Distrib. Comput. (2012) 24:323–355
DOI 10.1007/s00446-011-0151-7

Reconciling fault-tolerant distributed computing and systems-on-chip

Matthias Függer · Ulrich Schmid

Received: 17 August 2010 / Accepted: 10 November 2011 / Published online: 11 December 2011

Abstract Classic distributed computing abstractions do not match well the reality of digital logic gates, which are the elementary building blocks of Systems-on-Chip (SoCs) and other Very Large Scale Integrated (VLSI) circuits: Massively concurrent, continuous computations undermine the concept of sequential processes executing sequences of atomic zero-time computing steps, and very limited computational resources at gate-level make even simple operations prohibitively costly. In this paper, we introduce a modeling and analysis framework based on continuous computations and zero-bit message channels, and employ this framework for the correctness & performance analysis of a distributed fault-tolerant clocking approach for Systems-on-Chip (SoCs). Starting out from a "classic" distributed Byzantine fault-tolerant tick generation algorithm, we show how to adapt it for direct implementation in clockless digital logic, and rigorously prove its correctness and derive analytic expressions for worst case performance metrics like synchronization precision and clock frequency. Rather than on absolute delay values, both the algorithm's correctness and the achievable synchronization precision depend solely on the ratio of certain path delays. Since these ratios can be mapped directly to placement & routing constraints, there is typically no need for changing the algorithm when migrating to a faster implementation technology and/or when using a slightly different layout in an SoC.

Keywords Clock synchronization · Fault-tolerant distributed systems · Modeling approaches · VLSI

1 Introduction

Shrinking feature sizes and increasing clock speeds are the most visible signs of the tremendous advances in VLSI design, which will accommodate billions of transistors on a single chip in the near future [34]. This comes at the price of increased system-level complexity, however: With today's deep submicron technology with GHz clock speeds, wiring delays dominate transistor switching delays, and electrical signals cannot traverse the whole chip within a single clock cycle any more. Consequently modern VLSI chips can no longer be viewed as monolithic blocks of synchronous hardware, where all the chip's state-holding gates simultaneously perform state transitions: The engineering of the clock tree, which disseminates the high-speed clock — typically generated by a quartz oscillator in conjunction with a clock multiplier — throughout the whole chip with zero skew is becoming more and more difficult [6,22,52,64] and ineffective. Large VLSI chips are hence nowadays increasingly being considered as more or less loosely-coupled systems of interacting subsystems — the advent of Systems-on-Chip (SoC) and Networks-on-Chip (NoC).

Moreover, the smaller feature sizes and the reduced voltage swing needed for high clock speeds and low power consumption dramatically increase the adverse effects of upsets due to α-particle or neutron hits [4,15,29,36,57,73], crosstalk and ground bouncing [50]. The resulting increase of the

This work originates in our DARTS project, which has been a joint effort of Vienna University of Technology and RUAG Space, see http://ti.tuwien.ac.at/darts for details. It has been supported by the Austrian bm:vit FIT-IT project *DARTS* (809456-SCK/SAI) and the Austrian FWF projects Theta (P17757), PSRTS (P20529) and FATAL (P21694).

M. Függer (✉) · U. Schmid
Technische Universität Wien, Embedded Computing Systems Group (E182/2), Treitlstraße 3, 1040 Vienna, Austria
e-mail: fuegger@ecs.tuwien.ac.at

U. Schmid
e-mail: s@ecs.tuwien.ac.at

transient failure rate (soft-error rate) [46] and crosstalk sensitivity [59] has hence raised concerns about the dependability of future generation VLSI chips [11]. Mitigation techniques exist at different levels of abstraction, including replication of a chip's functional units at system-level. Since in synchronous hardware designs the oscillator and its clock tree make up a non-negligible single point-of-failure [70], consequent fault-tolerant designs, too, comprise SoCs and NoCs of interacting functional units that independently perform state-transitions.

Modern SoCs have hence much in common with loosely-coupled distributed systems that have been studied by the fault-tolerant distributed algorithms community for decades. Recent work e.g. on scheduling of memory requests [54], transactional memory in multicores [19], and self-stabilizing microprocessors [13], as well as our own work introduced below, confirms that it is indeed possible to utilize distributed algorithms research in the VLSI context. Conversely, results and methods from VLSI-related research, as error-correcting codes, have proved useful e.g. for tolerating Byzantine adversaries [14], and blend nicely with distributed algorithms research on fault-tolerant consensus [23], for example. Attempts to systematically bridge the gap between distributed algorithms and VLSI design are lacking, however, [9,67]; our paper makes a first step in this direction.

This work originates in our DARTS project, which is devoted to fault-tolerant clock generation in SoCs: As the zero-skew clock requirement had to be dropped in modern VLSI circuits anyway, replacing the classic centralized clock generation approach (an obvious single-point-of-failure) by a distributed solution becomes a feasible option. Like standard GALS (globally asynchronous, locally synchronous) architectures [8], DARTS replaces the traditional common clock source by multiple clock sources. In sharp contrast to GALS, however, which employs multiple unsynchronized clock devices (typically quartz oscillators), DARTS utilizes a Byzantine fault-tolerant distributed tick generation algorithm, which guarantee some bounded clock skew also between the different clock sources and their clock domains. Such *multi-synchronous* [72,78] GALS systems are beneficial from a designers point of view, since they combine the convenient local synchrony with a global time base across the whole chip. It has been shown in [61] that these properties facilitate even metastability-free high-speed communication across clock domains via bounded-size buffers.

The DARTS clocking scheme has been implemented both in an FPGA [20] and in a custom radiation-hardened ASIC [24,27], which proves that the approach is feasible in practice and indeed works very well. Although the implementation complexity of DARTS is definitely not negligible, it must be considered as the price for a fault-tolerant clocking system. Moreover, it is not clear how an improved and fully engineered version of DARTS, which consists of standard gates and wires only, would actually compare w.r.t. area to traditional carefully balanced clock trees with their many strong clock drivers.

In our attempts to devise a rigorous correctness proof for DARTS, we realized that classic distributed computing abstractions do not match well the peculiarities of hardware implementations:

(i) Inherent fine-grained parallelism, which is caused by a large number of digital logic gates that *concurrently* and *continuously* compute their outputs based on their inputs. This undermines the abstraction of a (typically small) collection of processors that perform atomic computing steps at discrete points in time, which is common to (almost) all existing distributed computing models.
(ii) Very limited resources, which make even apparently simple operations like addition or comparison of k-bit numbers, as well as sending such numbers via messages, prohibitively costly. This is in conflict with the basic atomic operations that are typically used in existing distributed algorithms.

Contributions: This paper introduces a novel framework for modeling and analysis of distributed algorithms implemented in VLSI, which explicitly addresses the above issues. Using this framework, we present a complete and rigorous correctness proof and worst case performance analysis of DARTS. The detailed contributions of our paper are the following:

(1) We introduce a novel modeling and analysis framework based on *signals*, which allow modeling of continuous computations and binary message channels with delay. The framework facilitates "switching" between different — but consistent — abstractions of the same signal (e.g., state view). In sharp contrast to existing modeling frameworks capable of expressing timed executions,[1] these features allow to express the properties — and reason about correctness and performance — of fault-tolerant distributed algorithms designed for a direct implementation in digital logic in a very natural and simple way.
(2) We adapt the simple Byzantine fault-tolerant distributed tick generation algorithm introduced in [82] for a direct implementation in clockless digital logic. Major modifications concern the enforcement of certain atomic actions ("interlocking") via implicit handshaking between parts of the algorithm, and by replacing the k-bit messages used for communication in the original

[1] Our framework is substantially different from existing modeling frameworks for clockless VLSI circuits ("trace theory"), which are time-free and hence cannot deal with failures [21].

algorithm by zero-bit messages, i.e., by up/down signal transitions.

(3) We prove that the adapted algorithm is correct, and derive worst case bounds for its performance metrics like synchronization precision and minimum/maximum clock frequency. Since our system-level proof rests on fundamental properties of certain elementary building blocks only, it effectively reduces the complex problem of guaranteeing system correctness to the problem of guaranteeing the correctness of fairly simple basic blocks. Consequently, in sharp contrast to classic distributed algorithms and their correctness proofs, our low-level modeling approach leaves only a small "proof gap" with respect to the actual implementation.

The paper is organized as follows: Following an overview of related work in Sect. 2, we informally explain the original tick generation algorithm and the required modifications in Sect. 3. Section 4 introduces our modeling framework and provides the system and failure model used in the subsequent analysis, as well as the detailed specification of our hardware tick generation algorithm and its (elementary) building blocks. Section 5 provides the detailed correctness proof and the worst case performance analysis. Some conclusions and directions of further research in Sect. 6 complete the paper. A glossary of our notation can be found in (Table 1).

2 Related work

VLSI clock generation: There exists a huge body of work on classic fault-tolerant clock synchronization [63,74], including hardware-assisted clock synchronization [68], where a set of free-running physical clocks are to be synchronized. In contrast to these approaches, DARTS does *not* compute adjustment values for synchronizing free-running (e.g., driven by a quartz oscillator) local hardware clocks, but generates clock ticks that are inherently synchronized (by means of a distributed algorithm) in a closed-loop fashion. Note that there is research on "extracting" clock synchronization from the actual communication in a distributed computation [1,31,58,60] that bears some relation to our approach.

The few approaches for distributed clock generation without local clock sources we are aware of are essentially based on a (distributed) ring oscillator, which is formed by gates arranged in a feedback loop. Instead of being dictated by a quartz, the frequency of the generated clock signal is determined by the end-to-end delay of the feedback loop. In [51], a regular structure of closed loops of an odd number of inverters is used for distributed clock generation. Similarly, [17,18] employs local tick 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. Since clock synchronization theory [12] reveals that high connectivity is required for bounded synchronization tightness in the presence of failures, however, the sparsely connected designs proposed in [17,18,51] are not fault-tolerant.

Modeling approaches: The theory of asynchronous (clockless) distributed systems — in the absence of failures — has been used in the VLSI community for decades [10]: Research on transition signaling [7,71], delay-insensitivity [16,49], micropipelines [77], etc. has established a sound basis for dealing with self-timed systems [32]. Since then, much research has been conducted on benefits and limitations of clockless circuits. There is a wealth of literature on the arbiter problem [41,47], which is — like the latch, the inertial delay and the mutex — impossible to solve in a delay-insensitive way [3,49]. Both arbiter-free problems [43] and a few ways to circumvent the impossibility of implementing arbiters by adding (some) timing properties [48] or order properties [76] have been thoroughly investigated.

Existing modeling approaches for clockless circuits are based on algebraic trace theory [16,49] or Petri-nets [43,83]; such specifications are time-free. Time-augmented clockless circuits can be handled by using timed Petri-nets, which assign time intervals to each transition [5,55,65,84]. However, to the best of our knowledge, none of these modeling frameworks deals with failures. On the other hand, modeling frameworks developed in distributed algorithms research, like timed I/O automatons [37] or TLA [42], can deal with failures, but are not tailored to the specific needs of VLSI circuits.

Fault-tolerance in VLSI: Dependability concerns have also stimulated a large body of research work devoted to fault-tolerance and fault-prevention in VLSI systems [40]. Fine-grained fault tolerance, e.g. at transistor and gate level, encoding, error detection & recovery/reconfiguration, and radiation hardening techniques are the methods of choice here, see e.g. [35,53,56,79,80] for some examples. The proposed techniques are very different from the "system-level approach" employed in fault-tolerant distributed algorithms, which we will exploit in this paper. We note, however, that those approaches are complementary, in the sense that VLSI fault-tolerance techniques can reduce component failure rates and, hence, the required system-level redundancy.

Although we are not aware of any work that deals with the fine-grain parallelism inherent in VLSI implementations, there is a sizeable body of work devoted to hardware implementations of fault-tolerant algorithms. Well-known examples are MAFT [38], SAFEBUS [33], GUARDS [62] and TTP [39]. However, in sharp contrast to our problem, these systems incorporate hardware *assistance* only. The major part of the algorithms is still implemented in conventional software and executed on general-purpose processors. As a consequence, minimizing the gate-level resource

Table 1 Glossary of our Notation

Name	Description	Definition
B	Maximum node booting completion times	Sect. 4.2
$\#b_p(t)$	# Ticks generated by node p	Sect. 4.2
$b^{max}(t)$	# Ticks generated by first correct node	Sect. 5.2
$b^{min}(t)$	# Ticks generated by last correct node	Sect. 5.2
C	Set of correct nodes in the system	Sect. 4.2
$\#d_{p,q}(t)$	#Ticks commonly removed from (p,q)	Sect. 4.2
$\delta(X;t)$	Value of channel X's delivery function at time t	Sect. 4.1
F	Set of faulty nodes in the system	Sect. 4.2
f	Maximum number of faulty nodes	Sect. 4.2
FCR	Fault-containment region	Sect. 4.2
n	Total number of nodes in the system	Sect. 4.2
P	Set of all n nodes in the system	Sect. 4.2
(p,q)	Pipepair for remote q at local node p	Sect. 4.2
$\widetilde{\mathcal{P}}_{p,q}^{GEQ,o/e}(t)$	Status (p,q): # ticks remote $\geq$ local	Sect. 4.2
$\widetilde{\mathcal{P}}_{p,q}^{GR,o/e}(t)$	Status (p,q): # ticks remote $>$ local	Sect. 4.2
π	Precision bound	Sect. 4.4
$\#r_{p,q}^{loc}(t)$	#Ticks stored in local pipe of (p,q)	Sect. 4.2
$\#r_{p,q}^{rem}(t)$	#Ticks stored in remote pipe of (p,q)	Sect. 4.2
$\widehat{S}$	Event trace representation of signal S	Sect. 4.1
$\widetilde{S}(t)$	Status representation of signal S	Sect. 4.1
$\#S(t)$	Counting function representation of signal S	Sect. 4.1
S_{loc}	Maximum size local pipes	Sect. 4.2
S_{rem}	Maximum size remote pipes	Sect. 4.2
$\#s_{p,q}^{loc}(t)$	Current size local pipe of (p,q)	Sect. 4.2
$\#s_{p,q}^{rem}(t)$	Current size remote pipe of (p,q)	Sect. 4.2
$T_{BS}(k)$	Booting-induced tick k generation time	Eq. 42
T_{first}^{-}	Minimum tick generation interval	Eq. 39
T_{max}	Maximum local loop delay	Constr. 1
T_{min}	Minimum local loop delay	Constr. 1
$T_{min,dis}$	Stripped minimum local loop delay	Constr. 1
T_P	Maximum tick generation interval	Eq. 40
T_{QS}	Maximum tick catchup time	Eq. 41
T_{del}	Maximum time until removal of tick	Eq. 70
T_{del}^{loc}	Maximum time until removal of tick, 2nd bound	Eq. 75
$t_{first,k}$	Time when first correct node sends tick k	Sect. 4.2
$t_{last,k}$	Time when last correct node sends tick k	Sect. 4.2
t_b	Time when first correct node completes booting	Lem. 9
$t_{p,b}$	Time when node p completes booting	Sect. 4.2
$t_{rmv,k}$	Time of removal of tick k from (p,q)	Sect. 4.2
$\left[\tau_{GEQ}^{-},\tau_{GEQ}^{+}\right]$	Min./max. delay GEQ rule	Sect. 4.2
$[\tau_{GR}^{-},\tau_{GR}^{+}]$	Min./max. delay GR rule	Sect. 4.2
$[\tau_{loc}^{-},\tau_{loc}^{+}]$	Min./max. delay local channel + pipe	Sect. 4.2
$[\tau_{rem}^{-},\tau_{rem}^{+}]$	Min./max. delay remote channel + pipe	Sect. 4.2
$[\tau_{TH}^{-},\tau_{TH}^{+}]$	Min./max. delay threshold modules	Sect. 4.2

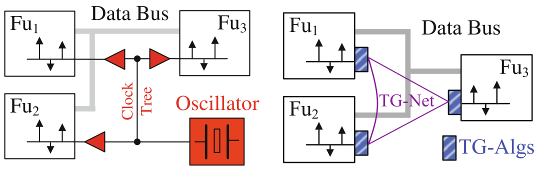

Fig. 1 Replacing synchronous clocking by fault-tolerant distributed tick generation

consumption implied by these algorithms has never been considered. A notable exception is [2], however, where it has been shown that consensus can be solved with 1-bit messages.

3 DARTS informal overview

As shown in Fig. 1, the basic idea of DARTS is to replace the common quartz oscillator and the clock tree by a fully distributed GALS-like approach [8]: Every functional unit Fu_i has attached a dedicated fault-tolerant tick generation block (TG-Alg) here, which generates Fu_i's local clock signal. To accomplish this, all TG-Alg blocks communicate with each other over a simple "network" of clock signals (TG-Net). In contrast to standard GALS, however, DARTS ensures that the local clock signals of different Fu's are synchronized to each other to within a few clock cycles (termed multi-synchronous clocking [72,78]). It has been proved elsewhere [62] that even such loose synchrony suffices for implementing metastability-free high-speed communication between different Fu's using bounded-size buffers.

DARTS clocks (patented in [69]) provide a number of additional advantages, which makes them particularly promising for critical applications in the aerospace domain: First of all, the approach entirely circumvents quartz oscillators, which are fairly big and sensitive devices (shock, vibration, temperature etc.), as well as the cumbersome clock tree engineering issue [6,22,52,64]. It is fault-tolerant, in the sense that the correctness of the clock signals supplied by correct TG-Algs is not affected by transient and permanent failures occurring in other TG-Algs and/or in the TG-Net. DARTS clocks can also be guaranteed to indeed start operating at booting time, a feature that is difficult to ensure for oscillator-based clocking approaches in space applications. Moreover, the clocks always run at the maximum speed and adapt to the current communication delays within the TG-Algs and the TG-Net, both of which may vary with the current operating conditions, such as supply voltage and temperature. And last but not least, since different Fus are driven by slightly different clock signals, DARTS clocks alleviate EM radiation and ground bouncing problems [50] that typically plague devices using synchronous clocking.

The TG-Algs developed and analyzed in this paper derive from a simple synchronizer algorithm for the Θ-Model

```
1: /* Initialization */
2: VAR $k$ : integer := 0;
3: initially send tick(0) to all;
4:
5: if received tick($\ell$) from at least $f+1$
     distinct nodes with $\ell > k$ then
6:     send tick($k+1$), ..., tick($\ell$) to all; $k := \ell$;
7: if received tick($k$) from at least $2f+1$
     distinct nodes then
8:     send tick($k+1$) to all; $k := k+1$;
```

Fig. 2 Algorithm for generating approximately simultaneous tick(k) messages

[44,81,82] introduced in [82]. The (core of this) algorithm, which is based on Srikanth and Toueg's consistent broadcasting primitive [75], is shown in Fig. 2. It assumes a message-driven system (where nodes make atomic receive-compute-send steps whenever they receive a message) of $n = 3f+1$ nodes (= TG-Alg instances), at most f of which may behave arbitrarily faulty, i.e., Byzantine. The number of nodes n is equal to the number of Fus the SoC is partitioned into, and also depends on the intended number of faults f the SoC should sustain. Typical DARTS systems will probably have some f in the order of 1 to 4, resulting in some n in the range of 4 to 13 TG-Algs. All correct nodes are connected by a reliable[2] point-to-point message-passing network (= TG-Net): No spurious messages are ever generated, no messages are lost or altered, and all messages sent at time t are eventually received within the interval $t+[\tau^-, \tau^+]$, where τ^- respectively τ^+ denote the (possibly unknown) lower respectively upper bound on the end-to-end delay of messages exchanged between correct nodes. Let $\varepsilon = \tau^+ - \tau^-$ be the maximum uncertainty of the message delay, and $\Theta = \tau^+/\tau^-$ the maximum delay ratio.

The idea of the algorithm is the following: Initially, every node broadcasts tick(0) in `line 3`. If a correct node p receives $f+1$ tick(ℓ) messages (`line 5`), it can be sure that at least one of those was broadcast by a correct node. Therefore, p can safely catch up and send tick($k+1$), ..., tick(ℓ). If some node p receives $2f+1$ tick(k) messages (`line 7`) and thus sends tick($k+1$), one can be sure that all tick(k) messages broadcast by correct nodes, i.e., at least $f+1$, will be received within time ε by every other correct node. Hence, every correct node will execute `line 5` and send tick(k), if it has not already done so.

In conjunction with the fact that the fastest correct node cannot send consecutive tick(k), tick($k+1$),...arbitrarily fast, this implies a bound on the synchronization precision: Our detailed analysis will reveal that correct nodes generate a sequence of consecutive messages tick(k), $k \geq 1$, in

[2] Note that this reliable network assumption is not unduly restrictive, since communication failures can be mapped to failures of the sending node.

a synchronized way (see Sect. 4.4): If $\#b_p(t)$ denotes the number of tick(k) messages broadcast by node p by real-time t (which is identical to the value of variable k at real-time t, cf. Fig. 2), it will turn out that $(t_2 - t_1)\alpha_{\min} \leq \#b_p(t_2) - \#b_p(t_1) \leq (t_2 - t_1)\alpha_{\max}$ for any correct node p and $t_2 - t_1$ sufficiently large ("accuracy"); the constants $\alpha_{\min}$ and $\alpha_{\max}$ depend on τ^-, τ^+. Moreover, every two correct nodes p, q maintain $|\#b_p(t) - \#b_q(t)| \leq \pi$ ("precision"), for all $t \geq 0$ with a small constant π that depends on Θ only. Note carefully that the algorithm automatically adapts to the instantaneous timing characteristics of any involved computation and communication.

Since the algorithm in Fig. 2 looks very simple, it is tempting to conclude that it is easily translated into a hardware description language: Node p's TG-Alg just needs to drive a Boolean-valued clock signal, where it outputs the k-th signal transition when the algorithm sends its tick(k) message; the TG-Net is formed by feeding all clock signals to all TG-Algs. It turns out, however, that a number of challenging issues must be solved to actually accomplish this:

- *How to implement the TG-Net efficiently?* The algorithm assumes a fully connected network, consisting of n^2 links,[3] so anything beyond a single wire per link is unacceptable [26]. Moreover, for implementation simplicity and performance, the information transmitted via the TG-Net must be kept to a minimum. Ideally, and almost mandatory, the TG-Net should just feed the emitted clock ticks, i.e., signal transitions, of every TG-Alg node to every other TG-Alg node.
- *How to adapt the original algorithm for zero-bit messages?* By just sending signal transitions, no information except the occurrence time can be conveyed over the TG-Net. Thus, the tick number k contained in the messages of Fig. 2 must be maintained at every receiver, individually for every sender.
- *How to map hardware faults to node failures?* Given that the algorithm shown in Fig. 2 tolerates Byzantine failures, we are on the safe side here. Interestingly, there is evidence [30] that assuming less severe failures is inappropriate in the presence of real hardware faults: Even simple stuck-at faults could produce early and/or inconsistently perceived signal transition, and cannot hence be modeled as a crash or omission node failure. More severe hardware faults, like delay faults or early/spurious clock transitions induced e.g. by particle hits or crosstalk, can also easily lead to Byzantine failures.
- *How to ensure atomicity of actions in a VLSI implementation?* This turned out to be the most demanding challenge: All fault-tolerant distributed computing models assume atomic computing steps at the level of a single node. For example, the algorithm presented in Fig. 2 assumes that: (i) messages are received, (ii) the number of received messages is checked with respect to a threshold, and (iii) possibly a new message is broadcast and variable k is updated; all this happens in one atomic computing step. This abstraction does not apply when the algorithm is implemented via clockless digital logic gates, which concurrently and continuously compute their outputs based on their inputs. Explicit synchronization (serialization of actions/interlocking) must be introduced if two local computations must not interfere with each other.

