

On the Stability and Robustness of Non-Synchronous Circuits with Timing Loops

Matthias Függer, Gottfried Fuchs, Ulrich Schmid and Andreas Steininger
Vienna University of Technology — Embedded Computing Systems Group
{fuegger, fuchs, s, steininger}@ecs.tuwien.ac.at

Abstract—Among the most pressing issues in current deep submicron VLSI circuits are large parameter variations, primarily caused by process technology imperfections, that result in excessive delay variations, and continuously increasing soft error rates. The former seriously challenges the classic synchronous design paradigm, and coping with the latter requires suitable fault-tolerant architectures.

In this paper, we present first results of the parameter sensitivity analysis of our DARTS fault-tolerant clock generation scheme. Unlike conventional clocking techniques for GALS systems, DARTS does not use quartz oscillators, but generates approximately synchronized clocks by means of an asynchronous distributed algorithm. Initially, both measurement and simulation results indicated deterministic fluctuations of the DARTS clock frequency, which depend strongly on the delay parameters, but could not be explained by noise or dynamic parameter variations. By means of non-linear control theory we develop an explanation of these effects, and establish that any non-trivial closed-loop asynchronous circuit exhibits such deterministic timing variations. Our simulation results confirm that majority voting, as employed to attain fault-tolerance in DARTS, improves the robustness against parameter variations, but also considerably increases the complexity of the underlying control theory problem.

I. INTRODUCTION

During the last decades, the reduction of the feature size down to the sub-100 nm range has enabled tremendous advances in semiconductor technology: Chips became faster, denser and cheaper at a remarkable pace. There have always been substantial technological challenges associated with this miniaturization, such as excessive subthreshold currents, thermal problems, and oxide leakage. Although all these problems were solved by technological means in the past, the associated costs have dramatically increased. Further miniaturization, targeting feature sizes well below 90 nm, will hence require alternative ways for coping with the resulting effects, which are primarily substantial delay (and other parameter) variations and increased susceptibility to transient faults [1]. This raises a number of interesting research issues.

More specifically, with parameter variations that affect the attainable operation frequency by as much as 30% [2] (note that the variations due to noise, power supply fluctuations and temperature add to this figure), the traditional synchronous design approach with static corner case analysis becomes questionable: Since, as stated in [3] “safety margins are becoming an important path delay component”, the margins required for accommodating all the delay variations may exceed the performance gain obtained by a further reduction of the feature

size [4]. Expensive measures like postsilicon tuning [5] are hence being applied to counter the deteriorating effect of parameter variations.

One possible solution to this problem is statistical timing analysis, which exploits the fact that the most pessimistic corner cases of the worst case analysis are usually extremely unlikely to be ever encountered. Using the synchronous paradigm in conjunction with the resulting clock frequency may work satisfactory for typical applications. For dependable systems, however, such a solution is typically considered unacceptable since synchronous circuits do not provide graceful degradation in case of a timing violation but rather fail arbitrarily. It is hence more natural to use more relaxed design paradigms like asynchronous quasi-delay insensitive logic [6], GALS (globally asynchronous locally synchronous [7]), or any other approach that is sufficiently adaptive to delay variations.

The other major weakness of present nanoelectronics is its susceptibility to radiation-induced transient faults [8]–[10]. Studies like [11] reveal that the critical charge for 100 nm technologies is below 5 fC, while particle strikes can generate charges in excess of 100 fC. It is argued that, without mitigation techniques, the soft error rate (SER) of e.g. DRAMs could easily exceed 50000 failures in 10^9 device hours [12]. Note that radiation-induced errors have even been reported for the clock distribution network in classic synchronous chips [13]. Although technology improvements like silicon-on-insulator intrinsically improve the SER [14], circuit- and/or on system level fault-tolerance techniques [15], [16] are deemed necessary for critical applications. At the system level, such techniques typically involve replication of components plus some form of majority voting — which in turn is difficult without a synchronous clock. For these reasons, a promising research challenge is to look out for replicated architectures that work also with relaxed synchrony assumptions. One example is our DARTS (Distributed Algorithms for Robust Tick-Synchronization)¹ fault-tolerant tick generation scheme developed for GALS systems.

Rather than using quartz oscillators, DARTS generates approximately synchronized clocks by means of a simple distributed algorithm that has been implemented using asynchronous digital logic. Since the DARTS clock frequency is entirely determined by certain path delays, it automatically

¹The project DARTS received funding from the Austrian bm:vit (FIT-IT, contract no. 809456-SCK/SAI). DARTS project Webpage: <http://ti.tuwien.ac.at/ecs/research/projects/darts/>

adapts to delay variations caused by parameter variations, including dynamic ones like temperature, supply voltage, etc. Note that such an adaptive behavior is characteristic for the advocated flexible timing approaches, since variations are not just hidden in some safety margins. Given this strong dependence on the parameters, DARTS clocks are an excellent application for studying the effects of parameter variations.

