464	2.1 %	29.0	3.4 %
exp	502	31.0	480	4.4 %	29.0	6.4 %
in3	599	40.0	593	1.0 %	33.0	17.5 %
in2	590	34.0	558	5.4 %	29.0	14.7 %
b10	681	37.0	691	-1.5 %	35.0	5.4 %
in0	751	42.0	777	-3.5 %	37.0	11.9 %
LGSynth93 Benchmarks						
	SIS		CYCLIFY			
	Area	Delay	Area		Delay	
5xp1	210	23.0	180	14.3 %	22.0	4.3 %
planet	964	40.0	938	2.7 %	38.0	5.0 %
s386	222	21.0	217	2.2 %	20.0	4.7 %
bw	280	28.0	254	9.3 %	20.5	26.8 %
cse	337	29.5	333	1.2 %	27.5	6.7 %
s510	452	28.0	444	1.8 %	24.0	14.3 %
ex1	526	40.0	522	0.7 %	34.0	15.0 %
s1	566	36.0	542	4.2 %	31.0	13.9 %
duke2	742	38.0	716	3.5 %	34.0	10.5 %
styr	821	39.0	827	-0.7 %	36.0	7.7 %
s1488	1016	43.0	995	2.1 %	34.0	20.9 %
s1494	1090	46.0	1079	1.0 %	39.0	15.2 %

Fig. 7. Area and Delay of Berkeley SIS vs. CYCLIFY for Benchmarks with "*script.delay*" optimizations, and mapping to NAND2/NOR2 gates and inverters.

problem of timing analysis from a topological perspective. Path-based reasoning about cyclic topologies seems tortuous and inefficient.

Exact timing analysis is, of course, a difficult problem, whether circuits are cyclic or not. However, using a functional approach based on event propagation, existing techniques can be applied effectively. In principle, timing analysis is no more difficult for cyclic circuits than for acyclic circuits. In practice, many further aspects need to be addressed. For instance, existing techniques for *incremental* timing analysis assume a topological ordering.

In future work, we will incorporate more realistic timing models into our analysis algorithm, and implement more sophisticated search heuristics in our synthesis procedure.

References

[1] R. I. Bahar et al., "Timing Analysis of Combinational Circuits Using ADD's," European Design Automation Conf., pp. 625–629, 1994.

[2] R. E. Bryant, "Graph-Based Algorithms For Boolean Function Manipulation," IEEE Trans. Computers, Vol. C-35, No. 6, pp. 677–691, 1986.

[3] R. E. Bryant, "Boolean Analysis of MOS Circuits," IEEE Trans. Computer-Aided Design, pp. 634–649, 1987.

[4] J. A. Brzozowski and C.-J. H. Seger, "Asynchronous Circuits," Springer-Verlag, 1995.

[5] S. A. Edwards, "Making Cyclic Circuits Acyclic," Design Automation Conf., pp. 159–162, 2003.

[6] D. A. Huffman, "Combinational Circuits with Feedback," Recent Developments in Switching Theory, A. Mukhopadhyay, ed., pp. 27–55, 1971.

[7] W. H. Kautz, "The Necessity of Closed Circuit Loops in Minimal Combinational Circuits," IEEE Trans. Comp., Vol. C-19, pp. 162–166, 1970.

[8] T. Larrabee, "Test Pattern Generation Using Boolean Satisfiability," IEEE Trans. Computer-Aided Design, Vol. 11, No. 1, pp. 4–15, 1992.

[9] C. Y. Lee, "Representation of Switching Circuits by Binary-Decision Programs," Bell System Technical Journal, Vol. 38, pp. 985–999, 1959

[10] S. Malik, "Analysis of Cyclic Combinational Circuits," IEEE Trans. Computer-Aided Design, Vol. 13, No. 7, pp. 950–956, 1994.

[11] C. R. McCaw, "Loops in Directed Combinational Switching Networks," Engineer's Thesis, Stanford University, 1963.

[12] M. Mendler and M. Fairlough, "Ternary Simulation: A Refinement of Binary Functions or an Abstraction of Real-Time Behavior, ", Workshop on Designing Correct Circuits, 1996.

[13] M. Riedel and J. Bruck, "Cyclic Combinational Circuits: Analysis for Synthesis," Int'l Workshop Logic and Synthesis, pp. 105–112, 2003.

[14] M. Riedel and J. Bruck, "The Synthesis of Cyclic Combinational Circuits," Design Automation Conf., pp. 163–168, 2003.

[15] M. Riedel, "Cyclic Combinational Circuits," Ph.D. Dissertation, Caltech, 2004.

[16] R. L. Rivest, "The Necessity of Feedback in Minimal Monotone Combinational Circuits," IEEE Trans. Comp., Vol. C-26, No. 6, pp. 606–607, 1977.

[17] E. Sentovich et al., "SIS: A System For Sequential Circuit Synthesis," Electronics Research Lab, U. C. Berkeley, Tech. Rep., UCB/ERL M92/41, 1992.

[18] T. R. Shiple, "Formal Analysis of Synchronous Circuits," Ph.D. Dissertation, U.C. Berkeley, 1996.

[19] R. A. Short, "A Theory of Relations Between Sequential and Combinational Realizations of Switching Functions," Ph.D. Dissertation, Stanford University, 1961.

[20] L. Stok, "False Loops Through Resource Sharing," Int'l Conf. Computer-Aided Design, pp. 345–348, 1992.

[21] H. Yalcin and J. Hayes, "Event Propagation Conditions in Circuit Delay Computation," ACM Trans. Design Automation of Electronic Systems, Vol. 2, No. 3, pp. 249—280, 1997

[22] M. Yoeli and S. Rinon, "Application of Ternary Algebra to the Study of Static Hazards," Journal of the ACM, Vol. 11, No. 1, pp. 84–97, 1964.