2017

Energy Efficient Loop Unrolling for Low-Cost FPGAs

Naveen Kumar Dumpala

University of Massachusetts Amherst

ENERGY EFFICIENT LOOP UNROLLING FOR LOW-COST FPGAS

A Thesis Presented

by

NAVEEN KUMAR DUMPALA

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

September 2017

Electrical and Computer Engineering

ENERGY EFFICIENT LOOP UNROLLING FOR LOW-COST FPGAS

A Thesis Presented

by

NAVEEN KUMAR DUMPALA

Approved as to style and content by:

_____

Russell Tessier, Chair

_____

Daniel Holcomb, Member

_____

Amir Arbabi, Member

_____

Christopher V. Hollot, Department Chair
Electrical and Computer Engineering

ACKNOWLEDGMENTS

I would like to thank Prof. Russell Tessier and Prof. Daniel Holcomb for their guidance on this thesis. I would like to express my gratitude to Prof. Amir Arbabi for his time and for accepting to be on my thesis committee. I would also like to thank Shivukumar Patil for his help while working on the project.

ABSTRACT

ENERGY EFFICIENT LOOP UNROLLING FOR LOW-COST FPGAS

SEPTEMBER 2017

NAVEEN KUMAR DUMPALA

B.E., GITAM COLLEGE OF ENGINEERING, ANDHRA UNIVERSITY

M.S.E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Russell Tessier

Many embedded applications implement block ciphers and sorting and searching algorithms which use multiple loop iterations for computation. These applications often demand low power operation. The power consumption of designs varies with the implementation choices made by designers. The sequential implementation of loop operations consumes minimal area, but latency and clock power are high. Alternatively, loop unrolling causes high glitch power. In this work, we propose a low area overhead approach for unrolling loop iterations that exhibits reduced glitch power. A latch based glitch filter is introduced that reduces the propagation of glitches from one iteration to next. We explore the optimal number of filters to be inserted for different applications that give a good balance between area and power. We also implement partial unrolling with glitch filters. This approach consumes less area while still giving energy savings comparable to the fully unrolled implementation.

Our approach is targeted to Xilinx and Altera FPGAs. We simulate different implementation choices and compare energy results to evaluate the savings. We

demonstrate our approach on SIMON-128 and AES-256 block ciphers and a sorting algorithm. We prototype our design on Xilinx Artix-7 and Altera Cyclone-IV-GX FPGA development boards and measure the actual power savings. Results show up-to 90% dynamic energy reduction in Xilinx designs, and 97% reduction in Altera designs with our glitch filtering approach due to glitch power reduction.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The growth of battery operated devices has increased demand for embedded system implementations that use low power semiconductor devices. Application specific integrated circuits (ASICs) are good for low power design, but changing industry standards and time to market pressures for the products limit their use for many products. Field programmable gate arrays (FPGAs) are well suited for embedded applications with shorter design cycles as they are reprogrammable.

Embedded systems that make use of FPGAs implement cryptographic algorithms like block ciphers for security applications [22] and sorting and search algorithms for signal processing [24] and data analysis applications [23]. These algorithms use multiple iterations for computing output values. If these functions are implemented sequentially, i.e. the output of one iteration is fed to next, it can be hard to meet latency requirements and energy consumption will be high. The latency issue can be addressed by implementing multiple iterations in a single cycle, a process known as loop unrolling [16]. There are two kinds of loop unrolling. In a fully unrolled implementation all the iterations are replicated in hardware and executed in a single clock cycle. In the partially unrolled implementation, only a few of the iterations are unrolled and multiple clock cycles are required.

Unrolling a design reduces latency but increases glitch power. Glitches are unwanted transitions caused by unbalanced delays in the signals at the input of combinational logic. In an unrolled design, glitches generated in one iteration propagate through successive iterations causing more glitches and increasing dynamic power

consumption [4]. This effect is particularly visible in designs requiring a high number of iterations. For example, SIMON-128 [6] needs 68 rounds for computation. The increase in power consumption in a Xilinx Artix-7 device due to glitch propagation as unrolling increases is shown in Figure 1.1.
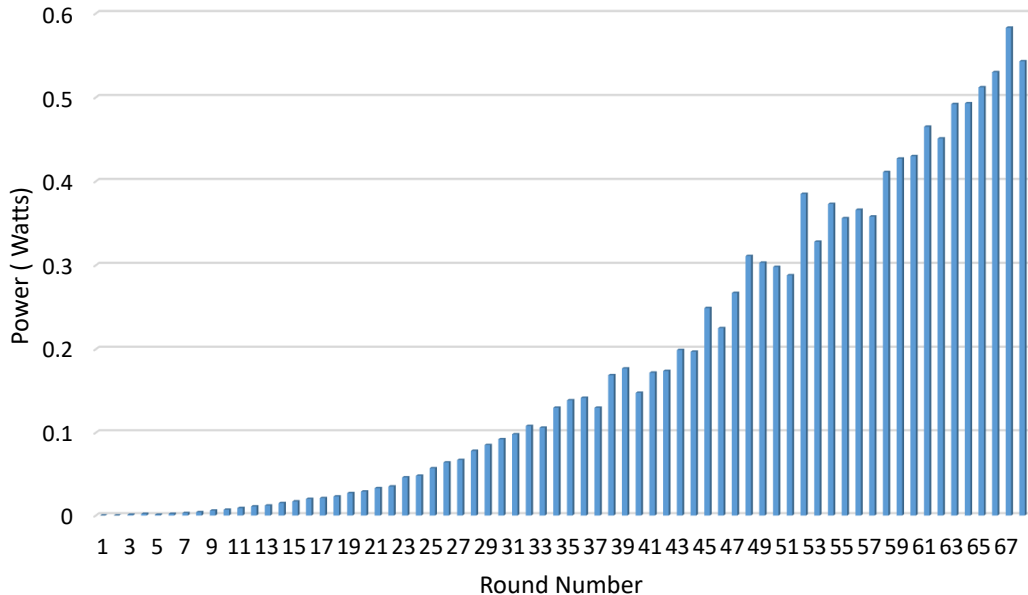
Figure 1.1: Power consumption trends for SIMON-128 block cipher for increasing round counts in Xilinx Artix 7 FPGA, using simulation based estimation

We propose an approach to efficiently unroll designs to reduce dynamic energy consumption. Our approach inserts filters between the unrolled iterations such that glitches generated in the current iteration do not propagate to the next. We use FPGA latches as glitch filters. These latches are enabled only when the current iteration is complete and its output is stable. The latch enable signals are generated using triggers with predictable delays created from the FPGA logic resources.

The specific contributions of this thesis are

- The design of a latch based glitch filter that reduces glitch energy in unrolled loop implementations in FPGAs is presented in Chapter 3.

- The generation of signals with predictable delays in the FPGA using FPGA logic resources is presented in Chapter 3. A range of delay values is supported.

- The deployment of prototype applications of unrolled loops on Artix-7 and Cyclone-IV-GX based boards, found in Sectons 3.4.3 and 3.4.4.

- The determination of the optimal placement of glitch filters in unrolled FPGA loop iterations is presented in Secton 4.2. The unrolling that gives the best trade-off between energy and area is considered.

- An evaluation of partial design unrolling to compare the area and energy numbers of loop-based designs with fully unrolled and serialized implementations is presented in Section 4.4.

- An assessment of our glitch filtering scheme integrated in a real application is presented in Secton 5.

1.1 Thesis Outline

The remainder of this thesis document is organized as follows. Section 2 presents prior work on reducing glitches in FPGAs. Section 3 describes the implementation details and our experimental methodology. Results and analysis are presented in Section 4. An embedded application which uses our glitch filtering approach is detailed in Section 5. Section 6 summarizes the results and presents our conclusions.

CHAPTER 2

BACKGROUND

Block ciphers and sorting and searching algorithms are often implemented in low cost embedded applications [22] [23] [24]. These algorithms use the same function iteratively for computation. For example, the AES 256 [21] block cipher implements 14 rounds (iterations) for the computation of cipher text. Each round takes two inputs, a data value and a key. The round input data is obtained from the previous round output or from the module input port. The key is derived from another iterative function that generates a different key for each round. The round function can be implemented by reusing combinational logic (rounding hardware) or by instantiating multiple copies of a round function and connecting them in sequence. The sequential approach [11] [12] [26] (Figure 2.1) uses less hardware but has high latency and slow clock frequencies. Using multiple copies of the round function [11] [12] [19] i.e. unrolling the design (Figure 2.2) increases the hardware but minimizes the latency to compute the output in terms of clock cycles. For AES-256, a sequential implementation takes fourteen cycles while a fully-unrolled design takes one cycle for computation.

Several prior works have implemented unrolling for reducing the latency in the cryptographic and other iterative designs. In [3], unrolling was adapted to implement CORDIC algorithm on FPGAs. CORDIC algorithm is handy in implementing trigonometric, hyperbolic, linear and logarithmic functions. The works [11] [12] explored multiple architecture options for implementing AES candidate final algorithms and compared area and performance results. Sequential, full and partial loop unrolling

4
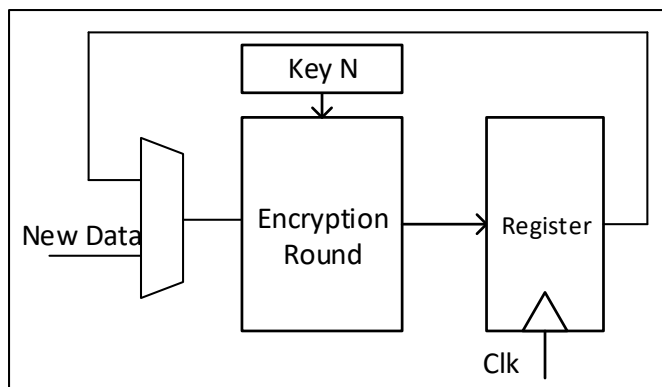
Figure 2.1: Sequential block cipher implementation

schemes are evaluated in terms of latency and throughput. Although these papers analyzed the area and frequency, they lacked the power analysis. We do comprehensive analysis on area, frequency and power with different architectural options for iterative functions.

A sequential loop implementation spends energy storing round function output state in registers. Unrolling saves this energy as the round output directly drives the next round and intermediate state is not stored. Although unrolling saves register energy, it increases glitch energy [5]. Glitches occur due to the difference in input signal arrival times of combinational logic. The input arriving first at combinational encryption round logic can create an output transition and the input arriving next can cause an additional transition. These transitions pass through multiple round functions causing even more transitions at succeeding round outputs [4]. These unwanted transitions unnecessarily increase the design dynamic power consumption.

Several techniques have been developed to address glitch filtering in FPGA designs. The *Glitchless* approach [17] adds programmable delay elements to FPGA logic blocks. Delay elements are added at the inputs and/or outputs of the basic logic elements and are programmed to align the delays at look-up table (LUT) in-
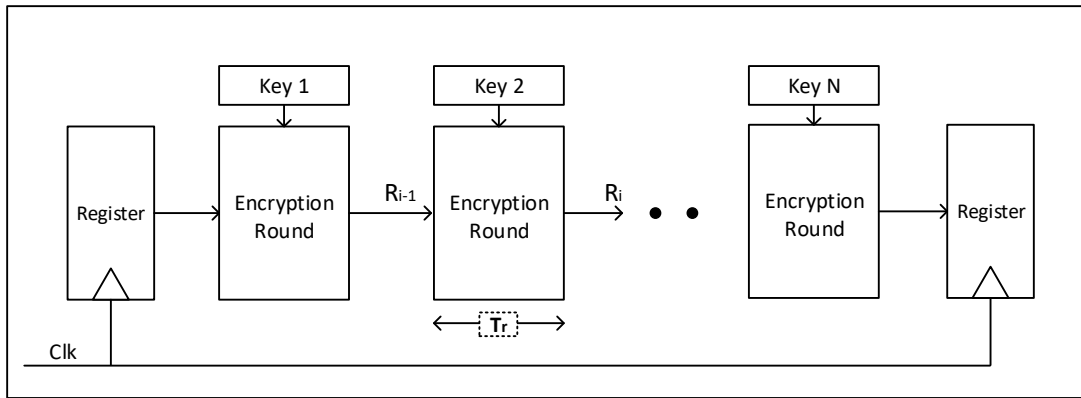
Figure 2.2: Unrolled block cipher implementation

puts. Computer-aided design (CAD) algorithms determine the configuration of these programmable delay elements. Architectural changes are needed in this approach as current FPGA architectures don't have programmable delay elements in logic elements. The modified delay element is shown in Figure 2.3b. The narrow rectangles are the filters. The unmodified design appears in (a).

A recent paper [14] proposes adding adjustable delay circuitry at the output buffers of the basic logic element. Any pulse shorter than a threshold is suppressed by this circuitry. The architecture of the logic element after adding the extra circuitry is shown in Figure 2.4. This approach also requires CAD modifications to set the pulse width threshold for the glitch filter circuits.

Don't care conditions [25] of input signals can be used to reduce output signal transitions. This technique simulates glitch behavior and sets don't care values to either zero or one based on signal activity. This technique is used after the placement and routing stage of the FPGA CAD flow. Although the above approaches reduce glitches, they need significant architectural improvements to the FPGA devices or changes to the FPGA CAD tools.
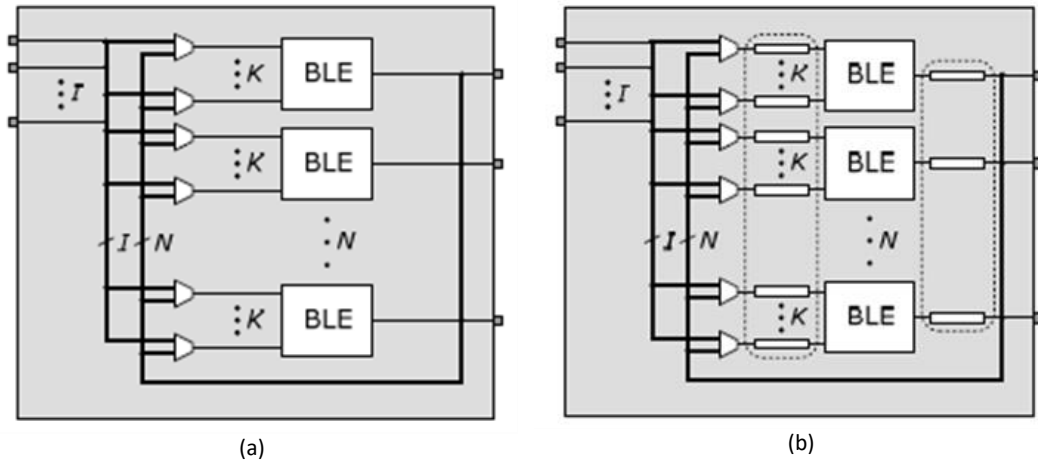
Figure 2.3: Glitch elimination using *Glitchless* a) Original logic block b) Delay insertion at LUT inputs and outputs [17]
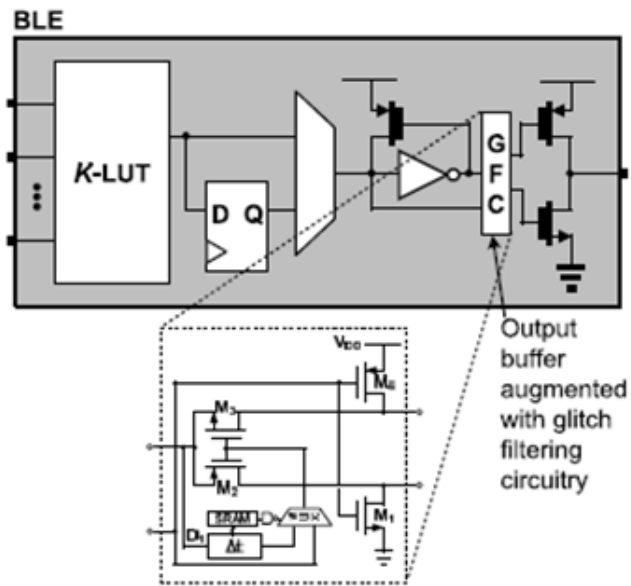
Figure 2.4: Glitch reduction using additional circuitry in a LUT [14]

Another FPGA glitch filtering approach [9] uses negative triggered flip-flops inserted between data path logic to filter glitches. This approach needs extra clock, phase shifted by 180-degrees generated from the FPGA phase locked loop. Although this technique doesn't change the system behavior, it is ineffective for the unrolled designs which have high glitching activity.

