DOI: 10.21917/ijme.2018.0112

# FPGA IMPLEMENTATION OF SSPA DECODER

**A. Rajagopal[1], K. Karibasappa[2] and K.S. Vasundara Patel[3]**
[1]*Department of Electronics and Communication Engineering, Dayananda Sagar College of Engineering, India*
[2]*Department of Electronics and Communication Engineering, Dayananda Sagar Academy of Technology and Management, India*
[3]*Department of Electronics and Communication Engineering, BMS College of Engineering, India*

***Abstract***

***SSPA decoder is one of the efficient coding techniques that belongs to LDPC codes. LDPC codes are gaining greater importance in the applications requiring efficient and reliable transfer of information over communication channel. LDPC codes are innovative techniques which play a significant role in satellite communication, CDMA, Bluetooth etc. where the main purpose is to achieve the reliable data transmission maintaining efficiency, quality and minimum bandwidth. Thus, in the present paper, an effort is made on FPGA implementation of LDPC decoding algorithm using Simplified Sum Product Algorithm (SSPA) technique to obtain better efficiency with lesser error rate. The SSPA algorithm is verified in MATLAB for different parity check matrix combinations. Then this is coded in HDL and implemented in real time, integrating with FPGA using Artix-7.***

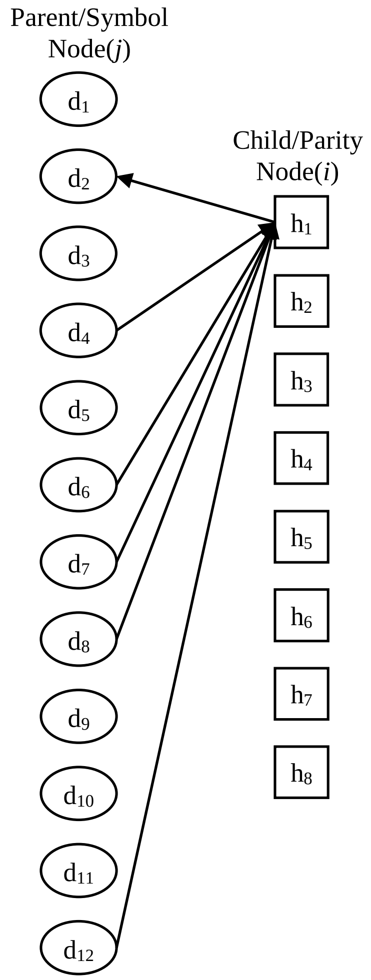
***Keywords:***

***Sum Product Algorithm (SPA), Simplified Sum Product Algorithm (SSPA), Low Density Parity Check (LDPC), Variable Node (VN), Check Node (CN)***

## 1. INTRODUCTION

LDPC codes were first discussed by Gallager in 1963 in his PhD thesis, but were not considered for more than three decades as the technology at that time was unable to support the hardware implementation of decoder and encoder. But nearly after 30 years Mackay- Neal rediscovered LDPC codes in 1993 [2]. LDPC codes show the BER performance near to that of Turbo codes i.e., Shannon's limit and possess excellent error correcting and decoding characteristics hence gain significant importance in practical applications. LDPC codes are obtained by a sparse parity matrix which contains '0's and relatively few number of '1's. A LDPC code may be regular or irregular. In regular LDPC codes, there will be fixed number of '1's per row and per column whereas irregular LDPC codes contain variable number of '1's per row and per column. The decoding of LDPC can be done using either Soft Decision Decoding technique or Hard Decision Decoding technique [9]. Bit Flipping algorithm is an example for hard decision decoding, whereas Sum-Product Algorithm, Simplified Sum-Product Algorithm, LogSPA, Simplified Soft Distance algorithm, Min-sum decoding (Max-Product algorithm) are different soft decision decoding techniques. The soft decision decoding scheme works on iterative methods. Because of iterative method, the performance of error detection and correction is high.

The SPA was modified to reduce the computational complexity and thus the Simplified Sum-Product Algorithm (SSPA) came into existence, where the number of product terms in horizontal step are greatly reduced, thus reducing the decoding time. In case of SSPA, the prior probabilities of received vector are given as the input to the decoder, the outputs taken from the decoder are called as posterior probabilities.




