Improving the Fault Tolerance of Nanometric PLA Designs

Federico Angiolini[1], M. Haykel Ben Jamaa[2], David Atienza[2,3], Luca Benini[1], and Giovanni De Micheli[2]

[1]DEIS, University of Bologna, 40136 Bologna, Italy.

[2]LSI, EPFL, 1015 Lausanne, Switzerland.

[3]DACYA, Complutense University, 28040 Madrid, Spain.

ABSTRACT

Several alternative building blocks have been proposed to replace planar transistors, among which a prominent spot belongs to nanometric filaments such as *Silicon NanoWires (SiNWs)* and *Carbon NanoTubes (CNTs)*. However, chips leveraging these nanoscale structures are expected to be affected by a large amount of manufacturing faults, way beyond what chip architects have learned to counter. In this paper, we show a design flow, based on software mapping algorithms, to improve the yield of nanometric *Programmable Logic Arrays (PLAs)*. While further improvements to the manufacturing technology will be needed to make these devices fully usable, our flow can significantly shrink the gap between current and desired yield levels. Also, our approach does not need post-fabrication functional analysis and mapping, therefore dramatically cutting on verification costs. We check PLA yields by means of an accurate analyzer after Monte Carlo fault injection. We show that, compared to a baseline policy of wire replication, we achieve equal or better yields (8% over a set of designs) depending on the underlying defect assumptions.

1. INTRODUCTION

The end of the road for silicon CMOS scaling has already been forecast many times. Until today, however, chip makers have always found ways to solve or work around most new technological issues. Semiconductor roadmaps show that, at least for the next five to ten years, technological breakthroughs will still allow for further evolutionary improvements [1].

Yet, the sheer number of critical issues indicates that CMOS miniaturization may become too expensive to continue at the current pace. Increasing leakage worsens power consumption and prevents voltage scaling, each subsequent submicron lithography node requires more complex and expensive masks, tunneling effects forbid the arbitrary trimming of gate insulators, and the ever stronger spreading of device parameters is going to bring uncertain yield and reliability trends as a tradeoff for the manufacturing of faster components [31].

Among the emerging technologies, one of the most promising employs filaments of micrometric length and nanometric width, the so-called nanowires. Two main flavours of nanowires can currently be manufactured, the *Silicon NanoWires (SiNWs)* and *Carbon NanoTubes (CNTs)*. While their underlying physics are different, they share the ability to be variably doped, to resistively conduct current, to form p-n-junctions and to allow field-effect control [6].

Although this observation leads to the idea of using these devices as basic blocks for computation, making a chip with nanowires is still quite far from becoming reality. A major hurdle is the immature manufacturing technology. For the foreseeable future, it seems that laying nanowires on a chip floorplan in almost arbitrary patterns will be impossible. Instead, regular patterns of parallel wires will have to be created, for example within the so-called *Programmable Logic Array (PLA)* architectures. Computation can be made possible by overlapping two layers of wires in perpendicular fashion, which results in grids of programmable crosspoints.

Even this relatively simple architecture is expected to be quite unreliable. As a matter of fact, the current consensus seems to be that 10% to 15% of the elementary devices may be defective [4]; this figure is orders of magnitude larger than what silicon chip designers have learned to tackle. Hence, traditional fault tolerance approaches are not adequate [28]. Even though it is almost certain that further technological improvements will be needed to allow the manufacturing of reliable logic, research at the architectural and design levels might help to more quickly close the feasibility gap.

This paper brings two main contributions. First, we describe a design methodology for nanometric PLAs which is more tolerant to high percentages of faults. Our approach is fully software-based, and could be integrated in a standard mapping *Computer Aided Design (CAD)* flow. It is based on the generation of redundant covers for the target function; the covers are also optimized according to a yield-aware criterion, *i.e.* the minimization of the number of PLA crosspoints. We compare the resulting yields to those achieved by hardware-only solutions, such as wire duplication [32, 33]. To accurately assess the yield, and as a second novel contribution, we also present a PLA analyzer that can inject a parameterizable amount of faults in the augmented architecture with a Monte Carlo approach, and subsequently assess the functionality of the PLA.

2. RELATED WORK

In the search for a replacement of the MOS transistor, several technologies have been proposed. Among them are SiNWs, for which growth is discussed in [17] and which can be doped either longitudinally [16] or radially [24]. They can be organized on the substrate either by self-assembly in Langmuir films [2] or by direct on-chip patterning by means of the multispacer patterning technique [3]. Junction (diode) and transistor behaviour of crossed SiNWs is proved in [6], and Boolean logic is built in [21]. A whole SiNW crossbar is presented in [34]. The crossbar layout implies a grid of crosspoints, that may be individually programmed and used for computation or storage. A mixed approach [4] suggests to sandwich a layer of amphiphilic molecules among the two perpendicular planes of nanowires; the molecules are bi-stable, and can be configured in a state which is held for months. About 85% crosspoint yield is reported.

The availability of nanowire crossbars as building blocks enables the exploration of the architectural design space. Several designs have been proposed, aiming either at implementing memories or logic circuits. An architecture called NanoFabrics [15] proposes the interleaving of computation planes with nanometric latches. A two-plane NOR/OR PLA is discussed in [8], relying upon imprinted decoders [5] for wire selection. An architecture with cascaded sequences of OR/NOT planes is also proposed by the same authors in [9].

One of the biggest problems associated with nanometric PLAs is yield, which is expected to be low due to the presence of multiple faults in the assembly of such small circuit elements. Fault occurrences are faced in [36], which strives to find the optimum size of a nanowire crossbar to achieve the maximum amount of functional crosspoints. On the other hand, [20] explores open and bridging faults.

