Enabling Design and Simulation of Massive Parallel Nanoarchitectures

# Enabling Design and Simulation of Massive Parallel Nanoarchitectures

S. Frache[a], D. Chiabrando[b,a], M. Graziano[a,c], M. Vacca[a], L. Boarino[b], M. Zamboni[a]

[a] *Electronics and Telecommunications Department, Politecnico di Torino, c.so Duca degli Abruzzi 24, Torino, Italy*
[b] *NanoFacility Piemonte, Electromagnetism Division, INRIM (Istituto Nazionale di Ricerca Metrologica) Torino, Italy*
[c] *Corresponding author: E-mail: mariagrazia.graziano@polito.it*

**Abstract**

A common element in emerging nanotechnologies is the increasing complexity of the problems to face when attempting the design phase, because issues related to technology, specific application and architecture must be evaluated simultaneously. In several cases faced problems are known, but require a fresh re-think on the basis of different constraints not enforced by standard design tools.
Among the emerging nanotechnologies, the two-dimensional structures based on nanowire arrays is promising in particular for massively parallel architectures. Several studies have been proposed on the exploration of the space of architectural solutions, but only a few derived high-level information from the results of an extended and reliable characterization of low-level structures.

The tool we present is of aid in the design of circuits based on nanotechnologies, here discussed in the specific case of nanowire arrays, as best candidate for massively parallel architectures. It enables the designer to start from a standard High-level Description Languages (HDL), inherits constraints at physical level and applies them when organizing the physical implementation of the circuit elements and of their connections. It provides a complete simulation environment with two levels of refinement. One for DC analysis using a fast engine based on a simple switch level model. The other for obtaining transient performance based on automatic extraction of circuit parasitics, on detailed device (nanowire-FET) information derived by experiments or by existing accurate models, and on spice-level modeling of the nanoarray. Results about the method used for the design and simulation of circuits based on nanowire-FET and nanoarray will be presented.

## 1. Introduction

The remarkably successful era in computing, where Moore's Law reigns and processing power per dollar doubles every year, is approaching its end. Many attempts to keep up according to Moore's law have been put forward. Among

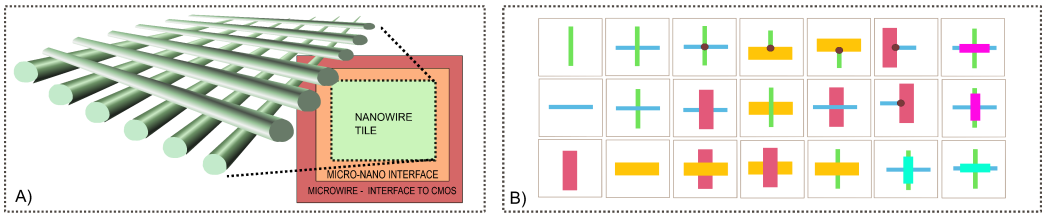


Figure 1: A) generic organization of a nanoarray fabric structure: the structure requires interface to the external world, a system of interconnects both based on micro and nano wires, and the array of nanowires. B) Different cross points (sub-tiles) composing the design if the focus is on an inner element of the matrix; elements can be microwires (bigger rectangles), nanowires (smaller rectangles), crossing among those sub-elements without or with contact (case with the small dot), nanowire FET transistors of N-type or P-type (represented by different colors here in the right column).

them there is massive parallelism. Actually, parallel computation has been subject of interest and driving topic up to 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, even though parallelism levels now feasible are more sizable than ever, they allow to achieve only a tiny portion of what could really be faced in certain breakthrough applications (i.e. biological related processing in medicine [8]). Thus, even though research and technology is expected to greatly improve in this field during the following years, the predicted limits of CMOS technology [1][2] will prevent substantial revolutions in the amount of information that can be processed in parallel.

Hence, the new era of nanoelectronics is on the horizon, with ever smaller devices and higher densities [9]. Different nano-structures have been recently explored [10], however some of them will be rejected on account of the feasibility [11][3][4]. For what concerns massive parallelism, nanowire arrays [21] [35][18], organized in matrices [22], which allow the creation of active nanodevices (diodes and FETs) in their crosspoints [23], have been proposed. These structures are generally organized in two-dimensional tiled arrays. In particular, nanoscale programmable logic arrays, e.g. nanoPLA [24], or molecular/nanowire array, e.g. CMOL [25], have been investigated. NanoASICs (NASICs) designs have been indicated as a valid way to reach denser designs with better fabric exploitation and efficient cascading of circuits with respect to general-purpose programmable fabrics (PLAs) [26], [27], [28]. Some authors [29], [30] proposed these structures for an optimal deployment of massively parallel architectures in specific applications, like cellular neural networks or image processors. Nevertheless, despite their promising characteristics, such structures have to deal with not negligible defect rates [12], mainly on account of critical manufacturing processes at nanoscale level. Defect tolerant techniques proposed for nanoscale arrays [31], [32], [33], [34], showed the importance of faults analysis and fault techniques application in nanoarray-based structures.

Nanotechnologies in general represent an emerging field of study in which

many questions are open. For example, the maximum density of devices obtainable after solving all the problems of reliability is still not known, neither which nanoarchitecture is best suited for a particular type of application. Epoch-making changing brings new challenges into play. New devices solicit the development of novel fabrication and integration approaches, novel simulation methods, and novel architectures to take full advantages from their potentialities. Moreover, design tools able to capture the specificities of these technologies are required, to explore the space of possible solutions, and to validate the proposed circuits and architectures. Hence, researchers need to revise the methodologies and design tools involved.

We propose a methodology to approach the abovementioned issues included in a tool we are developing: ToPoliNano (after Torino Politecnico Nanotech design tool). Even though not mature as an industrial level tool might be, the essential steps enabling the design and characterization of nanoarray based circuits are already implemented and are here presented. Starting from a VHDL circuit description, ToPoliNano can aim at different disruptive nanotechnologies (Nanoarray-based circuits as a possible example, as in Figure 1.A, but not limited to), to place and route circuits on a low-level floorplan (please, see later section for details) and then to simulate it, in an integrated fashion, at both switch level and device level.

The aim of this tool is to analyze complex systems based on emerging electronic nanotechnologies. The study was specifically focused on the assessment of dimensions, performance, power consumption and, as a consequence, in developing optimized architectures. The ability to base its high level analysis of complex structures onto low-level information is a key aspect of ToPoliNano. This low-level information is derived from actual device technology, circuit topology and circuit layout post-placement 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: previous work in the field and our case study structure are exposed section II; the general organization and features of ToPoliNano software are described in section III; its low-level floorplanning capabilities and related results are shown in section IV; logic simulation and results are reported in section V; timing simulation are faced in section VI.

