Karlsruhe Institute of Technology

Institut für Technische Informatik
Chair for Embedded Systems – Prof. Dr. J. Henkel

Vorlesung im SS 2016

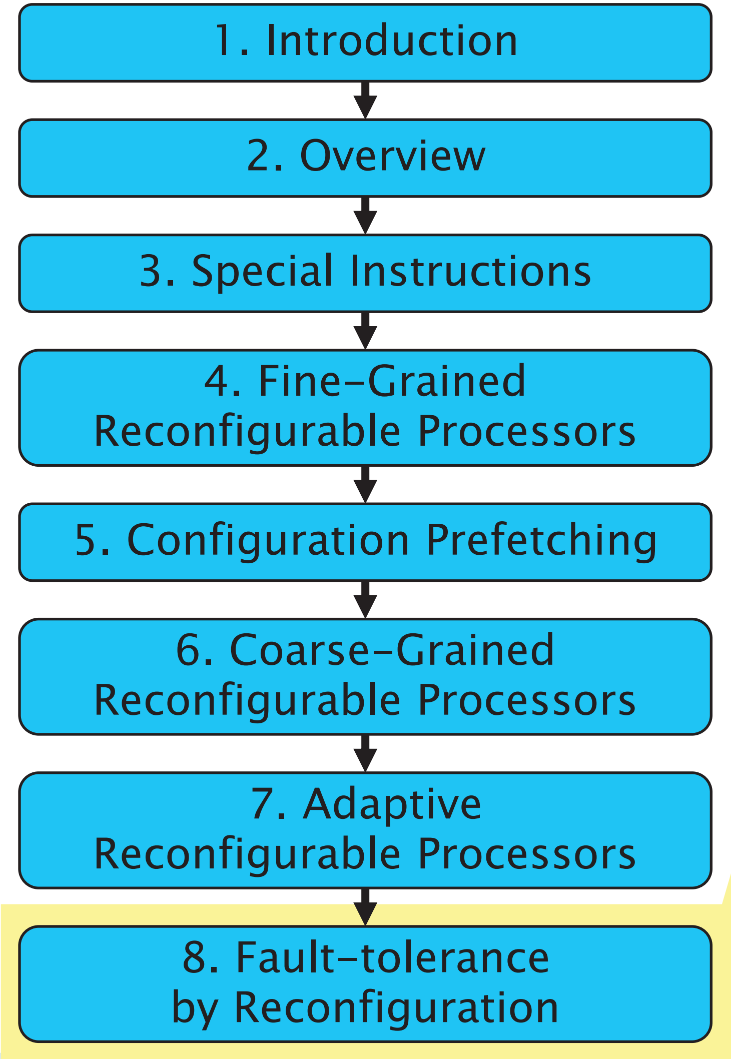
Reconfigurable and Adaptive Systems (RAS)

Marvin Damschen, Lars Bauer, Artjom Grudnitsky, Hongyan Zhang, Jörg Henkel

Reconfigurable and Adaptive Systems (RAS)

8. Fault Tolerance and Reliability in FPGA based Systems

RAS Topic Overview



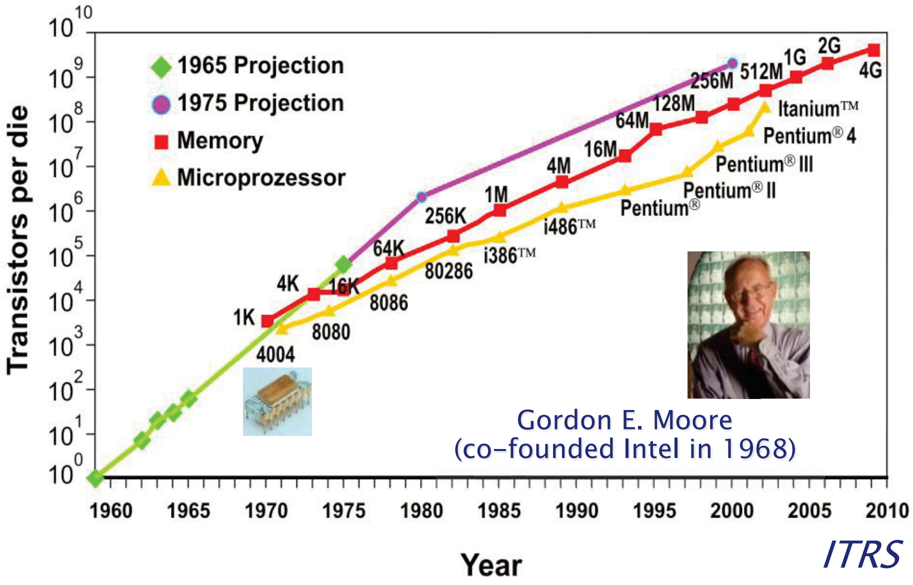
1. Introduction
2. Overview
3. Special Instructions
4. Fine-Grained Reconfigurable Processors
5. Configuration Prefetching
6. Coarse-Grained Reconfigurable Processors
7. Adaptive Reconfigurable Processors
8. Fault-tolerance by Reconfiguration

- Introduction
- Fault Detection and Mitigation Techniques
- Reliability for LHC
- Scrubbing
- Reliability in Space
- OTERA:
    - Online test
    - Stress balancing for aging mitigation
    - Adaptive Redundancy

8.1 Introduction

Why Fault Tolerance?

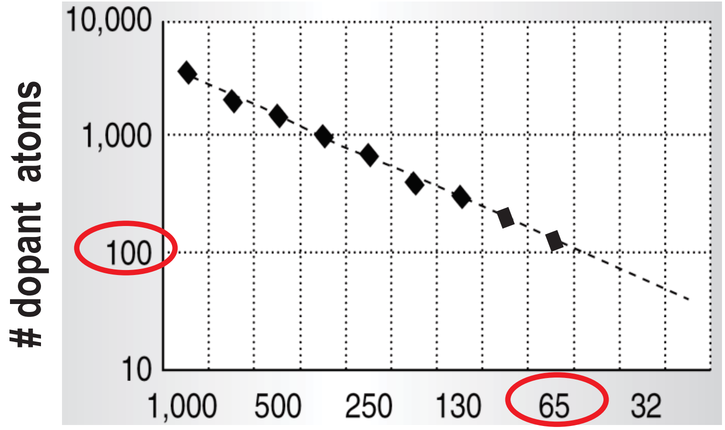
- CMOS Scaling increases occurrence of
  - Manufacturing defects
  - Post-deployment degradation
  - Especially important for FPGAs as they have a high amount of transistors and interconnect wires
- Environmental conditions can incur temporary faults
  - E.g. Aerospace industry – use hardened devices for mission critical tasks, FPGAs for non-critical data processing
- Unlike ASICs, FPGAs can adapt to deal with permanent and temporary faults



Gordon E. Moore
(co-founded Intel in 1968)

ITRS

# of dopant atoms in Transistor-channel

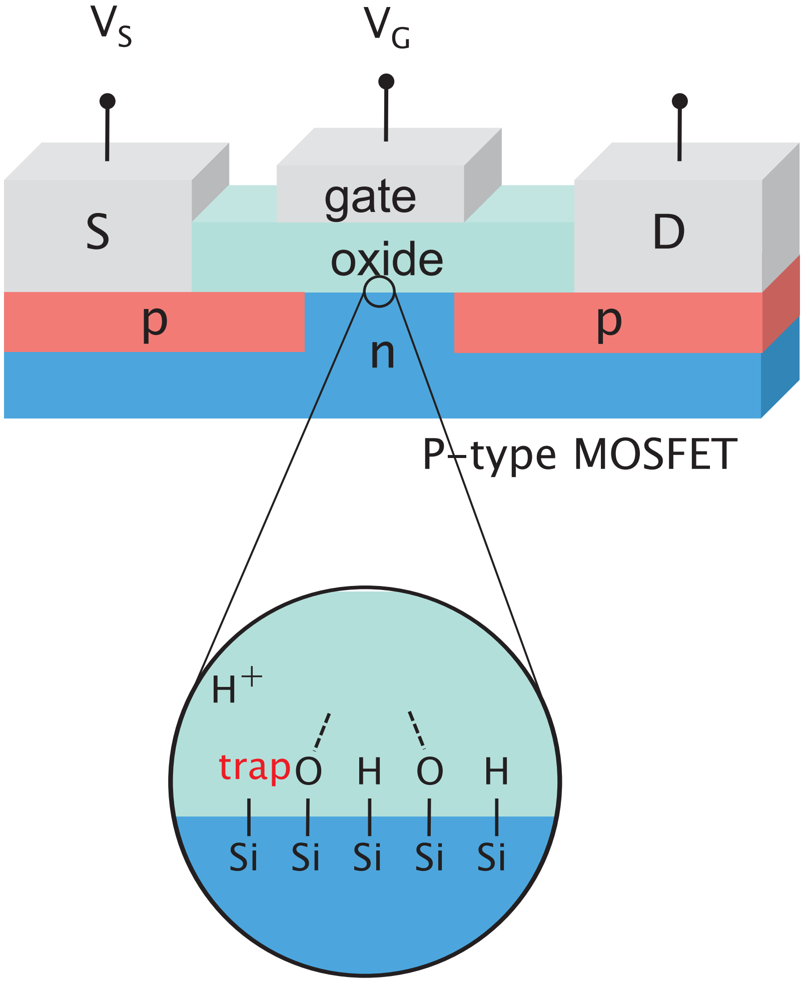


Types of Faults

- Permanent Faults: e.g. stuck-at failures in CLBs and opens, bridges, shorts in the programmable switching matrix
  - Could occur during the fabrication process without being detected
  - Damage of device resources may also appear in the life cycle of FPGAs
- Intermittent Faults: have a permanent cause in the structure of the circuit but their effect is intermittent, e.g. depending on temperature or power consumption
- Transient Faults: have a temporary cause that can alter signal values or state stored in memory cells, which creates indefinite and incorrect states in the computation
  - E.g. by a high energy particle strike resulting in an energy exchange and charge displacement

Negative Bias Temperature Instability (NBTI)

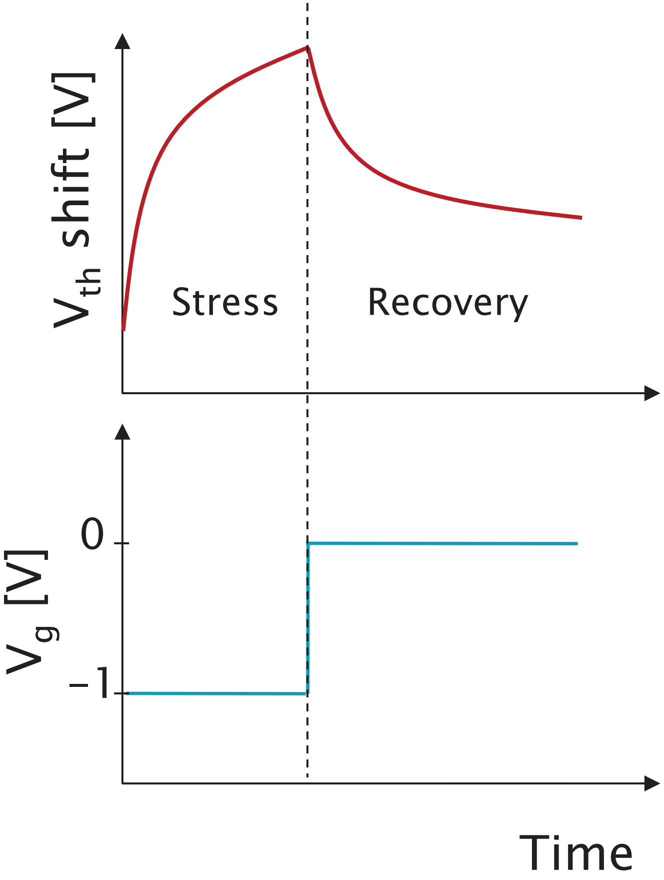
- Breakdown of Si-H bonds at the silicon-oxide interface due to voltage/thermal stress → causes interface traps
- Affects mostly P-MOSFETs because of negative gate bias
  - Effect in N-MOSFETS is negligible
- Despite research focus: NBTI is observed, but not yet fully understood



$V_{GS} < 0$ → STRESS!

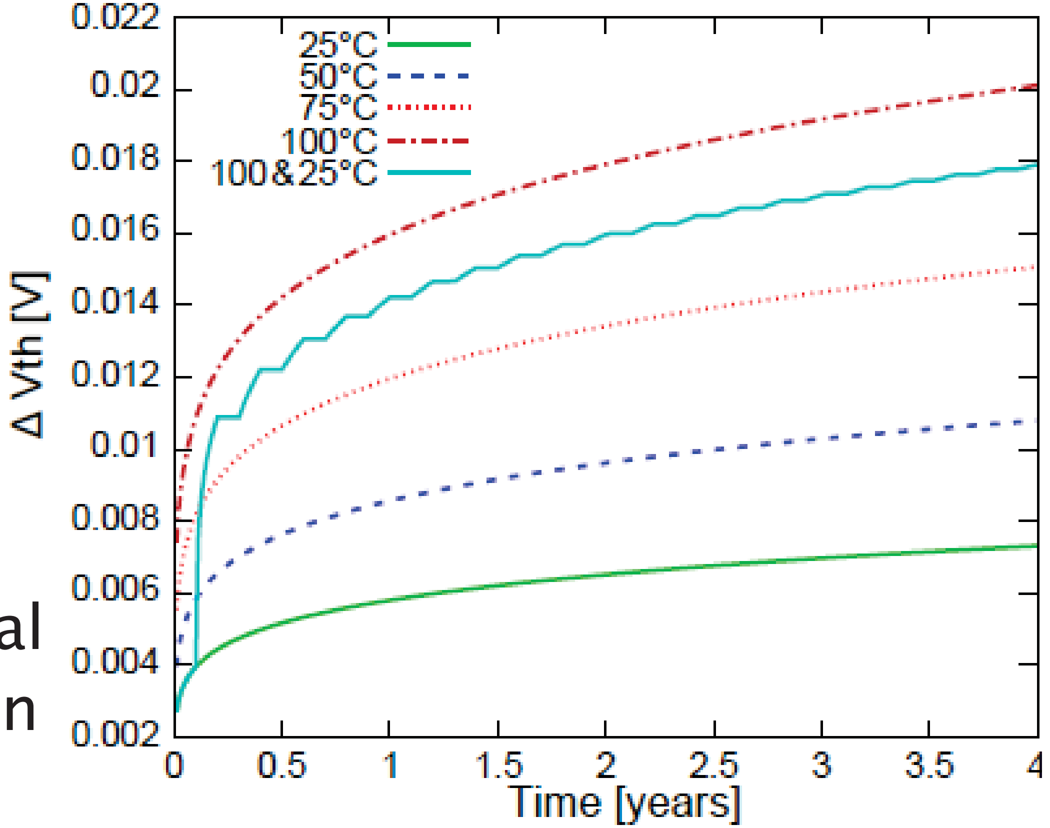
Negative Bias Temperature Instability (NBTI) (cont‘d)

- NBTI manifests itself as a shift in V_{th}
  - Causes increase in transistor delay
  - NBTI leads to delay faults and resulting circuit failure
- Recovery effect in periods of no stress
  - When voltage and temperature are low, V_{th} can shift back towards its original value
  - Full recovery from a stress period only possible in infinite time → In practice, overall V_{th} shift increases over longer periods, e.g. months or years



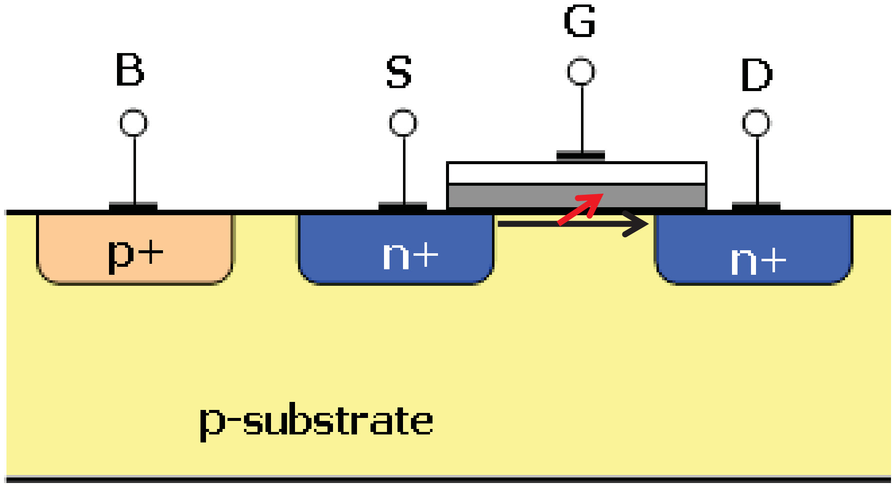
NBTI and Temperature

- Temperature plays important aspect in NBTI modeling
- Higher temperatures increase shift in threshold voltage
- ΔVth approximately 50% higher at 75°C than 55°C
- NBTI effect at 75°C is approximately equal to alternating between 85°C and 25°C



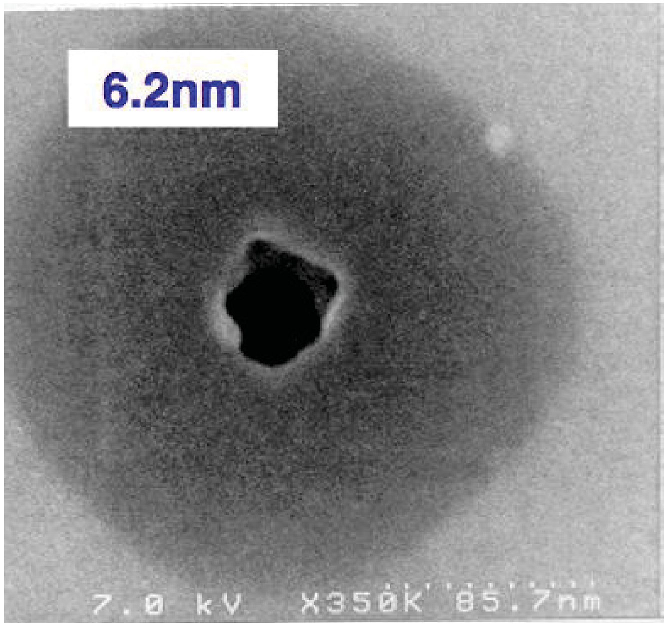
Types of Degradation (cont'd)

- Hot-Carrier Injection (HCI)
  - build up of trapped charges in the gate-channel interface region
  - progressive reduction of carrier mobility → increase in CMOS threshold voltage
  - Switching speed slower, leads to timing problems

