# Loop Unrolling for Energy Efficiency in Low-Cost FPGAs

NAVEEN KUMAR DUMPALA, SHIVUKUMAR B. PATIL, DANIEL HOLCOMB, AND RUSSELL TESSIER, University of Massachusetts Amherst

FPGAs are used for a wide variety of computations in low-cost embedded systems. Although these systems often have modest performance constraints, their energy consumption must typically be limited. Many FPGA applications employ repetitive loops that cannot be straightforwardly split into parallel computations. Performing a loop sequentially generally requires high-speed clocks that consume considerable clock power and sometimes require clock generation using a phase-locked loop (PLL). Loop unrolling addresses the high-speed clock issue, but its use often leads to significant combinational glitch power.

In this work, a computer-aided design (CAD) approach that unrolls loops for designs targeted to low-cost FPGAs is described. Our approach considers latency constraints in an effort to minimize energy consumption for loop-based computation. To reduce glitch power, a glitch filtering approach is introduced that provides a balance between glitch reduction and design performance. Glitch filter enable signals are generated and routed to the filters using resources best suited to the target FPGA. Our approach automatically inserts glitch filters and associated control logic into a design prior to processing with FPGA synthesis, place, and route tools. Our energy-saving loop unrolling approach has been evaluated using five benchmarks often used in low-cost FPGAs. The energy-saving capabilities of the approach have been evaluated for an Intel Cyclone IV and a Xilinx Artix-7 FPGA using board-level power measurement. The use of unrolling and glitch filtering is shown to reduce energy by at least 65% for an Artix-7 device and 50% for a Cyclone IV device while meeting design latency constraints.

CCS Concepts: • **Computer systems organization → Reconfigurable computing**;

Additional Key Words and Phrases: Field-Programmable Gate Array, loop unrolling, energy

**ACM Reference format:**
Naveen Kumar Dumpala, Shivukumar B. Patil, Daniel Holcomb, and Russell Tessier. 2018. Loop Unrolling for Energy Efficiency in Low-Cost FPGAs. *ACM Trans. Reconfig. Technol. Syst.* 10, 4, Article 39 (December 2018), 24 pages.
https://doi.org/0000001.0000001

## 1 INTRODUCTION

Field-programmable gate arrays (FPGAs) are vital components in many embedded computing applications. These devices are used in a wide array of embedded systems ranging from health monitors to low-end portable appliances. A substantial fraction of these systems deploy low cost FPGAs that use a slow system clock of less than 100 MHz to limit clock power [17]. These designs

This work was supported by the National Science Foundation under grant CNS-1619558 and a contract from the Semiconductor Research Corporation.
Author's addresses: N. Dumpala, S. B. Patil, D. Holcomb and R. Tessier, Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003.

1936-7406/2018/12-ART39 $15.00
https://doi.org/0000001.0000001

generally benefit from low dynamic energy consumption due to reduced design throughput and limited clock generation circuitry. The use of a single system clock for the entire design eliminates the need for using high-energy phase-locked loops (PLL) for clock generation. The systems often have strict data latency requirements to ensure that data results are produced at an acceptable pace. Given the use of these systems in mobile platforms, minimizing energy consumption while meeting application latency constraints becomes a key issue.

Iterative loop-based computations are important parts of many low-cost FPGA application implementations. For example, block ciphers such as AES-256 and SIMON-128 require multiple repetitive rounds of computation for each data input. Due to inherent data dependencies, these rounds cannot be easily parallelized. As a result, sequential loop implementations in which a single round of logic is performed repetitively at high clock speed are often used to process intermediate results until a final result is generated. Although dynamic energy can be conserved in these cases by running the sequential implementations at reduced clock speeds, application latency requirements may not be met. For computations such as block ciphers that typically process data in transit to or from an embedded FPGA, a large latency increase may be unacceptable.

Loop unrolling, which involves the combinational chaining of multiple copies of a loop body, has been used to reduce dynamic power consumption in FPGAs [3]. The approach can save dynamic power by allowing the use of a lower frequency clock across a reduced number of iterations while still meeting the required latency of the loop-based computation. A limitation of the combinational chaining used by unrolling is the creation of numerous signal glitches that propagate through the chain. These spurious signal transitions are caused by combinational signal path mismatches and can consume significant dynamic power. Typically, the more sizable the amount of unrolling, the higher the amount of glitching. As we show later in the article, for block ciphers such as SIMON-128 that contains 68 rounds, glitch power can be dominant.

This article presents a low-overhead approach to precisely unroll loop-based computations targeted to low-cost FPGAs. User-specified latency constraints are considered during the unrolling process to minimize dynamic energy while meeting system performance constraints. To minimize glitch energy, we explore the implementation of glitch filters using both level-sensitive latches and edge-triggered flip flops. These implementations are customized for both Xilinx Artix-7 and Intel Cyclone IV devices. In Artix-7 devices, a series of carry chains are used to delay enable signals for glitch filters. In Cyclone IV devices, chains of lookup tables (LUTs) are used for this purpose. When possible, the system clock used to clock other system circuitry outside of the loop-based computation is used for unrolled circuitry to eliminate the need to generate an additional clock using a phase locked loop (PLL) inside the FPGA. The circuitry needed to perform glitch filtering is automatically generated by our computer-aided design system using a script and inserted into a user's register transfer level (RTL) design.

Our research makes the following specific contributions:

- We show that unrolling can be performed to match loop-based computation latency constraints while minimizing dynamic energy.
- We demonstrate that glitch filtering can reduce dynamic energy for both Cyclone IV and Artix-7 devices for unrolled circuits using circuitry already available in these devices.

In this work, five loop based designs with iteration counts ranging from 14 iterations (AES-256) to 68 iterations (SIMON-128) were examined. The designs were evaluated both with and without unrolling and glitch filters for both Cyclone IV and Artix-7 devices. Variants of each design were mapped to Artix-7 and Cyclone IV-based boards and power measurements were taken using on-board measurement circuitry. It is observed that unrolling and glitch filtering can reduce dynamic energy for the benchmarks by up to 91% for Artix-7 and 97% for Cyclone IV. Our approach is shown

to work even if almost all the logic in the FPGA is used, indicating that glitch filters can effectively be implemented even in area-constrained environments.

In Section 2, related work on loop unrolling for FPGAs and glitch reduction is reviewed. Section 3 provides an overview of our unrolling approach and the implementation of glitch filters and associated control signals. The experimental methodology used to generate our results is described in Section 4 and a numerical analysis of our experimental results is presented in Section 5. In Section 6, conclusions are presented and directions for future work are offered.

## 2 RELATED WORK

### 2.1 Implementing Loop-Based Computation

Loop-based algorithms are widely executed on FPGA-based platforms ranging in complexity from high-end compute stations to low-end embedded devices. Following the arrival of an input value, a loop-based algorithm performs a function for a prespecified number of iterations. The iterations can be implemented by the sequential deployment of a combinational circuit for each iteration, or iterations can be unrolled. If no unrolling is used (sequential implementation), one iteration is computed during each system clock cycle, and the cycle count for completing the computation matches the number of algorithm iterations. For low-cost FPGA implementations that function at low clock speeds, the system clock period may be much larger than the delay of the iteration critical path. Thus, the overall latency of the algorithm is longer than necessary due to serial operation using the system clock. Loop unrolling instantiates multiple algorithm rounds and completes them within a system clock cycle. Fewer clock cycles are required to complete the computation, although required combinational area is increased. Data storage energy for clocked registers is also saved as data is not stored after every loop iteration.

A number of effective loop unrolling techniques have been developed for high-performance FPGA-based computing environments. References [3] and [24] showed that security hash, compression, and image processing algorithms can be unrolled and parallelized to allow for simultaneous access to small memories. In Dragomir *et al.* [12], both loop unrolling and code reorganization is used. This latter technique shifts function calls outside of the loop body to reduce hardware complexity. This work targets performance optimization and memory access constraints. None of these approaches are targeted to low-cost embedded FPGAs in which minimizing clock generation circuitry and lowering dynamic energy consumption are crucial. Loop unrolling has recently been considered for energy reduction in ASICs [18] although clock generation in this environment can be easily customized. Our work optimizes unrolling to minimize energy consumption within loop latency constraints for low-cost FPGAs.

### 2.2 Glitches and Glitch Filtering

A dominant issue for energy consumption in loop-based computation is switching (dynamic) energy. The issue is particularly acute for unrolled computation since glitches generated near an iteration input propagate and fan out through combinational logic to generate additional glitches. Generally, glitches in FPGAs are generated due to mismatched signal arrival times at lookup tables (LUTs). This issue can cause a LUT output to transition multiple times in response to each input change. This transistioning output is then forwarded to additional downstream logic.

A variety of techniques have been developed to address glitch filtering in FPGAs. Most techniques [11], [16], [19] involve the use of combinational path balancing to ensure that logic input signals arrive at the same time, minimizing glitch generation. In general, attempting to match combinational path lengths in FPGA routing in the face of varying process, voltage, and temperature (PVT) is difficult and requires precise analysis and per-device implementation. Reference [16] provides a

PVT-tolerant low overhead approach (< 3% area increase) that involves adding circuit-level delay structures to the FPGA. An alternative CAD-based approach [11] performs a glitch-reducing routing pass after initial FPGA routing to balance the wiring length of signals that generate glitches. The timing of these signals must be precise to avoid glitch generation. In contrast, our approach provides a timing *window* for signal arrival at the glitch filter using existing FPGA structures, avoiding the need for new architectural features. As long as the correct data value enters the filter before the window closes, glitches can be reduced and the correct final value of the signal is forwarded.

Several computer-aided design approaches to address glitching have been proposed. Guarded evaluation for FPGAs [25] can be used to prevent evaluation of combinational paths that do not affect the final circuit output. An alternative CAD approach selectively uses don't care conditions [27] to prevent the formation of glitches. Although guarded evaluation and don't care analysis are effective for shallow paths, unrolled circuitry is often too complex to completely eliminate glitches in this manner. The retiming of logic to prevent glitches is also not effective for long combinational paths [26] since flip flops are widely spaced. We view guarded evaluation, don't care synthesis, and retiming as complementary to our work because they can be used in conjunction with our approach.

