Design of Efficient Secure Data Communication Techniques for Reconfigurable Hardware Platform

Suman Sau

A.K.Choudhury School of Information Technology

University of Calcutta

A thesis submitted for the degree of

Doctor of Philosophy (Technology)

date

I would like to dedicate this thesis to my loving parents ...

Acknowledgements

I would like to acknowledge my supervisor, Professor Amlan Chakrabarti, for his guidance and mentorship during the course of this research work. I would also like to thank Dr Subhasis Chattopadhyay, Scientific Officer (H+), Variable Energy Cyclotron Centre, Department of Atomic Energy, Govt of India, Kolkata for his support in helping me to carry out my Ph.D study in Compressed Baryonic Matter-Facility for Antiproton and Ion Research (CBM-FAIR) India group. I sincerely thank all my colleagues in the research lab of A.K. Choudhury School of IT, University of Calcutta and CBM India group.

I would like to express my sincere gratitude towards my mother Niyati Sau, my father Haripada Sau, my brother Sibnarayan Sau and my sisters Nila Sau, Sila Sau and Mala Sau. Their cooperation and encouragement helped me in the completion of this project.

I would like to express my special gratitude and thanks to all the other faculties and staff of A.K. Choudhury School of IT.

Thanks to Sangita Ghosh for her motivation and support during this journey of mine. Finally, thanks to all my friends, mentioned or not mentioned here, for always being there with me whenever asked for.

Abstract

We are in the era of emerging and inclusive technologies where data communication plays a pivotal role. The key enablers are high data rate, secured communication and and fault resiliency which can assure both performance and reliability. This dissertation focuses on these key performance parameters for design of efficient data communication techniques with a special motivation towards embedded applications. The design strategies proposed in this thesis though emphasizes on high speed data communication applications, but can also be adopted for general purpose embedded applications through proper customization.

State of the art methodologies for designing efficient data communication systems in embedded platform includes explicit software, explicit hardware and software-hardware co-design approaches. In this dissertation, we have emphasized upon all these approaches and have proposed various design and implementation techniques which can suitably meet the requirements of high speed data communication, data security and error detection and correction policies. Field Programmable Gate Array (FPGA) is a popular choice for implementation of present day embedded systems, as it offers a shorter turnaround time and also provides the scope of hardware reconfigurability (full and partial) along with spatial parallelism in terms of logic operations. We have chosen FPGA as the implementation platform for prototyping the various systems and subsystems designed in the course of this research work. We have successfully designed systems based on embedded processor cores, dedicated hardware logic cores and software hardware co-design involving processor with dedicated

hardware logic cores to meet the demands of the various application scenarios briefed in this dissertation.

This dissertation proposes a hardware optimization for some of the important cryptographic functions like exponential modulus of a large key, parallelization of the advanced encryption system (AES) function for speedup and also enhancing the AES hardware core through the incorporation of error detection and correction schemes. In order to validate our design, we have designed and implemented various test setups involving communication links like RS232, Ethernet and optical which can estimate the performance of the designed sub system in terms of real time parameters, security and error resiliency. Apart from developing the suitable hardware infrastructure for efficient data communication applications, in this dissertation we have also proposed a first of its kind dedicated hardware logic core for the implementation of gigabit transceiver (GBT) protocol for inter system communication catering the need of high data rate applications. A new scheme of error handling for FPGA configuration memory has been designed, which exploits Bose, Chaudhuri, Hocquenghem (BCH), modified matrix code (MMC) etc. coding techniques and also utilizes partial reconfiguration methodology for correcting the affected logic during the run time. The novelty of this technique is well justified as it increased the performance of the system in terms latency, speed up and error resiliency. In a nutshell, this dissertation contributes to the sate of the art research in the domain of embedded systems through the various new methodologies and their system level implementations that have been achieved in the due course of this research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the present era of high speed data communication, higher throughput and lower delay are the key drivers for efficient electronic system design. Successful operation of different sensitive embedded applications like Satellite link communication, Bio Medical device to device communication, High Energy Physics (HEP) read out chain, Internet of things (IoT) sensor etc. require efficient inter/intra system data communication methodologies. Due to increasing communication data rate, data is being moved within systems at much higher speed, which creates challenge in the design integration and implementation of the various sub-systems. Experiments like Compressed Baryonic Matter-Facility for Antiproton and Ion Research (CBM-FAIR), European Organization for Nuclear Research (CERN) need to capture and store data at a rate of over 1Tb/s. The success of the embedded applications developed for CBM-FAIR, CERN etc. depends on the communication speed between the front-end receiver sub-systems and the data aggregation unit and also the communicated data among data processing units and back end storage systems. So, an effective high speed communication protocol mechanism is very much needed for these type of embedded applications, otherwise the overall performance may be decreased. In general, the design of embedded systems for present day communication applications essentially needs to have various performance attributes such as high throughput communication protocols, error resiliency, security etc. Next, we give a brief overview of these performance attributes as below:

- **High speed communication Protocol design**: Several existing methods of communication mechanisms like RS232, Ethernet etc. can be used for inter/intra communications. RS232 is the one of the oldest methods in the era of digital communication through a serial link using RS232 standard. A RS232 standard Data Circuit-Terminal Equipment (DCE) and (Data Terminal Equipment (DTE) port transmit and receive wire connection is shown in Figure 1.1, where TX, RX,GND stands for transmit, receive and ground pin. This standard specifies signal timing, signal functions, signal voltages, a protocol for information exchange and mechanical connections. It has a maximum throughput of 120Kbps.

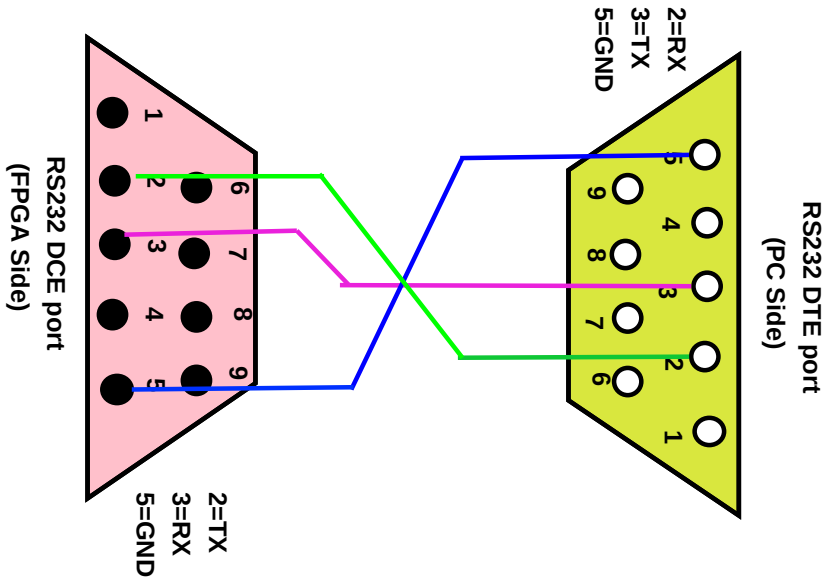

Figure 1.1: RS232 DCE and DTE connection port

On the other hand, Ethernet is a globally used standard protocol for data communications. But there are several disadvantages of using Ethernet in high speed communication process. Firstly, it uses predefined protocol like TCP/IP or UDP/IP. They are convenient but having several layers and all of them must be implemented and traversed for each data transmit/receive. Secondly, the hardware implementation of the protocol stack is costly and they are generally implemented in software layers. Lastly, overhead with respect to data packet traverse time, traversing the protocol stack also re-

sults in considerable overhead in the packet size, as a header is added at each protocol layer. Thus to support high data rate in present day communication systems there is a need of high speed data communication protocol design (CBM-FAIR [4]), radio telescope read out chain where data throughput may be higher than 1Tb/s. In order to achieve such high speed data communication, we need to design a suitable protocol for link management, flow control, synchronization, error control etc. and their efficient embedded implementation.

- **Error control coding and correction**: Hardware systems designed for data communication applications like satellite communication, HEP applications, IoT etc. are often affected by natural and experimental radiation and channel noises, which may lead to errors in the system's outcome. Errors can occur at (i) communication link (ii) internal processing blocks and (iii) memory. Here, we are considering the errors in the memory that can be two types soft and hard errors [5]. Faults created due to radiation effects can be categorized into (i) temporary and (ii) permanent. Temporary faults lead to temporary malfunctions that occur in solid state devices and their effect is manifested as soft error. Soft errors are not reproducible and sometimes lead to Single Event Upset (SEU) and multiple bit upsets (MBU) in different embedded devices. Permanent faults cause breakdown or punch-through in the solid state devices that creates non-recoverable errors in the systems termed as hard error. Effect of soft errors in any logic will be transferred to the output of flip-flop or memory if its period of occurrence is higher than the period of the clock which drives the circuit. On the other hand hard errors are of two types: one is permanent recoverable errors and other one is permanent nonrecoverable errors. When charge particles permanently damage the logic or switching blocks within the systems, permanent nonrecoverable errors occur and it can only be sorted out by replacing the defective logic blocks physically (normally it is done by routing). On the other hand when SEUs or MBUs directly affect the configuration memory of embedded system, permanent recoverable errors may occur which can be corrected before they affect other logic blocks either by

rewriting the content of memory and logic cell or by built-in error detection and correction code. This issues justify the need of different error correction and detection algorithms (Reed Solomon codes (RS) , BCH encoding, modified matrix code (MMC) etc) to be incorporated within the high speed communication protocol to minimize the errors causing SEUs and MBUs in the system. Designing efficient error correction scheme in the hardware should also take into account the trade off between error coverage and hardware resource utilized. Attaining higher code coverage with low resource requirement is one of the key research issues in error correction mechanism.

- **Secure communication:** For inter/intra system communications, there is a high probability of tampering of the original data sequence by intentionally/unintentionally by unknown agents, which is a threat to security and authorized access of the system. In figure 1.2 a basic communication model is shown where 'A' and 'B' two nodes, communicating over an unsecured channel, and a attacker is trying to eavesdrop their communication. Data security can be implemented using different cryptographic algorithms on the channel data where instead of sending the original data over the unsecured channel, encryption and decryption is used. Encryption is the process of encoding messages or information with some randomly generated keys in such a way that only authorized persons who knows the exact key, can read (decrypt) it. In general, encryption techniques can be broadly classified as symmetric and asymmetric key encryption. In symmetric key encryption (advanced encryption standard (AES) [6], Rivest Cipher 4 (RC4) [7] etc.) the same key is used for both encryption and decryption process whereas in asymmetric key encryption (Rivest-Shamir Adleman (RSA) [8], Diffie-Hellman key exchange [9], Elliptic curve cryptography (ECC) [10] etc.) different set of key is used for encryption and decryption process. Due to the inherent algorithmic computational complexity which occurs due to mathematical functions like exponential and mod operation in case of RSA, rounds operations depending on the key size for AES encryption/decryption etc. An efficient design and implementation of these algorithms needs to be performed in the design phase to enable secured communication. Authen-

tication is the process by which the access signature of an user is verified for authorized access of the system. Authentication algorithms like Secure Hash Algorithm (SHA) [11], MD5 [12] etc. needs to be implemented to prevent unauthorized access of the system. So, the present generations of data communication system should be able to handle security issues like encryption and authentications in a resource optimized way.

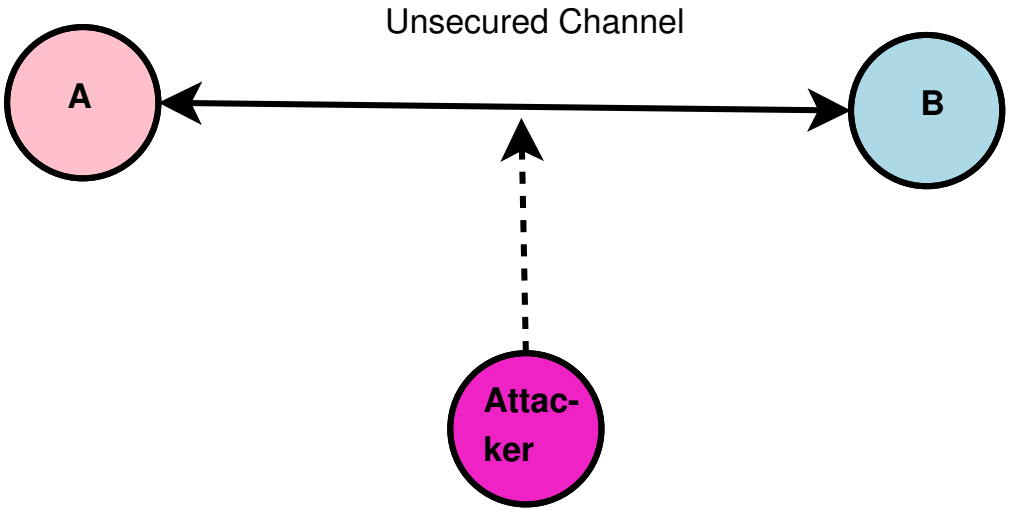

Figure 1.2: Basic communication model

As a choice of target platform, the embedded designers have three choices namely, (i) standard micro-controller/microprocessor platform (ii) custom hardware platform based on Application-Specific Integrated Circuits (ASIC) and (iii) custom as well as re-configurable hardware platform based on Field Programmable Gate Arrays (FPGA) . Basic design architectures and design flow of these three platforms are described in the Section 2.2.1, Section 2.2.2 and Section 2.2.3 respectively.

1.1 Motivation and scope of the work

A very famous quote by Cilardo et al. [13] who stated, "A mathematical construction with practical applications, such as a cryptographic algorithm, has no real interest, in an engineering sense, as long as methods

for feasible implementation are not available." In other words, efficient implementation techniques for cryptographic algorithms, secure communication protocol design, error correction and detection model in hardware level are necessary in order to use them in practice.

1.1.1 Need of efficient hardware design for secure data communication

Applications those are using cryptographic algorithms generally fixed with very fastened requirements for implementations. Low-cost applications, such as mobile handset devices, sensor networks, or smart cards, to name a few, often set tight constraints on available resources, such as logic, memory, or power . On the other hand, high-speed applications, such as data acquisition system used in satellite/ some real experiments bio medical application([14] [15]CBM-FAIR [4], CERN [16] etc.), high-speed network servers require very high computation speeds in order to prevent cryptography from becoming the bottleneck [17], [18]. Therefore, design and implementation of fulfilling high speed communications protocol with security features (using cryptographic algorithm) requirements is often a challenging task and efficient implementation of cryptographic algorithms in hardware is a challenging research motivation.

Normally, general purpose microprocessors are used for implementing cryptographic algorithms but, in that case, the execution of computationally expensive secure cryptographic algorithms has bounded by the the limiting factor for overall performance'[19]. Hence, dedicated hardware implementations are commonly required to cryptographic algorithmic implementation for better performance and security. Re programmable hardware such as FPGA has established itself as one of the most attractive platforms for cryptographic algorithms because of the combination of speed and flexibility [20]. This thesis concentrates on FPGA which are commonly considered very feasible implementation platforms among implementors of cryptographic algorithms [20].

The need of FPGA based real-time data communication is of special interest where FPGA based data acquisition systems are built to capture, store process and transmit the data generated from real time experiments [21] [22]. Also FPGA based system can work as an embedded system for network devices and communication gadgets, which requires security for real time data communication. FPGA vendors are providing hard and soft processor cores for this purpose, which can be effectively used for running the crypto algorithms on FPGA devices which is describe in section 1.1.2

1.1.2 Cryptographic Algorithms in FPGAs

In order to accelerate the execution of complex mathematical tasks that are commonly performed in cryptographic processes, direct hardware execution of these tasks is a good choice. For an example, over one half of the processing time in Secure Sockets Layer (SSL) application is consumed in executing cryptographic algorithms [23]. A good plan is to have the most expensive operations of a cryptographic algorithm to be implemented as a custom logic block in the FPGA to have an accelerated performance, while the other parts of the application is implemented through software execution, performed by the general purpose processor like Microblaze, ARM cortex etc. Consider RSA algorithm as an example where the most expensive task is the modular exponentiation and hence, RSA can be accelerated by implementing exponentiations as a custom logic in the hardware while the main processor performs rest of the operations concurrently. The classification of reconfigurable systems are shown in figures mentioned below.

The external stand-alone processing unit (a) [24] shown in Figure 1.3 is best for applications which are computationally intensive but require little communication and the reconfigurable functional unit, e.g., reconfigurable instructions etc.

The attached co-processor classes (b) and (c) can be seen as a trade off between the fast dedicated bus [25] [8]. Figure 1.4 and Figure 1.5 describes co-processor classes (b) and (c) respectively.

The last class (d), where the processor is embedded in reconfigurable logic, is currently emerging research areas [8]. Figure 1.6 shows the block representation. This class of design implementation requires a device having a large number of

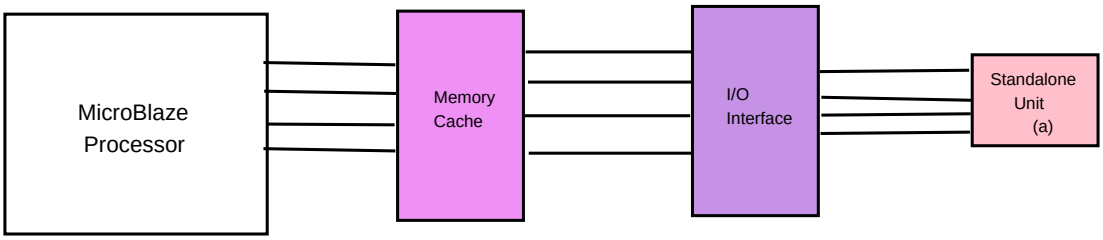

Figure 1.3: External standalone processing unit (a)

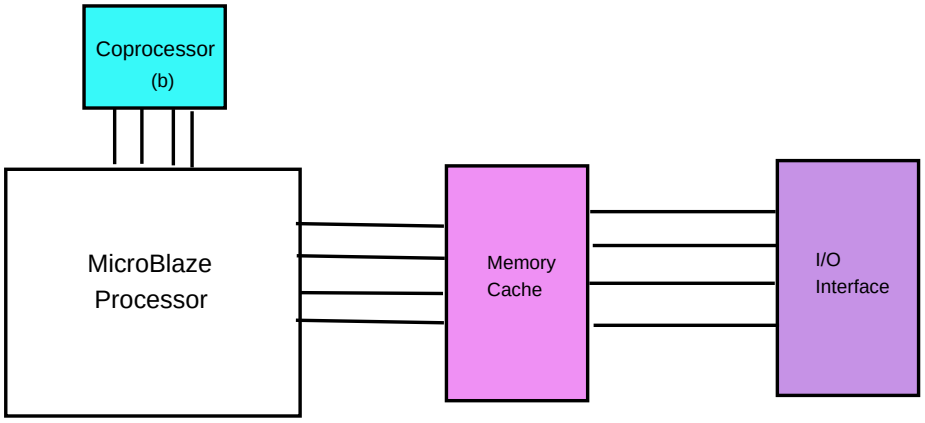

Figure 1.4: Co-processor (b) unit with faster bus

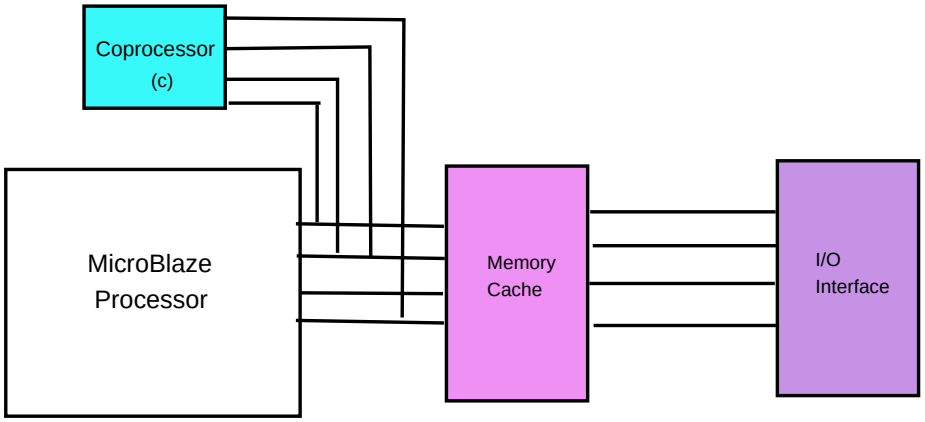

Figure 1.5: Co-processor (c) unit with slower processor bus

configurable blocks and thus will increase the system cost.

The embedded processors that are available in FPGAs can be either hard core such as PowerPC in Xilinx Virtex-4 FX , or soft core such as Microblaze in Xilinx FPGAs or Nios II in Altera FPGAs. All classes explained above, can be

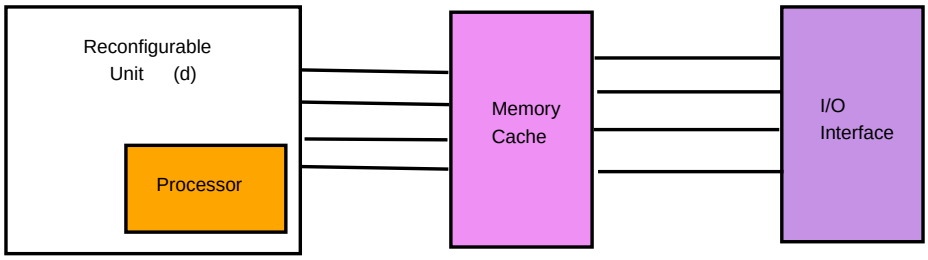

Figure 1.6: Processor embedded in reconfigurable logic (d)

implemented with a stand-alone FPGA.

So to have the best design we have to find a correlation between these hardware and software design metrics that will be suitable for embedded secure communication, which seems to be a complex job.

1.1.3 Need of detection and correction of errors in high speed secure data communication beyond 1Tb/s

High speed device to device data communication is needed in various applications like satellite communication, HEP experiments and Bio-medical analysis and visualization (involving audio/image/video data) for combined transmission of data, trigger, timing, fast and slow control and monitoring. In HEP experiments effect of radiation is pre-dominant in the detector side where radiation hardened electronics systems are used having higher cost and lower flexibility for upgradation. Whereas, the effect of radiation is much lower as we move outside the detector cave where the data acquisition (DAQ) electronics exist hence we can use commercial-off-the-self (COTS) FPGAs having efficient error handling techniques thus lowering the overall system cost. The success of these type of applications depend on the throughput, security and error handling capability of the entire DAQ chain. We need to built an error detection and correction mechanism to handle both SEU and MBU which can occur due to radiation in the environment. Forward error correction (FEC) has been considered as a strong and cost-effective way to improve the quality of transmission (QoT) that is used here for correction of errors happened due to SEU and MBU . The major research problems that have been considered in this phase of our research are given below:

- Enabling security in the embedded data communication process through encryption/decryption
- Design and implementation of an FPGA based system that can handle high speed communication (1 Tb/s) in embedded communication domain.
- Design and implementation of FEC as a hardware module for error correction capabilities for 1Tb/s optical fiber links.
- Handling configuration memory error in FPGA targets through MMC using partial reconfiguration techniques

1.2 Organization of the Thesis

In this thesis, our contributions target some specific design and implementation issues to overcome some of the challenges existing in the present state-of-the-art embedded data communication systems. Related research work is described in Chapter 2. Crypto algorithms and its embedded processor based implementation along with a complete test setup for serial RS232 communication infrastructure is briefed in Chapter 3. The design of crypto co-processor and its implementation is described in Chapter 4. Chapter 5 and Chapter 6 described high speed secure communication system design using Bose, Chaudhuri, Hocquenghem (BCH) error detection and correction model and multi-bit error detection and correction model for crypto co-processor based architecture respectively. Chapter 7 briefs soft error mitigation technique caused by SEU or MBU in FPGA configuration bit memory using modified matrix code. The pictorial presentation of the dissertation is shown in Figure 1.7

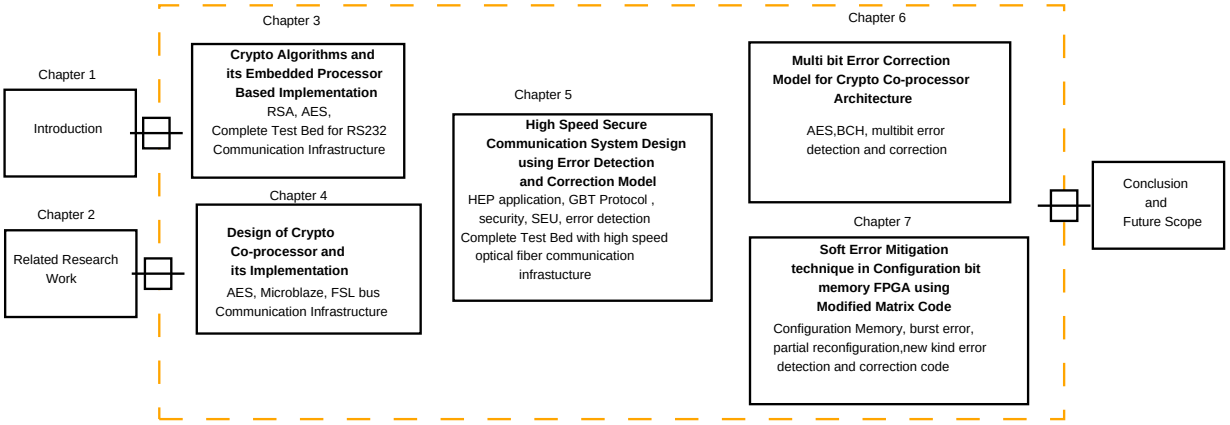

Figure 1.7: Chapter Organization

Chapter 2

Related Research Works

2.1 Introduction

In the field of real time communications applications like radar communication [26], satellite communications networks, provide a global reach and wide area coverage to remote, rural and inaccessible regions in the earth. A few of its common applications include weather prediction, telephony services, telemedicine, multimedia services, internet connectivity, navigation through GPS, imaging through remote sensing satellites for resource monitoring and many important military applications. However security over these communication link is still a major concern in satellite/radar communications. On the other hand, applications like HEP applications [16] [4], bio-medical application [15] etc, also need a high speed, low latency and fault resilient communication channels for data sharing among different nodes of their network. In an other applications like smart grid [27], a next generations power control systems, there also a high speed communications network is needed among the different power grid for the data exchange. However with the increasing of the cyber attack in every critical applications which are mentioned here, the integration of high speed, reliable

, error resilient and secure data communication networks is needed to manage the complex applications like radar communication [26], HEP applications [16] [4], bio-medical application [15] and smart grid [27] etc. with more effectively and intelligently

2.2 Basics of available design platform

2.2.1 Microprocessor and Micro-controller

Microprocessor based embedded system design relies on the on board microprocessor chip and the other component chips like memory and I/O controllers for successful implementation of the system. The functionality of the system is implemented through software instructions which are executed by the host processor utilizing the on board resources. The microprocessor based systems offer a generic infrastructure and not a custom infrastructure suited for the specific embedded application and hence cannot guarantee optimized performances. Some of the well known microprocessor platforms for embedded applications are 8008, 8085, 8086 etc.

Micro-controller based embedded systems are the most common choice for the traditional embedded programmers. Micro-controllers are single chip microcomputers, which include processing, memory and I/O elements in the same chip. They are normally used for specific application as they have limited resource and are driven by software instructions. Their performance is limited by the fact that they only provide generic processing core, which cannot satisfy the need of higher performance as compared to custom application specific cores. Some of the well known micro-controller platforms for embedded applications are 8051, ARM etc.

2.2.2 Application Specific Integrated Circuits (ASICs)

ASICs are integrated circuits designed for specific application [28]. Unlike the general-purpose circuits, such as microprocessors and micro-controllers, ASICs are designed to perform a specific task or a function. Designs implemented in ASICs cannot be changed once the chip is fabricated or manufactured. There are

many embedded applications used in large experiments like CBM-FAIR, CERN etc. where upgradation of embedded hardware happens in repeated interval. In such cases to minimize the downtime of a system as well as to support the upgradation of the embedded hardware, FPGA is the only solution. Moreover, with the inclusion of partial programming capabilities in modern FPGAs, the downtime of the system is further reduced as the hardware upgradation in the partial reconfiguration region can work concurrently while, the other regions are on execution. The circuit implementation can be highly optimized in terms of speed, area and power consumption, but its inability to adapt design changes after manufacturing is a serious disadvantage. Modern ASIC fabrication processes are also costly and time consuming and hence, they are used predominantly in large volume different embedded gadgets, such as mobile phones, TV, micro oven medical devices etc..

2.2.3 Field Programmable Gate Arrays (FPGAs)

Programs and their implementation are flexible using general-purpose processors, in the sense that programs can be updated as per need by using high-level programming languages without modifying the underlying processor architecture. General purpose processors cannot cater the architectural requirement for a specific algorithm instead their focus is to support the processing need of a large variety of algorithms. Hence, on an average for any given algorithm the speed of execution in a general purpose processor is moderate and that leads to increasing processing time and energy expense. ASICs, on the other hand, offer higher performance and lower power consumption but are less flexible. Thus, there is a significant gap between general purpose processors and ASICs. Reconfigurable logic fills this gap and in an ideal case, combines the best of both; namely, the speed of ASICs to the flexibility of general-purpose processors. Figure 2.1 shows performance-flexibility graph .

Reconfigurable logic includes a wide scope of programmable devices including Programmable Logic Array (PLA), Programmable Array Logic (PAL), Complex Programmable Logic Device (CPLD) , and FPGA. Though there are different programming devices available as said but from hardware cryptography imple-

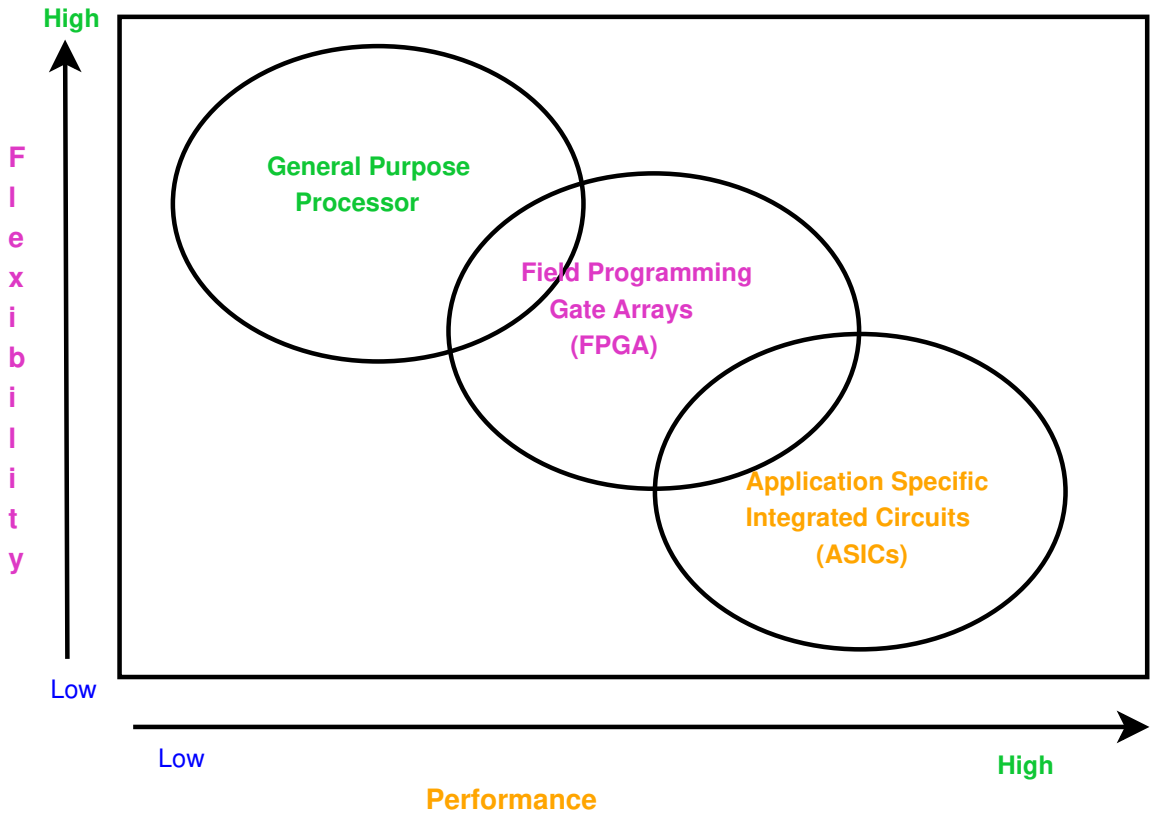

Figure 2.1: Performance-flexibility graph among ASIC, FPGA and general purpose processor

mentation angles, we are interested on FPGAs and henceforth, the terms reconfigurable logic and device always refer to FPGAs in this thesis. In FPGA manufacturing world there are two leading vendors: Altera [29] and Xilinx [30]. Other FPGA manufacturers are Achronix, Actel, Atmel etc. Main units inside the FPGA are reconfigurable functional units, programmable switch matrix and I/O interfaces. For implementing any logic in the hardware, reconfigurable functional units are used and hardware blocks are interconnected by the re-configurable/programmable switch matrix.

Reconfigurability depends on the size of reconfigurable units referred to as granularity. There are two types of architecture used depending upon size of reconfigurable units : i) fine-grained and ii) coarse-grained. In fine-grained architecture, reconfigurable functional units are small whereas in coarse-grained recon-

figurable units are larger. Look up table (LUT) is the basic unit in fine-grained architecture where with a small number of inputs and a single bit output [24]. Whereas in coarse-grained functional unit is an Arithmetic Logic Unit (ALU) [24] and, thus, complex arithmetic operations are fit better in coarse-grained architectures. The granularity of reconfigurable interconnections also varies in different re-programmable architectures so that fine-grained interconnections allow each bit to be connected separately whereas coarse-grained interconnections connect several bits at a time [24]. Present commercial FPGAs are mainly follow fine-grained architectures where reconfigurable functional units are typically 4-to-1-bit LUTs which are arranged in clusters. A collection of LUT and flip-flop in FPGA is called configurable logic block (CLB) . For an example a Xilinx Virtex-5 FPGA CLB contains 6 LUTs and 8 flip-flops. The number of LUTs and flip-flops per CLB varies depending on the different vendor specific FPGA boards. Internal interconnection among the CLBs are done by the switch box interconnection. A generic FPGA architecture with input/output block, CLB, block RAM (BRAM) is illustrated in the Figure 2.2. An example, Xilinx FPGA Virtex-E/II simplified CLB/slice architecture is shown in Figure 2.3. Modern FPGAs are also integrated with hard processor core (PowerPC, ARM etc.) and soft core(MicroBlaze, NIOS etc.) which can be used for any processor based complex embedded design using FPGAs.

Programming an FPGA means, implementation of the hardware in the FPGA logic blocks whereas for general purpose processor programming means providing a set of instructions to be executed by the processor. . To know about the immanence of FPGA and its key features one can look into [31].

FPGAs bring new concepts due to their portability and their ability to re-configure. In a nutshell they can offer the designers a sustained development process much faster and as efficient compared to the ASIC and microprocessor based design [31].

Nowadays FPGAs are widely used in embedded systems for their multiple advantages briefed as follows:

- FPGA can be used as a co-processor rather than a peripheral in a modern CPU using a high speed bus, like Peripheral Component Interconnect Express (PCIe)

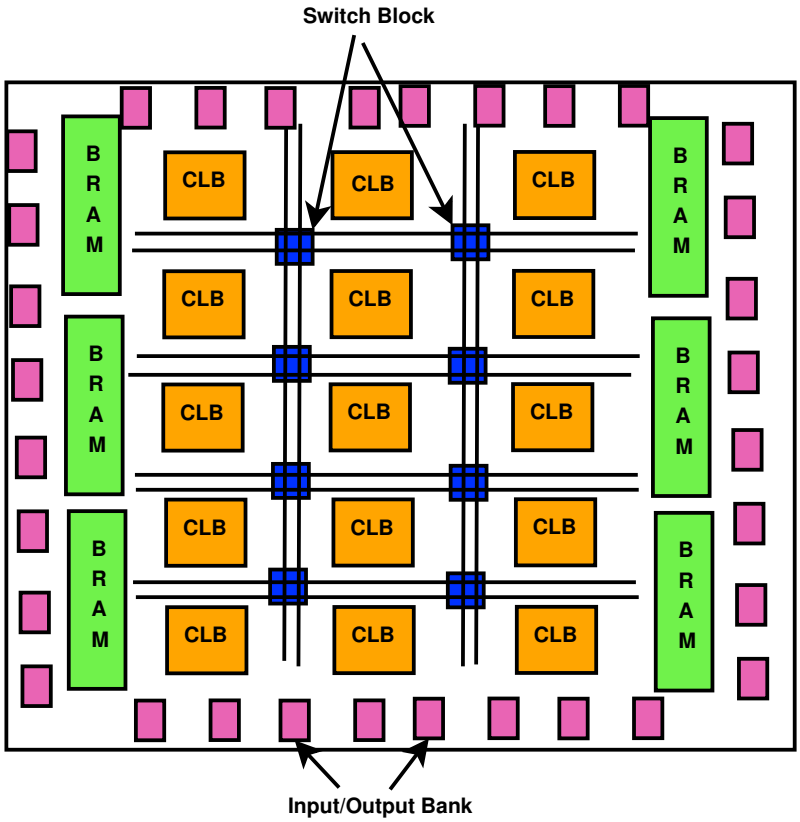

Figure 2.2: A generic FPGA internal architecture

- Memory access inside the FPGA is just one clock cycle typically [32], and with an FPGA we have a far more fine-grained control over the memory data as compared to as the central processing unit (CPU) . Low latency of memory access is one of the promising factors in increasing the FPGA performance as the waiting for the data during computations is less than a CPU
- FPGA supports partial reconfiguration where changing of a portion of reconfigurable hardware circuitry while the other part is still running/operating.
- They reduce time to market by reducing development cycle time
- They permit remote reconfiguration.

FPGAs are commonly used to achieve high performance computation as they offer spatial parallelism involving the configurable blocks.

