# Hot-Swapping Architecture with Back-biased Testing for Mitigation of Permanent Faults in Functional Unit Array

Zoltán Endre Rákossy*‡, Masayuki Hiromoto*§, Hiroshi Tsutsui*,
Takashi Sato*, Yukihiro Nakamura†, and Hiroyuki Ochi*
*Department of Communication and Computer Engineering, Kyoto University,
Yoshida-honmachi, Sakyo, 606-8501 Kyoto, Japan, Email: paper@easter.kuee.kyoto-u.ac.jp
†Research Organization of Science and Engineering, Ritsumeikan University,
1-1-1 Noji-higashi, Kusatsu, 525-8577 Shiga, Japan, Email: y-nakamr@fc.ritsumei.ac.jp

***Abstract*— Due to latest advances in semiconductor integration, systems are becoming more susceptible to faults leading to temporary or permanent failures. We propose a new architecture extension suitable for arrays of functional units (FUs), that will provide testing and replacement of faulty units, without interrupting normal system operation. The extension relies on data-path switching realized by the proposed hot-swapping algorithm and structures, by use of which functional units are tested and replaced by spares, at lower overheads than traditional modular redundancy. For a case study architecture, hot-swapping support could be added with only 29 % area overhead. In this paper we focus on experimental evaluation of the hot-swapping system from a fabricated chip in 65 nm CMOS process. Autonomous testing of the hot-swapping system is enhanced with back-bias circuitry to attain an early fault detection and restoration system. Experimental measurements prove that the proposed concept works well, predicting fault occurrence with a configurable prediction interval, while power measurements reveal that with only 20 % power overhead the proposed system can attain reliability levels similar to triple modular redundancy. Additionally, measurements reveal that manufacturing randomness across the die can significantly influence identical sub-circuit reliability located in different parts in the die, although identical layout has been employed.**

## I. Introduction

Increasing dependence on computing systems in business and life-critical applications demands reliable designs at low cost and long life-time. Concurrently, flexible, low-power, but high-performance designs are in high demand, especially in the fields of wireless communication, bio-informatics and avionics. Post-manufacturing tests like BIST, MISER [1,2] and burn-in [3], ensure defect-free delivery, but there are a host of environmental and physical effects which shorten the lifetime and reliability of the circuits. Such effects like hot-carrier degradation, single event upsets (SEU), electro-migration, and negative bias temperature instability (NBTI), get stronger with transistor scaling, low supply voltages and high operating frequencies.

While transient faults can be mitigated by various software methods, permanent faults, which damage the circuit (e.g., stuck-at-0 faults), need mitigation methods like modular redundancy: several copies of the circuit are used in parallel and their results are compared using a majority voter at the cost of increased area and power. Advanced versions employ additional spare units to protect the already replicated units, and have been proposed for over 30 years (hybrid redundancy) [4].

This gives conflicting requirements for high-performance and high reliability at low cost, which are hard to satisfy. In this paper, we address both, by employing a coarse-grained reconfigurable array (CGRA) and proposing a way to guard it against permanent faults. In CGRAs, several large configurable functional units (FU) are connected with a programmable interconnect, representing the perfect balance between flexibility and computational power. The lower amount of configuration bits and bus-interconnect make it less flexible than FPGAs, but several parallel FUs yield better performance than general purpose processors. Guarding such a structure from permanent faults with traditional methods like triple modular redundancy (TMR) would be unfeasible.

Enhancing array yield in the presence of faulty processing elements by autonomously reconfiguring the data path has been researched in works like [5,6]. A good example of an FPGA-based fault tolerant system is presented in [7]. On-line testing of free portions of the FPGA is done to detect and repair permanent faults, then active portions are moved via partial run-time reconfiguration to test locations which were in use before. Such solutions either come with huge resource costs, require pausing system operation while recovery is performed, presume 'golden' parts, or are limited to FPGAs.

The proposed architectural extension for CGRAs, targets mitigation of permanent faults due to aging during circuit life-time. While being realized at lower overheads than modular redundancy, it provides additionally automatic detection, testing and replacement of faulty units based on the concept of hot-swapping, without needing to suspend execution of running applications. The proposed hot-swapping algorithm and the structures which implement it, test and replace units with spares transparently, successively swapping spares with FUs in-use, until full FU test coverage is attained. The testing is performed on those units which are swapped out of the active data-path, keeping execution uninterrupted. The system is designed to be scalable to large arrays and has a constant implementation area cost per additional FU.

