Douglas Cunningham, Petra Hofstedt, Klaus Meer, Ingo Schmitt (Hrsg.): INFORMATIK 2015
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015

Detection and Correction of Logic Errors Using Extra Time Slots

Davide Dicorato, Heinrich T. Vierhaus

Abstract: Digital integrated circuits fabricated in nano-technologies have first shown to be more vulnerable to transient errors effects than their predecessors. But they also show effects of stress-induced defects resulting in early life-time failures. In general, power dissipation problems and dielectric stress, due to high field strength, are the main reasons for shortened life-time expectations. On the other hand, system designers require highly reliable and long-time dependable hardware, for example in automotive applications. On-line error detection and-compensation using either codes or, in the more general case, double or triple modular redundancy (DMR and TMR), has been used for decades, but causes higher power dissipation in nano-logic, additional stress, and is therefore no cure in terms of life-time extension. Savings on hardware and power are possible, if resources can be re-allocated to produce local TMR upon demand. However, such techniques may cause sudden signal delays after the detection of errors, which are not easy to handle in synchronous systems. In this paper we present a pseudo-TMR approach, which has little influence on timing in the "good case" and performs a regular error correction within 3 extra clock cycles under error correction without limits on the fault model.

Keywords: Fault tolerant computing, error correction, pseudo-TMR

1 Introduction

New and enhanced fault effects to occur in nano-electronic circuits have been predicted for more than a decade [Ba05,Br04, Fa98,Li09, Lie05]. Intensive research has been done also in the area of aging mechanisms, which harm the switching speed of digital systems and may finally result in delay faults [Al05]. While most research has gone into technologies that may handle either extra delays or transient fault effects [Mi05,Mi06], handling of permanent faults caused e. g. by dielectric breakdowns [Ba09] or by metal migration [Lie05,Lu04] has found much less attention so far. Methods of fast error correction, primarily developed for transient faults such as triple modular redundancy (TMR) or error correcting codes [Pr96,La01], will also work for a limited number of permanent faults, but will not be able to handle transient faults on top of permanent faults in arbitrary combinations. Some research has gone into schemes that provide triple modular redundancy (TMR) by multiple usage of the same elements in time [She09,Ga00] or into schemes that apply TMR selectively only to specific signals [Goe11]. For the repair of permanent faults, repair technologies based on cold redundancy [Ko12,Ko11] are not very fast, since the repair mechanism has to include error detection, fault diagnosis, redundancy allocation and eventually a validation. Such a process may take milliseconds at best and can possibly be part of a system start-up process. Hence on-line available "hot" redundancy is necessary for the detection and compensation of permanent faults that occur in hot operation, followed by an off-line repair process. In order minimize the total overhead associated with fast error compensation on one side and self repair on the other hand, compatible schemes that can somehow share the overhead have been proposed [Ko12]. The necessary overhead is triplication at least, if all errors have to be compensated in the same clock cycle. If, however, delays of a few extra clock cycles can be accommodated, then a basic scheme of "virtual" triplication is possible, which is based on sharing redundancy between functions that use the same hardware resources [Ko13,Ko14]. Clock management specifically for pipeline structures with on-line detection and correction of delay faults have been dealt with in previous work [Bl04,Bl09, Bl13] and can partly be used for on-line error correction on a wider scale. The basic shortcoming of such methods is the high overhead for the extra circuitry for delay detection and glitch filtering, which causes an overhead of more than 100 % when applied to all signals in a processor design [Bl13].

2 On-Line Error Correction by virtual TMR

A re-configurable logic block (RLB) is considered first. A total of n functions is executed on functional units (FUs) of identical nature. If we protect all functional units by full triplication (fig. 1), the system will be able to handle transient and, with certain limits, permanent faults.

