Non-Silicon Non-Binary Computing: Why not?

Elena Dubrova Yusuf Jamal Jimson Mathew
Department of Microelectronics and Information Technology
Royal Institute of Technology
Stockholm, Sweden
{elena,jamal,jimson}@ele.kth.se

Abstract

Non-silicon based computing technologies open new possibilities for designing electronic circuits which employ more than two discrete levels of signal. Such circuits, called multiple-valued logic circuits, have a number of theoretical advantages over standard binary circuits. In this paper, we give an introduction to alternative to binary number representations and multiple-valued logic. We discuss possibilities for implementing multiple-valued functions using chemically assembled electronic nanotechnology.

1 Introduction

CMOS-based integrated circuit technology is built on the principle ”a data bit is always zero or one”. This is due in large part to the availability of simple, cheap and reliable CMOS transistors with just two stable states: on and off. However, novel computing technologies based on molecular electronics, quantum mechanics or biological processes open interesting possibilities for utilizing *multiple-valued logic* and alternative to binary number systems.

Multiple-valued logic is a generalization of classical Boolean logic. It provides a theoretical base for designing electronic circuits with more than two logic levels of signal. Multiple-valued logic circuits have a number of theoretical advantages over standard binary circuits. For example, on and off chip interconnect can be reduced if signals in the circuit assume four or more levels rather than only two [1]. In memory design, storing two instead of one bit of information per memory cell doubles the density of the memory in the same die size [2]. Applications using arithmetic circuits often benefit from using alternatives to binary number systems. For example, residue and redundant number systems can reduce or eliminate the ripple-through carries which are involved in normal binary addition or subtraction, resulting in high-speed arithmetic operations [3, 4].

In spite of these potential advantages, practicality of multiple-valued logic design heavily depends on the availability of circuit realizations which must be competitive with present-day binary technologies. The attempts to built integrated circuits employing multi-stable state devices and signals with more than two discrete levels can be traced back to 1970, starting from the early works on ternary designs. Multiple-valued logic circuits have been implemented in bipolar technology, such as integrated injection logic (I^2L) and emitter-coupled logic (ECL), in complementary metal oxide semiconductor (CMOS) technology and in n-type MOS technology [5]-[10]. However, with exceptions of flash [11]-[14] and DRAM [15] memory applications, silicon-based multiple-valued designs are inferior to their binary counterparts [16].

The impressive advances of non-silicon based technologies in recent years brings new hopes for non-binary computing to become a reality. A number of fully functional computing devices employing more than two stable states have been already proposed, including resonant tunneling transistors (RTT), resonant tunneling diodes (RTD) [17]-[20], surface tunneling transistors (STT) [21], [22]. Negative-differential-resistance characteristics which appear in these devices have clear multiple thresholds and therefore are very promising for non-binary computing [23]-[25].

The purpose of this paper is to stimulate the reader's interest beyond conventional binary case. We first give an introduction to alternative to binary number representations and cover a part of multiple-valued logic theory. We then look at the possibilities for implementing multiple-valued logic circuits using molecular electronics. We show how chemically assembled electronic nanotechnology can be used for multiple-valued computing. Finally, we conclude with a discussion of some open problems.

2 Number representation

This section briefly describes potential advantages of non-binary number systems. For a more detailed description the reader is referred to [30].

A digital system represents information with discrete symbols rather than with continuously varying quantity, as in an analog system. Digital binary systems use two symbols, 0 and 1, to represent all information. There are two major conventions for labeling values in a multiple-valued logic system over a set of m values. The most common is $0, 1, 2, \ldots, m-2, m-1$, extending binary notation in one direction only. It is called *unbalanced* (or *unsigned*, or *positive*). The second one requires an odd $m = 2r+1$. It extends binary notation in both directions as $-r, 1-r, \ldots, -1, 0, 1, \ldots, r-1, r$. It is called *balanced* (or *signed*).

A string of digits $(a_{n-1} \ldots a_0)$ over a set of m values represents the number

$$a_{n-1}m^{n-1} + a_{n-2}m^{n-2} + \ldots a_0$$

For example, in the binary case of $m = 2$, $a_i \in \{0, 1\}$. In the ternary case of $m = 3$, $a_i \in \{0, 1, 2\}$ for the unbalanced system, and $a_i \in \{-1, 0, +1\}$ for the balanced system.

