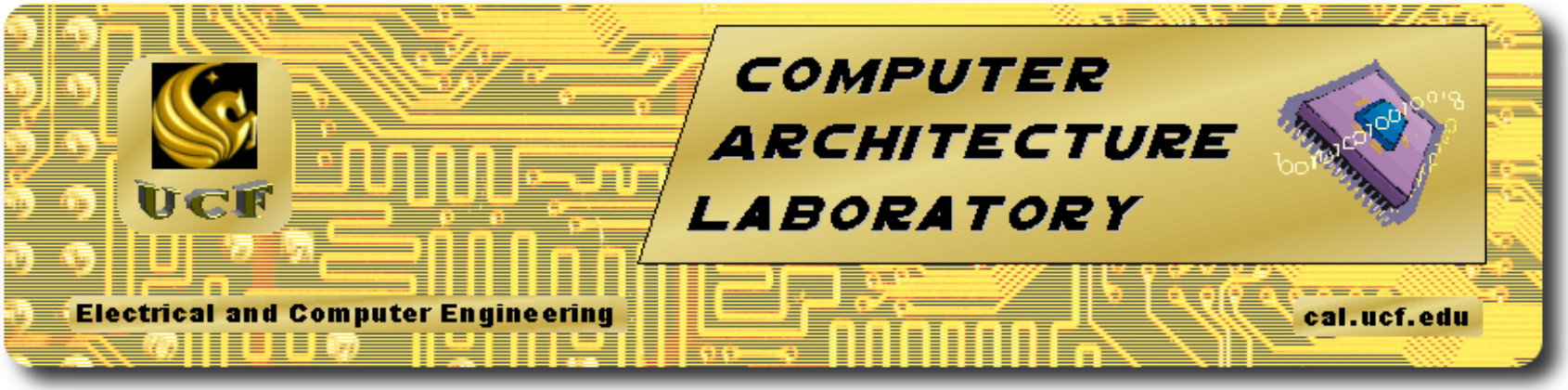


Scalability of Sustainable Self-Repair to Mitigate Aging Induced Degradation in SRAM-based FPGA devices

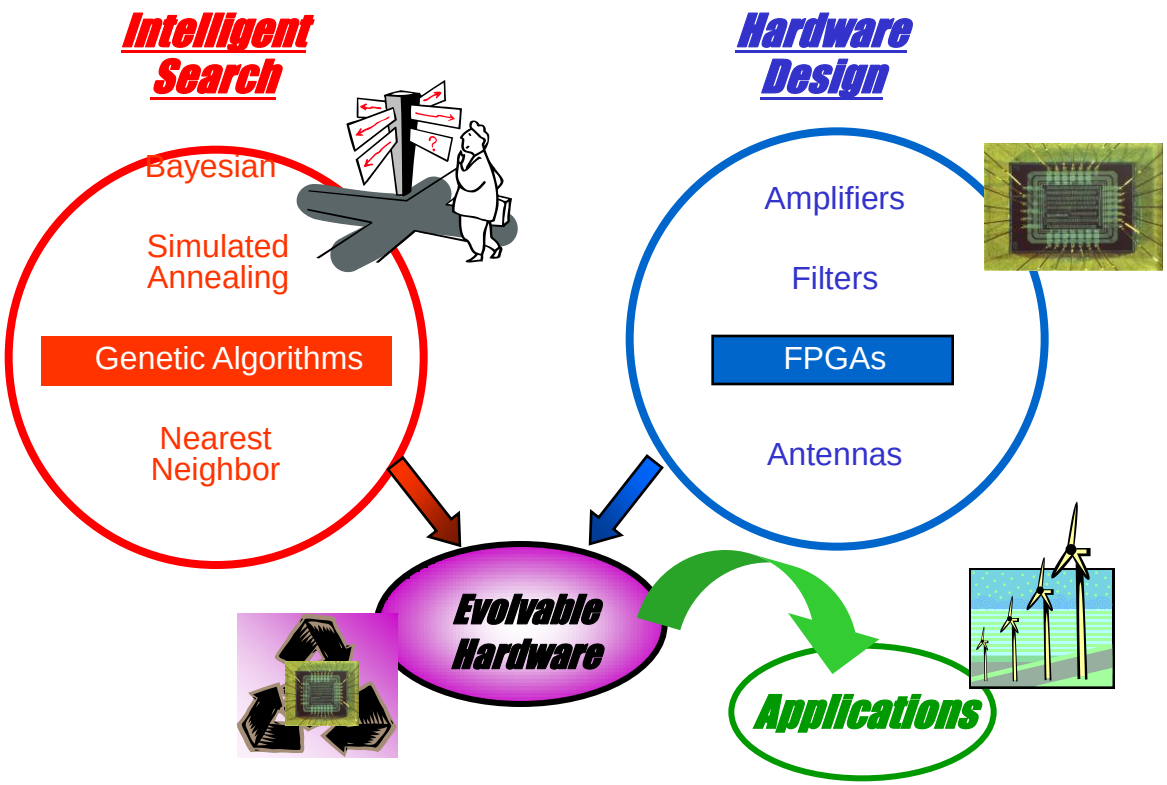
ReSpace/MAPLD 2011 Conference,
22-25 August 2011,
Albuquerque, NM

Rizwan A. Ashraf, Rashad S. Oreifej, and
Ronald F. DeMara
Department of Electrical Engineering and Computer Science,
University of Central Florida

Evolvable Hardware

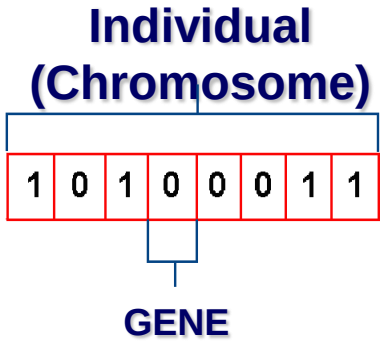
Automated Construction: **develop *Electronic Circuits* by *Intelligent Search***

