Mitigation of Cold Boot Attack Using an Encrypted Memory Controller

Student Name: Preet Kaur Walia

IIIT-D-MTech-ECE-VLSI-14-16
July 10, 2016

Indraprastha Institute of Information Technology
New Delhi

Advisor
Mohammad S. Hashmi

Submitted in partial fulfillment of the requirements for the Degree of M.Tech. in Electronics and Communication, with specialization in VLSI and Embedded Systems

©2016 Preet Kaur
All rights reserved

Certificate

This is to certify that the thesis titled **"Mitigation of Cold Boot Attack Using an Encrypted Memory Controller"** submitted by **Preet Kaur Walia** for the partial fulfillment of the requirements for the degree of *Master of Technology* in *VLSI & Embedded Systems* is a record of the bonafide work carried out by him under my guidance and supervision in the Security and Privacy group at Indraprastha Institute of Information Technology, Delhi.This work has not been submitted anywhere else for the reward of any other degree.

Dr. Mohammad S. Hashmi
Indraprastha Institute of Information Technology, New Delhi

Abstract

RAMs which are widely used in PCs and laptops are likely to break the popular supposition that data in them is lost whenever power supply is cut down. It has been proved in 2008 by a group of researchers that RAMs are no longer volatile in nature. The data remanence behaviour of DRAMs have allowed to retrieve memory contents when the power has been switched off by an attack known as cold boot attack. In cold boot attack, memory can be frozen using a refrigerant and then removed from the computer. It is then quickly placed into a specially designed system that reads out its content, targeting encryption keys and other sensitive information. Also, the standard mechanisms for protecting sensitive data on laptops, smartphones, PCs and Macs is no longer safe and secure. Several design practices have been discussed in the thesis which are being adopted to mitigate such attacks.

Mitigation of Cold Boot Attack by providing encryption support in the memory controller is a promising technique which has been proposed in the thesis. The proposed memory security model protects digital content and software stored in untrusted system memory from physical tamper. The thesis introduces a set of architecture innovations that aim for the implementation of the proposed security model. Further, a new key storage mechanism is proposed. Also, analysis on the overhead occurred due to the adoption of this scheme has been discussed.

Acknowledgments

Towards the end of my M.Tech. degree, I would like to pay my gratitude to several individuals who contributed from various perspectives. First of all, I would like to express my profound gratitude to my thesis supervisor, Dr. Mohammad S. Hashmi for his support, direction and motivation all along the way. He has been a great source of inspiration for me. I would like to thank the director Dr. Pankaj Jalote and all my professors at IIITD for providing a positive learning atmosphere to work. I also take this opportunity to express a deep sense of gratitude towards Mr. Arpan Jati for his support and encouragement for conquering every hurdle that I have encountered throughout the process. My regards to Ravneet Saluja and all my friends at IIIT-D who made this journey a wonderful one. Last but not the least; I would like to thank my Parents for supporting me spiritually and emotionally.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Security is a vital concern for everyone in day to day life. Information Security is a problem being faced by modern organizations considering, many organizations today are fully dependent on information technology for survival. The increasing variety of threats and fierceness of cyber-attacks has made protecting information a complex challenge. As the records show, between 2002 and 2007, 773 breaches of US organizations were reported with a total of 267 million private records lost. Over 42% of these breaches were a result of lost or stolen hardware including laptops, PDAs and portable memory devices [1].

To exploit data sensitivity, many attacks have been innovated and explored. Some of them include Bus snooping, DMA fire-wire attack and Cold boot attack. Cold boot attack is an attack developed by researchers in Princeton University in 2008 which is of high concern.

Many solutions have been developed to prevent and counter attack in such situations. Three general approaches include complex hardware enhancements, software enhancements and adding specialized industrial devices. Most commonly used hardware enhancement being done to prevent such attacks is memory encryption. Memory encryption (ME) has been used at the center of operating system designs to provide privacy of information.This forms the motivation for the proposed work which is discussed elaborately in the next section.

1.1 Motivation and Objective

Several applications store encryption keys in memories for the purpose of system security. This key comprises of the passwords and other confidential information entered by the users. These can also be loaded directly from the memory. Even when the system is idle or in a sleep mode, the memory can be manually handled for the extraction of the keys stored in them. This breach of security and confidentiality is one of the major reasons of apprehension in the modern digital world. One of the techniques used by hackers and cyber-criminals for extraction of sensitive information from PCs and laptops is cold boot attack.

In cryptography, a cold boot attack also known as platform reset attack which is a type of side channel attack in which an attacker with physical access to a computer is able to retrieve encryption keys from a running operating system after using a cold reboot to restart the machine. Cold-booting refers to rapid switching of power without letting the operating system to shut down cleanly. The attack relies on the data remanence property of DRAM to retrieve memory contents that remain readable for few seconds even after power has been removed.

This came to light in 2008 when a group of researchers at Princeton University proved that the standard mechanism for protecting sensitive data on laptops, smartphones, netbooks, PCs and Macs is no longer safe and secure.

Random access memory was thought to be volatile. This property means that anything stored in random access memory is temporary–when the machine is switched off and the memory loses power, the data is soon lost. But, the Princeton group showed that data stored in the random access memory turns out to be preserved over a period of many seconds after it loses power. They also showed that cooling the memory can extend this period to many minutes and possibly hours. Cooling can be done by using a refrigerant such as an inverted can of compressed air Fig 1.1 [2].

Figure 1.1: Inverted can of compressed air

This section discusses the work done by Princeton University researchers [2]. It explains the execution of cold boot attack. Also, it discusses briefly about the techniques which have been used by the researchers to extract the keys and data present in memory after the cold boot attack.

1.1.1 Execution of attack

- Memory was filled up with pseudo-random pattern.
- Power was cut off for memory chips for different time periods.
- Dusting air canned was directly sprayed on the chips.

- At different temperature, patterns were read by varying the time periods of refresh.
- Error rate of different samples were measured. Error rate has been defined as number of bit errors in each sample divided by the total number of bits measured.
- Different results were obtained for different memory samples as shown in table 1.1 [2] .
- The results showed that 1% of bits decayed in the samples even after 10 minutes of power cut down.
- For temperature decrease, liquid nitrogen was also be poured on the chips. Results further showed that there was only a decay of 0.17% after 60 minutes. The results are shown in 1.2 [2].

Table 1.1: Test system used experiment

	Memory Type	**Chip Maker**	**Memory Density**	**Make/Model**	**Year**
A	SDRAM	Infineon	128Mb	Dell Dimension 4100	1999
B	DDR	Samsung	512Mb	Toshiba Portege	2001
C	DDR	Micron	256Mb	Dell Inspiron 5100	2003
D	DDR2	Infineon	512Mb	IBM T43p	2006
E	DDR2	Elpida	512Mb	IBM x60	2007
F	DDR2	Samsung	512Mb	Lenovo 3000 N100	2007

Table 1.2: Effect of cooling on error rates

	Second w/o power	**Error % at operating temperature**	**Error % at -50 degree C**
A	60	41	(no errors)
	300	50	0.000095
B	360	50	(no errors)
	600	50	0.000036
C	120	41	0.00105
	360	42	0.00144
D	40	50	0.025
	80	50	0.18

1.1.2 Reading the contents of memory

Tiny special programs were used by the researchers to correctly dump data present in memory to external medium. One of the techniques to successfully read the data present in memory is by providing lightweight booting and dumping the contents of pre-boot memory to a file so that

content does not get overwritten. This can be done by using USB drives. Also special plugins were developed which saved the data to a file before providing to the BIOS. [2]

1.1.3 Keys Regeneration

Next step is to look for keys from the data obtained from the memory. Brute force technique can be used to extract the keys. In this technique, every possible sequence can be tested and each such sequence is considered to be a potential key. But this technique is naive and time consuming. Therefore, special algorithms were developed to extract the key. These algorithms were used to attack BitLocker which comes with Windows Vista, FileVault, which comes with Mac OS X and dm– crypt which is used with Linux [2].

1.1.4 Existing Solutions

Various methods have been developed to overcome the security issues caused by cold boot attack. Some of them have been listed below:

Full memory encryption
In full memory encryption, everything present in RAM is in encrypted form. The keys required for encryption can be stored in CPU or hard disk and CPU can encrypt and decrypt data when required from the memory. This technique has been implemented in Microsoft XBOX and Intel processors. PrivateCore is a product which is based upon full memory encryption. Although FDE adds to overhead due to the time required in encryption and decryption but such a design can be used when security is of prime concern [3].

Dismounting encrypted disks
Cold boot attacks can be prevented by removing or dismounting the disk when not in use. This will prevent the case when authorized personnel are not present and attacker tries to steal the disk. But this technique has many disadvantages. Most of the times operating system is in active mode as in the case of sleep mode and therefore disk cannot be mounted. Therefore our system might be at risk.

Hardware enhancements
Special chips can be placed on the mother board to encrypt the data being stored in memory or for storing keys between CPU and memory. One of the earliest papers, highlights an execute only memory (XOM) architecture [4].

Software enhancements
Changes in software can be used to diminish the effects of cold boot attack without much change in hardware. Operating system based encryption technique can be used to prevent such attacks.

Advanced encryption modes

Bitlocker is a widely used tool which uses advanced encryption schemes to prevent cold boot attacks. Bitlocker uses TPM(Trusted Platform Module which is attached to a typical motherboard via the low pin count (LPC) bus. TPM chip allows to store key securely and provides chain of trust booting. But, it has been shown that TPM is susceptible to bus snooping and injection attacks. Also, at the time of functioning TPM copies key to RAM making it highly vulnerable to cold boot attack [5].

Power management
One of the solution can be to move the computer to completely powerless state when it is not being unattended by authorized personnel can alleviate cold boot attacks. .

