FPGA Defect Tolerance: Impact of Granularity[*]

Anthony J. Yu Guy G. Lemieux
Department of Electrical and Computer Engineering
University of British Columbia, Vancouver, BC, Canada
Email: { anthonyy | lemieux } @ ece.ubc.ca

Abstract

As device sizes shrink, FPGAs are increasingly prone to manufacturing defects. The ability to tolerate **multiple** *defects is anticipated to be very important at 45nm and beyond. One possible approach to this growing problem is to add redundancy to create a defect-tolerant FPGA architecture. Using area, delay and yield metrics, this paper compares two redundancy strategies: a coarse-grain approach using spare rows and columns and a fine-grain approach using spare wires. For low defect levels and large array sizes, the coarse-grain approach offers a lower area overhead, but it is relatively intolerant to an increase in defect count. In contrast, the fine-grain approach has a fixed overhead of up to 50%, but the architecture can tolerate an increasing number of defects as array size grows. To achieve a similar level of yield recovery, the coarse-grain approach requires an area overhead in excess of 100%.*

1 Introduction

As FPGA density increases through transistor geometry scaling, FPGA devices are increasingly susceptible to manufacturing defects [3, 4, 9]. FPGAs also adopt advanced technology nodes as early as possible, well before they reach mature yield rates. In the future, FPGAs based on nanotechnology may experience defect rates as high as 20% [15]. Hence, it is imperative that FPGA manufacturers have an approach for tolerating device-level defects.

In terms of silicon area, modern FPGAs are predominantly programmable interconnect, where interconnect includes the physical wiring as well as transistors making up switches, buffers, and configuration bits. As a result, defects are more likely found in the interconnect than logic blocks. This makes the ability to tolerate interconnect defects extremely important. In this paper, we consider schemes to tolerate interconnect defects by adding spare or redundant interconnect resources. Since interconnect is often made up of irregular connection patterns, it can be very difficult to determine exactly where to place these spares.

[*]Code available at http://www.ece.ubc.ca/∼lemieux/downloads

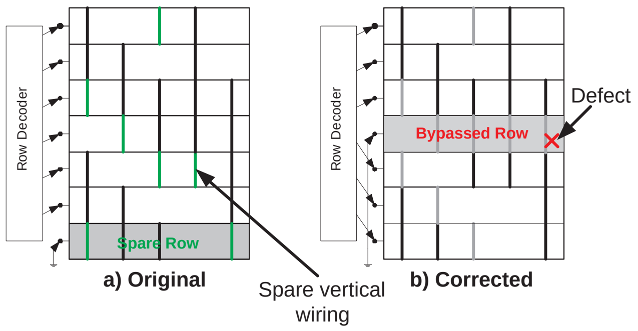

Figure 1: Overview of coarse-grain hardware redundancy

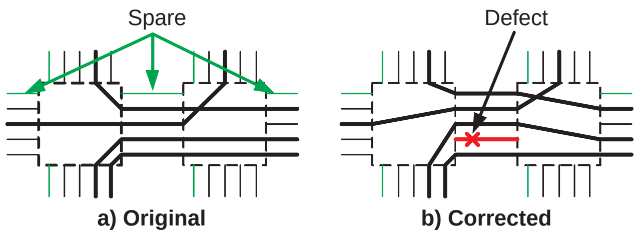

Figure 2: Overview of fine-grain hardware redundancy

The traditional *coarse-grain* approach for defect tolerance uses spare rows and columns [10]. As shown in Figure 1, defects are tolerated by bypassing the defective row or column and utilizing the spare. This tolerates clustered defects, but the consolidation of spare resources limits its ability to tolerate multiple, distributed random defects.

In comparison, a *fine-grain* approach [18] can tolerate multiple, distributed random interconnect defects. Figure 2 shows how a defect is avoided by shifting individual signals. This eliminates the need for rerouting, localizes the repair region, and minimizes timing variance due to correction.

This paper presents a yield model and compares the defect recovery ability of coarse-grain and fine-grain redundancy architectures. The comparison is based on four architectural parameters that influence yield: switch implementation, switch flexibility, array size and wire length.

2 Background

Fault-tolerant redundancy can be loosely classified into three groups: hardware redundancy, software redundancy and run-time redundancy. In hardware redundancy, spare resources facilitate the correction and avoidance of defects by swapping them with defective resources [8, 10]. In software redundancy, CAD tools are used to map around defective resources [11, 12]. This method typically has no hardware overhead but requires some form of reprogramming. Run-time redundancy deals exclusively with errors that occur during the operation of the FPGA, including transient faults caused by single-event upsets [2, 9].

This paper focuses exclusively on the yield of the two different approaches to hardware redundancy: coarse-grain redundancy (CGR) and fine-grain redundancy (FGR). Each of these are presented in detail below.