One problem with binary number system is the representation of negative numbers. There are three common methods: (1) *sign-magnitude*, where a sign is attached in front of the string of digits; (2) *1's complement*, where the representation for a negative number is obtained by subtracting each digit from 1; (3) *2's complement*, where the same technique as in (2) is used, but with a final addition of 1 to the number.

All three of these techniques suffer from drawbacks. Both (1) and (2) have two representations for 0 (-0 and $+0$), while (3) permits the representation of one more negative number than positive. Alternatively, in a balanced system over a set of m values, all numbers can be represented without using an explicit sign. The sign of a number is the sign of the most significant non-zero digit. Furthermore, in a balanced ternary system, the negative of a number can be computed by interchanging all 1 by -1 an vice versa. Hence, by changing the sign of the addend and subtrahend, we can perform addition and subtraction using the same hardware.

Another problem with binary number system is that, during addition, the sum depends on the carry from lower bits. Two alternative number systems have been studied to reduce or eliminate the ripple-through carries. The first one is *residue* number system, in which there are no carries between bits. In such a representation, operations at each digit occur independently of the other digits, resulting in fast arithmetic operations [4], [27]. A disadvantage is that the size of the digits may vary, and thus different circuits might be needed for processing of different digits.

The second way to represent numbers, allowing to reduce the ripple-through carries, is *redundant* number system. All numbers except 0 have two or more representations by a string of digits. This allows us to make carry dependent on the next two lower digits at most, but no other. Fast multiple-valued arithmetic in redundant balanced number system as well as in redundant unbalanced number system have been presented in [3], [29] and [8], [28], correspondently.

3 Multiple-valued functions

A multiple-valued function is a discrete function whose input and output variables take two or more values. Formally, an n-variable multiple-valued function $f(x_1, \ldots, x_n)$ is a mapping $f : M^n \rightarrow M$, with the variables x_i taking values from the set $M = \{0, 1, 2, \ldots, m-1\}$.

A popular algebraic system for manipulation of multiple-valued functions is *chain-based Post algebra*, corresponding to the first multiple-valued logic developed by Emil Post in 1921 [31]. It is based on a totally ordered set M of elements $0 < 1 < \ldots < m-1$ and use the operations maximum (MAX), minimum (MIN) and literal. These set of operations is functionally complete for multiple-valued functions, meaning that is possible to define any function over M as a composition of MIN, MAX and literals. Completeness is essential for any system which is to be used as a basis for practical logic design. Once a complete set of functions is identified, any logic circuit can be constructed from the gates implementing the primitive functions from this set. For $m = 2$, chain-based Post algebra reduces to a Boolean algebra over $\{0, 1\}$.

4 Chemically assembled electronic nanotechnology

In this section we discuss possibilities for implementing multiple-valued functions using chemically assembled electronic nanotechnology.

4.1 Binary case

A promising alternative to CMOS-based technology is chemically assembled electronic nanotechnology (CAEN) [26]. CAEN exploits the quantum-mechanical effect of nanometer-scale devices. The electronic circuits are constructed by self-alignment and self-assembly of these small devices rather than using lithography. A CAEN switch is a two-terminal device behaving like a diode. Compared to a CMOS transistor, such a switch uses much less power, since only a few electrons are used for switching. Moreover, a two-terminal device can be implemented in CAEN with inexpensive chemical assembly. On the other hand, three-terminal electronic nanotechnology devices require precise alignment to allocate three wires at the device, which makes them unsuitable for producing real circuits [26]. The logic functions are implemented in CAEN using two-terminal configurable switches and diodes only. A consequence is that no inverters can be build, and therefore all logic signals should be available in both complemented and non-complemented form to perform computations.