$Q_{ij}$ - Information sent from VN to CN
$R_{ij}$ - Information sent from CN to VN

Fig.1. Tanner graph showing nodes and check nodes

## 2. SIMPLIFIED SUM PRODUCT ALGORITHM (SSPA)

The example of the tanner graph is shown in above Fig.1. The tanner graph has two kinds of nodes such as Symbol nodes and Check Nodes (CN), indicated as $j$ and $i$ respectively [7]. The symbol node is also called as variable node or parent node, whereas check node is also called as child node. The line is drawn between VN and CN, if and only if that bit is involved in that parity equation. The information sent from VN to CN is determined as $Q_{ij}^x$ and the information sent back from check nodes to variable nodes is determined as $R_{ij}^x$.

The aim of the SPA [2] [3] [7] is to find decoded vector *d*. The error introduced during the transmission of the encoded message through channel can be detected and corrected in the decoding technique. The decoded vector will be the estimation of the code vector [c], which is actually transmitted, and the decoded vector should satisfy the syndrome condition,

$$H \circ \hat{d} = 0 \tag{1}$$

The estimate $Q_{ij}^x$ is sent from each symbol node $d_j$ to each child parity CN $h_i$. This estimate is based on the information which has been received by all the other children parity node CN that is depending upon the corresponding parity CN is in a state *x*, where *x* representing either 0 or 1.

Similar to this, the estimate $R_{ij}^x$ is sent to each parent symbol node $d_j$ by each parity CN $h_i$, where the estimate is based on the information which is provided by all the other VN.

The channel information is determined by using probability density function given by the following expression:

$$f_j^1 = \frac{1}{1+e^{-\frac{2Ay_i}{\sigma^2}}} \tag{2}$$

$$f_j^0 = 1 - f_j^1 \tag{3}$$

where, $Y_j$ indicates channel output at time instant *j*, which is transmitted with amplitude ±A in polar format. In the present algorithm it is normalized as ±1 in polar format.

In SPA algorithm [2][3][7] all the combinations possible for the code bits are considered so as to satisfy the parity check equation to determine the values of coefficients $R_{ij}^0$ and $R_{ij}^1$ as a function of the values of coefficients $Q_{ij}^0$ and $Q_{ij}^1$. There is a need for the new technique which avoids considering all these possible combinations in the parity check equation for the calculation of values of coefficients $R_{ij}^0$ and $R_{ij}^1$. Hence, Mackay and Neal introduced an algorithm, which simplified the calculation of SPA [2].

The simplified SPA method consists of following 5 steps [2][7]:

## 2.1 INITIALIZATION

The probability density function which are already calculated using Eq.(1) and Eq.(2) are used to apply the values for $Q_{ij}^x$, this step of initialization is same as that of used is the traditional SPA algorithm [2][3][7]. The following equation shows the initialization steps.

$$Q_{ij}^x = f_j^x \tag{4}$$

where *x* being either in state 1 or state 0, thus reducing to the following equations,

$$Q_{ij}^0 = f_i^0 \text{ and } Q_{ij}^1 = f_i^1 \tag{5}$$

The modified version of SPA also follows the iterative procedure like SPA and thus implements two steps, such as horizontal step and vertical step, by considering the values taken from the corresponding H matrix.

## 2.2 HORIZONTAL STEP

In order to simplify the steps involved in sending $Q_{ij}^0$ and $Q_{ij}^1$ values separately to all the parity CN for the calculation, distance between them is sent instead, that is

$$\delta Q_{ij} = Q_{ij}^0 - Q_{ij}^1 \tag{6}$$

The intermediate quantity $\delta R_{ij}$ is taken for the pair of nodes *i* and *j*, as given in the following equation,

$$\delta R_{ij} = \prod_{j' \in \frac{N(i)}{j}} \delta Q_{ij} \tag{7}$$

where, *N*(*i*) defines the set of indexes of all the VN which are connected to the parity check node $h_i$, *N*(*i*)/*j* defines the same symbol set with the exclusion of the parent VN $d_j$.

By using these intermediate values, the information sent from check nodes to the VN node is calculated indicated by the coefficients $R_{ij}^0$ and $R_{ij}^1$, calculated as:

$$R_{ij}^0 = \frac{1}{2}\left(1 + \delta R_{ij}\right) \tag{8}$$

$$R_{ij}^1 = \frac{1}{2}\left(1 - \delta R_{ij}\right) \tag{9}$$

By this step the Sum-Product-Algorithm can be simplified, and thus named as Simplified-Sum-Product-Algorithm. Here the main advantage of this step is that the product terms are reduced.