## 2. Previous works and case study structure

Circuits based on nanoarrays have been largely investigated in the last years [11], [12], [24], [25], [28]. Their main attractive features are the high theoretical density of devices that can be obtained, the high frequency of operation, the low power consumption [8], and the possibility of being integrated with CMOS structures. However, the detailed study of circuits and architectures, considering at the same time device density, device defectiveness, power consumption, frequency of operation, etc, appears critical, on account of the interdependence of the obtained results. The assessment of main parameters affecting the feasibility of an architecture and its performance plays a fundamental role, in order

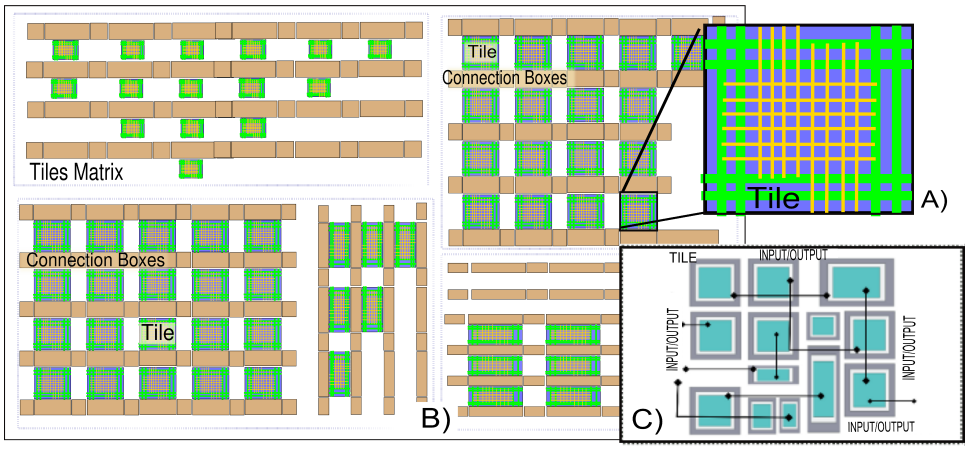


Figure 2: A) A sketch of the nanoarray like in figure 1.A recalling in this example the NASIC structure. B) A higher level architecture based on matrices of tiles. C) Identification of constraints for the place and route phase arising when considering a complex architecture.

to establish the priority in actions to be undertaken in a given technology and to identify the key aspects to be further investigated. Authors in [13][36] give an important overview of a possible architecture based on the regular composition of variably sized NASIC tiles, and this kind of exploration is essential towards a real implementation. Actually, the second and natural step in this method is the possibility to evaluate the effect of the modification of a transistor's technological parameter on the performance of a realistic circuit, as proposed in [13]. To be consistent, a third step should be undertaken, based on three main points. i) The actual layout of the circuits, after all the constraints of placement and routing have been taken into account together with all the device and interconnects parasitics, has to be considered in the performance evaluation to reduce over or under estimation. ii) This evaluation should be executed easily and if possible automated in order to allow the architect to make his own analysis. Moreover, iii) a different resolution in terms of accuracy should be available, to enable evaluation speed on one side or accurate device level analysis on the other side, according to the designer needs. This third step is the one we are proposing with our method.

Besides the benefits of these technologies listed before, there is research interest about the importance of handling the high defectiveness of these structures, which has been investigated also in [12] [14] [32], [34]. Different techniques of fault tolerance have been applied to these fabrics; in the form 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 underlined by the literature is the reduced availability of design and

simulation tools expressly focused on the detailed design and simulation of these structures. One system level approach (the first step) has been developed in [37], 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 specific fabric (e.g. computational, architectural, technological and fault), this framework fits the need of the designer to compare different nanoarray approaches. Actually, high level models of a whole tiled nanoarray is the ground on which the system is based. Whereas, 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 (the second step) has been proposed in [38], 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. Recently an accurate analysis including noise resiliency has been discussed in [13] considering architectural level implications. In our case, it will be shown by means of the obtained results, the tool enables to simultaneously consider technological parameters, circuit style, topology, layout, and to efficiently analyze the behavior of complex architectures. This is a pre-requisite for the further step, i.e. a simulator able to adapt to the evolving fabric styles proposed in literature and to accurately take into account the extracted circuit parasitics and the post fabrication device characteristics.

About the defect analysis, we have addressed the point in a previous work, with a preliminary version of this tool [39]. The possibility to perform this kind of analysis has been broaden to arbitrarily complex circuits, and is recently integrated into ToPoliNano. However, it is not dealt with in the present paper. Similarly, for the sake of brevity, we overlook the investigation and implementation of the power consumption analysis capability in this paper, but information about the topic can be found in [8].

To introduce ToPoliNano, we chose a crossed-nanowire based architecture, as is shown in Figure 1.A and 2.A. Actually, this tool is able to manage different nanoarray-based architectures. The way in which ToPoliNano is capable of treating modification of the same technology, and ultimately totally different nanotechnologies, is detailed in the next section. In particular, we choose the NASIC architecture as a case study as one of the most interesting nanoarray structure. According to its proponents [26], 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 already highlighted, 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,

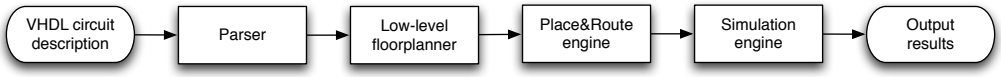


Figure 3: Basic ToPoliNano flow.

NASICs have been proposed in both static-ratioed and dynamic styles [26], with the latter that enables pipelining and overcomes the many restriction of a static design. Eventually, micro-wires are used to carry power and control signals from the CMOS level. Faults are controlled by masking them in the circuit and/or architecture design itself, implementing a multi-tiered built-in fault tolerance approach [14]. Dataflow in NASICs is through a multi-phase progression and the control signals from the CMOS level coordinate these phases. Internally, the tile can be seen as a matrix of elements that we baptized sub-tiles, give by the various combinations of micro-wires, nanowires, connections and nanowire-FET (as in figure 1.B). These will be referred to later in the description of the tool behavior.

## 3. ToPoliNano general organization

The description is here organized in a few subsections addressing the most important features and ideas behind the method.

### *3.1. How to cope with a multi-technological scenario.*

ToPoliNano is a CAD tool for design and simulation that supports a variety of nanotechnology, from NML [4][20] to nanoarrays based on SiNWs or CNTs. This choice stems from the importance of evaluating several promising technologies, among which a winner, if ever one will prevail, is yet to be determined. It is necessary to evaluate nanotechnologies in different fields of application and with specific architectures for each one.

