2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines

Energy Efficient Loop Unrolling for Low-Cost FPGAs

Naveen Kumar Dumpala, Shivukumar B. Patil, Daniel Holcomb, and Russell Tessier
University of Massachusetts, Department of Electrical and Computer Engineering, Amherst, MA 01003

***Abstract*—Many FPGA computations, including block ciphers, require repetitive loop operations that are difficult to parallelize. Sequential loop implementation leads to significant clock power while loop unrolling can lead to significant glitch power. In this paper, we provide a low overhead approach to unroll block ciphers and other loops in low-cost FPGAs to reduce energy consumption. A latch-based glitch filter is introduced for unrolled loops that reduces loop energy per operation by over an order of magnitude. Our filters and associated control for unrolled loops can be automatically instantiated as a macro for FPGA designs, allowing for easy designer use. We demonstrate our approach for SIMON-128 and AES-256 block ciphers implemented on a Xilinx Artix-7 FPGA.**

I. Introduction

The energy consumption and latency of block ciphers used for encryption and other loop based functions are significant issues for many low-cost, embedded FPGA applications. Simply reducing the clock frequency needed to sequence loop-based ciphers reduces energy at the cost of increased latency. Although often modest in size compared to other computing elements, block ciphers are critical system components that typically serve at the boundary of device I/O. A well-known technique to reduce block cipher latency while controlling energy consumption is the use of loop unrolling [1]. However, unrolling leads to significant glitch power, increasing the block cipher's effects on FPGA dynamic energy. This effect is particularly apparent for high round count ciphers (e.g. SIMON-128 has 68 rounds) used in embedded FPGA applications.

In this paper we present an automated tool that will unroll block ciphers to meet required latency constraints while minimizing energy consumption. We address the glitch power issue by inserting latch-based glitch filters made from available FPGA logic resources in the unrolled computation path. The key aspect of our work is the use of a chain of FPGA carry elements that generate a set of signals with predictable delay to open latches after latch inputs have settled. When possible, the system clock is used to clock block cipher circuitry, eliminating the need to generate additional FPGA clock signals. We have macroized the delay generation and latch insertion circuitry so necessary circuitry can be easily inserted into a user's design using a script. We demonstrate our approach for Xilinx Artix-7 devices operating at a range of clock speeds below 100 MHz. AES-256 and SIMON-128 block ciphers are used for experimentation.

II. Related Work

Block ciphers for encryption algorithms such as AES and DES are modestly-sized but important parts of many low-cost embedded systems. These components encrypt or decrypt data under predefined latency constraints. Most block ciphers contain numerous simple round computations using keys derived from a user-specified key. The rounds can be performed sequentially with registers buffering data after every round. Alternatively, multiple copies of the round logic can be generated, effectively unrolling the computation. The use of sequential implementations of block ciphers has the benefit of small area and power at the cost of long latency, especially for ciphers like SIMON-128 that requires 68 rounds. In many cases the latency incurred by simply using the low frequency system clock to clock the sequential version will not meet block cipher latency constraints. Unrolling block cipher loops reduces the number of clock cycles needed to encrypt or decrypt data. Register energy is also reduced since flip flops toggle less per cipher operation. Although unrolling is known to reduce energy [1] [2] in some cases, it is prone to cause energy-consuming glitching as combinational pulses formed early in the combinational path propagate through the combinational circuit.

A variety of techniques have been developed to address glitch filtering in FPGAs. Most techniques [3] [4] involve the use of combinational path balancing to ensure that logic input signals arrive at the same time, minimizing glitch generation. Although effective, these techniques require the insertion of delay elements that are not present in commercial FPGAs. Several computer-aided design approaches to address glitching have been proposed. Guarded evaluation for FPGAs [5] 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 [6] 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. We view guarded evaluation and don't care synthesis as complementary to our work because they can be used in conjunction with our approach.

Although many FPGA designs now operate at clock speeds well in advance of 100 MHz, low-cost embedded FPGA applications still use system clocks that are below this frequency [7]. For example, low-energy optical processing can be performed at system frequencies of 30 MHz [8]. These low frequency designs 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 the use of high-energy phase-locked loops (PLL) for clock generation.