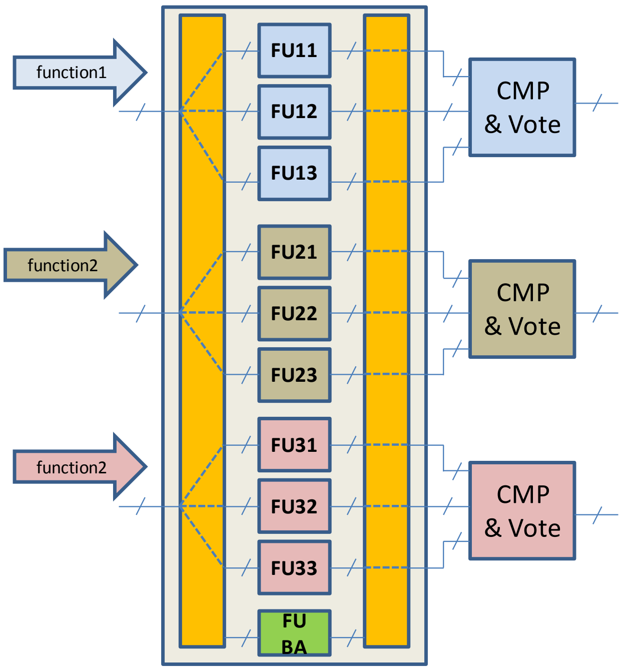

Figure 1. Re-configurable block (RLB) with full TMR and BISR capabilities

With extra power and hardware involved, extra thermal stress may cause wear-out induced faults earlier than in the single system. Then it is still possible to add a limited extra amount of redundancy for the off-line repair of permanent faults. Arrays of switches are needed for the re-location of functions to functional units (FUs). By shifting inputs and outputs between just 2 adjacent FUs, we can use the backup block (FUBA) replace any other defect block (FUkn) in the scheme in an indirect manner. The switching process has to be done off-line in a separate repair mode under either hardware or software control. Hence the existing TMR also has to secure the handling also of permanent faults that occur during normal operation. The drawback of this scheme is the more than triplication in hardware and power. A more cost-effective scheme is shown in figure 2. Now all functions run on duplicate units only. In case of a "fault detect", a third functional unit has to be "borrowed" from a neighboring function, which will then temporarily operate without backup (figure 2).

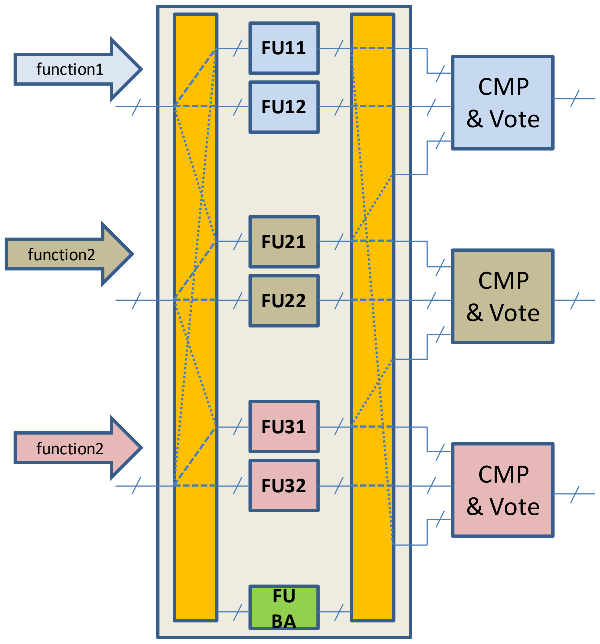

Figure 2. Re-configurable block (RLB) with duplication and selectively "borrowed" TMR

While the overhead now is now essentially determined by duplication, we need an on-line reconfiguration process by switching functions. The comparison of alternatives is shown in Table I. The essential drawback is the demand for extra timing which arises randomly after a "fault detect". Table I compares as alternatives full TMR (fig. 1), a TMR approach with full backup units, where the 3 rd unit is only activated after a "fault detect", and the alternative using borrowed resources from neighboring functions (figure 2).

	Act. TMR	Pass. TMR	Pseudo-TMR
Redund. Overh.	> 200 %	> 200 %	> 100 %
Power Overh.	> 200%	> 100 %	> 100 %
Switching	slow	fast	fast
Timing	regular	irregular	irregular

Table 1. Comparison of TMR-based Architectures

Essentially, it is possible to achieve TMR-capabilities with a reduced overhead in power and hardware, but this goes at the expense of extra clock cycles, which are needed to establish, use, and evaluate temporary TMR. Then the question is, how digital designs can handle "sudden" delays, and how this type of selective TMR can be organized in a robust control scheme. Some previous experimental studies showed that allocation the switching process at inputs and outputs plus the operation of a functional block in one clock cycle is possible, but requires very careful timing and results in reduced clock rates [Mu14]. Therefore we investigated on an alternative approach, where clock cycles need not be stretched in normal operation, but alternatively extra clock cycles are added only in case of fault events.

