Waseda University Doctoral Dissertation

Ultra-low Power LDPC Decoder Design with High Parallelism for Wireless Communication System

PENG, Xiao

Graduate School of Information, Production and Systems
Waseda University

July 2011

ABSTRACT

Low-Density Parity-Check (LDPC) codes were originally proposed in 1962 by Gallager. Since the contemporary investigations in concatenated coding shaded LDPC codes and the hardware at that time could not support effective codec implementations, the paper of LDPC codes was left on the shelf for over thirty years. Until Mackay published his work in 1996, LDPC codes began to be strongly promoted in the error correcting code (ECC) area because of the excellent error correction capability. The bit error rate (BER) performance which comes very close to the Shannon limit (within 0.0045dB) of AWGN channel capacity was achieved with constructed irregular LDPC codes and long code length (on the order of 10^6 to 10^7).

Taking LDPC codes into real applications, the outstanding performance and partial parallel property make the structured quasi-cyclic LDPC (QC-LDPC) codes become ECC scheme of many emerging wireless communication standards, such as Wireless Metropolitan Area Network (WMAN or WiMAX), Wireless Local Area Network (WLAN or WiFi) and Wireless Personal Area Network (WPAN). All the wireless communication systems encounter challenges on two contrary aspects: on one hand, high throughput and low power. Sufficient throughput is the base of various next generation mobile communication applications including world-wide wireless internet and high quality multimedia service; on the other hand, low power requirements also become extremely urgent in mobile phones and other handheld terminals, such as iPad, Kindle and etc. Thus, LDPC decoder of wireless communication systems should deliver both the capability of high throughput and high energy efficiency. For this purpose, fast convergent decoding algorithm and highly parallel architecture are the most efficient approaches.

In the algorithm level, currently the Turbo-decoding Message-Passing (TDMP, also called the layered decoding) algorithm has shown its significance which achieves about two times faster converging speed compared with the conventional Two-Phase Message-Passing (TPMP, also called flooding) algorithm for QC-LDPC codes. Based on this TDMP algorithm, the partial parallel architecture realizes balance between high throughput and reasonable hardware cost.

In the architecture level, generally the decoding process of TDMP algorithm is carrying out layer by layer. Within each layer, different designs with the corresponding scheduling give different partial parallel architectures. The original architecture processes the nonzero sub-blocks block by block. As a result, the clock cycle number of one iteration is approximately equal to the nonzero sub matrix number of parity check matrix (PCM). The state-of-art design by Bo in VLSI Symposium 2010 uses two sets of processing units to improve the parallelism and processes two nonzero sub-blocks in one clock cycle. However, it does not bring double parallelism gain because of the data conflicting problem. By utilizing the nonzero sub-matrix reordering and complex bypass controlling scheme, it can partially solve the data conflicting problem. Therefore, two novel architectures for TDMP algorithm are proposed in this dissertation to overcome this shortcoming.

This dissertation contains 5 chapters which are listed as follows:

Chapter 1 [**Introduction**] introduces the basic knowledge of LDPC codes, including representation of LDPC codes, the property of QC-LDPC codes and the evolution of LDPC decoding algorithm.

Chapter 2 [**Permutation Network of QC-LDPC Decoder**] presents a novel generic architecture of permutation network, which is the critical part of the reconfigurable QC-LDPC decoder.

The PCM in QC-LDPC codes is composed of several cyclic shift sub-matrices which specify the interconnection between the variable nodes and check nodes. Since modern wireless communication systems provide multiple code rates and code lengths in order to adapt various environments, the PCMs defined in these systems contain different sizes of sub-matrices. As a result, the permutation network in the reconfigurable LDPC decoder needs to provide cyclic shift ability for multiple input numbers (IN) and shift numbers (SN). Generally, logarithm barrel shifter is a natural approach for cyclic shift. However, it is hard to accommodate parallel permutation and various IN, especially the IN which is not power of 2. Some other approaches based on Benes network which are composed of several stages of cross-bar switch units, could provide multiple transmission paths. Dedicated look up table (LUT) or control signal generators are designed to control all the switch units and realize the desired permutation. However, the approaches based on the Benes network cannot avoid the latency problem not only because the number of stages is large but also the control signals are generated stage by stage.

This dissertation first proposes the method which enables barrel shifter based permutation network adapt the IN which is not power of 2. Then it puts forward the permutation network based on the Banyan Network with much less stages, complexity and higher parallelism. As a consequence, it presents the architecture of generic permutation network (GPN), which is capable of constructing required permutation network for any given application with efficient control signal generating algorithm and high parallelism. Compared with the design by C. H. Liu in ISCAS 2008, it reduces 26% hardware cost for one group permutation and 51% hardware cost for parallel permutation. For WLAN standard, it saves 40.4% hardware cost furthermore and improve 33.3%

timing performance.

Chapter 3 [**Bit-serial Layered Scheduling based QC-LDPC Decoder Architecture**] describes the architecture for TDMP algorithm based on the bit-serial layered scheduling.

Since the QC-LDPC codes defined in WiMAX are the most typical and complicated codes in all the wireless communication standards, which contain 114 modes from 6 code rates and 19 code lengths, the demonstrated implementation of the proposed architecture is designed according to WiMAX standard. Compared with the previous designs, this architecture changes the block by block scheduling of TDMP algorithm, which achieves higher parallelism. The key schemes of this architecture contain three aspects:

1) It is full parallel in each layer, which is named as full parallel layered decoding. All messages within one layer are calculated and updated simultaneously. In order to avoid the interconnection problem, the arithmetic units and permutation network are designed as 1-bit form, which make the messages be transferred and updated bit by bit. With 6-bit quantization, the number of clock cycles for each layer is 6, and each iteration needs 24 to 72 clock cycles for different code rates;

2) The parallelism is improved furthermore by dedicated PCM reordering and the two-layer concurrent processing for low code rates. The clock cycles are finally reduced to 24~48.

3) Due to the high parallelism, all the messages are stored in registers, not plenty of small and inefficient memory banks. The power increasing from using registers is eliminated by reducing operating frequency and clock gating. Moreover, the variable node (VN) messages and the a-posterior probability (APP) messages share the same storing cells, which save at least 22.2% memory bits than the previous works.

Based on these schemes, the fabricated QC-LDPC decoder ASIC for WiMAX system realized ultra-low power. It occupies 3.36 mm^2 in SMIC 65nm low leakage LVT CMOS, and achieves 1Gbps (1056Mbps) throughput at 1.2V, 110MHz and 10 iterations with the power 115mW and power efficiency 10.9pJ/bit/iteration. The power reduces 42.1% and the power efficiency reduces 63.6% in the normalized comparison with the state-of-art publication in VLSI Symposium 2010.

Chapter 4 [**Semi-layer Scheduling based QC-LDPC Decoder Architecture**] introduces another architecture for TDMP algorithm based on the semi-layer scheduling.

In this architecture, half blocks in one layer are processed concurrently, which costs at most 2 clock cycles to process one layer in the iterative decoding procedure. Compared with the clock cycles per iteration $K_q \times N_{layer}$ (K_q is the quantization bit number of message, which usually varies from 5 to 8) in bit-serial based architecture, this architecture only needs $2 \times N_{layer}$ for one iteration, which improve the parallelism furthermore. The implementation for WiMAX system realizes 8-16 clock cycles for each iteration. Compared with the state-of-art work, this design achieves up to 6.5x higher parallelism and 82.4% power reduction with only 1.4x hardware cost. The dedicated clock gating and power gating scheme guarantees even lower power, which indicates the energy/bit/iteration of this design is only 1/6 of the best of published work.

Chapter 5 [**Conclusion**] summaries the proposals and draws conclusion of this dissertation.

ACKNOWLEDGEMENT

First of all, I would like to express my sincere gratitude to my advisor, Professor Satoshi Goto, who is not only a teacher in Waseda University but also a mentor during my doctor course. I will regard him as the most important benefactor in my life.

I would also like to express my appreciation to my second advisor Professor Fumiaki Maehara, Professor Shinji Kimura, Professor Tomohisa Wada and Professor Borivoje Nikolic for the help and guidance to my research.

I also thank Dr. Dajiang Zhou, Dr. Xinjin, Mr. Zhixiang Chen, Mr. Xiongxin Zhao, Mr. Qian Xie, Ms. Ying Cui, Mr. Yichao Lu, Mr. Nanfan Qiu and Dr. Wen Ji for working together with me on the LDPC and other topics. I also express my thanks to all the members of Goto Lab. for their help. I am grateful to Professor Takeshi Ikenaga at Waseda University for the valuable knowledge I learned in his courses.

I should thank my parents Zhongde Peng and Jiangqiu Cao, my wife Muchen Li and my best friend Yimeng Zhang for their support over the years.

Finally, I aexpress my gratitude to Waseda University Ambient SoC Global COE Program of MEXT and CREST project of Japan Science and Technology Agency for the support to my research. I also thank Ms. Kozue Ohata and Mr. Yasuyuki Saito for their help.

CONTENT

FIGURE LIST

TABLE LIST

1 INTRODUCTION

1.1 LDPC codes in ECC

Low-Density Parity-Check (LDPC) codes were originally proposed in 1962 by Gallager [1]. Since the contemporary investigations in concatenated coding shaded LDPC codes and the hardware at that time could not support effective codec implementations, the paper of LDPC codes were left on the shelf for over thirty years. Until Mackay published his work [2] in 1996, LDPC codes began to be strongly promoted. As a hotspot in channel coding area, there are significant advantages of LDPC codes which are discussed in [3].

Firstly, both abstractly and practically, LDPC codes have been proved to be capable of closely approaching the channel capacity.

Secondly, LDPC codes have better performance than turbo codes which are also popular channel coding approaches in some cases, with iterative decoding algorithms which are easy to implement with parallel architecture.

Thirdly, LDPC codes of almost any rate and block length can be created simply by specifying the shape of the parity check matrix, and the flexibility in rate is obtained only through considerable design effort.

Lastly, on the commercial side, LDPC codes are not patent protected.

1.2 LDPC codes representation

1.2.1 Parity check matrix

We shall consider only binary LDPC codes for the sake of simplicity in this dissertation. A LDPC code is a linear block code given by the null space of an

$m \times n$ parity check matrix H that has a low density of 1s. If the column weight and row weight of the parity check matrix are both constant, the LDPC code is called regular LDPC code, otherwise it is irregular LDPC code.

The definition "low-density" is unavoidably vague and cannot be precisely quantified, although the density (ratio of number of 1s over total number of entries in parity check matrix) of 0.01 or lower can be called low-density. In fact, the density need only be sufficiently low to permit effective iterative decoding. This is the key innovation behind the invention of LDPC codes. As is well known, it is unpractical to realize optimum (e.g., maximum-likelihood) decoding for the general linear block code due to the vast complexity involved. The low-density property of LDPC codes accommodates iterative decoding, which typically has near maximum likelihood performance at error rates of interest for many applications.

In order to guarantee the iterative decoding performance, almost all LDPC code constructions impose the following additional structural property on H: no two rows (or columns) have more than one position in common that contains a nonzero element. This property is called the row–column constraint, or simply, the RC constraint.

1.2.2 Tanner graph

Tanner graph [4] is a bipartite graph, whose nodes may be separated into two types, with edges connecting only nodes of different types. The two types of nodes in a tanner graph are the variable nodes (or bit nodes) and the check nodes (or constraint nodes), which are denoted as VNs and CNs respectively. The tanner graph of a LDPC code is drawn according to the $m \times n$ parity check matrix H as follows: list the VNs and CNs as two lines of vertices, CN i is connected to VN j whenever element h_{ij} in H is a 1. Thus in a tanner graph there are m CNs, one for each check equation, and n VNs, one for each code bit. The degree of each node in tanner graph is the number of edges connected to this node. Figure 1.1 shows an example of the tanner graph representation. All the variable nodes and c_3 have degree one. c_1 and c_2 have degree 2.

$$\begin{array}{c|c} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} & \begin{array}{c} C_1 \\ C_2 \\ C_3 \end{array} \\ \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} & \end{array}$$

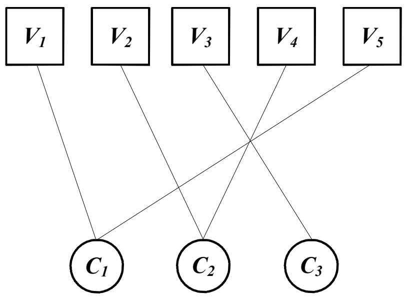


Figure 1.1 Example of tanner graph representation

According to the property of bipartite graph, the sum of check nodes degree always equal to the variable nodes degree, that is, the entry in the parity check matrix is restricted by both of row and column.

We denote that w_c and w_r is the column weight and row weight of the PCM respectively. For regular LDPC codes, w_c of all columns are the same, and so are the w_r of all rows. Since the number of 1s in the PCM is fixed,

$$w_c n = w_r m \tag{1.1}$$

The code rate r can be expressed as:

$$r = 1 - \frac{m}{n} = 1 - \frac{w_c}{w_r} \tag{1.2}$$

In irregular LDPC codes, w_c of all columns and w_r of all rows are different, which could be described by degree distribution.

Assuming the maximum degree of variable nodes is d_v, the degree distribution could be represented by index $\{\lambda_1, \lambda_2 \cdots\cdots \lambda_{d_v}\}$, where λ_i denotes the fraction of all edges connected to degree-i VNs. In the same way, the maximum degree of check nodes is d_c. The degree distribution of check nodes is $\{\rho_1, \rho_2 \cdots\cdots \rho_{d_c}\}$, where ρ_j denotes the fraction of all edges connected to degree-i CNs.

Then, the degree distribution functions are

$$\lambda(x) = \sum_{i=1}^{d_v} \lambda_i x^{i-1} \tag{1.3}$$

$$\rho(x) = \sum_{j=1}^{d_c} \rho_j x^{j-1} \tag{1.4}$$

Obviously, these functions satisfy

$$\lambda(1) = \rho(1) = 1 \tag{1.5}$$

If the total number of edges in the tanner graph is E, the number of i degree variable nodes is

$$v_i = \frac{E\lambda_i}{i} \tag{1.6}$$

Then,

$$n = \sum_{i=1}^{d_v} v_i = E\sum_{i=1}^{d_v} \frac{\lambda_i}{i} = E\int_0^1 \lambda(x)dx \tag{1.7}$$

The number of j degree check nodes is

$$c_j = \frac{E\rho_j}{j} \tag{1.8}$$

$$m = \sum_{j=1}^{d_c} c_j = E\sum_{j=1}^{d_c} \frac{\rho_j}{j} = E\int_0^1 \rho(x)dx \tag{1.9}$$

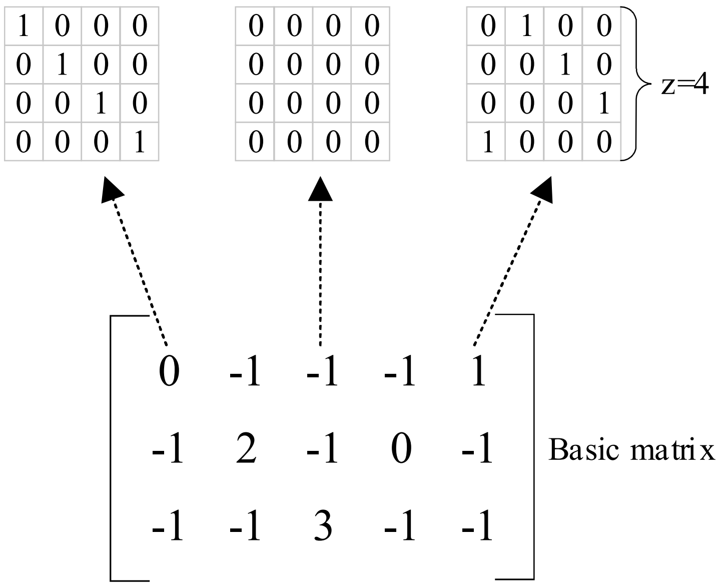
The derivation of $E\sum_{i=1}^{d_v} \frac{\lambda_i}{i} = E\int_0^1 \lambda(x)dx$ is

$$\begin{aligned} \int_0^1 \lambda(x)dx &= \int_0^1 \sum_{i=1}^{d_v} \lambda_i x^{i-1} dx = \sum_{i=1}^{d_v} \lambda_i \int_0^1 x^{i-1} dx \\ &= \sum_{i=1}^{d_v} \lambda_i \frac{1}{i} x^i \Big|_0^1 = \sum_{i=1}^{d_v} \frac{\lambda_i}{i} \end{aligned} \tag{1.10}$$

When the PCM is full rank, the code rate is

$$r = 1 - \frac{m}{n} = 1 - \frac{\sum_{i=1}^{d_v} \frac{\lambda_i}{i}}{\sum_{j=1}^{d_c} \frac{\rho_j}{j}} = 1 - \frac{\int_0^1 \lambda(x)dx}{\int_0^1 \rho(x)dx} \tag{1.11}$$

1.3 QC-LDPC codes in wireless standards



(a) Parity check matrix construction method of QC-LDPC

-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	27	-1	-1	-1	22	79	9	-1	-1	-1	12	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	24	22	81	-1	33	-1	-1	-1	0	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1
61	-1	47	-1	-1	-1	-1	-1	65	25	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1
-1	-1	39	-1	-1	-1	84	-1	-1	41	72	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	46	40	-1	82	-1	-1	-1	79	0	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1
-1	-1	95	53	-1	-1	-1	-1	-1	14	18	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1
-1	11	73	-1	-1	-1	2	-1	-1	47	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1
12	-1	-1	-1	83	24	-1	43	-1	-1	-1	51	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1
-1	-1	-1	-1	-1	94	-1	59	-1	-1	70	72	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1
-1	-1	7	65	-1	-1	-1	-1	39	49	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0
43	-1	-1	-1	-1	66	-1	41	-1	-1	-1	26	7	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0

(b) Basic matrix of 5/6 code rate defined in WiMAX

Figure 1.2 QC-LDPC codes in wireless standards

Since the ordinary LDPC codes encountered the complexity problem in hardware implementation, QC-LDPC code [5] was proposed for implementation friendly architecture with little performance loss. This property comes from the PCM construction procedure, in which the basic matrix is predefined and the PCM is expanded from the basic matrix. The expansion follows the following rules:

1) Each entry of the basic matrix is replaced with z×z sub-matrix, where z is called the expansion factor,

2) The entry “-1” is replaced by a zero sub-matrix

3) The entry “0” is replaced by an identity sub-matrix

4) The entry “k” (k>0) is replaced by a cyclically shifted identity sub-matrix, where k is the shift number

In the example shown in Figure 1.2 (a), the expansion factor is 4. In the wireless standards, the situations are much more complex. For example, WiMAX standard [6] defines 19 expansion factors and 6 basic matrices for 6 different code rates. Figure 1.2 (b) shows the basic matrix of code rate 1/2.

The irregular repeat accumulate LDPC (IRA-LDPC) codes have similar expansion property which are adopted by DVB-S2 and ISDB-S2 standards. The difference between IRA-LDPC and QC-LDPC codes relies on two aspects:

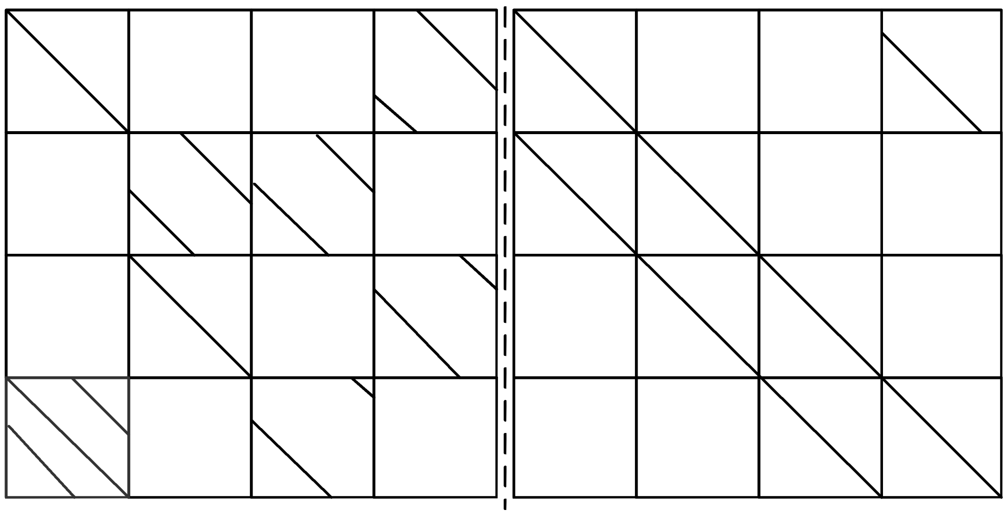


Figure 1.3 QC-form of PCM in IRA-LDPC codes

1) As shown in Figure 1.3, there exists some exception blocks in the

QC-form of the PCM in IRA-LDPC codes, which are discussed in Wen's doctoral dissertation [6].

2) The IRA-LDPC codes in DVB-S2 and ISDB-S2 contain long code lengths which are above ten thousand. The code lengths in QC-LDPC codes of wireless communication standards are varying from several hundred to several thousand.

1.4 LDPC decoding algorithm

1.4.1 Bit flipping algorithm

Bit flipping algorithm is originally proposed by Gallager, which is a natural approach of decoding. The procedure of this algorithm is shown in Figure 1.4.

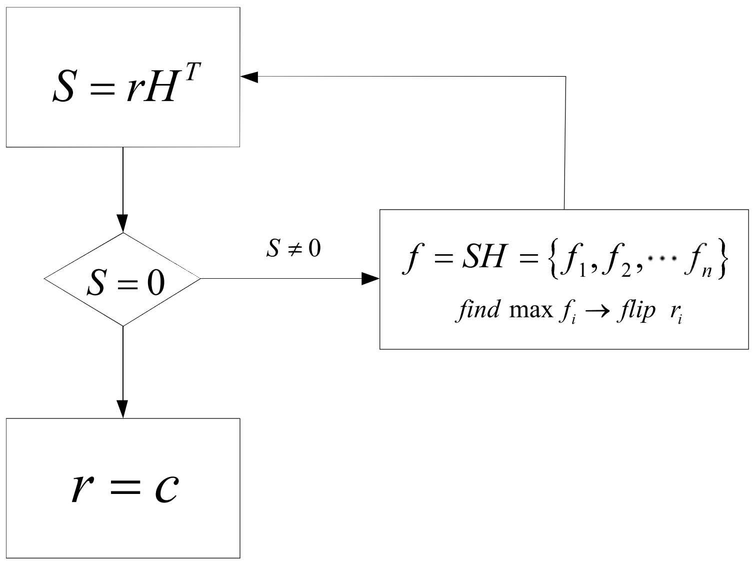

$S = rH^T$
$S = 0$
$S \neq 0$
$f = SH = \{f_1, f_2, \cdots f_n\}$
$find \max f_i \rightarrow flip\ r_i$
$r = c$

Figure 1.4 Bit flipping decoding

Assuming the original codword is $c = \{c_1, c_2, \cdots c_n\}$, and the received codeword is $r = \{r_1, r_2, \cdots r_n\}$, compute the syndrome S as

$$S = rH^T = r(c_1, c_2, \cdots c_m) = (rc_1, rc_2, \cdots rc_m) \neq 0 \tag{1.12}$$

It means that the current received codeword contains some bits which are different from the original one. That is some parity check equations are not satisfied. Among the unsatisfied check equations, the number of the *ith* bit participate in these check equations can be courted by

$$f = sH = (rc_1, rc_2, \cdots rc_m)(b_1, b_2, \cdots b_n) = \{f_1, f_2, \cdots f_n\} \tag{1.13}$$

The bit flipping decoding is to find the maximum f_i and flip the corresponding bit. The decoding procedure is an iterative process between calculating the syndrome and flipping the most possible error bit until all the check equations are satisfied.

1.4.2 General sum-product algorithm

The general sum-product algorithm is widely used in many areas [8]. It is assumed that the function could be factorized into the product of some sub functions.

$$g(x_1, x_2, \cdots, x_m) = \prod_{j \in J} f_j(x_j) \tag{1.14}$$

J is the discrete index set, $J = \{1, 2, \ldots, m\}$, that is the variables of the sub functions are the sub set of original variables set.

$$x_j \subseteq \{x_1, x_2, \cdots, x_m\} \tag{1.15}$$

Factor graph is used to represent this factorization, which contains variable nodes and function nodes. When the function contains the variable, there is an edge between the two corresponding nodes. An example is given as follows.

$$g(x_1, x_2, x_3, x_4, x_5) = f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3) f_D(x_3, x_4) f_E(x_3, x_5) \tag{1.16}$$

The factor graph of $g(x_1, x_2, x_3, x_4, x_5)$ is shown in Figure 1.5.

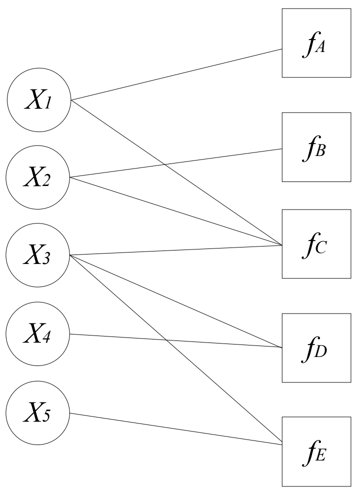


Figure 1.5 Example of factor graph

Through summation operation, (1.16) can be represent as single variable form, which is

$$g(x_1,x_2,x_3,x_4,x_5)=$$
$$f_A(x_1)\left(\sum_{x_2} f_B(x_2)\left(\sum_{x_3} f_C(x_1,x_2,x_3)\left(\sum_{x_4} f_D(x_3,x_4)\right)\left(\sum_{x_5} f_E(x_3,x_5)\right)\right)\right) \quad (1.17)$$

In the factor graph, this transformation is the process of forming a tree, which is shown in Figure 1.6.

Most of multiple variable functions with independent variables could be represented as this single variable form. Since the process contains addition and multiplication operations, the algorithm involves this process is called as general sum product algorithm. When the function is the probability function of code, it becomes an iterative decoding process.

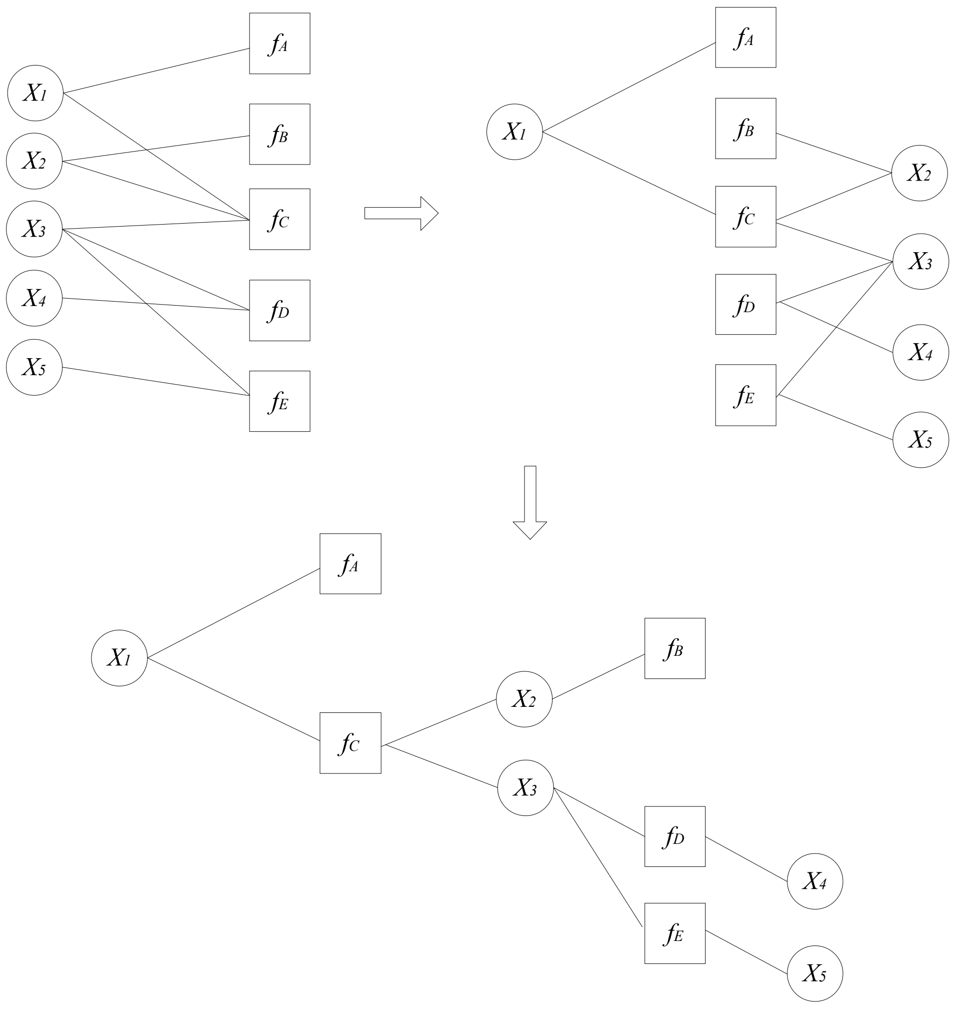


Figure 1.6 Tree structure of factor graph

As Figure 1.6 shows, in the general sum product algorithm, the information from variable node *x* to the function node *f* is the multiplication of all information sent by linked function node except *f*.

$$\mu_{x\to f} = \prod_{h\in n(x)\backslash f} \mu_{h\to x} \tag{1.18}$$

$n(x)$ is the neighborhood set of variable node *x*.

