HEX: Scaling Honeycombs is Easier than Scaling Clock Trees*

Danny Dolev
Hebrew University of Jerusalem
Edmond Safra Campus, 91904 Jerusalem, Israel
dolev@cs.huji.ac.il

Christoph Lenzen
Massachusetts Institute of Technology
32 Vassar Street, 02139 Cambridge, USA
clenzen@csail.mit.edu

Matthias Függer, Martin Perner, and Ulrich Schmid
Vienna University of Technology
Treitlstrasse 3, 1040 Vienna, Austria
{fuegger,mperner,s}@ecs.tuwien.ac.at

ABSTRACT

We argue that grid structures are a very promising alternative to the standard approach for distributing a clock signal throughout VLSI circuits and other hardware devices. Traditionally, this is accomplished by a delay-balanced clock tree, which distributes the signal supplied by a single clock source via carefully engineered and buffered signal paths.

Our approach, termed HEX, is based on a hexagonal grid with simple intermediate nodes, which both control the forwarding of clock ticks in the grid and supply them to nearby functional units. HEX is Byzantine fault-tolerant, in a way that scales with the grid size, self-stabilizing, and seamlessly integrates with multiple synchronized clock sources, as used in multi-synchronous Globally Synchronous Locally Asynchronous (GALS) architectures. Moreover, HEX guarantees a small clock skew between neighbors even for wire delays that are only moderately balanced. We provide both a theoretical analysis of the worst-case skew and simulation results that demonstrate very small typical skew in realistic runs.

Categories and Subject Descriptors

B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault-Tolerance; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

Keywords

fault-tolerant distributed algorithms; time distribution in grids; Byzantine fault-tolerance; self-stabilization

1. INTRODUCTION

Being able to distribute a synchronized clock signal to a large number of spatially distributed functional units is crucial for the synchronous design paradigm. In *Very Large Scale Integration* (VLSI) circuits and other hardware devices (as well as in master-slave-type network clock synchronization approaches like IEEE1588 [12]), this is accomplished by means of a *clock tree*, which distributes the clock signal supplied by a single clock source to all functional units attached as leaf nodes. Topologies that guarantee equal wire lengths from the root to the leaves, like H-trees (recursively constructed from a H-shaped wiring topology by attaching four smaller H-shapes to the four open ends), combined with carefully engineered wire geometries, clock signal regeneration buffers, etc. are used to ensure that clock pulses arrive at all functional units (that is, those making up a synchronous sub-system) simultaneously. This must be achieved with a *clock skew*, i.e., the maximum difference of the occurrence real-times of corresponding clock pulses at different functional units, well below half the clock cycle time: When a functional unit sends some data, say, on local clock count 1000, the receiver is expected to receive and process the data when pulse 1001 occurs according to its clock count.

Clock trees are attractive for several reasons. Besides conceptual simplicity, their height is only logarithmic in the number of the leaves (which is proportional to the die area), and the number of internal clock wires is linear in this number. As trees are planar graphs, it would (in principle) even be possible to route these links on a single interconnect layer.

These advantages come at a price, though: Spatially close functional units may be clocked via root-leaf paths that share few nodes, possibly only the root. The worst-case skew perceived between such functional units is the maximal difference of the signal propagation delay on these disjoint paths, which prohibits deep clock trees. Moreover, clock tree engineering must ensure that the maximum delay discrepancy remains below the acceptable clock skew, which is very

*This work has been supported by the Swiss National Science Foundation (SNSF), by the German Research Foundation (DFG, reference number Le 3107/1-1), by the Austrian Science Foundation (FWF) project FATAL (P21694), by The Israeli Centers of Research Excellence (I-CORE) program, (Center No. 4/11), by the ISG (Israeli Smart Grid) Consortium, administered by the office of the Chief Scientist of the Israeli Ministry of Industry and Trade and Labor, and by grant 3/9778 of the Israeli Ministry of Science and Technology. Danny Dolev is Incumbent of the Berthold Badler Chair.

SPAA'13, July 23–25 2013, Montréal, Québec, Canada.

difficult for clock speeds in the GHz range [1, 8, 19, 23, 27, 29]. In particular, given that a clock tree typically supplies a significant part of the chip (possibly even the entire die), the logarithmic height of a clock tree inevitably results in long wires. Making the resulting signal propagation delays as equal as possible, despite non-negligible wire resistances and coupling capacitances etc., is an engineering challenge that requires considerable efforts and costs. The resulting clock trees incorporate complex wire geometries and strong clock buffers, and thus suffer from large area and power consumption [15]. Moreover, further skew reduction typically requires extended clock tree topologies, such as trees with cross-links, meshes and multi-level trees [16, 29].

An even more serious issue with clock trees, which also arises in applications where there are no severe skew requirements, is lacking robustness. First of all, at the top level, a single clock source obviously constitutes a single point of failure. This is avoided by *Globally Asynchronous Locally Synchronous* (GALS) [2] architectures, where different parts of a chip are clocked by different clock sources & clock trees. However, using independent and hence unsynchronized clock domains gives away the advantages of global synchrony and thus requires non-synchronous cross-domain communication mechanisms or synchronizers [4, 13, 21]. Multi-synchronous clocking [25, 28] (also called *mesochronous* clocking [18]), which guarantees some upper bound on the skew between clock domains, has been invented to avoid this. The resulting multi-synchronous GALS architectures can rely on a common time base, which is attractive not only for applications programmers but also for metastability-free high-speed communication between different clock domains [20].

Still, the problem of limited robustness of clock trees persists even in GALS architectures: If just one internal wire or clock buffer in a clock tree breaks, e.g., due to some manufacturing defect or electromigration, *all* the functional units supplied via the affected subtree will stop working correctly. Therefore, it is desirable to have fairly small clock trees in a GALS system, necessitating a large number of synchronized clock domains. Overcoming the fundamental scalability and robustness issues of clock trees hence introduces the new challenge of robustly establishing a tight synchronization among a large number of clock domains.

Contribution: In this paper, we tackle this problem by proposing an alternative way for distributing a synchronized clock signal throughout an integrated circuit. Our approach, termed HEX, is based on a sufficiently connected wiring topology, namely, a *hexagonal grid*.[1] At each grid point, we place an (intermediate) node that controls when the clock pulses are forwarded to adjacent nodes and supplies the clock to nearby functional units, typically using a *small* local clock tree. It will turn out that HEX compares favorably to clock trees in most aspects.

In particular, with respect to robustness, our approach supports multiple synchronized clock sources and tolerates Byzantine failures of both clock sources and nodes. Its resilience to failures even scales with the size of the grid, in the sense that it supports a constant density of isolated Byzantine nodes, and it can handle a larger number of more benign failures like broken wires and mute clock sources and nodes. It is also self-stabilizing [3], in the sense that it can recover from an arbitrary number of transient failures.[2]

Furthermore, HEX has enticing properties with respect to the achievable skew between neighbors in the grid, which are typically the ones who need to communicate synchronously with each other. First of all, given that length and width of the grid grow as $\sqrt{n}$ and the density of HEX nodes should be roughly constant (with respect to n), wires are much shorter than in a clock tree. HEX hence neither requires strong clock buffers nor special wires, such that the maximal difference ε of the end-to-end delays between neighbors in the grid should be easily brought down to small values even by moderate engineering efforts. Second, for a proper embedding of the HEX topology, physically close nodes are also well-synchronized: In the fault-free case, HEX guarantees a worst-case skew between neighbors that is quadratic in ε. Depending on the number and severity of faults, this bound gracefully degrades. Moreover, it obeys a locality property: The adverse effect of faults on the neighbor skew decreases with the distance from the fault in the grid.

Related Work: Apart from the rich literature on clock tree engineering and extended topologies for skew reduction, see e.g. [1, 8, 15, 16, 19, 22, 23, 26, 29], we are not aware of much research on alternative clock distribution techniques. An exception is the work on distributed clock generation without local oscillators, which inherently also solves the problem of clock distribution. These approaches are essentially based on (distributed) ring oscillators, which are formed by gates arranged in a feedback loop. In [17], a regular structure of closed loops of an odd number of inverters is used for distributed clock generation. Similarly, [6, 7] employ local pulse generation cells, arranged in a two-dimensional grid, with each cell inverting its output signal when its four inputs (from the up, down, left and right neighbor) match the current clock output value. A more elaborate approach along the same lines uses an array of PLLs that are mutually synchronized among each other, using digital feedback exchanged across some (sparse) communication topology [11, 14, 24] like a grid. However, to the best of our knowledge, none of these approaches has been analyzed for its fault-tolerance properties, not to speak of self-stabilization.

The only fault-tolerant clock generation approaches for multi-synchronous GALS systems known to us are the Byzantine fault-tolerant DARTS approach [9, 10] and our self-stabilizing Byzantine fault-tolerant FATAL algorithm proposed in [5]. However, both approaches are complex and require a fully-connected interconnect topology. Consequently, they are not useful for distributing a synchronized clock to a large number of functional units, but are of course suitable candidates for the clock sources required by our HEX grid.

2. ALGORITHM & TOPOLOGY

We consider a set of nodes executing a pulse generation and forwarding algorithm, which communicate by message passing over a communication network whose underlying undirected communication graph is a cylindric hexagonal grid. Formally, the directed communication graph $G = (V, E)$ of our HEX grid is defined as follows (see Figure 1):

[1]Note that clock distribution by means of our HEX grid is fundamentally different from using a clock mesh [29] for averaging out large clock skews among near-by leaf nodes.

