ToPoliNano: Nanoarchitectures Design Made Real

S. Frache, D. Chiabrando, M. Graziano, F. Riente, G. Turvani, and M. Zamboni
Electronics and Telecommunications Department, Politecnico di Torino, c.so Duca degli Abruzzi 24, Torino, Italy
Email: {stefano.frache, mariagrazia.graziano, maurizio.zamboni}@polito.it

***Abstract*—Many *facts* about emerging nanotechnologies are yet to be assessed. There are still major concerns, for instance, about maximum achievable device density, or about which architecture is best fit for a specific application. Growing complexity requires taking into account many aspects of technology, application and architecture *at the same time*. Researchers face problems that are not new per se, but are now subject to very different constraints, that need to be captured by design tools. Among the emerging nanotechnologies, two-dimensional nanowire based arrays represent promising nanostructures, especially for massively parallel computing architectures. Few attempts have been done, aimed at giving the possibility to explore architectural solutions, deriving information from extensive and reliable nanoarray characterization. Moreover, in the nanotechnology arena there is still not a clear winner, so it is important to be able to target different technologies, not to miss the next big thing.**

We present a tool, ToPoliNano, that enables such a multi-technological characterization in terms of logic behavior, power and timing performance, area and layout constraints, on the basis of specific technological and topological descriptions. This tool can aid the design process, beside providing a comprehensive simulation framework for DC and timing simulations, and detailed power analysis. Design and simulation results will be shown for nanoarray-based circuits.
ToPoliNano is the first real design tool that tackles the top down design of a circuit based on emerging technologies.

I. Introduction

The end of a remarkably successful era in computing is approaching. Its the era where Moore's Law reigns and processing power per dollar doubles every year.

There have been many attempts to keep pace according to Moore's law: massive parallelism is one of these. In fact, parallel computation has been around for a while, and it has been a driving topic since the development of integrated architectures. It is now even more a reality with multiprocessors systems, thanks to the integration capabilities reached by scaled technologies. However, parallelism levels now feasible, even though more relevant than ever, allow to achieve only a tiny portion of what could really be faced in certain breakthrough applications (biological related processing in medicine could be one of the examples [1]). Thus, even though research and technology is expected to greatly improve in this field during the following years, the predicted limits of CMOS technology [2] will prevent substantial revolutions in the amount of information that can be processed in parallel.

A new era, though, is on the horizon [3]: the nanoelectronics era, with ever smaller devices and higher densities, to keep up with pace.

Manifold nano-structures have been proposed in recent years [4], and probably few of them will survive feasibility and selection [5]. The explored solutions for what concerns massive parallelism are based on nanowire arrays [6], organized in matrices [7], which allow the creation of active nanodevices (diodes and FETs) in their crosspoints [8]. In general these structures are conceptually organized in two-dimensional tiled arrays. In particular, nanoscale programmable logic arrays, e.g. nanoPLA, have been proposed in [9], while [10], suggests molecular/nanowire array based solutions, e.g. CMOL. NASICs designs have been proposed in [11], [12], [13], as a way to achieve denser designs with better fabric utilization and efficient cascading of circuits with respect to general-purpose programmable fabrics (PLAs). Authors in [14] and [15] show how such structures are suitable for developing massively parallel architectures like cellular neural networks or image processors. Nevertheless, despite their promising characteristics, these structures have to cope with not negligible defect rates, primarily due to the critical manufacturing processes at nanoscale level. Defect tolerant techniques have been widely proposed in connection with nanoscale arrays [16], [17], [18], [19], thus clarifying that faults analysis is mandatory when dealing with nanoarray-based structures.

Nanoscale array structures, albeit still in their infancy both from technological and design points of view, show promising perspectives in the direction of massively parallel computing structures [20].

Epoch-making changing brings new challenges into play. New devices require newly developed approaches to fabrication and integration, new methods to simulate them, and new architectures to unleash their power. There is, then, a need for design tools that capture the specificities of these technologies, to explore the space of possible solutions, and to validate proposed circuits and architectures. In other words, researchers need to rethink the methodologies and design tools involved.

Here we present such a tool, ToPoliNano (Torino Politecnico Nanotech design tool), along with preliminary results. It has the ability, starting from a VHDL circuit description, to aim at different disruptive nanotechnologies (Nanoarray-based circuits, as in Figure 1, targeted here as a possible example, but not limited to), to place and route circuits on a low-level floorplan (details in a later section) and then to simulate it, in an integrated fashion.

The aim of this tool is the study of complex systems based on emerging electronic nanotechnologies. More specifically, we are interested in studying dimensions, performance, power consumption and, as a consequence, in developing optimized architectures. A key feature of ToPoliNano is the ability

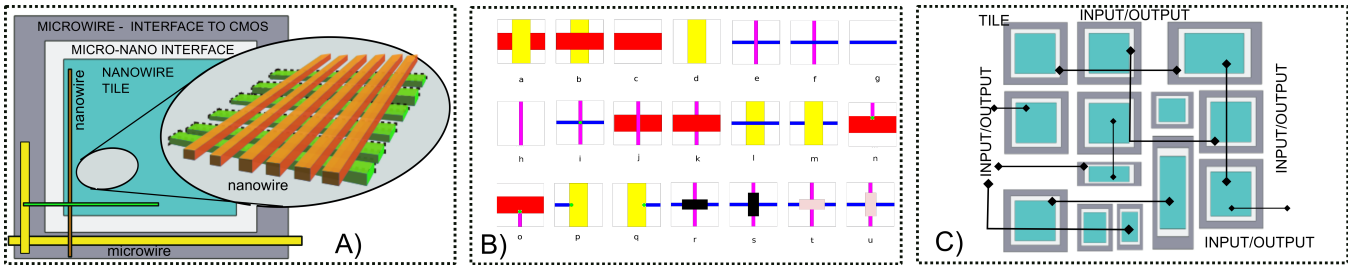

Fig. 1. NASIC tile A) General structure of the fabric B) Different sub-tiles composing the design C) Exemplification of design constraints.

to base its high level analysis of complex structures onto low-level information, derived from actual device technology, circuit topology and post-placement circuit layout parasitics extraction, in an efficient way. The expected overall result is thus an accurate characterization of the design, based on detailed technological parameters and device-level simulation results.

