Finite Digital Synchronous Circuits are characterized by 2-Algebraic Truth Tables

Jean Vuillemin[1]

Ecole Normale Supérieure, 45 rue d'Ulm, 75230 Paris cedex 05, France.
Jean.Vuillemin@ens.fr

Abstract. A *digital* function maps sequences of binary *inputs*, into sequences of binary *outputs*. It is *causal* when the output at cycle N is a boolean function of the input, from cycles 0 through N.
A causal digital function f is characterized by its *truth table*, an infinite sequence of bits $(F_{\mathbf{N}})$ which gathers all outputs for all inputs. It is identified to the power series $\sum F_{\mathbf{N}} z^{\mathbf{N}}$, with coefficients in the two elements field $\mathbf{F}_2$.

Theorem 1. *A digital function can be computed by a* finite digital synchronous circuit*, if and only if it is* causal*, and its truth table is an* algebraic number *over* $\mathbf{F}_2[z]$*, the field of polynomial fractions* $\pmod 2$.

A data structure, *recursive sampling*, is introduced to provide a *canonical* representation, for each *finite causal* function f. It can be mapped, through finite algorithms, into a circuit $SDD(f)$, an automaton $SBA(f)$, and a polynomial $\mathrm{poly}(f)$; each is *characteristic* of f. One can thus automatically synthesize a canonical circuit, or software code, for computing any *finite causal* function f, presented in some effective form. Through recursive sampling, one can verify, in finite time, the validity of any hardware circuit or software program for computing f.

1 Physical Deterministic Digital System

Consider a *discrete time digital system*: at each integer cycle N[1], the system receives input bits $x_{\mathbf{N}} \in \mathbf{B} = \{0, 1\}$, and emits output bits $y_{\mathbf{N}} \in \mathbf{B}$. The *function* f of this system, is to map *infinite* sequences of input bits $x = (x_{\mathbf{N}})$, into infinite sequences of output bits $y = f(x) = (y_{\mathbf{N}})$. Call *digital* such a function $f \in \mathbf{D} \mapsto \mathbf{D}$, where $\mathbf{D} = \mathbf{N} \mapsto \mathbf{B}$ is the set of infinite binary sequences. Our aim is to characterize which functions can be computed by deterministic digital *physical* systems, such as electronic circuits, and which *cannot*.

To simplify, we exclude *analog* [1], and *asynchronous* systems. As long as the function of such *exotic* systems remains deterministic, and digital, an equivalent system may be implemented through a *digital synchronous* electronic chip. The

[1] Throughout this text, reserve the letter N to range over the natural numbers: $\mathrm{N} \in \mathbf{N}$.

concept of *Digital Synchronous Circuit* **DSC**, provides a mathematical model for the *form* and *function* of this class of physical systems.

Established techniques exist to map finite **DSC** descriptions, into silicon chips [2]. With *reconfigurable systems* [3], the process can be fully automated: from a finite **DSC** representation for *mathematically* computing f, *compile* a binary configuration, and download into some *programmable* device, in order to *physically* compute f. Without further argument, admit here that the class of functions defined by *finite* **DSC** captures the proper mathematical concept, from the motivation question. No regard is given to size limitations, arising from technology, economics, or else.

- Physical circuits are constrained by *time causality*: output $y(t)$ at time t may only depend upon inputs $x(t')$, from the *past* $t' < t$.
- From their physical nature, electronic circuits must be *finite*.

Causality and finiteness are thus necessary conditions, for digital functions to be computable by deterministic *physical* devices. We show that they are sufficient, and characterize *finite causal* functions, in a constructive way.

1.1 Infinite SDD Procedure

A first answer to the motivating question is provided in [4], through an *infinite* construction, the *Synchronous Decision Diagrams* **SDD**.

Theorem 2 (Vuillemin [4]).

1. *To any causal function f, one can associate a canonical circuit SDD(f) $\in$* **DSC** *for computing f.*
2. *Circuit SDD(f) is* finite*, if and only if function f is computable by some finite system.*

Yet, the *infinite* **SDD** construction, relies on the ability to test for equality $g = h$ between digital functions g, h. This operation is *not computable* in general, even when g and h are both computable. Also, the definition of *"finiteness"* is not made explicit in Theorem 2, and the input to the *"procedure"* is ill-specified.

Such limitations are partly removed, by Berry [5] and Winkelman [6]: both base implementations of the **SDD** procedure, on representing digital functions by a *Finite State Machines* **FSM**.