‡Presently with RWTH Aachen University, Germany.
§Presently with Panasonic Corporation, Japan.
978-3-9815370-0-0/DATE13/©2013 EDAA

A proof-of-concept chip has been fabricated in a 65 nm CMOS process for a CGRA of 12×6 FUs. In this chip, fault detection employs N-well bias selectors to raise the bias voltage of FUs under test, to predict permanent faults due to aging before manifestation. Experimental results prove that each 0.1 V back-bias increment translates into 4 MHz prediction window. This concurrent testing method enables detection of faulty FUs before the aging effects can trigger faults and serves as an early detection system preventing manifestation of errors at circuit run-time.

In Section II we will make a description of the proposed algorithm and its structures, followed by a reliability analysis in Section III. Sections IV and V will focus on implementation and experimental data.

## II. Hot-Swapping architecture extension

Hot-swapping is a concept used to describe the function of removing or replacing system components while the system is still in operation, without disruption or any effect on the operation being carried out.

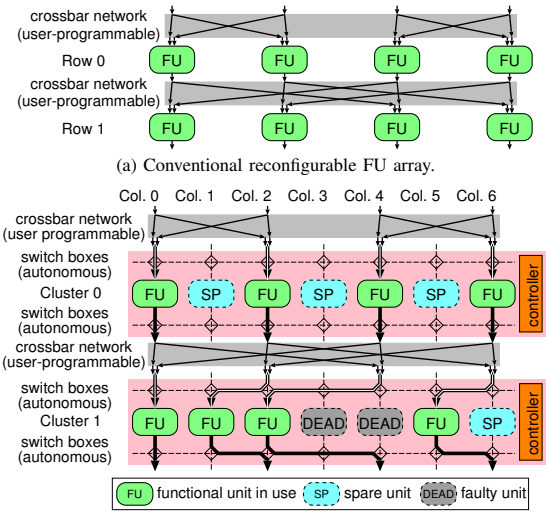
### A. Structure

Applying the hot-swapping concept to a regular structure such as CGRA, high reliability can be achieved at low cost, if some of the FUs of the array are assigned as spares, which then can logically roam the array and functionally replace faulty units. Starting from an initial CGRA structure shown in Fig. 1(a), the necessary structural units to realize the hot-swapping system are described as follows.

The basic structural unit is the *cluster* as shown in Fig. 1(b). The cluster realizes efficient spare sharing, testing and hot-swapping at lower area and power cost than TMR. After application mapping on the CGRA, any unassigned FU within a cluster group can take the role of a spare unit shared among all assigned FUs, improving reliability. Every cluster contains a *switch matrix* and a *controller*, which are responsible for seamless testing and replacement of faulty units. The role of a spare unit is hot-swapped across the cluster, enabling testing of units in use by the application, without interrupting the application and ultimately allowing all FUs access to a spare, without employing a complex routing network.

Unlike TMR, which requires two additional units to protect the function of one, the cluster allows sharing of one spare to all cluster members, minimizing area and power overhead. When no spares are assigned or none are available due to FU failures, the hot-swapping system is disabled.

Spare sharing (data-path switching) is realized by the *switch matrix*, a structure which consists of leading and trailing switch boxes for each FUs in the array. Each switch box can redirect data in four ways: straight (north to south), left (north to west and east to south), right (north to east and west to south) and pass-through (east to west or west to east) as shown in Fig. 2. Straight state binds an FU at the same column as input and output ports. Left- and right-redirect states are used to bind an FU of a different column than the input and output ports. Pass-through state is activated when multi-hop redirects are needed over disabled FUs (Fig. 1(b) cluster 1).




(a) Conventional reconfigurable FU array.

(b) Clustered FU array with the hot-swapping switch matrix.

Fig. 1. Hot-swapping architecture extension. Cluster 0 has three spares and cluster 1 has two faulty FUs. Switch matrix adapts accordingly, while keeps testing for other faulty units while spares still exist.

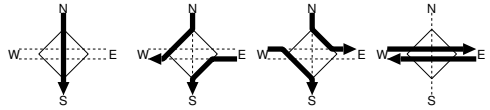



Fig. 2. Internal states of a switch box.

### B. Hot-swapping algorithm

The proposed hot-swapping algorithm is implemented in the *controller*, which controls the switching, assigns testing of the units and keeps track of faulty units.

