Implementation of Power Optimized 2 Bit Polar Decoder Architectures

Badugu Neelima[1], A.Vijaya Lakshmi[2]

[1]PG Scholar, CEC, ECE Department, AP, India
[2]Assistant Professor, CEC, ECE Department, AP, India

ABSTRACT:

There are so many different error correcting codes are existed for the design environment correction. Among those polar codes, as the first provable capacity-achieving codes over binary-input discrete memoryless channel (B-DMC). However, for polar codes, the long latency is a bottleneck for designing. In this paper, a mitigation technique is proposed and adopted to avoid latency drawbacks in polar codes. Maintaining less prone to errors in circuits is very important to avoid data corruption in the system. This paper presents a new built-in 2-D Hamming product code (2–D HPC) scheme to provide reliable operation of polar codes in hostile operating environment applications such as space.

Error correction method which in used in this paper is 2-D HPC which can enhances the reliability. By the Simulation waveforms the functionality of the 2-D HPC method which is used for polar codes should be understand with the clear sense. The HDL code is developed with VERILOG language and the synthesis and simulation is done by XILINX ISE EDA Tool.

Key words: *Soft errors, polar codes, Xilinx, Verilog.*

1. INTRODUCTION

The effective communication importance is immeasurable in the world of business and also in personal life. From a perspective point of business, effective communication must be absolute, as it is generally accounts for the variation between victory and failure or return and loss. It is obvious that efficient business communication is significant to the unbeaten operation of up to date enterprise. Every business person desires to recognize the basics of efficient communication. It has been confirmed that unfortunate communication reduces excellence, weakens yield, and it ultimately leads to anger and a lack of belief among folks within the organization. The communication process is the channel in realizing efficient communication. It is from end to end the communication process that, the distribution of a common meaning between the dispatcher and the recipient takes place. Effective communication leads to understanding.

The communication process will begin at the dispatcher and ends at the recipient. In electronics and computing, a soft error is a type where a signal has been wrong. A soft error will be a signal which would be incorrect, but is not assumed to simply such a mistake or breakage. A soft error will not damage a system's hardware, the only damage is to the data that is being processed. These soft errors are classified into 2 types which are chip level and system level.

In the mitigation of Soft error a developer can challenge to reduce the rate of soft errors by sensible device design, choosing a ideal semiconductor, package and also a substrate resources, and the right device geometry. Often, however, this is restricted by the need to trim down device size and voltage, to increase operating speed and also to diminish the power dissipation. The vulnerability of devices to upsets will be demonstrated in the industry using the JEDEC JESD-89 standard. One method that can be used to diminish the soft error rate within the digital circuits is recognized as radiation hardening. Detecting soft errors has been a work for addressing the soft errors in processor and memory property using both hardware and software methods. Quite a few research hard work addressed soft errors by propose error detection and recovery via hardware-based redundant multi-threading. These approach using unique hardware to reproduce an application execution to recognize errors in the output, which improved hardware design complexity and cost together with high performance overhead.

Developers can decide to allow that soft errors will occur, and design systems with suitable error detection and correction used to improve kindly. Typically, a semiconductor memory design may use forward error correction, incorporate laid off data into each word for create an error correcting code. Instead, roll-back error correction can be used, detect the soft error with an error-detecting code such as parity, and rewriting correct data from a different source. This type of method is in general used for write-through cache memories. Soft errors in logic circuits are occasionally detected and corrected using the technique of fault tolerant design. Which are often take account of the use of disused circuitry or computation of data, and usually come at the cost of circuit area, decrease performance, and/or superior power consumption.

2. POLAR CODES AND ITS PRELIMNARIES

The history of transferring the data through the channel began with Shannon's information theory. According to the Golay work and Hamming, the main stream of linear codes established in the initial days of coding theory in which "error correction" codes in the logic that their goal is to spot-on errors made by the channel. The channel was universally assumed to be a Binary Symmetric Channel (BSC). Studying of error correction codes concluded with the invention of Reed-Solomon codes. In 1960, which are Maximum Distance Separable (MDS) over non-binary fields and hence are guaranteed to correct or else to detect the largest number of errors possible for a given code length and dimension.