The technique proposed in [18] uses phase shifted clocks timed according to LUT delays to drive flip-flops inserted within combinational logic. Multiple phase shifted clocks are generated using the Digital Clock Manager (DCM) and routed using the clock buffers. As the number of dedicated clock lines in the FPGA is limited, only few phase shifted clocks can be generated. Flip-flop and clock assignment is done using an optimization algorithm which maximizes the glitch power savings.

The limitations of this approach are: a) constraint on the number of clock lines in FPGA devices; b) resolution of phase shifted clocks; c) number of DCMs/PLLs on the FPGA and the output clocks per PLL are limited; d) PLL blocks increase power, which works against our goal of power reduction;

Combinational *checkpointing* proposed in [10] uses latch based glitch filters to reduce glitch propagation in unrolled block ciphers. This technique is targeted to application specific integrated circuits (ASICs) and cannot be directly applied to FPGAs. The use of latches and precise delays available in ASICs is not straightforward in FPGAs as the FPGA routing network is fixed. Our approach is similar to checkpointing. We address the challenges of latch and delay implementation in FPGAs.

CHAPTER 3

IMPLEMENTATION AND APPROACH

3.1   Glitch Filter Operation

We implemented glitch filtering on AES-256 and SIMON-128 block ciphers and a bitonic sort algorithm. Figure 3.1 [10] shows glitch filter approach for unrolled block ciphers.
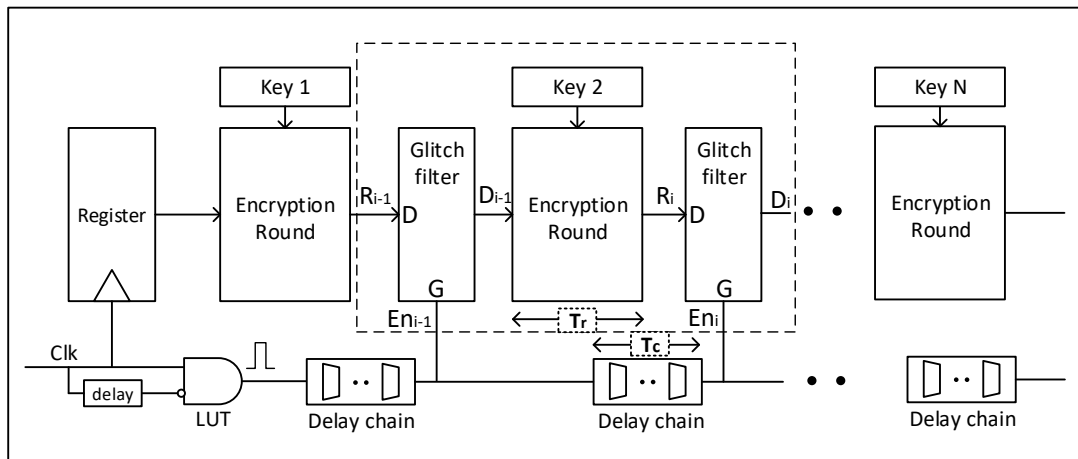
Figure 3.1: Unrolled block cipher with latch-based glitch filtering [10]

The block cipher input is provided from a launch register from where it goes through one or more encryption rounds. The round output is latched using a delayed enable (G) signal. The enable pulse to the glitch filter is generated by ANDing an inverted, delayed version of the input clock with itself. This pulse propagates through carry chain or LUT based delay elements and connects to the enable inputs of glitch filters. To effectively filter glitches, the propagation delay of carry chains or LUTs

(T_c) should be greater than the round delay (T_r). This delay ensures that latches are enabled only after glitches at the input of the latch have settled.

A timing waveform that illustrates the glitch filtering approach for one round of computation is shown in Figure 3.2. The input to the first latch R_{i-1} is sampled using enable En_{i-1} and produces D_{i-1} at the output with 69 transitions. Round computation (R_i) introduces glitches and increases the number of transitions to 302. A latch enabled by En_i filters glitches and reduces the number of transitions to 63.
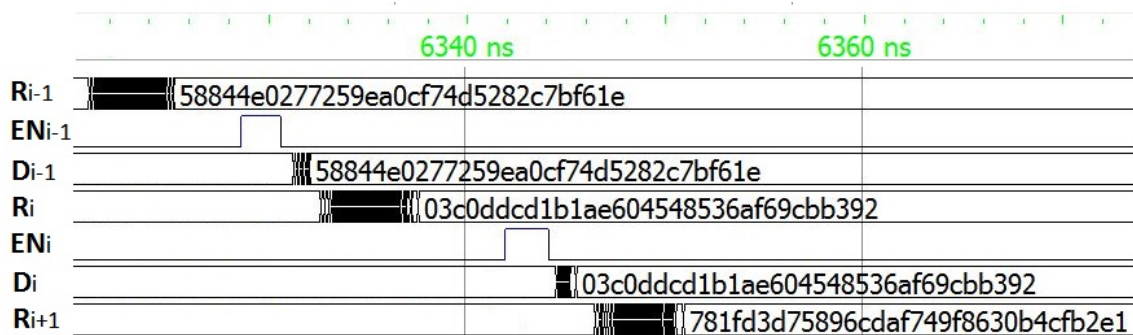
Figure 3.2: Timing waveform after the insertion of glitch filters

3.2 Glitch Filter Implementation

Glitch filters stop the propagation of glitches between iterations by sampling the stabilized data. These filters can be either flip-flop based or latch based. The significant difference between above mentioned implementations is latch based filters have a longer time window to capture the incoming data than flip-flop based filters. If the glitching activity is not stable before the arrival of the active edge of an enable pulse, flip-flop based filters propagate incorrect data to next iterations or enter a metastable state (Figure 3.3 shows incorrect data propagation). But with latch based filters, all the changes in data before the enable level transitions to zero get captured (Figure 3.4).
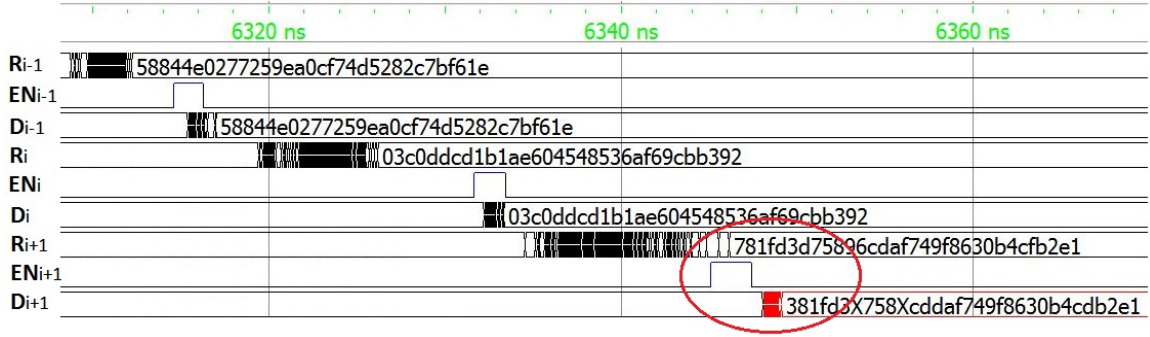
Figure 3.3: Flip-flop based glitch filter timing waveform
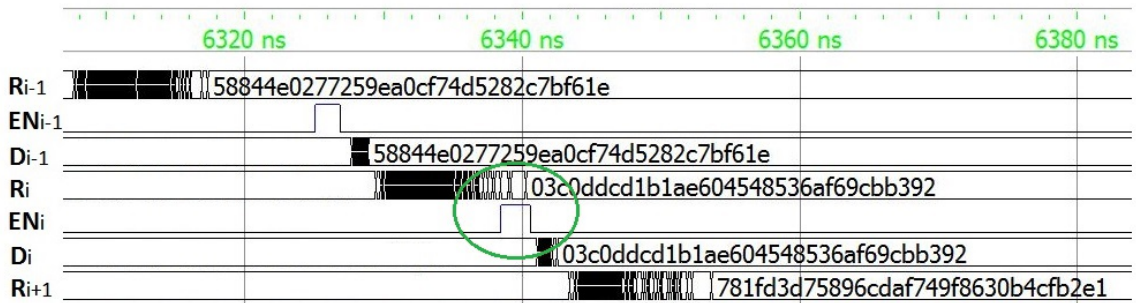
Figure 3.4: Latch based glitch filter timing waveform

In most of our experiments, glitch filter are implemented using the transparent D-latch primitive available in Xilinx Artix-7 and Altera Cyclone IV FPGAs. Since a latch structures are not available in the Altera devices, the latch-based glitch filter can be implemented using a LUT connected as a combinational loop.

3.3   FPGA hardware

The key issues to consider for FPGA glitch filter implementation include determining how long enable signals should be delayed and generating the circuits to create accurate delays. Round delay can be computed by synthesizing the design for the target FPGA device and measuring the associated propagation delays using a timing analysis tool. For generation of accurate delays, we use logic resources available in the FPGA. By cascading the carry chain multiplexers of slices in Xilinx devices or

cascading LUTs in Altera devices, we can generate multiple tap delays. The delay generation circuit is parameterized to generate the precise delays required for an application. Our approach for delay chain implementation is described in Sections 3.3.1 and 3.3.3.

3.3.1 Delay chain implementation in Xilinx FPGAs

In Xilinx devices, carry chains [20] are provided to perform fast arithmetic operations. As block ciphers do not use arithmetic circuitry, the carry chains can be repurposed for delay generation. The organization of a carry-chain-based delay chain is shown in Figure 3.5.

Each slice is equipped with four carry multiplexers (MUXCY) which can be cascaded into the carry chain of next vertical slice. An enable pulse is given to the first MUXCY and an output is taken from the third MUXCY. The MUXCY select lines are connected to logic high to allow the propagation of the enable pulse through the carry chains. The carry chains generate fine grain propagation delay. Multiple slices are connected to generate delays greater than the encryption round. The output from the previous slice is connected to the glitch filter latch and the next delay element. Delay through a single slice carry chain (CYCINIT to CO[2]) is a nearly constant 780 ps (in Xilinx Artix 7 xc7a35ticsg324-1L) but routing delay varies depending on the placement of slices. To achieve uniform delays in routing, the propagation delay from the input to the output of the carry chain is tightly constrained. This approach results in predictable routing between slices.

3.3.2 Latch implementation in Xilinx FPGAs

A glitch filter is implemented using the latches available in the Artix 7 FPGA. Each slice has eight storage elements of which four can be configured as a latch. The latches are level sensitive and are transparent only when their enable inputs are high. The Xilinx Vivado synthesis tool automatically infers LDCE latch modules during
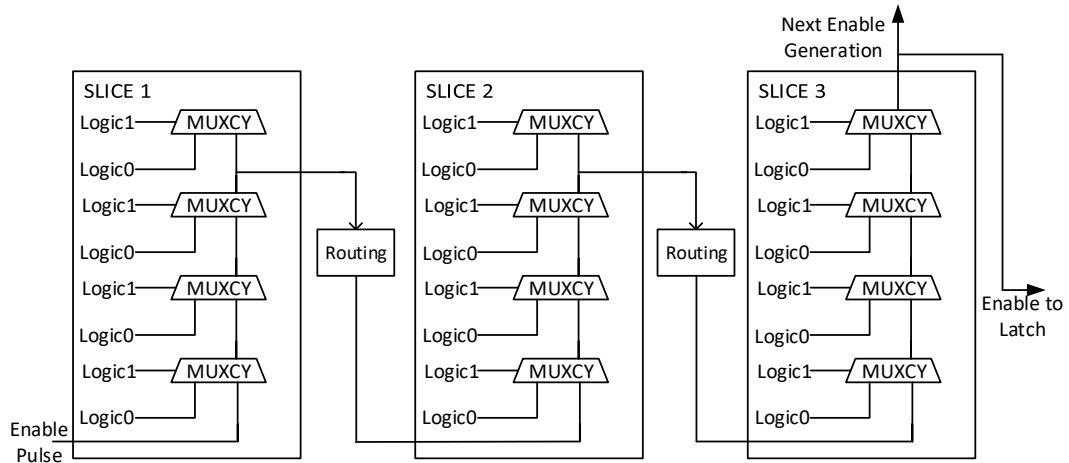
Figure 3.5: Delay chain implementation in a Xilinx Artix-7 FPGA [20]

synthesis. An Artix 7 flip flop configured as a latch is shown in Figure 3.6. The placement of carry chains and latches in a Xilinx Artix 7 chip is shown in Figure 3.7. We can observe that the fanout from carry chain drives 128 glitch filtering latches and next carry logic.
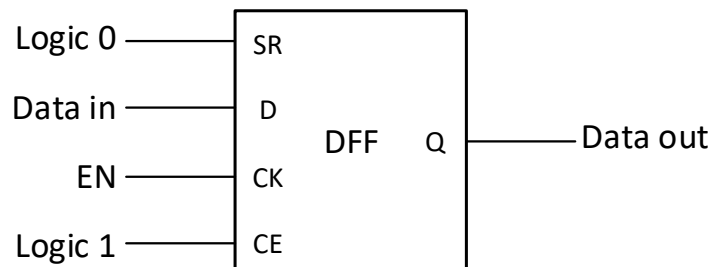
Figure 3.6: Latch implementation in a Xilinx Artix-7 FPGA [20]

3.3.3 Delay chain implementation in Altera FPGAs

In Altera 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 a flip-flop. Adjacent LEs in a LAB can be connected input-

13
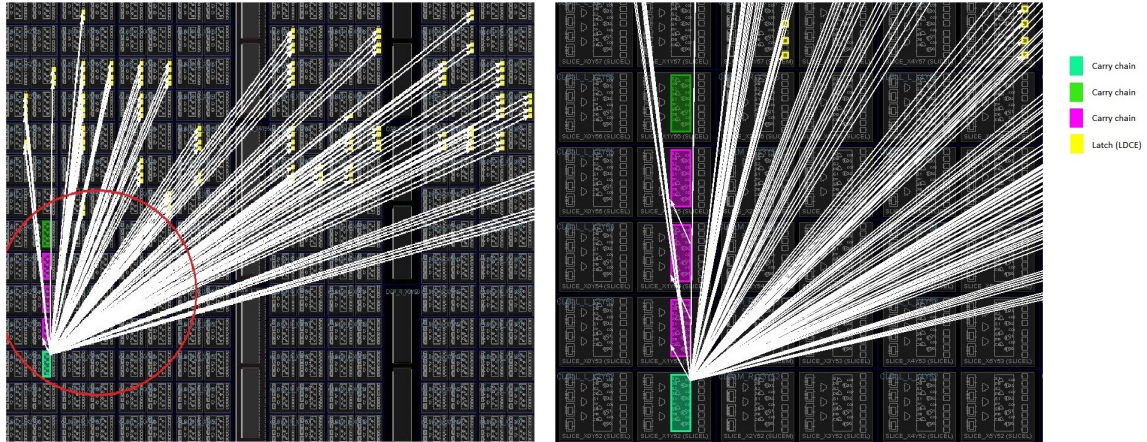
Figure 3.7: Latch and carry chain placement in an Artix 7 (xc7a35ticsg324-1L) FPGA

to-output to generate a delay greater than round delay. Flip-flops in the LE can be used for user logic. The organization of the Altera delay element is shown in Figure 3.8. The delay through a single LUT was determined via simulation to be 155 ps and the average routing delay between LUTs with in a LAB is 200 ps for Cyclone IV EP4CGX150DF21-C7 FPGA.

