On Circuits and Numbers

Jean E. Vuillemin
Digital Equipment Corp.,
Paris Research Laboratory (PRL),
85 Av. Victor Hugo.
92563 Rueil-Malmaison, Cedex France.

October 13, 1993

Abstract

We establish new, yet intimate relationships between the *2adic integers* ${}_2\mathbf{Z}$ from arithmetics and *digital circuits*, finite and infinite, from electronics.

(a) Rational numbers with an odd denominator correspond to output only synchronous circuits.

(b) Bit-wise 2ading mappings correspond to combinational circuits.

(c) On-line functions, $\forall n \in \mathbf{N}, x \in {}_2\mathbf{Z} : \ f(x) = f(x \bmod 2^n) \pmod{2^n}$, correspond to synchronous circuits.

(d) Continuous functions ${}_2\mathbf{Z} \mapsto {}_2\mathbf{Z}$, correspond to circuits with output enable.

The proof is obtained by constructing *synchronous decision diagrams* SDD. They generalize to sequential circuits what classical BDD constructs achieve for combinational circuits.

From simple identities over ${}_2\mathbf{Z}$, we derive both classical and *new* bit-serial circuits for computing: $\{+, -, \times, 1/(1+2x), \sqrt{1+8x}\}$. The *correctness* of each circuit directly follows from the 2adic definition of the corresponding operator.

All but the adders $(+, -)$ above are *infinite*. Yet, the use of *reset* signals reduces all previously infinite operators to *finite* circuits.

We indicate which features from this abstract 2adic semantics help synthesize some of the largest synchronous hardware designs ever implemented, through the 2Z language.

Keywords: Synchronous, sequential, combinational circuit semantics and synthesis. Bit-serial 2adic arithmetics, arbitrary precision. Periodic numbers. Bit-wise, on-line, continuous functions. Enable, reset. Programmable Active Memories.

1 Introduction

Modern electronic circuits fall in two categories, *analog* and *digital*.

The dynamic analysis of analog circuits involves physical parameters, such as currents and voltages, whose value $v_t \in \mathbf{R}$ vary continuously with *real time* $t \in \mathbf{R}$. Carver Mead's book [M89] provides an excellent introduction to analog circuits.

Digital circuits are characterized by a finite number of physical variables, whose value $v_t \in \mathbf{B}$ is identified with either zero or one through discretization: say 0 when voltage $< 1V$ and 1 when voltage $> 2V$.

Digital circuits may further be classified as *asynchronous* or *synchronous*. In asynchronous circuits, communication of values between the constituting units may occur at any real instant $t \in \mathbf{R}$.

Within synchronous circuits, all variables are sampled at integer multiples ($t = n\Delta t$ for $n \in \mathbf{N}$) of the same global clock period Δt. Setting $\Delta t = 1$ through a suitable choice of the physical units thus allows to identify *digital time* $t \in \mathbf{N}$ with the set of natural numbers.

The present work is exclusively concerned with synchronous circuits, which we characterize mathematically as follows:

Definition 1 (Digital Synchronous Circuit) *In a synchronous circuit C, the value of any variable $v \in \mathcal{V}(C)$ is a* bit $v_t \in \mathbf{B} = \{0, 1\}$ *which may only change at integer time $t \in \mathbf{N} = \{0, 1, 2, \cdots\}$:*

$$\forall v \in \mathcal{V}(C), t \in \mathbf{R}, \exists n = \lfloor t \rfloor \in \mathbf{N} : \quad v_t = v_n \in \mathbf{B}.$$

Here, $\lfloor t \rfloor$ denotes the largest integer which is less than or equal to t. A direct consequence of definition 1 is that all delays in a digital circuit are *exact integers*. In particular, *combinational* circuits have *zero delay*: the output response to changes in the inputs is instantaneous; time plays no part in their mathematical analysis. With *synchronous* (a. k. a. sequential) circuits, changes in the digital values of variables are equally instantaneous; they precisely occur at digital times $t \in \mathbf{N}$. Every physical implementation of such mathematical circuits has (hopefully very) small delays, but (certainly) not zero delays. As a consequence, the physical behavior matches its mathematical idealization only when operated on with a clock whose period $\Delta t > \delta$ exceeds the maximum physical delay δ (*critical path*).

Synchronous circuits naturally operate upon *infinite* binary sequences: within any computation performed by circuit C, each variable $v \in \mathcal{V}(C)$ takes on consecutive binary values $v_0, v_1, \cdots v_t, \cdots \in \mathbf{B}$ as digital time progresses through the natural numbers $t \in \mathbf{N}$. Synchronous circuits map infinite binary sequences, representing the successive input values at each clock tick $t \in \mathbf{N}$, into infinite binary sequences, representing the corresponding output values.

1.1 Hensel's Numbers

Infinite binary sequences have a rich mathematical structure, namely that of the *2adic integers* ${}_2\mathbf{Z}$, whose algebraic properties are presented in section 2. Set ${}_2\mathbf{Z}$ supports most[1] of the usual operations over the ordinary integers $\mathbf{Z} = \{\cdots, -2, -1, 0, 1, 2, \cdots\}$ hence the name. In addition, it supports all operations over the subsets $2^{\mathbf{N}}$ of the natural numbers. In short, the 2adic integers ${}_2\mathbf{Z}$ are both a *boolean algebra* $(\emptyset, \overline{\ }, \cup, \cap)$ and an integer *ring* $(0, +, -, 1, \times)$ (see proposition 1).

The p-adic integers ${}_p\mathbf{Z}$ were introduced around 1900 by K. Hensel [H13], for each prime $p \in \mathbf{N}$. They play a central role in arithmetic (see [A75] and [K77]). Such numbers are obtained by extending indefinitely the ordinary base p representation, as computed by the rule:

$$\mathcal{B}_p(n) \quad \Rightarrow \quad (n \cdot|\cdot p)\, \mathcal{B}_p(n \div p). \tag{1}$$

We use $n \div p$ to represent the *quotient* and $n \cdot|\cdot p$ the *remainder* in the integer division of n by $p \neq 0$, so $n = p \times (n \div p) + (n \cdot|\cdot p)$ with $0 \leq (n \cdot|\cdot p) < p$. For example, we compute the infinite binary ($p = 2$) representation of decimal number 22=2+4+16 by:

$$\mathcal{B}_2(22) \stackrel{*}{\Rightarrow} 011010\cdots0\cdots = {}_201101(0).$$

In the above equality, subscript ${}_2$ indicates the representation base 2 as well as the reading order, from low order bits to high order bits; the (0) denotes an infinite (periodic) sequence of zeroes. Similarly, we find:

$$\begin{array}{rrcll} & \mathcal{B}_2(-7) & \stackrel{*}{\Rightarrow} & 100111\cdots1\cdots & = {}_2100(1), \\ & \mathcal{B}_2(\frac{-4}{5}) & \stackrel{*}{\Rightarrow} & 0011\cdots0011\cdots & = {}_2(0011), \\ \text{and} & \mathcal{B}_2(\frac{1}{3}) & \stackrel{*}{\Rightarrow} & 11010\cdots10\cdots & = {}_21(10). \end{array}$$

In general, we represent the *semantic* value of each variable v within each synchronous circuit by its infinite binary expansion as a 2adic number:

$${}_2v_0\cdots v_t\cdots \;=\; \sum_{t\in\mathbf{N}} v_t 2^t.$$

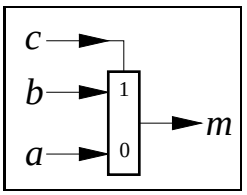
[1] but not all: we loose integer comparison, integer division, and exact division by an even number;

1.2 Synchronous Circuits

Synchronous circuits can all be built from two atomic gates, the *multiplexor* and the *register*, also known as flip-flop.

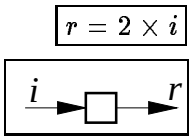
Definition 2 (The Multiplexor $? \in {}_2\mathbf{Z}^3 \to {}_2\mathbf{Z}$**)** *mux*
The value of output $m_t \in \mathbf{B}$ *at clock tick* $t \in \mathbf{N}$ *is determined from the three input values* $c_t, b_t, a_t \in \mathbf{B}$ *by:*

$$m_t = ?(c_t, b_t, a_t) = \begin{cases} b_t & \text{if } c_t = 1, \\ a_t & \text{if } c_t = 0. \end{cases}$$



Note that: $?(c, b, a) = (c \wedge b) \vee (\neg c \wedge a) =$ if c then b else a. Together with $0 = {}_2(0)$, the electrical ground and $-1 = {}_2(1)$, the electrical power supply whose bits are one at all times, the multiplexor provides a *basis* for boolean algebra: $\neg b = ?(b, 0, -1)$, $a \vee b = ?(a, a, b)$ and $a \wedge b = ?(a, b, a)$.

Definition 3 (The Register $2\times \in {}_2\mathbf{Z} \to {}_2\mathbf{Z}$**)** *reg*
The output r_t *of a register is 0 at initial time:* $r_0 = 0$*; for* $t \geq 1$*, it is equal to the value* i_{t-1} *of its input, sampled at the previous clock tick:* $r_t = i_{t-1}$.



