Yield Enhancement of Asynchronous Logic Circuits through 3-Dimensional Integration Technology

Abstract

This paper presents a systematic design for yield enhancement of asynchronous logic circuits using 3-D (3-Dimensional) integration technology. In this design, the target asynchronous circuits on one planar device layer which is fabricated with aggressive technology, are built on fault tolerant graph models with extra spare resources, and can be reconfigured by autonomous reconfiguration logic on another planar device layer which is fabricated with conservative technology, in the presence of hard errors. The yield analysis shows that this method can result in 20–30% overall yield enhancement. This method can be conveniently applied to clocked designs without significant changes.

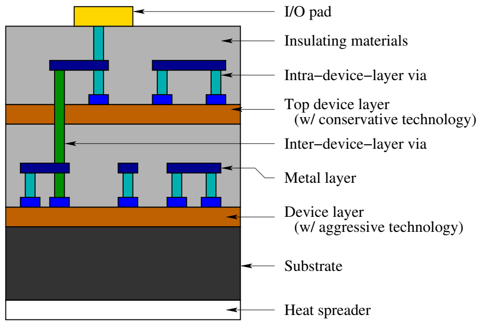


Figure 1. Three-dimensional integrated circuits with multiple planar device layers.

1 Introduction

Aggressive technology scaling results in continuously increasing process complexity and wafer handling cost, making it more difficult and expensive to achieve high fabrication yield by identifying and eliminating most defects [13]. Due to increased time-to-market pressures, however, foundries are often forced to start volume fabrication on a given technology before the fabrication process becomes completely mature. Hence, to optimize the circuit design for better yield and to shorten yield learning stage have become important issues to semiconductor industry.

With the challenges facing conventional device scaling, 3-D (3-Dimensional) integration technology allows scaling to continue by shifting the focus from device scaling to circuit scaling. In 3-D integrated circuits (IC), planar device layers are stacked, one on top of another in a 3-D structure where adjacent device planes can be connected by short, vertical vias [2]. Using 3-D integration, the designer can construct VLSI systems that exhibit lower interconnect latencies; higher packing densities of circuits; and heterogeneous integration of devices of different materials (such as SiGe and III-V materials) or technologies (such as submicron and nanometer) [2]. During the fabrication of 3-D structure, each planar device layer can be manufactured and tested separately. All layers are then stacked and assembled together with inter-layer interconnects. Figure 1 shows the diagram of a 3-D structure with two device layers.

Higher clock frequency and decreased feature sizes by rentless technology scaling present a growing challenge to global clock distribution, making it difficult and expensive to design a singly-clocked, globally synchronous VLSI system. On the other hand, the absence of clock makes asynchronous circuits not suffer such problem, and their expected performance is decided by average-case (instead of worst-case) path delay. Besides, it is easy for an asynchronous system to achieve high energy efficiency because computations are purely data-driven and no dynamic energy is consumed for an idle component. All these facts make asynchronous design become an increasingly practical alternative [4]. Future VLSI systems could utilize both types of circuits: clocked logic for local computations facilitates circuit design and asynchronous logic for global computations alleviates clock distribution and timing headaches.

Previous work on circuit design for yield is primarily based on hardwired N-modular redundancy (NMR) [6, 12] or fully programmable logic/gate arrays [1, 7, 8]. However, it is non-trivial to apply hardwired NMR method to asynchronous circuits without significant timing assumptions [11] because the local handshake in asynchronous circuits makes it unclear when the not-directly-communicated outputs are expected to match, making the duplication-and-comparison philosophy ineffective. For programmable logic/gate arrays, the large number of configuration bits significantly slow down the circuit, and explicit fault diagnosis effort required for manual reconfiguration, considerably increases product test cost.

In this paper, we propose a new design method for yield enhancement of asynchronous circuits. By utilizing 3-D integration technology and adding self-reconfiguration logic onto a separate reliable device layer, the VLSI system can recover from hard errors automatically and results in less product failure probability. Unlike NMR method, no significant timing assumption has to be made, making this design suited for asynchronous circuits. Moreover, partial redundancy of this design leads to smaller silicon area, helping reducing hard errors. Compared with fully programmable arrays, only a limited number of configuration bits are added in this design, largely reducing performance and wiring overheads. Besides, self-healing behavior spares fault diagnosis of target circuits, resulting in less cost of product testing. Section 2 presents that design in detail. Section 3 evaluates the potential of yield improvement. Section 4 draws the conclusions.