2.1 Coarse-grain Redundancy (CGR)

The spare row and column technique was one of the first proposed hardware redundancy approaches [10], and has been successfully applied in industry [1, 6]. This architecture adds one spare row and one spare column to the existing FPGA layout. It also changes the detailed routing switch design so that, in the event of a defect, the defective row or column can be bypassed. The spare takes up the slack as a large chunk of the design is shifted by one row/column.

This architecture can tolerate multiple defects or defect clusters within the same channel. However, as array size grows, it becomes increasingly unlikely that multiple randomly distributed defects will lie in the same row/column. Hence, additional spares are needed.

This paper considers two coarse-grain redundant (CGR) architectures which add multiple spares:

- **CGR-G**n denotes a multiple *global* spare architecture with n global spare rows/columns ($2n$ total spare rows+columns).
- **CGR-L**n**-S**p denotes a multiple *local* spare architecture divided into p row and p column subdivisions with n spares each ($2pn$ total spare rows+columns).

Figure 3 illustrates CGR-G2 and CGR-L1-S2.

The added cost of the global spare approach is: (1) the increased length of the routing wires, which increases capacitance, delay, and power; (2) the $2n$ spare rows+columns; and (3) the extra multiplexing in the routing switches to perform the bypass. Usually, n is small, so the spare rows/columns do not add significantly to area. However, the extra multiplexing required in *every* switch element to bypass 1 to n defective rows/columns is very significant [17].

In the local spare approach, the spares are evenly divided among subdivisions of the chip where defect correction is

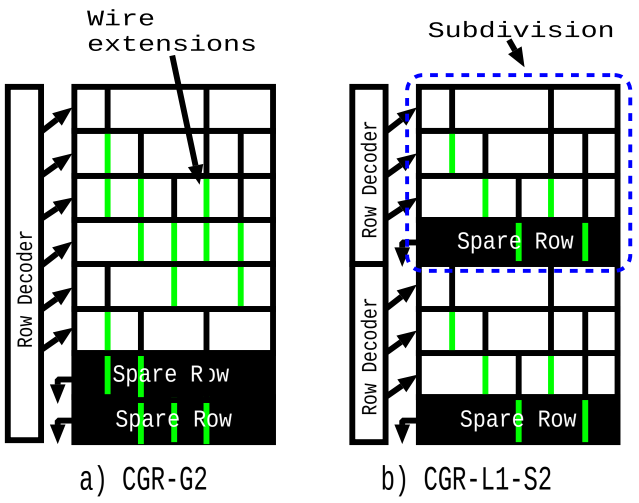

Figure 3: Multiple spare rows and columns

handled locally. This approach can add more total spares at a lower cost by increasing p and making n smaller, hence reducing the spare wire extensions and extra multiplexing.

2.2 Fine-grain Redundancy (FGR)

Rather than consolidating the spare resources into rows and columns, a fine-grain approach can be used instead [7, 18]. Spare routing resources are distributed evenly across the FPGA, allowing the architecture to tolerate multiple randomly distributed defects.

The first approach [7] adds spare switches in the switch block to bypass defective switches. However, the proposed design cannot tolerate defective wires, and defect correction imposes a severe timing penalty, making it impractical.

In the latter approach [18], two types of shifting multiplexers are introduced in the switch block. The *omux* multiplexer allows signals to shift up by 1 or 2 tracks and is used for defect avoidance. The *imux* multiplexer allows signals to shift down by 1 or 2 tracks and is used for signal restoration. The up/down shifts by 2 are required to correct wire bridging defects, but these can be left out if bridging is rare. Together, the *imux* and *omux* allow signals to route around defective resources. This approach is can tolerate defects in the switch block and in the wiring. Furthermore, defect correction does not significantly change signal timing.

To accommodate the shifting, spare wires are inserted into every trackgroup[1]. For trackgroups of length L, a total of $4L$ spare wires are needed with bridging support, and $2L$ without. These spare wires are normally unused. In the event of a defect, the faulty wire is bypassed and the shifting multiplexers reroute signals onto these spare wires.

[1] A trackgroup is a group of wires with the same start and end points.