3 Handling Extra Clock Cycles

Previous research on the detection and correction of delays in pipeline structures has dealt with clocking schemes in several ways [Bl04,Bl09,Bl13]. A global control signal that will "freeze" the clock for one (or more) cycles in order to perform a re-calculation after "fault detect" is one way, which can work with edge-triggered flip-flops and a single clock [Bl04,Bl09].

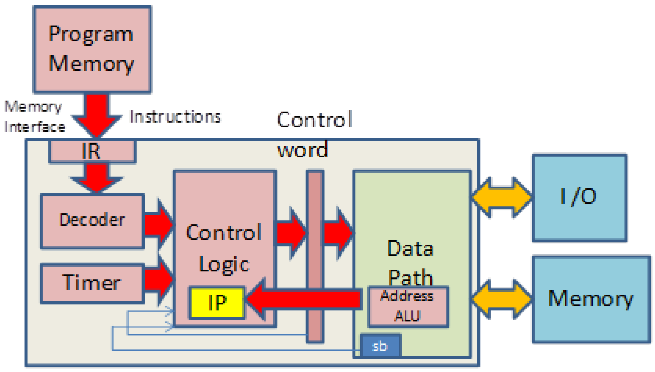

Figure 3. Critical functions for control errors in a typical processor architecture

Specific for the delay fault correction scheme is the problem of “fast” next signal transitions, which may arrive early in the next clock cycle, while the latch that may catch late transitions is still open. The final solution for this problem has been found by the “bubble razor” architecture, which uses a pipeline structure with two non-overlapping clocks, which are alternatively applied to successive pipeline stages [Bl13]. While control logic seems to be less prone to faults induced by path delays [Bl04], it will also need capabilities for on-line error detection and correction. In a typical processor architecture (figure 3), we can define partial functions which are highly control-flow relevant and will therefore need supervision and error correction. These are the instruction memory interface, instruction decoder, the control logic itself, and the ALU needed for address generation. Errors in data path elements not involved in control operations may be tolerated more or less, depending on the application. The methods developed for delay faults in pipelines [Bl04,Bl09,Bl13] are mainly suited for pipelined data paths. Their capability to detect also transient errors has been shown [Bl09], but they are not suited to cover permanent faults. Furthermore, the necessary overhead to protect all signals in a pipeline this way becomes prohibitively high [Bl13]. Assuming delay faults, permanent and intermittent faults as the prime concern, the more universally applicable TMR-based schemes seem to show superior properties. The methods of clock control developed in [Bl04,Bl09,Bl13], however, may also be applied for timing control in partial TMR schemes, which need one or up 3 clock cycles for organization of fault tolerance after an “error detect”. Unlike faults that occur in a pipeline structure, control logic faults should implement a “freeze” function on a local clock network, while a majority vote is performed. The problem then is to organize a repair function which safety executes the majority vote based on “borrowed” resources, while the rest of the system is stalled. The repair process needs to be implemented under hardware control in a robust timing scheme at minimum extra delay for normal operation.

4 Clock and Timing for Borrowed TMR

The simplified version of a pseudo-TMR stage is shown in figure 4. Configuration switches are omitted. We assume master-slave flip-flops at inputs and outputs, composed of D-latches. These latches need specific clock signals upon a “fault detect” condition, which is triggered by comparing the outputs of e. g. blocks FU11 and FU12.Thís signal sets a fault-latch Fl. The outputs from blocks FU11 and FU12 are also latched in the master-latches at the output. The MC-elements, however, pass these signals to the slave latch only, if they are equal, assuming a correct operation. Otherwise the MC-stages show “high impedance”. Then an “error detect” signal from an ongoing calculation must be available, before the output slave latch becomes active and can absorb the faulty input from the master. Subsequently, the master latches of the flip-flops feeding the inputs will have to deliver the same input again for the additional calculations that deliver a correct result by majority vote. This is achieved by manipulating the clock signals applied to the flip-flops at feeding inputs and at the outputs selectively.