[2]Note, however, that our very simple algorithm is *not* self-stabilizing in the presence of ongoing Byzantine failures. There are non-stabilizing executions with Byzantine faults, although they seem unlikely to occur in practice.

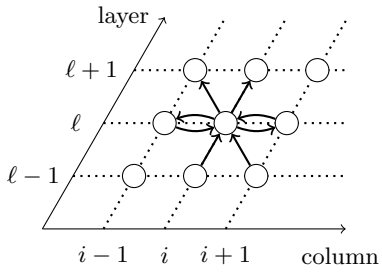

Figure 1: Node (ℓ, i) and its incident links in the cylindric hexagonal grid topology. Column coordinates are modulo W and layer coordinates (rows) are between 0 and L.

Letting $L \in \mathbb{N}$ denote 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 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: 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); if (ℓ, i) is in a layer greater than 0, then it has incoming links from $(\ell-1, i)$, called its lower left neighbor, and $(\ell-1, i+1 \bmod W)$, called its lower right neighbor; if (ℓ, i) is in a layer smaller than L, then it has outgoing links to $(\ell+1, i-1 \bmod W)$, its upper left neighbor, and $(\ell+1, i)$, its upper right neighbor. Figure 1 depicts the structure of the resulting HEX grid and shows a node's communication channels within the grid. The neighboring nodes of node (ℓ, i) form a hexagon, hence the name HEX grid. Due to the fact that column coordinates are modulo W, the HEX grid has a cylindric shape; we will briefly discuss the issue of embedding a HEX grid on a chip in Section 5.

Each node of the grid runs an algorithm that can broadcast *trigger messages* (representing clock pulses) over its outgoing links, as well as receive trigger messages over its incoming links. Each (fault-free) link guarantees an end-to-end communication delay (i.e., the time between sending and processing a trigger message) within $[d^-, d^+] \subset (0, \infty)$, where $\varepsilon := d^+ - d^- \leq d^+/2$. Each node further has access to a (possibly inaccurate) clock to measure timeouts.

Nodes at layer 0 are special in that they execute a pulse generation algorithm like the one of [5, 10], whose purpose is to generate synchronized and well-separated consecutive initial trigger messages. For each pulse number $k \in \mathbb{N}$, the time between any (non-faulty) node in layer 0 generating its k^{th} trigger message and another node in layer 0 generating its $(k+1)^{\text{th}}$ trigger message is sufficiently large. The precise meaning of "sufficiently large" depends on the desired fault-tolerance properties; we will elaborate on this in Section 3.2. Note that it is desirable to keep the maximal time between pulses small in order to guarantee a high operating frequency.

Nodes at layers larger than 0 run the simple pulse forwarding algorithm specified in Algorithm 1. Basically, nodes forward pulse k once they received trigger messages for pulse k from two adjacent neighbors. Since clock pulses and trigger messages are anonymous, i.e., can carry no information except their sole occurrence, care must be taken in order not to generate multiple trigger messages for a single pulse. The simple solution we use here relies on a sufficiently large separation between pulses, which relieves us from locally keeping track of pulse counts. Nodes can simply go to sleep for a while after forwarding a pulse and clear their history upon waking up again. In order to support practical (hence inaccurate) ways of implementing local timeouts, we allow the sleeping time to vary within the interval $[T^-, T^+]$. In the fault-free case, comparing the minimal and maximal possible speeds of pulse propagation shows that $T^- \in \mathcal{O}(\varepsilon L)$ is sufficient; with faults and in particular for self-stabilization, more care is required. The respective constraints on T^- and T^+ will be discussed in Section 3.2. Due to its simplicity, the forwarding algorithm can easily be implemented by means of an asynchronous state machine.

Algorithm 1: Pulse forwarding algorithm for nodes in layer $\ell > 0$.

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

While the purpose of the nodes in layer 0 is to generate pulses in a synchronized way, which typically necessitates full interconnection between all these nodes, the purpose of the nodes at higher layers is to propagate these pulses throughout the entire system via a low-connectivity network. Synchronized pulses generated by layer 0 propagate as "waves" through the HEX grid up to the very last layer. In the fault-free case, the propagation of every pulse can hence be analyzed independently.

3. SKEW & RESILIENCE

We are now going to analyze the skew and fault-tolerance properties of the algorithm and topology presented in the previous section. By $t_{\ell,i}^{(k)}$, we denote the *triggering time* of node (ℓ, i), i.e., the time when it forwards the k^{th} pulse. Generally, we will use superscript $^{(k)}$ to denote variables associated with the k^{th} pulse. When indexing nodes and trigger times, we will usually omit "mod W" to simplify the notation.

3.1 The Fault-free Case

As discussed earlier, in the fault-free case, we can restrict our attention to a single pulse and hence omit all pulse indices. We just assume that, initially, all nodes have cleared their memory and are waiting for the next pulse generated by the nodes in layer 0.

Definition 3.1 (Causal Links and Paths). *We say that a node is* left-triggered / centrally triggered / right-triggered*, if the satisfied guard from Algorithm 1 causing the node to trigger has received trigger messages from the* left and lower left / lower left and lower right / lower right and right neighbors*, respectively. In each case both of the respective links are* causal*. A* causal path *consists of causal links only.*

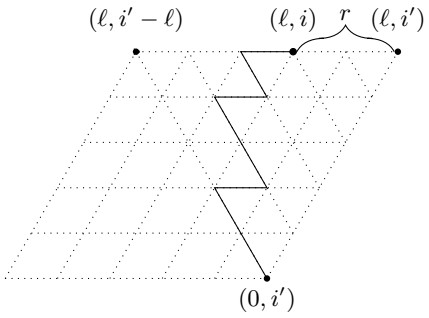

Figure 2: Illustration of the situation in Lemma 3.3.

Note that a link being causal implies that its endpoint is triggered at least d^- time after its origin. For instance, if (ℓ, i) is left-triggered, the links $((\ell, i-1), (\ell, i))$ and $((\ell-1, i), (\ell, i))$ are causal, while $((\ell, i+1), (\ell, i))$, $((\ell-1, i+1), (\ell, i))$ are not.

The following definition backtraces a sequence of causal links, either to the node in layer 0 starting the causal chain or to some specific column of interest.

Definition 3.2 (Left Zig-Zag Paths). *Given are a layer $0 < \ell \in [L+1]$ and column indices $i, i' \in [W]$, $i < i'$. The causal* left zig-zag path $p_{\text{left}}^{i' \to (\ell, i)}$ *is composed of rightward links $((\ell', j-1), (\ell', j))$ and up-left links $((\ell'-1, j+1), (\ell', j))$. It is inductively defined as follows. We start with the 0-length path $((\ell, i))$. Suppose that in some step of the construction the current path originates at node (ℓ', j) with $\ell' > 0$. If (ℓ', j) is left-triggered, we extend the path by adding the rightward link $((\ell', j-1), (\ell', j))$ as first link (and $(\ell', j-1)$ as its origin). Otherwise, the up-left link $((\ell'-1, j+1), (\ell', j))$ is causal and can be added as prefix to the path (and $(\ell'-1, j+1)$ as its origin). In the case of adding an up-left link the construction terminates if $\ell'-1 = 0$ or if both $j+1 = i'$ and the path now contains more up-left than rightward links.*

Since causal paths are acyclic, there must be fewer than W left-links before we go from ℓ' to $\ell'-1$, implying that the construction terminates after a finite number of steps. We now prove a useful technical lemma, which reveals a connection between the triggering times of two nodes at the same layer and left zig-zag paths, and thereby gives a bound on the maximum difference of their skew.

Lemma 3.3. *Suppose that path π is a prefix of some left zig-zag path $p_{\text{left}}^{i' \to (\ell'', i'')}$ and that π starts at node (ℓ', i') and ends at node (ℓ, i). Let r be the number of up-left links minus the number of rightward links along π. Then*

$$t_{\ell, i'} \leq t_{\ell, i} + r d^- + (\ell - \ell')\varepsilon.$$

Proof. From Definition 3.2, it follows that left zig-zag paths contain more up-left than rightward links or start at layer 0; hence, $\ell' < \ell$. W.l.o.g., we set $\ell' = 0$, i.e., we shift all layer indices by ℓ' and the new value of ℓ now represents $\ell - \ell'$. Moreover, since the path $p_{\text{left}}^{i' \to (\ell'', i'')}$ starts at node (ℓ', i'), we must have $r > 0$, since otherwise the construction of the left zig-zag path would have terminated before reaching (ℓ', i'), cf. Figure 2.

Consider the set S of nodes in the triangle with corners $(0, i')$, $(\ell, i'-\ell)$, and (ℓ, i') in Figure 2.[3] Observe that $p_{\text{left}}^{i' \to (\ell'', i'')}$ starts at the lower corner of the triangle and the prefix π never leaves it. By induction on the k^{th} diagonal of the triangle $(k, i'), \dots, (\ell, i'-(\ell-k))$ (for $k \in [\ell+1]$), we will prove that each node p that is both on the diagonal k and either on π or to the right of π is triggered at the latest at time $t_p \leq t_{\ell,i} - (\ell - r)d^- + kd^+$. Since this implies that $t_{\ell,i'} \leq t_{\ell,i} + rd^- + \ell\varepsilon$, this will establish the claim of the lemma.