Additional chronicles in channel coding was prepared by Gottfried Ungerboeck by connecting coding to modulation for convolutional codes. In 1993, Claude Berrou and co-authors surprised the coding research community in by designing a coding system known as "turbo codes" that accomplished a quantum spring in the recital of codes over common channels. Nowadays it is well-known that both turbo and LDPC codes can be viewed as sparse codes on graphs. As it prolongation a lot of possessions are shared and the work is imitation vice versa. Especially some minute deviations between LDPC or turbo codes may tilt the balance towards one or the other in specific applications.

Shannon, in his seminal work, he highlighted the troubles of data storing and communicating and also resolute its fundamental barriers. They provided a mathematical framework for understanding the problems systematically which have lead to the advances in the previous years. The generality of the approach makes us to study even modern day scenarios, like mobile communication or adhoc-networks. The fundamental model explained by Shannon will consists of a source, which generates the information, a sink which receives the information, and a channel, which models the physical transfer of information.

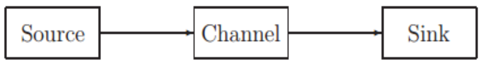


Fig.1: communication model.

A crucial by-product work of Shannon's will be splitting the source and channel coding operations as shown in Fig. 1 does not incur any loss in performance (if measured only with the achievable rates).The source coding module had compresses the source data to as low a rate as possible given a desired upper limit on the distortion whose simplified communication model is shown in fig 2. This part adds redundancy to protect the data against noise. At the decoder, we initially execute the channel decoding to eliminate the noise by causing from the transmission. Then the source is decoded, i.e., we restructure the source data from its compacted illustration.

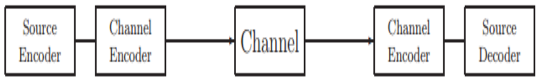


Fig.2: simplified communication model

Processing Element (PE) For F and G Nodes

As shown in Fig.3, The propagated log likelihood ratios (LLR) values are calculated based on the f and g nodes which are used in the every stage of the decoding procedure. The main purpose of introducing the unified processing elements (PEs) is to implement the f and g nodes. To convert one form to another means from sign-magnitude to 2's complement form and vice versa S2C & C2S blocks are used, which is shown in fig 4. Along with that, the blocks subtraction and adder are used to obtain the results whose blocks are shown in fig 4. For which the corresponding sum and difference are obtained from the partial sum generator (PSG) block. To determine the LLR output, the control signal is used at the PE output end and propagated to the next stage.