978-1-5386-4037-1/17 $31.00 © 2017 IEEE
DOI 10.1109/FCCM.2017.22

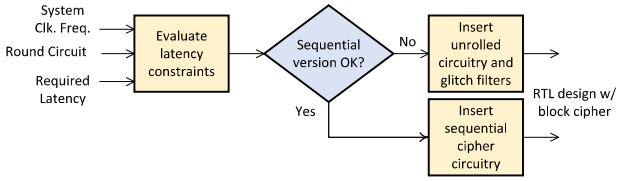

Fig. 1: Software flow for automatic block cipher unrolling

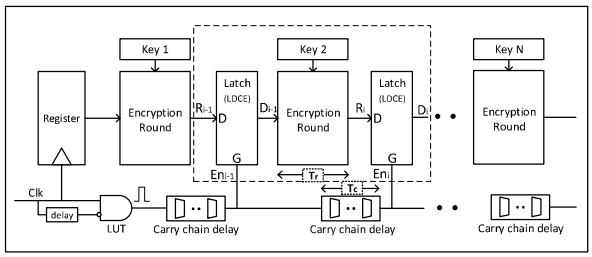

Fig. 2: Unrolled block cipher with latch-based glitch filtering

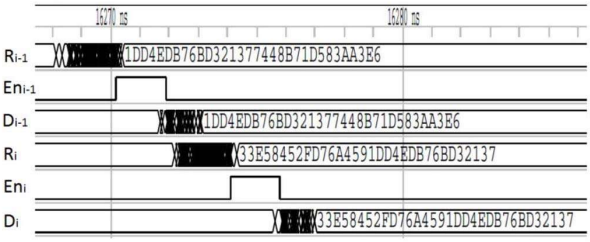

Fig. 3: Glitch filtering timing waveform

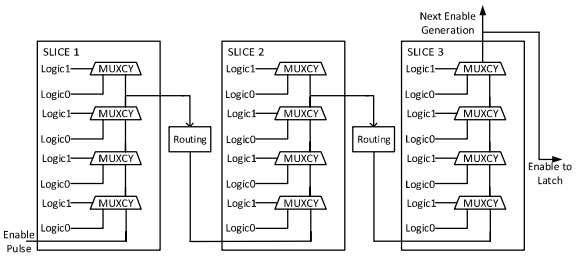

Fig. 4: Carry chain delay implementation in Artix-7 FPGA

III. Approach

The steps in our automated flow are shown in Fig. 1. The inputs to the flow are the RTL code of a single round of the cipher, the frequency of the system clock, and the latency constraint of the block cipher. The first block in the system evaluates whether the latency constraint can be met by simply sequentially using a single round and the system clock. If not, the amount of unrolling needed is determined. In a final step, glitch filtering is added to the partially (or wholly) unrolled block cipher design that is clocked by the system clock.

In general, determining the amount of unrolling needed to meet block latency constraints can straightforwardly be determined by synthesizing the single round of the loop and determining its latency. The more difficult task is the implementation and insertion of the glitch filtering in the FPGA.

A. Glitch Filter Insertion

In our system, we use the glitch filter implementation shown in Fig. 2 for unrolled designs. The block cipher input provided from the launch flip flops goes through one or more encryption rounds. The output of this circuitry is latched using a delayed enable (En) signal. The glitch filter is implemented using the transparent D-latch which, to avoid glitching, is enabled after combinational outputs of the round(s) have settled. The amount of delay needed is determined as part of the system outlined in Fig. 1. In our implementation, latches are implemented using Artix-7 LDCE primitives.

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 of carry chains (T_c) used to create delay elements must be greater than the round delay (T_r). 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 Vivado Xsim. Value R_{i-1} drives the first latch which is sampled using enable En_{i-1} and produces a 128-bit D_{i-1} with 70 toggled signals. If latching is not used, glitching increases the number of transitions to 130.

The key issues in glitch filter deployment are determining how long latch enables should be delayed for each delay line tap and accurately implementing the delay line in the FPGA so that taps are generated at these times. In our FPGA implementation, we use the multiplexers that make up the logic slice carry chain to generate predictable delays. By connecting chains in adjacent slices, long multi-tap delay chains can be formed.

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. The organization of a carry chain based delay chain is shown in Fig. 4. 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 encryption round. The output from the last slice is connected to the glitch filter latch and the input of the next delay element carry chain. Delay through a single slice carry chain was determined to be around 650 ps, Routing delays between multiplexers in adjacent slices are tightly controlled with routing constraints

to use a predictable fast path. Since slices are adjacent, these delays can be predicted.

IV. Experimental Methodology

Our automated energy-efficient unrolling approach was applied to AES-256 and SIMON-128 block ciphers. AES-256 uses 128 bit data, a 256-bit key, and 14 encryption rounds. SIMON-128 uses 128-bit data and key and requires 68 encryption rounds. The synthesizable AES design was obtained from Opencores.org. The synthesizable SIMON core was written from scratch. The cores were tested for correctness by comparing the generated ciphertext with golden reference values.

