HEX: Scaling Honeycombs is Easier than Scaling Clock Trees[☆]

Danny Dolev[a,1], Matthias Függer[b], Christoph Lenzen[c,2], Martin Perner[b,*], Ulrich Schmid[b]

[a]*School of Engineering and Computer Science, The Hebrew University of Jerusalem, Edmond J. Safra Campus, 91904 Jerusalem, Israel*
[b]*Department of Computer Engineering, TU Wien, Treitlstrasse 3, 1040 Vienna, Austria*
[c]*Max Planck Institute for Informatics, Campus E1 4, 66123 Saarbrücken, Germany*

Abstract

We argue that a hexagonal grid with simple intermediate nodes is a robust alternative to buffered clock trees typically used for clock distribution in VLSI circuits, multi-core processors, and other applications that require accurate synchronization: Our HEX grid is Byzantine fault-tolerant, 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 a very small average skew.

Keywords: clock distribution, Byzantine fault-tolerance, self-stabilization, skew analysis, simulations

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, multi-core processors, and other hardware devices (as well as in master-slave-type network clock synchronization approaches like IEEE1588 [1]), 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 are used to ensure that clock pulses arrive at all functional units (i.e., those making up a synchronous sub-system) simultaneously. Such topologies are, for example, H-trees,[3] combined with carefully engineered wire geometries, clock signal regeneration buffers, etc. This must be achieved with a *clock skew*, i.e., 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 clock 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

[☆]This work has been supported by the Austrian Science Foundation (FWF) projects FATAL (P21694) and SIC (P26436), 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.

[*]Corresponding author. Phone: +43 (1) 58801-18261.

Email addresses: `dolev@cs.huji.ac.il` (Danny Dolev), `fuegger@ecs.tuwien.ac.at` (Matthias Függer), `clenzen@mpi-inf.mpg.de` (Christoph Lenzen), `mperner@ecs.tuwien.ac.at` (Martin Perner), `s@ecs.tuwien.ac.at` (Ulrich Schmid)

[1]Danny Dolev is Incumbent of the Berthold Badler Chair.

[2]Christoph Lenzen has been supported by the Swiss National Science Foundation (SNF), the Swiss Society of Friends of the Weizmann Institute of Science, and the German Research Association (DFG, reference number Le 3107/1-1) for this work.

[3]H-trees are recursively constructed from a H-shaped wiring topology by attaching four smaller H-shapes to the four open ends

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: Elaborate clock tree engineering must ensure that the maximum delay discrepancy remains below the acceptable clock skew, which is very difficult for clock speeds in the GHz range [2, 3, 4, 5]. Modern clock trees thus incorporate complex wire geometries and strong clock buffers, implying large area and power consumption [6]. Moreover, mitigating the inevitable skews from disjoint root-leaf paths requires extended topologies, such as trees with cross-links, meshes and multi-level trees [5, 7].

An even more serious issue with clock trees is lacking robustness, which also arises in applications where there are no severe skew requirements. 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) [8] architectures, where different parts of a chip are clocked by different clock sources and 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 [9, 10, 11, 12]. Multi-synchronous clocking [13, 14] (also called *mesochronous* clocking [15]), which guarantees some upper bound on the skew between clock domains, has been invented to avoid this. The resulting architectures can rely on a common time base, which is attractive not only for application programmers, but also for metastability-free high-speed communication between different clock domains [16].

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 breakdown [17], *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 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*.[4] 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 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, or mute clock sources and nodes. It is self-stabilizing [18], in the sense that it recovers from an arbitrary number of transient faults, despite persisting isolated Byzantine failures.

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, wires between HEX nodes are much shorter than in a clock tree: Assuming a constant spatial node density, the total number of nodes $n = \Theta(s^2)$ is proportional to the square of the width and height s of a quadratic grid. As $s = \Theta(\sqrt{n})$, the wire length between neighbors in HEX is, with optimal layout (cf. Section 5), only $\Theta(1)$. By contrast, the height of a clock tree is $\Theta(\log n)$ thus, even with optimal layout, their will be neighbors which are separated by a wire length of $\Omega(\sqrt{n})$. HEX hence neither requires strong clock buffers nor special wire geometries, so that the maximal difference ε of the end-to-end delays between neighbors in the grid could easily be kept small even by moderate engineering efforts.

Second, for a proper embedding of the HEX topology, physically close nodes are reasonably well-synchronized. It is well-known, e.g., from [19], that no deterministic clock synchronization algorithm can

[4]Note that clock distribution by means of our HEX grid is fundamentally different from using a clock mesh [5] for averaging out large clock skews among nearby leaf nodes.

guarantee a worst-case skew between *all* pairs of nodes that is better than $D\varepsilon/2$, where D is the diameter of the underlying communication graph. Moreover, the gradient clock synchronization lower bounds established in [20] reveal that the skew between *neighbors* cannot be better than $\Omega(\varepsilon \log D)$. We will show that the neighbor skew provided by HEX is at most $\mathcal{O}(D\varepsilon^2)$, where D is in fact the width of the grid. Depending on the number and severity of faults, this skew bound gracefully degrades.

Paper organization

After a short description of related work, the HEX topology and algorithm as well as the system model are introduced in Section 2. In Section 3, we provide a detailed analysis of the worst-case neighbor skew of HEX (Section 3.1), discuss the effect of faulty nodes on the skew (Section 3.2), and analyze the self-stabilizing properties of HEX (Section 3.3). As the complexity of the skew analysis explodes for an increasing number of faults, the detailed analysis in the case of a single Byzantine faulty node is provided in Appendix A. In Section 4, we provide the corresponding results of our extensive simulation experiments, which have been obtained via a custom Modelsim-based simulation and analysis framework. In Section 5, we discuss practical extensions of HEX, in particular, frequency multiplication and alternative grid topologies. Some conclusions and directions of future work in Section 6 complete the paper. A glossary of the notations used can be found on the last page of this paper.

Related work

Apart from the rich literature on clock tree engineering and extended topologies for skew reduction, see for example [2, 3, 4, 5, 6, 7, 17, 21, 22], 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 [23], a regular structure of closed loops of an odd number of inverters is used for distributed clock generation. Similarly, [24, 25] 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, like a grid [26, 27, 28]. 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 [29, 30] and our self-stabilizing Byzantine fault-tolerant FATAL algorithm proposed in [31]. However, both approaches are complex and require a fully connected topology. Consequently, they are not useful for distributing a synchronized clock to a large number of functional units, but are rather 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 (V, E) of our HEX grid is defined as follows (see Figure 1): 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 $(\ell, i) \in [L+1] \times [W]$. Here, $[L+1] := \{0, \dots, L\}$ 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 $(\ell, i) \in V$, $0 < \ell \in [L+1]$, $i \in [W]$, 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)); (ℓ, i) also has incoming links from $(\ell-1, i)$, called its lower left neighbor, and $(\ell-1, i+1 \bmod W)$, called its lower right neighbor. Hence, if (ℓ, i) is in a layer $\ell \in [L]$, then it has outgoing links to $(\ell+1, i-1 \bmod W)$, its upper left neighbor, and $(\ell+1, i)$, its upper

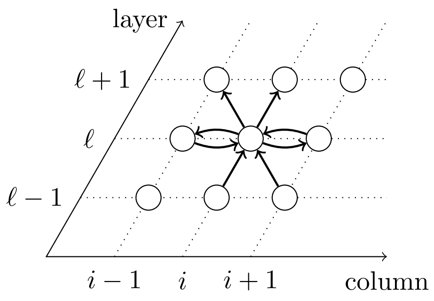


Figure 1: Node (ℓ, i) and its incident links in the cylindric hexagonal grid topology. Column coordinates are modulo W and layer coordinates (rows) are between 0 and L.

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 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 a communication delay (i.e., the time between sending and receiving a trigger message) within $[d^-, d^+] \subset (0, \infty)$, where $\varepsilon := d^+ - d^-$. Having ε without any constraint, however, could render a few worst-case constructions (based on Definition 2) overly conservative. This can be avoided by the additional constraint $\varepsilon \leq d^+/2$, which guarantees a property similar to the triangle inequality. Each node further has access to a (possibly inaccurate) clock to measure timeouts.

Nodes at layer 0 are special as they act as primary clock sources, i.e., they execute a pulse generation algorithm like the one of [30, 31] that generates 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 3.3. 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 HEX 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 carry no information beside their 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 the k^{th} and $(k+1)^{\text{th}}$ pulse (for each k), which relieves us from locally keeping track of pulse counts. For each link, a node memorizes a received trigger message only for some time between $T^-_{\textbf{link}}$ and $T^+_{\textbf{link}}$, $T^+_{\textbf{link}} \geq T^-_{\textbf{link}}$, where the slack $T^+_{\textbf{link}} - T^-_{\textbf{link}}$ accounts for inaccurate local timers, and then forgets the reception of the message by clearing the *memory flag* associated with the link. After having forwarded its pulse, a node goes to sleep (i.e., will not locally trigger further pulses) for some time between $T^-_{\textbf{sleep}}$ and $T^+_{\textbf{sleep}} \geq T^-_{\textbf{sleep}}$. Upon waking up, it clears all its memory flags. Note that there would be no need for the individual link timeout mechanism ($[T^-_{\textbf{link}}, T^+_{\textbf{link}}]$) described above if the algorithm always started from a properly initialized state. It is required, however, for also guaranteeing self-stabilization from arbitrary states in the presence of persistent Byzantine faults.

The precise conditions for $T^-_{\textbf{link}}$ and $T^-_{\textbf{sleep}}$ follow from the analysis and are discussed in Section 3.3. Due to its simplicity, Algorithm 1 can easily be implemented by means of an asynchronous state machine, see Figure 7a in Section 4.

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

```
upon receiving trigger message from neighbor do
    memorize message for $\tau \in [T^-_{\textbf{link}}, T^+_{\textbf{link}}]$ time;
upon having memorized trigger messages from (left and lower left) or
        (lower left and lower right) or (lower right and right) neighbors do
    broadcast trigger message; // produce pulse
    sleep for $\tau \in [T^-_{\textbf{sleep}}, T^+_{\textbf{sleep}}]$ time;
    forget previously received trigger messages;
```

3. Skew & Resilience Analysis

In this section, we analyze skew and fault-tolerance properties of the HEX algorithm in the topology presented in the previous section. Recall that nodes in layer 0 generate synchronized pulses, which the nodes in higher layers just propagate upwards; this results in a "pulse wave" as depicted in Figure 8. By $t^{(k)}_{\ell,i}$, we denote the *triggering time* of node (ℓ, i), i.e., the time when it forwards the k^{th} pulse of the grid. Generally, we will use superscript $^{(k)}$ to denote variables associated with the k^{th} pulse.

3.1. The Fault-free Case

We will now analyze the propagation of a single pulse wave,[5] assuming that the constraints (C1) and (C2) below are satisfied and no nodes are faulty. In a nutshell, our constraints ensure that, initially, all nodes have cleared their memory flags and are waiting for the next pulse generated by the nodes in layer 0.

(C1) $T^-_{\textbf{link}}$ is sufficiently large so that no trigger message from a neighbor is "forgotten" before the corresponding message from another neighbor arrives. Thus we can be sure that a node which is not sleeping will be triggered by a wave.

(C2) $T^-_{\textbf{sleep}}$, $T^+_{\textbf{sleep}}$, and the time between pulses (controlled by the layer 0 nodes) are large enough so that (i) every node will be triggered at most once per wave and (ii) no node sleeps when the next wave arrives.

Specific values for the parameters that ensure (C1) and (C2) will be given in Section 3.3.

We first introduce the concept of "left zig-zag paths", which will play an essential role in bounding the worst-case triggering times of adjacent nodes in a single pulse wave.

Definition 1 (Causal Links and Paths). A node is *left-triggered / centrally triggered / right-triggered* in a given execution, if the satisfied guard from Algorithm 1 causing the node to trigger is having 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.

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 from a given destination node, either to the node in layer 0 starting the causal chain or to some specific column of interest. Note that this can be done for any destination node in a given execution.

Definition 2 (Left Zig-Zag Paths). Given are a layer $0 < \ell \in [L+1]$ and column indices $i, i' \in [W]$, $i < i'$.[6] The causal *left zig-zag path* $p^{i' \to (\ell,i)}_{\text{left}}$ is composed of rightward links $((\ell', j-1), (\ell', j))$ and up-left

[5] To keep the notation simple, we will drop the superscript $^{(k)}$ indicating the pulse number during this section.

[6] Recall that column indices are implicitly taken mod W, so in principle we would have to account for this in the definition. However, in our proofs we will exploit the translation and mirror symmetry of the grid w.r.t. column indices so that we can always assume that $i < i'$, avoiding more involved notations. For example, if $i = W - 1$, then some node in the column to the right (which has column index 0) is considered as having index $i' = W$.

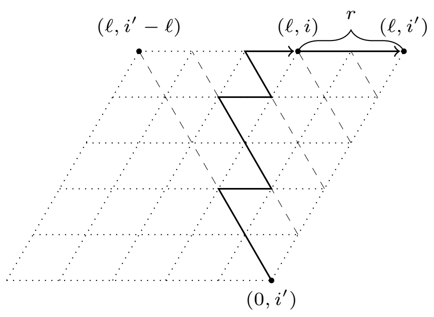


Figure 2: Illustration of the situation in Lemma 2.

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 either (i) $j+1 = i'$ and the path now contains more up-left than rightward links (we will call $p_{\text{left}}^{i' \to (\ell,i)}$ a *triangular path* in this case) or (ii) $\ell' - 1 = 0$ and $j+1$ arbitrary (a non-triangular path).

The following simple facts about left zig-zag paths follow almost immediately from their definition.

Lemma 1. *Every left zig-zag path $p_{\text{left}}^{i' \to (\ell,i)}$ constructed according to Definition 2 is finite. If $p_{\text{left}}^{i' \to (\ell,i)}$ is a triangular path and starts at (ℓ', i'), for some $0 \leq \ell' < \ell$, then each of its prefixes π is also a triangular path.*

Proof. Since causal paths are acyclic, there must be fewer than W left links before the construction goes down one layer; the finiteness of ℓ hence implies the finiteness of $p_{\text{left}}^{i' \to (\ell,i)}$. Now assume that some prefix π of a triangular path $p_{\text{left}}^{i' \to (\ell,i)}$ starting at (ℓ', i') is not a triangular path, i.e., has at least as many rightward links than up-left ones. Then, the suffix of π must start in (ℓ'', i'') with $i'' \geq i'$, and must have more up-left links than rightward ones. Since the suffix must hence cross column i' from right to left, the construction of Definition 2 would already have terminated here. $\square$

We now provide a very important technical lemma, which reveals a connection between the triggering times of two nodes (ℓ, i) and $(\ell, i+1)$ at the same layer ℓ: If the left node is the end of a left zig-zag triangular path starting at node (ℓ', i') and has a distance of $r > 0$ columns to the right node, the latter cannot trigger later than the left node plus a time offset of at most $rd^- + (\ell - \ell')\varepsilon$. Note that we assume here that $i' - \ell \geq 0$, i.e., that W is large enough such that no wrap-around occurs. The bound provided by Lemma 2 also holds in the general case, but then it may not be tight.

Lemma 2. *Suppose that path π is a prefix of some left zig-zag triangular path $p_{\text{left}}^{i' \to (\ell'',i'')}$, and that π starts at node (ℓ', i') and ends at node (ℓ, i) with $\ell > 0$. Let $r > 0$ be the number of up-left links minus the number of rightward links along π. Then $t_{\ell,i'} \leq t_{\ell,i} + rd^- + (\ell - \ell')\varepsilon$.*

Proof. By Lemma 1, π is a triangular path and hence indeed $r > 0$. For simplicity of our arguments, we set $\ell' = 0$ (i.e., we shift all layer indices by ℓ' and the new value of ℓ now represents $\ell - \ell'$) and assume that $i' - \ell \geq 0$, i.e., that W is large enough such that no wrap-around occurs within the triangle.[7] Consequently, we only need to consider the set S of nodes in the triangle with corners $(0, i')$, $(\ell, i' - \ell)$, and (ℓ, i') shown in Figure 2.

[7]Our proof also holds for the general case, though, provided (i) one just neglects the fact that some of the index pairs may actually refer to the same node (which does not affect our argument) and that (ii) certain left zig-zag paths in our construction cannot occur (which may lead to an overly conservative bound).

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} left-diagonal $(k, i'), \ldots, (\ell, i' - (\ell - k))$ (for $k \in [\ell + 1]$) of the triangle, 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

$$t_p \leq t_{\ell,i} - (\ell - r)d^- + kd^+. \tag{1}$$

Since (ℓ, i') is on diagonal ℓ, this implies $t_{\ell,i'} \leq t_{\ell,i} + rd^- + \ell\varepsilon$. Undoing the initial index shift (i.e., replacing ℓ by $\ell - \ell'$), the claim of the lemma follows.

First, we show directly that Eq. (1) holds for each node p 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

$$t_p \leq t_{\ell,i} - hd^- \leq t_{\ell,i} - (\ell - r)d^- + kd^- \leq t_{\ell,i} - (\ell - r)d^- + kd^+, \tag{2}$$

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 $(k + 1)^{\text{th}}$ diagonal that is strictly to the 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 $t_p \leq t_{\ell,i} - (\ell - r)d^- + kd^+ + d^+$ as required. On the other hand, nodes lying on π are covered by Eq. (2), which completes the induction step. □

In the following definition, we will introduce the different notions related to the skew between nodes. Besides the maximum (unsigned) intra-layer skew of neighboring nodes at the same layer and the (signed) maximum inter-layer skew w.r.t. the layer below, we also define the *skew potential* of layer ℓ. Informally, the latter provides a measure for the adversary's ability to exploit the existing skew of the nodes in layer ℓ to increase the skew of neighboring nodes in layer $\ell + 1$. By this, we mean that the adversary can, in the worst case, force a node at layer $\ell + 1$ to left-trigger stricly before it is centrally triggerd by its layer-ℓ neighbors. It is not too difficult to prove[8] that this is only possible if the skew between neighbors at layer ℓ is strictly larger than d^-. Hence, we define the skew potential below in a way that results in a positive value only in the latter case.