Applications: ***Design*, *Optimization*, or *Failure Recovery* phases**



GAs frequently use binary strings to represent candidate solutions: *genotype*

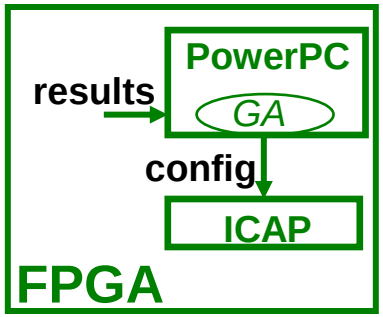
- Translation to FPGA Configuration bitstream maps genotype to phenotype FPGAs for evolving digital logic



Operational Flow of EHW Techniques

1. Objective for EHW procedure is specified
   - realize a **8-bit adder circuit** or program a digital chip to perform a function such as **tone discrimination**
   - Relative ranking called ***Fitness Function*** is defined
2. Population of alternative designs is created
   - completely **at random** or **seeded** with hand designed
3. Genetic Algorithm invoked to evolve each alternative
   - **Fitness evaluated** for alternatives using FPGA ***programmable*** *logic* and *interconnect* resources to realize arbitrary number of circuits
   - ***Genetic Operators*** used to increase fitness
4. Fitness Exit Criteria checked
   - If ***max(fitness)<threshold*** then repeat Step 3
5. Best design represents desired hardware configuration
   - FPGA final configuration implements the circuit

Example: GA running on PowerPC platform configures the reprogrammable SRAM-based FPGA through ICAP



GA-based Refurbishment

Genetic Algorithms:

- Implement ***guided trial-and-error search*** using principles of Darwinian evolution
- ***Iterative selection*** enforces "survival of the fittest"
- Genetic operators - ***mutation***, ***crossover, …*** - can be used to refurbish FPGA-based designs

Previous GA-based Refurbishment on FPGAs:

- Vigander used three faulty modules in a *Triple Modular Redundancy (TMR)* arrangement partially refurbished through GA to circumvent the problem of refurbishment of large-sized modules (circuits) [1].
- DeMara *et. al.* used a technique referred to as *Competitive Runtime Reconfiguration (CRR)* [2] which performs fitness evaluation on actual runtime inputs and without a static pre-defined fitness function.
- Oreifej *et. al.* developed an intrinsic evolutionary platform for refurbishment of digital circuits on a Xilinx Virtex-II Pro FPGA device by directly manipulating its bitstream using GA [3]

Focus of this work

- **Effectively utilize GA based technique for refurbishment of SRAM-based FPGAs at *application level***
- **Address the notion of sustainability of *multi-year* *<u>space missions</u>*, employing SRAM-based FPGAs.**
  - FPGAs attractive platform due to short design deployment time and flexibility in reconfiguration at runtime.
- **Counter the affects of aging induced hard faults in FPGAs.**
  - Trending concern in sub 90nm technology [4]
  - Xilinx launches 28nm FPGAs [5]; Lower gate length & lower operating voltages affects reliability of FPGA devices.
  - Hot Carrier Effect (HCE), Time Dependent Dielectric Breakdown (TDDB) and Electromigration (EM) can cause permanent faults.
  - Aging induced degradation can impact application sized cores in as soon as 3 years under normal operating conditions [4]
- **Aging may cause multiple faults to occur, the single fault assumption may not be widely applicable in future systems [6]**
- **Most works focus on Single Event Upset (SEU) based transient or soft faults, which are caused due to radiation effects, not aging.**
  - Scrubbing can be efficiently employed to alleviate the effects of SEUs [7]

Related Work

- **McCluskey's Reconfigurable Architecture for Autonomous Self-Repair [8]**
  - Utilize the reconfiguration feature of FPGAs to recover from permanent faults
  - Multiple precompiled configurations (at design time) with disjoint unused portions of FPGA are used to tolerate faults in these portions
  - Can only handle a single *failure* throughout the operational period of an application, per unused block.
  - Online detection of error using Concurrent Error Detection (CED)
  - **Issue:** *Sustainability?*