Many ways lead to the support of different technologies. The most obvious one, but also the most expensive in terms of lines of code and development time, is to write specific code for each technology. A less obvious approach, but effective in saving both lines of code and time, is to exploit the similarities existing even between totally different nanotechnologies, to unify large portions of code, which turn out to be shared. One might think that this way will necessarily lead to inefficiencies, maybe in terms of memory usage, or perhaps in the form of long simulation times. The benchmarks we performed to assess the timing of design and simulation phases say quite the opposite, as we show in the results (see section 4.2). Each nanotechnology fabric has unique characteristics but, at the same time, shares technological constraints, which allow for unification of certain fundamental aspects. A basic underlying requirement is a two-dimensional array that act as the fundamental structure for computing [35], [22], [24]. This is evident from the crossbar structures, but for instance it is perfectly applicable also to the case of NML.

*3.2. The "building brick" principle.*

In literature there are numerous variations on the theme of architectures based on crossed nanowires [28],[25]. 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 and foremost, the regularity of the 2D array. One can imagine the plane of the circuit covered with set of tiles of different sizes, like in figure 2.B. Each tile, as in figure 2.A contains a maximum of three components, in definite positions.

With reference to Figure 1.B, the variety of basic sub-tiles that can be created with a reduced set of elementary components can be seen. To design a NASIC circuit, in fact, only 5 such elemental components are necessary, rotations excluded: p-type and n-type transistor, contact, nanowire, microwire. By composing these items one can get all that is needed to describe an arbitrarily complex circuit.

*3.3. The recursive hierarchical composition principle.*

Another key aspect in the generalization of the approach is the application of a principle of recursive hierarchical composition. In other words, each element may be part of the architecture at any level of the hierarchy. On condition that 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. This has enabled us to develop a distinct piece of software (a component generator) that can be used to describe sub-circuits to be used in the design of complex and optimized circuits 2.B.

*3.4. The fundamental steps.*

ToPoliNano integrates all the tools required to design and simulate circuits based on the so-called disruptive nanotechnologies in just one cross-platform tool. It actually runs on the three major operating system platforms: Linux, Mac OS, Windows. It has been developed in C++, and currently counts nearly 100k lines of code, external libraries excluded. The main application organization and flow is briefly discussed: see Figure 3 for a very simplified one.

On application's first launch, the user is presented a wizard, to simplify the configuration of the tool. It will ask the user 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 or LSI files.

*Parser.* The tool features a HDL parser, presently implementing essential parts of the VHDL93 specification. A design can be described at different abstraction levels, i.e. in terms of elemental components (basic patterns of subtiles, 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

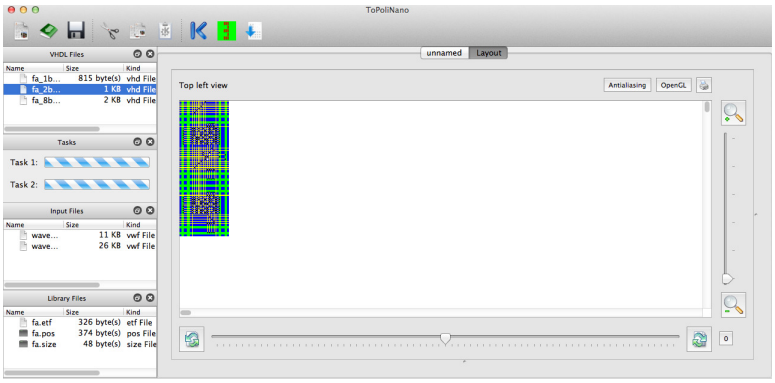


Figure 4: A snapshot of the tool GUI during the layout phase execution and display.

description of the circuit; the VHDL description recalls the components through the 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. As an alternative to VHDL, the circuit can be entered in the form of LSI files.

*Place and Route engine.* 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. Figure 2.C shows the constraints (nanowire and microwire pitch and size, non routable areas, microwire dedicated areas, position of inputs and outputs, etc.) that have to be taken into account at the lowest floorplanning level (see in next session a discussion on this point). This is one of the design steps which differs largely 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. In Figure 4 the tool interface when the layout is executed is shown.

*Simulation engine.* Since the tool supports a variety of nanotechnologies, so does the simulation engine, which must 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 aspect of the low level simulations that must be customized, because the physical implementations differ. To maximize the portion of code that can be shared across technologies, an event driven approach was adopted. This will be further discussed in section 5. In order to refine the analysis, not only a switch level simulation engine is implemented (section 5), but an automatic extraction system is available which describes the circuits in terms of spice-level elements and includes table models for accurate simulation (see section 6).

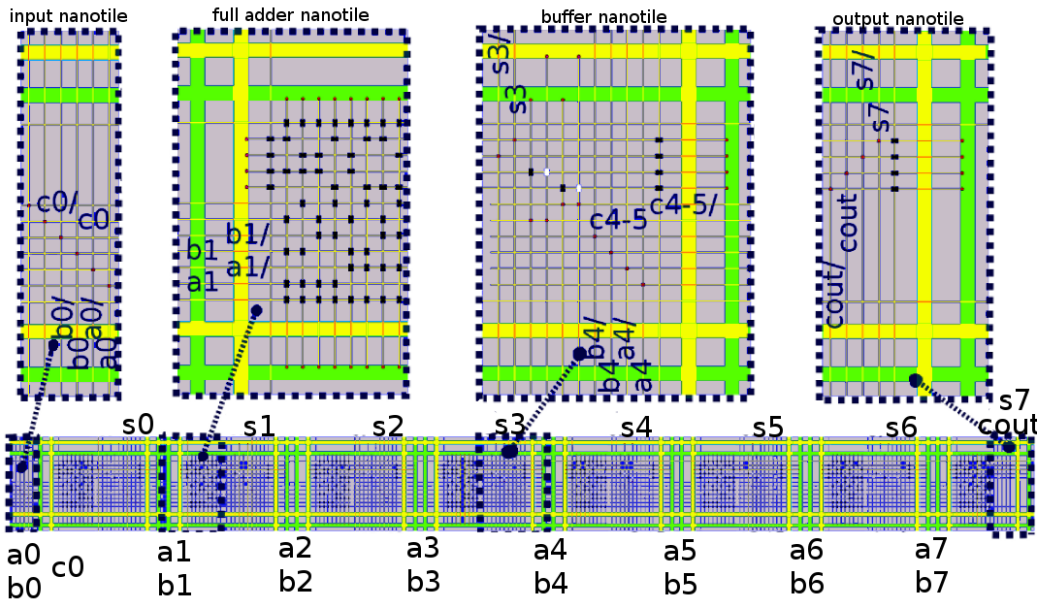


Figure 5: Layout of a eight bit adder (bottom row). A zoom is given for the input tile, one of the 1bit FA tile, a buffer tile to connect different FA, and an output tile.