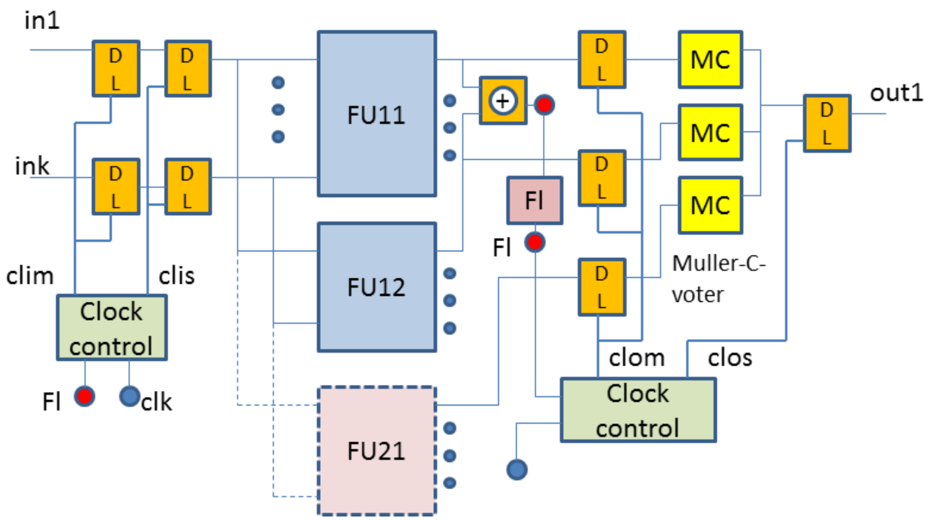

Figure 4. Signal control for fault correction using a third borrowed functional unit (configuration switches omitted)

The "fault detect" signal must arrive early enough to trigger this action. We assume that a fault condition is detected by comparison in clock cycle 0 in time before the high-low-transition. The circuit shown in figure 3 may serve as an example, assuming a faulty behavior in unit FU12.

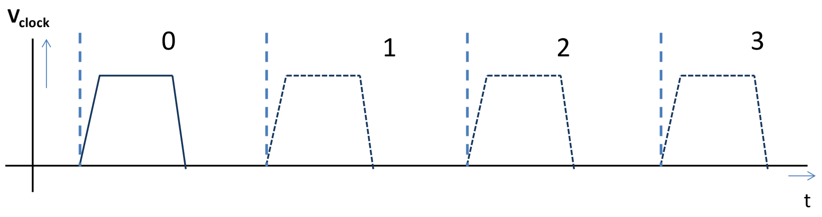

Figure 5. Timing for error detection and compensation

Then the "fault" signal, which is stored in a special fault latch Fl, must prevent false outputs from being transferred to the slaves at the outputs. All slaves receive an inverted "clock high" normally, which is set to low after the fault detect. Next a "fault counter" for detailed control of further timing signals is started, which serves to control the re-configuration and repair action. In phase 1 (figure 5), with all flip-flop outputs frozen, the configuration switches are set in order to re-allocate the third unit from the neighboring function, i. e. FU21 from function 2. This activity should be finished well before the low-high transition of clock phase 1. Then the frozen outputs of the source flip-flops remain attached to the "native" functional units (e. g. FU1, FU2) plus the additional unit FU21, which is the "decider". Then the time from the falling edge of cycle 1 to the falling edge of cycle 2 is used for re-calculating the triple output and performing a majority vote, e. g. by using a Muller-C-voter. This output is then stored in the output latches right after the high-low transition in phase 2. At the falling edge in clock cycle 2, the correct output is available in the output slave latches. The “fault” signal gets a reset. The next activity, from the falling edge of clock 2 to the rising edge of cycle 3, should be used to re-allocate the switches, e. g. connecting FU21 back to function 2. This can be done while the next input pattern, which was in the inputs of the master-latches at inputs in clock cycle 0, is applied. While the configuration switches are set back, the master latches of the outputs are not active, because the previous pattern that has triggered the fault signal must not be applied again. Only after the falling edge in clock cycle 3, flip-flops at the output should be allowed to take over new data. For controlling the clock signals, we need a single 2-stage counter, which amounts to about 32 transistors. We have also considered to trigger the repair process by “fault” signals only. So far, we use a full comparator only for the outputs of 2 functional units for every function. The detection of a “fault compensated” information would actually mean triple parallel comparison and evaluation by a multi-input NOR gate, triggering “no fault” if the comparison has yielded one in 3 correct outputs.