2 Defect Tolerant Asynchronous Design with 3-D Integration Technology

A systematic way to build an defect tolerant system is to make each module defect tolerant. In this case, not only design complexity can be largely reduced but also smaller hardware cost is required by adding less redundancy. Figure 2 shows the framework of our design.

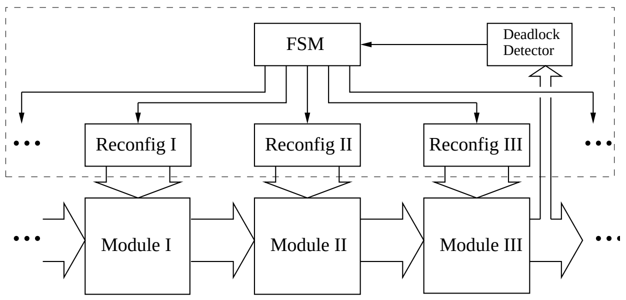


Figure 2. Reconfigurable defect-tolerant asynchronous system.

In Figure 2, self-checking logic is augmented to each hardware module so that circuit deadlocks when it is faulty. Each model is built on a fault tolerant (FT) graph with extra spare resources. Pass-gates whose control inputs (configuration bits) come from the reconfiguration logic, are applied to all inputs/outputs and necessary internal wires of each VLSI module so that the circuit topology can be changed dynamically. Whenever any error occurs in a module, this module deadlocks due to the fail-stop logic. Because of handshake-based data communication, any deadlocked module will cause the whole asynchronous system to stall. Thus, only one deadlock detector (implemented as a delay line of a current-starved inverter chain [10]) is required to detect the stall and trigger self-reconfiguration logic which reconfigures the modules by replacing the faulty sub-modules with the spare workable ones. After reconfiguration completes, computation restarts from either the beginning or the last architectural checkpoint.

Due to the absence of comparison procedure, no significant timing assumption is required, making this design suited for asynchronous logic. Unlike fully-programmable arrays, reconfiguration is implemented at coarse granularity (module-level instead of gate-level), resulting in much less configuration overhead. Self-reconfiguration spares fault diagnosis and manual reprogramming, significantly reducing product test cost. In addition, the extra reconfiguration circuitry does not increase packaging expense (which is the significant part of overall manufacturing cost) because no additional I/O pin is introduced. The following subsections further explain this design framework.

2.1 Implementation with 3-D IC technology

In Figure 2, all circuitry (self-reconfiguration and deadlock detection) in the dashed box are critical (error-sensitive) and must be made highly reliable. Otherwise, the system cannot be reconfigured in the presence of errors. With 3-D integration technology, the system can be implemented as follows. (i) All the error-sensitive transistors are placed onto one planar device layer (shown as the top layer in Figure 1) which is fabricated with conservative technology (such as micrometer or submicron technology with very high reliability). (ii) The target VLSI circuits (the VLSI modules which are outside the dashed box) are placed onto other planar device layers which are fabricated with aggressive technology (such as deep-submicron or nanometer technology with low reliability).

The 3-D implementation has the following advantages. (1) By fabricating error-sensitive transistors with reliable technology, this 3D layout makes those critical circuitry unlikely affected by hard errors. (2) Inter-device-layer vias largely reduces wiring overhead between reconfiguration logic and target circuit, causing less planar silicon area. (3) Since the error-sensitive circuitry stand by when the system is not faulty, using reliable technology causes less leakage current and further reduces overall power overhead of the defect tolerant design, while without noticeably hurting system performance.

2.2 Fail-stop behavior and fault tolerant graph models

A widely-used asynchronous circuit template, precharge half-buffer (PCHB) [9], is chosen for asynchronous logic implementation. A PCHB circuit can have multiple inputs and outputs, and it can be used to construct almost any asynchronous logic. For example, an asynchronous MiniMIPS microprocessor uses PCHBs for more than 90% of its circuits. Similar to a precharge domino circuit in synchronous design, a PCHB circuit performs computations using pull-down (NMOS) networks, making it fast and compact. In this circuit, each input/output variable X is usually dual-rail encoded (X^0, X^1) with an explicit acknowledge (X^e). Validity and neutrality of the variables are checked and synchronized, generating the common acknowledge to all inputs as well as the precharge/enable signal for computation of current stage. Further details can be found in [9].

Due to the multi-rail encoded data and explicit handshake-based event-ordering, the authors in [3] proved that any failure by a single stuck-at fault in the PCHB circuit either deadlocks the circuit or produces an illegally encoded output (X^0X^1=‘11’). Thus, fail-stop behavior can be implemented by adding a checker (NAND gate) onto each output of a PCHB circuit which blocks the incoming handshake signals in the presence of any illegal output.