The paper is organized as follows: section II introduces previous works in the field and our case study structure; section III describes the general organization and features of the ToPoliNano software; section IV introduces its low-level floorplanning capabilities and related results; section V is about logic simulation and results; section VI deals with timing simulation.

II. Previous works and case study structure

Circuits based on nanoarrays have been the subject of several previous studies [5], [9], [10], [13]. They have been investigated, among other reasons, for the high theoretical density of devices that can be obtained, for the high frequency of operation, for the low power consumption [1], and for the possibility of being integrated with CMOS structures.

What seems problematic, however, is to study in details circuits and architectures, taking *simultaneously* into account aspects like device density, device defectiveness, power consumption, frequency of operation, etc, since it is clear the interdependence of the results obtained.

It is important to assess which parameters have the greatest impact on the feasibility of an architecture and its performance, to choose the actions to be undertaken in a given technology and to determine which are the key aspects to be further investigated. Authors in [21] give an important overview of a possible architecture based on the regular composition of variably sized NASIC tiles, and this kind of exploration is of aid towards a real implementation.

An high level approach to the evaluation of global performance of these technologies can not determine which is the impact of the variation in a transistor's technological parameter on the performance of a realistic circuit. Moreover, it is not possible to neglect the impact of the actual layout of the circuits, after all the constraints of placement and routing have been taken into account, because it could lead to an overestimation of the expected performance of these technologies.

Besides all the attractive aspects of these technologies (device density, low power consumption, high frequency of operation), there are other reasons of study, related to the importance of handling the high defectiveness of these structures, which also has been investigated [17], [19]. Different techniques of fault tolerance have been applied to these fabrics; of the so-called built-in type, either through the reconfiguration of the fabrics. The defectiveness of these nanofabrics is a delicate point to be treated, because it is much higher than in the CMOS case. High defect rates are likely to considerably reduce the advantage in terms of maximum device density that can be reached after the implementation of effective fault tolerance techniques.

One evident problem arising from literature analysis is the lack of design and simulation tools specifically dedicated to nanoarray architectures. One system level approach has been developed in [22], where a framework based on a variety of models allows the architect to map an application on a wide range of emerging nanofabrics. Based on models of a particular fabric, e.g. computational, architectural, technological and fault, this framework suits the need of the designer to compare different nanoarray approaches. Anyway, it is based on high level models of a whole tiled nanoarray, while the specific nanoarray organization, its logic topology, the nanowire and nanodevices technological description are not directly included, thus allowing a system level perspective and not a detailed array behavior characterization. A device-level approach has been proposed in [23], where a crossed nanowire field-effect transistor is 3-D modeled and device level characteristics are extracted to validate at SPICE level the dynamic circuit style adopted in the NASIC approach. Though this is a fundamental step for the validation phase, it cannot be inherited at higher description levels, for the inefficiency in both describing and simulating a complex tiled structure or even more a nanoarray architecture.

Though we aim, like in [18], to maintain the simulator general, so that it can be adapted to the evolving fabric styles proposed in literature, we underline that the key feature of our simulator is the ability to take into account technological parameters, circuit style, topology, layout and, at the same time, to efficiently analyze the behavior of complex architectures, as will be shown by means of the obtained results. As far as the defect analysis is concerned, we have addressed the problem in a previous work, with a preliminary version of this tool

[24]. The capability to perform this type of analysis has been extended to work on arbitrarily complex circuits, and is now integrated into ToPoliNano, but it is not dealt with in this paper. Similarly, we investigated and implemented the power consumption analysis capability in [1]: for the sake of brevity, it is not discussed in this paper.

To introduce the ToPoliNano tool we choose a crossed-nanowire based architecture, of the kind depicted in Figure 1A. ToPoliNano, in fact, is able to handle manifold nanoarray-based architectures. The manner in which the tool is capable of treating variants of the same technology, and ultimately different nanotechnologies, is detailed in the next section. In particular, we choose the NASIC architecture.

According to its proponents [11], the elemental units in NASIC are called tiles. These are circuits for adders, multiplexers, and flip-flops. Individual tiles can then be connected with nanowires or microwires to form a larger, multi-tile structure. As we pointed out before, all nanoscale computing systems have to deal with the high defect rates of nanodevices and faults introduced by manufacturing of fabrics, and so do NASICs. Their nanoscale underpinning is based on a grid of NWs (or CNTs). The grid crossings can be programmed either as FETs, P-N type diodes, or can be disconnected, thus implementing a two-level logic architecture. NASIC designs do not have logic planes of fixed size and wiring/routing between them, as in PLA-type designs. Furthermore, NASICs have been proposed in both static-ratioed and dynamic styles [11], with the latter that enables pipelining and overcomes the many limitations of a static design.

