

# Cyclic Boolean Circuits<sup>☆</sup>

Marc D. Riedel

*Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455*

Jehoshua Bruck

*Electrical Engineering, California Institute of Technology, Pasadena, CA 91125*

---

## Abstract

A Boolean circuit is a collection of gates and wires that performs a mapping from Boolean inputs to Boolean outputs. The accepted wisdom is that such circuits must have *acyclic* (i.e., loop-free or feed-forward) topologies. In fact, the model is often defined this way – as a directed acyclic graph (DAG). And yet simple examples suggest that this is incorrect. We advocate that Boolean circuits should have *cyclic* topologies (i.e., loops or feedback paths). In other work, we demonstrated the practical implications of this view: digital circuits can be designed with fewer gates if they contain cycles. In this paper, we explore the theoretical underpinnings of the idea. We show that the complexity of implementing Boolean functions can be lower with cyclic topologies than with acyclic topologies. With examples, we show that certain Boolean functions can be implemented by cyclic circuits with as little as *one-half* the number gates that are required by equivalent acyclic circuits.

*Keywords:* Boolean circuits, Boolean functions, combinational circuits, cyclic circuits, DAG, cycles, loops, feedback

---

---

<sup>☆</sup>This work is partially supported by an NSF CAREER Award (grant CCF0845650), by the NSF Expeditions in Computing Program (grant CCF-0832824), by a grant from the MARCO Focus Center Research Program on Functional Engineered Nano-Architectonics (FENA), and by the Caltech Lee Center for Advanced Networking.

*Email addresses:* `mriedel@umn.edu` (Marc D. Riedel), `bruck@paradise.caltech.edu` (Jehoshua Bruck)

## 1. Introduction

This paper presents circuit constructs that are counter-intuitive and run against the accepted practice. Accordingly, throughout the paper, we adopt a discursive tone and elucidate the ideas with many examples.

### 1.1. Boolean Circuit Model

The **Boolean circuit** model is a fundamental one in complexity theory. It is also the core model underpinning practical digital circuit design. In general terms, a Boolean circuit consists of **gates** and **wires**. Each gate corresponds to a Boolean function that performs a mapping from several input bits to an output bit. The gates are connected by wires, with the outputs of some being the inputs of others. Accordingly, the topology of the circuit may be described as a directed graph  $G = (V, E)$ , with the nodes  $V$  representing logic gates and the edges  $E$  representing wires.

Central to the definition of the Boolean circuit model is the idea that such circuits perform mappings from designated inputs to designated outputs. So each output of a Boolean circuit computes a specific Boolean function:

$$f : \{0, 1\}^n \rightarrow \{0, 1\}. \tag{1}$$

Of course, a collection of gates and wires can produce outputs that are not Boolean functions. Such circuits can exhibit memory. We will use the term “Boolean circuit” to refer to a circuit that produces outputs that depend only on the current inputs; digital circuit designers call these **combinational circuits**. We will use the term “not a Boolean circuit” to refer to a circuit that produces outputs that depend not only on the current inputs, but also on the previous inputs; digital circuit designers call these **sequential circuits**.

The common way to categorize circuits is that Boolean circuits have acyclic topologies, while circuits with cyclic topologies have non-Boolean behavior. This conforms to intuition. In an acyclic circuit, such as that shown in Figure 1, the input values propagate forward and determine the values of the outputs. The outcome can be asserted regardless of the prior values on the wires and so independently of the past sequence of inputs. The circuit is clearly Boolean (it implements the exclusive-or of  $x$  and  $y$ .)

In a cyclic circuit, such as that shown in Figure 2, the behavior is more intricate. A common approach is to characterize the output values and the next state in terms of the input values and the current state. The current state, in turn, depends on the prior sequence of inputs (starting from some known initial state). The circuit in Figure 2 is not a Boolean circuit: given input values  $s = 0$  and  $r = 0$ , we cannot determine the output without knowing the value of  $z_3$  or  $z_4$ . (In fact, the circuit implements a one-bit memory element, called a **latch**.)

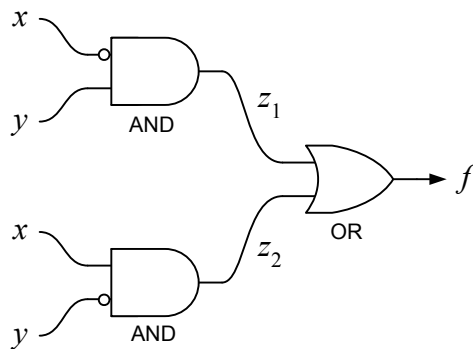


Figure 1: An acyclic circuit

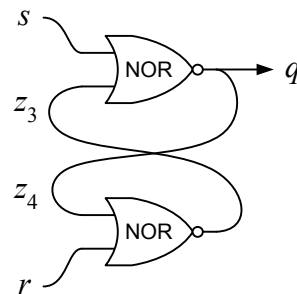


Figure 2: A cyclic circuit.

Clearly:

- An acyclic topology is *sufficient* for a circuit to be Boolean.
- Conversely, a cyclic topology is *necessary* for a circuit *not* to be Boolean.

But we ask:

- Is an acyclic topology *necessary* for a circuit to be Boolean?

The accepted wisdom is that, yes, Boolean circuits *must* have acyclic topologies. In fact, Boolean circuits are often defined as **directed acyclic graphs** (DAGs): e.g., (Papadimitriou, 1995, p. 80) and (Vollmer, 1999, p. 8). Circuit practitioners design combinational circuits without cycles: e.g., (Katz, 1992, p. 14) and (Wakerly, 2000, p. 193). Design automation tools enforce this constraint (Sentovich et al., 1992).

### 1.2. Cyclic Boolean Circuits

The accepted wisdom and theoretical practice are wrong: a circuit with a cyclic topology can be Boolean. A trivial circuit, shown in Figure 3, demonstrates this. It consists of an AND gate and an OR gate connected in a cycle, both with input  $x$ . Recall that the output of an AND gate is 0 iff either input is 0; the output of an OR gate is 1 iff either input is 1. Consider the two possible values of  $x$ .

- On the one hand, if  $x = 0$ , the output of the AND gate is fixed at 0, and so the input from the OR gate has no influence, as shown in Figure 4 (a).
- On the other hand, if  $x = 1$ , the output of the OR gate is fixed at 1, and so the input from the AND gate has no influence, as shown in Figure 4 (b).

Although useless, this circuit qualifies as Boolean: the value of the output  $f$  is completely determined by the current input value  $x$  (actually  $f = x$ ).

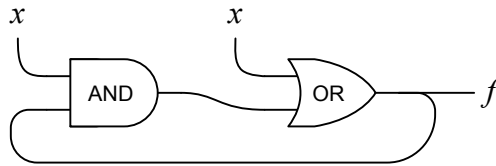


Figure 3: A (useless) cyclic Boolean circuit.

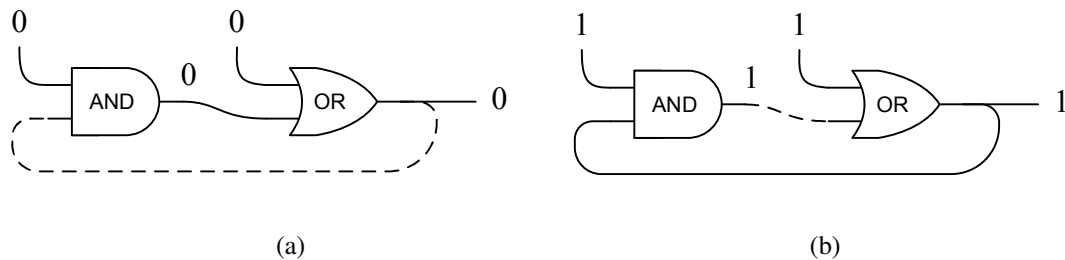


Figure 4: The circuit of Figure 3 with (a)  $x = 0$  and (b)  $x = 1$ .

One might argue that, while Boolean circuits can be cyclic, such examples are contrived or inconsequential. A convincing example suggesting otherwise is shown in

Figure 5. It consists of six alternating AND and OR gates, with inputs  $x_1, x_2$  and  $x_3$  repeated.

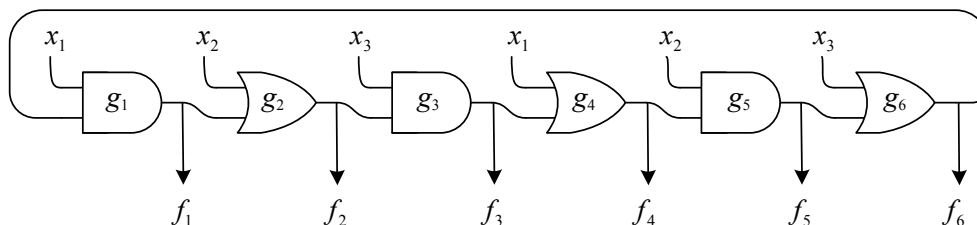


Figure 5: A cyclic Boolean circuit due to Rivest (Rivest, 1977).

To show that the circuit is Boolean, we label the feedback path with an unknown value  $y$ , as shown in Figure 6.<sup>1</sup> Applying standard Boolean identities, such as distribution and absorption, we obtain:

$$\begin{aligned}
 f_1 &= x_1 y \\
 f_2 &= x_2 + f_1 = x_2 + x_1 y \\
 f_3 &= x_3 f_2 = x_3(x_2 + x_1 y) \\
 f_4 &= x_1 + f_3 = x_1 + x_3(x_2 + x_1 y) = x_1 + x_2 x_3 \\
 f_5 &= x_2 f_4 = x_2(x_1 + x_2 x_3) = x_2(x_1 + x_3) \\
 f_6 &= x_3 + f_5 = x_3 + x_2(x_1 + x_3) = x_3 + x_1 x_2.
 \end{aligned}$$

We see that  $f_4$ ,  $f_5$  and  $f_6$  do not depend upon the unknown value. Thus, we compute

$$\begin{aligned}
 f_1 &= x_1 f_6 = x_1(x_3 + x_1 x_2) = x_1(x_2 + x_3) \\
 f_2 &= x_2 + f_1 = x_2 + x_1(x_2 + x_3) = x_2 + x_1 x_3 \\
 f_3 &= x_3 f_2 = x_3(x_2 + x_1 x_3) = x_3(x_1 + x_2).
 \end{aligned}$$

Each output depends on the current input values, not on the prior values, and so the circuit is Boolean. Unlike the circuit in Figure 3, the one in Figure 5 computes something useful. The six output functions are distinct, and each depends on all three input

<sup>1</sup>We use the standard arithmetical convention that addition represents disjunction (OR) and multiplication represents conjunction (AND). Also, when writing arithmetical expressions, we assume that multiplication has precedence over addition, so  $x + yz$  represents  $x + (yz)$ .

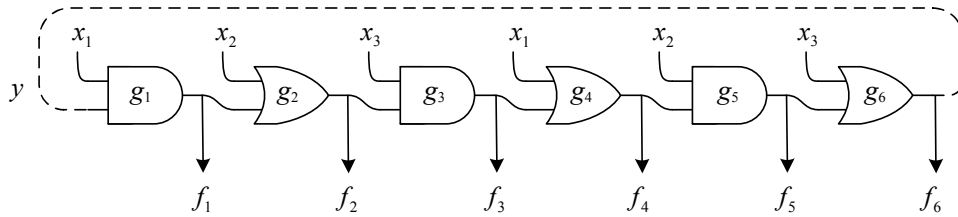


Figure 6: Analyzing the circuit of Figure 5.

variables. Moreover, we can show that this cyclic circuit has *fewer* gates than any equivalent acyclic circuit. To see this, note that any acyclic configuration contains at least one gate producing an output function that does not depend on the output of any other gate producing an output function. (If this were not the case, then every output gate would depend upon another and so the circuit would be cyclic.)