In order to achieve reconfiguration, each VLSI module is built on a fault tolerant graph with extra spare resources so that faulty components can be replaced with workable spare ones. In [11], we developed a K-fault tolerant linear array (called *min-spare array*) by adding minimum (K) nodes and necessary redundant internal connections between the nodes. The left graph in Figure 3 shows a construction example of 2-FT 2-node linear array, where 2 spare nodes (shaded), 4 external edges (dashed), and 5 internal edges (bold) are added for any 2-node fault tolerance. As to VLSI modules which consist of identical components, they generally can be modeled as a linear array (e.g., full adder) or a collection of linear arrays (e.g., multiplier, FIR filter), given that either data or control propagates linearly through them. The collection of linear arrays usually can be further simplified into a linear array through coalescing all nodes of the same row (column). For all these VLSI modules, min-spare array model can be reasonably applied.

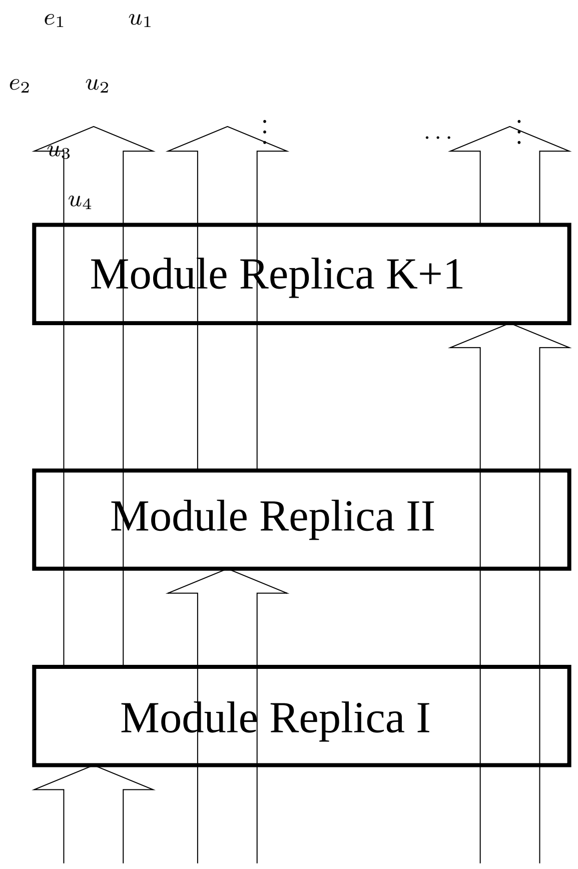
For VLSI modules of other topologies, the K-fault tolerant graph can be simply ($K + 1$) replicas of the target module (called *full-duplication graph*, shown as the right graph in Figure 3). One and only one replica is selected by enabling its connections with the outside. Compared with NMR method, only K (instead of $2K$) full replicas are added to this graph. Other efficient fault tolerant graph models can be applied to further reduce hardware cost.

2.3 Self-reconfiguration

Reconfiguration in Figure 2 is both dynamic and autonomous so that no manual fault diagnosis and external reprogramming is required, significantly reducing product test cost. For asynchronous circuits, on-the-fly fault location is non-trivial and generally results in large hardware overhead, compromising the overall reliability by exposing more transistors to unreliable environment. In this paper, we propose a new method: self-reconfiguration is achieved by searching a workable configuration out of all possible ones, without fault location. To implement exhaustive search, finite state machines are utilized to remember the current configuration and decide the next configuration to take. Reconfiguration is activated repeatedly until a workable setting is found. The following explains further details.

Reconfiguration. The reconfiguration can be implemented using synchronous logic which is triggered by time-out signals from deadlock detector(s). Conservative timing can be applied to guarantee circuit functionality.

Let an asynchronous system consist of M modules and K defects are to be tolerated. Each module is built on a fault tolerant graph. Whenever hard error(s) occur, the system stalls (due to fail-stop) and the deadlock detector acti-



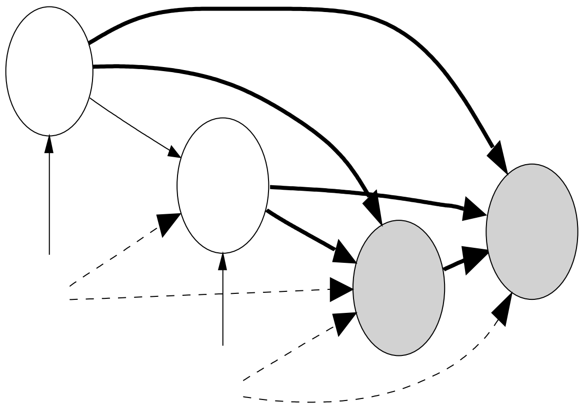


Figure 3. Fault tolerant graphs.

