# Power Reduction Techniques for LDPC Decoders

Ahmad Darabiha, *Student Member, IEEE*, Anthony Chan Carusone, *Member, IEEE*, and Frank R. Kschischang, *Fellow, IEEE*

***Abstract*—This paper investigates VLSI architectures for low-density parity-check (LDPC) decoders amenable to low- voltage and low-power operation. First, a highly-parallel decoder architecture with low routing overhead is described. Second, we propose an efficient method to detect early convergence of the iterative decoder and terminate the computations, thereby reducing dynamic power. We report on a bit-serial fully-parallel LDPC decoder fabricated in a 0.13-$\mu$m CMOS process and show how the above techniques affect the power consumption. With early termination, the prototype is capable of decoding with 10.4 pJ/bit/iteration, while performing within 3 dB of the Shannon limit at a BER of $10^{-5}$ and with 3.3 Gb/s total throughput. If operated from a 0.6 V supply, the energy consumption can be further reduced to 2.7 pJ/bit/iteration while maintaining a total throughput of 648 Mb/s, due to the highly-parallel architecture. To demonstrate the applicability of the proposed architecture for longer codes, we also report on a bit-serial fully-parallel decoder for the (2048, 1723) LDPC code in 10GBase-T standard synthesized with a 90-nm CMOS library.**

***Index Terms*—10 Gigabit Ethernet, channel coding, iterative message passing, low-density parity-check codes, very-large-scale integration.**

## I. Introduction

LDPC codes [1] have been adopted for several new digital communication standards due to their excellent error correction performance, freedom from patent protection, and inherently-parallel decoding algorithm [2]–[4]. Most of the research on LDPC decoder design so far has focused on code designs, decoding algorithms, and decoder architectures that improve decoder throughput. Fewer papers have discussed low-power architectures for LDPC decoders. Analog decoders have been proposed for low-power decoding of LDPC [5] and Turbo codes [6]. However, analog decoders have only been demonstrated on codes with block lengths less than 250 bits. Scaling analog decoders to longer block lengths will be complicated by device mismatches and the need to store and buffer hundreds of analog inputs to the decoder. The performance of such short block-length codes is insufficient for the targeted applications, and the throughput of analog decoders is limited to less than 50 Mb/s. In nanoscale CMOS processes, digital LDPC decoders appear to be the best solution for future communication applications that demand performance near the limits of channel capacity.

Manuscript received December 21, 2007; revised February 24, 2008. Published July 23, 2008 (projected). This work was supported by Gennum Corporation, Canada.

The authors are with The Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario M5S 3G4, Canada (e-mail: ahmadd@eecg.utoronto.ca; tcc@eecg.utoronto.ca; frank@comm.utoronto.ca).

Digital Object Identifier 10.1109/JSSC.2008.925402

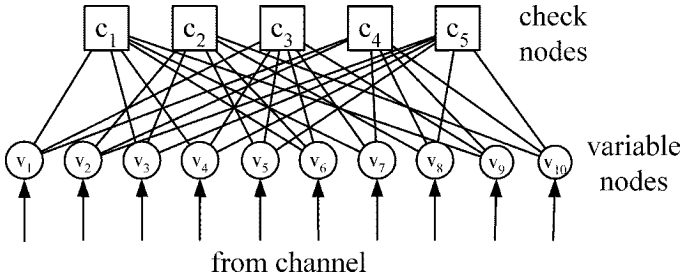


Fig. 1. LDPC code Tanner graph.

In this paper, we discuss techniques for low-power digital LDPC decoders. In Section II, a highly-parallel decoder architecture with low routing overhead is described. The parallelism permits operation from a low supply voltage, thereby providing low-power consumption. In Section III, we investigate an early termination scheme to reduce power consumption by stopping the decoding iterations as soon as a valid codeword is detected. Section IV-A reports results from a prototype bit-serial fully-parallel LDPC decoder fabricated in a 0.13-$\mu$m CMOS process.

## II. Low-Power Parallel Decoders

### A. Background

LDPC codes are a subclass of linear error control codes and can be described as the null space of a sparse {0,1}–valued parity-check matrix, $H$. They can also be described by a bipartite graph, or *Tanner graph*, in which check nodes $\{c_1, c_2, \ldots, c_C\}$ represent the rows of $H$ and variable nodes $\{v_1, v_2, \ldots, v_V\}$ represent the columns. An edge connects the check node $c_m$ to the variable node $v_n$ if and only if $H_{mn}$ is nonzero. A code is called ($d_v$, $d_c$)-regular if every column and every row of $H$ has $d_v$ and $d_c$ ones, respectively. As an example, Fig. 1 shows the Tanner graph for a (3, 6)-regular LDPC code with $V = 10$ variable nodes and $C = 5$ check nodes.

Min-sum decoding [7] is a type of iterative message-passing decoding that is commonly used in LDPC decoders due to its simplicity and good BER performance. Each decoding iteration consists of updating and transferring *extrinsic* messages between neighboring variable and check nodes. A message is a *belief* about the value of corresponding received bit and is expressed in the form of log-likelihood ratio (LLR). At the beginning of min-sum decoding, the variable nodes pass the LLR value of the received symbols (i.e., the *intrinsic* message) to all the neighboring check nodes. Then each iteration consists of check update phase followed by variable update phase. During the check update phase the outgoing message on each edge of the check node is calculated as a function of the incoming messages from all the other edges: the magnitude of the output is the minimum of the input magnitudes and the sign is the parity

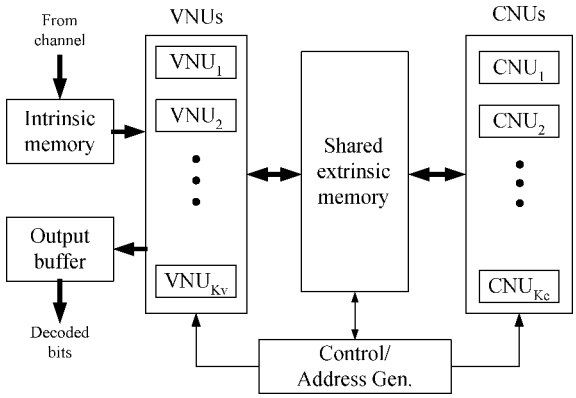


Fig. 2. Partially-parallel LDPC decoder.

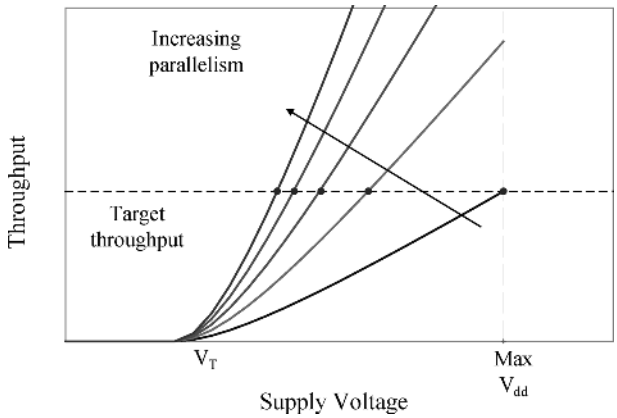


Fig. 3. Increased parallelism allows reduced supply voltage.

of the signs of the inputs. During the variable update phase the outgoing message on each edge of a variable node is calculated as the sum of all the incoming messages from all other edges plus the intrinsic message from the channel.

A generic LDPC decoder architecture is shown in Fig. 2. It comprises $K_v$ shared variable node update units (VNUs), $K_c$ shared check node update units (CNUs), and a shared memory fabric used to communicate messages between the VNUs and CNUs. Inputs to each CNU are the outputs of VNUs fetched from memory. After performing some computation (e.g., MIN operation for the magnitude and parity calculation for the signs in min-sum decoding), the CNU's outputs are written back into the extrinsic memory. Similarly, inputs to each VNU arrive from the channel and several CNUs via memory. After performing the message update (e.g., SUM operation in min-sum decoding), the VNU's outputs are written back into the extrinsic memory for use by the CNUs in the next decoding iteration. Decoding proceeds with all CNUs and VNUs alternately performing their computations for a fixed number of iterations, after which the decoded bits are obtained from one final computation performed by the VNUs.