Finally, micro-wires are used to carry power and control signals from the CMOS level. Faults are handled by masking them in the circuit and/or architecture design itself, implementing a multi-tiered built-in fault tolerance approach [17]. Dataflow in NASICs is through a multi-phase progression and the control signals from the CMOS level coordinate these phases.

III. ToPoliNano general organization

How to cope with a multi-technological scenario. ToPoliNano is a design and simulation tool that supports a variety of nanotechnology, from NML (Nano Magnetic Quantum Dot Cellular Automata) [25] to nanoarrays based on SiNWs or CNTs. This choice is dictated by the importance of evaluating several promising technologies, among which it is not yet clear what will be successful. It is necessary to evaluate them in different fields of application and with specific architectures for each of them.

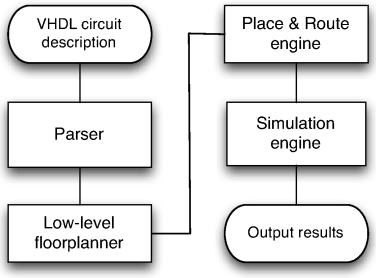

Fig. 2. Basic ToPoliNano flow.

To support different technologies one can follow several approaches. The most obvious, but also less economical in terms of lines of code and development time, is to write specific code for each technology. A less obvious approach, but that saves you both lines of code and time, is to exploit the similarities existing even between different nanotechnology to unify large portions of code, which turn out to be shared. One might think that this way will necessarily lead to inefficiencies, perhaps in terms of memory usage, or long simulation times. The benchmarks that we performed to assess the timing of design and simulation phases say quite the opposite, as we show in the results (see section IV-B). Each nanotechnology fabric, in fact, has unique characteristics but, at the same time, shares technological constraints, which are a main cause for unification of certain fundamental aspects. One of these aspects is the requirement of a two-dimensional array as the underlying structure for computing [20], [7], [9]. This is evident from the crossbar structures, but is perfectly applicable also to the case of NML, for example.

The "building brick" principle. Described in literature are numerous variations on the theme of architectures based on crossed nanowires [13],[10]. To be able to treat them all within the tool, the tool itself has been designed to exploit some common elements in these structures. First of all is the regularity of the 2D array. One can imagine the plane of the circuit covered with tiles of different sizes, each containing a maximum of three components, in precise positions.

With reference to Figure 1B, one can see the variety of basic tiles (from *a* to *u*) that can be created with a reduced set of elementary components. To design a NASIC circuit, in fact, requires only 5 such elemental components, rotations excluded: p-type and n-type transistor, contact, nanowire, microwire. By composing these pieces one can get all the richness of expression needed to describe an arbitrarily complex circuit.

The recursive hierarchical composition principle. Another key aspect in the generalization of the different architectures is the application of a principle of recursive hierarchical composition. Each element, thus, may be part of the architecture at any level of the hierarchy. This allows for maximum extendibility of the tool, because you can always improve on the design of circuits. Provided you can describe it in terms of elemental components, whatever your library of elemental components is, you can aggregate them and reuse the aggregates in turn as components. A brief example of the application of this principle to NML will be given in section IV.

The fundamental steps. ToPoliNano integrates all the tools involved in the design and simulation of circuits based on so-called disruptive nanotechnologies in just one cross-platform tool. It actually runs on three platforms: Linux, Mac OS, Windows. It has been entirely developed in C++, and currently counts around 87k lines of code, external libraries excluded.

The main application organization and flow is briefly discussed: see Figure 2 for a very simplified one.

After application's first launch, the user is presented a wizard, to choose the target technology among the supported ones (currently NML and Nanofabrics) and the technological node of interest, beside performing early configuration of the related simulation parameters. At this stage, or at a later one, the user can describe the circuit by means of VHDL files.

Parser. The tool features a HDL parser, presently implementing essential parts of the VHDL specification. A design can be described at different abstraction levels, i.e. in terms of elemental components (patterns of sub-tiles, etc.), as well as in terms of complex tiles (i.e. plain NASIC tiles) based on the available elemental components in a dedicated Component Library or with previously defined tiles. This works in a hierarchical fashion, through the Component Library which is user-expandable. The user inputs a VHDL description of the circuit, by recalling the components through the familiar component statement of the VHDL language. The parser will analyze the code and create an internal representation of the circuit, used to compose it with items from the library itself or the output from a synthesizer, which will then feed the library.

Place and Route engine. At this stage, an intermediate form representation is used to place the circuit, once a new low-level floorplan is defined or a previously defined one is chosen. Fig.1c shows the constraints (nanowire and microwire pitch and size, non routable areas in gray, microwire dedicated areas, position of inputs and outputs, etc.) that have to be taken into account at the lowest floorplanning level. This is one of the design steps which differs a lot with respect to conventional CMOS technology. In fact, the standard cell approach has totally different constraints to enforce. In the context of nanotechnologies, we have to handle very different constraints across nanotechnologies, even if they can all be thought-of in terms of elemental components. The automatic constrained routing phase can then take place.

Simulation engine. The tool supports a variety of nanotechnologies, and so does the simulation engine, which must therefore allow cross-technology operation. Parts of the tasks related to simulation (i.e. input/output vectors handling) can be shared by all technologies, but there are specific aspects of the low level simulations that must be customized, because the underlying physics differ. To maximize the portion of code that can be shared across technologies an event driven approach was adopted, and will be further discussed in section V.

