# A Compact and Fast FPGA Based Implementation of Encoding and Decoding Algorithm Using Reed Solomon Codes

Aqib Al Azad and Md Imam Shahed

***Abstract*—This paper presents a compact and fast Field Programmable Gate Array (FPGA) based implementation technique of Encoding and Decoding Algorithm using Reed Solomon (RS) codes, widely used in numerous applications ranging from wireless and mobile communications units to satellite links for correcting multiple errors especially burst-type errors. The main objective of this paper is to provide the reader with a deep understanding of the theory of RS code and encoding and decoding of the codes to achieve efficient detection and correction of the errors. The paper will cover the properties of RS code, RS Encoding and Decoding algorithm, simulation, synthesis and Verilog HDL based hardware implementation in FPGA device of the proposed RS Encoder and Decoder architecture. The results of fast and compact implementations of RS Encoder and Decoder architecture using Xilinx's Vertex and Spartan3E FPGA device are presented and analyzed. The design can also be synthesized to other FPGA architectures and is fully flexible & parameterized since block lengths and symbol sizes can be readily adjusted to accommodate a wide range of message sizes.**

***Index Terms*—FPGA, RS, Verilog, HDL.**

## I. Introduction

Reed Solomon codes [1] are systematic linear block codes and are an important sub class of nonbinary BCH codes. RS codes operate on the information by dividing the message stream into blocks of data, adding redundancy per block depending only on the current inputs. The symbols in RS coding are elements of a finite field or Galois Field (GF). GF arithmetic is used for encoding and decoding of reed Solomon codes. GF multipliers are used for encoding the information block. The multiplier coefficients are the coefficients of the RS generator polynomial. Encoding is achieved by affixing the remainder of a GF polynomial division into the message. This division is accomplished by aLinear Feedback Shift Register (LFSR) implementation [2] [3].

At the decoder, the syndrome of the received codeword is calculated using the generator polynomial to detect errors. Then to correct these errors, an error locator polynomial iscalculated. From the error locator polynomial, the location of the error and its magnitude is obtained.Consequently a correct codeword is obtained.

The results constitute simulation of Verilog codes of different modules of the Reed Solomon Encoder and Decoder in Xilinx. The design was successfully implemented in the Vertex and Spartan3E FPGA device. This paper describes a compact and fast design and implementation technique of RS Encoder and Decoder in FPGAs. It is organized as follows: In section II, basics of RS algorithm is discussed. Then in section III, the RS encoding algorithm is given. In Section IV, RS decoding algorithm is described briefly. Then in section V, design hierarchies for both RS Encoder and Decoder are shown and the basic blocks are described. Section VI will give experimental framework and results and section VII will give the conclusion based on our results.

## II. RS Basics

The basics of RS theory is as follows-

RS codes are generally represented as an RS ($n$, $k$), with m-bit symbols, where

Block Length: $n$

No. of Original Message symbols: $k$

Number of Parity Digits: $n - k = 2t$

The relationship between the symbol size, m, and the size of the codeword n, is given by

$$n = 2^m - 1$$

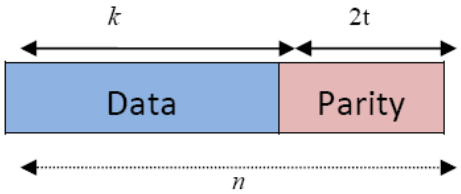


Fig. 1. The structure of a RS codeword [4].

The RS encoder provided at the transmitter end encodes the input message into a codeword and transmits the same through the channel. Noise and other disturbances in the channel may disrupt and corrupt the codeword. This corrupted codeword arrives at the receiver end (decoder), where it gets checked and corrected message is passed on to the receiver.

Reed Solomon (RS) codes, encoders and decoders are extremely powerful error correcting tools that increase transmission quality to a great extent. RS codes have the highest code rate of all binary codes. The number and type

Manuscript received December 22, 2012; revised April 1, 2013.

The authors are with the Department of Electrical Engineering and Computer Science, North South University, Dhaka,Bangladesh. He is now with the Department of Electrical and Electronic Engineering, Brac University, Dhaka, Bangladesh (e-mail:aqibazad@ymail.com)

of errors that can be corrected depends on the characteristics of the RS code used.