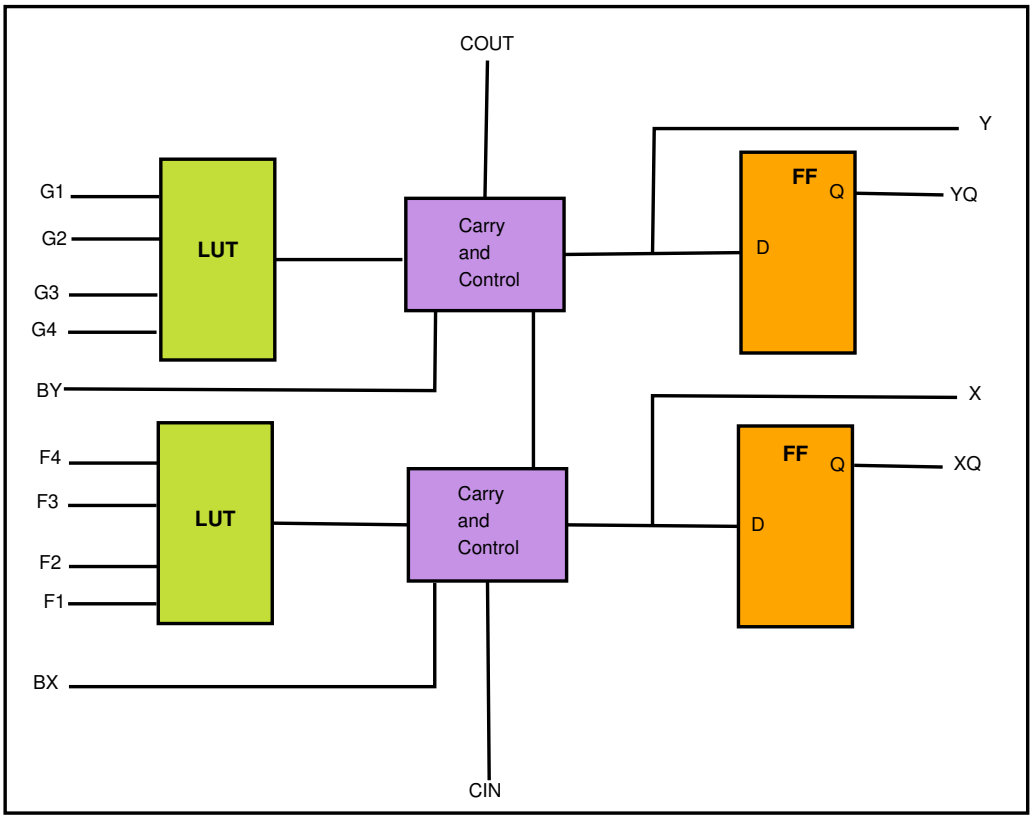

Figure 2.3: Xilinx Virtex-E/II FPGA CLB/slice architecture

2.2.4 FPGA Design Flow

The design flow starts with the design specification and finishes into a working design in an FPGA. Figure 2.4 shows the design flow of FPGA-based implementations. The functionality of the hardware description language (HDL) is verified by functional simulations. After that a gate-level netlist with logic synthesis is mapped from the verified HDL. Using programmable switch matrix place and route process for the functional units are done. The place and route returns a programming bit file and a timing model description corresponding to the logic architecture. The timing model is verified to fulfill the timing requirements set in the specifications with timing simulations. Lastly, using the programming file commonly known as bit file, the FPGA is configured or programmed. Design softwares offered by the FPGA vendors support the complete design flow. Automatic mapping canbe also done by some tools like automatic HDL generators [33].

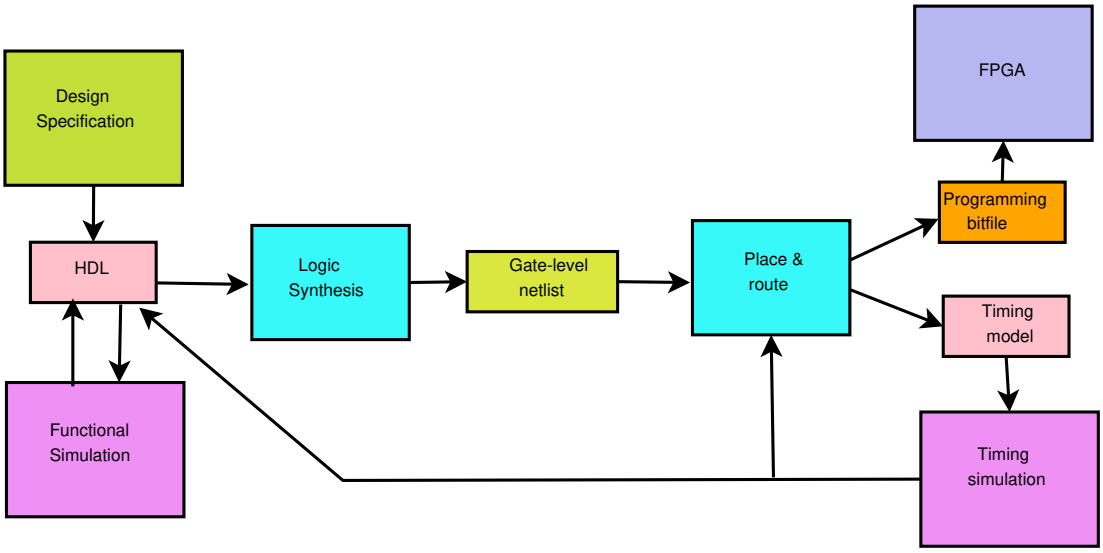

Figure 2.4: HDL based design flow for FPGA based implementation

Embedded systems are particularly vulnerable since physical access to these devices can be done at an ease. Indeed, the owner can be the attacker [34] explains that security has become an important issue in the embedded systems. Also an attacker may try to inject fault intentionally in the secure embedded system, secure communication channel etc. and thus comprising the security. In many of the cases the faults are caused unintentionally due to the natural effects e.g radiation, cosmic ray. Detecting the errors occurring due to these faults and also correcting them in the hardware will make the system more trustworthy. The design and implementation of efficient secure data communication techniques for reconfigurable hardware platform and making the system fault tolerant against the intentional/unintentional faults are the principal aim of this research work.

2.3 Related work

Establishing communication security means integrity and confidentiality in the delivery of information to the receiver. Describe security in terms key features like confidentiality, authentication, integrity, access control, and key management, so that information over the communications remain correct and cannot be interpreted other than the actual

receiver [35]. While dealing with the security in satellite, radar, HEP and smart grid etc. communication, these key points always remain a major concern for the security measures. In general define the security key features as follows:

- Confidentiality means that only the appropriate users have access to the information.
- Authentication requires verification of a user's identity and right to access. It is achieved using a public key interchange protocol that ensures not only authentication but also the establishment of encryption keys.
- Integrity means that the information has not been corrupted. Information can be corrupted by the attacker injecting a error on it.
- Access control means that the system cannot be compromised by unauthorized access (e.g. pirating a satellite, pirating DAQ for HEP etc).
- Key management is the way how to manage the security keys (how dynamically generated, concealed and distributed, finally how well-kept by the actual target) users.

Thus for implementing secure communication framework, different cryptographic algorithms are need to be implemented in the transmitter and receiver sub-systems of a communication intensive application for secure data communication.

2.3.1 Related work in secure communication

Most public-key algorithms involve operations such as modular multiplication and exponentiation, which are much more computationally expensive and time consuming. **Among the existing public key algorithms the RSA is the most widely adopted [36]. Its security lies in the difficulty of factorizing large integers. Several Montgomery designs have been proposed for ASIC and FPGA implementations, based on limited resource availability to satisfy the computational burden [37], [38], [39]. A lot of research work**

exists in the implementation of RSA algorithm on FPGA [40], [41] but there exists a very few research papers on the FPGA based implementation of RSA for real time data communication [42], which essentially involves execution of the security algorithm in coherence with the input data coming from the communication channel in real time. Another public key cryptography namely ECC and its implementation in FPGA can be found in related research works [43], [44], [45], [10]. In [46], a parallelization method utilizing point operation interleaving is proposed in FPGA implementation of ECC. Parallelization of scalable point multiplication that can support all 5 NIST Koblitz curves without reconfiguring structure is proposed in [47]. The above mentioned ECC implementations on FPGAs are standalone hardware implementation. A special care needed in hardware design to make them more faster, resource and power efficient for use in real communication field. Thus the need of FPGA based real-time data communication is of special interest where FPGA based data acquisition systems are built to capture, store process and transmit data generated from real time experiments [21] [22]. Also FPGA based system can work as an embedded system for network devices and communication gadgets, which requires security for real time data communication.

Though public key cryptography solves the problem of secure data exchange but still it has a big disadvantage, mainly speed, in comparison with the symmetric key based cryptography. Symmetric key cryptography is more suitable for the encryption of a large amount of data[3]. In 2000, the National Institute of Standards and Technology (NIST) announced Rijndael as the winner of AES contest, in an effort to address the threatened key size of Data Encryption Standard (DES) [6]. Though, we have a variety of software applications that provides security [48], [49], [50] but in comparison to hardware based implementations they are slow and are more resource hungry. Hence, there is an increasing need of implementation of security algorithms at the hardware level.

Research efforts have been made for the implementation of AES security algorithms on hardware for embedded system applications using FPGA [3], [51]. In [3], the authors have presented a parallel reconfigurable hardware implementation

of the AES cryptographic algorithm, which takes more clock cycles for encryption and decryption as it is based on a single processor core architecture. In the designs[51], [52] the user designed core is added with the Microblaze processor core [1] using the on chip peripheral bus (OPB) , which is a slow bus and hence we need more clocks to send/receive data between the processor core and the user designed core. Microblaze is a Reduced instruction set computing (RISC) [53] soft core processor that is available in the Xilinx embedded devlopment kit [54]. A real time board to board communication with secure AES algorithm is shown in [55], where a software based implementation is done using Altera NIOS II processor core [56]. The other related research works on hardware based crypto-system design can be found in [57], [58], [55], [3], [59], [60], [51], [61]. As per the data provided by Xilinx, communication between a hardware and the Microblaze processor core is fastest for the case of Fast Simplex Link (FSL) bus connectivity [2], and hence FSL bus based design is well suited for high speed applications.

2.3.2 Related work in high speed communication and data acquisition

Real time embedded applications like radar communication, HEP applications, satellite communication, bio-medical application and smart grid applications etc discussed in section 2.1, a high speed, low latency and fault resilient data acquisition DAQ are of utmost need for control of the data communication. In a traditional DAQ system Frontend Electronics (FEE) data is captured from the sensors through a high speed differential Low-voltage differential signaling (LVDS) link [62], which is processed and then transmitted to the back end storage device using high speed links like Ethernet, PCIe , fiber optic etc. for further data analysis. A commonly faced problem in the traditional DAQ systems is adaptation with high data rate [63] and prone to SEU and MBU specially in case of radiation intense environment. Presently optical fiber and PCIe are the most suitable options in high speed data transmission over normal copper based LVDS line as they are less susceptible to noise and interference, bandwidth, range of transmission etc [64]. We are not considering open channel communication like

wireless medium in this context.

In [65] the authors brief a new protocol Gigabit Optical Serial Interface Protocol (GOSIP) for communication over optical fiber and implemented PCIe to optical link interface on FPGA for their DAQ system. This system gives a stable data rate of 1.6 Gbps. In [66], we find the implementation of optical link between two PCIe buses of computing nodes, which achieved the data rate of 8.5 Gbit/s. They have used PCIe hard IP available in ALTERA Stratix IV FPGA board. Liansheng Liu *et.al* presented the development of a fiber channel node with PCIe interface for avionics environments in [67]. Two nodes are connected by optical fiber through Small Form-factor Pluggable (SFP) and maximum data transfer rate achieved in this case is 2.125 Gbps. A high-speed data transmission protocol over optical fiber for real time data acquisition in Beijing Spectrometer III (BESIII) trigger system had been developed by Hao Xu *et.al* in [68]. Here they have used Multi Gigabit Transceiver (MGT) of Virtex-II Pro series FPGA for data transmission over optical fiber and achieved data rate of 1.75 Gbps. In [69], a high speed data transfer protocol named Serial Front Panel Data Port (SFPDP) over optical fiber is implemented on FPGA to capture the data from Digital Signal processor (DSP) through eXtended Attachment Unit Interface (XAUI) where the optical link can work on three distinct speeds: 1.0625 Gbaud, 2.125 Gbaud, and 2.5 Gbaud. In [70], authors have used a bus master, direct memory access (DMA) along with a 4-lane generation 2 PCIe link to transfer the stream data from FPGA to PC. The highest transfer rate achieved in this case is 784 Mbps. **The above mentioned works are only focused on the high speed communication protocol design, they are not considering the security aspects on it. In this thesis we have considered high speed communication with security protocol as well as error resilient techniques in it.**

2.3.3 Related work on High speed communication with error detection/correction

System designed for satellite communication, HEP experiments and others(section 2.1) with FPGA for high speed secure communications are effected by experimen-

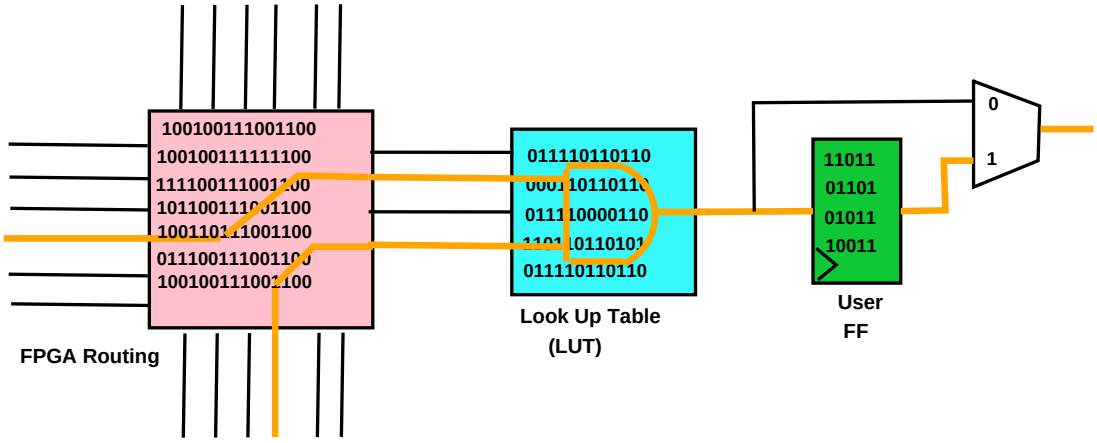

Figure 2.5: User design routing of logic AND in an FPGA configuration memory

tal and natural radiation is a common phenomena [71]. Radiation effects reflected in FPGA's configuration memory. FPGA configuration memory architecture is described here and explain how it effected by faults. In Figure 2.5 configuration memory of FPGA with the series of 1's and 0's represents the function of the routing and logic elements interconnection switch, LUT , and Flip-Flop to implement an user design logical AND gate into FPGA.

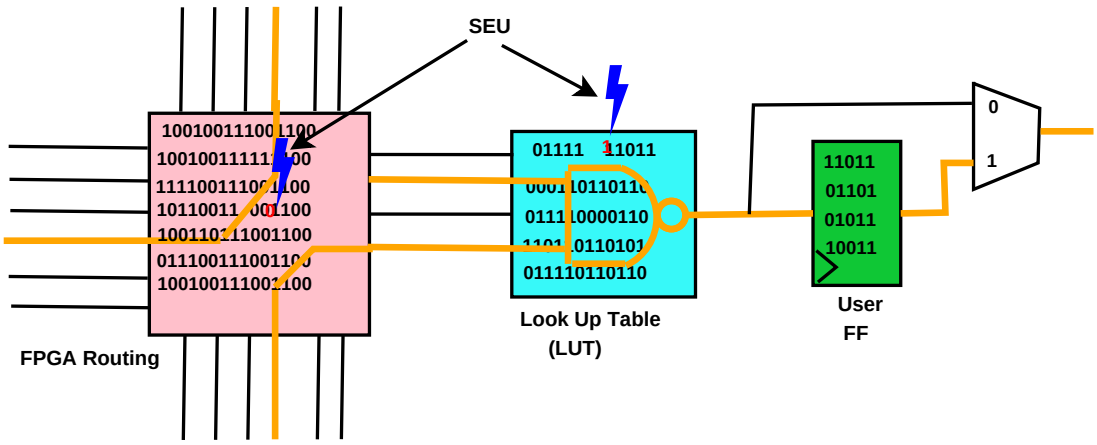

Figure 2.6: SEU causes error in user design routing of logic AND in an FPGA configuration memory

Figure 2.6 shows a SEU (sometimes known as bit flips) which changes the configuration memory bit and an unintended signal rerouting and change in the logical function of the design happened due to this change in the configuration

memory bit.

Effect of SEU and MBU in FPGA can be broadly classified into two categories: Temporary or transient and permanent [72]. Effect of transient faults into any combinational or sequential logic will be transferred to the output of flip-flop or memory if its period of occurrence is higher than the period of the clock which drives the circuit. The transient faults frequently affect the user logic and memory elements and it can be corrected within a few seconds either by rewriting the memory content and logic cell(s) or by built-in error detection and correction code. Sometimes SEU or MBU can directly affect the configuration memory of FPGA. Then this effect is permanent in the sense that this error cannot be corrected unless we reconfigure the FPGA. This is also one kind of permanent fault, but it is recoverable [72]. When charge particles permanently damage the logic or switching blocks within the FPGA then it is also termed as permanent fault, but they are nonrecoverable. This error can be sorted out only by replacing the defective logic block physically (normally it is done by re-routing [73]). In [73], permanent recoverable faultshave only been considered.

One of the common solution to prevent FPGA devices from the effect of radiation is to use radiation hardened (Radhard) FPGAs. But they are more costly compared to the COTS FPGA [74] and are also few generation behind than COTS FPGAs. So in different commercial applications, COTS FPGAs are used with various error mitigation techniques. Triple modular redundancy (TMR) [75] and concurrent error detection (CED) [76] are most commonly used techniques in FPGA for error mitigation. In TMR, three identical logic blocks having the same functions are connected in tandem and final output from the system will be obtained through majority voting. An additional error detection circuit is used along with the main circuit in CED [76]. When any error is detected, the main circuit recomputes or rolls back the whole operation from the beginning. The above mentioned schemes consume large area, huge power and are not suitable for real time applications. Large area overhead of TMR can be reduced by using TMR only in critical portions of a circuit instead on the whole design which is known as partial TMR as described in [77]. The problem related to extra overhead can be reduced by using scrubbing. In scrubbing, before downloading the bit file into the configuration memory, one copy of original bit file (also known as the

golden copy) is stored separately in a Radhard memory. During run time, this golden copy is downloaded into configuration memory with some periodic interval. It reduces the effect of accumulated error in FPGA and increases the lifespan of the FPGA [78]. An alternative to this method is known as configuration read back as described in [72]. In this scheme data from the configuration memory is continuously read back and cyclic redundancy check (CRC) operation is done in a separate memory. If error is detected, the FPGA has to be re-programmed again. Sometimes TMR can also be used intelligently with scrubbing to reduce the effect of single event upset as described in [79] for Virtex FPGAs.

The above mentioned schemes are not applicable in real time because they are based on either time or space redundancy or combination of both. The error detection and correction (EDAC) codes play an important role in many successful SEU and MBU mitigation schemes. The SEU and MBU can be protected by using single error correcting hamming code, which may not be sufficient for reliable communication in high speed DAQ system. So multiple error correction codes can be applied in this kind of high speed DAQ. Several multiple FEC have been presented in different papers *e.g.* BCH code [80], Reed Solomon encoding and Reed Muller codes [81]. Different commonly used error correcting codes like BCH , RS [82], low-density parity-check (LDPC) [83] can be an alternative solution to protect configuration memory of FPGA against soft error, but the hardware complexity and resource utilization of the above mentioned codes are very high. So simple low complexity error correcting codes are always preferable to protect configuration memory in FPGA. Error detection and correction on full configuration memory using different error correcting codes will increase system latency. At the same time downloading of the bit file after error correction will momentarily stop the system operations. These techniques are not a good choices for any real time applications. To alleviate these effects Dynamic Partial Reconfiguration (DPR) can be used with error detection and correction codes. In this case error correction code will correct only a particular portion of the configuration memory at a time and download only the corrected portion of the bit file without hampering the normal system operation involving the other blocks in the design.

In all of the above mentioned papers the authors did not considered anything

about SEU mitigation in high speed data acquisition system to work in an adverse environmental condition like deep space experiment [84] or in HEP experiment. The scope of our work also takes into consideration an efficient design of the various stages in the embedded DAQ chain to meet the high rate data acquisition requirements.

Chapter 3

Crypto Algorithms and its Embedded Processor Based Implementation

FPGA devices are coming very strongly in the embedded system design due to the availability of ready to use resources, parallel logic operations and reconfigurable designs. The usage of FPGA systems in real time domain is also a very fruitful proposition as the FPGA devices are coming with processing cores for real time data processing [85] [86]. In case of a complex embedded application environment involving a large amount of processing tasks and involving data communication among the nodes, we can use multiple FPGA devices acting as transmitter and receiver nodes respectively. To make this possible we have to establish a real time data communication between multiple FPGA devices and to make it even better, we have to apply data encryption techniques for making this communication secured.

In this chapter we demonstrate the design and implementation of a RSA algorithms by developing suitable embedded system and Software design on Xilinx Spartan-3E (XC3S500E-FG320) device, the implementation has been tested successfully for real time serial data communication between multiple FPGA devices using the RS232 serial interface. This development work is also useful for the embedded applications, which requires on board execution of security algorithms.

The system is optimized in terms of execution speed and also has been verified using Xilinx ChipScope Pro a real time debugging tool.

3.1 Introduction

Data security has been the key criterion of most of the present day applications like communication devices, network services on Internet, on-line money transactions, digital cash, electronic mail etc. These applications are implemented using various cryptographic techniques of varied nature. Cryptography is the art of using mathematics to address the issue of information security. Cryptographic systems can provide the objectives of information security: confidentiality, user authentication, data origin authentication, data integrity and non-repudiation [87]. In contrast to symmetric-key crypto systems, public-key crypto systems are capable of fulfilling all of these objectives. However, in order to be fast enough and feasibly practical in the applications mentioned above, public-key schemes have to be implemented in embedded system, which is likely to provide the ease of installation as well as security from tampering. Among the existing public key algorithms the RSA is the most widely adopted [36]. Its security lies in the difficulty of factorizing large integers. Several Montgomery designs have been proposed for ASIC and FPGA implementations, based on limited resource availability to satisfy the computational burden [37], [38], [39]. FPGAs have large configurable logic resources in addition to dedicated high-speed ALU logic for operations such as multiplication. The key reasons for choosing FPGA for the implementation of the security algorithms lies in the flexibility to change key length or modify the embedded algorithm to respond to design flaws or changes in standards or data formats. A lot of research work exists in the implementation of RSA algorithm on FPGA [40], [41] but there exists a very few research papers on the FPGA based implementation of RSA for real time data communication [42], which essentially involves execution of the security algorithm in coherence with the input data coming from the communication channel in real time. The need of FPGA based real-time data communication is of special interest where FPGA based data acquisition systems are build to capture, store process and transmit data generated from real time experiments [21] [22].

Novelty and contribution in this chapters are,

- Efficient modified modular and binary modular exponential method has been proposed for design of embedded RSA processor
- Efficient implementation of another public key cryptography algorithm ECC as embedded processor in FPGA
- Established secure real time communication between multiple FPGAs through respective RS232.

efficient implementation of RSA encryption and decryption algorithm for real time data communication involving multiple FPGA systems and it has been tested over a RS232 communication link established between two FPGA based embedded systems.[1] The algorithm and real time scenario of the system is analyzed and its hardware utilization proves to be better compared to the related research works.

The section plan for this chapter is briefed as follows: Section 3.2 brief the design overview, the RSA algorithm and the detailed embedded architecture. Algorithms for modular exponential operation is described in Section 3.3. Section 3.10 deals with the details of the implementation which include two experiments, the first which shows the loop back method of implementation of cryptography and second one shows the actual real time secure communication between multiple FPGA. Concluding remarks are presented in Section 3.12

3.2 Design Overview

An exceptional feature that can be found in RSA algorithm [36] is that it allows most of the components used in encryption to be re-used in the decryption process, which can minimize the resulting hardware area. In the next subsections we have described a basic background mathematics of RSA algorithm, architectural design description and components which are used in in FPGA, hardware

[1] Outcomes of this chapter was communicated in ICCCD, 2011,IIT Kharagpur, ICCCS 2011, Rourkela, and ICDCS 2012 (C1,C2,and C5)

implementation of mod operation, hardware synthesis with implementation and result

3.2.1 RSA Algorithm

Key generation in RSA is done by randomly selecting two primes p_1 and p_2 of the same bit length. We compute,

$$n = p_1 \times p_2 \quad and \quad \varphi(n) = (p_1 - 1) \times (p_2 - 1) \tag{3.1}$$

and select an integer e from the interval [1, $\varphi(n)$] so that gcd $(e,\varphi(n)) = 1$, i.e., the greatest common divisor (gcd) of e and $\varphi(n)$ is 1. We compute $d = e^{-1} mod \quad \varphi(n)$. The public-key is $K_e = (n, e)$ and the private-key is $K_d = (n, d)$.

Thus, a plain text block M is encrypted to a cipher text block C as follows:

$$C = M^e \quad mod \quad n \tag{3.2}$$

$$M = C^d \quad mod \quad n \tag{3.3}$$

Equation 3.2 and 3.3 represent the RSA encryption and decryption process. They are mutual inverses and commutative, due to symmetry in modular arithmetic [87].

3.3 First Algorithm for modulus exponentiation operation

The modular exponentiation operation is simply an exponential operation where the multiplication and squaring operations are modular. The exponential heuristics developed for computing M^e are applicable for computing $M^e(modn)$. In RSA algorithm e and d are selected to be very large numbers for the purpose of better security. As e and d are very large numbers so the value of M^e and C^d will result to a mammoth value. Due to computational limits (stack overflow) often the result of those values obtained are incorrect. To handle this situation

in the domain of hardware and software implementation, an intelligent algorithm is needed to achieve a higher efficiency. Integer M, e, n are given to calculate the cipher text C in the RSA algorithm. Here we use the properties of modulus operation i.e. if

$$x = ab \quad (mod \quad c) \tag{3.4}$$

then we can write

$$x = (a \quad mod \quad c) \quad (b \quad mod \quad c) \quad mod \quad c \tag{3.5}$$

to increase the efficiency of calculation. Here a, x, b and c all are integers in the Equations 3.4 and 3.5. Let us assume for a large number e=5, n=3233 and M= 123, the values marked in each iteration according to the algorithm shown in Figure 3.1, is given in the Table 3.1. Figure 3.1 explains the flow diagram for the Modular algorithm which is described in this section.

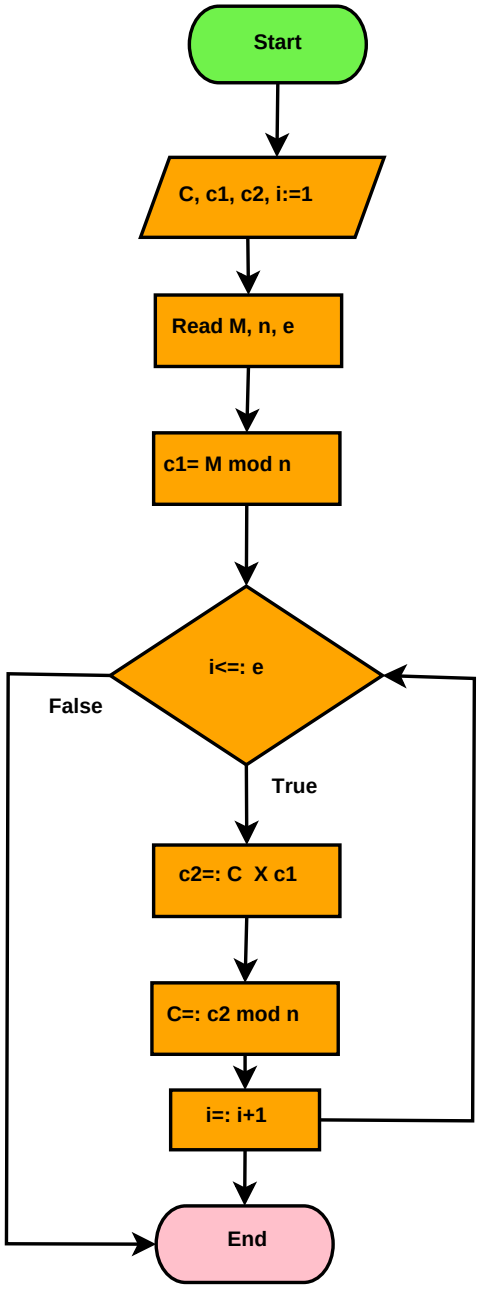

Figure 3.1: Flow diagram for modulas exponential operation's hardware implementation

Table 3.1: Stepwise results of mod calculation algorithm

i	c1	Step3 (c2)	Step4 (C)
1	123	123	123
2	123	15129	2197
3	123	270231	1892
4	123	232716	3173
5	123	390279	2319

From the results generated in the different pass of the iteration cycle shown in Table 3.1, it can be said that from Figure 3.1 the algorithm describes the value of *c2* in step 3 which is very small compared to the value of M^e. So we have successfully reduced the memory requirement in the implementation logic.

3.3.0.1 Modular Exponential operation with Binary Method

Doing a "modular exponentiation" means calculating the remainder when dividing by a positive integer n (called the modulus) a positive integer M (called the base) raised to the e-th power (e is called the exponent). The first rule of modular exponential is that we do not compute M^e as reported in [88] because both M and e may be very large in case RSA algorithm. If we going to store M^e a large no of memory space needed. The temporary result must be reduced modulo n at each step of exponentiation this is because of space requirement of Me is enormous . If M and e have 256 bit each we need 1080 bits to store Me .This number is approximately equal to the number of particle of universe. In order to compute this total bit capacity all computers in the world we can make an assumption there are 512 million computers, each of which has 512 Mbytes of memory. Then total number of bits available on all computers will be 1018 . So we have no way to compute M^e. We have many hardware compatible algorithms to implement RSA; i.e. *Binary Method*, *M* array Method etc as reported by [88]. For our implementation the main concern was about the resources usage and execution time. Due to these two merits processor should have minimum number of modular multiplication to compute M^e mod n. According to the reference [88] *binary method*

is one of the fastest methods to compute modulus exponent without computing M^e. In case of high secure data communication where exponent term (public and private keys) is large enough the encryption and decryption algorithm would take large processing time which would overwhelmed the processor for high speed communication like.

3.3.0.2 Description of Binary Method

We present the RSA algorithm in Fig. 3.2, where both the encryption and decryption algorithms have the same steps, but only differ in the exponent term. This proposed algorithm is a modified version of that in section 3.3, the modification is done to handle the exponential operation in a better way. Block diagram of the architecture of proposed algorithm is shown in Figure 3.3

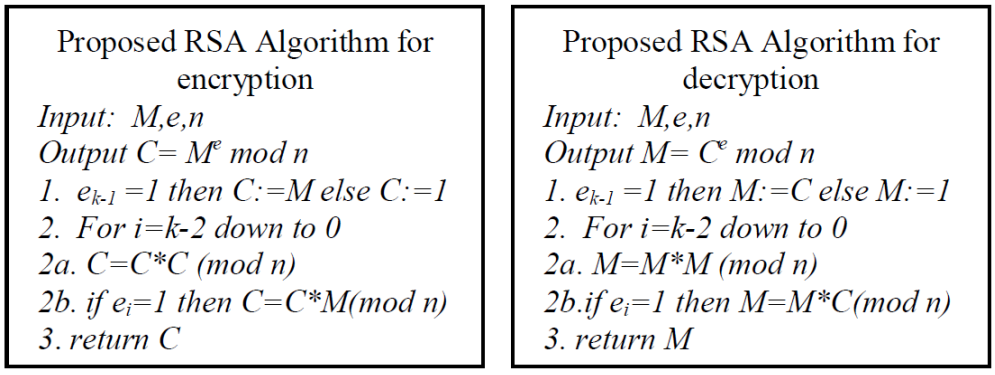

Figure 3.2: RSA Algorithm with modified binary method

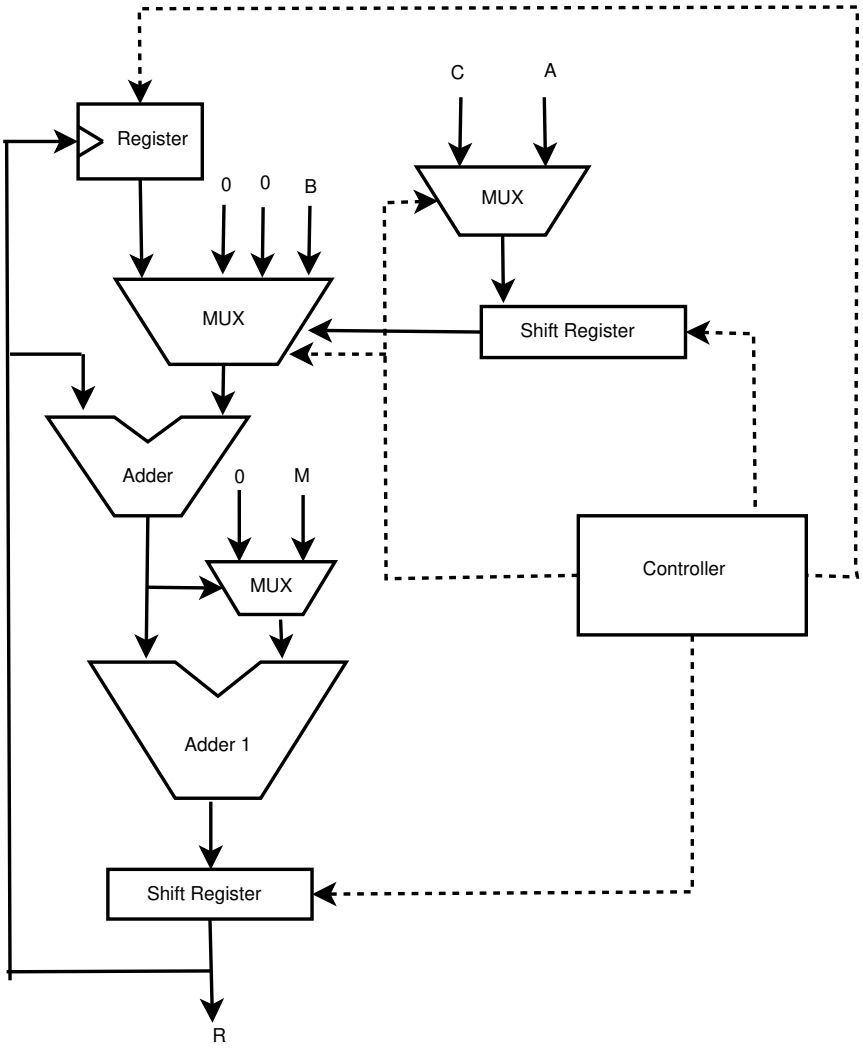

Figure 3.3: Architecture of modified binary method

Table 3.2: Stepwise Result of Mod Calculation Algorithm

i	ei	Step 2a	Step 2b
6	1	1	M
5	1	M^2	M^3
4	1	M^6	M^7
3	1	M^{14}	M^{15}
2	0	M^{30}	M^{30}
1	1	M^{60}	M^{61}
0	1	M^{122}	M^{123}

In our algorithm we have 3 inputs M, n and e where all the inputs have k number of bits. This *binary method* checks each of the bits of the exponent term from left to right [89]. Depending on the scan bit value a squaring and a subsequent multiplication operation is performed for each step (2a and 2b). As an example; let e=01111011, Which implies k=8. Since e^{k-1}=0, we take C=1. The *binary method* proceeds as shown in Table 3.2, which shows M^{123} is 7+5=12. It is very obvious the number of clocks to execute modulus exponent only depends on the number of modulus multiplications rather than the value of exponent term.

3.4 Embedded Architectural Design

This work is implemented using the Xilinx Embedded Development Kit (EDK) 11.4 and Xilinx Spartan 3E FPGA prototyping board has been used for the hardware and software implementation and testing. Using the Xilinx platform studio from EDK the hardware design of the embedded system has been developed. A soft core 32-bit RISC processor Micro Blaze has been used as a CPU for this embedded computing unit and all the other soft core peripherals that have been utilised in this design are Universal asynchronous receiver/transmitter (UART) 1(used for RS232 DCE port), UART 2(used for RS232 DTE port), LEDs, DIP switches as General-purpose input/output (GPIO) and EMAC (10/100 Mbps) for Ethernet connectivity. The blocks used to build up the FPGA based embedded computing unit is illustrated in Figure 3.4.