Beyond our interest in characterizing the stability of our DARTS clocks, we also wanted to study whether and how the fault tolerance mechanisms implemented in DARTS help in mitigating the effects of parameter variations on the clock frequency. Our conjecture was that voting would simply mask large variations of a certain number of parameters. Note that we are not aware of any work in the literature that deals with this problem.

The detailed contributions of our paper are the following: After a brief introduction of the system model in Section II, we will show by means of simulation in Section III that the frequency of our DARTS clocks exhibits some fluctuations that depend strongly on the delay parameters. Surprisingly, however, it turned out that they are not caused by noise or dynamic parameter variations. Rather, they are deterministic fluctuations that occur also if all delay parameters are totally stable (but different from each other). Non-linear control theory, applied to the non-fault-tolerant version of DARTS in Section IV allowed us to develop an explanation of these effects. It turned out that any non-trivial closed-loop asynchronous circuit exhibits such deterministic timing variations. Additional simulation results confirmed that majority voting, as employed in the fault-tolerant version of DARTS, indeed improves the robustness against parameter variations, but also considerably increases the complexity of the underlying control theory problem. The section is concluded by a study of the robustness of our system with respect to parameter variations, including a comparison to the synchronous case. As we consider our work as an initial step into a huge new class of problems, a list of open issues is finally presented in Section V.

II. TICK GENERATION SYSTEMS

We consider the following class of distributed systems: The system comprises n nodes $P = \{A, B, C, \dots\}$ together with a message passing communication topology, here modelled as a weighted digraph $\langle P, E \subseteq P^2, \omega \rangle$, where $\omega(e)$, for $e = \langle X, Y \rangle \in E$ is a non negative weight assigned to the edge from X to Y modeling the communication delay along this path in [s]. Further we assign a (message triggered) algorithm to each of the nodes. A node p 's algorithm performs as follows: It consists of a set of rules which are triggered whenever a message arrives at node p and may possibly update node p 's local memory as well as send messages to other nodes (a so called receive-compute-send step). Here, for simplicity we assume that receive-compute-send steps are atomic and of duration 0. We finally restrict the class of distributed systems to those which send only messages of ascending natural numbers at each node, so called tick generation systems, i.e.,

we demand that every node p sends messages $\langle 0 \rangle, \langle 1 \rangle, \langle 2 \rangle, \dots$ in the given order during its executions.

We further follow Leslie Lamport [17], [18] and define a synchronization problem to be a set of constraints on the order of tick messages sent by nodes during an execution. An algorithm solves a synchronization problem, iff all of its executions fulfill the synchronization problem's constraints. Typical constraints for synchronization problems are precision, accuracy and restricting a producer node to act before a consumer node acts on the same piece of data. Algorithms ensuring the latter are typically implemented by handshaking, i.e., a node waits until it has received the k^{th} piece of data (message $\langle k \rangle$) over all incoming links (following a "wait for all"-rule), processes the data, and sends out the $k + 1^{\text{st}}$ piece of data (message $\langle k + 1 \rangle$) over its outgoing links.

When considering fault-tolerant synchronizing systems, that is distributed tick generation systems which solve a synchronization problem in spite of faulty nodes, one has to replace the "wait for all"-rule with more intricate rules. One such system is the DARTS fault-tolerant tick generation approach.

A. DARTS

The key idea in DARTS is to use a distributed tick generation algorithm (TG-Alg) solving a synchronization problem in hardware. Several instances of TG-Algs are distributed over the System-on-Chip (SoC) and communicate over a fully connected (single-rail) point-to-point network (TG-Net). The TG-Net is responsible for exchanging the $\langle k \rangle$ messages and thus comparable to the clock distribution tree of a synchronous clocking paradigm or the handshake wires between asynchronous modules. Each such TG-Alg supplies a locally attached synchronous SoC module with a synchronized, fault tolerant clock signal $\langle 0 \rangle, \langle 1 \rangle, \dots$. The algorithm we implemented in the TG-Algs is a fault-tolerant simulated authentication protocol introduced by Srikanth and Toueg in [19] which was modified by Widder and Schmid in [20] to a tick generation algorithm depicted in Fig. 1.

```

1: /* Initialization */
2: VAR  $k$  : integer := 0;
3: initially send  $\langle 0 \rangle$  to all [once];
4: if received  $\langle l \rangle$  from at least  $f + 1$  distinct processes with  $l > k$ 
   then
5:   send  $\langle k + 1 \rangle, \dots, \langle l \rangle$  to all [once];  $k := l$ ;
6: if received  $\langle k \rangle$  from at least  $2f + 1$  distinct processes then
7:   send  $\langle k + 1 \rangle$  to all [once];  $k := k + 1$ ;