vates the top finite state machine (shown as block *FSM* in Figure 2) of self-reconfiguration logic. Block *FSM* selects K modules and reconfigures those modules concurrently by activating the per-module reconfiguration circuitry (shown as blocks *Reconfig I$\cdots$III*). Each per-module reconfiguration logic includes a finite-state machine which is used to search a workable configuration from all possible configurations of current VLSI module.

Block *FSM* has M outputs, and exactly K of them can be valid during a reconfiguration so that exactly K modules are activated for concurrent reconfigurations. Thus, the core of block *FSM* is a $\lceil log_2\binom{M}{K}\rceil$-bit cyclic counter which is used to select all possible K modules, together with necessary combinational logic to derive the M outputs according to current counter output. For per-module reconfiguration logic, it depends on the fault tolerant graph topology utilized in the VLSI module. For a VLSI module of full-duplication graph, the per-module reconfiguration logic is simply a $(K+1)$-bit one-hot counter (a cyclic shift register with unique bit-'1') for switching to a different replica. For a VLSI module of min-spare array, per-module reconfiguration is completed by selecting N nodes out of $N+K$ nodes (suppose a N-node array). Hence a dedicated $\lceil log_2\binom{N+K}{N}\rceil$-bit (when $K \ll N$, $M \simeq Klog_2N$) cyclic counter is utilized to search all possible $\binom{N+K}{N}$ configurations. All configuration bits (which correspond to an embedded isomorphic array with specific N nodes) are then derived from current counter output, through supporting combinational logic. Figure 4 shows an construction example of self-reconfiguration logic for 1-FT 2-module system.

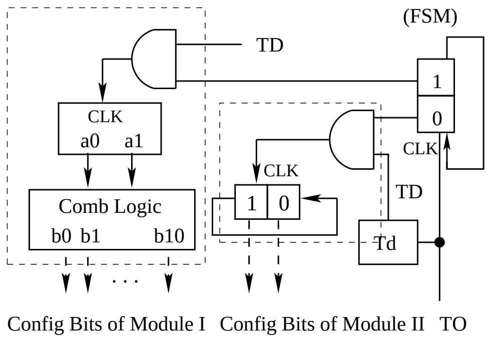


Figure 4. Reconfiguration logic.

In Figure 4, Block FSM is a 2-bit one hot counter which enables 1 module for reconfiguration. Per-module reconfiguration circuitry are shown in two dashed boxes. Module I is built on a 1-FT 3-node min-spare array: 2-bit counter searches all 4 configurations, and combinational logic derives all configuration bits (for 11 edges) accordingly. Module II is built on a 1-FT full-duplication graph (composed of 2 full replicas): a 2-bit one-hot counter chooses the corresponding replica by enabling its connections with external environment. Primary input *TO* comes from deadlock detector, and it becomes high when system deadlocks, activating reconfiguration circuitry. *TO* is delayed so that FSM becomes updated before per-module reconfiguration starts. At the end of each reconfiguration, *TO* will be reset to low.

Defect recovery time. Defect recovery time is decided by the number of configurations the system has tried before it finds a workable one. Although concurrent per-module reconfigurations might unnecessarily reconfigure non-faulty modules, they significantly reduce defect recovery time.

Let μ_d be the total time required by one reconfiguration. Depending on the system complexity, μ_d is generally in terms of microseconds or milliseconds. The worst defect recovery time T_r, which can be used to estimate the expected defect recovery time, is that system has tried all possible configurations and only the last one is workable. In this hierarchical reconfiguration framework, we have $T_r = \sum_{i=1}^{P}\tau_i$ where $P = \binom{M}{K}$ and τ_i is the longest worst defect recovery time of current selected K modules. If that module is built on full-duplication graph, $\tau_i = (K+1)\mu_d$; If that module is of min-spare N-node array, $\tau_i = \binom{N+K}{N}\mu_d$.

Defect recovery time is reduced with smaller M and N (given K). M is reduced by partitioning VLSI system at coarser granularity, and N becomes less by dividing linear array(s) into larger nodes. During product testing, the defect recovery (test-and-reconfiguration) time can be minutes or hours. Hence, the total configurations of this defect tolerant design can be up to hundreds of thousands or even millions (if μ_d is in terms of milliseconds). Thus, this self-recovery method can be conveniently applied to large VLSI designs.

