View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Dagstuhl Research Online Publication Server

Error Containment in the Presence of Metastability

Andreas Steininger

Institute of Computer Engineering / ECS group
Vienna University of Technology
A-1040 Vienna, Treitlstrasse 3, Austria
andreas.steininger@tuwien.ac.at

Abstract. Error containment is an important concept in fault-tolerant system design, and techniques like voting are applied to mask erroneous outputs, thus preventing their propagation. In this presentation we will use the example of DARTS, a fault-tolerant distributed clock generation scheme in hardware, to demonstrate that metastability is a substantial threat to error containment. We will illustrate how metastability can originate and propagate such that a single fault may upset the system. The main conclusion is that modeling efforts on all design levels are definitely required in order to mitigate and quantify the deteriorating effect of metastability on system dependability.

Keywords. metastability, fault tolerance, clock generation

1 Introduction

In our research project DARTS (Distributed Algorithms for Robust Tick Synchronization) we implemented a clock synchronization algorithm from the distributed systems community (in particular a slightly modified version of the consistent broadcast primitive by Srikanth-Toueg [1]) in hardware. An instance of this algorithm is implemented in a hardware module called *TG-Alg* that can be attached to a conventional synchronous function block, for which it serves as the clock source. In this way a set of n distributed identical TG-Algs (typically 5 to 14) run in the system. They communicate over a network of serial point-to-point connections called *TG-Net*. In some sense this network of coupled TG-Algs operates like a large (and very sophisticated) ring oscillator and generates a system wide clock without the need for an external clock source. For details on DARTS see [2,3]. Among the benefits of this DARTS approach are:

- The clock remains synchronous over all communicating (non-faulty) TG-Algs, provably within a bounded precision (typically 2 or 3 clock cycles)
- Up to f arbitrary (Byzantine) failures in the TG-Algs and/or TG-Net can be tolerated in a system comprising $n \geq 3f + 2$ TG-Algs; i.e. all non faulty TG-Algs still generate a clock within the specified precision.

Dagstuhl Seminar Proceedings 08371
Fault-Tolerant Distributed Algorithms on VLSI Chips
http://drops.dagstuhl.de/opus/volltexte/2009/1923

These properties are very interesting, since they allow eliminating the single point of failure usually formed by the synchronous clock source, while—in contrast to the GALS approach [4]—still retaining the important notion of synchrony. The question we want to answer in the following is how our system that is proven to withstand unrestricted, namely Byzantine, failures on the algorithmic level, behaves under metastability conditions.

2 Metastability

Metastability is an undesired property of bistable elements whose input space is continuous-valued [5]. In terms of hardware the usual showcase is a latch cell whose function is to properly output a HI or LO, while its input voltage and/or time between certain edges are ultimately continuous. The actual problem is that for a borderline case at the input the bistable element may need an unbounded time to decide which of the discrete output states to assume. This is a fundamental problem that has been proven to be unsolvable within bounded time [6]. In a properly designed digital system, however, the input voltage is either clearly HI or clearly LO (with steep transitions in between), thus avoiding the borderline case with respect to voltage. Similarly the design style (either synchronous or handshake based/asynchronous) rules out the occurrence of borderline cases with respect to the time between transitions (In the synchronous case this simply means observing the setup- and hold time of a flip-flop, e.g.). Therefore metastability is usually encountered only in the very restricted context of synchronizers and arbiters, where it is well researched [7,8,9]. Although, as already mentioned, metastability cannot be avoided in principle, there are means to make its occurrence arbitrarily improbable, and models exist to (statistically) estimate the mean time between upsets (MTBU) due to metastability [10,11].

There are three different ways in which metastability can manifest:

1. Excessive delay of a transition. This will normally cause timing violations in the subsequent (synchronous) circuit.
2. The output assumes an undefined value in between HI and LO for an unbounded time. This can lead to a propagation of the undefined logic level or to ambiguous interpretation by different subsequent circuit elements and hence to malfunction.
3. Oscillation of the output. The problem here is that the edges generated by this self-oscillation are not in causal relation with the input.