To this end, let us show the induction hypothesis first for each node on π. Observe that node (ℓ, i) is on diagonal $(\ell - r)$. Hence, a node p that is h hops from (ℓ, i) on π must be on a diagonal $k \geq (\ell - r) - h$. Since $p_{\text{left}}^{i' \to (\ell'', i'')}$ is causal, it follows that $t_p \leq t_{\ell,i} - hd^- \leq t_{\ell,i} - (\ell - r)d^- + kd^+$, showing the statement for nodes on π.

Note that all nodes on diagonal 0 are either on or to the left of π, hence we already covered the induction anchor at $k = 0$. For the induction step from k to $k+1$, observe that any node will be left-triggered within at most d^+ time once both its left and lower-left neighbors are triggered. For any node p on the $(k+1)^{th}$ diagonal that is strictly right of π, its left and lower-left neighbor are on the diagonal k of S and either on π or to the right of π. The statement for diagonal k thus implies $t_p \leq t_{\ell,i} - (\ell - r)d^- + kd^+ + d^+$. Since we already covered nodes on π, the induction step succeeds. □

Based on the following definition, we will now develop bounds on the maximal skew in a given layer.

Definition 3.4 (Distance and Skew Potential). *For $i, j \in \mathbb{Z}$, let $d := i - j \bmod W$ and define $|i-j|_W := \min\{d, W-d\}$. For $\ell \in [L+1]$, the* skew potential on layer ℓ *is Δ_ℓ, where $\Delta_\ell := \max_{i,j \in [W]} \{t_{\ell,i} - t_{\ell,j} - |i-j|_W d^-\}$.*

We first prove a weak bound on the maximal skew at the upper layers that holds *independently* of the initial potential Δ_0. Notice that this result implies tolerance of HEX against arbitrary layer 0 skews.

Lemma 3.5. *For all $\ell \in \{W-2, \dots, L\}$ we have that $\Delta_\ell \leq 2(W-2)\varepsilon$.*

Proof. W.l.o.g. assume that $\ell = W-2$ and fix $i, i' \in [W]$, $i < i'$ (wrap-around cases are symmetrical). We distinguish two cases.

Case 1: $p_{\text{left}}^{i' \to (\ell, i)}$ starts at node (ℓ', i') for some $\ell' \in \{1, \dots, \ell-1\}$. Then, by Lemma 3.3,

$$t_{\ell,i'} \leq t_{\ell,i} + (i'-i)d^- + (\ell - \ell')\varepsilon \leq t_{\ell,i} + (i'-i)d^- + \ell\varepsilon.$$

Case 2: $p_{\text{left}}^{i' \to (\ell, i)}$ starts at node $(0, j)$, $j \in [W]$. Then the path has length at least $2\ell - (i'-i)$, since at least ℓ up-left and $\ell - (i'-i)$ right links are required for the path to originate at layer 0. Denote by t_0 the earliest time when a pair of two adjacent nodes in layer 0 are both triggered. Clearly, the second node on $p_{\text{left}}^{i' \to (\ell, i)}$ cannot be triggered before time $t_0 + d^-$ because it is in layer 1. Hence, $t_{\ell,i} \geq t_0 + (2\ell - (i'-i))d^-$ and thus

$$t_{\ell,i} \geq t_0 + (2(W-2) - (i'-i))d^-. \tag{1}$$

[3] Here we assume w.l.o.g. that $i' - \ell \geq 0$, i.e., no wrap-around within the triangle. The general case is treated analogously, ignoring that some of the index pairs may actually refer to the same node.

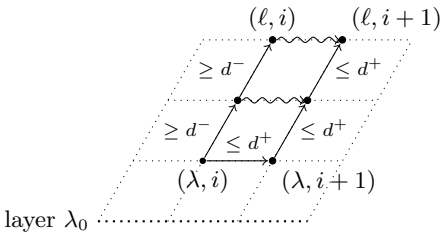

Figure 3: Illustration of Case 1 of Lemma 3.6.

Denote by $(0, j_0)$ a node with $\max\{t_{0,j_0}, t_{0,j_0+1}\} = t_0$; by definition of t_0 such a node exists. We claim that all nodes in layer $W-2$ are triggered no later than time $t_0+2(W-2)d^+$. This follows by induction on the layers $\lambda \in [W-1]$, where the hypothesis is that all nodes $(\lambda, j-\lambda), (\lambda, j-\lambda+1), \dots, (\lambda, j+1)$ are triggered until time $t_0 + 2\lambda d^+$. Since in layer λ these are $2+\lambda$ nodes, i.e., all W nodes in layer $W-2$, this will prove the claim. By the definition of t_0, the induction hypothesis holds for $\lambda = 0$. To perform the step from λ to $\lambda+1$, observe that all nodes $(\lambda+1, j-\lambda), (\lambda+1, j-\lambda+1), \dots, (\lambda+1, j)$ are triggered no later than time $t_0 + (2\lambda+1)d^+$, since by the hypothesis their lower left and lower right neighbors are triggered at least d^+ before that time. Until time $t_0 + 2(\lambda+1)d^+$, nodes $(\lambda+1, j-(\lambda+1))$ and $(\lambda+1, j+1)$ then must follow, completing the induction. In particular, this shows that $t_{\ell,i'} \leq t_0 + 2(W-2)d^+$ and hence, by using (1),

$$t_{\ell,i'} - t_{\ell,i} \leq (i'-i)d^- + 2(W-2)\varepsilon.$$

Overall, since i and $i' > i$ were arbitrary, from the two cases and symmetry for $i' < i$, we conclude that

$$\Delta_\ell = \max_{i,i' \in [W]} \{t_{\ell,i'} - t_{\ell,i} - |i'-i|_W d^-\} \leq 2(W-2)\varepsilon,$$

as claimed. □

Next, we derive more refined bounds on the skew between two neighboring nodes in the same layer $\ell > 0$. In contrast to Lemma 3.5, the next lemma takes the maximal skew in previous layers into account.

LEMMA 3.6. *For all $\ell_0 \in [L]$ and $\ell \in \{\ell_0+1, \dots, L\}$, it holds for each $i \in [W]$ that*

$$|t_{\ell,i} - t_{\ell,i+1}| \leq d^+ + \left\lceil \frac{(\ell-\ell_0)\varepsilon}{d^+} \right\rceil \varepsilon + \Delta_{\ell_0}.$$

PROOF. Fix some value of $\ell \geq 1$, set $i+1 = W-1$, and assume that $\ell_0 = 0$ and $t_{\ell,i} < t_{\ell,i+1}$ (all wrap-around cases are symmetrical). Define $\lambda_0 := \lfloor \ell d^-/d^+ \rfloor$, such that

$$\ell - \lambda_0 = \ell - \left\lfloor \frac{\ell d^-}{d^+} \right\rfloor = \left\lceil \frac{\ell\varepsilon}{d^+} \right\rceil. \tag{2}$$

We distinguish three cases.

Case 1: $t_{\lambda,i+1} \leq t_{\lambda,i} + d^+$ for some $\lambda \geq \lambda_0$ (Figure 3). W.l.o.g. let λ be maximal with this property. Hence, for all $\lambda' \in \{\lambda+1, \dots, \ell\}$, it follows that $t_{\lambda',i+1} > t_{\lambda',i} + d^+$. Notice that the choice of λ implies that for all such λ', node (λ', i) cannot be right-triggered, as the links $((\lambda', i+1), (\lambda', i))$ cannot be causal. Hence, all links $((\lambda'-1, i), (\lambda', i))$ must be causal. By induction on λ', we infer $t_{\ell,i} \geq t_{\lambda,i} + (\ell-\lambda)d^-$.

Furthermore, since the condition ensures that the trigger message from (λ', i) to $(\lambda', i+1)$ arrives well before time $t_{\lambda',i+1}$, node $(\lambda', i+1)$ will be triggered at the latest when the trigger message from its lower left neighbor $(\lambda'-1, i+1)$ arrives. Again by induction on λ', we infer that $t_{\ell,i+1} \leq t_{\lambda,i+1} + (\ell-\lambda)d^+$, and hence

$$t_{\ell,i+1} \leq t_{\lambda,i} + (\ell-\lambda+1)d^+. \tag{3}$$

By (2), thus $t_{\ell,i} - t_{\ell,i+1} \leq (\ell-\lambda)\varepsilon + d^+ \leq d^+ + \lceil \ell\varepsilon/d^+ \rceil \varepsilon$.

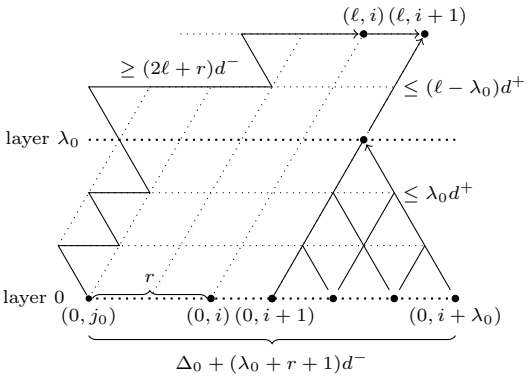

Figure 4: Illustration of Case 2 of Lemma 3.6.

Case 2: Case 1 does not apply and $p_{\text{left}}^{i+1\to(\ell,i)}$ starts at some node $(0, j_0)$, for $j_0 \neq i+1$ (Figure 4). If $p_{\text{left}}^{i+1\to(\ell,i)}$ contained more left-up links than rightward links, it would contain a subpath originating at a node in column $i+1$ that also would have more left-up than rightward links. This is not possible, since then the construction would have terminated at this node, either resulting in the path originating at a layer $\ell' > 0$ or at node $(0, i+1)$. Hence $p_{\text{left}}^{i+1\to(\ell,i)}$ is of length $2\ell + r$ for some $r \geq 0$ and $j_0 = i - r \bmod W$.