## 4. Low level floorplan: discussion and results

### *4.1. Dynamic floorplanning and placement*

Each nanotechnology copes with the constraints related to the physical characteristics of the devices it exploits. If Silicon nanowires and Carbon nanotubes show similarities in the physical structure, though they feature distinct technological parameters, things are different for technologies such as NML, based on nanomagnets. The constraints to which the devices are subject are essentially different [4][5][6][7].

In the case of nanowires/nanotubes the constraints are those common for a crossbar structure.

In Figure 2 the main constraints of a nanoarray design are shown. A slicing design is required for the tool to be able to operate. Each part of the design can feature different tile size. For the tiles to operate correctly, and for proper routing of information, interconnection boxes are automatically placed by the tool. They do serve different purposes: they allow inputs to come from different directions, they permit routing of signals from one tile to another, not necessarily adjacent of course, and they output data from previous tile in the last phase of the dynamic control sequence. In particular, with reference to Figure 2.C, besides the wire pitch is among the constraints to take into account (affecting the size of the area in aquamarine), there is the area allocated to the microwires (in light gray), the routability of interconnections among tiles (black lines with

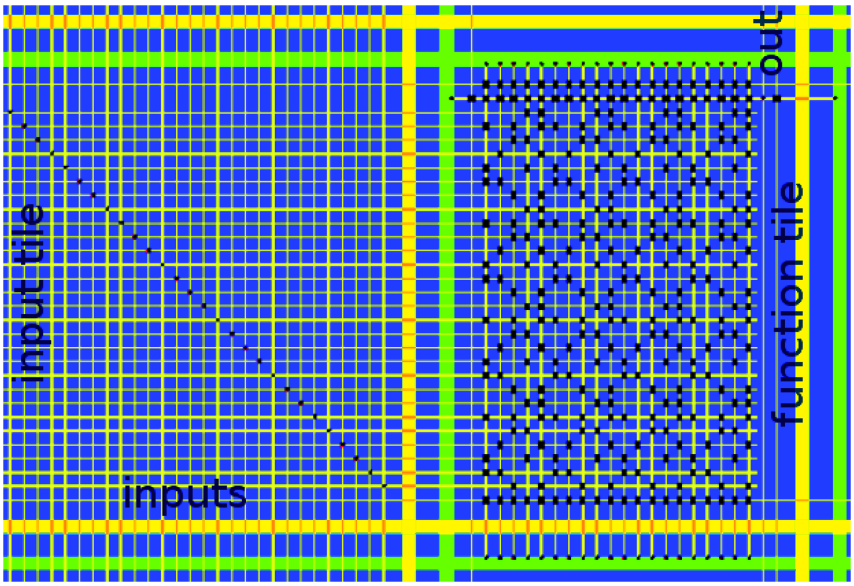


Figure 6: 28 input logic function layout.

ending dots), the position of inputs and outputs (both primary and secondary), the alignment of blocks of different size, etc.

The constraints that the different technologies put to the positioning the devices on the plane of the circuit require an appropriate partitioning of the space available. We call *low-level floorplan* the peculiar partitioning of the space at device level that depends on the specific technological constraints. With the term floorplan we do not refer to partitioning at the level of functional units, to which it usually refers to.

To capture the constraints of different technologies, while maintaining a common data structure, we introduced an abstraction at circuit level, in particular regarding its subdivision. 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, allowing a greater reuse of the code.

Dynamic low-level floorplanning is performed through operations on the common graph structure. This allows for resizing, shifting 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, and are referred to the root of the part with a single pointer, it is lighting-fast to move entire portions of a circuit from one point of the low-level floorplan to another.

*4.2. Results*

In Figure 5 the layout of an 8-bit adder, based on eight 1bit Full Adders, after VHDL parsing, placement and routing on a low-level floorplan is shown. 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 is not a conventional NASIC designs, but represents an evolution of the original design. The evolution here proposed derives by the analysis of the technological constraints when conceiving the method to implement the whole feasible layout. This evolution that comprises a modification of the structure to manage the control signals of the vertical nanowires, now included in the buffer tile. Clearly this is not the only possible solution, however it allows to demonstrate the correctness of the top-down design approach. It is possible for the tool to automatically generate this kind of tile that is responsible for the routing of the input signals, as shown in the same figure.

To demonstrate the capability of the tool we show in Figure 5 the layout of a 28-input function obtained after automatic synthesis and placement on a low-level floorplan. The function is as in equation (1): the function significance is not important here, while it is worth noticing the complexity in terms of number of inputs and number of logic combinations.

$$\begin{aligned} f = \quad & a_1a_4a_6a_7a_8b_2b_4b_5b_6b_9c_2c_3c_4c_7c_9d_1 + a_3a_5a_7b_1b_3b_5b_8c_1c_3c_6c_7d_1 \\ + \quad & a_1a_5a_6a_8b_3b_4b_6c_1c_2c_4c_8c_9 + a_2a_4a_9b_2b_7b_9c_5c_7 \\ + \quad & a_2a_3a_6a_9b_1b_4b_7b_8c_2c_5c_6c_9 \end{aligned} \tag{1}$$

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.

The performance of the tool have been tested on a pretty basic machine, to underline the efficiency of the implementation. Preliminary benchmarks conducted on a Linux box (Ubuntu 10.10, Core2DUO t8300 processor) have been performed on designs of different sizes. The instantiation of one FA tile 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 $2.6{\cdot}10^6$ subtile actually instantiated, just required 819 Mb main memory and 837 Mb Virtual Memory.

## 5. Logic simulation: discussion and results

*5.1. Logic simulation engine*

The simulation engine belongs to the class of the event-driven simulators. It monitors the information flow inside the structure under simulation and induces specific events, when necessary, to correctly handle the propagation of information. In order to better understand this process, it is helpful to turn back to the sub-tiles and their regular structure, as shown in Figure 1.B. Each sub-tile can be seen as a four-port device, with each port identified by a cardinal point. A

