Energy-Efficient Decoders of Near-Capacity Channel Codes

by

Youn Sung Park

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering)
in The University of Michigan
2014

Doctoral Committee:

Assistant Professor Zhengya Zhang, Chair
Professor David Blaauw
Assistant Professor Jianping Fu
Professor Trevor Mudge

To my loving family

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Professor Zhengya Zhang for his support and guidance. It has been an honor to be his first Ph.D. student. His tremendous effort, valuable ideas, and warm encouragement helped me make my Ph.D. experience productive and stimulating. I would like to thank Professor David Blaauw and Dennis Sylvester for the insightful discussions and ideas. I would also like to thank Professor Trevor Mudge and Jianping Fu for participating in my dissertation committee and evaluating my research proposal and reviewing this dissertation.

My research was supported by NSF under grant CCF-1054270, DARPA under cooperative agreement HR0011-13-2-0006, NASA, Intel, Broadcom Foundation, and the University of Michigan. The chip fabrication donation was provided by ST Microelectronics. Dr. Pascal Urard at ST Microelectronics, Dr. Engling Yeo at Marvell Semiconductor, and Dr. Andrew Blanksby at Broadcom offered valuable feedback in reviewing my chip designs.

I consider myself very fortunate to be able to work with very talented individuals here at the University of Michigan. I will never forget the enjoyable memories that are shared with Jungkuk Kim, Dongsuk Jeon, Yaoyu Tao, Yoonmyung Lee, Inhee Lee, Chia-Hsiang Chen, Phil Knag, Shuanghong Sun, Wei Tang, Thomas Chen, Shiming Song, Gyouho Kim, Dongmin Yoon, Yejoong Kim, Suho Lee, Suyoung Bang, Sechang Oh, Jaeyoung Kim, Seok-Hyeon Jeong, Taehoon Kang, Hyunsoo Kim, Inyong Kwon, Jaehun Jeong, Zhiyoong Foo, Donghwan Kim, and Myungjoon Choi.

Last but not least, I thank my parents in Korea as well as my wife Yeori Choi and daughter Ji Yu Park, for their endless support and trust. I dedicate this work to them for their love and support.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

Energy-Efficient Decoders of Near-Capacity Channel Codes

by

Youn Sung Park

Chair: Zhengya Zhang

In state-of-the-art communication and storage systems, channel coding, or error control coding (ECC), is essential for ensuring the reliable transmission and storage. State-of-the-art communication and storage systems have adopted channel codes such as LDPC and turbo codes to close the gap towards the ultimate channel capacity known as the Shannon limit. Their goal is to achieve high transmission reliability while keeping the transmit energy consumption low by taking advantage of the coding gain provided by these channel codes. The lower transmit energy is at the cost of extra energy to decode the channel codes. Therefore a decoder that provides a good coding gain at high energy efficiency is essential for achieving the lowest total energy. This work focuses on reducing the decode energy of near-capacity channel codes, including LDPC codes, nonbinary LDPC codes, and polar codes.

LDPC code is one of the most widely used ECC in communication and storage systems due to its capacity-approaching error correcting performance. State-of-the-art LDPC decoder implementations have demonstrated high-throughput in the Gb/s range through the use of highly parallel architectures. However, these designs consumed high memory power due to the use of registers for the high access bandwidth. A non-refresh embedded DRAM is proposed as a new memory solution to replace the most power-hungry parts of

the decoder. The proposed eDRAM takes advantage of the deterministic memory access pattern and short access window to eliminate its refresh circuitry and trades off excess retention time for faster read access time. Architectural techniques can be employed to improve throughput and to accommodate the eDRAM memory. A prototype 1.6 mm^2 65 nm decoder for a (672, 336) LDPC code compatible with the IEEE 802.11ad standard achieves a peak throughput of 9 Gb/s at 89.5 pJ/b. With voltage and frequency scaling, the power consumption is further reduced to 37.7 mW for a 1.5 Gb/s throughput at 35.6 pJ/b.

Nonbinary LDPC (NB-LDPC) code achieves even better error-correcting performance than a binary LDPC code of comparable block length at the cost of significantly higher decoding complexity and low decoding throughput. However, the factor graph of a NB-LDPC code consists of much fewer edges compared to binary LDPC code. In this work, a Gb/s fully parallel NB-LDPC decoder architecture is proposed to take advantage of the low wiring overhead of NB-LDPC codes. With new architectural techniques including a one-step look-ahead check node design and interleaving of variable node and check node operations, both the clock frequency and iteration latency are significantly improved over the state-of-the-art. By a node level convergence detection strategy, a fine-grained dynamic clock gating can be applied to save dynamic power. A 1.22 Gb/s NB-LDPC decoder test chip for a (160, 80) GF(64) NB-LDPC code is designed as a proof-of-concept. The test chip consumes 3.03 nJ/b, or 259 pJ/b/iteration, at 1.0 V and 700 MHz. Voltage scaling to 675 mV improves the energy efficiency to 1.04 nJ/b, or 89 pJ/b/iteration for a throughput of 698 Mb/s at 400 MHz.

The recently invented polar code is provably capacity-achieving compared to capacity-approaching codes. Although the achievable error-correcting performance of a polar code of a practical block length is similar to LDPC code of comparable block length, the recursive construction of polar codes allows for a very regular structure that reduces the wiring complexity of the encoder and decoder design. This work proposes a belief propagation polar decoder, which delivers a much higher throughput over a conventional successive cancellation decoder. Architectural improvements using unidirectional processing element and double-column parallelism further reduce the decoding latency and improve throughput. A latch-based register file is designed to maximize the memory bandwidth while keeping a small

footprint. A 1.48 mm^2 65 nm polar decoder test chip is designed for a 1024-bit polar code. The decoder achieves a peak throughput of 4.68 Gb/s at 15.5 pJ/b/iteration. With voltage and frequency scaling, the energy efficiency is further improved to 3.63 pJ/b/iteration for a throughput of 779 Mb/s at 50 MHz.

This work has demonstrated energy-efficient decoders for LDPC, NB-LDPC, and polar codes to advance the state-of-the- art. The decoders will enable the continued reduction of decode energy for the latest communication and storage applications. The new techniques developed in this work, including non-refresh embedded memory, bit-splitting register file, and fine-grained dynamic clock gating are widely applicable to designing low-power DSP processors.

CHAPTER I

Introduction

Communication and storage of information have become a ubiquitous part of modern technology. The goal of efficient communication and storage is to transmit or store the most information using the least energy. The ultimate theoretical limit of efficient communication and storage is defined by the Shannon capacity, which captures the least transmit energy, or signal-to-noise ratio (SNR), needed for reliable transmission. For a given information reliability measured in terms of bit error rate (BER), a system with weak or no error-correcting code (ECC) will require a high SNR, whereas a system with a strong ECC will be able to reduce the necessary SNR and the transmit energy. Fig. 1.1 illustrates the difference between a coded system versus an uncoded system. A near-capacity code is the most efficient in terms of SNR, but the decoding can be complex, adding significant decode energy. Therefore it is essential to design good decoders for near-capacity channel codes to achieve both good SNR and high decode energy efficiency to reduce the total energy cost. This research is focused on ECC algorithms and their very-large scale integration (VLSI) implementation through algorithm, architecture, and circuits co-optimizations. State-of-the-art near-capacity codes will be considered, including low-density parity-check (LDPC) [5], nonbinary LDPC (NB-LDPC) [6], and polar codes [7].

1.1 Near-Capacity Channel Codes

Turbo code was invented in 1996 [8]. Soon after, LDPC code was rediscovered [5, 6]. Turbo and LDPC codes have been widely adopted in commercial applications, such as

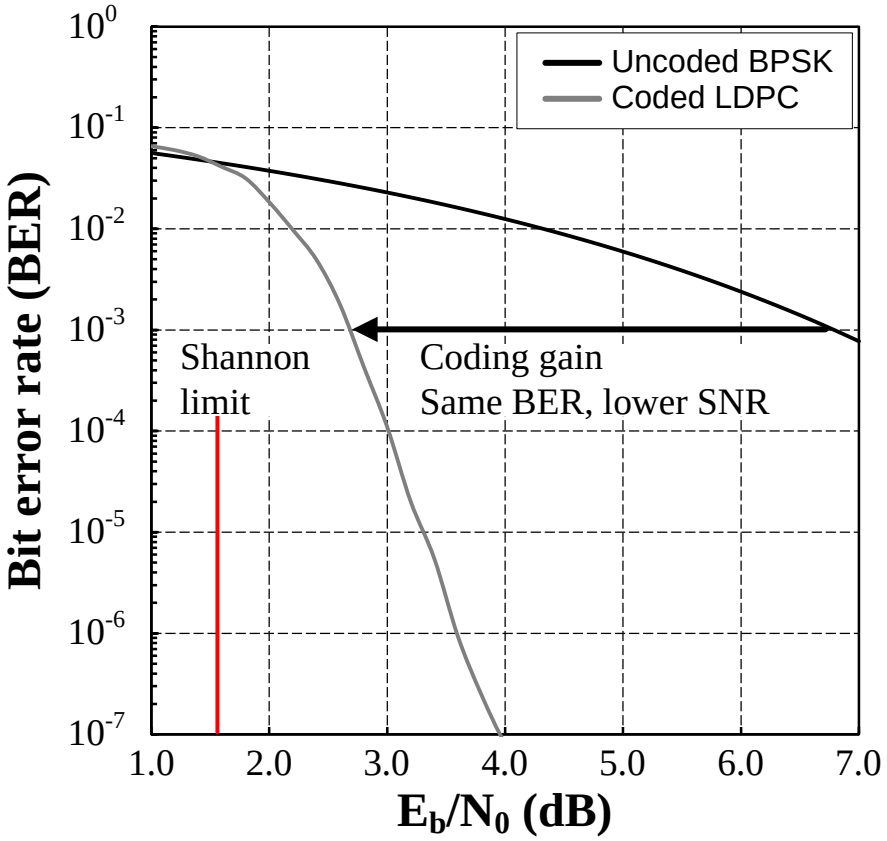


Figure 1.1: Bit error rate comparison between uncoded and encoded systems.

3GPP-HSDPA [9], 3GPP-LTE [10], WiFi (IEEE 802.11n) [3], WiMAX (IEEE 802.16e) [11], digital satellite broadcast (DVB-S2) [12], 10-gigabit Ethernet (IEEE 802.3an) [13], magnetic [14] and solid-state storage [15]. This section reviews LDPC, nonbinary LDPC and polar codes, their current state-of-the-art decoder designs, and major challenges.

1.1.1 Low-Density Parity-Check Codes

An LDPC code is a block code defined by a $M \times N$ parity-check matrix H [5, 16], where M is the block length (number of bits in the codeword) and N is the number of parity checks. The elements of the matrix H(i, j) are either 0 or 1 to represent whether bit j of the codeword is part of parity check i. An H matrix can be represented using a factor graph composed of two sets of nodes: a variable node (VN) for each column of the H matrix and a check node for each row. An edge is drawn between VN(j) and CN(i) if H(i, j) = 1.

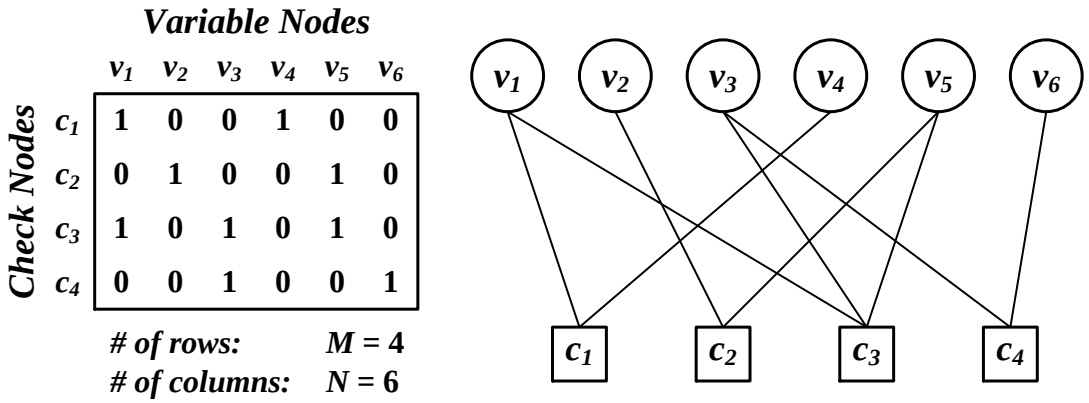


Figure 1.2: An example H matrix and factor graph representation of an LDPC code

An example H matrix with its corresponding factor graph is shown in Fig. 1.2.

Due to their excellent error-correcting performance, LDPC codes have been used in a wide range of applications. The bit error rate and frame error rate (FER) of three wireless standards are illustrated in Fig. 1.3. In addition, the iterative belief propagation (BP) decoding algorithm can be efficiently implemented in the min-sum algorithm [17]. The algorithm enables simple processing nodes that are easily implemented in hardware. Therefore the decoder complexity can be kept low while achieving good error-correcting performance.

Table 1.1 summarizes some key metrics of state-of-the-art LDPC decoders. A 3.03 mm^2 0.13 μm LDPC decoder for WiMAX consumes more than 340 mW for a throughput up to 955 Mb/s [18]. With technology scaling, the area and power consumption of LDPC decoders continue to improve. A 1.56 mm^2 65 nm LDPC decoder for the high-speed wireless standard IEEE 802.15.3c consumes 360 mW for a throughput of 5.79 Gb/s [19]. For a higher throughput, the decoder architecture can be further parallelized, but the power and area increase accordingly. A 5.35 mm^2 65 nm 10-gigabit Ethernet LDPC decoder consumes 2.8 W for up to 47 Gb/s [20].

Parallelizing LDPC decoder for high throughput increases the interconnect complexity [21, 22, 23, 24, 25] and memory bandwidth [26]. Though the interconnect challenge has largely been addressed through the use of structured codes and row-parallel [20, 24, 19]

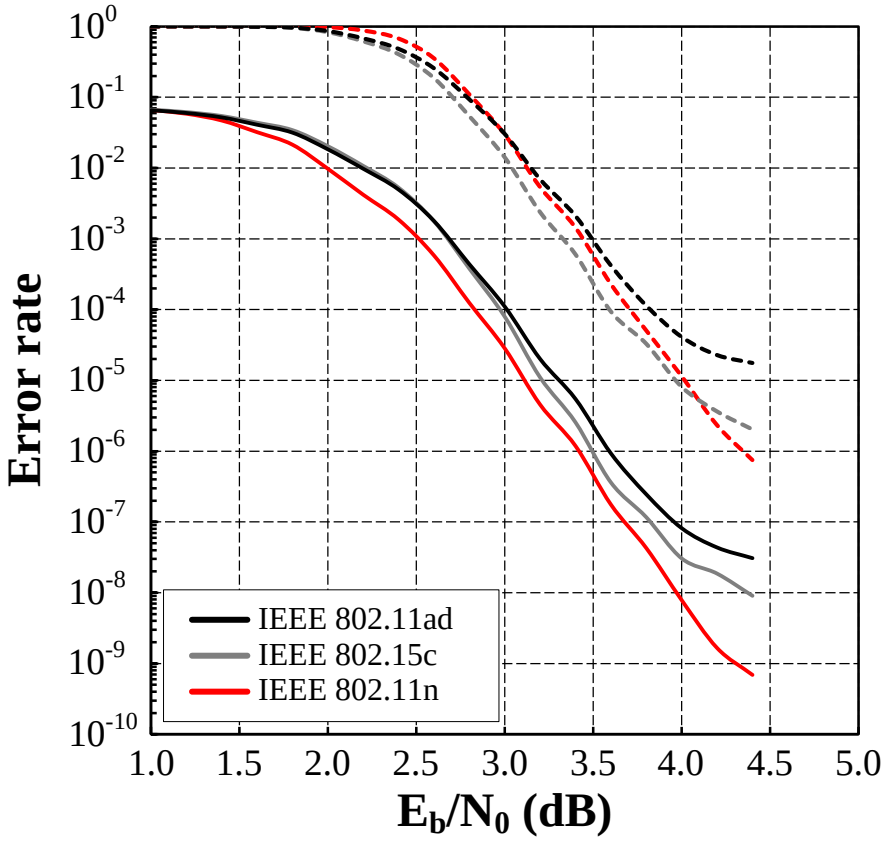


Figure 1.3: BER (solid-line) and FER (dashed-line) of rate-1/2 LDPC codes used in wireless communication standards [1, 2, 3].

Table 1.1: Summary of state-of-the-art LDPC decoder ASIC implementations.

	JSSC'12 [19]	JSSC'11 [18]	JSSC'10 [20]	ASSCC'10 [32]
Standard	802.15.3c	802.16e	802.3an	802.11n
Core Area (mm^2)	1.56	3.03	5.35	1.77
Throughput (Gb/s)	5.79	0.955	6.67[a]	0.679
Norm. Energy Eff. (pJ/bit)[b]	62.4	207.9	61.7	79
Norm. Area Eff. ($Gb/s/mm^2$)[b]	3.70	0.63	0.44	0.77

[a] Early termination enabled.
[b] Normalized to 65nm, 1.0V. Throughput is normalized to 10 decoding iteration for flooding decoders and 5 decoding iteration for layered decoders.

or block-parallel architectures [27, 26, 28, 29, 30, 31, 32, 18, 33], memory bandwidth still remains a major challenge. To support highly parallel architectures, SRAM array needs to be partitioned into smaller banks, resulting in very low area efficiency. High-throughput LDPC decoders use registers for high-speed and wide access, at the expense of high power and area. As a result, memory dominates the power consumption and area of LDPC decoders [34].

1.1.2 Nonbinary Low-Density Parity-Check Codes

Nonbinary LDPC codes, defined over Galois field GF(q), where $q > 2$, offers better coding gain than binary LDPC codes [6]. NB-LDPC codes' excellent coding gain can be achieved even at a short block length, and a low error floor has also been demonstrated.

The main difference between an LDPC and an NB-LDPC code is that an NB-LDPC code is formed by grouping multiple bits to symbols using GF elements, as illustrated in an example in Fig. 1.4. In this example, two bits are grouped to a 2-bit symbol using GF(2^2) or GF(4). From the 4×6 binary LDPC H matrix on the left-hand side, 2×2 submatrices are replaced with single GF(4) elements, resulting in the 2×3 GF(4) nonbinary H matrix on the right-hand side. An NB-LDPC code can also be illustrated using a factor graph composed of variable nodes (VN), check nodes (CN). An edge connects VN v_j and CN c_i if the corresponding entry in the H matrix $H(i, j) \neq 0$. Similarly, 2 VNs of the binary LDPC factor graph are merged to a single VN in the NB-LDPC factor graph. The same applies to the CNs.

The decoding of NB-LDPC codes follows the same BP algorithm [6] that is used in the decoding of binary LDPC codes. However, the complexity of an NB-LDPC decoder is notably higher: each message exchanged between processing nodes in an NB-LDPC decoder carries an array of log-likelihood ratios (LLR) as illustrated in Fig. 1.4; parity check processing follows a forward-backward algorithm; and high-order GF operations require expensive matching and sorting, in contrast to the much simpler addition and compare-select used in binary LDPC decoding.

As shown in Fig. 1.5, the error-correcting performance of NB-LDPC surpasses that of binary LDPC introduced in the previous section. In addition, no error floor is observed for

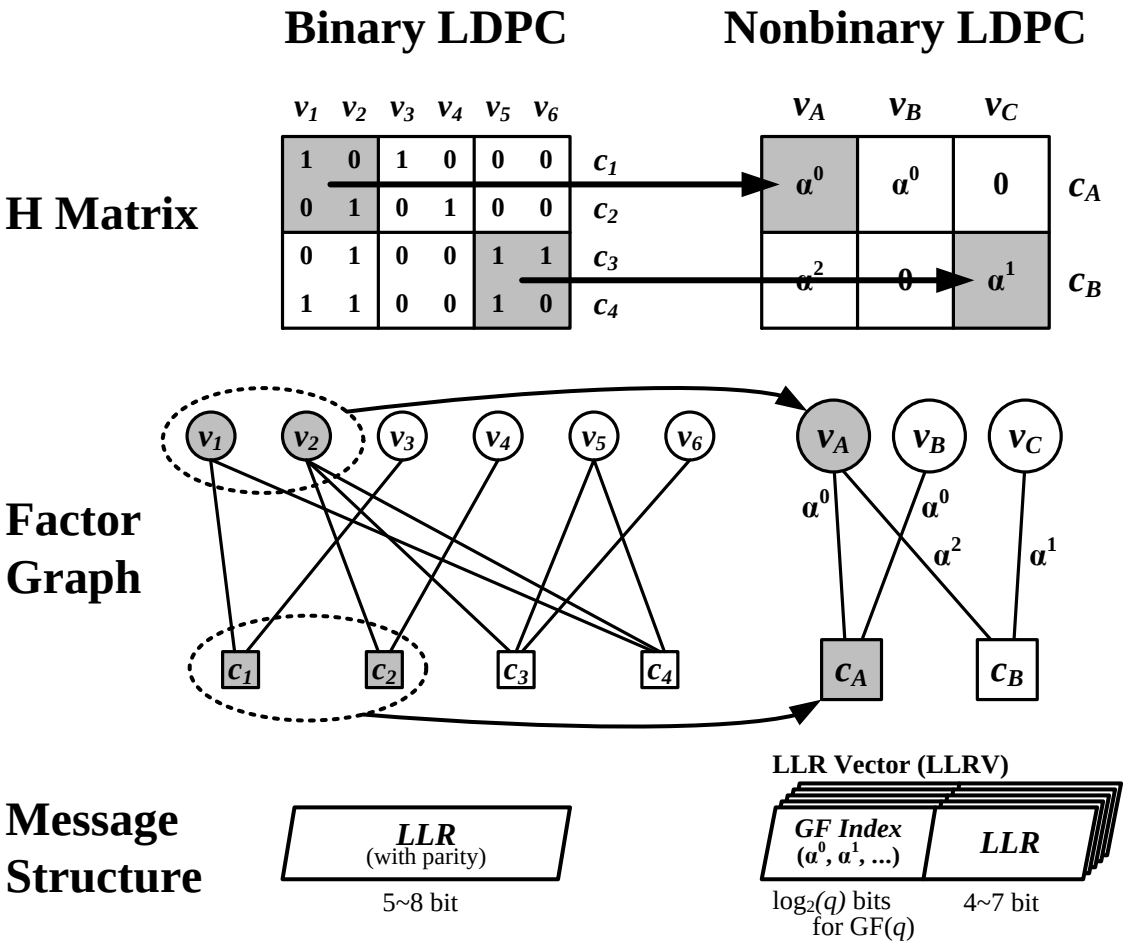


Figure 1.4: Comparison of a binary LDPC code and a nonbinary LDPC (NB-LDPC) code.

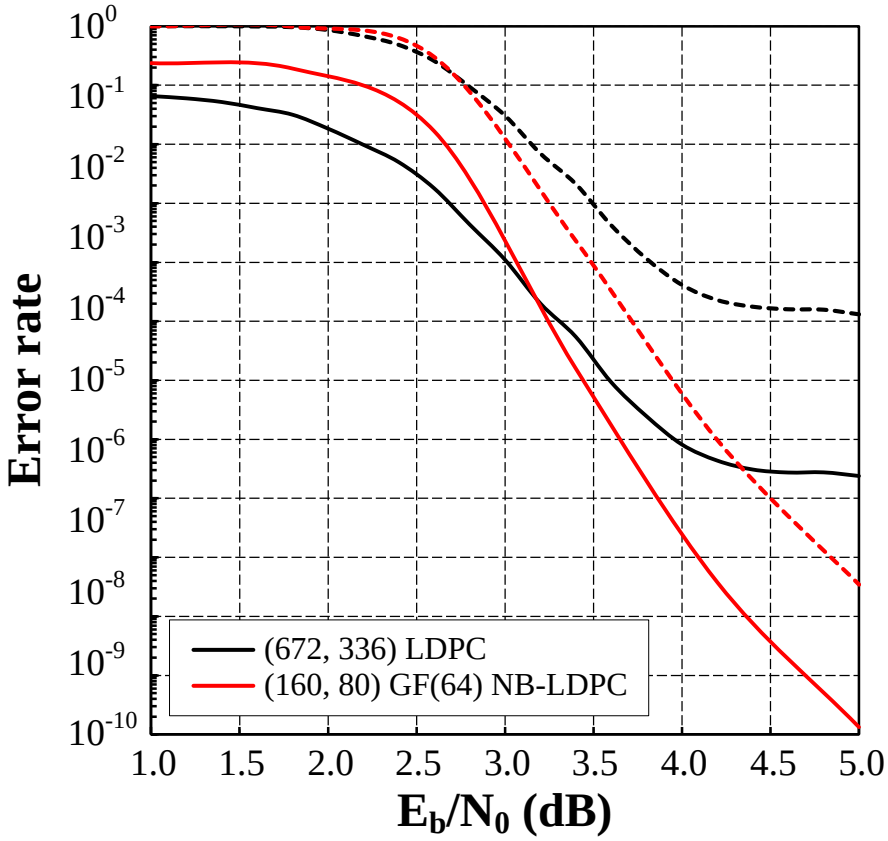


Figure 1.5: BER (solid-line) and FER (dashed-line) comparison of an LDPC code and a NB-LDPC code.

NB-LDPC even at very low error rates. This is an important characteristic as many LDPC codes of finite block length suffer from error floors preventing them from achieving very low error rates.

The high complexity in the processing elements and large memory requirements have prevented any large-scale high-throughput chip implementations in silicon. Only FPGA, synthesis and layout based designs have been demonstrated prior to this work [35, 36, 37, 38, 39, 40, 41, 42, 43]. Table 1.2 summarizes some key metrics of state-of-the-art NB-LDPC decoder layout implementations. A 10.33 mm^2 90 nm NB-LDPC decoder achieves a throughput of 47.7 Mb/s [37]. Another 46.18 mm^2 90 nm NB-LDPC decoder achieves a throughput of 234 Mb/s [40]. With technology scaling, architecture improvements, and algorithm simplifications, the throughput of NB-LDPC decoders continue to improve. A 1.289 mm^2 28 nm NB-LDPC decoder and a 6.6 mm^2 90 nm NB-LDPC decoder achieve

Table 1.2: Summary of state-of-the-art NB-LDPC decoder ASIC layout implementations.

	TVLSI'13 [43]	TVLSI'14 [42]	TSP'13 [40]	TCAS-I'12 [37]
Design	layout	layout	layout	layout
Block Length	837 GF(32)	110 GF(256)	837 GF(32)	248 GF(32)
Core Area (mm^2)	6.6	1.289	46.18	10.33
Throughput (Mb/s)	716	546	234	47.7
Energy Efficiency (nJ/b)[a]	-	4.15	2.76	7.27
Area Efficiency ($Mb/s/mm^2$)[a]	288	33.9	13.4	12.3

[a] Normalized to 65nm, 1.0V.

throughputs of 546 Mb/s and 716 Mb/s, respectively [42, 43]. However, these throughputs are still low compared to Gb/s throughputs of recent LDPC decoders.

1.1.3 Polar Codes

Polar code is a block code recently invented by Arikan which provably achieves the symmetric capacity $I(W)$ of any given binary-input discrete memoryless channel (B-DMC)[7]. Compared to traditional capacity-approaching codes such as turbo and LDPC, polar code is currently the first code that is provably capacity-achieving. Through the channel polarization effect described by Arikan in [7], N independent channels are combined systematically using a recursive function, and only the k most reliable channels are used for sending information while the remaining $N-k$ channels are frozen to known values for both encoder and decoder. An example of channel polarization is illustrated in Fig. 1.6. The plot shows the capacity of each channel index for a block length $N = 2^{14}$ for a BEC channel with erasure probability $\epsilon = 0.5$. From the polarization effect, a group of channels, shown in the green circle, approach a capacity of 1 which means they are reliable channels to transmit information. On the other hand, a group of channels, shown in the red circle, approach a capacity of 0, which mean they are unreliable channels that need to be frozen to known values. The remaining channels would either be frozen or used for information transmission depending on the code rate.

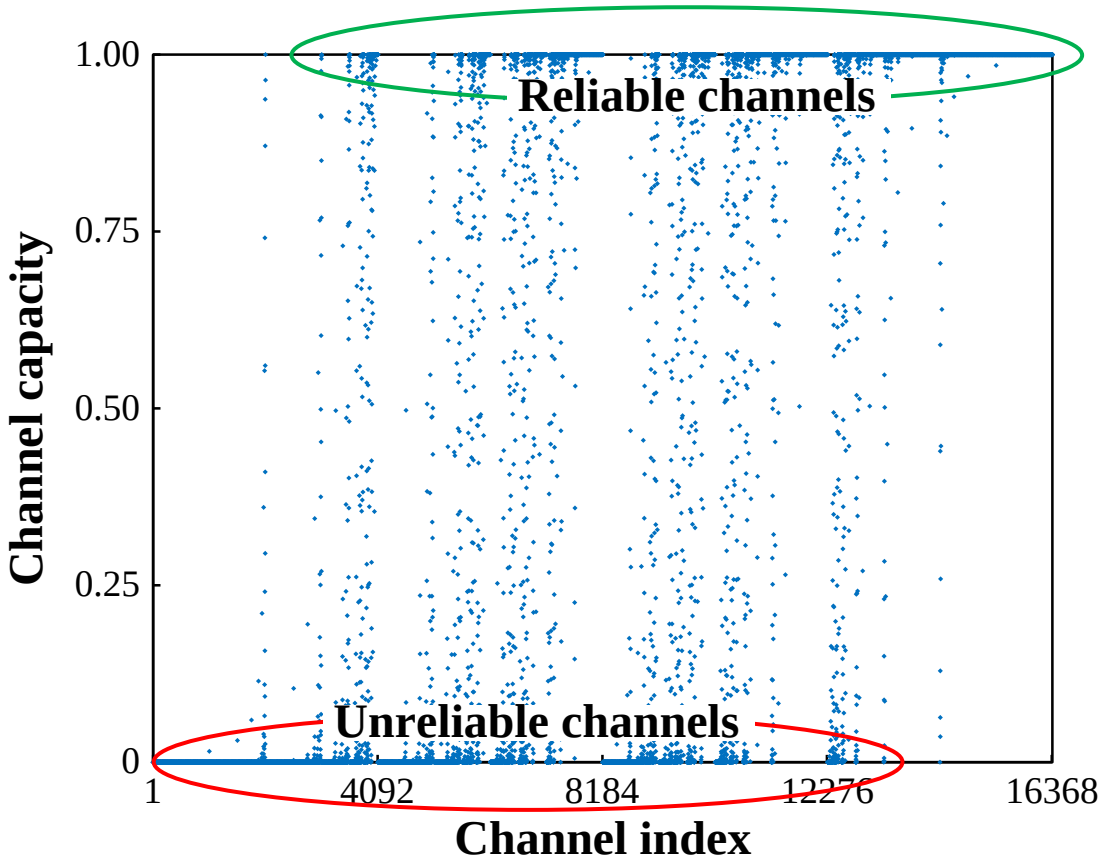


Figure 1.6: Example of polarization effect on a $N = 2^{14}$ polar code in a BEC channel with $\epsilon = 0.5$.

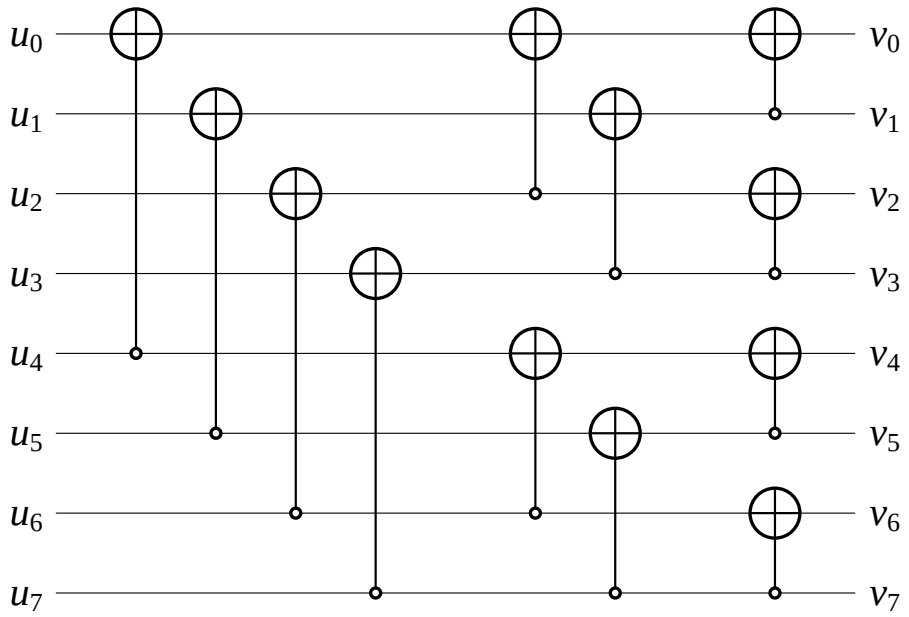


Figure 1.7: Polar code encoder example for $N = 8$.

Polar codes can be constructed with block length $N = 2^n$ and generator matrix

$$F_N = F^{\otimes n}, \text{where } F = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]. \tag{1.1}$$

where $\otimes$ is the Kronecker product operation. Using $n = 3$ as an example,

$$F_8 = F^{\otimes 3} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

A graphical representation of the encoder using the generator matrix F_8 is shown in Fig. 1.7 where u represents the original message, v represents the encoded message to be sent through the channel, and $\oplus$ represents the modulo-2 (or xor) operation.

Although Arikan proved that polar code achieves capacity as the block length N approaches infinity, the error-correcting performance of polar codes of finite block lengths are still away from the Shannon limit. Fig. 1.8 shows the error rate of a 1024-bit polar code

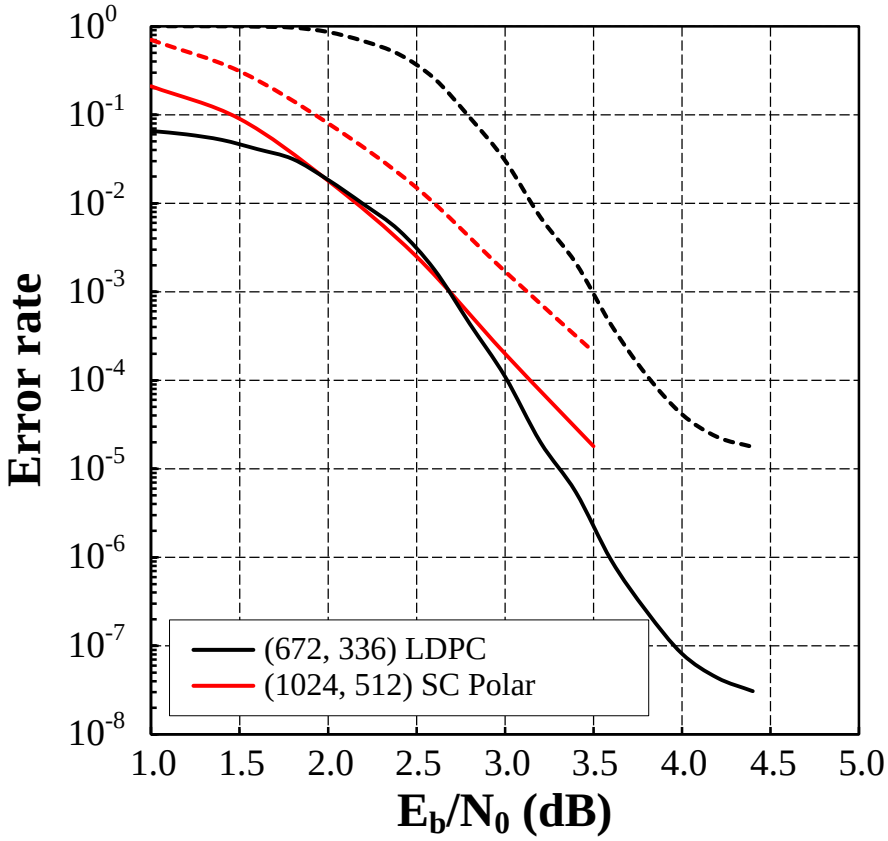


Figure 1.8: BER (solid-line) and FER (dashed-line) comparison of an LDPC code and polar code under successive cancellation decoding.

using successive cancellation (SC) decoding. Current polar codes have similar performance as binary LDPC codes of similar block length.

