An Efficient 10GBASE-T Ethernet LDPC Decoder Design with Low Error Floors

Zhengya Zhang, *Member, IEEE,* Venkat Anantharam, *Fellow, IEEE,*
Martin J. Wainwright, *Member, IEEE,*
and Borivoje Nikolić, *Senior Member, IEEE*

Abstract

A grouped-parallel low-density parity-check (LDPC) decoder is designed for the (2048,1723) Reed-Solomon-based LDPC (RS-LDPC) code suitable for 10GBASE-T Ethernet. A two-step decoding scheme reduces the wordlength to 4 bits while lowering the error floor to a 10^{-14} BER. The proposed post-processor is conveniently integrated with the decoder adding minimal area and power. The decoder architecture is optimized by groupings so as to localize irregular interconnects and regularize global interconnects and the overall wiring overhead is minimized. The 5.35 mm^2, 65nm CMOS chip achieves a decoding throughput of 47.7 Gb/s. With scaled frequency and voltage, the chip delivers a 6.67 Gb/s throughput necessary for 10GBASE-T while dissipating 144 mW of power.

Index Terms

Low-density parity-check (LDPC) code; message-passing decoding; iterative decoder; error floors.

This research was supported in part by NSF CCF grant no. 0635372, Marvell Semiconductor, and Intel Corporation through a UC MICRO grant. The design infrastructure was developed with the support of Center for Circuit & System Solutions (C2S2) Focus Center, one of five research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation program. The grant NSF CNS RI-0403427 provided the computing infrastructure and ST Microelectronics donated the chip fabrication.

Z. Zhang was with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley and is now with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, 48109 USA (e-mail: zhengya@eecs.umich.edu).

V. Anantharam, M. J. Wainwright, and B. Nikolić are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, 94720 USA e-mail: ({ananth, wainwrig, bora}@eecs.berkeley.edu).

Manuscript received August 24, 2009.

I. INTRODUCTION

Low-density parity-check (LDPC) codes have been demonstrated to perform very close to the Shannon limit when decoded iteratively using message-passing algorithms [1]–[4]. A wide array of the latest communication and storage systems have chosen LDPC codes for forward error correction in applications including digital video broadcasting (DVB-S2) [5], [6], 10 Gigabit Ethernet (10GBASE-T) [7], broadband wireless access (WiMax) [8], wireless local area network (WiFi) [9], deep-space communications [10], and magnetic storage in hard disk drives [11]. The adoption of the capacity-approaching LDPC codes is, at least in theory, one of the keys to achieving a lower transmission power for a more reliable communication.

There is a challenge in implementing high-throughput LDPC decoders with a low area and power on a silicon chip for practical applications. The intrinsically-parallel message-passing decoding algorithm relies on the message exchange between variable processing nodes (VN) and check processing nodes (CN) in the graph defined by the **H** matrix. A direct mapping of the interconnection graph causes large wiring overhead and low area utilization. In the first silicon implementation of a fully parallel decoder, Blanksby and Howland reported that the size of the decoder was determined by routing congestion and not by the gate count [12]. Even with optimized floor plan and buffer placement technique, the area utilization rate is only 50%.

Architectures with lower parallelism can be attractive, as the area efficiency can be improved. In the paper [13], the **H** matrix is partitioned: partitions are time-multiplexed and each partition is processed in a fully parallel manner. With structured codes, the routing can be further simplified. Examples include the decoders for DVB-S2 standard [14], [15], where the connection between memory and processors is realized using Barrel shifters. A more compact routing scheme, only for codes constructed with circulant **H** matrices, is to fix the wiring between memory and processors while rotating data stored in shift registers [16]. The more generic and most common partially-parallel architecture is implemented in segmented memories to increase the access bandwidth and the schedules are controlled by lookup tables. Architectures constructed this way permit reconfigurability, as demonstrated by a WiMAX decoder [17].

Solely relying on architecture transformation could be limiting in producing the optimal designs. Novel schemes have been proposed in achieving the design specification with no addition (or even a reduction) of the architectural overhead. In the work [18], a layered decoding

schedule was implemented by interleaving check and variable node operations in order to speed up convergence and increase throughput. This scheme costs additional processing and a higher power consumption. Other authors [19] have used a bit-serial arithmetic to reduce the number of interconnects by a factor of the wordlength, thereby lowering the wiring overhead in a fully parallel architecture. This bit-serial architecture was demonstrated for a small LDPC code with a block length of 660. More complex codes can still be difficult due to the poor scalability of global wires.

Aside from the implementation challenges, LDPC codes are not guaranteed to perform well in every application either. Sometimes the excellent error-correction performance of LDPC codes is only observed up until a moderate bit error rate (BER); at a lower BER, the error curve often changes its slope, manifesting a so-called error floor [20]. With communication and storage systems demanding data rates up to Gb/s, relatively high error floors degrade the quality of service. To prevent such degradation, transmission power is raised or a more complex scheme, such as an additional level of error-correction coding [5], is created. These approaches increase the power consumption and complicate the system integration.

This work implements a post-processing algorithm that utilizes the graph-theoretic structure of LDPC code [21], [22]. The post-processing approach is based on a message-passing algorithm with selectively-biased messages. As a result, it can be seamlessly integrated with the message-passing decoder. Results show performance improvement of orders of magnitude at low error rates after post-processing even with short wordlengths. The wordlength reduction permits a more compact physical implementation.

In formulating the hardware architecture of a high-throughput decoder, a grouping strategy is applied in separating irregular local wires from regular global wires. The post-processor is implemented as a small add-on to each local processing element without adding external wiring, thus the area penalty is kept minimal. A low wiring overhead enables a highly parallel decoder design that achieves a very high throughput. Frequency and voltage scaling can be applied to improve power efficiency if a lower throughput is desired.

The remainder of this paper is organized as follows. Section II introduces the LDPC code and the decoding algorithm. Emphasis is placed on LDPC codes constructed in a structured way and its implication on the decoder architecture. In Section III, hardware emulation is applied in choosing the decoding algorithm and wordlength. In particular, the post-processing algorithm is

demonstrated to achieve an excellent decoding performance at a very short wordlength of 4 bits. In Section IV, the architecture of the chip is determined based on a set of experiments to explore how architectural grouping affects implementation results. Section V explains individual block designs and Section VI presents steps in optimizing the overall area and power efficiencies. The performance and power measurements of the fabricated test chip are presented in Section VII.

II. BACKGROUND

A low-density parity-check code is a linear block code, defined by a sparse $M \times N$ parity check matrix $\mathbf{H}$ where N represents the number of bits in the code block (block length) and M represents the number of parity checks. An example of the $\mathbf{H}$ matrix of an LDPC code is shown in Fig. 1a. The $\mathbf{H}$ matrix can be represented graphically using a factor graph as in Fig. 1b, where each bit is represented by a variable node and each check is represented by a factor (check) node. An edge exists between the variable node i and the check node j if $\mathbf{H}(j,i) = 1$.

A. Decoding Algorithm

Low-density parity-check codes are usually iteratively decoded using the belief propagation algorithm, also known as the message-passing algorithm [1]. The message-passing algorithm operates on a factor graph, where soft messages are exchanged between variable nodes and check nodes. The algorithm can be formulated as follows: in the first step, variable nodes x_i are initialized with the prior log-likelihood ratios (LLR) defined in (1) using the channel outputs y_i, where σ^2 represents the channel noise variance. This formulation assumes the information bits take on 0 and 1 with equal probability.

$$L^{pr}(x_i) = \log \frac{\Pr(x_i = 0 \mid y_i)}{\Pr(x_i = 1 \mid y_i)} = \frac{2}{\sigma^2} y_i, \tag{1}$$

The variable nodes send messages to the check nodes along the edges defined by the factor graph. The LLRs are recomputed based on the parity constraints at each check node and returned to the neighboring variable nodes. Each variable node then updates its decision based on the channel output and the extrinsic information received from all the neighboring check nodes. The marginalized posterior information is used as the variable-to-check message in the next iteration.