The top-level finite state machine of the controller is shown in Fig. 3. In the CONFIG state (07), the controller receives parameters, such as the columns of data-in ports used within the cluster and the intervals of testing and idle states. In IDLE state (00), the controller disables the spares, and enters a sleep state for an arbitrary period. Testing (and hot-swapping) is comprised of states 01-05, executed independently for each FU. Tracking of test results, available spares and their location, but also setting valid switch-box states for each FU is non-trivial, due to combinations of 'faulty', 'spare' or 'in-use' states of the FUs within the cluster. The number of possible states increases exponentially to the number of FUs in a cluster and one change (exchanging spares) can cause switch-box state recursive dependency.

Therefore, the proposed algorithm makes switching state decisions, tracks spares and tests FUs in a way that the state of one FU switch-box only depends on adjacent unit state. In this way, clusters can be scaled to any size and mapping to hardware results in simple combinational logic.

The listing in Fig. 4(a) elaborates on how the algorithm independently checks and migrates the spares during FSM states 01-05 recursively across the cluster in two migration phases (left: L1, right: L2). Moves are done only if the switch can accommodate it without breaking the data-path.

For clarity, the testing phase is shown in Fig. 4(b), for

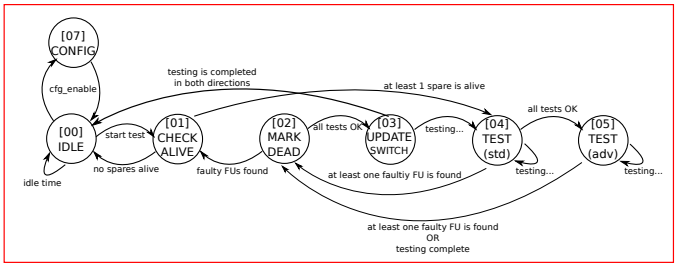
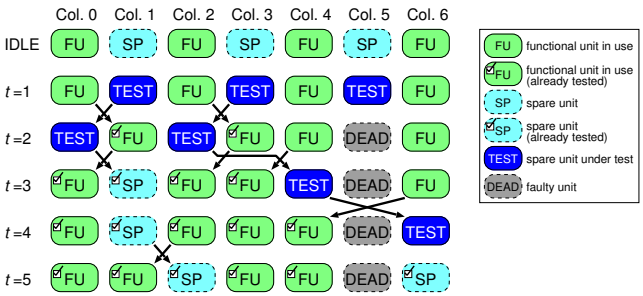



Fig. 3. Controller core finite state machine.

```
CONF: download parameters
IDLE: sleep for time t, power off spares -> L0
L0:
IF there are any units marked as spare
    test them; mark any faulty spares as dead;
ELSE ->IDLE;
FOR ALL spares that passed L0 test:
L1: (left spare migration)
    IF no spares can move left -> L2
        ELSIF funit(left)=used;
            IF switching state allows it
                swap spare with FU;
                test new spare;
                IF test=ok -> L1;
                ELSE mark dead, -> L0;
            ELSE -> L1;
        ELSE -> L1;
L2: (right spare migration)
    IF no spares can move right -> IDLE;
        ELSEIF  funit(right)=used;
           IF switching state allows it
                swap spare with FU;
                test new spare;
                IF test=ok -> L2;
                ELSE mark dead, ->L0;
           ELSE -> L2;
        ELSE -> L2;
```

(a) Algorithm details during independently testing each spare. Two migration phases ensure full test coverage while switch-box states allow it.




(b) Time-lapse of the algorithm for an example cluster (in-use:spare 4:3).

Fig. 4. Hot-swapping algorithm for testing and replacing faulty units.

a cluster of four active FUs with three spares. At $t$=0, the controller is in IDLE state. When a testing cycle begins at $t$=1, first all available spares are tested. Already tested units are marked in Fig. 4(b) with a check box. In this example, the third spare is deemed faulty, so it will be marked as 'faulty', while the other two spares are hot-swapped with the left-adjacent in-use FUs, shown by the arrows. After testing the FUs at $t$=2, the spares cannot be swapped to the left anymore, so they will shift to the right, to test all FUs within the cluster. From $t$=3 to $t$=4 it is required to jump over the faulty FU, the data-path being rerouted using the 'pass-through' state of the switch-box above the faulty FU, completing the testing cycle. The algorithm stops if all active FUs have been tested returning to IDLE and spares can be turned off.