Most proposed approaches to fault tolerance in nanometric chips feature redundancy at the logic block level [18]. However, these approaches seem to scale only up to about the 1% fault threshold for logic blocks; any fault rate greater than that will reduce overall yields to almost zero. Reconfiguration-based approaches [26], deriving from the Teramac [19] experience, assume the availability of an unrealistic amount of interconnection resources. A novel approach to majority voting [29] tolerates high fault rates while still providing good yields, but only as long as the circuit under test is extremely small (a few transistors), and in the assumption of ideal voters.

In some proposed approaches [27, 30], any individual chip is tested for functionality of single crosspoints, and the functional mapping is subsequently performed on the known-good elements. In [27], an area overhead of just 8% is sufficient to achieve a successful mapping on a device with 15% of faulty crosspoints. However, the assumption here is that testing equipment for chip-by-chip verification and functionality mapping will be available at a low cost, which is not true at present.

Recent research has been focusing on strategies to improve the yields of nanometric PLAs, while not requiring the large overhead of a post-manufacturing mapping flow. For example, in [32, 33] the authors propose a technique where hardware-level wire redundancy is applied to alleviate yield problems. Unfortunately, this approach still falls short of representing a full solution. Even despite the unrealistic assumption of ideal decoding logic to access the nanometric PLA, and by applying nanometric-scale yield improvement techniques in combination with traditional Triple Modular Redundancy at the micrometric scale (which implies large area penalties), the yields claimed by the authors are still not compatible with industrial manufacturing.

As these works prove, the problem of yield in nanometric technologies is challenging. The contribution we bring is to propose a strategy which works at the design flow level, *i.e.* during PLA logic mapping; this choice enables several optimizations, some of which are also accessible to the hardware designer (redundancy), but some of which are not (generating PLA mappings which are yield-aware). Our approach is therefore partially overlapping and partially complementary to hardware-level techniques. In the following, we will compare the results of these two choices as far as they overlap, and assess the incremental improvements which are only enabled by our suggested approach.

3. REFERENCE ARCHITECTURE FOR A NANOMETRIC PLA

As previously mentioned, nanowire-based PLAs are implemented by means of nanometric crossbars, where "horizontal" and "vertical" wires define a grid of crosspoints. Crosspoints are programmable at runtime by applying a high voltage among the two wires. Depending on the polarity of the voltage, the crosspoint can be programmed *off* or *on*, that is, disconnected (no interaction among the horizontal and vertical planes) or connected. In the latter case, the exact functionality of the crosspoint depends on the specific nanowires used. For example, SiNWs can be doped in the same way bulk silicon is; by crossing a p-type SiNW and an n-type SiNW, an "on" crosspoint will act as a junction (a diode). Transistor behaviour can be achieved in a similar fashion, just by radially coating one or both nanowires with silicon oxide.

One of the toughest problems to face in nanowire arrays is decoding, *i.e.* the process of selecting one of the crosspoints of the grid. In memories, this is clearly needed for normal operation. In PLAs, the decoding process only needs to be performed once, just after manufacturing, to map the desired logic function onto the chip (programming of the PLA). Two main decoder architectures have been proposed where few microscale wires act as address lines to drive a much larger number of nanoscale wires [8, 14]. The decoder is an extremely critical piece of hardware. In order to moderate the dramatic impact of a fault on wire functionality, k-hot encodings (requiring more address wires) have been deployed [10].

For the present paper, we assume a reference architecture as shown in Figure 1, almost directly derived from [8]. This PLA has a NOR input plane, where crosspoints act as FETs, and an OR output plane, where crosspoints act as diodes. The NOR/OR combination of planes is functionally complete, *i.e.* allows any logic function to be implemented. The presence of FETs in the input plane guarantees that, when cascading multiple PLAs, signal restoration is provided (a cascade of diode planes would be completely passive and would quickly degrade logic values). Decoders control access

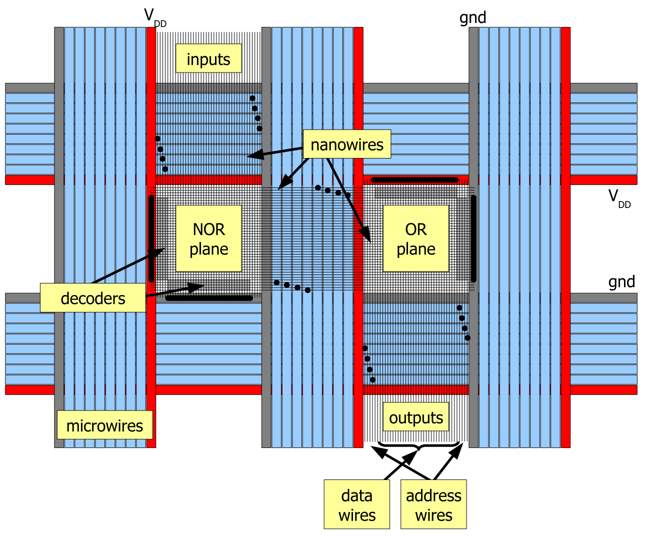


Figure 1: The NOR/OR PLA reference architecture

to both planes. The PLA of Figure 1 can be considered as a stackable tile within a larger design, and to this extent the orientation of its planes might be changed in the actual chip floorplan.