By increasing the number of VNUs and CNUs, $K_v$ and $K_c$, the decoder performs more computations in parallel. When the decoder is operated from a fixed supply voltage, such increased parallelism may be used to achieve higher throughput, with attendant increases in power and area. However, it is well known that increased parallelism can also permit a digital system to operate from a lower supply voltage with constant throughput resulting in greatly decreased power consumption [8]. In general, the power advantages offered by parallelism are mitigated by the overhead associated with multiplexing and demultiplexing the system's inputs and outputs amongst several parallel computing units. However, in the case of an LDPC decoder, all of the signals required for each iteration are already available in parallel in the extrinsic memory (Fig. 2). The inherent parallelism of LDPC iterative decoding with long block lengths is, therefore, well suited to implementation with a low supply voltage. Until now, this property has not been fully exploited to design a low-voltage, low-power LDPC decoder.

### B. Analysis

The reduced supply voltage obtainable using increased parallelism is described qualitatively in Fig. 3. There is a practical limit to the decoder's parallelism power savings when the number of VNUs and CNUs equal the total number of variable and check node computations required in each iteration. Further increases in $K_v$ or $K_c$ are not straightforwardly possible since the required input messages are not available in memory. As shown in Fig. 3, unless the targeted throughput is low the supply voltage will remain significantly higher than the MOS threshold voltage. Although subthreshold circuits have been shown to be energy efficient, they are mostly suitable for low-to-mid performance systems [9] with relaxed constraints on throughput. Since many present and future applications of LDPC codes target a multi-gigabit-per-second throughput, our analysis will proceed assuming a square-law MOS model.

To quantify the power reduction that can be offered by highly-parallel LDPC decoding architectures, let us compare two decoders: a reference design with $K_v$ VNUs and $K_c$ CNUs; and a design with increased parallelism having $(k \cdot K_v)$ VNUs and $(k \cdot K_c)$ CNUs, $k > 1$. The dynamic power consumption of these decoders, operated at a clock frequency $f$ from a supply voltage $V_{\rm dd}$ is $fC_{\rm eff}V_{\rm dd}^2$, where $C_{\rm eff}$ is the effective capacitance of each decoder including an activity factor.

The total effective capacitance $C_{\rm eff}$ consists of two parts. First, the effective capacitance due to the computational and control logic inside the CNUs and VNUs, $C_L$. Second, the effective capacitance due to the memory and storage elements, $C_M$.

By increasing the parallelism by $k$, $C_L$ also scales with $k$ because the number of VNUs and CNUs instantiated in hardware is increased by the same factor. The number of required storage elements on the other hand is a function of the number of edges in the code graph and is independent of the parallelism factor. However, since there are $k$ times more processing units in the new decoder, the average number of total memory accesses per clock cycle scales with $k$. As a result the memory capacitance activity factor, and hence $C_M$, also scale with $k$. Therefore, the effective capacitance of the parallel design is $C_2 = kC_L + kC_M = kC_1$.

The parallel decoder can operate at a clock frequency $f_2$ that is $k$ times lower than the reference design clock frequency, $f_1$, while maintaining the same throughput: $f_2 = f_1/k$. Since we are striving for low-power operation, each decoder operates from the lowest supply voltage that will support its targeted clock frequency. Hence, the parallel design can be operated

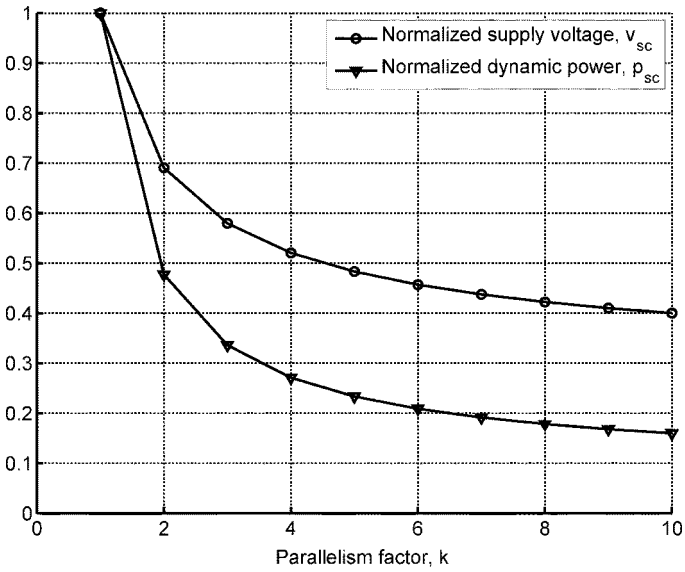


Fig. 4. Power reduction as a result of a parallel architecture.

from a lower supply voltage ($V_{\rm dd2}$) than the reference design ($V_{\rm dd1}$). Following an analysis similar to [10], we have $V_{\rm dd2} = v_{sc} V_{\rm dd1}$, where

$$v_{sc} = m + \frac{(1-m)^2}{2k} + \sqrt{\left(m + \frac{(1-m)^2}{2k}\right)^2 - m^2} \quad (1)$$

and $m = V_t / V_{\rm dd1}$. Therefore, the power savings offered by the parallel design is

$$p_{sc} = \frac{P_2}{P_1} = \frac{1}{k} k v_{sc}^2 = v_{sc}^2. \quad (2)$$

Fig. 4 shows the normalized supply voltage, $v_{sc}$, required for different values of $k$ to maintain a constant throughput based on (1) for a typical 0.13-$\mu$m CMOS process where $V_t = 0.3$ V and $V_{\rm dd1} = 1.2$ V. It also shows the normalized power, $p_{sc}$, for the same range of $k$ based on (2).

The preceding analysis makes two assumptions that have not yet been discussed:

a) Power consumption is dominated by dynamic power dissipation. Our measurements for the decoder presented in this work suggest that leakage power constitutes less than 1% of the total power dissipation when operating at the maximum clock frequency and with typical supply voltage values. This is also consistent with the power measurements reported in [11].

b) The overhead associated with the increased parallelism is negligible. If, for example, interconnect limits the critical path delay or dominates the power consumption of the design, the benefits of increased parallelism will be less than predicted above. Hence, the focus of Section II-C is to minimize the overhead associated with highly-parallel decoders.

### C. Fully-Parallel Decoder With Bit-Serial Message Passing

Following the power efficiency discussion above, we have adopted a fully-parallel architecture where a separate VNU or CNU is designated for each variable node or check node in the code Tanner graph. Another advantage of fully-parallel decoder architecture is that unlike most partially-parallel decoders that are based on a particular code construction (such as the (3, k)-regular construction in [12], the Architecture-Aware code construction in [13], or the irregular and quasi-cyclic codes constructed in [14] and [15]), the fully-parallel architecture can be applied to irregular codes with no constraint on the code structure. This is done simply by instantiating VNUs and CNUs of the desired degree and connecting them based on the code graph. The only consideration is that the timing performance of the decoder for irregular codes will be typically limited by a critical path through the nodes with highest degree.

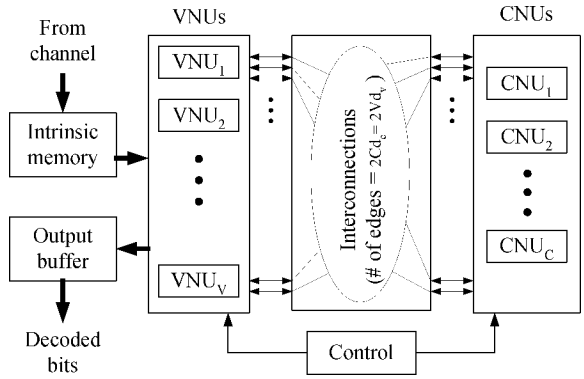


Fig. 5. Fully-parallel iterative LDPC decoder architecture.

