SCHEMES TO REDUCE POWER IN FPGA IMPLEMENTATIONS OF THE ADVANCED ENCRYPTION STANDARD

By

JASON DANIEL VAN DYKEN

A thesis submitted in partial fulfillment of
The requirements for the degree of

MASTER OF SCIENCE IN COMPUTER ENGINEERING

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

December 2007

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of JASON DANIEL VAN DYKEN find it satisfactory and recommend that it be accepted.

_____

Chair

_____

_____

Acknowledgements

The author would like to thank the following people for their support, encouragement and advice throughout this research and academic pursuit. Without them this achievement would not have been possible.

Dr. José Delgado-Frias

Dr. Sirisha Medidi

Gordy Hayes

Keith and Mary Van Dyken

Crystal Van Dyken

SCHEMES TO REDUCE POWER IN FPGA IMPLEMENTATIONS OF THE ADVANCED ENCRYPTION STANDARD

Abstract

By Jason Daniel Van Dyken, M.S.
Washington State University
December 2007

Chair: José G. Delgado-Frias

Since its introduction in 2001 the Advanced Encryption Standard has been the subject of vast amounts of research in such areas as speed of encryption, size of encryption device, ultra low power encryption, and algorithm integrity. One area that has been relatively neglected is the area of power conscious encryption. This focus is intended to reduce the average power consumption of an encryption core while maintaining a similar level of performance so that it can be easily and reliably integrated into systems with varying requirements.

In this thesis three designs will be proposed to achieve this goal of power efficient encryption. This includes a standard data in/data out design, a key storage design, and a multistage design. These designs along with two reference designs, from NIST and the Open Cores project, will be analyzed to determine their power consumption rates utilizing Xilinx XPower and Mentor Graphics' ModelSim software packages. Once the preliminary analysis has been completed, the architecture of the designs will be examined along with the effects

of FPGA choice and clock rate to obtain a better understanding of how to satisfy the stated goal and to further optimize the proposed designs to meet that goal.

When analyzing the proposed designs, it was shown that the most promising design used 201 mW or 20.7% less power then the best performing reference design when using a Virtex II FPGA. Similarly when using a Spartan 3 FPGA the results showed the proposed design used 37.9 mW or 18% less power than the best performing reference design. Additionally by reducing the clock rate of an FPGA the design's power can be reduced to less than six percent of the chip's total required power, with 94% of the power being allocated to satisfy the quiescent power requirements. The proposed designs' individual component power analysis also showed that no component consumed more power than was expected, based on the individual component's complexity.

TABLE OF CONTENTS

Acknowledgements iii

Abstract iv

List of Tables ix

List of Figures x

Chapter

1. Introduction 1

    1.1. Options for Implementation 2

    1.2. Purpose and Thesis Organization 4

2. Advanced Encryption Standard 6

    2.1. AES Historical Background 6

    2.2. Basic Components of AES 8

        2.2.1. Terms and Concepts 9

        2.2.2. Byte Substitution 10

        2.2.3. Shift Rows 11

        2.2.4. Mix Columns 11

        2.2.5. Add Round Key 13

        2.2.6. Key Expansion 13

    2.3. Component Assembly 14

2.4. Cipher Organization 17

3. Implementation 20

3.1. Design Medium 20

3.2. Design Choices 22

3.2.1. Data Path Width 23

3.2.2. Design Organization 24

3.2.3. Substitution Box Structure 26

3.2.4. Mix Column Structure 26

3.2.5. Number of Data Input and Output Pins 28

3.2.6. Key Input, Storage, and Generation 29

3.3. Current Designs 31

3.3.1. NIST Design 32

3.3.2. Open Core Design 35

3.4. Proposed Designs 38

3.4.1. Standard Design 38

3.4.2. Hard Key Design 42

3.4.3. Dual Stage Design 46

3.5. Design Verification 48

4. Performance Evaluation 51

4.1. Approach 51

4.2. Chosen Methodology 54

4.3. Comparison Analysis 59

4.4. FPGA Choice Analysis 64

4.5. Speed and Power Analysis 67

4.6. Component Power Analysis 71

4.6.1. Substitution Box 72

4.6.2. Shift Rows 72

4.6.3. Mix Columns 73

4.6.4. Add Round Key 74

4.6.5. Key Expansion 74

5. Concluding Remarks 77

5.1. Discussion 78

5.2. Future Work 79

Bibliography 82

LIST OF TABLES

2.1 Key size and the number of rounds required to complete an encryption..... 9

2.2 SBOX lookup table 11

2.3 Round constants listed by round..... 15

3.1 Design verification 50

4.1 Power estimation for different amounts of test vectors..... 57

4.2 Comparative analysis for designs at 25 MHz..... 61

4.3 Comparative analysis for designs at 5 MHz 62

4.4 FPGA power analysis for Hard Key design 66

4.5 Speed vs. power analysis for Hard Key design on Spartan 3..... 69

4.6 Speed required to fully utilize bandwidth with encrypted packets..... 70

4.7 Component power consumption analysis 72

4.8 Key generation power estimation for component analysis..... 75

List of Figures

2.1 Data input handling and organization by the AES algorithm 10

2.2 The SBOX affine transformation 11

2.3 The shift rows operation 12

2.4 The mix column operation 13

2.5 The add round key operation 14

2.6 The key expansion process 15

2.7 Round block organization 16

2.8 Key scheduler example 1 17

2.9 Key scheduler example 2 18

2.10 AES cipher structure 19

3.1 Varying data path widths 24

3.2 Optimized mix column structure 27

3.3 Multiply by 2 optimization 27

3.4 NIST design interface 33

3.5 Waveform to initiate an encryption for the NIST design 34

3.6 Block diagram of the NIST design's architecture 35

3.7 Open Core design interface 36

3.8 Waveform to initiate an encryption for the Open Core design 37

3.9 Block diagram of the Open Core design Architecture 38

3.10 Standard design interface 40

3.11 Waveform to initiate an encryption for the Standard design 41

3.12 Block diagram of the Standard design architecture..... 42

3.13 Hard Key design interface 43

3.14 Waveform to initiate an encryption for the Hard Key design 44

3.15 Block diagram of the Hard Key design architecture 45

3.16 Dual Stage design interface 46

3.17 Block diagram of the Dual Stage design architecture..... 47

3.18 Waveform to initiate an encryption for the Dual Stage design 49

4.1 Possible power consumption models 58

4.2 Total power analysis for 100 vectors at 25 MHz 60

4.3 Total power analysis for 100 vectors at 5 MHz 62

4.4 Logic power analysis for 100 vectors at 25 MHz..... 63

4.5 Hard Key power consumption for different FPGAs..... 66

4.6 Power consumption versus clock rate analysis 68

4.7 Power consumption versus clock cycle time 69

4.8 Component power consumption for Hard Key design..... 71

5.1 Possible circular key storage structure..... 80

Chapter 1

Introduction

With the advancement of modern technology, communications have not only become the norm, but have become a necessity. Examining the many modern devices that historically have not utilized communications, but have begun to adopt and rely on this technology shows this. Examples include modern appliances that can notify you if they need to be serviced and high-definition movie players providing extra content and system updates. An additional area where improvements in communication technology have been applied is in the development of sensor networks, which have quickly adopted the growing wireless technologies to provide information about a given area without having to physically be there.

While all of these devices are continually expanding the uses for communications, one area that has not always been as thoroughly examined is the security of the

communications. The most reliable and developed way to secure communications is through the encryption of the packet data, however most of the research being devoted to encryption is aimed at having the fastest encryption device possible. This typically means that most of the developed encryption devices are not ideologically congruent with the needs and architectures of the growing number of basic systems requiring secured communications. This helps to emphasize the need that the information in this study tries to fulfill, which is how best to build an encryption device that is power efficient while not dismissing its speed of encryption. The encryption standard that will be used throughout this study is the advanced encryption standard (AES), which is formally introduced in the next chapter.

1.1 Options for Implementation

When implementing a system for encryption there are typically two choices, which are to build software or hardware based systems. A software based system is typically easy to implement, because a high level language may be used, with optimizations being dependent on the compiler. This option is the most convenient since additional hardware is not needed for the device to encrypt its communications. The downfall of software based encryption is that it is slow compared to hardware and requires the system to have more memory to store the code and lookup tables required to complete the encryptions. Additionally some devices are built around a special purpose processor

and may not be suitable for carrying out encryptions.

The alternative to software based encryption is to utilize a hardware based encryption core. There are two main options for building the core, the first of which is an application-specific integrated circuit (ASIC). ASICs allow for both high optimization and fast performance through custom optimizations where both the design speed and power consumption may be considered. The problem with ASICs though is that there is an extremely large overhead when it comes to design and fabrication, because the silicon must be laid out and a chip fabricated for physical testing to be possible.

The other hardware option for design implementation is to use field programmable gate arrays (FPGAs), which offer a balanced alternative to both ASICs and software based implementations. While FPGA implementations are faster than software, they are typically slower than ASICs, because the designs are mapped into a series of look up tables that emulate logic gates. The primary benefit of FPGAs is that they allow for cost effective and fast development, since the devices are programmable and fully reconfigurable. With this in mind the focus of this study is geared toward FPGA development, because of the ease of development and that it can be used as a test bed for a future ASIC design if it was to become necessary.

1.2 Purpose and Thesis Organization

As previously mentioned, when looking at the research that has been presented for encryption cores the vast majority seems to focus on developing high speed cores without regard to the device's power consumption, this makes it irresponsible to use the devices in power limited systems or devices which do not need high throughput. As a result it is the goal of this study to investigate how to design a power conscious encryption core that does not severely hamper the devices performance on an FPGA. With the purpose of this thesis now clearly stated, the rest of this section will be dedicated to outlining what the remaining chapters will cover and how the thesis is organized.

Chapter two will serve as an introduction to the AES algorithm. The history of the algorithm and some basic terms will first be covered. This will be followed by a thorough description of the basic building blocks of the algorithm, and how they are assembled to make up an AES encryption device.

Chapter three will begin with a discussion of the major design choices that were made that have affected each of the proposed designs. This will be followed by formal introductions to the chosen reference designs. Lastly the proposed designs will be introduced, which will include explanations about what sets them apart from the other designs, and the reason each design was made.

Chapter four focuses on the comparative analysis of the designs. The choice of how best to analyze the designs will first be discussed. The remainder of the chapter will then be dedicated to the results of the different analyses, including why each analysis was completed, as well as what its results were.

Chapter five is the conclusion of the thesis and will begin with a discussion of the analytical results from chapter four. This will include what was learned from the research, and how it could be applied. The chapter will then conclude with a description of what direction future research should take to best use the knowledge gained through this research.

Chapter 2

Advanced Encryption Standard

Before the designs can be introduced it is important to understand what encryption method was chosen, why it was chosen, and how it functions. In this chapter the Advanced Encryption Standard (AES) will be discussed. This will begin with a history of AES, and why it is being used. The chapter will then move to the basic components of the algorithm, describing how they function and their purpose. Finally the Algorithm's assembly will be described in terms of how the basic components are connected.

2.1 AES Historical Background

To properly introduce the AES cipher it is first important to understand why it was developed. This is because to understand why the AES algorithm is designed the

way it is, you need to know what it was trying to improve upon and the problems it was trying to correct. During the mid 1990's the dominant private key encryption algorithm was the Data Encryption Standard (DES) developed by IBM in mid 1970's. DES was originally designed to be used for any commercial application requiring privacy, but as computer processing power continued to increase a number a problems with the DES algorithm were identified. These problems include small block sizes (64-bits), poor architecture for software implementation and the biggest problem for any encryption algorithm, susceptibility to attacks through brute force, linear cryptanalysis, and differential cryptanalysis. By the late 1990's the National Institute for Standards and Technology (NIST) had advised that only the modified version of DES known as triple DES (3DES), due to its use of 3 keys and 3 rounds of DES encryption, be used for legacy systems and applications requiring security.

During the time that 3DES algorithm was being finalized and published, NIST also put out a call for submissions for what would become AES. In the request for submissions several key criteria were established for how the submissions would be judged, which included that it must be a symmetric cipher, be able to handle keys of at least 128-bits, efficient in both hardware and software, low memory requirements, and 128-bit data blocks. With these guiding factors it was also imperative that the algorithm provide a high level of general security by means of its output seeming random, while being based on sound mathematical principals and operations. During the first round of

evaluations 15 algorithms were considered and narrowed to 5 possible algorithms. In 2001 NIST published FIPS 197 establishing the Rijndael algorithm, developed by Daemen and Rijmen as the AES cipher.

Since its first introduction in 2001, AES has become the most widely used private key cryptographic algorithm, and in 2003 was given approval for use in encrypting classified material of the U.S. government. It is because of this wide spread acceptance of the AES cipher and the fact that the only attack identified as being effective in weakening the security of the ciphertext is a side channel attack, that this algorithm has been chosen for use in this research.

2.2 Basic Components of AES

The basic structure of the AES cipher is a series of repeating operations, which when grouped together are termed rounds. While mentioned in the previous section that the cipher had to be able to handle 128-bit keys the accepted AES cipher supports keys of 128, 192, and 256 bits, by increasing the number of rounds during an encryption or decryption, table 2.1 shows the correlation between the number of rounds in the encryption process and the key size. Before the exact structure of the cipher can be explained it is first important to understand the basic functions of the cipher and their purpose, as well as some of the basic terms and concepts relied upon for the AES cipher.

Table 2.1: Key Size and the number of rounds required to complete an encryption

Key Size (bits)	Rounds
128	10
192	12
256	14

2.2.1 Terms and Concepts

To fully understand the components that make up the AES cipher a few terms must first be explained. The first term that will be covered is adding, because it does not follow the standard mathematical definition of addition. In cryptography the term addition is used in reference to the exclusive or (XOR) function. The reason for this is because in digital system the size of stored values is limited and cannot expand to account for any carry out. The solution to this problem was to treat the numbers as polynomial coefficients and when two polynomials are added in the set of Z_2, the adding operation is identical to the XOR function [1].

The next term that must be explained is the state, and this is a reference to how the AES cipher organizes data. The basic structure of the state is a four by four matrix of 8-bit values initially populated by the incoming data block as shown in figure 2.1. The primary reason why the data block is organized like this is so that embedded system and

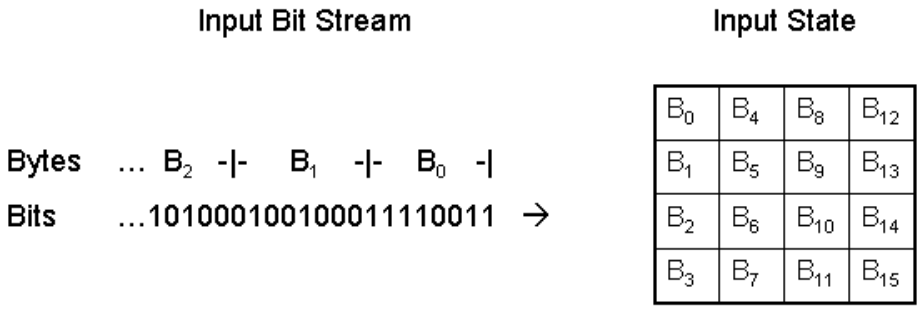

Figure 2.1: Data input handling and organization by the AES algorithm

processing constrained systems, which typically use 8 or 16-bit microcontrollers, can implement the AES algorithm.

2.2.2 Byte Substitution

The substitution transformation is a non-linear byte substitution [2] and is the most mathematically complex transformation of the cipher transformations. The first step of the substitution is to take the multiplicative inverse of the byte in the finite field of 2^8, with the byte value {00} mapped to itself. An affine transformation is then performed, which is shown in figure 2.2. This process is not easily implemented in hardware and as a result is typically turned into the lookup table shown in table 2.2. The combination of the multiplicative inverse and affine transform operations or the look up table is usually termed SBOX, as an abbreviation.

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Figure 2.2: The SBOX affine transformation

Table 2.2: SBOX lookup table

		y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
x	**0**	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

2.2.3 Shift Rows

The shift rows transform is a straightforward transformation involving circularly left shifting the rows of the state by varying amounts. This operation is shown in figure 2.3. As is described in the figure the rows of the state matrix are circularly shifted left by their index value, which means that the first row (index zero) is not shifted, the second row (index one) is shifted left one, and so on. The main purpose of this is to further mix the original data together so that when subsequent blocks perform transformations requiring more that one 8-bit value the inputs are not always based on the same original data.