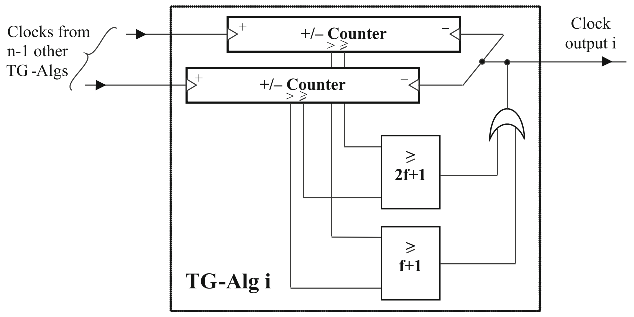

Fig. 3 Basic architecture of a TG-Alg

3.1 Informal overview of the TG-Alg design problems

Taking into account the above issues, we arrived at the basic architecture of a single TG-Alg shown in Fig. 3.

The major building blocks of a single TG-Alg are the $(n - 1)$ $+/-$ *counters*, one for each of the $n - 1$ other TG-Algs in the system. Each such device counts the difference of (i) the number of clock ticks seen from the respective peer, and (ii) the number of clock ticks generated locally so far. Counting the difference between these two numbers is sufficient to implement the algorithm of Fig. 2: To decide when to broadcast the next message, the algorithm needs to know when there are enough remote messages tick(ℓ) for which $\ell > k$ or $\ell \geq k$. Since $\ell > k \Leftrightarrow \ell - k > 0$ and $\ell \geq k \Leftrightarrow \ell - k \geq 0$, it suffices to know when the difference $\ell - k$ is > 0 or ≥ 0. Thus, the $+/-$ counters are supposed to provide two binary status functions, $\widetilde{GR}$ and $\widetilde{GEQ}$, which are true when the counter's actual value is > 0 and ≥ 0, respectively. In addition, a $\geq f + 1$ and a $\geq 2f + 1$ *threshold circuit* is required for implementing the rules in (`line 5`) and (`line 7`) in Fig. 2, respectively. Finally, there is a device (shown as an OR-gate in Fig. 3), which is responsible for generating every local clock tick exactly once when the $\geq f + 1$ *or* the $\geq 2f + 1$ threshold circuit triggers the generation of a new tick message.

Again, the above architecture is deceptively simple. The major problem encountered when trying to implement Fig. 3 in hardware is the lack of a common clock signal that could

[3] Note that a bus of n broadcast links that provide every TG-Alg with the messages from all other TG-Algs is in fact sufficient here.

be used for a synchronous design. Obviously, such a clock signal is not available here. Hence, one has to resort to a (quasi) delay insensitive clockless implementation [32] of the algorithm depicted in Fig. 3, which raises a number of intricate problems:

How to reconcile transition signaling and fault-tolerance? The probably most elegant paradigm for clockless logic is *transition signaling*, where information is conveyed exclusively via signal transitions, rather than via signal states as in conventional logic. Any reasonable delay-insensitive clockless circuit can be composed from a small set of elementary building blocks here, which includes the Muller C-Element that forms the equivalent of the logical AND of two input signals, see [7,48,77] for details.

The "expressive" power of transition signaling is restricted to time-free systems, however, where causality is the only meaningful relation between events. Consequently, there is no full equivalent to the conventional logical OR of two input signals: If only one of the two inputs can provide a transition, the EXOR (exclusive OR) gate can safely be used for this purpose. However, there is no meaningful transition signaling OR if both inputs could—but need not—provide a (somehow related) transition. In fact, generating an OR output transition when the first input transition arrives would destroy the causality relation of the second input transition and the generated output transition.

Unfortunately, incorporating fault-tolerance, as instantiated by our threshold gates, requires exactly this semantics: The clock signal generated by some TG-Alg must not depend causally on the clocks generated by faulty TG-Algs, since this may lead to blocking. Hence, there is no way but to "switch" from transition signaling to state signaling and vice versa to circumvent this problem.

How to manage a clean switch between transition signaling and state signaling? In our implementation of Fig. 3, transition signaling logic is used for processing clock ticks (+/− counters). The +/− counters, on the other hand, output signals with status $\widetilde{GR}$ and $\widetilde{GEQ}$. These signals are then processed by the threshold circuits, which themselves output signals with status $\widetilde{THGR}$ (respectively $\widetilde{THGEQ}$), signalling whether the $\widetilde{GR}$ signals have reached the $f+1$ threshold or not (respectively whether the $\widetilde{GEQ}$ signals have reached the $2f+1$ threshold or not). This information is finally fed into a circuit (depicted by the OR-gate in Fig. 3) responsible for generating exactly one state transition tick k for any k, which is again implemented in transition logic.

The intermediate switching to status signalling performed by the +/− counters, however, bears a problem: it is not feasible to decrement all +/− counters at the same time, since it is inevitable that a local tick(k) message is received at slightly different times by a node's +/− counters. This results in the fact that some of the +/− counters produce a status of $\widetilde{GR}$ and $\widetilde{GEQ}$ based on local tick(k) and others based on the old local tick($k-1$), at least until they finally receive local tick(k). This problem is solved as follows: We will see during the paper that by proper constraints it can be enforced that no three +/− counters set their ports' status based on local ticks k, $k-1$ and $k-2$ or even less at the same time. Thus it suffices to distinguish whether $\widetilde{GR}$ and $\widetilde{GEQ}$ is based on k or on $k-1$, which is done by simply duplicating both signals $\widetilde{GR}$ and $\widetilde{GEQ}$, one for status that is based on even local ticks and one for odd local ticks.

More specifically, since transitions of binary-valued clock signals must strictly alternate between low-to-high (= odd clock tick) and high-to-low (= even clock tick), it is obvious that the next tick to be generated after an even tick must be odd, and vice versa. Hence, in our solution, we just (i) duplicate the signals $\widetilde{GEQ}$, $\widetilde{GR}$, obtaining the signals $\widetilde{GEQ}^e$, $\widetilde{GR}^e$, $\widetilde{GEQ}^o$ and $\widetilde{GR}^o$, (ii) duplicate the two threshold circuits, (iii) duplicate the signals $\widetilde{THGEQ}$ and $\widetilde{THGR}$, obtaining the four signals $\widetilde{THGEQ}^e$, $\widetilde{THGR}^e$, $\widetilde{THGEQ}^o$ and $\widetilde{THGR}^o$, generated by one of the threshold circuits each, and (iv) use the rules:

- Generate an even tick $k+1 \in \mathbb{N}_{even} := 2\mathbb{N}$ if the last tick generated was odd ($k \in \mathbb{N}_{odd} := 2\mathbb{N}+1$) and either (a) the same or a greater number of ticks have been seen from $\geq 2f+1$ TG-Algs ($\widetilde{GEQ}^o$ true), thereby enabling $\widetilde{THGEQ}^o$, or (b) a greater number of ticks have been seen from $\geq f+1$ TG-Algs ($\widetilde{GR}^o$ true), thereby enabling $\widetilde{THGR}^o$. Note that condition (a) ensures that the odd tick k has been seen from $\geq 2f+1$ remote TG-Algs, whereas (b) guarantees that the even tick $k+1$ has already been seen from $\geq f+1$ ones.
- Generate an odd tick $k+1 \in \mathbb{N}_{odd}$ if the last tick generated was even ($k \in \mathbb{N}_{even}$) and either (a) the same or a greater number of ticks have been seen from $\geq 2f+1$ TG-Algs ($\widetilde{GEQ}^e$ true), thereby enabling $\widetilde{THGEQ}^e$, or (b) a greater number of ticks have been seen from $\geq f+1$ TG-Algs ($\widetilde{GR}^e$ true), thereby enabling $\widetilde{THGEQ}^e$. Again, condition (a) ensures that the even tick k has been seen from $\geq 2f+1$ remote TG-Algs, whereas (b) guarantees that the odd tick $k+1$ has already been seen from $\geq f+1$ ones.

Glitches[4], due to the non-simultaneous arrival of clock signal transitions from the peers, are masked by simply ignoring the output of the threshold circuits (say, $\widetilde{THGR}^o$ and $\widetilde{THGEQ}^o$) that generated the even tick k when the next tick

[4] A glitch is a wrong state transition.

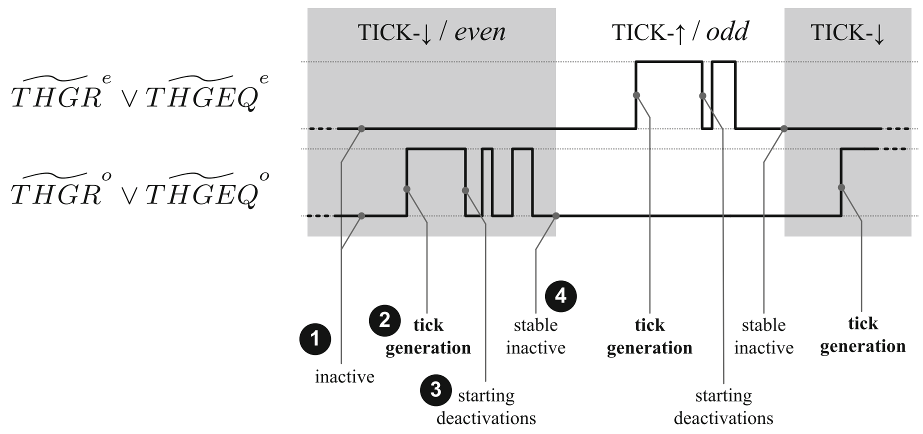

Fig. 4 Tick generation by splitting up in even and odd ticks

$k+1$ to be generated is odd[5]. This "gap" thus allows $\widetilde{GR}^o$ and $\widetilde{GEQ}^o$ (which were responsible for activating $\widetilde{THGR}^o$ or $\widetilde{THGEQ}^o$) for tick k to first become inactive and then become active again for tick $k+2$. As will be revealed by Lemma 3, this is sufficient to avoid mixing up old and new instances of $\widetilde{GR}^o$ (respectively $\widetilde{GEQ}^o$).

As an example consider Fig. 4. It depicts a trace of signal $\widetilde{THGR}^e \vee \widetilde{THGEQ}^e$, responsible for generating odd ticks, as well as signal $\widetilde{THGR}^o \vee \widetilde{THGEQ}^o$, responsible for generating even ticks: ① at the beginning of a new even tick generation, both signals are inactive. ② $\widetilde{THGR}^o$ or $\widetilde{THGEQ}^o$ becomes active due to the arrival of sufficiently many remote ticks, thereby enabling the signal $\widetilde{THGR}^o \vee \widetilde{THGEQ}^o$ that generates an even tick k. ③ The first local tick k arrive at p's $+/-$ counters, which thus disable their $\widetilde{GEQ}^o$ and $\widetilde{GR}^o$ signals, and hence finally $\widetilde{THGR}^o$ and $\widetilde{THGEQ}^o$. Note that $\widetilde{THGR}^o \vee \widetilde{THGEQ}^o$ may intermittently become enabled again, due to the arrival of remote ticks from other TG-Algs. ④ After some time, all of p's $+/-$ counters have received the local tick k and $\widetilde{THGR}^o \vee \widetilde{THGEQ}^o$ has stabilized being inactive. Now the next odd tick $k+1$ is generated as soon as $\widetilde{THGR}^e \vee \widetilde{THGEQ}^e$ becomes active.

How to implement the $+/-$ counters? This task turned out to be the most delicate part of the hardware design work. Actually, implementing a clockless up/down counter is inherently difficult due to the fact that the up-clock ("+ port") and the down-clock ("$-$ port") transitions are totally unrelated. They can hence occur arbitrarily close in time to each other, which usually causes metastability problems [77]. Another problem is how to correctly generate the status of $\widetilde{GR}^o$ and $\widetilde{GEQ}^o$ (respectively $\widetilde{GR}^e$ and $\widetilde{GEQ}^e$). They should truly reflect the current counter value, at least during times when they are used. We will specify the detailed properties that must be maintained by these signals in Sect. 4. Note that our correctness proof and performance analysis will only rely on these properties, i.e., is valid for every implementation of the $+/-$ counters that fulfills these properties.

Our particular implementation of the $+/-$ counters consists of two elastic pipelines [77], which can be seen as shift registers/FIFO buffer for signal transitions. One of the elastic pipelines is attached to the remote clock signal ("+ port"), the other one is fed by the local clock signal ("$-$ port"). They are fitted together at their ends via a special *Diff-Gate*, which removes "matching" transitions, that is, transitions representing the same tick number, as soon as they traveled through the pipelines. The signals with status $\widetilde{GR}^o$, $\widetilde{GEQ}^o$, $\widetilde{GR}^e$ and $\widetilde{GEQ}^e$ are derived from monitoring the last few stages of both pipes. Further details are provided in Sect. 4 and in the descriptions of our implementations [20,24,27].

4 Modeling and analysis framework

In this section, we introduce a modeling framework for fault-tolerant distributed algorithms that are implemented by means of clockless digital circuits, which is amenable to mathematical correctness proofs and worst-case performance analysis. The presented framework addresses the multitude of issues raised in the previous section: It is based on a continuous model of computation and time, and avoids the use of design elements and abstractions that are not available or too costly at the hardware implementation level. To handle the design complexity challenge at such low levels of abstraction, it also supports hierarchical modeling: At the top-level of DARTS, for example, there is the entire system, consisting of n TG-Algs interconnected via the clock signal wires making up the TG-Net; every TG-Alg can be further partitioned into several building-blocks (like the $+/-$ counters), which are interconnected in some non-regular way. Before we formally state the framework, we give an informal overview of the main ingredients.

[5] Of course, no further glitches may occur as soon as this next tick has occurred.

Our modeling framework is based on *modules*, which possess input and output *ports*. An *execution* of a module's ports is an assignment of a *signal* (which captures continuous computation over time) to each of the module's ports. A module's *allowed behavior* is specified by a set of executions of the module's ports. Note that modules differ from classic distributed computing abstractions like timed automatons [45] primarily in that they *continuously* compute their outputs.

Compound modules consist of multiple sub-modules and their interconnect, which specifies how sub-module ports are connected to each other and to the module's input and output ports. The interconnect specification itself assumes zero delays; modeling non-zero interconnect delays, e.g., for real wires, requires intermediate channels: A *channel* is a module that possesses a single input port and a single output port, and its behavior specifies delayed FIFO delivery of input port signal transitions at the output port. Modules that are not further refined are called *basic modules*. Elementary basic modules are those that calculate zero-delay Boolean functions (AND, OR, ...) and channels.

Clearly, the behavior of a (non-faulty) compound module is determined by the behavior of its constituent sub-modules; the behavior of a basic module must be given *a priori*. Correctness proofs establish properties of the behaviors of higher-level compound modules, based on the assumption that (1) the system and failure model holds, and (2) that the implementations of (non-faulty) basic modules indeed satisfy their behavioral specification.

4.1 Signals and zero-bit message channels

Since we target implementations using clockless circuits, our formal framework will be based on a continuous notion of real-time $t \in \mathbb{R}_0^+$. We assume that the system initialization (reset) is triggered at time $t = 0$; different modules may complete their reset at different times, however.

Signal: A signal S is an event trace, i.e., a set of time/value tuples. Formally,

$$S \subseteq \mathbb{R}_0^+ \times \{0, 1\},$$

where event $(t, 1) \in S$ respectively $(t, 0) \in S$ means that S takes on value 1 respectively 0 at time t. We require non-simultaneity of contradicting events on a single signal, i.e.,

$$((t, x) \in S) \wedge ((t, y) \in S) \Rightarrow x = y,$$

and assume that the initial event $(0, I)$, with either $I = 0$ or $I = 1$, is always present in S. We also disallow alternating Zeno behavior in our event traces, i.e., we require that at most finitely many events with different value can occur in any finite time interval, cp. [45, p. 737f]. Note, however, that S may still contain arbitrarily many idempotent events.[6] Consequently, if

$$\begin{aligned} \text{pre}(S, t) &:= \{(t', v') \in S \mid t' \leq t\} \\ \text{suff}(S, t) &:= \{(t', v') \in S \mid t' \geq t\} \end{aligned}$$

denotes the prefix and suffix of S at time t, respectively, there need not be a maximum element $(v_{\max}, t_{\max})$—with respect to the time component—in pre(S, t) and/or no minimum element $(v_{\min}, t_{\min})$ in suff(S, t). However,

$$\begin{aligned} &\text{last-val}(S, t) := v' \text{ such that } \exists (t', v') \in \text{pre}(S, t) : \\ &\qquad \forall (t'', v'') \in \text{pre}(S, t) : (t'' \geq t') \Rightarrow (v'' = v') \end{aligned}$$

is well-defined.[7]

Since our modeling framework is primarily devoted to "real" systems like DARTS, we will restrict our attention to systems made up of well-formed circuits only. A well-formed circuit does not contain zero-delay wires, branches with infinite fan-in/out and other non-implementable assumptions. We say a signal S is *well-formed* if (i) it does not show alternating Zeno behavior and (ii) the function $t \mapsto \text{last-val}(S, t)$ is right-continuous. It can be shown that well-formed circuits never produce signals that are not well-formed, unless their input signals are not well-formed. Therefore, during this paper, we may safely assume that all signals by which the behavior of DARTS is modeled, are well-formed. Well-formed signals allow for more abstract representations than just event traces, which we will now introduce.

In fact, specifying systems in terms of event traces is sometimes overly complicated. More convenient in this regard are two higher-level representations of signals: (i) status, and (ii) counting function. All three representations will be consistent, in a well-defined way, and can hence be used interchangeably.

Status: The status representation of a signal S, denoted by $\widetilde{S}$, is a function

$$\widetilde{S} : \mathbb{R}_0^+ \to \{0, 1\}$$

from real-time to its instantaneous Boolean value, defined by

$$\forall t \geq 0 : \widetilde{S}(t) := \text{last-val}(S, t).$$

Since S is well-formed, the resulting $\widetilde{S}(t)$ is obviously right-continuous.

Status functions may be composed out of already defined status functions by using arbitrary Boolean predicates, e.g., $\widetilde{A} := \widetilde{B} \wedge \widetilde{C}$, with status function $\widetilde{B}$, $\widetilde{C}$, is defined as $\widetilde{A}(t) := \widetilde{B}(t) \wedge \widetilde{C}(t)$.

[6] This is the reason why we use the term "event" here, rather than the more common term "transition".

[7] If alternating Zeno traces were not forbidden, one could construct event traces S where last-val(S, t) is not well-defined. For instance, this happens in a Zeno-trace with alternating $0, 1, 0, \ldots$ events approaching, but not reaching, some time t.

Counting function: Finally, a signal S can be represented by the number of non-idempotent events (excluding the initial event) that occur during $(0, t]$, denoted as the signal's *counting function* $\#S(t)$. For example, if S's event trace is given by $S = \{(0, 0), (1, 1), (1.5, 1), (2, 0)\}$ and $I = 0$, then $\#S(0) = 0, \#S(0.5) = 0, \#S(1) = 1, \#S(1.5) = 1$, and $\#S(2) = 2$. Sometimes, we will also employ generalized counting functions $\#S'(t)$ that have an initial value other than 0: We define $\#S'(t) := \#S(t) + S_0$, where $\#S$ is the standard counting function of S and S_0 an arbitrary offset. It follows immediately from the properties of signals that $t_1 \leq t_2 \Rightarrow \#S(t_1) \leq \#S(t_2)$ for any counting function $\#S$, and that $\#S$ is right-continuous at any point of discontinuity since S is well-formed.

In the sequel, we will use the most convenient representation of a signal S interchangeably, namely S itself, $\widetilde{S}$ or $\#S$.

Execution: We begin with the formal definition of a system. A *system* is a set $\mathcal{P}$ of ports, whereby a *port* can be thought of as a measurement point on a digital chip. An *execution* (of ports $\mathcal{P}$) is a function that assigns each $p \in \mathcal{P}$ a signal $\widehat{p}$. To avoid cluttered notation we simply write $\widetilde{p}$ respectively $\#p$ when we refer to the abstract signal representations $\widetilde{\widehat{p}}$ respectively $\#\widehat{p}$. To specify a system's allowed executions in a convenient way, *modules* are introduced: a *module* is a triple comprising (i) a set of input ports $\mathcal{I}$, (ii) a set of output ports $\mathcal{O}$, and (iii) a set of allowed executions $\mathcal{E}$ of ports $\mathcal{I} \cup \mathcal{O}$. It is the specification of $\mathcal{E}$, where the convenience of a threefold representation of signals comes into play, and it will be extensively used in Sect. 4.3. For example, to specify a module with no input port and a single output port o that produces a constant-0 signal at o, we simply require that $\mathcal{E}$ comprises all executions of ports $\{o\}$ that fulfill $\widetilde{o} = 0$.

The system's allowed executions can thus be specified by stating a set of ports $\mathcal{P}$ together with a set of modules that have input/output ports in $\mathcal{P}$.

Channel: A channel models a reliable FIFO channel for signal transitions with finite delay. Since signal transitions must be alternating, only the occurrence time but no data can be conveyed over a single channel ("zero-bit messages"). Formally, the semantics of a channel X is as follows: Let X^s be the channel's single input port [which will be connected to an output port of a single sender module], and X^r be its single output port [which will be connected to the input ports of some receiver module(s)]. Intuitively a channel maps events at the input port occurring at some time t to (delayed) events at the output port occurring at some delivery time t', where the delay $t' - t$ is not necessarily the same for each event, if the channel has non-constant delay. Formally, we demand that there exists a continuous and strongly monotonically increasing *delivery function* $\delta : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ for X, which maps sending time t to delivery time $\delta(t)$. Note that we will use $\delta(X; t)$ to refer to $\delta(t)$ when the corresponding channel X is not clear from the context. We assume that the channel delay is within $\left[\tau_X^-, \tau_X^+\right]$, i.e., for all times t in $\mathbb{R}_0^+$,

$$\delta(t) - t \in \left[\tau_X^-, \tau_X^+\right]. \tag{1}$$

From the properties of δ, it follows immediately that δ is a bijection from $\mathbb{R}_0^+$ to its codomain $\delta(\mathbb{R}_0^+)$. More specifically, δ maps every closed interval $[t_1, t_2]$ bijectively to the closed interval $[\delta(t_1), \delta(t_2)]$. Clearly, the inverse function δ^{-1} of δ also exists and has the same properties. In addition, we will assume that the channel output has some well-defined initial state (is initialized to) $I \in \{0, 1\}$, which is $I = 0$ if not specified otherwise.

Given δ, the channel's behavior in terms of event traces is specified by two properties, namely

$$(0, I) \in \widehat{X}^r \wedge \nexists (t, v) \in \widehat{X}^r \text{ with } v \in \{0, 1\}, t \in [0, \delta(0))$$

which ensures that before the first event is delivered from the input port to the output port, i.e., before $\delta(0)$, only one event occurs at the output port: the event which sets the channel output port to its initial value I. Secondly we demand that

$$(\delta(t), x) \in \widehat{X}^r \Leftrightarrow (t, x) \in \widehat{X}^s. \tag{2}$$

Since δ carries over the total order of the events (t, x) in $\widehat{X}^s$ to the events $(\delta(t), x)$ in $\widehat{X}^r$ [called *matching* events in the sequel], it follows immediately that $\widehat{X}^r$ is an event trace.

A more abstract specification of a channel in terms of states is by,

$$\forall t \in [0, \delta(0)) : \widetilde{X}^r(t) := \widetilde{X}^r(0) = I \text{ and}$$
$$\forall t \geq 0 : \widetilde{X}^r(\delta(t)) := \widetilde{X}^s(t).$$

Note that this definition is consistent in the sense that an execution fulfilling the event trace specification from above fulfills the abstract state definition.