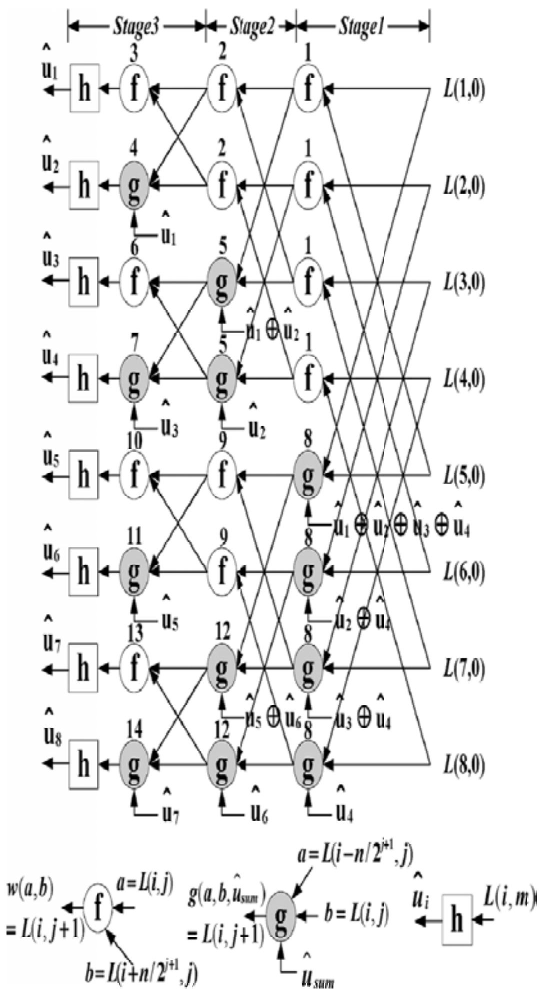


Fig.3: Decoding procedure

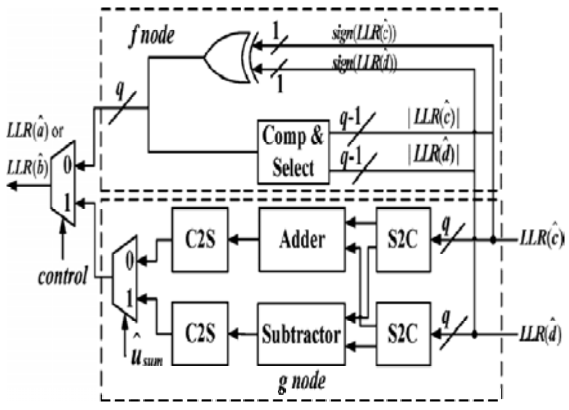


Fig.4: The architecture of PE.

POLAR ENCODING AND DECODING

With an efficient construction approach, the reliability of decoded bits will be polarized based on their different positions at the source data. An efficient polar-based encoder shown in Fig 5 can be constructed

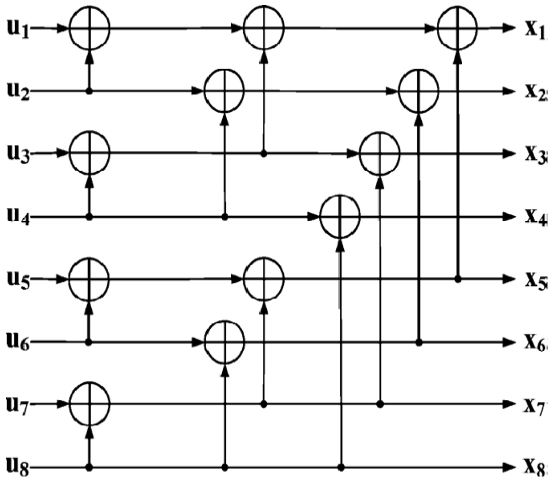


Fig.5: polar encoder with n=8.

Based on the following principles, Sending required message bits at good positions, which can strongly guarantee the consistency of communication. Referring fixed 0 at bad positions, since later the communication any decoded bits at these bad positions are extremely defective. In, those 0 bits are called frozen bits since these are fixed and their positions are known at both the encoder and the decoder.

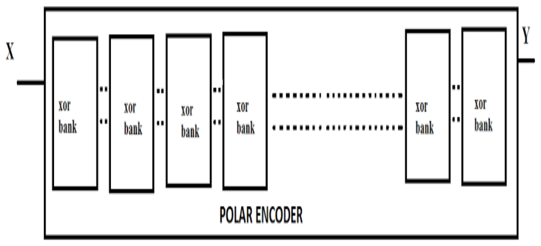


FIG.6: BLOCK DIAGRAM FOR POLAR ENCODER

Here in the polar encoding block which is shown in fig 6 is a series of XOR bank modules. Each XOR bank consists of an XOR operation between the corresponding inputs and output bits. The main intension of using XOR operation is to detect about the error presence in the information. The XOR operation results 1 if both the inputs are differ and it results 0 if inputs are same.

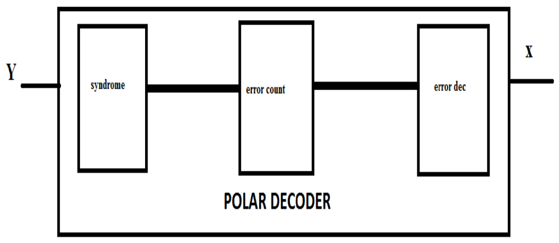


FIG.7: BLOCK DIAGRAM FOR POLAR DECODER

The polar decoder block which is shown in fig 7 mainly consists of syndrome calculation, error counting and the error detection blocks. The main function of the syndrome block is to perform the XOR operation to ensure the correctness of data transmission through the channel in the real case scenario. Means the syndrome calculation block gives the data perfectness regarding the data is error or error free. The generated syndrome are taken as inputs by the error count block and performs the corresponding error bits counting action to ensure how many bits are transmitted successfully. The error detector block detects the error bits and corrects the corresponding vector bits.