2 Binary Algebra

Infinite binary sequences $\mathbf{D} \rightleftharpoons \mathbf{N} \mapsto \mathbf{B}$ have a rich mathematical structure. A digital sequence $a \in \mathbf{D}$ codes, in a unique way: the set $\{a\} = \{\mathrm{N} : a_{\mathbf{N}} = 1\}$ of integers; the formal power series $a(z) = \sum a_{\mathbf{N}} z^{\mathbf{N}}$; the 2-adic *"integer"* $a(2) = \sum a_{\mathbf{N}} z^{\mathbf{N}}$: $\mathbf{D} \rightleftharpoons \wp(\mathbf{N}) \rightleftharpoons \mathbf{F}_2(z) \rightleftharpoons \mathbf{Z}_2$. We identify all representations, and write (see [4]), for example:

$$(01) = \{1 + 2\mathrm{N}\} = -\frac{2}{3} = z/(1 + z^2).$$

Binary Algebra imports all underlying operations, into a single structure:

$$\langle \mathbf{D}, \neg, \cup, \cap, z, z^{-}, \oplus, \otimes, +, -, \times, \uparrow, \downarrow \rangle .$$

1. $\langle \mathbf{D}, \neg, \cup, \cap \rangle$ is a *Boolean Algebra*, isomorphic to sets $\wp(\mathbf{N})$ of integers;
2. $\langle \mathbf{D}, z, \oplus, \otimes \rangle$ is a ring, isomorphic to the formal power series $\mathbf{F}_2(z)$;
3. $\langle \mathbf{D}, 0, 1, +, -, \times \rangle$ is a ring, isomorphic to the 2-adic integers $\mathbf{Z}_2$.

The *up-sampling* operator is noted $\uparrow x = x(z^2) = x \otimes x$. The *down-sampling* operator is noted $\downarrow x = \downarrow (x_{\mathbf{N}}) = (x_{2\mathbf{N}})$. See the related Noble identities, in the appendix.

In addition to the axiomatic relations implied by each of the three structures in $\mathbf{D}$, *hybrid* relations exist between the operators in Binary Algebra. Some are listed in the appendix. There are more: indeed, each arithmetical circuit implements some hybrid relation [4]. For example, base -2 coding, is defined by

$$\sum_{k \leq \mathbf{N}} x_k 2^k \ = \ \sum_{k \leq \mathbf{N}} y_k (-2)^k \pmod{2^{\mathbf{N}}}. \tag{1}$$

It is also known as Booth coding $y = \text{booth}(x)$, Polish code (in [7]), and may be computed by the hybrid formula:

$$\text{booth}(x) = (01) \oplus (x + (01)).$$

The infinite Binary Algebra $\mathbf{D} = \mathbf{Z}_2$, contains noteworthy sub-structures:

$$\mathbf{F}_2 \subset \mathbf{B}^{\mathbf{N}} \subset \mathbf{N} \subset \mathbf{Z} \subset \mathbf{P}2 \subset \mathbf{P} \subset \mathbf{A}_2 \subset \mathrm{Z}_2 \subset \mathbf{Z}_2.$$

Here: $\mathbf{B}^{\mathbf{N}}$ are the finite sequences, $\mathbf{P}$ the *ultimately periodic* sequences, $\mathbf{P}2$ those of period length $2^{\mathbf{N}}$, $\mathbf{A}_2$ the 2-algebraic (definition 6), and Z_2 the *computable* 2-adic integers. The appendix lists the closure properties of these sets, with respect to Binary Algebra operations.

3 Causal Function

Let $\|x\| \in \mathbf{Q}$ denote the *2-adic norm* of $x \in \mathbf{D}$: $\|0\| = 0$, $\|1 + zx\| = 1$, and $\|zx\| = \|x\|/2$. The *distance* $\|a - b\|$, between digital sequences $a, b \in \mathbf{D}$, is *ultra-metric*: $\|a + b\| \leq \max\{\|a\|, \|b\|\}$. Note that: $\|a - b\| = \|a \oplus b\|$.

Definition 1. *A digital function is* causal*, when the following (equivalent statements) hold:*

1. $\forall a, b \in \mathbf{D} : \ \|f(a) - f(b)\| \leq \|a - b\|$.
2. *Each output bit is a* Boolean function $f_{\mathbf{N}} \in \mathbf{B}^{\mathbf{N}+1} \mapsto \mathbf{B}$*, which exclusively depends on the first* $\mathbf{N} + 1$ *bits of input:*

$$y_{\mathbf{N}} = f_{\mathbf{N}}(x_0\, x_1 \cdots x_{\mathbf{N}-1}\, x_{\mathbf{N}}) = f_{\mathbf{N}}(x),$$
$$y = (y_{\mathbf{N}}) = f(x) = (f_{\mathbf{N}}(x)) = \sum f_{\mathbf{N}}(x) z^{\mathbf{N}}.$$

The operators $\neg, \cap, \cup, \oplus, z, \otimes, \oslash, \uparrow, +, -, \times, /$ are *causal*. The *antiflop* z^{-} (defined by $y_{\mathbf{N}} = x_{\mathbf{N}+1}$), and down-sampling $\downarrow$ are *not causal*. We simply say *causal* f, when f is a *causal* digital function, with a single input x, and a single output $y = f(x)$; otherwise, we explicitly state the number of inputs, and outputs.

