FLINT+: A Runtime-Configurable Emulation-Based Stochastic Timing Analysis Framework

M. Weißbrich[a], L. Gerlach[a], H. Blume[a], A. Najafi[b], A. García-Ortiz[b], G. Payá-Vayá[a,*]

[a]*Institute of Microelectronic Systems, Leibniz Universität Hannover, Appelstraße 4, 30167 Hannover, Germany*
[b]*Institute of Electrodynamics and Microelectronics, Universität Bremen, Otto-Hahn-Allee 1, 28359 Bremen, Germany*

Abstract

ASICs for *Stochastic Computing* conditions are designed for higher energy-efficiency or performance by sacrificing computational accuracy due to intentional circuit timing violations. To optimize the stochastic behavior, iterative timing analysis campaigns have to be carried out for a variety of circuit timing corner cases. However, the application of common event-driven logic simulators usually leads to excessive analysis runtimes, increasing design time for hardware developers. In this paper, a gate-level netlist-oriented FPGA-based timing analysis framework is proposed, offering a runtime-configuration mechanism for emulating different timing corner cases in hardware without requiring multiple FPGA bitstreams. Logic gates are instrumented with a quantization-based delay model and a critical path selection algorithm is used to reduce the FPGA resource overhead. For an exemplary design space exploration of stochastic CORDIC units, speed-up factors of up to 48 for 10 ps or 476 for 100 ps timing quantization are achieved while maintaining timing behavior deviations lower than 1.5% or 4% to timing simulations, respectively.

Keywords: Stochastic Computing, Timing Analysis, Timing Behavior Emulation, FPGA, CORDIC

1. Introduction

In recent portable embedded *Computer Vision* systems, computational intense tasks with stringent performance constraints have to be performed within a very limited power budget of just a few Watts. To compete with these challenges, *Stochastic Computing* [1] has emerged as a design approach for energy-efficient systems. By relaxing specification constraints like operation frequency, supply voltage, temperature, etc., the design space of a processor architecture is extended. Thus, designers can exploit additional improvements in processing performance and energy efficiency while accepting occasional circuit timing violations and therefore incorrect computational results. A common characteristic of *Computer Vision* applications is that precise computations are not often necessary [2]. For example, during an analysis of typical edge detection or image filtering algorithms, a high noise tolerance and error-resilient operation is found, so there exists a margin for stochastic arithmetic errors occurring beyond the tight conventional limits for accurate circuit operation [3, 4].

One of the main challenges in *Stochastic Computing* is the increasing development time for optimizing the trade-offs between the power consumption, performance benefits and computational accuracy tolerance. Most frequently, iterative gate-level timing simulations are performed in order to evaluate the impact of path timing violations on arithmetic results and to optimize stochastic circuit techniques and mechanisms. These netlist simulations with annotated timing information are very time-consuming, so designers are forced to heavily constrain the completeness of timing analysis campaigns in order to obtain acceptable simulation runtimes.

In a previously published paper [5], an emulation-based timing analysis framework (FLINT+) is proposed to tackle the challenge of speeding-up *Stochastic Computing* analysis campaigns. The timing behavior of gate-level ASIC netlists is emulated using FPGA devices, taking advantage of high-speed parallel processing possible in hardware implementations.

This paper presents the extension of this framework and describes all implemented concepts of *Stochastic Computing* in detail, including a new case study to demonstrate the potential of the proposed FLINT+ framework. The main contributions are:

- A FPGA-synthesizable delay model for instrumenting gate-level ASIC netlists with timing behavior, optimized for carrying out timing analysis campaigns at high emulation speed.
- Implementation of a parameter configuration chain

[*]Corresponding author
Phone +49-511-762-19605
Fax +49-511-762-19601
Email addresses: weissbrich@ims.uni-hannover.de (M. Weißbrich), gerlach@ims.uni-hannover.de (L. Gerlach), blume@ims.uni-hannover.de (H. Blume), ardalan@item.uni-bremen.de (A. Najafi), agarcia@item.uni-bremen.de (A. García-Ortiz), guipava@ims.uni-hannover.de (G. Payá-Vayá)

mechanism suitable for configuring the timing behavior at runtime to offer adaptation to different stochastic timing conditions without requiring the generation of a new FPGA configuration bitstream.

- Usage of a critical path selection to collect only relevant logic gates for a desired worst-case timing instrumentation condition, decreasing the resource overhead and routing complexity of the FPGA design.
- Exhaustive analysis of the proposed framework in terms of timing emulation accuracy, speed-up to software simulations and FPGA resource utilization.
- A Case study in which the accuracy-performance stochastic design space of accurate and approximate CORDIC hardware implementations is explored by using the FLINT+ framework.

The contribution outline of this paper is given as follows: Section 2 presents a selection of related work for instrumenting netlists and speeding-up timing analyses. In Section 3, the proposed instrumentation model, configuration mechanism, iterative critical path selection, framework and toolchain are described in detail. Performance evaluation results are given in Section 4. The CORDIC stochastic design space exploration case study is presented in Section 5 to demonstrate the analysis potential of the proposed framework. Finally, in Section 6, a conclusion is given.

2. Related Work

Previously published approaches to instrument circuits with timing information are mainly designed for speeding-up fault injection and propagation campaigns. In Table 1, a selection of related work is presented with key features of the different implementations. A common feature of all related work is to perform a gate-level netlist analysis with higher speed compared to a standard timing simulation. First of all, there are techniques exploiting operation properties of event-driven logic simulators. In [6], propagation delays of combinational gate circuits are transformed into structural information by expanding the circuit topology. Multiple circuit outputs are generated, each representing the output at a specific quantized time step. This allows to analyze transition propagation with pure functional zero-delay simulations, gaining a speed-up of two to four orders in magnitude. A drawback of this method is the rising expansion complexity with finer time quantization and non-unit or asymmetric gate propagation delays.

Other related work in the field of fault injection achieve simulation speed-up by exploiting the transient nature of radiation-induced events. In [7], this is performed by making probabilistic assumptions on register setup time violations and omitting the timing simulation for propagation events without fault effects. In [8], a timing simulation is applied only in fault injection intervals and a faster parallel functional RTL simulation is used in other intervals. The

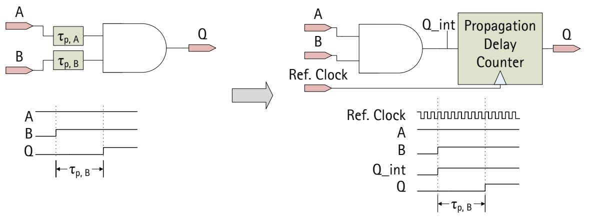


Figure 1: Approach used to model the propagation delay of a standard gate using a counter. The number of reference clock cycles until `Q_int` is propagated to `Q` depends on the chosen time quantization and the specific delay parameters for a gate.

authors of [14] use software-based timing analysis to obtain fault latching probabilities in a gate-level circuit and study the effects of probable faults by using a fast accompanying FPGA emulation and a runtime scan chain-based injection mechanism. The described procedures are beneficial for analyzing localized fault injections, but are not applicable to stochastic operation conditions, where errors are continuously induced by circuit timing violations. Thus, in the research field of *Stochastic Computing*, a complete circuit timing analysis has to be performed.

Since the availability of high-density FPGA devices, the emulation of the circuit timing behavior has become feasible to utilize speed advantages associated with hardware implementations. The authors of [9] and [10] propose timing instrumentation models for standard cell gates by using shift registers or non-linear counters for delaying gate input transitions and emulating propagation behavior. Both approaches correlate an emulation clock cycle to a *time quantum*. Specific propagation delays for each gate are defined by the shift register length or by counter values. In [11], these gate-level timing approaches are combined with RTL co-emulation in the AMUSE soft-error evaluation framework, achieving a speed-up of up to three orders of magnitude compared to simulation-based fault injection campaigns. A different timing emulation approach in the field of fault injection is presented in [12], where shift registers and *timing quantization* are applied to model fault duration patterns within a single circuit clock cycle. Also an abstract histogram-based fault model simulator is implemented for FPGA devices by using *timing quantization* and a clock cycle counter [13].

Emulation-based approaches that embed fixed timing information in the netlist instrumentation during FPGA synthesis are disadvantageous for stochastic timing analysis. Changes to the circuit timing require time-consuming re-synthesis of the design, which is narrowing the usability in variable timing cases, e.g., due to different voltage and temperature environments. For example, the chip design presented in [15] contains 16 different precise and approximate adder architectures for the purpose of exploring the stochastic behavior of arithmetic circuits within a wide range of ambient temperature and supply voltage conditions. Fabricated in a 1 µm high-temperature SOI CMOS technology, the operation range includes temperatures from

Table 1: Comparison of Circuit Timing Implementation Approaches

Year	Ref.	Modeling Level	Target Circuit	Timing Representation	SIM	EMU	Configuration Stage
2003	[6]	Gate propagation delay	Gate netlist	Circuit topology expansion	✓		-
2004	[7]	Setup time violations	Gate netlist	Probabilistic assumptions	✓		-
2007	[8]	Gate propagation delay	Gate netlist & RTL	Gate/RTL co-simulation	✓		-
2007	[9]	Gate propagation delay	Gate netlist	Shift registers		✓	FPGA synthesis
2009	[10]	Analog output voltage	Gate netlist	Non-linear counters		✓	FPGA synthesis
2012	[11]	cf. [9, 10]	Gate netlist & RTL	cf. [9, 10], Co-EMU		✓	FPGA synthesis
2014	[12]	Fault duration	Gate netlist & RTL	Shift registers		✓	Runtime scan chain
2015	[13]	Fault duration	Circuit model	Time-Step Counter		✓	Runtime
2015	[14]	Gate propagation delay	Gate netlist	Probabilistic assumptions	✓	✓	Runtime scan chain
2017	this, [5]	Gate propagation delay	Gate netlist	Counters / decrementers		✓	Runtime scan chain

25 to 250 °C and supply voltages from 1.8 to 3.6 V. A fine-grained design space exploration of the stochastic regimes could require the analysis of about 50 different temperatures and 20 supply voltages, resulting in 1000 timing corner cases. Therefore, when using fixed-timing emulation-based approaches, the generation of 1000 FPGA bitstreams for a single circuit design would be necessary.

To cope with FPGA-based stochastic timing analysis, a runtime-configurable timing emulation methodology, suitable for executing timing analysis campaigns under varying stochastic conditions with a single instrumented gate netlist design, is presented in [5]. The configuration mechanism is a scan chain-based approach, which extends a previously published *FauLt INjection Tool* (FLINT) [16] framework.

In this paper, an additional register-to-register path analysis of the gate-level netlist is included into the framework. Since the useful stochastic operation regime is limited by the error tolerance of a certain application, only a fraction of all possible timing paths will be ever violated. The presented path analysis allows to omit logic gates from instrumentation that will not cause any timing violation. Thus, the instrumentation overhead can be reduced. Because a full path analysis of a gate-level design is too complex and computationally impractical, a critical path selection and path pruning algorithm is implemented to consider only relevant paths for instrumentation. Similar approaches are applied in the field of process variation analysis to identify testable long paths in a circuit [17] and to prune paths that are insignificant and redundant for delay fault testing [18].

3. The Runtime-Configurable Timing Emulation Framework

The proposed emulation framework utilizes four basic concepts, which are time quantization, instrumented standard cells, iterative critical path selection and timing parameter configurability. Detailed implementation aspects are covered in this section. Additionally, information is provided on the used FPGA communication framework and the software toolchain to perform gate-level netlist instrumentation.

```
(CELLTYPE "XOR2")
  (INSTANCE mul_27_31/g13201)
  (DELAY
    (ABSOLUTE
      (PORT A (::1.2))
      (PORT B (::1.2))
      (IOPATH (posedge A) Q (::2629) (::2932))
      (IOPATH (negedge A) Q (::3004) (::3142))
      (IOPATH (posedge B) Q (::2392) (::2781))
      (IOPATH (negedge B) Q (::3132) (::3078)) ))
```

Listing 1: Excerpt from a SDF file defining eight propagation delay parameters on a picosecond scale for a 2-input XOR gate. The IOPATH statement defines rising and falling edge output propagation delays for each possible gate input transition. PORT values are model constructs to account for additional input interconnection delays.

For time quantization, a decrementing counter-based approach is applied and injected for each instrumented standard cell of the gate-level netlist under analysis. Besides this gate instrumentation model, all instrumented cells identified by the path selection process are interconnected by a configuration chain for timing parameters. This offers the runtime configurability required for analyzing variable timing corner cases.

3.1. Time Quantization

Fig. 1 illustrates a common approach to model gate propagation delays. Each gate input is annotated with specific time weights τ_p, by which logic transitions at the gate output `Q` are delayed when a transition at the corresponding input occurs. For an ASIC netlist, these values are determined during synthesis and stored in a *Standard Delay Format* (SDF) file [19] with a maximum of four theoretically independent propagation parameters per input for each combination of input and output rising and falling transitions. For an n-input combinational gate, the number of delay parameters therefore scales with $4n$ as exemplarily shown in Listing 1.

