

# Reduced Recurrent Minimum Power Encoding Scheme for On-Chip Interconnects

Bharathi Subramaniam and Gowrison Gengavel

**Abstract---** Network on Chip (NoC) has been proposed as a suitable solution for today's on-chip communication challenges. Many of the existing on-chip interconnects based power dissipation methods are in parallel link communication. In this work, the serial link transmission based new data encoding plan is proposed to reduce the power in data transmission and lessen the number of self transition. The proposed Reduced Recurrent Minimum Power encoding (RRMP) approach estimates the correlation among each bit in the data stream. The suggest RRMP serial links in NoC architectures can provide savings in the power consumption, reduction of area, and timing delay by adjusting the transition of serial data bit. The encoding method has been verified with different order of data streams. The proposed RRMP encoder and decoder have been coded in VHDL, simulated using Modelsim simulation tool and synthesized using Xilinx ISE 13.2. The simulation result shows the possible reduction in power distribution than other existing systems. The proposed RRMP scheme is targeting on Xilinx Virtex-6 XC6VLX75T device.

**Keywords---** Network on Chip, Serial Communication, Power Dissipation, Reduced Recurrent Minimum Power Encoding, Bit Transition.

## I. INTRODUCTION

The advancement in VLSI allowed the scientists to plan a framework on a chip called System on Chip (SoC). A lot of research has been given to decrease the power utilization of interconnections in SoCs. But, SoC has a few downsides,

like the absence of adaptability and reusability and so, the Network on Chips (NoCs) has been proposed for today's communication. The NoCs are reusable, adaptable and can handle many disservices of SoCs. Reducing the swing voltage of force supply, utilizing double edge voltage, voltage-frequency islands [1], Dynamic Voltage Scaling (DVS) and Dynamic Power Management (DPM) [2] were some power decreasing techniques used. The low power encoding techniques have been adopted to lessen the power utilization in chip interconnections by reducing the number of switching activities.

### *Literature Survey*

In every new design of the computer architecture improves not only the system performance but also enhances the significant parameters memory capacity, reliability, power efficiency, and cost effectiveness. Energy efficiency becomes an inevitable motivation in new architecture. Energy efficiency and proportionality are main challenges in new architecture because their cost and scalability influence. Many strategies have been suggested in the earlier period attempts to reduce the power utilization by several encoding procedures. [3], [4], [5]. More works have been developed in encoding schemes to reduce the switching activity on external buses. By exploiting the memory reference locality characteristics, the switching activity on the address bus have been reduced to some extent.

In the Power Protocol approach [6], to reduce the dynamic power dissipation on off-chip data buses. The numbers of bus lines used for data transfer have been reduced and a small cache was introduced in both sides of the off-chip data bus. While considering the parameters

---

*Bharathi Subramaniam, Assistant Professor (Sr), Department of Electronics and Communication Engineering, Institute of Road and Transport Technology, Erode, India.*

*Gowrison Gengavel, Assistant Professor (Sr), Department of Electronics and Communication Engineering, Institute of Road and Transport Technology, Erode, India.*

skew, clock synchronization, crosstalk, area cost and wiring complexity, on-chip synchronous serial communication has several advantages than parallel communication. But, the serial wire transmission dissipates more energy than parallel bus due to the bit multiplexing. To attain switching activity reductions in the data bus many encoding algorithms were proposed [7], [8]. A serialized low-energy transmission (SILENT) coding [9] proposed strategy to limit the switching energy of the on-chip serial communication by minimizing the number of transitions on the serial wire and applied the coding method in the CMOS SoC functioning process which incorporates many processing units with packet switched on-chip networks.

In graphics SDRAM, to attain low cost as well as high bandwidth, the numbers of I/O lines up to 32 lines per DRAM have been extended and correspondingly, increase the data rate to some Gb/s per pin while considering single-ended signaling. But, DRAM suffered from the bottlenecks of parallel single-ended signaling, reference voltage noise and crosstalk [10], [11]. In data bus inversion (DBI) method [12], an analog majority voter insensitive to mismatch for small area and delay and reduces the peak-to-peak jitter and the voltage fluctuation has been used. A lot of research works have explained on reducing power consumption in the off-chip buses. Although many techniques reside for reducing bus power in address buses, only smallest amount techniques have been proposed for off-chip data bus power reduction [13].

In the large capacity content addressable memory (CAM), the lookup table function in a single clock cycle using dedicated comparison circuitry has been implemented used in network routers for packet forwarding as well as classification. The CAM-design challenge is to reduce power consumption related with the huge amount of parallel active circuitry. Three architectures level [14] has been outlined to reduce CAM power that is bank-selection, pre-computation, and dense encoding. Two data bus encoding methods [15] have been proposed for reducing power consumption in the data buses. The Variable Length Value

Encoder (VALVE) scheme able to detect and encoded the variable lengths of repeated bit patterns in the data and Tunable Bus Encoder (TUBE) encoded the recurrence in contiguous as well as noncontiguous bit positions of data values.

Low power codes were employed in a reprogrammable application in specific manner and the procedure obtain significant power savings. The micro architectural support enabled the reprogram ability of the encoding transformations in order to follow the code particularities efficiently. Such reprogram ability is accomplished by using small tables which store related application information. Some transformations that effect in optimal power reductions for each application have been selected by means of short indices stored in the table and accessed only once at the commencement of the transformed bit sequence [16]. In [17], an adaptive bus coding method was used to decrease the transition activity by exploiting the value repetition in Media bench benchmark set by providing an extra bit line analogous to bus-invert coding.

The Bus-Invert Coding [18] is one of the low power encoding which has been used for the consistent dispersion data and the parallel bus with spatial repetition. Another scheme Lightweight coding (LWC) [17] tried to reduce the number of transition in value by introducing dynamic detection of frequent values and their use in encoding data values by means of frequent value (FV) encoding system. An adaptive encoding scheme was proposed [19] based on self-organizing lists to get reduction in transition movement by exploiting the spatial and temporal locality of the addresses which significantly decrease the transition activity on data and multiplexed address buses without any additional bit lines. Shoreline coding [20] presented the general theory of limited weight codes for a class of parallel terminated buses and power dissipation on such a bus line is more for a logical. A perfect  $k/2$  limited weight code and a non-perfect 3-limited weight code were illustrated algorithmically in implementation.

The proposed low power encoding method Reduced Recurrent Minimum Power (RRMP) utilizes a less switching transition based framework with the reduction in the number of switching activities or diminishing the number of binary transition incorporated in the code words. The new proposed encoding strategy presents an on-chip data serialization method with encoding for serial communications. The responsibilities of this work are as

- Diminishes the number of moves virtually and ensures a low-control distribution among on-chip communication through NOC links.
- The technique for data encoding reduces the design complexity than other conventional methods.

- Experimentation has been completed over different kinds of data records with different augmentations and sizes.
- Performance of the RRMP coding scheme outperforms than the existing encoding schemes and results of the framework gives less power for multiple data streams.