A key point to stress here is that the possible implementations of a nanometric PLA are also dependent on the actual manufacturing techniques available to the designer. Just as an example, nanowires can be grown off-chip and then deposed stochastically, *e.g.* by fluidic techniques [34], or patterned directly on-site [3]. This has an impact on the amount of manufacturing faults. Given the early stage of the research in this field, it is difficult to assume a fixed manufacturing technology. Therefore, while modeling the defects, we leave some flexibility in terms of how the PLA blocks are built, *i.e.* we assume different defect scenarios depending on the underlying technology (this will be reflected in Section 5.2).

4. INCREASING THE YIELD OF NANOMETRIC PLAS

The yield of nanometric PLAs can be improved in several ways: (i) architectures can be augmented at the hardware level (often by using redundant structures [32, 33]); (ii) faults can be accepted, detected and worked around by using the functional parts of the chip [27, 30, 11]; (iii) yield-aware design tools can be conceived. We focus on the third strategy. We would like to stress that, since the technology is still immature and the device defect rates are several orders of magnitude higher than the defect rates of usual CMOS devices, the resulting yields are not directly targeted at industrial scale manufacturing. The described methods ought to be applied on technologically improved devices, and combined with the other defect tolerance techniques mentioned above, in order to manufacture chips with admissible yields.

Our work is based on two main ideas. The first one is a variant of the well-known redundancy principle; yields can be improved by replicating PLA structures. In our case, instead of replicating the hardware devices themselves, we achieve the same effect by working on the mapping of the target function onto the PLA, and by devising a redundant cover of the function. The second idea is to optimize the mapping algorithm, so that the generated cover is minimally sensitive to manufacturing faults; namely, we minimize the amount of crosspoints which must be functional for proper PLA operation. It is interesting to notice that these approaches may actually also be useful to improve the reliability of the PLA in case of transient or permanent failures after the manufacturing stage.

4.1 Redundant Covering Approach

The common flow of logic circuit mapping can be summarized as follows. After the definition of the logic function by means of a list of minterms belonging to the *ON*, *OFF* and *Don't Care (DC)* sets, the function is minimized in order to cover it with the smallest number of implicants (called primes here). Since each prime is represented by a nanowire in the physical circuit, minimizing the function saves area and improves yield (less nanostructures are needed and wires are shorter). The formal problem of covering a function *f* at minimal cost can be expressed by the following linear problem: minimize $|\mathbf{x}|$ so that:

$$\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{1} \tag{1}$$

where $\mathbf{A}$ is the prime implicant table listing all possible primes and the minterms they cover and $\mathbf{x}$ is a binary vector expressing which primes create the cover [25]. To solve this problem, digital designers use logic minimizers such as Espresso [25].

Then, this problem can be generalized into a form which considers redundancy. The task is to minimize $|\mathbf{x}|$ so that:

$$\mathbf{A} \cdot \mathbf{x} \geqslant \mathbf{n} \tag{2}$$

where all entries of $\mathbf{n}$ are identical arbitrary natural numbers. This redundant covering of *f* has no impact on the functionality of the circuit, but might improve its reliability [22, 23]. Such linear problems are represented in their most general formulation as a set of linear inequations with a constraint (maximum or minimum) on a multilinear form of the variables, and can be solved by an *Integer Linear Programming (ILP)* solver. We utilize the `GLPK-4.9` [13] tool, which uses the numerical procedure called *simplex method* [7].

The ILP solver operates on the set of linear inequations in Eq. 2, therefore covering the function in a redundant way, and is asked to minimize a cost function which can be formulated differently in order to explore the cover under different conditions:

- Minimize the number of primes $|\mathbf{x}|$ (*i.e.* nanowires), which results in the minimal solution for a given redundancy level,
- As above, and then, between all possible minimal solutions, also minimize (maximize) the number of literals in the cover (*i.e.* molecular switches).

Both formulations provide minimal PLA covers. The first criterion returns simply a cover insuring the lowest number of nanowires, thus the smallest PLA size; whereas the second criterion gives two covers representing two PLAs with the same minimal size, but having the highest and the lowest number of crosspoints, respectively.

As an example, for a given function *f* (see Table 1), the impact of both covering criteria on the circuit is illustrated in Figure 2.

	A′·B	A′·B′	A·B′	A·B
C′·D	1	-	-	1
C′·D′	1	-	0	-
C·D	-	1	-	1
C·D′	-	0	-	0

Table 1: Karnaugh map of an example function *f*

4.2 Redundant Covering and Wire-Level Replication

The redundant covering approach has a roughly equivalent hardware implementation, close to what suggested in [32, 33], *i.e.* the addition of redundant wires. The two approaches would achieve identical results if every product term of the function were to be simply replicated; in mathematical terms, this would mean solving the inequalities of Eq. 2 with the same vector $|\mathbf{x}|$ that solves the inequalities in Eq. 1, just replicated twice. However, this does not guarantee that the solution is minimal with respect to $|\mathbf{x}|$. In fact, a better solution can be achieved by exploiting the *cyclic core* of the function, thus producing a redundant cover of the function minterms with less than twice the product terms.

To compare our approach to a reference result, we also implement a plain wire duplication scheme as suggested in [32]. Evidently, both wire replication and redundant function covering have an area overhead, and the resulting addition of hardware structures

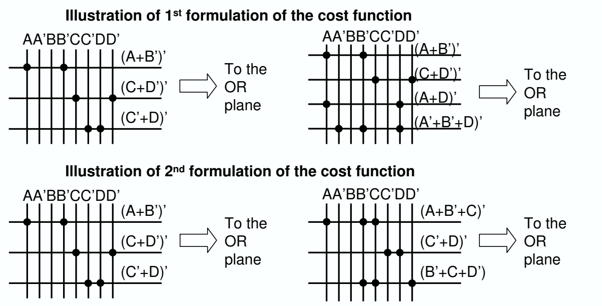


Figure 2: Relationship among cost functions for the covers and the resulting physical circuits. Upper half: minimal cover (left) *vs.* non-minimal cover (right). Lower half: minimal number of literals (left) *vs.* non-minimal number of literals

increases the number of components that may fail. This will be further discussed in Section 6.