Definition 3 (Distance, Skew, and Skew Potential). For $i, j \in \mathbb{Z}$, let $d := i - j \bmod W$ and define the cyclic distance as $|i - j|_{\text{W}} := \min\{d, W - d\}$. For $\ell \in [L + 1]$, we define

(i) the *intra-layer skew* of layer ℓ as $\sigma_\ell := \max_{i \in [W]}\{|t_{\ell,i} - t_{\ell,i+1}|\}$,

(ii) the *skew potential* on layer ℓ as $\Delta_\ell := \max_{i,j \in [W]}\{t_{\ell,i} - t_{\ell,j} - |i - j|_{\text{W}} d^-\}$.

For $\ell \in [L + 1] \setminus \{0\}$, we define

(iii) the *inter-layer skew* of layer ℓ as $\hat{\sigma}_\ell := \max_{i \in [W]}\{t_{\ell,i} - t_{\ell-1,i}, t_{\ell,i} - t_{\ell-1,i+1}\}$.

Note that every pair of nodes i, j occurs twice (as i, j and j, i) in the max-term of the skew potential in (ii) above, which implies that only a non-negative time difference can determine Δ_ℓ. Moreover, as $j = i$ is not excluded, we always have $\Delta_\ell \geq 0$.

We first prove a weak bound on the maximal skew at the upper layers that holds *independently* of the initial skew potential Δ_0. Note that this result implies tolerance of HEX against arbitrary layer 0 skews, at the expense of "losing" layers $\ell \in [W - 2]$. This behavior is also clearly visible in the simulation results shown in Figures 9 and 12.

[8]In fact, this proof is embedded in the proof of Lemma 4. In a nutshell, it shows that such an early left-triggering would only be possible if the left neighbor itself was early left-triggered as well. By continuing this argument inductively over the entire layer, a left neighbor will eventually be reached that cannot be left-triggered, which provides the required contradiction.

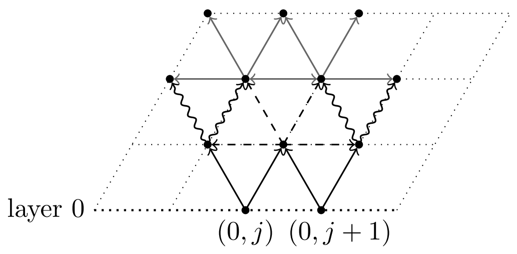


Figure 3: Illustration of the induction in the proof of case 2 of Lemma 3. The solid lines mark the trigger messages sent by time t_0, the dashed black lines symbolize the trigger messages that will be sent by time $t_0 + d^+$. The black wiggly lines resp. the gray solid ones represent (some) trigger messages that will sent by time $t_0 + 2d^+$ resp. $t_0 + 3d^+$.

Lemma 3. *For $W > 2$ and all $\ell \in \{W-2, \dots, L\}$, $\Delta_\ell \leq 2(W-2)\varepsilon$.*

Proof. Consider any fixed $i, i' \in [W]$, $i < i'$ (wrap-around cases are symmetrical) and assume that $\ell = W-2$; we will argue later on why the proof below also covers $\ell > W-2$. 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 2,

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

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 k-th node on $p_{\text{left}}^{i' \to (\ell,i)}$, $k \geq 2$, cannot be triggered before time $t_0 + (k-1)d^-$ because it is connected by a causal path of length $k-1$ to a layer-0 node that is triggered at or after time t_0. Hence, $t_{\ell,i} \geq t_0 + (2\ell - (i'-i))d^-$ and thus

$$t_{\ell,i} \geq t_0 + (2(W-2) - (i'-i))d^-. \tag{3}$$

Denote by $(0,j)$ a node with $\max\{t_{0,j}, t_{0,j+1}\} = t_0$; by the 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 $t_0 + 2\lambda d^+$; an illustration of the first layers is shown in Figure 3. Since in layer λ these are $2+\lambda$ nodes, i.e., all W nodes in layer $W-2$, this will prove the claim of our lemma.

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 $(\lambda+1, j-(\lambda+1))$ resp. $(\lambda+1, j+1)$ must also follow since they are right- resp. left-triggered (if not triggered differently before), which completes the induction.

The result of our induction proof implies $t_{\ell,i'} \leq t_0 + 2(W-2)d^+$ and hence, by using Eq. (3),

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

Overall, since i and $i' > i$ were arbitrary, from the two cases and the symmetry properties of the grid, 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.

Finally, since the above proof did not require any specific property to be respected by layer 0 nodes, it also applies literally if we replace layer $W-2$ by ℓ and layer 0 by layer $\ell - W + 2$. This concludes the proof of Lemma 3. $\square$

Next, we derive more refined bounds on the intra-layer skew between two neighboring nodes at the same layer $\ell > 0$: In contrast to Lemma 3, we now take the maximal skew in previous layers into account. Unfortunately, its proof is complicated by the fact that we need to distinguish three different cases that might lead to the worst-cast skew $s_{\ell,i}$ between nodes (ℓ,i) and $(\ell,i+1)$. They depend on whether the skew

$s_{\lambda,i}$ of the corresponding nodes (λ, i) and $(\lambda, i+1)$ at some (suitably chosen) layer λ is $s_{\lambda,i} \leq d^+$ (Case 1) or else $s_{\lambda,i} > d^+$ (Cases 2 and 3).

Informally, Case 1 is characterized by a V-shaped growth of the worst-case skew: Our detailed proof will show that $s_{\lambda,i}$ increases a most by ε with every layer. By contrast, in the other cases, the nodes (λ, i) and $(\lambda, i+1)$ at layer λ are already "torn apart". The worst-case skew in this case is determined by a left zig-zag path $p_{\text{left}}^{i+1\to(\ell,i)}$ that causes $t_{\ell,i}$ to be as small as possible on the one hand, and a "slow" causal path ending at $(\lambda, i+1)$ that makes $t_{\ell,i+1}$ as large as possible on the other hand. Cases 2 and 3 are distinguished according to the two possibilities where $p_{\text{left}}^{i+1\to(\ell,i)}$ can start here: Case 2 applies when it originates at some node $(0, j_0)$ with $j_0 \neq i+1$, whereas Case 3 is characterized by a triangular path starting at $(\ell', i+1)$.

Lemma 4. *For all $\ell_0 \in [L]$ and $\ell \in \{\ell_0 + 1, \ldots, L\}$, 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 $\ell \geq 1$ and assume, without loss of generality, that $\ell_0 = 0$. To simplify our arguments, we also assume $t_{\ell,i} < t_{\ell,i+1}$ (as the other cases are symmetric, this is sufficient).

Define $\lambda_0 := \lfloor \ell d^-/d^+ \rfloor$, which maximimes λ_0 under the constraint that $\lambda_0 d^+ \leq \ell d^-$. Thus, a "slow" chain of trigger messages will complete λ_0 hops within the time a "fast" chain requires for ℓ hops. We obtain

$$\ell - \lambda_0 = \ell - \left\lfloor \frac{\ell d^-}{d^+} \right\rfloor = \left\lceil \frac{\ell\varepsilon}{d^+} \right\rceil; \tag{4}$$

recall that $-\lfloor x \rfloor = \lceil -x \rceil$.

We distinguish three cases:

Case 1 V-shaped skews (Figure 4a): $t_{\lambda,i+1} \leq t_{\lambda,i} + d^+$ for some $\lambda \geq \lambda_0$. We choose λ maximal with this property, so that $t_{\lambda',i+1} > t_{\lambda',i} + d^+$ for all $\lambda' \in \{\lambda + 1, \ldots, \ell\}$. Notice that this 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 in this case. By induction on λ', we can thus infer $t_{\ell,i} \geq t_{\lambda,i} + (\ell - \lambda)d^-$.

Furthermore, $t_{\lambda',i+1} > t_{\lambda',i} + d^+$ ensures that the trigger message from (λ', i) to $(\lambda', i+1)$ arrives well before time $t_{\lambda',i+1}$. Thus, 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

$$t_{\ell,i+1} \leq t_{\lambda,i} + (\ell - \lambda + 1)d^+. \tag{5}$$

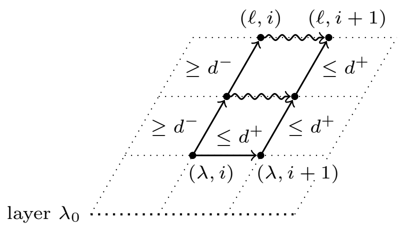
Combining these bounds and applying Eq. (4), we obtain

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

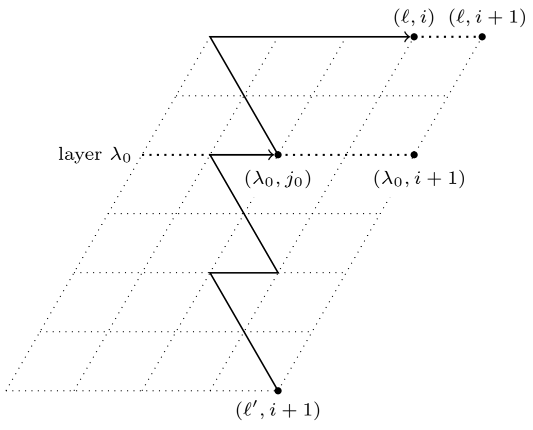
Case 2 Non-V-shaped skews, non-triangular (Figure 4c): Case 1 does not apply and $p_{\text{left}}^{i+1\to(\ell,i)}$ starts at some node $(0, j_0)$, for $j_0 \neq i+1$. If $p_{\text{left}}^{i+1\to(\ell,i)}$ contained more left-up links than rightward links, it would contain a subpath originating at a 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 $j \in \{i+1, i+2, \ldots, i+1+\lambda_0\}$, we have that $|j - j_0|_{\mathrm{W}} \leq j - i + r$, and hence, by definition of the skew potential, also

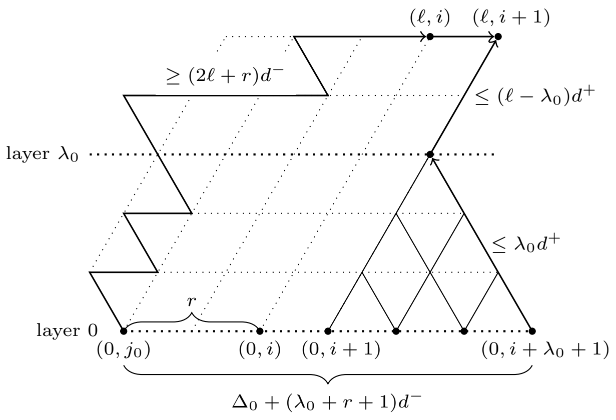
$$t_{0,j} - t_{0,j_0} = t_{0,j} - t_{0,j_0} - |j - j_0|_{\mathrm{W}} d^- + |j - j_0|_{\mathrm{W}} d^- \leq \Delta_0 + |j - j_0|_{\mathrm{W}} d^- \leq \Delta_0 + (\lambda_0 + r + 1)d^-.$$



(a) Case 1: V-shaped skews



(b) Case 3: Non-V-shaped skews (triangular)



(c) Case 2: Non-V-shaped skews

Figure 4: Illustrations of the different cases in the proof of Lemma 4.

Recalling the length of $p_{\text{left}}^{i+1\to(\ell,i)}$ established above, we obtain that

$$\begin{aligned} t_{\ell,i} &\geq t_{0,j_0} + (2\ell + r)d^- \\ &\geq t_{0,j} - \Delta_0 - (\lambda_0 + r + 1)d^- + (2\ell + r)d^- \\ &= t_{0,j} - \Delta_0 + (2\ell - \lambda_0 - 1)d^-. \end{aligned} \tag{6}$$

Moreover, by induction on $\lambda \in \{0, \ldots, \lambda_0\}$, it follows that all nodes $(\lambda, j') \in \{(\lambda, i+1), \ldots, (\lambda, (i+1+\lambda_0 - \lambda) \bmod W)\}$ are triggered at time $t_{\lambda,j'} \leq \max_{i<j\leq i+\lambda_0+1}\{t_{0,j}\} + \lambda d^+$. Plugging in Eq. (6) implies

$$t_{\lambda,j'} \leq t_{\ell,i} + \Delta_0 - (2\ell - \lambda_0 - 1)d^- + \lambda d^+ \tag{7}$$

and hence

$$t_{\lambda_0,i+1} \leq t_{\ell,i} + \Delta_0 - (2\ell - \lambda_0 - 1)d^- + \lambda_0 d^+ \leq t_{\ell,i} + \Delta_0 - (\ell - \lambda_0 - 1)d^-;$$

the second inequality holds by the definition of λ_0, which implies $\lambda_0 d^+ \leq \ell d^-$.

Since Case 1 does not apply, we have $t_{\lambda,i+1} > t_{\lambda,i} + d^+$ for all $\lambda_0 \leq \lambda \leq \ell$. We can hence use the same argument as used for deriving Eq. (5) to show that $t_{\ell,i+1} \leq t_{\lambda_0,i+1} + (\ell - \lambda_0)d^+$. It follows that

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

where the last equality follows from Eq. (4).

Case 3 Non-V-shaped skews, triangular (Figure 4b): Neither Case 1 nor Case 2 apply. In this case, $t_{\lambda,i+1} > t_{\lambda,i} + d^+$ for all $\lambda \in \{\lambda_0, \dots, \ell\}$, and $p_{\text{left}}^{i+1\to(\ell,i)}$ is a triangular path starting at node $(\ell', i+1)$ for some $\ell' < \lambda_0 - 1$: By construction, the first (causal) link of $p_{\text{left}}^{i+1\to(\ell,i)}$ is $((\ell', i+1), (\ell'+1, i))$, implying that node $(\ell'+1, i+1)$ is triggered no later than time $t_{\ell'+1,i} + d^+ = \max\{t_{\ell',i+1} + d^+, t_{\ell'+1,i} + d^+\}$. Hence, since Case 1 does not apply, we must indeed have $\ell' + 1 < \lambda_0$.

Let (λ_0, j_0) be the last node on the causal path $p_{\text{left}}^{i+1\to(\ell,i)}$ that is still in layer λ_0. Observe that $j_0 + r - u = i$, where r (resp. u) is the number of rightward (resp. up-left) hops of $p_{\text{left}}^{i+1\to(\ell,i)}$ after (λ_0, j_0). We apply Lemma 2 to the prefix π of $p_{\text{left}}^{i+1\to(\ell,i)}$ ending at (λ_0, j_0), i.e., set $i := j_0$, $i' = i+1$ and $r := i+1-j_0$ in this lemma, which yields

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

Since Case 1 does not apply, we can use the same induction as used before Eq. (5) to prove that $t_{\ell,i+1} \leq t_{\lambda_0,i+1} + (\ell - \lambda_0)d^+$. We thus obtain

$$\begin{aligned} t_{\ell,i+1} &\leq t_{\lambda_0,j_0} + (i+1-j_0)d^- + (\lambda_0 - \ell')\varepsilon + (\ell - \lambda_0)d^+ \\ &= t_{\lambda_0,j_0} + (\ell - \lambda_0 + i + 1 - j_0)d^- + (\ell - \ell')\varepsilon. \end{aligned}$$

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 $2(\ell - \lambda_0) + (i - j_0)$. As this suffix is a causal path, we have

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

Altogether, we arrive at

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

according to Eq. (4).

Since the claimed bound holds in each of the (exhaustive) cases considered, the proof of Lemma 4 is completed. □

We remark that it is possible to construct, by deterministically choosing appropriate link delays, worst-case executions that almost match the bounds established in Lemma 4; an example is shown in Figure 5.

In the proof of Lemma 4, in particular, in Case 2 (Figure 4c), we silently assumed that the starting node $(0, j_0)$ of the left zig-zag path $p_{\text{left}}^{i+1\to(\ell,i)}$ on the left side does not "collide" (due to a wrap-around) with one of the slow nodes $(0, i+1), \dots, (0, i+\lambda_0+1)$ on the right side. Whereas this is reasonable for wide grids, this is not realistic if W is small. Considering such a collision prohibits some of the worst-case scenarios considered, and hence possibly makes the worst-case skew result provided by Lemma 4 overly conservative. We therefore provide the following corollary, which takes this width constraint into account.

Corollary 1. *Set $\delta := d^-/2 - \varepsilon$. For each layer $\ell \in \{W, \dots, L\}$ and all $i \in [W]$, it holds that*

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

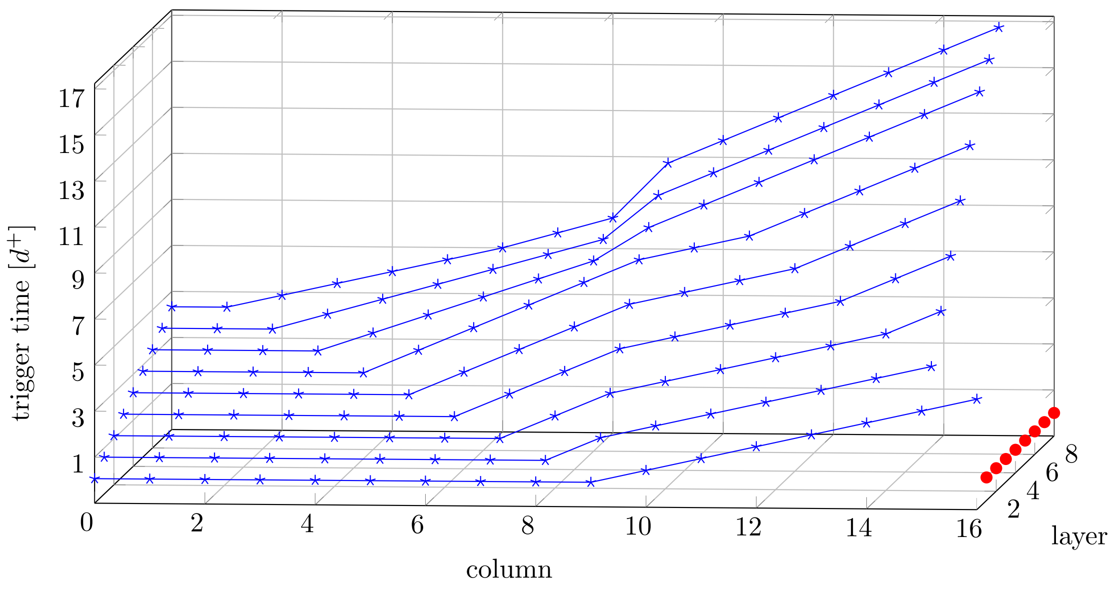


Figure 5: Visualization of a worst-case pulse propagation wave, maximizing the skew between the top-layer nodes in columns 8 and 9. To focus on the essential central part of the grid, we introduced a barrier of "dead" nodes in column 16. Nodes in and left of column 8 are left-triggered (except for the "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 parts of layer 0.

