A Flexible Simulation Methodology and Tool for Nanoarray-based Architectures

A Flexible Simulation Methodology and Tool for Nanoarray-based Architectures

Stefano Frache, Mariagrazia Graziano and Maurizio Zamboni
Electronics Department, Politecnico di Torino, Italy
{*stefano.frache, mariagrazia.graziano, maurizio.zamboni*}*polito.it*

***Abstract*— Nanoscale arrays based on nanowires are expected to have a promising future thanks to their amazing density and regularity. Experiments demonstrated the feasibility of this technology and pointed out that accurate reliability analyses should be accomplished to assure proper yield requirements. Due to the complexity of these systems and the arising necessity of thorough fault analysis, design automation tools are mandatory in order to explore architectural solutions and fault tolerant approaches deriving information from reliable nanoarray characterisation.**

We present a simulator, never attempted at this level of detail, based on specific technological and topological tiled nanoarray descriptions, conceived to carry on characterisations in terms of logic behaviour, defect-induced error rate assessment, switching activity and other figures of merit like power and timing performance (not discussed in this paper). It is formulated in a flexible and modular way to assure the simulation of manifold advancing technological solutions, among which the winner has not been determined yet. Marking a difference with respect to the state of the art, the algorithm is based on an event-driven engine and not on cost functions evaluations. Thus even dynamic control sequences can be processed and their evolution followed throughout all the inner components of the array allowing to obtain system level characterization as a projection of the real internal parameters.

In this paper we show results attained for one of the possible nanoarray structures proposed in literature, the NASIC: logic behaviour, defect error rates and switching activity for two types of function demonstrate the simulator trustworthiness, its effectiveness for extensive nanoarrays characterisation and its suitability as a foundation for both higher architectural and lower device simulation levels.

I. Introduction

Parallel computation has been a driving topic since the development of integrated architectures, and is now even more a reality with multiprocessors systems thanks to the integration capabilities reached by scaled technologies. However, parallelism levels now possible, 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). 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. On the contrary, nanoscale array structures, albeit still in their infancy both from technological and design points of view, show promising perspectives in many possible applications and in particular in the direction of massive parallel computing structures [3].

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. Recently NASICs designs have been proposed in [1], [11] as a way to achieve denser designs with better fabric utilisation and efficient cascading of circuits with respect to general-purpose programmable fabrics (PLAs). Authors in [12] and [13] 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 [14], [15], [16], [17], thus clarifying that faults analysis is mandatory when dealing with nanoarray structures.

There has been extensive work on the subject of defect tolerant architectures, for instance by [18], and there is no need to further emphasize the importance of this kind of work based on computation models for nanoscale logic gates. Notwithstanding, there is still a lack for a comprehensive simulation tool that can work on actual nanofabric designs, taking into account technological process-dependent fault-probability distributions that characterise elemental components (wires and devices in a well defined technology) and that can devise information such as the reliability of a processor made up of the said components.

One system level approach has been developed in [19], 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 behaviour characterization.

In the field of synthesis for reconfigurable nanofabrics extensive work has been done [20], but many promising architectures are inherently not reconfigurable. The aim of this work is to propose a new methodology and tool to validate nanoscale designs with respect to different parameters (such as fault rates, switching activity and more to come) in order to assess their ability to meet the required reliability and performance levels. We are not providing a synthesis tool, but a simulation one that has been conceived from the ground up to be tailored to different nanoscale architectures, be they based on static or dynamic logic style and irrespective of the actual underlying technology.

A device-level approach has been proposed in [21], 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.

The aim of the simulator we are developing is thus the study of complex systems based on emerging electronic nanotechnologies, with particular emphasis on architectures that can exploit their many peculiarities. More specifically, we are interested in exploring techniques proposed to solve the problems of reliability of these devices [16], in identifying the most suitable methods of control in dynamic systems, in studying power consumption, dimensions, performance and, as a consequence, in developing optimised architectures.

Though we aim, like in [19], 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 include in nanofabric design descriptions characteristics derived from technology, dynamic style, topology, and, at the same time, the possibility to efficiently analyze the behaviour of complex architectures. The expected overall result is thus nanoarray characterisation based on specific parameters related to a well defined tiled structure and not a set of figures of merit based on metamodels.

This paper is organized as follows: in section II the general features of the *multilevel simulator* are described, while in section III the fabric example (NASIC) used here to describe the simulator behavior is recalled. In section IV the specific simulator organization is reported and preliminary simulation results are commented in section V.

II. Multilevel simulator

Each nanotechnology fabric, currently undergoing intense research activity, has unique characteristics but, at the same time, share technological constraints, which are a main cause for unification of certain fundamental aspects. One of these is the requirement of a two-dimensional array as the underlying structure for computing [3], [7], [9].

