Dynamic Power Management on LDPC Decoders

Erick Amador and Raymond Knopp
EURECOM
06904 Sophia Antipolis, France
name.surname@eurecom.fr

Vincent Rezard
Infineon Technologies France
06560 Sophia Antipolis, France
vincent.rezard@infineon.com

Renaud Pacalet
TELECOM ParisTech
06904 Sophia Antipolis, France
renaud.pacalet@telecom-paristech.fr

***Abstract*—This paper presents a dynamic power management strategy for the iterative decoding of low-density parity-check (LDPC) codes. We propose an online algorithm for adjusting the operation of a power manageable decoder. Decision making is based upon the monitoring of a convergence metric independent from the message computation kernel. Furthermore we analyze the feasibility of a VLSI implementation for such algorithm. Up to 54% savings in energy were achieved with a relatively low loss on error-correcting performance.**

I. Introduction

Low-density parity-check (LDPC) codes [1] are among the best known error-correcting codes because of their capacity-approaching performance. These codes have been adopted by various wired and wireless communication standards. This type of codes can be decoded by an iterative message-passing algorithm. Iterations are executed until a stopping criterion is satisfied. In practice the decoder is set to run for a maximum number of iterations such that a timing deadline is met for the worst case scenario. It is typical for codewords to converge well before the maximum number of iterations, representing a nonuniform workload. When targeting wireless communications, a battery-powered mobile terminal requires ultra-low power consumption. In this paper we argue that by monitoring the dynamics of the decoding process it should be possible to control a power-manageable decoder such that energy efficiency is improved.

Previous works on energy efficiency for LDPC decoders have concentrated on stopping criteria ([2] and the references therein) in order to detect early an undecodable block and avoid unnecessary decoder operation. In [3] and [4] the authors proposed a preprocessing stage that estimates the required decoding effort and proceed to adjust the system power mode (voltage and frequency) in order to have a constant decoding-time.

We propose a dynamic power management policy that looks for opportunities to slowdown the system in order to reclaim the slack due to a codeword that converges before the task deadline. The policy is based upon the monitoring of a convergence metric, namely the number of satisfied parity-check constraints at each decoding iteration. The rest of the paper is organized as follows: Section II summarizes LDPC codes and their iterative decoding. In Section III dynamic power management and the proposed control policy are explained, along with implementation details. Section IV shows the experimental setup and results for implementation of the proposed policy on a $65nm$ CMOS process. Section V concludes the paper.

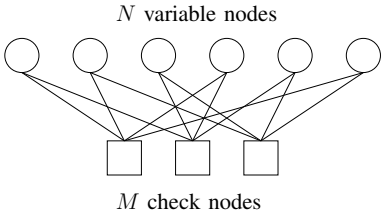


Fig. 1. LDPC code graph example

II. LDPC Codes

LDPC codes are linear block codes defined by a sparse parity-check matrix $\boldsymbol{H}_{M\times N}$. This matrix defines M parity-check constraints among N codeword symbols. A codeword $\boldsymbol{c}$ satisfies the condition:

$$\boldsymbol{H}\cdot\boldsymbol{c}^T=\boldsymbol{S}=\boldsymbol{0}\,, \tag{1}$$

where $\boldsymbol{S}$ is the *syndrome*. Furthermore the code can be represented by a bipartite graph in which rows of $\boldsymbol{H}$ are mapped to *check nodes* and columns to *variable nodes*. The nonzero elements in $\boldsymbol{H}$ define the connectivity between the nodes. Figure 1 shows an example code graph representation.

LDPC codes may be decoded by an iterative message-passing algorithm [1] where check nodes and variable nodes exchange extrinsic reliability messages associated with each codeword symbol. The operation of each node is independent and in general can be executed in parallel with other nodes. This characteristic enables the use of different scheduling techniques that impact the convergence speed of the decoding task. The expression in (1) is evaluated to indicate when convergence has been achieved and is used to stop the decoding task. Otherwise a maximum number of iterations is completed.