The information from function node to variable node is weighted summation of message from other variables.

$$\mu_{f\to x} = \sum_{-x} f(X) \prod_{y\in n(x)\backslash x} \mu_{y\to f} \tag{1.19}$$

This weighted summation eliminate other variables except x, that is in the tree structure whose root is x, the updating message is only single variable function.

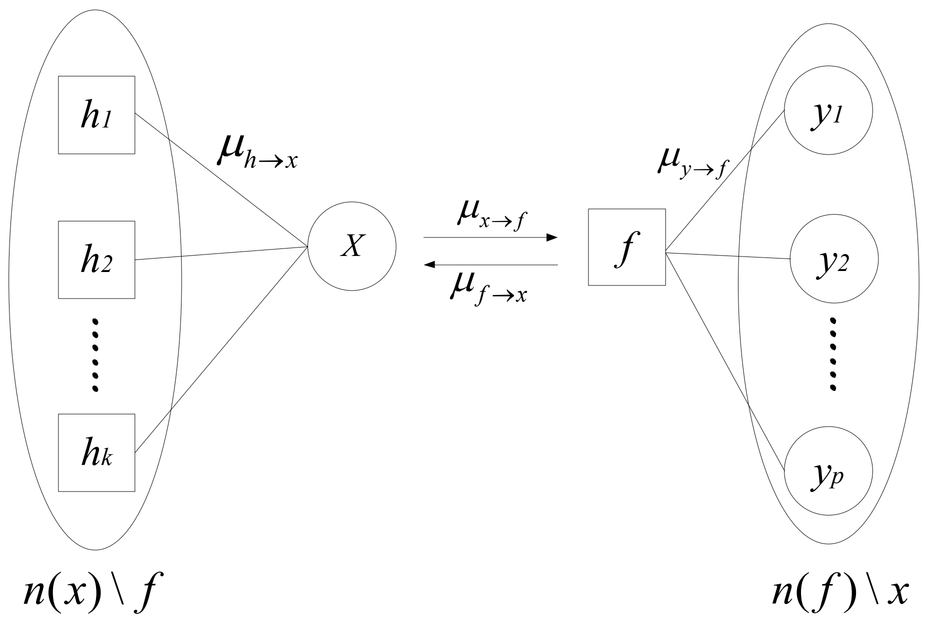


Figure 1.6 General sum product algorithm

1.4.3 Belief propagation algorithm

Based on the general sum product algorithm, the well-known belief propagation algorithm or message passing algorithm is derived as following process.

The target of decoding algorithm is to find

Max P(c$_i$ | *r*, all c_i participates check equations are satisfied) (1.20)

P is the conditional probability of the bit c_i in case of received codeword *r* and all the check equations which c_i takes part in are satisfied.

In order to calculate the conditional probability, we denote

$$q_n(x) = P(c_n = x \mid r, z_m = 0, m \in M_n) \tag{1.21}$$

$$q_{mn}(x) = P(c_n = x \mid r, z_m = 0, m \in M_{m\ n}) \tag{1.22}$$

$$r_{mn}(x) = P(z_m = 0 \mid c_n = x, r) \tag{1.23}$$

Zm is the parity check equation, *Mn* is the set of index that codeword bit *cn* participate in, *Mm,n* is the sub set of *Mn* which exclude the *mth* check equation, *rmn* is the conditional probability that the check equation is satisfied in case of received codeword *r* and *cn* is equal to *x*.

Since

$$P(A|B,C)=\frac{P(A,B,C)}{P(B,C)}=\frac{P(A,B|C)P(C)}{P(B|C)P(C)}=\frac{P(A,B|C)}{P(B|C)} \qquad (1.24)$$

$$q_n(x)=P(c_n=x|r,z_m=0,m\in M_n)$$

$$=\frac{P(c_n=x,z_m=0,m\in M_n|r)}{P(z_m=0,m\in M_n|r)}$$

$$=\frac{P(c_n=x|r)P(z_m=0,m\in M_n|c_n=x,r)}{P(z_m=0,m\in M_n|r)}$$

In case of independent variables, only *rn* is involved.

$$=\frac{P(c_n=x|r_n)P(z_m=0,m\in M_n|c_n=x,r)}{P(z_m=0,m\in M_n|r)}$$

$$=\frac{P(c_n=x|r_n)}{P(z_m=0,m\in M_n|r)}\prod_{m\in M_n}P(z_m=0|c_n=x,r)$$

$P(z_m=0,m\in M_n|r)$ is independent of x

$$=\alpha P(c_n=x|r_n)\prod_{m\in M_n}P(z_m=0|c_n=x,r)$$

$$=\alpha P(c_n=x|r_n)\prod_{m\in M_n}r_{mn}(x) \qquad (1.25)$$

$\prod_{m\in M_n}P(z_m=0|c_n=x,r)$ is names as extrinsic probability, which comes from check equations. $P(c_n=x|r_n)$ is called intrinsic probability, which is from channel estimation.

In the same way,

$$q_{mn}(x)=\alpha P(c_n=x|r_n)\prod_{m\in M_{n,m}}r_{mn}(x) \qquad (1.26)$$

On the other hand,

$$r_{mn}(x) = P(z_m = 0 \mid c_n = x, r)$$

Considering all the

$$= \sum_{x'_n, n' \in N_{m,n}} P(z_m = 0, \{c'_n = x'_n, n' \in N_{m,n}\} \mid c_n = x, r)$$

$$= \sum_{x'_n, n' \in N_{m,n}} [P(z_m = 0 \mid \{c'_n = x'_n, n' \in N_{m,n}\}, c_n = x, r)$$
$$\times P(\{c'_n = x'_n, n' \in N_{m,n}\} \mid c_n = x, r)]$$

In case of independent variables,

$$= \sum_{x'_n, n' \in N_{m,n}} [P(z_m = 0 \mid \{c'_n = x'_n, n' \in N_{m,n}\}, c_n = x, r))$$
$$\times P(\{c'_n = x'_n, n' \in N_{m,n}\} \mid r)]$$

$$= \sum_{x'_n, n' \in N_{m,n}} [P(z_m = 0 \mid \{c'_n = x'_n, n' \in N_{m,n}\}, c_n = x, r))$$
$$\times \prod_{x_i \in N_{m,n}} P(c_i = x_i \mid r)]$$

$$= \sum_{x'_n, n' \in N_{m,n}} [P(z_m = 0 \mid \{c'_n = x'_n, n' \in N_{m,n}\}, c_n = x))$$
$$\times \prod_{x_i \in N_{m,n}} P(c_i = x_i \mid r)]$$

$P(z_m = 0 \mid \{c'_n = x'_n, n' \in N_{m,n}\}, c_n = x)$ is either 0 or 1, it just needs to keep the 1 situation, which is $\sum_{n' \in N_m} x_{n'} = 0$, or $x_n = \sum_{n' \in N_{m,n}} x_{n'}$

$$= \sum_{x'_n, n' \in N_{m,n} \cap x = \sum x_l} \prod_{x_i \in N_{m,n}} P(c_i = x_i \mid r) \tag{1.27}$$

Define the modulo-2 sum of the first *k* participating nodes,

$$\zeta(l) = \sum_{i=1}^{l} x_{N_{m,n}(i)} = \sum x_l \tag{1.28}$$

$P(\zeta(l) = 1)$ and $P(\zeta(l) = 1)$ can be calculated by recursive process

If the $P(\zeta(1))$ is known,

$$P(\zeta(2) = 0) = P(\zeta(1) = 0) P(x_{N_{m,n}(2)} = 0) + P(\zeta(1) = 1) P(x_{N_{m,n}(2)} = 1)$$

$$P(\zeta(2)=1)=P(\zeta(1)=1)P(x_{N_{m,n}(2)}=0)+P(\zeta(1)=1)P(x_{N_{m,n}(2)}=0)$$

In the same way, denote $w_k(x)=P(\zeta(k)=x)$, then

$$\begin{aligned} w_k(0) &= w_{k-1}(0)P(x_{N_{m,n}(k)}=0)+w_{k-1}(1)P(x_{N_{m,n}(k)}=1) \\ &= w_{k-1}(0)q_{m,N_{m,n}(k)}(0)+w_{k-1}(1)q_{m,N_{m,n}(k)}(1) \end{aligned} \tag{1.29}$$

$$\begin{aligned} w_k(1) &= w_{k-1}(0)P(x_{N_{m,n}(k)}=1)+w_{k-1}(1)P(x_{N_{m,n}(k)}=0) \\ &= w_{k-1}(0)q_{m,N_{m,n}(k)}(1)+w_{k-1}(1)q_{m,N_{m,n}(k)}(0) \end{aligned} \tag{1.30}$$

Compare with (1.27), we get

$$r_{mn}(0)=w_L(0) \tag{1.31}$$

$$r_{mn}(1)=w_L(1) \tag{1.32}$$

(1.29)-(1.30),

$$w_k(0)-w_k(1)=[w_{k-1}(0)-w_{k-1}(1)][q_{m,N_{m,n}(k)}(0)-q_{m,N_{m,n}(k)}(0)] \tag{1.33}$$

Based on the recursive property,

$$w_k(0)-w_k(1)=\prod_{i=1}^{k}[q_{m,N_{m,n}(i)}(0)-q_{m,N_{m,n}(i)}(1)] \tag{1.34}$$

Denote

$$\delta r_{mn}=r_{mn}(0)-r_{mn}(1) \tag{1.35}$$

$$\delta q_{ml}=q_{ml'}(0)-q_{ml'}(1) \tag{1.36}$$

Then

$$\delta r_{mn}=\prod_{i=1}^{k}\delta q_{mi} \tag{1.37}$$

Since $r_{mn}(0)+r_{mn}(1)=1$

$$r_{mn}(0)=\frac{1+\delta r_{mn}}{2} \tag{1.38}$$

$$r_{mn}(1)=\frac{1-\delta r_{mn}}{2} \tag{1.39}$$

In summary, the belief propagation algorithm is listed as follows:

Input:

Channel posterior probabilities $p_n(x) = P(c_n = x \mid r_n)$

Maximum iteration times $N_{\max}$

Initialization:

Set all $q_{mn}(x) = p_n(x)$ for all $H(m,n) = 1$

Iteration:

Horizontal Step:

In the row sequence

Compute $\delta q_{ml} = q_{ml}(0) - q_{ml}(1)$

For all $H(m,n) = 1$ in this row, compute $\delta r_{mn} = \prod_{i=1}^{k} \delta q_{ml'}$

Compute $r_{mn}(0) = \dfrac{1 + \delta r_{mn}}{2}$

Compute $r_{mn}(1) = \dfrac{1 - \delta r_{mn}}{2}$

Vertical Step:

In the column sequence

Compute

$$q_{mn}(0) = \alpha_{mn} P(c_n = 0 \mid r_n) \prod_{m \in M_{n,m}} r_{mn}(0) = \alpha_{mn} P_n(0) \prod_{m \in M_{n,m}} r_{mn}(0)$$

Compute

$$q_{mn}(1) = \alpha_{mn} P(c_n = 1 \mid r_n) \prod_{m \in M_{n,m}} r_{mn}(1) = \alpha_{mn} P_n(1) \prod_{m \in M_{n,m}} r_{mn}(1)$$

α_{mn} is set to satisfy that $q_{mn}(0) + q_{mn}(1) = 1$

Compute $\alpha_{mn} = \dfrac{1}{P_n(0) \prod_{m \in M_{n,m}} r_{mn}(0) + P_n(1) \prod_{m \in M_{n,m}} r_{mn}(1)}$

Decision:

After the vertical step

Compute $q_n(0) = \alpha_n P_n(0) \prod_{m \in M_n} r_{mn}(0)$

Compute $q_n(1) = \alpha_n P_n(1) \prod_{m \in M_n} r_{mn}(1)$

α_n is set to satisfy that $q_n(0) + q_n(1) = 1$

Compute $\alpha_n = \dfrac{1}{P_n(0) \prod_{m \in M_n} r_{mn}(0) + P_n(1) \prod_{m \in M_n} r_{mn}(1)}$

Compute $\begin{cases} q_n(0) > 0.5 \to \hat{c}_n = 0 \\ q_n(1) > 0.5 \to \hat{c}_n = 1 \end{cases}$

Iteration times $N = N + 1$

If $cH^T = 0$ or $N > N_{\max}$, STOP

Else go to **Iteration**

Through introducing the expression of ***LLR*** (Log-Likelihood Ratio) [9], computation complexity could be greatly reduced.

We denote that

$$LLR(f) = \ln \frac{P(f=0)}{P(f=1)}. \tag{1.40}$$

Lemma: The probability of even 1s in a binary sequence $\{c_i\}$ is

$$\frac{1}{2} + \frac{1}{2} \prod_{i=1}^{n} [1 - 2P(c_i = 1)] \tag{1.41}$$

In check equation, this probability is equal to the probability of that check equation is satisfied to 0.

That is

$$r_{mn}(0) = \frac{1}{2} + \frac{1}{2} \prod_{l=1}^{n} [1 - 2q_{ml}(1)] \tag{1.42}$$

Then

$$r_{mn}(1) = 1 - r_{mn}(0) = \frac{1}{2} - \frac{1}{2}\prod_{l=1}^{n}[1 - 2q_{ml}(1)] \tag{1.43}$$

Introduce the hyperbolic tangent function $\tanh(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$,

$r_{mn}(1) + r_{mn}(0) = 1$

$$1 - 2r_{mn}(1) = r_{mn}(0) - r_{mn}(1) = \frac{r_{mn}(0) - r_{mn}(1)}{r_{mn}(0) + r_{mn}(1)} = \tanh(\frac{1}{2}\ln\frac{r_{mn}(0)}{r_{mn}(1)}) \tag{1.44}$$

In the same way,

$q_{ml}(1) + q_{ml}(0) = 1$

$$1 - 2q_{ml}(1) = \tanh(\frac{1}{2}\ln\frac{q_{ml}(0)}{q_{ml}(1)}) \tag{1.45}$$

(1.42)-(1.43),

$$1 - 2r_{mn}(1) = \prod_{l=1}^{n}[1 - 2q_{ml}(1)] \tag{1.46}$$

Based on (1.44), (1.45) and (1.46),

$$\tanh(\frac{1}{2}\ln\frac{r_{mn}(0)}{r_{mn}(1)}) = \prod_{l=1}^{n}\tanh(\frac{1}{2}\ln\frac{q_{ml}(0)}{q_{ml}(1)})$$

$$\tanh[\frac{1}{2}LLR(r_{mn})] = \prod_{l=1}^{n}\tanh[\frac{1}{2}LLR(q_{ml})]$$

$$LLR(r_{mn}) = 2\tanh^{-1}(\prod_{l=1}^{n}\tanh[\frac{1}{2}LLR(q_{ml})]) \tag{1.47}$$

For the q_{mn} calculation,

$$q_{mn}(0) = \alpha_{mn}P(c_n = 0 \mid r_n)\prod_{m \in M_{n,m}} r_{mn}(0) = \alpha_{mn}P_n(0)\prod_{m \in M_{n,m}} r_{mn}(0) \tag{1.48}$$

$$q_{mn}(1) = \alpha_{mn}P(c_n = 1 \mid r_n)\prod_{m \in M_{n,m}} r_{mn}(1) = \alpha_{mn}P_n(1)\prod_{m \in M_{n,m}} r_{mn}(1) \tag{1.49}$$

(1.48)/(1.49)

$$\frac{q_{mn}(0)}{q_{mn}(1)}=\frac{P_n(0)\prod_{m\in M_{n,m}} r_{mn}(0)}{P_n(1)\prod_{m\in M_{n,m}} r_{mn}(1)}=\frac{P_n(0)}{P_n(1)}\prod_{m\in M_{n,m}}\frac{r_{mn}(0)}{r_{mn}(1)}$$

$$LLR(q_{mn})=LLR(P_n)+\sum_{m\in M_{n,m}} LLR(r_{mn}) \tag{1.50}$$

Thus, the *LLR* based belief propagation algorithm is

Input:

Channel posterior probabilities $p_n(x)=P(c_n=x\,|\,r_n)$

Maximum iteration times $N_{\max}$

Initialization:

Set all $LLR(q_{mn})=\ln\frac{p_n(0)}{p_n(1)}$ for all the *n* variable nodes

Iteration:

Horizontal Step:

In the row sequence

Compute $LLR(r_{mn})=2\tanh^{-1}(\prod_{l=1}^{n}\tanh[\frac{1}{2}LLR(q_{ml})])$

Vertical Step:

In the column sequence

Compute $LLR(q_{mn})=LLR(P_n)+\sum_{m\in M_{n,m}} LLR(r_{mn})$

Decision:

After the vertical step

Compute $LLR(q_n)=LLR(P_n)+\sum_{m\in M_n} LLR(r_{mn})$

Compute $LLR(q_n)\begin{cases}>0\rightarrow\hat{c}_n=0\\<0\rightarrow\hat{c}_n=1\end{cases}$

Iteration times $N=N+1$

If $cH^T=0$ or $N>N_{\max}$, STOP

Else go to **Iteration**

Generally the channel posterior probabilities are provided by channel estimation. In the AWGN channel and BPSK modulation with equal probability resource assumption, they can be expressed as

$$P_n(0) = P_r(c_n = 1 \mid r_n) = P_r(x_n = -1 \mid r_n) = \frac{1}{1+e^{\frac{2r_n}{\sigma^2}}} \tag{1.51}$$

$$P_n(1) = P_r(c_n = 0 \mid r_n) = P_r(x_n = 1 \mid r_n) = \frac{1}{1+e^{\frac{-2r_n}{\sigma^2}}} \tag{1.52}$$

Proof: According to the property of AWGN channel,

$$P(r_n \mid x_n = x) = \frac{1}{\sqrt{\pi n_0}} e^{\frac{-(r_n-x)^2}{n_0}}$$

Induce the Bayes theorem,

$$P(B \mid A) = \frac{P(AB)}{P(A)} = \frac{P(A \mid B)P(B)}{P(A \mid B)P(B) + P(A \mid \overline{B})P(\overline{B})}$$

$$\begin{aligned}
P_n(x) &= P_r(c_n = x \mid r_n) = P_r(x_n = x \mid r_n) \\
&= \frac{P_r(r_n \mid x_n = x)P(x_n = x)}{P_r(r_n \mid x_n = x)P(x_n = x) + P_r(r_n \mid x_n = \overline{x})P(x_n = \overline{x})} \\
&= \frac{\frac{1}{2}\frac{1}{\sqrt{\pi n_0}} e^{\frac{-(r_n-x)^2}{n_0}}}{\frac{1}{2}\frac{1}{\sqrt{\pi n_0}} e^{\frac{-(r_n-x)^2}{n_0}} + \frac{1}{2}\frac{1}{\sqrt{\pi n_0}} e^{\frac{-(r_n+x)^2}{n_0}}} \\
&= \frac{e^{\frac{-(r_n-x)^2}{n_0}}}{e^{\frac{-(r_n-x)^2}{n_0}} + e^{\frac{-(r_n+x)^2}{n_0}}} = \frac{e^{\frac{-(r_n-x)^2}{n_0}} e^{\frac{(r_n-x)^2}{n_0}}}{e^{\frac{-(r_n-x)^2}{n_0}} e^{\frac{(r_n-x)^2}{n_0}} + e^{\frac{-(r_n+x)^2}{n_0}} e^{\frac{(r_n-x)^2}{n_0}}} \\
&= \frac{1}{1+e^{\frac{-4r_n x}{n_0}}} \left(\frac{n_0}{2} = \sigma^2\right) \\
&= \frac{1}{1+e^{\frac{-2r_n x}{\sigma^2}}} (x = \pm 1)
\end{aligned}$$

1.4.4 Min-sum, normalize and offset simplification

To simplify the decoding algorithm for hardware design, the min-sum, normalize and offset simplification are induced to the *LLR* based algorithm.

Firstly, min-sum simplification [10] is used to simplify the horizontal step.

Lemma: $2\tanh^{-1}(\tanh(\frac{x}{2})\tanh(\frac{y}{2})) = \ln\frac{1+e^{x+y}}{e^x+e^y}$

Proof: According to the definition of hyperbolic tangent function

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$\tanh(\frac{x}{2}) = \frac{e^x - 1}{e^x + 1}$$

Then

$$\tanh(\frac{x}{2})\tanh(\frac{y}{2}) = \frac{(e^x-1)(e^y-1)}{(e^x+1)(e^y+1)}$$

$$\tanh(\frac{1}{2}\ln\frac{1+e^{x+y}}{e^x+e^y}) = \frac{\frac{1+e^{x+y}}{e^x+e^y}-1}{\frac{1+e^{x+y}}{e^x+e^y}+1} = \frac{1+e^{x+y}-e^x-e^y}{1+e^{x+y}+e^x+e^y} = \frac{(e^x-1)(e^y-1)}{(e^x+1)(e^y+1)}$$

Using the Jacobian logarithm, the simplest situation in (1.47) is

$$\begin{aligned}&2\tanh^{-1}(\tanh[\frac{1}{2}LLR(q_{m1})]\tanh[\frac{1}{2}LLR(q_{m2})])\\&=\mathrm{sign}(LLR(q_{m1}))\mathrm{sign}(LLR(q_{m2}))\min(|LLR(q_{m1})|,|LLR(q_{m2})|)\\&\quad+\log(1+e^{-|LLR(q_{m1})+LLR(q_{m2})|})-\log(1+e^{-LLR(q_{m1})-LLR(q_{m2})})\end{aligned}$$

Drop the last two items,

$$\begin{aligned}&2\tanh^{-1}(\tanh[\frac{1}{2}LLR(q_{m1})]\tanh[\frac{1}{2}LLR(q_{m2})])\\&=\mathrm{sign}(LLR(q_{m1}))\mathrm{sign}(LLR(q_{m2}))\min(|LLR(q_{m1})|,|LLR(q_{m2})|)\end{aligned} \tag{1.53}$$

This simplification could be extended to numerous variables, then

$$LLR(r_{mn}) = \prod_{l=1}^{n} sign(LLR(q_{ml}))\cdot\min_{l\in N_{m,n}}|LLR(q_{ml})| \tag{1.54}$$

Since min-sum simplification would bring some performance loss, the normalized and offset modifications are used to improve the performance.

The so-called normalized method [11] is multiply a normalize factor in (1.54), which would make the calculation of *LLR(r_{mn})* much accuracy.

$$LLR(r_{mn}) = k \cdot \prod_{l=1}^{n} sign(LLR(q_{ml})) \cdot \min_{l \in N_{m,n}} |LLR(q_{ml})| \tag{1.55}$$

The offset approach is make a offset subtraction of the minimum value [12] in (1.54)

$$LLR(r_{mn}) = \begin{cases} \prod_{l=1}^{n} sign(LLR(q_{ml})) \cdot (\min_{l \in N_{m,n}} |LLR(q_{ml})| - V_{offset}) & \min |LLR(q_{ml})| > V_{th} \\ \prod_{l=1}^{n} sign(LLR(q_{ml})) \cdot \min_{l \in N_{m,n}} |LLR(q_{ml})| & \min |LLR(q_{ml})| \le V_{th} \end{cases} \tag{1.56}$$

All the constant in these two methods are depend on the code property. In the implementation, they should be selected for hardware friendly value.

1.4.5 Layered scheduling

The so-called layered decoding algorithm or TDMP algorithm [13] is proved to be an efficient decoding algorithm for QC-LDPC codes. The basic concept of this algorithm is regarding every z rows expanded from the same row of the basic matrix as one layer and updating the LLR form messages layer by layer. This algorithm achieves about 2X throughput than TPMP algorithm since it increases the convergence speed. The combined offset min-sum TDMP algorithm is usually implemented in the hardware, which is describes as follows:

Input:

Parity-check matrix H

Channel posterior probabilities δ

```
Initialization:
    Variable node (VN) message matrix $\Lambda=0$
    A-posterior probability (APP) message $L=\delta$
    Check node (CN) message matrix $\lambda=0$
Iteration:
    For each iteration $t$
        For each layer $k$
            For each non-zero entry $H_{mn}=1$
                $\lambda_{mn} \leftarrow L_n - \Lambda_{mn}^{t-1}$    (1.57)
                $\Lambda_{mn}^{t} \leftarrow \max(\min_{N(m)\backslash n} |\lambda_{mn}| - \beta, 0) \prod_{N(m)\backslash n} sign(\lambda_{mn})$    (1.58)
                $L_n \leftarrow \lambda_{mn} + \Lambda_{mn}^{t}$    (1.59)
    $x_n = \begin{cases} 0 & if\ L_n > 0 \\ 1 & if\ L_n \leq 0 \end{cases}$
    If $cH^T = 0$ or Iteration times > threshold, iteration stops
```

In the above description of layered scheduling, *N(m)\n* denotes the neighboring variable nodes for check node *m* excluding variable node *n*. *β* is a constant, which is different for different wireless standards. According to (1.57), (1.58) and (1.59), the arithmetical computations in each iteration of layered scheduling include subtraction, minimum value search and addition. Since the computation should exclude current variable node, the actual minimum search should find the first and second minimum value in the involved variable node messages. The iteration will stop when the decision code words x_n are satisfied to all the parity check equation or the iteration number exceeds a predefined maximum number.

1.5 LDPC decoder

The general LDPC decoder architecture in the designs and implementations of [14]-[37] are shown in Figure 1.7.

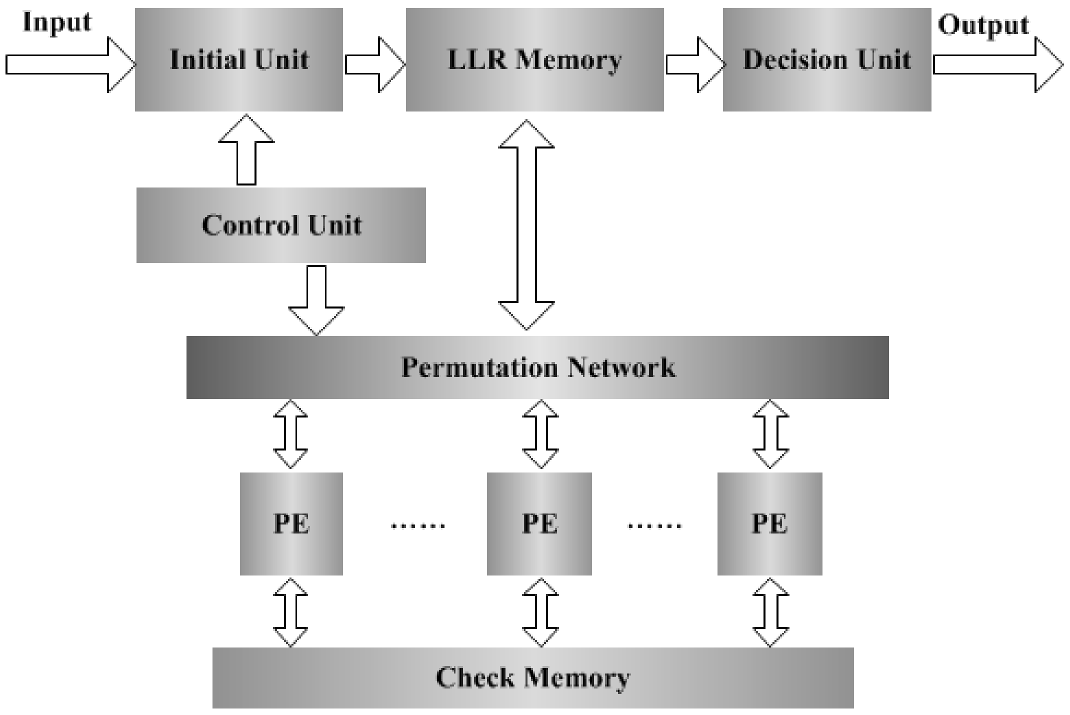


Figure 1.7 General architecture of LDPC decoder

In the above architecture, during the iterative decoding procedure, the permutation network transfers the log-likelihood ratio (LLR) information from LLR memory to the correct process engine (PE) to calculate the check information. Therefore, the reconfigurable capability of decoder is determined by the permutation network. The permutation network is the critical part of LDPC decoder for multiple code rate and code length.

2 Permutation Network of QC-LDPC Decoder

In QC-LDPC codes, the parity check matrix (PCM) is composed of numerous cyclic shift sub-matrices which specify the interconnection between the check nodes (CNs) and variable nodes (VNs).

Since wireless communication systems should provide accommodation for multiple code rates and code lengths in order to adapt various environments, the LDPC codes used in these systems contain several different sizes of sub-matrices. For example, the WiMAX LDPC codes specify 19 kinds of sub-matrices, in which the length vary from 24 to 96. Thus, the permutation network between CNs and VNs need to be dynamically reconfigurable.

Currently, the reconfigurable permutation networks designed for multiple lengths QC-LDPC codes are mainly based on barrel shifter or the well-known Benes network [38]. Generally, barrel shifter is a natural approach for performing cyclic shift, but it would face difficulty for various input number (IN) and shift number (SN). C. H. Liu et al [39] add a front-end stage and a back-end stage to barrel shifter, which make the whole network support the IEEE 802.16e and 802.11n. The shortcoming is that an additional sub-network should be used to choose the data fed into the last stage. Hardware cost would become much larger for more IN and SN.