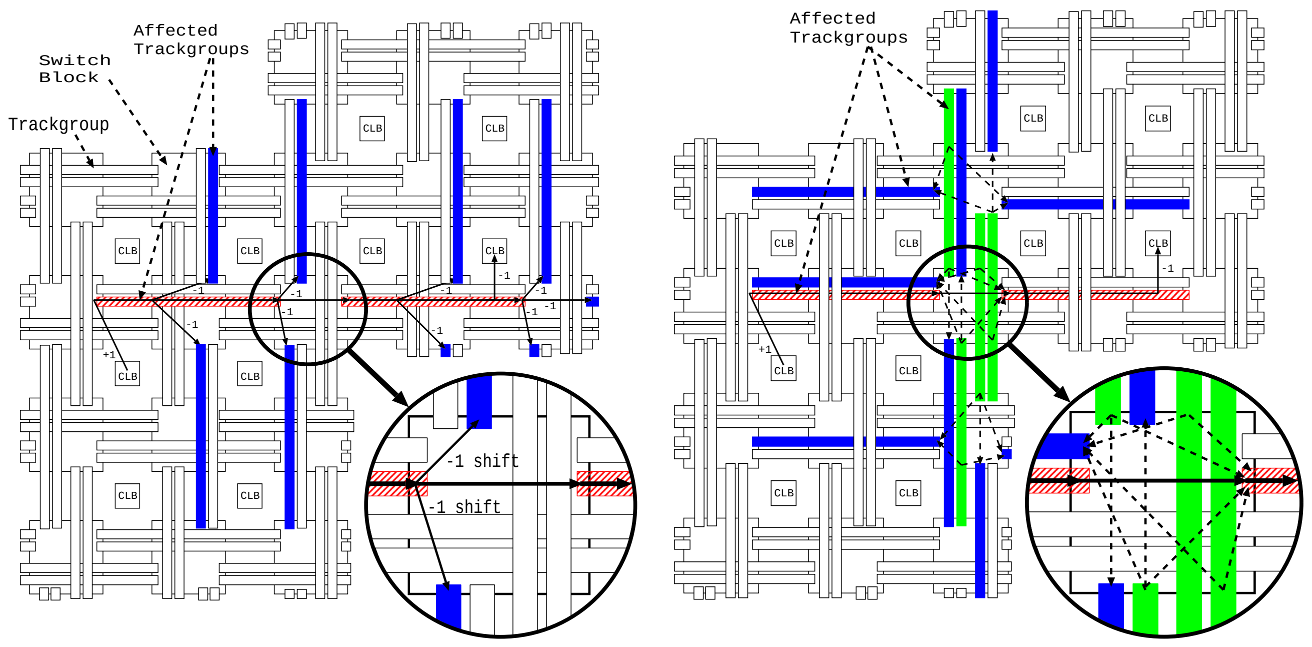

Figure 4: Track shifting to correct a double-length defect: a) direct fanout regions (left), b) direct fanin regions (right)

Figure 4 presents an example of defect correction. The defect in question spans two trackgroups (red/striped) and affects the same track number. To avoid the defect, all signals in these two defective trackgroups are shifted by “+1”. To localize the defect, all direct fanouts, *i.e.* downstream trackgroups, are restored by “-1” (Figure 4a, blue/dark-gray). Signals entering the defective trackgroup from the far left are already shifted and avoid the defect. Signals entering the defective trackgroup from the vertical channel in the circled switch block are considered *direct fanins* to the defective trackgroup (Figure 4b, green/light-gray). The defect and contention prevents correction at this switch block. Instead, the “+1” is applied at their source, and all of their fanouts are restored by “-1” (Figure 4b, blue/dark-gray).

As Figure 4 shows, correcting a defect involves participation of a number of neighbouring trackgroups. The maximum number of trackgroups affected by defect correction is called the *minimum fault-free radius (MFFR)*. The MFFR of the previous example encompasses direct fanouts, direct fanins, and all fanouts of the direct fanins.

In this paper, we assume that multiple defects can be tolerated if their MFFR areas are non-overlapping. This is slightly pessimistic, but it simplifies our calculations.

2.3 Types of Defects

In CGR, all defects are treated equally. A channel containing any defect is always replaced with a spare row or column. This simplifies the correction process, but has the drawback of being inefficient with resource usage.

In FGR, interconnect defects can be classified as: single-length, double-length, and bridging defects [18]. Each category differs in the MFFR required to repair the defect.

3 Computing Yield

For our yield analysis, we assume that all faults are bridging defects (worst case). We also assume that there are no logic faults, which are intolerable in the present FGR architecture. Furthermore, we assume that all implementation options require the same amount of silicon die area. This allows us to compare architectures in terms of how many defects they can tolerate, but in so doing this ignores the increase in defect count that arises from an area increase.

3.1 Course-grain Redundancy (CGR)

Our CGR yield model assumes the following:

- all channels have identical routing resources and thus have an equal probability of being defective;
- the vertical and horizontal channels are disjoint (a defect can be isolated to just a row or just a column);
- $M \times M$ FPGAs are perfectly symmetrical; and
- equal spare rows and columns to keep the array square.