When considering the counting functions of a constant delay channel's ports, we observe that the output counting function is obtained by shifting the input counting function in time by the constant delay, say τ_X, obtaining $\#X^r(t + \tau_X) = \#X^s(t)$. For non-constant delay channels, this equality does not hold in general, but has to be replaced by inequalities (Pmax) and (Pmin) of the following lemma, summarizing important channel properties:

Lemma 1 *If X is a channel with sending port X^s, receiving port X^r and delays in $\left[\tau_X^-, \tau_X^+\right]$, then the following properties hold:*

(Ps) $t_1 \leq t_2 \Rightarrow \#X^s(t_1) \leq \#X^s(t_2)$
(Pr) $t_1 \leq t_2 \Rightarrow \#X^r(t_1) \leq \#X^r(t_2)$
(Pmax) $\#X^r(t + \tau_X^+) \geq \#X^s(t)$
(Pmin) $\#X^r(t + \tau_X^-) \leq \#X^s(t)$

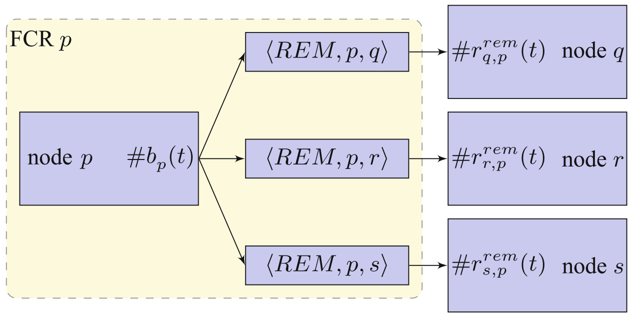
FCR p
node p
$\#b_p(t)$
$\langle REM, p, q\rangle$
$\langle REM, p, r\rangle$
$\langle REM, p, s\rangle$
$\#r^{rem}_{q,p}(t)$ node q
$\#r^{rem}_{r,p}(t)$ node r
$\#r^{rem}_{s,p}(t)$ node s

Fig. 5 Fault-containment region for TG-Alg p

Proof (Ps) and (Pr) follow immediately from the definition of a counting function and the fact that both $\widehat{X}^s$ and $\widehat{X}^r$ are event traces. (Pmax) and (Pmin) follow from the fact that δ bijectively maps the (non-idempotent) events occurring in $\widehat{X}^s$ by time t to the (non-idempotent) events occurring in $\widehat{X}^r$ by time $\delta(t)$, in conjunction with $\delta(t) \in t + \left[\tau_X^-, \tau_X^+\right]$ according to (1). □

4.2 System model

Having introduced the basics of our formal framework, we can now define our system model. A DARTS system consists of a set P of $n := |P|$ top-level modules, where $n \in \mathbb{N}$. The top-level modules will interchangeably be called TG-Alg or node, and are usually denoted by letters p, q etc. Every TG-Alg p in P has exactly one output port with the counting function $\#b_p(t)$, where it broadcasts its clock, and one input port per remote TG-Alg $q \in P \setminus \{p\}$ with the counting function $\#r^{rem}_{p,q}(t)$, where it *receives* q's clock. We assume a fully connected system, i.e., from every TG-Alg p to every TG-Alg $q \in P \setminus \{p\}$, there is a channel $\langle REM, p, q\rangle$ with input $\#b_p(t)$, output $\#r^{rem}_{q,p}(t)$, and delay in $[\tau^-_{rem}, \tau^+_{rem}]$. Figure 5 shows the resulting outbound channels of TG-Alg p.

The following notation will be used throughout the paper: For a function f, let $f(t^{\rightarrow})$ be its *left limit* at time t, i.e., $f(t^{\rightarrow}) := \lim_{\xi \to t} f(\xi)$ and ξ approaches t from the left. For any $k \geq 1$, we say that node p sends tick k, at time $t_{p,k}$, if the kth event (without counting idempotent events) occurs at $t_{p,k}$. In terms of counting functions: $t_{p,k}$ is the time for that $\#b_p(t_{p,k}) = \#b_p(t^{\rightarrow}_{p,k}) + 1 = k$. The time when the first (respectively the last) correct node sends tick k is denoted by $t_{first,k}$ (respectively $t_{last,k}$); note that the node who is the first (respectively last) one to send tick k may be different for different k.

Failure model: Since hardware faults easily lead to Byzantine failures [30], we assume this failure semantics here: The adverse power of Byzantine failures in our context lies in the ability of a faulty node to generate wrong clock ticks (early timing failures or even spurious) that are perceived inconsistently at different remote nodes. Such failures could be the consequence of manufacturing defects or electrostatic breakdown [40], particle hits [4,57], or electromagnetic noise [50], which may affect any module in a TG-Alg. Due to different wire lengths and signal-level detection thresholds, such faults typically propagate differently to different receivers. Note that we allow faulty nodes to create even metastability [41], but we must assume that metastability cannot propagate beyond FCRs (see below); we have already some convincing evidence [25] that this is ensured by the elastic pipelines in the $+/-$ counters with large probability.

We partition our system into multiple *fault-containment regions* (FCRs), i.e., sets of (sub-)modules that are potentially affected by a single fault like a particle hit and thus cannot be assumed to fail independently. More specifically, we define FCR p to consist of the single TG-Alg p together with all its outgoing channels, as depicted in Fig. 5. If FCR p is faulty, any of its (sub-)modules may behave arbitrarily (Byzantine).[8] Since every FCR is associated with exactly one TG-Alg, we will also use these terms interchangeably.

Throughout the paper, let C be the set of correct FCRs, and F, with $f := |F|$, the set of faulty FCRs. Clearly $P = C \cup F$ and $C \cap F = \emptyset$, i.e., C and F are partitions of P. We will prove that correct nodes behave as specified in Sect. 4.4 in the presence of up to f Byzantine faulty FCRs, provided that the total number of nodes is $n \geq 3f + 2$. Note that this is slightly more than the required lower bound of $n \geq 3f + 1$ for clock synchronization [12], but facilitates a considerably better precision and accuracy (attained by counting only remote messages when calculating the $f + 1$ respectively $2f + 1$ thresholds; including self-reception would lead to $\tau^-_{rem} = \tau^-_{loc}$ in Theorem 2 and hence spoil the achievable worst-case precision). In case one does not want to spend an extra node to the required $3f + 1$ nodes, an alternative is to add self-reception and to artificially increase its delay, e.g., by feeding the signal through inverter chains, so that τ^-_{loc} becomes of the order of remote delays. Note that both a $3f + 2$ node system without and a $3f + 1$ node system with self-reception have about the same overall size for reasonably small f, since the extra digital logic needed for self-reception at $3f + 1$ nodes is about the size of a node without self-reception. This allows the designer to choose between a system with a slightly larger number of nodes and a system which is slightly more resilient, without changing the overall system size. While, throughout this paper, we focus on the solution using $3f + 2$ nodes without self-reception, an adaptation of the algorithm and its correctness proof to the alternative solution can be done in a straightforward manner.

Booting: We assume that the whole system is simultaneously reset at time $t = 0$. However, we allow the modules to

[8] In terms of standard failure models for distributed algorithms, our assumption corresponds to mapping link failures to sender node failures.

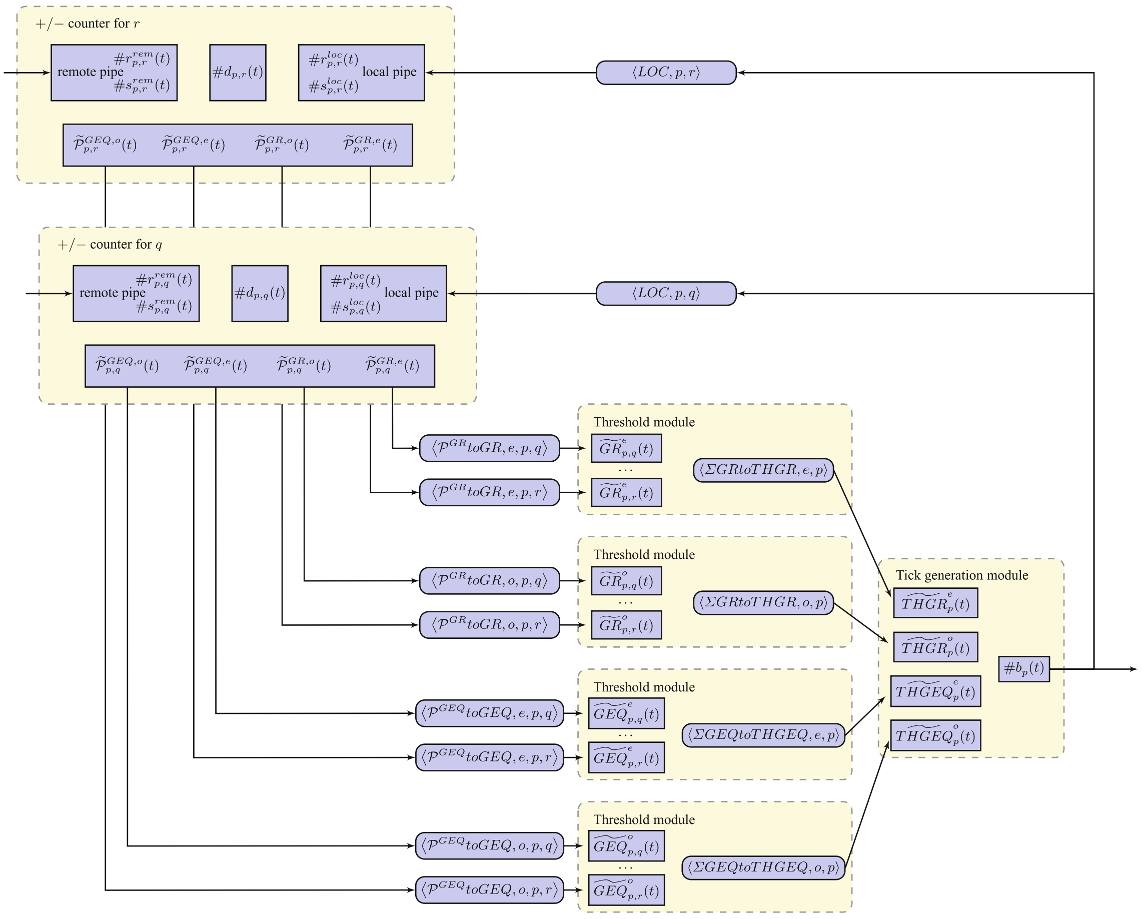

Fig. 6 Architecture of node p

complete booting at (slightly) different times: If $t_{p,b}$ denotes the time by which all modules in a correct FCR p have completed booting, we require that $t_{p,b} \in [0, B]$ for some constant $B \le \tau^-_{rem}$. Note that the latter condition will ensure that messages sent by p are never lost at a correct node q because of late booting.

4.3 TG-Alg architecture and module specification

In this section, we will describe the architecture of a TG-Alg, i.e., its sub-modules and interconnect, and formally specify their behaviors. It is important to note here that the behavioral properties defined in this subsection are *assumed* properties, i.e., basic properties that must a priori be guaranteed by the implementation of the modules. (Modules in FCRs hit by a failure may deviate (arbitrarily) from their correct behavior, however.) Based on these basic properties, the correctness proofs provided in Sect. 5 will show that the system of TG-Algs will maintain the system-level properties (precision and accuracy) as specified in Sect. 4.4.

Figure 6 shows the general architecture of the TG-Alg p, cp. Fig. 3. It consists of one +/− counter module per remote TG-Alg (only two are depicted), four threshold modules implementing the $f + 1$ and $2f + 1$ rules in Fig. 2, and a tick broadcast module that finally generates p's clock ticks $\#b_p(t)$. Every +/− counter is refined into several additional sub-modules: A pair of elastic pipes (remote pipe, local pipe) that form FIFO buffers for (remote, local) clock ticks, a Diff-Gate module that removes matching remote and local ticks from the pipes, and a Pipe Compare Signal Generator (PCSG) module that generates the signals' status reflecting the difference of the number of ticks present in the local and remote pipe.

We will now specify the ports and the behavior of all these modules in detail.

(i) Pairs of elastic pipes: Every TG-Alg p incorporates $n-1$ pairs of elastic pipelines [77], each of which corresponds to a single remote TG-Alg $q \in P \setminus \{p\}$. We will denote the pair of pipes at p corresponding to q by (p, q) in the sequel. (p, q) consists of a remote pipeline that can store up to S_{rem} clock ticks sent by q, and a local pipeline that can hold up to S_{loc} clock ticks sent by p locally. Note that the numbers S_{rem} and S_{loc} are implementation parameters that have to be chosen in accordance with Theorems 4 and 6; in the specifications of this section, they are just assumed to be unbounded (= always sufficiently large).

The local pipe in (p, q) has a single input port that is fed by TG-Alg p's local clock ticks, i.e., $\#b_p(t)$, supplied via the channel $\langle LOC, p, q\rangle$, and a single output port represented by the counting function $\#r^{loc}_{p,q}(t)$. Similarly, the remote pipe in (p, q) has a single input port that is fed by TG-Alg q's local clock ticks supplied via the channel $\langle REM, q, p\rangle$, cp. Fig. 5, and a single output port represented by the counting function $\#r^{rem}_{p,q}(t)$.

We say that p receives tick k from the remote node q at time t, or, equivalently, that remote tick k is received in the pipepair (p, q) at p if $\#r^{rem}_{p,q}(t) = \#r^{rem}_{p,q}(t^{\rightarrow}) + 1 = k$. Analogously, we say that local tick k is received in the pipepair (p, q) at p, if $\#r^{loc}_{p,q}(t) = \#r^{loc}_{p,q}(t^{\rightarrow}) + 1 = k$.

Behavioral description: Both pipes in the pair (p, q) have the behavior of a zero-delay[9] channel: For any $t \geq 0$, $\#r^{loc}_{p,q}(t)$ respectively $\#r^{rem}_{p,q}(t)$ gives the number of clock ticks that reached the end of the local respectively remote pipe by time t. Upon reset, both pipes are pre-filled with the virtual tick 0 (such a tick is called virtual since it has never been sent), and $\#r^{rem}_{p,q}(t_{p,b}) = \#r^{loc}_{p,q}(t_{p,b}) := 0$.

(ii) Diff-Gate module: To avoid pipes with infinite capacity, each pair of pipes is equipped with a special Diff-Gate circuit that removes matching clock ticks, i.e., clock ticks contained in both pipes. However, it deletes matching ticks only if at least one tick remains in both the local and remote pipe. The Diff-Gate for (p, q) has two input ports connected to $\#r^{rem}_{p,q}(t)$ and $\#r^{loc}_{p,q}(t)$, and a single output port represented by the counting function $\#d_{p,q}(t)$, which gives the largest tick number that has been removed from both the remote and local pipe of (p, q) by time t. To allow removal of the virtual tick 0, the initial value is $\#d_{p,q}(t_{p,b}) = -1$.

Behavioral description: Recall that virtual tick 0 shows up at the output of the local and the remote pipe at booting completion time $t_{loc,0} = t_{p,b}$.

Ticks are removed from the pipes as follows:

- $k = 0$: If
  - tick $k + 1$ shows up at the output $\#r^{rem}_{p,q}(t_{rem,k+1})$ of the remote pipe of (p, q) at time $t_{rem,k+1}$, and
  - tick $k+1$ shows up at the output $\#r^{loc}_{p,q}(t_{loc,k+1})$ of the local pipe of (p, q) at time $t_{loc,k+1}$,

  then tick $k = 0$ is removed at time $t_{rmv,k}$ within

  $$\max\{t_{rem,k+1}, t_{loc,k+1}\} + \left[\tau^-_{Diff}, \tau^+_{Diff}\right].$$

- $k \geq 1$: If
  - tick $k + 1$ shows up at the output $\#r^{rem}_{p,q}(t_{rem,k+1})$ of the remote pipe of (p, q) at time $t_{rem,k+1}$, and
  - tick $k + 1$ shows up at the output $\#r^{loc}_{p,q}(t_{loc,k+1})$ of the local pipe of (p, q) at time $t_{loc,k+1}$, and
  - tick $k - 1$ has been removed at time $t_{rmv,k-1}$,

  then tick k is removed at time $t_{rmv,k}$ within

  $$\max\{t_{rem,k+1}, t_{loc,k+1}, t_{rmv,k-1}\} + \left[\tau^-_{Diff}, \tau^+_{Diff}\right].$$

Thereby τ^-_{Diff} and τ^+_{Diff} are timing parameters of the Diff-gate that specify how fast it removes matching ticks. On top of $\#r^{rem}_{p,q}(t)$, $\#r^{loc}_{p,q}(t)$ and $\#d_{p,q}(t)$, we define the size of the local and remote pipe of (p, q) at time t as

$$\#s^{loc}_{p,q}(t) := \#r^{loc}_{p,q}(t) - \#d_{p,q}(t)$$
$$\#s^{rem}_{p,q}(t) := \#r^{rem}_{p,q}(t) - \#d_{p,q}(t).$$

Note that our definitions imply that a tick occupies space in a pipe only after it reached its end, i.e., when it shows up in $\#r^{rem}_{p,q}(t)$ after some delay $\in \left[\tau^-_{rem}, \tau^+_{rem}\right]$, for example.

(iii) Pipe Compare Signal Generator (PCSG) module: The signals provided by the pair of pipes (p, q) and its Diff-Gate are connected to the PCSG, which generates four signals with status $\widetilde{\mathcal{P}}^{GEQ,o}_{p,q}(t)$, $\widetilde{\mathcal{P}}^{GEQ,e}_{p,q}(t)$, $\widetilde{\mathcal{P}}^{GR,o}_{p,q}(t)$ and $\widetilde{\mathcal{P}}^{GR,e}_{p,q}(t)$ that characterize the difference of the number of clock ticks stored in the remote and local pipes by time t. Different signals are provided for odd and even clock ticks. For example, status $\widetilde{\mathcal{P}}^{GEQ,o}_{p,q}(t)$ signals when the number of remote clock ticks is greater or equal than the number of local clock ticks, provided that the last clock tick that showed up in the local pipe was odd; $\widetilde{\mathcal{P}}^{GR,o}_{p,q}(t)$ does the same for "greater" replacing "greater or equal".

All these signals are fed, via dedicated channels that add some delay, to the threshold modules of the TG-Alg p.

Behavioral description: The signals generated by the PCSG associated with (p, q) must satisfy the following properties:

$$\widetilde{\mathcal{P}}^{GEQ,o}_{p,q}(t) := [\#r^{rem}_{p,q}(t) \geq \#r^{loc}_{p,q}(t)] \wedge [\#r^{loc}_{p,q}(t) \in \mathbb{N}_{odd}] \wedge [\#s^{loc}_{p,q}(t) = 1]$$

$$\widetilde{\mathcal{P}}^{GEQ,e}_{p,q}(t) := [\#r^{rem}_{p,q}(t) \geq \#r^{loc}_{p,q}(t)] \wedge [\#r^{loc}_{p,q}(t) \in \mathbb{N}_{even}] \wedge [\#s^{loc}_{p,q}(t) = 1]$$

$$\widetilde{\mathcal{P}}^{GR,o}_{p,q}(t) := [\#r^{rem}_{p,q}(t) > \#r^{loc}_{p,q}(t)] \wedge [\#r^{loc}_{p,q}(t) \in \mathbb{N}_{odd}] \wedge [\#s^{loc}_{p,q}(t) = 1]$$

$$\widetilde{\mathcal{P}}^{GR,e}_{p,q}(t) := [\#r^{rem}_{p,q}(t) > \#r^{loc}_{p,q}(t)] \wedge [\#r^{loc}_{p,q}(t) \in \mathbb{N}_{even}] \wedge [\#s^{loc}_{p,q}(t) = 1]$$

[9] Actually, the pipe delays are accounted for in the delays of the real channels in the signal path, namely, $\langle REM, q, p\rangle$ and $\langle LOC, p, q\rangle$.

Note that these signals need to be true only if the local pipes contain exactly one tick ($\#s^{loc}_{p,q}(t) = 1$), which makes it easier for an implementation to fulfill these properties.

The above signals are fed into four dedicated channels that connect the PCSG with the threshold modules, all of which are initialized to 0:

- Channel $\langle \mathcal{P}^{GEQ} toGEQ, o, p, q\rangle$ with input $\widetilde{\mathcal{P}}^{GEQ,o}_{p,q}(t)$ and output $\widetilde{GEQ}^{o}_{p,q}(t)$ and delay in $\left[\tau^-_{GEQ}, \tau^+_{GEQ}\right]$.
- Channel $\langle \mathcal{P}^{GEQ} toGEQ, e, p, q\rangle$ with input $\widetilde{\mathcal{P}}^{GEQ,e}_{p,q}(t)$, output $\widetilde{GEQ}^{e}_{p,q}(t)$ and delay in $\left[\tau^-_{GEQ}, \tau^+_{GEQ}\right]$.
- Channel $\langle \mathcal{P}^{GR} toGR, o, p, q\rangle$ with input $\widetilde{\mathcal{P}}^{GR,o}_{p,q}(t)$, output $\widetilde{GR}^{o}_{p,q}(t)$ and delay in $\left[\tau^-_{GR}, \tau^+_{GR}\right]$.
- Channel $\langle \mathcal{P}^{GR} toGR, e, p, q\rangle$ with input $\widetilde{\mathcal{P}}^{GR,e}_{p,q}(t)$, output $\widetilde{GR}^{e}_{p,q}(t)$ and delay in $\left[\tau^-_{GR}, \tau^+_{GR}\right]$.

(iv) Threshold modules: $\widetilde{GEQ}^{o/e}_{p,q}(t)$ and $\widetilde{GR}^{o/e}_{p,q}(t)$ are further processed at four threshold modules: If the number of active $\widetilde{GEQ}^{o/e}_{p,q}(t)$ respectively $\widetilde{GR}^{o/e}_{p,q}(t)$ signals exceeds the threshold of $2f + 1$ respectively $f + 1$, the corresponding threshold signal $\widetilde{THGEQ}^{o/e}_{p}(t)$ respectively $\widetilde{THGR}^{o/e}_{p}(t)$ becomes active within $\left[\tau^-_{TH}, \tau^+_{TH}\right]$. This property will be formalized below as a submodule which continuously computes a Boolean predicate [which is a function of time here], involving the sum of the status functions of a threshold module's input ports, and feeding the result into a channel.
Behavioral description: The threshold modules comprise of modules with the threshold modules' input ports and outputs

$$\Sigma\widetilde{THGEQ}^{o}_{p}(t) = \sum_{q\in Q} \widetilde{GEQ}^{o}_{p,q}(t) \geq 2f + 1$$
$$\Sigma\widetilde{THGEQ}^{e}_{p}(t) = \sum_{q\in Q} \widetilde{GEQ}^{e}_{p,q}(t) \geq 2f + 1$$

$$\Sigma\widetilde{THGR}^{o}_{p}(t) = \sum_{q\in Q} \widetilde{GR}^{o}_{p,q}(t) \geq f + 1$$
$$\Sigma\widetilde{THGR}^{e}_{p}(t) = \sum_{q\in Q} \widetilde{GR}^{e}_{p,q}(t) \geq f + 1,$$

as well as successive channels, which are all initialized to 0 and have delay within $\left[\tau^-_{TH}, \tau^+_{TH}\right]$:

- Channel $\langle \Sigma GEQtoTHGEQ, o, p\rangle$ whose input port is $\Sigma\widetilde{THGEQ}^{o}_{p}$ and the output port is $\widetilde{THGEQ}^{o}_{p}$.
- Channel $\langle \Sigma GEQtoTHGEQ, e, p\rangle$ whose input port is $\Sigma\widetilde{THGEQ}^{e}_{p}$ and the output port is $\widetilde{THGEQ}^{e}_{p}$.
- Channel $\langle \Sigma GRtoTHGR, o, p\rangle$ with input $\Sigma\widetilde{THGR}^{o}_{p}$ and output port $\widetilde{THGR}^{o}_{p}$.
- Channel $\langle \Sigma GRtoTHGR, e, p\rangle$ with input $\Sigma\widetilde{THGR}^{e}_{p}$ and output port $\widetilde{THGR}^{e}_{p}$.

