Low Power Data Encoding Schemes in LDPC Applications

[1]V.HANUMANTH GOUD
[1]*Student, M.Tech(VLSI Design),*
SRM University, Chennai, India

[2]Mrs. MARIA JOSSY
[2]*Assistant Professor (O.G),*
Department of Electronics and Communication/VLSI Design, SRM University, Chennai, India

***Abstract*— As technology improves, the power dissipated by the links of a network-on-chip(NoC) starts to compete with the power dissipated by the other elements of the communicate ion subsystem, namely, the routers and the network interfaces (NIs). Here, we present a set of data encoding schemes to reduce the power dissipated by the links of an NoC. In this paper, the encoder in LDPC is replaced with our data encoding schemes in order to reduce the power consumption in Low Density Parity Check Techniques. Experiments carried out on both synthetic and real traffic scenarios show the effectiveness of the proposed schemes, which allow saving up to 27% of power dissipation.**

***Index Terms- D*ata encoding, Interconnection on chip, Low density parity check, Majority logic decoding, Power analysis.**

1.INTRODUCTION

1) Data Encoding Techniques

The data encoding techniques are developed to reduce the power consumption caused by the transitions in the interconnect on the chip[1]. The data encoding techniques are based on reducing the number of transitions by considering the types of transitions in the interconnects and by considering them as discussed in the table below and also consider the transitions as different types of inversions available for us.

The different types of inversions available for us are odd inversion, full inversion and even inversions. By reducing these inversions we can control the number of transitions in the interconnects which reduces the power consumption caused by these transitions in the links.

TABLE I:Table shows transitions versus types

Time	Normal			Odd Inverted		
	Type I			Types II, III, and IV		
$t-1$	00, 11	00, 11, 01, 10	01, 10	00, 11	00, 11, 01, 10	01, 10
t	10, 01	01, 10, 00, 11	11, 00	11, 00	00, 11, 01, 10	10, 01
	T1*	T1**	T1***	Type III	Type IV	Type II
$t-1$	Type II			Type I		
t	01, 10			01, 10		
	10, 01			11, 00		
$t-1$	Type III			Type I		
t	00, 11			00, 11		
	11, 00			10, 01		
$t-1$	Type IV			Type I		
t	00, 11, 01, 10			00, 11, 01, 10		
	00, 11, 01, 10			01, 10, 00, 11		

By assuming these we have designed three schemes which are as follows:

A .Scheme I

In scheme I, we focus on reducing the numbers of Type I transitions (by converting them to Types III and IV transitions) and Type II transitions (by converting them to Type I transition). The scheme compares the current data with the previous one to decide whether

ISSN: 2278 – 7798

odd inversion or no inversion of the current data can lead to the link power reduction[1].

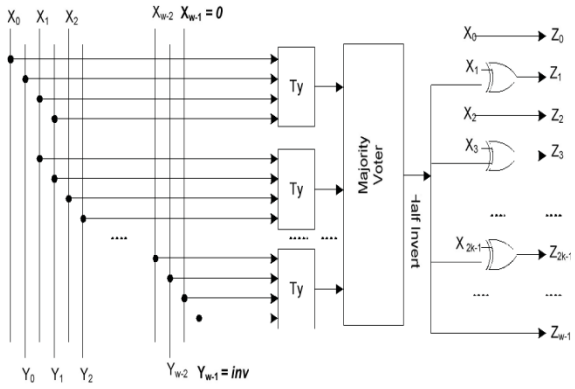


Fig.1. Encoder architecture scheme I. (a) Circuit diagram [27]. (b) Internal view of the encoder block.