For all indices $j \in \{i+1, i+2, \dots, i+1+\lambda_0\}$ we have that $|j-j_0|_W \leq j - i + r$. We obtain that

$$\begin{aligned} t_{\ell,i} &\geq t_{0,j_0} + (2\ell+r)d^- \\ &= t_{0,j} - (t_{0,j} - t_{0,j_0} - |j-j_0|_W d^-) \\ &\quad -|j-j_0|_W d^- + (2\ell+r)d^- \\ &\geq t_{0,j} - \Delta_0 - (j-i)d^- + 2\ell d^- \\ &\geq t_{0,j} - \Delta_0 + (2\ell - \lambda_0 - 1)d^-. \end{aligned}$$

Moreover, by induction on $\lambda \in \{0, \dots, \lambda_0\}$, it follows that all nodes $(\lambda, i+1), \dots, (\lambda, (i+1+\lambda_0-\lambda) \bmod W)$ are triggered by time

$$\max_{j \in [i+\lambda_0+2] \setminus [i+1]} \{t_{0,j}\} + \lambda d^+ \leq t_{\ell,i} + \Delta_0 - (2\ell - \lambda_0 - 1)d^- + \lambda d^+.$$

In particular, by the definition of λ_0, $t_{\lambda_0,i+1} \leq t_{\ell,i} + \Delta_0 - (\ell - \lambda_0 - 1)d^-$. Since Case 1 does not hold, we can use similar arguments as for deriving (3) to obtain that $t_{\ell,i+1} \leq t_{\lambda_0,i+1} + (\ell-\lambda_0)d^+$. It follows that

$$t_{\ell,i+1} \leq t_{\ell,i} + d^- + (\ell-\lambda_0)\varepsilon + \Delta_0 = t_{\ell,i} + d^- + \left\lceil \frac{\ell\varepsilon}{d^+} \right\rceil \varepsilon + \Delta_0.$$

Case 3: Neither Case 1 nor Case 2 apply (Figure 5). In other words, $p_{\text{left}}^{i+1\to(\ell,i)}$ starts at node $(\ell', i+1)$ for some $\ell' < \ell$, and $t_{\lambda,i+1} > t_{\lambda,i} + d^+$ for all $\lambda \in \{\lambda_0, \dots, \ell\}$. Note that by construction the first link of $p_{\text{left}}^{i+1\to(\ell,i)}$ is $((\ell', i+1), (\ell'+1, i))$. Hence, we have that $\ell' < \lambda_0 - 1$, as otherwise node $(\ell'+1, i+1)$ would be triggered no later than time

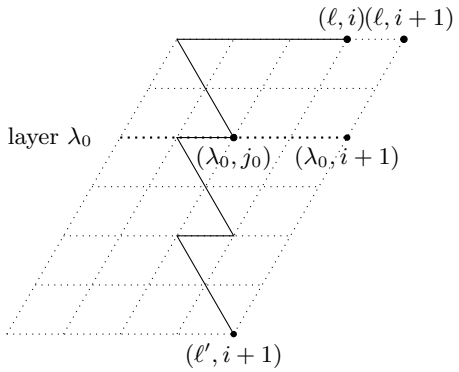

Figure 5: Illustration of Case 3 of Lemma 3.6.

$\max\{t_{\ell',i+1} + d^+, t_{\ell'+1,i} + d^+\} = t_{\ell'+1,i} + d^+$, contradicting the fact that Case 1 does not apply.

Let (λ_0, j_0) be the last node on the causal path $p_{\text{left}}^{i+1\to(\ell,i)}$ in layer λ_0. Observe that $j_0 + r - l = i$, where r (resp. l) is the number of rightward (resp. up-left) hops of $p_{\text{left}}^{i+1\to(\ell,i)}$ after (λ_0, j_0). The prefix of $p_{\text{left}}^{i+1\to(\ell,i)}$ ending at (λ_0, j_0) satisfies the precondition of Lemma 3.3, yielding $t_{\lambda_0,i+1} \le t_{\lambda_0,j_0} + (i+1-j_0)d^- + (\lambda_0 - \ell')\varepsilon$. Recall that since Case 1 does not apply, we have by (3) that $t_{\ell,i+1} \le t_{\lambda_0,i+1} + (\ell - \lambda_0)d^+$. Using that $d^+ = d^- + \varepsilon$, we obtain

$$t_{\ell,i+1} \le t_{\lambda_0,j_0} + (\ell - \lambda_0 + i + 1 - j_0)d^- + (\ell - \ell')\varepsilon.$$

By construction, $p_{\text{left}}^{i+1\to(\ell,i)}$ is of length $2(\ell - \ell') - 1$ and its prefix ending at node (λ_0, j_0) is of length $2(\lambda_0 - \ell') - (i+1-j_0)$. Therefore, the length of the suffix of $p_{\text{left}}^{i+1\to(\ell,i)}$ starting at (λ_0, j_0) is $2(\ell - \lambda_0) + (i - j_0)$. Because this suffix is a causal path, we may infer that

$$t_{\ell,i} \ge t_{\lambda_0,j_0} + (2(\ell - \lambda_0) + (i - j_0))d^-.$$

Altogether, we arrive at

$$\begin{aligned} t_{\ell,i+1} - t_{\ell,i} &\le (\ell - \ell')\varepsilon - (\ell - \lambda_0 - 1)d^- \\ &\le \ell\varepsilon - \left(\frac{\ell\varepsilon}{d^+} - 1\right) d^- = d^- + \frac{\ell\varepsilon^2}{d^+}. \end{aligned}$$

Since these cases are exhaustive and for each the claimed bound holds, this concludes the proof. □

Finally, by some slightly modified reasoning, it is also possible to derive a skew bound in W.

Corollary 3.7. *Set $\delta := d^-/2 - \varepsilon$. For each layer $\ell \in \{W, \dots, L\}$ and all $i \in [W]$ it holds that*

$$|t_{\ell,i} - t_{\ell,i+1}| \le \max\left\{d^+ + \left\lceil \frac{W\varepsilon}{d^+} \right\rceil \varepsilon, \Delta_{\ell-W} + d^+ - W\delta\right\}.$$

Proof Sketch. The proof is analogous to the one of Lemma 3.6, where Case 2 is treated slightly differently. Assuming w.l.o.g. that $\ell = W$, in Case 2 we bound for all indices $j \in \{i+1, i+2, \dots, i+\lambda_0+1\}$

$$t_{\ell,i} \ge t_{0,j} - \Delta_0 - |j - j_0|_{\mathrm{W}} d^- + 2\ell d^- \ge t_{0,j} - \Delta_0 + \frac{3\ell d^-}{2},$$

where in the second step we exploit that $|j - j_0|_{\mathrm{W}}$ is upper bounded by $W/2 = \ell/2$. Proceeding as in the proof of Lemma 3.6, the claimed bound follows. □

We are now ready to derive our main result, namely, bounds on the worst-case skews between neighbors.

Theorem 3.8 (Skew Bounds—Fault-free Case). *Suppose that $\varepsilon \le d^+/7$. 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,*

$$\forall \ell \in \{1, \dots, 2W-3\} : \sigma_\ell \le d^+ + 2W\varepsilon^2/d^+ + \Delta_0.$$
$$\forall \ell \in \{2W-2, \dots, L\} : \sigma_\ell \le d^+ + \lceil W\varepsilon/d^+\rceil\varepsilon.$$

Moreover, regarding the inter-layer skew of layer $\ell \in [L]$, for all $i \in [W]$ that

$$\begin{aligned} t_{\ell,i} - \sigma_\ell + d^- &\le t_{\ell+1,i} \le t_{\ell,i} + \sigma_\ell + d^+ \text{ and} \\ t_{\ell,i+1} - \sigma_\ell + d^- &\le t_{\ell+1,i} \le t_{\ell,i+1} + \sigma_\ell + d^+. \end{aligned}$$

Proof. Assume first that $\Delta_0 = 0$. For the sake of the argument, imagine that the HEX grid would start at layer $-(W-1)$, where for all $i \in [W]$ and all $\ell \in \{-(W-1), \dots, 0\}$ we would have that $t_{\ell,i} = \ell d^+$. Clearly, starting from any execution on the actual grid, this would result in a feasible execution on the extended grid if we choose all link delays on the imagined links to be d^+. It follows that $\Delta_\ell = 0$ for all $\ell \in \{-(W-1), \dots, 0\}$. From Lemma 3.5, we obtain that $\Delta_\ell < (2W-1)\varepsilon$ for all $\ell \in \{1, \dots, L\}$ (since we have negative layer indices until $-(W-1)$, the lemma also applies to layers $1, \dots, W-3$). Now we apply Corollary 3.7 to all layers $\{1, \dots, L\}$, yielding that

$$\sigma_\ell \le \max\left\{d^+ + \left\lceil \frac{W\varepsilon}{d^+} \right\rceil \varepsilon, W(2\varepsilon - \delta) + d^-\right\}.$$

Since $\varepsilon \le d^+/7$, we have that $2\varepsilon - \delta \le 0$, showing the first statement.