2.2.4 Mix Columns

The mix column transformation is designed to scramble the contents of a single column of the state. This is accomplished by treating each column as a four-term polynomial [2] and carrying out the matrix multiplication shown in figure 2.4. This is

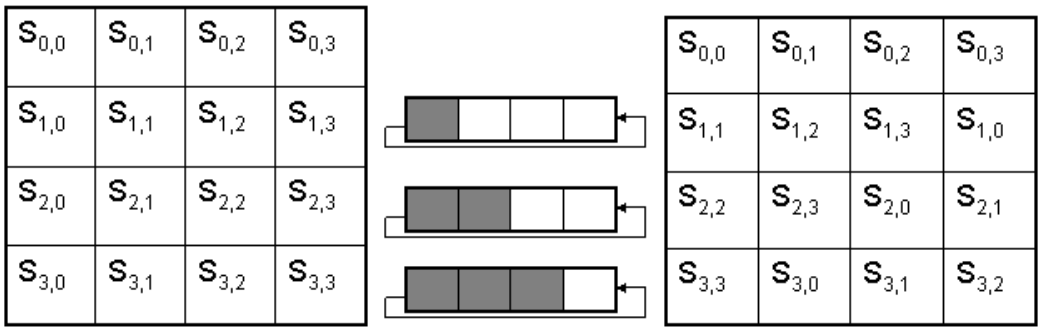

Figure 2.3: The shift rows operation

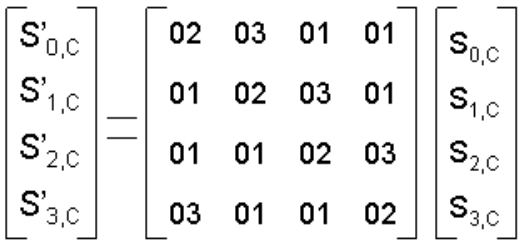

Figure 2.4: The mix column operation

the only transformation that requires more than one element of the input state to determine another element of the state and as a result is largely responsible for the scrambling of the state data, which contributes significantly to the level of security in the cipher.

2.2.5 Add Round Key

The add round key transformation is a very simple operation and is where the encryption key is used to further disassociate the state from the original data block. The key expansion block, which will be described next, generates a 128-bit round key that is managed by a key scheduler and outputs the current round key, which added to the current state, as shown in figure 2.5. This transformation is what makes decryption without the encryption key virtually impossible.

2.2.6 Key Expansion

The key expansion transformation is the largest transformation in the cipher. A block diagram of the transformation is shown if figure 2.6. When a key is sent to the key

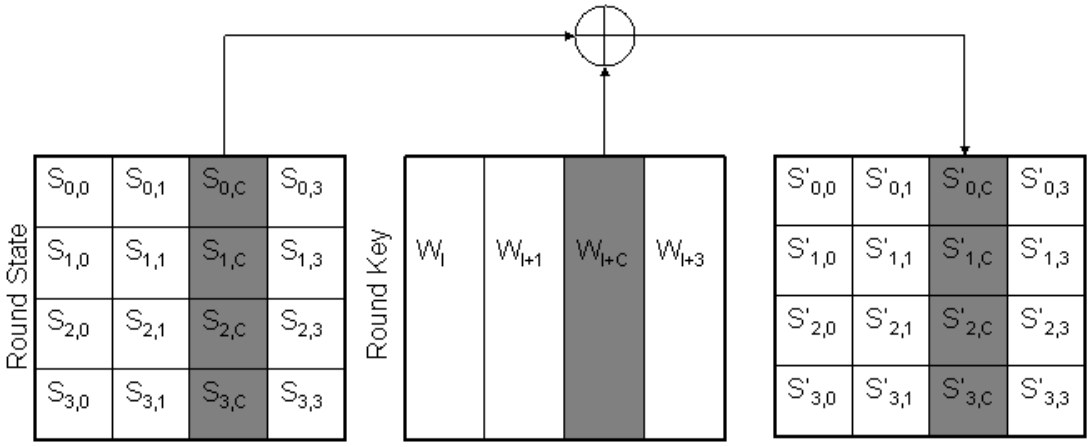

Figure 2.5: The add round key operation

expansion block the first thing that happens is that the values in the last key state column are circularly shifted up once, and transformed by the previously described substitution transform. The first row value also is added with a round constant, which is unique to each round and outlined in table 2.3. The result is then added with the first column of the original key state to generate the first column of the current round key state. Adding the corresponding column from the original key state and the previous column from the new key state then generates the final three columns.

2.3 Component Assembly

Now having a firm understanding of the basic transformations the overall organization of the cipher can be described, including how the components are connected. The first step to doing this is to describe the most used and depended upon

structure in the cipher, which is the round block and is shown if figure 2.7. The round block is the repeated structure that handles most of the work during the encryption

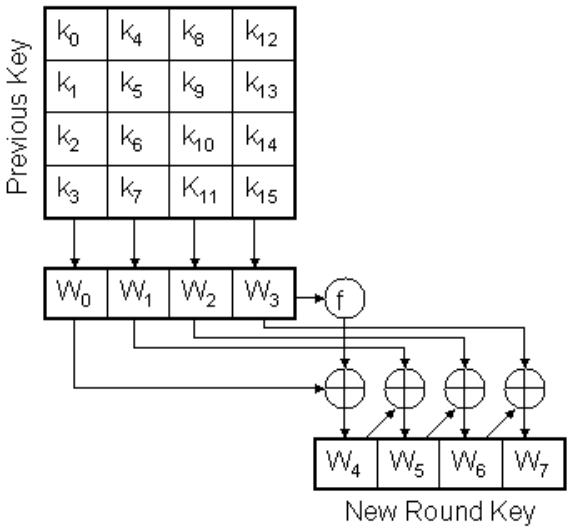

Figure 2.6: The key expansion process

Table 2.3: Round Constants listed by round

Round	Round Constant (Hexadecimal)
1	01
2	02
3	04
4	08
5	10
6	20
7	40
8	80
9	1B
10	36

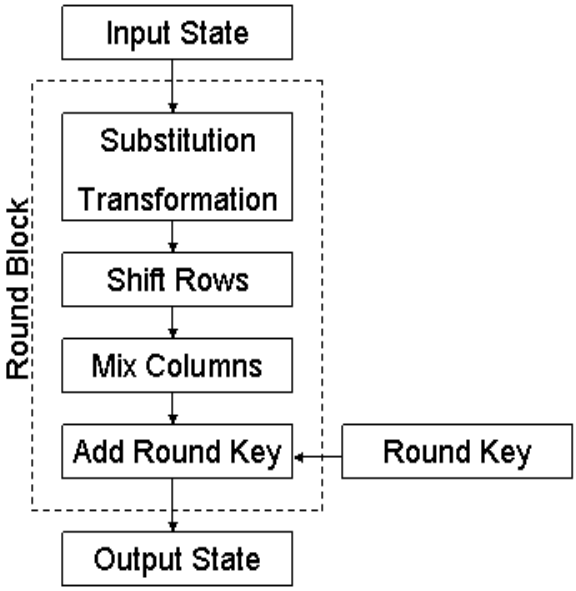

Figure 2.7: Round block organization

process. When a state is passed to a round block the first thing that happens is the substitution transformation is applied to the state. The state is then transformed by the shift rows transformation. From the shift rows transformation the state is sent to the mix columns transformation, and finally to the add round key transformation, where the unique round key is obtained from the key scheduler for the addition.

As mentioned before there needs to be some form of a key scheduler to manage the key generation and ensure that the correct round keys are available at the appropriate time. This component is not explicitly laid out in AES, which gives the designer some flexibility in deciding how to ensure proper key availability, as well as key generation timing. One way in which this has been accomplished is to include a series of registers for storing the individual round keys generated by the key expansion transformation and

output the appropriate key when needed, as shown in figure 2.8. Another option is to use the key expansion transformation to generate the needed round key while the round block is completing the first three transformation, so that it is ready for the add round key transformation as shown in figure 2.9.

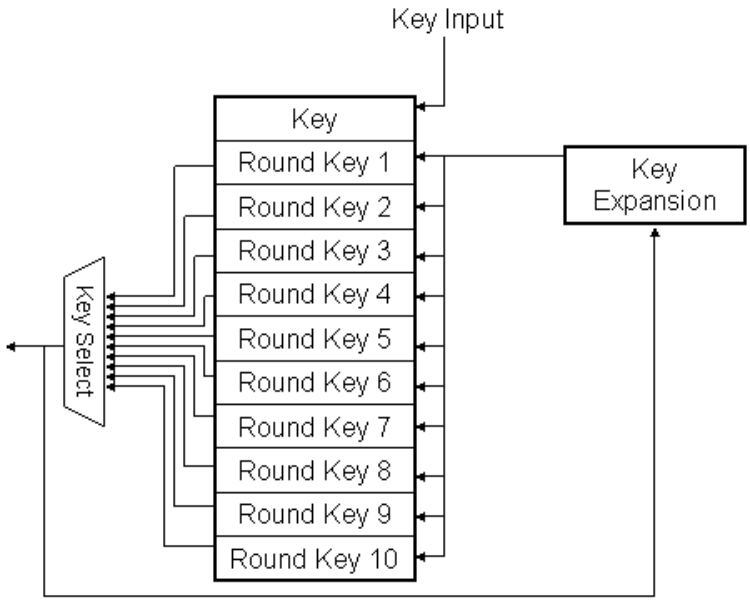

Figure 2.8: Key scheduler example 1

2.4 Cipher Organization

While the most critical elements of the cipher have been laid out it is now time to discuss the overall organization of the AES cipher, which has been diagramed in figure 2.10. When the key and plaintext are set to be encrypted the first thing that happens is the original key and plaintext are added together to form the initial state, which is sent to the round block component while the key is sent to the key scheduler system. The state is sent to nine consecutive round blocks while the key scheduler ensures that the proper

round keys are available to for each round block. After the state has been processed by the nine round blocks it is then sent to a modified round block for the last round of processing. This modified round block is identical to the standard round block except that it does not contain a mix column transformation to make the decryption process easier.

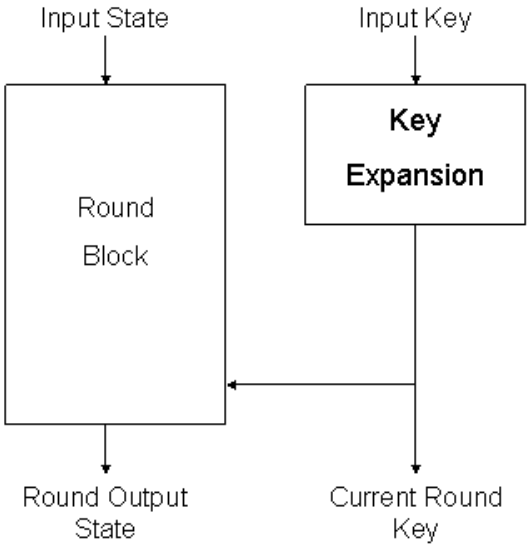

Figure 2.9: Key scheduler example 2

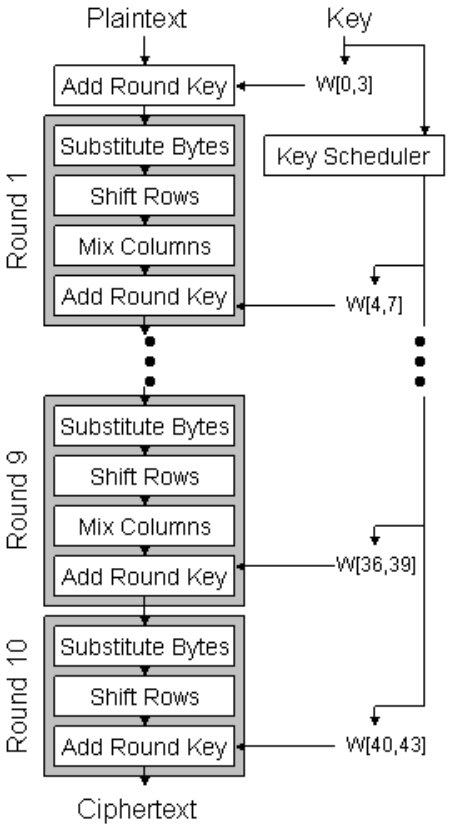

Figure 2.10: AES cipher structure

Chapter 3

Implementation

In this chapter the choices made in how to implement the designs will first be discussed. This will be followed by a discussion of the critical design decisions and finally a formal introduction of the chosen reference designs and the proposed designs themselves. The chapter will conclude with a brief description of the process used to verify that the designs functioned according to the AES guidelines.

3.1 Design Medium

To implement a design on a Xilinx FPGA there are typically 3 methods for doing so, all of which are contained in the ISE development studio. The first method is through a block diagram design utility. This method is very straightforward and allows for quick assembly of components, and can incorporate custom components. However, to design a

custom component for use in the block diagram design utility that is more complex than a couple of basic logic gates typically requires the use of one of the other 2 design methods that are available to the designer. This means that if the designer is capable of using either of those methods for designing custom blocks, it would be more beneficial to utilize that design medium, because both alternatives offer a greater amount of flexibility in design.

The two other methodologies are very similar in that they are both are categorized as hardware description languages (HDLs). The first one is the Verilog Hardware Description Language (Verilog HDL) and the other being Very High Speed Integrated Circuit Hardware Description Language (VHDL). Verilog HDL was designed to be a C like language so the syntax is relatively straight forward, and all allowable data types have been predefined. Comparatively VHDL syntax is not as intuitive as its Verilog counterpart and the designer is required to define the needed data types by how they are made from a set of basic predefined data types. In contrast though the VHDL compilation process is much more flexible as files are free to have multiple components and the order of file compilation is not critical, as is the case with the Verilog HDL. Additionally VHDL also allows for design components to be put into packages and reused in other designs, which is not included in the standard Verilog HDL.

The Strength of the HDLs compared to the block design method is that the designer is given a more complete control over how each component in the system will

be implemented. Also as previously mentioned if a designer would like to specially design the components while using the block design method, the designer would need to use one of the HDLs, if the design is more complex than the assembly of basic logic gates such as ands, ors, and inverters. Along with this greater amount of design control the HDLs also allow for a design that once verified on an FPGA can easily be transitioned to an application specific integrated circuit (ASIC) using design suites such as Cadence, Synopsys, and others.

With the pros and cons of each design method in mind, VHDL was chosen as the primary design medium for the proposed designs. This was primarily due to the ability to package components, which would allow for different architectures to be tested with out having to copy and paste the code describing the basic components of the design to different files every time a new system needed to be evaluated. This would also allow for an easy transition to an ASIC for testing and further optimizations if it was deemed necessary. In the following section the design choices that arose based on the decision to use VHDL will be discussed.

3.2 Design Choices

Before introducing the proposed designs for an efficient AES encryption core it is important to understand the varying options that were considered when implementing these designs. In this section six of the major design choices will be outlined, with the

decisions that were made affecting all the designs being described. If however a design was made using different choices the reason for the choice will be described that specific designs introduction.

3.2.1 Data Path Width

One of the first considerations when implementing an AES compatible device is how much data will be processed during a clock cycle. As is outlined in the last chapter all operations produce an 8-bit output that is used to generate the next state in the encryption process. This means that if a designer were trying to build a compatible encryption device with the smallest possible footprint on a chip, a design could be made which has only a single instance of each of the key components described in section 2.2. Using this fact it can be determined that if a designer wants to use all of the components consistently you can make designs with data path widths of 8, 16, 32, 64, or 128 bits. As is shown in figure 3.1 there are proposals for different architectures that can utilize the different data path widths.