On the other hand, the minimization of the number of crosspoints needed for PLA functionality has no direct hardware equivalent, and to the best of our knowledge is a completely novel contribution of this work.

5. ANALYZING NANOMETRIC PLAS

While the yield analysis of PLA devices might be performed by mathematical means [36], such approaches are not easy to extend to take into account features such as redundant functional covers, number of required functional crosspoints or hardware-level wire redundancy. Therefore, we have developed a PLA analyzer. The tool is able to operate both on PLAs which have been mapped with our approach, *i.e.* with redundant covers, or to add redundancy under the form of replicated wires at the hardware level [33]. The tool requires two main inputs; one is a PLA description in the standard `.pla` file format, the second is a configuration file. The latter contains extra hardware specifications (such as the desired amount of redundancy at the wire level), physical constraints (such as the width of nanometric and micrometric wires) and fault rate estimations.

Depending on the analysis that we want to perform, we feed our tool with different `.pla` files, as will be discussed in Section 6. Subsequently, our analyzer operates by mapping the `.pla` file onto a model of the underlying hardware. The physical PLA is represented in detail, including assigning decoder patterns to select each wire of the array; the decoder is assumed to use a k-hot encoding, where k is a parameter. An area figure is computed. Based upon the values provided in the configuration file, faults are injected in the PLA model (see Section 5.1) and the PLA is analyzed. The output of the analysis is a report of the actual amount of faults present in the PLA, of the addressable and functional crosspoints, *etc.*. The report includes a detailed printout of the PLA, where each fault can be visually located. A configurable amount of analyses can be run in a loop, guaranteeing statistically accurate results. The reports are processed by scripts to automatically generate result spreadsheets. The complete flow we have developed is depicted in Figure 3. We are not aware of any other analyzer for the fault tolerance of nanometric PLAs with the same configurability and flexibility.

5.1 Monte Carlo Fault Injection

To provide a detailed assessment, we assume that four types of faults can manifest themselves in the nanometric structure, as described in [36]:

- Bad ohmic contacts among nanowires and microwires (with probability P_{bco})
- Faults in the decoder patterns (with probability P_{fde})
- Disconnected crosspoints instead of FETs or diodes, both in the PLA planes and in the decoders (with probability P_{dcr})
- Broken nanowires (with probability P_{bnw} per unit length)

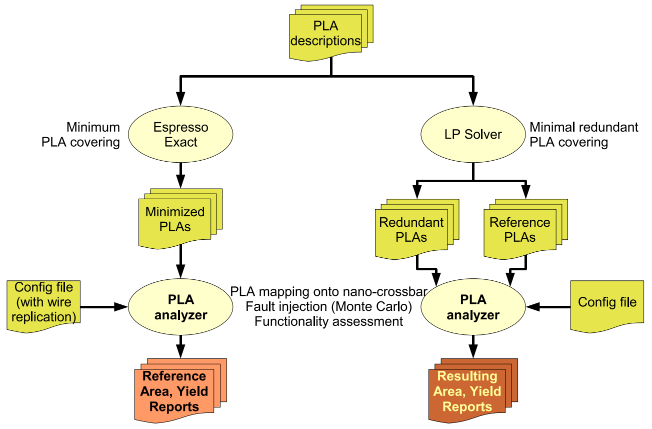

PLA descriptions
Minimum PLA covering
Espresso Exact
LP Solver
Minimal redundant PLA covering
Minimized PLAs
Redundant PLAs
Reference PLAs
Config file (with wire replication)
PLA analyzer
PLA mapping onto nano-crossbar
Fault injection (Monte Carlo)
Functionality assessment
PLA analyzer
Config file
Reference Area, Yield Reports
Resulting Area, Yield Reports

Figure 3: The analysis flow implemented in this paper

Scenario	worst	avg	best	wirebest	spt
P_{bco}	0.0005	0.00035	0.0002	0.0005	0.0000001
P_{fde}	0.0005	0.00035	0.0002	0.0005	0.0000001
P_{dcr}	0.0005	0.00035	0.0002	0.0005	0.03
P_{bnw}	0.0001	0.00005	0.00001	0.000001	0.0000001

Table 2: Fault rate scenarios for our analysis

Another fault category is that of bridged crosspoints. We simplify our analysis by merging it with the nanowire breakages; this is because bridged crosspoints [20] and broken nanowires have a similar effect on the PLA, corrupting the functionality of a whole row or column. For this same reason, designers normally tune the manufacturing process so that bridging faults are much less likely to occur than stuck open ones [8]. Therefore, in the following, we will mention "crosspoint failures" only when talking about disconnected crosspoints, consistently with [27].

We inject these faults in the PLA according to a Monte Carlo methodology. To assess the functionality of the whole device, we propagate faults to any downstream logic that may be affected. For example, as said above, broken wires render whole rows or columns of the PLA unaccessible and hence unusable.