Now consider the case where Δ_0 is arbitrary. The bound on σ_ℓ for $\ell \in \{1, \dots, 2W-3\}$ follows from Lemma 3.6. For $\ell \ge 2W-2$, observe first that we can apply Lemma 3.5 to all layers $\ell \in \{W-2, \dots, L\}$. Hence the same bound as in the previous case holds due to Corollary 3.7 applied to layers $\ell \in \{2W-2, \dots, L\}$.

The third inequality of the theorem holds since

$$\begin{aligned} t_{\ell,i} - \sigma_\ell + d^- &\le \min\{t_{\ell,i}, t_{\ell,i+1}\} + d^- \\ &\le t_{\ell+1,i} \\ &\le \max\{t_{\ell,i}, t_{\ell,i+1}\} + d^+ \\ &\le t_{\ell,i} + \sigma_\ell + d^+; \end{aligned}$$

the final inequality is proved analogously. □

We remark that it is fairly straightfoward to construct worst-case executions that almost match these bounds (cf. Figure 6), i.e., they are essentially tight.

3.2 Fault-Tolerance Properties

We will now shed light on the fault-tolerance properties of our approach. We will consider transient and permanent faults, and for permanent faults we distinguish between *crash faults*, where a node simply ceases to operate, and *Byzantine faults*, where a node behaves arbitrarily. For the sake of simplicity, we will not discuss link failures, which can be understood by essentially the same arguments and insights we present for node failures.

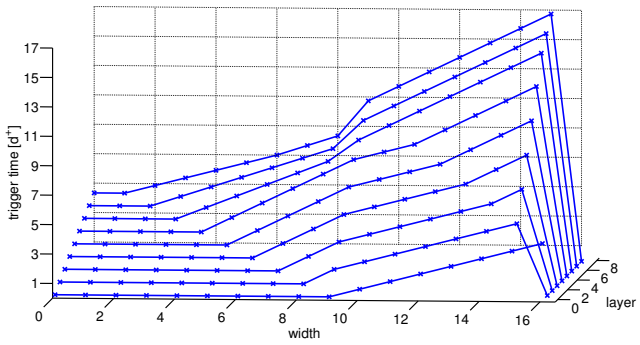

Figure 6: A worst-case pulse wave, with maximal intra-layer skew between the top-layer nodes in columns 8 and 9. Nodes in and left of column 8 are left-triggered (except for the "flat" region) with minimal delays of d^-. Nodes in and right of column 9 are slow due to large delays of d^+ and large initial skews in the respective region of layer 0. To focus on the essential central part of the grid, we introduced a barrier of "dead" nodes in column 16.

Due to the large number of possible fault patterns, we cannot hope for a comprehensive and exhaustive formal analysis of all cases within the scope of this paper. Thus, we will mostly confine ourselves to a qualitative discussion of the effects of faults; in Section 4, we will elaborate on some concrete scenarios by means of simulations.

Byzantine Faults. Since the communication structure of our algorithm is extremely simple, it is not difficult to understand the "options" of Byzantine nodes for disrupting the system's operation. If they have the possibility to generate "false" pulses, i.e., trigger non-faulty nodes without the support of other non-faulty nodes, this will clearly break our protocol: Once this happens, this will cause a chain reaction distributing the false pulse just like a correct one.

A similar problem arises if a node has a second faulty neighbor (even if it is just a crash fault) and the two faults are not just the left and right neighbor. If both faulty neighbors are not sending trigger messages, the node is not going to be triggered. However, if a Byzantine neighbor *does* send a trigger message, the node can be triggered. Hence, a Byzantine node can essentially trigger the node at an arbitrary time between pulses, which again enables it to create a false pulse. We therefore require the following.

Condition 3.9 (Fault Separation). *No node can be triggered by Byzantine nodes only. (This is equivalent to no two neighbors being Byzantine unless they are both in layer L.) Every node with a Byzantine neighbor can be triggered by a pair of non-faulty nodes.*

With Condition 3.9 in place, the power of a Byzantine node is reduced to (i) locally accelerating the progress of a pulse by sending messages prematurely, (ii) delaying or halting the progress of a pulse by sending a message late or not at all, or (iii) asymmetrically combining (i) and (ii) by speeding up the progress of a pulse on one side and delaying it on the other side. For a single faulty node, the respective effects are always limited to increasing skews by at most a few d^+, since we can "work around" the defective node in our reasoning. In particular, when inductively constructing a zig-zag path (which will imply a skew bound), we can switch to the other causal link of the current node if we run into a Byzantine node. Note that such a node could increase the skew caused along the original (shortest) zig-zag path, since it may trigger the next node along the path (and hence the fast node among the two suffering from the worst-case skew) d^- time earlier than a correct node. Similarly, since a Byzantine node could impair our reasoning why the slow node is triggered in a timely fashion, we can just avoid reasoning about the faulty node's behavior and make a short "detour". In both cases, the alternative paths considered have lengths almost identical to the ones constructed for the fault-free case, which results in a small increase of the corresponding skew bounds only.

Crash Faults and Fault Containment. For multiple faults, such arguments do not apply in all cases. For example, if every other node in a layer is crash-faulty, Condition 3.9 is not violated. Yet, no node in the layers above is triggered at all.

It is easy to see, however, that such problematic fault patterns entail faults in close vicinity to each other. If it is not the case that an entire layer "blocks" pulses and sufficiently many layers above it are fault-free the caused damage to the synchronization quality will "heal". An extreme case of this kind of resilience is illustrated by Theorem 3.8, which shows that within $2W-2$ layers the strong skew bounds of the fault-free case are established for *arbitrary* initial skews, provided that there are sufficiently many consecutive fault-free layers (and the pulse arrives at the lowest of them at all). Less severe problems will cause less hassle (cf. Section 4).

Pulse Separation. As discussed in Section 2, we require that pulses arrive at each node sufficiently well-separated so that the nodes do not need to keep track of the pulse number and can rely on the proposed simple sleeping mechanism instead. This amounts to ensuring that, for each node, the sleep period of at most T^+ after triggering pulse k is over before any signal of pulse $k+1$ arrives, i.e.,

$$t_{\ell,i}^{(k)} + T^+ \leq \max\{t_{\ell,i-1}^{(k+1)}, t_{\ell-1,i}^{(k+1)}, t_{\ell-1,i+1}^{(k+1)}, t_{\ell,i+1}^{(k+1)}\} + d^-.$$

In the fault-free case, the following condition is sufficient:

Condition 3.10 (Fault-free Separation Time). *For all $k \in \mathbb{N}$, we require that*

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

If there are no faults, $t_{\ell,i}^{(k)} \leq \max_{i\in[W]}\{t_{0,i}^{(k)}\} + \ell d^+$ and any node in layer $\ell-1$ or ℓ will not trigger pulse $k+1$ before time $\min_{i\in[W]}\{t_{0,i}^{(k+1)}\} + (\ell-1)d^-$. Since $\ell \leq L$, we see that Condition 3.10 is sufficient in the fault-free case.

In the presence of faults, a conservative bound would argue that the longest feasible time for a pulse to reach all nodes is bounded by WLd^+,[4] however, such a worst-case bound will be of little significance in practice. For the system to be operational, the number of faults must be sufficiently small to maintain reasonable skews. Moreover, faults significantly affecting the time to complete pulses, yet at the same time not causing large skews, appear extremely unlikely. Hence adding a slack of a few d^+ to Condition 3.10

[4]This follows since acceptable fault patterns disallow that non-faulty nodes are triggered by faulty ones, implying that the pulse must be complete if for more than d^+ time no node is triggered.

should suffice in all relevant cases. This view is supported by our simulations, see Section 4.

Self-stabilization. *Self-stabilization* is the ability of the system to recover from an unbounded number of arbitrary transient faults [3]. That is, once transient faults cease, the system will resume normal operation within a bounded *stabilization time*. This is equivalent to demanding that a system where the individual components behave according to their specification will resume normal operation from an arbitrary initial state, since arbitrary faults may result in arbitrary states. Recall that the pulse generation at layer 0 is outside the scope of this paper; to make the system as a whole self-stabilizing, clearly a self-stabilizing algorithm is to be employed for that purpose.

THEOREM 3.11. *Suppose* $\max_{k\in\mathbb{N}}\{\Delta_0^{(k)}\} \le \Delta$ *and denote* $\sigma_0 := \Delta + d^-$. *Assume that*[5]

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

that $T^- > \sigma_\ell + d^+ + \varepsilon + t_{pulse}$ *for all* $\ell \in [L+1]$, *where* σ_ℓ *is as in Theorem 3.8 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 3.8 apply.*

PROOF. For simplicity, we denote the first correctly generated pulse as pulse 0. Hence the induction hypothesis is that layer ℓ executes pulses $k \ge \ell$ correctly. Denote for $k \in \mathbb{N}_0$ by $t_-^{(k)}$ and $t_+^{(k)}$ the times when the first and last nodes in layer 0 trigger their k^{th} pulse, respectively. To perform the induction step from $\ell \in \mathbb{N}_0$ to $\ell+1$, observe that the induction hypothesis applied to layer ℓ implies that, for all $k \ge \ell$, no nodes in layer ℓ are triggered during

$$\left[t_+^{(k)} + \ell d^+, t_-^{(k+1)} + \ell d^-\right].$$

Consequently, no trigger messages from layer ℓ are received by nodes in layer $\ell+1$ during

$$\left[t_+^{(k)} + (\ell+1)d^+, t_-^{(k+1)} + (\ell+1)d^-\right].$$

Hence no node in layer $\ell+1$ can be triggered twice during this period. If for d^+ additional time no node in layer $\ell+1$ is triggered within this interval, this implies that all messages have been received and therefore no further nodes may trigger. Overall, it follows that no node on layer $\ell+1$ may be triggered during the interval

$$[t_+^{(k)} + (\ell+1)d^+ + Wd^+, t_-^{(k+1)} + (\ell+1)d^-],$$

implying that none of these nodes is sleeping during

$$[t_+^{(k)} + (\ell+1)d^+ + Wd^+ + T^+, t_-^{(k+1)} + (\ell+1)d^-].$$

By the prerequisites of the theorem, this time interval is non-empty. Hence, all nodes in layer $\ell+1$ will be ready to participate in pulse $k+1$.