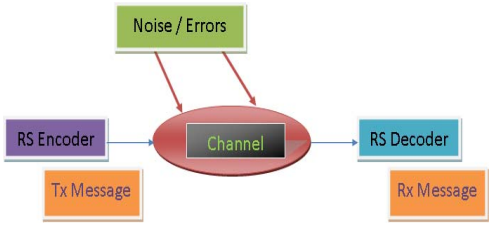


Fig. 2. RS protected communication channel.

## III. RS Encoder

The Reed Solomon Encoder reads in $k$ data symbols computes the $n - k$ symbols, append the parity symbols to the k data symbols for a total of n symbols. The encoder is essentially a 2t tap shift register where each register is m bits wide. The multiplier coefficients are the coefficients of the RS generator polynomial. The general idea is the construction of a polynomial, the coefficient produced will be symbols such that the generator polynomial will exactly divide the data/parity polynomial. [5]

The transmitted codeword is systematically encoded and defined in as a function of the transmitted message $m(x)$, the generator polynomial $g(x)$ and the number of parity symbols 2t as given below.

$$c(x) = m(x) \times 2t + m(x) \bmod g(x) \quad (1)$$

where, $g(x)$ is the generator polynomial of degree 2t and given by,

$$g(x) = (x+\alpha^{i})(x+\alpha^{i+1})\ldots(x+\alpha^{i+2t-2})(x+\alpha^{i+2t-1}) \quad (2)$$

Table 1 contains a list of some primitive polynomials.

TABLE I: Some Primitive Polynomials

| $m^a$ | | m | |
|---|---|---|---|
| 3 | $1+X+X^3$ | 14 | $1+X+X^6+X^{10}+X^{14}$ |
| 4 | $1+X+X^4$ | 15 | $1+X+X^{15}$ |
| 5 | $1+X^2+X^5$ | 16 | $1+X+X^3+X^{12}+X^{16}$ |
| 6 | $1+X+X^6$ | 17 | $1+X^3+X^{17}$ |
| 7 | $1+X^3+X^7$ | 18 | $1+X^7+X^{18}$ |
| 8 | $1+X^2+X^3+X^4+X^8$ | 19 | $1+X+X^2+X^5+X^{19}$ |
| 9 | $1+X^4+X^9$ | 20 | $1+X^3+X^{20}$ |
| 10 | $1+ X^3+X^{10}$ | 21 | $1+X^2+X^{21}$ |
| 11 | $1+X^2+X^{11}$ | 22 | $1+X+X^{22}$ |
| 12 | $1+X+X^4+X^6+X^{12}$ | 23 | $1+X^5+X^{23}$ |
| 13 | $1+X+X^3+X^4+X^{13}$ | 24 | $1+X+X^2+X^7+X^{24}$ |

The Encoder architecture shows that one input to each multiplier is a constant field element, which is a coefficient of the polynomial $g(x)$. For a particular block, the informationpolynomial $M(x)$ is given into the encoder symbol by symbol. These symbols appear at the output of the encoder after a desired latency, where control logic feeds it back through an adder to produce the related parity. This process continues until all of the k symbols of $M(x)$ are input to the encoder. During this time, the control logic at the output enables only the input data path, while keeping the parity path disabled. With an output latency of about one clock cycle, the encoder outputs the last information symbol at (k+1)th clock pulse. Also, during the first k clock cycles, the feedback control logicfeeds the adder output to the bus. After the last symbol has been input into the encoder (at the kth clock pulse), a wait period of at least n-k clock cycles occurs. During this waiting time, the feedback control logic disables the adder output from being fed back and supplies a constant zero symbol to the bus. Also, the output control logic disables the input data path and allows the encoder to output the parity symbols ($k$+2th to $n$+1th clock pulse). Hence, a new block can be started at the $n$+1th clock pulse [6].

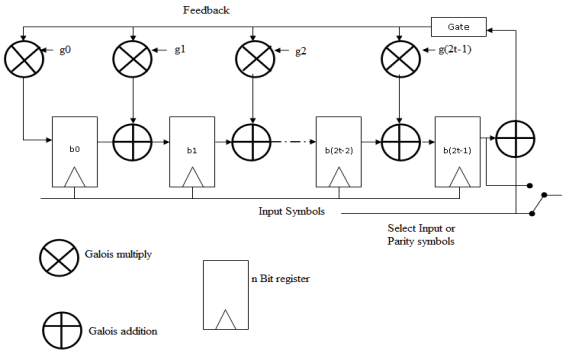


Fig. 3. Architecture of RS encoder.

