6

A Study of FPGA Implementations of Data Encryption Algorithms

Abstract

Field programmable gate arrays (FPGAs) are programmable hardware that may be configured after deployment into a system. FPGAs also present a much faster implementation than software. These characteristics make them a considerable candidate for utilization in an encryption system where an efficient and adaptable environment is a necessity. This project will explore the area between FPGAs and encryption algorithms, including implementations, comparisons, and data analysis and discussion.

1 Introduction

Data encryption is a necessarily advancing field as a result of high speed computing development and improved cryptanalysis methods. Cryptologists are continuously working on algorithms to encode and decode valuable information used in banking, online market transactions, medical information systems, and other applications. Within the last decade, field programmable gate arrays (FPGAs) have been increasingly used to implement cryptographic methods. This project proposes to explore multiple encryption algorithms that have been developed over the last few decades; examine implementations of these algorithms on FPGAs; compare the costs and benefits between FPGAs, application specific integrated circuits (ASICs), and software. Although there are multiple options for encryption implementation, including ASICs and software, FPGAs provide a unique balance between the benefits of both. They offer speed where software can not and flexibility where ASICs are lacking.

A good location for the implementation of FPGA encryption is on network nodes where encryption can be a bottleneck for a network. Using the speed and flexibility of FPGAs, a

component can increase its data encryption throughput and be easily updated if its current algorithm needs to be modified or changed completely.

This paper is organized as follows: section 2 will describe background information concerning FPGAs and encryption algorithms; section 3 states the thesis of the paper; section 4 describes the need for an exploration of this topic; section 5 will explain the methods for which this project will be carried out; section 6 details a time table for when the objectives of this project should be completed; section 7 concludes this proposal.

2 Background Information and Prior Work

2.1 Hardware

For this project, a Digilent Basys2 circuit board has been obtained, as can be seen in Figure 1. The Basys2 includes a Xilinx Spartan3E-100 FPGA[9]. The board also contains many I/O features, such as switches, LEDs, PS/2 and VGA ports.

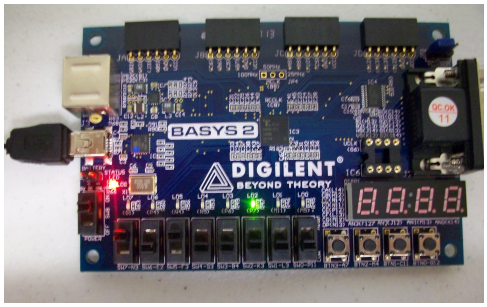

Figure 1: The Digilent Basys2 Board

An FPGA is a matrix of hundreds of configurable logic blocks (CLBs), as shown in Figure 2. These logic blocks are programmable to represent almost any electronic circuit component including logic operators, like AND, OR, and NOT, as well as memory. The main component in most logic blocks is a look up table (LUT). In the Spartan 3, LUTs accept four inputs and represent a truth table for any function of the given inputs [11]. For functions of more than four inputs, multiple LUTs can be used together.

The CLBs connect to a switch matrix which can create multiple types of connections between the logic blocks. I/O blocks surrounds the matrix of CLBs and allows the FPGA to communicate to the outside world.

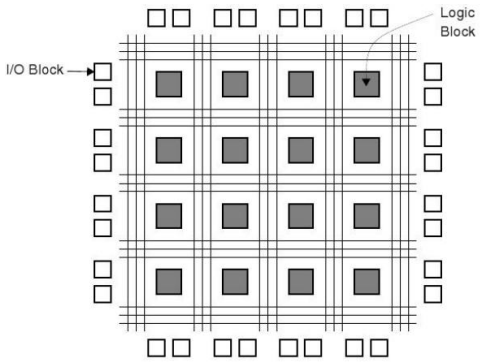

Figure 2: FPGA diagram [2]

2.2 Hardware Description Language

In order to describe the circuit to be performed by the FPGA, a hardware description language (HDL) is used. This project will make use of VHDL (VHSIC (very high speed integrated circuit) hardware description language), one of the most common languages to use for this type of application.