### C. Testing procedure

Testing occurs by applying a set of patterns to detect faults, some of them forcing traversal of the critical path of the FU. Since testing is periodical, any error that occurs between these testing cycles remains undetected until the next cycle. To solve this vulnerability, early fault detection is employed. Permanent faults, such as those caused by hot carrier degradation effect occur in time, often with graceful degradation. Therefore, forcing harsher operating conditions during the test cycle, such as applying back-bias voltages would cause any aged circuits to manifest faults during the test-phase. If tests complete successfully even under forced operating conditions, the tested FU can be considered 'fit' enough to operate without faults until the next test cycle, providing a dependability buffer. Hardware-wise such testing requires an additional supply voltage grid and test-pattern storage or generation, described in detail in Section IV.

## III. Reliability Analysis

To further highlight the flexibility of the hot-swapping system, a reliability analysis was conducted in comparison with TMR in the following.

### A. Basic considerations

For one component of a system, reliability depends on the cumulative distribution function (CDF) constructed from the lifetime distribution model, or probability density function (PDF) of the component. Such a function models the probability of failure due to different possible failure mechanisms of the component.

In our analysis we employ the exponential PDF model (Eq.1) but complex models can be constructed given accurate failure mechanism models which are strongly dependent on the process technology for integrated circuits.

$$f(t) = \lambda e^{-\lambda t}, \tag{1}$$

where $\lambda$ is the failure rate, representing the combined effect of all failure mechanisms.

### B. Reliability model

To calculate overall reliability of one system, consideration of the combined effect of the reliability of each component $R_c$ is needed. Both TMR and the Hot-Swapping cluster are $k$ of $n$ models, which mix series and parallel components, described by Eq.2.

$$R_{kofn}(t) = \sum_{i=k}^{n} \binom{n}{i} \left[R_c(t)\right]^i \left[1 - R_c(t)\right]^{n-i} \tag{2}$$

TMR uses a 2 out of 3 model combined with the effect of the voter. All three units are constantly operating to mask one fault.

$$\begin{aligned} R_{TMR}(t) &= R_{vot} \sum_{i=2}^{3} \binom{3}{2} \left[R_{FU}(t)\right]^2 \left[1 - R_{FU}(t)\right]^1 \\ &= R_{vot} \left[R_{FU}^3 + 3R_{FU}^2 \left(1 - R_{FU}\right)\right] \\ &= R_{vot} \left(3R_{FU}^2 - 2R_{FU}^3\right) \end{aligned} \tag{3}$$

The proposed hot-swapping system is also a $k$ out of $n$ model, however the spares are on warm or cold stand-by (are not necessary for system operation). For the time the spares are turned off, the failure rate is typically reduced (e.g. $\lambda_{off} = 0.1\lambda_{on}$). The total system lifetime for such a system is the sum of $n$ identically distributed random lifetimes, but with each component having its own CDF. In case of exponential lifetime distributions, the PDF follows the gamma function (Eq.4)

$$f_n(t) = \lambda^n \frac{t^{n-1} e^{-\lambda t}}{(n-1)!} \tag{4}$$

Given a set of $n$ identical elements with $n-k$ cold spares, the system reliability is the sum between the reliability of $k$ working elements and the incremental reliability of $n-k$ spares. Therefore, the reliability of $k$ active elements $R_0$ is the series reliability, for identical elements. One spare element reliability takes the different $\lambda_{off}$ into account(Eq.5).

$$R_0(t) = R_{FU}^k = \left(e^{-\lambda t}\right)^k;\ R_{spare}(t) = e^{-\lambda_{off} t} \tag{5}$$

The incremental reliability of each spare for element $i$ ($i = \overline{1, n-k}$) is shown in Eq.6.

$$R_i(t) = \frac{R_{i-1}(t)}{i} \left[ \left( (i-1)\,\lambda_{off} + k\lambda_{on} \right) \frac{1 - R_{spare}(t)}{\lambda_{off}} \right] \tag{6}$$

Hence, the overall system reliability of a stand-by $k$ of $n$ system is shown in Eq.7.

$$R_{standby}(t) = \sum_{i=0}^{n-k} R_i(t) \tag{7}$$

By using Eq.7 and considering the switch and controller reliabilities, the Hot-Swapping cluster reliability equation shown in Eq.8 can be calculated.

$$R_{hswap}(t) = R_{switch}(t) \cdot R_{ctrl}(t) \cdot \sum_{i=0}^{n-k} R_i(t) \tag{8}$$

To make a comparison, we assumed some typical values for $\lambda$, considering one failure in $10^6$h for $\lambda_{FU}$, $0.1\lambda_{FU}$ for the controller and off-line spares, and $0.3\lambda_{FU}$ for the switch matrix and the TMR voter. Controller failure late is low due to complete error correction code-protection of its functionality in the implementation.

