LOW POWER LOW-DENSITY PARITY-CHECKING (LDPC) CODES DECODER DESIGN USING DYNAMIC VOLTAGE AND FREQUENCY SCALING

A Thesis

by

WEIHUANG WANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2007

Major Subject: Electrical Engineering

LOW POWER LOW-DENSITY PARITY-CHECKING (LDPC) CODES DECODER DESIGN USING DYNAMIC VOLTAGE AND FREQUENCY SCALING

A Thesis

by

WEIHUANG WANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Gwan Choi
Committee Members,	A. L. Narasimha Reddy
	Jean-Francois Chamberland
	Duncan M. Walker
Head of Department,	Costas N. Georghiades

December 2007

Major Subject: Electrical Engineering

ABSTRACT

Low Power Low-Density Parity-Checking (LDPC) Codes Decoder Design Using Dynamic Voltage and Frequency Scaling. (December 2007)

Weihuang Wang, B.S., Peking University

Chair of Advisory Committee: Dr. Gwan Choi

This thesis presents a low-power LDPC decoder design based on speculative scheduling of energy necessary to decode dynamically varying data frame in both block-fading channels and general AWGN channels. A model of a memory-efficient low-power high-throughput multi-rate array LDPC decoder as well as its FPGA implementation results is first presented. Then, I propose a decoding scheme that provides the feature of constant-time decoding and thus facilitates real-time applications where guaranteed data rate is required. It pre-analyzes each received data frame to estimate the maximum number of necessary iterations for frame convergence. The results are then used to dynamically adjust decoder frequency and switch between multiple-voltage levels; thereby energy use is minimized. This is in contrast to the conventional fixed-iteration decoding schemes that operate at a fixed voltage level regardless of the quality of data received. Analysis shows that the proposed decoding scheme is widely applicable for both two-phase message-passing (TPMP) decoding algorithm and turbo decoding message passing (TDMP) decoding algorithm in block fading channels, and it is independent of the specific LDPC decoder architecture. A decoder architecture utilizing our recently published multi-rate decoding architecture for general AWGN channels is also presented. The result of this thesis is a decoder design scheme that provides a judicious trade-off between power consumption and coding gain.

To My parents

ACKNOWLEDGMENTS

This thesis would not have been possible without the boundless assistance of mentors, colleagues, and friends. I would like to express my gratitude to my advisor, Dr. Gwan Choi. Dr. Choi encouraged and supported my research. It is always a pleasure to sit together with Dr. Choi for discussion. He has provided me with tremendous help for not only my academic research, but also for my career and life. Thanks to Dr. Narasimha Reddy, Dr. Jean-Francois Chamberland and Dr. Duncan M. Walker, for serving on my committee and for taking the time to discuss with me.

I would also like to express my heartfelt thanks to the other professors who have always kept their doors open, ready to discuss and encourage new ideas; particular thanks go to Dr. Sunil Khatri and Dr. Alexander Sprintson. During my time at Texas A&M University, I have had the opportunity to collaborate with a number of different people. I would like to thank Dr. Kiran Gunnam for the successful and joyful collaboration. Dr. Gunnam also provided me with important suggestions and opportunities for my career.

Additionally, thanks to Qiuyang Li, Yang (Cindy) Yi, Zhanyuan (Jerry) Jiang, Hang Su, Sanghoan Chang. Finally, I would like to express my sincere thanks to my family whose unwavering support has been indispensable to me over the last years. Thank you to my family: Dad, Mom, my elder sisters and brothers.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Ultra-low-power decoding is a necessity in communication for wireless portable devices. A competing coding scheme to the popular Turbo Coding is Low Density Parity Check (LDPC) codes that are special cases of error correcting codes originally proposed by Gallager [1] in 1960's and rediscovered in late 1990's [2]. The LDPC codes inherently exhibit simple streamlined algorithmic flow that supports highly regular hardware structure to implement. And such structure lends to power-efficient strategies. Like the Turbo coding, LDPC also achieves Shannon limit approaching performance and a number of efficient implementation of LDPC decoders [3, 4] have been proposed. LDPC has successfully been adopted in next-generation standards, such as IEEE802.16e, DVB-S2, etc. Nevertheless, implementation of LDPC in these applications imposes significant challenges for the real-time requirements.

The commonly used message passing algorithms for LDPC decoding exchange information between the bit nodes and check nodes (parity checks constraining the bits) in an iterative fashion. There are again popular derived algorithm, including the two phase message passing (TPMP) algorithm and turbo decoding message passing (TDMP, also known as layered decoding) algorithm [5]. In practice, the LDPC decoder is typically set to run until a preset maximum number of iteration (e.g. 20), depending on the code rate, if no data convergence is observed. There have been researches on early termination of frames which can not be decoded even maximum iterations are applied [6, 7]. Hardware is dynamically switched off when additional iteration will not amount to increase in decoding performance. In both of these pa-

The journal model is *IEEE Transactions on Automatic Control.*

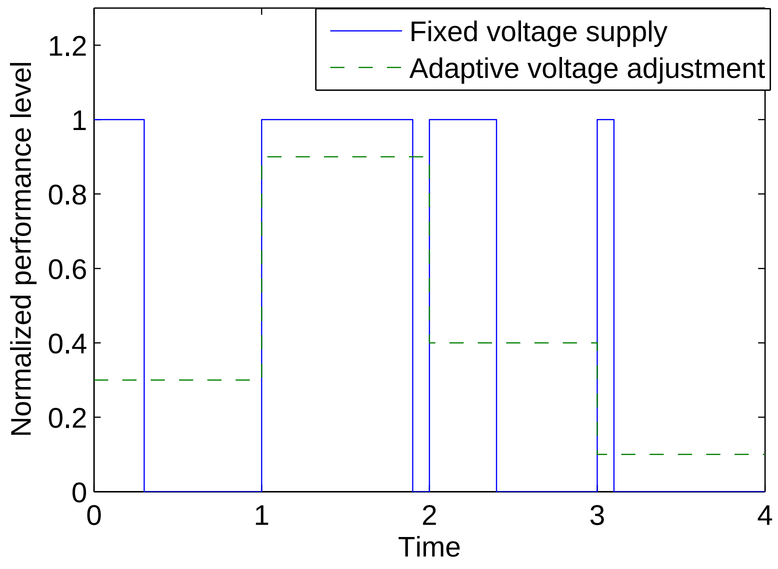


Fig. 1. Performance level adjustment for power reduction.

pers, however, early termination of the iterative process is determined by checking the messages during the decoding process. These approaches result in excessive hardware overhead and no predictions are made prior to the commencement of decoding iterations. The authors in [8] lower the power consumption by reducing precision at the least significant bits in soft digital signal processing system (filter for example).

Real-time applications pose restrictions on the decoding time schedule. In addition, power consumption is always an important design constraint for the mobile applications. In this paper, we propose a novel scheme to dynamically configure the decoding hardware to achieve, while guaranteeing quality of service (QoS) for time-sensitive data, minimum energy consumption for block-fading channel [9], where noise level changes slowly enough that variation within a frame is negligible. The proposed energy-scheduling scheme is illustrated in Figure 1. The figure shows clock frequency over a time period. Each frame requires different number of iterations to complete decoding. With fixed-voltage architecture, the result is decoder operating for a given amount of time before it is turned off. Decoding resumes when a next frame is received. In contrary, our proposed scheme adjusts clock frequency and voltage level appropriately to ascertain timely termination of decoding before the next frame is received. An increased number of decoding iterations are taken for lower-SNR data frames; thus the iteration process is completed in less amount of time. The aim is to keep the total decoding time constant for all data frames. This is achieved by utilizing the recently developed dynamic voltage and frequency scaling (DVFS) techniques [10, 11]. System power dissipation is proportional to occurrence of activity and quadratic voltage supply [12]. The number of decoding iterations predicted from channel data is used to determine the amount of energy and operating frequency necessary to decode each frame, within a fixed time period. Thereby, making available the output of each frame synchronized to the fixed rate at which the data is consumed in real-time application interface.

The rest of this thesis is organized as following: In chapter II, the background of LDPC code together with decoding algorithm and DVFS are briefly described. Chapter III presents modeling and FPGA implementation of a memory and energy efficient LDPC decoding architecture based on TDMP. In Chapter IV, the low-power real-time decoding policy for block-fading channel is presented. It is shown that the empirical policy works well across different iterative decoding algorithms and the policy is independent of any specific LDPC decoder architectures. Coding performance, as well as savings in decoding energy is shown. Chapter V discusses the low-power real-time decoding architecture for general non-fading AWGN channel. Chapter VI concludes the thesis.

CHAPTER II

BACKGROUND

This chapter describes the background of LDPC codes, the belief-propagation decoding algorithm as well as the layered decoding algorithm. The power estimation technique is also introduced.

A. LDPC codes and decoding algorithms

LDPC codes are defined by a sparse parity check matrix $H = [Hmn]$ that consists mostly of 0's. First introduced by Gallager [1], the H matrix of (n, d_c, d_v) regular LDPC code has the following properties. Each column contains a small fixed number d_v of 1's and each row contains a small fixed number $d_c > d_v$ of 1's. The block length of this code is n which is equal to the number of columns in the H matrix. Suppose that the number of data bits before the channel encoding is l, then the number of rows of this H matrix is $m = n - l$. Rate of this code is defined as $l/n = 1 - d_v/d_c$. The code words consist of all one-dimensional row vectors that span the null space of the parity check H matrix. The number for d_v and dc should be no less than 3 and 6 respectively, for good coding performance. The array LDPC codes [13] used in this paper is specified by three parameters: a prime number p and two integers k and j such that $j < p$ and $k < p$. It is given by Equation 2.1:

$$H = \begin{bmatrix} I & I & I & \cdots & I \\ I & \alpha & \alpha^2 & \dots & \alpha^{k-1} \\ I & \alpha^2 & \alpha^4 & \dots & \alpha^{2(k-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ I & \alpha^{j-1} & \alpha^{(j-1)2} & \cdots & \alpha^{(j-1)(k-1)} \end{bmatrix} \tag{2.1}$$

where I is the $p \times p$ identity matrix, and α is a $p \times p$ permutation matrix representing a single left or right cyclic shift of I. Power of α in H denotes multiple cyclic shifts, with the number of shifts given by the value of the exponent.