For the proposed designs the choice was made to use full 128 bit data paths. This was primarily due to the fact that when dealing with FPGAs there is a set number of gates than can be used and if the gates are not used, they are still present on the chip and factor in to the quiescent power consumption. Additionally if one wanted to further reduce the footprint of any of the proposed designs the component implementations in

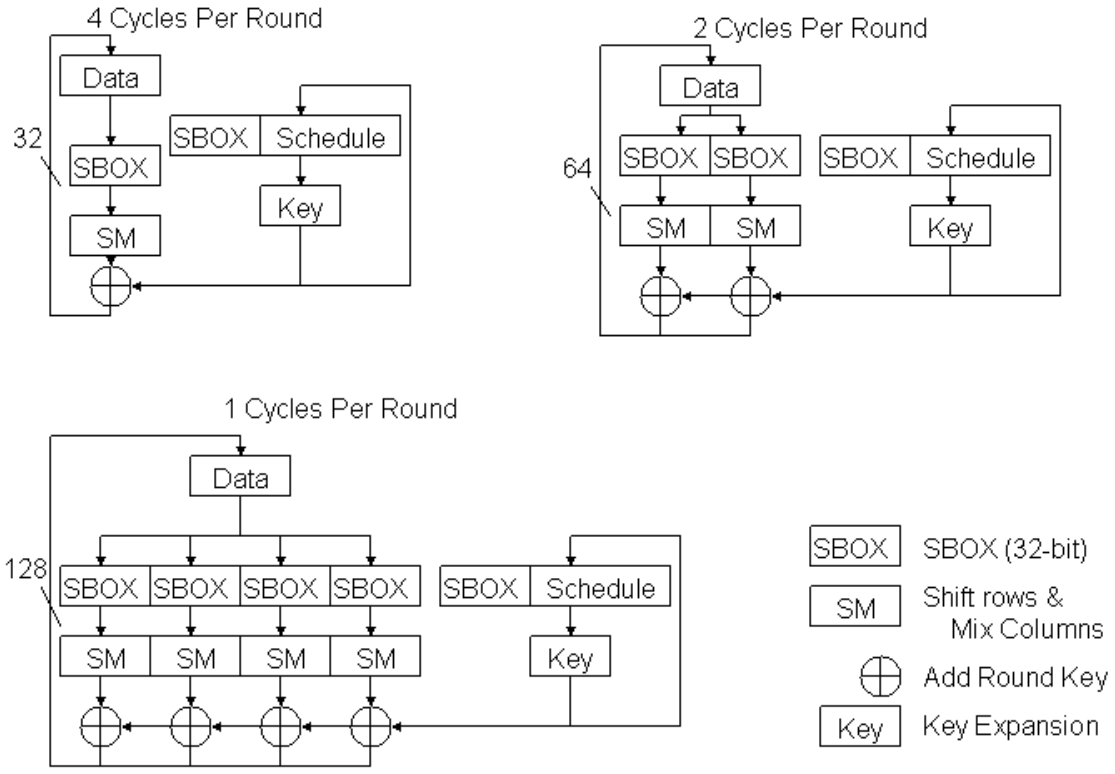

Figure 3.1: Varying data path widths

the VHDL package could be easily transitioned to another design that utilized one of the narrower data path architectures. It should also be noted that when dealing with narrow data paths the control circuitry becomes much more complex as the data has to be stored and accessed at certain times.

3.2.2 Design Organization

Design organization is another critical area where choices have to be made regarding how the core AES components are assembled. If a design was being solely optimized for speed the designer could insert registers in between the components to

allow for the highest possible clock rate. Conversely if design speed was not a critical component a designer could simply link as many components together as clock cycle time allows or optimize individual components for low power consumption as was discussed in [3]. This leads to the need to decide how many rounds will be completed in one clock cycle and how the result of one cycle worth of work should be routed to the input for the next cycle.

Another component of design organization that needed to be considered is whether or not process sensitivity lists should be used to control when the round block or its basic components processes data. This is one of the more straightforward ways to reduce power consumption because the designer can keep the system from constantly running memory access routines or any other power intensive tasks. The downside of utilizing sensitivity list is that when coupled with a 128-bit data path the circuitry required to monitor the full data path is not trivial and if used too liberally could cause the design to become larger and more power hungry.

The choices that were made with regards to questions raised in this section are part of what sets the proposed designs apart from each other and the reference designs. As a result the decisions will not be expounded upon here, but rather during their formal introduction in the following section.

3.2.3 SBOX Structure

When implementing the SBOX designers typically have one choice, which is to make a 256-byte look up table. This is due to the complex mathematical processes required to complete the multiplicative inverse, which significantly increase the time to complete an encryption. There have been attempts to create an optimized, efficient, and small S-box as presented [4], which are significantly smaller than look-up tables, but remain much slower. Satoh's design when implemented on an ASIC were shown to be one third the size of a look up table, but takes 130% more time to complete the transform, however [4] gave no concrete numbers for power consumption.
So the decision was made that since FPGAs are able to efficiently build and access look-up-tables through optimized memory blocks, 256-byte lookup tables would be utilized.

3.2.4 Mix Column Structure

The next design component that offers two very different design paths is the mix column transformation. As is laid out in its introduction the transform involves multiplications by two and three. This could be done by simply instantiating a multiplier using the Xilinx library of tools, except that multipliers take up large amounts of chip space and for each 8-bit value in the state two multipliers would be needed, or additional clock cycles would have to be added to the encryption length. The alternative to building multipliers is to use the properties of binary addition and simply remove the

multiplication by replacing it with XOR functions, which is possible because the incoming number is simply being scaled by a factor of two or three. An efficient method for implementing the mix column (figure 2.4) is laid out it [5] and is shown in figure 3.2. As is shown in figure 3.3 instead of using specific logic to implement a multiplication by three, the optimized design makes use of a times two block, as shown in figure 3.3, for both numbers multiplied by two and then adds in the number which would be multiplied by three again to avoid more complicated logic.

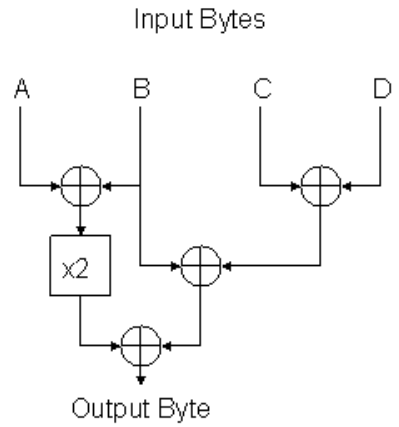

Figure 3.2: Optimized mix column structure

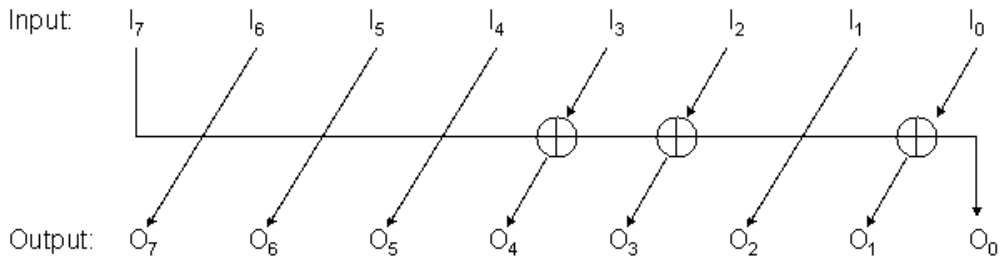

Figure 3.3: Multiply by 2 optimization

The optimized mix column structure from [5] was chosen for use in all of the proposed designs presented in this study. One reason for this was that after some analysis further optimization of the design would result in the inability to use this in systems with data paths of 8 and 16 bits, and while the design as laid out can not directly be put into a low bit data path system, it can be partitioned for use in those systems, and it was critical to keep the components as modular as possible for transitioning into other designs if needed.

3.2.5 Number of Data Input and Output Pins

One critical factor in the design process is how data will be sent to and from the device itself. Options for controlling this component can range from using control signals and placing a multiplexer on the inputs and outputs to reduce the size of the I/O busses, to the use of full 128-bit busses to expedite the data transmission process. This decision is a critical one, because if it is chosen to use a small input bus widths then the total time for encryption including data transmittal with rise dramatically. Additionally if 128-bit busses are to be utilized there will need to be a very large number of pins on the FPGA, and that will limit the number of FPGAs that can be used when the design is implemented.

For the proposed designs it was decided to use full 128 bit input and output busses, to reduce the overhead required to transmit the data and key to the chip and to receive

the results of an encryption. This means that any chip must have at least 256 pins for just the plaintext and ciphertext transmissions. Similarly a 128 pin bus would be needed for concurrent key transmittal; however, how this is handed is part of what sets the proposed designs apart and will be discussed in more detail in the next section. Additionally there will need to be some pins available for transmitting control signals, and with full 128 bit busses the control signals can be minimized by not having to have signals for which part of a key or data is being transmitted.

3.2.6 Key Input, Storage and Generation

Another critical decision that has to be made when implementing an AES device is how to handle the encryption key. This involves how to receive the key from other devices, whether to store the key for later use, and how to handle generating each of the round keys. The issues of how to handle key input will be the first issue discussed, because the choices made in how to deal with this issue can affect both the number of input pins needed and the outcome of the other stated issues. Possible choices for this issue are having a second input bus that mirrors the data input bus so that both the key and plaintext data are transmitted to the device at the same time. Conversely a multiplexer with a register could also be used to allow the key and data to be transmitted over the same input bus, which keeps the pin count down. This method for using a

multiplexer though, does increase the design footprint size, but also allows for chips with lower pin counts to be used.

The question of whether to use a multiplexer to eliminate the need for a second bus also raises the question of if a multiplexer is used, why not save the key in the input buffer, so that if subsequent encryptions require the same key then it will not have to be retransmitted. This option is particularly appealing when dealing with low power embedded systems where key changes occur rarely if at all, and the elimination of such transmissions would increase overall system performance and lessen the power consumed by key transmittal. For this issue and the previously raised issue of how to receive the encryption key the proposed designs followed different paths, and as a result the reasoning behind these choices will be discussed during the designs formal introduction.

The Final decision that must be made regarding key handling is how to generate the round keys. Some designs generate all the round keys needed and save them to registers before the encryption process starts, while others generate each round key as needed throughout the encryption process. Since both these methods rely on the same key expansion hardware it was decided to use the method of key generation as needed because it eliminates the need for registers to store those round keys and helps keep the design footprint small.

3.3 Current Designs

Since the introduction of AES many designs have been proposed, with most of them attempting to carry out the encryption process as fast as possible without concern for the power consumed by the design. Additionally the few designs meant to have small footprints and low power requirements use such small data paths that to send the plaintext and key to the device takes 16 clock cycles and 160 clock cycles to complete the whole encryption [6]. As a result there are very few designs that are freely distributed to use as reference designs with similar goals to those of this study, which meant that the reference designs had to be chosen carefully for other reasons. The two designs that were settled on both try to fulfill different needs, the first being to be fully compatible with all AES specifications, meaning it was capable of handling all encryption key sizes. This design was published by the NIST when the decision to adopt the Rijndael algorithm for AES was made and can be found at [7]. The second design which was chosen for a reference design came from the Open Cores project [8] and was published as example of a fast 128-bit AES device. Both of these designs focus on different goals and made different choices for how to implement the encryption mechanism. The following sections give a brief description of the two reference designs and how they are organized.

3.3.1 NIST Design

The primary purpose of the NIST design was to demonstrate how to build an AES device that is fully compatible with all the AES encryption key sizes. Since this design did not intend to be a demonstration of the fastest possible implementation or the most efficient it was chosen because it was representative of an average non-optimized implementation, which was concerned more about compatibility than speed or power consumption, and as a result this design requires the largest number of pins of any of the considered designs. There is a 128-bit data input bus, 256-bit key input bus, 128-bit output bus, and seven control lines (figure 3.4). Besides the standard clock, reset and data valid pins, there are key and data load pins along with a pin to specify encryption or decryption, and a 2-bit key size bus. For an encryption to be completed a key must be sent to the design by writing the key, and the key size code to the appropriate busses, followed by setting the key load pin. To begin an encryption or decryption the encryption or decryption pin must be set appropriately and the data store pin set to initiate the process. This process is shown in figure 3.5. In this design the key can be saved for multiple encryptions or stored concurrently with the data to allow for the most flexible key handling design of the designs tested.

Unlike the proposed designs the NIST design's round implementation is accomplished through function calls from a specially designed package file rather than through explicit round descriptions. This means that the physical layout is very

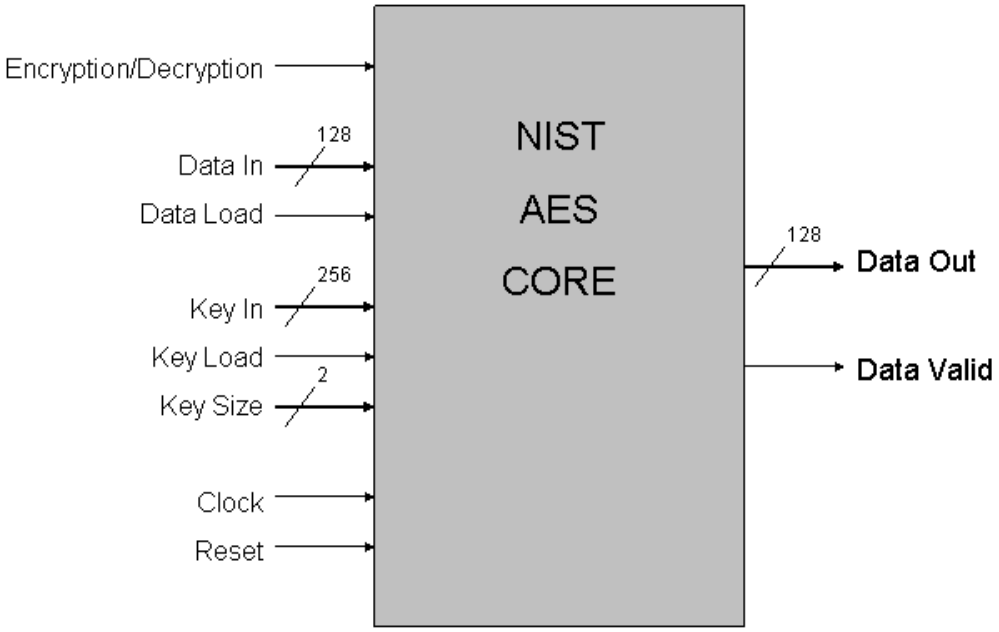

Figure 3.4: NIST design interface

dependant on the synthesizing, mapping, and routing functions of the design software, rather than the designer, a general basic diagram of this structure is shown in figure 3.6. Once the encryption process is started the design takes 12 cycles until the ciphertext is written to the data output bus. When the ciphertext is written to the bus the data valid pin is raised and after one cycle the data output bus and data valid pin are cleared.