3. HAMMING procedure.

In this paper the polar codes are implemented based on the hamming codes extension work. We focus on their performance at low error rates, which is essential for wireless multimedia applications. We present the basis and the complete set of techniques which have allowed one which analytically evaluate this performance without resorting to extremely long simulations. The key to the Hamming Code calculation is extra parity bits usage to identify an error. For that the code word has to be created as follows:

1. In the code word identify the parity bit positions based on the powers of two. Like 1, 2, 4, 8, 16, 32, 64....
2. Obviously the remaining bit positions are for the data which has to be encoded. So the Positions are nothing but 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, etc....
3. Now calculate the parity which describing the sequence of bits which are alternately skips and checks. The corresponding positions are described below.
   Position 1: check 1 bit, skip 1 bit, check 1 bit, skip 1 bit, etc. (1,3,5,7,9,11,13,15,...)
   Position 2: check 2 bits, skip 2 bits, check 2 bits, skip 2 bits, etc. (2,3,6,7,10,11,14,15,...)
   Position 4: check 4 bits, skip 4 bits, check 4 bits, skip 4 bits, etc. (4,5,6,7,12,13,14,15,20,21,22,23,...)
   Position 8: check 8 bits, skip 8 bits, check 8 bits, skip 8 bits, etc. (8-15,24-31,40-47,...)
   Position 16: check 16 bits, skip 16 bits, check 16 bits, skip 16 bits, etc. (16-31,48-63,80-95,...)
   Position 32: check 32 bits, skip 32 bits, check 32 bits, skip 32 bits, etc. (32-63,96-127,160-191,...)
   etc.
4. Based on the total number of ones the parity bit has to be set as 1 or 0. If total number of one's is odd the set parity as 1 and set parity as 0 if even number of one's are there. Here is an example:

A byte of data: 10011010

Create the data word, leaving spaces for the parity bits: _ _ 1 _ 0 0 1 _ 1 0 1 0 for each parity bit Calculate the parity (a? represents the bit position being set):

- Position 1 checks bits 1,3,5,7,9,11: **?** _ **1** _ **0** 0 **1** _ **1** 0 **1** 0. Even parity so set position 1 to a 0: **0** _ **1** _ **0** 0 **1** _ **1** 0 **1** 0
- Position 2 checks bits 2,3,6,7,10,11: 0 **? 1** _ 0 **0 1** _ 1 **0 1** 0. Odd parity so set position 2 to a 1: 0 **1 1** _ 0 **0 1** _ 1 **0 1** 0
- Position 4 checks bits 4,5,6,7,12: 0 1 1 **? 0 0 1** _ 1 0 1 **0**. Odd parity so set position 4 to a 1: 0 1 1 **1 0 0 1** _ 1 0 1 **0**
- Position 8 checks bits 8,9,10,11,12: 0 1 1 1 0 0 1 **? 1 0 1 0**. Even parity so set position 8 to a 0: 0 1 1 1 0 0 1 **0 1 0 1 0**
- Code word: 011100101010.

4. Error Detection and Correction

A representative example of scrubber operation block diagram is shown in Fig 8. This algorithm closely follows the Xilinx recommendations laid out. The most important operations in this diagram were error detection and error correction. The correction of a detected error is an involved operation. The designer has two basic alternatives. The first alternative is used to the redundant or parity bits of the error detection and correction code to infer the location of the error within a frame and correct it. The second alternative in case of the redundant bits was used only to allow error detection but not error correction is to fetch a golden version of the frame's information from a safe storage and use it to overwrite the corrupted frame. Fetching a golden frame for scrubbing will have the disadvantage of requiring a local and safe (not susceptible to SEU) storage, increasing the overall system's complexity. However, it can possible to ease the complexity by using the storage present in the FPGA's hardware platforms to store the initial bit stream. Programmable Read-Only Memory (PROM) is generally used for this purpose [9].