As we represent the input sequence by the 2adic integer $i = \sum_{t\in\mathbf{N}} i_t 2^t$, the 2adic integer $o = \sum_{t\in\mathbf{N}} o_t 2^t$ representing the output of the register is equal to $r = 2 \times i$.

Complex synchronous circuits are constructed by wiring together a number of muxes and registers. All registers are *synchronous* in that they share the same global clock signal. In order to always ensure that our circuits have a well defined *synchronous semantics* (definition 1), it is sufficient to require that all purely combinational pathes through the muxes in our circuits have a *finite* length. The purpose of this restriction is to eliminate from consideration cyclic combinational structures, such as $u = ?(u, 0, 1)$ whose value is *undefined* in the boolean domain.

Definition 4 *A* synchronous circuit $C \in \mathcal{C}(?, 2\times)$ *is a set* $\mathcal{V} = \mathcal{V}(C) = \mathcal{I} \cup \mathcal{M} \cup \mathcal{R}$ *of digital variables composed of:*

- *Inputs* $\mathcal{I} = \{i[0], \cdots, i[i-1]\}$ *in* finite *number* $i = |\mathcal{I}|$*; they may take arbitrary 2adic integer values.*
- *Muxes* $\mathcal{M} = \{m[0], \cdots, m[n] \cdots\}$*; each is defined by a mux equation:*

$$\forall m \in \mathcal{M}, \exists c, a, b \in \mathcal{V}(C) : \quad m = ?(c, b, a).$$

The mux-ordering $\prec$ induced over the variables $\mathcal{V}(C)$ by

$$m = ?(c, b, a) \text{ implies } c \prec m,\ b \prec m \text{ and } a \prec m,$$

must be well-founded: *every descending chain $v[1] \succ \cdots \succ v[n] \succ \cdots$ is* finite.

- *Registers $\mathcal{R} = \{r[0], \cdots, r[n], \cdots\}$; each is defined by a* reg *equation:*

$$\forall r \in \mathcal{R}, \exists i \in \mathcal{V}(C) :\ r = 2 \times i.$$

The outputs $\mathcal{O} = \{o_0, \cdots, o_{o-1}\} \subseteq \mathcal{V}(C)$ form a finite *subset $o = |\mathcal{O}|$ of the variables $\mathcal{V} = \mathcal{I} \cup \mathcal{M} \cup \mathcal{R}$.*

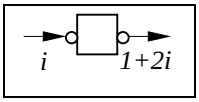
Note that definition 4 allows for infinite as well as finite circuits. When the graph of a synchronous circuit contains loops, it is straightforward to verify from the well-founded mux ordering that each loop through the circuit must traverse at least one register.

It also follows that each *finite* synchronous circuit has a well defined *critical path δ*. This ensures that such circuits may be physically implemented. They operate reliably with any synchronous clock of period $\Delta t > \delta$ greater than the circuit's largest delay. Such properties need not always be true of our forthcoming infinite synchronous circuits.

From an arbitrary assignment of boolean values to the inputs, provided by the 2adic integers $I[j] = \sum_{t \in \mathbf{N}} i_t[j] 2^t$ (for $0 \leq j < i = |\mathcal{I}|$), we may compute, from time $t = 0, 1, 2, \cdots$ and so on, the values of each mux and reg in circuit $C \in \mathcal{C}(?, 2\times)$. The results are the 2adic integers $O[j] = \sum_{t \in \mathbf{N}} o_t[j] 2^t$ (for $0 \leq j < o = |\mathcal{O}|$) which represent the circuit's output responses.

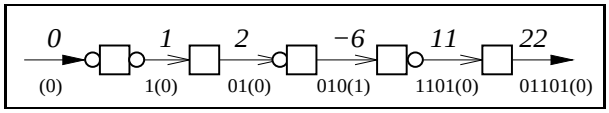
Example 1 (Register with initial value 1)
The circuit to the right computes $o = 1 + 2 \times i$

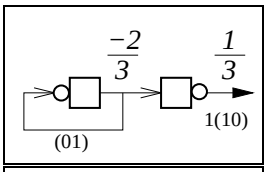
$$\begin{aligned} \{\overline{i} &= ?(i, 0, -1), \\ \textit{where} \quad 2\overline{i} &= 2 \times \overline{i}, \\ o &= ?(2\overline{i}, 0, -1)\} \end{aligned}$$

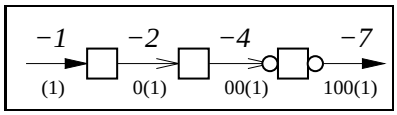


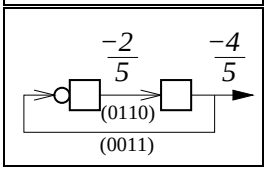
In the following examples of synchronous circuits with constant inputs (or no input), we are able to label each variable by its semantic value: it is given both in decimal and in binary, with low order bit first and periodic part in parenthesis.

Example 2 *The numbers $22, \frac{1}{3}$ and $-7, -\frac{4}{5}$ as circuits:*









In our schemas: circles denote inverters and squares denote registers with initial value *zero*. A register with inverters on both sides (example 1) is recognized as having initial value *one*: it computes $1 + 2x$.

1.3 Summary of Results

Section 2 regroups some definitions and elementary results about the 2adic integers: *where boolean algebra meets binary arithmetics.* Proposition 2 confirms that, as illustrated in examples 2, we can always label finite synchronous circuits which have *constant* inputs by rational numbers with an odd denominator.

Section 3 classifies the mappings ${}_2\mathbf{Z} \mapsto {}_2\mathbf{Z}$ over 2adic integers into three inclusive sets: bit-wise, on-line and continuous functions.

Section 4 gives classical results about bit-wise functions. Through the BDD construction of [A78] and [B86], we characterize them as finite combinational (a. k. a. memoryless) circuits in theorem 1.

The *synchronous decision diagrams* SDD introduced in section 5 give a strict generalization of the *binary decision diagrams* BDD algorithm to *sequential* circuits. It is used to prove in theorem 2 that a function is on-line if and only if it may be computed by some synchronous circuits. The SDD algorithm is a novel and powerful technique for the synthesis of synchronous circuits. We discuss some of its applications and limitations.

Section 6 shows that all *continuous* functions over the 2adic integers may be realized by synchronous circuits with *output enable* (theorem 3).

From simple identities over 2adic integers, we derive in section 7 both classical and *new* bit-serial circuits for computing:

$$\{+, -, \times, 1/(1+2x), \sqrt{1+8x}\}.$$

In each case, the *correctness* of the proposed implementation follows directly from the 2adic definition of the corresponding operator.

All but the adders $\{+, -\}$ above are *infinite*. We use reset signals in order to pipeline finite integer computations through arbitrary 2adic networks (theorem 4). In this context, all previously *infinite* arithmetic circuits become *finite*, in section 8.

We conclude in section 9 with some applications of the present 2adic *semantics* to synchronous design *synthesis* in the language 2Z.

2 The 2adic integers ${}_2\mathbf{Z}$

While Hensel's construction applies for any prime p, we need only concern ourselves with the case $p = 2$ of the 2adic integers ${}_2\mathbf{Z}$ for the purpose of studying digital circuits. The *arithmetic* properties of ${}_2\mathbf{Z}$ are (almost) similar to those of the p-adic integers ${}_p\mathbf{Z}$ for $p > 2$. The *logical* properties of ${}_2\mathbf{Z}$ are unique, a fact which we emphasize in the following definition.

Definition 5 *A 2adic integer $B \in {}_2\mathbf{Z}$ is the limit of three equivalent infinite sequences*

$$B = \lim_{n \mapsto \infty} {}_2b_0 \cdots b_n = \lim_{n \mapsto \infty} b(n) = \lim_{n \mapsto \infty} b\{n\},$$

respectively composed, for each $n \in N$, of:

1. *bits $b_n \in \mathbf{B}$, with* $b_n = (B \div 2^n) \cdot|\cdot 2$;
2. *natural numbers $b(n) \in \mathbf{N}$, with* $b(n) = B \cdot|\cdot 2^{n+1} = \sum_{0 \leq k \leq n} b_k 2^k$;
3. *finite integer sets $b\{n\} \subseteq \{0, 1, \cdots, n\}$, with* $b\{n\} = \{k \leq n : \ b_k = 1\}$.

Let us make explicit the meaning of the word *limit* in definition 5, by introducing a *distance* over binary sequences.