Pipelining suppresses glitches as they cannot pass through edge-triggered flip flops before clock signal arrival. Several FPGA projects have explored the use of pipelining to save energy [6], [9], [29], although this approach is generally not applied to low frequency designs. To meet latency constraints, these designs require the generation of higher-speed clocks to perform the pipelining, increasing clock power. For example, Lim *et al.* [20] uses phase-shifted clocks created with phase-locked loops and timed according to LUT delays to drive flip-flops inserted within combinational logic.

The use of glitch filtering to reduce ASIC power has previously been explored [4], [10], [21]. For glitch filtering, ASICs provide a simpler platform than FPGAs since clock generation is more straightforward and delays can be tightly controlled.

### 2.3 Relationship to Prior Work

This article greatly expands upon an earlier conference paper on a similar topic [13]. Here, we consider mapping to both a Xilinx and an Intel FPGA versus the Xilinx-only study performed earlier. Additionally, we test our approach using physical power measurements from FPGA-based boards rather than using simulation only. Our results show that board-level measurements are more reliable and less noisy as the amount of glitches increase. Finally, we consider using both latch and flip-flop based glitch filters (earlier work only used latches), a larger number of loop-based computations (5 versus 2), and evaluation of a design containing both an unrolled circuit and an additional large compute block.

## 3 APPROACH

Our unrolling approach considers the system clock frequency, the delay through each iteration of a loop and the delay associated with glitch filtering. In this section we first consider glitch filtering approaches and then provide details of the unrolling system including the algorithm used for unrolling and FPGA-specific implementations.

### 3.1 Glitch Filter Insertion

In our system, we use the glitch filter implementation shown in Fig. 1 for unrolled designs. The input provided from the launch flip flops goes through one or more iteration circuits. The output of this circuitry is latched using a delayed enable (En) signal. The glitch filter is implemented using

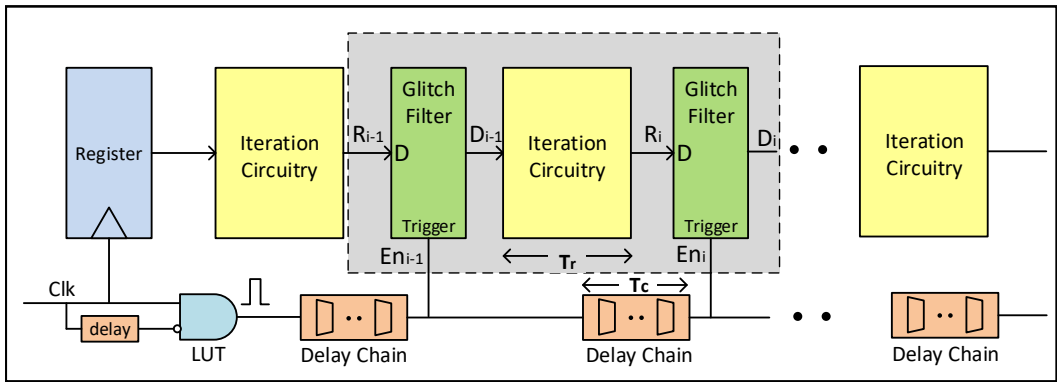


Fig. 1. Unrolled loop with inserted glitch filters

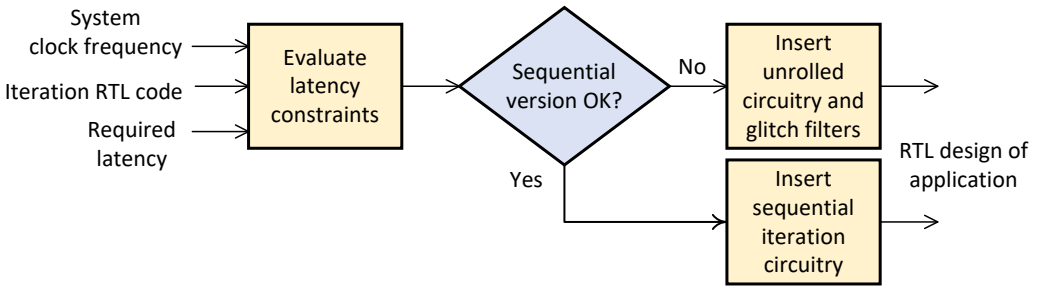


Fig. 2. Software flow for automatic unrolling

either level-sensitive latches or edge-triggered flip flops which, to avoid glitching, are enabled after combinational outputs of the iteration(s) have settled. The enable pulse to the glitch filter is generated by ANDing a delayed and inverted version of the system clock signal with itself. This pulse propagates through delay elements and connects to the enable input of glitch filters. To effectively filter glitches, the propagation delay ($T_c$) used to create delay elements must be greater than the iteration delay ($T_r$). The iteration delay can be computed by synthesizing the design for the target FPGA device and measuring the associated propagation delays using a timing analysis tool. Delay element generation can be parameterized to generate precise delays. In Sections 3.3 and 3.4, the implementation details of delay element generation and glitch filtering for Intel Cyclone IV and Xilinx Artix-7 FPGAs are presented.

### 3.2 Unrolling Algorithm

Our unrolling strategy includes a flow of steps to generate an unrolled and glitch filtered design that meets design latency requirements and minimizes dynamic energy. The basic steps in the unrolling flow are shown in Fig. 2. The inputs to the flow are the RTL code of a single iteration of the computation, the frequency of the system clock, and the latency constraint of the overall loop-based design. After synthesizing a single iteration, the first block in the flow evaluates whether the design latency constraint can be met by simply sequentially using a single iteration and the system clock. If not, the necessary amount of unrolling is determined. Using a script, glitch filtering

| | |
|---|---|
| $L$ | Required computation latency across all iterations |
| $N$ | Number of iterations in computation |
| $R$ | Latency for one iteration |
| $C$ | Clock period of the system clock |
| $F$ | Edge-triggered flip flop latency + setup time |
| $G$ | Per iteration latency for the glitch filter |
| $U$ | Number of times an iteration is unrolled |

Table 1. Parameter definitions for loop unrolling

is added to the partially (or wholly) unrolled block cipher design that is clocked by the system clock. The use of the system clock saves energy by removing the need for the insertion of an energy-hungry PLL.

More formally, our approach analyzes the iteration (round) function and automatically performs unrolling based on the required latency, $L$, of the loop based design from first iteration input until last iteration output. To minimize area, an attempt is made to meet the latency constraint using the system clock with period $C$ over $N$ iterations of the function. The parameters associated with unrolling are listed in Table 1. Formally, the *sequential* version of the design can be used if

$$L \geq (N \times C) \tag{1}$$

and

$$C \geq (R + F) \tag{2}$$

Parameters $R$, $F$, and $C$ are the iteration (round) latency, flip flop latency and setup time, and system clock period, respectively. It is assumed in this case that since only one instance of the computation iteration is used, no glitch filtering is needed. If the sequential version is unable to meet design latency constraints, unrolling is required. The per-iteration latency in this case is $F + U \times (R + G)$, where $G$ is the per-iteration delay of the glitch filter and $U$ is the level of unrolling. Unrolling proceeds until:

$$L \geq (\frac{N}{U} \times C) \tag{3}$$

where $U$ is minimized so the latency condition is met for minimal area. Unrolling can be considered for increasing $U$ while

$$C \geq (F + U \times (R + G)) \tag{4}$$

If unrolling is allowed beyond this point, a slower version of the system clock is needed, increasing hardware resources.

Although timing information from the synthesis and physical design of a single iteration ($R$) can be used to determine the delay through multiple unrollings (as shown in Fig. 2), a more precise timing approach considers synthesis, place and route of unrolled circuity and glitch filters for each unrolling $U$. In this case, the delay of $[U \times (R + G)]$ is determined for each unrolling. A detailed view of our unrolling algorithm is shown in Algorithm 1 which supports both one-time and per-unrolling delay determination. Our results in Section 5 were generated with the latter approach. In general, determining the amount of unrolling needed to meet iteration latency constraints can

1: Given $L$, $C$, $N$ *# comp. latency, clock per., and num iterations*
2: Values $R$, $G$, $F$ determined after synthesis *# determine iteration, glitch filt and FF latency*
3: $U = 1$ *# no unrolling, sequential version*
4: *# Test clock period and computation latency constraints for sequential version*
5: **if** $L \geq (N \times C)$ and $C \geq (R + F)$ **then**
6: Use sequential version; go to **EXIT**
7: **end if**
8: *# Increase unrolling*
9: $U = U + 1$
10: *# While the system clock period longer than unrolled circuitry plus flip flop latency*
11: **while** $C \geq (F + U \times (R + G))$ **do**
12: *# Test clock period and computation latency constraints for unrolled version*
13: **if** $L \geq \frac{N}{U} \times C$ **then**
14: go to EXIT
15: **end if**
16: $U = U + 1$
17: **end while**
18: **EXIT**

**Algorithm 1:** Area and energy-aware unrolling algorithm. Energy is saved by unrolling until the number of unrollings is sufficient at system clock period $C$ to meet the algorithm latency constraint $L$, avoiding the need for PLL insertion

be straightforwardly implemented using a script. If $(R + G)$ is only determined once for a single iteration, lines 11 through 17 in Algorithm 1 effectively reduce to an equation.

### 3.3 Glitch Suppression Circuitry for Cyclone IV FPGAs

The key issues in glitch filter deployment are determining how long each glitch filter trigger should be delayed, implementing the delay line in the FPGA so that triggers are provided at appropriate times, and physically implementing the glitch filter and delay lines. Each of these implementation issues are highly technology dependent and require the use of logic resources located within the FPGA. As mentioned in the previous section, the required delay for each tap $(R + G)$ can be determined through synthesis, place, and route.

*3.3.1 Cyclone IV Delay Chain.* In Intel Cyclone IV devices, delays are implemented using LUTs present in a logic array block (LAB). Each LAB [8] contains 16 logic elements (LE) and each LE consists of a LUT and an edge-triggered flip-flop. Adjacent LUTs in a LAB can be connected input-to-output to generate a delay greater than the iteration delay. Since different LUT inputs can lead to different LUT delays, the input pin that is used is fixed for each LUT in the chain.