```

Fig. 1. Tick generation algorithm

Clearly the Algorithm listed in Fig. 1 is not suited for direct implementation in hardware due to several reasons, the most significant being the extremely high degree of atomicity: at the reception of a message, evaluating line 4, broadcasting new tick messages and updating the local tick counter k must be performed before any other message arrives. Thus the algorithm has been further refined yielding a design depicted in Fig. 2. The Figure shows node A of an $n = 5$ node system, able to tolerate $f = 1$ arbitrary faults.

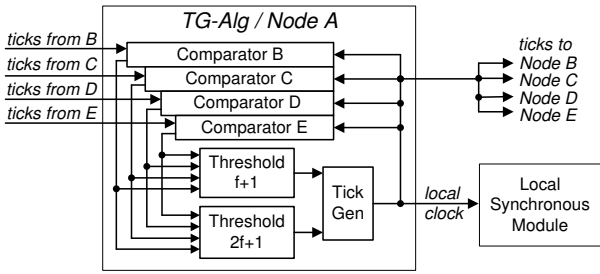


Fig. 2. Refined tick generation design

As part of the TG-Alg a set of $n - 1$ comparators is employed in every node (A in the figure) to determine whether the considered node (A) has received more ticks from the respective other node (B, C, D, E) than generated by itself. The comparators' input paths coming from the other nodes are called "remote" paths, while the inputs fed with the local clock are called "local" inputs. The associated path delays are called remote delay $\tau_{r,s}^{\text{rem}}$ for the delay from node r to node s and local delay $\tau_{r,s}^{\text{loc}}$ for the delay from node r to its own comparator responsible for node s , respectively. The comparison status is provided to two threshold gates that implement the algorithm's two rules. As soon as one of the two thresholds is exceeded, the tick generation unit is triggered to issue the respective tick.

As mentioned above, for implementation in hardware the original algorithm had to be further refined: The growing (k) messages could not be transported over the TG-Net and thus had to be replaced by "anonymous" binary signal transitions. The detailed discussion of the implementation can be found in [21]. In [22] it has been formally proven at this level of abstraction that, if a set of constraints on the delays inside the TG-Alg and TG-Net hold, then all correct TG-Algs provide their SoC modules with a synchronized clock in spite of f Byzantine (arbitrary) faulty TG-Algs in a system with $n \geq 3f + 2$ nodes — that is: They solve the synchronization problem with constraints: (i) *precision*: For any two correct nodes p, q and time t , the tick numbers sent by p and q until any time t do not differ by more than a constant π and (ii) *accuracy*: For any correct process p and any time interval I , the number of tick messages sent by p during I , say $\#_p(I)$, is bounded by $a|I| + b \leq \#_p(I) \leq c|I| + d$ for constants a, b, c, d .

In this paper, however, we concentrate on the design shown in Fig. 2, as it is easier to follow and still reflects the characteristics we encountered with our hardware implementation.

III. STABILITY AND ROBUSTNESS OF DARTS

From [22] we know worst case measures of the algorithm depicted in Fig. 2: Given the local and remote delays, we can calculate bounds for precision and accuracy. While this allows a worst case characterization of DARTS yielding border constraints on the local clock signals, we cannot deduce the clock signal's behavior within these bounds. For example we do not know whether a correct node exhibits high differences in its round times (the duration between the sending of tick $\langle k \rangle$ and tick $\langle k + 1 \rangle$), and if so whether these irregularities occur in a predictable, systematic pattern. We will call the absence

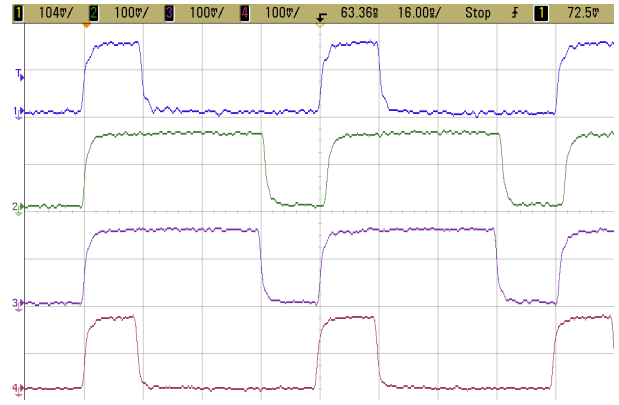


Fig. 3. Measured DARTS clocks over time. Top to bottom: A, B, C, E.

of irregularities *stability*, without giving a formal definition.