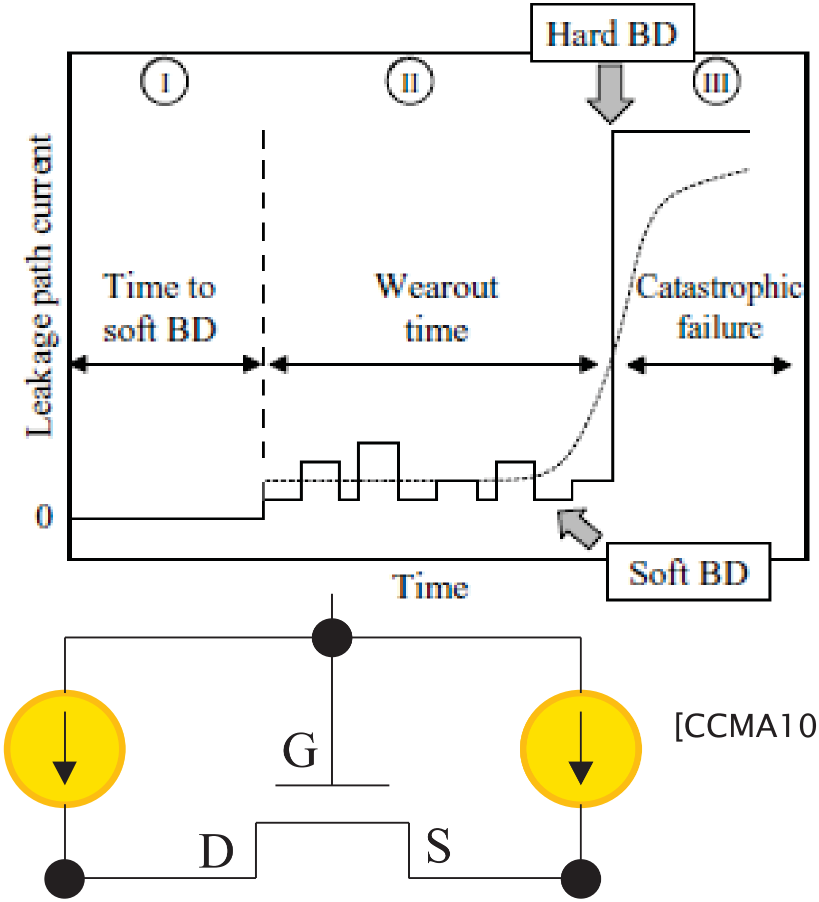


Types of Degradation (cont‘d)

- Time-Dependent Dielectric Breakdown (TDDB): over time conducting path forms in thin oxide layers



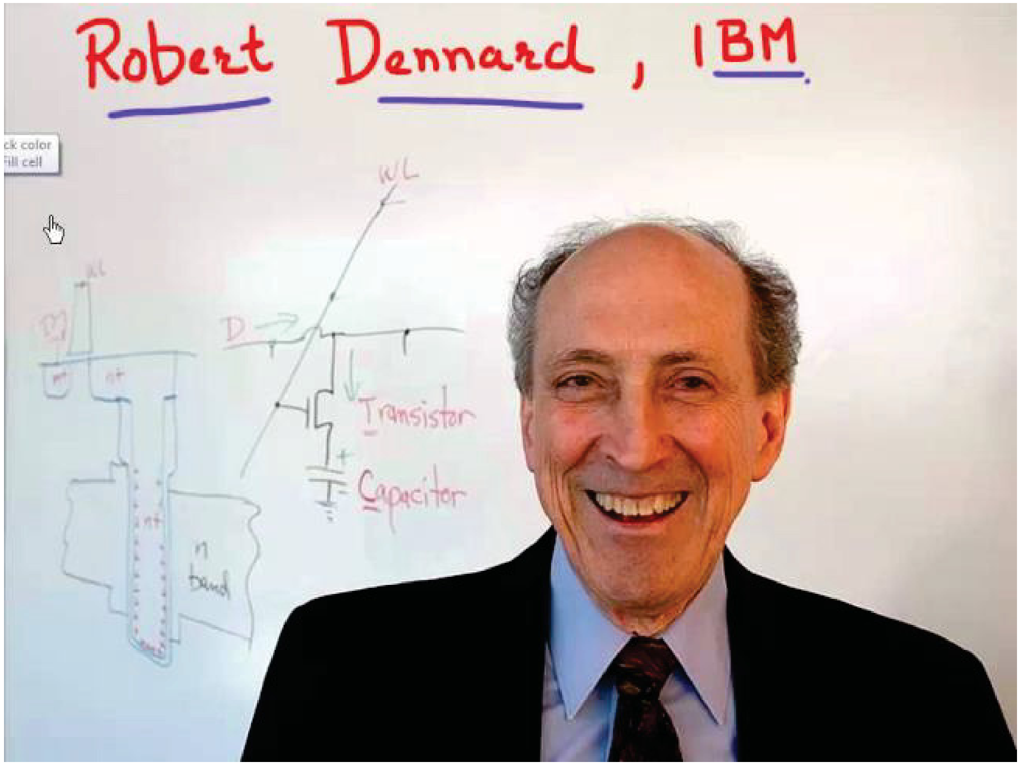
src: Stathis, IRPS (2008)



[CCMA10]

Main Reason for many of these effects: High-Fields

- Most of device problems can be tracked down to high-field effects
  - Nowadays in every new semiconductor technology: Structures getting significantly smaller; voltage nearly constant
- Failure to follow Dennard Scaling (next slide)



Dennard Scaling vs. Power Density

- Transistor and power scaling are no longer balanced
  - Scaling is limited by power
- Higher power density leads to thermal problems
  - Accelerates aging effects

Classical scaling (Dennard)

Device count	S^2	Assuming a constant area
Device frequency	S	Chip freq. may reduce due to wire delay
Device power (cap)	$1/S$	
Device power (V_{dd})	$1/S^2$	Voltage scales $1/S$ → Power squared
Power Density	**1**	[W/mm^2]

S: Scaling Factor; Device: Transistor

src: G. Venkatesh et al., “Conservation Cores: Reducing the Energy of Mature Computations”, ASPLOS ‘10

Dennard Scaling vs. Power Density

- Transistor and power scaling are no longer balanced
  - Scaling is limited by power
- Higher power density leads to thermal problems
  - Accelerates aging effects

Classical scaling (Dennard)

Device count	S^2
Device frequency	S
Device power (cap)	$1/S$
Device power (V_{dd})	$1/S^2$
Power Density	**1**

S: Scaling Factor; Device: Transistor

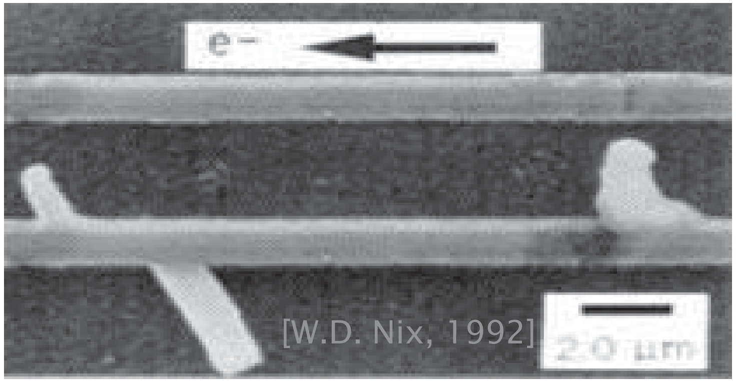
Power Limited Scaling

Device count	S^2
Device frequency	S
Device power (cap)	$1/S$
Device power (V_{dd})	~1
Power Density	**S^2**

src: G. Venkatesh et al., “Conservation Cores: Reducing the Energy of Mature Computations”, ASPLOS ‘10

Types of Degradation (cont‘d)

- Electromigration: thermally activated metal ions may leave their potential wells
  - electric field and momentum exchange through electrons direct metal ion migration
  - can lead to open/short circuits

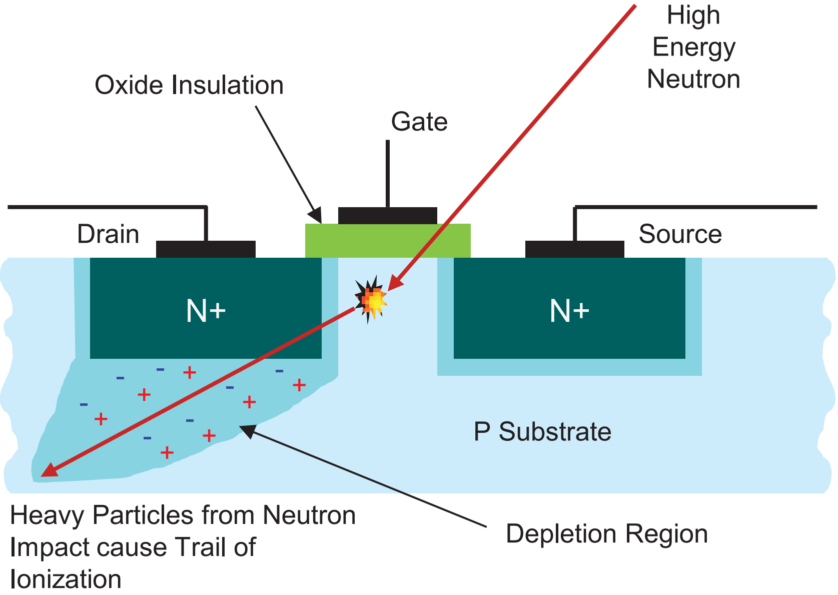


[W.D. Nix, 1992]

[wikipedia]

Radiation induced faults

- Radiation induced faults
  - Single Event Upsets (SEUs)/Single Event Transients
  - Most common: single bit flip in SRAM cell
  - SEU effect on ASIC
    - Transient (only variation is time duration of fault)
    - Even if latched, will be eventually overwritten
  - SEU effect on FPGAs
    - Permanent (until reset/reconfiguration) if configuration memory hit by SEU

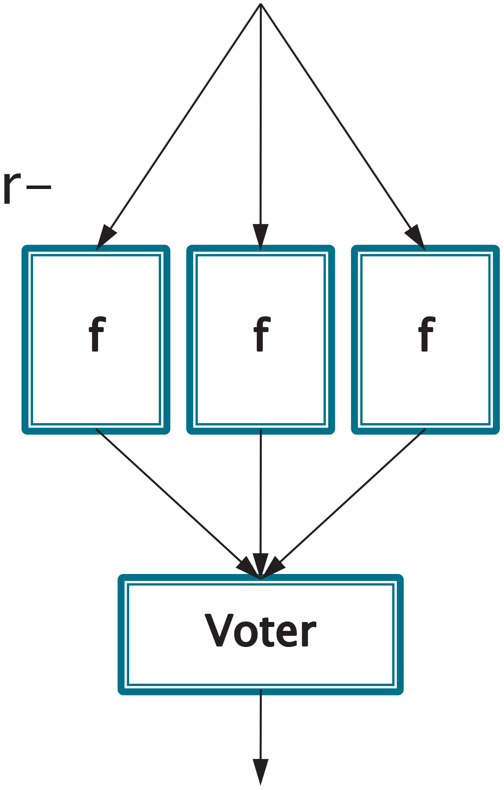


src: John P. Bendekovic, Director Actel Mil/Aero Sales

8.2 Fault Detection and Mitigation Techniques

Modular Redundancy

- Masks errors, but does not correct underlying fault
  - Problem: error accumulation
- External
  - Multiple FPGAs working in lockstep, i.e. performing the same operation in each cycle
  - Output sent to radiation hardened voter
- Internal
  - Replicate functional block in FPGA
- Popular configurations
  - Triple Modular Redundancy (TMR)
  - Duplication with Comparison (DWC)



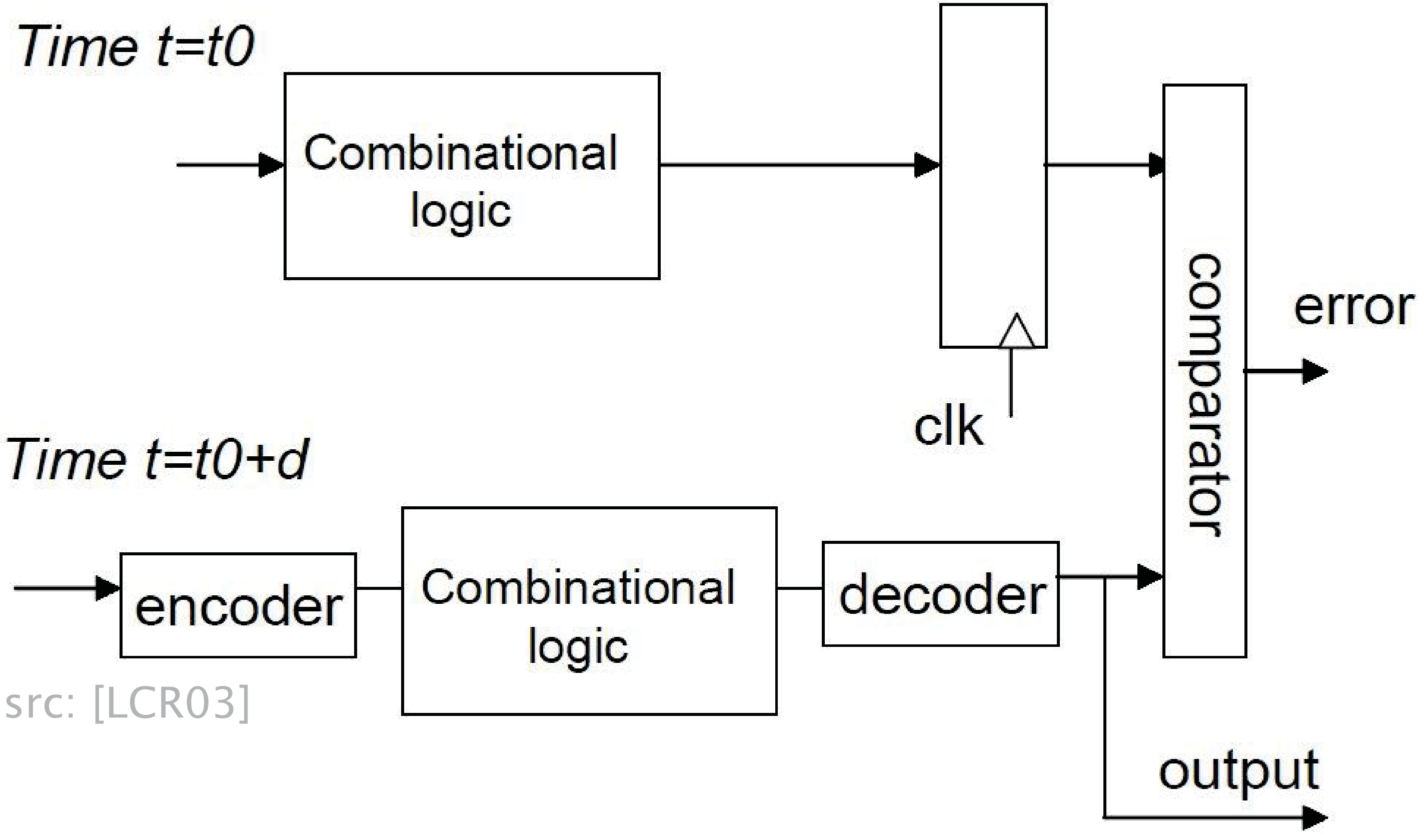
Fault detection methods comparison

src: [SCC08]	Detection Speed	Resource Overhead	Performance Overhead	Granularity	Coverage
Modular Redundancy	Fast: as soon as fault is manifest	Very large: triplicate + voter	Very small: Voter delay	Coarse: protect module sized blocks	Good: All manifest errors detected

Concurrent Error Detection

- More space efficient design than modular redundancy
- Error coding algorithms (e.g. parity) at data flows/stores
- Time redundancy can be used for concurrent error detection
  - Repeat computation in a way that allows errors to be detected
  - First computation at t0: compute result in combinational logic, store result
  - Second computation at t0+d: encode operands, compute in combinational logic, decode result, compare to first result

Concurrent Error Detection (cont'd)



src: [LCR03]

Concurrent Error Detection (cont'd)

- Different techniques for encode/decode, e.g. bit inversion to detect stuck-at faults
- Recomputation with shifted operands (RESO) for faulty arithmetic slices
  - Encode: left shift operands
  - Decode: right shift result
- Combine with Duplication with Comparison (DWC)
  - RESO determines which module is faulty, DWC uses result of other module
  - Less area required than TMR
  - Slightly slower (time-shifted re-computation)

Fault detection methods comparison

src: [SCC08]	Detection Speed	Resource Overhead	Performance Overhead	Granularity	Coverage
Modular Redundancy	Fast: as soon as fault is manifest	Very large: triplicate + voter	Very small: Voter delay	Coarse: protect module sized blocks	Good: All manifest errors detected
Concurrent error detection	Fast: as soon as fault is manifest	Medium: tradeoff with coverage	Small: CRC logic delay	Medium: tradeoff with resource	Medium: Not practical for all types of functionality

Offline BIST

- Built-in Self-Test: does not use external test equipment
- In FPGAs: test configurations containing
  - Test pattern generator (TPG)
  - Output response analyzer (ORA)
  - Between them: Device (i.e. logic and interconnect) under test (DUT)
- Can test for faults that are difficult to cover in online tests, e.g. clock network
- Major drawback: system must enter dedicated test mode

Fault detection methods comparison

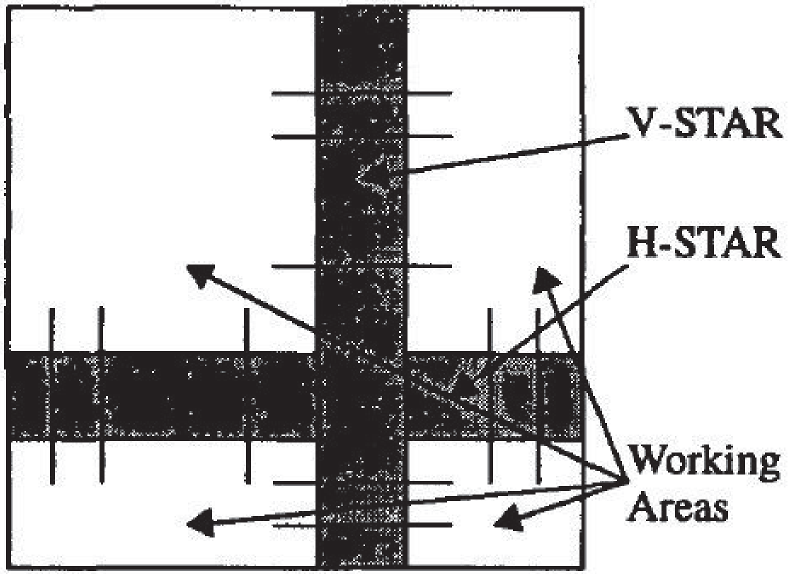
src: [SCC08]	Detection Speed	Resource Overhead	Performance Overhead	Granularity	Coverage
Modular Redundancy	Fast: as soon as fault is manifest	Very large: triplicate + voter	Very small: Voter delay	Coarse: protect module sized blocks	Good: All manifest errors detected
Concurrent error detection	Fast: as soon as fault is manifest	Medium: tradeoff with coverage	Small: CRC logic delay	Medium: tradeoff with resource	Medium: Not practical for all types of functionality
Offline BIST	Slow: only when offline	Very small	Small: start-up delay	Fine: possible to detect exact error	Very good: All faults including dormant

