TR-NIS- 0004

"S_1^3 - only" Logic

Andrzej BULLER

2005.6.22

国際電気通信基礎技術研究所　ネットワーク情報学研究所
〒619-0288 「けいはんな学研都市」光台二丁目2番地2
Tel: 0774-95-2641　Fax: 0774-95-2647

Advanced Telecommunications Research Institute International (ATR)
Network Informatics Laboratories
2-2-2, Hikaridai, "Keihanna Science City", Kyoto 619-0288, Japan
Tel: +81-774-95-1111　Fax: +81-774-95-2647

"S_1^3-only" logic

Andrzej Buller

Abstract—This report provides a consideration of a logic system based exclusively on the Boolean function S_1^n, called here MEXOR, that returns 1 iff exactly one of its n arguments equals 1. Proven lemmas show how to built various Boolean functions based exclusively on 3-input MEXORs. An up-to-3-input MEXOR can be embodied as a q-cell—a square-shaped device exchanging bits with neighbor cells and returning an output value after one clock. This report presents as examples of q-cell-based circuits, flat crossing, AND-, OR-, NOR-, and NAND-gate, multiplexer, more-than-one-of-three majority function, adder, and timer.

Key words—logic, symmetric functions, cellular arrays, pulse circuits, delay circuits

I. INTRODUCTION

An elementary symmetric Boolean function, denoted S_k^n, returns 1 iff exactly k out of its n arguments are equal to 1, where no permutation of arguments changes the returned value [1]. A non-elementary symmetric Boolean function can be created as the sum of elementary ones. Synthesis of arbitrary symmetric functions is an attractive topic within mainstream logic design, since, compared with synthesis of an arbitrary non-symmetric function, it is believed to be more tractable. Within the logic synthesis community the synthesized functions are presented mostly as sums of the products of non-negated and negated variables or as Reed-Muller expressions in which only AND and EXOR (EXclusive OR) functions are employed.

I propose to consider a logic system based exclusively on S_1^3. Even if the current VLSI industry, for reasons of economy, does not regard S_1^3 as the best solution, this situation will not necessarily last indefinitely. On the contrary, advantageous properties demonstrated by S_1^3 may encourage electronic engineers and physicists to search for a way to economically implement it.

In this report, I summarize the findings on S_1^n logic, especially for $n = 3$, and provide some examples of S_1^3-based circuits. To more conveniently discuss S_1^n and write formulas based on it, I coined the name MEXOR [2] and here introduce a simplified notation based on square

brackets. Using the proposed notation, $S_1^n(x_1, x_2, ..., x_n)$ is written as $[x_1\, x_2 \ldots x_n]$ and is read as "MEXOR of x_1, x_2, and so on up to x_n."

A logical system based only on a 3-input MEXOR and constant 1 is universal, or, in other words, complete (which means that any Boolean function can be built based exclusively on 3-input MEXORs, provided that a constant signal equal to 1 is available). Moreover, several useful 2- and 3-input Boolean functions can be built from only few 3-input MEXORs. This report provides a collection of proven lemmas justifying the above assertions.

Let us also consider an up-to-3-input MEXOR embodied as square tile such that each side not used as an input can be an output and that an output value is always returned with the same, defined delay. I named such tile a q-cell [3], which is to not to be confused with quantum-dot cells, that are denoted using capital Q and work based on a substantially different paradigm. We can employ 2- and 3-input q-cells as logic units and single-input q-cells as pieces of wire and/or as fan-outs; consequently, we can build arbitrary Boolean functions as well as arbitrary generators and modifiers of Boolean time-series. Each of these constructions can be a single layer of q-cells.

As for related works, S_1^n, where $1 \le n \le 5$, together with a certain threshold element were employed to evolving neural-like modules using a dedicated hardware called the CAM-Brain Machine (CBM) [4]. However, this approach could not provide a satisfactory solution. The work done in [5], also devoted to the CBM-style of computing, suggested that all two-input and some 3-input Boolean functions could be built exclusively from S_1^n functions. The work achieved in [6] reported the first successful use of a genetic algorithm for the synthesis of a non-trivial "S_1^n-only" circuit.

