Waseda University Doctoral Dissertation

Research on Energy-Efficient VLSI Decoder for LDPC Code

Zhixiang CHEN

Graduate School of Information, Production and Systems

Waseda University

September 2012

Abstract

Low-Density Parity-Check code is an error correcting code, invented by Gallager in 1963 and rediscovered by MacKay in 1999. Due to inherently parallel decoding algorithm and excellent coding performance, LDPC code has witnessed tremendous advances in communication systems. It is widely adopted in newest standards, like WLAN (IEEE 802.11.n), WiMAX (IEEE 802.16e), 10GBASE-T (IEEE 802.3an) and WPAN (IEEE 802.15.3c), etc.

A VLSI LDPC decoder mainly contains two types of processing nodes. They are connected by a permutation network (PN), the pattern of which is defined by a parity check matrix (PCM) of a code. Decoders of different parallelisms have various performances. A decoder with higher parallelism achieves higher throughput while implemented chip silicon area becomes bigger since a larger size of circuit is included. Moreover, for many application cases, multiple code lengths and rates are supported in one decoder. Therefore, it challenges designers to find good architecture which fulfills the demands as throughput, silicon area and multi-code configurability. One of the most important metrics for evaluating VLSI architecture is energy efficiency, i.e. electrical energy used for one decoded bit. Therefore, this dissertation targets high energy-efficiency LDPC decoder architecture design with constrains on requirements of throughput, area and multi-code configurability.

As fully parallel architecture, the first decoder is presented by Blanksby (CICC'01) in 2001. In 2007, Darabiha (CICC'07) presents architecture with bit-serial PN and early termination (ET) to achieve better energy efficiency. In 2010, Hung (A-SSCC'10) gives the first multi-rate (WPAN codes) decoder with techniques on efficient PN.

As partially parallel architecture, Mansour (JSSC'06) presents prototype design for codes of both various lengths and rates in 2006. Later in 2008, Liu (JSSC'08) presents a decoder with an efficient PN for WiMAX code. In 2008, Oh (ISCAS'08) introduces a PN based on Benes network for QC-LDPC code aiming at better decoder energy efficiency.

This dissertation also focuses on energy-efficient VLSI decoder architecture for LDPC code. Contributions are given to both fully and partially parallel decoder architectures. As the partially parallel decoder for WiMAX and WiFi, a four-way–supporting PN and a coding performance lossless early termination (ET) algorithm are proposed. As the fully parallel decoder for WPAN, new decoder architecture with an efficient permutation scheme is proposed to improve the energy efficiency.

The dissertation contains the following six chapters.

Chapter 1 [Introduction] gives the general introduction of this dissertation by firstly pointing out advancement and challenges of LDPC coding system. Particularly, attention to design of energy efficient VLSI LDPC decoder is paid. Scope and purpose of this dissertation is given followed by a brief on related work. Finally, organization of the dissertation is presented.

Chapter 2 [Background] introduces the fundamentals on LDPC codes, decoding algorithms and architectures. Representations of LDPC and its sub-class Quasi Cyclic (QC) LDPC code are presented. Decoding algorithms and its optimization for VLSI LDPC decoders are described. Finally, two types of decoder architectures are introduced.

Chapter 3 [An Energy-Efficient PN for WiMAX and WiFi] presents a technique on PN used in partially parallel decoder for WiMAX and WiFi LDPC codes.

Efficient PN design for QC-LDPC partially parallel decoder is a challenging problem. It is solved by Oh (ISCAS'08) using Benes network. However, for Oh's PN, each time only one message group can be processed, making maximum three quarters of hardware idle for short code modes, which results in low decoder energy efficiency.

For the proposed four-way-supporting PN in this dissertation, each time up to four message groups can be processed which improves the decoder energy efficiency. From the implementation results at synthesis level, compared to conventional one-way PN, the proposed four-way-supporting PN has 5% hardware overhead. By implementing the proposed PN in a partially parallel decoder, at short code modes (576 bits to 768 bits), the decoder energy efficiency is improved by up to 18.6%.

Chapter 4 [A Lossless ET Algorithm for WiMAX and WiFi] presents a coding performance lossless ET algorithm for WiMAX and WiFi LDPC codes. Because decoding energy consumption is proportional to the number of iterations, reducing iterations introduces energy efficiency improvement. ET is used for iteration reduction by stopping the decoding earlier while guaranteeing the decoding performance. For conventional ET algorithms, such as Shih (JSSC'08), there exists decoding performance degradation and problem that decoding process is stopped too late.

For the ET algorithm proposed in this dissertation, decoding performance degradation is zero in terms lossless bit error rate. (BER) An error control code contains information and redundant bits. Conventional methods conduct processing without differentiating information and redundant bits. For the proposed algorithm in this dissertation, by applying syndrome information accumulation, errors in information bits and in redundant bits are differentiated. During a decoding process on LDPC codes defined in WiMAX and WiFi, information bits converge to correct faster than redundant bits. Based on this fact, a new ET algorithm is proposed in this dissertation, the world

first algorithm targeting only information bits, that it stops decoding as soon as all the information bits are corrected. According to simulation results, a 12% improvement on energy efficiency can be achieved if the proposed ET algorithm is applied in the decoder.

Proposed techniques in this chapter and chapter 3 are both integrated into a WiMAX/WiFi LDPC decoder which is implemented in synthesis level. According to the result, compared to Brack (PIMRC'06), a 28.4% improvement on energy efficiency is achieved for the entire decoder.

Chapter 5 [An Energy-Efficient WPAN LDPC Decoder] presents a novel fully parallel LDPC decoder for WPAN. According to WPAN standard, a 5.775Gb/s throughput and 4 code rates are required for LDPC decoder. Hung (A-SSCC'10) presents a simplified PN based decoder targeting high energy efficiency, where big reduction is not achieved. In this dissertation, a modified PCM based message permutation scheme is used that the PN of the decoder contains no logic circuits. Size of the decoder circuit is reduced by about half because PN contains no logic circuits but wires only. A frame level pipeline stage is needed in the decoder, and considering its hardware overhead, finally a 33.5% reduction on circuit size is achieved.

For the proposed decoder architecture, it is designed, implemented and evaluated in a 65nm CMOS chip. (2.1mm×2.1mm) It achieves gate count 430K, maximum clock frequency 450MHz, energy efficiency 8.0pJ/bit/iteration, and chip density 86.3%. Compared to Hung, (A-SSCC'10) improvements of 33.5%, 125%, 49%, and 17.7% are achieved in terms of gate count, maximum clock frequency, energy efficiency, and chip density, respectively.

Chapter 6 [Conclusion] concludes the dissertation and gives a plan on future work.

Acknowledgement

First of all, I would like to show my sincere gratitude to Professor Satoshi Goto of Waseda University for his continuous support, guidance and encouragement on research work toward my PhD. His enthusiasm on research and passions of life have been impressed me from the beginning to here. It cannot be doubt that working with him would be one of the most valuable assets in my life. I also would like to appreciate Professor Shinjin Kimura and Professor Maehara of Waseda University, and Professor Wada of Ryukyu University for their patient teaching and inspiring advices on my research, and for their valuable comments on my dissertation.

I would like to thank Dr. Xiao Peng, Mr. Xiongxin Zhao and Dr. Dajiang Zhou for their valuable discussions on the topic of LDPC coding and digital circuit deisgn. Specially, I would like to thank Mr. Reona Okamura for giving me guidance and support on chip back-end design. I also would like to thank all the members of Professor Goto's laboratory for bringing me so much fun during my PhD program.

I express my appreciation to Japan Society of the Promotion of Science (JSPS) for their support to my research. I also thank the support from Waseda University Ambient SoC Global COE Program of MEXT, Japan. Special thanks are given to VLSI Design and Education Center (VDEC) at the University of Tokyo, and Semiconductor Technology Academic Research Center (STARC) for their support on chip design and fabrication.

Finally, I thank my families for their kind support over the years.

Contents

List of Tables

List of Figures

1. Introduction

Communication systems are becoming extremely important in today's human society, such as mobile telephony, computer systems and wireless sensors. Error control coding (ECC) plays an important role in communication for providing reliable digital transmission and storage. [1]

Low-Density Parity-Check (LDPC) code is an error control coding scheme, the concept of which was proposed by Gallager in the early of 1963 in his doctorial dissertation.[2] However, LDPC code has been ignored for a long time mainly due to the high complexity decoding algorithm. Due to the advancements in very large scale integration (VLSI) technologies and rediscovery by Mackay in 1999 [3], LDPC code has received lots of attentions recently. It became popular because it demonstrates superior error correction performance under iterative message passing algorithm [3] and algorithm's inherently parallel decoding process. As a result, it has been adopted in various modern communication standards, such as wireless local area network (WLAN, IEEE 802.11n) [4], Worldwide Interoperability for Microwave Access (WiMAX, IEEE 802.16e) [5], 10 gigabit Ethernet (10GBASE-T, IEEE 802.3an) [6] and wireless personal area network. (WPAN, IEEE 802.15.3c) [7]

1.1. Scope of Work

Decoding of LDPC code is often realized by very large scale integration (VLSI) chips, which can achieve data rate up to 1Gbps required by the latest communication systems. Design considerations on VLSI LDPC decoder ranges from

silicon area, latency, throughput and power consumption, etc. Based on the parallelism of node processors, LDPC decoder architecture can be categorized into two types, fully parallel decoder and partially parallel decoder, respectively. Fully parallel decoder delivers higher decoding throughput, occupies bigger silicon area and dissipates larger power while partially parallel decoder is always used for low to moderate throughput, small area and low power applications. Nevertheless, energy efficiency, i.e. electrical energy dissipated for decoding one bit, as one of the most important metric is used to evaluate both two types of the decoders.

This work contributes to the energy-efficient LDPC decoders of both partially parallel and fully parallel architectures.

For partially parallel architecture, this work advances the decoding energy efficiency by two proposals. 1) An improved permutation network supporting four-code concurrent decoding at short code modes, gains energy efficiency by up to 18.6%. [8] 2) A lossless early termination algorithm reduces decoding workload by 12% at most, making a 12% energy efficiency improvement. [9] Combining these two techniques, a partially parallel decoder for WiMAX and WiFi applications is proposed with a logic synthesis level implementation. [10] The hardware overhead with proposed techniques is negligible.

For fully parallel architecture, this work advances the state-of-art energy-efficient WPAN LDPC decoder. [11][12] A novel scheme is proposed to alleviate the complexity of permutation circuitry. The scheme uses modified parity check matrix for permutation, which extremely utilizes the structure of the WPAN code.

As a result, the permutation network is implemented using no logic gates but only wires. Measurement results of ASIC decoder show that the energy efficiency is improved by 55.6%.

1.2. Related Work

In spite of the architectures, an LDPC decoder can be mainly partitioned into three components, a group of check node processors, a group of variable node processors, and a permutation (routing, interconnecting, shuffle used in some literatures) network connecting these two groups of nodes.

As first discussed in [13], implementation of permutation network is one of the biggest challenges of LDPC decoder design. On-chip global wires used to transmit signals between two groups of node processors not only degrade chip utilization ratio but also result in large amount of logic gates which are used as signal buffers. [14] Therefore, designing permutation network is critical for energy-efficient LDPC decoder.

For fully parallel decoder, Darabiha proposed a bit-serial permutation scheme to reduce routing overhead. [15][16] Processed message data is routed bit by bit in multiple cycles so that the number of global wires is reduced by p times, where p is the number of bits to represent one message. However, more clock cycles are needed during a process, which results in a low decoding throughput and energy efficiency. In [17]-[21], Mohsenin presented a row-split idea to reduce the complexity of the permutation network. Original globally interconnected nodes are divided into several local groups for independent processing and an algorithm-optimized minimum amount of information is exchanged between those groups. The idea does reduce the global

interconnection overhead but introduces decoding performance degradations because information is not fully utilized during the process. In [22]-[25], researchers present decoders for WPAN LDPC code. To achieve a 5Gbps throughput, all the designs are of fully parallel architecture. Moreover, to achieve an energy-efficient decoding, code structure based optimization techniques for permutation network are applied resulting in a large reduction of gate count.

For partially parallel decoder, it is firstly proposed with structured LDPC code by Mansour in [26]. Quasi-Cyclic LDPC code as a class of structured LDPC code is invented by Fan in [27] and studied in [28]-[30]. Thanks to the structure feature, permutation network in the decoder for QC-LDPC code is able to be shared by nodes related to various part of a code. Therefore, designing a configurable permutation network became a critical problem. Especially when it needs to support multiple code lengths and rates where cyclical shifts for a data group of variable size are supported, it is much more challenging. In [31], the first Benes network [32] based permutation network for partially parallel decoder was introduced by Olcer. In [33][34], the architecture is improved by Tang and Gunnam, respectively. The first efficient algorithm of switch control for Benes network based permutation network is given by Oh. [35][36] By his algorithm, the control signals for switches can be generated on the fly according to the input configurations. Compared to the previous work gate count and interconnecting wires are greatly reduced which contributes to an improved decoding energy efficiency. In addition to Benes network based architecture, Liu recently proposed a switch network based on self-routing technique in [37], it is the first

architecture that supports a two-way concurrent permutation for QC-LDPC code.

Besides the optimizations on permutation networks for both fully and partially parallel, implementing early termination algorithm in the decoder also improves the decoding energy efficiency. Based on the observation that most of the code blocks converge before a preset number of iteration is reached, criterion for stopping a decoding is designed to save unnecessary decoding process, resulting in energy saving. A straight-forward early termination is that stopping the decoding when the parity checks are all satisfied. [38] It is an equivalent condition to a correct code word which means there is no decoding performance loss when applied. The demerit relies on an implementation overhead for calculating the checks. Darabiha [16] gives a criterion based on a rough estimation on checks using sign bits of variable node messages instead of exact hard decisions. Because the calculations can be handled using the same hardware of check node processors, its hardware overhead is near zero. However, it does introduce some performance loss due to the check estimations. In [39], Shih introduces an early termination algorithm, called hard decision aided (HDA) which is originally applied in turbo decoding [40] to LDPC decoders. If hard decisions in two successive iterations are the same, the decoding stops. To mitigate performance degradation introduced by HDA, a reliability threshold is used for achieving a more confident judgment.

1.3. Organization

Chapter 2 introduces the fundamentals on LDPC codes, decoding algorithms and architectures. Representations of LDPC and its sub-class Quasi Cyclic (QC) LDPC

code are presented. Decoding algorithms and its optimization for VLSI LDPC decoders are described. Finally, two types of decoder architectures are introduced.

Chapter 3 presents a technique on PN in partially parallel decoder for WiMAX and WiFi LDPC codes. For my proposed four-way-supported PN, it is based on Oh's work where I improve the algorithm on switch control for Benes network. The PN using the proposed algorithm is the first PN which can achieve four-way concurrent data permutation. The hardware overhead is negligible which is proved by the synthesis level implementation. The integration to the partially parallel decoder of the proposed PN is given. In analysis, the energy efficiency for the entire decoder for WiMAX and WiFi code at short code modes can be improved up to 18.6%.

Chapter 4 presents a lossless ET algorithm based on syndrome accumulation for WiMAX and WiFi LDPC codes. For my proposed ET algorithm, it achieves no bit error rate performance loss for LDPC decoding. It is the first algorithm that stops decoding as soon as the information bits are corrected. It utilizes the features of bit convergence in WiMAX and WiFi LDPC code and provides energy efficiency improvement for the entire decoder by 12% at most.

Software simulation results and comparisons to the previous works are given for the proposed ET algorithm. Moreover, an energy efficient partially parallel LDPC decoder which applies the techniques proposed in chapter 3 and 4 is presented with a synthesis level implementation in the end of this chapter.

Chapter 5 presents a fully parallel decoder with an efficient message permutation scheme for WPAN LDPC codes. The energy efficiency of the decoder can

be largely improved by using my proposed message permutation scheme. It is the first scheme that uses a modified parity check matrix for message permutation. By extremely utilizing the features of parity check matrix, the proposed scheme based PN contains no logic gates but only wires. The proposed scheme based fully parallel decoder achieves 55.6% improvement in energy efficiency compared to the state-of-art.

An ASIC chip is implemented to verify the proposed technique. Details on chip implementation, measurement and comparisons are given in this chapter.

Chapter 6 concludes the dissertation.

2. Background

In this chapter, background knowledge on LDPC codes, decoding algorithms and decoder architecture is briefly introduced for a better understanding on this dissertation. Firstly, both parity check matrix (PCM) and graph based representations of LDPC code are given. As advancement to original random LDPC code, structure LDPC code which is friendly to VLSI decoder implementation is analysis. Based on the description of these representations, the classic decoding algorithm, namely belief propagation (BP) algorithm is discussed. Followed are the various optimized versions of decoding algorithms, such as min-sum algorithm for simplifying the hardware implementation and layered scheduling for an increased convergence speed. With the knowledge on LDPC code and decoding algorithms, classic decoder architectures are shown by a categorizing of two types, fully parallel and partially parallel architectures.

2.1. LDPC and QC-LDPC Codes

In this section, PCM and Tanner graph based representations of LDPC code are given in 2.1.1. In 2.1.2, Quasi-Cyclic LDPC (QC-LDPC) code [27]-[30] as a structure code and the mainly aimed code in this dissertation is introduced by using an example of WiMAX and WiFi codes.

2.1.1. Representation of LDPC codes

An LDPC code is a kind of linear block code that can be defined by a so called parity-check matrix (PCM) H, the size of which is $M \times N$ where N denotes the code length and M denotes the number of redundant bits in a code frame, respectively. The

code rate of an LDPC code is defined as $R=k/N$ where $k=N-M$ is the number of information bits in a code frame. For LDPC codes, the numbers of ones in columns and rows of its corresponding PCM are much less than those of zeros which is the reason that it is called low-density [41]. To evaluate the density of such an LDPC code, the column weight (row weight), referred as *Wc (Wr)*, is defined as the number of ones in a column (row) of the PCM. If all the column weights and row weights are the same respectively, the LDPC code is called a regular code otherwise it is an irregular one.

Besides the PCM description, a graphical representation of LDPC code, called Tanner Graph, was introduced in [42]. A Tanner Graph is a bipartite graph consisting of two kinds of vertex, called variable node and check node, and a number of edges connecting the nodes. A Tanner Graph of an LDPC code according to its PCM can be constructed by the following rules: each check node C_i (variable node V_j) is defined by ith row (jth column) of the PCM and a pair of check and variable nodes are connected by an edge if and only if the entity in the corresponding row and column in the PCM is one. The messages used in the iterative decoding algorithms which will be talked later are passed along the edges between the two kinds of nodes. An example of these two representations of an LDPC code (N=10, M=5) is shown in Figure 1 and Figure 2 with color marks, respectively.

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Figure 1 Example of an LDPC code Parity check matrix

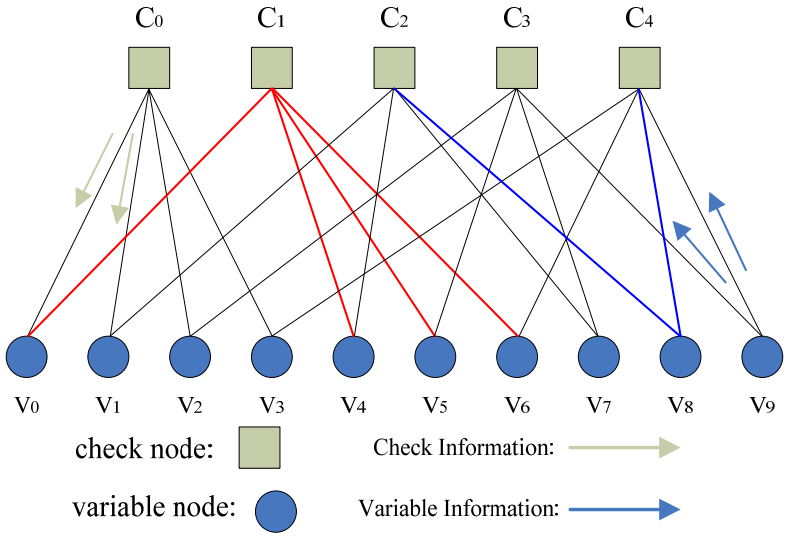


Figure 2 Example of a Tanner Graph

2.1.2. Quasi-Cyclic LDPC Codes

Structured LDPC codes such as quasi-cyclical LDPC (QC-LDPC) codes [27]-[30] have received significant attention since their friendliness to hardware implementation features and comparable BER performance to original randomly generated codes.

LDPC codes in WiMAX and WiFi standards are quasi-cyclic (QC) codes, the

PCM of which is an expansion of a $c \times d$ base matrix expanded by $z \times z$ sub-matrices where z is also called an expansion factor. The sub-matrices consist of cyclically shifted identity matrix and all-zero matrix. The base matrix is composed of nonnegative integers and "-1" ("-" for WiFi) where the nonnegative integer represents a $z \times z$ cyclically shifted identity matrix and "-1" represents a $z \times z$ all-zero matrix. For the cyclically shifted identity matrix, the number of shift equals to its corresponding nonnegative integer. The code length N equals to $z \times d$, where d is the number of columns in a base matrix. An expansion example of z=4 and d=3 is available in Figure 3. For all base matrices in WiMAX and WiFi, d equals to 24. A rate-1/2 base matrix in WiMAX is given in Figure 4.

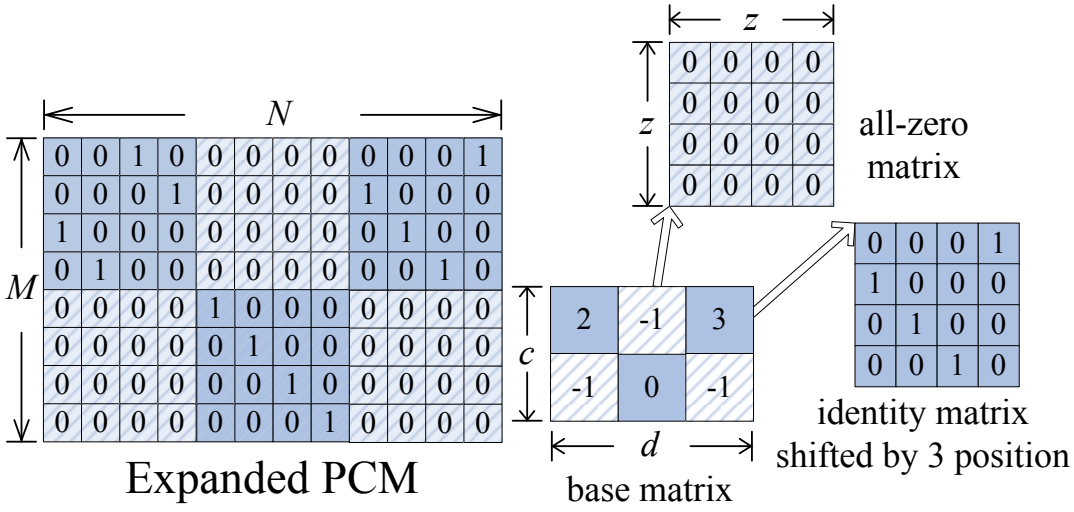


Figure 3 An example of expansion of QC-LDPC code

i \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	-1	94	73	-1	-1	-1	-1	-1	55	83	-1	-1	7	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	27	-1	-1	-1	22	79	9	-1	-1	-1	12	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	-1	-1	-1	24	22	81	-1	33	-1	-1	-1	0	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1
3	61	-1	47	-1	-1	-1	-1	-1	65	25	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1
4	-1	-1	39	-1	-1	-1	84	-1	-1	41	72	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1
5	-1	-1	-1	-1	46	40	-1	82	-1	-1	-1	79	0	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1
6	-1	-1	95	53	-1	-1	-1	-1	-1	14	18	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1
7	-1	11	73	-1	-1	-1	2	-1	-1	47	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1
8	12	-1	-1	-1	83	24	-1	43	-1	-1	-1	51	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1
9	-1	-1	-1	-1	-1	94	-1	59	-1	-1	70	72	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1
10	-1	-1	7	65	-1	-1	-1	-1	39	49	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0
11	43	-1	-1	-1	-1	66	-1	41	-1	-1	-1	26	7	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0

Figure 4 A rate-0.5 base matrix in WiMAX

2.2. Decoding Algorithms

2.2.1. Belief Propagation Algorithm

With the proposal of LDPC code, the first decoding algorithm was also introduced in Gallager's doctorial dissertation [2] in 1963. Until now the existing algorithms for decoding an LDPC code can be categorized into two types, hard or soft decoding algorithms, based on the types of messages input to the decoder, *i.e.* hard and soft decisions, respectively.

Generally, the superior error correction performance of LDPC code is exploited by soft decision based decoding algorithms, among which the most powerful and famous one is the original Gallager's proposal: Belief Propagation (BP) algorithm. BP algorithm is an iterative message passing algorithm which receives the channel soft messages in logarithmic likelihood ratio (LLR) form, passes and updates them between

check and variable processing nodes iteratively. After a maximum number of iterations are achieved, the hard decision for each bit in a code frame is made according to the messages updated and processed. Before further discussion on BP algorithm, some assumption and notation are given below.

Define the column index set $A(i)$ and the row index set $B(j)$ as in (01) and (02), respectively, where H_{ij} denotes the element with row index i and column index j in PCM H.

$$A(i) \triangleq \left\{ j \middle| H_{ij} = 1 \right\} \quad \textbf{(01)}$$

$$B(j) \triangleq \left\{ i \middle| H_{ij} = 1 \right\} \quad \textbf{(02)}$$

In this thesis, binary phase shift keying (BPSK) and additive white Gaussian noise (AWGN) channel are assumed for modulation and transmit channel, respectively. Signal y_j, a real number, represents the received power of bit x_j which is the jth code bit in the original code frame. The conditional probabilities are calculated by (03) and (04).

$$P(y_j \mid x_j = 0) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(y_j - 1)^2}{2\sigma^2}) \quad \textbf{(03)}$$

$$P(y_j \mid x_j = 1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(y_j + 1)^2}{2\sigma^2}) \quad \textbf{(04)}$$

The input of the BP algorithm is the channel initial message, referred as λ_j. As LLR form for the AWGN channel, it can be represented as in (05).

$$\lambda_j = \ln \frac{P(y_j \mid x_j = 0)}{P(y_j \mid x_j = 1)} = \frac{2y_j}{\sigma^2} \quad \textbf{(05)}$$

The BP algorithm performs iterative decoding with updating the following three kinds of messages, which are called a posterior probability (APP) message (APP_j),

check node message (α_{ij}) and variable node message (β_{ij}). After a maximum number, referred as T_{max}, of iterations are achieved, the algorithm outputs the hard decisions of the code frame based on the APP messages. The message updating and calculation can be described by the following steps.