Due to the recent introduction of polar codes, only a few hardware implementation of polar decoders are found in literature. Most of the work has been done in SC decoding because it is believed that SC decoding provides better error-correcting performance. Several architectures for SC polar decoder have been proposed including the FFT-like SC decoder, pipeline tree architecture, line SC architecture, vector-overlapping architecture [44], and simple successive cancellation (SSC) architecture [45]. On the other hand, little work has been done in BP decoding despite its advantage of higher degree of parallelism.

Table 1.3 summarizes the state-of-the-art polar decoder designs. A 1.71 mm^2 0.18 μm SC polar decoder implementing the 1024-bit polar code consumes 67 mW for a throughput

Table 1.3: Summary of state-of-the-art polar decoder ASIC synthesis implementations.

	ASSCC'12 [46]	JSAC'14 [48]	TSP'13 [47]	ISWCS'11 [49]
Design	silicon	fpga	synthesis	fpga
Code	SC-Polar (1024, 512)	SC-Polar (32768, 29492)	SC-Polar (1024, 512)	BP-Polar (512, 426)
Core Area (mm^2)	1.71	-	0.309	-
Throughput (Mb/s)	49	1044	246.1	52.03
Energy Efficiency (pJ/b/iter)[a]	292.2	-	-	-
Area Efficiency (Mb/s/mm^2)[a]	608.5	-	796.44	-

[a] Normalized to 65nm, 1.0V.

of 49 Mb/s [46]. It is the first reported hardware implementation of a SC polar decoder. Another 0.309 mm^2 65 nm synthesis-based design achieves 246.1 Mb/s at 500 MHz [47]. FPGA designs have also been proposed, one of which is a 32768-bit SC polar decoder which achieves 1044 Mb/s by using the SSC architecture with a high-rate code as a method to increase throughput [48]. An FPGA-based 512-bit BP decoder achieves 52 Mb/s [49] while a GPU-based 1024-bit design achieves 5 Mb/s [50].

1.2 Scope of this Work

In this work, new design techniques are proposed to improve upon the state-of-the-art designs reviewed in the previous section through the use of architecture and circuit techniques that are co-optimized to work with the decoding algorithms.

1.2.1 Low-Density Parity-Check Codes

Logic-compatible embedded DRAM (eDRAM) [4, 51, 52, 53] is proposed as a promising alternative to register-based memory that has been used in building high-throughput LDPC decoders. Logic-compatible eDRAM does not require a special DRAM process and it is both area efficient and low power – an eDRAM cell can be implemented in 3 transistors [4] and it supports one read and one write port, at half the size of a dual-port SRAM cell and its energy consumption is substantially lower than a register. A conventional eDRAM is

however slow. A periodic refresh is also necessary to maintain continuous data retention. Interestingly, we find that when eDRAM is used in high-speed LDPC decoding, refresh can be completely eliminated to save power and access speed can be improved by trading off the excess retention time.

In this work, we co-design a non-refresh eDRAM with the LDPC decoder architecture to optimize its read and write timing and simplify its addressing. An analysis of the LDPC decoder's data access shows that the access window of the majority of the data ranges from only a few to tens of clock cycles. The non-refresh eDRAM is designed to meet the access window with a sufficient margin and the excess retention time is cut short to increase the speed. The resulting 3T eDRAM cell balances wordline coupling to mitigate the effects on its storage. We integrate 32 5 $\times$ 210 non-refresh eDRAM arrays in the design of a 65 nm LDPC decoder to support the (672, 336) LDPC code for the high-speed wireless standard IEEE 802.11ad[1]. All columns of the eDRAM arrays can be accessed in parallel to provide the highest bandwidth. The decoder throughput is further improved using row merging and dual-frame processing to increase hardware utilization and remove pipeline stalls. The resulting decoder achieves a throughput up to 9 Gb/s and consumes only 37.7 mW at 1.5 Gb/s.

1.2.2 Nonbinary Low-Density Parity-Check Codes

The complex decoding and large memory requirement of NB-LDPC decoders have prevented any practical chip implementations. Compared to binary LDPC code, the reduced number of edges in NBLDPC codes factor graph permits a low wiring overhead in the fully parallel architecture. The throughput is further improved by a one-step look-ahead check node design that increases the clock frequency to 700 MHz, and the interleaving of variable node and check node operations that shortens one decoding iteration to 47 clock cycles. We allow each processing node to detect its own convergence and apply fine-grained dynamic clock gating to save dynamic power. When all processing nodes have been clock gated, the decoder terminates and continues with the next input to increase the throughput.

In this work, we present a 7.04 mm^2 65 nm CMOS NB-LDPC decoder chip for a GF(64) (160, 80) regular-(2, 4) code using the truncated EMS algorithm. With the proposed fully

parallel architecture and scheduling techniques, the decoder achieves a 1.22 Gb/s throughput using fine-grained dynamic clock gating and decoder termination for an efficiency of 3.03 nJ/b, or 259 pJ/b/iteration, at 1.0V and 700 MHz. Dynamic voltage and frequency scaling further improves the efficiency to 89 pJ/b/iteration for a throughput of 698 Mb/s, at 675 mV and 400 MHz.

1.2.3 Polar Codes

Due to the inherent serial nature of SC decoding, we turn our attention to BP decoding. A BP decoder is inherently more parallel than SC decoder due to the lack of inter-bit dependency on the decoded output bits. Therefore the decoder can be designed to implement a whole column of processing nodes in the factor graph to increase throughput. The simple computation performed in the processing elements allows for small node footprint which helps achieve high parallelism. By exploiting the order of computation in the BP algorithm, a unified shared memory can be used which reduces the memory size by half and the processing node logic by 33%. To achieve higher throughput and lower latency, a double-column architecture is used which implements twice as many nodes while the memory size remains constant. The double-column architecture increases the throughput at the cost of a slight increase in clock period. To reduce the memory footprint of the decoder, a bit-splitting latch-based register file is employed which enables an 85% placement density.

In this work, we present a 1.476 mm^2 65 nm CMOS polar decoder for the 1024-bit polar code using the BP algorithm. With the proposed architectural transformations and memory optimization, the overall decoder achieves a 4.68 Gb/s throughput while consuming 478 mW for an efficiency of 15.5 pJ/b/iteration, at 1.0 V and 300 MHz. Dynamic voltage and frequency scaling further improves the efficiency to 3.6 pJ/b/iteration for a throughput of 780 Mb/s, at 475 mV and 50 MHz.

CHAPTER II

LDPC Decoder with Embedded DRAM

2.1 Decoding Algorithm

Almost all the latest applications have adopted LDPC codes whose H matrix is constructed using submatrices that are cyclic shifts of an identity matrix or a zero matrix. For example, the newest high-speed wireless standard IEEE 802.11ad [1] specifies a family of four LDPC codes whose H matrices are constructed using cyclic shifts of the $Z \times Z$ identity matrix or zero matrix where $Z = 42$. The structured H matrix can be partitioned along submatrix boundaries, e.g., the H matrix of the rate-1/2 (672, 336) code can be partitioned to 8 rows and 16 columns of 42×42 submatrices as shown in Fig. 2.1.

LDPC encoding and decoding are both based on the H matrix. Encoding produces LDPC codewords that are transmitted over the channel. The receiver decodes the codewords based on the channel output. LDPC decoding uses an iterative soft message passing algorithm called belief propagation [16, 54] that operates on the factor graph in the following steps:

(a) Initialize each VN with the prior log-likelihood ratio (LLR) based on the channel output y and its noise variance σ^2

(b) VNs send messages (the prior LLRs in the first iteration) to the connected CNs

(c) Each CN computes an extrinsic LLR for each connected VN (i.e., the likelihood of each bit's value given the likelihoods from all other VNs connected to the CN), which is then sent back to the VN.

Rate 1/2 **$M = 336$ $N = 672$ $Z = 42$**

I_{40}		I_{38}		I_{13}		I_5		I_{18}						
I_{34}		I_{35}		I_{27}			I_{30}	I_2	I_1					
	I_{36}		I_{31}		I_7		I_{34}		I_{10}	I_{41}				
	I_{27}		I_{18}		I_{12}	I_{20}				I_{15}	I_6			
I_{35}		I_{41}		I_{40}		I_{39}		I_{28}			I_3	I_{28}		
I_{29}		I_0			I_{22}		I_4		I_{28}		I_{27}		I_{23}	
	I_{31}		I_{23}		I_{21}		I_{20}			I_{12}			I_0	I_{13}
	I_{22}		I_{34}	I_{31}		I_{14}		I_4				I_{13}		I_{22}	I_{24}

Rate 5/8 **$M = 252$ $N = 672$ $Z = 42$**

I_{20}	I_{36}	I_{34}	I_{31}	I_{20}	I_7	I_{41}	I_{34}		I_{10}	I_{41}				
I_{30}	I_{27}		I_{18}		I_{12}	I_{20}	I_{14}	I_2	I_{25}	I_{15}	I_6			
I_{35}		I_{41}		I_{40}		I_{39}		I_{28}			I_3	I_{28}		
I_{29}		I_0			I_{22}		I_4		I_{28}		I_{27}	I_{24}	I_{23}	
	I_{31}		I_{23}		I_{21}		I_{20}		I_9	I_{12}			I_0	I_{13}
	I_{22}		I_{34}	I_{31}		I_{14}		I_4					I_{22}	I_{24}

Rate 3/4 **$M = 168$ $N = 672$ $Z = 42$**

I_{35}	I_{19}	I_{41}	I_{22}	I_{40}	I_{41}	I_{39}	I_6	I_{28}	I_{18}	I_{17}	I_3	I_{28}			
I_{29}	I_{30}	I_0	I_8	I_{33}	I_{22}	I_{17}	I_4	I_{27}	I_{28}	I_{20}	I_{27}	I_{24}	I_{23}		
I_{37}	I_{31}	I_{18}	I_{23}	I_{11}	I_{21}	I_6	I_{20}	I_{32}	I_9	I_{12}	I_{29}		I_0	I_{13}	
I_{25}	I_{22}	I_4	I_{34}	I_{31}	I_3	I_{14}	I_{15}	I_4		I_{14}	I_{18}	I_{13}	I_{13}	I_{22}	I_{24}

Rate 13/16 **$M = 126$ $N = 672$ $Z = 42$**

I_{29}	I_{30}	I_0	I_8	I_{33}	I_{22}	I_{17}	I_4	I_{27}	I_{28}	I_{20}	I_{27}	I_{24}	I_{23}		
I_{37}	I_{31}	I_{18}	I_{23}	I_{11}	I_{21}	I_6	I_{20}	I_{32}	I_9	I_{12}	I_{29}	I_{10}	I_0	I_{13}	
I_{25}	I_{22}	I_4	I_{34}	I_{31}	I_3	I_{14}	I_{15}	I_4	I_2	I_{14}	I_{18}	I_{13}	I_{13}	I_{22}	I_{24}

Cyclic Shift of Identity Matrix

For $Z = 4$

$$\mathbf{I_0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{I_1} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{I_2} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad \mathbf{I_3} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 2.1: H matrices of the rate-1/2, rate-5/8, rate-3/4 and rate-13/16 LDPC code for the IEEE 802.11ad standard [1].

(d) Each VN computes the posterior LLR based on the extrinsic LLRs received and the prior LLR, and makes a hard decision (0 or 1). If the hard decisions satisfy all the parity checks, decoding is complete, otherwise the steps (b) to (d) are repeated.

A detailed description of the decoding algorithm can be found in [16]. In BP decoding, soft messages are passed back and forth between VNs and CNs until all the parity checks are met, which indicates the convergence to a valid codeword. In practice, a maximum iteration limit is imposed to terminate decoding if convergence cannot be reached within the given iteration limit.

A practical decoder design follows either the sum-product [16] or the min-sum algorithm [17], which are two popular implementations of the BP algorithm. Using the sum-product algorithm in the log-domain, the VNs perform sum operations and the CNs perform log-tanh, sum and inverse log-tanh operations. Min-sum simplifies the CN operation to the minimum function. The min-sum algorithm usually performs worse than the sum-product

algorithm, and techniques including offset correction and scaling [55] are frequently applied to improve the performance.

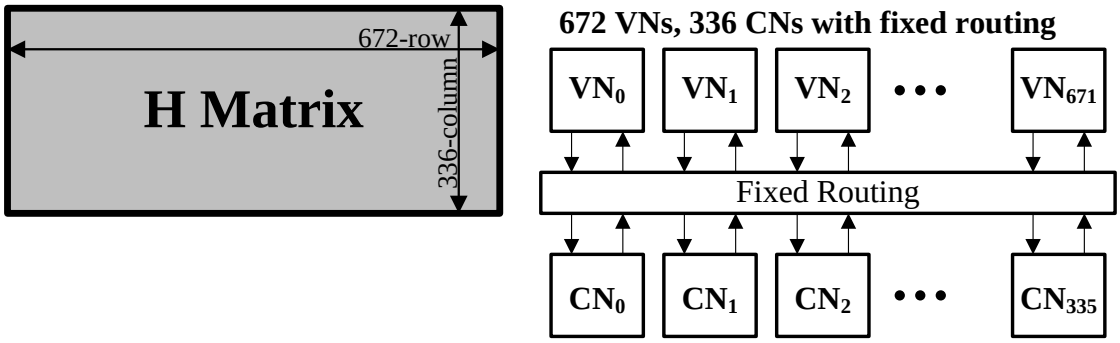
2.2 Decoder Architecture

Common LDPC decoder architectures belong to one of three classes: full-parallel, row-parallel and block-parallel [56] as shown in Fig. 2.2. The full-parallel architecture shown in Fig. 2.2(a) realizes a direct mapping of the factor graph with VNs and CNs mapped to processing elements and edges mapped to interconnects [21, 22, 25]. This architecture provides the highest throughput, allowing each decoding iteration to be done in one or two clock cycles, but it incurs a large area due to complex interconnects.

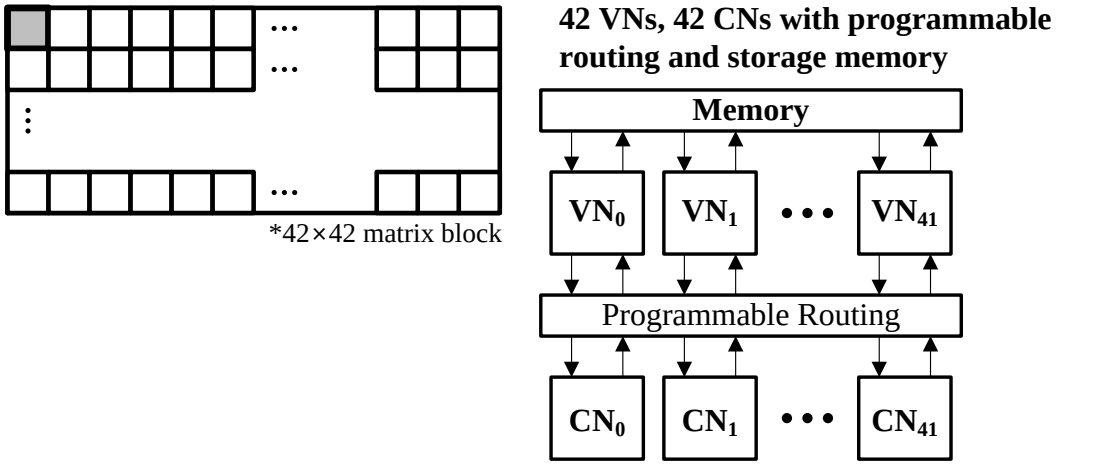
For a lower throughput of up to hundreds of Mb/s, the block-parallel architecture shown in Fig. 2.2(b) processes only one section of the factor graph that corresponds to one or a few submatrices of the H matrix per clock cycle [27, 26, 28, 29, 30, 31, 32, 18, 33]. The VNs and CNs are time-multiplexed, so it takes tens to hundreds of clock cycles to complete one decoding iteration. The more serialized processing requires memories to store messages and configurable routers to shuffle messages between VNs and CNs. The extra overhead in memory and routing results in worse energy and area efficiency. A row-parallel architecture improves upon the block-parallel architecture by processing a larger section of the factor graph that corresponds to an entire row of submatrices of the H matrix per clock cycle [20, 24, 19].

The row-parallel architecture [20, 24, 19] shown in Fig. 2.2(c) provides a high throughput of up to tens of Gb/s, while its routing complexity can still be kept low, permitting a high energy and area efficiency. To meet the 6 Gb/s that is required by the IEEE 802.11ad standard, we choose the row-parallel decoder architecture. The IEEE 802.11ad standard [1] specifies four codes of rate-1/2, rate-5/8, rate-3/4 and rate-13/16, whose H matrices are made up of 16 columns $\times$ 8 rows, 6 rows, 4 rows and 3 rows of cyclic shifts of the 42 $\times$ 42 identity matrix or zero matrix, as illustrated in Fig. 2.1. The four matrices are compatible, sharing the same block length and component submatrix size.

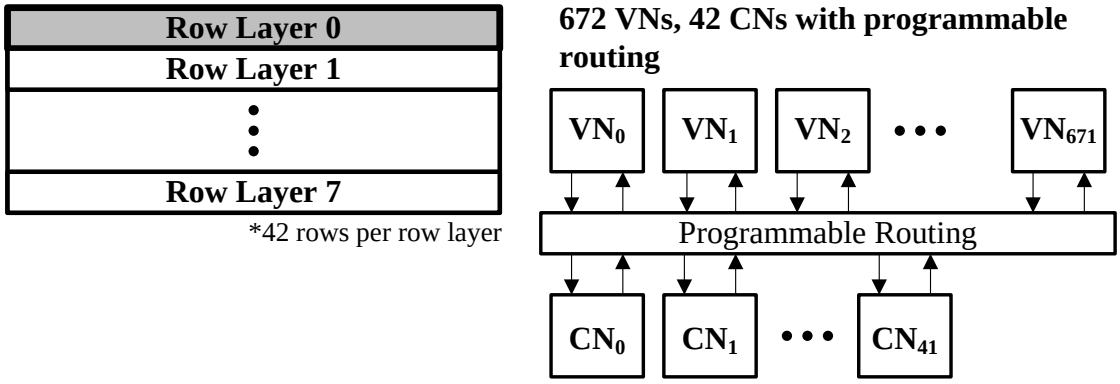
A row-parallel decoder using flooding schedule is designed using 672 VNs and 42 CNs.



(a) Full-parallel architecture



(b) Block-parallel architecture



(c) Row-parallel architecture

Figure 2.2: Illustration of LDPC decoder architectures. The shaded part represents the section of the H matrix that is processed simultaneously.

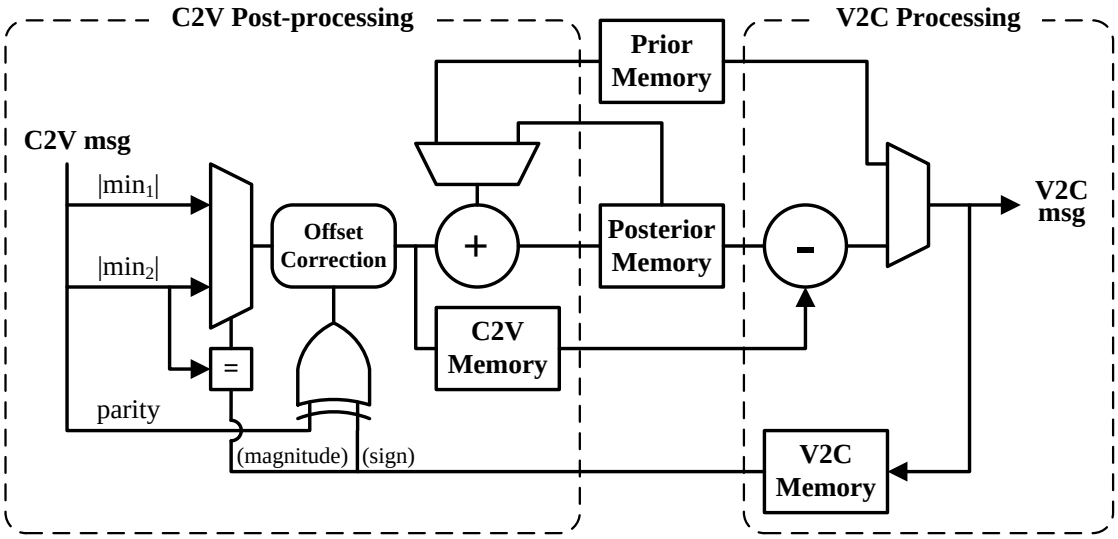
The 672 VNs process the soft inputs of 672 bits in parallel by computing VN-to-CN (V2C) messages and send them to the 42 CNs following the H matrix shown in Fig. 2.1. The 42 CNs compute the parity checks and send CN-to-VN (C2V) messages back to the VNs. The C2V messages are post-processed by the VNs and stored in their local memories. The row-parallel architecture operates on one block row of submatrices in the H matrix at a time, as highlighted in Fig. 2.2.

The VN and CN designs in detail are shown in Fig. 2.3. A VN computes a V2C message by subtracting the C2V message stored in the C2V memory from the posterior log-likelihood ratio (LLR). The V2C message is then sent to the CN while a copy is stored in the V2C memory for post-processing the C2V message later in the iteration. A CN receives up to 16 V2C inputs from the VNs and computes the XOR of the signs of the inputs to check if the even parity is satisfied. The CN also computes the minimum and the second minimum magnitude among the inputs by compare-select for an estimate of the reliability of the parity check. Both the XOR and the compare-select are done using a tree structure. The CN prepares the C2V message as a packet composed of the parity, the minimum and the second minimum magnitude.

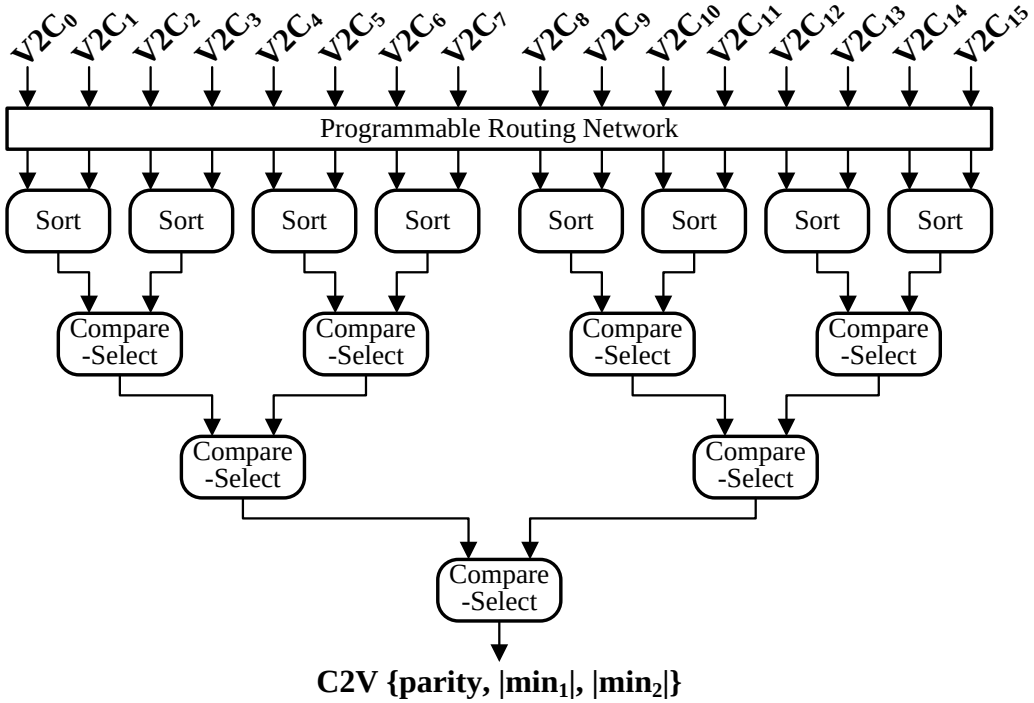
After the C2V message is received by the VN, it compares the V2C message stored in memory with the minimum and the second minimum magnitude to decide whether the minimum or the second minimum is a better estimate of the reliability of the bit decision. The sign and the magnitude are then merged and an offset is applied as an algorithmic correction. The post-processed C2V message is stored in the C2V memory. The C2V message is accumulated and summed with the prior LLR to compute the updated posterior LLR. A hard decoding decision is made based on the sign of the posterior LLR at the completion of each iteration. The messages and computations are quantized for an efficient implementation. We determine based on extensive simulations that a 5-bit fixed-point quantization offers a satisfactory performance.

2.2.1 Pipelining and Throughput

In the LDPC decoding described above, the messages flow in the following order: (1) each of the 672 VNs computes a V2C message, which is routed to one of the 42 CNs through



(a) variable node



(b) check node

Figure 2.3: (a) Variable node, and (b) check node design (an XOR gate is incorporated in the sort and compare-select logic of the CN to perform the parity check.)

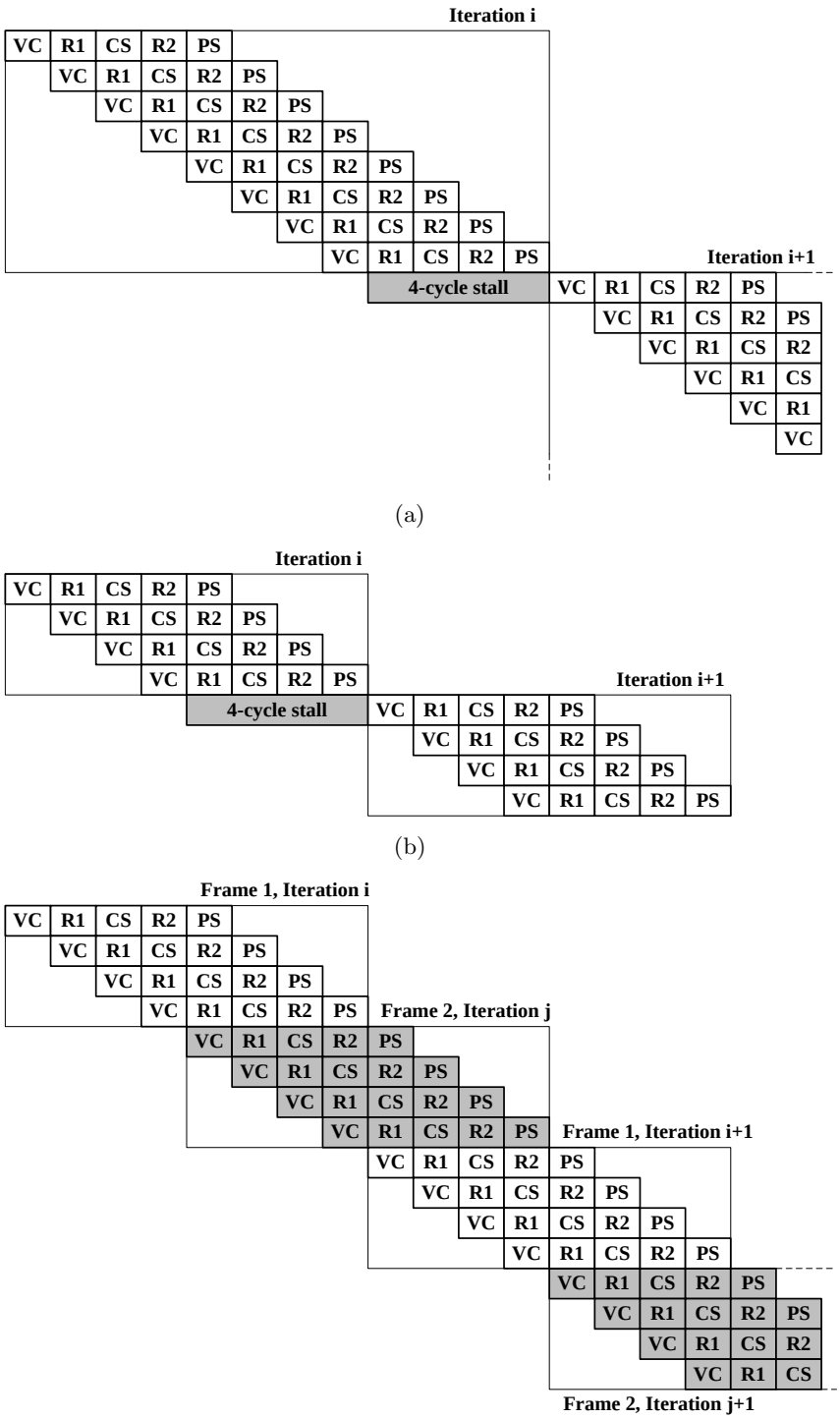


Figure 2.4: Pipeline schedule of (a) a conventional single-frame decoder without row merging, (b) a conventional single-frame decoder with row merging, and (c) proposed dual-frame decoder with row merging. Note that (a) and (b) require stalls in-between frames due to data dependency between the PS and VC stages.

point-to-point links; (2) each CN receives up to 16 V2C messages, and computes a C2V message to be routed back to the VNs through a broadcast link; and (3) each VN post-processes the C2V message and accumulates it to compute the posterior LLR. These steps complete the processing of one block row of submatrices. The decoder then moves to the next block row and the V2C routing is reconfigured using shifters or multiplexers. Based on these steps, we can design a 5-stage pipeline: (1) VN computing V2C message, (2) routing from VN to CN, (3) CN computing C2V message, (4) routing from CN to VN, and (5) VN post-processing C2V messages and computing posterior. For simplicity, the five stages are named VC, R1, CS, R2, and PS, as illustrated in Fig. 2.4(a). The throughput of a row-parallel architecture is determined by the number of block rows m_b and the number of pipeline stages, n_p. The H matrix of the rate-1/2, 5/8, 3/4, and 13/16 code has $m_b = 8$, 6, 4 and 3, respectively. Based on the pipeline chart in Fig. 2.4(a), the number of clock cycles per decoding iteration is $m_b + n_p - 1$. Suppose the number of decoding iteration is n_{it}, then the decoding throughput is given by

$$\text{TP} = \frac{f_{clk} N}{(m_b + n_p - 1) n_{it}} \tag{2.1}$$

where f_{clk} is the clock frequency and N is the block length of the LDPC code. $N = 672$ for the target LDPC code. The 1/2-rate LDPC code has the most number of block rows, $m_b = 8$. $n_p = 5$ for the 5-stage pipeline. To meet the 6 Gb/s throughput with 10 decoding iterations ($n_{it} = 10$), the minimum clock frequency is 1.07 GHz, which is challenging and entails high power consumption.

Each VN in this design includes two message memories, V2C memory and C2V memory. CN does not retain local memory. Each memory contains $m_b = 8$ words to support the row-parallel architecture for the 1/2-rate LDPC code. Each word is 5-bit wide, determined based on simulation. In each clock cycle, one message is written to the V2C memory and one is read from the V2C memory. The same is true for the C2V memory.

For a scalable design and a higher efficiency, the 672 VNs in the row-parallel LDPC decoder are grouped to 16 VN groups (VNG), each of which consists of 42 VNs. The V2C memories of the 42 VNs in a VNG are combined in one V2C memory that contains $m_b =$

8 words and each word is 5b $\times$ 42 = 210b wide. Similarly, the C2V memories of the 42 VNs in a VNG are combined in one C2V memory of 8 $\times$ 210b. In each clock cycle, one 210b word is written to the V2C memory and one 210b word is read from the memory. The same is true for the C2V memory. Each memory's read and write access latency have to be shorter than 0.933 ns to meet the 1.07 GHz clock frequency.

2.3 Throughput Enhancement

The throughput of the LDPC decoder depends on the number of block rows. To enhance the throughput, we reduce the number of effective block rows to process using row merging and apply dual frame processing to improve efficiency [34].

2.3.1 Row Merging

The H matrix of the rate-1/2 code has the most number of block rows among the four codes, but note that the H matrix of the rate-1/2 code is sparse with many zero submatrices. We take advantage of the sparseness by merging two sparse rows to a full row so that they can be processed at the same time (e.g., merge row 0 and row 2, row 1 and row 3, etc.), as illustrated in Fig. 2.5(a). To support row merging, each 16-input CN is split to two 8-input CNs, as in Fig. 2.5(b), when decoding the rate-1/2 code with minimal hardware additions.

The same technique can be applied to decoding the rate-5/8 code by merging row 2 and row 4, and row 3 and row 5. Row merging reduces the effective number of rows to process to 4, 4, 4, and 3 for the rate-1/2, 5/8, 3/4, and 13/16 codes, respectively. Row merging improves the worst-case throughput to

$$\mathrm{TP} = \frac{f_{clk} N}{(n_p + 3) n_{it}} \tag{2.2}$$

To meet the 6Gb/s throughput with 10 decoding iterations, the minimum clock frequency is reduced to 720 MHz. Row merging reduces the V2C memory and C2V memory in each VNG to 4 $\times$ 210b. Each memory's read and write access latency is relaxed, but it has to be below 1.4 ns to meet the required clock frequency.