In order to make this report comprehensive for a wide readership, Section II reviews the basic notation and some theorems/lemmas useful for further lemma proving, paying most attention to EXOR properties. However, it is assumed that the target reader knows the most basic laws of Boolean algebra (e.g. De Morgan's laws), so these are neither provided here nor referred to in the proofs of lemmas. Section III contains a collection of proven lemmas specific to MEXOR logic. Section IV shows some examples of circuits built from q-cells.

II. Basic concepts

Let us note that an n-input MEXOR, i.e. S_1^n, for $n = 1$ has only one argument, where the

returned value can only be the argument unchanged (so S_1^1 can be called *identity function*); similarly, for $n=2$, such a MEXOR is simply an EXOR denoted commonly using the operator $\oplus$.

Until a method of a "MEXOR-only" expansion of an arbitrary Boolean function is established, MEXOR-related lemma proving can be done by manipulation on equivalent NOT/AND/OR-based or Reed-Miller expressions. Therefore, assuming the notations $\bar{x}$, xy, and $x+y$ mean "NOT x", "x AND y", and "x OR y", respectively, let us first recall two Boolean-algebraic theorems.

Theorem 1 [7]: $x+\bar{x}y=x+y$.

Theorem 2 [7]: $xy+\bar{x}z+yz=xy+\bar{x}z$.

Let us also note that $$S_1^2(x,y)=x\oplus y=x\bar{y}+\bar{x}y. \tag{1}$$

By Eq. (1) it can be noted that

$$x\oplus 0=x, \tag{2}$$

$$x\oplus 1=\bar{x}, \tag{3}$$

$$x\oplus x=0, \tag{4}$$

$$x\oplus \bar{x}=1. \tag{5}$$

Other properties of EXOR are reflected by the following four lemmas.

Lemma 1 [8]:

i. $(x\oplus y)\oplus z=x\oplus(y\oplus z)$,

ii. $xy\oplus xz=x(y\oplus z)$,

iii. $x\oplus y=y\oplus x$.

Lemma 2 [4]:

$xy=0 \Leftrightarrow x+y=x\oplus y$.

Lemma 3 [9]:

$(x+y)\oplus(x+z)=\bar{x}(y\oplus z)$.

Lemma 4:

i. $\bar{x}y \oplus y = xy$,

ii. $\bar{x}y \oplus x = x + y$.

Proof: For 4i., $\bar{x}y \oplus y = y\bar{x} \oplus y1$, which, according to Lemma 1ii, equals $y(\bar{x} \oplus 1)$. By Eq. (3), $\bar{x} \oplus 1 = x$. Hence, $y(\bar{x} \oplus 1) = yx = xy$. For 4ii., by Eq. (1), $\bar{x}y \oplus x = \bar{x}y\bar{x} + \overline{\bar{x}y}x \quad = \bar{x}y + (x + \bar{y})x$ $= \bar{x}y + (xx + \bar{y}x) \quad = \bar{x}y + (x + x\bar{y}) \quad = \bar{x}y + (x + x\bar{y}) = \bar{x}y + x \quad = x + y$. □

Finally, let us define the 3-input majority function M.

$$M(x_1, x_2, x_3) = S_3^3(x_1, x_2, x_3) + S_2^3(x_1, x_2, x_3). \tag{6}$$

III. MEXOR LOGIC

Note that, a value returned by MEXOR does not change after removal of all arguments equal to 0. Therefore,

$$\underset{i>k}{\forall} x_i = 0 \Rightarrow [x_1\, x_2 \ldots x_k\, x_{k+1} \ldots x_n] = [x_1\, x_2 \ldots x_k]. \tag{7}$$

The following discussion only concerns a three-argument MEXOR, which expands as follows:

$$S_1^3(x, y) = [x\, y\, z] = x\,\bar{y}\,\bar{z} + \bar{x}\, y\,\bar{z} + \bar{x}\,\bar{y}\, z. \tag{8}$$

By Eq. (7) and (8), it can be noted that

$$[x] = x. \tag{9}$$

By Eq. (7), (8) and (1),

$$[x\, y] = x \oplus y. \tag{10}$$

By Eq. (10) and (3),

$$[x\ 1] = \bar{x}. \tag{11}$$

And by Eq. (8),

$$[x\, y\, y] = x\bar{y}. \tag{12}$$

The basic difference between a three-argument MEXOR and a three-input combination of EXORs is reflected in the following lemma.