IV. Low level floorplan: discussion and results

A. Dynamic floorplanning and placement

Each nanotechnology brings with itself the constraints relating to the physical characteristics of the devices it exploits. If Silicon nanowires and Carbon nanotubes have similar physical structure, though distinct technological parameters, things are different for technologies such as NML, based on nanomagnets. The constraints to which the devices are subject are fundamentally different [26].

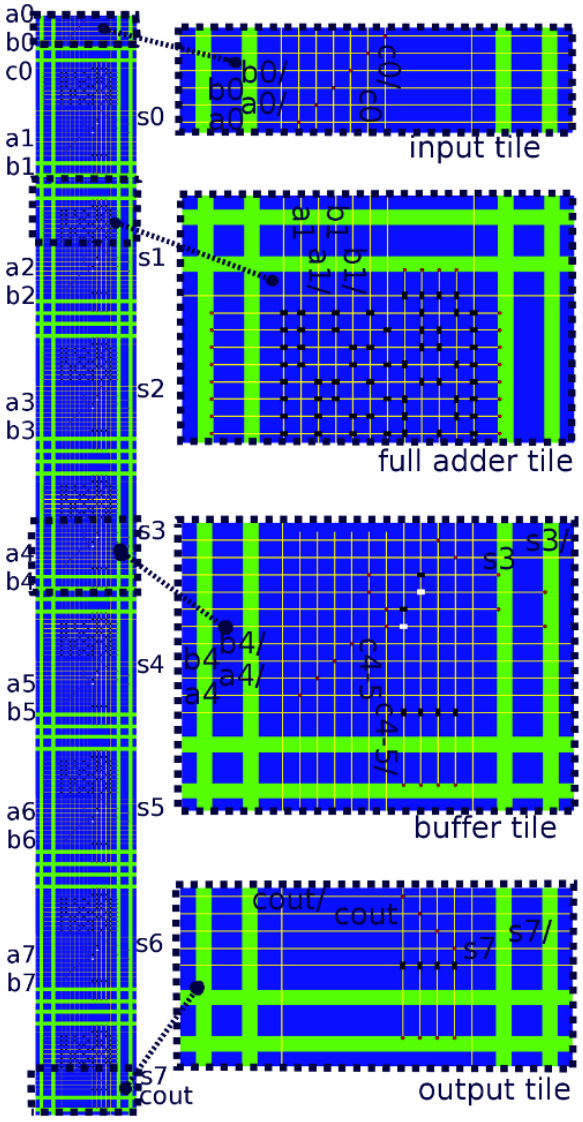

Fig. 3. 8 bit Adder layout, with zoom of the input tile, 1bit FA tile, buffer tile, and output tile.

In the case of nanowires/nanotubes, the constraints are those of a crossbar structure. With reference to Figure 1C, among the constraints to take into account there is the wire pitch (affecting the size of the area in aquamarine), the area devoted to the microwires (in light gray), the routability of interconnections among tiles (black lines with ending dots), the position of inputs and outputs (both primary and secondary), the alignment of blocks of different size, etc.

Different is the case of nanomagnets. These require a number of current flows below the plane of the magnets, in order to influence their magnetization vector. The structure through which the current is distributed under the plane of the magnets generates areas where you can not place magnets, and thus constraints. In Figure 4E a representation of the main constraints in NML technology.

The constraints that the different technologies arise, to position the devices on the plane of the circuit, require an

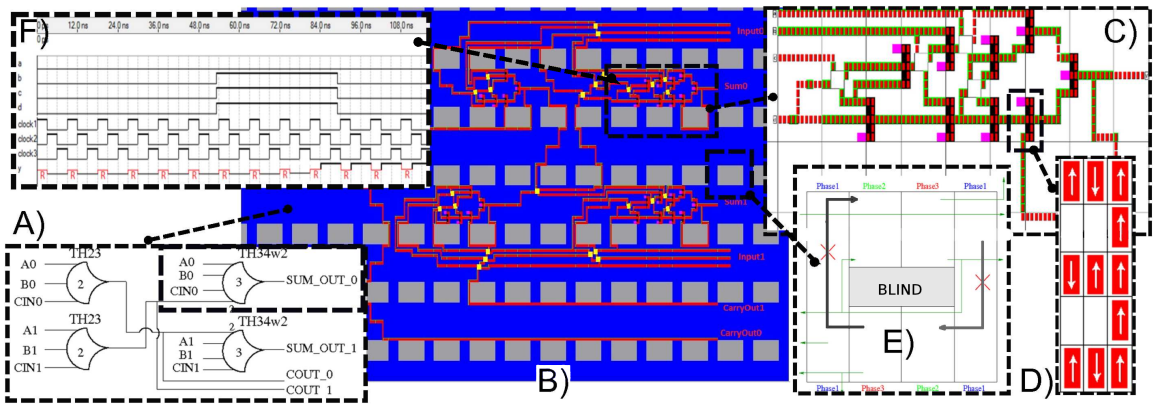

Fig. 4. NML FA (Full Adder). A) Null Convention Logic (NCL) based FA circuit B) FA layout: snake-clock-compliant floorplan and routing C) Layout of a Th34 NCL port D) Majority Voter: one simulation step E) Detail of a place & route constraint F) Th34 simulation.

appropriate partitioning of the available space. We call *low-level floorplan* the particular partitioning of the space at device level that depends on the specific technological constraints. We do not refer to partitioning at the level of functional units, to which the term floorplan usually refers to.