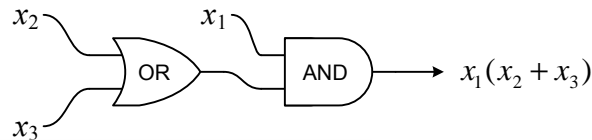


Figure 7: With fan-in two gates, two gates are needed to compute  $x_1(x_2 + x_3)$ .

With fan-in two gates, it takes two gates to compute any one of the six functions by itself. This is illustrated in Figure 7. We conclude that an acyclic implementation of the six functions requires seven gates, compared to the six in the cyclic circuit.

The circuit in Figure 5 was presented by Rivest in 1977, in a paper less than a page long (Rivest, 1977). His work on the topic, as well as that of a few others in the 1960s, seems to have gone largely unnoticed by theoreticians and practitioners alike. And yet his example hints at a fundamental misconception in the field, namely that “Boolean circuit” and “acyclic” are synonymous terms. In this paper, we demonstrate not only that it is feasible to design Boolean circuits with cyclic topologies, but it is generally advantageous to do so.

Given an upper bound on the time that it takes each gate to compute a value, the **delay** of a circuit is an upper bound on the time that it takes for the circuit to compute the values of the output functions. The delay is generally determined by the length of the longest path from an input to an output, called the **critical path**. In any circuit – cyclic or acyclic – there may exist **false paths**, that is to say, topological paths that are never sensitized. If the longest path is false, it does not impinge upon the delay. The critical path of the circuit then is the longest true path.

For a cyclic circuit, we can say that it is Boolean if all *cycles are false*; the sensitized paths in the circuit never bite their own tail to form true cycles. The delay of a cyclic circuit is not necessarily greater than that of an acyclic circuit. It is simply the length of the longest true path.

### 1.3. Prior and Related Work

In 1963, McCaw presented a thesis for his Engineer’s Degree at Stanford titled “Loops in Directed Combinational Switching Networks” (McCaw, 1963). He demonstrated the cyclic circuit shown in Figure 8 consisting of two AND gates and two OR gates, with five inputs and two outputs. His analysis of the circuit was in the same vein as that given in the previous section. He concluded that the circuit is Boolean and computes the functions:

$$\begin{aligned} f_1 &= a + b + x(c + d + \bar{x}f_1) = a + b + x(c + d) \\ f_2 &= c + d + \bar{x}(a + b + xf_2) = c + d + \bar{x}(a + b). \end{aligned}$$

As with circuit in Figure 5, McCaw argues that his circuit has fewer AND and OR gates than is possible with an acyclic circuit implementing the same functions.

Even earlier, in 1960, Short applied an abstract graphical model to the study of switching networks with directional switches (Short, 1961). He argued that networks must be cyclic for some minimal forms. In 1970, Kautz, Short’s Ph.D. advisor at Stanford, presented a short paper on the topic of feedback in circuits with logic gates (Kautz, 1970). He described a cyclic circuit consisting of 6 fan-in two NOR gates with three inputs and three outputs. Although plausible, his circuit is not Boolean according to the rigorous model that we propose. (It assumes that all wires have definite Boolean values at the outset.)

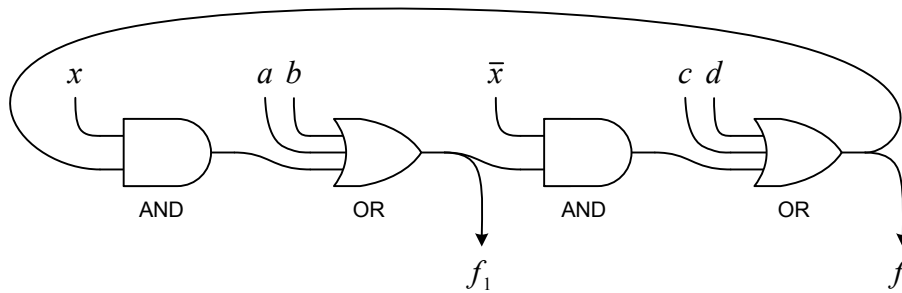


Figure 8: A cyclic Boolean circuit due to McCaw.

In 1971, Huffman discussed feedback in linear threshold networks. He claimed that an arbitrarily large number of input variables can be complemented in a network containing a single NOT element, provided that feedback is used (Huffman, 1971). This improved upon an earlier result by Markov, demonstrating that  $k$  NOT elements suffice to generate the complements of  $2^k - 1$  variables (Markov, 1958). As with Kautz's example, Huffman's doesn't quite work. Still, in an insightful commentary on his and Kautz's work, he hinted at the possible implications,

*“At this time, these [cyclic] examples are isolated ones. They do, however, provide tantalizing glimpses into an imaginable area of future research.”*

In 1977, Rivest presented a general version of the circuit in Figure 5 and proved that it is optimal (Rivest, 1977). For any odd integer  $n$  greater than 1, the general circuit consists of  $n$  fan-in two AND gates alternating with  $n$  fan-in two OR gates, with input variables  $x_1, \dots, x_n$  arranged as in Figure 5. It produces  $2n$  distinct output functions, each of which depends on all  $n$  input variables. He proved that any acyclic circuit implementing the same  $2n$  output functions requires at least  $3n - 2$  fan-in two gates. Thus, asymptotically, his cyclic implementation is at most two-thirds the size of the best possible acyclic implementation.

In 1978, Khrapchenko was the first to recognize that depth and delay in a circuit are not equivalent concepts (Khrapchenko, 1978). With false paths, the delay of a circuit can be less than that of the longest path.



In more recent work, we advocated the design of cyclic circuits in practice. We discussed techniques for validating cyclic circuits (Riedel and Bruck, 2003a, 2004; Backes et al., 2008). We proposed strategies for synthesizing cyclic circuits with computer-aided design tools (Riedel and Bruck, 2003b; Backes and Riedel, 2009). Our synthesis tool, CYCLIFY, optimizes the designs of circuits by introducing cycles in the restructuring and minimization phases of logic synthesis. It reduces the number of gates needed by overlapping the computation of multiple functions (as Rivest’s example hinted at). It reduces the delay of circuits by moving computation off of critical paths (as Khrapchenko’s work hinted at).

#### *1.4. Overview*

In this paper, we study cyclic Boolean circuits from a theoretical perspective: we exhibit families of cyclic circuits that are optimal in the number of gates and we prove lower bounds on the size of equivalent acyclic circuits. First, we present a cyclic circuit with only **two** gates and argue that an equivalent acyclic circuit requires **three** gates. Next, we present a cyclic circuit with **three** gates and argue that an equivalent acyclic circuit requires **five** gates. Finally, we present a family of cyclic circuits with  $n$  gates and prove that an equivalent family of acyclic circuits requires at least  $2n$  gates. Accordingly, we conclude that the cyclic complexity of functions can be as little as one-half of the acyclic complexity. Our lower bound is based on a fan-in argument: in order to compute a function that depends on a certain number of variables using gates with a certain fan-in, we require a tree of at least a certain size. We show that this is largest gap that can be obtained with this fan-in lower bound technique.

## 2. Framework

### 2.1. Notation

Algebraically, we use the standard notation: **addition** (+) denotes disjunction (OR), **multiplication** ( $\cdot$ ), denotes conjunction (AND), and an **overbar** ( $\bar{x}$ ) denotes negation (NOT). The **restriction** operation (also known as the cofactor) of a function  $f$  with respect to a variable  $x$  refers to the assignment of a constant value to  $x$  in the expression for  $f$ . We use the notation  $f|_x$  to denote the restriction of  $x$  to 1 and  $f|_{\bar{x}}$  to denote the restriction of  $x$  to 0. A function  $f$  **depends** upon a variable  $x$  iff  $f|_x$  is not identically equal to  $f|_{\bar{x}}$ . Call the variables that a function depends upon its **support set**.

We will often represent functions in so called *Reed-Muller* form, a canonical form consisting of exclusive-of (XOR) and conjunction (AND) operations. We denote XOR with the symbol  $\oplus$ . As early as 1929, Zhegalkin showed that this representation is canonical (Zhegalkin, 1929): if we multiply out all parentheses, perform the simplifications  $x \oplus x = 0$  and  $x \cdot x = x$ , and sort the product terms, the resulting expression is unique. For instance, the function

$$f = \bar{x}_1(\bar{x}_2 + \bar{x}_3).$$

is represented as

$$f = 1 \oplus x_1 \oplus x_2x_3 \oplus x_1x_2x_3.$$

Note that  $1 \oplus x = \bar{x}$ . When writing arithmetical expressions, we assume that AND has precedence over XOR, so  $x \oplus yz$  represents  $x \oplus (yz)$ . The Reed-Muller representation has distinct advantages when manipulating expressions algebraically. In particular, the dependence of a function on its variables is explicit: if a variable appears in the expression, then the function depends on it.

### 2.2. Circuit Model

Graphically, a **circuit** consists of **gates** connected by **wires**. Each gate has one or more inputs and a single output. The symbols for common gates are shown in Figure 9. A bubble is used to indicate that an input or output is negated, as illustrated in Figure 10.

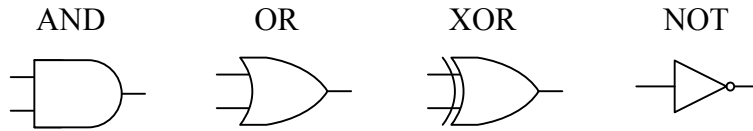


Figure 9: Symbols for different types of gates.

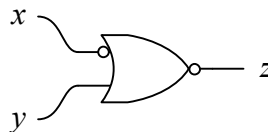


Figure 10: Bubbles on the inputs or the output of a gate indicate negation. Here  $z = \overline{\bar{x} + y}$ .

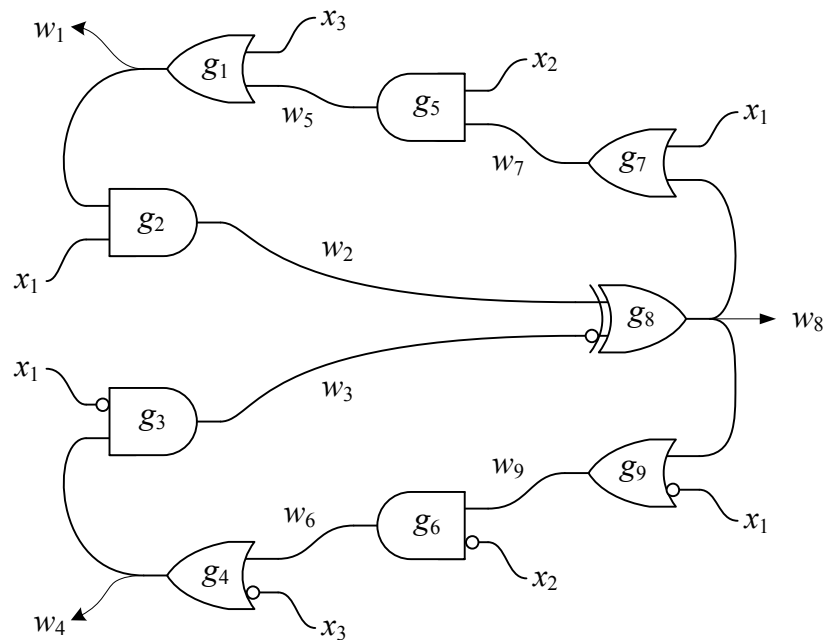
An example of a circuit is shown in Figure 11. Even though a wire may split in our diagrams, as is the case with wire  $w_8$  in Figure 11, conceptually there is a single instance of it. In general,

- A circuit accepts signals  $x_1, \dots, x_m$ , ranging over  $\{0, 1\}$ , called the **primary inputs**. Each primary input is fed into one or more gate inputs. Even though the symbol for a primary input may appear in several places, as is the case with  $x_1$ ,  $x_2$  and  $x_3$  in Figure 11, conceptually there is a single instance of it.
- The gates in the circuit produce **internal signals**,  $w_1, \dots, w_n$  ranging over  $\{0, 1, \perp\}$ . (We discuss the third value,  $\perp$ , in Section 2.4.)
- A subset of the set of internal signals is designated as the set of **primary outputs**.

