A FAST ARCHITECTURE FOR FINDING MAXIMUM (OR MINIMUM) VALUES IN A SET

Andrea Dario Giancarlo Biroli, Juan Chi Wang

Dipartimento di Elettronica e Telecomunicazioni, Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino 10129, Italy; E-Mail: andrea.biroli@polito.it

ABSTRACT

High speed architectures for extracting the best (maximum or minimum) values in a given set and their positions is of high importance in many signal processing applications. For example, the search of the two minimum values is an important part in iterative channel decoders for turbo and low-density-parity-check codes. In this paper, a new architecture is proposed to find the g^{th} best value in an unsorted list of k elements, where the ranking position g can be any integer between 1 and k. The architecture can also be used to find in the assigned set a generic subset of the largest or smallest values. The proposed solution is particularly efficient in terms of hardware complexity and latency when the cardinality of the assigned set k is large and the values are represented on a reduced number of bits n. Synthesis results obtained with a 90-nm CMOS standard cell technology are provided for specific choices of g, k and n. Moreover, the nice properties of regularity and scalability of the proposed architecture are exploited to develop a QCA (quantum cellular automata) based implementation, which achieves lower power consumption or higher speed.

***Index Terms*— Data processing, search methods, digital arithmetic, Application Specific Integrated Circuits, quantum cellular automata**

1. INTRODUCTION

Finding one or multiple maximum/minimum (max/min) values and/or its position in an unsorted list of elements plays an important role in a wide range of applications, such as bioinformatics [1], video processing [2], channel decoding [3], computer networks [4], real-time systems [5], sorting networks [6], artificial neural networks [7], and MIMO (Multiple Input Mutiple Output) systems [8]. In particular, several simplified algorithms proposed for decoding low-density-parity-check (LDPC) and turbo codes [9] [10] [11] need to find the first and second max/min in a set of k elements. The selection of more than two max/min values is adopted for example in MIMO detection [8], non-binary LDPC decoding [12] and joint source channel coding [13].

Several algorithms for max/min search and partial sorting are known in computer science and lead to efficient software implementation. However, in many of the mentioned fields of application, low latency or high throughput requirements are not compatible with the software approach and demand for dedicated hardware solutions, able to solve the following general problem: given an unsorted list of k elements, $L = (x_0, x_1, \dots x_{k-1})$, which represent binary values on n bits, the m highest (or lowest) values are extracted, together with their indexes in the original list L. Despite of the wide diffusion of the general problem in several signal processing applications, quite a low number of implementations have been proposed in the open literature. Efficient solutions for the case $m = 1$ (i.e. search of the single max/min) have been reported in [14]. Implementation architectures for the case $m = 2$ are available in [15], [16]. In [17] a reduced sorting network constraints $m = 2^i, i \in \mathbb{N}$ to solve the problem.

In this paper, the problem of extracting the g^{th} best value in an unsorted list of k elements, is addressed, where the ranking order g can be any integer between 1 and k. This problem can be stated as follows. Said max_1 the maximum value in a list L, $max_1 = max(L)$, the second maximum max_2 can be found by selecting the maximum value in a new list, obtained by removing the max_1 element from L, $max_2 = max(L \setminus max_1)$, where the $\setminus$ operator indicates the operation of removing an element from a list. The g^{th} largest element in L, max_g, can be obtained by iteratively finding the maximum values in a sequence of lists, each one generated by removing from the previous list the largest elements:

$$max_g = max(L \setminus max_1^{g-1}) \tag{1}$$

where $max_1^{g-1} = (max_1, max_2, \dots, max_{g-1})$ is the list of $g-1$ previously identified maximum values. According to (1), repetitions of the same value in a list leads to multiple extractions of the same value.

Using the same notation, the equivalent problem can be defined for the search of the minimum value of order g: $min_g = min(L \setminus min_1^{g-1})$.

A new architecture is proposed to efficiently solve this problem. The proposed solution is scalable with respect to order g, number of elements in the input list k, and bit width n. The closely related work is briefly reviewed in Section 2. The proposed architecture is given in section 3, together with synthesis results and comparisons with similar implementations. Finally, the mapping of the same architecture on QCA

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

(Quantum Cellular Automata) technology is presented in Section 4. Final conclusions are drawn in Sections 5.