Register-based key storage
In operating system nowadays, kernel can be modified in such a way that keys are stored inside registers in CPU rather than in RAM.

Cache-based key storage
One way to counter the attack could be to store the keys only in the computer cache, instead of RAM. However, caches are hard to control and one needs to make sure that keys are really frozen in the cache and are never written to the RAM

1.2 Terminologies

This section aims at presenting some important terminologies along with their definition.

- **Encryption**: Encryption is the process to convert data into a form called ciphertext, which cannot be understood if access to password or key is not available to decrypt it back to its original form.
- **Decryption**: Decryption is the process of conversion of ciphertext to plaintext which is the data in its original form.
- **Delay**: Delay stipulates how long will it take to transfer data from one entity to another.
- **Handshaking**: Handshaking in computer architecture is defined as a method by which initiation of communication process begins between two processor or peripherals.

1.3 Outline

The remaining thesis is organized as below:

Chapter 2 presents the preliminaries. It discusses the functionalities of Xilinx memory controller block. It also showcases the brief concepts of AXI bus.

Chapter 3 presents a novel design to mitigate the cold boot attack and the encryption of ARM AXI bus.

Chapter 4 This chapter concludes the thesis and suggests the possible directions for future work.

Chapter 2

Preliminaries

2.1 Introduction

Memory controller block and bus system constitute the main parts of any CPU-memory based architecture. This chapter explains ARM AXI bus specifications. AXI bus is a part of ARM AMBA (family of architecture developed in 1996) [6]. The design proposes the encryption using ARM AXI bus. Therefore, this section explains the stipulations and concepts of AXI bus, its signals and channels. Additionally, this chapter explains the working of Xilinx memory controller block and its crucial signals which have been critical in the design proposed.

2.2 AXI bus

AMBA(advanced microcontroller bus architecture) is an open standard ARM based architecture. It provides the facility to connect various IP (Intellectual block) on a chip efficiently. Advanced extensible interface(AXI) is the third generation of AMBA interface and is used mostly where high speed and high bandwidth is required. [6].

It provides the following features:

- It supports burst mode. In burst mode, read and write take place continuously depending upon the burst size and consecutive addresses are calculated from the first address available on the bus.
- The architecture provides low latency and high frequency transactions.
- It is flexible and fits in the existing CPU architectures easily.
- AXI is used to communicate between memory and CPU through memory controller block.
- It supports DMA (Direct memory access).

- AXI transactions can be converted to transactions acceptable by different functional blocks on chips without the use of complex bridges.
- AXI facilitate the completion of out of order transactions.
- Separate buses are available for address and data signals.
- Simple and easy to understand [6]

2.2.1 Channel Handshaking

Handshaking refers to establishment of rules so as to facilitate communication between two entities. Channel handshake in AXI takes place with the help of two signals namely VALID and READY. These signals help in the flow of information between the master(sender) and slave (receiver). The VALID signal goes high indicating that VALID data or control information is available. READY signal indicates that the receiver is ready to accept the data and control information. The transfer of information takes place only when both VALID and READY signal are high thus indicating that handshaking has taken place.

2.2.2 Channels

There are five channels in AXI architecture. These have been briefly discussed below.

- **Read channel** The read data channel is responsible for transferring the read data from the slave back to the master [6]. Read data channel signal are descripted in the table below:

Table 2.1: Read Data Channel Signals

Signal	Description
RDATA[31:0]	This 32 bit data bus symbolizes the data read from the memory. It can be of varying lengths of 8,16 up to 128 bits
RLAST	This signal goes high when the last data is being transferred in burst mode
RVALID	This signal is sent by slave (active high) when data which is to be read is available on bus and otherwise is low
RREADY	This signal is sent by master (active high) showing it is ready to accept the read data and control information

Address Read Address read channel is used to send the byte address from where read operation is to be performed. AXI works in burst mode [6]. Burst modes available are of

three types: wrapping, incrementing and non- incrementing bursts. Address Read signals are descripted below :

Table 2.2: Address Read Channel Signals

Signal	Description
ARADDR[31:0]	This is the starting byte from where read is to be done.
ARLEN[3:0]	This signal shows the number of data words read in a burst mode
ARSIZE[2:0]	This signal gives the burst size of the trsnsaction.
ARBURST[1:0]	This signal signifies the type of burst mode of the transaction.
ARVALID	This signal generated by master goes high when valid address is available and remains high till the slave sends the ARREADY signal.
ARREADY	This signal is sent by master where high signifies that slave is ready to accept the address.

- **Write** The write data channel is responsible for writing data from the master to the slave. Write data channel signals are descripted in the table below [6]:

Table 2.3: Write Data Channel Signals

Signal	Description
WDATA	This is the write data bus. It can be of varying width from 8 upto 128 bits.
WLAST	This signal goes high when the last data is being transferred in burst mode.
WVALID	This signal goes high when valid data sent by master is available on the bus.
WREADY	This signal goes high when the slave is ready to accept the write data.

- **Address Write** Similar to the address read channel, write address channel provide the write address (from where data is to be written) [6].Address write channel signals are descripted in the table below :

Table 2.4: Address Write Channel Signals

Signal	Description
AWADDR[31:0]	This is the address bus. In the write burst mode, address of first transaction is given on this bus and consecutive addresses are calculated by AXI itself.
AWLEN[3:0]	This signal shows the number of data words written in a burst mode.
AWSIZE[2:0]	This signal gives the burst size
AWBURST[1:0]	This signal signifies the type of burst mode of the transaction (Incremental, fixed).
AWVALID	This signal generated by master goes high when valid address is available on the AWADDR bus. This signal remains high till the slave sends the AWREADY signal.
AWREADY	This active high signal is sent by slave signifying that slave is ready to accept the address.

- **Response channel** The write channels use this channel to confirm if write has happened successfully or not [6]. Response channel signals are descripted in the table below :

Table 2.5: Write Response Channel Signals

Signal	Description
BRESP[1:0]	This signal is sent by the slave indicating the write transaction has occurred successfully or not. OKAY indicates successful write
BVALID	This signal is sent by the slave indicating that valid response is available to be,sent to the master
BREADY	This signal is sent by master specifying if it is ready to accept the write response from the slave or not

2.2.3 Communication between AXI channels

In address read channel the ARVALID goes high(sent) indicating valid address is available on the bus. It must remain asserted up till the slave sends the ARREADY signal.Thus, indicating that it is now ready to accept the address/control information. The RVALID signal sent by the slave should go high when the slave is ready to send the data read to the bus. RREADY is then sent by the master indicating that it can now avail the data sent by the slave. Also, RVALID should remain high until the RREADY signal goes high. It might also be possible that default value of RREADY is already high [6].

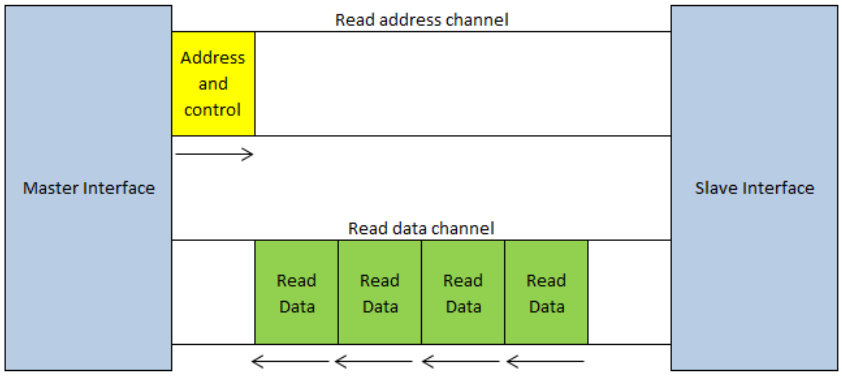

Figure 2.1: Read channel

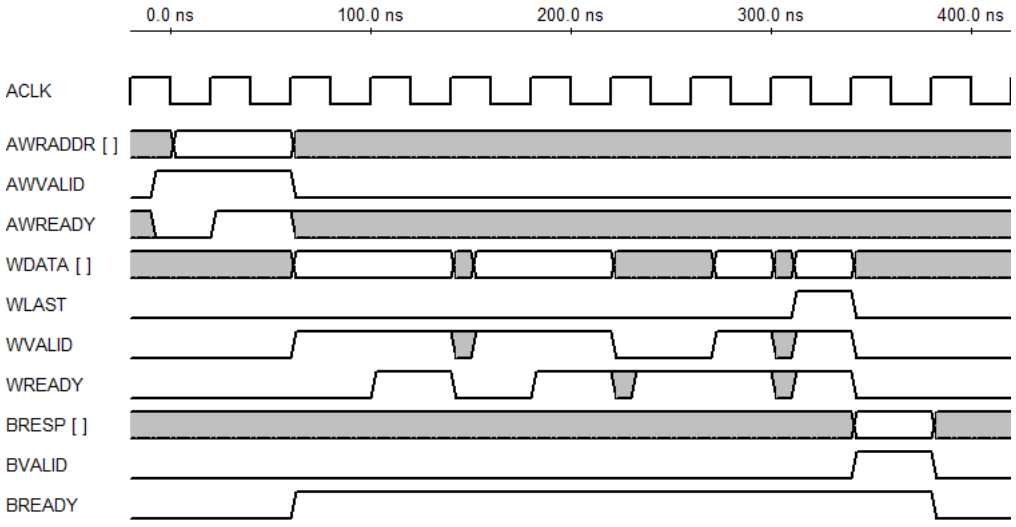

Figure 2.2: Read Burst

In address write channel the AWVALID goes high indicating valid address is available on the bus. It must remain asserted uptill the slave sends the AWREADY signal indicating that it is now ready to accept the address/control information.The default value of ARRAEDY signal can be high or low [6].