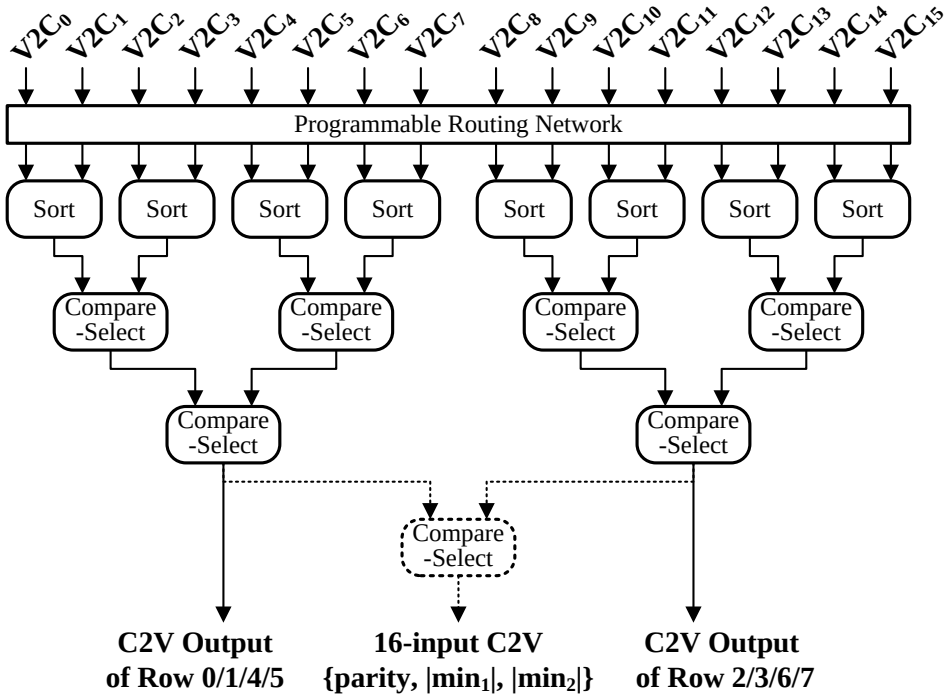
Rate 1/2 **M = 336 N = 672 Z = 42**

Row 0	I_{40}		I_{38}		I_{13}		I_5		I_{18}						
Row 1	I_{34}		I_{35}		I_{27}			I_{30}	I_2	I_1					
Row 2		I_{36}		I_{31}		I_7		I_{34}		I_{10}	I_{41}				
Row 3		I_{27}		I_{18}		I_{12}	I_{20}				I_{15}	I_6			
Row 4	I_{35}		I_{41}		I_{40}		I_{39}		I_{28}			I_3	I_{28}		
Row 5	I_{29}		I_0			I_{22}		I_4		I_{28}		I_{27}		I_{23}	
Row 6		I_{31}		I_{23}		I_{21}		I_{20}			I_{12}			I_0	I_{13}
Row 7		I_{22}		I_{34}	I_{31}		I_{14}		I_4				I_{13}		I_{22}	I_{24}

Rate 1/2 with row merging **M_{eff} = 168 N = 672 Z = 42**

Row 0 & 2 (M0)	I_{40}	I_{36}	I_{38}	I_{31}	I_{13}	I_7	I_5	I_{34}	I_{18}	I_{10}	I_{41}					
Row 1 & 3 (M1)	I_{34}	I_{27}	I_{35}	I_{18}	I_{27}	I_{12}	I_{20}	I_{30}	I_2	I_1	I_{15}	I_6				
Row 4 & 6 (M2)	I_{35}	I_{31}	I_{41}	I_{23}	I_{40}	I_{21}	I_{39}	I_{20}	I_{28}		I_{12}	I_3	I_{28}	I_0	I_{13}	
Row 5 & 7 (M3)	I_{29}	I_{22}	I_0	I_{34}	I_{31}	I_{22}	I_{14}	I_4	I_4	I_{28}		I_{27}	I_{13}	I_{23}	I_{22}	I_{24}

(a)



(b)

Figure 2.5: (a) Illustration of row merging applied to the H matrix of the rate-1/2 LDPC code for IEEE 802.11ad. The merged matrix has only 4 rows, shortening the decoding iteration latency; and (b) modified check node design to support row merging.

2.3.2 Dual-Frame Processing

The 5-stage pipeline introduces a 4 clock cycle pipeline stall between iterations, as shown in Fig. 2.4(a) and (b), because the following iteration requires the most up-to-date posterior LLRs from the previous iteration (i.e., the result of the PS stage) to calculate the new V2C messages. The stall reduces the hardware utilization to as low as 50%.

Instead of idling the hardware during stalls, we use it to accept the next input frame as shown in Fig. 2.4(c). The ping-pong between the two frames improves the utilization, while requiring only the prior and posterior memory to double in size. The message memories can be shared between the two frames and the computing logic and routing remain the same, keeping the additional cost low. With dual-frame processing, the worst-case throughput is increased to

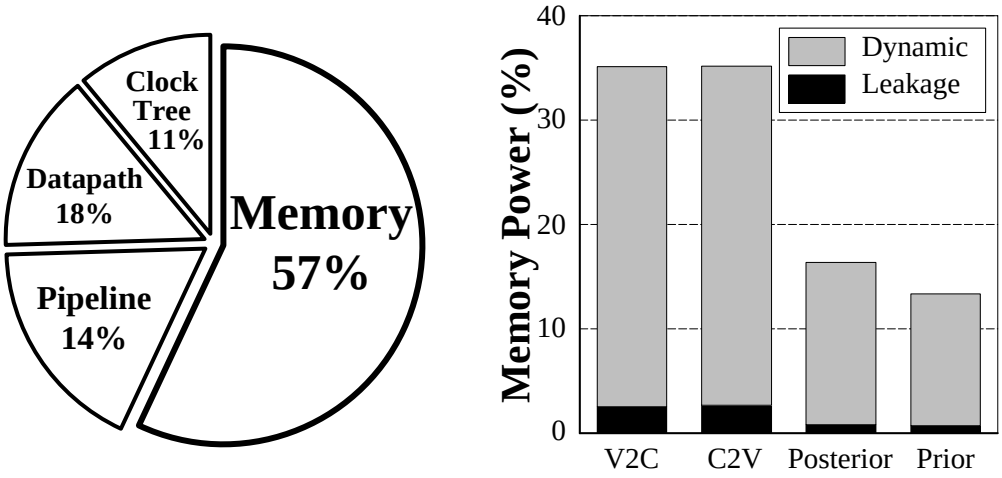
$$\mathrm{TP} = \frac{f_{clk} N}{4 n_{it}} \tag{2.3}$$

To meet the 6Gb/s throughput with 10 decoding iterations, the minimum clock frequency is reduced to 360 MHz. To avoid the read after write data hazard due to dual-frame processing, an extra word is added to the V2C and C2V memory. The size of each memory in a VNG is 5×210b. Each memory's read and write access latency is further relaxed, but it has to be below 2.8 ns to meet the required clock frequency.

2.4 Low-Power Memory Design

The memory in sub-Gb/s LDPC decoder chips is commonly implemented in SRAM arrays, while registers dominate the designs of Gb/s or above LDPC decoder chips. SRAM arrays are the most efficient in large sizes, but the access bandwidth of an SRAM array is very low compared to its size. Therefore SRAM arrays are only found in block-parallel architectures. A full-parallel or row-parallel architecture uses registers as memory for high bandwidth and flexible placement to meet timing.

To estimate the memory power consumption in a high-throughput LDPC decoder, we synthesized and physically placed and routed a register-based row-parallel LDPC decoder that is suitable for the IEEE 802.11ad standard in a TSMC 65 nm CMOS technology. The decoder follows a 5-stage pipeline and incorporates both row merging and dual-frame



(a) Total power

(b) Memory power

Figure 2.6: (a) Power breakdown of a 65 nm synthesized 200 MHz row-parallel register-based LDPC decoder for the IEEE 802.11ad standard, and (b) memory power breakdown. Results are based on post-layout simulation.

processing. In the worst-case corner of 0.9V supply and 125°C, the post-layout design is reported to achieve a maximum clock frequency of 200 MHz, lower than the required 360 MHz for a 6 Gb/s throughput.

The power breakdown of this decoder at 200 MHz is shown in Fig. 2.6. The memory power is the dominant portion, claiming 57% of the total power. In addition to memory, pipeline registers consume 14% of the total power. On the other hand, the datapaths, which include all the combinational logic, consume only 18% of the total power. The clock tree consumes 11% of the total power, the majority of which is spent on clocking the registers. Therefore, reducing the memory power consumption is the key to reducing the chip's total power consumption.

The memory power consumption can be further broken down based on the type of data stored. 35% of the memory power is spent on V2C memory; 35% for C2V memory; 16% for storing posterior LLRs (posterior memory) and 14% for storing prior LLRs (prior memory). The V2C memory and C2V memory account for 70% of the memory power consumption, so they will be the focus for power reduction.

2.4.1 Memory Access Pattern

The V2C memory and C2V memory access patterns are illustrated in Fig. 2.7. When a VN sends a V2C message to a CN, it also writes the V2C message to the V2C memory. The V2C message is finally read when the C2V message is returned to the VN for post-processing the C2V message. From this point on, the V2C message is no longer needed and can be overwritten.

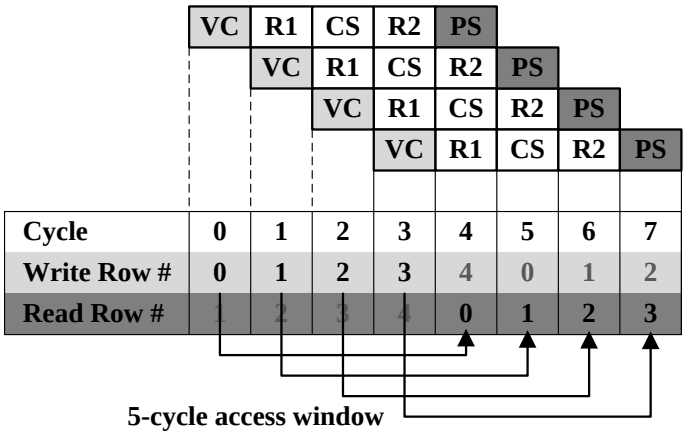
A VN writes every C2V message to the C2V memory, and the C2V message is finally read when the VN computes the V2C message in the next iteration, when the C2V message is subtracted from the posterior LLR to compute the V2C message. From this point on, the C2V message is no longer needed and can be overwritten.

The V2C and C2V memory are continuously being written and read in the FIFO order. The data access window, defined as the duration between when the data is written to memory to the last time it is read, is only 5 clock cycles. The IEEE 802.11ad standard specifies throughputs between 1.5 Gb/s and 6 Gb/s, which require clock frequencies between 90 MHz and 360 MHz using the proposed throughput-enhanced row-parallel architecture. The data access window for both the V2C memory and C2V memory is 5 clock cycles, which translates to 14 ns at 360 MHz (6 Gb/s) or 56 ns at 90 MHz (1.5 Gb/s). Therefore, the data retention time has to be at least 56 ns.

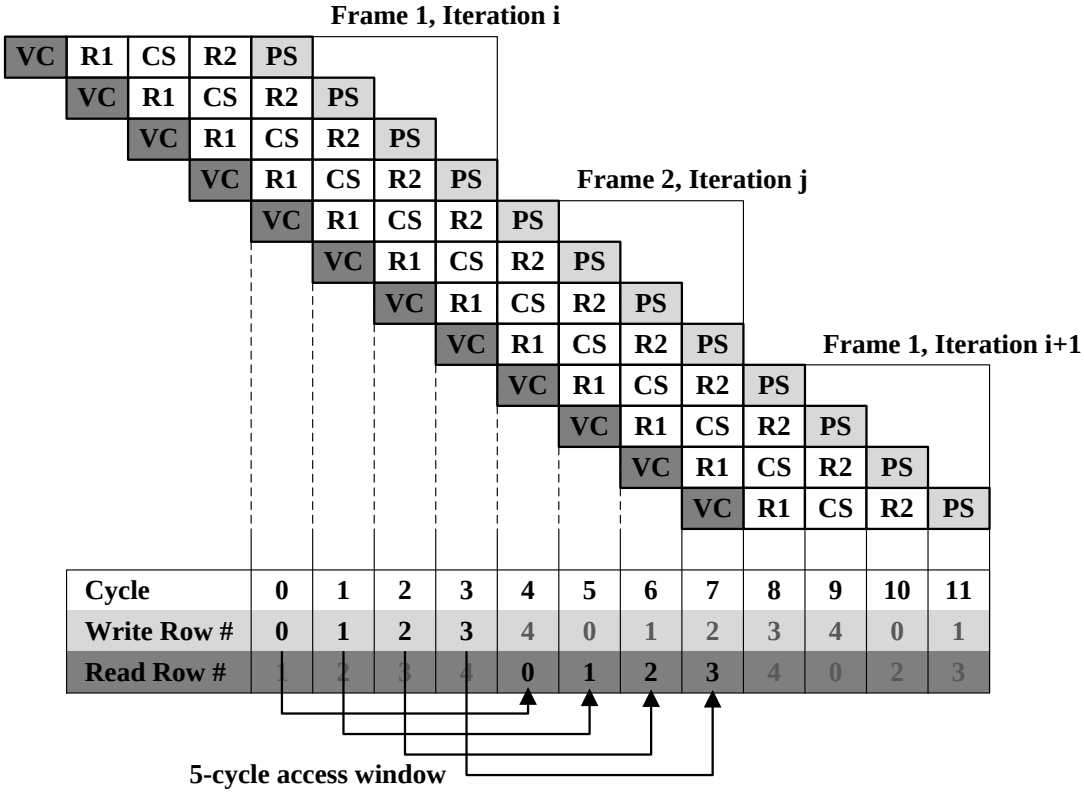
The short data access window, deterministic access order, and shallow and wide memory array structure motivate the design of a completely new low-power memory for the LDPC decoder. In the following we describe the low-power memory design to take advantage of the short data access window. The memory allows dual-port one read and one write in the same cycle to support pipelining and full-bandwidth access required by the decoder architecture.

2.4.2 Non-Refresh Embedded DRAM

Register memory found in highly parallel LDPC decoders consumes high power and occupies a large footprint. Embedded dynamic random access memory (eDRAM) [57, 58, 51, 59, 52, 53] is much smaller in size. A 3T eDRAM cell does not require a special process



(a)



(b)

Figure 2.7: (a) V2C memory access pattern, and (b) C2V memory access pattern.

option. It supports nondestructive read, so it is not necessary to follow each read with write, resulting in a faster performance. The 3T eDRAM cell also supports dual-port access that is required for our application. However, eDRAM is slower than register. A periodic refresh is also necessary to compensate the leakage and maintain continuous data retention. The refresh power is a significant part of eDRAM's total power consumption.

As discussed previously, the memory for LDPC decoder has a short data access window. As long as the access window is shorter than the eDRAM data retention time, refresh can be eliminated for a significant reduction in eDRAM's power consumption, making it attractive from both area and power standpoint. A faster cell often leaks more and its data retention time has to be sacrificed. In the LDPC decoder design, the memory access pattern is well defined and the V2C and C2V memory access window is only 5 clock cycles, therefore we can consider a low-threshold-voltage (LVT) NMOS 3T eDRAM cell to provide only enough retention time, but a much higher access speed.

2.4.3 Coupling Noise Mitigation

Consider the classic 3T eDRAM cell in Fig. 2.8(a) for an illustration of the coupling problem. To write a 1 to the cell, the write wordline (WWL) is raised to turn on T_1 and write bitline (WBL) is driven high and the storage node will be charged up. Upon completion, WWL drops and the falling transition is coupled to the storage node through the T_1 gate-to-source capacitance, causing the storage node voltage to drop. The voltage drop results in a weak 1, reducing the data retention time and the read current. On the other hand, the coupling results in a strong 0 as the storage node will be pulled lower than ground after a write. A possible remedy is to change T_1 to a PMOS and WWL to active low to help write a strong 1, but it results in a weak 0 instead.

To mitigate the capacitive coupling and the compromise between 1 and 0, we redesign the 3T cell as in Fig. 2.8(b) to create capacitive coupling from two opposing directions based on [52]. Similar ideas have also been discussed in [60, 61]. Compared to [52], we use LVT NMOS transistors to improve the access speed by trading off the excess retention time. In this new design, T_2 is connected to the read wordline (RWL), which is grounded when not reading. To write to the cell, WWL is raised. WWL coupling still pulls the storage node

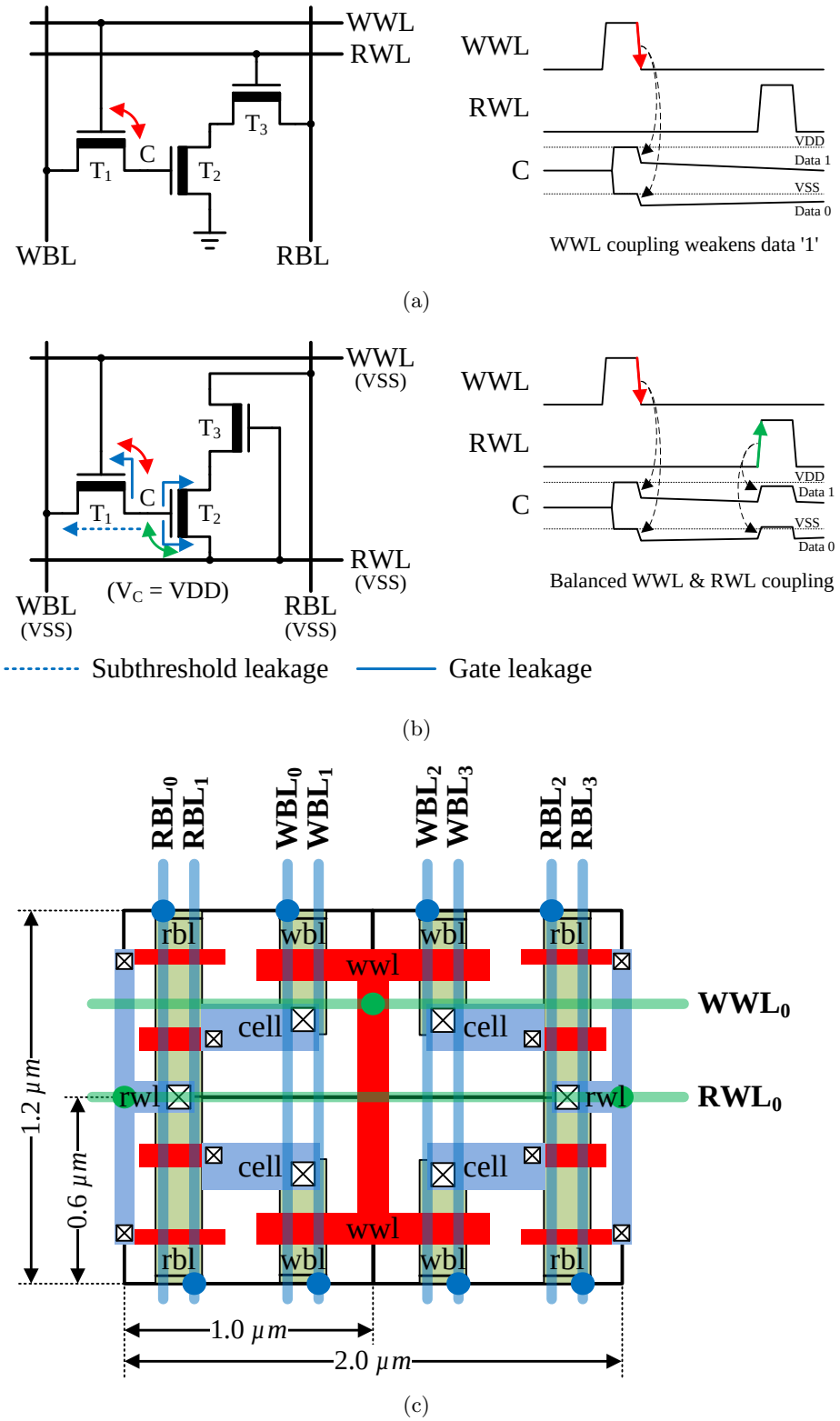


Figure 2.8: Schematic and capacitive coupling illustration of the (a) classic 3T cell [4], and (b) proposed 3T cell and (c) its 4-cell macro layout.

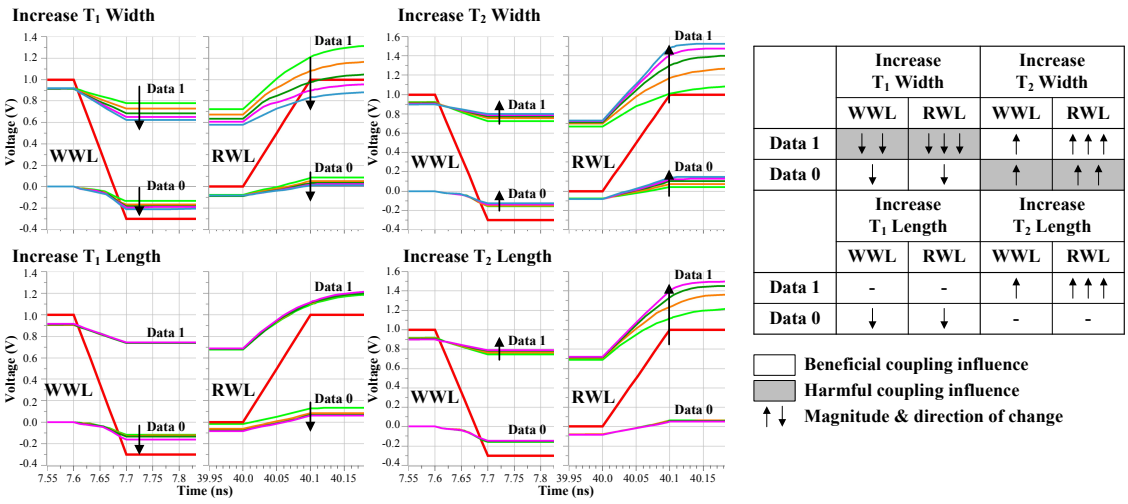


Figure 2.9: Effects of transistor sizing on WWL and RWL coupling noise. Only the falling transition of WWL and the rising transition of RWL are shown as they determine the cell voltage after write and before read.

lower after write, resulting in a weak 1 and strong 0. At the start of reading, the read bitline (RBL) is discharged to ground and RWL is raised. The rising transition of RWL is coupled to the storage node through the T_2 gate-to-drain capacitance, causing the storage node voltage to rise. The design goal is to have the positive RWL coupling cancel the negative WWL coupling. The sizing of T_1 and T_2 can be tuned to balance the coupling. Note that the focus here is on the falling WWL and rising RWL because they determine the critical read speed. Rising WWL in the beginning of write does not matter because the effect is only transient. Falling RWL in the end of read causes storage node voltage to drop, but it will be recovered when RWL rises in the beginning of the next read.

2.4.4 Retention Time Enhancement

After the cell design is finalized, we need to ensure that its data retention time is still sufficient to meet the access window required without refreshing. The data retention time of the 3T eDRAM cell is determined by the storage capacitance and the leakage currents: mainly the subthreshold leakage through the write access transistor T_1, and the gate-oxide leakage of T_1 and the storage transistor T_2. Fig. 2.8(b) illustrates the leakage currents for data 1. Data 1 is more critical than data 0 as it incurs more leakage and its read is critical.

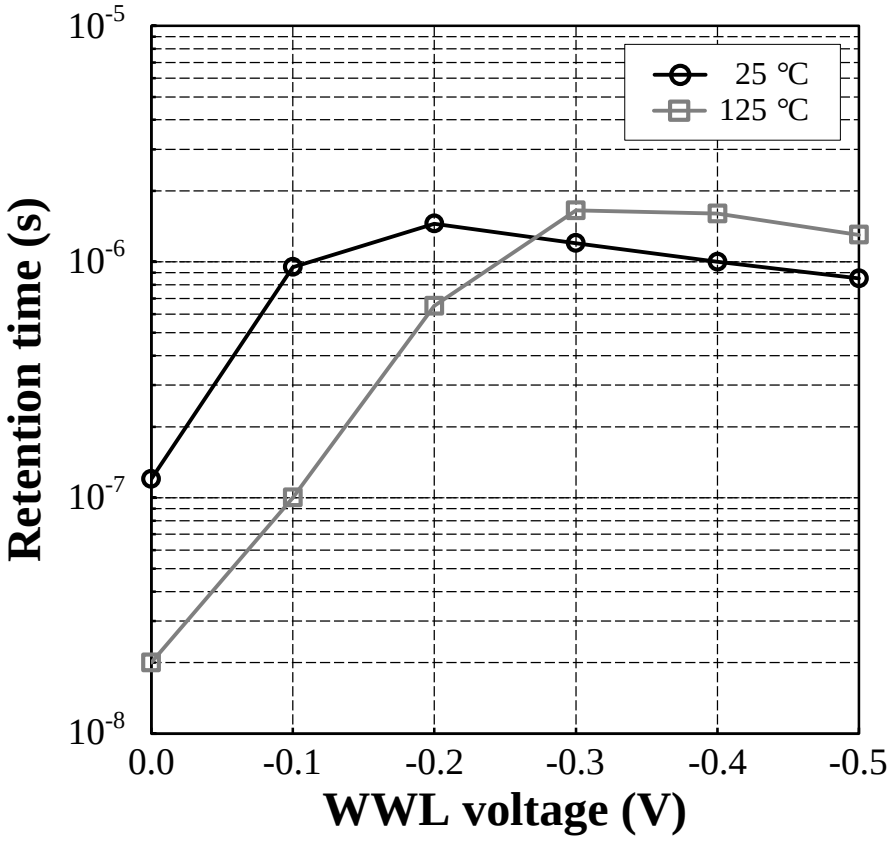


Figure 2.10: Cell retention time with negative WWL voltage.

Both subthreshold and gate-oxide leakage are highly dependent on the technology and temperature. For the 65 nm CMOS process used in this design, the subthreshold leakage is dominant over gate-oxide leakage. To reduce the subthreshold leakage current, we use negative WWL voltage [57] to super cut-off T_1 after write. Fig. 2.10 shows the effect of negative WWL voltage on data 1 retention time at 25°C and 125°C. At 25°C, the retention time improves from 100 ns to over 1μs with a -200 mV WWL. At 125°C, the retention time worsens to 20 ns, but it can be improved to over 1μs with a -300 mV WWL. A 100k-point Monte-Carlo simulation is used to confirm that a -300 mV WWL is still sufficient even after considering process variation. In Fig. 2.11, we measure the time for the storage node voltage to drop to 500 mV and 400 mV after data 1 is written at 125°C. Note that with the help of the RWL coupling, 400 mV is close to the minimum voltage that guarantees data 1 to be reliably read. The results show that the storage node drops to 500 mV in as short as 180 ns, and to 400 mV in 300 ns, which is still much longer than the required data access window of 56 ns (5 clock cycles at 90 MHz for the required minimum throughput of 1.5 Gb/s).

Note that as a proof-of-concept design, the negative WWL voltage is provided from an off-chip supply. However, based on [52], charge pumps can be included to generate the negative voltage on-chip with relatively small impact on the area and power.

The proposed eDRAM design is scalable to a lower technology node. However, managing the cell leakage will be important with the continued reduction of storage capacitance. In a future process technology where leakage becomes more significant, an LVT NMOS eDRAM may not be able to provide the necessary retention time. Regular or high threshold voltage devices and a low-power process may be necessary to ensure a reliable data retention.

2.5 Efficient Memory Integration

A compact 1.0 mm × 0.6 mm layout of the 3T eDRAM cell in a 65 nm CMOS technology using standard logic design rules is shown in Fig. 2.8(c). The length of T_1 and T_2 are increased slightly beyond the minimum length to keep good voltage levels for storing data 0 and 1. The increased T_1 length also reduces the subthreshold leakage. The width of both

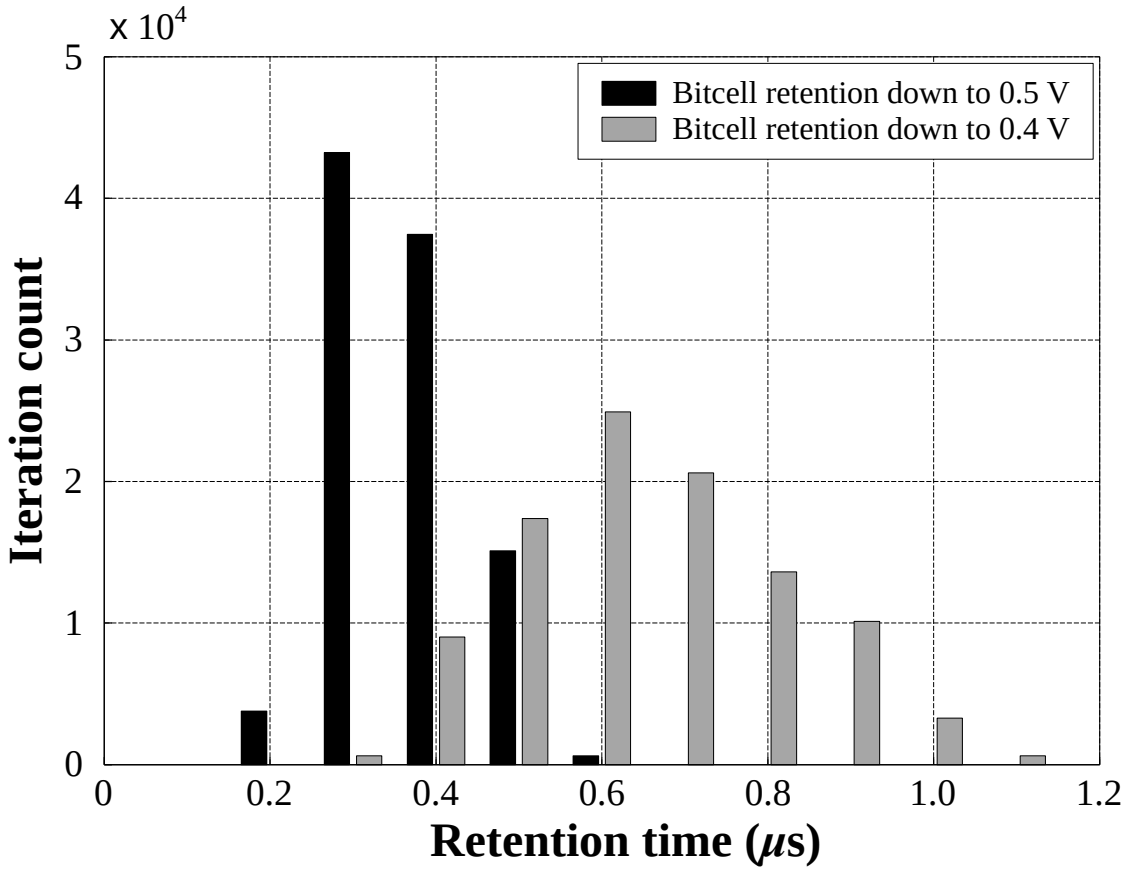


Figure 2.11: 100k Monte-Carlo simulation results of cell retention time at 125°C. The simulation was done on post-layout extracted netlist at 1.0V supply voltage with -300mV WWL. The retention time is measured as the time for the cell voltage to drop to 0.5V (in black) and 0.4V (in grey).

T_2 and T_3 are increased slightly to improve the read speed. The two bitlines WBL and RBL are routed vertically on metal 2 and the two wordlines WWL and RWL are routed horizontally on metal 3.

An area-efficient 4-cell macro can be created in a 2 $\times$ 2 block using a bit cell, its horizontal and vertical reflections, and its 180° rotation, as shown in Fig. 2.8(c). This layout allows poly WWL and diffusion RWL to be shared between neighboring cells to reduce area. 4 RBLs and 4 WBLs run vertically on metal 2. The 8 bitlines have fully occupied the metal 2 tracks.

A larger memory can be designed by instantiating the 4-cell macro. An illustration of a 5 row $\times$ 210 column eDRAM array for the V2C memory or C2V memory in a VNG is illustrated in Fig. 2.13. The array is broken to two parts to shorten the wordlines. 210 single-ended sense amplifiers [62] are attached to RBLs to provide 210 bits/cycle full-bandwidth access. The sense amplifier includes a self-reset function to save power and accommodate process variation.

The cell efficiency for the eDRAM IP is relatively low at 15% due to the shallow memory and full-bandwidth access without column multiplexing. The array efficiency can be improved for a deeper memory. Even at this array efficiency, the effective area per bit is 4.0 μm^2, much smaller than a register. The structured placement of the eDRAM cells improves the overall area utilization.

2.5.1 Sequential Address Generation

Memory address decoder is part of all standard random-access memories, but it is not necessary for the memory designed for LDPC decoder as it only requires sequential access. The memory access sequence can be understood using the multi-iteration pipeline chart in Fig. 2.7. For the V2C memory, in cycle 0 to cycle 3, V2C messages are written to row[0] to row[3]. Starting from cycle 4, there will be one read and one write in every cycle. In cycle 4, one V2C message is written to row[4], and another is read from row[0]. In cycle 5, one V2C message is written to row[0], and another is read from row[1], and so on.

We take advantage of the sequential access to simplify the address generation using a circular 5-stage shift register [63] shown in Fig. 2.12. The output of each register is attached

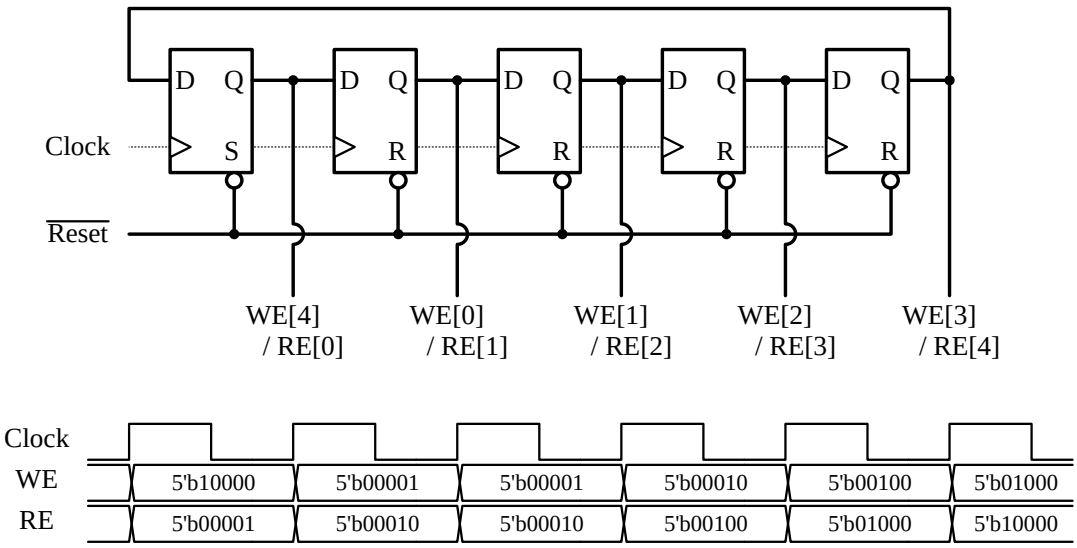


Figure 2.12: Schematic and waveform of sequential address generation based on 5-stage circular shift register.

to one write enable (WE) and one read enable (RE). Only one of the registers is set to 1 in any given cycle and the 1 is propagated around the ring to enable each word serially. The simple sequential address generation saves both power and area.

2.5.2 Simulation Results

The complete 5 row × 210 column eDRAM array layout is shown in Fig. 2.13. The simulation results of the read access time and power consumption of the memory are plotted in Fig. 2.14. At the nominal supply voltage of 1.0 V and WWL voltage of -300 mV, the read access time is 0.68 ns at 25°C. A higher temperature of 125°C decreases the read access time to 0.57 ns, due to the increasing leakage of the sense amplifier that accelerates the charging of the bitline. This effect on read access time becomes more significant when the supply voltage is lowered. At 0.7 V, the read access time is 4.1 ns at 25°C and 1.6 ns at 125°C.