Flip flops adjacent to these LUTs can be used for other user logic, if desired. The delay element and AND gate shown at the bottom left of Fig. 1 were implemented using five adjacent LEs, four for delay and one for the AND function. These LUTs also used a pre-assigned input pin to keep LUT delay predictable. The delay through a single LUT was determined to be 155 ps and the average routing delay between LUTs within a LAB is 390 ps for a Cyclone IV EP4CGX150DF21C7 FPGA. For delays greater than 16 LEs, a direct connection from a LUT output in one LAB to a vertically adjacent LAB was used.

*3.3.2 Cyclone IV Glitch Filter Circuitry.* In Cyclone IV devices, a LUT with a feedback path is used for latch implementation. The use of LUTs as latches increases the overhead of latch-based

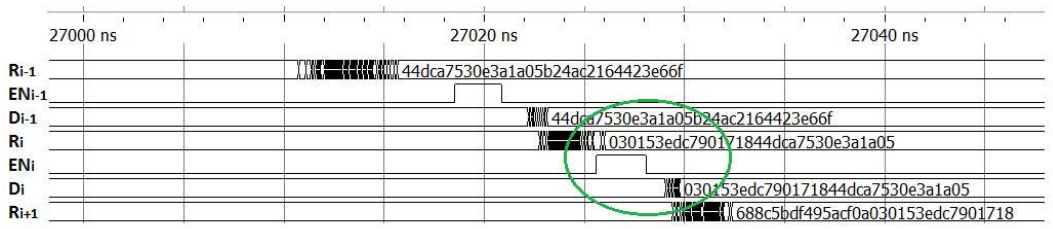


Fig. 3. Glitch filtering timing waveform using latches in a Cyclone IV FPGA

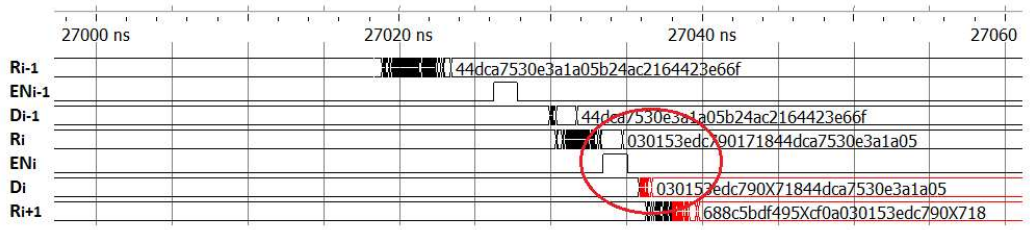


Fig. 4. Glitch filtering timing waveform using flip flops in a Cyclone IV FPGA. An incorrect $D_i$ value (one digit is X) is seen to the right of the red circle.

glitch filters as LUTs are more commonly needed for design logic versus edge-triggered flip flops. A timing waveform of the glitch filtering approach for one round of SIMON-128 computation is shown in Fig. 3. The waveforms were generated via gate-level simulation of the FPGA design using Modelsim, and we have confirmed similar on-chip behavior. Value $R_i$ drives a latch which is sampled using enable $En_i$ and produces a 128-bit $D_i$ with 70 toggled signals. If latching is not used, glitching increases the number of transitions to 130.

Latch-based filters can have an advantage versus flip flops since they provide a *window* for stable data arrival. If the filter enable pulse is slightly early, a small number of glitches may propagate, but the correct final result will be achieved if the pulse is long enough. This flexibility also provides some protection against PVT variations that may affect circuit delays. For example, the filter enable pulse could be slightly lengthened during the design phase for additional PVT protection. For edge-triggered filters, the correct data *must* be presented at the trigger arrival or the filters may propagate incorrect data or enter a metastable state. Fig. 4 shows the result of an enable signal arriving slightly early. An incorrect value is propagated forward (one of the $D_i$ digits is an $X$).

### 3.4 Glitch Suppression Circuitry Implementation for Artix-7 FPGAs

*3.4.1 Artix-7 Delay Chain.* Carry chains are provided in FPGAs to perform fast arithmetic operations. As many block cipher designs do not use arithmetic circuitry, carry chains can be repurposed for delay generation without creating significant area overhead [30]. The organization of a carry chain based delay chain is shown in Fig. 5. Each slice is equipped with four carry multiplexers (MUXCY) which can be cascaded into the carry chain of the next slice. The enable pulse is given to the first MUXCY and an output is taken from the third MUXCY. The MUXCY select lines are preset to allow propagation of the enable pulse through the carry chain. Multiple slices of carry chains are needed to generate delays greater than an iteration. The output from the last slice is connected to the glitch filter and the input of the next delay element carry chain. Delay through

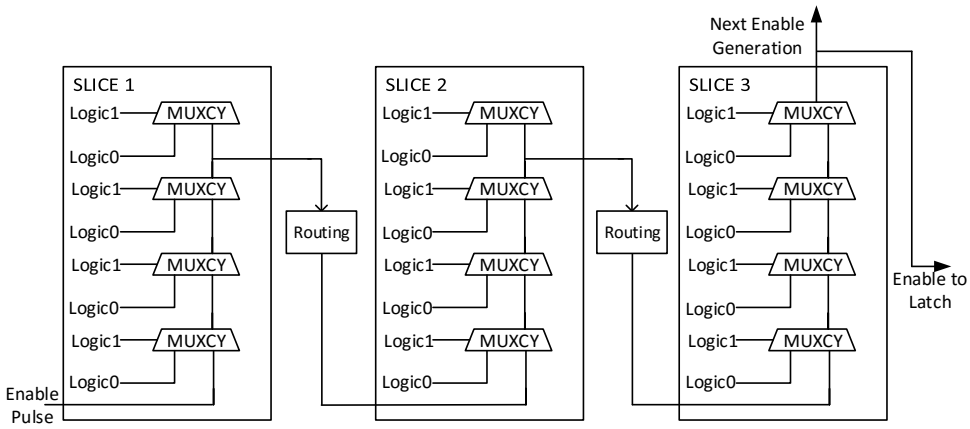


Fig. 5. Carry chain delay implementation in Artix-7 FPGA. The three slices shown in the figure are adjacent to each other making inter-slice routing delays predictable.

a single slice carry chain from the carry input to the output from the third MUXCY (second from the top) was determined to be around 86 ps for a XC7A35TICSG324-1L Artix-7 device. Routing delays between multiplexers in adjacent slices are tightly controlled with routing constraints to use a predictable fast path. The delay element shown at the bottom left of Fig. 1 was implemented using four carry-chain multiplexers and the associated AND gate was implemented in an adjacent LUT with pre-assigned input pins.

*3.4.2 Artix-7 Glitch Filter Circuitry.* The goal of the glitch filter circuitry is to stop the propagation of glitches between iterations by only sampling stable data. In Artix-7 devices, one approach to filtering is to use level-sensitive latches (LDCE primitives). The delayed enable signals ensure that glitches have settled before they are latched and forwarded to the next iteration circuitry.

In our experimentation, we also examined using edge-triggered flip flops that are present in the slice for glitch filtering instead of latches. Similar to the Cyclone IV case, the need for the precise arrival of the trigger edge limits the usefulness of this approach. For Artix-7 based implementations, the overhead of using a D-latch versus a D flip flop is negligible. Four of the eight D flip flops in each Artix-7 cluster can natively be converted to D-latch functionality. The latches are level sensitive and are transparent only when their enable inputs are high. In Section 5 it is seen that latch and flip-flop based filters have roughly equal delay and energy consumption, providing an advantage to latch-based glitch filters since they are also more reliable.

## 4 EXPERIMENTAL METHODOLOGY

In this section, the loop-based applications used for experimentation are described along with the experimental techniques used to measure power via simulation and using FPGA hardware.

### 4.1 Loop-based Applications

Our energy-efficient unrolling approach was applied to five loop-based benchmarks which are summarized in Table 2. The synthesizable SIMON core was written from scratch and validated for correctness against a software implementation. SIMON is a relatively new encryption standard

| Application | type | Iterations | Data (bits) | Key (bits) | Period Artix-7 (ns) | Period Cyclone IV (ns) |
|---|---|---|---|---|---|---|
| SIMON-128 | Security | 68 | 128 | 128 | 340 | 600 |
| AES-256 | Security | 14 | 128 | 256 | 175 | 300 |
| DES | Security | 16 | 64 | 56 | 100 | 360 |
| Bitonic | Sort | 15 | 512 | - | 120 | 220 |
| CORDIC | Math | 15 | 51 | - | 120 | 250 |

Table 2. Details of loop-based applications. SIMON-128, AES-256, and DES are encryption circuits that use an encryption key. The periods of the fully unrolled circuits are included.

optimized for hardware implementation that has a balanced Feistel network structure. We have implemented a version of SIMON-128 [5] which takes 128-bit text and key and requires 68 rounds to encrypt each block. Relative to SIMON, AES is a more complicated design, and we specifically use the most complicated variant, AES-256; which has 128-bit block size, a 256-bit key, and requires 14 rounds per encryption. AES refers to three standardized variants [22] of the Rijndael cipher, based on a substitution-permutation network. The RTL for our AES implementation is publicly available from opencores.org [15], and we validated its correctness against a known-correct AES software implementation. To further test our system, a circuit implementation of the data encryption standard (DES) from opencores.org [28] was used for experimentation. Similar to AES, each of the 16 iterations (rounds) of DES includes a combination of data substitutions via lookup tables, bit permutations, and XOR operations using a key. The subkeys for each round are generated via key shifting and bit permutation. The regularity of both the key and data manipulations facilitates unrolling for this design.

Bitonic sort is a parallel algorithm that can be formulated as a series of iterations. In each iteration, multiple comparisons and swaps of two data values take place in parallel. Although the comparison and data swapping operations in each iteration are the same, the interconnection pattern between the components changes from iteration to iteration. Due to this communication pattern, unrolling options for this benchmark are limited to sequential and fully unrolled. The source Verilog code used for experimentation was obtained from spiral.net [32]. Our experiments were performed with a design which sorts thirty-two 16-bit values. The unrolling algorithm was also applied to a hardware implementation of the CORDIC algorithm [2], which can be used for hyperbolic and trigonometry functions. The algorithm requires 15 iterations of data shifting and addition/subtraction to calculate the required function. In each iteration, the same type of operations are performed so the computation can be unrolled. The CORDIC source code used for experimentation with 51-bit data values was obtained from opencores.org.