In the proposed implementation, gate propagation delays are emulated in hardware by using decrementing delay counters (Fig. 1, right-hand side). When a specific *time quantum* (i.e., *timing resolution*) is assigned to the reference clock period, the number of elapsed clock cycles for a transition from `Q_int` to the instrumented output `Q` directly corresponds to a propagation delay. Emulation speed is

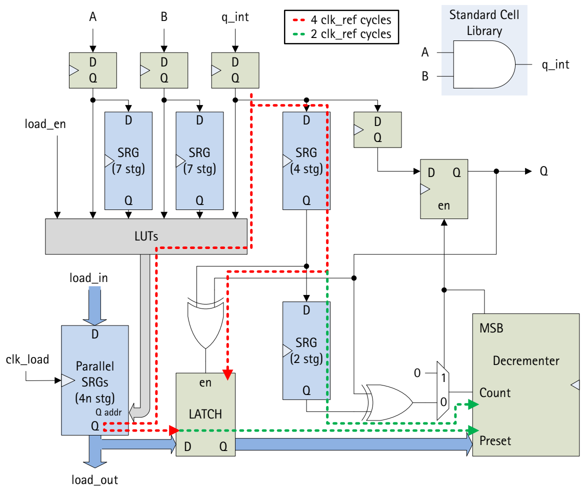


Figure 2: Schematic of instrumentation for combinational standard cells. Unless otherwise specified, the elements are clocked with the quantization reference clock **clk_ref**. For loading timing parameters, a separate clock **clk_load** is used. Red and green dashed lines denote paths that have a latency of four and two reference clock cycles, respectively.

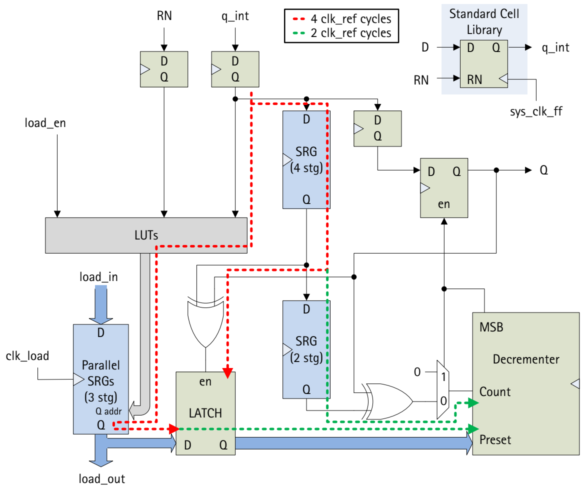


Figure 3: Schematic of instrumentation for edge-triggered sequential standard cells. Unless otherwise specified, the elements are clocked with the quantization reference clock **clk_ref**. For loading timing parameters, a separate clock **clk_load** is used. Red and green dashed lines denote paths that have a latency of four and two reference clock cycles, respectively.

then dependent on two factors, reference clock frequency and time quantization resolution.

3.2. Instrumented Standard Cells

For an ASIC standard cell library, timing instrumentation models for combinational and sequential gates need to be supplied. The schematic of a FLINT+-instrumented combinational standard cell is shown in Fig. 2. This cell consists of the original logic gate from the cell library, the decrementing delay counter, an output transition detection circuit to trigger the decrementing process, a look-up table-based address generator to select the correct propagation delay parameter, and a shift register-based storage element to keep the $4n$ programmable timing parameter values.

To understand the instrumentation functionality, an output transition at the original gate output `q_int` shall be considered. This output, combined with the current and previous cell inputs delayed by shift registers, are used to identify pre- and post-transition cell state and to select the correct propagation parameter by a look-up table-based address generator. The addressed parameter value, coming from the programmed shift register-based storage element, is latched onto the preset input of the decrementer. If the original gate output `q_int` and the instrumented output `Q` are not equivalent, an XOR gate triggers the counting process. When the decrementer output wraps around, the most significant bit is set, deactivating the counter and enabling the instrumented cell output `Q`, performing the delayed transition after the desired propagation interval. `Q` is connected to the inputs of other logic gates within the netlist, so the output transition itself triggers the propagation process in the next instrumented cells.

Since the number of address generator inputs and addressable delay parameters increases with the gate input number n, the required look-up table can become large and easily exceeds the size of 6-input LUTs on Xilinx Virtex-6 FPGA devices used for this framework evaluation [20]. In these cases, the table is mapped into several LUTs, introducing additional logic stages and routing, which lengthen the time until a stable delay parameter value is available. To avoid a reference clock frequency reduction and less emulation speed, the parameter latch enable is delayed by 4 clock cycles and the count trigger by 2 more cycles using additional shift registers. This allows the complete delay parameter selection process from the cell inputs to the decrementer to take place in 6 clock cycles instead of a single one, which greatly relaxes the path timing constraints for FPGA synthesis. Up to a depth of 32 stages, every shift register of an instrumented cell can be implemented efficiently within a single LUT on Virtex-6 FPGA devices [20], which minimizes the resource overhead. This implementation allows to keep the reference clock frequency high, but introduces a minimum propagation latency from the cell inputs to the instrumented output. However, as long as all quantized gate delays for a given timing resolution are large enough to exceed this amount of latency, the additional cycles can be compensated in the configurable delay parameter values.

In contrast to combinational cells, edge-triggered sequential flip-flop cells can be instrumented with a simplified address generator, as depicted in Fig. 3. Here, it is only necessary to evaluate the current state of `q_int`, as a transition occurs on system clock edges (`sys_clk_ff`, the quantized original circuit clock) exclusively and the previous state is implicitly known. Exceptions are asynchronous set or reset inputs like `RN`, which are also evaluated to unambiguously identify asynchronous transition trigger events. In total, only 3 delay parameters need to be configured for sequential cells, which are the low-to-high and the high-to-low

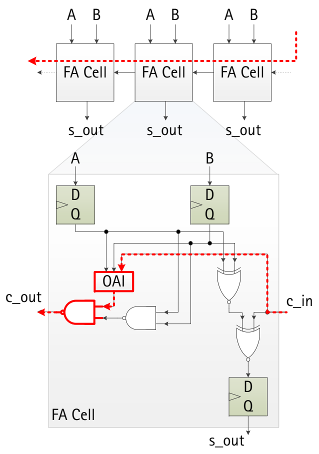


Figure 4: Example of critical path selection in a ripple-carry adder. The critical path is depicted as a dashed red line. If the critical path delay is violated, the gates marked in red have to be instrumented to emulate stochastic timing errors.

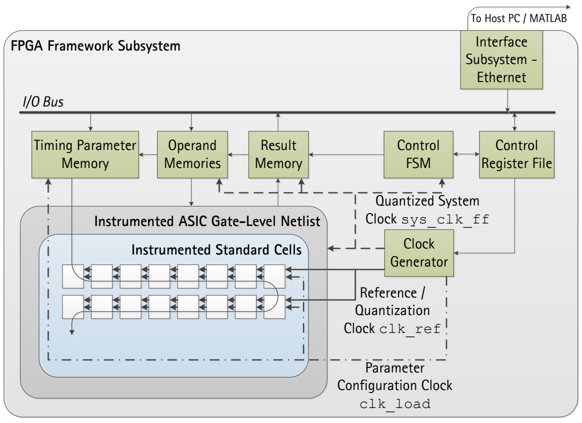
propagation delay of `Q` after a rising clock edge as well as one propagation delay parameter for an asynchronous set or reset.

Meta-stability of instrumented flip-flop outputs due to setup or hold time violations is intentionally ignored in the instrumentation model. However, this is no limitation for the evaluation of arithmetic results within the stochastic operation regime. When using standard gate-level timing simulations, the analysis of output results also requires the deactivation of setup and hold timing checks, otherwise undefined *'X'* states are induced for timing-violated signals which cannot be compared to arithmetic reference results. Therefore, the proposed instrumentation model implements the operation- and bit-true reference simulation behavior used in this work and does not require additional circuitry to handle meta-stability effects.

3.3. Critical Path Selection and Instrumented ASIC Gate-Level Netlist Generation

From a gate-level netlist perspective, three tasks have to be performed for instrumentation: selecting gates to be instrumented, replacing original gates by the instrumentation models and establishing a parameter configuration chain between all instrumented cells.

The selection of gates depends on to what extent the stochastic operation regime shall be exploited, i.e., which register-to-register timing paths are expected to be violated. Since the reasonable range is influenced by application properties and the architecture of the analyzed hardware itself, a minimum clock period within the stochastic regime is configured for the toolflow to define an extreme operation case. If the delay of a path is longer than the configured clock period, a path violation can occur and all gates forming this path need to be instrumented to obtain the precise timing error behavior. Gates that exclusively form shorter paths do not contribute to the actual stochastic error output of a circuit and therefore are omitted from instrumentation,



To Host PC / MATLAB
FPGA Framework Subsystem
Interface Subsystem - Ethernet
I/O Bus
Timing Parameter Memory
Operand Memories
Result Memory
Control FSM
Control Register File
Quantized System Clock sys_clk_ff
Instrumented ASIC Gate-Level Netlist
Instrumented Standard Cells
Clock Generator
Reference / Quantization Clock clk_ref
Parameter Configuration Clock clk_load

Figure 5: FPGA framework used for FLINT+ netlist instrumentation.

retaining the original functional gate description and saving instrumentation model overhead. As an example, consider Fig. 4. Gates marked in red form the overall critical path and require timing instrumentation if the circuit timing is violated. Any other gate does not need to emulate timing behavior unless the longest path that traverses the gate is also violated. The selection of critical paths for instrumentation is achieved during ASIC netlist synthesis. An iterative algorithm is used to find the longest path through each gate. The algorithm as well as the instrumentation software toolchain is described in detail in Section 3.5.

All standard cells in the netlist selected from the previous stochastic regime path analysis are replaced by the corresponding timing instrumentation model cell from Section 3.2. To load the shift register-based storage element with the desired timing information, a scan chain-like parameter configuration approach is implemented. Via programming ports `load_in` and `load_out` (see Fig. 2), a chain is formed by all instrumented cells to shift delay parameter values to the corresponding gate. Since the chain is pervaded throughout the whole standard cell design, parameter shifting is performed on a separate clock `clk_load` significantly slower than the quantization reference clock to account for large routing delays. Once a set of parameters is configured, exhaustive analyses for this timing case can be performed without re-configuration, mitigating the slow load clock in total.

3.4. FPGA Emulation Framework

For performing emulation-based timing analysis campaigns, the instrumented ASIC netlist is inserted into a *Unified EMUlation Framework* (UEMU) [21] depicted in Fig. 5. It contains the instrumented ASIC netlist itself, memories, control logic and control registers. A clock generator generates the reference clock `clk_ref`, the quantized system clock `sys_clk_ff` for sequential cells and the parameter configuration clock `clk_load`. An Ethernet link between the FPGA platform and a host PC is established to transfer timing parameters, test operands and results. Interface functions for MATLAB are available to easily com-

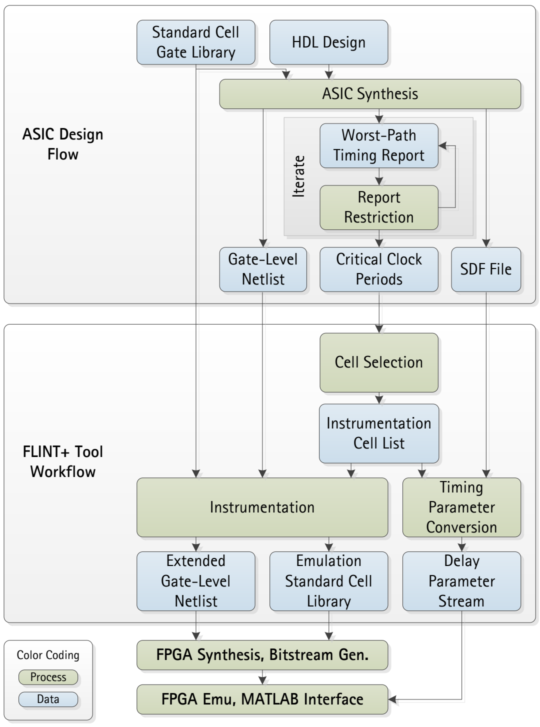


Figure 6: Tool workflow for FLINT+ timing instrumentation.

bine arithmetic test pattern generation, transfer to/from the emulation system and execution control as well as data evaluation.

After an initial transfer of timing configuration parameters and operand data, the execution is performed autonomously within the FPGA emulation framework. An integrated control FSM takes care of timing parameter configuration, applying operands to the netlist under test and storing result data. When the configured operations are done, a status flag is set, indicating that result data can be transferred back to the host. This procedure minimizes time-consuming transfer overhead between FPGA platform and the host PC.

3.5. Instrumentation Software Toolchain

Prior to FPGA synthesis, the instrumented ASIC netlist and standard cell library as well as timing parameters have to be generated. For this process, Python script-based instrumentation tools from the FLINT fault injection methodology presented in [16] are used in an extended version. The original ASIC gate-level netlist and the corresponding SDF timing information are generated from a typical ASIC synthesis design flow. An extension to the standard ASIC flow is used for instrumentation analysis.

