Majority-based Library Generation for QCA, SET and TPL Technologies

Mayler G. A. Martins[2], Vinicius Callegaro[1],Stephano Gonçalves[3], Melissa Colvara[3], Leomar S. da Rosa Jr. [3], Felipe de Souza Marques[3,] André I. Reis[1,2], Renato P. Ribas[1,2]

[1]*PPGC* / [2]*PGMICRO, Institute of Informatics, UFRGS, Porto Alegre, RS, Brazil*

[3] *Group of Architectures and Integrated Circuits UFPEL, Pelotas, RS, Brazil*

{mgamartins,vcallegaro,rpribas,andreis}@inf.ufrgs.br
{smmgoncalves,mcolvara,leomarjr,felipem}@inf.ufpel.edu.br

***Abstract* — CMOS technology scaling is reaching its physical limits, and new nanometric devices are being considered. Some of these technologies, such as quantum cellular automata (QCA), single electron tunneling (SET) and tunneling phase logic (TPL), use the majority voter and inverter as basic Boolean primitive elements. Commercial tools seem to be not able to synthesize circuits efficiently using only majority gates and inverters. In order to overcome this bottleneck, we propose a new approach that is able to generate cell libraries with up to 4-input Boolean functions using only majority and inverter gates. Previous approaches can only build libraries with up to 3-input Boolean functions. Experimental results over MCNC benchmarks have demonstrated that there is a significant reduction up to 41.7%, 19.6%, 16.2% and 79.6%, in logic depth, majority gate count, gate inputs and inverters, respectively, compared to the existing methods.**

***Keywords* — Functional composition, digital circuits, QCA, majority gate, cell library, logic gates.**

I. INTRODUCTION

The complementary metal-oxide semiconductor (CMOS) technology is reaching its physical limits. There are many challenges, as short channel effect, variability and even the difficulty to create masks at nano-scale. There are new candidates to replace the CMOS technology, such as quantum cellular automata (QCA) [1]. QCA can be used to design general-purpose computational and memory circuits, and it is expected to achieve high device density, extremely low power consumption, and very high switching speed. Tunneling phase logic (TPL) [2] and single electron tunneling (SET) [3] are also good candidates to replace the current CMOS technology. Those technologies are illustrated in Fig. 1.

All such technologies use majority gates as primitive logic elements. A majority gate is a simplified version of a threshold logic gate, where the input weights have the same value and the output goes to '1' when more than a half of inputs presents the logic level '1'. The output goes to '0', otherwise. A minority gate is the complemented version of a majority gate. The basic logic elements in these technologies are the inverters, common for all technologies (QCA, SET, TPL), and the majority gate for QCA and the minority gate for SET and TPL.

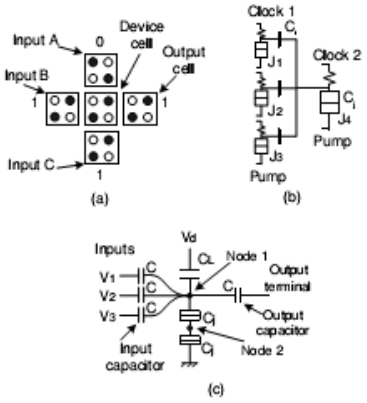

Figure 1: QCA majority gate (a), TPL minority gate (b), and SET minority gate (c) [4].

The majority logic synthesis is a sub-area from the threshold logic synthesis. Threshold logic synthesis research dates back to 1960's when Akers [12], Miller and Winder [5] and Muroga [6] employed several logic synthesis methods and techniques to generate threshold/majority gate circuits efficiently. Modern logic synthesis algorithms start from 2-level expressions (that can be generated by tools such as ESPRESSO [7]) represented as sum-of-products (SOP) or product-of-sums (POS) to generate a reduced factored expression. Factored expressions are used to reduce the final area of the circuit. Unfortunately, the SOP/POS cannot be used to effectively generate majority circuits due to the lack of algorithms that efficiently convert a two level expression in a majority circuit. In this sense, it is critical an efficient technique to synthesize circuits using majority gates as primitive elements, in order to enable further development of QCA, SET, TPL, and other emergent technologies.