2. PRIOR WORK

The most important circuit topologies proposed for the first max/min search [14] are: Array Topology (AT), Row Topology (RT), Selection Topology (ST), Traditional Binary Tree (TBT), Parallel Binary Tree (PBT), Multi-Level (ML), Leading-Zero Counter (LZC) and Array Based (AB). AT works as a filter in which all candidates are examinated in parallel from MSB to LSB, in order to progressively reduce the number of candidates at each bit-slice. RT can be seen as a single AT row used serially multiple times, with proper registers to store results. ST operates on bit-slices and, at each iteration, only good candidates are taken and passed to the following refined selection process. In TBT topology, simple blocks composed by a multiplexer driven by a comparator are connected in a binary tree configuration. To reduce the overall propagation delay of TBT, the partial carry evaluated by the comparator is used in a binary single bit multipler tree. This Parallel Binary Tree (PBT) achieves relevant speed-up, due to the fact that the descendant block can start comparison before the end of comparison in the predecessor.

Array based (AB) topology explicitly evaluates the greater of all possible combinations of inputs. In ML [18], inputs are divided in separate groups and the winner in each group is determined; then, a parallel comparison similar to AB is applied. LZC [18] is made of two cascade stages. In the 1^{st} stage, value of the maximum is determined, while, in the 2^{nd} stage, the address of the maximum is computed. In the 1^{st} stage, each one of k elements is converted into a 2^n-bit one-hot code. Then, a bit-wise OR operation is applied on these k 2^n-bit elements to obtain a 2^n-bit vector, in which the position of the first one gives the searched value.

Tree–like structures are used in [15] to find the first two max/min values in a set of k elements: the obtained results exhibit remarkably small area and delay. In [16], a large variety of low latency architectures for solving the same problem are given. The considered solutions include radix–2, radix–3 and mixed radix approaches, which cover a wide spectrum of possible area–speed trade–offs.

The cited works do not address the search of the generic g^{th} max/min value, which is the specific target of this paper. The proposed solution can be considered as a modified LZC architecture, extended to handle any generic g. Moreover this architecture can be easily scaled to cover wide ranges of k, n and g, or pipelined, according to the specific performance goals required (eg. timing, area, etc.).

3. PROPOSED ARCHITECTURE AND SYNTHESIS

The proposed architecture (PA) is based on the idea of comparing in parallel the received k values, $x_0, x_1, \ldots x_{k-1}$, against all possible binary values, for 0 to $2^n - 1$. From the comparisons, a thermometric representation is derived for each incoming value: said x_i the $i-th$ input, the corresponding thermometric representation is $c_{i,0}, c_{i,1}, c_{i,2}, \ldots, c_{i,2^n-1}$ where bit $c_{i,j}$ is equal to 1 if $x_i > j$ and 0 otherwise, with $j = 0, 1, 2, \ldots, 2^n - 1$. The generation of the thermometric representation for the k inputs is performed by the first block, COMP, in the high level architecture given in Figure 1 (details in Figure 1-(a)). Starting from the thermometric representation, the number of inputs larger or equal to value j is derived and compared to g. This task, performed by unit MUX_SEL in Figure 1-(b), where output sel_j^g is equal to 1 if the number of received inputs that pass threshold j is larger or equal to g. Algebraically, this condition can be stated as

$$\begin{aligned} sel_j^1 = \;& c_{0,j} + c_{1,j} + c_{2,j} + \ldots + c_{k-1,j} \\ sel_j^2 = \;& c_{0,j} \cdot c_{1,j} + c_{0,j} \cdot c_{2,j} + \ldots + c_{0,j} \cdot c_{k-1,j} + \\ & c_{1,j} \cdot c_{2,j} + c_{1,j} \cdot c_{3,j} + \ldots + c_{1,j} \cdot c_{k-1,j} + \\ & \ldots + c_{k-2,j} \cdot c_{k-1,j} \\ sel_j^3 = \;& c_{0,j} \cdot c_{1,j} \cdot c_{2,j} + c_{0,j} \cdot c_{1,j} \cdot c_{3,j} + \ldots + \\ & c_{0,j} \cdot c_{1,j} \cdot c_{k,j} + c_{1,j} \cdot c_{2,j} \cdot c_{3,j} + \ldots + \\ & c_{1,j} \cdot c_{2,j} \cdot c_{k,j} + \ldots + c_{k-3,j} \cdot c_{k-2,j} \cdot c_{k-1,j} \end{aligned} \tag{2}$$

