A 640-Mb/s 2048-Bit Programmable LDPC Decoder Chip

Mohammad M. Mansour, *Member, IEEE,* and Naresh R. Shanbhag, *Fellow, IEEE*

***Abstract*—A 14.3-mm^2 code-programmable and code-rate tunable decoder chip for 2048-bit low-density parity-check (LDPC) codes is presented. The chip implements the turbo-decoding message-passing (TDMP) algorithm for architecture-aware (AA-)LDPC codes which has a faster convergence rate and hence a throughput advantage over the standard decoding algorithm. It employs a reduced complexity message computation mechanism free of lookup tables, and features a programmable network for message interleaving based on the code structure. The chip decodes any mix of 2048-bit rate-1/2 (3,6)-regular AA-LDPC codes in standard mode by programming the network, and attains a throughput of 640 Mb/s at 125 MHz for 10 TDMP-decoding iterations. In augmented mode, the code rate can be tuned up to 14/16 in steps of 1/16 by augmenting the code. The chip is fabricated in 0.18-μm six-metal-layer CMOS technology, operates at a peak clock frequency of 125 MHz at 1.8 V (nominal), and dissipates an average power of 787 mW.**

***Index Terms*—Architecture-aware low-density parity-check (AA-LDPC) codes, iterative decoders, LDPC codes, turbo-decoding message-passing (TDMP) algorithm, VLSI decoder architectures.**

I. Introduction

With sustained growth in demands for multimedia, wireless, and broadband services, significant effort has been made to apply iterative forward error correction (FEC) coding techniques to advanced communications systems. These techniques have proved to be very effective in extending the limits and services of wireless communications, expanding the areal density of magnetic recording systems, and improving the throughput of terrestrial optical systems. Low-density parity-check (LDPC) codes [1] have emerged as one of the top contenders for such applications after their main rivals, turbo codes [2], have seen limited acceptance (particularly in optical applications) due to their high implementation complexity, decoding latency, as well as performance degradation for relatively short block-length and error-floors at high signal-to-noise ratios (SNRs). Research has shown that LDPC codes can achieve record-breaking performance for low SNR applications [3], [4], and are more amenable to rigorous analysis and design. They offer more flexibility in the choice of code parameters, and their decoders require simpler processing. These characteristics have made it possible to design appropriate LDPC codes for many communications scenarios; they have been adopted in next generation digital video broadcasting (DVB-S2) via satellite [5], and considered for adoption in wireless local area network (WLAN) air interface (802.11) [6], wireless personal area networks (WPANs) (802.12) [7], mobile broadband wireless access (MBWA) networks (802.20) [8], advanced magnetic and magneto-optic storage/recording systems [9], and long-haul optical communication systems [10].

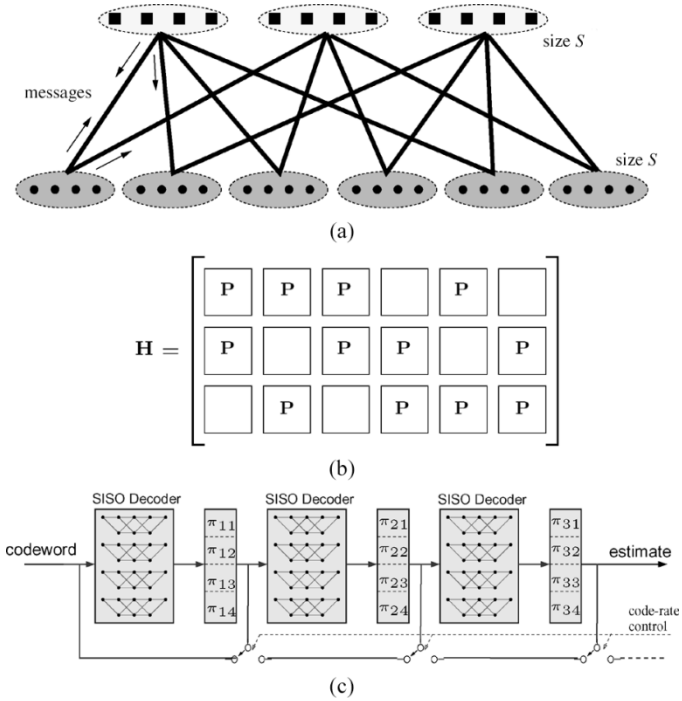

Fig. 1. Architecture-aware LDPC codes: (a) tanner graph, (b) parity-check matrix, and (c) decoder block diagram.