A gate implements a Boolean function, i.e., a mapping from Boolean inputs to a Boolean output value,

$$g : \{0, 1\}^d \rightarrow \{0, 1\}.$$

The set of inputs to a gate are called its **fan-in** set. When we say a “fan-in  $d$ ” gate, we mean a gate with fan-in set of cardinality  $d$ . The set of gates that are attached to a gate output are called its **fan-out** set.



input signals:  $x_1, x_2, x_3$   
 internal signals:  $w_1, \dots, w_9$   
 output signals:  $w_1, w_4, w_8$   
 gates:  $g_1, \dots, g_9$

Figure 11: An example of a circuit, consisting of gates and wires.

### 2.3. Gate Model

Any discussion of circuit complexity is predicated on a restricted gate model of some kind. Otherwise, with gates of arbitrary size and complexity, any function can be implemented with a single “gate.” We restrict the scope of gates in two ways. The first way is to bound the fan-in, as shown in Figure 12 (a). Each gate can have at most  $d$  inputs, for some integer  $d \geq 1$ . The second way is to restrict the type of gate. For instance, we can limit ourselves to so-called “AND-OR-NOT” (AON) gates, i.e., AND gates with the inputs and output possibly negated. An example of such a gate is shown in Figure 12 (b). The general form of the Boolean function realized by an AON gate is

$$g(x_1, x_2, \dots, x_d) = (x_1 \oplus c_1) \cdot (x_2 \oplus c_2) \cdots (x_d \oplus c_d) \oplus c_{d+1},$$

where  $c_1, \dots, c_{d+1}$  are arbitrary choices of 0 and 1.

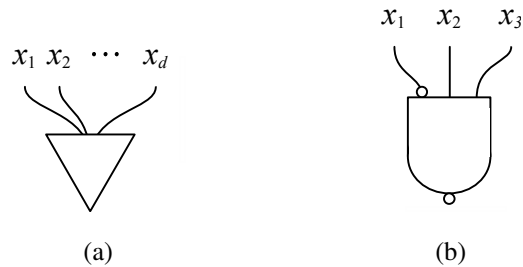


Figure 12: Restricting the scope of gates. (a) Bound the fan-in. (b) Use AND gates (with the inputs and output possibly negated).

### 2.4. Boolean Circuits

Analysis of an acyclic circuit is transparent. We first evaluate the gates connected only to primary inputs, and then gates connected to these and primary inputs, and so on, until we have evaluated all gates. The previous values of the internal signals do not enter into play.

In a cyclic circuit, there are one or more **strongly connected components**. Recall that in a directed graph  $G$ , a strongly connected component is an induced subgraph  $S \subseteq G$  such that

- there exists a directed path between every pair of nodes in  $S$ ;
- for every node  $s$  in  $S$  and every node  $n$  outside of  $S$ , if there exists a path from  $s$  to  $n$  (from  $n$  to  $s$ ) then there is no path from  $n$  to  $s$  (from  $s$  to  $n$ , respectively).

We can analyze each strongly connected component separately.

We adopt a **ternary framework** for analysis. We assume that, at the outset, all wires in a circuit have *unknown* or possibly *undefined* values, which we denote with the symbol  $\perp$ . We apply definite values to the inputs, and track the propagation of signal values.

With only the primary inputs fixed at definite values, each gate in a strongly connected component has some unknown/undefined inputs, valued  $\perp$ . Nevertheless, for each such gate we can ask: is there sufficient information to conclude that the gate output is 0 or 1, in spite of the  $\perp$  values? If yes, we assign this value as the output; otherwise, the value  $\perp$  persists. For instance, with an AND gate, if the inputs include a 0, then the output is 0, regardless of other  $\perp$  inputs. If the inputs consist of 1 and  $\perp$  values, then the output is  $\perp$ . Only if all the inputs are 1 is the output 1. This is illustrated in Figure 13. Input values that determine the gate output are called **controlling**.

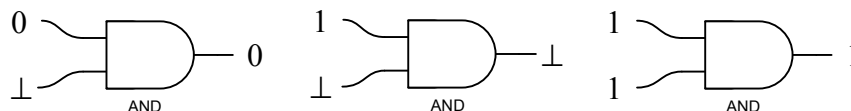


Figure 13: An AND gate with 0, 1, and  $\perp$  inputs.

Consider the circuit fragment in Figure 14. 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. And yet, by definition,  $\perp$  designates an unknown, possibly undefined value. (For instance, in an actual circuit, it could indicate a voltage value exactly half way between logical 0 and logical 1.) In our analysis, we remain agnostic: the output of the OR gate is  $\perp$ .

In the analysis, we track the propagation of well-defined signal values. Once a

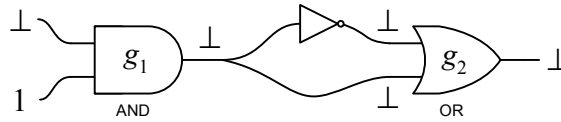


Figure 14: An illustration unknown/undefined values  $\perp$ .

definite value is assigned to an internal wire, this value persists for the duration (so long as the input values are held constant). For any input assignment, a circuit reaches a so-called **fixed point** in the ternary framework: a state where no further updates of controlling values are possible. This fixed point is unique (Brzozowski and Seger, 1995). We adopt the following definition.

**A circuit is Boolean** iff, for every assignment of input values, with all the wires initially set to  $\perp$ , the circuit reaches a fixed point that does not contain any  $\perp$  values on the outputs.

We illustrate the analysis with two cyclic examples: one that is not Boolean and one that is.

### Example 1

Consider the circuit shown in Figure 15, consisting of an AND gate  $g_1$ , an OR gate  $g_2$ , and an AND gate  $g_3$ , in a cycle. By inspection, note that if  $x_1 = 0$  then  $f_1$  assumes value 0; if  $x_2 = 1$  then  $f_2$  assumes value 1; and if  $x_3 = 0$  then  $f_3$  assumes value 0. But what happens if  $x_1 = 1$ ,  $x_2 = 0$  and  $x_3 = 1$ ? In this case, all the outputs equal  $\perp$ , as illustrated in Figure 16. The outcome for all eight cases is shown in Figure 1. We conclude that the circuit is not Boolean.

### Example 2

Consider the circuit in Figure 17. Suppose that we apply inputs  $x_1 = 1$ ,  $x_2 = 0$ ,  $x_3 = 1$ . Gates  $g_1$ ,  $g_3$ ,  $g_5$  and  $g_7$  produce outputs of 1, 0, 0, and 1, respectively. Gate  $g_2$  produces an output of 1. Gate  $g_8$  produces an output of 0. Gate  $g_9$  produces an output of 0. Gate  $g_6$  produces an output of 0. Finally, gate  $g_4$  produces an output of 0. These values are shown in Figure 17. The analysis for all eight input combinations is summarized in Table 18. We conclude that the circuit is Boolean.

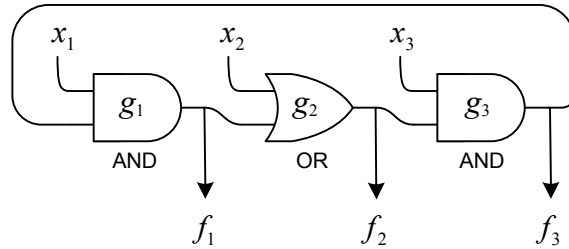


Figure 15: A cyclic circuit that is not Boolean.

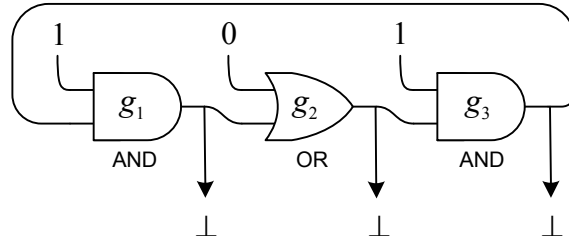


Figure 16: The circuit of Figure 15 with  $x_1 = 1$ ,  $x_2 = 0$  and  $x_3 = 1$ .

$x_1$	$x_2$	$x_3$	$f_1$	$f_2$	$f_3$
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	0	0	0
1	0	1	⊥	⊥	⊥
1	1	0	0	1	0
1	1	1	1	1	1

Table 1: Analysis of the circuit in Figure 15.



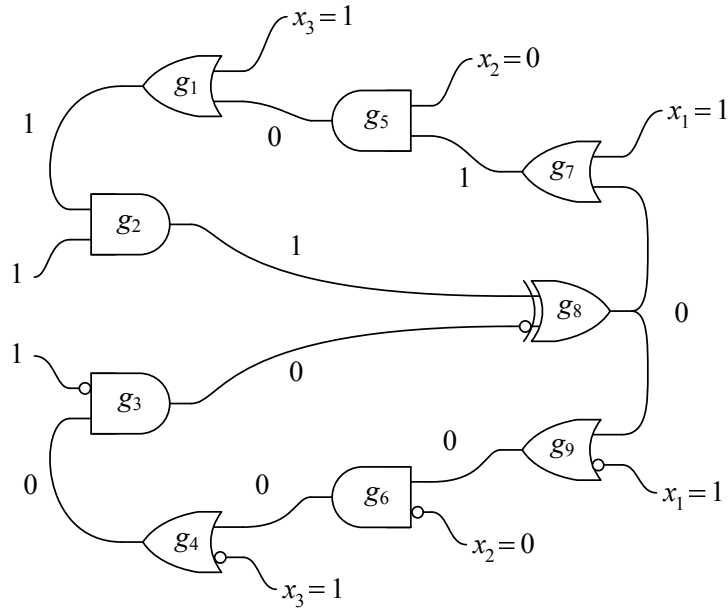


Figure 17: The circuit of Figure 11 with inputs  $x_1 = 1, x_2 = 0, x_3 = 1$ .

$x_1$	$x_2$	$x_3$	$g_1$	$g_2$	$g_3$	$g_4$	$g_5$	$g_6$	$g_7$	$g_8$	$g_9$
0	0	0	0	0	1	1	0	1	0	0	1
0	0	1	1	0	1	1	0	1	0	0	1
0	1	0	0	0	1	1	0	0	0	0	1
0	1	1	1	0	0	0	1	0	1	1	1
1	0	0	0	0	0	1	0	1	1	1	1
1	0	1	1	1	0	0	0	0	1	0	0
1	1	0	1	1	0	1	1	0	1	0	0
1	1	1	1	1	0	0	1	0	1	0	0

Figure 18: Analysis summary for the circuit of Figure 11.

### 3. Optimality and Lower Bounds

Our general strategy in the following constructions is to present a cyclic circuit that is optimal in the number of gates, and then prove a lower bound on the size of any acyclic circuit implementing the same functions. (For what follows, when we say **circuit size**, we mean the number of gates.) The argument for the optimality of the cyclic circuit rests on two properties:

#### Property 1

*Each of the output functions depends on all the variables.*

#### Property 2

*The output functions are pairwise distinct.*

The cyclic circuit is shown to be optimal according to the following trivial claim (true *regardless* of the gate model):

#### Claim 1

*A circuit implementing  $m$  distinct functions consists of at least  $m$  gates.*

Lower bounds on circuit size are notoriously difficult to establish. In fact, such proofs are related to fundamental questions in computer science, such as the separation of the  $P$  and  $NP$  complexity classes. (To prove that  $P \neq NP$  it would suffice to find a class of problems in  $NP$  that cannot be computed by a polynomially sized circuit.) Much of the recent work in circuit complexity has been spurred by these open problems.