Complexity of (2) grows linearly with k, as parallel prefix approach can be adopted to efficiently calculate partial terms with no repetitions. For example, in the $g = 2$ case, (2) can be rewritten as

$$\begin{aligned} & sel_j^2 = c_{k-2,j} \cdot c_{k-1,j} + c_{k-3,j} \cdot S_{k-1,k-2} + \\ & c_{k-4,j} \cdot S_{k-1,k-3} + \ldots + c_{0,j} \cdot S_{k-1,0} \end{aligned} \tag{3}$$

where $S_{k,l} = \sum_{h=k}^{l} c_{h,j} = S_{k,l-1} + c_{l,j}$. Calculation of sel_j^g for large g can be organized in similar recursive way, so obtaining a linear growing of complexity with k.

Selection bits generated by MUX_SEL unit are finally used to control a bank of multiplexers, which propagate the $g-th$ maximum to the output. A linear multiplexer structure is shown in Figure 1-(c), where each stage receives one of the 2^n possible values and the corresponding selection signal sel_j^g. Shorter latency is provided by better organization of the multiplexer, e.g. a tree like structure may reduce the latency from 2^n to n.

The bit based approach adopted in the architecture design leads to an area complexity growing fast with the data bit width. On the contrary, complexity depends on the number of elements k in a much weaker way. In particular, the area complexity and combinational delay can be expressed as

$$\begin{aligned} & A(k, n, g) = O(g \cdot 2^n \cdot (n + k)) \\ & D(k, n) = O(log_2(k) + log_2(n) + n) \end{aligned} \tag{4}$$

Based on (4), we can state that the proposed architecture is efficient for applications characterized by quite a low bit–width and a large set of elements. As an example, belief propagation decoding can be used for LDPC codes with small n (between 6 and 8) and k depending on node degree, which can be 32

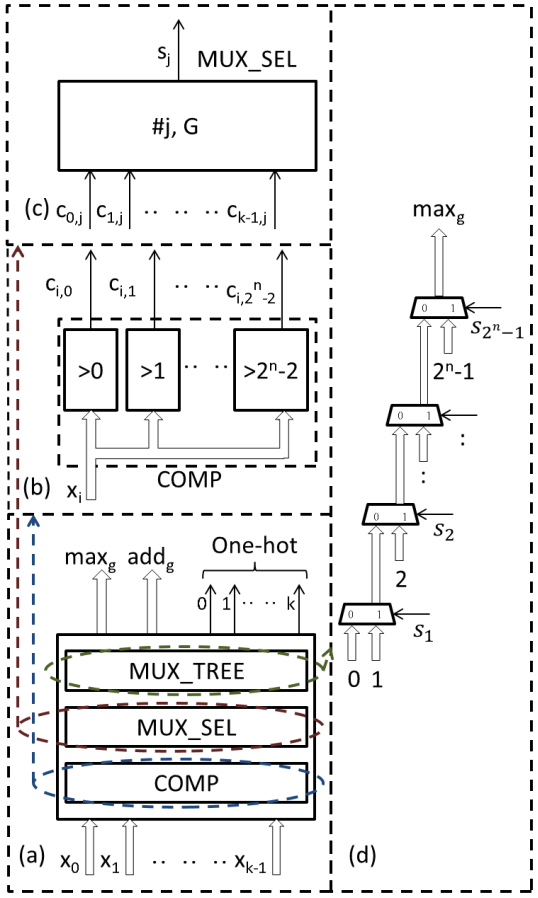

Fig. 1. Overall architecture (a), and main units: parallel comparators (b), generation of select signals(c), multiplexers (d)

or more. To compare the potential of the described solution against alternative architectures, two recent implementations have been considered: one of them [14] extracts the first maximum from the assigned list and it is used to make comparison in the $g = 1$ case, while the second implementation [16] finds the first two maximum values and allows for a comparison in the $g = 2$ case. Comparisons are given in Table 1,

The proposed architecture has been described using VHDL, validated for a wide range of n, k and g, and finally synthesized using the Z-2007.03-SP1 version of Synopsys Design Compiler and typical standard cell library 90nm CMOS technology. Collected synthesis results for both area and delays are reported in Figure 2, where $n = 8$, $g = 2$ and k ranges between 8 and 20. The shown results confirm the general high level estimations in (4).