LDPC codes are linear block codes defined by a sparse parity-check matrix. Such matrices can be efficiently represented by a bipartite (Tanner) graph [11]. The standard iterative decoding algorithm, known as Gallager's two-phase message-passing (TPMP) algorithm [1], passes messages along the edges of this graph in multiple rounds of updates between the two classes of nodes. Contemporary LDPC decoders are based on a parallel or serialized version of the TPMP algorithm. The complexity of a fully parallel decoder [12] grows linearly with the block length and is practical only for relatively short LDPC codes, while a serial decoder [13] trades off throughput with

Manuscript received July 20, 2005; revised September 21, 2005.

M. M. Mansour is with the Department of Electrical and Computer Engineering, American University of Beirut, Beirut 1107 2020, Lebanon (e-mail: mmansour@aub.edu.lb).

N. R. Shanbhag is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: shanbhag@uiuc.edu).

Digital Object Identifier 10.1109/JSSC.2005.864133

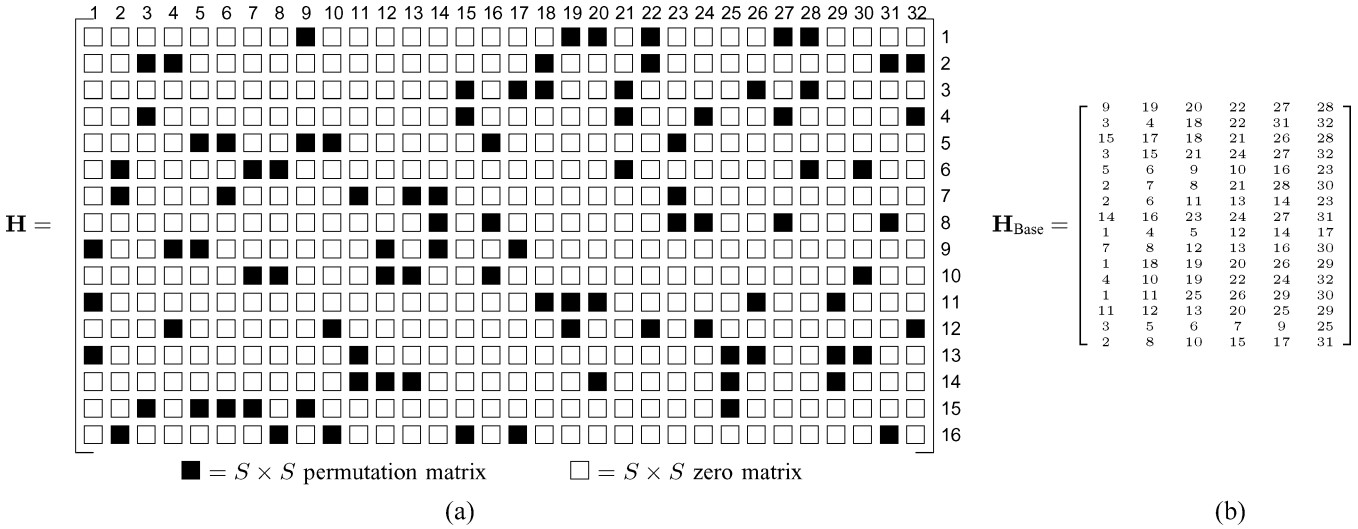

Fig. 2. Example parity-check matrix of an AA-LDPC code. (a) Matrix of size 1024 × 2048 composed of 16 × 32 rows of submatrices each of size 64 × 64. Black squares correspond to permutation matrices, while empty squares correspond to all-zero matrices. (b) Base parity-check matrix of size 16 × 6 containing column addresses of the permutation matrices in $\mathbf{H}$.

algorithmic performance by increasing the block length, and is primarily targeted for low-throughput applications.

In order to achieve throughputs and bit-error rate (BER) performance compliant with current and future trends in high-speed wireless, magnetic, and optical systems, a new class of architecture-aware (AA-)LDPC codes [see Fig. 1] targeted for *high-throughput* LDPC decoders of *long* codes has been proposed [14], [15]. This class of codes alleviates the interconnect complexity problem typical of current parallel decoders. AA-LDPC code design decouples the architectural dependence of the decoder from the code properties by performing optimizations at the code-design, decoding algorithm, decoder architecture, and physical layout levels. Fig. 1(a) shows the Tanner graph of an AA-LDPC code with nodes decomposed into clusters of size S, such that all nodes in one cluster connect to nodes of another cluster in some specified order. The parity-check matrix of an AA-LDPC code has the structure shown in Fig. 1(b), where $\mathbf{P}$ is a permutation matrix of size S denoting the order in which the nodes of two clusters are connected.[1] The structure of AA-LDPC codes allows for an efficient decoder implementation using the turbo-decoding message-passing (TDMP) algorithm [16]. Fig. 1(c) shows the block diagram of a TDMP decoder composed of constituent soft-input soft-output (SISO) decoders separated by interleavers factored according to the row structure of the parity-check matrix. If these interleavers can be programmed with the required permutations, a TDMP decoder reduces simply to one SISO decoder and one programmable interleaver.