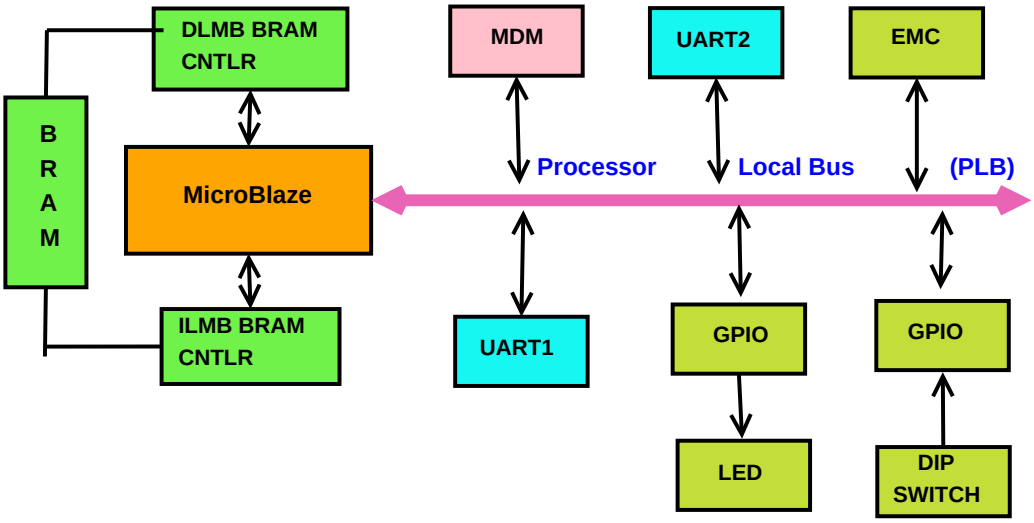

Figure 3.4: Processor and its peripherals of the designed system used in each FPGA System

Figure 3.5 shows the top level view of the designed system used to communicate between the FPGAs in real time. Two Spartan 3E boards are connected

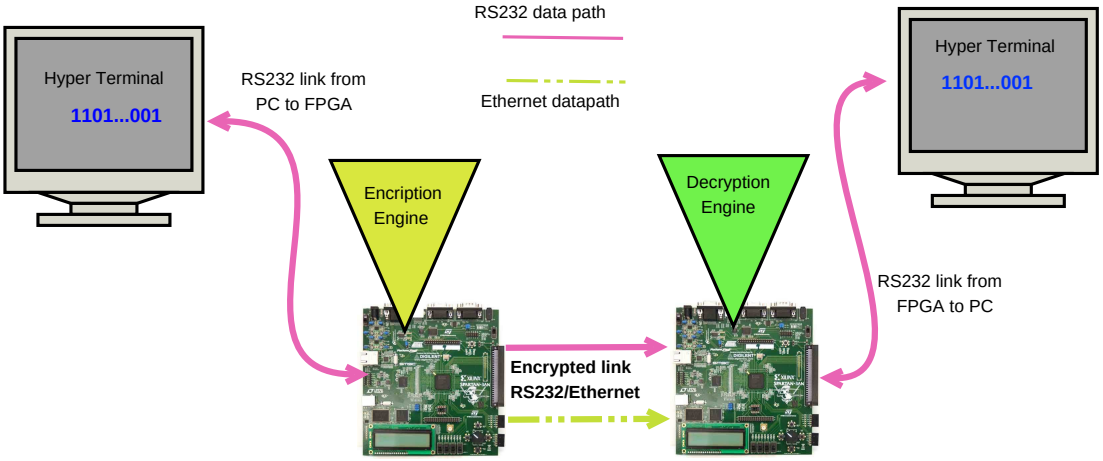

Figure 3.5: Abstract view of the complete system and secure communication flow using FPGA and Terminal

through the RS232 link and the host PC is also connected to one of the FPGA system for transferring the test data for communication, using a RS232 link. We have used one of the boards as the encryption engine and another as the decryption engine for setting up real time secure data communication. Real time data

is generated by the PC keyboard through the HyperTerminal and it works as the input from the host PC to the Spartan 3E board which is used as the encryption engine. Encrypted cipher data is then sent to another Spartan 3E i.e. decryption engine by using RS232 serial cable. Encrypted data is received and decrypted by the receiver side Spartan 3E FPGA and the decrypted data is displayed at its LED ports. The necessary software for this design is written using the feature-rich C/C++ code editor and compilation environment provided within the Xilinx Software Development Kit (SDK) [54]. SDK works with hardware designs created with Xilinx Platform Studio (XPS) . The implementation system setup is shown in Figure 3.5. The block diagram of the UART, the system that is used for establishing the serial communication between the two FPGA systems is shown in Figure 3.6.

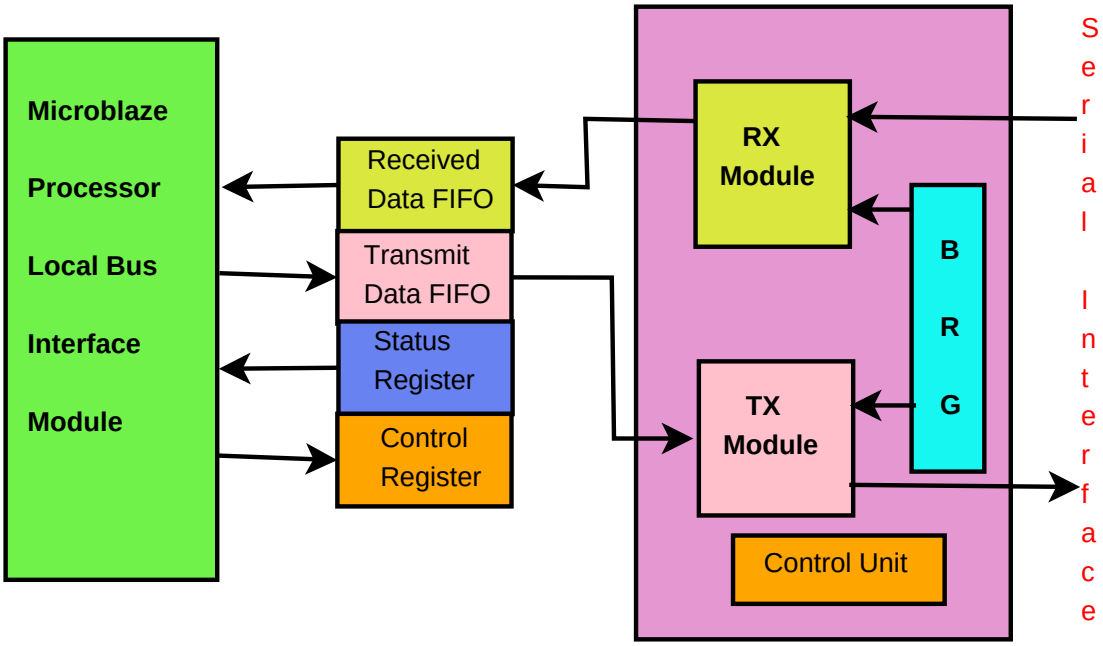

Figure 3.6: UART system module attached with the Processor Local Bus

Here "BRG" means "Baud Rate Generator" which controls the RS232 channel speed for data communication. For sender to receiver communication each side , *i.e* sender side and receiver side must maintain the same baud rate otherwise communication can not be established and data will be lost.There are separate First-In First-Out (FIFO) blocks used for reception and transmission, namely

Received Data FIFO and Transmit Data FIFO. FIFO full or empty status can be found in the Status register whereas the control of RX (Receiver) and TX modules are done by the Control register. UART module is connected with the Microblaze processor using the Processor Local Bus (PLB) as shown in Figure 3.4.

3.5 Hardware Architectural Design

The processor based design in xilinx platform is shown in Fig 3.7 where mainly two type of processors are available. The first is the Xilinx MicroBlaze soft processor core [90], which is synthesized using the available resources in the target FPGA device. The second is the IBM PowerPC hard processor core[91], which is integrated into certain versions of Virtex FPGA devices and therefore it does not require additional resources of the device.

3.5.1 MicroBlaze

The MicroBlaze core is a 32-bit Harvard RISC architecture with a rich instruction set optimized for embedded applications. The processor is a soft core, which is implemented using general logic primitives rather than a hard, dedicated block in the FPGA. The MicroBlaze soft processor is supported in the Xilinx Spartan and Virtex series of FPGAs.

The MicroBlaze solution is designed to be flexible, giving the user control of a number of features such as the cache sizes, interfaces, and execution units. The configurability allows the user to trade-off features for size, in order to achieve the necessary performance for the target application at the lowest possible cost point.

3.5.2 Power PC

The PowerPC (Performance Optimization With Enhanced RISC) Architecture, is a 64-bit specification with a 32-bit subset. Almost all PowerPCs with a few exceptions are 32-bit.

PowerPC processors have a wide range of implementations, from high-end server CPUs to the embedded CPU market. PowerPC processors have a strong

embedded presence because of good performance, low power consumption, and low heat dissipation. PowerPC 405 is a 32-bit RISC hard IP core. Virtex II Pro platform FPGAs provide upto two PowerPC 405 cores on a single device. IBM PowerPC 405 has wide acceptance in performance oriented applications as well as comprehensive 3rd party tools support. It offers excellent performance vs power characteristics in a small die area. PowerPC core supports a system frequency of atleast 300MHz, corresponding to more than 420 Dhrystone MIPS. The processor frequency can be dynamically changed for reduced system power dissipation.

3.5.3 Microblaze based Hardware Design

The hardware platform of an embedded system created with the EDK package includes one or two processors, along with various peripherals and memory blocks. Each processor and peripheral core can be customized. Implementation parameters control optional features of the cores and define the addresses assigned to each peripheral. These addresses are mapped in the memory address space.

Communication between a processor and on-chip BRAM memories is performed via a Local Memory Bus (LMB). This bus provides single-cycle access to dual-port BRAM memories and is split into instruction LMB and data LMB. Peripherals can either connect directly to the Processor Local Bus (PLB) or to the On-Chip Peripheral Bus (OPB). The PLB and OPB buses are connected via a bus bridge. After compilation of all hardware component synthesis, its generate bit file.

So, in XPS of EDK platform, we can select the required processors and the details of peripherals required for the design. FPGA-based embedded computing unit is shown in Fig. 3.7. Depending on the number of processor used in this

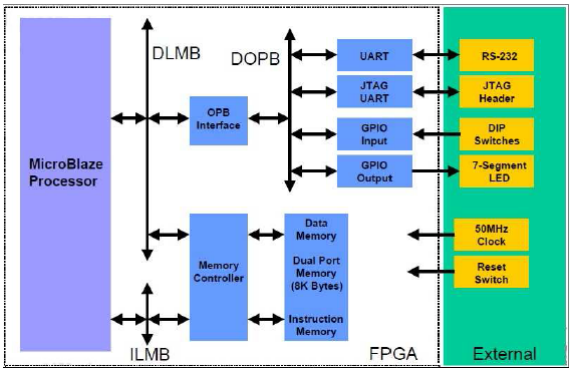

Figure 3.7: Processor Based Architecture

crypto applications two types of architectures are implemented in MIcroblaze bases FPGA platform.

3.5.3.1 RSA and ECC Single Microblaze

As shown in Fig. 3.8 in a single processor architecture, the FPGA receives the input via the RS-232 port through UART and then the input plaintext message is encrypted using RSA and ECC algorithms, which are running in the MicroBlaze processor of the FPGA. After encryption, the cipher text is sent back to the Host PC and is seen in the Hyper Terminal using serial communication between the board and the PC. Fig. 3.8 shows the flow of steps in the design. Processor working at 50 MHz clock frequency for RSA and ECC in Spartan 3E FPGA and 100MHz clock frequency for ECC in Virtex-4 FPGA. The ECC and other crypto alogorithm paper was published in coference papers **??**, **??** and **??**.

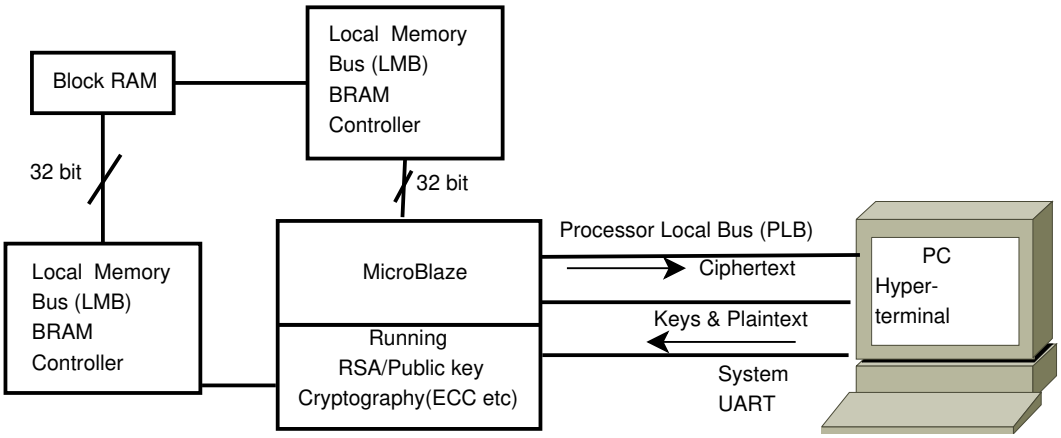

Figure 3.8: Single MicroBlaze architecture

3.6 Experiment 1

A real time data is received from the key board using the hyper-terminal application of a host computer which is then being sent to the board using RS232 (9600 bps, no parity bit) serial cable in the DCE port on the board, then the received data on the board is encrypted by executing the encryption program i.e. the data has been converted from plain text to cipher text. The cipher text is transferred to the PC hyper terminal via the RS232 DCE port (9600 bps, no parity bit) and

it is also shown on the Light-emitting diode (LED)s for the verification of the result. The example data for this experiment is shown as below and also the snapshot of the hyper-terminal message is shown in the Figure 3.9.

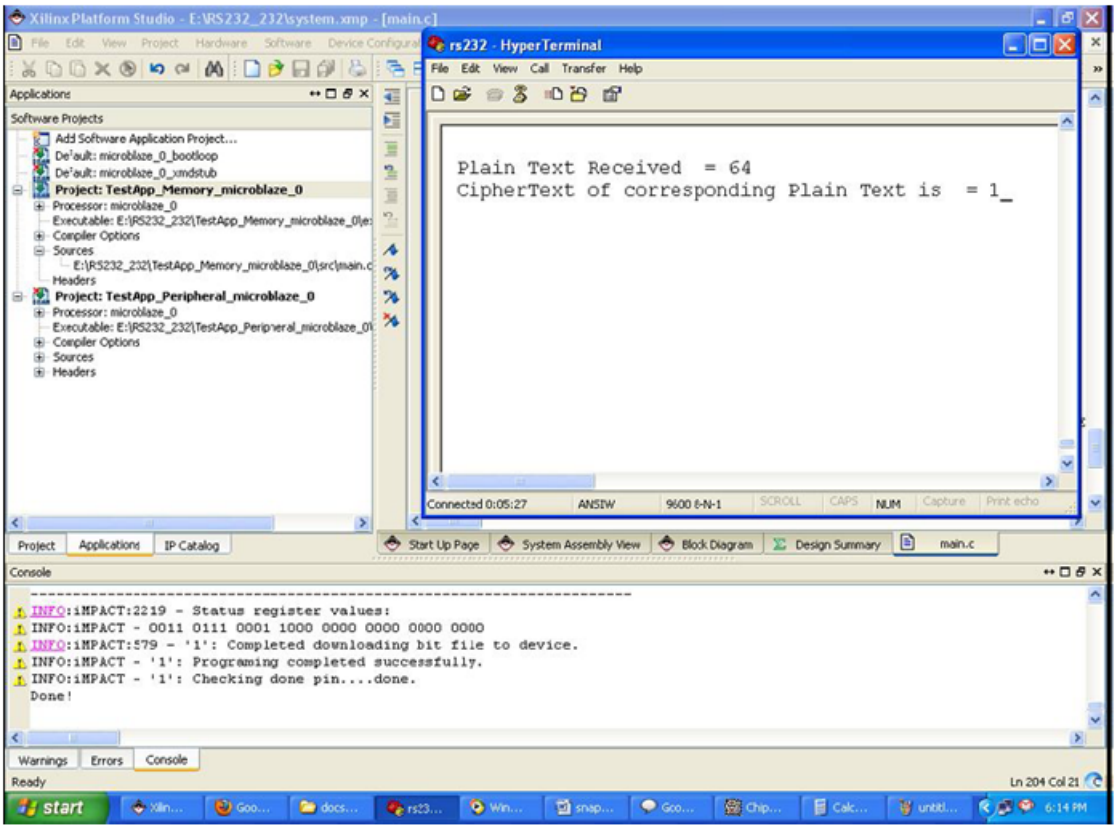

Figure 3.9: Real time data accepted from keyboard to FPGA and the cipher text is shown on the hyper-terminal CAP

Encryption: Message $M = 64_H$ (i.e."d" is pressed from the keyboard as a real time input; Public key $e = 3_H$, $n = 21_H$; Cipher text $C = 1_H$;

3.7 Experiment 2

Real Time data is given from keyboard to Spartan 3E FPGA (where encryption engine is running) and data is converted to cipher text and the cipher text is transmitted through the RS232 DTE port, using RS232 cable and sent to another Spartan 3E FPGA (where Decryption engine is running) through the DTE port

on the receiver Spartan 3E board and the cipher text is received which is converted to plain text by running the decryption process. The plain text data is displayed at its LED port. As there are only 8 LEDs available in the Spartan 3E board we can see the error free result on the board LED for 8 bit decrypted message, for larger size of decrypted message we have to use the other peripheral connections of the board to read the message. The example data for this experiment is shown as below: Encryption: Message: $7B_H$, Public key, e = 11_H n = $CA1_H$ Decryption: Cipher Text=357_H, Private key=$AC1_H$ Decrypted Message: $7B_H$ We have also used the Chip scope Pro Analyzer tool of the Xilinx for capturing the real time bus signals of the hardware design. The experiment-1 results are shown in Figure 3.10,

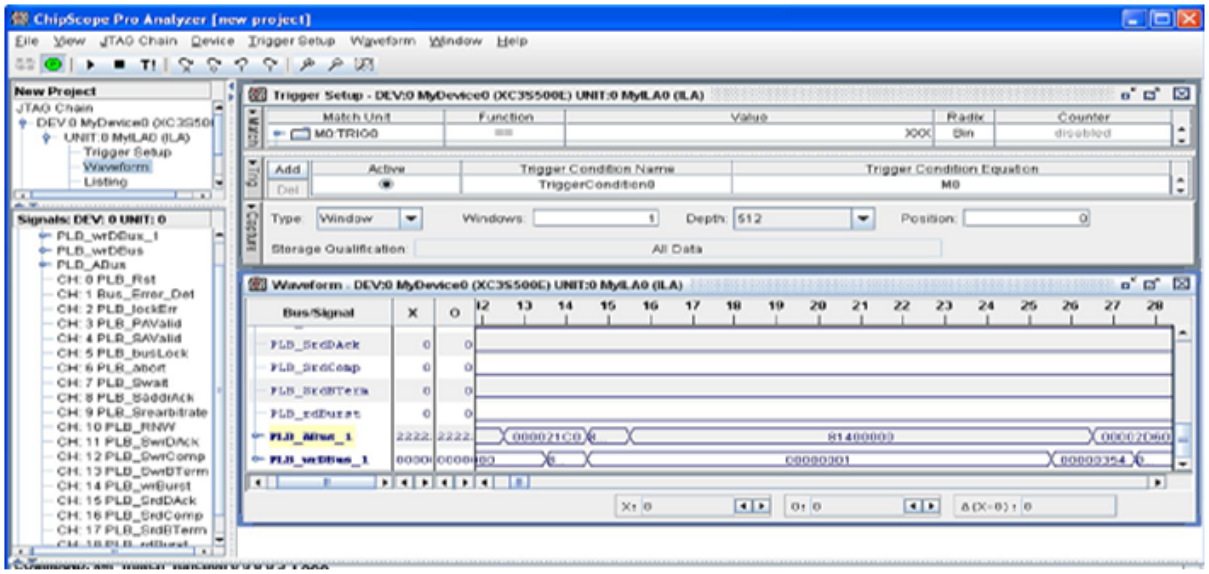

Figure 3.10: Chipscope Pro analyzer results for experiment 1

i.e. the cipher text data (00000001) appears at the addresses of the LED ports (81400000 address of the PLB). The results of Chipscope Pro Analyzer for the experiment 2 i.e. the decrypted message (0000007B) appearing at the LED ports ((81400000 address of the PLB bus) is shown in Figure 3.11,

The above results prove that we have successfully implemented the algorithm for encryption and decryption on the Xilinx Spartan 3E board and it has also catered the need of real-time secured data communication through the serial

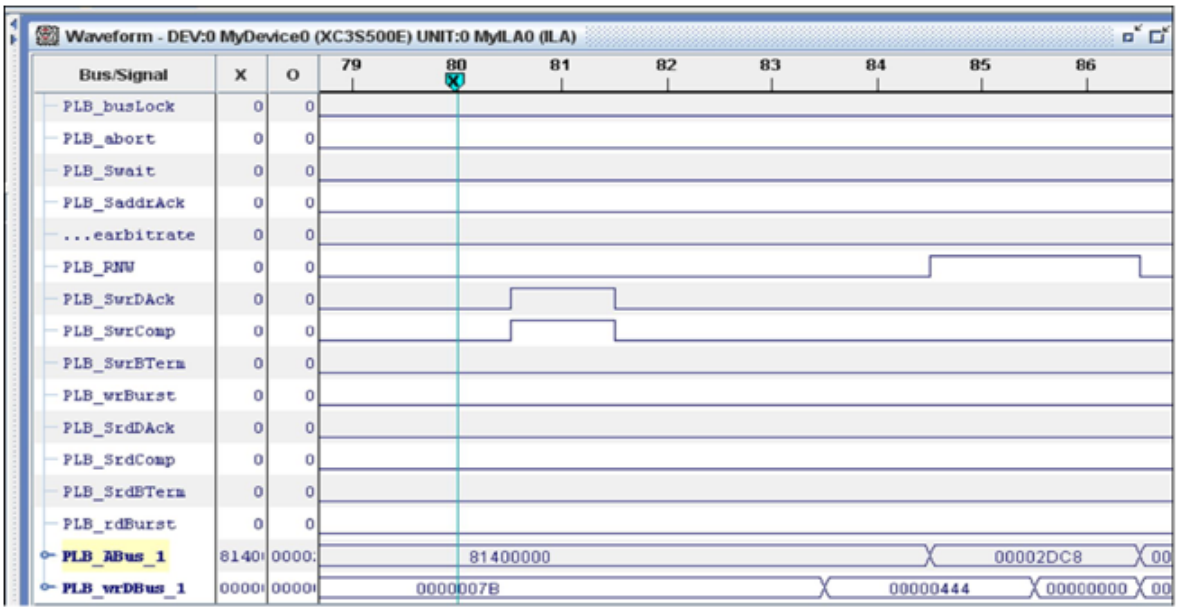

Figure 3.11: Chipscope Pro analyzer results for experiment 2

Table 3.3: Encryption and decryption time (32 bit key size)

Process	Clock cycles	Clock frequency (MHz)	Time (ms)
Encryption	5176	50	0.103
Decryption	1116878	50	22.337

RS232 interface between two FPGA systems. Real time Encryption and decryption time needed tor the experiment in Spartan 3E board is shown in Table 3.3.

3.8 Theoretical Result

Let us take a public key is ($n = 3233$, $e = 17$), and private key (n=3233, d=2753) for this algorithm. Now if we select a data M=123. Then in the by encryption process the generated cipher text will be

$C = 123^{17} \mod 3233 = 855$

To decrypt C=855, we used decryption algorithm and get

$M = 855^{2753} \mod 3233 = 123$

3.9 Synthesis

The VHSIC Hardware Description Language (VHDL) code is synthesized for the Xilinx Spartan 3E device family. From the synthesis result,we observed that the design has utilized 2667, 4 input LUT out of a total number of 9313, which shows a utilization of 28% of resources in our design. The

3.10 Implementation and Results

The required embedded architecture is successfully implemented (Figure 3.12) in the Xilinx EDK 11.4 using the XPS and the required C program is written

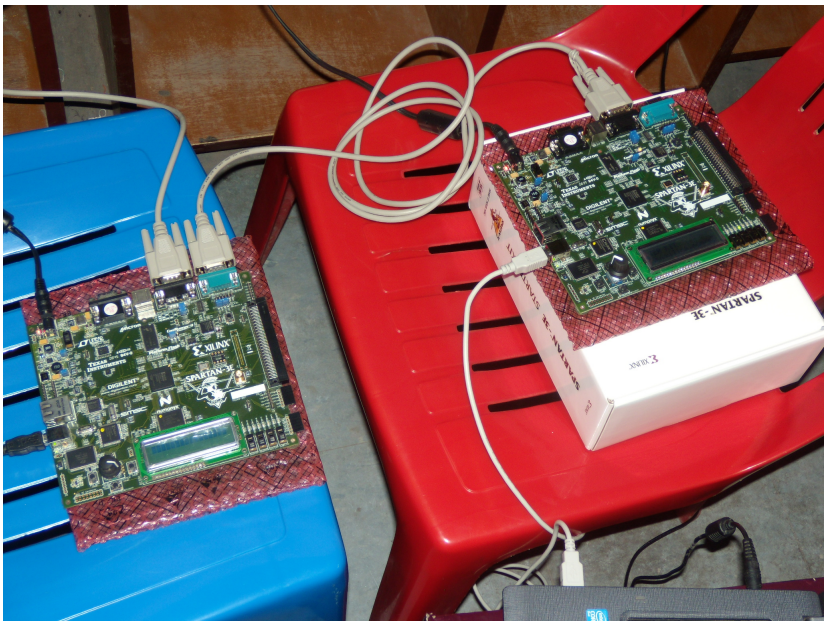

Figure 3.12: Real lab picture of two connected Spartan 3E FPGA boards through RS232 Cable

in the SDK [54] platform. After generating the 'configuration bit' file through successful synthesis, we downloaded this to our target Xilinx Spartan 3E board. The experimental Spartan 3E board is shown in the Figure 3.13. We have tested the real time execution of the program in the hardware in two ways, which are described below:

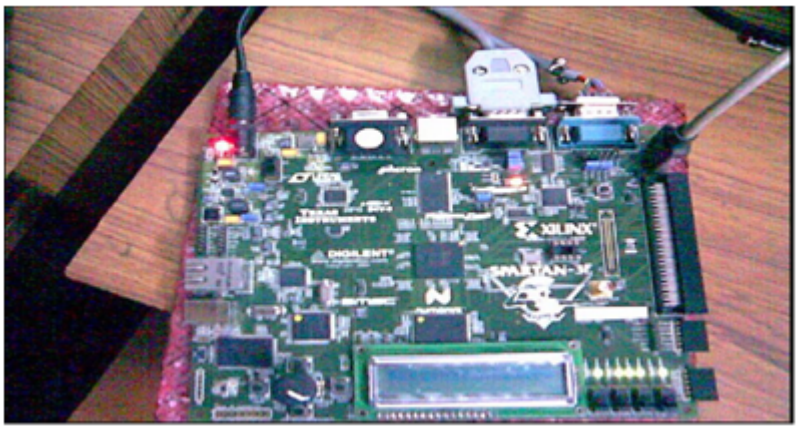

Figure 3.13: The Spartan 3E board

3.11 Comparison with Existing Work

Due to the limit of supported integer values (32 bit) in Microblaze based EDK design, we chose 32-bit key size for experiments results. But this limits is not a bound condition when we chose the same design for direct hardware leve design in FPGAs, but yes, may be sometime bottleneck can arise(in Spartan 3 or other lower xilinx family FPGAs) due to the limited LUT/FF resources for design large Key values. For real world key size (like 1024 bit value), a resource utilization and comparison table for 32 bit and 1024 bit key size RSA implementations are given in Table 3.4 Table 3.5 respectively.

Table 3.4: Comparison of RSA implementation of key size(32 bit)

Reference Design	Area (LUT)	Time(ms)
Xilinx Spartan-3 [92]	15235	1.57
Xilinx Spartan-3 [93]	28045	1.16
Altera [94]	10427	0.635
Xilinx spartan 3e[our design]	2667	0.103

Our design gives better performance with respect to other design. Though the implemented board families are different, but we chose to compare the design in

terms of required clock trigger values irrespective of their board supported clock frequency.

Table 3.5: Comparison of resource utilization and speed with existing 1024 bit RSA implementation

Device	Slices	Flip-Flop	Time (ms)
V2p100 [95]	12791	17932	1.0060
V4fx100[sub pipeline] [95]	14910	17831	1.00059
V4fx100[pipeline] [95]	14329	119794	0.1004
Xc5vlx110t[proposed design]	2083	1059	0.03007

Similarly in another Table 3.6, resource utilization and other parameters comparison for another public key cryptography algorithms (ECC) implementations is given with our proposed one.

Table 3.6: FPGA-based ECC point multiplication on Koblitz curves

Curve basis	Field	Board	Area	Frequency (Mhz)	Time(us)
Polynomial [96]	233	Stratix II(EP2S180F1020C3)	38056 ALM	181.06	8.09
Polynomial [97]	233	Virtex V4 (XC4VFX12)	2431	155.376	604
Gaussian Normal[98]	233	Stratix V	16421 ALM	246.1	6.8
Polynomial [99][single]	233	Virtex2(XC2V4000)	14091 slices	51.7	8.72
Polynomial [99][parallel]	233	Virtex 2(XC2V4000)	15916 slices	51.7	7.22
Polynomial[96]	163	Stratix II	20525 ALM	187.48	4.91
Polynomial[100]	163	Stratix II	13472 ALMs	155.5	26
Polynomial[101]	163	V5 (XC5LX110T)	2708 slices+5 BRAM	222.65	55
Gaussian[102]	163	V2 (XC2V2000-6)	6494 Slices+6 BRAMs	128	–
Polynomial[Proposed]	233	Kintex 7	22344	100	11.55

Time needed for execution of the decryption algorithm is higher than the encryption because, the private key value ($AC1_H$) is higher that of the public key value (11_H). So execution time for decryption message is higher (see Equation 3.2 and Equation 3.3).

3.12 Summary

The primary goal of this work was to perform a real time encrypted data transmission between multiple FPGA systems exhibiting a significant level of security and providing a faster processing time. Also the resource utilization has been optimized with respect to other research papers which dealt with this kind of implementation. We have also tested the signals on the bus in real time to prove the validity and the efficiency of our experiments, which is exactly the same as our theoretical results. The maximum bit length of the key used in our experiment is 32 bit, this is because the PLB of the Microblaze core is of 32 bit, the design can be implemented for higher bit length of key with certain modifications. Though we have performed the real time encryption and decryption of data through RS232 serial communication the technique remains the same for Ethernet data communication using the EMAC core [103]. Efficient design of other crypto algorithms like AES 128, 192, 256 have been done on embedded processors as well as dedicated logic cores involving parallelism, in the course of this research work, as presented in the following chapters.

Chapter 4

Design of Crypto Co-processor and its Implementation

Hardware implementation of cryptographic algorithms always gives a better performance and security in comparison to its equivalent software implementation. In an embedded system, presence of a dedicated processor/co-processor for handling the security algorithms, increases the performance of the system as well as gives an opportunity of parallelization of operations among the system main processor and that of the dedicate processor for running the cryptographic algorithms. In embedded systems terminology point of view the dedicated processor, whose role is only to handle the security tasks and is associated with the main processor as a custom processing block, is named as crypto co-processor.

In this chapter, a co-processor based architectural design and the related embedded system implementation for the advanced encryption standard algorithm (AES-256) is presented. The proposed design is an FPGA based architecture, having a custom crypto co-processor for executing the encryption and the decryption operations, which also communicates with the main processor core with the high speed FSL as and when required. The proposed design allows our hardware to perform multi-tasking, which is performed by a dedicated crypto co-processor executing the crypto functions and the main processor core executing the other application tasks at the same time. So, in a sense this co-processor based design is novel, as it provides a high speed hardware execution of the crypto functions

as well as provides the flexibility of executing the other application tasks on the main processor core of the system. The implemented work has been successfully tested for encryption and decryption of data over an Ethernet network [8]. To the best of our knowledge, the co-processor based architectural design of AES-256 using FPGA devices, is first of its kind. Our design proves to be efficient in terms of throughput and resource utilization in comparison with the related research works.

4.1 Introduction

Asymmetric crypto-systems, such as the RSA and elliptic curve algorithms, use different keys for the encryption and decryption tasks, whereas the symmetric crypto systems, such as the DES and AES use an identical key for both encryption and decryption processes, which eliminates the key management problem[104]. Most public-key algorithms involve operations such as modular multiplication and exponentiation, which are much more computationally expensive and time consuming. Though public key cryptography solves the problem of secure data exchange but still it has a big disadvantage, mainly speed, in comparison with the symmetric key based cryptography. Symmetric key cryptography is more suitable for the encryption of a large amount of data [3]. In the year 2000, the NIST announced Rijndael as the winner of AES contest, in an effort to address the threatened key size of DES [6]. Though we have a variety of software applications that provides security [48], [49], [50] but in comparison to hardware based implementations they are slow and are more resource hungry. Hence, there is an increasing need of implementation of security algorithms at the hardware level.

A good number of research works can be found that aim towards the hardware implementation of AES algorithm for embedded applications using FPGA [3], [51]. In [3], the authors have presented a parallel reconfigurable hardware implementation of the AES algorithm, which takes a large number of clock cycles for encryption and decryption as it performs all the tasks on a single processor core architecture. In the designs [51], [52] the user designed core is added with the Microblaze processor core [1] using the OPB , which is a slow peripheral bus

and hence we need more number of clock cycles to send/receive data between the processor core and the user designed core. Microblaze is a RISC [53] soft core processor that is available in the Xilinx EDK [54]. A real time board to board communication with secure AES algorithm is shown in [55], where a software based implementation is done using Altera NIOS II processor core [56]. The other related research works on hardware based crypto-system design can be found in [57], [58], [55], [3], [59], [60], [51], [61]. As per the data provided by Xilinx, communication between a hardware and the Microblaze processor core is fastest for the case of FSL bus connectivity [2] and hence FSL bus based design is well suited for high speed applications. In this chapter, the contributions are summarized as: [1]

- Design of a novel co-processor based architecture for executing AES-256 algorithm and its FPGA based implementation.
- The AES-256 crypto co-processor core runs in parallel with the main processor core and the mutual communication for data and key exchanges are achieved by a suitable implementation of the FSL bus.
- The design is optimized in terms of speed and resources in comparison with the other existing AES implementations.
- Development of Ethernet based communication between FPGA system and the back end storage system for testing the design over a real network .

The rest of this chapter organized as follows. Section 4.3 gives a description of the proposed system architecture and its major building blocks. In Section 4.4, we discuss the system evaluation results. Concluding remarks are presented in Section 4.5

4.2 Basics of AES algorithm

The AES algorithm gives high security and good performance and hence it is used in many cryptographic applications. The data size of the AES algorithm is

[1] Outcomes of this chapter was communicated in IEMCON 2012, ICACCI Chennai (C4,C7)

128-bit as mentioned by NIST [6] and is treated as block cipher . The key size of AES algorithm can vary from 128,192 and 256 bits with fixed data size of 128 bits. The exact number of rounds for which the cipher will be generated, varies for different key sizes. It may be 10, 12 or 14 rounds depending on the key sizes of 128, 192 or 256 bits respectively. Every byte of data is defined by Galois Field GF(2^8) with respect to a polynomial $x^8 + x^4 + x^3 + x + 1$. A Byte $b(x)$ in GF(2^8) will be written as:

$$b(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + 1 \tag{4.1}$$

representation of these bytes in a 4×4 a array form is termed as a state and is denoted by $S_{r,c}$, where $0 \leq r, c \leq 3$. Four Bytes in each column of the state is referred as a Word(w_c) having a length of 32 bits. Relationship between Bytes, Word and State is shown in Figure 4.1 where input sequence is denoted by in_0, in_1, ... in_{15} Four basic operations of the AES algorithm are discussed below.

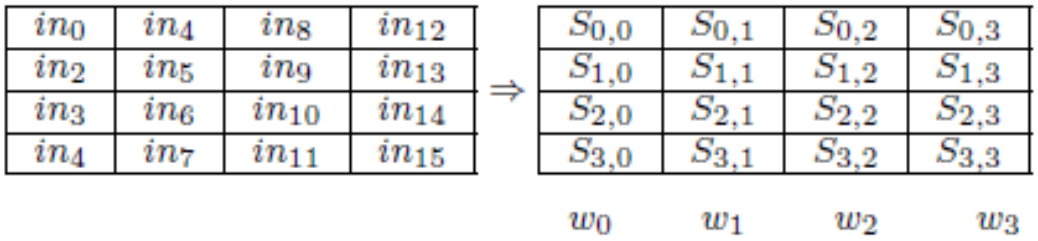

Figure 4.1: Input and State array

SubByte Substitution :- The S-box substitution is a non-linear substitution, which operates on every Bytes of S-box matrices through a predefined substitution table.

ShiftRow Transformation:- A cyclical shift operation with constant offset is applied on each row of the matrix. This step is applied in each round. This operation is done by permutation of the rows by the left circular shift; the first (leftmost, high order) i^{th} elements of row i, are shifted around to the end (right most, low order) where i is the row number.

MixColumn Transformation:- The third step of this algorithm is a resource intensive transformation on the columns of the state in which the four elements of each columns are multiplied over GF(2^8) modulo $x^8 + 1$ by a constant matrix $C(x) = 03x^3 + 01x^2 + 01x + 02$ as shown in equation 4.2

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 03 & 02 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \tag{4.2}$$

AddRoundKey Transformation:-In this operation, round key derived from the previous key state through the key expansion process is XORed with the result of the MixColumn transformation

Figure 4.3 and Figure 4.4 shows the basic flow diagram of AES-256 encryption and decryption without error control module.

4.3 System Architecture

The design efforts in the proposed system architecture consists of activities both in the software and hardware domain. The options for implementing a given algorithm in an FPGA can be done in three different ways as stated below:

- *Defining a dedicated hardware block*: Here the main idea is, a hardware description languages like VHDL, Verilog, Handle C etc. is used to design for hardware implementation any algorithm. This type of design can be found for an example in [3]. The disadvantage of this approach is that, they cannot execute other tasks apart from the specific task for which the design is customized, and hence cannot be used in a realistic system design, where we need to implement some other tasks also.

- *Executing the algorithm using a single processor core*: This is a good option, where we can build the software code of the algorithm for the target

processor core. AES implementation described in [55], based on NIOS II processor core architecture comes under this category. A similar design for RSA implementation has been presented in chapter 3 of this dissertation. In this type of design we do not have the high execution speed as that of the hardware based implementation but we have the option of executing various other tasks apart from the specific algorithm. The drawback lies in the fact that, as the system is based on a single processor core we cannot have multiple tasks in parallel, and so we cannot have the best performance.

- *Using co-processor core*: Our proposed design falls under this category. In this case, we can built a dedicated hardware block i.e the co-processor core, customized as per the requirement of the algorithm. The co-processor core communicates with the main processor core through a bus (FSL in the design), as and when required for data exchanges, and the main processor can execute the other required application tasks concurrently when it is not communicating with the co-processor.