The processing complexity of the decoding task resides in the operation performed at the check nodes of the code graph, indeed it is in here where the tradeoff between error-correcting performance and complexity takes place. Optimal message computation is performed by the Sum-Product algorithm [5] at the expense of high complexity. The Min-Sum (MS) algorithm [6] performs a suboptimal message computation at reduced complexity. Several correction methods have been proposed to recover the error-correcting performance loss of the MS algorithm by downscaling the messages computed using both

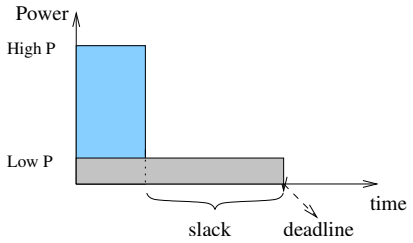


Fig. 2. Example power/slowdown scenario

a normalization (Normalized-MS) and an offset value (Offset-MS), [6]. In [7] a variant of the MS algorithm with quasi-optimal error-correcting performance is introduced, the Self-Corrected MS (SCMS) algorithm. The author defines variable-to-check node messages as either reliable or unreliable and proposes to eliminate the unreliable ones after each decoding iteration.

III. Dynamic Power Management

Dynamic power management (DPM) is a technique used to achieve energy-efficient operation on an electronic system. Several techniques exist to achieve this at different levels (system, gate levels) such as sleep, slowdown modes and clock gating. To apply DPM usually two premises are considered: the system experiences a nonuniform workload and it is possible to certain degree to predict the fluctuations of the workload.

A typical situation for iterative decoding is that operation is carried out on a high power mode, dimensioned to handle the worst case when the maximum number of iterations should be completed before a timing deadline. Figure 2 shows the power consumption of the same decoding task carried out at a high and low power mode. The area of each curve is the total energy spent. Depending upon the relationship among the power levels and the slowdown factor energy efficiency may be improved by reclaiming the slack left when running at high power.

A. Problem Definition

We consider an LDPC decoder to be a power manageable CMOS component governed by a power controller. The set $P = \{P_0, P_1, \ldots, P_{n-1}\}$ defines n power modes where, [3]:

$$\begin{aligned} P_k &= P_k^{sw} + P_k^{sc} + P_k^{leak} \\ &= \beta C_L V_k^2 f_k + I_{SC} V_k + I_{leak} V_k \end{aligned} \quad (2)$$

P_k^{sw} is the power due to the switching activity when charging and discharging the load capacitance C_L with switching factor β. P_k^{sc} is the power due to a short-circuit current when both NMOS and PMOS sections of the circuit are switched. P_k^{leak} is the power due to the leakage current I_{leak} (subthreshold plus reverse bias junction current), a technology dependent parameter. Each power mode P_k operates at a particular voltage level V_k and frequency f_k. In the following we assume that the first state P_0 consumes the most power, subsequent states consume each less power than the previous one. Given the quadratic relation between power and voltage and the linear relation between power and frequency, it is possible to slowdown the system (consequently augmenting processing time) such that the total energy expenditure is reduced. This is the principle behind the well-known concept of dynamic voltage and frequency scaling (DVFS), [8].

Given a workload of I iterations to be executed before a timing deadline d, we wish to find a subset of power modes $P' \subseteq P$ such that the total energy is minimized. As shown in [9] we assume the power function of each mode to be convex, such that it is more energy efficient to slow down a task execution. If an iteration k is executed in time t_k through a power mode P_k, the problem is stated as finding the optimal P' that minimizes the total energy spent:

$$\begin{aligned} minimize \ \sum_k^I P_k t_k \,, \ \ P_k \in P' \\ subject\ to \ \sum_k^I t_k \leq d \end{aligned} \quad (3)$$

B. Control Policy

An offline algorithm finds the optimal power mode to satisfy (3) assuming the total required number of decoding iterations is known. An online algorithm attempts to find an optimal power mode based on information available only at runtime. In order to formulate the online strategy it is necessary to look into the dynamics of the decoding process. The set M of parity-check constraints that must be satisfied after a decoding iteration can be used as a convergence metric in order to monitor the success or failure of the decoding task.