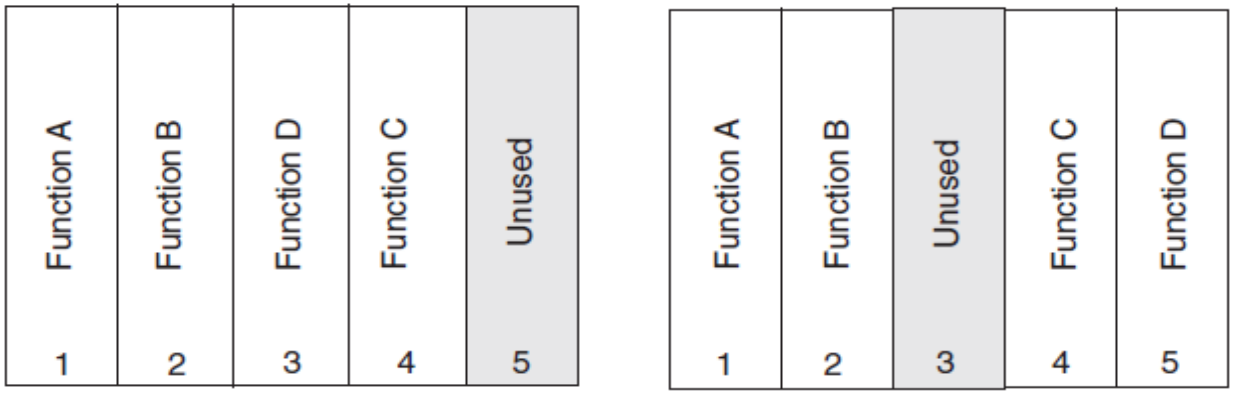


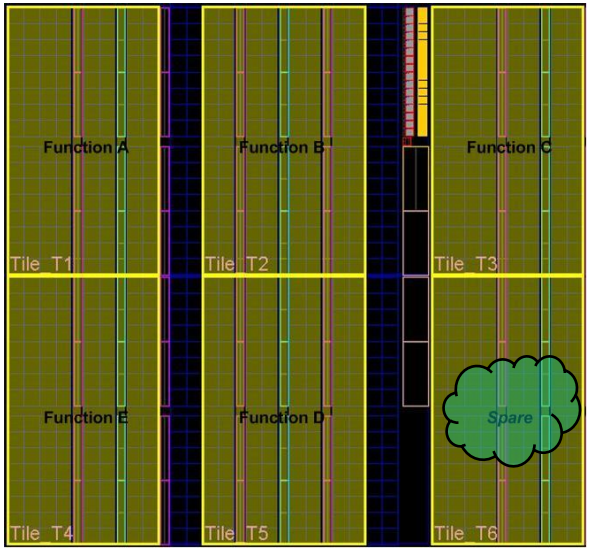
Figure from [8] illustrating the functionality based over-lapping scheme

DTRS: Coarse-grain Partitioning

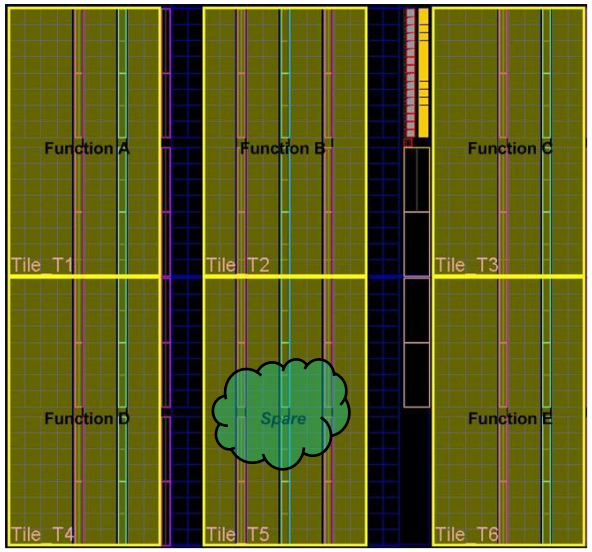
***Dynamic Tile Reconfiguration for Sustainability* (DTRS)**

- **Application is partitioned into functionally decoupled sub-circuits at design-time.**
- **Sub-circuits are to be mapped onto independent Reconfigurable Regions on the FPGA**
  - Minimum size of a region is based on the minimum configuration frame size of the FPGA being employed e.g. Xilinx Virtex-4 FPGA has a minimum frame of 16x1 Configurable Logic Blocks (CLBs) i.e. allows more than one non-overlapping regions in a given column.
- **Generate alternative configurations via functionality-based overlapping scheme [8]**
- **The independent regions are referred to as *tiles* in this work.**

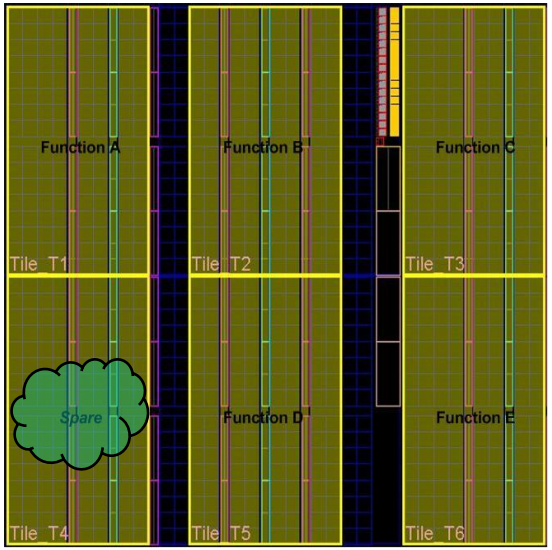
Example: Six design time configurations w/alternatively located spares