## IV. RS Decoder

When a RS Decoder corrects a symbol, it replaces the incorrect symbol with the correct one, whether the error was caused by one bit being corrupted or all of the bits being corrupted. Thus, if a symbol is wrong, it might as well be wrong in all of its bit positions. This gives RS codes tremendous burst-noise advantages over binary codes [7].

The decoding procedure for Reed- Solomon codes involves determining the locations and magnitudes of the errors in the received polynomial $R(x)$.Locations are those powers of $x$ ($x^2$, $x^3$, and others) in the received polynomials whose coefficients are in error. Magnitudes of the errors are symbols that are added to the corrupted symbol to find the original encoded symbol. These locations and magnitudes constitute the error polynomial. Also, if the decoder is built to support erasure decoding, then the erasure polynomial has to be found. An erasure is an error with a known location. Thus, only the magnitudes of the erasures have to be found for erasure decoding. A RS ($n$, $k$) code can successfully correct as many as 2t = n-k erasures if no errors are present. With both errors and erasures present, the decoder can successfully decode if $n$-$k \geq$ 2t +$e$, where t is the number of errors, and e is the number of erasures [8].

The first step is to calculate the syndrom values [9] from the received codeword. These are then used to find the coefficients of the error locator polynomial $\Lambda_1$ ....$\Lambda_v$ and the error magnitude polynomial $\Omega_0$ ....$\Omega_{v-1}$ using the Euclidean algorithm [10]. The error locations are figured out by the Chien search [11] and the error magnitudes are calculated

using Forney's method [12]. The error magnitude vector *Y* comes out of the Chien/Forney block in reverse order, so it is passed through a FIFO block before it is added to the received codeword *R*(*x*). As these calculations involve all the symbols of the received code word, it is necessary to restore the message until the results of the calculation are available. Then, to correct the errors, each error value is added (modulo 2) to the symbol at the appropriate location in the received codeword [13].

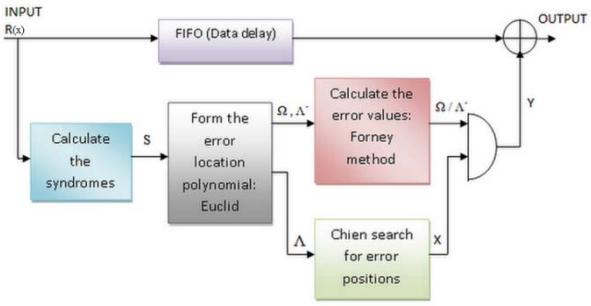


Fig. 4. General architecture of RS decoder [11].

## V. Implementation Details of Basic Blocks

### A. Design hierarchy

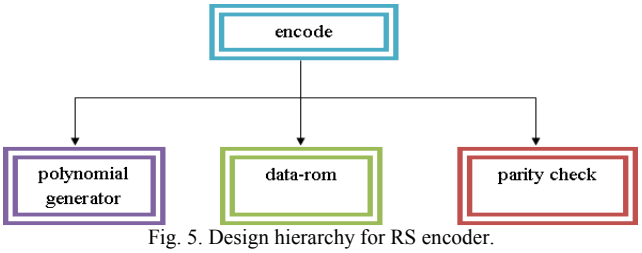


Fig. 5. Design hierarchy for RS encoder.

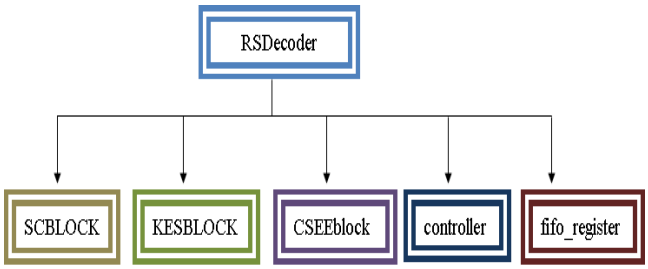


Fig. 6. Design hierarchy for RS decoder.