Following the design methodology proposed in [17], a high-throughput programmable TDMP decoder with variable code-rate for length 2048, (3,6)-regular AA-LDPC codes has been implemented. Section II describes the operation and architecture of the TDMP decoder. The message computation mechanism is discussed and the SISO engine of the decoder is presented. Section II also describes the message routing mechanism of the decoder using a programmable network. Section III describes the circuit style employed in designing the combinational logic of the decoder as well as the memory blocks. Finally, Section IV presents testing and measured results.

[1]Note the permutation matrices in $\mathbf{H}$ are distinct.

II. TDMP Decoder Architecture

The TDMP decoder receives soft channel observations proportional to the log-likelihood values of the received bits (also called reliabilities), and generates log-likelihood or reliability values of the decoded bits. The soft values are represented by 4-bit two's complement numbers. The sign of a number indicates the bit decision or hard binary value, while the magnitude represents the reliability of making a decision in favor of a 0 or a 1. Let δ_j denote the received channel soft value, and Γ_j denote the decoded soft output value corresponding to the jth bit, $j = 1, \ldots, 2048$.

A. LDPC Code Structure

The received bits are assumed to be encoded by an AA-LDPC code of length 2048 whose parity-check matrix $\mathbf{H}$ is similar to the one shown in Fig. 2(a). $\mathbf{H}$ is composed of 16 block rows by 32 block columns of 64 × 64 submatrices, where each submatrix is either a permutation matrix (black square) or all-zero matrix (empty square). A permutation matrix is a matrix obtained by permuting the rows and/or columns of an identity matrix. The location and structure of the permutation matrices can be arbitrary; they are specified by the user for a given code. The size of $\mathbf{H}$ is 1024 × 2048. Each block row contains exactly six permutation matrices, and each block column contains exactly three permutation matrices. Hence, the code is a (3,6)-regular AA-LDPC of rate 0.5.

A base parity-check matrix $\mathbf{H}_{\text{Base}}$ is associated with $\mathbf{H}$ as shown in Fig. 2(b). The entries of $\mathbf{H}_{\text{Base}}$ correspond to the column addresses of the permutation matrices in $\mathbf{H}$. The size of $\mathbf{H}_{\text{Base}}$ is 16 × 6. The decoder stores the matrix $\mathbf{H}_{\text{Base}}$ in addition to $16 \times 6 = 96$ permutation matrices instead of storing $\mathbf{H}$. The base and permutation matrices are programmed by the user according to the desired structure of $\mathbf{H}$.

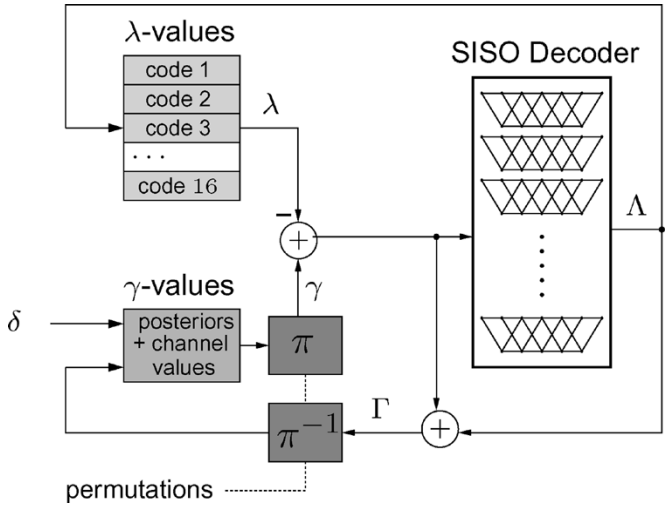

Fig. 3. Dataflow graph of the TDMP decoder.

B. Decoder Operation

According to the decomposition shown in Fig. 2(a), $\mathbf{H}$ is considered as the concatenation of 16 constituent codes corresponding to the 16 block rows. Each constituent code is an even parity-check code having support only in $6 \times 64 = 384$ positions. The TDMP decoder decodes a codeword iteratively in 16 sub-iterations, with one sub-iteration per constituent code, using a constituent SISO decoder and a programmable interleaver similar to decoding a serially concatenated turbo-code [16]. The dataflow graph of the decoder is shown in Fig. 3.

The SISO decoder accepts in a sub-iteration intrinsic input reliability messages (denoted by λ) previously generated from all other 15 sub-iterations. Intrinsic λ-messages pertaining to the code under consideration are excluded to minimize correlation between messages from different sub-iterations. The sum of all intrinsic λ-messages in addition to the channel values is denoted by γ, i.e.,

$$\gamma = \delta + \sum_{\text{all codes}} \lambda.$$

The decoder generates as output updated extrinsic messages (denoted by Λ) corresponding to the constituent code being decoded, as well as updated posterior messages (denoted by Γ). At the end of a sub-iteration, these Λ and Γ-values are stored as λ and γ-values to be used as inputs in the next sub-iteration.