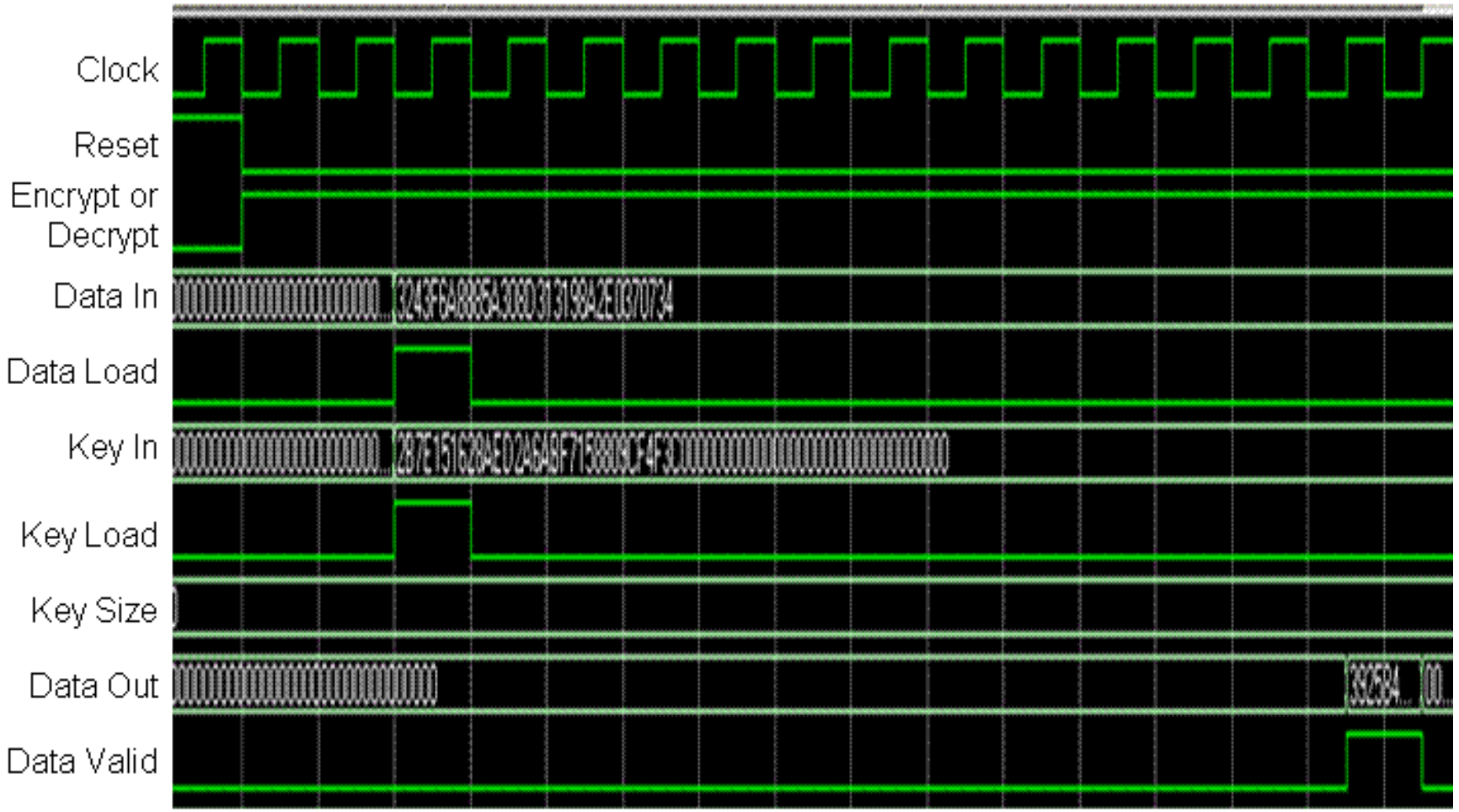

Figure 3.5: Waveform to initiate an encryption for the NIST design

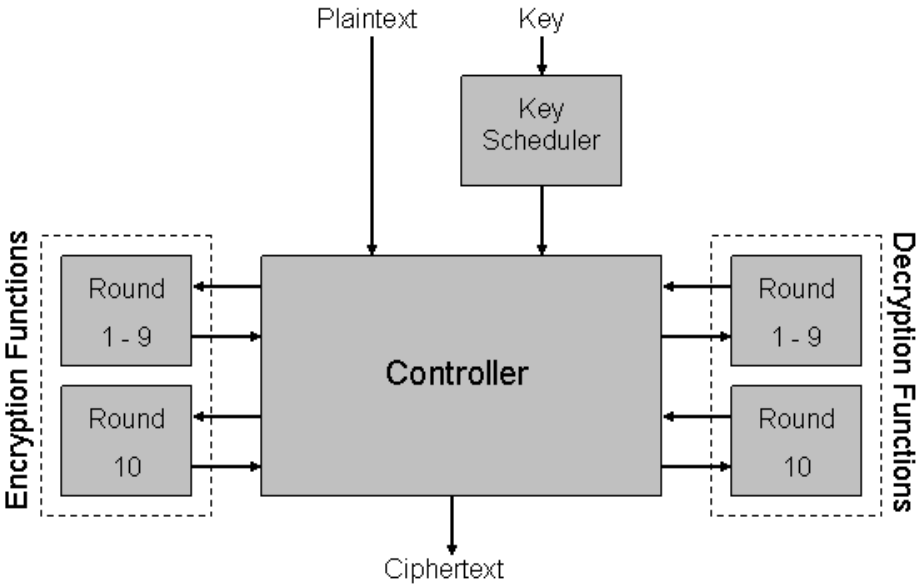

Figure 3.6: Block diagram of the NIST design's architecture

3.3.2 Open Core Design

The Open Core design was made to be a strictly 128-bit key encryption and decryption core. One of the things that sets this design apart from the NIST design and many of the other published designs is that the input bus for both the key and data is only 64-bits wide. These busses are complimented by five control signals including an encrypt or decrypt select, start, load, clock, and reset, and the usual data valid and the 128-bit data out busses (figure 3.7). This means that to enter the full 128-bits of the key and data a pulse must be sent over a load pin. The upper half of the key and data are stored when the transitions from low to high, with the lower halves are stored when the pin transition from high to low. The falling edge of the load signal is also what signals

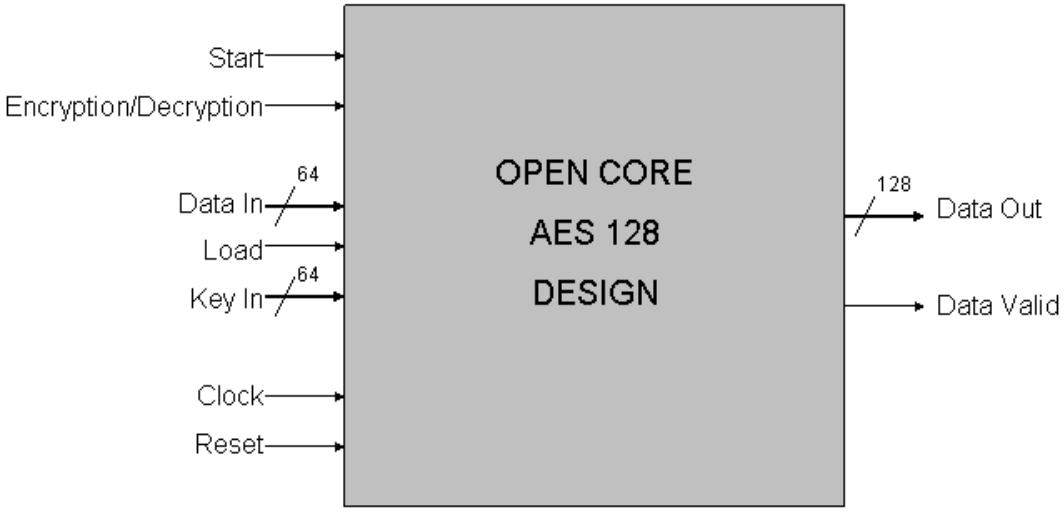

Figure 3.7: Open Core design interface

the core to commence its task. This process is shown in figure 3.8. The entire encryption processes takes 13 clock cycles for the data to written to the output bus.

The architecture of the Open Core design is similar to the designs which are proposed in the next section, as they are not primarily based on function calls and software optimization. The basic structure of this design is a round block which could be used for both encryption and decryption via a select bit, and the round's output state is rerouted to its input for the start of subsequent rounds, with a multiplexer to route the state to the output bus at the proper time. Similar to the previously mentioned choice for the proposed designs to use a key generation as needed system, this design continues in that endeavor, except that there are buffers to store the round keys if a decryption was required. This basic architecture is shown in figure 3.9.

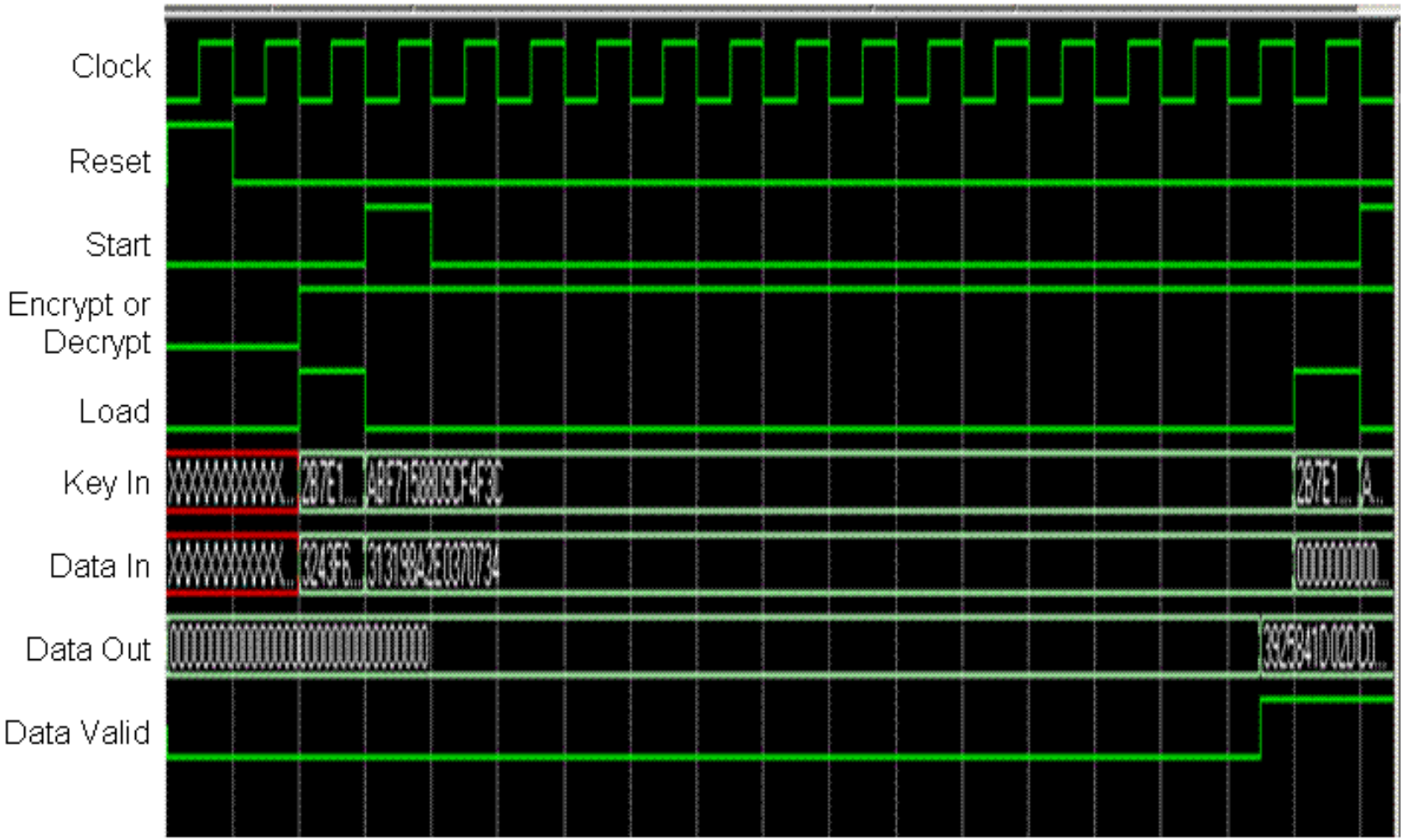

Figure 3.8: Waveform to initiate an encryption for the Open Core design

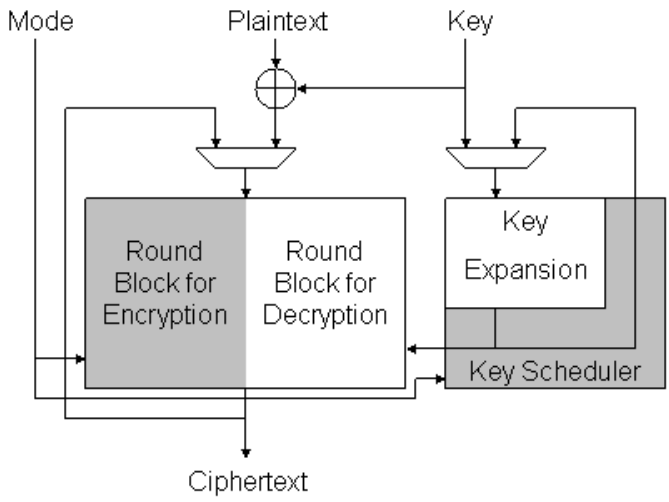

Figure 3.9: Block diagram of the Open Core design Architecture

3.4 Proposed Designs

The following section is a formal introduction to the designs that were built with the goal of achieving an efficient and flexible AES encryption device while not completely sacrificing performance. Each design's introduction will offer a formal description of how the device works and a block diagram showing the structure of the design. A diagram showing the designs interface and a waveform of the signals required to initiate an encryption will also be shown, similar to the reference design introductions.

3.4.1 Standard Design

The First design that was built was meant to function as a benchmark, so that once analyzed the other designs could be modified to allow for improved performance and efficiency, as a result this design has been termed the Standard Design. As was

previously explained in section 3.2.1 the design is based on using a full 128-bit data path width, which if needed could be modified to use narrower data paths. The way that data is transferred to and from the device is through three 128-bit data busses, one for the data, one for the key, and finally one for the encrypted data's output. Besides the clock there are two control inputs, one for device reset and one for encryption start, and 1 control output which is toggled to alert the receiving device that the data output is valid and the encryption completed. This interface is shown in figure 3.10.

The Standard Design was designed so that a single round would take only 1 clock cycle, which means that from the time the data and start signal are received it takes 11 clock cycles for the result to show up on the output bus. When the start signal is initially received the data and key signals are added together and sent to the state register, while the key is also sent to the key register. Key generation is very simple in that for each clock cycle the hardware is active the data stored in the key register is sent to the key expansion block to generate the new round key and that is routed back to the key register for input during the next round.

The output of the state register is connected to the round block through a multiplexer which routes either the state block or the round block's output to the round block input, which is used 10 times to complete an encryption. The input of the round block is tied directly to an array of 16 S-boxes, which are set so that a memory lookup occurs only if the input changes. The output of the Substitution block is then connected

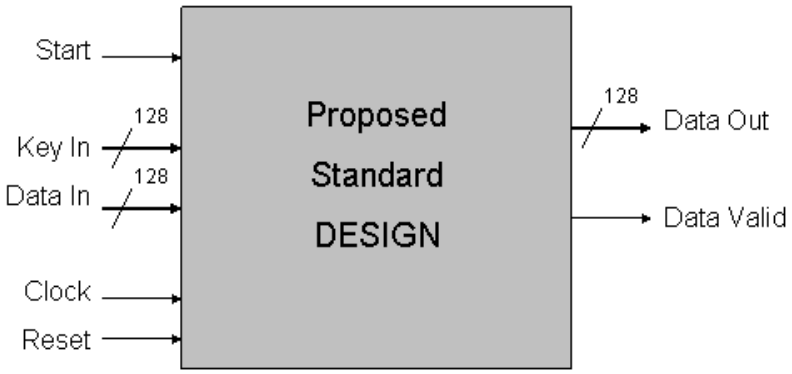

Figure 3.10: Standard Design interface

to the input of the mix column block. The shift row transformation is accomplished by reordering of bus wires as they are connecting to the mix column block. From the mix column block the data is then sent to the add round key block, which gets the round key from the output of key expansion unit and combines the two. The result is the next state in the encryption process and can be routed back to the data register for saving for the next clock cycle or sent to the output.