To identify gates to be instrumented for a given stochastic operation regime, the longest timing path traversing each particular gate in the netlist has to be found, which was already pointed out in Section 3.3. However, the generation of an ordinary timing report during ASIC synthesis is not feasible for this analysis task. Since static timing

```
# list of all cells to be possibly instrumented
set selected_cells [get_cells -hierarchical {*}]
set already_instrumented [list]

while 1 {
# find critical path through at least one sel. cell
  report timing -from [all_registers] \
    -through $selected_cells -to [all_registers] \
    -full_pin_names > TEMPFILE

# open generated timing report and read in lines
  set filepointer [open "TEMPFILE" "r"]
  set filedata [read $filepointer]
  set filedata [split $filedata "\n"]
  close $filepointer

  set to_instrument [list]
  foreach fileline $filedata {

# if report empty, clear selection and abort
    if { -- no paths in report -- } {
      set selected_cells [list]
      break
    }

# if a register cell is found, isolate cellname
# and remove it from cell selection
# to allow other register endpoints to be found
    if { -- clock input found -- } {
      set cellname -- instance name from report --
      set selected_cells [remove_from_collection \
        $selected_cells $cellname]
      set arrivaltime -- time from report --
    }
# if output of a gate is in path, instrument gate
# and remove it from cell selection
    if { -- gate output found -- } {
      set cellname -- instance name from report --
# if gate already instrumented, ignore it
      if {[lsearch -exact $already_instrumented \
        $cellname] == -1} {
        lappend to_instrument $cellname
        lappend already_instrumented $cellname
        set selected_cells [remove_from_collection \
          $selected_cells $cellname]
      }
      set arrivaltime -- time from report --
    }
  }

# write out newly instrumented gates for
# the current worst path period
  foreach cellname $to_instrument {
    puts "$arrivaltime\t$cellname"
  }

# if all cells are analyzed, abort
  if {[llength $selected_cells] == 0} break
}
```

Listing 2: Tcl script for Cadence RTL Compiler (simplified) of the iterative critical path selection process after ASIC synthesis.

analysis is endpoint-oriented, the report can be configured to cover the worst paths to each flip-flop or primary output in the design. The coverage of all other logic gates in the netlist is uncertain unless all existing paths are printed out. Thus, the definition of a longest path and consequently, a critical instrumentation clock period for every gate within the netlist with a single timing report is extremely inefficient and impractical in terms of storage memory and computation time, even for netlists of moderate size.

Therefore, instead of generating a single timing report for the complete design, multiple temporary timing reports with systematic iterative cell instance restrictions are generated during ASIC synthesis. A simplified Tcl implementation of the algorithm for the Cadence RTL Compiler is shown in Listing 2. Initially, the path analysis is allowed to traverse through any gate in the netlist, resulting in the overall critical path of the design. The gate instances forming the critical path are extracted by a timing report parser and annotated with the path delay. A violation of the annotated worst-case timing denotes the starting point for stochastic errors that may be produced by these gates. Then, the newly annotated gates are removed from the list of traversed gates that the path analysis should consider. Since at least one gate from the list has to be covered, the next timing report will not output the overall critical path, but the longest path through the remaining restricted netlist fraction. By iteratively generating new timing reports, annotating previously uncovered gate instances and further restricting the path analysis, it can be assured that exactly one longest path and one critical clock period for timing violations are reported for each gate instance in the netlist. This minimizes the necessary memory requirements and computation time for the longest path selection.

As depicted in Fig. 6, the remaining FLINT+ tool workflow is divided into three major parts, i.e., selecting cells for instrumentation, cell instrumentation and timing parameter conversion. The first step contains the analysis of the critical clock periods for each gate instance. Parameterized by user input, all standard gates that require instrumentation are identified for a certain stochastic regime, given by the minimum system clock period.

In the second step, a functional logic description of the standard cell library is extended with the instrumentation models from Figs. 2 and 3. Then, in the ASIC netlist, each standard cell to be instrumented is replaced by the modified cell and the timing parameter configuration chain is connected. In the current tool, the appearance order of cells in the netlist also determines the chain connection order. This is not an optimum solution, because it does not ensure that both netlist signal path and the configuration chain are routed efficiently. However, it is planned to refine this step by a circuit topology-aware connection algorithm, which eases FPGA routing by ordering the configuration chain in parallel to the netlist signal path whenever possible.

In the third step, propagation delay values for all instrumented gates are taken from the SDF file and converted to counter preset values for a given timing resolution. This parameter stream is transferred to the emulation system prior to execution.

4. Performance Evaluation

In order to analyze the dependencies between *time quantum* resolution, timing emulation accuracy, speed-up to simulations and FPGA resources overhead due to instrumentation, arithmetic units are used as evaluation samples. Structural HDL descriptions of a ripple-carry adder, ripple-carry array multiplier and a non-restoring array divider are synthesized to ASIC gate-level netlists. To obtain a set of different design sizes, the units are synthesized for operand bitwidths between 8 and 896 bits. The standard cell library used for synthesis belongs to a 1 µm high-temperature SOI CMOS technology provided by the Fraunhofer Institute for Microelectronic Circuits and Systems [22]. Worst-case propagation delay information is generated for a typical environmental corner case of 175 °C chip temperature and 3.3 V supply voltage. The netlists are evaluated with a fixed set of 10^5 random-generated arithmetic operations. A ML605 development board with a Xilinx Virtex-6 XC6VLX240T-1 FPGA device is used as the target platform for synthesis and emulation [23]. All SDF timing simulations for speed-up comparisons are performed using the ModelSim HDL simulator [24] with deactivated timing checks (*+notimingchecks* parameter) to suppress undefined arithmetic results on a compute server equipped with an Intel Xeon E5-2683 CPU (2.1 GHz).

4.1. Analysis of Error Occurrences

At first, the number of *error occurrences* in a set of arithmetic results shall be considered when timing paths are violated. The top row of Fig. 7 depicts this metric for the profiled 16-bit adder with a critical path length of 102 ns, for the 16-bit multiplier with 355 ns and for the 16-bit divider with 2300 ns, respectively. The number of *error occurrences* rises for all units when more timing paths are violated at shorter periods of the quantized system clock `sys_clk_ff`.

When the *time quantum* is increased from 1 ps to larger values, deviations appear due to delay parameter rounding errors. For a resolution of 500 ps, the number of *error occurrences* is heavily overestimated. This is due to the minimum propagation latency induced by the gate instrumentation circuit, described in Section 3.2, which is 9 reference clock cycles for the current implementation. Delays smaller than 4.5 ns are then distorted for 500 ps *timing resolution.* Thus, 1, 10 and 100 ps are used as representative time quantizations for the following evaluation. Larger *time quanta* invoke a higher speed-up at the expense of reduced timing behavior accuracy, since a larger time interval is processed per reference clock cycle.

When lower precision time quantization is used, rounding effects of the delay parameters have an impact on the timing accuracy. These effects are not unique to the presented methodology, time interval rounding also occurs in

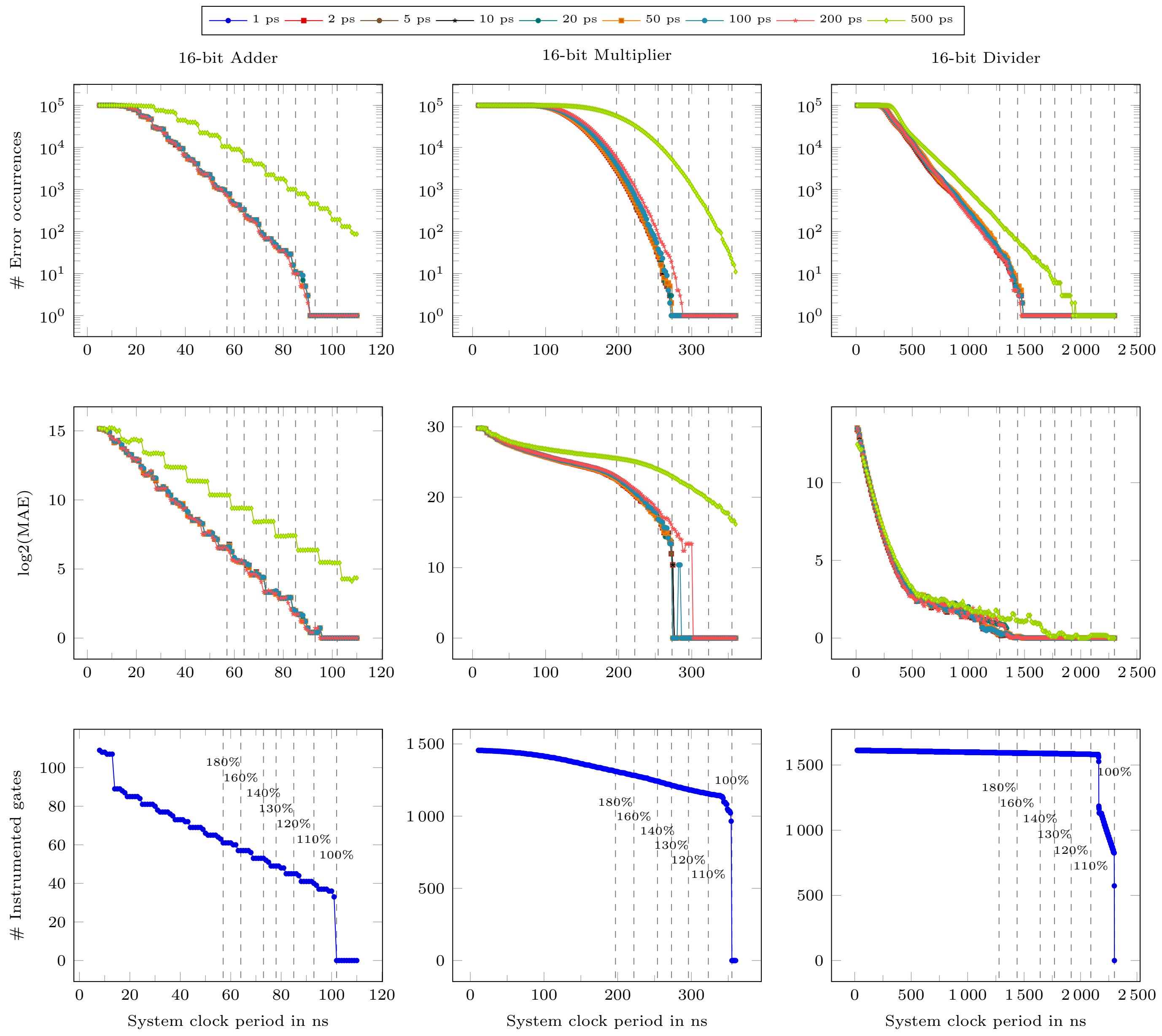


Figure 7: Top row: Influence of *time quantum* resolution on the number of error occurrences for a 16-bit adder (left subfigure), 16-bit multiplier (center subfigure) and 16-bit divider (right subfigure). 10^5 operations are performed for system clock periods from 8 to 110 ns in steps of 1 ns for the adder, from 8 to 360 ns in steps of 2 ns for the multiplier and from 10 to 2300 ns in steps of 10 ns for the divider, respectively. Center row: Influence of *time quantum* resolution on the mean absolute error magnitude of 10^5 operations for a 16-bit adder (left subfigure), 16-bit multiplier (center subfigure) and 16-bit divider (right subfigure). Bottom row: Number of required instrumented gates as a function of the system clock period for a 16-bit adder (left subfigure), 16-bit multiplier (center subfigure) and 16-bit divider (right subfigure). The gray dashed lines denote system clock periods that correspond to the critical path (100% *critical operation frequency*) and to frequency overscaling conditions of 110% to 180% *critical operation frequency*.

timing simulations during switching event generation when the provided SDF parameters have a higher resolution than the simulator timescale. But in contrast to event-driven simulators, where the user normally has no influence on propagation event generation, the instrumentation software toolchain of the proposed framework supports three ways to calculate delay parameters from SDF values: rounding to the nearest quantized time interval, flooring the parameter value or ceiling it. To present a more detailed view on the circuit timing emulation accuracy, a differential comparison to reference timing simulations with a delay parameter resolution of 1 ps shall be considered. The influence of the rounding method on the number of *error occurrences* is depicted in Fig. 8 as a normalized difference to SDF simulation. For *time quanta* smaller than 10 ps, the difference is below 0.8 % when rounding to the nearest time interval.