Assume binary phase shift keying (BPSK) modulation (a 1 is mapped to -1 and a 0 is mapped to 1) over an additive white Gaussian noise (AWGN) channel. The received values y_n are Gaussian with mean $x_n = 1$ and variance δ^2. The iterative two-phase message-passing (TPMP) algorithm, also known as belief-propagation (BP) algorithm [14, 15] is computed in two phases. One is a check node processing and the other is variable node processing. In the check node processing, each row of the parity matrix is checked to verify that parity check constraints are satisfied. In the variable node processing the probability will be updated by summing up the other probabilities from the rest of the rows and the a priori probabilities from the channel output. The message-passing algorithm can be simplified to the belief-propagation based algorithm (also called Min-Sum algorithm) [16]. While greatly reducing the decoding complexity in implementation, the Min-Sum degrades the coding performance. The improved BP based algorithm, Normalized-Min-Sum and Offset-Min-Sum [16] eliminates this performance degradation.

Following the same notation in [17], the check node processing can be expressed as:

$$R_{mn}^{(i)} = \delta_{mn}^{(i)} \max\left(\kappa_{mn}^{(i)} - \beta, 0\right), \tag{2.2}$$

$$\kappa_{mn}^{(i)} = \left|R_{mn}^{(i)}\right| = \min_{n' \in \mathrm{N}(m) \backslash n} \left|Q_{n'm}^{(i-1)}\right|, \tag{2.3}$$

where $Q_{nm}^{(i)}$ is the message from variable node n to check node m, $R_{mn}^{(i)}$ is the message from check node m to variable node n, and superscript i denotes the i^{th} iteration. $M(n)$ is the set of the neighboring check nodes for variable node n, and $N(m)$ is the set of the neighboring variable nodes for check node m, β is a positive constant and depends on the code parameters [16]. The sign of check-node message $R_{mn}^{(i)}$ is defined as

$$\delta_{mn}^{(i)} = \left(\prod_{n' \in \mathrm{N}(m) \backslash n} \operatorname{sgn}\left(Q_{n'm}^{(i-1)} \right) \right). \tag{2.4}$$

In the vairalbe node processing,

$$P_n = L_n^{(0)} + \sum_{m \in M(n)} R_{mn}^{(i)} \tag{2.5}$$

where the log-likelihood ratio of bit n is $L_n^{(0)} = y_n$. For final decoding

$$P_n = L_n^{(0)} + \sum_{m \in M(n)} R_{mn}^{(i)} \tag{2.6}$$

A hard decision is taken by setting $\hat{x}_n = 0$ if $P_n(x_n) \geq 0$, and $\hat{x}_n = 1$ if $P_n(x_n) \leq 0$. If $\hat{x}_n H^T = 0$, the decoding process is finished with $\hat{x}_n$ as the decoder output; otherwise, repeat processing of Equation 2.2-Equation 2.6. If the decoding process does not end within predefined maximum number of iterations, it_{max}, stop and output an error message flag and proceed to the decoding of the next data frame.

Mansour, *et al.* [5] introduce the concept of layered decoding (also known as turbo decoding message passing, TDMP) algorithm. In layered decoding algorithm, the array LDPC with j block rows are taken as a concatenation of j constituent sub-codes. After the check-node processing of one layer, the updated messages are immediately used to calculate the variable-node message, whose results are then applied to next layer of sub-code. Each iteration in the layered decoding algorithm is composed of j sub-iterations.

Mathematically, $\vec{R}_{l,n}^{(0)}$ is initialized to 0 and $\vec{P}_n$ to $\vec{L}_n^{(0)}$, then

For $i = 1,\ 2,\ \cdots,\ i_{tmax}$,

For $l = 1,\ 2, \cdots,\ d_v$,

For $n = 1,\ 2, \cdots,\ d_c$

$$\left[\vec{Q}_{l,n}^{(i)}\right]^{S(l,n)} = \left[\vec{P}_n\right]^{S(l,n)} - \vec{R}_{l,n}^{(i-1)} \tag{2.7}$$

$$\vec{R}_{l,n}^{(i)} = f\left(\left[\vec{Q}_{l,n'}^{(i)}\right]^{S(l,n')}\right), \forall n' = 1, 2, \cdots, d_c \tag{2.8}$$

$$\left[\vec{P}_n\right]^{S(l,n)} = \left[\vec{Q}_{l,n}^{(i)}\right]^{S(l,n)} + \vec{R}_{l,n}^{(i)} \tag{2.9}$$

This thesis has analyzed both TPMP and TDMP algorithms based on architecture from [3].

B. Circuit power estimation and reduction

There are three major sources of power dissipation in CMOS circuit [12]:

$$\begin{aligned} P_{total} &= P_{switching} + P_{SC} + P_{leakage} \\ &= \alpha C_L \Delta V V_{dd} f_{clk} + I_{SC} V_{dd} + I_{leakage} V_{dd}, \end{aligned} \tag{2.10}$$

$P_{switching}$ represents the switching power resulted from charging and discharging parasitic capacitances in the circuit. C_L is the loading capacitance, f_{clk} is the clock frequency, and a is the node transition factor defined as the probability that a power consuming transition occurs. In most cases, the voltage swing ΔV is the same as the supply voltage V_{dd}. The short circuit power P_{SC} is caused by direct-path short circuit current I_{SC} which arises when both NMOS and PMOS are simultaneously turned on. This is caused by the finite rising and falling time of input signal. The short circuit power can be kept within 15% of the switching power if carefully designed [18]. $P_{leakage}$ is the leakage component of power, where $I_{leakage}$ is the total leakage current in CMOS circuit. Further, delay of the circuit increase with decreased voltage supply, as shown in Equation 2.11:

$$\tau = \frac{1}{f_{clk}} = \frac{C_L V_{dd}}{I_{dsat}} \propto \frac{V_{dd}}{(V_{dd} - V_{th})^{1.3}} \tag{2.11}$$

Typically, switching power is the main source of power dissipation in the circuit. It should be noted that while power consumption decreases linearly with the operation frequency, the time for finishing the certain workload increases. As a result, the total energy consumption remains the same for the same workload if the power supply is not changed. Dynamic voltage and frequency scaling is an effective method to reduce the energy consumption, especially under wide variations in workload. A number of DVFS designs have been presented. This thesis follows the design described in [9].

CHAPTER III

MODELING AND FPGA IMPLEMENTATION OF LAYERED LDPC DECODER

This chapter presents modeling of an memory-efficient low-power multi-rate LDPC decoder as well as its FPGA implementation. Micro-architecture of the check-node processing unit is first introduced. Architecture of the decoder and its FPGA implementation results are then shown.

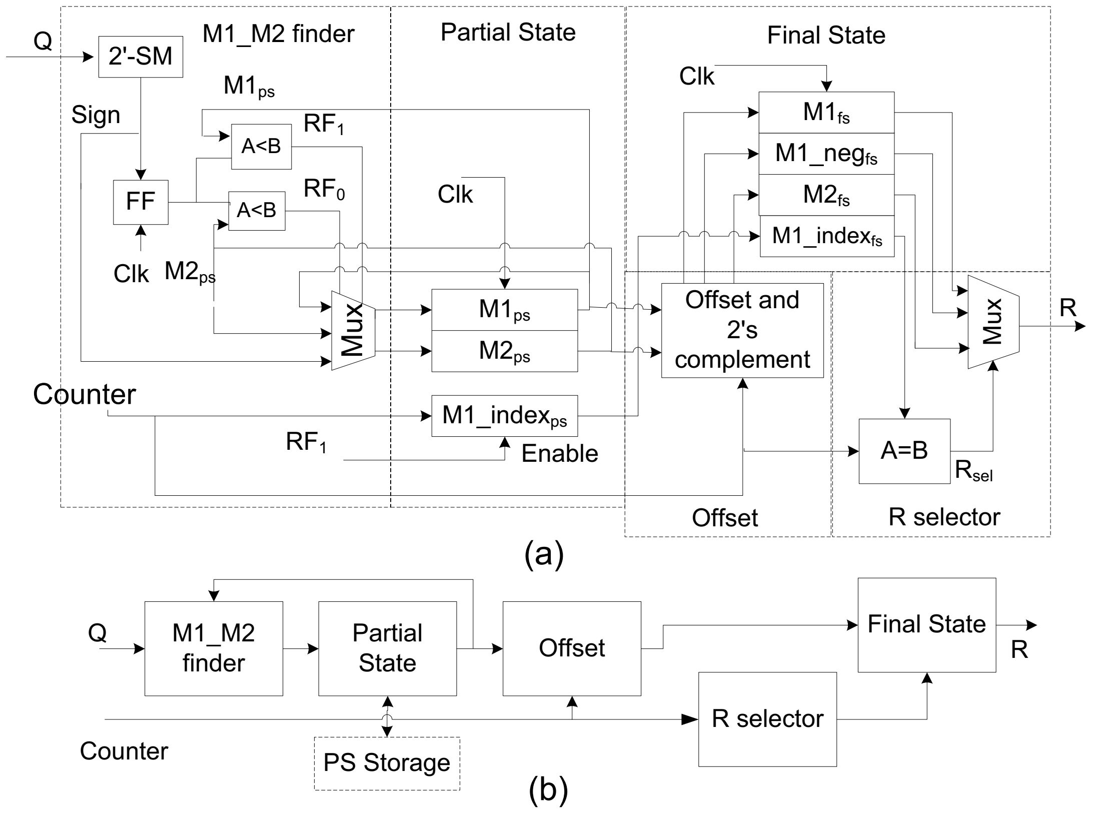


Fig. 2. (a) Micro-architecture of CNU. (b) Block diagram of CNU.