On the other hand, the N×N Benes network is composed of $2\log_2 N-1$ stages of 2×2 switch units. To properly control all the switch units and realize the desired permutation, normally the dedicated control signals can be stored in the memory [40]. However, the area consumption would be too large to be implemented for various IN and SN. In order to avoid big memory, the later works all used control signal generating circuits to replace the memory based controller. J. Tang et al [41] proposed the scheme consists of two Benes networks, one of which is used to execute the permutation while another is brought to generate the control signals. Although this approach can significantly reduce the hardware complexity compared to the direct implementation using multiplexers, it still can be optimized to reduce the area cost and latency. D. Oh et al [42] and J. Lin et al [43] stated their own control signal generating algorithm for Benes network respectively, which can be easily implemented using regular decoders. For WiMAX, [42] reduced the hardware complexity by using the 3×3 network to some of the 4×4 network in the original Benes network because the length 96 is not the power of 2 but contains the multiplication factor 3, while [43] optimized the Benes network by cutting off the unused switches, replacing the unchanged switches with wires and combining some of special switches.

Nonetheless, those permutation networks based on the Benes network would always face the latency problem not only because the number of stages is large but also the control signals are generated stage by stage. In this paper, we propose a novel reconfigurable permutation network based on the Banyan network. The proposed network has only $\log_2 N$ stages, which is nearly half of the Benes network. Moreover, the control signal of all stages is not related to the previous stages, but simply depended on the initial SN of input, which would greatly reduce the latency of the permutation network.

2.1 Previous permutation network

2.1.1 Logarithm barrel shifter based permutation network

Barrel shifter is the most widely used structure for cyclical permutation. It is typically realized with a series of multiplexers with several stages. The connections between stages depend on the shift number. Because of the property of circular shift, the barrel shifter should provide the capability of shifting the inputs up to n-1 positions if the total input number is n. As a basic digital circuit element, the barrel shifter is widely used in many applications, one of which is that it is used to realize the bit shift instruction in microprocessors.

Figure 2.1 shows a typical structure of 4 bit right shift barrel shifter.

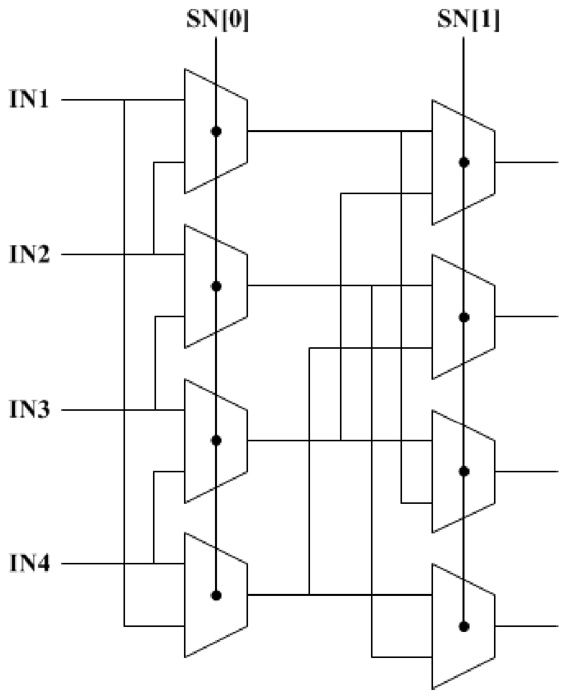


Figure 2.1 4 bit right shift barrel shifter

As Figure 2.1 shows, the number of multiplexers required for an n-bit word is $n\log_2 n$, and the stage of multiplexers is $\log_2 n$, where the input number (IN) n is required to be power of 2. From this point of view, the typical barrel shifter is also named as Logarithm Barrel Shitter (LBA). The control signals within each stage are the same, which are the bits of shift number (SN) from MSB to LSB.

The shortcoming of LBA mainly relies on two aspects when it is applied to QC-LDPC codes:

1) LBA cannot support the input number which is not power of 2.

2) LBA cannot support various input number when the maximum input

number is fixed.

2.1.2 Benes network based permutation network

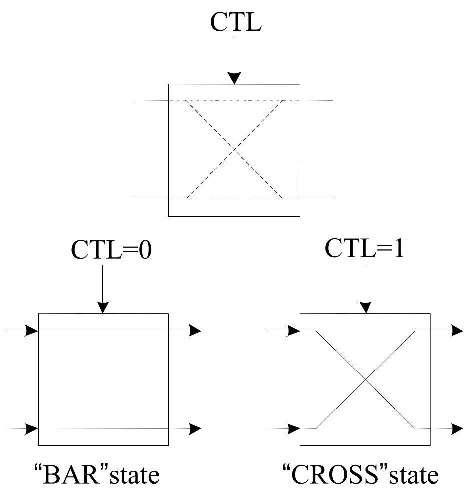
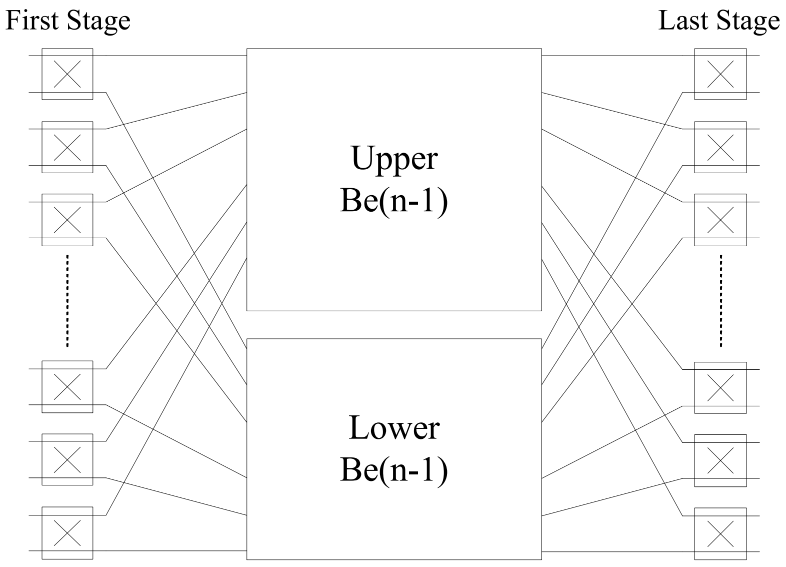


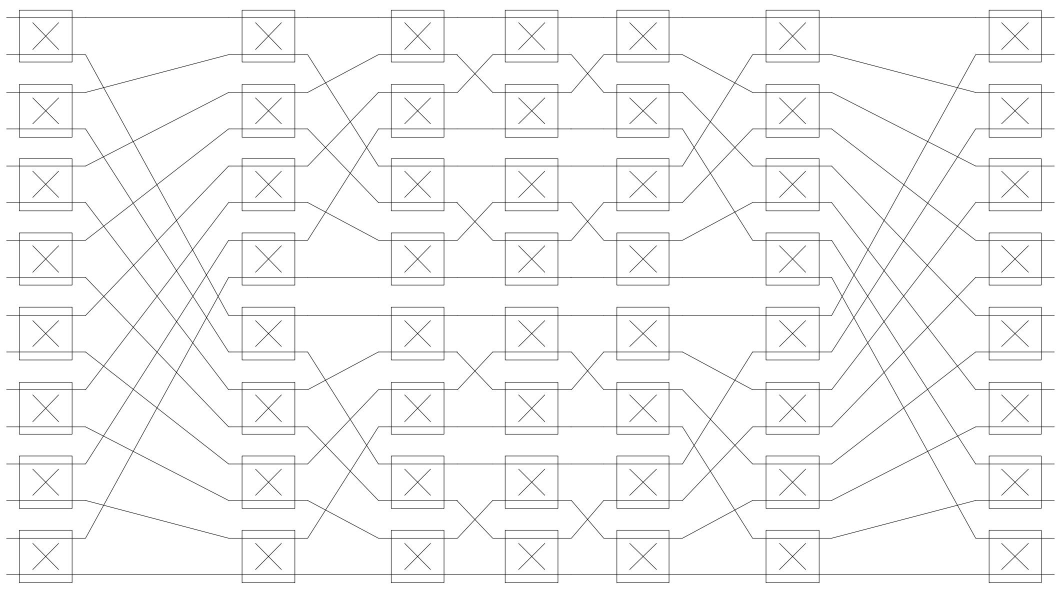
Figure 2.2 Cross-bar switch unit

As shown in Figure 2.2, the Benes network is constructed by numerous switch units. When the control signal CTL of this unit is 0, this unit is under so called “BAR” state, and conversely when the CTL is 1, the unit is under the “CROSS” state.

The construction of the Benes network is a recursive procedure. As shown in Figure 2.3 (a), Benes network can be built by stacking two sub-networks vertically and making shuffle connection between the bilateral stages and sub-networks, where Be(n) represents the Benes network which has 2^n inputs. Figure 2.3 (b) shows the example of a 16×16 Benes network.



(a) Benes network construction



(b) 16×16 Benes network

Figure 2.3 Benes network construction and 16×16 example

There are plenty of ways to generate the control signals of all the switch units in the Benes network. Compared with the dedicated look up table (LUT) [40] and bitonic sorting network [41], the most efficient method is generating the control signal in a recursive process like the Benes network itself [42], [43]. The basic idea of this recursive technique is producing the control signals of the first and last stages according to the parity property of input number (IN_n) and shift number (SN_n) for Be(n), where generally there are four situations (odd, even; odd, odd; even, odd; even, even) for IN_n and SN_n. Meanwhile, the control units of current stage also calculate the IN_{n-1} and SN_{n-1} for the sub-networks and pass these parameters stage by stage. The most advantage of this control signal generating scheme is that it greatly reduces the hardware complexity and saves much area. However, this approach would add much more delay in the critical path of the whole permutation network, which will lead to serious latency problem and throughput limitation of the entire decoder.

2.2 Improvement of LBA based permutation network

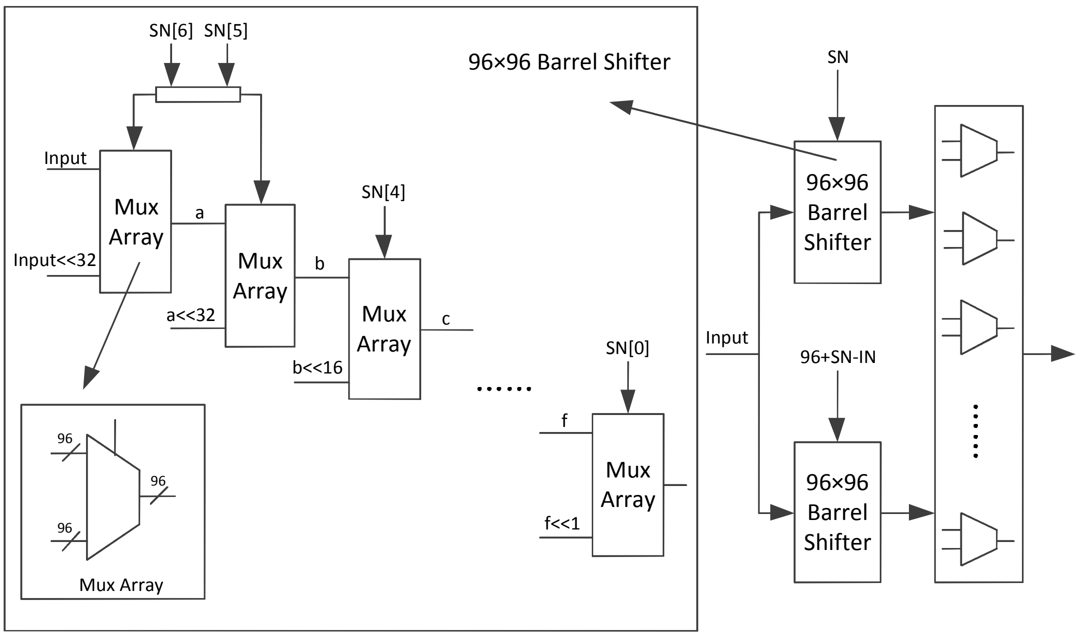
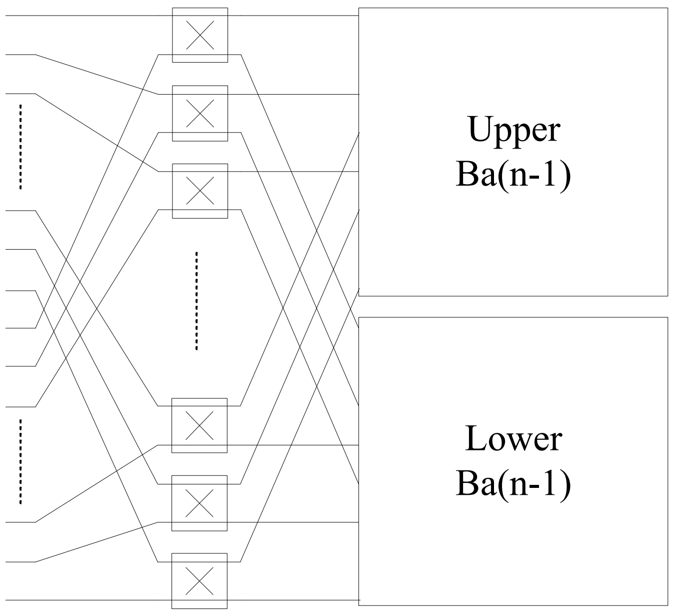


Figure 2.4 LBA based permutation network for QC-LDPC codes

Figure 2.4 shows the improvement of LBA based permutation network that can support various input number (IN) and shift number (SN). This structure contains two parallel 96×96 barrel rotators (BRs) for one direction cyclic shift and one multiplexer stage for selecting the required data at each port. BR is composed of 7-stage multiplexer array as the ordinary 128×128 logarithmic shifter, wherein the difference is that there are only 96 multiplexers and the first stage is capable of shifting number of 32. The control signals of the two BRs are SN and 96+SN-IN with the consideration of shift position offset, each bit of which is set as the control signal of each stage.

2.3 Banyan based permutation network

As shown in Figure 2.5, the Banyan network, proposed in [44], is also composed of the same switch units with Benes network, while it has only approximately half stages of the Benes network. Banyan network can also be constructed by the recursive manner, showed in Figure 2.5 (a), where Ba(n) represents the Banyan network which has 2^n inputs. Figure 2.5 (b) gives the 16×16 example. Compare with Be(n)'s 2n-1 stages, Banyan network has only n stages, which would greatly reduce the signal transformation time when performing the permutation. Furthermore, it is much easier to generate the control signals of Banyan network than Benes network, which would reduce the latency of the whole permutation network to a great extent.



(a)

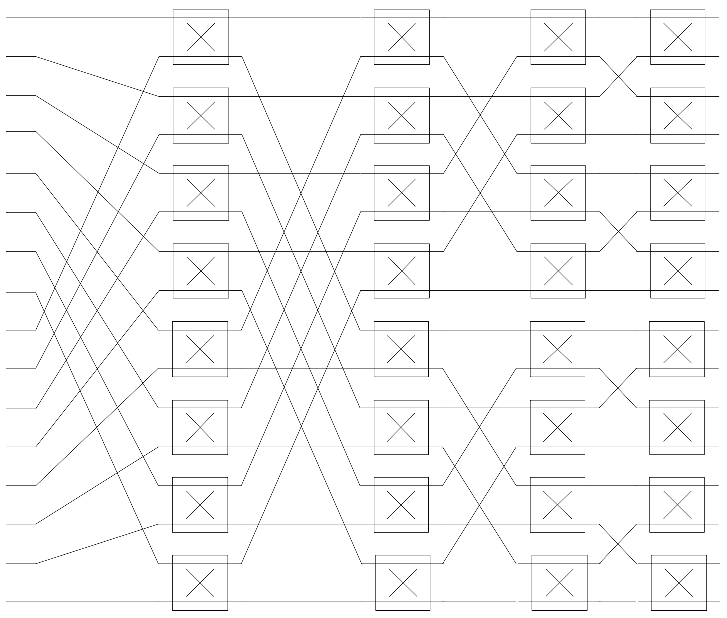

(b)

Figure 2.5 Banyan based permutation network for QC-LDPC codes

2.3.1 Non-blocking property under cyclic shift

Considering an n stage Banyan network, let the inputs and outputs be numbered from top to bottom by $x_1, x_2, \ldots, x_N$ and $y_1, y_2, \ldots, y_N$ (N=2^n). In stage *k* (*0<k <n*), there are $2^k\text{-}1$ sub-networks. For convenience, label the switch nodes according to the Beckmann number scheme. That is, a switch node in stage *k* is labeled by $(a_{n-k}a_{n-k-1}\cdots a_1, b_1b_2\cdots b_{k-1})$, where $a_{n-k}a_{n-k-1}\cdots a_1$ denotes the label of the switch node numbered from the top within the sub-network and $b_1b_2\cdots b_{k-1}$ represents the label of the sub-network. Figure 2.6 illustrates this numbering scheme for a four stage banyan network.

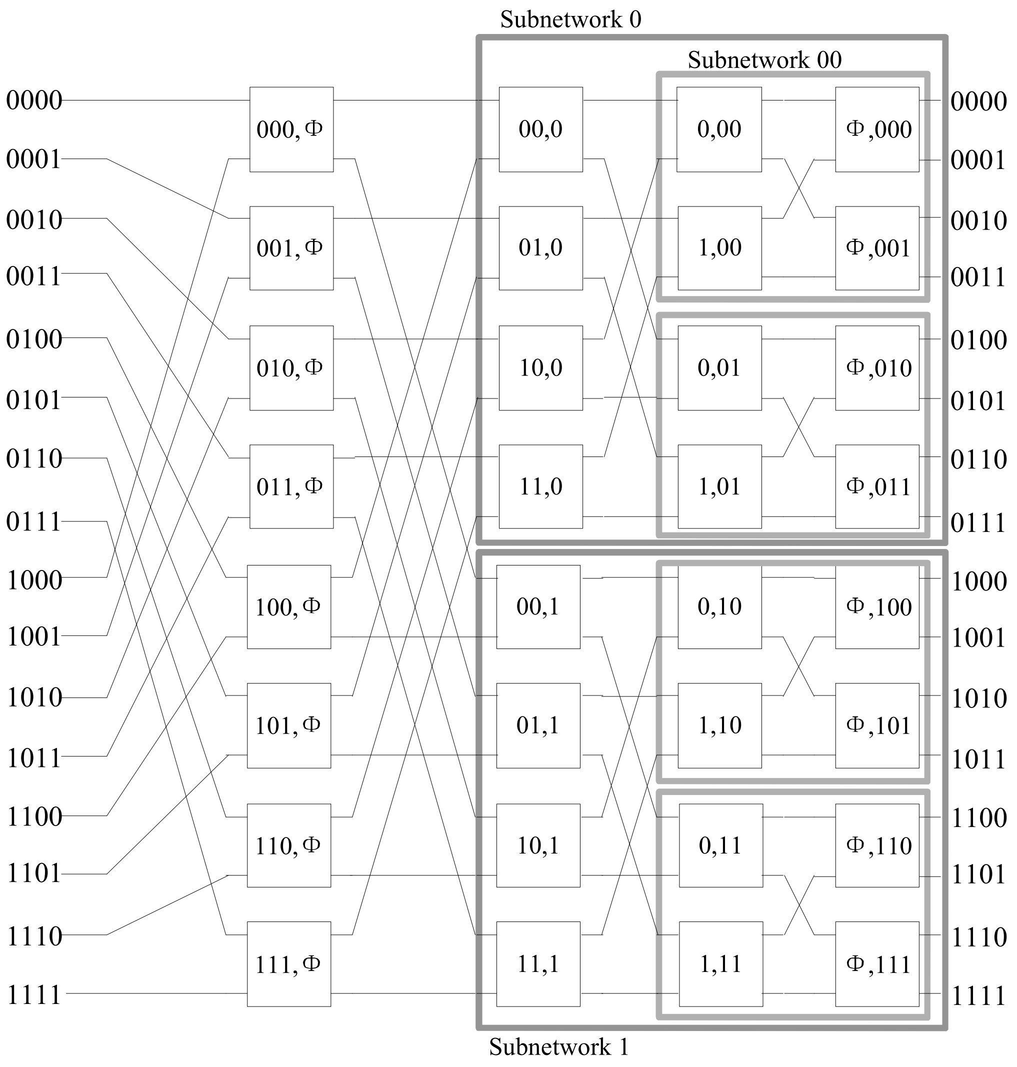


Figure 2.6 Numbering scheme of 16×16 Banyan network

According to the connection pattern of Banyan network, the node $(a_{n-k}a_{n-k-1}\cdots a_1, b_1b_2\cdots b_{k-1})$ in stage *k* links the node $(a_{n-k-1}a_{n-k-2}\cdots a_1, b_1b_2\cdots b_k)$ in stage *k+1* through the output link b_k (0 or 1). Thus, the path from input x_i $(i = a_{n-1}\cdots a_1)$ to output y_j $(j = b_1\cdots b_{n-1}b_n)$, consists of the following nodes: $(a_{n-1}\cdots a_1, \Phi)$, $(a_{n-2}\cdots a_1, b_1)$,…, $(a_{n-k}\cdots a_1, b_1\cdots b_{k-1})$,…, $(\Phi, b_1\cdots b_{n-1})$. Moreover, based on the property of cyclic shift, $j = (i + SN) \bmod 2^n$. Figure 2.7 illustrates a cyclic shift by 2 for 16×16 banyan network, where the input data $x_0 = 0$, …, $x_{15} = 15$ and output data $y_0 = 14$, $y_1 = 15$, …, $y_{15} = 13$.

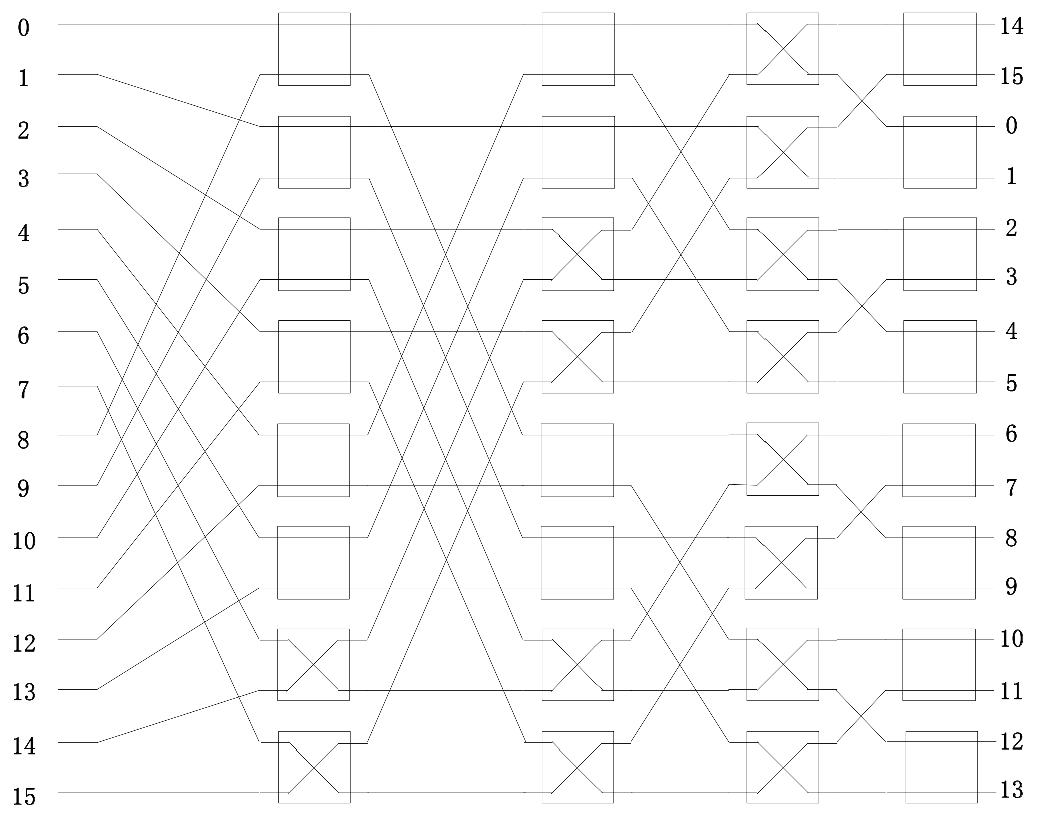


Figure 2.7 Cyclic shift example of 16×16 Banyan network

We shall prove that Banyan network is non-blocking with respect to cyclic shifts. This proof is an extension of H.S.Kim's discussion [45].

Proof: Considering the cyclic shifts in Banyan network, we set the input data as $x_1=1, x_2=2, ..., x_N=N$. Thus, the input data $x_1, x_2, ..., x_N$ is a monotonic increasing sequence. If $SN = 0$, clearly the data of the output $y_1, y_2, ..., y_N$ is also a monotonic increasing sequence. If $SN \neq 0$, on the basis of the property of cyclic shift, there are two monotonic increasing sequence in the data of outputs: $y_1=N\text{-}SN+1, ..., y_{SN}=N$, which are connected with $x_N\text{-}SN+1, ..., x_N$, and $y_{SN}+1=1, ..., y_N=N\text{-}SN$, which are connected with $x_1, ..., x_N\text{-}SN$.

Without loss of generality, it is denoted that the input $x_i(i=a_{n-1}\cdots a_1)$ is connected to the output $y_j(j=b_1\cdots b_n)$, and input $x_i'(i'=a_{n-1}'\cdots a_1')$ is connected to the output $y_j'(j'=b_1'\cdots b_n')$. As shown in Figure 2.8, there are two situations to check whether the two paths $x_i \rightarrow y_j$ and $x_{i'}' \rightarrow y_{j'}'$ are blocked or not.

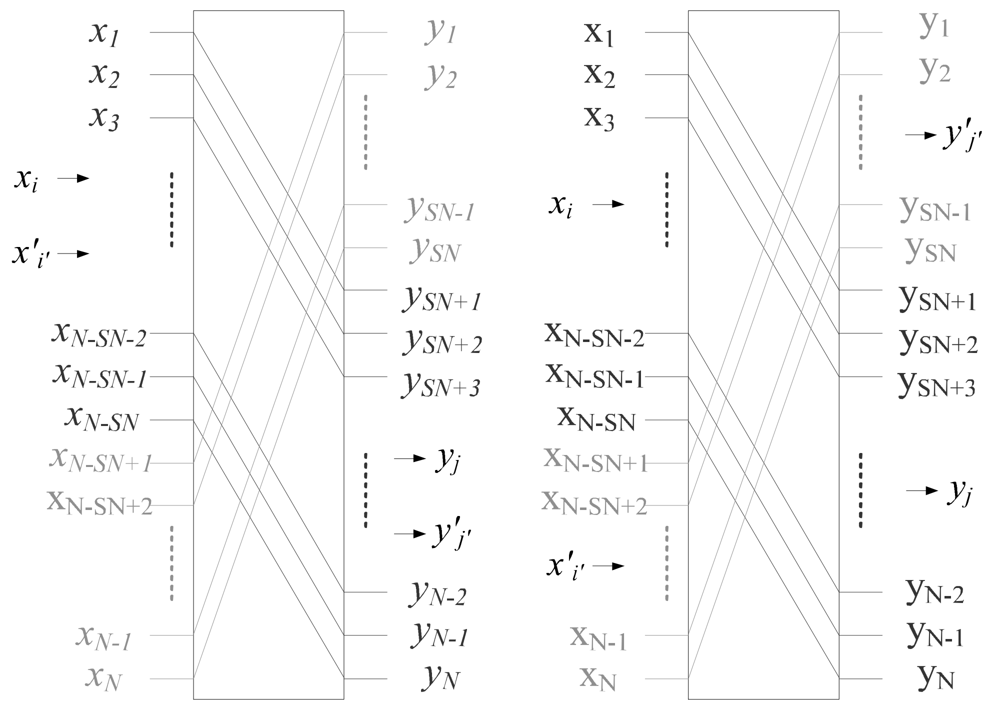


Figure 2.8 Two cyclic shift situation in Banyan network

First, Figure 2.8 (a) shows that the output y_j and $y'_{j'}$ are in the same monotonic increasing sequence. Under this condition, we suppose that the two paths collide at one node $(a_{n-k}a_{n-k-1}\cdots a_1, b_1b_2\cdots b_{k-1})$ in stage k, which means that $(a_{n-k}\cdots a_1, b_1\cdots b_{k-1}) = (a'_{n-k}\cdots a'_1, b'_1\cdots b'_{k-1})$ and the output link $b_k = b'_k$. Thus, we get

$$a_{n-k}a_{n-k-1}\cdots a_1 = a'_{n-k}a'_{n-k-1}\cdots a'_1 \tag{2.1}$$

$$b_1b_2\cdots b_{k-1}b_k = b'_1b'_2\cdots b'_{k-1}b'_k \tag{2.2}$$

According to the cyclic shifts property, we naturally get

$$i' - i = j' - j \tag{2.3}$$

Based on (2.1) and (2.2), we get

$$i' - i = a'_n\cdots a'_1 - a_n\cdots a_1 = 2^{n-k}(a'_n\cdots a'_{n-k+1} - a_n\cdots a_{n-k+1}) \geq 2^{n-k} \tag{2.4}$$

and

$$\begin{aligned} j'-j &= b_1'\cdots b_{n-1}'b_n' - b_1\cdots b_{n-1}b_n \\ &= b_{k+1}'\cdots b_{n-1}'b_n' - b_{k+1}\cdots b_{n-1}b_n \le 2^{n-k}-1 \end{aligned} \tag{2.5}$$

Obviously, (2.4) and (2.5) contradict (2.3), which means the assumption that two paths collide at one node is false. Therefore, the banyan network is non-blocking under the first situation. The situation that *SN = 0* can also be included in this situation as well.