Definition 6

- *The* valuation $v_2(b) \in \mathbf{N}$ *of a 2adic integer $b \in {}_2\mathbf{Z}$ is equivalently, the:*
  1. *index of the first non-zero bit in the binary representation of b;*
  2. *largest power of two which divides $b(n)$, for $n \in \mathbf{N}$;*
  3. *smallest element in the set $b\{n\}$, for $n \in \mathbf{N}$.*
- *The* norm *of a 2adic integer $x \in {}_2\mathbf{Z}$ is defined by ${}_2|x| = 2^{-v_2(x)}$.*
- *The* distance *between 2adic integers $x, y \in {}_2\mathbf{Z}$ is the norm of their difference: ${}_2|x - y|$.*

Note that $v_2(0) = \infty$ so ${}_2|x| = 0$ if and only if $x = 0$, and $0 < {}_2|x| \leq 1$ for $x \neq 0$. Norm ${}_2|x|$ is related to the (soon to be defined) sum and product by:

$$\begin{array}{ll} (i) & {}_2|x + y| \leq max\{{}_2|x|, {}_2|y|\}; \\ (ii) & {}_2|x \times y| = {}_2|x| \times {}_2|y|. \end{array}$$

Property (i), which is characteristic of *ultra-metric* norms (see [K77]) is *stronger* than the corresponding classical triangle inequality for real numbers $\mathbf{R}$: $|x + y| \leq |x| + |y|$.

It follows from definition 6 of the distance in ${}_2\mathbf{Z}$, that the *finite* approximants ${}_2b_0 \cdots b_n = b(n) = b\{n\}$ converge to number $B \in {}_2\mathbf{Z}$ according to:

$$\forall n \in \mathbf{N}: \ {}_2|B - {}_2b_0 \cdots b_n| = {}_2|B - b(n)| = {}_2|B - b\{n\}| \leq 2^{-n+1}.$$

As a consequence, each infinite binary sequence $A = (a_0 \cdots a_n \cdots)$ with $a_n \in \mathbf{B}$ for $n \in \mathbf{N}$ is *equal* to the *unique* 2adic integer $A = \sum_{n\in\mathbf{N}} a_n 2^n$ of which it is the base 2 representation (rule (1) where $p = 2$). Two infinite binary sequences A, B are *equal* $A = B$ when ${}_2|A - B| = 0$. This is equivalent to each of the following:

$$\forall n \in \mathbf{N}: \ a_n = b_n \Leftrightarrow \forall n \in \mathbf{N}: \ a(n) = b(n) \Leftrightarrow \forall n \in \mathbf{N}: \ a\{n\} = b\{n\}.$$

This lets us use the following notations for representing each $B \in {}_2\mathbf{Z}$:

$$B \ = \ {}_2b_0 \cdots b_n \cdots \ = \ \sum_{n\in\mathbf{N}} b_n 2^n = \bigvee_{n\in\mathbf{N}} b(n) \ = \bigcup_{n\in\mathbf{N}} b\{n\}.$$

One may choose either of the proposed representations - bits or integers or sets - in order to introduce operations over ${}_2\mathbf{Z}$:

Definition 7 *Let* $A = \bigvee_{n\in\mathbf{N}} a(n) = \bigcup_{n\in\mathbf{N}} a\{n\}$ *and* $B = \bigvee_{n\in\mathbf{N}} b(n) = \bigcup_{n\in\mathbf{N}} b\{n\}$ *be 2adic integers* $A, B \in {}_2\mathbf{Z}$. *We define the operations:*

- *not* $\neg A = \bigcup_{n\in\mathbf{N}}\{k \leq n: \ k \notin a\{n\}\}$,
- *or* $A \vee B = \bigcup_{n\in\mathbf{N}}(a\{n\} \cup b\{n\})$,
- *and* $A \wedge B = \bigcup_{n\in\mathbf{N}}(a\{n\} \cap b\{n\})$.
- *plus* $A + B = \bigvee_{n\in\mathbf{N}}(a(n) + b(n)) \cdot|\cdot 2^{n+1}$,
- *minus* $A - B = \bigvee_{n\in\mathbf{N}}(a(n) - b(n)) \cdot|\cdot 2^{n+1}$,
- *times* $A \times B = \bigvee_{n\in\mathbf{N}}(a(n) \times b(n)) \cdot|\cdot 2^{n+1}$.

It follows from elementary set theory and arithmetic modulo 2^{n+1} that:

Proposition 1 (Structure of ${}_2\mathbf{Z}$)

1. *The 2adic integers* $({}_2\mathbf{Z}, \neg, \vee, \wedge)$ *form a* boolean algebra*, isomorphic to* $(2^{\mathbf{N}}, \overline{\ }, \cup, \cap)$*. It contains* all *finite boolean algebras* $\mathbf{Z} \cdot/\cdot 2^n$ *for* $n \in \mathbf{N}$.

2. *The 2adic integers* $({}_2\mathbf{Z}, +, -, \times)$ *form a* ring *which contains the ordinary integers* $\mathbf{Z}$ *and the odd denominator rationals* $\mathbf{Z}/1+2\mathbf{N}$ *as proper sub-rings.*

Let us further characterize the set inclusions $\mathbf{Z} \subset \mathbf{Z}/1+2\mathbf{N} \subset {}_2\mathbf{Z}$, both arithmetically and in terms of synchronous circuits having *constant* inputs.

Proposition 2 *Assertions (i), (ii), (iii) and (iv) are equivalent:*

(i) *A 2adic integer* $B = z \in {}_2\mathbf{Z}$ *is an ordinary integer* $B \in \mathbf{Z}$.

(ii) $\exists z \in \mathbf{Z}, \forall k \in \mathbf{N} : \ B = z \pmod{2^{k+1}}$.

(iii) *The binary representation of* $B = {}_2b_0 \cdots b_{l-1}(b_l)$ *is* ultimately constant*:* $\exists l \in \mathbf{N}, \forall k \geq l : \ b_k = b_l$*; here,* $l+1$ *is the (ordinary) binary length of* $z \in \mathbf{Z}$*, in two's complement representation.*

(iv) $B = \sum_{t \in \mathbf{N}} b_t 2^t$ *is the output of some* finite acyclic *synchronous circuit having constant inputs, either* $0 = {}_2(0)$ *or* $-1 = {}_2(1)$.

Assertions (v), (vi), (vii) and (viii) are equivalent:

(v) *A 2adic integer* $B \in {}_2\mathbf{Z}$ *is an odd denominator rational* $B = \frac{z}{1+2d} \in \mathbf{Z}/1+2\mathbf{N}$.

(vi) $\exists z \in \mathbf{Z}, d \in \mathbf{N}, \forall k \in \mathbf{N} : \ B \times (1+2d) = z \pmod{2^{k+1}}$.

(vii) *The binary representation of* $B = {}_2b_0 \cdots b_{i-1}(b_i \cdots b_{i+p-1})$ *is* ultimately periodic*:* $\exists i \in \mathbf{N}, p \in \mathbf{N}+1, \ \forall k \geq i : \ b_k = b_{k+p}$.

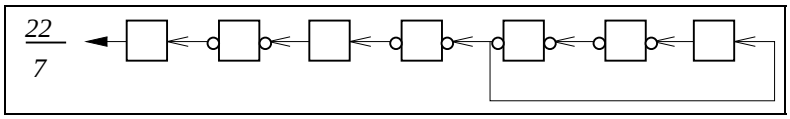
(viii) $B = \sum_{t \in \mathbf{N}} b_t 2^t$ *is the output of some* finite *synchronous circuit with constant inputs.*

The proof of this result can be found in the appendix section 11. The circuits introduced there are precisely those constructed by the SDD procedure (forward algorithm 2, section 5), upon constant (arity=0) input $f() = \frac{z}{1+2n} \in \mathbf{Z}/1+2\mathbf{N}$.

It establishes a direct correspondence between the ultimately periodic binary representation of an odd denominator rational, such as

$$\frac{22}{7} = {}_2 0101(110),$$

and its realization by a finite synchronous circuit containing exactly *one* loop, such as:



3 Mappings over 2adic integers

Computable functions over infinite binary sequences ${}_2\mathbf{Z}$ may be split into three classes.

Definition 8 *A (unary) function* $f \in {}_2\mathbf{Z} \mapsto {}_2\mathbf{Z}$ *over the 2adic integers mapping* $x = \sum_{t\in\mathbf{N}} x_t 2^t$ *into* $f(x) \in {}_2\mathbf{Z}$ *is:*

1. bit-wise *when* $f(x) = \sum_{t\in\mathbf{N}} g(x_t)2^t$ *for some* $g \in \mathbf{B} \mapsto \mathbf{B}$*;*
2. on-line *when* $f(x) = \sum_{t\in\mathbf{N}} g_t(x_0, \cdots, x_t)2^t$ *for some* $g_t \in \mathbf{B}^{t+1} \mapsto \mathbf{B}$*;*
3. continuous *when* $f(x) = \sum_{t\in\mathbf{N}} g_t(x_0, \cdots, x_{m(t)})2^t$ *for some* $m(t) \in \mathbf{N}$ *and* $g_t \in \mathbf{B}^{m(t)} \mapsto \mathbf{B}$.

It is clear from this definition that each bit-wise function is on-line, and each on-line function is continuous. Definition 8 is given for unary (single input) functions; it extends to n-ary (n inputs) functions in a straightforward manner. For say, two arguments $x, y \in {}_2\mathbf{Z}$, it becomes:

1. $f(x, y) = \sum_{t\in\mathbf{N}} g(x_t, y_t)2^t$ for some $g \in \mathbf{B}^2 \mapsto \mathbf{B}$;
2. $f(x, y) = \sum_{t\in\mathbf{N}} g_t(x_0, \cdots, x_t, y_0, \cdots, y_t)2^t$ for some $g_t \in \mathbf{B}^{2t+2} \mapsto \mathbf{B}$;
3. $f(x, y) = \sum_{t\in\mathbf{N}} g_t(x_0, \cdots, x_{m(t)}, y_0, \cdots, y_{m(t)})2^t$ for some $m(t) \in \mathbf{N}$ and $g_t \in \mathbf{B}^{2m(t)} \mapsto \mathbf{B}$.