The check node unit (CNU) design, as discussed more fully in [3, 19, 20], utilize the less known property of the offset-based min-sum algorithm. In this algorithm, the check node processing produces only two different output magnitude values irrespective of the number of incoming variable to check messages. This property greatly simplifies the number of comparisons required as well as the memory needed to store CNU outputs. Figure 2 shows the serial CNU architecture for a ($d_v = 5$, $d_c = 25$) code. We use the same notation as in Chapter II: R for check node messages, Q for variable node messages, and P for the sum of variable node messages and channel values. In the first 25 clock cycles of the check node processing, incoming variable messages are compared with the two up-to-date least minimum numbers (partial state, PS) to generate the new partial state. The partial state includes the least minimum value, $M1$, the second minimum value $M2$, and index of $M1$. The final state (FS) is then computed by offsetting the partial state. It should be noted that the final state includes only $M1$, $-M1$, $+/-M2$ with offset correction. This results in a reduction of around 50%-90% of R memory based on the rate of the code, which has dependence on check node degree dc, when compared to BCJR algorithm [5] or sum of products algorithm. Since the sign information of R memory is still stored, the total savings in R memory is around 40%-72% for 5-bit messages based on the rate of the code.

The data flow graph, based on the TDMP algorithm and the value reuse property of min-sum algorithm is shown in Figure 3. For ease of discussion, we will illustrate the architecture for a specific structured code: array code of length 1525 described in Chapter II, $d_v = 5$, $d_c = 25$ and $p = 61$, the discussion can be easily generalized to any other structured codes. First, functionality of each block in the architecture is explained. A check-node process unit (CNU) is the serial CNU based on OMS described in the previous section. The CNU array is composed of p computation units that compute the partial state for each block row to produce the R messages in block serial fashion. Since final state of previous block rows, in which the compact information for CNU messages is stored is needed for TDMP, it is stored in register banks.

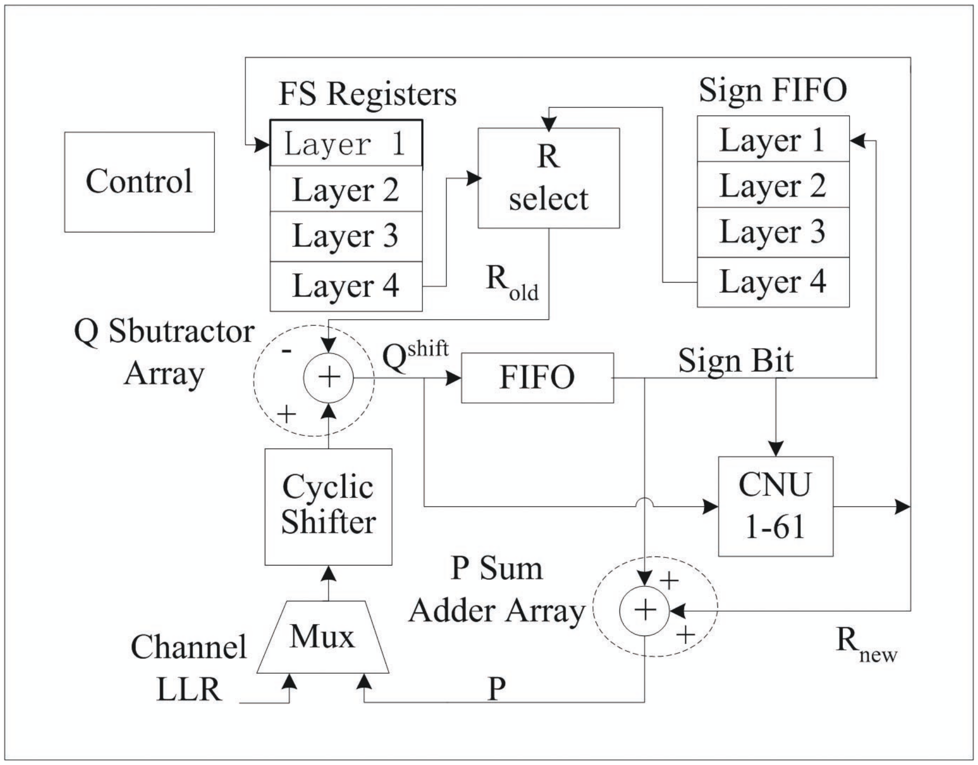


Fig. 3. LDPC decoder using layered decoding and OMS.

There is one register bank of depth $d_v - 1$, which is 4 in this case, connected with each CNU. Each final state is the same as the final state register bank in the CNU. Besides the shifted Q messages, the CNU array also take input of the sign information for previous computed R messages in order to perform R selection operation. The sign bits are stored in sign FIFO. The total length of sign FIFO is d_c and each block row has p one bit sign FIFOs. We need $d_v - 1$ of such FIFO banks in total. p number of R select units is used for R_{old}. An R select unit generates the R messages for d_c edges of a check-node from three possible values stored in final state register associated with that particular check-node in a serial fashion. Its functionality and structure is the same as the block denoted as R select in CNU. This unit can be treated as de-compressor of the check node edge information which is stored in compact form in FS registers. The generation of R messages for all the layers in this way amounts to significant memory savings, which would be quantified in a later section. The shifter is constructed as cyclic down logarithmic shifter to achieve the cyclic shifts specified by the binary encoded value of the shift. The logarithmic shifter is composed of $log_2(p)$ stages of p switches. Since cyclic up shift is also needed in the operation of the decoder, cyclic up shift by u can be simply achieved by doing cyclic down shift with p-u on the vector of size p. The decoding operation proceeds as per the vector equations described in Chapter II. The shift value can be either positive or negative indicating that a down shift or up shift need to be performed by the cyclic shifter.

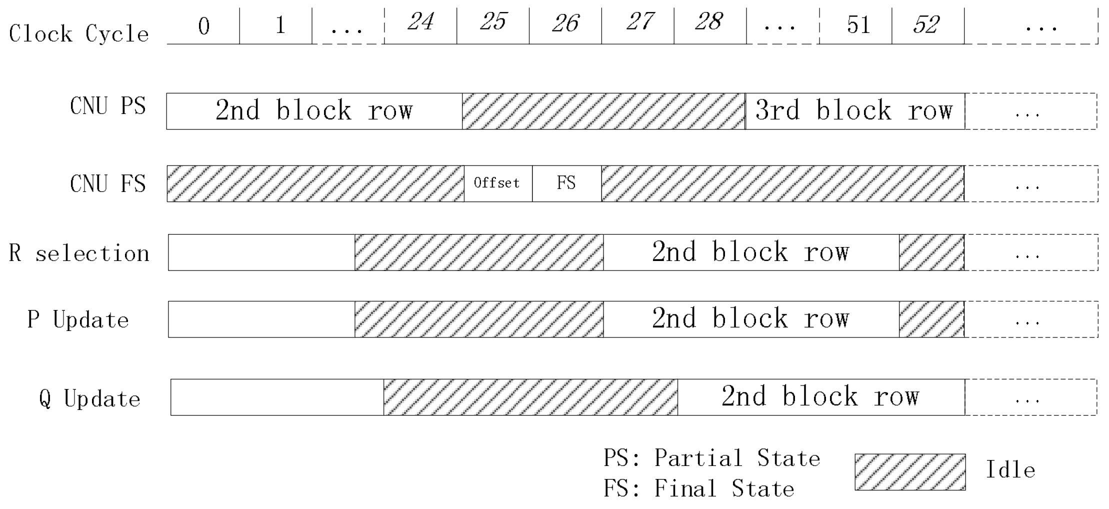


Fig. 4. Simplified diagram for block serial processing and 3-stage pipelining for TDMP using OMS.

Snapshot of the pipeline of the decoder is shown in Figure 4. The partial state stage in CNU (CNU PS) is first operating on the 2^{nd} block row. Final state stage in CNU (CNU FS) can not start until the end of PS processing. As soon as the FS is done, R select is able to select the output R messages, and P and Q messages processing starts. With the first block of Q message ready, PS for the next block row can be started immediately.

This architecture is scalable for scalable for low-throughput application. Assume the desired parallelization is M, and $M < p$, and M is power of 2 here. Number of instantiations of in the CNU array would be M and the cyclic shifter needed is $M \times M$. Since it is needed to achieve $p \times p$ cyclic shift with consecutive shifts of $M \times M$,it is necessary that the complete vector of size p is available in M banks with the depth $s = (ceil(p/M))$ and shifting is achieved in part by the cyclic shifter, and in part by the address generation. Now, all the CNU and variable node processing is done in a time division multiplexed fashion for each sub-vector of length M, so as to process the vector of size p to mimic the pipeline in Figure 4.