(v) Tick generation module: The TG-Alg p generates the next clock tick, at some time t, when (i) both threshold outputs for the previously generated tick, say, $\widetilde{THGEQ}^{o}_{p}(t)$ and $\widetilde{THGR}^{o}_{p}(t)$, are inactive again, and (ii) at least one threshold output $\widetilde{THGEQ}^{e}_{p}(t)$ or $\widetilde{THGR}^{e}_{p}(t)$ for the current tick becomes active. We will refer to (i) as the *disabling path* and to (ii) as the *enabling path* in the sequel. The Tick broadcast module hence has four input ports connected to the threshold outputs, and a single output port represented by the counting function $\#b_p(t)$, which is the number of ticks broadcast by p by time t. Finally, $\#b_p(t)$ is distributed to the local pipe in (p, q) at TG-Alg p and to the remote pipe in (q, p) at TG-Alg q, for all $q \in P \setminus \{p\}$, via dedicated channels $\langle LOC, p, q\rangle$ and $\langle REM, p, q\rangle$, respectively.
Behavioral description: Let the signal $\widehat{b}_p$ be the set for which

$$(0, 0) \in \widehat{b}_p,$$
$$(t, 0) \in \widehat{b}_p \Leftrightarrow (\widetilde{THGEQ}^{o}_{p}(t) \vee \widetilde{THGR}^{o}_{p}(t)) \wedge \neg(\widetilde{THGEQ}^{e}_{p}(t) \vee \widetilde{THGR}^{e}_{p}(t)),$$
$$(t, 1) \in \widehat{b}_p \Leftrightarrow \widetilde{THGEQ}^{e}_{p}(t) \vee \widetilde{THGR}^{e}_{p}(t)) \wedge \neg(\widetilde{THGEQ}^{o}_{p}(t) \vee \widetilde{THGR}^{o}_{p}(t)).$$

Then, $\#b_p(t)$ is defined as the counting function of $\widehat{b}_p$, with the initial value $\#b_p(0) = 0$.
(vi) Interconnect: The $n - 1$ channels, one for each $q \in P \setminus \{p\}$, $\langle LOC, p, q\rangle$ and $\langle REM, p, q\rangle$ for distributing $\#b_p(t)$ are all initialized to 0 and adhere to the following specifications:

- Local channel $\langle LOC, p, q\rangle$ with input signal $\#b_p(t)$, output $\#r^{loc}_{p,q}(t)$ and delay in $\left[\tau^-_{loc}, \tau^+_{loc}\right]$.
- Remote channel $\langle REM, p, q\rangle$ with input signal $\#b_p(t)$, output $\#r^{rem}_{q,p}(t)$ and delay in $\left[\tau^-_{rem}, \tau^+_{rem}\right]$.

Recall that the channel delays also include the delays of the pipelines.

4.4 System-level properties

In all executions complying to the system and failure model introduced in the previous sections, the following properties must be guaranteed:

(P) **Precision:** (Theorem 2) There is a constant π, such that for every pair of correct nodes p, q in C:

$$\forall t : |\#b_q(t) - \#b_p(t)| \leq \pi. \tag{3}$$

(A) **Accuracy:** (Theorem 3) There are constants R^-, O^-, R^+, $O^+ > 0$, such that for every correct node p in C:

$$O^-(t_2 - t_1) - R^- \leq \#b_p(t_2) - \#b_p(t_1) \leq O^+(t_2 - t_1) + R^+. \tag{4}$$

(S) **Size:** (Theorems 4 and 6) There are constants S_{rem} and S_{loc}, such that for every pair or correct nodes p, q in C:

$$\#s^{loc}_{p,q}(t) \leq S_{loc} \text{ and } \#s^{rem}_{p,q}(t) \leq S_{rem}.$$

Informally, the *precision requirement* (P) just states that the difference of the number of clock ticks generated by any two different correct nodes is bounded, whereas the *accuracy requirement* (A) guarantees some relation of the progress of the clock ticks with respect to the progress of real-time. Note that (A) is also called *envelope requirement* in literature, and effectively bounds the frequency of the generated clock ticks. Finally, the *size requirement* guarantees that the size of the pipelines remains bounded.

In the following Sect. 5, we will show that the system of TG-Algs indeed satisfies the above properties in all executions complying to our system and failure model, provided that (a) the implementations of non-faulty basic modules specified in Sect. 4.3 indeed fulfill their specifications, and (b) the additional "global" Constraints 1–3 introduced later hold. Our Theorems 2, 3, 4 and 6 will also establish numerical values for all the constants introduced above, which only depend on the delay parameters introduced in the specifications of the TG-Alg basic modules in Sect. 4.3.

5 Correctness proofs

Our correctness proof and performance analysis has a layered structure, where the lemmas and theorems in a layer establish higher-level abstractions atop of the lower-level abstractions provided by the layer below:

1. The lowest proof layer (Sect. 5.1) deals with the problem of creating the abstraction of uniquely labeled tick k messages atop of anonymous up/down transitions: Provided that Constraint 1 (which bounds the relative speed ratio of all local channels) holds, the pivotal Interlocking Lemma 3 proves that ticks $k-2, k-4, \ldots$ are never falsely interpreted as tick k by the algorithm.
   The result of the Interlocking Lemma allows us to bound any correct node's maximum tick frequency (Lemma 4), and to rule out the possibility of unbounded queueing effects in the pipes (Lemma 5); the latter requires Constraint 2 to hold, which ensures that the Diff-Gate can digest ticks at the maximum clock frequency. Atop of these results, it is possible to prove that both the $f+1$-rule (GR, Lemma 6) and the $2f+1$-rule (GEQ, Lemma 7) work as expected.
2. The intermediate proof layer (Sect. 5.2) establishes elementary synchronization properties of the ticks generated at different correct nodes, namely, Progress (P), Unforgeability (U), Quasi-Simultaneity (QS) and finally Booting Simultaneity (BS). Note that these results correspond to the synchronization properties of the classic algorithm of Fig. 2, cp. [44,66,75,81,82]. Our major Theorem 1 requires Constraint 3 to hold, which essentially guarantees that even the slowest local channel is faster than the fastest remote channel.
   Based on the elementary synchronization properties, it is possible to bound the progress of the fastest node (Lemmas 10–13) and the slowest node (Lemmas 15–17) in the system.
3. The top layer of our proof (Sect. 5.3) establishes our major results: Bounds for precision (Theorem 2), accuracy (Theorem 3) and pipeline sizes (Theorems 4 and 6).

5.1 Bottom proof layer

We start our detailed treatment with the technical Lemma 2, which asserts a certain persistence of the number of ticks present in the local and remote pipe for a certain time.

Lemma 2 *If, for a correct node $p \in C$ and a different correct node $q \in C \setminus \{p\}$, at time t it holds that*

$$(\#r^{loc}_{p,q}(t) = k) \wedge (\#s^{loc}_{p,q}(t) = 1)$$

for some $k \geq 1$, then it must hold that

$$\#r^{loc}_{p,q}(t - \tau^-_{Diff}) \geq k \text{ and}$$
$$\#r^{rem}_{p,q}(t - \tau^-_{Diff}) \quad \geq k.$$

Proof See Lemma 2 in Appendix A. □

We next establish a main result, namely our Interlocking Lemma 3, which states that an "old" tick $k-2, k-4, \ldots$ is never falsely interpreted as a "new" tick k in the GR and GEQ rules of the algorithm. This is not immediately evident: An even tick $k+1$ is generated by $\widetilde{THGEQ}^o$ or $\widetilde{THGR}^o$ being active (depending on whether the GEQ or the GR rule triggered the broadcast); assume that it was triggered by the $\widetilde{THGEQ}^o$ signal. $\widetilde{THGEQ}^o$ is only enabled, if enough (at least $2f+1$) $\widetilde{GEQ}^o$ signals are active. For all of these signals, it must hold that the responsible $+/-$ counter has received an even number of ticks locally, that is, any $\widetilde{GEQ}^o$ may be based on local tick k or $k-2$ or .... We, however, require tick $k+1$ to depend solely on $\widetilde{GEQ}^o$ signals based on tick k only.

The Interlocking Lemma will require that for each TG-Alg, all the delays along a path through the Threshold module, via the PCSG module and along the local loop are within about a factor 2 of each other, which is expressed formally in Constraint 1.

Constraint 1 (Interlocking Constraint). *We abbreviate*

$$\begin{aligned}T_{max} &:= \tau^+_{TH} + \max(\tau^+_{GR}, \tau^+_{GEQ}) + \tau^+_{loc}\\ T_{min} &:= \tau^-_{TH} + \min(\tau^-_{GR}, \tau^-_{GEQ}) + \tau^-_{loc} + \tau^-_{Diff} \\ T_{min,dis} &:= \tau^-_{TH} + \min(\tau^-_{GR}, \tau^-_{GEQ}) + \tau^-_{loc}.\end{aligned} \quad (5)$$

Then the relation $T_{max} \le T_{min} + T_{min,dis}$ must hold.

Lemma 3 (Interlocking lemma) *If, for some correct node p and $k+1 \ge 2$, $\#b_p(t) = k+1$, then:*

(i) *There exists a set Q of size $|Q| \ge 2f+1$ such that for $t' := t - \tau^-_{TH} - \tau^-_{GEQ}$:*

$$k \in \mathbb{N}_{even} \Rightarrow \forall q \in Q : \exists t_q \le t' : \widetilde{\mathcal{P}}^{GEQ,e}_{p,q}(t_q) \wedge \#r^{loc}_{p,q}(t_q) \ge k$$
$$k \in \mathbb{N}_{odd} \Rightarrow \forall q \in Q : \exists t_q \le t' : \widetilde{\mathcal{P}}^{GEQ,o}_{p,q}(t_q) \wedge \#r^{loc}_{p,q}(t_q) \ge k$$

(ii) *or there exists a set Q of size $|Q| \ge f+1$ such that for $t' := t - \tau^-_{TH} - \tau^-_{GR}$:*

$$k \in \mathbb{N}_{even} \Rightarrow \forall q \in Q : \exists t_q \le t' : \widetilde{\mathcal{P}}^{GR,e}_{p,q}(t_q) \wedge \#r^{loc}_{p,q}(t_q) \ge k$$
$$k \in \mathbb{N}_{odd} \Rightarrow \forall q \in Q : \exists t_q \le t' : \widetilde{\mathcal{P}}^{GR,o}_{p,q}(t_q) \wedge \#r^{loc}_{p,q}(t_q) \ge k$$

Proof The proof is by induction on $k+1 \ge 2$, the number of ticks sent by node p. The lemma is first shown for tick $k+1 = 2$. Then we assume that some tick $k+1 > 2$ is the first tick for which the lemma does not hold. By investigating the cause which triggered the sending of this tick, we obtain a contradiction to Constraint 1.

Begin ($k+1=2$): Assume p sends tick 2 at time $t_{p,2}$. Then, by the algorithm (specification of the tick generation module), (a) $\overset{o}{\widetilde{THGEQ}}_p(t_{p,2})$ or (b) $\overset{o}{\widetilde{THGR}}_p(t_{p,2})$ must have held. We consider both cases:

case (a): If $\overset{o}{\widetilde{THGEQ}}_p(t_{p,2})$, then by the algorithm (specification of the threshold modules), there must be a set $Q \subseteq P \setminus \{p\}$, of size $|Q| \ge 2f+1$, such that, for time $t' := \delta^{-1}(\langle \Sigma GEQtoTHGEQ, o, p\rangle; t_{p,2})$

$$\forall q \in Q : \overset{o}{\widetilde{GEQ}}_{p,q}(t').$$

Again by the algorithm (specification of the PCSG to threshold module channels), $t_q := \delta^{-1}(\langle \mathcal{P}^{GEQ} toGEQ, o, p, q\rangle; t')$ defined for every $q \in Q$, we obtain

$$\forall q \in Q : \widetilde{\mathcal{P}}^{GEQ,o}_{p,q}(t_q)$$

and by this

$$\widetilde{\mathcal{P}}^{GEQ,o}_{p,q}(t_q) \equiv [\#r^{rem}_{p,q}(t_q) \ge \#r^{loc}_{p,q}(t_q)] \wedge [\#r^{loc}_{p,q}(t_q) \in \mathbb{N}_{odd}] \wedge [\#s^{loc}_{p,q}(t_q) = 1]. \quad (6)$$

Since $\#r^{loc}_{p,q}(t_q) \ge 0$ from reset on and $\#r^{loc}_{p,q}(t_q) \in \mathbb{N}_{odd}$,

$$\#r^{loc}_{p,q}(t_q) \ge 1 = k. \quad (7)$$

Finally, from the channel properties, we know that

$$t_{p,2} - t_q = t_{p,2} - \delta^{-1}\Big(\big\langle \mathcal{P}^{GEQ} toGEQ, o, p, q\big\rangle; \delta^{-1}\big(\langle \Sigma GEQtoTHGEQ, o, p\rangle; t_{p,2}\big)\Big) \ge \tau^-_{TH} + \tau^-_{GEQ}.$$

The lemma follows.

case (b): If $\overset{o}{\widetilde{THGR}}_p(t_{p,2})$, then, by the algorithm (specification of the threshold modules), there must be a set $Q \subseteq P \setminus \{p\}$, of size $|Q| \ge f+1$, such that, for time $t' := \delta^{-1}(\langle \Sigma GRtoTHGR, o, p\rangle; t_{p,2})$

$$\forall q \in Q : \overset{o}{\widetilde{GR}}_{p,q}(t').$$

By the algorithm (specification of the PCSG to threshold module channels), and with $t_q := \delta^{-1}(\langle \mathcal{P}^{GR} toGR, o, p, q\rangle; t')$ defined for every $q \in Q$, this implies

$$\forall q \in Q : \widetilde{\mathcal{P}}^{GR,o}_{p,q}(t_q) \quad (8)$$

and by this

$$\widetilde{\mathcal{P}}^{GR,o}_{p,q}(t_q) \equiv [\#r^{rem}_{p,q}(t_q) > \#r^{loc}_{p,q}(t_q)] \wedge [\#r^{loc}_{p,q}(t_q) \in \mathbb{N}_{odd}] \wedge [\#s^{loc}_{p,q}(t_q) = 1]. \quad (9)$$

Since $\#r^{loc}_{p,q}(t_q) \ge 0$ from reset on and $\#r^{loc}_{p,q}(t_q) \in \mathbb{N}_{odd}$,

$$\#r^{loc}_{p,q}(t_q) \ge 1 = k. \quad (10)$$

From the channel properties, we know that

$$t_{p,2} - t_q = t_{p,2} - \delta^{-1}\Big(\big\langle \mathcal{P}^{GR} toGR, o, p, q\big\rangle; \delta^{-1}\big(\langle \Sigma GRtoTHGR, o, p\rangle; t_{p,2}\big)\Big) \ge \tau^-_{TH} + \tau^-_{GR}.$$

The lemma follows, again.

Step ($k+1 \ge 3$): Assume by contradiction that $k+1$ is the first tick for which the lemma does not hold. Let $t_{p,k+1}$ be the time p sends tick $k+1$. Assume wlog. that $k \in \mathbb{N}_{odd}$[10]. We will establish two delay bounds, one on the enabling path and the other on the disabling path.

[10] The proof for $k \in \mathbb{N}_{even}$ is analogous.

Enabling path: To send tick $k+1$ at time $t_{p,k+1}$, by the algorithm (specification of the tick generation module), at least one of the two threshold signals must have been enabled, i.e., (a) $\widetilde{THGEQ}^{o}_{p}(t_{p,k+1})$ or (b) $\widetilde{THGR}^{o}_{p}(t_{p,k+1})$ must have held. We consider both cases:
case (a): If $\widetilde{THGEQ}^{o}_{p}(t_{p,k+1})$, then, by the algorithm (specification of the threshold modules), there must be a set $Q \subseteq P \setminus \{p\}$, of size $|Q| \geq 2f+1$, such that, at time $t' := \delta^{-1}(\langle \Sigma GEQtoTHGEQ, o, p\rangle; t_{p,k+1})$

$$\forall q \in Q : \widetilde{GEQ}^{o}_{p,q}(t'). \tag{11}$$

Again by the algorithm (PCSG to threshold module channels), at $t_q := \delta^{-1}(\langle \mathcal{P}^{GEQ}toGEQ, o, p, q\rangle; t')$ defined for every $q \in Q$, this implies

$$\forall q \in Q : \widetilde{\mathcal{P}}^{GEQ,o}_{p,q}(t_q) \tag{12}$$

and by this

$$\widetilde{\mathcal{P}}^{GEQ,o}_{p,q}(t_q) \equiv [\#r^{rem}_{p,q}(t_q) \geq \#r^{loc}_{p,q}(t_q)] \wedge \\ \wedge[\#r^{loc}_{p,q}(t_q) \in \mathbb{N}_{odd}] \wedge [\#s^{loc}_{p,q}(t_q) = 1]. \tag{13}$$

By the channel properties

$$\tau^{-}_{TH} + \tau^{-}_{GEQ} \leq t_{p,k+1} - t_q \leq \tau^{+}_{TH} + \tau^{+}_{GEQ}. \tag{14}$$

Assuming that $\forall q \in Q : \#r^{loc}_{p,q}(t_q) \geq k$ yields the desired result of the Lemma. Thus we only have to investigate the negation:

$$\exists q \in Q : \#r^{loc}_{p,q}(t_q) < k.$$

Since, by (13), $\#r^{loc}_{p,q}(t_q) \in \mathbb{N}_{odd}$, we obtain

$$\exists q \in Q : \#r^{loc}_{p,q}(t_q) \leq k-2.$$

Thus, tick $k-1$ must be received in the local pipe of pipepair (p,q) at time $t_{rcv,k-1}$, with

$$t_{rcv,k-1} > t_q.$$

The combination with (14) yields

$$t_{p,k+1} - t_{rcv,k-1} < \tau^{+}_{TH} + \tau^{+}_{GEQ}. \tag{15}$$

Let $t_{p,k-1}$ be the sending time of tick $k-1$. Clearly, by the local channel properties, we arrive at

$$t_{p,k-1} \geq t_{rcv,k-1} - \tau^{+}_{loc} \\ \Rightarrow t_{p,k+1} - t_{p,k-1} < \tau^{+}_{TH} + \tau^{+}_{GEQ} + \tau^{+}_{loc}. \tag{16}$$

Before proceeding further, we will handle the disabling path.
Disabling path: Let $t_{p,k}$ be the sending time of tick k. By the induction hypothesis, we know that the lemma holds for tick k. According to the lemma, we have to distinguish two cases (i) and (ii):

case (a.i): Assume that implication (i) is valid, i.e., there exists a set Q' of size $|Q'| \geq 2f+1$, such that, for

$$t_{q'} := \delta^{-1}\Big(\big\langle \mathcal{P}^{GEQ}toGEQ, e, p, q'\big\rangle; \\ \delta^{-1}\big(\langle \Sigma GEQtoTHGEQ, e, p\rangle; t_{p,k}\big)\Big)$$

it holds that

$$\forall q' \in Q' : \widetilde{\mathcal{P}}^{GEQ,e}_{p,q'}(t_{q'}) \wedge \#r^{loc}_{p,q'}(t_{q'}) \geq k-1. \tag{17}$$

Thus, local tick $k-1$ must have been received in pipepair (p,q') at time $t_{q',rcv,k-1}$, with

$$t_{q',rcv,k-1} \leq t_{q'} - \tau^{-}_{Diff}$$

by Lemma 2. Furthermore, by the properties of the local channels,

$$t_{q',rcv,k-1} - t_{p,k-1} \geq \tau^{-}_{loc}.$$

Thus, we find

$$t_{p,k} - t_{p,k-1} \geq \tau^{-}_{TH} + \tau^{-}_{GEQ} + \tau^{-}_{Diff} + \tau^{-}_{loc}. \tag{18}$$

From the algorithm (specification of the tick generation module), it follows that at time $t_{p,k+1}$

$$\neg\widetilde{THGEQ}^{e}_{p}(t_{p,k+1}) \tag{19}$$

[and also $\neg\widetilde{THGR}^{e}_{p}(t_{p,k+1})$, which is handled analogously, cf. (a.ii) below] must hold, i.e., the threshold signals that generated (the odd) tick k must be inactive again. Therefore, for the times $t_{q''}$ defined as

$$t_{q''} := \delta^{-1}\Big(\big\langle \mathcal{P}^{GEQ}toGEQ, e, p, q'\big\rangle; \\ \delta^{-1}\big(\langle \Sigma GEQtoTHGEQ, e, p\rangle; t_{p,k+1}\big)\Big) \tag{20}$$

it must hold that

$$\nexists Q'', |Q''| \geq 2f+1 : \forall q'' \in Q'' : \widetilde{\mathcal{P}}^{GEQ,e}_{p,q''}(t_{q''}). \tag{21}$$

Let us choose $Q'' := Q'$. Because of the FIFO property of the channels and because of $t_{p,k} < t_{p,k+1}$, we obtain

$$t_{q'} < t_{q''}.$$

Clearly, there has to be at least one $q' \in Q'$ for which

$$\widetilde{\mathcal{P}}^{GEQ,e}_{p,q'}(t_{q'}) \text{ but } \neg\widetilde{\mathcal{P}}^{GEQ,e}_{p,q'}(t_{q''}), \tag{22}$$

since otherwise $Q'' := Q'$ would have been a choice for Q'', contradicting (21). However, (22) can only hold if local tick k has been in pipepair (p,q') at time $t_{q',rcv,k}$, with

$$t_{q',rcv,k} \leq t_{q''}.$$

In combination with (20) and the channel properties, this implies

$$t_{p,k+1} - t_{q',rcv,k} \geq \tau^{-}_{TH} + \tau^{-}_{GEQ} \tag{23}$$

and, by the local channel properties,

$$t_{q',rcv,k} - t_{p,k} \geq \tau^-_{loc}. \tag{24}$$

Finally, (18) together with (23) and (24) yields

$$t_{p,k+1} - t_{p,k-1} \geq (\tau^-_{TH} + \tau^-_{GEQ} + \tau^-_{Diff} + \tau^-_{loc}) + (\tau^-_{TH} + \tau^-_{GEQ} + \tau^-_{loc}). \tag{25}$$

case (a.ii): Assuming that implication (ii) is valid, i.e., that there exists a set Q' of size $|Q'| \geq f+1$, such that, for

$$t_{q'} := \delta^{-1}\Big(\Big\langle \mathcal{P}^{GR} toGR, e, p, q' \Big\rangle; \delta^{-1}(\langle \Sigma GRtoTHGR, e, p\rangle; t_{p,k})\Big)$$

it holds that

$$\forall q' \in Q' : \widetilde{\mathcal{P}}^{GR,e}_{p,q'}(t_{q'}) \wedge \#r^{loc}_{p,q'}(t_{q'}) \geq k-1.$$

By analogous arguments as in case (a.i), we obtain

$$t_{p,k+1} - t_{p,k-1} \geq (\tau^-_{TH} + \tau^-_{GR} + \tau^-_{Diff} + \tau^-_{loc}) + (\tau^-_{TH} + \tau^-_{GR} + \tau^-_{loc}). \tag{26}$$

Combination of (a.i) and (a.ii): Combining (16), (25) and (26) leads to

$$\begin{aligned}&(\tau^-_{TH} + \min\{\tau^-_{GEQ}, \tau^-_{GR}\} + \tau^-_{Diff} + \tau^-_{loc})\\ &\quad +(\tau^-_{TH} + \min\{\tau^-_{GEQ}, \tau^-_{GR}\} + \tau^-_{loc})\\ &\leq t_{p,k+1} - t_{p,k-1} < \tau^+_{TH} + \tau^+_{GEQ} + \tau^+_{loc},\end{aligned}$$

which is a contradiction to Constraint 1. The lemma follows.

case (b): If $\widetilde{THGR}^o_p(t_{p,k+1})$, then, by the algorithm (specification of the threshold module), there must be a set $Q \subseteq P \setminus \{p\}$, of size $|Q| \geq f+1$, such that, for time $t' := \delta^{-1}(\langle \Sigma GRtoTHGR, o, p\rangle; t_{p,k+1})$

$$\forall q \in Q : \widetilde{GR}^o_{p,q}(t').$$

By analogous arguments as in case (a), we obtain an analogon to (16) as

$$t_{p,k+1} - t_{p,k-1} < \tau^+_{TH} + \tau^+_{GR} + \tau^+_{loc}, \tag{27}$$

as well as analogons to (25) and (26), namely,

$$t_{p,k+1} - t_{p,k-1} \geq (\tau^-_{TH} + \tau^-_{GEQ} + \tau^-_{Diff} + \tau^-_{loc}) + (\tau^-_{TH} + \tau^-_{GEQ} + \tau^-_{loc}). \tag{28}$$

and

$$t_{p,k+1} - t_{p,k-1} \geq (\tau^-_{TH} + \tau^-_{GR} + \tau^-_{Diff} + \tau^-_{loc}) + (\tau^-_{TH} + \tau^-_{GR} + \tau^-_{loc}). \tag{29}$$

Combination of (b.i) and (b.ii): Combining (27), (28) and (29) leads to

$$\begin{aligned}&(\tau^-_{TH} + \min\{\tau^-_{GEQ}, \tau^-_{GR}\} + \tau^-_{Diff} + \tau^-_{loc})\\ &\quad +(\tau^-_{TH} + \min\{\tau^-_{GEQ}, \tau^-_{GR}\} + \tau^-_{loc})\\ &\leq t_{p,k+1} - t_{p,k-1} < \tau^+_{TH} + \tau^+_{GR} + \tau^+_{loc},\end{aligned}$$

which is a contradiction to Constraint 1. The lemma follows. □

The next lemma establishes a minimum duration between any two successive ticks generated at a correct node, i.e., an upper bound on the clock frequency that could possibly be generated locally at any correct node.