Let us comment on the terms of definition 8. For ease of notation, return to the unary case.

1. Bit-wise functions will naturally be associated with combinational (no register) synchronous circuits in the next section 4. The multiplexor is bit-wise; so are the three boolean operations and their compositions.

2. From definition 6 of the distance between 2adic integers, we see that a function is on-line if and only if it is a *norm contraction*:

$$\forall x, y \in {}_2\mathbf{Z} : \qquad {}_2|f(x) - f(y)| \leq {}_2|x - y|. \tag{2}$$

   We prove in theorem 2 that on-line functions are precisely those computed by synchronous circuits. The register and the arithmetic operations $+, -, \times$ are all on-line; none is bit-wise.

3. Expressed in terms of distances, our notion of continuity is equivalent to the following classical definition:

$$\begin{array}{c} \forall n \in \mathbf{N}, \forall x \in {}_2\mathbf{Z}, \exists m = m(x,n) \in \mathbf{N}, \forall y \in {}_2\mathbf{Z} : \\ {}_2|x - y| < 2^{-n} \text{ implies } {}_2|f(x) - f(y)| < 2^{-m}. \end{array} \tag{3}$$

   Heine's theorem (see e.g. [A75]) shows that a function is *continuous* over a topologically compact set (namely the whole of ${}_2\mathbf{Z}$) if and only if it is *uniformly continuous*. That is to say, (3) is equivalent to:

$$\begin{array}{c} \forall n \in \mathbf{N}, \exists m = m(n) \in \mathbf{N}, \forall x, y \in {}_2\mathbf{Z} : \\ {}_2|x - y| < 2^{-n} \text{ implies } {}_2|f(x) - f(y)| < 2^{-m}. \end{array} \tag{4}$$

Example 3 (below) presents two functions, the Peano projections, which are *continuous* yet *not on-line:* it will follow that each may be realized by a synchronous circuit with output enable (see section 6); neither may be realized by any synchronous circuit.

The test for zero function $z \in {}_2\mathbf{Z} \to {}_2\mathbf{Z}$ defined by $z(0) = 0$ and $z(x) = 1$ for $x \neq 0$ is *not* continuous at 0; it will therefore follow that is is not computable by *any* digital circuit; nor by *any* form of computer for that matter. The related z' defined by $z'(0) = 0$ and $z'(2^v(1 + 2x)) = 2^{-v}$ is on-line, and so computable by some synchronous circuit $z' \in \mathcal{C}(?, 2\times)$.

Example 3 (Peano Pairing) *Define the* Cartesian product $\pi \in {}_2\mathbf{Z} \times {}_2\mathbf{Z} \to {}_2\mathbf{Z}$ *of 2adic integers by* interleaving *the binary representations of each operand. The inverse first projection* $\pi_0 \in {}_2\mathbf{Z} \to {}_2\mathbf{Z}$ *extracts the* even *bits; the second* $\pi_1 \in {}_2\mathbf{Z} \to {}_2\mathbf{Z}$ *the* odd *bits. They are computed by the recursive system:*

$$\begin{array}{rcl} \pi(a,b) & = & a \cdot|\cdot 2 + 2\pi(b, a \div 2), \\ \pi_0(a) & = & a \cdot|\cdot 2 + 2\pi_0(a \div 4), \\ \pi_1(a) & = & \pi_0(a \div 2). \end{array}$$

We easily verify that $\forall a, b \in {}_2\mathbf{Z} :\ \pi_0(\pi(a,b)) = a,\ \pi_1(\pi(a,b)) = b$. Product π establishes a one to one correspondance $\mathbf{N} \leftrightarrow \mathbf{N} \times \mathbf{N}$ between natural numbers and pairs of natural numbers; similarly for odd denominator rationals $\mathbf{Z}/1 + 2\mathbf{N}$; similarly for the 2adic integers ${}_2\mathbf{Z}$. While Peano's Cartesian product $\pi \in \mathcal{C}(?, 2\times)$ is *on-line*, neither is π_0 nor π_1. We leave it as an interesting design exercise for the reader, to implement π by some synchronous circuit.

4 Binary Decision Diagrams

Theorem 1 *A function over the 2adic integers is* bit-wise *if and only if it may be computed by some combinational synchronous circuit (no register).*

The multiplexor is bit-wise. Bit-wise functions are closed under composition. So, the converse of the above equivalence is easy. In order to prove the direct implication, we must show that each boolean function $g \in \mathbf{B}^n \mapsto \mathbf{B}$ may be implemented with finitely many multiplexors, and no register.

This classical result is obtained by constructing *binary decision diagrams* BDD, as introduced by [A78] and [B86].

Algorithm 1 (BDD) *Let $f \in \mathbf{B}^n \to \mathbf{B}$ be some boolean function with n inputs. A combinational circuit $BDD(f) \in \mathcal{C}(?)$ for computing f may be constructed as follows.*

1. *Recursively decompose f by Shannon's formula:*

$$f(x_0, \cdots, x_{n-2}, x_{n-1}) = ?(x_{n-1}, f(x_0, \cdots, x_{n-2}, 1), f(x_0, \cdots, x_{n-2}, 0)).$$

   What we get at the end of this process is a complete binary tree composed of $2^n - 1$ muxes, whose leaves are labelled by the 2^n entries in f's truth table.

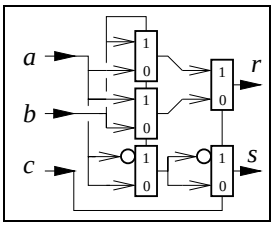
2. *Systematically share all equal sub-expressions generated during this process.*

While phase 2 of the BDD procedure is not required in order to establish theorem 1, it will be used latter. As shown by R. Bryant [B86], it leads to a *normal form* for combinational functions, in which all mux controls are primary inputs.

A further reduction in the size of the constructed circuit results from also sharing sub-expressions which are related by logical negation. This was proposed by J. P. Billon who preserves the normal form property by restricting inverters to only appear on branches where the synthesized function is one on all zeroes inputs (see [B87]).

Example 4 (FullAdder) *The following 5 muxes and 2 inverters result from applying algorithm BDD to the synthesis of a full-adder.*

$$
\begin{array}{lrcl}
FullAdd(a,b,c) & = & (s,r) \\
\textit{where} & \{o & = & ?(b,b,a), \\
& e & = & ?(b,a,b), \\
& r & = & ?(c,o,e), \\
& \overline{a} & = & \neg a, \\
& x & = & ?(b,\overline{a},a), \\
& \overline{x} & = & \neg x, \\
& s & = & ?(c,\overline{x},x)\}
\end{array}
$$



From now on, a trapezoid with two + signs inside denotes a full-adder.

5 Synchronous Decision Diagrams

Theorem 2 *A function over the 2adic integers is* on-line *if and only if it may be computed by some synchronous circuit.*

The converse implication states that the outputs of a synchronous circuit at time $t \in \mathbf{N}$ may only depend upon the values of its inputs during the first t clock cycles; this is obvious.

The direct implication is proved by constructing *synchronous decision diagrams* (SDD algorithm 2) which generalize the (BDD algorithm 1) to the synthesis of (sequential) circuits with memory.

The basic step in the SDD construction expresses each on-line function f in the form:

$$f(x) = ?(x,\, b_1 + 2f^{(1)}(x),\, b_0 + 2f^{(0)}(x)), \tag{5}$$

for some bits $b_0, b_1 \in \mathbf{B}$ and on-line functions $f^{(0)}, f^{(1)} \in {}_2\mathbf{Z} \to {}_2\mathbf{Z}$. From:

$$f(x) = \sum_{t \in \mathbf{N}} f_t(x_0, \cdots, x_t)2^t,$$

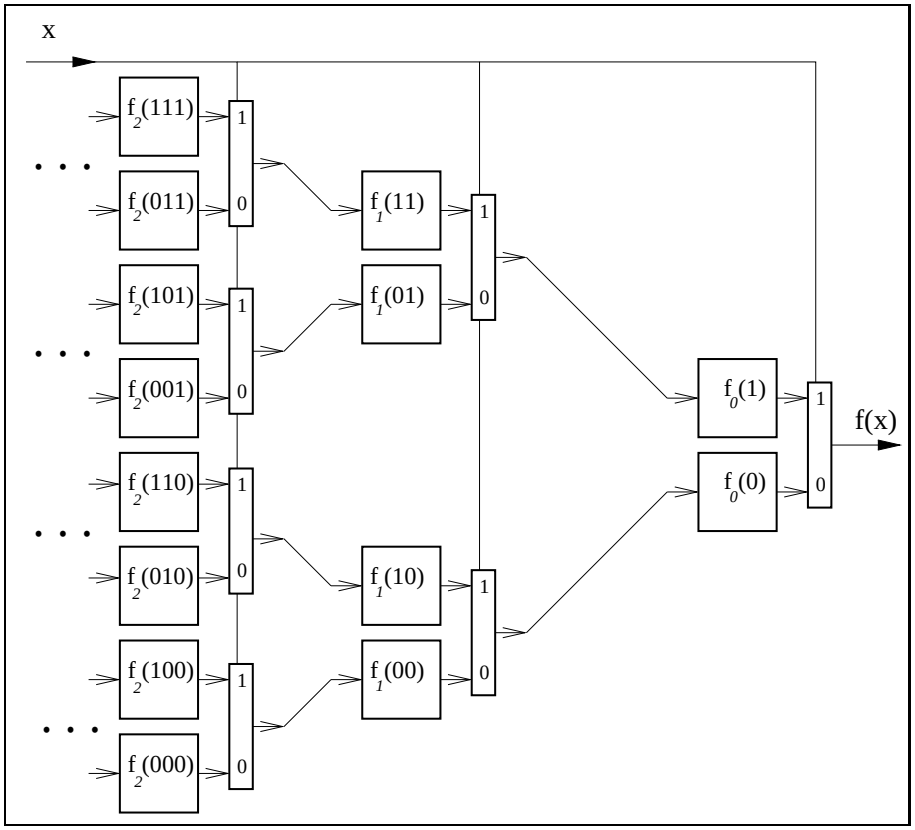
we see that $b_0 = f_0(0)$, $b_1 = f_0(1)$; so we have, for $b \in \mathbf{B}$:

$$f^{(b)}(x) = \sum_{t \in \mathbf{N}} f_{t+1}(x_0, \cdots, x_t, b)2^t.$$

Functions $f^{(0)}$ and $f^{(1)}$ *tabulate* what the value of $f(x)$ *will* be when the *last* input bit of x becomes b. We call them the *order one predictors* of function f, for $b = 0$

and $b = 1$. Through the same process, we compute four order two predictors $f^{(b_1)(b_0)} = f^{(b_0 b_1)}$ for $b_0 b_1 = 00, 01, 10$ and 11.

By indefinite iteration, we construct the infinite *predictor tree* for f. It may be drawn as:



In this schema, each rectangle denotes a multiplexor with control input x (label 1 on the high input, 0 on low); each square denotes a register: with initial value 0 when the label inside is zero, one otherwise (example 1).

Algorithm 2 (SDD) *Let $f \in {}_2\mathbf{Z} \to {}_2\mathbf{Z}$ be some on-line function defined by*

$$f(x) = \sum_{t \in \mathbf{N}} f_t(x_0, \cdots, x_t) 2^t,$$

with $f_t \in \mathbf{B}^{t+1} \mapsto \mathbf{B}$ for $t \in \mathbf{N}$. A synchronous circuit SDD(f)$\in \mathcal{C}(?, 2\times)$ for computing f may be constructed as follows:

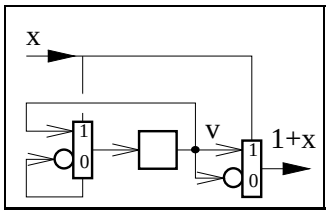
1. *Build the infinite predictor tree for f.*

2. *Systematically share all equal sub-expressions generated during this process, as well as their logical negations.*

Example 5 (Serial increment) *The following circuit results from applying the SDD procedure to the synthesis of the increment function* $1 + x$.

$$\begin{aligned}\{SDD(1+x) &= ?(x, v, \neg v),\\ v &= 2\times ?(x, v, \neg x)\}\end{aligned}$$

Note the relation $v = -2^{v_2(x)+1}$, *where* $v_2(x)$ *is the valuation of* x *(definition 6).*

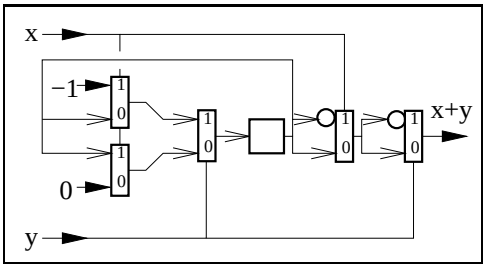


Notice the sharing of sub-expressions (predictors) which are the logical negation of one another.

To simplify notations, the SDD procedure is presented in the generic case of a one input ($k = 1$), one output on-line function. The generalization to an arbitrary number of inputs $k \geq 0$ and outputs is direct. Examples 2 show the result of the procedure for zero input ($k = 0$), and example 6 below has two inputs ($k = 2$).

When applied to a combinational function, the SDD procedure (algorithm 2) generates the same circuit as the BDD procedure (algorithm 1).

Example 6 (SDD Adder) *Apply the SDD procedure to the synthesis of a serial adder (see section 7.1) computing* $x + y$. *The synthesized adder SDD(*$x + y$*) is:*



While it is by nature an *infinite* process, algorithm SDD(f) generates a finite synchronous circuit (in finite time) if and only if function f may be defined by some *finite automaton*:

Proposition 3 (SDD Termination) *An on-line function* f *may be computed by some finite synchronous circuit if and only if the SDD algorithm 2 terminates, i.e. function* f *has finitely many predictors.*

Proof: If the SDD procedure terminates, function f may be computed by a *finite* synchronous circuit, namely SDD(f).

Conversely, let $f \in {}_2\mathbf{Z} \mapsto {}_2\mathbf{Z}$ be some on-line function:

$$y = f(x) = f(\sum_{t\in\mathbf{N}} x_t 2^t) = \sum_{t\in\mathbf{N}} f_t(x_0, \cdots, x_t)2^t = \sum_{t\in\mathbf{N}} y_t 2^t;$$

assume that it is computed by a finite synchronous circuit $f \in \mathcal{C}(?, 2\times)$, with n registers: $R = \{r[1], \cdots, r[n]\}$. This implies that, for $1 \leq k \leq n$, each register is defined by an equation of the form: $r[k] = 2 \times \mathcal{S}_k(x, R)$, where $\mathcal{S}_k \in \mathbf{B}^{n+1} \mapsto \mathbf{B}$ is a combinational function; in addition, the output $f(x) = \mathcal{O}(x, R)$ is also given by some combinational function $\mathcal{O} \in \mathbf{B}^{n+1} \mapsto \mathbf{B}$.

Define the *state of the registers* at time $t \in \mathbf{N}$ by the natural number $R_t = \sum_{1\leq k\leq n} r_t[k]2^k$, so that $R_0 = 0$, $R_{t+1} = \mathcal{S}(x_t, R_t)$ and $y_t = \mathcal{O}(x_t, R_t)$.

Consider the two first order predictors $y^{(b)} = f^{(b)}(x)$, defined by equation (5) for $b = 0$ and $b = 1$. Rewrite $y_t^{(b)} = f_{t+1}(x_0, \cdots, x_t, b)$ in terms of the state at time $t + 1$, and obtain $y_t^{(b)} = \mathcal{O}(b, R_{t+1})$.

The circuit defining $f^{(b)}$ has therefore the *same* state function $\mathcal{S}$ as f, and a different output function. By induction, this holds of each predictor of f. All predictors may thus be defined by circuits which all share the same state functions and registers; they only differ by their output functions.

Termination of the SDD procedure follows, as there are only finitely many such combinational output functions $\mathcal{O} \in \mathbf{B}^{n+1} \mapsto \mathbf{B}$. ∎

The SDD procedure provides a *normal form* for synchronous circuits. In this normal form the control input of each mux is one of the circuit's primary inputs, except for the inverters which arise from sharing logical negations. As for BDDs, this guarantees that SDD circuits having a small number of primary inputs are *electrically fast*, by construction.

The SDD yields excellent results when applied to the synthesis of some simple functions, such as $1 + 2x$, $1 + x$, $2x$, $x + y$ and $x - y$.

R. Bryant [B86] has shown that BDD synthesis is not applicable in practice to the synthesis of multipliers. The same holds for SDD. As a point in case, SDD($3x$) has 8 muxes and 5 registers; the circuit given in example 7 has only 2 registers, and requires only five muxes.

The arithmetic functions $\times$, $i(x) = \frac{1}{1+2x}$ and $r(x) = \sqrt{1 + 8x}$ are on-line. Although they are amenable to the SDD algorithm, we construct in section 7 much smaller bit-serial circuits for implementing such operations.

6 Circuits with output enable

By theorem 2, we know that there exists no synchronous circuit which computes the output sequence $x \div 2 = {}_2x_1x_2\cdots$ in response to the input $x = {}_2x_0x_1x_2\cdots$. In general, no synchronous circuit is capable of producing strictly less output bits than it consumes inputs. To get around this problem, experienced designers add a signal $en \in \mathcal{V}(C)$ which is used to *enable* the outputs from circuit C: it is set to $en_t = 1$ on cycles $t \in \mathbf{N}$ when the outputs of C are significant; it is set to $en_t = 0$ on cycles when the outputs of C are irrelevant.