Starting from block row 1 in $\mathbf{H}$, Λ-values are computed for each bit in code 1 by the SISO decoder assuming that the bit belongs to the code defined by block row 1. This is done by subtracting the λ-values of code 1 from the posterior γ-values after interleaving:

$$\underbrace{\left(\delta + \sum_{\text{all codes}} \lambda\right)}_{\gamma} - \lambda_{\text{code1}}.$$

The newly generated extrinsic Λ-values for code 1 are stored back as λ-values. The posterior Γ-values are updated by adding

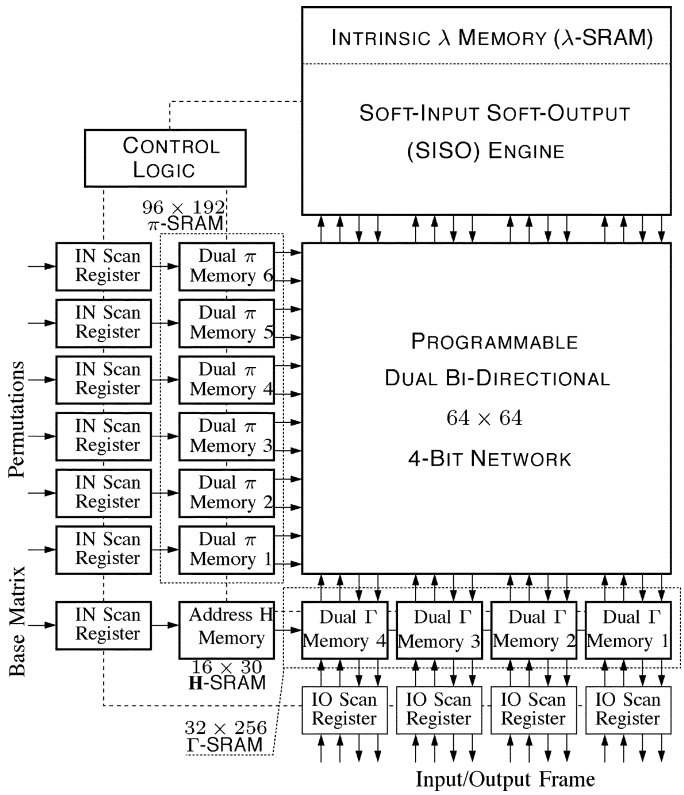

Fig. 4. System architecture of the TDMP decoder chip.

these Λ-values to the old sum excluding the λ-values of code 1 (i.e., to $\gamma - \lambda$)

$$\underbrace{\Gamma}_{\text{new}} = \underbrace{\left(\delta + \sum_{\text{all codes}} \lambda\right)}_{\gamma} - \underbrace{\lambda_{\text{code1}}}_{\text{old}} + \underbrace{\Lambda_{\text{code1}}}_{\text{new}}.$$

The updated Γ-values are de-interleaved and stored back as γ-values, concluding the first sub-iteration. The same steps are repeated for codes 2–16. The 16 sub-iterations together form one complete decoding iteration. During each sub-iteration, the interleavers are programmed according to the permutations of the block row corresponding to the code being decoded. The TDMP algorithm has a faster convergence rate (it requires 50% less iterations to converge at moderate to high SNR) and hence a throughput advantage over the standard algorithm [16].

The code rate of the decoder can be varied by puncturing the parity-check matrix across block row boundaries. Encoded LDPC codewords are augmented accordingly. The decoder decodes augmented codewords by running over unpunctured constituent codes (or block rows in $\mathbf{H}$), bypassing those that are punctured. Hence, the code rate can be tuned in steps of 1/16 between 8/16 and 14/16 corresponding to the 16 block rows of $\mathbf{H}$. The motivation behind increasing the code rate is to use the same hardware at improved channel conditions to transmit more information bits. This comes at the expense of algorithmic performance loss (e.g., at rate 9/16, the loss in performance is around two orders of magnitude in BER at 2.6 dB). The reader is referred to [16] and [17] for details regarding the TDMP algorithm.