3.3.4 Latch implementation in Altera FPGAs

As level-sensitive latches are not directly available in flip flop structures in most Altera devices including the Cyclone IV, latches for our work were implemented using LUTs configured in a combinational loop. Logic element LUTs were used (Figure 3.9). The placement of LUTs and latches in an Altera Cyclone IV chip is shown in Figure 3.10. We can observe that a LUT based delay chain drives a CLKCTRL block (global routing) which in turn drives 128 glitch filtering latches.
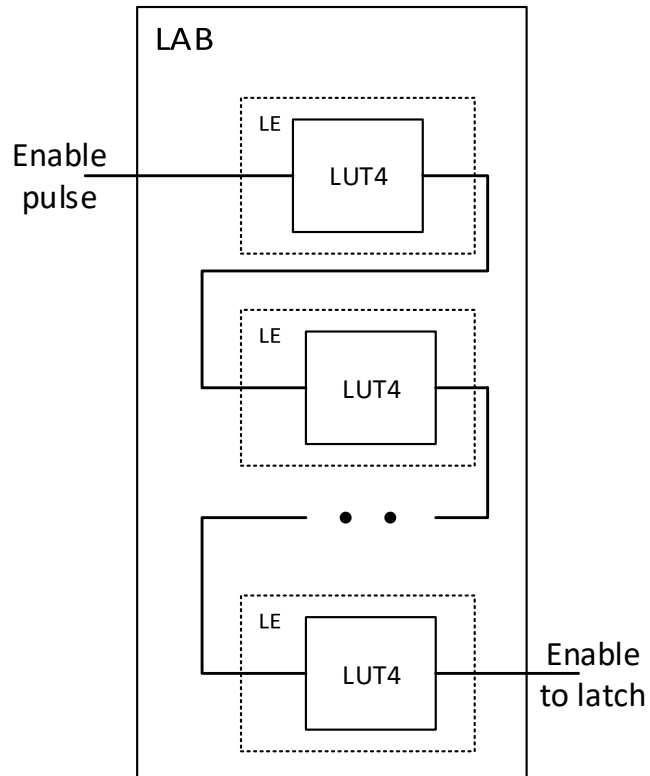
Figure 3.8: Delay chain implementation in an Altera Cyclone IV FPGA [8]
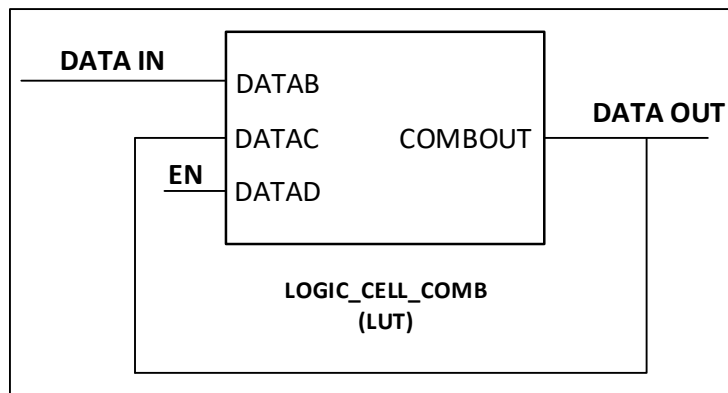
Figure 3.9: Latch implementation in an Altera Cyclone IV FPGA [8]
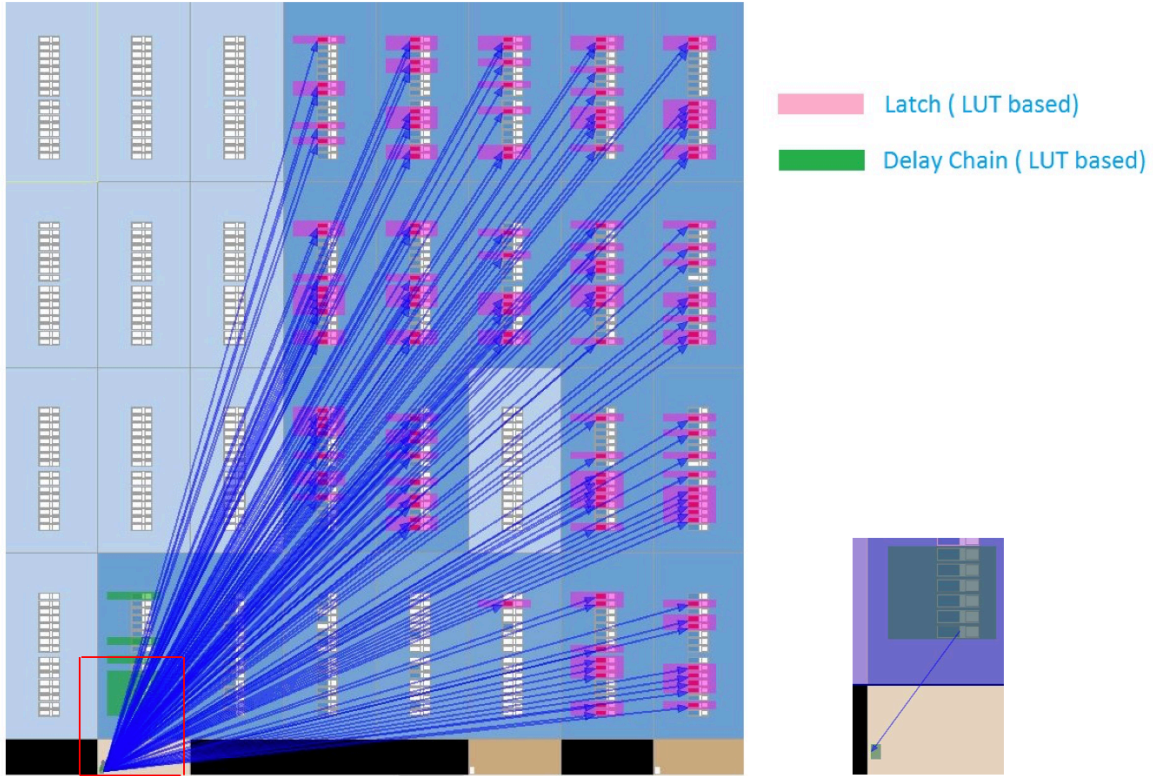
Figure 3.10: Latch and Delay chain placement in a Cyclone IV FPGA

3.4   Experimental Methodology

We implemented our unrolling approach for AES-256 and SIMON-128 block ciphers and bitonic sort algorithm on Xilinx and Altera devices. AES-256 uses 128 bit data, a 256 bit key and 14 rounds (iterations). SIMON-128 uses 128 bit data and a key with 68 encryption rounds. We obtained AES-256 RTL from Opencores.org [13]. A Verilog version of SIMON-128 RTL was written from scratch. Testbenches were created for both designs and the generated outputs were compared to known correct values generated from software. We got the unrolled and sequential versions of the bitonic sort algorithm from Spiral.net [28]. The algorithm has 15 stages of swapping logic and is capable of sorting 32 numbers of 16 bit width. In Xilinx Artix 7 devices, we used seven slice delays per round for AES-256 , three slice delays per round for SIMON-128 and seven slice delays per stage for a bitonic sort. In Altera Cyclone IV

```
                    ┌─────────────────────────┐
                    │  Unrolled design specified │
                    │  in Verilog HDL/VHDL     │
                    │  (.v/.vhd)               │
                    └───────────┬─────────────┘
                                │
                                ▼
                    ┌─────────────────────────┐
                    │  Insert Xilinx specific glitch │
                    │  filter and delay circuitry │
                    └───────────┬─────────────┘
                                │
┌──────────────────────┐        ▼
│  Timing and Placement │──────▶┌─────────────────────────┐
│  Constraints(.xdc)    │        │  Xilinx Vivado          │
└──────────────────────┘        │  Synthesis , Place &Route │
                                └───────────┬─────────────┘
                                            │
                                            ▼
                    ┌─────────────────────┐      ┌─────────────────────┐
                    │  Gate level netlist and │──▶│  Timing simulation  │
                    │  delays in standard delay │   │  performed using Vivado │
                    │  format (.sdf)          │    │  simulator          │
                    └───────────┬─────────┘      └──────────┬──────────┘
                                │                           │
                                ▼                           ▼
┌──────────────────────┐  ┌─────────────────────┐  ┌─────────────────────┐
│  Constraints and Operating │─▶│  Vivado power analyzer │◀─│  Switch activity Information │
│  conditions(voltage/ │    │                     │    │  file (.saif) with signal │
│  Temperature)        │    │                     │    │  activity           │
└──────────────────────┘  └───────────┬─────────┘  └─────────────────────┘
                                      │
                                      ▼
                          ┌─────────────────────┐
                          │  Power Analysis Report │
                          └───────────┬─────────┘
                                      │
                                      ▼
                          ┌─────────────────────┐
                          │  Integrate with simulation │
                          │  time to get Energy in pJ/bit │
                          └─────────────────────┘
```

Figure 3.11: CAD flow in Xilinx devices for simulation

devices, we used 36 LUT delays per round for AES-256, 18 LUT delays per round for SIMON-128 and 36 LUT delays per stage for a bitonic sort.

3.4.1 CAD flow for Xilinx Artix 7 devices for simulation-based power estimation

We first performed simulation-based experiments using a Xilinx Artix 7 (xc7a35ticsg324-1L) device model. Artix 7 FPGAs are of low cost and consume the least power among Xilinx 7 series FPGA family devices. The CAD flow for power measurement using our approach is shown in Figure 3.11. As a first step, an unrolled version of the RTL design with glitch filters inserted is synthesized using Vivado.

17

The synthesized netlist is constrained during synthesis, place and route by timing constraints. Gate level simulation is performed on the routed netlist, which captures the glitching behavior of the design. Signal switching activity is recorded in a signal activity interchange format (SAIF) file. The Xilinx Vivado power analyzer is provided with the gate level netlist, SAIF file, constraints and operating conditions to generate power consumption report. Power values are integrated over time to generate the energy per bit of encryption.

3.4.2 CAD flow for Altera Cyclone IV devices for simulation-based experimentation

For Altera devices, we performed experiments using a model of an Altera Cyclone-IV-GX (EP4CGX150DF21C7) device. Cyclone IV-GX FPGAs are of low cost and suitable for low power applications. The CAD flow for power and energy measurement using our approach is shown in Figure 3.12. The unrolled RTL code with glitch filters inserted is synthesized using Altera Quartus Prime. The synthesized netlist is constrained during Analysis and Fitter stages by timing constraints (.sdc). Signal switching activity is recorded in a value change dump (VCD) file by performing gate level simulations using ModelSim Altera simulator. The Altera PowerPlay power analyzer is provided with the gate level netlist, VCD file, constraints and operating conditions to generate power consumption report.

3.4.3 Power measurement on the Xilinx Artix 7 ARTY board

An Artix-7 development board (ARTY) [27] was used to measure power consumed by the Xilinx FPGA in a series of benchtop experiments. The Arty board includes an Artix 7 FPGA (xc7a35ticsg324-1L) which contains 5200 slices. Artix-7 devices A35T and A50T uses the same silicon die, which allows us to use all the available slices in a 50T device (8150 slices). This board includes circuitry for monitoring current consumed by the FPGA core and for monitoring the 5V board level supply voltage
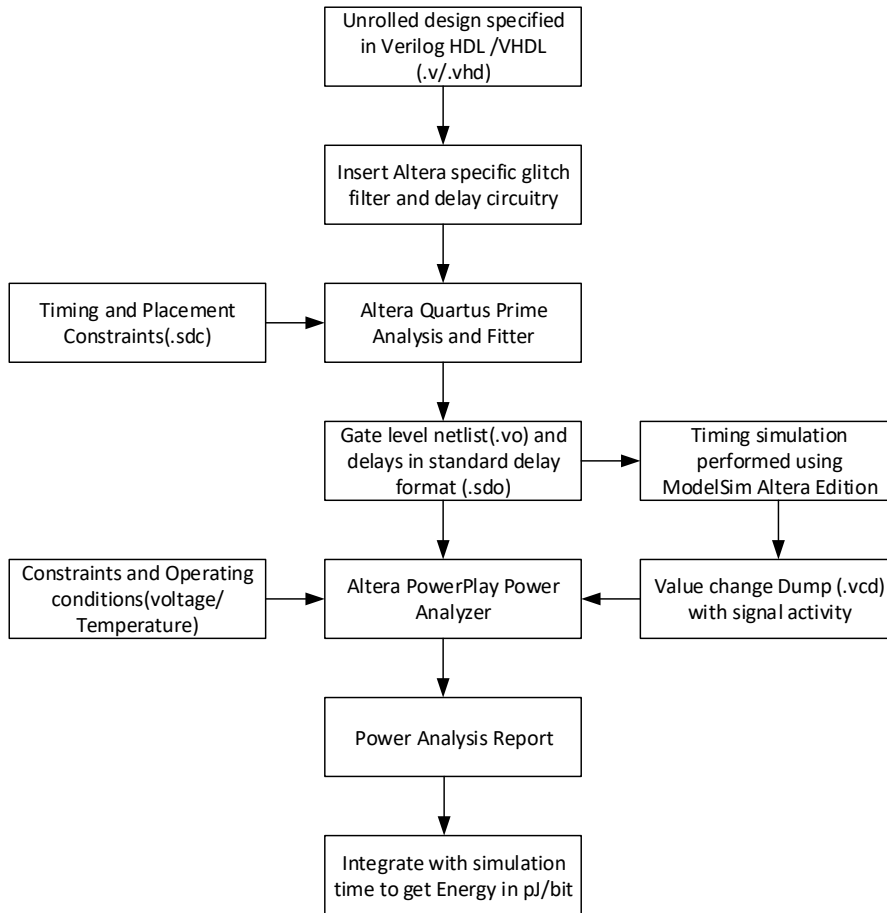
Figure 3.12: CAD flow for Altera devices under simulation

and associated current. Xilinx Arty 7 development board is shown in Figure 3.13. The supply voltages we consider for dynamic power measurement experiments are listed in Table 3.1.