Lemma 4 *If correct node p sends tick $k \geq 1$ at time $t_{p,k}$, then it cannot send tick $k+1$ before $t_{p,k} + T_{min}$.*

Proof Assume by contradiction that p sends tick $k+1$ at time $t_{p,k+1}$, wlog. for $k \in \mathbb{N}_{even}$, with

$$t_{p,k+1} - t_{p,k} < T_{min}. \tag{30}$$

We apply Lemma 3 for tick $k+1$. Thus, implication (i) or (ii) has to be true:

case (i): There exists at least one node $q \in Q$, such that, for some $t_q \leq t_{p,k+1} - \tau^-_{TH} - \tau^-_{GEQ}$

$$\widetilde{\mathcal{P}}^{GEQ,e}_{p,q}(t_q) \wedge (\#r^{loc}_{p,q}(t_q) \geq k) \Rightarrow (\#s^{loc}_{p,q}(t_q) = 1) \wedge (\#r^{loc}_{p,q}(t_q) \geq k).$$

By applying Lemma 2, we obtain

$$\#r^{loc}_{p,q}(t_q - \tau^-_{Diff}) \geq k. \tag{31}$$

Furthermore, by the local channel property and (Pmin) in Lemma 1,

$$\#r^{loc}_{p,q}(t_q - \tau^-_{Diff}) \leq \#b_p(t_q - \tau^-_{Diff} - \tau^-_{loc}). \tag{32}$$

Combining (31) and (32) yields

$$\#b_p(t_q - \tau^-_{Diff} - \tau^-_{loc}) \geq k,$$

i.e., tick k must have been sent at time $t_{p,k}$ with

$$\begin{aligned}t_{p,k} &\leq t_q - \tau^-_{Diff} - \tau^-_{loc}\\ &\leq t_{p,k+1} - \tau^-_{Diff} - \tau^-_{TH} - \tau^-_{GEQ} - \tau^-_{loc}\\ &\leq t_{p,k+1} - \tau^-_{Diff} - \tau^-_{TH} - \min\{\tau^-_{GEQ}, \tau^-_{GR}\} - \tau^-_{loc}\\ &= t_{p,k+1} - T_{min},\end{aligned}$$

contradicting (30). The lemma follows.

case (ii): Starting from (ii) in Lemma 3, the contradiction is derived analogously as for case (i).

The lemma follows in both cases. □

Lemma 5 together with Constraint 2 allows us to exclude the possibility of unbounded queuing effects inside a pipepair (p, q) of local/remote pipes $\#r^{loc}_{p,q}(t)$ and $\#r^{rem}_{p,q}(t)$ situated at correct node p and corresponding to correct node q. Constraint 2 ensures that the Diff-gate can digest matching ticks at least as fast as any node can generate them.

Constraint 2 $\tau^+_{Diff} \leq T_{min}$

Lemma 5 *For any pair of distinct correct nodes p, q and $k \geq 1$: If correct node p sent tick k at $t_{p,k}$ and q sent tick k at $t_{q,k}$, then tick $k-1$ is removed from the local and remote pipe of pipepair (p, q) by time $\max\{t_{p,k} + \tau^+_{loc}, t_{q,k} + \tau^+_{rem}\} + \tau^+_{Diff}$, if Constraint 2 holds.*

Proof The proof is by induction on the number of ticks $k \geq 1$ that are sent by p and q.

Begin ($k = 1$)**:** Assume that p sends tick 1 at $t_{p,k}$. Tick 1 will be received in any of p's local pipes by $t_{p,k} + \tau_{loc}^{+}$. Furthermore, assume that q sends tick 1 at $t_{q,k}$. It will certainly be received in the remote pipe of the pipepair (p, q) by $t_{q,k} + \tau_{rem}^{+}$. Since there is no tick that could block the removing of tick 0, tick 0 is removed by $\max\{t_{p,k} + \tau_{loc}^{+}, t_{q,k} + \tau_{rem}^{+}\} + \tau_{Diff}^{+}$ from both the local and remote pipe, according to the Diff-gate properties. The lemma follows.

Step ($k > 1$)**:** As our induction hypothesis, assume that tick $k-2$ is removed from both pipes by $\max\{t_{p,k-1} + \tau_{loc}^{+}, t_{q,k-1} + \tau_{rem}^{+}\} + \tau_{Diff}^{+}$. By analogous arguments as above, tick k is received in both pipes of (p, q) by t_{both}, defined as

$$t_{both} := \max\{t_{p,k} + \tau_{loc}^{+}, t_{q,k} + \tau_{rem}^{+}\}.$$

Because of Lemma 4, consecutive ticks cannot be generated with less than T_{min} distance in-between, i.e.,

$$t_{p,k} - t_{p,k-1} \geq T_{min} \text{ and}$$
$$t_{q,k} - t_{q,k-1} \quad \geq T_{min}.$$

Thus

$$\begin{aligned} t_{both} &\geq \max\{t_{p,k-1} + \tau_{loc}^{+}, t_{q,k-1} + \tau_{rem}^{+}\} + T_{min} \\ &\geq \max\{t_{p,k-1} + \tau_{loc}^{+}, t_{q,k-1} + \tau_{rem}^{+}\} + \tau_{Diff}^{+} \end{aligned}$$

by Constraint 2. According to the induction hypothesis, we know that tick $k-2$ has already been removed by t_{both}. By the properties of the Diff-gate, tick $k-1$ is hence removed by $t_{both} + \tau_{Diff}^{+}$. The lemma follows. □

The next lemma establishes the result, that the GR-rule really does its duty, that is, sends the next tick if sufficiently many ticks are available.

Lemma 6 *For all correct nodes p and ticks $k \geq 1$: If there exists a set Q of correct nodes with $|Q| \geq f+1$, such that:*

(i) *all nodes $q \in Q$ send tick $k+1$ by $t_{Q,k+1}$.*
(ii) *p sends tick k by $t_{p,k}$,*

then p sends tick $k+1$ by $t' := \max\{t_{p,k} + \tau_{loc}^{+} + \tau_{Diff}^{+} + \tau_{GR}^{+}, t_{p,k} + \tau_{loc}^{+} + \tau_{GEQ}^{+}, t_{Q,k+1} + \tau_{rem}^{+} + \tau_{GR}^{+}\} + \tau_{TH}^{+}$.

Proof Wlog. assume that $k \in \mathbb{N}_{even}$. We distinguish two cases:

(a) Suppose that $\exists q \in Q : \#r_{p,q}^{loc}(t') > k$. Since $\#r_{p,q}^{loc}(t') \leq \#b_p(t')$, as no tick can be locally received if it has not been sent before, $\#b_p(t') > k$ must hold. Thus, p has sent tick $k+1$ by t'. The lemma follows.

(b) Otherwise, assume that

$$\forall q \in Q : \#r_{p,q}^{loc}(t') \leq k. \tag{33}$$

By Lemma 4, it follows that all nodes q send tick k by $t_{Q,k}$ with $t_{Q,k} \leq t_{Q,k+1} - T_{min}$. By applying Lemma 5, we conclude that tick $k-1$ is removed from all of p's pipepairs (for all $q \in Q$) by

$$\begin{aligned} t_{del,k-1} &:= \max\{t_{p,k} + \tau_{loc}^{+}, t_{Q,k} + \tau_{rem}^{+}\} + \tau_{Diff}^{+} \\ &\leq \max\{t_{p,k} + \tau_{loc}^{+}, t_{Q,k+1} - T_{min} + \tau_{rem}^{+}\} + \tau_{Diff}^{+} \\ &\leq \max\{t_{p,k} + \tau_{loc}^{+} + \tau_{Diff}^{+}, t_{Q,k+1} + \tau_{rem}^{+}\} \end{aligned}$$

due to Constraint 2.

Clearly, since all nodes $q \in Q$ are correct and have sent tick $k+1$ by $t_{Q,k+1}$, tick $k+1$ must be received in the remote pipe of (p, q) by $t_{rcv,Q} := t_{Q,k+1} + \tau_{rem}^{+}$. Furthermore, p must receive tick k in all its local pipes by $t_{rcv,p} := t_{p,k} + \tau_{loc}^{+}$. Consequently, at time $t_{rcv} := \max\{t_{rcv,p}, t_{rcv,Q}, t_{del,k-1}\} \leq \max\{t_{p,k} + \tau_{loc}^{+} + \tau_{Diff}^{+}, t_{Q,k+1} + \tau_{rem}^{+}\}$ with $t_{rcv} \leq t'$, it holds that:

$$\begin{aligned} &\#r_{p,q}^{loc}(t_{rcv}) \geq k, \\ &\#r_{p,q}^{rem}(t_{rcv}) \geq k+1, \text{ and} \\ &\#d_{p,q}(t) \geq k-1. \end{aligned} \tag{34}$$

By combination of (34) with (33), it holds that for all $\xi \in [t_{rcv}, t']$:

$$\#r_{p,q}^{loc}(\xi) = k. \tag{35}$$

Furthermore,

$$\begin{aligned} \widetilde{\mathcal{P}}_{p,q}^{GR,e}(\xi) &\equiv (\#r_{p,q}^{rem}(\xi) > \#r_{p,q}^{loc}(\xi)) \\ &\quad \wedge (\#r_{p,q}^{loc}(\xi) \in \mathbb{N}_{even}) \wedge (\#s_{p,q}^{loc}(\xi) = 1) \\ &\equiv (\#r_{p,q}^{rem}(\xi) > k) \wedge (\#r_{p,q}^{loc}(\xi) - \#d_{p,q}(\xi) = 1) \\ &\equiv (\#d_{p,q}(\xi) = k-1). \end{aligned} \tag{36}$$

Assuming $\exists q \in Q : \#d_{p,q}(\xi) > k-1$ implies $\#r_{p,q}^{loc}(\xi) > k$, since by definition of the Diff-gate's behavior, there must be at least one tick in the local pipe which was not deleted, thereby contradicting (35). Thus it must hold that $\forall q \in Q : \#d_{p,q}(\xi) = k-1$. Therefore $\forall q \in Q : \widetilde{\mathcal{P}}_{p,q}^{GR,e}(\xi)$ is true for times $\xi \in [t_{rcv}, t']$.

By the algorithm (specification of the PCSG to threshold module channels), $\forall q \in Q : \widetilde{GR}_{p,q}^{e}(\xi)$ is true for times $\xi \in [t_{rcv} + \tau_{GR}^{+}, t']$. Again by the algorithm (threshold module),

$$\widetilde{THGR}_{p}^{e}(\xi) \tag{37}$$

is true for each time ξ within $\left[t_{rcv}+\tau^+_{GR}+\tau^+_{TH}, t'\right] \subseteq \left[\max\left\{\begin{matrix} t_{p,k}+\tau^+_{loc}+\tau^+_{Diff} \\ t_{Q,k+1}+\tau^+_{rem}\end{matrix}\right\}+\tau^+_{GR}+\tau^+_{TH}, t'\right]$.
It remains to be shown that the disabling path cannot inhibit the generation of tick $k+1$ at p. For the sake of contradiction, assume that the disabling path can enforce $t_{p,k+1} > t'$.
Because of the lemma's assumption (ii), tick k must eventually be received in all of p's local pipes corresponding to correct nodes r, i.e., $\forall r \in C \setminus \{p\}$: $\#r^{loc}_{p,r}(\xi) \geq k$, for $\xi \in \left[t_{p,k}+\tau^+_{loc}, t'\right]$. Since tick $k+1$ is not generated by t', we actually have $\#r^{loc}_{p,r}(\xi) = k$. As $k \in \mathbb{N}_{even}$, this implies

$$\forall r \in C \setminus \{p\} : \neg\left(\widetilde{\mathcal{P}}^{GR,o}_{p,r}(\xi) \vee \widetilde{\mathcal{P}}^{GEQ,o}_{p,r}(\xi)\right).$$

Consequently, by the algorithm (specification of the PCSG to threshold module channels and the threshold modules), together with the fact that there are only up to f faulty nodes, who cannot reach a threshold themselves,

$$\neg\left(\widetilde{THGR}^o_p(\xi) \vee \widetilde{THGEQ}^o_p(\xi)\right) \tag{38}$$

for $\xi \in \left[t_{p,k}+\tau^+_{loc}+\max\{\tau^+_{GR},\tau^+_{GEQ}\}+\tau^+_{TH}, t'\right]$.
Combining (37) and (38) and noting that $\max\{\tau^+_{GR}+\tau^+_{Diff}, \max\{\tau^+_{GR},\tau^+_{GEQ}\}\} = \max\{\tau^+_{GR}+\tau^+_{Diff}, \tau^+_{GEQ}\}$, it is apparent that p must send tick $k+1$ by t', providing the required contradiction. The lemma follows.

The lemma follows in both cases. □

Analogous to Lemma 6, the next lemma states that the GEQ-rule does its duty.

Lemma 7 *For all correct nodes p and ticks $k \geq 1$: If there exists a set Q of correct nodes with $|Q| \geq 2f+1$, such that:*

(i) *all nodes $q \in Q$ send tick k by $t_{Q,k}$,*
(ii) *p sends tick k by $t_{p,k}$,*

then p sends tick $k+1$ by $t' := \max\{t_{p,k}+\tau^+_{loc}+\tau^+_{Diff}+\tau^+_{GEQ}, t_{p,k}+\tau^+_{loc}+\tau^+_{GR}, t_{Q,k}+\tau^+_{rem}+\tau^+_{Diff}+\tau^+_{GEQ}\}+\tau^+_{TH}$.

Proof See Lemma 7 in Appendix A. □

5.2 Intermediate proof layer

Based on the results of the bottom proof layer, we can now establish elementary synchronization properties of the ticks generated at different correct nodes. The following Theorem 1 corresponds to well-known classic results on consistent broadcasting [44,66,75,82], which are expressed and proved in our new modeling framework. Major differences to the existing proofs are the far lower level of abstraction at which the proofs have to be carried out, and the problems arising from queueing effects that are due to bounded queue sizes of the local and remote queues and the fact that a TG-Alg can process and generate ticks not arbitrarily fast, both of which is in contrast to the original algorithm stated in Fig. 2. For the theorem to hold, Constraint 3 must hold, which essentially guarantees that even the slowest local channel is faster than the fastest remote channel.

Constraint 3 *For*

$$T^-_{first} := \tau^-_{rem}+\tau^-_{Diff}+\tau^-_{GEQ}+\tau^-_{TH}, \tag{39}$$

the relation $T^-_{first} \geq \tau^+_{loc}+\max\{\tau^+_{Diff}+\tau^+_{GR}, \tau^+_{GEQ}\}+\tau^+_{TH}$ must hold.

Theorem 1 (Synchronization Properties). *The algorithm satisfies the synchronization properties Progress (P), Unforgeability (U), Quasi-Simultaneity (QS), and Booting-Simultaneity (BS), if Constraints 1, 2, 3 and $n \geq 3f+2$ hold.*

(P) Progress. If all correct nodes send tick $k \geq 1$ by time t, then every correct node sends at least tick $k+1$ by time $t+T_P$, with

$$T_P := \max\left\{\begin{matrix}\tau^+_{loc}+\tau^+_{Diff}+\tau^+_{GEQ}, \\ \tau^+_{loc}+\tau^+_{GR}, \\ \tau^+_{rem}+\tau^+_{Diff}+\tau^+_{GEQ}\end{matrix}\right\}+\tau^+_{TH}. \tag{40}$$

(U) Unforgeability. If no correct node sends tick $k \geq 1$ by time t, then no correct node sends tick $k+1$ by time $t+T^-_{first}$ or earlier, where T^-_{first} is given by (39).
(QS) Quasi-Simultaneity. If some correct node p sends tick $k+1 \geq 2$ by time t, then every correct node (p included) sends at least tick k by time $t+T_{QS}$, with

$$\begin{aligned} T_{QS} := \max\Big\{&(\tau^+_{rem}-\tau^-_{rem})+(\tau^+_{GR}-\tau^-_{GEQ}-\tau^-_{Diff}), \\ &B+\left(\max\left\{\begin{matrix}\tau^+_{GEQ}, \\ \tau^+_{GR}\end{matrix}\right\}-\min\left\{\begin{matrix}\tau^-_{GEQ}, \\ \tau^-_{GR}\end{matrix}\right\}\right)-T^-_{first}\Big\} \\ &+(\tau^+_{TH}-\tau^-_{TH}). \end{aligned} \tag{41}$$

(BS) Booting-Simultaneity. If some correct node sends tick $k \geq 1$ by time t, then every correct node sends at least tick k by time $t+T_{BS}(k)$, with

$$\begin{aligned} T_{BS}(k) := B+&\left(\max\left\{\begin{matrix}\tau^+_{GEQ}, \\ \tau^+_{GR}\end{matrix}\right\}-\min\left\{\begin{matrix}\tau^-_{GEQ}, \\ \tau^-_{GR}\end{matrix}\right\}\right) \\ &+(\tau^+_{TH}-\tau^-_{TH})+(T_P-T^-_{first})(k-1). \end{aligned} \tag{42}$$

We will show the properties Progress (P), Unforgeability (U), Quasi-Simultaneity (QS) and Booting-Simultaneity (BS) one after the other.

Progress (P)

Proof Assume that all correct nodes C, with $|C| \geq 2f+2$, sent tick $k \geq 1$ by time t. Now focus on a correct node $p \in C$: We can apply Lemma 7 with $Q = C \setminus \{p\}$ and $t_{p,k} = t_{Q,k} = t$. Thus, p must send tick $k+1$ by

$$\begin{aligned} t' &= \max \left\{ \begin{array}{l} t_{p,k} + \tau^+_{loc} + \tau^+_{Diff} + \tau^+_{GEQ}, \\ t_{p,k} + \tau^+_{loc} + \tau^+_{GR}, \\ t_{Q,k} + \tau^+_{rem} + \tau^+_{Diff} + \tau^+_{GEQ} \end{array} \right\} + \tau^+_{TH} \\ &= t + \max \left\{ \begin{array}{l} \tau^+_{loc} + \tau^+_{Diff} + \tau^+_{GEQ}, \\ \tau^+_{loc} + \tau^+_{GR}, \\ \tau^+_{rem} + \tau^+_{Diff} + \tau^+_{GEQ} \end{array} \right\} + \tau^+_{TH} \\ &= t + T_P. \end{aligned}$$

The property follows. □

Unforgeability (U)

Proof Let p be the first correct node that sends tick $k+1 \geq 2$ at time $t_{p,k+1}$, and assume wlog. that $k \in \mathbb{N}_{even}$. We apply Lemma 3 and consider the two possible cases:

(i) Since $|Q| \geq 2f+1$, there must be a subset $C' \subseteq Q$ of correct nodes of size $|C'| \geq f+1$. Clearly, it must hold that

$$\forall r \in C' : \exists t_r \leq t' : \widetilde{\mathcal{P}}^{GEQ,e}_{p,r}(t_r) \wedge \#r^{loc}_{p,r}(t_r) \geq k$$

with $t' = t_{p,k+1} - \tau^-_{TH} - \tau^-_{GEQ}$. By applying Lemma 2, we obtain for each of the $r \in C'$

$$\#r^{rem}_{p,r}(t' - \tau^-_{Diff}) \geq k.$$

By the remote channel properties, this implies $\#b_r(t'') \geq k$ with

$$\begin{aligned} t'' &:= t' - \tau^-_{Diff} - \tau^-_{rem} \\ &= t_{p,k+1} - \tau^-_{TH} - \tau^-_{GEQ} - \tau^-_{Diff} - \tau^-_{rem} \\ &= t_{p,k+1} - T^-_{first}, \end{aligned}$$

i.e., node r—and by this the first correct node which sent tick k—has sent tick k by $t_{p,k+1} - T^-_{first}$. The property follows.

(ii) Since $|Q| \geq f+1$, there must be at least one correct nodes $r \neq p$ among Q for which $\#r^{rem}_{p,r}(t') \geq k+1$ with $t' = t_{p,k+1} - \tau^-_{TH} - \tau^-_{GR}$. For r, this implies $\#b_r(t') \geq k+1$—a contradiction to the assumption that p was the first correct node to send tick $k+1$. Again, the property follows.

The property follows in both cases. □

Before turning to the proof of (QS), we proceed with two technical lemmas.

Lemma 8 *If the first correct node p sends tick $k+1 \geq 2$ at time $t_{p,k+1}$, then at $t' := t_{p,k+1} - \tau^-_{TH} - \tau^-_{GEQ} - \tau^-_{Diff}$ it must hold that: There exists a set Q of size $|Q| \geq 2f+1$ such that*

$$\forall q \in Q : \#r^{rem}_{p,q}(t') \geq k$$

Proof Analogous to the proof of (U), we apply Lemmas 3 and 2, and consider the two possible cases:

(i) This case exactly matches the implication of our lemma.

(ii) Since, $|Q| \geq f+1$, there must be at least one correct node $r \neq p$ among Q which has already sent tick $k+1$ before p did; this contradicts the assumption that p is the first correct node to send tick $k+1$.

The lemma holds in both cases. □

Lemma 9 *Let t_b be the time when the first correct node completes booting. Every correct node must send tick 1 within the interval $[t_{first,1}, t_{last,1}]$ with*

$$t_{first,1} \geq t_b + \min\{\tau^-_{GEQ}, \tau^-_{GR}\} + \tau^-_{TH}, \text{ and}$$
$$t_{last,1} \leq t_b + B + \max\{\tau^+_{GEQ}, \tau^+_{GR}\} + \tau^+_{TH}.$$

Proof Let p be the first correct node that completes booting, i.e., $t_b = t_{p,b}$. Since the virtual tick 0 is already available in all pipepairs (p,q) at p at time t_b, the first correct node that sends tick 1 does so at $t_{first,1}$, with

$$t_{first,1} \geq t_b + \min\{\tau^-_{GEQ}, \tau^-_{GR}\} + \tau^-_{TH}.$$

Since all other correct nodes boot by $t_b + B$, they all must send tick 1 by time $t_{last,1}$ with

$$t_{last,1} \leq t_b + B + \max\{\tau^+_{GEQ}, \tau^+_{GR}\} + \tau^+_{TH}.$$

The lemma follows. □

Quasy Simultaneity (QS)

Proof The proof is by induction on the number of ticks $k+1 \geq 2$ sent by the first correct node.