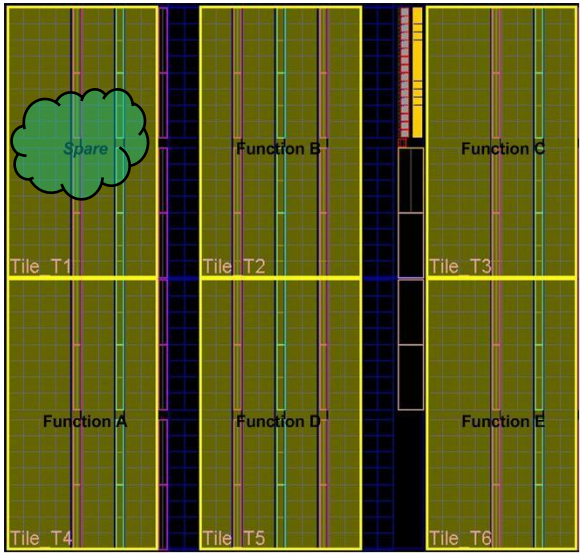
Configuration #1



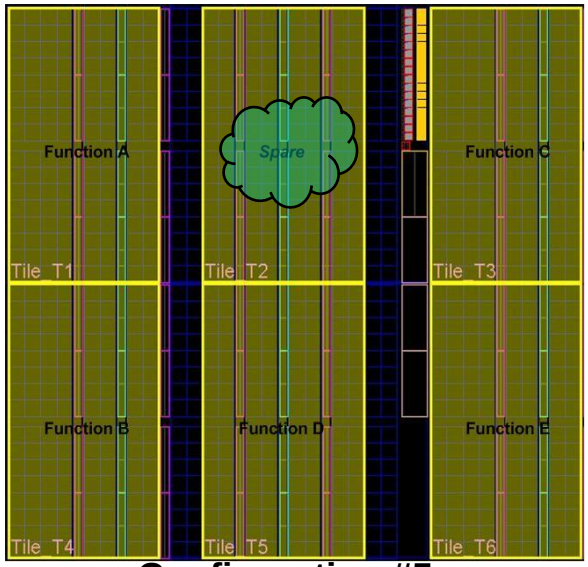
Configuration #2



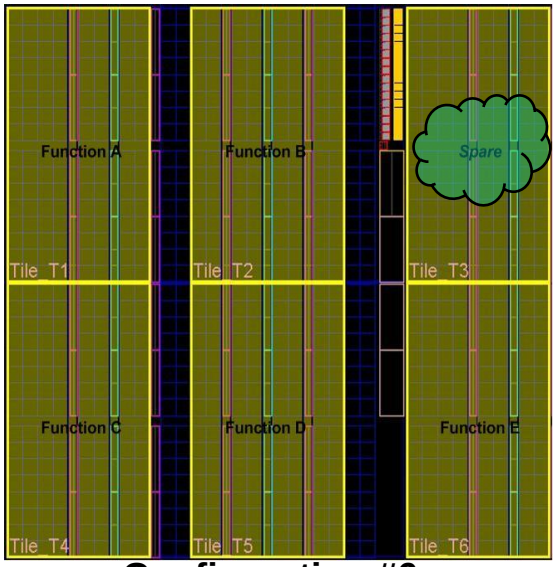
Configuration #3



Configuration #4



Configuration #5

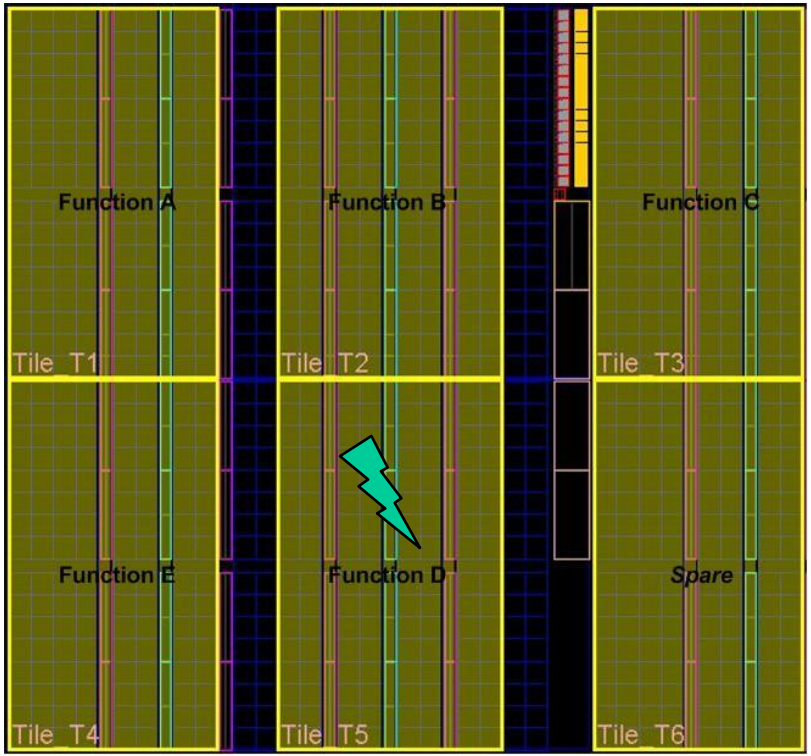


Configuration #6

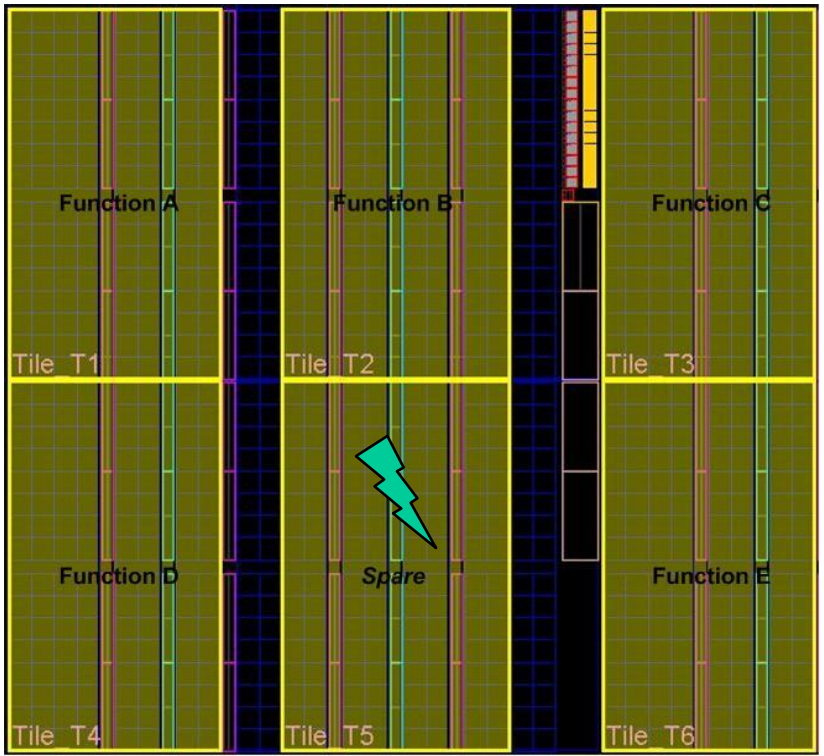
DTRS: Fault Recovery

- **CED is used to determine the location of the fault**
- **If location of fault is not known, then *"Blind Reconfiguration"* has to employed to recover from faulty situation**
- **System remains online to identify suitable spare configuration**
- *Example: Assuming Configuration 1 is configured and a fault is observed in tile T5. Configuration 2 can be downloaded to avoid the fault as it implements the spare tile at location of tile T5.*

Fault Detected



Fault Recovered



DTRS: Sustainability

- **Restore the original pool of configurations – so as to sustain future failures**
- **The faulty tile is restored for the function(s) as in the original pool of configurations**