3.1 Truth Table

Definition 2. *The* truth table *of a causal function* $f(x) = (f_{\mathbf{N}}(x))$*, combines the tables for each Boolean function* $f_{\mathbf{N}} \in \mathbf{B}^{\mathbf{N}+1} \mapsto \mathbf{B}$*, into a unique digital sequence* $truth(f) = F = (F_{\mathbf{N}}) \in \mathbf{D}$*, defined by*

$$F_{\mathbf{N}} \ = \ f_m(b_0\, b_1 \cdots b_{m-1}\, b_m);$$

here: $m = \lfloor log_2(\mathrm{N}+2)\rfloor - 1$*, and* $\sum_{k\leq m} b_k 2^k = \mathrm{N} + 2 - 2^{m+1}$.

Proposition 1. *The truth table* $F = truth(f) \in \mathbf{D}$ *is a* one-to-one *digital code, for each causal function* $f = truth^{-}(F) \in \mathbf{D} \mapsto \mathbf{D}$.

Proposition 2. *For causal* f *and* g*:*

$$\begin{aligned} truth(\neg f) &= \neg truth(f), \\ truth(f \cup g) &= truth(f) \cup truth(g), \\ truth(f \cap g) &= truth(f) \cap truth(g), \\ truth(\widetilde{zf}) &= 1 + z \uparrow z\, truth(\widetilde{f}). \end{aligned}$$

3.2 Automatic Sequence

Although it is traditionally associated to a *finite* causal f, which is explicitly presented by a *finite state automaton*, the definition of an *automatic sequence* [9], may be extended to all causal functions, finite and infinite.

Definition 3. *The* automatic sequence $auto(f) = (a_{\mathbf{N}}) \in \mathbf{D}$*, is associated to the causal function* f*, by:*

$$a_{\mathbf{N}} = f_m(b_0\, b_1 \cdots b_{m-1}\, b_m),$$

where $m = 0$ *if* $\mathrm{N} = 0$*, else* $m = \lfloor log_2(\mathrm{N})\rfloor$*, and* $\mathrm{N} = \sum_{k\leq m} b_k 2^k$.

In general, the value $y = (y_{\mathbf{N}}) = f(x)$ of causal f, at $x = (x_{\mathbf{N}})$, *cannot* be reconstructed, from its automatic sequence auto(f). Indeed, consider the causal: firstbit$(x) = x\cap 1$, and zerotest$(x) = \neg z^{-}(-x \oplus x)$. Both have the *same* automatic number: auto(firstbit) $=$ auto(zerotest) $= 1(0)$. While truth(firstbit) $= 1$, we have $T =$ truth(zerotest) $= 10100010000000100000000000000100\cdots \neq 1$.

Proposition 3. *Let* f *be causal. The derived* causal *functions,* $g(x) = f(\neg z \neg x)$ *and* $h(x) = zf(z^{-}x)$*, are such that:*

$$\begin{aligned} auto(f) &= truth(g), \\ truth(f) &= z^{-2}\, auto(h). \end{aligned}$$

3.3 Time Reversal

Definition 4. *The* time reversed *function* $\widetilde{f}$*, is defined by*

$$\widetilde{f}(x) = \sum f_{\mathbf{N}}(x_{\mathbf{N}} \cdots x_0) z^{\mathbf{N}},$$

where the causal function f *is given (definition 1) by:*

$$f(x) = \sum f_{\mathbf{N}}(x_0 \cdots x_{\mathbf{N}}) z^{\mathbf{N}}.$$

The *reversed* truth table $\mathrm{truth}(\widetilde{f}) = (F_{\widetilde{\mathbf{N}}})$, is related to $\mathrm{truth}(f) = (F_{\mathbf{N}})$ through:

$$\widetilde{\mathbf{N}} = (0\ 1\ 2\ 4\ 3\ 5\ 6\ 10\ 8\ 12\ 7\ 11\ 9\ 13\ 14\ 22 \cdots).$$

Let $\mathrm{prefix}(f) = \{z^{-b} f(a + z^b x) : a, b \in \mathbf{N}, a < 2^b\}$, and $\mathrm{suffix}(f) = \mathrm{prefix}(\widetilde{f})$.