When associating hardware faults to their functional effect, it is possible to notice different correlations. For example, a missing crosspoint in the output (OR) plane causes an output function to not be affected by a product term; this reduces the ON set of the function, but can be recovered if a redundant product term is available and its connection to the output is functional. On the other hand, a missing crosspoint in the input (NOR) plane causes a product term to not be affected by an input; this enlarges the ON set of the function. This fault can also be recovered if a redundant input is provided, and its crosspoint is functional. Our analyzer is capable of automatically verifying such occurrences, marking the PLA as failed only when redundancy is not enough to compensate for manufacturing faults.

5.2 PLA Fault Scenarios

As previously discussed, given the early state of research on nanotechnologies, it is almost impossible to identify well-characterized manufacturing processes and the associated fault models, and it is still difficult to quantify numerical values for their occurrences. We first of all borrow the ranges suggested in [36] (see Table 2). The three scenarios `worst`, `avg` and `best` bracket those ranges. We then add a fourth scenario `wirebest`, which postulates noticeable improvements in the manufacturing of nanowires, but otherwise assumes most parameters to be worst-case. Finally, with `spt` we try to model a system where nanowires can be produced with techniques such as the *Spacer Patterning Technique (SPT)* on the substrate [3]. SPT is already common in manufacturing FinFETs and was extended to PLAs; it would drastically improve the reliability of nanowires and their connection to the outer CMOS circuitry [3]. On the other hand, SPT-based crossbars would need molecular switches at the crosspoints, and it is not yet clear how reliably these could be manufactured. Therefore, we assume in this latter scenario a much higher occurrence of crosspoint faults, equal to 3%.

It is key to understand the effects of even the smallest fault ratios. For example, a single fault occurring in a decoder can render tens or hundreds of crosspoints non-addressable, non-programmable, and therefore non-functioning. Address wires and their contacts to the external microwires are some of the most critical structures, since a fault cascades to the decoders, leading to severely malfunctioning planes. In this perspective, P_{dcr} may even bring the most negligible contribution to PLA failures, unless its value is very high such as in the `spt` case. Previous literature reports 15% of the elementary devices to be defective [4]; this value is however what can be measured when electrically testing a chip, and is therefore a measure of the overall malfunctions of the crosspoints (including non-programmability) rather than of P_{dcr} in isolation. We will show that, after compounding the effects of all faults, the total quantity of unusable crosspoints in the PLA can in fact easily amount to 10-20%.

6. EXPERIMENTAL RESULTS

To assess the usefulness of our proposed methodology, we test it against a set of PLA designs taken from the Microelectronics Center of North Carolina suite [35]. For each sample, we generate both a redundant cover (initially according to the first criterion only, see Section 4.1) and a minimum cover. The latter is created with Espresso Exact, an exact logic minimizer. Our analyzer maps both onto hardware models of the PLAs, applying wire redundancy to the minimum cover to simulate the hardware-level wire replication approach. In the wire replication case, input, output and product wires can enjoy redundancy, while our redundant covering approach is at this stage only working on product terms; to get a fair comparison, we manually instruct our analyzer to provide the same level of replication for the inputs and outputs. Therefore, the two approaches are functionally equivalent and feature equal redundancy levels, but their physical implementation is different.

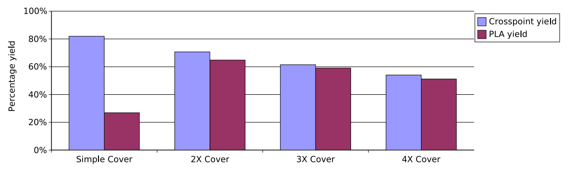
We configure the PLA analyzer with the fault rates assumed in Section 5.2 and these physical constraints: the nanowire width W_{nw}, inclusive of nanowire pitch, is assumed to be 10 nm, while the microwire width W_{mw} is set to 210 nm [12] (see Figure 1).

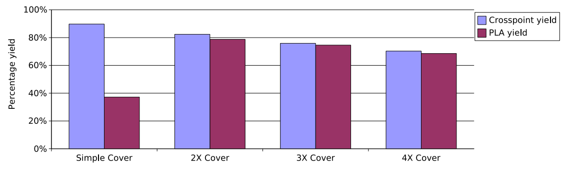
We then inject random faults into the hardware models, analyze each circuit for correct functionality, repeat 3000 times, and derive the yield. We define a PLA to yield if at least one of the redundant copies of each output correctly maps the function. We define a crosspoint of the PLA to yield if it is fault-free and *addressable*, that is, if the micro- and nano-structures required to access it for programming and reading are functional. With 3000 experiments, the 95% confidence interval half-width is $1.96 \cdot \sqrt{0.5^2/3000} \approx 1.8\%$, *i.e.* there is a 95% likelihood that the actual yield does not differ by more than 1.8% from the estimated yield.

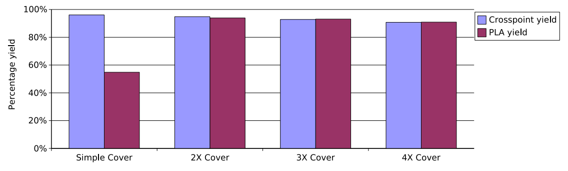
6.1 Yield and Area for Varying Redundancy Levels

In Figure 4, we plot the average yield, over all the PLA suite, of mappings with and without added redundancy. In presence of such high amounts of malfunctioning elementary devices, it comes as no surprise that the yield of plain PLAs is poor; even under comparatively mild fault scenarios (`best`), the yield is about 55%. Under `worst` case conditions, the yield goes down to 27%. Double covering has an extremely beneficial effect, with yields going up to the range 70%-96%. Adding more redundancy in the cover, however, does not usually improve yields; since this implies larger PLAs, the likelihood of experiencing broken nanowires increases steeply, affecting overall results. This is especially critical for vertical (input and output) nanowires, since our benchmark PLAs typically feature more products than inputs and outputs, leading to very unbalanced rectangular floorplans. To verify this result, we test the `wirebest` fault configuration, where the nanowire yield is assumed to be drastically improved, even though the fault rates of other components are kept pessimistic. In this scenario, the PLA yield increases when adding further redundancy up to 3X, after which the extra breakages more than compensate for the added replication.