Case (1) either leads to a delayed recognition of the input change, or, in the worst case, to a propagation of the metastability to the next bistable element. In context with usual technologies case (2) has most often been encountered and researched especially for synchronous systems. Case (3) has quite rarely been reported from experiments and observations in practical applications, and has received very limited attention in the literature. In a synchronous design chances are that the oscillation goes unnoticed due to temporal masking, while in case of transition based signal encoding (like in QDI circuits [12]) the acausal edges are likely to upset the circuit. While the *effect* of metastability is hence strongly dependent on the design style, existing research indicates that the *type of manifestation* depends on technology parameters rather than the conditions of stimulation, which suggests an independence of the design style for the latter.

3 Metastable Upsets in DARTS

3.1 DARTS Block diagram

As shown in Fig. 1 a TG-Alg comprises a set of *Counter Modules* (one for each "remote" clock coming from an other TG-Alg) and a *Threshold Modules* unit.

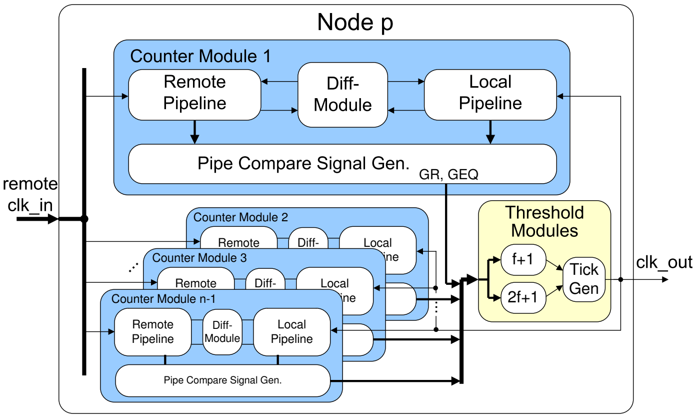

Fig. 1. The DARTS implementation.

Both these modules are implemented in asynchronous logic, since the purpose of the TG-Algs is to generate a clock in the first place. A Counter Module is essentially an up/down-counter that is internally built from two elastic pipelines [13] that serve as buffers for transitions. One is fed from the remote clock input (*Remote Pipeline*) and the other one from the local clock (*Local Pipeline*). A so-called *Difference Module*—essentially a Muller C-Element [13]—removes matching transition pairs from these pipelines, leaving the difference of transitions in one of the pipelines. This information is extracted by a combinational logic block, the *Pipe Compare Signal Generator* and provided to the Threshold Modules. This unit comprises four threshold gates, each of which is a purely combinational function block that outputs a `HI` when it receives a `HI` on at least k of its $n-1$ inputs. Actually the four threshold gates operate partially in parallel (different thresholds k) and partially in alternation (separately for rising and falling edges), for details see [2]. Finally, the Tick Generation module translates the status outputs received from the threshold gates into a sequence of transitions that forms the clock output of the respective TG-Alg. In short, the purpose of the Threshold Modules unit is to fire a new transition on the clock output as soon as a sufficient number of nodes has requested to do so. According to the algorithm's principle this threshold mechanism masks erroneous or missing requests and thus ensures error containment.

3.2 Metastability generation and propagation

A closer analysis of these modules with respect to metastability yields the following status:

- The purely combinational blocks, namely PCSG and threshold gate, are no bistable elements (no internal state, or no positive feedback, resp.) and hence not prone to metastability. They are, however, capable of propagating a metastable state. Consider the threshold gate as an example: With $k-1$ `HI` inputs the element is just below its threshold, such that a metastable state on one of the remaining inputs may be propagated to the output. Notice, however, that in all other cases a metastable input will not be propagated.
- Elastic Pipeline, Difference Module and Tick Generation Module contain state-holding elements, in particular Muller C-Elements. We have formally derived timing constraints under which these blocks are guaranteed to find proper operating conditions [14], and we have considered all of these constraints in our implementation. So there is no threat of metastability in the fault free case. In case of faults, however, these Muller C-Elements do have the potential of going metastable. Consider the case of a glitch or runt on one of the connections within the TG-Net being caused by electromagnetic interference or by an SEU hitting the sender's output. In general it cannot be ruled out that such an event will create that very borderline condition that makes the first Muller C-Element in the Elastic Pipeline go metastable. Its metastable output may cause a borderline condition for the subsequent Muller C-Element, and so on, such that the metastability propagates.