Proof. The proof is mostly analogous to the one of Lemma 4, with ℓ_0 resp. Δ_0 replaced by $\ell - W$ resp. $\Delta_{\ell-W}$. Case 2 needs a slightly different treatment, though, by assuming w.l.o.g. $\ell = W$ and recomputing the bound Eq. (6) for all indices $j \in \{i+1, i+2, \ldots, i+\lambda_0+1\}$ as

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

where we conservatively set $r = 0$ and exploit that $|j-j_0|_{\mathrm{W}} \leq W/2 = \ell/2$ in the second step. The analogon of Eq. (7) in the proof of Lemma 4, for $\lambda \in \{0, \ldots, \lambda_0\}$ and $(\lambda, j') \in \{(\lambda, i+1), \ldots, (\lambda, (i+1+\lambda_0-\lambda) \bmod W)\}$, hence reads

$$t_{\lambda,j'} \leq t_{\ell,i} + \Delta_0 - \frac{3\ell d^-}{2} + \lambda d^+$$

and thus leads to

$$t_{\lambda_0,i+1} \leq t_{\ell,i} + \Delta_0 - \frac{3\ell d^-}{2} + \lambda_0 d^+ \leq t_{\ell,i} + \Delta_0 - \frac{\ell d^-}{2},$$

where we used that $\lambda_0 d^+ \leq \ell d^-$ by the definition of λ_0. Finally, re-using the result $t_{\ell,i+1} \leq t_{\lambda_0,i+1} + (\ell - \lambda_0)d^+$ from the proof of Lemma 4, we arrive at

$$\begin{aligned} t_{\ell,i+1} &\leq t_{\ell,i} + \Delta_0 + (\ell - \lambda_0)d^+ - \frac{\ell d^-}{2} \\ &\leq t_{\ell,i} + \Delta_0 + \ell\varepsilon + d^+ - \frac{\ell d^-}{2} \\ &= t_{\ell,i} + \Delta_0 + d^+ - W\delta, \end{aligned}$$

where we used Eq. (4) to derive the second inequality.

Checking the bounds from Case 1 and Case 3 in Lemma 4, we see that they are smaller or equal to the left term in the maximum on the right hand side of the claimed bound. The bound for the differently treated Case 2 matches the right term in the maximum. □

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

Theorem 1 (Skew Bounds—Fault-free Case)**.** *Suppose that $\varepsilon \leq d^+/7$. If $\Delta_0 = 0$, then the intra-layer skew σ_ℓ (recall Definition 3) is uniformly bounded by $d^+ + \lceil W\varepsilon/d^+ \rceil \varepsilon$ for any $\ell \in [L+1]$. In the general case,*

$$\begin{aligned} \forall \ell \in \{1,\ldots,2W-3\}: &\qquad \sigma_\ell \leq d^+ + 2W\varepsilon^2/d^+ + \Delta_0. \\ \forall \ell \in \{2W-2,\ldots,L\}: &\qquad \sigma_\ell \leq d^+ + \lceil W\varepsilon/d^+ \rceil \varepsilon. \end{aligned}$$

The inter-layer skew of layer $\ell \in [L+1] \setminus \{0\}$, for all $i \in [W]$, is determined by

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

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 $\ell \in \{-(W-1),\ldots,0\}$ 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 $\ell \in \{-(W-1),\ldots,0\}$. From Lemma 3, we obtain that $\Delta_\ell \leq 2(W-2)\varepsilon$ for all $\ell \in \{1,\ldots,L\}$ (since we have negative layer indices until $-(W-1)$, the lemma also applies to layers $1,\ldots,W-3$). Now we apply Corollary 1 to all layers $\ell \in \{1,\ldots,L\}$, yielding that

$$\sigma_\ell \leq \max\left\{ d^+ + \left\lceil \frac{W\varepsilon}{d^+} \right\rceil \varepsilon, W(2\varepsilon - \delta) + d^+ \right\}. \tag{8}$$

Since $\varepsilon \leq d^+/7$, we have $d^- \geq 6d^+/7$ and $\delta \geq 2d^+/7$ and thus $2\varepsilon - \delta \leq 0$; the maximum in Eq. (8) is hence dominated by the first term. This proves the first statement.

Now consider the case where Δ_0 is arbitrary. The bound on σ_ℓ for $\ell \in \{1,\ldots,2W-3\}$ follows from Lemma 4. For $\ell \geq 2W-2$, observe first that we can apply Lemma 3 to all layers $\ell \in \{W-2,\ldots,L\}$. Hence the same bound as in the previous case holds due to Corollary 1 applied to layers $\ell \in \{2W-2,\ldots,L\}$.

The third inequality of the theorem holds since

$$t_{\ell-1,i} - \sigma_{\ell-1} + d^- \leq \min\{t_{\ell-1,i}, t_{\ell-1,i+1}\} + d^- \leq t_{\ell,i} \leq \max\{t_{\ell-1,i}, t_{\ell-1,i+1}\} + d^+ \leq t_{\ell-1,i} + \sigma_{\ell-1} + d^+;$$

the last inequality is proved analogously. □

3.2. Byzantine Faults

We now extend the analysis from the fault-free case to the case of some faulty nodes in the grid. We still confine our examination to a single pulse; we will show later that the necessary preconditions for this type of analysis will eventually be satisfied, no matter what the initial states of the nodes are.

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 within a single pulse, given that all correct nodes have cleared their memory and await the next pulse. If faulty nodes have the possibility to cause a correct node to generate a "false" pulse (i.e., to trigger without the immediate support of other correct 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 correct node (ℓ, i) has a second faulty neighbor (even if it is just a crash fault) and the two faults are not the left and right neighbors. If both faulty neighbors omit to send trigger messages, the node is not going to be triggered. However, if a Byzantine neighbor *does* send a trigger message, at some time after (ℓ, i) has received a trigger message from another neighbor, but before the respective link timeout expires, it is immediately triggered. Hence, a Byzantine node can trigger (ℓ, i) late in this case, again creating a "false" pulse.

Finally, if both the left and right neighbors of correct nodes may fail, we could have every second node in an entire layer failing, which would prevent the propagation of pulses if these nodes do not send messages. With these issues in mind, we arrive at the following sufficient condition for triggering all nodes exactly once per pulse.

Condition 1 (Fault Separation). For each node, no more than one of its incoming links connects to a faulty neighbor.

This condition is equivalent to declaring, for each faulty node, all other nodes that are in-neighbors of some node who has the faulty node as its in-neighbor (i.e., up to 12) as "forbidden region" for additional faults. If we place f faults uniformly at random in a grid of $n := W \times (L+1)$ nodes, the probability that this condition is satisfied for all faulty nodes is bounded from below by $(\binom{n}{f}f!)^{-1}\Pi_{i=0}^{f-1}(n-13i) > (1-13(f-1)/n)^f$. In expectation, a uniformly random subset of $\Theta(\sqrt{n})$ nodes may hence fail before it becomes violated.

We use the following definition to summarize skews.

Definition 4. For $\ell \in [L+1] \setminus \{0\}$, we say that *layer ℓ has skew at most σ* if, for any two correct neighbors (ℓ, i) and (ℓ', i'), $|t_{\ell,i} - t_{\ell',i'}| \leq \sigma$ with $\ell - 1 \leq \ell' \leq \ell$, and layer $\ell - 1$ has skew at most σ. The skew for layer 0 is given by the used clock generation scheme. If layer L has skew at most σ (i.e., any two correct neighbors have skew at most σ) we say that *the pulse has skew at most σ*.

If Condition 1 is satisfied and all correct nodes have cleared all memory flags before a pulse arrives, it is straightforward to derive a (fairly coarse) skew bound.

Lemma 5. *Suppose all correct nodes in layer* 0 *send trigger messages during* $[t_{\min}, t_{\max}]$*, Condition 1 holds, and no correct node in any layer $\ell' \in [\ell+1]$, where $\ell \in [L+1]$, memorizes a trigger message from another correct node or is sleeping at time $t_{\min} + \ell' d^-$. With $f_\ell \leq f$ denoting the number of layers $\ell' \in [\ell]$ containing some faulty node, all correct nodes on layer ℓ are triggered at times within $[t_{\min} + \ell d^-, t_{\max} + (\ell + f_\ell)d^+]$. In particular, the pulse has skew at most $\sigma(f) < t_{\max} - t_{\min} + \varepsilon L + f d^+$.*

Proof. By induction on ℓ. Clearly the statement is true for $\ell = 0$. The step is trivial for the lower bound, since Condition 1 implies that each node needs to receive a trigger message from a correct neighbor to be triggered, which is delayed by at least d^- time. If the upper bound is satisfied for $\ell \in [L]$ and all nodes in layer ℓ are correct, certainly all nodes in layer $\ell + 1$ are triggered within d^+ time. If there is a faulty node in layer ℓ, the upper bound allows for $2d^+$ time for all nodes on layer $\ell + 1$ to be triggered. If a correct node on layer $\ell + 1$ has a faulty neighbor on layer ℓ, Condition 1 necessitates that either its right or its left neighbor is correct and has only correct neighbors on layer ℓ; the claim hence follows, as the node will be left- or right-triggered within $2d^+$ time. □

This lemma shows that even in the presence of multiple faults the time to complete a pulse increases only moderately. Thus, increasing the time between pulses (originating from layer 0) accordingly will maintain a clean separation of pulses.

While Lemma 5 guarantees bounded skew and suggests that actually $\sigma(f) = \sigma_\ell + \mathcal{O}(fd^+)$, where σ_ℓ is the fault-free layer ℓ intra-layer skew given in Theorem 1, a more detailed reasoning is required to prove such a bound. Unfortunately, the number of cases that needs to be considered in a formal proof explodes quickly. A large number of cases needed to be examined already in the fault-free case, and dealing with just a single fault became sufficiently tedious for being relegated to Appendix A. Informally, the reasoning employs the following arguments: For a Byzantine faulty node, there are only two options for increasing the skew between neighbors: (i) "shortcut" a causal path to the fast node and (ii) refrain from triggering nodes to inhibit the propagation of the pulse to the slow node.

Dealing with (i) is straightforward: If during the construction of a causal path we run into a Byzantine node, we follow the *other* incoming causal link of the predecessor node instead, thereby avoiding the Byzantine node, and resume the construction. This is particularly simple for the left zig-zag paths used in Cases 2 and 3 of Lemma 4; see Figure 6 for an example of how this might look like, where the faulty node is located at $(\ell - 1, j_0)$. When dealing with (ii), the situation is similar. Instead of circumventing faulty nodes in the construction of a causal path, we now need to avoid relying on them to trigger correct nodes. Figure 6 also gives an example for this, for a faulty node at $(1, i + 1)$. Consult Appendix A for the detailed discussion of these (and similar) cases.

Consequently, in order to increase skews significantly, *several* faulty nodes need to be in a region that causally affects two neighbors. In a setting where delays are random, it seems unlikely that the elaborate

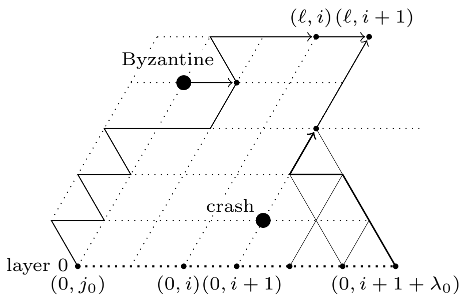


Figure 6: Illustration of alternative causal path construction to bypass a Byzantine node (upper left) and circumventing a crashed (or Byzantine) node when arguing why $(\ell, i+1)$ is triggered in a timely fashion (lower right).

patterns required for large skews will arise, in particular if faults are not in close vicinity of each other. The simulations in Section 4 support this view, showing moderate increase of skews despite a significant number of faults.

We point out that *crash failures*, where nodes simply cease operating, are more benign. Instead of breaking the entire system, two adjacent crash failures on some layer just effectively crash their common neighbor in the layer above and affect the skews of surrounding nodes. We refer to [32] for simulations concerning crash faults.

3.3. Self-Stabilization

Self-stabilization is the ability of the system to recover from an unbounded number of transient faults [18], which may put the system into an arbitrarily erroneous state. When transient faults cease, the system will resume normal operation within a bounded *stabilization time*—ideally even in the presence of a bounded number of persistent faults. In this section, we will show that HEX is self-stabilizing even in the presence of up to f Byzantine faulty nodes that satisfy Condition 1; under the assumption that correct nodes faithfully execute the HEX algorithm, the pulse generation will eventually work as specified in Lemma 5, even when all nodes start from arbitrary internal states. Note that this can also be guaranteed for the pulse generation at layer 0, by using a self-stabilizing and Byzantine fault-tolerant algorithm like FATAL$^+$ [31]; the details are outside the scope of this paper, however.

The analysis in Section 3.1 assumed that $T^-_{\textbf{link}}$, $T^-_{\textbf{sleep}}$, and the time between pulses are sufficiently large for all correct nodes to clear their memory and complete their sleeping period before the next pulse arrives. In the previous section, we argued that faulty nodes have an adverse, but bounded, effect on the skew and the time to complete a pulse. To ensure self-stabilization, we account for this by some additional slack in the time between the k^{th} and $(k+1)^{\text{th}}$ pulses, enabling nodes to reach consistent states even when the initial states are arbitrary. Any choice of parameters thus represents a trade-off between the frequency at which pulses can be issued and the fault-tolerance properties of the system. The following condition provides conservative bounds for the parameters $T^-_{\textbf{link}}, T^+_{\textbf{link}}, T^-_{\textbf{sleep}}$, and $T^+_{\textbf{sleep}}$, as a function of the number of faults and the inaccuracy of the implementation's local time measurements. We stress that, to ensure self-stabilization, the timers must be designed so that they expire within $T^+_{\textbf{sleep}}$ and $T^+_{\textbf{link}}$ time, respectively, even when started from an *arbitrary* internal state.

Condition 2 (Timing Constraints). For $k \in \mathbb{N}$, define

$$t^{(k)}_{\min} := \min_{\substack{i\in[W]\\ (0,i)\text{ correct}}} \left\{ t^{(k)}_{0,i} \right\} \quad \text{and} \quad t^{(k)}_{\max} := \max_{\substack{i\in[W]\\ (0,i)\text{ correct}}} \left\{ t^{(k)}_{0,i} \right\}.$$

An execution of Algorithm 1 has *pulse separation time* $\mathcal{S}$, if, for all $k \in \mathbb{N}$, it holds that $t^{(k+1)}_{\min} \geq t^{(k)}_{\max} + \mathcal{S}$.

For a given number of Byzantine faults f in the grid with *stable skew* $\sigma(f)$, we define

$$\begin{array}{rcl} T^-_{\textbf{link}}(f) & := & \sigma(f)+\varepsilon \\ T^+_{\textbf{link}}(f) & := & \vartheta T^-_{\textbf{link}}(f) \\ T^-_{\textbf{sleep}}(f) & := & 2T^+_{\textbf{link}}(f)+2d^+ \\ T^+_{\textbf{sleep}}(f) & := & \vartheta T^-_{\textbf{sleep}}(f) \\ \mathcal{S}(f) & := & T^-_{\textbf{sleep}}(f)+T^+_{\textbf{sleep}}(f)+\varepsilon L+fd^+. \end{array}$$

Here, $\vartheta \geq 1$ bounds the maximum clock drift, in the sense that $t'-t \leq T'-T \leq \vartheta(t'-t)$ for all real-times $t' \geq t$ with clock readings T', T.

In this definition, the *stable skew* $\sigma(f)$ is meant to be a bound on the skew between *any* two correct neighboring nodes, assuming that the system has already "stabilized", i.e., when the preconditions of Lemma 5 are satisfied for each pulse. This flexibility allows to plug in either a conservative or a more optimistic skew bound $\sigma(f)$ into the stabilization analysis. Note that the constraints $T^-_{\textbf{link}}(f) \geq \sigma(f)+\varepsilon$ and $\mathcal{S}$ being sufficiently large must be satisfied for any realistic $\sigma(f)$, since it is necessary to ensure that nodes do not "forget" a pulse before they are triggered and wake up on time for the next; recall that these assumptions were also implicit in the analysis in Section 3.1.

Before we can cast the algorithm's self-stabilization properties into a theorem, we need to specify what it means for the HEX pulse propagation to have stabilized up to a certain layer.

Definition 5 (Stabilized Pulse Propagation). For an execution of Algorithm 1 on the HEX grid, we say that *layer* $\ell \in [L+1]$ *is stable with skew at most* σ *in pulse* k, if:

- All layers $\ell' \in [\ell]$ are stable with skew at most σ in pulse k;
- Node (ℓ,i), $i \in W$, is not sleeping at time $t^{(k)}_{\min}+\ell d^-$, and it does not memorize any trigger messages from correct neighbors at this time;
- Layer ℓ has skew at most σ in pulse k.