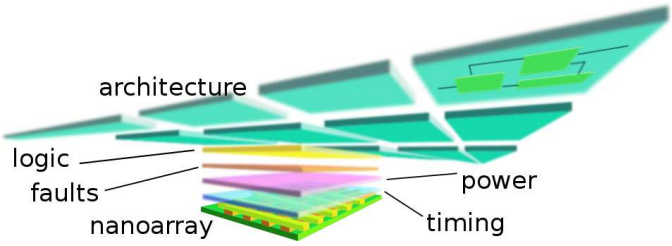

Fig. 1. Simulation levels.

On account of the aforementioned consideration, alongside the uncertainty over future developments of various technologies, one of the requirements for generality of the simulator being described rises. We considered appropriate to retain the opportunity of studying the behaviour of different nanofabric structures exploiting their common points. Therefore the question of how to cope with the particularities of each technology arises. Our approach has been to develop a way to describe the particularities of each technology, in spite of the common simulation structure, and put this information in technology-dependent libraries. Moreover, because the number of devices that would make future parallel architectures will be several orders of magnitude higher than the current, achieving a density as high as 10^{12} switches/cm^2[23], it seems appropriate to hierarchically decompose this study, taking into account manufacturing, fabric and device constraints.

We thus envision a simulator organized as in Figure 1, which, starting from the nanoarray tiled structure, associates to every performance figure a dedicated simulation layer, like for example timing, power, faults, logic behaviour..., if necessary interleaved when cross information are needed. This allows to lighten to the minimum the computational weight according to the requested outputs. Once the nanoarray is characterised an architectural simulation is feasible, so that logic organization, fault-tolerant techniques choices, parallelisation level trade-off can be explored as a function of technology, reliability, dynamic structure, fabric type, and so on.

We have chosen the NASIC fabric as a case of study as we deem it promising from both the manufacturability and the fault-tolerance point of view. A brief summary on its structure, functional to the following discussion only, is in section III. Our data structure, anyway, has on purpose been chosen general enough so that other two-dimensional nanoarray grids can be easily included and simulated. At the current stage of development, the simulator (described in section IV) is able to study the NASIC nanoarray from the logical point of view, to extract the switching activity and to explore reliability issues, according to a model proposed in literature [24], and gather reliability information on the design under analysis.

III. NASIC STRUCTURE

According to its proponents [1], the elemental units in NASIC are the *tiles* (see Figure 2). 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.

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 Nano Wires (NWs) or Carbon Nano Tubes (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 [1], with the latter that enables pipelining and overcomes the many limitations of a static design. An example of a NASIC tile (i.e. a specific type of "nanotile") for a two output AND function ($a \cdot b$, $\overline{a \cdot b}$) without complementary outputs is sketched in Figure 2. In this example we have chosen the NAND-NAND structure [21]. Each rectangle represents a nFET, with gate (G), source (S) and drain (D) connected to the nanowires. nFETs are organised into two logical planes, horizontal and vertical. Finally, micro-wires are used to carry power and control signals from the CMOS level.

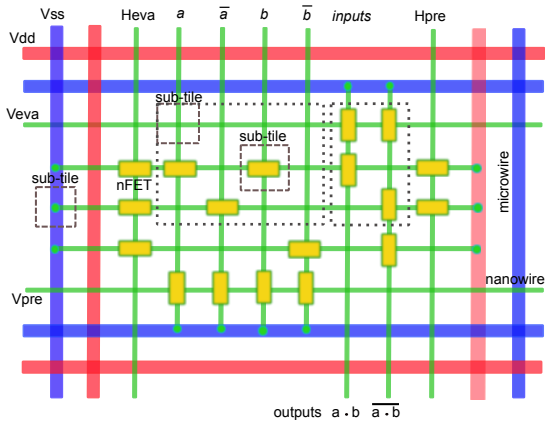

Fig. 2. NASIC tile and examples of subtiles. Two outputs AND: $a \cdot b$, $\overline{a \cdot b}$.

Dataflow in NASICs is through a 3-phase progression and the control signals from the CMOS level coordinate these phases. An example is given in Figure 3 [21], where a typical dynamic control style input sequence is shown. An evaluation phase for both the horizontal and vertical planes (Heva and Veva respectively) follows the necessary precharge step (Hpre and Vpre).

Faults are handled by masking them in the circuit and/or architecture design itself, implementing a multi-tiered built-in fault tolerance approach [24]. Simulations suggests that this built-in approach would be able to achieve 25-30% yield at 10% defect rate on a fabric grid implementing a simple processor [25].