Due to the constraints associated with the direct assembly of nanometer-scale components, CAEN technology is unsuitable for fabricating non-regular architectures. In [26], an regular architecture called *nanoFabric*, allowing to exploit the advantages of molecular electronics was introduced. The nanoFabric is a two-dimensional homogeneous array of interconnected *nanoBlocks*, which can be configured to implement any logic function. Reconfigurability of nanoFabric is essential for handling fabrication defects. Since CAEN-based devices have a much higher defect density than CMOS-devices, their yield is very low. Thus, post-fabrication self-diagnosis should first be applied to locate the defects. Then, the fabric is configured to implement the desired functionality by routing around the defected spots.

A nanoBlock is a square-shaped molecular array which can be programmed to implement a 3-input 3-output Boolean function and its complement. The inputs are placed on north and east sides of the array and the outputs are taken from the south and west sides of the array. The functions are implemented by programming the configurable switches which lie at each intersection of two orthogonal wires. NanoBlocks can also be used as switches for routing. In addition to local routing between the nanoBlocks, there are long-lines that run between the clusters of nanoBlocks to provide communication over longer distances and promote scalability.

4.2 Multiple-valued case

Next, we show how nanoFabric can be used to implement multiple-valued functions. A number of possibilities for such a generalization exist, depending on which properties of nanoFabric we would like to preserve. We have chosen to keep the following features of nanoFabric:

- the architecture is a two-dimensional homogeneous array;
- it can be configured to implement any logic function;
- only two-terminal devices are used, i.e. diode-resistor logic;

The third point implies that, as in binary case, no inverters are available. First, we discuss what is ”inverter” in multiple-valued case.

In binary case, inverter implements the complement operation defined by $x' = 1 - x$, where x is a Boolean variable $x \in \{0,1\}$. It can easily be generalized to the complement of a multiple-valued variable $x \in \{0,1,2,\dots,m-1\}$ as $x' = (m-1) - x$. The AND operation can be extended to a MIN, whose output is the smaller of the two values of its variables. The set of operations {AND, NOT} is known to be functionally complete for Boolean functions $\{0,1\}^n \rightarrow \{0,1\}$. However, the extended set {MIN, NOT} is *not* functionally complete for multiple-valued functions $M^n \rightarrow M$. For example, it not possible to express the 3-valued 1-variable function $f(x)$ shown in Figure 1 as a composition of MIN and NOT.

x	$f(x)$
0	2
1	0
2	0

Figure 1. A function which cannot be expressed using MIN and NOT.

Our generalization of NOT is not successful because we are trying to replace the binary NOT by a *single* multiple-valued NOT. In fact, it is not possible to extend the binary NOT to any single unary operation which would result in a functionally complete system in combination with MIN [32]. To achieve completeness of an m-valued system, a variable $x \in M$ should have $m-1$ different ”complements” rather than just one. Note that, if $m = 2$, then $m-1$ degenerates 1, so in the binary case this phenomena is hidden.

To make an m-valued system functionally complete, we extend NOT to a set of unary operations which are able to distinguish between the m values of the set M. Such unary operations can be defined as follows.

Definition 1 *A literal of a multiple-valued variable x is a unary operation defined by*

$$x^S = \begin{cases} 1 & \textit{if } x \in S \\ 0 & \textit{otherwise} \end{cases}$$

where $S \subseteq M$.

If $m = 2$, then Definition 1 gives us $x^{\{1\}} = x$ and $x^{\{0\}} = x'$. So, literal is indeed a generalization of NOT. To simplify the exposition, we omit brackets when S is a single-element set, i.e. we write x^1 instead of $x^{\{1\}}$.

In binary case, a consequence of the absence of inverters is that all logic signals should be available in both complemented and non-complemented form [26]. In multiple-valued case, the absence of literal gates implies that the signals for all literals have to be available. For example, if $m = 3$, then a three-valued variable x should be available as x^0, x^1 and x^2. Similarly, the output function f has to be represented as f^0, f^1 and f^2. This way of computing may look complicated at first, a further analysis shows that dealing with literals is much easier than dealing with multiple-valued variables. Moreover, as we show below, it allows us to implement multiple-valued functions with the same diode-resistor logic as the one used for Boolean functions in [26].