### B. Block description of RS Encoder

*Polynomial generator*: For the encoding process the generator polynomial is,

$$g(x) = g_0 + g_1(x) + g_2(x^2) + \cdots + g_{2t-1}(x^{2t-1}) + x^{2t} \quad (5)$$

*Data-rom:* We provide a ROM as the message to encode.

*Parity check:* The parity check block allows the encoder to output parity symbols.

### C. Block Description of RS Decoder

- *SCBlock:* At the end of received word, all cells store the syndrome values and Syndrome Computation (SC) block sets its flag high if one or more syndrome values are not zero.
- *KESBlock:*The Key equation solver (KES) block provides two polynomial -error locator polynomial and error magnitude polynomial.
- *CSEEBlock*: Chien Search and Error Evaluator (CSEE) block *identifies* error location while computes its error magnitude.
- FIFO register: FIFO register stores received word *symbols*.To match the order of the bytes in error vector and received codeword FIFO is applied as the error vector is produced in the reverse order of the received codeword.
- *Controller*: The controller provides synchronization among all four modules -SC, KES, CSEE and FIFO Registers.

## VI. Experimental Framework and Results

In this section we describe the design procedure and the architecture of RS Encoder and Decoder. Fig. 7 shows the different stages of our design. The Verilog model was synthesized with Xilinx Software targeted for Spartan 3E (XC3S1200E) and Vertex 5 (XC5VLX50)device and simulated with Modelsim. FPGA technology was chosen because it provides some important advantages over general purpose processors and application specific integrated circuits (ASICs).

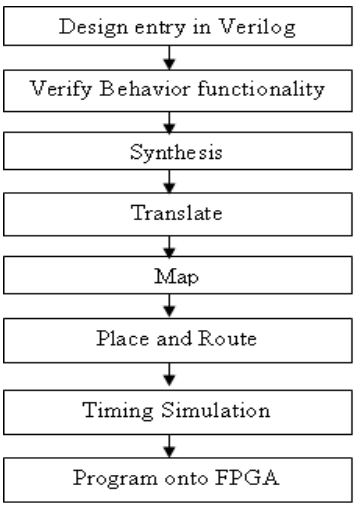


Fig. 7. Implementation flow.

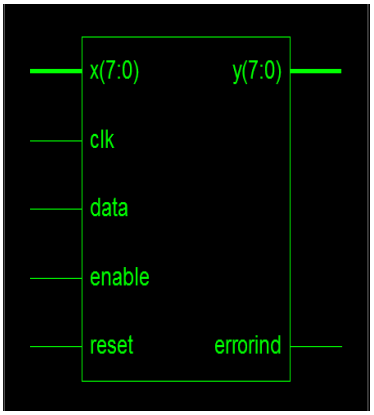


Fig. 8. RTL schematic of RS encoder.

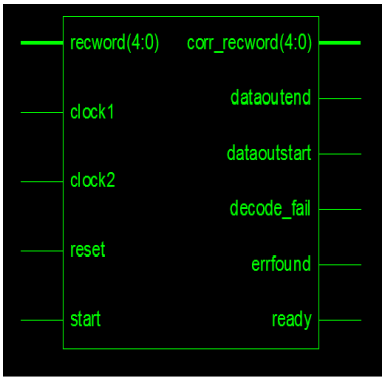


Fig. 9. RTL schematic of RS decoder.

The RTL architecture of RS Encoder and Decoder is shown in Fig. 8 and Fig. 9 respectively.

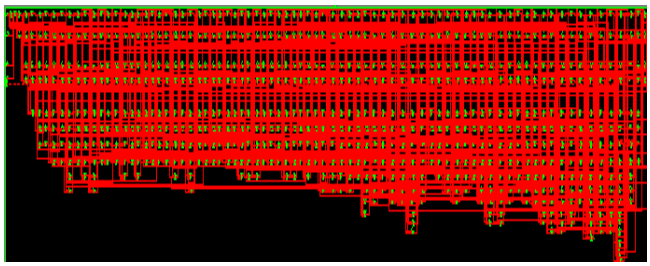

Fig. 10. Technology schematic of RS encoder.

Fig. 10 and Fig. 11 illustrate the implemented components inside the chip. Additionally, the interconnections of the components are shown.