For example, the following line of code describes a simple circuit which takes two inputs, A and B, and turns output C on if both A and B are on or if both A and B are off.

C <= (A and B) or (A nor B);

2.3 Introduction to Cryptography and Encryption

Cryptography is "the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication." [18] Cryptography is a long-studied science, dating back to Julius Ceasar and earlier.

One component of cryptography is encryption, which is the conversion from plaintext, or the original data, to ciphertext, the data in a form only readable to those with certain privileges, using some algorithm or function. The data to be encrypted can include anything from top secret government blueprints, to a customer's credit card number, to a note between two friends.

An algorithm that performs encryption is called a cipher. All three of the algorithms described in this project are symmetric block ciphers. They are symmetric because they

use the same key for both encryption and decryption of data. A key is an input parameter to the cipher that contributes to determining the output. A block cipher is one that takes in a finite block of data as input, as opposed to a continual stream.

3 Thesis

My project will demonstrate the following thesis:

> This project proposes to examine FPGAs and encryption algorithms, allowing both areas to be explored separately and in union through research, implementation, observation, and analysis. The use of this study will allow one to make more informed security decisions and develop better security techniques.

This project intends to deliver the following items:

- A survey of multiple encryption algorithms including xTEA, DES, and Blowfish;
- A study of FPGAs: their history, architecture, designs, and comparisions to ASICs;
- A report on implementations of stated encryption algorithms using FPGAs;
- A thorough results analysis and discussion of the implementations.

4 Motivation

Currently, the National Institute of Standards and Technology (NIST) is performing a search to find a new algorithm to obviate SHA-2 (a secure hash algorithm), the current national standard. A hash function is one that can convert a data block, containing a message, to a hash value. This hash value should be unique to the message and it should be difficult to find the original message given its hash value. For this reason, hash functions are very useful for integrity and authentication for multiple resources such as files and passwords. In order to offset any possible security vulnerabilities in SHA-2, a search for a new hashing standard, to be called SHA-3, is currently underway.

The NIST has narrowed its search to fourteen semi-finalists. One of these is an algorithm created by Bruce Schneier et al. called Skein [3]. In a September 1, 2010 blog post [25],

Schneier asked for help in testing implementations of Skein on both FPGAs and ASICs. He claims that results produced from his team have performed much better than those from others. His goal is to understand why there are such differences in performance and how to create better implementations.

Although this project will not explore this challenge directly, it represents work that will lead to a better understanding of encryption alogorithms, a close relative to hashing functions, and their FPGA implementations.

5 Implementation

5.1 Deliverables

A survey of encryption algorithms will be created by studying background material such as [19] and [22]. These sources will aid in the development of a thorough, yet compact explanation of the history and design of selected encryption algorithms. A separate survey including details of the history and design of FPGAs will be constructed through previous work as well as experiences with working on an FPGA throughout the duration of the project. During the course of the implementation and analysis process of the project, a journal will be kept to document experiences and difficulties encountered. This journal will be used as a main source of information and feedback in order to accurately formulate a report on the implementations and findings. Data collected from the running of the implementations will be analyzed and discussed in order to achieve a better understanding of how well the studied algorithms can maintain their effectiveness and efficiency, as well as the ability of FPGAs to serve as an environment for these algorithms.

5.2 FPGA Design and Implementation

In order to design and configure the FPGA, two tools will be used. Xilinx's ISE WebPack software allows one to design on FPGA using a HDL. WebPack then automatically performs many of the tasks necessary to convert this code into a bit file, such as synthesis, map, and place and route. Digilent's Adept software then transfers the bit file onto the device for configuration.