The WVALID signal sent by the master should go high when the slave is ready to send the data read to the bus. But the actual transfer of data takes place when WREADY signal goes high. WREADY is sent by the slave indicating that it can now avail the data sent by the slave. Also, WVALID should remain high until the WREADY signal goes high. It might also be possible that default value of WREADY is high.The WLAST signal shows that the last data is being written [6].

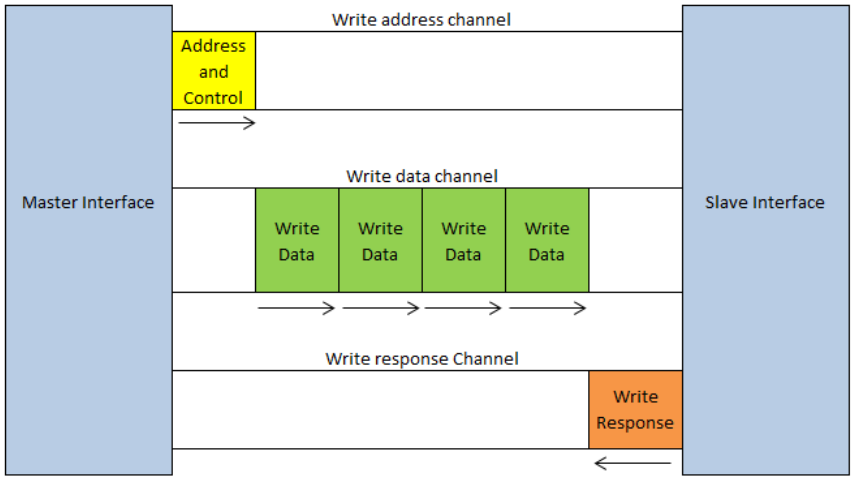

Figure 2.3: Write channel

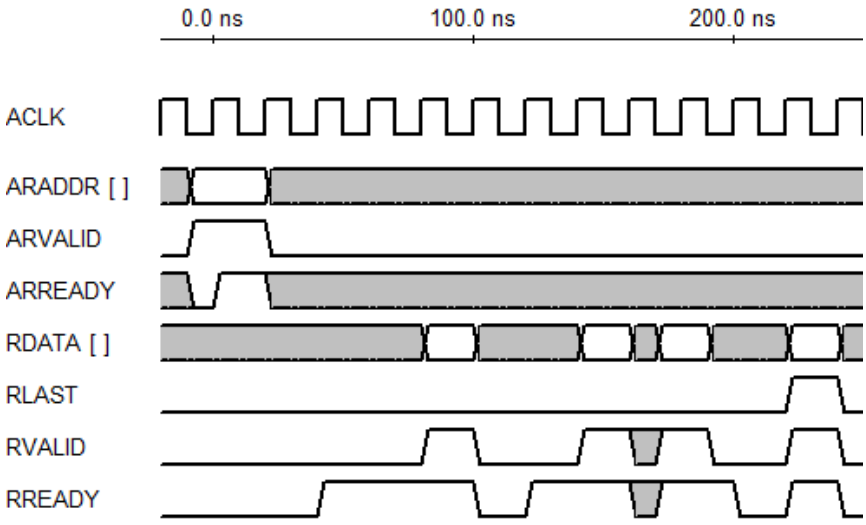

Figure 2.4: Write Burst

The slave then asserts the BVALID signal when it has received the data written to it by master. BRREADY goes high indicating that master is ready to accept the response information from the slave.

2.2.4 Dependencies between different handshake signals are discussed

To prevent and avoid deadlock situation, it is important that following two situations must be taken care of:

- VALID signal should be independent of the Ready signal. VALID may or may not be high

before the assertion of ready signal [6] .

- Ready may wait to be asserted before the assertion of valid signal [6].

Read Handshake dependencies

- ARVALID may or may not wait for ARREADY [6].
- RVALID can only be high(sent by the slave) after both ARREADY and ARVALID are asserted high by the master [6] .

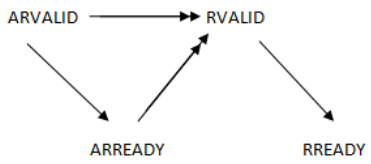

Figure 2.5: Read Handshake Dependencies

Write handshake dependencies

- The master should be independent of the assertion of AWREADY or WREADY before it makes WVALID or AWVALID high [6].
- The slave may or may not wait for AWVALID or WVALID or both , before it makes the signal AWREADY high [6].
- BVALID can only be made high only after both WVALID and WREADY have been asserted [6] .

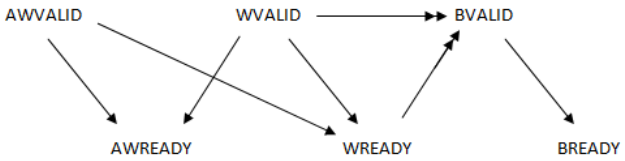

Figure 2.6: Write Handshake Dependencies

2.3 Memory Controller Block

A memory controller block is defined as ondie-chip which manages the flow of write transactions from CPU to memory and read transactions from memory back to the CPU (read). In Spartan 6 FPGAs MCB are usually used for interfacing between memory standards and CPU.

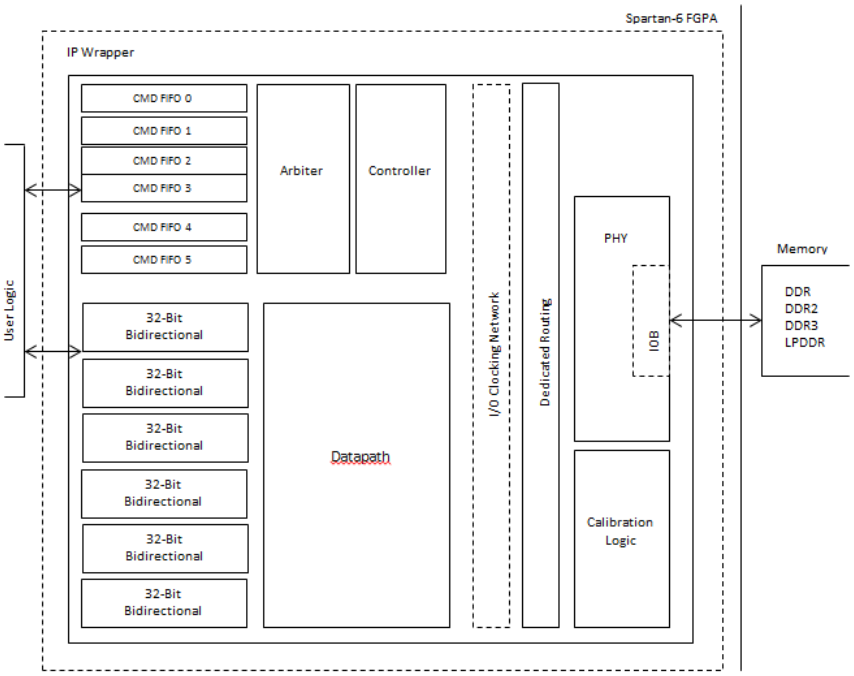

Figure 2.7: Memory controller block functional view

The key features of Xilinx MCB design used are as follows:

- Four MCB cores can be supported.
- DDR,DDR2,DDR3 and LPDDR memory interface support.
- 2 Bidirectional and 4 unidirectional port options.
- Maximum data bus of width 128 bits is available.
- Can work up to 400Mhz.
- Maximum memory size of 4GB and bandwidth of 12.8 GB/s can be made available.
- Pin-out configuration is already defined.

2.3.1 Memory controller block functional view

- **User logic** User interface contains the mandatory signals required for interfacing support between the FPGA and MCB. Some of the important signals between the logic and MCB are the refresh signals, clocking and reset signals. Also, user logic contains signals for providing calibration (startup) to the memory module [7].
- **Arbiter**As already stated this MCB can support 6 ports. Arbiter is responsible for deciding which MCB port to select and whose request has to be serviced first [7].

- **Controller**The controller converts the request made by user to the signals and instructions understandable to memory module. [7].
- **Command FIFO** There are six command FIFOs in the memory controller block. Command FIFOs are responsible for storing and converting the commands received from the user to the instructions compatible with the memory module. The command FIFOs have a depth size of four and store instructions such as read burst, write or refresh [7].
- **Datapath** Here, datapath is responsible for providing path between the read or write instructions between MCB and memory. The datapath FIFO have depth of size 64 which allow 64 data words to be stored starting from a given address [7].
- **Calibration logic** There is a startup sequence for the MCB after the clocking and reset requirements are fulfilled which is carried out by the calibration logic [7].
- **Clocking network** The clocking network consists of the PLL and clock. Clocking network converts the clock generated by the FPGA to the clock required by MCB modules [7].
- **Physical Interface** Physical Interface converts the controller instructions into the actual timing relationships and DDR signaling necessary to communicate with the memory device [7].

2.3.2 MCB operation

Here, the complete MCB operation is explained along with its three basic datapath functions: read, write and command. The table 3.1 [7] descripts the various operations MCB performs and their corresponding instruction code.

Table 2.6: MCB Instructions and codes

Instruction	Code
Write	000
Read	001
Write with auto precharge	010
Read with auto precharge	011
Refresh	1xx

2.3.3 Command Signals

Table 2.7: Command Path Signals

Signal	Description
Px_cmd_addr[29:0]	This signal give starting byte addresses for read and write transaction.
Px_cmd_bl[5:0]	This signal tells the number of user words being written or read. This signal varies from 0 to 63 user data words and can be of 32 bits,64 bits or 128 bits each.
Px_cmd_clk	This is the command block input clock
Px_cmd_empty	This output signal from the MCB signifies that there is no command yet present in the command FIFO although commands may be in flight.
Px_cmd_en	This is write enable signal for the commands FIFO.
Px_cmd_full	This output signal shows that the command FIFOs are full and no more command can be entered.
Px_cmd_instr[2:0]	This signal depicts the operation requested by user to the MCB module whether it is a read, write or refresh.

