Minimum-Energy LDPC Decoder for Real-Time Mobile Application

Weihuang Wang, Gwan Choi
Department of Electrical and Computer Engineering, Texas A&M University, TX 77840
Email: {whwang, gchoi}@ece.tamu.edu

***Abstract*— This paper presents a low-power real-time decoder that provides constant-time processing of each frame using dynamic voltage and frequency scaling. The design uses known capacity-approaching low-density parity-check(LDPC) code to contain data over fading channels. Real-time applications require guaranteed data rates. While conventional fixed-number of decoding-iteration schemes are not energy efficient for mobile devices, the proposed heuristic scheme 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. Energy use is then reduced appropriately by adjusting power supply voltage to minimum necessary for the given frequency. The resulting design provides a judicious trade-off between power consumption and error level.**

I. Introduction

There exists a multitude of applications that require real-time content delivery for wireless portable devices, especially in multi-media domain. These applications require efficient coding scheme, and one example is low-density parity-check(LDPC) codes. These are special cases of error correcting codes originally proposed by Gallager[1] in 1960's and rediscovered in late 1990's[2]. LDPC codes have recently gained a significant attention because of their near Shannon-limit performance and high throughput, and there have been a number of efficient implementations of LDPC decoders [3]-[4]. 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 widely used message-passing algorithms exchange information between the bit nodes and check nodes (parity checks constraining the bits) in an iterative fashion. In practice, the LDPC decoder is typically set to run for data convergence until a prescribed maximum number of iterations (e.g. 20) depending on the code rate. There have been researches on early termination of the frame that can not be decoded even if the maximum iterations are applied[5]-[6]. In both of papers, early termination of the iterative process is determined by checking the messages during the decoding. Their attempts were to dynamically switch off the hardware when no additional iteration will amount to improvement in decoding performance. Such architectures yield unpredictable frame-completion time which makes interfacing with the application modules rather difficult.

Real-time applications, however, 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 minimum energy consumption for block-fading channel[7], while guaranteeing quality of service (QoS) for time-sensitive data. In the proposed scheme, 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[8]-[9]. System power dissipation is proportional to occurrence of activity and quadratic voltage supply[10]. 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 the paper is organized as following: In section II, the background of LDPC code together with decoding algorithm and DVFS are briefly described. Section III gives the analysis of channel data to adaptively adjust the decoding stages. The low-power real-time decoding policy is generated empirically. Coding performance, as well as savings in decoding iterations of such policy is shown. In section IV, ASIC design of the voltage and frequency scaling controller is presented. Section V gives conclusion and future work.

II. Background

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

A. LDPCCodes

LDPC codes are defined by a sparse parity check matrix $H = [H_{mn}]$ 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 n is equal to the number of columns in the H matrix. Suppose that the number of data bits before the channel encoding is k, then the number of rows of this H matrix is $m = n - k$. Rate

978-3-9810801-2-4/DATE07 © 2007 EDAA

of this code is defined as $k/n = 1 - d_v/d_c$. 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 d_c should be no less than 3 and 6, respectively, for acceptable coding performance. Another type of LDPC code is irregular codes, in which the number of 1's in each row and column is not constant. Such codes, though generally produce high coding performance, is more complex for implementation. LDPC codes can also be represented by a bipartite graph with two sets of nodes: check nodes and variable nodes. The check nodes correspond to parity check constraints, i.e. rows of the H matrix, while the other set of nodes correspond to the data symbols, i.e. the columns of the parity check matrix.

The decoding of LDPC codes is based on the iterative message-passing algorithm, also known as belief-propagation algorithm[11]-[12]. The algorithm consists of two phases, a check-node processing and variable-node processing. In the check-node processing, each row of the parity matrix is checked to verify that parity constraints are satisfied. In the second phase, the variable-node probability is 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 (BP) based algorithm (also called Min-Sum algorithm)[13]. While significantly reducing the decoding complexity in implementation, the Min-Sum algorithm degrades the coding performance. The improved BP based algorithm, Normalized-Min-Sum and Offset-Min-Sum [13] eliminates this performance degradation. Specifically, this paper is based on the VLSI design of LDPC decoder using Offset-Min-Sum algorithm[3].