The fully-parallel decoder architecture implies that changing the underlying LDPC code in general requires resynthesizing the decoder based on the new parity check matrix. Although this is acceptable for applications such as 10GBase-T which specify only one fixed code in the standard, other applications such as WiMAX need to be able to decode multiple LDPC codes with different lengths and rates. One possible solution is to implement the fully-parallel decoder for a Tanner graph that contains all the individual codes as its subgraphs. In such a decoder, different nodes and edges need to be activated or deactivated depending on the specific code. This approach is particularly applicable to cases such as the WiMAX standard in which all the codes are punctured and/or shortened versions of one single rate-1/2 2304-bit code [3]. As a result, a fully-parallel LDPC decoder compliant with the WiMAX standard can be realized by implementing this code and adding the control logic to disable some VNUs, CNUs and edges depending on the target subcodes.

Fig. 5 shows the high-level architecture of the 0.13-$\mu$m CMOS bit-serial LDPC decoder implemented in this work. The decoder is based on a (4, 15)-regular LDPC code with $V = 660$ variable nodes and $C = 176$ check nodes. This code was constructed using a progressive edge-growth algorithm [16] that minimizes the number of short cycles in the code's Tanner graph. It can be seen that the extrinsic memory block of Fig. 2 is replaced with the interconnections. This is because in a fully-parallel architecture each extrinsic message is only written by one VNU or CNU, so the extrinsic memory can now be distributed amongst VNUs and CNUs and no address generation is needed.

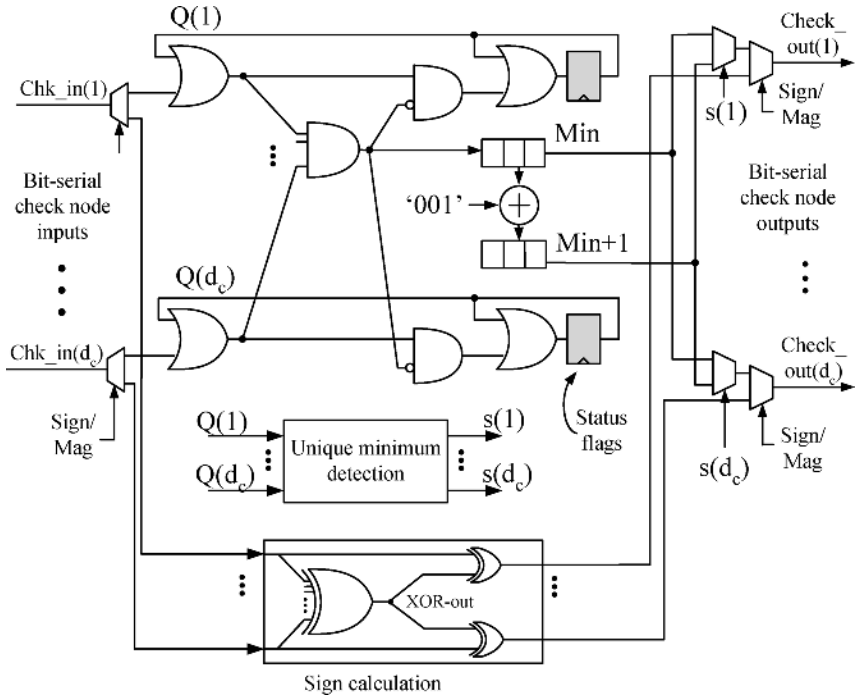


Fig. 6. CNU schematic for approximate min-sum decoding.

The major challenge in implementing highly-parallel decoders [11] is the large area and the overhead effects such as the routing complexity that are not modeled in the discussion in Section II-B. To reduce the effect of routing complexity, we have used a bit-serial message-passing scheme in this work where multi-bit messages are communicated between the nodes over multiple clock cycles [17]. In addition to reducing the routing complexity, the bit-serial message-passing requires less logic to perform min-sum LDPC decoding because both the MIN and SUM operations are inherently bit-serial. As a result, bit-serial VNUs and CNUs can be efficiently implemented to generate only partial 1-bit extrinsic messages every clock cycle.

Although bit-serial message-passing reduces the amount of global wiring, the routing complexity will eventually limit the maximum length of the LDPC codes that can be implemented in a bit-serial fully-parallel decoder. However, the important point is that the bit-serial scheme pushes the practical code length limit to higher values, making it feasible to implement fully-parallel decoders for emerging high-speed standards such as 10GBase-T or Mobile WiMAX which specify code lengths of 2048 and 2304, respectively.

The decoder in this work performs an approximate min-sum decoding algorithm that reduces the area of the CNUs by more than 40% compared with conventional min-sum decoding with only a 0.1 dB performance penalty at BER $= 10^{-6}$ [17]. Fig. 6 shows the CNU schematic where the inputs and outputs are communicated bit-serially in sign-magnitude MSB-first format. The top section of the schematic is for calculating the output magnitudes as in [17] and the lower block in the figure calculates the output sign using an XOR-tree. The VNU logic in min-sum decoding must take the sum of all its inputs. Unlike the CNUs, the SUM operations in the VNUs are more efficiently performed for inputs in LSB-first 2's complement format. So, the message formats are converted accordingly at the output of VNUs and

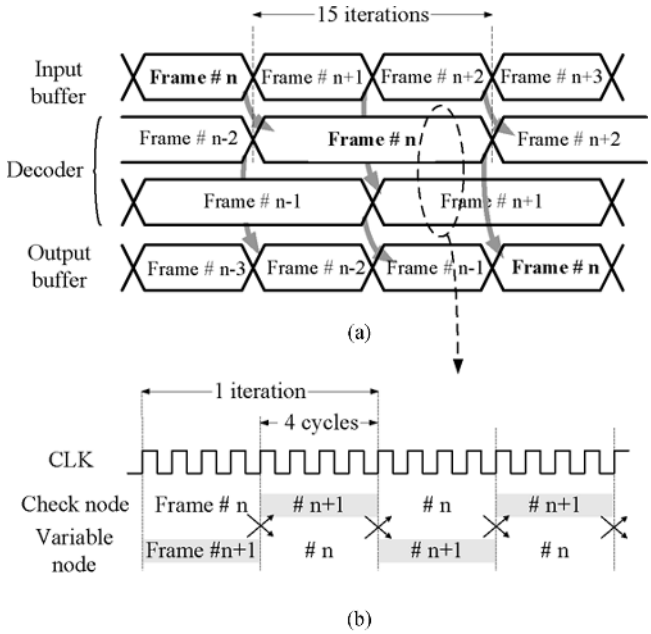


Fig. 7. Timing diagram for block-interlaced bit-serial decoding.

CNUs. Converting between LSB-first and MSB-first bit-serial communication requires additional registers to store the messages. However, these registers are already present in the CNUs and VNUs for the block interleaving as explained below. The design has a core utilization of 72%, compared with 50% in the fully-parallel LDPC decoder reported in [11] that does not employ bit-serial message passing. The high utilization implies that there is little routing overhead associated with the decoder's parallelism.

The timing diagram of the decoder is shown in Fig. 7. In this decoder, 4-bit quantized LLR messages are transferred between

VNUs and CNUs bit-serially in four clock cycles. As a result, each decoding iteration takes four clock cycles in the check node and four cycles in the variable node. After every four cycles, the variable and check nodes swap messages, allowing two different frames to be simultaneously decoded in an interleaved fashion.

