Byzantine Self-Stabilizing Clock Distribution with HEX: Implementation, Simulation, Clock Multiplication

Martin Perner, Martin Sigl, Ulrich Schmid
Vienna University of Technology
Vienna, Austria
{mperner,msigl,s}@ecs.tuwien.ac.at

Christoph Lenzen
Massachusetts Institute of Technology
Cambridge, MA, USA
clenzen@csail.mit.edu

***Abstract*—We present a prototype implementation and simulation-based evaluation of a recently proposed novel approach for Byzantine fault-tolerant and self-stabilizing clock distribution in multi-synchronous GALS architectures. Fault-tolerant clock generation and clock distribution is a mandatory prerequisite for highly dependable multicore processors and Systems-on-Chip, as it removes the single point of failure typically created by central oscillators and conventional clock distribution trees. Our scheme, termed HEX, is based on a hexagonal grid topology, which connects simple intermediate nodes implemented using the UMC 90 nm standard cell library. Their purpose is to (i) forward synchronized clock signals throughout the grid and (ii) to supply the clock to nearby application modules. To achieve (ii), we show how to construct a *fast clock* on top of the clock signal provided by HEX and analyze its properties. In sharp contrast to existing solutions, HEX is not only Byzantine fault-tolerant, but also self-stabilizing, i.e., it can recover from arbitrarily corrupted system states. ModelSim-based simulation experiments confirm the excellent performance and fault-tolerance properties of our approach achieved in practice, which were already suggested by an earlier theoretical worst-case analysis.**

Keywords*—*fault-tolerance; self-stabilization; clock distribution; fault-injection; simulation analysis

I. INTRODUCTION

Thanks to the advances in *very large scale integration* (VLSI) technology, which nowadays allows clock speeds in the GHz range, complex hardware architectures such as multi-core processors and *Systems-on-Chip* (SoC) typically comprise multiple clock domains: Individual system components (e.g., a single core) execute synchronously, driven by a common clock signal. Different components may reside in different clock domains, which, however, are driven by clock signals generated by multiple unsynchronized clock sources. Communication between components in different domains must hence be synchronized or performed asynchronously, as in classic *globally asynchronous locally synchronous architectures* (GALS) [1].

This material is based upon work supported by the National Science Foundation under Grant Nos. CCF-AF-0937274, CNS-1035199, 0939370-CCF and CCF-1217506, the AFOSR under Contract No. AFOSR Award number FA9550-13-1-0042, the Swiss Society of Friends of the Weizmann Institute of Science, the German Research Foundation (DFG, reference number Le 3107/1-1), and the Austrian Science Foundation (FWF) project FATAL (P21694).

Multisynchronous GALS [2][3] architectures assume bounded synchrony (a maximum skew of a few clock cycles) also between different clock domains. This *mesochronous* clocking [4] not only allows to build a global notion of time throughout the chip, which, e.g., facilitates time-triggered transmission scheduling/routing in *Networks-on-Chip* (NoC) like Aelite [5], but also enables metastability-free high-speed cross-domain communication via FIFO buffers [6][7].

As mesochronous clocking inherently facilitates *distributed* clock generation, it also allows to address the lacking robustness of conventional centralized clocking approaches: If the central oscillator or some wire in the clock tree (used for distributing the clock signal to the functional units on the chip) breaks in such an architecture, even replicated functional units are of no use. This is also true for the few non-fault tolerant approaches for distributed mesochronous clock generation [8][9][10][11][12][13] described in literature, which either use distributed ring oscillators or distributed *phase-locked loops* (PLL). The only fault-tolerant clock generation approaches we are aware of are our Byzantine fault-tolerant DARTS [14][15] and our self-stabilizing Byzantine fault-tolerant FATAL approach [16]. However, both approaches focus on the distributed generation of a small number of synchronized clock signals and do not address the question of how to distribute these to a large number of functional units.

In [17], we proposed a novel approach, termed HEX, for reliably distributing a synchronized clock signal in multi-synchronous GALS systems. It works with arbitrary clock sources, ranging from a central oscillator to a fully distributed synchronized clock generation scheme like DARTS or FATAL. HEX is based on a sufficiently connected wiring topology, namely, a *hexagonal grid*. Intermediate nodes placed at each grid point control when the clock signal transitions are forwarded to adjacent nodes, and supply the clock signal to the functional units in their vicinity via small local clock trees. This way, HEX ensures that the clock signals at physically close nodes are well-synchronized throughout the chip.

The analytical worst-case analysis presented in [17] revealed excellent synchronization and fault-tolerance properties. In particular, in sharp contrast to the alternative approaches [8][9][10][11][12][13], HEX supports multiple synchronized clock sources, tolerates Byzantine failures of both clock

 ISBN: 978-1-61208-301-8

sources and nodes as well as link failures, and self-stabilizes [18] quickly from arbitrary states, as, e.g., caused by an overwhelming number of transient failures. Its resilience to failures scales with the size of the grid, in the sense that it tolerates a constant density of isolated Byzantine nodes; it can handle an even larger number of more benign failures like broken wires and mute clock sources and nodes. Moreover, HEX obeys a fault locality property: The adverse effect of failures in the grid on the clock skew between non-faulty neighbors decreases with the distance.

Contributions: The present paper provides the first step towards a physical implementation of HEX and complements the theoretical analysis provided in [17]:

(i) We show how the HEX nodes can be implemented and synthesized using the UMC (United Microelectronics Corporation) 90 nm standard cell library.

(ii) We study the average-case performance of our HEX implementation, in particular, in the presence of faults, using ModelSim-based simulation and fault-injection in a custom testbed. Our results confirm that the quite "exotic" worst-case scenarios identified in [17] are extremely unlikely to occur in practice even in the presence of failures; note that the MATLAB-simulations described in [17] have only been devoted to the fault-free case.

Moreover, we address the question of how to utilize the synchronized clock generated by HEX in applications. We consider a high-speed point-to-point communication subsystem for this purpose, which is, e.g., instrumental for multi-hop communication between HEX nodes and thus for any higher-level communication service. It primarily requires a synchronized clock with high frequency (to facilitate high-speed communication) and small clock skew (to avoid large data buffers). The last major contribution of our paper is hence the following:

(iii) We show how to augment HEX nodes to generate a synchronized *fast clock*, and analyze its skew.

Our paper is organized as follows: Section II provides an overview of HEX and some results from our previous work [17]. All subsequent sections present novel results: In Section III, we briefly describe how to add a particular communication subsystem to the HEX nodes, and provide the resulting requirements for the fast clock. Section IV gives an overview of our HEX implementation and the features of our custom testbed, which has been used to obtain the results described in Section V. Finally, Section VI is devoted to the implementation and analysis of a fast clock suitable for our communication subsystem. Some conclusions and directions of further research in Section VII round off our paper.

II. Overview of HEX

HEX [17] assumes a system consisting of (simple) computing nodes that exchange zero-bit messages (i.e., the only information they contain is their occurrence time) over a directed cylindrical hexagonal grid $G = (V, E)$ defined as follows (see Figure 1): With $L \in \mathbb{N}$ denoting its length and $W \in \mathbb{N}$ its width, the set of nodes V is the set of tuples $(\ell, i) \in [L+1] \times [W]$. Herein, $[L+1] := \{0, \dots, L\}$ denotes the row index set, referred to as *layers*, and $[W] = \{0, \dots, W-1\}$ the column index set, referred to as *columns*, of the nodes in the grid. For each node $(\ell, i) \in V$, $\ell \in [L+1]$, $i \in [W]$, the following links are in E: (i) Incoming and outgoing links to neighboring nodes of the same layer, namely from (ℓ, i) to $(\ell, i-1 \bmod W)$, called the left neighbor of (ℓ, i), and to $(\ell, i+1 \bmod W)$, called the right neighbor, and vice versa from the left and the right neighbor to (ℓ, i); (ii) if (ℓ, i) is in a layer greater than 0, incoming links from $(\ell-1, i)$, called its lower left neighbor, and $(\ell-1, i+1 \bmod W)$, called its lower right neighbor; (iii) if (ℓ, i) is in a layer smaller than L, outgoing links to $(\ell+1, i-1 \bmod W)$, its upper left neighbor, and $(\ell+1, i)$, its upper right neighbor.

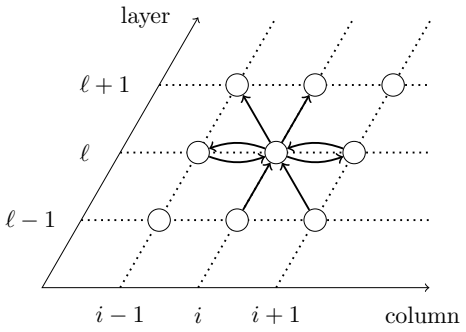

Fig. 1. Node (ℓ, i) and its incident links in the cylindrical hexagonal grid topology. Columns are modulo W and layers (rows) are between 0 and L.

1: **once** received trigger messages from (left and lower left) or (lower left and lower right) or (lower right and right) neighbors **do**
2: broadcast trigger message; ▷ local clock pulse
3: sleep for some time within $[T^-, T^+]$;
4: forget previously received trigger messages;

Fig. 2. Pulse forwarding algorithm for nodes in layer $\ell > 0$.