5 Comprehensive Control for Re-Configuration

So far the re-allocation of resources was done assuming that no functional block in the RLB has permanent faults already. Under realistic conditions, it must be prevented that faulty blocks can be used for the installation of borrowed TMR, since the error correction works only under single fault assumptions within a specific time slot (figure 6).

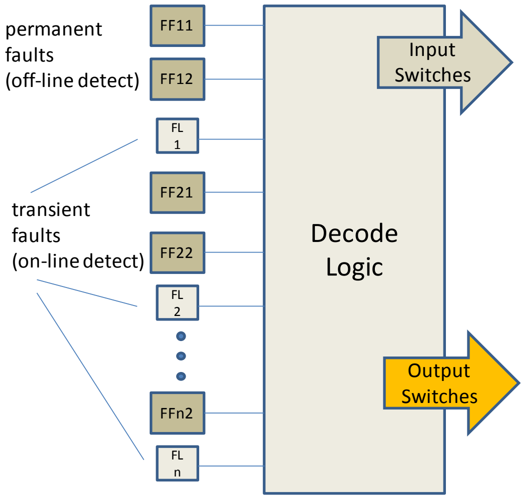

Figure 6. Inclusion of permanent and transient fault states

Hence there must be an additional control of switching inputs and outputs, depending on permanent faults, and units that are needed for permanent repair. The setting of fault bits FF1, FF2… FFn, indicating permanent faults in functional units, can be done during a start-up procedure of the system. If a permanent fault has been detected, one of two parallel functions will execute the associated function permanently. Then a TMR function is no longer possible for this function. Also the support of a neighboring function for "borrowed TMR" is no longer possible. By setting the fault-bits FF1, .. FFn arbitrarily, the system can also be forced into a low-power operating mode. Even then, however, the usage of parallel flip-flops and the MC-element at outputs makes sense, since this circuitry is able to detect and compensate short transient faults in the flip-flops themselves [Mi05]. If a delay line is inserted at the input of two D-latches working in parallel on a single output, also the compensation of short transient faults in the combinational logic blocks is possible [Mi06].

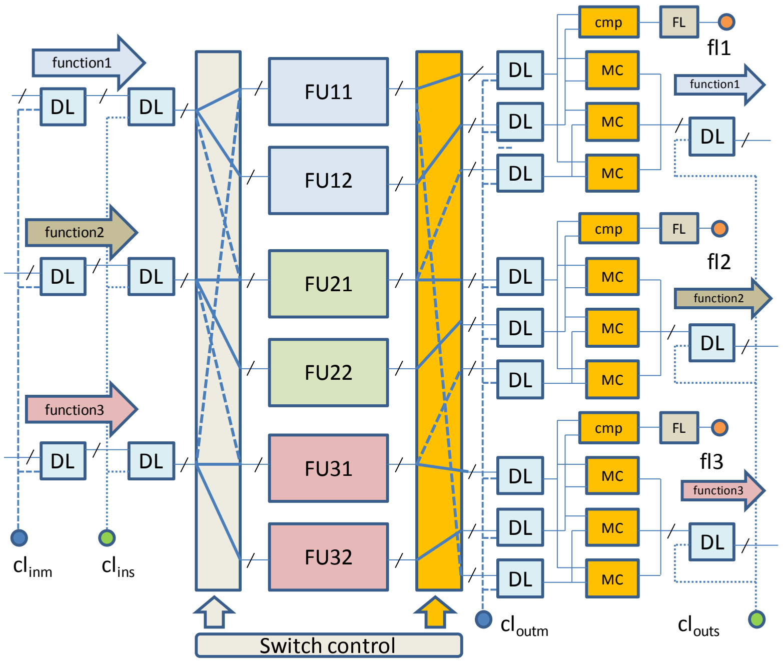

Figure 7. Overall architecture of the re-configurable block

In the overall structure of a re-configurable logic block (fig. 7) we assume that separate clock signals are needed for master- and slave flip-flops at inputs and outputs. At outputs, the slave latches can be composed into a single one, which also serves as the keeper-latch for the MC-elements. We have cl_{inm} for master latches at inputs, cl_{ins} for slave latches at inputs, and, respectively, cl_{outm} and cl_{outs} for the master and slave latches at the outputs. These clock signals can be derived from the normal clock cl and the fault flag bit rather directly.