This paper explores the Functional Composition (FC) paradigm [8] to generate a library to synthesize circuits using only majority gates and inverters as primitive logic elements. There are two possible approaches for circuit synthesis using majority gates: using or not a cell library to aid the circuit synthesis. This paper addresses a cell library based technique. The cell-library based technique is an interesting approach, since there are several commercial tools that perform technology mapping exploiting the such methodology. In this

sense, almost all synthesis flow can be maintained, with only minor adjustments. Unfortunately, there is no algorithm capable of generating optimal synthesis for functions with more than 3 inputs. This paper proposes a novel technique to generate a library having optimal cells with up to 4-input functions.

II. Related Work

Recently, some majority logic reduction methods targeting QCA, TPL and SET circuits have been proposed. All these technologies use majority or minority gates as primitive elements.

In [9], Zhang *et al.* pointed out a set of 13 functions of 3 variables implemented using only majority gates and inverters. This set is also the set of functions called 3-NPN. A NPN set is a class of functions equivalent to each other, considering the permutation of its inputs, complementation (negation) of its inputs, and/or inversion (negation) of its output. This set aims to reduce the hardware requirements for a QCA design, working as a cell library, but the 13 cells implementing the set of 3-NPN class of functions are not in minimal form, i.e., the minimal number of majority gates.

In [4], Zhang *et al.* proposed a different flow. First of all, the circuit is decomposed in subcircuits having 3 or less inputs, and each subcircuit is optimized using factoring algorithms to synthesize the functions, instead of using a cell library. However, in the most of time the algorithm convert AND/OR gates from a factored expression to majority gates, which can negatively impact the total number of majority gates in a circuit.

Momenzadeh *et al*., in [10], optimized two functions of [9] that were not implemented in minimal number of majority gates, and proposed an And-Or-Inverter (AOI) structure composed by 2 majority gates connected in series to reduce even more the number of majority gates present in a circuit.

In [11], Kong *et al*., in turn, improved the library provided in [9], ensuring and proving optimality for 3 variable functions. The algorithm is divided in three steps. The first is a decomposition of the circuit in Boolean networks having at most three inputs. The second step is the matching, generating best implementations for each Boolean network. The last step is a post-optimization, eliminating Boolean networks that have the same complemented or uncomplemented functions.

However, those methods only support three variable Boolean functions. In order to synthesize arbitrary multi-variable Boolean functions, a QCA majority synthesis methodology was introduced in [9]. In majority logic synthesis, there are still some important aspects that have not been solved or considered by the existing methods [4-6][9-12]. For instance, those methods are not capable to generate optimal structures with majority gates with more than 3 variables. The 4-NPN function class has 222 functions, and the 4-P function class has 3984 functions, so making the generation of these libraries unfeasible without computational aiding. For this reason, it is important to have available an automated method which can synthesize more than 3 inputs.

Table I compares the previous works and the majority-based circuit synthesis algorithm proposed in this paper. The column "Majority generation" indicates how the algorithm performs the circuit, or by real time synthesis or using a library to perform technology mapping. It is interesting to use a library, since the generation of the cells is a pre-computation effort, saving CPU time in the synthesis flow. The column "Allowed templates" shows the allowed primitive elements (beside inverters) in the method.

TABLE I: COMPARISON BETWEEN MAJORITY-BASED SYNTHESIS ALGORITHMS.

Method	Exactness	Max inputs	Allowed templates	Majority generation
[9]	heuristic	3	MAJ	library
[10]	exact	3	MAJ,AOI	library
[4]	heuristic	3	MAJ,MIN	synthesis
[11]	exact	3	MAJ,MIN	synthesis
This paper	exact	4	*any*	library

III. Majority-Based Library Generation

Functional Composition (FC) [8] is a novel synthesis paradigm that performs bottom-up association of Boolean functions as opposed to the top-down functional decomposition approach. By performing such a bottom-up approach, the costs of initial functions are necessarily known, the logic operations are simple, the subfunctions have sub-optimal and optimal implementations, and a control cost can be easily set.

The FC paradigm is based on some general principles. These principles include the use of bonded-pair representation, the use of initial functions set, the association between simple functions to create more complex functions, the control of costs achieved by using a partial order that enables dynamic programming, and the restriction of allowed functions to reduce execution time/memory consumption. Each principle is explained detailed in [17].