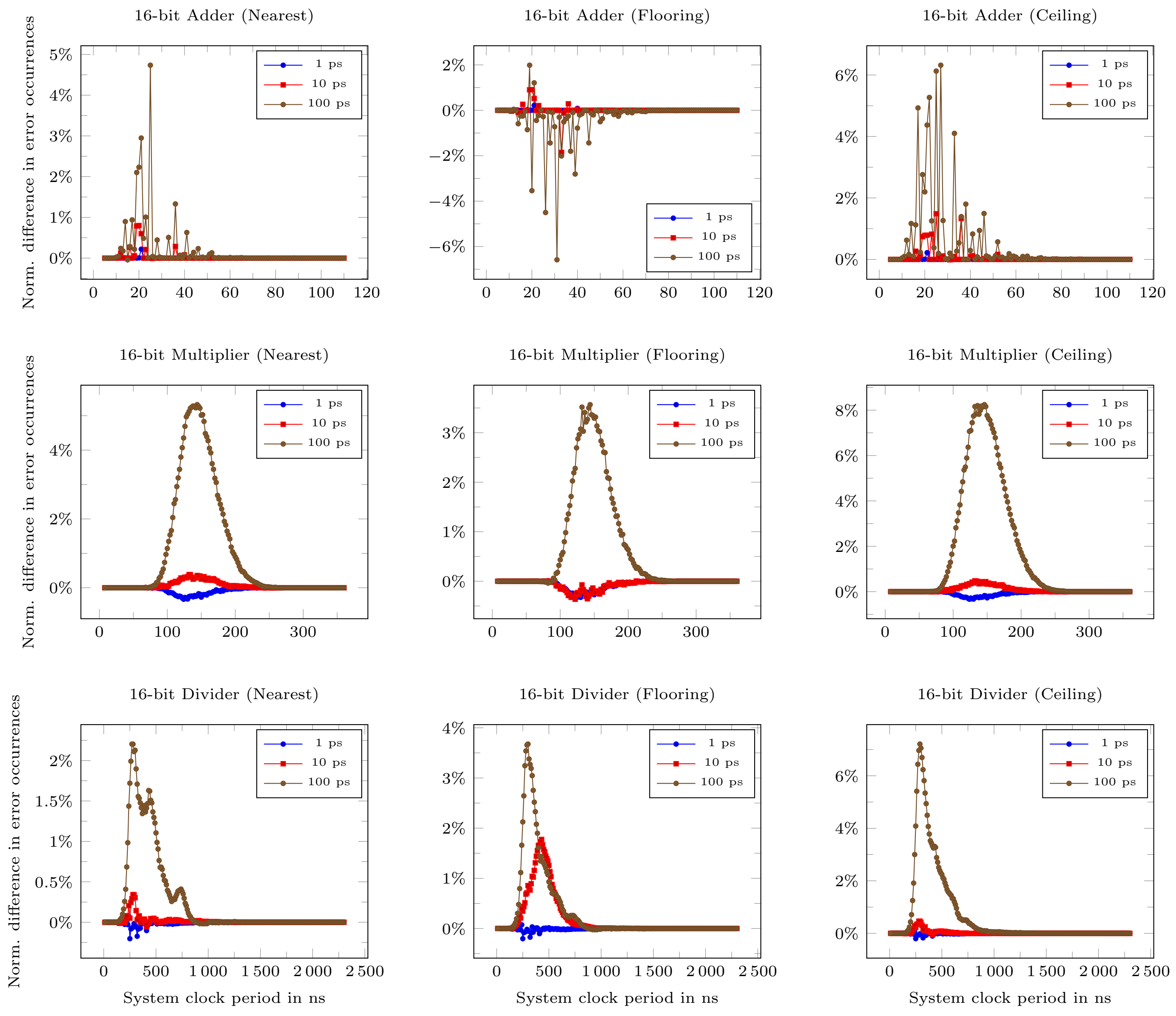


Figure 8: Difference in number of *error occurrences* to SDF simulation for a 16-bit adder (top row), 16-bit multiplier (center row) and 16-bit divider (bottom row) when rounding timing parameters for lower precision *time quanta*. From left to right: Rounding to nearest quantized step, flooring, and ceiling. 100 normalized percent correspond to 10^5 operations performed for each system clock period.

4.2. Analysis of Error Magnitude

For arithmetic units like adders, multipliers and dividers, the *error magnitude*, i.e., the difference between the arithmetic results of timing emulation and timing simulation is another metric to evaluate under stochastic conditions. The center row of Fig. 7 shows the absolute logarithmic mean error magnitude for each unit and system clock period. Fig. 9 shows the normalized difference to reference simulations. For all units and *time quanta* smaller than 10 ps, the difference is below 0.3% when rounding to the nearest time interval and grows with coarser quantization. The choice of parameter rounding could be used to force a specific analysis behavior, like systematic over- or underestimation of error metrics, but this aspect is out of scope of this paper and nearest time interval rounding will be assumed in the following evaluation, which gave the overall smallest differences to the simulation reference.

Finally, instrumentation-inherent sources for deviations to SDF simulation results should be mentioned. The address generator used for selecting the propagation delay is working with a fixed prioritization scheme. When multiple input transitions occur at the same time, the parameter of the higher-priority input is selected, independent from the actual delay values. Furthermore, if the counter is active, it is not restarted by following input transitions, even if a faster propagation would be introduced. These implications prevent the timing emulation from being absolutely equivalent to the simulation reference. In favor of a smaller

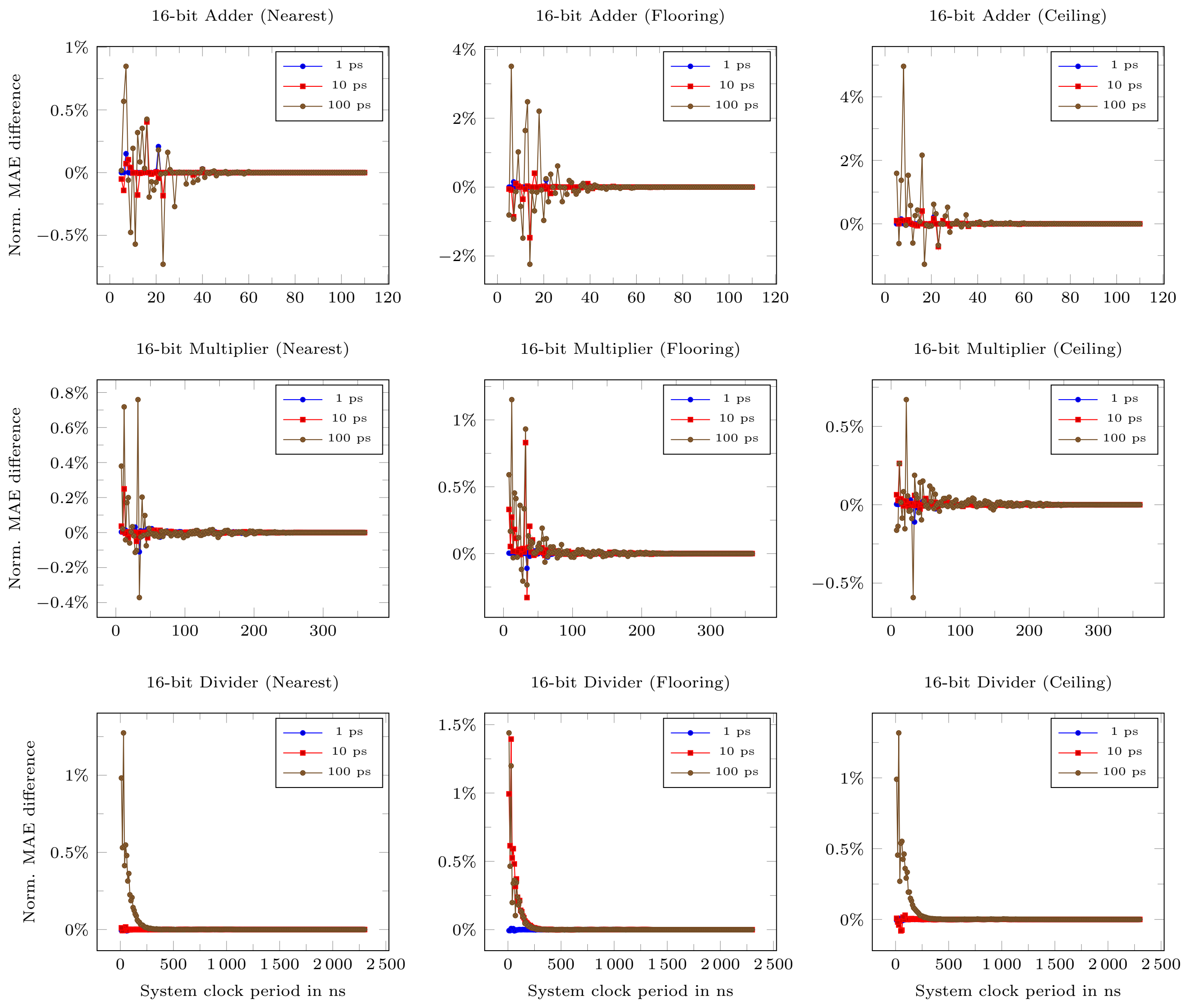


Figure 9: Mean emulation *error magnitude* difference to SDF simulation for a 16-bit adder (top row), 16-bit multiplier (center row) and 16-bit divider (bottom row) when rounding timing parameters for lower precision *time quanta*. From left to right: Rounding to nearest quantized step, flooring, and ceiling. 100 normalized percent correspond to the maximum addition output value of $(2^{17} - 2)$, to the maximum multiplication output value of $(2^{16} - 1)^2$ for the multiplier or to the maximum division output value of $(2^{16} - 1)$ for the divider.

FPGA resource overhead, the additional treatment of these cases was relinquished.

4.3. Analysis of Critical Path

The bottom row of Fig. 7 shows the required amount of cell instrumentation for analysis of a certain stochastic regime, given by the minimum system clock period applied to the designs. Since only timing-violated paths may contribute to the occurrence of stochastic errors, all gates that exclusively form shorter paths can be omitted from the instrumentation. The instrumentation characteristic depends on the netlist architecture and the distribution of path lengths in the design.

When the operation frequency of an error-free circuit is progressively increased, the system will eventually experience timing violations of one or more timing paths at the *critical clock frequency*. This frequency denotes the entry point of the stochastic regime. Any frequency overscaling (FOS) with respect to the *critical operation frequency* increases the number of timing path violations and the amount of logic gates affected by violated paths. In Table 2, the percentage of instrumented gates for stochastic FOS of up to 180% of the *critical operation frequency* is denoted for the profiled 16-bit adder, multiplier and divider gate-level netlists. A FOS value of 100% is defined as a timing violation of the critical path only. For any system, the total number of instrumented gates at the *critical operation frequency* gives a hint about the length of the critical

Table 2: Required Percentage of Netlist Instrumentation for Maximum Stochastic Frequency Overscaling Conditions

Design	Max. Percentage of FOS						
	100	110	120	130	140	160	180
ADD 16	26%	32%	36%	39%	42%	45%	48%
MUL 16	69%	78%	80%	82%	84%	86%	88%
DIV 16	50%	96%	96%	96%	96%	97%	97%

path within that particular circuit.

For the ripple-carry arithmetic units in this analysis, the overall critical path and therefore the base fraction of instrumented gates is determined by the longest carry propagation chain through all basic building blocks of the design. In the analyzed ripple-carry adder design, the initial instrumentation requirement is 26% of the total netlist. The instrumentation effort increases linearly with increasing FOS as sum bit outputs and operand inputs have to be instrumented. For a maximum of 180% FOS, only 48% of the netlist are instrumented, saving FPGA resources compared to a full instrumentation.

In the ripple-carry array multiplier basic cell, about 69% initial instrumentation effort are required for the critical path, which is significantly more than for the adder design. For the non-restoring divider array, an initial instrumentation estimation of 50% is obtained, but the instrumentation amount increases quickly and reaches 96% of the netlist for only 110% FOS. Therefore, it is concluded that the FPGA resource saving capability of path-selected instrumentation is heavily influenced by the netlist design and more efficient for simple one-dimensional carry propagation in adders than for two-dimensional multiplier and divider arrays.

4.4. Emulation vs. Simulation Speed-Up

In the top row of Fig. 10, the required time for simulating and emulating 10^5 arithmetic operations is depicted as a function of the design size in standard cells. Additionally, the influence of timing resolution reduction is evaluated by increasing the *time quantum* from 1 to 10 and 100 ps for emulation and switching the minimum time step to 10 and 100 ps for SDF-supported simulation.

For a specific *timing resolution*, the elapsed time increases with the design size for SDF simulations. The FLINT+ emulation speed only depends on the reference clock and the *time quantum* associated with it. Simultaneous transitions occur in parallel as the whole circuit is instrumented and mapped onto the FPGA, so the design size does not directly affect the emulation duration. An indirect influence is observable as the maximum achievable reference clock frequency is decreased with larger designs due to FPGA net routing delays. The initial delay parameter configuration effort is also netlist-dependent because of a variable amount of delay parameters. But with a maximum configuration time below 10 ms for the profiled designs, it is negligible compared to the operation execution duration of up to 500 seconds. In contrast, event-based logic simulators process transition events sequentially, so more transitions in larger netlists require more processing time.