Figure 5 plots reliability decrease of simplex FU, TMR and various hot-swap combinations (in-use+spare). The lower continuous line is the simplex FU (no protection), while the highest one is for the turned off FU. When many units have to be protected by a single spare, the Hot-Swapping reliability goes below that of TMR and even below of the unprotected unit. Adding an extra spare yields high reliability values for early mission times even if there are many protected units. For one spare protecting two units, similar reliability can be gained as for TMR. For an equal or higher number of spares, the Hot-Swapping has superior reliability than both TMR and simplex, even for longer mission times, based on the current assumptions.

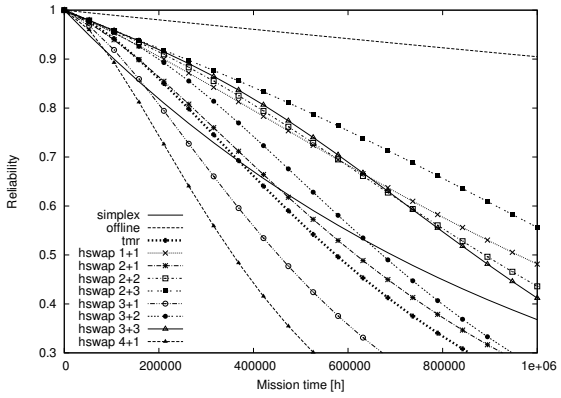



Fig. 5. Reliability plot for various hot-swap (in-use+spare) configurations

## IV. Implementation

To evaluate the proposed algorithm and testing scheme we have chosen a medium-sized homogeneous CGRA of 12×6 FUs. The interconnect of the CGRAs is a cascade style crossbar, allowing outputs of one row to connect to the inputs of any FU of the next row. The array ends are connected to a 512-word dual combined-port data memory, split into 6 banks of 64-bit wide words serving 2 FUs each. FU instruction set contains arithmetic operations (ADD, ADD-immediate, SUB, SUB-immediate, MULT, MULT-immediate-and-ADD, DIV) and logic operations (NOT, AND, OR, EX-OR, SHIFT-Left, SHIFT-Right, and NOP). All FUs can take up to two operands and yield one output value of 16-bit word size. Application-specific array interconnect and FU op-codes are user-configurable. Such a CGRA could execute data-flow centric applications such as FFT, filtering and video streaming applications.

The array is clustered into 6 clusters and for each cluster the hot-swapping structures are added (Fig. 6). The switch matrix is placed between the user-configurable interconnect and FUs, thus input operands are routed to the correct FU within the cluster, along with the respective opcode during data-path switching. For our case study architecture, separate SRAMs were assigned to each controller of a cluster, such that individual sets of test patterns can be applied to each cluster. In a real-world application case, compressed test patterns or a generator can be shared among all clusters for area efficiency. The test RAMs contain 512 patterns, tailored for stuck-at faults and critical path traversal for each FU opcode. New Test RAM contents can be uploaded during run-time if special testing is desired.

Additionally to the test patterns, fault detection is enhanced by a back-bias circuit, which forces less favorable conditions to the FUs under test, for early detection. Figure 7 shows the schematic of an N-well bias selector, which consists of an inverter, two cross-latch level-shifters, and two high-voltage PMOS transistors. When SEL is low level, bottom PMOS turns on and VNW is driven to VDD level (=1.2 V). Otherwise, top PMOS turns on and VNW is driven to VNWIN level, which is given a little higher voltage than VDD (e.g., 1.3 or 1.5 V).