The flexibility of FC allows taking advantage of bonded-pairs complex association to generate structures (implementations) with only majority gates and inverters.

In this paper, a majority function will represent a 3-input majority gate function. The optimal factored form of a majority function (MF) is expressed as:

$$maj(a,b,c) = a \cdot (b + c) + b \cdot c \qquad (1)$$

A method to compose a MF using Boolean functions needs four logic operations, the same number of operators in the Equation 1. Since all variables of MF are positive unate, there is the necessity of inverters to represent negative unate and binate variables in the functions. The MF is symmetric, thus changing the order of inputs does not change the logic function. If a MF has one of its variables assigned the constant ZERO (ONE), it represents the logic function AND (OR). In this sense, all functions can be represented with majority gates and inverters.

Another interesting property of a MF is to be a self-dual function. This property allows easy conversion from majority-based circuits to minority-based circuits. If a majority gate has

the output negated, the majority gate acts as a minority gate. If the minority gate has the inputs complemented, the minority gate act as majority gate and vice-versa.

Considering all MF properties, the bonded-pair association needs to be performed using three bonded-pairs. The circuit structure (implementation) stores the majority gates and its connections. This information is important, since it allows a traverse backward in the structure, having control of all criterions of the circuit that implements the function.

A. *Logic Depth Approach*

According to [11], the most important criterions to optimize a majority gate circuit (in QCA) are (from most important to least important): logic depth, majority gate count, gate inputs and inverter count. Logic depth is the maximum number of gates (in this algorithm, majority gates) a signal needs to travel from the input to output. The logic depth is related to the delay of a logic gate. The gate inputs is the number of majority gate inputs that are not connected to constant '0' or constant '1'. Reducing gate input count makes the routing task easier and should reduce the final area. These criterions are adopted in this work. The partial order used is the logic depth, considering only majority gates to compute the logic depth. If there is a tie (two implementations representing the same function with the same logic depth), the other criterions are then used as tiebreakers.

B. *Synthesizing a library*

A library is a finite set of primitive logic gates, including combinational, sequential (e.g. flip-flops) and interface (e.g. drivers) elements. The interest in this work is only the combinational part, where each element implements a Boolean function. There are some papers that discuss sequential elements as [14] and [15], but those elements are not in the scope of this work.

It is interesting to have the maximum number of functions, allowing flexibility in the technology mapping. The majority gate based library is composed of cells implemented with majority gates and inverters as primitive structure to represent the Boolean functions.

In [11], Kong used an algorithm based on an initial set of 40 functions/implementations that need at most 1 majority gate to implement. If the targeted function is not in this set, the algorithm analysis all minterms to check if three functions selected from the set can generate the target function. In the worst case a 3-combination of all functions will be needed to generate the resulting function. This can be computationally costly and does not scale for 4 variables. The approach in this work is the opposite. All function up to 'n' variables are generated in one execution and can be stored in a look-up table to be used afterwards, reducing greatly the computational effort and allowing the algorithm scale up to 4 variables. Another difference is the use of Boolean functions represented using a CPU word (an integer number, for example), being faster than minterms traversal and comparison. Further optimizations in the algorithm include reducing the number of combinations using some logic depth properties.

C. *Library Results*

The proposed algorithm generated all 4-input functions. That is the first algorithm to synthesize functions with 4-inputs minimally in logic depth. The results are not guaranteed minimal for majority gate count. The "number of majority gates" partial order approach will be implemented in a future work to compare an optimal logic depth library and an optimal majority gate count library.

The distribution of the majority gates is shown in Figure 22. The histogram is shown in log scale for better visualization. It is worth to mention that only the XOR4 and XNOR4 are the only ones having logic depth equal 4. All functions can be synthesized with 11 or less majority gates. The distribution of logic depth in the 4-P and 4-NPN is shown in Figure 3. Only XOR4 and XNOR4 need 4-depth, all others 3-depth or less.