B. Circuit power estimation and reduction

There are three major sources of power dissipation in CMOS circuit[10]:

$$\begin{aligned} P_{total} &= P_{switching} + P_{SC} + P_{leakage} \\ &= \alpha C_L \Delta V V_{dd} f_{clk} + I_{SC} V_{dd} + I_{leakage} V_{dd} \end{aligned} \quad (1)$$

$P_{switching}$ 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 α 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 [14]. $P_{leakage}$ 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 (2):

$$\tau = \frac{1}{f_{clk}} = \frac{C_L V_{dd}}{I_{dsat}} \propto \frac{V_{dd}}{(V_{dd} - V_{th})^{1.3}} \quad (2)$$

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 constant for the same workload if the power supply is not changed. Dynamic voltage and frequency scaling is an effective method to address this energy consumption problem, especially under wide variations in workload. A number of DVFS designs have been presented in the literatures[8].

III. Proposed low-power real-time decoding

A. Policy

The key observation made for adaptively decoding is that for most of the data frames, the decoding process is finished before the maximum number of decoding iterations. The decoder then stays idle waiting for next frame, which comes at constant time interval. Significant energy saving can be achieved by lowering the decoder performance level when possible as discussed in section II. Such performance adjustment is feasible because severity of noise corruption of channel data, which has direct influence on number of decoding iterations needed, can be estimated in advance from the decoding process itself.

Based on statistical analysis of the received data from channel, we propose a heuristic low-power real-time decoding policy of LDPC codes. In the proposed scenario, the maximum number of decoding iteration for each frame is dynamically adjusted. The maximum number of decoding iteration is set to be close to optimum in terms of energy and coding performance, without violating the real-time constraint. In this paper, simulation is carried out based on randomly constructed $(3, 6)$ rate $1/2$ code with block length of 2048 over a block fading channel, assuming Gaussian noise and code length equal to fading block length. Our approach can be extended for other LDPC decoder designs.

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 respect to the H matrix, the numbers of checks in error are analyzed statistically. Fig. 2 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. Fig. 3 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. Fig. 4 shows a

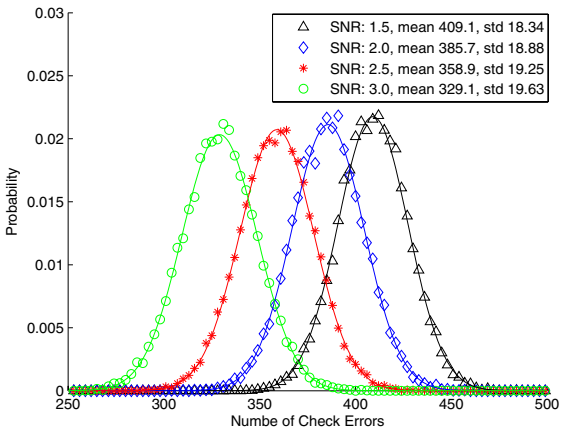

Fig. 1. Probability density function of number of check error. (std: standard deviation)

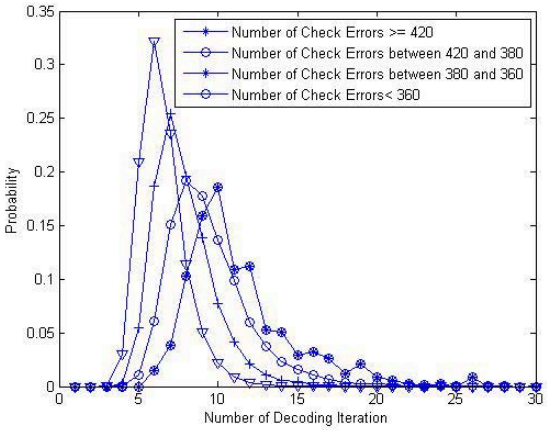

Fig. 2. Distribution of decoding iteration for different number of check errors

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.