1) Sum-Product Algorithm: The sum-product algorithm is a common form of the message-passing algorithm. A simplified illustration of which is shown in Fig. 2a. The block diagram is for one slice of the factor graph showing a round trip from a variable node to a check node back to the same variable node as highlighted in the Fig. 2b. Variable-to-check and check-to-variable messages are computed using equations (2) and (3), where $\Phi(x) = -\log\left(\tanh\left(\frac{1}{2}x\right)\right), x \geq 0$. The messages q_{ij} and r_{ij} refer to the variable-to-check and check-to-variable messages, respectively, that are passed between the ith variable node and the jth check node. In representing the connectivity of the factor graph, $Col[i]$ refers to the set of all the check nodes adjacent to the ith variable node and $Row[j]$ refers to the set of all the variable nodes adjacent the jth check node. The posterior LLR is computed in each iteration using the update (4). A hard decision is made based on the posterior LLR in every iteration. The iterative decoding algorithm is allowed to run until the hard decisions satisfy all the parity check equations or when an upper limit on the iteration number is reached, whichever occurs earlier.

$$L(q_{ij}) = \sum_{j' \in Col[i] \backslash j} L(r_{ij'}) + L^{pr}(x_i), \tag{2}$$

$$L(r_{ij}) = \Phi^{-1}\left(\sum_{i' \in Row[j] \backslash i} \Phi(|L(q_{i'j})|)\right)\left(\prod_{i' \in Row[j] \backslash i} \mathrm{sgn}(L(q_{i'j}))\right). \tag{3}$$

$$L^{ps}(x_i) = \sum_{j' \in Col[i]} L(r_{ij'}) + L^{pr}(x_i), \tag{4}$$

2) Min-Sum Approximation: Equation (3) can be simplified by observing that the magnitude of $L(r_{ij})$ is usually dominated by the minimum $|L(q_{i'j})|$ term, and thus this minimum term can be used as an approximation of the magnitude of $L(r_{ij})$, as shown in the papers [23], [24]. The magnitude of $L(r_{ij})$ computed using such min-sum approximation is usually overestimated and correction terms are introduced to reduce the approximation error. The correction can be in the form of an offset [25], [26], shown as β in the update (5).

$$L(r_{ij}) = \max\left\{\min_{i' \in Row[j] \backslash i} |L(q_{i'j})| - \beta, 0\right\} \prod_{i' \in Row[j] \backslash i} \mathrm{sgn}(L(q_{i'j})). \tag{5}$$

3) Reordered Schedule: The above equations can also be rearranged by taking into account the relationship between consecutive decoding iterations. A variable-to-check message of iteration n can be computed by subtracting the corresponding check-to-variable message from the posterior LLR of iteration $n-1$ as in (6), while the posterior LLR of iteration n can be computed by updating the posterior LLR of the previous iteration with the check-to-variable message of iteration n, as in (7).

$$L_n(q_{ij}) = L_{n-1}^{ps}(x_i) - L_{n-1}(r_{ij}), \tag{6}$$

$$L_n^{ps}(x_i) = L_{n-1}^{ps}(x_i) - L_{n-1}(r_{ij}) + L_n(r_{ij}), j \in Col[i]. \tag{7}$$

B. Structured LDPC Codes

A practical high-throughput LDPC decoder can be implemented in a fully parallel manner by directly mapping the factor graph onto an array of processing elements interconnected by wires. Each variable node is mapped to a variable processing node (VN) and each check node is mapped to a check processing node (CN), such that all messages from variable nodes to check nodes and then in reverse are processed concurrently. Practical high-performance LDPC codes commonly feature block lengths on the order of 1kb and up to 64kb, requiring a large number of VNs. The ensuing wiring overhead poses a substantial obstacle towards efficient silicon implementations.

Structured LDPC codes of moderate block lengths have received more attention in practice recently because they prove amenable for efficient decoder architectures and recent published standards have adopted such LDPC codes [7]–[9]. The **H** matrices of these structured LDPC codes consist of component matrices, each of which is, or closely resembles, a permutation matrix or a zero matrix. Structured codes open the door to a range of efficient high-throughput decoder architectures by taking advantage of the regularity in wiring and data storage. In this work, a highly parallel LDPC decoder design is demonstrated for a (6,32)-regular (2048,1723) RS-LDPC code. This particular LDPC code has been adopted for the forward error correction in the IEEE 802.3an 10GBASE-T standard [7], which governs the operation of 10 Gigabit Ethernet over up to 100 m of CAT-6a unshielded twisted-pair (UTP) cable. The **H** matrix of this code contains M = 384 rows and N = 2048 columns. This matrix can be partitioned into 6 row groups and 32 column groups of 64 $\times$ 64 permutation submatrices.

III. EMULATION-BASED STUDY

An FPGA-based hardware emulation has been used to initially investigate the low error rate performance of this code, and it has been discovered that a class of (8,8) absorbing-set errors dominate the error floors [22], [27]. A subgraph illustrating the (8,8) absorbing set is shown in Fig. 3, representing a substructure of the factor graph associated with the LDPC code. Consider a state with all eight variable nodes of an (8,8) absorbing set in error – a state that cannot be decoded successfully by a message-passing decoder because the variable nodes that constitute the absorbing set reinforce the incorrect values among themselves through the cycles in the graph. More precisely, each variable node receives one message from a unsatisfied check node attempting to correct the error, which is overpowered by five messages from satisfied check nodes that reinforce the error.

It was also found that a sum-product decoder implementation tends to incur excessive numerical saturation due to the finite-wordlength approximation of the Φ functions. The reliability of messages is reduced with each iteration until the message-passing decoder is essentially performing majority decoding, and the effect of absorbing sets is worsened. In comparison, an offset min-sum decoder implementation eliminates the saturation problem due to the Φ functions. A 6-bit offset min-sum decoder achieves a 0.5 dB SNR gain compared to a 6-bit sum-product decoder as seen in Fig. 4. Despite the extra coding gain and lower error rate performance of the offset min-sum decoder, its error floor emerges at a BER level of 10^{-11}, which still renders this implementation unacceptable for 10GBASE-T Ethernet that requires an error-free operation below the BER level of 10^{-12} [7].

Brute-force performance improvement requires a longer wordlength, though the performance gain with each additional bit of wordlength diminishes as the wordlength increases over 6 bits. Further improvement should rely on adapting the message-passing algorithm to combat the effect due to absorbing sets. A two-step decoding strategy can be applied: in the first step, a regular message-passing decoding is performed. If it fails, the second step is invoked to perform post-processing [21], [22]: the unsatisfied checks are marked and the messages via these unsatisfied checks are strengthened and/or the messages via the satisfied checks are weakened. Such a message biasing scheme introduces a systematic perturbation to the local minimum state. Message biasing is followed by a few more iterations of regular message-passing decoding

until post-processing converges or a failure is declared. The post-processor proves to be highly effective: a 4-bit offset min-sum decoder, aided by the post-processor, surpasses the performance of a 6-bit decoder below the BER level of 10^{-11}.

IV. Architectural Design

A high decoding throughput requires a high degree of parallelism and a large memory access bandwidth. With the structured RS-LDPC code, VNs and CNs can be grouped and wires bundled between the node groups, as illustrated in Fig. 5b for the **H** matrix in Fig. 5a. Irregular wires are sorted within the group, similar to a routing operation. The fully parallel architecture with all the routers expanded is shown in Fig. 5b.

Even with node grouping and wire bundling, the fully parallel architecture might not be the most efficient for a complex LDPC decoder. To reduce the level of parallelism, individual routers are combined and routing operations are time-multiplexed. Fig. 5c shows how the two routers in every column are combined, leading to the creation of local units – 3 variable node groups (VNG) and 1 check node group (CNG), that encapsulate irregular local wiring, and wires outside of local units are regular and structured. The number of local units determines the level of parallelism. A less parallel design uses fewer local units, but each one is more complex as it needs to encapsulate more irregular wiring to support multiplexing; a highly parallel design uses more local units and each one is simpler, but the amount of global wiring, though regular and structured, would increase accordingly.

To explore the optimal level of parallelism targeting a lower wiring overhead, a new metric, the area expansion factor, or AEF is defined as the ratio between the area of the complete system and the total area of stand-alone component nodes. A few selected decoder architectures were investigated for the (2048,1723) RS-LDPC code, listed in Table I with increasing degrees of parallelism from top to bottom. The AEF of the designs is shown in Fig. 6 with the horizontal axis displaying the approximate decoding throughput. The upward-facing AEF curve features a flat middle segment at the 16VNG-1CNG architecture and the 32VNG-1CNG architecture. Designs positioned in the flat segment achieve a balance of throughput and area – doubling the throughput from 16VNG-1CNG to 32VNG-1CNG requires almost twice as many processing nodes, but the AEF remains almost constant, so the area doubles. In the region where the AEF is constant, the average global wiring overhead is constant and it is advantageous to increase

the degree of parallelism for a higher throughput.