It remains to show that each node will trigger exactly once per pulse and that the skew bounds hold. To see this, we show by induction on the layers $\ell \in \{1,\dots,L\}$ that (i) for pulses $k > \ell$, no node in layer $1,\dots,\ell$ will trigger due to a "falsely" memorized trigger message from a neighbor from which the trigger message for pulse k has not been received yet and (ii) that the skew bounds from Theorem 3.8 apply to layers $1,\dots,\ell$ for pulses $k > \ell$. Note that (ii) follows from (i) and the fact that we just showed that no node on layer $\ell' \in \{1,\dots,\ell\}$ is sleeping when pulse k arrives, since this implies that the assumptions on the behaviour of the nodes made in Section 3.1 hold and Theorem 3.8 applies.

To anchor the induction at $\ell = 0$, we just observe that the skew bounds are satisfied with respect to σ_0, as we will not require more than that for the induction step. For the step from ℓ to $\ell+1 \in \mathbb{N}$, recall that we already know that all nodes on layer $\ell+1$ will be triggered at least once for each pulse $k \ge \ell+1$. Hence, for node $(\ell+1,i)$, $i \in \{1,\dots,W\}$, we can define $t_{\ell+1,i}^{(k)}$ as the first such triggering time (uniqueness will follow later). For each i, one of the links $((\ell,i),(\ell+1,i))$ or $((\ell,i+1),(\ell+1,i))$ is causal. W.l.o.g. let it be $((\ell,i),(\ell+1,i))$ (the other case is symmetrical). Then $t_{\ell+1,i}^{(k)} \ge t_{\ell,i}^{(k)} + d^-$. It follows that $t_{\ell,i+1}^{(k)} \le t_{\ell,i}^{(k)} + \sigma_\ell \le t_{\ell+1,i}^{(k)} + \sigma_\ell - d^-$. Hence, since nodes $(\ell+1,i-1)$ and $(\ell+1,i+1)$ follow at most d^+ after $t_{\ell,i+1}^{(k)}$, we observe

$$\max\left\{t_{\ell+1,i-1}^{(k)}, t_{\ell,i}^{(k)}, t_{\ell,i+1}^{(k)}, t_{\ell+1,i+1}^{(k)}\right\} \le t_{\ell+1,i}^{(k)} + \sigma_\ell + \varepsilon,$$

implying that node $(\ell+1,i)$ will not receive any message from pulses $\le k$ after time $t_{\ell+1,i}^{(k)} + \sigma_\ell + d^+ + \varepsilon \le t_{\ell+1,i}^{(k)} + T^-$, showing (i) for $\ell+1$. As (ii) follows by Theorem 3.8, this completes the induction and also the proof. □

Note that the same reasoning applies in the presence of crashed nodes, provided the skew bounds (and thus also T^-) are adapted accordingly. For Byzantine faults, stabilization is more involved and requires to modify the algorithm: If a node has e.g. a Byzantine lower-left neighbor and misses the trigger signal of pulse k from its right neighbor due to sleeping, but just wakes up before the signal from its lower right neighbor arrives, the Byzantine neighbor can trigger the node at any later point in time. In particular, it can send the node to sleep such that the same situation occurs in pulses $k+1, k+2, \dots$ Slightly more complex triggering rules that take into account the timing of the received signals may circumvent this issue.

It should also be noted that the required pulse separation time of Theorem 3.11 can be reduced, since a more involved analysis can be used to prove stabilization without a linear additive term of Wd^+. The detailed exploration of these issues, which determine the maximum sustainable clock frequency, is an important part of our future work.

4. SIMULATION EXPERIMENTS

In order to get an idea of the behavior of HEX in realistic scenarios, we developed a framework for simulation experiments. Rather than simulating the simple VHDL implementation[6] of Algorithm 1 in Modelsim directly, we used Matlab due to its greater flexibility and controllability.

[5] We assume that messages are zero-length signal pulses here.

[6] Algorithm 1 just consists of a 3-state asynchronous state machine in conjunction with memory flags (flip-flops) for memorizing the reception of trigger signals from neighbors. State *fire*, entered when the node is triggered, signals a clock pulse to its neighbors and is immediately left for state *sleep*, where the clock pulse is reverted and the sleep timeout is started. When it expires, the state machine clears all memory flags and enters the state *wait*, where it remains until one of the triggering conditions becomes true.

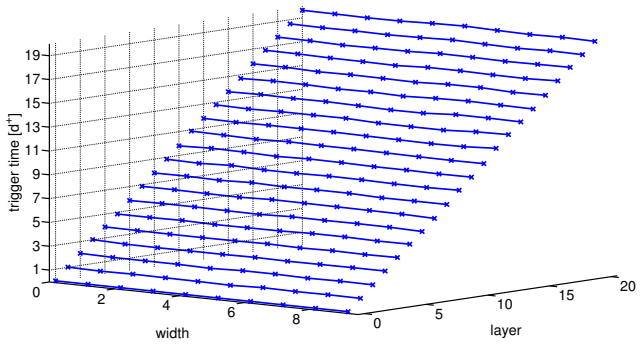

Figure 7: Pulse wave propagation for uniformly chosen link delays in $[50, 55]$ and $\sigma_0 = \Delta_0 = 0$.

To complement the analytic results provided in the previous section, this section provides a glimpse of our simulation results. Their primary purpose is the following:

(1) *Demonstrating small typical skew in the fault-free case.* The quite fancy scenarios required for establishing the worst-case skews in Theorem 3.8 suggest that they are very unlikely to occur in practice. Our simulation studies support this view.

(2) *Demonstrating fault locality.* As argued in Section 3.2, HEX implicitly confines the effects of faulty nodes. Our simulation results show that this is indeed true, even for clustered crash faults and multiple (but separated) Byzantine nodes.

For our simulations, we developed a Matlab implementation of Algorithm 1 that can be plugged into arbitrarily sized grids with configurable topologies. We implemented different choices for the individual link delays, including uniformly at random from $[d^-, d^+]$ and fixed values. Different choices for the triggering times of the nodes in layer 0 are also provided.

The primary quantities of interest observed in our simulation runs are the intra-layer neighbor skews $\max\{|t_{\ell,i} - t_{\ell,i+1}|, |t_{\ell,i} - t_{\ell,i-1}|\}$ of every node (ℓ, i) in layer ℓ, as well as the inter-layer neighbor skews, i.e., $t_{\ell+1,i-1} - t_{\ell,i}$ and $t_{\ell+1,i} - t_{\ell,i}$ of every node (ℓ, i) relative to its direct layer $\ell+1$ neighbors $(\ell+1, i-1)$ and $(\ell+1, i)$, respectively. Note 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 of the difference in trigger times. This is of interest, since the (non-zero) *expected* clock skew between adjacent layers can be compensated at the level of the (final) local clocks (to be synchronized by means of the HEX pulses), thereby providing clocks that are also well-synchronized between different layers.

We compute the skews from the matrix of all triggering times $t_{\ell,i}$ obtained in a simulation run; this data also allows us to visualize the detailed propagation of a pulse throughout the grid.

The Fault-free Case. We conducted a suite of simulations that complement the analytic intra- and inter-layer worst-case skew bounds given in Theorem 3.8. As an appetizer, Figure 7 shows a 3D plot of a typical pulse propagation wave in a grid with $W = 10$ and $L = 20$. The entire 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 $t_{\ell,w}$ of the corresponding node (ℓ, i). To improve the readability of intra-layer skews, we connected all points $(\ell, i, t_{\ell,w})$ and $(\ell, i + 1, t_{\ell,i+1})$, $i \in [W - 1]$. It is apparent that the wave propagates evenly throughout the grid, nicely smoothing out differences in link delays. We note, though, that by choosing the delays accordingly, we were able to produce pulse propagation waves exhibiting the worst-case skew of Lemma 3.6 (see Figure 6).

	intra-layer		inter-layer		
init. layer 0	avg	max	min	avg	max
0	2.19	13	50	53.95	67
rand. $[0, d^-]$	3.31	44	50	54.84	103
rand. $[0, d^+]$	3.42	49	50	54.94	108
ramp d^+	12.99	55	34	60.08	110

Table 1: Intra- and inter-layer skews of all nodes in the grid from 100 simulation runs, for uniformly random link delays in $[50, 55]$.

	intra-layer		inter-layer		
init. layer 0	avg	max	min	avg	max
0	9.66	56	50	68.89	124
rand. $[0, d^-]$	10.37	62	50	69.43	131
rand. $[0, d^+]$	10.94	66	46	69.86	146
ramp d^+	23.4	79	-15	74.51	150

Table 2: Intra- and inter-layer skews of all nodes in the grid from 100 simulation runs, for uniformly random link delays in $[50, 75]$.