The IEEE 802.11ad LDPC decoder requires 32 5 × 210 eDRAM arrays, 2 for each of the 16 VNGs as V2C memory and C2V memory. To achieve the highest required throughput of 6 Gb/s, the clock period is set to 2.8 ns, and the memory supply voltage has to be set to

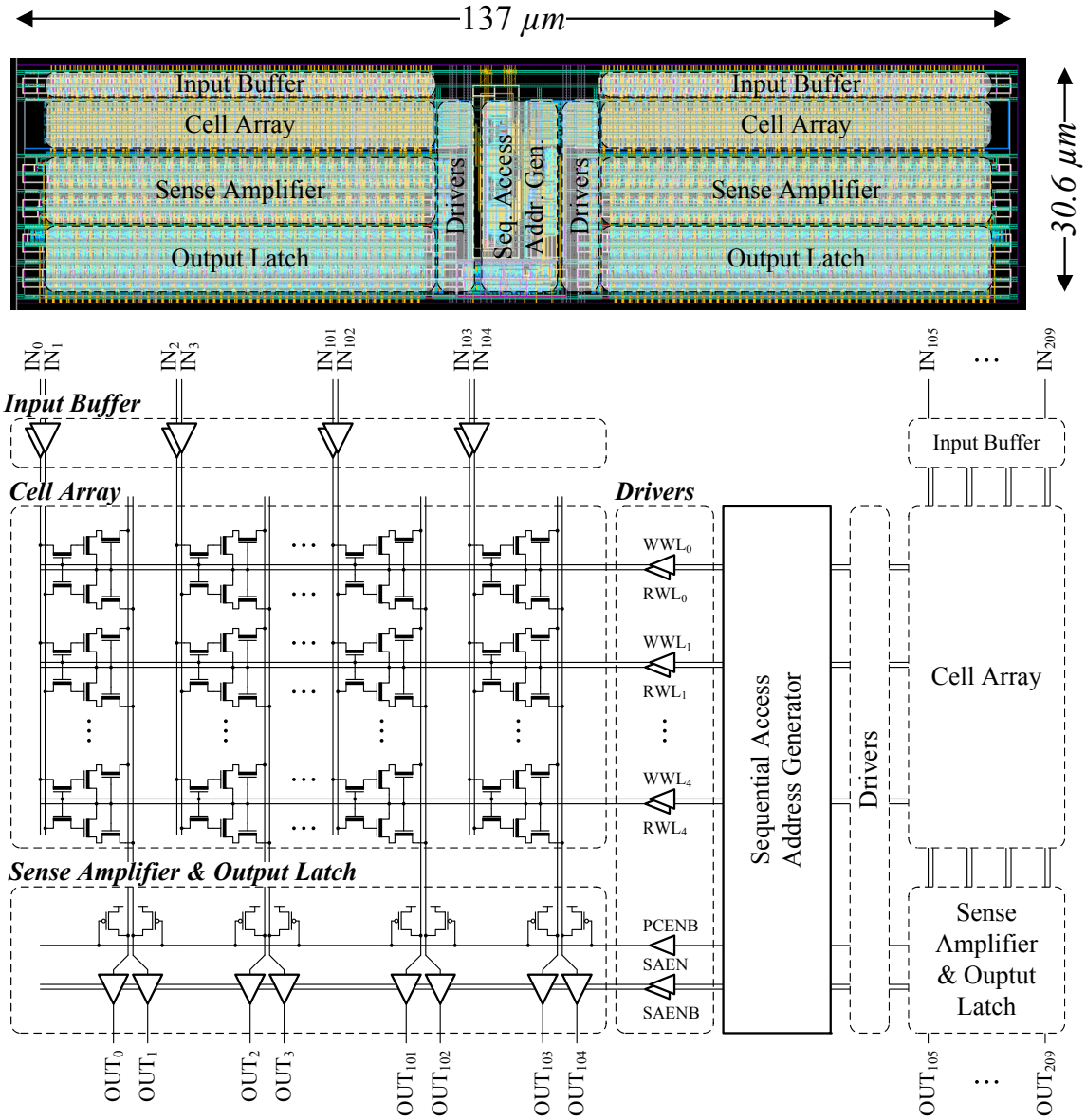


Figure 2.13: Layout and schematic illustration of a 5 × 210 eDRAM array including cell array and peripherals.

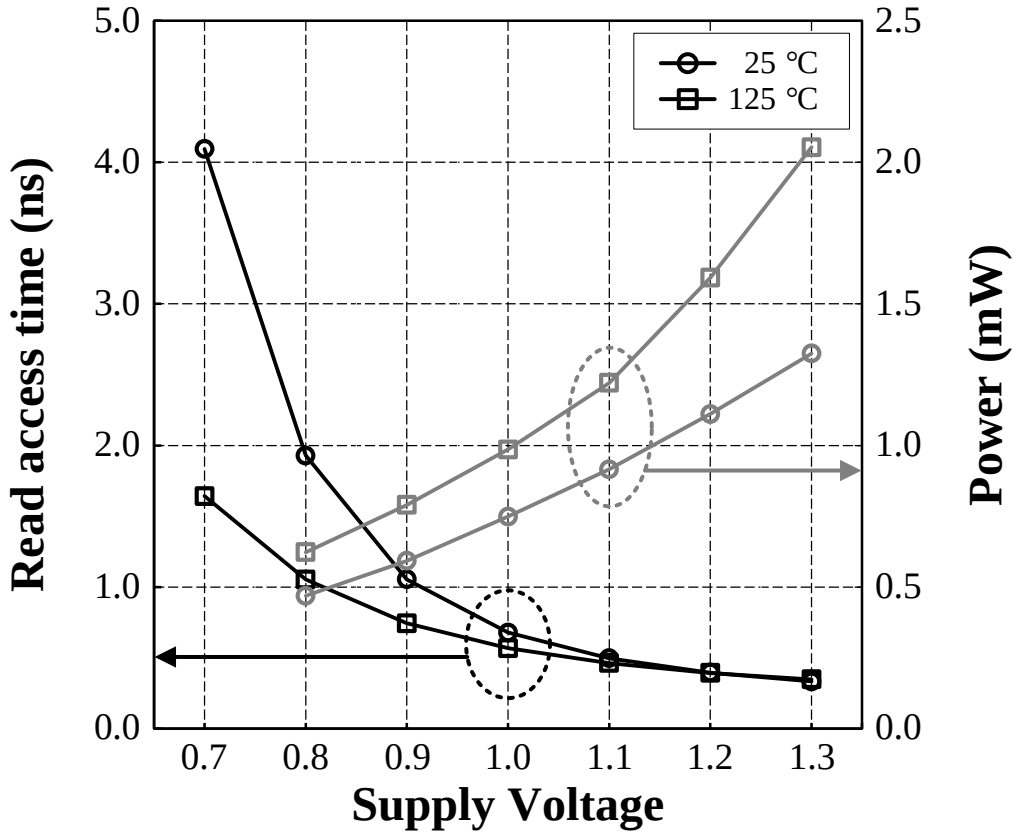


Figure 2.14: Simulated read access time (in black) and power consumption (in grey) of the eDRAM array at 25°C and 125°C. Results are based on post-layout simulation using a -300mV WWL and power is measured at a 180 MHz clock frequency.

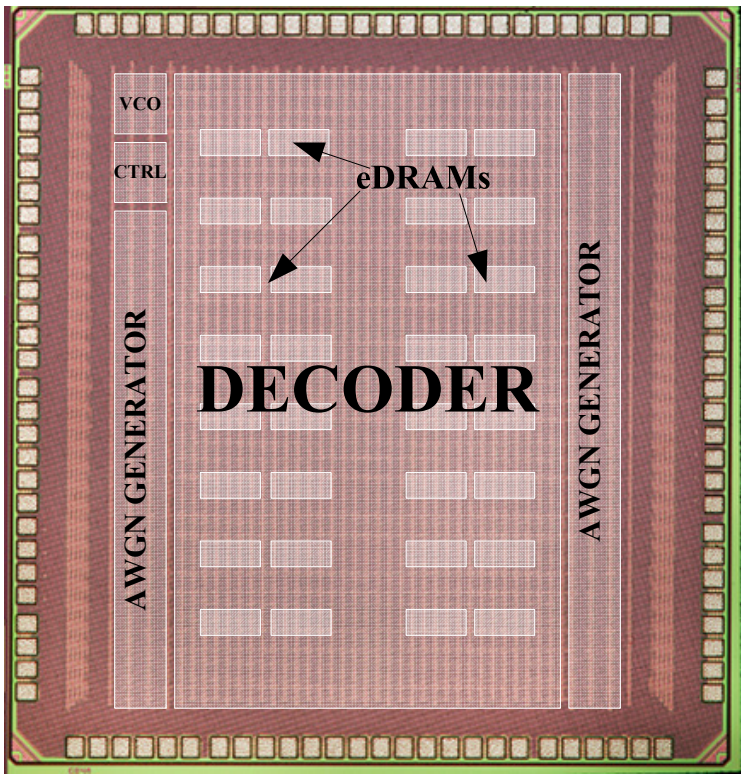


Figure 2.15: Chip microphotograph of the decoder test chip. Locations of the 32 eDRAM arrays inside the LDPC decoder and the testing peripherals are labeled.

about 0.9V.

2.6 Decoder Chip Implementation and Measurements

A decoder test chip was implemented in a TSMC 65 nm 9-metal general-purpose CMOS technology [64]. It was designed as a proof-of-concept to support the rate-1/2 (672, 336) LDPC code for the IEEE 802.11ad standard, but the architecture also accommodates the three higher rate codes. The chip microphotograph is shown in Fig. 2.15. The test chip measures 1.94 mm $\times$ 1.84 mm and the core measures 1.6 mm $\times$ 1.0 mm including 32 5 $\times$ 210 eDRAM arrays.

The decoder test chip uses separate supply voltages for the decoder core logic and eDRAM memory arrays to allow each supply voltage to be independently set to achieve the throughput targets with the lowest power. Clock is generated on-chip, and it can also be provided through an external source. The decoder incorporates AWGN generators to

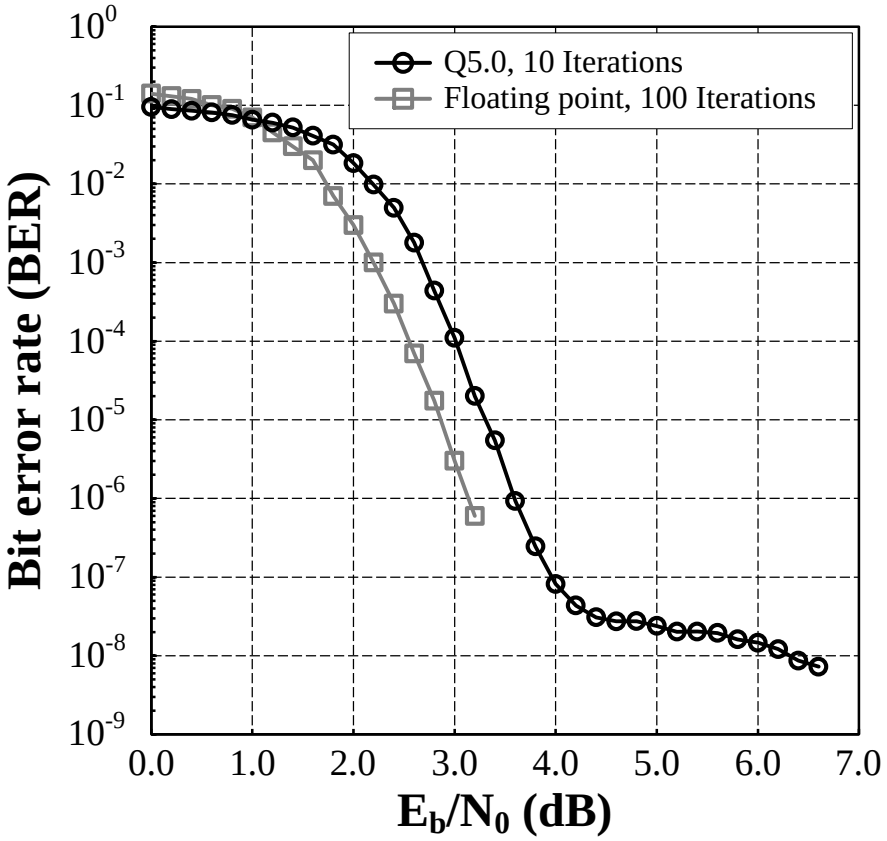


Figure 2.16: BER performance of the (672, 336) LDPC code for the IEEE 802.11ad standard using a 5-bit quantization with 10 decoding iterations and floating point with 100 iterations.

model the communication channel and provide input vectors in real time. Decoding errors are collected on-chip to compute the BER and FER.

The decoder supports two test modes: a scan mode for debugging and an automated mode for gathering error statistics. In the scan mode, input vectors are fed through scan chains and the decoding decisions are scanned out for inspection. In the automated mode, the decoder takes inputs from the on-chip AWGN generators, and decoding decisions are checked on-chip for errors. The AWGN noise variance and scaling factors are tuned to provide a range of SNR. We step through a number of SNR points and collect sufficient error statistics to plot BER against SNR waterfall curves. The waterfall curves are checked against the reference waterfall curve obtained by software simulation.

2.6.1 Chip Measurements

The test chip operates over a wide range of clock frequencies from 30 MHz up to 540 MHz, which translate to a throughput from 0.5 Gb/s up to 9 Gb/s using a fixed 10 decoding iterations. Early termination is built-in to increase throughput at high SNR if needed. The decoder BER is shown in Fig. 2.16. An excellent error-correction performance is achieved down to a BER of 10^{-7}, which is sufficient for the application.

Fig. 2.17 shows the measured power consumption of the decoder chip, the core and the eDRAM arrays at each clock frequency. The decoder consumes 38 mW, 106 mW, and 374 mW to achieve a throughput of 1.5 Gb/s, 3 Gb/s, and 6 Gb/s, respectively, at the optimal core and memory supply voltages listed in Table 2.1. The power consumption of the non-refresh eDRAM increases almost linearly with frequency compared to the quadratic increase in core logic power, demonstrating the advantage of the eDRAM at high frequency. At 6 Gb/s, the eDRAM consumes only 23% of the total power, and the proportion is further reduced to 21% at 9 Gb/s. The power consumption over the SNR range of interest is shown in Fig. 2.18. The power is the highest when the decoder is operating near the middle of the waterfall region, a result of high switching activities. The power decreases in the high SNR region due to the improved channel condition that leads to fewer switching activities.

2.6.2 Comparison with State-of-the-Art

The three metrics of an LDPC decoder implementation are throughput, power and silicon area. Two efficiency measures can be derived based on the three metrics: power/throughput (in pJ/b) gives energy efficiency, and throughput/area (in b/s/mm^2) gives area efficiency. Table 2.2 summarizes the results of the test chip along with other state-of-the-art LDPC decoders published in the last three years. For a fair comparison, we normalize the throughput to 10 iterations for a flooding decoder and 5 iterations for a layered decoder that converges faster.

As Table 2.2 shows, our results have advanced the state of the art by improving the best energy efficiency to 21 pJ/b in the low power mode and the best area efficiency to 5.63 Gb/s/mm^2 in the high performance mode. We provide a range of operating points in

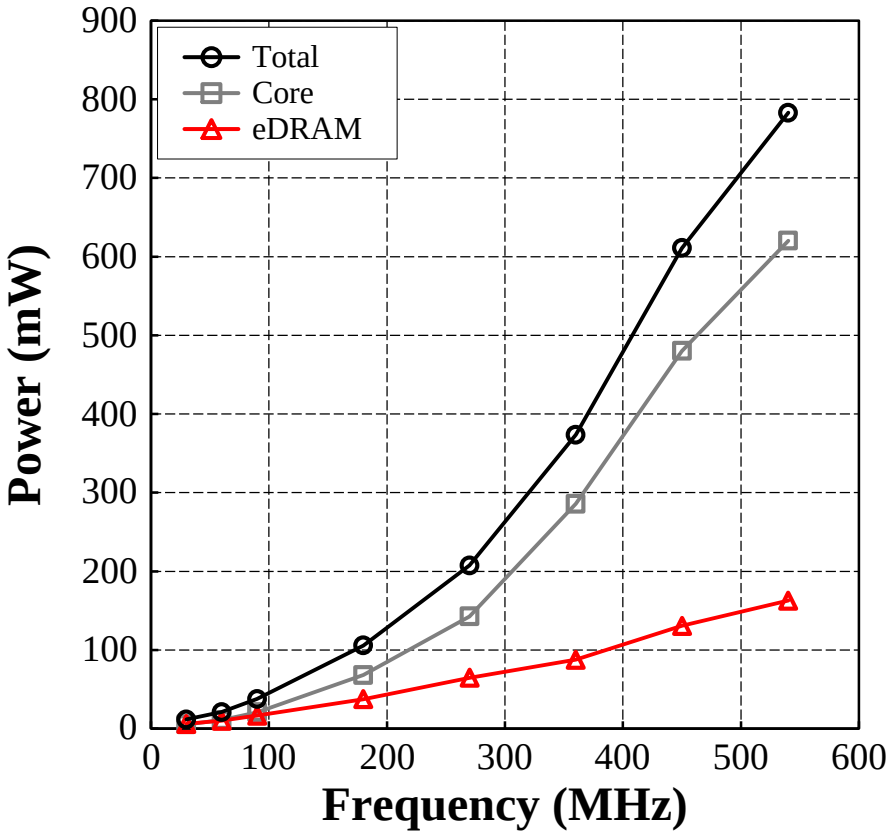


Figure 2.17: Measured LDPC decoder power at 5.0 dB SNR and 10 decoding iterations. The total power is divided into core and eDRAM power. Voltage scaling is used for the optimal core and eDRAM power.

Table 2.1: Measurement summary of the LDPC decoder at 5.0 dB SNR and 10 decoding iterations

Frequency (MHz)		30	60	90	180	270	360	450	540
Core	Supply (V)	0.41	0.45	0.51	0.64	0.76	0.94	1.06	1.15
	Power (mW)	5.6	11.0	21.0	68.2	142.8	285.8	480.1	620.1
eDRAM	Supply (V)	0.69	0.73	0.80	0.92	1.03	1.11	1.22	1.30
	Power (mW)	6.2	10.2	16.7	37.6	64.8	87.8	130.8	162.8
Total Power (mW)		11.8	21.2	37.7	105.8	207.6	373.6	610.9	782.9
eDRAM Fraction (%)		52	48	44	36	31	23	21	21
Throughput (Gb/s)		0.5	1.0	1.5	3.0	4.5	6.0	7.5	9.0
Energy Efficiency (pJ/bit)		21.0	21.9	35.6	34.5	44.8	61.7	76.4	89.5
Area Efficiency (Gb/s/mm^2)		0.31	0.63	0.94	1.88	2.81	3.75	4.69	5.63

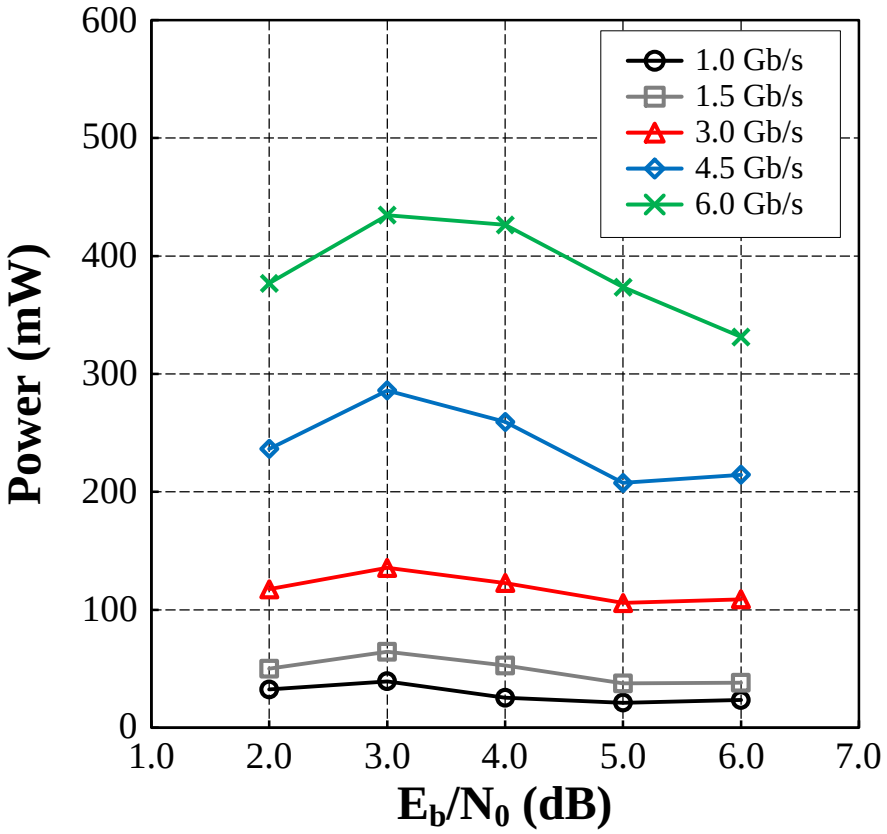


Figure 2.18: Measured LDPC decoder power across SNR range of interest at 10 decoding iterations. Voltage scaling is used for optimal core and eDRAM power.

Table 2.2: Comparison of state-of-the-art LDPC decoders

	This Work			JSSC'12 [19]	JSSC'11 [18]		JSSC'10 [20]		ASSCC'11 [33]	ASSCC'10 [32]	ASSCC'10 [24]	
Technology	65nm			65nm	130nm		65nm		65nm	90nm	90nm	
Block Length	672			672	576-2304		2048		576-2304	648-1944	2048	
Code Rate	1/2			1/2-7/8	1/2-5/6		0.84		1/2-5/6	1/2-5/6	0.84	
Decoding Algorithm	Offset Min-Sum			Layered Normalized Min-Sum	Layered Normalized Min-Sum		Offset Min-Sum		Layered Offset Min-Sum	Layered Offset Min-Sum	Layered Offset Min-Sum	
Core Area (mm^2)	1.60			1.56	3.03		5.35		3.36	1.77	5.35	
Iterations	10			5	10		8		10	10	4	
Input Quantization (bit)	5			6	6		4		6	5	7	
Core Supply (V)	0.41	0.94	1.15	1.0	1.2		0.7	1.2	1.2	1.0	0.8	1.2
Memory Supply (V)	0.69	1.11	1.30									
Clock Frequency (MHz)	30	360	540	197	214		100	700	110	346	84.7	137
Throughput (Gb/s)	0.5	6.0	9.0	5.79	0.874	0.955	6.67[a]	47.7[a]	1.056	0.679	7.23	11.69
Power (mW)	11.8	373.6	782.9	361	342	397	144	2800	115	107.4	386.8	1559
Norm. Throughput (Gb/s)[b]	0.5	6.0	9.0	5.79	1.748	1.91	2.335[c]	16.695[c]	2.112	1.36	5.784	9.352
Norm. Energy Eff. (pJ/bit)[d]	21.0	61.7	89.5	62.4	195.7	207.9	61.7	167.7	54.9	79	66.9	166.7
Norm. Area Eff. ($Gb/s/mm^2$)[d]	0.31	3.75	5.63	3.70	0.58	0.63	0.44	3.12	0.63	0.77	1.08	1.75

[a] Early termination enabled.
[b] Throughput is normalized to 10 decoding iterations for flooding decoders and 5 decoding iterations for layered decoders.
[c] Early termination requires an average of 2.5 iterations at a 5.5 dB SNR. One additional iteration is needed for convergence detection. [20]
[d] Energy and area efficiency are normalized to 65 nm, 1.0V and computed based on the normalized throughput.

Table 2.1 to show the tradeoff space between energy efficiency and area efficiency.

2.7 Summary

We present a low-power logic-compatible eDRAM design for a high-throughput LDPC decoder. The eDRAM retains storage for the necessary data access window, eliminating refresh for a significant power reduction. A new 3T LVT NMOS eDRAM cell design trades off the excessive retention time for a fast 0.68 ns read access at 1.0 V. To ensure a reliable storage, the coupling noise is mitigated by balancing the write and read wordline coupling, and the subthreshold leakage is minimized by a negative write wordline.

A row-parallel LDPC decoder is designed using 32 5 $\times$ 210 non-refresh eDRAM arrays for the (672, 336) LDPC code suitable for the IEEE 802.11ad standard. We use row merging and dual-frame processing to increase hardware utilization and remove pipeline stalls, resulting in a significant reduction of the clock frequency from 1.07 GHz to 360 MHz. The 1.6 mm^2 65 nm LDPC decoder test chip achieves a peak throughput of 9 Gb/s at 89.5 pJ/b, of which

only 21% is spent on eDRAMs. With voltage and frequency scaling, the energy efficiency is improved to 35.6 pJ/b for a 1.5 Gb/s throughput.

CHAPTER III

Nonbinary LDPC Decoder with Dynamic Clock Gating

3.1 Decoding Algorithm

The complexity of the NB-LDPC decoder and its error-correcting performance are determined by the code construction. Quasi-cyclic LDPC codes have been invented to provide a good error-correcting performance [65, 66, 67], and their regular structures are amenable to efficient decoder architectures [35, 36, 37, 38, 41]. An equally good error-correcting performance can be achieved with a class of regular (2, d_c) codes constructed based on the algebraic properties of their binary images [68]. Compared to the quasi-cyclic LDPC codes, the (2, d_c) codes feature a very low variable node degree $d_v = 2$, and a check node degree as low as $d_c = 4$, reducing the processing complexity, the wiring, and the quantization loss. Therefore, the (2, d_c) code is attractive for a practical and efficient implementation. An NB-LDPC code offers a competitive error-correcting performance even at a short block length. The performance can be further improved by increasing q, the order of the GF field, but higher q increases the size and complexity of the decoder.

The direct implementation of the BP decoding algorithm results in a check node complexity of $O(q^2)$ and a variable node complexity of $O(q)$. A fast Fourier transform (FFT) implementation [69] reduces the check node complexity to $O(q \log q)$, but it requires check node processing in the linear domain and the conversion between linear- and log-domain messages. The extended min-sum (EMS) algorithm [70] in the log domain simplifies the check node complexity to $O(qn_m)$ using only a small subset of n_m values among an array of q LLRs in a message, where $n_m \ll q$. A further simplification of the EMS algorithm trun-

cates the least significant values in a message and keeps only the most significant n_m values in memory [71]. The processing is done entirely using the truncated messages, thereby reducing the complexity of the check node to $O(n_m \log n_m)$ and variable node to $O(n_m)$. The truncated EMS algorithm has demonstrated minimal loss in error-correcting performance at low SNR compared with BP, while the performance surpasses BP at high SNR [71]. The truncated EMS algorithm makes it possible to design an NB-LDPC decoder with a reasonable complexity that is within the range of binary LDPC decoders. A further simplification using the min-max algorithm [72] incurs a noticeable degradation in the error-correcting performance.

Similarly to LDPC code, an NB-LDPC code is decoded using belief propagation (BP) by iteratively passing messages between VNs and CNs over the factor graph. Compared to a binary LDPC code, the factor graph of an NB-LDPC code is more compact with fewer nodes and much fewer edges, suggesting a simpler wiring in its decoder implementation. However, grouping $\log_2 q$ binary bits to a GF(q) symbol expands the message memory from $\log_2 q$ words to q words. The truncated EMS algorithm [71] reduces the message memory to n_m ($n_m < q$) words, e.g., a GF(64) NB-LDPC code can be decoded using $n_m = 16$, requiring 16 words in message storage, but still higher than what is needed in a binary LDPC decoder.

The following section describes the truncated EMS decoding algorithm that will be used as basis for the proposed NB-LDPC decoder. The VN to CN message will be referred to as the V2C message, or $U_{j,i}$ (from v_j to c_i); and the CN to VN message as the C2V message, or $V_{i,j}$ (from c_i to v_j).

3.1.1 VN Initialization

The decoding starts by initializing each VN with the prior LLRs based on the information received from the communication channel. Because each VN in an NB-LDPC code represents a GF(q) element, the prior LLR for a VN v_j, L_j, is an LLR vector (LLRV) of

length q, and each element of the LLRV corresponds to a GF(q) element α_i, $i \in \{1, \ldots, q\}$.

$$
\begin{aligned}
&L_j = [L_j(1), L_j(2), \ldots, L_j(q)], \text{where} \\
&L_j(i) = \log \frac{P(v_j = \widehat{\alpha}_i | y)}{P(v_j = \alpha_i | y)}, \text{and } \widehat{\alpha}_i = \{\arg\max_{\alpha_i \in GF(q)} P(v_j = \alpha_i | y)\}.
\end{aligned} \tag{3.1}
$$

and y is the channel information. $\widehat{\alpha}_i$ is the GF(q) element with the maximum $P(v_j = \alpha_i|y)$. Therefore a lower LLR value suggests a higher likelihood of the GF(q) element. The GF index vector associated with the prior LLRV is $L_j^{gf} = [1, 2, \ldots, q]$. In the following, we assume that the LLRV is sorted in ascending order unless specified otherwise, and the GF index vector is used to track the GF element that corresponds to each entry of the LLRV. In the GF index vector, each GF(q) element is stored in its $\log_2 q$-bit binary representation. An example is shown in Fig. 1.4. Using the truncated EMS algorithm, only the minimum n_m entries, $n_m < q$, in the LLRV are kept. In the first decoding iteration, the prior LLRV is used as the V2C message, i.e., $U_{j,i} = L_j$.

3.1.2 CN Operation

Each GF element α_k in the GF index vector of the V2C message $U_{j,i}$ is multiplied by $H(i, j)$ before the message is sent to the CN. α_k is stored in the binary representation and $H(i, j)$ is known, so the GF(q) multiplication is described by a q-entry lookup table and synthesized to logic gates. This GF multiplication is known as permutation.

Suppose a CN receives messages from d_c VNs, v_j, $j \in \{1, 2, \ldots, d_c\}$, where d_c is the degree of the CN. The CN computes the C2V messages for each VN using the forward-backward algorithm in three steps: forward, backward, and merge that are illustrated in Fig. 3.1. The forward and backward steps can be carried out in parallel.

As Fig. 3.1 shows, in the forward step, the message from v_1 is combined with the message from v_2, and message combining continues until reaching the message from v_{d_c-2} following equation (3.2a). The "combine" operation is known as the elementary CN (ECN) that is represented by $\oplus$ in equation (3.2a). An ECN takes two length-n_m LLRV inputs, e.g., U_1 and U_2, and calculates a length-n_m LLRV output $U_{1:2}$ that contains the n_m minimum values in the set $\{U_1(i) + U_2(j), i \in [1, n_m], j \in [1, n_m]\}$. An ECN requires an insertion sorter of

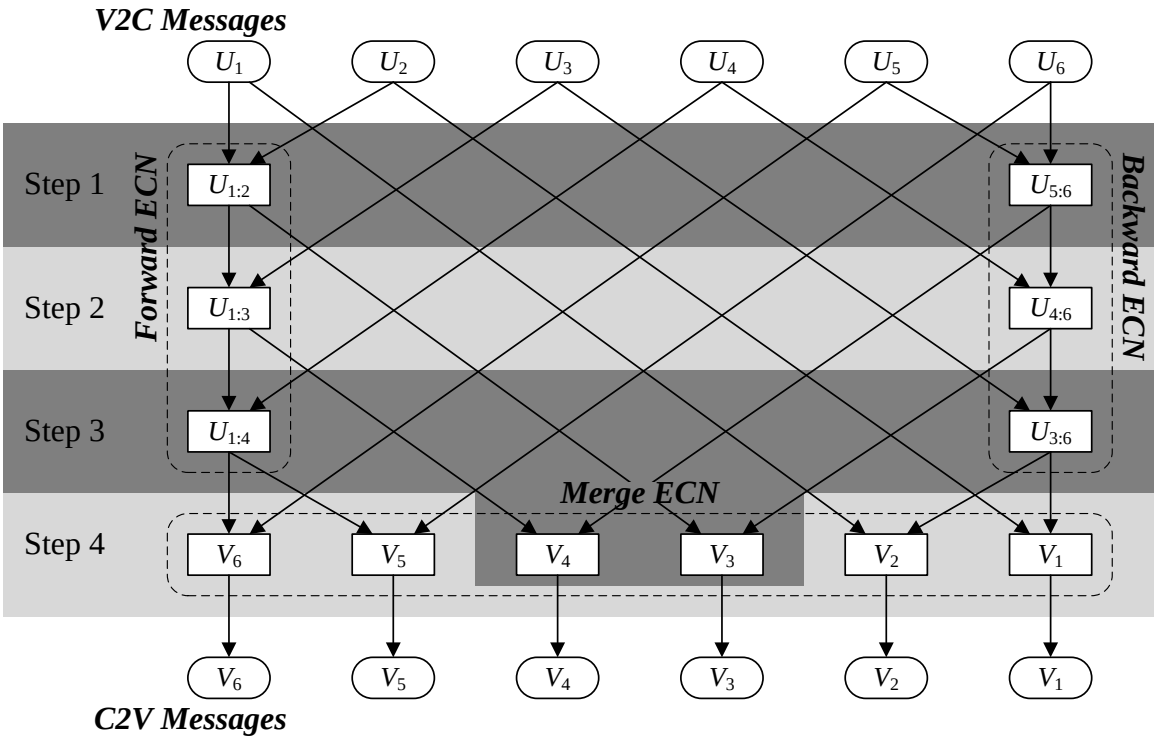


Figure 3.1: Illustration of forward-backward algorithm with $d_c = 6$.

length n_m, and the complexity of the ECN is $O(n_m^2)$. An efficient bubble check algorithm [73] reduces the insertion sorter length to $\lceil \frac{1+\sqrt{1+8(n_m-1)}}{2} \rceil$ and the operation complexity to $O(n_m\sqrt{n_m})$. The forward step requires $d_c - 3$ ECNs.

$$\text{Forward: } U_{1:j+1} = U_{1:j} \oplus U_{j+1}, j = 1, \ldots, d_c - 3 \ (U_{1:1} = U_1). \tag{3.2a}$$

$$\text{Backward: } U_{j-1:d_c} = U_{j:d_c} \oplus U_{j-1}, j = d_c, \ldots, 4 \ (U_{d_c:d_c} = U_{d_c}). \tag{3.2b}$$

$$\text{Merge: } V_j = U_{1:j-1} \oplus U_{j+1:d_c}, j = 2, \ldots, d_c - 1. \tag{3.2c}$$

The backward step follows equation (3.2b), and it is identical to the forward step, except that it is done in the reverse direction, as shown in Fig. 3.1. After the forward and backward are complete, the C2V messages can be readily calculated by merging the messages obtained from the forward and backward, as described by equation (3.2c) and illustrated in Fig. 3.1. Merge requires d_c ECNs. To sum up, the forward-backward algorithm for CN requires

$3d_c - 6$ ECNs in total, and each ECN is of complexity $O(n_m\sqrt{n_m})$.

Each GF element α_k in the GF index vector of the C2V message $V_{i,j}$ is divided by $H(i,j)$ before the message is sent to the CN. α_k is stored in the binary representation and $H(i,j)$ is known, so the GF(q) division is described by a q-entry lookup table and synthesized to logic gates. This GF division is known as inverse permutation.

3.1.3 VN Operation

Each VN receives d_v C2V messages and computes the posterior LLR, L_j^{post}, and the V2C messages following equation (3.3).

$$L_j^{post} = L_j + \sum_{i'=1}^{d_v} V_{i',j},\ U_{j,i} = L_j + \sum_{i'=1,i'\neq i}^{d_v} V_{i',j}. \tag{3.3}$$

Note that the operator $+$ and $\sum$ are not ordinary addition and summation. They represent pair-wise elementary VN (EVN). An EVN takes two length-n_m LLRV inputs, V_1 and V_2, and calculates a length-n_m LLRV output V_3 that contains the n_m minimum values in the set $\{V_1(i) + V_2(j), V_1^{gf}(i) = V_2^{gf}(j), i \in [1, n_m], j \in [1, n_m]\}$. The EVN requires matching of GF index, which is done using a content-addressable memory (CAM). An EVN uses an insertion sorter of length n_m, and the complexity of the EVN is $O(2n_m)$. The VN makes a hard decision in each iteration based on the most likely GF element. If the hard decisions of all VNs meet all parity checks defined by the H matrix, decoding terminates.