The failure model of HEX assumes that every node has at most one Byzantine faulty neighbor on an incoming edge, which can exhibit arbitrary behavior. However, a faulty node must not be able to prevent a correct neighbor from triggering correctly on behalf of its other neighbors. Hence, it is not allowed to output excessive voltages that destroy the receiver, or cause metastability that also upsets the receiver, for example. Note that tolerating two or more faulty neighbors would require an in-degree of at least 5, resulting in a non-planar communication graph due to the directed propagation of pulses utilized in HEX. All fault-free links in the graph respect FIFO order, with end-to-end delays that may vary non-deterministically within $[d^-, d^+] \subseteq (0, \infty)$.

Nodes at layer 0 execute a clock pulse generation algorithm like the ones of [15][16], whose purpose is to generate synchronized and well-separated initial messages (i.e., pulses). Nodes at layers larger than 0 run the simple pulse forwarding algorithm specified in Figure 2. Basically, nodes forward pulse k once they received trigger messages for pulse k from two adjacent neighbors. Therefore, synchronized pulses generated by the layer 0 nodes propagate as "waves" through the HEX grid up to the very last layer (cf. Figure 5).

We denote the distance of column i and j in the grid by $|i-j|_{\mathrm{w}} := \min\{(i-j) \bmod W, (j-i) \bmod W\}$ and the k^{th}

triggering time of node (ℓ, i), i.e., the time when it forwards the k^{th} pulse, by $t_{\ell,i}^{(k)}$. In [17], quite "exotic" (but possible) worst-case propagation paths have been used to develop the worst-case skew bounds. These bounds are very robust against initial skews, which are captured by the *skew potential on layer* 0 denoted by $\Delta_0 := \max_{i,j\in[W]}\{t_{0,i} - t_{0,j} - |i-j|_{\mathrm{w}} d^-\}$. The latter is always non-negative, and is 0 when nodes in layer 0 that are separated by h hops trigger their pulse at times that are at most hd^- apart. Dropping the superscript (k) for readability, we restate the main result from [17] in Theorem 1. From here on we require that $\varepsilon = d^+ - d^- \leq d^+/7$ holds.

Theorem 1 (Skew Bounds—Fault-free Case [17, Thm. 3.8]): Suppose that $\varepsilon \leq d^+/7$. Then the following upper bounds hold on the intra-layer skew $\sigma_\ell := \max_{i\in[W]}\{|t_{\ell,i} - t_{\ell,i+1}|\}$ in layer ℓ: If $\Delta_0 = 0$, then σ_ℓ is uniformly bounded by $d^+ + \lceil W\varepsilon/d^+\rceil\varepsilon$ for any $\ell \in [L+1]$. In the general case,

$$\forall \ell \in \{1,\dots,2W-3\}: \ \sigma_\ell \leq d^+ + 2W\varepsilon^2/d^+ + \Delta_0. \quad (1)$$

$$\forall \ell \in \{2W-2,\dots,L\}: \ \sigma_\ell \leq d^+ + \lceil W\varepsilon/d^+\rceil\varepsilon. \quad (2)$$

Moreover, regarding the inter-layer skew of layer $\ell \in [L]$ to its neighboring layer(s), it holds for all $i \in [W]$ that

$$t_{\ell,i} - \sigma_\ell + d^- \leq t_{\ell+1,i} \leq t_{\ell,i} + \sigma_\ell + d^+ \text{ and} \quad (3)$$

$$t_{\ell,i+1} - \sigma_\ell + d^- \leq t_{\ell+1,i} \leq t_{\ell,i+1} + \sigma_\ell + d^+. \quad (4)$$

In the presence of failures, the above worst-case skew is only moderately increased (depending on the number of faulty nodes encountered along the worst-case propagation path). In addition, Theorem 2 shows that HEX self-stabilizes quickly from an arbitrary system state, e.g., caused by excessive transient failures.

Theorem 2 (Stabilization [17, Thm. 3.11]): Suppose $\max_{k\in\mathbb{N}}\{\Delta_0^{(k)}\} \leq \Delta$ and denote $\sigma_0 := \Delta + d^-$. Assume that

$$\min_{i\in[W]}\{t_{0,i}^{(k+1)}\} \geq \max_{i\in[W]}\{t_{0,i}^{(k)}\} + Wd^+ + L\varepsilon + T^+, \quad (5)$$

that $T^- > \sigma_\ell + d^+ + \varepsilon$ for all $\ell \in [L+1]$, where σ_ℓ is as in Theorem 1 with $\Delta_0 = \Delta$, and that the pulse generation algorithm employed at layer 0 is self-stabilizing. Then, HEX self-stabilizes within L pulses once layer 0 stabilized, in the sense that each node triggers exactly once per pulse, and for each pulse the bounds from Theorem 1 apply.

III. An Application: HEX-based Communication

In order to add a point-to-point communication subsystem to HEX, the canonical choice is to augment every (unidirectional) link in the HEX grid by additional signal wires for data communication, in both directions. The result is a (cylindrical) hexagonal grid with bidirectional communication between adjacent nodes. Every node in the grid now consists of two distinct parts, an extended HEX node and a communication node that uses the HEX clock.

In order to demonstrate the benefits of a synchronized clock for communication, and to reason about desired properties, we briefly describe a communication solution based on bi-synchronous *first in first out* (FIFO) buffers (shortly denoted FIFO) that has been adopted from [19]. More specifically, every communication node has a FIFO for each incoming communication link. Each FIFO has a write interface for writing data into and a read interface for reading data from the FIFO. The write interface is accessed and clocked by the sender, the read interface by the receiver.

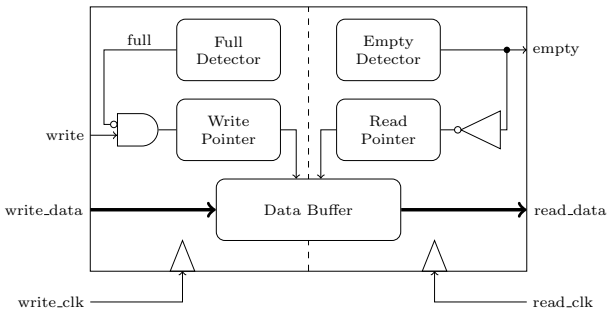

Fig. 3. Bi-Synchronous FIFO. Every Node has one FIFO for each incoming communication link. The sender is connected to the write interface, the receiver to the read interface.

The bi-synchronous FIFO is composed of five modules: *data buffer*, *read pointer*, *write pointer*, *full detector* and *empty detector* (see Figure 3). For the *data buffer*, we use a dual-ported *random access memory* (RAM). The sender- and the receiver-data lines are directly connected to the *data buffer*. The *read pointer*, as well as the *write pointer*, consist of a shift-register. The *read pointer* resp. *write pointer* determines the current read resp. write address. Furthermore, the two pointers are used by the *full detector* and the *empty detector*. *Write_data* is queued into the FIFO if *write* is high and *full* is low at the rising edge of *write_clk*. Data is dequeued to *read_data* if *empty* is low at the rising edge of *read_clk*. With every write access, the write address is incremented by one unless the FIFO is full. The read address is incremented by one with every read access, unless the FIFO is empty.

We have chosen this implementation because of its simple and metastability-aware design and its good fault-tolerance and containment properties. Notice that the obvious way, for a faulty sender to corrupt the corresponding receiver, is by writing to the address that is currently read and thus inducing metastability. Even if a faulty sender permanently sends messages, the full flag is set before the sender can write into the current read address. Thus the sender is prevented from corrupting the receiver. Another advantage arises from the fact that HEX provides bounded synchrony: A FIFO with a depth twice the synchronization precision plus one (plus a constant value to compensate the difference in end-to-end delays) is sufficient to ensure no message loss. Note carefully that this holds independently of the clock frequency (which determines the communication speed) and for any clock duty-cycle. In fact, the implementation also works with a quite irregular clock - as long as it remains synchronized. In Section VI, we will show how to build a fast clock atop of the synchronized HEX clocks, which allows high-speed communication.

IV. HEX Implementation and Testbed

Every HEX Node consists of a couple of very simple design entities. Its core is the asynchronous state machine shown in Figure 4, which implements the algorithm shown in Figure 2. It consists of three states only: *fire*, *sleep* and *ready*. The initial state is *ready*, where the state machine waits

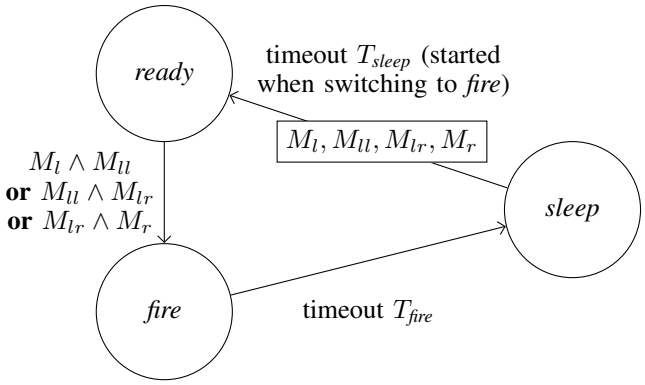

Fig. 4. Main state machine of a HEX node. Circular nodes represent states connected by transitions, labeled with their transition guards. A boxed node lists the memory flags to be cleared when taking the transition.

for the trigger condition in Figure 2 to become true. Resettable *memory flags* are used to memorize the occurrence of a pulse (indicated by active high (1)) from the neighbors. For example, the left neighbor is represented by the memory flag M_l. It is reset to low (0) when the state machine takes the transition from *sleep* to *ready*, as indicated by the boxed node. M_l is set to 1 when the input from the left neighbor becomes 1 after the most recent reset of M_l.