Second, Figure 2.8 (b) shows that the output y_j and $y'_{j'}$ are in the different monotonic increasing sequence. Similarly we set the blocking assumption, and get (2.4) and (2.5). On the other hand, we also point out both the upper bound and lower bound here.

Since

$$i'-i = a_n'\cdots a_1' - a_n\cdots a_1 = 2^{n-k}(a_n'\cdots a_{n-k+1}' - a_n\cdots a_{n-k+1}) \tag{2.6}$$

we get

$$2^{n-k} \le i'-i \le 2^{n-k}\sum_{i=0}^{k-1} 2^i = \sum_{i=n-k}^{n-1} 2^i \tag{2.7}$$

and

$$j-j' = b_1\cdots b_n - b_1'\cdots b_n' = b_{k+1}\cdots b_n - b_{k+1}'\cdots b_n' \tag{2.8}$$

implies that

$$1 \le j-j' \le 2^{n-k}-1 \tag{2.9}$$

For cyclic shifts, in both the two monotonic increasing sequence, we get

$$N-i' = SN-j' \tag{2.10}$$

and

$$i-1 = j-SN-1 \tag{2.11}$$

Thus, (2.10)+(2.11), we get

$$N = (i'-i)+j-j' \tag{2.12}$$

Then, (2.8)+(2.9), we get

$$2^{n-k}+1 \le (i'-i)+j-j' = N = 2^n \le \sum_{i=n-k}^{n-1} 2^i + 2^{n-k} - 1 \qquad (2.13)$$

We notice that $\sum_{i=n-k}^{n-1} 2^i + 2^{n-k} = 2^n$, thus, it comes that $2^n \le 2^n - 1$, the contradiction proof is also established in this condition.

2.3.2 Cyclic shift algorithm of Banyan network

As the previous proof, when the IN is power of 2, the Banyan network is non-blocking for cyclic shift. However, contradiction proof is useless for hardware implementation. In order to design the controller of the Banyan network for generating all the control signals of switch units, we propose our cyclic shift algorithm in the following.

Algorithm I

Input:

IN (Input Number) $=2^n$,

SN (Shift Number), $0 \le SN < IN$

Output:

```
CTL (Control Signal) of each switch unit
For stage $j = 1:n$
  For $i = 1:\frac{IN}{2^j}$
    If $SN\%(2^{n-j+1}) \le \frac{IN}{2^j}$
      $ctl[i] = 0,\ i = 1 \sim (\frac{IN}{2^j} - SN\%(2^{n-j+1}))$
      $ctl[i] = 1,\ i = (\frac{IN}{2^j} - SN\%(2^{n-j+1})) + 1 \sim \frac{IN}{2^j}$
```

$$If\ SN\%(2^{n-j+1}) > \frac{IN}{2^j}$$

$$ctl[i]=1,\ i=1\sim(SN\%(2^{n-j+1})-\frac{IN}{2^j})$$

$$ctl[i]=0,\ i=(SN\%(2^{n-j+1})-\frac{IN}{2^j})+1\sim\frac{IN}{2^j}$$

①. % represents the modulus computation.

②. All sub-networks in the same stage have same control signals pattern.

This algorithm reveals the high symmetry property of Banyan network for cyclic shifts, which is summarized as follows:

First, the control signals of the first stages of every sub-networks Ba(n-1) is same, which would significantly reduce the hardware complexity of control signal generator.

Second, for the first stage of Ba(n), as SN varying from 0 to 2^n-1, the control signals basically has 2^n kinds of pattern. However, it can be found that the control signal pattern of 0~2^{n-1}-1 and 2^{n-1}~2^n-1 has binary complementary relationship. That is, $ctl_i=\overline{ctl_{i+2^{n-1}}}$, where $\bar{x}$ means the bitwise not operation. This property can reduce the hardware cost further more. Since the descriptions of the algorithm cannot be directly perceived through the senses, we give the control signal patterns example of 8×8 Banyan network for cyclic shifts.

2.3.3 Cyclic shifts with IN is not power of 2

When the IN is not power of 2, the cyclic shift of Banyan network will not be non-blocking. Figure 2.9 shows the example of obstructed situation in the 8×8 Banyan network with IN is 6 and SN is 2. As Figure 2.9 shows, if we want to achieve that cyclic shift, the path 1 and path 5, path 2 and path 6 will collide with each other in stage 1 and stage 2.

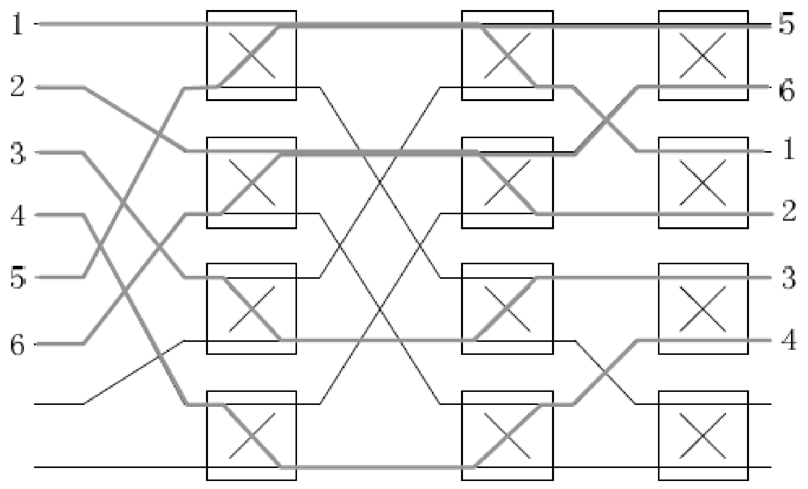


Figure 2.9 Blocking example of 8×8 Banyan network

In Banyan network, because every permutation has a unique path, there would exist block situation of two permutation path if the IN of not power of 2. Unfortunately, most of the QC-LDPC codes sub-matrices size defined in wireless standards mentioned above is not power of 2. In order to realize cyclic shift with any IN and SN, we propose introducing the “bypass network” to eliminate the obstruction. Figure 2.10 shows the example of solving the blocking problem in Figure 2.9.

As shown in Figure 2.10, path 5 and path 6, conflicted by path 1 and path 2, go through the bypass network and arrive at the destination output posts. Generally, for any IN (IN<2^n) and any SN (0≤SN<IN), it is certain that 2^n×2^n Banyan network can execute non-blocking permutation for either the lower SN inputs or the rest (IN−SN) inputs. The essential reason why blocking will appear is that the first stage is acting as a sorting switch to sort the inputs into the upper sub-network or the lower sub-network. If the IN is power of 2, every pair $(x_i, x_{i+2^{n-1}})$ will be split into two different sub-network, which guarantees the non-blocking property. Once the IN is not power of 2, $(x_i, x_{i+2^{n-1}})$ may go into the same sub-network, and the related two paths will clash in some nodes of the network.

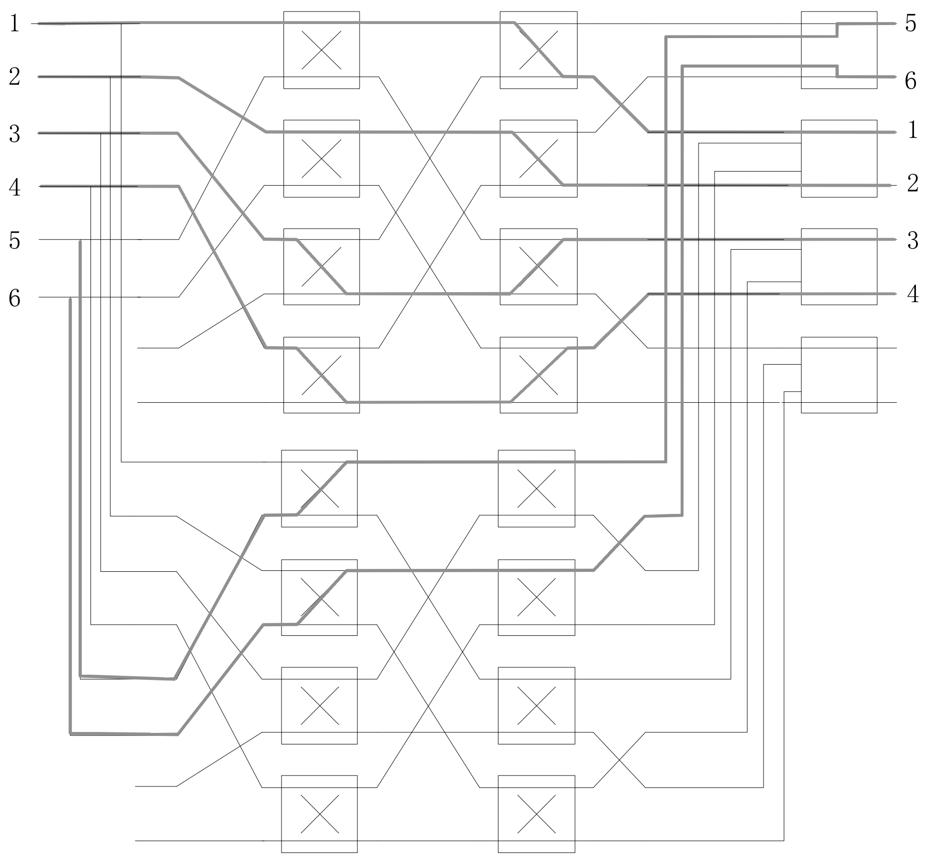


Figure 2.10 Eliminating obstruction example in 8×8 Banyan network

Therefore, if the lower SN inputs are transformed by other paths, the collision will never occur. As Figure 2.10 shows, the bypass network has n-1 stages. The switch units of the last stage of the original network are replaced by the 4 to 2 switch units, in which we add another control signal to judge the final outputs selecting the original network outputs or bypass network outputs.

2.3.4 Implementation

For the hardware implementation of the permutation network, we design it to satisfy the IEEE 802.16e and IEEE 802.11n standards. The maximum length of sub-matrix is 96, which determines that our design is the 128×128 network. Thus, there are 7 stages in our implementation. As shown in Figure 2.11, the inputs of bypass network are duplication of inputs of original network. At the last stage, we use 4 to 2 switch units to merge the two paths from original network and bypass network into one. Both the original network and the bypass network use the same control signal generator structure. The only difference of the two controllers is that the input of original network controller is SN, while the input of the bypass network controller is 128+SN−IN.

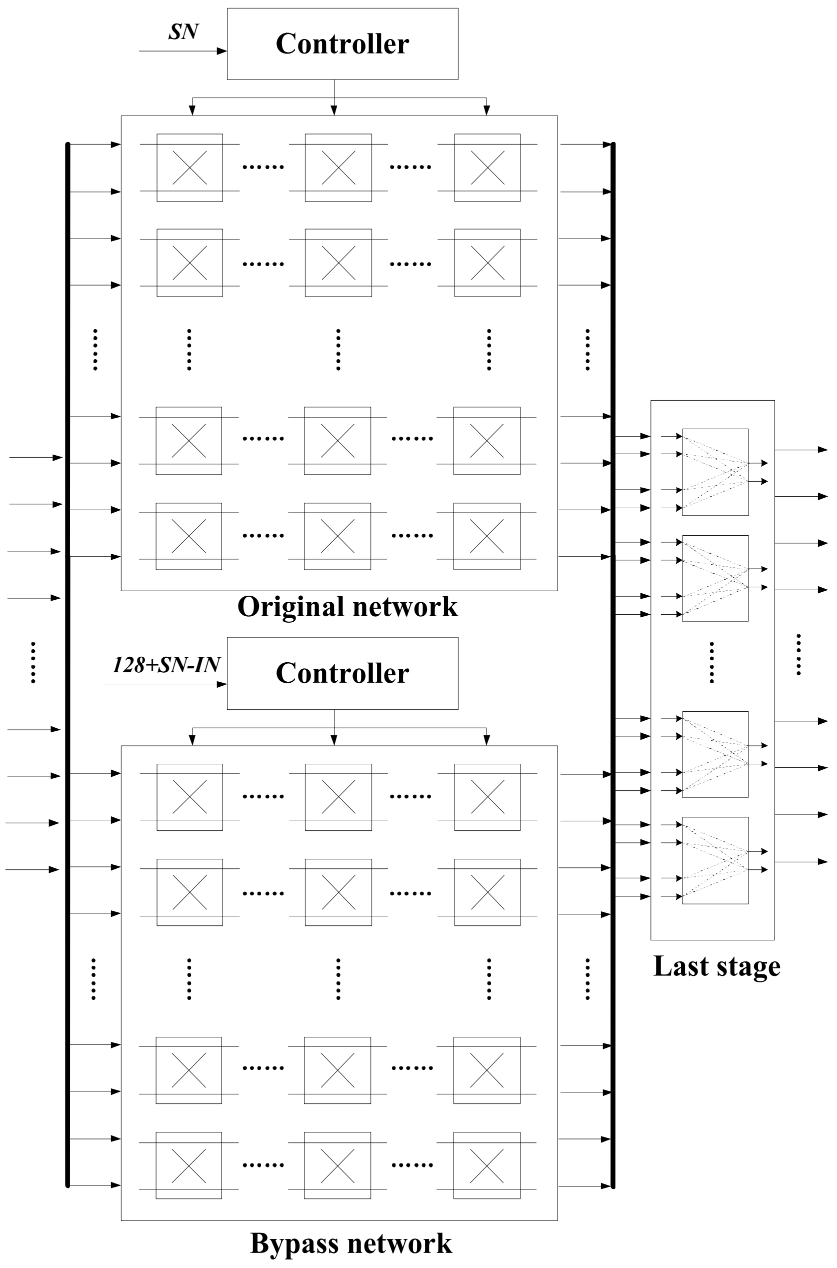


Figure 2.11 Top level architecture of proposed permutation network

Figure 2.12 illustrates the architecture of the control signal generator. According to Algorithm I, the control signals of every stage are only determined by $SN\%(2^{n-j+1})$. For hardware design, the modulus operation by 2 is just getting the low bits of the number. Consequently, we just feed the required low bits of SN into every stage control signal generator (SCSG) without any other operation. Compared with the stage by stage scheme in [42], [43], our proposal can greatly reduce the length of critical path.

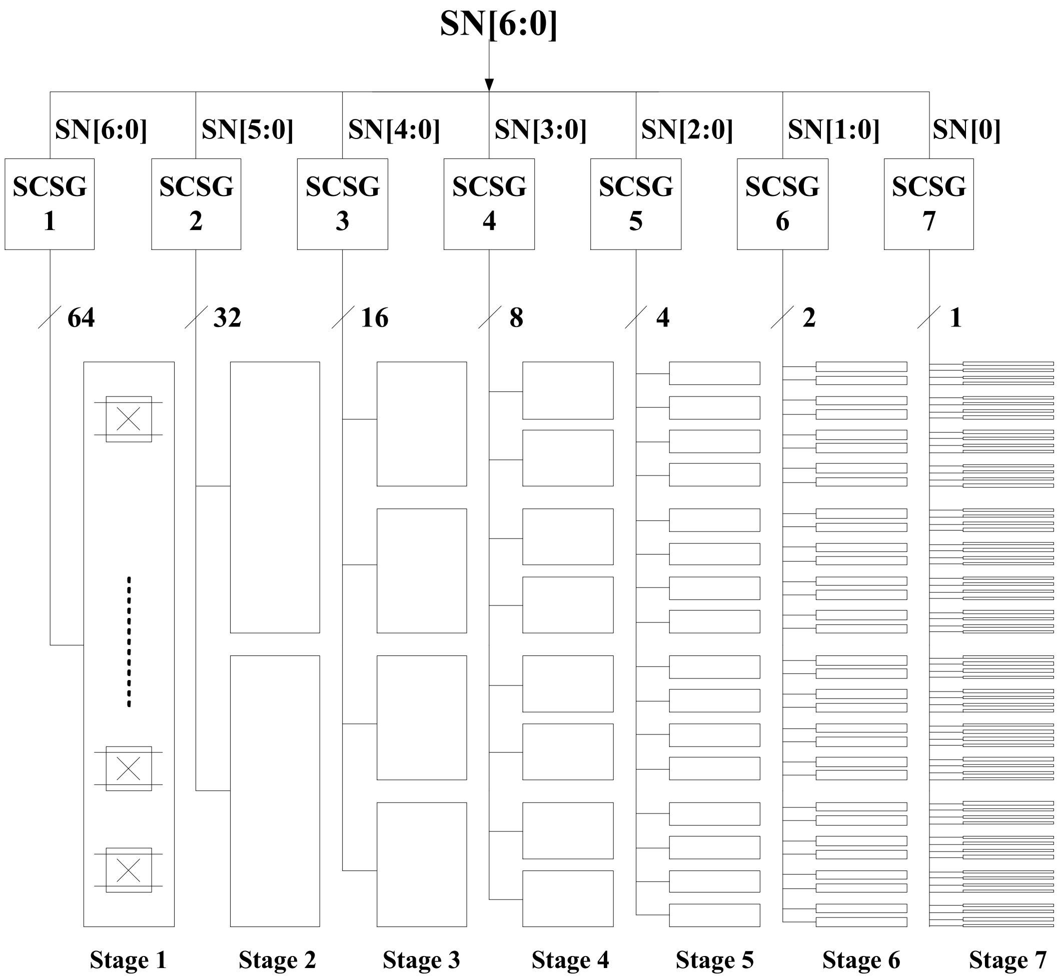


Figure 2.12 Architecture of control signal generator

For each SCSG, the architecture is almost the same except the output number. Figure 2.13 shows the construction of SCSG1.

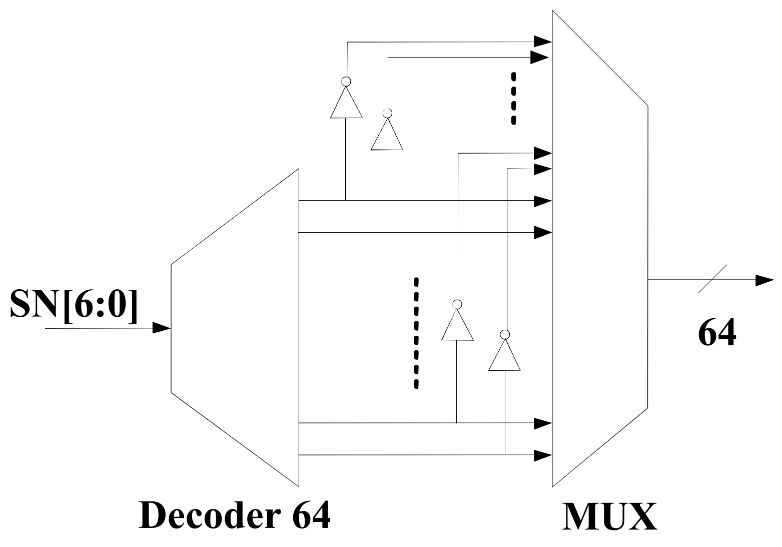


Figure 2.13 Construction of SCSG1

In the 128×128 Banyan network, there are 2^7=128 kinds of control signal pattern in stage 1. Based on the complementary symmetry property of the pattern discussed in section 3, we need only a 7 to 64 regular decoder but not a 7 to 128 regular decoder. The SCSG1 can be simply implemented use the decoder 64, NOT gates and a MUX.

The last stage of the permutation network is responsible of merging the original paths and bypass paths. The switch unit of this stage is 4 to 2 unit and additional control signals are added to this stage.

As shown in Figure 2.14, there are four kinds of control signals to control the switch units of the last stage. Firstly, the last stage control signals of the original network and bypass network have only two patterns: all 0s and all 1s. Moreover, other control signals are desired for choosing the original network outputs or bypass network outputs. For the situation that SN is even, only weight control (W_CTL) signals which can also be produced by a decoder are needed since the two outputs of all units come from the same network, either the original one or the bypass one. If the SN is odd, there exists a unit in which the upper output is from bypass network and the lower output comes from original network. Therefore, the extraordinary control (X_CTL) signal is brought to point out this exceptional unit.

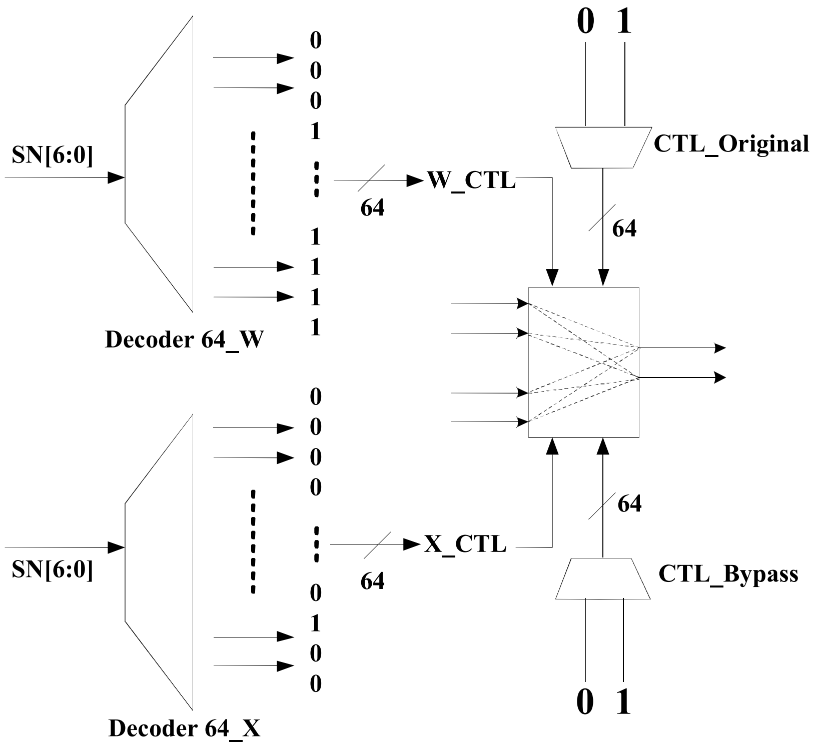


Figure 2.14 Construction of the last stage

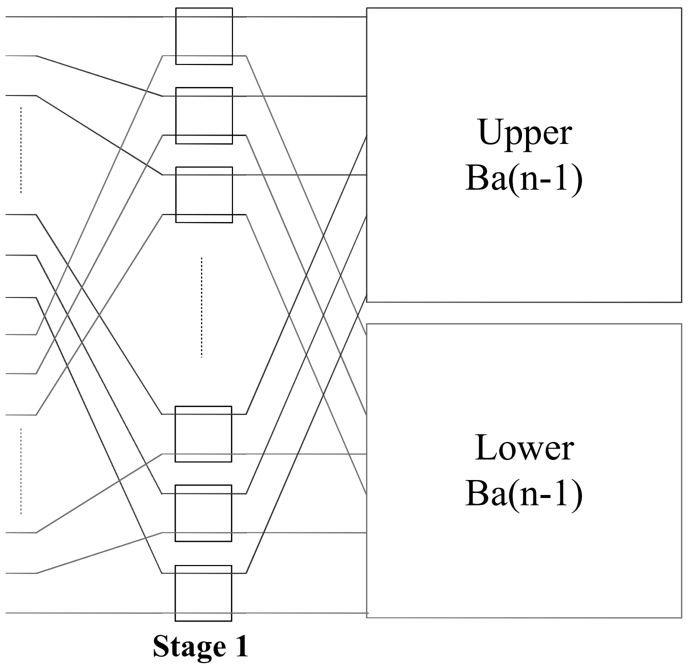


Figure 2.15 Parallel configuration of Banyan network

When the IN is less than 64, such as the block length 24, 28, …, 64, defined by WiMAX, the proposed network could perform two cyclic shifts in parallel with a small justification in the controller.

As shown in Figure 2.15, according to the recursive property of the Banyan network, if we set all the control signals of the first stage to “0”, the first 64 input data are fed into the upper sub-network and the rest data go into the lower sub-network. Therefore, the same shift algorithm and controller design could be used for both of the two sub-networks.

For the synthesis result of TSMC 0.18μm, the proposed permutation network based on Banyan network, can be implemented with area of 0.546 mm2 and a maximum frequency of 292 MHz.

Table 2.1 summarizes comparison among the proposed permutation network and existing networks for QC-LDPC decoders, where all the data are the synthesis result. Compared with [9] which is based on Benes network, the controller area of our proposal is reduced by 75%, the total area of the permutation network is reduced by 24%, and the maximum frequency of our scheme is 3 times of [41]. [42] and [43] can only support 19 and 22 kinds of IN respectively, while our proposal can accommodate 128 kinds of IN.

TABLE 2.1 COMPARISONS OF THE PERMUTATION NETWORK

	[41]	[42]	[43]	[39]		Proposed Design	
Network size	128×128	96×96	96×96	96×96	96×96	128×128	128×128
Message bits	8 bits	8 bits	6 bits	6 bits	6 bits	8 bits	8 bits
Technology	0.18μm	0.18μm	0.18μm	0.13μm	0.13μm	0.18μm	0.18μm
Controller Area (mm^2)	1.262	0.114	——	——	——	0.029	0.039
Total Area (mm^2)	2.171	0.722	0.160	0.110	0.187	0.546	0.586
Gate Count	——	——	16.9k	23.1k	37.4k	54.7k	58.7k
Max Frequency	85MHz	94MHz	300MHz	384MHz	384MHz	292MHz	292MHz
Kinds of Input Number	128	96	19	22	22	128	128
Parallelism	1x	1x	1x	1x	2x	1x	2x

In addition, when the IN is less than the half of IN_{max}, the proposed network can use little hardware cost to realize the parallel permutation because of the symmetry and recursiveness. The area of parallel permutation network is 107% of the unparallel network in the proposed design, while the cost of parallel permutation is 162% of the unparallel one in [39].

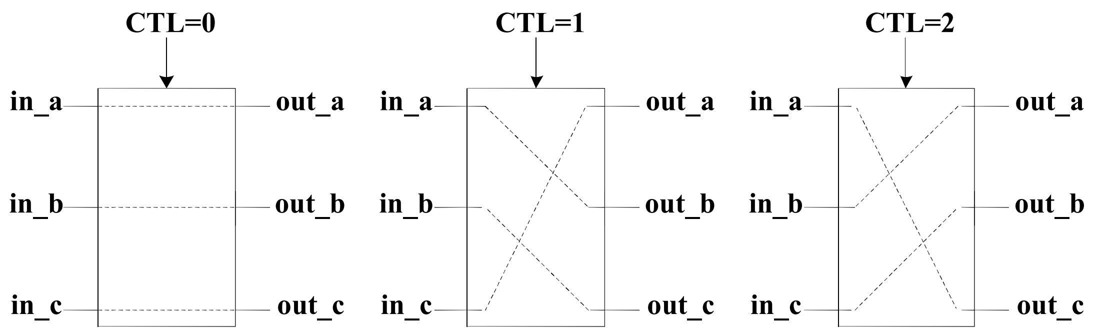
2.4 Generic permutation network

All the approaches above fail to optimize the network according to the requirement of application exactly. For example, it is quite inefficient to use the previous 96×96 even 128×128 network for the WiFi application since the maximum input number defined in WiFi is 81. As a consequence, this paper presents the architecture of generic permutation network (GPN), which is capable of constructing required permutation network for any given application with efficient control signal generating algorithm and high parallelism.

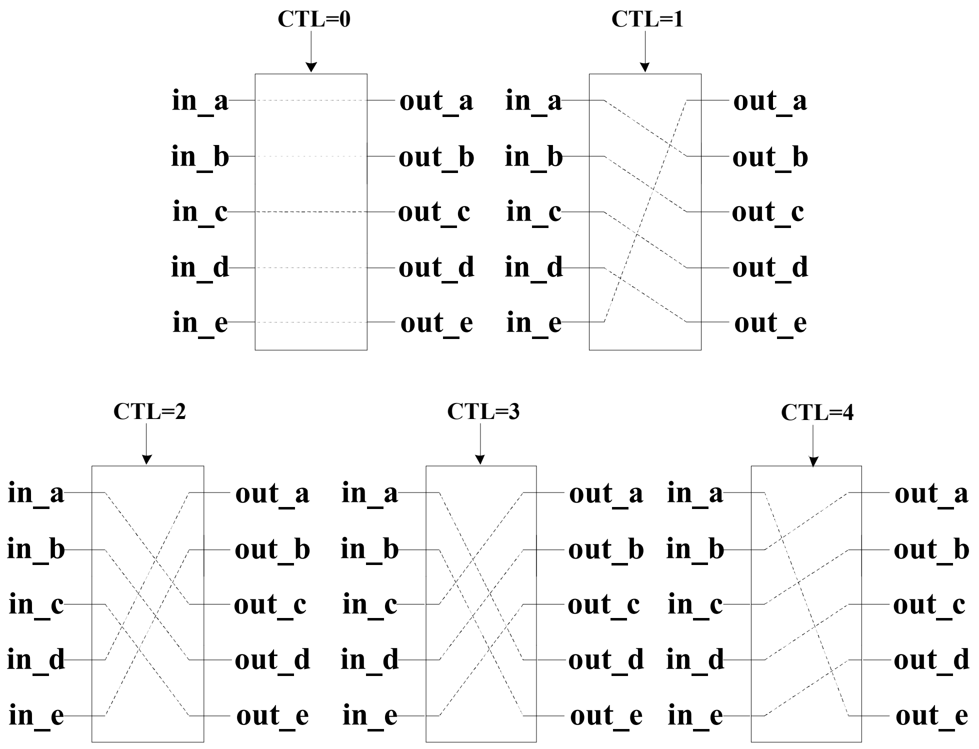
2.4.1 Switch Unit of generic permutation network

The switch units of GPN include not only the 2×2 switch unit, but also the $p \times p$ switch units, where the p is prime number, such as 3, 5 and 7. The function of all the $p \times p$ switch units is the cyclic shift capability. Figure 2.16 shows the 3×3 and 5×5 switch unit.

