State Recovery of Coarse-Grained TMR Circuits based on Scan Chains and Clock Gating

Jakob Lechner

RUAG Space GmbH, Vienna, Austria
jakob.lechner@ruag.com

Abstract. Triple modular redundancy (TMR) is a wide-spread technique for mitigating soft-errors in digital circuits. Replication of synchronous circuits is often performed at gate-level, where a single clock tree needs to be maintained and a specialised redundant implementation of the circuit is necessary. In case of replication at module-level on the other hand, any non-redundant module implementation can be easily triplicated without modification. The replicas can then be placed at physically isolated locations on the die for optimal fault tolerance. The drawback of this approach is that voting can only be performed at the module outputs and extra effort is required to recover the internal state of a compromised replica. In this paper we therefore present a scan chain-based recovery mechanism, which allows for scrubbing the internal state. The replicated modules do not need to be within a single clock domain, albeit they are likely to operate with the same clock frequency. Clock gating is used to compensate for run-time differences, whenever a state recovery needs to be performed.

1 Introduction

Fault-tolerant circuit architectures are essential for applications that have high reliability requirements and/or are exposed to harsh environmental conditions such as radiation in space. In space electronics, e.g. a shift towards feature sizes of 65 nm and below can be observed due to increasing processing demands. With decreasing feature sizes, however, soft error rates due to radiation are increasing. Furthermore, there is currently a large momentum for small-satellite platforms that rely on inexpensive commercial-of-the-shelf (COTS) parts, which obviously cannot offer the same radiation-hardness as space-grade semiconductor ICs.

All these trends demand for fault-tolerant solutions that can be implemented at architectural level. Triple-modular redundancy is a widely used technique to improve circuit reliability. Replication can be done at gate level, where individual gates and flip-flops are triplicated (fine-grained TMR), or by replicating entire modules, where each module instance is effectively an unchanged copy of the non-redundant circuit implementation (coarse-grained TMR). Both solutions have their upsides and downsides and depending on the specific application needs one might match better than the other. Fine-grained TMR requires EDA tools to perform the desired replication and to insert voters that mask errors after flip-flops. Due to these voters SEUs in flip-flops cannot spread within the circuit and are recovered in the next clock cycle that overwrites the affected flip-flop. This obviously requires that the entire replicated circuit belongs to a single clock domain with one common clock tree, which remains a single point of failure. The use of voters inside a fine-grained TMR circuit increases the complexity of the interconnect and thereby has direct implications on the physical implementation of the modular redundant circuit. Limited routing capacities and strict timing constraints enforce a compact circuit layout, where replicated components have to be placed in close proximity. This might reduce the reliability of the resulting system, especially if multiple-bit upsets have to be considered.

Coarse-grained or module-level replication on the other hand can be done manually by the RTL designer, simply by instantiating the reliability-critical modules threefold and by implementing voting functions for the module outputs. As the replicated modules are not internally connected via voters, physical separation is possible, minimising the probability of *spatial proximity faults* [1]. Furthermore, the circuit layout of the replicas can remain identical to a non-redundant implementation avoiding any timing penalties that might occur due to the increased routing complexity of a fine-grained solution. The disadvantage of a coarse-grained setup, however, is the fact that SEU in flip-flops are not implicitly scrubbed, since voting is only performed outside the modules. A single upset thus might quickly poison the state of other flip-flops in the affected module replica, whose outputs might then become fully inconsistent to the remaining two healthy replicas. In this situation the failure probability is twice as high compared to a non-redundant system, since a soft error in one of the two healthy modules will cause a system failure. Therefore, a recovery mechanism is required that quickly brings a faulty module replica back to a correct and consistent state. In this paper we want to discuss the concept for such a recovery mechanism.