Command path timing

The instruction type, address, and burst length for the requested transaction are all loaded into Command FIFO. As it can be seen from the timing diagram, the px_cmd_en signal goes high as soon as the write command is sent by the user. Note that command is accepted at rising edge of the enable signal when the px_cmd_full signal is low. The px_cmd_instr contains the instruction (read or write) and px_cmd_bl shows the burst length (number of data words) being written or read. As soon as first command enters the command FIFO the px_cmd_empty signal goes low [7].

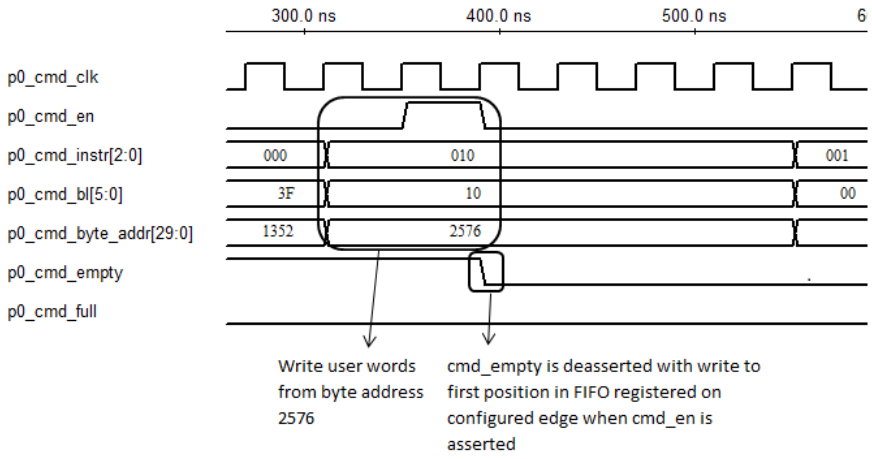

Figure 2.8: Command Path Timing(Write)

2.3.4 Write Signals

Table 2.8: Write Datapath Signals

Signal	Description
pX_wr_count[6:0]	It shows the number of user data words (up to 64) in the write FIFO that need to be written to memory.
pX_wr_data[PX_SIZE-1:0]	The signal contains the write data value input by the user that needs to be written to memory. PX_SIZE can be 32, 64 or 128 bits.
pX_wr_empty	This active high output signal indicates whether there is write data present in write FIFO or not.
pX_wr_en	This is the write enable signal for the write transaction.
pX_wr_full	This active high signal indicates that write FIFO is full and no more data can be entered into the FIFO

Write operation

To execute a write transaction, sequencing between the command and write data-path is required. Firstly, the write FIFOs are loaded with sufficient data words which is consistent with the value in burst length signal shown in figure 3.3. If the data words entered by the user is lesser than the burst length value ,the underrun condition occurs.Px px_wr_count indicates the number of datawords in the write FIFO. The data is written at the rising edge of px_wr _clk when px_wr v en is active. Px _wr_full should be low to accept data in write FIFO and px_wr_empty goes

low as soon as first data is written to the memory as shown in Figure 3.3 [7].

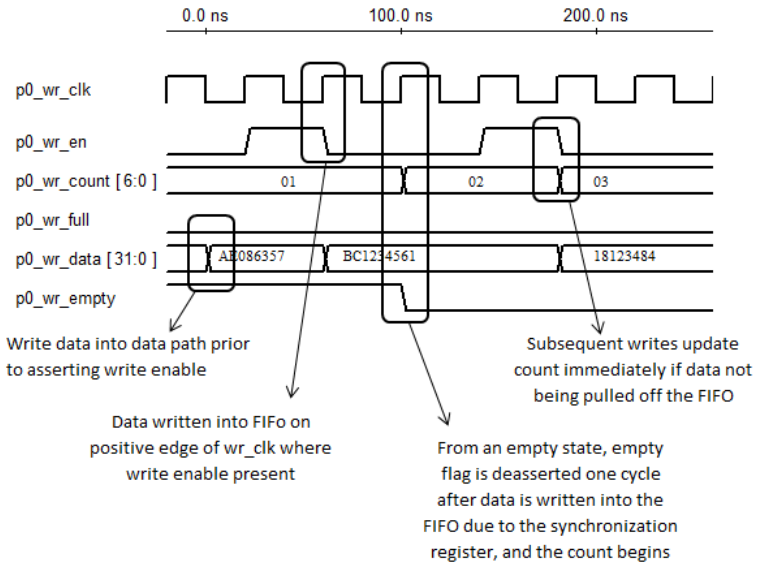

Figure 2.9: Loading the Write Data FIFO

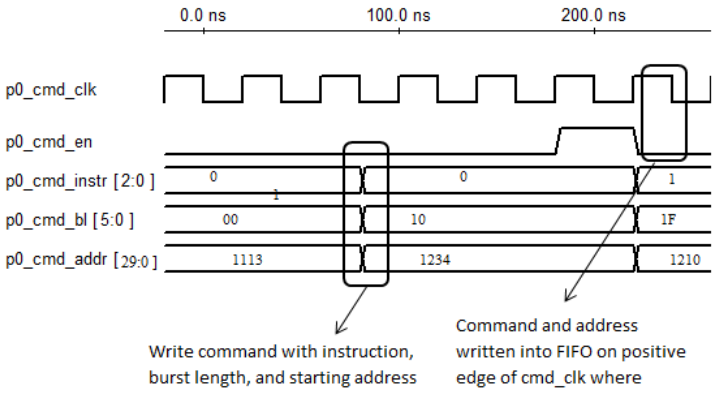

Figure 2.10: Entering the Write Request into Command FIFO

2.3.5 Read Signals

Table 2.9: Read Datapath Signals

Signal	Description
pX_rd_en	This is the read enable signal for the read transaction.
pX_rd_data[PX_SIZE-1:0]	The signal contains the read data value input by the user read from the memory. PX_SIZE can be 32, 64 or 128 bits.
pX_rd_full	This indicates that read FIFO is full and cannot accept more read data from the memory.
pX_rd_empty	This active high output signal indicates presence of read data in read FIFO.
pX_rd_count[6:0]	It shows the number of user data words in the read FIFO.

Read operation

The burst length indicates the number of data words that are to be read, therefore read data FIFO should be consistent with the burst length value otherwise overflow condition occurs as shown in fig 3.5. The data available on the Px_rd_data bus can be accessed by the FPGA logic (Fig 2.12). The Px_rd _en signal goes high and data is loaded from the read FIFO to the FPGA . The Px_rd_count value gets updated accordingly.