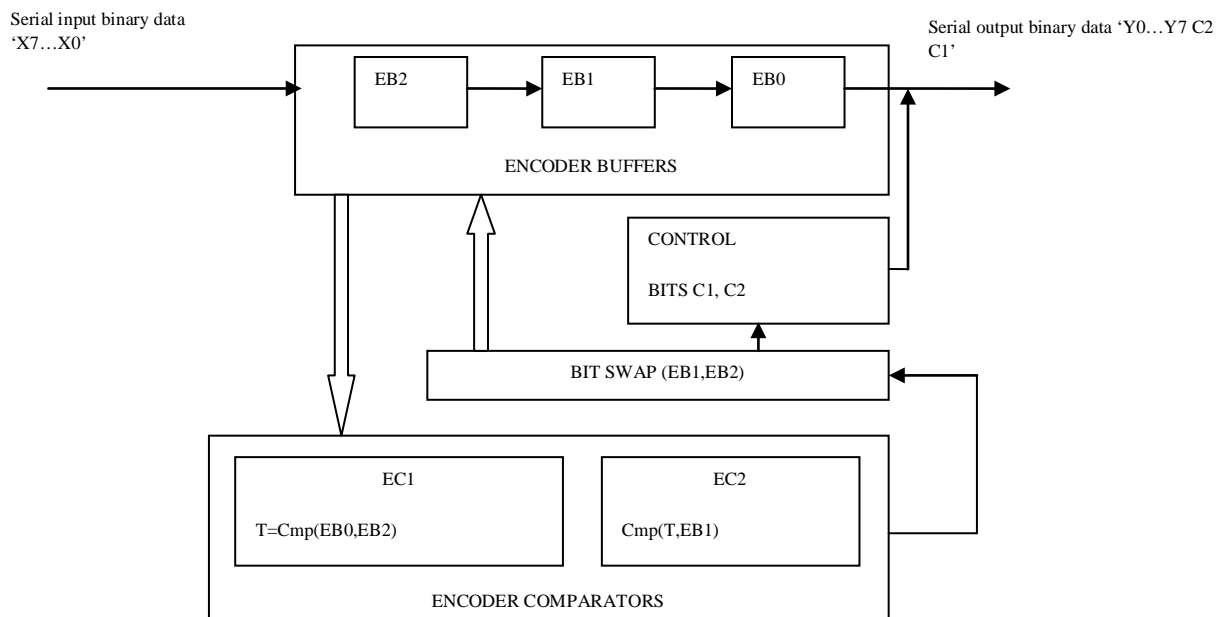


Fig. 1: Block Diagram of Encoder

## II. ENCODER

The objective of the proposed scheme is to swap the bits of data to control immediate 0 and 1 transitions. The block diagram of the encoder is shown in Fig.1. The 'n' bit 8 4 2 1 weighted serial binary data (X) 'X<sub>n-1</sub> ... X<sub>0</sub>' is given to the input of the encoder by means of single link. The proposed work has been implemented for '8' bit 8 4 2 1 weighted serial binary data (X) 'X<sub>7</sub> ... X<sub>0</sub>'. Primarily, the input bit has been entered into the earliest buffer EB2 of the buffer

division and at regular interval(s), the bit shifted to right buffers. The process continues and the buffers EB2, EB1 and EB0 hold the lower three bit values of 'X' (X<sub>2</sub> X<sub>1</sub> X<sub>0</sub>) respectively. To begin with shifted and buffered '3' bit data, the bit values of buffers EB2 and EB0 have been forwarded to comparator EC1 through buses. In comparator EC1, the equality analysis has been executed by comparing the bit values of EB2 and EB0. The further operations depend upon the logical value of the comparator EC1.

In first condition, if both the bit values of the buffers EB2 and EB0 are identical, the corresponding either '0' or '1' bit has been temporally stored to 'T'. Yet again, the equality assessment has been performed by comparing the bit values of 'T' with buffer EB1 in comparator EC2. The swap operation has been performed between EB2 and EB1 if the equality test value is similar. Concurrently, the value of EB0 comes outside of the buffer as an encoded bit of X0 and stored as temporary data. In the second condition, if both the bit values of the buffers EB2 and EB0 are not identical, the corresponding either '0' or '1' in EB0 comes outside of the buffer as an encoded bit of X0 and stored as temporary data.

As soon as, after performing any one of above conditions, the bit values of EB2 and EB1 right shifted to EB1 and EB0 correspondingly and the later bit X3 enters into buffer EB2. This process continues for 'n-2' times and every group of '3' bits in buffers EB2, EB1 and EB0. After 'n-2' times the bit values in EB2 and EB1 have been right shifted and comes out as an encoded bits of 'X' sequences.

If any swap operation carried out in the 'X4X3X2X1X0' binary group, correspondingly the control bit C1 will be set to '1' value. Likewise, anyone swap operation carried out in the 'X7X6X5X4X3' binary group, correspondingly the control bit C2 will be set to '1' value. In the end, the encoded data will be generated along with the control bit values like 'Y0 ---- Y7C2C1' have been generated and transmitted serially. Fig.2 gives Flow diagram of encoding scheme. Algorithm for proposed encoding technique to reduce the number of transitions

Input: Dataset (X= X7X6X5X4X3X2X1X0)

Output: Encoded Dataset (Y= Y0Y1Y2Y3Y4Y5Y6Y7C2C1)

Begin

n= size (X); /\* Total number of bits in the sequence\*/

```

C1 = 0; C2=0; /* control bits*/
t1=t2=0: /*temporary variables*/
/*transition check in bits sequence*/
for i= 0 to 2 step 1
    if (xi = xi+2) ≠ xi+1 then
        swap (xi, xi+1);
        update the X sequences;
        t1=t1+1;
    end if
end for
if t1>=1 then
    C1=1;
else
    C1=0;
end if
for i=3 to n-2 step 1
    if (xi = xi+2) ≠ xi+1 then
        swap (xi, xi+1);
        update the X sequences;
        t2=t2+1;
    end if
end for
if t2>=1 then
    C2=1;
else
    C2=0;
end if
/*encoded data*/
Y=[ updated X sequences C2C1]
end
    
```