When the state *fire* is entered, both the sleep timer T_{sleep} and the minimal pulse duration timer T_{fire} are started and 1 is outputted on all outgoing links, to signal the pulse. After the timeout T_{fire} has occurred, the output is reverted to 0 and the state *sleep* is entered. The state *sleep* is a dormant state that is only left when the timeout of T_{sleep} (within $[T^-, T^+]$, as specified in Figure 2) occurs. Note that the reset of all memory flags occurring upon the transition to *ready* causes the node to forget all pulses sensed before.

The implementation of the above state machine is completely asynchronous and has been generated using the Petrify tool [20]. Both timers T_{sleep} and T_{fire} are driven by the same start/stoppable ring oscillator, which (like the memory flags) has already been used in our implementation of the FATAL$^+$ clock generation approach [16]. The resulting design of the HEX node was finally synthesized with Synopsis® Design Compiler version C-2009.06-SP4, using the UMC 90 nm standard cell library [21]. Note that we had to augment this library by a custom Muller C-Gate [22] developed in the context of the DARTS project [15][23]. Since an accurate timing characterization of this C-Gate is not available, its timing information was just copied from an AND-Gate of the standard library. The resulting inaccuracy is negligible w.r.t. our purposes, though.

The HEX grid itself was implemented by means of a custom testbench, which has the following purposes:

(1) Set the grid size and instantiate the corresponding number of nodes and interconnecting wires.
(2) Provide the layer 0 clock sources, i.e., generate the clock pulse of node $(0, i)$, $i \in [W]$, at time $t_{0,i}$, with some pre-selected skews. Clock sources for a single pulse and for multiple pulses are supported.
(3) Control the individual link delays during the simulation. Both random delays (uniform within lower and upper bound) and deterministic delays are supported.
(4) Control fault injection during the simulation: Both nodes and links can be declared *correct*, *Byzantine* (for each pulse and link, randomly choose output constant 0 or 1, which corresponds to no/fast pulse), or *fail-silent* (output constant 0). The selection of faulty nodes and/or links can be done deterministically or randomly (but static for multi-pulse simulations).

In order to support a reasonably systematic evaluation, we developed a software infrastructure in Haskell that allowed us to generate testbenches for different parameter settings (1)-(4). Every such testbench was then evaluated by means of pre-layout timing simulations, using Mentor Graphics® ModelSim 10.1d. The simulation results were recorded via event lists, which facilitated post-processing by our software infrastructure.

V. Simulation Results

The primary purpose of our simulation experiments is to complement the analytic worst-case bounds by statistics and average-case results, which are difficult to determine analytically. In view of the quite exotic worst-case scenarios obtained in [17], we (correctly) conjectured that the latter should be much better in reality. In the fault-free case, this has already been confirmed by means of high-level MATLAB-simulations of the algorithm in Figure 2 in [17]. The primary focus of the simulations presented in this paper, which employ the digital HEX implementation described in Section IV, lies (A) on scenarios including faulty nodes and (B) on stabilization time.

More specifically, using the testbed described in Section IV, we conducted the following simulation experiments:

(A) Statistical evaluation of the neighbor skews. These experiments require simulations involving a single pulse only. The primary quantities of interest here are:

- the (absolute) *layer ℓ intra-layer neighbor skews* $|t_{\ell,i} - t_{\ell,i-1}|$ of every node (ℓ, i) in layer ℓ,
- the (signed) *inter-layer neighbor skews* $t_{\ell,i} - t_{\ell-1,i}$ and $t_{\ell,i} - t_{\ell-1,i+1}$ of every node (ℓ, i) relative to its direct layer $\ell - 1$ neighbors $(\ell - 1, i)$ and $(\ell - 1, i + 1)$, respectively.

We remark that the former is defined in terms of the absolute values due to the symmetry of the topology (and thus skews) within a layer, whereas the latter respects the sign and thus correctly captures the non-zero bias (of at least d^-) in the inter-layer neighbor skew. Note that such a known bias can be compensated at the application layer if desired, see Section VI. Let $\sigma_\ell^{\mathrm{op}} := \mathrm{op}_{i\in[W]}\{|t_{\ell,i} - t_{\ell,i+1}|\}$ with $\mathrm{op} \in \{\mathrm{avg}, \max\}$ denote the average and maximum (absolute) layer ℓ intra-layer skew, respectively. Similarly, let $\hat{\sigma}_\ell^{\mathrm{op}} := \mathrm{op}_{i\in[W]}\{t_{\ell,i} - t_{\ell-1,i}, t_{\ell,i} - t_{\ell-1,i+1}\}$, where $\mathrm{op} \in \{\min, \mathrm{avg}, \max\}$, be the (signed) inter-layer skew between layer ℓ and $\ell - 1$. The global intra-layer resp. inter-layer skews in the entire system are defined as $\sigma^{\mathrm{op}} = \mathrm{op}_{\ell\in[L+1]}\{\sigma_\ell^{\mathrm{op}}\}$ resp. $\hat{\sigma}^{\mathrm{op}} = \mathrm{op}_{\ell\in[L+1]\setminus\{0\}}\{\hat{\sigma}_\ell^{\mathrm{op}}\}$.

(B) Statistical evaluation of the stabilization time. These experiments require multiple pulses. Essentially, the system is started, with every node in an arbitrary state, and then used to forward a sequence of correct pulses generated at layer 0.

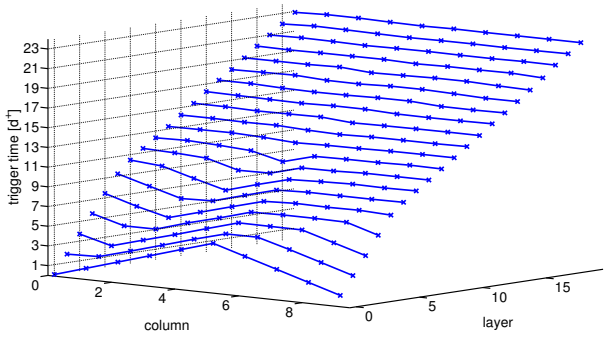

Fig. 5. Pulse wave propagation for uniformly chosen link delays in $[7, 8]$ and layer 0 neighbor skews ramping up/down by d^+.

TABLE I. INTRA- AND INTER-LAYER SKEW σ^{op} AND $\hat{\sigma}^{\text{op}}$ OF ALL NODES IN A 100×25 GRID FOR UNIFORM RANDOM END-TO-END DELAYS IN $[7.125, 8.165]$ NS. VALUES ARE DETERMINED OVER 300 TEST RUNS.

	intra-layer σ^{op}		inter-layer $\hat{\sigma}^{\text{op}}$		
init. layer 0	avg	max	min	avg	max
0	0.40	3.63	7.13	7.91	11.58
rand. $[0,d^-]$	0.46	6.97	7.13	7.94	15.07
rand. $[0,d^+]$	0.46	7.86	7.13	7.94	15.88
ramp d^+	1.41	8.16	0.96	8.36	16.28

Using post-processing of the recorded triggering times, we compute the stabilization time as the number of pulses needed for all intra- and inter layer skews to persistently go below their respective thresholds (determined according to $\sigma_\ell^{\max}$ and $\hat{\sigma}_\ell^{\max}$ determined in (A)). Unfortunately, lack of space does not allow us to adequately present these results in this paper.

Both types of experiments were performed with and without faulty nodes of different types. Note that the triggering times of faulty nodes are of course not considered when computing the inter- and intra-layer skews.

A. Neighbor skew evaluation

As an appetizer, Figure 5 shows a 3D plot of a typical pulse propagation wave in a fault-free grid with $W = 10$ and $L = 20$, with uniformly distributed link delays in $[7.125, 8.165]$ ns and layer 0 skews ramping up/down by d^+. The grid (sliced between width $W - 1$ and $0 \equiv W$) lies in the $(\ell \in [L + 1], i \in [W])$ plane, the z-axis gives the triggering time $t_{\ell,i}$ of the corresponding node (ℓ, i). To improve the readability of intra-layer skews, we connected all points $(\ell, i, t_{\ell,i})$ and $(\ell, i + 1, t_{\ell,i+1})$, $i \in \{0, \dots, W - 2\}$. It is apparent that the wave propagates evenly throughout the grid, nicely smoothing out differences in link delays and the large skews on layer 0.