As explained in section 2.4 during the final round of an encryption the mix column transformation is not used, so rather than build a special round block that is only used during the final round a multiplexer has been added between the mix column block and the add round key block that can either pass through the input of the mix column block or its output based on the current round. While this adds to the physical size of the round block it reduces the overall size of the design since a modified final round block is not needed that would include a SBOX array, shift rows, and add round key component. Once this final round has been completed the output state is written to the output bus,

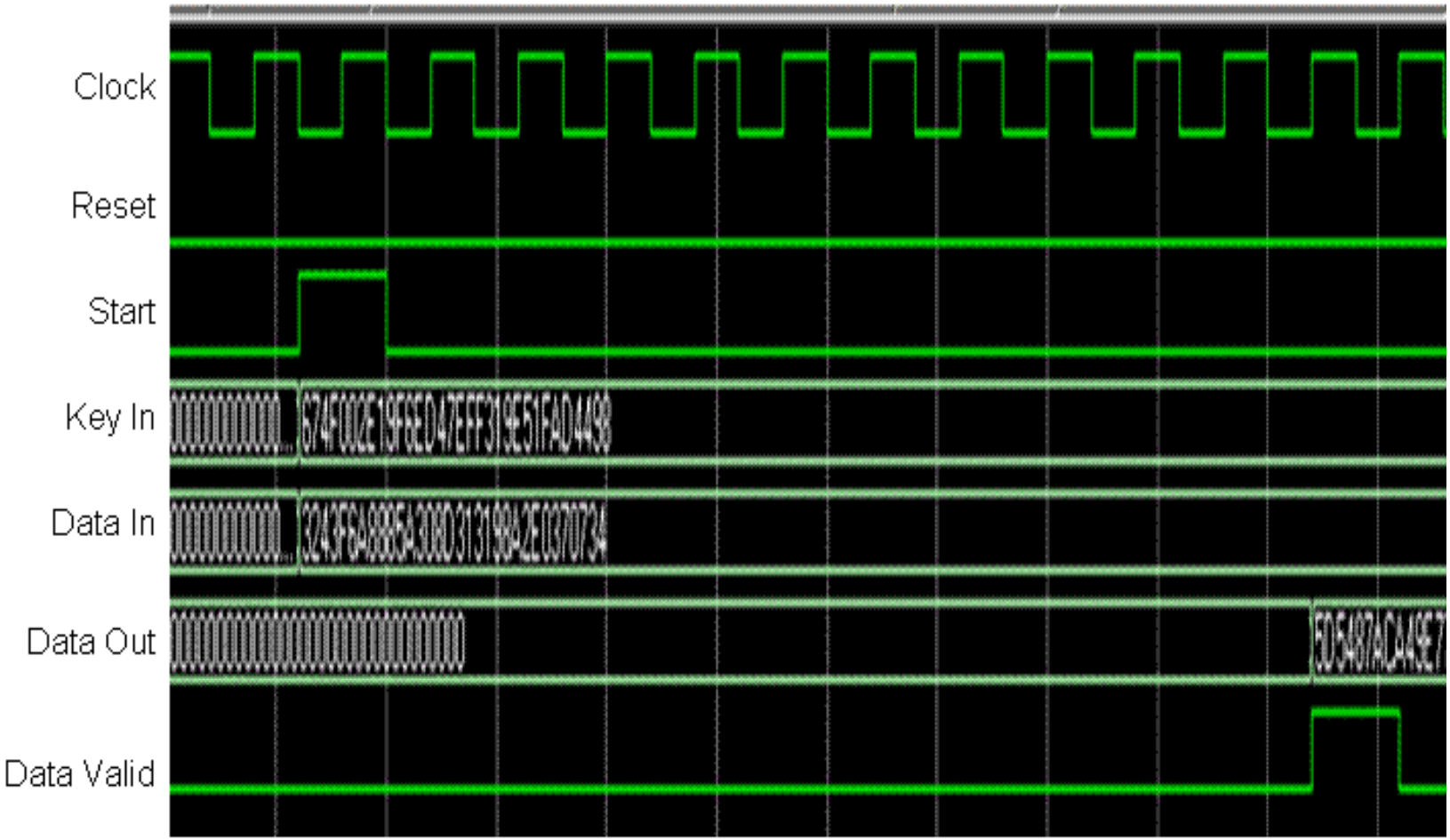

Figure 3.11: Waveform to initiate an encryption for the Standard Design

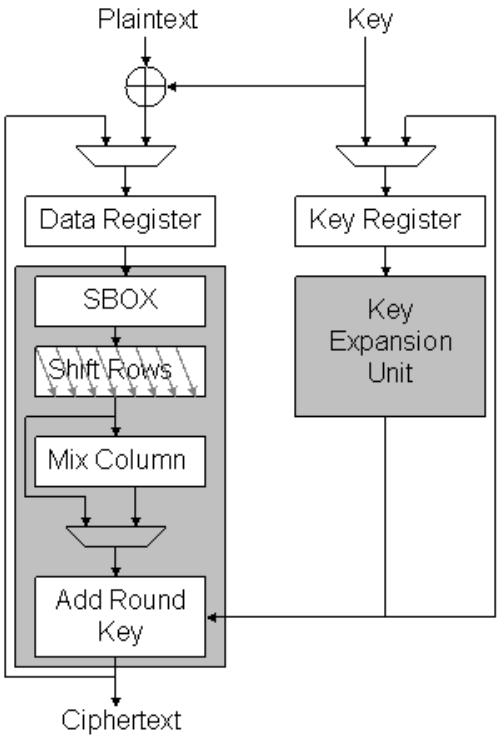

Figure 3.12: Block diagram of the Standard Design architecture

the output valid pin is toggled and the encryption process is complete. It should also be noted that as soon as the output valid bin is raised another encryption could be initiated. An example of the process required to start this encryption can be seen in figure 3.11, with a block diagram of the Standard Design's architecture in figure 3.12.

3.4.2 Hard Key Design

The second design, which has been named Hard Key, is based on the Standard Design previously introduced, with modifications being made to the interface logic. The goals of the modifications are to allow for the storage of an encryption key and to reduce the pin count so that smaller chips could be utilized if necessary. The implementation of

the round hardware remains identical to the standard design's implementation that is previously described. The first modification was to add an initial key register which fed into the multiplexed key register input, this allows for multiple encryptions that share a single key to be completed without the need to repeatedly receive the encryption key.

By eliminating the need to constantly transmit a key to the encryption core, the benefit of having a dedicated key input bus was lessened. This meant that the goal of reducing the needed pin count could be accomplished by placing a multiplexer on data input bus and adding an extra control signal to indicate whether it is a key or plaintext data on the bus. The modified control logic, which is shown in figure 3.13, was then designed so that if the new pin called store key is raised the data on the bus is written to the initial key register and if the original start pin is toggled the data on the bus is added to the contents of the initial key register and sent to the state register for processing. This

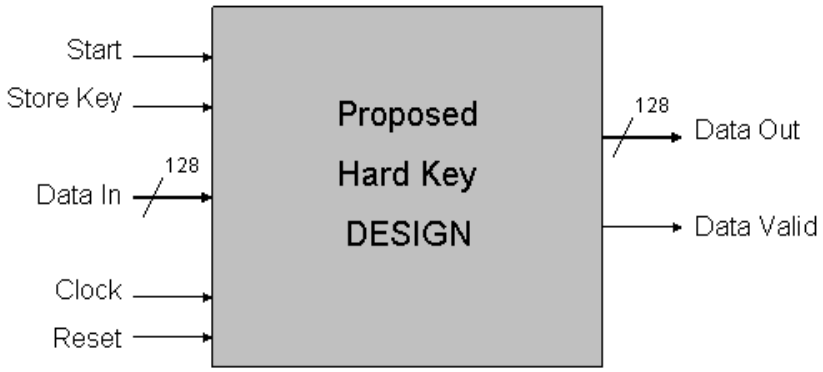

Figure 3.13: Hard Key design interface

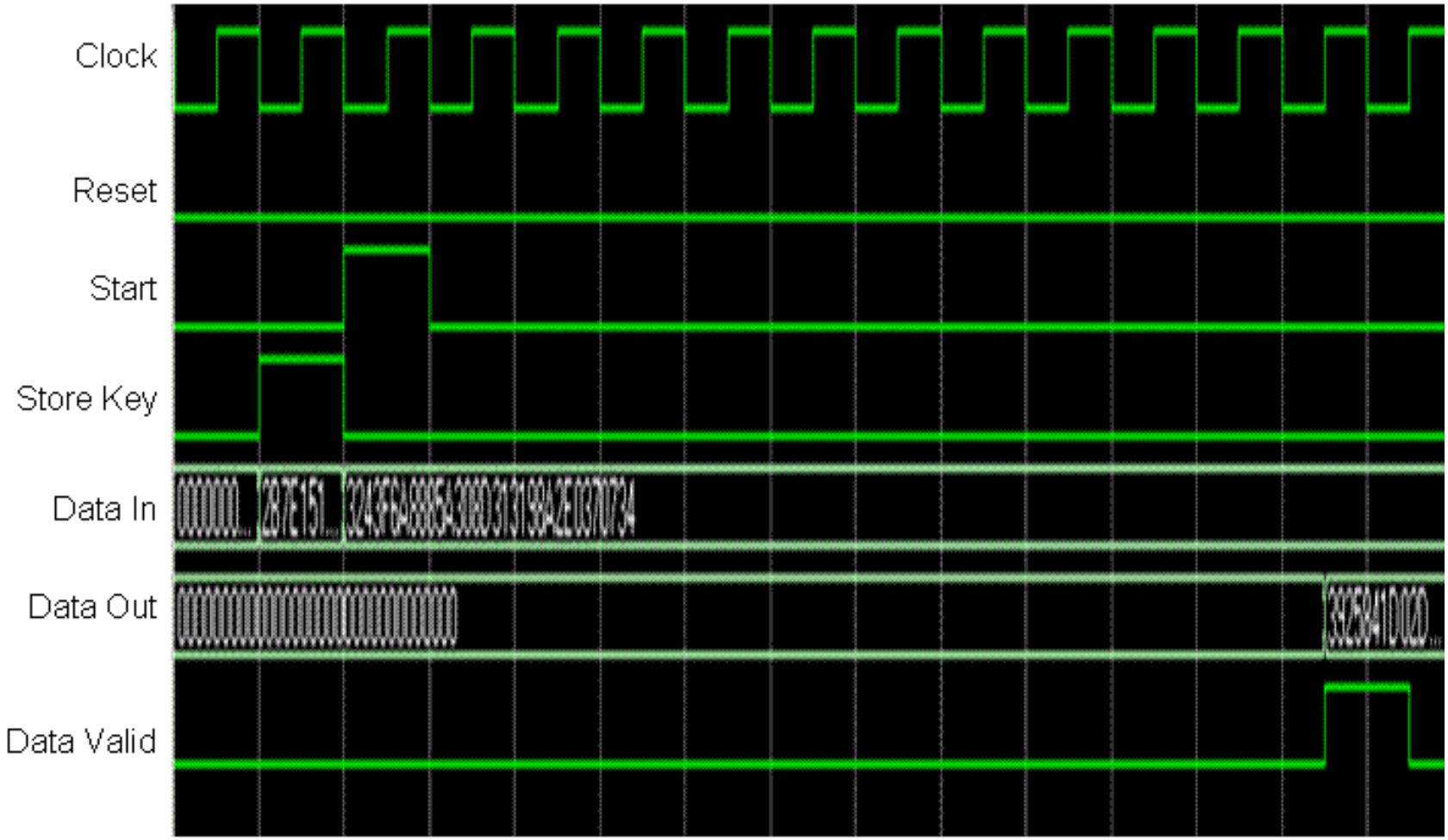

Figure 3.14: Waveform to initiate an encryption for the Hard Key design

process is also shown in figure 3.14. An additional change was made to the control logic so that when the reset signal is sent the store key pin has to be set for the initial key register to be cleared; otherwise the stored key is retained during the device reset process.

Just like the standard design the hard key design takes 11 cycles to encrypt data and on the 12th cycle the result will be present on the data out bus with the data valid pin being toggled. The penalty incurred for changing keys is the one clock cycle needed to write data to the bus and raise the store key pin. A block diagram showing the architecture of this design can be found in figure 3.15.

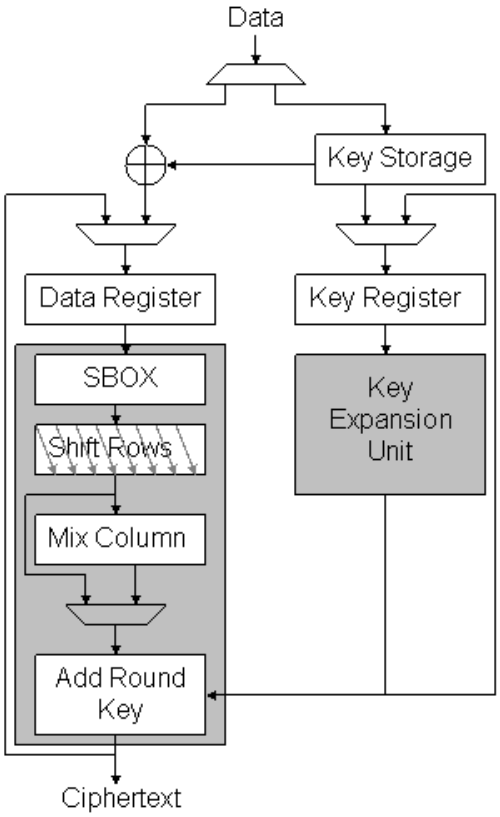

Figure 3.15: Block Diagram of the Hard Key design architecture

3.4.3 Dual Stage Design

The last of the proposed designs has been termed the Dual Stage design, because this design was built with the intention of reducing the number of clock cycles required to complete an encryption. To accomplish this it was decided that the most promising method would be to chain multiple rounds together. After considering varying round depths it was decided that two rounds would be the most effective because it would cut the round completion cycles in half while being a number that easily factored to 10 and would not increase the size of the design by more than a factor of 2.

The basic structure and interface of this design is identical to that of the hard key design, meaning the round block and key expansion units are identical to the standard design, but with the altered key storage and interface changes (figure 3.16). To add another round to the design it was decided to simply chain one round and key expansion block to another, which would allow for the addition of more rounds if this design were

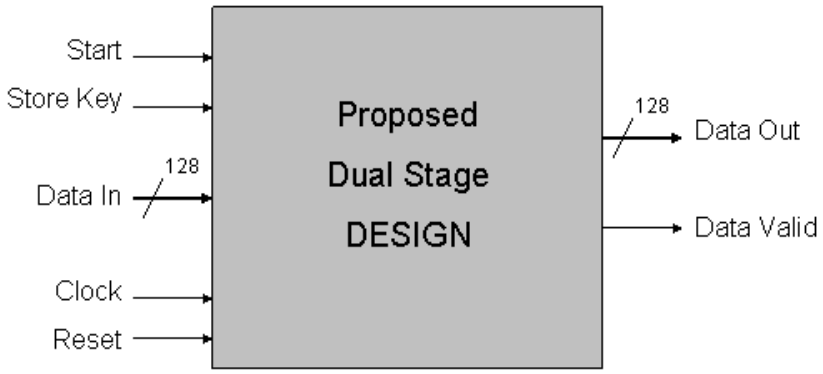

Figure 3.16: Dual Stage design interface

to show promise in its comparisons to the other designs. This meant that instead of feeding the output of the first round back into the state register it was linked directly into the second round and key expansion blocks. The output of the second round and key expansion blocks were then routed back to the state and key registers. The multiplexer placed in the first round block to allow for the bypassing of the mix column hardware was then hard wired to zero so that the Xilinx ISE software could remove it during design optimization, while the multiplexer in the second round block was wired to the control logic to function as the final round. A Block diagram of this design can be found in figure 3.17.

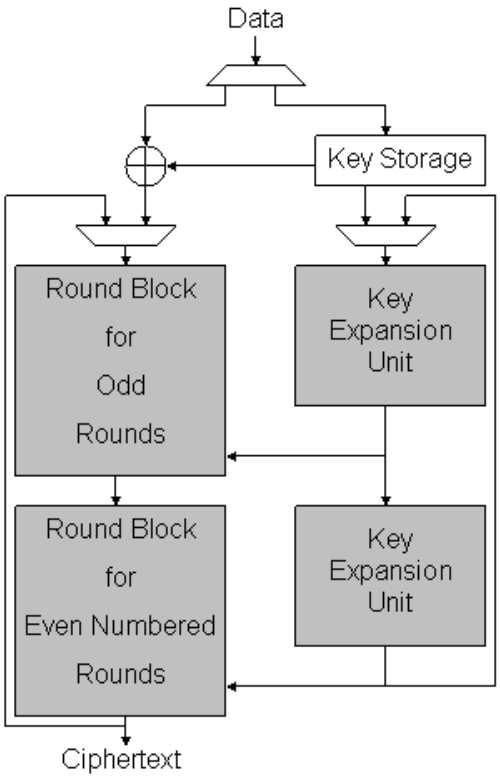

Figure 3.17: Block diagram of the Dual Stage design architecture

By chaining two rounds together the cycles needed to complete an encryption would fall to six, from the start of the start pin being toggled. Similarly the penalty for changing keys remained at a single clock cycle, and to clear the key a new key must be set or a system reset along with raised set key pin. This along with the signals required to start an encryption are shown in figure 3.18.

3.5 Design Verification

Before any analysis of the three designs could be undertaken it was critical to ensure that that each of the designs constituted a valid implementation of an AES encryption device. This was done using 300 test vectors from the NIST random vector files published with the initial release of the AES standard, which can be found at [7]. Once test bench files were written for each of the designs Mentor Graphics' ModelSim software was used to verify that each of the designs successfully completed all of the required test encryptions. As can be seen in table 3.1 each of the proposed designs as well as the reference designs encrypted all of the test vectors successfully.