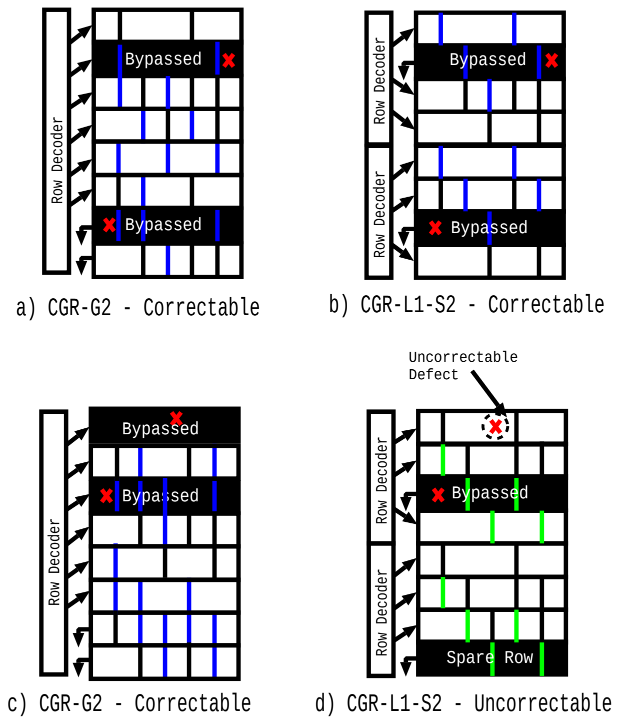

Figure 5: Comparison between CGR-G2 and CGR-L1-S2

To inject a random defect, a row/column is randomly selected and the defect count for that row/column is incremented. For CGR-G1, a failure occurs when there are defects in two different rows or two different columns. Architectures with multiple *global* spare rows and columns are evaluated in a similar manner. A failure occurs when there are more defective rows or columns than there are spares.

In CGR-L*n*-S*p*, each subdivision has exactly *n* designated spare rows/columns and can tolerate at most *n* defective rows/columns per subdivision. Figure 5 highlights the differences in terms of defect correction between CGR-G2 and CGR-L1-S2.

3.2 Fine-grain Redundancy (FGR)

To model the behaviour of defect correction for FGR, we assign state variables to every trackgroup within the FPGA. A trackgroup can have one of three states: *perfect*, *faulty*, or *must be perfect*. The *faulty* state indicates the presence of a defect. The *must be perfect* state is used to mark the MFFR of a defect. As mentioned before, the MFFR defines the region needed for shifting to avoid and restore around the defect. To guarantee that a defect can be correctable, the MFFR must be defect-free.

Defects are injected into the model by randomly selecting a trackgroup and setting its state to *faulty*. The neighbouring trackgroups as defined by the MFFR are marked as *must be perfect*. The MFFR will vary depending on the defect type and the underlying routing architecture. Chip failure occurs when the *faulty* or *must be perfect* trackgroups for a newly injected defect are already in a non-*perfect* state.

Our yield approximation for FGR is pessimistic in two ways. First, we always inject bridging defects into our model which have very large MFFRs. Second, we do not allow MFFR overlaps. In reality, not all defects are bridges, so significantly smaller MFFRs are possible. Further, the avoidance and correction of certain defects can be overlapped. We suspect that accounting for these two factors will greatly improve yield for the fine-grain architecture.

4 Architectural Considerations

Several factors can affect yield. This section discusses a few important ones. Results will follow in the next section.

4.1 Switch Implementation and Flexibility Impact on Yield

As presented in [18], the routing switches in FGR can be implemented in a variety ways. First, the *imux* can be built out of an encoded tree of pass transistors or as a flattened 1-level multiplexer. These two essentially trade off between area and delay, and do not impact yield. The *imux* can also be embedded into the directional switch. This inflates the area of the switch, but the added connectivity was shown to improve circuit routability and reduce the MFFR of defects. Out of delay considerations, the *omux* was built using a flattened 1-level multiplexer.

The shifting ability of the switch can also be varied. If the ability to shift by 2 is eliminated, the size of the switch will decrease. Bridging defects and source-drain shorts can still be tolerated, but avoiding them requires two "+1" shifts followed by two "-1" shifts. In fact, any combination of shifts, a "+2" followed by two "-1", two "+1" followed by a "-2" is acceptable. However, changing the shifting ability of the multiplexers increases both the number of defect categories and the MFFR.

Of the seven different switch implementations investigated in [18], we chose the "EM22" switch for this work: an embedded *imux* with shifts by 1 and 2. This has the highest area overhead, but tolerates the most defects.

The number of wires connected to a given switch block wire is defined as its flexibility, F_s [16]. With long wires, F_s can describe different flexibility in switch blocks at *endpoints* and at *midpoints* of a wire [14]. Lower F_s values equate to fewer connections, and thus smaller MFFRs. Based on [18], we chose $F_s = 3$ for endpoints and $F_s = 1$ for midpoints. This is "E3M1" in the notation used in [18].

4.2 Array Size Impact on Yield

In future technology nodes, the expected number of defects per fixed die area will increase [3, 9]. This is attributed to an increased sensitivity to smaller defects and increased process variation. Future technologies also allow construction of bigger array sizes in the same fixed die area. It is thus desirable to have an architecture that can tolerate an increasing number of defects as the array size grows.