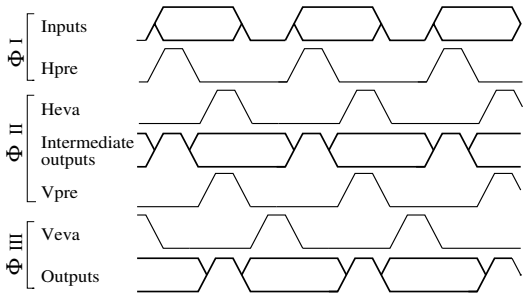

Fig. 3. Three phases dynamic control scheme.

IV. SIMULATOR STRUCTURE

The overall description of the basic simulator structure will be carried out with reference to the simplified flow diagram depicted in Figure 4: the representation has been organised into three parts (a, b, c) for the sake of clarity. Details on the algorithm are in subsection IV-E, while methodology criteria are described in subsection IV-A, IV-B, IV-C and IV-D.

A. Two-dimensional array data structure

One of the ideas behind our simulator is to look at the two-dimensional array, which is the main unifying aspect of many different technologies, and to ideally superimpose an identical array, staggered on both axes half the basic pitch of the grating array. In this way, we can highlight what we call *sub-tiles* (for some examples see details in Figure 2), as they are actual tiles, constituted by a vertical element, a horizontal element and one at the intersection of the previous two, denoted as the central element. We call these elements *components* and observe that a dynamic style NASIC circuit with n-type FETs only [21] can be described with a very small number of such components: microwire(s), nanowire(s), n-type FETs (with two orientations, horizontal and vertical) and contact(s).

Any sub-tile which is part of the nanotile can be obtained by placing one of the five components listed above in horizontal, vertical or central position. One of the reasons behind this structure choice is that we can describe a library of components for each technology we are interested in, and the components will realize the sub-tiles that will form the actual design. If we remove the overhead related to symmetries, due to rotations of the sub-tiles in the NASIC tile, we can greatly reduce the number of sub-tiles necessary to describe a complete NASIC design. Another reason is that with such a structure we are separating the technology of the components from the structure of the circuit we are willing to simulate.

Therefore, in this work sub-tiles are the elemental units of a design: as far as we know, no nanoarray simulators have been attempted at this level of detail. This view is particularly useful if one describes the sub-tiles in terms of information routing, that is to say one can describe the properties of the components, as well as the properties of the sub-tiles, in terms of their effect on the information they are routing. For instance, a nanowire in whatever sub-tile will propagate

information from one end to the other, irrespective of the technology of the nanowire itself. The very same can be said for microwires. A contact will propagate information cross the junction between the components it puts in contact. FETs will act as switches for the propagation of information, and they will act as a function of the information at the gate input. This kind of information is not related to the technology, but describes the behaviour of the components from an informational point of view.

B. Technological parameters

Technology will only come into play when technological dependent electrical parameters have to be considered: this does not impact the logic of information routing. Technological parameters for the components are stored in dedicated libraries, in the form of XML files.

We put routing information in an XML based description language to be able to simulate the propagation of information inside a nanoarray: we called it NEEDL (Nanoscale Electronics Easy Description Language). At present time it can operate at two levels: component level and sub-tile level, but it is being revised and expanded to support simulation at higher abstraction levels. The component level is the most informational, of course, but other levels can be useful in higher abstraction simulations.

C. Event driven logic simulation

The simulation is said to be event driven because 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, let's turn back for a while to the sub-tiles and their regular three-component structure. We can imagine each sub-tile as a four-port device, each identified by a cardinal point: N, S, W, E.

A change in the information at a port may need to be propagated inside the sub-tile, if there is the appropriate component to support propagation (e.g. a nanowire). 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. 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.

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 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.