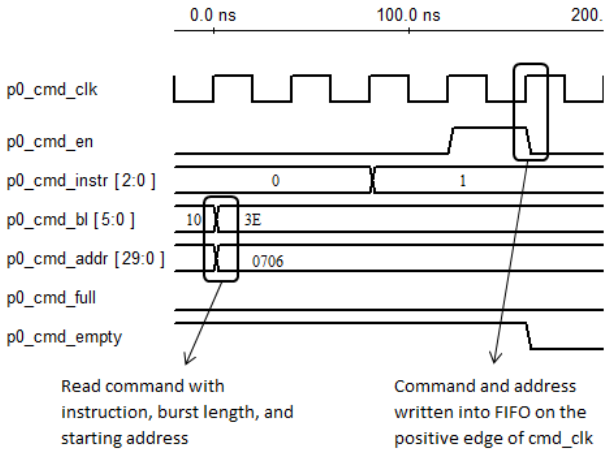

Figure 2.11: Entering the Read Request into Command FIFO

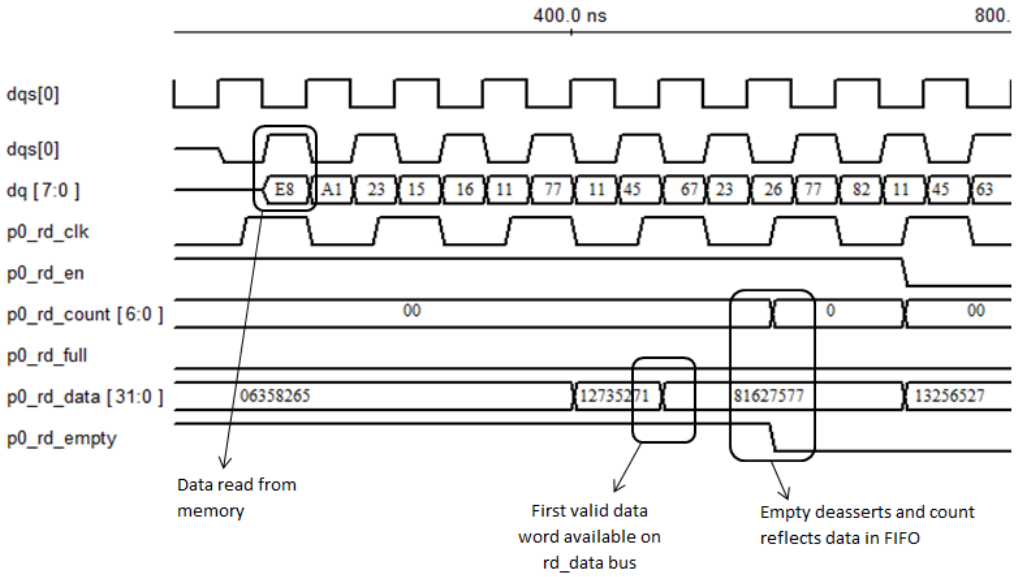

Figure 2.12: Read Data Returning from the Memory Device

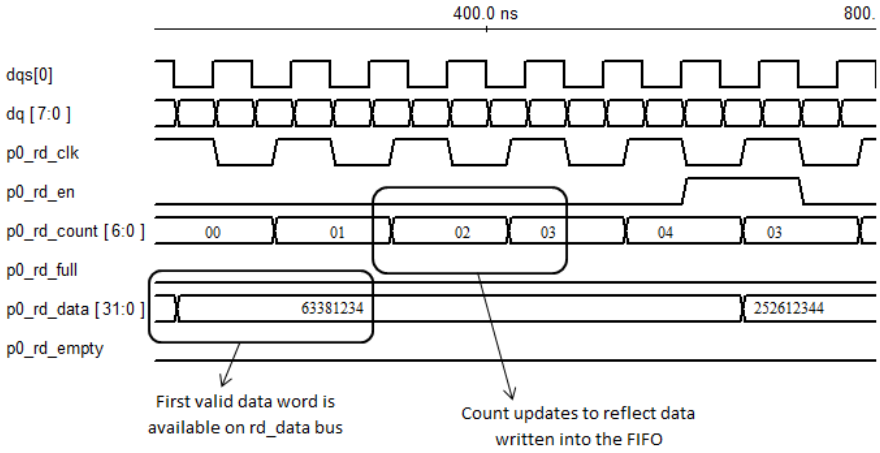

Figure 2.13: Transferring Read Data into FPGA Logic

2.4 Advanced encryption Algorithm

Advanced encryption algorithm is an encryption algorithm which is specially designed for the security and protection of electronic data. AES was developed and invented by cryptographers Joan Daemen and Vincent Rijmen in Belgium and therefore is also known as Rhindjael algorithm. AES is a 128-bit symmetric and block cipher.The block size of AES is fixed to be of size 128

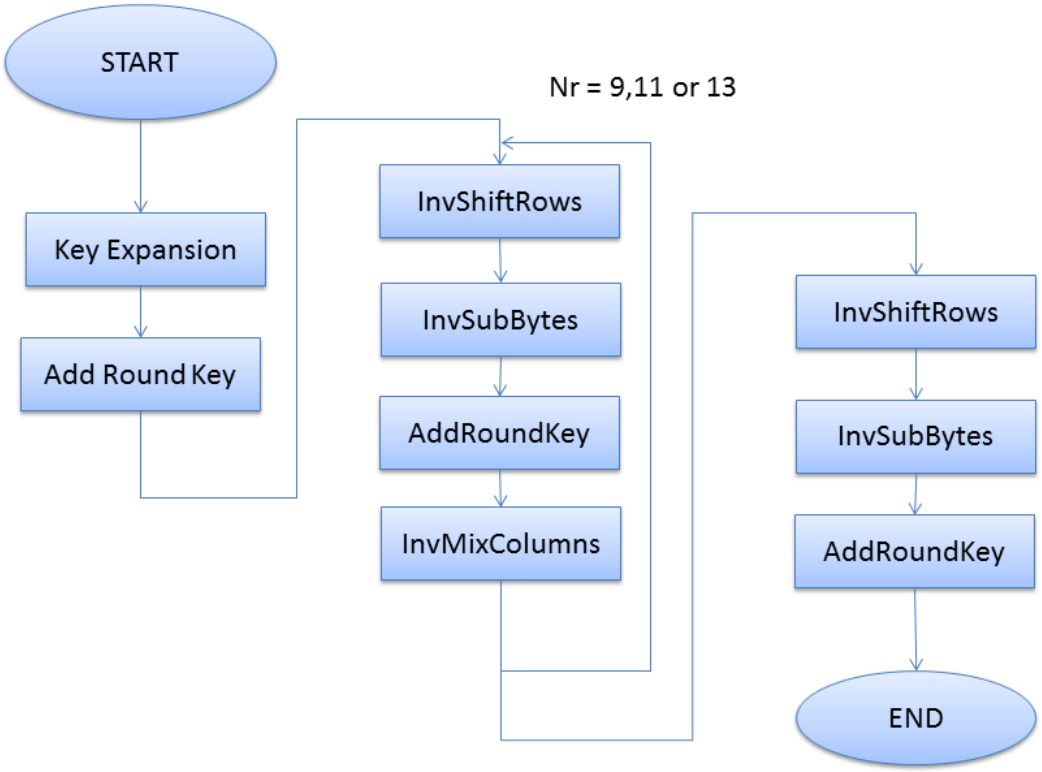

Figure 2.14: AES Algorithm

bits. The key expansion requires different number of rounds for keys of different sizes .

Chapter 3

Proposed Work

3.1 Introduction

Cold boot attack is a high priority attack in the modern digital world. Many methodologies to counter attack the cold boot problem have been discussed in chapter 1. Disk encryption seems to be a promising technique which can be adopted to prevent such attacks. But, many products using disk encryption have been proved inefficient by the Princeton group including Bitlocker, True Crypt and FileVault [2]. How to protect confidentiality and integrity of random access memory remains a great challenge.

In this chapter a novel design has been proposed to mitigate cold boot attack by using disk encryption technique. This design modifies the existing architecture on which most of the modern processors are based which will be further discussed in the subsequent sections. This chapter has been divided into three main subsection. It presents the basic platform architecture our solution is targeted for. It also shows the types of attacks the solution is aimed to prevent and detect.In the first subsection the design methodology has been proposed. The general approach which can be followed to alleviate cold boot attack has been showcased. In the second subsection the proposed design has been implemented and analysed on Xilinx real time environment.

3.2 Memory Encryption based a novel design methodology

3.2.1 Architectural View

In this section, the detailed security model and architectural support for mitigation of cold boot attack has been presented.

The generic approach based upon the modern processor is shown in figure 3.1. The CPU commands the memory controller to perform the appropriate functions - read or write. The memory

controller block accepts the instruction from CPU and carries out the subsequent tasks.

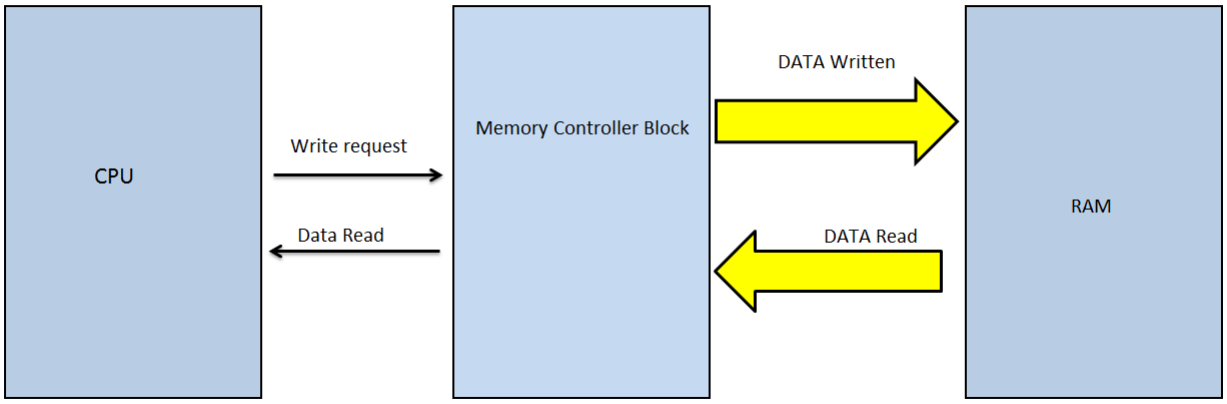

Figure 3.1: Read-Write(General approach)

In context to the cold boot attack, the architecture behaves in a different manner as shown in figure 3.2. The flow of data from CPU to the RAM is handled by the memory controller. Here, the security model comes into picture. The scheme proposes that data when being written to RAM is encrypted by the memory controller in the case of write operation.Therefore, the RAM behaves as encrypted RAM and the data present in encrypted form in RAM. Digging deeper into the architecture, in the case of write operation as shown in Figure 3.3 when write request has been placed by the CPU the encryption module present in the MCB encrypts the data and writes on to the RAM.

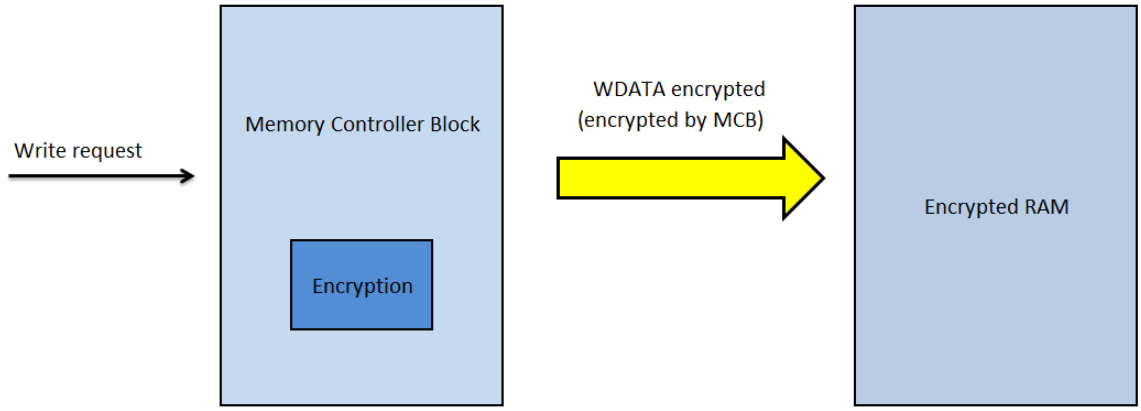

Figure 3.2: Write Request

The opposite sequence occurs in the case of read operation. The CPU when requests a read to

the memory controller. The memory controller reads the data from the RAM,decrypts it and make the original data available to the CPU. Note that the data on the bus is in original from. The modifications happen to be between the MCB and RAM level. While reading, the CPU sends the read request to the MCB to read the data from RAM. The MCB reads the encrypted data from RAM. The decryption module present in MCB decrypts the data from RAM and returns the original data to the CPU.