The hardware description of the proposed crypto co-processor is done using VHDL and it behaves as a custom hardware connected with the Microblaze processor core. In the next subsections we will brief the key building blocks of our proposed system. The diagrammatic illustration of the proposed architecture is shown in the Figure 4.2. Proposed AES core is added here with the Micro blaze using FSL bus. Other peripherals of the system Ethernet, UART etc. are added in general PLB.

4.3.1 AES Core Design

The AES algorithm is a round based block cipher and it was initially proposed to operate on 128 bit data blocks as specified by NIST [6]. It is one of the most widely used symmetric encryption algorithms and is advanced in terms of both security and performance. The algorithm is flexible in a sense that any combination of data and key size of 128, 192 and 256 bits are supported. The encryption or decryption takes 10 to 14 rounds of array operations depending on the key size. The key steps of the 256 bit AES encryption and decryption

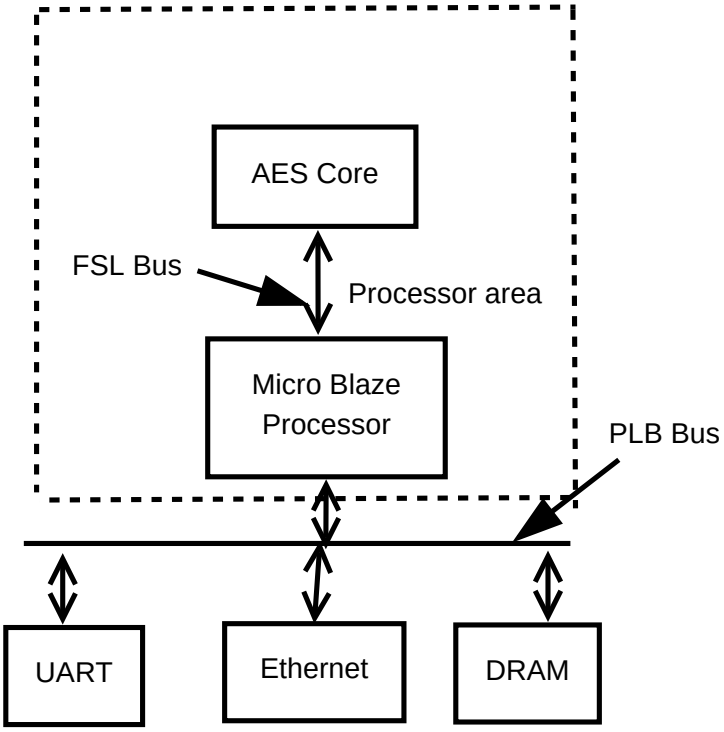

Figure 4.2: Proposed system architecture with Micro blaze

algorithm is shown in Figure 4.3 and Figure 4.4. Our proposed design accepts 128 bit data and 256 bit key to produce 128 bit cipher data.

The AES core consists of four components which are: *AES Algorithm block, Controller, Key Schedule* and the *Input Interfaces*, and all of them are designed using VHDL based modeling and design entry, the logic verification of the core was done using *Modelsim SE*. The internal architectural view of the AES core is shown in Figure 4.5. The *Enc_Dec* signal of the AES core changes the encryption and decryption mode of the core. *FSL_Clk* is the clock signal interface for the FSL bus clock. Key size is controlled in the AES core by the value of the *Size_key* signal. Depending upon whether the *Size_keysignal* values are 00, 01 or 10, the core will perform 128, 192 or 256 bit encryption or decryption respectively and 11 is regarded as don't care. The *Enc_Dec* signal determines (1 for encryption and 0 for decryption) whether the circuit will perform encryption or decryption, i.e. the mode of operation. The *data_load* signal (treated as a valid data when the signal value is 1) states the data is valid and needs to be loaded for encryption

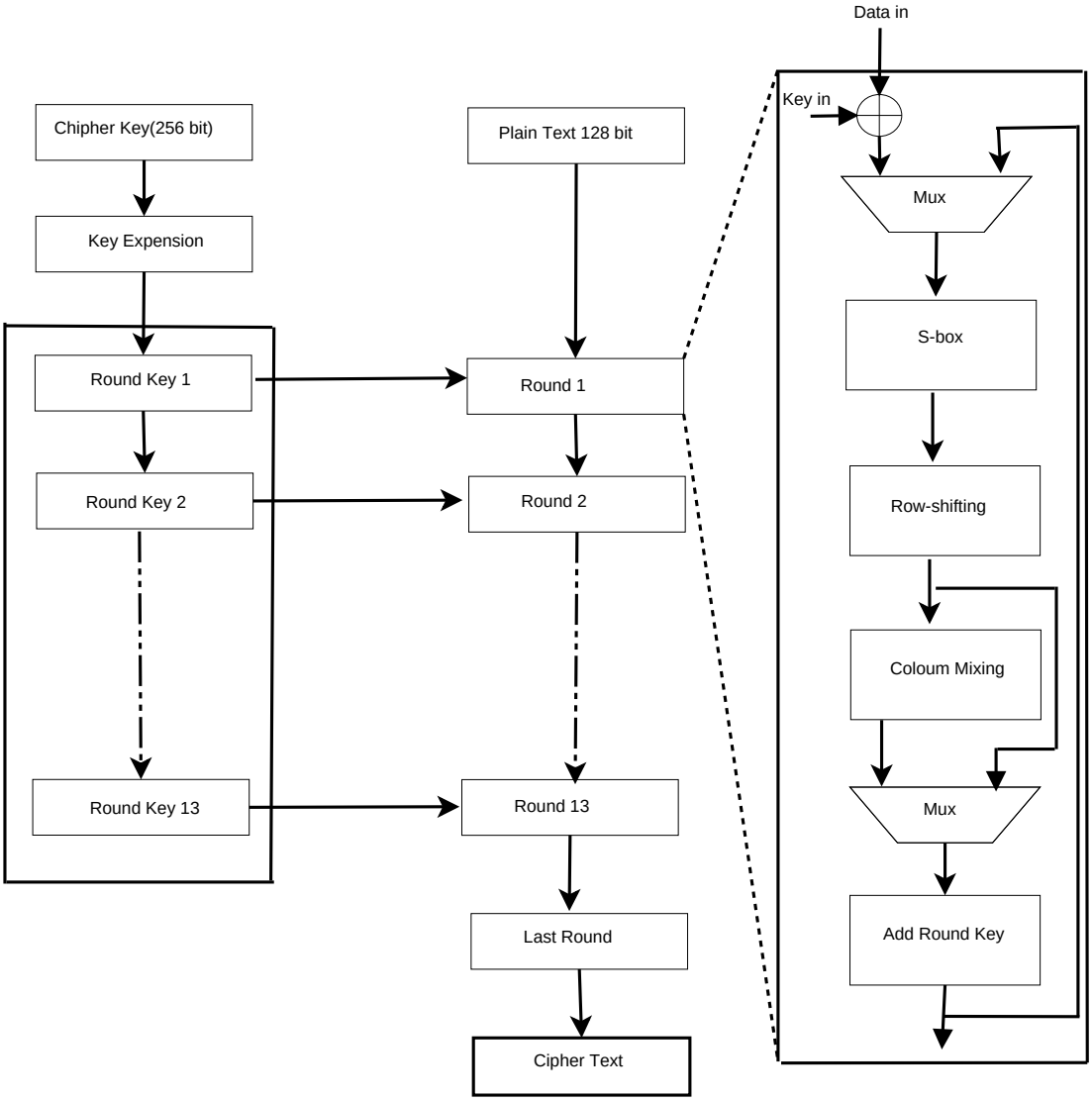

Figure 4.3: AES 256 encryption logic flow diagram

/decryption. The *key_load* (1 for valid) in the *KeySchedule* module, denotes that the key is valid and needs to be loaded. The *KS_Start* (1 for start) and the *Alg_Start* (1 for start) signals are controlled by the controller block and depending upon these two signal values the *key schedule* and the *AES algorithm* block will start functioning. The *Done* signal, when 1, denotes that the process is completed and the result is available at the *Data Out*. The *FSL_clk* is the clock signal for the AES core. Figure 4.3 shows the AES encryption logic flow diagram[105].

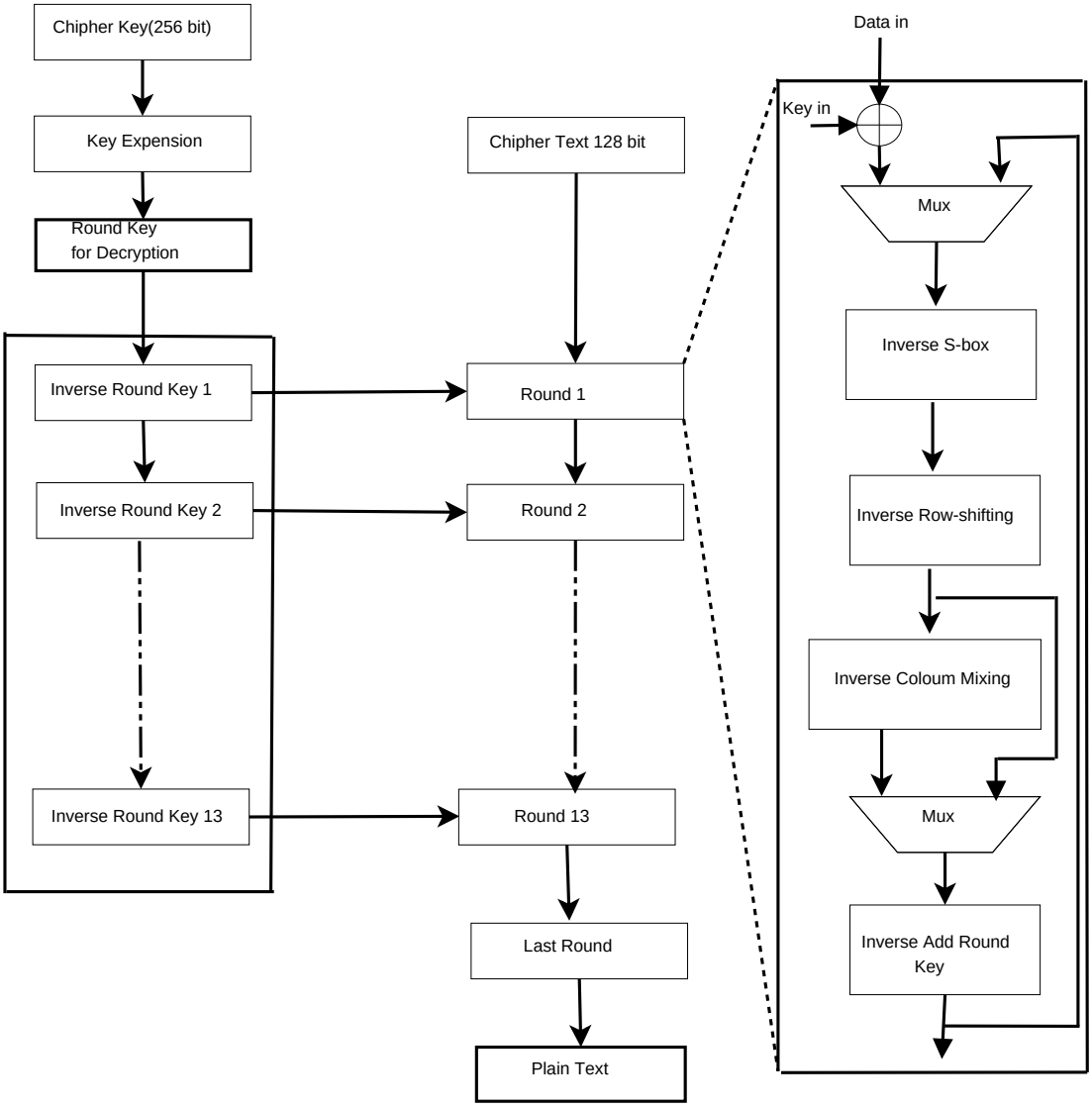

Figure 4.4: Decryption logic flow of the AES algorithm

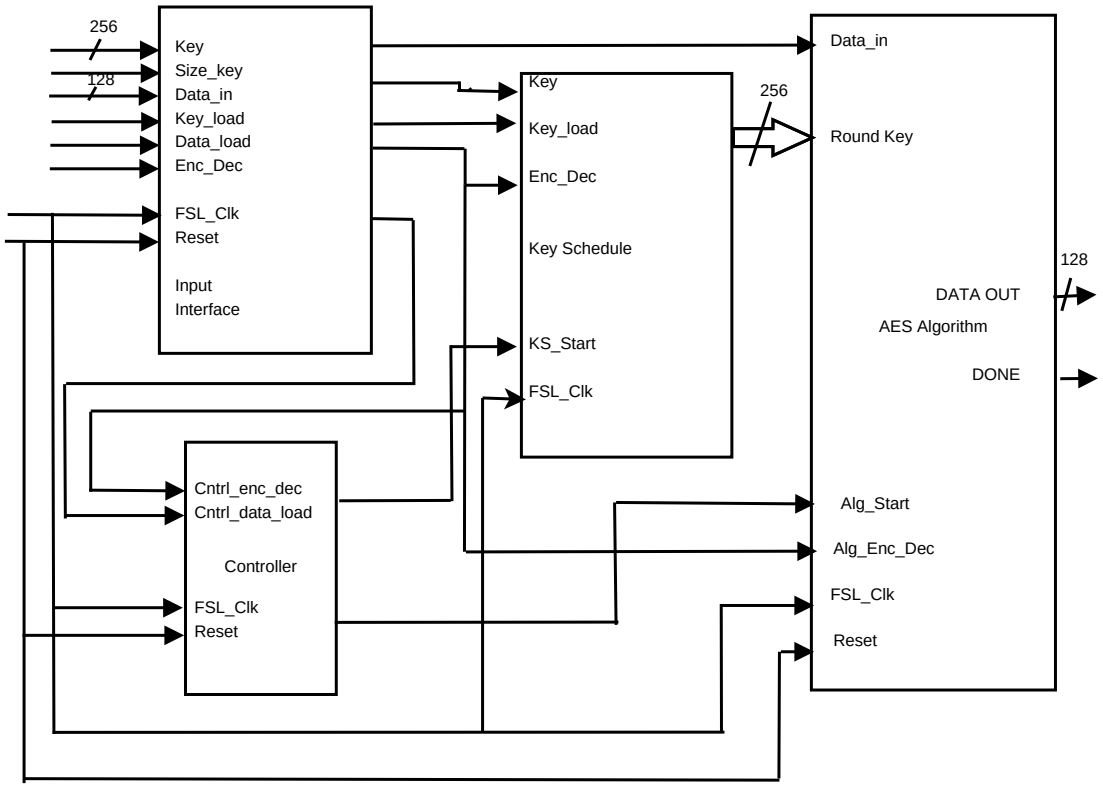

Figure 4.5: Internal architecture of the AES Core

There are 14 rounds and one key expansion rounds in AES-256.So for the encryption (AES-256) process, 15 cycles are taken.The other clock requirements are: one clock each for *key_load*, *data_load* and *done* signal. As FSL bus is of 32 bit wide, 32-bit data can be read or written in the FSL bus in one clock cycle. So, for transferring 128 bit data, 256 bit key, and the generated 128 bit cipher data between the Microblaze and the AES core, we require $(128 + 256 + 128)/32 = 16$ clock pulses. So in total 18 clock pulses for encryption and 16 clock pulses for data transfer are required. In the case of decryption it will take more clock pulses as we need to generate the first round of key. For an implementation having a fixed key value we will have 4 less clock cycles as there will be no key transfer from the Microblaze processor core, and it can be set within the AES core itself.

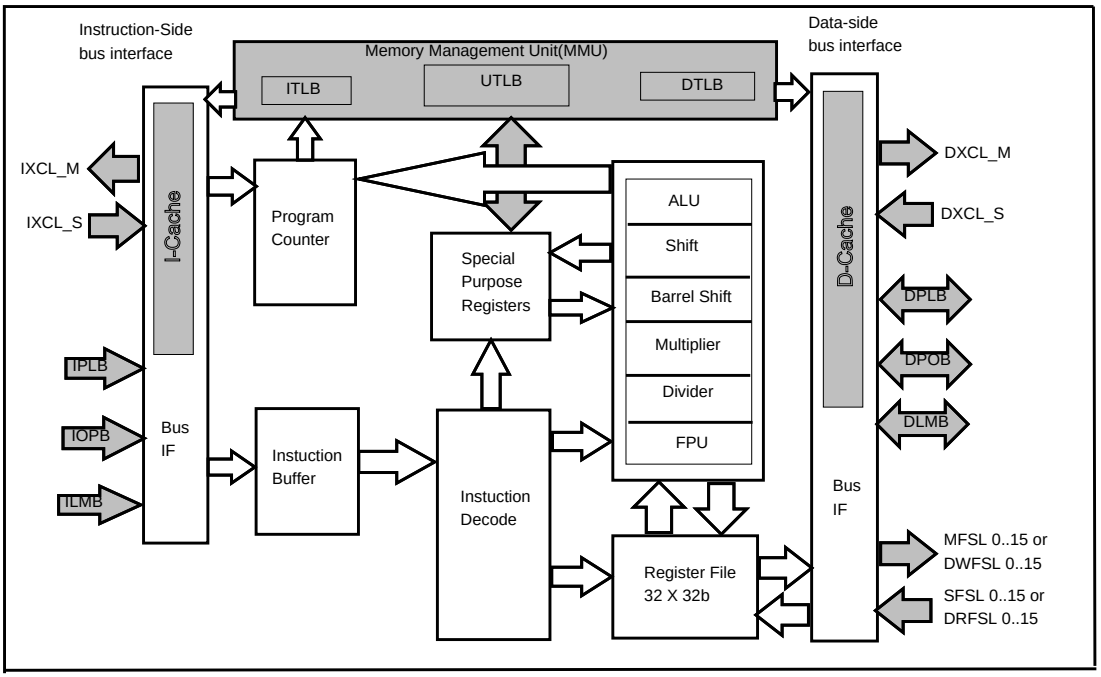

Figure 4.6: Internal architecture of Microblaze processor core [1]

4.3.2 Implementation of the Microblaze Soft Core Processor

Microblaze has been used as a soft core processing unit in our design and its internal functional blocks are shown in Figure 4.6. The other important soft core peripherals needed to build the system are: UART for RS232 DCE port, LEDs as GPIO , EMAC (10/100/1000 Mbps) for Ethernet connectivity and dynamic random-access memory (DRAM) for catering the memory requirements.

A co-processor can be added to the system in three different ways, it can be added to the system as a peripheral to the processor core using the PLB or with the OPB [52] or can also be directly attached to the processing unit through the FSL bus interface. Figure 4.2 shows the proposed design, where the AES co-processor core is attached with the Microblaze processor core through the FSL bus and the other required peripherals are connected with the Microblaze processor core through the PLB . The Microbalze soft core processor has a dedicated FSL interface, which provides a high speed link to add any user design customized core. We can use more than two dynamic inputs and one output in the core

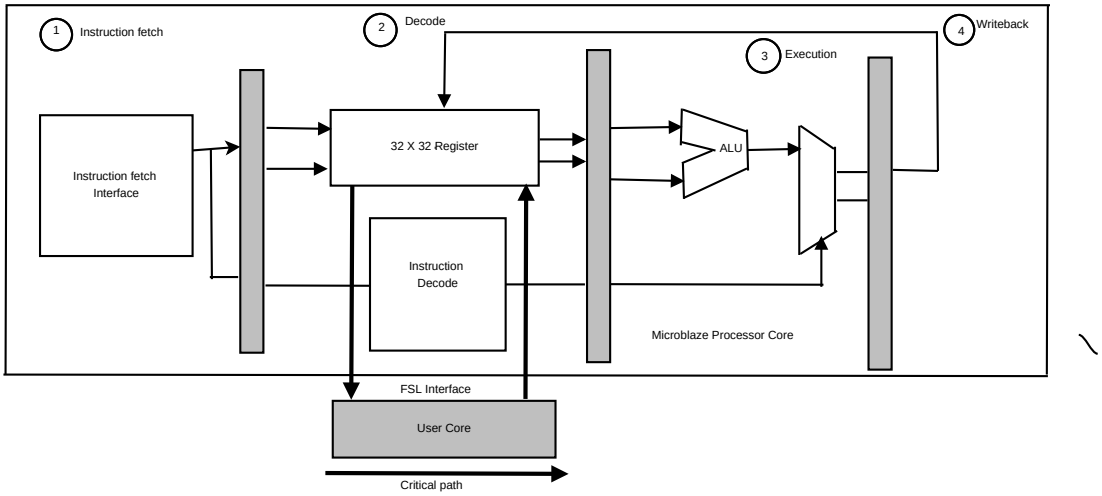

Figure 4.7: User core connected with the Microblaze processor through the FSL link [2]

design, as Microblaze soft core provides sixteen FSL interface buses [1]. Out of the sixteen buses, eight can be used as inputs to the user core and the other eight for the outputs. Figure 4.7 describes how to connect a user defined core to the Microblaze processor core as a co-processor using the FSL bus. There is also a scope of connecting external signals directly to the co-processor core, if required. For sending parameters to the hardware unit and to receive the results, the FSL core uses predefined C-macros. The AES algorithm is written in VHDL, which describes the hardware for the AES engine. The Figure 4.8 shows the communication interface between the co-processor and the Microblaze processor core through the FSL bus and the needed data buffering. In Figure 4.9, we show the different ways of connecting the co-processor with the Microblaze processor core, either through the FSL bus or using the PLB. The dotted line in Figure 4.9 indicates the block, which can be added to the PLB as a peripheral.

4.3.3 FSL Bus And AES Hardware Engine

The FSL interface on the Microblaze processor core supports control and data communication port, and it is a FIFO based communication[2]. Figure 4.10 shows the FSL bus signals. Stream data can be passed through the FIFO as per

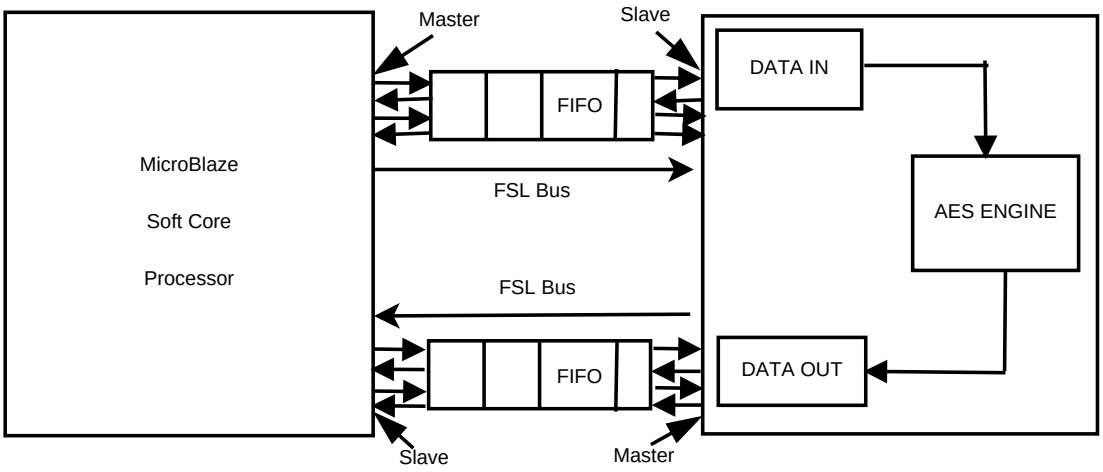

Figure 4.8: FSL bus communication between the AES co-processor and the Microblaze

the clock rate. Generally depth of FIFO is chosen of the same length as that of the data vector. If the input data size is higher than the maximum depth size of the FIFO, we need to compromise with the speed for data processing. The bus protocol is straight forward and involves a few signals which are shown in Figure 4.10. The FSL_M_Write signal controls the write operation on the FSL bus, the FSL_M_Write signal will go high (set to 1) when FSL_M_Data and $FSL\_M\_Control$ are ready to write data on FIFO, the FSL_M_Full signal indicates whether the FIFO is full or not. The read in FSL bus is controlled by the FSL_S_Read signal and the FSL_S_Exists signal will go high if there exists data in the FIFO. The data in the FSL_S_Data and control bit in $FSL\_S\_Control$ signal is immediately available on the FSL bus.

4.4 System Evaluation

The required hardware architecture in Figure 4.2 is successfully implemented using *Xilinx EDK 11.1* [54]. *Virtex5LX110t-ft1136* board as shown in Figure 4.11 was used for the implementation of the design.

The AES core runs at the processor core speed of 125 MHz. The custom hardware based AES-128 implementation described in [3] takes much more clock cycles (93) for only to complete the encryption process alone, as compared to the

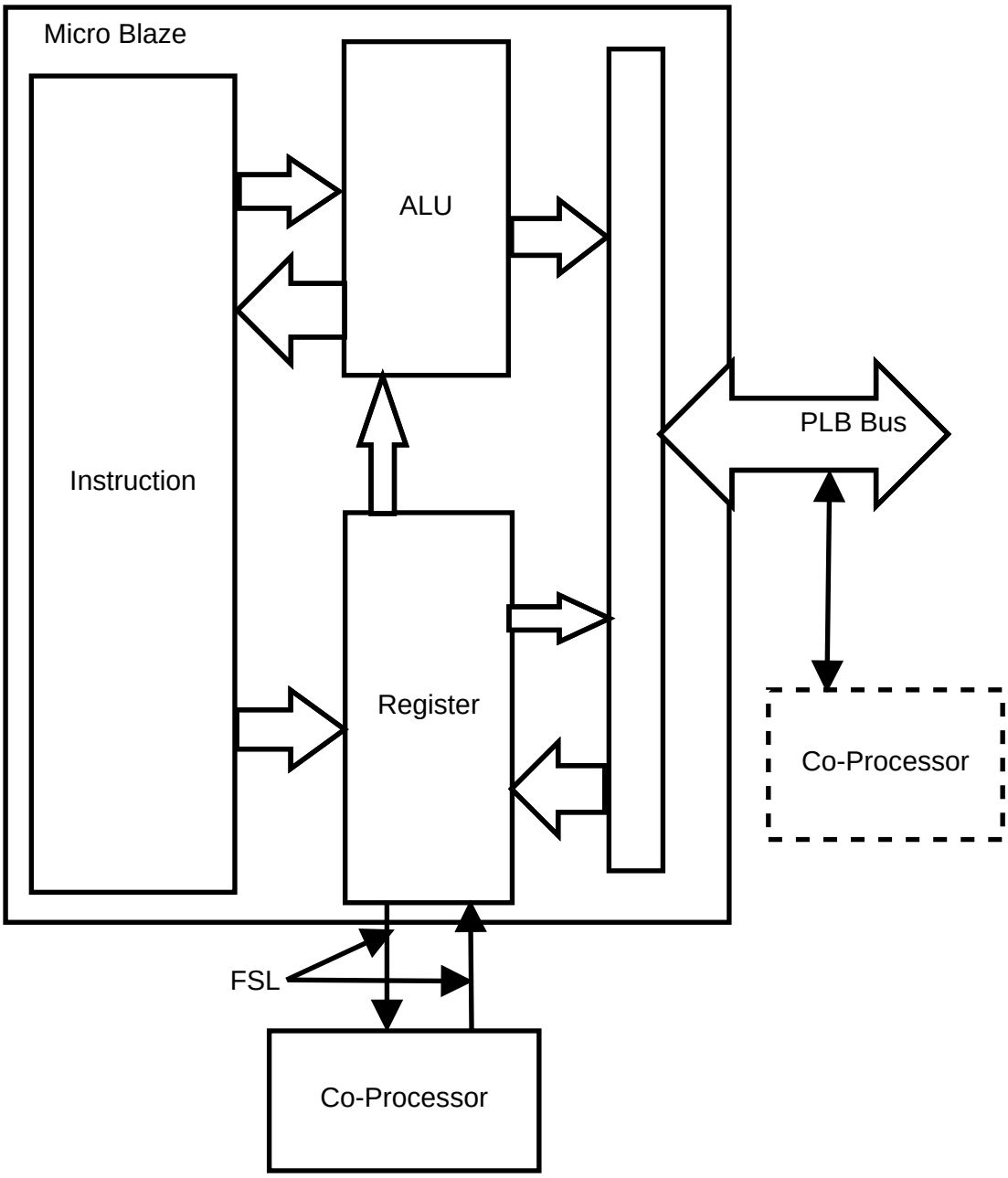

Figure 4.9: Different ways of attaching a co-processor with the Microblaze

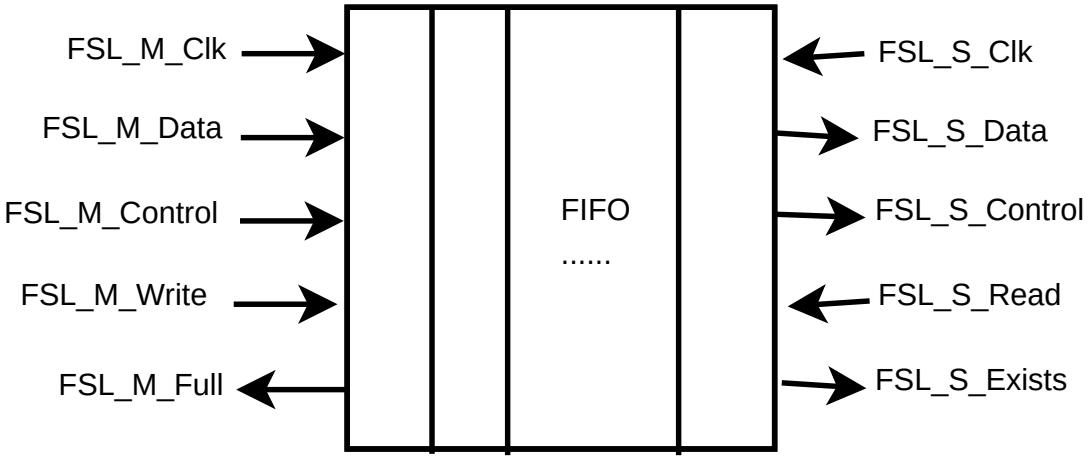

Figure 4.10: FSL bus signals [2]

Figure 4.11: Xilinx Virtex-5 LX110t-ft1136 board picture

our work. For the variability in the basic hardware infrastructure in the available research works, beside throughput we have also compared the parameters like Mbps/Slices, Mbps/MHz. We have implemented the AES with 256 bit key size for providing more randomness in the key and hence better security of our data, which seems to be an improvement over the key size compared to most of the existing works. The throughput for the encryption and decryption processes of the designed AES core are 1066.67 Mbps and 533.33 Mbps respectively, which are better compared to the previous implementations [3], [55]. Design proposed in [106] achieves higher throughput (though it is based on AES-128) with respect to our design, but it takes much more resources than ours. The throughput measurement is directly proportional to the system clock, so there remains the scope of increasing the throughput of the design if we can have a higher frequency clock source. The implementation in [106] is not based on a co-processor architecture, hence the design is better suited whenever there is a need of multi tasking specially in the real-time application scenario. In a nutshell the design is one of the most advanced implementations of the AES-256 algorithm, having high throughput and utilizing small amount of resources. Comparison with the existing implementations in the related works was difficult, as the performance estimation for the same FPGA board as that of ours were not available. But still we have found that our design and implementation achieves higher throughput and lower utilization of resources related to the other previous works. **For the fair comparison between our work and existing work, calculate the design throughput and efficiency as per the equation** 4.3 and 4.4, given below.

$$Throughput = \frac{Numberofprocessedoutputbit}{Numberofprocessedoutputbit} \tag{4.3}$$

$$Efficiency = \frac{Throughput}{Totalresourceused} \tag{4.4}$$

Here we calculate the critical path delay on the basis of the number of clock cycles which is taken to complete the function (here it is AES), so that the comparison of throughput is board independent. Because the number of clock cycles needed for complete any functions

will be the same irrespective of the boards family and frequency. The performance of the design in comparison to the other existing works is illustrated in Table 4.1 the best performance values are made bold-faced.

Table 4.1: Comparison with other AES implementations

Work-device	**FPGA resources (Slices,BRAM)**	**Clock frequency (MHz)**	**Throughput (Gbps)**	**Efficiency (Mbps/Slices)**	**Efficiency (Mbps/MHz)**
Spartan-3 [107]	523	63.70	0.127	0.243	1.993
VirtexE [108]	3750	50.00	0.243	0.064	4.860
Virtex-II[109]	3474,15 BRAM	80.30	0.275	0.073	3.424
Stratix [110]	5605 logic cells	50.00	0.285	0.052 Gbps/ logic cells	5.700
ASIC module[111]	-	-	0.054	-	-
Processor[112]	-	66.00	0.275	-	4.166
Processor[113]	-	1500.00	>2	-	1.333
Virtex4-LX[106]	1921,20 BRAM	149.00	1.484	0.992	12.590
Virtex5lx110t *[our design]*	**1774**	**125.00**	**1.067**	**0.601**	**8.533**

4.5 Summary

The performance of the implemented design, in terms of throughput together with the resource utilization, is proved to be the best in comparison with the other existing works. The proposed design uses concurrent processing over multiple cores involving Microblaze processor core and a custom AES-256 co-processor core, which makes it a very suitable system for real-time applications. The design have also been verified over a real network, which proves that it is a ready to use design. The work briefed in this chapter creates a perfect platform for setting up a real-time data communication infrastructure, which can use the AES blocks both in the transmitter (encryption) side and the receiver side (decryption) as described in this chapter. The following chapter will describe our proposed design for setting up a high data rate communication link in an embedded system environment, which will be further augmented with AES blocks and error correction schemes in the next subsequent chapters.

Chapter 5

High Speed Secure Communication System Design using Error Detection and Correction Model

Many present day applications require high throughput real time data acquisition systems which transfer the data from a large number of front-end sensors to a back-end storage block for further processing. The facilities created for high energy physics experiments like (CBM-FAIR) require such a high speed communication system for experimental data collection. These type of real-time data acquisition systems are also required in applications like security and surveillance, unmanned vehicular control systems, robotic applications etc. These applications require data acquisition (DAQ) read out chain, which satisfy a very precise time synchronization, compact hardware, radiation tolerance, self-triggered front-end electronics, efficient data aggregation schemes and capability to handle high data rate (up to several TB/s). As a part of the implementation of read out chain of these experiments [114] in India, we have tried to implement FPGA based emulator of Gigabit transmitter (GBTx) in India. GBTx is a radiation tolerant ASIC that can be used to implement multipurpose high speed bidirectional optical links for HEP experiments. GBTx will be used in highly irradiated area and more prone to be affected by multi bit error. In order to mitigate this effect instead of single bit error correcting RS code, we have used two bit error correcting

(15, 7) BCH code. It will increase the redundancy, which in turn increases the reliability of the coded data. So, the data errors occurred by the radiation noises, can be efficiently corrected using BCH code up to a certain limit. The data will flow from detector to the computer based storage through multiple nodes through the communication channel. The computing resources are connected to a network which can be accessed by authorized person to prevent unauthorized data access which might happen by compromising the network security. Thus data encryption is essential. In order to make the data communication secure, hence AES [6] and RSA [115], [8] are used after the channel coding. We have implemented GBTx emulator on two Xilinx Kintex-7 boards (KC705). One will act as transmitter and other will act as receiver and they are connected through optical fiber through SFP port. We have tested the setup in the runtime environment using Xilinx Chipscope Pro Analyzer. We also measured the resource utilization, throughput, power optimization of implemented design.

5.1 Introduction

High speed and fault resilient DAQ system is an integral part of the signal processing unit in real time applications like HEP, satellite communication etc. Traditional DAQ system front end electronics (FEE) board captures data from the sensors through high speed LVDS link, processes it and sends it to storage device using high speed link like Ethernet, PCIe, fiber optic etc. In [63], a slow data rate (1.6 Gbps) DAQ architecture is described. In [66], authors have shown a high speed (8.5 Gbps) optical communication between two ALTERA Stratix IV FPGA boards. Here, PCIe buses are used to transfer data between board and computing node, but no error correction or cryptographic mechanism is used. In the Beijing Spectrometer III (BESIII) trigger system, a high speed data transmission protocol over optical fiber for real time data acquisition was developed by Hao Xu *et.al* in [68]. This system used MGT of Virtex-II Pro series FPGA for optical fiber communication (1.75 Gbps). In [69], [70] a high speed data transfer protocol was implemented using different FPGAs, which achieved highest data rate of 2.5 Gbps and 784 Mbps respectively.

SEU and MBU occur when a charged particle hits and transfers sufficient

energy to the silicon area of a circuit. Two types of approach are taken for the SEU and MBU mitigation: prevention and recovery. Prevention methods are mainly considered during the ASIC design. In recovery methods different online recovery mechanisms *e.g.* fault tolerant computing, error detecting/correcting code and online testing are used. The concurrent error detection [76] is one of such techniques where an extra error detection circuit is attached to the main circuit that simply recomputes the whole operation from the beginning when an error is detected. In TMR [116] the same functional replica is used thrice and final result is taken with the majority voting system.

In the above mentioned papers the authors did not propose any idea to handle the SEU mitigation in high speed data acquisition system that are used in an adverse environmental condition as found in HEP experiments [4] [16]. For data communication among different nodes secure transmission is also an issue. In real experimental environment [4] the data communication requirements are expected to be as follows:

- # channels >100k
- read out frequency > 100 KHz
- synchronization limit <100 ps
- data capacity >1 Tb/s

In our research, we proposed a high speed data acquisition system design with secure communication using gigabit transceiver (GBT) [117] emulator, which is protected from SEU by multi-bit error correcting BCH code [80] and interleaver. Scrambling is used as the line coding technique to maintain the DC balance and to obtain 20% extra throughput as compared to 8b/10b coding. In this chapter, the contributions are summarized as: [1] in [65], [66], [67]. We have achieved maximum data rate of 4.8 Gbps compared to 1.6 Gbps, 2.125 Gbps, 1.75 Gbps in [65], [67], [68] respectively. In this section, our key contributions are:

[1] Outcomes of this chapter was communicated in VLSID Design Track 2015, ISVLSI 2015, SPIE 2015, IOP Journal, IEEE NSS 2016, IEEE Transaction on Nuclear Physics (C8, C9, C10, C12, J1, T1)

- Efficient implementation of high speed secure DAQ with optical link using FPGA based GBT emulator.
- Enhancing the DAQ system with error correction capability for SEU / MBU types of error using multi-bit error correction technique.
- Implementing cryptographic algorithms in hardware level to ensure secure backend storage system communication.
- Interfacing the DAQ with back end storage computing node through PCIe (2nd generation, 8 lanes) and scatter gather direct memory access (SGDMA) .