As shown in Figure 2.16 (a), this 3×3 switch unit can perform 3×3 cyclic shift, resulting in the 3 type of control signal CTL. In the same way, 5×5 switch unit has 5 types of CTL.



(a) 3×3 switch unit



(b) 5×5 switch unit

Figure 2.16 3×3 switch unit and 5×5 switch unit of GPN

2.4.2 Construction and cyclic shift algorithm of GPN

As we known, each positive integer except prime number could be represented as the product of several prime numbers. That is,

$$N = \prod_i p_i \tag{2.14}$$

Utilizing this factorization in the permutation network, the GPN could be constructed using some cascading stages of switch units.

As shown in Figure 2.17, the SU p_i represents the $p_i \times p_i$ switch unit. When each p_i is 2, the GPN become the original Banyan network. In other words, the original Banyan network is special case of GPN.

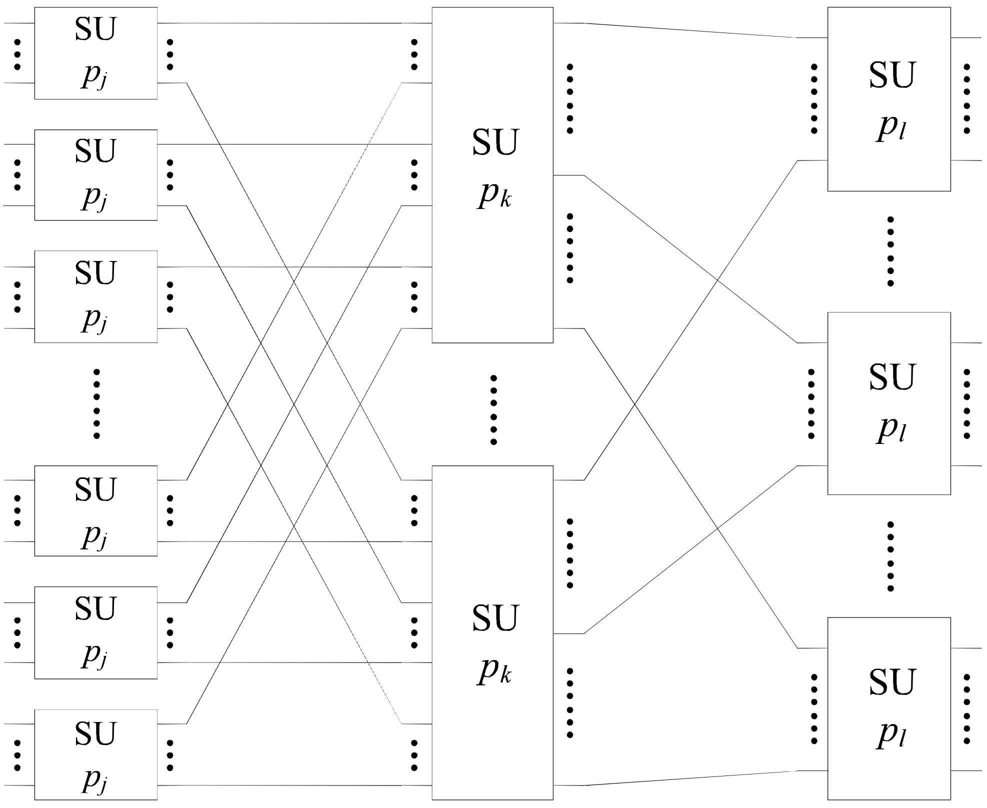


Figure 2.17 Construction of GPN

Figure 2.18 shows an example of 30×30 GPN. Since 30=2×3×5 in factorization, the GPN contains the corresponding 3 kinds of switch units. This example reveals the mathematical essence of GPN, which can be regarded as grouping. When the data pass though one stage, they are fed into several identical sub-networks which can be deemed to be assigned into numerous groups. On the base of cyclic shift, GPN should satisfy the following properties:

First, each stage in GPN is made up of N/p_i $p_i \times p_i$ switch units. The position of every stage in the whole GPN is equal. Namely, the order of stages in GPN could be freely changed.

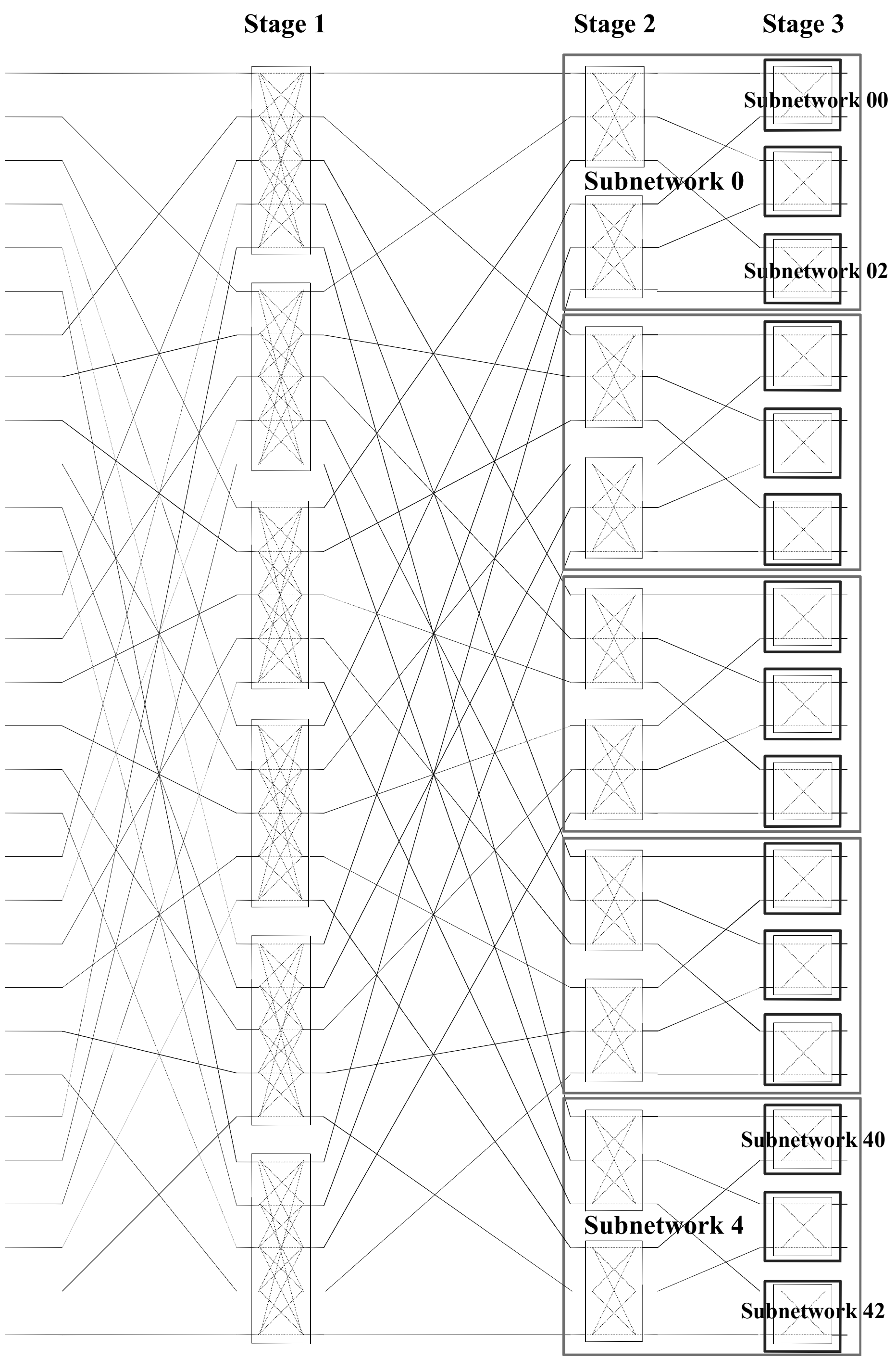


Figure 2.18 30×30 generic permutation network

Second, there exist shuffle connections before the first stage. Since in implementation these connections are just wires and cost nothing, they are not need to be regarded as one stage. This connection pattern is related to the choice of the first stage switch unit. Generally, the first N/p_i inputs connect the first input of each switch unit in sequence, the second N/p_i inputs hold all the second input of each switch unit, etc.

Third, the connections between stages have similar manner. As shown in Figure 2.18, assuming the stage i is made up of $p_i \times p_i$ switch units, the rest stages could be divided into p_i sub-networks. In this way, the outputs of first switch unit in stage i connect the first input of each sub-network in stage $i+1$, and the outputs of second switch unit in stage i occupy all the second position in stage $i+1$, and so on.

The control signal generating algorithm for cyclic shift in GPN is the natural product of the construction manner. Without loss of generality, assuming that input number $IN=mp$, the first stage has m $p \times p$ switch units. The corresponding control signals of switch unit in the first stage could be generated as follows:

Algorithm II

Input:

IN (Input Number) = mp

SN (Shift Number), $0 \le SN < IN$

Output:

CTL (Control Signal) of first stage

```
compute $SN = xm + y,\ 0 \le x < p,\ 0 \le y < m$
  For $i = 1:(m-y)$
      $ctl[i] = x$
  end
  For $j = (m-y)+1:m$
      if $x = (p-1)$
         $ctl[j] = 0$
      else
         $ctl[j] = x+1$
      end
  end
```

Based on the recursive property, the algorithm II is effective for each first stage in every sub-network, which means that this algorithm could generate all the control signals of GPN. It is worth mentioning that the control signals for all the equal level sub-networks is same, which would significantly reduce the hardware complexity of control signal generator.

Table 2.2 gives the example of control signals for the 9×9 GPN, which is composed of two stages and six 3×3 switch units.

TABLE 2.2 CONTROL SIGNALS FOR 9×9 GPN

SN	stage		SN	stage		SN	stage	
	1	2		1	2		1	2
	0	0		1	0		2	0
0	0	0	3	1	0	6	2	0
	0	0		1	0		2	0
	0	1		1	1		2	1
1	0	1	4	1	1	7	2	1
	1	1		2	1		0	1
	0	2		1	2		2	2
2	1	2	5	2	2	8	0	2
	1	2		2	2		0	2

We shall prove that the above structure and algorithm could realize cyclic shifts when $IN=IN_{max}$.

Proof: Since the shifting through GPN is the process of assigning the data into sub-networks until the last stage, it is needed to prove two aspects:

1) After passing through each stage, all the data are fed into the proper sub-network

2) The cyclic shift property should be guaranteed when the network fed the data into the sub-network.

Without loss of generality, we consider the process of passing the first stage.

First, we will prove that the after passing the first stage, the data will go to the desired sub-network.

We use the assumptions and expressions in the algorithm II that is the first stage has m $p \times p$ switch units,

$$IN = mp,\ SN = xm + y,\ 0 \le x < p,\ 0 \le y < m \tag{2.15}$$

Then, the rest stages could be divided into p sub-networks. The denotation method is shown in Figure 2.18, from which we denote the sub-networks as 0 to p-1.

Consider a certain input i, we denote the data with its position from top to bottom,

$$i = x'm + y',\ 0 \le x' < p,\ 0 \le y' < m \tag{2.16}$$

It means that this input will be fed into the y' switch unit. In this $p \times p$ switch unit, the input position of i is x'. According to algorithm II, the control signal of this switch unit is

$$ctl[y'] = \begin{cases} x & y' \le m - y \\ x+1 & y' > m - y \ and \ x < p-1 \\ 0 & y' > m - y \ and \ x = p-1 \end{cases} \tag{2.17}$$

Since the switch unit can only perform shifter like cyclic shift operation, the output position is

$$k = \begin{cases} x + x' & y' \le m - y \,\&\, x + x' < p \\ x + x' - p & y' \le m - y \,\&\, x + x' \ge p \\ x + x' + 1 & y' > m - y \,\&\, x < p - 1 \,\&\, x + x' + 1 < p \\ x + x' + 1 - p & y' > m - y \,\&\, x < p - 1 \,\&\, x + x' + 1 \ge p \\ x' & y' > m - y \,\&\, x = p - 1 \end{cases} \tag{2.18}$$

Based on the structure of GPN, this position is connected to the *kth* sub-network.

On the other hand, the desired sub-network depends on the destination of the input i, in which there are two situations shown in Figure 2.19.

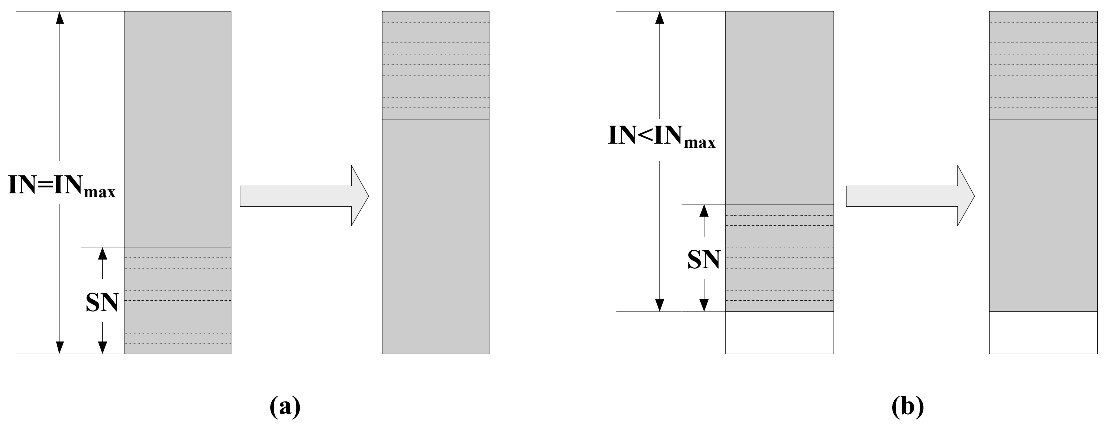


Figure 2.19 Two situations of input *i*

For the first situation in Figure 2.19, the output position is

$$o = i + SN = (x + x')m + y + y' \tag{2.19}$$

Since there are *p* sub-networks, the size of sub-network is *IN/p=m*. Then this output position is belong to the *kth* sub-network, where

$$k' = \left\lfloor \frac{o}{m} \right\rfloor \tag{2.20}$$

According with the distribution in (2.18),

$$k' = \left\lfloor x + x' + \frac{y + y'}{m} \right\rfloor = \begin{cases} x + x' & y' < m - y \\ x + x' + 1 & y' \geq m - y \end{cases} \tag{2.21}$$

Similarly, for the second situation in Figure 2.19, the output position is

$$o = i + SN - IN = (x + x' - p)m + y + y' \tag{2.22}$$

$$k'' = \left\lfloor x + x' - p + \frac{y + y'}{m} \right\rfloor = \begin{cases} x + x' - p & y' < m - y \\ x + x' + 1 - p & y' \geq m - y \\ x' & y' \geq m - y \,\&\, x = p - 1 \end{cases} \tag{2.23}$$

Compare with (2.18) and (2.22), (2.23), it is obvious that after passing the first stage, the sub-network which the certain input data is fed into is the same with the one that its final desired output position belongs to. Namely, through the first stage, all the data are fed into the proper sub-network.

Second, we will prove that the data fed into each sub-network still hold cyclic shift property.

Consider certain sub-network *j*, the m input of this sub-network come from the *jth* output position of m switch unit in the first stage. According to algorithm I, the corresponding input position in each switch unit is

$$in\_switch[i] = \begin{cases} j-x & i \le m-y \\ j-x-1 & i > m-y \end{cases} \tag{2.24}$$

For simplicity, we just consider one situation. The other situation can be verified in the same way.

Based on the structure of GPN, the input from the same position in every switch unit come from the same increasing sequence. Thus, from (2.24), the *0th* to *(m-y)th* input of sub-network *j* form an increasing sequence, and the rest form another increasing sequence. Moreover, consider the first data in the first sequence, it comes from the first switch unit in the first stage, then it is

$$a = (j-x-1)m+1 \tag{2.25}$$

Then consider the last data of the second sequence, it comes from the last switch unit in the first stage, it is

$$b = (j-x-1)m \tag{2.26}$$

Thus, the two sequences form a whole increasing sequence. Namely, the input of sub-network *j* hold the cyclic property.

Take Figure 2.18 as an example, if SN=3, input of the sub-network from top to bottom are 6, 7, 8, 3, 4, 5.

Above all, the two proved aspects ensure that all the data could be divided into proper group with respect to cyclic shift.

2.4.3 Cyclic shift with IN less than IN_{max}

Like original Banyan network, since every permutation in GPN has the unique path, there exist block situations among permutation paths if the IN is less than IN_{max}. The essence of blocking issue is that when IN is less than IN_{max}, the recursive cyclic structure between network and sub-network is broken. Namely, a single GPN cannot realize reconfigurable cyclic shift for various IN. In order to realize cyclic shift with any IN, the bypass network is introduced to eliminate the obstruction. Figure 2.20 shows the example of solving the blocking problem.

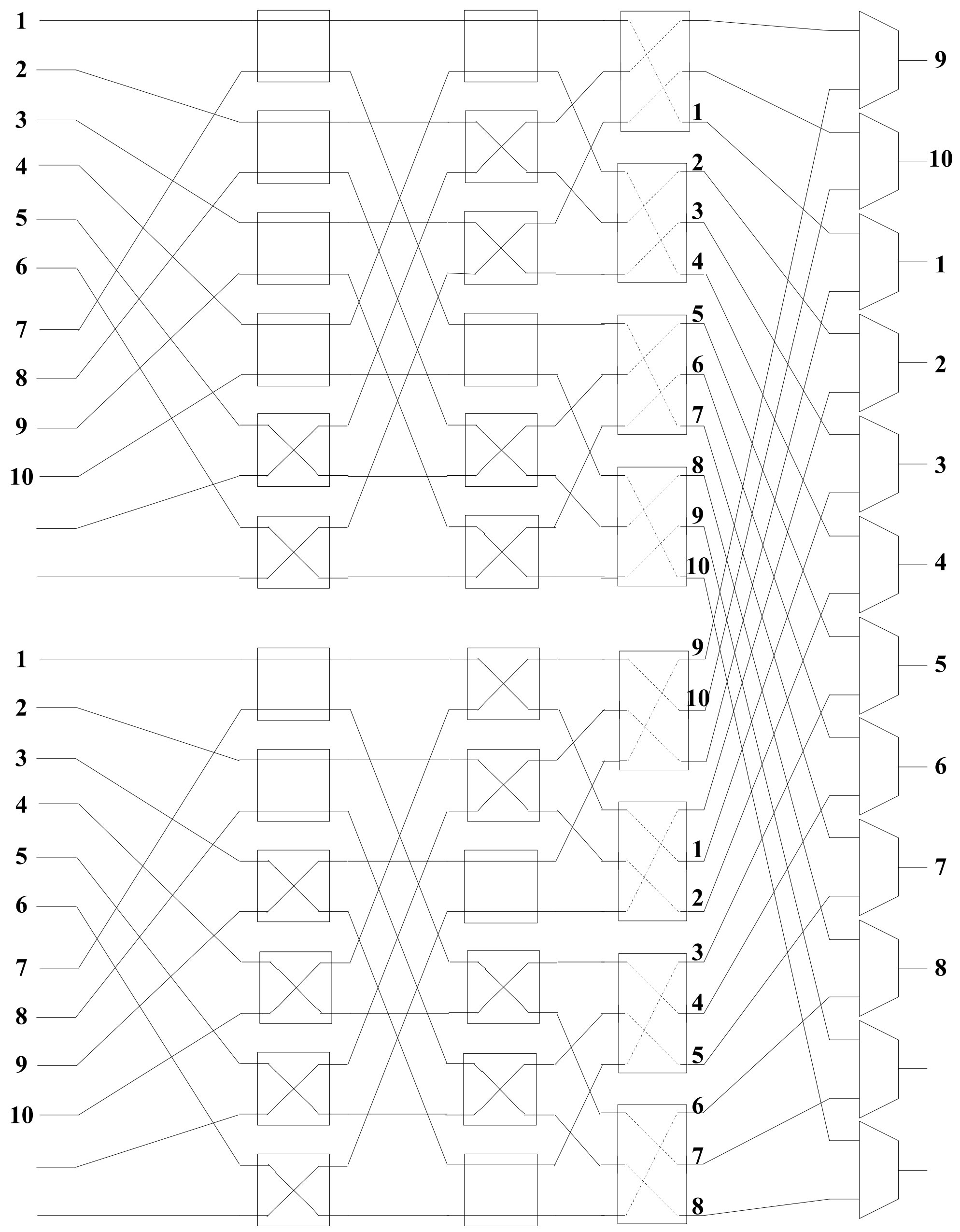


Figure 2.20 Example of cyclic shift in 12×12 GPN (IN=10, SN=2)

As shown in Figure 2.20 of the 12×12 GPN, IN is 10 and SN is 2. If the input 9 and 10 go through the bypass network, and finally be chosen by the last multiplexer stage, there would be no blocking situation.

Generally, for cyclic shift, the lowest SN inputs should be transferred to the top SN position, if we regard that the network has its height. The conflicting situations comes from that some of the input which are not the lowest, such as the input 9 and 10 in Figure 2.20, should go to the top position. If these inputs are transformed by other paths, the collision will never occur.

2.4.4 Hardware Implementation of GPN

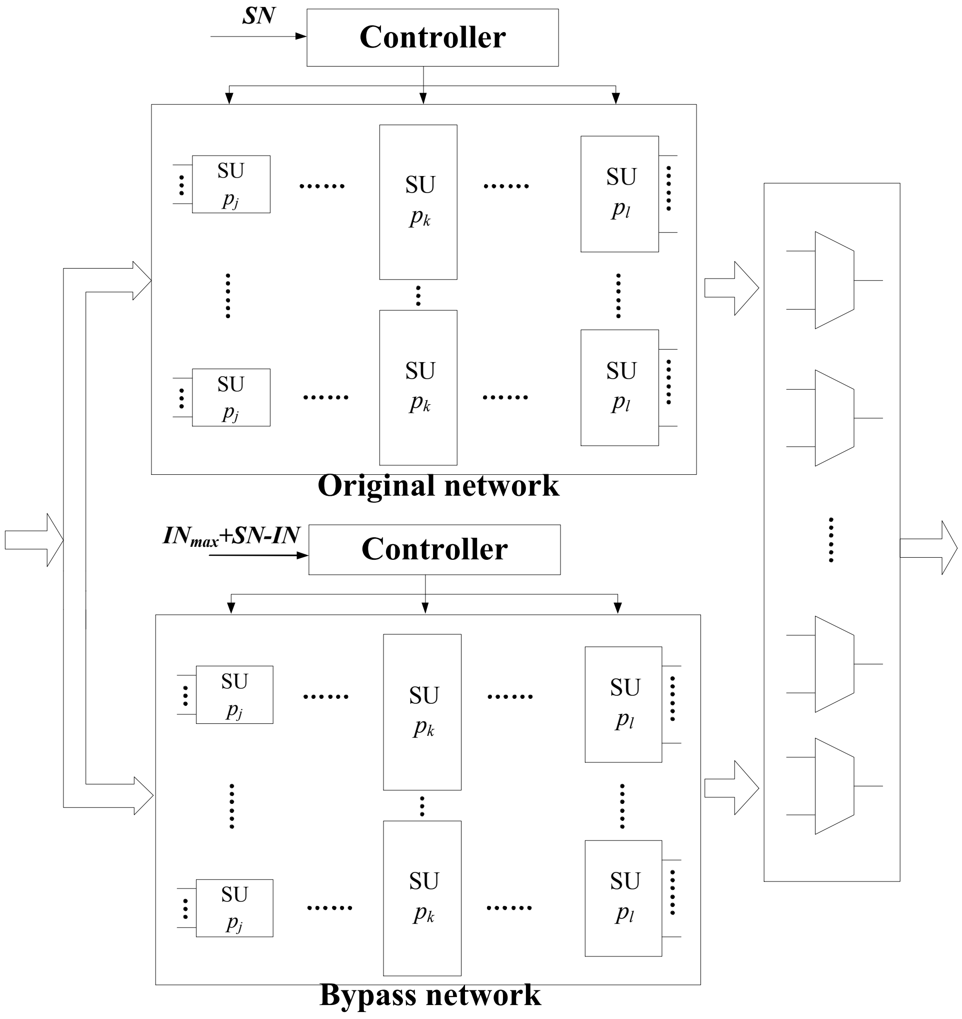


Figure 2.21 Proposed architecture of GPN

As shown in Figure 2.21, the inputs of bypass GPN are duplication of inputs of original one. Both the original GPN and the bypass GPN use the same control signal generator structure. The only difference of the two controllers is that the input of original GPN controller is SN, while the input of the bypass GPN controller is $IN_{max}+SN-IN$. This architecture guarantees that the lower SN data pass through the bypass network which would not block the upper IN-SN data any more. At the last stage, multiplexers merge the two paths from original GPN and bypass GPN into one.

Comparing this architecture with the conventional Benes network and barrel shifter based approach, the advantage of this architecture mainly relies on the non 2’s power IN. The conventional Benes network has $2\log_2 IN-1$ stages, and each stage contains IN/2 cross bar switch units. The barrel shifter has $\log_2 IN$ stages and each stage contains IN 2-to-1 multiplexer. Considering the bypass network and that a cross bar switch is equal to 2 multiplexers, the approach in [43] has $2\log_2 IN+1/2$ stages. When IN is power of 2, GPN becomes the Banyan network. Similarly, Banyan network has the same switch units in each stage, but contains $\log_2 IN$ stages. Considering the bypass network and the last multiplexer stage, the total hardware cost could be regards as $2\log_2 IN+1/2$ stages. In this case, the Banyan network approach and barrel shifter approach use small hardware cost to achieve high speed. The total hardware cost of barrel shifter is the least since the controller is very simple.

However, when IN is not power of 2, such as the application in WiMAX and WiFi, the conventional Benes network and barrel shifter still hold the same size, while GPN could make the network size much smaller and the corresponding stages and the number of switch units in every stage become much less.

For the hardware implementation of GPN, the design of $p\times p$ switch unit is the most important factor which determines the performance of the whole permutation network. The 2×2 switch unit can be easily implemented with 2 inputs multiplexer, which is showed in Figure 2.22.

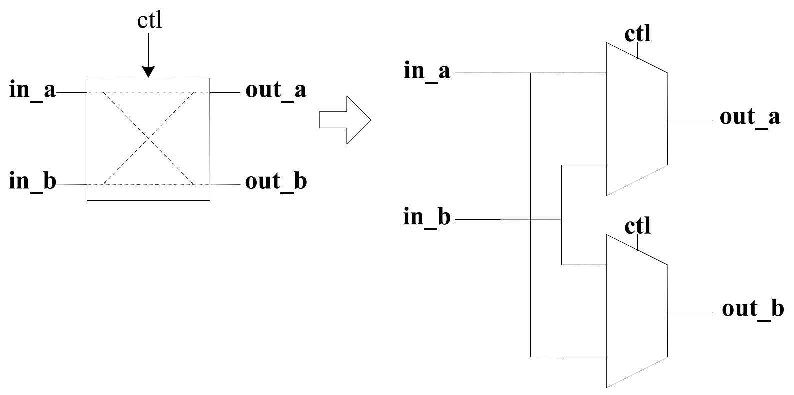


Figure 2.22 Implementation of 2×2 switch unit

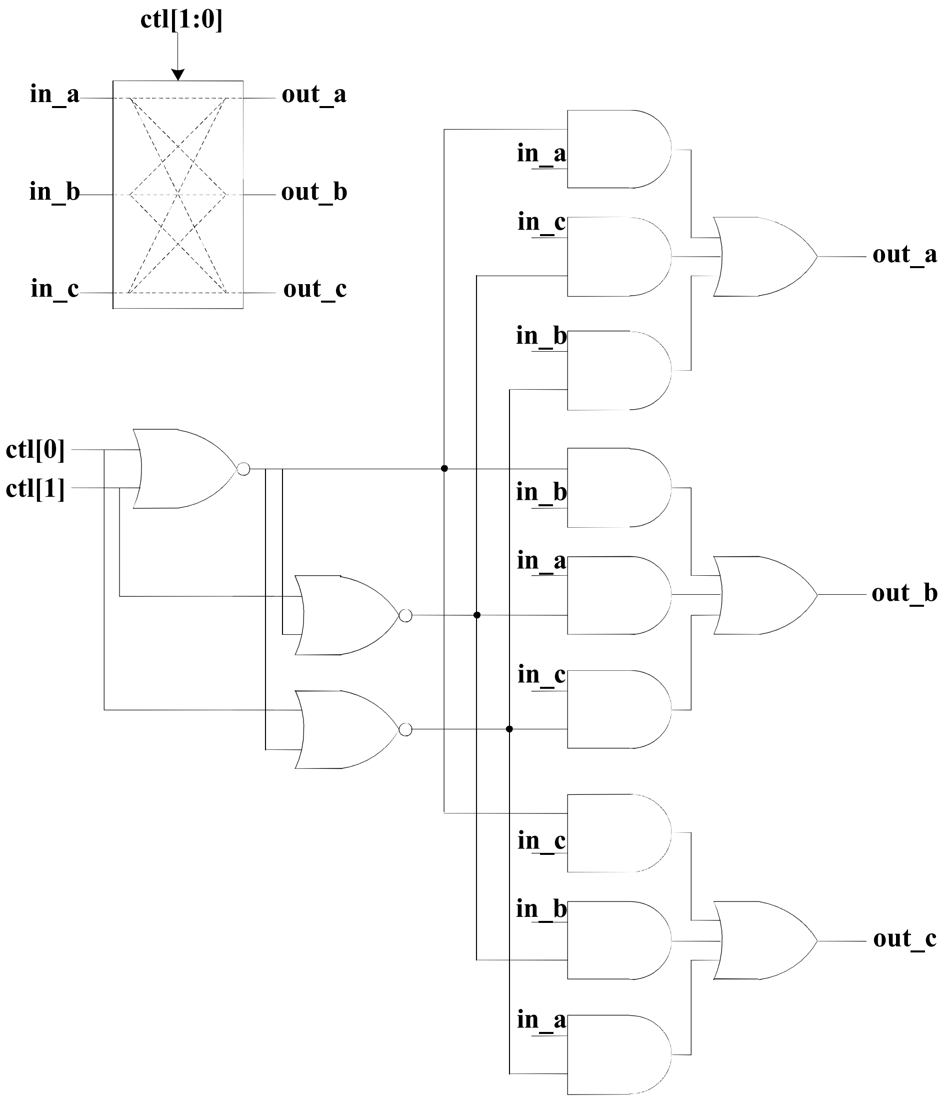


Figure 2.23 Implementation of 3×3 switch unit.