We then check these results against a baseline represented by a hardware-based wire replication approach. We first of all notice


(a) `worst`


(b) `avg`


(c) `best`

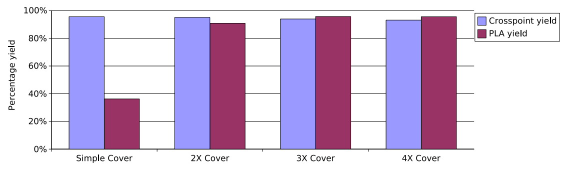

(d) `wirebest`

Figure 4: Yield as a function of the cover redundancy for four different fault rates)

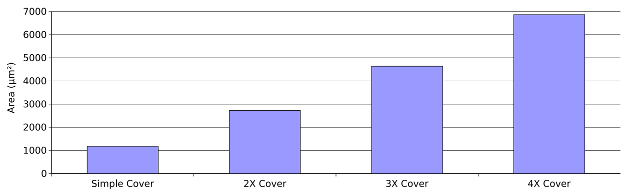


Figure 5: Area occupation for the whole suite under different replication strategies

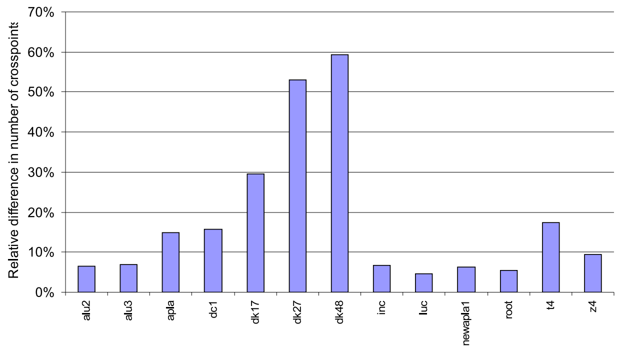


Figure 6: Relative difference in number of crosspoints between the best and the worst minimal cover

that the total number of primes in the n-redundant cover is, for all studied samples, circa n times larger than the number of primes in the exact minimum cover. This is due to the properties of the cyclic core of the functions of our benchmark suite; better results could be achieved on functions with larger cyclic cores. We simulate the yield of the physically replicated PLAs, and we observe it to be on average just slightly lower than that of n-redundant covers, but always within the 95% confidence interval of the latter. The two techniques yield about equally due to the fact that the same amount of redundancy is present, and, since the number of nanowires is almost the same in both PLAs, even the defects whose probability depends upon the PLA size (wire breakages) manifest themselves in a similar way. Since the two approaches yield very similarly under the fault assumptions of Figure 4, we only report the results for the redundant covering approach.

Figure 5 reports the area requirements of the whole PLA suite for these levels of redundancy. Given that wires are being added both horizontally and vertically, a quadratic slope under increasing covering levels could be expected; however, as can be seen, the trend is just a bit steeper than linear. This can be explained by the fact that a large slice of the area is taken by decoders, addressing microwires and power rails, all of which do not scale, or scale weakly, with added redundancy. While increased fault tolerance still incurs an overhead, the area of the redundant nanometric circuits is expected not to exceed that of the equivalent CMOS circuits, since nanometric PLAs are expected to achieve one to two orders of magnitude higher density than even 22 nm CMOS [9].

Seeing the results of Figures 4 and 5, under our fault models, duplication seems to be representing a good design choice from the yield and area points of view, and will therefore be used for the following experiments.

6.2 Mapping Flow with Crosspoint Minimization

The results discussed above (Figure 4) show that a software-based approach to PLA mapping yields at least as well as a wire replication one. However, these results are under scenarios that emphasize the fragility of nanowires. The `spt` scenario, on the other hand, is based on a process for manufacturing reliable nanowires, in which the molecular switches at the crosspoints are likely to be the most critical part of the PLA. This type of faults is clearly largely sensitive to the amount of crosspoints which are needed to be functional for proper PLA operation. Our logic mapping approach can be tuned to provide higher yields in this scenario, by minimizing the number of connected crosspoints required by the mapping. This is an example of support for yield-aware features which is unavailable to the hardware designer.

We generate double covers featuring the minimum and the maximum possible number of crosspoints (according to the second criterion, see Section 4.1). Figure 6 reports the relative differences between these two covers for a few PLAs; a carefully crafted cover can cut by up to 60% the amount of crosspoints which must be functional for the PLA to yield. For most of the functions available to us, the reduction is between 0% and 20%.

We test these alternate covers and a regular wire duplication policy with our analyzer under different fault models (Figure 7). Under `worst`, `avg`, `best` and `wirebest`, the global yield is relatively high (for nanowire standards), and the yield difference between the mappings is not significant. This is because, under these scenarios, faults tend to cluster. For example, a broken nanowire causes many crosspoint failures (no crosspoints can be accessed), but all of them are aligned; a redundant wire can correct all of them simultaneously. There is still a significant percentage of PLAs that fail, *e.g.* due to even more catastrophic (and almost impossible to fully correct) events, such as decoder problems, where a whole region of the PLA becomes inaccessible.