Our delay chain circuit for each FPGA technology is parameterized and tunable based on the combinational delay of iteration circuitry. The delay chain for each application was determined after synthesis, place, and route of one or more iterations. For Artix-7 experiments, carry chains of three slices were used for SIMON-128, six slices were used for DES, and seven slices were used for AES-256, Bitonic, and CORDIC. For Cyclone IV experiments, LUT chains of 18 LUTs were used for SIMON-128 and 36 LUTs were used for AES-256, DES, Bitonic, and CORDIC. Unless otherwise noted in Section 5, glitch filters are implemented using latches.

### 4.2 Cyclone IV Experimentation

Intel Cyclone IV FPGAs are low-cost and low power 60 nm counterparts to Stratix IV family components. Experimentation was performed using a Cyclone IV GX (EP4CGX150DF21C7-2)

Table 3. Relevant voltages for Cyclone IV FPGA under test

| Rail | Voltage (V) | Description |
|---|---|---|
| VCCA | 2.5 | PLL analog power |
| VCCD_PLL | 1.2 | PLL digital power |
| VCC | 1.2 | FPGA core power |

device. LUT-based delay chains and filters were instantiated as parameterized macroblocks in the Verilog-based designs described in Section 4.1. Synthesis, place, and route using Quartus Prime version 16.0 was guided by timing constraints specified in a Synopsys design constraints (SDC) file.

*4.2.1 Simulation-based Power Estimation.* Following the generation of a gate-level netlist by the Quartus Prime fitter, gate-level simulation was performed. Signal switching activity was recorded in a value change dump (VCD) file by performing gate level simulations using ModelSim. The Quartus PowerPlay power analyzer was provided with the gate level netlist, VCD simulation trace file, constraints and operating conditions to generate a power consumption report. The *Enable Glitch Filtering* option was ON to allow the simulator to consider the inertial delay of components in the circuit. Sampled power values were scaled by execution time to generate energy values. An initialized on-chip ROM controlled by a free running cycle counter sent new data to the tested design every system clock cycle. The power consumed by the ROM and counter were determined independently and subtracted from the overall FPGA dynamic power to determine the power of each design under test (DUT).

*4.2.2 Board-Level Power Measurement.* A Cyclone IV-GX development board [1] was used to monitor the power consumed by the Cyclone IV FPGA which contains 9,360 logic array blocks (LABs). The board uses a MAX II EPM2210GF256 CPLD and other circuitry for on-board power measurement, as shown in Fig. 6.

The EP4CGX150DF21C7 is powered by 8 supply rails. The board has an 8-channel differential input 24-bit ADC which measures current drawn from these rails with the help of low-value sense resistors. An SPI bus connects the ADC to a MAX II CPLD system controller. Dynamic and static power information was collected from the board and transferred to a PC via a JTAG interface. The same DUT stimulus approach used for Artix-7 testing was used for the Cyclone IV device. The supply rails considered for dynamic power measurement experiments are listed in Table 3. The product of the measured VCC current and the 1.2V supply voltage is the core power. For designs which use a PLL, the total power is the sum of the power contributed by VCCD_PLL, VCCA, and VCC. We measured the static power by programming the FPGA with a circuit with no dynamic activity and then monitoring the supply current. This value was measured to be 141.4 mW in an EP4CGX150DF21C7 FPGA. Only dynamic power values are reported in Section 5. An initialized ROM with the same contents used during simulation was controlled by a free running cycle counter to send new data to the DUT every cycle. The results were collected in an output buffer. The power consumed by the ROM, counter and buffer were determined independently and subtracted from VCC power to determine the power of the DUT.

### 4.3 Artix-7 Experimentation

Xilinx Artix-7 devices were chosen for experimentation based on their reduced cost versus architecturally-similar Virtex-7 devices. These devices are lower power and less costly than their Series 7 family counterparts for the same amount of logic. The five applications used for experimentation were targeted to a Xilinx Artix-7 (XC7A35TICSG324-1L) device using Vivado v2015.3.

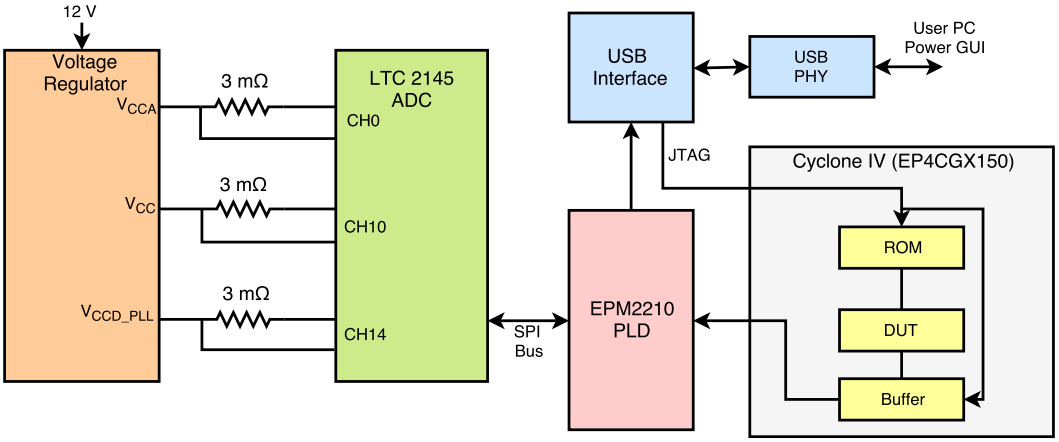


Fig. 6. Current measurement circuitry on the Intel Cyclone IV GX development board

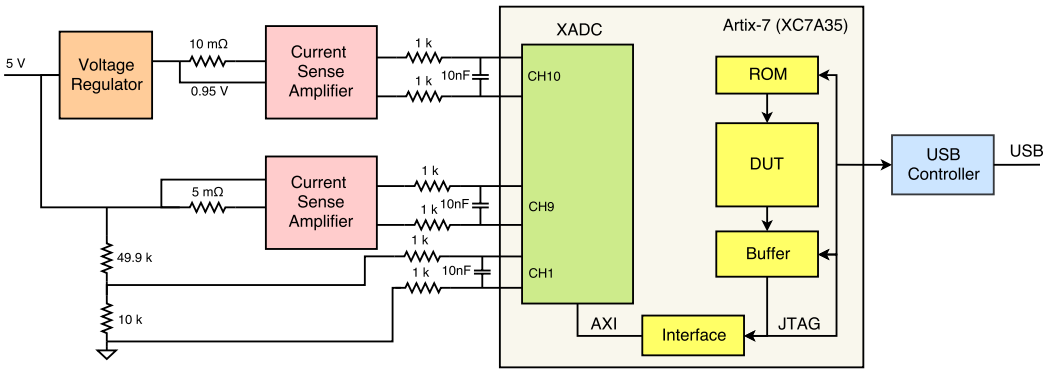


Fig. 7. Current and voltage measurement circuitry on the ARTY Board

The application designs were specified in Verilog HDL and the carry chain delays and glitch filters were inserted as generated macroblocks. Vivado place and route was guided by timing constraints. Compiled designs were evaluated using an Artix-7 development board (ARTY) [31] after bitstream generation.

*4.3.1 Board-Level Power Measurement.* The ARTY board was used to measure power consumed by the Xilinx Artix-7 FPGA in a series of benchtop experiments. The FPGA is equipped with an XADC [7] which has 12-bit, 1 mega sample per second (MSPS) analog-to-digital converter (ADC). The ADC can be connected to 17 analog channels and the converted data is stored in status registers inside the FPGA. On-chip sensors monitor internal supply voltage and die temperature. The XADC interface allows for the monitoring of the supply current to the FPGA internal core (VCCINT) on channel 10 and the overall 5V board supply current and associated voltage on channels 9 and 1, respectively. The circuitry for measuring these power values is shown in Fig. 7. The product of the VCCINT core supply current and the VCCINT core voltage (0.95V) gives the power consumption of the design under test (DUT) and surrounding logic.

| Dyn. energy | Filter Spacing | 1 | 2 | 3 | 5 | 7 | max |
|---|---|---|---|---|---|---|---|
| Artix-7 | | | | | | | |
| SIMON-128 | Measured Energy (pJ/bit) | 43.8 | *34.3* | 41.2 | 54.5 | 64.3 | 240.4 |
| | Area (slices) | 2,344 | 1,436 | 1,378 | 1,374 | 1,363 | 1,133 |
| AES-256 | Measured Energy (pJ/bit) | 57.0 | *54.5* | 145.5 | 256.4 | 347.6 | 584.5 |
| | Area (slices) | 4,628 | 4,338 | 4,335 | 4,356 | 4,288 | 4,258 |
| Bitonic | Measured Energy (pJ/bit) | 10.6 | *9.5* | 10.9 | 13.3 | 17.1 | 30.9 |
| | Area (slices) | 5,481 | 3,792 | 3,744 | 3,841 | 3,439 | 3,074 |
| DES | Measured Energy (pJ/bit) | *24.9* | 27.3 | 62.5 | 71.0 | 73.0 | 73.4 |
| | Area (slices) | 399 | 391 | 390 | 409 | 392 | 264 |
| CORDIC | Measured Energy (pJ/bit) | *8.7* | 9.8 | 14.8 | 14.8 | 15.9 | 26.2 |
| | Area (slices) | 544 | 459 | 400 | 372 | 397 | 271 |
| Cyclone IV | | | | | | | |
| SIMON-128 | Estimated Energy (pJ/bit) | 109.7 | 99.0 | 100.7 | 122.1 | 153.6 | 3,481.8 |
| | Measured Energy (pJ/bit) | 101.3 | *90.0* | 101.3 | 118.1 | 146.3 | 3,420.0 |
| | Area (LABs) | 2,198 | 1,962 | 1,838 | 1,835 | 1,808 | 1,111 |
| AES-256 | Estimated Energy (pJ/bit) | 163.7 | 361.7 | 596.3 | 1,102.8 | 1,411.9 | 4,760.7 |
| | Measured Energy (pJ/bit) | *168.8* | 337.5 | 554.1 | 1,046.3 | 1,333.1 | 3,515.6 |
| | Area (LABs) | 4,714 | 4,514 | 4,395 | 4,331 | 4,342 | 4,338 |
| Bitonic | Estimated Energy (pJ/bit) | 35.2 | 28.1 | 30.4 | 36.5 | 50.4 | 133.2 |
| | Measured Energy (pJ/bit) | 31.9 | *24.7* | 28.4 | 34.1 | 43.8 | 100.5 |
| | Area (LABs) | 3,233 | 3,084 | 2,054 | 1,861 | 1,785 | 1,559 |
| DES | Estimated Energy (pJ/bit) | 136.4 | 233.6 | 536.6 | 575.1 | 540.0 | 340.9 |
| | Measured Energy (pJ/bit) | *135.0* | 229.5 | 492.8 | 533.3 | 513.0 | 337.0 |
| | Area (LABs) | 321 | 304 | 302 | 281 | 270 | 215 |
| CORDIC | Estimated Energy (pJ/bit) | 51.4 | 46.4 | 148.8 | 164.7 | 222.9 | 94.1 |
| | Measured Energy (pJ/bit) | *41.2* | 41.4 | 129.4 | 117.6 | 141.2 | 82.4 |
| | Area (LABs) | 228 | 182 | 165 | 135 | 135 | 124 |