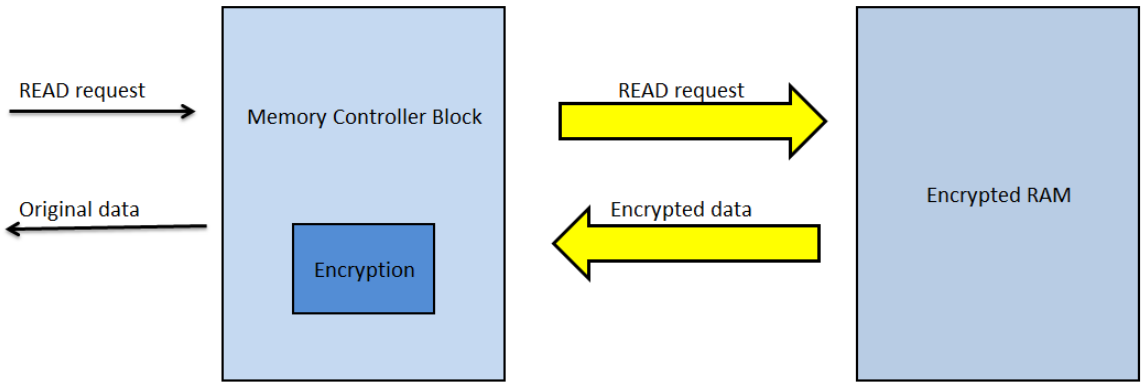

Figure 3.3: Read Request

The storage of key is a crucial point which will be discussed in the next chapter. The proposed memory protection scheme can be used to ensure both integrity and confidentiality for RAM from cold boot attack. The data present in RAM is in encrypted form. Therefore, the attacker when gains access to RAM and tries to read the contents of RAM by doing cold boot attack, he/she shall fails to gain access to original data . This is due to the reason that he has the access to the keys required for encryption.

It can be said that it is straightforward to extend the solution to situations where each processor maintains a memory controller to access the RAM. Most of the Intel and AMD processor come with an on-die memory controller block. Therefore, this design methodology fits into the existing architecture easily.

3.2.2 Design Model

Furthermore, a simple way to extend the existing architecture to support the proposed methodology is shown in Fig 3.4 and Fig 3.5. Address encryption has been used to implement full disk encryption.

While writing,

- Store address in a register for a write transaction.
- Encrypt the address .
- XOR the Encrypted Address with the corresponding data which is to be written at the address.
- Write the XORed value in the memory.

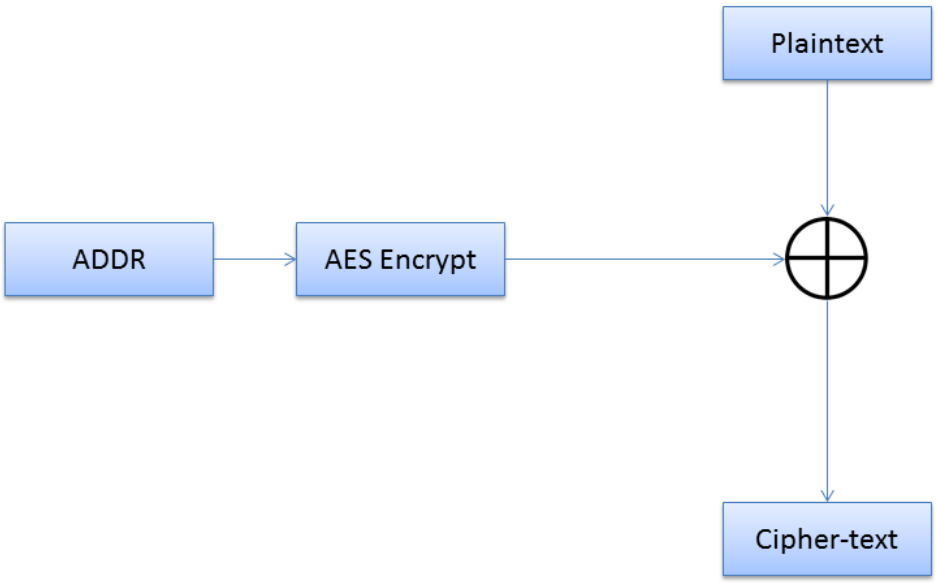

Figure 3.4: Design Model (Write)

While reading,

- Store the address in a register for a read transaction.
- Encrypt this address.
- XOR the Encrypted Address with the corresponding data(Cipher-text) which is read from that address.

- Original read data value is obtained by memory controller and sent to the CPU.

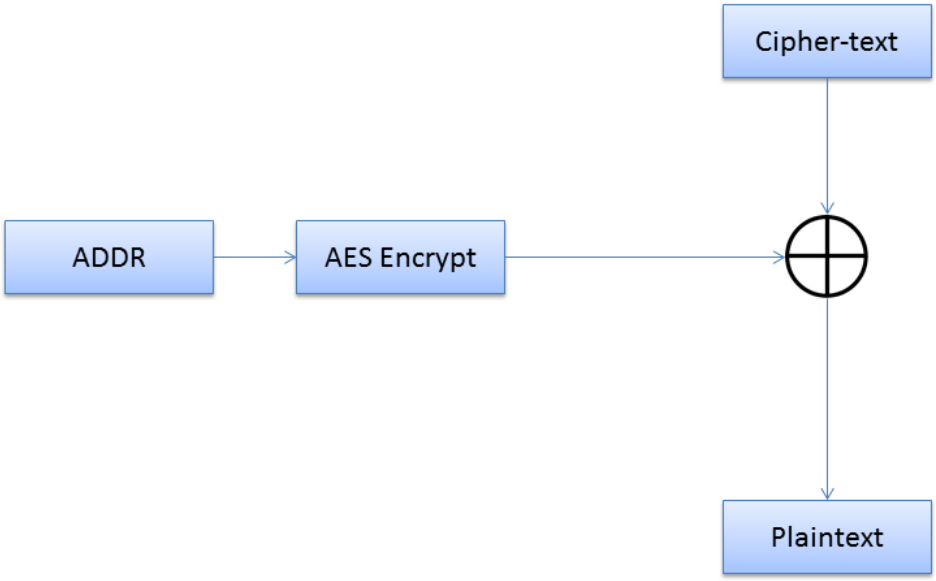

Figure 3.5: Design Model (Read)

3.2.3 Implementation and Results

The thesis introduces a set of architecture innovations that aim for secure implementation of the proposed security model. The real time implementation is therefore a crucial step to check how the design works, challenges and overhead faced by it. Xilinx real time environment has been therefore used to deploy the security model. As discussed in chapter two, Xilinx EDK has been used to design the processor architecture. The Xilinx EDK(Embedded Development Kit) architecture is as shown in figure 3.6. Xilinx SDK(Software Development Kit) has been used to write memory test applications and to test the hardware implementation.

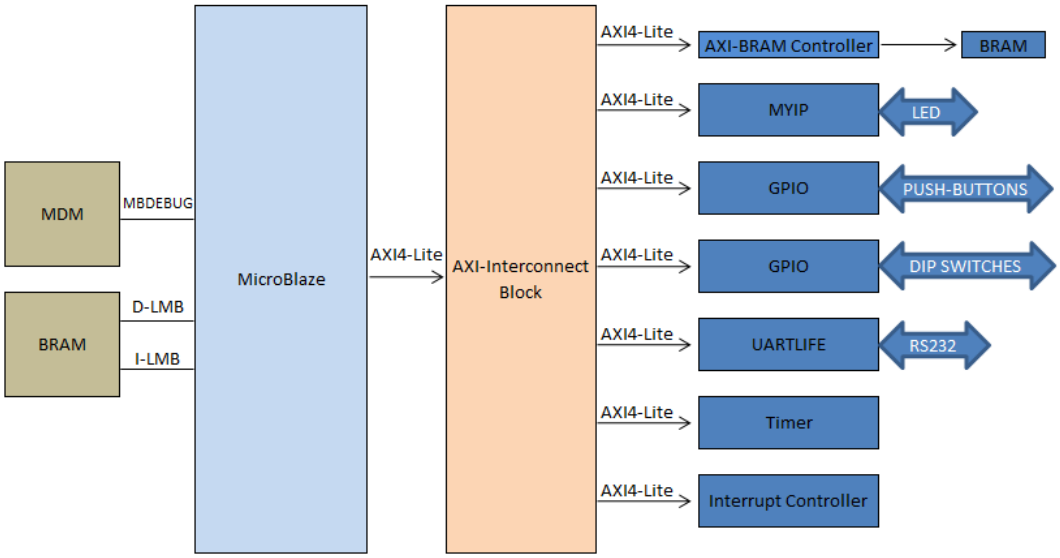

Figure 3.6: Xilinx processor architecture

The Xilinx Memory Controller is a VHDL/Verilog IP block designed to perform memory handling tasks. This has been modified to support memory encryption using AES. The MCB is highly-configurable and may be optimized for various size,throughput, and latency trade-offs.

The Xilinx processor Microblaze communicates with the memory via AXI bus. Now, to implement the security model Xilinx memory controller has to be modified so that it supports encryption. Therefore in this section, the algorithm which has been adopted to make the Xilinx memory controller block, encryption supportive has been presented.

For encryption of data, it is necessary that data on bus be held for the time duration required to perform encryption before being written to the memory. Also, crucial conditions have to be taken care of. Data on AXI bus works in burst mode, therefore the next data should not arrive before the previous data (encrypted) gets written on the memory. Similarly, for decryption also the data must be held on the bus for the time duration during which decryption takes place. Following is the strategy adopted to delay the data on the bus.