Step 0: Initialization

Set APP message and variable node message β_{ij} according to (06);

Set all check node messages to 0;

Set iteration counter t to 0;

$$APP_j = \beta_{ij} = \lambda_j \quad (0 \le j < N, i \in B(j)) \tag{06}$$

Step 1: Row Operation

For all the check nodes C_i in the order from i=0, 1, 2, …, M-1, compute the check node message α_{ij} according to (07), where function $f(x)$ (called *Gallager Function* in some literatures) in (07) is defined as in (08);

$$\alpha_{ij} = \left(\prod_{j' \in A(i) \setminus j} sign(\beta_{ij'}) \right) \times f\left(\sum_{j' \in A(i) \setminus j} f(\left|\beta_{ij'}\right|) \right) \tag{07}$$

$$f(x) = \ln \frac{\exp(x)+1}{\exp(x)-1} \tag{08}$$

Step 2: Column Operation

For all the variable node V_j in the order from j=0, 1, 2,…, N-1, compute the APP message APP_j and the variable node message β_{ij} according to (09) and (10), respectively;

$$APP_j = \lambda_j + \sum_{i' \in B(j)} \alpha_{i'j} \tag{09}$$

$$\beta_{ij} = \lambda_j + \sum_{i' \in B(j) \backslash i} \alpha_{i'j} \qquad \textbf{(10)}$$

Step 3: Termination and Hard Decision

if $t<T_{max}$-1, set iteration counter $t \leftarrow t+1$ and ***go to*** *step 1*;

else stop the decoding and output the hard decisions according to (11);

$$\hat{x}_j = \begin{cases} 0 & (APP_j \geq 0) \\ 1 & (APP_j < 0) \end{cases} \quad (0 \leq j < N-1) \qquad \textbf{(11)}$$

2.2.2. Optimized Versions on BP Algorithm

2.2.2.1. Min-Sum Algorithm

Although the BP algorithm performs an optimal decoding for ideal LDPC code (no cycles in Tanner Graph), there still exist application limitations in hardware LDPC decoder mainly because the complexity of implementing Gallager Function (8) and calculation of (7) is too high.

Therefore, a tradeoff between error correction performance and hardware overhead was made in [43] with a proposal called Min-Sum (MS) algorithm. The key idea of MS algorithm is using (12) to replace (07) for simplifying the calculation and implementation.

$$\alpha_{ij} = \left(\prod_{j' \in A(i) \backslash j} sign(\beta_{ij'}) \right) \times \min_{j' \in A(i) \backslash j} \{ \left| \beta_{ij'} \right| \} \qquad \textbf{(12)}$$

However, simplification using minimum function introduces big error correction performance degradation. To improve the performance, some correction factors were added to (12). Such works can be found in [44] where a normalized factor

p was multiplied to the right side of (12) as shown in (13) and in [45] where an offset factor q was used as shown in (14). According to the types of correction factors they use, those upgraded versions were called Normalized Min-Sum (NMS) and Offset Min-Sum (OMS) algorithms, respectively.

$$\alpha_{ij} = p \times \left(\prod_{j' \in A(i) \setminus j} sign(\beta_{ij'}) \right) \times \min_{j' \in A(i) \setminus j} \{ \left| \beta_{ij'} \right| \} \tag{13}$$

$$\alpha_{ij} = \left(\prod_{j' \in A(i) \setminus j} sign(\beta_{ij'}) \right) \times \max \{ 0, \min_{j' \in A(i) \setminus j} \{ \left| \beta_{ij'} \right| \} - q \} \tag{14}$$

2.2.2.2. Layered Schedule

Another big improvement on original BP algorithm was the schedule by which the algorithm updates and calculates the messages during the iterative decoding process.

Originally, after row operations (step 1) on all the rows of the PCM, the column operation (step 2) then is started. This message updating schedule was called two phase message passing (TPMP) algorithm or flooding schedule.

Proposed by Mansour in [46], Turbo decoding message passing (TDMP) algorithm was applied on LDPC decoding. For TDMP algorithm, after each row operation the column operation is performed immediately. In fact, the algorithm treats each row of the PCM as a sub-code and performs individual decoding on consecutive sub-codes. Since several rows can be treated as one layer and the algorithm can be applied on each layer, TDMP algorithm is also called layered algorithm. Because the messages are updated much more frequently, the decoding convergence of layered algorithm is much faster than TPMP algorithm which is proved in [47] that it convergences two times faster than TPMP algorithm.

In this dissertation decoding algorithm that I use is the NMS algorithm in section A and layered schedule in section B.

2.3. Decoder Architectures

In this section, both fully parallel and partially parallel decoder architectures are discussed in 2.3.1 and 2.3.2, respectively.

2.3.1. Fully Parallel Decoder Architecture

Figure 5 depicts a fully parallel decoder architecture introduced in [13]. A fully parallel decoding on an LDPC code frame is achieved by implementing *N* column and *M* row operation units which are responsible for message updating and calculation according to the decoding algorithm mentioned above. Messages, categorized as initial message, variable node message and check node message are stored in the corresponding memories. The communication between column and row operation units is via a so called permutation network, the topology of which is exactly the same as the Tanner Graph of the code. After a predefined number of decoding iterations are achieved, the hard decision of the code frame will be made and stored in the hard decision register array for output. All the modules and memories are controlled by a controller in the decoder.

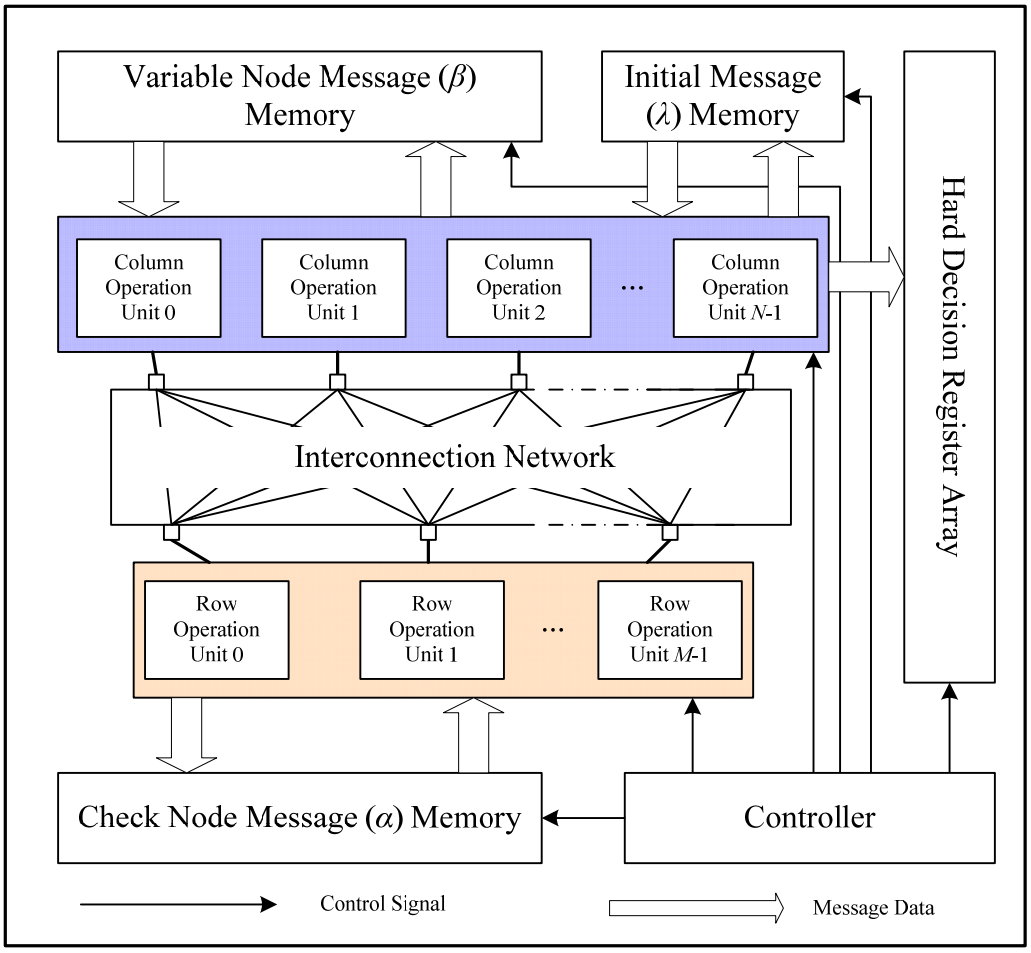


Figure 5 Block diagram of a fully parallel decoder architecture

2.3.2. Partially Parallel Decoder Architecture

Based on layered algorithm and PCM structure of QC-LDPC code, a partially parallel decoder was proposed in [26]. Besides the advantage of high flexibility that supports multiple code lengths and code rates, partially parallel decoder does not have wire congestion problem which dominates the chip area as mentioned before. Recently, based on partially parallel decoder architecture, research work on LDPC decoder [48]-[50] has been done for WiMAX or WiFi applications.

The architecture block diagram is shown in Figure 6. APP messages are stored in

APP memory and passed via PN to process engine (PE) array. As introduced above, PN module performs a cyclical shift for a group of APP messages (z messages). All the message updating and calculations according to layered algorithm are processed in PE array. Thanks to the independence of rows in one block of QC-LDPC code, *z* rows (one layer) can be processed in *z* PEs concurrently. Check node message are stored in Check Message Memory. Memory address, control signals and PCM information are generated or stored in Controller.

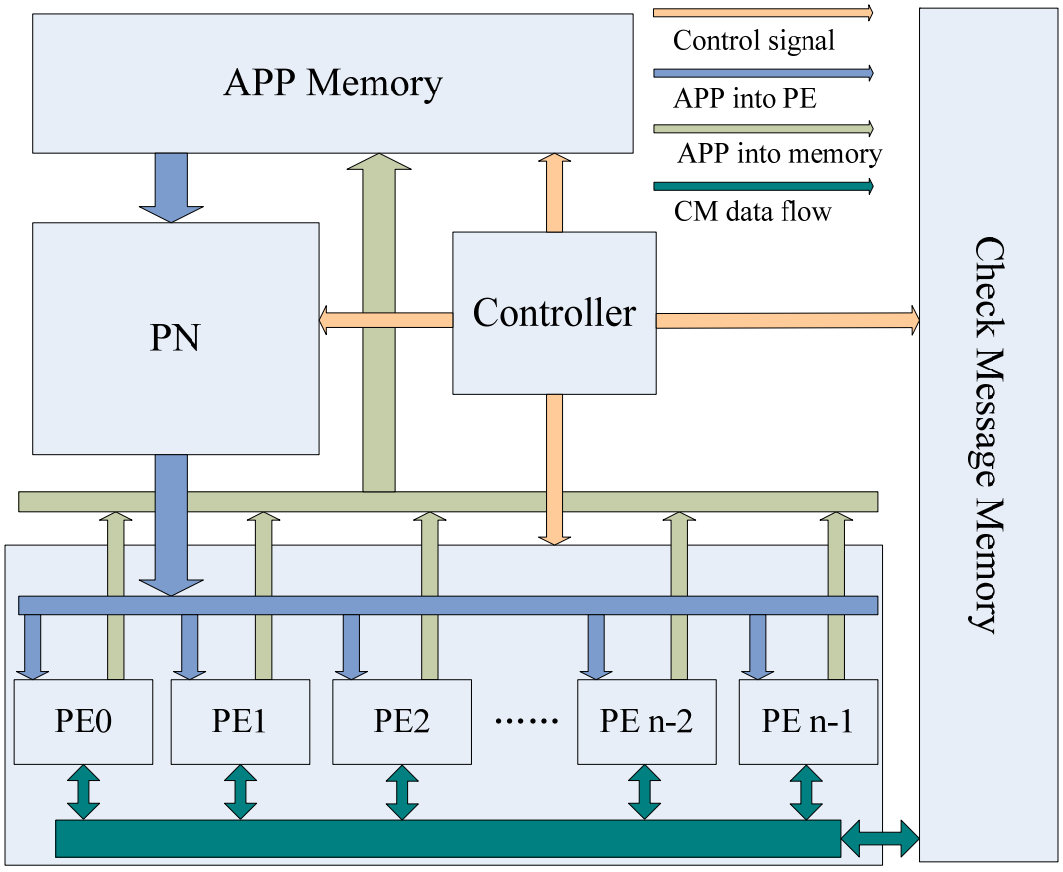


Figure 6 Architecture of partially parallel decoder

3. An Energy-Efficient PN for WiMAX and WiFi

3.1. Introduction

Structured LDPC codes such as quasi-cyclical LDPC (QC-LDPC) codes have received significant attention since their friendliness to hardware implementation features and acceptable bit error ratio (BER) performance compared to the original random codes. Recently, QC-LDPC codes were included in WiMAX (IEEE 802.16e) [5] and WiFi (IEEE 802.11n) [4] which support multiple code rates and code lengths.

For WiMAX and WiFi LDPC codes, Brack [48] proposed a partially parallel decoder architecture which can be reconfigured for variable parity check matrix (PCM) on the fly. To support the multiple code lengths, a reconfigurable permutation network (PN) is needed to cyclically shift data of variable group sizes according to the QC-LDPC PCMs and code lengths. In this chapter, I propose a technique on PN in partially parallel decoder for WiMAX and WiFi LDPC codes. PN using the proposed algorithm is the first PN which can achieve four-way concurrent data permutation. The hardware overhead is negligible which is proved by the synthesis level implementation.

In the following of 3.1, introduction on permutation network with previously published results are depicted followed by a bottleneck of previous partially parallel decoder. In section 3.2, I present my proposed 4-way-supported PN. Integration to partially parallel decoder using proposed PN is given in section 3.3. Finally, summery is given for this chapter in section 3.4.

3.1.1. Permutation Network for WiMAX and WiFi Decoder

Permutation network is needed to cyclically shift data of variable group sizes according to the QC-LDPC PCMs which is shown in Figure 7.

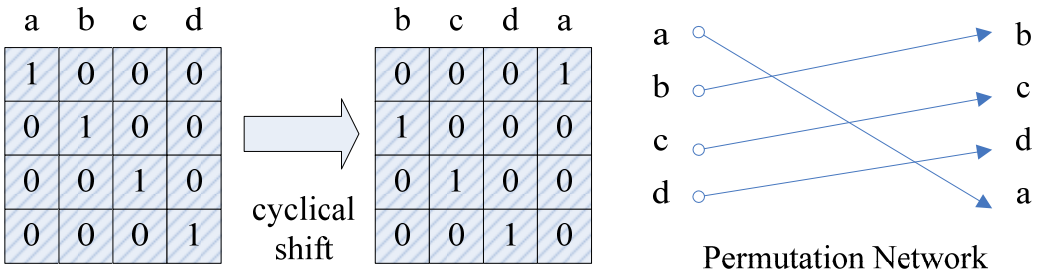


Figure 7 Functionality of PN

In IEEE 802.11n and 802.16e standards, multiple code lengths are achieved by applying different PCM base matrix and expansion factor z which equals to the size of the sub-matrix. Table 1 lists the parameters, expansion factor *z* of QC-LDPC codes, of the codes included in these two standards.

Table 1 Code Parameters of IEEE 802.11n and 802.16e Standards

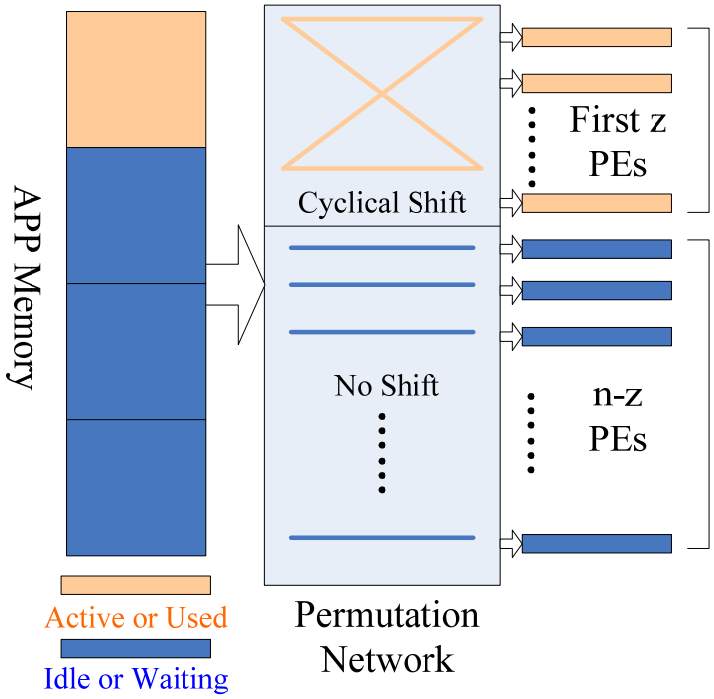
	Code Rate	Expansion Factor z	Block Length N
IEEE 802.11n	4 kinds	27,54,81	648,1296,1944
IEEE 802.16e	6 kinds	24+4K K=[0,1,2,...,18]	576+96K K=[0,1,2,...,18]

In [31], the first Benes network [32] based permutation network for partially parallel decoder was introduced by Olcer. In [33][34], the architecture is improved by

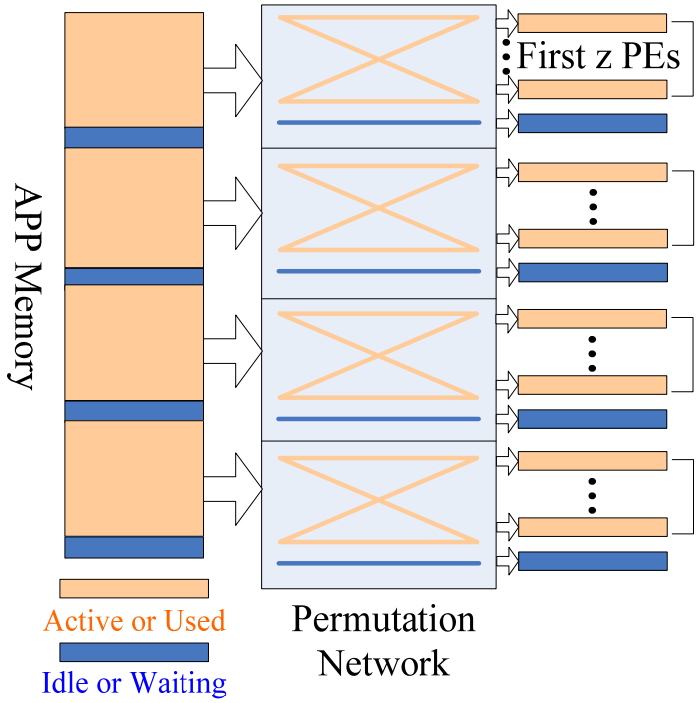
Tang and Gunnam, respectively. The first efficient algorithm of switch control for Benes network based permutation network is given by Oh. [35][36] By his algorithm, the control signals for switches can be generated on the fly according to the input configurations. Compared to the previous work gate count and interconnecting wires are greatly reduced which contributes to an improved decoding energy efficiency. In addition to Benes network based architecture, Liu recently proposed a switch network based on self-routing technique in [37], it is the first architecture that supports a two-way concurrent permutation for QC-LDPC code.

3.1.2. Decoder Inefficiency at Short Code Decoding

To support different code lengths, PN in decoder has to be flexible to variable size of data group which equals to variable z. However, for almost all the early works, only one group of data can be shifted at one time even if the size of the data group (equal to z) is much smaller than the maximum input size. Because of this demerit, the throughput of the decoder will become very low when it is processing a short code. Note that throughput is proportional to the sub-matrix size which equals to the size of the data group fed into the permutation network.



(a). Conventional PN



(b). Proposed PN

Figure 8 Inefficiency of a partially parallel decoder at short codes

As shown in Figure 8(a), if the expansion factor is z, only z APP messages from memory are shifted and only z PEs are actively working at that time resulting low throughput and low hardware usage. For example, assume there are n PEs, if the code length is 576 (z = 24), the throughput will be almost one-fourth of that the code length is 2304(z = 96) and the PE usage will be only 25% (n= 96). By concurrently decoding 3 more short-length code frame which are stored in the rest of the APP memory, I can solve both throughput and hardware utilization problems. One way to realize this idea is to add 3 more PNs for additional 3 independent code frames. However, since one PN occupies almost 10%-15% area or gate count of the whole decoder, it is not reasonable or practical for us to place 4 PNs in one decoder. My solution is to improve the parallelism of PN, making it possible to cyclically shift several groups of data (up to 4) from memory to PEs at one time, shown in Figure 8(b).

3.2. A Four-Way-Supporting Permutation Network

As emphasized before, the permutation network plays an important role in partially parallel decoder. Therefore, how to design PN becomes a hot topic in research area. Previous works have been done in [31] and recently some Benes network (BN) [32] based PNs have been reported in [33][34]. Except one PN in [37] all of the early works can only cyclically shift one group of data at one time even if the size of the data group is much smaller than the input size of the PN. This demerit leads to low throughput and low hardware utilization for LDPC decoder when the code length is small as shown in 3.1.2. In the following of this chapter, based on BN and the control

algorithm proposed in [35][36], I propose a four-way-supported PN. Several groups (up to four) of data can be cyclically shifted concurrently by my proposed PN when the size of the data group is no bigger than half or quarter of the maximum input size.

3.2.1. Benes Network

Benes network (BN) is constructed by cross-bar switches, the state of which is controlled by a one-bit signal shown in Figure 9. The output signals of each switch can be exactly the same as input ones or the ones with exchanged orders. BN as a switch network is capable of performing arbitrary data permutation by appropriately controlling the cross-bar switches in it. A permutation example of an 8×8 BN is shown in Figure 10.

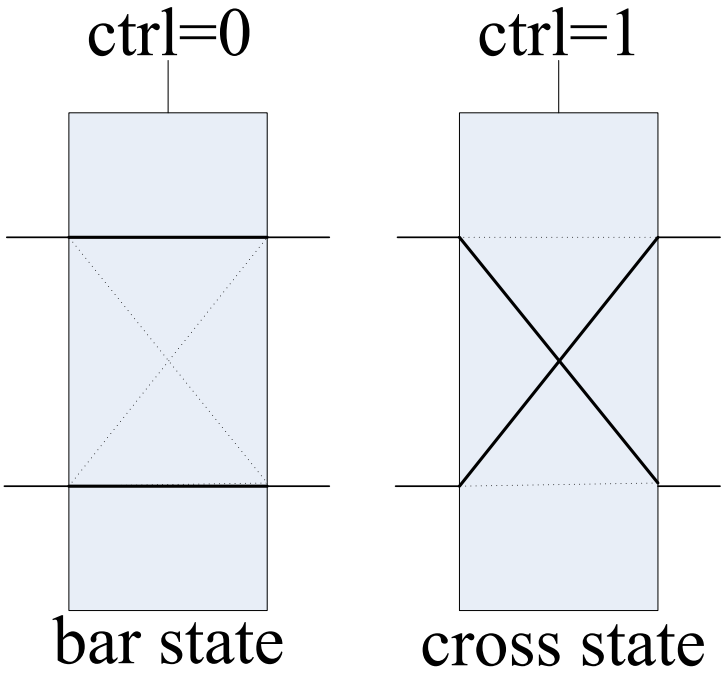


Figure 9 Cross-bar switch, left is a switch with bar state and right cross state

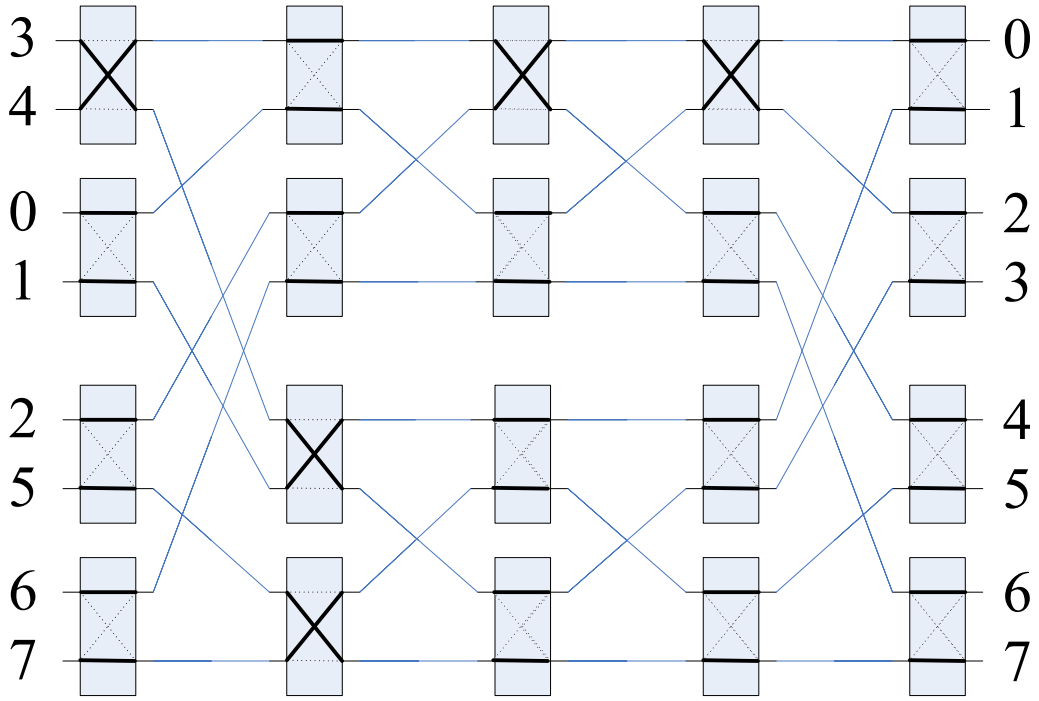


Figure 10 An 8x8 Benes network set for a cyclical permutation

Carefully watch the 8×8 BN in Figure 10, it can be found that an $N \times N$ BN can be constructed by two stages (input and output) and two $N/2 \times N/2$ BNs as shown in Figure 11. By exploiting this recursion property, a PN based on BN and an implementable cross-bar switch control algorithm were given in [35][36] that performs cyclical shift for one data group of variable sizes.

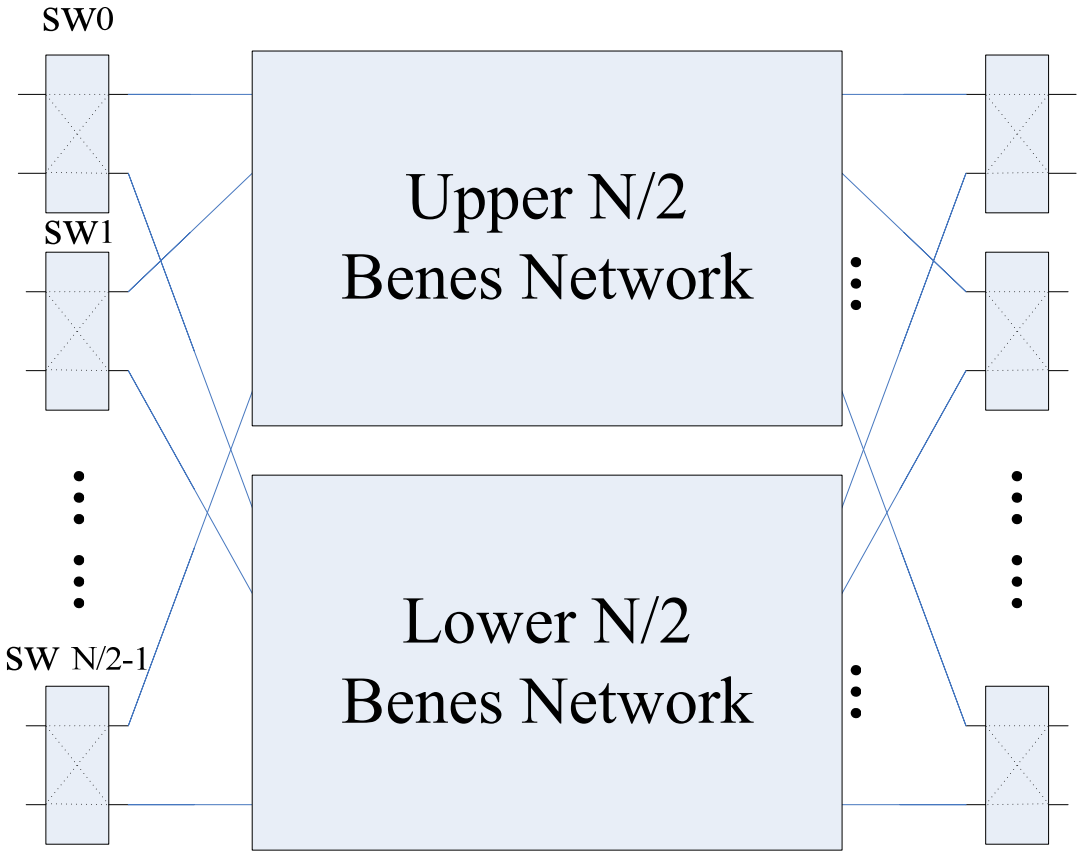


Figure 11 Recursion property of Benes network

3.2.2. Proposed Switch Control Algorithm

3.2.2.1.Symmetry Property of BN

Besides the recursion property mentioned above, without loss of generality, a symmetry property of BN can be explained using a 16×16 BN shown in Figure 12-Figure 14.