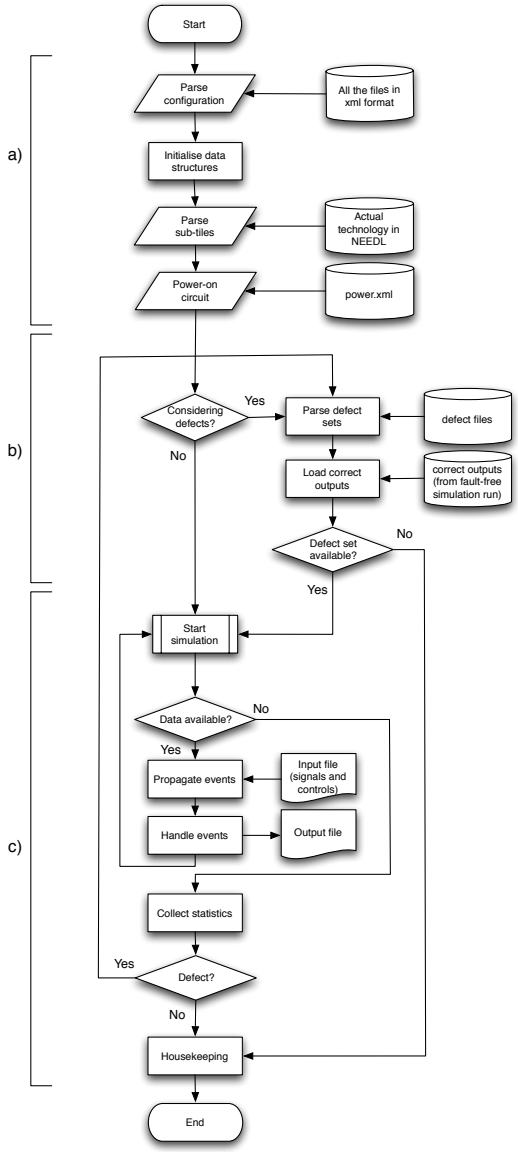

Fig. 4. Simplified flow diagram for the general simulation algorithm.

D. Defects analysis

There could also be defects inside the sub-tile, so the propagation of information process must be informed of this possibility. We separated the information about the component from its position inside the nanotile. In this way we are able to take into account local deviations of electrical parameters from their nominal values as a function of virtually any kind of probability distribution, hence we better describe real operating conditions. In section V details on the actual defect generation will be given.

Now it is time to take into account this kind of positional information. The simulator can be loaded with defect sets, that is to say directories of files describing the defects

and their location on the nanoarray. This information is loaded, in a sense, as a level of the base representation in computer memory of the nanoarray setting up a defect map (with reference to Figure 1). Every time information has to propagate inside a component that can be defective, the simulator checks with the current defect map to see if propagation can proceed or not, and this impacts event generation.

In this way we can get accurate data about the design under test that can be useful, for example, to characterise complex nanotiles and to support a more abstract simulation layer built on top of these results.

E. The algorithm

Now we are going to have a closer look at the general simulation algorithm referring to Figure 4: there is a risk of oversimplification in this diagram, but it is necessary to focus on key aspects.

- The simulation starts, as in Figure 4a), by parsing a number of files in XML format that hold the information for the simulation that is being run. At the very beginning, a configuration file that holds many flags (that can be changed inside the simulator) to drive the ongoing simulation (type of analysis performed, taking into account defects or not, etc.) is read, then a parse of some supporting information about the design follows. If the required analysis is not purely logical, the simulator will also parse the appropriate technological process file, holding information about the available components in the selected technology. For static timing analysis, FETs can be modeled by means of resistors of a given value (two, actually: R_{ON} and R_{OFF}): these values are technologically dependent.
  We then initialise basic data structures to carry out the required computations, as defined by the selected configuration. Components information in NEEDL format is loaded from the selected component set. A description of the basic sub-tiles that will embody the design (NASIC in this case) is loaded as well.
  Then we build an efficient memory representation of the nanofabric under test from a description file (or, possibly, many description files): this structure is lightweight from a computational point of view, because it only holds references to a very few number of sub-tiles actually instantiated in memory, hierarchically built in the previous steps from their basic components.
  Once the nanofabric has a representation in computer memory we have to power it up: for the sake of brevity, we will not go in details about this phase. The power.xml file describes the contact points between the nanofabric and power supply rails, to be as general as possible with respect to possible power issues. Power propagates into the circuit and generates events whenever necessary (e.g. when a nanowire is put in contact with a microwire by means of a contact component).
  This marks the beginning of the simulation, because the simulator starts propagating events and handling them, as in Figure 4c), but in quite different ways as a function of the kind of analysis required.
- In fact, Figure 4b) shows that the simulation branches here: if we are evaluating an ideal design without faults we can get directly to the next section of Figure 4c), while if we deal with defects we have to first parse defect sets and correct outputs of the circuit for the given input sequence, that was generated by the simulator itself in a prior faults-free run. A defect set is a file (then a structure in memory) that has been previously generated according to technology-related fault-probability distributions, providing information about which defects are present in the circuit under test. During a simulation, the tool will use thousands of such defect sets to characterise the circuit behaviour.
- Figure 4c) shows that as long as the simulator gets new data from input file(s) it will keep propagating and handling events in the structure as necessary, with the process above portrayed, and will collect information in the form of output file(s) and statistical analysis results.