CGR-G1 demonstrates a decreasing amount of area overhead as array size increases [4]. However, as array size grows, it also becomes increasingly unlikely that multiple defects will lie in the same row/column. Thus, to maintain a fixed yield for growing array sizes in future technologies, more spare rows and columns will be needed.

In FGR, spare resources are distributed across the FPGA. Increasing the array size increases the number of trackgroups in the FPGA, and thus the amount of available spare resources. Since the amount of spare resources naturally grows with size, the number of tolerable defects increases.

4.3 Wire Length Impact on Yield

In CGR, we assume that the routing network within the rows and columns are identical and disjoint. Since all rows and columns contain the same routing resources, the ability to replace a defective row/column containing wires of any wire length with a spare row/column is guaranteed. This eliminates the dependency of wire length on defect correction for the spare row and column architecture.

For FGR, increasing wire length increases both the fanout and fanin of all routing wires due to an increase in the number of midpoint locations. This increases the MFFR and negatively impacts yield.

Current FPGA routing architectures utilize multiple wire lengths within their routing architecture. In FGR, wires of different lengths are modeled as disjoint routing networks. Each routing network has its own unique set of spare resources. Defect correction is restricted to the individual routing networks. These disjoint networks "join together" at the connection blocks which can correct for defects in any of them.

4.4 Limitations

Our yield model does not account for switch area. The area of the switch is an important consideration because larger circuits have a greater probability of being defective than smaller ones. For CGR-Gn and CGR-Ln-Sp, significantly larger switch areas are needed because of the necessary bypass circuitry. We do not model this, so we are overestimating CGR yield.

We assume defects in CGR requires either a spare row or a spare column to be tolerated. Some defects require both, hence we over estimate yield for this architecture.

Power/ground shorts have been ignored in our fault simulation. These defects cannot be tolerated in either architecture. We also assume no defects in the logic blocks. For a more accurate estimation, the yield model should be supplemented with manufacturing data.

We assume that routing tracks within the same channel are laid out beside one another. When we inject bridging defects into the model, two adjacent tracks are made unusable. This assumption allows us to bypass bridging defects by performing shifts by 2. Larger faults (*e.g.*, 3 wires) would not be tolerable in the fine-grain architecture.

Our FGR model assumes that there can be at most one defect per trackgroup. However, certain types of single and double-length faults can in fact be overlapped. For example, two defects can be overlapped if the underlying defect-tolerant architecture supports bridging defects, the faults themselves are not bridging defects, and the defects do not reside on adjacent tracks. Provided that these conditions are met, the defect on the lower track can be avoided using a "+1" shift while the one on the higher track with a "+2" shift. Also, defects in the same tracks should be tolerable.

When computing yield, we assume that the delay bypass path of the *omux* is not being used [18]. If it is utilized, we must either perturb timing when correcting a defect or limit device defects to one-per-channel.

Lastly, when we compute the MFFR for FGR, we assumed that the defects are injected into the middle of an infinitely sized FPGA. This results in worst-case MFFRs for the defects. Since the trackgroups near the edge of the FPGA have lower connectivity than those in the center, the use of the worst-case MFFR value for all defects is overly pessimistic.

5 Results

The yield estimates for CGR and FGR were obtained through Monte Carlo simulations. For a given number of defects, random faults were injected into the interconnect for 100,000 different FPGA die. We do not consider intolerable defects such as power/ground shorts.

The area and delay numbers were obtained from an enhanced version of the VPR place and route tool, VPRx [14]. VPRx was used to map the large *clma* MCNC benchmark circuit [5] into an island-style FPGA consisting of directional wires [13] and CLBs containing eight 4-input LUTs.

5.1 Fixed Array Size

Figure 6 presents the yield for an 32×32 FPGA with the indicated number of *additional* global rows/columns. The yield for CGR-Gn remained at 100% up until the defect count became greater than the total spare rows+columns

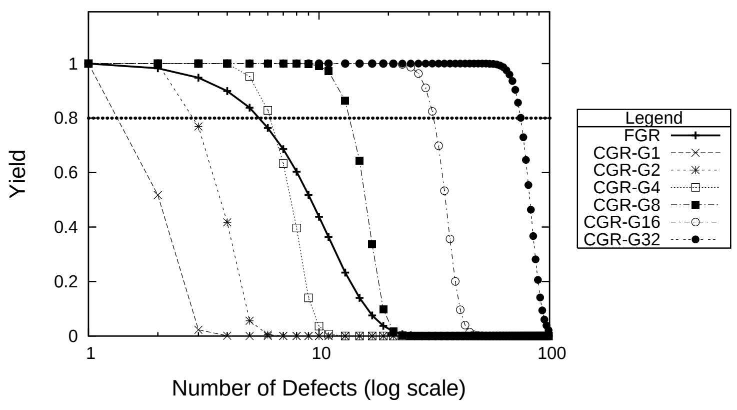

Figure 6: Increasing number of global spares (*M*=32)