Delay in AXI bus

In the usual scenario master(Microblaze) sends the WVALID indicating that valid data is present on the bus and slave(RAM) replies by sending WREADY indicating that it is ready to accept the valid data on the bus. Now we want data to be delayed for the time required for encryption ,the WREADY (sent by slave) signal is delayed by the number of clock cycles through which encryption needs to be done.

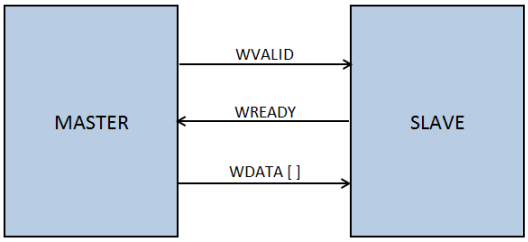

Figure 3.7: Write scenario

Similarly, master(Microblaze) sends the RREADY indicating that it is ready to accept the data present on the bus(read data) and slave(RAM) replies by sending RVALID (valid data is present on the bus to be read). As data is to be delayed for the time required for encryption, the RVALID (sent by slave) signal is delayed by the number of clock cycles through which encryption needs to be done 3.8.

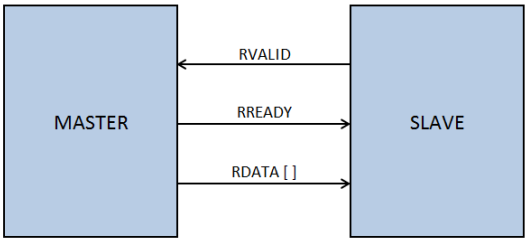

Figure 3.8: Read Scenario

The algorithm adopted to delay the write data for encryption to take place is shown in Figure 3.9. Figure 3.10 and 3.11 and shows the timing diagram for the method implemented. It can be seen, as WVALID goes high, executing flag goes high indicating that valid write data is present on the bus and delay should begin. Encryption starts and therefore post_processing flag also high indicating encryption has begun. Therefore this flag prevents the slave to send the WREADY signal(for write to take place). As encryption takes place, post_processing goes low making executing low. Therefore as executing flag goes low, WREADY goes high and data write takes place.

In the read situation, RREADY and RVALID behave opposite to that in write case. As it can be shown in the figure 3.11 as RREADY goes high, RVALID is delayed till encryption takes place.

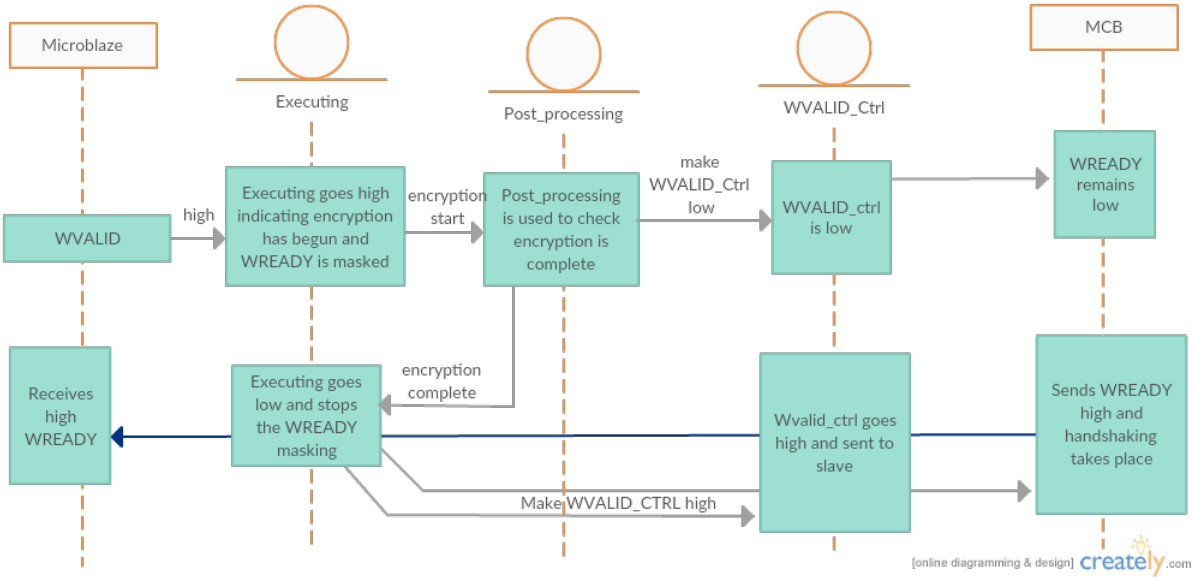

Figure 3.9: Sequence Diagram showing algorithm adopted

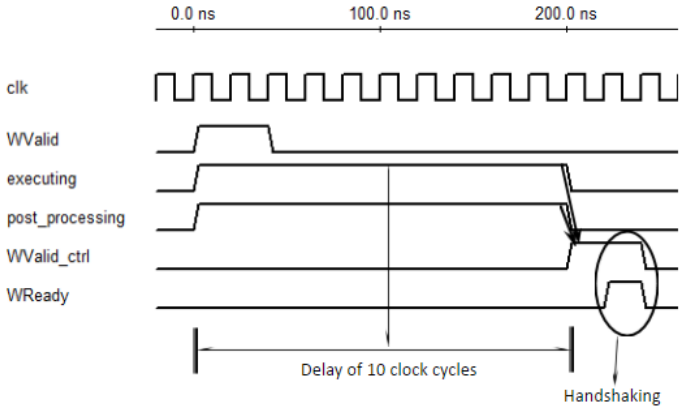

Figure 3.10: Delay in Write Channel

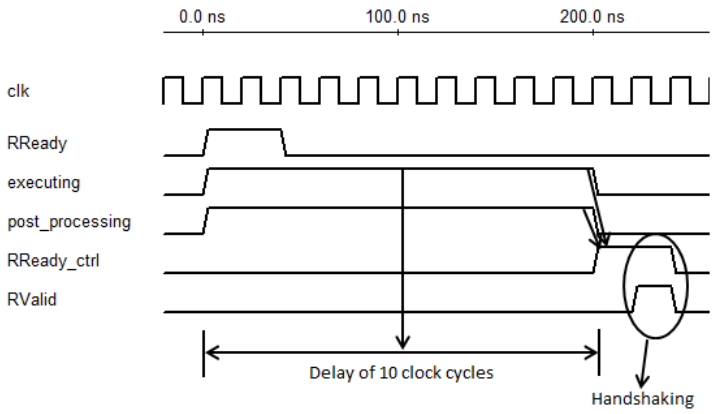

Figure 3.11: Delay in Read Channel

3.2.4 Project on ISE

First step of implementation included simulating the RAM model via a simulator to get an understanding of the internal signals. The following steps were followed to simulate the RAM design in ISIM simulator:

- Generate a testbench to simulate the RAM simulation model.
- DDR interface has to be generated to make connections between testbench and the RAM simulation model.

Figure 3.12, 3.13, 3.14 show the timing for read and write for the RAM simulation model.