The TDMP LDPC decoding architecture has been modeled, clock cycle by clock cycle in Matlab. It has also been implemented on Xilinx 2V8000 FPGA. The synthesis results and performance comparison with other recent state of the art implementations are given in Table I. The work in [21] and [22] only supports one fixed code length. Similarly the work in [23] supports one code rate only while supporting different lengths from 1000-3000. The amount of memory that needs to be used in multi-rate architectures is dependent on the maximum values of code parameters that need to be supported.

Table I. FPGA implementation and performance comparison.

	Hocevar, *et al.* [21]	M. Karkooti, *et al.* [22]	T. Brack, *et al.* [23]	Implemented ($M = p = 61$)	Implemented ($M = 61$,$p = 347$)
Slices	N/A	11,35	14,475	6,002	6,182
LUT	72,621	20,374	N/A	7,713	8,022
SFF	6,779	N/A	N/A	9,981	10,330
BRAM	32	66	165	12	102
Frequency (MHz)	44	121	100	112	112
Throughput (Mbps)	80	127	180	$\frac{1{,}366 \times code\ rate}{number\ of\ iterations}$, 68.3-125.4	$\frac{2{,}277 \times code\ rate}{number\ of\ iterations}$ 113-159

CHAPTER IV

SPECULATIVE ENERGY SCHEDULING IN DECODING FOR BLOCK FADING CHANNEL

The key observation made for speculative-energy decoding is that for most of the data frames, the decoding process is finished before the maximum number of decoding iterations. Thus the aim is to allot only sufficient amount of energy to complete decoding before the next frame is received. The mechanism to achieve this is that frequency and voltage can be tuned as discussed in Chapter II. Such performance adjustment is feasible because the observation that severity of noise corruption of channel data can be estimated in advance.

Based on statistical analysis of the received data from channel, we develop a heuristic decoding policy of LDPC codes wherein a judicious consumption of energy is made. In the proposed scenario, decoding process is dynamically adjusted so that the maximum number of decoding iteration for each frame is set close to optimum in term of energy and coding performance.

A. Analysis of TPMP decoding algorithm

In this thesis, simulation of the TPMP decoding algorithm is carried out based on randomly constructed (3, 6) 1/2 rate code with block length of 2048 over a block fading channel, assuming Gaussian noise and code length equal to fading block length. The discussion and conclusion could be also extended to other LDPC codes.

Severity of noise corruption is first observed from the number of checks in error from the channel data. For a (d_c, d_v) regular LDPC code, suppose there is one bit in error, the number of checks violated will be d_c if a hard decision is made. When multiple data bits are flipped, depending on the position of the flipped bits with

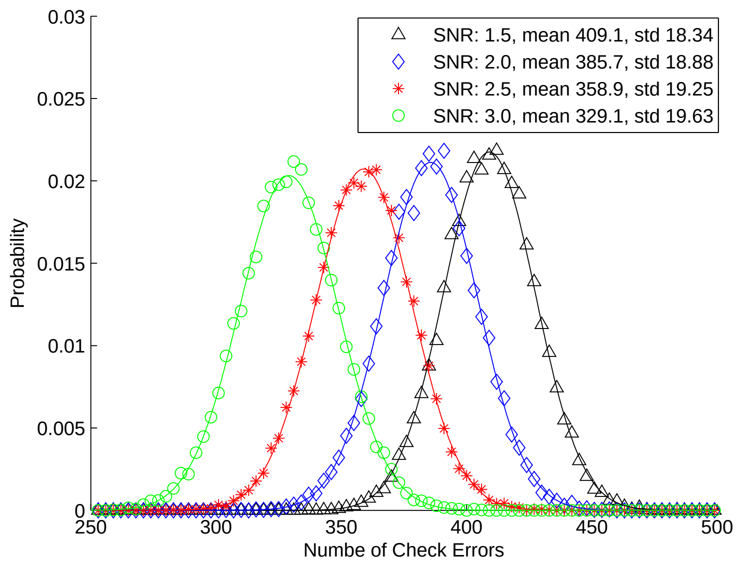


Fig. 5. Probability density function of number of check error. (std: standard deviation)

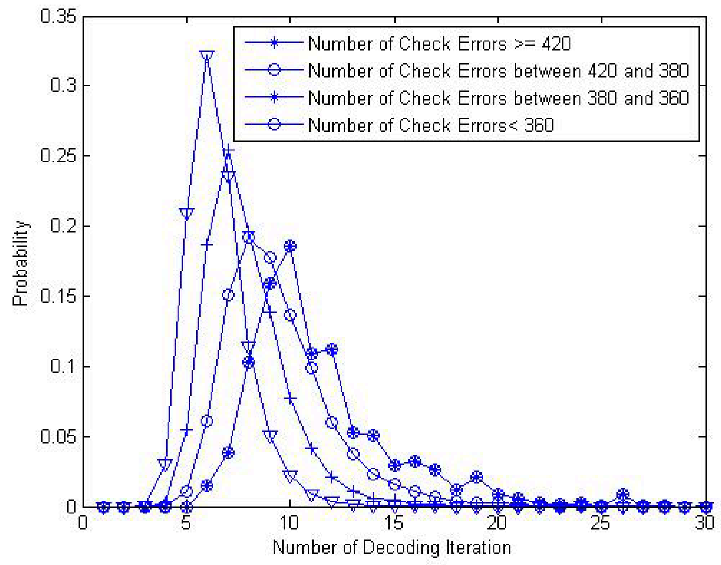


Fig. 6. Distribution of decoding iteration for different number of check errors

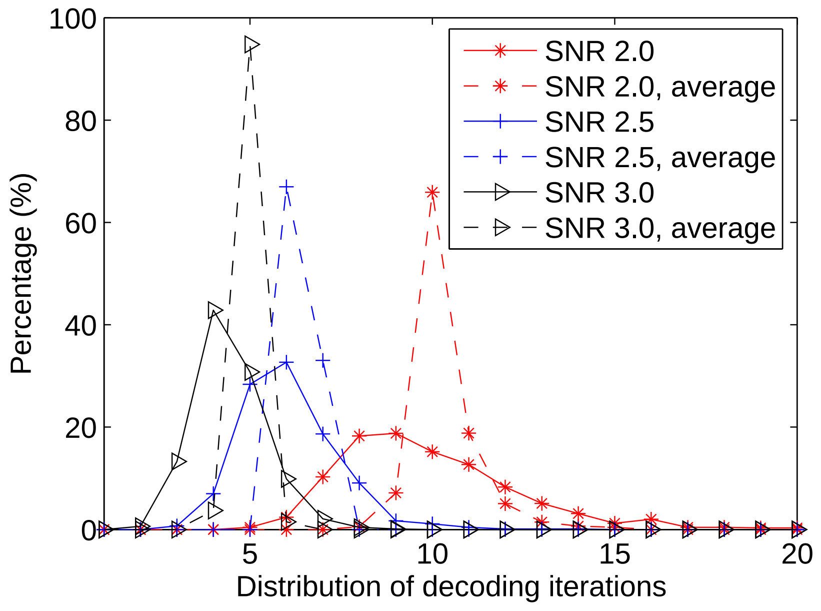


Fig. 7. Distribution of decoding iterations at different SNR levels for each frame as well as average decoding iterations of every 10 frames

respect to the H matrix, the numbers of checks in error are analyzed statistically. Figure 5 shows that at given SNR, the number of received bits in error is consistent with Gaussian distribution. The average number of check errors decreases linearly. Figure 6 shows that the number of soft decoding iteration required varies with different number of checks in error. The more checks in error, the more decoding effort is needed. Number of checks in error carries part of the information about the severity of the damage in the frame.

In addition, it is prudent to track the number of decoding iterations of the past frames. That can be used to compensate the large variance in decoding iterations from check-error estimation. It is generally accepted that higher SNR level requires less number of decoding iterations. Figure 7 shows a statistical relationship between SNR and number of decoding iterations. The average number of decoding iterations for multiple frames is highly correlated with SNR, and almost all frames are decoded after 1.5 times of the average decoding iterations. In a slowly fading communication channel, channel condition is unlikely to change abruptly, which means that it is possible to estimate the SNR level for incoming channel data. Based on the information of average decoding iterations of past few frames and number of checks in error for incoming frame, we can estimate an upper bound for the decoding iteration with high confidence level.

B. Analysis of TDMP decoding algorithm

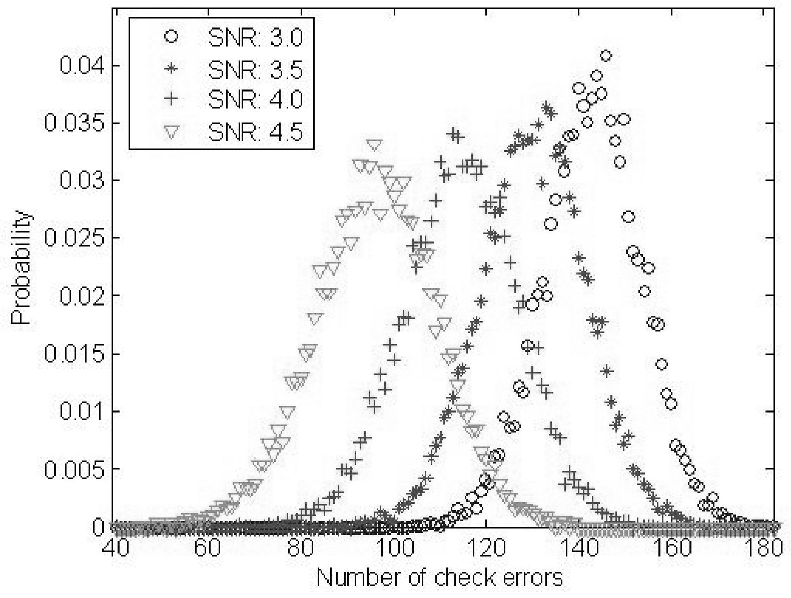


Fig. 8. Probability density function of number of check errors for array codes ($p = 67$, $d_c = 25$ and $d_v = 5$).