## 2.3 VERTICAL STEP

In vertical step, $Q_{ij}^0$ and $Q_{ij}^1$ coefficients are updated. Similar to horizontal step the $Q_{ij}^0$ and $Q_{ij}^1$ coefficients are calculated for each pair of nodes *i*, *j* with every possible value of *x*, where *x* is either 0 or 1.

$$Q_{ij}^x = \alpha_{ij} f_j^x = \prod_{i' \in \frac{M(j)}{i}} R_{i'j}^x \tag{10}$$

where *x* being either in state 1 or state 0, thus reducing to the following equations,

$$Q_{ij}^0 = \alpha_{ij} f_j^0 = \prod_{i' \in \frac{M(j)}{i}} R_{i'j}^0 \tag{11}$$

$$Q_{ij}^1 = \alpha_{ij} f_j^1 = \prod_{i' \in \frac{M(j)}{i}} R_{i'j}^1 \tag{12}$$

where *M*(*i*) indicates set of indexes of all the children parity CN which are connected to the symbol node $d_j$. *M*(*j*)/*i* indicates the same children parity check nodes set where the children parity CN $h_i$ is excluded. $\alpha_{ij}$ is a constant used for normalization and this constant is considered such that $Q_{ij}^0 + Q_{ij}^1 = 1$.

$$\alpha_{ij} = \frac{1}{f_j^0 \prod_{i' \in \frac{M(j)}{i}} R_{i'j}^x + f_j^1 \prod_{i' \in \frac{M(j)}{i}} R_{i'j}^1} \tag{13}$$

## 2.4 ESTIMATION

In order to determine the decoded vector there is need for calculation of a posterior probabilities $Q_j^0$ and $Q_j^1$ for the current iteration. The posterior probabilities $Q_j^0$ and $Q_j^1$ are given as:

$$Q_j^x = \alpha_j f_j^x = \prod_{i \in M(j)} R_{ij}^x \tag{14}$$

where *x* being either in state 1 or state 0, thus reducing to the following equations,

$$Q_j^0 = \alpha_j f_j^0 = \prod_{i \in M(j)} R_{ij}^0 \tag{15}$$

$$Q_j^1 = \alpha_j f_j^1 = \prod_{i \in M(j)} R_{ij}^1 \tag{16}$$

where $M(j)$ indicates the set of all children parity CN connected to the VN $d_j$. Here, in this step all the children nodes are considered. $\alpha_j$ indicates a constant term used for normalization and it is selected in such a way that $Q_{ij}^0 + Q_{ij}^1 = 1$.

$$\alpha_j = \frac{1}{f_j^0 \prod_{i \in \frac{M(j)}{i}} R_{ij}^x + f_j^1 \prod_{i \in \frac{M(j)}{i}} R_{ij}^1} \tag{17}$$

After finding the $Q_j^x$ coefficients the decoded vector need to be determined by using the following equation,

$$\hat{d}_j = \max\left(Q_j^x\right) \tag{18}$$

This can be simplified as, if $Q_j^0 > Q_j^1$ then $d_j$ is estimated as:

$$\hat{d}_j = 0, \text{ Else } \hat{d}_j = 1 \tag{19}$$

## 2.5 SYNDROME CHECK

After calculating the decoded vector $\hat{d}$ , this estimated vector should satisfy the syndrome condition [ $H \circ \hat{d} = 0$ ], such that the decoded vector is considered as a valid code and is equal to the code vector that is $c = \hat{d}$ . If the decoded vector $\hat{d}$ doesn't satisfy the syndrome condition [ $H \circ \hat{d} != 0$ ], then the algorithm goes for next iteration and this continues for specified set of iterations or till decoded vector satisfies the syndrome condition.

## 3. SIMULATION RESULTS

The input message is encoded and sent through AWGN channel, the output of the channel is fed to decoder, which detects and corrects the error. The Fig.2 shows the system level output simulation result obtained for SSPA algorithm, G of matrix (32×96) and H of matrix (64×96). The code was tested for 3 different H matrix combination they are 8×12, 64×96, 504×1008 [8]. The decoder was tested with both SPA and SSPA algorithm. The BER graph was plotted for 8×12 H matrix that is (4, 12) for 1000 iterations and 64×96 (32,96) H matrix for 2000 iterations. SPA and SSPA algorithm performance is almost similar, this can be seen in the BER graph shown in Fig.3 and Fig.4.