Another question of interest not answered by our worst-case characterization is the system's *robustness*. Informally, we will call a tick generation algorithm robust (with respect to parameter set \mathcal{P}), if changing one of \mathcal{P} 's parameters has only a small relative effect on the correct node's round times.

Clearly it is desirable to build stable and robust systems. In case of DARTS the parameter set of interest is the set of all remote delay $\tau_{r,s}^{\text{rem}}$ and local delay $\tau_{r,s}^{\text{loc}}$ parameters for all nodes r, s . Note that by this assumption we consider all influences from the environment (temperature, supply voltage, noise) reflected by potential changes in just these delays that are under our full control in the following.

Measurements of our DARTS system comprising five ASIC nodes, suggests that stability and robustness do not come for free: Fig. 3 shows an instable DARTS system with periodically varying round times.

To investigate whether DARTS is stable and robust, a system model had to be found which both supports simulation and formal analysis. It turned out that the distributed algorithm can be naturally formalized as a set of (non-linear) difference equations. For this purpose we denote the time node p sends tick $k \geq 0$ with $t_p[k]$. $t_p[k], k \geq 1$ then can be expressed as a function with parameters from $t_r[k - 1], \tau_{r,s}^{\text{rem}}$ and $\tau_{r,s}^{\text{loc}}$, where $r, s \in P$. The non-linearities stem from the use of max, min terms in the formal representation of the threshold gates.

A. Initial simulation results

We have simulated a DARTS system with $n = 5$ (and hence $f = 1$) in a fault-free case, both with respect to stability and with respect to robustness. The difference equations have been numerically solved with MATLAB, and in the following we present typical simulation results.

a) *Stability*: Here we (i) chose the remote delays randomly according to a normal distribution $N(\mu, \sigma^2)$ with $\mu = 9 \cdot 10^{-9}$ and $\sigma^2 = 10^{-9}$, (ii) chose the local delays randomly from a $N(5 \cdot 10^{-9}, 0.5 \cdot 10^{-9})$ normal distribution and (iii) calculated $t_p[k]$ for every $p \in P$ and $0 \leq k \leq 100$, keeping the delay values unchanged.

Fig. 4 shows the round times of all five nodes obtained in two simulation runs with different choices of the delays. The

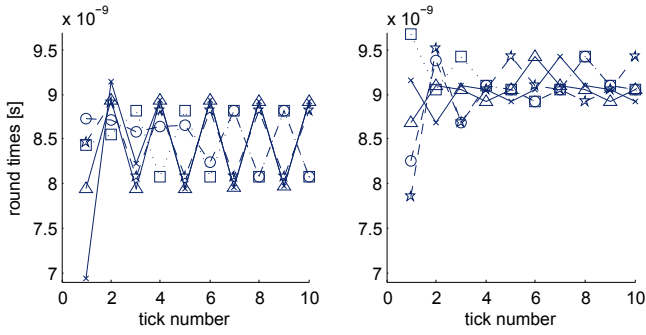


Fig. 4. Round times of simulated five node DARTS system, (a) short and (b) long stabilizing prefix

result is quite surprising in several respects: First, one would expect the round time to stabilize and remain constant, at least after the decay of potential transient effects; after all we keep the delays constant during the whole simulation run. As can be seen in the figures, this is not the case: the round times show a significant variation (about 10% around the average in both cases) that appears to be periodic. The second remarkable observation is that shape and period of this variation are largely different for the two simulation runs, which indicates a strong dependence on the choice of the delays.

b) Robustness: Here we systematically varied one selected delay, namely $\tau_{E,D}^{\text{rem}}$, while keeping all other delays fixed at the value determined through an initial random selection (distributions like above). Fig. 5 shows the (mean) round time observed for node A over $\tau_{E,D}^{\text{rem}}$ in two different simulations (i.e. with different choices for the other delays).

Fig. 5 (a) meets our expectation quite well: There is a limited range within which $\tau_{E,D}^{\text{rem}}$ has a (linear) impact on the round time. If the value of $\tau_{E,D}^{\text{rem}}$ becomes too small, it cannot contribute to the set of slowest paths and if it becomes too large, it contributes to a “slowest path” that is masked by the threshold function. In both cases its actual value becomes