3 Evaluation

In this section, we investigate yield enhancement of the defect tolerant design with 3-D implementation. One problem for traditional yield modeling is that model parameters are specific to fabrication process and layout, and they are generally hard to be reasonably estimated [6]. In order to make general conclusions, we examine the yield of the defect tolerant design based on the assumed yields of baseline circuit, instead of physical parameters such as critical area and fault density. Let Y_o be the yield of baseline circuit, and Y_{ft} be the yield of K-defect tolerant circuit. To make the conclusions independent on fault clustering factors, we assume faults to occur independently. Thus, Y_{ft} is pessimistic and the yield enhancement results are conservative.

For the defect-tolerant circuit of min-spare array topology (suppose it is a N-node array and Y_n is the survival rate of each node), the yield is the probability that at least N out of $N+K$ nodes survive.

$$Y_{ft} = \sum_{i=N}^{N+K}\binom{N+K}{i}(Y_n)^i(1-Y_n)^{N+K-i} \qquad (1)$$

Because faults occur independently, $Y_n = \sqrt[N]{Y_o}$.

For the defect-tolerant circuit of full-duplication graph topology, the yield is the probability that at least 1 out of $K+1$ replicas is workable.

$$Y_{ft} = \sum_{i=1}^{K+1}\binom{K+1}{i}(Y_o)^i(1-Y_o)^{K+1-i} \qquad (2)$$

By choosing defect tolerance at coarse granularity (e.g., large module and array node sizes), we can easily make the overhead of pass gates and extra wiring in fault tolerant graphs negligible so that the extra area of vias between configuration logic and target circuits can be reasonably omitted. Thus, the calculated yields using equations (1) and (2) are good approximations to the real results. Next we extend the yield calculations to 3-D structure.

For the baseline (non-FT) circuit of 3-D IC with L layers, we assume that all device layers are fabricated with the same technology and the yield of each layer is the same (Y_o). Thus, the overall yield can be calculated as

$$Y_o^{3D} = (Y_o)^L \tag{3}$$

For the defect-tolerant circuit of 3-D IC, an extra layer (called *configuration layer*) is added for reconfiguration (and deadlock detection) for the target circuitry on the other L layers (called *target layers*). For simplicity, we assume the yield of each target layer is the same (Y_{ft}). We also assume all circuits on target layers are of uniform fault tolerant graph topology (either min-spare array or full-duplication graph) and investigate the corresponding yield enhancement. The yield of a general defect tolerant design with hybrid graph topologies, can be estimated as some intermediate between these two extremes. The overall primitive yield of defect-tolerant design is $Y_{cfg}(Y_{ft})^L$. Due to the extra configuration layer, however, some silicon dies which are supposed to be used for target circuitry are allocated for reconfiguration logic. To count such silicon cost penalty, the overall (effective) yield of defect-tolerant circuit is calculated as follows,

$$Y_{ft}^{3D} = \frac{L}{L+1} Y_{cfg}(Y_{ft})^L \tag{4}$$

where Y_{cfg} the yield of configuration layer.

Y_{ft} in equation (4) can be derived from Y_o using equation (1) or (2), depending on the graph topology. Because configuration layer uses conservative technology, Y_{cfg} is calculated in a different way. Suppose the transistor count of reconfiguration circuit for the ith target layer is T_{cfg}^i and the transistor count of the circuit on the ith target layer is T_o^i. Let λ_{cfg} and λ_o be the minimum feature sizes of configuration and target layers respectively. Since all layers of 3-D structure are of the same area and all reconfiguration logic share the same configuration layer, the technology scaling factor for configuration layer can be estimated using transistor counts as follows,

$$S = \frac{\lambda_{cfg}}{\lambda_o} = \sqrt{\frac{1}{L} \frac{\sum_{i=1}^{L} T_o^i}{\sum_{i=1}^{L} T_{cfg}^i}} \tag{5}$$

With yield scaling model in [5], the yield of configuration layer can be estimated as

$$Y_{cfg} = \sqrt[S^{P-1}]{Y_o} \tag{6}$$

where, P is a constant of defect-size probability density function and usually $P \simeq 3$ [5]. By choosing appropriate fault tolerance granularity, it is easy to make T_{cfg}^i to be no more than $4-5\%$ of T_o^i [11]. Hence S can be approximated to be $5/\sqrt{L}$. Yield of Y_{ft}^{3D} for both graph topologies can be calculated by plugging equation (6) into equation (4). In the remaining of this section, we investigate the changes of yield enhancement ($Y_{ft}^{3D} - Y_o^{3D}$) by varying Y_o with respect to different Ns, Ks and Ls respectively.