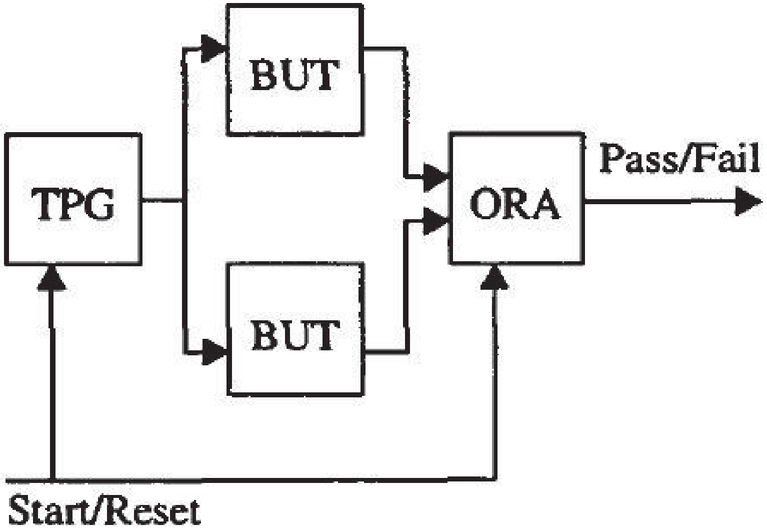
Online BIST

- Split FPGA into equal-sized regions
- One region performs self-test, others perform design function
- When test complete: swap test region with untested functional region and test the new region
- Lower area overhead (1 region + controller logic)
- Problems:
  - swapping may "stretch" connections between regions
    → slower timing (may require clock speed reduction)
  - Functional blocks may be inoperable during swap (depends on how it is implemented)

Self Testing AReas (Roving STARs)

- STARs consist of tiles performing BIST
  - STARs rove over FPGA left↔right (H-STAR) and up↔down (V-STAR)
  - Test Pattern Generator (TPG) sends data to Block under test (BUT); Output Response Analyzer (ORA) detects fault





src: [ESSA00]

Roving STARs

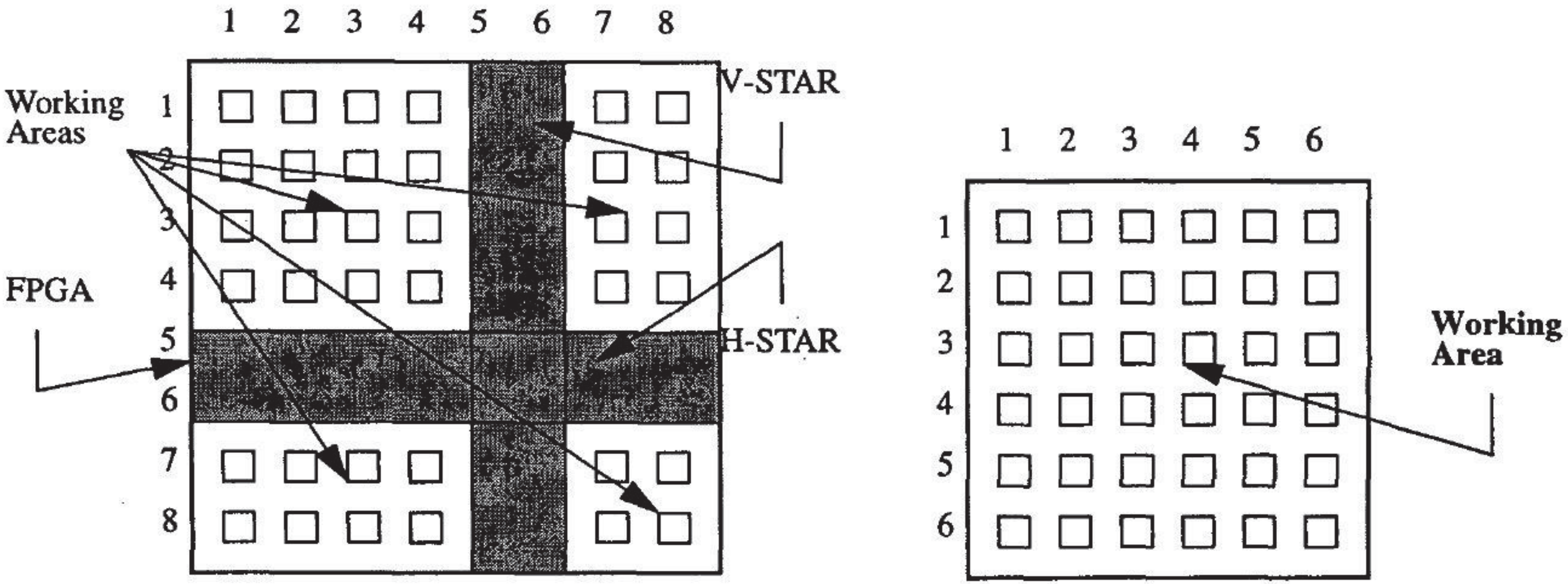
- Roving controlled by embedded processor
- Blocks under Test tested in different configurations, e.g. User RAM, LUT, adder, etc.
- Test strategy does not use signature analysis, but tests 2 identically configured blocks and compares response
  - Each block in tile tested twice with different partner block

TPG	BUT
BUT	ORA

- H-STAR 2 rows high, V-STAR 2 columns wide
  - Tiles not necessarily 2x2, can also be 2x3, etc.

Roving STARs

- Depending on current location of STARs, working area of FPGA is divided into 1, 2 or 4 regions
  - Virtual coordinate system of working area without STARs



src: [ESSA00]

Roving STARs

- Model: FPGA system function composed by “logic cell functions”
  - Each fits into 1 Configurable Logic Block (CLB) on the FPGA
  - “Logic cell functions” defined by coordinates in virtual coordinate system
  - CLBs defined in physical coordinate system
  - Mapping depends on the position of the STAR
- Blocks can be faulty, partially usable, fault-free
  - Partially faulty blocks can implement some, but not all logic cell functions
  - STARs test blocks in different modes and can determine which mode is fault-free

Roving STARs

- Fault Tolerance approach, 3 Steps:
  I. STAR Parking: when fault detected, STAR that detected fault stops moving. User application notified for possible rollback. Determine fault and report to controller
  II. Reconfigure system function: if logic cell can use block (usable or sufficiently partially usable), do not reconfigure. Otherwise remap logic cell to spare working block. Remapping performed by controller while STARs parked. When done: STARs continue roving
  III. STAR stealing: when no spares are available, take out part of the STARs and use them as spares. Tiles may no longer be able to perform BIST. Try to maintain at least 1 roving STAR

Fault detection methods comparison

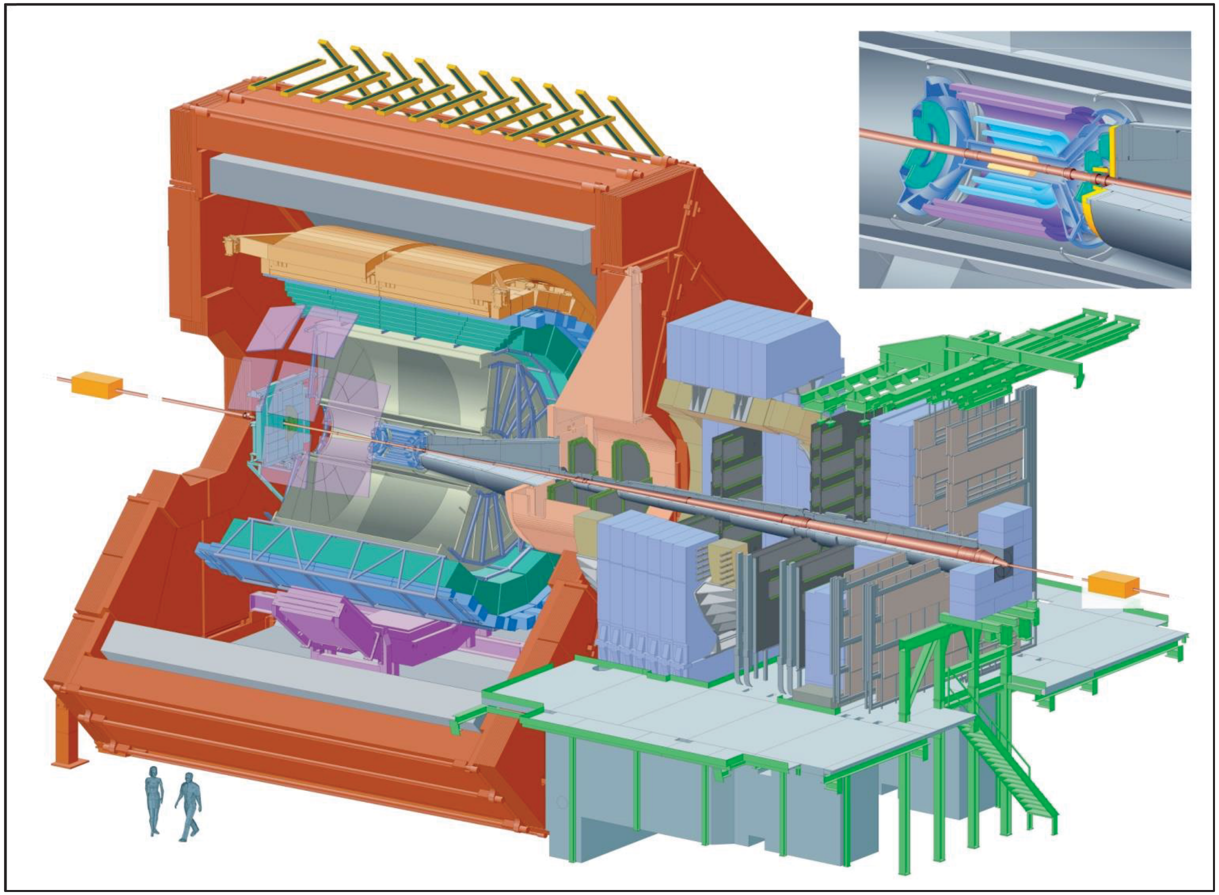
src: [SCC08]	Detection Speed	Resource Overhead	Performance Overhead	Granularity	Coverage
Modular Redundancy	Fast: as soon as fault is manifest	Very large: triplicate + voter	Very small: Voter delay	Coarse: protect module sized blocks	Good: All manifest errors detected
Concurrent error detection	Fast: as soon as fault is manifest	Medium: tradeoff with coverage	Small: CRC logic delay	Medium: tradeoff with resource	Medium: Not practical for all types of functionality
Offline BIST	Slow: only when offline	Very small	Small: start-up delay	Fine: possible to detect exact error	Very good: All faults including dormant
Roving STARs	Medium: order of 1 second	Medium: empty test block + controller	Large: stop clock to swap blocks. Critical paths may lengthen	Fine: possible to detect exact error	Very good: multiple manifest and latent faults detected

Other Fault repair techniques

- Column/Row shifting: spare lines of cells at end of array
  - When error detected in row/col → bypass whole row/col via multiplexers and use spare
- Alternative configurations: split FPGA into tiles such that multiple configurations for each tile implement same functionality
  - Once error located, load configuration that does not use faulty resource
- Others: online re-routing, …

8.3 Reliability for LHC

ALICE – A Large Ion Collider Experiment



src: CERN, ALICE Set Up, http://aliceinfo.cern.ch/Public/Objects/Chapter2/ALICE-SetUp-NewSimple.jpg

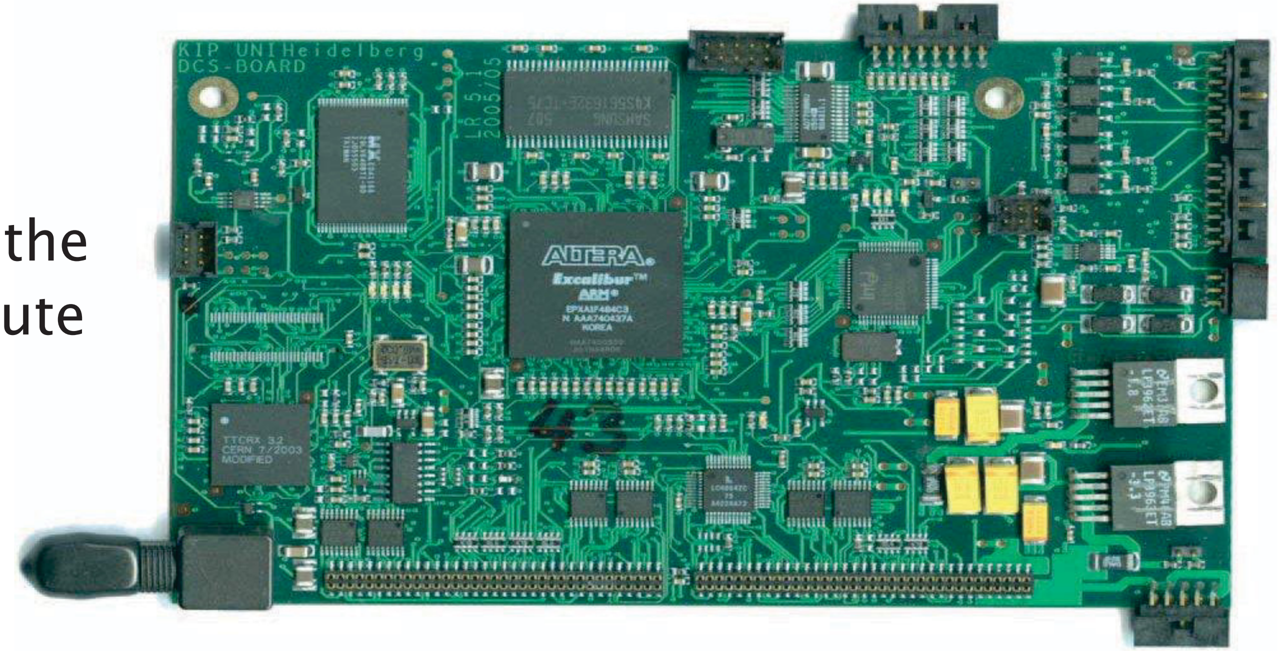
- One of the experiments using the Large Hadron Collider (LHC) at CERN
- Task: Characterize quark gluon plasma produced through collisions of heavy ions
- Transition Radiation Detector (TRD) identifies fast electrons in central barrel
  - Consists of 540 readout chambers

Detector Control System (DCS)

- Task: ensure safe operation of TRD
  - Provide front-end electronics with configuration and calibration data
- Some Design Goals from the Design Report:
  - Coherent and homogeneous: to allow for integration of independently developed components
  - Flexible and scalable: e.g. hardware upgrades, procedural changes
  - Must be operational throughout lifetime of experiment, even during shutdown phases
  - …

Detector Control System (DCS)

- DCS Board
  - Developed at the Kirchoff Institute of Physics (Heidelberg)



src: [K08]

- Several variants for different components of the detector, but using FPGA allows using same board layout
  - Interface with front end electronics in readout chambers – 540 boards
  - Low/high voltage power control & trigger control – 50 boards
  - Control & configure readout control units (which pass measurement data to data acquisition systems) – 216 boards

DCS Board – Hardware

- Altera Excalibur FPGA
  - SRAM based
  - 4190 Logic Elements (about 100k gates)
  - Embedded ARM 9 processor
- MMU, SDRAM Controller, UART, watchdog, etc
- 32 MB SDRAM, 8 MB Flash (FPGA configuration data, bootloader, software)
- ARM's Advanced High Performance Bus (AHB) used for on-board interconnect
- Ethernet (↔ PC), LVDS (↔ front end electronics)

DCS Board – Flash Reconfiguration

- If a board fails to start up (e.g. flash image corrupted by radiation), it can be reconfigured from a neighbor board
  - Boards connected in a ring in addition to Ethernet
  - Accessible via JTAG
  - Special FPGA Configuration that receives data over Ethernet and writes it to flash → bypasses CPU and reduces reconfiguration time

DCS Board Radiation Tolerance

- FPGA has more potential points of failure than a dedicated ASIC Controller
  - But: also more mechanisms to deal with such faults
- Expected: no permanent damage to hardware, only Single Event Upsets (SEU) in memory/registers
- Radiation tests at level of radiation expected in detector: 1 SEU every few hours <u>per board</u>

DCS Test Mechanisms

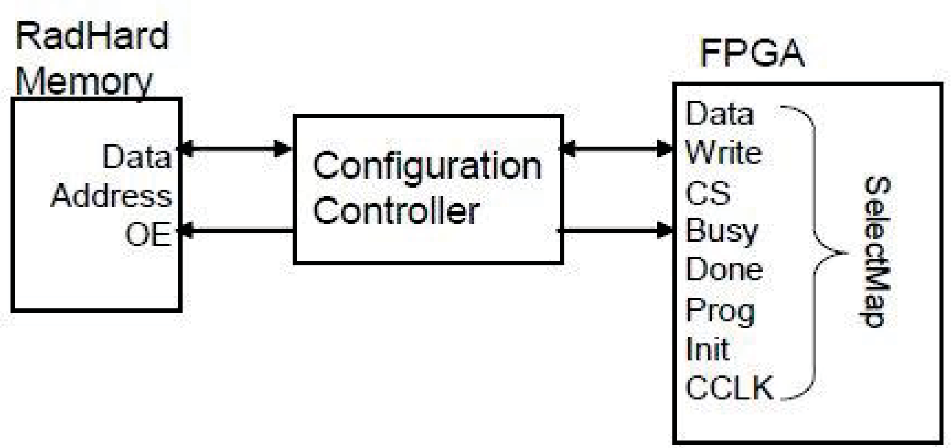
- SDRAM: fill memory with pattern, read out and verify, send UDP packet via network on error
  - CPU not used, OS not needed → 100% of memory can be tested
- FPGA Configuration SRAM
  - Triple modular redundancy + majority voter detect functional error
  - No readback of configuration data possible with this FPGA
  - Find configuration error by testing TMR functionality
- SDRAM and SRAM Tests can be used to estimate radiation susceptibility – not used in regular operation
- Online Memory Self-Test
  - Fill unused memory with test patterns and verify
  - Implemented as kernel module

8.4 Scrubbing

Scrubbing

- Repair faults in configuration memory by updating affected configuration frame
- For Xilinx FPGAs there are 3 ways to access configuration memory: JTAG (slow external), SelectMAP (fast external), ICAP (fast internal)
- Scrubbing protects only configuration data, not memory elements
  - Can not scrub LUTs that are used as User RAM ("Distributed RAM")
  - Ca not scrub BlockRAM (embedded memory in FPGAs)
  - Use other protection schemes for memory elements, e.g. parity or error-correcting codes