By this convention, we can now compute $x \div 2$ through the identity circuit with output enable $en = {}_20(1) = -2$. The same identity circuit with output enable $en = {}_2(10) = -1/3$ computes Peano's first projection $\pi_0(x_0x_1x_2\cdots x_t\cdots) = x_0x_2\cdots x_{2t}\cdots$. Indeed, these are special cases of the following general result:

Theorem 3 *Every continuous function $f \in {}_2\mathbf{Z} \to {}_2\mathbf{Z}$ is computable by some synchronous circuit with output enable.*

Proof: By (4), we may express f as

$$f(x) = \sum_{k\in\mathbf{N}} 2^k f_k(x_0\cdots x_{m(k)}).$$

Since a boolean function with k inputs may be considered as a function with $k+1$ inputs which ignores the last one, we assume without loss of generality that $m(k) < m(k+1)$ for all $k \in \mathbf{N}$.

For each $t \in \mathbf{N}$ such that $m(k) \le t < m(k+1)$, define $g_t(x_0\cdots x_t) = f_k(x_0\cdots x_{m(k)})$. Function g is *on-line* by construction:

$$g(x) = \sum_{t\in\mathbf{N}} 2^j g_t(x_0\cdots x_t).$$

We know from theorem 2 that g may thus be computed by some synchronous circuit $C_g \in \mathcal{C}(?, 2\times)$. It follows that function f is computed by C_g with output enable:

$$en = \sum_{k\in\mathbf{N}} 2^{m(k)}.$$

■

Let us compose circuits with output enables. Suppose that $A \in \mathcal{C}(?, 2\times)$ has output enable en$\mathcal{A}$. We want to connect the inputs of A to the outputs $\mathcal{B}$ of some other synchronous circuit with output enable en$\mathcal{B}$. The rules are:

1. Replace every register $a = 2 \times b$ in A by the enabled register $a = 2\times?(en\mathcal{B}, b, a)$.

2. Set the output enable of the outputs of A to $en = en\mathcal{A} \wedge en\mathcal{B}$.

7 Arithmetic circuits

We now introduce bit-serial synchronous circuits for computing the arithmetic operations $+, -, \times, \frac{1}{1+2x}$ and $\sqrt{1+8x}$. They form the *core* of arithmetic circuits: from each of these atomic synchronous circuits, we may derive an arbitrary number of parallel implementation through time unfolding (see [L92]), and/or optimize them through retiming (see [LS91]).

The *only* finite circuits in this section are the serial + and $-$, since:

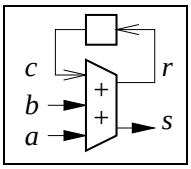
Proposition 4 *Any synchronous circuit for squaring a 2adic integer contains infinitely many registers.*

Proof: Suppose the contrary, namely some circuit $C \in \mathcal{C}(?, 2\times)$ with n registers produces output x^2 for each input $x \in {}_2\mathbf{Z}$. Circuit C may reach at most 2^n different states S. There are 2^{n+1} integers in the set $I = \{y \in \mathbf{N} : y = 2^{n+1}x, 0 \leq x < 2^{n+1}\}$. Take the inputs to circuit C from set I, and consider what happens at time $t = 2n + 2$:

1. all outputs produced up to this point are zero, since C is squaring;

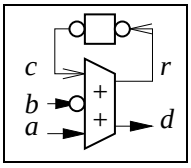
2. there are more elements in I than possible states; hence there must exist two *different* numbers $y \neq y'$ in I which bring C into the *same* state $S_{2n+2}(y) = S_{2n+2}(y')$, on inputs y and y'. All subsequent inputs are zero. Thus, all subsequent outputs must be equals. We have $C(y) = C(y')$, yet $y^2 \neq y'^2$, a contradiction. ∎

7.1 Addition



The basic arithmetic invariant of the full-adder is: $a+b+c = s + 2r$. Solve this system by letting $c = 2r$, and define addition by: $s = a+b$ where $(s, c) = FullAdd(a, b, 2\times c)$.

7.2 Substraction



Binary substraction is computed as $a - b = a + (-b)$, with $-b = 1 + \neg b$ the opposite of b. Define substraction by $d = a - b$ where $(d, c) = FullAdd(a, \neg b, 1 + 2 \times c)$.

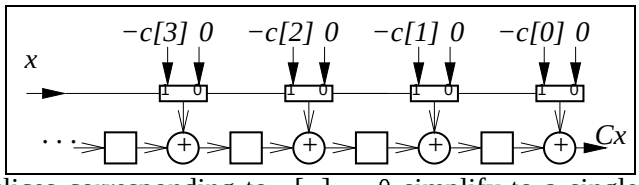
From now on, a circle with a + inside denotes a serial adder; a serial subtracter with $-$.

7.3 Serial-Parallel Multiplier

In order to multiply input $x \in {}_2\mathbf{Z}$ by some 2adic integer $C = \sum_{k\in\mathbf{N}} c[k]2^k$, consider the following elementary identity:

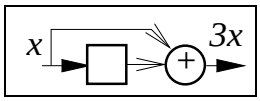
$$C \times x = (c[0] + 2(C \div 2)) \times x = (-c[0]) \wedge x + (C \div 2) \times 2x.$$

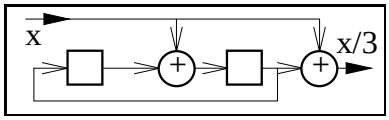
It provides a direct justification for the following infinite multiplier:



When C is constant, slices corresponding to $c[n] = 0$ simplify to a single register; muxes may all be eliminated. For $C = {}_2c_0 \cdots c_{i-1}(c_i \cdots c_{i+p-1})$ a constant periodic rational, the multiplier becomes finite with $i + p$ slices: the input to the last slice is the output from the i-th slice. When $C = {}_2c_0 \cdots c_{e-1}(c_e)$ is a constant integer, the multiplier becomes identical to the classical two's complement Lyon's multiplier [L76], after retiming.

Example 7 *The following circuits compute respectively* $3 \times x = {}_211(0) \times x$ *and* $x/3 = {}_21(10) \times x$*:*





7.4 Serial-Serial Multiplier

Let $x = {}_2x_0 \cdots x_t \cdots$ and $y = {}_2y_0 \cdots y_t \cdots$ be the operands to be multiplied in order to compute serially the product $p = x \times y = {}_2p_0 \cdots p_t \cdots$. The invariant of this synchronous multiplier $M \in \mathcal{C}(?, 2\times)$ is:

$$M(2^t, x, y) = 2^t(x \div 2^t) \times (y \div 2^t).$$

From the elementary identity in the 2adic ring,

$$(x \div 2^t) \times (y \div 2^t) = x_t y_t + 2x_t(y \div 2^{t+1}) + 2y_t(x \div 2^{t+1}) + 4(x \div 2^{t+1}) \times (y \div 2^{t+1}),$$

we derive the recurrence relation,

$$M(2^t, x, y) = A(2^t, x, y) + 2M(2^{t+1}, x, y),$$

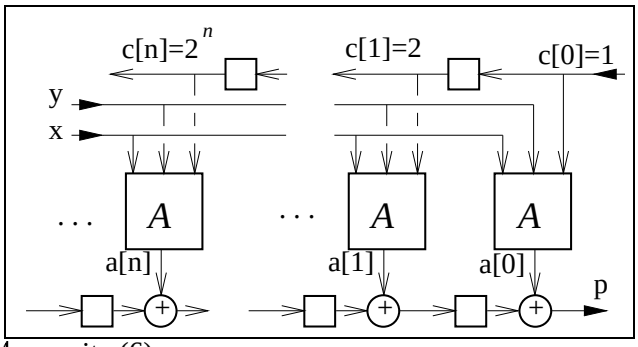
where A is the auxiliary function:

$$2^{-t}A(2^t, x, y) = a[t] = x_t y_t + 2x_t(y \div 2^{t+1}) + 2y_t(x \div 2^{t+1}). \qquad (6)$$

The final product is obtained as $p = M(1, x, y) = \sum_{t \in \mathbf{N}} a[t]2^t$, where:

$$M(c, x, y) = A(c, x, y) + 2M(2 \times c, x, y).$$

The resulting cellular structure looks like:

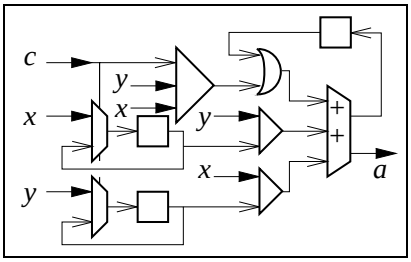


In order to design cell A, rewrite (6) as

$$A(c, x, y) = (x \wedge y \wedge c) + x \wedge (-2(c \wedge y)) + y \wedge (-2(c \wedge x))$$

which is equal to $2^t a[t]$, provided that $c = 2^t$.

This equation translates to the finite circuit to the right, where triangles denote *and* gates, and half-circles *or* gates.



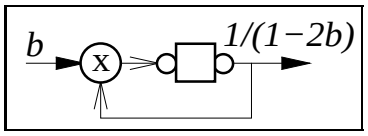
This design leads to more economical circuits than the Atrubin (see [K81]) or the Chen and Willoner constructions [CW79]. From here on, a circle with an $\times$ inside denotes this serial-serial multiplier.