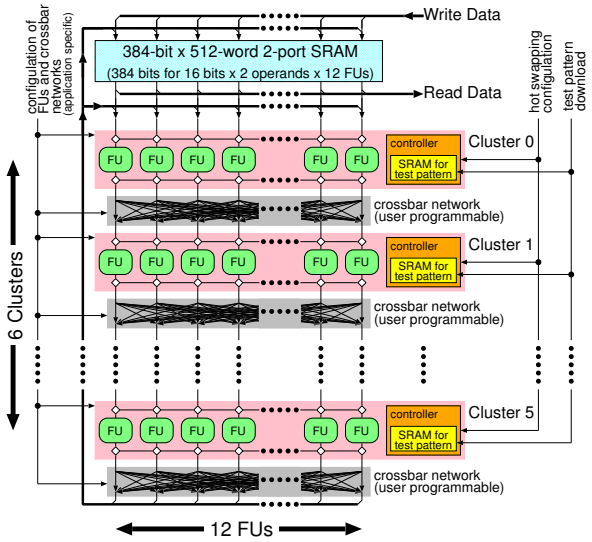



Fig. 6. Overview of the cluster array.

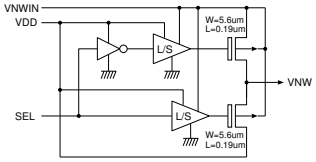



Fig. 7. N-well bias selector.

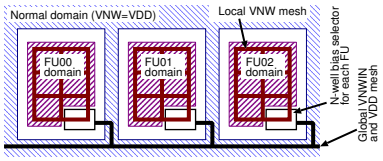



Fig. 8. N-well voltage domain.

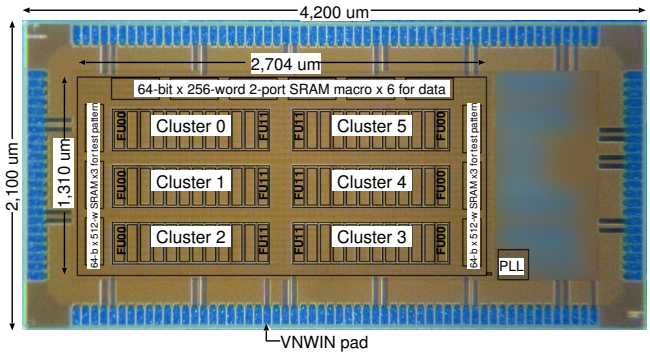



Fig. 9. Chip micrograph.

TABLE I

AREA OVERHEAD FOR HOT SWAPPING.

| | |
|---|---|
| **(a) cluster-level core components** (FU×12) | **189,252** $\mu$m$^2$ |
| **(b) cluster-level hot swapping components** | **132,552** $\mu$m$^2$ |
| N-well bias selector×12 | 1,104 $\mu$m$^2$ |
| switch boxes + state machine | 93,805 $\mu$m$^2$ |
| test pattern SRAM | 37,643 $\mu$m$^2$ |
| **(c) cluster-array core components** | **2,746,928** $\mu$m$^2$ |
| (a)×6 | 1,135,512 $\mu$m$^2$ |
| data SRAM | 264,582 $\mu$m$^2$ |
| other (crossbar network etc.) | 1,346,834 $\mu$m$^2$ |
| **(d) cluster-array hot swapping components** ((b)×6) | **795,312** $\mu$m$^2$ |
| **cluster-array-level overhead ((d)/(c))** | **29 %** |
| **(e) useful 8×6 cluster-array core components** | **2,083,621** $\mu$m$^2$ |
| **(f) hot swapping comp. incl. spares** (d)+FU×24 | **1,173,816** $\mu$m$^2$ |
| **cluster-array-level overhead incl. spares (f)/(e)** | **56 %** |

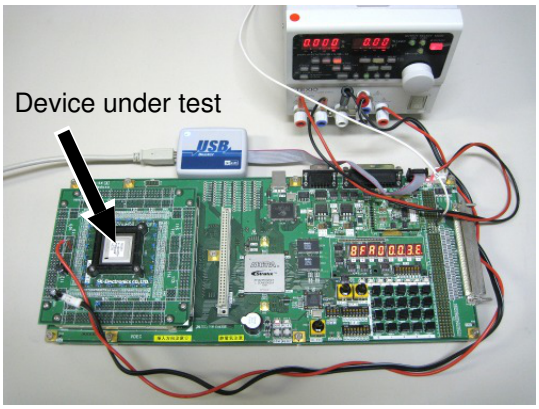



Fig. 10. Measurement setup.

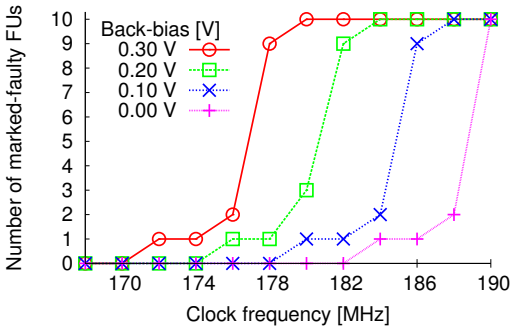



Fig. 11. Measured number of marked-faulty FUs in a cluster.