2 Related Work

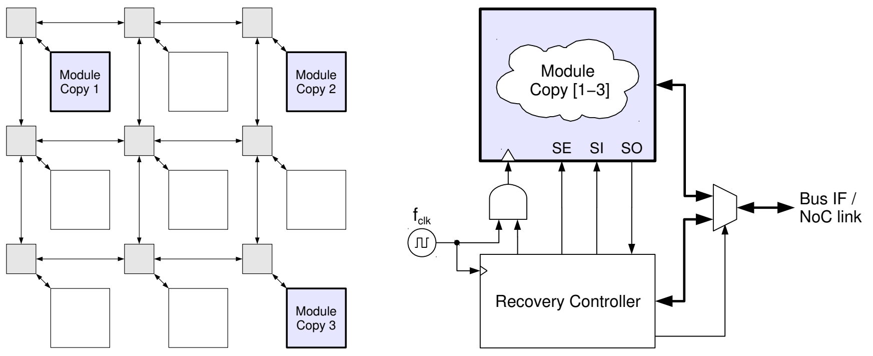
In [2] Yu et.al. presented a roll-forward mechanism to improve the reliability of TMR circuits. They propose a state restoration scheme where faulty registers are overwritten with the data values provided by correct registers. Like in our solution recovery is performed at predetermined checkpoints.

Ebrahimi et.al. [3] employ a voting mechanism to detect faults at the outputs of triplicated components. If an error is detected, the recovery process is triggered. Like in our solution register scan chains are used to recover a faulty state. In [4] the same authors propose an extension of their previous work to recover from multiple transient faults in one or two replicas.

In [5, 6] we have already introduced a scan-chain based recovery mechanism for TMR modules in GALS systems. In such a system the entire module communication is performed over asynchronous interfaces. Synchronization is performed with stoppable clock generators built from on-chip ring oscillators. These clock generators are also used to stall the module operation during the recovery process, i.e. they fulfil the same purpose as the clock gating mechanism in this paper. However, the design of robust ring oscillators with low jitter also constitutes a major challenge for practical circuit implementations.

3 Concept

Integrated circuits are typically composed of several modules or IP cores, which are performing various tasks needed for the operation of the chip. They usually communicate over standardised bus systems or a network on chip (NoC). When replicating a critical module there are two alternatives how the module copies can be connected to the bus/NoC: 1) Via a single bus interface or NoC link, where a voter protects the system against faulty accesses to the interconnection network, or 2) via individual, independent interfaces, where each copy can directly provide output data for other components of the system. In first approach failures of a single module copy will remain completely transparent to the rest of the system, whereas in the second solution components that process output data of the triplicated modules need to acquire all the redundant outputs and perform voting themselves. The benefit of the latter approach, however, is that maximum independence of the replicated modules can be achieved. The chip designer can place them anywhere on the die, e.g. in different corners of an NoC mesh. Figure 1 illustrates this idea for an NoC-based chip architecture. Note that the redundant modules do not necessarily need to be in a single clock domain, since they do not exchange data directly. In a mesochronous NoC [7], e.g. each module would likely be operated with the same clock frequency but their clock signals could have an arbitrary (constant) phase offset.



(a) NoC-based chip architecture. (b) Module with recovery controller.

Fig. 1. Coarse-grained TMR concept.

To support the required recovery process we need a mechanism to access state information and exchange it among the replicas. In [5] we therefore proposed to use a scan-chain based approach to tackle this problem. Scan chains provide a simple mechanism to access the state of a module's flip-flops, both for reading and writing their stored values. Considering that scan chains are available anyway in most ASIC designs for testability reasons, they provide an area-neutral means to perform state recovery. All we need are recovery controllers, one for each replicated module, which perform a readout of their module's scan chain, exchange data with the other replicas and perform majority voting on these data bit by bit and shift the voting results back into the scan chain. Obviously, this process can only be executed when the redundant modules have the exact same state, at least for the flip-flops that store data relevant for the execution after the recovery. Also the recovery controller needs to be in a single clock domain with the module it is associated to.

The data exchange can be done by re-using the module's local link to the system bus or NoC. Depending on the link width a certain number of bits can be read from the scan chain and then transmitted. Concurrently, the recovery controller waits for the same data chunk to arrive from the other replicas. This could take a few clock cycles since the replicated modules are likely to experience a different number of wait cycles when accessing a shared bus or a network-on-chip with other communicating nodes (note that the replicated modules could already be off by some cycles due to different timing of I/O operations during the regular execution). While a recovery controller waits for incoming state recovery data, clock gating can be used to stall the scan-chain operation.