In [2] the number of satisfied parity-check constraints was proposed as a decision metric for a stopping criterion of the iterative decoding. Given the syndrome $\boldsymbol{S} = [s_1\ s_2\ \ldots\ s_M]^T$ the number of satisfied constraints at iteration i is:

$$N_{spc}^i = M - \sum_{m=1}^{M} s_m \quad (4)$$

We propose to use the complement of this metric as a means to predict the decoding effort at runtime after each decoding iteration. Once the remaining number of iterations is known a power mode that minimizes energy expenditure and respects the task deadline may be selected. Taking N_{spc} as a convergence metric has the benefit that it is independent from the decoding algorithm used.

Figure 3 shows the selected convergence metric (total unsatisfied constraints)

$$\epsilon_i = \sum_{m=1}^{M} s_m \quad (5)$$

per decoding iteration for an instance of an undecodable and a decodable block. This corresponds to the decoding of the code of length 1944 and rate $1/2$ defined in [10], simulated through the AWGN channel ($E_b/N_0 = 1dB$) with QPSK modulation and a maximum of 60 iterations. It is observed that this metric fluctuates around a mean value for undecodable

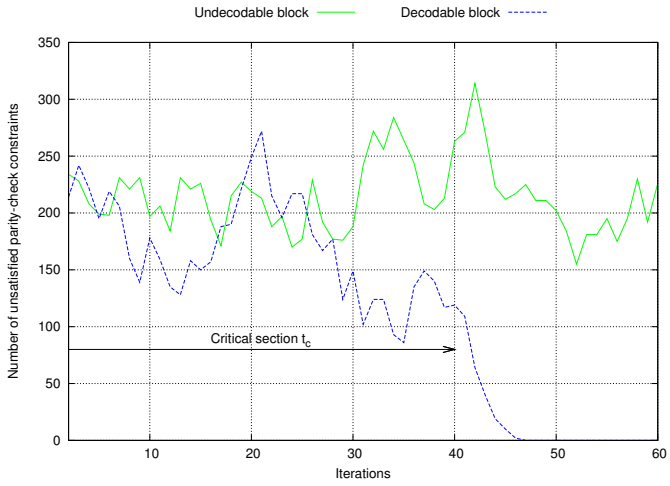


Fig. 3. Convergence metric for a decodable/undecodable block

blocks, but for decodable codewords it fluctuates for a period of time t_c and later enters a *convergence* mode characterized by a monotonic decreasing behavior. We refer to the period t_c as a *critical* period since no decision can be made regarding the convergence of the code. Based upon this behavior we propose the online strategy outlined in algorithm 1.

The algorithm essentially starts the system on the highest power mode since it tries to exit the critical period as fast as possible. During this period there is uncertainty with respect to the convergence of the block and indeed the energy cap E_c represents a stopping criterion. As such, because of this there could be false alarms, i.e. codewords that would have been succesfully decoded in the absence of such criterion. This translates into a loss on error-correction performance. Convergence is detected when the last q values of the metric ϵ_i are strictly decreasing. If the consumed energy is below the energy cap the convergence metric is used to estimate the remaining decoding effort. Equation (6) shows a function based on the w last values of the metric, where prediction functions of different degrees of complexity may be used. A linear approximation provides the simplest prediction function. Figure 4 shows how the two last values of the metric are used to approximate the convergence region to a line segment.

Given the two latest values of the convergence metric ϵ_n and ϵ_{n+1} (at iterations n and $n+1$ respectively) the total estimated decoding iterations $\hat{I}$ is given by:

$$\hat{I} = n - \frac{\epsilon_n}{\epsilon_{n+1} - \epsilon_n} \tag{8}$$

Overheads associated with the switching between power modes (e.g. transition times and energy cost) are not explicitly mentioned in algorithm 1 for simplicity.