Table 4. Dynamic energy per bit (pJ/bit) and overall application area if glitch filter insertion is performed every $n$ iterations where $n$ is 1 to 7. *Max* indicates no glitch filtering was used. All designs were fully unrolled to generate these results. The filter insertion with the lowest energy for a design is underlined. Estimated values were determined via simulation.

The auxiliary supply (VCCAUX = 1.8V) drives phase-locked loops (PLLs), JTAG, and other circuitry. In our experiments, only sequential designs without any unrolling use PLLs. Although the ARTY board does not include circuitry for measuring auxiliary power, PLL power can be measured indirectly as the difference of the overall 5V power consumed with and without the PLL. All current and voltage values were obtained from the XADC with a JTAG connection to a Vivado Tcl console. FPGA static power is consumed by the core in the absence of design switching activity. It was measured using the JTAG and XADC circuitry to be 30.0 mW in the XC7A35TICSG324-1L. Static power was subtracted from the power values reported in Section 5. The same testbench setup as described in Section 4.2.2 was used for data input generation into the DUT.

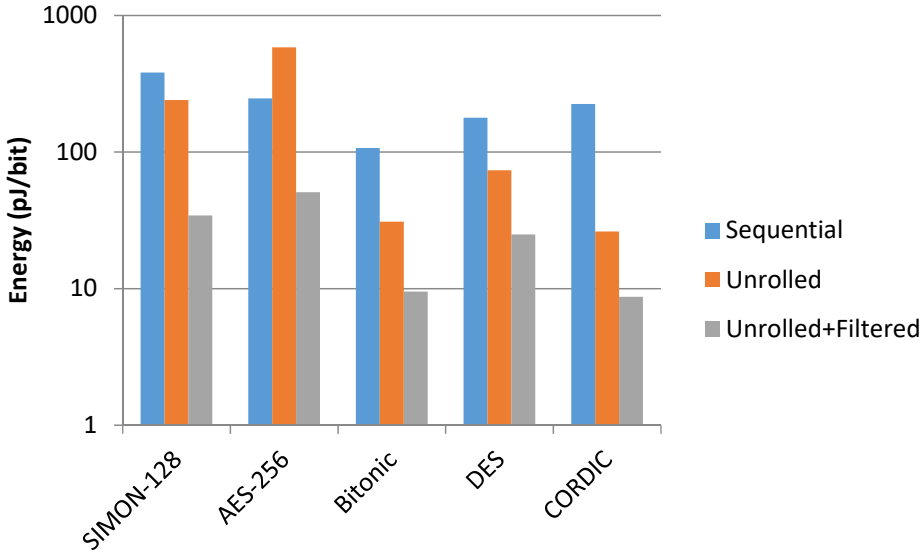


Fig. 8. Energy in pJ/bit for loop-based applications in an Artix-7 FPGA. Each value indicates the amount of energy required to generate one output. The local clock for the sequential version was generated with a PLL.

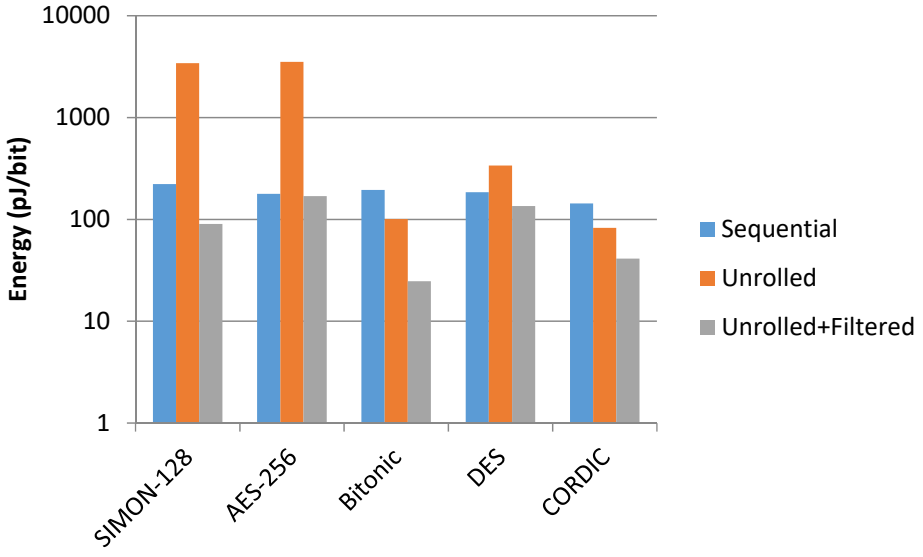


Fig. 9. Energy in pJ/bit for loop-based applications in a Cyclone IV FPGA. Each value indicates the amount of energy required to generate one output. The local clock for the sequential version was generated with a PLL.

## 5 RESULTS

### 5.1 Location of Per-Application Filter Insertion

In our first experiment, we assess the tradeoff between energy reduction and area increase for glitch reduction latch insertion. To obtain this comparison, all application loops (one per application) were fully unrolled. Glitch filters introduced between iterations reduce overall energy but add area and can consume energy themselves. Note that instead of placing latches after the circuitry for every iteration, it might be advantageous to place them after multiple iterations due to this issue. To explore the optimum spacing between glitch filters for each application, we measured energy and area with latches inserted after each iteration and after multiple iterations. Table 4 shows energy consumption with regards to distance of glitch filter insertion. The metric we use for energy consumption for the loop based computations is pJ/bit. This value is determined from dynamic power measurements via the following equation:

$$pJ/bit = \frac{average\ power\ (mW) * L\ (ns)}{Input\ bit\ width} \tag{5}$$

where $L$ is the latency across all loop iterations defined in Table 1. Latency and data input bit widths for the benchmarks are shown in Table 2.

Energy-optimal glitch filter insertion was determined to be after each iteration for AES-256 (Cyclone IV), DES, and CORDIC and after two rounds for AES-256 (Artix-7), bitonic and SIMON-128. For the Artix-7 fully unrolled cases, glitch filter insertion leads to a a 9% slice increase in the best case (AES-256). In the worst case, slice count is doubled (CORDIC), although it should be noted that the CORDIC design is quite small. For the Cyclone IV fully unrolled cases, a best case area increase of 9% (AES-256) and a worst case area increase of 84% (CORDIC) are observed. The reduced overhead for AES is expected since it has fewer, larger rounds.

Table 4 highlights some differences between dynamic energy consumption values measured on the benchtop and values obtained via simulation for Cyclone IV devices. Estimation can overstate the consumed energy and the disparity generally becomes more pronounced as the amount of glitch filtering is reduced (e.g. towards the right of the table) and glitching increases. The scope and amount of these findings are consistent with a previous FPGA study that contrasted dynamic energy consumption estimation versus benchtop measurement. Oliver *et al.* [23] determined that dynamic power estimation errors of up to 63% can occur if significant design glitching is present. The remainder of our dynamic energy results were determined from on-board measurements.

### 5.2 Fully-Unrolled versus Sequential Loop Implementation

We now consider a scenario in which the system clock frequency is slow enough to allow all iterations of the computation to complete within one system clock cycle. Three alternative design styles are considered for completing the computation within one cycle of the system clock: (1) fully unrolled designs complete computation in a single system cycle but consume high energy due to glitch propagation across rounds; (2) fully unrolled designs with latch-based glitch filters to prevent glitch propagation between rounds; (3) a sequential design, which is inherently not prone to glitches, that is executed many times within the system clock cycle by using a fast *local clock* generated from a phase-locked loop.

Figs. 8 and 9 show the energy consumption of the three different single system-cycle implementations of the applications for Artix-7 and Cyclone IV FPGAs, respectively. For the Artix-7 device, the unrolled implementation consumes significant energy but glitch filtering reduces this energy by between 65% and 93%. For the Cyclone IV device, the reduction is between 50% and 97%. The sequential implementation reduces data energy but the PLL used to generate a local clock from the


Fig. 10. SIMON-128, AES-256, and bitonic energy versus area for different implementations in an Artix-7 device. All implementations have the same latency.