Table 1 and Table 2 show, for four different choices of the layer 0 skews between neighbors, the average and maximal intra-layer skews and the minimal, average, and maximal inter-layer skews, respectively. These values where computed over all nodes and 100 simulation runs.

Table 1 shows the skews for link delays chosen uniformly at random from $[50, 55]$ (such that $d^+ = 1.1d^-$ and $\varepsilon/d^+ = 1/11$) in all settings. 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 $0 \leq i < W/2$ and $t_{0,i+1} = t_{0,i} - d^+$ for $W/2 \leq i < W - 1$ (i.e., $\sigma_0 = d^+$ and $\Delta_0 = W\varepsilon/2 = 25$). 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.

Figure 8 shows the histogram of the skew distribution of (i). For (ii) and (iii) the distributions look very similar, whereas for the extreme case of (iv) the large initial skews result in noticeably worse bounds for the first few layers. Due to lack of space, we summarize the results in Table 2.

Our simulation results show that the typical skews are considerably smaller than our analytic worst-case bounds: For example, plugging the above parameters and $\Delta_0 = 0$ into Theorem 3.8 yields an intra-layer skew bound of $d^+ + \lceil W\varepsilon/d^+ \rceil \varepsilon \approx 80$ and an inter-layer skew in the range of $\approx [-30, 135]$. By contrast, even for setting (iii) with $\Delta_0 \approx \varepsilon$, the average of the observed intra-layer skew is below 4, with a maximum of 49, and the range of inter-layer skews is $[50, 108]$. It should be particularly noted that for settings (i)-(iii) the minimal inter-layer skew is always 50, showing

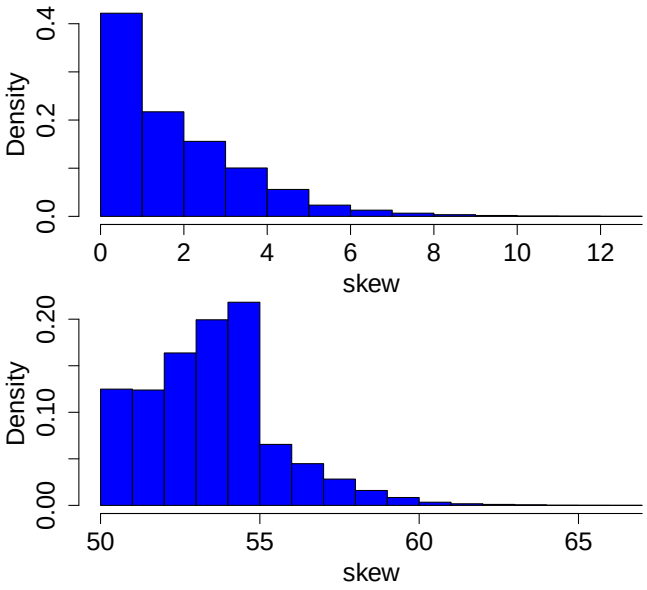

Figure 8: Cumulated histograms for intra-layer (top) and inter-layer (bottom) skew for uniformly chosen link delays in $[50, 55]$ and $\sigma_0 = \Delta_0 = 0$.

that all nodes were centrally triggered and no "intervention" by neighbors within a layer was ever necessary.

Table 2 gives the skews for link delays chosen uniformly in $[50, 75]$ (such that $d^+ = 1.5d^-$ and $\varepsilon/d^+ = 1/3$). Note that the latter choice of d^+ and d^- violates the constraint $\varepsilon \leq d^+/7$ of Theorem 3.8; we included this scenario to also shed some light on the behavior of HEX in extreme situations, however. Qualitatively, the situation is very similar to the previous one, except that the grid does not recover as well from the large initial skews of (iv). Small values of ε hence provide excellent skew bounds and fast recovery from initial skews in the fault-free setting, which has been supported by preliminary simulation runs as well.

Failures. To back up our considerations related to failures in Section 3.2, we provide two 3D plots of wave propagation in the presence of crashed and Byzantine faulty nodes, respectively, which clearly demonstrate the claimed failure locality property. Figure 9 shows a setup where a whole cluster of nodes $((3, 7), (3, 8), (4, 6), (4, 7), (4, 8), (5, 6)$ and $(5, 7))$ has crashed.[7] Observe that the disturbances (= increase) of the skew emanating from the faulty nodes fade with the distance from the fault location, and how nodes above establish synchrony via left- and right-triggering.

Figure 10 shows a scenario with a single Byzantine node $(2, 8)$ that sends a zero-delay trigger message to its right and up-right neighbors when triggered itself and delays the messages to its other neighbors for $3d^+$. It is evident from the figure that the disruptive power of this fault is not much worse than that of a crash fault. This is also confirmed by Table 3, which provides the analog to the fault-free results given in Table 1. Further simulations, shown in Figure 11 and Figure 12, confirmed that this also holds in the presence of multiple (separated) Byzantine faults.

[7]Note that the Node $(6, 6)$ cannot make progress as both neighbors in layer 5 have crashed.

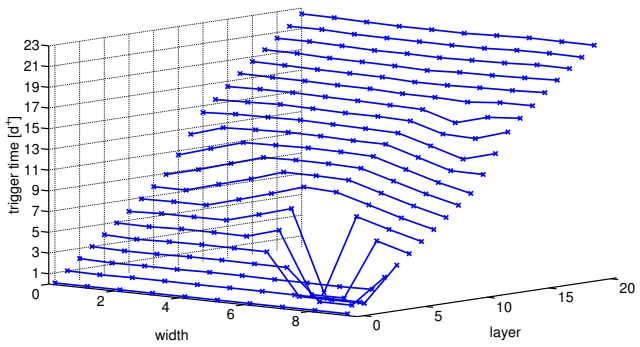

Figure 9: Pulse wave propagation for uniformly chosen link delays in $[50, 55]$ and $\sigma_0 = 0$, with a cluster of crashed nodes ("fake" trigger time -1).

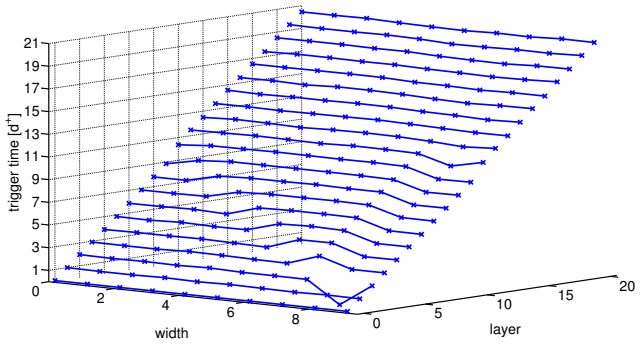

Figure 10: Pulse wave propagation for uniformly chosen link delays in $[50, 55]$ and $\sigma_0 = 0$, with one Byzantine node ("fake" trigger time of -1).

5. CONCLUSIONS & FUTURE WORK

In this work, we proposed a candidate for a scalable and fault-tolerant alternative for clock distribution in VLSI circuits. Whereas our analysis proved that HEX has excellent features, several important issues still need to be addressed for making the approach feasible in practice. An important part of our future work is devoted to the following topics:

Clock Frequency. Due to the tick separation requirement required for self-stabilization (cf. Theorem 3.11) of Algorithm 1, HEX cannot be used for distributing very high-frequency clock pulses. Several solutions are conceivable to eventually achieve this: Local clock multipliers, e.g. (i) Phase-Locked Loops (PLL) that lock on the low-frequency pulses, or (ii) pausible high-frequency local clocks started upon a tick that generate a fixed number of fast clock pulses. Note that (i) provides smoother clocks but requires low-jitter input clocks (guaranteed by HEX at least for static crash failures). Alternatively, (iii) pipelining of ticks as in [9] could be used. This would avoid the need for sleeping times and tick separation, but requires additional hardware for locally counting ticks and a considerably more involved self-stabilization analysis.

Embedding. The presented topology can be embedded into a VLSI circuit using two interconnect layers: One simply "squeezes" the cylindric shape of the HEX grid flat. However, this simplistic solution has two substantial drawbacks. First, the now physically close nodes from opposite "sides"

	intra-layer		inter-layer		
init. layer 0	avg	max	min	avg	max
0	6.51	64	41	56.15	113
rand. $[0, d^-]$	7.16	82	27	56.79	140
rand. $[0, d^+]$	7.24	85	24	56.87	143
ramp d^+	13.05	165	-18	59.85	162

Table 3: Skews of all nodes in the grid from 100 simulation runs with a single Byzantine node, for uniformly chosen link delays in $[50, 55]$.

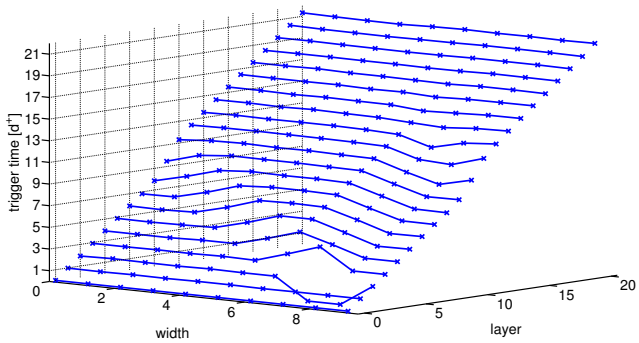

Figure 11: Pulse wave propagation for uniformly chosen link delays in $[50, 55]$ and $\sigma_0 = \Delta_0 = 0$, with two byzantine nodes ("fake" trigger time -1).

of the original cylinder are distant in the grid and therefore may suffer from larger skews. This might entail that actually half of the nodes cannot be used for clocking. Second, it might be difficult to synchronize the nodes at layer 0 unless they are physically close. This requires that either $W \ll L$ and the chip is rectangular with a large discrepancy in side lengths, or a way of (reliably and accurately) distributing a clock signal to nodes arranged in a line.