Attention has to be paid to the fact that the recovery controllers themselves might be affected by SEUs. A recovery controller could deadlock or become inconsistent to the other controllers and therefore not join the recovery process at the time it is supposed to. Since the other controllers wait for data (while gating the clock), the healthy module replicas would get stuck. To avoid this a watchdog timer can be implemented in every recovery controller, which detects such a deadlock situation. If a timeout happens in the middle of the recovery process, the healthy controllers can abort the recovery and finish the scan and shift back operation locally without voting. This brings the healthy replicas back into a state where they can go on with their regular operation.

To recover a deadlocked or inconsistent recovery controller – a necessary action to bring the TMR ensemble back to a fully functional state – a mechanism needs to be implemented for the healthy controllers to force their erroneous counterpart into the recovery operation. This can be done with a control signal that is asserted by every (functional) recovery controller at the beginning of the recovery process and is routed to the other redundant controllers. Consider this signal to be a *recovery request*, which is mutually exchanged among the controllers. By voting over local and remote request signals, every controller can determine, if the majority of controllers wants to start the recovery. This information activates another timeout-controlled circuit, which drives the recovery controller's state machine into recovery mode when it fails to do so on its own before the timeout occurs. The replicated recovery controllers therefore form a TMR system themselves. Note that the recovery request signals are mesochronous inputs and have to be synchronized to the clock domain of the receiving controller. Furthermore, timeout values need to be carefully selected in order for this mechanism to work. For details the interested reader is referred to [6].

4 Conclusion

In this paper we presented the concept of a coarse-grained TMR implementation. Replicated modules provide their outputs to other modules over a bus system or NoC. The replicas do not need to be synchronized to each other and therefore can be implemented independently with no physical constraints on their location on the die. The state recovery with scan-chains does not add extra circuits to the modules and the recovery controllers are simple state machines, which would have a very small area footprint.

Acknowledgements. The ideas for this work stem from the concepts and methods developed for my PhD thesis. A cornerstone for my PhD thesis and the better part of my scientific work so far was a lecture that Alex Yakovlev gave in 2011 at Vienna University of Technology. The lecture obviously was on asynchronous circuits and systems design and helped me a lot to better understand the asynchronous design philosophy. But most of all Alex was able to electrify me with his immense enthusiasm for the field. Thank you so much for your friendship and support, Alex!

References

1. Kopetz, H.: Real-Time Systems: Design Principles for Distributed Embedded Applications. 2st edn. Springer New York Dordrecht Heidelberg London (2011)
2. Yu, S.Y., McCluskey, E.: On-line testing and recovery in tmr systems for real-time applications. In: Test Conference, 2001. Proceedings. International. (2001) 240 –249
3. Ebrahimi, M., Miremadi, S., Asadi, H.: Sctmr: A scan chain-based error recovery technique for tmr systems in safety-critical applications. In: Design, Automation Test in Europe Conference Exhibition (DATE), 2011. (march 2011) 1 –4
4. Ebrahimi, M., Miremadi, S.G., Asadi, H., Fazeli, M.: Low-cost scan-chain-based technique to recover multiple errors in tmr systems. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on **PP**(99) (2012) 1
5. Lechner, J., Veeravalli, V.S.: Modular redundancy in a gals system using asynchronous recovery links. In: Asynchronous Circuits and Systems (ASYNC), 2013 IEEE 19th International Symposium on. (2013) 23–30
6. Lechner, J.: Building Robust GALS Circuits – Fault-Tolerant and Variation-Aware Design Techniques for Reliable Circuit Operation. PhD thesis, Vienna University of Technology (April 2014)
7. Ludovici, D., Strano, A., Gaydadjiev, G.N., Bertozzi, D.: Mesochronous noc technology for power-efficient gals mpsocs. In: Proceedings of the Fifth International Workshop on Interconnection Network Architecture: On-Chip, Multi-Chip. INA-OCMC '11, New York, NY, USA, ACM (2011) 27–30