To allow some qualitative comparison with the MATLAB-simulation results provided in [17], Table I shows the average (σ^{avg}) and maximal ($\sigma^{\max}$) intra-layer skew and the minimal ($\hat{\sigma}^{\min}$), average ($\hat{\sigma}^{\text{avg}}$), and maximal ($\hat{\sigma}^{\max}$) inter-layer skew, respectively, in the absence of faulty nodes. These values were computed over all nodes and 300 simulation runs, in the following setting (used throughout the remainder of this section): $L = 100$, $W = 25$ and link delays uniformly chosen within $[d^-, d^+]$ for $d^- = 7.125$ ns and $d^+ = 8.165$ ns ($\varepsilon = 1.04$ ns); these values result from combining assumed wire and routing delays within $[7, 8]$ ns with the switching delay bounds $[0.125, 0.165]$ ns determined during the HEX node synthesis. For the timeout T_{sleep}, a nominal value has been chosen that ensures a timeout within $[T^-, T^+]$ ns with $T^- = 20.393$ ns and $T^+ = 23$ ns, given the ring oscillator drift bounds determined during synthesis.

Table I shows the results of our experiments, using four different choices for the layer 0 skews between neighbors: The triggering times of the layer 0 nodes $t_{0,i}$ are (i) all 0 (resulting in $\sigma_0 = 0$ and skew potential $\Delta_0 = 0$), (ii) uniformly in $[0, d^-]$ (i.e., $\sigma_0 \approx d^-$ and $\Delta_0 = 0$), (iii) uniformly in $[0, d^+]$ (i.e., $\sigma_0 \approx d^+$ and $\Delta_0 \approx \varepsilon$), and (iv) ramping-up/down by d^+, i.e., $t_{0,i+1} = t_{0,i} + d^+$ for $0 \le i < W/2$ and $t_{0,i+1} = t_{0,i} - d^+$ for $W/2 \le i < W - 1$ (i.e., $\sigma_0 = d^+$ and $\Delta_0 \approx W\varepsilon/2 \approx 13$). Note that (iii) resp. (iv) reasonably model the average case and worst-case input provided by a layer 0 clock generation scheme with neighbor skew bound d^+, respectively. It is noteworthy that not a single instance in the collected data showed a skew of $\hat{\sigma}_\ell^{\max} > 2d^+$, and in scenarios (i) to (iii) we always had $\hat{\sigma}_\ell^{\min} < d^-$, i.e., all nodes were always triggered by their lower neighbors (obviously, this latter property is violated in scenario (iv) due to the excessive initial skews). The histogram of the skew distributions in case (i) are shown in Figure 6; the other cases look similar.

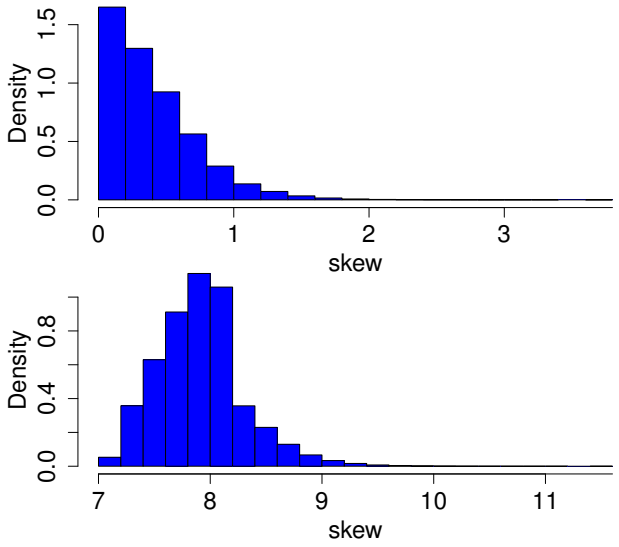

Fig. 6. Cumulated histograms for global intra-layer (top) and inter-layer (bottom) skew (in ns), from 300 simulation runs in scenario (i).

A comparison with the results of Theorem 1, which predicts $\sigma^{\max} \le 14$ ns and $[\hat{\sigma}^{\min}, \hat{\sigma}^{\max}] \subseteq [-5, 22]$ ns for scenarios (i) to (iii), reveals a much better behavior in the average case. Moreover, Figure 6 shows a sharp concentration with an exponential tail, i.e., *most* skews are very small in comparison to the worst-case bounds. For scenario (iv), the large initial skew (cf. Figure 5) leads to larger observed skews, which are also in accordance with our theorem, however.

Given the considerable differences between the minimum ($\hat{\sigma}^{\min}$) and maximum ($\hat{\sigma}^{\max}$) inter-layer skew in Table I, in conjunction with its non-zero bias, the question of layer-dependence arises. Table II provides $\hat{\sigma}_\ell^{\min}$, $\hat{\sigma}_\ell^{\text{avg}}$ and $\hat{\sigma}_\ell^{\max}$ for selected layers ℓ in case of scenario (iii). It reveals that there is no significant layer-dependent bias. Last but not least, the strong concentration of skews on each layer around the average

 ISBN: 978-1-61208-301-8

TABLE II. AVERAGE AND STANDARD DEVIATION OF $\hat{\sigma}_\ell^{\min}$, $\hat{\sigma}_\ell^{\text{avg}}$, AND $\hat{\sigma}_\ell^{\max}$, TAKEN OVER 250 SIMULATION RUNS OF SCENARIO (III).

layer ℓ	$\hat{\sigma}_\ell^{\min}$	$\hat{\sigma}_\ell^{\text{avg}}$	$\hat{\sigma}_\ell^{\max}$
1	7.19±0.04	9.01±0.22	14.22±0.78
2	7.20±0.05	8.41±0.15	12.88±1.03
3	7.21±0.05	8.18±0.12	11.73±1.20
5	7.21±0.06	8.02±0.08	10.33±0.97
9	7.23±0.07	7.94±0.06	9.36±0.52
10	7.22±0.06	7.94±0.06	9.29±0.47
11	7.22±0.06	7.92±0.06	9.23±0.45
12	7.22±0.06	7.93±0.06	9.17±0.45
15	7.23±0.07	7.92±0.06	9.11±0.35
19	7.21±0.06	7.92±0.06	9.09±0.36
20	7.23±0.06	7.92±0.06	9.11±0.36

(i.e., the very small standard deviation of $\hat{\sigma}_\ell^{\text{avg}}$), in particular in layers $\ell > 5$, shows that the smoothing out of local skews observed in Figure 5 is very typical.

Next, we consider the case where a certain number f of randomly chosen isolated nodes may be Byzantine faulty. All our simulations use the "average-case" scenario (iii) and the "worst-case" scenario (iv) described above. First, we choose f locations in the grid where faulty nodes are to be placed, subject to the constraint that no node has more than one faulty neighbor on an incoming edge. For a small number and a uniform distribution of faults, it is very likely that this constraint is satisfied. For an excessive number of faults, the birthday paradox kicks in: for $f = 5, 10, 20, 30$, the respective probabilities are roughly 0.95, 0.79, 0.36, and 0.09, respectively.

The behavior of the Byzantine faulty nodes is chosen randomly in every run on a per-link basis. Due to the usage of memory flags on the link port at the receiver side, a Byzantine faulty node has only two options: slowing down the firing of a node, or speeding it up. These two options are implemented as constantly sending pulses (i.e., constant high) and sending no pulse at all.

Then, 300 simulation runs of single pulse propagation are executed, each of which takes less than 2 minutes to simulate. We then later compute average and maximum of the intra-layer skew and minimum, average and maximum of the inter-layer skew. These computations where made several times on the generated data set, each time considering only pairs of correct nodes that are not reachable from any faulty node within at most $h = 0, 1, 2, 3$ directed hops in the grid. From these runs, we finally compute the average and standard deviation of σ^{avg}, $\sigma^{\max}$ and $\hat{\sigma}^{\min}$, $\hat{\sigma}^{\text{avg}}$, $\hat{\sigma}^{\max}$.

The results are shown in Table III. It is apparent that HEX copes very well with a considerable number of faults; minimal, average, and maximal skews are almost insensitive to the number of faults. For scenario (iv), we observe that standard deviations are generally larger, and that, in particular, nodes in directed distance 1 or 2 from faulty nodes experience larger skews. This is quite natural, given the large skews of the input; if the system suffers from a considerable number of faults, this limits its ability to reduce the initial skews.

Table IV shows the analogous results for fail-silent nodes; for brevity, we leave out $f = 2, 3$, for which we observed a behavior that interpolates between $f = 1$ and $f = 5$. The main difference to Byzantine faults are much more stable, but worse skews, the latter in particular demonstrated by $\hat{\sigma}^{\min}$. This can be easily understood, since the random behavior of Byzantine nodes increases the volatility of the setup, but decreases the number of "dead" links that inhibit the propagation of the pulse wave. With many faults, the wave needs to navigate a "maze" of dead nodes.

TABLE III. AVERAGE ± STANDARD DEVIATION OF σ^{avg}, $\sigma^{\max}$, $\hat{\sigma}^{\min}$, $\hat{\sigma}^{\text{avg}}$, AND $\hat{\sigma}^{\max}$, EXCLUDING ALL NODES WITH DIRECTED DISTANCE $\leq h$ FROM f ISOLATED BYZANTINE FAULTY NODES, OVER 300 SIMULATION RUNS OF SCENARIO (III) [TOP] AND (IV) [BOTTOM].