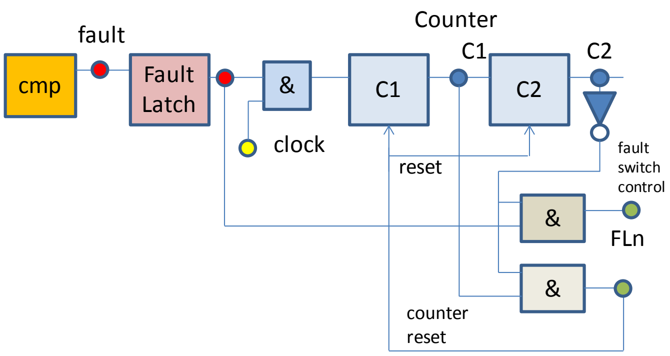

Figure 8. Control signals for fault indication and reset

The FLn signal that controls the setting of input / outputs switches is set to “high” after the arrival of a fault bit from the comparator. It is set back for the passing of the corrected output to the slaves in clock cycle 2. The counter gets a reset in clock cycle 3 (figure 8). At the outputs, the master latches serve to register and hold the outputs of the functional units directly. Results are passed to the output latches (one per bit) only after filtering by the MC-elements, which effectively create a voter, and at the right point of time. Fault effects in the blocks shown in figure 7 will not only influence the affected function, for example function 1 in case of a fault arising in FU12, but possibly also the other functions. Function 2 is in no danger of getting false inputs, while FU 21 is removed temporarily, and the re-attachment of FU21 after error correction is a problem only, if the time for re-setting input switches and for a signal transition through FU21 / FU22 exceeds the clock period. Function 3 is not affected at all. However, looking at global clocking schemes, it may be easier to apply the modified clock with suitable delays to all units. Then a single timing unit (counter and gates) is necessary for the total re-configurable block.

6 Overhead and Limitations

We have published overhead figures for similar architectures in previous work [Ko13,Ko14]. Unlike the architecture in figure 9, which uses separate master and slave latches at outputs, it is also possible to use full master-slave or edge-triggered flip-flops. In any case, the Muller C-elements need to feed a storage device, since their “high impedance” behavior in case of unequal inputs will create a floating node. Other authors have used an extra “keeper latch” at outputs, since then the full flip-flops with master and slave latches are allocated before the MC-elements and benefit from fault correction. Then, however, the keeper latch takes the role as a “single point of failure”. The overhead associated with the circuit according to figure 7 is shown in table 2. For comparison, table 3 indicates the overhead for the same type of architecture and basic blocks for full TMR using the same comparators.

Circuit	Trans. Count				Overhead in %					
	Trans. funct.	Ins per f. unit	Outs per f. unit	Trans. total	b-up	sw	FFs	Cmp/MC	ctrl	total
2-ALU	352	10	6	174	76	13	21	28	16	154
4-ALU	699	14	8	228	82	9	15	20	9	135
8-ALU	1367	24	12	372	86	8	12	17	5	128
16-ALU	2696	38	20	664	88	7	10	15	2	122
32-ALU	5343	70	36	1167	89	6	10	14	1	120

Table 2. Overhead for the RLB-architecture according to figure 9, 3 functions in parallel

Circ.	Transistors unit	Transistors total	Transistors Voter	Overhead in % back-up	voter	total
2-ALU	352	1056	72	200	20	220
4-ALU	699	2097	96	200	13	213
8-ALU	1367	4101	144	200	10	210
16-ALU	2696	8088	240	200	8.9	208.9
32-ALU	5343	16029	432	200	8.0	208.0

Table3. Overhead for full TMR implementation

Results show that the overhead is dominated by the cost for control and comparison only for relatively small functional units up to about 350 transistors, while in larger entities the cost of duplication is dominating. For comparison, table 4 also shows the cost of a scheme, where at all times all 3 functions are operated on 3 functional units one after the other in a scheme of permutation [Ko13]. This scheme has 3 functional units only and an even larger overhead for flip-flops. Since all calculations are done 3 times regardless fault events, there is a continuous reduction in throughput by a factor of 3, but there is also a constant delay. This scheme has 3 functional units only and an even larger overhead for flip-flops. Since all calculations are done 3 times regardless fault events, there is a continuous reduction in throughput by a factor of 3, but there is also a constant delay.