The `spt` scenario exhibits a very different behaviour. In this case, the support structures are expected to be more reliable, while a larger amount of crosspoints themselves may fail. These crosspoint failures are very likely to be evenly spread across the whole PLA, and it is much more difficult to recover from them even with redundancy; for this reason, yields are much lower. A crosspoint-aware mapping policy can however improve mapping results by a significant amount. On average, our policy improves yield by 10%

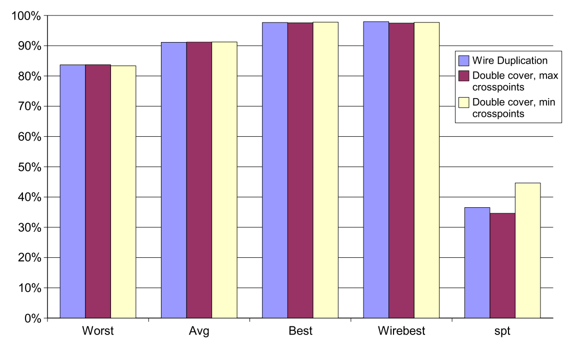


Figure 7: Yield of minimal covers with different amounts of crosspoints under different fault models

compared to the minimal double cover with more crosspoints, and by 8% compared to the baseline wire duplication policy. The improvement can be achieved thanks to the requirement for a smaller number of functional crosspoints, and therefore depends on the specific function at hand; more specifically, it depends on the amount of minimal covers which are available and on how much they differ in terms of literals (crosspoints). The maximum observed gains (not depicted) are 18% and 16%, respectively, for the `dk48` PLA. These results also show that plain wire replication for a generic given input function is not likely to produce optimal results, since the function is not normally tuned for the minimum number of literals; in fact, Figure 7 shows that, for our benchmark suite, wire duplication yields almost as bad as minimal covers which are tuned on purpose for the maximum number of crosspoints.

If compounded to hardware-level techniques for yield boosting, our results can help in making SPT architectures more viable. No area penalty is paid compared to any other duplication strategy, since the cover is anyway minimal.

7. CONCLUSIONS

We have presented an approach to improve the fault tolerance of nanometric PLAs. Contrary to previously proposed techniques, it does not need chip-by-chip, crosspoint-by-crosspoint verification prior to the mapping of the logical functions, therefore cutting on post-manufacturing costs. When testing over a suite of benchmark PLAs, injecting faults in a Monte Carlo fashion, our analyses show yields of up to 96%. The redundant covering behaves at least as well as a plain wire replication policy, while boosting the yields by 8% on average for technologies where the dominant faults are crosspoint-related. Specific parameters, such as nanowire breakages, are shown to be more critical than others for a successful yield improvement. The area overhead required by our methodology is less than quadratic on the added redundancy, and comparable to that of hardware-based replication. While the obtained yields are still way below the usual values for current CMOS processes, this is mainly due to the immaturity of the underlying technology, and the gap can be significantly reduced with our technique.

Future work includes studying our approach in combination with complementary methods by other researchers and under additional fault models, as the technological development will demand.

8. ACKNOWLEDGMENTS

This work is partially supported by STMicroelectronics, the Swiss FNS Research Grant 20021-109450/1 and the Spanish Government Research Grant TIN2005-5619.

9. REFERENCES

[1] International technology roadmap for semiconductors (ITRS) 2001. Technical report, International Technology Roadmap for Semiconductors, 2001.

[2] Christopher L. Brown, Ulrich Jonas, Jon A. Preece, Helmut Ringsdorf, Markus Seitz, and J. Fraser Stoddart. Introduction of [2]catenanes into langmuir films and langmuir-blodgett multilayers. a possible strategy for molecular information storage. *American Chemical Society*, 16(4):1924–1930, 1999.

[3] G.F. Cerofolini, G. Arena, M. Camarelli, C. Galati, S. Reina, L. Renna, D. Mascolo, and V. Nosik. Strategies for nanoelectronics. *Microelectronic Engineering*, 81:405–419, 2005.

[4] Yong Chen, Gun-Young Jung, Douglas A. A. Ohlberg, Xuema Li, Duncan R. Stewart, Jan O. Jeppesen, Kent A. Nielsen, J. Fraser Stoddart, and R. Stanley Williams. Nanoscale molecular-switch crossbar circuits. *Nanotechnology*, 14:462–468, 2003.

[5] S. Y. Chou, P. R. Krauss, W. Zhang, L. Guo, and L. Zhang. Sub-10 nm imprint lithography and applications. *Journal of Vacuum Science and Technology B*, 15(6):2897–2904, 1997.

[6] Yi Cui and Charles M. Lieber. Functional nanoscale electronic devices assembled using silicon nanowire building blocks. *Science*, 291:851–853, 2001.

[7] George B. Dantzig, Alex Orden, and Philip Wolfe. The generalized simplex method for minimizing a linear form under linear inequality restraints. *Journal of Mathematics*, 5(2):183–195, 1955.

[8] A. DeHon. Array-based architecture for FET-Based, nanoscale electronics. *IEEE Transactions on Nanotechnology*, 2(1):23–32, 2003.

[9] A. DeHon. Design of programmable interconnect for sublithographic programmable logic arrays. In *Proceedings of the International Symposium on Field-Programmable Gate Arrays (FPGA)*, pages 127–137, 2005.

[10] A. DeHon, Patrick Lincoln, , and John E. Savage. Stochastic assembly of sublithographic nanoscale interfaces. *IEEE Transactions on Nanotechnology*, 2(3):165–174, 2003.