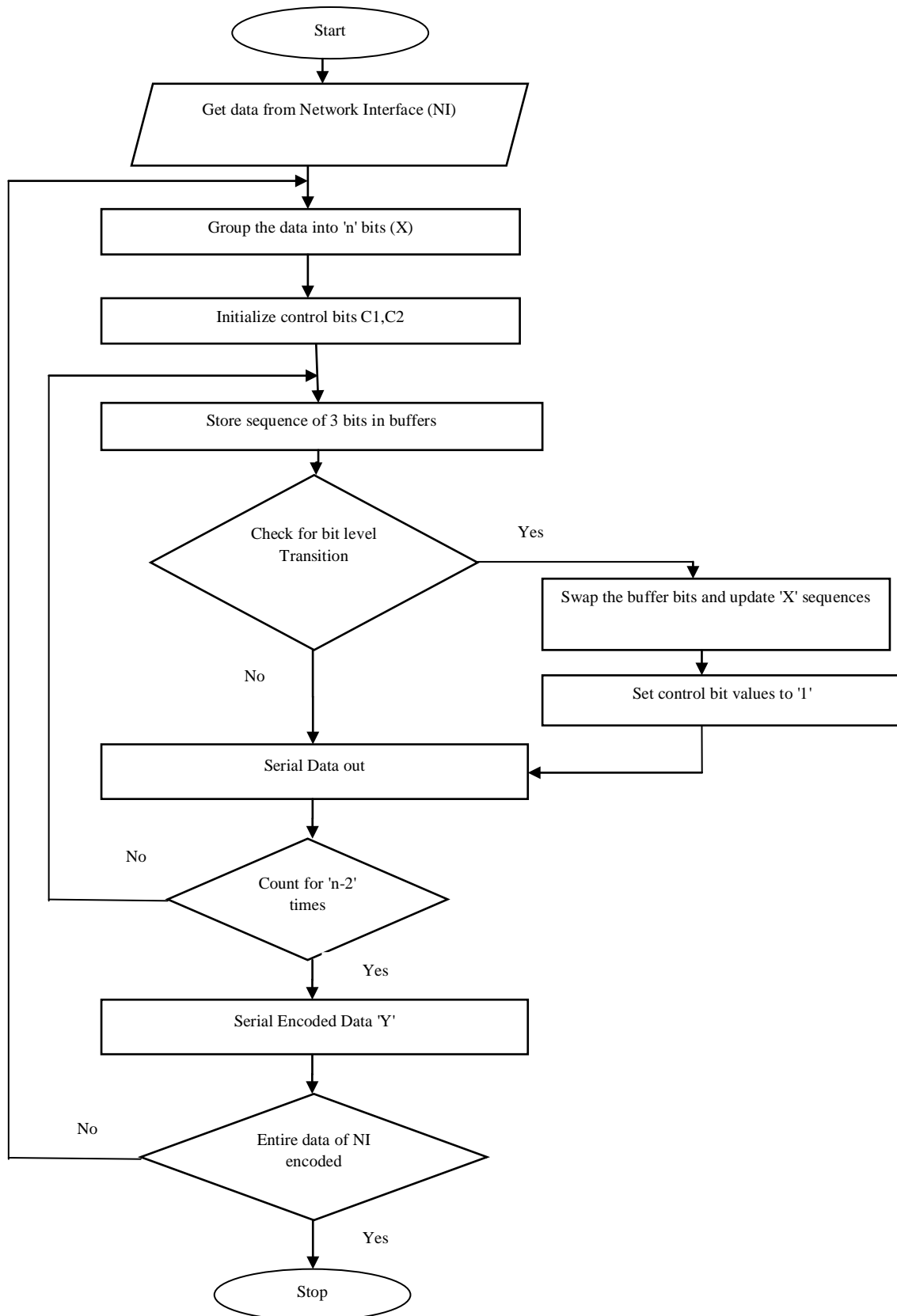
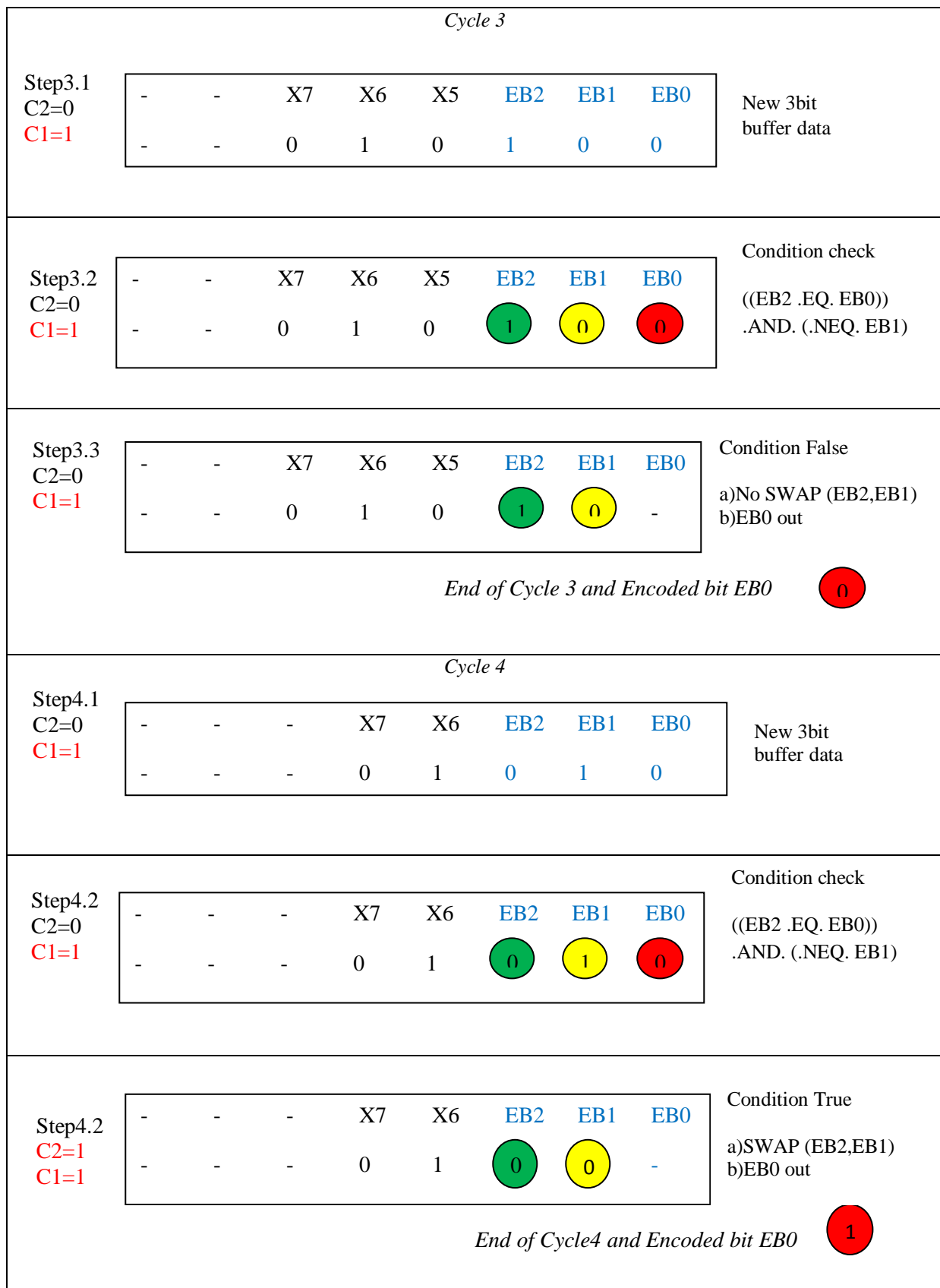