For analysis of TDMP deciding algorithm, the sample LDPC code is conducted based on rate 0.8 array code with expansion factor p of 67, check-node degree of 25, variable-degree of 5, and block length of 1675 over a block fading channel, assuming Gaussian noise where code length is equal to fading block length. It aims to show that out proposed decoding scheme can be applied to different algorithms for different codes.

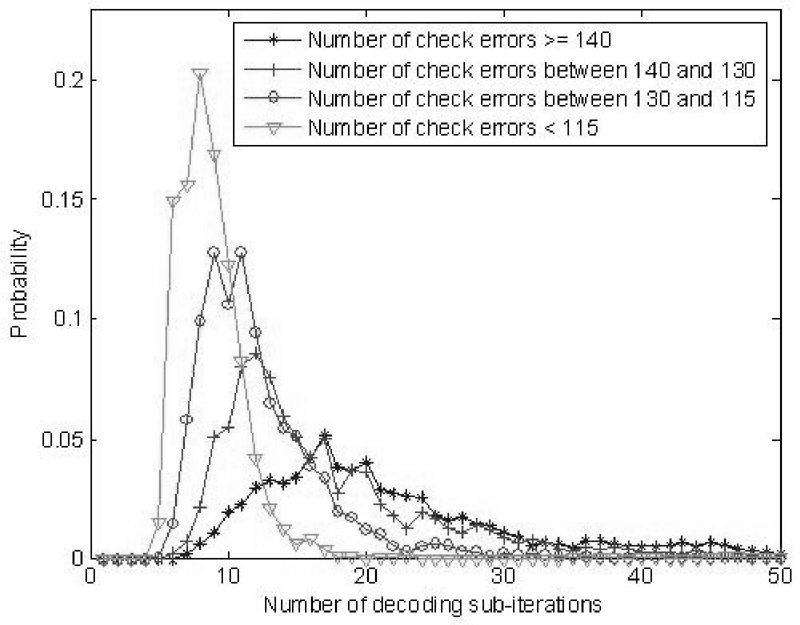


Fig. 9. Distribution of decoding iteration for different number of check errors.

Figure 8 shows that at given SNR, the number of checks in error for received data is consistent with Gaussian distribution, with the average number of check errors decreases linearly. Figure 9 shows that the number of soft decoding iteration required for different frames varies with different number of checks in error. The more checks in error, the more decoding iteration is required on average. Number of checks in error carries part of the information about the severity of the corruption in the frame. The average decoding sub-iterations (E) for the four cases (i.e. number of check errors larger than 140, between 140 and 130, between 130 and 115, as well as less than 115) are 19.6, 17.3, 12.5 and 8.8, respectively, and the corresponding standard deviation (δ) is 17.7, 12.3, 7.1 and 3.3. The $E+2\delta$ points are then 55, 42, 27 and 16 respectively. There are only 4.7%, 4.3%, 2.6% and 1.1% loss of erroneous frames in each category as compared with the situation where 100 sub-iterations are applied. In summary, we utilize the information of number of check errors to determine the decoding process.

The adaptively decoding scheme is implemented by truncating the distributions of decoding efforts at a point where a tradeoff between performance and energy is achieved. For example, if the number of check nodes in error is greater than a threshold, then the number of decoding iteration is fixed to a pre-determined number. In similar fashion, several threshold levels derived from the empirical data are generated and a set of respective decoding iteration numbers affixed. The policy is described in the following pseudo-codes:

```
if(Num_Check_Err>=Check_Err_Theshold1)
    Num_Dec_Iteration = num1;
elseif(Num_Check_Err>=Check_Err_Theshold2)
    Num_Dec_Iteration = num2;
elseif(Num_Check_Err>=Check_Err_Theshold3)
    Num_Dec_Iteration = num3;
else
    Num_Dec_Iteration = num4;
end if
```

As aforementioned, the $E + 2\delta$ point is found to be a reasonable choice based on the number of check-errors threshold selections for this particular sample LDPC code. Other sets of numbers can also be selected depending on different power and coding-performance trade-offs. The threshold values $Check\_Err\_Theshold(i)$ $(i = 1, 2, 3)$ and numbers of decoding iterations $num(j)$ for $j = 1, \cdots, 4$ are chosen through simulation.

C. Implementation and results

The above policy could be implemented with low hardware complexity. It has been reported [24] that for deep sub-micro VLSI technology, the leakage power is becoming so large that the best solution in terms of power is maintaining the highest performance for the longest time as possible and then turning the circuit into sleep mode. In the case of LDPC decoder, however, this is not feasible because of the real-time constrain, constantly incoming data, as well as power overhead of turning off and on the circuit. The clock frequency is determined by the constant decoding time, for instance, the clock frequency for frames required 20 iterations is twice as high as those requiring only 10 iterations. Operating at low clock frequency, the voltage supply could also be lowered correspondingly.

Diagram of the adaptively decoding controller is presented in Figure 10(a). Number of check errors in incoming data frame is calculated as cH^T, where c is the code word based on hard decision of the incoming log-likelihood channel data. Since cH^T is also implemented inside the decoder for decoding termination decision and it can be reused in the controller. Therefore this unit does not impose any additional hardware resource or power consumption. Because level of the voltage supply can not be changed instantly, frame buffer is required for incoming channel data, a frequency-selection buffer that stores decoding iteration information for corresponding data frames. The buffer size K is determined by time response of voltage supply V_{ddl} as well as expected decoding time, *i.e.* throughput. As presented later, buffer size of 2 frames ($K = 2$) is needed typically. The overhead is buffer of size 1 frame since a buffer size of 1 frame is intrinsic for the decoder. cH^T of the incoming data frame F_i, is used to predict the number of decoding cycles, hence the decoding frequency can be determined. Decoding frequency of frames F_{i-1} to F_{i-K+1} also participate

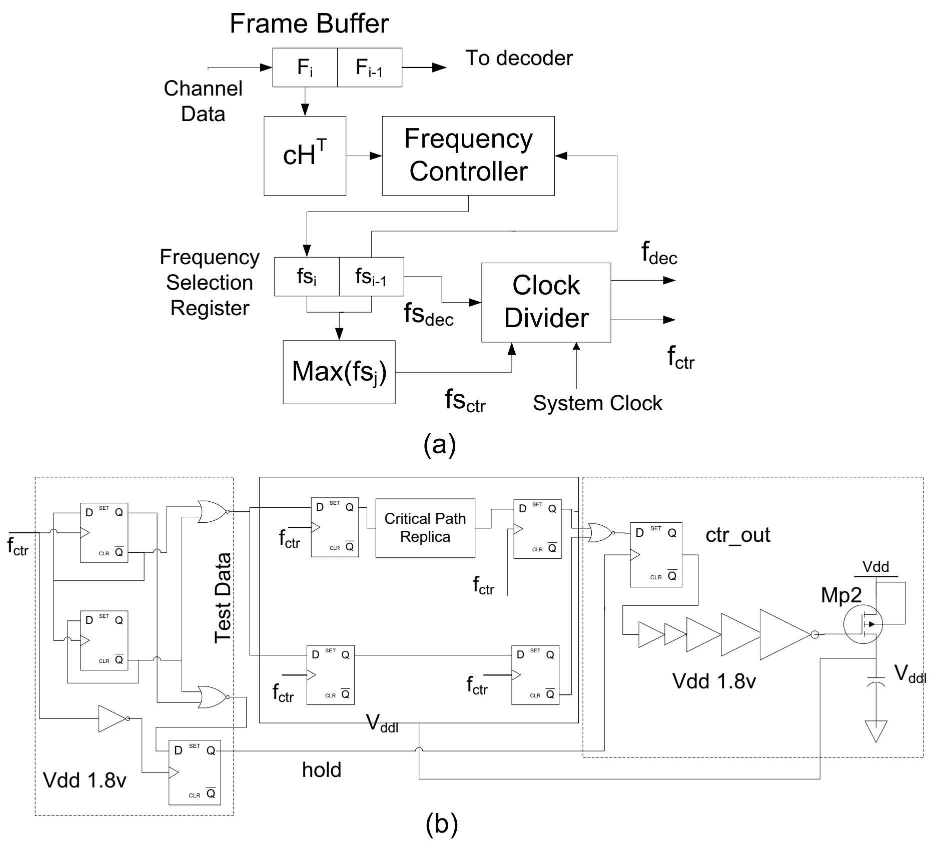


Fig. 10. Block diagram (a) and buck converter (b) of the proposed DVFS controller.

in the process because of the finite voltage response time. The clock divider divides the fast system clock into slower clock signals according to the frequency selection register. Clock divider is preferred over other designs such as phase-loop locker (PLL) in [11], because it provides reasonable frequency resolution for the decoding policy and capability to change immediately. f_{dec} clocks the decoder for current frame, and f_{ctr} is sent to the voltage scaling controller. f_{ctr} is conservatively generated as the fastest clock such that the voltage supply will be within safe region for operation.

A variety of VLSI implementations of the voltage-scaling controller have been reported in the literatures. Firstly, this paper adapted the design in [10] of the buck converter, because it is of reasonable complexity. Other control schemes can also be used. Secondly, a design based on Min-Sum algorithm in [3] is used for the decoder.