5.2 High speed DAQ design with secure communication for MUCH experiments

High data rate, error correction capabilities, secure communication and efficient storage mechanism remain the main criteria of DAQ design for Muon Chamber (MUCH) experiments for future analysis [114]. In our system for high data transmission, optical fiber is used as the communication media. Several multi-bit error correction methods for efficient communication have been discussed in Section 5.1, where BCH coding is the most suitable for random error correction. The interleaver block has been introduced after the encoder block to enhance the error correction efficiency. For ensuring secure communication, different cryptographic algorithms can be used. We have chosen RSA and AES algorithms which are discussed in section 5.2.5 for testing our prototype. In the receiver side data is directly transfered to back end system through PCIe from the FPGA board. Functional blocks of the proposed system is shown in Figure 5.1. The details of each block are discussed in the following subsections.

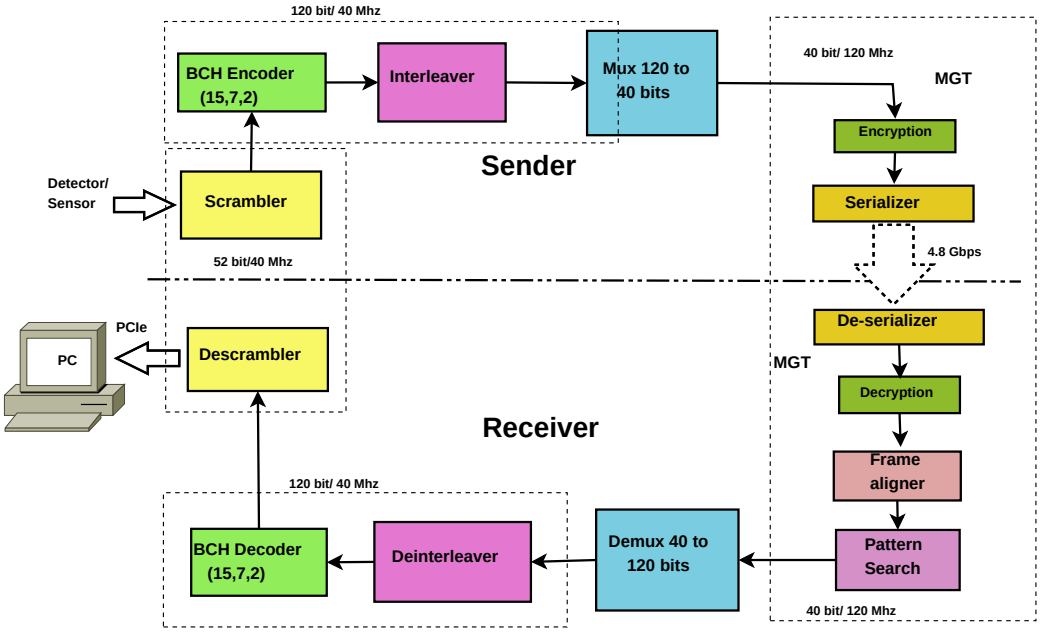

Figure 5.1: Internal blocks of the proposed system

5.2.1 Scrambler/Descrambler

Scrambler [118] helps to maintain the DC balance in the input signal coming from the detector/sensor and also helps in accurate timing recovery on the receiver side. This reduces occurrence of long sequences of '1' (or '0') in the input data stream. The scrambler does not add any overhead in the system like the $8b/10b$ or $7b/8b$ line coding [119] except a latency of one clock cycle. In our system, 52 bit incoming data is divided into four blocks of 13 bit data and each block is scrambled simultaneously.

5.2.2 BCH Encoder/Decoder

5.2.2.1 BCH Coding Theory

Now, we brief the basic theory of BCH coding. BCH code is presented by BCH(N,K,t) where N is the length of codeword, that means it is the sum of message length(K) and parity bits(p), t is the number bits for error correcting code. Construction of BCH code is given as below:
Primitive element of GF(2^m): α

Block length: $N = 2^m$ - 1,
Parity check bits ≤ mt,
Generator polynomial $= g(X)$,
Let $\lambda_i(X)$ the minimal polynomial of α^i then $g(X)$ can be represented as:

$$g(X) = LCM\{\lambda_1(X), \lambda_2(X), \lambda_{2t}(X)\} \tag{5.1}$$

for primitive element α and its polynomial and vector representation in GF(2^4).

Message polynomial :

$u(X) = u_{N-K}X^{N-K} + u_{N-K-1}X^{N-K-1} + u_1X + u_0$

Generator polynomial :

$g(X) = u_{K-1}X^{K-1} + u_{K-2}X^{K-2} + u_1X + u_0$

Encoding:

$r(X) = X^{N-K}u(X) \quad mod \quad g(X)$

$\Rightarrow c(X) = X^{N-K}u(X) + r(X)$ where minimum distance $d_{min} \geq 2t + 1$. Using an example the encoding and decoding theory of BCH is discussed below. Let us select BCH(15,7,2) with (GF(2^4) and $(X^4 + X + 1)$
Message :(0110011) $\rightarrow u(X) = X^6 + X^5 + X^2 + X$ with error correcting bit value $t = 2$

$$\begin{aligned} g(X) &= LCM\{\lambda_1(X), \lambda_3(X)\} \\ &= \left(X^4 + X + 1\right) \times \left(X^4 + X^3 + X^2 + X + 1\right) \\ &= X^8 + X^7 + X^6 + X^4 + 1 \end{aligned} \tag{5.2}$$

Now, $X^{15-7}u(X) = X^{14} + X^{13} + X^{10} + X^9$,
and $r(X) = X^8u(X) \, mod \, g(X) = X^3 + 1$

$$\begin{aligned} c(X) &= X^8u(X) + r(X) \\ &= X^{14} + X^{13} + X^{10} + X^9 + X^3 + 1 \end{aligned} \tag{5.3}$$

So, from equation 5.3 binary value of the codeword after encoding is (100100000110011).

For decoding, the steps are given below.

1. Compute syndromes

2. Determine the error locater polynomial ($\sigma(X)$)

3. Find error location (Chien search [80])

4. Decode the correct result

5.2.2.2 Encoder/Decoder Hardware Block

In this design block a binary error correcting code BCH (15,7,2) code is used to correct the error due to SEU or MBU. Here (15, 7, 2) means, 7 bit of input data, 8 redundant bit, which are appended with the input data and it can correct up to two bits of error. So the code rate (ratio of input data to coded data) is 0.467. In the proposed DAQ, 56 bits of data (52 bit data with 4 bit header) are broken down into eight 7 bits of data, which are encoded with BCH encoder in parallel. After encoding the 15 bits of data from the eight blocks, they are assembled to generate a total 120 bits of data. This block increases the reliability in data transfer but adds one clock cycle latency in the system. The coded data is decoded in the following three steps: determination of the error locater polynomial, detection of error location using Chien Search Algorithm [80] and location of the data at the error position. One can find the details of BCH algorithm in [80]. In our present work we have designed the BCH encoding/decoding block as a custom hardware. Instead of selecting BCH code with larger block size like (31, 26, 1) or (63, 57, 1), we have used eight BCH (15,7,2) in parallel for faster error correction without compromising the time complexity. Hence, each BCH encoder block can correct up to 2 bit of error within 7 bit of input. So the total $8 \times 2 = 16$ bits can be corrected using this technique without any extra resources. Similarly, we can also use triple error correcting BCH code [80] but that will reduce the code rate.

5.2.3 Interleaver/De-interleaver

In order to reduce the effect of burst error in the consecutive bytes of data in transmission process, the interleaving block [120] is used. In general, there are two types of interleaving strategies in communication system, block interleaver and convolutional interleaver. Here we have used block interleaver. Output of 120 bit data from encoder is divided into two blocks of 60 bit data and then interleaving operation is done on each of the 60 bit data using block interleaver. This block

increases the code correction capabilities by dispersing the error without any clock latency and overhead. De-interleaving process is used to reorder the data again in the receiver side.

5.2.4 MUX/DEMUX and Clock Domain Crossing

This block consists of a dual port RAM and a read-write controller. It breaks down 120 bit frame into three words of 40 bit width and helps to maintain the data rate same throughout the system. Here, we have used 120 MHz clock to drive the MGT available from Xilinx IP core to keep the data rate same with the internal blocks those are running with 40 MHz frequency. The data rate and clock frequency can be changed according to the data communication requirement. This block is used to synchronize the data rate between MGT and the other parts of the design. Figure 5.2 shows the architectural block diagram of the MUX-DEMUX and clock domain crossing.

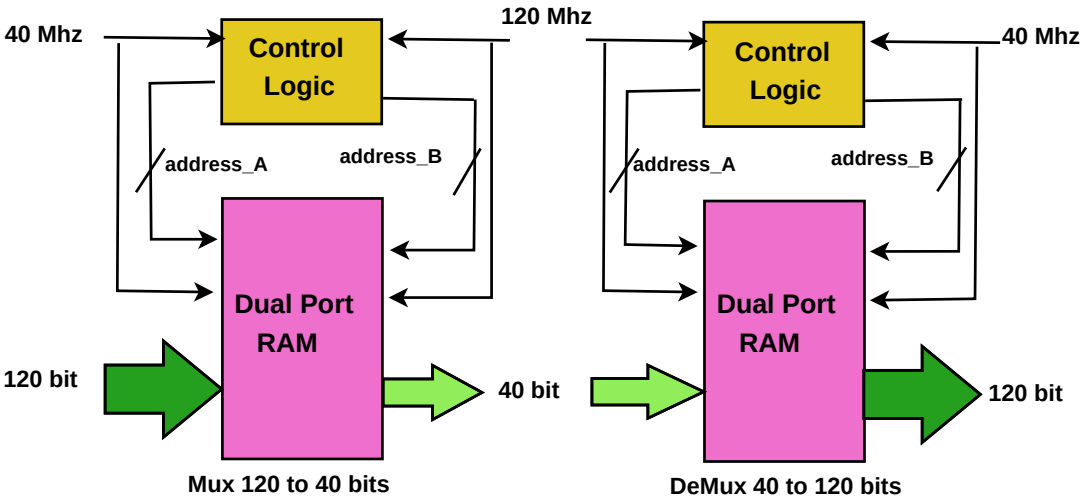

Figure 5.2: Mux DeMux for clock domain crossing

5.2.5 Encryption/Decryption

Encryption block is used to cipher the data, which will be transmitted over the channel. Different cryptographic algorithms may be used for this purpose. In our design, we tested the system with two different algorithms AES [6] and RSA [115] . AES is a symmetric key cryptographic algorithm. The key size of AES

algorithm can vary from 128,192 and 256 bits with fixed data size of 128 bits. Depending on the key size of the encryption and decryption operations, rounds may vary within 10,12 and 14 respectively. Whereas RSA works on asymmetric key based cryptographic algorithm. Two keys are used namely public and private key. Public key is used to encrypt the data whereas the private key is used for decryption of the cipher data. Decryption in the receiver side is used for deciphering the received data. Figure 4.3 and Figure 4.4 shows the AES-256 encryption and decryption flow respectively in Chapter 4.

5.2.6 Serializer/De-serializer

A serializer is used to convert the parallel data to serial data, which is transmitted over the communication channel. Similarly, a de-serializer simply converts the serial data to parallel data in the receiver side [121].

5.2.7 Frame Aligner and Pattern Search

Frame aligner block is used in the receiver side for aligning and indexing the frames in a proper order. Pattern search algorithm is used to detect the frame header. Figure 5.3 shows the flow chart of this block. Two types of frames are considered in this design:(i) standard frame and (ii) frame without FEC . Standard frame consists of four fields: Header field (4 bit), Slow Control (4 bit), Data field of width 48 bit, FEC field of width 64 bit. Whereas, frame format without FEC consists of all the fields of the standard frame except the FEC field. Header value of the standard frame format is 1010. The frame aligner and pattern search blocks consist of two sub blocks (Pattern search block, Right shift block) as shown in the Figure 5.4.

The pattern search block is used to check whether the header field is properly received or not. It will be stable after a continuous search for another 32 headers of subsequent frames, after the first frame header is received.

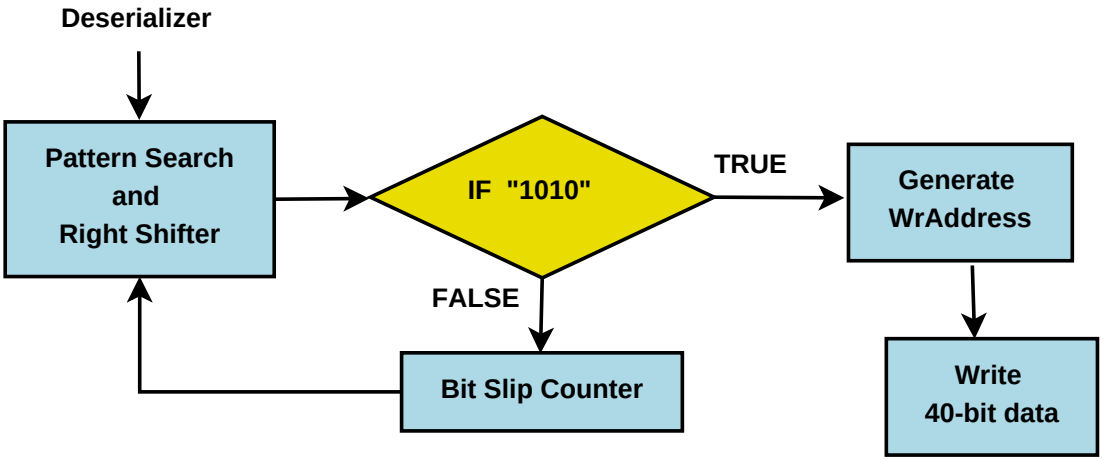

Figure 5.3: Algorithm for Frame Aligner and Pattern Search

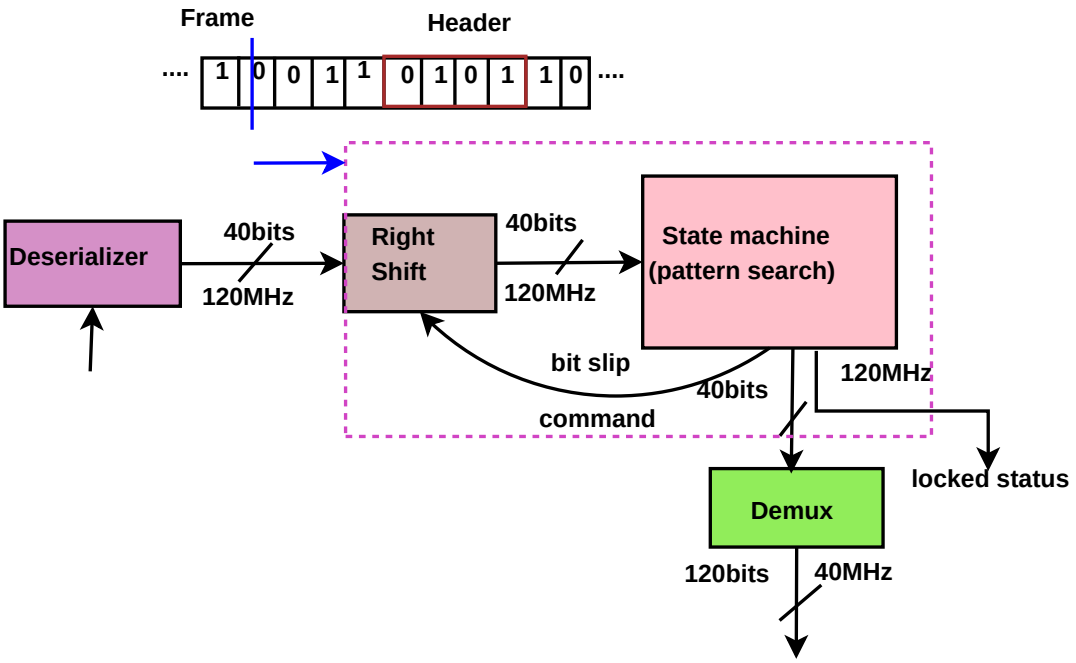

Figure 5.4: Data flow diagrams of the Frame Aligner and Pattern Search block

5.2.8 Data Transfer to back end system through PCIe

Transfer of data from FPGA board to back end system is done by PCIe with the help of asynchronous FIFO and SGDMA. We have used PCIe gen2 IP core available from Xilinx. Interconnection of FPGA to PC through PCIe is shown in Figure 5.5. Data is written into FIFO at a frequency of 120 MHz and data will

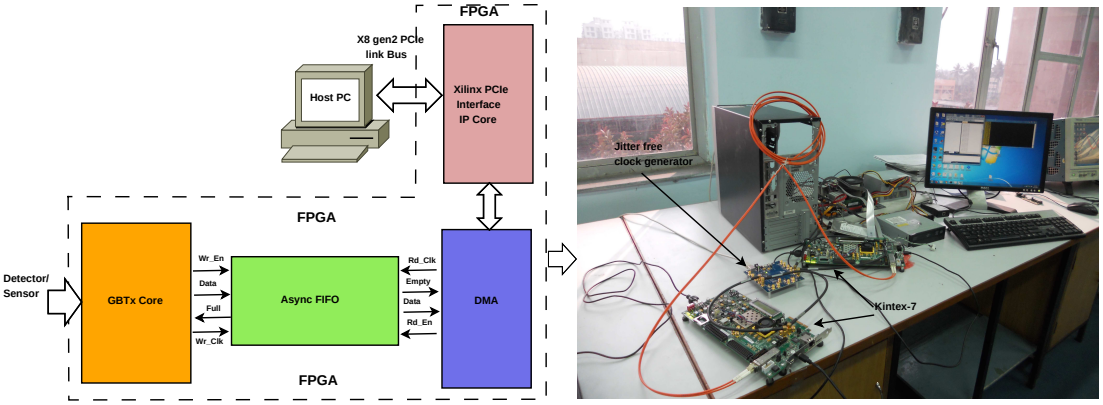

Figure 5.5: PCIe interfacing with blocks and Experimental setup of proposed DAQ

be read from FIFO at a frequency of 125 MHz by which PCIe core is running. A software is written using windows SDK and C language for further processing of data in the PC.

5.2.9 Overview of Secure Data Flow

The complete chain of the functional blocks as shown in Figure 5.1 for the high speed secure DAQ with multi-bit error correction (considering two bits of error correction) has been implemented on the FPGA board. Figure 5.6 shows the complete mechanism of the standard frame generation and the error correction flow with encryption/decryption mechanism.

52 bits of data received from the detector is divided into four 13 bit blocks of data and each block is scrambled in parallel. The scrambled data with the 4 bit header is mapped in the input line of the eight BCH (15,7,2) encoders, which are running in parallel. Output of all the encoders are combined to get a frame of 120 bit data. These 120 bits of data first are interleaved and then passed to the next functional block that is the MUX. Interleaving is used to reduce the effect of burst error [120]. But the header position is never changed in the frame format (red color in Figure 5.6) even after the interleaving process, which helps to synchronize the frame in the receiver side. An encryption block is added after interleaving process for secure data communication in the channel. In order to prevent the unauthorized access in the network, encryption is used after

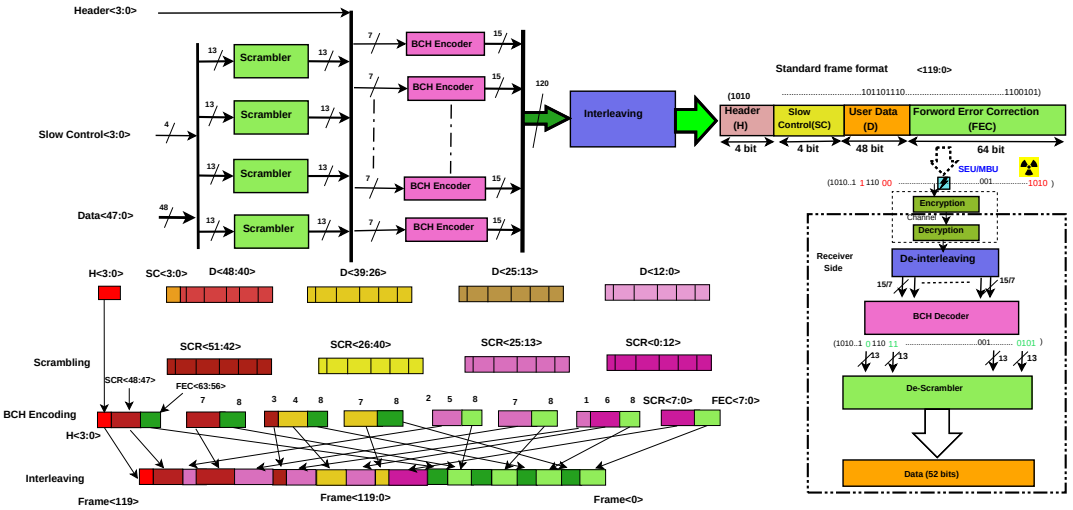

Figure 5.6: Standard frame generation and Error correction flow with encryption/decryption

the FEC block. If we use encryption before FEC then there also is a chance of compromising the Header or the Slow Control bits.Here, we use RSA and AES-128 algorithms for the testing of our design. Details of the resource utilization of this block is described in Section 5.3. In Mux-DeMux and clock domain crossing block, a dual port random-access memory (RAM) is used to write 120 bits of data using 40 MHz clock and read the same data at 120 MHz clock rate with 40 bit word size. So the data rate for writing (40×120 = 4.8 Gbps) and for reading (120×40 = 4.8 Gbps) is same. The 40 bit data is first serialized and then passed to the transmitter (TX) for communication over the optical fiber cable. In the receiver (RX) side functional blocks are Deserializer, DEMUX, Decryption, De-interleaver, BCH Decoder (15, 7, 2) and Descrambler. They perform reverse function with respect to Serializer, MUX, Encryption, Interleaver, BCH Encoder (15, 7, 2) and Scrambler respectively. The frame aligner and pattern search in the receiver side of this chain whose functional description have been described in Section 5.2.7.

5.3 Experimental Setup and performance Analysis

The full prototype of the secure communication chain is implemented in the Xilinx Kintex-7 boards (KC705 from Avnet) using the Xilinx ISE 14.5 platform and VHDL for design entry. We have used an external jitter cleaned clock source (CDCE62005EVM of TI) to drive MGT of two Kintex-7 boards. One Agilent DC power supply has been used to drive the whole system. Two Kintex-7 boards are connected through single mode optical fiber using SFP from *Finisar* (FTLX8571D3BCL). For board to PC communication we have used eight lane PCIe gen2.

The block diagram and the experimental setup of the system are shown in Figure 5.5. We achieved maximum bit rate of 4.8 Gbps in our system. In standard mode, a frame contains only 52 bits of data, 64 bits for error correction (FEC) and 4 bits of header. 64 bits for FEC can correct up to 16 bits of error, as it is applied on 8 encoder blocks in parallel (2 bit error correction for each block). So the data rate achieved considering only the data field (D) in this mode is:
$40MHz \times 52bit = 2.08Gbps$
In frame format without FEC, where error correction code is not used, the frame can carry $(52 + 64 = 116)$ bit of data out of 120 bit frame. So in this mode data rate is measured:
$40MHz \times 116bit = 4.64Gbps$
Hence, the data transfer efficiency for the above mentioned two modes are $(2.08/4.80) \times 100 = 43.33\%$ and $(4.64/4.80 = 96.6\%$ respectively. Our system gives better performance in comparison with the system described in [66], [68], [69], [70]. Resource utilization ans power consumption of for each of the functional blocks of the system is given in Table 5.1 and Table 5.2.

In Figure 5.7 we show the critical time, which is the maximum delay time to get the output of a circuit for each of the circuit blocks. Power consumption is estimated using Xilinx Xpower tool and we show the estimated average logic and signal power for the various model of the proposed design. To the best of our knowledge, we are reporting the critical time and power consumption of this type of system for the first time.

Table 5.1: Resource Utilization

Board	Module Name	Slice Register	Slice LUTs	LUT Flip Flop	BRAM
Kintex 7- 325t	BCH Encoder (15,7,2)	7//407600	951/203800	0	7/951
	BCH Decoder (15,7,2)	135/407600	446/203800	0	119/462 (25%)
	Scrambler	52	53	5	0
	Descrambler	104	56	5	0
	Interleaver	44	40	40	0
	DeInterleaver	201	82	80	0
	Frame Aligner	115	308	72	0
	Encryption (AES)	1311	4300	864	0
	Encryption (RSA)	116	31612	75	0
	PCIe	5882	5287	2694	10
	Top Module Without PCIe	3665	9003	1998	5
	Top Module With PCIe	8360	8555	3779	26

Table 5.2: Module wise power consumption

Board	Module Name	Logic Power(mW)	Signal Power(mW)
Kintex 7-325t	BCH Encoder(15,7,2)	0.02	0.01
	BCH Decoder(15,7,2)	0.05	0.07
	Scrambler	0.04	0.00
	Descrambler	0.01	0.00
	Interleaver	0.01	0.01
	DeInterleaver	0.01	0.02
	Frame Aligner	1.34	1.07
	Encryption (RSA)	2.37	1.57
	Decryption (RSA)	3.64	1.89
	Encryption (AES)	2.76	1.41
	Decryption (AES)	2.90	1.29
	PCIe	253.24	45.55
	Top Module Without PCIe	474.18	2.91
	Top Module With PCIe	304.24	56.31

SEU error is random in nature. We have emulated the SEU error by generating random error on the input data using random error generator [122]. The simulation results of BER is shown in the Figure 5.8 with respect to the noise (Eb/N), which varies from 0 dB to 10 dB. Here, we have used Poisson distributed noise [123]. Figure 5.8 shows the efficiency of our system comprising of BCH code with interleaver, scrambler, gives the best performance with respect to the BER in presence the noise. The throughput of the DAQ system is measured as 4.8 Gbps using in Xilinx platform installed in Fedora OS. **A comparison among existing related works and our proposed work is given in Table 5.3. From the table it seems that our work is its first kind of work which includes security protocol incorporated with the error detection and correction mechanism.**

Table 5.3: Comparison of related existing work with the proposed design

Device Used	Speed (Gps)	Error correction (mechanism)	Security	Line coding used
Lattice SCM40 [65]	1.60	Not used	Not used	8b/10b
Altera EP2SGX [124]	2.125	Not used	Not used	8b/10b
Virtex-II Pro [68]	1.75	CRC used for error detection only	Not used	8b/10b 8b/10b
Kintex 7[**Proposed**]	4.80	BCH (15,7) code detection and correction	Yes, AES	Scrambler

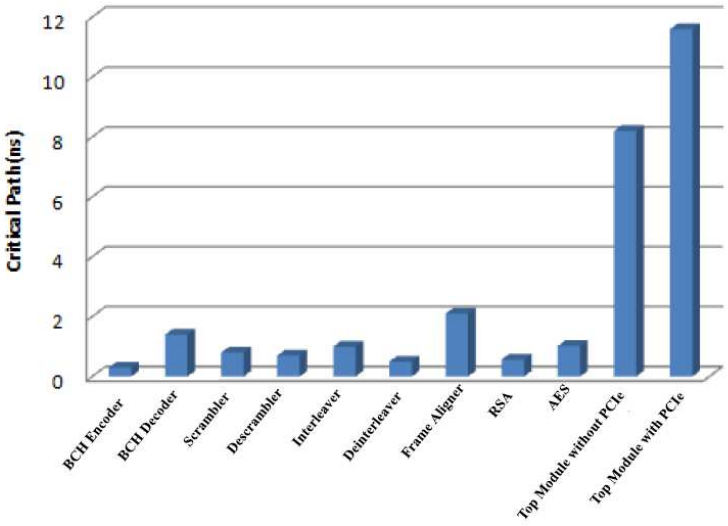

Figure 5.7: Critical time for different block

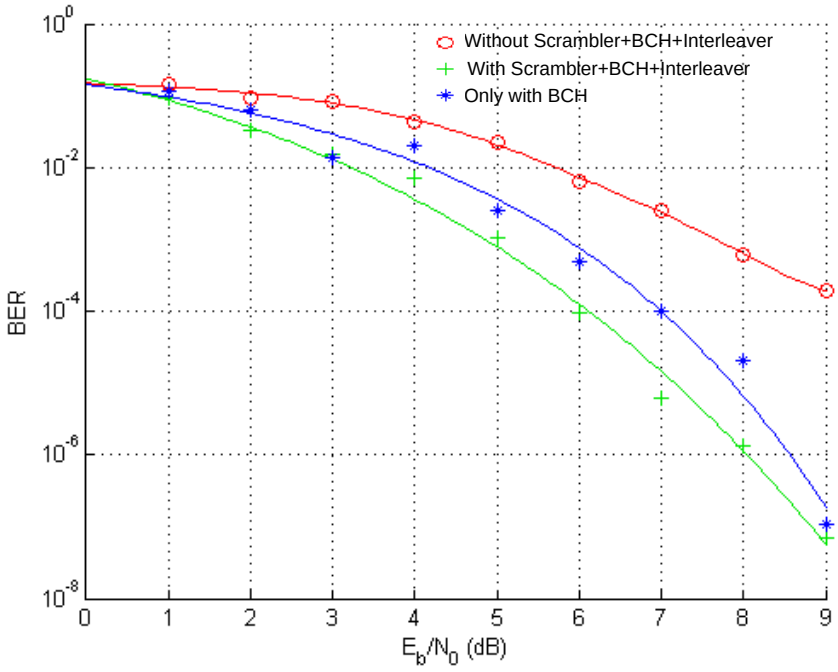

Figure 5.8: Study of of BER with noise

5.4 Summary

In this work we have proposed a DAQ design with fault tolerant secure communication for HEP experiments. The proposed DAQ supports high speed (in terms of Gbps) optical data communication and also corrects multi-bit error. The

design has been implemented on Xilinx Kintex-7 board and real test setup has been developed involving board to board communication and PCIe interfacing with the back end storage system. A detailed performance analysis of the design implementation is presented in terms of timing diagram, resource utilization and critical timing for of each of the blocks (FPGA), power consumption and BER .

Chapter 6

Multi bit Error Correction Model for Crypto Co-processor Architecture

Comprehensive design efforts are needed for implementing algorithms onto an embedded system, which generally have a low power and area budget as well as high throughput requirement. Another important criterion of such systems is fault resiliency, which help them to perform in a robust environment. The general motivation in this direction is to have the general purpose instructions to be taken care by the software, whereas specialized hardware blocks are used to accelerate the execution of complex blocks. FPGA offer an excellent platform for custom hardware design due to its reconfigurability and ability to perform spatial parallelism, it also serves as a good target for embedded applications due to availability of general purpose CPU as well as other needed functional blocks in form of soft cores and dedicated resources (hard cores).

Crypto-hardware can generate faults due to various reasons such as data communication, radiation induced faults resulting to SEU and MBU , intentional attacker etc. Though a good number of research work exists on detecting errors for crypto-hardware but very few works can be found on error correction using hardware techniques. In this chapter, we present a novel crypto-hardware design involving error detection and correction for multi bit errors using BCH .

The proposed design was implemented on FPGA hardware and our implementation shows better performance in terms of resource usage, throughput, error performance compared to the existing research works.

6.1 Introduction

Cryptographic algorithms can be implemented as a co-processor in an embedded system to accelerate the total performance of the system as it executes tasks in parallel with the main processor. A crypto co-processor may give faulty results due to the presence of malicious or injected faults [125].

Electronic devices are affected by two types of fault, which may occur due to thermal effect, electro-migration etc., and they are commonly termed as permanent or transient faults [126]. Due to the material impurity and fabrication related problems, permanent faults occur, whereas transient faults are generated due to environmental effects such as radiation.

Our research is mainly focused on transient errors, which occur due to transient faults through external influences like electromagnetic radiation. Different techniques are used to overcome such transient errors. CED [76] is one those of the techniques, which is based on time redundancy. An extra error detection circuit is used in the main circuit and when an error is detected, the main circuit recompute or roll back the whole operation from the beginning. In [127],AES [6] is proposed with CED. It simply recomputes the basic operation and compares it with the output. In [128], speed up of the system is achieved by computing the function on both edges of the clock. The system is very complex to implement. Moreover using CED a high time delay is generated for the resultant output that may not be suitable in many critical applications where time is an important factor. TMR [129] is another scheme where the same function replica is used and correctness is measured with the majority of the same result collected from all the replicated functional blocks. If two out of three blocks show the same result then it will be treated as the final correct output and is accepted. This scheme is based on majority of the votes. In [129], TMR approach is used for low cost reliable architecture for the S-boxes in AES processor for fault detection. The main disadvantage of TMR scheme is the hardware redundancy, which will be

always more than 200% . The method proposed in [130] is based on information redundancy, which is based on error detecting codes. In [130] parity bits are used to protect the S-box implementation. Parity checker circuit is also used in alogorithms Grϕstl [131]. But this type of simple parity bit implementation has some limitations. If multi bit fault occurs in a byte, this model fails to detect the errors. But, in this type of technique hardware redundancy is less than 100% . For avoiding SEU , Banu et al. used Hamming codes on AES processor [132]. Satoh et al. presented an alternative method that merges data path of encryption and decryption together without any additional arithmetic operators, and double clock cycles are required in one round time [133].

From the review of the previous research works it has been understood that more research effort is needed for multi bit error detection and correction to protect crypto hardware system from transient errors. The key contributions in this chapter can be summarized as follows: [1]

- In this chapter, we propose a first of its kind novel multi-bit error detection and correction scheme (MBEDAC) using BCH technique for AES S-boxes. Our proposed technique also takes care of the error detection and correction in the redundant bits of the error code itself.

- FPGA based hardware implementation of the design has been achieved, which shows better performance in terms of throughput, resource utilization and efficiency metric as compared to most of the existing research works.

Our scheme detects faults that occur in the S-Boxes, which occupy the three fourths of area in the AES processor [134].

Section 4.2 describes the basic methodology of AES algorithm. In Section 6.2.1, theory of BCH encoding and decoding are discussed. Multi bit error correction scheme using BCH is described in Section 6.2.2 and error detection and correction model for AES are discussed in Section 6.3. Section 6.4 gives the details FPGA implementation and results of the proposed model.

[1] Outcomes of this chapter was communicated in ICACCI 2012, IET computers and digital techniques (C7, J2)

6.2 Preliminaries

The basics of AES algorithms had discussed in the earlier chapter. Here we will discuss the basic mathemetics behind the BCH coding.

6.2.1 BCH Coding

BCH coding is used in our system for detecting and correcting errors in the MBEDAC block which is added in the encryption and decryption process described here. The mathematics behind the BCH coding is described in section 5.2.2.1. Here, we incorporate the same hardware block with the AES algorithm for multi-bit error detection and correction.

6.2.2 Multi bit Error Correction Scheme using BCH code

BCH code is a polynomial code over a finite field and also holds cyclic code properties where the encoding is done by the generator polynomial $g(X)$. Generator polynomial is calculated through least common multiplier (LCM) from minimal polynomial over Galois field (GF) (2^m). In BCH, messages are treated as a block and the whole block is encoded at once. BCH can detect and correct multi bit error in the received data arbitrarily depending upon the error correcting code value of (t). The main advantage in using BCH over the other error correction code area are:

- It is the first algorithm, which detects and corrects the multi bit error from the information.
- Can select message block length and error correcting bit value for a given specification.
- The task can be completed using small hardware, which means for encoding and decoding low power device may be used instead of large machines.

The theory behind the BCH code is discussed in section 6.2.1. In data communication data is encrypted with the $BCH(N, K, t)$ where N is the length of the codeword. If any errors occur due to any intentional fault attacks or unintentional

faults in the hardware, output will be generated with errors. But this error can be detected and can be corrected up to maximum t-bits of error. If more than t-bits of error occur in the data, system will fail to correct the error and if it happen repeatedly, message will be given for hardware replacement. The error detection and correction process is illustrated in Figure 6.1. Data with n-bits error goes through error detection module for t-bits where t is the number of error correcting bits in BCH. If the number of error bits is greater than t then the system will fail to correct the error, in that case the data needs to be rejected, otherwise error position will be detected and it will be corrected.