Impact of N. According to equations (1) and (2), only the yield of min-spare array circuit depends on N. The impact of N to the overall (effective) yield enhancement is investigated with given K and L. We choose the number of target layers (L) to be 3, as this number has been proved to be reasonable in terms of area-performance efficiency and heat dissipation [2]. Figure 5 shows the results with $K = 1$. The curves are similar for other Ks.

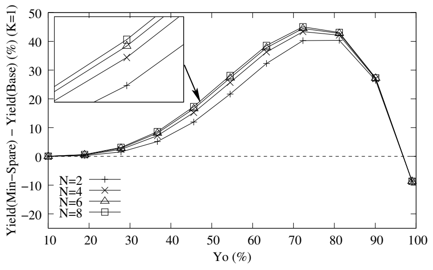
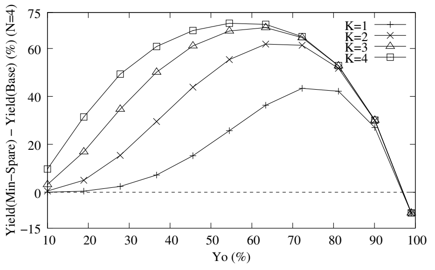


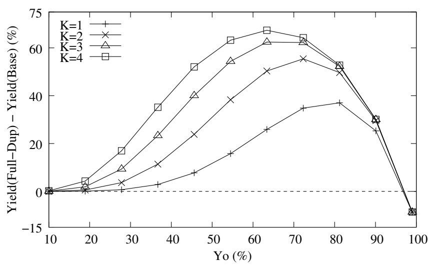
Figure 5. Overall yield enhancement of 3-D FT design with varying N ($K = 1$, $L = 3$).

Figure 5 shows that the defect tolerant design can improve the overall yield by up to 45%. When baseline yield is too low (<30%), a very small yield improvement can be earned from adding redundancies, resulting in only a little overall (effective) yield increase. When baseline yield is high enough (>95%), the potential of primitive yield increase becomes very limited so that even the ideal primitive yield improvement cannot compensate the inherent silicon cost penalty (due to extra configuration layer), resulting in even lower overall (effective) yield. For other cases, there is significant yield improvement (20%–30% on average). These results show that this defect tolerant design is able to noticeably reduce fabrication loss for immature technologies, helping profitability of foundries in the presence of increased time-to-market pressures.

For the cases of different Ns, finer fault tolerance granularity (larger N) helps reduce overall failure probability, resulting in higher yield enhancement (given all other parameters). Such improvement, however, becomes less significant with larger N. Note that N cannot be too large. Otherwise, the transistor count of reconfiguration logic will be increased dramatically [11], which may break the assumption made for scaling factor approximation and cause the aforementioned yield estimate inaccurate. In the remaining of this section, we choose N to be 4 in order to preserve that assumption while without significant yield loss compared with other Ns (according to Figure 5).



(a) Min-spare array ($N = 4$)



(b) Full-duplication graph

Figure 6. Overall yield enhancement of 3-D FT design with varying K ($L = 3$).

Impact of K. Figure 6 shows the overall yield enhancement of both fault tolerant graphs with respect to different Ks. Generally speaking, more faults can be tolerated with higher K, and the system is more likely to survive from defects. Thus, the overall yield enhancement becomes higher with larger K (given all other parameters). Because min-spare array achieves fault tolerance at finer granularity, it generally results in more yield increase than full-duplication graph (given K and L).

Impact of L. We examine the changes of overall yield enhancement with respect to different number of target layers. Figure 7 shows the results for $K = 1$, and the curves are similar for other Ks. In this figure, *X-l* denotes $L = X - 1$ (target layers). The overall (effective) yield calculation takes silicon cost penalty (due to extra configuration layer) into account, and such fixed penalty becomes more significant for less target layers, generally reducing the yield enhancement with smaller L. For *2-l* case, such penalty becomes 100% and there is no yield enhancement. At the same time, the primitive yields (without considering such penalty) are always increased (for instance, the curves of *X-l-p*), and the overall (effective) yield is improved as long as the primitive yield increase can compensate the configuration cost. Although more target layers (larger L) tend to amortize silicon cost penalty, it results in more configuration circuitry on the configuration layer and thus more transistors there, making the fabrication technology for reconfiguration logic less conservative and reducing Y_{cfg}. For low Y_o, the reduction of Y_{cfg} by larger L is more significant than the savings of amortized configuration cost penalty. Hence less target layers results in higher yield enhancement. When Y_o increases (>40–50%), silicon cost penalty savings begin to dominate, and larger L results in better yield enhancement. Note that there cannot be too many device layers in a 3-D structure. Otherwise, inter-layer via overhead must be taken into account, and heat dissipation as well as electromagnetic interaction become problematic issues [2].