The VN and CN operations in an NB-LDPC decoder as described above are more complex than a binary LDPC decoder. The CN of a binary LDPC decoder performs compare select and XOR in a tree structure of complexity $O(d_c)$, thus the CN can be easily parallelized for a high throughput. The CN of an NB-LDPC decoder performs forward, backward and merge with a complexity of $O(d_c n_m\sqrt{n_m})$ using the truncated EMS algorithm [71] with bubble check ECN [73]. The VN of an NB-LDPC decoder is also more complex than the VN of a binary LDPC decoder, with a complexity of $O(d_v n_m)$ compared to $O(d_v)$. For practical implementations of NB-LDPC decoders, the CN and VN operations have to be serialized, resulting in a lower throughput. The larger memory, expensive sorters and CAMs all contribute to larger VNs and CNs.

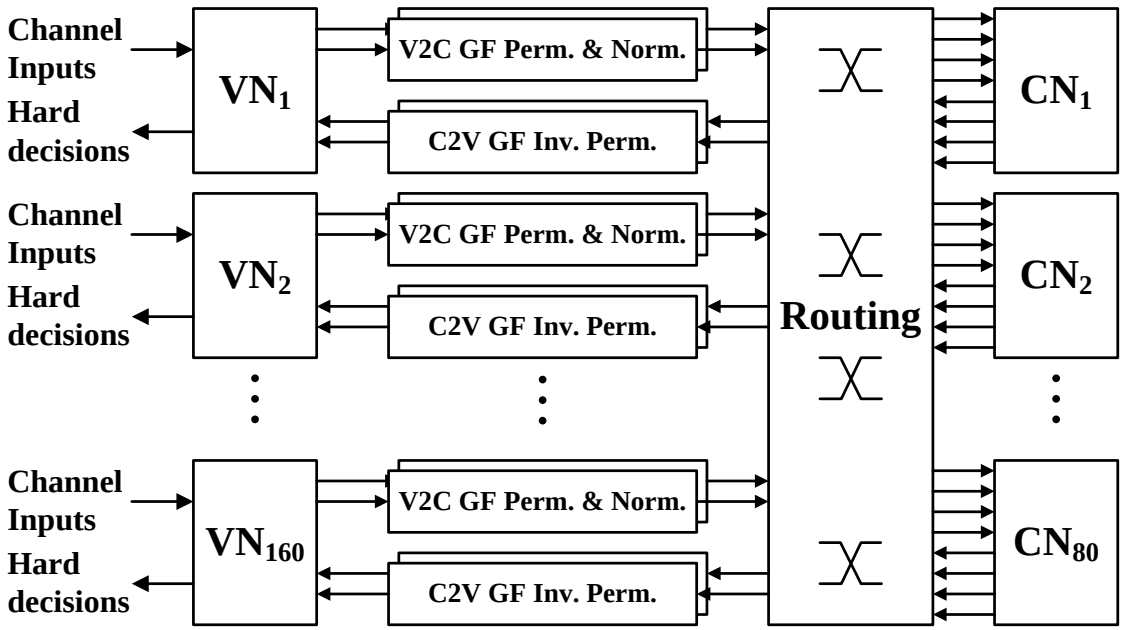


Figure 3.2: Architecture of the fully parallel nonbinary LDPC decoder.

3.2 High-Throughput Fully Parallel Decoder Architecture

The NB-LDPC decoder is heavy on logic and memory but low on wiring compared to the binary LDPC decoder. A parallel implementation of NB-LDPC decoder does not incur the same wiring overhead seen in the implementations of binary LDPC decoder. A fully parallel implementation simplifies the control and message scheduling, leading to a more efficient design.

The GF(64) (160, 80) regular-(2, 4) NB-LDPC code constructed based on the algebraic properties of their binary images [68] features low VN and CN degrees, thus the complexity of VN and CN can be kept low. The block diagram of the fully parallel decoder is illustrated in Fig. 3.2. The 960 bits of a codeword are grouped into 160 6-bit GF(64) symbols. The factor graph of the code contains 160 VNs and 80 CNs. The fully parallel decoder is the direct mapping of the factor graph with 160 2-input VNs and 80 4-input CNs as shown in Fig. 3.2. Each edge in the factor graph carries an LLRV. The entries of the LLRV are sent serially to reduce the bit width of the wires and to match the pipelined CN and VN processing. Permutation and inverse permutation are placed between the VNs and

CNs, and messages are normalized in each iteration to prevent saturation. The messages are quantized to 5 bits to minimize storage. The decoder implements the truncated EMS algorithm with $n_m = 16$. The word length and truncated EMS setting have been simulated extensively to ensure a good error-correcting performance down to very low BER levels.

We further improve the throughput of the fully parallel decoder using architecture transform and scheduling techniques: (1) by applying a one-step look-ahead to the ECN bubble check algorithm, we remove the data dependency to produce a fast ECN design; (2) by dividing the ECN and EVN schedules into two phases, we allow the interleaving of VN and CN for a short iteration latency.

3.2.1 Look-Ahead Elementary Check Node

CN takes 4 V2C messages, U_1, U_2, U_3, U_4, and computes 4 C2V messages, V_1, V_2, V_3, V_4, using the forward-backward algorithm illustrated in Fig. 3.1. The forward step takes U_1 and U_2 to compute $U_{1:2}$; and concurrently, the backward step takes U_4 and U_3 to compute $U_{3:4}$. Next, the four merges are done in parallel to compute V2C messages, as illustrated in Fig. 3.3. The forward step, backward step, and merge are all done using ECN.

ECN implements the bubble check algorithm to find the n_m minimum values in the set $T_\Sigma = \{U_1(i) + U_2(j), i \in [1, n_m], j \in [1, n_m]\}$, where U_1 and U_2 are two input LLRVs. The set T_Σ is represented in a 2-dimensional matrix. The entries of T_Σ are computed on the fly by reading one entry from $U_1(i)$ and one from $U_2(j)$ and summing them. The corresponding GF element of the sum is computed by adding the GF element associated with the entry $U_1(i)$ and the GF element associated with the entry from $U_2(j)$. Since the pair of GF elements are stored in binary representation, the addition is done by the bitwise logical XOR of the pair. ECN uses an insertion sorter of length $n_b = 6$ for $n_m = 16$. The ECN sorter is initialized with $T_\Sigma(1,1)$, $T_\Sigma(2,1)$, ... $T_\Sigma(6,1)$. The ECN sorter outputs the minimum entry, e.g., $T_\Sigma(i_1, j_1)$, every step and a new entry $T_\Sigma(i_n, j_n)$ is inserted. ECN is complete after n_m steps. Note that we allow duplicate GF outputs because it simplifies the control logic and ensures a constant latency per iteration. Our simulation results show that the loss in error-correcting performance due to duplicate GF outputs is negligible.

Using bubble check [73], the new entry from T_Σ to be inserted to the sorter is determined

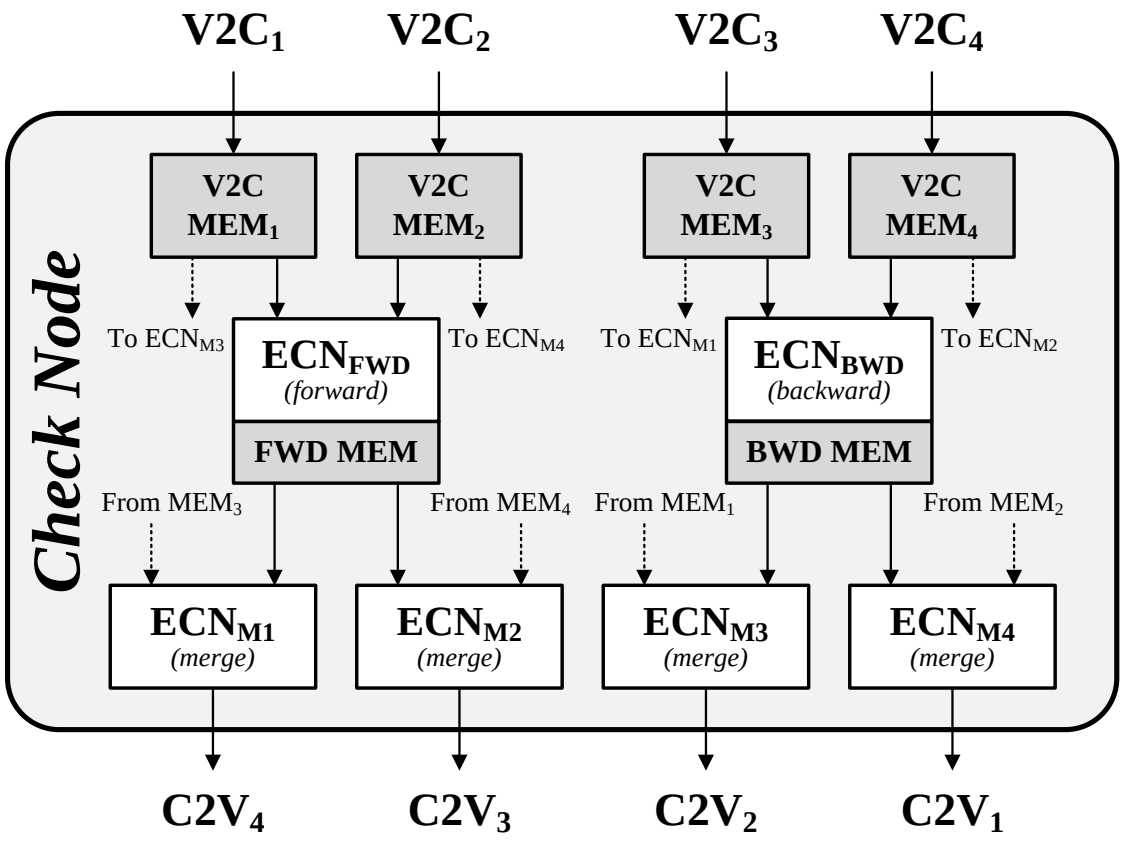


Figure 3.3: Architecture of the check node.

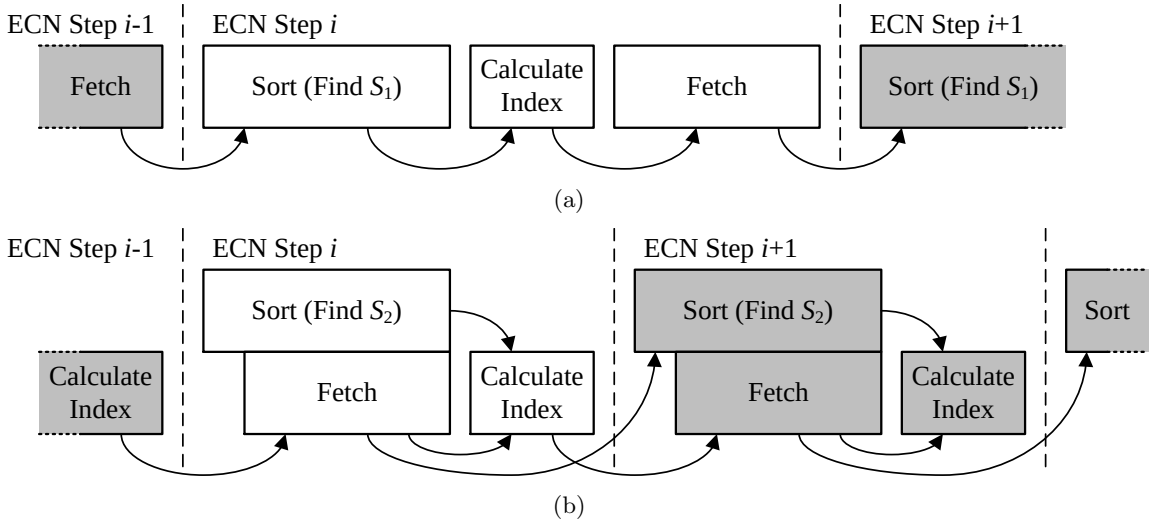


Figure 3.4: Sub-operation schedule of (a) the bubble check elementary check node and (b) the look-ahead elementary check node.

based on the minimum entry in the sorter. Each ECN step consists of three substeps as illustrated in Fig. 3.4(a): (1) sort: find the minimum entry in the sorter $T_\Sigma(i_1, j_1)$; (2) bubble check: calculate the index of the new entry (i_n, j_n) in T_Σ to be inserted to the sorter based on the bubble check algorithm using a "horizontal flag" H described below [73]; and (3) fetch: read $U_1(i_n)$ and $U_2(j_n)$, calculate the sum and insert it to the sorter. Each substep depends on the previous one: fetch depends on sort for the index of the new entry; and sort depends on fetch for the new entry. The data dependency requires that the three substeps to be done in series, which results in a long clock period $T = t_{sort} + t_{bubble} + t_{fetch}$, where t_{sort}, t_{bubble}, and t_{fetch} are the maximum time needed for the sort, bubble check and fetch.

if $i_1 = 1$ **then**

$H = 1$, $\bar{H} = 0$

end if

if $j_1 = 1$ **and** $i_1 \geq n_b$ **then**

$H = 0$, $\bar{H} = 1$

end if

if $T_\Sigma(i_1 + \bar{H}, j_1 + H)$ has never been inserted to the sorter **then**

$i_n = i_1 + \bar{H}$, $j_n = j_1 + H$

else

$i_n = i_1 + H$, $j_n = j_1 + \bar{H}$

end if

We apply one-step look-ahead to shorten the clock period. The new sorter keeps track of not only the minimum $T_\Sigma(i_1, j_1)$, but also the second minimum $T_\Sigma(i_2, j_2)$. With this change, each ECN step is done in three substeps that can be partially overlapped: (1) sort: find the second minimum $T_\Sigma(i_2, j_2)$ (the minimum $T_\Sigma(i_1, j_1)$ is found in the previous ECN step); (2) fetch: read $U_1(i_n)$ and $U_2(j_n)$, calculate the sum and insert to the sorter; (3) bubble check: compare $T_\Sigma(i_2, j_2)$ with $T_\Sigma(i_n, j_n)$, one of which will be the new minimum $T_\Sigma(i_1, j_1)$ to be output next, and the index of the new entry (i_n, j_n) is calculated based on the bubble check algorithm above. Though the three substeps still remain, the look-ahead design allows sort and fetch to be done in parallel. The new sequence illustrated in Fig. 3.4(b) allows the

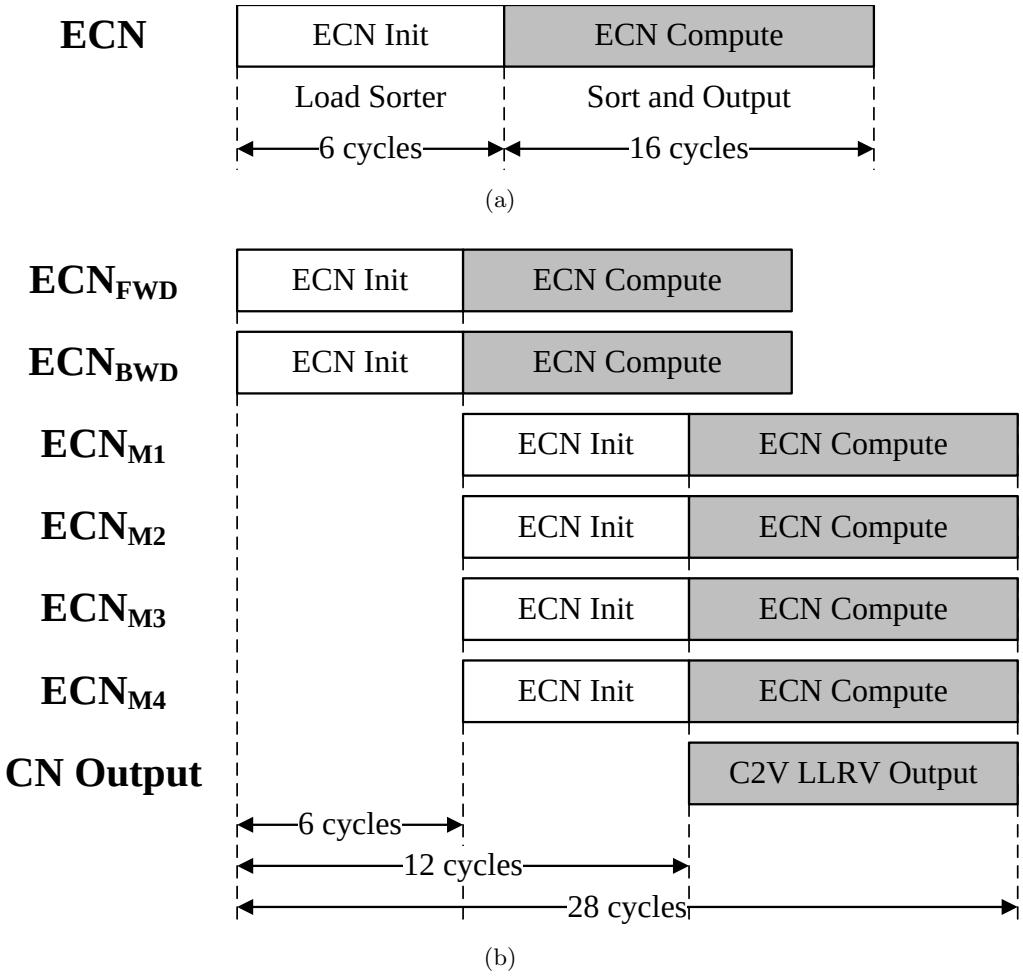


Figure 3.5: Operation schedule of (a) the elementary check node and (b) the check node.

overlapping of the substeps to shorten the clock period to $T = \max\{t_{sort}, t_{fetch}\} + t_{bubble}$. Since t_{sort},t_{fetch} are significantly longer than t_{bubble}. The clock period is almost halved compared to the baseline version.

The schedule of the ECN is divided into two phases: initialization phase and compute phase, according to Fig. 3.5(a). The initialization phase spans the first $n_b = 6$ cycles to initialize the sorter. The compute phase spans $n_m = 16$ cycles, during which ECN outputs one value every cycle. In the CN schedule shown in Fig. 3.5(b), the forward and backward ECNs are started at the same time. After the initialization phase, the forward and backward ECNs (ECN_{FWD}, ECN_{BWD}) move to the compute phase, while the four merge ECNs ($\text{ECN}_{\text{M1-4}}$) start their initialization phase. The phase pipelining shortens the

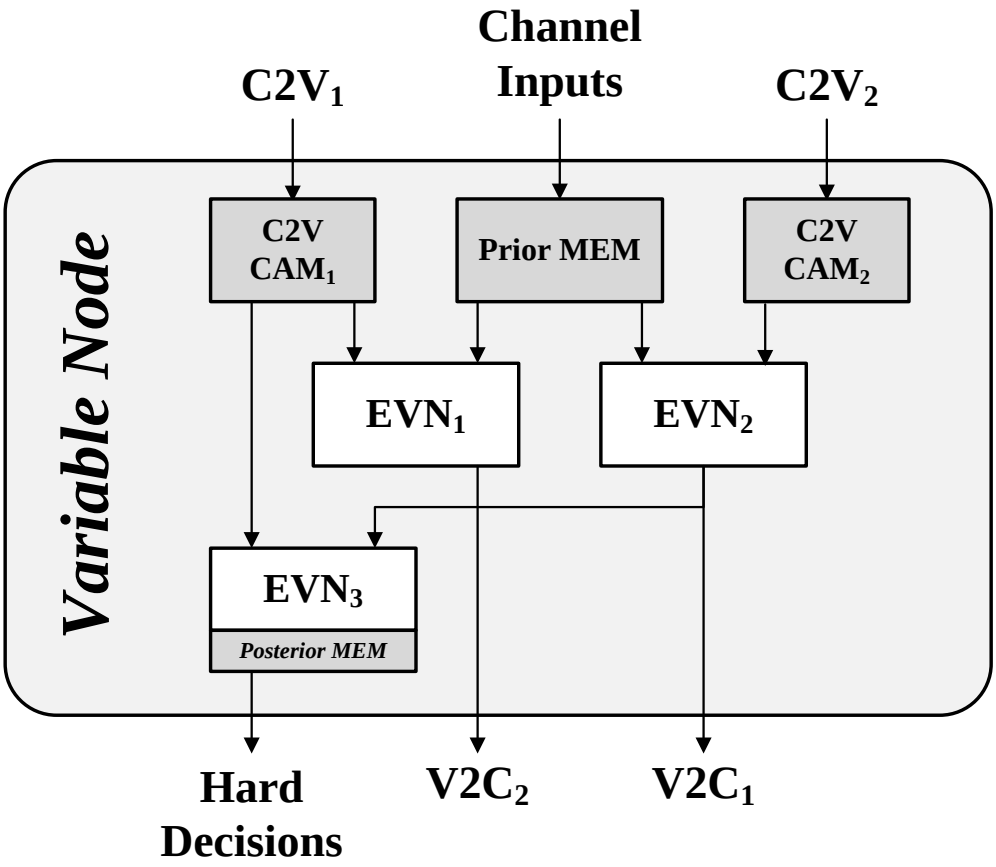


Figure 3.6: Architecture of the variable node.

latency of the CN to 28 cycles.

3.2.2 Two-Pass Variable Node

VN takes 2 C2V messages, V_1, V_2, and the prior LLRV to compute 2 V2C messages, U_1, U_2, and the posterior LLRV. The low VN degree of 2 simplifies the implementation, as shown in Fig. 3.6. Three EVNs are used: EVN_1 and EVN_2 start first to compute U_2 and U_1, followed by EVN_3. This design shortens the VN critical path, as EVN_3 has been excluded from the critical path.

EVN finds the n_m minimum values in the set $\{V_1(i)+V_2(j), V_1^{gf}(i) = V_2^{gf}(j), i \in [0, n_m - 1], j \in [0, n_m - 1]\}$, where V_1 and V_2 are two input LLRVs. The condition $V_1^{gf}(i) = V_2^{gf}(j)$ requires matching of GF indices. Therefore, one of the input LLRVs, e.g., V_2, is stored in a

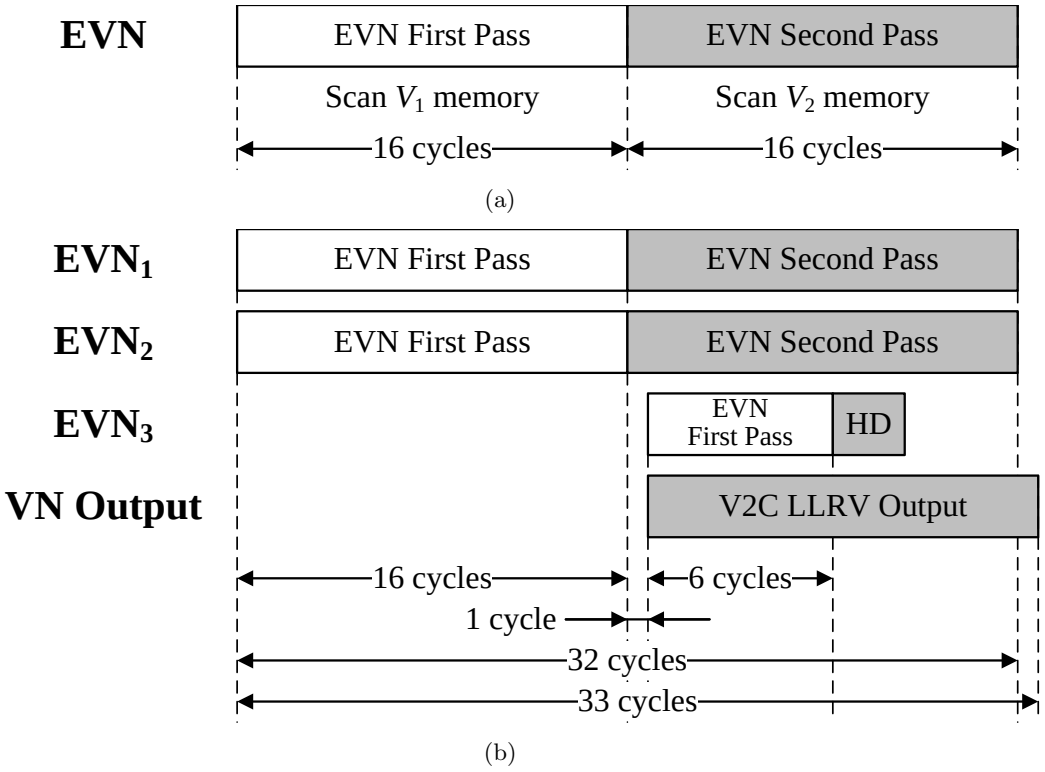


Figure 3.7: Operation schedule of (a) the elementary variable node and (b) the variable node. Note that EVN_3 uses a shorter sorter length since only the minimum is required.

content-addressable memory (CAM) to enable searching of the GF index. EVN implements a two-pass scan: (1) in the first pass, EVN scans V_1 memory, and searches matching GF index in V_2 memory. If a matching entry is found, e.g., $V_1^{gf}(i) = V_2^{gf}(j)$, the entry $V_2(j)$ is read to calculate $V_1(i) + V_2(j)$; if no matching entry is found, a fixed offset is added to $V_1(i)$ and the sum is inserted to the EVN sorter; (2) in the second pass, EVN scans V_2 memory. A fixed offset is added to $V_2(j)$ and the sum is inserted to the sorter. The insertion sorter performs a sort every cycle and keeps its stored items in ascending order.

To support the two-pass scan, the EVN sorter length is kept at least $n_m + 1$ to consider all n_m V_1 entries of the first pass and the first V_2 entry of the second pass. Simulations show that the EVN sorter length directly impacts the BER performance of the decoder. Therefore we choose the EVN sorter length $L_{EVN} = 17$ to avoid a degradation in BER. However, note that EVN_3 is different from EVN_1 and EVN_2 in that only the top (minimum) entry in the posterior LLRV determines the hard decision. We take advantage of this finding

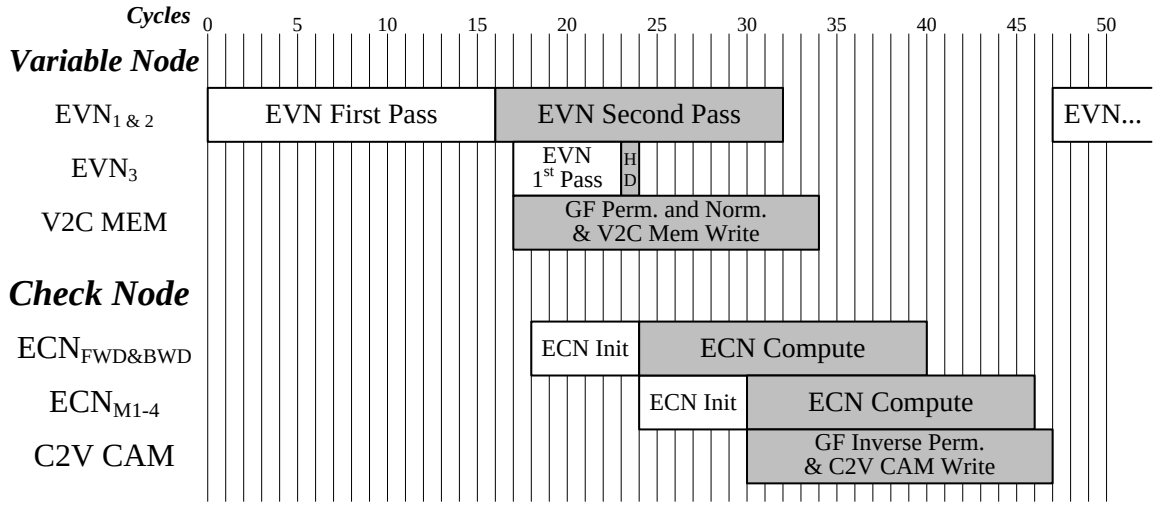


Figure 3.8: Operation schedule of the decoder which includes the variable node, check node, permutation & normalization, and inverse permutation stages.

to shorten the two passes performed by EVN_3 to scan only a small number of top entries in V_1 memory and V_2 CAM. The shortening of the two passes are verified by simulation to guarantee accurate hard decision, and EVN_3 is simplified.

The EVN schedule is divided into two phases: first pass and second pass, as in Fig. 3.7(a). In the VN schedule shown in Fig. 3.7(b), EVN_1 and EVN_2 start in parallel. The first pass takes $n_m = 16$ cycles, followed by n_m cycles for the second pass. EVN_1 and EVN_2 each outputs one value every cycle beginning from the second cycle of the second pass. After EVN_1 and EVN_2 start producing outputs, ECN_3 starts with a 6-cycle first pass and a 1-cycle second pass to obtain the hard decision.

3.2.3 Interleaving Check Node and Variable Node

The phased schedule of ECN and EVN allows the CN and VN to be interleaved for a shorter latency and higher throughput. The interleaved schedule is illustrated in Fig. 3.8 and it is executed in the following order: (1) EVN_1 and EVN_2 first pass, followed by second pass. In the second pass, each EVN outputs one entry of the V2C message per cycle to be permuted and forwarded to the CNs; (2) forward and backward ECN initialization, followed by merge ECN initialization and compute. Merge ECN outputs one entry of the C2V message per cycle in the compute phase to be inverse permuted and forwarded to

the VNs. EVN_1 and EVN_2 need to wait until all n_m entries of the C2V message to be ready before starting the next iteration because the two-pass scan requires one complete C2V message to be stored in the CAM for searching. The overall latency of one decoding iteration is 47 cycles according to Fig. 3.8. Note that EVN_3 calculates the posterior. The posterior is not needed by the ECN, therefore EVN_3 is not in the critical path and it can be overlapped with EVN operations.

3.3 Low-Power Design by Fine-Grained Dynamic Clock Gating

To estimate the power consumption, the fully parallel nonbinary LDPC decoder has been synthesized and place and routed in a 65 nm CMOS process. Fig. 3.9(a) shows the power breakdown of the decoder. The switching power of sequential circuits is the dominant portion, claiming 65% of the total power. The leakage power and the switching power of combinational circuits claim the remaining 21% and 14% of the total power, respectively. Further breakdown of the switching power of sequential circuits in Fig. 3.9(b) shows that the switching power of the VN and CN memories and the sorters in EVNs and ECNs account for almost all of the sequential switching power.

The high dynamic power consumption prompts us to design a dynamic clock gating strategy to reduce the power consumption of the decoder. Clock gating disables the clock input to sequential circuits to save switching power, which in turn cuts the switching of combinational circuits. The use of clock gating is motivated by the observation that the majority of the VNs converge within a few decoding iterations before reaching the decoding iteration limit. Therefore, it is possible to clock gate the VNs and CNs that have reached convergence to save power.

To achieve the most power savings, the clock gating is implemented at a fine-grained node level, i.e., at each VN and CN, and the clock gating is enabled dynamically during run time. The fine-grained dynamic clock gating requires convergence detection at the node level, i.e., each VN detects when it has reached convergence and can be clock gated. The node-level convergence detection is different from the conventional convergence detection done at the global level by checking whether all parity checks have been met [28]. Although

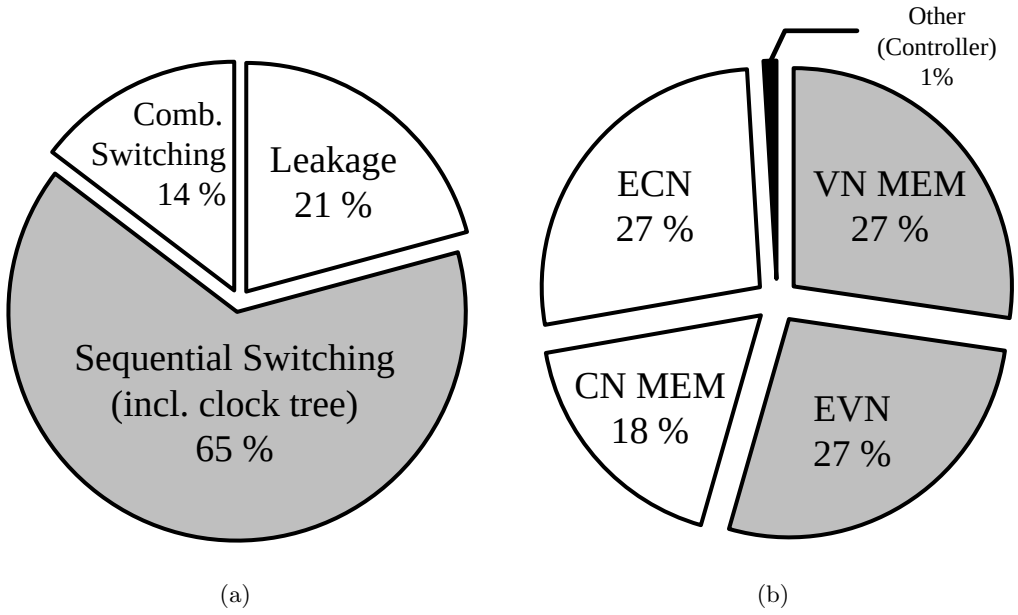


Figure 3.9: (a) Power breakdown of the 65 nm synthesized fully parallel nonbinary LDPC decoder, and (b) the distribution of sequential logic used in the decoder.

clock gating can also be based on global convergence detection, the power savings would be greatly diminished.

3.3.1 Node-Level Convergence Detection

Node-level convergence detection is not equivalent to global convergence detection. Our proposed node-level convergence detection is designed to match the accuracy of the global convergence detection without causing BER degradation. The node-level convergence detection is based on two convergence criteria: (1) meet the minimum number of decoding iterations M, and (2) VN's hard decisions remain unchanged for the last T consecutive iterations. The two criteria are designed to prevent false convergence and ensure stability. Each VN checks the criteria upon completing each decoding iteration. If the criteria are met, the VN is clock gated. If a VN is clock gated, parts of the CN that are used for storing and processing messages from and to the VN are also clock gated. CN is completely clock gated when all its connected VNs have been clock gated.