The reported delay values are related to the fully combinational version of the architecture, which is able to process one elements list per cycle and return the wanted g^{th} element in the same clock cycle. A pipelined version of the architecture has also been generated, to increase the processing throughput, at the cost of a longer latency: due to the regular feed–forward structure of the architecture, the pipelined version can reach a clock period as low as 0.32 ns, when synthesized on the same 90 nm technology.

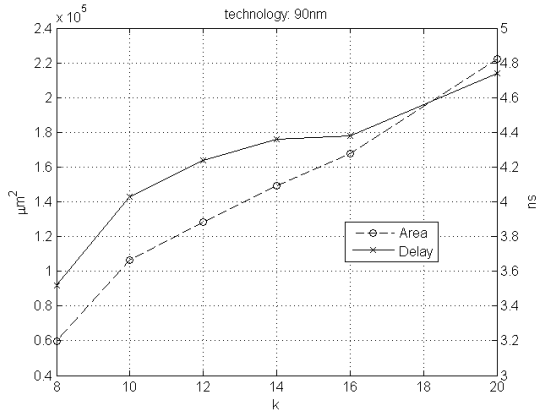

Fig. 2. Post synthesis area(left axis) and delay (right axis) results for $g = 2$, $n = 8$.

4. QCA IMPLEMENTATION

We implemented the proposed architecture on an emerging technology, Quantum Dot Cellular Automata [19], using the method we applied in [20] and [21]. The aim was to verify to what extent improvements can be derived by the use of emerging technologies instead of scaled CMOS transistors. In QCA technology identical bistable cells are used to represent logic values. Information propagation is driven by the interaction among neighbor cells. There are two main implementations of this principle: NanoMagnet Logic (NML) [22] where the basic cell is a single domain nanomagnet (Fig. 3.A) and Molecular QCA [23] which uses complex molecules instead. The former is promising for its low power consumption [22], the second is is expected to reach, theoreticallly, very high clock frequency: about 1 THz [23]. Circuits are built aligning cells on a plane. An example of NML circuit is shown in Fig. 3.A, where magnets are 50x100 nm^2. Molecular QCA circuits have a similar structure. Cells are driven in an intermediate (RESET) state with the help of an external force, which is a magnetic field or a mechanical strain for NML technology, while an electric field is required for molecular QCA. When the external mean is removed cells switch in the correct state (Fig. 3.A). The maximum number of cells that can switch without computational errors due to thermal noise is limited. Circuits are therefore divided in small areas (clock zones) including a limited number of cells Fig. 3.B). The same clock signal is applied to each zone, but with a phase difference of 120°. This assures the correct signals propagation. This characteristic gives the circuit an intrinsic pipelined behavior. Every group of three consecutive clock zones has a delay of a exactly one clock cycle. This feature can be exploited to

Table 1. Comparison of proposed architecture and alternative solutions for $g = 1, 2$.

	PA		AB[14]	
	Time delay	Area complexity	Time delay	Area complexity
$m = 1, g = 1$	$O(log_2(k) + log_2(n) + n)$	$O(g \cdot 2^n \cdot k + 2^n \cdot n)$	$O(log_2(k) + log_2(n))$	$O(k^2 \cdot (n + 1) + k)$
			MRA level=1 [16]	
$m = 2, g = 1, 2$	$O(log_2(k) + log_2(n) + n)$	$O(g \cdot 2^n \cdot k + 2^n \cdot n)$	$O(log_2(n) + 4log_2(k))$	$O(k^2 \cdot n + 4k)$