All existing lower bounds on circuit size are linear in the number of variables (Vollmer, 1999). In 1949, Shannon showed by a straight forward counting argument that nearly *all* functions require circuits with an exponential number of gates (Shannon, 1949). Yet there is no known *explicit* example (Wegener, 1987).

We prove that some cyclic Boolean circuits are smaller than equivalent acyclic circuits based on a fan-in lower bound. We show that this is largest gap that we can prove using the fan-in lower bound technique.

### 3.1. Fan-in Lower Bound

Our lower bound on the size of an acyclic circuit is formulated as a fan-in argument. The essence of the argument was presented by Rivest (Rivest, 1977), although we present it in a more general form.

A circuit can only compute a function of a given set of input variables if it “sees” all of them. For example, in Figure 19, gate  $g_2$  can compute a function of  $x_1, x_2$  and  $x_3$ ; gate  $g_1$  cannot compute a function of  $x_3$  since it does not see  $x_3$ . In an acyclic

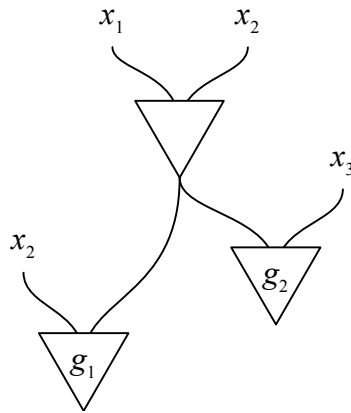


Figure 19: A gate can only compute functions of variables that it “sees”.

circuit, there is a partial ordering among the gates: if a gate  $g_i$  depends on a gate  $g_j$ , directly or indirectly, then  $g_j$  cannot depend on  $g_i$ , directly or indirectly. With a partial ordering on the output functions, there must be at least one output function at the top which depends upon no other. If this function depends on  $v$  input variables, the gate producing it must be the root of a tree that sees all these  $v$  variables as leaves. The lower bound is based on a calculation of the minimum number of gates in this tree.

#### Claim 2

An acyclic circuit implementing  $m$  distinct output functions, each depending on  $v$  input variables, consisting of gates with fan-in at most  $d$  has at least

$$\left\lceil \frac{v-1}{d-1} \right\rceil + m - 1$$

gates. This is true for any fan-in value  $d \geq 2$ .

**Proof:** Consider a connected directed acyclic graph (DAG). Call nodes with no incoming edges **leaves**, and all other nodes **internal** nodes. We show, by a simple inductive argument, that a connected DAG with  $k$  internal nodes, each with in-degree at most  $d$ , has at most  $k(d - 1) + 1$  leaves. Obviously, a graph consisting of a single such internal node has at most  $d$  leaves. Suppose an internal node with in-degree at most  $d$  is added to a connected DAG. If the resulting graph is to be a *connected* DAG, the new node can replace an existing leaf or it can be attached to an existing internal node. The former case is illustrated with node  $g_1$  in Figure 20, and the latter with node  $g_2$ . In both cases there is a net gain of at most  $d - 1$  leaves. We conclude that connected DAG with  $k$  internal nodes has at most

$$\begin{aligned} & d + (k - 1)(d - 1) \\ = & k(d - 1) + 1 \end{aligned}$$

leaves, as expected. Suppose that a connected DAG has  $v$  leaves. Since

$$v \leq k(d - 1) + 1,$$

the number of internal nodes  $k$  is bounded by

$$k \geq \left\lceil \frac{v - 1}{d - 1} \right\rceil.$$

Now, in an acyclic circuit implementing  $m$  output functions, at least one of the output functions depends on no other. By the argument above, this output function requires at least

$$\left\lceil \frac{v - 1}{d - 1} \right\rceil$$

gates. With *distinct* output functions, each output function must emanate from a different gate, so at least  $m - 1$  gates are required to implement the remaining  $m - 1$  functions.

□

### 3.2. Improvement Factor

Suppose that we have a cyclic circuit with  $m$  gates, each with fan-in at most  $d$ , that implements  $m$  distinct functions, each of which depends on all  $v$  input variables.

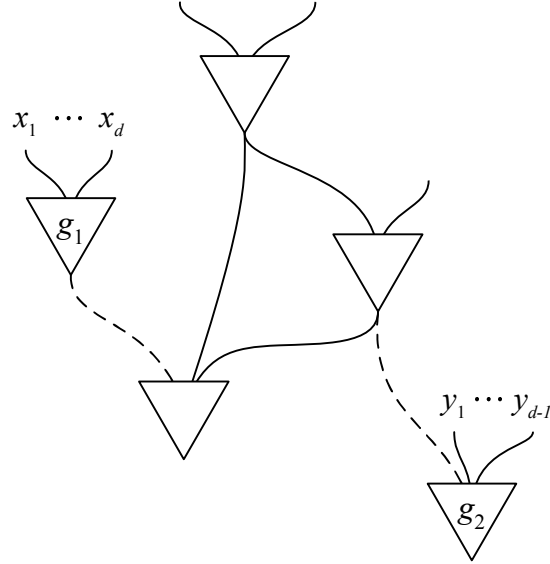


Figure 20: Adding a node with in-degree  $d$  to a connected DAG results in net gain of at most  $d - 1$  leaves.

Call the **improvement factor** the ratio of size of the cyclic circuit to the lower bound on the size of the acyclic circuit:

$$\frac{\text{size of cyclic}}{\text{size of acyclic}} = \frac{m}{\left\lceil \frac{v-1}{d-1} \right\rceil + m - 1}.$$

With an improvement factor of  $C$ , we can say that our cyclic circuit is  $C$  times the size of any equivalent acyclic circuit.

**Claim 3**

*The improvement factor is bounded below by  $\frac{1}{2}$ .*

**Proof:** For a given  $d$ , the improvement factor is minimized if the term

$$\frac{v-1}{d-1}$$

in the denominator is maximized. Now, the number of variables  $v$  in a cyclic circuit is at most  $m(d - 1)$ , and this is achieved if all the gates have fan-in  $d$ . For such a circuit,

the improvement factor is

$$\frac{m}{\left\lceil m - \frac{1}{d-1} \right\rceil + m - 1} = \frac{m}{2m-1} \geq \frac{1}{2}$$

if  $d \geq 3$  and

$$\frac{m}{2m-2} \geq \frac{1}{2}$$

if  $d = 2$ . □

### 3.3. Rivest's Circuit

Consider the circuit shown in Figure 21, due to Rivest (Rivest, 1977). This is the same circuit as in Figure 5, shown again for the reader's convenience; however, here we provide a slightly different analysis of it, based on controlling values. We first verify

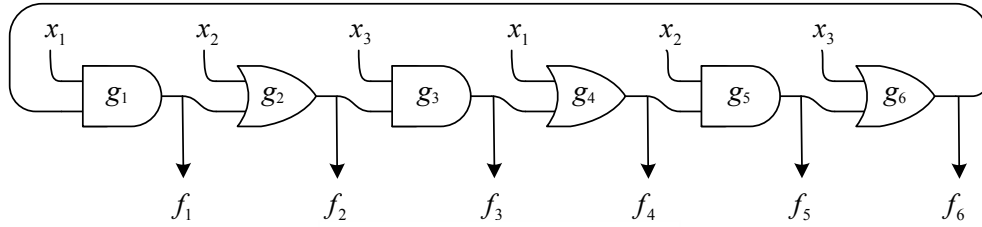


Figure 21: A cyclic Boolean circuit with 3 inputs, due to Rivest (Rivest, 1977).

that this circuit is Boolean. For gate  $g_1$ , an AND gate,  $x_1 = 0$  is a controlling value. Setting  $x_1 = 0$  we have

$$\begin{aligned} f_1 |_{\bar{x}_1} &= 0, \\ f_2 |_{\bar{x}_1} &= (f_1 |_{\bar{x}_1}) + x_2 = x_2, \\ f_3 |_{\bar{x}_1} &= (f_2 |_{\bar{x}_1}) x_3 = x_2 x_3, \\ f_4 |_{\bar{x}_1} &= (f_3 |_{\bar{x}_1}) + 0 = x_2 x_3, \\ f_5 |_{\bar{x}_1} &= (f_4 |_{\bar{x}_1}) x_2 = x_2 x_3, \\ f_6 |_{\bar{x}_1} &= (f_5 |_{\bar{x}_1}) + x_3 = x_3. \end{aligned}$$

All outputs assume definite Boolean values. For gate  $g_4$ , an OR gate,  $x_1 = 1$  is a controlling value. Setting  $x_1 = 1$ , we have

$$\begin{aligned}
f_4|_{x_1} &= 1, \\
f_5|_{x_1} &= (f_4|_{x_1}) x_2 = x_2, \\
f_6|_{x_1} &= (f_5|_{x_1}) + x_3 = x_2 + x_3, \\
f_1|_{x_1} &= (f_6|_{x_1}) 1 = x_2 + x_3, \\
f_2|_{x_1} &= (f_1|_{x_1}) + x_2 = x_2 + x_3, \\
f_3|_{x_1} &= (f_2|_{x_1}) x_3 = x_3.
\end{aligned}$$

Again, all outputs assume definite Boolean values. Since  $x_1$  must either have value 0 or value 1, we conclude that the network is Boolean. We assemble the output functions from these two cases. Applying distributivity and absorption, we obtain

$$\begin{aligned}
f_1 &= \bar{x}_1 \cdot f_1|_{\bar{x}_1} + x_1 \cdot f_1|_{x_1} = \bar{x}_1 \cdot 0 + x_1 \cdot (x_2 + x_3) = x_1(x_2 + x_3) \\
f_2 &= \bar{x}_1 \cdot f_2|_{\bar{x}_1} + x_1 \cdot f_2|_{x_1} = \bar{x}_1 \cdot x_2 + x_1 \cdot (x_2 + x_3) = x_2 + x_1 x_3 \\
f_3 &= \bar{x}_1 \cdot f_3|_{\bar{x}_1} + x_1 \cdot f_3|_{x_1} = \bar{x}_1 \cdot x_2 x_3 + x_1 \cdot x_3 = x_3(x_1 + x_2) \\
f_4 &= \bar{x}_1 \cdot f_4|_{\bar{x}_1} + x_1 \cdot f_4|_{x_1} = \bar{x}_1 \cdot x_2 x_3 + x_1 \cdot 1 = x_1 + x_2 x_3 \\
f_5 &= \bar{x}_1 \cdot f_5|_{\bar{x}_1} + x_1 \cdot f_5|_{x_1} = \bar{x}_1 \cdot x_2 x_3 + x_1 \cdot x_2 = x_2(x_1 + x_3) \\
f_6 &= \bar{x}_1 \cdot f_6|_{\bar{x}_1} + x_1 \cdot f_6|_{x_1} = \bar{x}_1 \cdot x_3 + x_1 \cdot (x_2 + x_3) = x_3 + x_1 x_2.
\end{aligned}$$

Rivest presented a more general version of this circuit. For any odd integer  $n$  greater than 1, the general circuit consists of  $n$  two-input AND gates alternating with  $n$  two-input OR gates in a single cycle, with inputs  $x_1, \dots, x_n$  repeated, as shown in Figure 22. Analyzing the general circuit in the same manner as above, we find that it implements

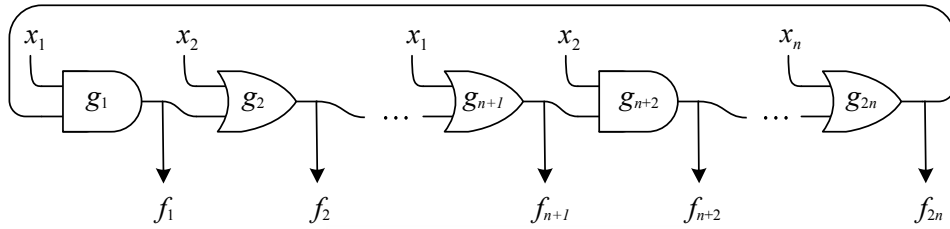


Figure 22: A cyclic Boolean circuit with  $n$  inputs (for any odd  $n \geq 3$ ) due to Rivest.

the functions

$$\begin{aligned}
 f_1 &= x_1(x_n + x_{n-1}(\cdots(x_3 + x_2)\cdots)) \\
 f_2 &= x_2 + x_1(x_n + \cdots(x_4x_3)\cdots) \\
 &\vdots \\
 f_{2n} &= x_n + x_{n-1}(x_{n-2} + \cdots(x_2x_1)\cdots).
 \end{aligned}$$

Note that the functions are symmetrical with respect to a cyclic permutation of the variables. More precisely, the functions  $f_3, f_5, \dots, f_{2n-1}$  are obtained from  $f_1$  by the cyclic permutation of the variables  $x_1, x_2, \dots, x_n$ :

$$\begin{aligned}
 f_3(x_1, x_2, \dots, x_n) &= f_1(x_2, x_3, \dots, x_1), \\
 f_5(x_1, x_2, \dots, x_n) &= f_1(x_3, x_4, \dots, x_2), \\
 &\vdots \\
 f_{2n-1}(x_1, x_2, \dots, x_n) &= f_1(x_n, x_1, \dots, x_{n-1}).
 \end{aligned}$$

Similarly, the functions  $f_4, f_6, \dots, f_{2n}$  are obtained from  $f_2$  through a cyclic permutation of the variables.