When the *time quantum* is increased by a factor of 10, the emulation time is reduced by a factor of 10, since a larger time interval is processed per reference clock cycle. For SDF simulations, increasing the minimum time step has less impact, because the number of occurring logic transition events is not reduced. This gives rise to a large emulation speed-up when high timing accuracies are not necessary, e.g., for the 20-bit multiplier, an acceleration by a factor of 5 for 1 ps, 57 for 10 ps and 519 for 100 ps resolution is achieved. With larger *time quanta* and speed-up, the emulation accuracy decreases, as it is annotated in the bottom row of Fig. 10. For smaller designs, the speed-up advantage is reduced, emphasizing the emulation framework benefits for more complex designs.

It must be pointed out that the maximum frequency for the emulation reference clock is mainly target architecture-dependent, i.e., netlist-dependent due to FPGA net routing delays. Table 3 shows the maximum reference clock frequency for the evaluated designs, ranging from 500 MHz for the smallest adder designs to 125 MHz for the 22-bit multiplier, which therefore shows smaller speed-up factors than the 20-bit multiplier.

In general, a higher reference clock frequency and a higher emulation speed-up can be achieved by reducing the instrumentation overhead and simplifying the place-and-route process. Besides coarser timing quantization and a smaller delay parameter bitwidth, critical path selection is employed to save FPGA resources. For example, if the maximum stochastic operation range is reduced from full instrumentation to 110% FOS, the reference clock frequency of the 22-bit multiplier with 10 ps *quantization resolution* is increased from 278 to 400 MHz. This results in an additional speed-up of 1.4, depicted as split line branches in Fig. 10. When a limited *timing resolution* of 100 ps with a small stochastic analysis regime of 110% FOS is sufficient, very large speed-up factors of up to 1400 for the 896-bit adder are possible.

4.5. Resource Overhead

FPGA resource counts are collected and depicted in Fig. 11. These numbers represent the LUT and register utilization for the instrumented netlists on the Virtex-6 FPGA device only and do not contain overhead introduced by the UEMU framework. Every instrumented gate in the original design netlist is replaced by a larger cell with injected timing circuitry. Thus, resource overhead is generated compared to the synthesis results for the original uninstrumented netlist.

As it can be deduced from Fig. 11, the resource requirements for registers and LUTs increase linearly with the amount of instrumented standard cells. A resource overhead factor of approximately 30 to 40 is introduced when compared to the number of standard cells at 1 ps timing parameter resolution. When sacrificing timing emulation accuracy by choosing less *timing resolution*, the

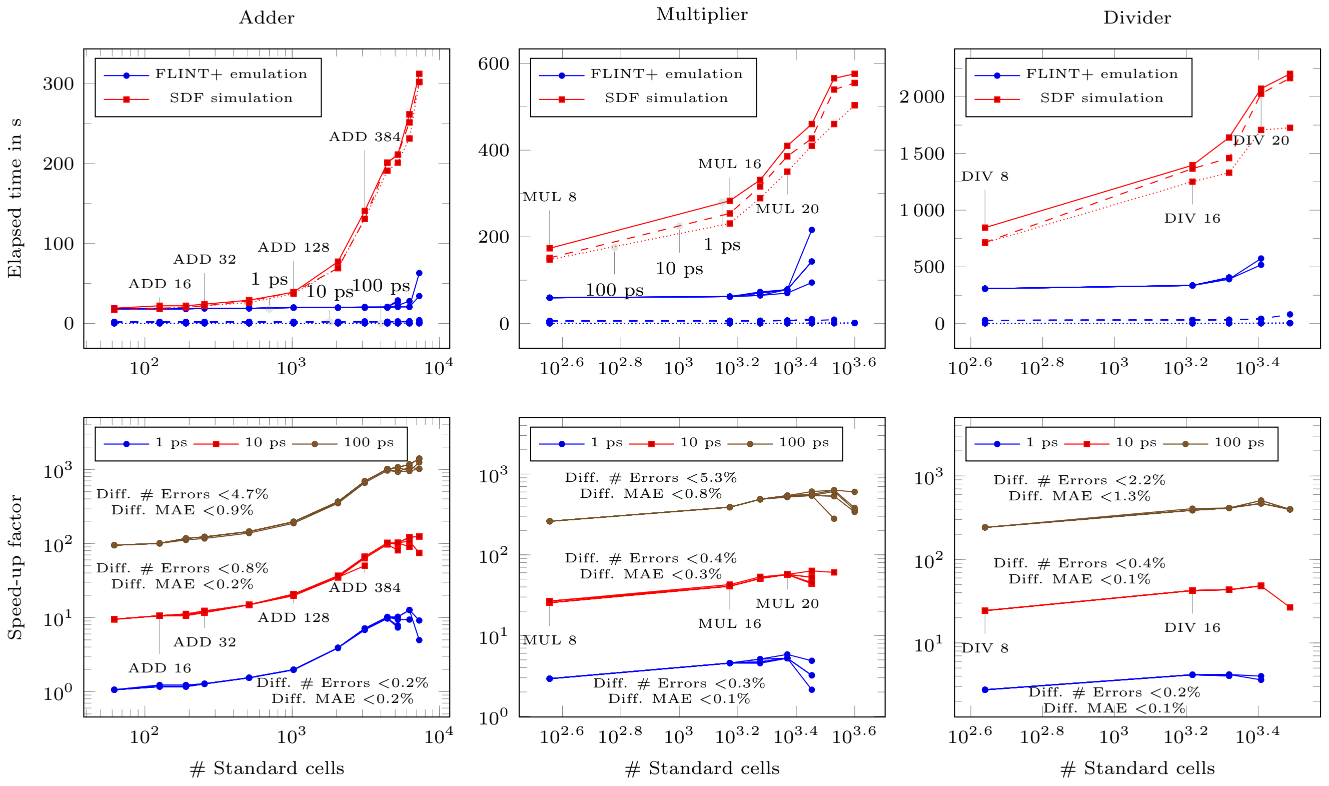


Figure 10: Comparison of elapsed time (top row) and speed-up factor (bottom row) for instrumented netlist emulation and SDF timing simulation. The evaluated adder bitwidths are 8, 16, 24, 32, 64, 128, 256, 384, 512, 640, 768 and 896 bits. Multipliers are evaluated for bitwidths of 8, 16, 18, 20, 22 and 24 bits. Dividers are evaluated for bitwidths of 8, 16, 18, 20 and 22 bits. All values are generated for 10^5 operations and the maximum FLINT+ reference clock frequency for each netlist. The system clock period is set to 90 ns for all adders, 270 ns for all multipliers and 1400 ns for all dividers. The influence of the quantization resolution is denoted as full (1 ps), dashed (10 ps) and dotted (100 ps) lines for the elapsed time. For partial instrumentation applying path instrumentation selection, the elapsed times and speed-up factors for 110%, 120%, 130%, 140%, 160% and 180% maximum FOS are depicted as split branches. Resolution-dependent accuracy comparisons to SDF simulation with 1 ps timing resolution are maximum values for 16-bit adder, multiplier and divider from Sections 4.1 and 4.2 when timing parameter values are rounded to the nearest quantized interval.

delay parameter bitwidth can be reduced, resulting in a resource reduction of about 30 % for 100 ps parameters. If the analysis is constrained to a minimum system clock period, i.e., a limited stochastic operation regime, a full netlist instrumentation is not necessary. By applying a partial instrumentation using the iterative critical path selection feature of the framework, the resource requirement can be reduced to a minimum of about 30% of the full instrumentation according to Table 2. Furthermore, the reduced overhead allows larger designs to be analyzed, e.g., for the 512 to 896-bit adders, a fully-instrumented FPGA bitstream generation is not possible for the Virtex-6 XC6VLX240T-1 FPGA device (denoted by '-' in Table 3). By using partial instrumentation, these units become accessible for a certain stochastic operation range.

In cases where the FPGA resource overhead constrains the applicability of the framework, the toolchain also offers a hierarchical instrumentation mechanism. This mechanism applies instrumentation only to a selection of netlist sub-modules and keeps the original gate-level description for the rest. The functional behavior of the whole circuit is preserved, while the instrumented modules are additionally emulating timing behavior. For example, in pipelined microprocessors, it could be sufficient to just analyze specific datapath units of interest. In such cases, the proposed emulation-based timing analysis framework is applicable by using only a fraction of the FPGA resources needed for a complete instrumentation. Furthermore, for circuits composed of several similar arithmetic blocks, e.g., dedicated image convolution accelerators, time-domain multiplexing techniques may be used for timing emulation, where only one arithmetic block is instrumented and sequentially used for analysis of the complete block-based system. However, this aspect is out of scope of this paper, since the utilization ability of such methods and the expected effective speed-up compared to simulating the complete system is completely application-dependent.

5. Case Study: Stochastic CORDIC Unit

In this case study, the accuracy-performance design space of a CORDIC hardware implementation is explored within the stochastic operation regime. In [25], approximate adders are used in a CORDIC hardware unit to intro-

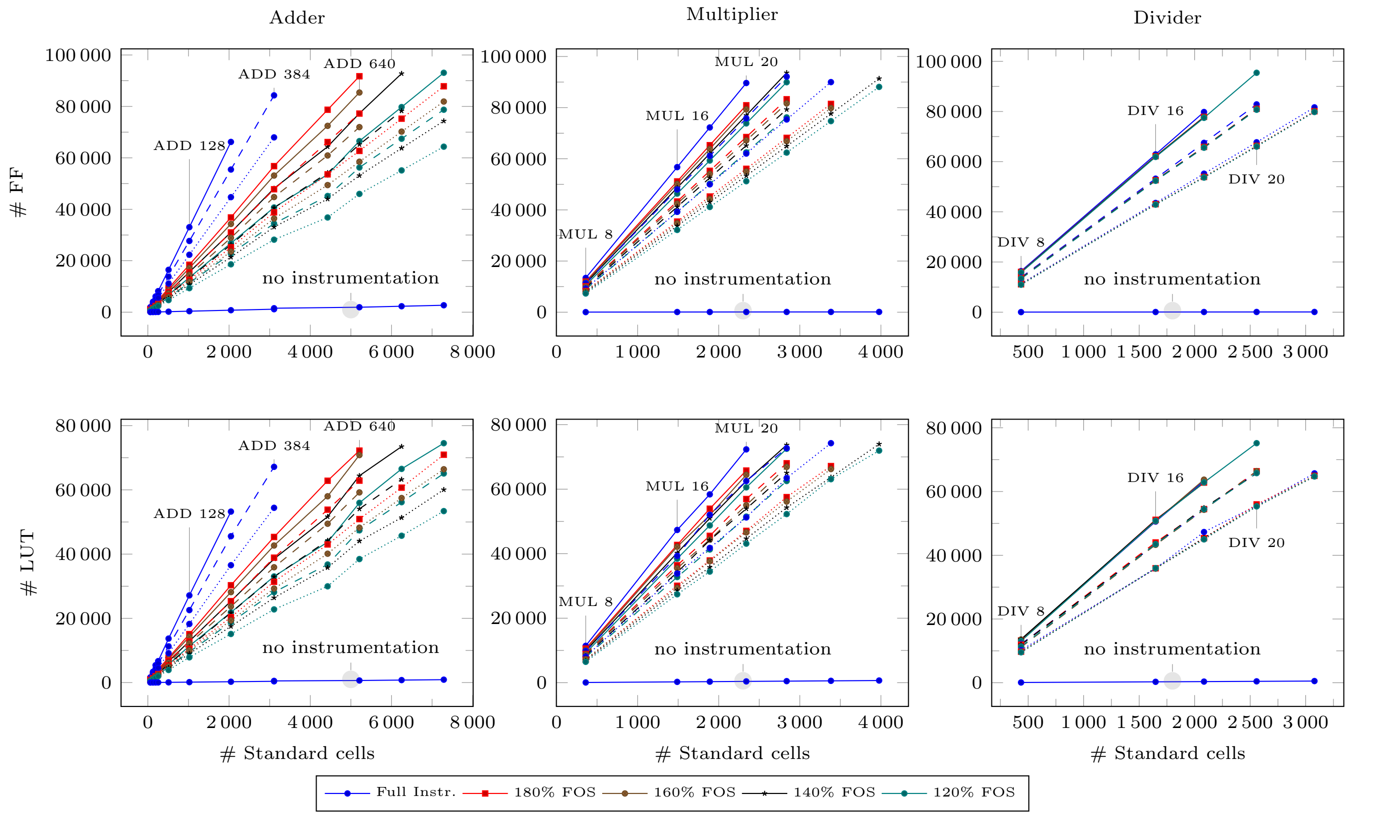


Figure 11: Resource overhead in FFs (top row) and LUTs (bottom row) of instrumented netlists synthesized for a Virtex-6 FPGA device as a function of the maximum frequency overscaling operation regime. The evaluated adder bitwidths are 8, 16, 24, 32, 64, 128, 256, 384, 512, 640, 768 and 896 bits. Multipliers are evaluated for bitwidths of 8, 16, 18, 20, 22 and 24 bits and dividers for bitwidths of 8, 16, 18, 20 and 22 bits. For lower precision *time quanta*, the delay parameter bitwidth and thus the resource overhead is reduced. The influence of the quantization resolution is denoted as full (1 ps), dashed (10 ps) and dotted (100 ps) lines.

duce computational imprecision in the iterative nature of the CORDIC algorithm. The authors of this paper explore the accuracy-efficiency design space in terms of a trade-off between the computational mean absolute error, critical path delay and energy dissipation for different approximate adder configurations. Contrary to the *Approximate Computing* approach of designing imprecise hardware without any timing violations, the proposed FLINT+ framework assists designers to evaluate *Stochastic Computing* methods by violating the circuit timing of either precise or approximate arithmetic units within a reasonable amount of analysis time.