The Xilinx Artix-7 FPGA is equipped with an XADC [7] which has 12-bit, 1 Mega sample per second (MSPS) analog-to-digital converter (ADC) and on-chip sensors. The ADC can be connected to 17 analog channels and the converted data is stored in status registers. On-chip sensors monitor supply voltage and die temperature. The XADC interface allows monitoring supply current to the FPGA core. The circuitry for measuring power in an Artix 7 device on the ARTY development board is shown
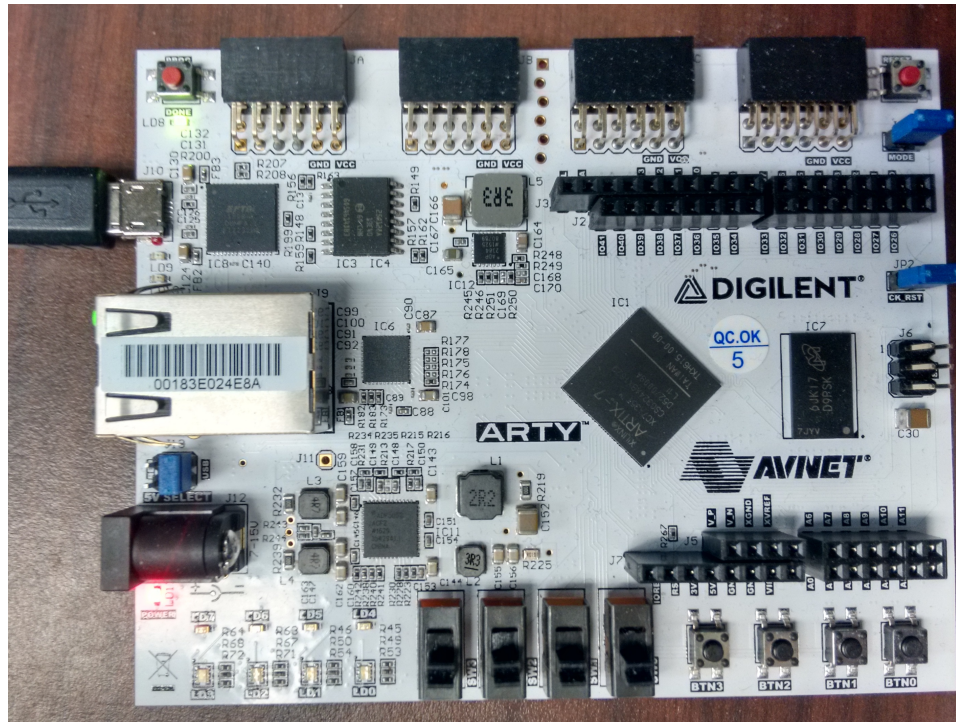
Figure 3.13: Xilinx Artix 7 ARTY Board [27]

Table 3.1: ARTY board supply voltages

Supply	Voltage (V)	Description
VCCINT	0.95	FPGA Core power
VCCAUX	1.8	PLL and JTAG power
VCCADC	1.8	XADC power

in Figure 3.14. The voltage across a 10 mΩ resistor is amplified using a current sense amplifier to produce an output voltage of 500 millivolts per Ampere of current. The output voltage of the current sense amplifier is fed in to the XADC auxiliary channel to receive a digital value of the current that corresponds to the amplified voltage. The product of core supply current and the FPGA core voltage (0.95V) will give the power consumption of the design. Similarly, a 5 mΩ resistor and the associated circuitry allows measuring total current drawn from the 5V supply. The CAD flow for power measurement on xilinx ARTY board is shown in Figure 3.15.
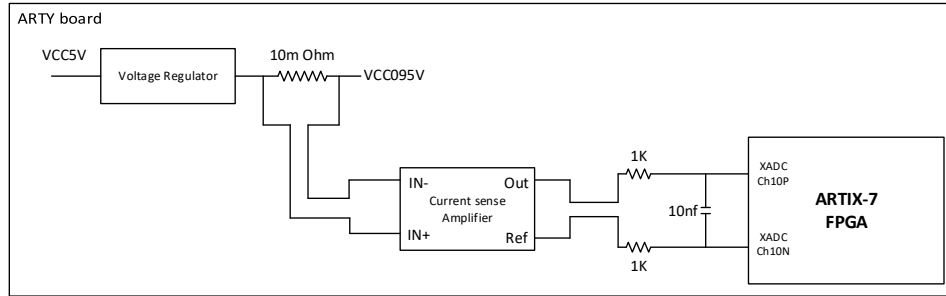
Figure 3.14: Current measurement circuitry on ARTY Board
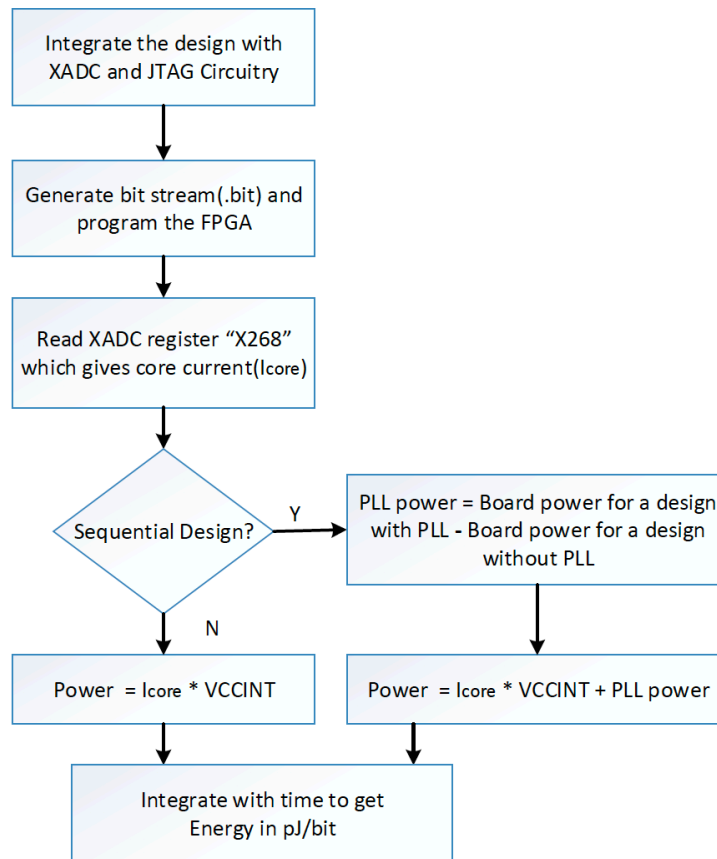
Figure 3.15: CAD flow in Xilinx devices for board level power measurement

The experimental setup for measuring power in an Artix 7 device on the ARTY development board is shown in Figure 3.16. An initialized ROM controlled by a free running cycle counter sends new data to the design every cycle. The module input ports (e.g. the key and the round constant in block ciphers) are stored in internal

registers. XADC and JTAG circuitry facilitates measurement of the current drawn by the FPGA core.

The XADC auxiliary channel-10 monitors the FPGA core current. The binary value of the current is obtained using an AXI-stream interface with a JTAG connection using the Vivado Tcl console. Channel-10 data is stored in a register with offset 0x268 that can be read using this Tcl command

```
run_hw_axi [create_hw_axi_txn rd [get_hw_axis hw_axi_1]
-address (BASE+0268) -len 1  -type read -f]
```

The value returned is the voltage across the 10 mΩ resistor represented in binary. The equivalent current for the returned binary value X is $\frac{X}{4096} \times 2$ A. The product of core voltage (0.95V) and the measured current is the power consumed by the design.
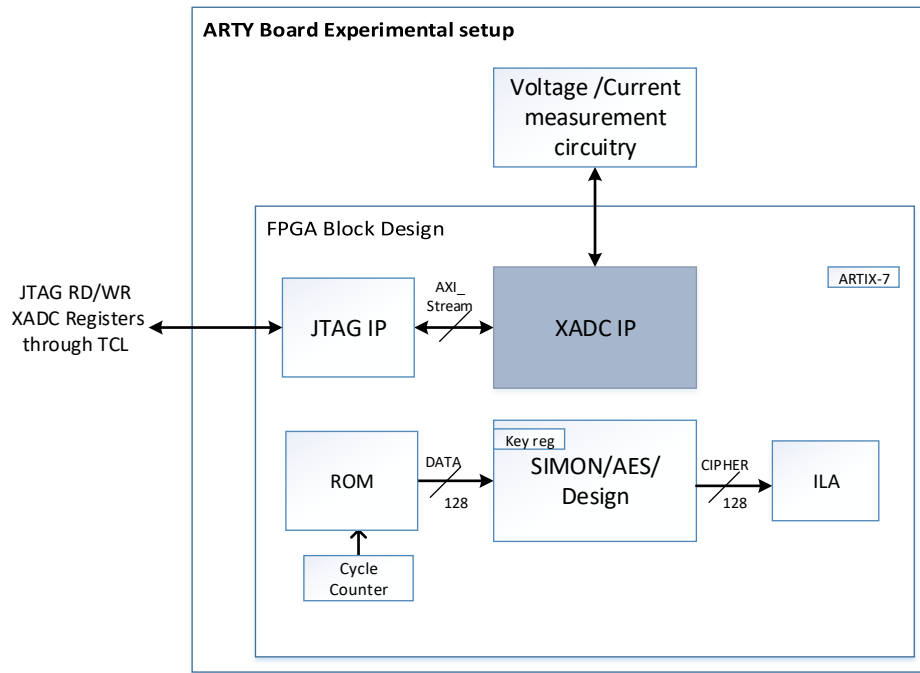
Figure 3.16: Experimental setup for physical power measurement in Xilinx Artix 7 device using ARTY development board

The FPGA core supply (VCCINT) powers internal logic elements such as slices and block RAMs (BRAMs). The auxiliary supply ($VCCAUX = 1.8$V) drives phase

locked loops (PLL), 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 power consumed by design with and without the PLL.

FPGA static power is the power 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. This static power is subtracted from the total core power to determine the dynamic power. Only dynamic power values are reported in the following sections of this thesis document.

We verified our results by replicating the design and observed the power scaling with the number of instances. The block design as it appears in Xilinx Vivado with the IP integrator flow is shown in Figure 3.17. A user design can be included and subjected to power consumption measurement.
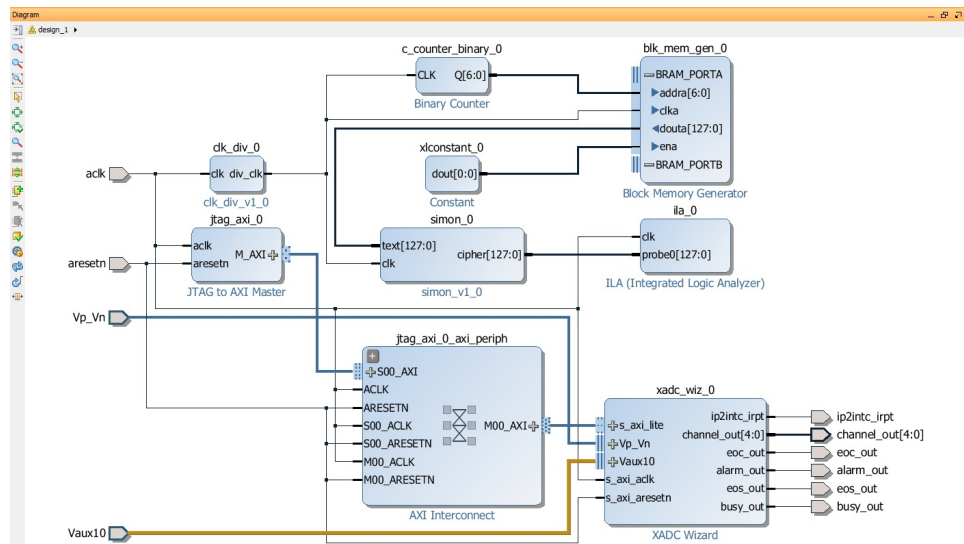
Figure 3.17: Xilinx Vivado Block design with IP integrator flow. A SIMON-128 is integrated into the middle of the figure.