The offline algorithm can perform a decoding task with a minimum energy expenditure by finding the proper power mode that slows down the system such that the timing constraint is satisfied. For each power mode there is a corresponding slowdown factor α_k, where for the fastest mode $\alpha_0 = 1$. The performance of the online algorithm can be evaluated by the competitive ratio with respect to the cost of the offline alternative. The notion of cost in this case is taken as the total consumed energy by each strategy. The upper bound of this ratio is given by:

$$c = \frac{P_0}{P_{n-1}\alpha_{n-1}} \tag{9}$$

[1]Overheads associated with switching between power modes are not explicitly noted.

Algorithm 1 Online Power Management

i: current decoding iteration, $i \in \{1, 2, \ldots, I_{max}\}$
ϵ_i: total unsatisfied constraints at iteration i
1. Decoding starts, $i = 1$
  Set power mode P_0
2. Critical section
for $j = 1$ to I_{max} **do**
  if $(P_0 t_0 i \geq E_c)$ **then**
    Halt decoding
  else
    Check convergence state
    if $(\epsilon_i < \epsilon_{i-1} < \ldots < \epsilon_{i-q})$ **then**
      Go to 3.
    end if
  end if
end for
3. Convergence section
for $j = i$ to I_{max} **do**
  Estimate required iterations

$$\hat{I} = f(\epsilon_i, \epsilon_{i-1}, \ldots, \epsilon_{i-w}) \tag{6}$$

  Switch to power mode P_k such that [1]

$$\sum_{m=1}^{i} t_m + (\hat{I} - i)t_k \leq d \tag{7}$$

  Syndrome verification
  if $(\boldsymbol{S} = \mathbf{0})$ **then**
    Halt decoding
  end if
end for

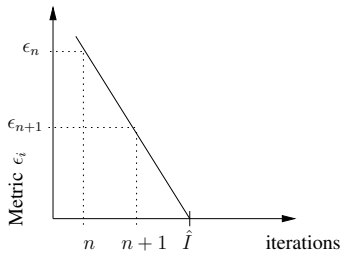


Fig. 4. Estimation of decoding effort by linear approximation

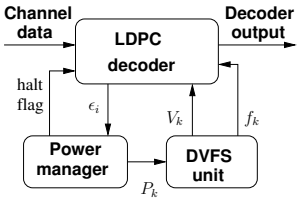


Fig. 5. System block diagram

TABLE I
DECODER POWER MODES

Mode	Voltage	Frequency	Power	Slowdown factor
P_0	V_{max}	f_{max}	P_{max}	1.0
P_1	$0.9V_{max}$	$0.66f_{max}$	$0.5346P_{max}$	1.5
P_2	$0.75V_{max}$	$0.5f_{max}$	$0.2812P_{max}$	2.0
P_3	$0.68V_{max}$	$0.4f_{max}$	$0.1849P_{max}$	2.5

C. Implementation

The proposed system for dynamic power management is shown in figure 5. The LDPC decoder receives channel observations in the form of log-likelihood ratios and outputs hard-decision bits for the decoded message. The decoder constantly feeds the convergence metric signal ϵ_i to a power manager unit. This unit executes the control policy from algorithm 1 and sets the state for a DVFS unit.

Table I shows the characterization of the power modes of the target system considered, where $V_{max} = 1.32V$ and $f_{max} = 400MHz$. From this specification the power manager unit can be properly dimensioned. Figure 6 shows the architecture of the power manager unit. This unit executes three main tasks duly represented in this figure: convergence and energy cap detection, estimation of remaining decoding effort and power mode selection. The processing time per iteration of each power mode t_k is used to calculate the cumulative decoding time. This value is constantly being added to the estimated remaining time in order to update the power mode selection. The estimation of the remaining decoding time is performed by the linear approximation of the convergence region according to figure 4. The two latest values of ϵ_i are used for this estimation as well as to track the behavior of the decoding task in order to halt it if necessary. This corresponds to $q = w = 2$ in algorithm 1.