To capture the constraints of different technologies, while maintaining a common data structure, we introduced an abstraction at the level of subdivision of the plane of the circuit. A graph represents the plane of the circuit, its partitioning and all its parts. The constraints are represented by specific classes, which became part of the class hierarchy as one or more concrete classes: they implement the abstract base class to represent constraints in the specific technology. In this way, to introduce a new technology does not require extensive rewriting of the code, but only the introduction of specific classes for the new constraints and/or the extension of existing classes, wherever a greater reuse of the code is possible.

Dynamic low-level floorplanning is possible by performing operations on the common graph structure. This allows for resizing, displacement of circuit parts, and also for easy placement of the circuit elements. Because all the classes that represent circuit parts must conform to a common interface, due to the adherence to a hierarchical recursive composition principle, it is lighting-fast to move entire portions of a circuit from one point of the low-level floorplan to another.

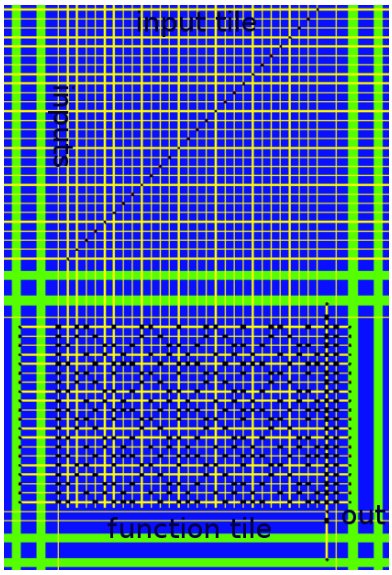

Fig. 5. A 28-input function layout after synthesis. $f = a_2a_3a_6a_9b_1b_4b_7b_8c_2c_5c_6c_9 + a_2a_4a_9b_2b_7b_9c_5c_7 + a_1a_5a_6a_8b_3b_4b_6c_1c_2c_4c_8c_9 + a_3a_5a_7b_1b_3b_5b_8c_1c_3c_6c_7d_1 + a_1a_4a_6a_7a_8b_2b_4b_5b_6b_9c_2c_3c_4c_7c_9d_1$

B. Results

Figure 3 shows the layout of an 8-bit adder, based on eight 1bit Full Adders, after VHDL parsing, placement and routing on a low-level floorplan. In particular, details are shown of the complementary structures: the input and output tiles, as well as a buffer tile between one stage and the next. This represents an evolution of traditional NASIC designs, that comprises a modification of the structure that manages the control signals of the vertical nanowires, now included in the buffer tile. It is possible for the tool to automatically generate this kind of tile that, among its tasks, also features the routing of the input signals, as shown in the same figure.

Figure 5 shows the layout of a 28-input function after synthesis and placement on a low-level floorplan. Also depicted in figure, the routing of the input signals from the left, which perfectly shows the cost of routing the signals and the need to carefully plan the orientation of the tiles in a design.

In order to hint to the multi-technological capabilities of ToPoliNano, Figure 4 summarizes results for an example targeted to NML. The Full Adder circuit (A) was described in VHDL in terms of Null Convention Logic [27] gates (twoTh23 ports and two Th34). It has been placed and routed (B) respecting NML constraints (E), e.g. properties of the snake clock, magnet phase, size, BLIND zone, etc. A detail for a Th34 gate layout based on nanomagnets is in Figure 4C. The whole circuit has been simulated at low-level considering the correct magnetization switching (D). A transient waveform for the Th34 is in figure inset F.

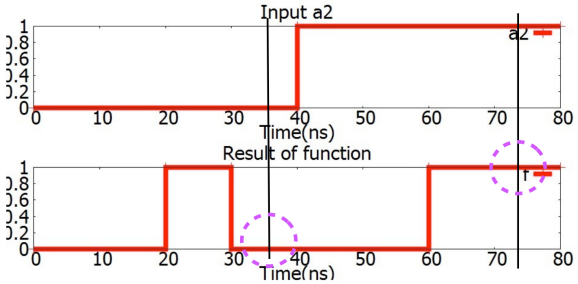

Fig. 6. 28-input function logical simulation result. The function $f = a_2a_3a_6a_9b_1b_4b_7b_8c_2c_5c_6c_9 + a_2a_4a_9b_2b_7b_9c_5c_7 + a_1a_5a_6a_8b_3b_4b_6c_1c_2c_4c_8c_9 + a_3a_5a_7b_1b_3b_5b_8c_1c_3c_6c_7d_1 + a_1a_4a_6a_7a_8b_2b_4b_5b_6b_9c_2c_3c_4c_7c_9d_1$ is evaluated for input $a2$ transitioning from logic value 0 to logic value 1.

To give an idea of the performance of the tool, preliminary benchmarks on a Linux box (Ubuntu 10.10, Core2DUO t8300 processor) have been performed on designs of different sizes. The instantiation of one Th23 gate (2280 nanomagnets area) took 0,231 seconds. A 10^3 times increase in components and low-level floorplan size required an increase in time of 10^2. The maximum memory occupation for an area of $22.8{\cdot}10^6$ nanomagnets, with $2.6{\cdot}10^6$ magnets actually instantiated, just required 819 Mb main memory and 837 Mb Virtual Memory.

V. Logic simulation: discussion and results

A. Logic simulation engine

The simulation engine belongs to the class of the event-driven simulators. It follows the information flow inside the structure under simulation and generates specific events, when necessary, to correctly handle the propagation of information. To better understand this process, lets turn back for a while to the sub-tiles and their regular structure, as portrayed in Figure 1B. We can imagine each sub-tile as a four-port device, with each port identified by a cardinal point.