Lemma 5: $xyz = 0 \Leftrightarrow [x\, y\, z] = x \oplus y \oplus z$.

Proof: For $(\Rightarrow)$, by Eq. (1)

$x \oplus y \oplus z = (x \otimes y)\bar{z} + (\overline{x \oplus y})z = (x\bar{y} + \bar{x}y)\bar{z} + (\overline{x\bar{y} + \bar{x}y})z = x\bar{y}\bar{z} + \bar{x}y\bar{z} + (xy + \bar{x}\bar{y})z$

$= x\bar{y}\bar{z} + \bar{x}y\bar{z} + xyz + \bar{x}\bar{y}z = x\bar{y}\bar{z} + \bar{x}y\bar{z} + 0 + \bar{x}\bar{y}z,$ which by Eq. (8) equals $[x\,y\,z]$.

For $(\Leftarrow)$, since $(p \Rightarrow q) \equiv (\bar{q} \Rightarrow \bar{p})$, it is enough to prove that $xyz \neq 0 \Rightarrow [x\,y\,z] \neq x \oplus y \oplus z$.

Note that $[1\,1\,1] = 0$ and $xyz \neq 0 \equiv x = y = z = 1$, whereas $1 \oplus 1 \oplus 1$, by Lemma 1 and Eq. (4) and (2), equals 1, which completes the proof. □

The following four lemmas provide practical hints for "MEXOR-only" logic design.

Lemma 6: i. $[[x\,x\,y]\,y] = xy$,

ii. $[[x\,x\,y]\,x] = x + y$.

Proof: Let $[[x\,x\,y]\,y] = u$, $[[x\,x\,y]\,x] = w$. By Eq. (12), $[x\,x\,y] = \bar{x}y$. Hence $u = [(\bar{x}y)\,y]$, $w = [(\bar{x}y)\,x]$, which, by Eq. (10), gives $u = (\bar{x}y) \oplus y$, $w = (\bar{x}y) \oplus x$. Hence, by Lemma 4i, $u = xy$, and by Lemma 4ii, $w = x + y$. □

Lemma 7: i. $[x\,y\,1] = \overline{x + y}$,

ii. $[[x\,y\,1]\,x\,y] = \overline{xy}$.

Proof: For 7i., by definition, since one of MEXOR's arguments equals 1, the only possibility of returning 1 is when both of the other arguments are equal to 0. Hence, $[x\,y\,1] = \bar{x}\bar{y}$, which equals $\overline{x + y}$. For 7ii., by Lemma 7i, $[[x\,y\,1]\,x\,y] = [(\overline{x + y})\,x\,y] = (\overline{x + y})\bar{x}\bar{y} + (x + y)x\bar{y} + (x + y)\bar{x}y$

$= \bar{x}\bar{y}\bar{x}\bar{y} + xx\bar{y} + yx\bar{y} + x\bar{x}y + y\bar{x}y = \bar{x}\bar{y} + x\bar{y} + y\bar{x} = \bar{y}(\bar{x} + \bar{x}) + y\bar{x} = \bar{y} + y\bar{x}$, which by Theorem 1 equals $\bar{y} + \bar{x} = \overline{xy}$. □

Lemma 8: $[[x[x\,x\,y]][y\,z\,z]] = \bar{y}x + yz$.