Assuming that all parameters are chosen in accordance with Condition 2 and $\sigma(f)$ is indeed a valid bound on the stable skew, we can now show that the system will recover from arbitrary initial states.

Theorem 2. *Suppose that, given values f and $\sigma(f)$, an execution of Algorithm 1 satisfies the following prerequisites:*

- *There are at most f Byzantine faulty nodes satisfying Condition 1.*
- *The stable skew is at most $\sigma(f)$.*
- *The parameters $T^-_{\textbf{link}}$, $T^+_{\textbf{link}}$, $T^-_{\textbf{sleep}}$, and $T^+_{\textbf{sleep}}$ in Algorithm 1 are chosen in accordance with Condition 2.*
- *The pulse separation time is larger than $\mathcal{S}(f)$, as specified by Condition 2.*

Then, each layer $\ell \in [L+1]$ is stable with skew at most $\sigma(f)$ in all pulses $k > \ell$. Moreover, for each $i \in W$ such that (ℓ,i) is correct and each pulse $k > \ell$, there is a unique triggering time $t^{(k)}_{\ell,i}$ of (ℓ,i) during $[t^{(k)}_{\min}+\ell d^-, t^{(k+1)}_{\min}+\ell d^-)$.

Note that this, in particular, implies that all correct neighbors will satisfy the skew bound $\sigma(f)$ in all pulses $k > L$. In order to prove the theorem, we first show a helper statement saying that if all layers up to layer ℓ are stable in *some* pulse, they will satisfy the claim of the theorem in *all* subsequent pulses.

Lemma 6. *Assume that the preconditions of Theorem 2 are satisfied and that all layers $\ell \in [L+1]$, $\ell' \in [\ell+1]$, are stable with skew at most $\sigma(f)$ in pulse $k \in \mathbb{N}$. Then, these layers are also stable with skew at most $\sigma(f)$ in pulse $k+1$, and for each correct node (ℓ, i), there is a unique triggering time $t_{\ell,i}^{(k)}$ during $[t_{\min}^{(k)} + \ell d^-, t_{\min}^{(k+1)} + \ell d^-)$.*

Proof. Since we have that all layers $\ell' \in [\ell+1]$ are stable, Lemma 5 shows that all nodes in these layers are triggered during $[t_{\min}^{(k)} + \ell' d^-, t_{\max}^{(k)} + (\ell' + f_{\ell'})d^+]$. For each node (ℓ', i) in such a layer, let $t_{\ell',i}^{(k)}$ be the minimal such triggering time (we still need to establish that there is only one). Since the stable skew is at most $\sigma(f)$, for any node (ℓ', i) with $0 \neq \ell' \leq \ell$, the triangle inequality yields that its correct neighbors on layers ℓ' and $\ell'-1$ trigger within $2\sigma(f)$ of each other. Hence, all trigger messages from correct neighbors are received within a time window of duration $2\sigma(f) + \varepsilon$. We have $T_{\mathbf{sleep}}^-(f) > 2T_{\mathbf{link}}^-(f) > 2\sigma(f) + \varepsilon$, implying that nodes will not memorize any late pulse k trigger messages from correct neighbors after waking up. We thus conclude that, for each node (ℓ', i), $t_{\ell',i}^{(k)} < t_{\min}^{(k+1)} + \ell' d^-$ is unique. With $T_{\mathbf{sleep}} := T_{\mathbf{sleep}}^+(f) + T_{\mathbf{sleep}}^-(f)$, our assumptions yield

$$t_{\max}^{(k)} \leq t_{\min}^{(k+1)} - \mathcal{S}(f) < t_{\min}^{(k+1)} - T_{\mathbf{sleep}} - \varepsilon L - f d^+.$$

Hence, the upper bound on the pulse k triggering times of layer ℓ' nodes established in Lemma 5 leads to

$$\begin{aligned} t_{\ell',i}^{(k)} &\leq t_{\max}^{(k)} + (\ell' + f_{\ell'})d^+ \\ &< t_{\min}^{(k+1)} - T_{\mathbf{sleep}} - \varepsilon L - f d^+ + (\ell' + f_{\ell'})d^+ \\ &\leq t_{\min}^{(k+1)} - T_{\mathbf{sleep}} + \ell' d^-. \end{aligned} \tag{9}$$

We thus observe that no node will be sleeping or have memorized any trigger messages from other correct nodes at time $t_{\min}^{(k+1)} + \ell' d^-$. Since we assumed that $\sigma(f)$ is a bound on the stable skew, the requirements of Definition 5 are met for pulse $k+1$ and all layers $\ell' \in [\ell+1]$, as claimed. □

With this lemma, the proof of Theorem 2 boils down to showing that if layer ℓ is stable in pulse k, then layer $\ell+1$ is stable in pulse $k+1$.

Proof of Theorem 2. We prove the theorem by induction on ℓ, where the hypothesis is that the claims of the theorem are satisfied by all layers $\ell' \in [\ell+1]$. For $\ell = 0$, the statement is trivial. For the step from ℓ to $\ell+1$, repeated use of Lemma 6 reveals that it is sufficient to show that layer $\ell+1$ is stable in pulse $\ell+2$.

By the induction hypothesis and Lemma 5, no correct node in layer ℓ sends trigger messages during $[t_{\max}^{(\ell+1)} + (\ell+f)d^+, t_{\min}^{(\ell+2)} + \ell d^-]$. Using exactly the same derivation as for Eq. (9) in the proof of Lemma 6, we find that any triggering message originating in a correct layer ℓ node must have arrived at any correct layer $\ell+1$ node before time $t := t_{\min}^{(\ell+2)} - (T_{\mathbf{sleep}}^+(f) + T_{\mathbf{sleep}}^-(f)) + (\ell+1)d^-$. We will complete the proof by showing that this entails that correct nodes $(\ell+1, i)$ will neither sleep nor memorize trigger messages from correct nodes at time $t_{\min}^{(\ell+2)} + (\ell+1)d^-$.

To this end, suppose that starting from the above time t, nodes on layer $\ell+1$ do not receive trigger messages from correct nodes on the previous layer. By Algorithm 1, nodes will forget messages that arrived more than $T_{\mathbf{link}}^+(f)$ time ago. Thus, the latest time when a correct node $(\ell+1, i)$ in layer $\ell+1$ could be triggered due to a remembered message from some *correct* neighbor on layer ℓ is smaller than $t + T_{\mathbf{link}}^+(f)$.

Hence, consider the case that $(\ell+1, i)$ has a faulty neighbor on layer ℓ. W.l.o.g., assume that the faulty neighbor is node $(\ell, i+1)$ (the other case is symmetric). By the above reasoning, $(\ell+1, i)$ can only be left-triggered at or after time $t + T_{\mathbf{link}}^+(f)$, which in addition requires a memorized trigger message from $(\ell+1, i)$. Thus, if neither $(\ell+1, i)$ nor $(\ell+1, i+1)$ are triggerred during $[t - d^+, t + T_{\mathbf{link}}^+(f))$, neither node can be triggered anymore: Condition 1 guarantees that both nodes have no other faulty neighbor than $(\ell, i+1)$, and the above reasoning applies also to $(\ell+1, i)$.

Therefore, assume that one of them, say $(\ell+1, i+1)$, is triggered at time $t_{\ell+1,i+1} \in [t - d^+, t + T_{\mathbf{link}}^+(f))$. It will not wake up (and therefore not be triggered again) before time $t_{\ell+1,i+1} + T_{\mathbf{sleep}}^-(f)$. Node $(\ell+1, i)$ receives the trigger message from $(\ell+1, i+1)$ by time $t_{\ell+1,i+1} + d^+$ and therefore either (a) forgets it by time

$t_{\ell+1,i+1}+T^+_{\mathbf{link}}(f)+d^+$ or (b) is triggered during $[t_{\ell+1,i+1}, t_{\ell+1,i+1}+T^+_{\mathbf{link}}(f)+d^+)$. If $(\ell+1, i)$ is triggered, its corresponding trigger message to $(\ell+1, i+1)$ arrives by time $t_{\ell+1,i+1}+T^+_{\mathbf{link}}(f)+2d^+ \leq t_{\ell+1,i+1}+T^-_{\mathbf{sleep}}(f)$. This message thus arrives at $(\ell+1, i+1)$ before it wakes up again. When $(\ell+1, i+1)$ wakes up, it clears its memory and will not be re-triggered before $(\ell+1, i)$ is triggered again (or the next pulse arrives on layer ℓ). But $(\ell+1, i)$ cannot be triggered again: If it was not triggered (Case (a)), it forgot any previous trigger messages, and if it was (Case (b)), it clears its memory upon waking up. Consequently, $(\ell+1, i+1)$ cannot be triggered again either.

Overall, no node on layer $\ell+1$ is triggered after time $t_{\ell+1,i+1}+T^+_{\mathbf{link}}(f) \leq t+2T^+_{\mathbf{link}}(f)+d^+$ or receives trigger messages from correct nodes after time $t+2T^+_{\mathbf{link}}(f)+2d^+ \leq t+T^-_{\mathbf{sleep}}(f)$ (until time $t^{(k+1)}_{\min}+(\ell+1)d^-$). Hence, all nodes are awake by time $t+T^-_{\mathbf{sleep}}(f)+T^+_{\mathbf{sleep}}(f) > t+T^-_{\mathbf{sleep}}(f)+T^+_{\mathbf{link}}(f)$ and have forgotten all spurious messages. By the previous observations, this concludes the proof. $\square$

4. Simulation Experiments

As mentioned earlier, the analytic results obtained in the previous sections are limited to worst-case skews. Hence, we conducted extensive simulation experiments to complement them with representative statistical data. This data reveals the following major facts about HEX:

(1) *Average skews considerably smaller than worst-case.* The quite fancy scenarios required for establishing the worst-case skews in Theorem 1 already suggested that they are very unlikely to occur in practice. Indeed, we were not able to generate neighbor skews that came close to the worst case under uniformly and independently distributed link delays.

(2) *Small, localized fault effects.* As argued in Section 3.3, HEX should implicitly confine the effects created by a faulty node to a small neighborhood. Our simulation results revealed that faults typically affect direct neighbors only, and become invisible after one additional hop. This is true even for clustered crash faults and multiple (separated) Byzantine faults.

(3) *Stabilization times much smaller than guaranteed by theoretical analysis.* The layer-wise stabilization used for bounding the worst-case stabilization time in Section 3.3 assumes an involved worst-case scenario *on each layer*, which appears to be very unlikely in practice. Simulations of the original version [33] of the HEX algorithm already revealed typical stabilization times that are much smaller than the predicted worst case. The link timeouts added in Algorithm 1 cause HEX to reliably stabilize within two clock pulses, even in scenarios with multiple faults.

Below, we present selected results supporting these claims. Since we argue about different executions of HEX in this section, for a set R of executions, we denote by $t^{(k)}_{\ell,i,\rho}$ the time when (ℓ, i) forwards the kth pulse in execution $\rho \in R$.

4.1. Simulation Framework

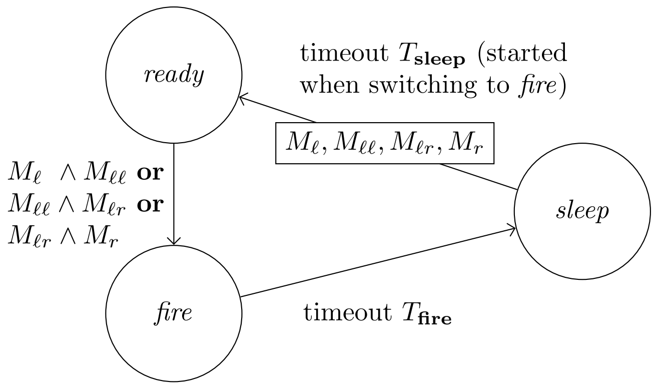
Our simulation framework has been built around Mentor Graphics® ModelSim 10.1d, which allows accurate digital timing simulations of circuit designs specified in VHDL (*Very High Speed Integrated Circuit Hardware Description Language*). It is used to simulate an entire HEX grid, which consists of multiple instances of a VHDL implementation of Algorithm 1 embedded into a custom testbench. The latter has the following purposes:

(1) Set the grid size and instantiate the corresponding number of nodes as well as interconnecting wires.

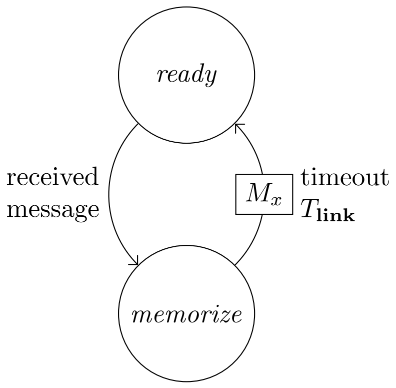
(2) Provide the layer 0 clock sources (i.e., generate the clock pulse of node $(0, i)$, $i \in [W]$) at time $t_{0,i,\rho}$ in simulation run ρ, with some pre-selected skews. Clock sources for a single pulse and for multiple pulses per run are supported.

(3) Control the individual link delays during the simulation. Both random delays (uniform within $[d^-, d^+]$) and deterministic delays are supported.

(4) Control fault injection during the simulation: links can be declared *correct*, *Byzantine* (choose output constant 0 resp. 1 corresponding to no resp. fast triggering), or *fail-silent* (output constant 0); declaring a node Byzantine or fail-silent is equivalent to doing so for each of its outgoing links. The selection


ready
timeout $T_{\mathbf{sleep}}$ (started when switching to *fire*)
$M_\ell, M_{\ell\ell}, M_{\ell r}, M_r$
$M_\ell \wedge M_{\ell\ell}$ **or**
$M_{\ell\ell} \wedge M_{\ell r}$ **or**
$M_{\ell r} \wedge M_r$
sleep
fire
timeout $T_{\mathbf{fire}}$

(a) State machine of the firing algorithm.



(b) State machine for the resettable memory flag M_x.

Figure 7: The state machines 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. The right state machine describes the link timers, while the left state machine implements the firing algorithm as defined by Algorithm 1.

of faulty nodes and/or links can be done deterministically or randomly in different simulation runs ρ, but remains fixed for all pulses generated in a multi-pulse simulation run ρ.

In order to support a flexible evaluation, a custom simulation control and evaluation infrastructure was developed in Haskell [34]. It enables to generate testbenches for different parameter settings (1)–(4), to execute the actual timing simulations, and to post-process the simulation results, which consist of the set of matrices of triggering times $t^{(k)}_{\ell,i,\rho}$ obtained for each pulse k in a simulation run ρ.

The VHDL implementation of Algorithm 1 consists of a few fairly simple design entities only: An asynchronous state machine, a threshold gate, sleep and link timers driven by start/stoppable oscillators, and resettable memory flags. The asynchronous state machine, shown in Figure 7a, consists only of three states, which are visited cyclically: In the (initial) *ready* state, it waits for the trigger condition in Algorithm 1 to become true. The state machine then proceeds to the *firing* state, where it emits a pulse, and then moves into the *sleeping* state. One resettable memory flag per incoming link is used to memorize the occurrence of a pulse from the respective neighbor. It is reset when the state machine takes the transition from the *sleeping* state to *ready*, or when the respective link timer generates a timeout. Figure 7b provides the simple state machine for the memory flag: Starting from the (initial) *ready* state, it waits for the reception of a trigger message, which results in a transition to the state *memorize* and the setting of the corresponding memory flag. After the timeout $T_{\mathbf{link}}$, the memory flag is reset and the state machine returns to the *ready* state.

The implementation of the above state machine is completely asynchronous and has been generated using the Petrify tool [35]; the designs of the timers, start/stoppable oscillators, and memory-flags are based on the designs provided in [36]. The complete HEX node was synthesized with Synopsys® Design Compiler version C-2009.06-SP4, using the UMC 90 nm standard cell library [37]. Note that we had to augment this library by a custom Muller C-Gate [38] developed in the context of the DARTS project [30, 39]. The detailed timing analysis of the HEX node implementation revealed an end-to-end switching delay, without wire delays, in the interval $[0.161, 0.197]$ ns.

Although HEX is, by design, metastability-free in the absence of failures, we cannot rule out the possibility that faulty nodes cause metastable upsets [40] of state-holding devices such as memory flags. However, as laid out in more detail in [36], the probability that a faulty node produces a signal transition within the picosecond-range window of vulnerability is extremely small and can be further decreased by means of synchronizers [11] or elastic pipelines [38]. In our simulations, we may hence safely exclude metastability-inducing behavior.

Using our testbed, we conducted the following types of simulation experiments:

(A) Statistical evaluation of the neighbor skews. These experiments require simulation runs involving only a single pulse propagating through the HEX grid. The primary quantities of interest in simulation run ρ are:

- the (absolute) skews $|t_{\ell,i,\rho} - t_{\ell,i-1,\rho}|$ between neighbors (ℓ, i) and $(\ell, i-1)$ in the same layer ℓ,
- the (signed) skews $t_{\ell,i,\rho} - t_{\ell-1,i,\rho}$ and $t_{\ell,i,\rho} - t_{\ell-1,i+1,\rho}$ of every node (ℓ, i), $\ell > 0$, relative to its direct layer $\ell - 1$ neighbors $(\ell - 1, i)$ resp. $(\ell - 1, i + 1)$.

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.