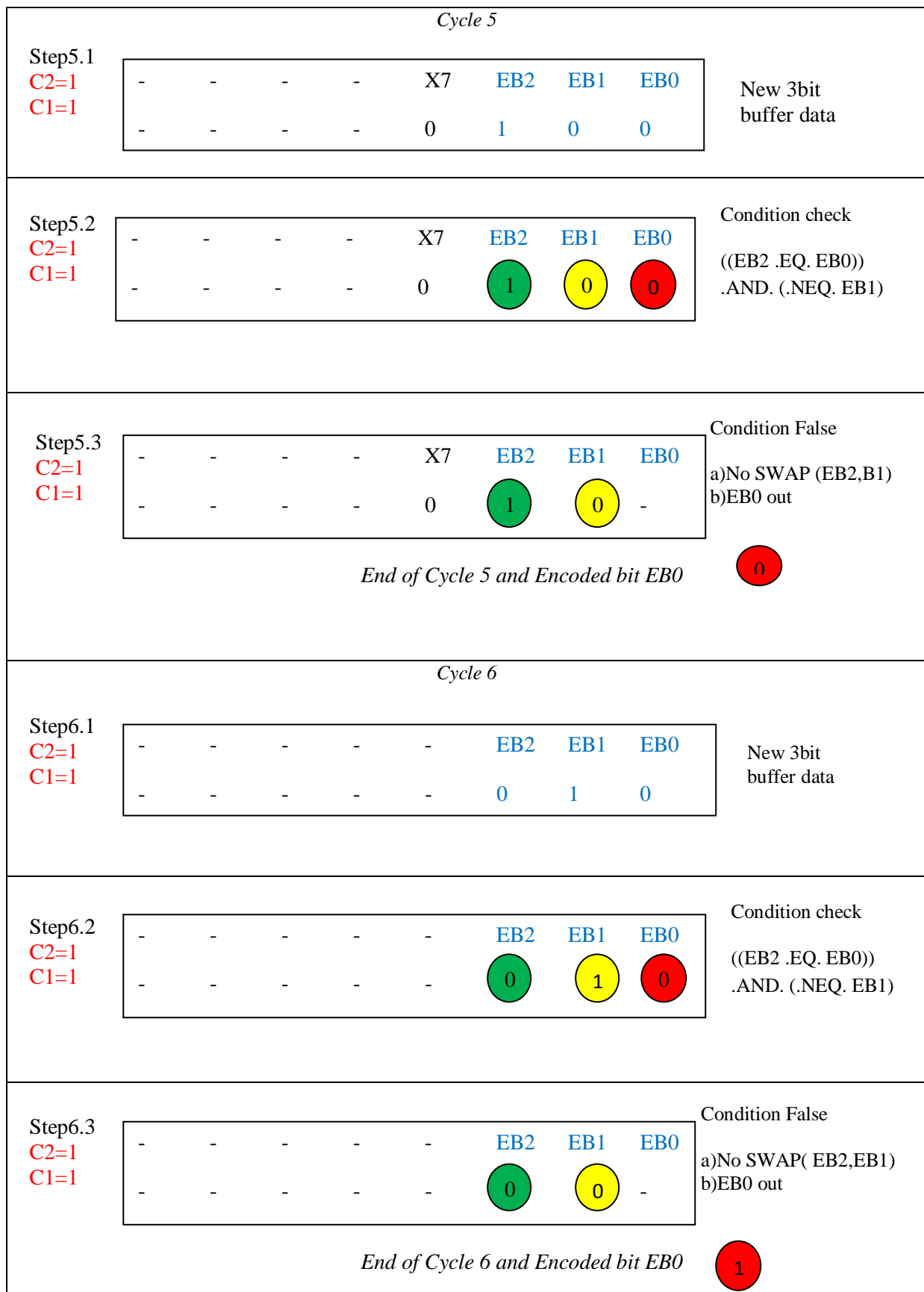
Fig. 2: Flow Diagram of Encoding Scheme

The procedure involved in encoder operations are given in the example.