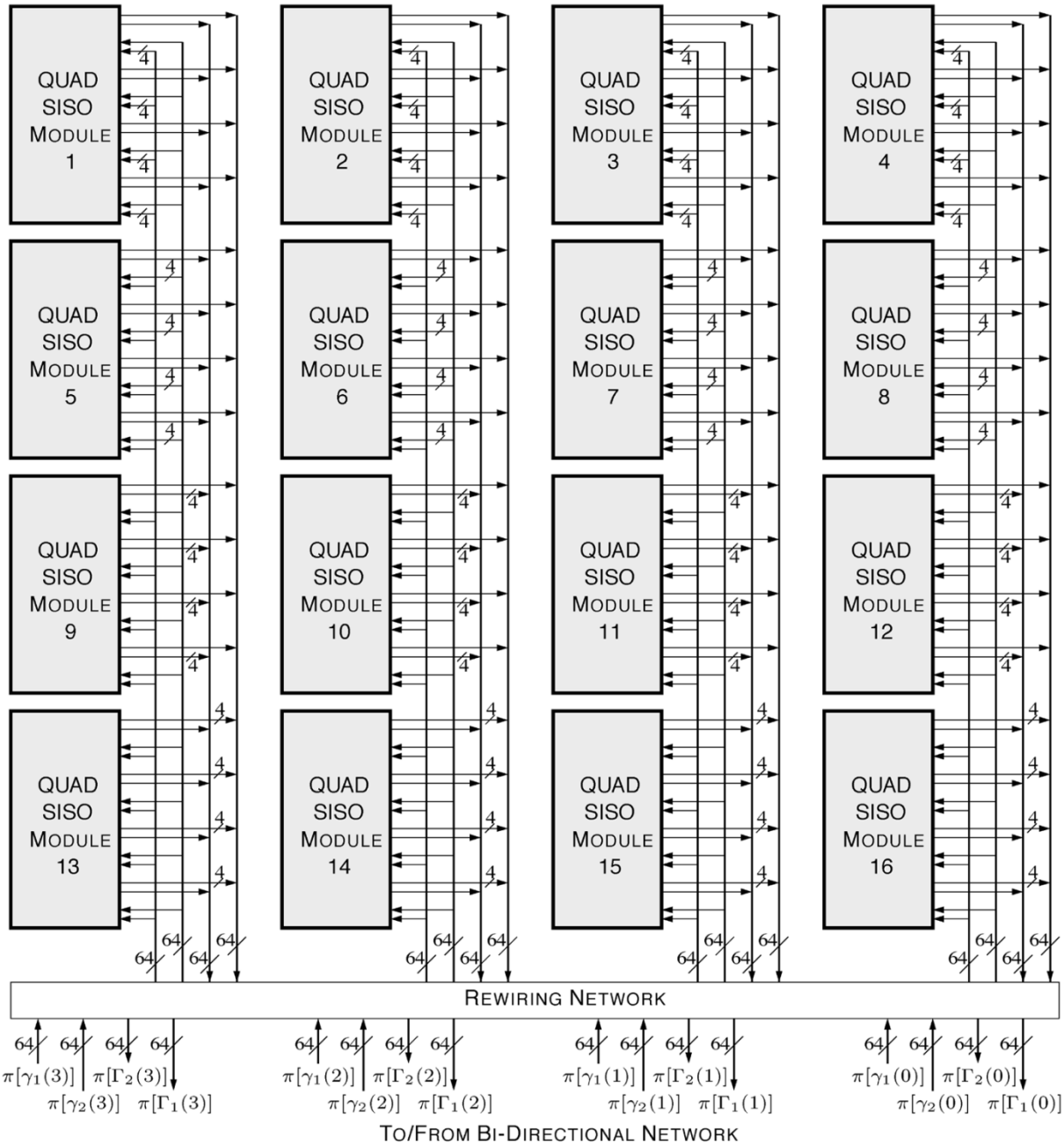

Fig. 5. Architecture of the SISO engine composed of 16 QUAD-SISO modules. Each module accepts eight 4-bit γ-messages and generates eight 4-bit Γ-messages. The rewiring network forms the 4-bit input messages to the modules by grouping one bit from each of the four 64-bit input buses $\pi[\gamma_i(3)], \pi[\gamma_i(2)], \pi[\gamma_i(1)], \pi[\gamma_i(0)]$, $i = 1,2$, coming from the dual bi-directional network. The resulting 64 4-bit messages are divided into four groups and supplied to the QUAD-SISO modules in each column. The same is done with the 64-bit output buses $\pi[\Gamma_i(3)], \pi[\Gamma_i(2)], \pi[\Gamma_i(1)], \pi[\Gamma_i(0)]$, $i = 1,2$.