At first, divide the input and output signals into two groups (upper part and lower part), except for the middle stage. All the cross-bar switches in the original network, called Network A, shown in Figure 12 can be categorized into two groups according to connections with upper and lower input and output signals, in blue and orange.

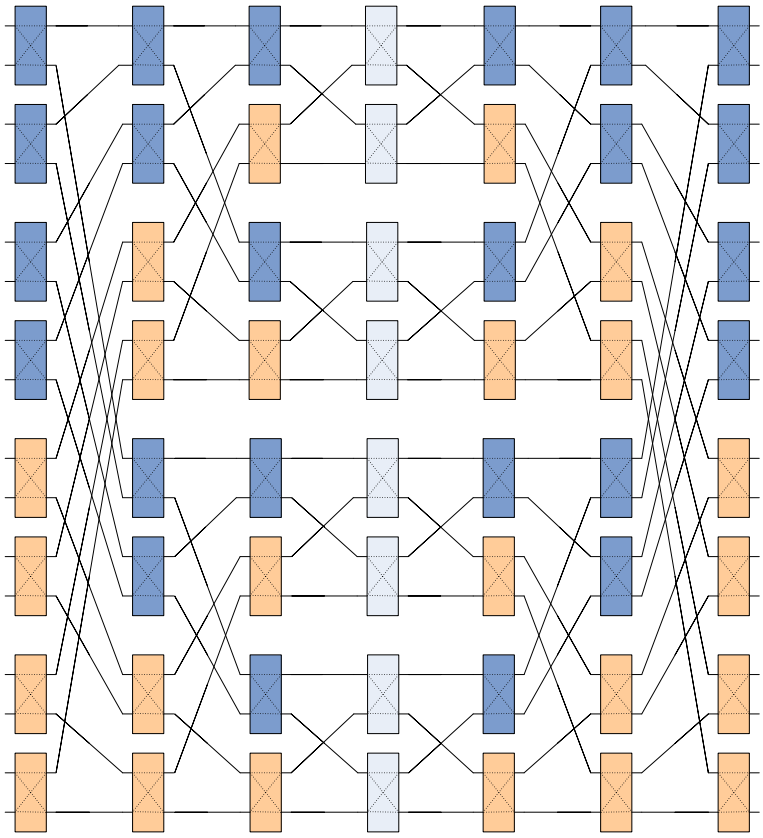


Figure 12 Original 16x16 Benes network, Network A

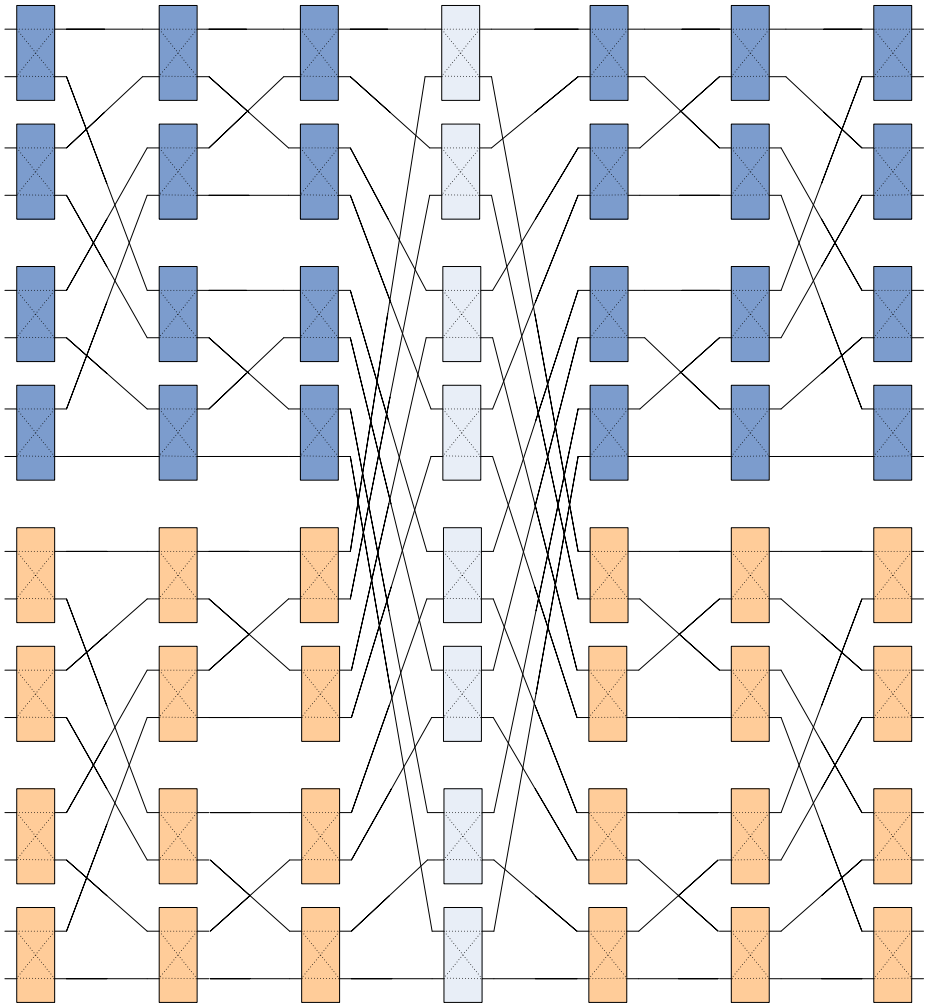


Figure 13 Benes network after switch rearrangement, Network B

After switch rearrangement, moving blue ones to upper and orange ones to lower part we get Network B, shown in Figure 13. Two independent networks referred as Network C, shown in Figure 14 can be obtained by setting the switches of the middle stage in Network B to BAR. Moreover, if the two middle stages of Network C are combined together, Network C becomes an 8×8 BN. This is called symmetry property of BN.

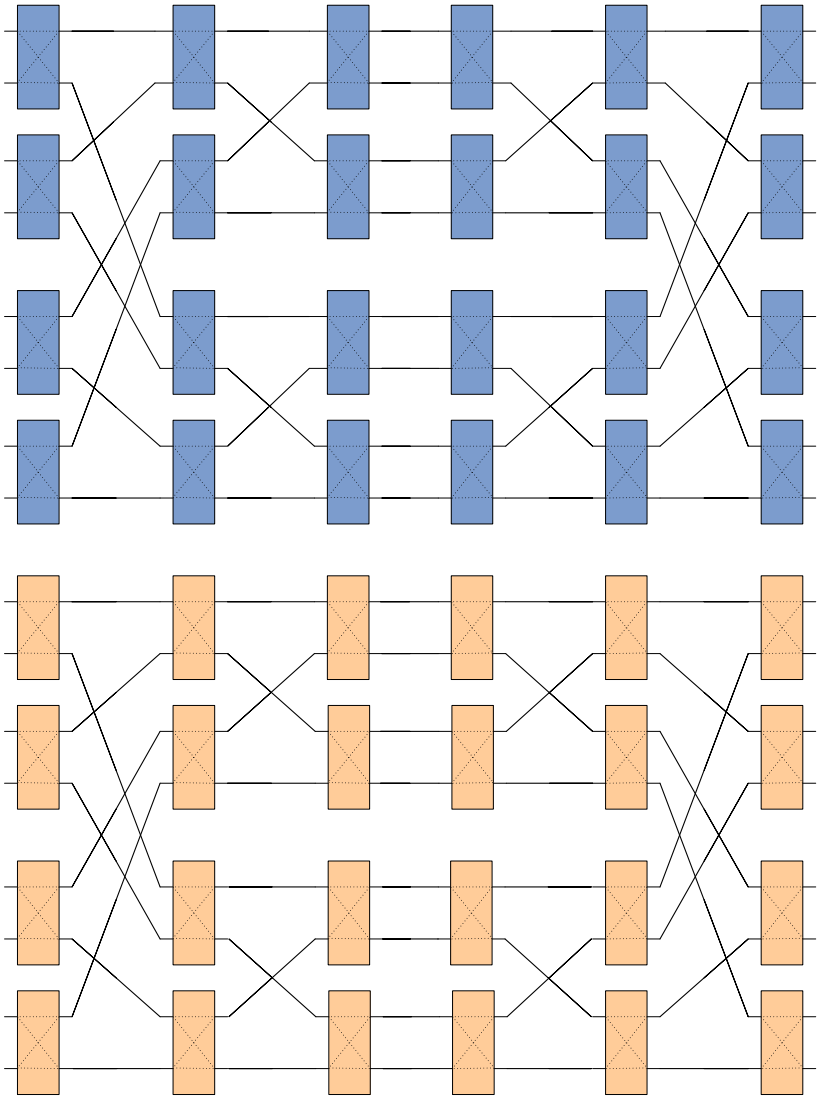


Figure 14 Two independent networks, Network C

Therefore, two same permutations can be performed on two independent groups of input data by setting the same control signals for cross-bar switches at symmetric positions and setting all the switches in the middle stage to BAR. This property can be deduced to 4, 8,. . . , *N*/2 groups.

3.2.2.2. Control Algorithm of Proposed PN and Architecture

Now I propose a four-way-supported permutation network. The architecture is shown in Figure 15. Generally, the PN consists of BN and cross-bar switch control signal generator. Control Signal Generator K (CSG-K) generates control signals for K switches in the input and output stages of BN with K inputs (BN-K). Each BN-K is constructed by two BN-K/2s with CSG-K/2. And each BN-2 is a single cross bar switch.

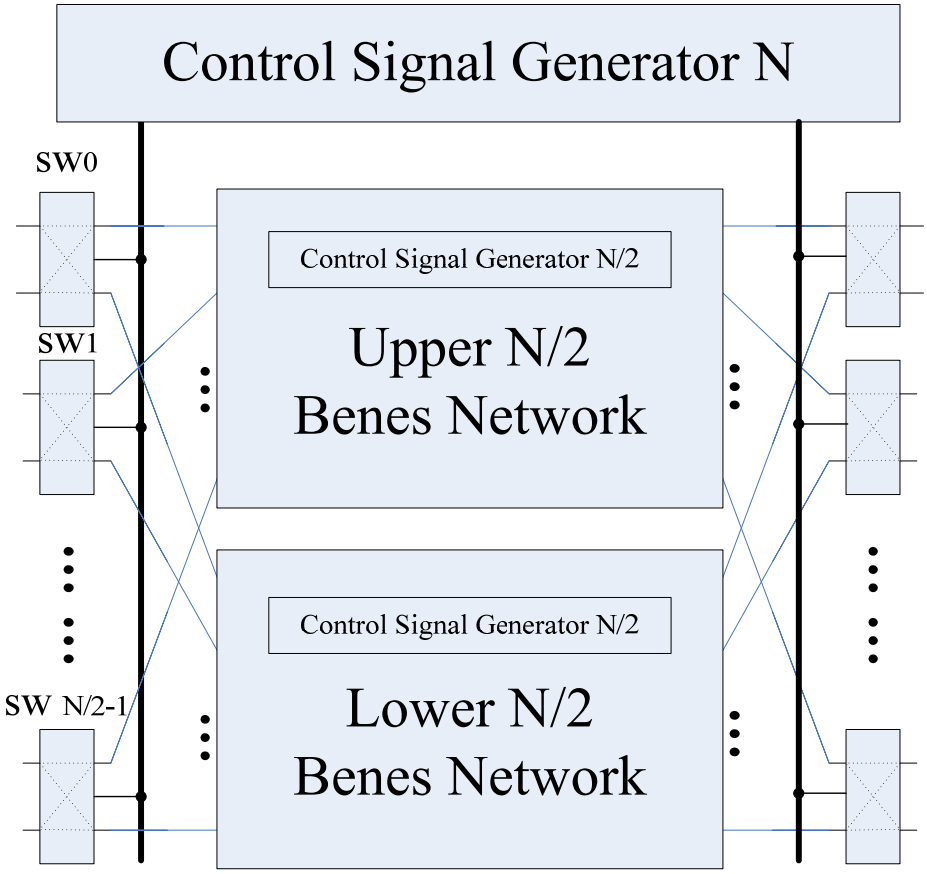


Figure 15 Architecture diagram of proposed four-way-supporting PN

Switch control signal generating algorithm is presented in Figure 16. Parameter *z*, *s*, *N* and *m* denote the size of data group, number of shift positions, size of

network input and number of data groups respectively. Function CS(z, s, N, m) means cyclically shifting m size-z groups of data by s positions through an input-size-N BN. Expression "$\lfloor a \rfloor$" denotes the biggest integer which is not larger than number a. The basic idea of this algorithm is to divide a big shift task to two small ones and combine their results together. For example if we wants to cyclically shift one size-2K data group by 2s positions, we divide it to two size-K data groups and shift them by s positions respectively. The final result can be obtained by combining two shifted size-K data groups together. Based on the symmetric property of BN, two or four groups of data can be shifted concurrently by appropriately copying the control signals of cross bar switches that is listed in the algorithm.

A permutation examples is given in Figure 17 and where N=16, z=3, s=1 and m=4. At first, CS (3,1,4,16) is called for cyclically permuting 4 size-3 data group by one position shift using a 16×16 BN. The first recursive call results in one CS(1,0,8,4) for upper 8×8 BN and one CS (2,1,8,4) for lower 8×8 BN, respectively. Note that number 1 in parameter list (1, 0, 8, 4) is calculated by [3/2]=1 and the second parameter 0=[1/2] where [a] denotes the biggest integer smaller than number a. For the second function, CS(2,1,8,4)=CS([(3+1)/2],[(1+1)/2],16/2,4). Then another four functions CS(0,0,4,4), CS(1,0,4,4), CS(1,0,4,4) and CS(1,1,4,4) are called by the former two at the second recursive call on four 4×4 BN.

ALGORITHM 1:

recursively call ***function*** CS(z ,s, N, m) until N=2;

for N=2, set control signal for cross-bar switch directly;

function: CS(z, s, N, m)

begin

Step 0: Compute $\lfloor (z-s)/2 \rfloor$, $\lfloor z/2 \rfloor$, $\lfloor (z+1)/2 \rfloor$, $\lfloor s/2 \rfloor$, and $\lfloor (z+1)/2 \rfloor$

Step 1: Set the switches in the first stage

if z==even

set all the switches to **BAR**

else

set upper $\lfloor (z-s)/2 \rfloor$ switches to **BAR**

if m==1

set others to **CROSS**

else if m==2

set others in the first half to **CROSS**

copy signals in first half to second half

else if m==4

set others in first quarter to **CROSS**

copy signals in first to 2nd , 3rd and 4th quarters

Step 2: Call ***function*** CS($\lfloor z/2 \rfloor$,$\lfloor s/2 \rfloor$, N/2, m) and CS($\lfloor (z+1)/2 \rfloor$, $\lfloor (s+1)/2 \rfloor$, N/2, m)

Step 3: Set the switches in the last stage

if z==even && s==even

set all the switches to **BAR**

else if z==odd && s==odd

set all the switches to **CROSS**

else if z==even && s==odd

set upper $\lfloor z/2 \rfloor$switches to **CROSS**

if m==1

set others to **BAR**

else if m==2

set others in the first half to **BAR**

copy first half to second half

else if m==4

set others in the first quarter to **BAR**

copy first to 2nd , 3rd and 4th quarters

else if z==odd && s==even

set upper $\lfloor z/2 \rfloor$switches to **BAR**

if m==1

set others to **CROSS**

else if m==2

set others in the first half to **CROSS**

copy first half to second half

else if m==4

set others in the first quarter to **CROSS**

copy first to 2nd , 3rd and 4th quarters

end

Figure 16 Proposed switch control signal generating algorithm

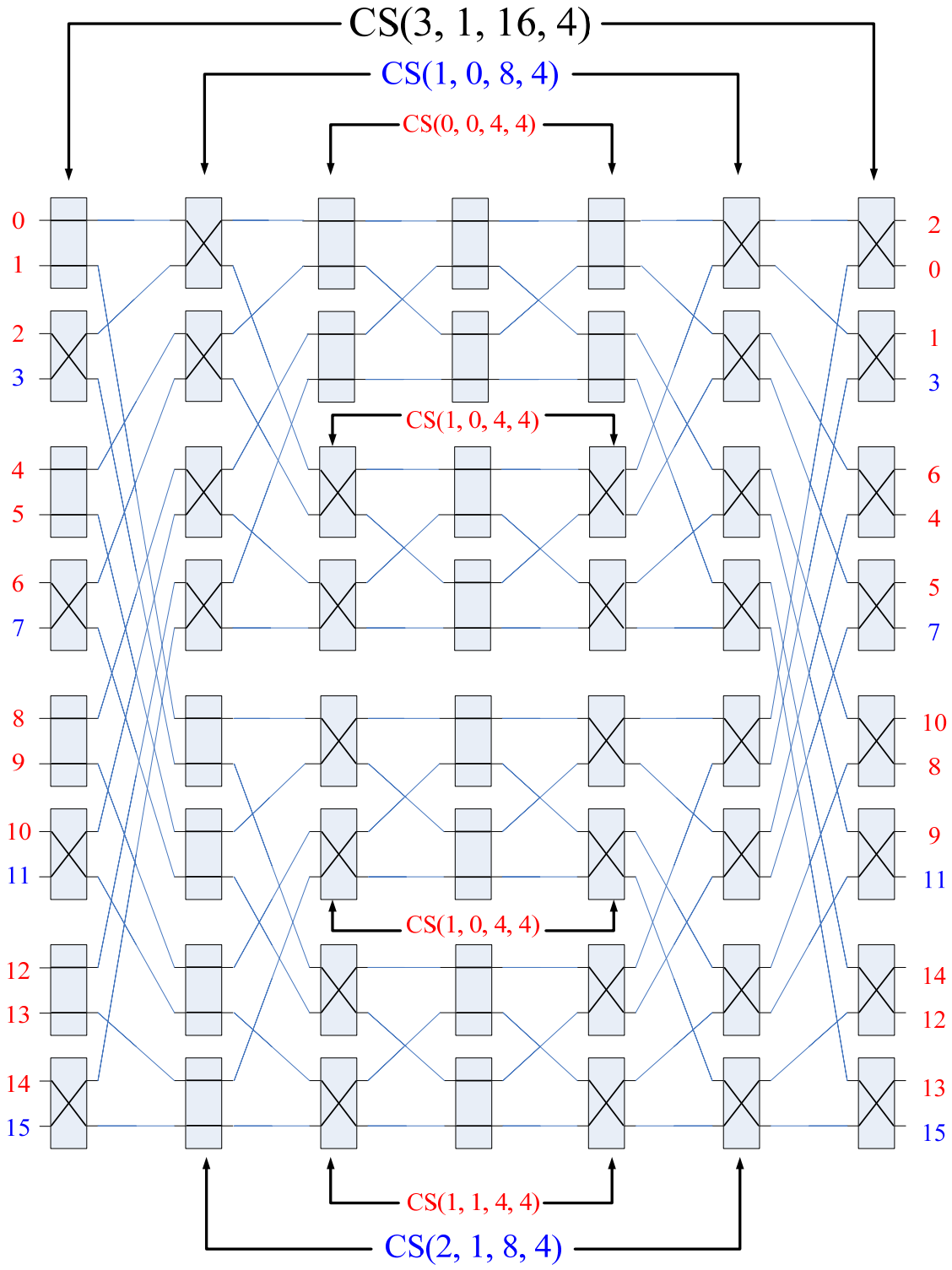


Figure 17 An example of cyclical permutation

3.2.2.3. Hardware Implementation and Comparisons

To evaluate the hardware implementation overhead of my proposed four-way-supported permutation network, it is analyzed by the following three terms.

(1) Benes network without control signal generator (Overhead A);

(2) Control signal generator for one data group permutation (Overhead B);

(3) Extra hardware overhead on control signal generation for multi data groups permutation. (Overhead C)

For overhead A, the Benes network is composed of cross-bar switches each of which can be constructed by two 2-1 multiplexers. For 128×128 Benes network used in my design, there are 13x64=832 cross-bar switches equivalent to 1,664 2-1 multiplexers. Considering the data width 9, totally, we need 1,664x9=14,976 two-one one-bit multiplexers to construct the Benes network. From the synthesis result, it occupies about 40.4k gate count.

For overhead B, control signal generation algorithm for one data group, hardware overhead for computation of [(z-s)/2], [z/2], [(z+1)/1], [s/2] and [(s+1)/2] is very small and can be ignored. Implementation of "set top k switches to BAR (CROSS)" is the dominant part. For an input or output stage of an $N \times N$ BN, a $\log_2(N)$ to N/2 unique decoder is needed, e.g. a 7-64 decoder is needed for the input stage of a 128×128 BN. If the decoder input is a binary number k, then first k bits of the output are set to zeros (BAR) and remaining N/2-2 to ones (CROSS), as shown in Figure 18. The gate counts of these decoders needed in the PN are listed in the Table 2.

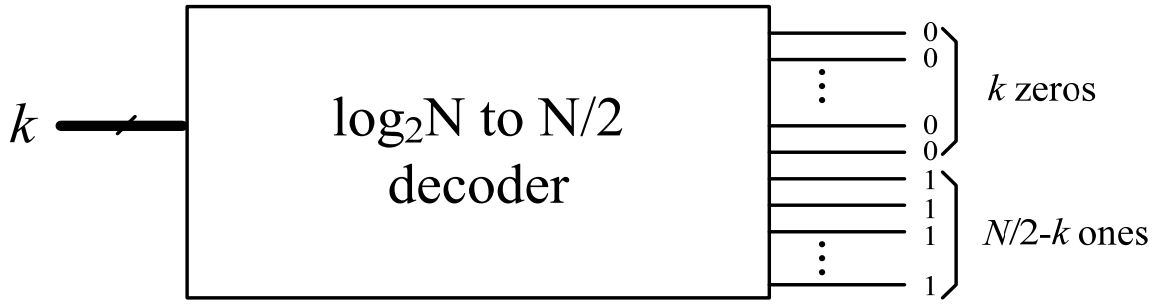


Figure 18 Block diagram of a unique decoder

Table 2 Hardware overhead of unique decoders

Decoder (Input- output)	Single gate count	Copies needed	Sub total
7-64	135	2	270
6-32	69.2	4	276.8
5-16	35.3	8	282.4
4-8	20	16	320
3-4	11.4	32	364.8
2-2	8.5	64	544
Total			2,058

Therefore, it takes about 2.5k gate count to implement the control signal generator for one data group permutation.

For overhead C, the extra hardware cost for multi-data group permutation is on implementing signal copying in the algorithm, like “copy signals in the first half to second half”. In my design, multiplexers are used to select or copy signals for switches.

For each input or output stage of a Benes network, switches in the top 1/4 part do not need multiplexer to select control signals. For the second and third 1/4 parts, a 2-1 one bit multiplexer is needed for each switch. For the last 1/4 part, a 4-1 one-bit multiplexer is needed for each switch to select three possible signals from the decoder. An example of control signal generator for the input stage of a 16×16 BN is given in Figure 19. For a 128×128 PN it needs 832/2=416 two-one and 832/4=208 four-one multiplexers to achieve multi-group data permutation. Approximately, this part consumes 2k gate count.

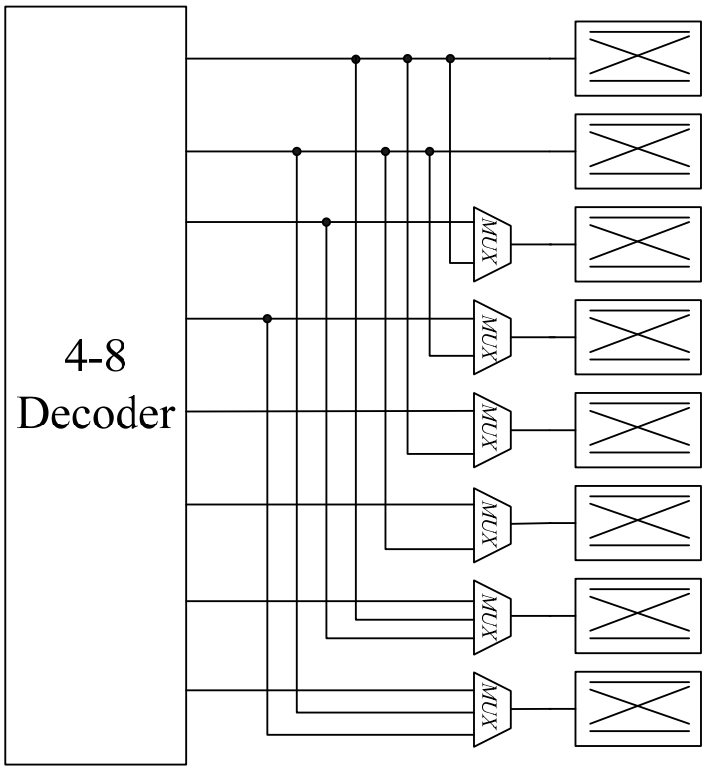


Figure 19 Extra overhead of multi-way permutation for a 16x16 BN

From the above analysis, total hardware cost for control signal generating is around 4.5k-5k which is about 10% over the entire PN. For the speed overhead, synthesis result shows the smallest delay of a 128×128 BN (without control) at 90nm

node is 2ns and the control logic will introduce an extra 1.2ns signal delay, resulting in a total delay 3.2ns of PN. (311Mhz)

To verify the hardware complexity of proposed PN, it is synthesized using 90nm SMIC library by Synopsys Design Complier. The synthesis result and comparisons to previous work are given in Table 3. My proposed permutation network achieves highest parallelism and supports multiple standards with good area and frequency results.

Table 3 Comparisons of Implementation Results of PNs

	Proposed	**PN in [48]**	**PN in [33]**	**PN in [36]**	**PN in [37]**		**PN in [59]**	
Data Width	9bits	6bits	8bits	6bits	6bits		6bits	
CMOS Process	90nm	130nm	180nm	180nm	180nm		90nm	
Area (mm^2)	-	0.511	2.171	0.722	-		-	
Gate Count	47.4k	-	-	-	23.1k	37.4k	17.1	18.3
MAX Freq. (MHz)	311	333	85	94	384		600	
Network Size	128x128	128x128	128x128	96x96	96x96		96x96	
Parallelism (z=64)	Two	One	One	One	One	One	One	One
Parallelism (z=32)	Four	One	One	One	One	Two	One	Two

3.3. Integration into Partially Parallel Decoder

In this section, an integration of my proposed PN with partially parallel decoder is given. The main discussion is on the interface between PN and APP message memory.

As shown in Figure 6 PN is used to cyclically shift APP messages from APP memory to appropriate PEs according to the PCMs which are stored in the ROM of the controller. The word width of the PN equals to APP message's width which is 8 bits. Data input size of PN is set as 128×128 to support the biggest code length (2,304 bit) in IEEE 802.16e where the cyclically shifting data group size is 96. APP memory is 1024-bit-wide (8×128 blocks) and 24-deep. The decoder works in 3 different modes where one, two or four independent blocks of codes can be decoded simultaneously according to the code lengths. The APP memory organizations in different modes are given in Figure 20.