Proposition 4. *The class of causal functions is closed under composition, prefix, suffix, and time reversal operations.*

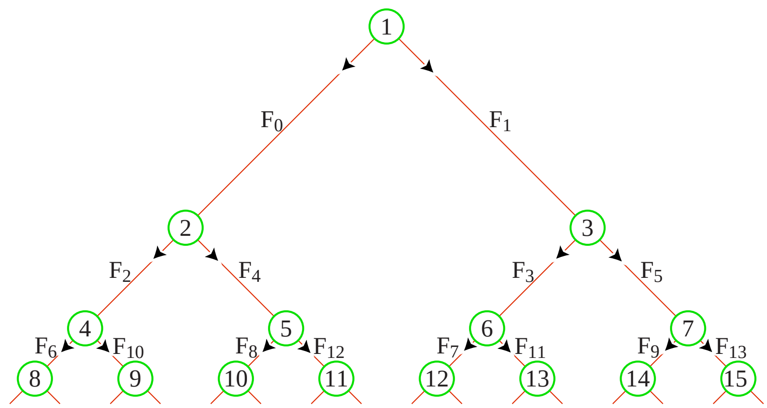

Fig. 1. Sequential Decision Tree, for the truth table $(F_{\mathbf{N}})$.

4 Universal Causal Machines

4.1 Sequential Decision Tree

Definition 5. *The* Sequential Decision Tree $sdt(f)$*, for computing causal* f*, is a complete infinite binary tree - fig. 1. A digital input* $x \in \mathbf{D}$*, specifies a unique path through the tree: start at the root, for cycle 0; at cycle* N*, move down, to the left if* $x_{\mathbf{N}} = 0$*, right otherwise. Arcs in the tree are labeled, in hierarchical order, by bits from the time reversed* $\mathrm{truth}(\widetilde{f})$*. Output* $y = f(x)$*, is the digital sequence of arc labels, along the path specified by input* x*.*

4.2 Sequential Multiplexer

A Digital Synchronous circuit **DSC** is obtained, by composing *primitive* components: the *register* **reg**, and Boolean (combinational, memoryless) operators. There is a restriction on composition: all *combinational* paths, through a chain

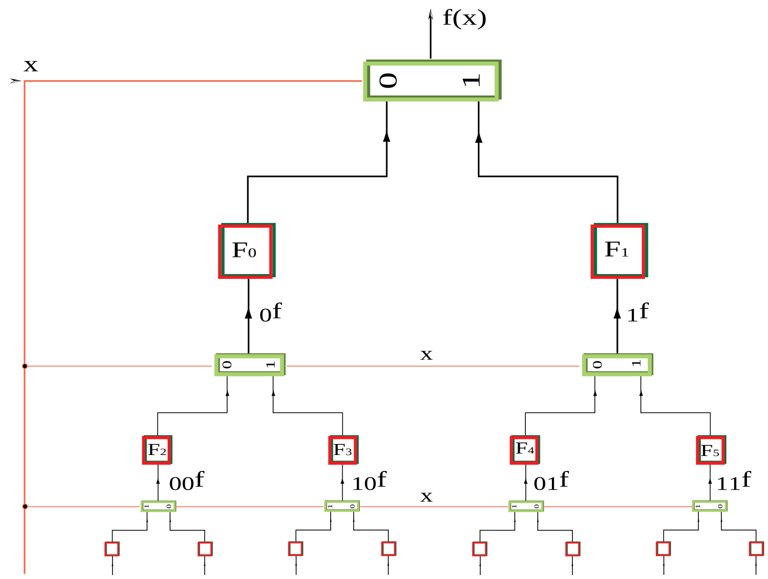

Fig. 2. Sequential multiplexer, for the truth table $(F_{\mathbf{N}})$.

of Boolean operators, must be *finite*. This implies that each feedback loop *must* contain, at least one memory element **reg** (positive feedback).

The operators $(\mathbf{reg}_0, \mathbf{reg}_1, \mathbf{mux})$ serve as a base, for the SDD procedure: registers $\mathbf{reg}_0(x) = \mathbf{reg}(x) = zx = 2x$, $\mathbf{reg}_1(x) = \neg z \neg x = 1 + 2x$, and multiplexer $\mathbf{mux}(c, b, a) = (c \cap b) + (\neg c \cap a) = c \cap (b \oplus a) \oplus a$.