C. Decoder Architecture

Fig. 4 shows the architecture of the TDMP decoder chip. It consists of a SISO engine for processing messages (i.e., generating Λ and Γ-messages), a dual bi-directional 64×64 shuffle-exchange network for routing messages, a dual-port 32×256 bit Γ-SRAM for storing posterior messages (stores $S = 64$ 4-bit messages for $B = 32$ submatrices), a dual-port 96×192 bit π-SRAM for storing routing permutations (stores $16 \times 6 = 96$ permutations, each permutation is on $S = 64$ elements; the network has $\log(S) = 6$ stages, each stage requires $S/2 = 32$ control bits), and a 16×30 $\mathbf{H}$-SRAM for storing the code's base parity-check matrix $(\mathbf{H}_{\text{Base}})$. Input 4-bit channel observations δ and intermediate posterior γ-messages are stored in Γ-SRAM. Soft outputs are generated back in Γ-SRAM after decoding is over.

A codeword is decoded partially in 16 sub-iterations in which posterior γ-messages (from Γ-SRAM) and intrinsic λ-messages (stored locally in the SISO engine) pertaining to one sub-iteration are combined to generate updated Λ and Γ-messages for subsequent sub-iterations based on the dual extrinsic principle [17]. During a sub-iteration, the network routes 128 4-bit messages in parallel from Γ-SRAM to the SISO engine using six shuffle-exchange network layers controlled by π-SRAM. The

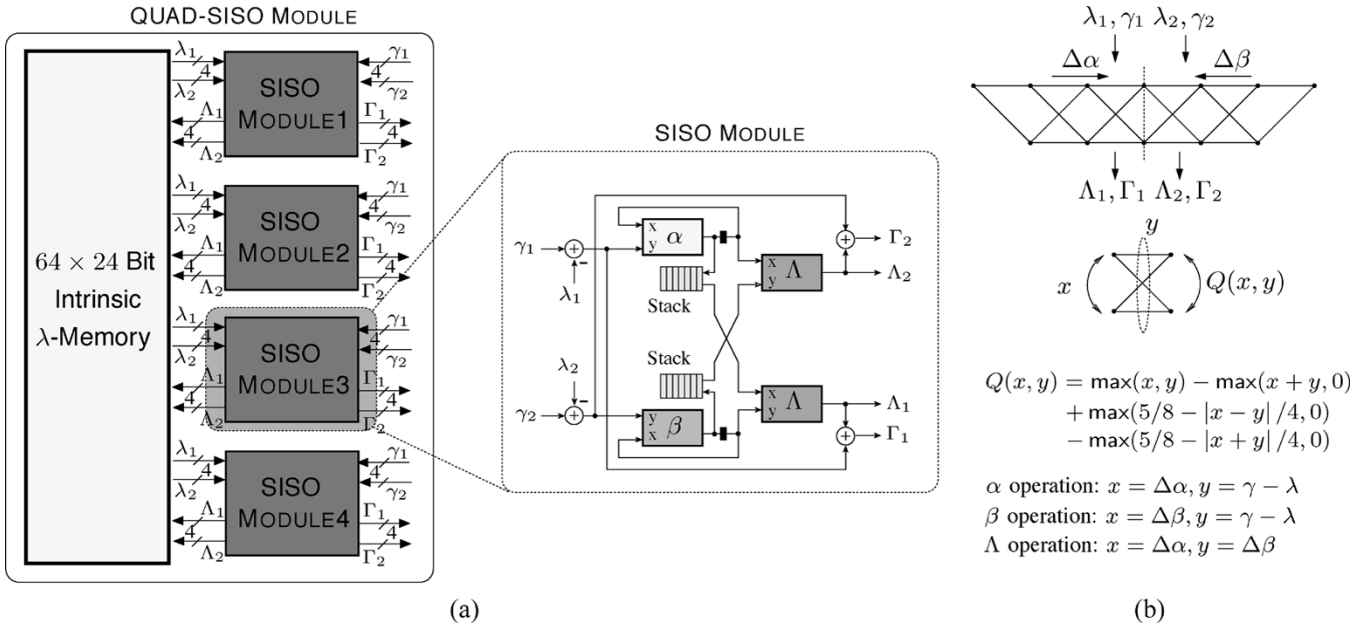

Fig. 6. QUAD-SISO module. (a) Architecture composed of four SISO modules and an intrinsic λ-memory. A SISO module is composed of four max-quartet Q-function modules for bi-directional processing of the trellis. (b) Illustration of the operations of a SISO module and its constituent Q-function modules on a six-stage trellis.

SISO engine processes the 128 messages in parallel using 16 QUAD-SISO clusters that store locally intrinsic messages in λ-SRAM (the size of λ-SRAM is $16 \times 24 \times 64 = 24\,576$ bits). The clusters generate updated extrinsic messages that are stored back locally, as well as posterior messages that are routed back through the network to Γ-SRAM. The rate of the code can be tuned up from 0.5 by truncating decoding sub-iterations. The total memory of the chip is 51 680 bits $(32 \times 256 + 96 \times 192 + 16 \times 30 + 16 \times 24 \times 64 = 51\,680)$, 75% less than the requirements of comparable serial LDPC decoders [13].