For $\mathrm{op} \in \{\mathrm{avg}, q_{95}, \max\}$, we define the average, 95%-quantile, and maximum (absolute)

- *layer ℓ intra-layer skew in run ρ*: $\sigma^{\mathrm{op}}_{\ell,\rho} := \mathrm{op}_{i\in[W]}\{|t_{\ell,i,\rho} - t_{\ell,i+1,\rho}|\}$;
- *intra-layer skew in run ρ*: $\sigma^{\mathrm{op}}_{\rho} := \mathrm{op}_{i\in[W],\ell\in[L+1]\setminus\{0\}}\{|t_{\ell,i,\rho} - t_{\ell,i+1,\rho}|\}$;
- *intra-layer skew in simulation set R*: $\sigma^{\mathrm{op}} := \mathrm{op}_{i\in[W],\ell\in[L+1]\setminus\{0\},\rho\in R}\{|t_{\ell,i,\rho} - t_{\ell,i+1,\rho}|\}$.

With $T_{\ell,i,\rho} := \{t_{\ell,i,\rho} - t_{\ell-1,i,\rho}, t_{\ell,i,\rho} - t_{\ell-1,i+1,\rho}\}$, for $\mathrm{op} \in \{\min, q_5, \mathrm{avg}, q_{95}, \max\}$ we define the minimal, 5%-quantile, average, 95%-quantile, and maximum (signed)

- *inter-layer skew between layer ℓ and $\ell - 1$ in run ρ*: $\hat{\sigma}^{\mathrm{op}}_{\ell,\rho} := \mathrm{op}_{i\in[W]}\, T_{\ell,i,\rho}$;
- *inter-layer skew in run ρ*: $\hat{\sigma}^{\mathrm{op}}_{\rho} := \mathrm{op}_{i\in[W],\ell\in[L+1]\setminus\{0\}}\, T_{\ell,i,\rho}$;
- *inter-layer skew in simulation set R*: $\hat{\sigma}^{\mathrm{op}} := \mathrm{op}_{i\in[W],\ell\in[L+1]\setminus\{0\},\rho\in R}\, T_{\ell,i,\rho}$.

(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 attempts to forward a sequence of pulses generated at layer 0 (with bounded skew and a certain separation time). Using post-processing of the recorded triggering times, we compute the stabilization time as the number of pulses needed for the intra- and inter-layer skews to persistently fall below a layer-dependent threshold.

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.

4.2. Simulation Results: Fault-free Case

We conducted a suite of simulations that complement the analytic intra- and inter-layer worst-case skew bounds given in Theorem 1 by statistical data. First, Figure 8 resp. Figure 9 show a 3D plot of a typical pulse propagation wave in a fault-free grid with $L = 50$ and $W = 20$, end-to-end delays in $[7.161, 8.197]$ ns ($\varepsilon = 1.036$ ns), and layer 0 skews all 0 resp. ramping up/down by d^+. The end-to-end delays result from combining the assumed wire and routing delays with the switching delay bounds. The latter were determined by the ModelSim timing analysis of the HEX node to lie within $[0.161, 0.197]$ ns. For the wire and routing delays, we more or less arbitrarily assumed a value within $[7, 8]$ ns. Since the absolute values do not really matter for our simulations, our choice just reflects the facts that (i) communication delays dominate switching delays and (ii) that the delay uncertainty ε is not too large compared to d^- in modern hardware devices. Throughout this section, we hence assume delay values that are uniformly distributed within $[d^-, d^+] = [7.162, 8.197]$ ns. The grid (sliced between width $W - 1$ and $0 \equiv W$ for readability, and truncated to the first 30 layers) lies in the $(\ell \in [L + 1], i \in [W])$ plane and the z-dimension shows the triggering time $t_{\ell,i}$ of the corresponding node (ℓ, i). To further improve readability, 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 the inital skew differences.

Table 1 shows average (σ^{avg}), 95%-quantile ($\sigma^{q_{95}}$), and maximal ($\sigma^{\max}$) intra-layer and minimal $\left(\hat{\sigma}^{\min}\right)$, 5%-quantile ($\hat{\sigma}^{q_5}$), average ($\hat{\sigma}^{\mathrm{avg}}$), 95%-quantile ($\hat{\sigma}^{q_{95}}$), and maximal ($\hat{\sigma}^{\max}$) inter-layer skews, respectively, in the absence of faulty nodes and taken over all nodes and 250 simulation runs, in the setting described above.

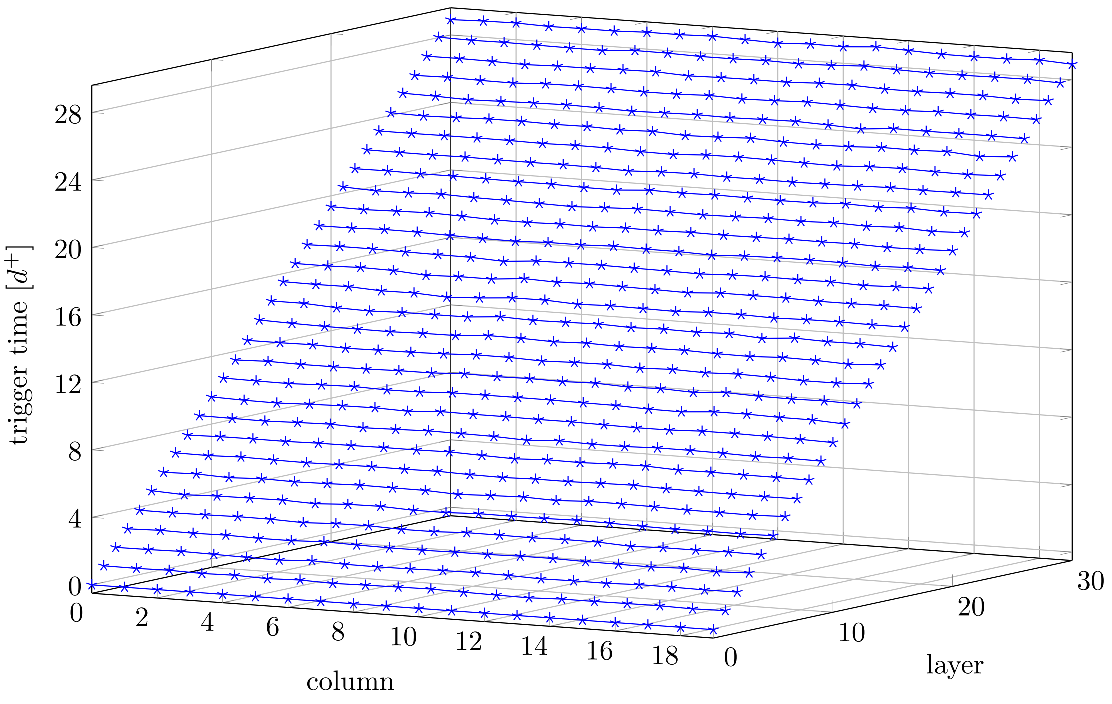


Figure 8: Pulse wave propagation with layer 0 skews of 0.

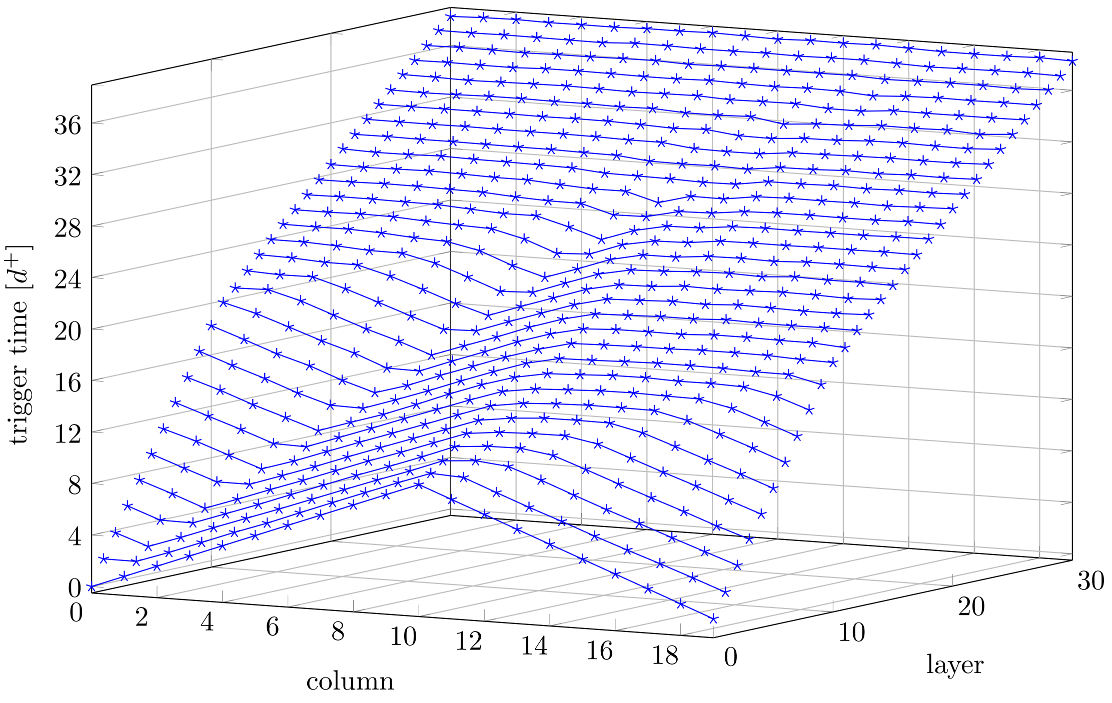


Figure 9: Pulse wave propagation with layer 0 skews ramping up/down by d^+.

Table 1: Intra- and inter-layer skews σ^{op} and $\hat{\sigma}^{\text{op}}$ (in [ns]) in 250 simulation runs on a 50×20 grid.

		intra-layer			inter-layer				
Scenario	initial layer 0 skew	avg	q_{95}	max	min	q_5	avg	q_{95}	max
(i)	0	0.40	1.00	3.10	7.16	7.36	7.94	8.63	11.03
(ii)	random in $[0,d^-]$	0.46	1.23	6.89	7.16	7.35	7.99	8.80	15.20
(iii)	random in $[0,d^+]$	0.47	1.26	7.79	7.16	7.35	8.00	8.81	16.22
(iv)	ramp d^+	1.86	7.64	8.19	0.36	7.26	8.64	14.83	16.39

Table 2: Intra- and inter-layer skews σ^{op} and $\hat{\sigma}^{\text{op}}$ (in [ns]) in 250 simulation runs on a 50×20 grid with a Byzantine node.

		intra-layer			inter-layer				
Scenario	initial layer 0 skews	avg	q_{95}	max	min	q_5	avg	q_{95}	max
(i)	0	0.54	1.34	10.39	5.58	7.35	8.01	8.76	17.55
(ii)	random in $[0,d^-]$	0.61	1.72	10.12	4.21	7.34	8.06	9.00	20.03
(iii)	random in $[0,d^+]$	0.62	1.79	10.36	3.52	7.34	8.07	9.03	20.72
(iv)	ramp d^+	1.97	7.66	34.59	-19.70	7.26	8.69	14.87	24.31

Four different choices for the layer 0 skews between neighbors were used: 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^+: $t_{0,i+1} = t_{0,i} + d^+$ 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 $\Delta_0 \approx W\varepsilon/2 = 10.36$. 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.

Inspecting Table 1 reveals that not a single instance in the collected data showed a skew $\sigma^{\max} > d^+ = 8.197$ ns resp. $\hat{\sigma}^{\max} > 2d^+ = 16.394$ ns. In scenarios (i) to (iii), $\hat{\sigma}^{\min} \approx d^-$, 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). A comparison with the worst-case results of Theorem 1, which bound $\sigma^{\max} \leq 21.63$ ns and $[\hat{\sigma}^{\min}, \hat{\sigma}^{\max}] \subseteq [-14.47, 29.83]$ ns for scenarios (i) and (ii), reveals a much better typical skew in every scenario.

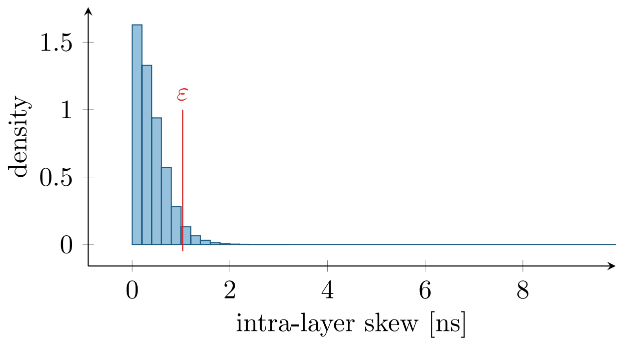
The histograms of the skew distributions in scenario (i) are shown in Figure 10; scenarios (ii) and (iii) look similar. One observes a sharp concentration with an exponential tail. Only in scenario (iv), as already indicated by the large values of q_{95} in Table 1, there is a visible cluster near the end of the tail that is again caused by the large initial skews, see Figure 11.

Given the considerable differences between the minimum ($\hat{\sigma}^{\min}$) and maximum ($\hat{\sigma}^{\max}$) inter-layer skew in Table 1, in conjunction with its non-zero bias, the question of layer-dependence arises. Figure 12 provides $\hat{\sigma}_\ell^{\min}$, $\hat{\sigma}_\ell^{\text{avg}}$ and $\hat{\sigma}_\ell^{\max}$, along with their standard deviations, over the layers $\ell \in [30] \setminus \{0\}$ in case of scenario (iii) resp. (iv). The diagrams reveal that the fairly discrepant skews observed in lower layers start to smooth out after layer $W - 2$, in accordance with Lemma 3, which shows that the behavior observed in Figure 9 is very typical. To avoid cluttering the diagrams, we omitted $\hat{\sigma}_\ell^{q_5}$ and $\hat{\sigma}_\ell^{q_{95}}$, which are close to $\hat{\sigma}_\ell^{\min}$ resp. $\hat{\sigma}_\ell^{\max}$.

4.3. Simulation Results: Failures

Next, we back up the results of our analysis in Section 3.3: We consider f uniformly placed failures under the constraint that Condition 1 holds. In each run, each Byzantine node randomly selects its behavior on each outgoing link as either constant 0 (fail-silent) or constant 1. A typical pulse with a single faulty node $(1, 19)$, marked as a red dot, is shown in Figure 13; Figure 14 depicts a sample scenario with 5 Byzantine faulty nodes.

In Table 2, we list the statistical results for $f = 1$. One observes that the behavior for scenarios (i) to (iii) is very similar, hence we will showcase the main points by examining scenario (iii) and (iv) more closely.



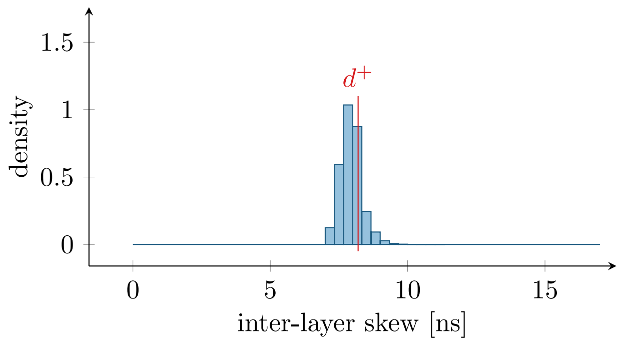
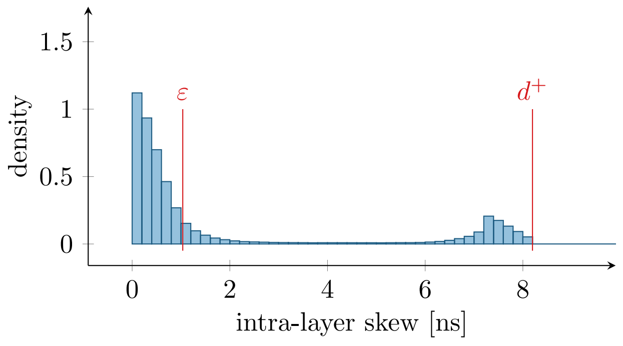


Figure 10: Cumulated skew histograms, from 250 simulation runs in scenario (i).



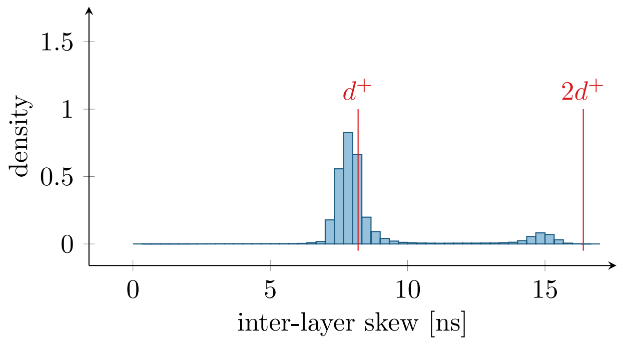
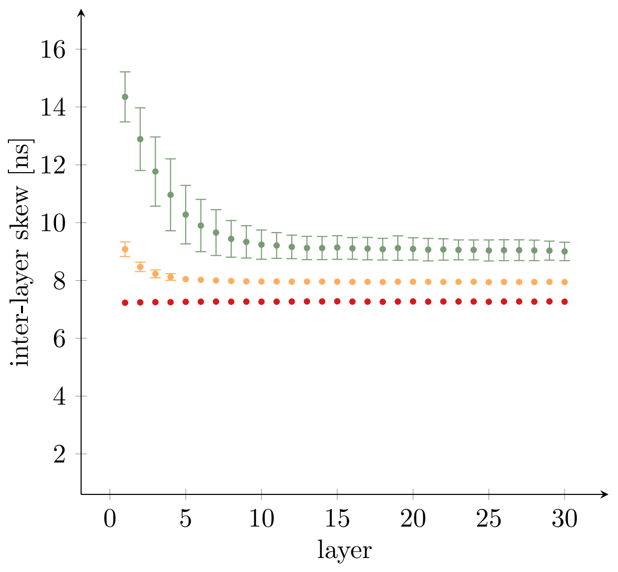


Figure 11: Cumulated skew histograms, from 250 simulation runs in scenario (iv).