Section IV-A will report the measured timing and power performance of the implemented decoder. It will show how voltage scaling can be used to trade high throughput for low-power decoding. Even without the techniques described in the next section, voltage scaling results in an energy efficiency of 7.4 pJ/bit/iter at 648 Mb/s throughput which is lower than the best previously-reported digital and analog iterative decoders [11], [5].

## III. LDPC Decoding With Early Termination

### A. Background

LDPC decoders generally correct most bit errors within the first few decoding iterations. Subsequent iterations provide diminishing incremental improvements in decoder performance. The number of iterations performed by the decoder, $I_M$, is usually determined *a priori* and hard-coded based on worst-case simulations. Therefore, the decoder performs $I_M$ iterations even though it will usually converge to its final output much sooner. We propose a decoder architecture that automatically detects when it has converged to its final output and shuts off all VNUs and CNUs for the remainder of each frame to save power.

Earlier work in this area has focused on identifying particular bits within each frame that appear likely to have converged [18], [19]. They have suggested that one can stop updating extrinsic messages for those reliable bits while other unreliable bits are still being decoded. The resulting power savings depends on the specific criteria used to identify the reliable bits. Unfortunately, these bits are sometimes incorrectly identified, so the decoder's performance suffers. In [20], an additional post-processing decoder is introduced to mitigate this performance degradation. Naturally, there is overhead associated with identifying the reliable bits and with the post-processing decoder. The overhead reduces the potential power savings of this approach.

In this work, instead of trying to identify individual bits that appear to have converged early, we monitor the entire frame to determine when the decoder has converged to a valid codeword. We then deactivate the entire decoder for the remaining iterations to save power. The remainder of this section describes a hardware-efficient implementation of this technique with significant power savings and no performance degradation.

### B. Early Termination

Although EXIT charts can be used to determine the average number of iterations required for convergence of an LDPC decoder operating on very long block lengths [21], for practical block lengths of 1000 to 10,000 bits the estimates so obtained are inaccurate. Instead, we have used extensive simulations to investigate the convergence behavior of two practical LDPC codes.

Fig. 8 shows the BER versus input SNR for two different LDPC codes under 4-bit-quantized min-sum decoding. The code in Fig. 8(a) is the Reed–Solomon based (6, 32)-regular 2048-bit LDPC code as specified for the 10 Gigabit Ethernet standard [2], while the code in Fig. 8(b) is the same code employed in the hardware prototype described in Section II. Each code is simulated with different number of iterations, $I_M$. These simulations indicate that little performance improvement is observed for either code as the number of iterations is increased from $I_M = 12$ to $I_M = 16$. Therefore, no more than $I_M = 16$ iterations are required for either code.

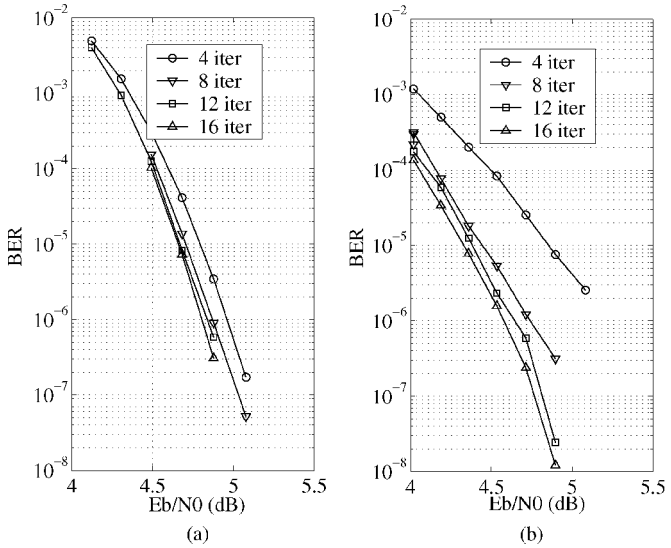


Fig. 8. BER versus maximum number of iterations under 4-bit quantized min-sum decoding: (a) Reed-Solomon based (6, 32)-regular 2048-bit code and (b) PEG (4, 15)-regular 660-bit code.

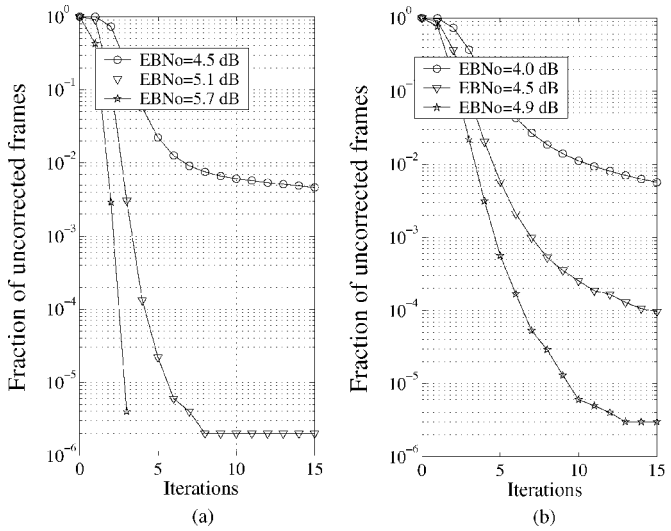


Fig. 9. The fraction of uncorrected frames versus iteration number for (a) a Reed–Solomon based (6, 32)-regular 2048-bit code, and (b) a PEG (4, 15)-regular 660-bit code.

The convergence behavior of the same two codes is shown in Fig. 9 which plots the average fraction of uncorrected frames versus the iteration number. These two figures show that the vast majority of frames are correctly decoded in the first few iterations. For example, for the code in Fig. 9(a), at an $Eb/No$ of 5.1 dB more than 99.99% of all frames have been successfully decoded during the first five iterations.

Fig. 10 plots the ratio of the average number of required iterations to $I_M$, $\alpha$, versus input SNR for the same two codes as

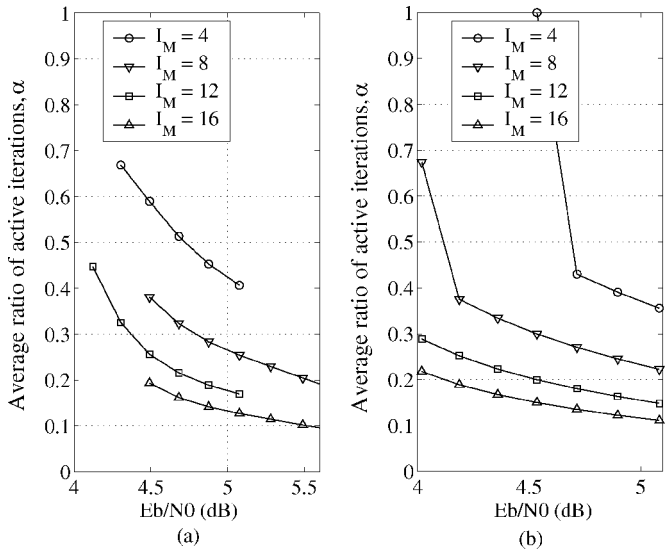


Fig. 10. Ratio of active iterations of (a) a Reed–Solomon based (6, 32)-regular 2048-bit code, and (b) a PEG (4, 15)-regular 660-bit code.

in Fig. 9. The figure shows the graphs for $I_M = 4$, 8, 12 and 16. For example, based on Fig. 10(b), for the code implemented in this work with $I_M = 15$, on average less than three iterations are needed per frame at SNR = 4.3 dB (corresponding to BER $= 10^{-5}$). As will be shown in the results section, by exploiting this behavior and turning off the decoder in the remaining unneeded iterations, the total dynamic power is reduced by 65%.