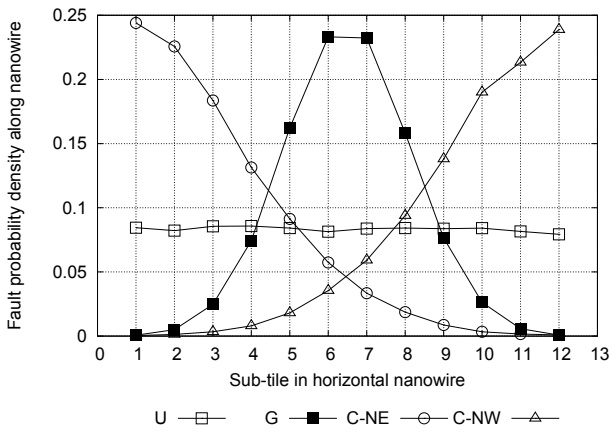

Fig. 5. Probability distribution of faults along tile nanowires. U: uniform distribution of faults in all the tiles; G: gaussian distribution of faults with mean value centered in tile; C-NE: gaussian distribution with mean value in North-East corner of the tile; C-NW: gaussian distribution with mean value in North-West corner of the tile

TABLE I
INPUTS AND OUTPUTS VALUES SIMULATED PHASES FOR AND DESIGN

Φ	Inputs								Outputs					
	Heva	a	$\bar{a}$	b	$\bar{b}$	Hpres	Veva	Vpre	Veva	Vpre	Heva	$a \cdot b$	$\bar{a} \cdot \bar{b}$	Hpre
I	0	**1**	0	**1**	0	1	0	0	1	1	1	1	1	1
II	1	1	0	1	0	0	0	1	1	1	1	1	1	0
III	0	1	0	1	0	0	1	0	1	0	0	**1**	**0**	1
I	0	**1**	0	**0**	1	1	0	0	0	1	1	1	1	1
II	1	1	0	0	1	0	0	1	1	1	1	1	1	0
III	0	1	0	0	1	0	1	0	1	0	0	**0**	**0**	1
I	0	**0**	1	**1**	0	1	0	0	0	1	1	1	1	1
II	1	0	1	1	0	0	0	1	1	1	1	1	1	0
III	0	0	1	1	0	0	1	0	1	0	0	**0**	**0**	1
I	0	**1**	0	**1**	0	1	0	0	1	1	1	1	1	1
II	1	1	0	1	0	0	0	1	1	1	1	1	1	0
III	0	1	0	1	0	0	1	0	1	0	0	**0**	**1**	1

TABLE II

INPUTS AND OUTPUTS VALUES SIMULATED PHASES FOR FA DESIGN.

DETAILS OF SWITCHING ACTIVITY ARE ON THE RIGHTMOST COLUMN.

Φ	Inputs										Outputs								Switching activity		
	Heva	a	$\overline{a}$	b	$\overline{b}$	c_i	$\overline{c_i}$	Hpre	Veva	Vpre	Veva	Vpre	Heva	$\overline{c_o}$	c_o	$\overline{s}$	s	Hpre	0→1	1→0	total
I	0	**0**	1	**0**	1	**0**	1	1	0	1	1	1	1	1	1	1	1	1	0	14	
II	1	0	1	0	1	0	1	0	0	1	1	1	1	1	1	1	1	0	9	25	57
III	0	0	1	0	1	0	1	0	1	0	1	0	0	**1**	**0**	**1**	**0**	1	9	0	
I	0	**0**	1	**0**	1	**1**	0	1	0	1	0	1	1	1	1	1	1	1	5	14	
II	1	0	1	0	1	1	0	0	0	1	1	1	1	1	1	1	1	0	9	9	46
III	0	0	1	0	1	1	0	0	1	0	1	0	0	**1**	**0**	**0**	**1**	1	9	0	
I	0	**0**	1	**1**	0	**0**	1	1	0	1	0	1	1	1	1	1	1	1	6	19	
II	1	0	1	1	0	0	1	0	0	1	1	1	1	1	1	1	1	0	9	9	52
III	0	0	1	1	0	0	1	0	1	0	1	0	0	**1**	**0**	**0**	**1**	1	9	0	
I	0	**0**	1	**1**	0	**1**	0	1	0	1	0	1	1	1	1	1	1	1	5	14	
II	1	0	1	1	0	1	0	0	0	1	1	1	1	1	1	1	1	0	9	9	46
III	0	0	1	1	0	1	0	0	1	0	1	0	0	**0**	**1**	**1**	**0**	1	9	0	
I	0	**1**	0	**0**	1	**0**	1	1	0	1	0	1	1	1	1	1	1	1	11	24	
II	1	1	0	0	1	0	1	0	0	1	1	1	1	1	1	1	1	0	9	9	64
III	0	1	0	0	1	0	1	0	1	0	1	0	0	**1**	**0**	**0**	**1**	1	11	0	
I	0	**1**	0	**0**	1	**1**	0	1	0	1	0	1	1	1	1	1	1	1	5	14	
II	1	1	0	0	1	1	0	0	0	1	1	1	1	1	1	1	1	0	9	11	50
III	0	1	0	0	1	1	0	0	1	0	1	0	0	**0**	**1**	**1**	**0**	1	11	0	
I	0	**1**	0	**1**	0	**0**	1	1	0	1	0	1	1	1	1	1	1	1	10	19	
II	1	1	0	1	0	0	1	0	0	1	1	1	1	1	1	1	1	0	9	11	60
III	0	1	0	1	0	0	1	0	1	0	1	0	0	**0**	**1**	**1**	**0**	1	11	0	
I	0	**1**	0	**1**	0	**1**	0	1	0	1	0	1	1	1	1	1	1	1	5	14	
II	1	1	0	1	0	1	0	0	0	1	1	1	1	1	1	1	1	0	9	11	50
III	0	1	0	1	0	1	0	0	1	0	1	0	0	**0**	**1**	**0**	**1**	1	11	0	