3.4.4 Power measurement on the Altera Cyclone-IV-GX development board

A Cyclone IV-GX development board [1] was used for monitoring the power consumed by an Altera EP4CGX150DF21 (speed grade C7) FPGA. which contains 9360 Logic array blocks (LABs). The board also includes a MAX II EPM2210GF256 CPLD and on-Board power measurement circuitry. A picture of the board is shown in Figure 3.18.
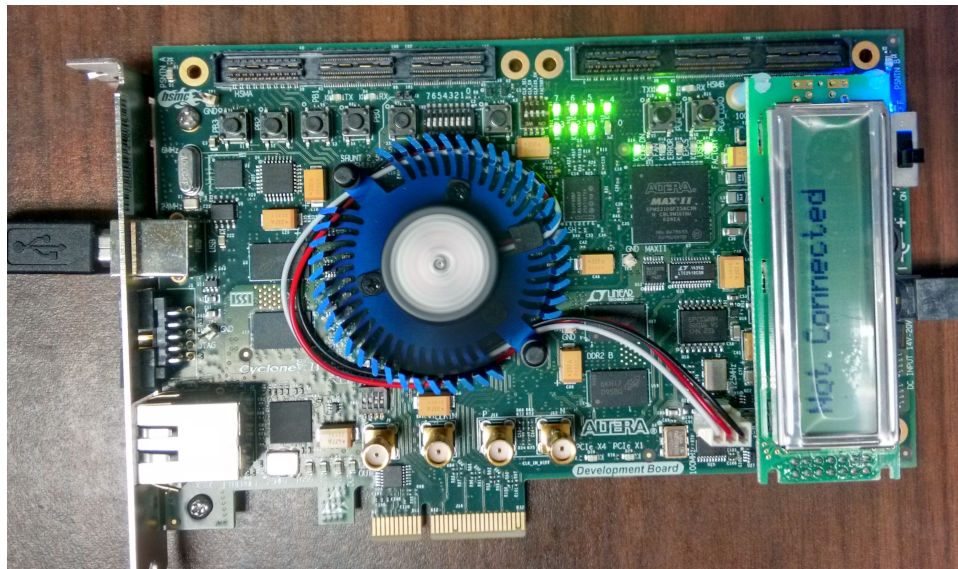
Figure 3.18: Altera Cyclone IV GX development Board [1]

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. The circuitry for measuring power on the Cyclone IV-GX development board is shown in Figure 3.19. An SPI bus connects the ADC device to the MAX II CPLD system controller. The experimental setup for measuring power in the FPGA on the Cyclone IV GX development board is shown in Figure 3.20. The CAD flow for power measurement on Altera Cyclone IV GX board is shown in Figure 3.21
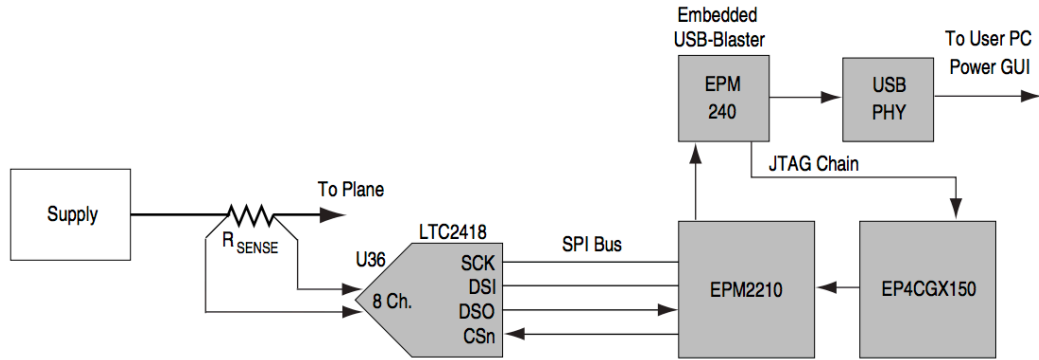
Figure 3.19: Current measurement circuitry on Cyclone IV Board [1]
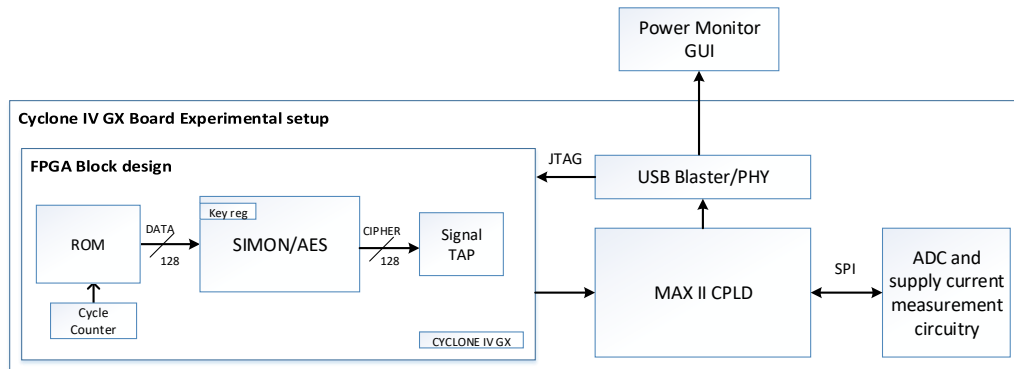
Figure 3.20: Experimental setup for power measurement in Altera Cyclone IV GX devices using the development board

Cyclone IV board is powered by 16V DC input and is connected to PC using USB type-B connector. To measure the power consumed by a design, the FPGA is programmed with an .sof file and the power is monitored in a GUI running on a PC. Figure 3.22 shows the power monitor GUI associated with the Altera Cyclone IV GX development board. It monitors the current drawn from all board supply voltages. The supply rails we consider for dynamic power measurement experiments are listed in Table 3.2.

The product of VCC current measured 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 con-

Table 3.2: Rail voltages in Altera power monitor GUI

Rail	Voltage (V)	Description
VCCA	2.5	PLL analog power
VCCD_PLL	1.2	PLL digital power
VCC	1.2	FPGA core power
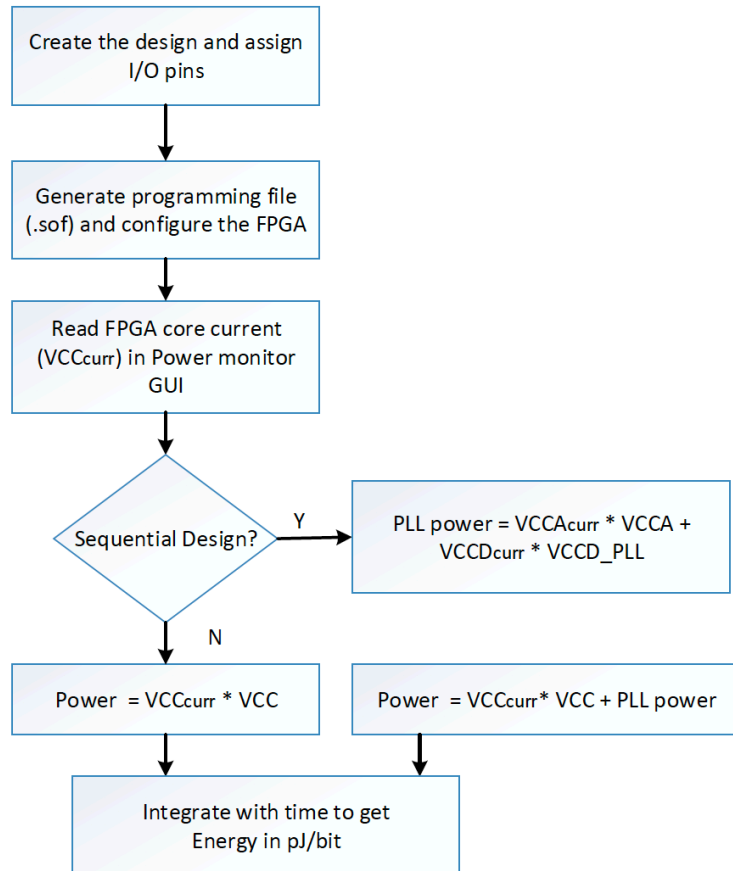
Figure 3.21: CAD flow in Altera devices for board level power measurement

tributed by VCCD_PLL, VCCA, and VCC. We measured the static power by programming the FPGA with a blank design and then monitoring the supply current. This value is measured to be 141.4 mW in an EP4CGX150DF21C7 FPGA. The dynamic power is the difference of the power measured with the user design and the measured static power. Only dynamic power values are used in Section 4.
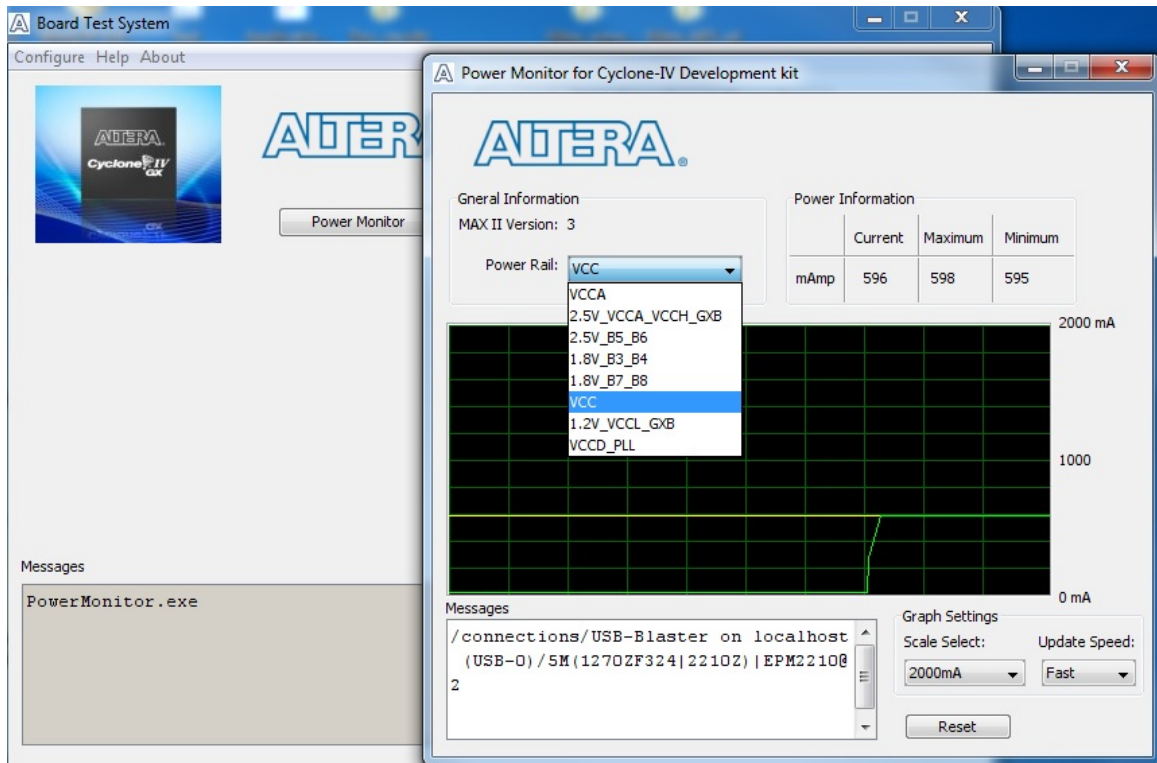
Figure 3.22: Altera board power GUI which monitors current drawn from different voltage rails

CHAPTER 4

RESULTS AND ANALYSIS

4.1 Latch-based and Flip flop-based Glitch Filters

The impact of implementing glitch filters using flip-flops instead of latches was discussed in Section 3.2. We compare the energy and area of latch-based and flip-flop based glitch filter implementations in this subsection. Glitch filters are inserted after optimal number of rounds (detailed in Section 4.2). Table 4.1 shows the energy and area comparison for the different glitch filter implementations in the Artix 7 device. The energy consumed per bit is similar in both implementations since a latch and a flip-flop uses the same circuitry in an Artix 7 FPGA. The difference in area results occurs because only four out of eight flip flop blocks in an Artix 7 FPGA slice can be configured as a latch.

Table 4.2 shows the energy and area comparison for the different glitch filter implementations in the Cyclone IV device. We observe that the designs with LUT-based and flip-flop based glitch filter implementations consume comparable energy. The number of LUTs required with the latch-based implementations is higher as latches are mapped to LUTs in Cyclone IV device. The number of LABs (LUTs and registers) utilized is similar in both the implementations.

4.2 Optimal Glitch Filter Placement

The main idea of our filtering approach is to reduce glitching in an unrolled design while minimizing area overhead. Glitch filters introduced into the design are capable of reducing glitches but they consume area and energy. In our experiments, we

Table 4.1: Energy and area comparison for SIMON-128, AES-256 and bitonic sort with Xilinx Artix 7 latch based and flip-flop based implementations

	Filtering Approach	Xilinx Latch	Xilinx Flip-Flop
SIMON	Energy Measured(pJ/bit)	34.3	32.1
	Area(slices)	1436	1412
AES	Energy Measured(pJ/bit)	57.0	51.0
	Area(slices)	4628	4371
bitonic sort	Energy Measured(pJ/bit)	9.5	8.0
	Area(slices)	3792	3431

Table 4.2: Energy and area comparison for SIMON-128, AES-256 and bitonic sort with Altera Cyclone IV latch based and flip-flop based implementations

	Filtering Approach	Altera Latch (LUT)	Altera Flip-Flop
SIMON	Energy Measured (pJ/bit)	90.0	90.0
	Area (LUTs/LABs)	24219/1962	19968/2020
AES	Energy Measured(pJ/bit)	168.8	168.8
	Area (LUTs/LABs)	67067/4714	65403/4510
bitonic sort	Energy Measured(pJ/bit)	24.7	21.1
	Area (LUTs/LABs)	30584/3084	23416/3034

analyzed the area and energy tradeoffs of inserting glitch filters. To find the optimal placement of the glitch filters, We unrolled the AES, SIMON and bitonic sort designs and inserted filters after a variable number of rounds. We compare the dynamic energy consumption and area overhead at different points and choose the best filter placement in terms of reduced energy consumption[1].

Tables 4.3 and 4.4 show the energy per bit and area overhead with filter placement at different numbers of rounds for AES-256, SIMON-128 and bitonic sort. We observe that placing a filter after every round for AES-256, every two rounds for SIMON-128,

---

[1]Optimal glitch placement results in this section were generated by Mr. Shivukumar Patil.

Table 4.3: Dynamic energy per bit (pJ/bit) and overall area a Xilinx Artix-7 device if glitch filter insertion is performed every n rounds where n is 1 to 7. *Max* indicates no glitch filtering was used. Both block ciphers and the sorting algorithm were fully unrolled to generate these results.

	Filter Spacing	1	2	3	5	7	MAX
SIMON	Energy Estimated (pJ/bit)	196.8	133.7	131.2	151.4	194.3	1304.6
	Energy Measured (pJ/bit)	43.8	**34.3**	41.2	54.5	64.3	240.4
	Area(slices)	2344	1436	1378	1374	1363	1133
AES	Energy Estimated (pJ/bit)	16.7	54.5	145.5	1580.6	983.2	3549.7
	Energy Measured (pJ/bit)	**57.0**	118.1	194.2	256.4	347.6	584.5
	Area(slices)	4628	4338	4335	4356	4288	4258
bitonic sort	Energy Estimated (pJ/bit)	9.5	29.8	39.5	48.1	54.5	78.3
	Energy Measured (pJ/bit)	10.6	**9.5**	10.9	13.3	17.1	30.9
	Area(slices)	5481	3792	3744	3841	3439	3074

and every two stages for bitonic sorting is optimal in terms of energy consumption and appropriate in terms of area.

We observe that Xilinx and Altera power estimation tools make relatively accurate estimates for circuits with low glitching activity and overestimate power for glitchy circuits [15]. We notice the deviation of estimated energy values from the measured values as we increase the filter spacing indicating the presence of additional glitching.

Table 4.4: Dynamic energy per bit (pJ/bit) and overall area in an Altera Cyclone IV device

	Filter Spacing	1	2	3	5	7	MAX
SIMON	Energy Estimated (pJ/bit)	200.1	180.8	188.8	344.3	459.7	23163.7
	Energy Measured (pJ/bit)	101.3	**90.0**	101.3	118.1	146.3	3420.0
	Area(LABs)	2198	1962	1838	1835	1808	1111
AES	Energy Estimated (pJ/bit)	209.5	887.1	2120.6	5360.5	7531.9	20804.0
	Energy Measured (pJ/bit)	**168.8**	337.5	554.1	1046.3	1333.1	3515.6
	Area(LABs)	4714	4514	4395	4331	4342	4338
bitonic sort	Energy Estimated (pJ/bit)	36.7	29.4	36.1	49.7	80.1	368.2
	Energy Measured (pJ/bit)	31.9	**24.7**	28.4	34.1	43.8	100.5
	Area(LABs)	3233	2299	2054	1861	1785	1559

In all the following results energy values were generated on the Arty or Cyclone IV board using latch-based glitch filters and area numbers are taken from the place and route report.