Manipulating literals is convenient because a literal is a characteristic function of type $M \rightarrow \{0,1\}$. Therefore, the operations on literals are Boolean operations of type $\{0,1\}^n \rightarrow \{0,1\}$. For example, a sum of two literals $x^i + x^j$ or a product of two literals $x^i \cdot x^j$ are binary Boolean operations of type $\{0,1\}^2 \rightarrow \{0,1\}$. Hence, we can define the sum "+" as OR and the product "$\cdot$" as AND and represent multiple-valued functions as sum-of-products of literals. Next, we show that these three operations are sufficient to define an arbitrary multiple-valued input binary-valued output function of type $M^n \rightarrow \{0,1\}$. The proof can be found in [32, chapter 4].

First, we observe that the constants 0 and 1 can be defined as $0 = \prod_{j \in M} x_i^j$ and $1 = \sum_{j \in M} x_i^j$, for any $i \in \{1,2,\ldots,n\}$. Next, we extend the Shannon decomposition theorem from binary to multiple-valued case as follows.

Theorem 1 *Every multiple-valued input binary-valued output functions $M^n \rightarrow \{0,1\}$ can be decomposed with respect to a variable x_i, $i \in M$, as*

$$f(x_1,\ldots,x_n) = \sum_{j=0}^{m-1} x_i^j \cdot f|_{x_i=j}$$

where $f|_{x_i=j} = f(x_1,\ldots,x_{i-1},j,x_{i+1},\ldots,x_n)$ are co-factors of f with respect to x_i, $\sum$ stands for OR and "$\cdot$" stands for AND.

By subsequently applying Theorem 1, we can decompose an n-variable function with respect to all its variables and derive the following canonical sum-of-products expression. Let "+" = OR, "$\cdot$" = AND and $L = \{x^0, x^1, \ldots, x^{m-1}\}$ be the set of literals specified by Definition 1.

Theorem 2 *Every multiple-valued input binary-valued output function $M^n \rightarrow \{0,1\}$ has a canonical expression in terms of $\{+,\cdot,L\}$ of type*

$$f(x_1,\ldots,x_n) = \sum_{i=0}^{m^n-1} c_i\, x_1^{i_1}\, x_2^{i_2} \ldots x_n^{i_n}$$

where $c_i \in \{0,1\}$ are binary constants, and $(i_1 i_2 \ldots i_n)$ is the m-ary expansion of i with i_1 being the least significant digit.

For example, the function $f : \{0,1,2\}^2 \rightarrow \{0,1\}$ shown in Figure 2 has the following canonical form:

$$f(x_1,x_2) = x_1^0 x_2^0 + x_1^1 x_2^0 + x_1^0 x_2^1 + x_1^2 x_2^1 + x_1^1 x_2^2 + x_1^2 x_2^2.$$

$x_2 \backslash x_1$	0	1	2
0	1	1	0
1	1	0	1
2	0	1	1

Figure 2. An example function.

The set M of m elements together with the set A of functions over M and two distinct elements $\mathbf{0}, \mathbf{1}$ in M generate a finite algebra $\langle M; A; \mathbf{0}, \mathbf{1} \rangle$. For example, the set $B = \{0,1\}$ with the functions AND, OR and NOT generate the Boolean algebra $\langle B; +, \cdot, '; 0, 1 \rangle$. The constants **0** and **1** are two particular elements of M which have special properties. They are called the *zero* and the *unit* of the algebra, respectively. The following result follows directly from Theorem 2.

Theorem 3 *The algebra $\langle M; +, \cdot, L; \mathbf{0}, \mathbf{1} \rangle$, where "+" is OR, "$\cdot$" is AND, $L = \{x^0, x^1, \ldots, x^{m-1}\}$ is the set of literals, $\mathbf{0} = 0$ and $\mathbf{1} = 1$, is functionally complete for multiple-valued input binary-valued output functions $M^n \rightarrow \{0,1\}$.*

It follows from Theorem 3, that we can implement any multiple-valued function using the diode-resistor logic similar to the one used for Boolean functions in [26]. The difference is that, instead of using the complemented and non-complemented values of the signals, we are going to use the literals $x^0, x^1, \ldots, x^{m-1}$ of the input variables and the literals $f^0, f^1, \ldots, f^{m-1}$ of the output functions.