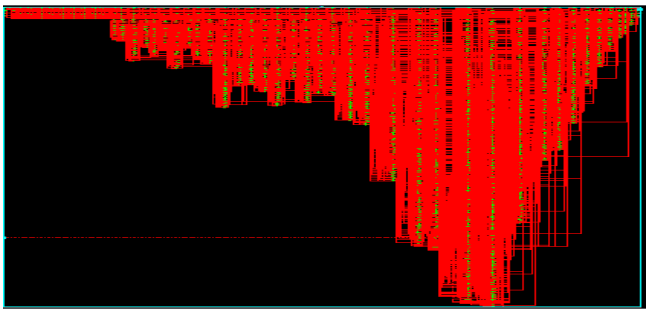

Fig. 11. Technology schematic of RS decoder

Table II and Table III show synthesis results of RS Encoder and RS Decoder for Spartan 3E and Vertex5 device respectively.

TABLE II: SYNTHESIS RESULT FOR SPARTAN 3E

| | **RS Encoder** | | **RS Decoder** | |
|---|---|---|---|---|
| | Spartan 3E (XC3S1200E, package fg320,speed grade-5) | | Spartan 3E (XC3S1200E, package fg320,speed grade-5) | |
| Logic Utilization | Used | Utilization | Used | Utilization |
| Number of Slices | 251 | 2% | 775 | 8% |
| Number of Slice Flip Flops | 274 | 1% | 517 | 2% |
| Number of 4 input LUTs | 461 | 2% | 1,459 | 8% |
| Number of bonded IOBs | 21 | 8% | 19 | 7% |
| Number of GCLKs | 1 | 4% | 2 | 8% |

TABLE III: SYNTHESIS RESULT FOR VERTEX 5

| | **RS Encoder** | | **RS Decoder** | |
|---|---|---|---|---|
| | Vertex 5 (XC5VLX50, package ff676,speed grade -1 ) | | Vertex 5 (XC5VLX50, package ff676,speed grade -1 ) | |
| Logic Utilization | Used | Utilization | Used | Utilization |
| Number of Slice Registers | 265 | 1% | 517 | 1% |
| Number of Slice LUTs | 342 | 1% | 1,010 | 3% |
| Number of fully used LUT-FF pairs | 265 | 77% | 511 | 50% |
| Number of bonded IOBs | 21 | 4% | 19 | 4% |
| Number of BUFG/BUFGCTRLs | 1 | 3% | 2 | 6% |

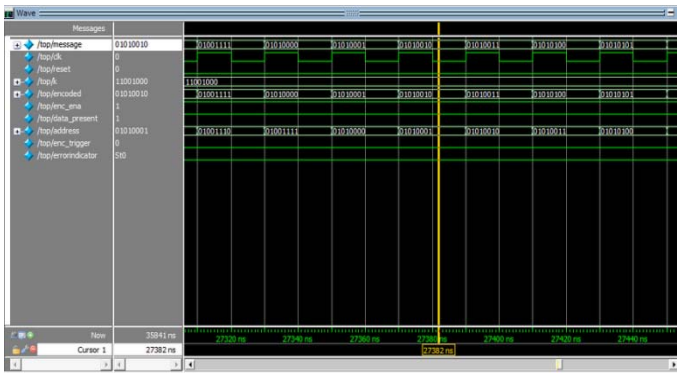

Fig. 12. Output of RS encoder

Fig. 12 and Fig. 13 show the simulation waveform of RS Encoder and Decoder. For RS Encoder, the simulation result verifies our design. The encoder has an error indicator that points out the presence of errors in the data.

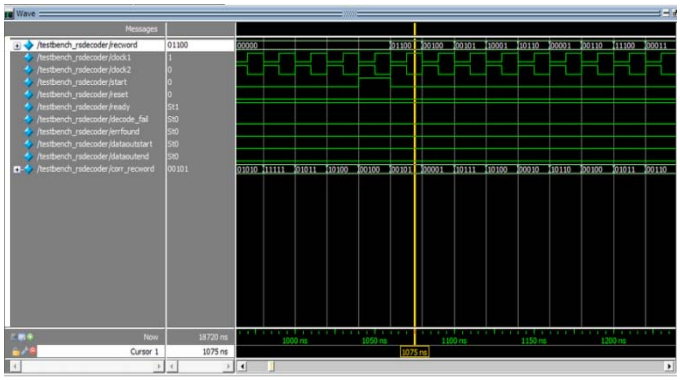

Fig. 13 Output of RS decoder

For RS Decoder the simulation waveform contains many different received word vectors. When the number of errors is greater than correction capability, decoder will resume decoding failure in that case at the end of outputted received word. As decoding failure is detected after all elements in GF have been evaluated , so uncorrectable word is summed with error values and it gives some wrong message at the output only in the case where the number of errors are greater than the decoder correction capability.