V. RESULTS

At the current stage of development our multilevel simulator has been tested on NASIC NAND-NAND nanoarrays for a few logic functions and simple designs. In this paper we report preliminary results for the two input AND port depicted in Figure 2 and for the 1-bit Full Adder.

The simulator has been developed in C++, with the support of a custom XML-based language for all the input and configuration information, and of Perl scripting for the generation of faults sets. All the software is written paying attention to portability issues. Different computing platforms are suitable: UNIX, Linux and Mac OS X are currently supported.

The event driven logic simulation described in section IV.C requires input and control signals: all the input combinations for the designs under test have been tested and initial (Φ I), intermediate (Φ II) and final (Φ III) digital values for the three-phase control scheme are reported in Table I and in Table II, where main inputs and outputs values are highlighted in bold. Control signals are evolved as required by the NASIC technology (as in Figure 3).

This simulation technique allows us to gather precise information about the switching activity that occurs inside the target design. In Table II we show the results of such a simulation for the FA. With reliable technological information about power consumption of both $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions we could accurately determine the power consumption of the simulated nanoarray with the chosen control style and technology.

The analysis of the impact of defects described in section IV.D is based on set of files which include fault information statistically generated in a Montecarlo style (2000 trials for each faults set). According to literature [9], NWs could break along their axes during the assembly process, hence defects could have a not uniform distribution along the wire length. To show the impact of different fault distributions in both wire and device on the output error rate, we performed multiple simulations. In each one, we associate a Gaussian (G) distribution or a Uniform (U) one, as reported in Figure 5. In particular, the Normal distributions where staggered, to simulate an increase in wire failure towards one of the corner (C-NE, C-SE, C-NW, C-SW), instead of failing with preference in the middle (G). Devices were supposed to fail in line with a Uniform distribution in each and every simulation run.

Chosen randomly under such distributions the defect point along the wire, we also randomly decide its presence: the maximum error rate is varied from 1% to 20% [25].

In Figure 5 we show a 2D and 3D representation of defects distribution in the case of a 20% error rate. Top left and top right show a Uniform and Gaussian distribution respectively, while in bottom left an example of faults condensed in the North-West corner is shown. On the bottom right we have a 3D view contrasting the bottom left case at 20% and 5% error rate.

The output error rate for the AND port (for the four input combinations in Table I) due to faults distributed along both horizontal and vertical nanowires according to the above-mentioned statistics is in Figure 7.left. Output error rate shows a linear increase with nanowire defect rate with a unexpectedly high ratio: 2% of nanowire defects cause a $\approx 28\%$ output error rate, whilst 10% causes a $\approx 40\%$ to $\approx 50\%$ as output error probability, depending on distribution type. In the C-NW and C-NE cases, an early failure is propagated throughout the whole structure, leading thus to a greater output error rate.