Below is the VHDL code for a binary adder. C1, C2, and C3 are the 2^2, 2^1, and 2^0 position in the sum, respectively. A1 and B1 are the 2^0 positions in the addends, while A2 and B2 are the 2^1 positions in the addends.

```
entity circuit1 is
    Port ( A1 : in  STD_LOGIC;
           B1 : in  STD_LOGIC;
           A2 : in  STD_LOGIC;
           B2 : in  STD_LOGIC;
           C1 : out  STD_LOGIC;
           C2 : out  STD_LOGIC;
           C3 : out  STD_LOGIC);
end circuit1;
}
architecture Behavioral of circuit1 is
begin
C3<=(A1 xor B1);
C2<=(A1 and B1) xor (A2 xor B2);
C1<=((A1 and B1) nand (A2 xor B2)) nand (A2 nand B2);
end Behavioral;
```

Figure 1 displays the Basys2 board running this binary adder (showing $10_2+10_2 = 100_2$).

5.3 Encryption Algorithms

One algorithm to be studied in the course of this project is the Extended Tiny Encryption Algorithm (xTEA). The xTEA algorithm was design by Roger Needham and David Wheeler to be quick and easy to implement in any environment [19]. xTEA is a block cipher, which is one that transforms a plaintext block of a given size to a ciphertext block of the same size, which is 64 bits in xTEA. During execution of the xTEA algorithm, each of these blocks is divided in two, called y and z. A new value for each half is then computed as follows:

- a permutation function, represented as f in Figure 3:

$$f(z) = (z \ll 4 \oplus z \gg 5) + z$$

where $\oplus$ represents the XOR operation and $\ll$ and $\gg$ represent a left and right shift, respectively.

- a subkey generation funtion, represented as $Keygen$ in Figure 3:

$$sum + k(sum)$$

where $k(sum)$ is a function that chooses a block of the encryption key depending on specific bits of sum.

The variable sum is initialized at 0 and at every round of the cipher is incremented by the constant DELTA.

- the new value of each half is the result of an XOR of these two functions.

The finding of both of these values is considered one round of the cipher. Typically, xTEA completes 32 rounds as part of the algorithm.

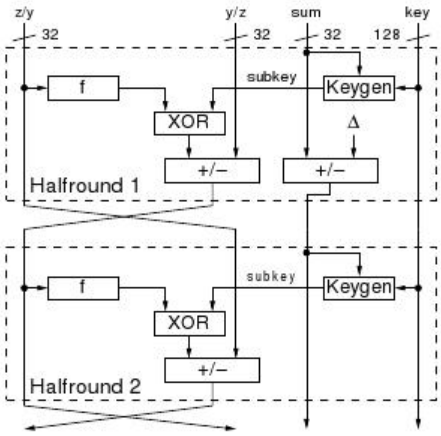

Figure 3: A high level diagram of xTEA [14]

Here an implementation of the algorithm in the C programming language is supplied from [19]. Additional comments were added for clarification.

```
int tean(long * v, long * k, int N)
{
        /* Set up: the data input, v, is split into two halves, y and z */
        unsigned long y=v[0], z=v[1], DELTA=0x9e3779b9 ;
        if (N > 0) { /* encoding */
                unsigned long limit=DELTA*N, sum=0 ;
                while (sum!=limit) {
                        /* first half of round with permutation and subkey transpose
                           generation functions */
                        y+= (z << 4 ^ z >> 5) + z ^ sum + k[sum & 3];
                        sum+=DELTA;
                        /* second half of round */
                        z+= (y << 4 ^ y >> 5) + y ^ sum + k[sum >> 11 & 3];
                }
        }
        /* decoding source code omitted */
}
```

An example of xTEA's implementation in VHDL is presented here, corresponding to one half round:

```
-- y's Permutation function
zLeftShift <= std_logic_vector(unsigned(z) sll 4);
zRightShift <= std_logic_vector(unsigned(z) srl 5);
zShiftXOR <= zLeftShift xor zRightShift;
yPermFun <= zShiftXOR + z;

-- y's Subkey generation function
kSum1_0 <= std_logic_vector(k(to_integer(unsigned(sum
        and "00000000000000000000000000000011"))));
ySumFun <= sum + kSum1_0;

-- Calculate y
yXOR <= yPermFun xor ySumFun;
y <= yTemp + yXOR;
}
```

This code contains multiple intermediate values for readablility.