4.3 Fully Unrolled Energy and Area Results

To show the energy savings of our approach, we implement the block cipher designs in three different modes.

- Fully unrolled implementation with no glitch filtering - This implementation has high energy due to glitch propagation.

- Fully unrolled implementation with latch based glitch filters placed after the best number of rounds - This implementation suppresses glitches.

- Sequential implementation, which uses the outputs of each round as feedback - To maintain the same encryption latency as the fully-unrolled with glitch filter case, a PLL is needed to generate a fast clock that can be used for registering values after every round.

The fully unrolled implementations require a lower clock frequency compared to the sequential implementations to achieve the same encryption latency. Measured encryption latencies for the unrolled versions with glitch filters are 175 ns for AES, 340 ns for SIMON-128 and 120 ns for bitonic sort in Xilinx devices and 300 ns for AES, 600 ns for SIMON-128 and 220 ns for bitonic sort in Altera devices. Tables 4.5 and 4.7 show the energy comparison for the different implementations of SIMON-128, AES-256 and bitonic sort. Our filtering approach consumes the least energy among all implementations.

Compared to the fully unrolled implementation without filters, the energy savings are about 86% for SIMON, 90% for AES and 69% for bitonic sort in Artix 7 devices. The energy savings are about 97% for SIMON, 95% for AES and 76% for bitonic sort in Cyclone IV devices. Tables 4.6 and 4.8 shows an area comparison for the implementations in Xilinx Artix-7 and Altera Cyclone IV devices. The glitch filters lead to an area overhead of 26.7% in SIMON, 8.6% in AES and 23.3% in bitonic sort in Artix 7 devices and 76.5% in SIMON, 8.6% in AES and 47.4% in bitonic sort in Cyclone IV devices.

Sequential implementation reduces data energy as it needs minimal hardware. But these savings are offset by the energy needed to generate the high frequency local clock

Table 4.5: Energy comparison for SIMON-128, AES-256 and bitonic sort in Artix 7 devices for three different versions which generate an output every 175 ns, 340 ns and 120 ns respectively.

	Energy (pJ/bit)		
Design	Unrolled	Glitch Filtered	Sequential (with PLL)
SIMON-128	240.4	34.3	381.4
AES-256	584.5	57.0	246.8
Bitonic sort	30.9	9.5	107.1

Table 4.6: Area comparison for SIMON-128, AES-256 and bitonic sort for the three different versions in an Artix 7 FPGA

	Area (Slices)		
Design	Unrolled	Glitch Filtered	Sequential (with PLL)
SIMON-128	1133	1436	84
AES-256	4258	4628	370
Bitonic sort	3074	3792	4049

from the system clock. The PLLs are power hungry and they increase the dynamic power consumption of the sequential implementations. The energy breakdowns for SIMON-128 in Artix 7 and Cyclone IV implementations are shown in Figures 4.1 and 4.2. These energy breakdown values are estimated from simulation.

In the Xilinx Artix 7 FPGA, the clock energy is negligible compared to data energy in unrolled and glitch filtered scenarios, whereas it contributes the most to energy consumption in the sequential implementations. In the Altera Cyclone IV FPGA, PLL power is contributed by two components, analog 2.5V, and digital 1.2V. Both supplies have to be powered [2] even if PLLs are not used in the design which leads to PLL static power being much higher than PLL dynamic power. This issue makes sequential implementation energy efficient in dynamic power for Cyclone IV devices. Sequential versions consume almost the same energy as the filtered implementations and need the least area across all implementations.

Figures 4.3 and 4.4 show the energy versus area tradeoffs for SIMON-128, AES-256 and bitonic sort using the three implementation choices. These results suggest

Table 4.7: Energy comparison for SIMON-128, AES-256 and bitonic sort in Cyclone IV devices for three different versions which generate an output every 600 ns, 300 ns and 220 ns respectively.

	Energy (pJ/bit)		
Design	Unrolled	Glitch Filtered	Sequential (with PLL)
SIMON-128	3420.0	90.0	221.7
AES-256	3515.6	168.8	178.1
Bitonic sort	100.5	24.7	194.5

Table 4.8: Area comparison for SIMON-128, AES-256 and bitonic sort for the three different versions in a Cyclone IV FPGA

	Area (LABs)		
Design	Unrolled	Glitch Filtered	Sequential (with PLL)
SIMON-128	1111	1962	40
AES-256	4338	4714	121
Bitonic sort	1559	2299	1507

that the fully unrolled implementation is not suitable for low power applications. Glitch filters must be inserted if we unroll the design as it gives huge power savings with minimal area penalty. The sequential implementation with a PLL is both energy and area efficient in Cyclone IV devices. In Artix-7 devices, the sequential version is suitable only for applications which demand low area.

4.4 Effect of Partial Unrolling

In Section 4.3, we assessed the energy and area of fully unrolled designs that operate at low frequency (e.g. 5 MHz). For most FPGA applications, clock frequencies are much higher and these frequencies can be achieved via partial unrolling. Partial unrolling attains higher frequencies as the critical path traverses a small number of rounds. For example, in SIMON-128, by rolling four rather than 68 rounds, a 50 MHz clock can be used. Area is also saved as hardware is reused for multiple clock cycles. Energy savings with partial unrolling is reduced from full unrolling since registers
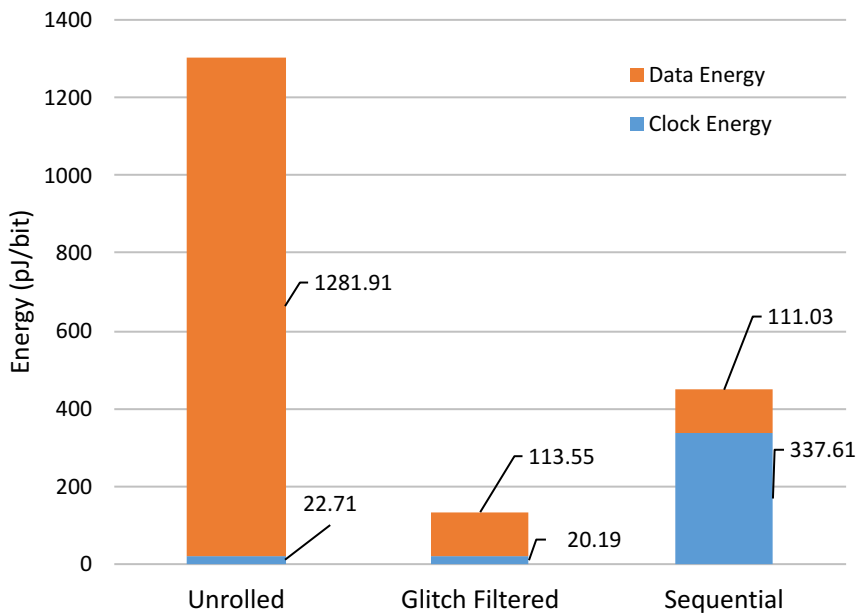
Figure 4.1: SIMON-128 energy breakdown for the three implementations in a Xilinx Artix-7 FPGA

store the intermediate state between the iterations. Energy savings are also limited by the repeated key computation logic with partial unrolling which is not required in fully unrolled implementations.

4.4.1 Energy Consumption with Partial Unrolling

We repeated the experiments from Section 4.3 to evaluate the area and energy required at different degrees of unrolling. We again use a constant latency for AES-256 (340 ns) and SIMON-128 (175 ns). Tables 4.9 and 4.10 list the energy per bit of encryption and the maximum clock frequency achieved by the designs with different degrees of unrolling for SIMON and AES in the Artix-7 device. Tables 4.11 and 4.12 list the energy details with the degree of unrolling for SIMON and AES in the Cyclone IV device.

The energy consumed by partial unrolling increases with the degree of unrolling. This effect is due to the increase in the glitches at the higher number rounds. It is beneficial to insert glitch filters even for partially unrolled designs as it offers good
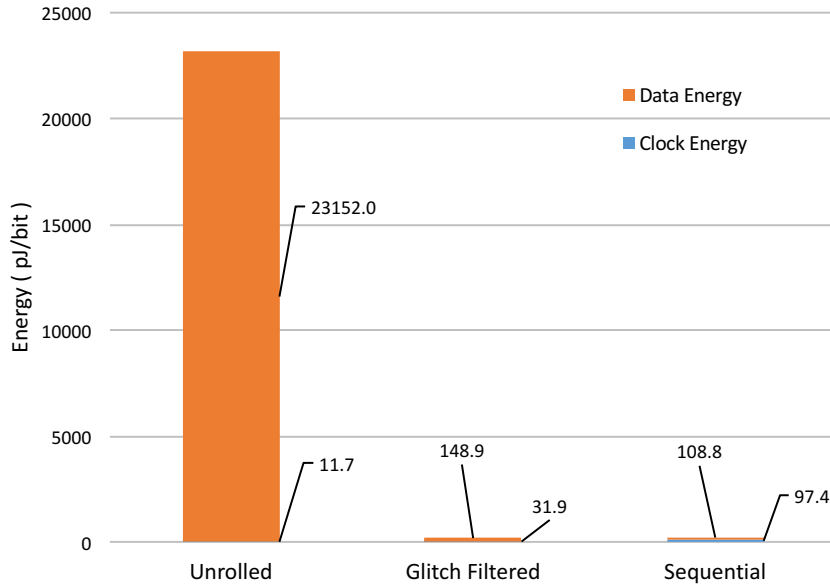
Figure 4.2: SIMON-128 energy breakdown for the three implementations in an Altera Cyclone IV FPGA

Table 4.9: Energy consumption with different degrees of unrolling for SIMON-128 in a Xilinx Artix-7 FPGA

Degree of Unrolling	Frequency (MHz)	Energy (pJ/bit)		
		Unrolled	Glitch filtered	Sequential
2	100.0	83.7	78.4	
4	50.0	100.9	76.8	
5	41.0	115.5	83.7	
7	29.5	142.1	83.7	381.4
10	20.5	189.9	86.3	
17	11.7	293.5	81.0	
68	2.9	240.4	34.3	

energy savings. Glitch filtered energy is lower than the partially unrolled version without filtering and the percentage of savings increases with a higher degree of unrolling.

Figures 4.5 and 4.6 show the energy per bit for SIMON and AES at different degrees of unrolling implemented in the three alternative design choices on an Artix-7 device. PLL energy refers to the energy consumed by the phase locked loop used
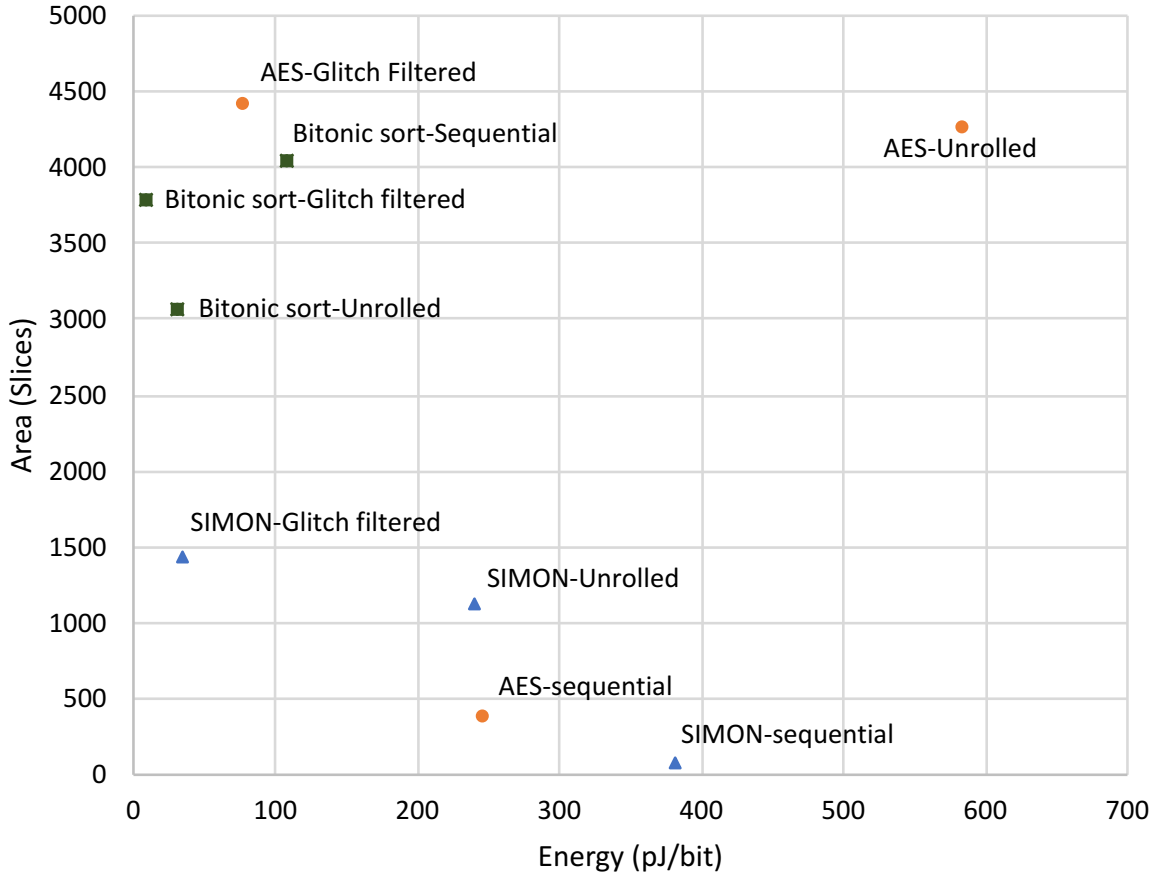
Figure 4.3: Energy versus area for AES and SIMON implemented using the three design choices in an Artix 7 device

Table 4.10: Energy consumption with different degrees of unrolling for AES-256 in a Xilinx Artix 7 FPGA