A change in the information at a given port may need to be propagated inside the sub-tile, if there is an appropriate component to support propagation (e.g. a nanowire). We do not need to know anything about the electrical properties of the component to perform a logical analysis. As a function of the port at which the change in information happens, and the original direction of propagation of this piece of information, we can check whether there is support for further propagation and, if this is the case, to change the information on another port of the sub-tile by means of the supporting element. This, in turn, will trigger an update event over the sub-tile, if any, connected to the first one by means of the output port. By following the very same process, the information is propagated inside the structure, only where it is needed.

There could be an active device inside the sub-tile, and the propagation of information could lead to a change in its status. Should this happen, another kind of event would be enqueued in the event queue, waiting to be processed to take into account a possible change of information in a direction of propagation that is orthogonal with respect to the one that originated the event. This approach is very flexible, indeed, because it allows for different kind of control of dynamic circuits (number of phases) since the phase sequence is not embedded into the simulator but is coded in the input control sequence (like the example in Figure 3) and the same approach can thereby be used in many different scenarios. It is also quite efficient, as pointed out by the following results.

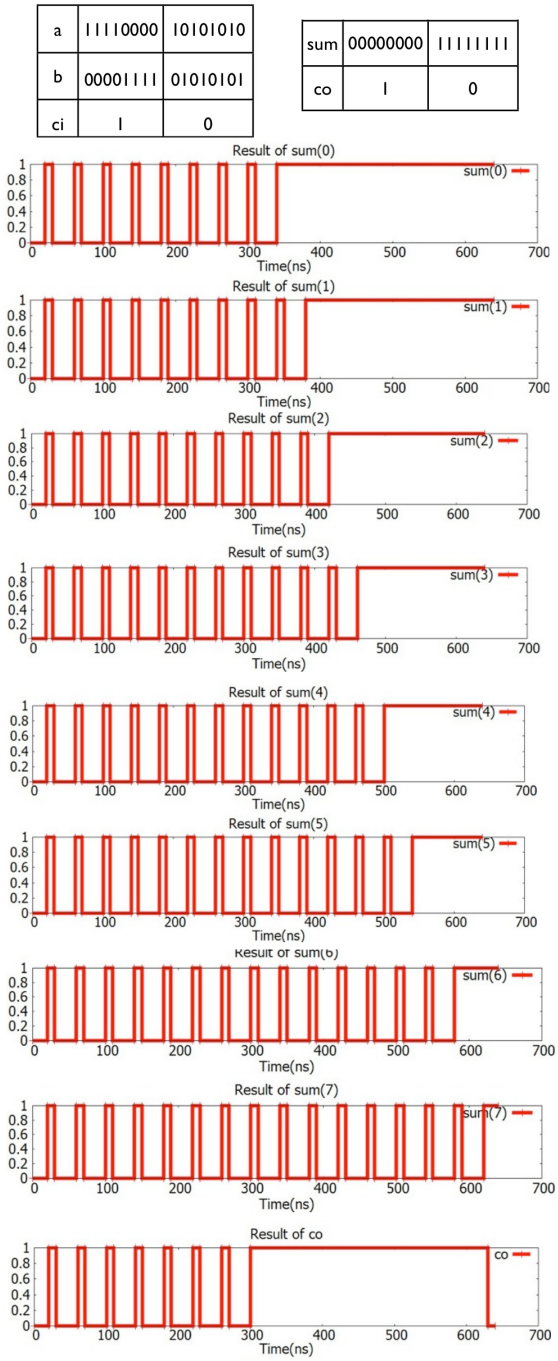

Fig. 7. 8bit Adder logic simulation results.