The 3×3 switch unit should realize the cyclic shift of the 3 input data. Unlike the 3×3 switch network proposed in [42], the 3×3 switch unit in our proposed network only performs 3 situations of permutation showed in Figure 2.16 (a), which make the control signals need only 2 bits. Figure 2.23 shows the hardware implementation of this 3×3 switch unit.

In the same way, the 5×5 switch unit could be implemented as shown in Figure 2.24.

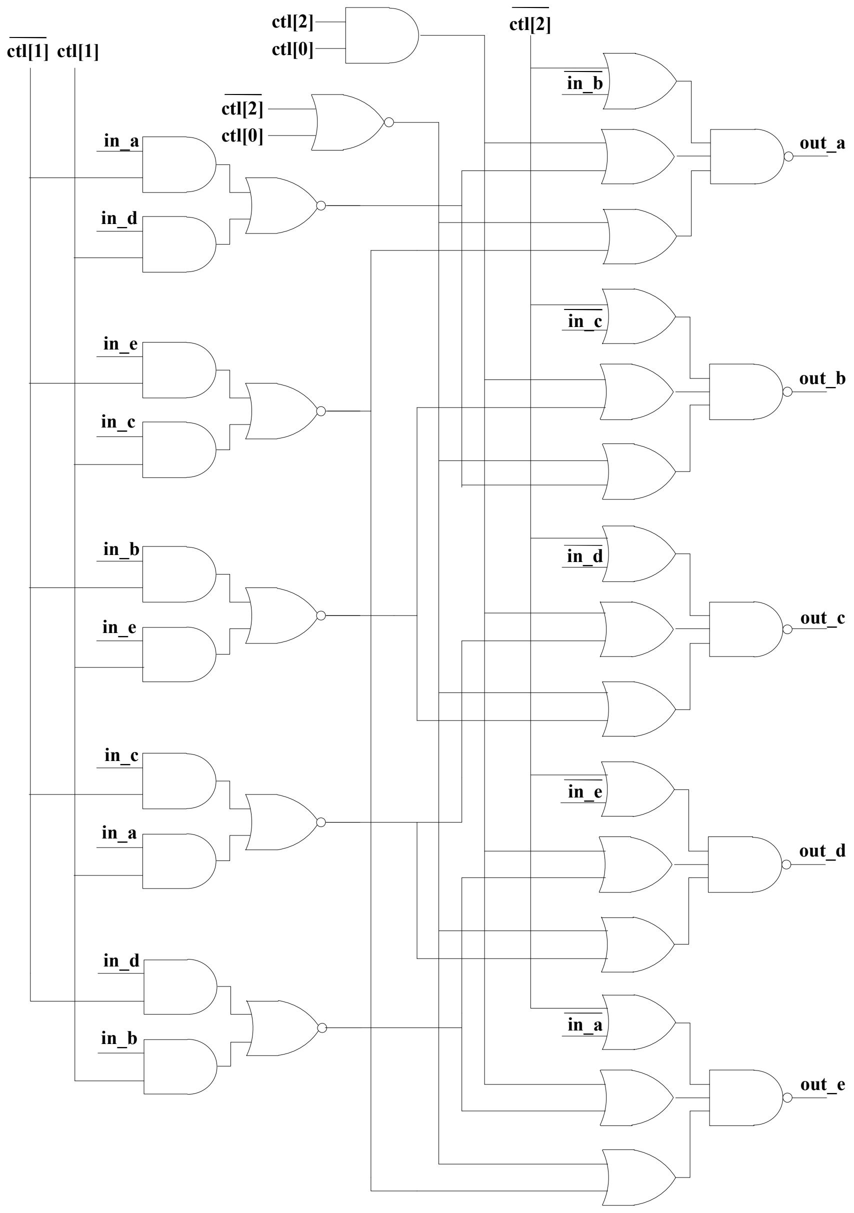

$\overline{\text{ctl[1]}}$
ctl[1]
ctl[2]
ctl[0]
$\overline{\text{ctl[2]}}$
ctl[0]
$\overline{\text{ctl[2]}}$
in_a
in_d
in_e
in_c
in_b
in_e
in_c
in_a
in_d
in_b
$\overline{\text{in\_b}}$
$\overline{\text{in\_c}}$
$\overline{\text{in\_d}}$
$\overline{\text{in\_e}}$
$\overline{\text{in\_a}}$
out_a
out_b
out_c
out_d
out_e

Figure 2.24 Implementation of 5×5 switch unit.

Although these design could achieve best performance, it is impractical to design all the $p \times p$ switch unit for the prime number p is larger than 11. Thus for large prime number, it is recommended to choose the size which is slightly larger than the required prime number which could be factored to multiplication of 2, 3, 5 and 7.

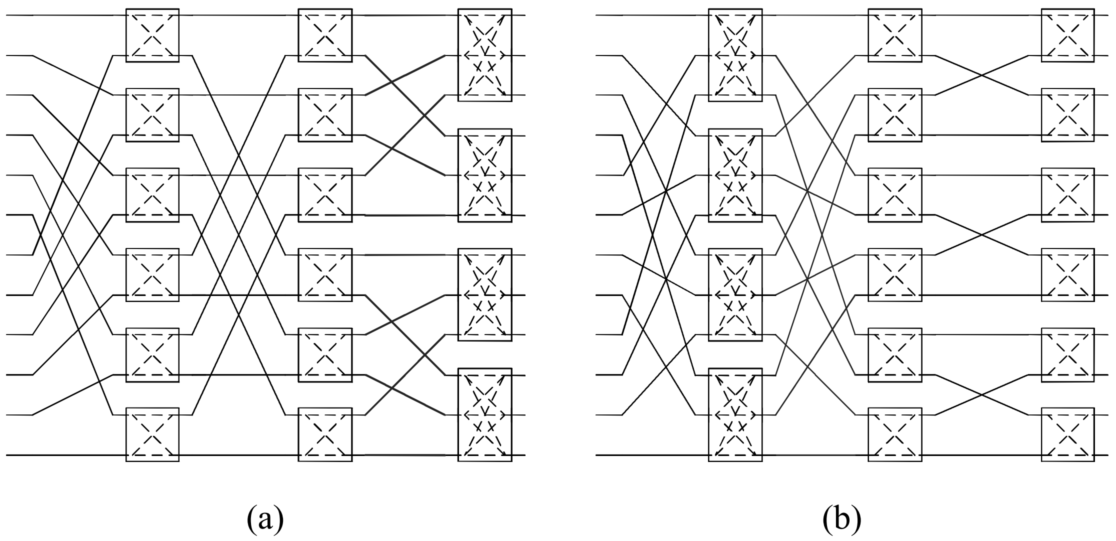


Figure 2.25 Shuffle connections difference of 12×12 GPN

The maximum block size defined in WiMAX is 96, which can be divided into 96=25×3. Thus, there is one stage which consists of 3×3 switch units. As shown in Figure 2.25, the 12×12 GPN shows the two approaches of GPN, (a) is put the 3×3 switch units at the last stage, while (b) is use this stage as the first stage. Obviously, Figure 2.25 (b) is constructed as the rule described in section 3.2, thus its control signal keep the Algorithm II. On the other hand, Figure 2.25 (a) uses different stage order. As a result, the shuffle connections between stage 2 and stage 3 are also different so that the control patterns need small modification. Both of these two architectures could be used for WiMAX. The difference of them relies on the parallelism character.

According to the partial parallel QC-LDPC decoder architecture, the number of working PE is the number of expansion factor of the PCM. That is in LDPC decoder for WiMAX [46], if the expansion factor is 24, the usage of PEs is only 25%, which would waste hardware and lead to low throughput of LDPC decoder. In order to use those idle PEs, the permutation network should permute other frames of data concurrently and feed them into the proper PEs. Therefore, we explore the parallel property of the generic permutation network.

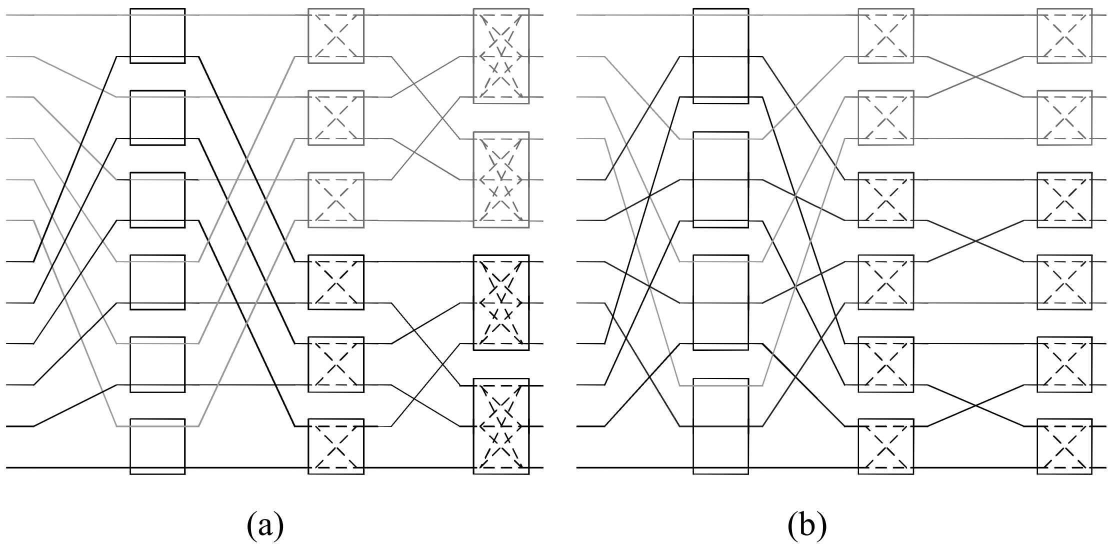


Figure 2.26 Parallelism difference of 12×12 GPN

As shown in Figure 2.26, the shuffle connection pattern ensure that the input sequence would remain the same when all the control signals in the first stage are set to be zero. Since there is no interconnection among the sub-networks, the data in the sub-networks would be cyclically shifted independently if separate control signals are given.

In General, since the stage order can be freely changed in GPN, the parallel character is determined by the switch unit size of the first several stages. Giving that input number *IN*=*mp* and the first stage has *m* $p\times p$ switch units, the GPN would show *p* way parallelism at the second stage and more parallelism at the following stages. Thus, the parallelism character could be chosen according to the actual application.

For the case shown in Figure 2.26, the difference of the two approaches is that (a) would realize 2 way parallelism at the second stage and 4 way parallelism at the last stage, while (b) can achieve 3 way parallelism at the second stage and the 6 way parallelism at the last stage is just cross-bar switch. We choose the (a) approach in the implementation for WiMAX although (b) is more typical. When the *IN* is equal or less than 48, such as the block length 28, …, 48, defined by WiMAX, the proposed network could perform two cyclic shifts in parallel with a small justification in the controller. Moreover, when the *IN* is 24, 4 groups of data could be cyclically shifted simultaneously. If we set all the control signals of the first stage to bar state, the first 48 input data are fed into the upper sub-network and the rest data go into the lower sub-network. Therefore, the similar shift algorithm and controller design could be used for both of the two sub-networks. The only difference is that there are two *SN* fed into the control signal generator. In the same way, for the 4-parallel way design, we could just set all the switches in stage 1 and stage 2 to bar state, and give the controller 4 *SNs*.

For the control signal generator, according to Algorithm II, the control signals of stage 1 to stage 5 are determined by SN%2n-i, the control signals of stage 6 are determined by SN%3, where "%" represent the modulus operation. The last multiplexer stage's control signals are also determined by SN. Consequently, we just feed the required low bits of SN into every stage control signal generator (SCSG). SCSG could be directly realized by corresponding decoder introduced in previous section.

Using SMIC 90nm 1.0V library, the proposed GPN can be implemented with gate count of 18.3k at the frequency of 600 MHz. Table 2.4 summarizes comparison among the proposed permutation network and existing networks for WiMAX, in which the frequency are normalized to the 90nm process. Compared with [41], [42] which are based on Benes network and can fully support all the IN, the frequency of our proposal could be much higher. [43] is also based on Benes network and make optimization according to WiMAX standard, which lead to the input number can only be 24 to 96 with increment of 4. [39] is also based on barrel shifter and can support WiMAX and WiFi standard, which make the number of network size becoming 22. However, our proposal can accommodate all the IN within the maximum value. In addition, only [39] could realize two parallelism when the IN is small, but it cost too much for parallel design. Our proposed network can use little hardware cost to realize the two and four parallelism because of the high symmetry property of GPN. Therefore, the proposed network can achieve the best tradeoff between hardware cost and permutation efficiency.

The maximum block size defined in WiFi is 81, which can be divided into 81=3×3×3. It is certain that the 96×96 network described above can be used for this standard. However, in practical application most of the LDPC decoder for WiFi would not need to be compatible with WiMAX standard. As viewed from this perspective, 81×81 permutation network is the most efficient choice for WiFi standard. Thus, the 81×81 GPN is composed 4 stages of 3×3 switch units. Figure 2.27 shows the 27×27 GPN. The 81×81 GPN could be constructed by stacking 3 this 27×27 GPN and adding a front stage including 27 3×3 switch units.

For the synthesis result of SMIC 90nm 1.0V library, this 81×81 permutation network can be implemented with gate count of 10.9k at the frequency of 800 MHz. Compared with the 96×96 permutation network, it saves 40.4% hardware cost and improve 33.3% timing performance. It is to be noted that the hardware reduction exceeds the ratio which is linearly correlated with the input number. This is because that the number of control signals for the 81×81 permutation network is 216, while the 96×96 network needs 304 control signals, which make the controller part cost greatly decreases.

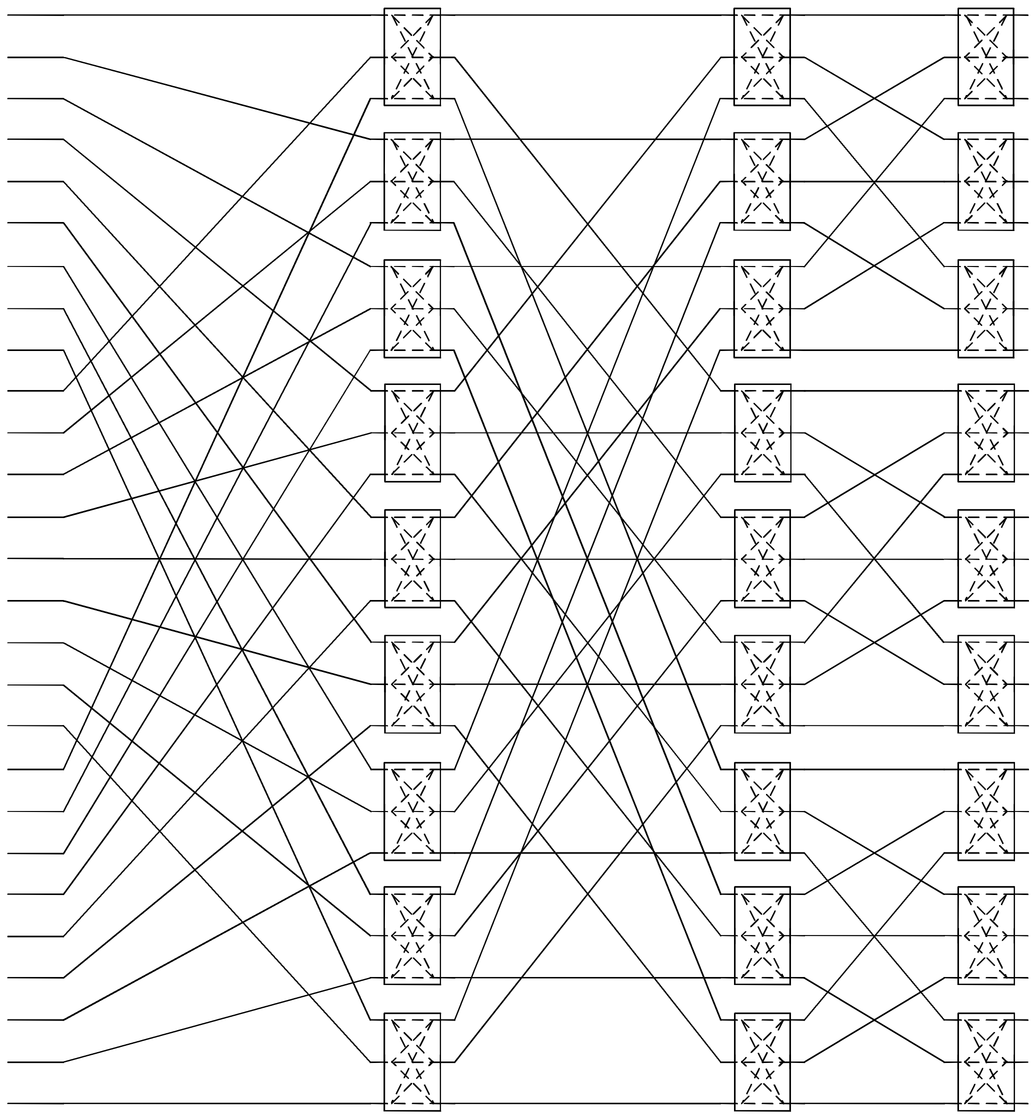


Figure 2.27 Structure of 27×27 GPN

2.5 Conclusion

In this section, we propose the generic permutation network (GPN) for the QC-LDPC decoder. This proposal specifies a common architecture of permutation network, which is suitable for any specified application with high hardware efficiency. Moreover, the proposed scheme could greatly reduce the latency because of much less stages and efficient control signal generating algorithm. In addition, the paper reveals the essence of parallel property of

GPN, and explores this nature for parallel cyclic shift. The implementation results for WiMAX standard and WiFi standard prove that GPN could not only reduce the hardware cost but also improve the timing performance.

3 Bit-serial Layered Scheduling based QC-LDPC Decoder Architecture

Structured QC-LDPC codes have become the ECC (error correcting code) part of many emerging wireless communication standards, among which WiMAX is one of the approved standards for the fourth-generation mobile communication system (4G). To be used in handheld terminals, the QC-LDPC decoder for WiMAX system needs to deliver high throughput with low power dissipation.

With the purpose of best tradeoff between the hardware complexity and decoding performance, the normalized min-sum algorithm or offset min-sum algorithm with the layered scheduling (also called TDMP) [13] has shown its significance for QC-LDPC codes. In the normalized min-sum algorithm, the complicated computation of Gallager function is replaced with simple minimum value finding operation at the cost of decoding performance, which is recovered by the normalized factor. The layered scheduling achieves about two times faster convergence speed than the conventional flooding scheduling (also called TPMP).

Generally the iterative decoding process of normalized min-sum layered algorithm is carrying out layer by layer. Within each layer, different designs with the corresponding scheduling give different partial parallel architectures. The architecture in [46] processes the nonzero sub-blocks block by block. As a result, the clock cycle number of one iteration is approximately equal to the nonzero sub-matrix number of parity check matrix (PCM), varying from 76 to 88 according to different code rates defined in WiMAX. The design in [49] uses two sets of processing units to improve the parallelism and processes two nonzero sub-blocks in one clock cycle. However, it does not bring double parallelism gain because of the data conflicting problem. By utilizing the nonzero sub-matrix reordering and complex bypass controlling scheme, it partially solved the data conflicting problem, which reduces the number of clock cycles for each iteration to 48~54.

Compared with the previous designs, this work changes the block by block scheduling of layered algorithm, which achieves higher parallel architecture. The key schemes of this architecture contain three aspects:

1) It is full parallel in each layer, which is named as full parallel layered decoding. All messages within one layer are calculated and updated simultaneously. In order to avoid the interconnection problem, the arithmetic units and permutation network are designed as 1-bit form, which make the messages be transferred and updated bit by bit. With 6-bit quantization, the number of clock cycles for each layer is 6, and each iteration needs 24 to 72 clock cycles for different code rates;

2) The parallelism is improved furthermore by dedicated PCM reordering and the two-layer concurrent processing for low code rates. The clock cycles are finally reduced to 24~48.

3) Due to the high parallelism, all the messages are stored in registers, not plenty of small and inefficient memory banks. The power increasing from using registers is eliminated by reducing operating frequency and clock gating. Moreover, the variable node (VN) messages and the a-posterior probability (APP) messages share the same storing cells, which saves at least 22.2% memory bits than the previous works.

3.1 Block diagram of the decoder architecture

Figure 3.1 shows the block diagram of the proposed QC-LDPC decoder architecture. The 96 processing units (PUs) are the kernel of the decoder, which is used to process the variable node and check node operations for one layer by updating messages from LSB to MSB serially. Permutation network (PN) contains 42 one-bit 96×96 cyclic shifters, in which PN A is used to shift 21 blocks of messages according to the base matrix (no shift needed for the rightmost 3 block columns) and PN B is designed to shift overflow bit and hard decision bit. For the two-layer concurrent processing, the cross-bar switch array (CBSA) in PN is implemented to divide 24 blocks of messages into 2 groups.

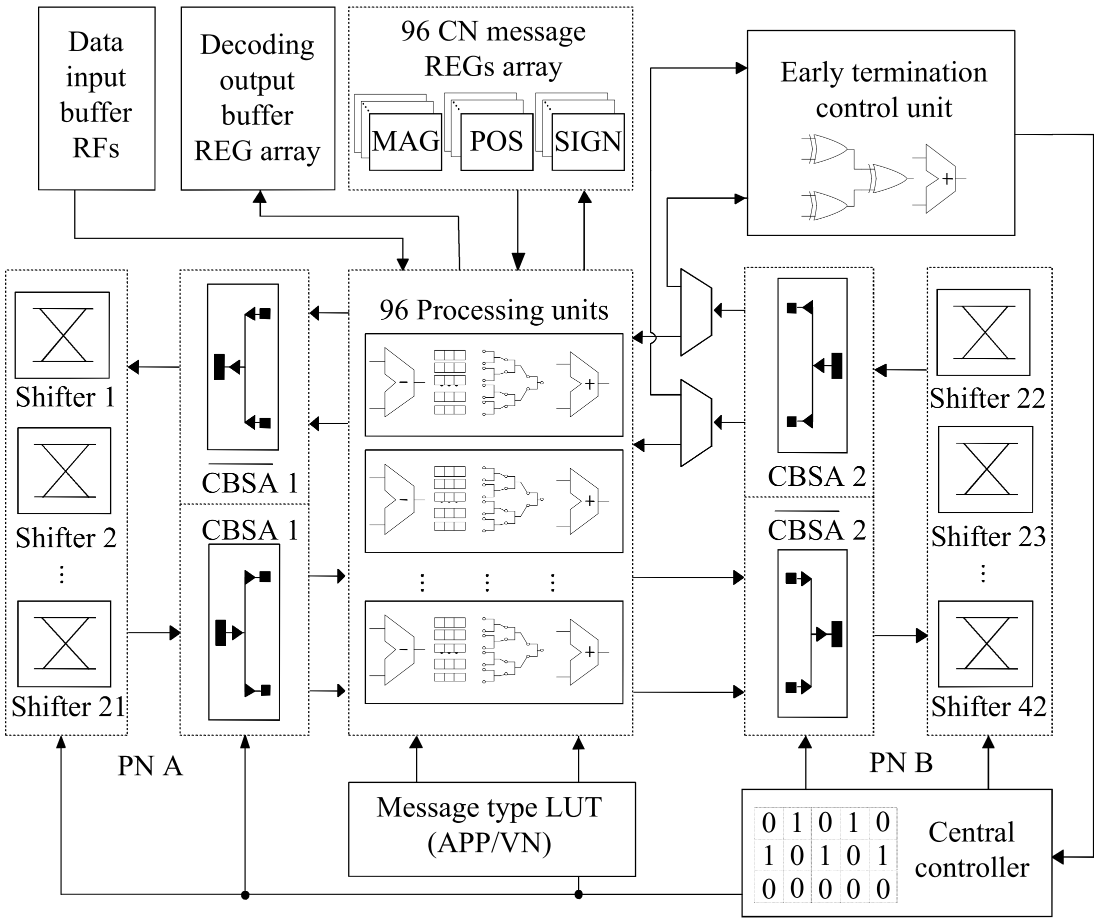


Figure 3.1 Block diagram of the architecture

3.2 Quantization consideration

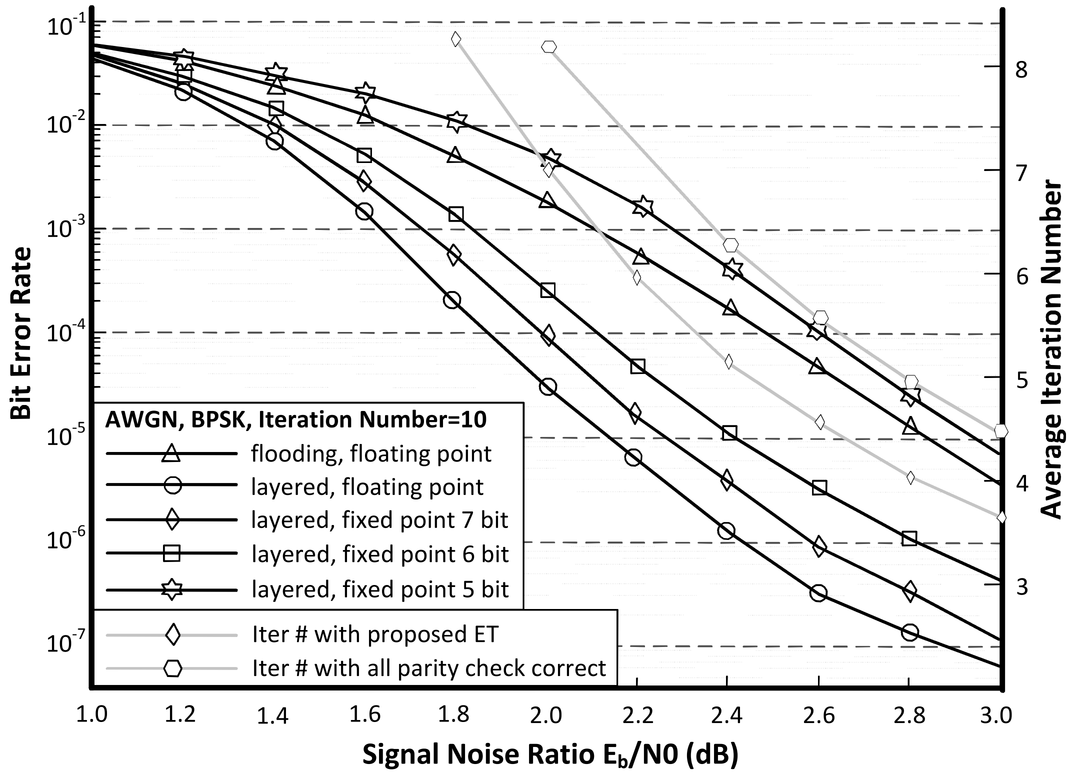


Figure 3.2 BER performance and average iteration number

As the Figure 3.2 shows, the layered algorithm converges much faster than the flooding algorithm under the floating point condition. However, the 5-bit quantization of layered decoding brings about 0.8dB performance loss at the 10^{-5} BER, which is even worse than the flooding decoding. Conversely, the 6-bit and 7-bit quantization leads to around 0.2dB and 0.1dB loss respectively. Thus, the 6-bit quantization is adopted for best trade-off between decoding performance and hardware complexity.