		Scenario (iii) Byzantine				
f	h	σ^{avg}	$\sigma^{\max}$	$\hat{\sigma}^{\min}$	$\hat{\sigma}^{\text{avg}}$	$\hat{\sigma}^{\max}$
1	0	0.54±0.05	7.76±0.92	6.95±0.36	7.98±0.03	15.56±1.02
1	1	0.54±0.05	7.35±0.58	7.12±0.06	7.98±0.03	15.13±0.63
1	2	0.53±0.05	7.21±0.58	7.13±0.03	7.97±0.02	14.99±0.62
1	3	0.53±0.05	7.11±0.57	7.13±0.02	7.97±0.02	14.89±0.62
2	0	0.60±0.07	8.15±0.87	6.75±0.79	8.00±0.03	16.09±1.11
2	1	0.59±0.07	7.56±0.48	7.12±0.07	8.00±0.03	15.39±0.52
2	2	0.58±0.06	7.36±0.51	7.12±0.06	8.00±0.03	15.17±0.56
2	3	0.57±0.06	7.23±0.52	7.13±0.04	7.99±0.03	15.01±0.59
3	0	0.65±0.07	8.48±0.87	6.33±1.36	8.03±0.04	16.47±1.27
3	1	0.64±0.07	7.73±0.35	7.10±0.12	8.03±0.03	15.56±0.32
3	2	0.63±0.07	7.53±0.39	7.12±0.07	8.02±0.03	15.35±0.36
3	3	0.62±0.07	7.38±0.44	7.12±0.07	8.02±0.03	15.21±0.41
5	0	0.76±0.10	8.91±1.01	5.54±1.99	8.09±0.05	17.13±1.72
5	1	0.74±0.09	7.86±0.29	7.05±0.21	8.08±0.05	15.70±0.28
5	2	0.72±0.09	7.70±0.30	7.10±0.09	8.07±0.05	15.53±0.30
5	3	0.70±0.09	7.59±0.33	7.11±0.07	8.06±0.04	15.40±0.33
10	0	0.95±0.12	9.55±1.18	3.62±2.59	8.18±0.06	18.85±2.52
10	1	0.91±0.11	8.01±0.38	6.89±0.54	8.16±0.06	15.84±0.17
10	2	0.87±0.11	7.84±0.17	7.05±0.15	8.14±0.05	15.63±0.19
10	3	0.84±0.11	7.75±0.22	7.07±0.14	8.13±0.05	15.53±0.22
20	0	1.27±0.13	10.61±1.46	0.65±2.54	8.34±0.07	21.29±2.17
20	1	1.18±0.13	8.45±1.15	6.20±1.51	8.31±0.07	15.92±0.15
20	2	1.11±0.13	7.96±0.12	6.96±0.23	8.28±0.06	15.75±0.18
20	3	1.05±0.13	7.88±0.16	7.01±0.17	8.25±0.06	15.64±0.19

		Scenario (iv) Byzantine				
f	h	σ^{avg}	$\sigma^{\max}$	$\hat{\sigma}^{\min}$	$\hat{\sigma}^{\text{avg}}$	$\hat{\sigma}^{\max}$
1	0	1.50±0.24	8.51±1.44	2.42±1.73	8.40±0.10	16.44±1.65
1	1	1.50±0.24	8.27±0.88	2.65±1.05	8.40±0.10	15.89±0.13
1	2	1.49±0.24	8.15±0.10	2.72±0.96	8.40±0.10	15.89±0.13
1	3	1.49±0.24	8.14±0.02	2.75±0.93	8.40±0.10	15.89±0.13
2	0	1.58±0.31	9.30±2.78	1.89±2.64	8.44±0.14	17.11±2.28
2	1	1.57±0.31	8.53±1.61	2.49±1.33	8.43±0.14	15.91±0.12
2	2	1.56±0.31	8.17±0.34	2.69±1.08	8.43±0.14	15.89±0.12
2	3	1.56±0.31	8.14±0.04	2.75±1.02	8.43±0.14	15.89±0.12
3	0	1.62±0.28	10.03±3.52	1.13±3.32	8.45±0.13	17.86±2.74
3	1	1.61±0.28	8.90±2.19	2.19±1.50	8.45±0.12	15.90±0.13
3	2	1.60±0.28	8.21±0.40	2.51±1.15	8.44±0.12	15.89±0.13
3	3	1.59±0.28	8.14±0.05	2.59±1.09	8.44±0.13	15.89±0.13
5	0	1.78±0.52	10.82±3.92	0.24±3.75	8.53±0.22	18.75±2.93
5	1	1.76±0.52	9.25±2.63	1.87±1.72	8.52±0.22	15.92±0.13
5	2	1.75±0.53	8.25±0.54	2.37±1.24	8.51±0.22	15.89±0.12
5	3	1.74±0.54	8.14±0.03	2.52±1.18	8.51±0.23	15.88±0.12
10	0	2.12±0.71	12.21±4.67	-1.36±4.18	8.67±0.29	21.09±2.61
10	1	2.09±0.72	10.06±3.31	1.09±2.11	8.66±0.29	15.93±0.13
10	2	2.06±0.73	8.39±0.90	1.89±1.55	8.65±0.30	15.89±0.13
10	3	2.04±0.76	8.14±0.11	2.12±1.53	8.64±0.31	15.88±0.15
20	0	2.59±0.98	16.20±7.91	-4.17±4.33	8.87±0.41	22.73±2.56
20	1	2.52±1.00	12.53±5.91	-0.14±2.25	8.85±0.41	16.16±1.93
20	2	2.48±1.04	9.20±4.99	1.20±1.66	8.83±0.43	16.01±1.74
20	3	2.44±1.07	8.68±4.84	1.63±1.71	8.82±0.46	15.92±1.20

VI. FAST CLOCKS

HEX provides each node with a local clock signal that is well-synchronized even in a system with multiple persistent

TABLE IV. AVERAGE ± STANDARD DEVIATION OF σ^{avg}, $\sigma^{\max}$, $\hat{\sigma}^{\min}$, $\hat{\sigma}^{\text{avg}}$, AND $\hat{\sigma}^{\max}$, EXCLUDING ALL NODES WITH DIRECTED DISTANCE $\leq h$ FROM f ISOLATED FAIL-SILENT NODES, OVER 300 SIMULATION RUNS OF SCENARIO (III) [TOP] AND (IV) [BOTTOM].

		Scenario (iii) fail silent				
f	h	σ^{avg}	$\sigma^{\max}$	$\hat{\sigma}^{\min}$	$\hat{\sigma}^{\text{avg}}$	$\hat{\sigma}^{\max}$
1	0	0.57±0.04	7.83±0.26	7.09±0.11	7.99±0.02	15.90±0.65
1	1	0.56±0.04	7.58±0.32	7.13±0.04	7.99±0.02	15.44±0.32
1	2	0.56±0.04	7.40±0.39	7.13±0.03	7.99±0.02	15.24±0.38
1	3	0.55±0.04	7.26±0.44	7.13±0.02	7.98±0.02	15.11±0.43
5	0	0.85±0.09	8.13±0.43	6.87±0.46	8.14±0.04	17.42±1.95
5	1	0.82±0.09	7.92±0.14	7.04±0.15	8.12±0.04	15.79±0.19
5	2	0.80±0.09	7.81±0.19	7.08±0.13	8.11±0.04	15.61±0.21
5	3	0.77±0.08	7.71±0.24	7.09±0.12	8.10±0.04	15.51±0.23
10	0	1.12±0.12	9.09±1.96	5.95±1.86	8.28±0.06	19.62±2.44
10	1	1.07±0.12	8.04±0.30	6.91±0.33	8.26±0.06	15.91±0.16
10	2	1.01±0.11	7.93±0.13	6.99±0.17	8.23±0.06	15.74±0.17
10	3	0.97±0.11	7.87±0.16	7.03±0.16	8.21±0.06	15.64±0.20
20	0	1.54±0.14	11.60±3.50	3.49±3.43	8.51±0.07	22.22±1.38
20	1	1.42±0.14	8.42±1.11	6.48±1.07	8.46±0.07	15.99±0.12
20	2	1.33±0.14	8.03±0.10	6.87±0.21	8.41±0.07	15.82±0.15
20	3	1.24±0.14	7.97±0.12	6.93±0.20	8.37±0.07	15.73±0.15
		Scenario (iv) fail-silent				
f	h	σ^{avg}	$\sigma^{\max}$	$\hat{\sigma}^{\min}$	$\hat{\sigma}^{\text{avg}}$	$\hat{\sigma}^{\max}$
1	0	1.50±0.05	9.49±3.96	1.95±2.91	8.40±0.02	16.71±1.98
1	1	1.50±0.05	8.69±1.87	2.62±1.18	8.40±0.02	15.90±0.13
1	2	1.49±0.05	8.17±0.26	2.86±0.83	8.40±0.03	15.89±0.13
1	3	1.49±0.05	8.14±0.02	2.87±0.83	8.40±0.03	15.89±0.13
5	0	1.77±0.09	14.46±6.75	-1.25±5.08	8.54±0.05	19.98±2.95
5	1	1.74±0.09	10.74±3.56	1.66±2.14	8.53±0.05	15.94±0.12
5	2	1.71±0.10	8.42±0.92	2.88±0.85	8.51±0.05	15.90±0.12
5	3	1.69±0.10	8.14±0.02	2.92±0.85	8.50±0.05	15.89±0.12
10	0	2.02±0.13	18.38±6.47	-4.03±5.21	8.67±0.06	22.04±2.09
10	1	1.96±0.13	12.68±3.92	0.59±2.61	8.64±0.07	15.97±0.12
10	2	1.91±0.14	8.74±1.33	2.92±1.00	8.62±0.07	15.91±0.11
10	3	1.87±0.15	8.15±0.22	3.03±0.94	8.60±0.08	15.89±0.11
20	0	2.40±0.16	22.58±4.20	-7.39±3.60	8.87±0.08	23.19±1.16
20	1	2.29±0.16	15.26±3.03	-1.08±2.66	8.83±0.08	16.03±0.11
20	2	2.20±0.17	9.49±1.85	2.81±0.97	8.78±0.09	15.92±0.11
20	3	2.12±0.19	8.15±0.15	3.11±0.91	8.74±0.10	15.89±0.12

faults. Furthermore, HEX is able to recover from an unbounded number of concurrent transient faults. It should not come as a surprise that these impressive fault-tolerance properties come at a cost: The HEX clock signal has a fairly low and unstable frequency and a large jitter. This makes HEX unsuitable for applications that need to determine real-time durations very accurately. One such negative example is clock multiplication based on PLLs, which sustain only moderate input jitter. Fortunately, as exemplified by our communication subsystem in Section III, many applications do not have such stringent requirements.