[11] A. DeHon and H. Naeimi. Seven strategies for tolerating highly defective fabrication. *IEEE Design & Test of Computers*, 22(4):306–315, 2005.

[12] A. DeHon and Michael J. Wilson. Nanowire-based sublithographic programmable logic arrays. In *Proceedings of the International Symposium on Field-Programmable Gate Arrays (FPGA)*, pages 123–132, 2004.

[13] http://www.gnu.org/software/glpk/.

[14] B. Gojman, E. Rachlin, and J.E. Savage. Decoding of stochastically assembled nanoarrays. In *Proceedings of the IEEE Computer society Annual Symposium on VLSI*, pages 11–18, 2004.

[15] Seth Copen Goldstein and Mihai Budiu. NanoFabrics: spatial computing using molecular electronics. In *Proceedings of the 28th Annual International Symposium on Computer Architecture*, pages 178–189, 2001.

[16] Mark S. Gudiksen, Lincoln J. Lauhon, Jianfang Wang, David C. Smith, and Charles M. Lieber. Growth of nanowire superlattice structures for nanoscale photonics and electronics. *Nature*, 415:617–620, 2002.

[17] M.S. Gudiksen, J. Wang, and C.M. Lieber. Synthetic control of the diameter and length of single crystal semiconductor nanowires. *Journal of Physical Chemistry B*, 105:4062–4064, 2001.

[18] Jie Han, J. Gao, P. Jonker, Yan Qi, and J.A.B. Fortes. Toward hardware-redundant, fault-tolerant logic for nanoelectronics. *IEEE Design & Test of Computers*, 22(4):328–339, 2005.

[19] James R. Heath, Philip J. Kuekes, Gregory S. Snider, and R. Stanley Williams. A defect tolerant computer architecture: Opportunities for nanotechnology. *Science*, 280:1716–1721, 1998.

[20] Jing Huang, M.B. Tahoori, and Fabrizio Lombardi. On the defect tolerance of nano-scale two-dimensional crossbars. In *19th IEEE International Symposium on Defect and Fault Tolerance (DFT) in VLSI Systems*, pages 96–104, 2004.

[21] Yu Huang, Xiangfeng Duan, Yi Cui, Lincoln J. Lauhon, Kyoung-Ha Kim, and Charles M. Lieber. Logic gates and computation from assembled nanowire building blocks. *Science*, 294:1313–1317, 2001.

[22] Parag Lala. *Fault Tolerant & Fault Testable Hardware Design*. Prentice Hall International, 1985.

[23] Parag Lala. *Self-Checking and Fault-Tolerant Digital Design*. Morgan Kaufmann Publishers, 2001.

[24] Lincoln J. Lauhon, Mark S. Gudiksen, Deli Wang, and Charles M. Lieber. Epitaxial coreshell and coremultishell nanowire heterostructures. *Nature*, 420:57–61, 2002.

[25] Giovanni De Micheli. *Synthesis and optimization of digital circuits*. McGraw-Hill, 1994.

[26] M. Mishra and S.C. Goldstein. Defect tolerance at the end of the roadmap. In *Proceedings of the International Test Conference*, pages 1201 – 1210, 2003.

[27] H. Naeimi and A. DeHon. A greedy algorithm for tolerating defective crosspoints in nanoPLA design. In *Proceedings of the IEEE International Conference on Field-Programmable Technology*, pages 49–56, 2004.

[28] K. Nikolic, A. Sadek, and M. Forshaw. Architectures for reliable computing with unreliable nanodevices. In *Proceedings of the 1st IEEE Conference on Nanotechnology (IEEE-NANO)*, pages 254–259, 2001.

[29] A. Schmid and Yusuf Leblebici. Robust circuit and system design methodologies for nanometer-scale devices and single-electron transistors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(11):1156 – 1166, 2004.

[30] Greg Snider, Philip Kuekes, and R. Stanley Williams. CMOS-like logic in defective, nanoscale crossbars. *Nanotechnology*, 15:881–891, 2004.

[31] Jayanth Srinivasan, Sarita V. Adve, Pradip Bose, and Jude A. Rivers. Lifetime reliability: Toward an architectural solution. *IEEE Micro*, 25(3):70–80, 2005.

[32] Teng Wang, Mahmoud Bennaser, Yao Guo, and Csaba Andras Moritz. Combining circuit level and system level techniques for defect-tolerant nanoscale architectures. In *Proceedings of the 2nd IEEE International Workshop on Defect and Fault Tolerant Nanoscale Architectures (NanoArch)*, 2006.

[33] Teng Wang, Mahmoud Bennaser, Yao Guo, and Csaba Andras Moritz. Self-healing wire-streaming processors on 2-D semiconductor nanowire fabrics. In *Proceedings of the NSTI (Nano Science and Technology Institute) Nanotechnology Conference 2006*, pages 312 – 315, 2006.

[34] Dongmok Whang, Song Jin, Yue Wu, and Charles M. Lieber. Large-scale hierarchical organization of nanowire arrays for integrated nanosystems. *Nano Letters*, 3(9):1255–1259, 2003.

[35] Saeyang Yang. Logic synthesis and optimization benchmarks user guide version 3.0. Technical report, Microelectronics Center of North Carolina, 2001.

[36] Shanrui Zhang, Minsu Choi, and Nophill Park. Modeling yield of carbon-nanotube/silicon-nanowire FET-based nanoarray architecture with h-hot addressing scheme. In *19th IEEE International Symposium on Defect and Fault Tolerance (DFT) in VLSI Systems*, pages 356–364, 2004.