An example in Fig. 3.10 shows the proposed clock gating in action. In this example, we use decoding iteration limit $L = 30$ and convergence criteria $M = 10$ and $T = 5$. Fig. 3.10(a)

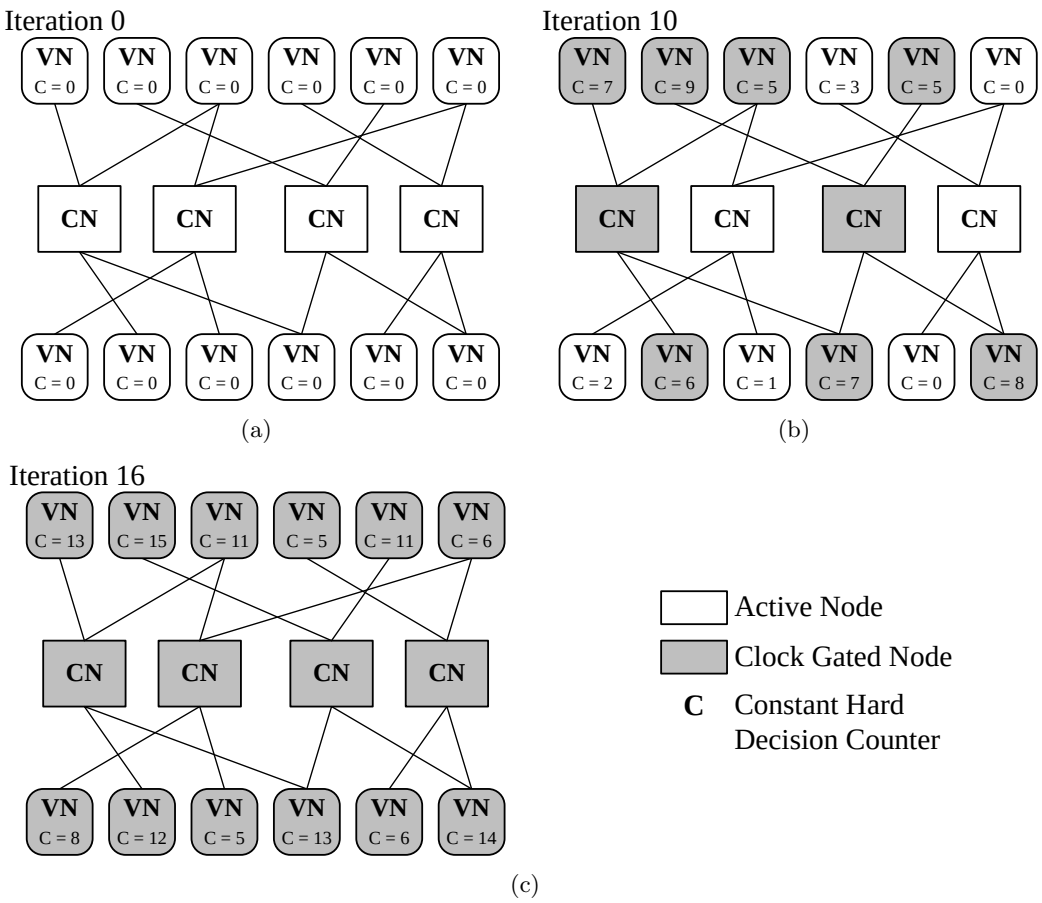


Figure 3.10: Example of clock gating showing active and clock gated nodes at different iterations during the decoding process of one frame.

shows the decoder state at the start of the first iteration where all VNs and CNs are active. Fig. 3.10(b) shows the decoder state at the start of iteration 11, when the minimum iteration criterion has been met, and some of the VNs have reached the same hard decisions over the last 5 or more iterations, therefore these VNs are clock gated. Two CNs are also completely clock gated because all of their connected VNs are clock gated. The remaining CNs are partially clock gated. In a few more iterations all VNs and CNs are clock gated, shown in Fig. 3.10(c), and the decoder only consumes leakage power.

Fine-grained dynamic clock gating can be compared to early termination [28, 22] that is commonly used in existing decoder designs. Early termination relies on global convergence detection, whereas fine-grained dynamic clock gating is based on node-level convergence detection, and it allows a large fraction of the VNs and CNs to be turned off before the

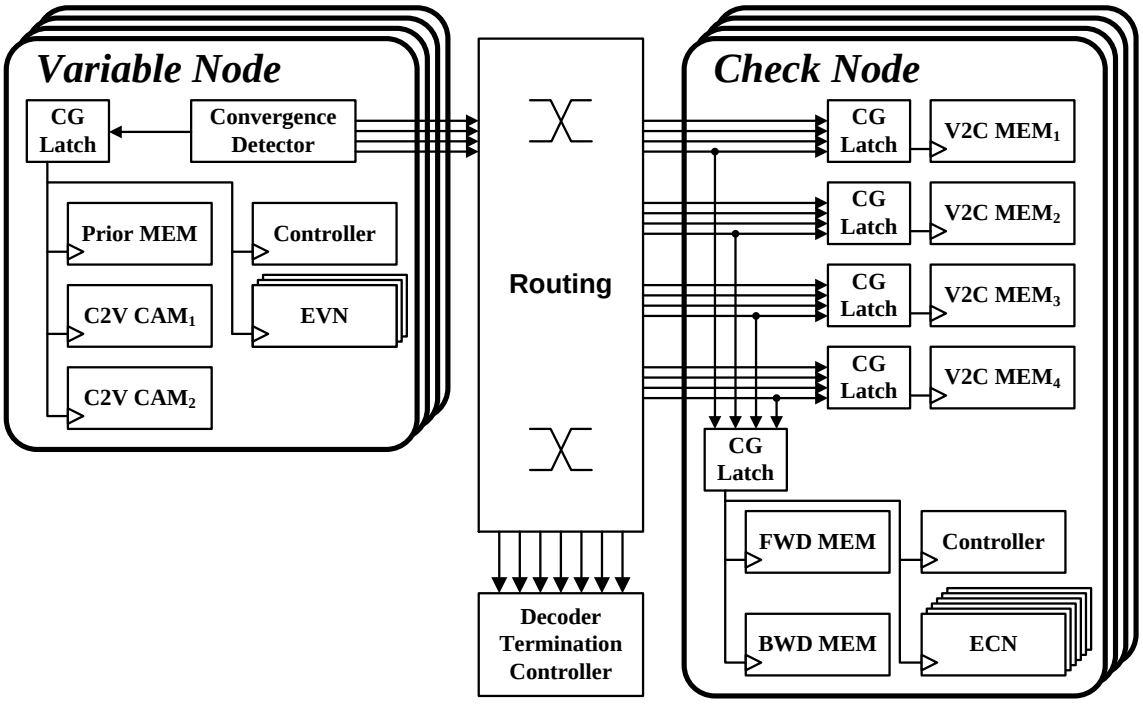


Figure 3.11: Implementation of fine-grained dynamic clock gating for the variable and check node.

global convergence is reached, therefore it saves significant power compared to early termination. The other difference is that early termination uses the excess decoding iterations to improve throughput, whereas fine-grained dynamic clock gating eliminates the dynamic power consumption of the excess decoding iterations to save power, and the throughput is kept constant.

The idea of early termination can be combined with fine-grained dynamic clock gating to save power and improve throughput by terminating the decoder once all the VNs and CNs are clock gated. We term the approach decoder termination to differentiate it from early termination, because decoder termination relies on node-level convergence detection, whereas early termination commonly relies on global convergence detection.

3.3.2 Fine-Grained Dynamic Clock Gating

The clock gating architecture is illustrated in Fig. 3.11. The convergence detector inside each VN monitors the hard decisions in each iteration to check whether the hard decisions

have changed between iterations. A counter keeps track of the number of consecutive iterations that the hard decisions have remained unchanged. When the convergence criteria are met, the convergence detector enables the clock gating latch (CG latch) to turn off the clock input to all sequential circuits with the exception of essential control circuits that are needed for recovering from the clock gating state. The majority of the VN's dynamic power is saved, leaving only leakage.

The convergence detector propagates the clock gating signal to the CNs to enable the CG latch of V2C message memories in the CNs, as noted in Fig. 3.11. Clock gating V2C memories eliminates the unnecessary memory updates to save dynamic power. In this way, CN is partially clock gated. When all the connected VNs are clock gated, as indicated by their clock gating signals, a central CG latch is enabled to completely turn off the CN.

A decoder termination controller monitors the VN clock gating signals. When all the VNs are clock gated (and CNs are clock gated as a result), decoder terminates the decoding of the current code frame and moves on to the next input code frame. Decoder termination reduces the average number of decoding iterations per code frame and therefore improves the decoding throughput for a net gain in energy efficiency.

In our implementation, each VN stores only the hard decision (6 bit) from the previous iteration. In each iteration, the VN compares the hard decision with the previous hard decision, and increments a 4-bit counter if they agree. If not, the counter is reset. After the comparison, the stored hard decision is replaced by the current hard decision for the next iteration. The node-level convergence detection requires only 6 bits of storage per VN (or 960 bits for the entire decoder), and a small logic in each VN to compare a pair of 6-bit decisions, and a 4-bit counter. Compared to the size of the non-binary VN and CN, the overhead for node-level convergence detection is negligible.

To check the effectiveness of fine-grained dynamic clock gating, we simulated the decoder's behavior with the fine-grained dynamic clock gating using node-level convergence detection. Fig. 3.12 shows the percentage of nodes that have been clock gated in each decoding iteration across various SNR levels. The decoding iteration limit is set to 30, and the convergence criteria are set to $M = 10$ and $T = 10$. Even at a low SNR (E_b/N_0) of 2.8dB, more than 85% of the VNs are clock gated after 12 iterations. After 14 iterations,

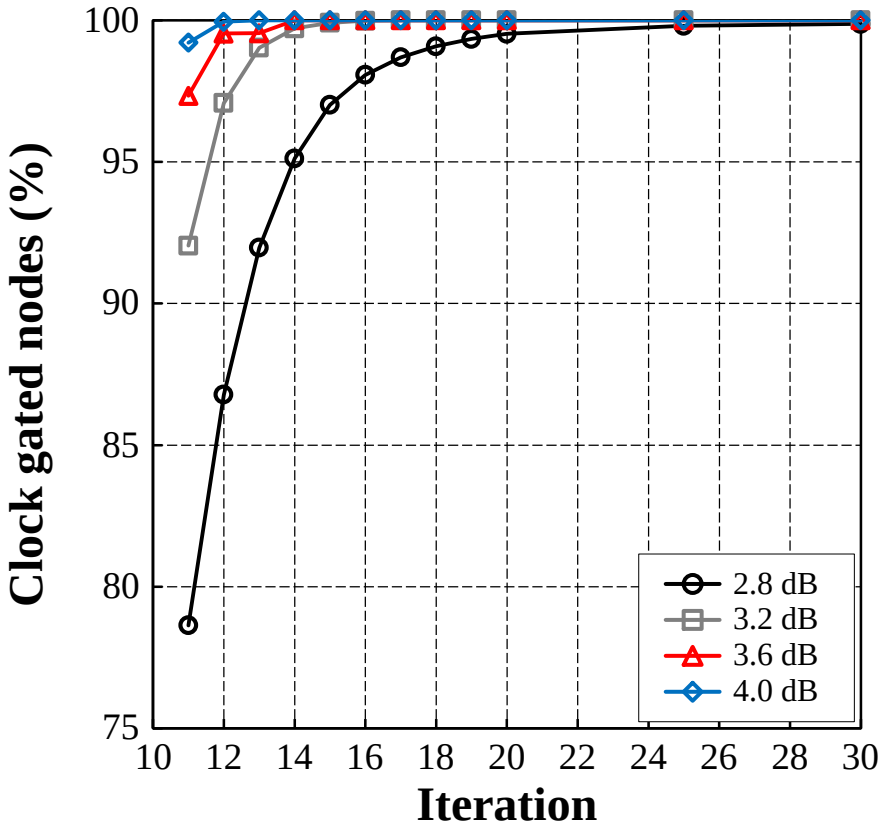


Figure 3.12: Cumulative distribution of clock gated nodes at each iteration for various SNR levels with a decoding iteration limit of 30. The parameters used for clock gating are $M = 10$ and $T = 10$.

95% of the VNs are clock gated. At higher SNRs, the VNs are clock gated at an even faster pace.

3.4 Decoder Chip Implementation and Measurement Results

A decoder test chip for the GF(64) (160, 80) regular-(2, 4) NB-LDPC code was implemented in a STMicroelectronics 65 nm 7-metal general-purpose CMOS technology [74]. The chip microphotograph is shown in Fig. 3.13. The test chip measures 4.40 mm $\times$ 2.94 mm, and the core measures 3.52 mm $\times$ 2.00 mm, or 7.04 mm^2. The memory used in this decoder is implemented using registers. The test chip incorporates AWGN generators to model the communication channel and provide input vectors in real time. An on-chip controller keeps track of the decoding errors for calculating the BER and FER.

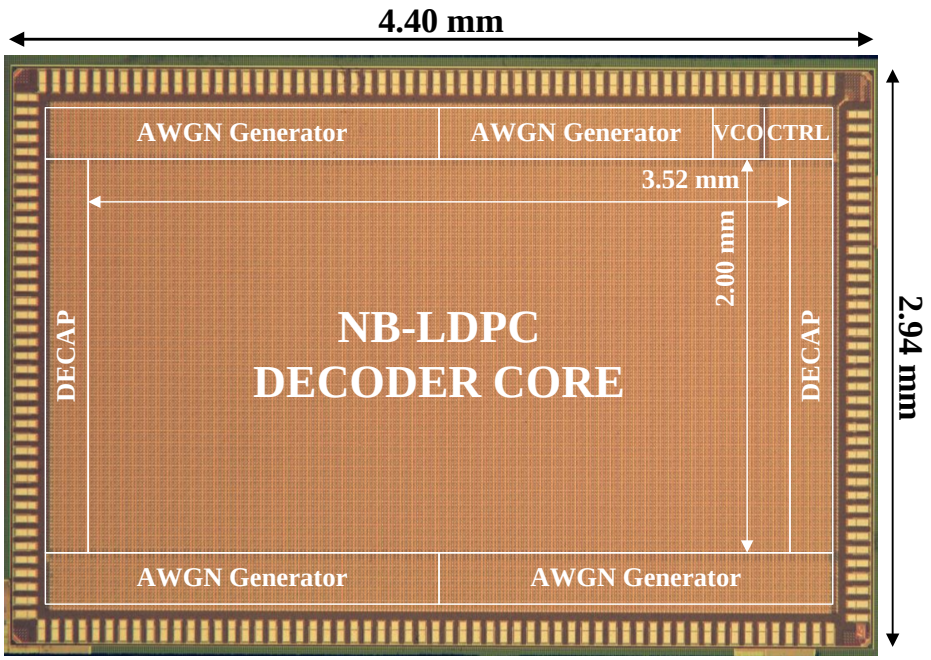


Figure 3.13: Chip microphotograph of the decoder test chip. Locations of the test peripherals and the decoder are labeled.

The chip supports two test modes: a scan mode which takes external inputs through scan chains and provides outputs through scan chains for functional verification, and an automated mode for the real-time testing of the decoder using on-chip generated AWGN noise to emulate the communication channel. Error statistics are collected for plotting BER and FER against SNR.

3.4.1 Chip Measurements

Fig. 3.14 shows the BER and FER curves for various configurations. The error rate reported is based on two months of extensive testing. With a decoding iteration limit of 100, the decoder achieves a BER of 7.53×10^{-11} at 4.2 dB, a significant improvement over binary LDPC codes of similar block length, e.g., the rate-1/2 672-bit binary LDPC code for the IEEE 802.11ad standard provides a BER of only 4.36×10^{-8} at 4.2 dB [75]. Structured binary LDPC codes of similar block length also encounter severe error floors, which is not seen in the NB-LDPC code. With a more practical 30 iterations and our proposed node-level convergence criteria of $M = 10$ and $T = 10$, the decoder still provides an excellent

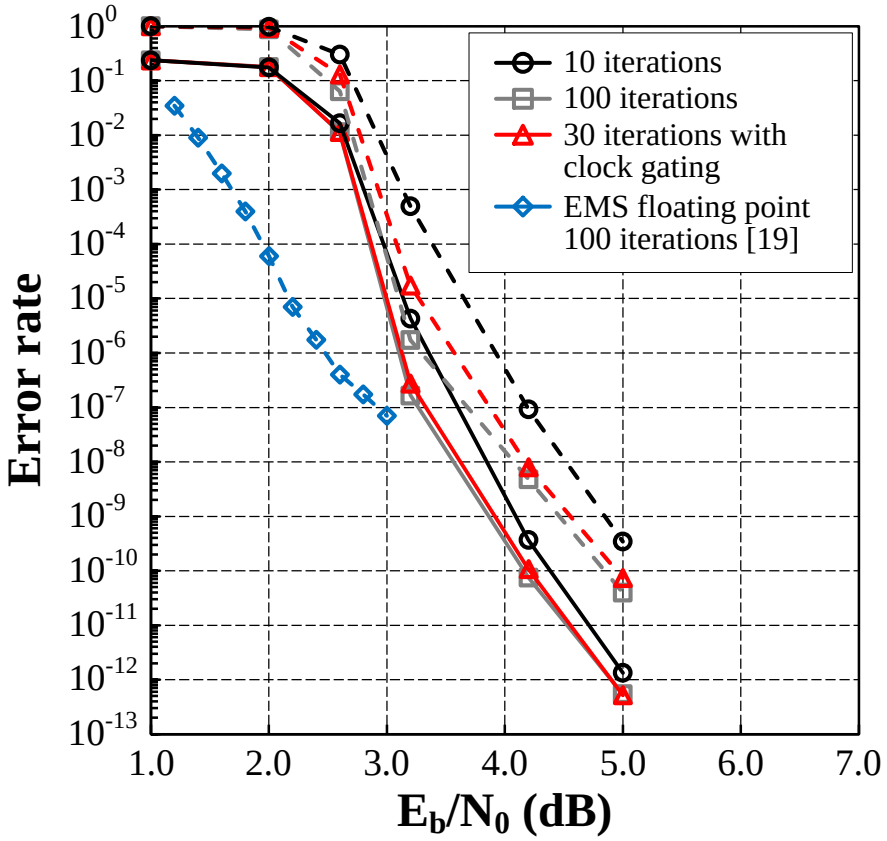


Figure 3.14: BER and FER performance of the GF(64) (160, 80) regular-(2, 4) NB-LDPC code using 5-bit quantization.

BER performance that is very close to the 100-iteration BER performance.

The NB-LDPC decoder test chip operates at a maximum clock frequency of 700 MHz at 1.0 V and room temperature for a coded throughput of 477 Mb/s with 30 decoding iterations. The test chip consumes 3.993 W, which translates to an energy efficiency of 8.38 nJ/b. Fig. 3.15, 3.16, and Table 3.1 summarize the measured power consumption of the NB-LDPC decoder test chip.

To improve the energy efficiency, the fine-grained dynamic clock gating is enabled with node-level convergence criteria of $M = 10$ and $T = 10$, reducing the power consumption by 50% and improving the energy efficiency to 4.14 nJ/b. To achieve a higher throughput, decoder termination is enabled to increase the throughput from 477 Mb/s to 1.22 Gb/s at 5.0 dB SNR (E_b/N_0). The power consumption increases due to a higher activity, but

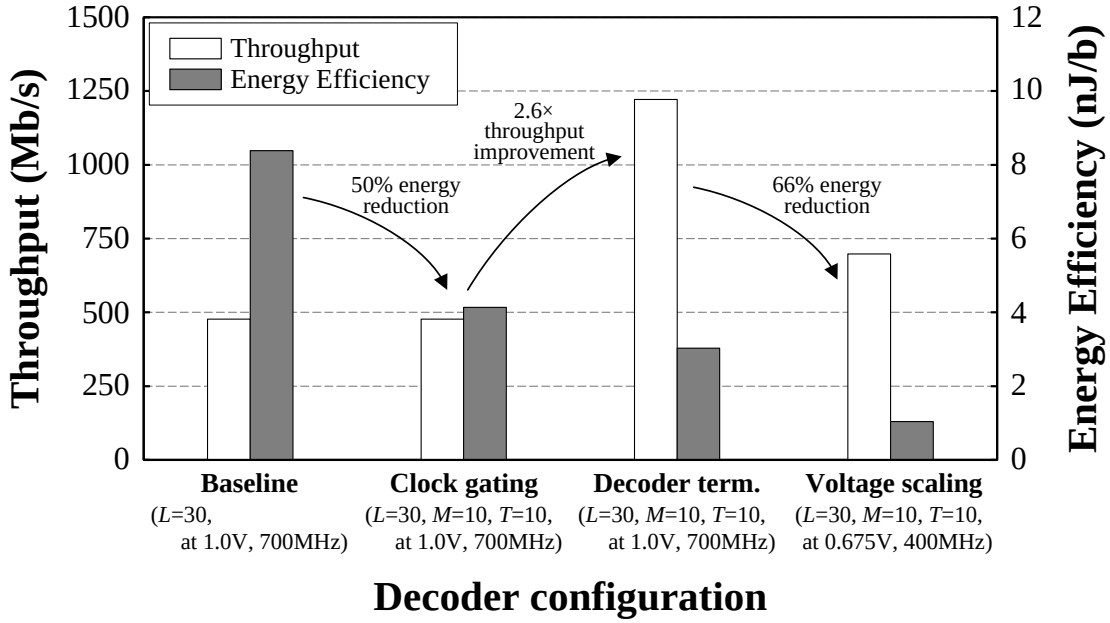
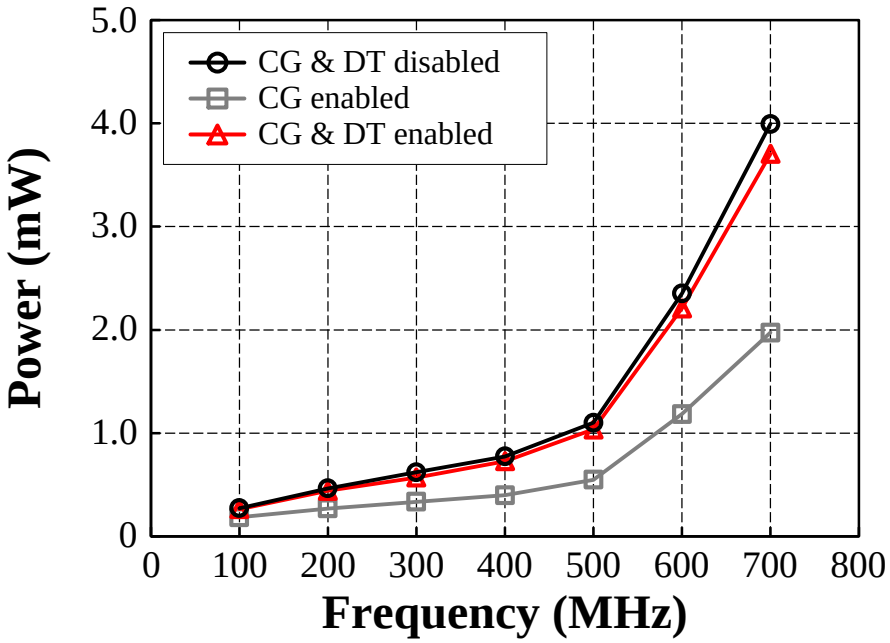


Figure 3.15: Illustration of throughput and energy efficiency of various decoder configurations at 5.0 dB SNR. L, M, and T represents decoding iteration limit, minimum decoding iteration, and consecutive iteration threshold, respectively.

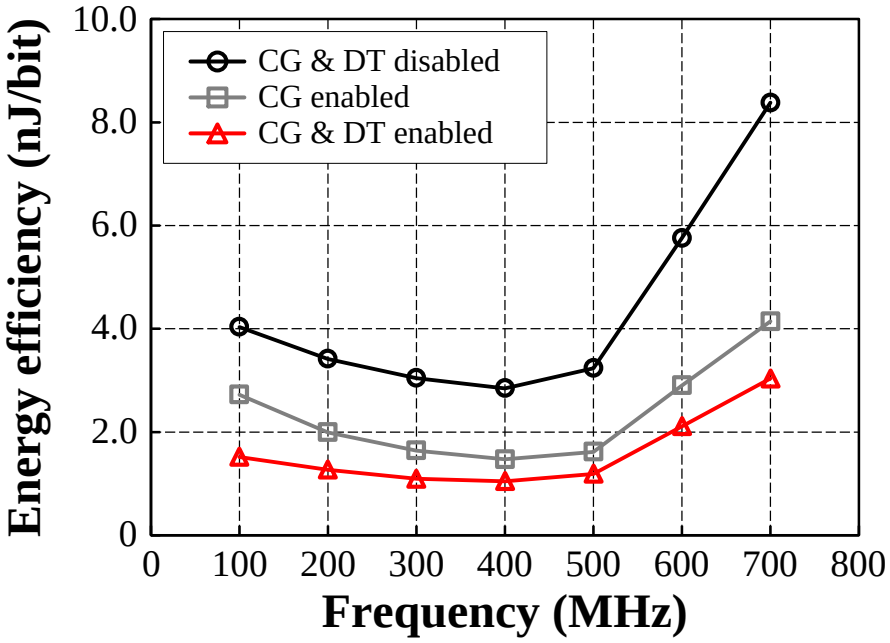
Table 3.1: Measurement summary of the NB-LDPC decoder at 5.0 dB SNR

Decoder Configuration	Throughput[1] (Mb/s)	Power[1] (W)	Energy Efficiency[1] (nJ/b)
700MHz @ 1.0V 30 Iterations	477	3.993	8.38
700MHz @ 1.0V 30 Iterations \w CG[2]	477	1.974	4.14
700MHz @ 1.0V 30 Iterations \w CG & DT[2]	1221	3.704	3.03
400MHz @ 0.675V 30 Iterations \w CG & DT[2]	698	0.729	1.04

[1] Measured at 5.0 dB SNR.
[2] CG denotes clock gating, and DT denotes decoder termination. The parameters used for clock gating and decoder termination are M = 10 and T = 10.



(a)



(b)

Figure 3.16: Measured NB-LDPC decoder (a) power and (b) energy efficiency at 5.0 dB SNR and 30 decoding iterations. CG denotes clock gating and DT denotes decoder termination. The parameters used for clock gating and decoder termination are $M = 10$ and $T = 10$.

the energy efficiency improves to 3.03 nJ/b, or 259 pJ/b/iteration. Voltage and frequency scaling can be applied to further reduce the power consumption and improve the energy efficiency. Scaling the supply voltage from 1.0 V to 675 mV reduces the maximum clock frequency from 700 MHz to 400 MHz and improves the energy efficiency to 1.04 nJ/b, or 89 pJ/b/iteration, at a reduced throughput of 698 Mb/s.

3.4.2 Comparison with State-of-the-Art

Table 3.2 and 3.3 summarize the results of the nonbinary LDPC decoder test chip along with other state-of-the-art synthesized designs published recently [76, 36, 37, 77, 40, 78, 42, 43]. It is important to note that all the previous designs have not been fabricated in silicon. This work is the first silicon that has been published to the best of our knowledge. The decoder claims higher throughput and energy efficiency (in pJ/b/iter), when normalized to 65 nm and 1.0 V, than the best previously reported post-layout results. The truncated EMS algorithm allows us to achieve excellent BER performance compared to other simplified algorithms.

Table 3.2: Comparison of state-of-the-art NB-LDPC decoders (ASIC layout)

	This Work		TVLSI'13 [43]	TVLSI'13 [42]	TSP'13 [40]	TCAS-I'12 [37]
Technology	65nm		90nm	28nm	90nm	90nm
Design	silicon		layout	layout	layout	layout
Code Length (symbols)	160		837	110	837	248
Code Rate	0.5		0.86	0.8	0.87	0.55
Galois Field	GF(64)		GF(32)	GF(256)	GF(32)	GF(32)
Decoding Algorithm	Truncated Extended Min-Sum		SES-GBFDA[a]	RTBCP[a]	Trellis-based Max-Log-QSPA[a]	Selective-input Min-Max
Core Area (mm^2)	7.04		6.6	1.289	46.18	10.33
Utilization (%)	87		-	75.7	-	-
Gate Count	2.78M (NAND)		0.468M (XOR)	2.57M (NAND)	8.51M (NAND)	1.92M (NAND)
Core Supply (V)	1.0	0.675	-	-	-	-
Clock Frequency (MHz)	700	400	277	520	250	260
Iterations	10-30[b]	10-30[b]	10	10	5	10
Throughput (Mb/s)	1221	698	716	546	234	47.7
Power (mW)	3704	729	-	976	893	480
Energy Efficiency (nJ/b)	3.03	1.04	-	1.78	3.82	10.06
Energy Efficiency (pJ/b/iter)	259	89	-	178	765	1006
Area Efficiency ($Mb/s/mm^2$)	173	99.1	108	424	5.06	4.62
Normalized to 65nm, 1.0V[c]						
Energy Efficiency (nJ/b)	3.03	2.29	-	4.15	2.76	7.27
Energy Efficiency (pJ/b/iter)	259	196	-	415	552	727
Area Efficiency ($Mb/s/mm^2$)	173	99.1	288	33.9	13.4	12.3

[a] SES-GBFDA stands for simplified enhanced serial generalized bit-flipping decoding algorithm, RTBCP stands for reduced memory complexity trellis-based check node processing, QSPA stands for q-ary sum-product algorithm.
[b] Iteration varies from 10 to 30 iterations based on decoder termination. The average iteration at 5.0 dB SNR is 11.71.
[c] General scaling theory is used to scale area, frequency (and throughput), and power by $1/s^2$, s, and $1/u^2$ respectively where s is the dimension scale factor and u is the voltage scale factor. The core supply of [37], [40], [42] are assumed to be 1.0V for normalization purpose.

Table 3.3: Comparison of state-of-the-art NB-LDPC decoders (ASIC synthesis)

	This Work		TVLSI'13 [78]	TCAS-I'12 [77]	TCAS-I'12 [36]	TVLSI'11 [76]
Technology	65nm		180nm	180nm	180nm	-
Design	silicon		synthesis	synthesis	synthesis	analysis
Code Length (symbols)	160		837	837	837	837
Code Rate	0.5		0.87	0.87	0.87	0.87
Galois Field	GF(64)		GF(32)	GF(32)	GF(32)	GF(32)
Decoding Algorithm	Truncated Extended Min-Sum		Relaxed Min-Max	Simplified Min-Sum	Min-Max	Reduced-complexity Min-Max
Gate Count	2.78M (NAND)		0.871M (NAND)	1.29M (NAND)	1.37M (NAND)	0.639M (XOR)
Clock Frequency (MHz)	700	400	200	200	200	150
Iterations	10-30[a]	10-30[a]	15	15	15	15
Throughput (Mb/s)	1221	698	64	64	16	10
Normalized to 65nm[b]						
Throughput (Mb/s)	1221	698	183	177	44	-

[a] Iteration varies from 10 to 30 iterations based on decoder termination. The average iteration at 5.0 dB SNR is 11.71.
[b] General scaling theory is used to scale frequency (and throughput) by s where s is the dimension scale factor.

3.5 Summary

We present a fully parallel NB-LDPC decoder to take advantage of the low wiring overhead that is intrinsic to NB-LDPC codes. To further enhance the throughput, we apply a one-step look-ahead to the elementary CN design to reduce the clock period, and interleave the CN and VN operations for a short iteration latency of 47 cycles. We implement a fine-grained clock gating at the node level to allow the majority of the processing nodes to be clock-gated long before reaching the iteration limit. A 7.04 mm^2 65 nm decoder test chip is designed for the GF(64) (160, 80) regular-(2, 4) NB-LDPC code. The decoder implements fine-grained dynamic clock gating and decoder termination to achieve a throughput of 1.22 Gb/s at 700 MHz, consuming 3.03 nJ/b, or 259 pJ/b/iteration. The test chip demonstrates a superior error correcting performance compared to binary LDPC decoders. Voltage and frequency scaling of the test chip to 675 mV and 400 MHz further improve the energy efficiency to 89 pJ/b/iteration at a reduced throughput of 698 Mb/s.

CHAPTER IV

Belief-Propagation Polar Decoder

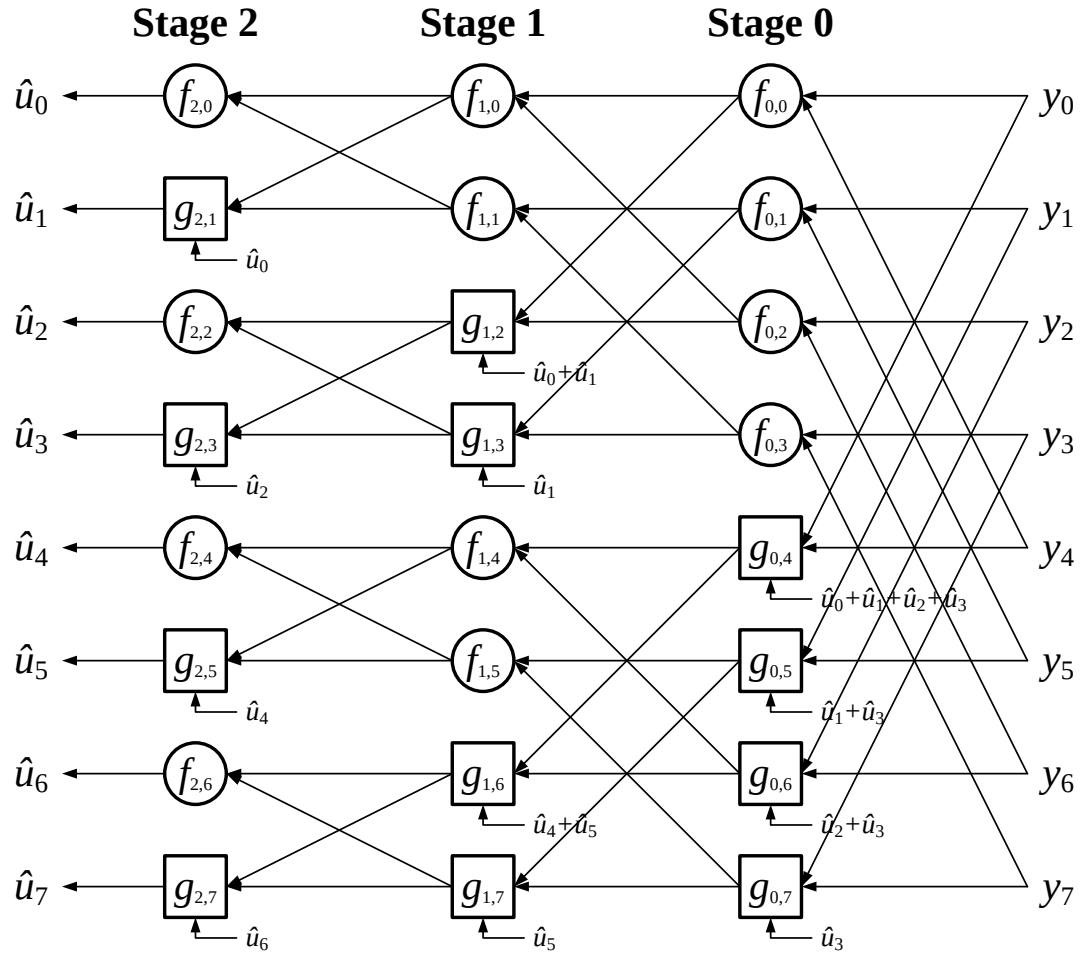
4.1 Decoding Algorithm

Two decoding algorithms exist for polar codes, namely the SC decoding [7] and the BP decoding algorithm [49]. Both algorithms work on the same generator matrix F_N but differ in the operation schedule. SC decoding is a non-iterative algorithm, serial in nature due to inter-bit dependence on the decoded outputs, whereas BP is an iterative algorithm similar to the BP decoding of LDPC codes. BP decoding is more parallelizable than SC due to the lack of inter-bit dependence.