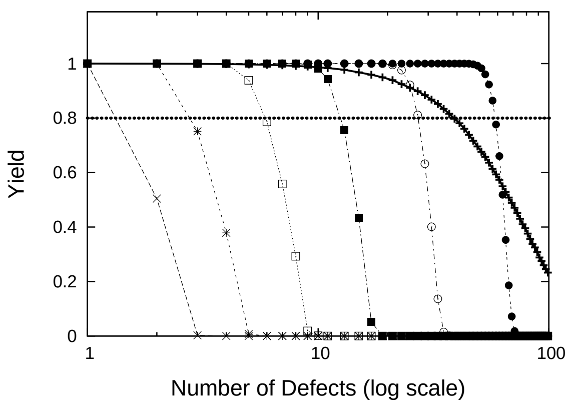

Figure 8: Increasing number of global spares (*M*=256)

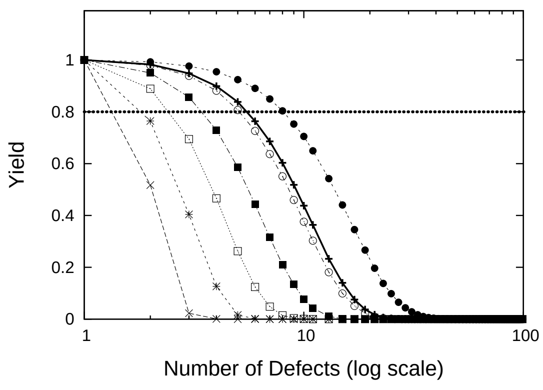

Figure 7: Increasing number of local spares (*M*=32)

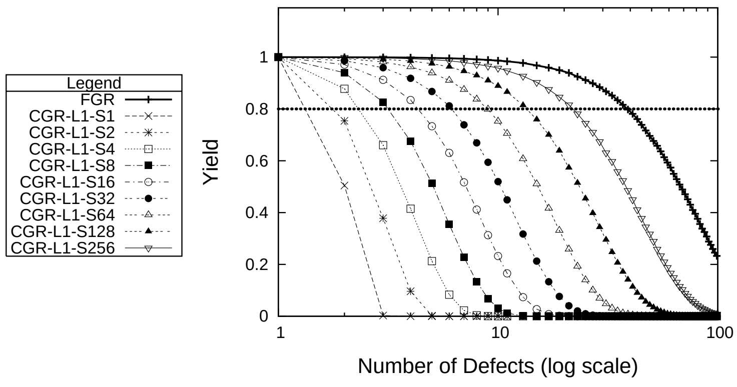

Figure 9: Increasing number of local spares (*M*=256)

in the architecture. After this threshold, the yield decreases dramatically. This curve represents the best case yield for the spare row and column architecture. Yield for this particular architecture is especially sensitive to the number of spare rows/columns in the system. There is a significant yield improvement when we move from one *global* spare to two *global* spares. FGR demonstrate similar yields as CGR-G4.

The CGR-L*n*-S*p* architecture demands that defects are spaced far apart from one another. If too many defects reside in the same subdivision, chip failure occurs. The impact of this restriction is shown in Figure 7. The figure shows the yield of a 32×32 FPGA with the indicated number of *additional* local spare rows/columns. Notice that the yield decreases almost immediately and is significantly less than that of the global approach. The break even point for this architecture is 16 partitions with one spare row/column each. At this point, CGR-L1-S16 demonstrate yields similar to FGR.

Figures 8 and 9 presents the yield curves for CGR-G*n* and CGR-L*n*-S*p*, but the array size of 256×256 is used. At this array size, we notice that number of defects tolerated by FGR increases. FGR is now approximately equivalent to the yield of CGR-G16 and has higher yield than all implementations of CGR-L1-S*p*. Note that a large number of global spare rows/columns is impractical (and potentially infeasible) because of the long wire extensions and significantly larger switch area needed for the bypassing circuitry.

5.2 Increasing Array Size

CGR can tolerate multiple defects in the channel. However, as array size grows, it becomes increasingly unlikely that defects will lie in the same channel. The yield for a fixed number of spare rows and columns was observed to be largely independent of array size. The only way to increase yield is through the addition of spare resources.