Proof: $[[x[x\,x\,y]][y\,z\,z]]$, according to Lemma 4ii. and Eq. (12), equals $[(x + y)\,(y\bar{z})]$, which, by Eq. (10) equals $(x + y) \oplus y\bar{z}$, which by Eq. (1) equals $(x + y)\overline{y\bar{z}} + \bar{x}\bar{y}y\bar{z} = (x + y)(\bar{y} + z) = x\bar{y} + xz + yz$, which by Theorem 2 equals $\bar{y}x + yz$. □

Lemma 9: $[x\,[x\,x\,y]\,[[x\,y]\,z\,z]] = M(x,y,z)$.

Proof: By Eq. (8) $[x[x\,x\,y][[x\,y]z\,z]] = L = p + q + r$, where $p = x(\overline{\bar{x}y})(\overline{\bar{z}(x \oplus y)}) = x(x+\bar{y})(z+xy+\overline{xy})$,

$q = \bar{x}(\overline{\bar{x}y})(\overline{\bar{z}(x \oplus y)}) = \bar{x}y(z+xy+\bar{x}\,\bar{y})$, $r = \bar{x}(\overline{\bar{x}y})\bar{z}(x \oplus y) = \bar{x}(x+\bar{y})\bar{z}(x\bar{y}+\bar{x}y) = 0$.

Hence, $L = p + q = x(z+xy+\overline{xy}) + \bar{x}y(z+xy+\overline{xy}) \quad = xz + xy + \bar{x}yz = x(y+z) + \bar{x}yz$. Let $R = M(x,y,z)$, which by Eq. (6) equals $S_3^3(x,y,z) + S_2^3(x,y,z)$

$= xyz + xy\bar{z} + x\bar{y}z + \bar{x}yz = xy + x\bar{y}z + \bar{x}yz \quad = x(y+\bar{y}z) + \bar{x}yz$, which by Theorem 1 equals $x(y+z) + \bar{x}yz$. Therefore, $L = R$. □

Lemma 8 shows a MEXOR-based multiplexer, whereas Lemma 9 shows a MEXOR-based 3-input majority function. The latter construction is fully MEXOR-specific, i.e. it processes values of three different variables provided to the three inputs.

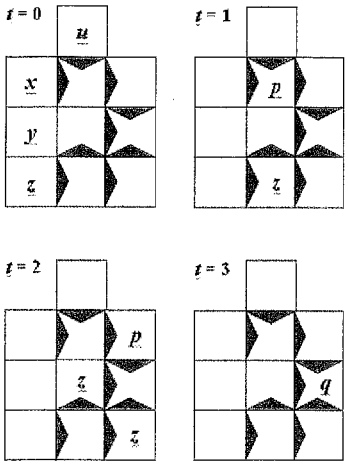
IV. *q*-CELLULAR CIRCUITS

Let us depict a *q*-cell using a square and triangles marking inputs, as in Fig. 1. When no triangle is attached to a given side, that side can serve as an output. If only one input is defined, the related tile is an identity function (returning only the input's value) with the possibility to fan-out the input value in three directions. If two and only two inputs are defined, such a *q*-cell serves as an EXOR-gate with the possibility to send a calculated value in two directions. If three inputs are defined, the related tile serves as a MEXOR with the possibility to send a calculated value in one direction (Fig. 2).

When a chessboard-like structure is built from a number of *q*-cells, the propagation of the values calculated by the cells depends on input configuration (Fig. 3). For transparency, triangles in single-input *q*-cells can be replaced with appropriate lines showing the implemented wiring (Fig. 4).

The remaining figures show a *q*-cell-based flat crossing (Fig. 5), AND and OR (Fig. 6), NOR and NAND (Fig. 7), a multiplexer (Fig. 8), a more-than-two-of-three majority function (Fig. 9), a full adder (Fig. 10), and an example of a timer (Fig. 11).

Fig. 1. Graphical representation of a q-cell. The triangles mark inputs, i.e. the cell's sides through which a Boolean value can be received from related neighbor cells. A given cell can have 0, 1, 2, or 3 inputs.