As a remedy, we propose the use of a slightly modified topology that arranges the nodes of each layer in a circular pattern. To avoid large variations in link lengths, we include "doubling layers" where we "duplicate" the nodes of a standard layer to quickly increase the number of nodes (see Figure 13). This topology has several advantages over a cylindric HEX grid: (i) only a few links require an additional interconnect layer, (ii) doubling layers will help disperse skews and thus may only improve the bounds,[8] and (iii) since the "initial" width W close to the clock source is very small and most doubling layers are close to the source as well, any initial skew will be mitigated very quickly.

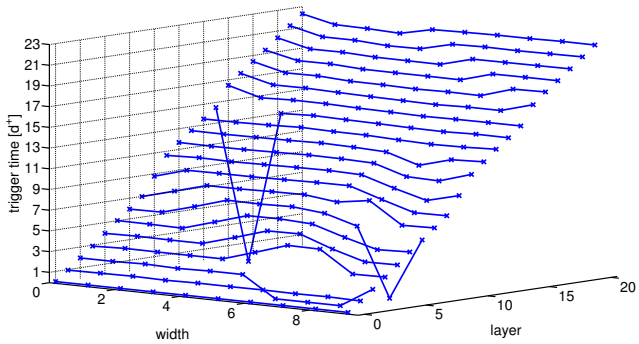

Figure 12: Pulse wave propagation for uniformly chosen link delays in $[50, 55]$ and $\sigma_0 = \Delta_0 = 0$, with multiple byzantine nodes ("fake" trigger time -1).

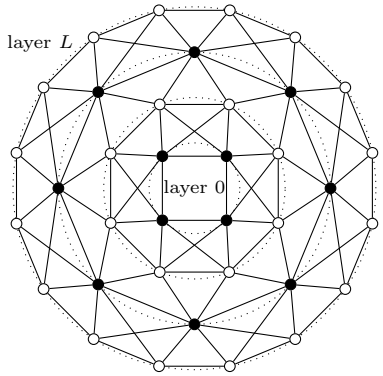

Figure 13: Alternative Topology. White nodes are in doubling layers. Doubling layers become less frequent with increasing distance from the center.

Fault-tolerance Properties. A more detailed theoretical analysis as well as measurements that determine realistic fault patterns are in order to make best use of the potential for resilience of the HEX topology. In particular, a simple solution for self-stabilization despite ongoing Byzantine faults seems both possible and highly desirable, and a quantitative analysis of the skews in the presence of a larger number of faults is of interest. Concerning the latter, suitable probabilistic fault models are to be established and analyzed.

Skews in Realistic Executions. Regarding the skew bounds from Theorem 3.8, even for a fairly large value of ε, say $0.1d^+$, it appears reasonable to assume that $\min\{L, W\} \cdot \varepsilon^2/d^+ \leq d^+$, since otherwise the chip would comprise at least 10,000 HEX nodes. Moreover, since link lengths are near-identical (or vary within a small constant factor for the alternative topology proposed above), smaller ε should be affordable at reasonable costs. The skew bound is thus essentially $\mathcal{O}(d^+)$, where the constant depends on the fault patterns and the fault locality properties of HEX. This should be contrasted with the skew on a clock tree, where the dominating term is the maximal difference of signal propagation times to the leaves. While the latter may be small compared to the total delays in a tree, one needs to take into account that the total distance bridged by the signal is (for an H-tree) close to half of the circumference of the (rectangular) chip. In particular for large chips, HEX may thus be competitive in terms of the skew between physically close functional units, despite its excellent resilience to faults! Improving our understanding of the skew incurred by HEX in realistic executions with multiple faults is thus a pivotal part of our future work.

Static Systems. An important special case occurs when the delay of each individual link is the same in each pulse (up to negligible fluctuations) and faults are *static* in the sense that the respective nodes respond identically in each

[8] This follows from the proofs in Section 3.1; the rightward/leftward links vital for the slower node catching up are in smaller layers than those required to trigger a fast node early.

pulse. Then the system will stabilize to a fixed triggering pattern that is identically reproduced for each pulse. Note that this covers an important class of realistic fault patterns, in particular, manufacturing defects and electric wear-out as well as the result of some dynamic grid reconfiguration in case of failures. Understanding this setting is thus of high interest for potential applications of our scheme.

6. REFERENCES

[1] R. Bhamidipati, A. Zaidi, S. Makineni, K. Low, R. Chen, K.-Y. Liu, and J. Dalgrehn. Challenges and Methodologies for Implementing High-Performance Network Processors. *Intel Technology Journal*, 6(3):83–92, 2002.

[2] D. M. Chapiro. *Globally-Asynchronous Locally-Synchronous Systems*. PhD thesis, Stanford University, 1984.

[3] E. W. Dijkstra. Self-Stabilizing Systems in Spite of Distributed Control. *Communications of the ACM*, 17(11):643–644, 1974.

[4] C. Dike and E. Burton. Miller and Noise Effects in a Synchronizing Flip-Flop. *IEEE Journal of Solid-State Circuits*, SC-34(6):849–855, 1999.

[5] 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)*, pages 163–177, 2011.

[6] S. Fairbanks. Method and Apparatus for a Distributed Clock Generator, 2004. US patent no. US2004108876.

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

[8] E. G. Friedman. Clock Distribution Networks in Synchronous Digital Integrated Circuits. *Proceedings of the IEEE*, 89(5):665–692, 2001.

[9] 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)*, pages 230–239, 2010.

[10] M. Függer and U. Schmid. Reconciling Fault-Tolerant Distributed Computing and Systems-on-Chip. *Distributed Computing*, 24(6):323–355, 2012.

[11] V. Gutnik and A. Chandrakasan. Active GHz Clock Network Using Distributed PLLs. *IEEE Journal of Solid-State Circuits*, 35(11):1553–1560, 2000.

[12] IEEE-SA Standards Board. IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pages c1–269, 2008.

[13] D. J. Kinniment, A. Bystrov, and A. V. Yakovlev. Synchronization Circuit Performance. *IEEE Journal of Solid-State Circuits*, SC-37(2):202–209, 2002.

[14] A. Korniienko, E. Colinet, G. Scorletti, E. Blanco, D. Galayko, and J. Juillard. A Clock Network of Distributed ADPLLs Using an Asymmetric Comparison Strategy. In *Proc. 2010 Symposium on Circuits and Systems (ISCAS)*, pages 3212–3215, 2010.

[15] D.-J. Lee, M.-C. Kim, and I. Markov. Low-power Clock Trees for CPUs. In *Proc. 2010 Conference on Computer-Aided Design (ICCAD)*, pages 444–451, 2010.

[16] D.-J. Lee and I. Markov. Multilevel Tree Fusion for Robust Clock Networks. In *2011 Conference on Computer-Aided Design (ICCAD)*, pages 632–639, 2011.

[17] 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)*, pages 41–44, 2003.

[18] D. G. Messerschmitt. Synchronization in Digital System Design. *IEEE Journal on Selected Areas in Communications*, 8(8):1404–1419, 1990.

[19] C. Metra, S. Francescantonio, and T. Mak. Implications of Clock Distribution Faults and Issues with Screening them During Manufacturing Testing. *IEEE Transactions on Computers*, 53(5):531–546, 2004.

[20] 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)*, pages 578–592, 2009.

[21] C. L. Portmann and T. H. Y. Meng. Supply Noise and CMOS Synchronization Errors. *IEEE Journal of Solid-State Circuits*, SC-30(9):1015–1017, 1995.

[22] S. Reddy, G. Wilke, and R. Murgai. Analyzing Timing Uncertainty in Mesh-based Clock Architectures. In *Proc. Design, Automation and Test in Europe (DATE)*, volume 1, pages 1–6, 2006.

[23] P. Restle, T. McNamara, D. Webber, P. Camporese, K. Eng, K. Jenkins, D. Allen, M. Rohn, M. Quaranta, D. Boerstler, C. Alpert, C. Carter, R. Bailey, J. Petrovick, B. Krauter, and B. McCredie. A Clock Distribution Network for Microprocessors. *IEEE Journal of Solid-State Circuits*, 36(5):792–799, 2001.

[24] 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)*, pages 30–35, 2001.

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

[26] R. Shelar. Routing with Constraints for Post-Grid Clock Distribution in Microprocessors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(2):245–249, 2010.

[27] C. N. Sze. ISPD 2010 High Performance Clock Network Synthesis Contest: Benchmark Suite and Results. In *Proc. 19th Symposium on Physical Design (ISPD)*, pages 143–143, 2010.

[28] P. Teehan, M. Greenstreet, and G. Lemieux. A Survey and Taxonomy of GALS Design Styles. *IEEE Design and Test of Computers*, 24(5):418–428, 2007.

[29] C. Yeh, G. Wilke, H. Chen, S. Reddy, H. Nguyen, T. Miyoshi, W. Walker, and R. Murgai. Clock Distribution Architectures: a Comparative Study. In *Proc. 7th Symposium on Quality Electronic Design (ISQED)*, pages 85–91, 2006.