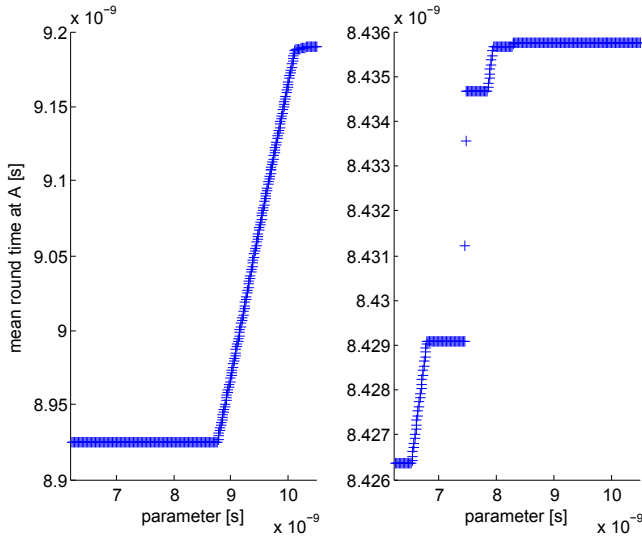


Fig. 5. Parameter variation, (a) ramp and (b) multiple plateaus.

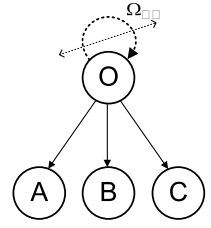
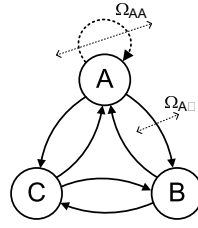


Fig. 6. 3 node wait-for-all system Fig. 7. 3 node synchronous system

irrelevant. We would further expect that position, cut-off and width of the ramp are determined by the other delays, and this is the main difference we would therefore anticipate in further simulation runs. Fig 5 (b), however, shows a different type of dependence: We can observe one or even more distinct plateaus in the curve, where the value of $\tau_{E,D}^{\text{rem}}$ is also irrelevant in a limited interval, with intervals of linear dependence (but different slope!) in between.

Clearly an analytic treatment like finding an explicit function $t_p[k]$ dependent on the local and remote delays only would clarify the involved processes. There exists interesting work on the analysis of periodic behavior on min-max functions: In [23] a general treatment of two dimensional min-max functions is provided (corresponding to a two node tick generation system, here). [24] states and [25] proves the duality conjecture, which reduces the problem of finding the average round time $\lim_{k \rightarrow \infty} (t_p[k]/k)$ of a min-max system to finding it in a set of max only systems. Unfortunately, however, stability and robustness cannot be deduced from the average round time. Summarizing, we must admit that the non-linear difference equations are too complex to solve ad hoc by usual means. We hope to gain a better understanding that might help us solving the DARTS solution from an investigation of less complex systems and further generalization. The next section will thus be devoted to the analysis of these systems.

IV. ANALYSIS OF SIMPLIFIED SYSTEMS

A. Stability

The simplest synchronizing systems are those where the algorithm waits until it has received $\langle k \rangle$ on all its incoming links and then sends out $\langle k + 1 \rangle$. We will call this class of algorithms “wait-for-all” algorithms. Consider a distributed system consisting of three nodes $P = \{A, B, C\}$ with links between all pairwise distinct node-pairs, and a loop at node A , cf. Fig. 6.

Interestingly for these systems, there exists a calculus from non-linear control theory which yields an explicit formula for $t_A[k]$ (see [26] for an introduction).

Let Ω be the adjacency matrix of the communication graph, i.e., $\Omega_{p,q} := \omega(\langle p, q \rangle)$. The equation for $t_A[k]$, $k \geq 1$ can then be formulated as

$$t_A[k] = \max \left\{ \begin{array}{l} t_A[k-1] + \Omega_{A,A} \\ t_B[k-1] + \Omega_{B,A} \\ t_C[k-1] + \Omega_{C,A} \end{array} \right\} \quad (1)$$

and analogous equations are obtained for $t_B[k]$ and $t_C[k]$. There exists a convenient notation that allows to express this

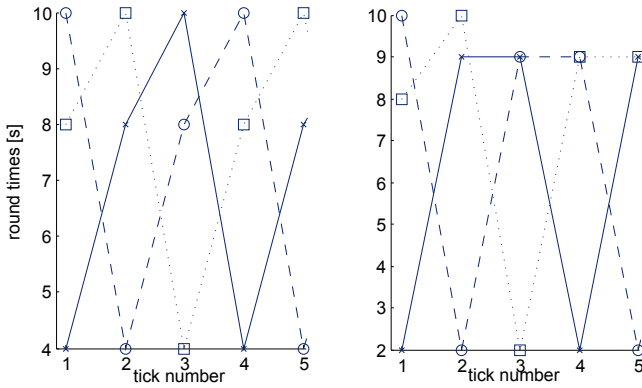


Fig. 8. A (x), B (\square), C (\circ), (a) Example 1, (b) Example 2

system as a matrix multiplication after transforming it to max-plus algebra, namely

$$\begin{pmatrix} t_A[k] \\ t_B[k] \\ t_C[k] \end{pmatrix} = \Omega^T \begin{pmatrix} t_A[k-1] \\ t_B[k-1] \\ t_C[k-1] \end{pmatrix} \quad (2)$$

The max-plus matrix multiplication is performed as in linear algebra with subsequent replacement of $+$ with \max and \cdot with $+$. The solution to solving this equation is inspired by an equivalent problem from graph theory: How long is the longest (with respect to the ω weights) path of k hops with sink p . Interestingly the answer to the latter is $t_p[k]$.