Begin ($k+1=2$): By Lemma 9, the first correct node must send tick 1 at $t_{first,1}$ with

$$t_{first,1} \geq t_b + \min\{\tau^-_{GEQ}, \tau^-_{GR}\} + \tau^-_{TH}.$$

Let $t_{first,2}$ be the time when the first correct node sends tick 2. By Unforgeability,

$$t_{first,2} \geq t_{first,1} + T^-_{first} \\ \geq t_b + \min\{\tau^-_{GEQ}, \tau^-_{GR}\} + \tau^-_{TH} + T^-_{first}.$$

By Lemma 9, all other correct nodes must send tick 1 by $t_{last,1}$ with

$$t_{last,1} \leq t_b + B + \max\{\tau^+_{GEQ}, \tau^+_{GR}\} + \tau^+_{TH}.$$

Thus,

$$t_{last,1} - t_{first,2} \leq B + \max\{\tau^+_{GEQ}, \tau^+_{GR}\} \\ - \min\{\tau^-_{GEQ}, \tau^-_{GR}\} + (\tau^+_{TH} - \tau^-_{TH}) - T^-_{first} \leq T_{QS}.$$

Step ($k + 1 \geq 3$)**:** Let $t_{first,k}$ be the time when the first correct node sends tick k. As our induction hypothesis, we assume that all correct nodes send tick $k-1$ by $t_{last,k-1} \leq t_{first,k} + T_{QS}$.

Let $t_{first,k+1}$ be the time the first correct node, say p, sends tick $k+1 \geq 3$. By Lemma 8, there exists a set Q of size $|Q| \geq 2f+1$, such that at time $t' = t_{first,k+1} - \tau^-_{TH} - \tau^-_{GEQ} - \tau^-_{Diff}$:

$$\forall q \in Q : \#r^{rem}_{p,q}(t') \geq k. \tag{43}$$

Clearly, there is a subset $\widetilde{Q} \subseteq Q$ of correct nodes among Q of size at least $|\widetilde{Q}| \geq f+1$. Let $\widetilde{Q}' := C \setminus (\widetilde{Q} \cup \{p\})$. Now consider the partitioning of correct nodes $C = \widetilde{Q} \cup \{p\} \cup \widetilde{Q}'$. We will prove the lemma separately for each of the three partitions: In case $q \in \widetilde{Q}$, by (43) and the remote channel properties, q has sent tick k by $t' - \tau^-_{rem} < t_{first,k+1}$. In case $q = p$, it has sent tick k by $t_{first,k+1}$. For the only non-trivial case $q \in \widetilde{Q}'$, it follows from the remote channel properties that any $\widetilde{q} \in \widetilde{Q}$, as a correct node, must have sent tick k by

$$t_{\widetilde{Q},k} := t_{first,k+1} - \tau^-_{TH} - \tau^-_{GEQ} - \tau^-_{Diff} - \tau^-_{rem}. \tag{44}$$

Now consider an arbitrary correct node $r \in \widetilde{Q}'$. By the induction hypothesis and (U),

$$t_{r,k-1} \leq t_{first,k} + T_{QS} \\ \leq t_{first,k+1} - T^-_{first} + T_{QS}. \tag{45}$$

We may now apply Lemma 6 to node r and the set of correct nodes $\widetilde{Q}$ with $t_{r,k-1}$ and $t_{\widetilde{Q},k}$ from (45) and (44). This yields: r sends tick k by $t_{r,k}$ with

$$t_{r,k} := \max\left\{\begin{array}{l} t_{r,k-1} + \tau^+_{loc} + \tau^+_{Diff} + \tau^+_{GR}, \\ t_{r,k-1} + \tau^+_{loc} + \tau^+_{GEQ}, \\ t_{\widetilde{Q},k} + \tau^+_{rem} + \tau^+_{GR} \end{array}\right\} + \tau^+_{TH}$$

$$\leq t_{first,k+1} + \tau^+_{TH} \\ + \max\left\{\begin{array}{l} -T^-_{first} + T_{QS} + \tau^+_{loc} + \tau^+_{Diff} + \tau^+_{GR}, \\ -T^-_{first} + T_{QS} + \tau^+_{loc} + \tau^+_{GEQ}, \\ -\tau^-_{TH} - \tau^-_{GEQ} - \tau^-_{Diff} - \tau^-_{rem} + \tau^+_{rem} + \tau^+_{GR} \end{array}\right\}$$

$$\leq t_{first,k+1} + \max\Big\{T_{QS}, T_{QS}, -\tau^-_{TH} - \tau^-_{GEQ} \\ -\tau^-_{Diff} - \tau^-_{rem} + \tau^+_{rem} + \tau^+_{GR} + \tau^+_{TH}\Big\} \tag{46}$$

$$\leq t_{first,k+1} + T_{QS}, \tag{47}$$

where (46) follows by applying Constraint 3 and (47) from the definition of T_{QS}. □

Booting Simultaneity (BS)

Proof The proof is by induction on $k \geq 1$.
Begin ($k = 1$)**:** By analogous means as in the (QS) proof, we obtain

$$t_{first,1} \geq t_b + \min\{\tau^-_{GEQ}, \tau^-_{GR}\} + \tau^-_{TH} \\ t_{last,1} \leq t_b + B + \max\{\tau^+_{GEQ}, \tau^+_{GR}\} + \tau^+_{TH}.$$

Thus,

$$t_{last,1} - t_{first,1} \leq B + \max\{\tau^+_{GEQ}, \tau^+_{GR}\} \\ - \min\{\tau^-_{GEQ}, \tau^-_{GR}\} + (\tau^+_{TH} - \tau^-_{TH}) \leq T_{BS}(1).$$

The proposition follows.
Step ($k > 1$)**:** Assume the Lemma is true for k. From (U) and (P), it follows that

$$t_{first,k+1} - t_{first,k} \geq T^-_{first} \\ t_{last,k+1} - t_{last,k} \leq T_P.$$

In combination with the induction hypothesis, this yields:

$$t_{last,k+1} - t_{first,k+1} = \big(t_{last,k+1} - t_{last,k}\big) \\ + \big(t_{last,k} - t_{first,k}\big) + \big(t_{first,k} - t_{first,k+1}\big) \\ \leq T_P + T_{BS}(k) - T^-_{first} = T_{BS}(k+1).$$

The proposition follows. □

Lemma 10 (Fastest Progress). *Assume that p is the first correct node that sends tick number $k \geq 1$ at time $t_{first,k}$. Then no correct node can send tick $k' \geq k$ before time $t_{first,k} + (k'-k)T^-_{first}$.*

Proof The proof is by induction on $k' - k$.
Begin ($k' = k$)**:** The lemma trivially holds since the first correct node cannot send tick k before time $t_{first,k}$.
Step ($k' \geq k+1$)**:** Assume that p is the first correct node that sends tick k. The first correct node $q \in C$, by the induction hypothesis, does not send tick k' before $t + (k'-k)T^-_{first}$. Because of (U), no other correct node can send tick $k'+1$ by time $t + (k'-k)T^-_{first} + T^-_{first} = t + (k'+1-k)T^-_{first}$. The lemma follows. □

Having completed the proof of our major Theorem 1, we proceed with Lemmas 11, 12 and 13 that bound the progress of the ticks generated by the fastest node. For this purpose, we define $b^{max}(t)$ as the maximum of $\#b_p(t)$ over all correct nodes C, i.e., $b^{max}(t) := \max\{\#b_p(t) \mid p \in C\}$. Similarly, we define $b^{min}(t) := \min\{\#b_p(t) \mid p \in C\}$. Recall that $f(t^{\rightarrow})$ denotes the left limit of function f at time t. For example, if node p sends tick $k \geq 1$ at time $t_{p,k}$, then $\#b_p(t_{p,k}^{\rightarrow}) = k - 1$ (whereas $\#b_p(t_{p,k}) = k$).

Lemma 11 (Maximum Increase of b^{max} in $\big(t_{first,k}, t\big)$). *If the first correct node sends tick $k \geq 1$ at $t_{first,k}$, then for all times $t > t_{first,k}$*

$$b^{max}(t^{\rightarrow}) - b^{max}(t_{first,k}) \leq \left\lceil \frac{t - t_{first,k}}{T^{-}_{first}} \right\rceil - 1$$

or, equivalently: The number N of ticks sent by the correct first node in the interval $I = \big(t_{first,k}, t\big)$ is upper bounded by $\left\lceil \frac{t - t_{first,k}}{T^{-}_{first}} \right\rceil - 1$.

Proof Let $t_{first,j}$ be the time when the first correct node sends tick j, and assume by contradiction that

$$N \geq \left\lceil \frac{t - t_{first,k}}{T^{-}_{first}} \right\rceil. \tag{48}$$

According to the definition of N,

$$t_{first,k+N} < t. \tag{49}$$

By applying Lemma 10 to $t_{first,k}$ and $t_{first,k+N}$, we find

$$\begin{aligned} t_{first,k+N} - t_{first,k} &\geq N T^{-}_{first} \\ &\geq \left\lceil \frac{t - t_{first,k}}{T^{-}_{first}} \right\rceil T^{-}_{first} \\ &\geq t - t_{first,k}. \end{aligned} \tag{50}$$

Clearly, (50) contradicts (49). The lemma follows. □

Lemma 12 (Maximum Increase of b^{max} in $\big(t_{first,k}, t\big]$). *If the first correct node sends tick $k \geq 1$ at $t_{first,k}$, then*

$$\forall t > t_{first,k} : b^{max}(t) - b^{max}(t_{first,k}) \leq \left\lfloor \frac{t - t_{first,k}}{T^{-}_{first}} \right\rfloor$$

or, equivalently: The number N of ticks sent by the correct first node in the interval $I = \big(t_{t_{first,k}}, t\big]$ is upper bounded by $\left\lfloor \frac{t - t_{first,k}}{T^{-}_{first}} \right\rfloor$.

The following Lemma 13 is a weaker form of Lemma 12, where the beginning of the interval I not necessarily coincides with the sending of a tick by the first correct node, i.e., where I is not "aligned" with $t_{first,k}$.

Lemma 13 (Maximum Increase of b^{max} in $\big(t, t'\big]$).

$$\forall t' > t : b^{max}(t') - b^{max}(t) \leq \left\lceil \frac{t' - t}{T^{-}_{first}} \right\rceil$$

or, equivalently: The number N of ticks sent by the correct first node in the interval $I = \big(t, t'\big]$ is upper bounded by $\left\lceil \frac{t' - t}{T^{-}_{first}} \right\rceil$.

The following Lemma 14 is an analogon to Lemma 13 for any correct node p.

Lemma 14 (Maximum Increase of $\#b_p$ in $\big(t, t'\big]$).

$$\forall t' > t : \#b_p(t') - \#b_p(t) \leq \left\lceil \frac{t' - t}{T_{min}} \right\rceil$$

or, equivalently: The number N of ticks sent by the correct node p in the interval $I = \big(t, t'\big]$ is upper bounded by $\left\lceil \frac{t' - t}{T_{min}} \right\rceil$.

The next Lemma 15 bounds the progress of the last correct node.

Lemma 15 (Last Progress) *If the last correct node sends tick $k \geq 1$ at $t_{last,k}$, then the last correct node sends tick $k + N$, $N \geq 0$, by $t_{last,k+N}$, with*

$$t_{last,k+N} - t_{last,k} \leq N T_P.$$

Proof The proof is by induction on N:
Begin ($N = 0$)**:** Clearly $t_{last,k} - t_{last,k} \leq 0$ is true. The lemma follows.
Step ($N > 0$)**:** As induction hypothesis, assume that the lemma is true for $N - 1$. By applying (P), we immediately obtain

$$\begin{aligned} t_{last,k+N} - t_{last,k} &= \big(t_{last,k+N} - t_{last,k+N-1}\big) \\ &\quad + \big(t_{last,k+N-1} - t_{last,k}\big) \\ &\leq T_P + (N-1)T_P = N T_P. \end{aligned} \tag{51}$$

The lemma follows. □

The following two simple technical lemmas complete the intermediate proof layer.

Lemma 16 (Progress by (QS)) *If p is a correct node which sends tick $k \geq 2$ at $t_{p,k}$, then p sends tick $k + N$, $N \geq 1$, at $t_{p,k+N}$ with*

$$t_{p,k+N} - t_{p,k} \leq (N+1)T_P + T_{QS}.$$

Proof With Lemma 15 and (QS), it follows that

$$\begin{aligned} &t_{p,k+N} - t_{p,k} \leq t_{last,k+N} - t_{first,k} \\ &\quad = (t_{last,k+N} - t_{last,k-1}) + (t_{last,k-1} - t_{first,k}) \\ &\quad \leq (N+1)T_P + T_{QS}. \end{aligned}$$

The lemma follows. □

Lemma 17 (Progress by (BS)) *If p is a correct node which sends tick $k \geq 1$ at $t_{p,k}$, then p sends tick $k+N$, $N \geq 1$, at $t_{p,k+N}$ with*

$$t_{p,k+N} - t_{p,k} \leq NT_P + T_{BS}(k).$$

Proof With Lemma 15 and (BS), it follows that

$$\begin{aligned} t_{p,k+N} - t_{p,k} &\leq t_{last,k+N} - t_{first,k} \\ &= (t_{last,k+N} - t_{last,k}) + (t_{last,k} - t_{first,k}) \\ &\leq NT_P + T_{BS}(k). \end{aligned}$$

The lemma follows. □

5.3 Top proof layer

We are now ready for establishing our major results. The first one, Theorem 2, bounds the precision π of our algorithm, i.e., shows that for every pair of correct nodes $p, q \in C : \forall t \geq 0 : |\#b_q(t) - \#b_p(t)| \leq \pi$.

Theorem 2 (Precision). $\pi := \left\lceil \frac{T_{QS}}{T^-_{first}} \right\rceil + 1$ *is a valid precision-bound.*

Proof Let p, q be two distinct correct nodes. Clearly for all t, $|\#b_q(t) - \#b_p(t)| \leq b^{max}(t) - b^{min}(t)$. We will bound this term by distinguishing three cases for t: (i) $t \in [0, t_{last,1})$, (ii) $t = t_{last,k}$ for some $k \geq 1$ and (iii) $t \in (t_{last,k}, t_{last,k+1})$ for some $k \geq 1$.

ad (i): Since $t \in [0, t_{last,1})$, we have $b^{max}(t) \leq b^{max}(t^{\rightarrow}_{last,1})$ and $b^{min}(t) = 0$. Thus,

$$\begin{aligned} & b^{max}(t) - b^{min}(t) \\ &= b^{max}(t) \leq b^{max}(t^{\rightarrow}_{last,1}) \\ &= \left(b^{max}(t^{\rightarrow}_{last,1}) - b^{max}(t_{first,1})\right) + b^{max}(t_{first,1}) \\ &\leq \left\lceil \frac{t_{last,1} - t_{first,1}}{T^-_{first}} \right\rceil && (52) \\ &\leq \left\lceil \frac{B + \max\left\{\begin{matrix}\tau^+_{GEQ}, \\ \tau^+_{GR}\end{matrix}\right\} + \tau^+_{TH} - \left(\min\left\{\begin{matrix}\tau^-_{GEQ}, \\ \tau^-_{GR}\end{matrix}\right\} + \tau^-_{TH}\right)}{T^-_{first}} \right\rceil && (53) \\ &\leq \left\lceil \frac{T_{QS} + T^-_{first}}{T^-_{first}} \right\rceil \\ &= \left\lceil \frac{T_{QS}}{T^-_{first}} \right\rceil + 1, && (54) \end{aligned}$$

where (52) follows from Lemma 11 and (53) from Lemma 9.

ad (ii):

$$\begin{aligned} & b^{max}(t_{last,k}) - b^{min}(t_{last,k}) \\ &= \left(b^{max}(t_{last,k}) - b^{max}(t_{first,k+1})\right) \\ &\quad + \left(b^{max}(t_{first,k+1}) - b^{min}(t_{last,k})\right) \\ &= \left(b^{max}(t_{last,k}) - b^{max}(t_{first,k+1})\right) + 1 \\ &\leq \left\lfloor \frac{t_{last,k} - t_{first,k+1}}{T^-_{first}} \right\rfloor + 1 && (55) \\ &\leq \left\lfloor \frac{T_{QS}}{T^-_{first}} \right\rfloor + 1 && (56) \end{aligned}$$

where (55) follows from Lemma 12 and (56) from (QS).

ad (iii): Since $t \in (t_{last,k}, t_{last,k+1})$, we have $b^{max}(t) \leq b^{max}(t^{\rightarrow}_{last,k+1})$. Thus,

$$\begin{aligned} & b^{max}(t) - b^{min}(t) \\ &= \left(b^{max}(t) - b^{max}(t_{first,k+2})\right) \\ &\quad + \left(b^{max}(t_{first,k+2}) - b^{min}(t)\right) \\ &= \left(b^{max}(t) - b^{max}(t_{first,k+2})\right) + (k+2-k) \\ &\leq \left(b^{max}(t^{\rightarrow}_{last,k+1}) - b^{max}(t_{first,k+2})\right) + 2 \\ &\leq \left\lceil \frac{t_{last,k+1} - t_{first,k+2}}{T^-_{first}} \right\rceil + 1 && (57) \\ &\leq \left\lceil \frac{T_{QS}}{T^-_{first}} \right\rceil + 1, && (58) \end{aligned}$$

where (57) follows from Lemma 11 and (58) from (QS).

Combining (54), (56) and (58) provides a precision bound for arbitrary $t \geq 0$:

$$\max\left\{\left\lfloor \frac{T_{QS}}{T^-_{first}} \right\rfloor, \left\lceil \frac{T_{QS}}{T^-_{first}} \right\rceil\right\} + 1 = \left\lceil \frac{T_{QS}}{T^-_{first}} \right\rceil + 1 = \pi$$

The theorem follows. □

Our next major result, Theorem 3 (Accuracy), bounds the number of ticks generated locally at a correct node p during some real-time interval $(t_1, t_2]$, i.e., allows to make statements about the local clock frequency. For example, it reveals that the long-term frequency is within $\left[1/T_P, 1/T^-_{first}\right]$.

Theorem 3 (Accuracy). *Given t_1 and t_2 with $t_2 > t_1 \geq t_{p,1}$, the accuracy $\#b_p(t_2) - \#b_p(t_1)$ of any correct node p is bounded by*

$$\begin{aligned} & \max\left\{0, 1 + \left\lfloor \frac{t_2 - t_1 - \max\left\{T_{BS}(1), T_{QS} + T_P\right\}}{T_P} \right\rfloor\right\} \\ &\leq \#b_p(t_2) - \#b_p(t_1) \\ &\leq \min\left\{\left\lceil \frac{t_2 - t_1}{T^-_{first}} \right\rceil + \pi, \left\lceil \frac{t_2 - t_1}{T_{min}} \right\rceil\right\}. \end{aligned}$$

Proof To prove the accuracy upper bound, we start from

$$\begin{aligned}\#b_p(t_2) - \#b_p(t_1) &\le b^{max}(t_2) - b^{min}(t_1)\\ &\le \left(b^{max}(t_2) - b^{max}(t_1)\right) + \left(b^{max}(t_1) - b^{min}(t_1)\right).\end{aligned} \tag{59}$$

Both terms of (59) can be bounded by applying Lemma 13 and Theorem 2, respectively, which yields

$$b^{max}(t_2) - b^{max}(t_1) \le \left\lceil \frac{t_2-t_1}{T^-_{first}} \right\rceil \tag{60}$$

$$b^{max}(t_1) - b^{min}(t_1) \quad \le \pi, \tag{61}$$

thus yielding

$$\#b_p(t_2) - \#b_p(t_1) \le \left\lceil \frac{t_2-t_1}{T^-_{first}} \right\rceil + \pi. \tag{62}$$

Moreover, from Lemma 14, it follows that

$$\#b_p(t_2) - \#b_p(t_1) \le \left\lceil \frac{t_2-t_1}{T_{min}} \right\rceil. \tag{63}$$

A combination of both bounds (62) and (63) leads to

$$\#b_p(t_2) - \#b_p(t_1) \le \min\left\{\left\lceil \frac{t_2-t_1}{T^-_{first}} \right\rceil + \pi, \left\lceil \frac{t_2-t_1}{T_{min}} \right\rceil\right\}$$

The upper bound follows.

To prove the accuracy lower bound, let $k = \#b_p(t_1) \ge 1$ and $N \ge 0$, such that, $k + N = \#b_p(t_2)$. Clearly such a k and N exist since p has sent tick 1 by t_1. By the definition of k and N,

$$t_{p,k} \le t_1 \tag{64}$$

$$t_{p,k+N+1} > t_2. \tag{65}$$

For $k \ge 2$ we can apply Lemma 16 together with (64) and (65), yielding

$$\begin{aligned} t_2 - t_1 &< t_{p,k+N+1} - t_{p,k} \qquad (66)\\ &\le (N+2)T_P + T_{QS}\\ \Rightarrow N &> \frac{t_2 - t_1 - T_{QS} - 2T_P}{T_P}\\ \Rightarrow N &\ge \left\lfloor \frac{t_2 - t_1 - T_{QS} - T_P}{T_P} \right\rfloor + 1. \qquad (67)\end{aligned}$$

For $k \ge 1$, we apply Lemma 17 to (66), which yields

$$\begin{aligned} t_2 - t_1 &< (N+1)T_P + T_{BS}(k)\\ \Rightarrow N &> \frac{t_2 - t_1 - T_{BS}(k) - T_P}{T_P}\\ \Rightarrow N &\ge \left\lfloor \frac{t_2 - t_1 - T_{BS}(k)}{T_P} \right\rfloor + 1. \qquad (68)\end{aligned}$$

Combining the bounds (67), (68) and the trivial bound 0 yields (with $\#b_p(t_1) = k$)

$$\begin{aligned}\#b_p(t_2) - \#b_p(t_1) &= N\\ &\ge \begin{cases} \max\left\{0, \left\lfloor \frac{t_2-t_1-T_{BS}(1)}{T_P} \right\rfloor + 1\right\} & \text{if } k = 1,\\ \max\left\{0, \left\lfloor \frac{t_2-t_1-\min\{T_{BS}(k), T_{QS}+T_P\}}{T_P} \right\rfloor + 1\right\} & \text{if } k \ge 2.\end{cases}\end{aligned} \tag{69}$$

Note that the term $\min\{T_{BS}(k), T_{QS} + T_P\}$ for $k \ge 2$ in (69) accounts for the fact that correct nodes may be synchronized very tightly after booting (within $T_{BS}(k)$), such that $T_{QS} + T_P$ would be too conservative. From the definition of $T_{BS}(k)$ in (42), however, it follows that the initial synchrony from booting usually cannot be maintained: For $T^-_{first} < T_P$, which is typically the case in real systems, we obtain $\lim_{k\to\infty} T_{BS}(k) = \infty$. Thus, for some k_0, $\forall k \ge k_0 : T_{QS} + T_P < T_{BS}(k)$, i.e., the constant bound from (QS) will be tighter, which prevents the nodes from drifting apart further.

In case k is not known, a valid bound is the minimum of all lower bounds. By the reasons given above, it is not conservative to utilize the bound $\min\{T_{BS}(k), T_{QS} + T_P\} \le T_{QS} + T_P$ for $k \ge 2$, yielding,

$$\begin{aligned}&\#b_p(t_2) - \#b_p(t_1)\\ &\quad = N \ge 1 + \min\left\{\left\lfloor \frac{t_2-t_1-T_{BS}(1)}{T_P} \right\rfloor, \left\lfloor \frac{t_2-t_1-(T_{QS}+T_P)}{T_P} \right\rfloor\right\}\end{aligned}$$

The lower bound follows.