B. Scheme II

In the proposed encoding scheme II, we make use of both odd (as discussed previously) and full inversion[2]. The full inversion operation converts Type II transitions to Type IV transitions. The scheme compares the current data with the previous one to decide whether the odd, full, or no inversion of the current data can give rise to the link power reduction.[2]

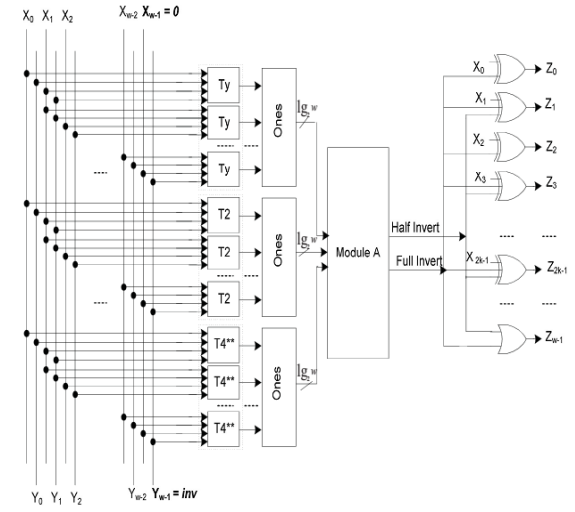


Fig. 2 **INTERNAL VIEW OF ENCODER**

C. Scheme III

In the proposed encoding Scheme III, we add even inversion to Scheme II. The reason is that odd inversion converts some of Type I (*T*1***) transitions to Type II transitions. As can be observed from Table II, if the flit is even inverted, the transitions indicated as *T*** 1 / *T*1*** in the table are converted to Type IV/Type III transitions. Therefore, the even inversion may reduce the link power dissipation as well[2]. The scheme compares the current data with the previous one to decide whether odd, even, full, or no inversion of the current data can give rise to the link power reduction.

2) Low Density Parity Check Code:

The LDPC code is based on a set of one or more fundamental LDPC codes.[2] Each of the fundamental codes is a systematic linear block code. The fundamental codes can accommodate various code rates and packet sizes.

Each LDPC code in the set of LDPC codes is defined by a matrix **H** of size *m*-by-*n*, where *n* is the length of the code and *m* is the number of parity check bits in the code.[3] The number of systematic bits is $k=n-m$.

The matrix **H** is defined as

$$\mathbf{H}=\begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} & \cdots & \mathbf{P}_{0,n_b-2} & \mathbf{P}_{0,n_b-1} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \cdots & \mathbf{P}_{1,n_b-2} & \mathbf{P}_{1,n_b-1} \\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2} & \cdots & \mathbf{P}_{2,n_b-2} & \mathbf{P}_{2,n_b-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ \mathbf{P}_{m_b-1,0} & \mathbf{P}_{m_b-1,1} & \mathbf{P}_{m_b-1,2} & \cdots & \mathbf{P}_{m_b-1,n_b-2} & \mathbf{P}_{m_b-1,n_b-1} \end{bmatrix}=\mathbf{P}^{H_b}$$

where $\mathbf{P}_{i,j}$ is one of a set of *z*-by-*z* permutation matrices or a *z*-by-*z* zero matrix.

The encoding of a packet at the transmitter generates parity-check bits $\mathbf{p}=(p_0, \ldots, p_{m-1})$ based on an information block $\mathbf{s}=(s_0, \ldots, s_{k-1})$, and transmits the parity-check bits along with

ISSN: 2278 – 7798

the information block. Because the current symbol set to be encoded and transmitted is contained in the transmitted codeword,[3] the information block is also known as systematic bits. The encoder receives the information block $\mathbf{s}=(s_0, \ldots, s_{k-1})$ and uses the matrix $\mathbf{H}_{bm}$ to determine the parity-check bits. The expanded matrix **H** is determined from the model matrix $\mathbf{H}_{bm}$. Since the expanded matrix **H** is a binary matrix, encoding of a packet can be performed with vector or matrix operations conducted over GF.

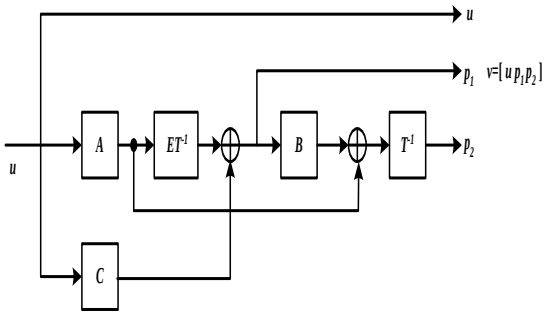


Fig. 3 Block diagram of the encoder architecture for the block LDPC code

Using two decoding techniques majority logic decoding and majority logic decoder/detector in LDPC to reduce the delay occurring in these techniques.