### C. Hardware Implementation

The remaining task is to efficiently implement early termination in hardware. In other words, to detect that the decoder has converged to a correct codeword. A standard approach is to make final decisions in each VNU at the end of each iteration and then check if all parity constraints are satisfied. This is referred to as syndrome checking and one form of it is implemented in [22] for a decoder with a layered mode belief propagation algorithm. Although straightforward, the conventional syndrome checking has a considerable hardware cost in fully-parallel decoders. This is because in every iteration the hard decision results must be distributed from variable nodes to the destination check nodes where syndrome checking can be performed. This distribution can be done either by dedicating extra hard wires from VNUs to the neighboring CNUs, or by sharing the same wires used for transferring extrinsic messages in a bit-serial time multiplexed fashion. But neither of these approaches are efficient because they either increase the routing complexity by adding global wires or decrease decoding throughput by increasing the number of clock cycles per iteration.

Alternatively, in this work we check the parity of the sign bit of the normal variable-to-check messages that are already required by the decoding iterations. If the parity of the sign bit of all these messages are satisfied, we compute the final hard decision at the beginning of next iteration and then turn off the VNUs and CNUs for the remaining iterations. Although not mathematically equivalent to the standard syndrome checking, we have simulated the two LDPC codes of Fig. 8 with the same set of $10^6$ frames both without and with early termination at $Eb/No$ ranging from 4 dB to 5.1 dB. The simulations show identical performance between the two approaches for these codes.

For the two codes discussed in this paper, our method on average needs one extra iteration to terminate compared with the conventional syndrome checking method. This difference reduces the amount of power savings achieved compared to the conventional syndrome checking. For example, in the 660-bit decoder presented in Section II-C, conventional syndrome checking could have improved the percentage of power savings from 49% to 51% for low-SNR inputs ($Eb/No \approx 3$ dB) and from 66% to 72% for high-SNR inputs ($Eb/No \approx 6$ dB). In spite of the reduced power savings, we have adopted this new termination method for two reasons. First, in contrast to conventional early termination our termination method does not increase the number of VNU-to-CNU wires, nor does it require extra clock cycles per iteration to distribute the hard decision results to the CNUs. Second, this approach requires minimal hardware overhead since most of the calculations are already part of the normal VNU and CNU operations.

Fig. 11 shows the block diagram of a decoder with early termination logic. It is similar to the one in Fig. 5 with a few added blocks: First, all the parity results are ORed. The output of the OR tree is zero only when all the parities are satisfied. Second, a termination logic block generates the proper disable/enable signals for the VNUs and CNUs depending on the value of the OR tree output. If the output of the OR tree is zero, it keeps the VNUs and CNUs disabled for the remaining iterations. Fig. 12 shows the timing diagrams of the decoder, with and without early termination. It shows that the decoding throughput is the same in both cases since the start time for decoding the frames is identical. However, the power consumption is reduced in Fig. 12 because the decoder is turned off as soon as a correct codeword is detected.

The synthesis results show that the added OR tree and the enable/disable functionality required in CNUs and VNUs adds only less than 0.1% and 0.5% to the total decoder gate count, respectively. It should also be noted that no additional logic is required inside the CNUs to generate the XOR-out signals as this value is already available from the sign-calculation block inside the CNUs (Fig. 6).

## IV. Results

### A. A (660, 484) LDPC Decoder

Fig. 13 shows the die photo of the fabricated (660, 484) LDPC decoder. The decoder performs 15 decoding iterations per frame as it was shown in Fig. 8(b) that performing more than 12 iterations results in a negligible BER enhancement. It occupies 7.3 mm$^2$ core area and operates at maximum frequency of 300 MHz with a 1.2 V core supply voltage, which results in a 3.3 Gb/s total throughput. Since the code rate is 0.74, this corresponds to an information throughput of 2.44 Gb/s. The measured BER performance of the decoder matches bit-true simulations. The BER curve is practically identical to the BER graph in Fig. 8(b) for $I_M = 16$.

The total decoder power consumption is shown in Fig. 14 as a function of input SNR at 300 MHz with 1.2 V supply

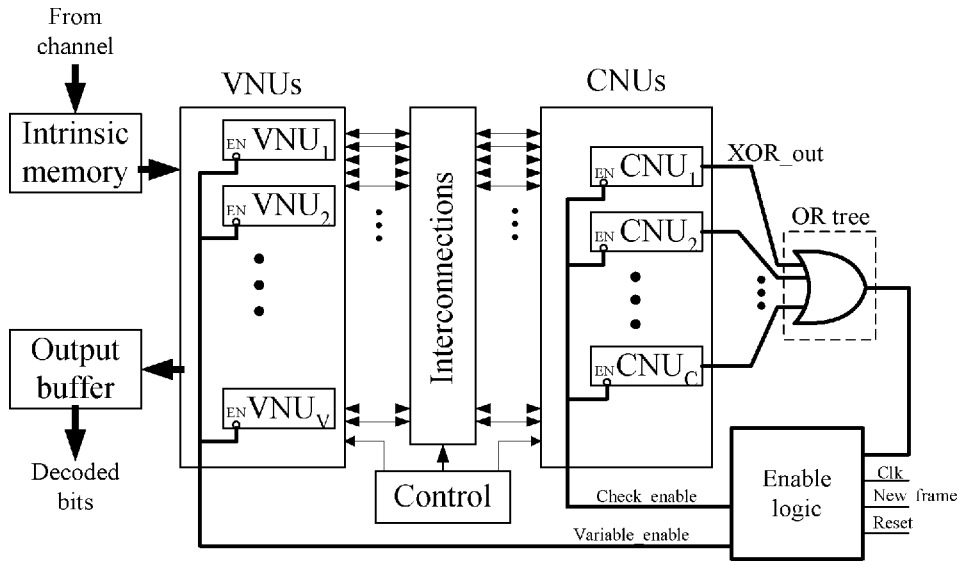


Fig. 11. Fully-parallel iterative LDPC decoder with early termination functionality.

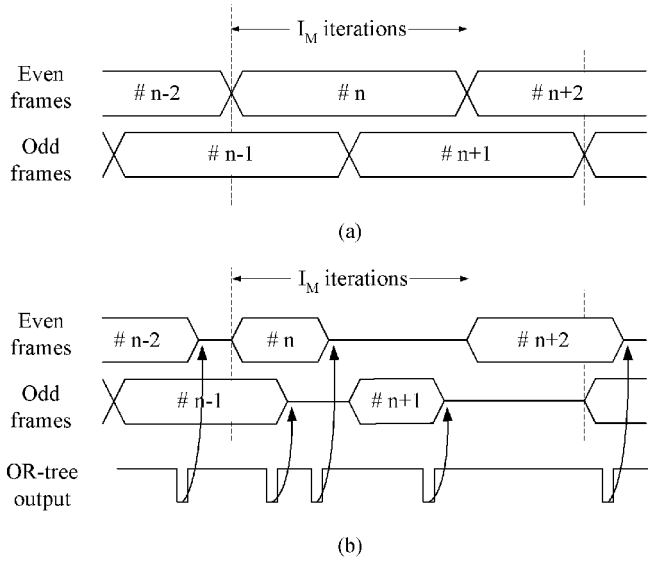


Fig. 12. Block-interlaced decoding timing diagram (a) without early termination, and (b) with early termination.