We implemented and synthesized a SCMS-based LDPC decoder for the codes defined in [10] on a CMOS $65nm$ technology process. A message quantization of 6-bits was used. The decoder architecture is shown in figure 7. The decoder essentially updates posterior messages that are distributed by an interconnection network (acting as the edges of the underlying code graph) to a set of processing units. These units process the messages along with the previous results (extrinsic messages) and write back the newly calculated posterior messages (again the messages are properly distributed by an interconnection network).

The processing core consists of 15 units that perform the SCMS decoding of a row in $\boldsymbol{H}$ in serial fashion as outlined in [7]. Each unit has an extrinsic messages memory that holds the

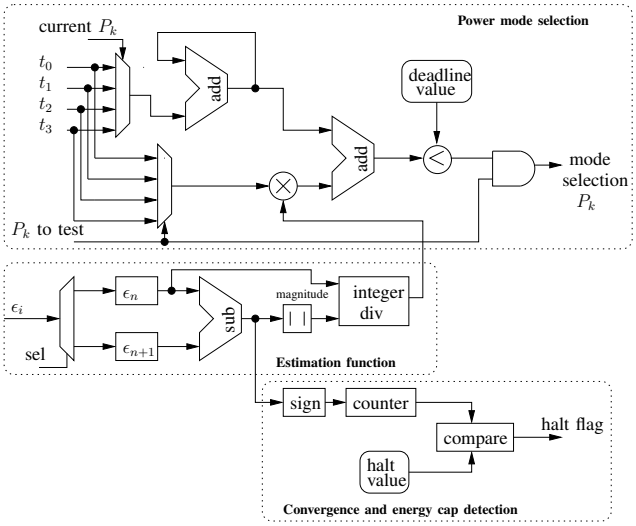


Fig. 6. Architecture of power manager unit

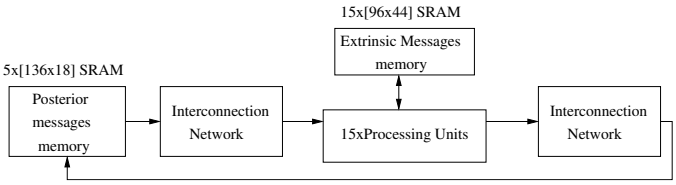


Fig. 7. LDPC decoder architecture

messages obtained after each decoding round per row as well as the status whether each message was erased or not. Figure 8 shows the architecture of this unit. This unit compares the sign of an input message (subtraction of extrinsic message from posterior message) with its previous sign and erases messages (introduces zeros) according to the SCMS algorithm. These *corrected* messages are passed on to a minimum finder block that outputs the first and second minima and the position (index) of the first minimum value. The extrinsic message and its sign, along with the newly calculated posterior message are written back to the corresponding locations.

There are numerous works on how to implement a DVFS unit, using different techniques where several tradeoffs take place: size and power overhead, mode switching speed and

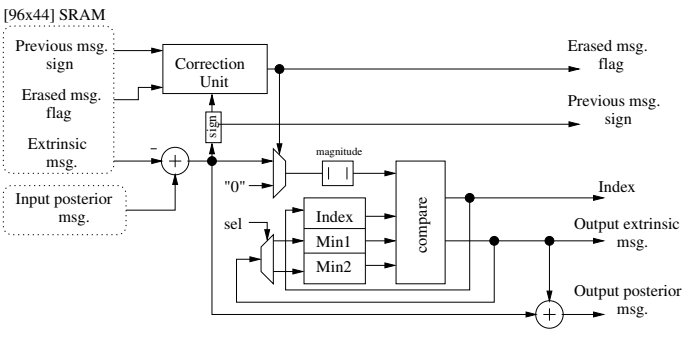


Fig. 8. Architecture of processing unit

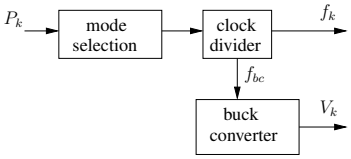


Fig. 9. DVFS unit diagram

TABLE II
AREA AND POWER COMPARISON

Component	Area $[mm^2]$	Power $[mW]$
Decoder (No DPM)	0.85	70
Power manager	0.08	5
DVFS unit	0.12	10