### 3.3.1. Optimality

To show that Rivest's circuit is optimal, we must show that it satisfies Properties 1 and 2.

1. To show that each function depends on all  $n$  input variables, we note that in the parenthesized expression, each variable appears exactly once. Without loss of generality, consider the  $i$ -th function  $f_i$  in the list, for an odd  $i$ , and consider the  $j$ -th variable appearing in its expression, from the left-hand side. To show the dependence on this variable, set each variable preceding a product to 1, and each variable preceding a sum to zero, beginning on the left-hand side, until we arrive at  $x_j$ . Set the variable following  $x_j$  to 1 and all variables following that to 0. The result is

$$f_i = 1(0 + 1(0 + \cdots + x_j(1 + 0(0 + 0(\cdots)))))) = x_j.$$



2. To show that all the functions are distinct, we exhibit an assignment that sets any chosen function to 0 if it is odd-numbered (to 1 if it is even-numbered), while setting all the other functions to 1 (to 0, respectively). Without loss of generality, consider function  $f_i$ , for an odd  $i \leq n$ . This function is the output of an AND gate with input  $x_i$ . Set  $x_i$  to 0 and set all the other the variables to 1. Clearly,  $f_i$  has value 0 while all the other functions have value 1 in this case.

(Rivest stated these conditions without proof.)

### 3.3.2. Acyclic Lower Bound

Note that the Rivest circuit has  $n$  input variables and implements  $2n$  distinct output functions with  $2n$  fan-in 2 gates. According to Claim 2, an acyclic circuit implementing the same functions requires at least

$$\left\lceil \frac{n-1}{2-1} \right\rceil + 2n - 1 = 3n - 2$$

fan-in 2 gates. For large  $n$ , the improvement factor is

$$\frac{\text{size of cyclic}}{\text{size of acyclic}} = \frac{2n}{3n-2} \approx \frac{2}{3}.$$

Rivest's cyclic circuit is two-thirds the size of any acyclic circuit implementing the same functions.

Given a circuit with a single cycle, we can always obtain a corresponding acyclic version by breaking the feedback and doubling the length of the chain, as shown in Figure 23. (The input  $\perp$  indicates any constant value.)

In general, this will not yield an optimal acyclic circuit. However, in the case of Rivest's circuit, the bound of  $3n - 2$  is, in fact, tight. To obtain an acyclic circuit with  $3n - 2$  gates, we break the cycle and prepend a copy of the last  $n - 2$  gates. For  $n = 3$ , we simply prepend an OR gate with inputs  $x_2$  and  $x_3$ , as shown in Figure 24.

Rivest's circuit is also optimal seen from a different perspective. The circuit consists of AON gates, and yet none of the output functions are implementable with a single AON gate, regardless of the fan-in. Thus, any acyclic circuit implementing the functions requires at least one more gate.

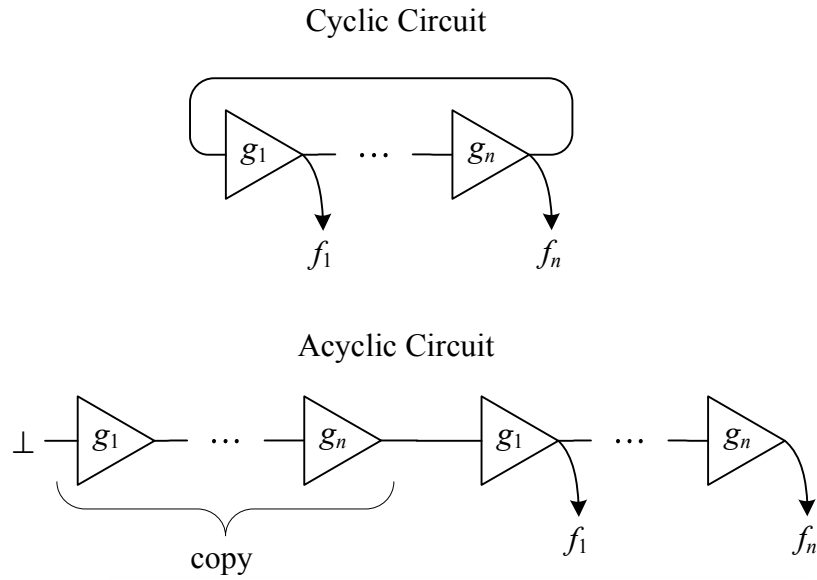


Figure 23: Obtaining an equivalent acyclic circuit from a cyclic circuit.

### 3.3.3. A Generalization

We note that Rivest's circuit can be generalized to AND and OR gates with arbitrary fan-in. The circuit shown in Figure 25 consists of  $2n$  fan-in  $d$  AND and OR gates, with  $n(d-1)$  inputs repeated, for  $n \geq 3$ ,  $n$  odd, and  $d \geq 2$ . This circuit produces outputs

$$\begin{aligned}
 f_1 &= y_1(y_n + y_{n-1}(\cdots(y_3 + y_2)\cdots)) \\
 f_2 &= y_2 + y_1(y_n + \cdots(y_4y_3)\cdots) \\
 &\vdots \\
 f_{2n} &= y_n + y_{n-1}(y_{n-2} + \cdots(y_2y_1)\cdots),
 \end{aligned}$$

where

$$\begin{aligned}
 y_1 &= x_1 \cdots x_{d-1} \\
 y_2 &= x_d + \cdots + x_{2d-2} \\
 &\vdots \\
 y_n &= x_{(n-1)(d-1)+1} + \cdots + x_{n(d-1)}.
 \end{aligned}$$

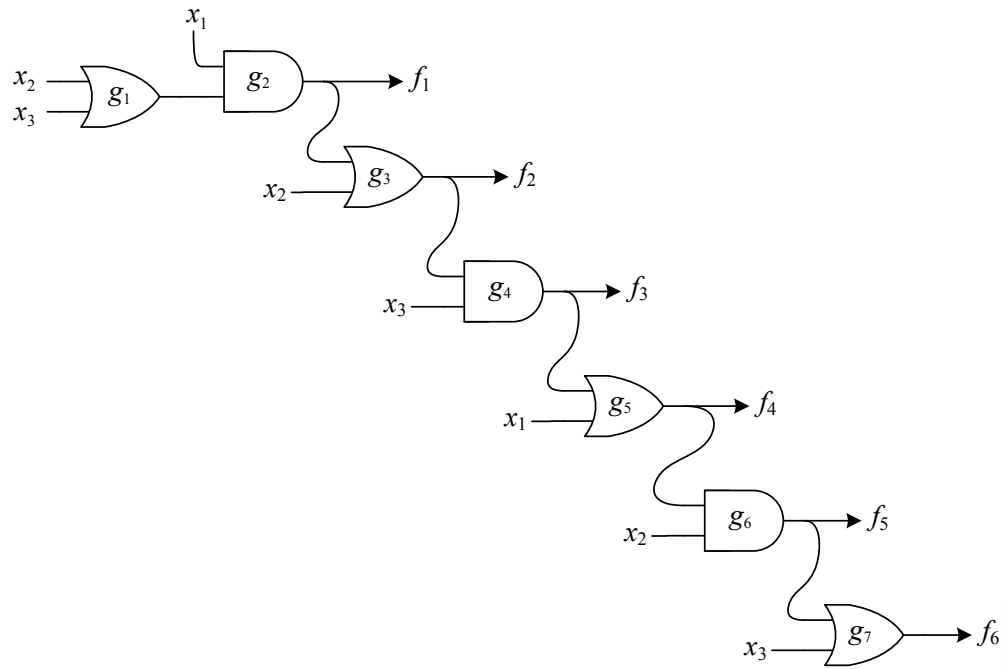


Figure 24: An acyclic circuit implementing the same functions as the circuit in Figure 21.

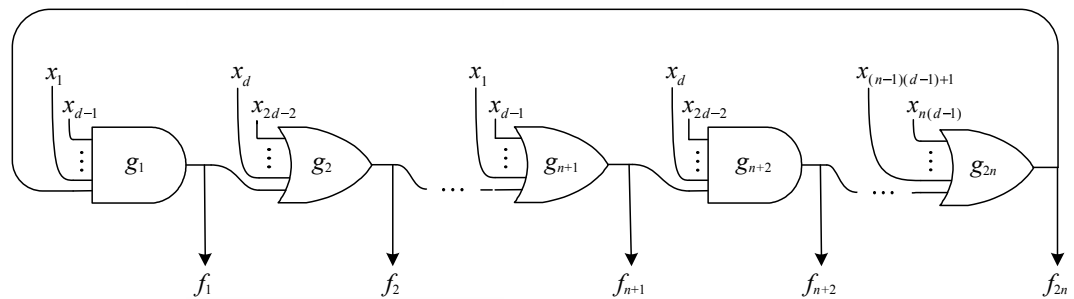


Figure 25: A generalization of Rivest's circuit to gates with fan-in greater than 2.

It may be shown that all  $2n$  functions are distinct, and that each depends on all  $n(d - 1)$  input variables.

### 3.3.4. Variants

We note that many different circuits of the same general form as Rivest's example exist. In Figure 26, we show a circuit with 4 variables and 8 gates in a single cycle. As with Rivest's circuit, this one produces distinct output functions each of which depends on all four variables.