Degree of Unrolling	Frequency (MHz)	Energy (pJ/bit)		
		Unrolled	Glitch filtered	Sequential
2	40.0	216.7	97.8	
4	23.0	486.0	118.3	
6	17.0	878.4	111.4	246.8
8	11.0	1131.3	126.5	
14	5.7	584.5	57.0	

in generating the high frequency clock required for the sequential implementation. As expected, the PLL is the biggest contributor to sequential energy rendering it infeasible for low power applications. The unrolled implementation using glitch fil-
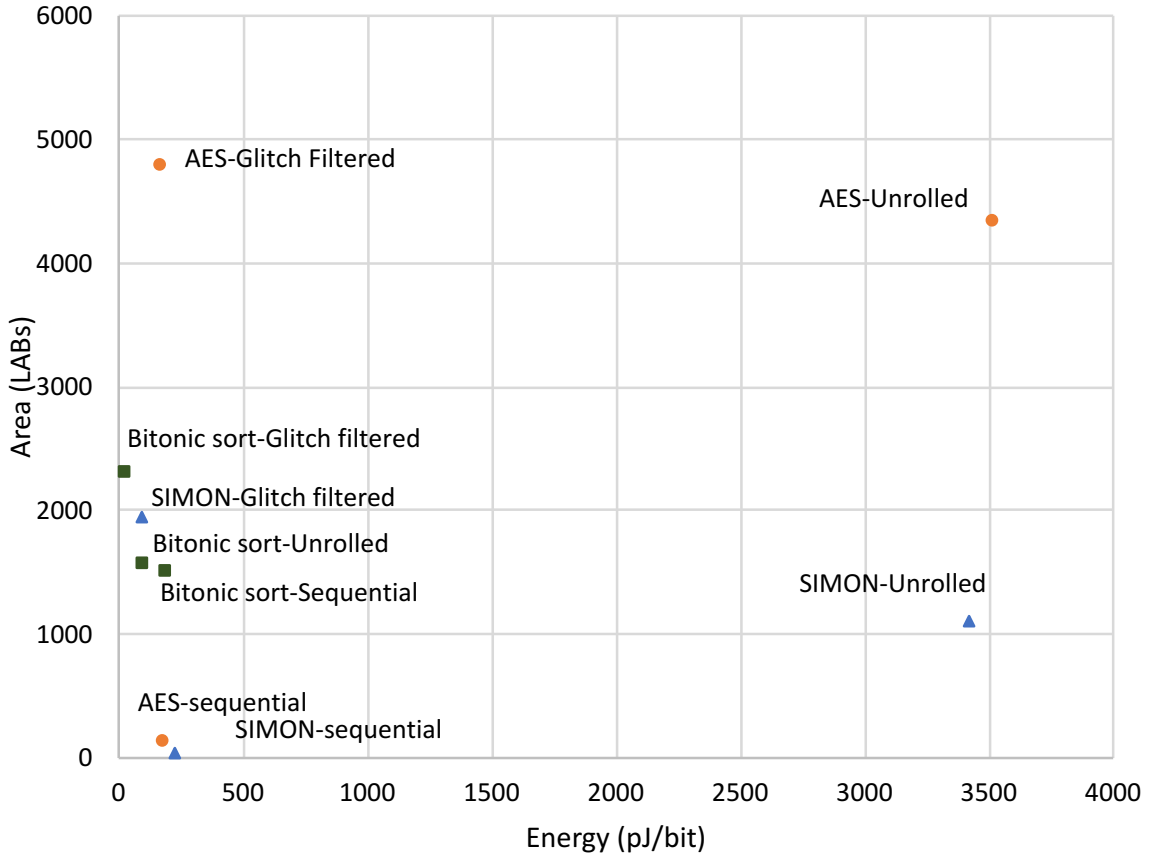
Figure 4.4: Energy versus area for AES and SIMON implemented using the three design choices in a Cyclone IV device

Table 4.11: Energy consumption with different degrees of unrolling for SIMON-128 in an Altera Cyclone IV FPGA

		Energy (pJ/bit)		
Degree of Unrolling	Frequency (MHz)	Unrolled	Glitch filtered	Sequential
2	57.0	135.0	**478.1**	
4	28.5	225.0	**247.5**	
5	23.5	225.0	**230.6**	
7	16.7	326.3	202.5	221.7
10	11.6	438.8	202.5	
17	6.6	984.4	208.1	
68	1.67	3420.0	90.0	

ters remains the most energy efficient across all frequencies but the approach has a significant area penalty.

Table 4.12: Energy consumption with different degree of unrolling for AES-256 in an Altera Cyclone IV FPGA

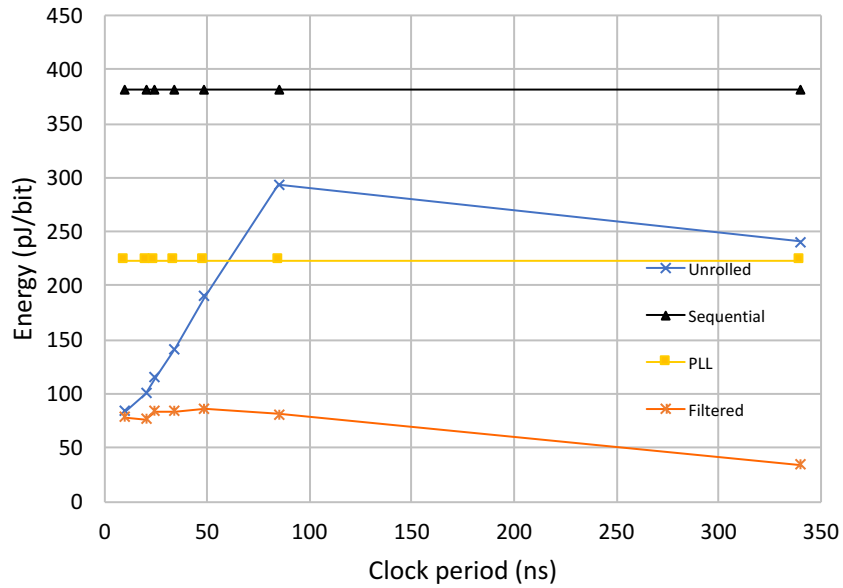| Degree of Unrolling	Frequency (MHz)	Energy (pJ/bit)		Sequential
		Unrolled	Glitch filtered	
2	23.3	720.0	317.8	
4	13.3	1929.4	345.9	
6	10.0	3287.8	390.9	178.1
8	6.6	3990.9	323.4	
14	3.3	3515.6	168.8	

Figure 4.5: Energy comparison with partial unrolling for SIMON-128 on Xilinx Artix-7 devices

Figures 4.7 and 4.8 show the energy per bit for SIMON and AES at different degrees of unrolling implemented in the three alternative design choices on Altera Cyclone IV devices. For SIMON-128, inserting glitch filters with partial unrolling of degree two, four and five does not give any energy savings. Filters should not used for the above cases. We observed that for both SIMON-128 and AES-256, unrolled version with filters and sequential version consumes similar energy across all frequencies.
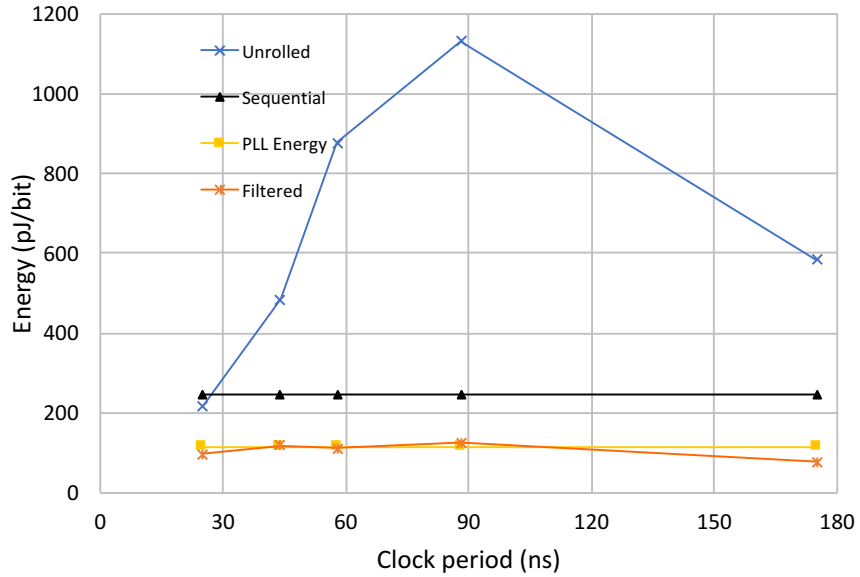
Figure 4.6: Energy comparison with partial unrolling for AES-256 on Xilinx Artix-7 devices

4.4.2   Area Utilization with Partial Unrolling

Figures 4.9 and 4.10 show an area comparison for SIMON and AES at different degrees of unrolling implemented in the three alternative design styles on Artix-7 devices. Of all the implementations, the sequential implementation requires the least area. The area with partial unrolling increases with the degree of unrolling as more rounds are implemented. The filtered implementation has higher area than the unrolled implementation without filtering due to latch and delay circuitry.

Figures 4.11 and 4.12 show an area comparison for SIMON and AES at different degrees of unrolling implemented in the three alternative design styles on Cyclone IV devices. Again, sequential implementation utilizes the least area and the filtered implementation has minimal area overhead compared to the unrolled implementation.
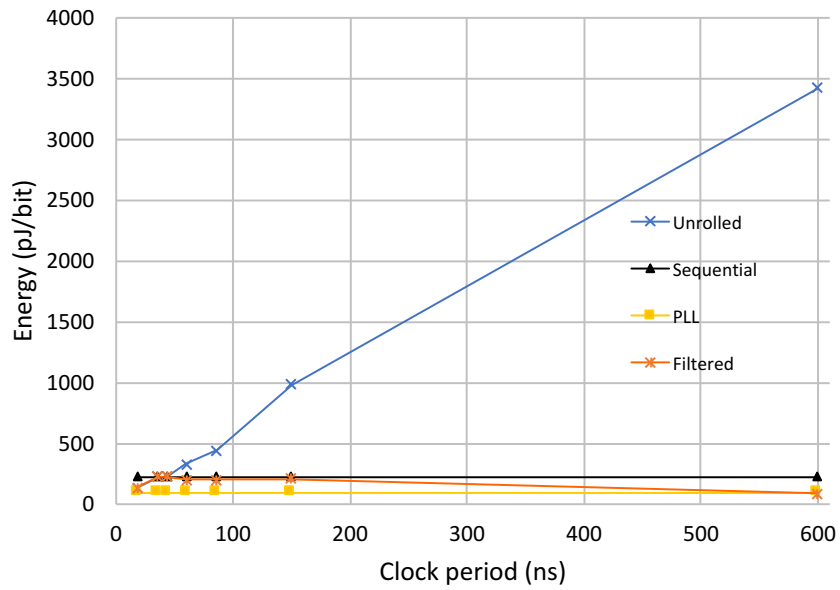
Figure 4.7: Energy comparison with partial unrolling for SIMON-128 on Altera Cyclone IV devices
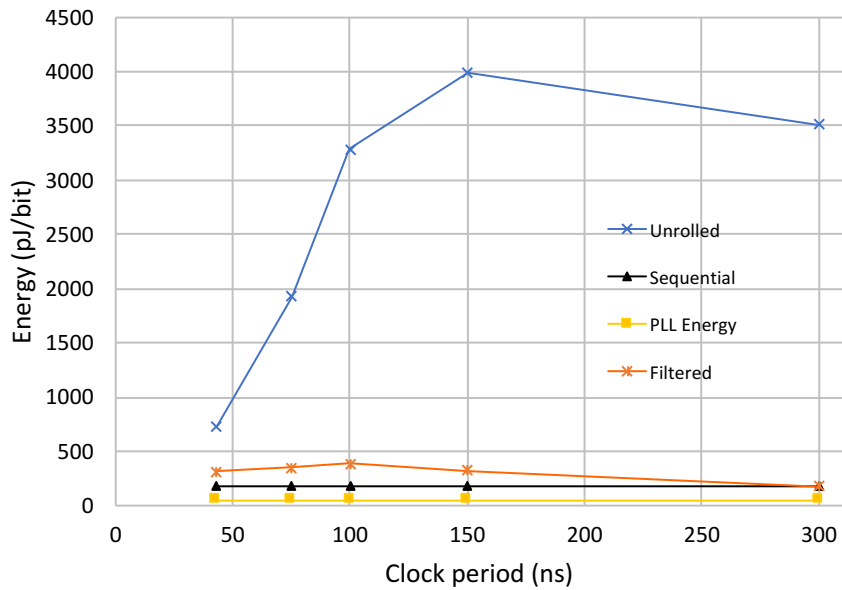

Figure 4.8: Energy comparison with partial unrolling for AES-256 on Altera Cyclone IV devices
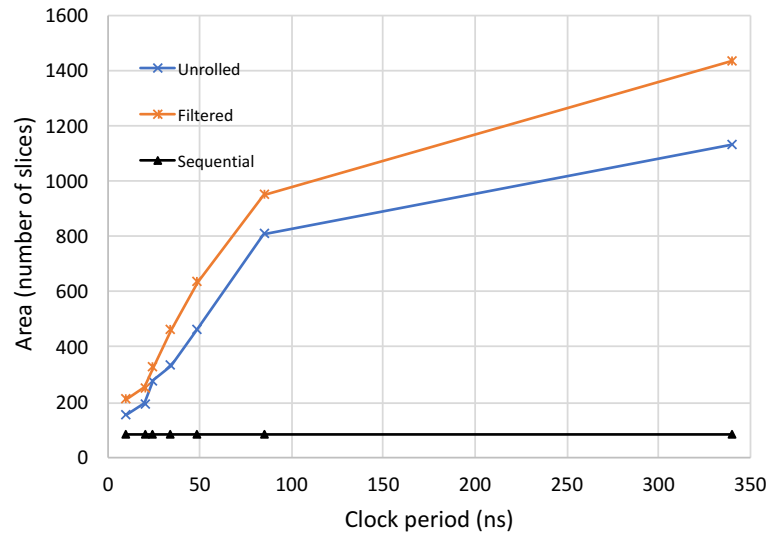
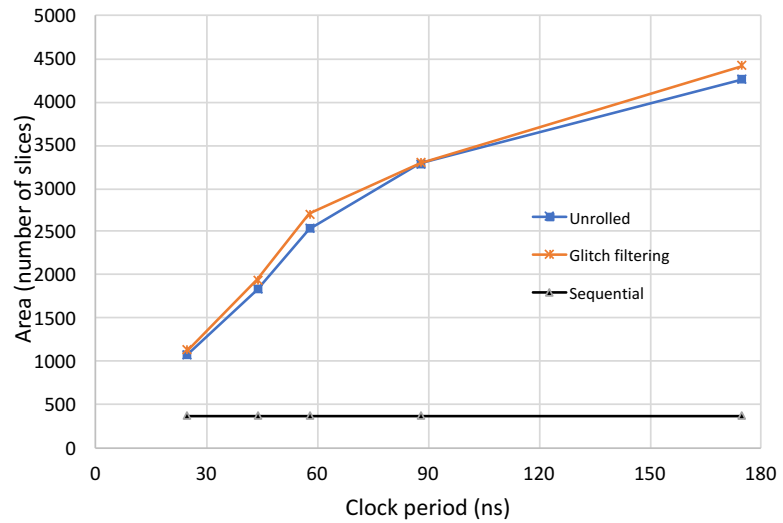Figure 4.9: Area comparison with partial unrolling for SIMON-128 on Xilinx Artix-7 devices