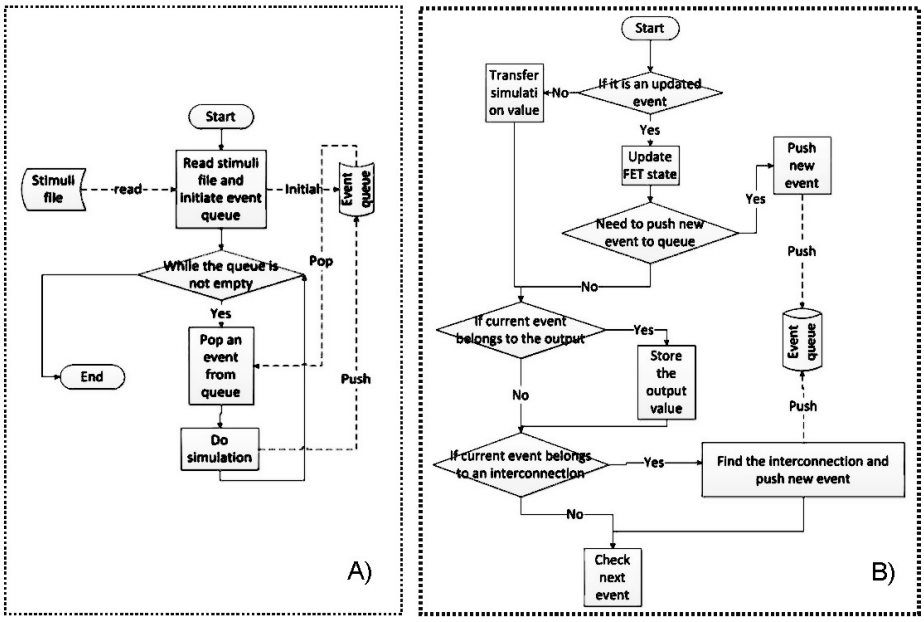


Figure 7: Flow diagram of the switch level simulation engine. A) High level view of the sequence of event management. B) Detailed view of the events management at sub-tile level.

change in the information at a certain port may need to be propagated inside the sub-tile, if an appropriate component supporting the propagation (e.g. a nanowire). Nothing about the electrical properties of the component is needed to be known to perform a logical analysis. Actually, the possibility to support further propagation of the information is a function of the port at which the change in information occurs, and the original direction of propagation of this piece of information. This is true also for the evaluation of a change of 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. A generic scheme on the event management is in figure 7.A. By following the very same process, the information is propagated inside the structure, only if necessary. An active device inside the sub-tile could be present, and the propagation of information could lead to a change in its status. In this case, 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. A more detailed flow for this algorithm is in figure 7.B. This approach is very pliable, indeed, because it allows for different kind of control of dynamic circuits (number of phases). This is possible thanks to the phase sequence coded in the input control sequence and not embedded into the simulator. The same approach can thereby be used in many different scenarios. In addition, it is also quite efficient, as pointed out by the following

results.

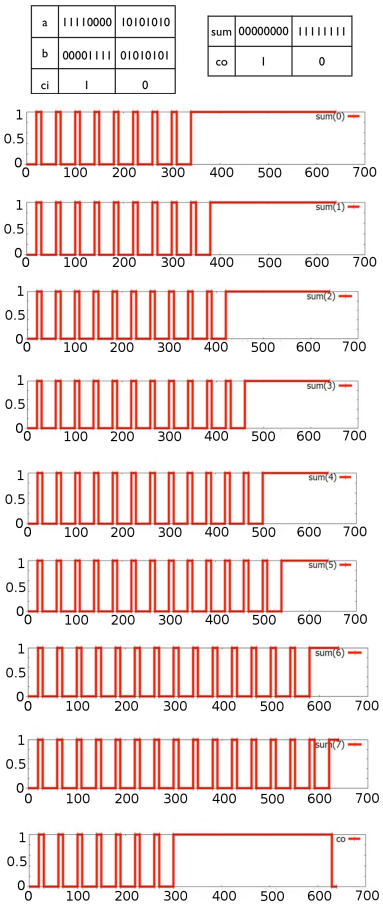


Figure 8: 8bit Adder logic simulation results.

*5.2. Results*

We performed logic simulations on both the 8bit Adder of Figure 5, and the 28-input function of Figure 6.

Figure 9 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 8 shows the correct output result for the full adder in the case of th input values reported in top of 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.

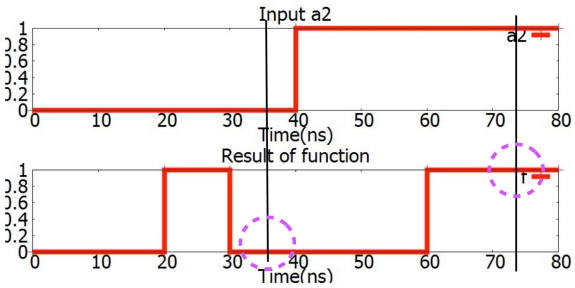


Figure 9: 28-input function logical simulation result. The function is evaluated for input $a2$ transitioning from logic value 0 to logic value 1.

## 6. Electrical simulation: discussion and results

### *6.1. Electrical modeling and spice-compatible netlist extraction*

In order to be able to perform both accurate DC and transient simulations, care must be taken in accounting for transistor and interconnects behavior and in evaluating all the parasitics that, inevitably, affect the layout of a circuit once it has been placed and interconnected. The extraction of these parasitic parameters takes place automatically in ToPoliNano, from the actual circuit post-placement, i.e. after the final layout is set.

The R and C parameters have been calculated from geometrical data, alongside with materials characterizations. Capacitances, in particular, account for coupling among interconnections, and strongly depend upon circuit layout.

To illustrate the principle by which parameter extraction is being conducted, we refer to Figure 10.A, where a pair of crossed wires is shown. In a real-life circuit, wires may be either nanowires, or microwires or both kind, and in some of the crossing point a transistor might be present. In Figure 10.B a representation in terms of equivalent model of the same couple of crossing wires is depicted. Our approach has been to associate RC ladders to interconnects, and to include specific models for the transistors.
Wires in Figure 10.C have a distributed capacitance and a resistance, and also a coupling capacitance, as shown in Figure 10.D. It is possible to calculate these quantities by geometric considerations, starting from the structure of the circuit and the definition of the materials of the parts. We previously showed (see Figure 1.B) how a NASIC circuit can be thought-of as composed by sub-tiles. This approach still holds 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, as a function of the technology node set by the user, and automatically calculate the required values. In Figure 10.E as scheme of the extraction flow executed by ToPoliNano is shown. Once these parameters have been calculated, the tool can generate a spice-compatible netlist. In fact, there are small differences in syntax among simulators, e.g. UC Berkeley and Mentor Graphics Eldo Spice, so the netlist can be generated in both formats.

