Exploiting Multi-Grained Parallelism in Reconfigurable SBC Architectures

Joseph Zambreno, Dan Honbo, and Alok Choudhary
Department of Electrical and Computer Engineering
Northwestern University
Evanston, IL 60208, USA
{*zambro1, dkh301, choudhar*}*@ece.northwestern.edu*

Abstract

In recent years, reconfigurable technology has emerged as a popular choice for implementing various types of cryptographic functions. Nevertheless, an insufficient amount effort has been placed into fully exploiting the tremendous amounts of parallelism intrinsic to FPGAs for this class of algorithms. In this paper, we focus on block cipher architectures and explore design decisions that leverage the multi-grained parallelism inherent in many of these algorithms. We demonstrate the usefulness of this approach with a highly parallel FPGA implementation of the AES standard, and present results detailing the area/delay tradeoffs resulting from our design decisions.

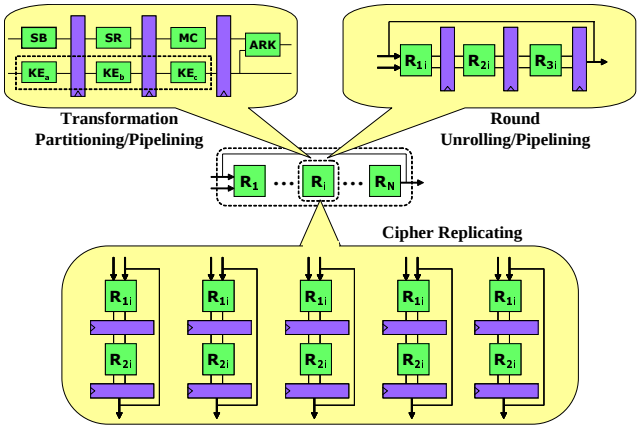

Figure 1. Exploiting parallelism in AES-128E

1. Parallelizing SBC Architectures

Field-Programmable Gate Arrays (FPGAs) are becoming an extremely popular target for implementing cryptographic algorithms. There are various reasons for this increase in attention from the security community. Firstly, since the individual operations required are relatively simple (often just combinational logic), any hardware design can reduce the overhead introduced by an equivalent software implementation. Also, the reconfigurable nature of FPGAs is attractive to cipher designers, as the implementation can be modified after the original time of programming.

When compared to other efforts in implementing Symmetric Block Cipher (SBC) architectures on FPGAs, our work is unique in that we combine previous attempts at exploiting the different types of inherent parallelism. This approach is consequently better suited to modern FPGA devices with large amounts of configurable resources and I/O. By utilizing both types of parallelism, we can draw nearer to the goal of fully maximizing the potential of FPGAs for accelerating this class of applications. Focusing on 128-bit AES encryption [2], in this paper we explore the area, delay, and efficiency tradeoffs inherent in our design decisions and present an FPGA implementation that is capable of achieving throughputs greater than 30 Gbps.

When considering an SBC operating in a non-feedback mode, the resulting algorithm is often embarrassingly parallel. For the purposes of implementation on FPGA hardware, this means that a large input can be easily split across several identical computational units and the output recombined after processing. This optimization is referred to as cipher *replication* (see Figure 1). In theory, given a sufficiently large FPGA this approach should lead to linear increases in throughput, with no negative effects on latency or area efficiency. This is not often the case, since the linear increase in reconfigurable resource usage also greatly complicates the interconnect routing. Also, cipher replication increases the number of I/O resources needed by the design. This idea of replicating block cipher architectures is inspired by Swankoski et al. [3] and is related to the concept of multi-threaded architectures proposed by Alam et al. [1].

Many SBC algorithms at their highest level iterate through round structures. Consequently, it is possible to exploit fine-grained parallelism using classical loop layout