The AEF plot alone actually suggests a more serial architecture, e.g., 8VNG-1CNG, as it incurs the lowest average global wiring overhead. However, the total on-chip signal wire length of the 8VNG-1CNG architecture is still significant – an indication of the excessive local wiring in supporting time-multiplexing. To supplement the AEF curve, the incremental wiring overhead (measured in on-chip signal wire length) per additional processing node is shown in Fig. 6. As the degree of parallelism increases from 8VNG-1CNG, the local wiring should be decreasing more quickly while the global wiring increasing slowly, resulting in a decrease in the incremental wiring overhead. The incremental wiring overhead eventually reaches the minimum with the 32VNG-1CNG architecture. This minimum corresponds to the balance of local wiring and global wiring. Any further increase in the degree of parallelism causes a significant increase in the global wiring overhead. The 32VNG-1CNG architecture is selected for implementation.

V. Functional Design

A. Components

The 32VNG-1CNG decoder architecture consists of 2,048 VNs, representing the majority of the chip area. The block diagram of the VN for the offset min-sum decoder is illustrated in Fig. 7. Each VN sequentially sends six variable-to-check messages and receives six returning messages from CNs per decoding iteration, as illustrated in the pipeline chart of Fig. 8. Three storage elements are allocated: the posterior LLR memory which accumulates the check-to-variable messages, the extrinsic memory which stores the check-to-variable messages in a shift register, and the prior LLR memory.

Each VN participates in the operations in six horizontal rows. In each operation, a variable-to-check message is computed by subtracting the corresponding check-to-variable message (of the previous iteration) from the posterior LLR (of the previous iteration) as in equation (6) (refer to Fig. 7). The variable-to-check message is converted to the sign-magnitude form before it is sent to the VNG routers destined for a CN. The returning messages to the VN could be from one of the six CNs. A multiplexer selects the appropriate message based on a schedule. The check node operation described in equation (5) is completed in two steps: 1) the CN computes the minimum (min_1) and the second minimum (min_2, $min_2 \geq min_1$) among all the variable-to-check messages received from the neighboring VNs, as well as the product of the signs (prd)

of these messages; 2) the VN receives min_1, min_2, and prd, computes the marginals, which is followed by the conversion to the two's complement format and the offset correction. The resulting check-to-variable message is accumulated serially to form the posterior LLR as in equation (7). Hard decisions are made in every iteration.

Post-processing is enabled in the VN in three phases: pre-biasing, biasing, and follow-up. In the pre-biasing phase (one iteration before post-processing), *tag* is enabled (refer to Fig. 7). If a parity check is not satisfied, as indicated by prd, the edges emanating from the unsatisfied check node are tagged by marking the messages on these edges, and the variable nodes neighboring the unsatisfied check are also tagged. In the biasing phase, *post-proc* is enabled (refer to Fig. 7). Tags are inspected, such that if a tagged variable node sends a message to a satisfied check node, the magnitude of this message is weakened with the intention of reducing the reinforcement among the possibly incorrect variable nodes. Finally in the follow-up phase, regular message passing decoding is performed for a few iterations to clean up the possible errors after message biasing.

The VNG routers follow the structure shown in Fig. 5c with 64 6:1 multiplexers. The CN is designed as a compare-select tree. The 32 input variable-to-check messages are sorted in pairs, followed by four stages of 4-to-2 compare-selects. The outputs min_1, min_2, and product of signs, prd are buffered and broadcast to the 32 VNGs.

B. Pipeline

A 7-stage pipeline is designed as in Fig. 8. One stage is allocated for the VN in preparing variable-to-check message, and one stage for the delay through the VNG routers. Three stages are dedicated to the compare-select tree in the CN – one for the sorting and the first-level compare-select, one for the following two levels of compare-select, and one for the final compare-select as well as the fanout. Two stages are set aside for processing the return messages from the CN – one for preparing the check-to-variable message and one for accumulating the check-to-variable message.

With the 7-stage pipeline and the minimum 2.5-ns clock period for the CMOS technology being used, a decoding throughput of 6.83 Gb/s can be achieved, assuming 10 decoding iterations. Trial placement and routing are performed to identify the critical paths and characterize the global wiring delays. The clock period is set such that it accommodates the longest wire delay and the wire's driving or receiving gate delay with a sufficient margin. A deeper-pipelined design would

require wire pipelining and an increase in area and power due to additional pipeline registers.

The two-iteration pipeline diagram is shown in Fig. 9. Due to the data dependency between consecutive iterations, a 6-cycle stall is inserted between iterations such that the posterior LLR can be fully updated in the current iteration before the next iteration starts. The stall means that the first VC stage (refer to Fig. 9) of iteration $i+1$ has to wait for 6 cycles for the last PS stage of iteration i to complete. No useful work is performed during the stall cycles, so the efficiency is lower. The efficiency would be reduced even more if a turbo decoding schedule (also known as a layered schedule) [28] or a shuffled schedule [29] is applied to such a pipeline, where data dependency arises between layers within an iteration. If more pipeline stalls are inserted to resolve the dependency, the efficiency is degraded to as low as 1/7, defeating the purpose of a slightly higher convergence rate achieved with these schedules.

C. Density

The optimal density depends on the tradeoff between routability and wire distance. A lower-density design can be easily routed, but it occupies a larger area and wires need to travel longer distances. On the other hand, a high-density design cannot be routed easily, and the clock frequency needs to be reduced as a compromise. Table II shows that timing closure can be achieved with initial densities of 70% to 80%. The total signal wire length decreases with increasing density due to the shorter wire distances even with increasing wire counts. An initial density above 80% results in routing difficulty and the maximum clock frequency has to be reduced to accomodate longer propagation delays. To maximize density without sacrificing timing, an 80% initial density is selected.

VI. Area and Power Optimizations

The block diagram of the complete decoder is shown in Fig. 10. Steps of area, performance, power, and throughput improvements of this decoder are illustrated in Fig. 11a and 11b based on synthesis, placement and routing results reported by CAD tools at the worst-case corner of the ST 65nm low-power CMOS standard cell library at 0.9 V supply voltage and temperature of 105°. The baseline design is a 6-bit sum-product decoder. It occupies 6.83 mm^2 of core area and consumes 1.38 W of power to deliver the 6.68 Gb/s throughput (assuming 8 decoding iterations)

at the maximum 310 MHz clock frequency. This implementation incurs an error floor at a BER level of 10^{-11}.

FPGA emulation shows that the 6-bit sum-product decoder can be replaced by a 6-bit offset min-sum decoder to gain 0.5 dB in SNR. The core area increases to 7.15 mm^2 due to additional routing required to send both min_1 and min_2 from CN to VN and this overhead is reflected in the 15.6% increase in wiring and a lower clock frequency of 300 MHz. Despite the area increase, the offset min-sum decoder consumes less power at 1.03 W – a saving attributed to the reduction in dynamic power in the CN design. At high SNR levels or when decoding approaches convergence, the majority of the messages are saturated and the wires in a compare-select tree do not switch frequently, thus consuming less power.

To reduce the area and power further, the wordlength of the offset min-sum decoder is reduced from 6 bits to 4 bits. Wordlength reduction cuts the total wire length by 41.2%, shrinks the core area by 37.9% down to 4.44 mm^2. With a reduced wiring overhead, the maximum clock frequency can be raised to 400 MHz, reaching a 8.53 Gb/s throughput while consuming only 690 mW. Wordlength reduction causes the error floor to be elevated by an order of magnitude, as seen in Fig. 4. To fix the error floor, the post-processor is added to the 4-bit decoder. The post-processor increases the core area by 13.7% to 5.05 mm^2 and the power consumption by 17.6% to 810 mW. However, as an internal addition to the VN, the post-processor does not contribute to the wiring external to the VN. Overall wiring overhead increases by only 1.7%, indicating that the majority of the area and power increase is attributed to the extra logic and storage in the VN. The almost constant wiring overhead allows the maximum clock frequency to be maintained, and the decoding throughput is kept at 8.53 Gb/s.