**Example Encoder**

Input Data- 0101 0010																																
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 12.5%;">128</td> <td style="width: 12.5%;">64</td> <td style="width: 12.5%;">32</td> <td style="width: 12.5%;">16</td> <td style="width: 12.5%;">8</td> <td style="width: 12.5%;">4</td> <td style="width: 12.5%;">2</td> <td style="width: 12.5%;">1</td> </tr> <tr> <td>X7</td> <td>X6</td> <td>X5</td> <td>X4</td> <td>X3</td> <td>X2</td> <td>X1</td> <td>X0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>1</td> <td>0</td> </tr> </table>								128	64	32	16	8	4	2	1	X7	X6	X5	X4	X3	X2	X1	X0	0	1	0	1	0	0	1	0	8bit binary input data
128	64	32	16	8	4	2	1																									
X7	X6	X5	X4	X3	X2	X1	X0																									
0	1	0	1	0	0	1	0																									
<i>Cycle 1</i>																																
Step1.1 C2=0 C1=0	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 12.5%;">X7</td> <td style="width: 12.5%;">X6</td> <td style="width: 12.5%;">X5</td> <td style="width: 12.5%;">X4</td> <td style="width: 12.5%;">X3</td> <td style="width: 12.5%;">EB2</td> <td style="width: 12.5%;">EB1</td> <td style="width: 12.5%;">EB0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>1</td> <td>0</td> </tr> </table>							X7	X6	X5	X4	X3	EB2	EB1	EB0	0	1	0	1	0	0	1	0	3bit buffer data								
X7	X6	X5	X4	X3	EB2	EB1	EB0																									
0	1	0	1	0	0	1	0																									
Step1.2 C2=0 C1=0	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 12.5%;">X7</td> <td style="width: 12.5%;">X6</td> <td style="width: 12.5%;">X5</td> <td style="width: 12.5%;">X4</td> <td style="width: 12.5%;">X3</td> <td style="width: 12.5%;">EB2</td> <td style="width: 12.5%;">EB1</td> <td style="width: 12.5%;">EB0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>1</td> <td>0</td> <td style="text-align: center;"><span style="color: green;">0</span></td> <td style="text-align: center;"><span style="color: yellow;">1</span></td> <td style="text-align: center;"><span style="color: red;">0</span></td> </tr> </table>							X7	X6	X5	X4	X3	EB2	EB1	EB0	0	1	0	1	0	<span style="color: green;">0</span>	<span style="color: yellow;">1</span>	<span style="color: red;">0</span>	Condition check  ((EB2 .EQ. EB0)) .AND. (.NEQ. EB1)								
X7	X6	X5	X4	X3	EB2	EB1	EB0																									
0	1	0	1	0	<span style="color: green;">0</span>	<span style="color: yellow;">1</span>	<span style="color: red;">0</span>																									
Step1.3 C2=0 C1=1	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 12.5%;">X7</td> <td style="width: 12.5%;">X6</td> <td style="width: 12.5%;">X5</td> <td style="width: 12.5%;">X4</td> <td style="width: 12.5%;">X3</td> <td style="width: 12.5%;">EB2</td> <td style="width: 12.5%;">EB1</td> <td style="width: 12.5%;">EB0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>1</td> <td>0</td> <td style="text-align: center;"><span style="color: yellow;">0</span></td> <td style="text-align: center;"><span style="color: green;">0</span></td> <td style="text-align: center;">-</td> </tr> </table>							X7	X6	X5	X4	X3	EB2	EB1	EB0	0	1	0	1	0	<span style="color: yellow;">0</span>	<span style="color: green;">0</span>	-	Condition True  a)SWAP (EB2,EB1) b)EB0 out								
X7	X6	X5	X4	X3	EB2	EB1	EB0																									
0	1	0	1	0	<span style="color: yellow;">0</span>	<span style="color: green;">0</span>	-																									
<i>End of Cycle 1 and Encoded bit EB0</i> <span style="color: red; font-size: 2em; vertical-align: middle;">1</span>																																
<i>Cycle 2</i>																																
Step2.1 C2=0 C1=1	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 12.5%;">-</td> <td style="width: 12.5%;">X7</td> <td style="width: 12.5%;">X6</td> <td style="width: 12.5%;">X5</td> <td style="width: 12.5%;">X4</td> <td style="width: 12.5%;">EB2</td> <td style="width: 12.5%;">EB1</td> <td style="width: 12.5%;">EB0</td> </tr> <tr> <td>-</td> <td>0</td> <td>1</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> </tr> </table>							-	X7	X6	X5	X4	EB2	EB1	EB0	-	0	1	0	1	0	0	0	New 3bit buffer data								
-	X7	X6	X5	X4	EB2	EB1	EB0																									
-	0	1	0	1	0	0	0																									
Step2.2 C2=0 C1=1	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 12.5%;">-</td> <td style="width: 12.5%;">X7</td> <td style="width: 12.5%;">X6</td> <td style="width: 12.5%;">X5</td> <td style="width: 12.5%;">X4</td> <td style="width: 12.5%;">EB2</td> <td style="width: 12.5%;">EB1</td> <td style="width: 12.5%;">EB0</td> </tr> <tr> <td>-</td> <td>0</td> <td>1</td> <td>0</td> <td>1</td> <td style="text-align: center;"><span style="color: green;">0</span></td> <td style="text-align: center;"><span style="color: yellow;">0</span></td> <td style="text-align: center;"><span style="color: red;">0</span></td> </tr> </table>							-	X7	X6	X5	X4	EB2	EB1	EB0	-	0	1	0	1	<span style="color: green;">0</span>	<span style="color: yellow;">0</span>	<span style="color: red;">0</span>	Condition check  ((EB2 .EQ. EB0)) .AND. (.NEQ. EB1)								
-	X7	X6	X5	X4	EB2	EB1	EB0																									
-	0	1	0	1	<span style="color: green;">0</span>	<span style="color: yellow;">0</span>	<span style="color: red;">0</span>																									
Step2.3 C2=0 C1=1	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 12.5%;">-</td> <td style="width: 12.5%;">X7</td> <td style="width: 12.5%;">X6</td> <td style="width: 12.5%;">X5</td> <td style="width: 12.5%;">X4</td> <td style="width: 12.5%;">EB2</td> <td style="width: 12.5%;">EB1</td> <td style="width: 12.5%;">EB0</td> </tr> <tr> <td>-</td> <td>0</td> <td>1</td> <td>0</td> <td>1</td> <td style="text-align: center;"><span style="color: green;">0</span></td> <td style="text-align: center;"><span style="color: yellow;">0</span></td> <td style="text-align: center;">-</td> </tr> </table>							-	X7	X6	X5	X4	EB2	EB1	EB0	-	0	1	0	1	<span style="color: green;">0</span>	<span style="color: yellow;">0</span>	-	Condition False  a)No SWAP (EB2,EB1) b)EB0 out								
-	X7	X6	X5	X4	EB2	EB1	EB0																									
-	0	1	0	1	<span style="color: green;">0</span>	<span style="color: yellow;">0</span>	-																									
<i>End of Cycle 2 and Encoded bit E B0</i> <span style="color: red; font-size: 2em; vertical-align: middle;">0</span>																																







### III. DECODER

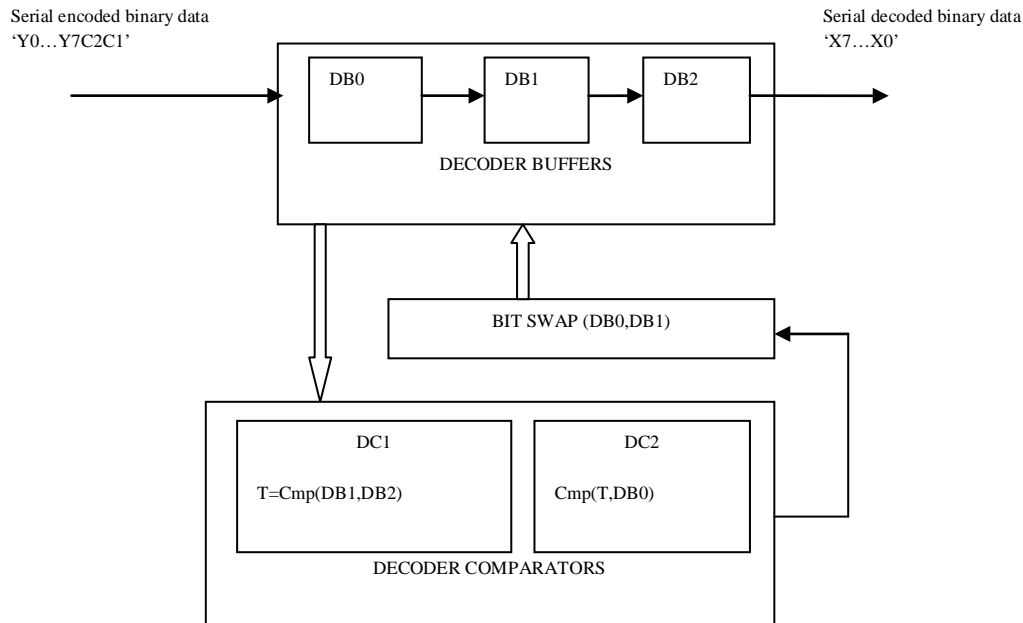


Fig. 3: Block Diagram of Decoder

The block diagram of the decoder is shown in Fig.3. The encoded serial binary data 'Y0---- Yn-1C2C1' is given to the input of the decoder by means of single link. Initially, the values of control bits C1 and C2 are identified and also prevented to enter into buffers. The C1 and C2 values and corresponding operations are as follows.

00 - Swap operations are not necessary.

01 - Swap operations have been performed in 'Y3Y4Y5Y6Y7' sequence.

10 - Swap operations have been performed in 'Y0Y1Y3Y4Y5' sequence.

11 - Swap operations have been performed in entire Y sequence.

The sequence ('Y0 ---- Yn-1) has been entered into the earliest buffer DB0 of the buffer division and at regular interval(s), the bit shifted to right buffers. The process continues and the buffers DB0, DB1 and DB2 occupy the three bit values of 'Y' ('Y5 Y6 Y7') respectively. To begin with shifted and buffered '3'bit data, the bit values of buffers DB1 and DB2 have been forwarded to comparator DC1 through buses. In comparator DC1, the equality

analysis has been executed by comparing the bit values of DB1 and DB2. The further operations depend upon the logical value of the comparator DC1.