- *Example: Further failures cannot be tolerated, as all other configurations will articulate the fault. The original pool of configurations can be restored by successfully implementing the function D in the presence of the fault in tile T5 (best case scenario)*

- *Additional Example illustrating the worst case scenario: Assuming initially Configuration 1 is configured and a fault is observed in tile T4. In this unique case, to successfully restore the original pool of configurations, all functions will have to be implemented in tile T4.*

DTRS: Sustainability

- **GA-based technique is used to implement the desired function(s) in the presence of hard fault(s) for a given tile**
- **Dynamic partial reconfiguration feature of FPGAs is to be used, so as to maintain good throughput during the refurbishment process.**
- **CED scheme, if implemented can be used in the fitness function of the GA**
- **Exhaustive or pseudo-exhaustive fitness evaluation can also be performed as introduced in [2]**

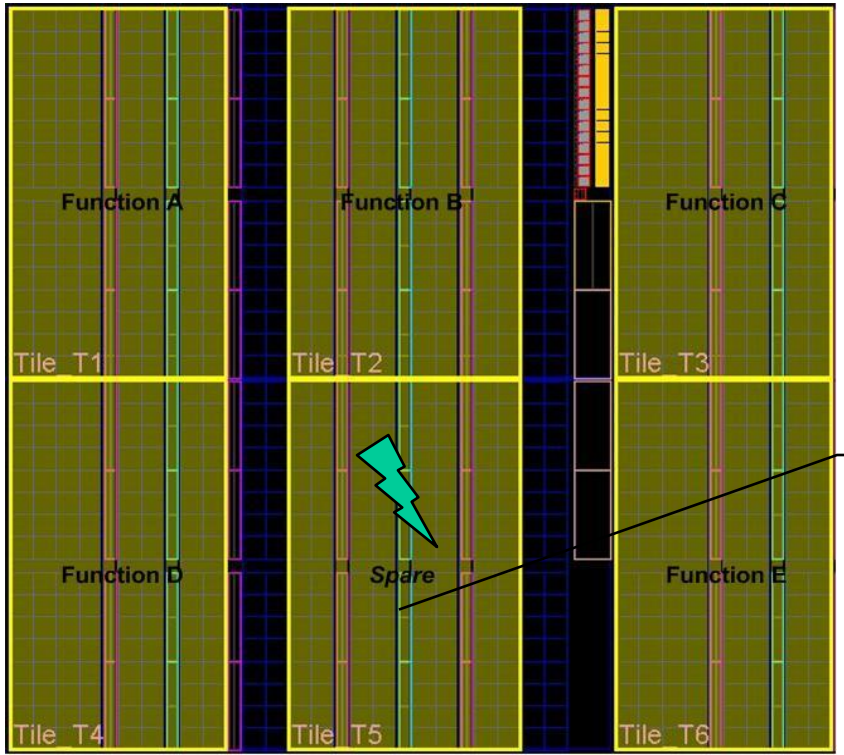


Figure illustrating **best case** scenario based on design time partitioning arrangements for the presented example

Evolve function D for tile T5 – using dynamic partial reconfiguration

DTRS: Sustainability

- **GA-based technique is used to implement the desired function(s) in the presence of hard fault(s) for a given tile**
- **Dynamic partial reconfiguration feature of FPGAs is to be used, so as to maintain good throughput during the refurbishment process.**
- **CED scheme, if implemented can be used in the fitness function of the GA**
- **Exhaustive or pseudo-exhaustive fitness evaluation can also be performed as introduced in [2]**

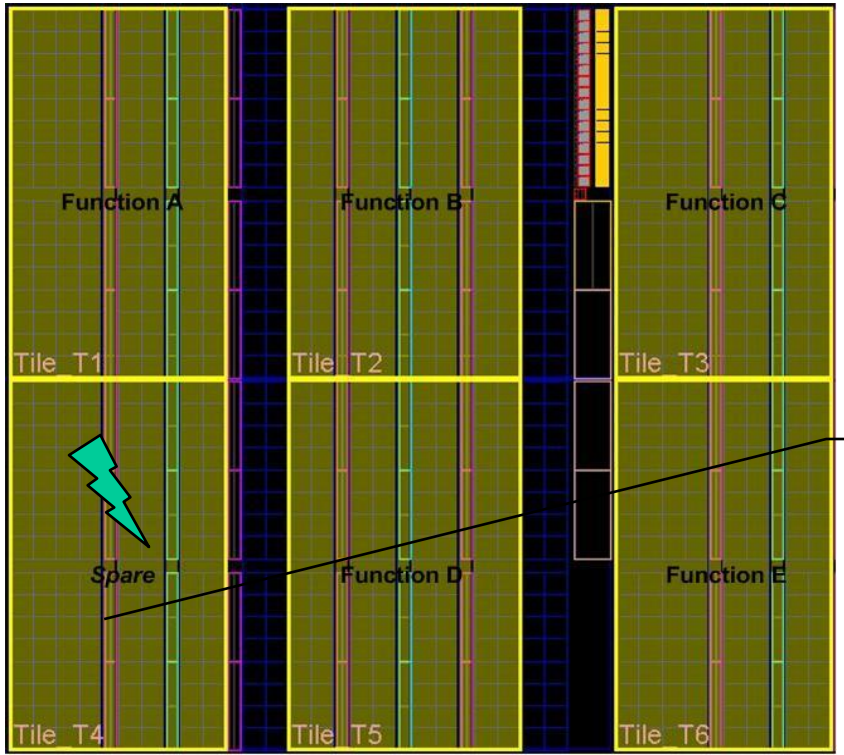


Figure illustrating **worst case** scenario based on design time partitioning arrangements for the presented example

Evolve functions A, B, C, D, and E for tile T4 – using dynamic partial reconfiguration

Objective of the experiments

Determination of a tractable tile size that can be refurbished by the GA in a reasonable amount of time