7.5 Odd Inverse

An even 2adic integer $b = 2b' \in {}_2\mathbf{Z}$ has *no* inverse $b^- \in {}_2\mathbf{Z}$: indeed $2b \times b^-$ is even and $2b \times b^- \neq 1$ for all b^-. So ${}_2\mathbf{Z}$ is not a field, but it comes close: we can define the *odd inverse* $i = 1/(1-2b) \in {}_2\mathbf{Z}$ of any 2adic integer $b \in {}_2\mathbf{Z}$ by the formula:

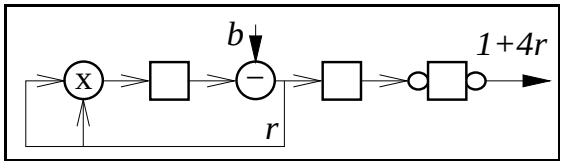
$$i = \frac{1}{1-2b} \quad = \quad \sum_{k \in \mathbf{N}} (2b)^k. \qquad (7)$$

Rewriting (7) as $i(1-2b) = 1$, we obtain $i = 1+2ib$ which translates to the synchronous circuit to the right. While it looks finite in the picture, this is yet another infinite circuit since it contains the multiplier from the previous section.

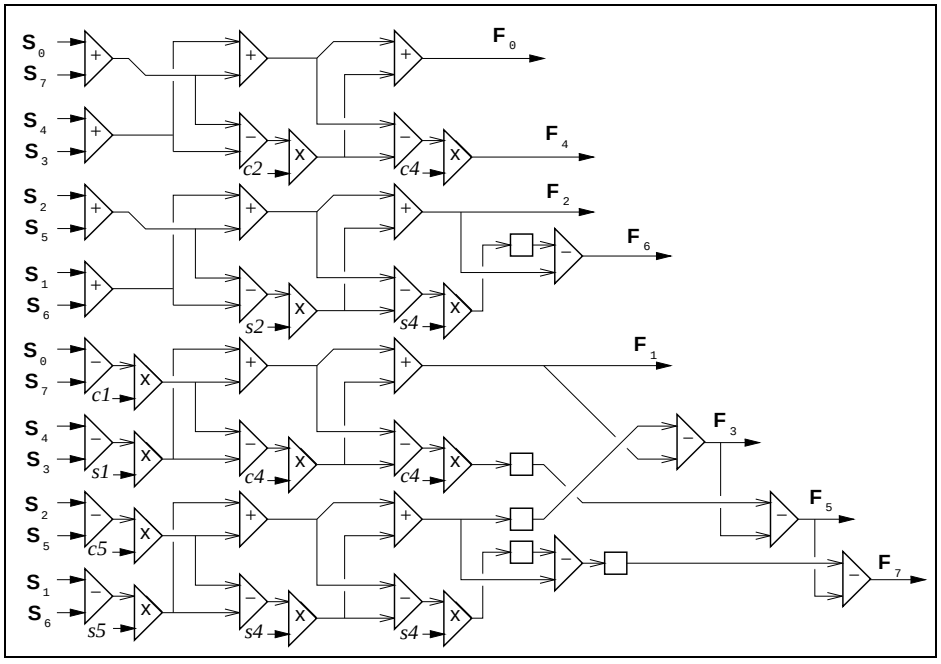


7.6 Square Root

An odd 2adic integer has a square root if and only if it is congruent to one modulo 8. Such a number $1 + 8b \in {}_2\mathbf{Z}$ has exactly two square roots: $(+\sqrt{1+8b}) \cdot|\cdot 4 = 1$ and $(-\sqrt{1+8b}) \cdot|\cdot 4 = 3$. We compute the former $\sqrt{1+8b} = 1+4r$ so as to verify: $(1+4r)^2 = 1+8r+16r^2 = 1+8b$. Simplifying this last expression to $r+2r^2 = b$, we see that the square root $1 + 4r = +\sqrt{1+8b}$ is given by $r = b - 2r^2$ which translates to the following (infinite) synchronous circuit:



8 Synchronous circuits with Reset



Consider an arbitrary synchronous network, such as the one drawn above which computes the fast discrete cosine transform for appropriate coefficients $s1, \cdots, s7, c1, \cdots, c7 \in \mathbf{N}$. How can we pipe-line a sequence of finite precision integer computations through such a network? By simply adjoining a (synchronous) *reset* input, and replacing each register equation $r = 2 \times i$ in the network by the register with reset:

$$r = 2 \times ?(reset, 0, i).$$

Theorem 4 *In order to pipe-line a network $f \in {}_2\mathbf{Z} \to {}_2\mathbf{Z}$ over integer input sequences $i_0\, i_1 \cdots i_t \cdots \in \mathbf{Z}$ whose binary lengths vary in time $\forall t \in \mathbf{N}, \exists m(n) \in \mathbf{N} : i_t < 2^{m(n)}$, we adjoin the following reset signal to all registers:*

$$rst = 1 + \sum_{t \in \mathbf{N}} 2^{m(t)}.$$

With this reset, network f now computes:

$$F(\sum_{t \in \mathbf{N}} i_t z^t) = \sum_{t \in \mathbf{N}} (f(i_t) \cdot | \cdot 2^{m(t)}) z^t.$$

Note that all the operators which have so far received an infinite definition, such as multiplication, become *finite* as soon as the number $m = max_{n \in \mathbf{N}} m(n)$ is itself finite; indeed, we can truncate the whole network at m bits since all final results are correct modulo 2^{m+1}. The reset signals of two circuits get or-ed together, during composition.

9 Conclusion

With G. Berry and F. Bourdoncle, we have used this 2adic theory as the semantic basis for a new circuit language called 2Z [BVB94].

Our main motivation is quite practical. *Programmable active memory* PAM technology, as introduced by [BRV89], is based on large arrays of configurable logic. It already permits to implement synchronous designs, such as those reported in [BRV93] and [SV93] which are much larger than whatever may currently fit on single silicon chips: well over a million of active gates, excluding RAM. We have learned from experience that the main obstacle to the development of PAM technology comes from the time required to program such large innovative hardware designs.

We expect the 2Z language to help overcome this bottleneck, as it incorporates a number of advanced synthesis features, based on the view of 2adic synchronous logic presented here. Before commenting on these synthesis features, let us provide the reader with a feel for the language through a small example.

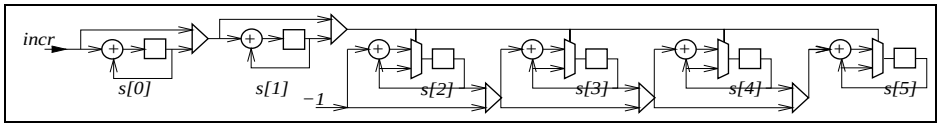
The following $2Z$ source code generates the counters from [V91] whose operating speed is, for all practical purposes, *independent* of the counter's length.

```
SlowCounter[n](incr) = (s[n],ovfl)
where
  c[0]=incr;
  for k<n do
    c[k+1] = c[k] and s[k];
    s[k]   = reg(c[k] xor s[k])
  end for;
  ovfl = c[n]
end where;

FastCounter[n](incr) = s[n]
where
  k=3;    // this parameter is technology-dependent //
  if n<k
```

```
    then (s,ovfl) = SlowCounter[n](incr);
    else (s[0..k-1],en) = SlowCounter[k](incr);
         enable en in
           (s[k..n-1],cn) = SlowCounter[n-k](-1);
         end enable;
         ovfl = cn and en;
  end if
end where
```

The schema resulting from to the execution of `FastCounter[6]` is:



Let us summarize the circuit synthesis techniques which 2Z incorporates, in relation to the present work.

1. The language knows 2adic rationals. The expression `ctrl=#(0011)`, or equivalently `ctrl=-4/5` will generate the proper circuit from example 2, based on proposition 2 and the SDD algorithm. This provides a valuable help in synthesizing the local control structures normally associated with each part of the data pathes. Note that 2Z truly handles *arbitrary precision*.

2. The language knows bit-serial arithmetics, as exposed in section 7. Writing `c=a-b` will generate the proper serial substract, and `d=252*c` the proper multiplier. This is useful for synthesizing data pathes from graphs of synchronous operators.

3. The language knows about enable, reset and how to compose such powerful non local circuit constructions.

4. Finally, the 2Z language knows about fine grain boolean logic synthesis and retiming (see [LS91]). Both are mandatory within the electrical optimization of delays in PAM designs. In the 2adic framework, they express commutation properties between mux, reg and arbitrary functions. These can be stated, for two input functions, by:

   Proposition 5 • *A continuous function* $f \in {}_2\mathbf{Z}^2 \to {}_2\mathbf{Z}$ *commutes with the mux,*

   $$\forall a, b, c, d, e \in {}_2\mathbf{Z} : ?(a, f(b, c), f(d, e)) = f(?(a, b, d), ?(a, c, e)),$$

   if and only if it is a combinational function.