The theorem follows. □

Note that by Constraint 3, $T^-_{first} \ge T_{min}$. Thereby the term $\left\lceil \frac{t_2-t_1}{T^-_{first}} \right\rceil + \pi$ becomes asymptotically determining for the upper bound for large enough $t_2 - t_1$. From this and the lower bound, we immediately obtain the long-term frequency bounds of $\left[1/T_P, 1/T^-_{first}\right]$.

Our final Theorems 4 and 6 establish bounds on the size of the local and remote pipeline. We start with a technical Lemma 18, which bounds the maximum time some tick can exist in the system before it is eliminated by the Diff-Gate in all pipepairs associated with correct nodes.

Lemma 18 *If the first correct node has sent tick $k \ge 2$ by time t, then every correct node $p \in C$ has removed tick $k-2$ from all its pipepairs corresponding to correct nodes $q \in C$ by time $t + T_{del}$, with*

$$T_{del} := T_{QS} + \max\{\tau^+_{loc}, \tau^+_{rem}\} + \tau^+_{Diff}, \tag{70}$$

or, which is equivalent

$$\#d_{p,q}(t + T_{del}) \ge b^{max}(t) - 2.$$

Proof See Lemma 18 in Appendix A. □

Theorem 4 (Local pipeline size bound). *For every pair of distinct correct nodes $p, q \in C$, $\#s^{loc}_{p,q}(t)$ is bounded by S_{loc}, with*

$$S_{loc} := \left\lfloor \frac{T_{del} - \tau^-_{loc}}{T^-_{first}} \right\rfloor + 3. \tag{71}$$

Proof Choose two arbitrary distinct correct nodes p, q and consider $\#s^{loc}_{p,q}(t)$. We distinguish between two cases for t: (i) $t \geq t_{first,2} + T_{del}$ and (ii) $t < t_{first,2} + T_{del}$.
ad (i):

$$\begin{aligned}
\#s^{loc}_{p,q}(t) &= \#r^{loc}_{p,q}(t) - \#d_{p,q}(t)\\
&\leq \#b_p(t-\tau^-_{loc}) - \#d_{p,q}(t)\\
&\leq b^{max}(t-\tau^-_{loc}) - \#d_{p,q}(t)\\
&= \left(b^{max}(t-\tau^-_{loc}) - b^{max}(t-T_{del})\right)\\
&\quad + \left(b^{max}(t-T_{del}) - \#d_{p,q}(t)\right)\\
&\leq \left\lceil \frac{T_{del}-\tau^-_{loc}}{T^-_{first}} \right\rceil + 2\\
&\leq S_{loc}, \qquad (72)
\end{aligned}$$

where (72) follows from applying Lemmas 13 and 18.
ad (ii): Since $\#s^{rem}_{p,q}(t) \leq \#r^{rem}_{p,q}(t) + 1$ must always hold, because the local pipe may contain at most all ticks received so far plus the initial tick 0, we obtain

$$\begin{aligned}
\#s^{loc}_{p,q}(t) &\leq \#r^{loc}_{p,q}(t) + 1\\
&\leq \#b_p(t-\tau^-_{loc}) + 1\\
&\leq b^{max}(t-\tau^-_{loc}) + 1\\
&\leq b^{max}(t_{first,2} + T_{del} - \tau^-_{loc}) + 1\\
&\leq \Big(b^{max}(t_{first,2} + T_{del} - \tau^-_{loc})\\
&\quad - b^{max}(t_{first,2})\Big) + b^{max}(t_{first,2}) + 1 \qquad (73)\\
&\leq \left\lfloor \frac{T_{del}-\tau^-_{loc}}{T^-_{first}} \right\rfloor + 3 \qquad (74)\\
&\leq S_{loc},
\end{aligned}$$

where (74) follows from Lemma 12.
The theorem follows in both cases. □

In order to bound the size of the remote pipelines, we can use exactly the same derivation based on Lemma 18 as used in Theorem 4 (with remote delays instead of local delays) to obtain the following Theorem 5.

Theorem 5 *For every pair of distinct correct nodes $p, q \in C$, $\#s^{rem}_{p,q}(t)$ is bounded by $\left\lfloor \frac{T_{del}-\tau^-_{rem}}{T^-_{first}} \right\rfloor + 3$.*

However, the resulting bound is overly large in most parameter settings: In order to maximize $\#s^{rem}_{p,q}(t)$, we need a scenario where the local node p is slow and the remote node q is fast. The time T_{del} established by Lemma 18 is too conservative for this case, however, since it actually considers q being slow. Lemma 19 provides a refined result.

Lemma 19 *If $k := b^{max}(t) \geq 2$, then for every pair of correct nodes $p, q \in C$, with*

$$\begin{aligned}
T^{loc}_{del} &:= T_{QS} + \tau^+_{loc} + \tau^+_{Diff}, \qquad (75)\\
t' &:= t + T^{loc}_{del} - \tau^+_{rem} - \tau^+_{Diff},\\
k' &:= \#b_q(t'),
\end{aligned}$$

the following holds:

(a) *If $k' \geq k-1$, then tick $k-2$ is removed from pipepair (p, q) by time $t + T^{loc}_{del}$, i.e.,*

$$\#d_{p,q}(t + T^{loc}_{del}) \geq b^{max}(t) - 2.$$

(b) *If $k' \leq k-2$, then tick $k'-1$ is removed from pipepair (p, q) by time $t + T^{loc}_{del}$, i.e.,*

$$\#d_{p,q}(t + T^{loc}_{del}) \geq \#b_q(t') - 1.$$

Proof To prove case (a), assume $k' \geq k-1$, which implies $t_{q,k-1} \leq t'$. Hence,

$$t_{q,k-1} + \tau^+_{rem} + \tau^+_{Diff} \leq t' + \tau^+_{rem} + \tau^+_{Diff} = t + T^{loc}_{del},$$

and, by (QS) and our assumption $t_{first,k} \leq t$,

$$\begin{aligned}
t_{p,k-1} + \tau^+_{loc} + \tau^+_{Diff} &\leq t_{first,k} + T_{QS} + \tau^+_{loc} + \tau^+_{Diff}\\
&\leq t + T^{loc}_{del}. \qquad (76)
\end{aligned}$$

We can now apply Lemma 5, which reveals that tick $k-2$ is removed from the pipepair (p, q) by time $t_{rmv,k-2}$, with

$$\begin{aligned}
t_{rmv,k-2} &\leq \max\{t_{p,k-1} + \tau^+_{loc}, t_{q,k-1} + \tau^+_{rem}\} + \tau^+_{Diff}\\
&\leq t + T^{loc}_{del}
\end{aligned}$$

as asserted.

To prove case (b), assume $k' \leq k-2$, which implies $t_{q,k-1} > t'$ and $t_{q,k'} \leq t'$. Hence,

$$\begin{aligned}
t_{q,k'} + \tau^+_{rem} + \tau^+_{Diff} &\leq t' + \tau^+_{rem} + \tau^+_{Diff} = t + T^{loc}_{del},\\
t_{q,k-1} + \tau^+_{rem} + \tau^+_{Diff} &> t' + \tau^+_{rem} + \tau^+_{Diff} = t + T^{loc}_{del}.
\end{aligned}$$

Since (76) also holds in case (b) and trivially $t_{p,k'} \leq t_{p,k-1}$, we find

$$t_{p,k'} + \tau^+_{loc} + \tau^+_{Diff} \leq t + T^{loc}_{del}.$$

We can again apply Lemma 5, which reveals that tick $k'-1$ is removed from the pipepair (p, q) by time $t_{rmv,k'-1}$, with

$$\begin{aligned}
t_{rmv,k'-1} &\leq \max\{t_{p,k'} + \tau^+_{loc}, t_{q,k'} + \tau^+_{rem}\} + \tau^+_{Diff}\\
&\leq t + T^{loc}_{del}.
\end{aligned}$$

as asserted. The lemma follows. □

Now we can establish our final major Theorem 6.

Theorem 6 (Remote pipeline size bound). *For every pair of distinct correct nodes $p, q \in C$, $\#s^{rem}_{p,q}(t)$ is bounded by S_{rem}, with $S_{rem} :=$*

$$\max\left\{\left\lfloor\frac{T^{loc}_{del}-\tau^-_{rem}}{T^-_{first}}\right\rfloor+3, \left\lceil\frac{\tau^+_{rem}-\tau^-_{rem}+\tau^+_{Diff}}{T_{min}}\right\rceil+1\right\}.$$

Proof Choose two arbitrary distinct correct nodes p, q and consider $\#s^{rem}_{p,q}(t)$. Since we will apply Lemma 19, we distinguish the following cases:
Case (a): Suppose it holds that $t \geq t_{first,2}+T^{loc}_{del}$ as well as $\#b_q(t-\tau^+_{rem}-\tau^+_{Diff}) \geq b^{max}(t-T^{loc}_{del})-1$. Then, we find

$$\begin{aligned}\#s^{rem}_{p,q}(t) &= \#r^{rem}_{p,q}(t)-\#d_{p,q}(t)\\ &\leq \#b_q(t-\tau^-_{rem})-\#d_{p,q}(t)\\ &\leq b^{max}(t-\tau^-_{rem})-\#d_{p,q}(t)\\ &= \left(b^{max}(t-\tau^-_{rem})-b^{max}(t-T^{loc}_{del})\right)\\ &\quad+\left(b^{max}(t-T^{loc}_{del})-\#d_{p,q}(t)\right)\\ &\leq \left\lceil\frac{T^{loc}_{del}-\tau^-_{rem}}{T^-_{first}}\right\rceil+2\\ &\leq S_{rem}, \end{aligned} \tag{77}$$

where (77) follows from applying Lemmas 13 and 19.
Case (b): Suppose it holds that $t \geq t_{first,2}+T^{loc}_{del}$, but now $\#b_q(t-\tau^+_{rem}-\tau^+_{Diff}) \leq b^{max}(t-T^{loc}_{del})-2$. Then, we find

$$\begin{aligned}\#s^{rem}_{p,q}(t) &= \#r^{rem}_{p,q}(t)-\#d_{p,q}(t)\\ &\leq \#b_q(t-\tau^-_{rem})-\#d_{p,q}(t)\\ &= \#b_q(t-\tau^-_{rem})-\#b_q(t-\tau^+_{rem}-\tau^+_{Diff})\\ &\quad+\#b_q(t-\tau^+_{rem}-\tau^+_{Diff})-\#d_{p,q}(t)\\ &\leq \left\lceil\frac{\tau^+_{rem}-\tau^-_{rem}+\tau^+_{Diff}}{T_{min}}\right\rceil+1\\ &\leq S_{rem}, \end{aligned} \tag{78}$$

where (78) follows from Lemmas 14 and 19. Note that $\tau^+_{rem}-\tau^-_{rem}+\tau^+_{Diff}$ is always non-negative.
Case (c): Finally suppose that $t < t_{first,2}+T^{loc}_{del}$. Since $\#s^{rem}_{p,q}(t) \leq \#r^{rem}_{p,q}(t)+1$ must always hold, because the remote pipe may contain at most all ticks received so far plus the initial tick 0, we obtain

$$\begin{aligned}\#s^{rem}_{p,q}(t) &\leq \#r^{rem}_{p,q}(t)+1\\ &\leq \#b_q(t-\tau^-_{rem})+1\\ &\leq b^{max}(t-\tau^-_{rem})+1\\ &\leq b^{max}(t_{first,2}+T^{loc}_{del}-\tau^-_{rem})+1\\ &\leq \Big(b^{max}(t_{first,2}+T^{loc}_{del}-\tau^-_{rem})\\ &\qquad -b^{max}(t_{first,2})\Big)+b^{max}(t_{first,2})+1 \qquad (79)\\ &\leq \left\lfloor\frac{T^{loc}_{del}-\tau^-_{rem}}{T^-_{first}}\right\rfloor+3 \qquad (80)\\ &\leq S_{rem}, \end{aligned}$$

where (80) follows from Lemma 12.
The theorem follows in all cases. □

6 Conclusions

We introduced the DARTS clock generation approach, which has been derived from a well-known distributed fault-tolerant tick generation algorithm and adapted for direct implementation in hardware. Major modifications had to be applied to the original distributed algorithm in order to adapt to the inherent fine-grain parallelism and limited resources of VLSI hardware implementations. DARTS provides a set of local clock signals, driving the subsystems of a SoC, for example, which are closely synchronized to each other. Our approach does not need quartz oscillators or the like, is guaranteed to start initially, and generates a clock frequency that automatically adapts to the current operating conditions.

The resulting algorithm (and its synchronization precision) are only weakly dependent on the implementation technology and the actual placement and routing on a chip: It just requires that the ratio of certain path delays, rather than the delays themselves, satisfy a few moderate constraints. The algorithm itself depends on these constraints only via the number of stages used in the elastic pipelines. Hence, there is usually no need to modify the algorithm when using a different implementation technology and/or a different placement & routing in an SoC.

We also provided a rigorous correctness proof and a worst case performance analysis, which employs a novel framework for the specification and analysis of distributed algorithms that are directly implementable in clockless digital logic. It shows that a system incorporating $n \geq 3f+2$ clock generation units (TG-Algs) can cope with up to f Byzantine faulty TG-Algs. Our proof rests on simple properties of some elementary building blocks only, which can be verified by digital design tools. Hence, the correctness of a system of any size n can be guaranteed if the low-level building blocks are implemented correctly. Note that a comparable result cannot be established via model checking.

Backed up by the lessons learned in the DARTS project, and encouraged by the conclusions of a recent Dagstuhl seminar [9] devoted to the topic, we feel justified to claim that the proposed modeling framework, the building blocks, and the problems solved in DARTS are paradigmic for fault-tolerant clockless algorithms in VLSI in general. We will conclude our paper with a few arguments in favor of this claim.

In classic clockless VLSI circuits, each module handshakes with its predecessor and its successor modules in a "wait-for-all" manner, i.e., it waits until it has received a valid handshake signal from all its predecessor modules before it generates the next handshake signal for its predecessor modules. Clearly, this approach is infeasible if some modules may fail and hence generate erroneous or no handshake signals. In this case, however, it is natural to replace the "wait-for-all" by thresholds: Instead of waiting for all predecessors, a module only waits for a sufficiently large subset of its predecessors to complete the handshaking. Still, for repeated threshold-based handshaking, it is instrumental not to mix up handshake signals that have not been used for passing the threshold earlier with "fresh" ones. Obviously, it is exactly this kind of behavior that is encountered in a single DARTS TG-Alg module: It generates the next tick if a sufficiently large number of its predecessor modules (i.e., other TG-Alg modules) have generated a tick, and never mixes up old and new ticks.

As a consequence, we are convinced that both (i) the building blocks of DARTS, and (ii) the switch from transition logic (tick broadcast, elastic pipelines), which is typical for standard clockless circuits, to state logic (PCSGs and threshold modules, which are standard synchronous circuits), and back to transition logic, are not specific to DARTS but paradigmic for fault-tolerant clockless circuits in general [28].

Part of our current and future research is devoted to further substantiating this claim: We recently completed some work on an important building block for a self-stabilizing variant of DARTS, which will allow to dismiss the simultaneous booting restriction and to transparently recover from transient failures, at the price of higher circuit complexity. Moreover, we started working on how to make other fault-tolerant distributed algorithms, including consensus, amenable to a direct implementation in digital hardware. Needless to say, our modeling framework, which captures all the peculiarities of such systems without unnecessary overhead, proved instrumental for all this work.

Acknowledgments The contributions of Johann Vilanek (preliminary simulations and experiments, and Diff-Gate design), Markus Ferringer (FPGA prototype), Thomas Handl (tools and library setup), and Andreas Dielacher (pipelined DARTS) are gratefully acknowledged. Valuable feedback on the design and implementation of the TG-Algs, which were primarily conducted by Gottfried Fuchs and Gerald Kempf, was provided by Andreas Steininger and Josef Widder. We would further like to thank the anonymous reviewers for their valuable feedback.

Appendix A: Proof of technical lemmas

In this appendix, we provide the proofs of some technical lemmas, which have been omitted from the main text in order to improve readability.

Lemma 2 *If, for a correct node $p \in C$ and a different correct node $q \in C \setminus \{p\}$, at time t it holds that*

$$(\#r^{loc}_{p,q}(t) = k) \wedge (\#s^{loc}_{p,q}(t) = 1)$$

for some $k \geq 1$, then it must hold that

$$\#r^{loc}_{p,q}(t - \tau^-_{Diff}) \geq k \text{ and}$$
$$\#r^{rem}_{p,q}(t - \tau^-_{Diff}) \quad \geq k.$$

Proof

$$\begin{aligned}&(\#r^{loc}_{p,q}(t) = k) \wedge (\#s^{loc}_{p,q}(t) = 1)\\ &\quad \equiv (\#r^{loc}_{p,q}(t) = k) \wedge (\#r^{loc}_{p,q}(t) - \#d_{p,q}(t) = 1)\\ &\quad \Rightarrow \#d_{p,q}(t) = k - 1 \end{aligned} \tag{81}$$

Let $t_{rmv,k-1}$ be the time tick $k-1$ is removed from the pipepair (p, q). From (81) it follows that

$$t_{rmv,k-1} \leq t. \tag{82}$$

Now assume by contradiction that

$$\begin{aligned}&\#r^{loc}_{p,q}(t - \tau^-_{Diff}) < k \text{ or}\\ &\#r^{rem}_{p,q}(t - \tau^-_{Diff}) < k.\end{aligned} \tag{83}$$

Denoting by $t_{loc,k}$ (respectively $t_{rem,k}$) the time the local (respectively remote) tick k is received in the pipepair (p, q), it follows that

$$t_{loc,k} > t - \tau^-_{Diff} \text{ respectively} \tag{84}$$

$$t_{rem,k} > t - \tau^-_{Diff}. \tag{85}$$

Combination of (82) with (84) (respectively (85)) yields

$$t_{rmv,k-1} < t_{loc,k} + \tau^-_{Diff} \text{ respectively} \tag{86}$$

$$t_{rmv,k-1} < t_{rem,k} + \tau^-_{Diff}. \tag{87}$$

From the behavioral specification of the Diff-gate, however, we know that

$$t_{rmv,k-1} \geq t_{loc,k} + \tau^-_{Diff} \text{ and}$$
$$t_{rmv,k-1} \quad \geq t_{rem,k} + \tau^-_{Diff},$$

contradicting (86) and (87). The lemma follows. □

Lemma 7 *For all correct nodes p and ticks $k \geq 1$: If there exists a set Q of correct nodes with $|Q| \geq 2f + 1$, such that, for all $q \in Q$:*

(i) *all nodes q send tick k by $t_{Q,k}$,*
(ii) *p sends tick k by $t_{p,k}$,*

then p sends tick $k+1$ by $t' := \max\{t_{p,k} + \tau^+_{loc} + \tau^+_{Diff} + \tau^+_{GEQ}, t_{p,k} + \tau^+_{loc} + \tau^+_{GR}, t_{Q,k} + \tau^+_{rem} + \tau^+_{Diff} + \tau^+_{GEQ}\} + \tau^+_{TH}$.

Proof Wlog. assume that $k \in \mathbb{N}_{even}$. We distinguish two cases:

(a) Suppose that $\exists q \in Q : \#r^{loc}_{p,q}(t') > k$. Since $\#r^{loc}_{p,q}(t') \leq \#b_p(t')$, as no tick can be locally received if it has not been sent before, $\#b_p(t') > k$ must hold. Thus, p has sent tick $k+1$ by t'. The lemma follows.

(b) Otherwise, assume that

$$\forall q \in Q : \#r^{loc}_{p,q}(t') \leq k. \tag{88}$$

By applying Lemma 5, we conclude that tick $k-1$ is removed from all of p's pipepairs (for all $q \in Q$) by

$$t_{del,k-1} := \max\{t_{p,k} + \tau^+_{loc}, t_{Q,k} + \tau^+_{rem}\} + \tau^+_{Diff}.$$

Clearly, since all nodes $q \in Q$ are correct and have sent tick k by $t_{Q,k}$, tick k must be received in the remote pipe of (p,q) by $t_{rcv,Q} := t_{Q,k} + \tau^+_{rem}$. Furthermore, p must receive tick k in all its local pipes by $t_{rcv,p} := t_{p,k} + \tau^+_{loc}$. Consequently, at $t_{rcv} := \max\{t_{rcv,p}, t_{rcv,Q}, t_{del,k-1}\} \leq \max\{t_{p,k} + \tau^+_{loc}, t_{Q,k} + \tau^+_{rem}\} + \tau^+_{Diff}$ with $t_{rcv} \leq t'$, it holds that:

$$\#r^{loc}_{p,q}(t_{rcv}) \geq k \qquad \#r^{rem}_{p,q}(t_{rcv}) \geq k \qquad \#d_{p,q}(t) \geq k-1 \tag{89}$$

By combination of (89) with (88), it holds that for all $\xi \in [t_{rcv}, t']$:

$$\#r^{loc}_{p,q}(\xi) = k. \tag{90}$$

Furthermore,

$$\begin{aligned}
&\widetilde{\mathcal{P}}^{GEQ,e}_{p,q}(\xi)\\
&\quad \equiv (\#r^{rem}_{p,q}(\xi) \geq \#r^{loc}_{p,q}(\xi))\\
&\qquad \wedge (\#r^{loc}_{p,q}(\xi) \in \mathbb{N}_{even}) \wedge (\#s^{loc}_{p,q}(\xi) = 1)\\
&\quad \equiv (\#r^{rem}_{p,q}(\xi) \geq k) \wedge (\#r^{loc}_{p,q}(\xi) - \#d_{p,q}(\xi) = 1)\\
&\quad \equiv (\#d_{p,q}(\xi) = k-1).
\end{aligned} \tag{91}$$

Assuming $\exists q \in Q : \#d_{p,q}(\xi) > k-1$ implies $\#r^{loc}_{p,q}(\xi) > k$, contradicting (90). Thus $\forall q \in Q : \#d_{p,q}(\xi) = k-1$ must hold. Therefore, $\forall q \in Q : \widetilde{\mathcal{P}}^{GEQ,e}_{p,q}(\xi)$ is true for $\xi \in [t_{rcv}, t']$.
By the algorithm (specification of the PCSG to threshold module channels), $\forall q \in Q : \widetilde{GEQ}^{e}_{p,q}(\xi)$ is true for $\xi \in [t_{rcv} + \tau^+_{GEQ}, t']$. Again by the algorithm (specification of the threshold modules),

$$\widetilde{THGEQ}^{e}_{p}(\xi) \tag{92}$$

is true for any time ξ within $[t_{rcv} + \tau^+_{GEQ} + \tau^+_{TH}, t'] \subseteq$

$$[\max\{t_{p,k} + \tau^+_{loc}, t_{Q,k} + \tau^+_{rem}\} + \tau^+_{Diff} + \tau^+_{GEQ} + \tau^+_{TH}, t'].$$

It remains to be shown that the disabling path cannot inhibit the generation of tick $k+1$ at p. For the sake of contradiction, assume that the disabling path can enforce $t_{p,k+1} > t'$.
Because of the lemma's assumption (ii), tick k must eventually be received in all of p's local pipes corresponding to correct nodes r, i.e., $\forall r \in C \setminus \{p\} : \#r^{loc}_{p,r}(\xi) \geq k$, for $\xi \in [t_{p,k} + \tau^+_{loc}, t']$. Since tick $k+1$ is not generated by t', we actually have $\#r^{loc}_{p,r}(\xi) = k$. As $k \in \mathbb{N}_{even}$, this implies

$$\forall r \in C \setminus \{p\} : \neg\left(\widetilde{\mathcal{P}}^{GR,o}_{p,r}(\xi) \vee \widetilde{\mathcal{P}}^{GEQ,o}_{p,r}(\xi)\right).$$

Consequently, by the algorithm (specification of the PCSG to threshold module channels and the threshold modules), together with the fact that there are only up to f faulty nodes, who cannot reach a threshold themselves,

$$\neg\left(\widetilde{THGR}^{o}_{p}(\xi) \vee \widetilde{THGEQ}^{o}_{p}(\xi)\right) \tag{93}$$

for $\xi \in [t_{p,k} + \tau^+_{loc} + \max\{\tau^+_{GR}, \tau^+_{GEQ}\} + \tau^+_{TH}, t']$.
Combining (92) and (93) and noting that $\max\{\tau^+_{GEQ} + \tau^+_{Diff}, \max\{\tau^+_{GR}, \tau^+_{GEQ}\}\} = \max\{\tau^+_{GEQ} + \tau^+_{Diff}, \tau^+_{GR}\}$, it is apparent that p must send tick $k+1$ by t', providing the required contradiction. The lemma follows.