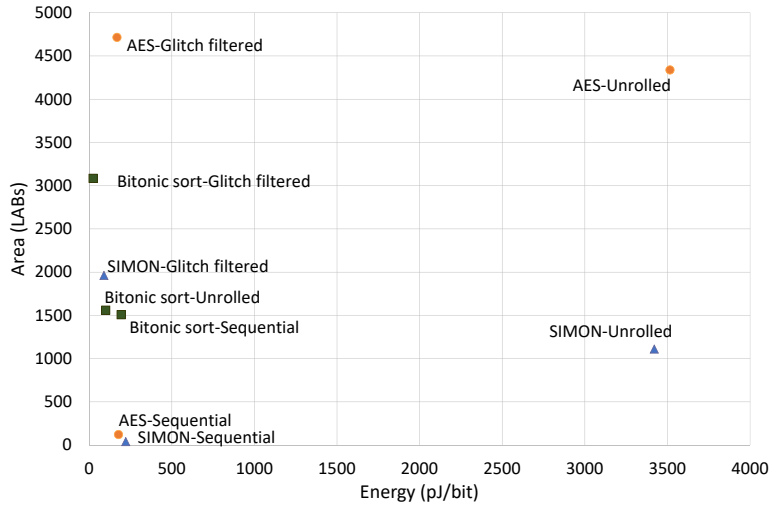


Fig. 11. SIMON-128, AES-256, and bitonic energy versus area for different implementations in a Cyclone IV device. All implementations have the same latency.

system clock makes overall energy higher than the glitch filtering implementation. For example, our on-board measurements indicate that an Artix-7 PLL consumes 83 mW (e.g., 223 pJ/bit for SIMON, 114 pJ/bit for AES-256). Similar on-board measurements for the Cyclone IV device indicate PLL power is about 20 mW (e.g., 92.3 pJ/bit for SIMON-128 and 46.1 pJ/bit for AES-256). It should be noted that Cyclone IV PLLs always consume power even if the PLLs are not used in a design while Artix-7 do not consume power in this condition. Clearly, PLLs add significant power and should be avoided in low energy designs.

Figs. 10 and 11 show the energy versus area tradeoffs of the three implementation choices for SIMON-128, AES-256, and bitonic. For Artix-7, the unrolled, glitch-filtered designs show a clear benefit in dynamic energy compared to sequential implementations. This benefit is enhanced due to the lack of PLL energy needed for filtered designs. For Cyclone IV versions, the energy benefit is much smaller primarily because sequential-version dynamic PLL energy is much lower. The graphs represent the area-energy tradeoffs that can be considered in loop-based implementation.

### 5.3 Partial Unrolling Tradeoffs

Full loop unrolling is only feasible for extremely slow system clocks, and we now evaluate loop implementation at a range of more typical system clock frequencies. As described in Section 3, our software system considers the required latency of an iteration and the system clock frequency as inputs in determining how much unrolling should be performed. Glitch filters are added to the unrolled design to suppress glitches. By unrolling to use the system clock, the insertion of an energy-hungry PLL is avoided. In Table 5, we illustrate energy consumption based on the amount of unrolling $U$, as defined in Table 1. For $U = max$, the loop is fully unrolled and the resulting design is fully combinational, saving clock and flip flop energy. None of these designs include a PLL so the required system clock for designs that are not fully unrolled is provided externally.

Several interesting observations can be seen from examining Table 5. For small designs, such as DES and CORDIC, a faster system clock with a smaller amount of circuitry is generally the best option for area and energy. For larger SIMON-128 and AES-256 designs, full unrolling with glitch filtering generally gives better energy results at the cost of increased area. In almost all cases, the insertion of glitch-filters helps to reduce the energy for partially-unrolled designs. However, for Cyclone IV DES with $U = 2$ and CORDIC with $U = 3$, glitch filtering increases energy consumption since the size of the filter relative to the protected circuit is significant. These results show clear tradeoffs that can be considered during the unrolling process. Finally, it can be seen that fully unrolled and filtered designs often have significantly better dynamic energy results than partially unrolled versions. For designs such as CORDIC, this improvement is primarily a result of the elimination of needed multiplexing circuitry at the input of partially-unrolled versions. For SIMON-128 and AES-256, full unrolling of the encryption circuit also allows for full unrolling of per-round key generation circuitry.

For each system clock frequency, we also consider the energy and area of the unrolled equivalent-latency alternatives versus sequential versions that require a PLL. We show a sampling of these relationships in Figs. 12, 13, 14, and 15. The energy consumption of the PLL at different clock frequencies is shown in the graphs for reference. At fast system clock speeds (left side of figure) only a small number of iterations (rounds) are computed in each system clock period, so glitching is not an issue. If the system clock is slow enough to allow computing multiple rounds within a system clock cycle, the unrolled design without glitch filters becomes inefficient due to glitching. The unrolled design with glitch filters remains efficient across the clock frequencies, but pays an ever-increasing price in area. The sequential design with PLL is the most compact in area, but due to the PLL power consumption it cannot match the efficiency of the glitch-filtered unrolled design unless the system clock is fast enough to support sequential computation. The discontinuity in the unrolled (without filtering) curves results from the lack of repetitive activity in the key generation circuitry for fully-unrolled versus partially-unrolled versions of the two crypto circuits.

Our results show that unrolling without glitch filters incurs a high cost in both area and energy, and is therefore not a justifiable design style. The choice between using unrolling with glitch filters and using a sequential design with a PLL-generated local clock depends on the particular constraints and cost objectives of the design which are considered in our flow. If energy is the

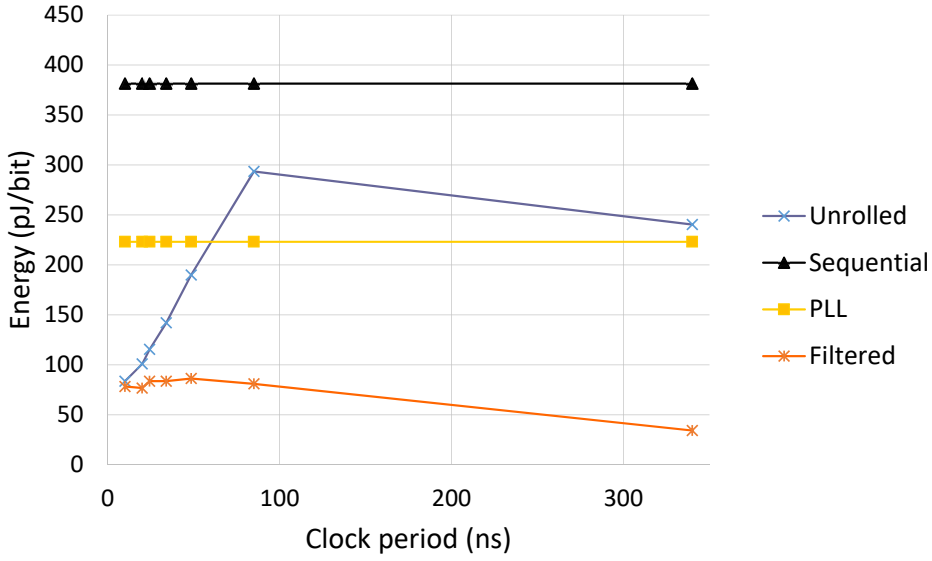


Fig. 12. Energy comparison of SIMON-128 implementations with the same block cipher latency of 340 ns for Artix-7 devices. The fully unrolled data points are on the right for *Unrolled* and *Filtered* and unrolled by a factor of 2 are on the left. Intermediate points indicate partial unrolling. *Filtered* indicates unrolled and filtered.

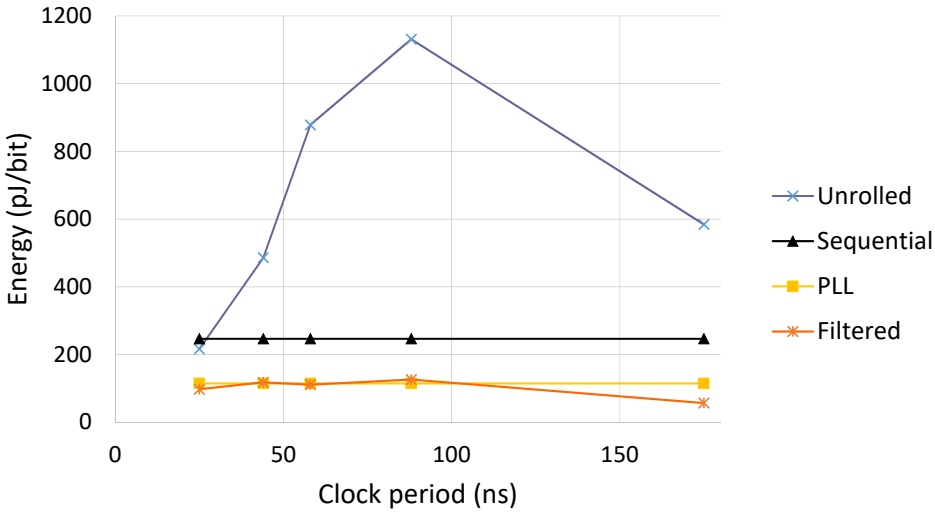


Fig. 13. Energy comparison of AES implementations with the same block cipher latency of 175 ns for Artix-7 devices.

primary objective, then the unrolled design with glitch filters can be the best choice; if reducing area is the primary objective and a PLL is available, then the sequential or partially unrolled design may be preferred.