Similarly to the operation of the literal over a variable, x^i, the literal over a function, f^i, is defined by

$$f^i(x_1,\ldots,x_n) = \begin{cases} 1 & \text{if } f(x_1,\ldots,x_n) = i \\ 0 & \text{otherwise} \end{cases}$$

where $i \in M$ is a constant. The literal f^i contains the minterms mapped to i by $f(x_1,\ldots,x_n)$. So, the input domain M^n of f is partitioned into m disjoint sets.

For example, the 3-valued $MIN(x,y)$ function, whose defining table is shown in Figure 3, is partitioned into f^0, f^1, f^2, defined by the following expressions:

$$\begin{aligned} f^0 &= x^0 + y^0 \\ f^1 &= x^1 \cdot y^1 + x^1 \cdot y^2 + x^2 \cdot y^1 \\ f^2 &= x^2 \cdot y^2 \end{aligned}$$

Since, for all $i \in M$, f^i is of type $M^n \rightarrow \{0,1\}$, by Theorem 3 it can always be expressed in terms of AND, OR and literals of the variables $x_1,\ldots,x_n$. Furthermore, since

$y\backslash x$	0	1	2
0	0	0	0
1	0	1	1
2	0	1	2

Figure 3. An 3-valued MIN(x,y) function.

all signals for the literals of the variables are available, any f^i can be implemented by the molecular logic array, similar to the binary molecular logic array proposed in [26].

The implementation of a 3-valued MIN gate in the molecular logic array is shown in Figure 4. Figure 5 shows the implementation of a 3-valued $MAX(x,y)$ gate.

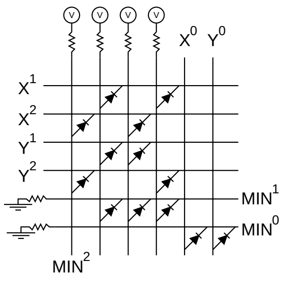

Figure 4. A three-valued MIN gate implemented in the molecular logic array.

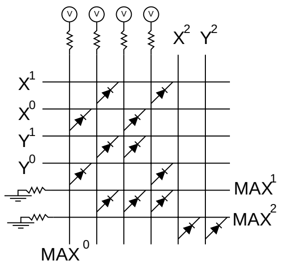

Figure 5. A three-valued MAX gate implemented in the molecular logic array.

Figure 7 shows the implementation of a four-valued adder. The defining tables for sum $s(x,y)$ and carry $c(x,y)$ are shown in Figure 6. From these tables, we can derive the following expressions for the output literals:

$s(x,y)$

$y\backslash x$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$c(x,y)$

$y\backslash x$	0	1	2	3
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	1

Figure 6. Defining tables for 4-valued sum and carry outputs of a four-valued adder.

$$\begin{aligned}
s^0 &= x^0y^0 + x^1y^3 + x^2y^2 + x^3y^1 \\
s^1 &= x^0y^1 + x^1y^0 + x^2y^3 + x^3y^2 \\
s^2 &= x^0y^2 + x^1y^1 + x^2y^0 + x^3y^3 \\
s^3 &= x^0y^3 + x^1y^2 + x^2y^1 + x^3y^0 \\
c^0 &= x^0 + y^0 + x^{\{1,2\}}y^1 + x^1y^2 \\
c^1 &= x^{\{1,2,3\}}y^3 + x^{\{2,3\}}y^2 + x^3y^1 \\
c^2 &= 0 \\
c^3 &= 0
\end{aligned}$$

The literals c^2 and c^3 are constant-zero functions. We show them in the diagram for completeness. The resulting size of the adder is 16×16 grid. If we apply conventional techniques to design a binary circuit of the same functionality (2-bit adder) using 9×9 grid binary half-adder from [26], then we get a larger circuit, since three half-adders and a 2-input OR gate have to be used. However, a more detailed analysis, e.g. using SPICE simulation, is needed to make a fair comparison of both designs.

5 Conclusion

We gave a brief introduction to non-binary number systems and covered a part of the theory of multiple-valued logic related to the design of electronic circuits: multiple-valued functions, functionally complete sets and multiple-valued algebraic systems. We discussed possibilities for implementing multiple-valued functions with chemically assembled electronic nanotechnology.