The *sequential multiplexer* $SM(f)$, from [4], is shown in fig. 2. The registers in $SM(f)$, are labeled, 0 for $\mathbf{reg}_0$ and 1 for $\mathbf{reg}_1$, by truth(f), in *direct* order.

4.3 Share Common Expressions

The next step, in the infinite **SDD** construction [4], is to *share* all common sub-expressions, which appear in the process: the result is the *Sequential Decision Diagram* SDD(f), for SM - see fig. 4. Similarly, for SDT, we obtain the *Sequential Binary Automaton* SBA(f) - see fig. 3.

5 Finite Causal Function

The causal functions mentioned so far may all be realized by *finite* circuits, and finite state machines **FSM**, except for $\times$, $/$, $\otimes$, $\oslash$ and $\uparrow$, which are *infinite* [4].

Definition 6. *Digital sequence b is* 2-algebraic*, when $b(z) = \sum b_{\mathbf{N}} z^{\mathbf{N}}$ is algebraic over $\mathbf{F}_2(z)$. Let $\mathbf{A}_2$ denote the set of 2-algebraic sequences.*

$T(z) = \text{truth}(\text{zerotest})$ is 2-algebraic, as root of: $1 + T + z^2T^2 = 0 \pmod 2$.

Proposition 5. *Causal f is finite, if and only if the following equivalents hold:*

a f is computed by a finite circuit **DSC***;*
b f is computed by a finite state machine **FSM***;*
c prefix(f) is finite;
d suffix(f) is finite;
e truth(f) is 2-algebraic.

Proof: The equivalence between (a) and (b) is well-known. The equivalence between (b), (c) and (d) follows from classical automata theory [7].

The equivalence between (d) and (e) is established, through a result in the theory of *automatic sequences*. Call *2-automatic*, a sequence $a \in \mathbf{D}$, such that $a = \text{auto}(f)$, for some **FSM** f.

Theorem 3 (Christol, Kamae, Mendès France, Rauzy [11]).
A digital sequence is 2-automatic*, if and only if it is* 2-algebraic*.*

Combine Theorem 3 with Proposition 3, to complete the proof of Proposition 5, hence that of Theorem 1. ■

Proposition 6. *Finite causal functions, are closed, under composition, prefix, suffix, and time reversal.*

Theorem 4. *The class $\mathbf{A}_2$, of 2-algebraic sequences, is closed under:*

1. *Boolean operations $\neg, \cup, \cap$, and shifts z, z^-;*
2. *carry-free polynomial operations $\oplus, \otimes, \oslash$;*
3. *up-sampling, down-sampling, and time reversal;*
4. *application of any finite causal function, hence $+, -$.*

Proof: Boolean closure follows from Proposition 2. Polynomial manipulations show the closure under carry-free operations: $\oplus, \otimes, \uparrow$, and shifts. Item 3 follows from Theorem 6. A novel construction is given, for proving item 4. It implies, in particular, that $\mathbf{A}_2$ is closed under ordinary addition, and subtraction, *with* carries. We conjecture that $\mathbf{A}_2$ is also closed under multiplication $\times$, and division $/$. ■

5.1 Transcendental Numbers

If one interprets a digital sequence $x = x(z) = x(2)$ in base $\frac{1}{2}$, rather than 2 or z, one gets a *real number*: $x(1/2) \in \mathbf{R}$. To each causal f, associate the real number $\text{real}(f) = \text{truth}(f)(1/2) \in \mathbf{R}$.

Theorem 5 (Loxton, van der Poorten [12]). *If $a(z) \in \mathbf{A}_2$ is 2-algebraic, then, either $a(\frac{1}{2}) \in \mathbf{Q}$ is rational, or it is* transcendental*, in the usual sense over* $\mathbf{Q}$.

As a consequence, real(zerotest) $= 1.2656860360875726\cdots$ is transcendental, over $\mathbf{Q}$. Similarly, for real(booth) $= 0.6010761186771489\cdots$.

Up-sampling $y =\uparrow x$ is causal, and $y_{\mathbf{N}} = f_{\mathbf{N}}(x_0 \cdots x_{\mathbf{N}})$ is the middle bit: $y_{2\mathbf{N}} = 0$, and $y_{2\mathbf{N}+1} = x_{\mathbf{N}}$. The *middle bit sequence* is the truth table $M = \mathrm{truth}(\uparrow)$: $M = 01000000110011000000000000000000000011110000111100\cdots$. No finite circuit exists, to implement up-sampling [4]. It follows, from Theorem 1, that the middle bit series $M(z)$ is *transcendental* over $\mathbf{F}_2[z]$. Similarly for truth($\otimes$), and truth($\times$). It is not known, if real($\uparrow$) $= 0.5062255860470657\cdots$ is *transcendental* over $\mathbf{Q}$, or not; similarly for real($\otimes$) and real($\times$).