| **SIMON-128** | | $U$ | 1 | 2 | 5 | 10 | 17 | max |
|---|---|---|---|---|---|---|---|---|
| Artix-7 | Unroll | $E$ measured (pJ/bit) | 158.4 | 83.7 | 115.5 | 189.9 | 293.5 | 240.4 |
| | | Area (slices) | 84 | 155 | 279 | 467 | 811 | 1133 |
| | Unroll-filter | $E$ measured (pJ/bit) | - | 78.4 | 83.7 | 86.3 | 81.0 | 34.3 |
| | | Area (slice) | - | 211 | 329 | 636 | 951 | 1436 |
| | System clock | Frequency (MHz) | 200 | 100 | 41 | 20.5 | 11.7 | 2.9 |
| Cyclone IV | Unroll | $E$ measured (pJ/bit) | 129.4 | 135.0 | 225.0 | 438.8 | 984.4 | 3,420.0 |
| | | Area (LABs) | 40 | 56 | 117 | 276 | 550 | 1111 |
| | Unroll-filter | $E$ measured (pJ/bit) | - | 135.0 | 225.0 | 202.5 | 208.1 | 90.0 |
| | | Area (LABs) | - | 87 | 166 | 319 | 524 | 1962 |
| | System clock | Frequency (MHz) | 114 | 57 | 23.5 | 11.6 | 6.6 | 1.7 |
| **AES-256** | | $U$ | 1 | 2 | 4 | 6 | 8 | max |
| Artix-7 | Unroll | $E$ measured (pJ/bit) | 132.8 | 216.7 | 486.0 | 878.4 | 1131.3 | 584.5 |
| | | Area (slices) | 370 | 1085 | 1832 | 2536 | 3283 | 4258 |
| | Unroll-filter | $E$ measured (pJ/bit) | - | 97.8 | 118.3 | 111.4 | 126.5 | 50.7 |
| | | Area (slices) | - | 1130 | 1948 | 2706 | 3296 | 4628 |
| | System clock | Frequency (MHz) | 80 | 40 | 23 | 17 | 11 | 5.7 |
| Cyclone IV | Unroll | $E$ measured (pJ/bit) | 132.0 | 720.0 | 1929.4 | 3287.8 | 3990.9 | 3515.6 |
| | | Area (LABs) | 121 | 870 | 1609 | 2365 | 2980 | 4338 |
| | Unroll-filter | $E$ measured (pJ/bit) | - | 317.8 | 345.9 | 390.9 | 323.4 | 168.8 |
| | | Area (LABs) | - | 927 | 1734 | 2636 | 3433 | 4714 |
| | System clock | Frequency (MHz) | 46.7 | 23.3 | 13.3 | 10 | 6.6 | 3.3 |
| **DES** | | $U$ | 1 | 2 | 4 | 8 | max | |
| Artix-7 | Unroll | $E$ measured (pJ/bit) | 46.8 | 37.9 | 60.4 | 66.3 | 73.4 | |
| | | Area (slices) | 53 | 81 | 108 | 183 | 264 | |
| | Unroll-filter | $E$ measured (pJ/bit) | - | 26.5 | 38.5 | 69.8 | 24.9 | |
| | | Area (slices) | - | 115 | 194 | 294 | 399 | |
| | System clock | Frequency (MHz) | 160 | 80 | 40 | 20 | 10 | |
| Cyclone IV | Unroll | $E$ measured (pJ/bit) | 74.2 | 87.8 | 121.5 | 249.8 | 337.5 | |
| | | Area (LABs) | 32 | 50 | 81 | 135 | 215 | |
| | Unroll-filter | $E$ measured (pJ/bit) | - | 229.5 | 162.0 | 310.5 | 135.0 | |
| | | Area (LABs) | - | 63 | 108 | 199 | 321 | |
| | System clock | Frequency (MHz) | 44 | 22 | 11 | 5.5 | 2.8 | |
| **CORDIC** | | $U$ | 1 | 3 | 5 | max | | |
| Artix-7 | Unroll | $E$ measured (pJ/bit) | 29.6 | 98.4 | 122.9 | 26.2 | | |
| | | Area (slices) | 55 | 167 | 246 | 271 | | |
| | Unroll-filter | $E$ measured (pJ/bit) | - | 100.6 | 105.1 | 8.7 | | |
| | | Area (slices) | - | 162 | 371 | 544 | | |
| | System clock | Frequency (MHz) | 120 | 40 | 24 | 8 | | |
| Cyclone IV | Unroll | $E$ measured (pJ/bit) | 47.0 | 105.9 | 170.6 | 82.4 | | |
| | | Area (LABs) | 26 | 78 | 125 | 124 | | |
| | Unroll-filter | $E$ measured (pJ/bit) | - | 235.3 | 123.5 | 41.2 | | |
| | | Area (LABs) | - | 95 | 159 | 182 | | |
| | System clock | Frequency (MHz) | 60 | 20 | 12 | 4 | | |

Table 5. Application $E$ (pJ/bit) comparison between optimal glitch filtering and the baseline design for various degrees of unrolling. Design labeled *max* are fully unrolled and are combinational only

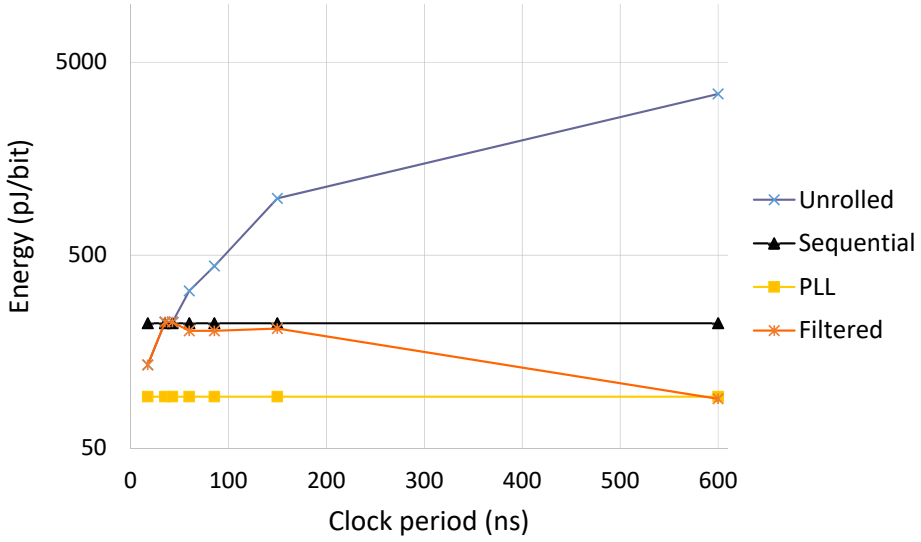


Fig. 14. Energy comparison of SIMON-128 implementations with the same block cipher latency of 600 ns for Cyclone IV devices. The fully unrolled data points are on the right and unrolled by a factor of 2 are on the left. *Filtered* indicates unrolled and filtered.

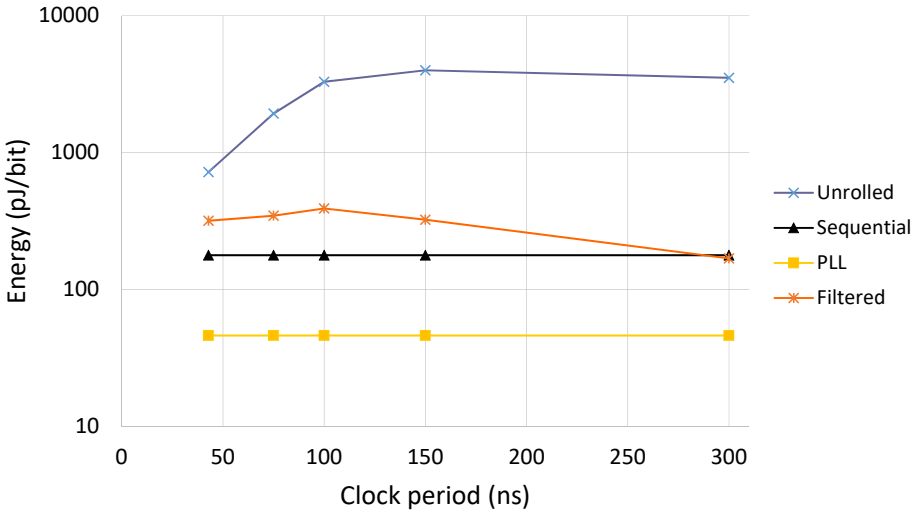


Fig. 15. Energy comparison of AES implementations with the same block cipher latency of 300 ns for Cyclone IV devices.

### 5.4 Unrolled Circuit Integration with FFT Core

For the next experiment we consider the effects of unrolling and glitch filtering when a loop-based circuit is used as part of a larger application. FPGAs often implement fast Fourier transform (FFT) operations to perform frequency analysis of incoming signals. To evaluate our glitch filtering

approach in the presence of other FPGA design components, we integrated the SIMON-128 and AES-256 block ciphers with an FFT application. We obtained the 32-point, 16-bit FFT RTL code from Spiral.net [32]. The FFT core was replicated three times with SIMON to ensure that the entire chip was utilized for experimentation. Only one FFT core was used with AES-256 to achieve full usage of the Artix-7 device.

The experimental setup to measure the power consumed by the FFT circuit was implemented on the ARTY board. A ROM with initialized contents drives new data into the FFT core (or cores) on every clock cycle. The output from the FFT block was given as an input to the block cipher. The block cipher (AES/SIMON) was implemented as fully unrolled with and without filters and sequentially with PLL for each experiment. An encryption key is stored in an initialized constant register inside each block cipher design. The output of the encryption core was connected to a debug port to verify correct circuit operation prior to energy consumption measurement experiments.

Table 6 shows the energy comparison for FFT integrated with three implementations of SIMON-128 and AES-256. Compared to the unfiltered unrolled implementation, the energy savings are about 72% for SIMON and 87% for AES for the unrolled, filtered versions. The sequential version consumes higher energy than the filtered version due to PLL circuitry. Table 6 also shows the area comparison for FFT integrated with the different implementations of SIMON-128 and AES-256. The glitch filters lead to an area overhead of 5.8% in SIMON and 2.0% in AES.

### 5.5 Glitch Filtering with Flip Flops

In Section 3.3, it was noted that level-sensitive latches are superior to edge-triggered flip flops for glitch filtering due to the difficulty of predicting exactly when valid data will arrive. In Tables 7 and 8 it is seen that the energy savings of using flip flops versus latches is limited. In three Cyclone IV cases for flip flops ($U$ = 2, 4 for SIMON-128, $U$ = 2 for AES-256) it was not possible to achieve valid results at the target clock speed using flip flops instead of latches for filtering due to the limitations discussed in Section 3.3. It is possible that more aggressive use of timing constraints on data and trigger paths during FPGA compilation could be attempted to improve the performance of flip flop-based glitch filters. This approach could potentially guarantee data value setup at the filter prior to trigger arrival. Evaluation of the technique is left as future work.

### 5.6 Power Overhead of PLL Use

In a final experiment, we consider the average power consumption of sequential implementations of the benchmarks using a fixed clock speed. In one case, the clock used to sequence the circuit is generated by a PLL and in the other it is input from outside the FPGA. For Artix-7 implementations, the 100 MHz external clock was either fed directly to the sequential circuit or it was used by a PLL to generate an internal 100 MHz clock. The power consumptions of both approaches for the five benchmarks are shown under the Artix-7 columns in Table 9. A similar experiment was performed using the Cyclone IV board. In this case a 50 MHz external clock was used to either directly drive the benchmark or to generate an internal 50 MHz clock using a PLL. The results of these experiments are shown the Cyclone IV columns in Table 9. The results in the table show the significant power overhead of PLL use and the benefit of using the system clock for loop-based computation.