In first condition, if both the bit values of the buffers DB1 and DB2 are identical, the corresponding either '0' or '1' bit has been temporally stored to 'T'. Yet again, the equality assessment has been performed by comparing the bit values of 'T' with buffer DB0 in comparator DC2. The swap operation has been performed between DB0 and DB1 if the equality test value is similar and concurrently, the value of DB2 comes outside of the buffer.. In the second condition, if both the bit values of the buffers DB1 and DB2 are not identical, the corresponding either '0' or '1' in DB2 comes outside of the buffer as an encoded bit.

As soon as, after performing any one of above conditions, the bit values of DB0 and DB1 right shifted to DB1 and DB2 correspondingly and the later bit Y4 enters into buffer DB0. This process continues for 'n-2' times and every group of '3' bits in buffers DB0, DB1 and DB2. After 'n-2' times the bit values in DB0 and DB1 have been right shifted and comes out as an encoded data. In the end, the encoded data 'X7 ---- X0' have been generated serially. Fig.4

gives Flow diagram of decoding scheme. Algorithm for proposed decoding technique

Input: Encoded Dataset ( $Y1 = Y0Y1Y2Y3Y4Y5Y6Y7C1$ )

Output: Decoded Dataset ( $X = X7X6X5X4X3X2X1X0$ )

Begin

$n = \text{size}(Y1) - 2$ ; /\* Total number of bits in the sequence\*/

$Y = Y0Y1Y2Y3Y4Y5Y6Y7$

/\* control bits verification\*/

If ( $C1 = 1$  .OR.  $C2 = 1$ ) then

/\*transition check in bits sequence\*/

for  $i = 0$  to  $n - 3$  step 1

if  $(y_{i+1} = y_{i+2}) \neq y_i$  then

swap ( $y_i, y_{i+1}$ );

update the Y sequences;