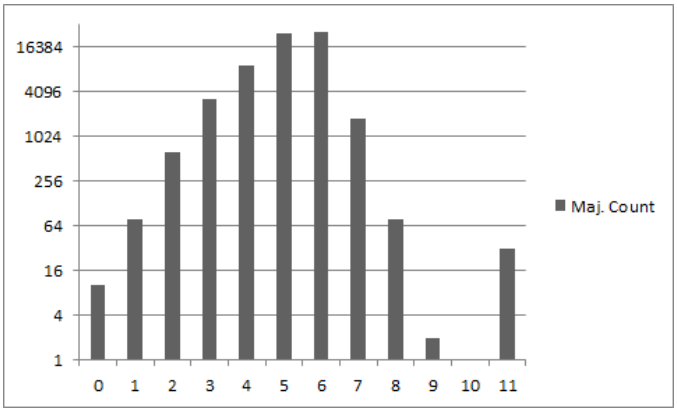

Figure 2: Histogram for 4-input library, considering the number of majority gates to implement the functions.

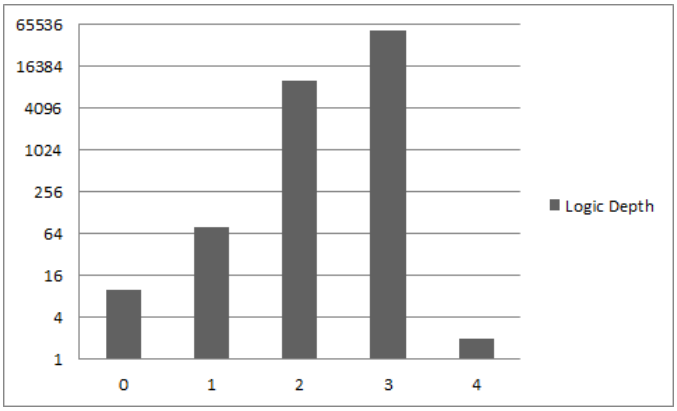

Figure 3: Histogram for 4-input library, considering the logic depth of the functions.

D. *MCNC Benchmark Results*

The experimental results and a comparison to the other methods are presented herein. The platform was an Intel Core *i5* processor with 2GB main memory. The benchmarks used are MCNC [16].

The first step is to decompose and remove redundancies. The decomposition and redundancy elimination scripts are based in scripts presented in [11], with the main difference of being decomposed in 4-feasible functions. After the decomposition and elimination, there is a matching step,

which consists in search each 4-feasible function in a lookup table. This lookup table is generated by the proposed algorithm, containing all functions up to 4 inputs, and is used to update the Boolean network with its implementation.

The experiments were conducted in 40 MCNC benchmarks [16] in order to compare with [11] results. Unfortunately, it was not possible to recreate the results from [11]. A reason is that there is not an official repository for the benchmark. The authors found at least three different versions of this benchmark, all these versions providing different results. Another possible reason is the version of SIS tool used in this work, being different from the SIS used to create the results in [11], probably affecting the decomposition algorithms and impacting in the final circuit.

In order to have a fair comparison, the authors implemented an in-house version of [11] and select one of the possible sets for the MCNC benchmark suite. All decomposition methods implemented by [11] were used and the best decomposition for each benchmark was selected and compared with the best result of [11]. According to this comparison, it is clear that the benchmark description impacts greatly the final results, e.g. the 'k2' benchmark, having a difference of 31,6%, -30,9% -30,9% and -34,9% in levels (logic depth), majority count, gate inputs and inverters, respectively. The 'vda' circuit have similar results of 'k2' one. This turns impossible a fair comparison of results, since the authors circuit description differs greatly of [11].

Comparing two different libraries, 70% of the benchmarks had a logic depth improvement, with gains up to 41.7% as seen in i1 benchmark. Only one benchmark, the '9symml' circuit had a slight worsening of 7.7% in logic depth. As the algorithm proposed optimizes first logic depth, there is a penalty in the other criterions. As expected, 50% of the benchmarks have an increase in majority gate count, but there are only 17.5% of these benchmarks with a majority gate count increase greater than 10%. Optimizing a circuit without reducing logic depth is possible, as seen in 'cht' benchmark, reducing 16.6%, 16.2% and 79.6% in majority gate count, gate inputs and inverters respectively. Reducing logic depth and majority gate count at same time is not uncommon, occurring in 30% of the benchmarks.