Since we consider only finite node systems, our communication graphs are finite and thus for sufficiently large k must contain cycles (closed path with distinct vertices in between). In case of our three node system, every path of length ≥ 3 must contain at least one cycle. In most systems (including our three node system), there exist more than one possible cycles. Let $\mathcal{C} \subseteq E$ be a cycle, then $\frac{\sum_{e \in \mathcal{C}} \omega(e)}{|\mathcal{C}|}$ is the mean cycle weight. It can be formally proven that if there exists a cycle \mathcal{C}_{\max} with maximum mean cycle weight then the maximum paths with sink p comprise (i) a prefix, (ii) a number of cycles through \mathcal{C}_{\max} and (iii) a postfix. In case p lies itself on cycle \mathcal{C}_{\max} , the prefix will be empty.

In the following we will consider four typical weight assignments Ω resulting in different round times each.

Example 1. Let all delays be 1, except for $\Omega_{A,B} = 10, \Omega_{B,C} = 8$ and $\Omega_{C,A} = 4$. The mean cycle weight of cycle $\mathcal{C}_{\max} = \{A, B, C\}$ is $\frac{10+8+4}{3}$, which represents the maximum. The resulting round times are depicted in Fig. 8 (a).

It can be observed, that the system has no stabilizing prefix and that its round times vary with a period of three. For example A's round times are 4, 8, 10, 4, \dots . By definition all nodes boot at time 0 and initially send tick $\langle 0 \rangle$, thus $t_A[0] = 0$. To solve $t_A[k]$ for $k \geq 1$, we consider the equivalent graph theoretic problem and observe, that the longest 1-hop path with sink A is of length 4 ($\langle C, A \rangle$), i.e., $t_A[1] = 4$. Further the longest 2-hops path with sink A has length $4 + 8$ ($\langle B, C \rangle, \langle C, A \rangle$) yielding $t_A[2] = 4 + 8$ and the longest 3-hop path with sink A has length $4 + 8 + 10$ ($\langle A, B \rangle, \langle B, C \rangle, \langle C, A \rangle$), yielding $t_A[3] = 4 + 8 + 10$. By then the cycle \mathcal{C}_{\max} is closed

and the process starts again. The round times are exactly the delays 4, 8, 10, 4, \dots .

Example 2. This example is only slightly different from Example 1: Let all delays be 1, except for $\Omega_{A,B} = 10, \Omega_{B,C} = 8$ and $\Omega_{C,A} = 2$. Clearly, again, the mean cycle weight of $\mathcal{C}_{\max} = \{A, B, C\}$ represents the maximum, namely $\frac{10+8+2}{3}$. Round times are depicted in Fig. 8 (b).

Here the maximum 1-hop path with sink A is of length 2 ($\langle C, A \rangle$), the maximum 2-hop path with sink A is of length $10 + 1$ ($\langle A, B \rangle, \langle B, A \rangle$) and the maximum 3-hop path with sink A is of length $10 + 8 + 2$ ($\langle A, B \rangle, \langle B, C \rangle, \langle C, A \rangle$) closing the cycle \mathcal{C}_{\max} . Round times at A are thus 2, 9, 9, 2, \dots . Clearly a view from B involves different paths: Although B 's longest path cycles through \mathcal{C}_{\max} , too, it comprises of a different prefix and postfix, leading to different round times as can be seen in Fig. 8 (b).

Example 3. This time we set all edge weights along \mathcal{C}_{\max} to 10, i.e., all delays are 1, except for $\Omega_{A,B} = 10, \Omega_{B,C} = 10$ and $\Omega_{C,A} = 10$. Round times then can be observed not to oscillate, since the maximum 1-hop, 2-hop and 3-hop path differ by the same amount 10.

Example 4. The same effect of non-oscillatory behavior is obtained if \mathcal{C}_{\max} has exactly one hop. This scenario is obtained by increasing $\Omega_{A,A}$ (e.g., $\Omega_{A,A} = 10$, while all other weights are 1).