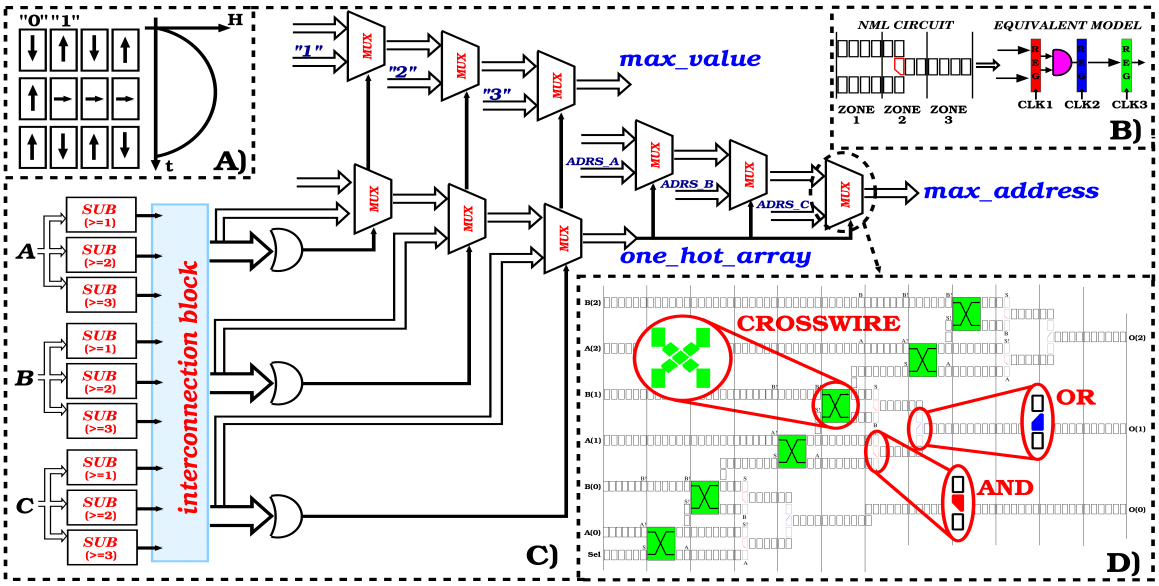

Fig. 3. A) NanoMagnet Logic (NML). Single domain nanomagnets represent logic values '0' and '1'. A clock mechanism is used to switch magnets from one state to the other. B) Circuits are divided in small areas (clock zones) made by a limited number of elements. At each clock zone a different clock signal is applied. This mechanism is used to develop a behavioral model written in VHDL. Registers emulate the propagation delay while ideal logic gates model the logic functions. C) Block schematic of the QCA circuit. D) Detailed layout of a multiplexer.

write a behavioral model of QCA circuits [24]. Registers emulate the propagation delay while ideal logic gates represent the logic function. This model can be described using VHDL language. Further details on the model and QCA technology can be found in [24], [22] and [23].

Figure 3.C reports the block schematic of the circuit implemented using QCA, while Figure 3.D shows the detailed layout of a multiplexer. The multiplexer is implemented with nanomagnets, but the molecular version has a similar layout. Simulations results are reported in Table 2. The NML version has a significant advantage over the CMOS implementation in terms of power, but it is slower and the area is much bigger. The molecular implementation reaches a very high frequency and the circuit is extremely small, but at the cost of an increased power consumption. Data from CMOS are estimated starting from the synthesis on the 90 nm technology node with the help of the ITRS roadmap.

Table 2. Area, power and frequency estimation

Case: Nbit=8 and K=8	Area (μm^2)	Power (mW)	Clock (GHz)
NML	52560	0.119	0.1
Mol. QCA	63	351	1000
CMOS 21nm	2312	0.89	0.488

5. CONCLUSIONS

In this paper high speed architecture for finding generic max/min values and their addresses is presented. The proposed solution extends previous works on LZC architectures and shows by experimental results that PA achieves lower period than other architectures, to the best of authors knowledge. Nevertheless area cost are not negligible. We have also demonstrated that the use of emerging technologies, like QCA, gives considerable advantages in terms of power consumption or clock frequency.

6. REFERENCES

[1] H. Seker A.T. Erdogan H.M. Hussain, K. Benkrid, "FPGA implementation of k-means algorithm for bioinformatics application: An accelerated approach to clustering microarray data," in *Conference on Adaptive Hardware and Systems*, 2011, pp. 248–255.

[2] B.Y. Hsieh L.G. Chen Y.W. Huang, S.Y. Chien, "Global elimination algorithm and architecture design for fast block matching motion estimaation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 6, pp. 898–907, 2004.

[3] N.R. Shanbhang M.M. Mansour, "High-throughput LDPC decoders," *IEEE Trans. Very Large Scale Integr.*, vol. 11, no. 6, pp. 976–996, Dec. 2003.

[4] B. Bhagyavati S.Kurkovskyt M. Yang, S.Q. Zheng, "Programmable weighted arbiters for constructing switch schedulers," in *Workshop on High Performance Switching and Routing*, 2004, pp. 203–206.

[5] B. Jacob P. Kohout, B. Ganesh, "Hardware support for real-time operating systems," in *Int. Conf. on Hardware/Software Codesign and System Synthesis*, 2003, pp. 45–51.