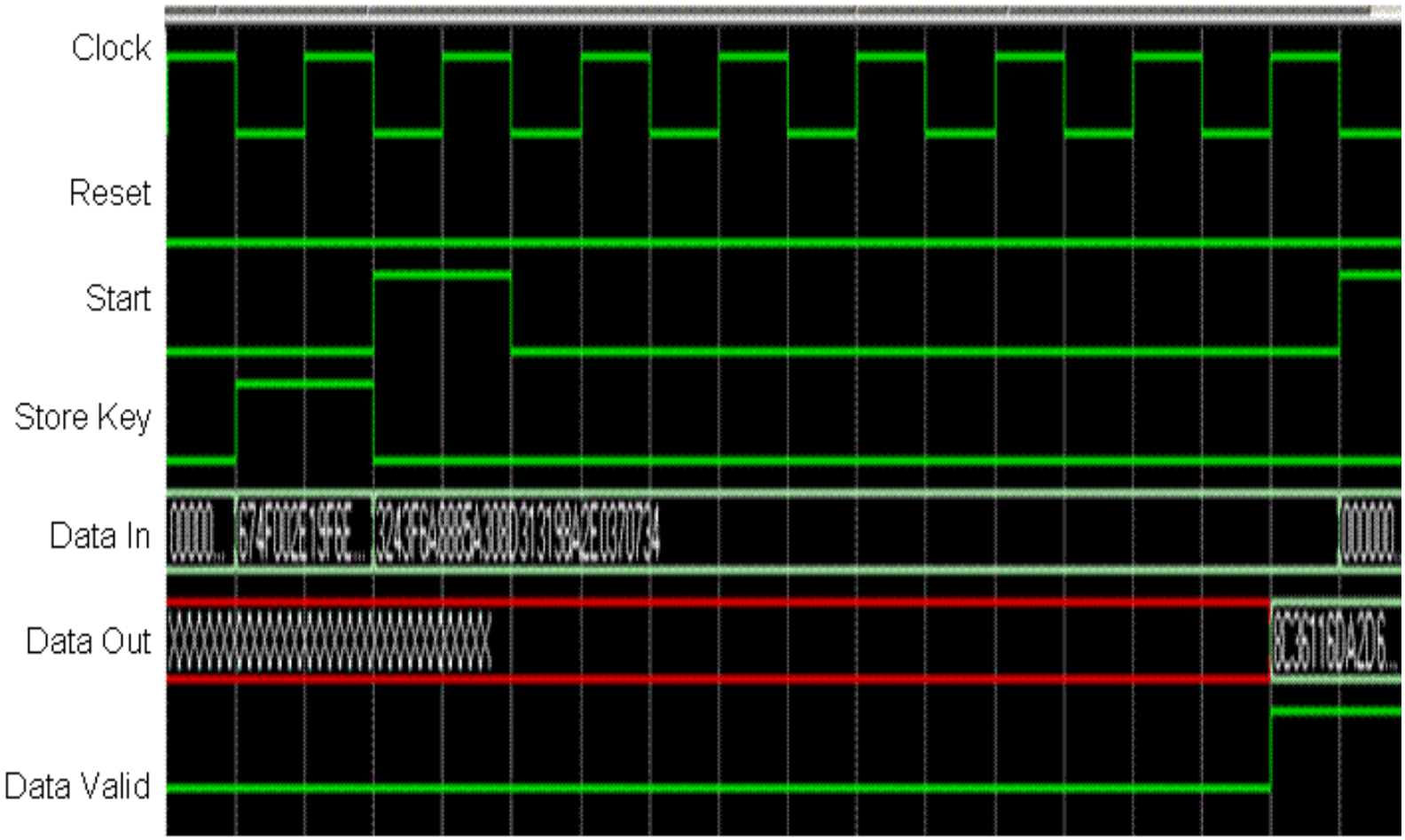

Figure 3.18: Waveform to initiate an encryption for the Dual Stage design

Table 3.1: Design verification

	Number of Test Vectors	Vectors Passed	Vectors Failed	Design Verified
NIST	300	300	0	Yes
Open Core AES-128	300	300	0	Yes
Proposed Standard	300	300	0	Yes
Proposed Hard Key	300	300	0	Yes
Proposed Dual Stage	300	300	0	Yes

Chapter 4

Performance Evaluation

Having discussed the architectural differences between the proposed designs and the reference designs in their introduction, the actual performance of these designs must be analyzed and compared, which will be the focus of this chapter. This will be accomplished by first discussing the different approaches that can be used, and followed by a discussion of what methodology was chosen. This chapter will then transition to the results of the different analyses and a discussion of the how the designs preformed.

4.1 Approach

When looking at the options for gathering empirical data about the designs it was critical to decide what the empirical data needed to measure. Since the focus of the proposed designs is Power conscious encryption it is imperative to understand what is

being looked at. Power consciousness as this pertains to encryption designs for FPGAs would most easily encompass the speed of encryption and the power consumed by the design. With these two metrics in mind the methods for obtaining these measures will be compared in the remainder of this section.

The first option that was considered was physically testing each design by using a prototyping board. Power measurements could be taken by setting up probes to monitor the chips power consumption. The problem with this method though, is that to implement the design on a prototyping board would require designing an additional interface wrapper to map the design to the board's given architecture. This could mean designing an enhanced parallel port (EPP) interface as is used for many Digilent Inc. boards [9]. This approach does not seem reasonable because the added interface wrapper would affect the overall design performance and add to the FPGA's power consumption, which would lead to inaccurate and inconsistent results due to varying design interfaces requiring different wrappers. Furthermore most prototyping boards do not have easily tapped power leads and so the boards would have to be modified to obtain accurate power consumption data.

The second option that was considered was to physically test the designs by using a logic analyzer to control the inputs and monitor the outputs of the chip; this method would require a special circuit be made for each design interface and footprint to power and program the FPGA as well as to connect it to the logic analyzer. It should be noted

that powering an FPGA requires 3 different voltage levels [10], which would mean building 3 separate power supplies and 3 probes for monitoring each supply. This method although the most complicated of the options considered would provide the most accurate results possible. However, the resources required to complete this type of analysis were too great in terms of both time and hardware.

The final option to be considered was to utilize software based simulation models. The way these programs typically work is by determining the frequency of change on the inputs to a given design, and then the software calculates how the internal nodes will be affected. Using these models, which are derived from physical testing, a result can be obtained that is an accurate estimation of a chip's power consumption. Additionally some simulators are able to use variable change dump (VCD) files, which are generated by simulators such as ModelSim by Mentor Graphics. These files not only monitor the activity rates of the designs inputs and outputs, but also monitor the changes in the design internally, and when given to a power estimation simulator is the most accurate way to estimate the power consumed by the design. Even though this technique produces only an estimation of the power consumption rate, it is able to provide more specific estimations than physical testing. This includes categorizing the power consumed into areas such as input, output, logic, and quiescent.

4.2 Chosen Methodology

Now having an understanding of the different ways in which the design's power characteristics can be obtained. A more in depth discussion of the method that was chosen for gathering the needed data will be outlined as well as the choices that needed to be made in this process. When looking at the varying option it became clear that the option based on utilizing software simulation was the most promising. This method would allow us to get reasonably accurate power estimations while not having the build the circuitry to program, and physically test the different FPGAs and designs. In the remainder of this section a presentation of what was required to complete the simulations will be laid out.

From the beginning of this research the provided hardware has always been Xilinx FPGAs and their ISE development suite, it for this reason that the all of the chips considered and used in this analysis were made by Xilinx. The first step in running the power simulations was to use the Xilinx ISE suite to build a post place and route simulation model. This model is a description of how the design would be physically programmed onto a chip, along with the delay constraints of each connection or path. Another output of this process besides the model is that it then calculated the minimum clock delay, which is easily converted to the maximum clock speed by taking its inverse. To accomplish this result the ISE suite first checks the syntax of the VHDL file and once

verified maps the design to the specific FPGA that is being tested. The mapping process ensures that the FPGA has enough pins for the inputs and outputs as well as gates to ensure that the design can fit on the designated FPGA.

After the mapping process completes the place and route process begins, and during this time the software tries to place the design onto the chip and route signals around the chip efficiently. The Xilinx tools allow for emphasis to be placed on speed or design footprint while completing this process, which led to a dilemma because while the proposed designs were striving for efficiency the reference designs were not. Since the primary constraint when deciding upon which FPGAs to use in analysis was the number of pins available for inputs and outputs and chips with a satisfactory number of pins were large enough for all the designs, it was decided to emphasize speed, because this would mean the components were being placed based on what came next which would help ensure small delay times. This would also reduce the signal power consumption because logic block would be placed as close to each other as possible and not where they would fit to help keep the footprint small.

Once the post place and route model had been generated it was time to use Mentor Graphics' ModelSim to generate the VCD file. Since the point of the simulations were to learn about power consumption it was important to design test benches that would complete as many encryptions as needed without containing a large number of clock cycles where no activity was to take place. This would help to shorten the

simulation time as well as ensure that the power analysis program would not be given a data file where there were large numbers of cycles with nothing happening. Additionally all simulations were set so that ModelSim would stop two clock cycles after the final encryption was complete, which would allow for the done bit to return to zero on all the designs except the Open Core design where it would remain raised until another encryption was started.

The final step to obtaining the power consumption characteristics was to use Xilinx's XPower program contained in the ISE suite. This program works by analyzing the design's native circuit description (NCD) file and physical constraint file (PCF) generated by the ISE software and the VCD file from the ModelSim simulation to determine the power consumption of a chip programmed with the design to within 10% accuracy. The XPower software also allows for the user to specify the ambient temperature and airflow around the chip for producing more accurate results for a given environment, but these values were left at their defaults of 25°C and zero respectively. Once the analysis has been completed, XPower generates a series of tables showing its results, which include but are not limited to input, output, internal, quiescent, logic, signal, and total power. It is with theses estimations that the subsequent analyses will be based.

Now that the method for obtaining the estimated power consumption has been outlined 3 issues must be covered. The first is how many test vectors should be run per

Table 4.1: Power Estimation for different amounts of Test Vectors

	Number of test vectors		
	10	100	200
Open Core	1490.28	1484.89	1494.93
+ 10%	1639.308	1633.379	1644.423
- 10%	1341.252	1336.401	1345.437

simulation? To determine an optimal number of test vectors simulations were run using the Open Core design and 10, 100, and 200 test vectors obtained from the random test vector file provided with the NIST design [7], the results are shown in table 4.1. By examining this table it should be quite obvious that all of these simulations fall within the 10% accuracy window, which means that any of the considered number of test vectors could be used. For the rest of the simulations it was decided to use 100 test vectors, so that the VCD file would contain a representative account of the encryption process, while not dramatically increasing the overall simulation time and trace file sizes.

The second issue that must be considered is what the results of the XPower simulation mean. Since the units in the XPower tables are only labeled as milliwatts, does that mean it is the total power consumed by the design over 1 clock cycle, or is it the expected power consumption of the design lasting for an entire clock cycle as is shown in figure 4.1. As with other simulators such as Synopsis it is the latter of the two options.

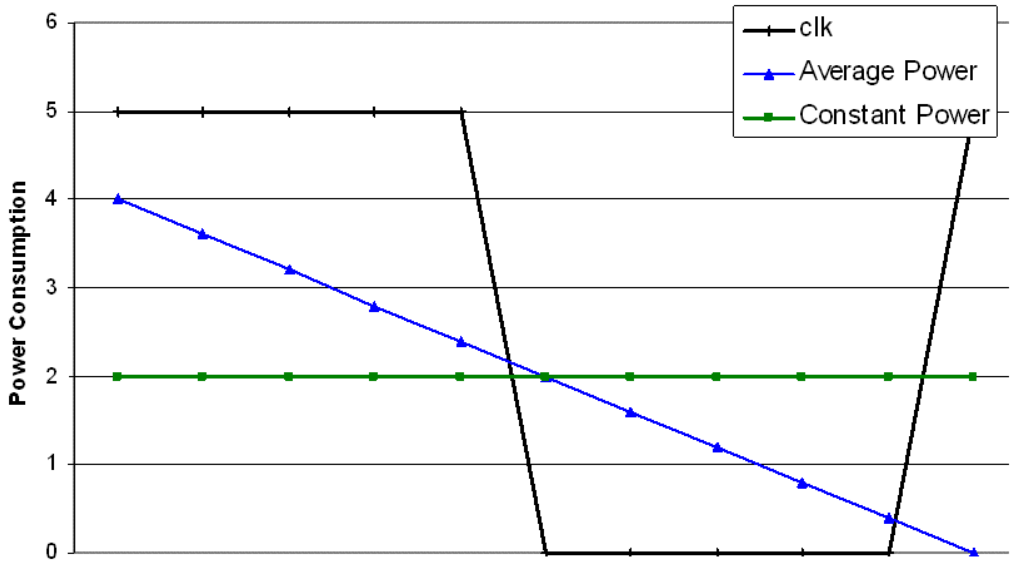

Figure 4.1: Possible power consumption models

The final issue that needs to be addressed is how the reference designs that also contain decryption hardware are affected during power testing. It is important to note that when a VCD file is generated by the ModelSim software the only time a value is recorded for a given design element is when its state changes. This means that if no decryptions are executed during the simulation that the decryption hardware will not be a factor in the power estimation and allow for the power simulation results to be used for comparisons. The one issue with this method is if a design is structured such that the data path is tied to both encryption and decryption blocks with a select bit on the output then both blocks will be active and the power consumption will increase greatly. As was explained in section 3.3.2, the Open Core design is designed with a hybrid round block capable of encryptions or decryptions, which meant that if round components were not

segregated properly then certain parts of the decryption hardware could get factored into the power analysis. However, it was decided to keep the Open Core design as a reference, because it could help to give insights into how to better design for efficiency in system design rather than in component design. Similarly the NIST design is capable of 256 bit encryption keys, but by limiting simulation keys to 128 bits the extra hardware required to handle the larger keys should not be factored into the power ratings and cause the simulation results to become irrelevant.

4.3 Comparison Analysis

To see how the proposed designs and reference designs functioned in comparison to each other, it was decided to use two simulations. The difference between the two simulations was in which chip would be selected to contain the design. In case an issue arose in the simulation process that would require physical testing to resolve the chips were chosen based on their physical availability for the research. These chips were the Xilinx Spartan 3 and the Xilinx Virtex 2 Pro. Additionally the use of these two chips allowed for another factor to be examined in the comparisons, which was whether having extra space in which to implement the design would make a difference in how the design would consume power. The Spartan 3 chip is a standard FPGA with logic cell counts ranging from 1,728 to 74,880 and I/O pin counts ranging from 124 to 784, which allows for a chip package to be chosen that most closely fits the requirements for

implementing the designs. The Spartan 3 chip that was chosen was the XC3S1000, because this was the smallest chip Xilinx offers in the Spartan 3 series that has enough I/O pins to satisfy the needs of all the presented designs after design optimizations. Conversely the Virtex 2 Pro is a full featured and flexible FPGA that contains two Power PC cores and logic cell counts ranging from 3,168 to 99,216 and I/O pin counts ranging from 204 to 1,164 [12]. A Virtex 2 Pro was chosen (XC2VP30-FF896) that contained 30,816 logic cells and 556 I/O pins. These two chips were then set to run at 25 MHz and 5 MHz during the simulations because all of the designs could perform at these rates.

It is also important to note that during the simulations all design used the 5fg676 package for the Spartan 3, except the Hard Key and Dual Stage design during the 25 MHz simulation, which used the -5fg456 package. This was done to gain an insight into whether there was a significant difference in the FPGAs power consumption rates within a specific model, based on its package type.