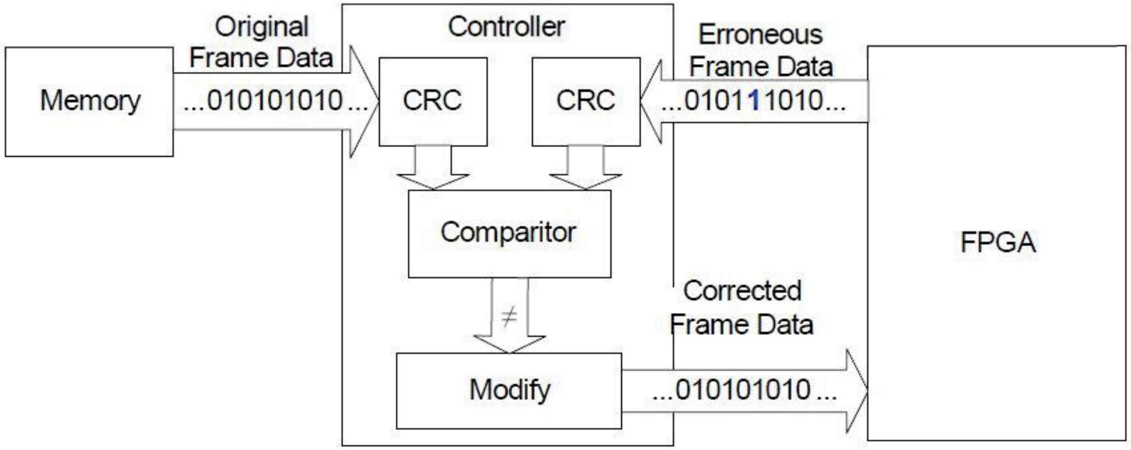
Blind Scrubbing



src: [HSWK09]

- Strategy: Continuous overwriting
  - Read original configuration frame from external memory
  - Write it to FPGA, even if no SEUs present
- Advantages: Simple implementations, minimal additional hardware, fast repair

Readback Scrubbing



src: [HSWK09]

- Strategy: only overwrite frame if fault detected
  - Read back configuration data
  - Check against original configuration data (e.g. CRC comparison)
  - On error: write corrected configuration data back to FPGA
- Advantages: SEU logging

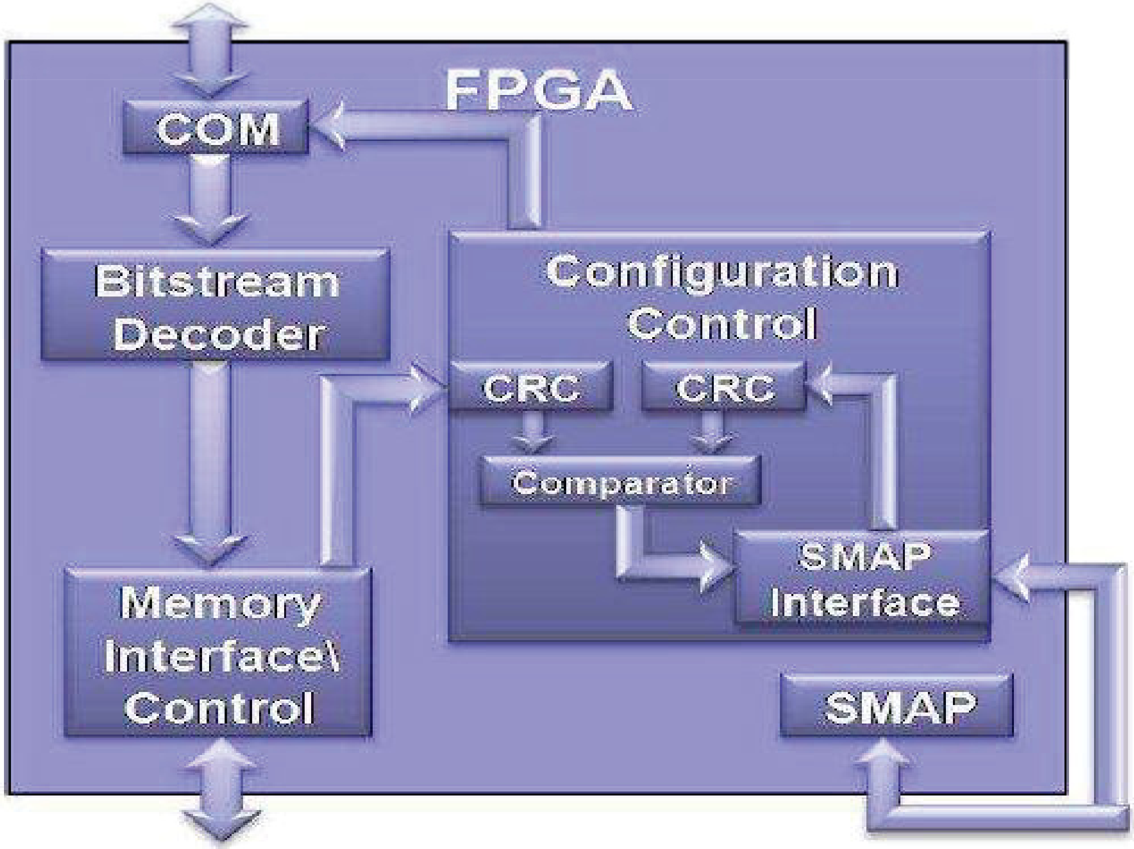
Internal Scrubbing

- Strategy:
  - Read configuration frame via ICAP
  - Check frame-internal CRC code and correct errors if necessary
  - Write configuration frame via ICAP
- Xilinx proprietary method
- No external memory required
- Uses BRAM → scrubber vulnerable to SEUs
- Error correction can only correct 1 Bit errors, 2 bit errors are detected but not corrected, 4 and 8 bit errors can go completely undetected

Partial Reconfiguration Scrubbing

- Traditional Scrubbing methods can not be used with partial reconfiguration
  - Scrubbing uses configuration port constantly
  - When loading PR bitstream, scrubber tries to read/write to configuration memory, while PR logic tries write to it
  - Even if scrubbing pauses for PR, scrubber will immediately overwrite PR region again (i.e. scrubber ‘repairs’ the region)
- Potential Solution: Update “golden” bitstream
  - Golden bitstream contains reference bitstream in radiation hardened memory used for scrubbing
  - Writing the PR modifications to golden bitstream in an atomic operation (i.e. scrubbing should not read that part from hardened memory in between)
  - Then, scrubbing will reconfigure the PR part to the FPGA after a short delay

Partial Reconfiguration Scrubbing



src: [HSWK09]

- Implemented on Virtex-4
- Communication Interface – UART, receive bitstreams from host computer
- Memory – 64 MB SDRAM for bitstream storage
  - Arbiter resolves decoder/scrubber memory access conflicts

Partial Reconfiguration Scrubbing

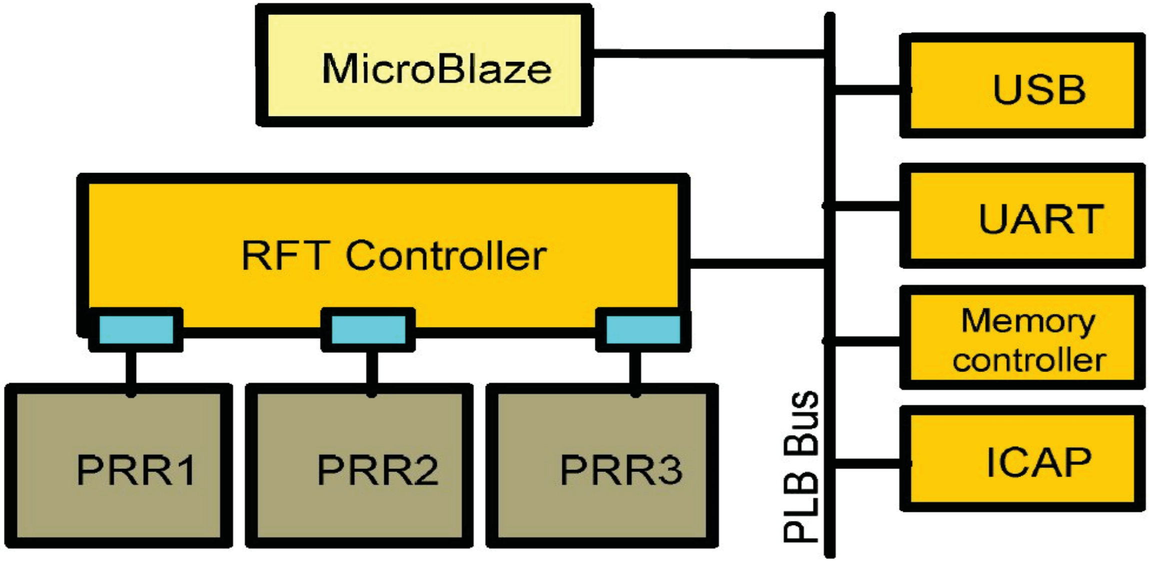
- Bitstream decoder – prepare bitstream for insertion into golden bitstream
- Configuration Controller – manage scrubbing
  - Read frame from golden bitstream and configuration memory
  - Compute CRC values
  - If different, write frame from golden bitstream to configuration memory
- Partial Reconfiguration done automatically by Configuration Controller
  - Golden bitstream updated with PR bitstream
  - Configuration Controller detects the difference in the modified frames
  - Frames in configuration memory are overwritten → PR complete

8.5 Reliability in Space

Reconfigurable Fault Tolerance (RFT) in Space

- Different scenario: FPGAs in space-based applications
- Preprocessing of data on-board to minimize downlink bandwidth
- Common fault detection/mitigation
  - Radiation hardened devices – very expensive, lower performance
  - TMR – Problem: area overhead (> 200% more), assumes worst-case scenario
- Use reconfiguration to adapt to desired level of redundancy/performance
- Developed at University of Florida

RFT Architecture



src: [JGC09]

- SoC with Partial Reconfiguration Regions (PRR) that contain additional processing modules/accelerators
  - All components except PRRs may be protected by TMR
- MicroBlaze keeps track of modules
  - active or not
  - switch fault tolerance strategies using ICAP
  - Initiate recovery when module encounters error

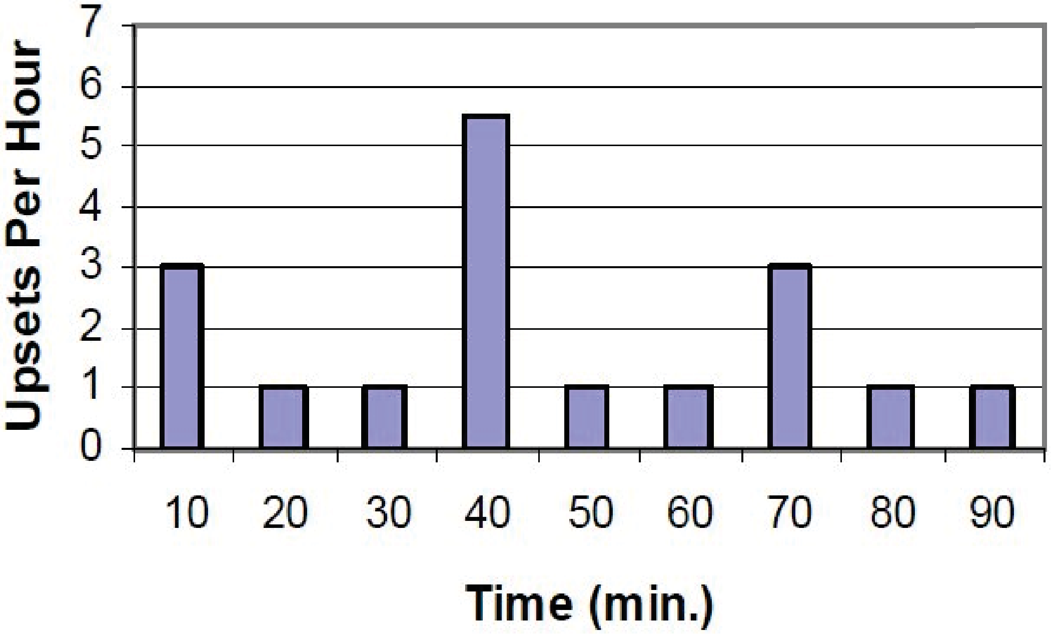
RFT Modes

- Triple modular redundancy (TMR) mode: replicate module in three different PRRs
  - Voting implemented in RFT Controller
  - Error → Interrupt to MicroBlaze, which initiates recovery
    - Save system state
    - Reconfigure PRR
    - Load module state back
- High Performance mode: no fault tolerance by system
  - Reliability through module-internal means still possible

RFT Modes

- Self-checking Pair (SCP) mode:
  - Replicate module in two different PRRs
  - Error → reconfigure both, repeat computation
- Switching Reconfigurable Fault Tolerance (RFT) modes:
  - Triggered by external events or prior knowledge of the environment
  - RFT controller disables affected PRRs, extracts their state and changes voting procedures
  - Partial bitstreams sent to ICAP
  - RFT controller re-enables bus connections

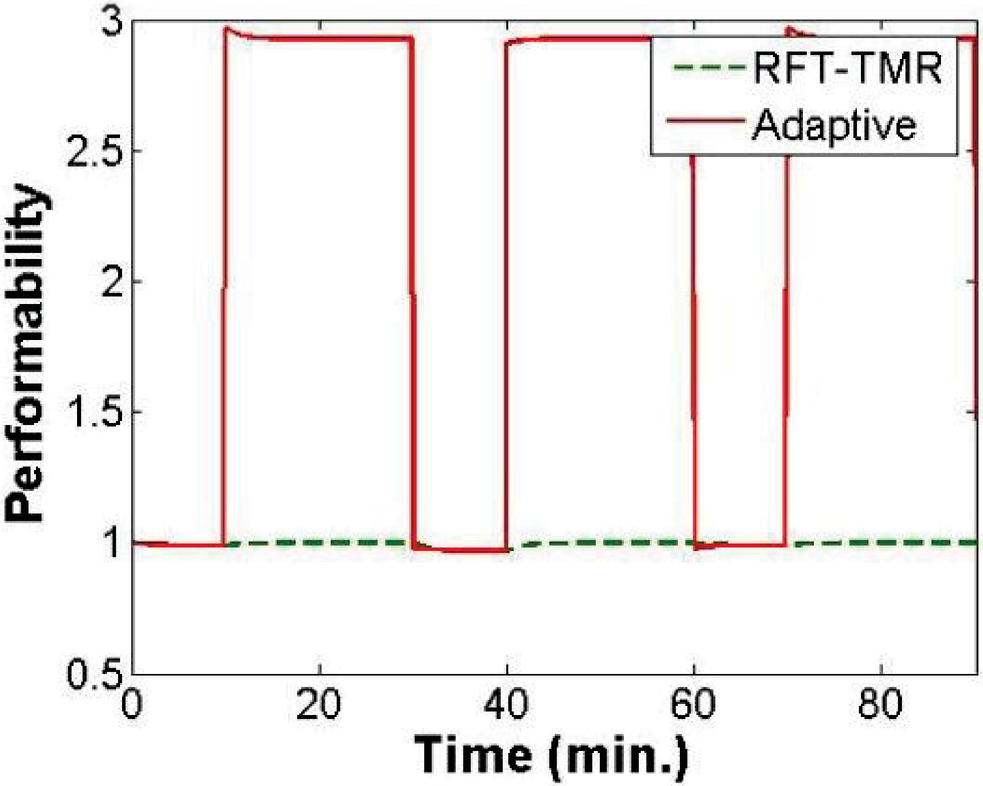
RFT Case Study – ISS



src: [JGC09]

- International Space Station
  - Low Earth Orbit – 400km height, 92 min to complete, avoids travel over poles to minimize radiation exposure to crew
- SEU rates depend on solar activity, particular device, etc.
  - Here: only estimates

RFT Case Study – ISS



src: [JGC09]

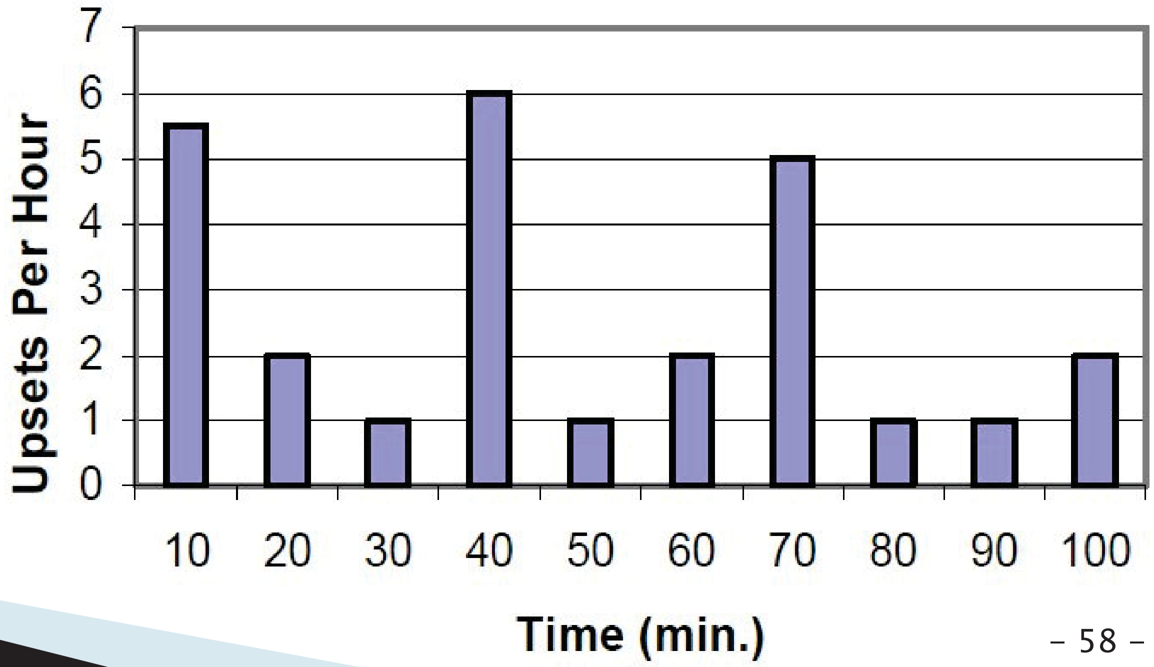
- Prior knowledge of orbit and solar conditions
- High Performance mode in sections with low SEU rates
- Reconfigure to TMR mode when radiation exposure high
- During both modes: Scrubbing of configuration memory in 30 sec. cycles