Future work includes making a more detailed comparison of multiple-valued and binary designs by modeling and simulating them using SPICE.

A challenge to practical utilization of multiple-valued logic is creation of an effective computer-aided design package. Although some of the concepts and algorithms necessary for such a package have been already developed [33], a significant amount of research remains to be done to make it competitive with standard CAD packages.

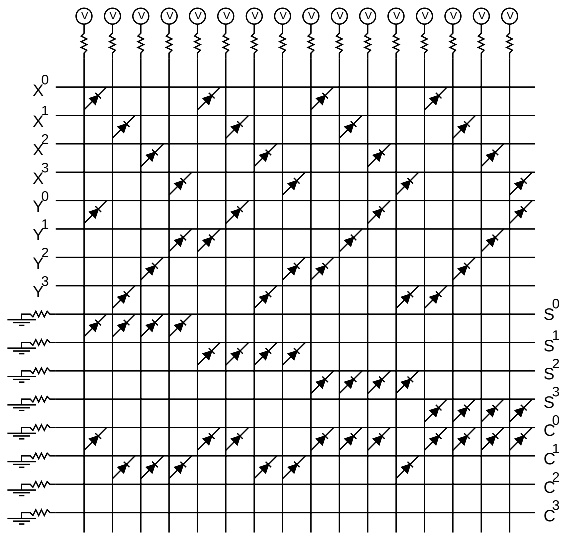

Figure 7. A four-valued adder implemented in the molecular logic array.

References

[1] I. Ben Dhaou, E. Dubrova and H. Tenhunen, Power efficient inter-module communication for digit-serial architectures in deep-submicron technology, *Proc. 30th Int. Symp. Multiple-Valued Logic*, (2000), 61-67.

[2] B. Ricco, G. Torelli, M. Lanzoni, A. Manstretta, H. E. Maes, D. Monatanari and A. Modelli, Non-volatile multilevel memories for digital applications, *Proc. IEEE*, **86**, 12, (1998), 2399-2421.

[3] T. Hanyu, M. Kameyama, A 200 MHz pipelined multiplier using 1.5 V-supply multiple-valued MOS current-mode circuits with dual-rail source-coupled logic, *IEEE Journal of Solid-State Circuits*, **30**, 11, (1995), 1239-1245.

[4] K. Shimabukuro, C. Zukeran, Reconfigurable current-mode multiple-valued residue arithmetic circuits, *Proc. 28th Int. Symp. Multiple-Valued Logic*, (1998), 282-287.

[5] K. C. Smith, The prospects for multivalued logic: A technology and applications view, *IEEE Trans. on Computers*, **C-30**, 9, (1981), 619-634.

[6] S. L. Hurst, Multiple-Valued Logic - its status and its future, *IEEE Trans. om Computers*, **C-33**, 12, (1984), 1160-1179.

[7] E. Dubrova, Multiple-Valued Logic in VLSI Design, *Multiple-Valued Logic, An International Journal*, (2002).

[8] B. Radanovic, M. Syrzycki, Current-mode CMOS adders using multiple-valued logic *Proc. Canadian Conference on Electrical and Computer Engineering*, (1996), 190-193.

[9] J. Shen et al., Neuron-MOS current mirror circuit and its application to multi-valued logic, *IEICE Trans. Inf. & Syst.*, **E82-D**, (1999), 940-948.

[10] D. H. Y. Teng, R. J. Bolton, A self-restored current-mode CMOS multiple-valued logic design architecture, *Proc. 1999 IEEE pacific Rim Conf. on Communications, Computers and Signal Processing (PASRIM'99)*, (1999), 436-439.

[11] K. Takeuchi et al., A multipage cell architecture for high-speed programming multilevel NAND flash memories, *IEEE J. Solid-State Circuits*, **33**, 8, (1998), 1228-1238.

[12] T.-S. Jung et al., A 117-mm^2 3.3-V only 128-Mb multilevel NAND flash memory for mass storage applications *IEEE J. Solid-State Circuits*, **31**, 11, (1996), 1575-1583.