- **Objective 1: Investigate the time to refurbish as a function of number of LUTs in a tile**
- **Objective 2: Assess the dependence of fan-out of the tile on the refurbishment time**
- **Objective 3: Investigate the time to refurbish in terms of the number of faults present in a given tile.**
  - Multiple faults may arise due to aging of a tile implementing a sub-circuit with high switching activity as suggested in [4]

Simulation Setup

- **Experiments conducted with MCNC benchmark circuits [9]**
- **The benchmark circuits are mapped by employing a custom cell library supported by the modeled FPGA platform**
  - FPGA Mapping operation is performed using the abc tool [10]
- **Standard finite population GA is used with the settings as listed below.**

Parameter	Value
Population Size	20, 50
Mutation Rate	0.005
Crossover Rate	0.6
Tournament Size	5

- **Stuck-at faults are employed at the inputs of LUTs.**
  - This can be used to model interconnect failures
- **Bitwise comparison is performed to measure the fitness of individuals under refurbishment**
  - c17 benchmark circuit: The maximum numerical value of fitness for a refurbished configuration is 2^(5-bit input)* 2-bit output=64
- **GA stops when it achieves the maximum possible fitness or a preset fitness threshold level i.e. 95% throughput.**

Experimental Results Summary – Single Fault Experiments (100% refurbishment)

	c17	cm42a	3-to-8 decoder	cm85a	3x3 Multiplier	misex1	Z9sym
No. of LUTs	8	20	24	36	40	72	148
Max Fitness	64	160	64	6144	384	1792	512
Fitness after Fault	46	159	57	6120	327	1648	420
Avg. no. of Generations	105	529	169	113.1	1428	77297	226745.5
95% Confidence Interval	102 → 109	428 → 630	145 → 193	92.6 → 133.6	1018 → 1837	51129 → 103464	N/A
No. of runs	20	20	20	20	20	20	2

- *GA-based **100 %** refurbishment results under single fault scenario*

I/O Characteristics of benchmark circuits used

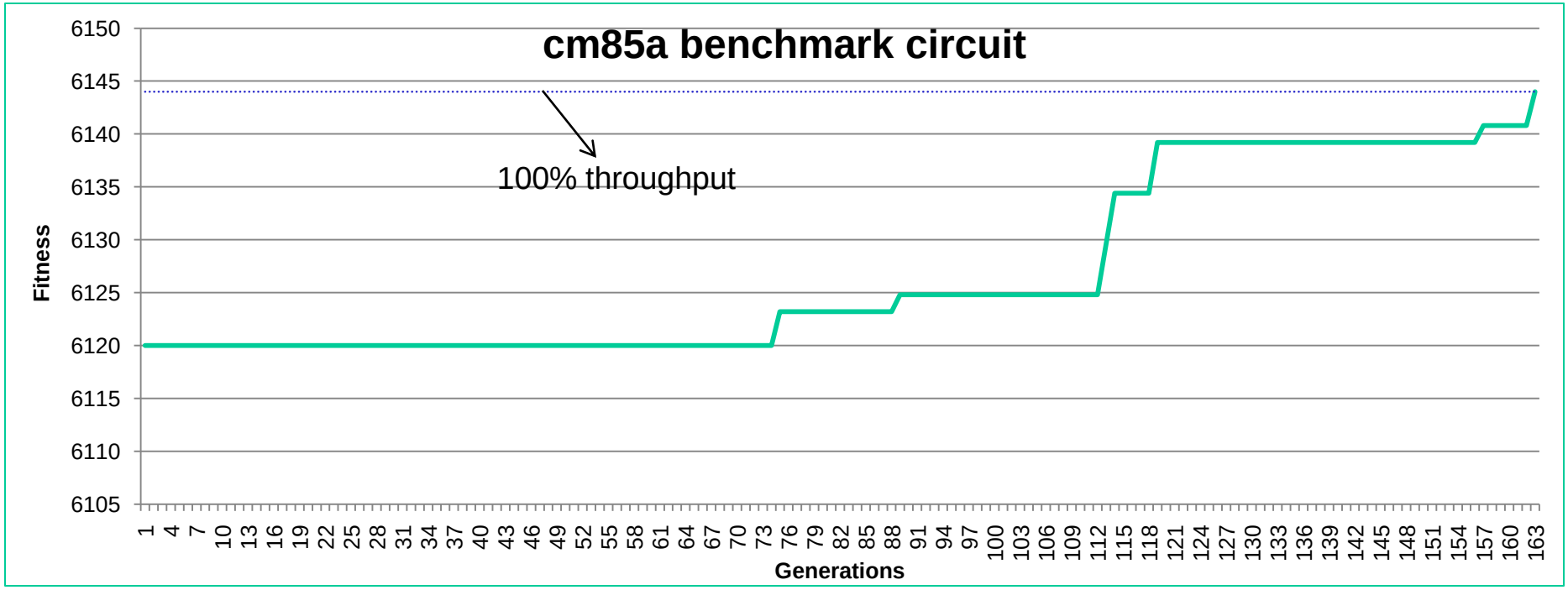
- `c17` (5 inputs, 2 outputs)
- `cm42a` (4 inputs, 10 outputs)
- `3-to-8 decoder` (3 inputs, 8 outputs)
- `cm85a` (11 inputs, 3 outputs)
- `3x3 multiplier` (6 inputs , 6 outputs)
- `misex1` (8 inputs, 7 outputs)
- `Z9sym` (9 inputs, 1 output)

Experimental Results Summary – Multiple Faults experiments (100% refurbishment)