Other existing algorithms will be studied, specifically DES and Blowfish. DES is also a block cipher which became the United States standard in 1977 [5] and has become one of the most commonly used encryption algorithms in a wide variety of applications, such as ATMs and smart cards. The popularity of DES makes it a necessity to study in a survey of encryption algorithms. The Blowfish encryption algorithm was also selected for study [24]. It is a potential challenger to the Advanced Encryption Standard (AES), the current national standard. Much of the motivation behind choosing Blowfish is the lack of prior work that has been done with this algorithm, especially implementation analysis.

These other algorithms to be implemented will also be constructed using a high level language. These implementations will then be reproduced with VHDL for use on a FPGA.

5.4 Metrics

There are multiple metrics that have been used to judge the "success" of implementations. In keeping with the common metrics used throughout other research pertaining to algorithm implementations, this project will likely use measures such as clock cycles, area, and throughput.

The clock cycles metric measures the number of clock cycles that passed during the execution of specific amount of instructions, such as for a single instruction, one round of an algorithm, or the entire algorithm.

The area measure pertains to the amount of physical resources on the FPGA that are used during the implementation of a specific algorithm. This can be measured in slices, or configurable logic blocks(CLBs), of the FPGA. This measure is important in not only judging the amount of resources that are needed for the application, but the amount that will not be available to other applications that may be running on the same FPGA.

Throughput is the amount of work performed over a given unit of time. For example, if an implementation processes 15Mb of data in 3 seconds, then its throughput = 5 Mbps (Mb per second).

6 Research and Writing Timetable

Below is a tentative timetable for this project:

- September [redacted]
  - Revise thesis proposal;
  - Experiment with FPGA using simple implementations;
- October [redacted]
  - Develop XTea implementation;
  - Implement XTea;
  - Start writing two chapters;
- November [redacted]

  - Implement XTea;
  - Collect XTea data;
  - Develop Blowfish implementation;
  - Continue writing two chapters;
- December [redacted]
  - Finish two chapters;
  - Implement Blowfish;
  - Collect Blowfish Data;
- January [redacted]
  - Finish Blowfish implementation;
  - Collect Blowfish data;
  - Continue writing chapters;
  - Develop DES implementation;
- February [redacted]
  - Implement DES;
  - Collect data for DES;
  - Data analysis;
  - Continue writing chapters;
- March [redacted]
  - Finish all implementations and data collections;
  - Continue data analysis and discussion;
  - Continue writing chapters;
- April [redacted]
  - Finish writing;

- Revise, revise, revise;
- Prepare and give oral defense;

7 Conclusion

This paper proposes a project that intends to explore FPGAs, encryption algorithms, and the space connecting the two. The resulting deliverables will compare, discuss, and describe the characteristics and reasons for and against implementing these algorithms on FPGAs. It is hoped that this project will help to make better security decisions, or at least make individuals and organizations more security technology aware.

References

[1] Peter J. Ashenden. *The Student's Guide to VHDL*. Morgan Kaufmann, 2008.

[2] Stephen Brown, , Stephen Brown, and Jonathan Rose. Architecture of fpgas and cplds: A tutorial. *IEEE Design and Test of Computers*, 13:42–57, 1996.

[3] et al. Bruce Schneier. The skein hash function family, 2009.

[4] Katherine Compton and Scott Hauck. Reconfigurable computing: a survey of systems and software. *ACM Comput. Surv.*, 34(2):171–210, 2002.

[5] D. Coppersmith. The data encryption standard (des) and its strength against attacks. *IBM J. Res. Dev.*, 38(3):243–250, 1994.

[6] Li Fu and Ming Pan. A simplified fpga implementation based on an improved des algorithm. *Genetic and Evolutionary Computing, International Conference on*, 0:227–230, 2009.