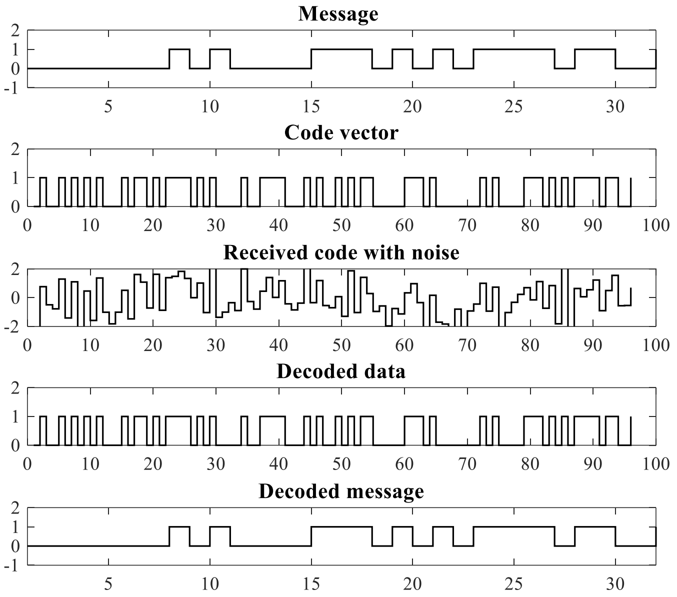



Fig.2. Simulation results obtained for SSPA algorithm, G of matrix (32×96) and H of matrix (64×96)