3) Majority Logic Decoding:

Majority-logic decoding is a simple and effective scheme for decoding certain classes of block codes,[4] especially for decoding certain classes of cyclic codes. Majority logic decoding is a method to decode repetition codes, based on the assumption that the largest number of occurrences of a symbol was the transmitted symbol. It will increase the power consumption. Syndrome vector is oldest technology, which is used to detect the error in the code word. Hamming code is one of the examples of syndrome decoder.

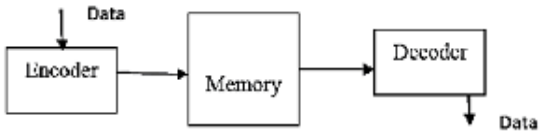


Fig.4 Simple memory system schematic

4) Majority Logic detector/decoder :

The ML detector/decoder (MLDD) has been implemented using the Euclidean Geometry LDPC. G-LDPC codes there is an subclass of codes that is one step majority logic decodable (MLD) This method is very practical to generate and check all possible error combinations for codes with small words and affected by a small number of bit flips[4]. When the size of code and the number of bit flip increases, it is difficult to exhaustively test all possible combinations. Therefore the simulations are done in two ways, the error combinations are exhaustively checked when it is feasible and in the rest of the causes the combinations are checked randomly. Since it is convenient to first describe the chosen design and also for simplicity.

I. Data Encoding Techniques in LDPC

These data encoding schemes which are discussed in the above sections are replaced by these encoding schemes instead of the encoder in the LDPC block:

1. Scheme1 encoding technique in LDPC.
2. Scheme2 encoding technique in LDPC.
3. Scheme3 encoding technique in LDPC.

1) Scheme1 encoding technique in LDPC:

The proposed encoding architecture, which is based on the odd invert condition defined is shown in Fig. 1. We consider a link

ISSN: 2278 – 7798

width of *w* bits[5]. If no encoding is used, the body flits are grouped in *w* bits by the NI and are transmitted via the link. In our approach, one bit of the link is used for the inversion bit, which indicates if the flit traversing the link has been inverted or not.

This encoding technique which is scheme 1 of our encoding techniques in used in low density parity check of majority logic decoding are replaced with our encoding scheme1 technique.

2) Scheme2 encoding technique in LDPC:

The principles of this encoder are similar to those of the encoder implementing Scheme I. The proposed encoding architecture, which is based on the odd invert condition and the full invert condition, is shown in Fig. 2[5]. Here again, the *w*th bit of the previously and the full invert operating condition is shown in Fig. 2. Here again, the *w*th bit of the previously encoded body flit is indicated with inv which defines if it was odd or full inverted (inv = 1) or left as it was (inv = 0). In this encoder, in addition to the *Ty* block in the Scheme I encoder, we have the *T*2 and *T*4** blocks which determine if the inversion based on the transition types *T*2 and *T*4** should be taken place for the link power reduction[5]. The second stage is formed by a set of 1s blocks which count the number of 1s in their inputs. The output of these blocks has the width of log2 *w*. The output of the top 1s block determines the number of transitions that odd inverting of pair bits leads to the link power reduction. The middle 1s block identifies the number of transitions whose full inverting of pair bits leads to the link power reduction. Finally, the bottom 1s block specifies the number of transitions whose full inverting of pair bits leads to the increased link power. The encoding technique discussed above which is scheme2[5] which is based on the logic of full inversion which is the proposed method of the odd inversion which is scheme1[5]. The technique which is mentioned above is LDPC using sheme1 now is replaced with scheme2 in the LDPC and thereby reducing some amount of power compared from scheme1.

3) Scheme3 encoding technique in LDPC:

The operating principles of this encoder are similar to those of the encoders implementing Schemes I and II. The proposed encoding architecture,[6] which is based on the even invert condition of (28),the full invert condition of (29), and the odd invert condition of (30), is shown in Fig. 4.