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. The policy is described in the following 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
if(Num_Dec_Iteration<1.5*Aver_Iteration)
    Num_Dec_Iteration = 1.5*Aver_Iteration;
end if
```

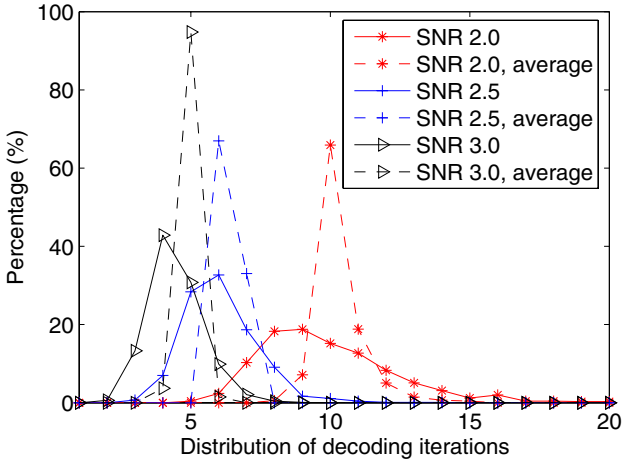

Fig. 3. Distribution of decoding iterations at different SNR level for each frame as well as average decoding iterations of every 10 frames

It should be noted that $Num\_Dec\_Iteration$ is the predicted maximum decoding iterations, and it is used in the decoding termination decision. The threshold values and numbers of decoding iterations are chosen during simulation.

B. Design

The above policy can be implemented with low hardware complexity. Even though it has been reported [15] that for modern VLSI technology, the leakage power is becoming so significant that the best solution for managing power is maintaining the highest performance as long 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 constraints, constantly incoming data, as well as power overhead associated turning off and on the circuit. The clock frequency is determined by the constant decoding time, for instance, the clock frequency for frames requiring 20 iterations is twice as high as those requiring only 10 iterations. Operating at low clock frequency, the voltage supply can be lowered correspondingly.

Diagram of the adaptively decoding controller is presented in fig. 4(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

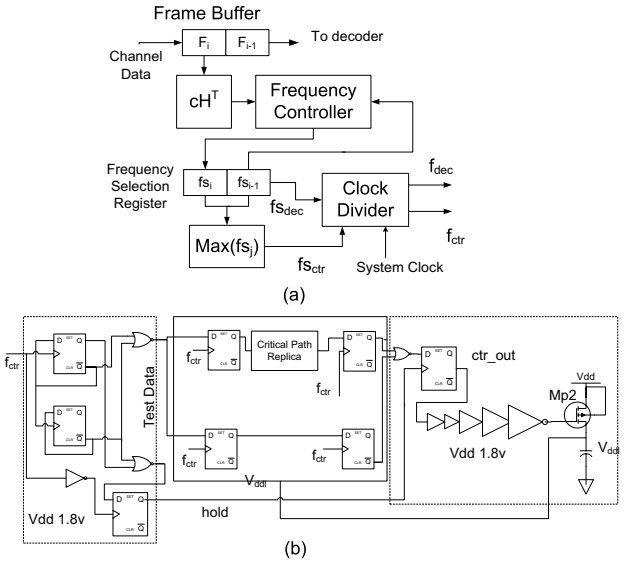

Fig. 4. Block diagram of controller and buck converter

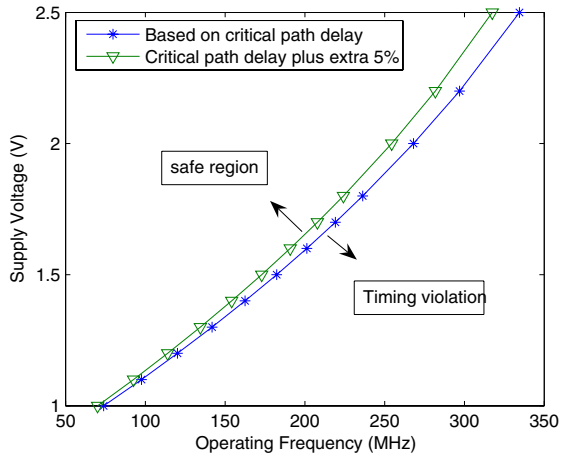

Fig. 5. Voltage supply requirement for different operating frequency

voltage supply V_{ddl} as well as expected decoding time, *i.e.* throughput. As presented later in section IV, 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 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 [9], 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 [8] 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. 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[8], considering the minimum power consumption. Fig. 4(b) shows the diagram of the circuitry.