Our carry chain based delay circuit is parameterized and tunable based on the round delay of an encryption or other loop-based circuit. In our experiments we use three carry chain slice delays per round for SIMON-128 and seven slice delays per round for AES-256. Each slice has eight storage elements of which four can be configured as latches. It is easy to automatically integrate this circuitry and the delay chain into the encryption design using a script after delay analysis is performed. These resources are automatically instantiated as macros in our system.

AES-256 and SIMON-128 block ciphers were implemented on an Xilinx Artix-7 (xc7a50tftg256-2) device using Vivado. The block cipher 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. Once timing was met, post place and route netlist and standard delay format (SDF) files were generated for gate level timing simulation. This simulation captured the glitching behavior of the circuit. The Vivado Xsim simulator was used to perform timing simulation and generate a signal activity interchange format (SAIF) file, which recorded all the switching activity of the signals. Vivado Power Analyzer tools were used with the post route netlist, timing constraints, signal activity file and device operating parameters to generate a power consumption report. We integrated the power consumption values over simulation time to compute the energy consumed for the encryption.

V. Results

In our first experiment, we assess the tradeoff between energy reduction and area increase for glitch reduction latch insertion. To obtain this comparison, the loops are fully unrolled. Glitch filters introduced between rounds reduce overall round energy but add area and can consume energy themselves. To explore the optimum spacing between glitch filters, we measured energy and area after inserting latches after different numbers of rounds. Table I shows energy consumption with regards to distance of glitch filter insertion. Given the modest area overhead of the delay chains and latches, the optimal placement is determined to be adding glitch filters after each round for AES-128 and after two rounds for SIMON-128. For the fully unrolled case, glitch filter insertion leads to a 16% area increase for SIMON (2,411 versus 2,073 slices) and a 1.8% increase for AES (4,029 versus 3,957). The reduced overhead for AES is expected since it has fewer, larger rounds.

	1	2	3	5	7	max
Dynamic energy	(pJ/bit)					
SIMON-128	31.9	26.5	37.2	79.7	135.5	1,083.8
AES-256	59.7	220.9	516.3	882.0	1394.7	2,273.0
Area	(slices)					
SIMON-128	2,775	2,411	2,337	2,296	2,289	2,073
AES-256	4,029	4,150	4,211	4,298	4,308	3,957

TABLE I: Dynamic energy per bit (pJ/bit) and overall block cipher area 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 were fully unrolled to generate these results.

	SIMON-128		
	Sequential	Unrolled	Unrolled + Filtered
Data+Key	47.8	1,081.1	21.2
Clocking	305.4	2.6	5.3
Total	353.2	1,083.7	26.5
	AES-256		
	Sequential	Unrolled	Unrolled + filtered
Data+Key	208	2,272	57
Clocking	181	1	3
Total	389	2,273	60

TABLE II: Energy breakdown in pJ/bit for block ciphers. All three SIMON and AES versions generate one encryption every 340 ns and 175 ns, respectively. The local clock for the sequential version was generated with a PLL

We now consider a scenario in which the system clock frequency is slow enough to allow all rounds of the block cipher 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 glitch filters add latch-based 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.

Table II shows the energy consumption of the three different single system-cycle implementations of SIMON-128 and AES-256. All three cases for SIMON and AES generated an encryption result every 340 ns and 175 ns, respectively. The unrolled implementation consumes significant energy but glitch filtering reduces this energy by 97% for SIMON-128 and AES-256. Sequential implementation reduces data energy but the PLL used to generate a local clock from the system clock makes overall energy higher than the glitch filtering implementation. For the SIMON sequential design, we generated high frequency local clocks using system clocks ranging from 10-100 MHz. Our experiments showed that for SIMON-128, the PLL consumes between 96 and 126 mW over this range of frequencies. Similarly, for AES with an input system clock range from 10-40 MHz, the PLL consumes 86 to 123 mW while the clock power is about 11 mW. Clearly, PLLs add significant power and should be avoided in low energy designs. Fig. 5 shows the energy versus area tradeoffs of the three implementation choices.

Full loop unrolling is only feasible for extremely slow system clocks, and we now evaluate the same three design

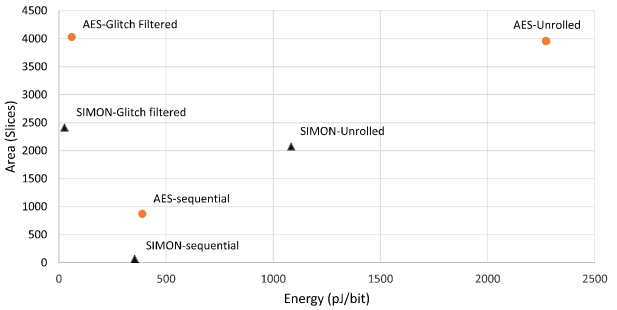

Fig. 5: SIMON-128 and AES-256 energy versus area for different implementations. All implementations of the same cipher have the same encryption latency.