For the FGR architecture, the amount of spare resources increases as array size grows. Since the amount of resources needed for defect correction is constant, increasing

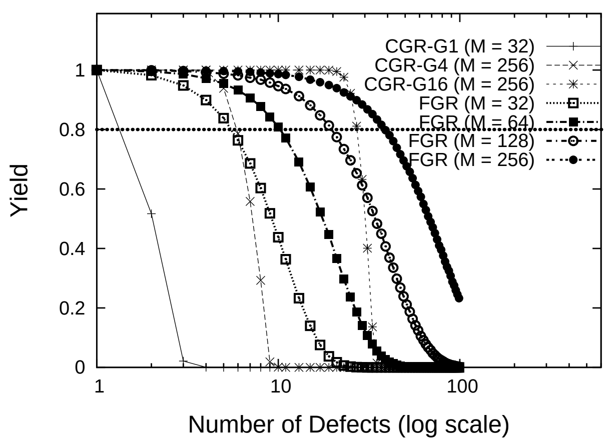

Figure 10: Increasing array size for FGR (L=4)

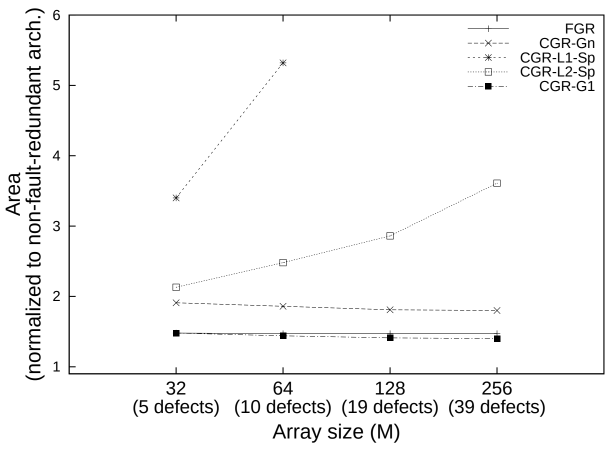

Figure 11: Area comparison between FGR and CGR at same number of defects (L=4)

the number of spare resources means the ability to tolerate more defects. The yield for increasing values of M for FGR is shown in Figure 10. We observe that this architecture can tolerate an increasing number of defects as array size grows.

Figure 11 presents an area comparison between FGR and CGR for different values of M. For FGR, the reported area includes **all** necessary shifting multiplexers and spare wires for the given value of M. For CGR-Gn and CGR-Ln-Sp, the n and p values were chosen to match the number of tolerable defects at the 80% yield level to the FGR architecture. As well, the area of CGR-G1 is shown which can tolerate only 1-2 defects. The area overhead for these CGR architectures is estimated from (1) the required number of spare rows/columns, and (2) the bypass circuitry. The bypass circuitry is estimated to be 30% for CGR-G1, and 50% (optimistically) for the other CGR variants [17]. For large values of M, the results show that CGR-Gn and CGR-Ln-Sp requires greater area to tolerate the same number of defects as FGR.

5.3 Wire Length

Long wires have a greater number of fanins and fanouts. This results in a larger MFFR and yield reduction. In Figure 12, we see that the yield for the FGR decreases as wire length L increases. Also note that the yield for mixed wire lengths is lower than the yield of the individual wire lengths it is composed of. This is largely the consequence of how mixed wires are implemented and modeled. In the fine-grain architecture, the routing network for different length wires are disjoint.

The area and delay numbers for the *clma* benchmark circuit are presented in Figure 13. The circuit was mapped into an FGR architecture of length 4, 8 and 16 wires. The delay results have been normalized to an architecture without redundancy. The channel width was fixed at 224 for all wire lengths. 224 is the minimum channel width needed to route *clma* using length 16 wires. Figure 13 shows the area breakdown for the circuit.

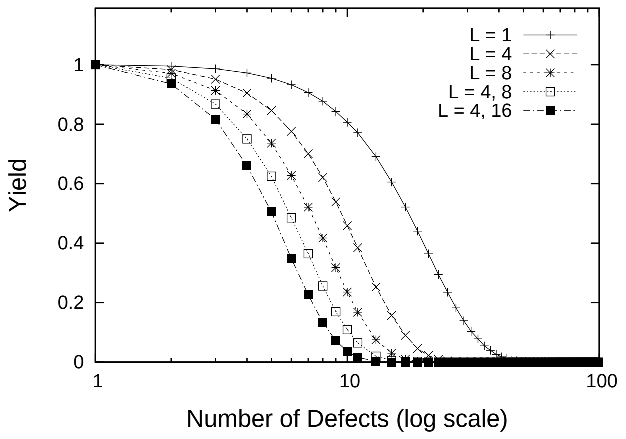

Figure 12: Yield for varying wire lengths for FGR (M=32)

6 Conclusions

This paper presented a comparison between coarse-grain redundancy (CGR) and fine-grain redundancy (FGR). Both approaches embody the idea of replacing a defective resource with a spare unused one; however our investigation indicates that the choice of defect tolerant architecture has a significant impact on yield and area overhead.