So in summary in our DARTS circuit we do find a potential for the *generation* of metastability—namely in case of a fault—and for its *propagation*. In particular it can be shown quite easily that a single metastable event can spread all over the system, causing all TG-Algs to fail. Although this appears to be an extremely unlikely scenario, it stands in sharp contrast to the formal proofs on the algorithmic level claiming the system can withstand (even more than one) arbitrary component failures. Obviously the problem here is that metastability can overcome usual error containment boundaries (like the Threshold Modules in our example) in a way that is not captured in the formal treatment.

So far we have implicitly assumed a metastability manifestation as an undefined logic level (case (2) in the above list). It goes without saying that no error containment boundaries will hold for the oscillatory case (3) either.

4 Conclusion

We have given evidence for the threat represented by metastability in our DARTS system, and we are convinced that the case we made here is not limited to

DARTS alone but applies to many (if not all) fault-tolerant architectures. Given the implicit electrical, logical and temporal masking effects as well as the explicit measures for metastability mitigation that are sometimes introduced, the residual risk may seem negligible. But even if this may be so, it cannot be reliably judged without a sound *quantitative* assessment. Unfortunately the state of the art with respect to metastability modeling is—while being quite elaborate for the synchronous case—not sufficiently developed for MTBU predictions in asynchronous systems, especially for fault-induced glitches and runts. So a sound quantitative MTBU prediction in this context remains an open research task. Further research is needed with respect to the inclusion of metastability in higher-level models, which might enable its mitigation on the algorithmic level.

References

1. Srikanth, T.K., Toueg, S.: Optimal clock synchronization. Journal of the ACM **34** (1987) 626–645
2. Ferringer, M., Fuchs, G., Steininger, A., Kempf, G.: VLSI Implementation of a Fault-Tolerant Distributed Clock Generation. IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT2006) (2006) 563–571
3. Fuegger, M., Schmid, U., Fuchs, G., Kempf, G.: Fault-Tolerant Distributed Clock Generation in VLSI Systems-on-Chip. In: Proceedings of the Sixth European Dependable Computing Conference (EDCC-6), IEEE CS Press (2006) 87–96
4. Chapiro, D.M.: Globally-Asynchronous Locally-Synchronous Systems. PhD thesis, Stanford University (1984)
5. Lamport, L.: Using time instead of timeout for fault-tolerant distributed systems. ACM Transactions on Programming Languages and Systems **6** (1984) 254–280
6. Marino, L.: General theory of metastable operation. IEEE Transactions on Computers **C-30** (1981) 107–115
7. Dike, C., Burton, E.: Miller and noise effects in a synchronizing flip-flop. IEEE Journal of Solid-State Circuits **SC-34** (1999) 849–855
8. Yang, S., Greenstreet, M.: Computing synchronizer failure probabilities. In: Proc. Intl. Conference on Design Automation and Test in Europe, IEEE CS Press (2007)
9. Kinniment, D.J., Dike, C.E., Heron, K., Russell, G., Yakovlev, A.V.: Measuring deep metastability and its effect on synchronizer performance. IEEE Transactions on VLSI Systems Circuits **15** (2007) 1028–1039
10. Kleeman, L., Cantoni, A.: Metastable behavior in digital systems. IEEE Design & Test of Computers **4** (1987) 4–19
11. Semiat, Y., Ginosar, R.: Timing measurements of synchronization circuits. In: Proc. IEEE Int. Symp. on Asynchronous Circuits and Systems, IEEE Computer Society Press (2003) 1–10
12. Martin, A.J.: Limitations to delay-insensitivity in asynchronous circuits. Technical report, California Institute of Technology, Pasadena, CA, USA (1990)
13. Sutherland, I.E.: Micropipelines. Communications of the ACM, Turing Award **32** (1989) 720–738 ISSN:0001-0782.
14. Fuchs, G., Fuegger, M., Steininger, A., Zangerl, F.: Analysis of constraints in a fault-tolerant distributed clock generation scheme. 3rd International Workshop on Dependable Embedded Systems (WDES'06) (2006)