Each of 72 FUs has its own N-well bias selector, so that N-well voltage can be controlled individually as illustrated in Fig. 8.

The prototype chip is fabricated in a 65 nm CMOS process using vendor's standard cell library (typical, 1.2 V, 25°C), with target operating frequency of 190 MHz. Hierarchical synthesis and layout was employed. Figure 9 shows the micrograph of the fabricated chip. The die size is 4,200 $\mu$m×2,100 $\mu$m in which the active area for the architecture except I/O and PLL macro is 2,704 $\mu$m×1,310 $\mu$m.

Table I shows the additional area (overhead) due to the addition of hot swapping support structures, which amounts to only **29 %** for a CGRA of 12×6 FUs. In the case of TMR, a failure of 33 % can be tolerated (1 replica out of 3). Reliability analysis done in Section III reveals that if in a cluster 33 % are spares, similar or better reliability can be achieved, since 33 % failure rate can be tolerated. In the hot-swapping cluster, the spares are not bound to any particular FU, tolerating multiple closely localized faults, which for TMR is fatal. Hence, 4 spares are needed to protect 8 FUs in a 12 FU cluster to cover TMR reliability. Counting these spares also as 'overhead' area, resulting total overhead is only **56 %**. Note that this is much smaller than TMR which needs additional 200 % area and power for a protected FU, not including a voter.

## V. MEASUREMENT RESULTS

*Measurement setup and procedure:* Figure 10 shows the measurement setup. The fabricated chip in QFP package is mounted on MU200-SX FPGA board using a daughter-card. The FPGA runs a test-bench which gives the necessary control signals and reads chip outputs. The FPGA analyzes the output

TABLE II
EXPERIMENTAL POWER CONSUMPTION DATA (170MHZ, AT 27°C).

| Input | Cluster configuration | Power (mW) | Cost |
|---|---|---|---|
| none | Leakage power (idle clock signal) | $< 1$ | |
| none | PLL only, 170MHz | 49 | |
| Static data | Hotswap: 8 FUs + 4 NOP/clus. | 54 | |
| | Hotswap: 8 FUs + 4 SP/clus. | 181 | 3.4× |
| | TMR: 8 FUs + 16 replicas/clus. (est.) | 108 | $> 2\times$ |
| Busy data | Hotswap: 8 FUs + 4 NOP/clus. | 420 | |
| | Hotswap: 8 FUs + 4 SP/clus. | 498 | 1.2× |
| | TMR: 8 FUs + 16 replicas/clus. (est.) | 840 | $> 2\times$ |

data and signals results on the on-board LEDs for quick evaluation.

Measurements have been conducted in two stages: critical path test with fault detection and power evaluation tests.

*Critical path testing:* One cluster was evaluated at a time while the rest have hot-swapping disabled. Cluster configuration used 2 FUs for processing and 10 spares were assigned for fault detection test. Tests occur every 64 k cycles (configurable) and one complete test takes max. 6 k cycles (variable based on available spare amount). Under critical path stress-test inputs, frequency was increased gradually to simulate aging. 0 V back-bias results are compared with 0.1 V incremental steps of back-bias voltage to observe the operation of the early fault detection mechanism. Figure 11 shows how each 0.1 V increment in back-bias voltage predicts a fault 4 MHz earlier than the actual fault frequency value. This can be directly translated into circuit lifetime. In a 190 MHz design, 4 MHz (or 2%) performance margin is worth monitoring, since a 2% performance loss can happen even within one hour using worst case input patterns, triggering faults. It is reported that static NBTI causes an exponentially steep degradation in early circuit life. In 65 nm process technology, a static NBTI causes 10 mV $V_{TH}$ degradation in one hour [8], which causes 2% performance degradation [9]. As NBTI degradation saturates exponentially, the circuits with a larger margin will have a much longer life-time. A 4 MHz early warning would support a quick-fix in early life and graph an estimated fail time in late circuit life.

Another interesting fact to note is that for the same kind of FU on the same die, fault manifestation times differ. Some FUs are 'weaker' while some last longer before fail. This is due to manufacturing randomness since exactly the same layout has been used for all FUs (hierarchical synthesis). In such a case early detection of faulty units is especially useful.