Comparison with NMR. We compare the overall yields of this defect-tolerant design with traditional NMR method. Both designs achieve defect tolerance without external intervention. However, our design is more suited for asynchronous circuits because it does not require significant timing assumption. With different amounts of extra hardware resources, two design methods achieves different primitive yields. For fair comparison, we take spare resource overheads into account for effective yield calculations, and we call the results *normalized yields*, which reflects the cost-effectiveness of a design for yield. Let the 3-D structure have L target layers.

Regarding the K defect-tolerant design of min-spare array (suppose N-node array), K out of $N + K$ nodes are spare resources. The overall normalized yield is,

$$Y_{min}^{3D} = \frac{L}{L+1} Y_{cfg} (\frac{N}{N+K} Y_{ft})^L \qquad (7)$$

Regarding the K defect-tolerant design of full-duplication graph, K full replicas are spare resources. The overall normalized yield is,

$$Y_{dup}^{3D} = \frac{L}{L+1} Y_{cfg} (\frac{1}{K+1} Y_{ft})^L \qquad (8)$$

For NMR design, $2K$ full replicas are spare resources[1]. The overall normalized yield is,

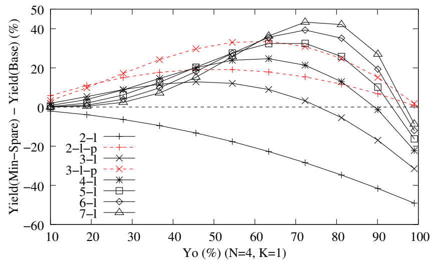
$$Y_{nmr}^{3D} = (\frac{1}{2K+1} Y_{nmr})^L \qquad (9)$$

where, $Y_{nmr} = \sum_{i=K+1}^{2K+1} \binom{2K+1}{i} (Y_o)^i (1 - Y_o)^{K+1-i}$

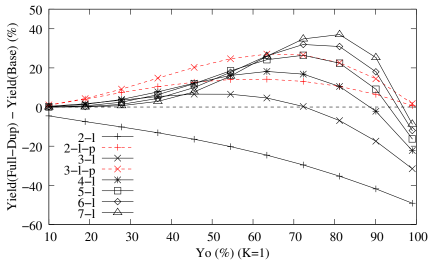
Figure 8 shows of the normalized yield enhancements of the defect-tolerant design with respect to NMR method ($Y_{min}^{3D} - Y_{nmr}^{3D}$ and $Y_{dup}^{3D} - Y_{nmr}^{3D}$) for different Ks. Here we choose N to 4 and L to 3.

Several observations can be made from Figure 8. First, this defect tolerant design results in higher normalized yield than NMR because fault tolerance is achieved at finer granularity: less silicon penalty is introduced, which is able to compensate its smaller primitive yield increase. With higher Yo, the primitive yield improvement reduces, making silicon penalty become dominant. This explains why the

[1]We omit the majority voter and the yield results are optimistic.

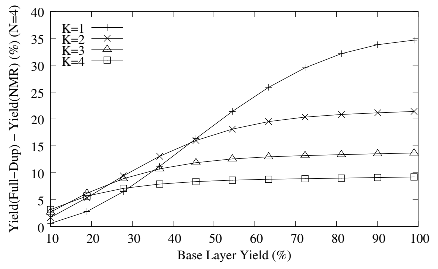


(a) Min-spare array ($N = 4$)

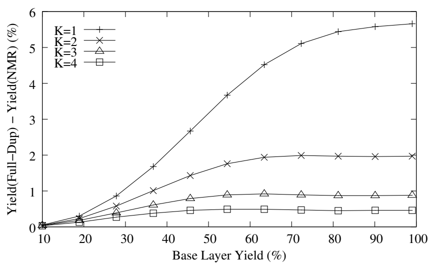


(b) Full-duplication graph

Figure 7. Overall yield enhancement of 3-D FT design with varying L ($K = 1$).