RFT Case Study – ISS

- Results
  - Configuration memory repair rate (scrubbing) much higher than SEU rate
  - During high radiation periods traditional TMR and RFT perform similar (RFT in TMR mode)
  - During low radiation parts RFT performs better
  - Average performance of RFT over TMR: 2.3x

RFT Case Study – HEO

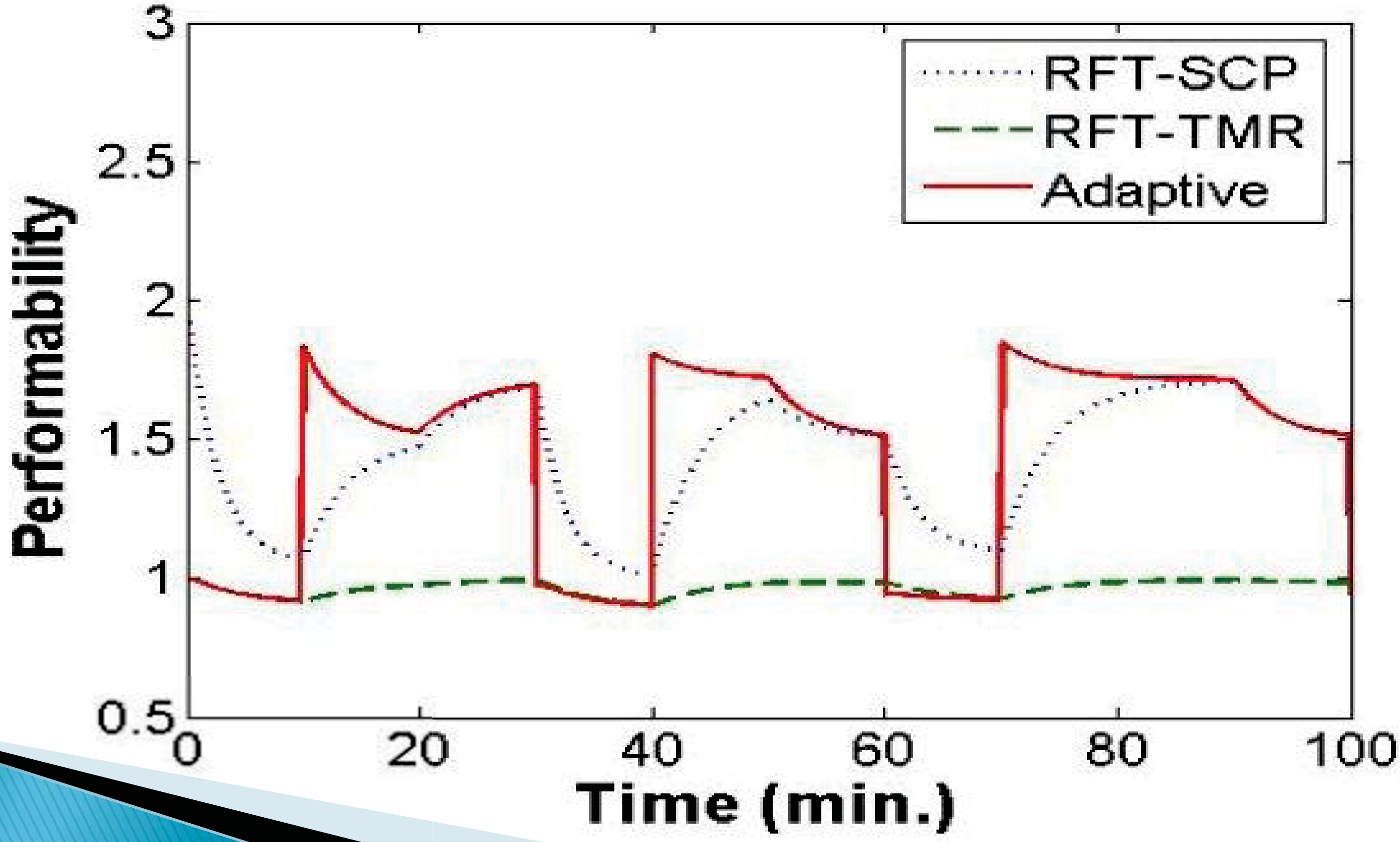
- Highly Elliptical Orbit (HEO)
- stay longer over an area and can cover polar regions
- Used by communication satellites
- Geostationary orbits only cover equatorial regions
- Average radiation higher



src: [JGC09]

RFT Case Study – HEO

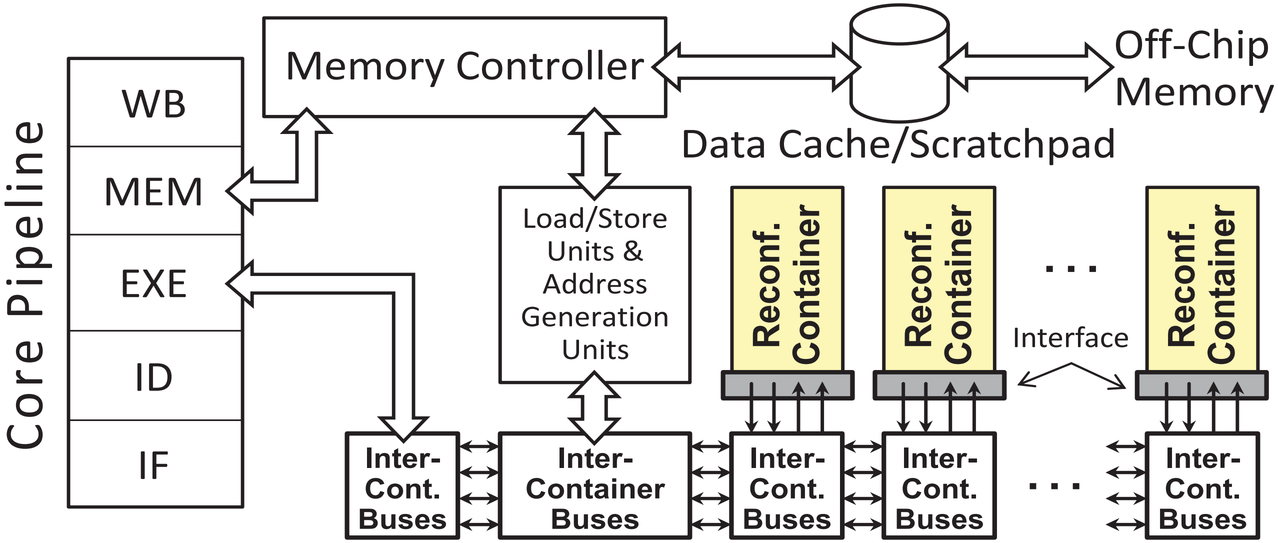
- System switches between TMR (3 PRRs used) and Self-checking Pair (4 PRRs used running 2 applications) modes
- Modules checkpoint state every 5 minutes



8.6 OTERA

OTERA – Online TEst strategies for runtime Reconfigurable Architectures

- RISPP revisited:



- **Reliable online-reconfiguration using online test**
  - Fabric fault free?
  - Reconfiguration process completed correctly?

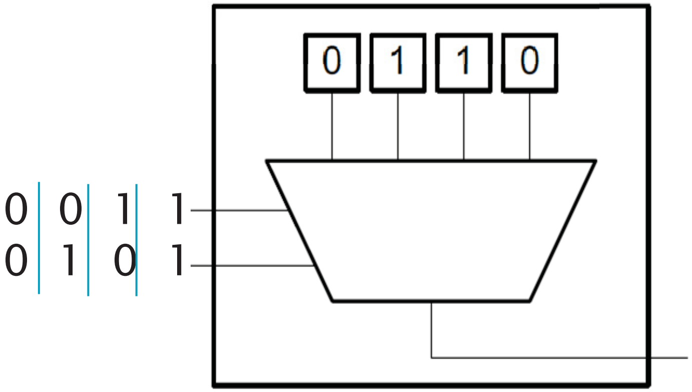
→Must be ensured *at runtime!*

OTERA – Test Methods

- Pre-configuration test (PRET)
  - Tests structural integrity of reconfigurable fabric
  - Executed online before reconfiguration with mission logic
- Post-configuration test (PORT)
  - Test correct reconfiguration and interconnection
  - Functional, software-based test
  - Execured online, at speed

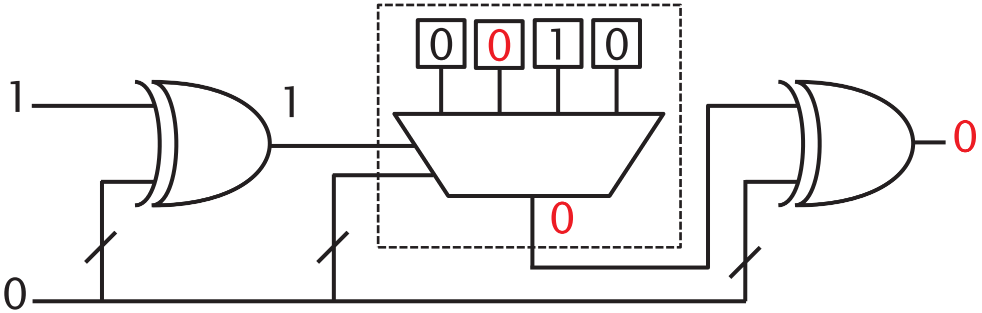
Example: Testing a Lookup-Table

- Principal structure:
  - Truth table, multiplexer
- 2 test configurations
  - Set each memory cell to 0 and 1
  - XOR and XNOR
  - Exhaustive test set (2n patterns)



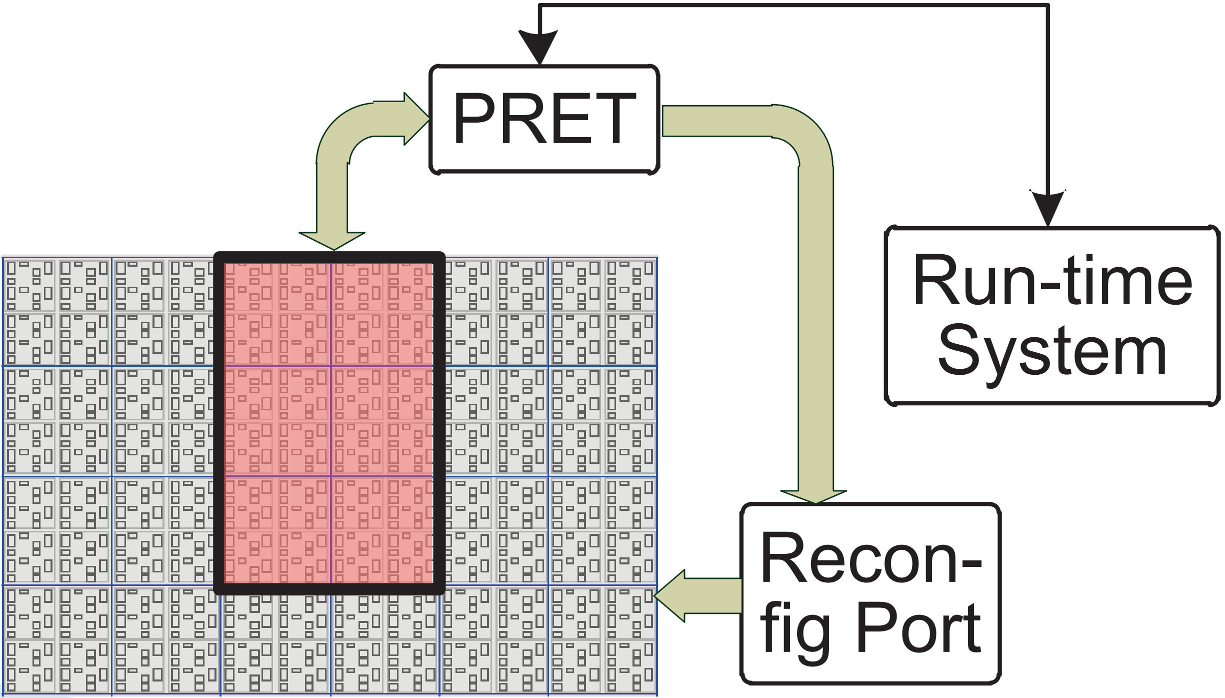
XOR configuration

- Optimizations:
- C-testable array
- Pipelining for at-speed test



OTERA – Test Procedure

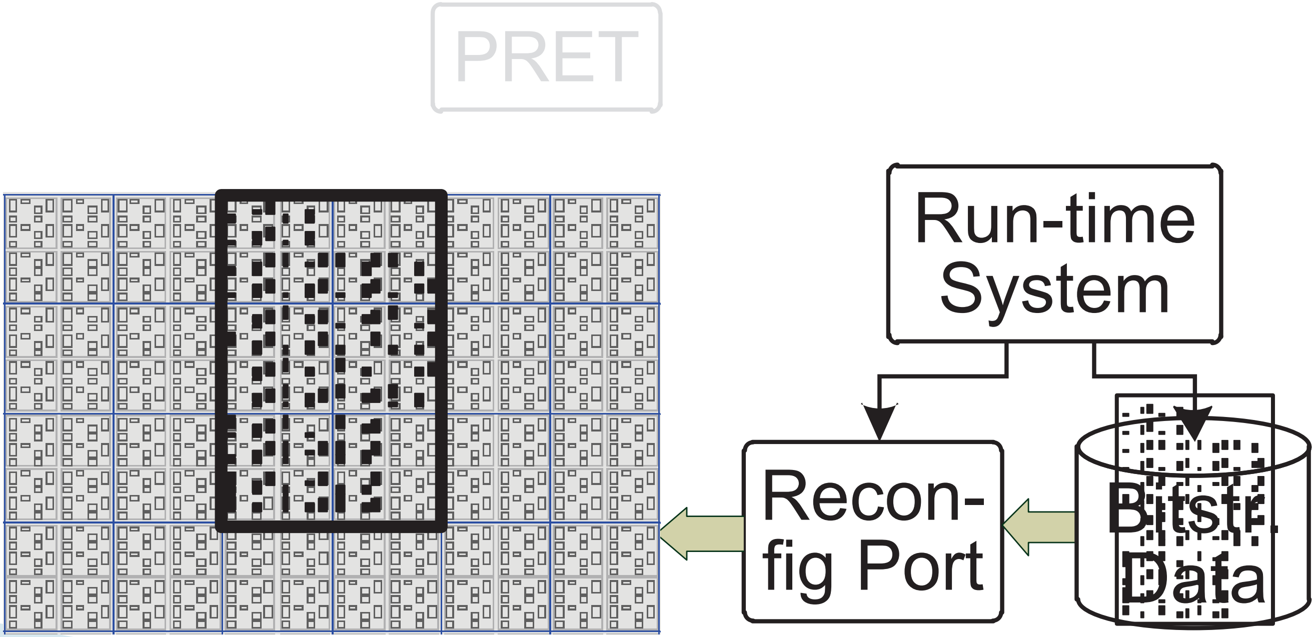
- 1. Basic pre-configuration online test



src: [BBI+12]

OTERA – Test Procedure

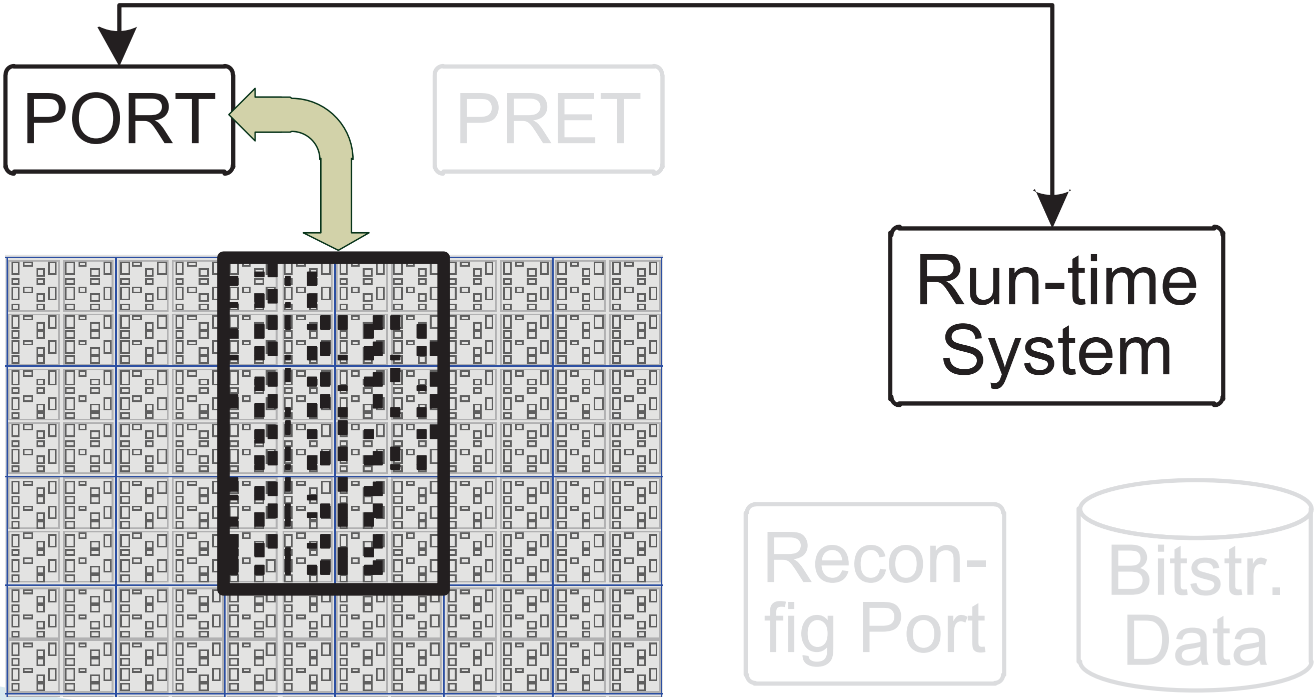
- 2. Reconfigure the accelerator into the container



src: [BBI+12]