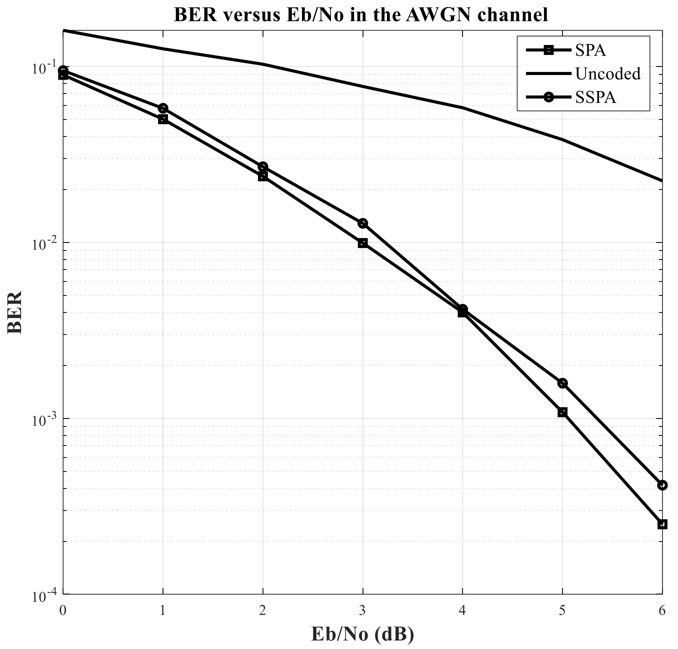



Fig.3. 8×12 matrix 1000 iterations, code rate 1/3

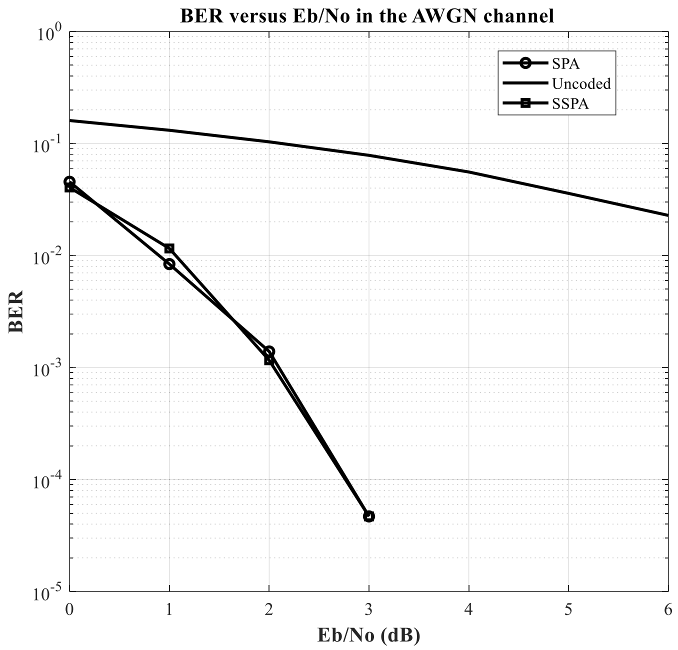



Fig.4. 64×96 2000 iterations, code rate 1/3

# 4. FPGA IMPLEMENTATION OF SSPA DECODER

The SSPA LDPC decoding technique has been implemented in FPGA. The overall methodology of the system is implemented with 10 stages as shown in Fig.5. When the system starts the initial values, prior probabilities are calculated in MATLAB. The code vector when passed through AWGN channel, noise is added to the original message and these output values from channel are taken as $f_0$ and $f_1$ values, which are further considered as input values to the SSPA decoder. The prior probability values $f_0$ and $f_1$ obtained are converted to a fixed point. Here 16-bit fixed point values are used, where 4 bits are before decimal point and 12 bit after decimal with the precision of 0.00024 ($1/2^{12}$). These values are assigned to the $Q_{ij}^x$ in initialization step and then stored in the memory. These stored values are considered for next calculations, where $\partial Q_{ij}^x$ is the distance between $Q_{ij}^0$ and $Q_{ij}^1$, are calculated and stored in the memory in matrix form.

In order to implement in FPGA, the following modifications are done in the algorithm in the vertical step.

***Vertical Step***: The vertical step in the Simplified sum product algorithm includes the normalization as in Eq.(13). The normalization is used such that while considering many values measured on different scales can be arranged as a common scale in order to make operation easier. Also, Normalization is used to bring many quantities with different measurement into alignment. The Eq.(10)-Eq.(12), by considering $\alpha$ value as constant equal to ‘1’ reduces as shown below,

$$Q_{ij}^x = f_j^x \prod_{i' \in \frac{M(j)}{i}} R_{i'j}^x \tag{20}$$

Thus, treating alpha factor as constant value and considering the probability density functions $f_j^x$ and the $R_{ij}^x$ in the vertical step, new $Q_{ij}^x$ values are obtained.

Thus, by considering $\alpha$ factor equal to unity, the computational complexity reduces, which reduces the decoding time and thereby increasing the speed of operation. The above SSPA algorithm is implemented for 8×12 matrix and verified. Further testing is to be done for higher matrix.

***Estimation***: As already discussed, the calculation of posterior probabilities $Q_j^0$ and $Q_j^1$ are required to determine the decoded vector. But by reducing the alpha factor, the Eq.(14) reduces as below,

$$Q_j^x = f_j^x = \prod_{i \in M(j)} R_{ij}^x \tag{21}$$

After finding the estimated vector, we need to find out the decoded vector in order to check if the decoded vector is equal to code vector or not. This can be done by the equations shown below,

$$\hat{d}_j = \max\left(Q_j^x\right) \tag{22}$$

This can be simplified as, if $Q_j^0 > Q_j^1$ then $d_j$ is estimated as: $\hat{d}_j = 0$, Else $\hat{d}_j = 1$.

## 4.1 ISE SIMULATION RESULTS

The Fig.6 shows the simulation result of SSPA decoding algorithms, designed for 8×12 parity check matrix, which has been written in verilog HDL and simulation result is taken using ISIM simulator.

The prior probability density functions $f_0$ and $f_1$ are calculated for the transmission codeword “111110001000”, using MATLAB and these values are converted to fixed point binary format with 12-bits after decimal and 4 bits before the decimal. These converted values are given as inputs to the decoder. When this codeword is transmitted over AWGN channel, the received decoded vector is “111110001001” which indicates the error in last bit.

