Discrete Applied Mathematics 160 (2012) 1877–1900

Contents lists available at SciVerse ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Cyclic Boolean circuits[☆]

Marc D. Riedel [a,*], Jehoshua Bruck [b]

[a] *Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, USA*

[b] *Electrical Engineering, California Institute of Technology, Pasadena, CA 91125, USA*

ARTICLE INFO

Article history:
Received 15 January 2010
Received in revised form 26 August 2011
Accepted 31 March 2012
Available online 14 May 2012

Keywords:
Boolean circuits
Boolean functions
Combinational circuits
Cyclic circuits
DAG
Cycles
Loops
Feedback

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 of gates that are required by equivalent acyclic circuits. We also show a quadratic upper bound: given a cyclic Boolean circuit with m gates, there exists an equivalent acyclic Boolean circuit with m^2 gates.

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:

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

[☆] 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.

[*] Corresponding author. Tel.: +1 612 625 6086; fax: +1 612 625 4583.
E-mail addresses: mriedel@umn.edu (M.D. Riedel), bruck@paradise.caltech.edu (J. Bruck).

0166-218X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2012.03.039

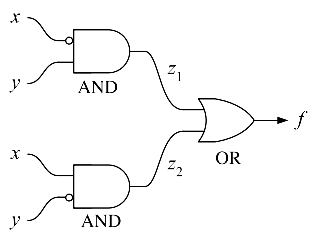

Fig. 1. An acyclic circuit.

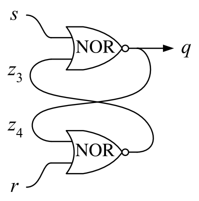

Fig. 2. A cyclic circuit.

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 Fig. 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 Fig. 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 Fig. 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*.)

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?

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., [19, p. 80] and [29, p. 8]. Circuit practitioners design combinational circuits without cycles: e.g., [10, p. 14] and [30, p. 193]. Design automation tools enforce this constraint [26].

1.2. Cyclic Boolean circuits

Accepted wisdom and theoretical practice are wrong: a circuit with a cyclic topology can be Boolean. A trivial circuit, shown in Fig. 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 Fig. 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 Fig. 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 $f = x$).