CORDIC [26] is an algorithm to compute a large variety of elementary functions. By applying iterative vector coordinate rotations with a defined set of rotation conditions, only addition and bit shift operations are required to calculate more complex functions. Fig. 12 shows a block diagram of the 32-bit fixed-point CORDIC hardware used in this case study. ASIC gate-level netlists of precise and approximate CORDIC units are synthesized from a structural VHDL description using a 1 µm high-temperature SOI CMOS technology [22].

The iterative coordinate rotation can be controlled to converge to different mathematical functions by selecting a

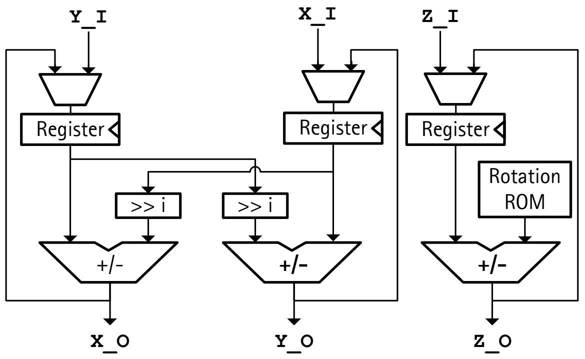


Figure 12: Exemplary CORDIC hardware implementation. `X_I`, `Y_I` and `Z_I` are the preset inputs for the vector coordinates X and Y and the rotation angle Z. `X_O`, `Y_O` and `Z_O` are the corresponding outputs for each iteration. The three addition-subtraction units are either precise or approximate adders, while subtraction is obtained by performing a bitwise invert operation of the second operand and setting the carry-in bit of the adder.

specific operation mode and a parameter ROM table for stepping the rotation angle Z. Since imprecise calculations may result in different convergence properties for these modes, sine calculations (rotation mode, circular step table), square root calculations (vectoring mode, hyperbolic

Table 3: Maximum FPGA Reference Clock Frequencies in MHz After Place-and-Route as a Function of the Netlist Design, Quantization Resolution and Amount of Frequency Overscaling for Partial Instrumentation

Design	Res.	Max. Percentage of FOS						
		110	120	130	140	160	180	Full
ADD 16	100 ps	500	500	500	500	500	500	500
	10 ps	500	500	500	500	500	500	500
	1 ps	500	500	476	476	476	476	476
ADD 64	100 ps	500	500	500	500	500	500	476
	10 ps	476	476	476	476	476	476	476
	1 ps	476	476	476	476	476	476	476
ADD 128	100 ps	476	476	476	476	476	476	455
	10 ps	476	476	476	476	476	455	455
	1 ps	455	455	455	455	455	455	455
ADD 256	100 ps	476	476	476	476	476	455	455
	10 ps	476	476	455	455	455	455	455
	1 ps	455	455	455	455	455	455	455
ADD 384	100 ps	476	476	476	476	476	455	455
	10 ps	455	455	455	455	455	435	345
	1 ps	455	455	455	455	455	435	-
ADD 512	100 ps	476	476	476	476	476	455	-
	10 ps	455	455	455	455	455	435	-
	1 ps	455	455	455	455	435	435	-
ADD 640	100 ps	476	476	476	476	417	417	-
	10 ps	435	435	435	435	417	345	-
	1 ps	435	435	417	400	333	313	-
ADD 768	100 ps	455	455	455	417	417	370	-
	10 ps	435	435	385	323	-	-	-
	1 ps	435	435	-	323	-	-	-
ADD 896	100 ps	417	417	417	370	303	303	-
	10 ps	370	370	222	-	-	-	-
	1 ps	263	143	-	-	-	-	-
MUL 16	100 ps	455	455	455	455	455	455	455
	10 ps	455	435	435	435	435	435	435
	1 ps	435	435	435	435	435	435	435
MUL 18	100 ps	455	455	455	455	455	455	455
	10 ps	455	435	435	435	435	435	435
	1 ps	417	417	385	385	385	385	370
MUL 20	100 ps	417	417	417	417	417	400	400
	10 ps	400	400	400	400	400	400	400
	1 ps	385	345	345	345	345	345	345
MUL 22	100 ps	400	370	357	357	357	357	357
	10 ps	400	333	333	286	278	278	278
	1 ps	286	189	189	125	-	-	-
MUL 24	100 ps	370	370	357	313	313	313	164
	10 ps	303	-	-	-	-	-	-
	1 ps	-	-	-	-	-	-	-
MUL 26	100 ps	323	204	192	182	-	-	-
	10 ps	-	-	-	-	-	-	-
	1 ps	-	-	-	-	-	-	-
DIV 16	100 ps	455	435	435	435	435	435	435
	10 ps	435	435	435	435	435	435	435
	1 ps	417	417	417	417	417	417	417
DIV 18	100 ps	435	435	435	435	435	435	435
	10 ps	417	417	417	417	417	417	417
	1 ps	357	357	345	345	345	345	345
DIV 20	100 ps	417	417	385	385	385	385	385
	10 ps	333	333	333	333	333	333	333
	1 ps	270	244	-	-	-	-	-
DIV 22	100 ps	323	323	323	323	323	323	323
	10 ps	172	-	-	-	-	-	-
	1 ps	-	-	-	-	-	-	-

step table) and exponential calculations (rotation mode, hyperbolic step table) are selected for the case study to cover typical functions in different operation modes. The operation arguments are given as 1° to 90° in steps of 1° for the sine function, 0.05 to 0.74 in steps of 0.01 for the square root function, and $-\frac{\pi}{4}$ to $\frac{\pi}{4}$ in steps of 0.01 for the exponential function, respectively.

In this case study, the proposed stochastic CORDIC

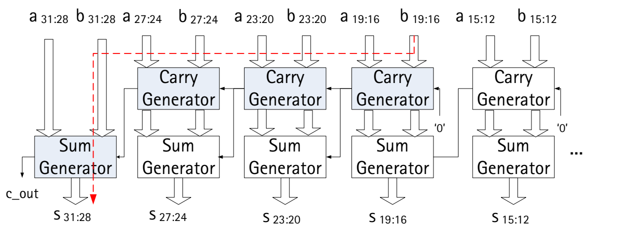


Figure 13: Block diagram of a ETA2-M approximate adder with a generator blocksize of 4 bits and a carry generator extension of two additional blocks. Following the name convention of this paper, this unit is called ETA4-2. The critical path of the design is depicted as a dashed red line.

unit is implemented either with precise *Ripple-Carry Adder* (RCA) or approximate *Modified Error-Tolerant Adder Type 2* (ETA2-M) [27] adders. The ETA2-M approximate adder is segmented into equally-sized sum and carry generator blocks, as depicted in Fig. 13. Then, RCA adders are used for the sum generators and a ripple-carry chain without sum bit generation is used for the carry generators. Since the carry propagation chain is interrupted, approximation errors occur when a carry bit has to be propagated across block boundaries to obtain a correct carry input for a sum generator. To reduce the error magnitude of the ETA2-M adder, several carry generators can form a longer carry chain for the most significant sum bits. In the following, a name convention of the format *ETAa-b* is applied to denote a generator blocksize of *a* bits and an additional extension of *b* carry generator blocks to obtain the carry input for the most significant sum bits. The ETA2-M adder produces *frequent errors with small magnitude* (FSM) and is therefore suitable for approximations of X and Y vector coordinates as well as of the rotation angle Z, because the iteration convergence of the algorithm is not disturbed by large error magnitudes [25].

5.1. Quality-Performance-Area Exploration

The goal of this case study is to find Pareto-optimal CORDIC configurations that offer high computational accuracy and performance. As accuracy metric, the *Signal-to-Noise Ratio* (SNR) compared to the double-precision floating-point MATLAB implementation of elementary functions is evaluated. The system clock period for one CORDIC iteration is defined as a performance metric, where smaller values result in a higher performance.

When *Stochastic Computing* methods are not applied, the only possibility to improve the processing performance is to reduce the critical path delay of the CORDIC unit. The ETA2-M approximate adder trades off accuracy for a smaller delay by interrupting carry propagation. When a RCA is used for X, Y and Z adders, the most critical paths are located in the X and Y branch through the shifter, adder and register input multiplexer. By inserting a ETA2-M architecture for X and Y adder, the critical path is moved to the Z branch of the CORDIC unit and can be reduced by inserting the same ETA2-M architecture for the Z adder. Then, the critical path is moved back to the X and Y branches. By applying the described scheme, a successive critical path delay reduction is achieved and shown in Table 4, while the SNR is reduced and the circuit area increases due to additional carry generators of the ETA2-M architecture. In the following, each configuration of a CORDIC unit is denoted as *adder_xy* & *adder_z*, as the same adder architecture is used for the vector coordinates X and Y. For the rotation angle Z, the adder unit may differ. It has to be pointed out that more ETA2-M architectures or other approximate adder designs can be evaluated in larger analysis campaigns, however, this aspect is out of scope of this paper.

The FLINT+ framework is then applied to explore the stochastic operation regime at environment temperature corner cases of 250 °C and 175 °C. The top and third row of Fig. 14 illustrate the analysis results. The stochastic regime of each adder unit is denoted by gray dashed lines in the plots on the number of instrumented cells (second and bottom row), where the upper limit is the critical path and the lower limit is the lowest clock period where at least one sum bit of each X, Y and Z adder is guaranteed to be still correct. For the sine, square root and exponential functions, the average SNR as a function of the clock period indicates that the ETA2-M configurations do not always outperform the RCA for a stochastic performance gain. For example, sine calculations reach the highest average SNR with an ordinary RCA for clock periods above 300 ns at 250 °C or 200 ns at 175 °C, although the critical path of the RCA is already violated. Below these clock periods, the SNR of the RCA drops significantly and the ETA8-1 and ETA8-0-based CORDIC configuration achieve a higher sine accuracy in the stochastic regime. Similar observations can be made for the exponential calculations.

For square root operations, the SNR-performance Pareto front (the upper slope of all curves) has a different shape because in vectoring mode, the sign bits of X and Y are responsible for iteration control instead of the rotation angle Z. A different iteration process therefore results in different error characteristics. When operated at 250 °C, the RCA has the highest SNR above a clock period of 300 ns. Below that, when using a ETA8-1 and ETA8-0 CORDIC configuration, higher average accuracy can be reached. At 175 °C, there is a small interval between 120 and 150 ns for which the ETA4-1 & ETA8-0 configuration has the highest accuracy. Summarizing the observations, the stochastic accuracy-performance design space offers trade-off potential even for a small case study.

Fig. 15 illustrates the normalized *mean absolute error* difference when the *timing resolution* of the emulation is reduced to 10 or 100 ps. A reasonable system clock period between 436 and 142 ns is assumed, which corresponds to the range from the critical path and thus the first path violation for the RCA & RCA CORDIC configuration down to the violation of all paths within at least one of the X, Y and Z adder. The deviations are below 1.5% for 10 ps resolution and below 4% for 100 ps resolution for most of the clock periods. If the analysis accuracy is sufficient, using a coarser timing quantization enables high speed-up factors for large exploration scenarios.

Table 4: Circuit Area, Critical Path (250 °C) and Non-Stochastic Average SNR for ETA CORDIC Configurations

CORDIC Config.	Area	Crit. Path	Non-Stoch. SNR (dB)		
X,Y & Z Adder	(mm^2)	(ns)	SINE	SQRT	EXP
RCA & RCA	1.978	436	168	147	168
ETA8-1 & RCA	2.086	375	100	109	121
ETA8-1 & ETA8-1	2.128	347	97	109	115
ETA8-0 & ETA8-1	2.179	307	96	83	114
ETA8-0 & ETA8-0	2.189	271	74	83	75
ETA4-1 & ETA8-0	2.229	254	55	56	55

5.2. Emulation vs. Simulation Speed-Up

In Fig. 16, the elapsed time for FLINT+ emulation and SDF timing simulation as well as the speed-up factor is depicted for the analysis of all evaluated CORDIC operations for one specific system clock period of 250 ns. Since the design size of the different CORDIC configurations only varies within a small range of 1533 to 1791 standard cells, the FPGA routing complexity and therefore the maximum reference clock frequency of about 400 MHz are comparable for all evaluated CORDIC configuration designs.