## VII. CONCLUSIONS

When a RS Decoder corrects a symbol, it replaces the

incorrect symbol with the correct one, whether the error was caused by one bit being corrupted or all of the bits being corrupted. Thus, if a symbol is wrong, it might as well be wrong in all of its bit positions. This gives RS codes tremendous burst-noise advantages over binary codes [10].

Here Error detection and correction techniques have been used which are essential for reliable communication over a noisy channel. A compact and fast hardware implementation technique of Reed Solomon Encoding and Decoding algorithm were presented. The design was implemented in real hardware with Spartan 3E and Vertex5 FPGA. The results demonstrate that the Reed Solomon codes are very efficient for the detection and correction of burst errors. As mentioned above, RS codes are based on the finite fields so they can be extended or shortened. Reed Solomon codes provide a wide range of code values that can be chosen to optimize performance. RS codes are used significantly in Wireless Communication (mobile phones, microwave links), Deep Space and Satellite Communications Networks (CCSDS), mass storage devices (hard disk drives, DVD, barcodes), digital TV, digital video broadcasting (DVB), and Broadband Modems (ADSL, VDSL, SDSL, HDSL etc). Technologies are becoming smarter and compact day by day, so we hope our work will add new dimension in that trend.

## ACKNOWLEDGMENT

At first the author would want to thank Allah for giving them the opportunity to do this work. They sincerely thank to their friends and all those whoever has helped them either directly or indirectly in the completion of this work.

## REFERENCES

[1] I. S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," *SIAM Journal of Applied Mathematics*, vol. 8, pp. 300–304.

[2] R. J. McEliece, *Finite Fields for Computer Scientists and Engi*neers, Boston, MA: Kluwer Academic, 1987.

[3] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Englewood Cliffs, N.J.: Prentice-Hall, 1994.

[4] M. Kaur and V. Sharma, "Study of Forward Error Correction using Reed—Solomon Codes," *International Journal of Electronics Engineering*, vol. 2, pp. 331 – 333, 2010.

[5] *Reed-Solomon (RS) Coding Overview*, VOCAL Technologies, Ltd., Rev, 2010.

[6] S. Kaur "VHDL Implementation of Reed-Solomon code," Thesis, Thapar Institute of Engg, 2006.

[7] Sklar, *Digital Communications: Fundamentals and Applications*, 2nd ed., Upper Saddle River, NJ: Prentice-Hall, 2001. pp. 441.

[8] S. S. Shah, S. Yaqub, and F. Suleman, "Self-correcting Codes Conquer Noise Part 2: Reed-Solomon Codecs," *EDN*, Mar. 15, 2001, pp. 107-120.

[9] C. K. P. Clarke, "Reed-Solomon error correction," BBC R&D White Paper, WHP 031, July 2002.

[10] M. Purser, *Introduction to Error Correcting Codes*, Artech House, Boston-London, 1995.

[11] R. T. Chien. "Cyclic Decoding Procedure for the Bose- Chaudhuri-Hocquenghem Codes," *IEEE Transactions on Information Theory*, pp. 357-363, Oct. 1964.

[12] G. D. Forney. "On Decoding BCH Codes." *IEEE Transactions on Information Theory*, Volume IT-11, pp. 549-557, Oct. 1965.

[13] R. Berlekamp, R. E. Peile, and S. P. Pope. "The Application of Error Control to Communications." *IEEE Communications Magazine*, vol. 25, no. 4, pp. 44-57, Apr. 1987.

**Aqib Al Azad** was born in Dhaka, Bangladesh. He received his BS degree in Electronics and Telecommunication Engineering from North South University with the distinction of Summa Cum Laude. He is currently enrolled in a Master's program in Electrical and Electronic Engineering at Brac University. He currently works as a Lab Instructor in the Department of Electrical Engineering and Computer Science at North South University. His research interests include Mobile Communication, Wireless Communication, Reconfigurable Computing, and SoC design.