conversion efficiency. The work in [11] provides a study on on-chip regulators for DVFS implementation on a dedicated core. This and similar work in [12] show sufficiently fast regulators (voltage transition times on the order of tens of nanoseconds) for tight applications as LDPC decoding. Based on this we target an on-chip solution to implement the DVFS unit, shown in figure 9. A mode selection signal selects the appropriate setting for a clock divider and a buck converter with hysteretic control to generate the signals f_k and V_k that drive the decoder unit.

In table II we compare the area and power of the different units involved in the implementation of the power management system. These results provide an initial assessment in the overhead due to the power management strategy.

IV. RESULTS

In order to tune and assess the impact of this power management technique simulations were performed to observe the behavior of the metric that drives the decision making of the proposed policy. Figure 10 shows the average decoding iterations and average critical iterations from simulations of the code in [10] with $N = 1944$ and $R = 1/2$ over the AWGN channel, QPSK modulation, and a maximum of 60 iterations. For the decoder described in the previous section it was estimated from several use cases and their switching activity that each decoding iteration consumes an average of $80nJ$. By observing the average number of critical iterations an energy cap $E_c = 2.96\mu J$ was set to investigate the impact on error-correction performance. Figure 11 shows the simulated performance (bit/frame-error rates) using the already described simulation setup with and without the proposed dynamic power management policy. For example, at a bit-error rate of 10^{-6} there is a performance loss of less than $0.05dB$.

In figure 12 we show the average energy savings obtained by the online algorithm with respect to the absence of a power management strategy (constant operation at full power). Three code lenghts N were used along with two coding rates in order to observe the behavior for several use cases. The optimal offline algorithm knows in advanced the total required decoding effort and immediately chooses the minimum power state that guarantees the execution of the task within the required deadline. In the other hand the online algorithm starts with the fastest power mode and tries to switch constantly

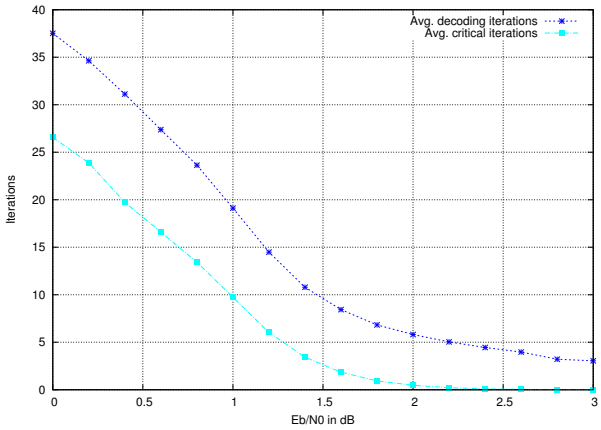


Fig. 10. Average decoding and critical iterations

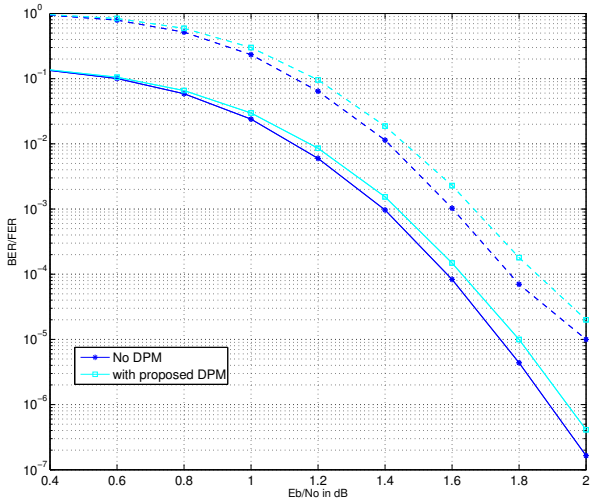
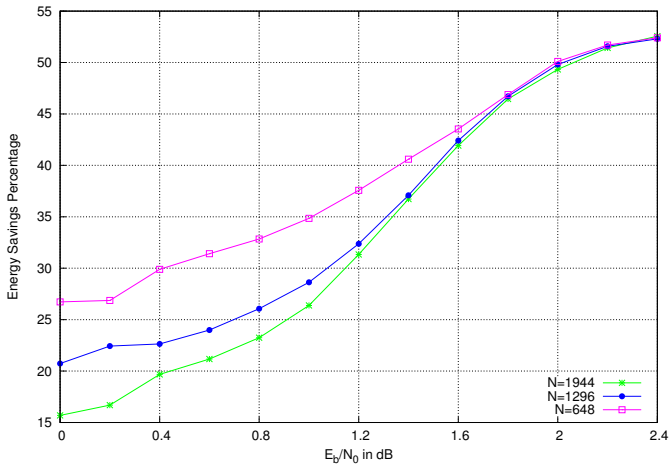