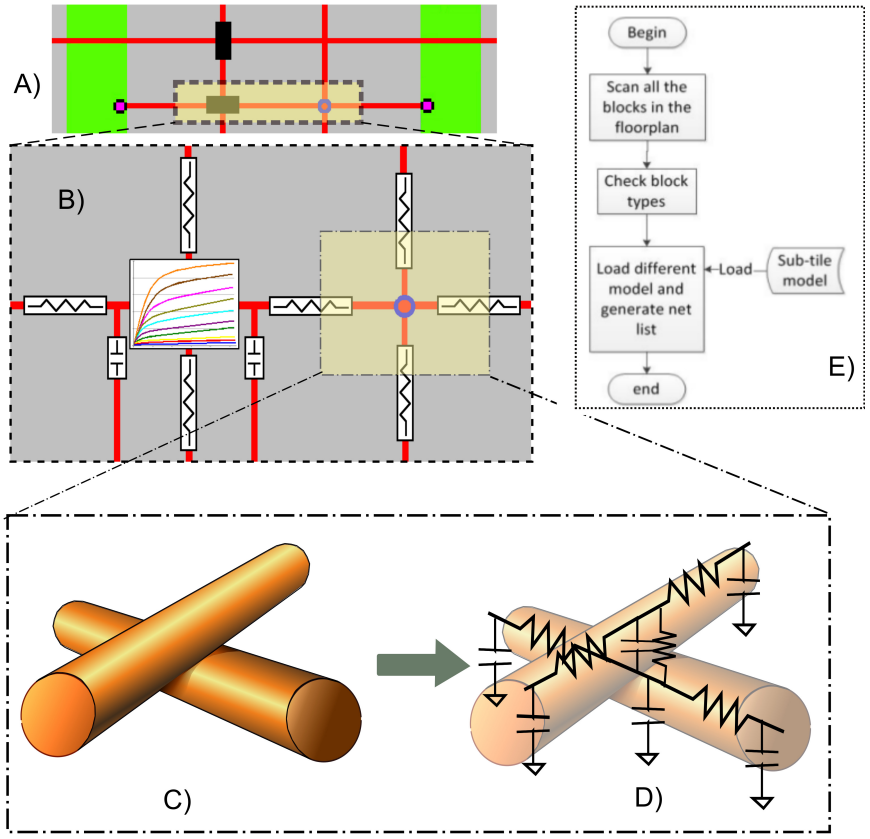


Figure 10: Example of parasitics in a pair of crossed wires.

In case the sub-tile encountered while scanning the tile is an active device (like in Figure 11) then we refer to a model in library and to geometrical consideration. To account for transistor behavior, in general, different approaches can be followed. Transistor action can be captured by means of numerical simulations. There are different approaches in numerical simulations too, but many commercial software simulators (e.g. Synopsis Sentaurus Device, Silvaco Atlas, etc.) are physics based and provide a fairly good degree of accuracy. The drawback with TCAD simulations is the time required to complete the analysis of a single device. TCAD simulations do not scale at circuit level, being confined by time constrains to the simulation of just few devices.

This well known problem has led to the development of compact models, to achieve accuracy and fast simulations. One of the main problems with compact models is the time required to develop one such model, a great effort for just solution space exploration. A third way, then, must be pursued if the objective of the research is to explore innovative devices exploitation at circuit level, like for example the approach in [19]. It is well known that is possible to

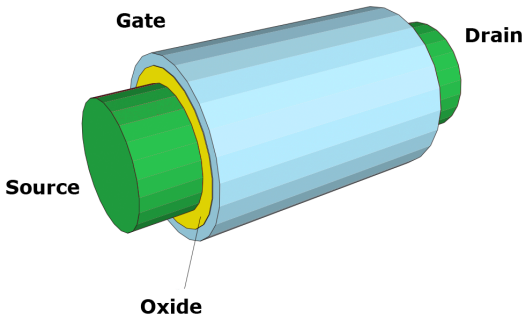


Figure 11: Structure of a Gate-All-Around Silicon nanowire-FET.

build Verilog A models, in the form of table models, with data coming from accurate numerical simulations. Out of a relatively small set of data, that can be further interpolated by the simulation software, it is possible to gain a thorough description of an active device. This kind of model can be fast to evaluate, according to designer choices that will be further discussed. What is lost is predictive capability, which is very limited in table models, while still strong in compact models.

The conceptual framework for such kind of models can be thought-of being made out of three parts: a table, whose entries capture electrical characteristics, a search function, that given bias values, searches in the aforementioned table for the nearest entries, and interpolation routines, to provide values not entered in the tables. About this third part, different interpolation and approximation methods exist: polynomial interpolation (linear, quadratic, exponential), B-spline approximation, combined interpolations, etc. A comparison among different interpolation methods is beyond the scope of this paper, but let us resume the reasons behind the choice of polynomial (linear) interpolation for the simulations in this work. Linear interpolation is computationally simple, and it preserves monotonicity of data, a stringent requirement. Accuracy is controlled by table density, so you can get the required accuracy level by entering more data into the table. Of course, this comes at the price of computational simplicity, but tradeoffs can be found. Among the main drawbacks there are discontinuous first derivatives and the relatively large tables that are needed for good accuracy.

The table model approach has several advantages also when it comes to contrast and compare the behavior of different devices into the same circuit topology. This is possible by simply modifying the data in the table. In this work we tested two models. A conventional model, hereinafter MODEL-1, is a Si-NW transistor, modeled with Fettoy [17] assuming ballistic behavior (optimistic), for the featured size and technological parameters in table 1.We also tested, hereinafter MODEL-2, the junctionless crossed nanowire field effect transistors

| | |
|---|---|
| Channel width: | 10nm |
| Gate width: | 10nm |
| Gate Oxide thickness: | 2nm |
| Bottom oxide: | 10nm |
| Channel doping: | $4 \times 10^{19}$ dopants/$cm^3$ |
| Gate doping: | $8 \times 10^{19}$ dopants/$cm^3$ |
| Substrate bias: | $-3$V |

Table 1: Characteristics of the device modeled for which table modes have been created.

proposed by [15], starting from data published in the referenced paper, coming from numerical 3D simulations. We built the table model accordingly, in order to simulate a device with the characteristics in table 1.

The resulting characteristic curves for the two models are in figure 12, where clearly MODEL-1 shows higher currents with respect to MODEL-2, being more ideal and based on theoretical derivations.

*6.2. Results*

We demonstrate here the capability the tool has to extract, to capture the correct model and to prepare a final netlist for an external simulator on three structures of increasing complexity. Two are in figure 13: a ring-oscillator and a Full-Adder with a standard scheme, and the third is the NASIC adder (8 bits) based on 8 FA previously described in the layout and switch level simulation sections.

We used for the ring-oscillator the two transistor models, MODEL-1 and MODEL-2. In both cases we show the results obtained both without and with extracted parasitic capacitance at the transistor Drain and Source connections. Figure 14 shows on the top row results from MODEL-1 and on the bottom row results from MODEL-2. Left results are obtained without capacitance, right results derived by including the following capacitances: $C_{gate-reference} = 7.5aF$, $C_{gate-source} = 4aF$, $C_{gate-drain} = 3aF$, $C_{drain-source} = 3aF$, $C_{source-reference} = 10aF$.

The presence of parasitic capacitances impact the semi-period of the oscillation of a similar factor (more than 3 times bigger) in both cases. MODEL-2 reduces the oscillation frequency of a factor bigger than 4, which is coherent with the on current.