6 Finite SDD Procedure

For f causal and finite, define size(f) as the number of states, in the *minimal* **FSM** (see [7]), for computing f. For $F \in \mathbf{D}$, define $S = \mathrm{sample}(F)$, as

$$S = \{F\} \cup (z^- \downarrow S) \cup (z^- \downarrow z^- S),$$

where the least fixed point $S \in \wp(\mathbf{D})$, is a set of digital sequences.

Theorem 6. *Each of the following (equivalent statements), provides a* canonical *representation for f finite causal, with size(f) $= n$, and $F =$ truth(f).*

1. *sample(F) is finite, of size n.*
2. *SBA(f) is the minimal* **FSM** *for computing f, with n states.*
3. *SDD($\widetilde{f}$) is a finite* **DSC** *circuit, with n multiplexers, and at most $2n$ registers,* $\mathbf{reg}_0$ *or* $\mathbf{reg}_1$.
4. *$F =$ truth(f) is the* unique *2-algebraic solution, to the system quadra(f), made of n binary quadratic equations.*

This is established through an effective algorithm - *recursive sampling* - and data structure. In this extented abstract, we simply present the (computer generated) output from the procedure, for one example: Booth coding, as defined by (1), and where size(booth) $= 4$.

6.1 Recursive Sampling

For $f1 = \mathrm{truth(booth)}$, compute $\mathrm{sample}(f1) = \{f1, f2, f5, f11\}$:

$$\begin{aligned}
f1 &= 0100110011110000000001111111110000000011111111111\cdots\\
f2 &= 0101100001111000011111111100000000000000001111111\cdots\\
f5 &= 1001100111100000000011111111100000000111111111111\cdots\\
f11 &= 1011000011110000111111111000000000000000011111111\cdots
\end{aligned}$$

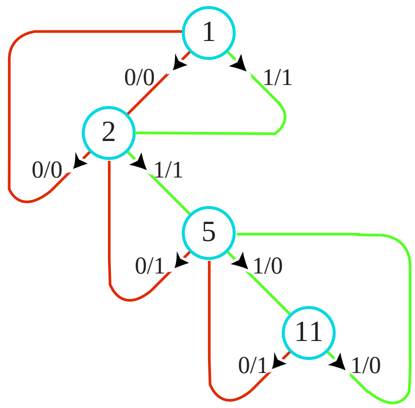

Fig. 3. The automaton SBA(booth), where $\mathrm{booth}(x) = (01) \oplus (x + (01))$.

6.2 SBA Procedure

6.3 Characteristic Circuit Polynomial

A *binary quadratic equation* has the form: $f = a + bz + z^2g^2 + z^3h^2 \pmod 2$, for $a, b \in \mathbf{F}_2$, and $f, g, h \in \mathbf{D}$. Truth tables in $\mathrm{sample}(\mathrm{booth}) = \{f1, f2, f5, f11\}$ are related by the following system of *binary quadratic equations*:

$$\begin{aligned} f1 &= z + z^2(1+z)f2^2, \\ f2 &= z + z^2 f1^2 + z^3 f5^2, \\ f5 &= 1 + z^2 f2^2 + z^3 f11^2, \\ f11 &= 1 + z^2(1+z)f5^2. \end{aligned}$$

Through *quadratic elimination*, derive $\mathrm{quad}(F)$:

$$\begin{aligned} \mathrm{quad}(\mathrm{booth}) = a + bF + c \uparrow^2 F + d \uparrow^4 F \pmod 2, \\ a = z + z^2 + z^3 + z^8 + z^{16} + z^{28} + z^{32}, \\ b = 1 + z + z^2 + z^3, \\ c = z^4(1 + z + z^2 + z^3 + z^4)^2, \\ d = z^{28}(1 + z + z^2 + z^3 + z^4)^4. \end{aligned}$$

Through *algebraic simplifications*, obtain the irreducible *characteristic polynomial* $\mathrm{poly}(\mathrm{booth})$, of which $F = \mathrm{truth}(\mathrm{booth}) = f1$ is the only root:

$$F = z + z^4 + z^5 + z^4(1 + z + z^2 + z^3)F^4 \pmod 2.$$

A *decimal* expression for $\mathrm{poly}(\mathrm{booth})$: $F = 50 + 240 \uparrow^2 F$.