4.1.1 Successive Cancellation Decoding

The SC decoding algorithm decodes each bit $\hat{u}_i$ sequentially based on the channel output y_i and the previously decoded bits $\{\hat{u}_0, \hat{u}_1, ..., \hat{u}_{i-1}\}$. An example 8-bit SC decoder factor graph and its decoding schedule is illustrated in Fig. 4.1 [47]. Decoding starts by a series of f functions that leads to the decoding of $\hat{u}_0$. In order to decode $\hat{u}_1$, the decoded $\hat{u}_0$ needs to be fed to the g function, $g_{2,1}$. Similarly, decoding $\hat{u}_2$ requires the decoded $\hat{u}_0$ and $\hat{u}_1$, and so forth. Therefore, the decoder requires at least $2N - 2 = 14$ cycles to decode the full 8 bits. The iteration latency is linearly dependent on the block length N and therefore increasing throughput becomes a challenge for SC decoders. The f and g functions used in the SC decoder are defined as [47]:

$$f(a,b) = \frac{1+ab}{a+b},\ g(s,a,b) = a^{1-2s}b \tag{4.1}$$



(a) Factor graph

Cycle	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Process Nodes	$f_{0,0}$	$f_{1,0}$	$f_{2,0}$	$g_{2,1}$	$g_{1,2}$	$f_{2,2}$	$g_{2,3}$	$g_{0,4}$	$f_{1,4}$	$f_{2,4}$	$g_{2,5}$	$g_{1,6}$	$f_{2,6}$	$g_{2,7}$
	$f_{0,1}$	$f_{1,1}$			$g_{1,3}$			$g_{0,5}$	$f_{1,5}$			$g_{1,7}$		
	$f_{0,2}$							$g_{0,6}$						
	$f_{0,3}$							$g_{0,7}$						
Output			$\hat{u}_0$	$\hat{u}_1$		$\hat{u}_2$	$\hat{u}_3$			$\hat{u}_4$	$\hat{u}_5$		$\hat{u}_6$	$\hat{u}_7$

(b) Decoding schedule

Figure 4.1: Example of successive cancellation: (a) factor graph for a $N = 8$ polar code and (b) successive cancellation decoding schedule.

which can be approximated in the LLR domain as [47]:

$$f(a,b) = sign(a) \cdot sign(b) \cdot min(|a|,|b|),\ g(s,a,b) = a(-1)^s + b \tag{4.2}$$

for a more efficient hardware implementation.

4.1.2 Belief Propagation Decoding

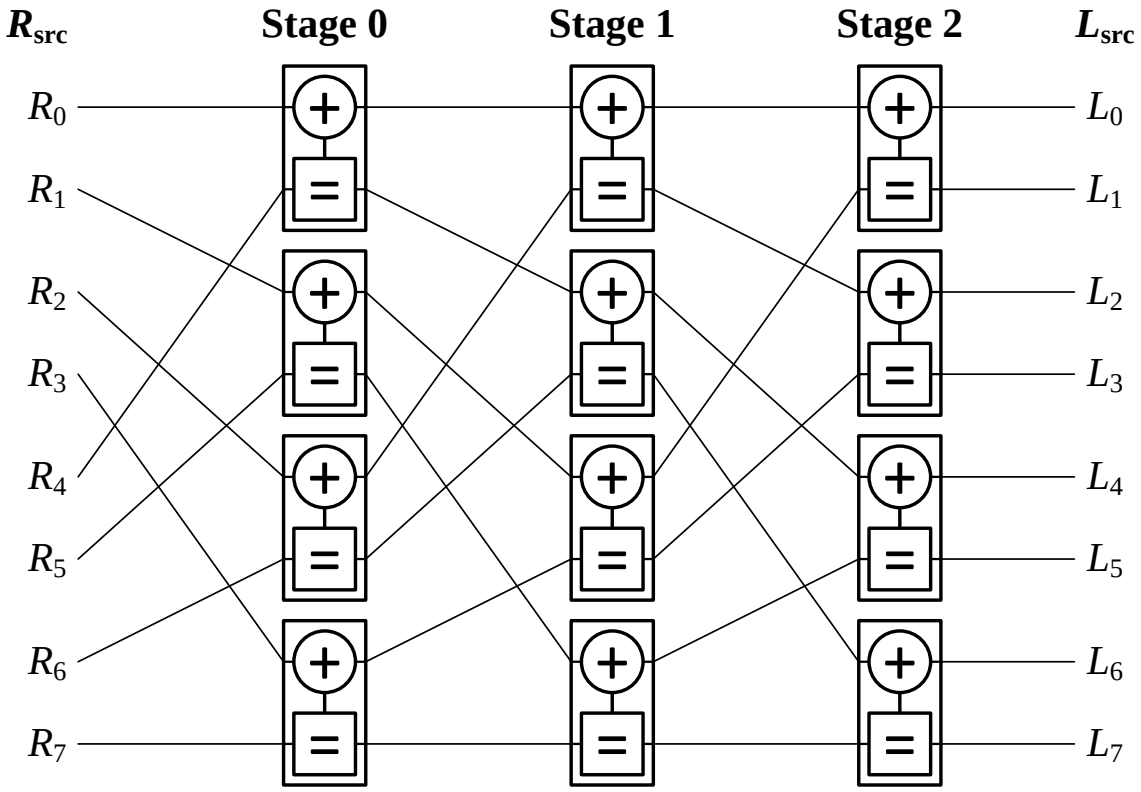
Using the BP algorithm, the polar code is decoded by iteratively passing the right-bound messages (R messages, or messages from the source bits that represent the frozen bit information) from left to right, and the left-bound messages (L messages, or the messages from the channel outputs) from right to left over the factor graph. An example factor graph is illustrated in Fig. 4.2(a). Note that the edges connecting the nodes in factor graph are regular and fixed between nodes. The connections are different from the polar encoder introduced in Fig. 1.7 and the SC decoder shown in Fig. 4.1(a), as the nodes have been intentionally shuffled such that the connections between stages are identical [49]. This shuffling simplifies the implementation of the decoder, which will become apparent later. The L message on the right-most side (L_{src}) represents the channel output y and the R message on the left-most side (R_{src}) represents whether a bit is used as a frozen bit (set to ∞) or as an information bit (set to 0).

The processing element (PE) shown in Fig. 4.2(b) performs 3 compare-selects and 3 sums to compute a pair of L messages and a pair of R messages using the following equations:

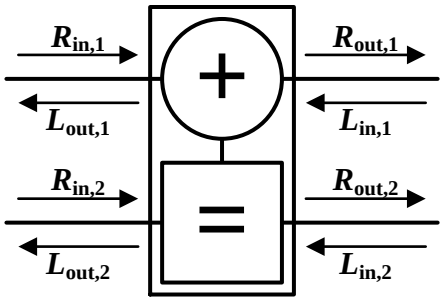
$$\begin{aligned} L_{out,1} &= f(L_{in,1}, L_{in,2} + R_{in,2}),\ L_{out,2} = f(R_{in,1}, L_{in1,}) + L_{in,2} \\ R_{out,1} &= f(R_{in,1}, L_{in,2} + R_{in,2}),\ R_{out,2} = f(R_{in,1}, L_{in1,}) + R_{in,2} \end{aligned} \tag{4.3}$$

where f is defined in (4.2).

The decoding schedule of the BP algorithm is illustrated in Fig. 4.2(c). The messages are passed iteratively back and forth through the stages of PEs in factor graph until the iteration limit is reached. Within an iteration, the R_{src} messages first propagate through the PEs in stage 0 and 1. Then the L_{src} messages propagate through the PEs in stage 2, 1,



(a) Factor graph



(b) Processing Element (PE)

Cycle	0	1	2	3	4	5	6	7	8	9	10	11	...
Iteration	Iteration 0					Iteration 1					Iteration 2		
	R prop.		L prop.			R prop.		L prop.			R prop.		...
Stages	*S0*	*S1*	*S2*	*S1*	*S0*	*S0*	*S1*	*S2*	*S1*	*S0*	*S0*	*S1*	...
Output					$\hat{u}_{0..7}$					$\hat{u}_{0..7}$			

(c) Decoding schedule

Figure 4.2: Example of BP factor graph for a $N = 8$ polar code.

and 0. The resulting intermediate L and R messages (L_{int} and R_{int}) are stored in memory for use in the next iteration. Once the iteration limit is reached, the posterior is computed by adding the R_{src} message to the leftmost L message, which is equivalent to replacing the frozen bit locations to the known values (set to 0). The decoding latency of the BP algorithm is $2\log_2 N - 1$ if a whole stage (column) of PEs work in parallel. The decoding latency can be reduced if multiple stages of PEs are available for parallel processing.

Most polar decoders presented in the past few years have used the SC algorithm which is the original algorithm proposed by Arikan in [7] to prove the capacity of polar codes. However, the algorithm has a major drawback which prevents a high-throughput implementation due to the inter-bit dependency that results in a serial decoding. Therefore we decide to use the BP algorithm which enables high degree of parallelism than SC. We will show that the error-correcting performance of BP decoders are comparable to SC decoders.

4.2 Decoder Architecture

The most basic architecture of the BP polar decoder is a direct implementation of the polar code factor graph illustrated in Fig. 4.2(a). The decoder can be implemented using a set of PEs and two memories to store the L_{int} and R_{int} messages produced by the PEs as done in [49]. The L_{src} and R_{src} messages are stored in separate memories. The decoding is performed by iteratively passing LLRs from left to right (right propagation) and then from right to left (left propagation) through the PEs to compute new R_{int} and L_{int} messages. A PE produces a pair of L and a pair of R messages based on (4.3). N_{PE} is the number of PEs used in the architecture. For example, if a full stage of PEs are used, $N_{PE} = N/2$. The decoding latency depends on N_{PE} and is calculated using the following equation:

$$I_{cycle} = N_{stage} \cdot \frac{N}{N_{PE}}, \text{ where } N_{stage} = \log_2 N \tag{4.4}$$

The regular wiring structure between stages permits a highly parallel decoder implementation without the complex wiring seen in LDPC decoders. The block diagram of a conventional 1024-bit BP decoder is shown in Fig. 4.3. The single-column bidirectional decoder architecture consists of 512 bidirectional PEs to compute one stage in parallel, with

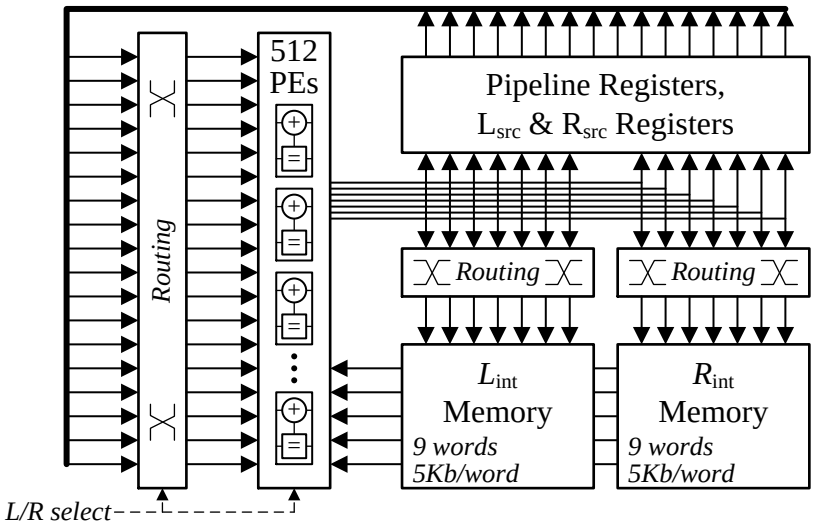


Figure 4.3: Conventional single-column bidirectional architecture of a 1024-bit BP polar decoder.

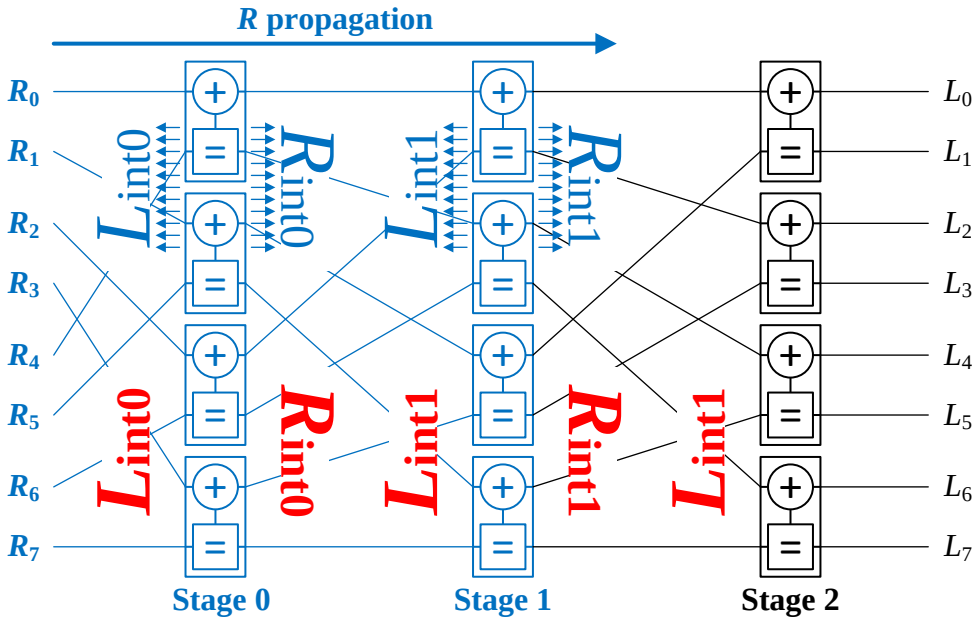
2 45-Kb message memories to store the left-bound and right-bound intermediate messages. A bidirectional PE computes both pairs of L and R messages (i.e., computes all 4 equations of (4.3)). Each memory stores 9 rows of 512 pairs of 5-bit LLR messages.

4.3 High-Throughput Double-Column Unidirectional Architecture

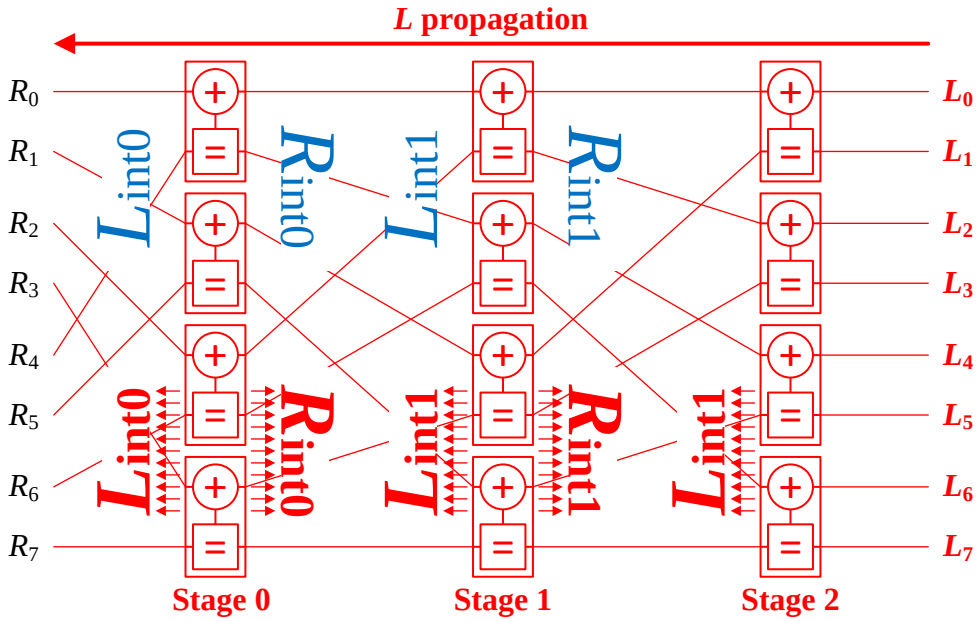
We improve on the basic architecture design by reducing the required memory by half through the use of unidirectional PE, which only computes either a pair of L or a pair of R message depending on the direction of propagation. The memory is reduced by half to 45 Kb and the PE logic reduced by 33%. The throughput is nearly doubled by the efficient use of 2 columns of PEs over one column of PEs.

4.3.1 Unidirectional Processing Architecture

The BP decoding of the polar code specifies one right-bound message propagation (R propagation) and one left-bound message propagation (L propagation) to complete one iteration. Fig. 4.4 illustrates the R and L propagation and the outputs created by the PEs in each stage for an 8-bit polar decoder using the conventional bidirectional architecture.

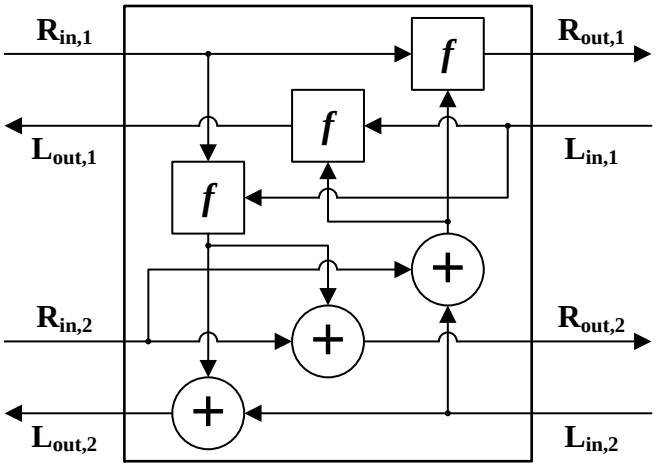


(a) R propagation

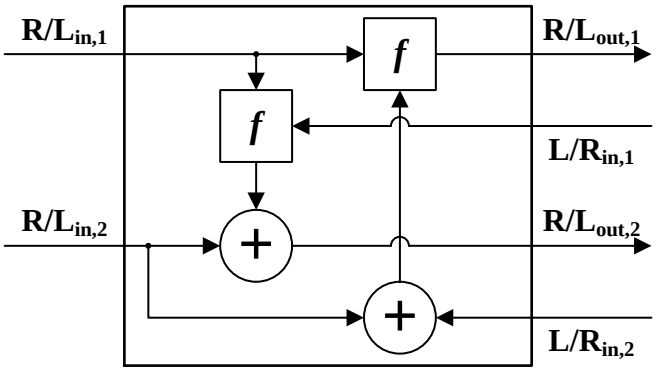


(b) L propagation

Figure 4.4: Illustration of the PE outputs in a bidirectional architecture. The outputs produced by the PEs in the R and L propagations are shown in blue and red, respectively.



(a) Bidirectional PE



(b) Unidirectional PE

Figure 4.5: Illustration of the (a) bidirectional PE which outputs both L_{out} and R_{out} and (b) unidirectional PE which outputs either L_{out} or R_{out} based on direction.

In each stage, one row of messages are read from each R_{int} and L_{int} memories and used as inputs to the PEs. The bidirectional PEs then process these inputs to create new messages to store in the R_{int} and L_{int} memories. For simplicity, we assume that L_{int0} is stored in the L_{int} memory instead of a separate memory as described in the previous section.

As can be seen in Fig. 4.4(a), the R messages created in the R propagation (R_{int0}, R_{int1} in blue) overwrites the R messages created in the previous L propagation (R_{int0}, R_{int1} in red). The R messages that have been produced in the previous L propagation are overwritten without being used. Therefore, the production of R messages in an L propagation is unnecessary. Similarly, the production of L messages in an R propagation is unnecessary, as they are overwritten in the next L propagation without being used, as

illustrated in Fig. 4.4(b)

The insight from the above discussion is that the message propagation is unidirectional to allow only one of L or R messages to be propagated. Therefore, a bidirectional PE is unnecessary and a unidirectional PE can be designed to match the unidirectional propagation to only compute L messages in L propagation, and only R messages in R propagation. This simplification reduces the complexity of the PE to 2 compare-selects and 2 sums as illustrated in Fig. 4.5.

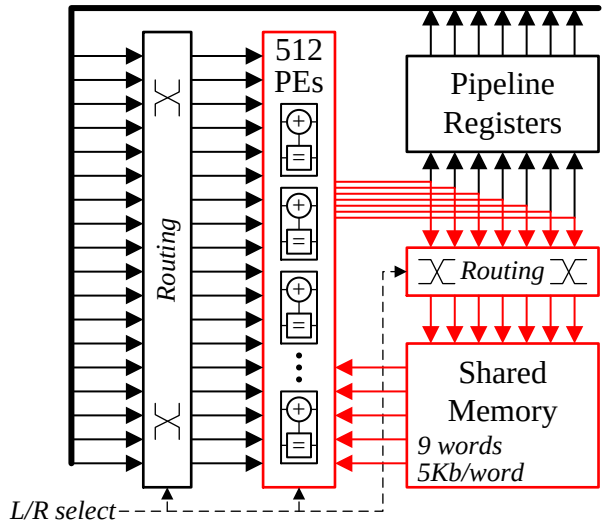
The unidirectional processing allows L messages and R messages to share only one 45 Kb memory for a 1024-bit polar code as illustrated in Fig. 4.6(a). The memory size is reduced by 50% and logic complexity by 33% without sacrificing throughput compared to the conventional bidirectional architecture shown in Fig. 4.3. Synthesis results show that the area is reduced by 35%, and the critical path is shortened to 3.5 ns.

4.3.2 Double-Column Architecture

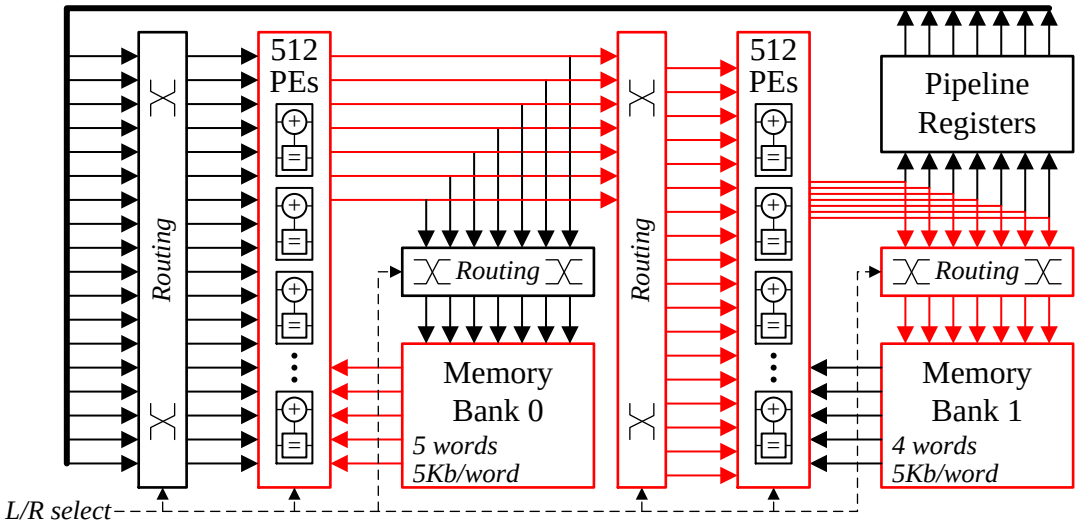
From the synthesis result, the critical path of the unidirectional single-column decoder architecture is highlighted in red in Fig. 4.6(a). It runs through the shared memory, PE, router and returns to the shared memory. Since a single column of 512 PEs are available to process a stage of the factor graph in one clock cycle, the R propagation takes 9 cycles and the L propagation takes 10 cycles adding up to 19 cycles per iteration. With the critical path of 3.5 ns, a decoding iteration lasts 66.5 ns.

Within the critical path, the processing and routing delays are relatively short, benefiting from the compact unidirectional PE design and the regular wiring in polar codes. Therefore the memory access time of the shared memory dominates the critical path.

For a better utilization of a clock period, we design a double-column architecture shown in Fig. 4.6(b). In this design, two columns of 512 PEs are used to process two stages of the factor graph in one clock cycle. The critical path, highlighted in red, increases from 3.5 ns to 4 ns. This is a rather small increase for the benefit of shortening the iteration latency to 10 cycles or 40 ns and improving the overall throughput by 66%. As summarized in Fig. 4.6(c), the number of PEs is doubled from 512 to 1024 as well as the number of PE input routers. Despite this increase, the memory size remains constant. The memory is split



(a) single-column architecture

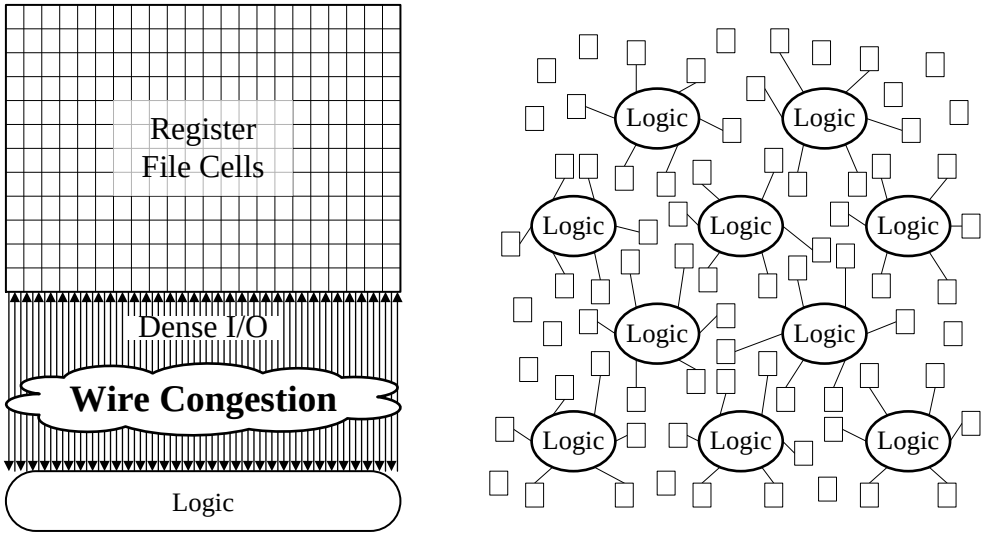


(b) double-column architecture

	Single-Column	Double-Column
# of PEs	512	1024
Memory	45Kb (1 bank)	45Kb (2 banks)
Critical Path	3.5 ns	4.0 ns
Latency – R prop.	9 cycles	5 cycles
Latency – L prop.	10 cycles	5 cycles
Iteration Latency	19 cycles (66.5 ns)	10 cycles (40 ns)

(c) architecture comparison

Figure 4.6: (a) single-column and (b) double-column unidirectional architecture and (c) their comparison. The critical paths are highlighted in red.



(a) Register file macro

(b) Distributed registers

Figure 4.7: Conventional memories: (a) standard register file and (b) distributed registers.

into 2 banks (a 25-Kb bank and a 20-Kb bank) to support the double-column processing. From synthesis, the overall decoder cell area only increases by 28% thanks to simpler PE design.

4.4 High-Density Bit-Splitting Register File

Using a 5-bit message quantization, the required memory read and write access bandwidth to support the 1024-parallel decoder is 20 Kb per cycle. Due to the small number of words for each memory – 5 words for bank 0 and 4 words for bank 1 – DRAMs and SRAMs are not good candidates as they suffer from very low cell efficiency for very shallow memory, where the peripherals such as the sense amplifier dominate the area. Furthermore, the very wide access would prevent using of column multiplexing as all bitlines would require sense amplifiers.

More viable memory implementation options include a register file macro or distributed registers as illustrated in Fig. 4.7. For the case of a register file macro, the wide access of 20 Kb would be available on one side of the macro as shown in Fig. 4.7(a). However, the very wide memory aspect ratio becomes problematic, and the ultra-dense port placement

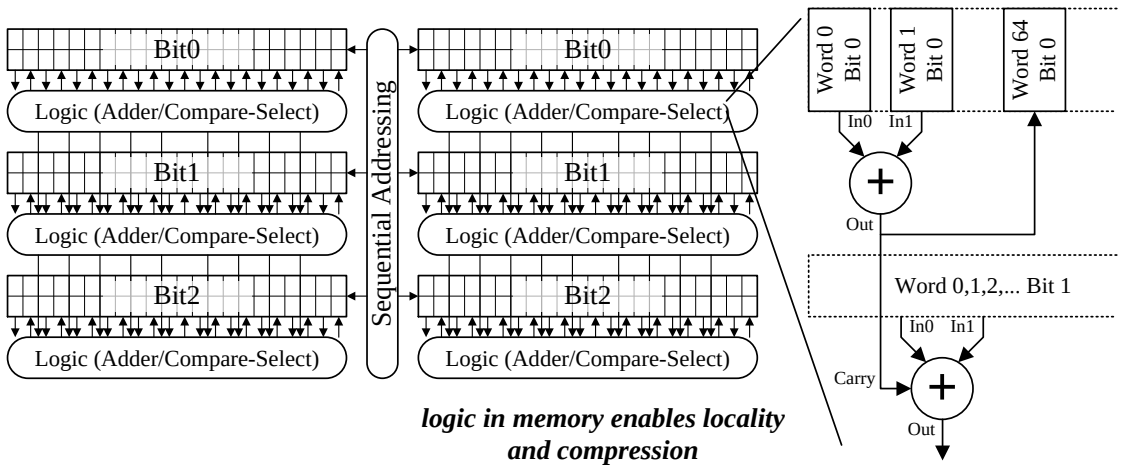


Figure 4.8: Illustration of the proposed bit-splitting register file.

consumes all available routing tracks required to route the inputs and outputs of the memory to the PE logic. Using distributed registers for the memory would solve the wire congestion problem and wide memory aspect ratio, but a design based on distributed registers suffers from high clock tree power as registers have to be spread out in the design and a power-hungry clock tree is necessary to deliver all the clock inputs of the registers.

A better solution would be a middle ground between the register file and distributed register topology, a trade-off between power and area. To relieve the wire congestion, we split the register file to bit rows to provide more tracks, and allocate PEs between bit rows to take advantage of locality and compression from the adders and compare-select logic in the PE. The compare-select and addition are done at bit level, right next to the bit memory, and the number of output wires over to the next bit row is substantially reduced.

As illustrated in Fig. 4.8, the memory is split into bit-groups and the corresponding PE logic are placed next to it to exploit locality and compression. For example, bit 0 of word 0 and 1 go through an adder and bit 0 of the output is stored to the location for word 64. The only propagating output to the next bit row is the carry. Therefore, instead of routing 4 wires (2 outputs and 2 inputs) from bit 0 row to bit 1 row, only 1 is routed and the rest are consumed locally.

To reduce the memory footprint, a simple sequential addressing scheme is used as the

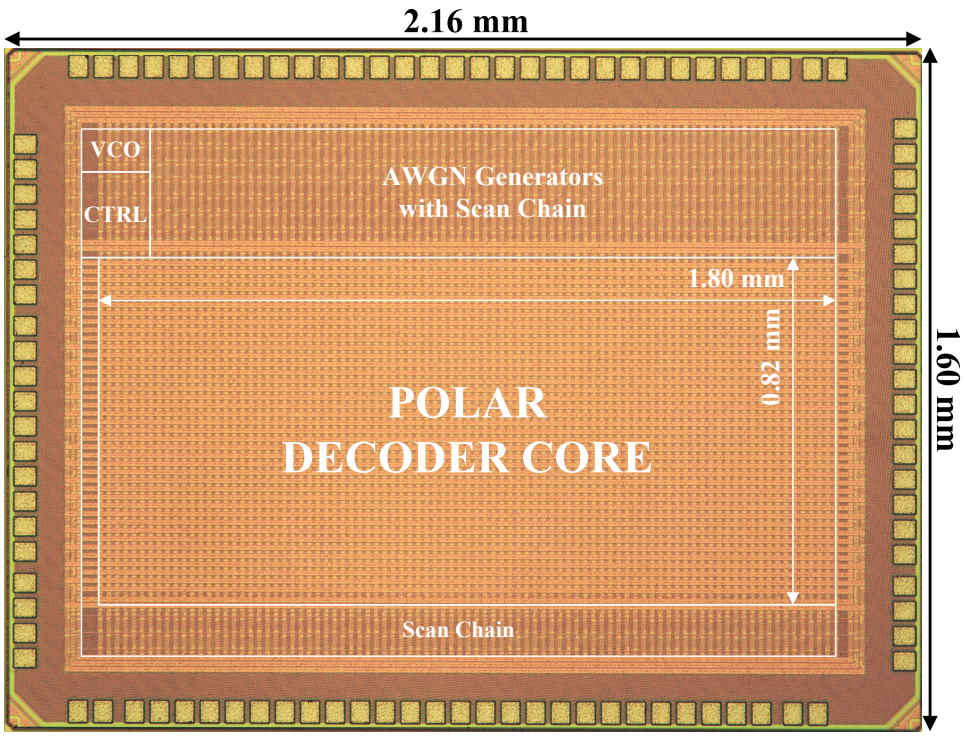


Figure 4.9: Chip microphotograph of the decoder test chip. Locations of the test peripherals and the decoder are labeled.

memory access pattern is fixed. In addition, registers are replaced by latches and the read and write word lines are shared to reduce the area and power. The resulting latch-based bit-splitting register file occupies 1.7 mm × 0.12 mm and 1.7 mm × 0.1 mm for bank 0 and bank 1, respectively, each providing 5 Kb read ports and 5 Kb write ports along one 1.7 mm side. It supports a denser integration of memory and logic than a standard register file, resulting in a final decoder area of 1.476 mm^2 with a high density of 85%.

4.5 Decoder Chip Implementation and Measurement Results

A double-column BP decoder test chip for the 1024-bit polar code incorporating bit-splitting register file was fabricated in a TSMC 65 nm CMOS process [79]. The chip microphotograph is shown in Fig. 4.9. The test chip measures 2.16 mm × 1.6 mm, and the core measures 1.8 mm × 0.82 mm, or 1.476 mm^2. The test chip incorporates AWGN generators to model the communication channel and provide input vectors in real time. An on-chip controller keeps track of the decoding errors for measuring the error correcting

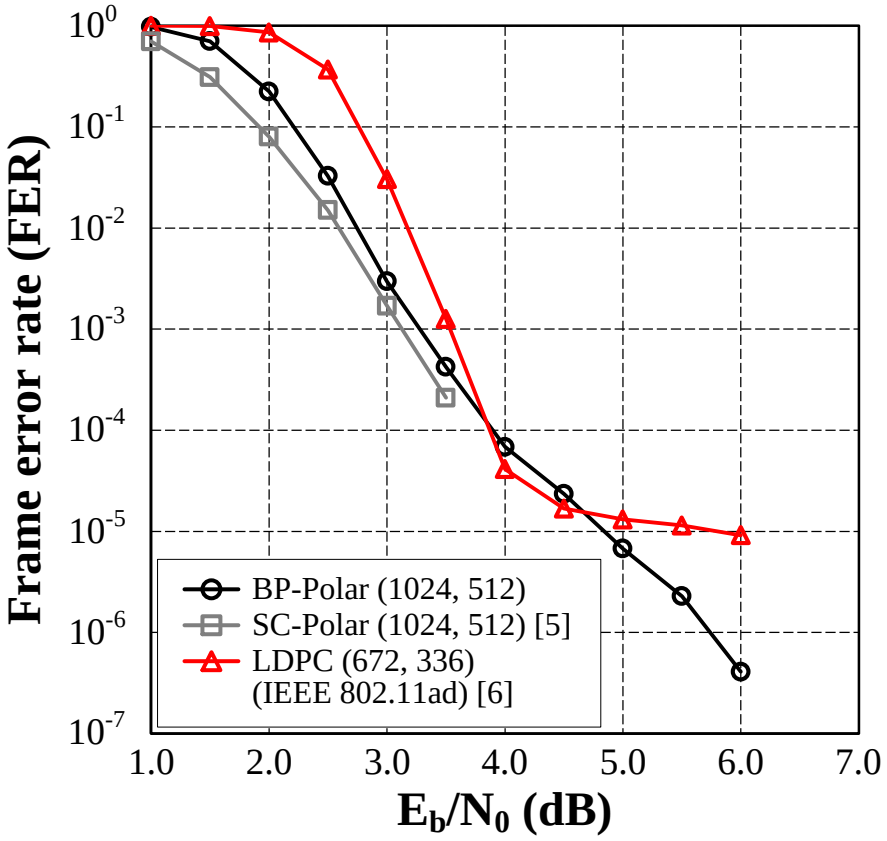


Figure 4.10: FER performance of the (1024, 512) polar code using SC and BP decoding algorithm, and the (672, 336) LDPC code for the IEEE 802.11ad standard for comparison.

performance. The code rate of the test chip is fully configurable by specifying the desired number of frozen bits through the scan chain.