The (inv = 0). The first stage of the encoder determines the transition types while the second stage is formed by a set of 1s blocks which count the number of ones in their inputs. In the first stage, we have added the *Te* blocks which determine if any of the transition types of *T*2, *T*1**, and *T*1*** is detected for each pair bits of their inputs[6]. For these transition types, the even invert action yields link power reduction.

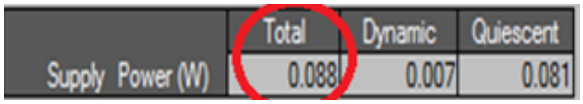
Again, we have four Ones blocks to determine the number of detected transitions for each *Ty*, *Te*, *T*2, *T*4** blocks[6]. The output of the Ones blocks are inputs for Module C. This module determines if odd, even, full, or no invert action corresponding to the outputs “10,”, “01,” “11,” or “00,” respectively, should be performed.

The encoding technique discussed above which is scheme3 which is based on the logic of even inversion which is the proposed method of the full inversion which is scheme2[6]. The technique which is mentioned above is ldpc using sheme1 now is replaced with scheme2 in the ldpc and thereby reducing some amount of power compared from scheme2. Hence ,we analyzed that scheme3 is the one which is used in LDPC technique.

ISSN: 2278 – 7798

II. RESULTS

Power report for scheme1:



	Total	Dynamic	Quiescent
Supply Power (W)	0.088	0.007	0.081

Area report for scheme1:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	37	4656	0%
Number of Slice Flip Flops	44	9312	0%
Number of 4 input LUTs	59	9312	0%
Number of bonded IOBs	20	232	8%
Number of GCLKs	1	24	4%

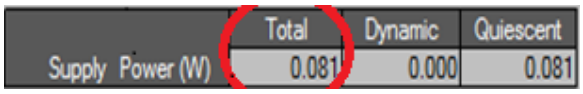
Power report for scheme2:

	Total	Dynamic	Quiescent
Supply Power (W)	0.086	0.005	0.081

Area report for scheme2:

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	29	9,312	1%
Number of 4 input LUTs	16	9,312	1%
Number of occupied Slices	17	4,656	1%
Number of Slices containing only related logic	17	17	100%
Number of Slices containing unrelated logic	0	17	0%
Total Number of 4 input LUTs	16	9,312	1%
Number of bonded IOBs	20	232	8%
Number of BUFGMUXs	1	24	4%

Power report for scheme3:



	Total	Dynamic	Quiescent
Supply Power (W)	0.081	0.000	0.081

Area report for scheme3:

Device Utilization Summary (estimated values)		
Logic Utilization	Used	Available
Number of Slices	19	4656
Number of Slice Flip Flops	29	9312
Number of 4 input LUTs	17	9312
Number of bonded IOBs	20	232
Number of GCLKs	1	24

IV.CONCLUSION

In this paper, we have presented data encoding techniques which are used in the place of encoders in LDPC which reduces the power consumption by eliminating the transitions as discussed before. Here, we analyzed the power consumption for these three schemes and compared their power and area performances.

V. REFERENCES

[1] International Technology Roadmap for Semiconductors. (2011) .

[2] M. S. Rahaman and M. H. Chowdhury, "Crosstalk avoidance and error correction coding for coupled RLC interconnects," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2009, pp. 141–144.

[3] W. Wolf, A. A. Jerraya, and G. Martin, "Multiprocessor system-on-chip MPSoC techn ology ," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 10, pp. 1701–1713, Oct. 2008.

[4] Pedro Reviriego, Juan A. Maestro, and Mark F. Flanagan, "Error Detection in Majority Logic Decoding of Euclidean Geometry Low Density Parity Check (EG-LDPC) Codes," IEEE Trans. Very Large Scale Integra. (VLSI) syst., vol. 21, no. 1, pp.156-159, Jan. 2013.

[5] Youn Sung Park, Yaoyu Tao, Zhengya Zhang, "A 1.15Gb/s Fully Parallel Nonbinary LDPC Decoder with Fine-Grained Dynamic Clock Gating," 2013 IEEE International Solid-State Circuits Conference.

[6] L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm," *Computer, vol. 35, no. 1, pp. 70–78, Jan. 2002.*

ISSN: 2278 – 7798