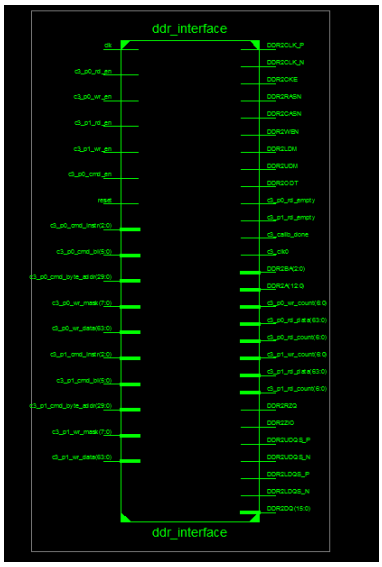

Figure 3.12: DDR Interface

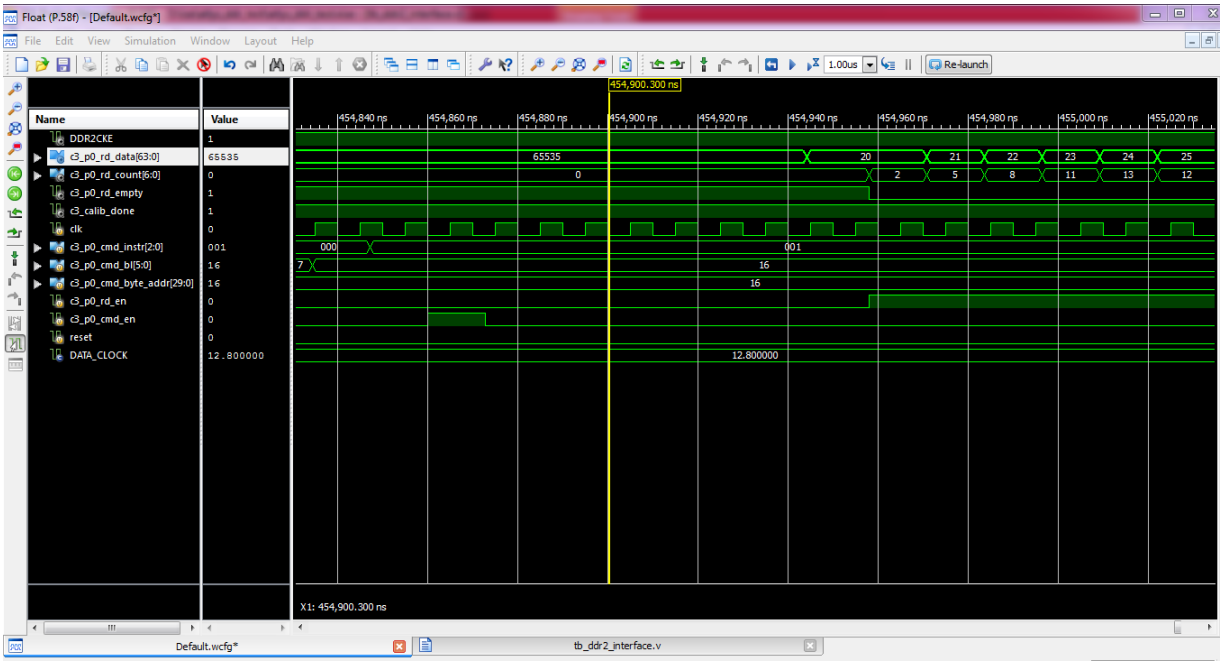

Figure 3.13: MCB Read path timing

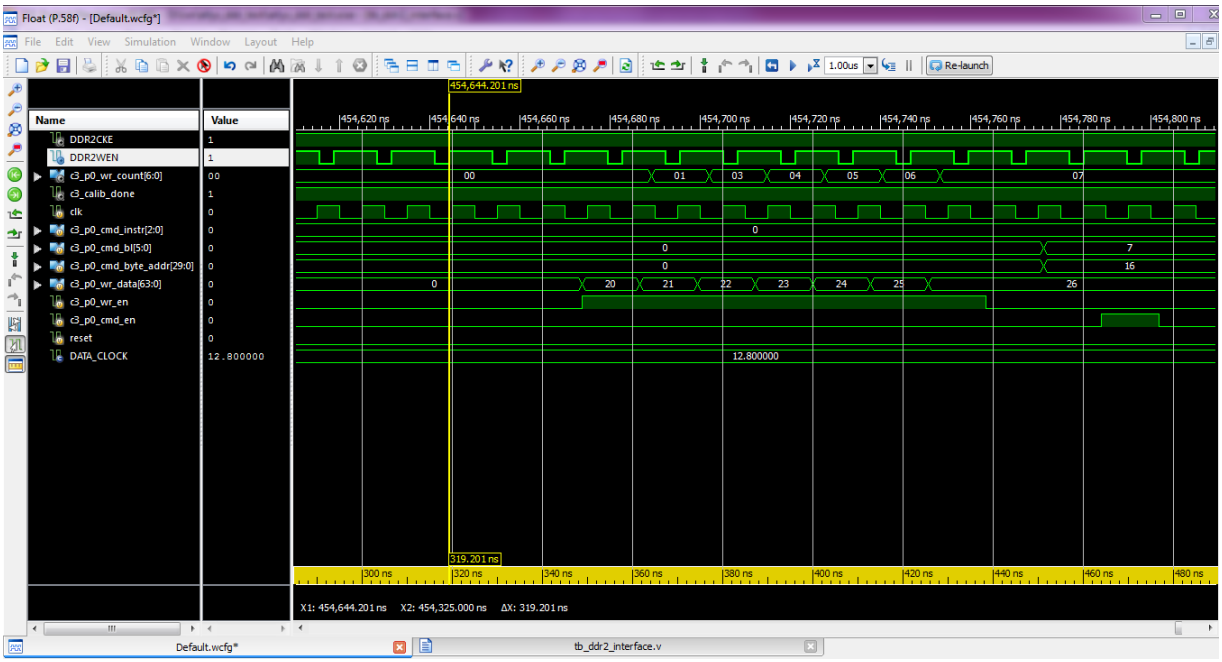

Figure 3.14: MCB Write path timing

3.2.5 Testing of AES module

The AES module used for encryption in the AXI bus has been tested using the ISE simulator. Fig 3.15 shows that it requires 10 clock cycles for the key schedule to take place. Key schedule takes place once at the beginning of encryption. Afterwards, for encryption of each data 10 clock cycles are required.

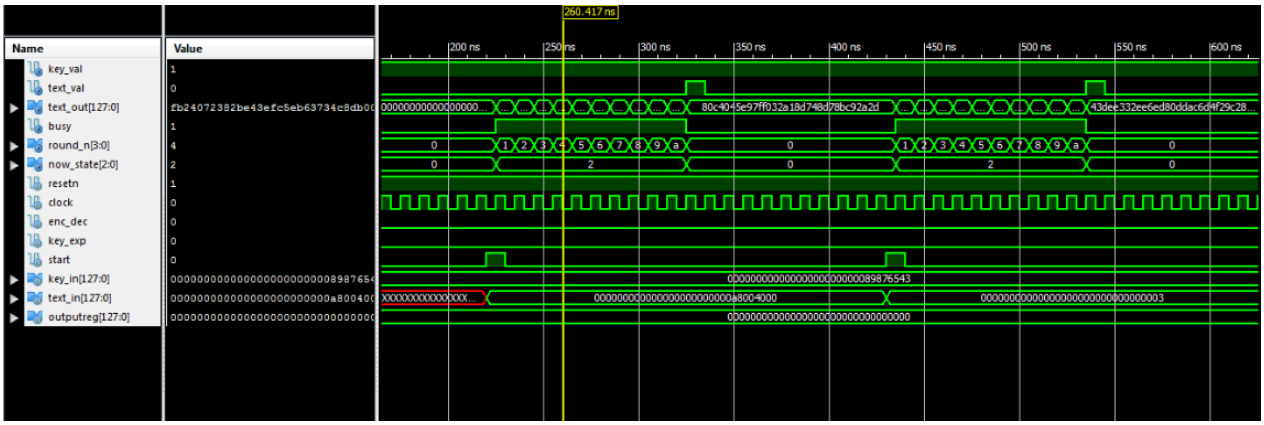

Figure 3.15: AES encryption timing diagram

3.2.6 Modifications in the existing Xilinx architecture

This section briefly describes the Verilog files which have been modified to adapt the proposed design.

Some of the important Verilog files description is given below:

axi_mcb.v –This module serves as a bridge to convert the AXI transactions to those compatible with MCB user interface.

axi_mcb_ar_channel.v – This is the AXI channel through which the address read commands are pipelined and understood by the hardware.

axi_mcb_axic_register_slice.v – It is a single-channel AXI pipeline register.

axi_mcb_aw_channel.v – This is the channel through which address write commands are pipelined and understood by the hardware.

axi_mcb_b_channel.v – This module is responsible for returning the write response to the master that initiated the write.

axi_mcb_cmd_translator.v – This module translates the AXI transaction to the MCB transaction.

axi_s6_ddrx.v – This module instantiates the AXI bridges.

axi_mcb_r_channel.v– This module is used to confirm handshaking so that read takes place

axi_mcb_w_channel.v– This module is used to confirm handshaking so that write takes place.

3.3 Analysis

Table 3.1 shows the performance analysis results for proposed security model. The overhead for a single read and write is 10 clock cycles each. Therefore, table shows the overhead that occurs for different existing processors adapting to the proposed design and their memory latency cycles [8].

Table 3.1: Performance Analysis

Processor	**Memory Latency(clock cycles)**	**Overhead(%)**
Intel Core i7-4770	230	6.89
Intel Xeon E7-4800	180	11.11
AMD Phenom 2	170	11.7

Chapter 4

Conclusion and Future Work

4.1 Conclusion

Mitigation of Cold Boot Attack by providing encryption support in the memory controller is a promising technique. The proposed memory security model protects digital content and software stored in untrusted system memory from physical tamper. The thesis introduces a set of architecture innovations that aim for secure and high performance implementation of the proposed security model and secure processor architecture. These include,

- General approach which can be adopted by processors nowadays to prevent cold boot attack.
- Encryption support in Xilinx Memory Controller Block.
- Address based encryption scheme to prevent data hampering in RAM.

Also, as analysed in section 3.3, there is overhead of 20 clock cycle in a read-write transaction.But, when security is a crucial concern such a minimal overhead can be neglected.

4.2 Future work

This section describes the modifications that must be made in the proposed in the design for successfully storing the key. The design has been shown in figure 4.1. In the proposed design, the MCB has been modified such that it prevents read-write at certain specified part of memory. At this certain part of memory, key can be stored. Some part of memory can therefore be made non-writable. This part of RAM can be used to store the keys for encryption. Figure 4.1 shows the way to implement the design.

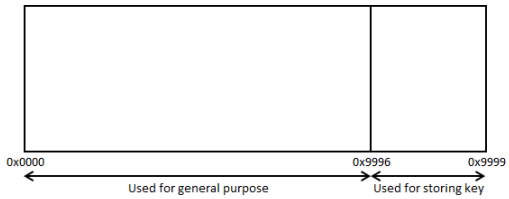

Figure 4.1: Key Storage

Also, kernel modifications can be made to encrypt the information only for specified vulnerable applications. Thee are the applications which have sensitive information stored in them. For other applications data in RAM may or may not be in encrypted form. Therefore, memory controller block can identify the instructions from specified applications and therefore take a decision to encrypt the information being stored in RAM or not.

Bibliography

[1] M. J. Henson, "Attack mitigation through memory encryption a thesis submitted to the faculty," Ph.D. dissertation, Dartmouth College Hanover, New Hampshire, 2014.

[2] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten, "Lest we remember: cold-boot attacks on encryption keys," *Communications of the ACM*, vol. 52, no. 5, pp. 91–98, 2009.

[3] M. Henson and S. Taylor, "Memory encryption: A survey of existing techniques," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 53, 2014.

[4] P. A. Peterson, "Cryptkeeper: Improving security with encrypted ram," in *Technologies for Homeland Security (HST), 2010 IEEE International Conference on.* IEEE, 2010, pp. 120–126.

[5] L. Dorrendorf, "Protecting drive encryption systems against memory attacks." *IACR Cryptology ePrint Archive*, vol. 2011, p. 221, 2011.

[6] A. AMBA, "Protocol specification," *ARM, June*, 2003.

[7] http://www.xilinx.com/support/documentation/user$_g$*uides*/*ug*388.*pdf*.

[8] C. Carvalho, "The gap between processor and memory speeds," in *Proc. of IEEE International Conference on Control and Automation*, 2002.