D. SISO Engine

The SISO engine of the TDMP decoder is composed of 16 clusters of QUAD-SISO modules as shown in Fig. 5. These modules operate in parallel to decode a constituent code in six clock cycles. The engine receives two sets of 64 4-bit γ-messages (128 messages in total) from the bi-directional network, and generates back two sets of 64 4-bit Γ-messages (128 messages in total) to the bi-directional network. The individual bits of each set of 64 input messages arriving from the bi-directional network come on four separate buses; buses $\pi[\gamma_1(3)], \pi[\gamma_1(2)], \pi[\gamma_1(1)], \pi[\gamma_1(0)]$ for the first set, and buses $\pi[\gamma_2(3)], \pi[\gamma_2(2)], \pi[\gamma_2(1)], \pi[\gamma_2(0)]$ for the second set. The rewiring network groups related bits in each set into 4-bit words, giving a total of 64 4-bit messages per set. These messages are supplied to the QUAD-SISO modules on eight 64-bit buses as shown in the figure, where each column of four QUAD-SISO modules shares two buses that feed in the γ-messages. A similar description holds for the 128 4-bit output messages generated by the SISO engine.

Each QUAD-SISO module accepts eight distinct 4-bit γ-messages and generates eight 4-bit Γ-messages. The architecture of a QUAD-SISO module is shown in Fig. 6. It consists of four SISO modules, and a local intrinsic 16×24 bit λ-SRAM for storing λ-messages during decoding sub-iterations. A SISO module processes a two-state, six-stage trellis simultaneously in both directions using the BCJR algorithm in differential form [16]. The module is pipelined and consists of four message processing units, one for forward (α), another for backward (β) state metric processing units, and two for output (Λ) message computation [see Fig. 6(a)]. Each clock cycle, two branch metrics (γ-values) are read per SISO module from the network and two new state metrics are generated and pushed onto the two stacks until the middle section of the trellis is reached. The state metrics are then popped and used to generate output messages using the Λ units, two at a time.

All four message processing units in a SISO module implement the max-quartet function [18] given by

$$Q(x,y) = \max(x,y) - \max(x+y,0) + \max\left(\frac{5}{8} - \frac{|x-y|}{4}, 0\right) - \max\left(\frac{5}{8} - \frac{|x+y|}{4}, 0\right).$$

It approximates the difference between two logsum operations

$$Q(x,y) \approx \log(e^x + e^y) - \log(e^{x+y} + 1).$$

In terms of $Q(x,y)$, the key equations of the BCJR algorithm simplify to

$$\begin{aligned} \Delta\alpha' &= Q(\Delta\alpha, \gamma - \lambda) \\ \Delta\beta' &= Q(\Delta\beta, \gamma - \lambda) \\ \Lambda &= Q(\Delta\alpha, \Delta\beta) \\ \Gamma &= \Lambda + (\gamma - \lambda). \end{aligned} \tag{1}$$

The logic circuit implementation of the Q-function is described in Section III-A.