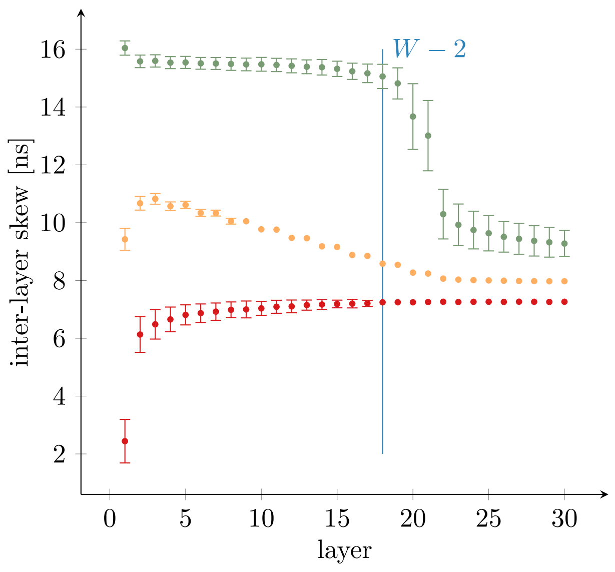


Figure 12: Visualization of the inter-layer skews in scenario (iii) and (iv), truncated to 30 layers. Averages and standard deviations over 250 simulation runs are plotted on a per-layer basis, to avoid clutter the standard deviation was dropped if it was below 0.1. The top data series (●) shows $\hat{\sigma}_\ell^{\max}$, the middle (●) $\hat{\sigma}_\ell^{\text{avg}}$, and the lower (●) $\hat{\sigma}_\ell^{\min}$. Note that the reduction of the maximal skew after layer $W-2$ in the scenario (iv) is in accordance with the vanishing initial skew dependence predicted by Lemma 3.

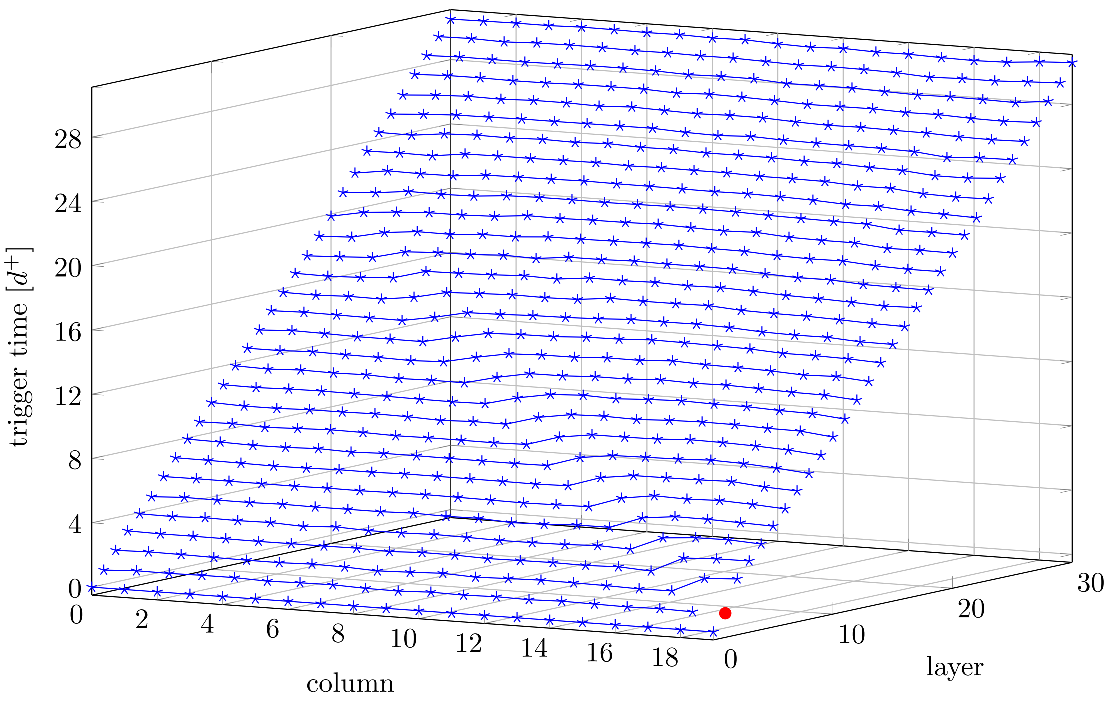


Figure 13: Pulse propagation for scenario (i), with one Byzantine node (marked red) at $(1, 19)$. It sends a constant 1 to its left and right neighbors, and a constant 0 to both upper-layer neighbors. Observe how the increase in skews emanating from the faulty node fades with the distance from the fault location.

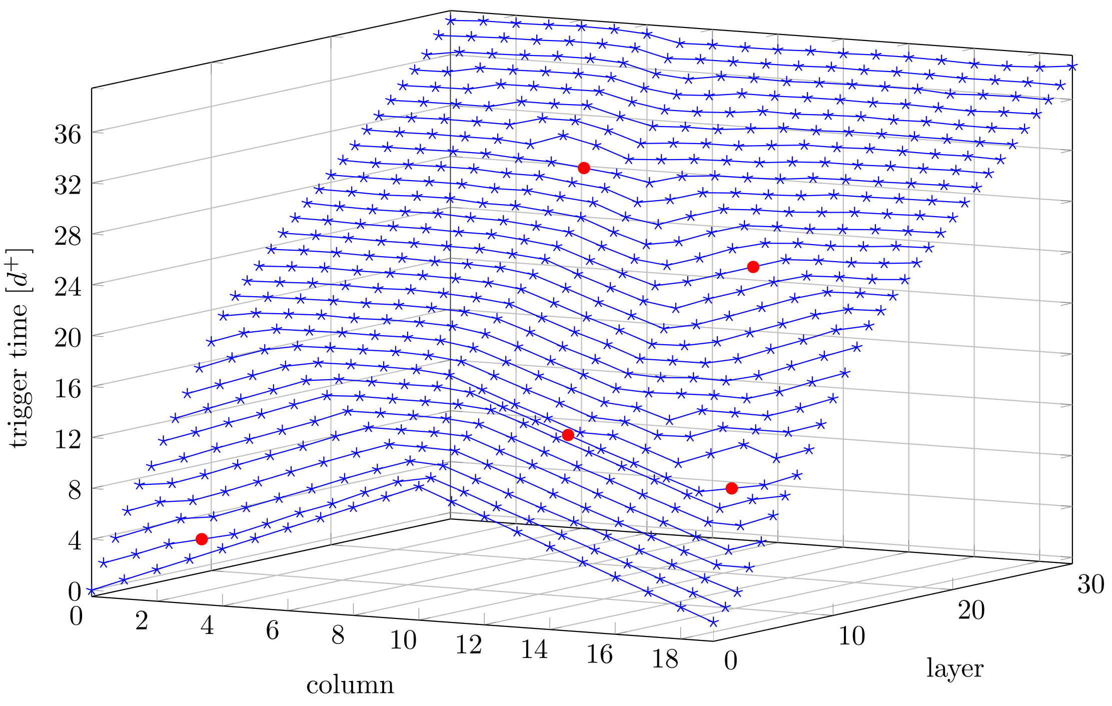


Figure 14: Pulse propagation for scenario (iv), with five Byzantine nodes (marked red).

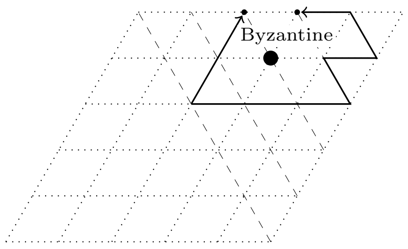


Figure 15: A single Byzantine node in scenario (iv) maximizes the skew between its upper neighbors. All delays are assumed to be d^+ and the triggering times of the nodes in the smallest layer are increasing from left to right, by d^+ per hop. Hence, in absence of faults, all nodes on the diagonals in the left-up direction would be triggered at identical times. The marked causal paths originating at the same node illustrate that the generated skew is $5d^+$; the inter-layer skew is smaller by d^+.

Table 3: Assumed stable skews σ and corresponding timeout values (in [ns]) used in stabilization experiments.

Scenario	initial layer 0 skews	σ	T^-_{link}	T^+_{link}	T^-_{sleep}	T^+_{sleep}	$\mathcal{S}$
(i)	0	28.48	31.98	33.58	83.56	87.74	264.08
(ii)	random in $[0, d^-]$	31.16	34.66	36.39	89.18	93.64	275.60
(iii)	random in $[0, d^+]$	31.75	35.25	37.01	90.42	94.94	278.14
(iv)	ramp d^+	40.64	44.14	46.34	109.08	114.53	316.40

In Figures 16a resp. 16c, we show box-plots of minimum, 5%-quantile, average, 95%-quantile, and maximum intra- resp. inter-layer skews from 250 runs with $f \in [6]$ faults for scenario (iii).[9] A comparison of values $f > 0$ with $f = 0$ reveals that skews increase moderately. In particular, the skews increase substantially slower than the derived worst-case bound of roughly $5fd^+$. Furthermore, Figures 16b and 16d show the same data, except that, in addition to the faulty nodes themselves, also their outgoing 1-hop neighbors are discarded from the data set ($h = 1$). Here, a comparison of the case $f = 0$ and any $f > 0$ reveals that all fault effects have essentially disappeared or are mitigated notably, which confirms that HEX exhibits strong fault-locality. Concerning fail-silent nodes, all results are qualitatively similar, albeit with smaller skews.

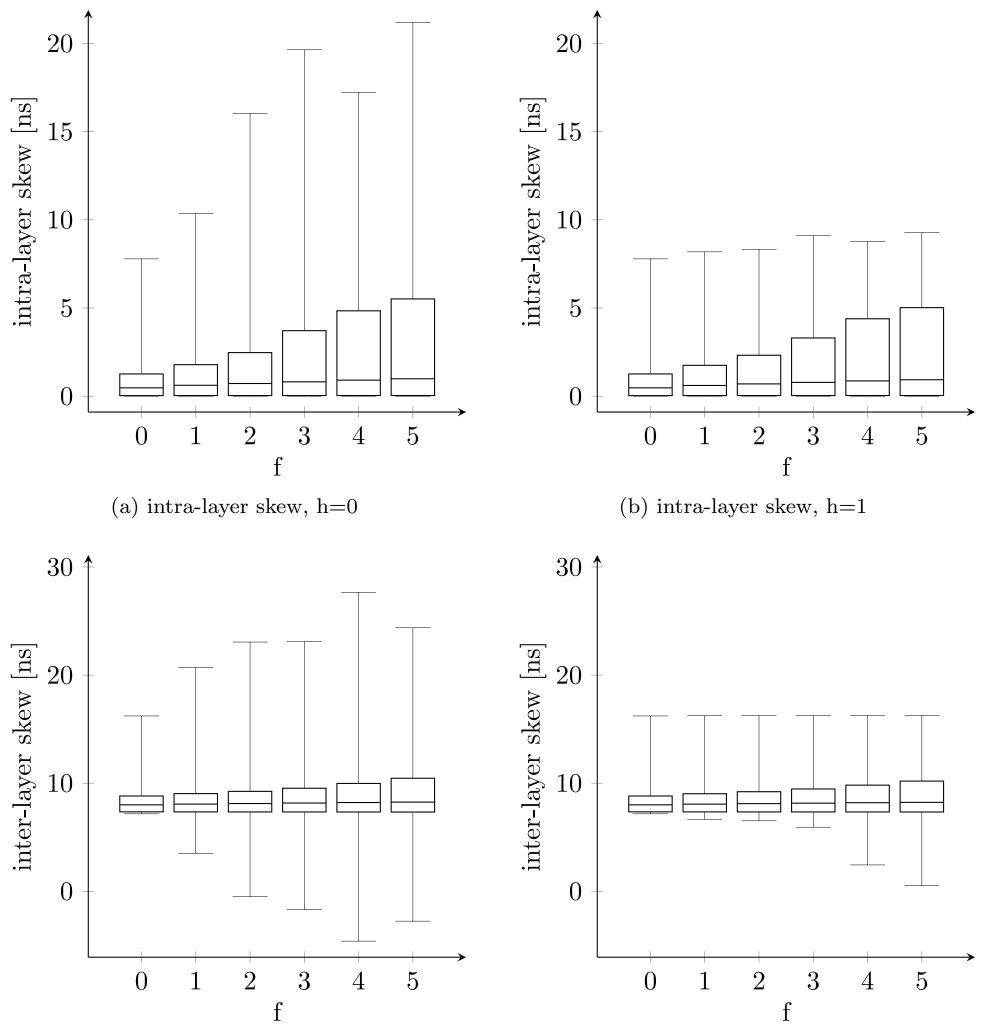
Figure 17 gives the same plots for scenario (iv). Apart from the expected increase in skews, we observe two points worth mentioning. First, a single fault essentially causes the "worst-case" skew. This demonstrates that, as already indicated by scenario (iii), skew effects of multiple faults do not accumulate, or do so in a very limited way. Second, the maximal intra-layer skews typically *exceed* the inter-layer skews. An explanation for this behavior is provided in Figure 15: Intuitively, for "ramped" triggering times generated at layer 0, the pulse propagates "diagonally" (cf. Figure 9) instead of "vertically" (= layer by layer). This limits the power of the HEX grid to mitigate Byzantine behavior, as it is implicitly optimized for propagating pulse waves vertically.

4.4. Simulation Results: Self-stabilization

We now present the results of the multi-pulse simulations conducted for evaluating stabilization time statistics. The same scenarios as in our single-pulse experiments were used here. For the timeouts and the pulse separation time $\mathcal{S}$, we used nominal values that are compatible with the (scenario-dependent) maximum skew observed for $f \in [6]$ Byzantine or fail-silent nodes. These skews where determined via the previous simulations, plus a slack of d^+ accounting for the fact that we work with a statistical sample showing an exponential tail. Timeouts and pulse separation times were computed according to (a modified[10] version of) Condition 2, assuming $\vartheta = 1.05$. The results are shown in Table 3.

[9] Note that for the absolute values of the intra-layer skew, the minimal and 5%-quantile values are close to zero and, thus, of little relevance.

[10] Condition 2 does not take into account that triggering signals in our HEX implementation have non-zero duration, resulting in slightly increased values.



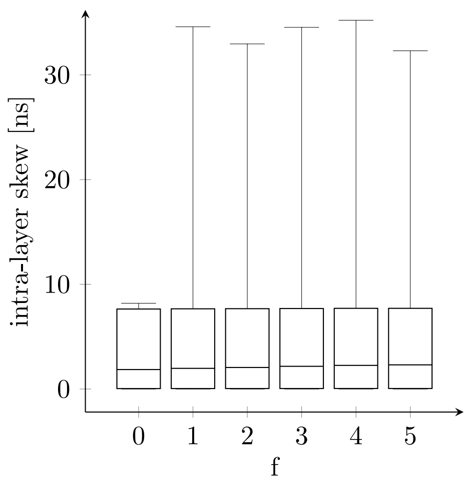
(a) intra-layer skew, h=0 (b) intra-layer skew, h=1

(c) inter-layer skew, h=0 (d) inter-layer skew, h=1

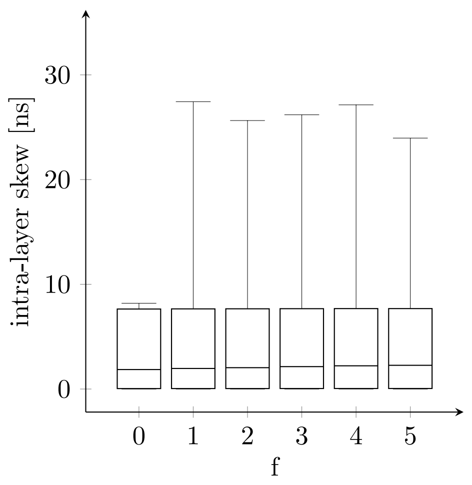
Figure 16: Box plots of inter-layer and intra-layer skews $\hat{\sigma}^{\mathrm{op}}$ from 250 runs in scenario (iii), with h-hop outgoing neighbors of the f Byzantine faulty nodes removed.

For each scenario, fault-number, and fault-type, we executed 250 simulation runs. For each run, f faulty nodes were placed uniformly at random under the constraint that Condition 1 held. Then, starting with all non-faulty nodes in random initial states, 10 consecutive pulses were generated, where placement and behavior of faulty nodes remained fixed during an individual run. For each run ρ, the firing times $t_{\ell,i,\rho}^{(k)}$ of all pulses $1 \leq k \leq 10$ were recorded; thanks to the large pulse separation times, unambiguously assigning a corresponding pulse number to a triggering time (after initial spurious triggering events ceased) was easy.

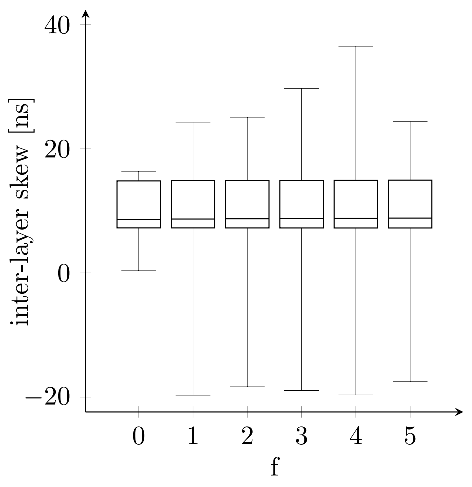
Due to the fact that our simulations only cover the first 10 pulses, it could occur that some runs do not stabilize (yet). Even worse, we cannot rule out the possibility that a run which seems stable violates the skew bounds in some later pulse. However, our actual experiments showed that non-stabilizing runs occur only in scenarios in which the a priori chosen skew bound $\sigma(f, \ell)$ was smaller than the maximal skews reported in Table 2, i.e., where $\sigma(f, \ell)$ was too small. We are hence confident that our stabilization time estimates can be considered representative for realistic skew bounds.