IV. Result

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 fig. III-B. Extra 5% timing margin has been added to the critical path replica in the controller to accommodate variations. Fig. IV 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 [8] align with our simulation results. Assuming a $500MHz$ system clock, it can be divided into $167MHz$, $125MHz$, $100MHz$ 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 $200mW$.

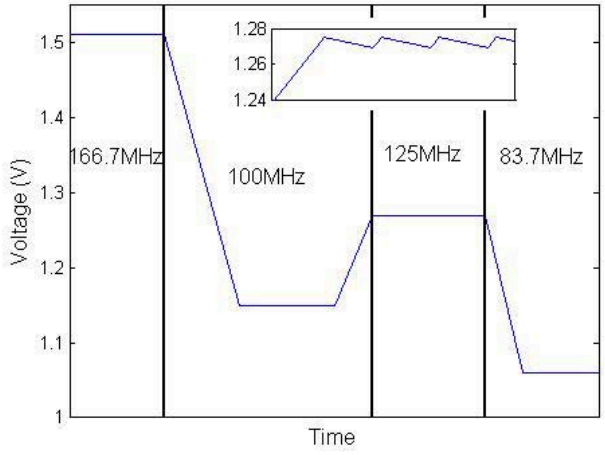

Fig. 6. Simulated voltage response V_{ddl}

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 bit-error rate (BER) is 1.5×10^{-5}, and frame error rate(FER)

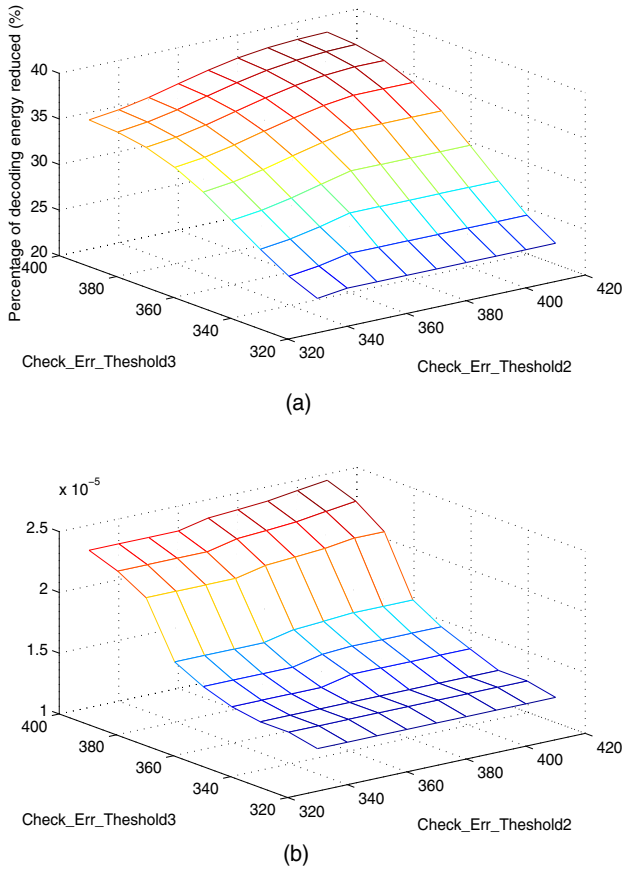

Fig. 7. (a) Saving in energy based on different thresholds of check errors. (b) Coding performance based on different thresholds of check errors.

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 fig. 2 and fig. 3. The numbers are chosen based on complexity of design and consideration 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 70% and 30%, respectively.

Fig. IV shows the resulted coding performance as well as power reduction corresponding to different choices of check-error thresholds. The value $checkErr\_threshold1$ is always set to be 420 empirically in this paper. BER is when the threshold values $checkErr\_threshold2$ and $checkErr\_threshold3$ described in section III 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.

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

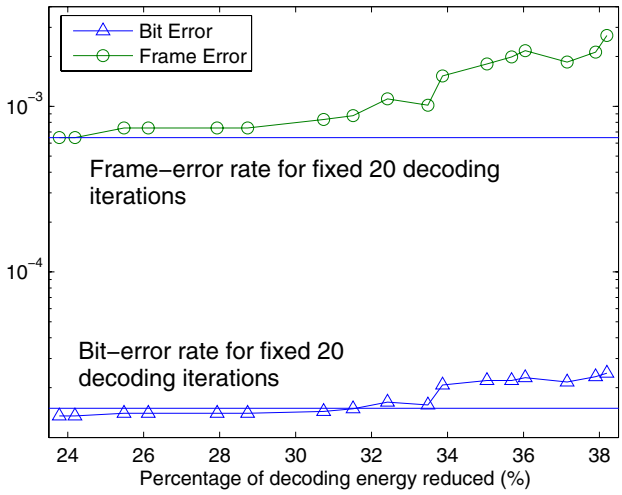

Fig. 8. Coding performance at different level of energy saving for different threshold value selection.

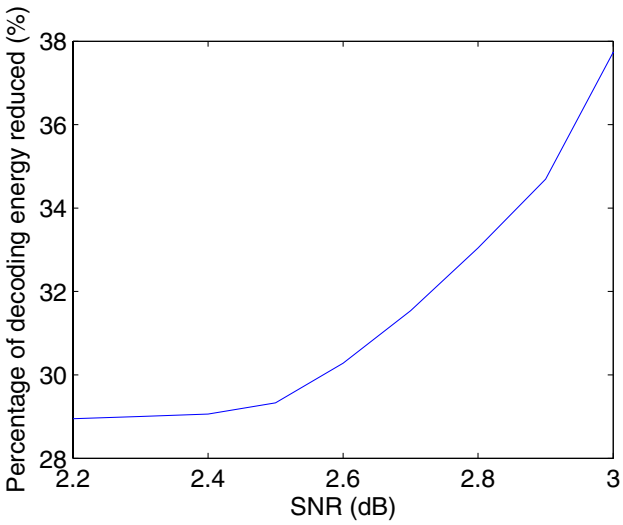

Fig. 9. Energy saving for different SNR levels.

in fig. 8. 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 fig. 9. 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.

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.

V. Conclusion

A LDPC decoder scheme suitable for portable device in real-time mobile communication is presented. 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.

REFERENCES

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