In summary we can conclude that oscillatory behavior of round times after the transient execution prefix is not due to the addition of fault-tolerance to the synchronization algorithm—“wait-for-all” algorithms exhibit the same behavior.

B. Robustness

We will consider two types of systems and investigate their robustness to delay variations: (1) synchronous design that should serve as a reference, and (2) a “wait-for-all” design.

Example 1. A three node (A, B, C) synchronous design can be modelled as a four node system with the additional node O modeling the oscillator. The communication graph comprises edges from O to all nodes (including itself) and is depicted in Fig. 7. The parameter varied is $\Omega_{O,O}$. Clearly the mean round duration of node A is $\Omega_{O,O}$, which forms the only cycle. Any variation in $\Omega_{O,O}$ will be directly mapped to the round time.

Example 2. Next we vary parameter $\Omega_{A,B}$ in a three node “wait-for-all” system with all other weights set to 1. The resulting mean round durations are plotted in Fig. 9. From the considerations of the last section we know that here $\mathcal{C}_{\max} = \{\langle A, B \rangle, \langle B, A \rangle\}$ and the mean round duration is given by $\frac{\Omega_{A,B} + 1}{2}$. This results in only 50% of the variations of $\Omega_{A,B}$ being visible in the round time. Moreover, we observe a lower cut-off for $\Omega_{A,B} < 1$ which can be explained by theory as well: Since, all other weights are 1, for $\Omega_{A,B} < 1$ the edge $\langle A, B \rangle$ is not in the cycle \mathcal{C}_{\max} determining the round time.

In summary we can confirm our initial expectation that the non-synchronous system exhibits higher robustness against parameter variations, at least for this simple case. While synchronous systems depend linearly on the varied delay

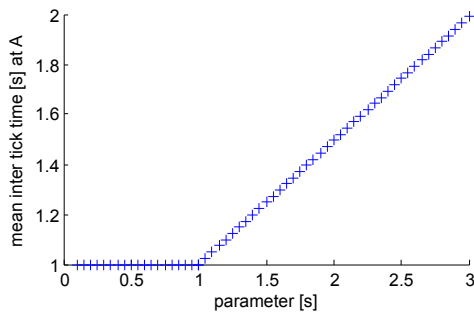


Fig. 9. “Wait-for-all” algorithm, mean round duration.

parameter, “wait-for-all” systems have a (different) linear dependence with lower cut-off. As soon as fault tolerant systems like DARTS are considered, they possess an additional upper cut-off, as already discussed in context with Fig. 5 (a).

V. CONCLUSION AND REMAINING QUESTIONS

We have analyzed the timing behavior of loop structures built from non-synchronous modules, where the operation of one module is triggered by more than one others, possibly by majority voting. Such architectures may form an attractive alternative to synchronous systems, offering higher flexibility for parameter variations as well as fault tolerance. Our example was the DARTS clock generation scheme that is composed of asynchronous components whose communication generates a globally synchronized fault tolerant clock.

We have pointed out that the execution runs of such systems (the clock period in DARTS) tend to show a periodically varying duration, even in a perfectly stable environment and in the stationary case. By application of results from non-linear control theory we have been able to model and predict this behavior, and to derive conditions for the existence of these variances. So far, however, we have only been able to treat specific cases, in particular those where no faults on the inputs can be tolerated. As soon as majority voting is introduced, the complexity of the problem becomes substantially higher. This case, however, is extremely relevant for fault-tolerant architectures in practice and therefore needs further investigation. It appears that such systems will be more robust against (static and dynamic) parameter variations, but this also needs confirmation by careful studies. In particular the question arises, whether normally distributed variations are projected to normally distributed variation of the execution run duration. If so, a characterization of such systems by the usual parameters, such as phase noise or Allen variance, would be interesting, while otherwise a suitable alternative statistical approach has to be sought.