The results of these simulations are shown in figure 4.2 and figure 4.3. These results are also detailed in table 4.2 and table 4.3. These results show that both the Standard and Hard Key designs are much more efficient than the two reference designs and the proposed Dual Stage design. This is evident by the numbers for the total power consumption as well as the categorized power consumption numbers. The most interesting results though were the high power consumption rates for the Open Core and Dual Stage designs. After reexamining the design architectures it became apparent that

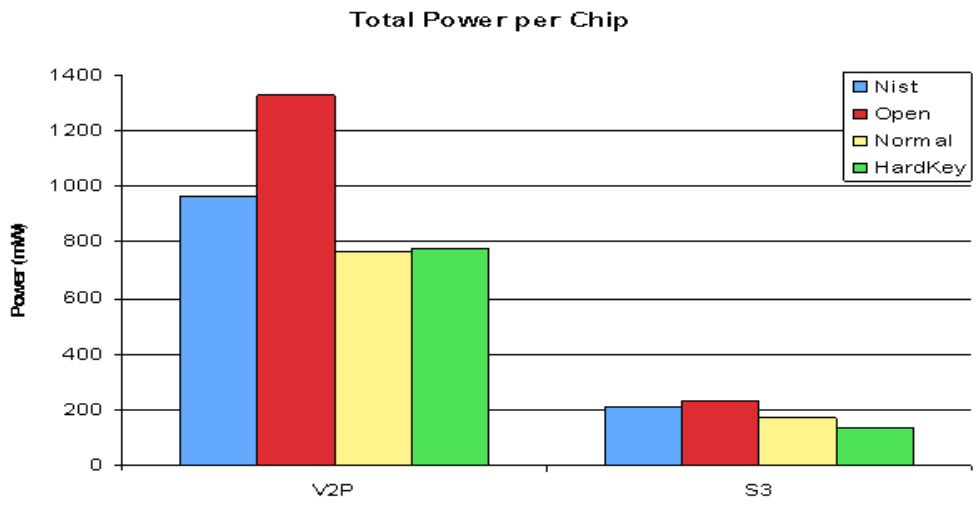

Figure 4.2: Total power analysis for 100 vectors at 25 MHz

Table 4.2: Comparative analysis for designs at 25 MHz

Design	Chip	Total Power	Quiescent	Vccint	Logic	Signal	Package
NIST	V2P	969.34	477.5	449.62	189.93	242.16	ff896
	S3	210.38	92	85.82	40.11	40.56	fg676
Open Core	V2P	1326.8	477.5	837.72	320.02	500.77	ff896
	S3	235.45	92	136.62	60.28	70.89	fg456
Standard	V2P	768.06	477.5	279.02	74.31	176.66	ff896
	S3	172.43	92	73.79	21.87	45.44	fg676
Hard Key	V2P	778.84	477.5	289.42	80.19	184.88	ff896
	S3	138.21	55.5	75.39	23.33	47.94	fg456
Dual Stage	V2P	3988.86	492.5	3490.27	1606.3	1855.7	ff896
	S3	714.85	55.5	644.77	298.02	343.13	fg456

** All simulation results are measured in milliWatts

the reason for the high power consumptions is improperly implemented design segregations. For the open core design the use of multiplexers to route data from the appropriate encryption or decryption blocks such as the mix column block do not keep the unused part of the block from processing the data and consuming power, but just

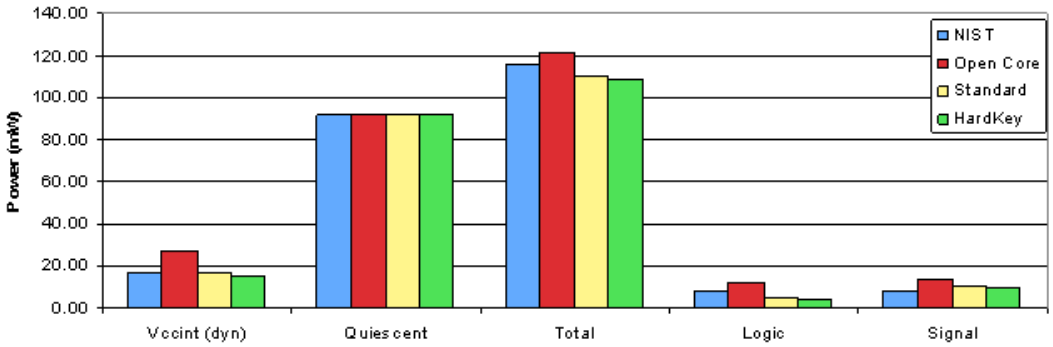

Figure 4.3: Total power analysis for 100 vectors at 5 MHz

Table 4.3: Comparative analysis for designs at 5 MHz

Design	Gate Count	Vccint (dyn)	Quiescent	Total	Logic	Signal	Cycles per encryption
NIST	95,201	17.16	92.00	115.68	8.02	8.11	14
Open Core	101,240	27.31	92.00	120.68	12.05	14.17	16
Standard	43,508	16.83	92.00	110.36	5.04	10.48	13
Hard Key	43,521	15.50	92.00	108.97	4.64	9.63	14
Dual Stage	78,938	119.13	92.00	214.05	54.69	63.04	7

** All simulation results are measured in milliWatts

ignore its output. Similarly the reason for the increased power consumption in the dual stage design can be traced to the increased hardware size and the blocks not having triggers in certain areas to prevent them from continuously running.

When studying the results of this round of simulations it was important not only to analyze the total power consumed result but also the categorized power consumed result, because these values could help to illuminate a given design's strengths. The first category that was examined was the logic power, because this is were the design components power would primarily be factored in. As is shown in figure 4.4 the logic power of the standard and hard key designs is much lower than that of the other designs,

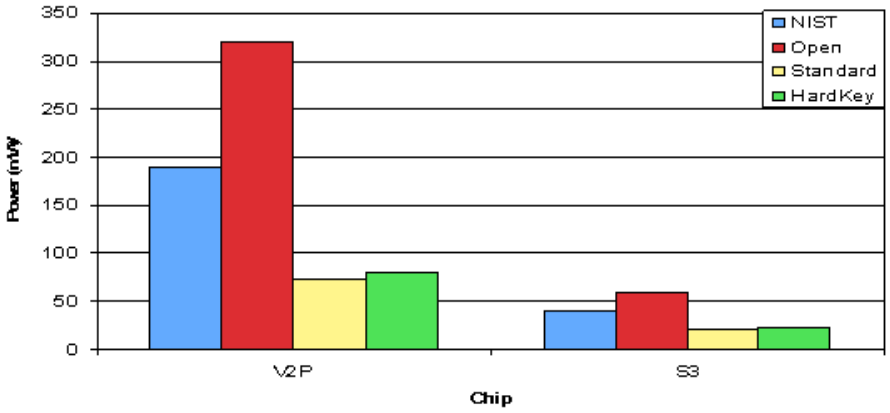

Figure 4.4: Logic power analysis for 100 vectors at 25 MHz

which means that the optimizations to the mix column blocks, the directly mapped shift rows function, and the general approach of trying to self optimize the data path while minimizing design size have produced promising results.

Additionally when looking at the signal power consumed the Standard and Hard Key designs are consistently top performers, which is good because it meant that the designs were partitioned well enough for the ISE suite's implementation functions to efficiently place and route the design on the chip. The only simulation where the Standard and Hard Key designs did not perform the best was in the signal power consumed on the Spartan 3 chip; however the results were close enough that the consistency of the two proposed designs proved to be more promising than the inconsistently performing NIST design.

The decision was made not to examine the input and output power consumed by these designs because they would have to deal with the same overheads involved in

receiving and transmitting data. The one exception to this is that the NIST, Hard Key, and Dual Stage design are able to store a key for a series of encryptions, which when completing a large number of encryptions would become a very positive addition, but would not offer as large a benefit in these simulations. Another interesting observation that should be noted is that the quiescent power consumed in the FPGAs was constant regardless of design and typically was one of the largest areas of power consumption. The results showed that while power consumption was dependant on the chip package as well as the model, the dominating factor was the model of FPGA chosen. To further understand this and how vital FPGA choice was the next series of simulations were conducted.

4.4 FPGA Choice Analysis

When examining the results of the comparison simulations one thing that really stuck out was how much quiescent power was being consumed in the chips. The quiescent power unlike other power consumption statistics was not dependent on the design being simulated but only what chip was being used. This was not unexpected, as FPGAs are not known for excelling at power management. The reason for this is that when FPGAs are designed they are almost always optimized for speed, which means large leakage currents across the entire chip that cannot be avoided. With this in mind it

was decided to do a comparative simulation of which FPGAs offered the best power consumption traits.

To complete this simulation 4 chips were chosen from different product lines to see which group's performance was the most appealing and to see how detrimental a poorly chosen FPGA could be. The choice of chips within each product group was primarily determined by which chip was supported in the XPower suite available for use. The chips that were chosen were the Spartan 2, Spartan 3, Virtex 2 Pro, and Virtex 4 LX. For each of these chips the smallest package that could accommodate the design was chosen. During the simulation it was decided to use the Hard Key design because while it typically performed second only to the standard design in power consumption, it was able to fit on smaller chips and the appeal of key storage made this design the most favorable. It was chosen to use a clock speed of 5 MHz, because in power conscious systems especially sensor networks high clock speeds are not typically used.

The results of this series of simulations are shown in figure 4.5 and table 4.4. As is shown the Spartan FPGAs, without the ability to add soft-core processors are much better suited than the Virtex FPGAs for power conscious systems. It is interesting to note that the between the Spartan 2 and 3 chips the Spartan 2 had a low quiescent power consumptions which was 60% of what the Spartan 3 consumed. However when it came to Internal dynamic power consumption the Spartan 2 consumed 741.5% more power

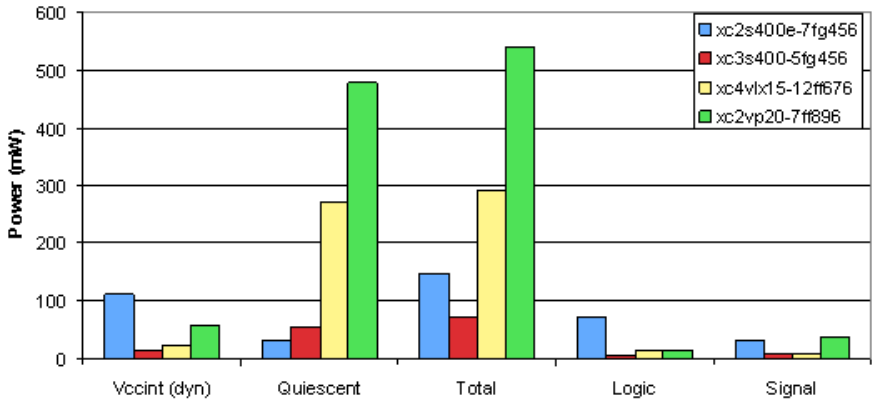

Figure 4.5: Hard Key power consumption for different FPGAs

Table 4.4: FPGA power analysis for Hard Key design

CHIP	Startup	Vccint(dyn)	Quiescent	Total	Logic	Signal
xc2s400e-7fg456	500	111.83	33.6	148.55	71.61	32.63
xc3s400-5fg456	0	15.08	55.50	72.04	4.67	9.59
xc4vlx15-12ff676	0	21.56	270.2	293.11	13.18	7.8
xc2vp20-7ff896	600	57.88	477.5	540.27	16.04	36.98

** All simulation results are measured in milliWatts

than the Spartan 3 and almost double the worst performing Virtex chip. Additionally the Virtex 2 Pro's performance when ignoring the quiescent power consumption was very competitive with the Spartan 3, and if a system was to be designed with a processor and encryption core on a single chip the quiescent power consumption of the Virtex 2 Pro might be acceptable when compared to the cost of connecting and powering two separate chips. Another observation that should be made is that both the Spartan 2 and Virtex 2 Pro requite 500 and 600 mW of power to start up, which if used for a large series of

encryptions may become irrelevant. But in a system were every milliwatt matters and only a few encryptions are required at random times this could be an issue.

These results all help to show how important it is to know the pros and cons of each FPGA that could be used for a design before deciding on a specific FPGA. It is also promising that in the last few years Xilinx and other FPGA manufacturers have started to put more of an emphasis on cutting the power that their chips consume. This is shown by the release of the Spartan 3AN that is based on using non-volatile memory to help reduce static power consumption [13]. Similarly the Virtex 5 has been introduced with lower power I/Os in the LXT version, and being built using 65 nm technologies.

4.5 Speed and Power Analysis

The next area that needed to be examined was if there was an optimal clock cycle time that would allow for the programmed chip to perform with the highest degree of efficiency. This is important to know, because if there is a range of optimal performance compared to power or a general model for performance compared to power then it would be possible to integrate a programmed FPGA into a system and tune its performance to meet the needs of the system without consuming an excessive amount of power. It was decided that the best medium for this analysis was to use the Spartan 3 chip programmed with the Hard Hey design, because of their promising performance in the previous two

simulation sets. Simulations would then be run starting near the designs minimum clock cycle time and increasing until clock cycle time was equivalent to a .5 MHz.

The result of this set of simulations is shown in figure 4.6, figure 4.7, and table 4.4. It can easily be seen that the power decreases exponentially with regard to the clock cycle time and linearly with respect to clock rate. As was noted during the comparison analysis in the previous sections, the quiescent power consumption is always a major factor, but in the simulations running at lower speeds the quiescent power consumption dominates the other components of power consumption. While minimizing the quiescent power would be very desirable it is not possible in modern FPGAs and so when the clock cycle time reaches and then exceeds 5 μSec the power saved does not compensate for the increased amount of time for which the encryption takes.

One of the applications for an encryption core as previously noted is to secure communications between sensor nodes by encrypting packet payload. In applications

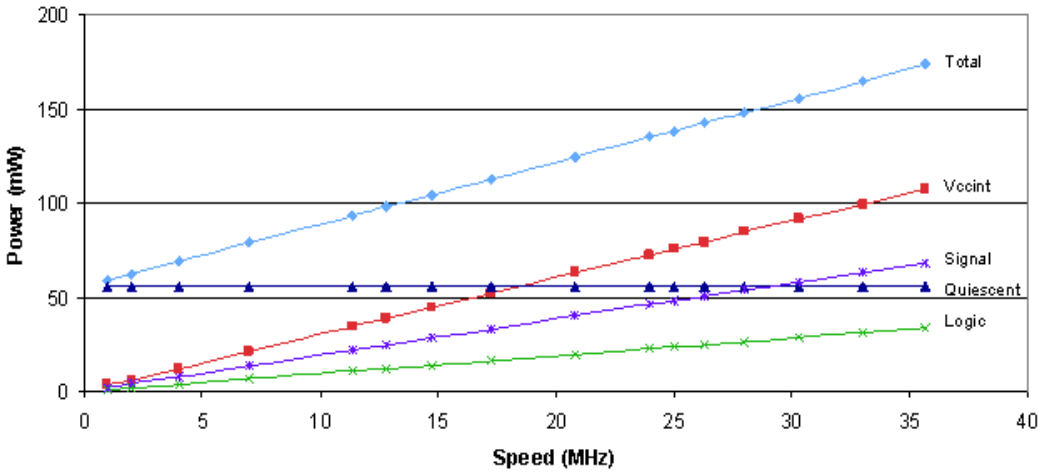

Figure 4.6: Power consumption versus clock rate analysis

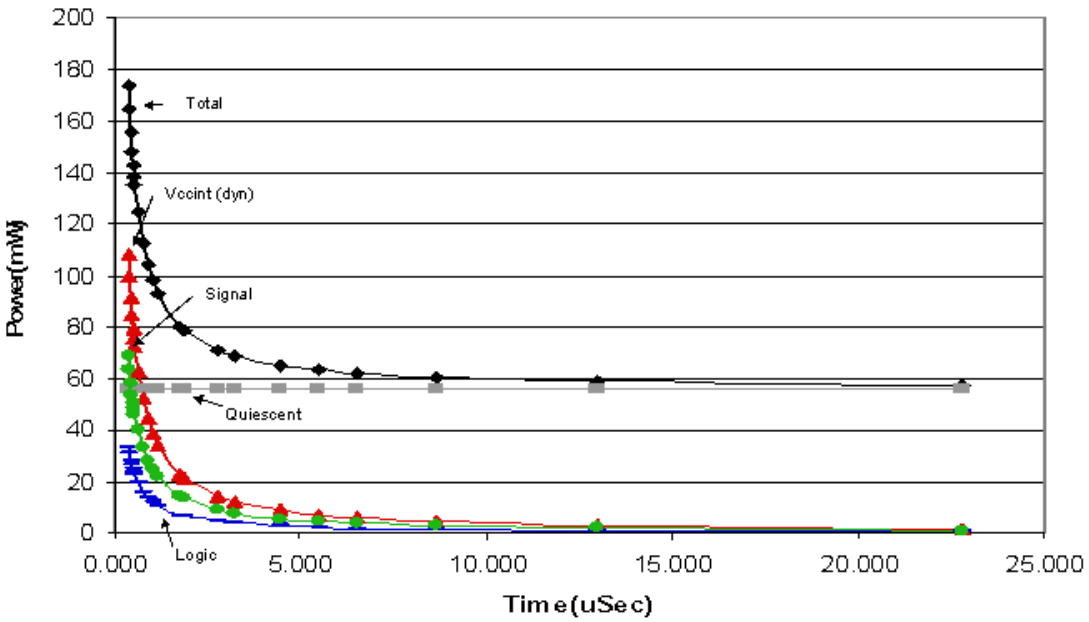

Figure 4.7: Power consumption versus clock cycle time

Table 4.5: Speed vs. power analysis for Hard Key design on Spartan 3

Speed (MHz)	Vccint(dyn)	Quiescent	Total	Logic	Signal
1	3.02	55.5	58.81	0.93	1.92
2	6.03	55.5	62.12	1.87	3.84
4	12.06	55.5	68.73	3.73	7.67
7	21.12	55.5	78.67	6.53	13.43
11.36	34.27	55.5	93.09	10.6	21.79
12.82	38.66	55.5	97.91	11.96	24.58
14.71	44.34	55.5	104.15	13.72	28.2
17.24	51.99	55.5	112.54	16.09	33.06
20.83	62.82	55.5	124.42	19.44	39.95
24	72.38	55.5	134.91	22.4	46.03
25	75.39	55.5	138.21	23.33	47.94
26.32	79.35	55.5	142.56	24.56	50.46
28.01	84.47	55.5	148.17	26.14	53.71
30.3	91.38	55.5	155.75	28.28	58.11
33	99.52	55.5	164.68	30.8	63.28
35.7	107.68	55.5	173.65	33.32	68.47

** All simulation results are measured in milliWatts

such as this low power communication mediums are ideal, and as a result the medium is typically a low bandwidth medium also. This means that to encrypt packets as they are being sent does not require a really fast clock to carry out a large the number of encryptions quickly, but rather a tuned clock which allows for as many packets to be sent over the medium as possible while not consuming too much power. To get a better grasp of what speeds the encryption core would have to be running at to utilize the full bandwidth of a given communication medium, three standards have been researched. The three standards that were chosen are 802.11b, Zigbee, and Wibree. To estimate the maximum number of packets per second, each standard's packet structure was used with 128 bits of data and then divided into the maximum bandwidth. As is shown in table 4.6 the minimum speed that the core would have to function at to send the maximum number of packets possible is still under .3 MHz, which means that the majority of the power consumed by a tuned FPGA would be from quiescent currents.