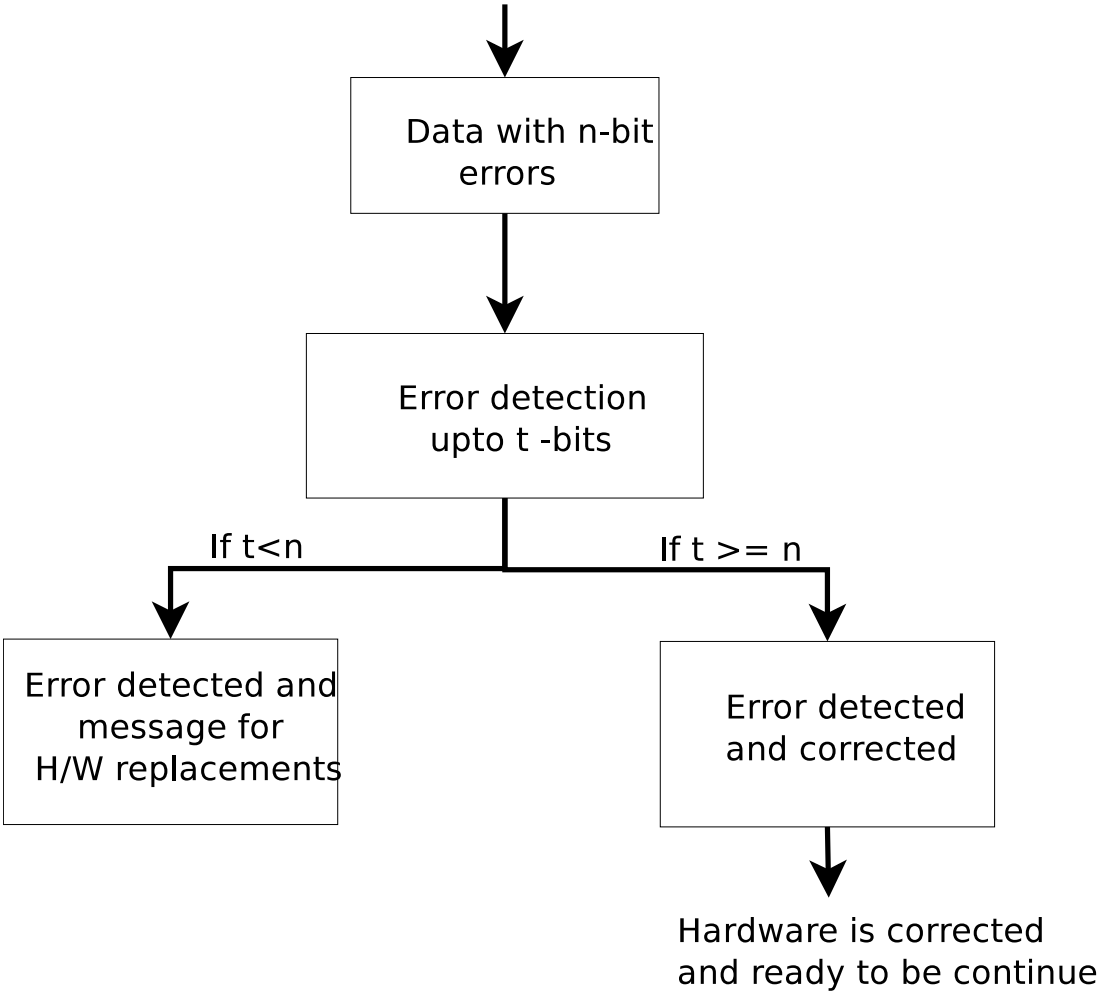

Figure 6.1: Error detection and correction logic flow

6.3 AES Encryption and Decryption with Multi-bit error detection and correction model

We propose the MBEDAC hardware model involving BCH code, which will detect multi-bit error as well as correct it. The MBEDAC fault models with AES encryption and AES decryption are shown in Figure 6.5 and Figure 6.6 respectively. The fault model is running in parallel with the AES encryption and also for decryption. Codewords may be calculated dynamically or can be calculated before, using the original S-box/Inverse S-box and can be saved in the memory. In AES encryption every round has S-box substitution, ShiftRows, MixColoum and AddRoundKey operations. Same for decryption, which has Inversion S-box substitution, Inverse ShiftRows, Inverse MixColoum and Inverse AddRoundKey operations. An error can occur in any of these above stated operations. But, most of the AES crypto hardware consumes (near to three fourth to implement SubByte substitution [129], therefore probabilities of error occurrence in this operation will be high. Our scheme shows detection and correction of error occurs in SubByte/Inv-SubByte substitution due to the transient faults. The scheme can be used in every round with a cost of hardware penalty. In the next subsections we brief the two schemes proposed by us for multi bit error detection and correction.

6.3.1 First fault model

The SubByte substitution is a non-linear substitution, which operates on every byte of S-box matrices through a predefined substitution table. So a new table has been created where every element of the table is recalculated with BCH coding with fixed error correcting value (t). Using BCH for error detection/correction, we calculate the encoder table 6.2 from the original S-box to the mapped encoder (codeword) table, which is stored in the memory of the system.

Here we assumed that this encoder table will not be affected by the faults. Figure 6.3 shows the basic building blocks of this architecture.

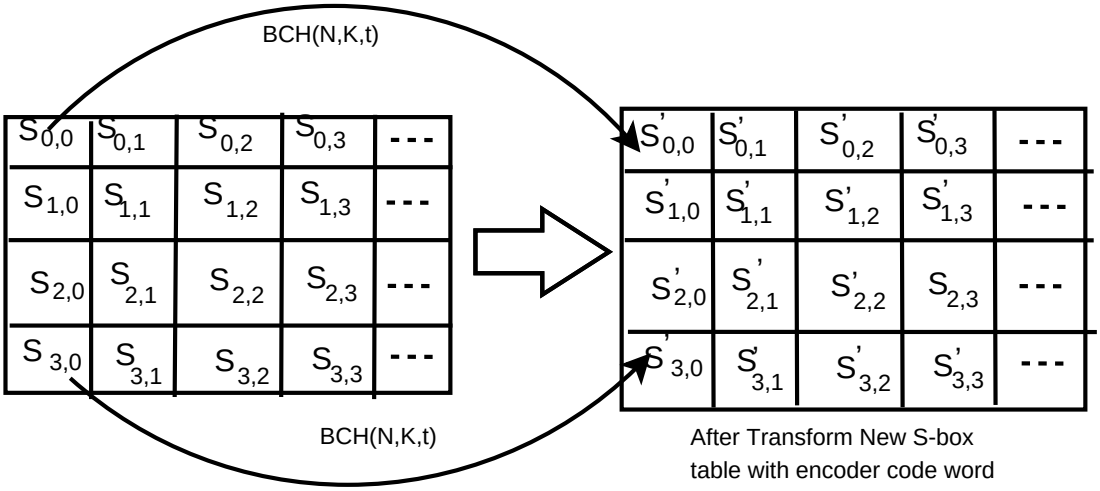

Figure 6.2: Mapped encoder (codeword) table

6.3.2 Second fault model

In spite of the calculated codeword that are stored in the memory we calculate the codeword in the runtime in parallel with the SubByte substitution logic. Then the detection logic circuit evaluates whether there is any error or not. If error exists then only we go for the error position detection and correction module otherwise we just pass the original SubByte transformation results for the next round. This reduces computation and runtime resource utilization. Figure 6.4 shows the data path for this scheme.

6.3.3 AES with fault model

Figure 6.5 and Figure 6.6 shows the AES algorithms with fault model. We are not considering transmission error here. In each round, BCH codeword is calculated by referring the stored BCH code table from the input state. BCH codeword (Equation 5.3) is also calculated from the output of the S-box and compared with the stored one. If they are not same then BCH decoding process is needed to find the error location. Once the error location is found, the value at the corresponding bit position is inverted to correct the error. In Figure 6.5, $C1$ is the stored codeword whereas $C2$ is calculated at runtime. Here we believe that pre-calculated BCH codeword table is kept safe through memory protection techniques [135] and is not affected by the transient faults.

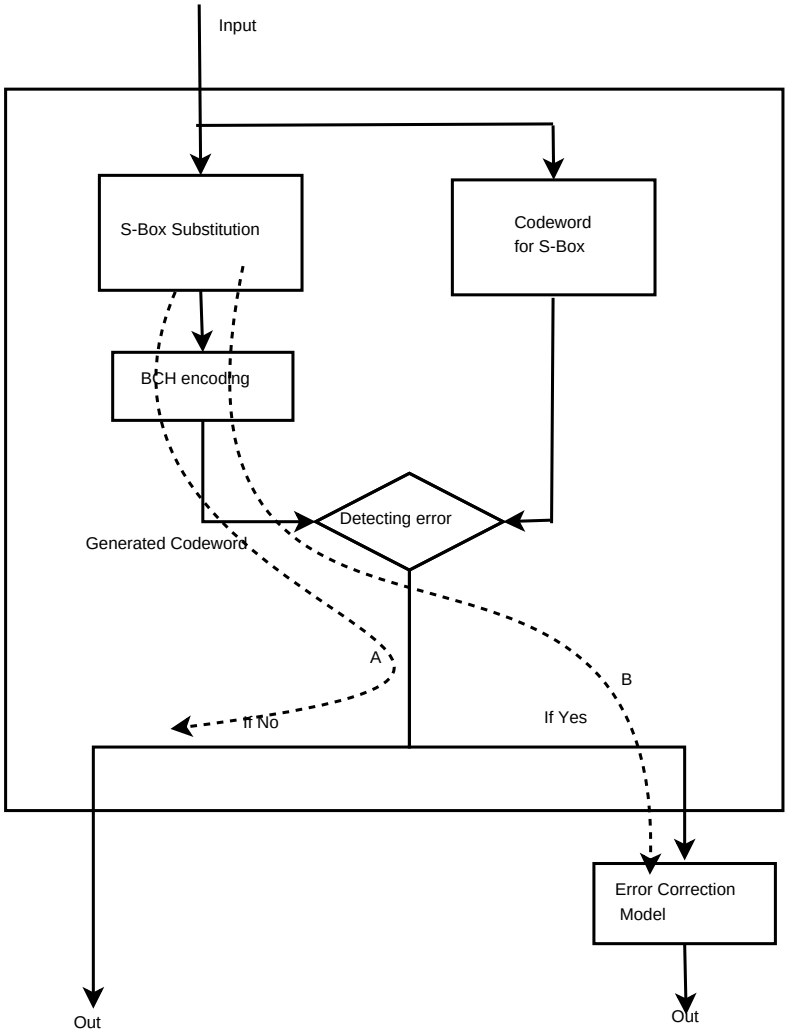

Figure 6.3: Sbox with fault model algorithm 1

6.4 FPGA implementation of the proposed model

The implementation and simulation of the AES-256 with MBEDAC module have been done with Xilinx Virtex-5Lx110t, Virtex4, Spartan6, Zynq and Altera Cyclone II boards. There are four components in the AES core namely *AES Algorithm block, Controller, Key Schedule* and the *Input Interfaces*. VHDL codes were used for hardware architectural design. The logic verification was done using the *Modelsim SE*. In Figure 6.7, the logic diagram of full system with error correction module is shown.

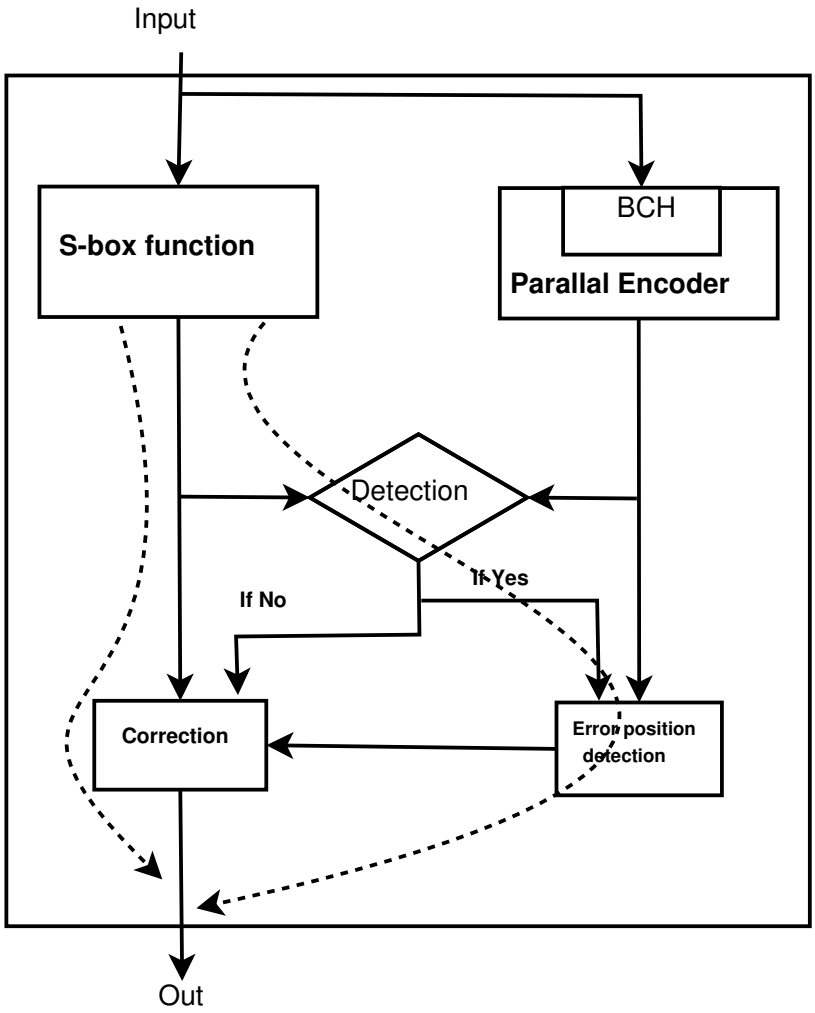

Figure 6.4: Sbox with fault model algorithm 2

Encryption and decryption are controlled by signals from controller block. Out put from Mux 2 that gets the input from S-box and Inv-S-Box will go through the MBEDAC block, which is implemented separately from AES core and assumed that no faults occur in this block. Inside the MBEDAC block, algorithms for error correction algorithm strategy are implemented which are described in Figure 6.3 and Figure 6.4.

Figure 6.5 and Figure 6.6 shows the complete AES encryption and decryption flow diagram with error correction model.

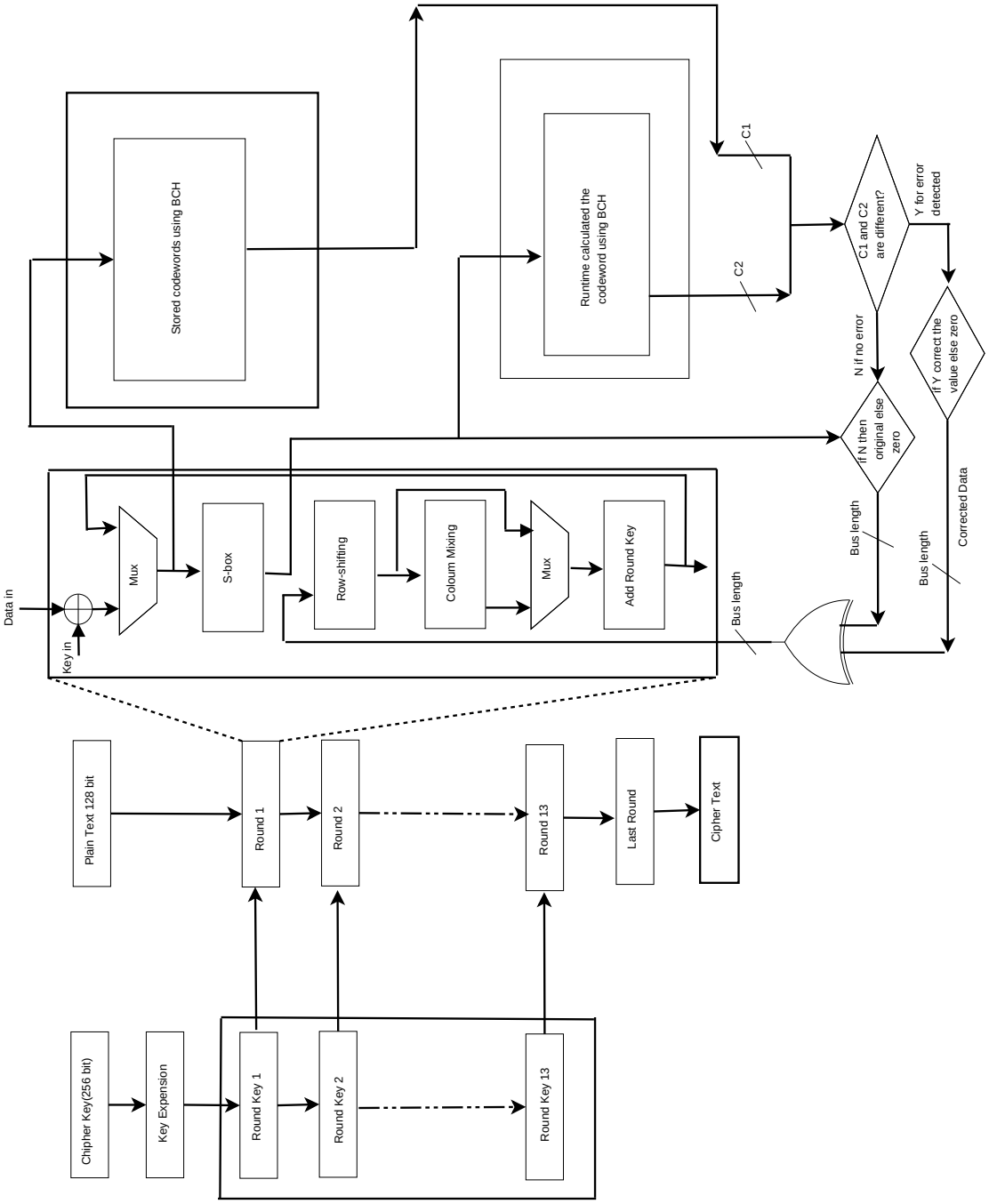

Figure 6.5: Encryption flow with fault model

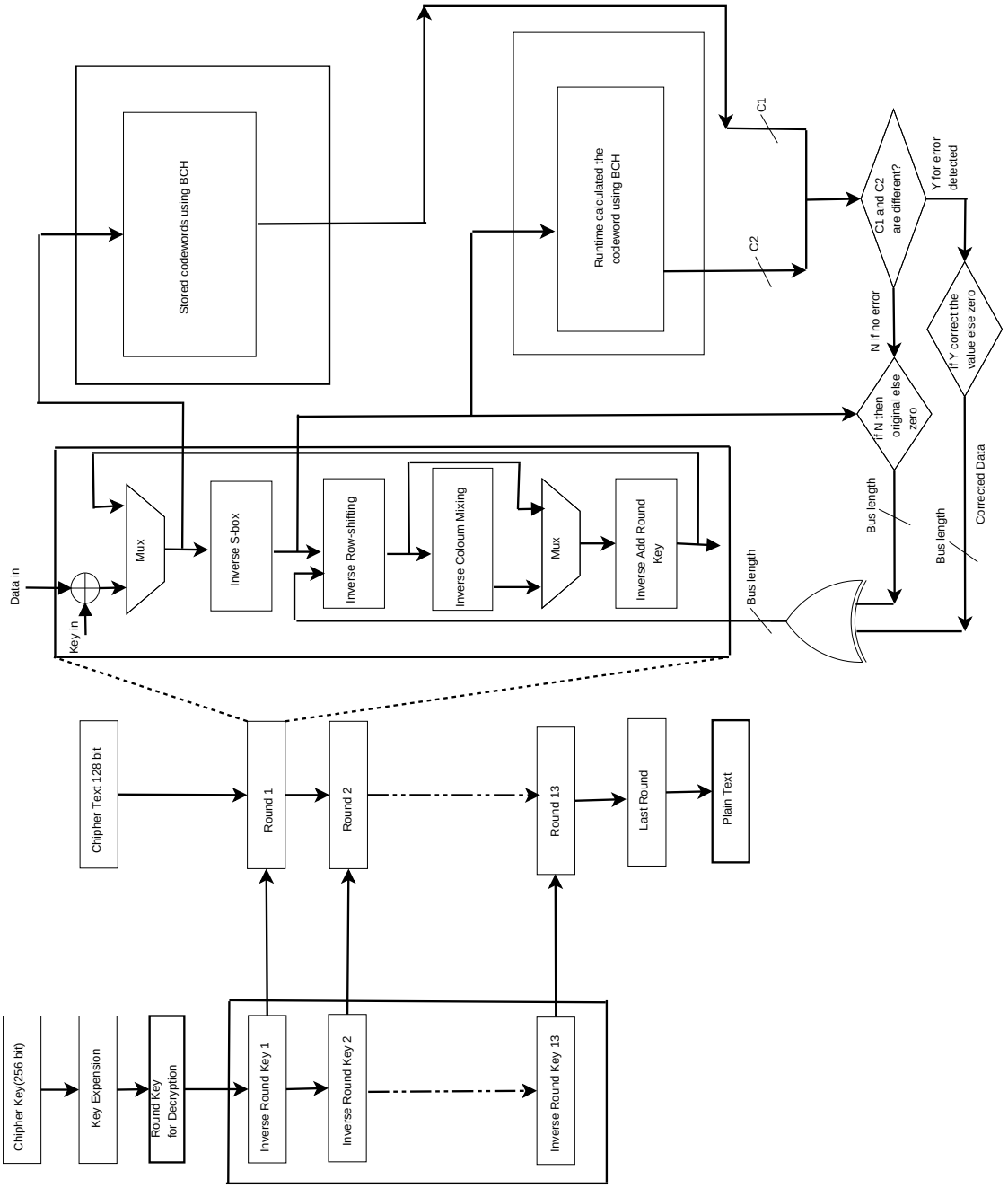

Figure 6.6: Decryption flow with fault model

6.5 Results and Discussion

With respect to the design described in [3], our implementation requires less clock cycles and generates higher throughput for AES implementation. Besides the

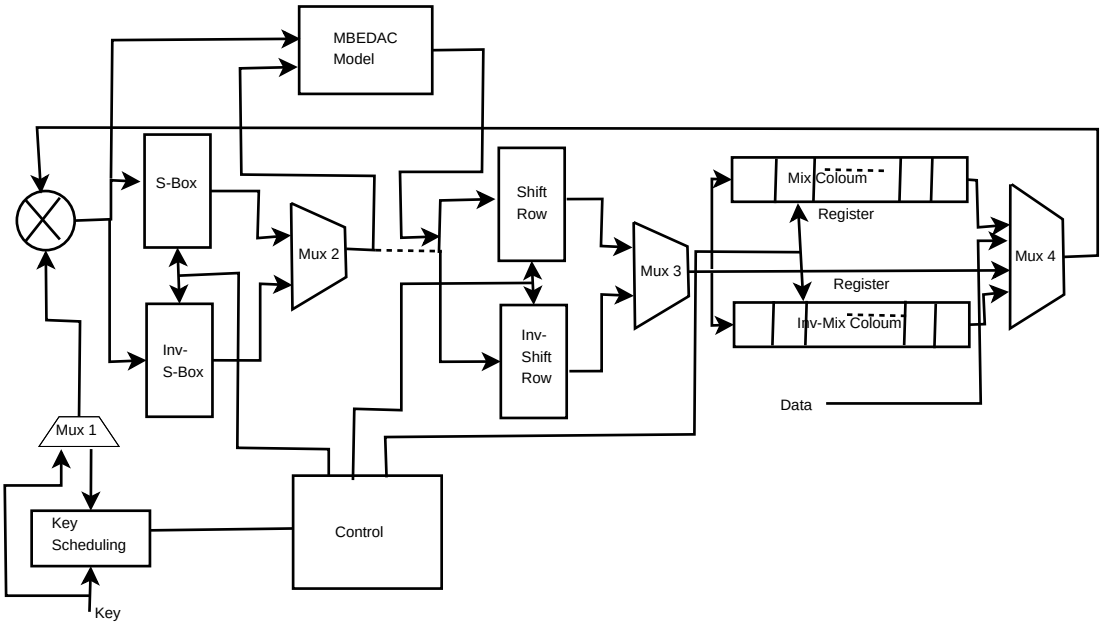

Figure 6.7: AES Core with multi bit error detection and correction model (MBEDAC)

throughput, we have also estimated the parameters like Mbps/Slices, Mbps/MHz for analyzing the performance of our system to compare it with the other related research results. In regards to security choosing of 256 bits for the key value gives maximum security for AES that differs from present works [55], [3], [59]. Throughput of our design is recorded as 1066.67 Mbps for encryption and 533.33 Mbps decryption that are higher with respect to same kind implementations [3], [55].

In [106] , a higher throughput is recorded but this is a AES 128 implementation and also used higher frequency for system implementation. We can also increase the throughput by selecting higher clock frequency with an increased power consumption. Moreover, as our design is a co-processor architecture based approach, it gives more flexibility of multi tasking over real time scenario that is not present in [106]. To draw a straight forward comparison with previous published works in this domain, it is not easy as different FPGA boards, clock frequencies have been used in often such implementations. We choose throughput, power and resource utilization as the metrics for comparison. Comparing on the basis of these matrices we claim that our implementation is better from the

previous published works. Table 6.1 shows a comparison among different works with our work with respect to metrics like throughput, resource, clock frequencies and efficiency.

Table 6.1: Comparison of synthesis results of various AES implementations with and without error correction using various FPGA evaluation boards [3]

Device used	Resources used (Slices,BRAM)	Clock frequency (MHz)	Throughput (Gbps)	Efficiency (Mbps/Slices)	Efficiency (Mbps/MHz)
Spartan-3[107]	523	63.70	0.127	0.243	1.993
VirtexE [136]	3750	50.00	0.243	0.064	4.860
Virtex-II[109]	3474,15 BRAM	80.30	0.275	0.073	3.424
Stratix [110]	5605 logic cells	50.00	0.285	0.052 Gbps/ logic cells	5.700
ASIC module [111]	-	-	0.054	-	-
Processor [112]	-	66.00	0.275	-	4.166
Processor [113]	-	1500.00	>2	-	1.333
Virtex4-LX[106]	1921,20 BRAM	149.00	1.484	0.992	12.590
Virtex5lx110t *[with fault model (proposed)]*	1657, 0 BRAM	125.00	1.067	0.643	8.536
Virtex4Fx140 *[with fault model (proposed)]*	3222 LUT, 0 BRAM	100.00	0.8	0.248/ LUT	8.00
Xilinx Zynq *[with fault model (proposed)]*	3503 LUT, 0 BRAM	100.00	0.8	0.228/ LUT	8.00
Spartan 6 *[with fault model (proposed)]*	3767 LUT, 0 BRAM	100.00	0.8	0.212	8.0
Altera DE150 *[with fault model (proposed)]*	13466 logic cell	100.00	0.8	0.0594/ logic cell	8.00

Table 6.2 shows the resource utilization of BCH module in different FPGA boards.

Table 6.2: Resource Utilization for BCH Model with Encryption and Decryption

FPGA Board	Module Name	Flip-Flop Used	LUT Used
Virtex5lx110t	AES without BCH Fault Module	1183	5199
	BCH Module	474	2289
	Encryption and Decryption Module with BCH	1657	7488
Virtex4Fx140	AES without BCH Fault Module	1189	10554
	BCH Module	488	5388
	Encryption and Decryption Module with BCH	1677	15942
Xilinx Zynq	AES without BCH Fault Module	826	5434
	BCH Module	329	2119
	Encryption and Decryption Module with BCH	1155	7553
Spartan 6	AES without BCH Fault Module	1134	3767
	BCH Module	412	1327
	Encryption and Decryption Module with BCH	1195	5489
Altera Cyclone II	AES without BCH Fault Module	27153(logic cell)	643(dedicated logic)
	BCH Module	13309	643
	Encryption and Decryption Module with BCH	40462	1973

6.5.1 Timing Analysis of AES core and Fault Model

Figure 6.8 shows the timing diagram of AES core. The core will start functioning when the signal reset goes low and it acts as an Encryptor or Decryptor on the basis of signal *enc_dec*.

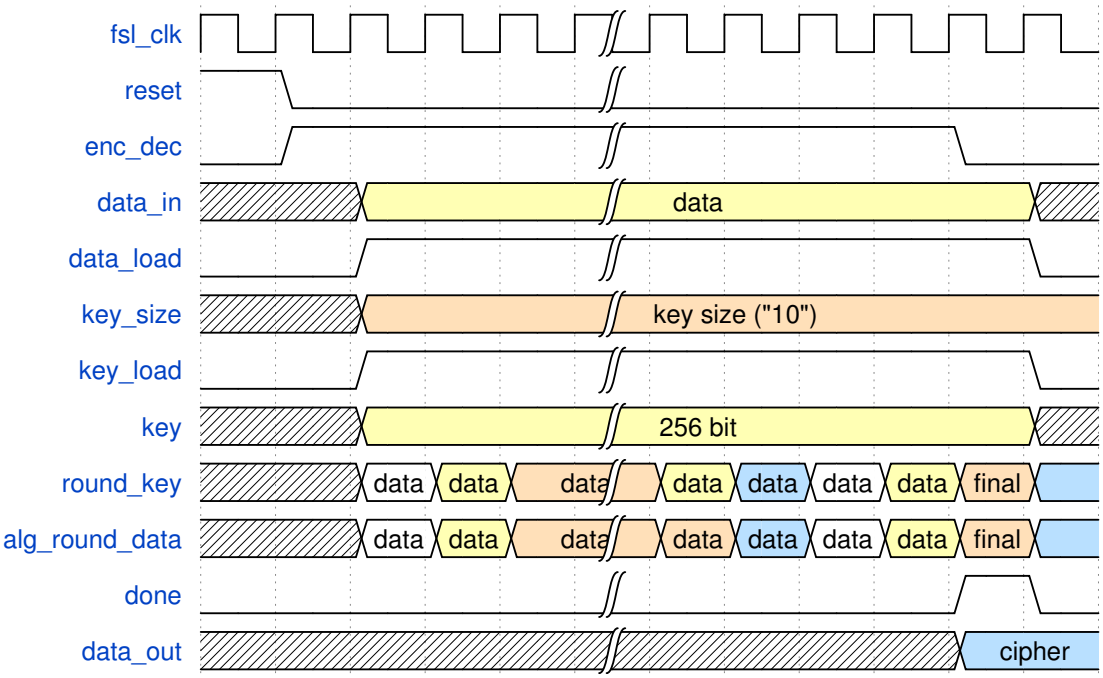

Figure 6.8: Timing diagram of AES

Data gets loaded to the core when *data_load* signal goes high. In the same way the key gets loaded when the *key_load* goes high. Key size is controlled by *key_size* signal. For key sizes of 256 bits we use binary 10 value in the *key_size* signal. In the timing diagram *round_key* and *alg_round_data* signals represent the intermediate round key and the data generated after *AddRoundKey* operation of the algorithm respectively. Timing diagram for error detection and correction block is shown in the Figure 6.9.

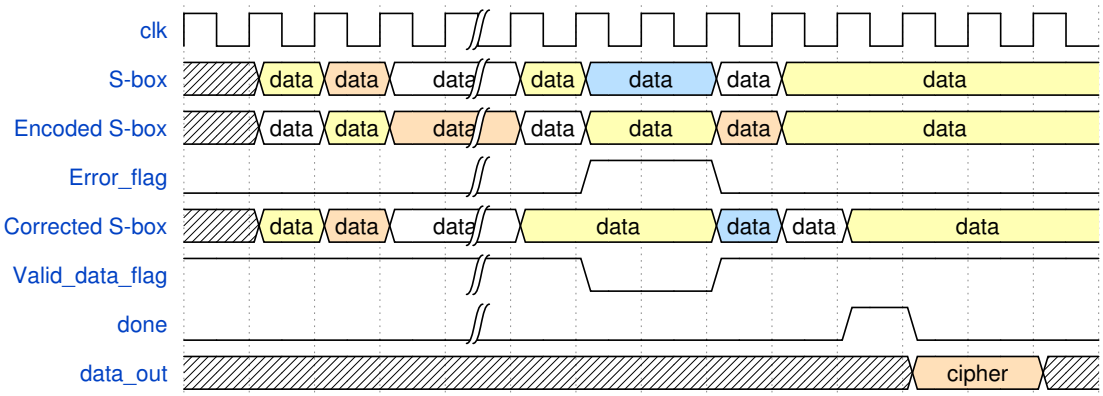

Figure 6.9: Timing diagram for multi bit error detection and correction

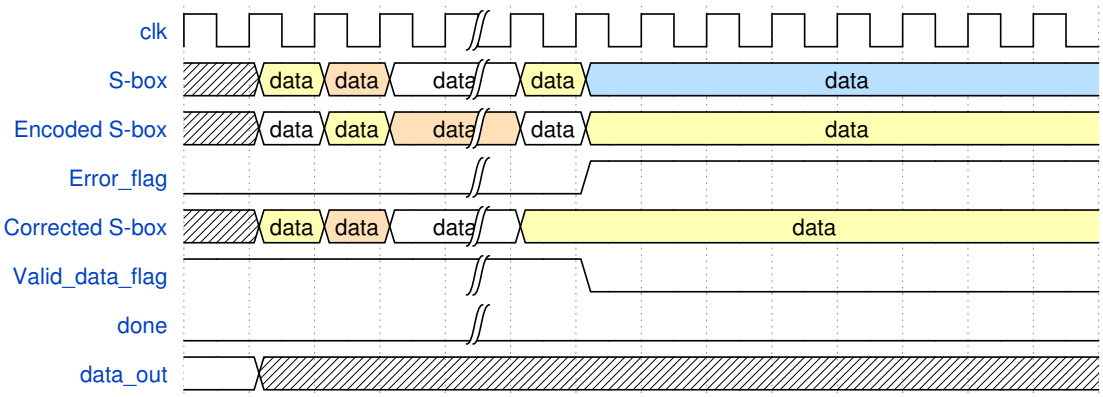

Figure 6.10: Timing diagram for multi bit error detection without any correction due to number of error bits > t

The output of this block is acceptable only when the signal $Error\_flag$ and $valid\_data\_flag$ are low and high respectively. Figure 6.10 represents timing signal of error modules when a error is detected as the $Error\_flag$ signal goes high but the error cannot be corrected as the number of error bits are greater than t which is already discussed in section 6.2.1. As errors cannot be corrected by the error detection and correction block, the value of the signal $Error\_flag$ and $Valid\_data\_flag$ will be always high and low respectively.

6.5.2 Complete Hardware design with MBEDAC

The complete hardware design consists of the various sub hardware components of the AES-256 core along with the MBEDAC module. In Table 6.3 and Table 6.4

we have shown the power utilization of the different sub-blocks as well as that of the entire design for Xilinx and Altera boards. The signal power and logic power of each of the modules are related to power consumption due to signal fluctuation and logic computation respectively. **The static power and dynamic power dissipations for the design based on the board Virtex 5 are 2.489W and 0.077W . Similarly for Virtex 4 , static and dynamic power dissipations values are 2.196W and 0.0165W respectively. In case of Altera Cyclone II board, the value for this proposed design are 2.447W and 0.9816W respectively.**

Table 6.3: Comparison of power dissipations of various blocks of the AES co-processor with error correction using the Xilinx Virtex FPGA board

	Structural Module Name	Power(mW)	Logic Power(mW)	Signal Power(mW)
Virtex5lx110t	Key Scheduling (AES)	692.7	22.35	46.92
	Interface Module (AES)	600.063	0.01	0.62
	Controller Module (AES)	0.02	0.01	0.01
	Algorithm Module (AES)	23.51	6.67	16.83
	MBEDAC Module	1250.53	0.01	25.23
	AES-256 with MBEDAC Module	2566.823	42.80	97.11
Virtex4Fx140	Key Scheduling (AES)	37.08	12.02	25.06
	Interface Module (AES)	22.04	22.03	0.01
	Controller Module (AES)	0.77	0.76	0.01
	Algorithm Module (AES)	39.80	1.56	38.24
	MBEDAC Module	36.53	1.71	34.82
	AES-256 with MBEDAC Module	136.22	38.08	98.14

Comparison of power consumption for the various hardware modules of our design for the different FPGA boards are shown in Figure 6.11. As their exist no related research works on FPGA based implementation of AES-256 with multi-bit error detection and correction using BCH, our presented results can be accepted as the 1st benchmark (baseline) results. The throughput for the encryption processes

Table 6.4: Comparison of power dissipations of various blocks of the AES co-processor with error correction using the Altera FPGA board

	Structural Module Name	Power(mW)	Core Static Thermal Power(mW)	I/O Thermal Power(mW)
Altera Cyclone II	Key Scheduling (AES)	239.03	155.11	83.92
	Interface Module (AES)	193.88	155.95	38.93
	Controller Module (AES)	238.11	155.10	83.01
	Algorithm Module (AES)	239.53	155.11	84.42
	MBEDAC Module	231.27	155.12	76.17
	AES-256 with MBEDAC Module	254.92	155.10	99.82

Table 6.5: Comparision with existing single/multi error correction techniques with proposed model

Design board	Error correction	Correction technique	overhead
Spartan 3E [138]	Single bit	Hamming code	> 100%
Virtex 2 Pro [139]	Single bit	LDPC	100%
ASIC [140]	Three bit	BCH	150%
Virtex 5(proposed)	Two bit	BCH	40%
Virtex 4(proposed)	Two bit	BCH	41%
Xilinx Zynq(proposed)	Two bit	BCH	39%
Altera Cyclone (proposed)	Two bit	BCH	49%

of the designed AES-256 core with MBEDAC is 1066.67 Mbps, which is better compared with that of the previous AES implementation with single bit error correction using Hamming code [132]. In other single bit error correction code resource utilization overhead expenses are very high. In TMR by multiple voting, resource expenses will always be more than 200% [129]. The design proposed by Mathew et al. (2008) for single hamming error detection the resource expenses is ~130% [137].

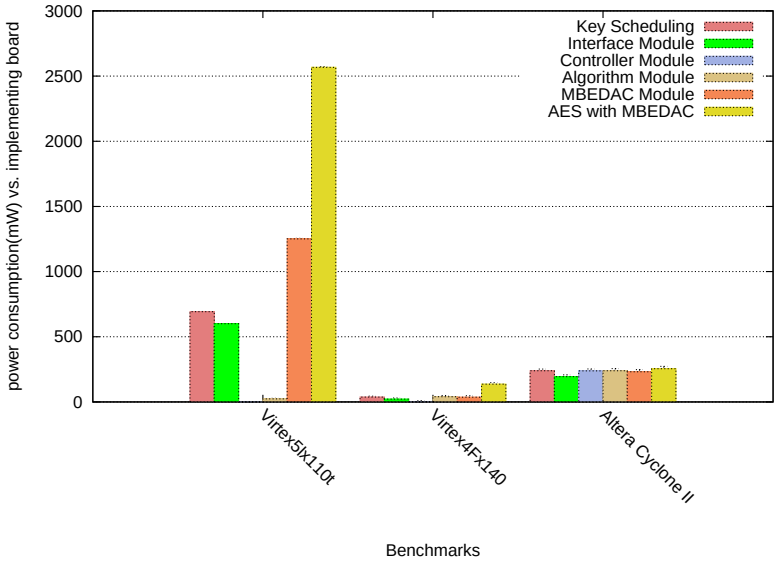

Figure 6.11: Power consumption comparison between different board

Summary

Multi bit error correction and detection model, which is discussed in this chapter will help in applications where SEU occurs randomly. High energy physics applications and other applications where hardware is normally exposed to high energy radiation are commonly affected with SEU errors. Our proposed hardware design can be very useful in such cases, as it assures secured data transfer with multi bit error resiliency. We have implemented our design on various FPGA platforms and our results show excellent performance in comparison to the state of the art research.