When the code length is 1,296 (one 802.11n code) shown in Figure 20, the decoder processes two independent blocks of codes (code 0 and code 1) whose APP messages are stored in 108 blocks of memory divided into two groups. The APP messages of each code are updated by 54 PEs in parallel so the PE usage is about 84% (54×2/128).

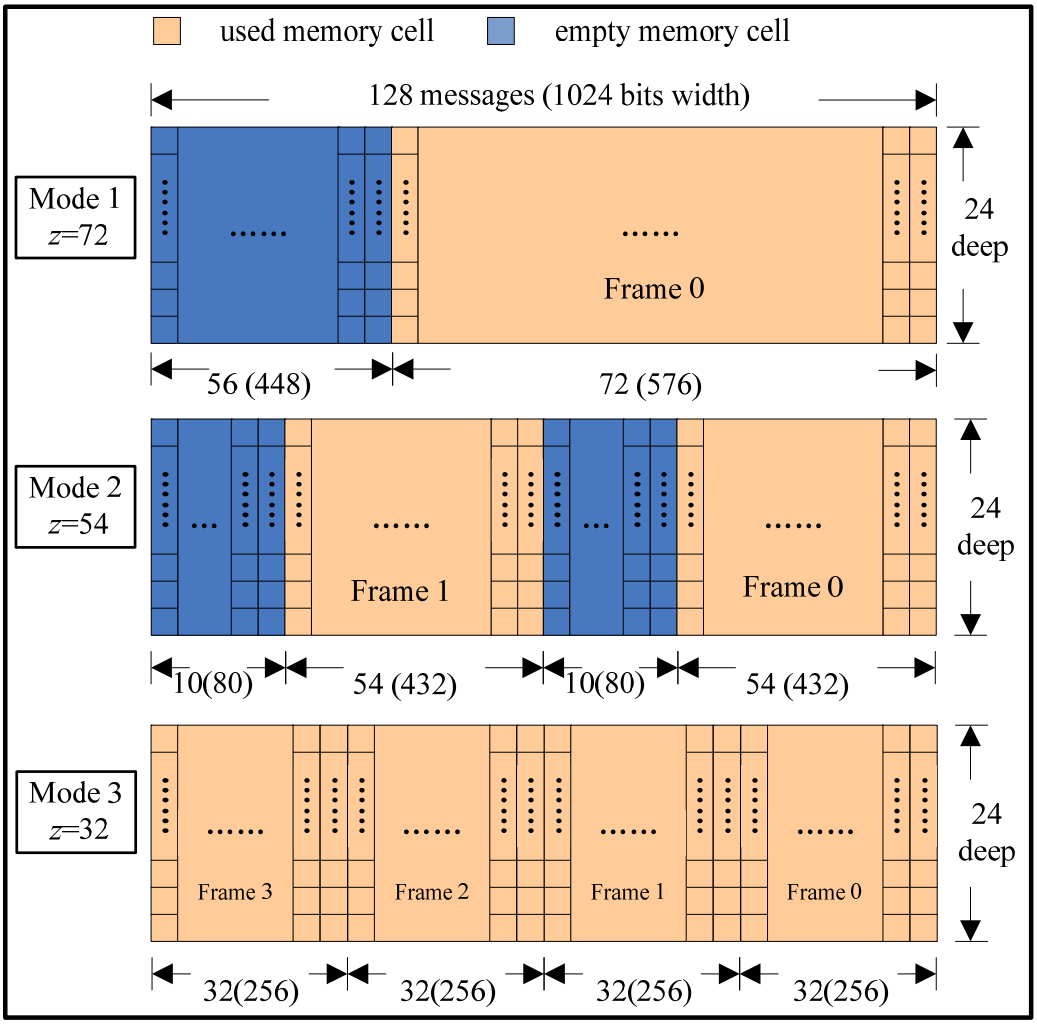


Figure 20 Memory interfacing with proposed PN in partially parallel decoder

3.4. Summery

In this chapter, I have proposed a Benes network based four-way-supported permutation network for LDPC decoder applied for WiMAX and WiFi. The proposed PN achieves up to 4 times parallelism compared to the early works while maintaining small area and high frequency. Meanwhile, an integration of proposed PN into a partially parallel decoder is presented. For improvement on energy efficiency contributed by my proposed technique, it will be shown in chapter 4 by a synthesis level partially parallel decoder.

4. A Lossless ET Algorithm for WiMAX and WiFi

4.1. Introduction

Conventionally, in an iterative LDPC decoder the decoding process is stopped if a pre-defined maximum iteration number is achieved. To compensate its low adaptability to variable communication channel conditions and speed up the decoding, early termination (ET) algorithm criterion is invented. During an iterative decoding process, if a defined stopping criterion (SC) is met, the decoding is stopped saving the unnecessary decoding iterations, which contributes to an energy efficiency improvement.

The most straightforward SC is a satisfaction of all parity-check equations (SPE) [38] defined by the rows of a PCM of a certain LDPC code, shown in Figure 21. SPE based SC can be met if and only if all the bits in a code word are correct, which means that it will not introduce any bit error rate (BER) performance loss.

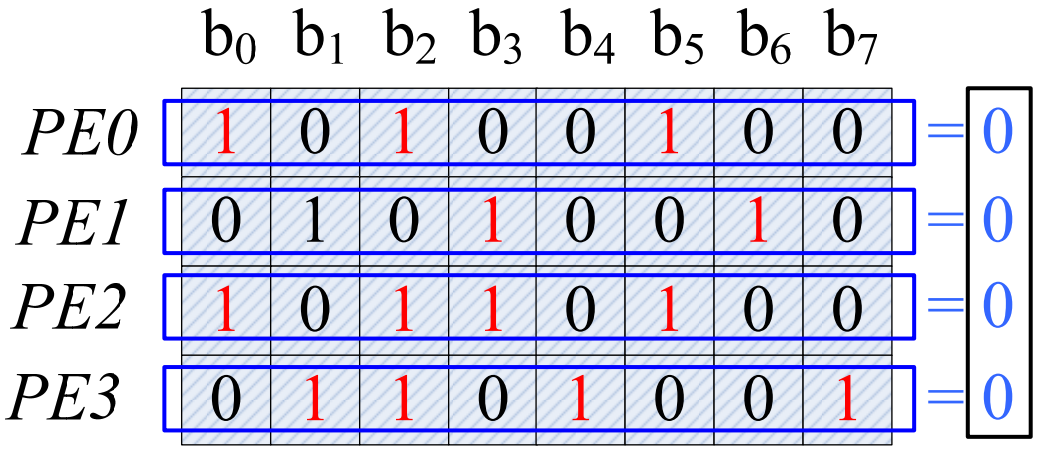


Figure 21 SPE based early termination

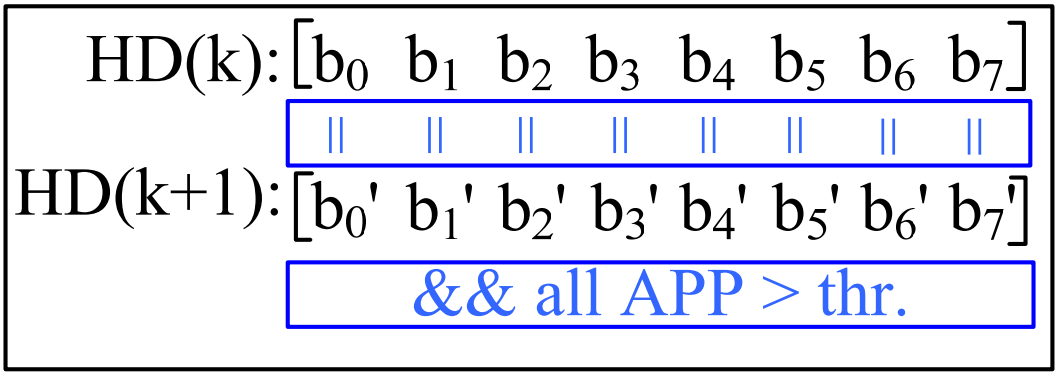


Figure 22 HDA based early termination

Another popularly used SC is called hard decision aided (HDA) SC which was originally invented for Turbo code decoding [40] and applied in LDPC code recently. For HDA SC, if the hard decisions made in two successive decoding iterations do not change, the decoding will be stopped as shown in Figure 22. However, HDA SC does introduce some BER performance loss since no changes of hard decisions made in two successive decoding iterations does not guarantee a correct code word. To overcome this demerit, an additional threshold for magnitudes of a posterior probability (APP) is added to make more restrictions, leading a BER performance improvement. [39]

In this chapter I first demonstrate an observation on decoding convergence of WiMAX and WiFi LDPC codes in 4.2. Based on the observation, a lossless ET algorithm based on syndrome accumulation for WiMAX and WiFi LDPC codes is proposed in 4.3. Software simulation results and comparisons to the previous works are given for the proposed ET algorithm. Moreover, in 4.4, an energy efficient partially parallel LDPC decoder which applies both the techniques which are proposed in chapter 3 and this chapter is presented with a synthesis level implementation in the end of this

chapter.

4.2. Observation on Decoding Convergence

In this section, an observation on convergence speed of information bits and redundant bits in a code word of WiMAX and WiFi is shown and discussed. Based on the observation I propose an idea to reduce the unnecessary decoding iterations.

LDPC code can be decoded using iterative message-passing algorithm. With the check and variable messages being passed and updated between check nodes and variable nodes in a decoder, a posterior probabilities (APP) messages, based on which the final hard decisions are made, converge to be more and more reliable. Simulation results revealed that the bigger the weights of columns in PCMs, the faster the convergence of the APP messages corresponding to those columns.

For LDPC PCMs defined in WiMAX and WiFi standards, except z columns, e.g. z columns expanded by the column 12 in base matrix shown in Figure 4 of 2.1.2, weights of all the columns in redundant part are two (red in Figure 4), which is the smallest column weight. This feature leads to an interesting phenomenon that averagely, APP messages of information bits in a code word converge to a reliable value much faster than redundant bits which is shown in Figure 23.

In Figure 23, curves for average magnitude of APP values in log likelihood ratio (LLR) form over iteration numbers observed at different channel conditions are drawn. With the increment in iteration number, the average magnitude of APP messages of information bit increases faster and achieves a bigger value than that of the redundant bits. I explain this phenomenon as follows: The bit node corresponding to the bigger

column weight is connected to more check nodes and has more constrains from check equations. Consequently, it will get more messages and be updated more frequently.

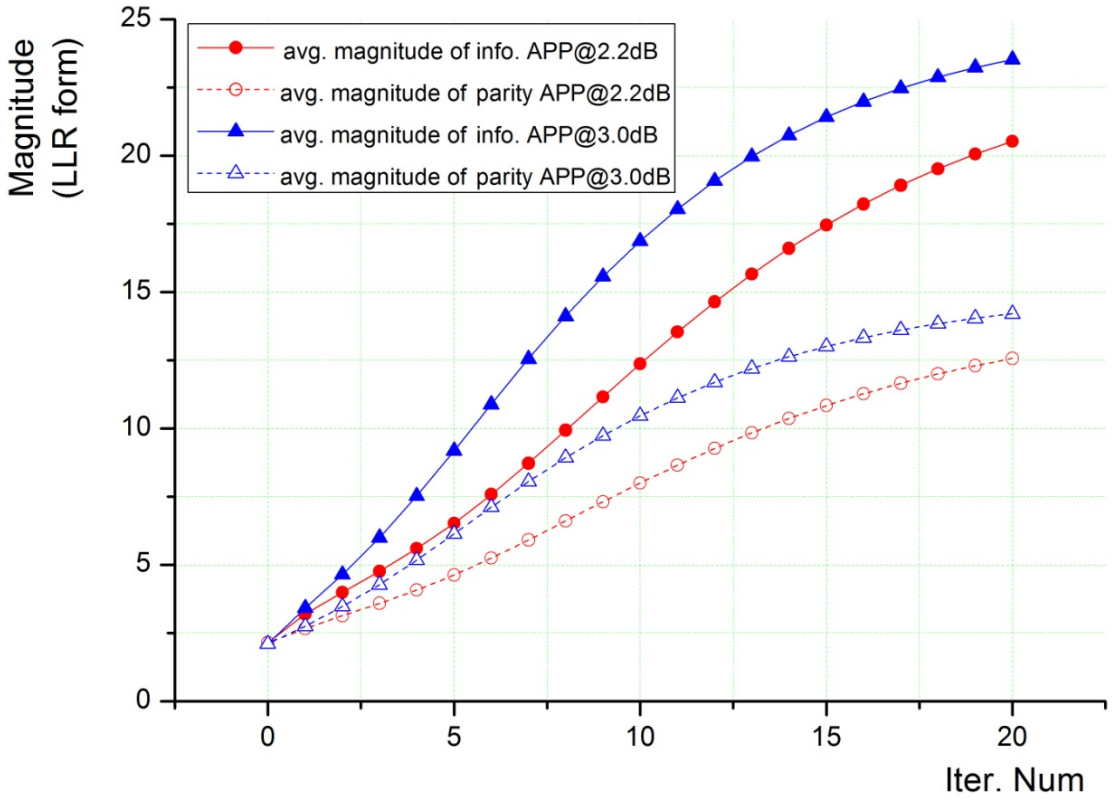


Figure 23 Decoding convergence of WiMAX LDPC code

More importantly, because the information bits converge much faster, errors among them will be corrected earlier during an iterative decoding. Code frames which are incorrectly decoded due to lack of decoding iterations always contain more errors in redundant part than information part. For a demonstration, a simulation is run where a rate-1/2 length-2304 code in WiMAX is applied. Incorrectly decoded code frames are categorized into two types: named Type I and Type II errors. Code frame of Type I contains errors in information part or first z redundant bits while Type II code frame contains errors only in rightmost M-z redundant bits. Proportions of two types of error

codes at different channel conditions are depicted in Figure 24. It can be found that the Type II error happens much more frequently than Type I, which implies that a lot of decoding iterations are performed just for correcting Type II errors which can be ignored. *If a decoder can detect a type II frame and stop the decoding process, decoding iteration can be saved to some extent.*

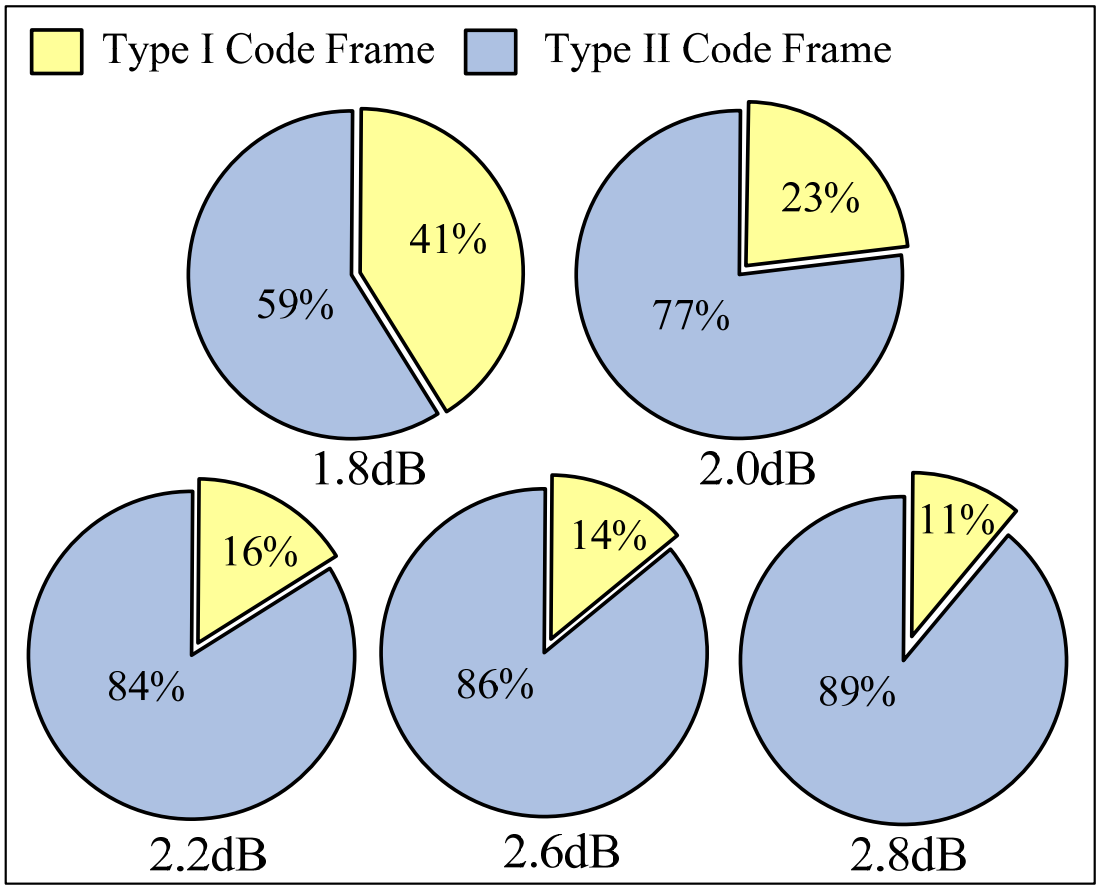


Figure 24 Rates of Type I and II error frames at different channel conditions

4.3. A Syndrome Accumulation based ET Algorithm

In this section, firstly, features of syndrome accumulation of WiMAX and WiFi LDPC codes in decoding are discussed with a mathematic proof. Based on these

features, a lossless ET algorithm is presented. Finally, software based performance simulation and hardware implementation overhead of my proposed ET algorithm are given.

4.3.1. Syndrome Features and Proposed ET Algorithm

The key idea of my proposed early SC is to save the unnecessary decoding iterations for correcting the Type II error. By exploiting the particular structure of PCMs in WiMAX and WiFi standards, a mechanism is devised to find the Type II code frames. Before further discussion, some notations and mathematic definitions are given below. A syndrome vector is defined as s (15) where M is the number of rows of a PCM and s_i (0 or 1) is the result of the parity-check equation corresponding to the ith row, indexed from 0, of PCM. A syndrome accumulation vector (SAV) a (16) is defined as a mapping from s according to (17) where z is an expansion factor of a PCM and c equals to the number of rows in the base matrix.

$$\mathrm{s}=[s_0, s_1, \ldots, s_{M-2}, s_{M-1}]^T \quad \textbf{(15)}$$

$$\mathrm{a}=[a_0, a_1, \ldots, a_{z-2}, a_{z-1}]^T \quad \textbf{(16)}$$

$$a_i = \sum_{k=0}^{c-1} s_{i+kz} \quad \textbf{(17)}$$

According to the standards, for a base matrix with expansion factor z and code rate R, generally, the right most M-z columns of the expanded PCM forms a $M\times(M-z)$ two-diagonal matrix where. A bit in a code word is marked as b_{kz+j}, where k is the index of a column block from 0 to 23 and j is an offset in a column block from 0 to z-1. An example of R=5/6, z=4 is shown in Figure 25.

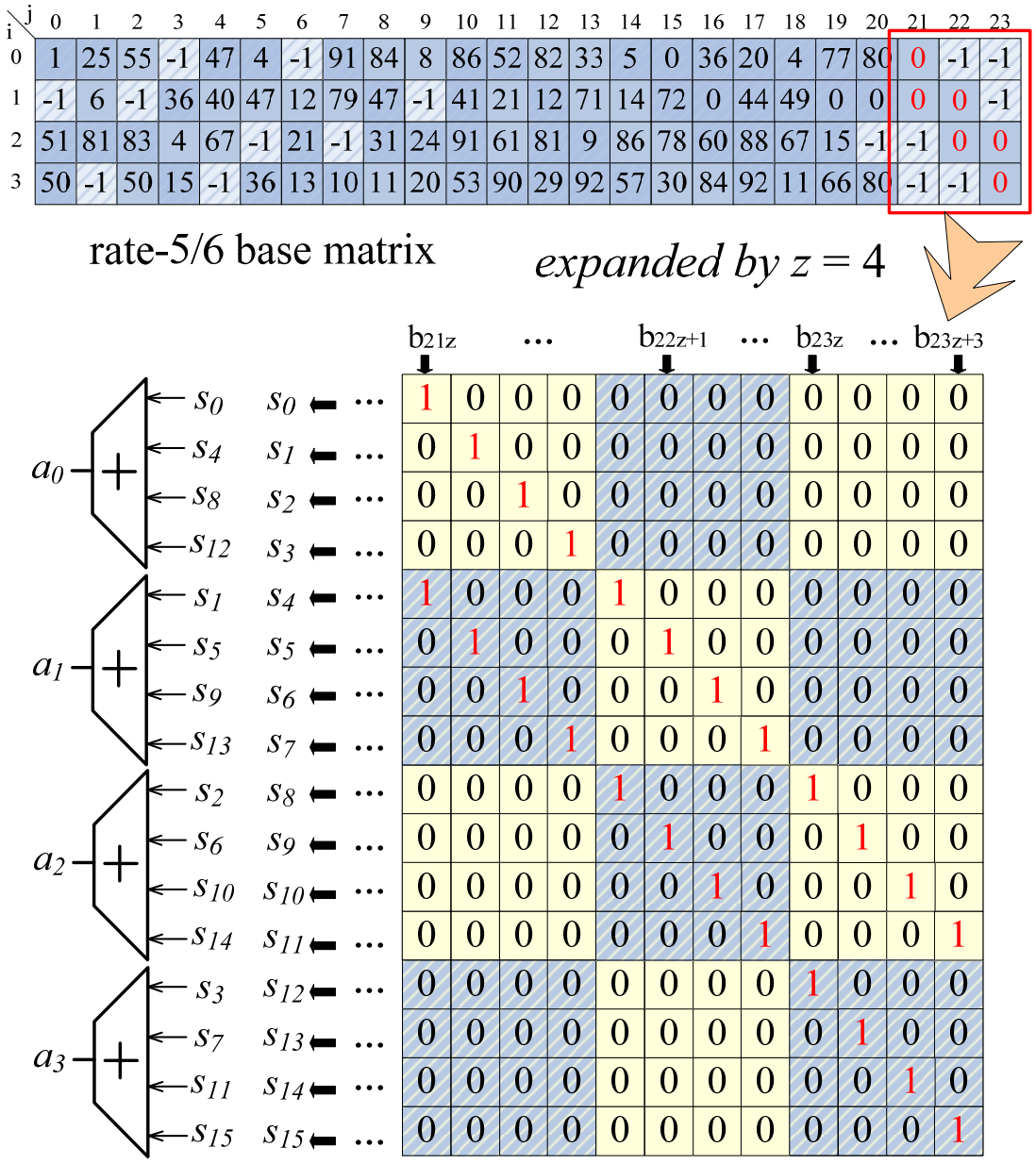


Figure 25 An example of a two-diagonal matrix and accumulation vectors

If a code frame is of Type II with *n* errors in redundant part, the SAV contains only even numbers. This is called SAV property which can be proven by the mathematic induction, shown in below.

Start of the Proof

1) For case n=1, the error bit $b_{k_1 z+j_1}$ makes only two parity-check equations corresponding to row $(k_1\text{-}1\text{-}24R)z+j_1$ and row $(k_1\text{-}24R)z+j_1$ of the PCM equal to ones where $24R<k_1<24$, and $0 \leqslant j_1 < z$. Therefore, the SAV contains z-1 zeros and 1 two for a_{j_1}. Thus, the statement is true for n=1;

2) Assume that for case n=q the statement is true which means if there are q errors, $b_{k_1 z+j_1}$, $b_{k_2 z+j_2}$,…, $b_{k_q z+j_q}$, among the right most M-z bits of the code word, the SAV contains only even numbers;

3) For case $n=q+1$, the $(q+1)$th error bit $b_{k_{q+1} z+j_{q+1}}$ only affects two parity-check equations corresponding to row $(k_{q+1}\text{-}1\text{-}24R)z+j_{q+1}$ and row $(k_{q+1}\text{-}24R)z+j_{q+1}$ of the PCM by either flipping them from zero to one or from one to zeros. Therefore, only $a_{j_{q+1}}$ in SAV of case $n=q$ will be added or subtracted by two or unchanged. Based on the assumption of case $n=q$, for case $n=q+1$ the SAV still contains only even numbers. Thus, the property is true for $n=q+1$;

Above all, we have proven that the statement is true for every possible n.

End of the Proof

With SAV property, a Type II code frame can be easily verified like the example shown in Figure 26. For this example, z=4, R = 5/6, n=4 and the SAV equals to $[0, 4, 0, 2]^T$.

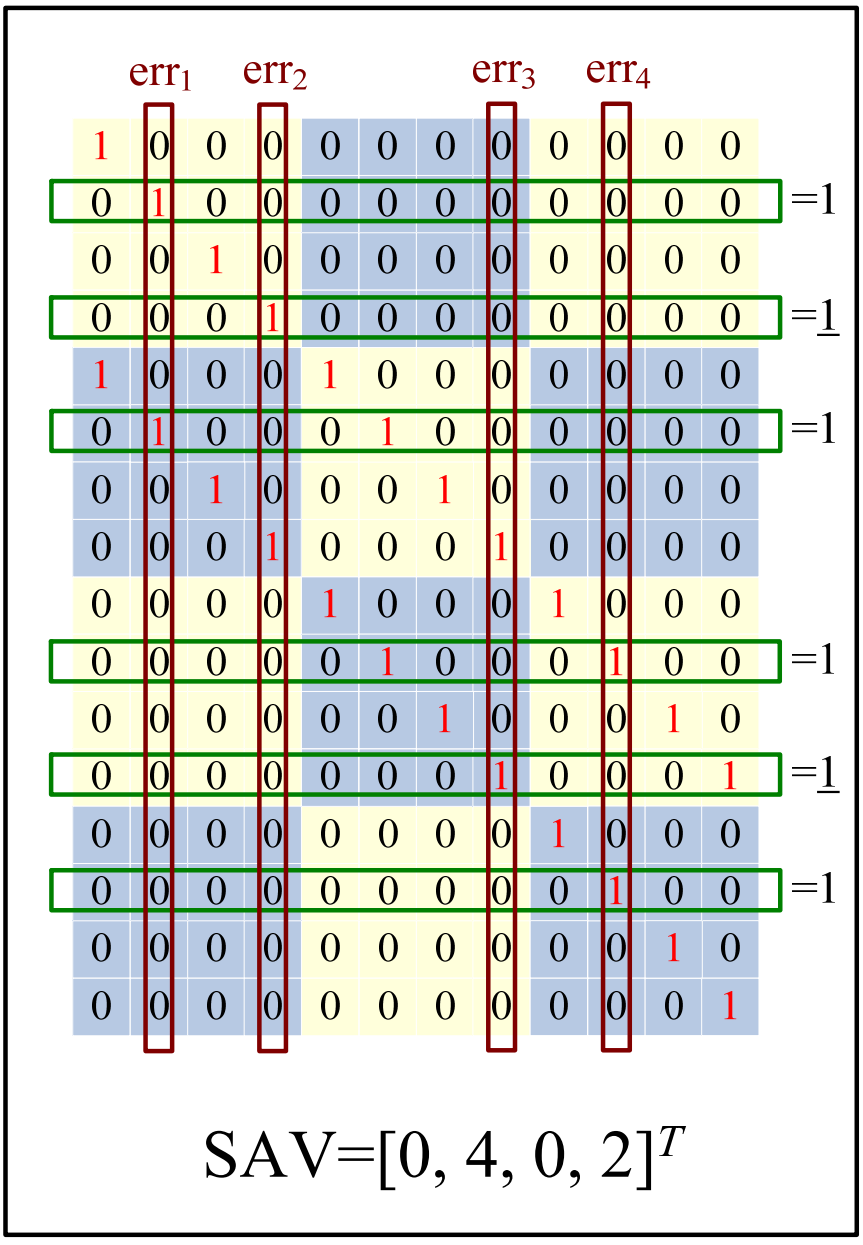


Figure 26 An example showing SAV property

Reversely, if SAV contains only even numbers, the corresponding code frame is of Type II for almost all the circumstances. This is because an error among information or first z redundant bits will at least affect three parity-check equations corresponding to three rows in random positions of a PCM, making the SAV disorderly. Other than strict proof, I run a simulation to test all the codes in WiMAX and WiFi. Simulation result shows the false detection rate for Type II code frame using SAV property is smaller than 10^{-6} for all kinds of codes in WiMAX and WiFi standards. Therefore, it is reasonable and effective for us to claim a Type II code frame if SAV contains only even numbers. Based on the discussion above, now I propose a new early stopping criterion as follows.