A. Trade-offs between Quality and Dependability

To understand that the undesirable properties of HEX are inherent to our approach, if not even inevitable under the given design goals, recall that Byzantine or fail-silent nodes locally affect the triggering times by (i) cutting off or delaying the clock distribution signal on a (shortest) path to some node or (ii) triggering a pulse early. Because of (ii), a node cannot locally trigger a pulse just based on one of its lower neighbors; because of (i), a faulty lower-left or lower-right neighbor entails that the node must be triggered with the help of a node in the same layer, thereby increasing the length of a causal chain [24] involved in triggering the node compared to the fault-free case. Consequently, the time when the node is triggered may be affected notably, and this effect may accumulate over several layers (in contrast to the skew, which is the *local* difference of triggering times).

From these observations, we can conclude that simultaneously guaranteeing tolerance of Byzantine faults *and* a stable clock frequency would entail a stronger connectivity of the grid and thus larger node degrees. In particular, nodes would need to receive clock signals from at least 3 nodes from the previous layer, as well as forwarding them to at least 3 nodes in the subsequent layer. Higher degrees, however, increase the complexity of nodes—and thus the likelihood that an individual node fails—as well as the probability that two neighbors are faulty (even for a fixed probability of failure). We hence conclude that there is a trade-off between frequency stability and resilience to failures.

Besides the inevitable switching delays of the components making up the HEX nodes, the low frequency of the generated HEX clock is also caused by our self-stabilization requirement: Ensuring that the nodes become ready for the next, well-synchronized pulse, a conservative pulse separation time must be granted to "flush out" spurious pulses from the system; otherwise, we might observe a phenomenon similar to "ventricular fibrillation".

B. Local Clocks

While increasing the frequency stability of the HEX clock signal would require a more dense topology, there is an obvious solution to the low clock frequency issue: frequency multiplication. By equipping each node with a high-frequency oscillator that is synchronized to the HEX clock, one can generate well-synchronized high-frequency clocks (termed *fast clocks* in the sequel).

However, synchronizing a fast clock to an unstable HEX clock involves a trade-off:

(a) If a stable fast clock frequency is desired, the HEX clock's jitter must be amortized over $m > 1$ pulses. For example, this could be implemented by first dividing the HEX clock by m and then using a PLL clock multiplier to generate the desired fast clock; m must be chosen appropriately to guarantee an acceptable PLL input clock jitter. Unfortunately, this approach may increase the skew of the fast clock considerably if the high-frequency oscillators driving the fast clocks have a large drift.

(b) If a fast clock with minimal skew is desired, which is our major objective, the HEX clock jitter must be amortized within a single pulse. Unfortunately, using solutions like [25] are complex and inherently lead to considerable frequency fluctuations of the fast clock.

With this in mind, we propose a simple, robust approach for (b) that achieves a high and reasonably stable fast clock frequency with good skews, at the expense of *burstiness*. For each HEX pulse, the fast clock generates a fixed number B of fast clock pulses [26] as shown in Figure 7. This is accomplished by means of a free-running start/stoppable ring oscillator, which is started by a HEX pulse and stops when it has generated B pulses; in fact, we may make use of the same oscillator design already used for the timeouts T_{fire} and T_{sleep} of the HEX state machine (Figure 4). Note that this can be implemented in a way that entirely avoids metastability.

 ISBN: 978-1-61208-301-8

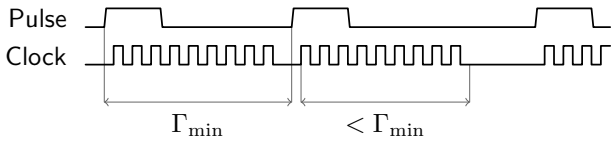

Fig. 7. A pulse generated by HEX will result in a fixed number of clock cycles being generated. The number of clock cycles and their frequency has to be chosen so that the required time span for them is less than the minimal time between pulses at correct nodes.

C. Analysis

First, we determine conservative timing constraints under the assumption that the system is fault-free. In this setting, the feasible number of clock cycles in each burst can be precisely expressed in terms of the *minimal pulse separation time*

$$\Gamma_{\min} := \inf_{\substack{i\in[W],\ell\in[L+1],k\in\mathbb{N}\\(\ell,i)\text{ correct}}} \{t_{\ell,i}^{(k)} - t_{\ell,i}^{(k-1)}\}. \tag{6}$$

Assuming that the fast clock sources are guaranteed to run at least with frequency $f_{\min}$, the number B of generated ticks per pulse must satisfy

$$B \leq f_{\min}\Gamma_{\min}. \tag{7}$$

Note that it is sufficient here if the frequency bound holds amortized over $\Gamma_{\min}$ time. To understand the resulting performance, assume for simplicity that this bound is an integer (i.e., we neglect that we may lose a "fractional tick") and choose $B = f_{\min}\Gamma_{\min}$. First, let us check the amortized frequency of the system. Clearly, we cannot guarantee a larger amortized frequency than $f_{\min}$. Of course, in addition $\Gamma_{\min}$ could be smaller than the long-term average time between pulses

$$\Gamma_{\text{avg}} := \lim_{k\to\infty} \frac{t_{0,0}^{(k)} - t_{0,0}^{(0)}}{k}. \tag{8}$$

Note that choosing the reference node to be $(0,0)$ in this definition is arbitrary, as using every other node must lead to the same result: Since the skew in the fault-free case is bounded, the influence of the difference of the triggering times vanishes in the limit (of course we assume here that the clock generation algorithm employed on layer 0 has bounded skew in the absence of faults). With this definition, the amortized frequency of the generated high-frequency clock of node (ℓ,i) can be expressed as

$$\lim_{k\to\infty} \frac{Bk}{t_{\ell,i}^{(k)} - t_{\ell,i}^{(0)}} = B \lim_{k\to\infty} \frac{k}{t_{0,0}^{(k)} - t_{0,0}^{(0)}} = f_{\min}\frac{\Gamma_{\min}}{\Gamma_{\text{avg}}}, \tag{9}$$

which equals $f_{\min}$ exactly if $t_{\ell,i}^{(k+2)} - t_{\ell,i}^{(k+1)} = t_{\ell,i}^{(k+1)} - t_{\ell,i}^{(k)}$ for all i, ℓ, and k, i.e., the HEX clock is perfectly stable. The "frequency loss" is determined by three factors:

1) The frequency fluctuation of the clock generation algorithm at layer 0.
2) The skew between the layer 0 nodes.
3) The variance in the speed at which pulses propagate through the grid.

The first two factors are determined by the clock generation algorithm and thus not to attribute to HEX. We remark that since we need to keep pulses well-separated, the first factor is likely to dominate the second. Similarly, large pulse separation times mitigate the influence of the third factor. If we increase the pulse separation time and take the limit, the resulting frequency of HEX will reflect the frequency provided by the clock generation algorithm at layer 0 (since the influence of skews vanishes in the limit). Hence, the frequency of the fast clocks will be $f_{\min}$ multiplied by the ratio between the frequency lower bound of the pulse generation algorithm—neglecting any additive variations that do not depend on the pulse separation time, in particular the skew between layer 0 nodes—and its average frequency.

These are good news, showing that a large pulse separation time does not hurt in terms of the overall frequency at which the system will be clocked. On the contrary, large pulse separation times essentially ensure the maximum frequency we can hope for! Even with fairly small pulse separation times, the system will run at a constant fraction of the frequency $f_{\min}$. However, there is no free lunch, as we will establish now by analyzing the fast clock skew of adjacent nodes.