(a) Min-spare array ($N = 4$)



(b) Full-duplication graph

Figure 8. Overall normalized yield enhancement of 3-D FT design with varying K ($L = 3$).

normalized yield enhancement curves all monotonically increase with respect to Yo. When Yo is close to 100%, there is no normalized yield enhancement for both designs. But our design results in less normalized yield loss than NMR. With the aforementioned analysis, it can be concluded that this fault-tolerant design is more cost-effective than NMR. Second, larger K results in more fault tolerance capability but requires more extra hardware resource. For the defect-tolerant design, less normalized yields are expected with larger K because the extra hardware cost required becomes more significant than the primitive yield improvement. The only exceptional case is min-spare array with very small baseline yields (<20–30%) where the primitive yield increase always dominates.

Comparison with PLA Method. We give a coarse comparison of our defect-tolerant design and PLA method, as PLA hardware structure strongly depends on the circuit functions and it is hard to make general quantitative comparisons. Although PLA method could result in lower hardware overhead due to its fault tolerance at even finer granularity (gate-level instead of module-level), explicit fault diagnosis and manual reprogramming make the defect tolerance offline and increase the product test cost. Besides, PLA could cause noticeable performance degradation due to the large number of programmable gates. Compared with PLA method, however, our design implements reconfiguration at coase granularity, makes fault tolerance autonomous (thus reducing product test cost) and requires less configuration bits (thus smaller performance overhead).

4 Conclusion

This paper proposed a general asynchronous design for yield enhancement. With extra spare resources and specific self-reconfiguration logic, the VLSI circuits can maintain functionality in the presence of hard errors. The evaluation showed that this design results in significant yield enhancement (∼20-30% on average) (even with pessimistic evaluation). Unlike traditional NMR method, this design is suited for asynchronous circuits (due to no comparison procedure) and more cost-effective in terms of yield improvement (due to fault tolerance at finer granularity); Compared with PLA and FPGA-based methods, this design largely reduces product test cost (due to purely hardware-based autonomous reconfiguration) and results in less performance overhead (due to less programming bits).

This defect tolerance method can be conveniently applied to synchronous designs without significant change, and similar yield enhancement curves are expected (although the numbers might be different). Note that different methods should be used to implement fail-stop behavior in target clocked circuits. One way to do this is duplicating the target circuit and comparing the results off the critical path on each clock cycle. If any mismatch is reported, the global clock will be shut down, activating online reconfiguration.

References

[1] M. Abramovici, C. Stroud, and M. Emmert. Using embedded FPGAs for SoC yield improvement. In *Proc. Design Automation Conference*, 2002.

[2] K. Banerjee, S. J. Souri, P. Kapur, and K. C. Saraswat. 3-D ICs: a novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration. *Proc. the IEEE*, 89(5), 2001.

[3] I. David, R. Ginosar, and M. Yoeli. Self-timed is self-checking. *Journal of Electronic Testing: Theory and Applications*, 6(2), 1995.

[4] K. Emerson. Asynchronous design – an interesting alternative. In *Proc. International Conference on VLSI Design*, 1997.

[5] A. V. Ferris-Prabhu. Yield implications and scaling laws for submicrometer devices. *IEEE Transactions on Semiconductor Manufacturing*, 1(2), 1988.

[6] I. Koren and Z. Koren. Defect tolerance in VLSI circuits: Techniques and yield analysis. *Proceedings of the IEEE*, 86(9), 1998.

[7] S-Y. Kuo and W. K. Fuchs. Fault diagnosis and spare allocation for yield enhancement in large reconfigurable PLAs. *IEEE Transactions on Computers*, 41(2), 1992.

[8] R. Leveugle, Z. Koren, and I. Koren et al. The hyeti defect tolerant microprocessor: a practical experiment and its cost-effectiveness analysis. *IEEE Transactions on Computers*, 43(12), 1994.

[9] A. M. Lines. Pipelined asynchronous circuits. Master's thesis, California Institute of Technology, 1995.

[10] N. R. Mahapatra, A. Tareen, and S. V. Garimella. Comparison and analysis of delay elements. In *Proc. the 45th Midwest Symposium on Circuits and Systems*, 2002.

[11] S. Peng and R. Manohar. Fault tolerant asynchronous adder throught dynamic self-reconfiguration. In *Proc. International Conference on Computer Design*, 2005.

[12] C. E. Stroud. Yield modeling for majority voting based defect-tolerant VLSI circuits. In *Proc. IEEE Southeast Regional Conference*, 1999.

[13] Y. Zorian. Optimizing manufacturability by design for yield. In *IEEE/CPMT/SEMI International Electronics Manufacturing Technology Symposium*, 2004.