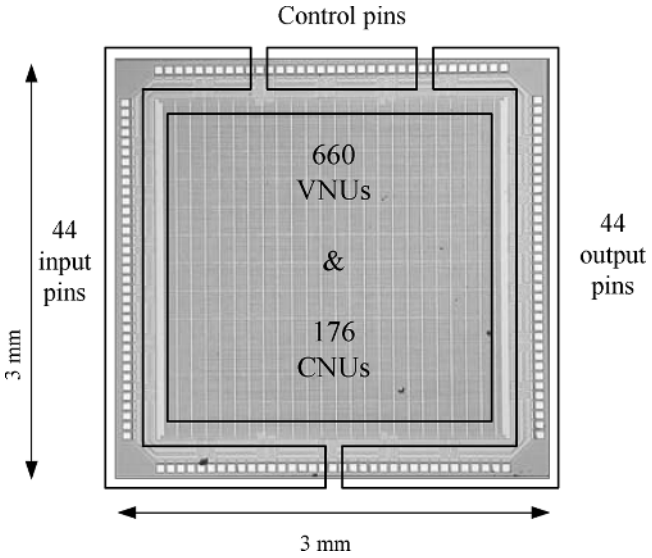


Fig. 13. Decoder die photo.

voltage. The solid line in this graph is directly obtained from measurements. It was observed that approximately 20% of the total power dissipation is due to the clock tree. It was also observed that only less than 1.4 mW (i.e., 0.1% of the total power consumption) is due to leakage current. The graph in Fig. 14 also shows that in contrast to the fully-parallel LDPC decoder in [11], the power consumption is relatively flat for the SNR values of interest in this work. This is mostly because of the bit-serial message-passing and the block interleaving architecture which tend to maintain high switching activity independent of the input SNR.

The power consumption resulting from early termination as proposed in this work is shown by the dotted line in Fig. 14. Since early termination logic was not included in the fabricated prototype, we have calculated the $P'_D$ data points on the dotted line from the $P_D$ data points on the solid line using

$$P'_D = (1+\gamma)(p_c + \alpha(1-p_c))P_D$$

where $\gamma$ accounts for the overhead of the early termination logic, $p_c$ is the fraction of dynamic power attributable to the clock tree, and $\alpha$ is the ratio of active iterations similar to the values plotted in Fig. 10(b). This expression accounts for the fact that early termination does not decrease the dynamic power in the clock tree. As explained in Section III, $\gamma$ for the reported decoder is estimated to be less than 0.006. As also mentioned, our measurements show that $p_c$ is approximately 0.2. The figure shows that early termination reduces the power consumption by between

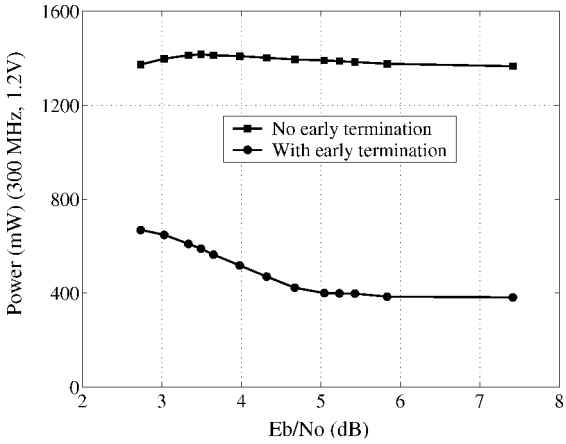


Fig. 14. Decoder power consumption versus input SNR.

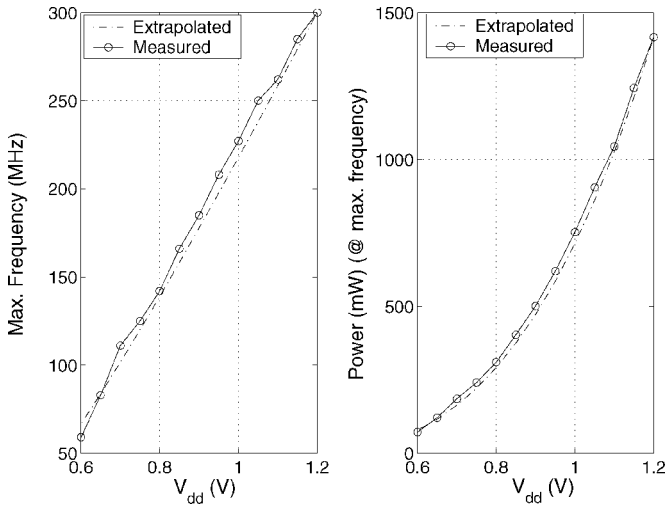


Fig. 15. Effect of supply voltage scaling on maximum frequency and power consumption.

58% and 66% in the practical SNR range of interest between $Eb/No = 4$ dB and $Eb/No = 5.5$ dB.

Fig. 15 shows the effect of supply-voltage scaling on the measured maximum frequency and the total power dissipation at that frequency. The dotted lines are the predicted values based on the MOS square-law equation with $V_t = 0.3$ V. It can be seen that the measured results closely follow the predicted results both for maximum frequency and for the power consumption.

Table I summarizes the characteristics of the fabricated decoder. In Table II, the results from other LDPC decoders reported in literature are listed. The decoder architecture in [11] is fully parallel, whereas the decoders in [23] and [24] are partially parallel. The power and throughput performance comparison between these works is shown in Fig. 16. To take into account the varying number of iterations per frame and the different code rates in the different decoders, the throughputs on the vertical axis are the information throughput normalized to $I_M = 15$ iterations per frame, which is the value used in our decoder. The horizontal axis is the energy efficiency of the decoders in pJ per bit per iteration. These values are obtained by dividing the decoder power consumption by total decoder throughput and the

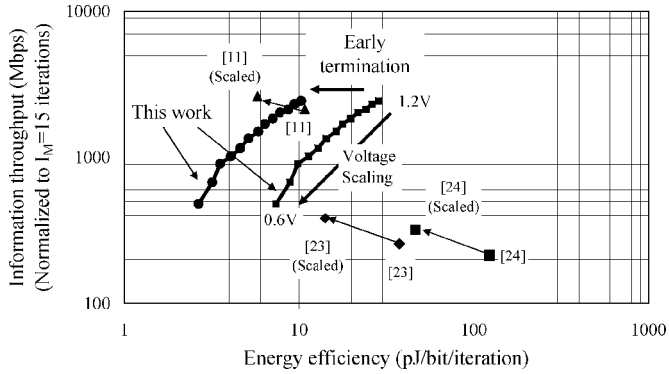


Fig. 16. Comparison with other works. The effect of early shut-down and supply voltage scaling on power consumption is illustrated.

TABLE I
CHARACTERISTICS SUMMARY AND MEASURED RESULTS

| | | | | |
|---|---|---|---|---|
| Process | 0.13-$\mu$m CMOS | | | |
| Architecture | Bit-serial fully-parallel | | | |
| Code construction | Regular 660-bit | | | |
| Code rate | 0.74 | | | |
| Decoding algorithm | Modified min-sum | | | |
| Total area | 9mm$^2$ | | | |
| Core area | 7.3mm$^2$ | | | |
| Gate count | 690 k | | | |
| Core area utilization | 72% | | | |
| Iterations per frame | 15 | | | |
| Message word length | 4 bits | | | |
| Package | QFP160 | | | |
| Supply voltage | 1.2 V | | 0.6 V | |
| Maximum frequency (MHz) | 300 | | 59 | |
| Total throughput (Mbps) | 3300 | | 648 | |
| Information throughput (Mbps) | 2440 | | 480 | |
| Early termination | No | Yes | No | Yes |
| Power @ (Eb/No=4dB) (mW) | 1408 | 518 | 72 | 26.5 |
| Power @ (Eb/No=5.5dB) (mW) | 1383 | 398 | 71 | 20.5 |
| Energy (pJ/bit/iter) (@ Eb/No=4dB) | 28.5 | 10.4 | 7.4 | 2.7 |
| Energy (pJ/bit/iter) (@ Eb/No=5.5dB) | 27.9 | 8.0 | 7.3 | 2.1 |

number of iterations per frame. For the bit-serial decoder presented in this paper, the iteration number of $I_M = 15$ is used when calculating the energy efficiency in all cases (please refer to Fig. 12).