*Power evaluation:* Table II shows the measured power consumption for 8 used FUs in a cluster, for two data sets: static data (low switching activity) and busily switching data (high switching activity). The datasets for dynamic power have been experimentally determined across multiple measurements, taking the set that attains maximum power. To measure how much the hot-swapping structure uses, comparisons are done in Table II between the case where hot-swapping is disabled or enabled, for both data sets. For a conservative comparison, TMR power consumption has been estimated from the disabled case, without considering voter power and glue logic for splitting input data, which can significantly contribute to power consumption. It can be observed that for low dynamic power, hot-swapping increases dynamic power significantly due to swapping the FU data across the cluster. When high switching activity data is used, with only 20 % extra power, hot-swapping can provide similar protection as TMR. These comparisons are extremely conservative, as no power/clock gating structures are implemented for the spares and each cluster contains its own test-SRAM. We are confident that an optimized version with the possibility of turning off spares and faulty units, and/or with shared test patterns across clusters would yield even better results.

## VI. CONCLUSIONS

In this paper, implementation and measurement work is presented for an automatic permanent fault detection and functional restoration for coarse-grained reconfigurable architectures. The proposed system employs the concept of hot-swapping, allowing testing, early detection and replacement of faulty units in an array at lower cost than traditional TMR, from both area and power perspective. Early detection of faults due to aging effects is realized by the addition of a back-bias circuit which can flexibly predict circuit failure with an arbitrary margin. A chip has been designed, fabricated and tested to evaluate the proposed system and measurement results highlight the advantages of the hot-swapping system. Also, measurements revealed that even with identical layout and same physical substrate, the reliability of sub-circuits greatly varies, underlining the need for early fault detection and repair systems.

## ACKNOWLEDGMENT

This research was performed by the authors for STARC as part of the Japanese Ministry of Economy, Trade and Industry sponsored "Silicon Implementation Support Program for Next Generation Semiconductor Circuit Architectures". Also it is part of a JST CREST project, and is supported by VDEC, the University of Tokyo in collaboration with e-Shuttle, Inc., Fujitsu Ltd., Synopsys, Inc., Cadence Design Systems, Inc., and Mentor Graphics, Inc.

## REFERENCES

[1] B. T. Murray and J. P. Hayes, "Testing ICs: Getting to the core of the problem," *IEEE Computer*, vol. 29, pp. 32–38, 1996.
[2] H. Al-Asaad and J. P. Hayes, "Logic design validation via simulation and automatic test pattern generation," *J. Electron. Test*, vol. 16, pp. 575–589, 2000.
[3] E. Wu, J. Sune, W. Lai, E. Nowak, J. McKenna, A. Vayshenker, and D. Harmon, "Interplay of voltage and temperature acceleration of oxide breakdown for ultra-thin gate oxides," *Solid State Electronics*, vol. 46, no. 11, pp. 1787–1798, 2002.
[4] D. Siewiorek and E. J. McCluskey, "An iterative cell switch design for hybrid redundancy," *IEEE Trans. Comput.*, vol. 100, no. 22, pp. 290–297, 1973.
[5] M. Fukushi and S. Horiguchi, "A self-reconfigurable hardware architecture for mesh arrays using single/double vertical track switches," *IEEE Trans. Instrum. Meas.*, pp. 357–368, Apr. 2004.
[6] S. Y. Kung, S. N. Jean, and C. W. Chen, "Fault-tolerant array processors using single track switches," *IEEE Trans. Comput.*, vol. 38, pp. 501–514, 1989.
[7] M. Abramovici, J. M. Emmert, and C. E. Stroud, "Roving STARs: An integrated approach to on-line testing, diagnosis, and fault tolerance for FPGAs," in *Proc. NASA/DoD Workshop on Evolvable Hardware*, 2001, pp. 73–92.
[8] W. Wang, S. Yang, S. Bhardwaj, R. Vattikonda, S. Vrudhula, F. Liu, and Y. Cao, "The impact of NBTI on the performance of combinational and sequential circuits," in *Proc. DAC*, Jun. 2007, pp. 364–369.
[9] W. Wang, S. Yang, S. Bhardwaj, S. Vrudhula, F. Liu, and Y. Cao, "The impact of NBTI effect on combinational circuit: modeling, simulation, and analysis," *IEEE Trans. VLSI Syst.*, vol. 18, no. 2, pp. 173–183, Feb. 2010.