[6] T. Nakano T. Ando K. Shirai S. Azuma, T. Sakuma, "High-performance sort chip," in *A Symposium on High Performance Chips*, 1999.

[7] D.C. Hendry, "Comparator trees for winner-take-all circuits," *Neurocomputing*, vol. 62, pp. 389–403, 2004.

[8] B. Cerato, G. Masera, and E. Viterbo, "Decoding the golden code: A VLSI design," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, no. 1, pp. 156–160, 2009.

[9] J.L. Danger F. Guilloud, E. Boutillon, "λ-min decoding algorithm of regular and irregular LDPC codes," in *Proc. Int. Symp. Turbo Codes Related Topics*, 2003, pp. 451–454.

[10] M. Martina, G. Masera, S. Papaharalabos, P.T. Mathiopoulos, and F. Gioulekas, "On practical implementation and generalizations of $\max^*$ operator for turbo and LDPC decoders," *IEEE Trans. onInstrumentation and Measurement*, vol. 61, no. 4, pp. 888–895, 2012.

[11] C. Condo, M. Martina, and G. Masera, "VLSI implementation of a multi-mode turbo/LDPC decoder architecture," *IEEE Trans. Circuits and Systems I*, vol. 60, no. 6, pp. 1441–1454, 2013.

[12] E. Boutillon and L. Conde-Canencia, "Simplified check node processing in nonbinary ldpc decoders," in *Int. Symp. on Turbo Codes and Iterative Information Processing*, 2010, pp. 201–205.

[13] S. Zezza, S. Nooshabadi, M. Martina, and G. Masera, "Efficient implementation techniques for maximum likelihood-based error correction for jpeg2000," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 4, pp. 591–596, 2009.

[14] S. Goren G. Dundar B. Yuce, H.F. Ugurdag, "A fast circuit topology for finding the maximum of N k-bit numbers," in *Symp. on Computer Arithmetic*, 2013, pp. 59 – 66.

[15] Chin-Long Wey, Ming-Der Shieh, and Shin-Yo Lin, "Algorithms of finding the first two minimum values and their hardware implementation," *IEEE Trans. Circuits and Systems I*, vol. 55, no. 11, pp. 3430–3437, 2008.

[16] G. Masera L.G. Amarú, M. Martina, "High speed architectures for finding the first two maximum/minimum values," *IEEE Trasactions on Very Large Scale Integration*, vol. 20, no. 12, pp. 2342–2346, 2012.

[17] A. Farmahini-Farahani and al., "Modular design of high-throughput, low-latency sorting units," *Computers, IEEE Trans.*, vol. 62, no. 7, pp. 1389–1402, 2013.

[18] S.B. Zhang F. Zhang X.P. Huang, X.Ya. Fan, "An optimized tag sorting circuit in wfq scheduler based on leading zero counting," in *Int. Conf. on Solid-State and Integrated Circuit Technology*, 2010, pp. 533–535.

[19] C.S. Lent, P.D. Tougaw, W. Porod, and G.H. Bernstein, "Quantum cellular automata," *Nanotechnology*, vol. 4, pp. 49–57, 1993.

[20] M. Awais, M. Vacca, M. Graziano, and G. Masera, "FFT Implementation using QCA ," *Int. Conf. on Electronics, Circuits and Systems*, pp. 741–744, 2012.

[21] M. Awais, M. Vacca, M. Graziano, and G. Masera, "Quantum dot Cellular Automata Check Node Implementation for LDPC Decoders ," *IEEE Transaction on Nanotechnology*, pp. 368–377, 2013.

[22] M. Vacca, M. Graziano, and M. Zamboni, "Majority Voter Full Characterization for Nanomagnet Logic Circuits," *IEEE T. on Nanotechnology*, vol. 11, no. 5, pp. 940–947, Sept. 2012.

[23] A. Pulimeno, M. Graziano, D. Demarchi, and G. Piccinini, "Towards a molecular QCA wire: simulation of write-in and read-out systems," *Solid State Electronics*, vol. 77, pp. 101–107, 2012.

[24] M. Vacca, M. Graziano, and M. Zamboni, "Nanomagnetic Logic Microprocessor: Hierarchical Power Model," *IEEE T. on VLSI Systems*, pp. 1410–1420, Aug. 2012.