end if

end for

end if

/\*decoded data\*/

$Y = [\text{updated Y sequences}]$

end

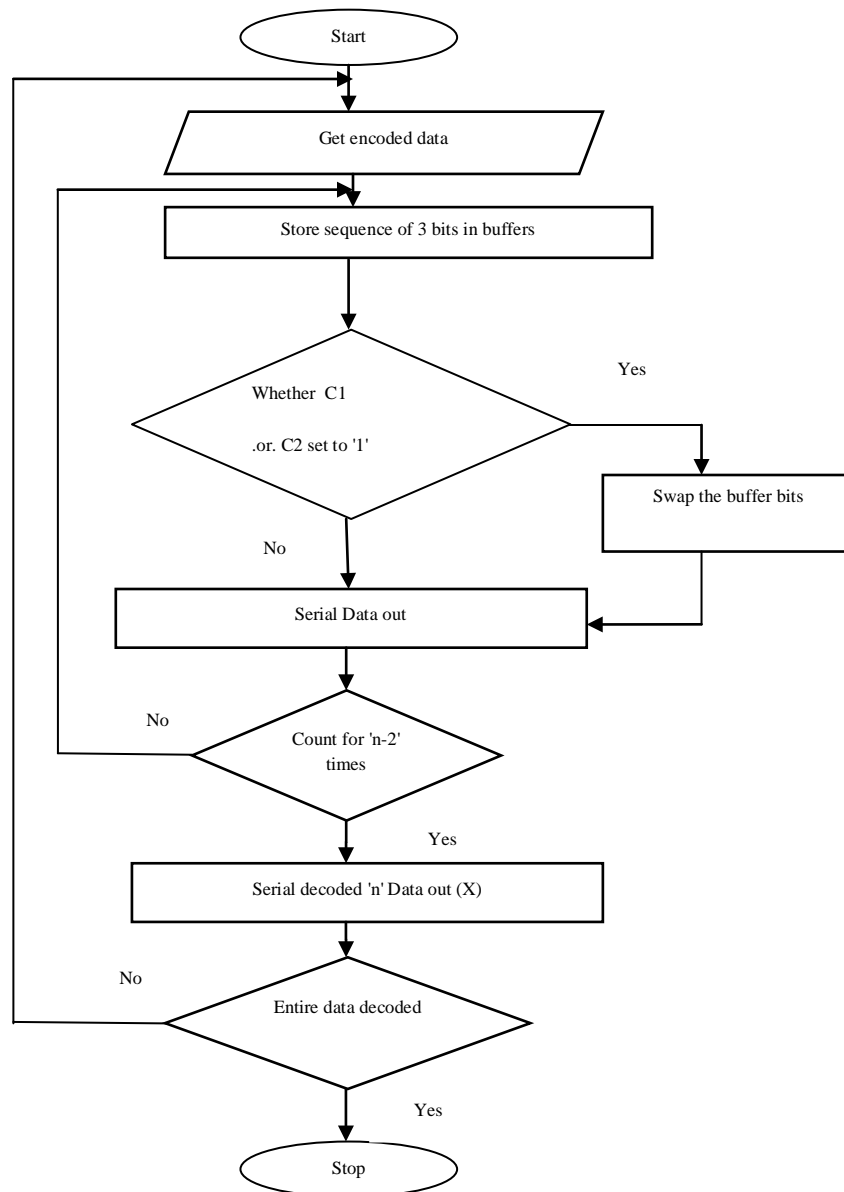














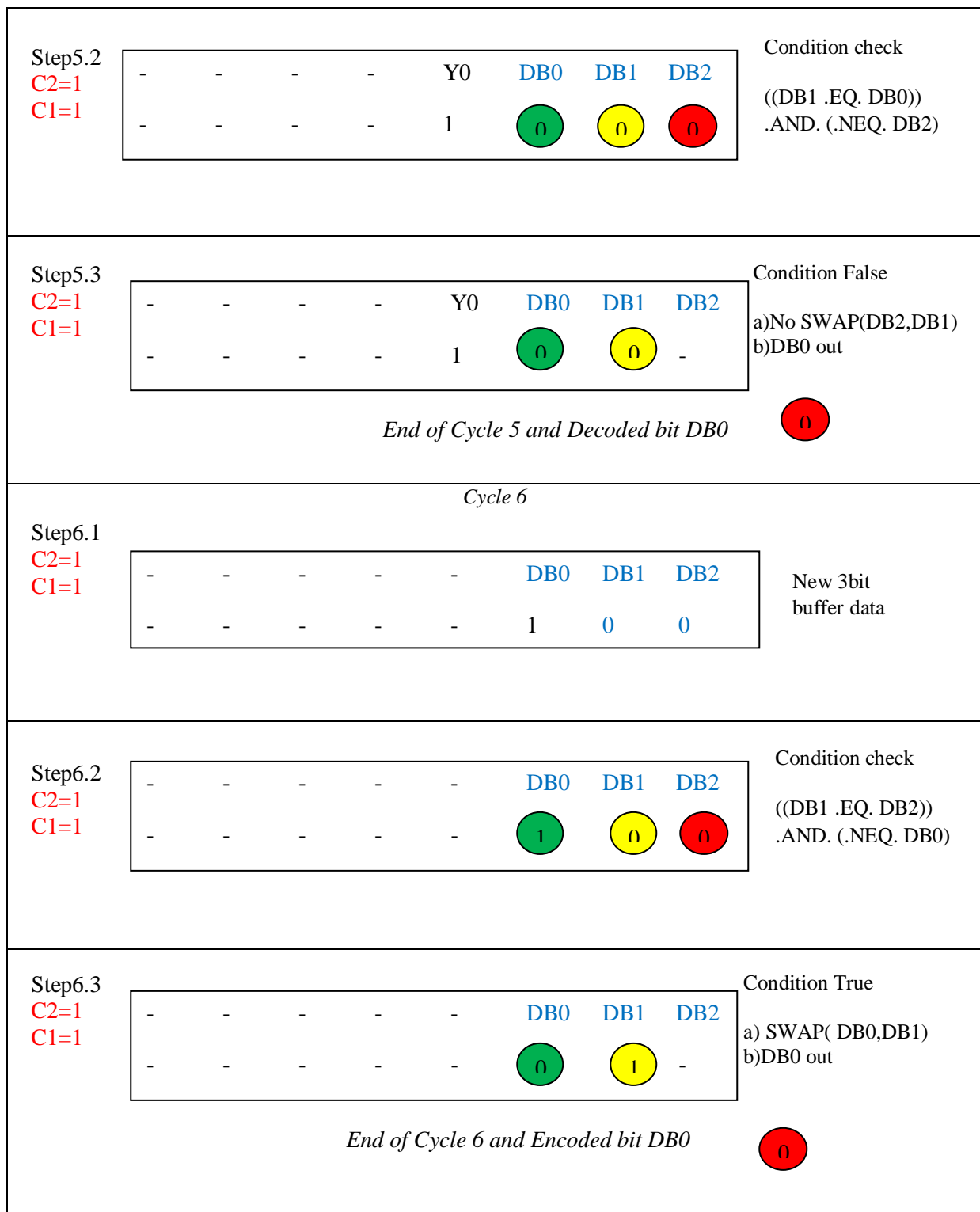
Fig. 4: Flow Diagram of Decoding Scheme

The procedure involved in encoder operations are given in the example.

**Example Decoder**

Encoded Data- 10010100(Y0...Y7)C2C1									
C2=1 C1=1	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	8bit binary encoded data
	1	0	0	1	0	1	0	0	
<i>Cycle 1</i>									
Step1.1 C2=1 C1=1	Y0	Y1	Y2	Y3	Y4	DB0	DB1	DB2	3bit buffer data
	1	0	0	1	0	1	0	0	
Step1.2 C2=1 C1=1	Y0	Y1	Y2	Y3	Y4	DB0	DB1	DB2	Condition check  ((DB1 .EQ. DB2)) .AND. (.NEQ. DB0)
	1	0	0	1	0				
Step1.3 C2=1 C1=1	Y0	Y1	Y2	Y3	Y4	DB0	DB1	DB2	Condition True  a)SWAP (DB0,DB1) b)DB0 out
	1	0	0	1	0			-	
<i>End of Cycle 1 and Decoded bit DB0</i> 									
<i>Cycle 2</i>									
Step2.1 C2=1 C1=1	-	Y0	Y1	Y2	Y3	DB0	DB1	DB2	New 3bit buffer data
	-	1	0	0	1	0	0	1	
Step2.2 C2=1 C1=1	-	Y0	Y1	Y2	Y3	DB0	DB1	DB2	Condition check  ((DB1 .EQ. DB2)) .AND.(.NEQ. DB0)
	-	1	0	0	1				
Step2.3 C2=1 C1=1	-	Y0	Y1	Y2	Y3	DB0	DB1	DB2	Condition False  a)No SWAP(DB2,DB1) b)DB0 out
	-	1	0	0	1			-	
<i>End of Cycle 2 and Decoded bit DB0</i> 									

<i>Cycle 3</i>									
Step3.1 C2=1 C1=1	-	-	Y0	Y1	Y2	DB0	DB1	DB2	New 3bit buffer data
	-	-	1	0	0	1	0	0	
Step3.2 C2=1 C1=1	-	-	Y0	Y1	Y2	DB0	DB1	DB2	Condition check  ((DB1 .EQ. DB2)) .AND. (.NEQ. DB0)
	-	-	1	0	0	<span style="background-color: green; border-radius: 50%; padding: 2px;">1</span>	<span style="background-color: yellow; border-radius: 50%; padding: 2px;">0</span>	<span style="background-color: red; border-radius: 50%; padding: 2px;">0</span>	
Step3.3 C2=1 C1=1	-	-	Y0	Y1	Y2	DB0	DB1	DB2	Condition True  a) SWAP(DB0,DB1) b)DB0 out
	-	-	1	0	0	<span style="background-color: green; border-radius: 50%; padding: 2px;">0</span>	<span style="background-color: yellow; border-radius: 50%; padding: 2px;">1</span>	-	
<i>End of Cycle 3 and Decoded bit DB0</i>									<span style="background-color: red; border-radius: 50%; padding: 2px;">0</span>
<i>Cycle 4</i>									
Step4.1 C2=1 C1=1	-	-	-	Y0	Y1	DB0	DB1	DB2	New 3bit buffer data
	-	-	-	1	0	0	0	1	
Step4.2 C2=1 C1=1	-	-	-	Y0	Y1	DB0	DB1	DB2	Condition check  ((DB1 .EQ. DB2)) .AND. (.NEQ. DB0)
	-	-	-	1	0	<span style="background-color: green; border-radius: 50%; padding: 2px;">0</span>	<span style="background-color: yellow; border-radius: 50%; padding: 2px;">0</span>	<span style="background-color: red; border-radius: 50%; padding: 2px;">1</span>	
Step4.2 C2=1 C1=1	-	-	-	Y0	Y1	DB0	DB1	DB2	Condition False  a) No SWAP (DB0,DB1) b)DB0 out
	-	-	-	1	0	<span style="background-color: green; border-radius: 50%; padding: 2px;">0</span>	<span style="background-color: yellow; border-radius: 50%; padding: 2px;">0</span>	-	
<i>End of Cycle4 and Decoded bit DB0</i>									<span style="background-color: red; border-radius: 50%; padding: 2px;">1</span>
<i>Cycle 5</i>									
Step5.1 C2=1 C1=1	-	-	-	-	Y0	DB0	DB1	DB2	New 3bit buffer data
	-	-	-	-	1	0	0	0	



#### IV. RESULTS AND DISCUSSION

The proposed RRMP scheme has been implemented and evaluated for its performance using the Modelsim Simulator and the target device FPGA. The execution of the process has been experimented with various data format and power computation, area and the time complexity have been

produced. The encoder and decoder have been coded in VHDL, and simulated using Modelsim simulation tool. Xilinx Integrated Software Environment (ISE) 13.2 software tool is for synthesis purpose.