REFERENCES

- [1] “International technology roadmap for semiconductors,” 2007. [Online]. Available: <http://www.itrs.net>
- [2] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, “Parameter variations and impact on circuits and microarchitecture,” *Proceedings of the Design Automation Conference, 2003*, pp. 338–342, June 2003.
- [3] O. Unsal, J. Tschanz, K. Bowman, V. De, X. Vera, A. Gonzalez, and O. Ergin, “Impact of parameter variations on circuits and microarchitecture,” *IEEE Micro*, vol. 26, no. 6, pp. 30–39, Nov.-Dec. 2006.
- [4] K. Bowman, S. Duvall, and J. Meindl, “Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration,” *IEEE Journal of Solid-State Circuits*, vol. 37, no. 2, pp. 183–190, Feb 2002.
- [5] V. Khandelwal and A. Srivastava, “Variability-driven formulation for simultaneous gate sizing and postsilicon tunability allocation,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 4, pp. 610–620, April 2008.
- [6] A. J. Martin, “Limitations to delay-insensitivity in asynchronous circuits,” in *Sixth MIT Conference on Advanced Research in VLSI*, 1990, pp. 263–278.
- [7] D. M. Chapiro, “Globally-Asynchronous Locally-Synchronous Systems,” Ph.D. dissertation, Stanford University, Oct. 1984.
- [8] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi, “Modeling the effect of technology trends on the soft error rate of combinational logic,” *Proceedings of International Conference on Dependable Systems and Networks, DSN*, pp. 389–398, 2002.
- [9] T. Karnik, P. Hazucha, and J. Patel, “Characterization of soft errors caused by single event upsets in CMOS processes,” *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 2, pp. 128–143, April-June 2004.
- [10] M. J. Gadlage, P. H. Eaton, J. M. Benedetto, M. Carts, V. Zhu, and T. L. Turflinger, “Digital device error rate trends in advanced CMOS technologies,” *Nuclear Science, IEEE Transactions on*, vol. 53, no. 6, pp. 3466–3471, Dec. 2006.
- [11] L. Wissel, S. Pheasant, R. Loughran, C. LeBlanc, and B. Klaasen, “Managing soft errors in ASICs,” *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 85–88, 2002.
- [12] R. Baumann, “Soft errors in advanced computer systems,” *IEEE Design & Test of Computers*, vol. 22, no. 3, pp. 258–266, May-June 2005.
- [13] N. Seifert, P. Shipley, M. Pant, V. Ambrose, and B. Gill, “Radiation-induced clock jitter and race,” in *Proceedings 43rd Annual IEEE International Reliability Physics Symposium*, 17–21, 2005, pp. 215–222.
- [14] V. Ferlet-Cavrois, P. Paillet, D. McMorrow, A. Torres, M. Gaillardin, J. Melinger, A. Knudson, A. Campbell, J. Schwank, G. Vizkelethy, M. Shaneyfelt, K. Hirose, O. Faynot, C. Jahan, and L. Tosti, “Direct measurement of transient pulses induced by laser and heavy ion irradiation in deca-nanometer devices,” *IEEE Transactions on Nuclear Science*, vol. 52, no. 6, pp. 2104–2113, Dec. 2005.
- [15] M. Nicolaidis, “GRAAL: A fault-tolerant architecture for enabling nanometric technologies,” *Proceedings 13th IEEE International On-Line Testing Symposium (IOLTS’07)*, pp. 255–255, July 2007.
- [16] F. Martorell, S. Cotoana, and A. Rubio, “An analysis of internal parameter variations effects on nanoscaled gates,” *IEEE Transactions on Nanotechnology*, vol. 7, no. 1, pp. 24–33, Jan. 2008.
- [17] L. Lamport, “The mutual exclusion problem: Part I—the theory of interprocess communication,” *Journal of the ACM*, vol. 33, no. 2, pp. 313–326, 1986.
- [18] —, “Arbitration-free synchronization,” *Distributed Computing*, vol. 16, no. 2/3, pp. 219–237, September 2003.
- [19] T. K. Srikanth and S. Toueg, “Optimal clock synchronization,” *Journal of the ACM*, vol. 34, no. 3, pp. 626–645, Jul. 1987.
- [20] J. Widder and U. Schmid, “Bootstrapping clock synchronization in partially synchronous systems with hybrid process and link failures,” *Distributed Computing*, vol. 20, no. 2, pp. 115–140, Aug. 2007.
- [21] M. Ferringer, G. Fuchs, A. Steininger, and G. Kempf, “VLSI Implementation of a Fault-Tolerant Distributed Clock Generation,” *IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT2006)*, Oct. 2006.
- [22] M. Fuegger, U. Schmid, G. Fuchs, and G. Kempf, “Fault-Tolerant Distributed Clock Generation in VLSI Systems-on-Chip,” *Sixth European Dependable Computing Conference (EDCC-6)*, Oct. 2006.
- [23] J. Gunawardena, “Periodic behaviour in timed systems with and or causality. part I: Systems of dimension 1 and 2.” Stanford University, Computer Science Dept., Tech. Rep., 1992.
- [24] —, “Cycle times and fixed points of min-max functions,” in *11th International Conference on Analysis and Optimization of Systems*. Springer, 1994, pp. 266–272.
- [25] S. Gaubert and J. Gunawardena, “The duality theorem for min-max functions,” HP Laboratories Bristol, Tech. Rep., 1997.
- [26] B. Heidergott, G. J. Olsder, and J. von der Woude, *Max plus at work*. Princeton Univ. Press, 2006.