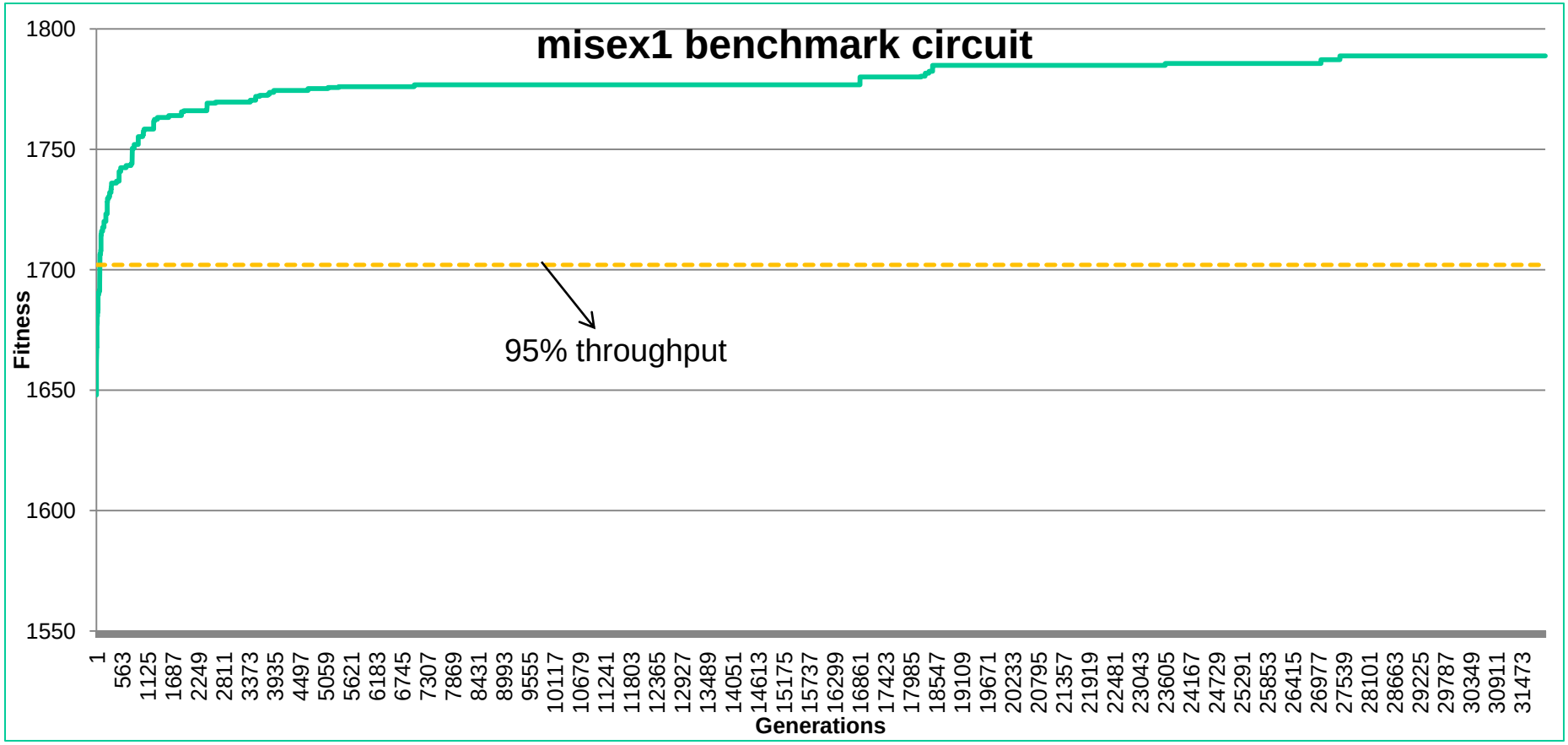
	3x3 Multiplier Logical Function			
No. of Faults	1	2	3	4
Fitness after Fault	327	282	239	247
Avg. no. of Generations	1428	18817	30963	71156
95% Confidence Interval	1018 → 1837	10839 → 26794	21382 → 40544	39934 → 102378
No. of runs	20	20	20	10

- *GA-based **100%** refurbishment results under multiple faults scenario*

Circuit Refurbishment: Fitness vs. Time



Circuit Refurbishment: Fitness vs. Time



Circuit Refurbishment: Fitness vs. Time

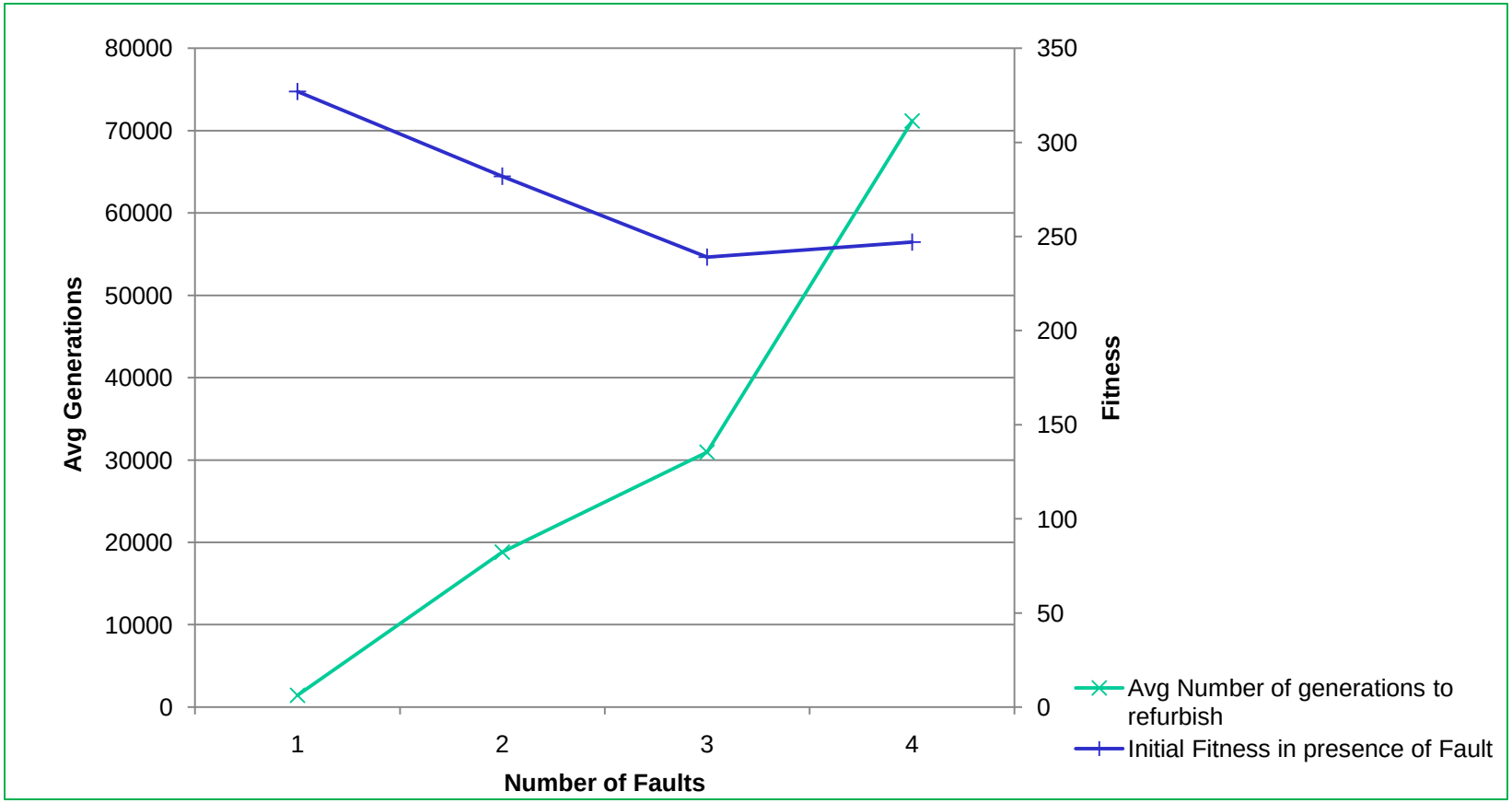
- Number of Generations required to find throughputs of 95% & 100% for various benchmarks under single fault scenario