Fig. 11. Error-correcting performance: BER solid lines, FER dashed lines

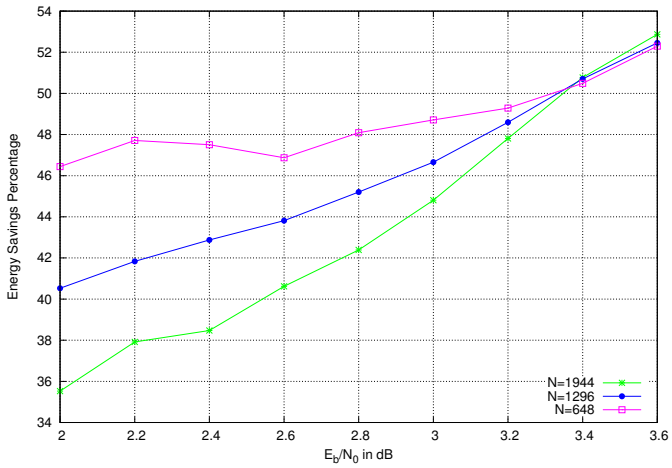
to a lower power mode given the estimation of the decoding effort carried out with $f(\epsilon_i)$. At a low SNR there are less opportunities for energy savings (20% with code rate 1/2 and 40% for rate 5/6) because of a higher decoding effort (longer critical time), but for the high SNR region up to 54% energy savings were obtained. For comparison purposes the work in [3] obtained up to 37% in savings while [4] obtained up to 30%.

In figure 13 we show how the online algorithm savings deviate from the savings obtained by the offline alternative for the case of coding rate 5/6. This deviation shows how well the online algorithm can perform, in fact it differs from the savings obtained by the offline alternative only by 1% as the SNR increases.

The competitive ratio for this setup is $c = 2.17$, meaning that the online algorithm can find a solution with cost (total energy expenditure) less than c times the cost of the optimal offline algorithm. Nevertheless from the simulation results it was observed that the competitive ratio had an upper bound of $c = 0.97$, not surprising as competitive analysis often provides an overly pessimistic bound for the performance of algorithms.