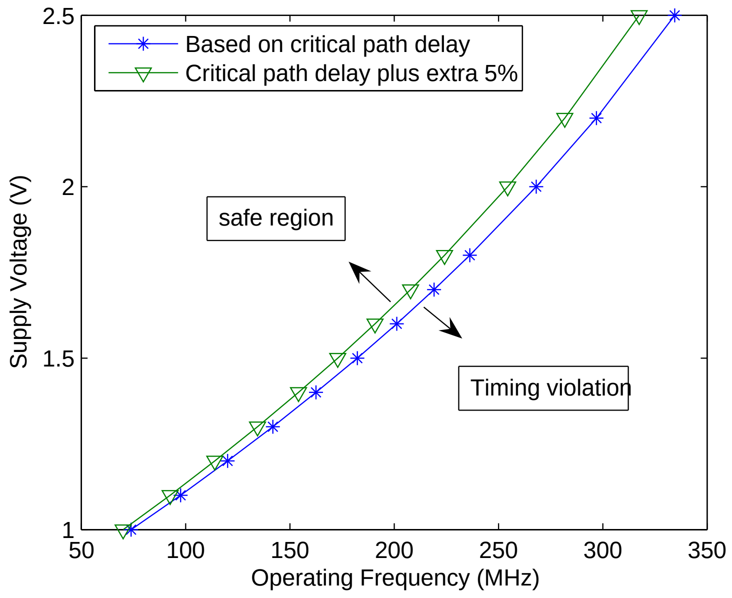


Fig. 11. Voltage supply requirement for different operating frequency

The critical path of the decoder is extracted and replicated for the voltage controller. Load capacitor of the voltage controller should be large enough to maintain a steady voltage level in presence of sudden change in the output current. The capacitor is chosen to be $0.65\mu C$. The power transistor *Mp2* is $400\mu m$ in width, which is driven by five stages of buffer, with a scaling up factor of 4 [10], considering the minimum power consumption. Figure 10(b) shows the diagram of the circuitry.

The design of voltage-scaling controller has been simulated using $TSMC0.13\mu m$ technology. With $1.5V$ voltage supply, the decoder can be clocked as fast as $175MHz$, as shown in Figure 11. Extra 5% timing margin has been added to the critical path replica in the controller to accommodate variations. Figure 12 demonstrates voltage response of the converter. The buck converter can scale up the output voltage level to a maximum of $60mV/\mu s$. Results presented in [10] align with our simulation results. Assuming a $500MHz$ system clock, it can be divided into $167MHz$, $125MHz$,

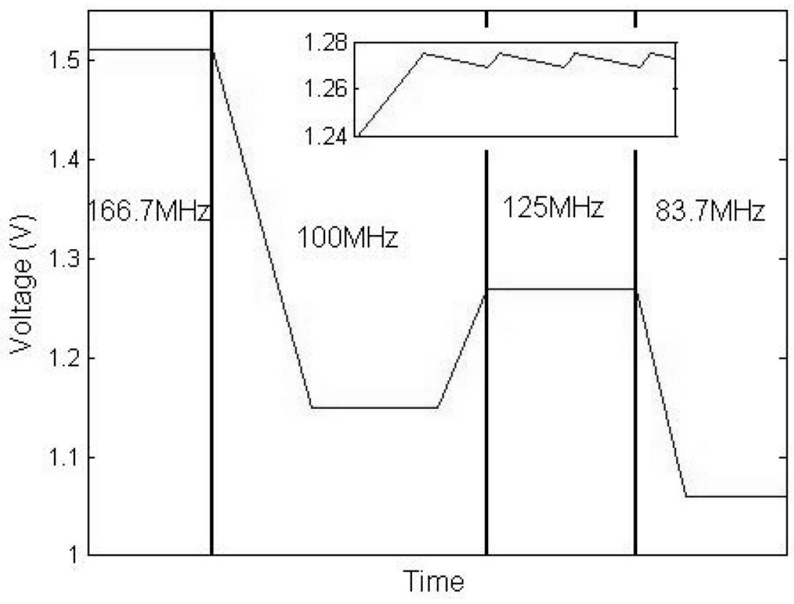
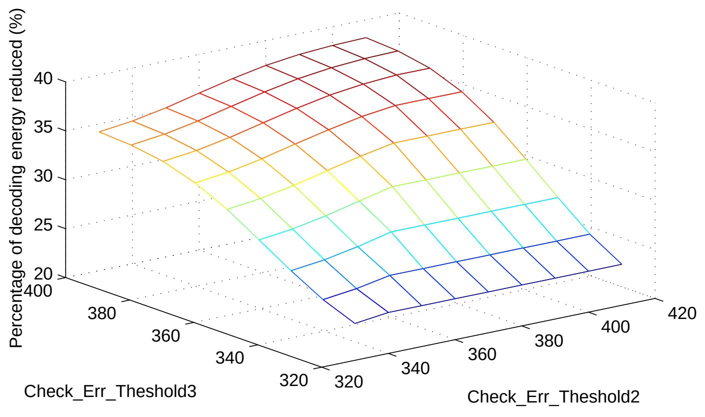


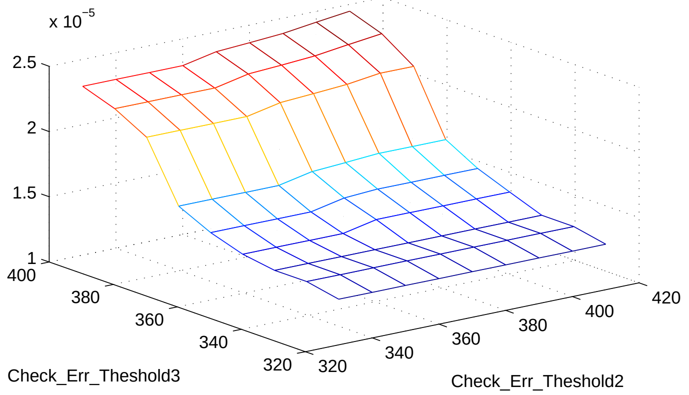
Fig. 12. Simulated voltage response V_{ddl}

$100MHz$ and $84MHz$ for decoder. The voltage supply varies from $1.45V$ to $1.05V$ within this frequency range. It takes about $10\mu s$ to scale the voltage up by $0.4V$. In the case of 2048 bits code-length, the voltage controller is able to respond to as much as $200Mbps$ decoder throughput with a frame-buffer size of 2. Current through the PMOS power transistor constitutes the majority of power overhead of the controller. It is simulated to be in the order of $10mW$, which is small comparing to the total power dissipation of the decoder, around $200mW$.

Coding gain and energy saving is a multi-dimensional function of threshold values and SNR. The effect of threshold values is first explored. In the simulation, SNR of the channel varies in a wide range, from 2.2 to 3.0, and the maximum number of decoding iterations is fixed at 20. The resulting average bit-error rate (BER) is 1.5×10^{-5}, and average frame error rate(FER) is 6.4×10^{-4}. The numbers of maximum decoding iterations are set to be 24, 18, 14 and 12, based on analysis shown in Figure 6 and Figure 7. The numbers are chosen based on complexity of design and consideration



(a)



(b)

Fig. 13. Saving in energy (a) and coding performance (b) based on different thresholds of check errors.

of performance requirement.

As the frequency selections are $167MHz$, $125MHz$, $100MHz$ and $84MHz$, the above numbers of decoding iterations yield constant-time decoding. Other sets of choices are also possible for different power-performance trade-offs. In the power estimation, the relative weights of dynamic power, which is proportional to the square of power supply, and leakage power which is proportional to power supply, are considered to be 80% and 20%, respectively.

Figure 13 shows the resulted power reduction as well as coding performance corresponding to different choices of check-error thresholds. The value $checkErr\_threshold1$ is always set to be 420 empirically in this paper. Average BER is 1.6×10^{-5} when the threshold values $checkErr\_threshold2$ and $checkErr\_threshold3$ described in section B are 380 and 350 respectively. There is 35% energy saving. Further decreasing the threshold values will not improve coding performance much, while the saving in number of decoding iterations decreases rapidly.

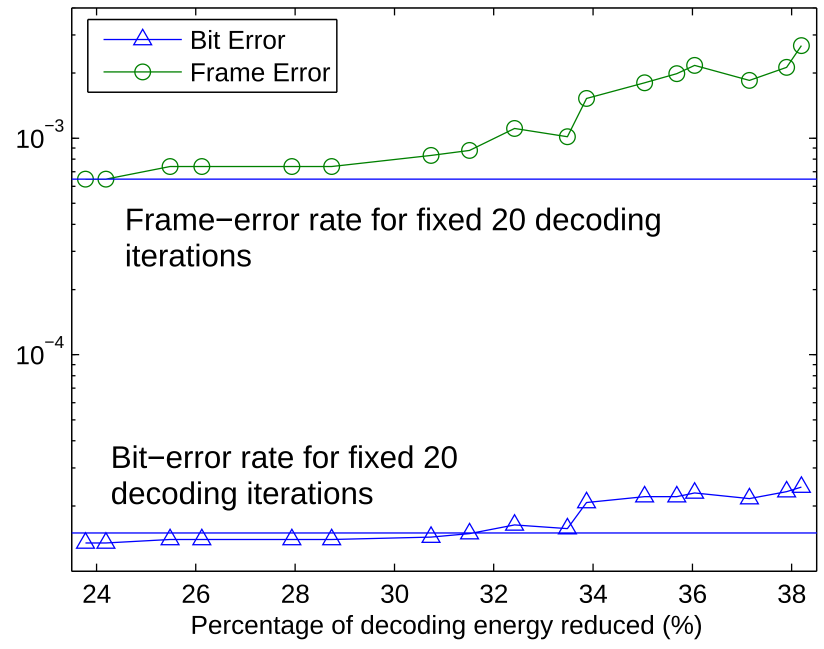


Fig. 14. Coding performance at different level of energy saving for different threshold value selection.