The lemma follows in both cases. □

Lemma 12 (Maximum Increase of b^{max} in $(t_{first,k}, t]$). *If the first correct node sends tick $k \geq 1$ at $t_{first,k}$, then*

$$\forall t > t_{first,k} : b^{max}(t) - b^{max}(t_{first,k}) \leq \left\lfloor \frac{t - t_{first,k}}{T^-_{first}} \right\rfloor$$

or, equivalently: The number N of ticks sent by the correct first node in the interval $I = (t_{t_{first,k}}, t]$ is upper bounded by $\left\lfloor \frac{t - t_{first,k}}{T^-_{first}} \right\rfloor$.

Proof Let $t_{first,j}$ be the time when the first correct node sends tick j, and assume by contradiction that

$$N \geq \left\lfloor \frac{t - t_{first,k}}{T^-_{first}} \right\rfloor + 1. \tag{94}$$

According to the definition of N,

$$t_{first,k+N} \leq t. \tag{95}$$

By applying Lemma 10 to $t_{first,k}$ and $t_{first,k+N}$, and recalling $x < \lfloor x \rfloor + 1$ for all real x, we find

$$\begin{aligned} t_{first,k+N} - t_{first,k} &\geq N T^-_{first} \\ &\geq \left(\left\lfloor \frac{t - t_{first,k}}{T^-_{first}} \right\rfloor + 1 \right) T^-_{first} \\ &> t - t_{first,k}. \end{aligned} \tag{96}$$

Clearly, (96) contradicts (95). The lemma follows. □

Lemma 13 (Maximum Increase of b^{max} in $\left(t, t'\right]$).

$$\forall t' > t : b^{max}(t') - b^{max}(t) \leq \left\lceil \frac{t' - t}{T^-_{first}} \right\rceil$$

or, equivalently: The number N of ticks sent by the correct first node in the interval $I = \left(t, t'\right]$ *is upper bounded by* $\left\lceil \frac{t'-t}{T^-_{first}} \right\rceil$.

Proof Let $t_{first,j}$ the time when the first correct node sends tick j. We distinguish two cases: (i) $N \geq 1$, that is, $\exists k : t_{first,k+1} \in I$ and (ii) $N = 0$.
ad (i): Assume by contradiction that

$$N \geq \left\lceil \frac{t' - t}{T^-_{first}} \right\rceil + 1. \tag{97}$$

According to the definition of N and the assumption $N \geq 1$, there must be some k such that

$$t_{first,k+1} > t \tag{98}$$

$$t_{first,k+N} \leq t'. \tag{99}$$

By applying Lemma 10 to $t_{first,k+1}$ and $t_{first,k+N}$, we find

$$\begin{aligned} t_{first,k+N} - t &> t_{first,k+N} - t_{first,k+1} \\ &\geq (N-1) T^-_{first} \\ &\geq \left\lceil \frac{t' - t}{T^-_{first}} \right\rceil T^-_{first} \\ &\geq t' - t. \end{aligned} \tag{100}$$

Clearly, (100) contradicts (99). The lemma follows.
ad (ii): Obviously $N = 0 \leq \left\lceil \frac{t'-t}{T^-_{first}} \right\rceil$ holds for $t' > t$. The lemma follows.
The lemma follows in both cases. □

Lemma 14 (Maximum Increase of $\#b_p$ in $\left(t, t'\right]$).

$$\forall t' > t : \#b_p(t') - \#b_p(t) \leq \left\lceil \frac{t' - t}{T_{min}} \right\rceil$$

or, equivalently: The number N of ticks sent by the correct node p in the interval $I = \left(t, t'\right]$ *is upper bounded by* $\left\lceil \frac{t'-t}{T_{min}} \right\rceil$.

Proof Let $t_{p,j}$ the time when node p sends tick j. We distinguish two cases: (i) $N \geq 1$, i.e., $\exists k : t_{p,k+1} \in I$ and (ii) $N = 0$.
ad (i): Assume by contradiction that

$$N \geq \left\lceil \frac{t' - t}{T_{min}} \right\rceil + 1. \tag{101}$$

According to the definition of N and the assumption $N \geq 1$, there must be some k such that

$$t_{p,k+1} > t \tag{102}$$

$$t_{p,k+N} \leq t'. \tag{103}$$

By applying Lemma 4 to $t_{p,k+1}$ and $t_{p,k+N}$, we find

$$\begin{aligned} t_{p,k+N} - t &> t_{p,k+N} - t_{p,k+1} \\ &\geq (N-1) T_{min} \\ &\geq \left\lceil \frac{t' - t}{T_{min}} \right\rceil T_{min} \\ &\geq t' - t. \end{aligned} \tag{104}$$

Clearly, (104) contradicts (102). The lemma follows.
ad (ii): Obviously $N = 0 \leq \left\lceil \frac{t'-t}{T_{min}} \right\rceil$ holds for $t' > t$ and the lemma follows.
The lemma follows in both cases. □

Lemma 18 *If the first correct node has sent tick $k \geq 2$ by time t, then every correct node $p \in C$ has removed tick $k-2$ from all its pipepairs corresponding to correct nodes $q \in C$ by time $t + T_{del}$, with*

$$T_{del} := T_{QS} + \max\{\tau^+_{loc}, \tau^+_{rem}\} + \tau^+_{Diff}, \tag{105}$$

or, which is equivalent

$$\#d_{p,q}(t + T_{del}) \geq b^{max}(t) - 2.$$

Proof Consider a pair of pipes (p, q) located at $p \in C$, corresponding to a different $q \in C$. Furthermore, assume that $b^{max}(t) \geq k$ holds at time t. We are interested in how much later tick $k-2$ is removed from this pipepair:

Clearly, $t_{first,k} \leq t$, and by (QS),

$$\begin{aligned} t_{p,k-1} &\leq t_{last,k-1} \\ &\leq t_{first,k} + T_{QS} \text{ and} \end{aligned} \tag{106}$$

$$t_{q,k-1} \leq t_{first,k} + T_{QS}. \tag{107}$$

We can now apply Lemma 5 in combination with (106) and (107), which reveals that tick $k-2$ is removed from the pipepair at p corresponding to q by time $t_{rmv,k-2}$, with

$$\begin{aligned} t_{rmv,k-2} &\leq \max\{t_{p,k-1} + \tau^+_{loc}, t_{q,k-1} + \tau^+_{rem}\} + \tau^+_{Diff} \\ &\leq t_{first,k} + T_{QS} + \max\{\tau^+_{loc}, \tau^+_{rem}\} + \tau^+_{Diff} \\ &= t_{first,k} + T_{del} \\ &\leq t + T_{del}. \end{aligned}$$

The lemma follows. □

References

1. Attiya, H., Herzberg, A., Rajsbaum, S.: Optimal clock synchronization under different delay assumptions. SIAM J. Comput. **25**(2), 369–389 (1996)
2. Bar-Noy, A., Dolev, D.: Consensus algorithms with one-bit messages. Distrib. Comput. **4**, 105–110 (1991)
3. Barros, J.C., Johnson, B.W.: Equivalence of the arbiter, the synchronizer, the latch, and the inertial delay. IEEE Trans. Comput. **32**(7), 603–614 (1983)
4. Baumann, R.: Soft errors in advanced computer systems. IEEE Des. Test Comput. **22**(3), 258–266 (2005)
5. Belluomini, W., Myers, C.J.: Verification of timed systems using posets. In: Computer Aided Verification, pp. 403–415 (1998)
6. Bhamidipati, R., Zaidi, A., Makineni, S., Low, K., Chen, R., Liu, K.-Y., Dalgrehn, J.: Challenges and methodologies for implementing high-performance network processors. Intel Technol. J. **6**(3), 83–92 (2002)
7. Black, D.L.: On the existince of delay-insensitive fair arbiters: trace theory and its limitations. Distrib. Comput. **1**, 205–225 (1986)
8. Chapiro, D.M.: Globally-Asynchronous Locally-Synchronous Systems. PhD thesis, Stanford University (1984)
9. Charron-Bost, B., Dolev, S., Ebergen, J., Schmid, U.: 08371 summary—fault-tolerant distributed algorithms on VLSI chips. In: Charron-Bost, B., Dolev, S., Ebergen, J., Schmid, U. (eds.) Fault-Tolerant Distributed Algorithms on VLSI Chips, number 08371 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2009. Schloss Dagstuhl—Leibniz-Zentrum fuer Informatik, Germany
10. Clarke, E.M.: Editorial: distributed computing issues in hardware design. Distrib. Comput. **1**, 185–186 (1986)
11. Constantinescu, C.: Trends and challenges in VLSI circuit reliability. IEEE Micro **23**(4), 14–19 (2003)
12. Dolev, D., Halpern, J.Y., Strong, H.R.: On the possibility and impossibility of achieving clock synchronization. J. Comput. Syst. Sci. **32**, 230–250 (1986)
13. Dolev, S., Haviv, Y.: Self-stabilizing microprocessors, analyzing and overcoming soft-errors. IEEE Trans. Comput. **55**(4), 385–399 (2006)
14. Dolev, S., Tzachar, N.: Brief announcment: Corruption resilient fountain codes. In: Taubenfeld, G. (ed.) Distributed Computing, Lecture Notes in Computer Science, vol. 5218, pp. 502–503. Springer, Berlin/Heidelberg (2008)
15. Dyer, C., Rodgers, D.: Effects on spacecraft & aircraft electronics. In: Proceedings ESA Workshop on Space Weather, ESA WPP-155, pp. 17–27. ESA, Nordwijk, The Netherlands (1998)
16. Ebergen, J.C.: A formal approach to designing delay-insensitive circuits. Distrib. Comput. **5**, 107–119 (1991)
17. Fairbanks, S.: Method and apparatus for a distributed clock generator, 2004. US patent no. US2004108876
18. Fairbanks, S., Moore, S.: Self-timed circuitry for global clocking. In: Proceedings of the Eleventh International IEEE Symposium on Asynchronous Circuits and Systems, pp. 86–96 (2005)
19. Ferri, C., Moreshet, T., Iris Bahar, R., Benini, L., Herlihy, M.: A hardware/software framework for supporting transactional memory in a MPSoC environment. SIGARCH Comput. Archit. News **35**(1), 47–54 (2007)
20. Ferringer, M., Fuchs, G., Steininger, A., Kempf, G.: VLSI Implementation of a Fault-Tolerant Distributed Clock Generation. In: IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT2006), pp. 563–571 (2006)
21. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. J. ACM **32**(2), 374–382 (1985)
22. Friedman, E.G.: Clock distribution networks in synchronous digital integrated circuits. Proc. IEEE **89**(5), 665–692 (2001)
23. Friedman, R., Mostefaoui, A., Rajsbaum, S., Raynal, M.: Asynchronous agreement and its relation with error-correcting codes. IEEE Trans. Comput. **56**(7), 865–875 (2007)
24. Fuchs, G.: Fault-Tolerant Distributed Algorithms for On-Chip Tick Generation: Concepts, Implementations and Evaluations. PhD thesis, Vienna University of Technology, Fakultät für Informatik (2009)
25. Fuchs, G., Függer, M., Steininger, A.: On the threat of metastability in an asynchronous fault-tolerant clock generation scheme. In: 15th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'09), pp. 127–136, Chapel Hill, N. Carolina, USA (2009)
26. Fuchs, G., Függer, M., Steininger, A., Zangerl, F.: Analysis of constraints in a fault-tolerant distributed clock generation scheme. In: 3rd International Workshop on Dependable Embedded Systems (WDES'06) (2006)
27. Fuchs, G., Steininger, A.: VLSI implementation of a distributed algorithm for fault-tolerant clock generation. J. Electr. Comput. Eng. **2011**, 23 (2011). doi:10.1155/2011/936712
28. Függer, M.: Analysis of On-Chip Fault-Tolerant Distributed Algorithms. PhD thesis, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-2, 1040 Vienna, Austria (2010)
29. Gadlage, M.J., Eaton, P.H., Benedetto, J.M., Carts, M., Zhu, V., Turflinger, T.L.: Digital device error rate trends in advanced CMOS technologies. IEEE Trans. Nucl. Sci. **53**(6), 3466–3471 (2006)
30. Grahsl, J., Handl, T., Steininger, A.: Exploring the usefulness of the gate-level stuck-at fault model for Muller C-elements. In: Proceedings 20. Workshop für Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TuZ'08), pp. 165–169, Vienna, Austria (2008)
31. Halpern, J.Y., Megiddo, N., Munshi, A.A.: Optimal precision in the presence of uncertainty. J. Complex. **1**(2), 170–196 (1985)
32. Hauck, S.: Asynchronous design methodologies: an overview. Proc. IEEE **83**(1), 69–93 (1995)
33. Hoyme, K., Driscoll, K.: Safebus. In: Proceedings IEEE/AIAA 11th Digital Avionics Systems Conference, pp. 68–73 (1992)
34. International technology roadmap for semiconductors (2007)
35. Jang, W., Martin, A.J.: SEU-tolerant QDI circuits. In: Proceedings 11th Int'l Symposium on Asynchronous Circuits and Systems (ASYNC'05), pp. 156–165 (2005)
36. Karnik, T., Hazucha, P., Patel, J.: Characterization of soft errors caused by single event upsets in CMOS processes. IEEE Trans. Dependable Secur. Comput. **1**(2), 128–143 (2004)
37. Kaynar, D.K., Lynch, N., Segala, R., Vaandrager, F.: Timed I/O automata: a mathematical framework for modeling and analyzing real-time systems. In: Proceedings 24th IEEE International Real-Time Systems Symposium (RTSS'03), vol. 00, 166–177 (2003)
38. Kieckhafer, R.M., Walter, C.J., Finn, A.M., Thambidurai, P.M.: The MAFT architecture for distributed fault tolerance. IEEE Trans. Comput. **37**, 398–405 (1988)
39. Kopetz, H., Grünsteidl, G.: TTP-A protocol for fault-tolerant real-time systems. Computer **27**(1), 14–23 (1994)
40. Koren, I., Koren, Z.: Defect tolerance in VLSI circuits: techniques and yield analysis. Proc. IEEE **86**(9), 1819–1838 (1998)
41. Lamport, L.: Buridan's principle. Technical report, SRI Technical Report (1984)
42. Lamport, L.: Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley, Boston (2002)
43. Lamport, L.: Arbitration-free synchronization. Distrib. Comput. **16**(2/3), 219–237 (2003)
44. Le Lann, G., Schmid, U.: How to implement a timer-free perfect failure detector in partially synchronous systems. Technical Report 183/1-127, Department of Automation, Technische Universität

Wien, January 2003. (Replaced by Research Report 28/2005, Institut für Technische Informatik, TU Wien, 2005.)
45. Lynch, N.: Distributed Algorithms. Morgan Kaufman, San Francisco (1996)
46. Maheshwari, A., Koren, I., Burleson, W.: Accurate estimation of Soft Error Rate (SER) in VLSI circuits. In: Proceedings of the 2004 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 377–385 (2004)
47. Marino, L.: General theory of metastable operation. IEEE Trans. Comput. **C-30**(2), 107–115 (1981)
48. Martin, A.J.: Compiling communicating processes into delay-insensitive VLSI circuits. Distrib. Comput. **1**, 226–234 (1986)
49. Martin, A.J.: The limitations to delay-insensitivity in asynchronous circuits. In: AUSCRYPT '90: Proceedings of the sixth MIT conference on Advanced research in VLSI, pp. 263–278. MIT Press, Cambridge, MA, USA (1990)
50. Maza, M.S., Aranda, M.L.: Analysis of clock distribution networks in the presence of crosstalk and groundbounce. In: Proceedings International IEEE Conference on Electronics, Circuits, and Systems (ICECS), pp. 773–776 (2001)
51. Maza, M.S., Aranda, M.L.: Interconnected rings and oscillators as gigahertz clock distribution nets. In: GLSVLSI '03: Proceedings of the 13th ACM Great Lakes symposium on VLSI, pp. 41–44. ACM Press (2003)
52. Metra, C., Francescantonio, S.D., Mak, T.M.: Implications of clock distribution faults and issues with screening them during manufacturing testing. IEEE Trans. Comput. **53**(5), 531–546 (2004)
53. Mitra, S., Seifert, N., Zhang, M., Shi, Q., Kim, K.S.: Robust system design with built-in soft-error resilience. IEEE Comput. **38**(5), 43–52 (2005)
54. Moscibroda, T., Mutlu, O.: Distributed order scheduling and its application to multi-core DRAM controllers. In: Proceedings of the 27th ACM Symposium on Principles of Distributed Computing (PODC'08), pp. 365–374, Toronto, Canada (2008)
55. Myers, C.J., Meng, T.H.Y.: Synthesis of timed asynchronous circuits. IEEE Trans. VLSI Syst. **1**(2), 106–119 (1993)
56. Nicolaidis, M.: GRAAL: a fault-tolerant architecture for enabling nanometric technologies. In: Proceedings 13th IEEE International On-Line Testing Symposium (IOLTS'07), pp. 255–255 (2007)
57. Normand, E.: Single-event effects in avionics. IEEE Trans. Nucl. Sci. **43**(2), 461–474 (1996)
58. Ostrovsky, R., Patt-Shamir, B.: Optimal and efficient clock synchronization under drifting clocks. In: PODC '99: Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing, pp. 3–12. ACM, New York, NY, USA (1999)
59. Palit, A.K., Meyer, V., Anheier, W., Schloeffel, J.: Modeling and analysis of crosstalk coupling effect on the victim interconnect using the ABCD network model. In: Proceedings of the 19th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'04), pp. 174–182 (2004)
60. Patt-Shamir, B., Rajsbaum, S.: A theory of clock synchronization (extended abstract). In: STOC '94: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of computing, pp. 810–819. ACM Press, New York, NY, USA (1994)
61. Polzer, T., Handl, T., Steininger, A.: A metastability-free multi-synchronous communication scheme for socs. In: Proceedings of the Stabilization, Safety, and Security of Distributed Systems, 11th International Symposium, SSS 2009, Lyon, France, November 3–6, 2009, pp. 578–592 (2009)
62. Powell, D., Arlat, J., Beus-Dukic, L., Bondavalli, A., Coppola, P., Fantechi, A., Jenn, E., Rabejac, C., Wellings, A.: GUARDS: a generic upgradable architecture for real-time dependable systems. IEEE Trans. Parallel Distrib. Syst. **10**(6), 580–599 (1999)
63. Ramanathan, P., Shin, K.G., Butler, R.W.: Fault-tolerant clock synchronization in distributed systems. IEEE Comput. **23**(10), 33–42 (1990)
64. Restle, P.J. et al.: A clock distribution network for microprocessors. IEEE J. Solid-State Circuits **36**(5), 792–799 (2001)
65. Rokicki, T., Myers, C.J.: Automatic verification of timed circuits. In: Computer Aided Verification, pp. 468–480 (1994)
66. Schmid, U.: How to model link failures: A perception-based fault model. In: Proceedings of the International Conference on Dependable Systems and Networks (DSN'01), pp. 57–66, Göteborg, Sweden (2001)
67. Schmid, U.: Keynote: distributed algorithms and VLSI. In: Proceedings of the 10th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS'08), Lecture Notes in Computer Science, vol. 5340, page 3, Detroit, USA, November 2008. Springer Verlag. (http://www.vmars.tuwien.ac.at/documents/extern/2467/sss08.pdf)
68. Schmid, U., Klasek, J., Mandl, T., Nachtnebel, H., Cadek, G.R., Kerö, N.: A network time interface M-module for distributing GPS-time over LANs. Real-Time Syst. **18**(1), 24–57 (2000)
69. Schmid, U., Steininger, A.: Dezentrale Fehlertolerante Taktgenerierung in VLSI Chips. Research Report 69/2004, Technische Universität Wien, Institut für Technische Informatik, 2004. International patent PCT WO2006/007619: EP 1769356, US 2009/0102534, ZL 200580024166.6, AT 501510
70. Seifert, N., Shipley, P., Pant, M.D., Ambrose, V., Gill, B.: Radiation-induced clock jitter and race. In: Proceedings 43rd Annual IEEE International Reliability Physics Symposium, pp. 215–222, 17–21 (2005)
71. Seitz, C.L.: System timing. In: Introduction to VLSI Systems, pp. 218–262. Addison Wesley, Boston (1980)
72. Semiat, Y., Ginosar, R.: Timing measurements of synchronization circuits. Int. Symp. Asynchr. Circuits Syst. **0**, 68 (2003)
73. Shivakumar, P., Kistler, M., Keckler, S.W., Burger, D., Alvisi, L.: Modeling the effect of technology trends on the soft error rate of combinational logic. In: Proceedings of International Conference on Dependable Systems and Networks, DSN, pp. 389–398 (2002)
74. Simons, B., Lundelius-Welch, J., Lynch, N.: An overview of clock synchronization. In: Simons, B., Spector, A. (eds.) Fault-Tolerant Distributed Computing, LNCS 448, pp. 84–96. Springer, Berlin (1990)
75. Srikanth, T.K., Toueg, S.: Optimal clock synchronization. J. ACM **34**(3), 626–645 (1987)
76. Stevens, K.S., Ginosar, R., Rotem, S.: Relative timing [asynchronous design]. IEEE Trans. VLSI Syst. **11**(1), 129–140 (2003)
77. Sutherland, I.E.: Micropipelines. Communications of the ACM, Turing Award, **32**(6), 720–738, June 1989. ISSN:0001-0782
78. Teehan, P., Greenstreet, M., Lemieux, G.: A survey and taxonomy of GALS design styles. IEEE Des. Test Comput. **24**(5), 418–428 (2007)
79. Thaker, D.D., Impens, F., Chuang, I.L., Amirtharajah, R., Chong, F.T.: Recursive TMR: scaling fault tolerance in the nanoscale era. IEEE Des. Test Comput. **22**(4), 298–305 (2005)
80. Verdel, T., Makris, Y.: Duplication-based concurrent error detection in asynchronous circuits: shortcomings and remedies. In: Proceedings 17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT 2002), pp. 345–353 (2002)
81. Widder, J., Le Lann, G., Schmid, U.: Failure detection with booting in partially synchronous systems. In: Proceedings of the 5th European Dependable Computing Conference (EDCC-5), LNCS, vol. 3463, pp. 20–37. Springer Budapest, Hungary (2005)
82. Widder, J., Schmid, U.: The theta-model: achieving synchrony without clocks. Distrib. Comput. **22**(1), 29–47 (2009)

83. Yakovlev, A., Lavagno, L., Sangiovanni-Vincentelli, A.: A unified signal transition graph model for asynchronous control circuit synthesis. In: Proceedings of the 1992 IEEE/ACM international conference on Computer-aided design (ICCAD'92), pp. 104–111. IEEE Computer Society Press, Los Alamitos, CA, USA (1992)

84. Yoneda, T., Kitai, T., Myers, C.J.: Automatic derivation of timing constraints by failure analysis. In: Proceedings 14th International Conference on Computer Aided Verification (CAV'02), Lecture Notes in Computer Science, vol. 2404, pp. 195–208. Springer, Berlin (2002)