3.3 Early Termination

In the layered decoding process, different code words cost different iterations to be corrected. The early termination (ET) scheme is used to stop the decoding process when the code word has already been corrected. The natural approach of ET is calculating all the parity check equations. In this design, the ET control unit terminates the decoding based on the stopping criterion which was proposed in our previous work [50]. The basic idea of this criterion is errors in the information part of code word are corrected faster than the errors in the parity part. When all the information bits are correct, the decoding process can be stopped. Figure 3.2 shows the average iteration number (AIN) of the proposed ET scheme. Compared with the all parity check correcting situation, it reduces 1 AIN on average.

3.4 Basic matrix reordering for increasing throughput

In layered decoding process, the processing order of the layers does not affect the decoding performance, which enables the row-wise reordering of basic matrix. Furthermore, since the processing at each layer is minimum value finding, the column order also does not affect the result, which means the column-wise reordering is also possible. As defined in the WiMAX standard, basic matrix reordering can be applied to the case of 1/2 and 2/3B code rate so as to improve the throughput. Figure 3.3 shows the original basic matrix and the reordered basic matrix. Taking the 1/2 code rate as an example, after the reordering, the 12 rows are divided into 6 groups with no data dependency, which results in 6 merged layers and the throughput is double. Utilizing this basic matrix reordering scheme, the numbers of clock cycles per iteration are reduce to 36 and 24 for code rate 1/2 and 2/3B respectively.

Original basic matrix

-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	27	-1	-1	-1	22	79	9	-1	-1	-1	12	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	24	22	81	-1	33	-1	-1	-1	0	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1
61	-1	47	-1	-1	-1	-1	-1	65	25	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1
-1	-1	39	-1	-1	-1	84	-1	-1	41	72	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	46	40	-1	82	-1	-1	-1	79	0	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1
-1	-1	95	53	-1	-1	-1	-1	-1	14	18	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1
-1	11	73	-1	-1	-1	2	-1	-1	47	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1
12	-1	-1	-1	83	24	-1	43	-1	-1	-1	51	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1
-1	-1	-1	-1	-1	94	-1	59	-1	-1	70	72	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1
-1	-1	7	65	-1	-1	-1	-1	39	49	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0
43	-1	-1	-1	-1	66	-1	41	-1	-1	-1	26	7	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0

Reordered basic matrix

-1	94	73	22	24	-1	81	55	33	83	-1	7	0	-1	0	0	0	0	-1	-1	-1	-1	-1	-1
43	-1	39	-1	-1	84	66	-1	41	41	72	7	26	0	-1	-1	-1	-1	0	-1	-1	-1	-1	0
12	-1	95	83	53	-1	24	-1	43	14	18	-1	51	-1	-1	-1	-1	0	-1	0	0	-1	0	-1
-1	27	7	-1	65	79	22	39	9	49	-1	-1	12	-1	0	-1	0	-1	-1	-1	-1	0	-1	0
61	-1	47	46	-1	-1	40	65	82	25	-1	0	79	0	-1	0	-1	0	0	-1	-1	-1	-1	-1
-1	11	73	-1	-1	2	94	-1	59	47	70	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0	-1

(a) Code rate 1/2

Original basic matrix

2	-1	19	-1	47	-1	48	-1	36	-1	82	-1	47	-1	15	-1	95	0	-1	-1	-1	-1	-1	-1
-1	69	-1	88	-1	33	-1	3	-1	16	-1	37	-1	40	-1	48	-1	0	0	-1	-1	-1	-1	-1
10	-1	86	-1	62	-1	28	-1	85	-1	16	-1	34	-1	73	-1	-1	-1	0	0	-1	-1	-1	-1
-1	28	-1	32	-1	81	-1	27	-1	88	-1	5	-1	56	-1	37	-1	-1	-1	0	0	-1	-1	-1
23	-1	29	-1	15	-1	30	-1	66	-1	24	-1	50	-1	62	-1	-1	-1	-1	-1	0	0	-1	-1
-1	30	-1	65	-1	54	-1	14	-1	0	-1	30	-1	74	-1	0	-1	-1	-1	-1	-1	0	0	-1
32	-1	0	-1	15	-1	56	-1	85	-1	5	-1	6	-1	52	-1	0	-1	-1	-1	-1	-1	0	0
-1	0	-1	47	-1	13	-1	61	-1	84	-1	55	-1	78	-1	41	95	-1	-1	-1	-1	-1	-1	0

Reordered basic matrix

2	28	19	32	47	81	48	27	36	88	82	5	47	56	15	37	95	-1	0	0	0	-1	-1	-1
32	69	0	88	15	33	56	3	85	16	5	37	6	40	52	48	0	0	0	-1	-1	0	-1	0
23	0	29	47	15	13	30	61	66	84	24	55	50	78	62	41	95	-1	-1	-1	0	-1	0	0
10	30	86	65	62	54	28	14	85	0	16	30	34	74	73	0	-1	0	-1	0	-1	0	0	-1

(b) Code rate 2/3B

Figure 3.3 Basic matrix reordering of code rate 1/2 and 2/3B

3.5 Full layer scheduling with bit-serial representation

Figure 3.3 shows the structure of the processing unit. At each clock cycle, 24 messages (APP or VN) stored in the APP/VN REG array are updated one bit from LSB to MSB by relevant APP/VN REG bit updaters. According to the message type LUT, the received APP bit corresponding to an inactive block (-1 in the base matrix) in current layer will be directly stored in the register array. Otherwise, it will be subtracted from the bit of CN message and a borrow bit caused by last update to generate the VN bit. Similarly, 24 APP bit updaters generate or bypass 24 new APP bits which will be transmitted by the PN A. Thus, the VN message updating of current layer and the APP message updating of previous layer are processed at the same time. Based on the 6-bit quantization, each layer processing occupies 6 clock cycles. As shown in the timing flow chart, in the first 5 clock cycles of current layer, least significant 5 bits of messages are updated and hard decisions are transmitted to ET unit for analyzing via PN B. At the 6th clock cycle, PN B is used to transmit the APP bit updater generated overflow signals. Based on them and the overflow signals generated by APP/CN REG bit updaters, the VN messages are corrected. Then the VN messages are converted from two's complement to sign-magnitude form by the 2C-SM unit and fed into the min-sorter.

The min-sorter employs the simplest tree structure to find the first and second minimum values among VN messages with 2 clock cycles. After the previous clock cycle for the first minimum value and its position, it finds the second minimum value by setting the first minimum value as the maximum value (11111_2). Compared with the one-clock design, this structure could reduce the critical path and save about 150k gate count. On the other hand, this two-clock design would not increase the clock cycles per layer. The first minimum value is found at the last clock cycle of current layer, and the second minimum value finding is arranged to the next layer.

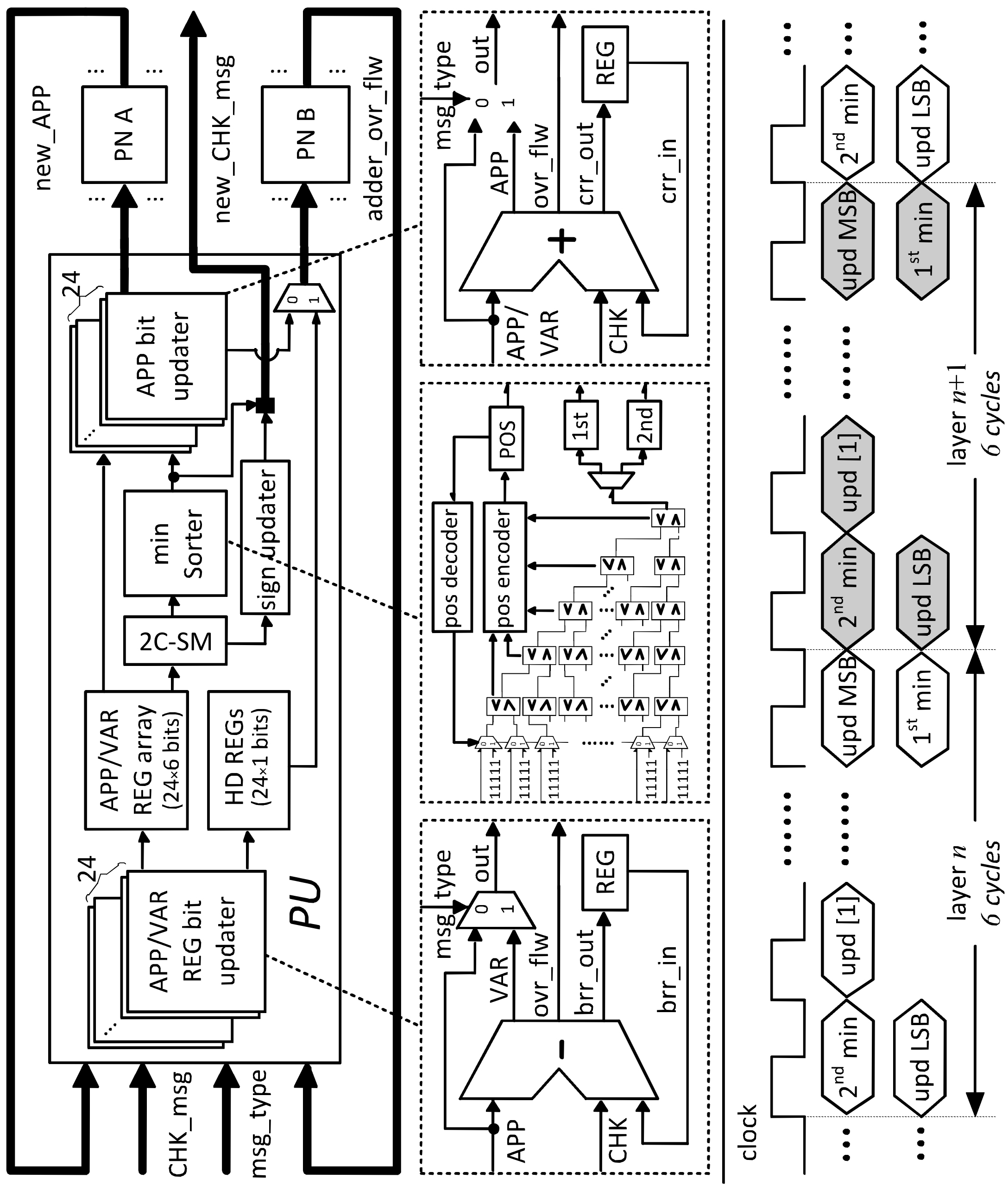


Figure 3.3 Architecture of process unit and full layer scheduling

3.6 Design flow and test consideration

As shown in Figure 3.4, the whole design flow before fabrication mainly contains six steps: RTL simulation, synthesis, formal verification, FPR (Floorplan, Placement and Routing), DRC/LVS and post-layout simulation.

After the RTL code was finished and passed the functional check with Mentor Modelsim SE, it was synthesized by the Synopsys Design Complier. Before floorplan, formal verification was carried out by using Standard Delay Format (SDF) file and netlist to check whether the netlist generated by synthesis was functionally correct or not. This step was still using Modelsim SE. Then, the netlist and Synopsys Design Constraints (SDC) file were fed into Synopsys IC Compiler (ICC), and went through the floorplan, placement and routing step by step. The GDS file produced by FPR was then loaded in Cadence Virtuoso integrated with Mentor Calibre for design rule check (DRC) and layout versus schematic (LVS) procedure. Most DRC errors are fixed by iterative FPR process, the rest are fixed manually in Virtuoso. The final post-layout simulation also ran in Modelsim SE in order to confirm function and timing.

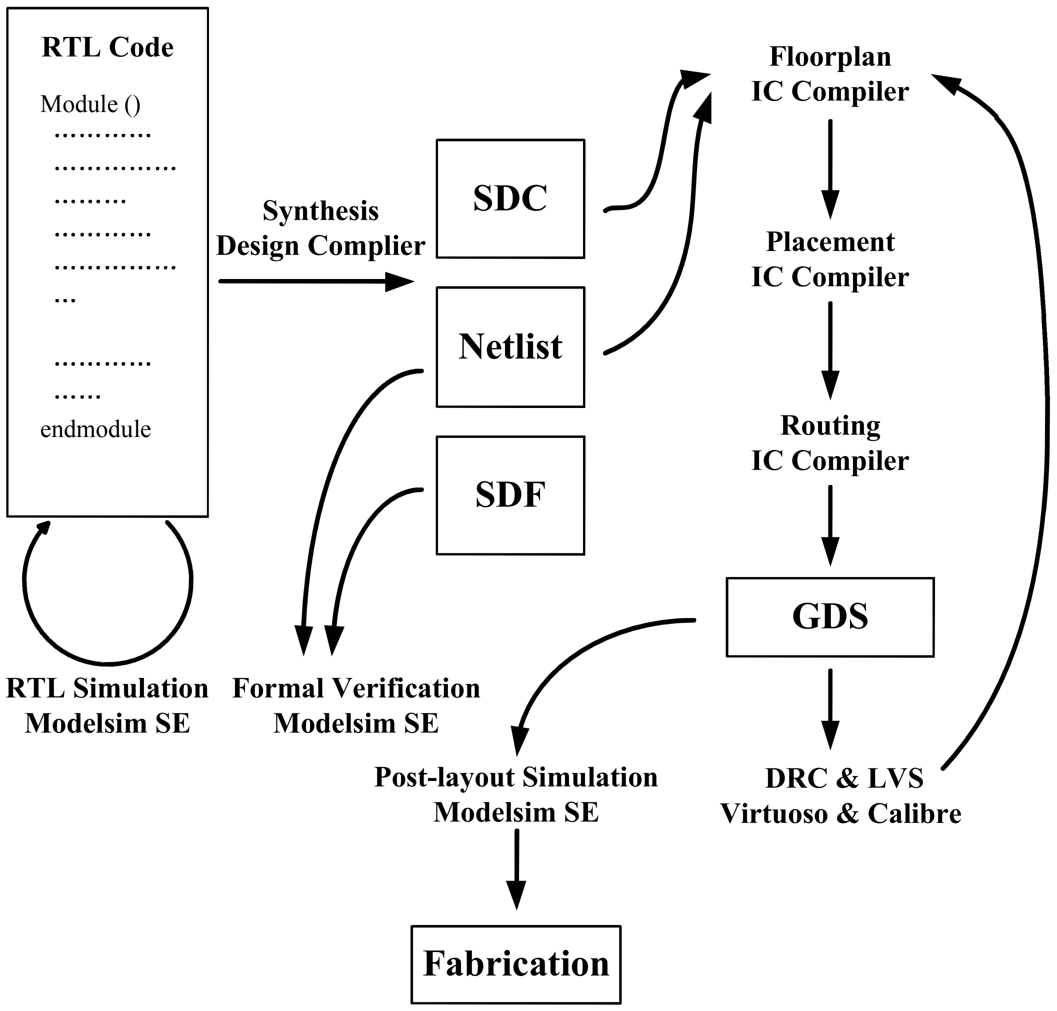


Figure 3.4 Design flow and EDA tools

The test system is shown in Figure 3.5. The Altera DE3-150 development and education board is used as the mother board. The QFP128 packaged decoder chip is mounted on the dedicated sub-board, which communicates with mother board through the High Speed Terasic Connector (HSTC) socket. The FPGA board mainly provides three functions:

1) The simple environment built with the embedded NIOS II in the Stratix III FPGA is used for communication between PC and the test system, which avoids frequent modifying test RTL code and downloading test files to FPGA.

2) AWGN noise generator is implemented in FPGA. The SNR could be adjusted for various condition testing.

3) The clock signal of decoder is provided by the FPGA built-in PLL.

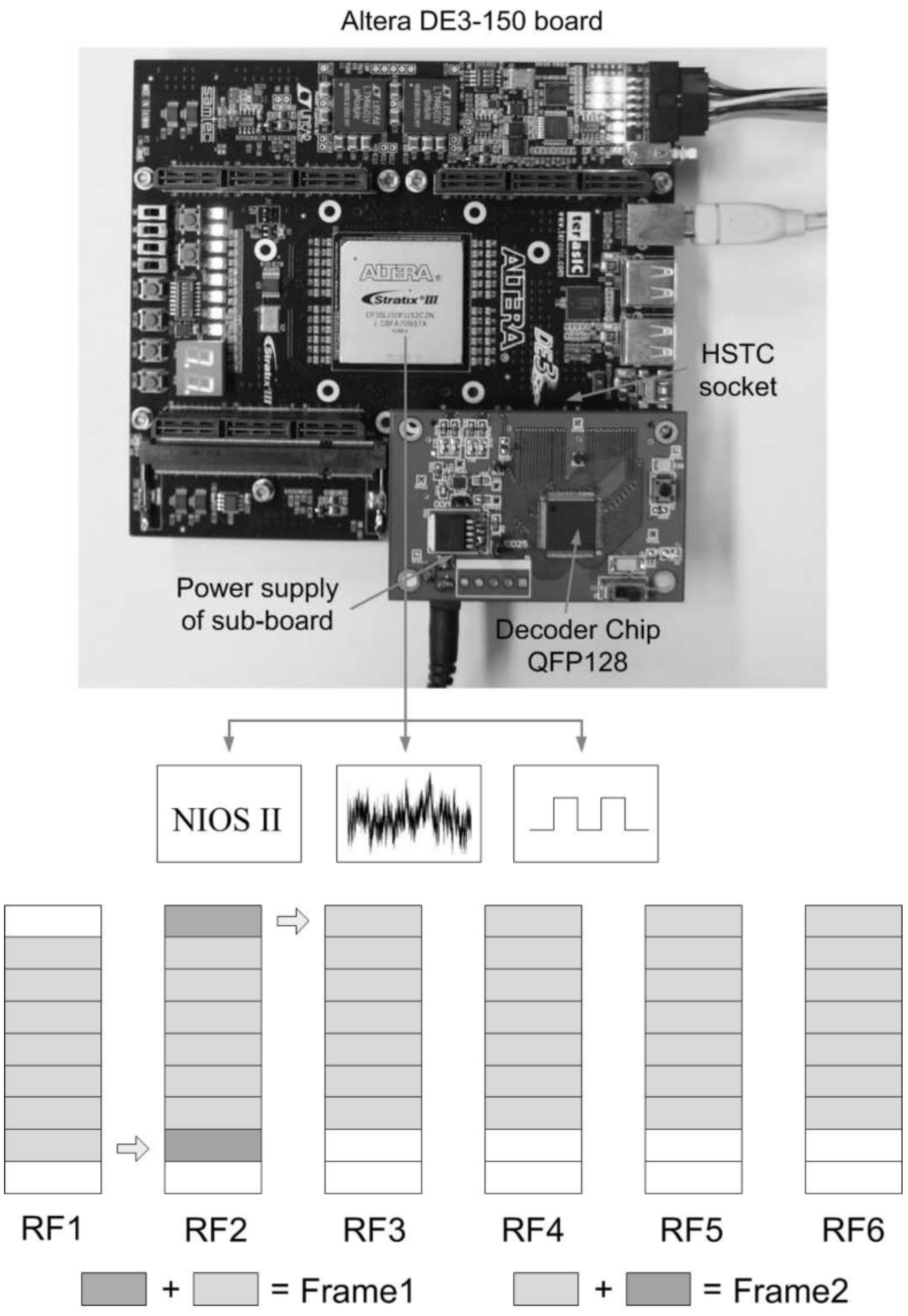


Figure 3.5 Test system and frame updating

During the testing process, the encoded and BPSK modulated code words plus the AWGN noise are quantized and fed into decoder chip through HSTC socket. The decoded bits are transmitted back to FPGA for verification, the results of which are fed back to PC.

Since the input pins of decoder are limited, in order to test the function and the power of the decoder, two operating modes are designed in the test strategy.

1) The first is the functional testing mode, in which the input buffers are previously filled before decoding. When all the frames in the input buffer are completely decoded, another input buffering process is started. In this mode, the chip is idle at most of the time since input pins are only 24, which take many clock cycles for input buffering. Thus, various encoded code frames are used to test the function of the decoder in this mode.

2) The second is the power testing mode, in which the decoder should run incessantly. It could be realized by decoding the same code frame, but the power measurement would not be accurate.

Utilizing the property that the encoded code of all zero code is still all zero, we can obtain a new frame by shifting the code frame head address. As the example shown in Figure 3.7, each code frame consists of 144 blocks in the 6 RF input buffers. After decoding of frame 1, frame 2 is obtained by shifting the frame head to the RF 3 and carrying the same operation in the frame end. The released block of RF 2 will be loaded new code in the next decoding process. Through this frame updating scheme, the testing carries out at continual working state, which avoids the inaccuracy of power measurement with only one code frame.

3.7 Implementation and comparison

This design is fabricated in SMIC 65nm low leakage LVT 1P10M CMOS. The fabricated die is packaged in QFP128, with 52 power and ground pins, 63 signal IO pins and 13 unused pins. Figure 3.6 shows the micrograph of the fabricated chip. The die area with pads is 5.46 mm^2 (2.1mm×2.6mm), and the core area is 3.36 mm^2 (1.6mm×2.1mm) The 96 PUs are the major part of the decoder, which are flattened in the placement and routing. The registers which store APP and CN messages are combined with associate PUs. This distribution guarantees that this design avoids the inter-connection complexity problem. The maximum routing metal layers is 7, and the area utilization ratio achieves 83.1%.

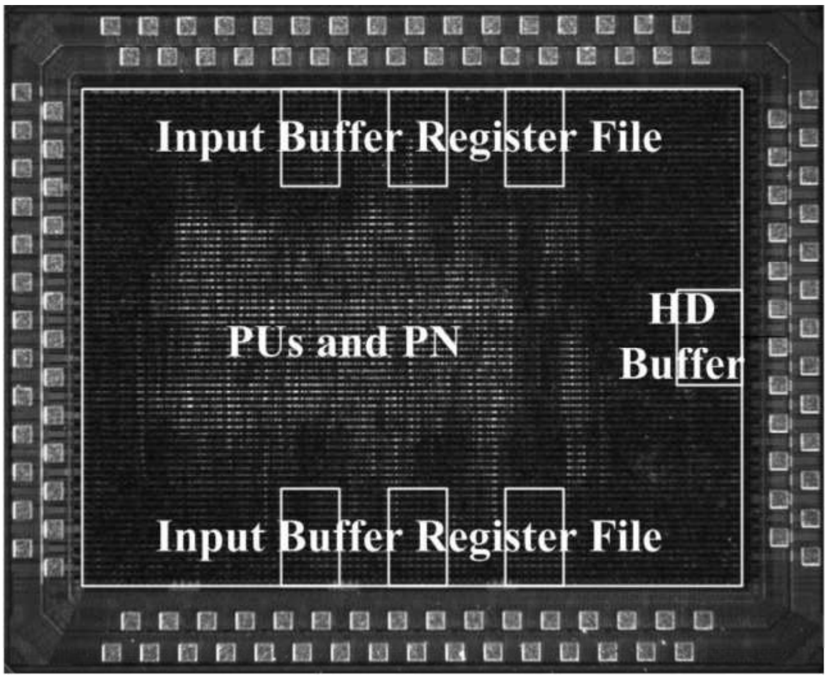


Figure 3.6 Chip micrograph

Table 3.1 summarizes the performance of this ASIC and the comparison with the other 3 published LDPC decoders [49, 51, 52] for WiMAX standard. Compared with previous works, the memory bits are reduced over 22.2%. Assuming each memory bit occupies about 6.6 gates which is given in [51], the total equivalent gate count of this design and previous works are almost the same. By supplying 1.2V, this design achieves the throughput 1056Mbps at the operating frequency 110MHz with the measured power 115mW and the power efficiency 10.9pJ/bit/iteration. In order to compare proposed architecture and the previous designs in the same process level, the normalizing factor of power is obtained by synthesizing this design with different libraries. Due to power consuming property of the LVT process, the ratio of 130nm process and 65nm LVT process is around 2.

TABLE 3.1 COMPARISONS OF PREVIOUS DESIGNS

	This work	[49]	[51]	[52]
Technology	65nm	0.13μm	90nm	0.13μm
Supply voltage	1.2V	1.2V	1.0V	1.2V
Code length	576~2304			
Code rate	1/2, 2/3A, 2/3B, 3/4A, 3/4B, 5/6			1/2
Cycle# /iteration	24~48	48~54	~160	~350
Logic gate count	597k	470k	380k	420k
Memory bits	56,448	72,522	89,856	76,800
Eq. gate count	968k	946k	970k	924k
Frequency	110MHz	214MHz	150MHz	83.3MHz
Iteration number	10	10	20	2~8
Throughput(Mbps)	1056	955	105	111
Power(mW)	115	397	264	52
Normalized power	230	397	484	52
Nor. Power Eff. (pJ/bit/iteration)	21.8	42	230	216

Since this architecture contains much more registers than the SRAM based architecture, it is quite efficient to adopt clock gating in this design. The clock gating scheme is not implemented in the fabrication. Based on the simulation result, power reduction ratio of clock gating is at least 26.8%. As Figure 3.7 shows, compared with state-of-art work [49] which also employed clock gating, this design finally realizes 63.6% reduction in power efficiency.

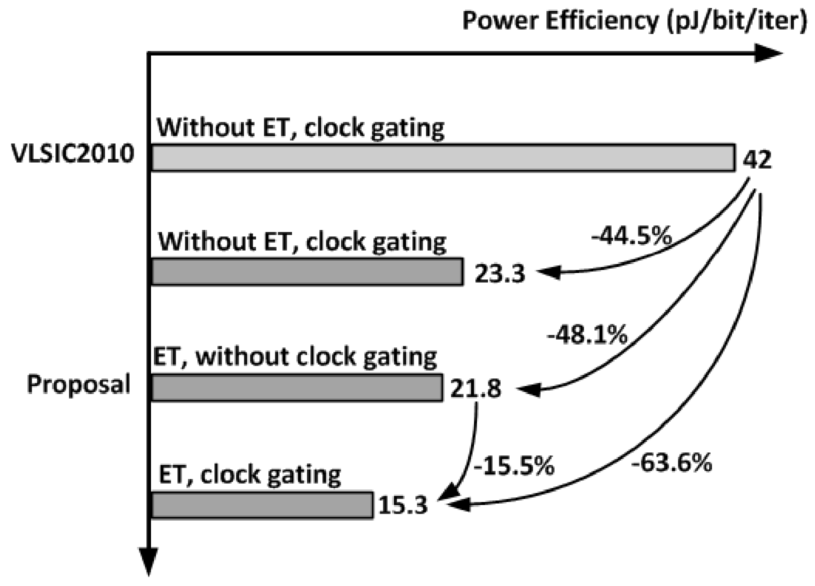


Figure 3.7 Power efficiency comparison with state-of-art work

3.8 Conclusion

This section introduces the proposed fully parallel layered scheduling architecture and presents the QC-LDPC decoder chip for WiMAX standard, which achieves higher parallelism than previous designs. This architecture adopts three key schemes in the implementation: the first is calculating and updating all the messages within one layer simultaneously, wherein the arithmetic units and permutation network are designed as 1-bit form so as to avoid interconnection problem; the second is improve the parallelism with dedicated PCM reordering and the two-layer concurrent processing for low code rates, with which the clock cycles of one iteration are reduced to 24~48; the third is storing all the messages in registers and sharing the storing cells of VN and APP. Based on these schemes, this design achieves over 1Gbps throughput. The power reduces 42.1% and the power efficiency reduces 63.6% in the normalized comparison with the state-of-art work.

4 SEMI-LAYER SCHEDULING BASED QC-LDPC DECODER ARCHITECTURE

Compared with the previous design, this architecture achieves much higher parallelism. The key schemes of this proposal contain three aspects:

1) Based on the row weight and column number of basic matrix for 6 code rates, the processing units compute 12 blocks at the same time. Compared with the clock cycles per iteration $K_q \times N_{layer}$ (K_q is the quantization bit number of message, which usually varies from 5 to 8) in bit-serial based architecture, this architecture only needs $2 \times N_{layer}$ for one iteration, which improve the parallelism furthermore. Thus, it takes 2 clock cycles to process one layer and 8-16 clock cycles for one iteration in the iterative decoding.
2) The manually inserted clock gating scheme brings great power reduction and much higher energy efficiency than previous work.