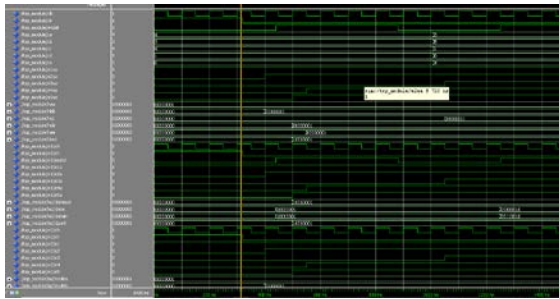


Fig. 5: Simulation Waveform for the Encoding and Decoding Modules

Fig. 5 shows the simulation waveform of the proposed encoder for input data, and corresponding output data. The input data is given to the input of the encoder and the encoder stage recognize the number of 0 to 1 transition, if the number of transition is consecutive, then swap function is performed as per the proposed algorithm. Table1 gives the device utilization summary of encoder and decoder and Table 2 gives various Comparison techniques.

Table 1: Device Utilization Summary

Device Part	Encoding	Decoding
Number of Slices	100 out of 11640 1%	97 out of 11640 1%
Number of Slice Flip Flops	3 out of 93 3%	3 out of 93 3%
Number of 4 input LUTs	226 out of 17222 1%	197 out of 17222 1%
Number of bonded IOBs	1 out of 32 3%	1 out of 32 3%

Device utilization report gives the percentage of slices, flip-flops, input LUTs and adjacent IOBs used in the utilization by the device hardware for the chip implementation. Device hardware includes number of

Table 2: Comparison of Various Techniques

S.No	Parameter	DBI	SILENT	POWER PROTOCOL	PROPOSED RRMP
1	Number of 4 input LUTs	734	634	583	423
2	Dynamic power consumption	50mW	43mW	37mW	21mW
3	Maximum time delay	27.123ns	25.556ns	19.953ns	7.492ns
4	Maximum frequency	500MHz	600MHz	750MHz	753.434MHz

## V. CONCLUSION

This paper introduces a novel encoding system to limit the number of moves in the serial connection of NoC switch to diminish efficient power utilization. The simulation result shows the possible reduction in power distribution than other existing systems. The proposed RRMP scheme has been simulated using the Modelsim Simulator tool and synthesized using Xilinx ISE 13.2. The encoder and decoder have been coded in VHDL, and simulated using Modelsim simulation tool. The proposed RRMP scheme is targeting on Xilinx Virtex-6 XC6VLX75T device. In this paper, self-switching activity action of the communication was considered. Additional two control lines causes a little area and power overhead, and it requires extra consideration. Future research will concentrate on reducing power and area overhead by adopting appropriate methods.

## REFERENCES

- [1] U.Y. Ogras, R. Marculescu, D. Marculescu and E.G. Jung, "Design and management of voltage-frequency island partitioned networks-on-chip", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.17, No.3, Pp.330-341, 2009.
- [2] S. Kiamehr, M. Ebrahimi, M.S. Golanbari and M.B. Tahoori, "Temperature-Aware Dynamic Voltage Scaling to Improve Energy Efficiency of Near-Threshold Computing", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.25, No.7, 2017.
- [3] H. Zheng, J. Lin, Z. Zhang and Z. Zhu, "Decoupled dimm: Building high-bandwidth memory system using low-speed dram devices", ISCA, 2009.
- [4] H. David, C. Fallin, E. Gorbatov, U.R. Hanebutte and O. Mutlu, "Memory power management via dynamic voltage/frequency scaling", ICAC, 2011.
- [5] K.T. Malladi, B.C. Lee, F.A. Nothaft, C. Kozyrakis, K. Periyathambi and M. Horowitz, "Towards energy-proportional datacenter memory with mobile dram", ISCA, 2012.

- [6] K. Basu, A. Choudhary, J. Pisharath and M. Kandemir, "Power protocol: reducing power dissipation on off-chip data buses", MICRO, 2002.
- [7] J. Yang, R. Gupta and C. Zhang, "Frequent value encoding for low power data buses", ACM Trans. Des. Autom. Electron. Syst., Vol. 9, 2004.
- [8] M. Stan and W. Burleson, "Bus-invert coding for low-power I/O", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 3, No. 1, 1995.
- [9] K. Lee, S.J. Lee and H.J. Yoo, "Silent: serialized low energy transmission coding for on-chip interconnection networks", ICCAD, 2004.
- [10] K. Sohn, T. Na, I. Song, Y. Shim, W. Bae, S. Kang, D. Lee, H. Jung, S. Hyun, H. Jeoung and K.W. Lee, "A 1.2 V 30 nm 3.2 Gb/s/pin 4 Gb DDR4 SDRAM with dual-error detection and PVT-tolerant data-fetch scheme", IEEE Journal of Solid-State Circuits, Vol.48, No.1, Pp.168-177, 2013.
- [11] T.Y. Oh, H. Chung, J.Y. Park, K.W. Lee, S. Oh, S.Y. Doo, H.J. Kim, C. Lee, H.R. Kim, J.H. Lee and J.I. Lee, "A 3.2 gbps/pin 8 gbit 1.0 v lpddr4 sdram with integrated ecc engine for sub-1 v dram core operation", IEEE Journal of Solid-State Circuits, Vol.50, No.1, Pp.178-190, 2015.
- [12] S.J. Bae, K.I. Park, J.D. Ihm, H.Y. Song, W.J. Lee, H.J. Kim, K.H. Kim, Y.S. Park, M.S. Park, H.K. Lee and S.Y. Bang, "An 80 nm 4 Gb/s/pin 32 bit 512 Mb GDDR4 graphics DRAM with low power and low noise data bus inversion", IEEE Journal of Solid-State Circuits, Vol.43, No.1, Pp.121-131, 2008.
- [13] G. Pekhimenko, V. Seshadri, O. Mutlu, P.B. Gibbons, M.A. Kozuch and T.C. Mowry, "Base-delta-immediate compression: Practical data compression for on-chip caches", Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques, 2012.
- [14] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (cam) circuits and architectures: a tutorial and survey", IEEE Journal of Solid-State Circuits, Vol. 41, No. 3, 2006.
- [15] D. Suresh, B. Agrawal, J. Yang and W. Najjar, "Tunable and energy efficient bus encoding techniques", IEEE Transactions on Computers, Vol. 58, No. 8, 2009.
- [16] Y. Aghaghi, F. Fallah and M. Pedram, "Irredundant address bus encoding for low power", International Symposium on Low Power Electronics and Design, 2001.
- [17] J. Yang and R. Gupta, "Fv encoding for low-power data i/o", International Symposium on Low Power Electronics and Design, 2001.
- [18] C. Liu, A. Sivasubramaniam and M. Kandemir, "Optimizing bus energy consumption of on-chip multiprocessors using frequent values", Proceedings. 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2004.
- [19] M. Mamidipaka, D. Hirschberg and N. Dutt, "Adaptive low-power address encoding techniques using self-organizing lists", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 11, No. 5, 2003.
- [20] K. Nakamura and M. Horowitz, "A 50% noise reduction interface using low-weight coding", Symposium on VLSI Circuits, Digest of Technical Papers, 1996.