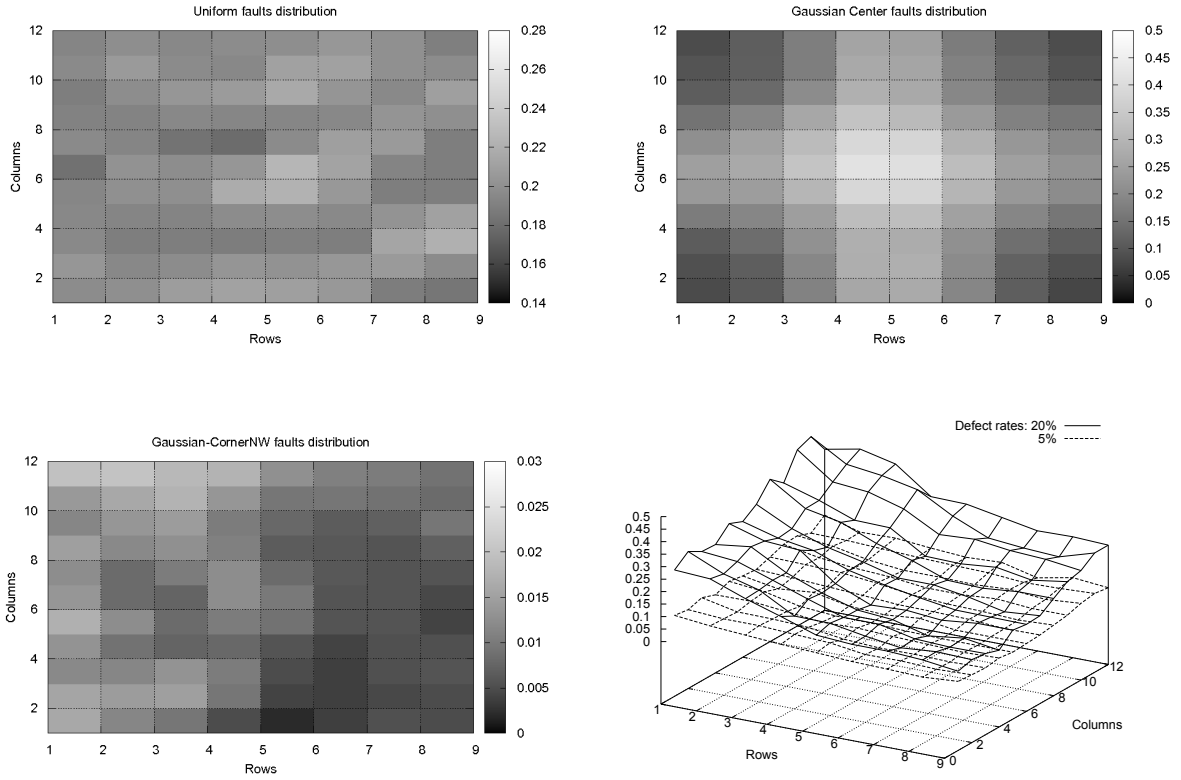

Fig. 6. Top left and top right: uniform and gaussian distribution respectively; bottom left: an example of faults condensed in the North-West corner; bottom right: 3D view contrasting the bottom left case at 20% and 5% error rate.

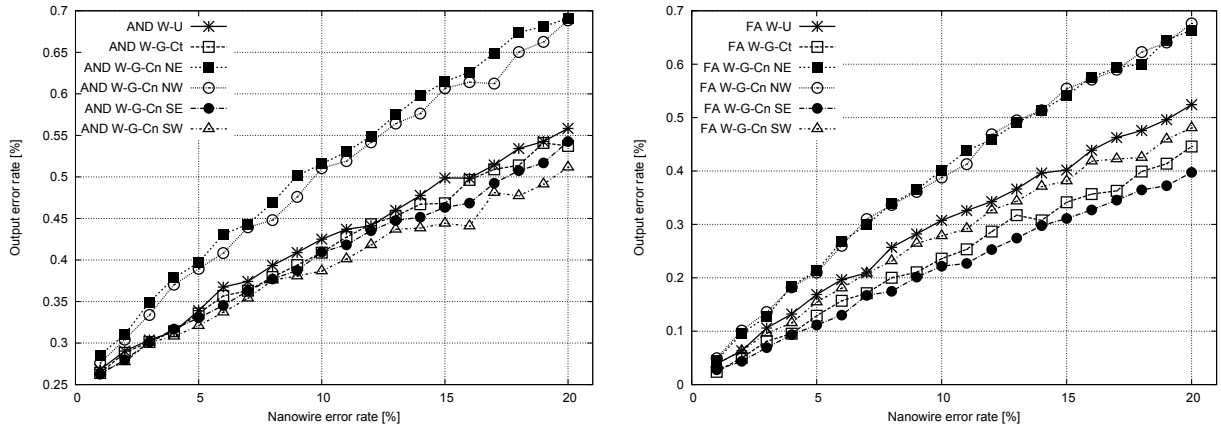

Fig. 7. Output error rate for AND port (left) and Full Adder (right) as a function of nanowire defect rate.