To increase the decoding throughput further, an early termination scheme [17], [19] is implemented on-chip to detect early convergence by monitoring whether all the check nodes are satisfied and if so, the decoder can immediately proceed to the next input frame. The early termination scheme eliminates idle cycles and the processing nodes are kept busy constantly. The throughput gain becomes significant at high SNR levels – at an SNR level of 5.5 dB, convergence can be achieved in 1.47 iterations on average. Even accounting for one additional iteration in detecting convergence, the average throughput can be improved to 27.7 Gb/s as shown in Fig. 12. With early termination, the power consumption increases by 18.4% to 960 mW due to a higher activity factor. Now with a much higher throughput, the clock frequency

and supply voltage can be aggressively scaled down to reduce the power consumption. To reach the required throughput of 6.67 Gb/s, the clock frequency can be scaled to 100 MHz and the supply voltage scaled to 0.7 V to reduce the power consumption by almost 85% to 145 mW.

The decoding throughput quoted for an early-termination-enabled decoder is an average throughput at a specific SNR point. A maximum iteration limit is still imposed to prevent running an excessive number of iterations due to the occasional failures. A higher maximum iteration limit calls for a larger input and output buffering to provide the necessary timing slacks. A detailed analysis can be performed to determine the optimal buffer length for a performance target [30].

VII. Chip Implementation

The decoder is implemented in ST 65nm 7-metal low-power CMOS technology [31]. An initial density of 80% is used in placement and routing to produce the final density of 84.5% in a 5.35 mm^2 core. The decoder occupies 5.05 mm^2 of area, while the remaining 0.30 mm^2 is dedicated to on-chip AWGN noise generation, error collection, and I/O compensation. The chip microphotograph is shown in Fig. 13, featuring the dimensions of 2.556 $\times$ 2.608 mm for a chip area of 6.67 mm^2. The nominal core supply voltage is 1.2 V. The clock signal is externally generated.

A. Chip Testing Setup

The chip supports automated, real-time functional testing by incorporating AWGN noise generators and error collection. AWGN noise is implemented by the Box-Muller algorithm and the unit Gaussian random noise is scaled by pre-stored multipliers to emulate an AWGN channel at a particular SNR level. The automated functional testing assumes either all-zeros or all-ones codeword. The output hard decisions are compared to the expected codeword, and the number of bit and frame mismatches are accumulated in the error counters.

An internally-developed FPGA board is programmed to be the equivalent logic analyzer that can be attached to the chip test board. In the simplest setup, the registers can be programmed on the FPGA to connect to the corresponding interface pins to the test board. A write operation to the register functions as an input to the chip under test and a read functions as an output from the chip. This simplest form is used in automated testing, where the control signals (start, load, reset) and configuration (limit on iteration count, SNR level, limit on the number of input

frames) are set via the FPGA board. The progress of decoding (number of frames processed) can be monitored by polling the corresponding registers. Decoding results (bit and frame error counts) are collected by the decoder chip and can be read through the FPGA.

In a more elaborate testing scheme, the FPGA is programmed to generate the input data which are scanned in. A functionally-equivalent LDPC decoder (of a much lower throughput due to resource limitations) is programmed on the FPGA, which runs concurrently with the decoder chip. The output from the chip through output scan chains is compared to the on-FPGA emulation to check for errors. This elaborate testing scheme enables more flexibility of operating on any codeword, however the decoder needs to be paused in waiting for scan-in and scan-out to complete loading and unloading, resulting in a much lower decoding throughput.

B. Measurement Results

The chip is fully functional. Automated functional testing has been used to collect error counts at a range of SNR levels to generate the waterfall curve. Early termination is applied in increasing the decoding throughput while the maximum iteration limit is set to 20 for regular decoding. Without post-processing, the waterfall curve displays a change of slope below the BER of 10^{-11}. After enabling post-processing, the error floor is lowered and an excellent error correction performance is measured below the BER of 10^{-14}, as shown in Fig. 14. The measured waterfall curve matches the performance obtained from hardware emulation shown in Fig. 4 with extended BER by more than two orders of magnitude at high SNR levels. The post-processor suppresses the error floor by eliminating the absorbing errors, which is evident in Table III. Five of the seven unresolved errors at the highest SNR point on the curve (5.2 dB) are due to undetected errors – errors that are valid codewords, but not the intended codeword. It was empirically discovered that the minimum distance is 14 for the (2048,1723) RS-LDPC code. The eventual elimination of absorbing errors and the emergence of weight-14 undetected errors indicate the near maximum-likelihood decoding performance.

The decoder chip operates at a maximum clock frequency of 700 MHz at the nominal 1.2 V supply, delivering a throughput of 47.7 Gb/s. The throughput is measured at an SNR level of 5.5 dB with early termination enabled on-chip. To achieve the required 6.67 Gb/s of throughput for 10GBASE-T Ethernet, the chip can be frequency and voltage scaled to operate at 100 MHz at a 0.7 V supply, while dissipating only 144 mW. At the maximum allowed supply voltage of

1.32 V, a decoding throughput of 53.3 Gb/s is achieved at the clock frequency of 780 MHz.

The maximum clock frequency and decoding throughput are measured at each supply voltage. The measurements are performed by fixing the supply voltage while ramping up the clock frequency until the FER and BER performance start to deviate. The power consumption and decoding throughput are shown against the clock frequency in Fig. 15. Quadratic power savings can be realized by the simultaneous voltage and frequency scaling. It is therefore more power efficient to operate at the lowest supply voltage and clock frequency to deliver the required throughput within this range of operation.

The features of the decoder chip are summarized in Table IV. At the nominal supply voltage and the maximum 700 MHz of clock frequency, the decoder experiences the worst latency of 137 ns assuming an 8-iteration regular decoding limit, or 206 ns if an additional 4-iteration post-processing is accounted for. The energy per coded bit is 58.7 pJ/bit. At the 100 MHz clock frequency and a 0.7 V supply voltage, the worst latency is 960 ns (or 1440 ns with a 4-iteration post-processing), but the energy per coded bit is reduced to 21.5 pJ/bit. These implementation results compare favorably to the state-of-the-art high-throughput LDPC decoder implementations.

VIII. Conclusion

A highly parallel LDPC decoder is designed for the (2048,1723) RS-LDPC codes suitable for 10GBASE-T Ethernet. A two-step decoding scheme shortens the minimum wordlength required to achieve a good decoding performance. A grouping strategy is applied in the architectural design to divide wires into global wires and local wires. The optimal architecture lies in the point where the incremental wiring per additional degree of parallelism reaches the minimum, which coincides with the balance point between area and throughput. The LDPC decoder is synthesized, placed and routed to achieve a 84.5% density without sacrificing the maximum clock frequency. The message-passing decoding is scheduled based on a 7-stage pipeline to deliver a high effective throughput.

The optimized decoder architecture, when aided by an early termination scheme, achieves a maximum 47.7 Gb/s decoding throughput at the nominal supply voltage. The high throughput capacity allows the voltage and frequency to be scaled to reduce the power dissipation to 144 mW while delivering a 6.67 Gb/s throughput. Automated functional testing with real-time noise generation and error collection extends the BER measurements below 10^{-14}, where no error

floor is observed.

Techniques applied in this decoder chip design can be extended to many other high-throughput applications, including data storage, optical communications, and high-speed wireless. Enabling the reconfigurability of such a high-throughput architecture is the topic of future work.

ACKNOWLEDGMENT

The authors would like thank Dr. Zining Wu, Dr. Engling Yeo and other members of the read channel group at Marvell Semiconductor for helpful discussions and Dr. Pascal Urard and his team at ST Microelectronics for contributing constructive suggestions on the chip design. This research is a result of past and ongoing collaboration with Dr. Lara Dolecek and Pamela Lee at UC Berkeley. The authors also wish to acknowledge the contributions of the students, faculty, and sponsors of Berkeley Wireless Research Center and Wireless Foundations. In particular, Brian Richards and Henry Chen assisted with design flow and test setup.

REFERENCES

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

[2] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, no. 6, pp. 457–458, Mar. 1997.

[3] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

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

[5] *ETSI Standard TR 102 376 V1.1.1: Digital Video Broadcasting (DVB) User guidelines for the second generation system for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2)*, ETSI Std. TR 102 376, Feb. 2005.

[6] A. Morello and V. Mignone, "DVB-S2: the second generation standard for satellite broad-band services," *Proceedings of the IEEE*, vol. 94, no. 1, pp. 210–227, Jan. 2006.

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

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

[9] *IEEE Draft Standard for Information Technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment : Enhancements for Higher Throughput*, IEEE Std. 802.11n/D2.00, Feb. 2007.