The chip supports two test modes: a scan mode which takes external inputs through scan chains and provides outputs through scan chains for functional verification, and an automated mode for the real-time testing of the decoder using on-chip generated AWGN noise to emulate the communication channel. Error statistics are collected for plotting error rate against SNR.

4.5.1 Chip Measurements

Fig. 4.10 shows the FER curve of the test chip compared to a SC polar decoder of the same polar code [47] and an LDPC code for the IEEE 802.11ad standard [75]. The FER performance of the BP polar decoder is comparable to SC polar decoder as well as the LDPC code. Furthermore, no error floor is observed even at low error rates.

At room temperature and a nominal 1.0 V supply voltage, the BP polar decoder test chip operates at a maximum frequency of 300 MHz for a throughput of 2.05 Gb/s using 15 iterations. With a simple early termination scheme based on agreement of 3 consecutive hard decisions after a minimum of 3 iterations, the average iteration count is lowered to 6.57 (including convergence detection latency) at a 4.0 dB SNR with no loss in error correcting performance. Early termination enables a higher throughput of 4.68 Gb/s at 478 mW, or 15.5 pJ/b/iteration.

4.5.2 Comparison with State-of-the-Art

Table 4.1 compares the result of the BP polar decoder test chip against a state-of-the-art SC polar decoder published recently [46]. Note that our design is the first BP polar decoder ASIC implementation in silicon to the best of our knowledge. The chip demonstrates a 34, 2.8, and 5.2 times improvement in throughput, energy efficiency, and area efficiency, respectively, over the latest SC polar decoder ASIC [46] (normalized to 65 nm and 1.0 V). Scaling the supply voltage to 475 mV reduces the throughput to 780 Mb/s for an improved energy efficiency of 3.6 pJ/b/iteration.

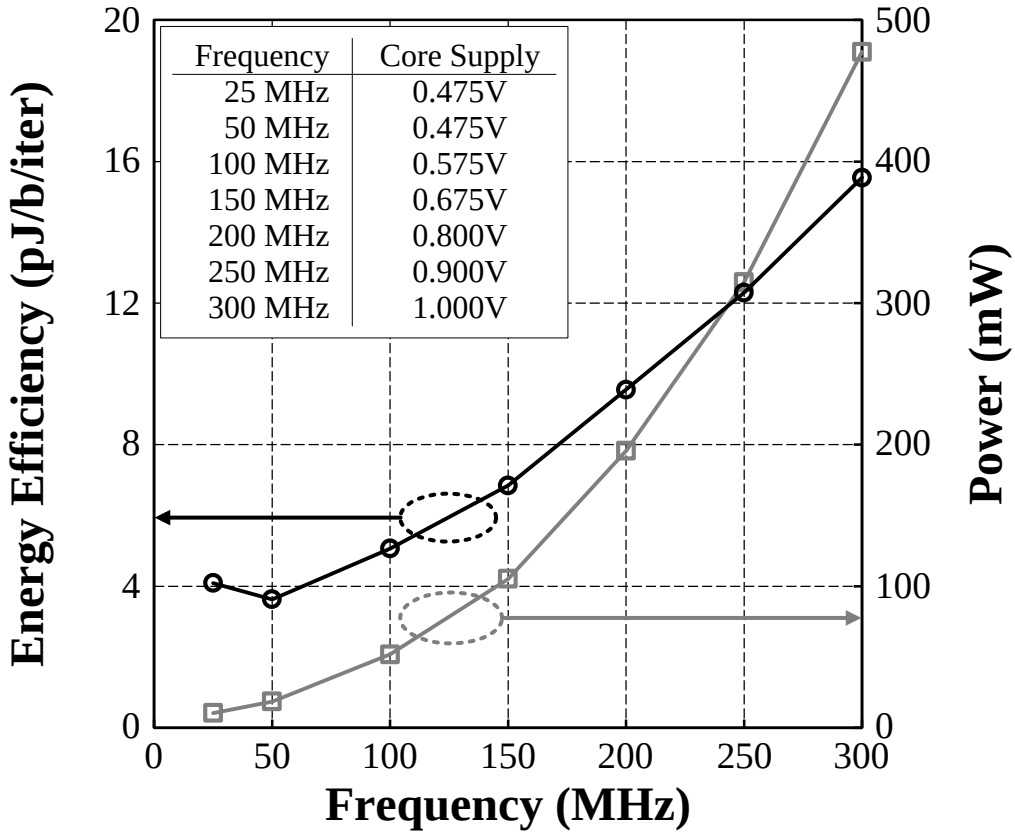


Figure 4.11: Measured power consumption and energy efficiency of the BP polar decoder at the minimum supply voltage for each clock frequency. (BP polar decoding using maximum 15 iterations with early termination enabled.)

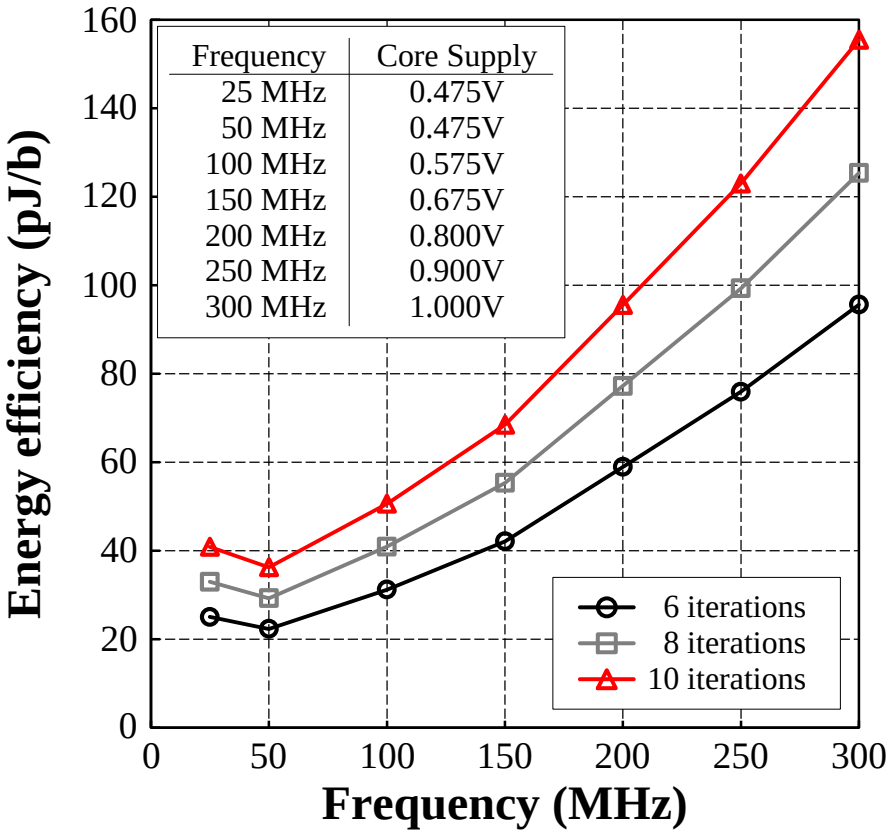


Figure 4.12: Measured energy efficiency of the BP polar decoder at the minimum supply voltage for each clock frequency at various decoding iteration limit.

Table 4.1: Comparison of state-of-the-art polar decoders.

	This Work		ASSCC'12 [46]
Code	BP Polar		SC Polar
Block Length	1024		1024
Process [nm]	65		180
Core Area [mm^2]	1.476		1.71
Utilization	85%		–
Supply [V]	1.0	0.475	1.3
Frequency [MHz]	300	50	150
Power [mW]	477.5	18.6	67
Iteration	6.57[a]	6.57[a]	–
Throughput [Mb/s]	4676	779.3	49
Energy Eff. [pJ/b]	102.1	23.8	1367
Energy Eff. [pJ/b/iter]	15.54	3.63	–
Area Eff. [$Mb/s/mm^2$]	3168	528.0	28.65
Normalized to 65nm, 1.0V			
Throughput [Mb/s]	4676	779.3	135.7
Energy Eff. [pJ/b]	102.1	23.8	292.2
Area Eff. [$Mb/s/mm^2$]	3168	528.0	608.5

[a] Average decoding iteration at 4.0dB with early termination enabled.

4.6 Summary

We present a 1.48 mm^2 1024-bit BP polar decoder designed in 65 nm CMOS. Using the conventional bidirectional single-column architecture as a baseline, we design a unidirectional processing architecture to reduce the memory size to 45 kb, and simplify the processing element logic by 33%. To enhance the throughput and decoding latency, we apply loop-unfolding to implement a double-column 1024-parallel architecture. This architecture improves the decoding throughput by 66% to 2.05 Gb/s at 300 MHz using 15 decoding iterations. A simple early termination technique is used to detect convergence and terminate, enabling a high throughput of 4.58 Gb/s. The memory used in the decoder accommodates logic in memory for an 85% cell placement density. The architecture and circuit techniques reduce the power consumption to 478 mW for an efficiency of 15.5 pJ/b/iteration at 1.0 V. Using voltage and frequency scaling, the energy efficiency is improved to 3.6 pJ/b/iteration at 475 mV for a throughput of 780 Mb/s at 50 MHz.

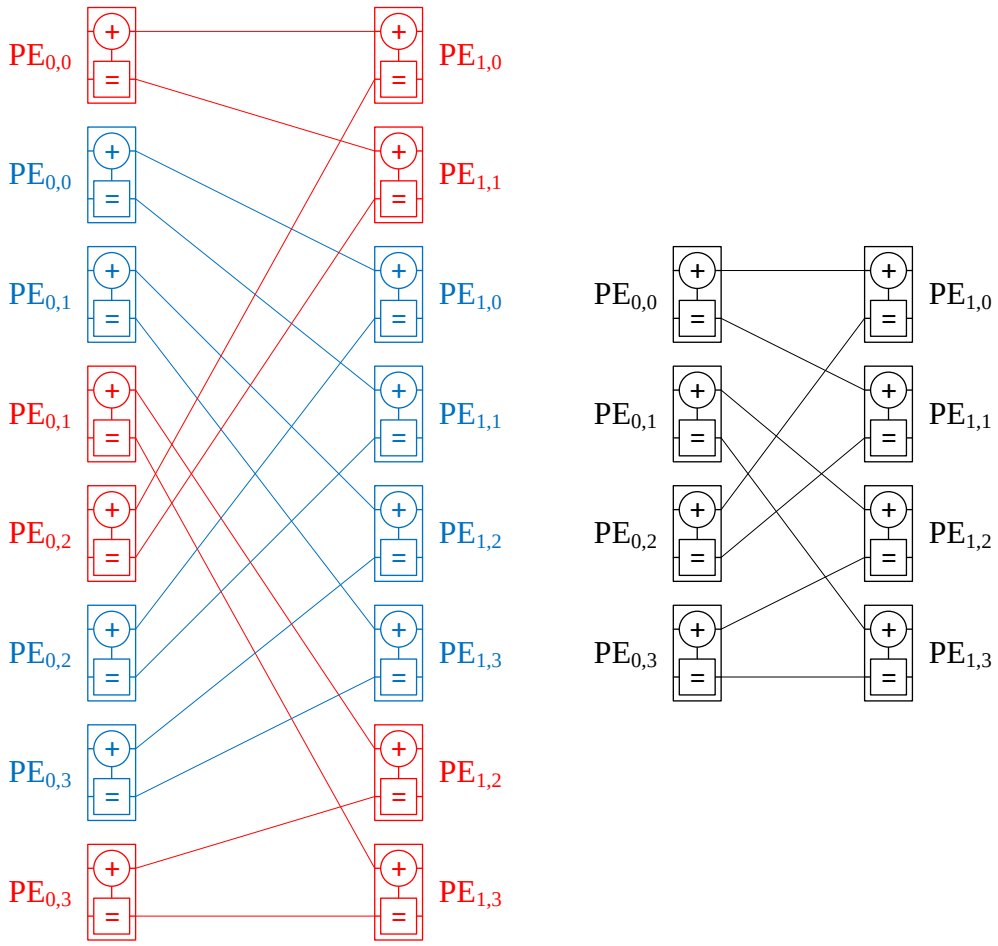
4.7 Future Research Directions

4.7.1 Polar Code Design

The structure of polar codes is defined by the generator matrix as discussed in Chapter I. Therefore the code design is essentially the selection of bits in the codeword to be frozen. Arikan proposed in [7] an explicit and efficient construction of polar code for the binary-erasure channel (BEC). However, this method cannot be used for other channel models, such as AWGN. Past work has proposed algorithms [80, 81] to find the frozen bit pattern for other channels. However, the analytical bit selection algorithms are based on SC decoding, not BP decoding, so these bit selections do not work well in BP decoding. On the other hand, BP decoding is not as tractable as SC decoding, so an analytical bit selection may have to rely on assumptions that render the bit selection algorithm not very applicable in practice. A simulation-based bit selection can be proposed for BP polar decoder to be used in AWGN and other practical channels. Monte-Carlo simulation can be used to measure the error rate of each bit position in BP decoding, and the bit selection can be made based on the ranking of the error rate of each bit. The simulation-based method is expected to find a more optimal bit selection for BP decoding to close the gap toward the best performance achieved by SC decoding.

4.7.2 Reconfigurable BP Polar Decoder

The regular wiring structure between stages in BP polar decoders allows polar codes of different block lengths to be decoded on the same hardware. For example, an 8-bit polar code can be decoded using a 16-bit BP polar decoder hardware. As illustrated in Fig. 4.13, the 16-bit polar decoder can be used to implement two 8-bit codes (shown in red and blue) without any change in the routing structure. In fact, polar codes of different block lengths are all compatible with each other, and they can be decoded on the same hardware as long as there is enough memory to store the intermediate messages. This reconfigurable architecture would be beneficial in applications where different block lengths need to be supported as well as different code rates, e.g., in multi-standard radios.



(a) 16-bit BP-polar decoder

(b) 8-bit BP-polar decoder

Figure 4.13: Illustration of 16-bit BP polar decoder factor graph containing two 8-bit BP polar decoder factor graphs.

CHAPTER V

Conclusion

Communication and storage applications are continuously evolving with growing requirements. In order to reduce the total energy cost of these systems, the use of channel codes have become absolutely necessary. The coding gain from channel codes allows for reduction in transmit power at the cost of decode power. It is therefore important to design a decoder with both good coding gain and high energy efficiency. We have looked into the design of LDPC, nonbinary LDPC, and polar codes.

LDPC code is the most popular choice in modern communication technology due to its good error-correcting performance and mature designs. However, with more parallel designs – due to higher throughput requirements – the memory becomes the most power hungry part of the decoder. A novel non-refresh eDRAM has been proposed to solve this issue. It takes advantage of the deterministic memory access pattern that can be found in most DSP applications in order to trade-off its power and access time. The new memory, replacing 70% of the memory used in the LDPC decoder, enables the 1.6 mm^2 65 nm LDPC decoder to achieve a peak throughput of 9 Gb/s at 89.5 pJ/b, and 1.5 Gb/s at 35.6 pJ/b with voltage and frequency scaling.

NB-LDPC code has a superior error-correcting performance than binary LDPC code at the cost of significantly higher decoding complexity. The merging of multiple bits to form a GF element reduces the number of edges in the NB-LDPC code's factor graph, permitting a fully parallel architecture. An NB-LDPC decoder has been implemented in 65 nm CMOS technology. With architectural improvements and dynamic clock gating, the decoder achieves a throughput of 1.22 Gb/s with energy efficiency of 3.03 nJ/b, and 698

Mb/s at 1.04 nJ/b with voltage and frequency scaling.

Polar code is a recently invented first provably capacity achieving code. Although its error-correcting performance is not capacity-achieving at finite block length, it is still competitive with binary LDPC codes. The main advantage of polar codes is its easy reconfigurability in code rate and regularly structured factor graph. These properties allow for a more efficient decoder design. With architectural improvements to the BP polar decoder, the resulting 1.48 mm^2 65 nm polar decoder achieves a throughput of 4.68 Gb/s at 15.5 pJ/b/iteration, and 779 Mb/s at 3.63 pJ/b/iteration with voltage and frequency scaling.

The presented channel decoders advance the state-of-the-art designs. Each decoder successfully reduces the total energy cost for future communication and storage applications.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] *IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems–Local and Metropolitan Area Networks–Specific Requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band*, IEEE Std. 802.11ad, Dec. 2012.

[2] *IEEE Standard for Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - Specific Requirements. Part 15.3: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs) Amendment 2: Millimeter-wave-based Alternative Physical Layer Extension*, IEEE Std. 802.15.3c, Oct. 2009.

[3] *IEEE Draft Standard for Information Technology–Telecommunications and Information Exchange between Systems-Local and Metropolitan Area Networks-Specific Requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment : Enhancements for Higher Throughput*, IEEE Std. 802.11n/D2.00, Feb. 2007.

[4] W. M. Regitz and J. A. Karp, "Three-transistor-cell 1024-bit 500-ns MOS RAM," *IEEE J. Solid-State Circuits*, vol. 5, no. 5, pp. 181–186, Oct. 1970.

[5] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[6] M. C. Davey and D. Mackay, "Low-density parity check codes over GF(q)," *IEEE Commun. Lett.*, vol. 2, no. 6, pp. 165–167, Jun. 1998.

[7] E. Arikan, "Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.

[8] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Trans. Commun.*, vol. 44, no. 10, pp. 1261–1271, Oct. 1996.

[9] *3GPP Standard TS 25.944 Rev. 4.1.0: 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Channel Coding and Multiplexing Examples*, 3GPP Organizational Partners Std. HSDPA, Jun. 2001.

[10] *3GPP Standard TS 36.212 Rev. 8.3.0: 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and Channel Coding (Release 9)*, 3GPP Organizational Partners Std. 3GPP-LTE, May 2008.

[11] *IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1*, IEEE Std. 802.16e, Feb. 2006.

[12] *ETSI Standard TR 102 376 V1.1.1: Digital Video Broadcasting (DVB) User Guidelines for the Second Generation System for Broadcasting, Interactive Services, News Gathering and Other Broadband Satellite Applications (DVB-S2)*, ETSI Std. TR 102 376, Feb. 2005.

[13] *IEEE Standard for Information Technology-Telecommunications and Information Exchange between Systems-Local and Metropolitan Area Networks-Specific Requirements Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, IEEE Std. 802.3an, Sep. 2006.

[14] A. Kavčić and A. Patapoutian, "The read channel," *Proc. IEEE*, vol. 96, no. 11, pp. 1761–1774, Nov. 2008.

[15] G. Dong, N. Xie, and T. Zhang, "On the use of soft-decision error-correction codes in NAND flash memory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 2, pp. 429–439, Feb. 2011.

[16] D. J. C. Mackay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[17] M. P. C. Fossorier, M. Mihaljević, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999.

[18] B. Xiang, D. Bao, S. Huang, and X. Zeng, "An 847-955 Mb/s 342-397 mW dual-path fully-overlapped QC-LDPC decoder for WiMAX system in 0.13 μm CMOS," *IEEE J. Solid-State Circuits*, vol. 46, no. 6, pp. 1416–1432, Jun. 2011.

[19] S.-W. Yen, S.-Y. Hung, C.-H. Chen, H.-C. Chang, S.-J. Jou, and C.-Y. Lee, "A 5.79-Gb/s energy-efficient multirate LDPC codec chip for IEEE 802.15.3c applications," *IEEE J. Solid-State Circuits*, vol. 47, no. 9, pp. 2246–2256, Sep. 2012.

[20] Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolić, "An efficient 10GBASE-T Ethernet LDPC decoder design with low error floors," *IEEE J. Solid-State Circuits*, vol. 45, no. 4, pp. 843–855, Apr. 2010.

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

[22] A. Darabiha, A. C. Carusone, and F. R. Kschischang, "Power reduction techniques for LDPC decoders," *IEEE J. Solid-State Circuits*, vol. 43, no. 8, pp. 1835–1845, Aug. 2008.

[23] Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolić, "A 47 Gb/s LDPC decoder with improved low error rate performance," in *IEEE Symp. VLSI Circuits Dig.*, Kyoto, Japan, Jun. 2009, pp. 286–287.

[24] A. Cevrero, Y. Leblebici, P. Ienne, and A. Burg, "A 5.35 mm^2 10GBASE-T Ethernet LDPC decoder chip in 90 nm CMOS," in *IEEE Asian Solid-State Circuits Conf.*, Beijing, China, Nov. 2010, pp. 317–320.

[25] M. Korb and T. G. Noll, "Area- and energy-efficient high-throughput LDPC decoders with low block latency," in *Proc. IEEE Eur. Solid-State Circuits Conf., ESSCIRC'11*, Helsinki, Finland, Sep. 2011, pp. 75–78.

[26] P. Urard, L. Paumier, V. Heinrich, N. Raina, and N. Chawla, "A 360 mW 105 Mb/s DVB-S2 compliant codec based on 64800b LDPC and BCH codes enabling satellite-transmission portable devices," in *IEEE Int. Solid-State Circuits Conf. Dig.*, San Francisco, CA, Feb. 2008, pp. 310–311.

[27] 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. 684–698, Mar. 2006.

[28] X.-Y. Shi, C.-Z. Zhan, C.-H. Lin, and A.-Y. Wu, "An 8.29 mm^2 52 mW multi-mode LDPC decoder design for mobile WiMAX system in 0.13 μm CMOS process," *IEEE J. Solid-State Circuits*, vol. 43, no. 3, pp. 672–683, Mar. 2008.

[29] C.-H. Liu, S.-W. Yen, C.-L. Chen, H.-C. Chang, C.-Y. Lee, Y.-S. Hsu, and S.-J. Jou, "An LDPC decoder chip based on self-routing network for IEEE 802.16e applications," *IEEE J. Solid-State Circuits*, vol. 43, no. 3, pp. 684–694, Mar. 2008.

[30] C.-L. Chen, K.-S. Lin, H.-C. Chang, W.-C. Fang, and C.-Y. Lee, "A 11.5-Gbps LDPC decoder based on CP-PEG code construction," in *Proc. IEEE Eur. Solid-State Circuits Conf., ESSCIRC'09*, Athens, Greece, Sep. 2009, pp. 412–415.

[31] F. Naessens, V. Derudder, H. Cappelle, L. Hollevoet, P. Raghavan, M. Desmet, A. M. AbdelHamid, I. Vos, L. Folens, S. O'Loughlin, S. Singirikonda, S. Dupont, J.-W. Weijers, A. Dejonghe, and L. V. der Perre, "A 10.37 mm^2 675 mW reconfigurable LDPC and Turbo encoder and decoder for 802.11n, 802.16e and 3GPP-LTE," in *IEEE Symp. VLSI Circuits Dig.*, Honolulu, HI, Jun. 2010, pp. 213–214.

[32] C. Roth, P. Meinerzhagen, C. Studer, and A. Burg, "A 15.8 pJ/bit/iter quasi-cyclic LDPC decoder for IEEE 802.11n in 90 nm CMOS," in *IEEE Asian Solid-State Circuits Conf.*, Beijing, China, Nov. 2010, pp. 313–316.

[33] X. Peng, Z. Chen, X. Zhao, D. Zhou, and S. Goto, "A 115 mW 1 Gbps QC-LDPC decoder ASIC for WiMAX in 65 nm CMOS," in *IEEE Asian Solid-State Circuits Conf.*, Jeju, Korea, Nov. 2011, pp. 317–320.

[34] M. Weiner, B. Nikolić, and Z. Zhang, "LDPC decoder architecture for high-data rate personal-area networks," in *Proc. IEEE Int. Symp. Circuits and Syst.*, Rio de Janeiro, Brazil, May 2011, pp. 1784–1787.

[35] X. Zhang and F. Cai, "Efficient partial-parallel decoder architecture for quasi-cyclic nonbinary LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 2, pp. 402–414, Feb. 2011.

[36] X. Chen, S. Lin, and V. Akella, "Efficient configurable decoder architecture for non-binary quasi-cyclic LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 1, pp. 188–197, Jan. 2012.

[37] Y.-L. Ueng, C.-Y. Leong, C.-J. Yang, C.-C. Cheng, K.-H. Liao, and S.-W. Chen, "An efficient layered decoding architecture for nonbinary QC-LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 2, pp. 385–398, Feb. 2012.

[38] C. Zhang and K. K. Parhi, "A network-efficient nonbinary QC-LDPC decoder architecture," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 6, pp. 1359–1371, Jun. 2012.

[39] X. Zhang, F. Cai, and S. Lin, "Low-complexity reliability-based message-passing decoder architectures for non-binary LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 11, pp. 1938–1950, Nov. 2012.

[40] Y.-L. Ueng, , K.-H. Liao, H.-C. Chou, and C.-J. Yang, "A high-throughput trellis-based layered decoding architecture for non-binary LDPC codes using Max-Log-QSPA," *IEEE Trans. Signal Process.*, vol. 61, no. 11, pp. 2940–2951, Jun. 2013.

[41] J. Lin and Z. Yan, "Efficient shuffled decoder architecture for nonbinary quasi-cyclic LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 9, pp. 1756–1761, Sep. 2013.

[42] ——, "An efficient fully parallel decoder architecture for nonbinary LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2013 (early access).

[43] F. García-Herrero, M. J. Canet, and J. Valls, "Nonbinary LDPC decoder based on simplified enhanced generalized bit-flipping algorithm," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2013 (early access).

[44] C. Leroux, I. Tal, A. Vardy, and W. J. Gross, "Hardware architectures for successive cancellation decoding of polar codes," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Prague, Czech Republic, May 2011, pp. 1665–1668.

[45] A. Alamdar-Yazdi and F. R. Kschischang, "A simplified successive-cancellation decoder for polar codes," *IEEE Commun. Lett.*, vol. 15, no. 12, pp. 1378–1380, Dec. 2011.

[46] A. Mishra, A. J. Raymond, L. G. Amaru, G. Sarkis, C. Leroux, P. Meinerzhagen, A. Burg, and W. J. Gross, "A successive cancellation decoder ASIC for a 1024-bit polar code in 180 nm CMOS," in *IEEE Asian Solid-State Circuits Conf.*, Kobe, Japan, Nov. 2012, pp. 205–208.

[47] C. Leroux, A. J. Raymond, G. Sarkis, and W. J. Gross, "A semi-parallel successive-cancellation decoder of polar codes," *IEEE Trans. Signal Process.*, vol. 61, no. 2, pp. 289–299, Jan. 2013.

[48] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast polar decoders: algorithm and implementation," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 946–957, May 2014.

[49] A. Pamuk, "An FPGA implementation architecture for decoding of polar codes," in *Proc. IEEE Int. Symp. Wireless Commun. Syst.*, Aachen, Germany, Nov. 2011, pp. 437–441.

[50] R. L. B. Kumar and N. Chandrachoodan, "A GPU implementation of belief propagation decoder for polar codes," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, Nov. 2012, pp. 1272–1276.

[51] D. Somasekhar, Y. D. Ye, P. Aseron, S.-L. Lu, M. M. Khellah, G. R. J. Howard, T. Karnik, S. Borkar, V. K. De, and A. Keshavarzi, "2 GHz 2 Mb 2T gain cell memory macro with 128 GBytes/sec bandwidth in a 65 nm logic process technology," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 174–185, Jan. 2009.

[52] K. C. Chun, P. Jain, J. H. Lee, and C. H. Kim, "A 3T gain cell embedded DRAM utilizing preferential boosting for high density and low power on-die caches," *IEEE J. Solid-State Circuits*, vol. 46, no. 6, pp. 1495–1505, Jun. 2011.

[53] K. C. Chun, P. Jain, T.-H. Kim, and C. H. Kim, "A 667 MHz logic-compatible embedded DRAM featuring an asymmetric 2T gain cell for high speed on-die caches," *IEEE J. Solid-State Circuits*, vol. 47, no. 2, pp. 547–559, Feb. 2012.

[54] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[55] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[56] C. Roth, A. Cevrero, C. Studer, Y. Leblebici, and A. Burg, "Area, Throughput, and Energy-Efficiency Trade-offs in the VLSI Implementation of LDPC Decoders," in *Proc. IEEE Int. Symp. Circuits and Syst.*, Rio de Janeiro, Brazil, May 2011, pp. 1772–1775.

[57] J. Barth, W. R. Reohr, P. Parries, G. Fredeman, J. Golz, S. E. Schuster, R. E. Matick, H. Hunter, C. C. Tanner, J. Harig, H. Kim, B. Khan, J. Griesemer, R. P. Havreluk, K. Yanagisawa, T. Kirihata, and S. S. Iyer, "A 500 MHz random cycle, 1.5 ns latency, SOI embedded DRAM macro featuring a three-transistor micro sense amplifier," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 86–95, Jan. 2008.

[58] P. J. Klim, J. Barth, W. R. Reohr, D. Dick, G. Fredeman, G. Koch, H. M. Le, A. Khargonekar, P. Wilcox, J. Golz, J. B. Kuang, A. Mathews, J. C. Law, T. Luong, H. C. Ngo, R. Freese, H. C. Hunter, E. Nelson, P. Parries, T. Kirihata, and S. S. Iyer, "A 1 MB cache subsystem prototype with 1.8 ns embedded DRAMs in 45 nm SOI CMOS," *IEEE J. Solid-State Circuits*, vol. 44, no. 4, pp. 1216–1226, Apr. 2009.

[59] J. Barth, D. Plass, E. Nelson, C. Hwang, G. Fredeman, M. Sperling, A. Mathews, T. Kirihata, W. R. Reohr, K. Nair, and N. Cao, "A 45 nm SOI embedded DRAM macro for the POWER^{TM} processor 32 MByte on-chip L3 cache," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 64–75, Jan. 2011.

[60] W. K. Luk, J. Cai, R. H. Dennard, M. J. Immediato, and S. V. Kosonocky, "A 3-transistor DRAM cell with gated diode for enhanced speed and retention time," in *IEEE Symp. VLSI Circuits Dig.*, Honolulu, HI, Jun. 2006, pp. 184–185.

[61] P. Meinerzhagen, A. Teman, R. Giterman, A. Burg, and A. Fish, "Exploration of sub-VT and near-VT 2T gain-cell memories for ultra-low power applications under

technology scaling," *J. Low Power Electron. Applicat.*, vol. 3, no. 2, pp. 54–72, Apr. 2013.

[62] S. Satpathy, Z. Foo, B. Giridhar, R. Dreslinski, D. Sylvester, T. Mudge, and D. Blaauw, "A 1.07 Tbit/s 128×128 swizzle network for SIMD processors," in *IEEE Symp. VLSI Circuits Dig.*, Honolulu, HI, Jun. 2010, pp. 81–82.

[63] S.-M. Yoo, J. M. Han, E. Haq, S. S. Yoon, S.-J. Jeong, B. C. Kim, J.-H. Lee, T.-S. Jang, H.-D. Kim, C. J. Park, D. I. Seo, C. S. Choi, S.-I. Cho, and C. G. Hwang, "A 256M DRAM with simplified register control for low power self refresh and rapid burn-in," in *IEEE Symp. VLSI Circuits Dig.*, Honolulu, HI, Jun. 1994, pp. 85–86.

[64] Y. S. Park, D. Blaauw, D. Sylvester, and Z. Zhang, "A 1.6-mm^2 38-mW 1.5-Gb/s LDPC decoder enabled by refresh-free embedded DRAM," in *IEEE Symp. VLSI Circuits Dig.*, Honolulu, HI, Jun. 2012, pp. 114–115.

[65] L. Zeng, L. Lan, Y. Y. Tai, S. Song, S. Lin, and K. Abdel-Ghaffar, "Construction of nonbinary quasi-cyclic LDPC codes: a finite field approach," *IEEE Trans. Commun.*, vol. 56, no. 4, pp. 545–554, Apr. 2008.

[66] S. Song, B. Zhou, S. Lin, and K. Abdel-Ghaffar, "A unified approach to the construction of binary and nonbinary quasi-cyclic LDPC codes based on finite fields," *IEEE Trans. Commun.*, vol. 57, no. 1, pp. 84–93, Jan. 2009.

[67] B. Zhou, J. Kang, S. Song, S. Lin, K. Abdel-Ghaffar, and M. Xu, "Construction of non-binary quasi-cyclic LDPC codes by arrays and array dispersions," *IEEE Trans. Commun.*, vol. 57, no. 4, pp. 1652–1662, Jun. 2009.

[68] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular (2, d_c)-LDPC codes over GF(q) using their binary images," *IEEE Trans. Commun.*, vol. 56, no. 10, pp. 1626–1635, Oct. 2008.

[69] S. Kim and G. E. Sobelman, "Scaling, offset, and balancing techniques in FFT-based BP nonbinary LDPC decoders," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 5, pp. 277–281, May 2013.

[70] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 633–643, Apr. 2007.

[71] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low-complexity decoding for non-binary LDPC codes in high order fields," *IEEE Trans. Commun.*, vol. 58, no. 5, pp. 1365–1375, May 2010.

[72] V. Savin, "Min-Max decoding for non binary LDPC codes," in *Proc. IEEE Int. Symp. Information Theory*, Toronto, Canada, Jul. 2008, pp. 960–964.

[73] E. Boutillon and L. Conde-Canencia, "Bubble check: a simplified algorithm for elementary check node processing in extended min-sum non-binary LDPC decoders," *IEEE Electron. Lett.*, vol. 46, no. 9, pp. 633–634, Apr. 2010.

[74] Y. S. Park, Y. Tao, and Z. Zhang, "A 1.15Gb/s fully parallel nonbinary LDPC decoder with fine-grained dynamic clock gating," in *IEEE Int. Solid-State Circuits Conf. Dig.*, San Francisco, CA, Feb. 2013, pp. 422–423.

[75] Y. S. Park, D. Blaauw, D. Sylvester, and Z. Zhang, "Low-power high-throughput LDPC decoder using non-refresh embedded DRAM," *IEEE J. Solid-State Circuits*, vol. 49, no. 3, pp. 783–794, Mar. 2014.

[76] X. Zhang and F. Cai, "Reduced-complexity decoder architectures for non-binary LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 7, pp. 1229–1238, Jul. 2011.

[77] X. Chen and C.-L. Wang, "High-throughput efficient non-binary LDPC decoder based on simplified min-sum algorithm," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 11, pp. 2784–2794, Nov. 2012.

[78] F. Cai and X. Zhang, "Relaxed min-max decoder architectures for nonbinary low-density parity-check codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 11, pp. 2010–2023, Nov. 2013.

[79] Y. S. Park, Y. Tao, S. Sun, and Z. Zhang, "A 4.68Gb/s belief propagation polar decoder with bit-splitting register file," in *IEEE Symp. VLSI Circuits Dig.*, Honolulu, HI, Jun. 2014, pp. 144–145.

[80] R. Mori and T. Tanaka, "Performance and construction of polar codes on symmetric binary-input memoryless channels," in *Proc. IEEE Int. Symp. Information Theory*, Seoul, Korea, Jun. 2009, pp. 1496–1500.

[81] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6562–6582, Oct. 2013.