n = 3

BasicBl.	FunctTrans.	In/Outs	Transistor Count Red.Tr.	Sw.Tr.	in FFs	Clog	Decod.	Total	Ovh. %
2-NAND	4/12	2 /1	-	18	180	48	-	258	330
H-Adder	12/36	2 /2	-	18	360	48	-	462	250
F-Adder	30 /90	3 /2	-	27	360	48	-	525	180
2-ALU	352 /1056	10/6	-	90	1080	48	-	2274	69
4-ALU	699 / 2097	14/8	-	126	1740	48	-	4011	61
8-ALU	1367/4101	24 /12	-	216	2760	48	-	7125	57

Table 4. Overhead for time-shared TMR

Unlike more costly schemes that allow for a self repair process based on "cold" redundancy, the architecture shown in figure 11 can handle permanent faults also in multiple units, but after the re-configuration for such purpose, there is no extra resource for handling arbitrary transient faults for a function already infected by a permanent fault. Any probability calculation that we did so far showed that the limitations of reliability and life time are mainly set by elements, which are not protected by duplication. In our case, these elements are the control logic for re-configuration, the switches, and the final slave latches at outputs. Thereby the share of unprotected elements is rather low, compared with more complex repair schemes. Furthermore, there is no extra delay in case of good signals, and the power needed is only a duplication at average, whereby an additional "low power" mode with virtually no extra power, but possible short transient error correction, is also possible.

7 Summary and Conclusions

We proposed a pseudo-TMR-architecture, which is able to detect and correct transient faults and permanent faults equally well. Since apparently delay faults have found most attention recently [Bl04,Bl09,Bl13], we need to compare overhead and limitations. First, delay faults are assumed to occur only on relatively few logic paths, which are specifically identified and protected. Advanced methods [Bl13] will also detect and correct, mostly by re-execution, short transient fault effects in logic and in flip-flops. This comes at relatively high cost per output, since a sophisticated distinction between transient faults and delay faults is implemented. Then the total cost, if applied to all outputs for detection of transient faults and wear-out induced delays, is more than 100 % [Bl13]. The coverage of delay faults in our architecture needs an extra discussion. In case of delay faults, it will not detect delays that occur identically in two functional units such as FU11 and FU12. As, however, delay faults are apparently often corrected by re-execution even under the same timing [Bl13], it is unlikely that delay faults occur as identical double faults. The overhead encompassed here is in the area of about 120 %, but the merit is the universal applicability to different types of faults. Unlike the specific architectures for delay fault correction, this comes at the cost of double power and energy consumption. For the off-line repair of permanent defects by re-configuration, there is a need for some extra overhead for redundancy and switching [Ko14]. Inherently, the scheme holds some promise where several functional units of equal nature can be identified, such as in processor data paths. The application to totally irregular control logic is less economical and would involve a triplication in hardware, but only a duplication in power.

8 Acknowledgment

This work was partly supported by the German Science Foundation (DFG) within the DOZU project.

9 References

[Ba05] R. Baumann: „Soft Errors in Advanced Computer Systems“, IEEE Design and Test of Computers, Vol. 22. No. 3, May / June 2005, pp. 258-266

[Br04] M. A. Breuer, S. K. Gupta, T. M. Mak: “Defect and Error Tolerance in the Presence of Massive Numbers of Defects”, IEEE Design and Test of Computers, Vol.21, No. 3, May / June 2004, pp. 216-227

[Fa98] P. Fang, J. Tao, J. F. Chen, C. Hu: “Design in hot-carrier reliability for high performance logic applications”, Proc. IEEE Custom Integrated Circuits Conference, May 1998, pp. 525-531

[Li09] Y. Li, Y. M. Kim, E. Mintano, D. S. Gardner, S. Mitra, “Overcoming Early-Life Failure and Aging for Robust Systems”, IEEE Design and Test of Computers, Vol. 26, No. 6, Nov. / Dec. 2009, pp. 28-39

[Lie05] J. Lienig. G. Jerke: „Electromigration-Aware Physical Design of Integrated Circuits”, Proc. 18th Int. Conf on VLSI Design, pp. 77-82, 2005