[10] K. S. Andrews, D. Divsalar, S. Dolinar, J. Hamkins, C. R. Jones, and F. Pollara, "The development of turbo and LDPC codes for deep-space applications," *Proceedings of the IEEE*, vol. 95, no. 11, pp. 2142–2156, Nov. 2007.

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

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

[13] H. Liu, C. Lin, Y. Lin, C. Chung, K. Lin, W. Chang, L. Chen, H. Chang, and C. Lee, "A 480Mb/s LDPC-COFDM-based UWB baseband transceiver," in *Proc. IEEE International Solid-State Circuits Conference*, San Francisco, CA, Feb. 2005, pp. 444–445.

[14] 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 *Proc. IEEE International Solid-State Circuits Conference*, San Francisco, CA, Feb. 2005, pp. 446–447.

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

[16] E. Yeo and B. Nikolić, "A 1.1-Gb/s 4092-bit low-density parity-check decoder," in *Proc. IEEE Asian Solid-State Circuits Conference*, Hsinchu, Taiwan, Nov. 2005, pp. 237–240.

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

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

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

[20] T. Richardson, "Error floors of LDPC codes," in *Proc. Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Oct. 2003, pp. 1426–1435.

[21] Z. Zhang, L. Dolecek, B. Nikolić, V. Anantharam, and M. J. Wainwright, "Lowering LDPC error floors by postprocessing," in *Proc. IEEE Global Communications Conference*, New Orleans, LA, Nov. 2008, pp. 1–6.

[22] Z. Zhang, "Design of LDPC decoders for improved low error rate performance," Ph.D. dissertation, University of California, Berkeley, Berkeley, CA, 2009.

[23] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.

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

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

[26] J. Zhao, F. Zarkeshvari, and A. H. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes," *IEEE Transactions on Communications*, vol. 53, no. 4, pp. 549–554, Apr. 2005.

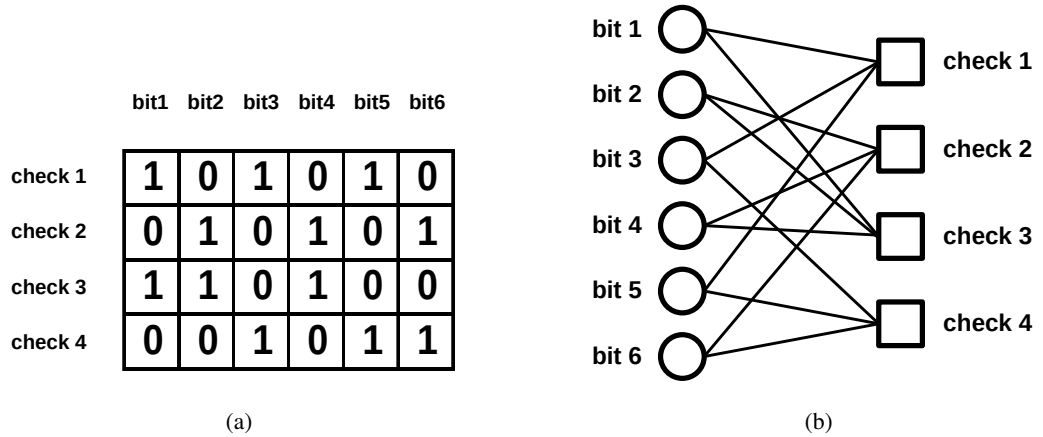
[27] Z. Zhang, L. Dolecek, B. Nikolić, V. Anantharam, and M. J. Wainwright, "Design of LDPC decoders for improved low error rate performance: quantization and algorithm choices," *IEEE Transactions on Communications*, to be published.

[28] M. M. Mansour and N. R. Shanbhag, "Turbo decoder architectures for low-density parity-check codes," in *Proc. IEEE Global Communications Conference*, Taipei, Taiwan, Nov. 2002, pp. 1383–1388.

[29] J. Zhang and M. P. C. Fossorier, "Shuffled iterative decoding," *IEEE Transactions on Communications*, vol. 53, no. 2, pp. 209–213, Feb. 2005.

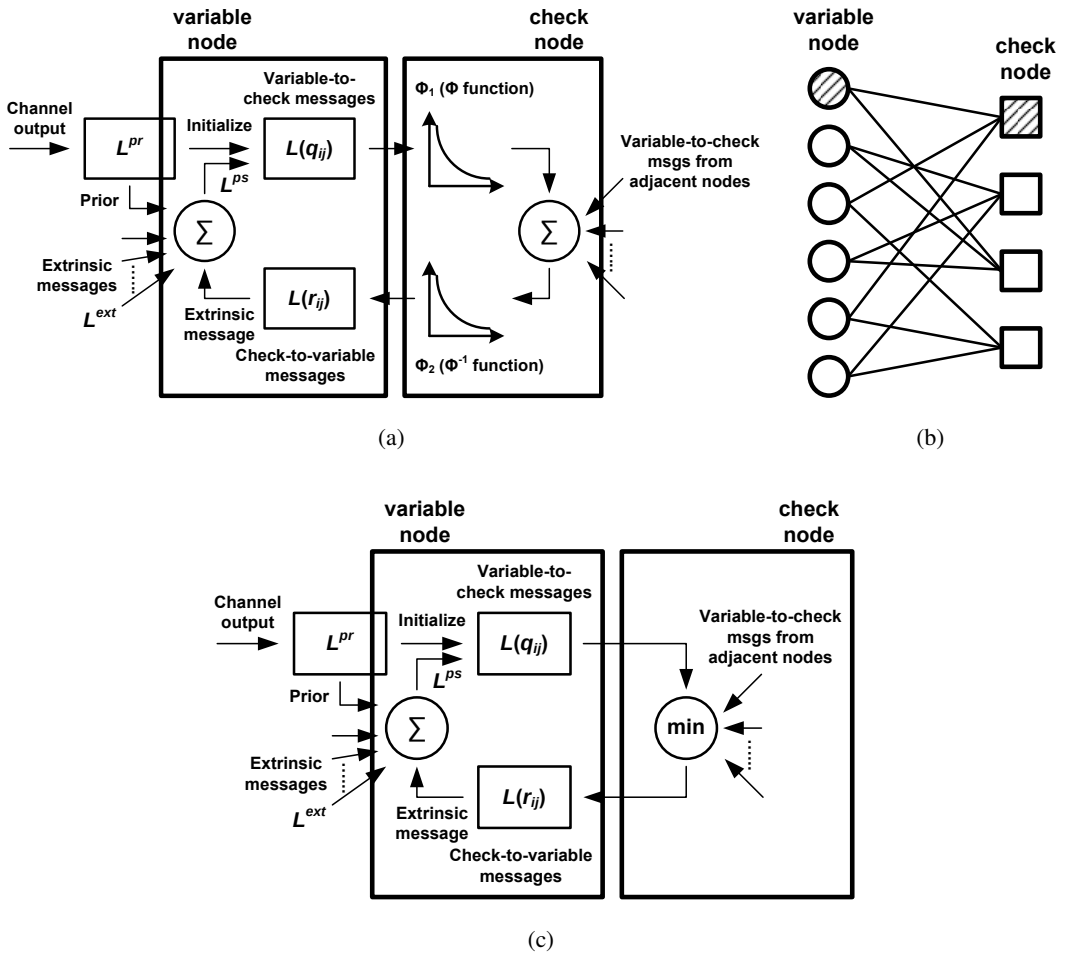
[30] G. Bosco, G. Montorsi, and S. Benedetto, "Decreasing the complexity of LDPC iterative decoders," *IEEE Communications Letteres*, vol. 9, no. 7, pp. 634–636, Jul. 2005.

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



(a) (b)

Fig. 1. Representation of an LDPC code in (a) a parity-check matrix (**H** matrix), (b) a factor graph.



(a) (b)

(c)

Fig. 2. Message-passing decoding implementation showing (a) sum-product message-passing decoding, (b) the corresponding one slice of a factor graph, (c) min-sum message-passing decoding.