For comparison purposes, we have also included scaled values for area, throughput and energy efficiency in Table II and Fig. 16. The area entries in the brackets in Table II are scaled down quadratically to a 0.13-$\mu$m CMOS process and also scaled linearly to a block length of 660 bits. The throughputs and energy efficiencies are scaled linearly and cubically to 0.13-$\mu$m CMOS process, respectively ([25, Ch. 16]). The comparison graph confirms that fully-parallel decoders provide better energy efficiency and decoding throughput compared to memory-based partially-parallel decoders.

TABLE II
COMPARISON WITH OTHER WORKS

| | [11] | [23] | [24] |
|---|---|---|---|
| Process | 0.16-$\mu$m CMOS | 0.18-$\mu$m CMOS | 0.18-$\mu$m CMOS |
| Architecture | Fully-parallel | Partially-parallel | Partially-parallel |
| Code construction | Irregular 1024-bit | Irregular 600-bit | Regular 2048-bit |
| Code rate | 0.5 | 0.75 | Programmable |
| Total area ($mm^2$) | 52.5 (18.1)[a] | 21.9 (12.6)[a] | 14.3 (2.4)[a] |
| Iterations per frame | 64 | 8 | 10 |
| Supply voltage (V) | 1.5 | 1.8 | 1.8 |
| Maximum frequency (MHz) | 64 | 80 | 125 |
| Total throughput (Mbps) | 1000 | 640 | 640 |
| Information throughput (Mbps) | 500 | 480 | 320 |
| Power (mW) | 690 | 192[b] | 787 |
| Energy (pJ/bit/iter) | 10.9 (5.8)[c] | 37.5 (14.1)[c] | 123 (46.3)[c] |

[a] Scaled linearly to 660-bit code length and quadratically to 0.13-$\mu$m process

[b] The power consumption due to clock tree is not included

[c] Scaled cubically to 0.13-$\mu$m process

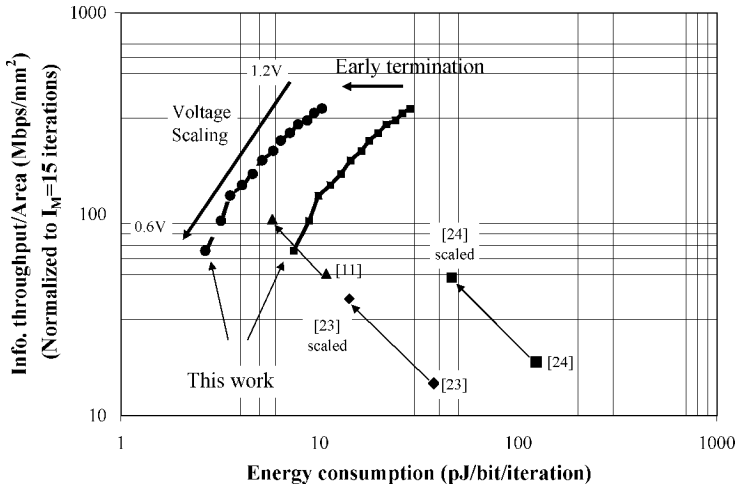


Fig. 17. Comparison with other works with decoder area also reflected on the vertical axis.

The high energy efficiency in [11] can be attributed to its high level of parallelism as predicted in this paper. It can also be explained with the fact that even though the decoder performs 64 iterations on each block, the vast majority of blocks converge in the first few iterations, resulting in minimal switching activity for the remaining iterations. This is in contrast with the bit-serial block-interlaced decoder presented in our work where the switching activity does not scale down with decoder convergence unless an early termination method is applied. Finally, the average variable node degree in [11] is 3.25 compared to the average degree of 4 in our decoder. For two decoders with the same code length and the same code rate, the decoder with lower average node degree computes less messages in each iteration, and hence, consumes less power.

One important dimension which is missing from Fig. 16 is the decoder total silicon area and its routing complexity. For example, although the fully-parallel decoder in [11] has good power and throughput performance, its large area makes it very costly in practice. The bit-serial fully-parallel scheme demonstrated in this work combined with the early termination scheme reduces routing complexity and area while maintaining the throughput and energy efficiency advantages of fully-parallel decoders. Compared to conventional fully-parallel decoders, the logic area is reduced in bit-serial fully-parallel decoders because only 1-bit partial results are generated in each clock cycle. In addition, the reduced routing congestion allows for higher area utilization. This can be observed from the 52.5 $mm^2$ total area (18.1 $mm^2$, if scaled for process and code length) with about 50% area utilization in [11] compared to the 9 $mm^2$ total area with 72% area utilization in our design. This comparison is demonstrated in Fig. 17, which is similar to Fig. 16 except for the vertical axis which is normalized with respect to the decoder area.

With the power reduction achievable by early termination, the decoder consumes only 10.4 pJ/bit/iteration from 1.2 V supply voltage and has a total throughput of 3.3 Gb/s. The

projected lines in the graph show that even further power reductions are achievable if supply voltage scaling is combined with early termination. A minimum of 2.7 pJ/bit/iteration is predicted with a 0.6 V supply voltage operating at 59 MHz and providing 648 Mb/s total throughput. These energy efficiency results even compare favorably with analog decoders which are aimed for energy efficiency. For example, the analog LDPC decoder reported in [5] consumes 0.83 nJ/bit (compared to less than 0.43 nJ/bit in this work) with and has a throughput of only 6 Mb/s.

### B. A (2048,1723) LDPC Decoder

To demonstrate usability of the proposed bit-serial architecture for longer codes, we have synthesized a bit-serial fully-parallel decoder for the (6,32)-regular (2048, 1723) LDPC code as featured in the 10GBase-T standard. The decoder uses $q = 4$ bit quantization for LLR messages and performs $I_M = 8$ iterations per frame.

The decoder is synthesized in a 90 nm CMOS library using Synopsys Design Compiler. It occupies 9.8 mm$^2$ of logic area (2.23 M equivalent NAND gates) and has a maximum operating clock frequency of 250 MHz. This corresponds to a total decoding throughput of 16 Gb/s which is significantly higher than that required by the 10GBase-T standard. This throughput margin can be traded for significantly lower power dissipation by reducing the supply voltage and/or for better BER performance by increasing the word length of the LLR messages.

## V. Conclusion

We have discussed two techniques to improve the power-efficiency of LDPC decoders. First, we analyzed how the increased parallelism coupled with a reduced supply voltage is a particularly effective technique to reduce the power consumption of LDPC decoders due to their inherent parallelism. Second, we proposed a scheme to efficiently implement early termination of the iterative decoding to further reduce the power consumption. In spite of their superior speed and energy efficiency, it is known that their large area and complex interconnect network limit the scalability of conventional fully-parallel LDPC decoders [11]. The bit-serial fully-parallel architecture proposed in this work addresses these concerns by reducing both interconnect complexity and logic area. Although the needs of applications specifying long block-length and low-throughput LDPC codes (such as DVB-S2 [4]) can be met with lower levels of parallelism (e.g., [24], [26], [23], [22]), a fully parallel decoder is preferable for applications such as 10GBase-T which use a medium-size LDPC code (e.g., 2048 bit) and require multi-Gb/s decoding throughput. We reported on a fabricated 0.13-$\mu$m CMOS bit-serial fully-parallel LDPC decoder and show the effect of the proposed techniques. The decoder has a 3.3 Gb/s throughput with a nominal 1.2 V supply and performs within 3 dB of the Shannon limit at a BER of $10^{-5}$. With more than 60% power saving achieved by early termination, the decoder consumes 10.4 pJ/bit/iteration at $Eb/No = 4$ dB. Coupling early termination with supply voltage scaling results in even lower consumption of 2.7 pJ/bit/iteration with 648 Mb/s total decoding throughput. Using a similar bit-serial fully-parallel architecture, we also reported on a synthesized decoder for (6, 32)-regular (2048, 1723) LDPC code specified in 10GBase-T standard.