- *An on-line function* $f \in {}_2\mathbf{Z}^2 \to {}_2\mathbf{Z}$ *commutes with the register,*

$$\forall a, b \in {}_2\mathbf{Z} : \; 2f(a, b) = f(2a, 2b),$$

if and only if $f(0, 0) = 0$.

Let us now consider some open directions for further research.

- As pointed out earlier, the SDD procedure is an interesting candidate for compiling finite state machine descriptions (hopefully produced by higher level systems) into real hardware (silicon or FPGA). It should be instructive in this respect to compare the resulting SDD implementation with the direct techniques reported by [B92], and others.
- We expect some CAD systems to incorporate rules for circuit verification, having to do with the *ring* properties of 2adic algebra; not just the boolean part. The need for such tools is clear when one considers the problem of proving functionally equivalent, such structurally different multipliers as, say the one from section 7.4, and the fully parallel one from [V83]. This appears to require the full 2adic apparatus introduced here; any proof through independent means has to discover (prove) and use the ring laws somewhere along the line; and their relations to boolean algebra, as expressed by propositions 1 and 2.
- While *finite* circuits play a central role, both from a theoretical and practical point of view, we have not been able to quantitatively characterize them: how many registers does each finite circuit truly requires?

10 Acknowledgements

This work has benefited from important contributions by Gérard Berry, François Bourdoncle, Hervé Touati and Patrice Bertin. Merci à chacun.

11 Proof of Proposition 2

(ii) Let $l \in \mathbf{N}$ be the least integer such that $|z| < 2^l$. After computing l bits of z by rule 1, we reach the state: $\mathcal{B}_2(z) \stackrel{l}{\Rightarrow} z_0 \cdots z_{l-1} \mathcal{B}_2(z \div 2^l)$. When $z \geq 0$, we have $z \div 2^l = 0$, so $\mathcal{B}_2(z \div 2^l) = {}_2(0)$; when $z < 0$, we have $z \div 2^l = -1$, so $\mathcal{B}_2(z \div 2^l) = {}_2(1)$.

(iii) Conversely, $z = {}_2z_0 \cdots z_{l-1}(z_l) = \sum_{k<l} z_k 2^k - 2^l z_l$ is an integer $z \in \mathbf{Z}$.

(iv) Integer $z = {}_2z_0 \cdots z_{l-1}(z_l)$ is computed by the following acyclic synchronous circuit, with l registers: $r[l-1] = z_{l-1} + 2 \times z_l, r[l-2] = z_{l-1} + 2 \times r[l-1], \cdots, r[0] = z_0 + 2 \times r[1]$. The inputs $-z_l = {}_2(z_l), 0, -1$ are constant, and the output is $r[0]$.

Conversely, it follows from definitions 2,3 of mux and reg that an acyclic synchronous circuit with $l \in \mathbf{N}$ registers produces a constant output after at most l cycles, upon constant input.

(vi) Let $B = \frac{z}{1+2n}$ be an odd rational, with $z \in \mathbf{Z}$ and $n \in \mathbf{N} + 1$. Define the *period* $p = p_2(1 + 2n)$ of the denominator $1 + 2n$ to be the order of 2 in the multiplicative group $\mathbf{Z} \cdot|\cdot (1 + 2n)$ of the integers modulo $1 + 2n$, namely the smallest natural number such that:

$$2^p = 1 \pmod{1 + 2n}.$$

Let $q = (2^p - 1) \div (1 + 2n)$, the corresponding remainder being 0. Compute $I = zq \div (2^p - 1)$ the quotient, and $P = zq \cdot|\cdot (2^p - 1) = {}_2p_0 \cdots p_{p-1}$ the remainder in the integer division of $z \times q$ by $2^p - 1$, so as to write:

$$B = \frac{z}{1 + 2n} = I - \frac{P}{2^p - 1}.$$

The binary representation of $\frac{-P}{2^p-1} = {}_2(p_0 \cdots p_{p-1})$ is purely periodic. It follows that the binary representation of B is ultimately periodic: $b_k = b_{k+p}$ for $k \geq i + p$.

(vii) Let number $B = {}_2b_0 \cdots b_{i-1}(b_i \cdots b_{i+p-1})$ be *ultimately periodic*, and consider the integers $I = {}_2b_0 \cdots b_{i-1}$ and $P = {}_2b_i \cdots b_{i+p-1}$. Number $B \in \mathbf{Z}/1 + 2\mathbf{N}$ is the odd rational:

$$B = I - \frac{2^i P}{2^p - 1}.$$

(viii) The periodic rational $B = {}_2b_0 \cdots b_{i-1}(b_i \cdots b_{i+p-1})$ is computed by the following synchronous circuit, with $i + p$ registers: $r[i + p - 1] = b_{i+p-1} + 2 \times r[i], r[i + p - 2] = b_{i+p-2} + 2 \times r[i + p - 3], \cdots, r[0] = b_0 + 2 \times r[1]$. The inputs 0 and -1 are constants, and the output is $r[0]$.

Conversely, let $r[0], \cdots, r[n-1]$ be the n registers of a finite synchronous circuit $C \in \mathcal{C}(?, 2\times)$. Define the *state* of C at time t by the integer

$S_t = \sum_{0 \leq k < n} r_t[k] 2^k$. This number is bounded by $S_t < 2^n$; so, there must exist two instants $0 \leq t_0 < t_1 < 2^N$ where we find the circuit in the *same* state: $S_{t_0} = S_{t_1}$. With a constant input, the output must therefore be periodic $B = {}_2 b_0 \cdots b_{i-1} (b_i \cdots b_{i+p-1})$ with $i = t_0$ and $p = t_1 - t_0$. ∎

References

[A75] Y. Amice, *Les nombres p-adiques*, in *Presses Universitaires de France*, 1975.

[A78] S. B. Akers, *Binary decision diagrams*, in *IEEE Trans. Computers*, 27:509–516, 1978.

[BRV89] P. Bertin, D. Roncin and J. Vuillemin, *Introduction to Programmable Active Memories*, in *Systolic Array Processors*, J. McCanny, J. McWhirter, E. Swartzlander Jr. editors, Prentice-Hall, 301–309, 1989. Also available as *PRL report 3*, Digital Equipment Corp., Paris Research Laboratory, 85, Av. Victor Hugo, 92563 Rueil-Malmaison Cedex, France, 1989.

[BRV93] P. Bertin, D. Roncin and J. Vuillemin, *Programmable Active Memories: a Performance Assessment*, in *Symposium on Integrated Systems*, Seattle, WA, USA, MIT Press, March 1993. Also available as *PRL report 24*, Digital Equipment Corp., Paris Research Laboratory, 85, Av. Victor Hugo. 92563 Rueil-Malmaison Cedex, France, 1993.

[B92] G. Berry, *A Hardware Implementation of Pure Esterel*, in *Academy Proceedings in Engineering Sciences, Indian Academy of Sciences*, Sadhana, 17:1:95–130, 1992. Also available as *PRL report 15*, Digital Equipment Corp., Paris Research Laboratory, 85, Av. Victor Hugo. 92563 Rueil-Malmaison Cedex, France, (1989).

[B87] J. P. Billon, *Perfect Normal Forms for Discrete Function*, BULL Research Report, 87019, 1987.

[BRV93] F. Bourdoncle, J. Vuillemin and G. Berry, *The 2Z reference manual, PRL report ??*, Digital Equipment Corp., Paris Research Laboratory, 85, Av. Victor Hugo. 92563 Rueil-Malmaison Cedex, France, 1994.

[B86] R. E. Bryant, *Graph-based Algorithms for Boolean Function Manipulation*, in *IEEE Trans. Computers*, 35:8:677–691, 1986.

[CW79] I. N. Chen, R. Willoner, *An O(n) parallel multiplier with bit-sequential input and output*, in *IEEE Trans. Computers*, 28:10, Oct. 1979.

[H13] K. Hensel, *Zahlentheorie*, in *Gőshen*, Berlin-Leiptzig, 1913.

[K77] N. Koblitz, *p-adic Numbers, p-adic Analysis and Zeta Functions*, Springer-Verlag, 1977.

[K81] D.E. Knuth, *Seminumerical Algorithms*, in *The Art of Computer Programming*, Addison Wesley, 2, 1981.

[L91] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures*, Morgan Kaufmann, 1991.

[LS91] C. Leiserson, J. Saxe, *Retiming synchronous circuitry*, in *Algorithmica*, 6:1:5–35, 1991.

[L76] R. F. Lyon, *Two's complement pipeline multipliers*, in *IEEE Trans. Comm.*, 24:418–425, 1976.

[M89] C. Mead, *Analog VLSI and Neural Systems*, Addison-Wesley, 1989.

[SV93] M. Shand, J. Vuillemin, *Fast Implementations of RSA Cryptography*, in *11-th IEEE Symposium on Computer Arithmetic*, 1993.

[V83] J. Vuillemin, *A very fast multiplication algorithm for VLSI implementation*, in *INTEGRATION, the VLSI Journal*, 1:1, Mars 1983.

[V91] J. Vuillemin, *Constant Time Arbitrary Length Synchronous Binary Counters*, in *10-th IEEE Symposium on Computer Arithmetic*, 301–309, 1991.