By reducing the *timing resolution* to 100 ps or performing a partial netlist instrumentation with the critical path selection algorihm for a maximum stochastic condition of 120% FOS, the reference clock frequency can be increased to 455 MHz. When a reference clock *quantization interval* of 100 ps is used, a maximum speed-up factor of 476 is obtained for the ETA8-1 & ETA8-1 CORDIC configuration. If a large stochastic timing analysis campaign of about six months in SDF timing simulations is considered, this factor scales to about half a day of FLINT+ timing analysis, massively accelerating analysis tasks of stochastic hardware designers when a reduced *timing resolution* is sufficient.

5.3. FPGA Resources

In Fig. 17, the required FPGA resources for timing emulation are illustrated. With a maximum amount of 72 000 flip-flops and 60 000 LUTs, all evaluated designs fit into the FPGA device at full *timing resolution* of 1 ps and a complete netlist instrumentation. Nevertheless, the critical path selection is an effective way to reduce the resource overhead for CORDIC designs. Because the reasonable stochastic regime for the adders within the CORDIC unit does not cover the full clock period range (second and bottom row in Fig. 14), 7% to 9% of the ASIC netlist gates never require instrumentation because paths traversing them will be violated below the border of stochastic adder operation (142 ns at 250 °C). Thus, critical path selection can be universally applied for all configurations to save unnecessary instrumentation amount if the analyzed stochastic regime is limited by a minimum clock period.

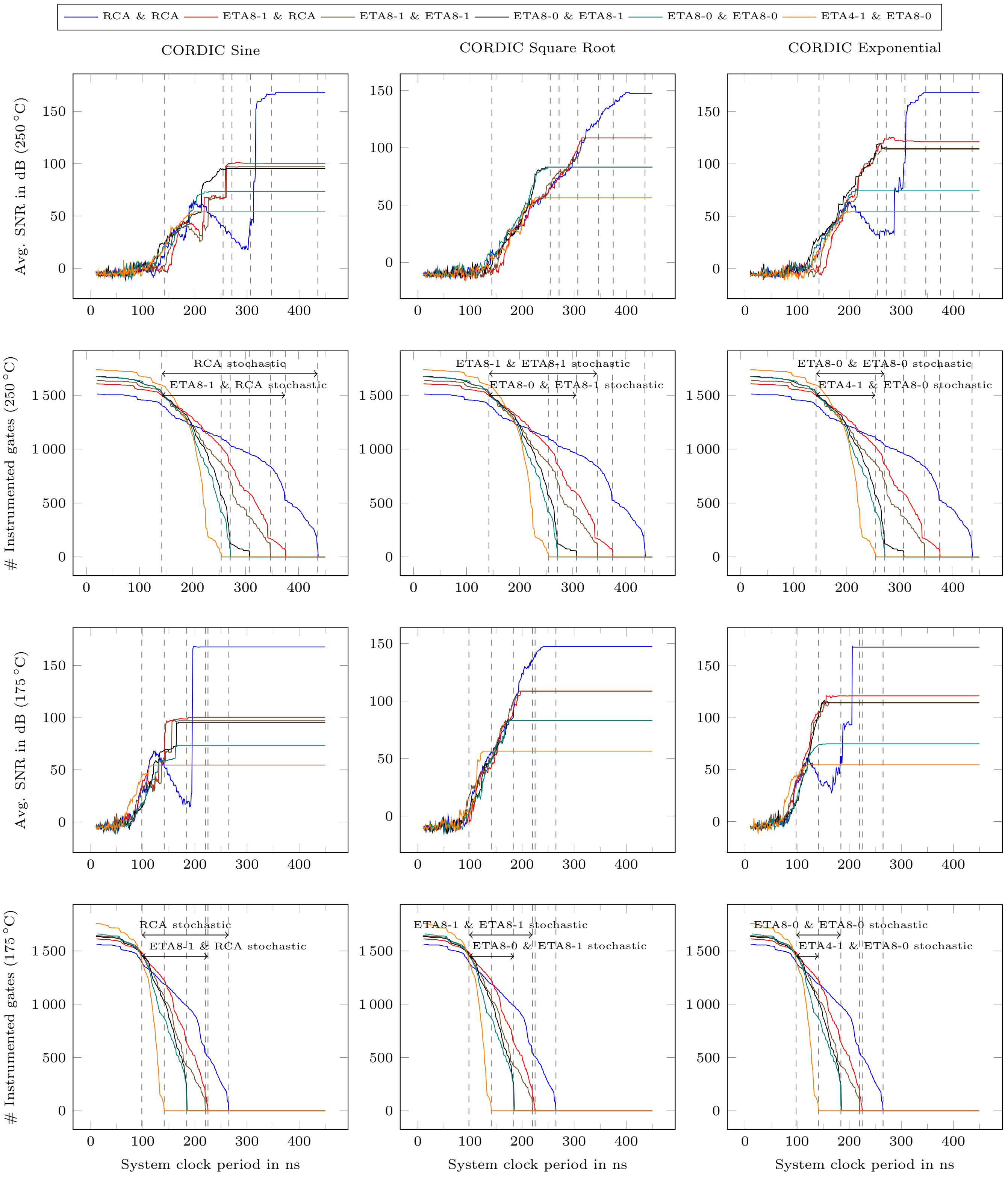


Figure 14: Top row: Average SNR of sine, square root and exponential functions for CORDIC calculations in the stochastic regime for a 250 °C temperature corner case. The adder configurations are denoted as follows: *adder_xy* & *adder_z*. Second row: Number of required instrumented gates as a function of the system clock period for a 250 °C temperature corner case. Third row: Average SNR of sine, square root and exponential functions for CORDIC calculations in the stochastic regime for a 175 °C temperature corner case. Bottom row: Number of required instrumented gates as a function of the system clock period for a 175 °C temperature corner case.

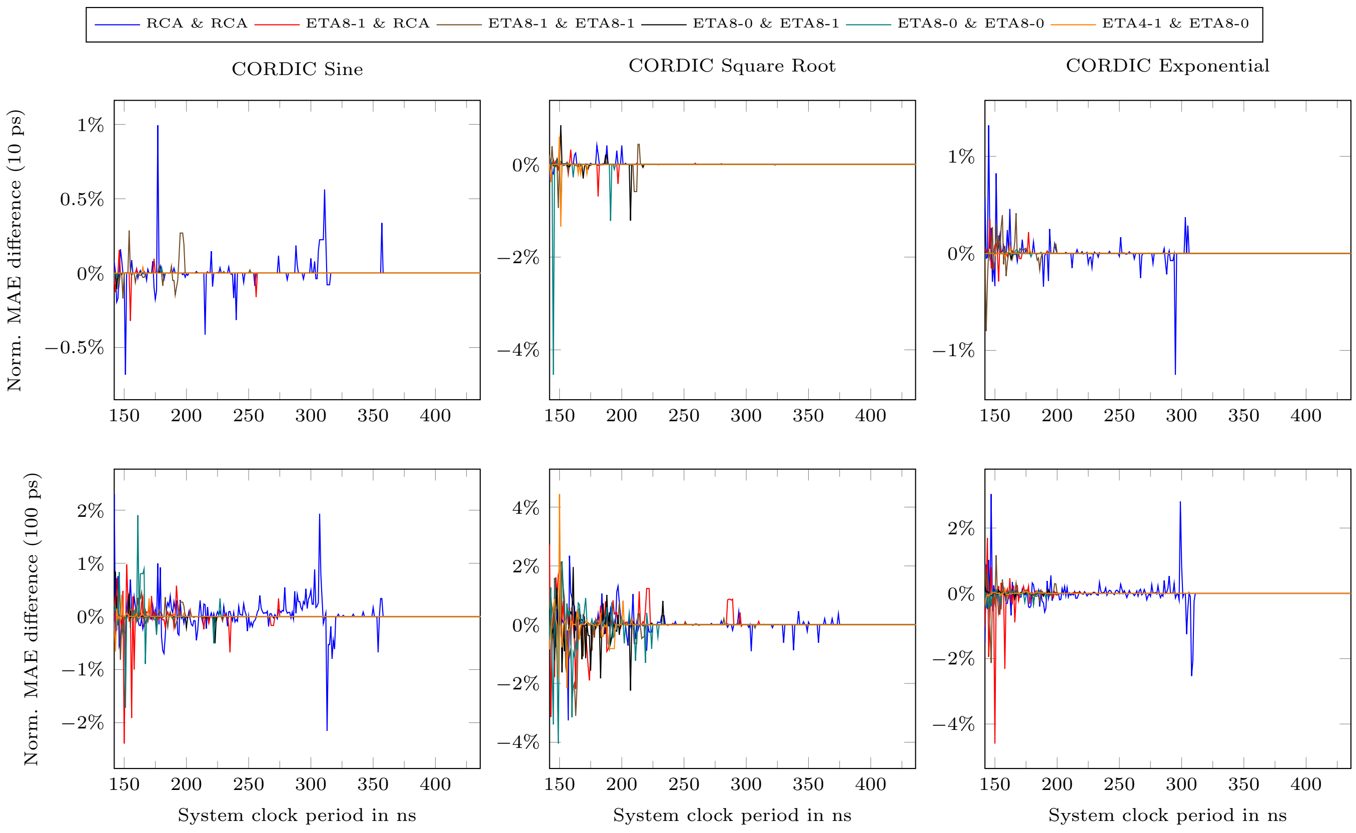


Figure 15: Top row: Mean emulation *error magnitude* difference of CORDIC functions for a quantization resolution of 10 ps and a 250 °C temperature corner case to the 1 ps reference. Bottom row: Mean emulation *error magnitude* difference of CORDIC functions for a quantization resolution of 100 ps and a 250 °C temperature corner case to the 1 ps reference.

Finally, it has to be pointed out that the stochastic regime scales approximately linear with the temperature corner case. For example, the considered clock period range for the RCA & RCA CORDIC configuration covers 436 ns down to 142 ns at 250 °C, which corresponds to a maximum 307% of frequency overscaling compared to the critical clock frequency. At 175 °C, the reasonable clock period covers a range of 265 ns down to 98 ns until all paths within one of the adders are violated, which corresponds to 270% FOS. Furthermore, the plots of required instrumented gates in Fig. 14 follow the same shape for 250 °C and 175 °C, so a single timing-instrumented netlist for a certain stochastic range at the worst temperature corner case can also reflect a comparable analysis range at other temperature corner cases. Thus, when applying partial instrumentation by critical path selection, a single runtime-configurable FPGA bitstream as provided by the FLINT+ framework can still be used to estimate the stochastic behavior in other temperature corner cases than the worst case for which the necessary gate instrumentation was derived prior to FPGA bitstream generation.

6. Conclusion

In this paper, a FPGA-based timing analysis framework is presented. A gate-level netlist instrumentation approach of emulating gate propagation delays in hardware via time quantization and reference clock cycle counters is implemented and enhanced by a scan chain-like runtime configuration mechanism for the delay parameters. This feature supports speeding-up timing analysis campaigns for *Stochastic Computing* at hundreds to thousands of environmental corner case combinations without the need to generate a FPGA bitstream for each corner case of one specific ASIC design. Furthermore, a critical path selection algorithm is integrated into the framework toolflow to select only timing-relevant gates for the instrumentation of a limited, user-defined stochastic analysis regime. By omitting the instrumentation model from gates that will not contribute to stochastic errors induced by path timing violations, the FPGA resource overhead due to instrumentation is reduced.

The choice of *timing resolution* is an important factor for the trade-off between emulation speed-up and the accuracy of the full-circuit timing emulation compared to logic simulations. For full timing instrumentation of profiled ripple carry adders, ripple carry array multipliers and non-restoring array dividers, speed-up factors of up to 57 are obtained for a resolution of 10 ps, while the mean error magnitude of the analyzed stochastic errors deviates by a maximum of 0.3 % from a SDF timing simulation at a 1 ps

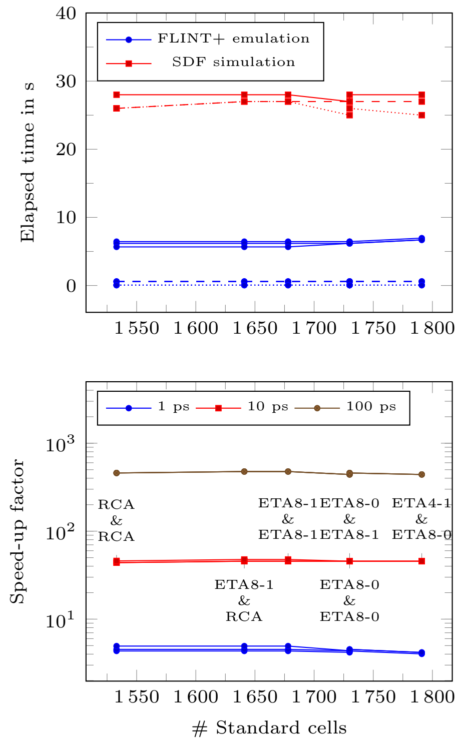


Figure 16: Comparison of elapsed time (top plot) and speed-up factor (bottom plot) for instrumented netlist emulation and SDF timing simulation. The system clock period is set to 250 ns for all CORDIC units and the timing corner case for 250 °C is used. The influence of the quantization resolution is denoted as full (1 ps), dashed (10 ps) and dotted (100 ps) lines for the elapsed time. For partial instrumentation applying path instrumentation selection, the elapsed times and speed-up factors for 110%, 120%, 130%, 140%, 160% and 180% maximum FOS are depicted as split branches.

timescale. If the quantization resolution is set to 100 ps, speed-up factors of up to 519 are obtained, while the mean error magnitude deviates by a maximum of 0.8 %. By using partial netlist instrumentation and the proposed critical path selection algorithm, speed-up factors of up to 1400 are reached at the trade-off of a reduced analysis range for stochastic operation.