Table 1. AES-128E implementation results for a Xilinx XC2V8000 FPGA

Design Name	Resource Usage			Performance			
	N_{slice}	N_{bram}	N_{iob}	f_{clk} (MHz)	$Tput$ (Gbps)	Lat (ns)	Eff ($\frac{\text{Mbps}}{\text{slice}}$)
`RF1-UF1-PP0d`	1786	**0**	387	75.34	0.96	132.74	0.54
`RF1-UF1-PP1b`	**485**	10	387	108.38	1.39	92.27	2.86
`RF1-UF2-PP0b`	764	20	387	84.63	2.17	**59.08**	2.84
`RF1-UF10-PP1b`	2515	100	**386**	75.09	9.61	133.17	3.82
`RF1-UF10-PP2b`	3765	100	386	117.21	15.00	170.64	**3.98**
`RF1-UF10-PP3d`	16941	0	386	**180.73**	23.13	165.99	1.37
`RF2-UF10-PP3d`	33757	0	643	123.89	**31.71**	242.16	0.94

optimizations in a similar fashion to what is performed in software compilers. *Unrolling* replaces a loop body with N copies of that loop body. Unrolling the rounds makes them highly amenable to *pipelining*, which is a technique that increases the number of blocks of data that can be processed concurrently. Pipelining in FPGA designs can be implemented by inserting registers between the modules that need to operate independently.

Unrolling and pipelining rounds will have no effect on the critical path when compared to the original iterative SBC architecture. In general, this maximum delay will be dependent on the individual round transformations. Fortunately, these sub-modules are also eligible for pipelining. As can be seen in Figure 1, the AES cipher contains several round transformations that can be pipelined, and previous work [4] has shown that the clock frequency can be further improved by *paritioning* the `KeyExpansion` operation across several pipeline stages.

2. Experimental Results

We applied these optimizations to a design of AES encryption using a 128-bit key (AES-128E). This algorithm was implemented using a single VHDL core, with a configuration file to drive preset macros for controlling the round layout and replication and explicit synthesis directives to determine the mapping of S-boxes. To synthesize the designs we used Synplify Pro 7.2.1 from Synplicity, which was configured to target a Xilinx XC2V8000 FPGA, the largest member of the Virtex-II device family. Xilinx ISE 5.2i was used for the place-and-route and timing analysis.

For each design we used these tools to measure the maximum possible clock rate (f_{clk}), the number of utilized slices (N_{slice}), the number of Block SelectRAMs (N_{bram}), and the number of I/O blocks (N_{iob}). From these base statistics we calculated the resultant maximum throughput ($Tput$), latency (Lat), and efficiency (Eff) measured in throughput rate per utilized CLB slice.

A selection of our experimental results can be found in Table 1. Each design is labeled `RF`W`-UF`X`-PP`YZ, where W corresponds to the cipher replication factor, X corresponds to the round unrolling factor, and the Y value specifies the amount of partitioning and pipelining [4]. The Z value specifies the S-box technology mapping in the design, where for $Z = [\text{b}]$ Block SelectRAM is chosen and for $Z = [\text{d}]$ distributed ROM primitives are chosen. For the sake of clarity and brevity, unexceptional results from the set of possible designs were pruned when forming Table 1.

The bolded values in Table 1 represent the designs which are maximal in terms of the displayed area, performance, and efficiency characteristics. The designs that required the least amount of area (see `RF1-UF1-PP1b` needing 485 slices) were those that used had minimal replicating, unrolling, and pipelining. The `RF1-UF2-PP0b` design demonstrates the usefulness of partial unrolling, as it had the lowest latency of all designs at 59.08 ns. The `RF1-UF10-PP2b` design had the highest area efficiency (3.98 Mbps/slice). The most aggressively partitioned and pipelined designs achieved the highest clock speeds, and the `RF1-UF10-PP3d` was overall the fastest at 180.73 MHz. That same cipher design replicated twice (`RF2-UF10-PP3d`) achieved the fastest theoretical throughput, running at 31.71 Gbps.

References

[1] M. Alam, W. Badawy, and G. Jullien. A novel pipelined threads architecture for AES encryption algorithm. In *Proceedings of the International Conference on Application-Specific Systems, Architectures, and Processors*, 2002.

[2] FIPS PUB 197. Advanced Encryption Standard (AES), National Institute of Standards and Technology, U.S. Department of Commerce. available at http://csrc.nist.gov, 2001.

[3] E. Swankoski, R. R. Brooks, V. Narayanan, M. Kandemir, and M. J. Irwin. A parallel architecture for secure FPGA symmetric encryption. In *Proceedings of the Reconfigurable Architectures Workshop*, 2004.

[4] J. Zambreno, D. Nguyen, and A. Choudhary. Exploring area/delay tradeoffs in an AES FPGA implementation. In *Proceedings of the 14th International Conference on Field Programmable Logic and Applications*, 2004.