For each one of decoding iterations, calculate the syndrome accumulation vector (SAV). If SAV contains only even numbers, stop the decoding.

4.3.2. Performance Evaluation

A PC based software simulation was run to evaluate the performance of my proposed ET algorithm. Two codes, a length-2304 rate-0.5 code in WiMAX and a length-1944 rate-2/3 code in WiFi, are used in the simulation. Floating point number, normalized min-sum algorithm with correction factor 0.75 and flooding schedule are applied. Maximum iteration numbers are set 30 and 50 for WiMAX and WiFi codes, respectively.

BER and frame error rate (FER) curves (solid for BER and dash for FER) are drawn in Figure 27 and Figure 28 for those two codes, respectively. All satisfaction of parity-check equation (SPE) based SC and HDA SCs with different thresholds for

APP messages are compared to my proposed SC. It is shown from the comparison results that, like SPE based SC, my proposed SC does not introduce BER performance loss so as HDA does.

The main purpose of ET algorithm is to reduce the unnecessary decoding iterations which accounts for additional power consumption or throughput decrease. To evaluate this functionality, average iteration number (AIN) was monitored in the simulations which are listed in Table 4.

It is obvious from Table 4 that my proposed SC achieves smallest AIN especially in case of the small code rate when up to 12% of iterations are saved compared to the conventional SPE based SC.

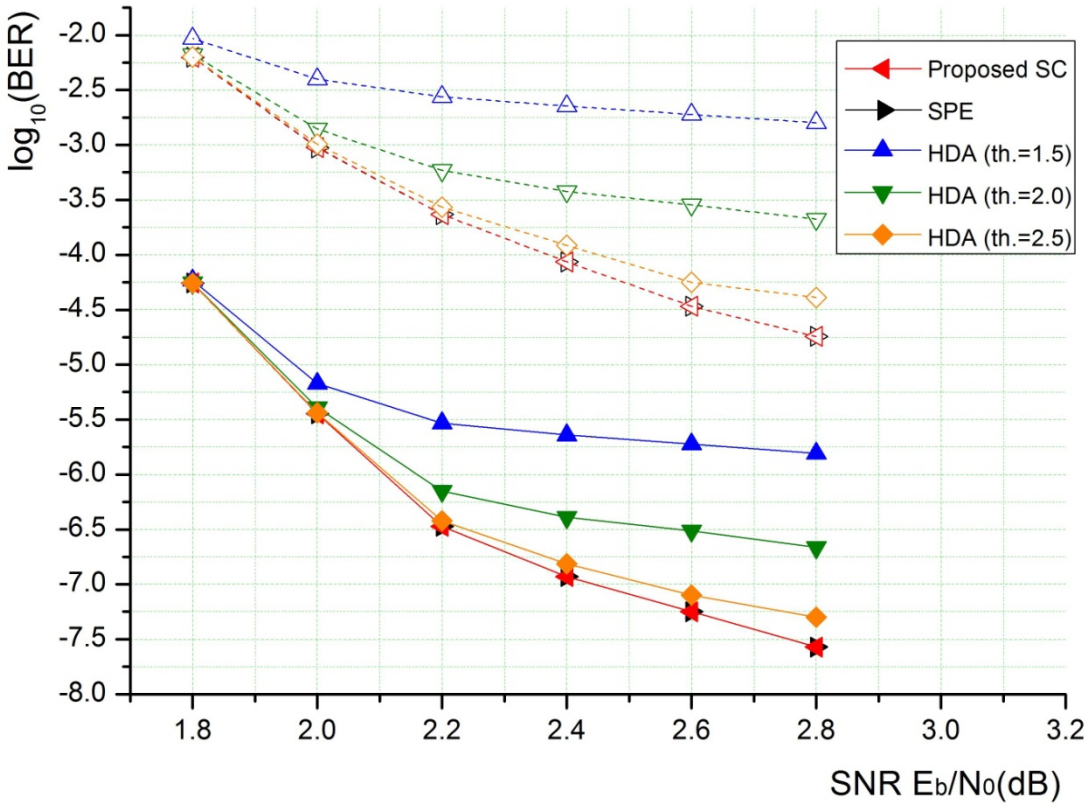


Figure 27 BER and FER performance 1 (WiMAX Code)

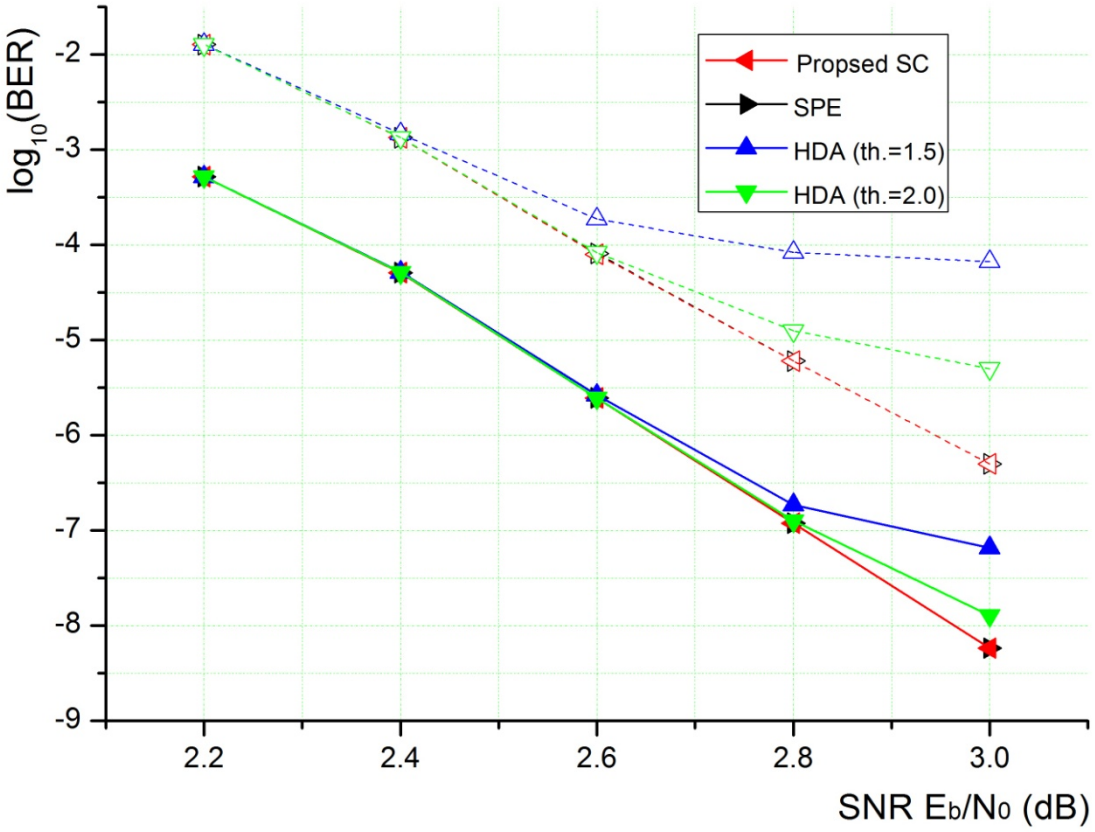


Figure 28 BER and FER performance 2 (WiFi Code)

Table 4 Comparisons of AIN

Code Type	**SNR (dB)**	**Average Iteration Number (AIN)**			**Reduction to SPE**
		Proposed	**SPE**	**HDA(1.5)**	
WiMAX Code[a]	2.2	10.62	12.01	12.63	11.57%
	2.4	9.50	10.75	11.33	11.63%
	2.6	8.62	9.76	10.32	11.66%
	2.8	7.91	8.96	9.50	11.70%
WiFi Code[b]	2.6	8.40	9.11	10.29	7.78%
	2.8	7.27	7.92	9.00	8.20%
	3.0	6.44	7.04	8.05	8.51 %
	3.2	5.79	6.35	7.31	8.77 %

a. Length-2304 Rate-0.5 Code in WiMAX

b. Length-1944 Rate-2/3 Code in WiFi

4.4. An Energy Efficient Decoder for WiMAX and WiFi

In this section, I present the implementation details of an improved-energy-efficiency decoder for WiMAX and WiFi LDPC codes. Synthesis results and performance comparisons are reported. The basic architecture of my design is based on partially parallel decoder [48], the block diagram of which can be found in Figure 6. APP messages are stored as two's complement form and quantized by 8 bits, one for sign, five for integer and two bits for fraction. A memory with 1024-bit (128 messages) width and 24-word depth is used to store APP messages. To support high-efficiency decoding at short code length, the proposed decoder works under three modes, where one, two ($z \leq 64$) and four ($z \leq 32$) code blocks are decoded simultaneously for mode 1, 2 and 3, respectively.

A 128×128 four-way-supported PN proposed in section chapter 3 and 128 PEs are implemented to perform layered iterative decoding. Check node messages are stored in a compressed form in check message memory. For each PE, a 16-bit wide 12-word deep memory bank is used to store the magnitude of check messages for 12 rows in 12 different layers. Five and six bits are used to represent a first minimum and a second minimum magnitude of check messages for one row, respectively. Another 5 out of 16 bits are used to encode the position of the first minimum value. Signs of the check messages are stored in a 128-bit wide 96-word deep memory. Therefore, 36,864 (12×16×128+128×96) memory bits are used for check messages. Sum with the APP memory (24,576 bits) 61,440 memory bits are needed in the decoder.

Hardware overhead for proposed early stopping criterion is depicted in Figure 29. In each PE, there is an SC module, which is detailed in the red dash block. For an

SC module, one D flip-flop and one 1-bit binary adder (*XOR* gate) are needed to calculate the least significant bit of a_i of SAV. A SC satisfaction signal (high effective) is obtained at the output of a *NAND* gate whose inputs are least significant bits of SAV. Totally, the hardware overhead of proposed SC is around 2k gate count which can be ignored over the entire design (650k).

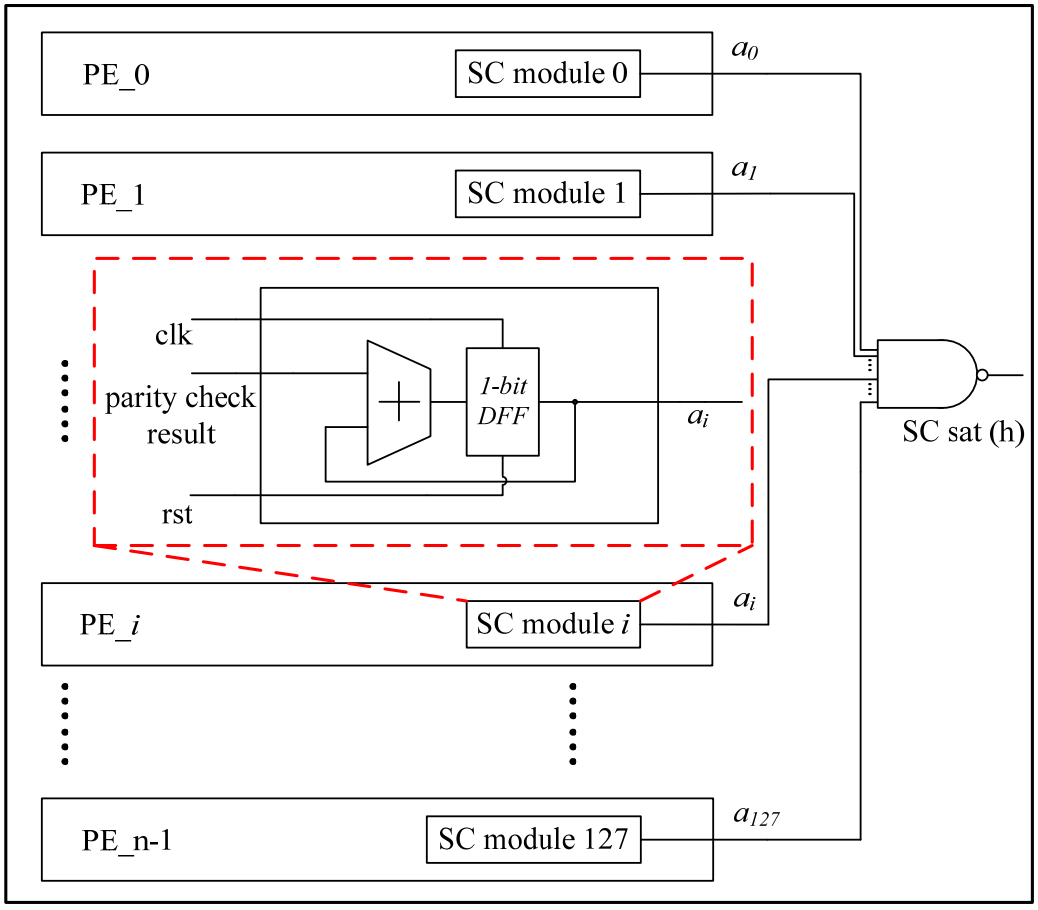


Figure 29 Hardware architecture of proposed lossless ET algorithm

Synthesis result and performance comparisons to previous work are given in Table 5. To show the throughput gain of the proposed decoder architecture, we define the throughput of a conventional decoder working at a fixed frequency and decoding a (2304, 1152) code with a fixed maximum iteration number as one unit, then draw

normalized throughput to code length histogram for 0.5-code-rate 802.16e codes in Figure 30. It shows that the throughput of my proposed decoder is two to four times of the conventional one when the code length is small. The biggest throughput, 1.33 units, can be achieved if the code length equals to 768 or 1536 for which all the PEs in the decoder are active.

Since the decoder is implemented at a synthesis level, a precise measurement based evaluation on energy efficiency is hard to be carried out. Hereafter, I evaluate it by a theoretic analysis.

For the lossless early termination algorithm, compared to conventional early termination, it can achieve an iteration reduction by up to 12%, which directly contributes a 12% energy reduction during an iterative decoding process. Thus I conclude that the proposed early termination algorithm contributes a maximum 12% improvement in terms of the energy efficiency.

For the proposed four-way-supporting permutation network, one instance module counts about 6.2% (42k/651.2k) of gate count in a decoder. Assume that power consumption of a digital circuit is proportional to the gate count at a fixed clock frequency and supply voltage. To achieve a same throughput for short code decoding (z=32 as an example), conventionally extra three permutation networks with an approximately equal hardware overhead are needed in a decoder. Therefore, an 18.6% (3x6.2%) more power is dissipated which implies that my proposed four-way-supporting permutation network contributes an energy efficiency improvement about 18.6%.

Table 5 Synthesis result of proposed decoder and comparisons to others

		Proposed	in [48]	in [49]	in [50]
CMOS Technology (nm)		65	130	180	90
Gate Count		651.2k	–	349.5k	970k
Area (mm^2)		–	3.834	3.39	6.25
Supported Standards		16e&11n	16e	11n	16e
Operating Frequency (Mhz)		264	333	208	150
MAX Iteration #		15	15	5	20
Throughput (Mb/s) @Operating Fre./Iter. z={16e(11n)}	z=32(27)	356	111	134	–
	z=64(54)	356	222	286	–
	z=96(81)	266	333	415	105

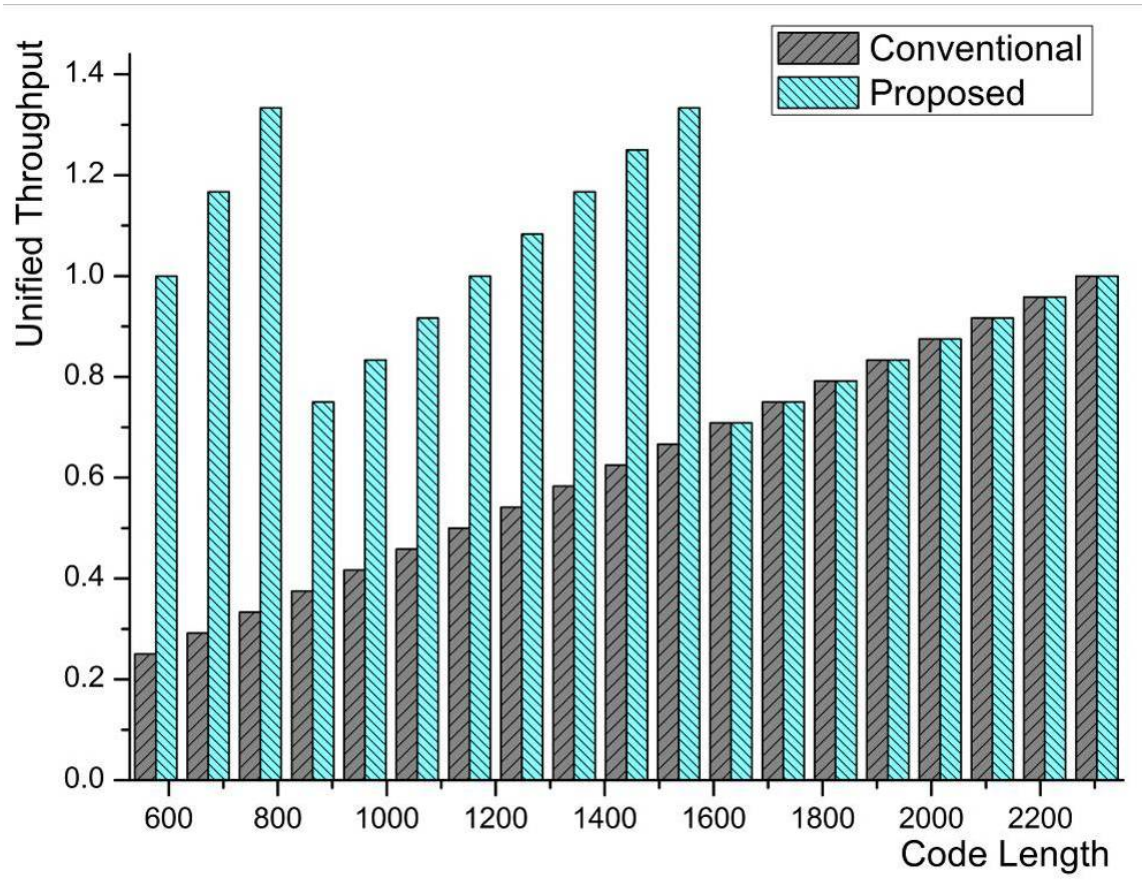


Figure 30 Normalized throughput of a rate-1/2 code in WiMAX

4.5. Summery

In this chapter a lossless ET algorithm based on syndrome accumulation for WiMAX and WiFi LDPC codes is proposed. The proposed ET algorithm is the first algorithm that stops the decoding iterations as soon as all the information bits are corrected. It reduces the number of decoding iterations by a maximum 12% with no BER performance degradation. Moreover, a partially parallel decoder for WiMAX and WiFi applying the techniques proposed in both chapter 3 and 4 is designed and implemented at a synthesis level. By a theoretic analysis, the technique of a four-way-supporting permutation network which is introduced in chapter 3 contributes a maximum 18.6% energy efficiency improvement. While the technique on ET algorithm proposed in this chapter results in a maximum 12% gain on energy efficiency.

5. An Energy-Efficient WPAN LDPC Decoder

5.1. Introduction

IEEE 802.15.3c, as a PHY extension of wireless personal area networks (WPAN) standards, aims at a high-speed wireless data transfer applications such as wireless download site and delivery of uncompressed, high-definition video stream. To support these applications, data rate (including information and redundant bits) up to 5.775Gb/s is required in WPAN.

As mentioned before, QC-LDPC code has been adopted as a forward error correction (FEC) scheme in WPAN. However, it is challenging to design a decoder supporting such a high throughput demand for a code of length 672 only, not mentioning that the decoder must also support four code rates. A kind of architecture called partially parallel decoder [51] was proposed for decoding QC-LDPC code and its relevant optimizations can be found in [10][52]. Unfortunately, owning to its low parallelism, even for the state-of-art work [52], this architecture seems impossible to fulfill the WPAN high data rate requirement. Therefore, high-parallelism QC-LDPC decoder architecture is needed for this novel standard. Recently, Hung et al. proposed a decoder with high parallelism for WPAN. [22] Hung's decoder was implemented on a VLSI chip which could deliver a data throughput 6.6Gbps with 647K gate count. [22]

However, when the parallelism increases, the interconnection network used for transmitting messages between the processing units, check and variable nodes [14], becomes much more complicated since more messages are routed at the same time. This leads to a large logic overhead and severe chip routing congestion. The latter always

results in a low chip area utilization ratio which is one of the critical problems for VLSI LDPC decoder design. [14]

In this chapter, I propose a high-parallelism decoder architecture based on a macro-layer level fully parallel layered decoding. That is to say, all the messages related to one macro layer as defined are processed or updated simultaneously and different macro layers are processed one by one. It takes 4 clock cycles only for the proposed architecture to process one iterations, where for each clock cycle 672 a-posterior probability (APP) messages corresponding to 672 code bits of one code block are updated simultaneously for one macro layer. Meanwhile, an efficient message permutation scheme is developed with utilizing the features of the parity check matrix (PCM). Based on the proposed scheme, the interconnection network which is used to connect the check and variable nodes can be realized without logic cell but only a few wires, where the check and variable nodes in terms of the proposed decoder refer to the message processing units and the registers storing APP messages, respectively. This technique leads to a remarkable reduction on gate count and improvement on area utilization ratio with the comparison to the work in [22]. The proposed novel architecture features high compatibility to support all four code rates defined in the standard. Moreover, to reduce the length of critical path, the decoding process is divided into two phases to employ a frame-level two-stage pipeline. To verify the proposed techniques, the decoder is implemented on a VLSI chip using Fujitsu 65nm LVT CMOS process. It occupies a chip core area of 1.30mm2 with area utilization ratio 86.3%. According to the measurement results, working at 1.2V, 400 MHz and 10 iterations the

proposed decoder delivers a 6.72Gb/s data throughput and dissipates a power of 537.6mW, resulting in an energy efficiency 8.0pJ/bit/iteration. Besides this chip, for the proposed architecture a design with no pipeline stage is also implemented and evaluated at a post-layout level.

In 5.2, following the introduction on WPAN LDPC code PCMs, observations on PCM features are given. Based on the exploited features an efficient permutation scheme based on PCM modification is presented. In 5.3, a novel decoder architecture aiming at high energy efficiency is proposed based on the permutation scheme presented. Details on CMOS chip implementation of proposed decoder are disclosed in 5.4.

5.2. An Efficient Permutation Scheme for WPAN

In this section, firstly, I review the PCMs of LDPC code defined in WPAN standard. After exploitation of the PCMs, two important features are discussed. Then scheme on how these features are used to simplify the permutation network are derived. Meanwhile, some definitions are given to help make a clear description in the rest of this chapter.

5.2.1. WPAN Code PCMs and Definitions

Due to VLSI implementation friendliness, Quasi-Cyclic (QC) LDPC code is adopted as one FEC scheme in WPAN. A PCM of length-672 LDPC code is obtained by replacing each sub-block in a 32-column wide base matrix with a 21x21 sub-matrix as

follows. Shown in the left of Figure 31, an empty sub-block corresponds to a 21x21 all-zero matrix. A sub-block with a non-negative number is replaced by either an identity matrix or its left- cyclically-shifted version, where the number indicates the shift offset. As a result, a PCM containing N=21x32x(1-R) rows and 672 columns is attained through expansion from base matrix, where Nrow and R represent the number of rows and code rate, respectively.

For a code PCM of rate R, the left 672xR and right 672x(1-R) columns correspond to the information and redundant bits in a code frame, respectively. (32xR and 32x(1-R) in base matrix) The PCMs of four code rates included in the standard are uniquely defined by four base matrices, the two of which, rate 5/8 and rate 7/8, are shown in Figure 31.

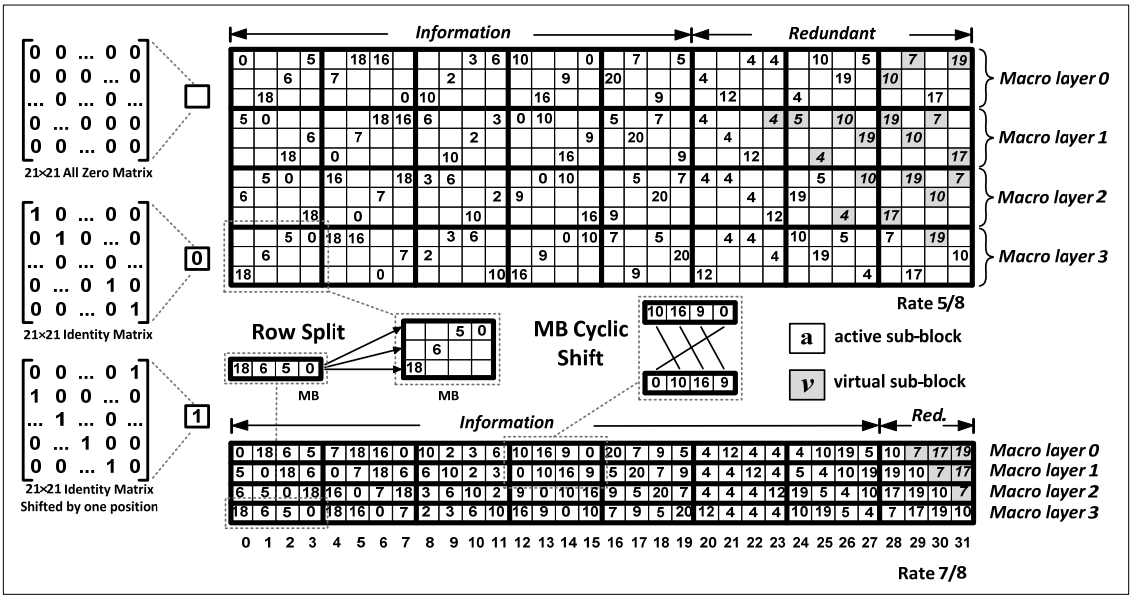


Figure 31 Rate-5/8 and rate-7/8 base matrix of WPAN LDPC codes

In this dissertation a Macro layer (ML) is defined as 8 macro blocks (MB) in a horizontal line in a base matrix or a PCM. For each base matrix, a macro layer contains 32 columns (each MB contains exactly 4 columns), thus for the expanded PCM a same macro layer contains 672 columns (32x21). Since the number of rows contained in a MB is different depending on the code rate, i.e. one for rate 7/8, three for rate 5/8. The corresponding row number of a macro layer is also different. For example, for base matrix, a ML in rate 5/8 has 3 rows matrix (3x21=63 rows for PCM) and a ML in rate 7/8 has only one row (1x21=21 rows for PCM). To describe this problem more clearly, an example of rate 5/8 code showing the relationship between concepts of frame, PCM, base matrix, APP and macro layer is given in Figure 32. (Suppose that the entry in base matrix is either blank or zero.)