OTERA – Test Procedure

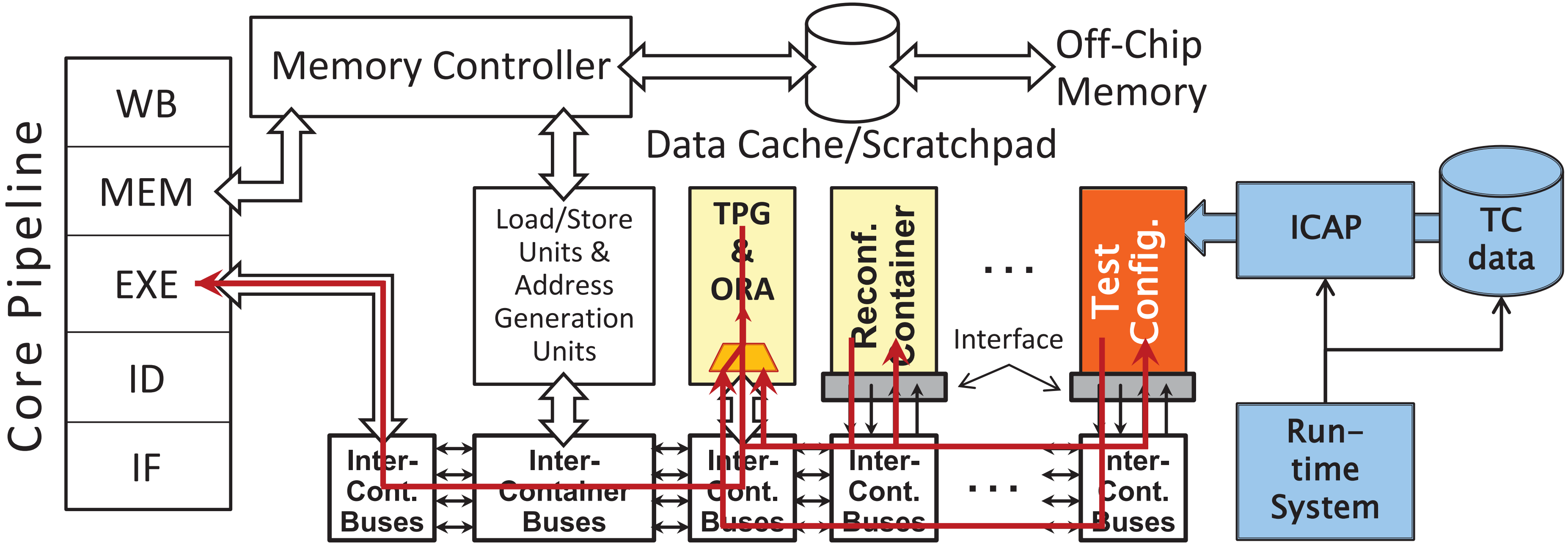
- 3. Post-reconfiguration online test (PORT)
  - After reconfiguration
  - Periodically during operation



src: [BBI+12]

OTERA – PRET System Integration

- Connect Test Pattern Generator (TPG) and Output Response Analyzer (ORA) with the Reconf. Containers
  - Can use the Inter-Container Buses for communication
- After loading a Test Configuration (TC), the test is performed like a regular application-specific Special Instruction



src: [BBI+12]

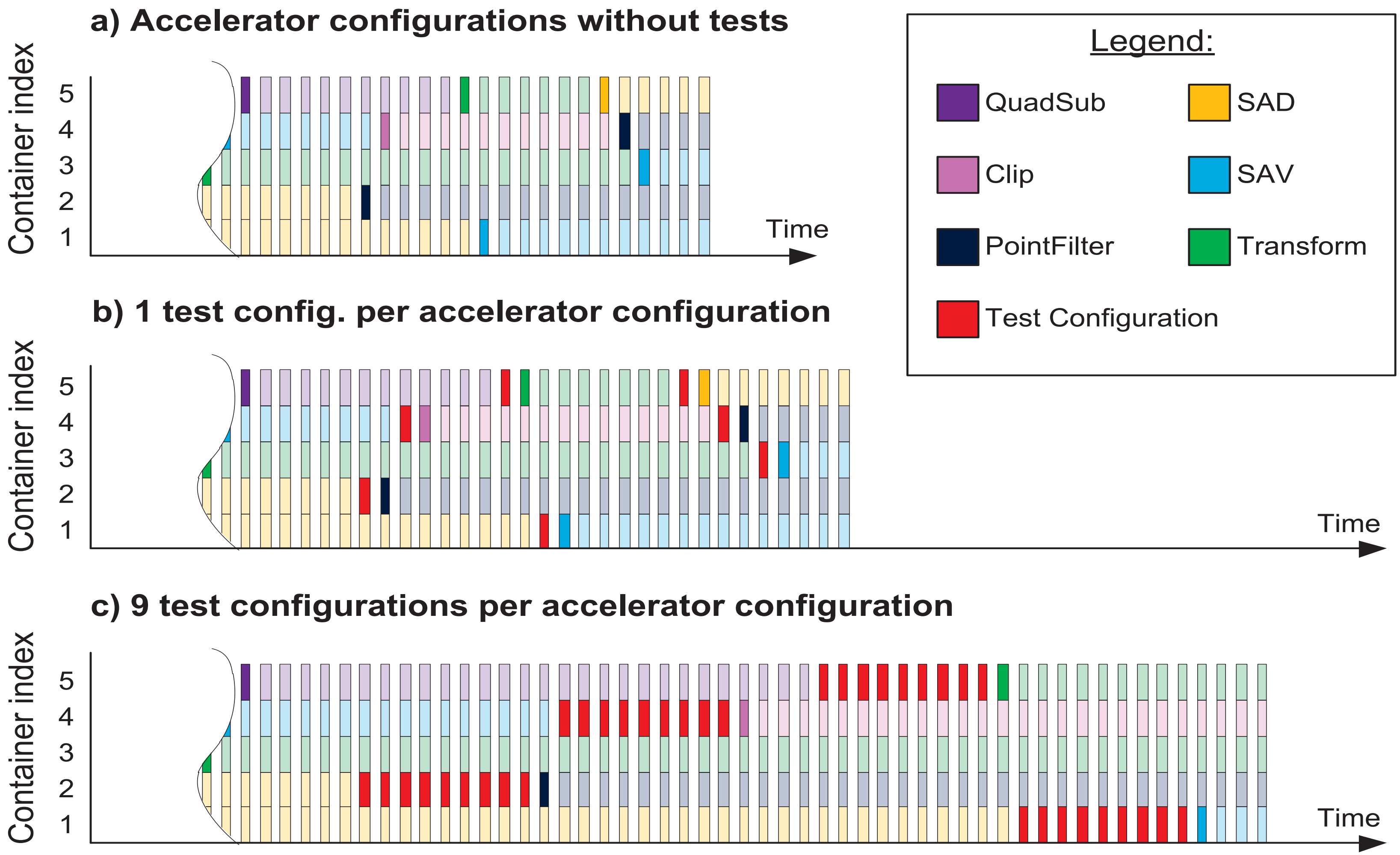
Test Configurations

- 9 Test configurations (TCs) to cover all targeted faults in CLBs

TC	Tested CLB subcomponents	PRET over-head [CLBs]	Bitstream size [KB]	Freq. [MHz]	Number of Patterns
1	LUT as XOR, via FF	2	24.0	207	64
2	LUT as XNOR, via FF	2	24.0	207	64
3	Carry MUX, via latch	1	28.6	168	6
4	Carry MUX, via latch	1	26.1	154	6
5	Carry XOR, via FF	1	28.0	168	6
6	Carry XOR, via FF	1	28.2	154	6
7	Carry-I/O multiplexed	1	27.1	183	6
8	LUT as Shift Reg. with slice MUX	1	22.9	157	6
9	LUT as RAM with slice output	7	22.3	225	320

- Test configuration scheduling integrated into system scheduling & configuration infrastructure

OTERA – Test Scheduling



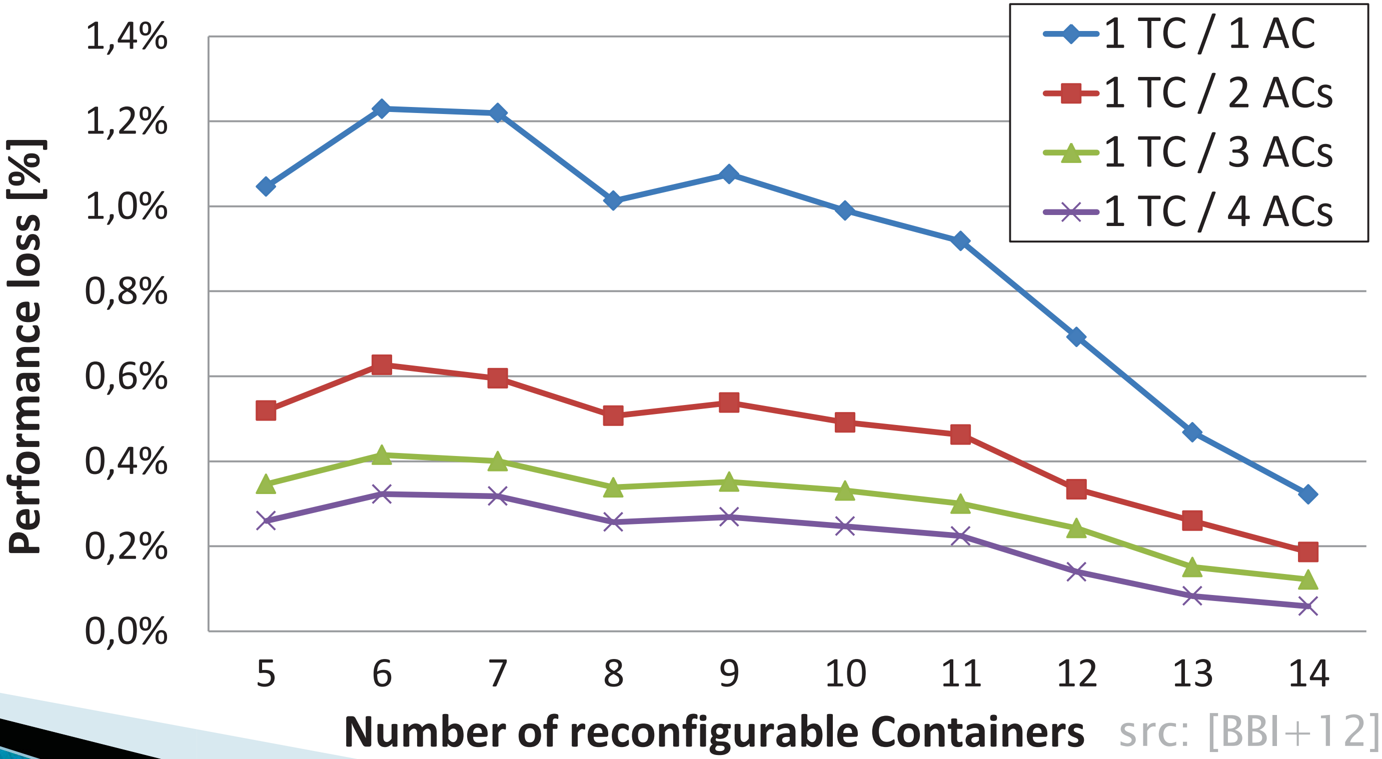
a) Accelerator configurations without tests

b) 1 test config. per accelerator configuration

c) 9 test configurations per accelerator configuration

OTERA – Performance Overhead

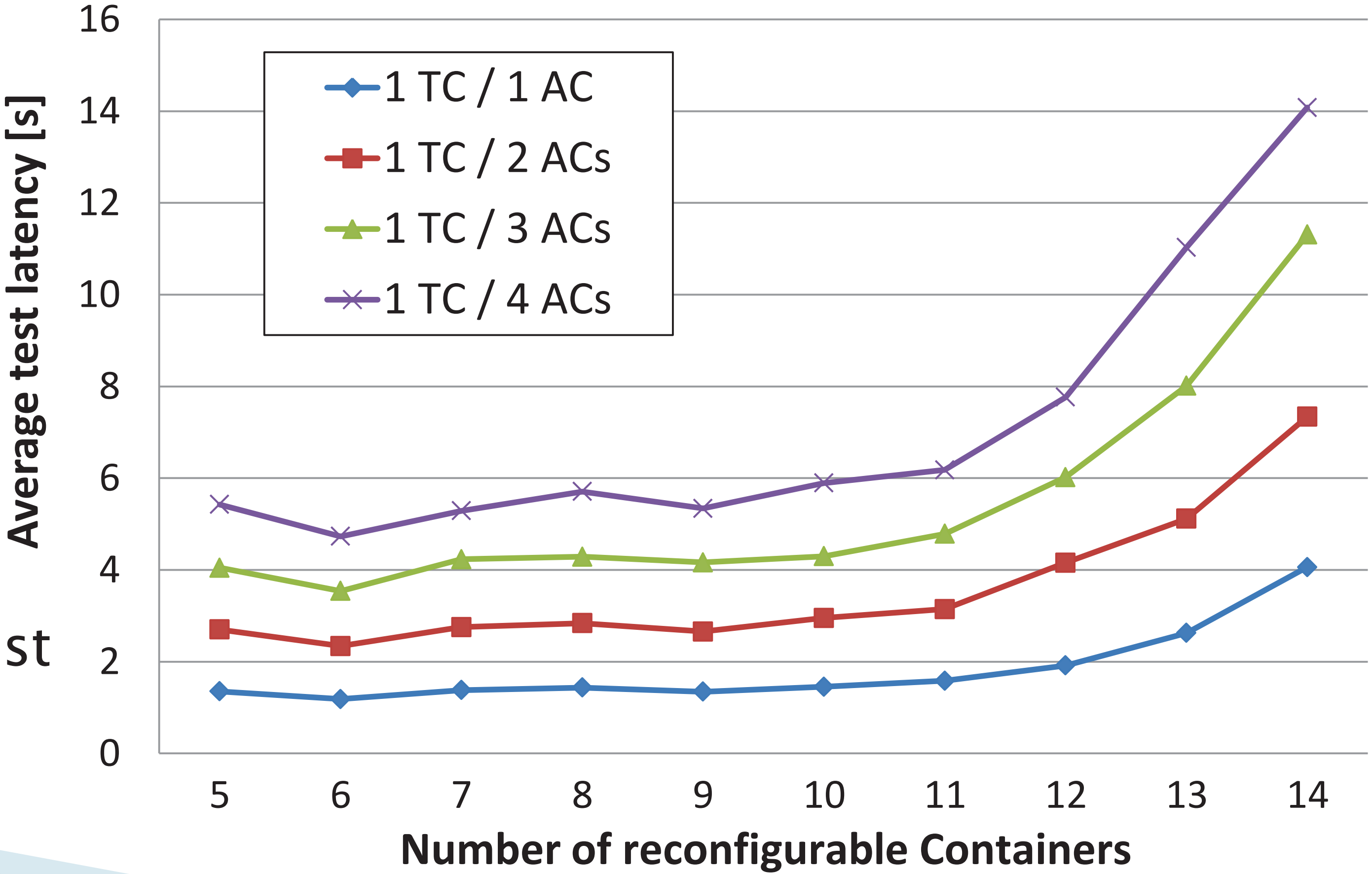
- H.264 video encoding running on reconf. system
- Investigating different test frequencies
  - 1 Test Config. (TC) per *X* Accelerator Configurations (AC)
- Negligible appl. perfor-mance impact
  - Typ. < 1%



src: [BBI+12]

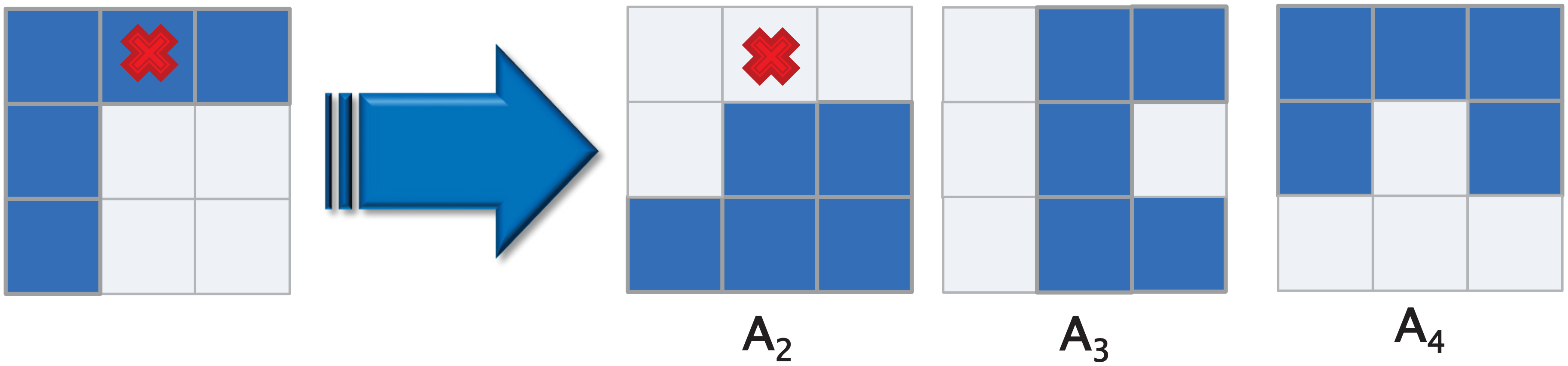
OTERA – Test Latency

- Test Latency: the time to complete all tests (9 test configurations for all containers)
- Short test latency (between 1.2 and 14.1 s)
- Depends on number of containers and test frequency



src: [BBI+12]

Module Diversification



🟦 used

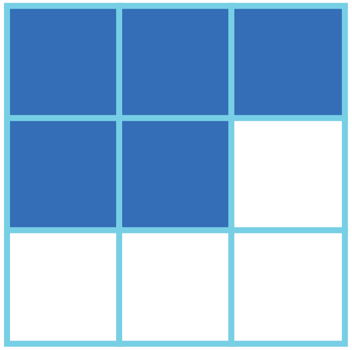
⬜ unused

✖ faulty

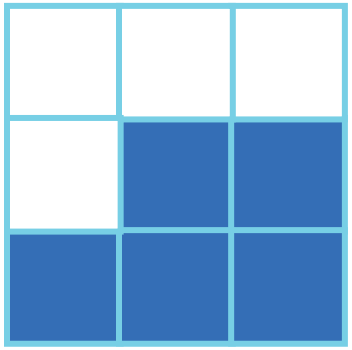
- Implement functional modules in different ways in terms of CLB usage (Placement constraint)
  - → Diversified configurations

src: [ZBK+13]