## 6 CONCLUSIONS AND FUTURE WORK

This paper has described a new approach to implement loops in low-cost FPGAs in an energy-efficient manner. The approach unrolls loops to an appropriate depth based on the system clock speed and inserts FPGA-specific filters to suppress glitches. The system clock used for other circuitry in the design can be reused for the unrolled circuitry eliminating the need for an energy-consuming

| Design | Unrolled | | Unrolled+Filtered | | Seq. (with PLL) | |
|---|---|---|---|---|---|---|
| | Slices | pJ/bit | Slices | pJ/bit | Slices | pJ/bit |
| FFT + SIMON-128 | 6,490 | 314.7 | 6,872 | 88.9 | 5,114 | 427.7 |
| FFT + AES-256 | 5,797 | 592.6 | 5,916 | 73.1 | 2,121 | 252.2 |

Table 6. Area and energy comparison for FFT integrated with SIMON-128 and AES-256 in an Artix 7 FPGA

| | $U$ | 2 | 4 | 5 | 7 | 10 | 17 | max |
|---|---|---|---|---|---|---|---|---|
| Artix-7 | FF | 74.0 | 75.7 | 74.0 | 81.8 | 84.2 | 77.0 | 32.1 |
| | latch | 78.4 | 76.8 | 83.7 | 83.7 | 86.3 | 81.0 | 34.3 |
| Cyclone IV | FF | - | - | 180.0 | 191.2 | 208.1 | 202.5 | 90.0 |
| | latch | 135.0 | 225.0 | 225.0 | 202.5 | 202.5 | 208.1 | 90.0 |

Table 7. SIMON-128 comparison of measured energy efficiency (pJ/bit) using glitch filtering with flip flop or latches for differing amounts of unrolling

| | $U$ | 2 | 4 | 6 | 8 | max |
|---|---|---|---|---|---|---|
| Artix-7 | FF | 107.3 | 114.3 | 121.2 | 100.0 | 51.0 |
| | latch | 97.8 | 118.3 | 111.4 | 126.5 | 57.0 |
| Cyclone IV | FF | - | 348.7 | 396.5 | 331.8 | 168.8 |
| | latch | 317.8 | 345.9 | 390.9 | 323.4 | 168.8 |

Table 8. AES-256 comparison of measured energy efficiency (pJ/bit) using glitch filtering with flip flop or latches for differing amounts of unrolling

PLL. Our approach has been tested on five loop-based circuits, including several that were integrated with much-larger cores.

In the future, we plan to explore using unrolling to minimize energy under additional constraints. Unrolling to reduce energy-consuming block memory accesses and static power consumption are likely targets. We will also consider the potential negative effects of using long FPGA delay chains in sub-20nm technology. The generation and use of delay pulses in Fig. 1 could potentially be impacted by *pulse swallowing* for overly long chains or mismatches in rise and fall times along a chain [14]. Our experimentation did not expose this effect for Cyclone IV or Artix 7 devices, although additional research is needed for newer device technologies and longer chains.

## REFERENCES


[1] Altera. 2017. Altera Cyclone IV GX Development Board. https://www.altera.com/products/boards_and_kits/dev-kits/altera/kit-cyclone-iv-gx.html. (2017).

[2] R. Andraka. 1998. A Survey of CORDIC Algorithms for FPGA-based Computers. In *Proc. of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. 191–200.

[3] J. Babb, M. Renard, C. Andras Moritz, W. Lee, M. Frank, R. Barua, and S. Amarasinghe. 1999. Parallelizing applications to silicon. In *Proc. of the IEEE International Symposium on Field-Programmable Custom Computing Machines*. 70–81.

[4] S. Banik, A. Bogdanov, F. Regazzoni, T. Isobe, H. Hiwatari, and T. Akishita. 2016. Round gating for low energy block ciphers. In *Proc. Symp Hardware Oriented Security and Trust*. 55–60.

[5] R. Beaulieu, S. Treatman-Clark, D. Shors, B. Weeks, J. Smith, and L. Wingers. 2015. The SIMON and SPECK lightweight block ciphers. In *Proc. IEEE/ACM Design Automation Conference*. 1–6.

[6] E. Boemo, J. Oliver, and G. Caffarena. 2013. Tracking the Pipelining-Power Rule along the FPGA Technical Literature. In *Proc. of FPGAWorld*. 9:1–9:5.

| Benchmark | Artix-7 @ 100 MHz Dynamic power (mW) | | Cyclone IV @ 50 MHz Dynamic Power (mW) | |
|---|---|---|---|---|
| | no-PLL | PLL | no-PLL | PLL |
| SIMON-128 | 16.0 | 99.0 | 10.8 | 36.5 |
| AES-256 | 102.7 | 180.9 | 57.6 | 84.5 |
| Bitonic | 107.0 | 204.0 | 154.8 | 182.9 |
| DES | 15.0 | 96.0 | 13.2 | 40.1 |
| CORDIC | 9.3 | 86.6 | 6.0 | 32.9 |

Table 9. Power consumption of sequential benchmarks with and without a PLL. Artix-7 and Cyclone IV implementations are clocked at 100 MHz and 50 MHz, respectively


[7] A. Collins. 2011. Agile Mixed Signal Addresses Analog Design Challenges. *White paper, WP398 (v1. 0) August* 15 (2011).

[8] Cyclone IV, Device Handbook. 2010. Vol. 1. *Altera, Dec* (2010).

[9] T. Czajkowski and S. Brown. 2007. Using Negative Edge Triggered FFs to Reduce Glitching Power in FPGA Circuits. In *IEEE/ACM Design Automation Conference*. 324–329.

[10] S. N. Dhanuskodi and D. Holcomb. 2016. Energy Optimization of Unrolled Block Ciphers using Combinational Checkpointing. In *Proc. RFIDSec*.

[11] Q. Dinh, D. Chen, and M. D. F. Wong. 2010. A Routing Approach to Reduce Glitches in Low Power FPGAs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 29, 2 (Feb. 2010), 235–240.

[12] O. Silvia Dragomir, T. Stefanov, and K. Bertels. 2009. Optimal Loop Unrolling and Shifting for Reconfigurable Architectures. *ACM Transactions on Reconfigurable Technology and Systems* 2, 4 (Sept. 2009), 25:1–25:24.

[13] N. K. Dumpala, S. B. Patil, D. E. Holcomb, and R. Tessier. 2017. Energy Efficient Loop Unrolling for Low-Cost FPGAs. In *Proc. of the IEEE Conference on Field-Programmable Custom Computing Machines*. Napa, CA, 17–20.

[14] D. Fick, N. Liu, Z. Foo, M. Fojtik, J. Seo, D. Sylvester, and D. Blaauw. 2010. In Situ Delay-Slack Monitor for High-Performance Processors Using An All-Digital Self-Calibrating 5ps Resolution Time-to-Digital Converter. In *International Solid State Circuits Conference*. 23–25.

[15] H. Hsing. 2015. tiny_aes AES Core. http://opencores.org/project,tiny_aes. (2015).

[16] S. Huda and J. Anderson. 2016. Towards PVT-Tolerant Glitch-Free Operation in FPGAs. In *Proc. of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. 90–99.

[17] K. Kepa, D. Coburn, J. C. Dainty, and F. Morgan. 2008. High Speed Optical Wavefront Sensing with Low Cost FPGAs. *Measurement Science Review* 8, 4 (2008), 87–93.

[18] S. Kerckhof, F. Durvaux, C. Hocquet, D. Bol, and F.-X. Standaert. 2012. Towards Green Cryptography: A Comparison of Lightweight Ciphers from the Energy Viewpoint. In *Proc. Conference on Cryptographic Hardware and Embedded Systems*. 390–407.

[19] J. Lamoureux, G. Lemieux, and S. Wilton. 2008. GlitchLess: Dynamic Power Minimization in FPGAs Through Edge Alignment and Glitch Filtering. *IEEE Transactions on VLSI Systems* 16, 11 (Nov. 2008), 1521–1534.

[20] H. Lim, K. Lee, Y. Cho, and N. Chang. 2005. Flip-flop insertion with shifted-phase clocks for FPGA power reduction. In *IEEE/ACM International Conference on Computer-Aided Design*. 335–342.

[21] E. Musoll and J. Cortadella. 1995. Low-power array multipliers with transition retaining barriers. In *Fifth International Workshop on Power and Timing Modeling*. 227–235.

[22] National Institute of Standards and Technology. 2001. *Advanced Encryption Standard (AES)*. Federal Information Processing Standards Publication FIPS-197.

[23] J. Oliver, J. Pérez, and E. Boemo. 2014. Power Estimations versus Power Measurements in Spartan-6 Devices. In *Southern Conference on Programmable Logic*. 1–5.

[24] J. Park, K. R. S. Shayee, and P. C. Diniz. 2004. Performance and Area Modeling of Complete FPGA Designs in the Presence of Loop Transformations. *IEEE Trans. Comput.* 53, 11 (Nov. 2004), 1420–1435.

[25] C. Ravishankar, J. H. Anderson, and A. Kennings. 2012. FPGA Power Reduction by Guarded Evaluation Considering Logic Architecture. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31, 9 (Aug. 2012), 1305–1318.

[26] N. Rollins. 2007. *Reducing Power in FPGA Designs Through Glitch Reduction*. Ph.D. Dissertation. Brigham Young University.

[27] W. Shum and J. H. Anderson. 2011. FPGA Glitch Power Analysis and Reduction. In *Proc. IEEE/ACM International Symposium on Low-Power Electronics and Design*. 27–32.
[28] R. Usselmann. 2009. DES Core. http://opencores.org/project,des. (2009).
[29] S. Wilton, S. Ang, and W. Luk. 2004. The impact of pipelining on energy per operation in field-programmable gate arrays. In *Proc. of Conference on Field Programmable Logic and Application*. 719–728.
[30] J. Wu. 2010. Several Key Issues on Implementing Delay Line Based TDCs Using FPGAs. *IEEE Transactions on Nuclear Science* 57, 3 (June 2010), 1543–1548.
[31] Xilinx. 2017. Artix-7 35T Arty FPGA Evaluation Kit. http://www.xilinx.com/products/boards-and-kits/arty.html#documentation. (2017).
[32] M. Zuluaga. 2012. Sorting Network IP Generator. http://www.spiral.net/hardware/sort/sort.html. (2012).