	3x3 Multiplier		misex1		Z9sym	
Throughput	**95%**	**100%**	**95%**	**100%**	**95%**	**100%**
run 1	9	355	10	7773	3628	60195
run 2	76	370	4	16865	21203	393296
run 3	17	463	102	18472	37586	DNF*
run 4	29	585	169	23610	55714	DNF
run 5	5	690	77	27466	85230	DNF
run 6	22	783	39	56388	37946	DNF
run 7	19	794	12	57155	20304	DNF
run 8	48	820	11	68145	18154	DNF
run 9	25	1104	12	89996	114711	DNF
run 10	30	1115	10	105816	24445	DNF

95% achievable for moderately sized circuits

* DNF=Did Not Find Solution in Max Number of Generations of 450000

Circuit Refurbishment in the presence of multiple faults



Results Summary

- **Results indicate**
  - **high throughputs such as 95% can be achieved in relatively quick time as compared to achieving 100%, which may be sufficient for majority applications**
  - **the time to refurbish increases in the presence of multiple faults**
  - **increasing the fanout of a tile increases the time to refurbish**

References

[1] S. Vigander, Evolutionary Fault Repair of Electronics in Space Applications, in *Dept. of Computer & Information Science*, Dissertation, Norwegian University of Science and Technology, Trondheim, Norway, 28 February 2001.

[2] Ronald F. DeMara, Kening Zhang, Carthik A. Sharma, "Autonomic fault-handling and refurbishment using throughput-driven assessment", Applied Soft Computing, Vol. 11, Issue 2, pp. 1588-1599, March 2011.

[3] R. S. Oreifej, R. N. Al-Haddad, H. Tan, R. F. DeMara, "Layered Approach To Intrinsic Evolvable Hardware Using Direct Bitstream Manipulation Of Virtex II Pro Device," in Proceedings of the 17th International Conference on Field Programmable Logic and Applications (FPL'07), Amsterdam, Netherlands, 27-29 August 2007.

[4] Suresh Srinivasan, Krishnan Ramakrishnan, Prasanth Mangalagiri, Yuan Xie, Vijaykrishnan Narayanan, Mary Jane Irwin, Karthik Sarpatwari, "Towards Increasing FPGA Lifetime", IEEE Trans. Dependable and Secure Computing. vol. 5, Issue 2, pp. 115-127, 2008.

[5] Xin Wu, Prabhuram Gopalan, Greg Lara, "Xilinx Next Generation 28 nm FPGA Technology Overview", Xilinx white paper, wp312, March 26, 2011.

[6] Wenjing Rao, Chengmo Yang, Karri, R., Orailoglu, A.,"Toward Future Systems with Nanoscale Devices: Overcoming the Reliability Challenge", IEEE Computer magazine, vol. 44, Issue 2, pp. 46-53, Feburary 14, 2011.

[7] Carl Carmichael, Chen Wei Tseng, "Correcting Single-Event Upsets in Virtex-4 FPGA Configuration Memory", Xilinx application note, xapp1088, October 5, 2009.

[8] S. Mitra, W. Huang. N.R. Saxena, S. Yu and E.J. McCluskey, “Reconfigurable Architecture for Autonomous Self-Repair,” IEEE Design & Test of Computers, Special Issue on Yield & Reliability, Vol. 21, Issue 3, pp. 228-240, May-June 2004.

[9] S. Yang, "Logic Synthesis and Optimization Benchmarks, Version 3.0," Tech. Report, Microelectronics Center of North Carolina, 1991.

[10] Berkeley Logic Synthesis and Verification Group, University of California, Berkeley, “ABC: A System for Sequential Synthesis and Verification”, www.eecs.berkeley.edu/~alanmi/abc.

[11] J. F. Miller, P. Thomson, and T. Fogarty., "Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study," in *Algorithms and Evolution Strategy in Engineering and Computer Science*, D. Quagliarella, J. Periaux, C. Poloni, and G. Winter, Eds. Chichester, England, 1998, pp. 105-131.