$q = [p\ z\ z] = [[u\ x]\ z\ z]$

Fig. 3. Example of value propagation in a q-cell-based circuit. Assume at time t=0 certain cells return Boolean values u, x, y, and z, respectively; y has to disappear since no neighbor had input allowing for y's propagation; z moves right owing to an input to the right q-cell available. At t=1 the value $p = [u\ x]$ appears as related cell's inputs allowed to receive u and x; p moves right. At t=2 z, having inputs to two different cells available, appears in two places. At t=3 q-cell with three inputs admits values returned by the three related neighbors and produces MEXOR of the values.

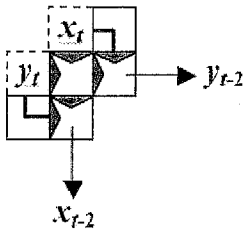


Fig. 5. Flat crossing made from five q-cells.

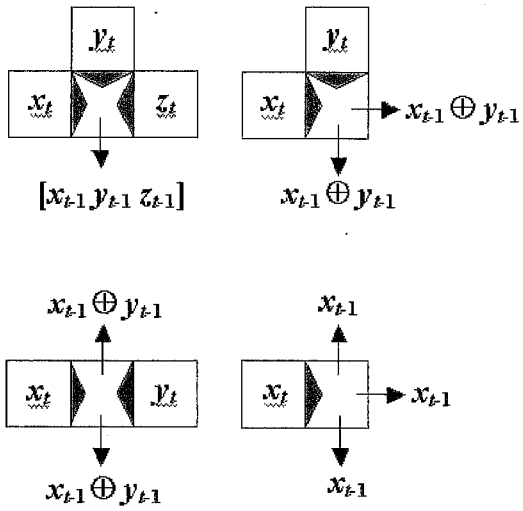


Fig. 2. Four possible layouts of a q-cell's inputs. Depending on the number of defined inputs a q-cell can accept 3 values, 2 values, or 1 value, and return a MEXOR of the values (that can be received by one neighbor cell), EXOR (that can be received simultaneously by two neighbor cells), or the input value unchanged (that can be

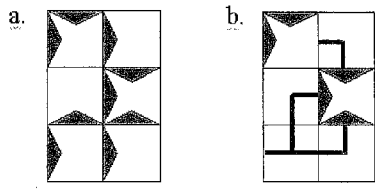


Fig. 4. For clarity, triangles in single-input q-cells can be replaced with appropriate lines to show implemented wiring. Circuit a., modified this way, turns into circuit b.

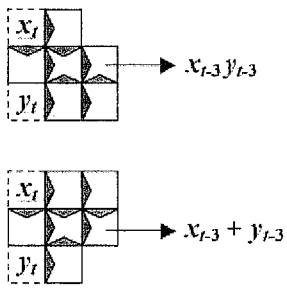


Fig. 6. AND-gate and OR-gate made from q-cells. Both gates return calculated values with a 3-clock delay.

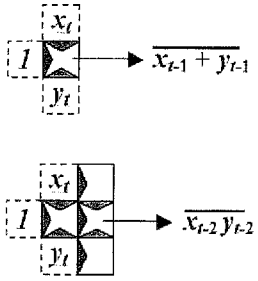


Fig. 7. NOR-gate and NAND-gate made from q-cells. The gates return calculated values with 1-clock and 2-clock delays, respectively.

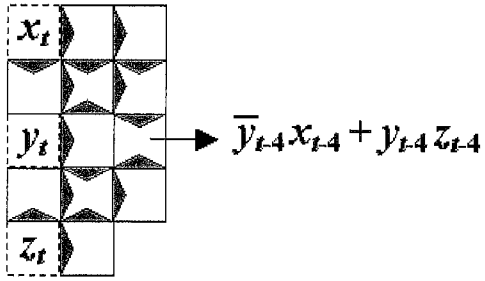


Fig. 8. A planar multiplexer made from eleven q-cells.