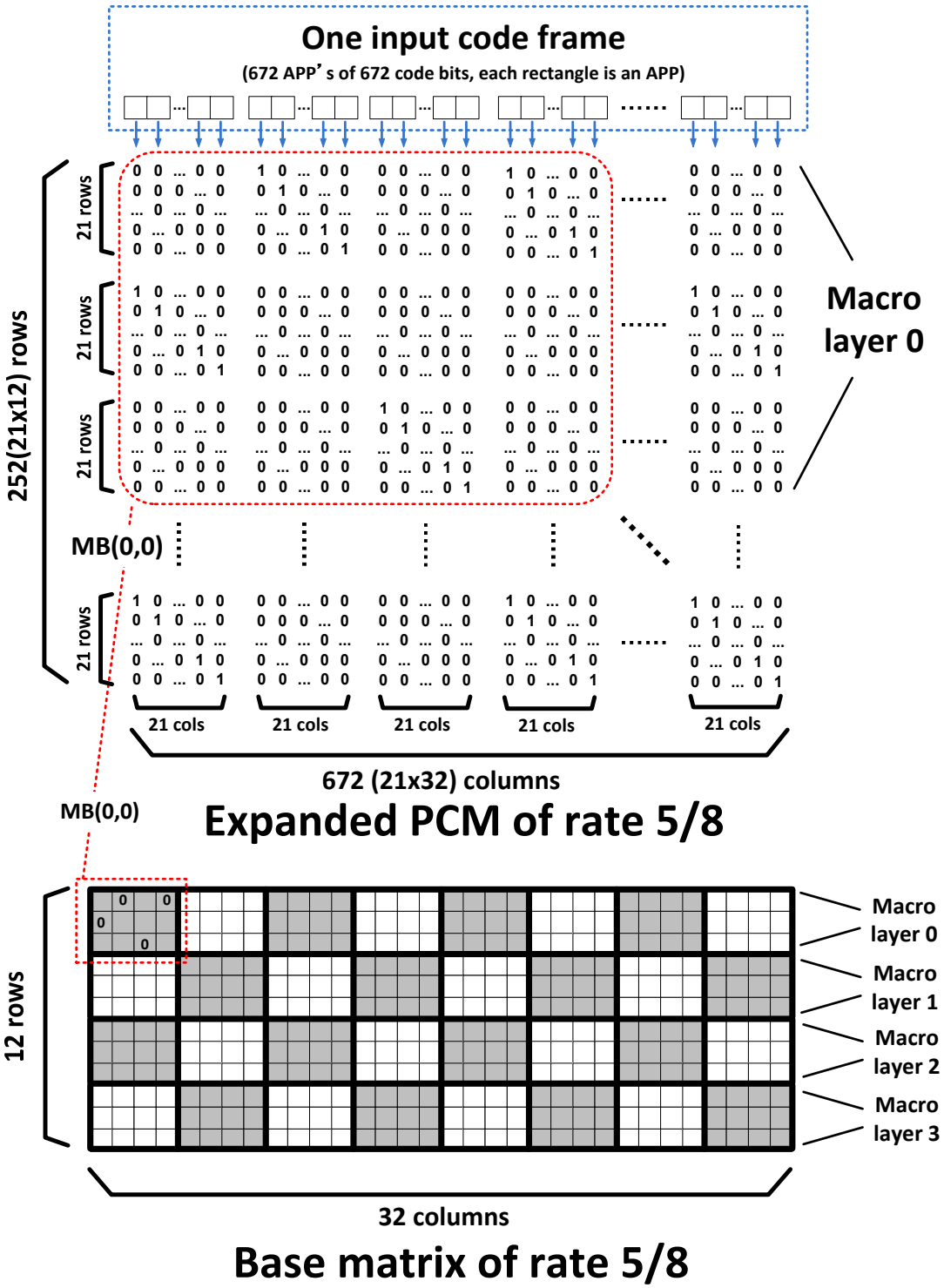


Figure 32 An example of rate 5/8 showing the relationship between frame, PCM, base matrix, APP and macro layer

5.2.2. PCM Features and Proposed Permutation Scheme

Specially, for WPAN length-672 code, base matrices of rate 1/2, 5/8 and 3/4 are all derived from rate-7/8 base matrix by splitting the rows of it. Take rate 5/8 in Figure 31 as an example, twelve rows are obtained by splitting each row from rate 7/8 into three.

As an example in Figure 31, the lower-left MBs of rate 5/8 and rate 7/8 are both highlighted in for your reference. In this example, the row of rate-7/8 MB is split to three rows of rate-5/8 MB, where four sub-blocks with number 18, 6, 5 and 0 are distributed into four positions of a twelve-sub-block MB of rate-5/8. Moreover, a Macro Layer (ML) is defined as a matrix partition consisting of 8 macro blocks, which are in a horizontal line. For all code rates, each code matrix contains 4 macro layers in vertical.

Row split feature indicates the relations between two corresponding MBs from two base matrices in same macro layer. The relations between MBs in the same base matrix from different macro layer is called MB cyclic shift. Except for the redundant bit part, MB in a macro layer can be attained by cyclically right shifting the MB in the above macro layer by one sub-block. Two MBs in the first and second macro layers of rate-7/8 base matrix are illustrated as an example to show this feature in Figure 31.

To make the two features also satisfied for the redundant bit part of the matrix, some virtual sub-blocks, the backgrounds of which are highlighted in grey, are added in the original base matrix, shown in Figure 31. Rate 1/2 and 3/4 are shown in Figure 33. Since the APP messages related to the virtual sub-block are actually not involved in the corresponding macro layer decoding, it is also called an inactive sub-block, the

corresponding column of which is called an inactive column. Nevertheless, a non-empty sub-block in the original base matrix is called active sub-block and its related column in a macro layer is called an active column. Each macro layer contains 32 active or inactive sub-blocks (columns) in total.

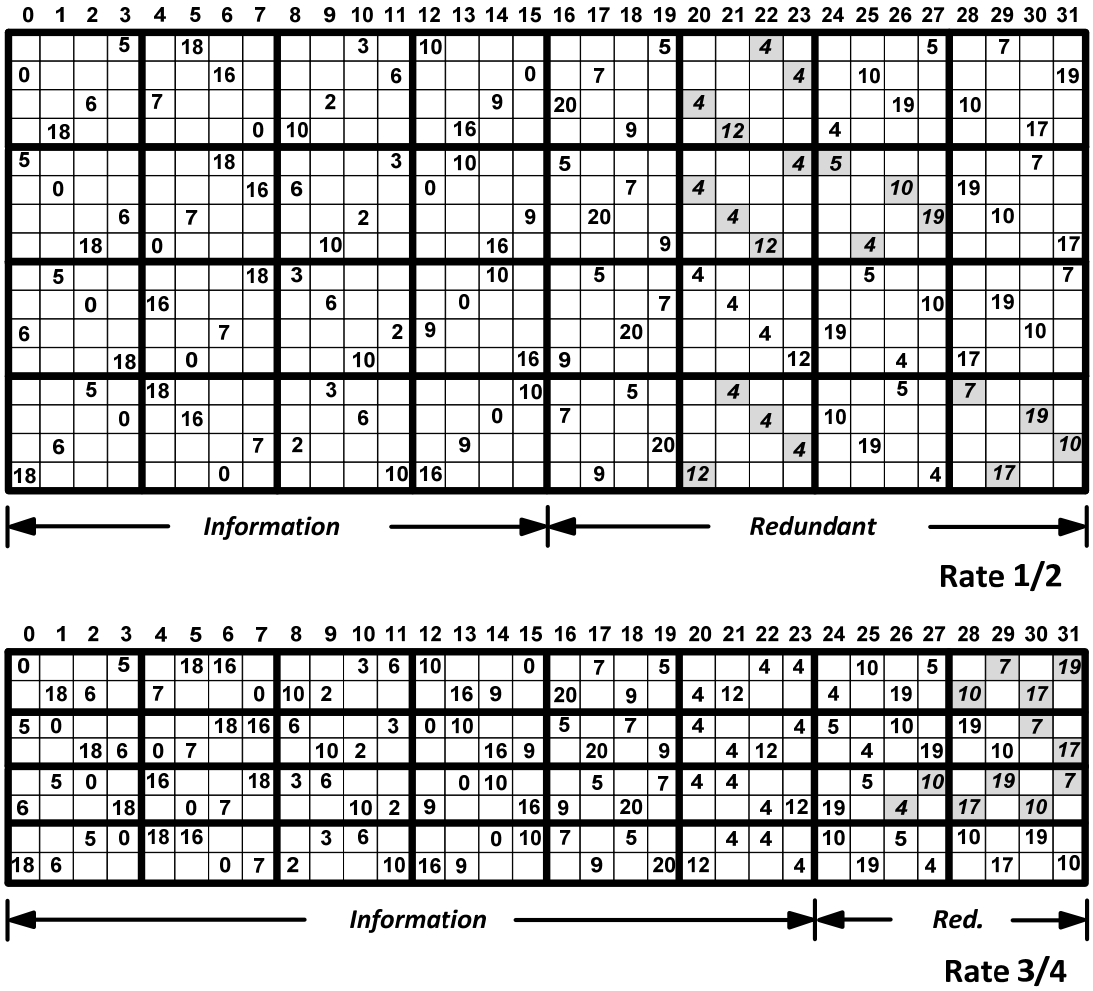


Figure 33 Based matrix with virtual sub-blocks for rate 1/2 and 3/4

Back to the PCM features, the row split feature implies that the interconnections between check and variable nodes inside two splitting related macro

layers of different code rates are almost the same. The only difference is the number of groups, to which the sub-blocks belong. For example, each macro layer of rate-5/8 contains three rows thus the 32 sub-blocks are distributed into 3 groups by allocating 16 in the first group and both 8 in the second and the third, respectively. For the MB cyclic shift feature, it implies that the interconnection inside one macro layer is the same as the macro layer above in the same base matrix if the MB is cyclically shifted by one sub-block. Therefore, according to the two implications above, it is possible for a simple network to be shared for all macro layers with the utilization of the two features.

5.3. A Novel Fully Parallel Decoder for WPAN

In this section, after brief introduction on decoding algorithm used in this design I propose a macro-layer level fully parallel decoder. Based on the descriptions on decoder architecture, we go deeper to see more details, breakdown of the macro layer processing engine (MLPE) which is the computation core of the design, the timing schedule, buildings of permutation networks and message storage, etc.

5.3.1. Decoder Architecture

For WPAN LDPC code, each column in one macro-layer as defined contains at most one active sub-block, which supports a layered decoding at a macro layer level. During the decoding process, two kinds of messages, APP and check (CHK) messages both in log likelihood ratio (LLR) forms are updated serially according to the algorithm. To reduce the hardware overhead for check message updating, normalized min-sum

algorithm with factor 0.75 proposed in [44] and discussed in chapter 2 is used.

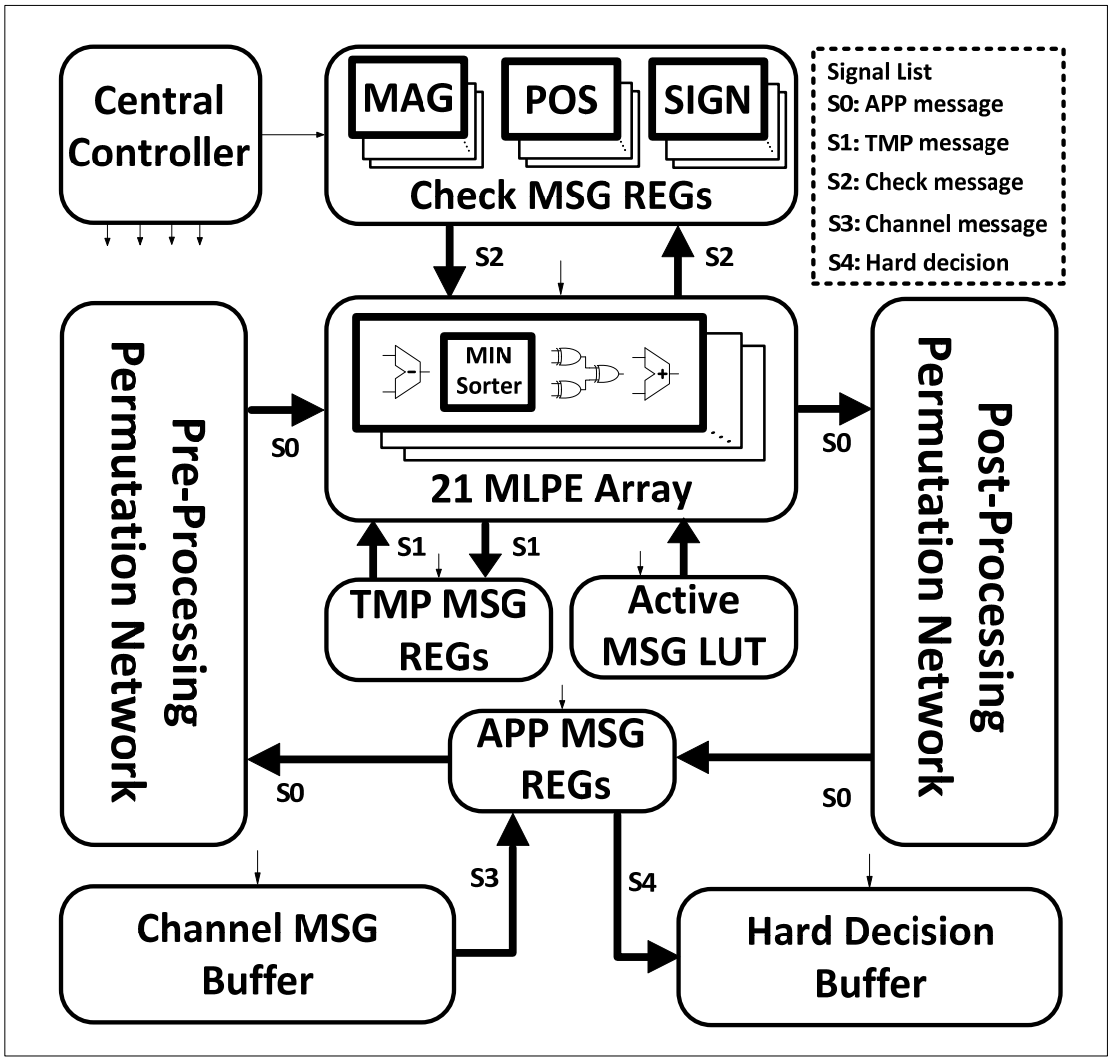


Figure 34 Block diagram of proposed decoder architecture

Block diagram of proposed decoder architecture is given in Figure 34. A decoding process starts from initializing the APP registers (REGs) by the channel messages in the buffer. APP messages are then sent into a message processing engine array through a pre-processing permutation network. Each macro layer processing engine (MLPE) in the array is capable of simultaneously updating 32 APP messages, a

part of one macro layer. Note that each macro layer contains 672(21x32) messages in total. The MLPE is flexible to 4 different code rates, the details of which will be discussed later. Since each row in base matrix is expanded to 21 rows in the PCM, 21 replicas of MLPE are implemented with each for one row to update 672 messages of an entire macro layer concurrently. Different macro layers are updated in serial.

To reduce the length of critical path, a code-frame level pipeline schedule is applied in the design. Therefore the decoding process is divided into two phases which are called temporary (TMP) and APP message updating phases, respectively. Two code frames are processed alternatively at two phases in the decoder. The temporary (TMP) messages and the check (CHK) messages generated during the decoding process are stored in the dedicated registers.

All 672 APP messages in 32 columns are sent to the MLPE array simultaneously. An active message look up table (LUT) is used to indicate which column among 32 is active or involved in the current macro-layer updating. If a column is inactive, the corresponding 21 APP messages are bypassed and unchanged in 21 MLPEs. Otherwise they will be updated according to the algorithm.

After MLPE processing, APP messages, including both updated and bypassed, are reordered via a post-processing permutation network and written back into APP registers for the next macro-layer processing. When the maximum decoding iteration number is achieved, hard decisions are outputted into the hard decision buffer.

5.3.2. Macro Layer Processing Engine

Macro layer processing engine (MLPE) is the main computation unit in the design. An array consisting 21 instantiates of MLPE is capable of processing 672 messages at the same time with each MLPE responsible for 32. As mentioned above, the critical path is shortened by introducing a frame-level pipeline schedule which means that two code frames are being updated at two updating phases alternatively.

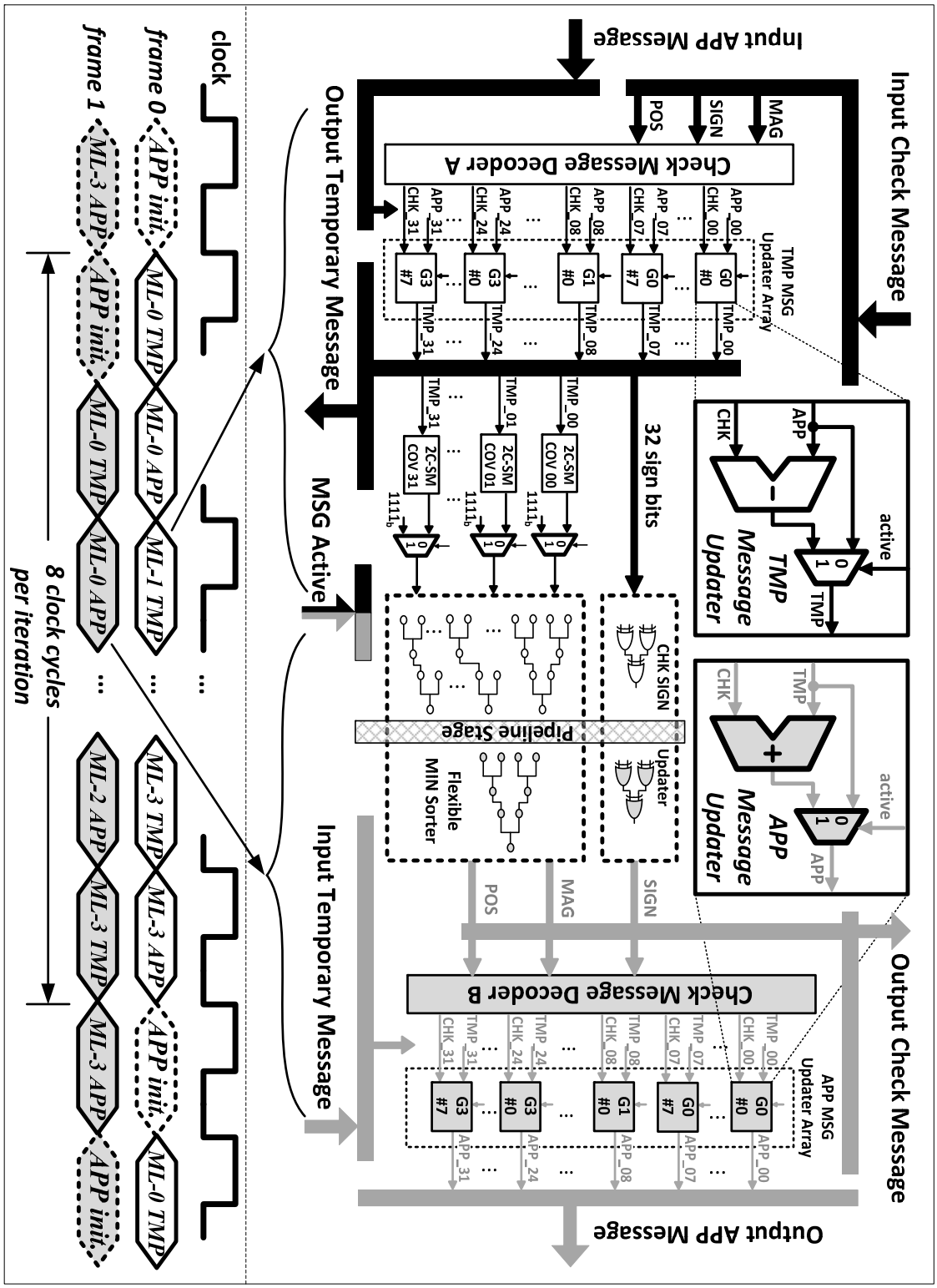


Figure 35 Building blocks of one MLPE and timing schedule

Building blocks of one MLPE and the timing schedule of the decoding process are both shown in Figure 35. For the MLPE status shown in Figure 35, frame 0 and frame 1 (in black and white) are two independent frames both containing 672 APPs processed. Due to the pipeline scheduling, they are processed independently and alternatively in two different decoding phases, called temporary message updating (first) and APP message updating (second) decoding phases where the frame 0 and 1 are in first and second phases, respectively.

Suppose the two code frames in Figure 35, frame 0 and 1, are both of rate-7/8. The code frame 0 is at a temporary message updating phase. 32 APP messages corresponding to different 32 columns in the base matrix are inputted into MLPE as well as the check message of the current row. Since the check messages are stored in a compressed way in the check message registers, i.e. magnitudes of first and second minimum values, positions of first minimum values and signs, a check message decoder is needed to generate 32 uncompressed check messages. A temporary message is generated by a temporary message updater for each APP message with the control signal from active message LUT. If an APP message is active, the temporary message is the result of APP message subtracted by check message, which is also called variable message in some literatures. Otherwise the inputted APP message is bypassed and unchanged as a temporary message. In one hand, the newly generated 32 temporary messages are outputted and stored in the registers for next-phase updating. Meanwhile they are also converted from two's complement number to sign-magnitude form by 32 converters. A 32-input sorter is implemented in each MLPE to find the first and second

minimum values among the variable messages for min-sum algorithm. The sorter is designed based on a tree structure which was previously proposed in [53]. Before being sorted the temporary messages are preprocessed by filtering out the APP messages bypassed before because they are not involved in the current macro layer updating. A bypassed APP message is replaced by a biggest magnitude number to eliminate its influence on sorting, which makes sure that the minimum values found are among variable messages only. This is easily realized by inserting 32 two-to-one multiplexers, the control signals of which are also from active message LUT as shown in Figure 35. Similarly, the sign bits are preprocessed before being inputted to the check sign updater composed of a tree of exclusive-or gates. Pipeline registers divide these two modules into two parts to improve the global timing performance.

For the code frame 1 at the APP message updating phase, in one hand, check messages which are newly generated by sorter and sign updater are outputted to the registers. Meanwhile they are converted to uncompressed forms by the check message decoder for updating APP messages. Similar to the temporary message generation, the APP messages are updated by an array of 32 APP message updaters with the control signals from active message LUT as shown in Figure 35.

Shown in the timing schedule chart of Figure 35, by changing the updating phases for each frame alternatively, processing of two code frames by one decoding iterations is finished in 8 clock cycles. Therefore, averagely, it takes 4 clock cycles to perform one decoding iterations for one code frame. For code rates other than 7/8, each row in the base matrix is related to either 8 or 16 active or virtual sub-blocks. Therefore

32 data paths are divided into 4 groups with each group dealing with 8 sub-blocks. Either a single or two combined groups of data paths are assigned to process one row in a macro layer and an entire macro layer can still be processed concurrently. Take rate-5/8 code as an example, there are 16, 8 and 8 sub-blocks relating to the first, second and third row of one macro layer, respectively. Therefore, the first two, the third and the fourth groups are assigned to those three rows relatively. To support multiple code rates, the sorter needs to find minimum values for different groups independently. It can be achieved by modifying the original sorter with small logic overhead thanks to its tree structure. The local wires of MLPE are regular and structured because multiplexing 32 APP messages into 4 groups is done outside of MLPE by the permutation networks which will be discussed later.

5.3.3. Message Permutation Networks

Implementation of interconnection between computation units is always a critical problem for LDPC decoder design not only because of the related logic overhead but also the routing congestion which largely affect the chip area utilization ratio. It becomes more severe when the decoding parallelism is higher. For the WPAN LDPC decoder design in the work [22], some techniques were used to mitigate the complexity of interconnections. As a result, the number of multiplexers used for message routing is reduced by 64% compared to a direct implementation and the area utilization ratio is 73.3%.

However, I argue that a finer result can be achieved if the features of the code

matrix exploited and discussed in section 2 are well utilized. For my proposed WPAN decoder, only two simple networks, namely, pre- and post-processing permutation networks are implemented to solve the interconnection problem. They are designed to concurrently route 672 APP messages which compose one macro layer, to designated inputs of MLPE array. The proposed two permutation networks contain only wires but no logic cells like multiplexers used in [22] and they are shared by all 16 macro layers of the four code rates.

Shown in Figure 36, pre-processing permutation network mainly contains three parts, 32 fixed cyclic shift networks (CSN), one group divider and one mapping network. 32 blocks of APP messages (21 in each block) corresponding to 32 columns in the base matrix are inputted into 32 CSNs. Each CSN performs fixed-offset-cyclic shift for one block of messages according to the offset number in active or virtual sub-blocks in the first macro layer of rate-7/8 base matrix with virtual sub-block added. An example of a CSN with offset 19 is given in Figure 36. After cyclic shift, 32 blocks of messages are then fed into a module, called group divider. According to the distribution of non-empty sub-blocks in four rows of the first macro layer in rate-1/2 base matrix (see the appendix), these 32 blocks are divided into 4 groups. The expression (i) in the figure represents the message block with column index i in the base matrix, where i is from 0 to 31.

For rate 1/2, these four groups of APP messages are updated independently in MLPE which is configured to four-row process mode. For rate 5/8, group 0 and 1 are merged and updated together in MLPE. Similarly, when processing rate-3/4 code,

MLPE is configured to two-row process mode with each row being dealt with two groups. After final mapping, 32 blocks of APP messages are shuffled into 21 clusters with each containing 32 messages to interface with the MLPE array. An APP message in block j with inner offset k at the input side of this mapping network is shuffled to cluster k with inner offset j at the output side.