Chapter 7

Soft Error Mitigation technique in Configuration bit memory FPGA using Modified Matrix Code

FPGA are readily affected by faults in the presence of radiation and other environmental hazards compared to Application Specific Integrated Circuits. In some critical application like space, satellite, HEP experiments where density of radiation is high, there is a high probability of FPGA configuration memory faults due to his radiation. The effects of faults in configuration memory may be created a permanent damage of circuit flip-flop or single or multiple bit flip(transient faults) of the hardware circuit flip-flop. In both the cases , the hardware will not function correctly after faults happen. In case of permanent faults , the circuit needs to re-route the faulty area or replace the hardware with a new one. In case of transient faults error mitigation and recovery techniques are necessary to protect the FPGA hardware from soft errors. In this chapter, a modified matrix code (MMC) is used for multi-bit error correction in FPGA-based systems, and dynamic partial reconfiguration is considered to reduce the reconfiguration time. We propose a first of its kind methodology for novel transient fault correction using MMC for FPGAs . To validate the design the proposed method has been tested in Kintex FPGA and also estimate its performance in terms of hardware complexity, power consumption, overhead resources and error correction efficiency.

7.1 Introduction

Soft errors, also known as transient errors are the temporary malfunction that occurs in solid state devices due to radiation. They are not reproducible and sometimes lead to SEU in different hardware devices like FPGA [141]. Meanwhile, the demand of FPGAs is gradually increasing in various critical applications like HEP experiments, biomedical instrumentations, deep space explorations *etc.* due to its numerous advantages over ASICs [141]. So fault mitigation techniques are required to protect these FPGA devices from soft errors.

Within the FPGA, most of the memory bits are configuration bits. So the probability of occurrence of an error is more in the configuration memory. It is required to protect the data in the configuration memory while the input data in data memory can change any time with the clock. Commonly used error correcting technologies in FPGA like TMR , CED [76] consume very high hardware resources. Scrubbing [78] and configuration read back [72] are alternative solutions to alleviate the effect of SEU but continuous access to external radiation hardened memories are required in both the schemes. It increases the cost and introduces delay. Delay can be avoided by downloading a part of the bit file using partial reconfiguration (PR) during scrubbing as shown in [142]. Another approach is to use the EDAC codes or CRC to protect the configuration memory without any external memory. EDAC with a less complex decoding circuit can be used to mitigate the effect of the soft error. In [141], the authors used 2D-Hamming product code (HPC) to correct MBU in SRAM-based FPGA. Xilinx uses CRC and EDAC in its soft error mitigation controller which can correct at most two adjacent bits [143]. Authors use convolutional code to mitigate the effect of SEU in [144], with a highly complex decoding circuit.

In this chapter, a novel technique for error detection and correction using MMC is provided which utilizes a very simple decoding circuit to protect the data in the configuration memory from soft errors. At the same time, MMC mounted custom architecture of Internal Configuration Access Port (ICAP) Intellectual Property (IP) is proposed where bit file reading, fault detection/correction, and writing process are pipelined to increase the throughput. Use of the PR with the proposed fault correction module reduces reconfiguration time. In this chapter,

the contributions are summarized as: [1]

7.2 Proposed Modified Matrix Code Algorithm

Coding structure of MMC is very simple because it does not use complex computation of Galois field unlike other commonly used error correcting codes like Bose Choudhury Hocquenghem (BCH) code and Reed-Solomon (RS) code. It also avoids complex decoding circuit used in LDPC and Turbo codes. Total configuration memory is partitioned virtually into some $M \times N$ matrices (also known as a window) and MMC code is used to correct and detect errors in each window. Here the MMC is described with M=7 and N=7, but M and N can be set to any other value.

A simplified example in Figure 7.1(a) illustrates the proposed MMC scheme which achieves multibit error detection and correction using diagonal and vertical parity bits.

[1] Outcomes of this chapter was communicated in ATS 2015, Elsevier MicPro, IET Embedded Letters, IEEE Transaction on Nuclear Physics (C11, J3, J4, T1)

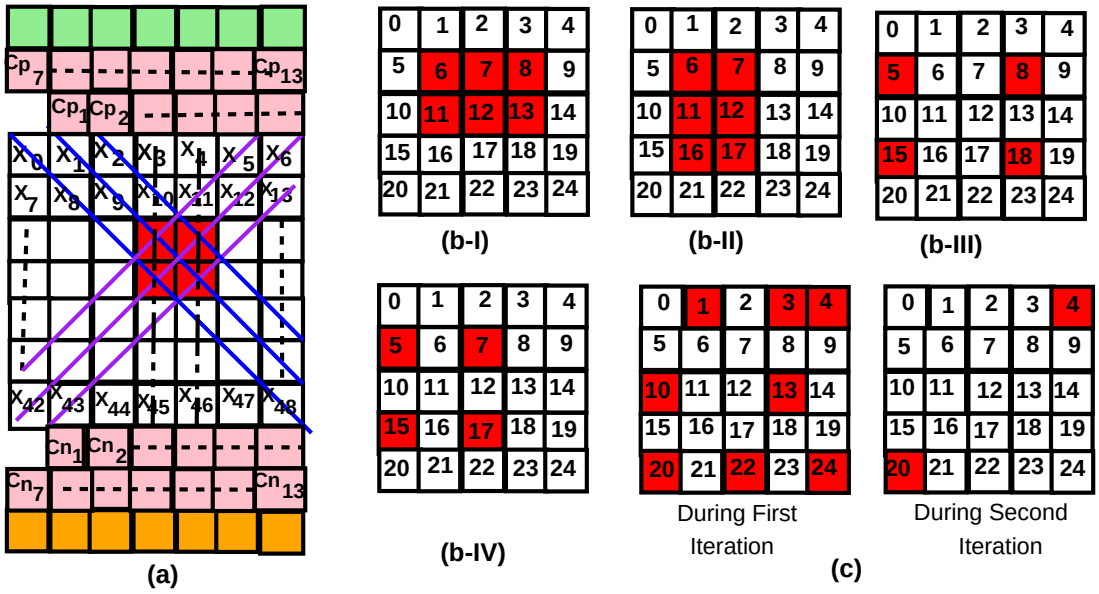

Figure 7.1: a) Encoding/ Decoding using 7x7 window b) Different error patterns c) Error Correction using Multiple Iterations

During the encoding process, parity bits are generated along both diagonal and vertical directions using equations (7.1) to (7.6). These parity bits along with the configuration data will be read back into the MMC block during the decoding process.

$$Cp_i = \bigoplus_{j=0}^{i-1} X_{(i-1)+((R-1)*j)} \qquad \forall i = 1 \quad to \quad R \tag{7.1}$$

$$Cp_i = \bigoplus_{j=i-R}^{R-1} X_{(i-1)+((R-1)*j)} \qquad \forall i = \quad (R+1) \quad to \quad (2*R-1) \tag{7.2}$$

$$Cn_i = \bigoplus_{j=0}^{i-1} X_{(i+(R*(R-1))-1)-((R+1)*j)} \qquad \forall i = \quad 1 \quad to \quad R \tag{7.3}$$

$$Cn_i = \bigoplus_{j=0}^{(2R-i-1)} X_{(i-R)+((R+1)*j)} \qquad \forall i = \quad (R+1) \quad to \quad (2*R-1) \tag{7.4}$$

$$Vpo_i = \bigoplus_{j=0}^{((R-1)/2)} X_{(i+(R*2*j)-1)} \qquad \forall i = \quad 1 \quad to \quad R \tag{7.5}$$

$$Vpe_i = \bigoplus_{j=0}^{((R-1)/2-1)} X_{((R+i-1)+(R*2*j))} \qquad \forall i = \quad 1 \quad to \quad R \tag{7.6}$$

In eqns (7.1) to (7.6) R is the number of rows and columns of each window (number of rows and columns can also be different). Here the symbol $\bigoplus$ indicates modulo-2 sum. Cp and Cn are an array of length (2R-1) bits and store parity bits for the diagonal bits of unit positive and negative slope respectively and is indicated by pink color blocks in Figure 7.1. Vpo and Vpe are arrays of R bits and store parity bits generated from the column information in each of the windows for odd and even positions respectively(indicated by yellow and green color respectively).

Error correction using MMC is explained with the help of the decoding algorithm as described in algorithm 1. In the decoding algorithm, the matrix $A^{'}$ stores 49 bits of erroneous data obtained from the configuration memory. I_{th} is the number of iterations set by the user. There is a trade-off between latency, BER and I_{th}. Increment of I_{th} will improve BER performance but at the same time, latency will also increase. For real time system, optimizing BER and latency the value of I_{th} is chosen as three for this design. But the value of I_{th} can be different for different applications. In Figure 7.1(a) four adjacent bits X_{17},X_{18},X_{24},$X_{25}$are affected by errors. Errors at X_{17} and X_{18} will be corrected by odd vertical parity equation and parity equation along positive diagonal with unit slope through X_{17} and X_{18} respectively. Similarly, errors at X_{24} and X_{25} will be corrected by the even vertical parity equation and parity equation along negative diagonal with unit slope through X_{24} and X_{25} respectively. So, error in these four bits can be corrected using algorithm 1. Some typical error patterns are discussed in Figure 7.1(b). In (I) bits at position 6, 7 and 8 will be corrected

Algorithm 1 Decoding Algorithm for proposed MMC

Input: $A^{'}[R,R], Cp, Cn, Vpo, Vpe, I_{th}$
Output: Corrected or partially corrected $A[R,R]$;

i = 1;$p1$ = 0;$p2$ = 0;
while ($i \leqslant I_{th}$) **do**
Calculate $Cp^{'}, Cn^{'}, Vpo^{'}, Vpe^{'}$ using equation 1 to 6
    for ($i1$ = 1 $\rightarrow (2R-1)$) **do**
$M1(i1)$=$Cp(i1) \oplus Cp^{'}(i1)$; $M2(i1)$=$Cn(i1) \oplus Cn^{'}(i1)$;
    end for
    for ($i2$ = 1 $\rightarrow 2R$) **do**
$M3(i2)$=$Vpo(i2) \oplus Vpo^{'}(i2)$; $M4(i2)$=$Vpe(i2) \oplus Vpe^{'}(i2)$;
    end for
$p1$ and $p2$ store the position of first one in $M1$ and $M2$;
    while (($p1 \leqslant (2R-1)$) *or* ($p2 \leqslant (2R-1)$)) **do**
        $c1$ = 0; $x1$ = 0; $store1$ = $p1$; $store2$ = $p2$;
        if (($p1 \leqslant R$))&&($M1(p1)$= 1)) **then**
            $t1$ = 1;$k1$ = $p1$;$u1$ =(($R-k1$)+1);$j1$ = $k1$;
            while (($c1$ = 0)&&($j1$ >= 1)) **do**
                $w1$ = $M2(u1)$;L1=store $M3^{'}(t1)$ if $j1$ is odd or $M4(t1)$ if $j1$ is even;
                if (((w1=1) *or* ($L1$ = 1))) **then**
                if ((w1=1)&(ones=0)&((p1+2)<=((2*R)-1))&((u1+2)<=((2*R)-1))) **then**
                f1=M1(p1+2) *or* M2(u1+2);
                ones=1;
                end if
                if (f1=0) **then**
                Update A'(j1,t1),M1(p1),M2(u1),M3(t1) or M4(t1),c1=1;
                end if
                end if
                $u1$ = $u1$ + 2;$t1$ = $t1$ + 1;$j1$ = $j1-1$;$f1$ = 0;
            end while
        end if
        if (($p1$ > R))&&($M1(p1)$= 1)) **then**
            $t1$ =($p1-R$)+1;$k1$ = R;$u1$ = $t1$;$j1$ = $t1$;
            while (($c1$ = 0)&&($j1 \leqslant R$)) **do**
                $w1$ = $M2(u1)$;L1=store $M3^{'}(t1)$ if $j1$ is odd or $M4(t1)$ if $j1$ is even;
                if (((w1 = 1) *or* ($L1$ = 1))) **then**
                if ((w1=1)&(ones=0)&((p1+2)<=((2*R)-1))&((u1+2)<=((2*R)-1))) **then**

```
35:                         f1=M1(p1+2) or M2(u1+2);
36:                         ones=1;
37:                     end if
38:                     if (f1=0) then
39:                         Update A'(j1,t1),M1(p1),M2(u1),M3(t1) or
   M4(t1),c1=1;
40:                     end if
41:                 end if
42:                 u1 = u1 + 2;k1 = k1 − 1;j1 = j1 + 1;f1 = 0;
43:             end while
44:             ones=0;
45:         end if
46:         if ((p2 ⩽ R)&&(M2(p2)= 1)) then
47:             b1 = 1;d1 =(R − p2 + 1);e1 = d1;j3 = e1;
48:             while ((x1 = 0)&&(j3 ⩽ R)) do
49:                 y1 = M1(d1); L2=store either Vpo'(b1) or Veo'(b1);
50:                 if (((y1 = 1) or (L2 = 1))) then
51:                     Update A'(j3,b1),M1(d1),M2(p2),M3(b1),M4(b1),x1=1;
52:                 end if
53:                 d1 = d1 + 2;b1 = b1 + 1;j3 = j3 + 1;
54:             end while
55:         end if
56:         if ((p2 > R)&&(M2(p2)= 1)) then
57:             b1 =(p2 − R)+1;d1 = b1;
58:             t2 = 1;j3 = b1;
59:             while ((x1 = 0)&&(j3 ⩽ R)) do
60:                 y1 = M1(d1);L2=store M3'(b1) if
61:                 t2 is odd or M4(b1) if t2 is even;
62:                 if (((y1 = 1) or (L2 = 1))) then
63:                     if ((y1=1)&(ones=0)&
64: ((p2+2)<=((2*R)-1))) then
65:                         f2=M2(p2+2);
66:                     end if
67:                     if (f2=0) then
68: Update A'(t2,b1),M1(d1),M2(p2),M3(b1),M4(b1),x1=1;
69:                     end if
70:                 end if
71: d1 = d1 + 2;b1 = b1 + 1;j3 = j3 + 1;t2 = t2 + 1;
72:             end while
73:         end if
74:         if (((c1 = 1) or (x1 = 1))) then
75:             p1=d1-2;p2=u1-2;
76:         end if
77:         if ((c1 = 0)&&(c2 = 0)) then
78:             p1=p1+1;p2=p2+1;
79:         end if
80:     end while
81:     i = i + 1;
82: end while
```

using parity equation along positive diagonal with unit slope and odd vertical parity equations through these bits. Similarly, bits at position 11, 12 and 13 will be evaluated using parity equation along negative diagonal with unit slope and even vertical parity equations through these bits. Errors in II can be detected by positive diagonal through bits 6 and 17, negative diagonal through bits 7 and 16 and vertical parity equations through bits 11 and 12. This type of error cannot be corrected. Apart from the errors in contiguous locations, the proposed MMC code is also able to detect and correct errors in discrete positions. In (III), the error occurred in bit positions 5, 8, 15 and 18 will be rectified using parity equation along positive diagonal with unit slope and negative diagonal with unit slope through these bits. The authors in [141], illustrate that multi-bit errors along both row and column (as in Figure 7.1 (b-III) and (b-IV)) in a matrix cannot be detected or corrected using 2D-HPC. But using MMC, (b-III) is fully correctable as described previously and (b-IV) is fully detectable(similar to the (b-II)). The uniqueness of the proposed MMC is that it can detect all kinds of error in a window and can also correct most of these errors which are not possible using 2D-HPC. In some cases error may not be corrected in a single iteration, it will take multiple iterations ($i.e.$value of I_{th} is greater than one). In Figure 7.1(c) error occurs at bit positions 1,3,10,13, 22, 24 will be corrected in the first iteration whereas, errors at bit positions 4 and 20 will be corrected in the second iteration. Hardware architecture of the proposed MMC is shown in Figure 7.2.

At the start of the decoding process erroneous data (49 bit) along with the parity bits (generated during encoding) will be read and stored in three dual port RAMs. Diagonal parity will be computed using data stored in the dual port RAM1 and will be sent to the comparator block. Within this block, it will be compared with the stored diagonal parity bits. If a mismatch occurs then, second parity computation block will calculate the vertical parity bits. Results of the first and second parity computation block will be sent to a decider block where the position of erroneous bit will be generated. Next, the position will be forwarded to RAM1 to correct the erroneous bits. After multiple iterations if error is not corrected then D-FF output will go high to indicate the presence of uncorrectable errors.

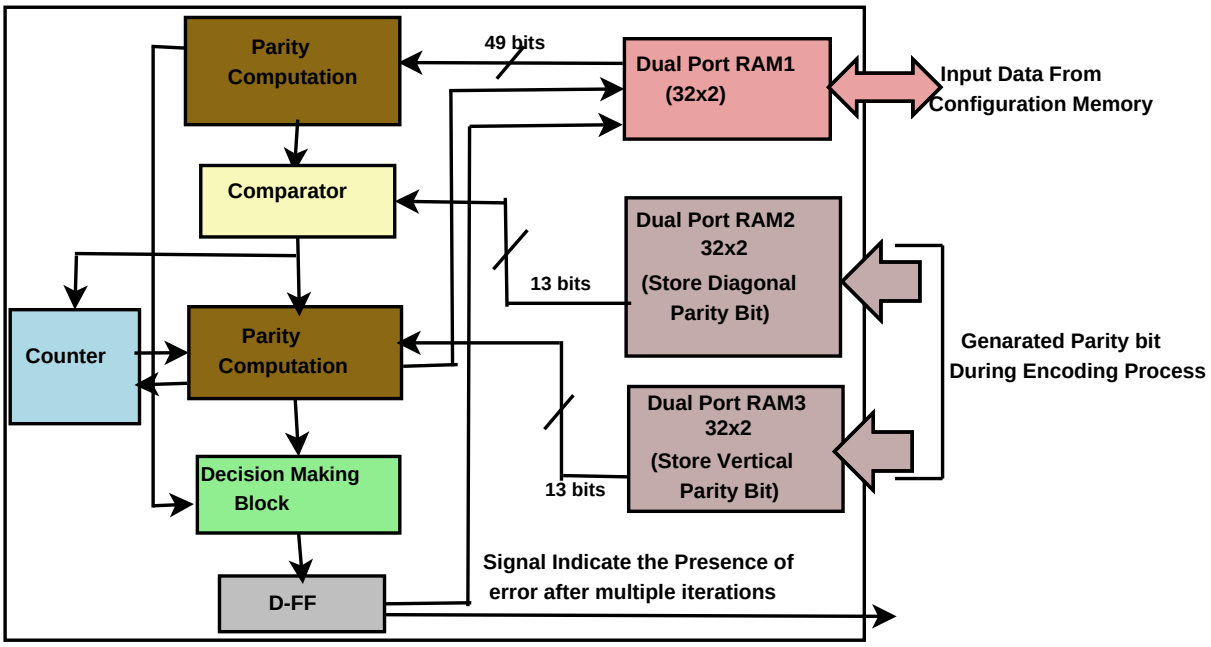

Figure 7.2: Hardware Architecture of the proposed MMC code

7.3 Hardware Architecture

7.3.1 Configuration Area

Application hardware, placed into the configuration area of an FPGA chip consists of few sub-components, which may be stated as standard IP or custom IPs. In this architecture each component takes one partitioned area of the FPGA. The fault correcting block (MMC) reads the binary file (bit file) of the whole application hardware through the ICAP ports. MMC block fixes the errors and stores addresses of the partition regions where faults have occurred. After finishing the bit scanning process, MMC sends only the bit files of the faulty partitions to ICAP. This partitioning reduces reconfiguration time otherwise the complete bit file downloading process may take more reconfiguration time.

7.3.2 Proposed ICAP block

The proposed ICAP block consists of three sub-blocks, slave interface, MMC block and Hardware ICAP (HWICAP) as shown in Figure 7.3.

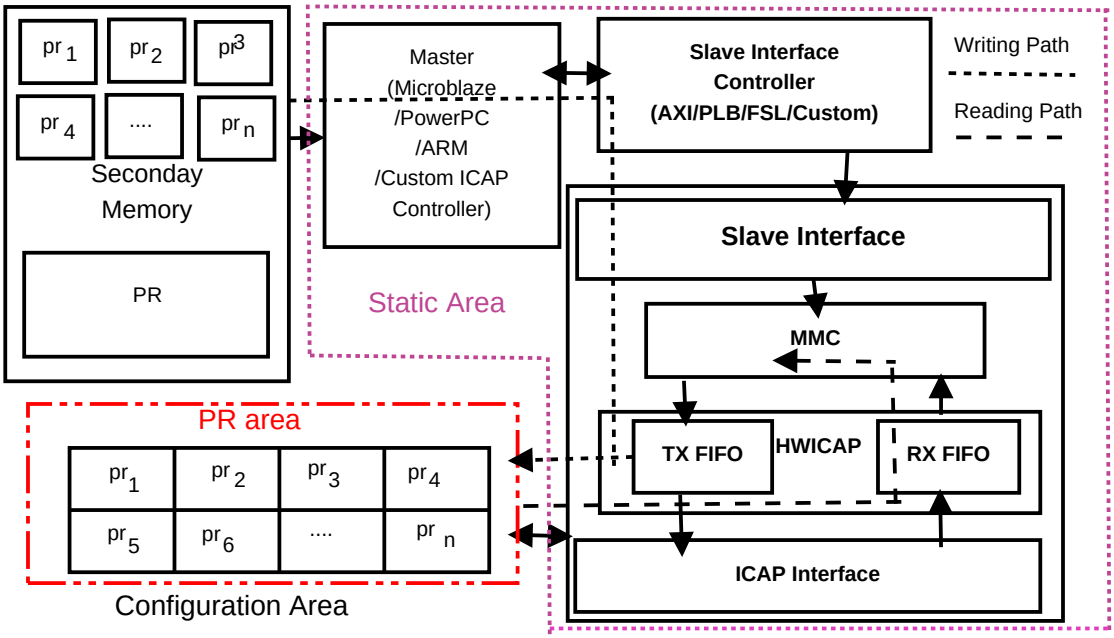

Figure 7.3: Architecture of the proposed ICAP block

7.3.2.1 Slave Interface:

Slave interface of ICAP gets the controlling information from the master. Master sends some information namely ICAP_start, bit addresses, and bit length. ICAP acknowledges master using ICAP_done port while dynamic configuration process is done.

7.3.2.2 MMC Block:

Detail hardware architecture of MMC block is described in Section 7.2. MMC is connected with the HWICAP IP provided by Xilinx. MMC can read or write bit file information from read and write buffer of HWICAP.

7.3.2.3 HWICAP:

The HWICAP is an interface to the ICAP. The write FIFO inside HWICAP stores the configuration bit in Block RAM area locally. Simultaneously, stored data from write FIFO is transferred to the MMC. The MMC scanned data is

stored back in the read FIFO buffer.

7.3.3 Slave Interface Controller

Slave interface controller is an interface between the proposed ICAP block and master. Mircroblaze, Power PC or ARM uses Advanced eXtensible Interface (AXI) , FSL and Processor Local Bus (PLB) to communicate with the proposed ICAP block whereas, this interface will be custom if custom ICAP controller processor is used as a master.

7.3.4 Master ICAP Controller and Secondary memory

Master ICAP Controller is used to move partial bit files from secondary memory to the ICAP controller. Microblaze, Power PC, ARM or custom light weight ICAP controller can be used as the master. Flash or Secure Disk card can be used as a secondary memory to store the bit files.

7.3.5 Work flow

The whole design is separated into three portions namely static region, partial regions and secondary memory. The static part consists of a master processor, slave bus interface and the proposed ICAP block. The partial region contains the application hardware. The work-flow of the proposed design is described below:

Step1: Only the bit file for the static part is downloaded in the configuration area of the FPGA from the secondary memory. The partial bit file of the whole application (PR) is stored in the secondary memory along with sub-component bit files as shown in Figure 7.3. Here $PR = pr_1 + pr_2 + ... + pr_i + ... + pr_N$.
Step2: The PR bit file is now downloaded into the partitioned region through the proposed ICAP block. During this pass, MMC inside the ICAP block calculates the FEC field. Once downloading of static and partial bit is completed; the whole system becomes functional.
Step3: After a specified time interval ICAP starts to read the PR file. If a fault occurs, MMC corrects the specific bits and re-downloads the corresponding

pr_i (i from 1 to N) files in the allocated partitions. If the number of faulty bits exceeds the correction capacity of MMC, the ICAP will request the master to re-download the pr_i file from the secondary memory. In Figure 7.4 two distinct phases are demarcated in the work-flow namely, configuration phase (step1-2) and run phase (step3).

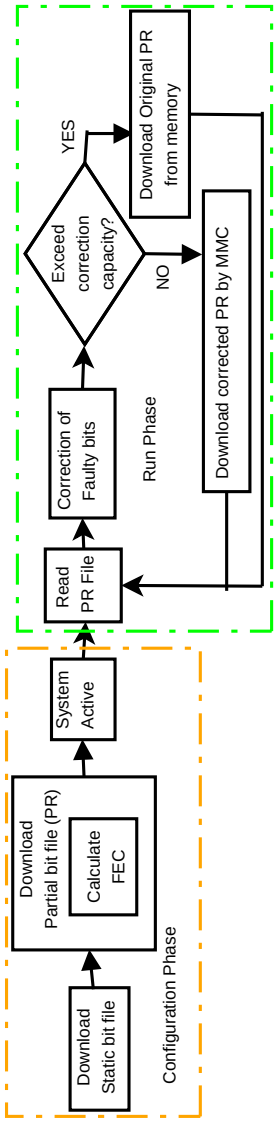

Figure 7.4: Hardware implementation workflow for MMC

7.4 Results and performance analysis

Proposed fault correcting model is implemented on the Xilinx Kintex7 board using Xilinx ISE 14.5 platform. An application design is used to generate the bit file. We have tested our design using behavioral simulations. Figure 7.5(a) compares the performance of the proposed MMC code with the other existing codes in terms

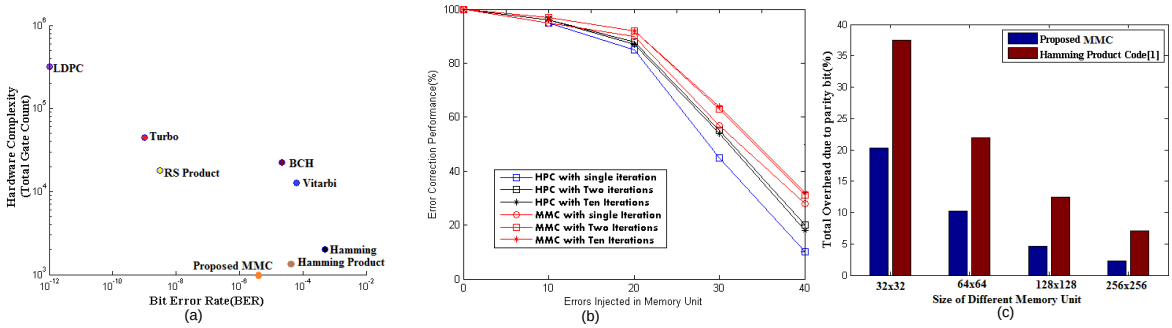

Figure 7.5: a) Hardware Complexity vs BER b) Error correction performance on 32x32 memory unit c) Overhead comparison between proposed MMC code and HPC

of hardware complexity and BER. During the BER calculation input signal to noise ratio is taken as 4dB [141]. Existing multibit error correcting codes with good error correcting performance like LDPC , Turbo, RS product code consume very high hardware resources which are not acceptable in FPGA implementation. Again Hamming code and 2DHPC [141] consume fewer hardware resources but their BER performance is not satisfactory. Our proposed MMC code shows good performance regarding BER compared to Hamming code and HPC and at the same time its hardware complexity is also very less as shown in Figure 7.5(a).

Figure 7.5(b) compares error correcting performance of MMC with HPC over 32x32 memory unit for multiple iterations. A different number of errors are injected randomly within the memory unit as in [141]. It can be observed that when a maximum number of errors are 10 bit, MMC gives quite a similar performance with that of the HPC but when number of erroneous bits increase, MMC outperforms HPC. Figure 7.5(c) compares memory overhead due to the parity bits, between the proposed MMC code and HPC [141] for different memory sizes. It clearly shows that the overhead due to the parity bits is less in MMC code compared to HPC. Table 7.1 compares our proposed code with different existing codes regarding error correcting performance, latency, decoding circuit complexity, resource utilization, power consumption and redundant data. Table 7.1 is prepared using multiple iterations over 32x32 memory unit keeping the code rate constant(0.57). Due to the random nature of the error due to MBU, a range of the error correcting capability for different error correcting code is given instead

Table 7.1: Comparison between Proposed Error Correcting scheme with the other existing Error Correcting scheme

	BCH Code(127,71) [145]	**Hamming Code(7,4) Code(7,4) [146]**	**LDPC [83]**	**Turbo [143]**	**Hamming Product Code [141]**	**Proposed MMC**
ECC performance	between 30% and 50%	<30%	>95%	>95%	between 60% and 90%	between 75% and 90%
Latency	Low	Very Low	Moderate	Long	Moderate	Low
Decoding Complexity	Moderate	Low	High	High	Low	Low
Overhead (Redundant bit)	840	768	1024	$2v(L+K)-L$	882	840
Resource Utilization	2349 (#Slice Reg.)	1282(# Slice Reg.)	1750k (# Gate)	4146 (# Slice Reg.)	-	852 (#Slice Reg.)
power(mw)	-	-	690	-	-	90.7

of exact value. In Turbo code, number of redundant data depends on number of shift registers(indicated by v), modulo two adders (indicated by K) and input data of length L bit. Here values of v, K, L are taken as 2,2 and 1024 bit respectively. It is clearly observed in terms of error correction LDPC and turbo code give the best performance but their decoding complexity and overhead are very high. Though overhead of hamming code is less but their error correcting performance is also not good. (127,71) BCH code and proposed MMC with 7x7 window size require the same redundant data to protect 32x32 memory unit but MMC gives better error correcting performance compare to BCH code. Again, proposed MMC shows low latency, low overhead and good error correcting and detecting performance compare to 2D-HPC. Our proposed code has low latency, moderate overhead, low decoding circuit complexity and strong error correcting capability compared to the other existing error correcting codes. Proposed MMC also consumes fewer hardware resources and power compared to the other codes as shown in Table 7.1. In Table 7.1 '-' represents that suitable data cannot be found in the existing literature.

To accelerate the encoding and decoding process, the proposed error correcting code uses pipelined architecture (*i.e* when error correction process is performed on one set of data of 49 bit width, another set of data can also be read from the configuration memory). Due to the space limitation, we have suppressed the timing simulation of the proposed MMC.

7.5 Summary

In this work, we have proposed a new error correcting code to protect FPGAs from soft error, which gives better performance compared to the other commonly used error correcting codes. We have also proposed one novel hardware architecture with partial reconfiguration for the hardware implementation of the proposed code. In future, we are planning to develop one fault injector emulator to test the error correcting capability of our proposed code in more robust scenario.

Chapter 8

Conclusion and Future Scope

8.1 Conclusion

In this dissertation, our research work emphasizes on design of secure and error resilient high speed data communication methodologies in the domain of embedded system applications. We have chosen high speed data acquisition system for high energy physics experiment as our example application for developing the hardware prototype, as this application gives us an opportunity to benchmark our design in terms of high throughput, error resiliency and security. We have used FPGA as the target architecture for prototyping and estimating the various performance metrics of the sub systems designed and implemented in the course of this research work.

In order to achieve security in the data communication process, we have performed real time encrypted data transmission between multiple FPGA systems exhibiting a significant level of security and providing an appreciable throughput. We were also successful in optimizing the resource utilization as compared to the related research works, which dealt with this kind of implementation. Overall LUT utilization for cryptographic algorithm (RSA) implementation in hardware is 28% whereas a similar implementation described in [42] takes 41% of the FPGA LUTs. Also time required for the implemented RSA encryption and decryption in hardware are 0.103 and 22.337 ms respectively using Spartan-3E FPGA board. Our test process involved capturing the signals on the bus in real time to prove the validity and the efficiency of our design.

Next, we approached for a specific cypto co-processor design. A co-processor based architectural design and the related embedded system implementation for the encryption standard algorithm (AES-256) has been performed. The proposed design is an FPGA based architecture, having a custom crypto co-processor for executing the encryption and the decryption operations, which also communicates with the main processor core with the high speed FSL bus as and when required. We have achieved higher efficiency 8.533 (Mbps/MHz) [113] which is higher with respect to previous implementations.

In order to achieve a complete data communication infrastructure we proposed a DAQ design which provides security and error correction capability HEP experiments. The proposed DAQ supports high speed (in terms of Gbps) optical data communication and also incorporates multi-bit error correction(our design for a single chain DAQ supports upto 4.8 Gbps). The design has been implemented on Xilinx Kintex-7 board and real test setup has been developed involving board to board communication and PCIe interfacing with a host PC. A detailed performance analysis of the design implementation has been performed in terms of timing diagram, resource utilization and critical timing for of each of the blocks (FPGA), power consumption and BER .

Multi bit error correction and detection model, which is discussed in this dissertation will help in applications where SEU occurs randomly. High energy physics applications and other applications where hardware is normally exposed to high energy radiation are commonly affected with SEU errors. Our proposed hardware design can be very useful in such cases, as it assures multi bit error detection and correction for a AES co-processor with multi bit error resiliency. We have implemented our design on various FPGA platforms such as Xilinx Virtex-5, Altera DE2, Xilinc Zynq etc and our results show excellent performance in comparison to the state of the art research works.

In this research, we have proposed a new error correcting code coined asMMC to protect FPGAs from soft error, which gives better performance compared to the other commonly used error correcting codes. We have also proposed one novel hardware architecture with partial reconfiguration for the hardware implementation of the error correction scheme. Partial reconfiguration allows us to correct or reload only the error affected area without changing the other hardware blocks,

in runtime.

8.2 Future Scope

The hardware security model proposed here may not directly fit into present generation of IoT based applications due to much lower resource and energy budgets. Hence, we plan to modify our proposed hardware crypto architectures to lightweight hardware cryptography modules satisfying the need of low resource and low energy requirements for IoT applications.

In this research we have designed and implemented FPGA based GBT transmitter and receiver sub-systems which communicates through an optical link. We were successful in testing point to point GBT communication in TB data rate. In future, we plan to develop an improved setup that can perform multiple GBT based communication based on Master Salve GBT architecture to optimize the hardware requirement as well as to synchronize all the GBT blocks in a better way and thus reducing synchronization error.

In error correction and detection activity, we mainly focused on logical errors assuming that they are created by the incident radiation. Our assumption was realistic as our error injector creates random error in the hardware, which is the most common approach in addressing hardware errors. But, truly speaking radiation can due to various particles (α, β, γ, neutrino etc.) and of various doses and their effect on the semiconductor transport phenomenon must be studied thoroughly to understand the faults that will be created due to radiation effect. These fault models will guide us in understanding the error patterns that will be created in the hardware. In future, we are planning to develop a fault injector emulator that will realize real faults occurring in the semiconductor in respect to various radiation environments, which will test the error correcting capability of our proposed code in presence of various radiation sources. This will make our hardware system more reliable and robust.

List of Publication :

8.2.1 Transaction

[Trns.1] Swagata Mandal, Jogender Saini, Wojciech M. Zabołotny **Suman Sau**, Amlan Chakrabarti and Subhasis Chattopadhyay "Integration of GBTX emulator with XYTER and Data Processing Board for CBM experiment", In Proc of: ***IEEE Transaction on Nuclear Science**, ISSN: 0018-9499*

8.2.2 Journals

[J.4] Swagata Mandal, Rourab Paul, **Suman Sau**, Amlan Chakrabarti, Subhasis Chattopadhyay"A Novel Method for Soft Error Mitigation in FPGA using Modified Matrix Code", Embedded System Letters, Date of Current Version: 26 August 2016, Print ISSN:-1943-0663 Online ISSN: 1943-0671 DOI: 10.1109/LES.2016.2603918 .

[J.3] Swagata Mandal, Rourab Paul, **Suman Sau**, Amlan Chakrabarti, Subhasis Chattopadhyay" Effcient Dynamic Priority Based Soft Error Mitigation Techniques For Configuration Memory of FPGA Hardware" Elsevier, Microprocessors and Microsystems, ISSN: 0141-9331

[J.2] **Suman Sau**, Swagata Mandal, Jogender Saini, Amlan Chakrabarti and Subhasis Chattopadhyay "High Speed Fault Tolerant Secure Communication for Muon Chamber using FPGA based GBTx Emulator", In Proc of: *IOP Journal of Physics :JPCS, ISSN.1742-6596*

8.2.3 Conference

[C.12] **Suman Sau** et.al "Integration of GBTx Emulator with XYTER and Data Processing Board (DPB) for CBM Experiment", In Proc of: *2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD)* , 29 Oct - 06 Nov 2016, Strasbourg, France

[C.11] Swagata Mandal, **Suman Sau**, Amlan Chakrabarti, Sushanta Pal and Subhasish Chattopadhyay "FPGA Implementation of High Speed Latency Optimized Optical Communication System Based on Orthogonal Concatenated Code", In Proc of: *24th IEEE Asian Test Symposium* , IIT-Mumbai, November 22-25, 2015 India

[C.10] Swagata Mandal Wojciech Zabolotny **Suman Sau** Amlan Chkrabarti Jogender Saini Subhasis Chattopadhyay Sushanta Kumar Pal "Internal monitoring of GBTx emulator using IPbus for CBM experiment ", In Proc of: *SPIE, Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2015* , Wilga, Poland May 25, 2015

[C.9] Swagata Mandal, **Suman Sau**, Amlan Chakrabarti, Sushanta Pal and Subhasish Chattopadhyay "FPGA based Novel High Speed DAQ System Design with Error Correction", In Proc of: *IEEE Computer Society Annual Symposium on VLSI (ISVLSI) 2015* Montepiller, July 8-10, 2015

[C.8] **Suman Sau** et.al "FPGA Based High Speed Data Acquisition System for High Energy Physics Experiment", In Proc of: 28^{th} *VLSI Design and Embedded System Conference 2015*, Bengaluru, 3-7 January, India [**Honorable Mention in Design Contest Track**]

[C.7] **Suman Sau**, Rourab Paul, Tanmay Biswas and Amlan Chakrabarti "A Novel AES-256 Implementation on FPGA Using Co-processor Based". In **ACM** Proc of: *International Conference on Advances in Computing, Communications and Informatics (ICACCI-2012)* , Chennai India, 3–5 August 2012, ACM 978-1-4503-1196-0. D.O.I.- 10.1145/2345396.2345499.