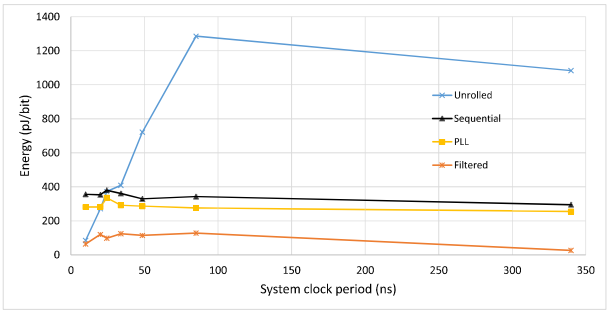

Fig. 6: Energy comparison of SIMON-128 implementations with the same block cipher latency of 340 ns. The fully unrolled data points are on the right and unrolled by a factor of 2 are on the left. *Filtered* indicates unrolled and filtered.

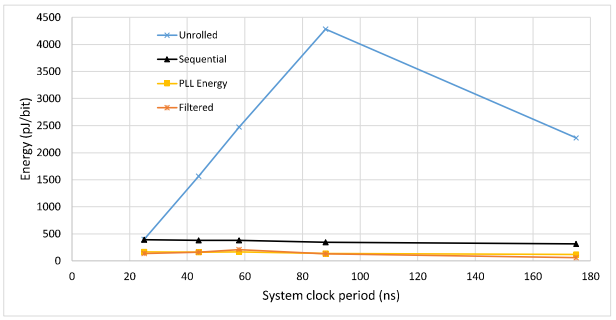

Fig. 7: Energy comparison of AES implementations with the same block cipher latency of 175 ns.

styles at a range of more typical system clock frequencies. As described in Section III, our software system considers the required latency of the block cipher and the input system clock frequency in determining how much unrolling should be performed. Glitch filters are added to the unrolled design to suppress glitches.

For each system clock frequency, we again evaluate the energy and area of the three equivalent-latency alternatives. Once again, the encryption latencies are 340 ns for SIMON and 175 ns for AES. Results are shown in Figs. 6 and 7. 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 single round is computed in each system clock period, so glitching is not an issue. As the system clock is slowed enough to allow computing multiple rounds within a system clock cycle, the unrolled design without glitch filter becomes inefficient due to glitching. The unrolled design with glitch filters remains highly efficient across all clock frequencies, but pays an ever-increasing price in area (Fig. 5). 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. 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.

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

VI. Conclusions and Future Work

This paper has described a new system to implement loops in low-cost FPGA in an energy-efficient manner. The system automatically unrolls loops to an appropriate depth and inserts latch-based 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 PLL. In the future we plan to test our approach on other loop computations such as sorting and searching algorithms.[1]

References

[1] S. Kerckhof, F. Durvaux, C. Hocquet, D. Bol, and F.-X. Standaert, "Towards green cryptography: A comparison of lightweight ciphers from the energy viewpoint," in *Proc. CHES*, Sep. 2012, pp. 390–407.

[2] S. N. Dhanuskodi and D. Holcomb, "Energy optimization of unrolled block ciphers using combinational checkpointing," in *Proc. RFIDSec*, Nov. 2016.

[3] J. Lamoureux, G. Lemieux, and S. Wilton, "Glitchless: Dynamic power minimization in FPGAs through edge alignment and glitch filtering," *IEEE TVLSI*, vol. 16, no. 11, pp. 1521–1534, Nov. 2008.

[4] S. Huda and J. Anderson, "Towards PVT-tolerant glitch-free operation in FPGAs," in *Proc. FPGA*, Feb. 2016.

[5] C. Ravishankar, J. H. Anderson, and A. Kennings, "FPGA power reduction by guarded evaluation considering logic architecture," *IEEE TCAD*, vol. 31, no. 9, pp. 1305–1318, Aug. 2012.

[6] W. Shum and J. H. Anderson, "FPGA glitch power analysis and reduction," in *Proc. ISLPED*, Aug. 2011.

[7] *Expanding Applications For Low Cost FPGAs*, Lattice Semiconductor, Apr. 2007.

[8] K. Kepa, D. Coburn, J. C. Dainty, and F. Morgan, "High speed optical wavefront sensing with low cost FPGAs," *Measurement Science Review*, vol. 8, no. 4, pp. 87–93, 2008.

[1]This research was supported by NSF/SRC grant CNS-1619558.