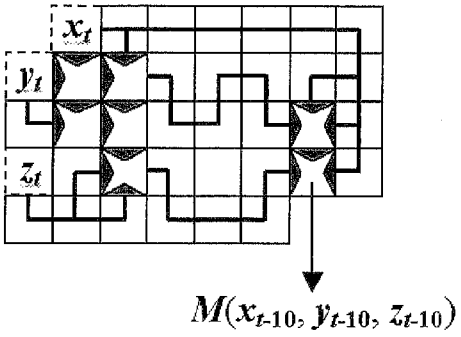
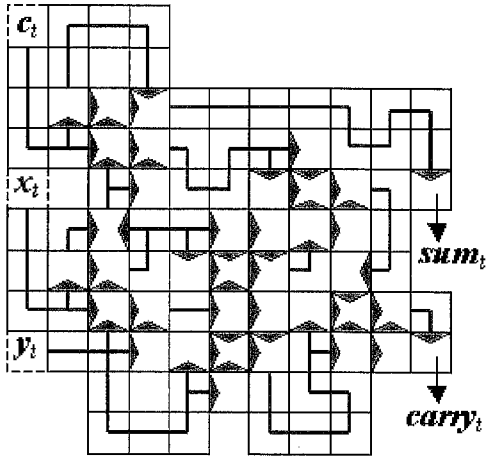


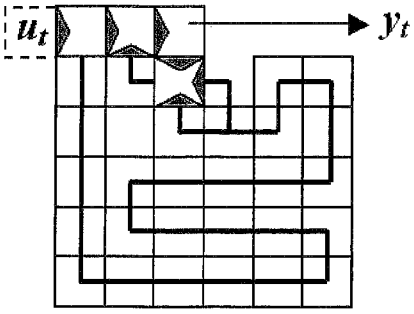
Fig. 9. A planar more-than-one-of-three majority function as a q-cell-based circuit.



$sum_t = x_{t-18} \oplus y_{t-18} \oplus c_{t-18}$

$carry_t = S_2^3(x_{t-22}, y_{t-22}, c_{t-22})$

Fig. 10. A planar adder as a q-cell-based circuit.



$u =$ 10000...

$y =$ 0001000100010001
0001000100010000
00000...

Fig. 11. Example of a timer built as a q-cell-based circuit.

ACKNOWLEDGMENT

This work was supported by the National Institute of Information and Communications Technology of Japan (NICT) under Grant 13-05. I thank Juan Liu and Daniel Jelinski for critical review of the manuscript of this work and for suggestions that helped me to present the ideas more clearly.

REFERENCES

[1] T. Sasao, *Switching Theory for Logic Synthesis*, Boston: Kluwer, 1999, p. 99.

[2] A. Buller, "CAM-Brain Machines and Pulsed Para-Neural Networks: Toward a hardware for future robotic on-board brains" in *Proc. 8th Int. Symposium on Artificial Life and Robotics*, Beppu, 2003, pp. 490-493.

[3] A. Buller, "From *q*-Cell to Artificial Brain," *Journal of Artificial Life and Robotics*, to be published.

[4] M. Korkin, G. Fehr and G. Jeffrey, "Evolving hardware on a large scale," in *Proc. 2nd NASA / DoD Workshop on Evolvable Hardware*, Pasadena, 2000, pp. 173-181.

[5] H. Eeckhaut & J. Van Campemhout, "Handcrafting Pulsed Neural Networks for the CAM-Brain Machine," In *Proc. 8th Int. Symposium on Artificial Life and Robotics*, Beppu, 2003, pp. 494-498.

[6] J. Liu and A. Buller, "Evolving Spike-Train Processors," *Proc. Genetic and Evolutionary Computation Conference (GECCO 2004)*, Seattle, pp. 408-409.

[7] G. D. Hachtel and F. Somenzi, *Logic synthesis and verification algorithms*, Boston: Kluwer, 1996, pp. 94-95.

[8] T. Sasao, *Switching Theory for Logic Synthesis*, Boston: Kluwer, 1999, pp. 44-45.

[9] G. D. Hachtel and F. Somenzi, *Logic synthesis and verification algorithms*, Boston: Kluwer, 1996, p. 471.