Table 4.6: Speed Required to fully utilize bandwidth with encrypted packets

Design	Protocol	Maximum Packets/sec.	Cycles per encryption	Min speed (MHz)
Hard Key	Zigbee	1543.2	11	0.016975
	Wibree	6172.8	11	0.067901
	802.11b	24691.4	11	0.271605
Dual Stage	Zigbee	1543.2	6	0.009259
	Wibree	6172.8	6	0.037037
	802.11b	24691.4	6	0.148148

4.6 Component Power Analysis

Now having an insight into how the proposed designs and FPGAs function when compared to each other and how the power consumption of a design varies compared to clock cycle time, it is necessary to examine how power is consumed within the design. To accomplish this test a test bench was made that would that would send the same 128-bit vectors to the logic as was sent to the encryption cores, which was again the Spartan 3. It was also decided to use the encryption elements from the Standard and Hard Key designs with there trigger protections, because the Dual Stage design's performance was not acceptable in the comparison simulations. The components were organized in the same way they were in the encryption core themselves to allow for a 128-bit wide data path. A brief description of how each component was tested and its result is contained in the following sections, figure 4.8 and table 4.7.

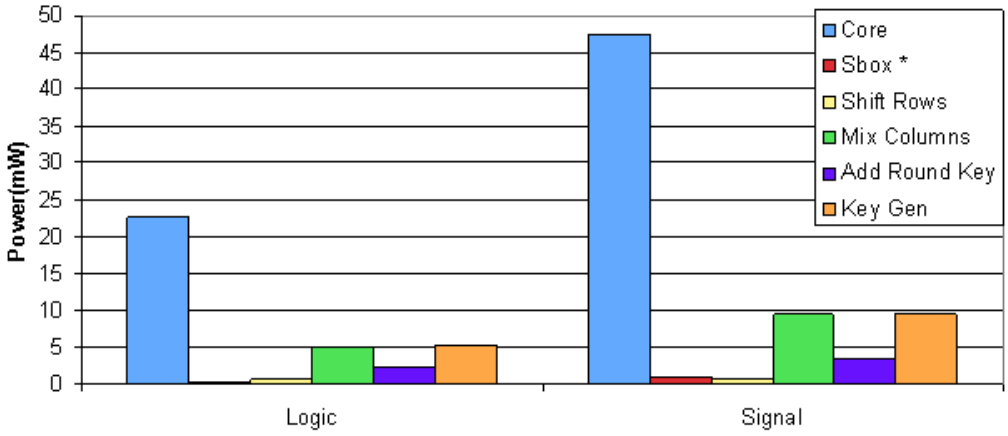

Figure 4.8: Component power consumption for Hard Key design

Table 4.7: Component Power Consumption analysis

Component	Vccint(dyn)	Quiescent	Total	Output	Total - Output	Logic	Signal
Core	77.51	92	176.83	7.32	169.51	23.22	48.14
SBOX	5.96	92	282.36	184.4	97.96	0.26	0.77
Shift Rows	5.98	92	281.14	183.15	97.99	0.26	0.23
Mix Columns	12.72	92	490.83	386.11	104.72	2.55	5.24
Add Key	8.78	92	284.93	184.15	100.78	1.98	1.72
Hard Key gen	11.35	92	664.13	560.79	103.34	2.59	5.22
Component Total	44.79	92				7.64	13.18

** All simulation results are measured in milliWatts

4.6.1 Substitution Box

The substitution box is able to operate independently on 8 bit blocks of data, and as a result all that was needed to test this component was to connect the inputs of the chip to an array of 16 substitution blocks and to then connect their output the FPGA's output pins. It should be noted that when the substitution blocks are optimized for the FPGA they are programmed into memory RAM blocks. The result of utilizing FPGA memory lookup is a very low amount of logic and signal power consumption for the substitution array, which is .26 and .84 mW respectively.

4.6.2 Shift Rows

The Shift rows operation in the designs is simply implemented in the designs by remapping the output of the substitution blocks to the inputs of the mix column blocks. The result should be that very little if any logic power is consumed and that the power

consumed be primarily signal power, because the operation are completed through physical remapping and not a series of logic operations. As expected the signal power (.74 mW) was higher than the logic power (.61 mW), but it was interesting that the logic power was as high as it was. After some consideration it was determined that the logic power consumed is the result of mapping the inputs and outputs of the FPGA to the design, and can be considered the overhead for the other designs.

4.6.3 Mix Columns

The mix column operation is the second most complicated block in the of AES encryption standard. As is explained in section 2.2.4, the operation takes in 4 inputs and by tripling one input, doubling another, and adding the results with the other inputs produces a single 8-bit output. To implement this design 16 of the optimized mix column units were connected to the proper inputs and outputs. Since the logic involved in this operation is more complicated then the other round block components, it was expected that the logic power consumed would be the highest of the round operations, and it was at a value of 5.14 mW. Similarly the signal power was also the highest of the round blocks with a power requirement of 9.58 mW, due to the cost of routing the data from one part of the operation to another.

4.6.4 Add Round Key

The last round operation to be simulated was the add round key block, which takes in the two 128-bit values, one being the output of the mix column block and the other being the round key. These values are added together and the result is the round's output state. To simulate this the 128 bit input to the FPGA was mapped both to the mix column input and the round key input, with the round key's input being scrambled so that no 8-bits would be added with itself unless two separate and distinct 8-bit values contain the same content. With the inputs mapped properly the simulation was carried out and resulted in a logic power of 2.4 mW and signal power of 3.4 mW.

4.6.5 Key Expansion

The final component to be tested was the key expansion block, which takes the previous round's key or initial key and generates the current round key. Due to the higher complexity of the expansion block a new test bench architecture had to be generated which sent the same test vectors to the core along with an round constant to properly simulate the actions of the expansion block. Once the test bench had been designed it was found that the design required too many input and output pins to be simulated with. The solution that was decided upon was to simulate the key expansion block and mix column block using a FPGA large enough to fit the expansion block. Then using the results of these simulations a reasonable estimation of the key expansion's

Table 4.8: Key generation power estimation for component analysis

Chip Simulation	Component	Vccint(dyn)	Quiescent	Total	Logic	Signal
Primary	Mix Columns	19.86	55.5	261.18	5.14	9.58
Secondary	Mix Columns	12.72	92	490.83	2.55	5.24
Secondary	Key generation	11.35	92	664.13	2.59	5.22
Primary	Key generation	17.72	55.5	353.40	5.22	9.54

** All simulation results are measured in milliWatts

power consumption can be made. The results of this process are shown in table 4.8, and as shown the key expansion consumes .04 mW more logic power and .02 mW less signal power than the mix column block. With these results the power consumption of the key expansion block was scaled by the mix column block's primary FPGA power consumption rate divided by the secondary FPGA's power consumption rate. The results show that the key expansion block consumes .04 mW more power than the mix column function when combining the logic and signal power consumption rates.

By totaling the power consumption results of each component a good estimation can be made of where power is consumed in the FPGA. This method shows that about 40% or 9.02 mW of the logic power and 49% or 22.4 mW of signal power are consumed by the control and glue logic in the design. This is a reasonable level because of the need for shifting the states and round key around the FPGA, as well as implementing a multiplexed bypass for the mix column function during the last round of the encryption process. These results show that to further decrease the power consumed by the design

more attention should be paid to optimizing the key expansion process, as well as optimizing the control logic.

Chapter 5

Concluding Remarks

So far this study has provided a thorough introduction of the AES standard, followed by an outline of some of the currently published designs by NIST and on the Open Cores website. A presentation of the proposed designs was presented next, including the Standard Design, Hard Key design, and the Dual Stage design. The analysis procedures and results of the analysis were then presented. In this chapter the insights that can be found by looking at the design analyses will be discussed and what work could be done to improve upon the results shown. In the final section a brief discussion will take place about possible future work and what aspect future research should focus on.

5.1 Discussion

Throughout this study the proposed designs and their analysis have helped to illustrate some significant things. The first thing that can be found by the analysis is the competitiveness of the proposed designs to the reference designs. Both the Standard design and the Hard Key design were able to be optimized to run faster and at much lower power levels than the other designs. The Dual Stage design with its larger footprint and increased complexity however, had such high power requirements that made the design unappealing even with the decreased cycles per encryption. While the reference designs also contained decryption hardware, if partitioned properly they would not have factored into the power consumption rates. The Open Core design's power was so high though, due to the fact that the decryption hardware was not properly partitioned away from the encryption hardware.

Additionally it was shown that while the design is critical for efficient power usage the choice of FPGA is also a major contributing factor. This shows that to properly implement a design with FPGAs it is crucial that the designer ensure that the possible startup power as well as the quiescent power is not too exorbitant for the power supply system. Similarly it has been shown that as the clock cycle time is reduced (frequency increased) the power required for the FPGA to operate rises exponentially. This means that in cases were the number of encryptions is being limited, such as secure

communications, the power saved by increasing the clock cycle time (reducing the frequency) could offer a greater benefit than the cost of adding another clock to the system or decreasing the overall system speed to reduce the FPGA's power consumption.

Finally it was shown that when implementing the AES algorithm for FPGAs utilizing the optimized memory structures provided by the design tools allows for reduced complexity of logic while not greatly affecting the power required by the whole design. This is opposite of how application specific integrated circuits function, since memory takes up more space and raises the power requirements more than implementing a series of logic gates to complete a simple one-to-one mapped substitution.

5.2 Future Work

Now having tested and analyzed the proposed designs there is a clear set of paths that could be taken to gain further insight into how best to design for and utilize FPGAs for efficient encryption cores. The first and most obvious path is to analyze the performance of the Hard Key design when modified to run with various sized data path widths. This would allow for even smaller FPGAs to be used which could help to lower the quiescent power required.

The next area which should be examined is the key expansion and storage architecture. As was shown in the component analysis this part of the design was one of the most demanding blocks, in terms of power consumption. Furthermore the designs

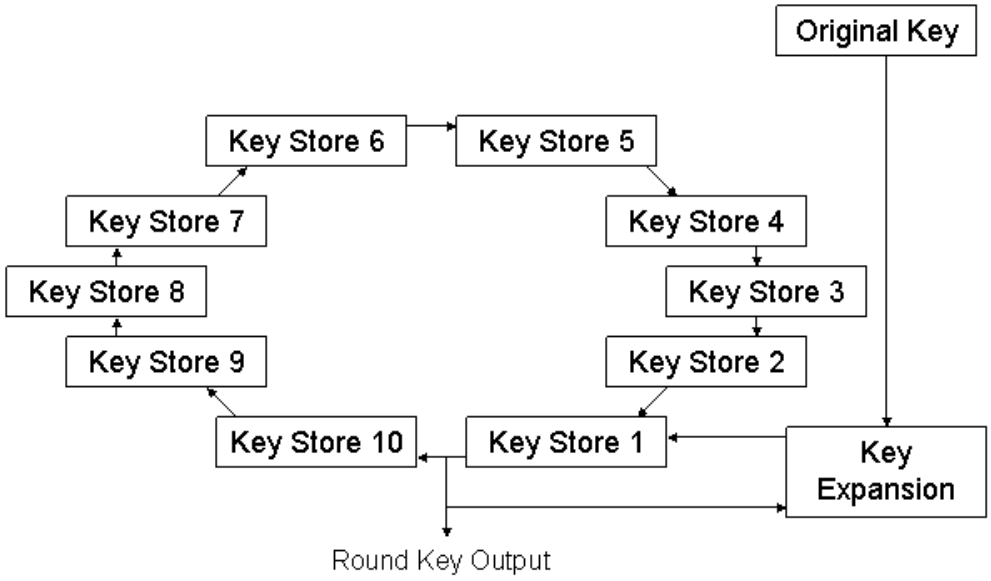

Figure 5.1: Possible circular Key Storage Structure

could be modified so that the key expansion process is carried out just once, and all round keys stored in memory. There are a number of possible ways in which to implement the round key storage, including a series of registers feeding a multiplexer (shown in figure 2.8), or a circular series of registers with a dedicated round key output as shown in figure 5.1.

Any of these key storage modifications could also help to ease the addition of decryption hardware into the design and this is the last area of further research that will be discussed. It is a reasonable assumption to expect that when the basic architecture and optimizations used in design of the encryption logic are applied to decryption hardware design that a balanced performing decryption device will be possible. With this completed the partitioning of the two data paths can be optimized to eliminate

unnecessary power consumption when one part of the design is not needed during a given task. Once that has been accomplished a well balanced and power efficient AES core should be within reach.

Bibliography

[1] Stallings, W. 2003. *Cryptography and Network Security.* New Jersey: Prentice Hall.

[2] National Institute of Standards and Technology (NIST). *Advanced Encryption Standard (AES).* 2001. FIPS-197

[3] Jens-Peter Kaps, Gunnar Gaubatz and Berk Sunar. 2007. Cryptography on a Speck of Dust. *IEEE Computer Society.* 40(Febuary): 38-44.

[4] Satoh, A.; Morioka, S.; Takano, K.; and Munetoh, S. *A Compact Rijndael Hardware Architecture with S-Box Optimization.* Proc. ASIACRYPT 2001, LNCS 2248, pp. 239-254.

[5] Li, H.; Friggstad, Z. 2005. An efficient architecture for the AES mix columns operation. In *IEEE International Symposium on Circuits and Systems.* Vol. 5, pp. 4637-4640. Piscataway, NJ: IEEE

[6] Hamalainen, P.; Alho, T.; Hannikainen, M.; Hamalainen, T.D. *Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core.* Proc. 9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools, pp. 577-583.

[7] National Institute of Standards and Technology (NIST). 2001. *AES.* http://csrc.nist.gov/archive/aes/index.html, Accessed 1 October 2007.

[8] Satyanarayana, H. 2004. *AES128.* OpenCores. Online. Available from internet, http://www.opencores.com/projects.cgi/web/aes_crypto_core/overview, accessed 1 October 2007.

[9] Digilent, Inc. *Programmable Logic.*
http://www.digilentinc.com/Products/Catalog.cfm?Nav1=Products&Nav2=Programmable&Cat=Programmable%20Logic, accessed 1 October 2007.

[10] Xilinx, Inc. *Spartan-3 FPGA Family Data Sheets.*
http://direct.xilinx.com/bvdocs/publications/ds099.pdf. Accessed 1 October 2007.

[11] Xilinx, Inc. *Spartan-3 FPGA.*
http://www.xilinx.com/products/silicon_solutions/fpgas/spartan_series/spartan3_fpgas/index.htm. Accessed 1 October 2007.

[12] Xilinx, Inc. *Xilinx: Virtex-II Pro FPGAs.*
http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex_ii_pro_fpgas/index.htm. Accessed 1 October 2007.

[13] Xilinx, Inc. *Xilinx Spartan-3AN FPGAs.*
http://www.xilinx.com/products/silicon_solutions/fpgas/spartan_series/spartan3an_fpgas/index.htm. Accessed 1 October 2007.