[2] D. MacKay, R. Neal, "Near Shannon Limit Performance of Low Density Parity Check codes," *Elec. Letters*, vol. 32, pp. 1645-6, Aug 1996.

[3] K. Gunnam, W. Wang, E. Kim, G. Choi and M. Yeary, "Decoding of Array LDPC Codes using On-The-Fly Computation," Accepted for *40th Asilomar Conference on Signals, Systems and Computers, October 2006*. Pre-print available: http://dropzone.tamu.edu/techpubs/2006/TAMU-ECE-2006-05.pdf

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

[5] F. Kienle, N. When, "Low Complexity Stopping Criterion for LDPC Code Decoders," *VTC, 2005-Spring*. vol. 1, pp. 696-609, 2005.

[6] G. Glikiotis, 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.

[7] L. Ozarow, S. Shamai, and A. Wyner, "Information Theoretic Considerations for Cellular Mobile Radio," *Vehicle Technology, IEEE Trans. on*, vol. 43, no. 2, pp. 359-378, May 1994.

[8] T. Kuroda, *et al.* "Variable Supply-Voltage Scheme for Low-Power High-Speed CMOS Digital Design," *Solid-State Circuit, IEEE Journal of*, Vol. 33, No. 3, pp. 454-462, March 1998.

[9] P. Macken, M. Degrauwe, M. van Paemel, and H. Oguey, "A voltage reduction technique for digital systems," *ISSCC Dig. Tech. Papers*, pp. 238-239, Feb. 1990.

[10] A. Chandrakasan, *et al.*, "Low-power CMOS Digital Design," *Solid-state Circuits, IEEE Journal of*, Vo1.27, Issue.4, pp. 473-484, 1992.

[11] T. Richarson, M. Shokrollahi, and R. Urbanke, "Design of Capacity-approaching Irregular Low-Density Parity-Check Codes," *Information Theory, IEEE Trans. on*, vol. 47, pp. 619-637, Feb. 2001.

[12] F. Kschischang, B. Frey, and H. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *Inform. Theory, IEEE Trans. on*, vol. 47, pp. 498-519, Feb. 2001.

[13] J. Chen, M. Fossorier, "Near Optimum Universal Belief Propagation Based Decoding of Low-Density Parity Check Codes," *Communications, IEEE Trans. on*, vol. COM-50, pp. 406-414, March 2002.

[14] M. Pedram, J. Rabaey, "Power Aware Design Methodologies", *Kluwer Academic Publishers*, 2002.

[15] R. Jejurikar, C. Pereira, R. Gupta, "Leakage Aware Dynamic Voltage Scaling for Real-Time Embedded Systems", *Design Automation Conf. 2004*, page 75-280, Jun. 2004.