Since pulses are anonymous, we interpret the generated local fast clock $L_{\ell,i}$ of node (ℓ,i) as a clock modulo B with the initial value being 0. For simplicity, we assume that the clock is continuous, i.e., it is real-valued from $[0,B)$, increasing modulo B at (possibly varying) rates from $[f_{\min}, f_{\max}]$, where $f_{\max}$ is its maximal frequency. The actual discrete clock reading at time t then is simply $\lfloor L_{\ell,i}(t)\rfloor$. With these definitions, the clock value at time $t \in [t_{\ell,i}^{(k)}, t_{\ell,i}^{(k+1)}]$ satisfies

$$\min\{f_{\min}(t - t_{\ell,i}^{(k)}), B\} \leq L_{\ell,i}(t) \leq \min\{f_{\max}(t - t_{\ell,i}^{(k)}), B\}, \tag{10}$$

since the clock is halted until the next pulse once it reaches value $B \equiv 0$.

Observe that at times where two adjacent nodes both locally triggered pulse k, but not pulse $k+1$, the worst possible clock skew is attained if (i) the difference in the triggering times is maximal, (ii) the clock of the node that triggered first runs at frequency $f_{\max}$ until its clock halts, and (iii) the other node's clock runs at frequency $f_{\min}$. For the skew between two adjacent nodes in layer ℓ, we thus get at times $t \in [t_{\ell,i}^{(k)}, t_{\ell,i}^{(k+1)}] \cap [t_{\ell,i+1}^{(k)}, t_{\ell,i+1}^{(k+1)}]$ that

$$\begin{aligned} &|L_{\ell,i}(t) - L_{\ell,i+1}(t)| \\ &\leq\ B - f_{\min}\left(\frac{B}{f_{\max}} - |t_{\ell,i}^{(k)} - t_{\ell,i+1}^{(k)}|\right) \\ &\leq\ \frac{f_{\max} - f_{\min}}{f_{\max}} \cdot B + f_{\min}\sigma_\ell^{\max} = \varrho B + f_{\min}\sigma_\ell^{\max}, \end{aligned} \tag{11}$$

where $\varrho := (f_{\max} - f_{\min})/f_{\max}$ is the *relative drift* of the high-speed clocks.

Now consider a time $t \in [t_{\ell,i}^{(k+1)}, t_{\ell,i}^{(k+2)}] \cap [t_{\ell,i+1}^{(k)}, t_{\ell,i+1}^{(k+1)}]$, i.e., node (ℓ,i) already triggered pulse $k+1$, but node $(\ell,i+1)$ has not done so yet. Since node i starts its clock—which in the worst case runs fast while the clock of node $i+1$ runs slow—at time $t_{\ell,i}^{(k+1)}$ again, a worst-case bound on the clock skew (modulo B) is obtained by comparing the clocks just before time $t_{\ell,i+1}^{(k+1)}$. However, this bound is subsumed by the previous one, as it covers the case $t = t_{\ell,i+1}^{(k+1)}$. Finally, the case $t \in [t_{\ell,i}^{(k)}, t_{\ell,i}^{(k+1)}] \cap [t_{\ell,i+1}^{(k+1)}, t_{\ell,i+1}^{(k+2)}]$ is symmetrical. This covers

 ISBN: 978-1-61208-301-8

all cases and therefore provides a worst-case bound for the intra-layer skew of the constructed fast clocks; replacing $\sigma_\ell^{\max}$ by σ_ℓ^{avg} in this result yields an average-case bound.

To also derive good bounds for the inter-layer skew, more care is necessary. Of course, we could simply replace $\sigma_\ell^{\max}$ by $\hat{\sigma}_\ell^{\max}$ and repeat the same analysis, but this would provide overly conservative results as it fails to leverage the known bias of the inter-layer skew. We will hence use a refined analysis for slightly modified clocks to improve the results.

On average, nodes at layer ℓ trigger roughly $\hat{\sigma}_\ell^{\text{avg}}$ after those at layer $\ell-1$ (recall that $\hat{\sigma}_\ell^{\text{avg}}$ respects the sign). Therefore, we can compensate the resulting bias in the inter-layer skew of the fast clocks by *shifting* the fast clocks of the nodes at layer $\ell > 1$ accordingly with respect to layer $\ell-1$. Similarly, to obtain a worst-case bound, we can consider $\hat{\sigma}_\ell^{\min}$ and $\hat{\sigma}_\ell^{\max}$ and apply a shift corresponding to $\bar{\sigma}_\ell := (\hat{\sigma}_\ell^{\max} + \hat{\sigma}_\ell^{\min})/2$.

Let us examine the latter case here; the former can be treated similarly. In $\bar{\sigma}_\ell$ time, a running fast clock makes progress of at least $f_{\min}\bar{\sigma}_\ell$. Therefore, we map the clock values $L_{\ell,i}(t)$ for all i and t to $L'_{\ell,i}(t) := (L_{\ell,i}(t) + f_{\min}\bar{\sigma}_\ell) \bmod B$. Analogous to the intra-layer skew, we now can bound

$$\begin{aligned} & L_{\ell-1,i}(t) - L'_{\ell,i}(t) \\ \leq\ & B - f_{\min}\left(\frac{B}{f_{\max}} - (t_{\ell,i}^{(k)} - t_{\ell-1,i}^{(k)}) + \bar{\sigma}_\ell\right) \\ \leq\ & \frac{f_{\max}-f_{\min}}{f_{\max}} \cdot B + f_{\min}\left(\hat{\sigma}_\ell^{\max} - \bar{\sigma}_\ell\right) \\ =\ & \varrho B + f_{\min} \cdot \frac{\hat{\sigma}_\ell^{\max} - \hat{\sigma}_\ell^{\min}}{2} = \varrho B + f_{\min}\tau_\ell, \quad (12) \end{aligned}$$

where $\tau_\ell := (\hat{\sigma}_\ell^{\max} - \hat{\sigma}_\ell^{\min})/2$ corresponds to $\sigma_\ell^{\max}$. The latter can be seen by noting that the the signed variant of the (symmetrical, i.e., absolute value) inter-layer skew is just $-\sigma_\ell$. Likewise,

$$\begin{aligned} & L'_{\ell,i}(t) - L_{\ell-1,i}(t) \\ \leq\ & B - f_{\min}\left(\frac{B}{f_{\max}} + (t_{\ell,i}^{(k)} - t_{\ell-1,i}^{(k)}) - \bar{\sigma}_\ell\right) \\ \leq\ & \frac{f_{\max}-f_{\min}}{f_{\max}} B + f_{\min}\left(\bar{\sigma}_\ell - \hat{\sigma}_\ell^{\min}\right) = \varrho B + f_{\min}\tau_\ell, \quad (13) \end{aligned}$$

which together with the bound on $L_{\ell-1,i}(t) - L'_{\ell,i}(t)$ implies that $|L'_{\ell,i}(t) - L_{\ell-1,i}(t)| \leq \varrho B + f_{\min}\tau_\ell$. An analogous reasoning shows that also $|L'_{\ell,i}(t) - L_{\ell-1,i+1}(t)| \leq \varrho B + f_{\min}\tau_\ell$.

Obviously, shifting all clocks in a layer by the same value will not affect the intra-layer clock skews. Applying the clock shifts inductively, i.e., shifting the clocks in each layer $\ell \geq 1$ by $\sum_{\ell'=0}^{\ell-1} \bar{\sigma}_{\ell'}$, we can bound the inter-layer skews on each pair of consecutive layers as above. We thus can conclude with the worst-case bounds

$$\varrho B + f_{\min}\sigma_\ell^{\max} \leq f_{\min}(\varrho\Gamma_{\min} + \sigma_\ell^{\max}) \text{ and} \quad (14)$$

$$\varrho B + f_{\min}\tau_\ell \leq f_{\min}(\varrho\Gamma_{\min} + \tau_\ell). \quad (15)$$

on the intra- and inter-layer skews of the (shifted) fast clocks. Note that they are tight for $B = f_{\min}\Gamma_{\min}$, which maximizes the amortized frequency of the fast clocks.

It can be seen that even if ϱ is fairly large (say, 10%) and $\Gamma_{\min}$ is significantly larger than $\sigma_\ell^{\max}$, say, up to factor 10, the maximal clock skew is still dominated by the term $f_{\min}\sigma_\ell^{\max}$. If one is willing to invest in more stable fast clock sources (e.g., $\varrho \approx 1\%$), large pulse separation times are feasible without incurring a significant impact on the skew. We stress that $f_{\min}$ and $f_{\max}$ in our bounds are—unless the clocks are extremely unstable—the amortized frequency upper and lower bounds over B fast clock pulses; large skews require a consistent difference in frequency for roughly $B/f_{\min}$ time. Moreover, ϱ captures the *relative drift* of the clocks. Any frequency change that applies to adjacent nodes in roughly the same way (i.e., a system-wide change in temperature or supply voltage) will not have noticeable effects on the skews.

D. Faults, Self-stabilization, and $\Gamma_{\min}$

So far, we neglected two key obstacles in our approach for constructing fast clocks:

- $\Gamma_{\min}$ is not known, and therefore we do not know an appropriate choice for B.
- Due to persistent or transient faults, $\Gamma_{\min}$ may be very small or not even well-defined.

Concerning the first issue, there are basically three options: We can rely on analytical bounds, simulation results, or experimental data. All approaches have pros and cons; for a final system, experimental data is the most valuable, but it is also the most difficult and expensive to create, as one needs a chip and a suitable apparatus for measurements first. From [17] and bounds for the layer 0 clock generation algorithm employed, analytical worst-case bounds can be derived. The results from Section V provide insight into the skew distributions for an average-case setting under some fairly conservative assumptions on the parameters.