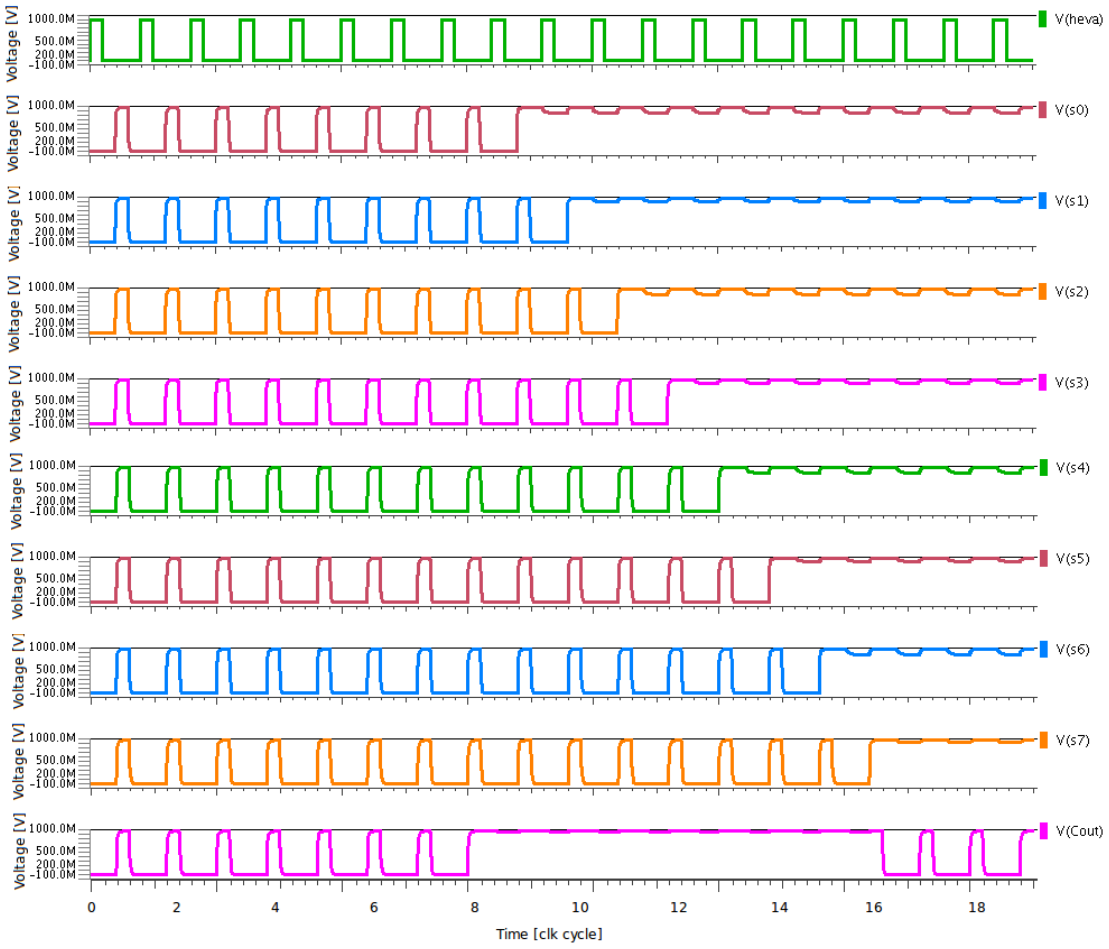

Fig. 8. Waveforms of the 8bit Adder simulations for the same inputs of Figure 7.

B. Results

We performed logic simulations on both the 8bit Adder of Figure 3, and the 28-input function of Figure 5.

Figure 6 shows the correct output result for input $a2$ transitioning from logic value 0 to logic value 1. The time required to complete the logic simulation of this 28-input function, with input $a2$ transitioning from logic level 0 to logic level 1 was 47ms. Figure 7 shows the correct output result for the input values reported in the figure itself. The time required to complete the logic simulation of the FA under the aforementioned input conditions was 140ms for each input vector.

VI. Electrical simulation: discussion and results

A. Electrical modeling and spice-compatible netlist extraction

In order to carry out an accurate timing simulation, care must be taken in evaluating all the parasitics that, inevitably, affect the layout of a circuit, once it has been placed and interconnected. The extraction of these parameters takes place automatically in ToPoliNano after the layout is defined.

To illustrate the principle by which it is being conducted, we refer to Figure 9, where a generic pair of crossed wires is shown. In a real circuit the wires may be either nanowires, or microwires or both kind. Obviously, these wire have a distributed capacitance and a resistance, and also a coupling capacitance, as shown. It is possible to calculate these quantities by geometric considerations, starting from the structure

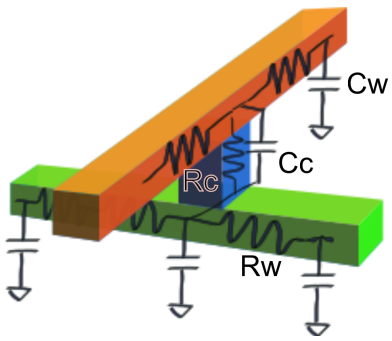

Fig. 9. Example of parasitics in a pair of crossed wires.

of the circuit and the definition of the materials of the parts. We previously showed (see Figure 1B) how a NASIC circuit can be thought-of as composed by sub-tiles. This approach is useful also in the present context. Each of the sub-tile comprises a maximum of three elements, for each of which the tool is able to determine the necessary technological parameters, depending on the technology node set by the user, and automatically calculate the required values. Once these parameters have been calculated, the tool can proceed to the generation of a spice-compatible netlist. In fact, there are small differences in syntax among simulators, e.g. UC Berkeley and Mentor Graphics Eldo Spice. The netlist can be generated in both formats.

Before being able to get accurate timing results, an accurate model of the nanowire FETs is mandatory, because the available transistor models are not well suited to this aim. In fact, they do not scale well down to the few nanometers channel length, typical of these structures. As a proof of concept, to show the complete netlist extraction capability, with all the parasitics, we show in Figure 9 the ELDO simulated waveforms, normalized to a clock period.

We are currently working on a compact model to accurately describe this kind of devices, that will be later used in the simulation engine to provide the required accuracy in the results. Without accuracy in the model, the prospected results would be unreliable.

VII. Conclusion

In this paper we mainly focused on a NASIC structure as a working platform to illustrate our design and simulation methodology and tool.

ToPoliNano is the first example in the literature that enables the design from a top to bottom of a circuit based on emerging nanotechnologies. It allowed to design, place, route and simulate the behavior of a basic arithmetic circuit (an 8bit Adder) and to synthesize, place, route and simulate a random 28-input function in NASIC nanotechnology. The 8bit Adder was also subject to post-layout parasitic parameter extraction, and it has been showed that a netlist can be automatically extracted and fed to spice-compatible simulation software, to get timing analysis of an arbitrarily complex circuit. All the simulations were benchmarked, and the benchmarking results show good performances, both in terms of execution speed and memory footprint.

A 1bit Full Adder was designed, placed routed and simulated in NML (magnetic QCA) technology too, to show the cross-technology capabilities of the tool.

These preliminary results show that the tool is well suited to design and test complex circuits and architectures, which are part of our future work plans.

References

[1] S. Frache, L.G. Amaru, M. Graziano, and M. Zamboni, *Nanofabric power analysis: Biosequence alignment case study*, in "Nanoscale Architectures (NANOARCH), IEEE/ACM International Symposium on", pp. 9198, 2011.