IV. Conclusions

In this paper, an algorithm was introduced for synthesizing circuits using only majority gates and inverters, suitable for use in new technologies, as QCA, SET and TPL. This algorithm generates the optimal structure of majority gates, given a function and can generate a library in an automated way, using the functional composition paradigm. All techniques in the literature can only handle 3-input functions and this algorithm can handle 4-inputs. The results in MCNC show that there is a significant reduction up to 41.7%, 19.6%, 16.2% and 79.6%, in logic depth, majority gate count, gate inputs and inverters, respectively.

Acknowledgment

Research funded by the Brazilian funding agencies CAPES, CNPq and FAPERGS, under grant 11/2053-9 (Pronem), and by the European Community's Seventh Framework Programme under grant 248538-Synaptic.

References.

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

[2] H. A. H. Faluny, and R. A. Kiehl, "Complete logic family using tunneling-phase-logic devices", Microelectronics, 1999. ICM '99. The Eleventh International Conference on , vol., no., pp. 153- 156, 22-24 Nov. 1999

[3] D. V. Averin and K. K. Likharev,. "Coulomb blockade of single-electron tunneling, and coherent oscillations in small tunnel junctions", Journal of Low Temperature Physics, vol. 62, pp. 345-373, Fev. 1986.

[4] R. Zhang, P. Gupta and N. K. Jha, "Synthesis of Majority and Minority Networks and Its Applications to QCA, TPL and SET Based Nanotechnologies", Proceedings of the 18th International Conference on VLSI Design 2005 pp. 229- 234.

[5] H. S. Miller and R. O. Winder. "Majority logic synthesis by geometric methods", IRE Trans. Electron. Comput., vol. EC-11, no. 1, pp. 89–90, Feb. 1962.

[6] S. Muroga, "Threshold logic and its applications". Wiley Interscience, New York, 1971.

[7] R. K. Brayton,, A. L. Sangiovanni-Vincentelli, C. T. McMullen and G. D. Hachtel. "Logic Minimization Algorithms for VLSI Synthesis". Kluwer Academic Publishers, Norwell, MA, USA. 1984.

[8] M. G. A. Martins, V. Callegaro, L. Machado, R. P. Ribas and A. I. Reis, "Functional Composition Paradigm and Applications". International Workshop on Logic and Synthesis (IWLS'2012). Berkeley, CA. 2012

[9] R. Zhang, K. Walus, K., W. Wang and G. A. Jullien, "A method of majority logic reduction for quantum cellular automata", IEEE Transactions on Nanotechnology, vol.3, no.4, pp. 443- 450, 2004.

[10] M. Momenzadeh, J. Huang and M. B. Tahoori and F. Lombardi, "Characterization, test, and logic synthesis of and-or-inverter (AOI) gate design for QCA implementation", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.24, no.12, pp. 1881- 1893, 2005.

[11] K. Kong, Y. Shang and L. Ruqian, "An Optimized majority Logic Synthesis Methodology for Quantum-Dot Cellular Automata, Nanotechnology", IEEE Transactions on , vol.9, no.2, pp.170-183, March 2010

[12] S. B. Akers, "Synthesis of combinational logic using three-input majority gates", in Proc. 3rd Annu. Symp. Switching Circuit Theory and Logical Des., Oct. 1962, pp. 149–157.

[13] S. Chatterjee, A. Mishchenko, R. Brayton, X. Wang, T. Kam, "Reducing structural bias in technology mapping", ICCAD-2005, pp. 519- 526.

[14] M. Torabi, "A new architecture for T flip flop using quantum-dot cellular automata", (ASQED), vol., no., pp.296-300, 19-20 July 2011

[15] S. Hashemi, K. Navi, "New robust QCA D flip flop and memory structures", Microelectronics Journal, Volume 43, Issue 12, December 2012, Pages 929-940

[16] R. Lisanke; "Logic synthesis and optimization benchmarks", Microelectron. Center North Carolina, Research Triangle Park, NC, Tech. Rep., 1988

[17] M.G.A. Martins, R. P. Ribas, A. I. Reis,, "Functional composition: A new paradigm for performing logic synthesis," Quality Electronic Design (ISQED), 2012 13th International Symposium on , vol., no., pp.236,242, 19-21 March 2012