We found that at low defect levels, CGR has a lower area overhead than FGR. For sufficiently low defect levels, the area overhead for CGR diminishes as array size in-

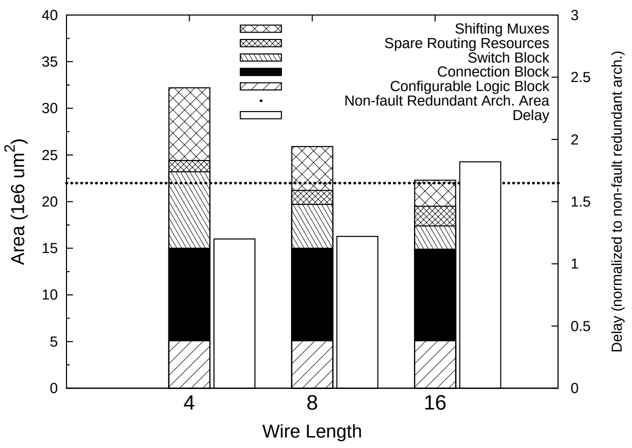

Figure 13: FGR area, delay with varying wire lengths

creases. This is not the case for FGR, where the area overhead for this approach is fixed at up to 50% for all array sizes. Despite the fixed cost of redundancy, FGR can tolerate an increasing number of defects as array size grows. This is extremely important as the expected number of defects increases as devices shrink. When comparing CGR and FGR at equal defect levels, we found that CGR requires more area overhead to tolerate the same number of defects as FGR.

Another factor that influences yield is wire length. For FGR, we observed that yield decreases as wire length increases. This is not so for CGR. Using spare rows and columns for defect correction is wire length independent.

Taking these factors into account, we feel that future redundancy architectures should take a median approach to defect tolerance. By combining CGR and FGR, we can efficiently tolerate random distributed defects as well as clustered and bridging defects.

Future work will investigate ways to reduce area overhead of CGR and FGR. As well, when defect densities are very high, it is unlikely that they will all appear as interconnect faults. Hence, it is important to extend FGR to tolerate clustered defects, logic defects, as well as seemingly intolerable defects. Perhaps this can be done by producing a FGR/CGR hybrid.

References

[1] Altera Corp. Altera's patented redundancy technology dramatically increases yields on high-density APEX 20KE devices. In *Press Release,* Nov. 27 2000.

[2] G. Asadi and M. B. Tahoori. Soft error rate estimation and mitigation for SRAM-based FPGAs. In *FPGA,* 2005.

[3] N. Campregher et al. Analysis of yield loss due to random photolithographic defects in the interconnect structure of FPGAs. In *FPGA*, pages 138–148, February 2005.

[4] N. Campregher et al. Yield modelling and yield enhancement for FPGAs using fault tolerant schemes. In *FPL*, 2005.

[5] Collaborative Benchmarking Laboratory. Lgsynth93 benchmark set: Version 4.0. Technical report, North Carolina State University, 1993.

[6] Crosspoint Solutions Inc. FPGA redundancy. In *United states patents #5,777,887,* 1998.

[7] A. Doumar and H. Ito. Design of switching blocks tolerating defects/faults in FPGA interconnection resources. In *IEEE Int'l Symp. on Defect and Fault-Tolerance in VLSI Systems*, pages 134–142, 2000.

[8] A. Doumar, S. Kaneko, and H. Ito. Defect and fault tolerance FPGAs by shifting the configuration data. In *Int'l Symp. on Defect and Fault-Tolerance*, pages 377–385, 1999.

[9] S. Hareland et al. Impact of CMOS process scaling and SOI on the soft error rates of logic processes. In *IEEE Nuclear and Space Radiation Effects Conf.*, pages 73–74, 2001.

[10] F. Hatori, T. Sakurai, et al. Introducing redundancy in field programmable gate arrays. In *IEEE CICC*, 1993.

[11] W.-J. Huang and E. McCluskey. Column-based precompiled configuration technique for FPGA fault tolerance. In *IEEE Field-Programmable Custom Computing Machines*, 2001.

[12] J. Lach, W. H. Mangione-Smith, and M. Potkonjak. Efficiently supporting fault-tolerance in FPGAs. In *FPGA*, 1998.

[13] G. Lemieux, E. Lee, M. Tom, and A. Yu. Directional and single-driver wires in FPGA interconnect. In *IEEE Int'l Conf on Field-Programmable Technology*, 2004.

[14] G. Lemieux and D. Lewis. *Design of Interconnection Networks for Programmable Logic*. Kluwer Academic Publishers, Boston, 2004.

[15] H. Naeimi and A. DeHon. A greedy algorithm for tolerating defective crosspoints in nanoPLA design. In *FPT*, 2004.

[16] J. Rose and S. Brown. Flexibility of interconnection structures in field-programmable gate arrays. *Journal of Solid State Circuits*, 26(3):277–282, 1991.

[17] A. J. Yu. Defect tolerance for yield enhancement of FPGA interconnect using fine-grain and coarse-grain redundancy. Master's thesis, Dept. of Electrical and Computer Engineering, Univ. of British Columbia, August 2005.

[18] A. J. Yu and G. G. Lemieux. Defect-tolerant FPGA switch block and connection block with fine-grain redundancy for yield enhancement. In *FPL*, 2005.