[2] International Technology Roadmap of Semiconductor, Ed. 2009.

[3] International Technology Roadmap of Semiconductors, *Update, Emerging Research Device*, http://public.itrs.net, 2010.

[4] European Commission IST programme Future and Emerging Technologies *Technology Roadmap for Nanoelectronics.*

[5] J. A. Hutchby et al., *Emerging Nanoscale Memory and Logic Devices: A Critical Assessment*, in "IEEE Computer", vol. 41, Issue 5, 2008.

[6] W. Lu et al., *Semiconductor nanowires*, in "J. Phys. D: Applied Physics", n. 39, pp. 387–406, Oct. 2006.

[7] Y. Luo et al., *Two-Dimensional Molecular Electronics Circuits*, in "ChemPhysChem", vol. 3, no. 6, pp. 519-525.

[8] Y. Huang et al., *Logic Gates and Computation from Assembled Nanowire Building Blocks*, in "Science", vol. 294, pp. 1313–1317, 9 Nov. 2001.

[9] A. DeHon, *Nanowire-Based Programmable Architectures*, in "ACM Journal on Emerging Technologies in Computing Systems (JETC)", vol. 1, Issue 2, pp. 109–162, July 2005.

[10] K. K. Likharev, A. Mayr, I. Muckra, Ö. Türel, *CrossNets: High-performance neuromorphic architectures for CMOL circuits*, in "Ann. New York Acad. Sci.", vol. 1006, pp. 146–156, 2003.

[11] C. A. Moritz et al., *Latching on the wire and pipelining in nanoscale designs*, in "3rd Non-Silicon Comput. Workshop (NSC-3)", Munich, Germany, 2004.

[12] P. Narayanan et al., *Manufacturing Pathway and Associated Challenges for Nanoscale Computational Systems*, in "9th IEEE Nanotechnology conference (NANO 2009)", July 2009.

[13] P. Narayanan, J. Kina, P. Panchapakeshan, P. Vijayakumar, S. Kyeong-Sik, M. Rahman, M. Leuchtenburg, I. Koren, C. Chi On, C.A. Moritz, *Nanoscale Application Specific Integrated Circuits*, in "Nanoscale Architectures (NANOARCH)", 2011 IEEE/ACM International Symposium on, pp. 99–106, 8-9 June 2011.

[14] P. Narayanan et al., *Image Processing Architecture for Semiconductor Nanowire Fabrics*, in IEEE Nanotechnology conference (NANO 2008)".

[15] P. Narayanan et al., *Comparison of Analog and Digital Nano-Systems: Issues for the Nano-Architect*, in "IEEE International Nanoelectronics Conference (INEC)", 2008.

[16] J. Dai et al., *Defect tolerance for molecular electronics-based nanofabrics using built-in self-test procedure*, in "IEEE International Symposium on Nanoscale Architecture", 2007.

[17] C. A. Moritz et al., *Fault-Tolerant Nanoscale Processors on Semiconductor Nanowire Grids*, in "IEEE Transactions on Circuits and Systems, Regular papers", vol. 54, n. 11, pp. 2422–2437, novembre 2007.

[18] T. Wang et al., *Heterogeneous 2-level Logic and its Density and Fault Tolerance Implications in Nanoscale Fabrics*, in "IEEE Transaction on Nanotechnology", vol. 8, n. 1, pp. 22–30, Jan. 2009

[19] S. Ahn et al., *A Floorprint-based Defect Tolerance for Nano-scale Application-Specific IC*, in "IEEE Transaction on Instrumentation and Measurement", vol. 58 , Issue 5, May 2009.

[20] K. L. Wang et al., *More than Moore's Law: Nanofabrics and Architectures*, in "Bipolar/BiCMOS Circuits and Technology Meeting, BCTM '07. IEEE", pp. 139–143, Sept. 30 2007 - Oct. 2 2007.

[21] C. Teodorov, P. Narayanan, L. Lagadec, and C. Dezan, *Regular 2D NASIC-based architecture and design space exploration*, in "Nanoscale Architectures (NANOARCH)", 2011 IEEE/ACM International Symposium on, pp. 70–77, 8-9 June 2011.

[22] C. Dezan et al., *Towards a framework for designing applications onto hybrid nano/CMOS fabrics*, Microelectronics J., Elsevier, n. 40, 2009.

[23] P. Narayanan et al., *CMOS Control Enabled Single-Type FET NASIC*, in "IEEE Computer Society Annual Symposium on VLSI", 2008.

[24] S. Frache, M. Graziano, and M. Zamboni, *A flexible simulation methodology and tool for nanoarray-based architectures*, Computer Design (ICCD), 2010 IEEE International Conference on, Amsterdam, pp. 60–67, 2010.

[25] A. Orlov, A. Imre, G. Csaba, L. Ji, W. Porod, and G.H. Bernstein, *Magnetic Quantum-Dot Cellular Automata: Recent Developments and Prospects*, in "ASP Journal of Nanoelectronics and Optoelectronics", vol. 3, n. 1, pp. 55–68, 2008.

[26] M. Graziano, M. Vacca, A. Chiolerio, M. Zamboni, *An NCL-HDL Snake-Clock-Based Magnetic QCA Architecture*, Nanotechnology, IEEE Transactions on , vol. 10, n. 5, pp. 1141–1149, September 2011.

[27] K.M. Fant, and S.A. Brandt., *NULL Convention Logic^{TM}, A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis*, International Conference on Application Specific Systems, pp. 261–273, Chicago-Illinois, USA, 1996.