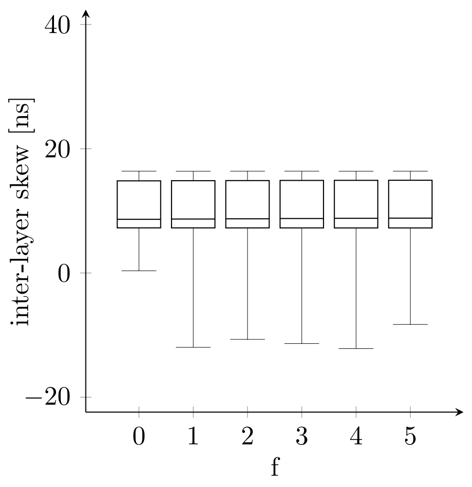
(a) intra-layer skew, h=0



(b) intra-layer skew, h=1



(c) inter-layer skew, h=0

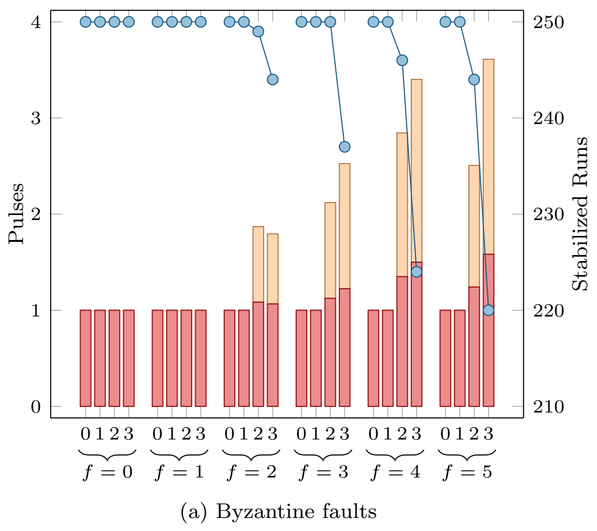


(d) inter-layer skew, h=1

Figure 17: Box plots of inter-layer and intra-layer skews $\hat{\sigma}^{\mathrm{op}}$ from 250 runs in scenario (iv), with h-hop outgoing neighbors of the f Byzantine faulty nodes removed.

Our stabilization time estimate for each run is computed (off-line) as the minimal pulse k with the property that the maximal layer ℓ intra- resp. inter-layer skew, for every layer $\ell \in [L+1]$, is below the a priori chosen skew bound $\hat{\sigma}(f,\ell)$ resp. $\sigma(f,\ell)$. As the former directly depends on the latter (cf. Theorem 1), we used 4 different choices $C \in \{0,1,2,3\}$ for the skew bound, obtained by setting $\sigma(f,\ell) = (4-C)d^+$ for $C \in \{1,2,3\}$ resp. the very conservative skew bounds resulting from Lemma 5 for $C = 0$.

Figure 18a resp. Figure 19a shows both the average and the average + standard deviation of the stabilization times for scenario (iii) resp. (iv) for Byzantine faults. Similarly, Figure 18b resp. Figure 19b shows scenario (iii) resp. (iv) in case of fail-silent faults. By and large, unless C is chosen aggressively large, resulting in too small $\sigma(f,\ell)$, HEX usually stabilizes after the very first pulse. For large C, the average stabilization time estimates go up moderately. Given that the number of simulation runs that did not stabilize within 10 pulses was low ($< 25\%$ even in the most unfavorable scenario), however, we can safely infer that the typical stabilization time of HEX is indeed much lower than the worst-case stabilization time bound $L+1$ established in Theorem 2. The results verify that the already good self-stabilization properties of the original HEX algorithm reported in [32] are further improved by the additional link timeouts.



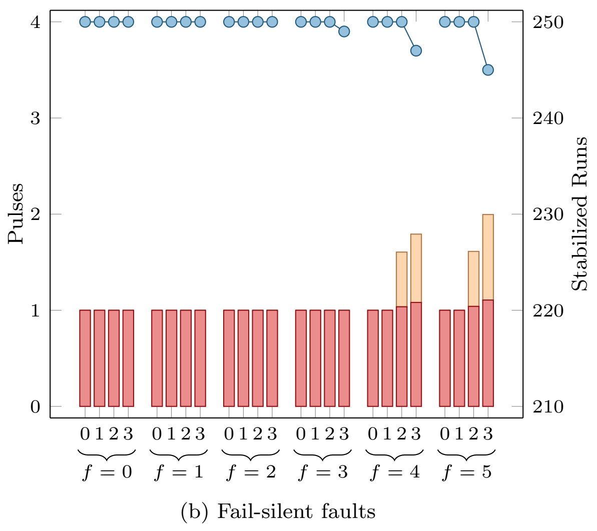
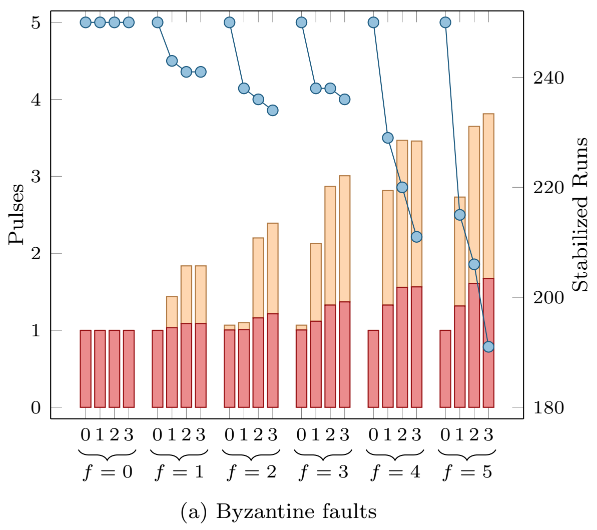


Figure 18: Plot of the average (🟥), the average + standard deviation (🟧) of the estimated stabilization time and the number of stabilized runs (—●—), under scenario (iii) in 250 multi-pulse simulations. For each number f of faulty nodes, $C \in \{0, \dots, 3\}$ encodes the (decreasing) choice of $\sigma(f, \ell)$.



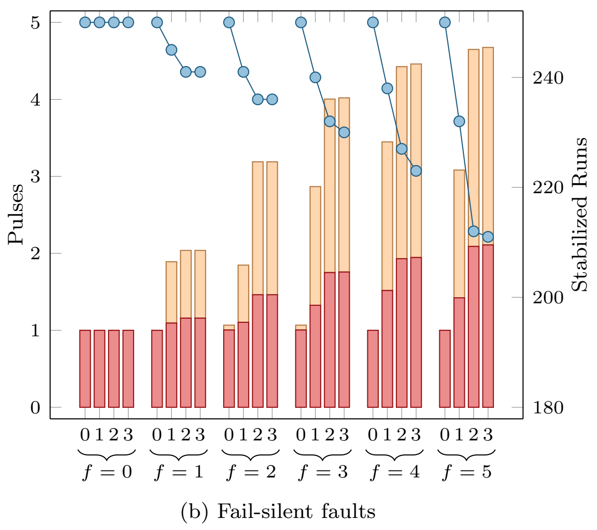


Figure 19: Plot of the average (🟥), the average + standard deviation (🟧) of the estimated stabilization time and the number of stabilized runs (—●—), under scenario (iv) in 250 multi-pulse simulations. For each number f of faulty nodes, $C \in \{0, \dots, 3\}$ encodes the (decreasing) choice of $\sigma(f, \ell)$.

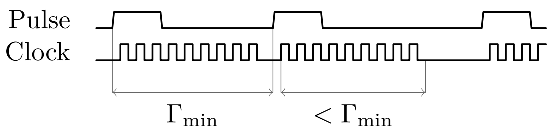


Figure 20: A pulse generated by HEX will result in a fixed number of clock cycles being generated. The number of clock cycles and their frequency has to be chosen so that the required time span for them is less than the minimal pulse separation time at correct nodes ($\Gamma_{\min}$).

Finally, we also computed the stabilization time estimates for the very same scenarios after also discarding the single-hop outgoing neighbors of faulty nodes ($h = 1$). In this case, HEX turned out to stabilize after the very first pulse in every run. This again confirms the strong fault locality property claimed for the HEX grid.

5. Discussion

Stabilization Time and Pulse Separation Time. The bound on $\mathcal{S}$ given by Condition 2 and Theorem 2 is not a large factor from being optimal: The stable skew σ, which clearly is a lower bound on $\mathcal{S}$, is at least roughly $2d^+$. Our values of $\mathcal{S}$ are at most roughly 10 times larger; by rule of thumb, we estimate that $\mathcal{S}$ cannot be reduced by more than roughly factor 2. Given that the used values of $\mathcal{S}$ guarantee stabilization within less than a microsecond, we consider increasing the (effective) frequency at which the system can be run the most important issue related to the pulse separation time: given that $\mathcal{S} > 100\,\text{ns}$, a naïve use of HEX as a clocking system results in fairly low operational frequencies. There are several remedies to increase the frequency of the resulting clocks, however.

First, one can try to decrease d^+, ε, and other parameters that affect skews and delays by engineering efforts. The amazing improvements that clock tree engineering achieved over the decades serve to demonstrate that there is great potential for improving HEX via this approach. Second, one may use frequency multiplication to derive faster clocks from less frequent HEX pulses. Last but not least, one can seek to improve HEX from an algorithmic point of view. We now briefly discuss the latter two options.

Frequency Multiplication. In [32], we presented a scheme that synchronizes local high-frequency start/stoppable oscillators to the HEX pulses. Each node's oscillator generates a high-frequency *fast clock* signal for less than the *minimum pulse separation time* $\Gamma_{\min}$, which has to be less than the pulse separation time $t_{\ell,i}^{(k+1)} - t_{\ell,i}^{(k)}$, for every node (ℓ, i) and every pulse k. Having the high-frequency oscillators generate pulses only in this restricted time window ensures a metastability-free restart of the oscillator with the next pulse (cf. Figure 20). Note that the pulse separation time may vary with each pulse at node (ℓ, i), as it depends on $\mathcal{S}$, the node's position in the HEX grid, the delay uncertainties from layer 0 to L, the initial layer 0 skews etc. It is of course beneficial in terms of achieving a stable amortized frequency of the fast clocks if $\Gamma_{\min}$ is large compared to the variability of the pulse separation times.

On the other hand, the achievable worst-case skew of the fast clock between neighbors turned out to be equal to the HEX clock skew plus an additive term of roughly $\varrho\Gamma_{\min}$, where $\varrho = \vartheta - 1$ is the relative drift of the oscillators. This deterioration of the achievable skew prohibits to make $\Gamma_{\min}$ arbitrarily large. Since $\vartheta < 1.05$ for practical oscillators, however, the skew of the HEX pulses will usually dominate the overall skew.

In any case, the performance of the frequency multiplication depends strongly on the engineering parameters d^+, ϑ and ε as well as on Δ_0. Sufficiently good parameters, in combination with stable low frequency pulses generated by HEX, are hence a prerequisite for achieving stable fast clocks with a small neighbor skew.

Decreasing Skews further. In the fault-free case, we observed intra-layer skews of roughly d^+. The inter-layer skews, on the other hand, have a higher bound but have a bias of $s \approx d^+$. Whereas one can compensate this

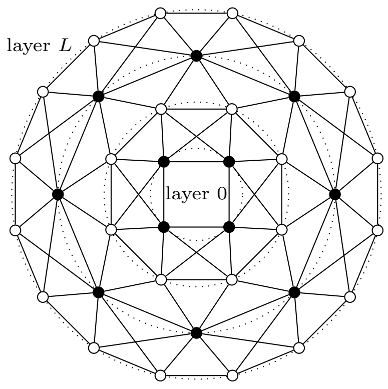


Figure 21: Alternative topology. White nodes are in doubling layers. Doubling layers become less frequent with increasing distance from the center.

bias by subtracting s at the application-level, as discussed in [32], which reduces the skew appropriately, this cannot mitigate the increase of the skew by a factor 2 or more even in the presence of few faults (cp. Figure 16): The reason is that the HEX nodes rely on their neighbors in the same layer to "help out" if one of their neighbors in the previous layer is faulty. This incurs an additional signal delay and therefore a substantial increase of the skew. This problem would be mitigated, if not eliminated entirely, via augmenting the HEX topology by connecting each node to additional in-neighbors from the previous layer. This would, through clock multiplication, allow to increase the effective system frequency and also reduce the stabilization time.

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" 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 due to physical distance.

As a remedy, a slightly modified topology that arranges the nodes of each layer in a circular pattern could be used. To avoid large variations in link lengths, "doubling layers" responsible for "duplicating" the nodes of a standard layer to quickly increase the number of nodes are used; see Figure 21 for an illustration. The resulting topology can be easily embedded into two interconnection layers with little distortion. The analysis from Section 3 provides strong evidence that the resulting skews would not be worse (if not better); conducting the detailed analysis of such variants of HEX is a topic of future research, however.

6. Conclusions

We proposed a candidate for a scalable and fault-tolerant alternative for clock distribution in VLSI circuits, multi-core processors, and other hardware devices. Our approach supports multiple synchronized clock sources and is self-stabilizing even in the presence of a large number of non-clustered Byzantine failures of both clock sources and HEX nodes. Theoretical worst-case analysis and elaborate simulation experiments have been used to show that HEX guarantees a small skew between neighbors in the grid.

Part of our current/future work on HEX is to explore the properties of alternative topologies, which may be chosen to improve clock skews and/or resilience. Moreover, we are working on self-stabilizing Byzantine fault-tolerant higher-level services that may benefit from the enticing properties of HEX.

Appendix A. Skew Analysis with one Fault

Recall that the various cases encountered in our analysis (i.e., mainly in Lemma 4) are always tackled by the following generic pattern:

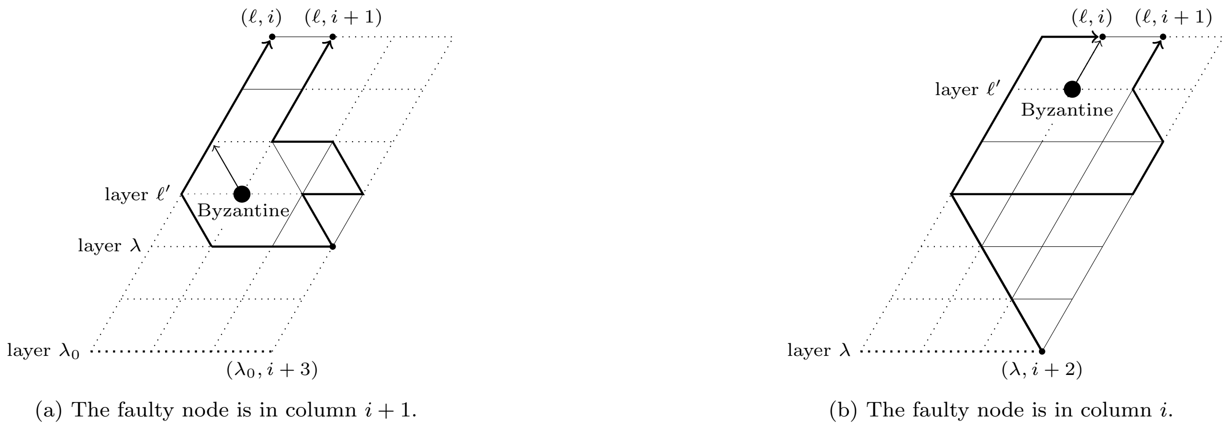


Figure A.22: How to bypass a Byzantine faulty node in Case 1. In each case, the thickly drawn paths are the constructed causal path to (ℓ, i) (on the left side) and a longest possible causal path to $(\ell, i+1)$. In both cases, we assume that the Byzantine node would be part of the regular causal path construction to (ℓ, i), indicated by the respective edges. The remaining non-dotted edges guarantee that the nodes on the long path to $(\ell, i+1)$ will be triggered in a timely fashion.

- Consider a causal path (typically a left zig-zag path) leading to a node (ℓ, i) on some layer ℓ ("fast node"), which is supposed to be triggered much earlier than its neighbor $(\ell, i+1)$ on the same layer ("slow node").
- Derive a lower bound on the triggering time of the fast node in relation to the triggering times of the nodes on the causal path, from the length of the path and the fact that delays are at least d^-.
- Derive an upper bound on the triggering time of the slow node, from the bounds on the triggering times of the nodes on the causal path, the fact that delays are at most d^+, and bounds on the skew in some previous layers.

Consequently, there are essentially two options for a faulty node to increase skews between neighbors: (i) "shortcut" a causal path to the fast node and (ii) refrain from triggering nodes to inhibit the propagation of the pulse to the slow node. In this section, we will discuss how to handle these possibilities for all steps of our analysis, in the case of a single fault. We will see that the bounds are affected by at most $\mathcal{O}(d^+)$, no matter where the fault is located and how it behaves.

We will follow the general structure of the proof of Lemma 4 in terms of the discussed cases, where we also generalize Lemma 2 once it is needed. It is straightforward to see that similar arguments apply to Lemma 3 and Corollary 1, so we will omit these from the discussion.

Case 1 V-shaped skews (Figure 4a): $t_{\lambda,i+1} \leq t_{\lambda,i} + d^+$ *for some maximal* $\lambda \geq \lambda_0$

Case 1 implicitly uses a simple causal path construction, which we now need to make explicit. Since the case considers columns i and $i+1$ only, we distinguish two subcases in which we need to modify the approach.