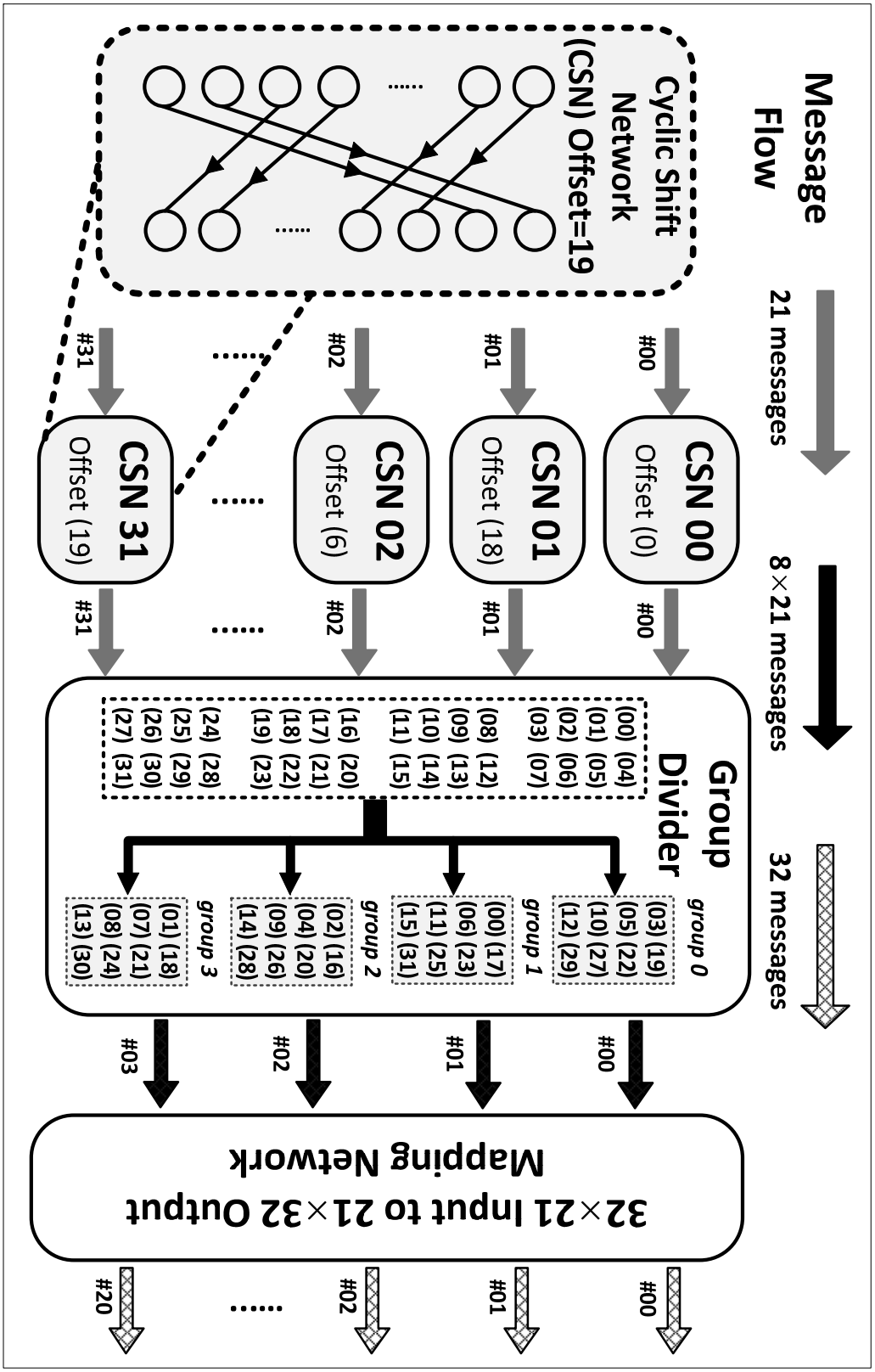


Figure 36 Block diagram of a pre-permutation network

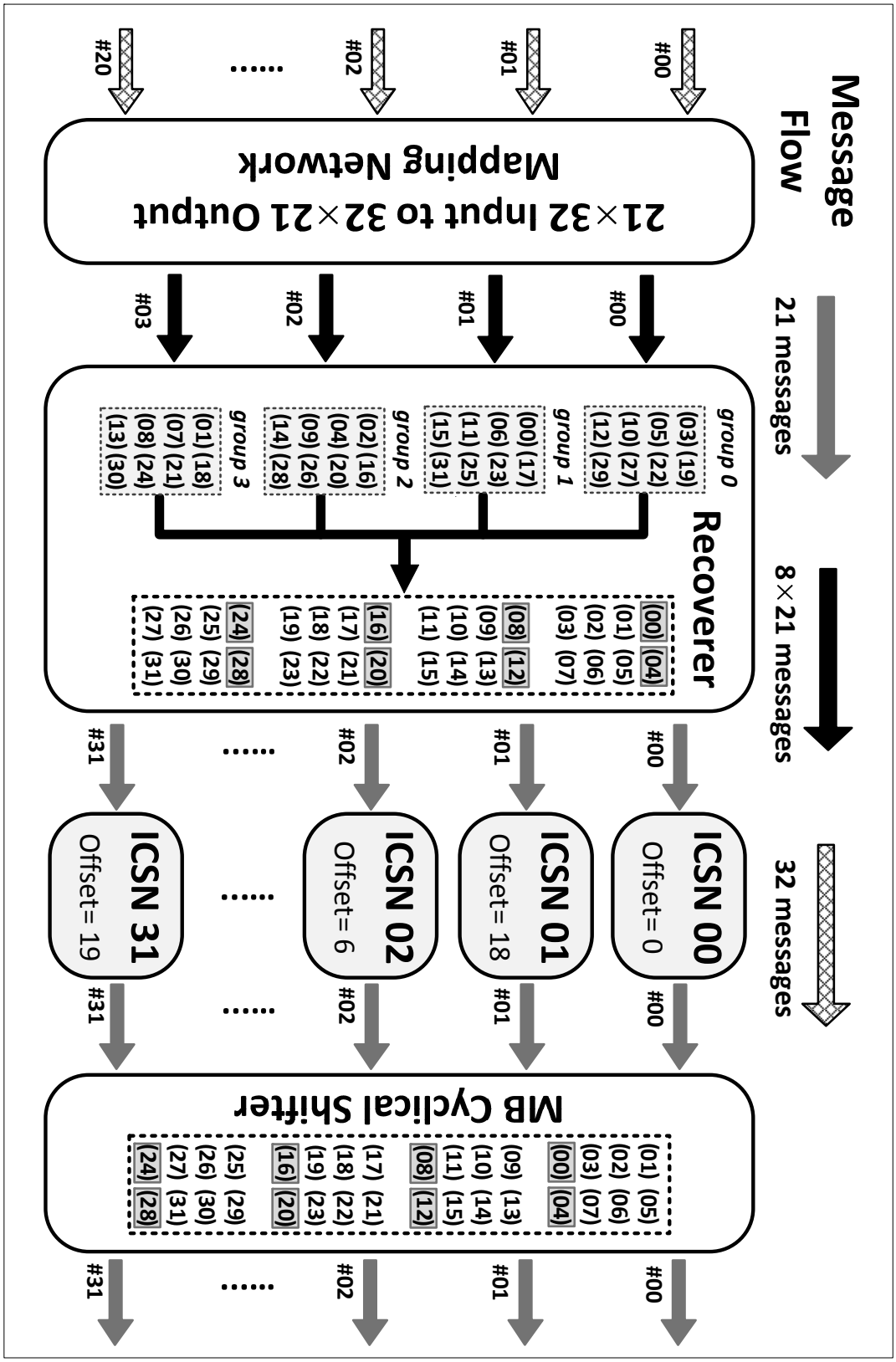


Figure 37 Block diagram of a post-permutation network

Figure 37 shows the details of the post-processing permutation network. Firstly, the processed 672 APP messages from 21 MLPEs are remapped to 32 blocks. Thirty-two blocks in different 4 groups are reordered to 8 MBs with original order via a module called recoverer. Thirty-two fixed inverse cyclic shift networks (ICSN) cyclically permute the messages in an inverse direction of CSN to recover the original order, 0 to 20, within a block. Following the MB cyclic shift feature mentioned in section 2, MB cyclical shifter is used to cyclically permute each MB by one block. In the figure, the first block of each MB is highlighted for your notice.

As mentioned before, utilization of the matrix features will mitigate the interconnection complexity. In my proposal, the group divider and the recoverer modules designed in the two permutation networks help to utilize the row-split feature. This makes the network reusable for all first macro layers in 4 code rates. The utilization of MB cyclic shift feature relies on the implementation of MB cyclical shifter in the post-processing permutation network. Consequently, messages related to the second, the third and the fourth macro layers can be permutated by using the same network. Therefore, these two permutation networks are reused for processing all 16 macro layers from 4 code rates for my proposed decoder.

Moreover, since all the connections between signals inside or outside the modules in these two permutation networks are fixed wires, electrically, both of them can be treated as 672-input-to-672-output fixed mapping networks only consisting 672 pairs of wires. Thus the routing congestion overhead introduced by the two permutation networks is very small, which is revealed by the implementation results given later

5.3.4. Message Storage and Non-pipeline Decoder

For the proposed design, all the messages or data are stored in the on-chip registers. The APP and the temporary message are quantized by 6 bits and they occupy 8,064 bit (2x672x6 bits) registers. Check message is quantized by 5 bits with one bit for sign and 4 for magnitude. Check messages of one macro layer are stored in one word using a compressed data form containing three parts, the magnitudes of the first and second minimum values, positions of first minimum values and signs. Considering the worst case which is rate-1/2, each word contains 76 bits in all where 32 bits (2x4x4 bits) are for magnitudes, 12 bits (4x3 bits) are for positions and 32 bits are for signs. Due to two code frame pipeline processing, 12,768 bits (2x21x76x4) as total are used for check message in this design.

A non-pipeline decoder with the same architecture aiming at smaller area, higher energy efficiency and moderate bit error ratio (BER) performance is also implemented and evaluated at a post-layout level. Two decoding phases are combined to one which means no temporary messages are needed thus saving the registers. Since only one code frame is processed at the same time, the overhead for the check message storage is as half as the pipeline decoder. However, without pipeline structure, the maximum clock frequency is lower. To achieve the required data rate, the number of iterations per code frame is reduced.

5.4. ASIC Implementation and Measurement Results

To verify the proposed architecture, the decoder with pipeline schedule is fabricated using 65nm CMOS technology. The chip is evaluated by an FPGA board based evaluation system. The chip power consumption is measured to evaluate the power efficiency. The decoder without pipeline is implemented at a post-layout level the performance of which is estimated based on the EDA simulation and measured results of the fabricated chip. In this section I give the details on implementation flow, evaluation and measurement as well as the performance comparisons to the previously published work.

5.4.1. Chip Implementation

RTL design is compiled and synthesized using Synopsys Design Complier. Cadence SoC Encounter is used for placement and routing. The decoder is implemented using Fujitsu 65nm LVT 1P12M CMOS design kit. A PLL macro provided by vender is implemented to generate the on-chip clock.

For the pipeline decoder it occupies a die area 4.41mm^2 (2.1mmx2.1mm) and the core area is 1.30mm2 (1.14mmx1.14mm). The die photo and distribution of top modules is shown in Figure 38.

The post-routing gate count for the decoder core is 430.8K which does not include test logic and buffers. By using my proposed message permutation networks the routing congestion overhead of the decoder chip is greatly reduced, which can be revealed by the following two facts:

(1). Compared to synthesis result, degradation of timing performance of post-layout is less than 7%.

(2). Final chip area utilization ratio achieves 86.3%.

For the non-pipeline decoder, the core area is 1.10mm^2 with area utilization ratio 82.6%. The gate count for decoder core is 334K. The maximum frequency in worst case is 227MHz according to the post-layout timing report.

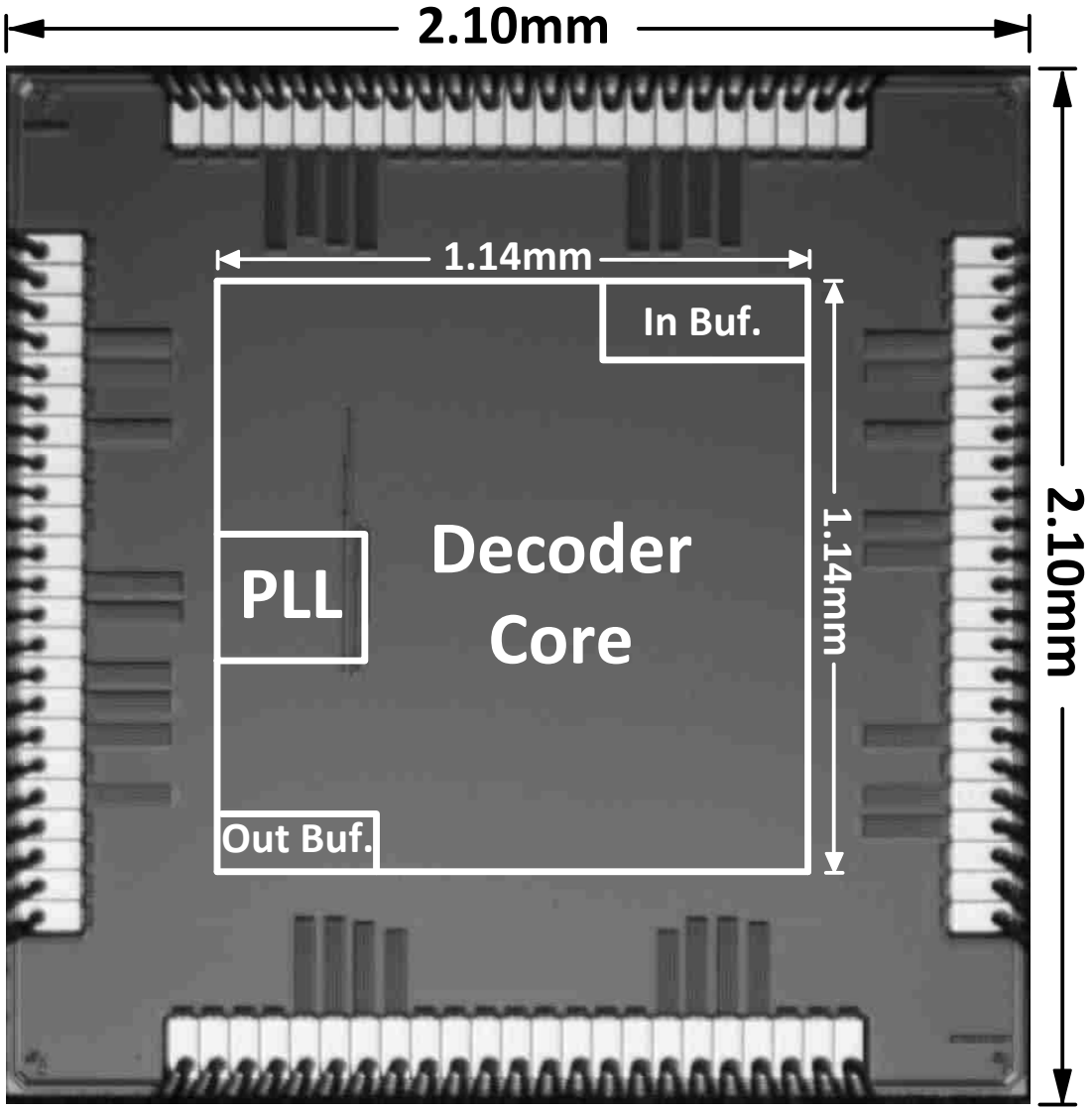


Figure 38 Chip die micro photo with top module distribution

5.4.2. Evaluation and Measurement

Evaluation and measurement work mainly includes decoding performance verification and power measurement, both of which relied on an evaluation system developed based on Mitsubishi-MU300 LSI evaluation board. [54]

5.4.2.1. Evaluation System

To verify the logic function of the chip, a verification system based on Mitsubishi MU300 LSI evaluation board is developed. The entire verification system includes PC, MU300 FPGA board, software and hardware interface between PC and board, DUT board as an interface to connect FPGA and ASIC chip.

In this system, an Altera Stratix II FPGA (EP2S130F1508C4) is used to implement the verification core which generates the test pattern and analyze the test response. Quartus 10.0 is used to compile the RTL code and generate the FPGA configuration file. Several Altera provided tools, Signal Tap II Logic Analyzer, In System Source and Probe Editor and Memory Content Editor, are interfaced with FPGA for data collection and debugging.

Block diagram of the verification core is shown in Figure 39. According to the run parameters set in this test, chip instructions such as, setting the code rate, iteration number and workload are automatically generated. Meanwhile, test code frames BPSK- AWGN log likelihood ratio (LLR) channel messages are generated by an array (3 copies) of AWGN noise generators, the core of which are implemented according to the algorithm and architecture proposed in [55][56]. This noise generator features high speed and low logic element cost with moderate noise quality, which is far

than enough for my case. The stimulus or test patterns are then sent into to sinks, the ASIC chip and the decoder core implemented on FPGA.

After run, chip responses including chip status and decoded code frame or hard decision data are sent back to verification core. A comparator is used to compare the responses from the chip and on-FPGA decoder core for evaluating the correctness of chip logic function. The chip is concluded working correctly if there is no difference between the responses from two decoders among all one million test code frames. Moreover, the BER performance of the ASIC decoder is evaluated by analyze the hard decision data with corresponding channel status. Counters are used to count the decoding failures like number of incorrect code frames, bits, etc. Finally, a run log is generated and written into the dedicated memory for further evaluation.

On chip clock is adjusted by on-chip PLL and the clock system in FPGA which can be easily designed using the Quartus platform. According to the design report, FPGA hardware overhead is listed in Table 6. Photos showing the real verification system are given in Figure 40.

Table 6 FPGA hardware overhead of evaluation system

FPGA Resource	**EP2s130F1508C4**
Combinational ALUTs	53,022/106,032 (50%)
Total registers	25,083/106,032 (24%)
Pins	99/1,127 (9%)
Block memory bits	833,632/6,747,840 (13%)
DSP block elements	72/504 (15%)
PLL	1/12 (9%)

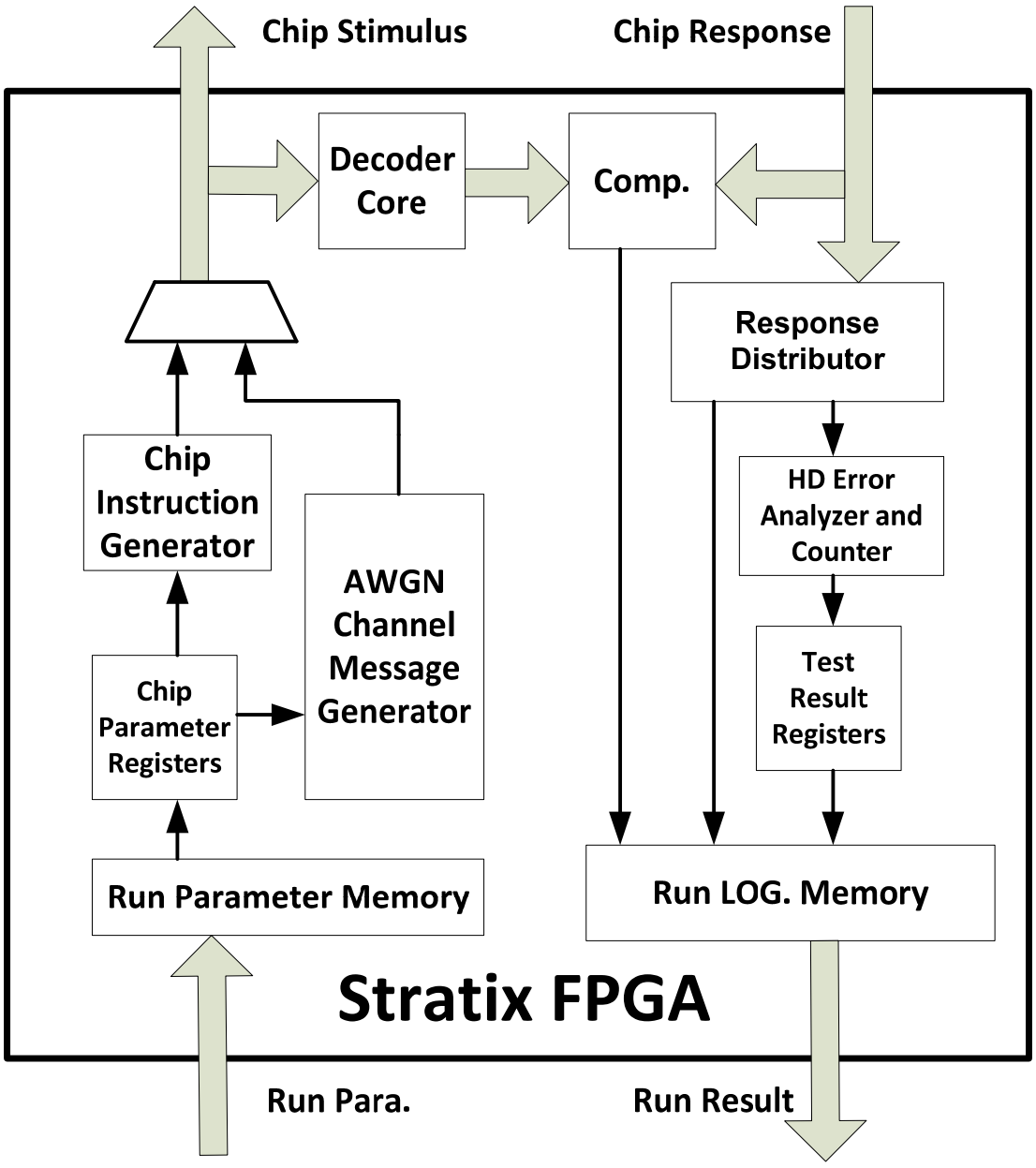


Figure 39 Block diagram of evaluation core based on FPGA

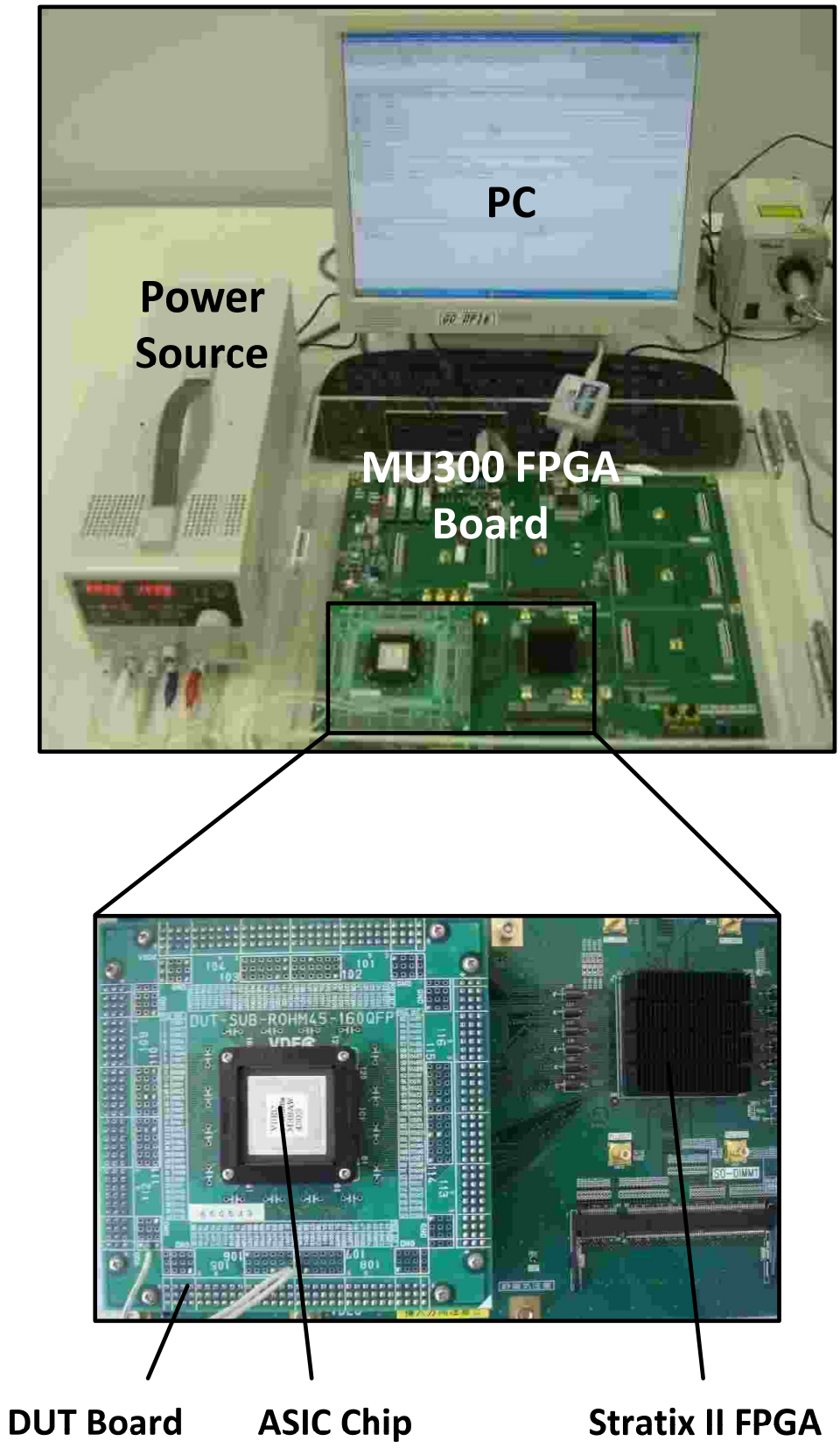


Figure 40 Photos of a real FPGA based chip evaluation system

5.4.2.2. Evaluation on Decoding Performance

For additive white Gaussian noise (AWGN) channel with binary phase shift keying (BPSK) modulation, BER over signal noise ratio (SNR) curves for four code rates with 10 iterations at data throughput 6.72Gb/s, clock frequency 400MHz, are depicted in Figure 41. At BER 10^{-5}, one can observe a maximum 0.25dB implementation loss due to fixed-point quantization compared to PC software based simulation using floating point with the same decoding algorithm, i.e. normalized min-sum layered algorithm with factor 0.75. BER curves at clock frequency 200MHz, 5 iterations are also given in Figure 41, where a maximum 0.75dB performance degradation compared to 10-iteration decoding is observed.

For a multipath Rayleigh fading channel with perfect phase compensation, the simulated results for both floating and fixed-point numbers with 10 iterations and same decoding algorithm are shown in Figure 42. Depending on the code rates, a 3.75 to 9.5dB performance degradation is observed compared to the AWGN channel at BER 10^{-5}. A maximum 1.2dB implementation loss at same BER carried out in the simulation also is shown in the figure.

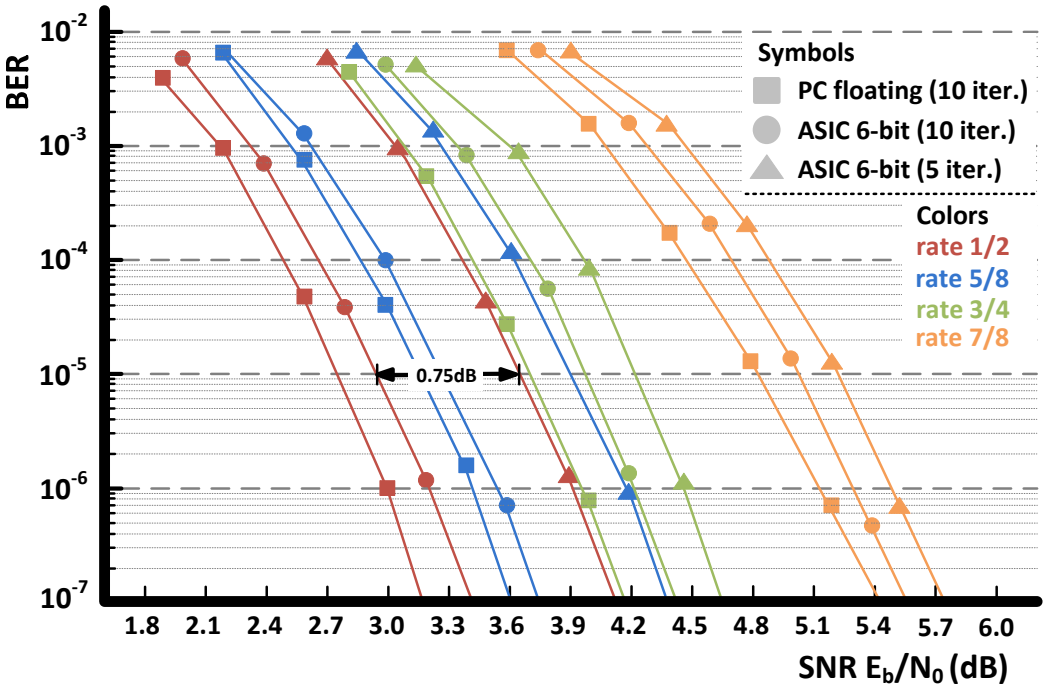


Figure 41 BER performance for AWGN channel

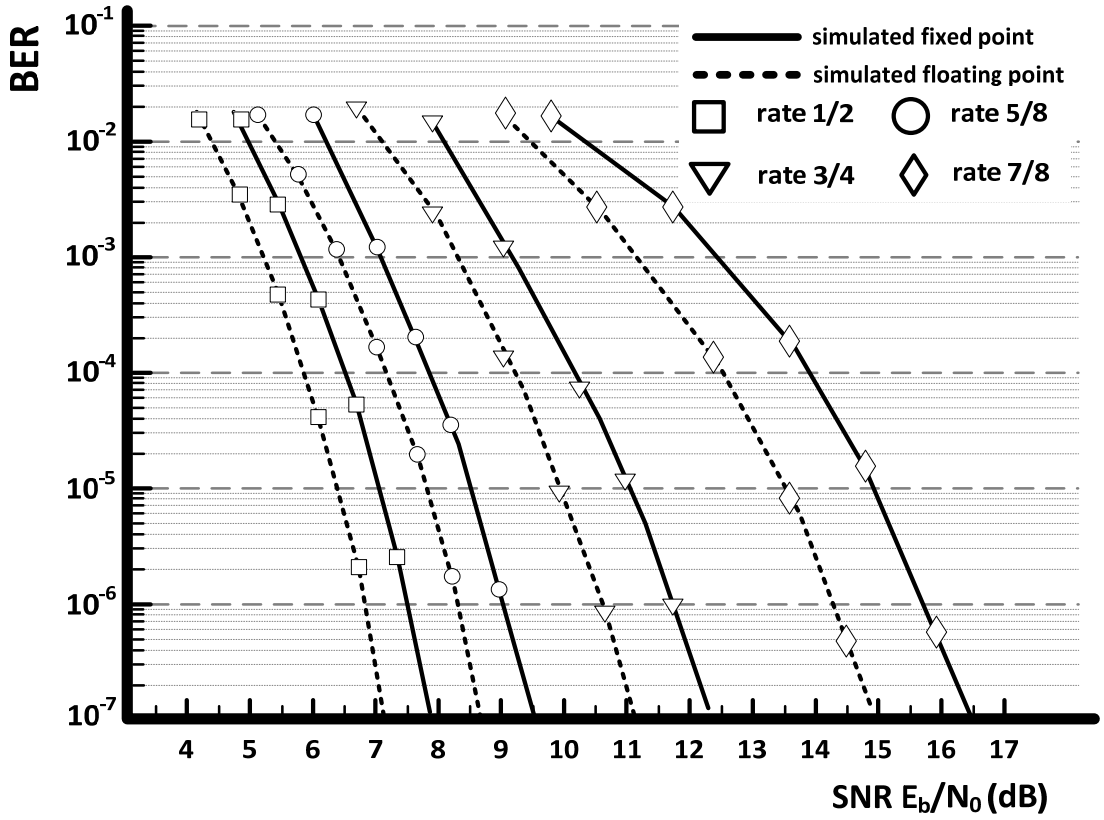


Figure 42 BER performance for multipath fading channel

5.4.2.3. Power Measurement

At 1.2V supply voltage, the chip maximum frequency observed is 450MHz. Data throughput (T) can be calculated following (18), where f, N and I indicate the clock frequency, code length and the iteration number, respectively. Working at 400MHz and 10 iterations, the chip delivers a data throughput 6.72Gb/s which is 16% higher than the standard's requirement.

$$T = \frac{f \cdot N}{4I} \tag{18}$$

Chip power is measured with rate-1/2 code for AWGN channel at SNR 2.8dB and temperature 300K. Results are shown in Table 7. An estimated power for non-pipeline design based on EDA simulation and chip measurement results is also listed in Table 7.