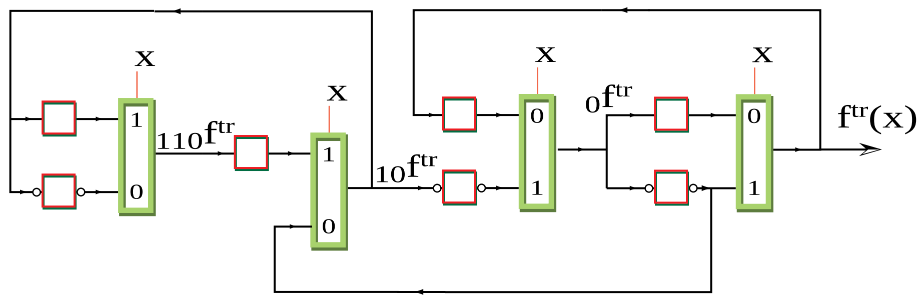

Fig. 4. The circuit SDD($\widetilde{\text{booth}}$).

6.4 SDD procedure

The circuit synthesized by the **SDD** procedure involves the *time reversed domain*. To keep the correspondence with Theorem 6.3, we show the circuit SDD($\widetilde{\text{booth}}$), in fig. 4. This circuit computes the function $\widetilde{\text{booth}}$, defined through time reversal in equation (1). For SDD(booth) $\rightleftharpoons$ SBA($\widetilde{\text{booth}}$), one finds 15 states.

7 Feed-forward Circuit

Proposition 7 (Feed-forward circuit). *The following are characteristic equivalents, for finite causal f to be free of* feed-back*:*

1. *SDD(f) is* acyclic*;*
2. *$F = truth(\widetilde{f}) \in \mathbf{P}2$ is ultimately periodic, with period length 2^b, for $b \in \mathbf{N}$;*
3. *$poly(\widetilde{f}) = a + (z^{2^b} - 1)F$, for $a \in \mathbf{N}$.*

Proposition 8 (Combinational circuit). *The following are characteristic equivalents, for finite causal f, with i inputs, to be* memoryless*:*

1. *SDD(f) contains no register;*
2. *$size(\widetilde{f}) = 1$;*
3. *$F = truth(\widetilde{f}) \in \mathbf{P}2$ is periodic, i.e. $-1 \le F(2) \le 0$, with period length $2^{i'}$, for some integer $i' \le i$;*
4. *$poly(\widetilde{f}) = a + (z^{2^{i'}} - 1)F$, for some integer $a < 2^{2^{i'}}$.*

For Boolean functions, the **SDD** procedure is the same as the *Binary Decision Diagrams* **BDD** procedure, from [13].

8 Appendix

We use 14 operators, from Binary Algebra: 5 unary operations $\{\neg, z, z^-, \uparrow, \downarrow\}$, and 9 binary operations $\{\cup, \oplus, \cap, \otimes, \oslash, +, -, \times, /\}$. The binary operators are listed here in order of increasing syntactic precedence, so as to save parentheses.

– $\langle \mathbf{D}, (0), (1), \neg, \cup, \cap \rangle$ is a Boolean algebra;
  $\langle \mathbf{D}, (0), (1), \oplus, \cap \rangle$ is a Boolean ring: $a = a \cap a$, $0 = a \oplus a$ (see [8]).

$$\begin{aligned} a &= z^{-} z a, \\ z z^{-} a &= a \cap -2, \\ \neg z^{-} a &= z^{-} \neg a, \\ \neg z a &= 1 + z \neg a, \\ z^{-}(a \odot b) &= z^{-} a \odot z^{-} b, \text{ for } \odot \in \{\cup, \cap, \oplus\}, \\ z(a \odot b) &= z a \odot z b, \text{ for } \odot \in \{\cup, \cap, \oplus, +, -\}, \\ z(a \odot b) &= z a \odot b = a \odot z b, \text{ for } \odot \in \{\times, \otimes\}, \end{aligned}$$

– $\langle \mathbf{D}, 0, 1, \oplus, \oplus, \otimes \rangle$ is an *Integral Domain*, i.e. a commutative ring without divisor of 0. An element $a \in \mathbf{D}$ has a (polynomial) inverse $1 \oslash a$, such that $a \otimes (1 \oslash a) = a$, if and only if a is *odd* $(1 = a(0))$: $1 \oslash (1 + zb) = \bigoplus (zb)^{\mathbf{N}}$.
– $\langle \mathbf{D}, 0, 1, +, -, \times \rangle$ is an Integral Domain.