(a) Rate 1/2



(b) Rate 5/6

Fig. 12. Average energy savings obtained for several use cases

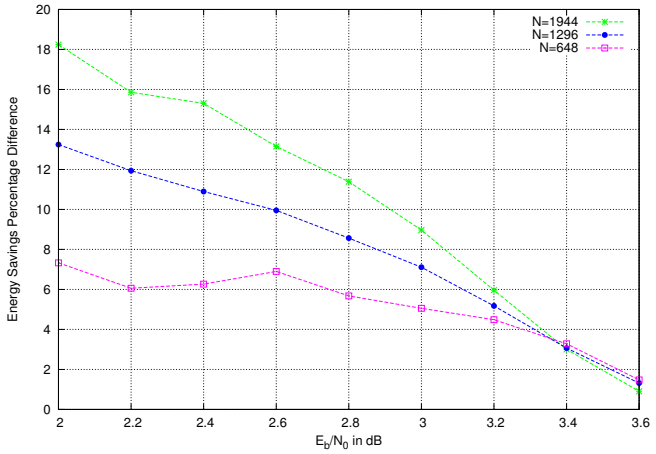


Fig. 13. Difference on savings between online and offline algorithms

V. Conclusions

We have presented a dynamic power management policy for the iterative decoding of LDPC codes. The policy is based upon an online algorithm that monitors the decoding task through a convergence metric. By making a recurrent estimation on the required decoding effort the policy adjusts the performance of the system so that it minimizes the energy consumption and meets the task deadline. The number of unsatisfied parity-check constraints captures the dynamics of the decoding task. This metric is used to estimate the decoding effort and to decide whether halting the task is necessary. In this way the decoder runs at a high power mode during a critical time before it enters a convergence state, subsequently the policy estimates the remaining decoding effort and adjusts the power mode such that energy expenditure is minimized. The online algorithm performed very close to the offline alternative, differences in energy savings among both algorithms were up to 18% at low SNR and approached 1% at high SNR. The proposed policy was implemented into a VLSI system of CMOS $65nm$ technology, the area and power overheads were marginal in comparison to the observed gains in energy efficiency. Energy savings up to 54% were achieved, an important figure for battery-powered terminals in a communication system. The error-correcting performance loss for this policy was lower than $0.05dB$ at a BER of 10^{-6}.

References

[1] R. Gallager, "Low-Density Parity-Check Codes," *IRE Trans. Inf. Theory*, vol. 7, pp. 21–28, January 1962.

[2] D. Shin, K. Heo, S. Oh, and J. Ha, " A Stopping Criterion for Low-Density Parity-Check Codes ," in *Proc. of IEEE VTC 2007-Spring* , 2007, pp. 1529–1533.

[3] W. Wang and G. Choi, " Speculative Energy Scheduling for LDPC Decoding ," in *8th International Symposium on Quality Electronic Design* , 2007, pp. 79–84.

[4] W. Wang, G. Choi, and K. Gunnam, " Low-Power VLSI Design of LDPC Decoder Using DVFS for AWGN Channels ," in *Proc. of the 22nd International Conference on VLSI Design* , 2009, pp. 51–56.

[5] F.R. Kschischang, B. Frey, and H.A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, pp. 498–519, Febraury 2001.

[6] J. Chen and M. Fossorier, " Near Optimum Universal Belief Propagation based Decoding of LDPC codes ," in *IEEE Trans. on Comm.* , 2002, pp. 406–414.

[7] V. Savin, " Self-Corrected Min-Sum Decoding of LDPC Codes ," in *Proc. of IEEE International Symposium on Information Theory* , 2008, pp. 146–150.

[8] P. Macken, M. Degrauwe, M.V. Paemel, and H. Orguey, " A Voltage Reduction Technique for Digital Systems ," in *IEEE International Solid State Circuits Conference* , 1990, pp. 238–239.

[9] S. Irani, S. Shukla, and R. Gupta, " Algorithms for Power Savings," in *Proc. of the 14th ACM-SIAM Symposium on Discrete Algorithms* , 2003, pp. 37–46.

[10] IEEE-802.11n, "Wireless LAN Medium Access Control and Physical Layer Specifications: Enhancements for Higher Throughput," *P802.11n/D3.07*, March 2008.

[11] W. Kim, M.S. Gupta, G.Y. Wei, and D. Brooks, " System Level Analysis of Fast, Per-Core DVFS using On-Chip Switching Regulators ," in *IEEE 14th International Symposium on High Performance Computer Architecture* , 2008, pp. 123–134.

[12] F. Su, W.H. Ki, and C.Y. Tsui, " Ultra Fast Fixed-Frequency Hysteretic Buck Converter with Maximum Charging Current Control and Adaptive Delay Compensation for DVS Applications ," *IEEE Journal of Solid-State Circuits* , vol. 43, no. 4, pp. 815–822, April 2008.