Table 7 Power consumption of the proposed decoder

	Pipeline decoder (chip measured)		**Non-pipeline (estimated)**	
Iteration	10		5	
Frequency	400MHz	200MHz	200MHz	100MHz
Supply voltage	1.2V	1.0V	1.2V	1.0V
Power	537.6mW	209.8mW	231.8mW	90.7mW
Throughput	6.72Gb/s	3.36Gb/s	6.72Gb/s	3.36Gb/s
Energy efficiency	8pJ/bit·iter	6.2pJ/bit·iter	6.9pJ/bit·iter	5.4pJ/bit·iter

5.4.3. Comparisons

Table 8 summarizes the key characteristics and comparison to other ASIC LDPC decoders published recently. My proposed decoder features small area, high throughput and achieves the best power efficiency when comparing to the designs for other code specifications.

Particularly, compared to the decoder designed for the same code in [22], the proposed pipeline (non-pipeline) decoder achieves 16.7% (29.5%), 33.5% (48.4%) and 49% (56.1%) improvements in terms of area, gate count and energy efficiency, respectively. Moreover, the chip area utilization ratio is improved from 73.3% to 86.3%. These improvements mainly come from the proposed message permutation scheme which largely reduces the hardware overhead and mitigates routing congestion of the design.

Table 8 Key characteristics and comparison to other measured decoders

Publication	this work		A-SSCC'10	ESSCIRC'09	JSSC'10	VLSI'10	JSSC'08
	pipeline	non-pipeline	Hung [22]	Chen [57]	Zhang [58]	Xiang [52]	Darabiha [16]
Technology (nm)	65	65	65	90	65	130	130
Supply voltage (V)	1.2	1.2	1.0	0.8	1.2	1.2	1.2
Code length	672	672	672	2048	2048	576-2304	660
Code rate	1/2, 5/8 3/4, 7/8	1/2, 5/8 3/4, 7/8	1/2, 5/8 3/4, 7/8	15/16	0.84	1/2, 2/3 3/4, 5/6	0.74
Quantization bits	6 bits	6 bits	6 bits	6 bits	4 bits	6 bits	4 bits
Algorithm	layered	layered	layered	layered	flooding	layered	flooding
Chip core area (mm^2)	1.30	1.10	1.56	3.84	5.35	3.03	7.3
Decoder core gate count	430K [1)]	334K [1)]	647K	708K	-	470K	690K
Clock frequency (MHz)	400	200	197	120	700	214	300
Iterations per frame	10	5	5	4	8	10	15
Data throughput (Gb/s)	6.72	6.72	6.6 [2)]	6.15 [2)]	13.3	0.955	3.3
Power consumption (mW)	537.6	231.8 [4)]	361	191.2	2,800	397	1383
Energy efficiency (pJ/bit/iteration)	8.0	6.9 [4)]	10.9	8.3	58.7	42	27.9
Normalized energy [3)] efficiency (pJ/bit/iteration)	8.0	6.9 [4)]	15.7	12.5	58.7	16.8	11.2

1) 1.92μm^2 for each two-input one-output NAND gate;

2) Original information throughput is scaled to data throughput;

3) Normalized to 65nm, 1.2V technology with $P_{65} = P_x \cdot \frac{C_{65}}{C_x} \cdot \left(\frac{V_{65}}{V_x}\right)^2$ assuming $\frac{C_{65}}{C_{90}} = 0.67$, $\frac{C_{65}}{C_{130}} = 0.4$;

4) Result estimated based on EDA simulation and chip measurement;

5.5. Summery

This chapter presents an LDPC decoder design for IEEE 802.15.3c application. To achieve the data rate required in the standard, macro-layer level fully parallel decoder architecture is proposed. To mitigate the permutation complexity raised by high parallelism processing, a matrix-exploited message permutation scheme is applied. A decoder with pipeline for high clock frequency is implemented in 65nm CMOS. Measurement result shows that it achieves 6.72Gbps at 400MHz, 10 decoding iterations with energy efficiency 8.0pJ/bit/iteration. A low-profile design without pipeline for the same architecture is also implemented at post-layout level. The proposed decoder achieves highest power efficiency compared to LDPC decoders for different applications with technology scaling taken into consideration. Compared to the state-of-art design for WPAN standard, my proposed pipeline (non-pipeline) decoder improves 16.7% (29.5%), 33.5% (48.4%) and 49% (56.1%) in area, gate count and energy efficiency.

6. Conclusion

This dissertation mainly contributes to energy-efficient VLSI LDPC code decoder design. The contribution is related to two types of decoders, partially parallel decoder and fully parallel decoder.

For partially parallel architecture, this work advances the decoding energy efficiency by two proposals. 1) An improved permutation network supporting four-code concurrent decoding at short code modes, gains up to 18.6% energy efficiency. 2) A lossless early termination algorithm reduces decoding workload by 12% at most, equivalent to a 12% energy efficiency improvement. Combining these two techniques, a partially parallel decoder for WiMAX and WiFi applications is proposed with a logic synthesis level implementation.

For fully parallel architecture, this work advances the state-of-art energy-efficient WPAN LDPC decoder. A novel scheme is proposed to alleviate the complexity of permutation circuitry. The scheme uses modified parity check matrix for permutation, which extremely utilizes the structure of the WPAN code. As a result, the permutation network is implemented using no logic gates but only wires. Measurement results of ASIC decoder show that the energy efficiency is improved by 49%.

For future work, I would be interested in LDPC convolutional code decoder and stochastic decoding algorithms.

References

[1] S. Lin and D. J. Castello Jr. Error Control Coding. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2004.

[2] R. Gallager, "Low-Density Parity-Check Codes," Cambridge, MA: MITPress, 1963.

[3] D.J.C. MacKay, "Good codes based on very sparse matrices," IEEE Trans.Inf. Theory, vol. 45, no. 3, pp. 399-431, Mar. 1999.

[4] "IEEE P802.11 Wireless LANs WWiSE Proposal: High throughout extension to the 802.11 Standard," Aug. 2004, IEEE 11-04-0886-00-000n.

[5] "IEEE Std 802.16e-2005 and IEEE Std 802.16-2004/Cor 1-2005 (Amendment and Corrigendum to IEEE Std 802.16-2004)," , 2006, IEEE Std 802.16e-2005.

[6] T.T.S.I. digital video broadcasting (DVB) second generation framing structure for broadband satellite applications. http://www.dvb.org.

[7] IEEE 802.15.3c. wireless medium access control (MAC) and physical layer specifications for high rate wireless personal area networks (WPANs), 2009.

[8] Z. Chen, X. Zhao, X. Peng, D. Zhou and S. Goto, "A High-Parallelism Reconfigurable Permutation Network for IEEE 802.11n and 802.16e LDPC Decoder", International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS 2009), pp. 85-88, Kanazawa, Japan, pp. 85-88, Dec. 2009.

[9] Z. Chen, X. Zhao, X. Peng, D. Zhou and S. Goto, "An Early Stopping Criterion for Decoding LDPC Codes in WiMax and WiFi Standards", 2010 IEEE International

Symposium on Circuits and Systems (ISCAS 2010), pp. 473-476, Paris, France, Jun. 2010.

[10] Z. Chen, X. Zhao, X. Peng, D. Zhou and S. Goto, "A High Parallelism LDPC Decoder with an Early Stopping Criterion for WiMax and WiFi Application", IPSJ Transaction on System LSI Design Methodology, Vol. 3, pp. 292-302, Aug. 2010.

[11] Z. Chen, X. Peng, X. Zhao, Q. Xie, L. Okamura, D. Zhou and S. Goto, "A Macro-Layer Level Fully Parallel Layered LDPC Decoder SoC for IEEE 802.15.3c Application," 2011 IEEE International Symposium on VLSI Design, Automation and Test (VSLI-DAT 2011), Hsinchu, Taiwan, Apr. 26, 2011.

[12] Z. Chen, X. Peng, X. Zhao, L. Okamura, D. Zhou and S. Goto, "A 6.72-Gb/s 8pJ/bit/iteration IEEE 802.15.3c LDPC decoder chip", IEICE TRANS. Fundamentals, vol. E94-A, No. 12, pp. 2587-2596, Dec. 2011.

[13] A. J. Blanksby and C. J. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity check code decoder," IEEE J. Solid-State Circuits, vol. 37, pp. 404–412, Mar. 2002.

[14] T. Mohsenin and B. Baas. Trends and challenges in ldpc hardware decoders. In Signals, Systems and Computers, 2009 Conference Record of the Forty-Third Asilomar Conference on, pp. 1273-1277, November. 2009.

[15] A. Darabiha, A.C. Carusone and F.R. Kschischang. "A 3.3-Gbps bit-serial block-interlaced Min-Sum LDPC decoder in 0.13-um CMOS," In IEEE Custom Integrated Circuits Conference, pp. 459-462, 2007.

[16] A. Darabiha, A.C. Carusone and F.R. Kschischang, "Power Reduction Techniques

for LDPC Decoders", IEEE Jounal of Solid-State Circuits, vol. 43, pp. 1835-1845, Aug., 2008.

[17] T. Mohsenin and B. Baas. High-throughput LDPC decoders using a multiple Split-Row method. In ICASSP, volume 2, pages 13{16, 2007.

[18] T. Mohsenin and B. Baas. A split-decoding message passing algorithm for low density parity check decoders. Journal of Signal Processing Systems, February 2010.

[19] T. Mohsenin, D. Truong, and B. Baas. An improved Split-Row Threshold decoding algorithm for LDPC codes. In ICC, 2009.

[20] T. Mohsenin, D. Truong, and B. Baas. Multi-Split-Row Threshold decoding implementations for LDPC codes. In ISCAS, May 2009.

[21] T. Mohsenin, D. Truong, and B. Baas. A low-complexity message-passing algorithm for reduced routing congestion in ldpc decoders. Circuits and Systems I: Regular Papers, IEEE Transactions on, 57(5):1048{1061, May. 2010.

[22] S. Hung, S. Yen, C. Chen, H. Chang, S. Jou, C. Lee, "A 5.7Gbps row-based layered scheduling LDPC decoder for IEEE 802.15.3c applications", in A-SSCC 2009, December 2010.

[23] S. Yen, S. Hung, C. Chen, H. Chang, S. Jou and C. Lee,"A 5.79-Gb/s Energy-Efficient Multirate LDPC Codec Chip for IEEE 802.15.3c Applications", IEEE J. Solid-State Circuits, vol. 99, 2012.

[24] K. Baek, H. Lee and C. Choi, "A High-Throughput LDPC Decoder Architecture for High-Rate WPAN Systems", in ISCAS, May 2011.

[25] H. Shirani-Mehr, T. Mohsenin and B. Baas, "A reduced routing network architecture for partial parallel LDPC decoders", in ACSSC, 2011.

[26] M. M. Mansour and N. R. Shanbhag, "A 640-Mb/s 2048-bit programmable LDPC decoder chip," IEEE J. Solid- State Circuits, vol.41, pp. 684–698, Mar. 2006.

[27] J. L. Fan, "Array codes as low-density parity-check codes," in Proc. 2nd Int. Symp. Turbo Codes, Brest, France, Sept. 2000, pp. 545–546.

[28] Kou, J. Xu, H. Tang, S. Lin, and K. Abdel-Ghaffar, "On circulant low density parity check codes," in Proc. 2002 IEEE Int. Symp. Information Theory, 2002.

[29] Z. Li and B. V. K. V. Kumar, "A class of good quasi-cyclic low-density parity check codes based on progressive edge growth graph," in Proc. 38th Asilomar Conf. Signals, Systems and Computers, 2004, vol. 2, pp. 1990–1994.

[30] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," IEEE Trans. Inform. Theory, vol. 50, pp. 1788–1793, Aug. 2004.

[31] S. Olcer, "Decoder architecture for array-code-based LDPC codes," in Proc. IEEE Global Telecommun. Conf. (GLOBECOM), 2003, vol. 4, pp. 2046–2050.

[32] V. E. Benes, "Optimal rearrangeable multistage connecting networks," Bell Syst. Tech. J., vol. 43, pp. 1641–1656, 1964.

[33] J. Tang, T. Bhatt, V. Sundaramurthy, and K. K. Parhi, "Reconfigurable shuffle network design in LDPC decoder," in Proc. Int. Conf. Appl.Specific Syst., Archit. Process. (ASAP), Sep. 2006, pp. 81–86.

[34] K. K. Gunnam, G. S. Choi, M. B. Yeary, and M. Atiquzzaman, "VLSI

architectures for layered decoding for irregular LDPC codes of WiMax," in Proc. Int. Conf. Commun. (ICC), Jun. 2007, pp. 4542–4547.

[35] D. Oh and K. K. Parhi, "Area efficient controller design of barrel shifter for reconfigurable LDPC decoders," in Proc. IEEE Int. Symp. Circuits Syst., May 2008, pp. 240–243.

[36] D. Oh and K. K. Parhi, "Low-Complexity Switch Network for Reconfigurable LDPC Decoders," IEEE Trans. VLSI System, vol. 18, issue 1, pp. 85-94, 2009.

[37] C. Liu, C. Lin, H. Chang, C. Lee, and Y. Hsu, "Multimode message passing switch networks applied for QC-LDPC decoder," in Proc. IEEE ISCAS, May 2008, pp. 752–755.

[38] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in Proc. IEEE Workshop on Signal Processing Systems, Oct. 2004, pp. 107–112.

[39] X. Shih, C. Zhan, C. Lin, A. Wu, "An 8.29 mm2 52 mW Multi-Mode LDPC Decoder Design for Mobile WiMAX System in 0.13 μm CMOS Process," IEEE Jounal of Solid-State Circuites, vol. 43, pp. 672-683, Sept., 2008.

[40] R. Y. Shao, S. Lin, and M. P. C. Fossorier, "Two simple stopping criteria for Turbo decoding," IEEE Trans. Commun., vol. 47, pp. 1117 – 1120, Aug. 1999.

[41] W.E. Ryan, "An introduction to LDPC codes," in CRC Handbook for Coding and Signal Processing for Recoding Systems (B. Vasic, ed.), CRC Press, 2004.

[42] R.M. Tanner, "A recursive approach to low complexity codes," IEEE Trans. Information Theory, pp. 533-547, Sep. 1981.

[43] M.P.C. Fossorier, “Reduced complexity iterative decoding of low-density parity check codes based on belief propagation,” IEEE Trans. Commun., vol.47, no.5, pp.673–680, May 1999.

[44] J. Chen and M. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," IEEE Trans. Commun., vol. 50, pp. 406–414, Mar. 2002.

[45] J. Zhao, F. Zarkeshvari, and A.H. Banihashemi, “On implementation of min-sum algorithm and its modifications for decoding low-density Paritycheck (LDPC) codes,” IEEE Trans. Comms., vol. 53, no. 4, pp. 549-554, Apr. 2005.

[46] M. M. Mansour, "High-Throughput LDPC Decoders," IEEE Trans. VLSI, vol.11, NO 6, DEC. 2003.

[47] E. Sharon, "Efficient Serial Message-Passing Schedules for LDPC Decoding,” IEEE Trans. Information Theory, Vol. 53, Nov., 2007, pp.4076–4091.

[48] T. Brack, "A Synthesizable IP Core for WIMAX 802.16E LDPC Code Decoding," in Proc. IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications, Sept. 2006, pp. 1-5.

[49] Studer, "Configurable high-throughput decoder architecture for quasi-cyclic LDPC codes," in Proc. 42nd ACSSC, Oct. 2008, pp. 1137-1142.

[50] C. Liu, "An LDPC Decoder Chip Based on Self-Routing Network for IEEE 802.16e Applications," IEEE Journal of Solid-State Circuits, Vol. 43, Mar. 2008, pp. 684-694.

[51] Y. Sun, M. Karkooti, and J.R. Cavallaro, "VLSI Decoder Architecture for High

Throughput, Variable Block-size and Multi-rate LDPC Codes," in Proc. IEEE Int. Symp. Circuits and System 2007(ISCAS'07), pp. 2104-2107, May, 2007.

[52] B. Xiang and X. Zeng, "A 4.84 mm2 847-955 Mb/s 397 mW Dual-Path Fully-Overlapped QC-LDPC Decoder for the WiMAX System in 0.13 μm CMOS," in Proc. IEEE Symp. VLSI Circuits 2010 (VLSIC'10), pp. 211-212, June, 2010.

[53] C. WEY, M. Shieh and S, Lin, "Algorithms of Finding the First Two Minimum Values and Their Hardware Implementation," IEEE Trans. Circuit and Systems I, vol. 55, No. 11, pp. 3430-3437, 2008.

[54] Online data sheet of MU-300 LSI evaluation board, available at http://www.mms.co.jp/powermedusa/products/em.html

[55] C. Wallace, “Fast pseudorandom generators for normal and exponential varieties,” ACM Trans. Math. Softw., vol. 22, no. 1, pp. 119-127, 1996.

[56] D. Lee, W. Luk, J.D. Villasenor, G. Zhang and P.H.W. Leong, "A Hardware Gaussian Noise Generator Using the Wallace Method," IEEE Trans. VLSI Syst., vol. 13, No. 8, pp. 911-920 2005.

[57] C. Chen, "A 11.5-Gbps LDPC Decoder Based on CP-PEG Code Construction," IEEE ESSCIRC 2009 Conf., pp. 412-415.

[58] Z. Zhang, V. Anantharam, M.J. Wainwright and B. Nikolic, "An Efficient 10GBASE-T Ethernet LDPC Decoder Design with Low Error Floors," IEEE Journal of Solid-State Circuit, vol. 45, No. 4, pp.843-855, 2010.

[59] X. Peng, Z. Chen, X. Zhao, F. Maehara and S. Goto, "High Parallel Variation

Banyan Network Based Permutation Network for Reconfigurable LDPC Decoder," in Proc. 21st IEEE International Conference on Application-specific Systems Architectures and Processors (ASAP'10), pp. 233-238, July, 2010.

Publication

Journal Papers

[1] Xiongxin Zhao, Zhixiang Chen, Xiao Peng, Dajiang Zhou and Satoshi Goto, "DVB-T2 LDPC Decoder with Perfect Conflict Resolution", IPSJ Transaction on System LSI Design Methodology Vol.5, to appear in Aug. 2012.

[2] Zhixiang Chen, Xiao Peng, Xiongxin Zhao, Leona Okamura, Dajiang Zhou and Satoshi Goto, "A 6.72-Gb/s 8pJ/bit/iteration IEEE 802.15.3c LDPC decoder chip", IEICE TRANS. Fundamentals, vol. E94-A, No. 12, pp. 2587-2596, Dec. 2011.

[3] Xiao Peng, Xiongxin Zhao, Zhixiang Chen, Fumiaki Maehara, Satoshi Goto, “Generic Permutation Network for QC-LDPC Decoder”, IEICE Trans. Vol.E93-A, No.12, pp. 2551-2559, Dec. 2010.

[4] Zhixiang Chen, Xiongxin Zhao, Xiao Peng, Dajiang Zhou and Satoshi Goto, "A High Parallelism LDPC Decoder with an Early Stopping Criterion for WiMax and WiFi Application", IPSJ Transaction on System LSI Design Methodology, Vol. 3, pp. 292-302, Aug. 2010.

[5] Xiao Peng, Zhixiang Chen, Xiongxin Zhao, Fumiaki Maehara and Satoshi Goto, "Permutation Network for Reconfigurable LDPC decoder based on Banyan Network", IEICE Trans. Electronics, Vol.E93-C,No.3, pp.270-278, Mar. 2010.

International Conference Papers

[1] Xiongxin Zhao, Zhixiang Chen, Xiao Peng, Dajiang Zhou and Satoshi Goto, "DVB-T2 LDPC Decoder with Perfect Conflict Resolution", 2011 IEEE International Symposium on VLSI Design, Automation and Test (VSLI-DAT 2011), Hsinchu, Taiwan, Apr. 23-25, 2012.

[2] Xiao Peng, Zhixiang Chen, Xiongxin Zhao, Dajiang Zhou and Satoshi Goto, “A 115mW 1Gbps QC-LDPC decoder ASIC for WiMAX in 65nm CMOS”, IEEE Asian Solid-State Circuits Conference (A-SSCC 2011), pp. 317-320, Jeju, Korea, Nov. 14-16, 2011.

[3] Qian Xie, Qian He, Xiao Peng, Ying Cui, Zhixiang Chen, Dajiang Zhou, Satoshi Goto," A High Parallel Macro Block Level Layered LDPC Decoding Architecture based on Dedicated Matrix Reordering", 2011 IEEE Workshop on Signal Processing Systems (SiPS 2011), pp. 122-127, Oct. 3-5, 2011.

[4] Ying Cui, Xiao Peng, Zhixiang Chen, Xiongxin Zhao, Yichao Lu, Dajiang Zhou and Satoshi Goto , "Ultra Low Power QC-LDPC Decoder with High Parallelism,"The 24th IEEE International SOC Conference (SOCC2011), pp. 142-145, Taipei, Taiwan, Sept.26-28, 2011.

[5] Zhixiang Chen, Xiao Peng, Xiongxin Zhao, Qian Xie, Leona Okamura, Dajiang Zhou and Satoshi Goto, “A Macro-Layer Level Fully Parallel Layered LDPC Decoder SoC for IEEE 802.15.3c Application,” 2011 IEEE International Symposium on VLSI Design, Automation and Test (VSLI-DAT 2011), Hsinchu, Taiwan, Apr. 26, 2011.

[6] Qian Xie, Zhixiang Chen and Satoshi Goto, "A Sorting-Based Architecture of Finding the First Two Minimum Values," 2011 IEEE 7th International Colloquium on Signal Processing and its Applications (CSPA 2011), Penang, Malaysia, Mar. 4, 2011.

[7] Xiongxin Zhao, Zhixiang Chen, Xiao Peng, Dajiang Zhou and Satoshi Goto, "A BER performance-aware early termination scheme for layered LDPC decoder", 2010 IEEE Workshop on Signal Processing Systems (SiPS 2010), pp. 416-419, Oct. 6th, 2010.

[8] Xiao Peng, Zhixiang Chen, Xiongxin Zhao, Fumiaki Maehara, Satoshi Goto, "High Parallel Variation Banyan Network Based Permutation Network for Reconfigurable LDPC Decoder", 2010 21st IEEE International Conference on Application-specific Systems Architectures and Processors (ASAP 2010), pp 233-238, July 7th, 2010.

[9] Zhixiang Chen, Xiongxin Zhao, Xiao Peng, Dajiang Zhou and Satoshi Goto, “An Early Stopping Criterion for Decoding LDPC Codes in WiMax and WiFi Standards”, 2010 IEEE International Symposium on Circuits and Systems (ISCAS 2010), pp. 473-476, Paris, France, Jun. 2010.

[10] Zhixiang Chen, Xiongxin Zhao, Xiao Peng, Dajiang Zhou and Satoshi Goto “A High-Parallelism Reconfigurable Permutation Network for IEEE 802.11n and 802.16e LDPC Decoder”, International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS 2009), pp. 85-88, Kanazawa, Japan, pp. 85-88, Dec. 2009.

[11] Zhixiang Chen, Xiongxin Zhao and Satoshi Goto, "A Memory Efficient Check

Message Quantization Scheme for LDPC decoder", 2009 International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC 2009), pp. 1412-1415, Jeju, Korea, Jul. 2009.

[12] Xiongxin Zhao, Zhixiang Chen and Satoshi Goto, "HIGH EFFICIENCY ARCHITECTURE FOR DVB-S2 BASED LDPC DECODER", 2009 International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC 2009), pp. 1558-1561, Jeju, Korea, Jul. 2009.

Domestic Conference Papers

[1] Xiongxin Zhao, Zhixiang Chen, Xiao Peng, Dajiang Zhou, Satoshi Goto, "A Low-power and Performance-aware DVB-S2 LDPC Decoder with Layered Scheduling", 電子情報通信学会技術研究報告（情報理論）2010, pp.93-97, Sep. 22th, 2010.

[2] Qian Xie, Zhixiang Chen and Satoshi Goto, "A Sorting-Based Architecture of Finding the First Two Minimum Values", 電子情報通信学会技術研究報告（情報理論）2010, pp. 57-61, Sep. 22th, 2010.

[3] Xiongxin Zhao, Zhixiang Chen, Xiao Peng and Satoshi Goto, “An Multi-rate LDPC decoder system on FPGA”, 電子情報通信学会技術研究報告（情報理論）2009, Tokyo, Japan, pp. 1-4, Sep. 2009.

[4] Zhixiang Chen, Xiongxin Zhao, Xiao Peng and Satoshi Goto, “A High-Parallelism Reconfigurable Permutation Network for IEEE 802.11n/802.16e LDPC Decoder”, 電子情報通信学会技術研究報告（情報理論）2009, Tokyo, Japan, pp. 5-8, Sep. 2009.