4.1 Block diagram of the decoder architecture

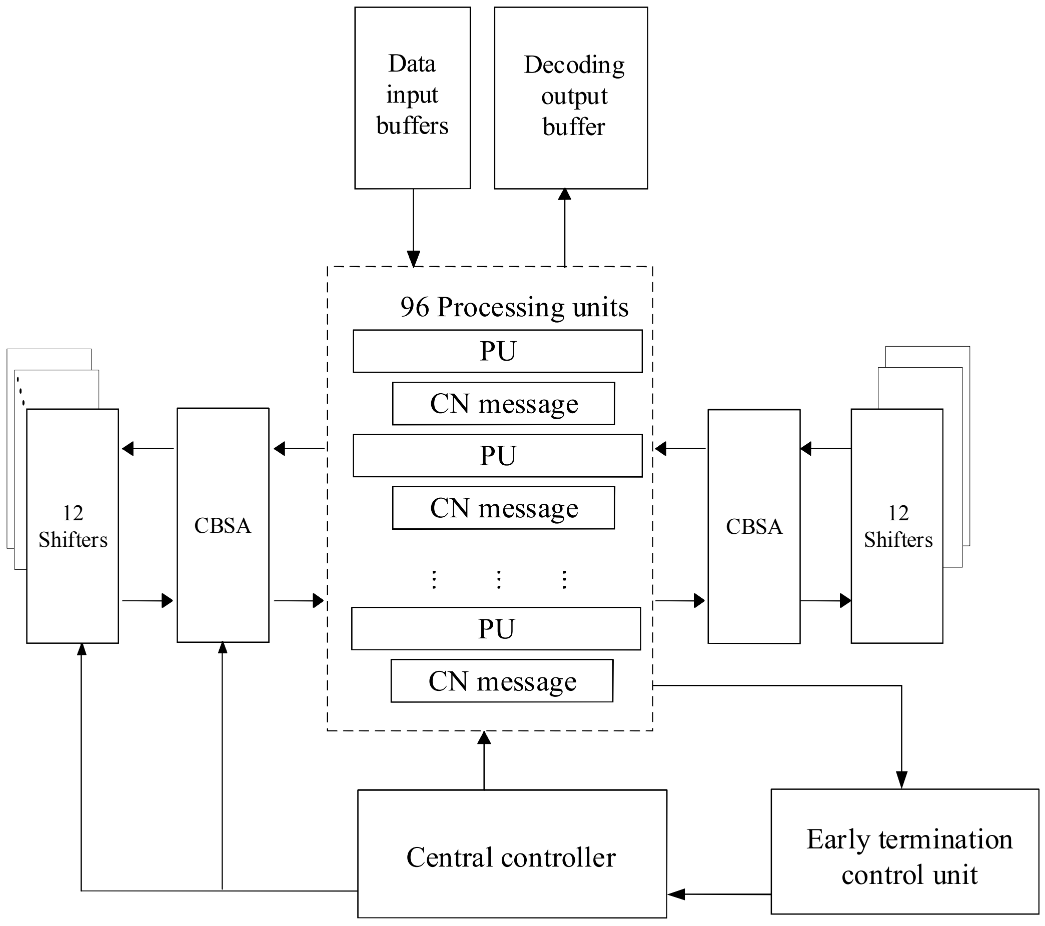


Figure 4.1 Block diagram of the semi-layer architecture

Figure 4.1 shows the block diagram of proposed semi-layer architecture. Compared with the previous design, there are two major differences:

(1) Since 12 blocks are processed at one clock, there are 12 cyclic shifters in each permutation network, wherein the bit width increases from 1 to 6 and so as the cross bar switched in CBSAs.

(2) In this design, APP and VN messages share the same storage cells in the PUs. Check node message storage cells are also arranged near the PUs, which can reduce the routing complexity. Because of the property of comparison, only the magnitude of the first and second minimum value, the sign of each check node and the position of the first minimum value should be stored. An early termination control unit terminates the decoding if certain criterion is satisfied.

4.2 Architecture of PU

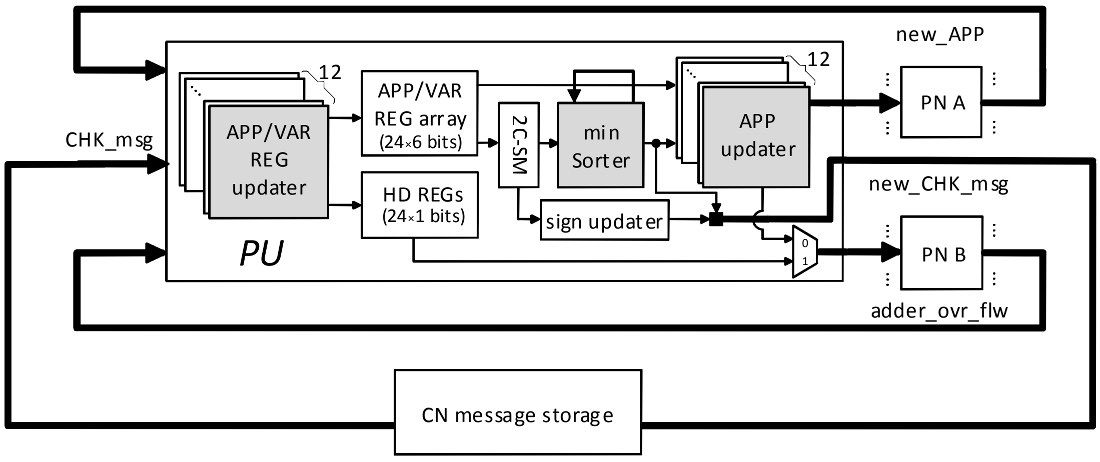


Figure 4.2 Structure of processing unit

Figure 4.2 shows the structure of processing unit in this design, wherein the different parts compared with the previous design are marked in grey.

At each clock cycle, 12 messages (APP or VN) stored in the APP/VN REG array are updated by relevant VN updaters or CN/APP updaters. According to the message type LUT, the received APP message corresponding to an inactive block (-1 in seed matrix) in current layer will be directly stored in the register array. Otherwise, it will be subtracted from the check message to generate the VN message. Similarly, 12 CN/APP updaters generate or bypass 12 new APP messages which will be transmitted by the PN. Thus, unlike bit level updaters in the previous design, all the updater in this design are composed of the 6-bit subtractors, adders and multiplexers. On the other hand, since the min-sorter finds the minimum value in half layer at one clock cycle and the total layer needs two clock cycles, there is exits the feedback path between these two clock cycles.

4.3 Clock gating

An automatic clock gating generation method through power-optimal control signal selection is proposed by Ms. Man. In this method, the control signal is selected from the signals existing in the original circuit to avoid additional hardware cost. Furthermore, optimal sharing of clock gating control by several registers is considered to minimize the overhead to insert the clock gating. Figure 4.3 shows the design flow of the proposed clock gating technique.

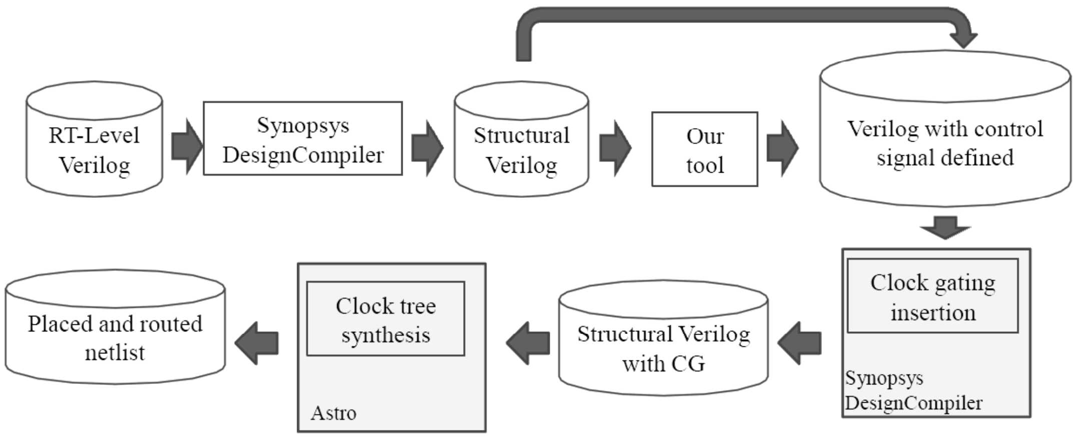


Figure 4.3 Clock gating design flow

The algorithm is carried out on the structural verilog to find out the optimum candidates for the control signals of the gated clock. In this algorithm, the sharing of one gated clock by several registers is also considered. After the candidates are decided, clock gating description is added in the verilog code and thus the structural verilog with clock gating can be generated.

We did clock gating for CBPE module. To reduce the computation time, the quantization bit is reduced to 3 bits. So there are mainly 24 3-bit registers in each PE. We generated 2 clock gating logics, each sharing by 36 bits. The results showed that the clock gating technique can achieve 85.3% power reduction for the 24 registers and 50.3% power reduction for the whole circuit of CBPE.

4.4 Implementation and comparison

This design is demonstrated with the SMIC 65nm low leakage 1P10M CMOS library. Because of using the clock gating technology, all the components including the digital logics and the registers are flattened and arranged by the EDA tools in automatic manner.

Table 4.1 shows the performance of this design and the comparison with previous design and state-of-art work [49] for WiMAX system.

With the assumption of one memory bit equals 6.5 NAND gates, the equivalent gate counts of this design shows that the hardware cost increases about 40% than the previous and the work [49].

TABLE 4.1 COMPARISON OF PREVIOUS LDPC DECODERS

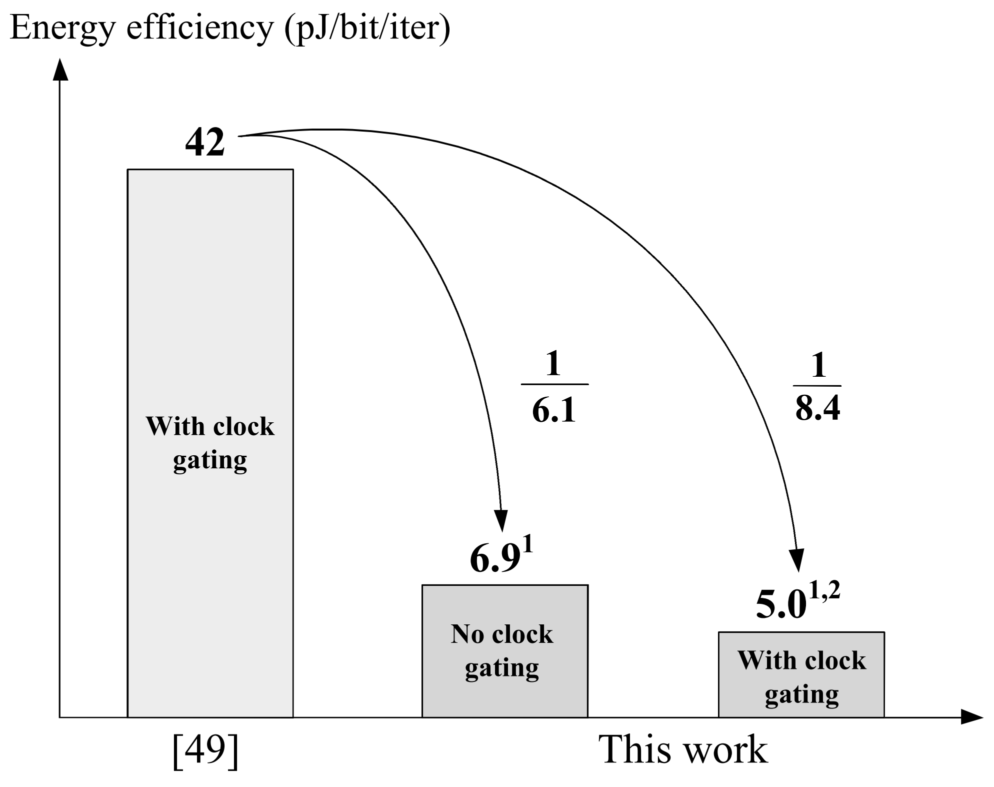
	Semi-layer based	Bit-serial based	[49]
Technology	65nm	65nm	0.13μm
Supply voltage	1.08V	1.2V	1.2V
Code length	576~2304		
Code rate	1/2, 2/3A, 2/3B, 3/4A, 3/4B, 5/6		
Cycle# /iteration	8~16	24~48	48~54
Logic gate count	1027k	597k	470k
Memory bits	47,232	56,448	72,522
Eq. gate count	1358k	968k	946k
Frequency	35MHz	110MHz	214MHz
Iteration number	10	10	10
Throughput(Mbps)	1008	1056	955
Power(mW)	21.8	115	397
Normalized power	69.8	230	397
Nor. Power Eff. (pJ/bit/iteration)	6.9	21.8	42

However, this design greatly reduces the necessary clock cycles number at one iteration. Namely, the high parallelism enables this design achieving the same throughput at low frequency. Moreover, the manually inserted clock gating cells reduce 27% power further more. Finally, this design achieves

1008Mbps throughput at the condition of 35MHz frequency and 1.08V voltage with the power 21.8mW and power efficiency 6.9pJ/bit/iteration.

TABLE 4.2 PARALLELISM COMPARISON WITH PREVIOUS DESIGNS

Code rate		1/2	2/3A	2/3B	3/4A	3/4B	5/6
Clock cycle number per iteration	[49]	48	48	48	48	54	52
	Bit-serial	36	48	24	36	36	24
	Semi-layer	12	16	8	12	12	8
Parallelism ratio to [49]		4	3	6	4	4.5	6.5



1. Normalized to SMIC 0.13μm process
   $E_{130}=E_{65}\times(V_{130}/V_{65})^2\times\alpha = 3.2E_{65}$
2. The clock gating is simulation result: 27%

Figure 4.4 Power efficiency comparison

Table 4.2 lists the clock cycles per iteration needed for the 6 code rates. Compared with [49], the parallelism of the proposed design boosts up 3x to 6.5x. Figure 4.4 shows the energy efficiency comparison with normalization to 0.13μm process. The proposed design with no clock gating achieves 6.1x improvement. Based on the clock gating with the reduction ratio of 27%, the improvement can reach up to 8.4x.

4.5 Conclusion

This section describes another scheduling and processing architecture named as the semi-layer scheduling decoding. It achieves higher parallelism than the previous bit-serial layered scheduling architecture, wherein one iteration only costs 8~16 clock cycles. With the high parallelism and clock gating scheme, this design needs 40% more hardware cost, but achieves up to 6.5X parallelism and 82.4% power reduction in the normalized power comparison than the state-of-art work.

5 CONCLUSION

As the critical part of QC-LDPC decoder, we propose a novel permutation network architecture based on Banyan network. We prove the nonblocking property of Banyan network under cyclic shifts with the input number is power of 2 and give the control signal generating algorithm of Bayan network. By utilizing the bypass network, we present the nonblocking scheme for any input number and shift number. Moreover, we give the hard architecture of the control signal generator, which can significantly reduce the hardware complexity and latency. Through extending the structure of the banyan based permutation network, we propose the generic permutation network (GPN) for the QC-LDPC decoder. This proposal specifies a common architecture of permutation network, which is suitable for any specified application with high hardware efficiency. Moreover, the proposed scheme could greatly reduce the latency because of much less stages and efficient control signal generating algorithm. The implementation results for WiMAX standard and WiFi standard prove that GPN could not only reduce the hardware cost but also improve the timing performance.

For the implementation of QC-LDPC decoder, we choose the codes defined in WiMAX standard, which are typical in the QC-LDPC codes of wireless standard. We introduces the proposed fully parallel layered decoding architecture, which achieves higher parallelism than previous published designs. This architecture adopts three key schemes in the implementation: the first is calculating and updating all the messages within one layer simultaneously, wherein the arithmetic units and permutation network are designed as 1-bit form so as to avoid interconnection problem; the second is improve the parallelism with dedicated PCM reordering and the two-layer concurrent processing for low code rates, with which the clock cycles of one iteration are reduced to 24~48; the third is storing all the messages in registers and sharing the storing cells of VN and APP. Based on these schemes, this design achieves over 1Gbps throughput. The power reduces 42.1% and the power efficiency reduces 63.6% in the normalized comparison with the state-of-art work.

In addition, another architecture, semi-layer scheduling architecture, is proposed even higher parallelism. In this architecture, half blocks in one layer are processed concurrently, which costs at most 2 clock cycles to process one layer in the iterative decoding procedure. As a result, one iteration costs only 8-16 clock cycles in the implementation. With the high parallelism and clock gating scheme, this design needs 40% more hardware cost, but achieves up to 6.5X parallelism and 82.4% power reduction in the normalized power comparison than the state-of-art work.

REFERENCE

[1] Robert G. Gallager, "Low-Density Parity-Check Codes", Monograph, MIT Press, 1963.

[2] Mackay, D. J. C., Neal, R. M., "Near Shannon limit performance of low density parity check codes", IET Electronics Letters, volume 32, issue 18, pp. 1645-1646, Aug. 1996.

[3] Todd K. Moon. "Error Correction Coding, Mathematical Methods and Algorithms", Wiley, 2005, ISBN 0-471-64800-0.

[4] R. M. Tanner, "A recursive approach to low complexity codes," IEEE Trans. Inf. Theory, vol. IT-27, no. 5, pp. 533–547, Sep. 1981.

[5] S. Lin and D. J. Costello, Error Control Coding: Fundamentals and Applications. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[6] Wen Ji, "Research on High Performance LDPC Decoder", Doctoral dissertation, Feb., 2011

[7] "IEEE std 802.16e^{TM}-2005," Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems, Feb. 2006.

[8] Dongfeng Yuan, Haigang Zhang, "LDPC code theory and application", Ptpress, April, 2008, ISBN 978-7-115-17668-4/TN.

[9] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," IEEE Trans. Inf. Theory, vol. IT-45, no. 2, pp. 399–431, Mar.

1999.

[10] N. Wiberg, “Codes and decoding on general graphs,” Ph.D. dissertation, Dept. Electrical Eng., Univ. Linkoping, Linkoping, Sweden, 1996.

[11] J. Chen and M. C. Fossorier, “Decoding low-density parity-check codes with normalized APP-based algorithm,” Proc. IEEE GLOBECOM, Nov. 2001, vol. 2, pp. 1026–1030.

[12] J. Chen, “Reduced complexity decoding algorithms for low-density parity-check codes and turbo codes,” Ph.D. dissertation, Dept. Electrical Eng., Univ. Hawaii, Honolulu, HI, 2003.

[13] M. M. Mansour and N. R. Shanbhag, “High-throughput LDPC decoders,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 11, pp. 976–996, Dec. 2003.

[14] A. J. Blanksby and C. J. Howland, “A 690-mW-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder,” IEEE J. Solid-State Circuits, vol. 37, no. 3, pp. 404–412, Mar. 2002.

[15] A. Darabiha, A. C. Carusone, and F. R. Kschischang, “A 3.3-Gbps bitserial block-interlaced min-sum LDPC decoder in 0.13- m CMOS,” in Proc. IEEE CICC, 2007, pp. 459–462.

[16] T. Mohsenin, D. Truong, and B. Baas, “Multi-split-row threshold decoding implementations for LDPC codes,” in Proc. IEEE ISCAS, 2009, pp. 2449–2452.

[17] Z. Cui, Z. Wang, and Y. Liu, “High-throughput layered LDPC decoding architecture,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 17, no. 4, pp. 582–587, Apr. 2009.

[18] Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolic, “A 47 Gb/s LDPC decoder with improved low error rate performance,” in IEEE VLSI Circuits Symp. Dig., 2009, pp. 286–287.

[19] Y. Chen and K.K. Parhi, “Overlappedmessage passing for quasi-cyclic low density parity check codes,” IEEE Trans. Circuits Syst., vol. 51, no.6, pp. 1106–1113, Jun. 2004.

[20] N. Chen, Y. Dai, and Z. Yan, “Partly parallel overlapped sum-product decoder architectures for quasi-cyclic LDPC codes,” in Proc. IEEE SIPS, 2006, pp. 220–225.

[21] M. M. Mansour and N. R. Shanbhag, “Low-power VLSI decoder architectures for LDPC codes,” in Proc. IEEE ISLPED, 2002, pp. 284–289.

[22] T. Zhang and K. K. Parhi, “A 54 Mbps (3,6)-regular FPGA LDPC decoder,” in Proc. IEEE SIPS, 2002, pp. 127–132.

[23] M. Karkooti and J. R. Cavallaro, “Semi-parallel reconfigurable architectures for real-time LDPC decoding,” in Proc. IEEE ITCC, 2004, vol.1, pp. 579–585.

[24] N. Jiang, K. Peng, J. Song, C. Pan, and Z. Yang, “High-throughput QC-LDPC decoders,” IEEE Trans. Broadcasting, vol. 55, no. 2, pp. 251–259, Jun. 2009.

[25] L. Zhang, L. Gui, Y. Xu, and W. Zhang, “Configurable multi-rate decoder architecture for QC-LDPC codes based broadband broadcasting system,” IEEE Trans. Broadcasting, vol. 54, no. 2, pp. 226–235, Jun. 2008.

[26] K. Shimizu, N. Togawa, T. Ikenaga, and S. Goto, “Power-efficient LDPC

code decoder architecture," in Proc. IEEE ISLPED, 2007, pp. 359–362.

[27] Z. Wang and Z. Cui, "Low-complexity high-speed decoder design for quasi-cyclic LDPC codes," IEEE Trans. Very Large Scale Integration (VLSI) Syst., vol. 15, no. 1, pp. 104–114, Jan. 2007.

[28] H. Zhong, W. Xu, N. Xie, and T. Zhang, "Area-efficient min-sum decoder design for high-rate quasi-cyclic low-density parity-check codes in magnetic recording," IEEE Trans. Magn., vol. 43, no. 12, pp. 4117–4122, Dec. 2007.

[29] Z. Wang and Z. Cui, "A memory efficient partially parallel decoder architecture for QC-LDPC codes," in Proc. IEEE Signals, Systems, and Computers, 2005, pp. 729–733.

[30] J. Zhang and M. Fossorier, "Shuffled iterative decoding," IEEE Trans. Commun., vol. 53, no. 2, pp. 209–213, Feb. 2005.

[31] P. Urard, E. Yeo, L. Paumier, P. Georgelin, T. Michel, V. Lebars, E. Lantreibecq, and B. Gupta, "A 135 Mb/s DVB-S2 compliant codec based on 64800b LDPC and BCH codes," in IEEE ISSCC Dig., Feb. 2005, vol. 1, pp. 446–609.

[32] B. Xiang, R. Shen, A. Pan, D. Bao, and X. Zeng, "An area-efficient and low-power multirate decoder for quasi-cyclic low-density parity-check codes," IEEE Trans. Very Large Scale Integration (VLSI) Syst., vol. 18, no.10, pp. 1447–1460, Oct. 2010.

[33] D. Bao, B. Xiang, R. Shen, A. Pan, Y. Chen, and X. Zeng, "Programmable architecture for flexi-mode QC-LDPC decoder supporting wireless LAN/MAN applications and beyond," IEEE Trans. Circuits Syst. I: Reg. Papers, vol. 57, no. 1, pp. 125–138, Jan. 2010.

[34] M. M. Mansour and N. R. Shanbhag, “A 640-Mb/s 2048-bit programmable LDPC decoder chip,” IEEE J. Solid-State Circuits, vol. 41, no. 3, pp. 634–698, Mar. 2006.

[35] D. Hocevar, “A reduced complexity decoder architecture via layered decoding of LDPC codes,” in Proc. IEEE SIPS, 2004, pp. 107–112.

[36] K. Gunnam, G. Choi, M. Yeary, and M. Atiquzzaman, “VLSI architectures for layered decoding for irregular LDPC codes ofWiMax,” in Proc. IEEE ICC, 2007, pp. 4542–4547.

[37] T.-C. Kuo and A. N. Willson, “A flexible decoder IC for WiMAX QC-LDPC codes,” in Proc. IEEE CICC, 2008, pp. 527–530.

[38] V. E. Benes, “Optimal rearrangeable multistage connecting networks,” Bell Syst. Tech. J., vol. 43, no. 4, pp. 1641–1656, Jul. 1964.

[39] C. H. Liu, C. C. Lin, H. C. Chang, C. Y. Lee, and Y. S. Hsu, "Multi-mode Message Passing Switch Networks Applied for QC-LDPC Decoder," IEEE Int. Symp. Circuits and Systems (ISCAS), pp. 752-755, May 2008.

[40] G. Masera, F. Quaglio, and F. Vacca, “Implementation of a flexible LDPC decoder,” IEEE Trans. Circuits. Syst. II, Exp. Briefs, vol. 54, no. 6, pp. 542–546, Jun. 2007.

[41] J. Tang, T. Bhatt, V. Sundaramurthy, and K. K. Parhi, “Reconfigurable shuffle network design in LDPC decoder,” Proc. Int. Conf. ASAP, Sep. 2006, pp. 81–86.

[42] Daesun Oh and Keshab K. Parhi, “Low-Complexity Switch Network for Reconfigurable LDPC decoders”, IEEE Tran. on VLSI systems, 2009

[43] J. Lin, Z. Wang, L. Li, J. Sha and M. Gao, "Efficient Shuffle Network

Architecture and Application for WiMAX LDPC Decoders" IEEE Trans on Circuits and Systems II: Express Briefs, vol. 56, pp. 215-219, Mar. 2009.

[44] L. Goke, G. Lipovski, “Banyan network for partitioning multiprocessor systems,” in Proc. First Ann. Symp. Compur. Arch., 1973, pp.21-30

[45] H.S.Kim et al, “Nonblocking property of Reverse Banyan network”, IEEE Transactions on Communications, Vol. 40, Issue 3, Mar. 1992, pp 472–476

[46] Yang Sun, et al, “VLSI Decoder Architecture for High Throughput Variable Block-size and Multi-rate LDPC Codes”, ISCAS 2007. IEEE International Symposium on 27-30 May 2007 pp. 2104 – 2107

[47] Studer, C., Preyss, N., Roth, C., Burg, A., "Configurable high-throughput decoder architecture for quasi-cyclic LDPC codes" Asilomar Conference on Signals, Systems and Computers (ACSSC), pp. 1137 - 1142, 2008.

[48] X.-Y Hu, E. Eleftherious, D.-M Arnold, and A. Dholakia, “Efficient implementation of the sum-product algorithm for decoding LDPC codes,” Proc. IEEE GLOBECOM, Nov. 2001, vol. 2, pp. 1036-1036

[49] B. Xiang and X. Zeng, “A 4.84 mm^2 847-955 Mb/s 397 mW Dual-Path Fully-Overlapped QC-LDPC Decoder for the WiMAX System in 0.13 μm CMOS,”, Symp. on VLSI Circuits Dig. Tech. Papers, pp. 211-212, Feb. 2010

[50] Z. Chen, et al., “An early stopping criterion for decoding LDPC codes in WiMAX and WiFi standards,” Proc. IEEE ISCAS, pp. 473-476, May 2010.

[51] C.-H. Liu, et al., “An LDPC Decoder Chip Based on Self-Routing

Network for IEEE 802.16e Applications," IEEE JSSC, vol. 43, no. 3, pp. 684-694, Mar. 2008

[52] X.-Y. Shih, et al, "A 19-mode 8.29 mm^2 52 mW LDPC Decoder chip for IEEE 802.16e system," Symp. on VLSI Circuits, pp. 16-17, 2007

[53] P. Meinerzhagen, C. Roth, A. Burg,"Towards generic low-power area-efficient standard cell based memory architectures", MWSCAS 2010, pp.129-132

PUBLICATION

Journal:

1. Xiao Peng, Xiongxin Zhao, Zhixiang Chen, Fumiaki Maehara, Satoshi Goto, "Generic Permutation Network for QC-LDPC decoder", IEICE Trans. Fundamentals, Vol.E93-A, No.12, pp. 2551-2559, Dec., 2010
2. Zhixiang Chen, Xiongxin Zhao, Xiao Peng, Dajiang Zhou, Satoshi Goto, "A High Parallelism LDPC Decoder with an Early Stopping Criterion for WiMax and WiFi Application," IPSJ Trans. TSLDM, Vol. 3, pp. 292-302, Aug. 2010
3. Xiao Peng, Zhixiang Chen, Xiongxin Zhao, Fumiaki Maehara, 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

Conference:

4. Zhixiang Chen, Xiao Peng, Xiongxin Zhao, Qian Xie, Renona Okamura, Dajiang Zhou and Satoshi Goto, "A Macro-Layer Level Fully Parallel Layered LDPC Decoder SOC for IEEE 802.15 Application", International Symposium on VLSI Design, Automation and Test (VLSI-DAT), Hsinchu, Taiwan, Apr., 2011
5. Ying Cui, Xiao Peng, Satoshi Goto, "Design of turbo codes without

4-cycles in Tanner graph representation for message passing algorithm", International Colloquium on Signal Processing & Its Applications (CSPA), Penang, Malaysia, pp. 118-121, Mar., 2011

6. Qian Xie, Zhixiang Chen, Xiao Peng, Satoshi Goto, "A Sorting-based Architecture of Finding the First Two Minimum Values for LDPC Decoding", CSPA, Penang, Malaysia, pp. 105-108, Mar., 2011
7. Xiongxin Zhao, Zhixiang Chen, Xiao Peng, Dajiang Zhou, Satoshi Goto, "A BER performance-aware early termination scheme for layered LDPC decoder", IEEE Workshop on Signal Processing Systems(SiPS), San Francisco, USA, pp. 416-419, Oct., 2010
8. Xiao Peng, Zhixiang Chen, Xiongxin Zhao, Fumiaki Maehara, Satoshi Goto, "High Parallel Variation Banyan Network Based Permutation Network for Reconfigurable LDPC Decoder", International Conference on Application-specific Systems, Architectures and Processors (ASAP), Rennes, France, pp. 233-238, Jul., 2010
9. Zhixiang Chen, Xiongxin Zhao, Xiao Peng, Dajiang Zhou, Satoshi Goto, "An Early Stopping Criterion for Decoding LDPC Codes in WiMAX and WiFi Standards", International Symposium on Circuits and Systems (ISCAS), Paris, France, pp. 473-476, May, 2010
10. Zhixiang Chen, Xiongxin Zhao, Xiao Peng, Dajiang Zhou, Satoshi Goto, "A high-parallelism reconfigurable permutation network for IEEE 802.11n/802.16e LDPC decoder", International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), Kanazawa, Japan, pp. 85-88, Dec., 2009
11. Xiao Peng, Satoshi Goto, "Performance Analysis of Sphere Detection Combined with the LDPC Decoding", CSPA, Kuala Lumpur, Malaysia, pp. 228-232, Mar., 2009
12. Xiao Peng, Satoshi Goto, "Implementation of LDPC decoder for 802.16e", International Conference on ASIC (ASICON), Changsha, China, pp. 501-504, Oct., 2009