[Al05] M. A. Alam, S. Mahapatra: “A comprehensive model of PMOS NBTI degradation”, Microelectronics Reliability, Vol. 45, No. 1, pp. 71-81, 2005

[Mi05] S. Mitra, N. Seifert, M. Zhang, Q. Shi, K. S. Kim, „Robust System Design with Built-In Soft Error Resilience“, IEEE Computer, Vol. 38, No.2, Febr. 2005, pp. 43-52

[Mi06] S. Mitra, M. Zhang, S. Waqas, N. Seifert, B. Gill, K. S. Kim, „Combinational Logic Soft Error Correction“, Proc. IEEE Int. Test Conf. 2006

[Ba09] M. Bashir, L. Milor, “Modeling Low-k Dielectric Breakdown to Determine Lifetime Requirements”, IEEE Design and Test Vol. 26, No. 6, Nov. / Dec. 2009, pp. 18-27

[Lu04] Z. Lu et al., “Interconnect Lifetime Prediction under Dynamic Stress for Reliability-Aware Design”, Proc. Int. Conf. On Computer Aided Design (ICCAD04), IEEE CS Press, pp. 327-334, 2004

[Pr96] D. K. Pradhan, "Fault Tolerant Computing", Prentice Hall, 1996

[La01] P. G. Lala, „Self-Checking and Fault Tolerant Digital Design“, Morgan Kaufmann Publishers, San Francisco,2001

[She09] X. She, K. S. McElvain, “Time Multiplexed Triple Modular Redundancy for Single Event Upset Mitigation”, IEEE Trans. Nuclear Science Vol.56, No. 4, August 2009, pp. 2443-2448

[Ga00] W. L. Gallagher, E. E. Swartlander, “Fault-Tolerant Newton-Raphson and Goldschmidt Dividers Using Time Shared TMR”, IEEE Trans. Computers, Vol. 49, No. 6, June 2000

[Goe11] M. Augustin, M. Gössel, R. Kraemer, „Implementation of Selective Fault Tolerance with Conventional Synthesis Tools“, Proc. IEEE DDECS 2011, Cottbus, pp. 213-218

[Ko12] T. Koal, M. Ulbricht, H. T. Vierhaus “Combining On-Line Fault Detection and Logic Self Repair”, Proc. 15th IEEE Int. Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS 2012), Tallinn, April 2012, Editor J. Raik

[Ko11] T. Koal, D. Scheit, M. Schölzel, H. T. Vierhaus, „On the Feasibility of Built-in Self Repair for Logic Circuits”, Proc. DFT 2011, Vancouver

[Ko13] T. Koal, M. Ulbricht, H. T. Vierhaus. „Virtual TMR Schemes Combining Fault Tolerance and Self Repair”, Proc. Euromicro Conference on Digital Systems Design (DSD 2013), Santander, Sept. 2013 ISBN: 978-3-902457-38-7

[Ko14] Tobias Koal, Mario Schölzel, Heinrich T. Vierhaus: „Combining Fault Tolerance and Self Repair at Minimum Cost in Power and Hardware”, Proc. IEEE DDECS 2014, Warsaw

[Mu14] Sebastian Müller, Tobias Koal, Mario Schölzel, Heinrich T. Vierhaus, „Timing for Virtual TMR in Logic Circuits“, Proc. IEEE On-Line Test Symp. 2014

[Bl04] D. Ernst, D. Blaauw, T. Austin et al., „Razor; Circut-Level Correction of Timing Errors for Low-Power Operation“, IEEE Micro, Nov. / Dec. 2004, pp. 10-20

[Bl09] S. Das, D. Blaauw et al., „Razor II: In Situ Error Detection and Correction for PVT and SER Tolerance, IEEE Journal of Solid State Circuits, Vol. 44., No. 1, January 2009, pp. 32-48

[Bl13] M. Fojtik, D. Fick, Y. Kim, N. Pcikney, D. M. Harris, D. Blaauw, D. Sylvester, “Bubble Razor: Eliminating Timing Margins in an ARM Cortex-M3 Processor in 45 nm CMOS Using Architecturally Independent Error Detection and Correction”, IEEE Journal of Solid State Circuits, Vol. 48, No.1. January 2013