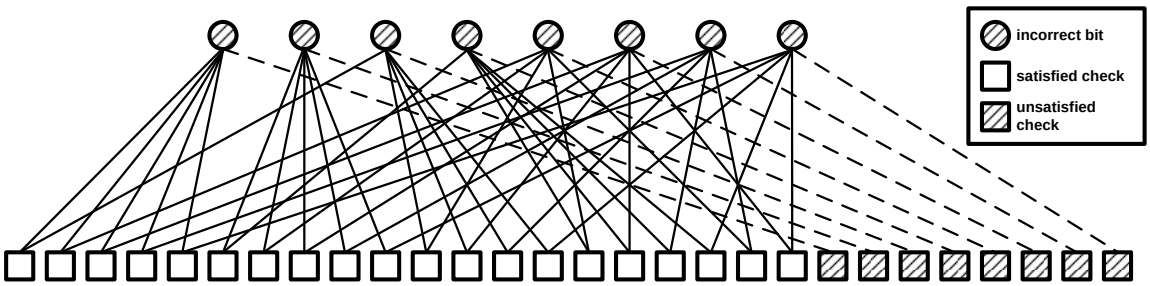


Fig. 3. Illustration of the subgraph induced by the incorrect bits in an (8,8) fully absorbing set.

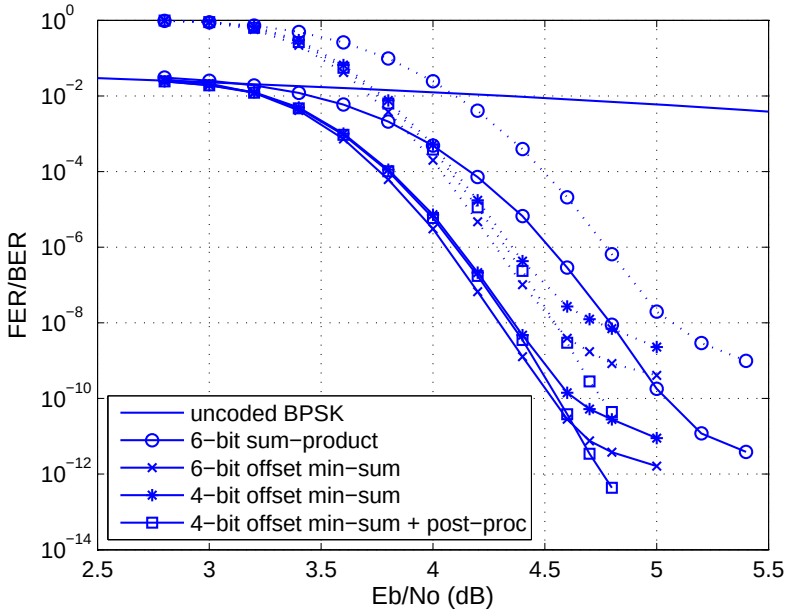
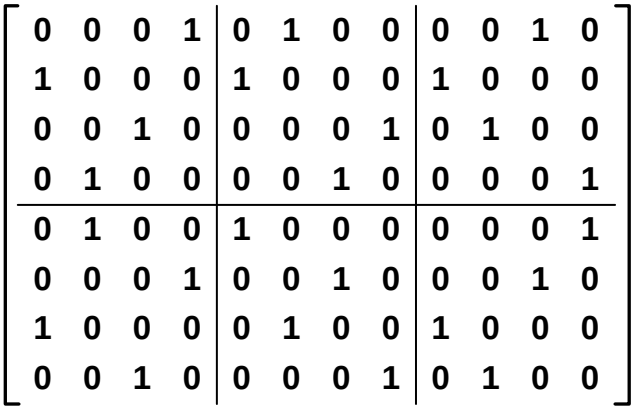
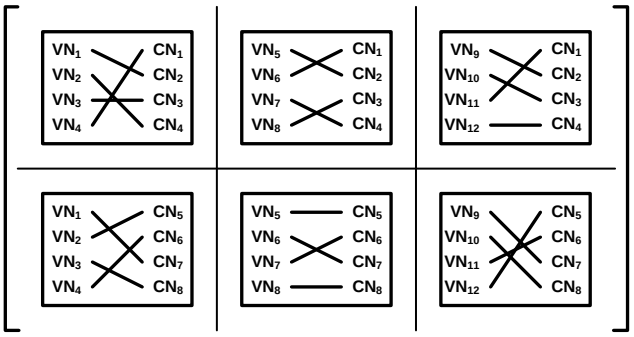


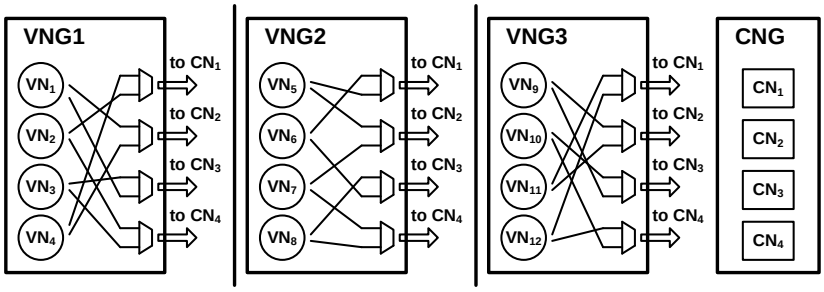
Fig. 4. FER (dotted lines) and BER (solid lines) performance of a (2048,1723) RS-LDPC code obtained by FPGA emulation using 20 decoding iterations.



(a)



(b)



(c)

Fig. 5. Architectural mapping and transformation: (a) a simple structured **H** matrix, (b) the fully parallel architecture, (c) a 3VNG-1CNG parallel architecture.

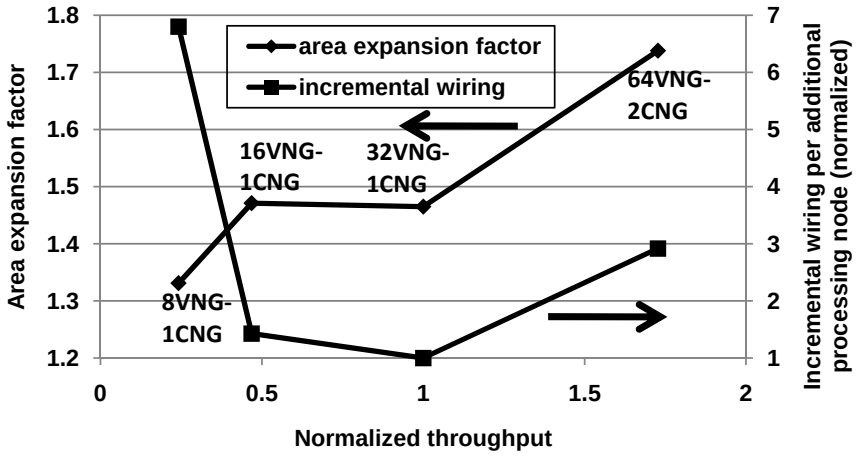


Fig. 6. Architectural optimization by the area expansion metric.

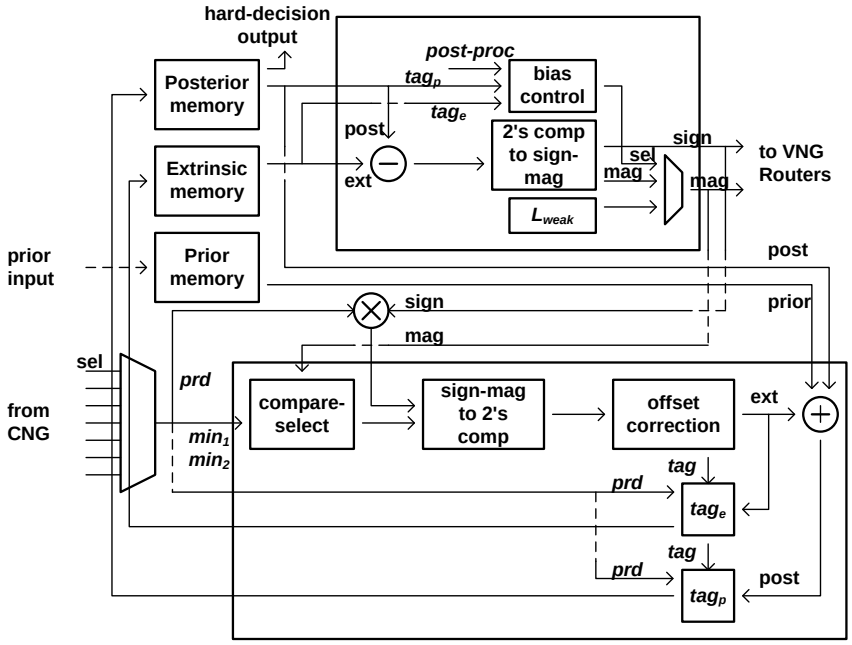


Fig. 7. VN design for post-processing.