By applying the proposed FLINT+ framework to an exemplary CORDIC quality-performance design space exploration, an overall speed-up factor of up to 48 for 10 ps and up to 476 for 100 ps quantization resolution is achieved, while the maximum mean *error magnitude* to timing simulations does not exceed 1.5 % and 4 %, respectively. This allows hardware designers to significantly shorten exhaustive campaign runtimes from several months to one day with sufficient accuracy to rate the effectiveness of the analyzed stochastic circuit techniques.

Acknowledgment

This work was partly funded by the German Research Council (DFG) under project number PA 2762/1-1.

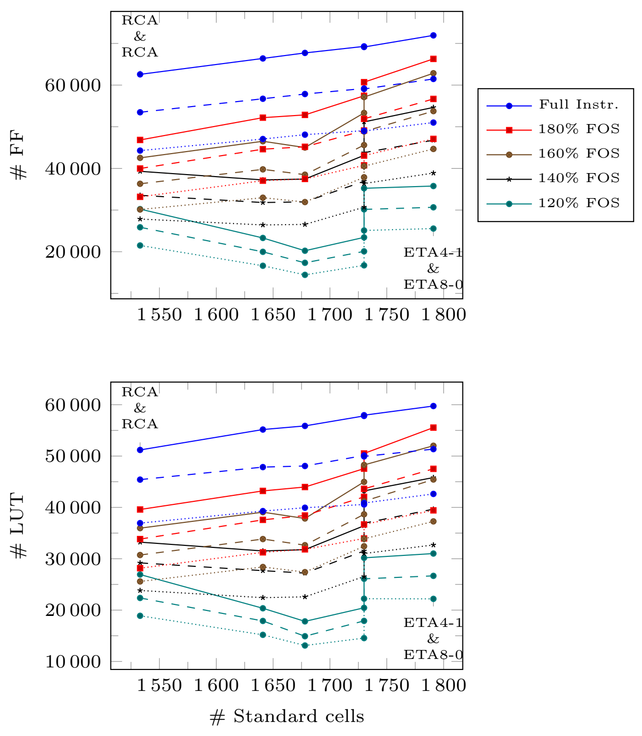


Figure 17: Resource overhead in FFs (top plot) and LUTs (bottom plot) of instrumented CORDIC unit netlists synthesized for a Virtex-6 FPGA device as a function of the maximum frequency overscaling operation regime. For lower precision *time quanta*, the delay parameter bitwidth and thus the resource overhead is reduced. The influence of the quantization resolution is denoted as full (1 ps), dashed (10 ps) and dotted (100 ps) lines.

References

[1] J. Sartori, J. Sloan, R. Kumar, Stochastic Computing: Embracing Errors in Architecture and Design of Processors and Applications, in: Compilers, Architectures and Synthesis for Embedded Systems (CASES), 2011 Proc. of the 14th Int. Conf. on, 2011, pp. 135–144.

[2] J. Han, M. Orshansky, Approximate Computing: An Emerging Paradigm for Energy-Efficient Design, in: Test Symposium (ETS), 2013 18th IEEE European, 2013, pp. 1–6.

[3] A. Alaghi, C. Li, J. P. Hayes, Stochastic Circuits for Real-Time Image-Processing Applications, in: 2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC), 2013, pp. 1–6.

[4] P. Li, D. J. Lilja, Using Stochastic Computing to Implement Digital Image Processing Algorithms, in: 2011 IEEE 29th Int. Conf. on Computer Design (ICCD), 2011, pp. 154–161.

[5] M. Weißbrich, G. Payá-Vayá, L. Gerlach, H. Blume, A. Najafi, A. Garcia-Ortiz, FLINT+: A Runtime-Configurable Emulation-Based Stochastic Timing Analysis Framework, in: Power and Timing Modeling, Optimization and Simulation (PATMOS), 2017 Proc. of the 27th Int. Conf. on, 2017.

[6] M. Violante, Accurate Single-Event-Transient Analysis via Zero-Delay Logic Simulation, IEEE Transactions on Nuclear Science 50 (6) (2003) 2113–2118.

[7] D. Alexandrescu, L. Anghel, M. Nicolaidis, Simulating Single Event Transients in VDSM ICs for Ground Level Radiation, Journal of Electronic Testing 20 (4) (2004) 413–421.

[8] M. Aguirre, V. Baena, J. Tombs, M. Violante, A New Approach to Estimate the Effect of Single Event Transients in Complex Circuits, IEEE Transactions on Nuclear Science 54 (4) (2007) 1018–1024.

[9] M. Valderas, R. Cardenal, C. L. Ongil, M. Garcia, L. Entrena, SET Emulation Under a Quantized Delay Model, in: Defect and Fault-Tolerance in VLSI Systems, 2007. DFT'07. 22nd IEEE Int. Symposium on, 2007, pp. 68–78.

[10] L. Entrena, M. G. Valderas, R. F. Cardenal, M. P. Garcia, C. L. Ongil, SET Emulation Considering Electrical Masking Effects, IEEE Transactions on Nuclear Science 56 (4) (2009) 2021–2025.

[11] L. Entrena, M. Garcia-Valderas, R. Fernandez-Cardenal, A. Lindoso, M. Portela, C. Lopez-Ongil, Soft Error Sensitivity Evaluation of Microprocessors by Multilevel Emulation-Based Fault Injection, IEEE Transactions on Computers 61 (3) (2012) 313–322.

[12] K. Wu, H. Pahlevanzadeh, P. Liu, Q. Yu, A New Fault Injection Method for Evaluation of Combining SEU and SET Effects on Circuit Reliability, in: 2014 IEEE Int. Symposium on Circuits and Systems (ISCAS), 2014, pp. 602–605.

[13] C. H. Chen, S. Song, Z. Zhang, An FPGA-Based Transient Error Simulator for Resilient Circuit and System Design and Evaluation, IEEE Transactions on Circuits and Systems II: Express Briefs 62 (5) (2015) 471–475.

[14] M. Ebrahimi, A. Evans, M. B. Tahoori, E. Costenaro, D. Alexandrescu, V. Chandra, R. Seyyedi, Comprehensive Analysis of Sequential and Combinational Soft Errors in an Embedded Processor, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 34 (10) (2015) 1586–1599.

[15] R. Nowosielski, J. Hartig, G. Paya-Vaya, H. Blume, A. Garcia-Ortiz, Exploring Different Approximate Adder Architecture Implementations in a 250 °C SOI Technology, in: Proc. of ICT.OPEN 2015, 2015.

[16] R. Nowosielski, L. Gerlach, S. Bieband, G. Paya-Vaya, H. Blume, FLINT: Layout-Oriented FPGA-Based Methodology for Fault Tolerant ASIC Design, in: Proc. of the 2015 Design, Automation & Test in Europe Conference & Exhibition, EDA Consortium, 2015, pp. 297–300.

[17] A. Murakami, S. Kajihara, T. Sasao, I. Pomeranz, S. M. Reddy, Selection of Potentially Testable Path Delay Faults for Test Generation, in: Proc. International Test Conference 2000, 2000, pp. 376–384.

[18] X. Lu, Z. Li, W. Qiu, D. M. H. Walker, W. Shi, Longest-Path Selection for Delay Test under Process Variation, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 24 (12) (2005) 1924–1929.

[19] Open Verilog International, Standard Delay Format Specification, version 3.0, May 1995.

[20] Xilinx, Virtex-6 FPGA Configurable Logic Block User Guide (UG364), version 1.2, 03.02.2012.

[21] M. Kock, S. Hesselbarth, M. Pfitzner, H. Blume, Hardware-Accelerated Design Space Exploration Framework for Communication Systems, Analog Integrated Circuits and Signal Processing 78 (3) (2014) 557–571.

[22] Fraunhofer Institute for Microelectronic Circuits and Systems [online]. `www.ims.fraunhofer.de` [accessed 24.09.2018].

[23] Xilinx, ML605 Hardware User Guide, version 1.8, 02.10.2012.

[24] Mentor ModelSim [online]. `www.mentor.com` [accessed 24.09.2018].

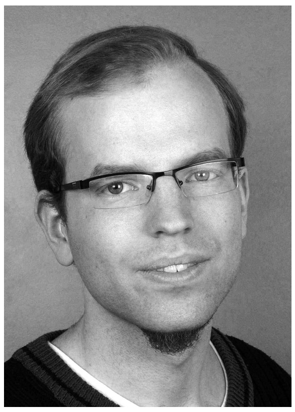
[25] J. Huang, J. Lach, Exploring the Fidelity-Efficiency Design Space Using Imprecise Arithmetic, in: 16th Asia and South Pacific Design Automation Conf. (ASP-DAC 2011), 2011, pp. 579–584.

[26] J. S. Walther, A Unified Algorithm for Elementary Functions, in: Proc. of the May 18-20, 1971, Spring Joint Computer Conf., AFIPS '71 (Spring), 1971, pp. 379–385.

[27] N. Zhu, W. L. Goh, K. S. Yeo, An enhanced low-power high-speed adder for error-tolerant application, in: Proceedings of the 2009 12th International Symposium on Integrated Circuits, 2009, pp. 69–72.

Biography

Moritz Weißbrich received his M.Sc. degree in electrical engineering from Leibniz Universität Hannover, Germany, in 2016. Mr. Weißbrich is currently working as a research engineer towards the Ph.D. degree at the Institute of Microelectronic Systems, Leibniz Universität Hannover. His research interests include high-performance and low-energy processor architectures using approximate arithmetic and Stochastic Computing approaches.

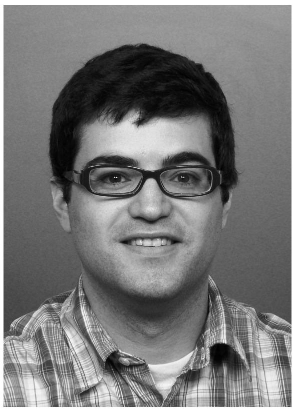


Lukas Gerlach received his diploma in electrical engineering from Leibniz Universität Hannover, Germany, in 2013. Currently, Mr. Gerlach is working as a Ph.D. research engineer at the Institute of Microelectronic Systems in the Architecture and Systems department, Leibniz Universität Hannover. His main research topics are low-power application-specific architectures for hearing aid devices.

Holger Blume received his diploma in electrical engineering in 1992 at the University of Dortmund, Germany. In 1997 he achieved his Ph.D. with distinction from the University of Dortmund, Germany. Until 2008 he worked as a senior engineer and as an academic senior councilor at the Chair of Electrical Engineering and Computer Systems (EECS) of the RWTH Aachen University. In 2008 he got his postdoctoral lecture qualification. Holger has been Professor for "Architectures and Systems" at the Leibniz Universität Hannover, Germany, since July 2008 and manages the Institute for Microelectronic Systems. His present research includes algorithms and heterogeneous architectures for digital signal processing, design space exploration for such architectures as well as research on the corresponding modeling techniques.

Ardalan Najafi received his M.Sc. degree in Electronics Engineering from Shahid Beheshti University of Tehran, Iran, in 2013. He is currently working towards the Ph.D. degree in the Institute of Electrodynamics and Microelectronics at the University of Bremen, Germany. His research interests focus mainly on low-power computing units using stochastic and sub-threshold approaches.

Alberto García-Ortiz is currently full professor for the chair of integrated digital systems at the University of Bremen, Germany. Dr. Garcia-Ortiz received the "Outstanding dissertation award" in 2004 from the European Design and Automation Association. In 2005, he received from IBM an innovation award for contributions to leakage estimation. He serves as editor of JOLPE and is reviewer of several conferences, journals, and European projects. His interests include low-power design and estimation, communication-centric design, SoC integration, and variations-aware design.



Guillermo Payá-Vayá obtained his Ing. degree from the School of Telecommunications Engineering, Universidad Politécnica de Valencia, Spain, in 2001. During 2001-2004, he was a member of the research group of Digital System Design, Universidad Politécnica de Valencia, where he worked on VLSI dedicated architecture design of signal and image processing algorithms using pipelining, retiming, and parallel processing techniques. In 2004, he joined the Department of Architecture and Systems at the Institute of Microelectronic Systems, Leibniz Universität Hannover, Germany, and received a Ph.D. degree in 2011. He is currently Junior Professor with the Institute of Microelectronic Systems, Leibniz Universität Hannover, Germany. His research interests include embedded computer architecture design for signal and image processing systems.