For the Full-Adder results are in figure 15 reported for MODEL-1 with capacitance only for the sake of brevity. Finally, figure 16 shows the behavior of the spice simulation of the whole adder based on NanoASIC previously designed in the case of MODEL-1 with capacitance and including parasitic resistance and capacitance for all the interconnects.

## 7. Conclusion

We illustrated the methodology we propose and the tool we developed for the design and simulation of circuits suitable for parallel architectures based on

emerging technologies. In particular we focused here on a NASIC structure as a working platform.

ToPoliNano allowed to design, place, route and simulate the behavior of: i) a basic arithmetic circuit (an 8bit Adder), ii) a random 28-input function in NASIC nanotechnology, iii) a ring-oscillator, iv) a standard Full-Adder. All circuits were subject also to post-layout automatic parasitic parameter extraction, and netlist based on RC networks and table models to include transistor behavior were fed to spice-compatible simulation software, to get transient analysis. All the simulations were benchmarked, and the benchmarking results show good performances, both in terms of execution speed and memory footprint.

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.

## 8. Acknowledgement

The authors would like to thank the Nanofacility Piemonte and Compagnia di San Paolo for the support.

## References

[1] International technology roadmap of semiconductors, 2008 update, http://public.itrs.net (2008).

[2] A. Pulimeno, M. Graziano, G. Piccinini, Udsm trends comparison: From technology roadmap to ultrasparc niagara2, Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 20 (7) (2012) 1341–1346.

[3] A. Pulimeno, M. Graziano, A. Sanginario, V. Cauda, D. Demarchi, G. Piccinini, Bis-ferrocene molecular qca wire: Ab initio simulations of fabrication driven fault tolerance, Nanotechnology, IEEE Transactions on 12 (4) (2013) 498–507.

[4] M. Graziano, M. Vacca, A. Chiolerio, M. Zamboni, A ncl-hdl snake-clock based magnetic qca architecture, IEEE Transaction on Nanotechnology (10) DOI:10.1109/TNANO.2011.2118229.

[5] M. Graziano, A. Chiolerio, M. Zamboni, A technology aware magnetic qca ncl-hdl architecture, in: International Conference on Nanotechnology, IEEE, Genova, Italy, 2009, pp. 763–766.

[6] M. Graziano, M. Vacca, D. Blua, M. Zamboni, Asynchrony in quantum-dot cellular automata nanocomputation: Elixir or poison?, IEEE Design & Test of Computers.

[7] M. Awais, M. Vacca, M. Graziano, M. Roch, G. Masera, Quantum dot cellular automata check node implementation for ldpc decoders, Nanotechnology, IEEE Transactions on 12 (3) (2013) 368–377.

[8] 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.

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

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

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

[12] P. Narayanan, M. Leuchtenburg, J. Kina, P. Joshi, P. Panchapakeshan, C. On Chui, C. A. Moritz, *Variability in Nanoscale Fabrics: Bottom-up Integrated Analysis and Mitigation*, ACM HJournal on Emerging technologies in Computing Systems. Vol. 9, issue 1, 2013.

[13] P. Narayanan, J. Kina, P. Panchapakeshan, C. O. Chui, C. A. Moritz, *Integrated Device-Fabric Explorations and Noise Mitigation in Nanoscale Fabrics*, IEEE Transactions on Nanotechnology, vol. 11, no. 4, pp. 687 -700, Jul. 2012

[14] M.M. Khan, P. Narayanan, P. Joshi, M. Leuchtenburg, P. Panchpakeshan and C.A. Moritz, *FastTrack: Towards Nanoscale Fault Masking with High Performance*, IEEE Transactions on Nanotechnology, vol. 11, no. 4, pp. 720-730, Jul. 2012.

[15] P. Narayanan, P. Panchapakeshan, J. Kina, C. O. Chui and C. A. Moritz, Integrated Nanosystems with Junctionless Crossed Nanowire Transistors, IEEE International Conference on Nanotechnology (IEEE NANO 2011), pp.845-848, 15-18 Aug. 2011.

[16] P. Narayanan, K. W. Park, C. O. Chui and C. A. Moritz, Validating Cascading of Crossbar Circuits with an Integrated Device-Circuit Exploration , IEEE/ACM Symposium on Nanoscale Architectures(NANOARCH'09), Jul. 2009.

[17] A. Rahman, J. Wang, J. Guo, Md..S. Hasan, Y. Liu, A. Matsudaira, S.S. Ahmed, S. Datta; M. Lundstrom (2009), "FETToy," http://nanohub.org/resources/fettoy.

[18] I. Ercan, N.G. Anderson, Tight-Binding Implementation of the Microcanonical Approach to Transport in Nanoscale Conductors: Generalization and Analysis, Journal of Applied Physics 107, 124318 (2010).

[19] E. Albert, A. Abdellah, G. Scarpa, P. Lugli Electronic transport modeling with HSPICE in random CNT networks Nanotechnology (IEEE-NANO), 2012 12th IEEE Conference on (2012)

[20] J Das, S.M. Alam and S. Bhanja, *Nanomagnetic Logic For Low Energy High Density Circuits* VLSI Systems I: Regular Papers, IEEE Transactions on, in press.

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

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

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

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.

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

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

[31] 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.

[32] 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.

[33] 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

[34] 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.

[35] 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.

[36] 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.

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

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

[39] 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.

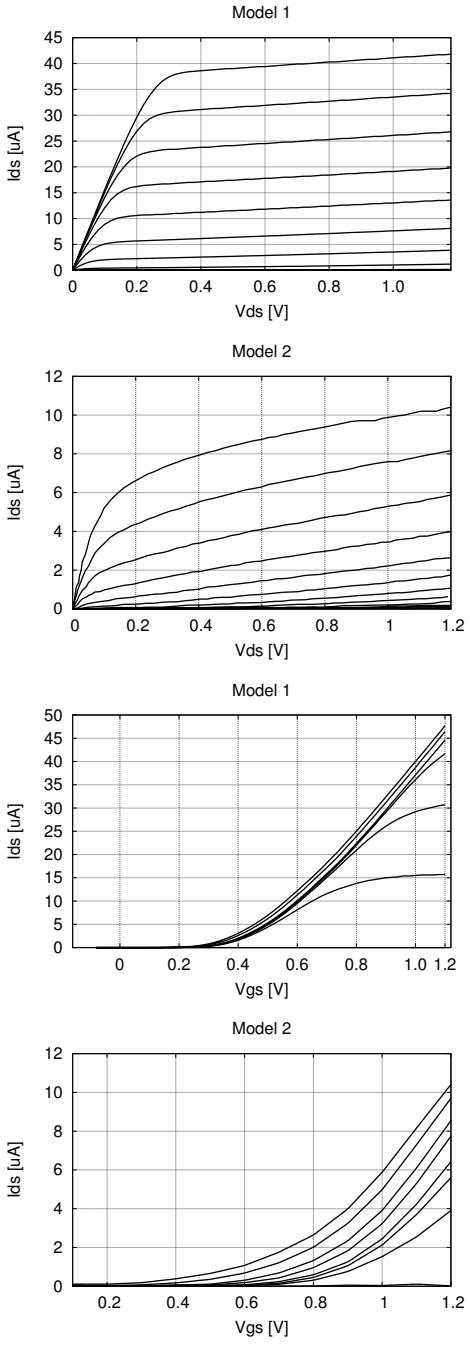


Figure 12: Gate-All-Around SiNWFET currents ($I_{ds}$ vs. $V_{ds}$ top and $I_{ds}$ vs. $V_{gs}$ bottom), for two different transistor models. Left figures are for MODEL-1 [17], while right figures are for MODEL-2 [15].