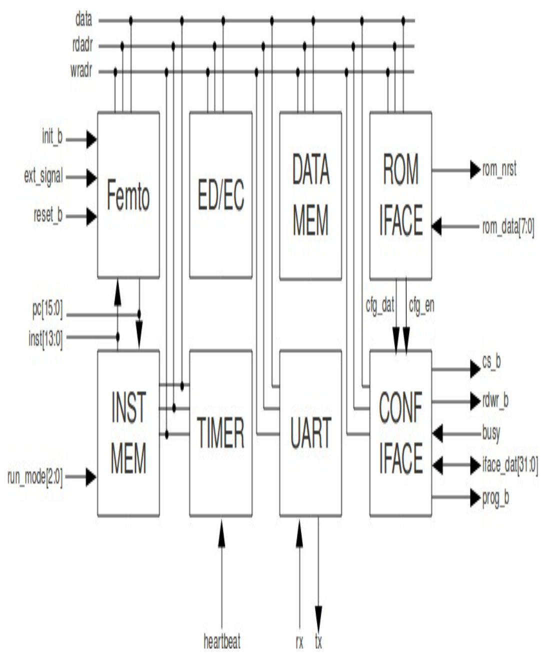
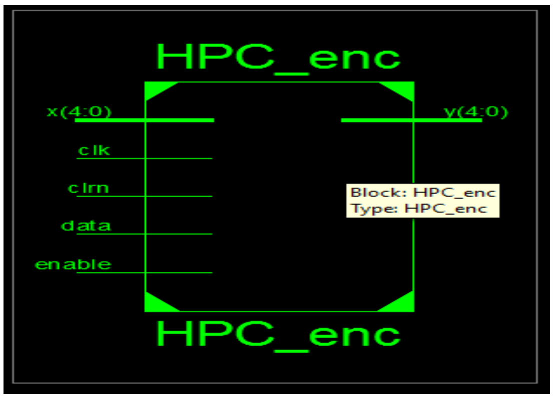


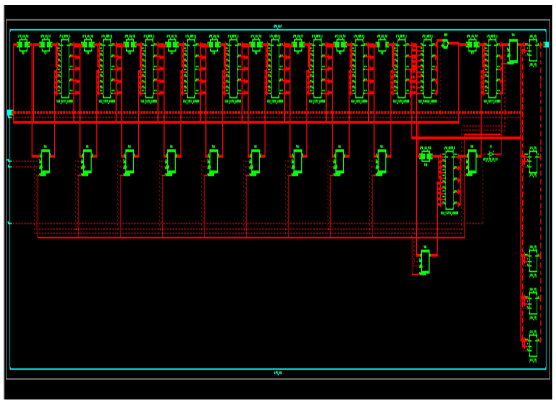
Fig.8: Scrubber block diagram

5 RESULTS

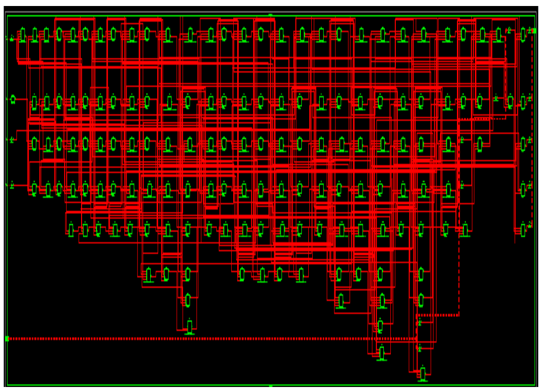
HPC_enc BLOCK RTL schematic



Internal structure of RTL schematic



Technological schematic of HPC_enc BLOCK



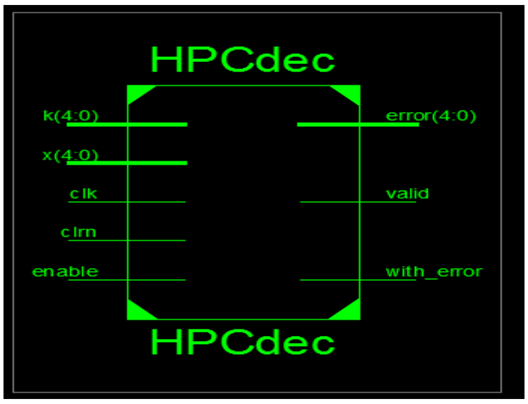
Design summary of HPC_enc BLOCK with xc3s500e—fg320

Device Utilization Summary (estimated values)