Case 1a: The faulty node is $(\ell', i+1)$ *for some* $\ell' \in \{\lambda_0, \lambda_0+1, \ldots, \ell-1\}$. We backtrace the "rightmost" causal (not necessarily zig-zag) path ending in (ℓ, i), i.e., if the current source of the path is right-triggered, the incoming link from the right neighbor is added; if it is centrally triggered, the link from the lower right neighbor; otherwise, the one from the lower left neighbor. Several possibilities exist:

(1) We reach column $i+1$ in a layer $\lambda > \ell'$. In this case, either $t_{\lambda,i+1} \leq t_{\lambda,i} + d^+$ (if $((\lambda, i+1), (\lambda, i))$ is in the causal path) or $t_{\lambda+1,i+1} \leq t_{\lambda+1,i} + d^+$ (if $((\lambda, i+1), (\lambda+1, i))$ is in the causal path). We can hence proceed exactly as in the fault-free setting in Case 1 of Lemma 4.

(2) The causal path reaches the faulty node, i.e., we would add one of the links $((\ell', i+1), (\ell'+1, i))$ or $((\ell', i+1), (\ell', i))$. In the first case, $(\ell'+1, i)$ must have been centrally triggered. We add the link $((\ell', i), (\ell'+1, i))$ instead and resume the construction. In the second case, (ℓ', i) must have been right-triggered. We add the link $((\ell'-1, i+1), (\ell', i))$ instead and resume the construction, which will eventually terminates in one of the following two ways:

    (2a) We reach column $i+3$ at some layer $\lambda \geq \lambda_0$ (see Figure A.22a for an example).

    (2b) We reach layer $\lambda_0 - 1$ before we reach column $i+4$.

Because of (2), we can be sure that the faulty node is not on the constructed path. For actually bounding the triggering time of the fast node (ℓ, i), we conservatively assume for simplicity[11] that all delays on the causal path are exactly d^- and the nodes are triggered when receiving the message from their predecessor.

Regarding (2a), we know that the causal path to (ℓ, i) originates in some layer $\lambda \geq \lambda_0$. By the same induction as used in Lemma 2, one can show that every node on the k^{th} left-diagonal on or to the right of the causal path has a triggering time that is at most kd^+ more than the maximal triggering time of any node on the causal path on diagonal 0. In fact, since the causal path is not necessarily a triangular left zig-zag path here (see Figure A.22a), the induction itself works only for nodes on or to the right of the causal path which are also above horizontal segments of the causal path. However, the direct proof that establishes the induction basis also shows that the assertion trivially holds for nodes on these horizontal segments also.

For the, in terms of the lower bound on $t_{\ell,i}$, worst-case causal path, which goes from $(\lambda, i+3)$ via $(\lambda, i+1)$ and $(\lambda+1, i)$ to (ℓ', i), the induction starts at the left-diagonal $(\lambda, i+1)$ to $(\lambda+1, i)$ with maximal triggering time $t_{\lambda+1,i}$. Observe that node $(\ell', i+3)$ is $\ell' - \lambda + 2$ diagonals from, e.g., node $(\lambda, i+1)$. It hence follows that all correct nodes in columns $i+1$, $i+2$, and $i+3$ in layers λ to ℓ' that are on or to the right of the causal path—in particular, nodes $(\ell', i+2)$ resp. $(\ell', i+3)$—will be triggered by time $t_{\lambda+1,i} + (\ell' - \lambda + 1)d^+$ resp. $t_{\lambda+1,i} + (\ell' - \lambda + 2)d^+$. Consequently, $(\ell'+1, i+1)$ will be triggered by time $t_{\lambda+1,i} + (\ell' - \lambda + 4)d^+$, and it follows that $t_{\ell,i+1} \leq t_{\lambda+1,i} + (\ell - \lambda + 3)d^+$. Since we have that $t_{\ell,i} = t_{\lambda+1,i} + (\ell - \lambda - 1)d^-$, comparison with Case 1 of Lemma 4 reveals that the resulting skew is $O(d^+)$ larger, so this case is covered.

To deal with (2b), we can argue similarly, provided that, for some layer $\lambda \geq \lambda_0$, nodes $(\lambda, i+1)$, $(\lambda, i+2)$, and $(\lambda, i+3)$ are triggered before or shortly after (λ, i), i.e., that $t_{\lambda,i+k} \leq t_{\lambda,i+k-1} + d^+$ for $1 \leq k \leq 3$. If this holds true, we are essentially in Case 1 of Lemma 4, except that the causal path leading to (ℓ, i) resp. to $(\ell, i+1)$ need not start at (λ, i) resp. $(\lambda, i+1)$ but rather at $(\lambda, i+k-1)$ resp. $(\lambda, i+k)$ for some $1 \leq k \leq 3$. Note that this shift results from the need to avoid the faulty node in the causal path construction; as in (2a), it changes the resulting skew by at most $O(d^+)$.

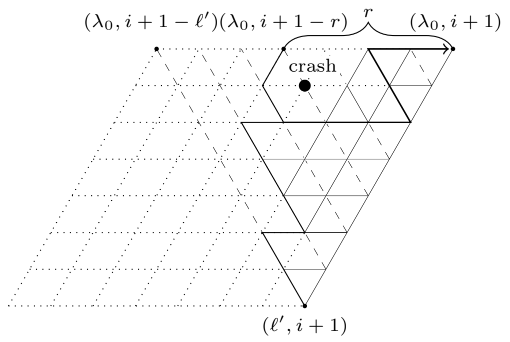
Still, there is also the possibility that there is no $\lambda \geq \lambda_0$ where $t_{\lambda,i+k} \leq t_{\lambda,i+k-1} + d^+$ for $1 \leq k \leq 3$ holds. We will deal with these cases (which correspond to Case 2 and 3 in Lemma 4) below.

Case 1b: The faulty node is (ℓ', i) for some $\ell' \in \{\lambda_0, \lambda_0 + 1, \ldots, \ell - 1\}$. In this event, we use essentially the same approach as for Case 1a, except that we shift it one column to the left: Now the construction of the causal path might encounter a left-triggered node whose lower-left neighbor is the faulty node, forcing the construction to evade to column $i-1$; in turn, column $i+3$ is not needed any more to circumvent the faulty node when deriving an upper bound on $t_{\ell,i+1}$. See Figure A.22b for an example.

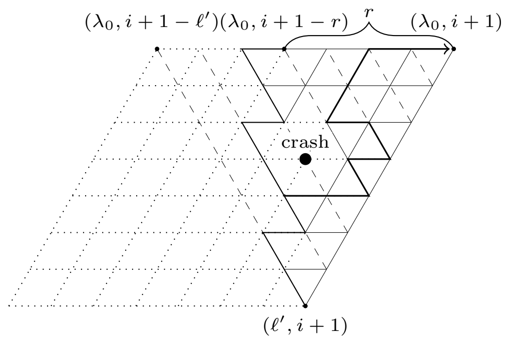
Case 2 Non-V-shaped skews, non-triangular (Figure 4c): Case 1 does not apply and $p_{\text{left}}^{i+1 \to (\ell,i)}$ starts at some node $(0, j_0)$, for $j_0 \neq i+1$

If the above Subcase (2b) of Case 1a does not apply, the causal path to the fast node (ℓ, i) can be constructed independently of the causal path to the slow node $(\ell, i+1)$: For every $\lambda \in \{\lambda_0, \ldots, \ell\}$, there are neighbors (usually (λ, i) and $(\lambda, i+1)$, except for the possible column shift needed for circumventing the faulty node) with a difference in triggering times of more than d^+.

[11]Note that increasing the difference in triggering times between neighboring nodes on a segment of the causal path that is also part of the other causal path leading to the triggering of the slow node $(\ell, i+1)$ relaxes the upper bound on $t_{\ell,i+1}$ no more than it strengthens the lower bound on $t_{\ell,i}$.



(a) The crashed (or Byzantine) node is placed such that it can only be bypassed from the right.



(b) The crashed (or Byzantine) node is placed such that the pulse wave bypasses it from both sides.

Figure A.23: Illustration of how to adapt Lemma 2, which is used in Case 3 of Lemma 4, to a single crash or Byzantine fault. Note that the faulty node cannot be too close to the right boundary of the triangle to be bypassed from the right, as in this case the boundary would be pushed to the right (by one or two columns).

When constructing the left zig-zag path ending at (ℓ, i), the faulty node may be approached either via some correct node's left link or lower-right link: In the former case, we evade via the other causal link to the lower-left neighbor. If we would reach the faulty node via the lower-right link, we first backtrack/go one hop to the right before we go down-right and finally down-left (see Figure 6). Clearly, this makes the causal path shorter and thereby decrease the triggering time bound for the fast node by at most $\mathcal{O}(d^-)$. The latter is of course only possible if the faulty node is in a column smaller than i.

If the faulty node is in column i or $i+1$, however, which led to the index shift by $k \in \{1, 2, 3\}$ mentioned in (2a) above, we need to construct $p_{\text{left}}^{i+2\to(\ell,i)}$ or $p_{\text{left}}^{i+3\to(\ell,i)}$ instead of $p_{\text{left}}^{i+1\to(\ell,i)}$. In these left zig-zag paths, we can pass around the faulty node to the right. Note that this may actually increase the fast node's triggering time bound by $\mathcal{O}(d^-)$, albeit the resulting decrease of the skew bound is by far outweight by the simultaneous increase $\mathcal{O}(d^+)$ of the triggering time of the slow node (which is caused by circumventing the faulty node in the causal path to $(\ell, i+1)$, as also discussed in (2a) above).

The same is true if the faulty node lies in the triangular region with corners $(0, i+1)$, $(0, i+1+\lambda_0)$, and $(\lambda_0, i+1)$ used in Case 2 (Figure 4c), see Figure 6. In fact, even a fail-silent node could not prevent that $(\lambda_0, i+1)$ is triggered quickly here: By adding a few nodes, if necessary, and arguing that the faulty node can be bypassed by right- or left-triggering, at most $\mathcal{O}(d^+)$ time is lost. This finally covers Case 2 of Lemma 4.

Case 3 Non-V-shaped skews, triangular (Figure 4b): Neither Case 1 nor Case 2 apply

The reasoning for circumventing the faulty node in the construction of the (now triangular) left zig-zag path in Case 3 of Lemma 4 is the same as for Case 2 above. For computing the triggering time of the fast node, though, we also need to adapt Lemma 2 to cope with a faulty node within the part of the triangle with corners $(\ell', i+1)$, $(\lambda_0, i+1-\ell')$, and $(\lambda_0, i+1)$ in Figure 4b that is to the right of $p_{\text{left}}^{i+1\to(\ell,i)}$. Note that the latter possibly needed to evade the faulty node as well, see Figure A.23a.

Recall that the proof of Lemma 2 proceeds by induction over the left-diagonals of the triangle. As already argued in (2a), the modified shape of the zig-zag path does not invalidate the statement of the lemma. Moreover, if the faulty node has column index smaller than i, its right and upper right neighbors will be centrally and right-triggered, respectively, at most $\mathcal{O}(d^+)$ later (if they are not triggered otherwise earlier) than indicated by the bounds derived in the original Lemma 2. Examples of the two basic cases that may occur here are shown in Figure A.23. This covers all possibilities for Case 3 of Lemma 4, except the ones where the faulty node has column index i or $i+1$. In this event, as already argued before, we fall back to considering $p_{\text{left}}^{i+2\to(\ell,i)}$ or $p_{\text{left}}^{i+3\to(\ell,i)}$, which again allows the causal path leading to the slow node to pass the faulty node on its right.

References

[1] 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) (2008) c1–269.

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

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

[4] 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, B. McCredie, A Clock Distribution Network for Microprocessors, IEEE Journal of Solid-State Circuits 36 (5) (2001) 792–799.

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

[6] D.-J. Lee, M.-C. Kim, I. Markov, Low-power Clock Trees for CPUs, in: Proc. 2010 Conference on Computer-Aided Design (ICCAD), 2010, pp. 444–451.

[7] D.-J. Lee, I. Markov, Multilevel Tree Fusion for Robust Clock Networks, in: 2011 Conference on Computer-Aided Design (ICCAD), 2011, pp. 632–639.

[8] D. M. Chapiro, Globally-Asynchronous Locally-Synchronous Systems, Ph.D. thesis, Stanford University (1984).

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

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

[11] D. J. Kinniment, Synchronization and Arbitration in Digital Systems, Wiley Publishing, 2008.

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

[13] Y. Semiat, R. Ginosar, Timing Measurements of Synchronization Circuits, in: Proc. 9th Symposium on Asynchronous Circuits and Systems (ASYNC), 2003, pp. 68–77.

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

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

[16] T. Polzer, T. Handl, 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.

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

[18] E. W. Dijkstra, Self-Stabilizing Systems in Spite of Distributed Control, CACM 17 (11) (1974) 643–644.

[19] S. Biaz, J. Welch, Closed Form Bounds for Clock Synchronization Under Simple Uncertainty Assumptions, Information Processing Letters 80 (3) (2001) 151–157.

[20] C. Lenzen, T. Locher, R. Wattenhofer, Tight Bounds for Clock Synchronization, in: Journal of the ACM, Vol. 57(2), 2010.

[21] S. Reddy, G. Wilke, R. Murgai, Analyzing Timing Uncertainty in Mesh-based Clock Architectures, in: Proc. Design, Automation and Test in Europe (DATE), Vol. 1, 2006, pp. 1–6.

[22] 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) (2010) 245–249.

[23] M. S. Maza, 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.

[24] S. Fairbanks, Method and Apparatus for a Distributed Clock Generator, US patent no. 7,126,405 B2, Filed December 2nd., 2003, Issued October 24th., 2006 (2006).

[25] S. Fairbanks, 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.

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

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

[28] M. Saint-Laurent, 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.

[29] M. Függer, A. Dielacher, 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.

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

[31] D. Dolev, M. Függer, C. Lenzen, U. Schmid, **F**ault-tolerant **A**lgorithms for **T**ick-generation in **A**synchronous **L**ogic: Robust Pulse Generation, JACMTo appear.

[32] C. Lenzen, M. Perner, M. Sigl, U. Schmid, Byzantine Self-Stabilizing Clock Distribution with HEX: Implementation, Simulation, Clock Multiplication, in: Proc. 6th Conference on Dependability (DEPEND), 2013.

[33] D. Dolev, M. Függer, C. Lenzen, M. Perner, U. Schmid, HEX: Scaling Honeycombs is Easier than Scaling Clock Trees, in: Proc. SPAA, 2013.

[34] M. Perner, Self-stabilizing byzantine fault-tolerant clock distribution in grids, Master's thesis, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 3/2/182-2, 1040 Vienna, Austria (2013).
[35] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, A. Yakovlev, Logic Synthesis for Asynchronous Controllers and Interfaces, Springer, 2002.
[36] D. Dolev, M. Függer, C. Lenzen, M. Posch, U. Schmid, A. Steininger, Rigorously Modeling Self-Stabilizing Fault-Tolerant Circuits: An Ultra-Robust Clocking Scheme for Systems-on-Chip, JCSS 80 (4) (2014) 860–900.
[37] UMC 90 nm, 90 nanometer, retrieved: November 20, 2014 from http://www.umc.com/English/process/e.asp (2003).
[38] I. E. Sutherland, Micropipelines, CACM 32 (6) (1989) 720–738.
[39] G. Fuchs, A. Steininger, VLSI Implementation of a Distributed Algorithm for Fault-Tolerant Clock Generation, J. Electr. Comput. Eng. 2011 (936712).
[40] L. Marino, General Theory of Metastable Operation, IEEE Transactions on Computers C-30 (2) (1981) 107–115.

Glossary

$[L]$ denotes the set $\{0, \ldots, L-1\}$. Page 3

d^- is the minimal time a trigger message is delayed between being sent on one and received at another node. Page 4

d^+ is the maximal time a trigger message is delayed between being sent on one and received at another node. Page 4

Δ_ℓ is the skew potential of layer ℓ, and is defined as $\Delta_\ell := \max_{i,j\in[W]}\{t_{\ell,i} - t_{\ell,j} - |i-j|_{\mathrm{W}} d^-\}$. Page 7

ε is the maximal difference of the end-to-end delays, i.e., $d^+ - d^-$. Page 4

L is the number of layers the grid has, which determine the height of the grid. Page 3

λ_0 defines the last layer were a path originating in some layer 0 node may deliver the trigger message not later than the fastest path originating in the same node but ending at layer ℓ. Page 9

(ℓ, i) is the node located in layer ℓ at column i. Page 3

$p_{\mathrm{left}}^{i'\to(\ell,i)}$ is a left zig-zag path. The path is composed of rightward and up-left links with the node (ℓ, i) as destination. The origin is either a node $(0, j)$ or (ℓ', i'). In the latter case, a left zig-zag path is called a triangular path. Page 5

$\mathcal{S}$ is pulse separation time, i.e., the minimal time between the generation to connsecutive pulses k and $k+1$ at layer 0. Page 15

σ_ℓ is the intra-layer skew of layer ℓ, i.e., the skew between neighboring nodes in the same layer. $\sigma_\ell := \max_{i\in[W}\{|t_{\ell,i} - t_{\ell,i+1}|\}$. Page 7

$\hat{\sigma}_\ell$ is the inter-layer skew of layer $\ell > 0$, i.e., the skew between neighboring nodes in different layers. $\hat{\sigma}_\ell := \max_{i\in[W]}\{t_{\ell,i} - t_{\ell-1,i}, t_{\ell,i} - t_{\ell-1,i+1}\}$. Page 7

$\sigma(f)$ is meant to be a bound on the skew between any two correct neighboring nodes, assuming that the system has already "stabilized" and f faults remaining. Page 16

$t_{\ell,i}^{(k)}$ is the triggering time of node (ℓ, i). It denotes the time when the node forwards the k^{th} pulse of the grid. Page 4

T_{link}^- is the shortest time a received trigger message must be memorized. Page 15

T_{link}^+ is the longest time a received trigger message can be memorized. Page 15

T_{sleep}^- is the shortest time a node must be sleep after being triggered. Page 15

T_{sleep}^+ is the longest time a node can be asleep after being triggered. Page 15

W is the number of *columns* the grid has, which determine the width of the grid. Page 3