## Acknowledgment

The authors would like to thank the editors and reviewers of IEEE Transactions on VLSI Systems for their valuable comments in the initial stages of this submission.

## References

[1] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.
[2] *LAN/MAN CSMA/CD Access Method*, IEEE 802.3 Standard [Online]. Available: http://standards.ieee.org/getieee802/802.3.html
[3] *Mobile WirelessMAN*, IEEE 802.16e Standard, 2005 [Online]. Available: http://standards.ieee.org/getieee802/download/802.16e-2005.pdf
[4] Draft, European Telecommunication Standards Inst., EN 302 307 V1.1.1, 2004–2006.
[5] S. Hemati, A. H. Banihashemi, and C. Plett, "A 0.18-$\mu$m CMOS analog min-sum iterative decoder for a (32,8) low-density parity-check (LDPC) code," *IEEE J. Solid-State Circuits*, vol. 41, no. 11, pp. 2531–2540, Nov. 2006.
[6] V. C. Gaudet and P. G. Gulak, "A 13.3-Mb/s 0.35-$\mu$m CMOS analog turbo decoder IC with a configurable interleaver," *IEEE J. Solid-State Circuits*, vol. 38, no. 11, pp. 2010–2015, Nov. 2003.
[7] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linkoping Univ., Linkoping, Sweden, 1996.
[8] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, Apr. 1992.
[9] B. H. Calhoun, A. Wang, and A. Chandrakasan, "Modeling and sizing for minimum energy operation in subthreshold circuits," *IEEE J. Solid-State Circuits*, vol. 40, no. 9, pp. 1778–1786, Sep. 2005.
[10] S.-J. Lee, N. R. Shanbhag, and A. C. Singer, "A low-power VLSI architecture for turbo decoding," in *Proc. Int. Symp. Low Power Electronics and Design (ISLPED'03)*, Aug. 2003, pp. 366–371.
[11] A. J. Blanksby and C. J. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check decoder," *IEEE J. Solid-State Circuits*, vol. 37, no. 3, pp. 404–412, Mar. 2002.
[12] T. Zhang and K. K. Parhi, "Joint (3, k)-regular LDPC code and decoder/encoder design," *IEEE Trans. Signal Process.*, vol. 52, no. 4, pp. 1065–1079, Apr. 2004.
[13] M. M. Mansour and N. R. Shanbhag, "High-throughput LDPC decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 6, pp. 976–996, Dec. 2003.
[14] D. E. Hocevar, "LDPC code construction with flexible hardware implementation," in *Proc. IEEE Int. Conf. Communications (ICC'03)*, May 2003, vol. 4, pp. 2708–2712.
[15] D. E. Hocevar, "Efficient encoding for a family of quasi-cyclic LDPC codes," in *Proc. IEEE Global Telecommunications Conf. (GlobeCom'03)*, Dec. 2003, vol. 7, pp. 3996–4000.
[16] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.
[17] A. Darabiha, A. Chan Carusone, and F. R. Kschischang, "A bit-serial approximate min-sum LDPC decoder and FPGA implementation," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS'06)*, Kos, Greece, May 2006, pp. 149–152.
[18] E. Zimmermann, G. Fettweis, P. Pattisapu, and P. K. Bora, "Reduced complexity LDPC decoding using forced convergence," in *Proc. Int. Symp. Wireless Personal Multimedia Communications (WPMC'04)*, Padova, Italy, Sep. 2004, vol. 3, pp. 243–246, WA2-2.
[19] L. W. A. Blad and O. Gustafsson, "An early decision decoding algorithm for LDPC codes using dynamic thresholds," in *Proc. Eur. Conf. Circuit Theory and Design*, Aug. 2005, pp. III/285–III/288.
[20] E. Zimmermann, P. Pattisapu, and G. Fettweis, "Bit-flipping post-processing for forced convergence decoding of LDPC codes," presented at the Eur. Signal Processing Conf., Antalya, Turkey, 2005.
[21] S. ten Brink, "Convergence of iterative decoding," *Electron. Lett.*, vol. 35, no. 10, pp. 806–808, May 1999.
[22] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proc. IEEE Workshop on Signal Processing Systems*, Oct. 2004, pp. 107–112.

[23] H.-Y. Liu, C.-C. Lin, Y.-W. Lin, C.-C. Chung, K.-L. Lin, W.-C. Chang, L.-H. Chen, H.-C. Chang, and C.-Y. Lee, "A 480Mb/s LDPC-COFDM-based UWB baseband transceiver," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC'05)*, Feb. 2005, vol. 1, pp. 444, 609.

[24] M. M. Mansour and N. R. Shanbhag, "A 640-Mb/s 2048-bit programmable LDPC decoder chip," *IEEE J. Solid-State Circuits*, vol. 41, no. 3, pp. 684–698, Mar. 2006.

[25] B. Razavi, *Design of Analog CMOS Integrated Circuits*. Boston, MA: McGraw-Hill, 2001.

[26] P. Urard, E. Yeo, L. Paumier, P. Georgelin, T. Michel, V. Lebars, E. Lantreibecq, and B. Gupta, "A 135Mb/s DVB-S2 compliant codec based on 64800b LDPC and BCH codes," in *IEEE Int. Solid-State Circuits Conf. (ISSCC'05) Dig. Tech. Papers*, Feb. 2005, pp. 446, 609.

**Ahmad Darabiha** (S'97) received the B.A.Sc. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1998, and received the M.A.Sc. and Ph.D. degrees in electrical engineering from the University of Toronto, Toronto, Ontario, Canada, in 2003 and 2008, respectively.

His main research interests include VLSI implementation of digital communications and digital signal processing algorithms. In particular, he has been working on algorithms, architectures and VLSI implementations for high-throughput and low-power error correction decoders.

**Anthony Chan Carusone** (S'96–M'02) received the B.A.Sc. and Ph.D. degrees from the University of Toronto, Toronto, Ontario, Canada, in 1997 and 2002 respectively, during which time he received the Governor-General's Silver Medal.

Since 2001, he has been with the Department of Electrical and Computer Engineering at the University of Toronto, where he is currently an Associate Professor.

Dr. Carusone was named an Ontario Distinguished Researcher and Canada Research Chair in Integrated Systems in 2002. He was a co-author of the best paper at the 2005 Compound Semiconductor Integrated Circuits Symposium and best student paper at the 2007 Custom Integrated Circuits Conference. He is a past chair of the Analog Signal Processing Technical Committee for the IEEE Circuits and Systems Society, a member of the technical program committee for the IEEE Custom Integrated Circuits Conference, and Deputy Editor-in-Chief of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, PART II: EXPRESS BRIEFS.

**Frank R. Kschischang** (S'83–M'91–SM'00–F'06) received the B.A.Sc. (Honors) degree from the University of British Columbia, Vancouver, BC, Canada, in 1985, and the M.A.Sc. and Ph.D. degrees from the University of Toronto, Toronto, ON, Canada, in 1988 and 1991, respectively, all in electrical engineering.

During 1997–1998, he spent a sabbatical year as a Visiting Scientist at the Massachusetts Institute of Technology, Cambridge, and in 2005 he was a Guest Professor at the Swiss Federal Institute of Technology (ETH), Zurich. Currently, he is a Professor and Canada Research Chair in the Department of Electrical and Computer Engineering, University of Toronto, where he has been a faculty member since 1991. His research interests include coding techniques, primarily on soft-decision decoding algorithms, trellis structure of codes, codes defined on graphs, and iterative decoders and in the application of coding techniques to wireline, wireless and optical communication systems.

Dr. Kschischang served as Associate Editor for Coding Theory of the IEEE TRANSACTIONS ON INFORMATION THEORY from 1997 to 2000 and also served as the Technical Program Co-Chair for the 2004 IEEE International Symposium on Information Theory. He is a recipient of the Ontario Premier's Research Excellence Award.