[7] Ted Huffmire, Brett Brotherton, Nick Callegari, Jonathan Valamehr, Jeff White, Ryan Kastner, and Tim Sherwood. Designing secure systems on reconfigurable hardware. *ACM Trans. Des. Autom. Electron. Syst.*, 13(3):1–24, 2008.

[8] Russell Tessier Ian Kuon and Jonathan Rose. Fpga architecture: Survey and challenges. *Foundations and Trends in Electronic Design Automation*, 2(2):135–253, 2008.

[9] Digilent Inc. Digilent basys2 board reference manual, 2009. `http://digilentinc.com`.

[10] Xilinx Inc. Spartan-3 generation configuration user guide, 2009. `http://xilinx.com`.

[11] Xilinx Inc. Spartan-3e fpga family: Data sheet, 2009. `http://xilinx.com`.

[12] Xilinx Inc. Spartan-3 generation fpga user guide, 2010. `http://xilinx.com`.

[13] Yang Jun, Li Na, and Ding Jun. A design and implementation of high-speed 3des algorithm system. *Future Information Technology and Management Engineering, International Seminar on*, 0:175–178, 2009.

[14] Jens-Peter Kaps. Chai-tea, cryptographic hardware implementations of xtea. In *INDOCRYPT '08: Proceedings of the 9th International Conference on Cryptology in India*, pages 363–375. Springer-Verlag, 2008.

[15] Franois Koeune, Gael Rouvroy, Franois-Xavier Standaert, Francois xavier St, Jean-Jacques Quisquater, Jean pierre David, and Jean didier Legat. An fpga implementation of the linear cryptanalysis. In *In 12th International Conference on Field Programmable Logic and Applications (FPL 2002*, 2002.

[16] Patrick Lysaght and P. A. Subrahmanyam. Guest editors' introduction: Advances in configurable computing. *IEEE Des. Test*, 22(2):85–89, 2005.

[17] F. Macé, F.-X. Standaert, and J.-J. Quisquater. Fpga implementation(s) of a scalable encryption algorithm. *IEEE Trans. Very Large Scale Integr. Syst.*, 16(2):212–216, 2008.

[18] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[19] Roger M. Needham and David J. Wheeler. Tea extensions. *Computer Laboratory*, 1997.

[20] Ersin Oksuzoglu and Erkay Savas. Parametric, secure and compact implementation of rsa on fpga. *Reconfigurable Computing and FPGAs, International Conference on*, 0:391–396, 2008.

[21] Christof Paar. *Understanding Cryptography*. Springer, 2009.

[22] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 26(1):96–99, 1983.

[23] Laurent Sauvage, Sylvain Guilley, and Yves Mathieu. Electromagnetic radiations of fpgas: High spatial resolution cartography and attack on a cryptographic module. *ACM Trans. Reconfigurable Technol. Syst.*, 2(1):1–24, 2009.

[24] Bruce Schneier. Description of a new variable-length key, 64-bit block cipher (blowfish). In *Fast Software Encryption, Cambridge Security Workshop*, pages 191–204. Springer-Verlag, 1993.

[25] Bruce Schneier. More skein news, 2010. `http://www.schneier.com/blog/archives/2010/09/more_skein_news.html`.

[26] David Wheeler and Roger Needham. Tea, a tiny encryption algorithm. pages 97–110. Springer-Verlag, 1995.

[27] Thomas Wollinger, Jorge Guajardo, and Christof Paar. Security on fpgas: State-of-the-art implementations and attacks. *ACM Trans. Embed. Comput. Syst.*, 3(3):534–574, 2004.

[28] Anderson Cattelan Zigiotto and Roberto d'Amore. A low-cost fpga implementation of the advanced encryption standard algorithm. In *SBCCI '02: Proceedings of the 15th symposium on Integrated circuits and systems design*, page 191. IEEE Computer Society, 2002.