Generate Diversified Configurations

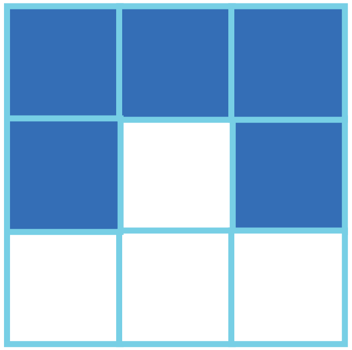


A$_1$



A$_2$

2	2	2
2	2	2
1	1	1

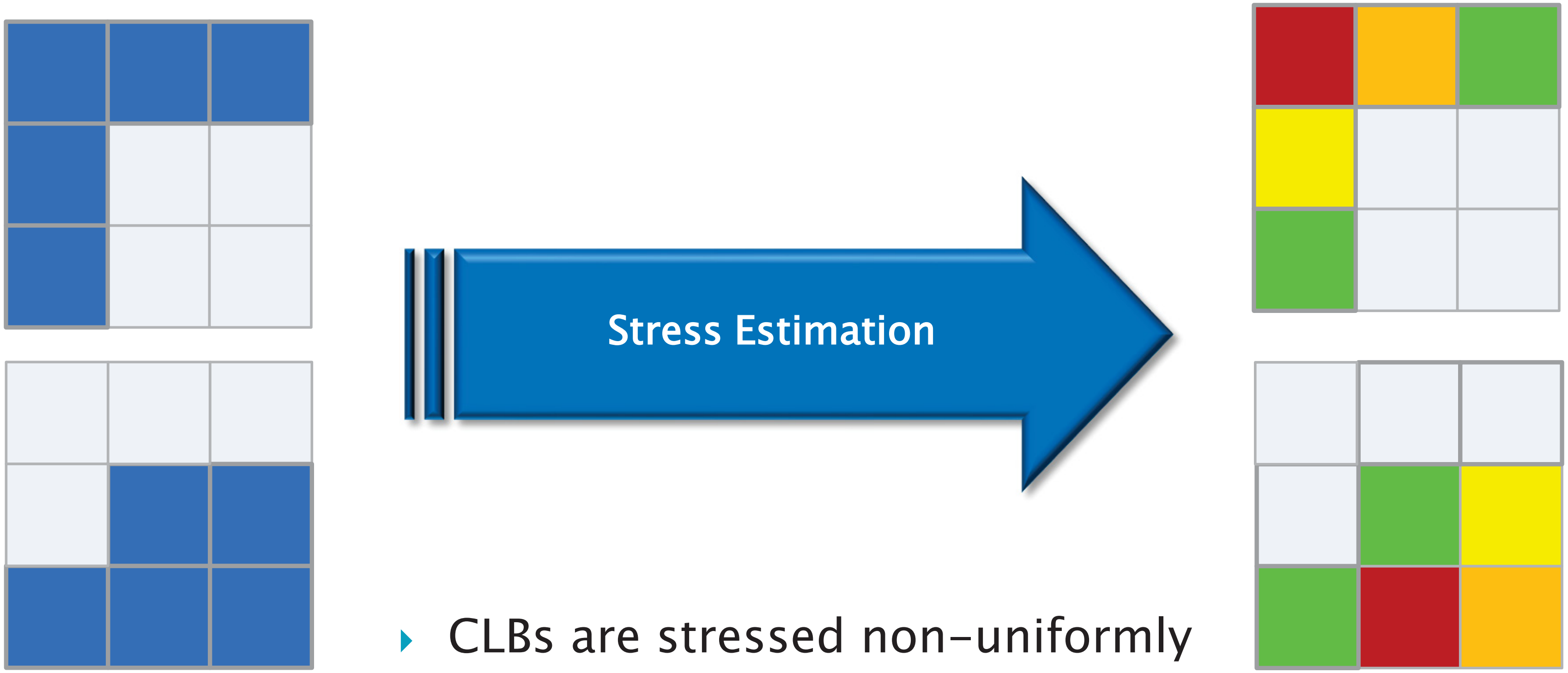


A$_3$

- Goal: Create a minimal set of diversified configurations that tolerate any single-CLB fault
  - Track for each CLB how many configurations already used it
  - Create a new configuration out of an existing one by swapping the most often used CLBs with the least often used ones

src: [ZBK+13]

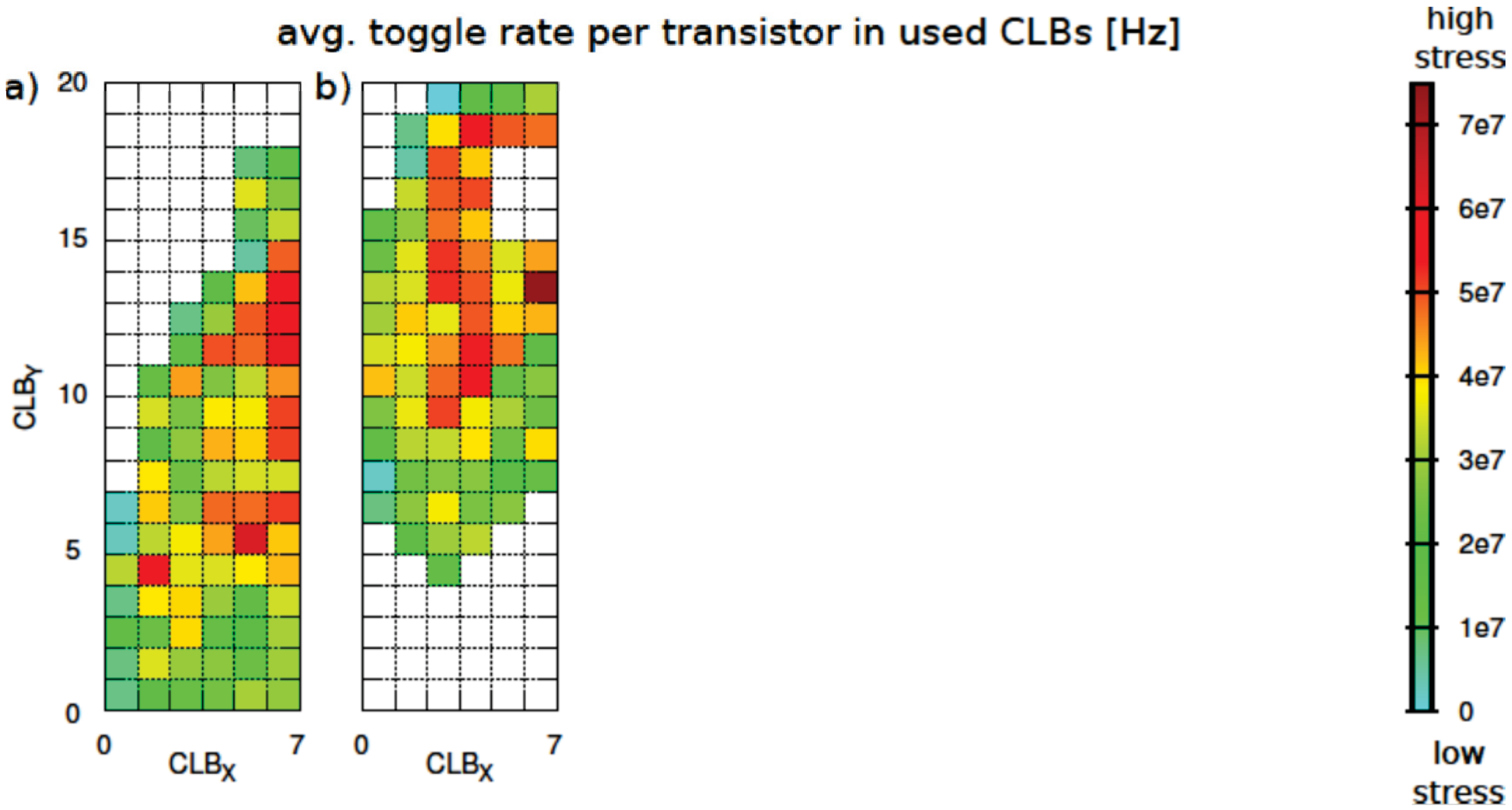
Stress Balancing for Aging Mitigation



- CLBs are stressed non-uniformly
- Decrease stress = reduce aging
- Distribute the stress over CLBs

src: [ZBK+13]

Stress Reduction

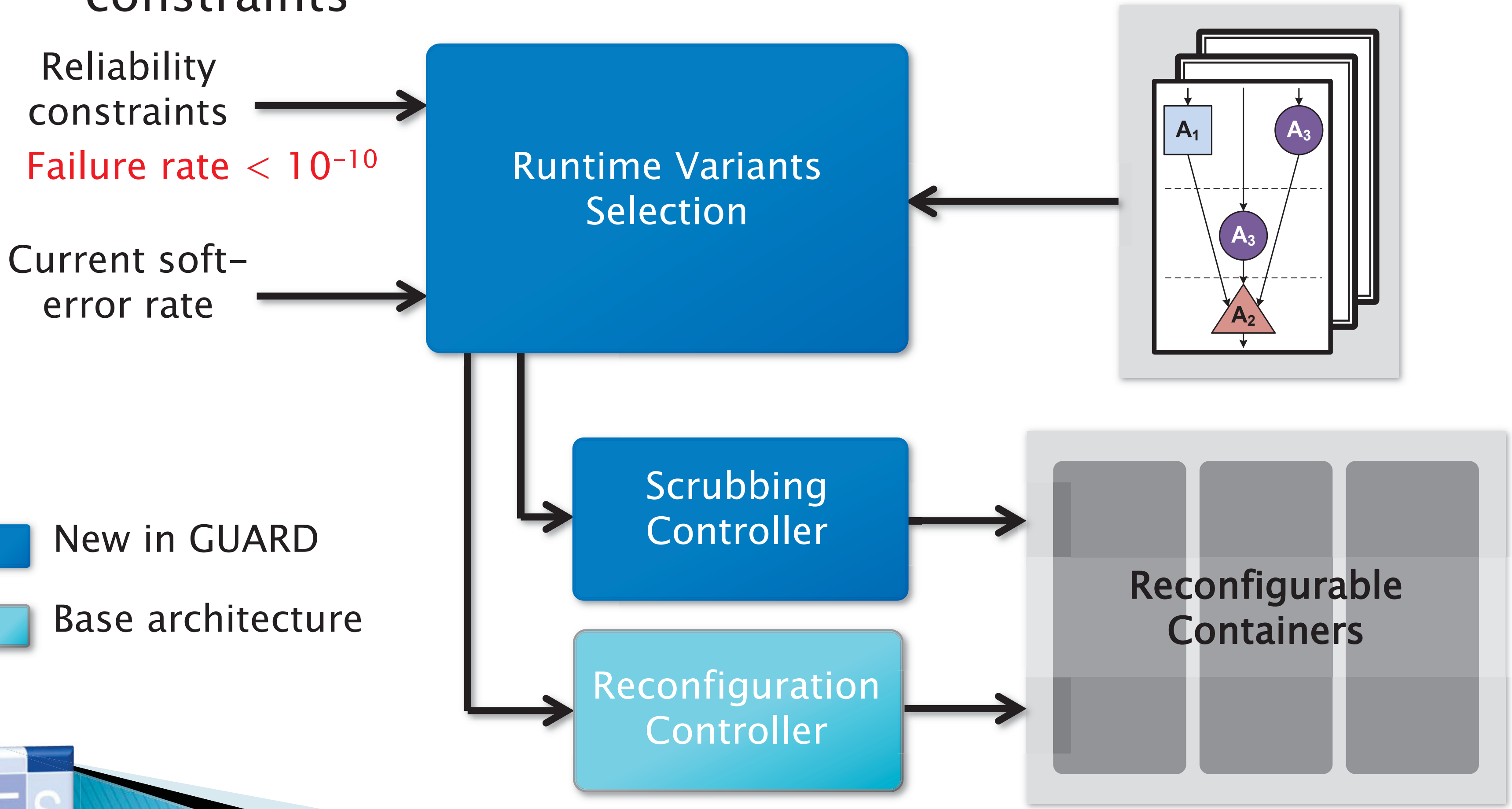


- a), b) two diversified configurations
- c) an alternating schedule
- d) a balanced schedule of the min. set (4 configurations)

src: [ZBK+13]

GUARD: GUAranteed Reliability in Dynamically Reconfigurable Systems

- Goal: Maximize performance under given reliability constraints



Reliability constraints

Failure rate < 10^{-10}

Current soft-error rate

Runtime Variants Selection

Scrubbing Controller

Reconfiguration Controller

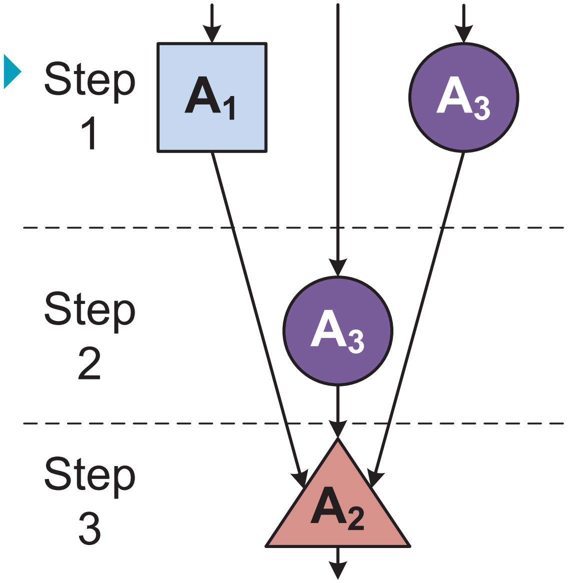
Reconfigurable Containers

🟦 New in GUARD

🟦 Base architecture

src: [ZKI+14]

Variants of Accelerated Functions



a) Example for an Accelerated Function

3 Containers 4 Containers 5 Containers

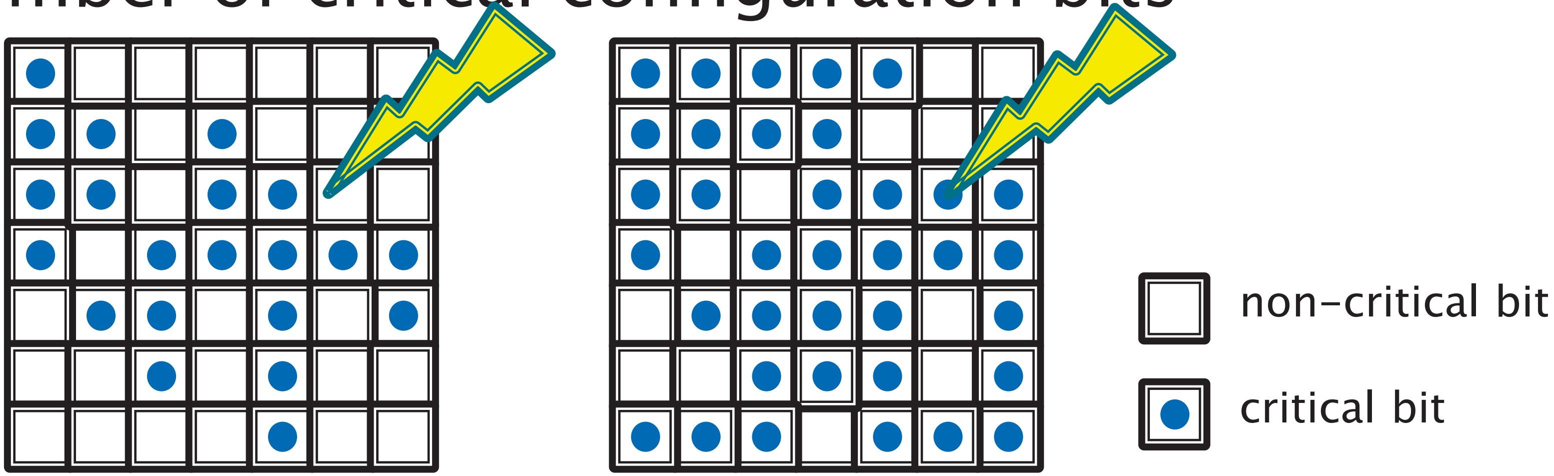
Legend: A_1 A_2 A_3 Different accelerator types

Trade-off performance with reliability

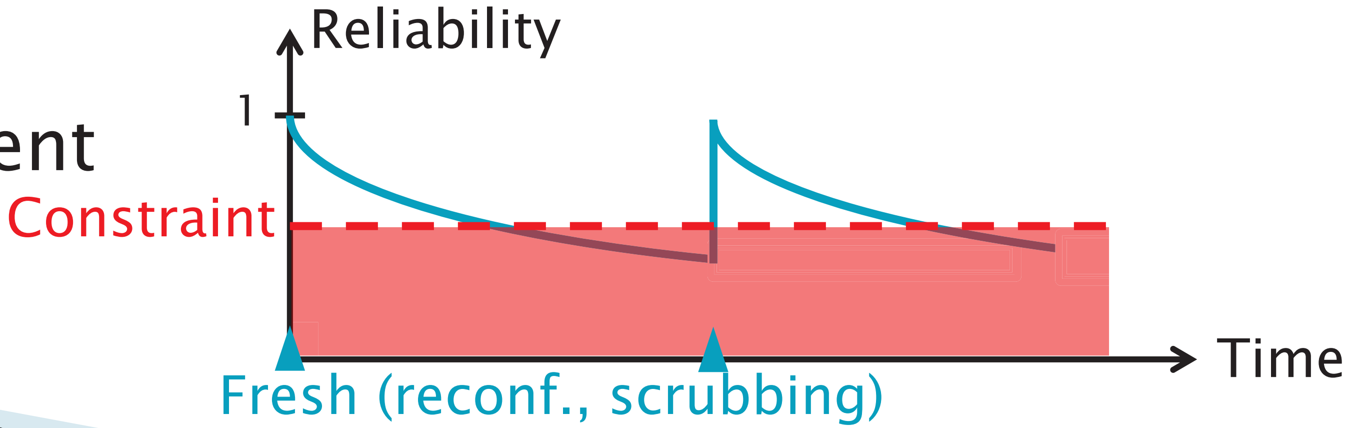
src: [ZKI+14]