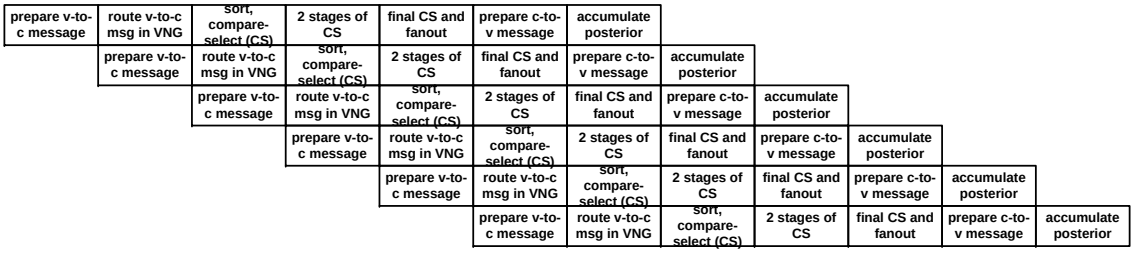


Fig. 8. Pipeline design of the 32VNG-1CNG decoder.

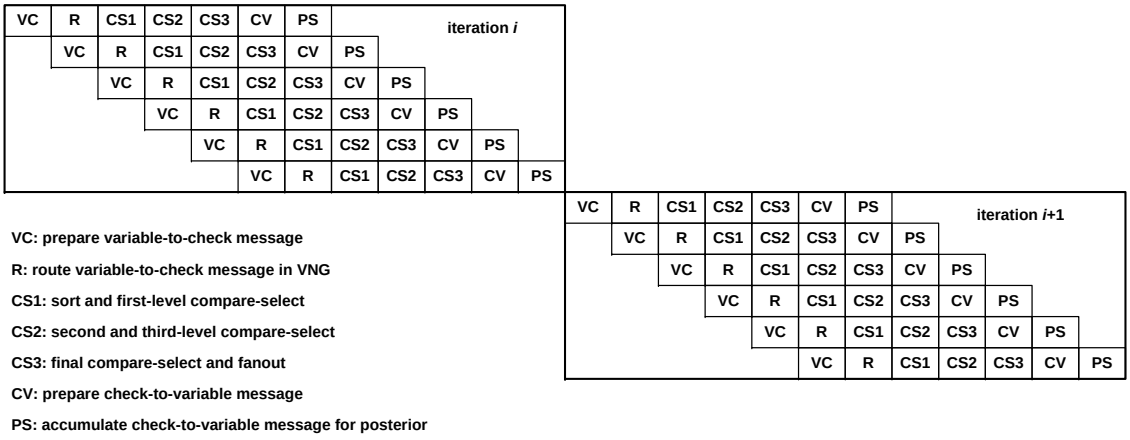


Fig. 9. Two-iteration pipeline chart with pipeline stalls.

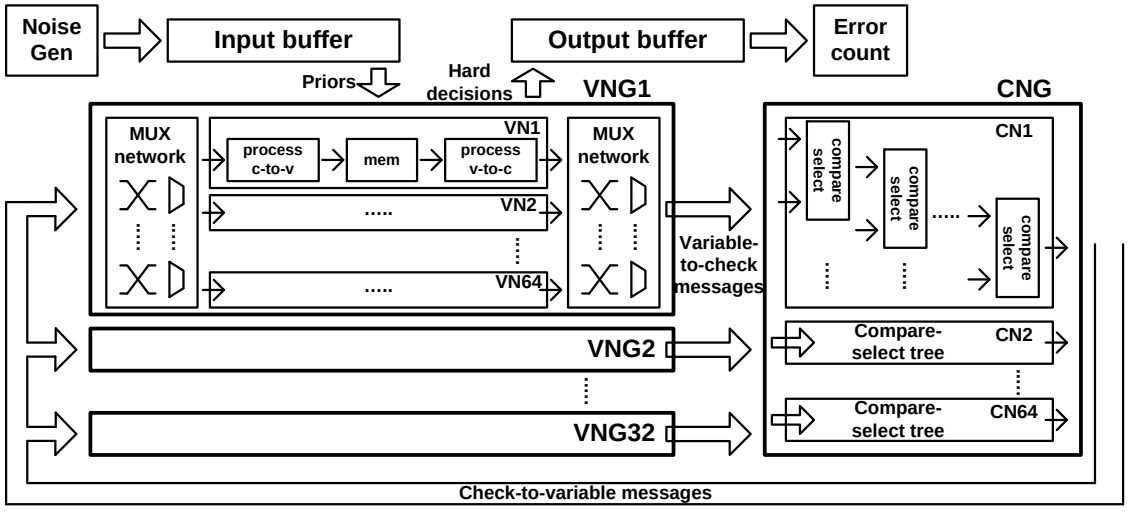
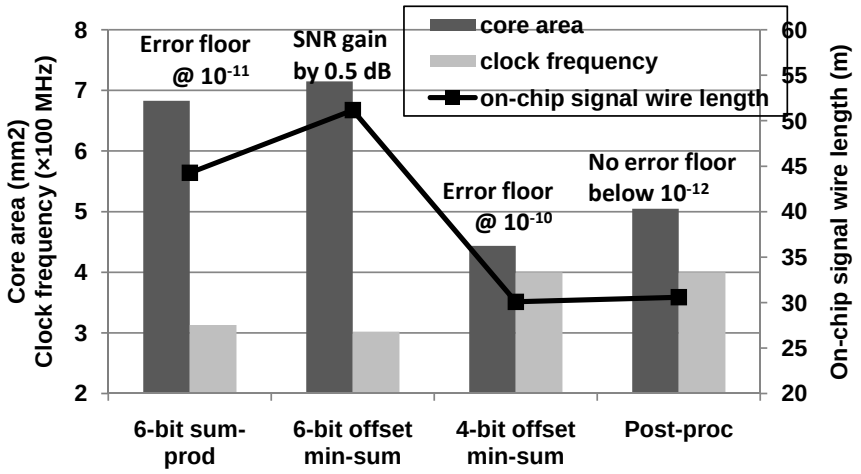
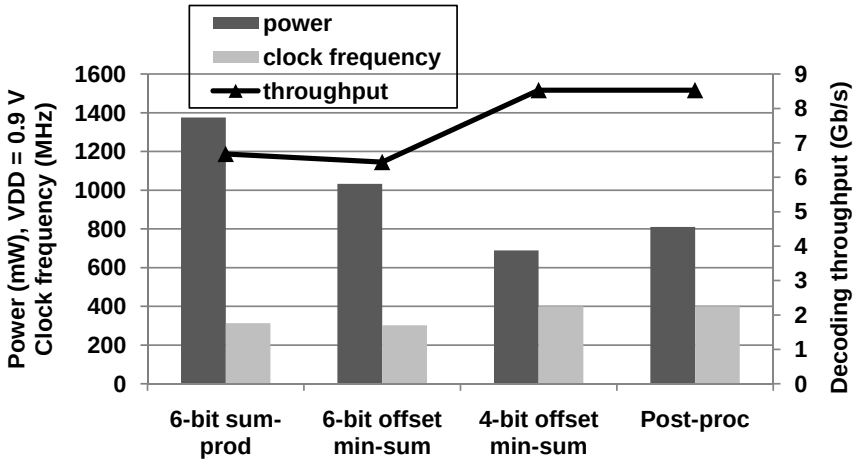


Fig. 10. The decoder implementation using the 32VNG-1CNG architecture.



(a)



(b)

Fig. 11. Steps of improvement evaluated on the 32VNG-1CNG architecture using synthesis, place and route results in the worst-case corner: (a) area and performance improvement, (b) power and throughput improvement.