$$\begin{aligned} \neg a &= -a - 1, \\ a + b &= (a \cup b) + (a \cap b) \\ &= (a \oplus b) + z(a \cap b), \\ a + b &= a \cup b = a \oplus b \text{ iff } a \cap b = 0, \\ 1/(1 - 2b) &= \textstyle\sum (2b)^{\mathbf{N}} = \prod (1 + (2b)^{\mathbf{N}}). \end{aligned}$$

–
$$\begin{aligned} \uparrow a &= a(z^2) = a \otimes a, \\ a &= \downarrow\uparrow a, \\ a &= \uparrow\downarrow a + z \uparrow\downarrow z^{-} a, \\ \neg \downarrow a &= \downarrow \neg a, \\ \neg \uparrow a &= (01) \cup \uparrow \neg a, \\ \downarrow z^2 a &= z \downarrow a, \\ \uparrow z a &= z^2 \uparrow a, \\ \downarrow (a \odot b) &= \downarrow a \odot \downarrow b, \text{ for } \odot \in \{\cup, \cap, \oplus\}, \\ \uparrow (a \odot b) &= \uparrow a \odot \uparrow b, \text{ for } \odot \in \{\cup, \cap, \oplus, \otimes, \oslash\}. \end{aligned}$$

We list the known closure properties, for operators and sub-structures, in Binary Algebra.

– ${\mathbf{F}_2}^{\mathbf{N}}$ is closed, under $\{\neg, \cup, \oplus, \cap, z, \uparrow, \otimes, \oslash, +, -, \times, /\}$.
– $\mathbf{N}$ is closed, under $\{\cup, \oplus, \cap, z, z^{-}, \uparrow, \downarrow, \otimes, +, \times\}$.
– $\mathbf{Z}$ is closed, under $\{\neg, \cup, \oplus, \cap, z, z^{-}, \downarrow, +, -, \times\}$.
– $\mathbf{P}2$ is closed, under $\{\neg, \cup, \oplus, \cap, z, z^{-}, \uparrow, \downarrow, \otimes, +, -, \times\}$.
– $\mathbf{A}_2$ is closed under $\{\neg, \cup, \oplus, \cap, z, z^{-}, \uparrow, \downarrow, \otimes, \oslash, +, -\}$. The closure under carry-free product is shown in [14]. It is shown in [15] that $\mathbf{A}_2$ is not closed under multiplication $\times$ with carries.
– $\mathbf{P}$, Z_2 and $\mathbf{Z}_2$ are closed, under all 14 operations.

References

1. C. Mead, *ANALOG VLSI AND NEURAL SYSTEMS*, Addison-Wesley, 1989.
2. N. Weste, K. Eshragian, *Principles of CMOS VLSI Design*, Addison-Wesley, 1985.

3. J. Vuillemin, P. Bertin , D. Roncin, M. Shand, H. Touati, P. Boucard *Programmable Active Memories: Reconfigurable Systems Come of Age*, IEEE Trans. on VLSI, Vol. 4, NO. 1, pp. 56-69, 1996.
4. J. Vuillemin, *On circuits and numbers*, *IEEE Trans. on Computers*, 43(8): pp. 868–879, 1994.
5. Gérard Berry, *private communication*, 1995.
6. Klaus Winkelman, *private communication*, 1996.
7. S. Eilenberg, *Automata, Languages, and Machines*, Academic Press, 1974.
8. W. J. Gilbert, *Modern Algebra with Applications*, A Wiley-Interscience publication, John Wiley & Sons, New York, 1976.
9. J. P. Allouche, *Automates finis en théorie des nombres*, in *Expositiones Mathematicae*, pp. 239–266, 5 (1987).
10. M. Mendès France, *Some applications of the theory of automata*, in *Prospects of Math. Sci.*, World Sci. Pub., pp. 127–144, 1988.
11. G. Christol, T. Kamae, M. Mendès France, G. Rauzy, *Suites algèbriques, automates et substitutions*, in *Bull. Soc. Math. France*, 108: pp. 401–419, 1980.
12. J.H. Loxton, A.J. van der Poorten, *Arithmetic properties of the solutions of a class of functional equations*, *J. Reine Angew. Math.*, 330, pp. 159–172, 1982.
13. R. E. Bryant, *Graph-based Algorithms for Boolean Function Manipulation*, in *IEEE Trans. on Computers*, 35:8: pp. 677–691, 1986.
14. J.P. Allouche, J. Shallit, *The ring of k-regular sequences*, in *Theoret. Comput. Sci.*, 98 (1992) pp. 163–187.
15. S. Lehr, J. Shallit, J. Tromp, *On the vector space of the automatic reals*, in *Theoret. Comput. Sci.*, 163 (1996) pp. 193–210.