While the number of maximum decoding iterations is set to be 20 for all data frames in the conventional decoding scenario, the proposed decoding scheme discriminately varies the number of iterations for each frame. The relationship between coding performance and power saving is presented in Figure 14. It is clearly seen that up to 30% power is saved without bit-error degradation and minimum frame-error rate loss. Additional saving in energy is achieved for high SNR, as shown in Figure 15. The increased saving is due to the fact that the small probability of a bit being corrupted by channel noise when SNR is high. Therefore, minimum number of decoding iteration will correct all errors and power supply V_{ddl} mostly stays at low level, which results in very low power dissipation.

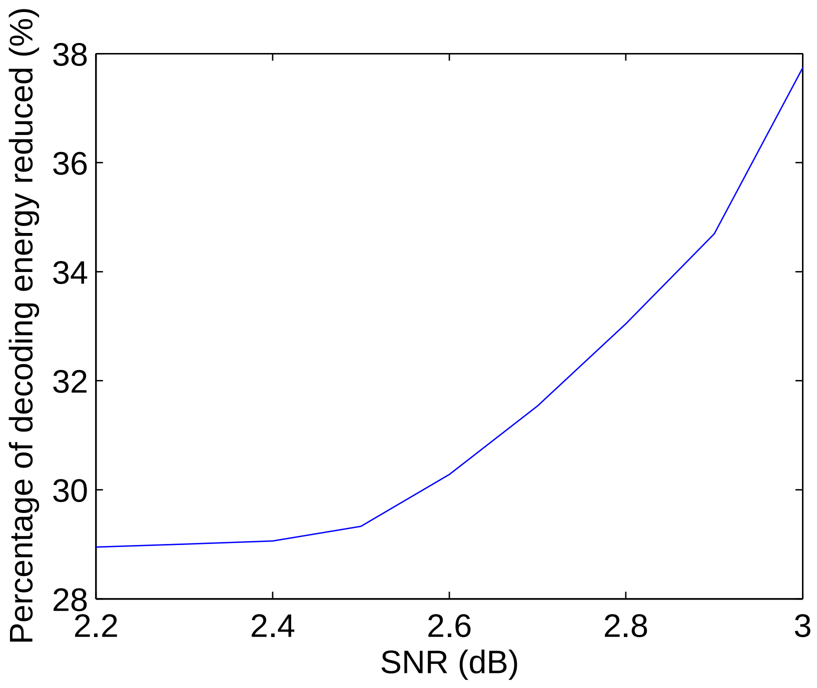


Fig. 15. Energy saving for different SNR levels.

In summary, the presented adaptively decoding scheme will achieve significant saving in decoding energy. Based on different choices of control parameters and channel conditions, different optimization objectives in terms of coding performance and power are achievable.

CHAPTER V

SPECULATIVE DECODING FOR NON-FADING AWGN CHANNEL

This chapter presents the adaptive decoder architecture for non-fading AWGN channel. Decoding effort is estimated before starting of the decoding process of every frame in the proposed decoding scheme for block-fading channel. Decoding energy is saved via truncating the distribution of decoding effort and applying DVFS technique. The aforementioned scheme is independent of any specific LDPC decoding architecture. LDPC decoding in non-fading channel, however, differs from that of fading channel, thus a variation of the aforementioned decoding schemes is proposed.

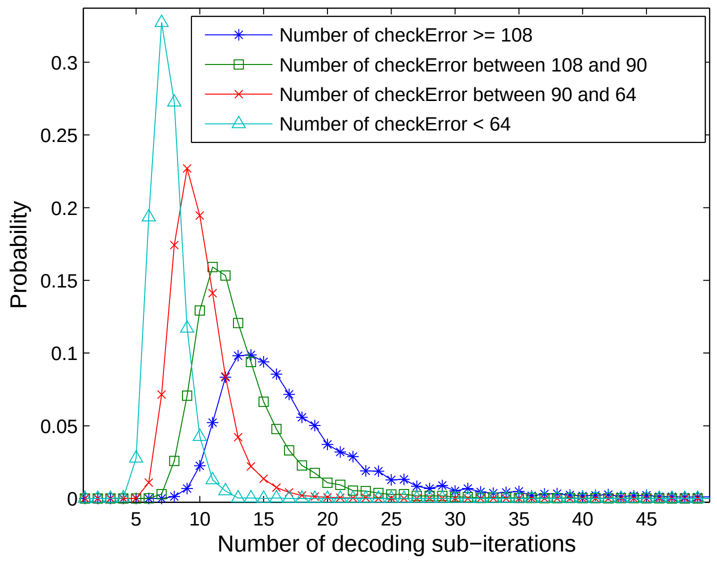


Fig. 16. Distribution of decoding iteration for different number of check errors after three sub-iterations with channel Eb/No at 3.7dB.

While number of check errors for the coming channel data still partially represents the degree of noise impairment, and indicate effort required for decoding, the correlation between number of check errors and number of decoding iteration though,

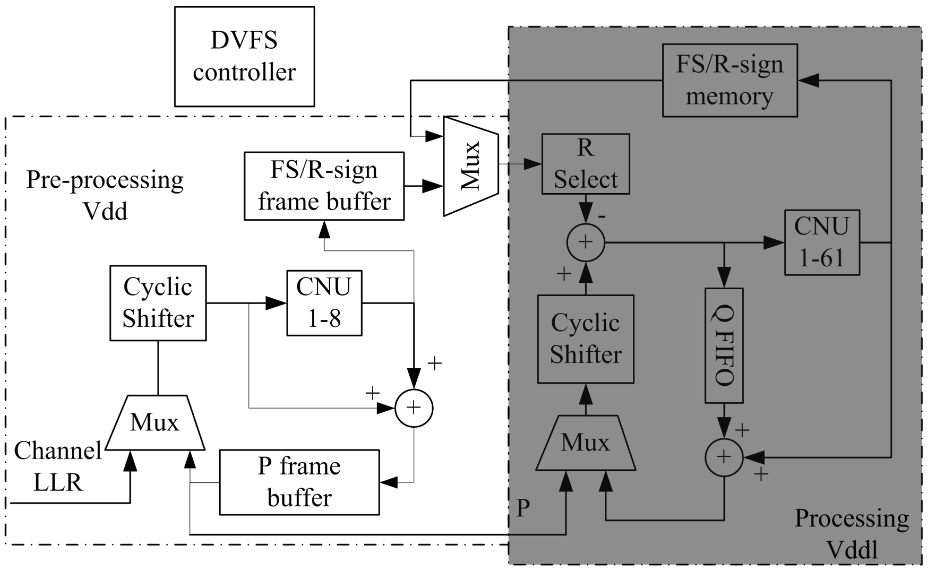


Fig. 17. Decoding architecture for non-fading AWGN channel.

becomes much less. This correlation can be recovered after several initial decoding iterations. Analysis in this chapter has been carried out based on a sample array LDPC code, with parameters $d_v = 5$, $d_c = 25$, and $p = 67$. And TDMP based on offset min-sum (OMS) algorithm is used for decoding. Similar analysis can be extended to other array LDPC codes. It is shown in Figure 16 that after three sub-iterations, the number of check errors remaining is correlated with the total number of decoding iterations. As such, similar approach as the adaptively decoding scheme in Chapter IV can be applied. However, special treatment is needed because prediction take place in the middle of the decoding process, instead of before decoding starts.

An efficient adaptively decoding architecture is proposed based on the decoding architecture presented in Chapter III. Block diagram of the architecture is shown in Figure 17. It can be divided into three main blocks. First is the pre-processing block, where the first three sub-iterations are carried out with little hardware overhead. This block lies in the regular power supply Vdd voltage domain. Second is the processing block. The main decoding process is carried out in this block, which is in the variable power supply Vddl domain. Third is the DVFS controller. It is the same as in Figure 10, except that now the cH^T value comes from after three sub-iterations instead of channel data.

Recall that for the architecture in Figure 3, Q^{shift} equals channel LLR or P message subtracting R_{old}, in which R_{old} is check node message from last iteration. Since both FS register and sign FIFO are reset to zero when decoding starts, and remains zero during the first decoding iteration, the subtraction is not necessary for the pre-processing block. As such, channel data will go directly to the CNU array through the shifter. Since only three sub-iterations will take place in the pre-processing block, less CNU units can be instantiated, to match its throughput with the processing block. Achieving $p \times p$ cyclic shifts by $M * M$ shifter, where $M < p$, is introduced in Chapter III. P messages, instead of Q messages are stored. P messages and check node messages are also passed from pre-processing block to processing block through the frame buffer. The size of frame buffer is to make sure that there will be sufficient time for voltage adjustment in the variable voltage domain. Study from Chapter IV has already shown that size of two frame is sufficient. Considering the memory-saving nature of the decoder architecture, this overhead is very small. Note that other than the frame buffer, the preprocessing block should not be considered as hardware overhead of the adaptively decoding architecture. This is because that it contribute directly to the final throughput.

When check node messages and P messages are ready from the pre-processing block, the process block start operation as described previously in Chapter III. Operation of the two blocks is pipelined, so no idle state will be introduced because of this adaptively decoding architecture.