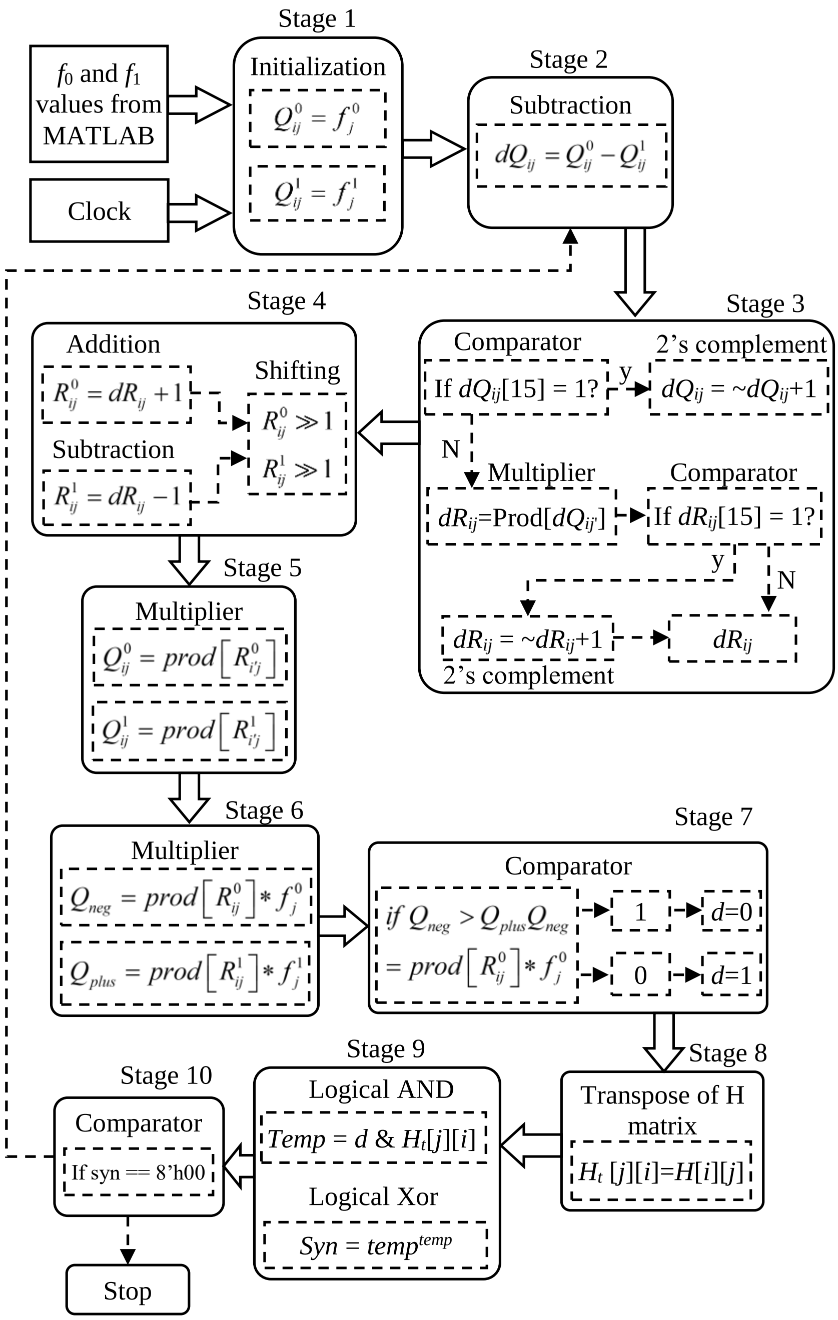



Fig.6. Architecture of SSPA LDPC decoder

For every iteration the syndrome vector is checked to prove that the decoded vector is same as coded vector. The SSPA algorithm rectifies the error in last bit, as shown in simulation results, within 7 iterations, thus indicating the syndrome 'syn' is "00000000". The obtained decoded vector is "111110001000", which is same as code vector. The whole operation completes within 25.255ns with a speed of 39.595MHz. The HDL coding which is done in verilog has been implemented in the Artix-7 FPGA kit and the obtained results are shown in Table.1 - Table.3.

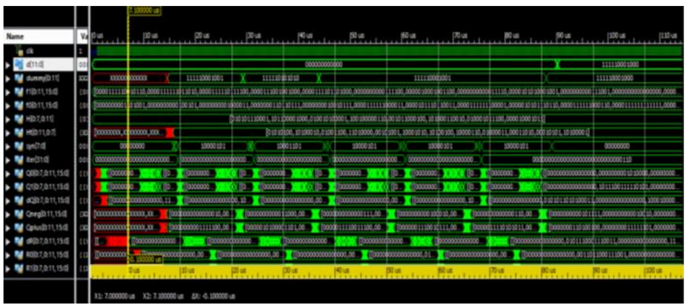


Fig.5. Simulation result of SSPA in ISIMTiming and Device Utilization Report

Timing summary of 8×12 SSPA LDPC Decoder:

Minimum period: 25.255ns (maximum frequency: 39.595MHz)