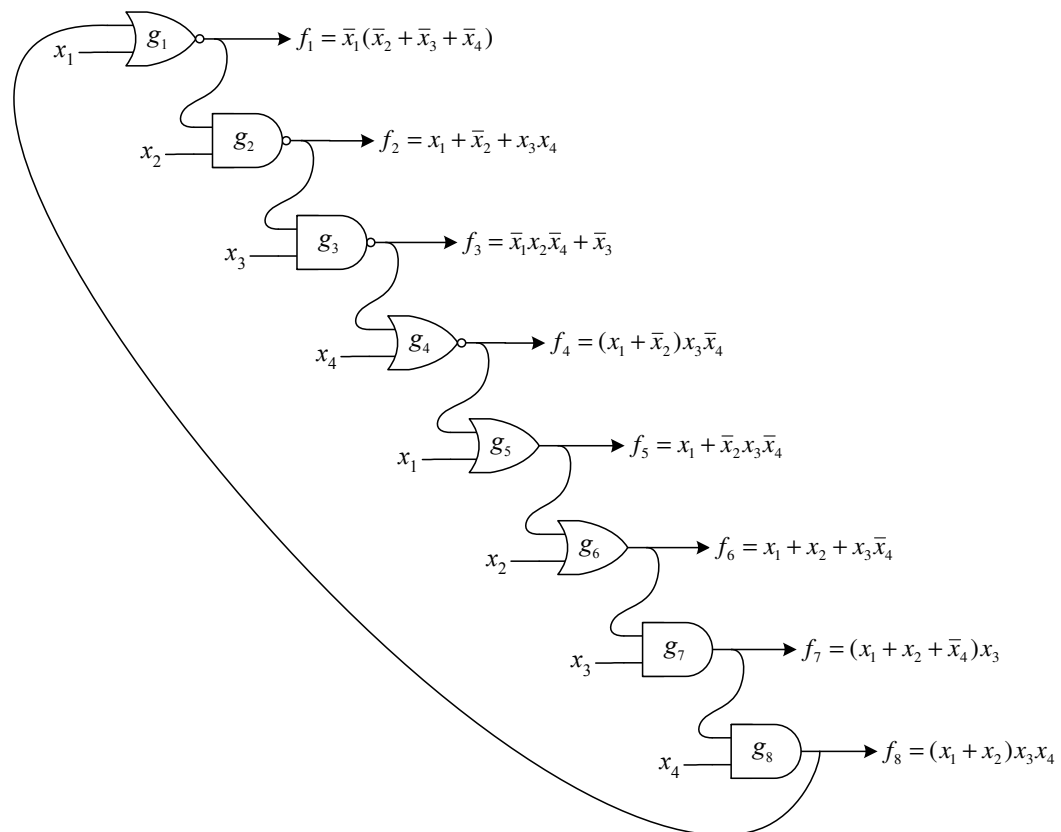


Figure 26: A circuit with the same properties as Rivest's example.

A more intriguing example is shown in Figure 27. It consists of two copies of Rivest's circuit with the outputs of the first fed as inputs into the second. Although not

shown here, we assert that this circuits produces 20 functions that are distinct, and each depends on all five variables.

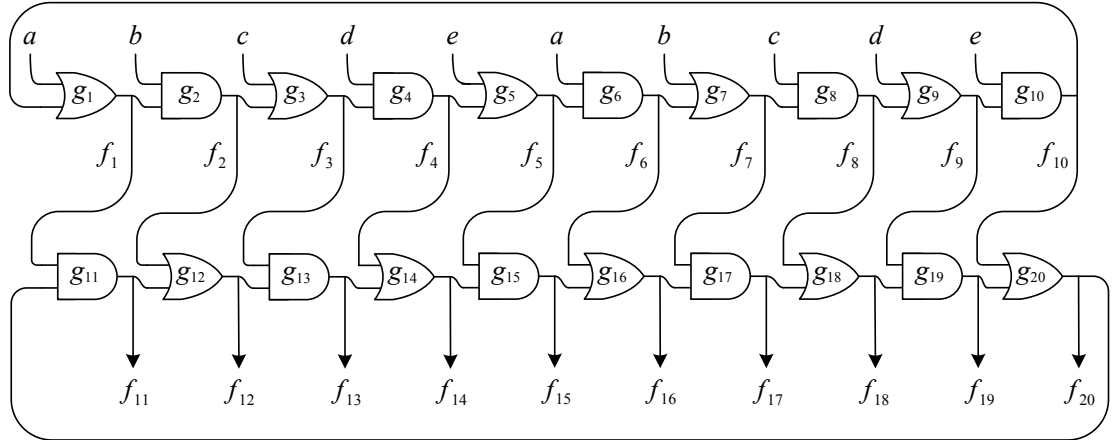


Figure 27: A pair of Rivest circuits,  $n = 5$ , stacked.

### 3.4. A Minimal Cyclic Circuit with Two Gates

We provide an example of a circuit with the same property as Rivest's circuit, but with only two gates. The circuit shown in Figure 28 consists of two fan-in 4 gates of the form

$$g(w, x, y, z) = wx \oplus yz.$$

connected in a cycle with 5 inputs,  $a, b, c, d, e$ . The circuit computes  $f$  and  $g$ :

$$\begin{aligned} f &= ab \oplus gc \\ g &= f\bar{c} \oplus de. \end{aligned}$$

To verify that the circuit is Boolean, note that if  $c = 0$ , then  $f$  and  $g$  assume definite values. We have

$$\begin{aligned} f|_{\bar{c}} &= ab \oplus (g|_{\bar{c}}) 0 = ab \\ g|_{\bar{c}} &= (f|_{\bar{c}}) 1 \oplus de = ab \oplus de. \end{aligned}$$

Similarly, if  $c = 1$ , then  $f$  and  $g$  also assume definite values. We have

$$\begin{aligned} g|_c &= (f|_c) 0 \oplus de = de \\ f|_c &= ab \oplus (g|_c) 1 = ab \oplus de. \end{aligned}$$

Assembling the output functions, we obtain

$$\begin{aligned}
 f &= \bar{c} \cdot (f|_{\bar{c}}) + c \cdot (f|_c) \\
 &= \bar{c}ab \oplus c(ab \oplus de) \\
 &= \bar{c}ab \oplus abc \oplus cde \\
 &= ab \oplus cde,
 \end{aligned}$$

$$\begin{aligned}
 g &= \bar{c} \cdot (g|_{\bar{c}}) + c \cdot (g|_c) \\
 &= \bar{c}(ab \oplus de) \oplus cde \\
 &= \bar{c}ab + \bar{c}dee + cde \\
 &= ab\bar{c} \oplus de.
 \end{aligned}$$

With the functions thus written in Reed-Muller form, we can readily assert that  $f$  and  $g$  are distinct and that each depends on all 5 variables. Now, consider an acyclic circuit, also with fan-in 4 gates, that computes the same functions. Since a single fan-in 4 gate cannot possibly compute a function of 5 variables, we conclude that the acyclic circuit must have at least 3 gates.

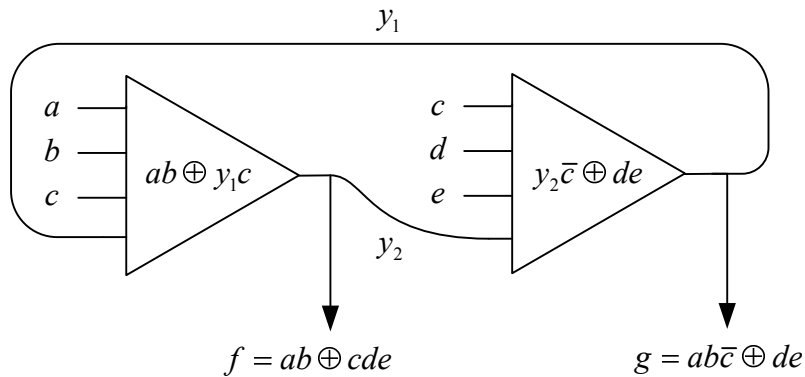


Figure 28: A cyclic circuit with two gates.

### 3.5. A Cyclic Circuit with Two Cycles

We continue the exposition of examples of cyclic Boolean circuits. We present an example with two cycles. This will be generalize to examples with multiple cycles,

culminating with the main result of the paper: a cyclic circuit that is one-half the size of any equivalent acyclic circuit.

Consider the circuit shown in Figure 29, written in a general form. The inputs are  $x_1, \dots, x_n$ , grouped together in the figure as  $X$ . (A diagonal line across a wire indicates that it represents multiple wires.) There are three gates, connected in a configuration consisting of two cycles:

$$\begin{aligned} f_1 &= \alpha_1 \oplus \beta_1 f_3 \\ f_2 &= \alpha_2 \oplus \beta_2 f_3 \\ f_3 &= \alpha_3 \oplus \beta_3 f_1 \oplus \gamma_3 f_2 \oplus \delta_3 f_1 f_2 \end{aligned}$$

where the  $\alpha$ 's,  $\beta$ 's,  $\gamma$ 's, and  $\delta$ 's are arbitrary functions of the input variables.

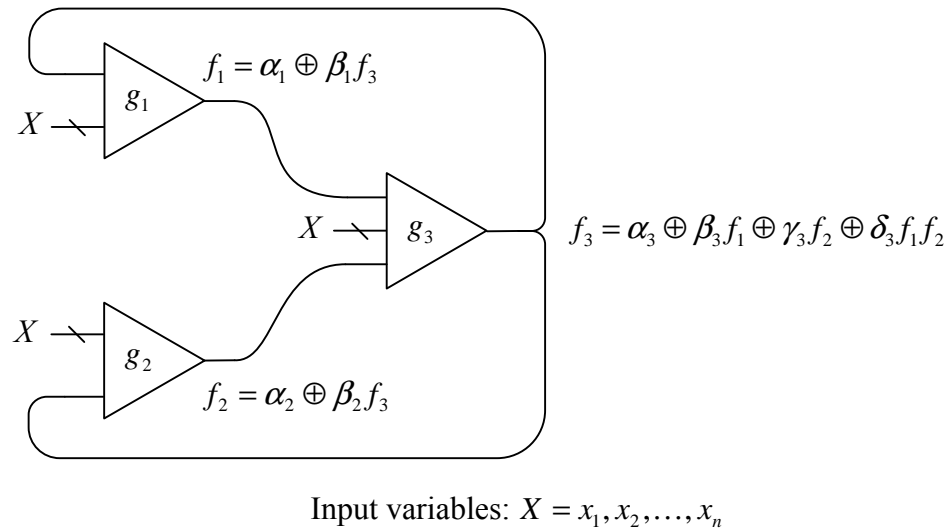


Figure 29: A cyclic circuit with two cycles.

We analyze this circuit with the goal of obtaining a necessary and sufficient condition for it to be Boolean, as well as expressions for the gate outputs in terms of the inputs. We proceed on a case-by-case basis. For what follows, when we speak of values of  $\alpha_i, \beta_i, \gamma_i$  and  $\delta_i$ , for  $i = 1, 2, 3$ , we mean the values evaluated at  $X$ .

**Case I**

Suppose that for some  $X$ , we have that  $\beta_1 = 0$ . In this case  $f_1$  assumes the definite value  $\alpha_1$ . This situation is shown in Figure 30. Now suppose further that  $\gamma_3 \oplus \delta_3 \alpha_1 = 0$ . In this

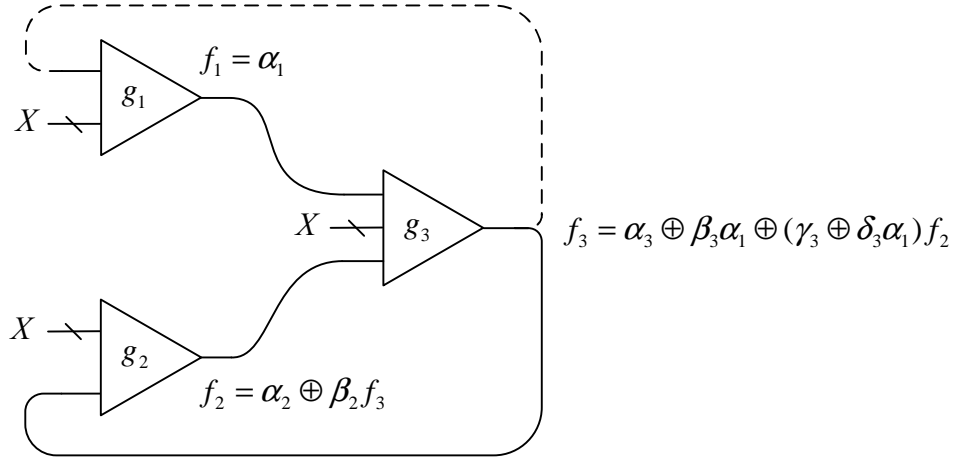


Figure 30: The circuit of Figure 29 if  $\beta_1 = 0$ .

case,  $f_3$  assumes a definite value of  $\alpha_3 \oplus \beta_3 \alpha_1$ . Given this value for  $f_3$ , it follows that  $f_2$  assumes the definite value of  $\alpha_2 \oplus \beta_2 \alpha_3 \oplus \beta_2 \beta_3 \alpha_1$ . This situation is shown in Figure 31. We conclude that the functions assume definite values if  $\beta_1 = 0$  and  $\gamma_3 \oplus \delta_3 \alpha_1 = 0$ .