[C.6] Rourab Paul, Sangeet Saha, **Suman Sau** and Amlan Chakrabarti, "Novel architecture of Modular Exponent on Reconfigurable System", ***IEEE*** *Proc of Students Conference on Engineering and Systems (SCES-2012), Motilal Nehru National Institute of Technology, Allahabad, 16-18th March, 2012*, **ISBN: 978-1-4673-0455-9**, doi 10.1109/SCES.2012.6199123

[C.5] Rourab Paul, Sangeet Saha, **Suman Sau** and Amlan Chakrabarti, "Real Time Communication between Multiple FPGA Systems in Multitasking En-

vironment Using RTOS", ***IEEE*** *Proc of International Conference on Devices, Circuits and Systems (ICDCS 2012), Karunya University, Coimbatore, 15-16th March, 2012*,**ISBN: 978-1-4577-1545-7.**,doi 10.1109/ICDCSyst.2012.6188714

[C.4] Rourab Paul, Sangeet Saha, **Suman Sau** and Amlan Chakrabarti, "Design and implementation of Real time AES-128 on Real time Operating System for Multiple FPGA Communication", *Proc of* ***IEEE*** *Sponsored International Conference on Innovative Techno-Management Solutions for Social Sector (IEMCON,2012), Science City, Kolkata ,17-18th January,2012, IEM International Journal of Management and Technology (IJMT)*, **ISSN: 2296-6611.**,
Digital version- http://arxiv.org/abs/1205.2153

[C.3] Rourab Paul, **Suman Sau** and Amlan Chakrabarti "Architecture For Real Time Continuous Sorting On Large Width Data Volume For FPGA Based Applications". In: *Proceedings of the RASTM,2011* ,Digital version-http://arXiv:1206.1567[cs.AR] November 12–13, 2011.

[C.2] **Suman Sau**,Chandrajit Pal, and Amlan Chakrabarti. "Design and implementation of real time secured RS232 link for multiple FPGA communication". In: *Proceedings of the 2011 International Conference on Communication, Computing & Security* , February 12–14, 2011, **ISBN: 978-1-4503-0464-1**,doi>10.1145/1947940.1948022

[C.1] **Suman Sau**,Chandrajit Pal, and Amlan Chakrabarti. "Design and Implementation of FPGA based Real Time Data Acqusition System". In: *Proceedings of the 2011 International Conference on Communication, Computing & Design* , December,2010,IIT Kharagpur

==========

[C.1] Swagata Mandal, Rourab Paul, **Suman Sau**, Amlan Chakrabarti and Subhasis Chattopadhyay "Efficient Dynamic Priority Based Soft Error Mitigation Techniques For Configuration Memory of FPGA Hardware", In Proc of: *Elsevier Microprocessors and Microsystems, 2016*

[C.2] Swagata Mandal, Rourab Paul, **Suman Sau**, Amlan Chakrabarti and Subhasis Chattopadhyay "A Novel Method for Soft Error Mitigation in FPGA using Modified Matrix Code", In Proc of: *IEEE Embedded System Letters*

[C.3] **Suman Sau**, Swagata Mandal, Jogender Saini, Amlan Chakrabarti and Subhasis Chattopadhyay "High Speed Fault Tolerant Secure Communication for Muon Chamber using FPGA based GBTx Emulator", In Proc of: *IOP Journal of Physics :JPCS*

[C.4] **Suman Sau** et.al "Integration of GBTx Emulator with XYTER and Data Processing Board (DPB) for CBM Experiment", In Proc of: *2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD)* , 29 Oct - 06 Nov 2016, Strasbourg, France

[C.5] Swagata Mandal, **Suman Sau**, Amlan Chakrabarti, Sushanta Pal and Subhasish Chattopadhyay "FPGA Implementation of High Speed Latency Optimized Optical Communication System Based on Orthogonal Concatenated Code", In Proc of: *24th IEEE Asian Test Symposium* , IIT-Mumbai, November 22-25, 2015 India

[C.6] Swagata Mandal Wojciech Zabolotny **Suman Sau** Amlan Chkrabarti Jogender Saini Subhasis Chattopadhyay Sushanta Kumar Pal "Internal monitoring of GBTx emulator using IPbus for CBM experiment ", In Proc of: *SPIE, Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2015* , Wilga, Poland May 25, 2015

[C.7] Swagata Mandal, **Suman Sau**, Amlan Chakrabarti, Sushanta Pal and Subhasish Chattopadhyay "FPGA based Novel High Speed DAQ System Design with Error Correction", In Proc of: *IEEE Computer Society Annual Symposium on VLSI (ISVLSI) 2015* Montepiller, July 8-10, 2015

[C.8] **Suman Sau** et.al "FPGA Based High Speed Data Acquisition System for High Energy Physics Experiment", In Proc of: 28^{th} *VLSI Design and Embedded System Conference 2015*, Bengaluru, 3-7 January, India [**Honorable Mention in Design Contest Track**]

[C.9] **Suman Sau**, Rourab Paul, Tanmay Biswas and Amlan Chakrabarti "A Novel AES-256 Implementation on FPGA Using Co-processor Based". In **ACM** Proc of: *International Conference on Advances in Computing, Communications and Informatics (ICACCI-2012)* , Chennai India, 3–5 August 2012(published in the **ACM digital library**).

[C.10] **Suman Sau**,Chandrajit Pal, and Amlan Chakrabarti. "Design and implementation of real time secured RS232 link for multiple FPGA communication". In: *Proceedings of the 2011 International Conference on Communication, Computing & Security* , February 12–14, 2011, **ISBN: 978-1-4503-0464-1**,>10.1145/1947940.1948022

[C.11] **Suman Sau**,Chandrajit Pal, and Amlan Chakrabarti. "Design and Implementation of FPGA based Real Time Data Acqusition System". In: *Proceedings of the 2011 International Conference on Communication, Computing & Design* , December,2010,IIT Kharagpur

Appdx A: Abbreviations

AES advanced encryption standard

ALU Arithmetic Logic Unit

ASIC Application-Specific Integrated Circuits

AXI Advanced eXtensible Interface

BCH Bose, Chaudhuri, Hocquenghem

BER Bit Error Rate

CBM-FAIR Compressed Baryonic Matter-Facility for Antiproton and Ion Research

CED concurrent error detection

CERN European Organization for Nuclear Research

CLB configurable logic block

COTS commercial-off-the-self

CPLD Complex Programmable Logic Device

CPU central processing unit

CRC cyclic redundancy check

DAQ data acquisition

DCE Data Circuit-Terminal Equipment

DES Data Encryption Standard

DMA direct memory access

DPR Dynamic Partial Reconfiguration

DRAM dynamic random-access memory

DSP Digital Signal processor

DTE (Data Terminal Equipment

ECC Elliptic curve cryptography

EDAC error detection and correction

EDK Embedded Development Kit

FEC Forward error correction

FEE Frontend Electronics

FIFO First-In First-Out

FPGA Field Programmable Gate Arrays

FSL Fast Simplex Link

GBT gigabit transceiver

GBTx Gigabit transmitter

GF Galois field

GPIO General-purpose input/output

GOSIP Gigabit Optical Serial Interface Protocol

HDL hardware description language

HEP High Energy Physics

HPC Hamming product code

ICAP Internal Configuration Access Port

IoT Internet of things

LDPC low-density parity-check

LED Light-emitting diode

LUT Look up table

LVDS Low-voltage differential signaling

MBEDAC multi-bit error detection and correction scheme

MGT Multi Gigabit Transceiver

MBU multiple bit upsets

MMC modified matrix code

MUCH Muon Chamber

NIST National Institute of Standards and Technology

OPB on chip peripheral bus

PAL Programmable Array Logic

PCIe Peripheral Component Interconnect Express

PLA Programmable Logic Array

PLB Processor Local Bus

QoT quality of transmission

RAM random-access memory

RC4 Rivest Cipher 4

RISC Reduced instruction set computing

RS Reed Solomon codes

RSA Rivest-Shamir Adleman

SDK Software Development Kit

SEU Single Event Upset

SFP Small Form-factor Pluggable

SFPDP Serial Front Panel Data Port

SGDMA scatter gather direct memory access

SHA Secure Hash Algorithm

SSL Secure Sockets Layer

TMR Triple modular redundancy

UART Universal asynchronous receiver/transmitter

VHDL VHSIC Hardware Description Language

XAUI eXtended Attachment Unit Interface

XPS Xilinx Platform Studio

References

[1] Microblaze processor reference guide. x, 22, 51, 60, 61

[2] Fast simplex link (fsl) bus ip. x, 22, 52, 61, 64

[3] O.-c. Mourad, S.-M. Lotfy, M. Noureddine, B. Ahmed, and T. Camel. Design of networked reconfigurable encryption engine. In *Adaptive Hardware and Systems, 2007. AHS 2007. Second NASA/ESA Conference on*, pages 285 – 286, april 2005. xii, 21, 22, 51, 52, 54, 62, 65, 95, 96, 98

[4] CBM. The compressed baryonic matter experiment at fair. *Nuclear Physics A*, 904:941c – 944c, 2013. 3, 6, 12, 13, 71

[5] Ghazanfar Asadi and Mehdi B. Tahoori. Soft error rate estimation and mitigation for sram-based fpgas. In *Proceedings of the 2005 ACM/SIGDA 13th International Symposium on Field-programmable Gate Arrays*, FPGA '05, pages 149–160, New York, NY, USA, 2005. ACM. 3

[6] Joan Daemen and Vincent Rijmen. Advanced encryption standard (aes). *NIST, U.S.department of Commerce, November 2001*, 1(1):1–51, November 2001. 4, 21, 51, 53, 55, 70, 76, 86

[7] Goutam Paul and Subhamoy Maitra. *RC4 Stream Cipher and Its Variants*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2011. 4

[8] Suman Sau, Chandrajit Pal, and Amlan Chakrabarti. Design and implementation of real time secured rs232 link for multiple fpga communication. In *Proceedings of the 2011 International Conference on Communication,*

Computing & Security, ICCCS '11, pages 391–396, New York, NY, USA, 2011. ACM. 4, 7, 51, 70

[9] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Group diffie-hellman key exchange secure against dictionary attacks. In *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, volume 2501 of *Lecture Notes in Computer Science*, pages 497–514. Springer, 2002. 4

[10] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003. 4, 21

[11] D. Eastlake, 3rd and P. Jones. Us secure hash algorithm 1 (sha1), 2001. 5

[12] R. Rivest. The md5 message-digest algorithm, 1992. 5

[13] A. Cilardo, L. Coppolino, N. Mazzocca, and L. Romano. Elliptic curve cryptography engineering. *Proceedings of the IEEE*, 94(2):395–406, Feb 2006. 5

[14] Tereza Otahalova, Zdenek Slanina, and David Vala. Embedded sensors system for real time biomedical data acquisition and analysis. In *11th IFAC Conference on Programmable Devices and Embedded Systems, PDeS 2012, Brno, Czech Republic, May 23-25, 2012.*, pages 261–264, 2012. 6

[15] Huei Lai Daniel Tze Seyedi, Mir Hojjat. *A Novel Intrabody Communication Transceiver for Biomedical Applications*. Springer. 6, 12, 13

[16] CERN. Cern - european organization for nuclear research. the lhc experiments. http://lhc-machine-outreach.web.cern.ch/lhc-machine-outreach/, 2016. Accessed February 2, 2013. 6, 12, 13, 71

[17] T. Good and M. Benaissa. Very small fpga application-specific instruction processor for aes. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 53(7):1477–1486, July 2006. 6

[18] A. Hodjat and I. Verbauwhede. Area-throughput trade-offs for fully pipelined 30 to 70 gbits/s aes processors. *IEEE Transactions on Computers*, 55(4):366–372, April 2006. 6

[19] W. Chou. Inside ssl: accelerating secure transactions. *IT Professional*, 4(5):37–41, Sep 2002. 6

[20] Thomas Wollinger, Jorge Guajardo, and Christof Paar. Security on fpgas: State-of-the-art implementations and attacks. *ACM Trans. Embed. Comput. Syst.*, 3(3):534–574, August 2004. 6

[21] C. Leong, P. Bento, P. Rodrigues, J.C. Silva, A. Trindade, P. Lousa, J. Rego, J. Nobre, J. Varela, J.P. Teixeira, and I.C. Teixeira. Design and test issues of a fpga based data acquisition system for medical imaging using pem. In *Real Time Conference, 2005. 14th IEEE-NPSS*, pages 5 pp.–, June 2005. 7, 21, 29

[22] J Adamczewski-Musch, H G Essel, N Kurz, and S Linev. Data acquisition backbone core dabc release v1.0. *Journal of Physics: Conference Series*, 219(2):022007, 2010. 7, 21, 29

[23] Srivaths Ravi, Anand Raghunathan, Paul Kocher, and Sunil Hattangady. Security in embedded systems: Design challenges. *ACM Trans. Embed. Comput. Syst.*, 3(3):461–491, August 2004. 7

[24] T. J. Todman, G. A. Constantinides, S. J. E. Wilton, O. Mencer, W. Luk, and P. Y. K. Cheung. Reconfigurable computing: architectures and design methods. *IEE Proceedings - Computers and Digital Techniques*, 152(2):193–207, Mar 2005. 7, 16

[25] Suman Sau, Rourab Paul, Tanmay Biswas, and Amlan Chakrabarti. A novel aes-256 implementation on fpga using co-processor based architecture. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, ICACCI '12, pages 632–638, New York, NY, USA, 2012. ACM. 7

[26] T. Srinivas and Meena D. High speed fiber-optic systems for radar applications. In *2015 International Conference on Microwave and Photonics (ICMAP)*, pages 1–2, Dec 2015. 12, 13

[27] Qiang Yang, J. A. Barria, and C. A. Hernandez Aramburo. A communication system architecture for regional control of power distribution networks. In *2009 7th IEEE International Conference on Industrial Informatics*, pages 372–377, June 2009. 12, 13

[28] Michael John Sebastian Smith. *Application-specific Integrated Circuits*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997. 13

[29] Altera fpga), 2017. 15

[30] Xilinx fpga), 2017. 15

[31] Katherine Compton and Scott Hauck. Reconfigurable computing: a survey of systems and software. *ACM Comput. Surv.*, 34(2):171–210, 2002. 16

[32] Altera fpga), 2017. 17

[33] W. Zhou, P. Karlstrom, and D. Liu. Automatic synthesizable hdl generator for nogap. In *2012 IEEE/ACIS 11th International Conference on Computer and Information Science*, pages 119–123, May 2012. 18

[34] Paul Kocher, Ruby Lee, Gary McGraw, and Anand Raghunathan. Security as a new dimension in embedded system design. In *Proceedings of the 41st Annual Design Automation Conference*, DAC '04, pages 753–760, New York, NY, USA, 2004. ACM. Moderator-Ravi, Srivaths. 19

[35] Matt Bishop. *Introduction to Computer Security*. Addison-Wesley Professional, 2004. 20

[36] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978. 20, 29, 30

[37] Alexandre F. Tenca and Çetin Kaya Koç. A scalable architecture for montgomery multiplication. In *Proceedings of the First International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '99, pages 94–108, London, UK, UK, 1999. Springer-Verlag. 20, 29

[38] Alexandre F. Tenca, Georgi Todorov, and Çetin Kaya Koç. High-radix design of a scalable modular multiplier. In *Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '01, pages 185–201, London, UK, UK, 2001. Springer-Verlag. 20, 29

[39] Colin D. Walter. Montgomery's multiplication technique: How to make it smaller and faster. In *Proceedings of the First International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '99, pages 80–93, London, UK, UK, 1999. Springer-Verlag. 20, 29

[40] A. Mazzeo, L. Romano, G. P. Saggese, and N. Mazzocca. Fpga-based implementation of a serial rsa processor. In *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1*, DATE '03, pages 10582–, Washington, DC, USA, 2003. IEEE Computer Society. 21, 29

[41] A. Michalski and D. Buell. A scalable architecture for rsa cryptography on large fpgas. In *Field-Programmable Custom Computing Machines, 2006. FCCM '06. 14th Annual IEEE Symposium on*, pages 331–332, April 2006. 21, 29

[42] K.Asaduzzaman M. Ibrahimy, M.B.Reaz and S.Hussain. *FPGA Implementation of RSA Encryption Engine with Flexible.* International Journal of Communications, 2007. 21, 29, 121

[43] A. K. Daneshbeh and M. A. Hasan. Area efficient high speed elliptic curve cryptoprocessor for random curves. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2 - Volume 2*, ITCC '04, pages 588–, Washington, DC, USA, 2004. IEEE Computer Society. 21

[44] Burak Gövem, Kimmo Järvinen, Kris Aerts, Ingrid Verbauwhede, and Nele Mentens. A fast and compact fpga implementation of elliptic curve cryptography using lambda coordinates. In *Proceedings of the 8th International Conference on Progress in Cryptology — AFRICACRYPT 2016 - Volume 9646*, pages 63–83, New York, NY, USA, 2016. Springer-Verlag New York, Inc. 21

[45] Bahram Rashidi, Sayed Masoud Sayedi, and Reza Rezaeian Farashahi. High-speed hardware architecture of scalar multiplication for binary elliptic curve cryptosystems. *Microelectron. J.*, 52(C):49–65, June 2016. 21

[46] Kimmo Järvinen and Jorma Skyttä. Fast point multiplication on koblitz curves: Parallelization method and implementations. *Microprocess. Microsyst.*, 33(2):106–116, March 2009. 21

[47] Diego F. Aranha, Armando Faz-Hernández, Julio López, and Francisco Rodríguez-Henríquez. Faster implementation of scalar multiplication on koblitz curves. In Alejandro Hevia and Gregory Neven, editors, *Progress in Cryptology – LATINCRYPT 2012*, pages 177–193, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. 21

[48] Juan A. Garay, Rosario Gennaro, Charanjit Jutla, and Tal Rabin. Secure distributed storage and retrieval. *Theor. Comput. Sci.*, 243(1-2):363–389, July 2000. 21, 51

[49] Warwick Ford and Burton S. Kaliski Jr. Server-assisted generation of a strong secret from a password. *Enabling Technologies, IEEE International Workshops on*, 0:176, 2000. 21, 51

[50] Philip MacKenzie and Michael K. Reiter. Networked cryptographic devices resilient to capture. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, pages 12–, Washington, DC, USA, 2001. IEEE Computer Society. 21, 51

[51] S. Fernando and H. Yajun. Aes embedded hardware implementation. In *Field-Programmable Custom Computing Machines, 2005. FCCM 2005. 13th Annual IEEE Symposium on*, pages 103 –109, aug. 2007. 21, 22, 51, 52

[52] E. Canto, F. Fons, and M. Lopez. Reconfigurable opb coprocessors for a microblaze self-reconfigurable soc mapped on spartan-3 fpgas. In *IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on*, pages 4940 –4944, nov. 2006. 22, 51, 60

[53] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3 edition, 2003. 22, 52

[54] Platform studio and the embedded development kit (edk), 2011. 22, 39, 46, 52, 62

[55] Peng Zhang, O. Elkeelany, and L. McDaniel. An implementation of secured smart grid ethernet communications using aes. In *IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the*, pages 394 –397, march 2010. 22, 52, 55, 65, 96

[56] Introduction to the altera nios ii soft processor. ser. Blue Book, No. 4, 2010. 22, 52

[57] Suman Sau, Chandrajit Pal, and Amlan Chakrabarti. Design and implementation of real time secured rs232 link for multiple fpga communication. In *Proceedings of the 2011 International Conference on Communication, Computing & Security*, ICCCS '11, pages 391–396, New York, NY, USA, 2011. ACM. 22, 52

[58] Peter Gutmann. An open-source cryptographic coprocessor. In *Proceedings of the 9th conference on USENIX Security Symposium - Volume 9*, SSYM'00, pages 8–8, Berkeley, CA, USA, 2000. USENIX Association. 22, 52

[59] Xinmiao Zhang and K.K. Parhi. High-speed vlsi architectures for the aes algorithm. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 12(9):957 –967, sept. 2004. 22, 52, 96

[60] Analysis of aes hardware implementations by song j. park. 22, 52

[61] Hua Li and Jianzhou Li. A high performance sub-pipelined architecture for aes. In *Computer Design: VLSI in Computers and Processors, 2005. ICCD 2005. Proceedings. 2005 IEEE International Conference on*, pages 491 – 496, oct. 2005. 22, 52

[62] Z. Huixin, H. Qi, L. Suhua, and Y. Haiguang. The design for lvds high-speed data acquisition and transmission system based on fpga. In *2011 IEEE 3rd International Conference on Communication Software and Networks*, pages 383–386, May 2011. 22

[63] Wang Lixin, Song Wei, and Lv Chao. Implementation of high speed real time data acquisition and transfer system. In *Industrial Electronics and Applications, 2009. ICIEA 2009. 4th IEEE Conference on*, pages 382–386, May 2009. 22, 70

[64] Abhijit Athavale and Carl Christensen. *High-Speed Serial I/O Made Simple A Designers' Guide, with FPGA Applications*. Xilinx. 22

[65] S. Minami, J. Hoffmann, N. Kurz, and W. Ott. Design and implementation of a data transfer protocol via optical fiber. *Nuclear Science, IEEE Transactions on*, 58(4):1816–1819, Aug 2011. 23, 71, 82

[66] E. Kadric, N. Manjikian, and Z. Zilic. An fpga implementation for a high-speed optical link with a pcie interface. In *SOC Conference (SOCC), 2012 IEEE International*, pages 83–87, Sept 2012. 23, 70, 71, 81

[67] Liansheng Liu, Chuan Liu, Yu Peng, and Datong Liu. A design of fibre channel node with pci interface. In *Instrumentation and Measurement Technology Conference (I2MTC), 2013 IEEE International*, pages 1817–1822, May 2013. 23, 71

[68] Hao Xu, Zhan'an Liu, Yunpeng Lu, Lu Li, Dixin Zhao, and Ya'nan Guo. Fpga based high speed data transmission with optical fiber in trigger system of besiii. In *Nuclear Science Symposium Conference Record, 2007. NSS '07. IEEE*, volume 1, pages 818–821, Oct 2007. 23, 70, 71, 81, 82

[69] C. Mattihalli. Design and realization of serial front panel data port (sfpdp) protocol. In *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, pages 2505–2509, April 2012. 23, 70, 81

[70] H. Kavianipour and C. Bohm. High performance fpga-based scatter/gather dma interface for pcie. In *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2012 IEEE*, pages 1517–1520, Oct 2012. 23, 70, 81

[71] M. Wirthlin. High-reliability fpga-based systems: Space, high-energy physics, and beyond. *Proceedings of the IEEE*, 103(3):379–389, March 2015. 24

[72] G.-H. Asadi and M.B. Tahoori. Soft error mitigation for sram-based fpgas. In *VLSI Test Symposium, 2005. Proceedings. 23rd IEEE*, pages 207–212, May 2005. 25, 26, 106

[73] S. Golshan and E. Bozorgzadeh. Single-event-upset (seu) awareness in fpga routing. In *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, pages 330–333, June 2007. 25

[74] M. Violante, L. Sterpone, M. Ceschia, D. Bortolato, P. Bernardi, M.S. Reorda, and A. Paccagnella. Simulation-based analysis of seu effects in sram-based fpgas. *Nuclear Science, IEEE Transactions on*, 51(6):3354–3359, Dec 2004. 25

[75] F.L. Kastensmidt, L. Sterpone, L. Carro, and M.S. Reorda. On the optimal design of triple modular redundancy logic for sram-based fpgas. In *Design, Automation and Test in Europe, 2005. Proceedings*, pages 1290–1295 Vol. 2, March 2005. 25

[76] Daniel P. Siewiorek and Robert S. Swarz. *Reliable Computer Systems (3rd Ed.): Design and Evaluation*. A. K. Peters, Ltd., Natick, MA, USA, 1998. 25, 71, 86, 106

[77] B. Pratt, M. Caffrey, J.F. Carroll, P. Graham, K. Morgan, and M. Wirthlin. Fine-grain seu mitigation for fpgas using partial tmr. *Nuclear Science, IEEE Transactions on*, 55(4):2274–2280, Aug 2008. 25

[78] S. Manz, J. Gebelein, A. Oancea, H. Engel, and U. Kebschull. Radiation mitigation efficiency of scrubbing on the fpga based cbm-tof read-out controller. In *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*, pages 1–6, Sept 2013. 26, 106

[79] Ignacio Herrera-Alzu and Marisa López-Vallejo. Self-reference scrubber for tmr systems based on xilinx virtex fpgas. In José L. Ayala, Braulio García-Cámara, Manuel Prieto, Martino Ruggiero, and Gilles Sicard, editors, *PATMOS*, volume 6951 of *Lecture Notes in Computer Science*, pages 133–142. Springer, 2011. 26

[80] R.T. Chien. Cyclic decoding procedures for bose- chaudhuri-hocquenghem codes. *Information Theory, IEEE Transactions on*, 10(4):357–363, 1964. 26, 71, 75

[81] B. Varghese, S. Sreelal, P. Vinod, and AR. Krishnan. Multiple bit error correction for high data rate aerospace applications. In *Information Communication Technologies (ICT), 2013 IEEE Conference on*, pages 1086–1090, April 2013. 26

[82] H.C. Chang and C. Shung. A reed-solomon product-code (rs-pc) decoder for dvd applications. In *Solid-State Circuits Conference, 1998. Digest of Technical Papers. 1998 IEEE International*, pages 390–391, 1998. 26

[83] Chih-Hao Liu, Shau-Wei Yen, Chih-Lung Chen, Hsie-Chia Chang, Chen-Yi Lee, Yar-Sun Hsu, and Shyh-Jye Jou. An ldpc decoder chip based on self-routing network for ieee 802.16e applications. *Solid-State Circuits, IEEE Journal of*, 43(3):684–694, March 2008. 26, 119

[84] Q. Yu, S. C. Burleigh, R. Wang, and K. Zhao. Performance modeling of licklider transmission protocol (ltp) in deep-space communication. *IEEE Transactions on Aerospace and Electronic Systems*, 51(3):1609–1620, July 2015. 27

[85] Mouna Baklouti and Mohamed Abid. Multi-softcore architecture on fpga. *Int. J. Reconfig. Comput.*, 2014:14:14–14:14, January 2014. 28

[86] Representative Acromag and Technical Note. *FPGA Modules with an Integrated Processor Drive Real-Time Applications*. 28

[87] Bruce Schneier. *Applied Cryptography (2Nd Ed.): Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1995. 29, 31

[88] Cetin Kaya Koc. *High-Speed RSA Implementation*. RSA Data Security, Inc, Redwood City, 1994. 34

[89] Donald E. Knuth. Satisfiability and the art of computer programming. In *Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*, page 15, 2012. 37

[90] Microblaze soft core processor. 40

[91] Power PC Xilinx. Ppc405 processor reference guide, http://www.xilinx.com/tools/ppc_405.htm. *Xilinx, Power PC*. 40

[92] Himanshu Thapliyal and M B. Srinivas. Vlsi implementation of rsa encryption system using ancient indian vedic mathematics. 5837, 10 2006. 47

[93] Vibhor Garg, V. Arunachalam, Hagai Bar-El, Yachao Zhou, Xiaojun Wang, Alfred J. Menezes, Paul C. van Oorschot, Samir Palnitkar, Omar Nibouche, Mokhtar Nibouche, Ahmed Bouridane, Ammar Belatreche, and John D. Carpinelli. Architectural analysis of rsa cryptosystem on fpga. 2011. 47

[94] Qasem Abu Al-Haija, Mahmoud Smadi, Monther Al-Ja'fari, and Abdullah Al-Shua'ibi. Efficient fpga implementation of rsa coprocessor using scalable modules. *Procedia Computer Science*, 34:647 – 654, 2014. The 9th International Conference on Future Networks and Communications (FNC'14)/The 11th International Conference on Mobile Systems and Pervasive Computing (MobiSPC'14)/Affiliated Workshops. 47

[95] E. A. Michalski and D. A. Buell. A scalable architecture for rsa cryptography on large fpgas. In *2006 International Conference on Field Programmable Logic and Applications*, pages 1–8, Aug 2006. 48

[96] K. U. Järvinen and J. O. Skyttä. High-speed elliptic curve cryptography accelerator for koblitz curves. In *2008 16th International Symposium on Field-Programmable Custom Computing Machines*, pages 109–118, April 2008. 48

[97] K.C. Cinnati Loi and Seok-Bum Ko. High performance scalable elliptic curve cryptosystem processor for koblitz curves. *Microprocessors and Microsystems*, 37(4):394 – 406, 2013. 48

[98] Paulo C. Realpe-Muñoz and Jaime Velasco-Medina. High-performance elliptic curve cryptoprocessors over on koblitz curves. *Analog Integr. Circuits Signal Process.*, 85(1):129–138, October 2015. 48

[99] Omran Ahmadi, Darrel Hankerson, and Francisco Rodríguez-Henríquez. Parallel formulations of scalar multiplication on koblitz curves. 14(3):481–504, feb 2008. 48

[100] K. Jarvinen and J. Skytta. On parallelization of high-speed processors for elliptic curve cryptography. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 16(9):1162–1175, Sept 2008. 48

[101] K. C. Cinnati Loi and Seok-Bum Ko. Parallelization of scalable elliptic curve cryptosystem processors in gf(2m). *Microprocessors and Microsystems*, 45:10 – 22, 2016. 48

[102] V. S. Dimitrov, K. U. Järvinen, M. J. Jacobson, W. F. Chan, and Z. Huang. Fpga implementation of point multiplication on koblitz curves using kleinian integers. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, pages 445–459, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. 48

[103] Emac lite controller. 49

[104] William Stallings. *Cryptography and network security - principles and practice (3. ed.)*. Prentice Hall, 2003. 51

[105] A. Akagic and H. Amano. A study of adaptable co-processors for cyclic redundancy check on an fpga. In *Field-Programmable Technology (FPT), 2012 International Conference on*, pages 119–124, Dec. 57

[106] Ignacio Algredo-Badillo, Claudia Feregrino-Uribe, René Cumplido, and Miguel Morales-Sandoval. Efficient hardware architecture for the aes-ccm protocol of the ieee 802.11i standard. *Comput. Electr. Eng.*, 36(3):565–577, May 2010. 65, 67, 96, 98

[107] Abdul Samiah, Arshad Aziz, and Nassar Ikram. A secure framework for robust secure wireless network (rsn) using aes-ccmp. In *fourth international Bhurban conference on applied sciences and technology*, June 2005. 67, 98

[108] Ho Yung Jang, Joon Hyoung Shim, Jung Hee Suk, In Cheol Hwang, and Jun Rim Choi. Compatible design of ccmp and ocb aes cipher using separated encryptor and decryptor for ieee 802.11i. In *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, volume 3, pages III – 645–8 Vol.3, may 2004. 67

[109] N. Smyth, M. McLoone, and J.V. McCanny. Wlan security processor. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 53(7):1506 – 1520, july 2006. 67, 98

[110] Duhyun Bae, Gwanyeon Kim, Jiho Kim, Sehyun Park, and Ohyoung Song. An efficient design of ccmp for robust security network. In *Information Security and Cryptology - ICISC 2005*, 2005. 67, 98

[111] embedded radio modules (802.11 b/g). datasheet, 2006. 67, 98

[112] the wimax security processor. datasheet, 2006. 67, 98

[113] Mpc8641d powerpc advanced mezzanine card. datasheet, 2006. 67, 98, 122

[114] The CBM Collaboration. Technical design report for the cbm(much). November 2014. 69, 72

[115] Rsa algorithm. 70, 76

[116] A Gabrielli, G. De Robertis, D. Fiore, F. Loddo, and A Ranieri. Architecture of a slow-control asic for future high-energy physics experiments at slhc. *Nuclear Science, IEEE Transactions on*, 56(3):1163–1167, June 2009. 71

[117] S Baron, J P Cachemiche, F Marin, P Moreira, and C Soos. Implementing the gbt data transmission protocol in fpgas. *CERN*. 71

[118] Chih-Hsien Lin, Chih-Ning Chen, You-Jiun Wang, Ju-Yuan Hsiao, and Shyh-Jye Jou. Parallel scrambler for high-speed applications. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, July. 73

[119] A. X. Widmer and P. A. Franaszek. A dc-balanced, partitioned-block, 8b/10b transmission code. 73

[120] Kenneth Andrews, Chris Heegard, and Dexter Kozen. A theory of interleavers. Technical report, 1997. 75, 79

[121] S. Safwat, E. E. D. Hussein, M. Ghoneima, and Y. Ismail. A 12gbps all digital low power serdes transceiver for on-chip networking. In *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, pages 1419–1422, May 2011. 77

[122] L. Antoni, R. Leveugle, and B. Feher. Using run-time reconfiguration for fault injection applications. *Instrumentation and Measurement, IEEE Transactions on*, 52(5):1468–1473, Oct 2003. 82

[123] Huiming Zhang and Bo Li. Characterizations of discrete compound poisson distributions. *Communications in Statistics - Theory and Methods*, 45(22):6789–6802, 2016. 82

[124] L. Liu, C. Liu, Y. Peng, and D. Liu. A design of fibre channel node with pci interface. In *2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1817–1822, May 2013. 82

[125] Christophe Giraud. Dfa on aes. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *Advanced Encryption Standard – AES*, volume 3373 of *Lecture Notes in Computer Science*, pages 27–41. Springer Berlin Heidelberg, 2005. 86

[126] S. D'Angelo, C. Metra, and G. Sechi. Transient and permanent fault diagnosis for fpga-based tmr systems. In *Defect and Fault Tolerance in VLSI Systems, 1999. DFT '99. International Symposium on*, pages 330–338, Nov 1999. 86

[127] TalG. Malkin, François-Xavier Standaert, and Moti Yung. A comparative cost/security analysis of fault attack countermeasures. In Luca Breveglieri, Israel Koren, David Naccache, and Jean-Pierre Seifert, editors, *Fault Diagnosis and Tolerance in Cryptography*, volume 4236 of *Lecture Notes in Computer Science*, pages 159–172. Springer Berlin Heidelberg, 2006. 86

[128] P. Maistri and R. Leveugle. Double-data-rate computation as a countermeasure against fault analysis. *Computers, IEEE Transactions on*, 57(11):1528–1539, 2008. 86

[129] Ting An, L.A. de Barros Naviner, and P. Matherat. A low cost reliable architecture for s-boxes in aes processors. In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2013 IEEE International Symposium on*, pages 155–160, 2013. 86, 90, 103

[130] M. Mozaffari-Kermani and A. Reyhani-Masoleh. Concurrent structure-independent fault detection schemes for the advanced encryption standard. *Computers, IEEE Transactions on*, 59(5):608–622, 2010. 87

[131] M. Mozaffari-Kermani and A. Reyhani-Masoleh. Reliable hardware architectures for the third-round sha-3 finalist grostl benchmarked on fpga platform. In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2011 IEEE International Symposium on*, pages 325–331, 2011. 87

[132] R. Banu and Tanya Vladimirova. Fault-tolerant encryption for space applications. *Aerospace and Electronic Systems, IEEE Transactions on*, 45(1):266–279, 2009. 87, 103

[133] Akashi Satoh, Takeshi Sugawara, Naofumi Homma, and Takafumi Aoki. High-performance concurrent error detection scheme for aes hardware. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 100–112. Springer Berlin Heidelberg, 2008. 87

[134] Sumio Morioka and Akashi Satoh. An optimized s-box circuit architecture for low power aes design. In BurtonS. Kaliski, çetinK. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 172–186. Springer Berlin Heidelberg, 2003. 87

[135] R. Mariani and G. Boschi. Scrubbing and partitioning for protection of memory systems. In *On-Line Testing Symposium, 2005. IOLTS 2005. 11th IEEE International*, pages 195–196, 2005. 91

[136] Ho Yung Jang, Joon Hyoung Shim, Jung-Hee Suk, In Cheol Hwang, and Jim Rim Choi. Compatible design of ccmp and ocb aes cipher using separated encryptor and decryptor for ieee 802.11i. In *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, volume 3, pages III–645–8 Vol.3, 2004. 98

[137] J. Mathew, A.M. Jabir, H. Rahaman, and D.K. Pradhan. Single error correctable bit parallel multipliers over gf(2m). *Computers Digital Techniques, IET*, 3(3):281–288, May 2009. 103

[138] A. Reyhani-Masoleh and M. A. Hasan. Low complexity bit parallel architectures for polynomial basis multiplication over gf(2m). *IEEE Transactions on Computers*, 53(8):945–959, Aug 2004. 103

[139] J. Mathew, J. Singh, A. M. Jabir, M. Hosseinabady, and D. K. Pradhan. Fault tolerant bit parallel finite field multipliers using ldpc codes. In *2008 IEEE International Symposium on Circuits and Systems*, pages 1684–1687, May 2008. 103

[140] M. Poolakkaparambil, J. Mathew, A. M. Jabir, D. K. Pradhan, and S. P. Mohanty. Bch code based multiple bit error correction in finite field multiplier circuits. In *2011 12th International Symposium on Quality Electronic Design*, pages 1–6, March 2011. 103

[141] Sang Phill Park, Dongsoo Lee, and K. Roy. Soft-error-resilient fpgas using built-in 2-d hamming product code. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 20(2):248–256, Feb 2012. 106, 112, 118, 119

[142] J. Heiner, B. Sellers, M. Wirthlin, and J. Kalb. Fpga partial reconfiguration via configuration scrubbing. In *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, pages 99–104, Aug 2009. 106

[143] Xilinx. Logicore ip soft error mitigation controller v3.4,product guide and 3gpp turbo decoder v4.0,product specification. 106, 119

[144] L. Frigerio, M.A. Radaelli, and F. Salice. Convolutional coding for seu mitigation. In *Test Symposium, 2008 13th European*, pages 191–196, May 2008. 106

[145] Priya Mathew, Lismi Augustine, Sabarinath G., and Tomson Devis. Hardware implementation of (63, 51) BCH encoder and decoder for WBAN using LFSR and BMA. *CoRR*, abs/1408.2908, 2014. 119

[146] Xilinx. Single error correction and double error detection, 2006. 119

Index