[13] A. Nozoe et al., A 256 Mb multilevel flash memory with 2 MB/s program rate for mass storage applications, *Proc. of 1999 IEEE Int. Solid-State Circuits Conference (ISSCC'99)*, (1999), 110-111.

[14] M. Ohkawa et al., A 98-mm^2 die size 3.3-V 64-Mb flash memory with FN-NOR type four-level cell, *IEEE J. Solid-State Circuits*, **31**, 11, (1996), 1584-1589.

[15] T. Okuda, Advanced circuit technology to realize post giga-bit DRAM, *Proc. 28th Int. Symp. Multiple-Valued Logic*, (1998), 2-5.

[16] D. Etiemble, M. Israel, Comparison of binary and multivalued ICs according to VLSI criteria, *Computer* **21**, 4 (1988), 28-42.

[17] T. Itoh et al., 10G Hz operation of multiple-valued quantizers using resonant tunneling diodes, *IEICE Trans. Inf. & Syst.*, **E82-D**, (1999), 949-954.

[18] T. Waho et al., Multi GHz A/D converter using resonant-tunneling multiple-valued logic circuits *Proc. of 1998 IEEE Int. Solid-State Circuits Conference*, (1998), 258 - 259.

[19] T. Waho et al., A novel multiple-valued logic gate using resonant tunneling devices, *Proc. 29th Int. Symp. Multiple-Valued Logic*, (1999), 2-8.

[20] T. Waho et al., Resonant-tunneling diode and HEMT logic circuits with multiple thresholds and multilevel output, *IEEE J. Solid-State Circuits*, **33**, 2, (1998), 268-274.

[21] T. Baba, Development of quantum functional devices for multiple-valued logic circuits, *Proc. 29th Int. Symp. Multiple-Valued Logic*, (1999), 2-8.

[22] T. Uemura, T. Baba, Demonstration of a novel multiple-valued T-gate using multiple-junctions surface tunnel transistor and its applications to three-valued data flip-flop, *Proc. 30th Int. Symp. Multiple-Valued Logic*, (2000), 305 -310.

[23] H. L. Chan et al., Compact multiple-valued multiplexers using negative differential resistance devices, *IEEE J. of Solid-State Circuits*, **31**, 8, (1996), 1151-1156.

[24] K.-J. Gan, Y.-K. Su, Novel multipeak current-voltage characteristics of series-connected negative differential resistance, *IEEE Electron Device Letters*, **19**, 4, (1998), 109-111.

[25] W.-C. Liu et al., Multiple negative differential resistance phenomena of metal-insulator-semiconductor-insulator-metal (MISIM)-like structure with step-composed $\mathrm{In}_x\mathrm{Ga}_{1-x}\mathrm{As}$ quantum wells, *IEEE Trans. on Electron Devices*, **45**, 2, (1998), 373-379.

[26] S. C. Goldstein, M. Budiu, NonFabrics: Spatial computing using molecular electronics, *Proc. of the 28th Annual Int. Symp. on Computer Architecture*, (2001)

[27] S. Wei, K. Shimizu, Residue arithmetic multiplier based on the radix-4 signed-digit multiple-valued arithmetic circuits, *12th Int. Conf. on VLSI Design*, (1999), 212-217.

[28] O. Ishizuka, D. Handoko, VLSI design of a quaternary multiplier with direct generation of partial products, *Proc. 27th Int. Symp. Multiple-Valued Logic*, (1997), 169-174.

[29] A. F. Gonzalez, P. Mazumder, Multiple-valued signed digit adder using negative differential resistance devices, *IEEE Trans. on Computers*, **47**, 9, (1998), 947 -959.

[30] J. C. Muzio and T. C. Wesselkamper, *Multiple-valued switching theory*, Bristol, Hilger, 1986.

[31] E. L. Post Introduction to a general theory of elementary propositions, *Amer. J. Math.*, **43**, (1921) 163-185.

[32] S. Hassoun and T. Sasao, eds., *Logic Synthesis and Verification*, Kluwer Academic Publishers, 2002.

[33] M. Gao, J.-H. Jiang, Y. Jiang, Y. Li, S. Sinha and R. Brayton, MVSIS, in *Proc. Int. Workshop on Logic Synthesis*, (2001), 138-144.