**Case II**

Similarly, it can be shown that the functions  $f_1$ ,  $f_2$  and  $f_3$  assume definite values if  $\beta_2 = 0$  and  $\beta_3 \oplus \delta_3 \alpha_2 = 0$ .

**Case III**

Suppose that for some  $X$ , we have  $\beta_1 = 0$  and  $\beta_2 = 0$ . In this case  $f_1$  and  $f_2$  assume definite values of  $\alpha_1$  and  $\alpha_2$ , respectively. Given these values for  $f_1$  and  $f_2$ , it follows that  $f_3$  assumes the definite value of  $\alpha_3 \oplus \beta_3 \alpha_1 \oplus \gamma_3 \alpha_2 \oplus \delta_3 \alpha_1 \alpha_2$ . This situation is shown in Figure 32.



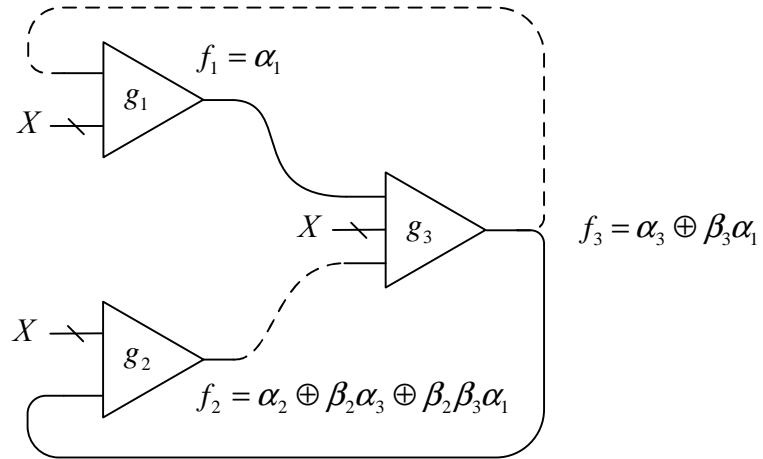


Figure 31: The circuit of Figure 29 if  $\beta_1 = 0$  and  $\gamma_3 \oplus \delta_3 \alpha_1 = 0$ .

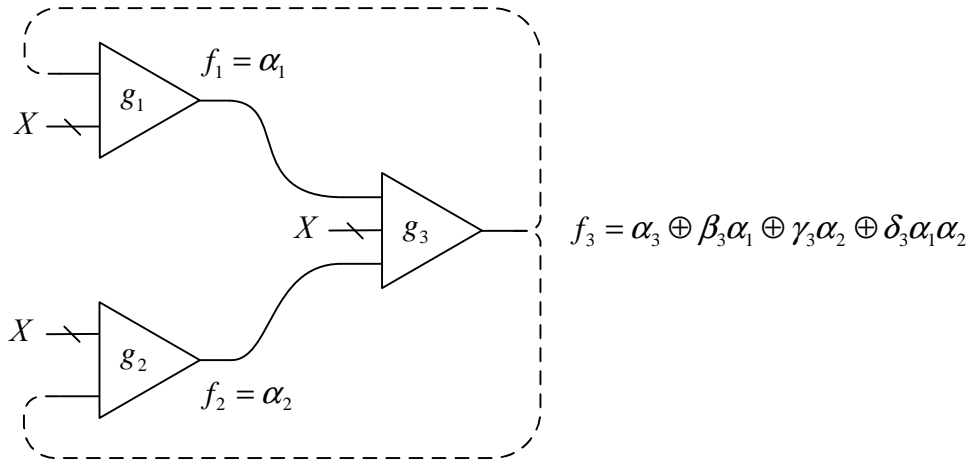


Figure 32: The circuit of Figure 29 if  $\beta_1 = 0$  and  $\beta_2 = 0$ .

**Case IV**

Suppose that  $\beta_3 = \gamma_3 = \delta_3 = 0$ . In this case  $f_3$  assumes the definite value  $\alpha_3$ . Given this value for  $f_3$ , it follows that  $f_1$  and  $f_2$  assume the definite values  $\alpha_1 \oplus \beta_1\alpha_3$  and  $\alpha_2 \oplus \beta_2\alpha_3$ , respectively. This situation is shown in Figure 33.  $\square$

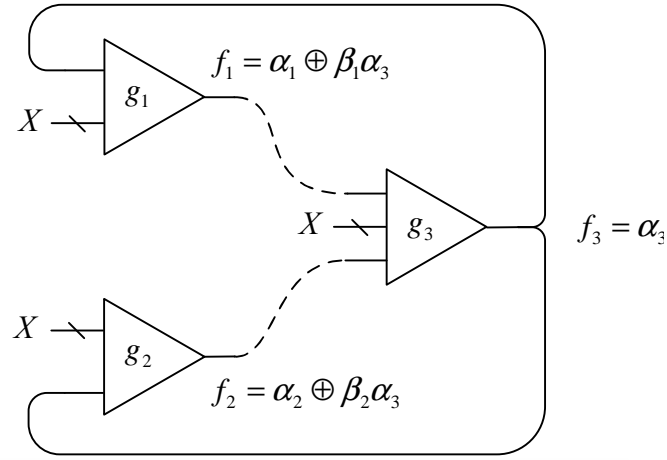


Figure 33: The Circuit of Figure 29 if  $\beta_3 = \gamma_3 = \delta_3 = 0$ .

The characteristic functions for the four cases are:

$$c_1(X) = \overline{\beta_1} \cdot \overline{(\gamma_3 \oplus \delta_3 \alpha_1)}, \quad (2)$$

$$c_2(X) = \overline{\beta_2} \cdot \overline{(\beta_3 \oplus \delta_3 \alpha_2)}, \quad (3)$$

$$c_3(X) = \overline{\beta_1} \cdot \overline{\beta_2}, \quad (4)$$

$$c_4(X) = \overline{\beta_3} \cdot \overline{\gamma_3} \cdot \overline{\delta_3}. \quad (5)$$

We conclude that the circuit is Boolean if and only if

$$c_1(X) + c_2(X) + c_3(X) + c_4(X) \equiv 1.$$

If this condition holds, then the functions have values

$$f_1(X) = c_1\alpha_1 + c_2(\alpha_1 \oplus \beta_1\alpha_3 \oplus \beta_1\gamma_3\alpha_2) + c_3\alpha_1 + c_4(\alpha_1 \oplus \beta_1\alpha_3) \quad (6)$$

$$f_2(X) = c_1(\alpha_2 \oplus \beta_2\alpha_3 \oplus \beta_1\beta_3\alpha_1) + c_2\alpha_2 + c_3\alpha_2 + c_4(\alpha_2 \oplus \beta_2\alpha_3) \quad (7)$$

$$f_3(X) = c_1(\alpha_3 \oplus \beta_3\alpha_1) + c_2(\alpha_3 \oplus \gamma_3\alpha_2) + c_3(\alpha_3 \oplus \beta_3\alpha_1 \oplus \gamma_3\alpha_2\delta_3\alpha_1\alpha_2) + c_4\alpha_3. \quad (8)$$

### 3.6. A Circuit Three-Fifths the Size

Let's make the circuit of Figure 29 somewhat more concrete. Suppose that the inputs are  $a, b, x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n, z_1, z_2, \dots, z_n$ . Suppose that the gates are defined by

$$\alpha_1 = \bar{a}X, \quad \alpha_2 = \bar{b}Y, \quad \alpha_3 = Z,$$

where

$$X = x_1x_2\cdots, x_n, \quad Y = y_1y_2\cdots, y_n, \quad Z = z_1z_2\cdots, z_n$$

and

$$\begin{aligned} \beta_1 &= a, & \beta_2 &= b, & \beta_3 &= \bar{a}, \\ \gamma_3 &= \bar{b}, & \delta_3 &= \bar{a}\bar{b}. \end{aligned}$$

The resulting circuit is shown in Figure 34. For this circuit, the conditions defined in Equations 2–5 evaluate to:

$$\begin{aligned} c_1 &= \bar{a}(b + X), \\ c_2 &= \bar{b}(a + Y), \\ c_3 &= \bar{a}\bar{b}, \\ c_4 &= ab. \end{aligned}$$

It may easily be verified that for every combination of values assigned to  $a$  and  $b$ , one of  $c_1, c_2, c_3, c_4$  is true. The functions defined in Equations 6–8 are

$$\begin{aligned} f_1(a, b, X, Y, Z) &= X \oplus a(X \oplus Y \oplus Z) \oplus abY, \\ f_2(a, b, X, Y, Z) &= Y \oplus b(X \oplus Y \oplus Z) \oplus abX, \\ f_3(a, b, X, Y, Z) &= X \oplus Y \oplus Z \oplus XY \oplus a(X \oplus XY) \oplus b(Y \oplus XY) \oplus abXY. \end{aligned}$$

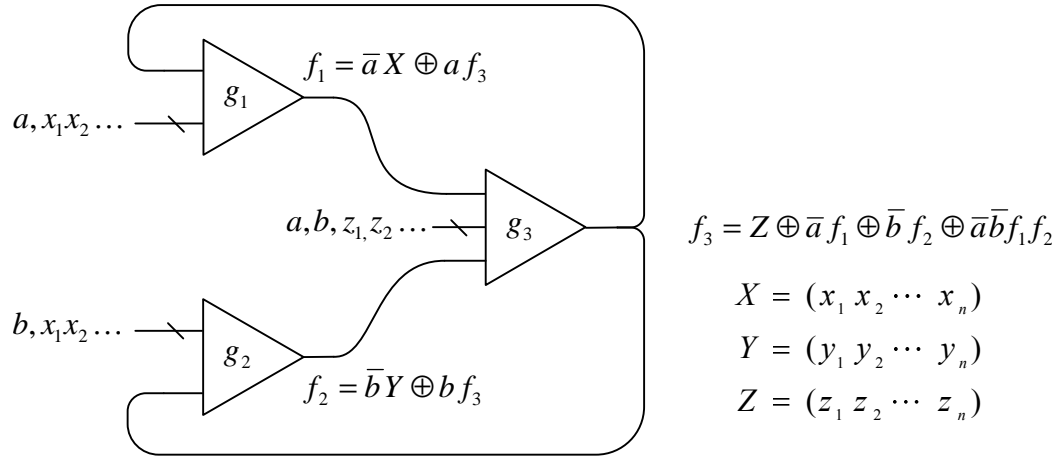


Figure 34: Variant of the circuit of Figure 29.

With the functions expressed in Reed-Muller form, we can assert that they are distinct and that each depends on all the variables. To make the situation more concrete, suppose that

$$X = c e g i, \quad Y = d f h j, \quad Z = k l.$$

There are a total of 12 variables ( $a$  through  $l$ ). Each gate has fan-in 6. This situation is shown in Figure 35. According to Claim 2, an acyclic circuit implementing the same functions requires at least

$$\left\lceil \frac{v-1}{d-1} \right\rceil + m - 1$$

gates, where  $v = 12$  (the number of variables),  $d = 6$  (the fan-in) and  $m = 3$  (the number of functions). Thus

$$\left\lceil \frac{v-1}{d-1} \right\rceil + m - 1 = \left\lceil \frac{12-1}{6-1} \right\rceil + 3 - 1 = 5.$$

We conclude that the circuit in Figure 35 is at most  $\frac{3}{5}$  the size of any equivalent acyclic circuit.