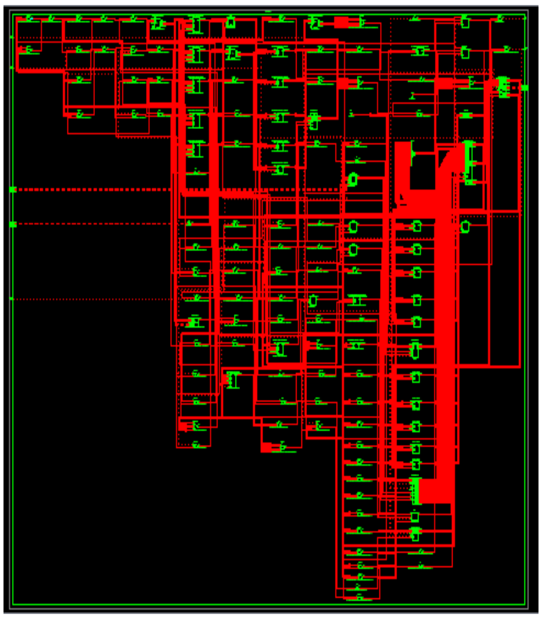
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	47	4656	1%
Number of Slice Flip Flops	60	9312	0%
Number of 4 input LUTs	91	9312	0%
Number of bonded IOBs	14	232	6%
Number of GCLKs	1	24	4%

The Number of 4 input LUTs used are 91 out of 9312. Based on this the Total power consumed is 0.000742w. The Delay and Maximum Frequency is 7.062ns and 231.750MHz.

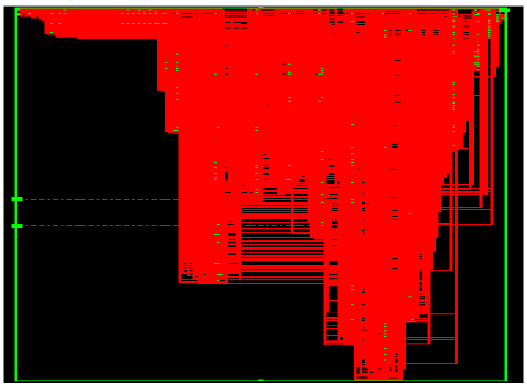
HPC_dec BLOCK RTL schematic



Internal structure of RTL schematic



Technological schematic of HPC_dec BLOCK



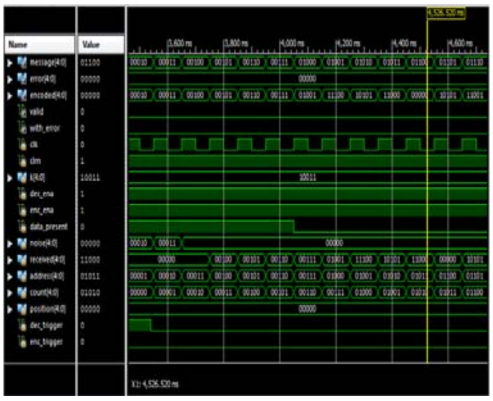
Design summary of HPC_dec BLOCK

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	651	4656	13%
Number of Slice Flip Flops	522	9312	5%
Number of 4 input LUTs	1191	9312	12%
Number of bonded IOBs	20	232	8%
Number of GCLKs	1	24	4%

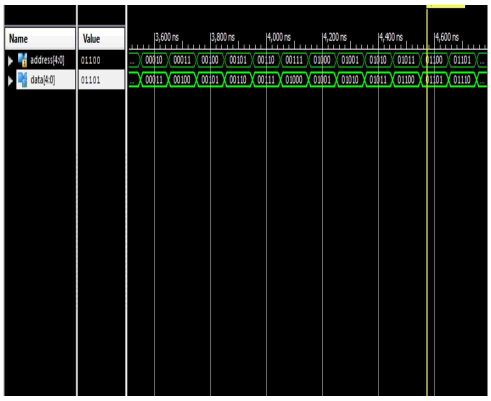
The Number of 4 input LUTs used are 1191 out of 9312. Based on this the Total power consumed is **0.00972w**. The Delay and Maximum Frequency is 9.744ns and 105.686MHz.