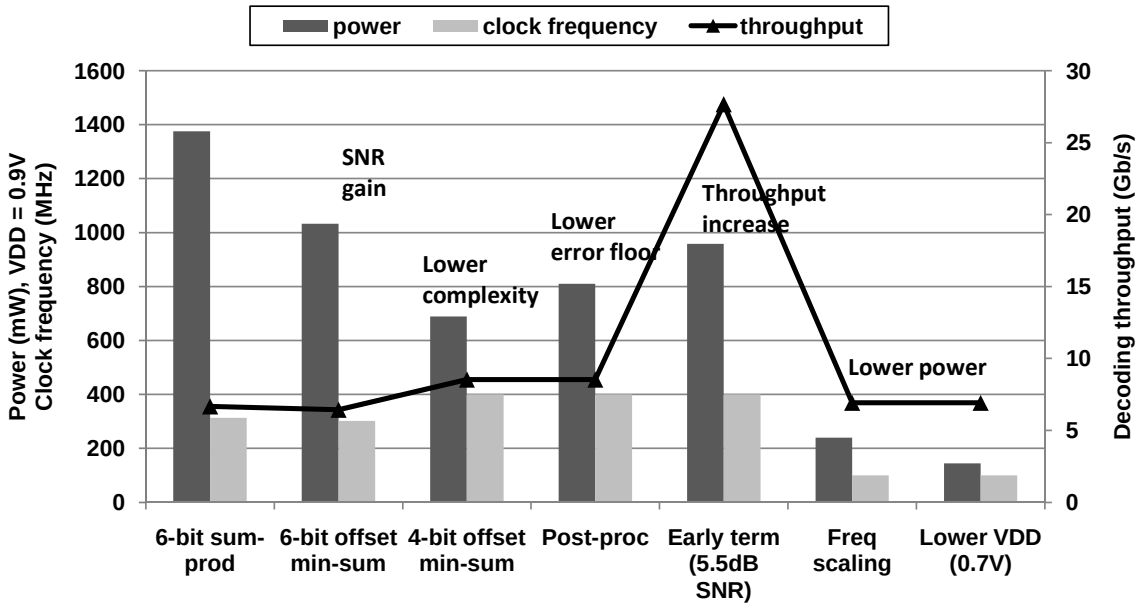


Fig. 12. Power reduction steps with results from synthesis, place and route in the worst-case corner.

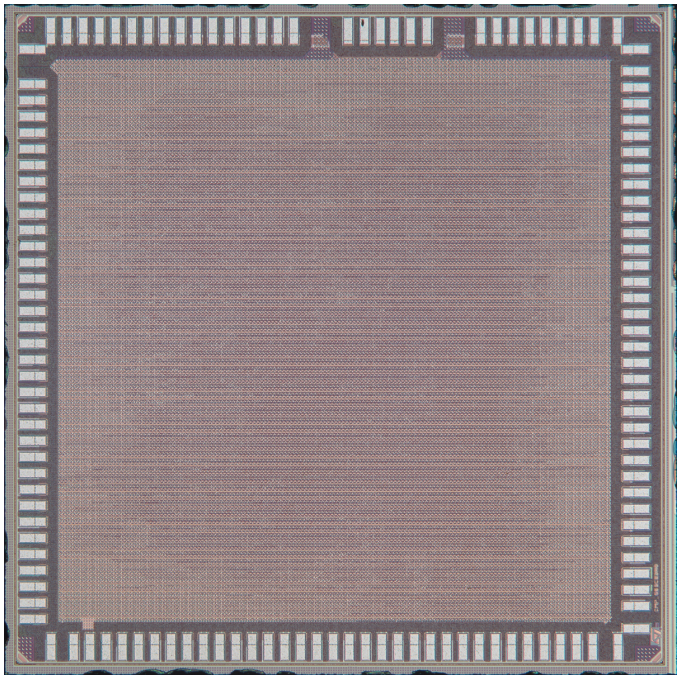


Fig. 13. Chip microphotograph.

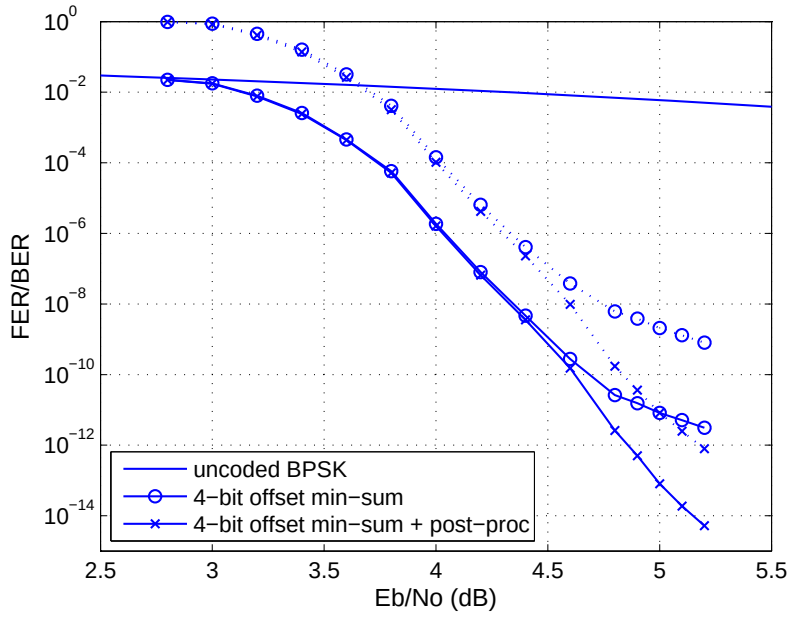


Fig. 14. Measured FER (dotted lines) and BER (solid lines) performance of the decoder chip using a maximum of 20 decoding iterations.

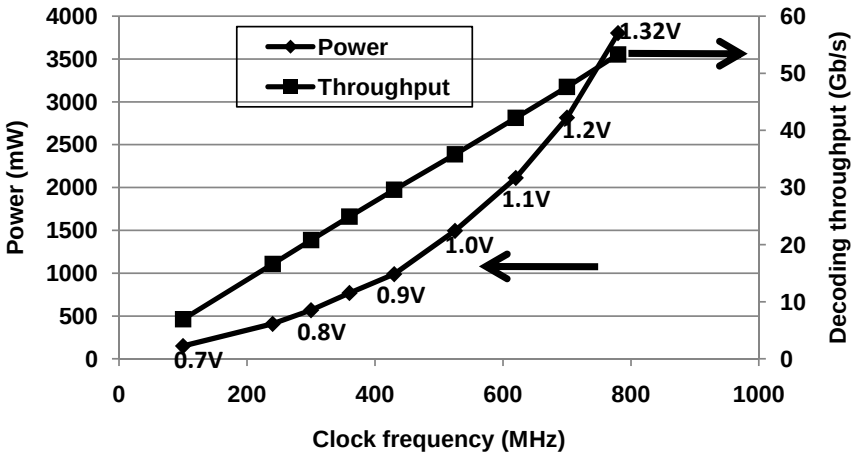


Fig. 15. Frequency and power measurement results of the decoder chip.

TABLE I

ARCHITECTURAL SELECTION BASED ON SYNTHESIS, PLACE AND ROUTE RESULTS IN THE WORST-CASE CORNER

Architecture	VN	CN	Freq(MHz)	Density	AEF	Wire length(m)
8VNG-1CNG	512	64	400	91.01%	1.331	20.343
16VNG-1CNG	1024	64	400	91.21%	1.471	24.614
32VNG-1CNG	2048	64	400	84.17%	1.465	30.598
64VNG-2CNG	4096	128	350	86.84%	1.738	65.504

TABLE II

DENSITY SELECTION BASED ON SYNTHESIS, PLACE AND ROUTE RESULTS IN THE WORST-CASE CORNER

Initial density	Final Density	Frequency(MHz)	Wire length(m)	Wire count(,000)
70%	74.43%	400	31.884	9,071
75%	79.73%	400	31.868	9,270
80%	84.17%	400	30.598	9,295
85%	90.48%	360	32.118	10,258
90%	96.29%	350	31.431	10,308

TABLE III

ERROR STATISTICS BASED ON CHIP MEASUREMENTS

SNR (dB)	Before post-processing		After post-processing	
	Errors	Average weight	Errors	Average weight
4.8	3396	8.72	95	30.66
4.9	4229	8.23	40	28.20
5.0	4553	8.08	18	20.22
5.1	5714	8.04	11	15.36
5.2	7038	8.01	7	13.43

TABLE IV

CHIP FEATURES

Technology	ST 65nm low-power CMOS	
Core area	2.316 $\times$ 2.311 mm	
Chip area	2.556 $\times$ 2.608 mm	
Area utilization	80% initial, 84.5% final	
Clock Frequency	100 MHz	700 MHz
Supply Voltage	0.7 V	1.2 V
Power Consumption	144 mW	2.80 W
Decoding Throughput[1]	6.67 Gb/s	47.7 Gb/s
Decoding Latency[2]	960 ns	137 ns
Energy Efficiency	21.5 pJ/bit	58.7 pJ/bit

[1] At SNR = 5.5 dB with early termination.

[2] Assumes an 8-iteration decoding limit for regular decoding at a low SNR point. The worst latencies are longer if the post-processing is accounted for.