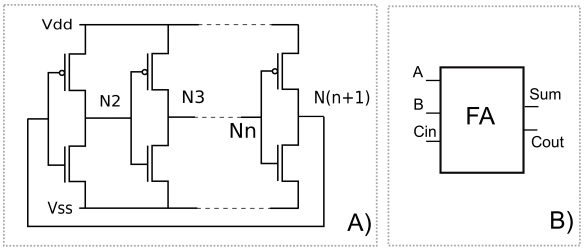


Figure 13: A) Scheme of the ring oscillator, B) Scheme of the Full Adder.

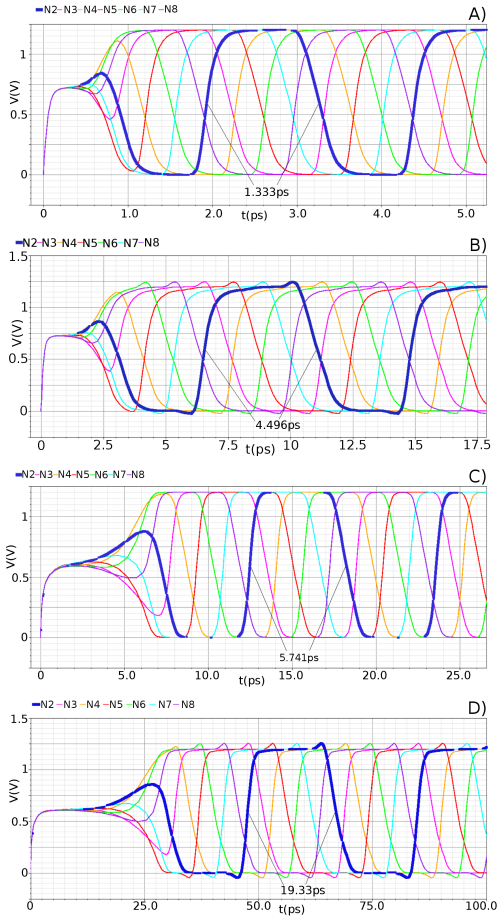


Figure 14: Simulation of the ring-oscillator, N2 in evidence. A) Model 1 without parasitics capacitance, B) Model 1 with parasitic capacitance, C) Model 2 without parasitic capacitance, D) Model 2 with parasitic capacitance.

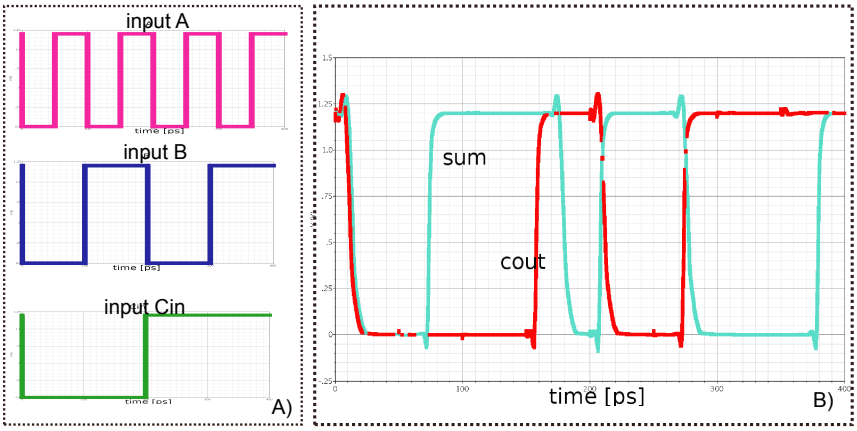


Figure 15: Simulation of the Full Adder obtained with Model 1 with capacitance.

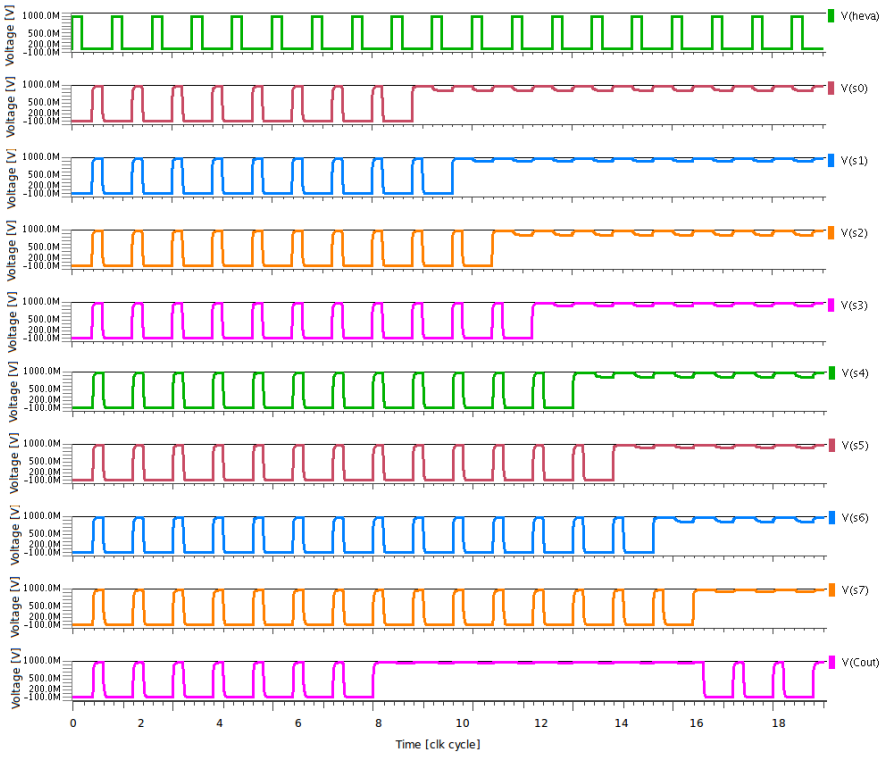


Figure 16: Waveforms of the 8bit Adder simulations for the same inputs of Figure 8.