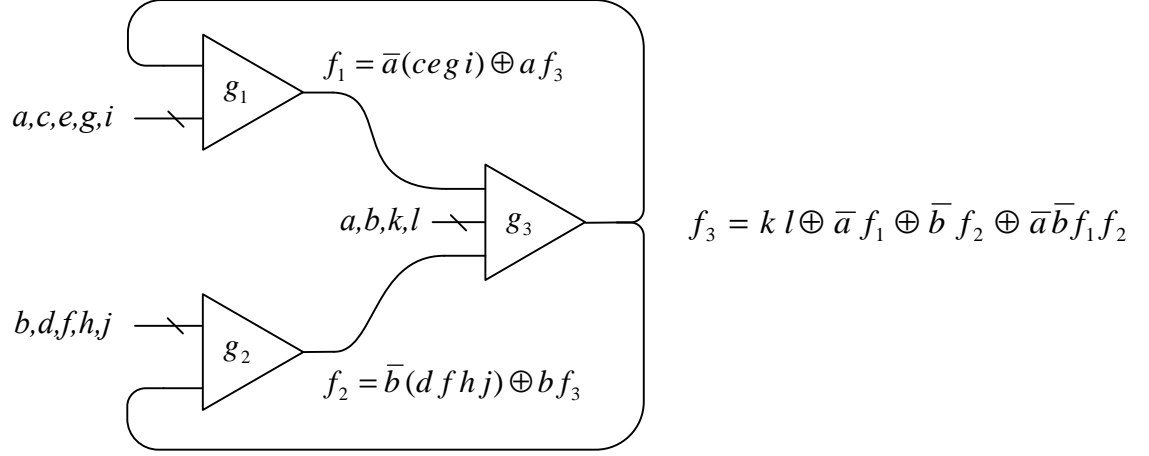


Figure 35: Circuit of Figure 34 with 12 variables.

### 3.7. A Circuit One-Half the Size

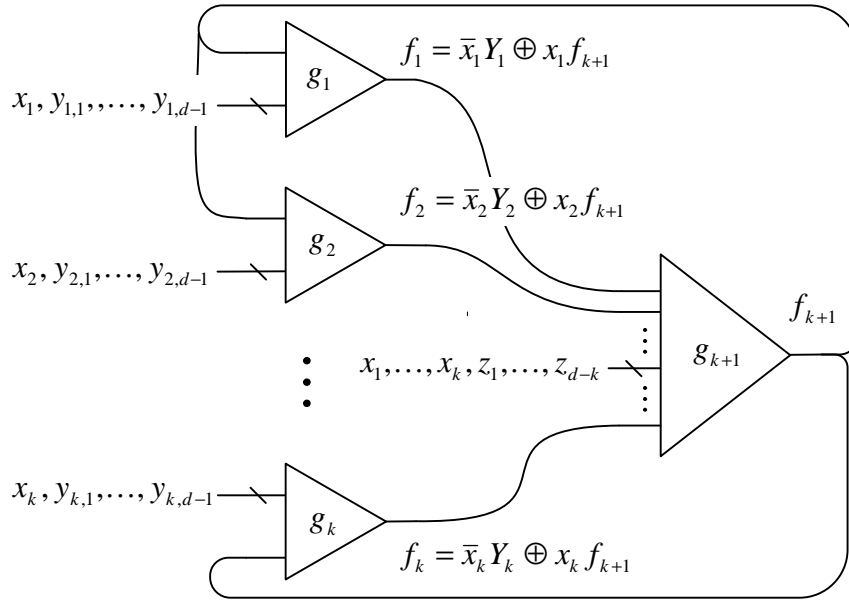
Consider the circuit shown in Figure 36, a generalization of the circuit in Figure 29 to  $k$  gates. We argue the validity of this circuit informally. On the one hand, for each variable  $x_i$ , if  $x_i = 0$  then  $f_i$  does not depend on  $f_{k+1}$ . On the other hand, if  $x_i = 1$ , then  $f_{k+1}$  does not depend on  $f_i$ . We conclude that none of the  $k$  cycles can be sensitized, and so the circuit is Boolean. Now consider the function  $f_i$  implemented by each gate. With  $x_i = 0$ , we see that  $f_i$  depends on the variables  $y_{1,1}, \dots, y_{1,d-1}$ . Since  $f_{k+1}$  depends on  $f_i$ , it also depends on these variables. Thus  $f_{k+1}$  depends all the variables  $y_{i,j}$  for  $i = 1, \dots, k$  and  $j = 1, \dots, d - 1$ . With  $x_i = 1$ , we see that  $f_i$  depends on  $f_{k+1}$ ; hence it also depends on all these variables. We conclude that each function depends on all the variables.

With the fan-in of the gates set to  $d$ , the number of variables is

$$v = k(d - 1) + d - 2k = (k + 1)d - 3k$$

and the number of gates is

$$m = k + 1.$$



$$Y_i = (y_{i,1} \cdots y_{i,d-1}) \quad i = 1, \dots, k$$

$$Z = (z_1 \cdots z_{d-k})$$

$$f_{k+1} = Z \oplus \bar{x}_1 f_1 \oplus \bar{x}_2 f_2 \oplus \cdots \oplus \bar{x}_k f_k \oplus \bar{x}_1 \cdots \bar{x}_k f_1 \cdots f_k$$

Figure 36: A generalization of the circuit of Figure 29.

According to Claim 2, an acyclic circuit implementing the same functions requires at least

$$\left\lceil \frac{v-1}{d-1} \right\rceil + m - 1$$

gates. The improvement factor is,

$$\frac{m}{\left\lceil \frac{v-1}{d-1} \right\rceil + m - 1} = \frac{k+1}{\left\lceil \frac{(k+1)d-3k-1}{d-1} \right\rceil + k} = \frac{k+1}{\left\lceil k+1 - \frac{2k}{d-1} \right\rceil + k}$$

gates. Suppose that  $d = 2k + 1$  and that  $k$  is large. Then the ratio is

$$\frac{k+1}{2k} \approx \frac{1}{2}.$$

We conclude that the circuit of Figure 36 is at most one-half the size of any equivalent acyclic circuit. According to Claim 3, this is the best possible improvement factor that we can obtain with the fan-in lower bound of Section 3.1.

#### 4. Discussion

In related work, we have explored the practical implications of cyclic Boolean circuits. We have described an efficient approach for *analyzing* cyclic circuits, and we have provided a general framework for *synthesizing* such circuits. Our synthesis tool, called CYCLIFY, is built within the Berkeley SIS and ABC environments (Sentovich et al., 1992; Mishchenko et al., 2007). Timing analysis is performed with binary decision diagrams (BDDs) and Boolean satisfiability (SAT)-based techniques (Riedel and Bruck, 2003a, 2004; Backes et al., 2008). Synthesis is performed by introducing cycles through the incremental application of restructuring and minimization operations (Riedel and Bruck, 2003b). We have explored the technique of Craig interpolation for synthesizing cyclic functional dependencies (McMillan, 2003; Backes and Riedel, 2009). Perhaps the most salient result to report is that cyclic solutions are not a rarity. On trials with industry-accepted benchmark circuits, we improved the design of many circuits significantly in terms of area and delay by introducing cycles.

In theoretical computer science, one of the main goals is to prove lower bounds on the resources, such as time and space, required for computation. The Boolean circuit model is often postulated as a representative model for computation with digital circuits. Unlike more abstract models of computation, such as Turing machines, Boolean circuits can be viewed as bit-level implementation of a computational procedure – in a sense, the most basic way that one can compute something. A lower bound on the circuit size is taken as a true measure of the computational requirements of a problem. For instance, lower bounds on circuit size have been used to justify the security of cryptographic algorithms (Rothe, 2002). Complexity theorists invariably define a Boolean circuit as a directed acyclic graph (DAG) (Papadimitriou, 1995, p. 80) and (Vollmer, 1999, p. 8). It is conceivable that some of the proofs of lower bounds on circuit size depend on this definition.

We hope that this paper will help promulgate the view that a Boolean circuit is not necessarily a DAG; rather it is a directed graph that may have cycles. This distinction is a fundamental one. We have demonstrated that, both in theory and in practice, it is not only feasible to design Boolean circuits with cyclic topologies, but this may well be the best way to design them.



- Backes, J., Fett, B., Riedel, M. D., 2008. The analysis of cyclic circuits with Boolean satisfiability. In: International Conference on Computer-Aided Design. pp. 143–148.
- Backes, J., Riedel, M. D., 2009. The synthesis of cyclic dependencies with Craig interpolation. In: International Workshop on Logic and Synthesis. pp. 24–30.
- Brzozowski, J., Seger, C.-J., 1995. Asynchronous Circuits. Springer-Verlag.
- Huffman, D. A., 1971. Combinational circuits with feedback. In: Mukhopadhyay, A. (Ed.), Recent Developments in Switching Theory. Academic Press, pp. 27–55.
- Katz, R., 1992. Contemporary Logic Design. Benjamin/Cummings.
- Kautz, W. H., 1970. The necessity of closed circuit loops in minimal combinational circuits. IEEE Transactions on Computers C-19 (2), 162–164.
- Khrapchenko, V., 1978. Depth and delay in a network (in Russian). Soviet Mathematics – Doklady 19, 1006–1009.
- Markov, A., 1958. On the inversion complexity of a system of functions. Journal of the ACM 5, 331–334.
- McCaw, C., 1963. Loops in directed combinational switching networks. Ph.D. thesis, Stanford University.
- McMillan, K. L., 2003. Interpolation and SAT-based model checking. In: International Conference on Computer Aided Verification. pp. 1–13.
- Mishchenko, A., et al., 2007. ABC: A system for sequential synthesis and verification. URL <http://www.eecs.berkeley.edu/~alanmi/abc/>
- Papadimitriou, C. H., 1995. Computational Complexity. Addison-Wesley.
- Riedel, M. D., Bruck, J., 2003a. Cyclic combinational circuits: Analysis for synthesis. In: International Workshop on Logic and Synthesis. pp. 105–112.
- Riedel, M. D., Bruck, J., 2003b. The synthesis of cyclic combinational circuits. In: Design Automation Conference. pp. 163–168.

- Riedel, M. D., Bruck, J., 2004. Timing analysis of cyclic combinational circuits. In: International Workshop on Logic and Synthesis.
- Rivest, R. L., 1977. The necessity of feedback in minimal monotone combinational circuits. *IEEE Transactions on Computers* 26 (6), 606–607.
- Rothe, J., 2002. Some facets of complexity theory and cryptography. *ACM Computing Surveys* 34 (4), 504–549.
- Sentovich, E. M., et al., 1992. SIS: A system for sequential circuit synthesis. Tech. rep., University of California, Berkeley.
- Shannon, C. E., 1949. The synthesis of two terminal switching circuits. *Bell System Technical Journal* 28, 59–98.
- Short, R., 1961. A theory of relations between sequential and combinational realizations of switching functions. Ph.D. thesis, Stanford University.
- Vollmer, H., 1999. *Introduction to Circuit Complexity*. Springer.
- Wakerly, J. F., 2000. *Digital Design: Principles and Practices*. Prentice-Hall.
- Wegener, I., 1987. *The Complexity of Boolean Functions*. John Wiley & Sons.
- Zhegalkin, I., 1929. The arithmetization of symbolic logic (in Russian). *Matematicheskii Sbornik* 36, 205–338.