Regarding the second issue, we clearly must exclude faulty nodes as well as triggering time differences from the self-stabilization period when estimating (an equivalent to) $\Gamma_{\min}$ in the general setting. Note that there is a trade-off: We can consider a non-faulty node that experiences large skews as incorrect, permitting the use of a less conservative bound on $\Gamma_{\min}$; in turn, we are losing the guarantee that this node will be able to complete each clock burst before the next pulse arrives, even after stabilization of the HEX pulse generation. The results from Section V give an idea on the behavior of the system under faults. It should be noted, though, that the specific values are highly dependent on the implementation of the HEX nodes, the layer 0 clock generation algorithm, and the used technology.

Once a suitable estimate $\tilde{\Gamma}_{\min}$ taking the role of $\Gamma_{\min}$ has been determined, one can choose B according to $B \leq f_{\min}\tilde{\Gamma}_{\min}$. Note that using a smaller value for B may be desirable in some situations, as a smaller value of B also leads to a smaller skew. On the downside, $B < f_{\min}\tilde{\Gamma}_{\min}$ is equivalent to using a smaller value of $\tilde{\Gamma}_{\min}$, which leads to a lower amortized frequency of the fast clocks. The resulting decrease in the amortized frequency can of course be mitigated by increasing the frequency of the layer 0 pulse generation algorithm, which reduces $\tilde{\Gamma}_{\min}$, Γ_{avg} and the gap $f_{\min}\tilde{\Gamma}_{\min} - B$.

VII. Conclusion and Future Work

In the work presented in this paper, we (i) implemented the HEX clock distribution grid [17] using the UMC 90 nm

standard cell library, (ii) performed extensive ModelSim-based simulations to gain insight on the average-case performance of the approach, and (iii) constructed small-skew high-frequency fast clocks and analyzed their performance. As argued in Section III, such clocks are a suitable basis for an efficient point-to-point communication subsystem. While it is still a long road to a fully-engineered solution, our results demonstrate the potential of the envisioned clocking scheme to serve in highly dependable, multi-synchronous GALS architectures.

Future Work. There are several vital improvements to the presented solutions that are subject to our ongoing research.

- The communication system proposed in Section III is not self-stabilizing, but merely *pseudo-stabilizing* [27]. That is, once transient faults cease, the system is guaranteed to eventually operate correctly, but there is no bound on the time the recovery process takes to actually complete. Based on the self-stabilizing fast clocks, it seems reasonable to assume that also fully self-stabilizing communication primitives can be devised.
- The skews offered by HEX compare unfavorably to those of less resilient solutions; in particular the gap to the performance of standard clock distribution trees (which cannot withstand any faults) is large. Because of the simplicity of the HEX state machine (Figure 4), we are positive that a HEX node can be implemented as a transistor-level custom standard cell. This would certainly decrease the resulting skews and pulse separation times further, and thus result in an improved HEX performance.
- The cylindrical hexagonal grid topology assumed in this work and [17] is convenient for analysis, but suffers from two drawbacks: (i) providing a synchronized clock to the entire layer 0 is difficult unless the grid is very narrow (i.e., W is small) and (ii) "folding" a cylindrical HEX grid onto an actual chip would result in two clock layers with physically close-by nodes that are far from each other in the grid. In [17], we proposed an alternative topology resolving these issues, and the analysis in the paper provides strong intuition that the resulting skews will not be larger. Generalizing the analysis from [17] to the new topology and performing corresponding simulations is thus of high interest.

REFERENCES

[1] D. M. Chapiro, "Globally-Asynchronous Locally-Synchronous Systems," Ph.D. dissertation, Stanford University, 1984.

[2] Y. Semiat and R. Ginosar, "Timing Measurements of Synchronization Circuits," in Proc. 9th Symposium on Asynchronous Circuits and Systems (ASYNC), 2003.

[3] P. Teehan, M. Greenstreet, and G. Lemieux, "A Survey and Taxonomy of GALS Design Styles," IEEE Design and Test of Computers, vol. 24, no. 5, 2007, pp. 418–428.

[4] D. G. Messerschmitt, "Synchronization in Digital System Design," IEEE Journal on Selected Areas in Communications, vol. 8, no. 8, 1990, pp. 1404–1419.

[5] A. Hansson, M. Subburaman, and K. G. W. Goossens, "Aelite: A Flit-Synchronous Network on Chip with Composable and Predictable Services," in Proceedings Design, Automation & Test in Europe Conference and Exhibition (DATE 2009), Nice, France. Los Alamitos: IEEE Computer Society, April 2009, pp. 250–255.

[6] T. Polzer, T. Handl, and A. Steininger, "A Metastability-Free Multi-Synchronous Communication Scheme for SoCs," in Proc. 11th Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), 2009, pp. 578–592.

[7] W. S. Coates and R. J. Drost, "Congestion and Starvation Detection in Ripple FIFOs," in Proceedings of the 9th International Symposium on Asynchronous Circuits and Systems, ser. ASYNC '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 36–45.

[8] M. S. Maza and M. L. Aranda, "Interconnected Rings and Oscillators as Gigahertz Clock Distribution Nets," in Proc. 13th Great Lakes Symposium on VLSI (GLSVLSI), 2003, pp. 41–44.

[9] S. Fairbanks, "Method and Apparatus for a Distributed Clock Generator," 2004, uS patent no. US2004108876 [retrieved: 06, 2013]. [Online]. Available: http://v3.espacenet.com/textdoc?DB=EPODOC&IDX=US2004108876

[10] S. Fairbanks and S. Moore, "Self-Timed Circuitry for Global Clocking," in Proc. 11th Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC), 2005, pp. 86–96.

[11] V. Gutnik and A. Chandrakasan, "Active GHz Clock Network Using Distributed PLLs," IEEE Journal of Solid-State Circuits, vol. 35, no. 11, 2000, pp. 1553–1560.

[12] A. Korniienko et al., "A Clock Network of Distributed ADPLLs Using an Asymmetric Comparison Strategy," in Proc. 2010 Symposium on Circuits and Systems (ISCAS), 2010, pp. 3212–3215.

[13] M. Saint-Laurent and M. Swaminathan, "A Multi-PLL Clock Distribution Architecture for Gigascale Integration," in Proc. 2001 IEEE Computer Society Workshop on VLSI (WVLSI), 2001, pp. 30–35.

[14] M. Függer, A. Dielacher, and U. Schmid, "How to Speed-Up Fault-Tolerant Clock Generation in VLSI Systems-on-Chip via Pipelining," in Proc. 8th European Dependable Computing Conference (EDCC), 2010, pp. 230–239.

[15] M. Függer and U. Schmid, "Reconciling Fault-Tolerant Distributed Computing and Systems-on-Chip," Distributed Computing, vol. 24, no. 6, 2012, pp. 323–355.

[16] D. Dolev, M. Függer, C. Lenzen, and U. Schmid, "Fault-Tolerant Algorithms for Tick-Generation in Asynchronous Logic: Robust Pulse Generation - [Extended Abstract]," in Proc. 13th Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), 2011, pp. 163–177.

[17] D. Dolev, M. Függer, C. Lenzen, M. Perner, and U. Schmid, "HEX: Scaling Honeycombs is Easier than Scaling Clock Trees," in Proc. 25th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA), 2013, in press.

[18] E. W. Dijkstra, "Self-Stabilizing Systems in Spite of Distributed Control," Communications of the ACM, vol. 17, no. 11, 1974, pp. 643–644.

[19] I. Miro Panades and A. Greiner, "Bi-Synchronous FIFO for Synchronous Circuit Communication Well Suited for Network-on-Chip in GALS Architectures," in Networks-on-Chip, 2007. NOCS 2007. First International Symposium on, May 2007, pp. 83–94.

[20] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, Logic Synthesis for Asynchronous Controllers and Interfaces. Springer, 2002.

[21] UMC 90 nm, [retrieved: 06, 2013]. [Online]. Available: http://www.umc.com/English/process/g.asp

[22] I. E. Sutherland, "Micropipelines," Communications of the ACM, Turing Award, vol. 32, no. 6, 1989, pp. 720–738.

[23] G. Fuchs and A. Steininger, "VLSI Implementation of a Distributed Algorithm for Fault-Tolerant Clock Generation," Journal of Electrical and Computer Engineering, vol. 2011, no. 936712, 2011.

[24] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," Commun. ACM, vol. 21, no. 7, 1978, pp. 558–565.

[25] T. Olsson and P. Nilsson, "Portable Digital Clock Generator for Digital Signal Processing Applications," Electronics Letters, vol. 39, no. 19, sept. 2003, pp. 1372 – 1374.

[26] T. Olsson, P. Nilsson, T. Meincke, A. Hemam, and M. Torkelson, "A Digitally Controlled Low-power Clock Multiplier for Globally Asynchronous Locally Synchronous Designs," in Proc. 2000 Symposium on Circuits and Systems (ISCAS), vol. 3, 2000, pp. 13–16.

[27] J. E. Burns, M. G. Gouda, and R. E. Miller, "Stabilization and Pseudo-Stabilization," Distributed Computing, vol. 7, no. 1, 1993, pp. 35–42.

 ISBN: 978-1-61208-301-8