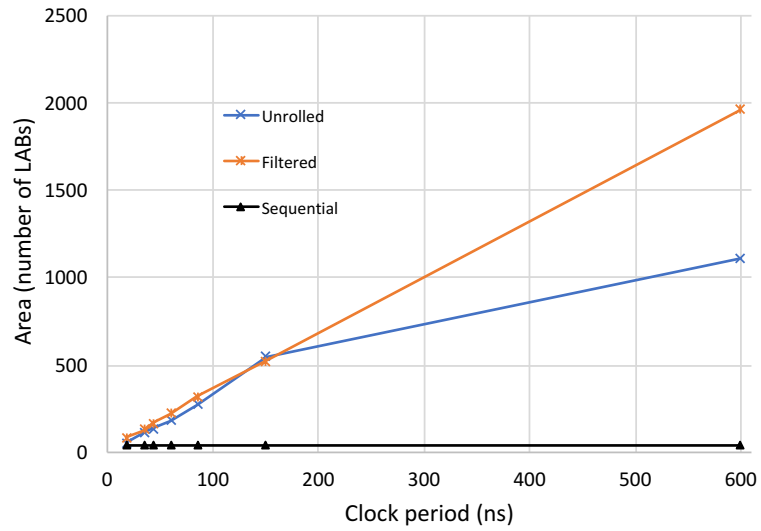Figure 4.10: Area comparison with partial unrolling for AES-256 on Xilinx Artix-7 devices

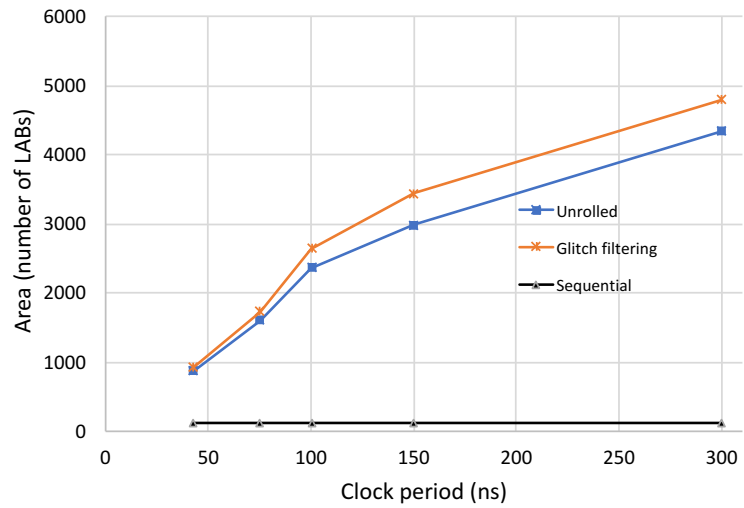Figure 4.11: Area comparison with partial unrolling for SIMON-128 on Altera Cyclone IV devices

Figure 4.12: Area comparison with partial unrolling for AES-256 on Altera Cyclone IV devices

43

CHAPTER 5

RESULTS WITH AN EMBEDDED APPLICATION

Embedded applications 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 block ciphers with an FFT application. We obtained FFT RTL from Spiral.net [28] and chosen transform size and sample width is 16. We replicated FFT core to ensure that the entire chip was utilized for experimentation.
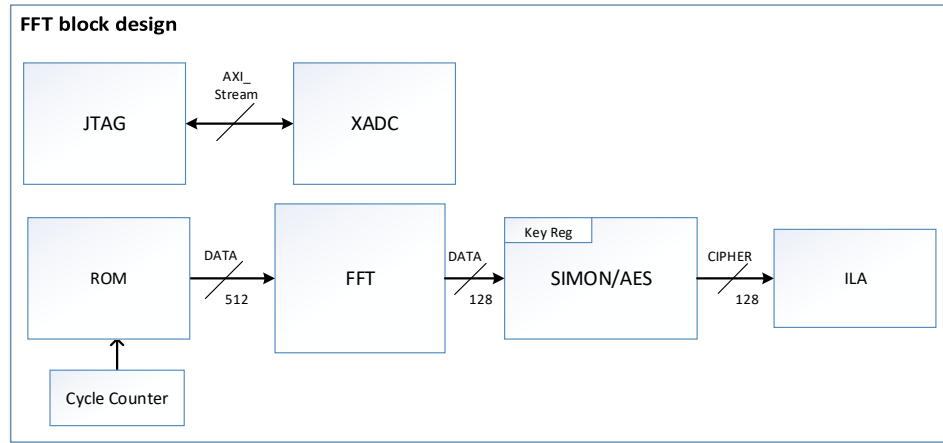
Figure 5.1: Experimental setup to measure power for FFT design integrated with a block cipher

The experimental setup to measure the power consumed by the FFT circuit on the Arty board is shown in Figure 5.1. A ROM with initialized contents drives new data to FFT core every clock cycle. The output from FFT block is given as an input to the block cipher. The block cipher (AES/SIMON) is implemented as fully unrolled or partially unrolled with filters or sequentially for each experiment. A key is stored

44

Table 5.1: Energy comparison for FFT integrated with SIMON-128 and AES-256 in an Artix 7 FPGA

Design	Energy (pJ/bit)		
	Unrolled	Glitch Filtered	Sequential (with PLL)
FFT + SIMON-128	314.7	88.9	427.7
FFT + AES-256	592.6	73.1	252.2

Table 5.2: Area comparison for FFT integrated with SIMON-128 and AES-256 in an Artix 7 FPGA

Design	Area (Slices)		
	Unrolled	Glitch Filtered	Sequential (with PLL)
FFT + SIMON-128	6490	6872	5114
FFT + AES-256	5797	5916	2121

as an initialized constant register inside the block cipher design. The output of the encryption core is connected to a debug port. About 98% of the LUTs in the chip are occupied for both SIMON and AES integrated designs.

Table 5.1 shows the energy comparison for FFT integrated with the different implementations of SIMON-128 and AES-256. Compared to the fully unrolled implementation, the energy savings are about 72% for SIMON and 87% for AES. The sequential version consumes higher energy than the filtered version due to PLL circuitry. Table 5.2 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.

Figure 5.2 shows relative placement of the FFT design, SIMON block cipher, glitch filter and debug circuitry in a 98% occupied Artix 7 FPGA. We observe that carry chain and latch circuitry is packed close to each other and shares slices with other SIMON round circuitry.
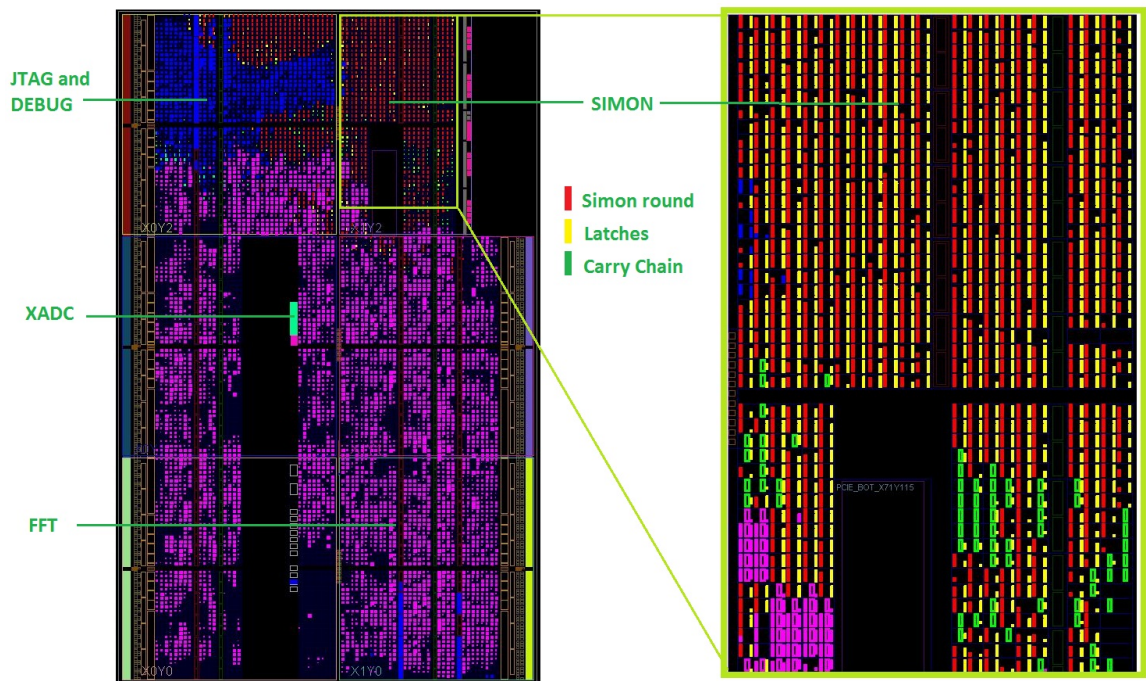
Figure 5.2: Relative placement of FFT, SIMON-128 and glitch filtering circuitry in a Xilinx Artix 7 FPGA. Red rectangles indicate LUTs, Yellow rectangles indicate registers configured as latches and green rectangles indicate carry chains

CHAPTER 6

CONCLUSION

In this thesis document we presented a new mechanism to implement loops in low-cost FPGA in an energy-efficient manner. We demonstrated our approach on Xilinx Artix-7 and Altera Cyclone IV FPGAs using loop-based functions SIMON and AES block ciphers and a bitonic sort algorithm. Compared to an unrolled implementation, we showed savings up to 90% dynamic energy in Artix-7 and 97% in Cyclone IV designs with our approach. We also showed the optimal placement of glitch filters which gives the best tradeoff between energy consumption and area. We partially unrolled the designs and showed that different degrees of unrolling and filter insertion lead to different energy-saving benefits. We also implemented an embedded FFT on an Artix-7 FPGA and evaluated our filtering approach for dynamic energy consumption in a highly utilized chip.

Future work could consider implementing glitch filtering in additional benchmarks and migrating the glitch filtering circuitry to other family devices and speed grades.

BIBLIOGRAPHY

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

[2] Altera. Altera Cyclone IV GX pin connection guidelines. `https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/dp/cyclone-iv/pcg-01008.pdf`.

[3] Andraka, Ray. A survey of CORDIC algorithms for FPGA based computers. In *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field Programmable Gate Arrays* (1998), ACM, pp. 191–200.

[4] Banik, Subhadeep, Bogdanov, Andrey, and Regazzoni, Francesco. Exploring energy efficiency of lightweight block ciphers. In *International Conference on Selected Areas in Cryptography* (2015), Springer, pp. 178–194.

[5] Batina, Lejla, Das, Amitabh, Ege, Barış, Kavun, Elif Bilge, Mentens, Nele, Paar, Christof, Verbauwhede, Ingrid, and Yalçın, Tolga. Dietary recommendations for lightweight block ciphers: power, energy and area analysis of recently developed architectures. In *International Workshop on Radio Frequency Identification: Security and Privacy Issues* (2013), Springer, pp. 103–112.

[6] Beaulieu, Ray, Treatman-Clark, Stefan, Shors, Douglas, Weeks, Bryan, Smith, Jason, and Wingers, Louis. The SIMON and SPECK lightweight block ciphers. In *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE* (2015), IEEE, pp. 1–6.

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

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

[9] Czajkowski, Tomasz S, and Brown, Stephen D. Using negative edge triggered FFs to reduce glitching power in FPGA circuits. In *Design Automation Conference, 2007. DAC'07. 44th ACM/IEEE* (2007), IEEE, pp. 324–329.

[10] Dhanuskodi, Siva Nishok, and Holcomb, Daniel. Energy Optimization of Unrolled Block Ciphers using Combinational Checkpointing. *IACR Cryptology ePrint Archive 2016* (2016), 1093.

[11] Elbirt, Adam J, Yip, Wei, Chetwynd, Brendon, and Paar, Christof. An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 9*, 4 (2001), 545–557.

[12] Gaj, Kris, and Chodowiec, Pawel. Fast implementation and fair comparison of the final candidates for Advanced Encryption Standard using Field Programmable Gate Arrays. In *Cryptographers Track at the RSA Conference* (2001), Springer, pp. 84–99.

[13] Hsing, Homer. tiny_aes AES core. `http://opencores.org/project,tiny_aes`, 2015.

[14] Huda, Safeen, and Anderson, Jason. Towards PVT-Tolerant Glitch-Free Operation in FPGAs. In *Proc. ACM/SIGDA International Symp. on FPGAs* (Feb. 2016).

[15] Jevtic, Ruzica, and Carreras, Carlos. Power measurement methodology for FPGA devices. *IEEE Transactions on Instrumentation and Measurement 60*, 1 (2011), 237–247.

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

[17] Lamoureux, Julien, Lemieux, Guy, and Wilton, Steven. GlitchLess: Dynamic Power Minimization in FPGAs Through Edge Alignment and Glitch Filtering. *IEEE Transactions on VLSI 16*, 11 (Nov. 2008), 1521–1534.

[18] Lim, Hyeonmin, Lee, Kyungsoo, Cho, Youngjin, and Chang, Naehyuck. Flip-flop insertion with shifted-phase clocks for FPGA power reduction. In *Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM International Conference on* (2005), IEEE, pp. 335–342.

[19] Maene, Pieter, and Verbauwhede, Ingrid. Single-Cycle Implementations of Block Ciphers. In *International Workshop on Lightweight Cryptography for Security and Privacy* (2015), Springer, pp. 131–147.

[20] Mehta, Nick. Xilinx 7 Series FPGAs: the logical advantage. *Xilinx WP405* (2012).

[21] Pub, NIST FIPS. 197: Advanced Encryption Standard (AES). *Federal Information Processing Standards Publication 197*, 441 (2001), 0311.

[22] Ravi, Srivaths, Raghunathan, Anand, Kocher, Paul, and Hattangady, Sunil. Security in embedded systems: Design challenges. *ACM Transactions on Embedded Computing Systems (TECS) 3*, 3 (2004), 461–491.

[23] Robson, Clyde CW, and Bohm, Christian. A high speed data acquisition collector for merging and sorting data. In *Nuclear Science Symposium Conference Record, 2008. NSS'08. IEEE* (2008), IEEE, pp. 855–856.

[24] Sekanina, Lukáš. Evolutionary design space exploration for median circuits. In *Workshops on Applications of Evolutionary Computation* (2004), Springer, pp. 240–249.

[25] Shum, Warren, and Anderson, Jason H. FPGA Glitch Power Analysis and Reduction. In *Proc. Int'l Symp. Low Power Electronics and Design* (Aug. 2011).

[26] Standaert, François-Xavier, Rouvroy, Gael, Quisquater, Jean-Jacques, and Legat, Jean-Didier. A methodology to implement block ciphers in reconfigurable hardware and its application to fast and compact AES RIJNDAEL. In *Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays* (2003), ACM, pp. 216–224.

[27] Xilinx. Artix-7 35T ARTY FPGA Evaluation Kit. `http://www.xilinx.com/products/boards-and-kits/arty.html#documentation`.

[28] Zuluaga, Marcela. Sorting Network IP Generator. `http://www.spiral.net/hardware/sort/sort.html`, 2012.