Reliability of Accelerators

- Number of critical configuration bits



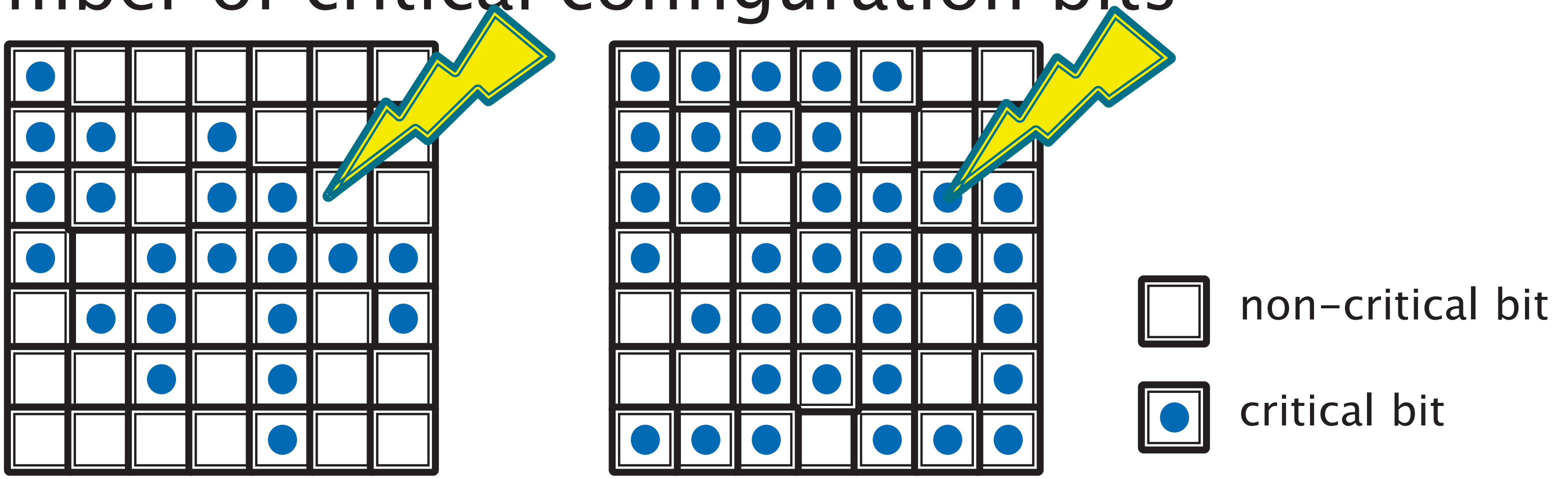
- Resident time



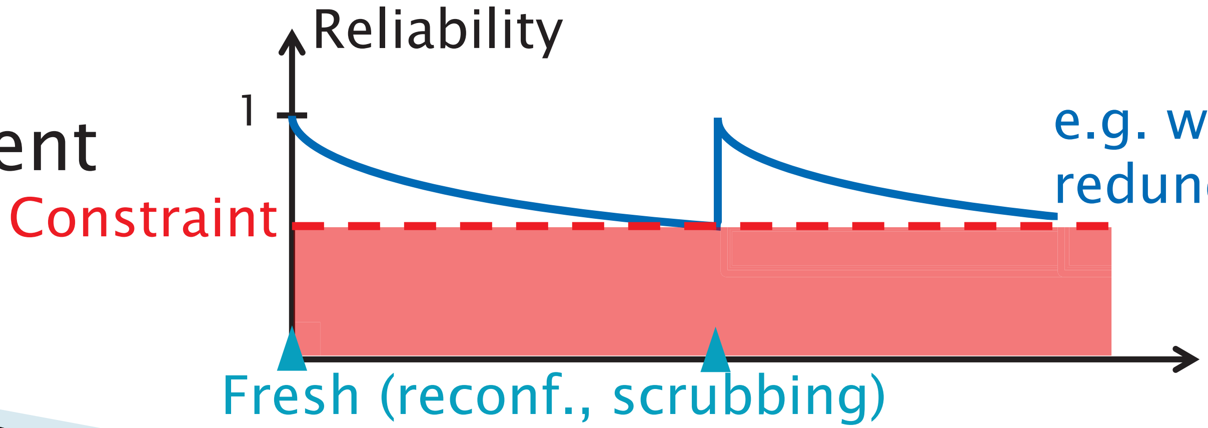
src: [ZKI+14]

Reliability of Accelerators

- Number of critical configuration bits



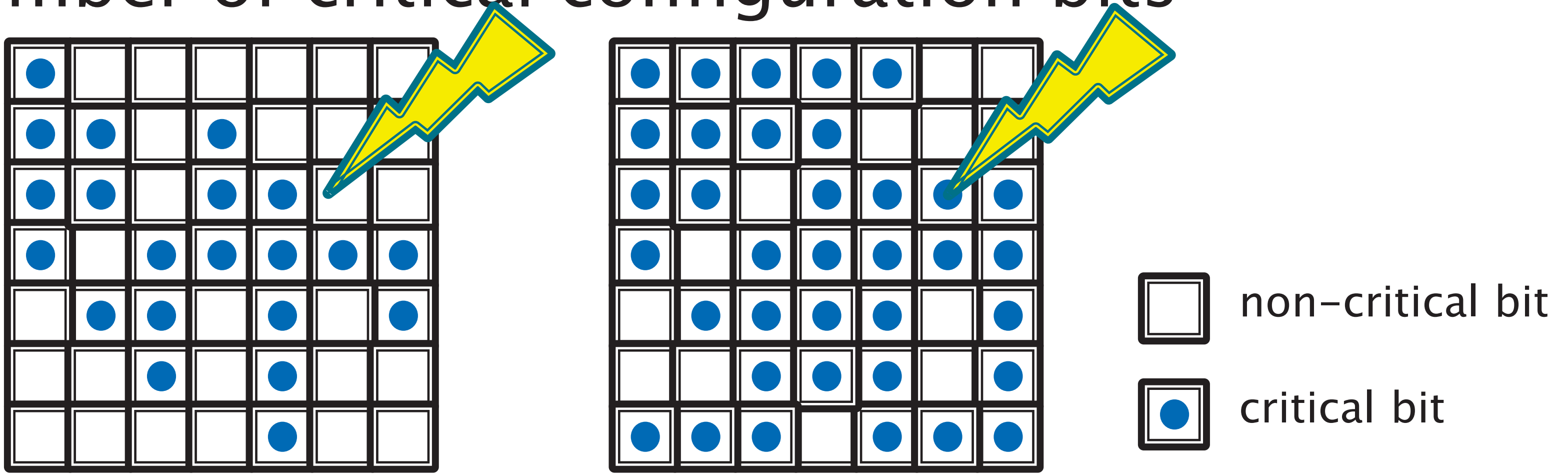
- Resident time



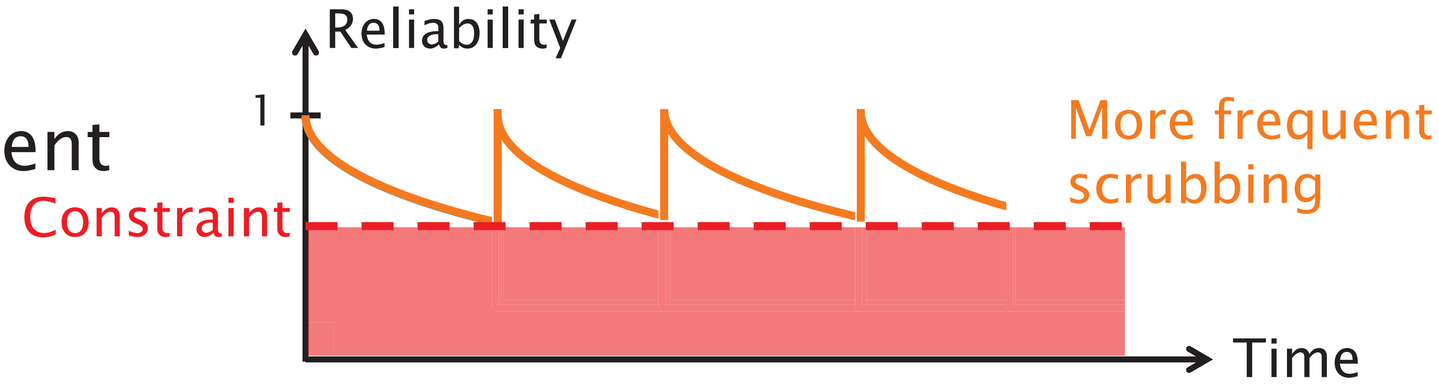
src: [ZKI+14]

Reliability of Accelerators

- Number of critical configuration bits

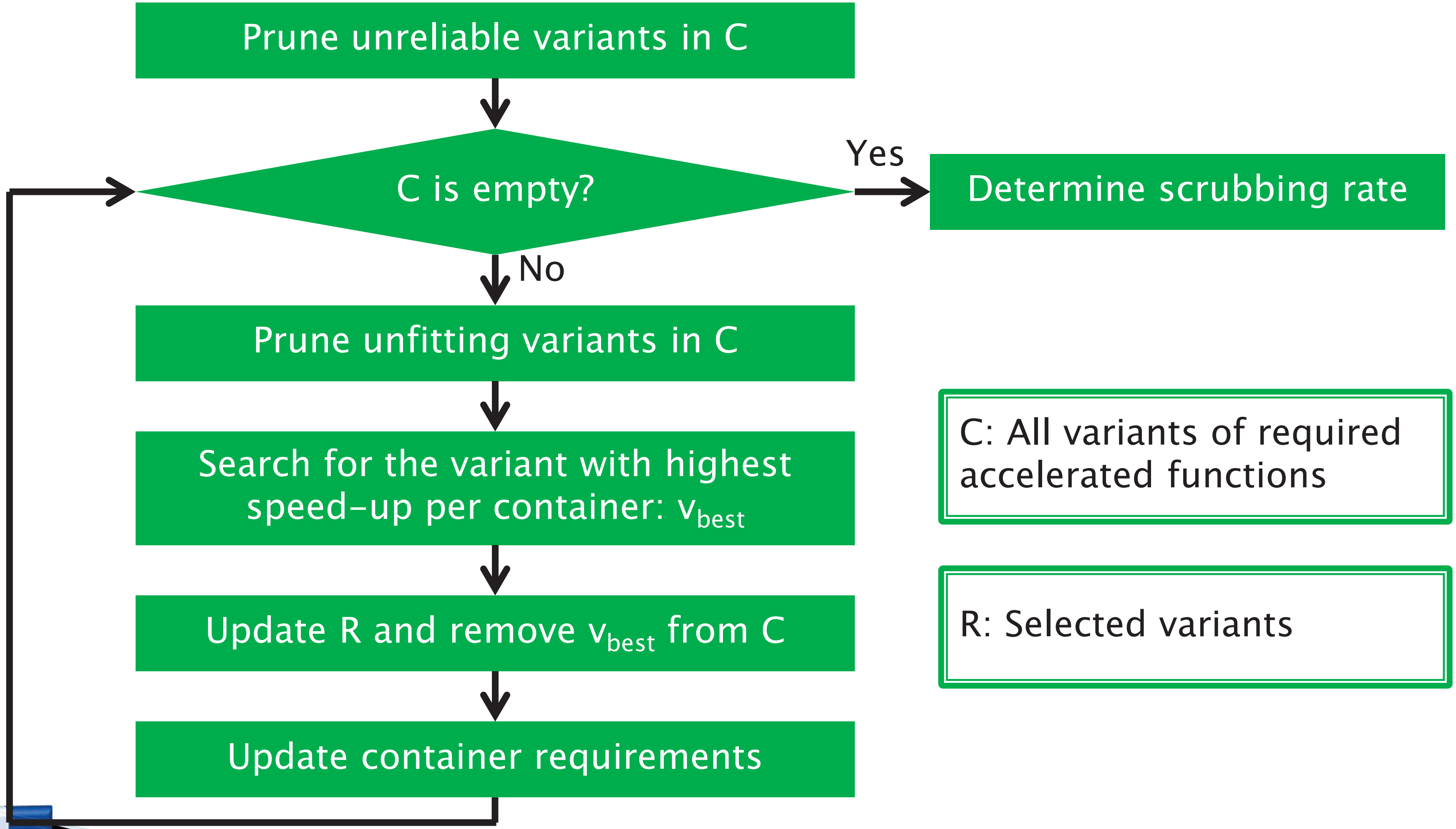


- Resident time



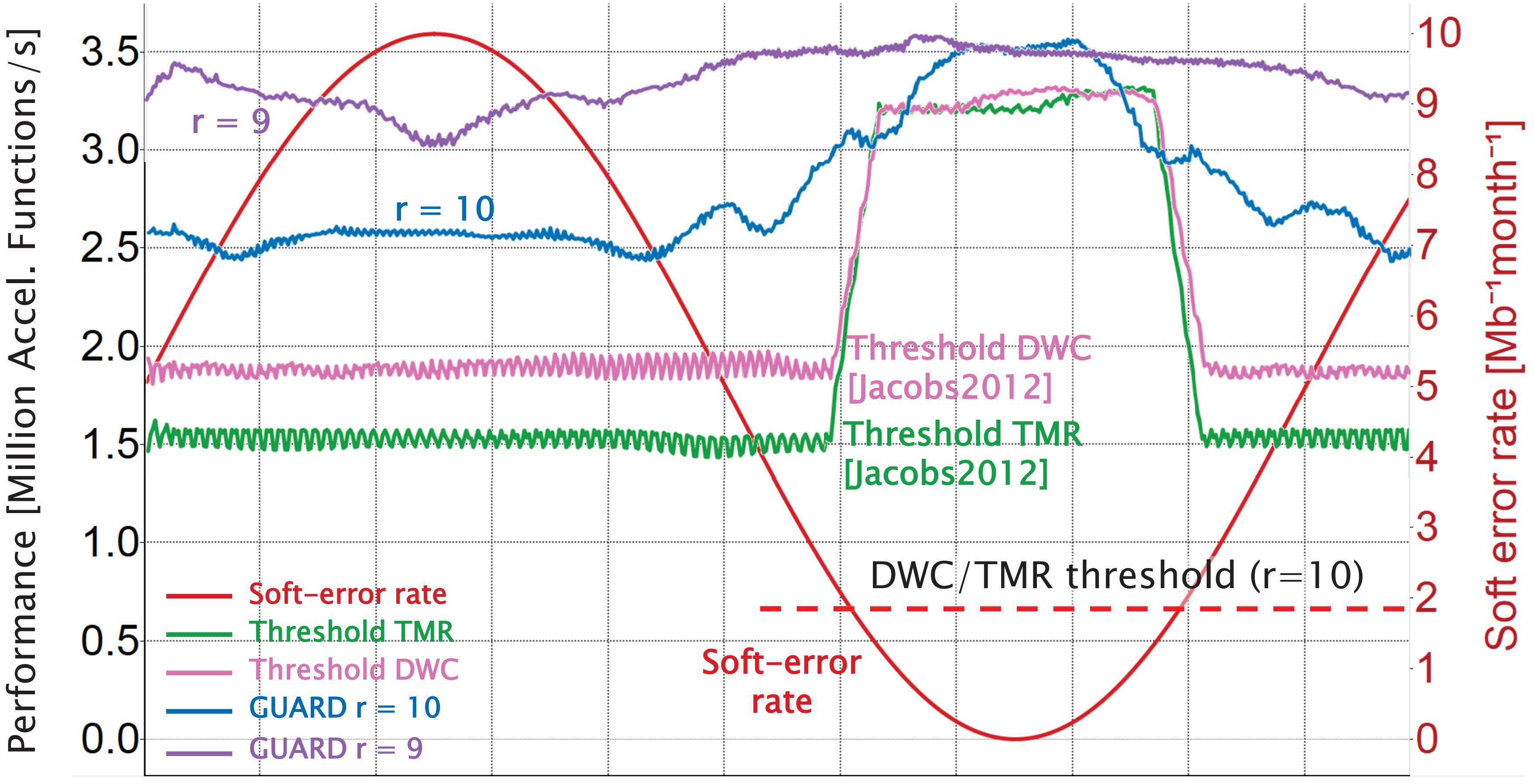
src: [ZKI+14]

Variants Selection – Greedy Algorithm


Prune unreliable variants in C
C is empty?
Yes
Determine scrubbing rate
No
Prune unfitting variants in C
Search for the variant with highest speed-up per container: v_{best}
Update R and remove v_{best} from C
Update container requirements
C: All variants of required accelerated functions
R: Selected variants

src: [ZKI+14]

Results: Runtime Adaptation



Average performance improvement: 42.6%

src: [ZKI+14]

Conclusion

- Developed a thorough CLB test and integrated it into a reconfigurable system
    - Using system facilities for reconfiguration and test access
    - Extended tool-chain to create partial bitstreams for Test Configurations
    - Transparent for the application
    - Very low area & performance overhead and fast test latency
- Realized fault-tolerance and aging mitigation via diversified module configurations
- Dynamic performance/reliability trade-off
- Validated on HW Prototype

Sources, References, Further Reading

[CCMA10] M. Choudhury, V. Chandra, K. Mohanram, and R. Aitken: “Analytical model for TDDB-based performance degradation in combinational logic”, In Proceedings of the Conference on Design, Automation and Test in Europe (DATE '10). Leuven, Belgium, 423-428. 2010.

[LCR03] F. Lima, L. Carro, R. Reis: “Designing fault tolerant systems into SRAM-based FPGAs”, Design Automation Conference (DAC), pp. 650-655, 2003.

[CCCV05] N. Campregher, P.Y.K. Cheung, G.A. Constantinides, M. Vasilko: “Analysis of yield loss due to random photolithographic defects in the interconnect structure of FPGAs”, 13th international symposium on Field-programmable gate arrays (FPGA), pp. 138-148, 2005.

[SSC08] E. Stott, P. Sedcole, P. Cheung: “Fault tolerant methods for reliability in FPGAs”, Int’l Conference on Field Programmable Logic and Applications (FPL), pp. 415-420, 2008.

[ESSA00] J. Emmert, C. Stroud, B. Skaggs, M. Abramovici: “Dynamic fault tolerance in FPGAs via partial reconfiguration”, IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 165-174, 2000.

[LC07] A. Lesea, K. Castellani-Coulie: “Experimental study and analysis of soft errors in 90nm Xilinx FPGA and beyond”, 9th European Conference on Radiation and it’s Effects on Components and Systems, pp. 1-5, 2007.

[B06] M. Berg: “Fault tolerance implementation within SRAM based FPGA designs based upon the increased level of single event upset susceptibility”, 12th IEEE International On-Line Testing Symposium (IOLTS), p. 89-91, 2006.

Sources, References, Further Reading

[HSWK09] J. Heiner, B. Sellers, M. Wirthlin, J. Kalb: “FPGA partial reconfiguration via configuration scrubbing,” Int’l Conf. on Field Programmable Logic and Applications (FPL), pp. 99–104, 2009.

[K08] T. Krawutschke: “A flexible and reliable embedded system for detector control in a high energy physics experiment”, Int’l Conf. on Field Pr. Logic and Appl. (FPL), pp. 155–160, 2008.

[M07] J. Mercado: “The ALICE Transition Radiation Detector Control System”, Int’l Conference on Accelerators and Large Experimental Physics Control Systems (ICALEPCS), pp. 181–183, 2007.

[ALCol03] ALICE Collaboration: “ALICE Technical Design Report of the Trigger Data Acquisition High-Level Trigger and Control System”, ISBN 92-9083-217-7, pp. 359 – 412, 2003.

[JGC09] A. Jacobs, A. George, G. Cieslewski: “Reconfigurable fault tolerance: A framework for environmentally adaptive fault mitigation in space”, International Conference on Field Programmable Logic and Applications (FPL), pp. 199–204, 2009.

[BBI+12] L. Bauer, C. Braun, M. E. Imhof, M. A. Kochte, H. Zhang, H.-J. Wunderlich, J. Henkel: “OTERA: Online Test Strategies for Reliable Reconfigurable Architectures”, NASA/ESA Conference on Adaptive Hardware and Systems (AHS), pp. 38–45, 2012.

[ZBK+13] H. Zhang, L. Bauer, M. A. Kochte, E. Schneider, C. Braun, M. E. Imhof, H.-J. Wunderlich, J. Henkel: “Module Diversification: Fault Tolerance and Aging Mitigation for Runtime Reconfigurable Architectures”, IEEE International Test Conference (ITC'13) , pp. 1–10, 2013.

[ZKI+14] H. Zhang, M. A. Kochte, M. Imhof, L. Bauer, H.-J. Wunderlich, J. Henkel: “GUARD: GUAranteed Reliability in Dynamically Reconfigurable Systems”, IEEE/ACM Design Automation Conference (DAC'14) , 2014.