Results

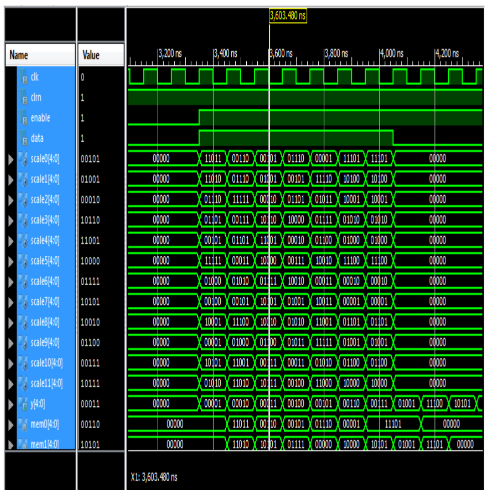
Top module waveform



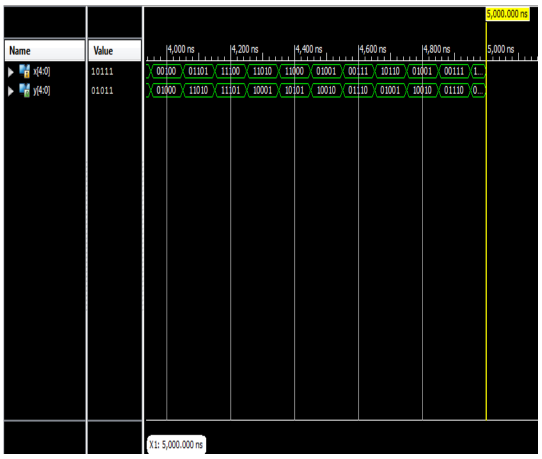
X0 module waveform:



X1 module waveform:



M0 module waveform:



CONCLUSION

In this paper, we propose an efficient built-in 2-D Hamming product code (2-D HPC) scheme for realizing polar codes which provides large performance improvement in error detection and correction. Based on the investigation of existing soft error mitigation techniques of SRAM arrays and their limitations in the state-of the- art FPGAs, we address the need of a new 2-D HPC error correction (EC) scheme for FPGA configuration data which should consume less gate count when compare with the existing paper. So obviously due to reduction of gate count the consumed power is also reduced. Implementation of the proposed scheme includes a novel SRAM architecture which provides a single-cycle bitwise column read of the contents of memory array. The functionality is verified and synthesis is carried out using XILINX ISE 12.3i. From the waveforms it should be conclude that how the noise is induced, corrected and how it is detected.

REFERENCES

[1] "*Virtex-5 FPGA User Guide,*" Xilinx Corp., San Jose, CA [Online]. Available: http://www.xilinx.com,

[2] A. Leseaet al., "The Rosetta experiment: Atmospheric soft error rate testing in different technology FPGAs, IEEE Trans. Device Mater. Rel., vol. 5, no. 3,

[3] E. Fuller et al., "Radiation testing update, SEU mitigation, and availability analysis of the Virtex FPGA for space reconfigurable computing," inProc. MAPLD.

[4] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies, "IEEE Trans. Device Mater. Rel.., vol. 5, no. 3, pp. 305–316.

[5] A. Lesea *et al.*, "The Rosetta experiment: Atmosperic soft error rate testing in differing technology FPGAs," *IEEE Trans. Device Mater.*

[6] E. Fuller *et al.*, "Radiation testing update, SEU mitigation, and availability analysis of the Virtex FPGA for space reconfigurable computing," in *Proc. MAPLD*, 2000.

[7] J. F. Ziegler, "Terrestrial cosmic ray intensities," *IBM J. Res. Develop.*, vol. 42, no. 1, pp. 117–139, 1998.

[8] E. Normand, "Single event effects in avionics," *IEEE Trans. Nucl. Sci.* vol. 43, no. 2, pp. 461–474, Apr. 1996.

[9] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Trans. Device Mater. Rel..*, vol. 5, no. 3, pp. 305–316, Sep. 2005.

[10] H. Asadi *et al.*, "Analytical techniques for soft error rate modeling and mitigation of FPGA-based designs," *(VLSI) Syst.*, vol. 16, no. 12,

[11] C. Carmichael *et al.*, "SEU mitigation techniques for Virtex FPGAs in space applications," in *Proc. MAPLD*, 1999.

[12] K. Chapman *et al.*, "SEU strategies for Virtex-5 devices," Xilinx Application

[13] M. Garvie *et al.*, "Scrubbing away transients and jiggling around the permanent: Long survival of FPGA systems through evolutionary selfrepair,"