The proposed adaptively decoding architecture save significant power and energy

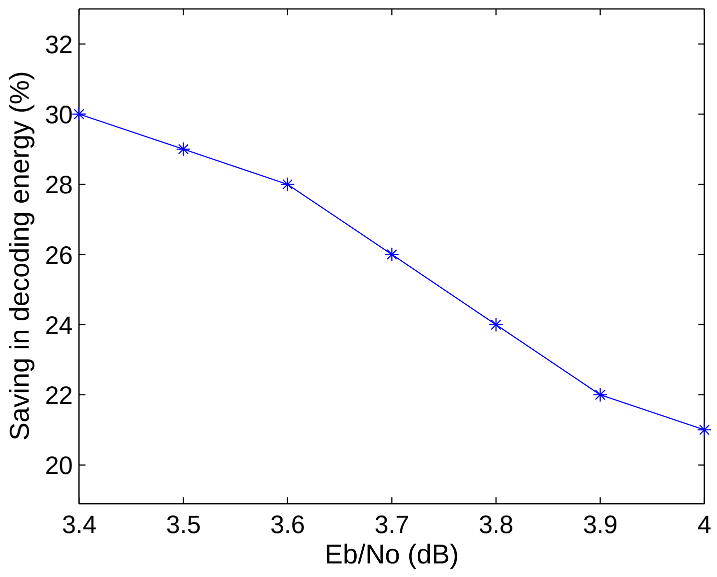


Fig. 18. Saving in decoding energy at different SNR levels.

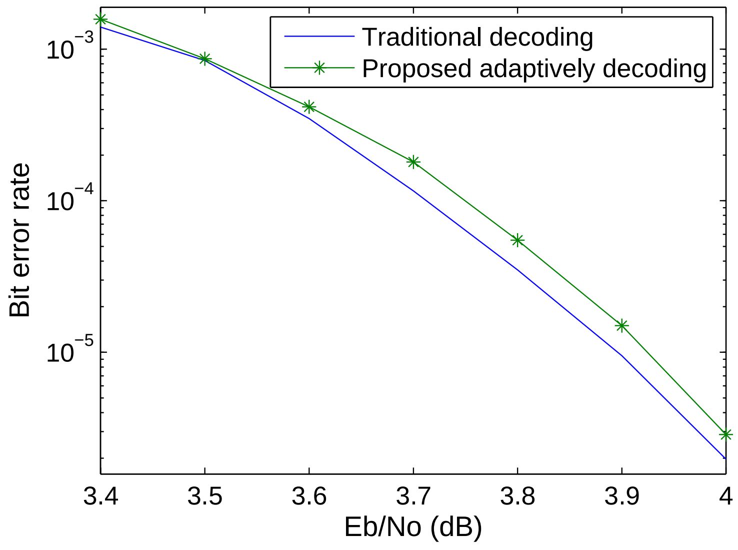


Fig. 19. BER comparison for traditional decoding and proposed speculative decoding.

consumption. Simulation results is shown in Figure 18. At relatively low SNR level, with Eb/No at 3.4dB for example, up to 30% percent saving in energy consumption is achieved. Energy consumption is lowered at hight SNR level. At Eb/No of 4.0dB, the saving, though decrease to 21%, is still significant. This decreasing occurs because the reduced average number of decoding iterations. Since the first three sub-iteration takes place in the fixed V_{dd} voltage domain, as SNR increases, decoding energy consumption of the pre-processing block rises relative to the processing unit. Figure 19 shows that the proposed adaptively decoding architecture performs closely with conventional decoding schemes in terms of code performance. As seen, less than 0.05dB BER degradation is observed.

CHAPTER VI

CONCLUSION

A LDPC decoder scheme suitable for portable device in real-time mobile communication is presented. First, modeling and FPGA implementation results for a memory and power efficient layered decoding LDPC architecture is discussed. Then, adaptively LDPC decoding is presented. For block-fading channels, incoming channel data is processed before decoding to determine the decoding process. While larger number of decoding iterations is used for critical data frames to maintain high coding performance, smaller number of iterations, lower frequency, and hence lower power supply are used for data frames less severely damaged by noise in order to save power. Power overhead of the adaptively decoding control unit mainly stems from the power transistor, and it is found to be small compared with power saved. Up to 30% power saving in decoding process is achieved without performance degradation. For non-fading AWGN channels, decoding effort is estimated after three sub-iterations, similar to that for block-fading channel. An efficient VLSI architecture is proposed and analyzed. 21% to 30% percent of decoding energy can be save with BER degradation of less than 0.05dB in Eb/No.

REFERENCES

[1] R. Gallager, “Low-density parity-check codes,” *IRE Trans. on Inform. Theory*, vol. 8, pp. 21–28, Jan. 1962.

[2] D. MacKay and R. Neal, “Near shannon limit performance of low density parity check code,” *Elec. Letters*, vol. 332, pp. 1645 – 1646, Aug. 1996.

[3] K. Gunnam, G. Choi, W. Wang, E. Kim, and M. Yeary, “Decoding of array ldpc codes using on-the-fly computation,” *Fortieth Asilomar Conference on Signals, Systems and Computers, 2006. ACSSC 0́6*, pp. 1192 – 1199, Oct. 2006.

[4] M. Mansour and N. Shanbhag, “High-throughput ldpc decoders,” *IEEE Trans. on Very Large Scale Integrated (VLSI) System*, vol. 11, no. 6, pp. 976 – 996, Dec. 2003.

[5] M. Mansour and N. Shanbhag, “A 640-mb/s 2048-bit programmable ldpc decoder chip,” *IEEE J. of Solid-State Circuits*, vol. 4, no. 3, pp. 684 – 698, Mar. 2006.

[6] F. Kienle and N. When, “Low complexity stopping criterion for ldpc code decoders,” *IEEE 61st Vehicular Technology Conference*, vol. 1, pp. 606 – 609, May 2005.

[7] G. Glikiotis and V. Paliouras, “A low-power termination criterion for iterative ldpc code decoder,” *IEEE Workshop on Signal Processing Systems Design and Implementation*, pp. 122 – 127, Nov. 2005.

[8] B. Shim, S. Sridhara, and N. Shanbhag, “Reliable low-power digital signal processing via reduced precision redundancy,” *IEEE Tran. On VLSI systems*, vol. 1, no. 5, pp. 497 – 510, May 2004.

[9] L. H. Ozarow, S. Shamai, and A. D.Wyner, "Information theoretic considerations for cellular mobile radio," *IEEE Trans. on Vehicle Technology*, vol. 43, no. 2, pp. 359 – 378, May 1994.

[10] T. Kuroda, K. Suzuki, S. Mita, T. Fujita, F. Yamane, and *et al.*, "Variable supply-voltage scheme for low-power high-speed cmos digital design," *IEEE J. of Solid-State Circuit*, vol. 33, no. 3, pp. 454 – 462, Mar. 1998.

[11] P. Macken, M. Degrauwe, M. van Paemel, and H. Oguey, "A voltage reduction technique for digital systems," *1990 IEEE International Solid-State Circuits Conference, Digest of Technical Papers. 37th ISSCC.*, pp. 238 – 239, Feb. 1990.

[12] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power cmos digital design," *IEEE J. of Solid-state Circuits*, vol. 27, pp. 473 – 484, 1992.

[13] A. Dholakia and S. Olcer, "Rate-compatible low-density parity-check codes for digital subscriber lines," *2004 IEEE International Conference on Communications*, vol. 1, pp. 415 – 419, Jun. 2004.

[14] T. J. Richarson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. on Information Theory*, vol. 47, pp. 619 – 637, Feb. 2001.

[15] F. Kschischang, B. Frey, and H. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. on Information Theory*, vol. 47, pp. 498 – 519, Feb. 2001.

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

[17] K. Gunnam, G. Choi W. Wang, and M.B. Yeary, "Multi-rate layered decoder architecture for block ldpc codes of the ieee 802.11n wireless standard," *IEEE International Symposium on Circuits and Systems*, pp. 1645–1648, May 2007.

[18] M. Pedram and J. Rabaey, *Power Aware Design Methodologies*, Norwell, MA: Kluwer Academic Publishers, 2002.

[19] K.Gunnam and G. Choi, "A low power architecture for min-sum decoding of ldpc codes," *TAMU, Electrical and Computer Engineering Technical Report TAMU-ECE-2006-02. Available at http://dropzone.tamu.edu/techpubs*, May 2006.

[20] K. Gunnam, G. Choi, and M. B. Yeary, "A parallel layered decoder architecture for array ldpc codes," *in Proceedings of the 20th International Conference on VLSI Design held jointly with 6th International Conference*, pp. 738–743, Jan. 2007.

[21] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of ldpc codes," *IEEE Workshop on Signal Processing Systems. SIPS 2004*, pp. 107–112, Oct. 2004.

[22] M. Karkooti and J. Cavallaro, "Semi-parallel reconfigurable architectures for real-time ldpc decoding," *IEEE Workshop on Signal Processing Systems*, pp. 107–112, Apr. 2004.

[23] T. Brack, F. Kienle, and N. Wehn, "Disclosing the ldpc code decoder design space," *Design Automation and Test in Europe (DATE) Conference*, pp. 200–205, Mar. 2006.

[24] R. Jejurikar, C. Pereira, and R. Gupta, "Leakage aware dynamic voltage scaling

for real-time embedded systems," *Design Automation Conference 2004*, pp. 275–280, Jun. 2004.

VITA

Weihuang Wang received his M.S. degree in the Department of Electrical and Computer Engineering at Texas A&M University in December 2007. He obtained his B.S. degree in Microelectronics from Peking University, Beijing, China in July 2005. His research interests include VLSI design of communication circuits and image processing, signal integrity analysis of high speed VLSI. He can be contacted at:

Department of Electrical and Computer Engineering
c/o Dr. Gwan Choi
Texas A&M University, M.S. 3259
College Station, TX, 77843