In Figure 7.right the output error rate is shown (for the eight input combinations in Table II) under the same conditions in the 1-bit Full Adder (FA) case. It shows a linear increase with nanowire defect rate, but with a lower ratio ($\approx 5\%$ for 2% of nanowire defects) at the lowest values of defect rate. Even if they start at quite different output error rates for low wire defect rates, both the AND port and the FA reach an $\approx 70\%$ output error rate for a 20% nanowire defect rate in the worst case. It is worth noticing that the span from worst-to-best case is about 30% for the latter design. The different behaviour of the AND port and FA is even more evident in Figure 8, where their output error rates are superposed when both nanowire and device defects are taken into account (20% defect rate with uniform distribution for devices). The impact of device error rate is of 5% in the worst case, thus reflecting its lower importance with respect to nanowire failure which influences larger parts of the design.

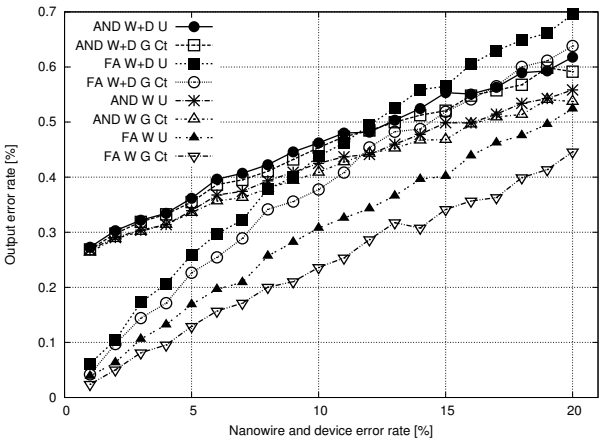

Fig. 8. Output error rate for both AND port and FA when device and nanowire faults are superposed.

The ability to handle defects distributions, not just a defect rate, is an important distinguishing factor from literature. Given a fault-rate we have shown there is a remarkable difference in the output failure depending on the actual distribution of the faults in the structure and we can precisely evaluate the effect of whatever such distribution. Therefore, we can also asses the effectiveness of fault tolerance techniques proposed in literature: this will be included in our future work points.

These results show how much important is to have a simulator which details the behaviour at sub-tile level, able thus to extract performance and topological data (in this case related to faults) useful for both higher architectural level simulation steps and synthesis and physical design tools.

VI. Conclusion

Our simulator can perform mid-to-low level simulations, taking into account the inner statistical nature of different type of nanofabrics, with a general approach that can virtually be applied to any parameter of interest affecting the structure.

Provided that information about probability distributions of many technological aspects is drawn from actual experiments, the simulator can supply quite accurate results. It can also be used with data set based on speculative models, of course, with the limits incidental to this kind of approach. In the former case, the produced results could be used at higher simulation levels as a strong foundation.

At present time, we can perform logical analysis of array-based nanostructures and thoroughly investigate the impact of both defects and fault-tolerance techniques, and devise accurate switching activity information. In particular, since we focused on NASIC designs, we developed all the supporting libraries for this architecture.

Much work has still to be done in order to expand the feature set of the simulator, to exploit the potential of the proposed simulation architecture: as a next step, we plan to take advantage of this event-driven variability-aware simulation technique to analyse delay.

References

[1] 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.

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

[3] 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.

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

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

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

[14] 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.

[15] 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.

[16] 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

[17] 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.

[18] D. Bhaduri and S. Shukla, *NANOLAB: A Tool for Evaluating Reliability of Defect-Tolerant Nano Architectures*, in "VLSI, IEEE Computer Society Annual Symposium on", IEEE Computer Society, p.25, Los Alamitos, CA, USA, 2004.

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

[20] C. He, M. F. Jacome, *RAS-NANO: a reliability-aware synthesis framework for reconfigurable nanofabrics*, in "DATE '06: Proceedings of the conference on Design, automation and test in Europe", European Design and Automation Association, pp. 1179–1184, Munich, Germany, 2006.

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

[22] T. Wang et al., *Self-healing wire-streaming processor on 2-D semiconductor nanowire fabrics*, in "NSTI/Nanotech 2006 Conf.", Boston, MA, May 2006.

[23] T. Rueckes et al., *Carbon Nanotube-Based Nonvolatile Random Access Memory for Molecular Computing*, Science, vol. 289, n. 94, 2000.

[24] C. A. Moritz et al., *Towards Defect-Tolerant Nanoscale Architectures*, in "Nanotechnology, 2006. IEEE-NANO 2006. Sixth IEEE Conference on", vol. 1, pp. 331–334, 2006.

[25] T. Wang et al., *NASICs: A Nanoscale Fabric for Nanoscale Microprocessors*, in "IEEE International Nanoelectronics Conference (INEC)", 2008.

[26] P. Narayanan et al. , *Validating Cascading of Crossbar Circuits with an Integrated Device-Circuit Exploration*, in "IEEE/ACM Symposium on Nanoscale Architectures (NanoArch'09)", July 2009.