Device utilization summary:

- Selected Device: 7a100tcsg324-2
- **Slice Logic Utilization**

Table.1. Table showing Number of slice registers utilized

| | Used | Available | Utility (%) |
|---|---|---|---|
| Slice Registers | 2877 | 126800 | 2 |
| Slice LUTs | 7416 | 63400 | 11 |
| LUT as a Logic | 6562 | 63400 | 10 |
| LUT as a Memory | 854 | 19000 | 4 |

- **Slice Logic Distribution**

Table.2. Table showing number of Flipflops used

| | Used | Available | Utility (%) |
|---|---|---|---|
| LUT Flip Flop pairs | 2792 | 7501 | 37 |
| LUT as Flip Flop | 4624 | 7501 | 61 |

- **Specific Feature Utilization**

Table.3. Table showing number of BufGCTRL and DSP48 slices used

| | Used | Available | Utility (%) |
|---|---|---|---|
| BUFGCTRL | 1 | 128 | 0 |
| DSP48E1s | 10 | 240 | 4 |

## 5. CONCLUSION

In this paper an attempt is done to design and implement SSPA LDPC decoder using FPGA Architecture. The SSPA is a soft decision decoder which can detect and correct errors with a less decoding time. The SPA shows the best error performance but is limited by its large computations and complexity in hardware implementation, whereas SSPA reduces Computations and memory requirements. The Computational Complexity is greatly reduced as Compared to SPA. The design has been tested for parity check matrix combinations such as (8×12) and (64×96). The same has been implemented in FPGA for parity check matrix (8×12) using Artix-7 board. Here the normalization factor in the vertical step has been eliminated, so as to increase the speed and decrease the complexity of the system, thus reducing the decoding time.

## REFERENCES


[1] Robert G. Gallager, "Low Density Parity Check Codes", *IRE Transactions on Information Theory*, Vol. 8, No. 1, pp. 379-423, 1962.

[2] D.J.C. Mackay and R.M. Neal, "Near Shannon Limit Performance of Low Density Parity Check Codes", *Electronics Letters*, Vol. 33, No. 3, pp 457-458, 1997.

[3] D.J.C. MacKay, "Good Error-Correcting Codes based on Very Sparse Matrices", Available at: http://www.inference.phy.cam.ac.uk/mackay/CodesGallager.html

[4] Chen Qian, Weilong Lei and Zhaocheng Wang, "Low Complexity LDPC Decoder with Modified Sum-Product Algorithm", *Tsinghua Science and Technology*, Vol. 18, No. 1, pp. 57-61, 2013.

[5] M.C. Davey and D.J.C. Mackay, "Low density parity check codes GF (q)", *IEEE Communication Letters*, Vol. 2, No. 6, pp. 165-167, 1998.

[6] S. Papaharalabos and P.T. Mathiopoulos, "Simplified Sum-Product Algorithm for Decoding LDPC Codes with Optimal Performance", *Electronics Letters*, Vol. 45 No. 2, pp. 113-119, 2009.

[7] Jorge Castineira Moreira and Patrick Guy Farrell, "*Essentials of Error-Control Coding*", John Wiley and Sons, 2006.

[8] D.J.C Mackay, "Online Database of Low Density Parity-Check Codes", Available at http://wol.ra.phy.cam.uk/mackay/codes/data.html

[9] P. Namrata and Brijesh Vala, "Design of Hard and Soft Decision Decoding Algorithms of LDPC", *International Journal of Computer Applications*, Vol. 90, No 16, pp. 1-8, 2014.

[10] M.C. Davey, "Error-Correction using Low-Density Parity-Check Codes", PhD Dissertation, University of Cambridge, 1999.

[11] L.M. Tanner, "A Recursive Approach to Low Complexity Codes", *IEEE Transactions on Information Theory*, Vol. 27, No. 5, pp. 533-547, 1981

[12] Shu Lin, "*Error Control Coding*", 2nd Edition, Pearson, 2004.

[13] T.K. Moon, "Error Correction Coding- Mathematical Methods and Algorithms", John Wiley and Sons, 2005.

[14] Iva Bacic, Kresimir Malaric and Zeljki Petrunic, "A LDPC Code/Decode Channel Coding Model based on sum-Product Algorithm Realization via Labview", *Proceedings of 20th International Conference on Applied Electromagnetics and Communications*, pp. 1-6, 2011.