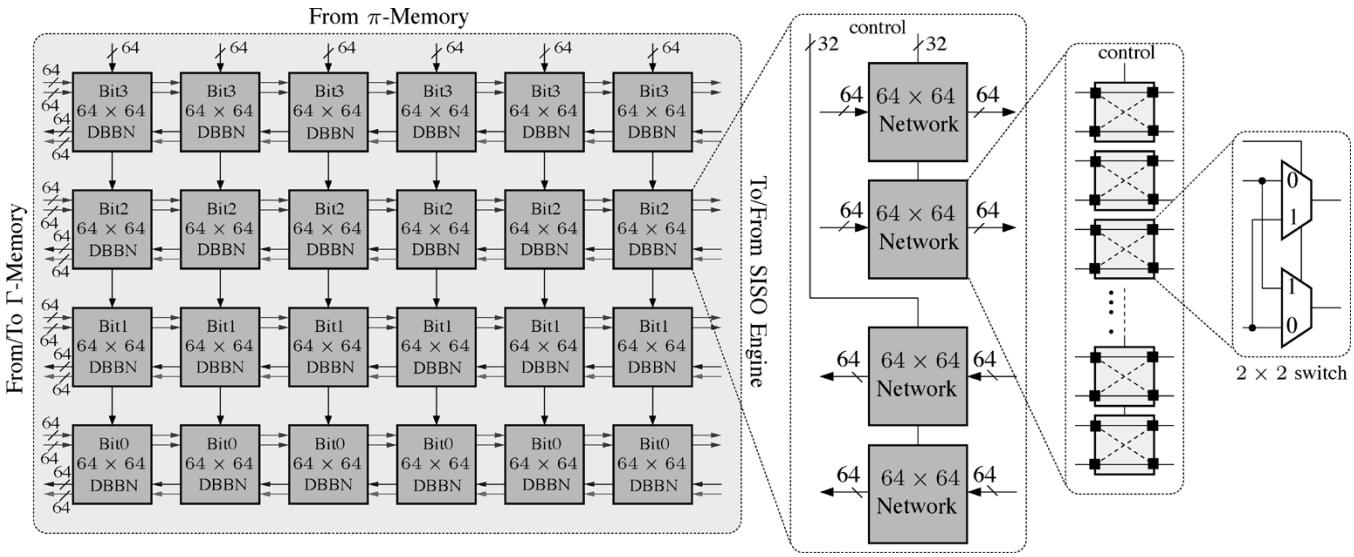

Fig. 7. Architecture of the programmable six-stage 64 × 64 dual bi-directional 4-bit network.

Unlike Gallager's update equations [1] which are complex, nonlinear, and prone to quantization noise, the max-quartet approximation is simple, lookup-table-free, and incurs a decoding loss of <0.05 dB compared to the ideal case [18], [19]. In addition, the area of the MPU is only 4864.1 μm^2 in 0.18-μm CMOS technology, which allows for efficient integration of multiple SISO modules in a small chip area.

E. Bi-Directional Network

From the dataflow graph shown in Fig. 3, input γ-messages to the SISO decoder have to be interleaved and then de-interleaved after decoding according to the permutations of the block rows in $\mathbf{H}$. The interleaver (π) and de-interleaver (π^{-1}) perform 16 types of interleaving patterns depending on the constituent code being decoded. These interleaving operations are performed by the dual bi-directional network shown in Fig. 7. It routes 128 4-bit messages in parallel (each direction) using six layers of shuffle-exchange networks. Each layer consists of four dual bi-directional bit-networks (DBBNs). Each DBBN includes four single-stage networks (two for forward routing and two for backward routing). A single stage network consists of a 64 × 64 switch composed of simple 2 × 2 switches. The switches in these single stage networks are controlled by π-SRAM that stores the routing configurations depending on the LDPC code parameters. Fig. 7 shows the detailed architecture of the dual bi-directional network.

The network has a critical path delay of 2 ns, an average interconnect length of 1 mm, and area of 3.28 mm^2. In [12], the network has a delay of 15.4 ns and an average interconnect length of 3 mm, and occupies 50% of the 52.5 mm^2 overall area, constituting a performance and area bottleneck that renders practical LDPC decoders of codes longer than 1024 more difficult to implement.

F. Chip Characteristics

The chip supports a wide range of length 2048 AA-LDPC codes by programming the code's base parity-check matrix

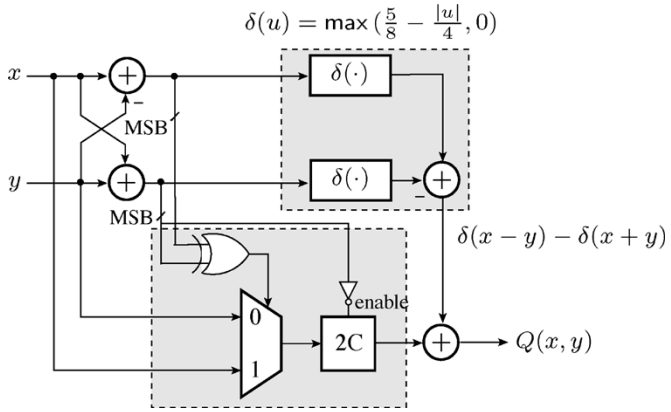

Fig. 8. Logic design of the Q-function block.

($\mathbf{H}$-SRAM) and permutation sub-matrices (π-SRAM). The decoder is capable of processing 128 symbols per clock cycle using 64 parallel SISO modules based on the TDMP architecture, requiring a memory bandwidth of 64 Gb/s. To sustain this bandwidth, messages are routed from Γ-SRAM to the SISO modules and back from the SISO modules to Γ-SRAM using a programmable dual bi-directional routing network with critical path delay of 2 ns. The complexity of this network scales as the square-root of the block length, compared to a linear complexity increase in the case of parallel decoders. A reduced-complexity message processing unit combines fast operation with a decoding loss of <0.05 dB. Multiple SISO modules can be efficiently integrated into the SISO engine using these units, hence increasing the degree of parallelism of the TDMP decoder.

The combination of the TDMP algorithm and architecture with the AA-structure of the code eliminates the interconnect bottleneck of existing parallel architectures [12] that complicates practical decoder implementations of long LDPC codes. In particular, the chip presented here occupies an area 3.7 times smaller, and decodes LDPC codes twice the length (resulting in higher coding gain) compared to the chip presented in [12]. In

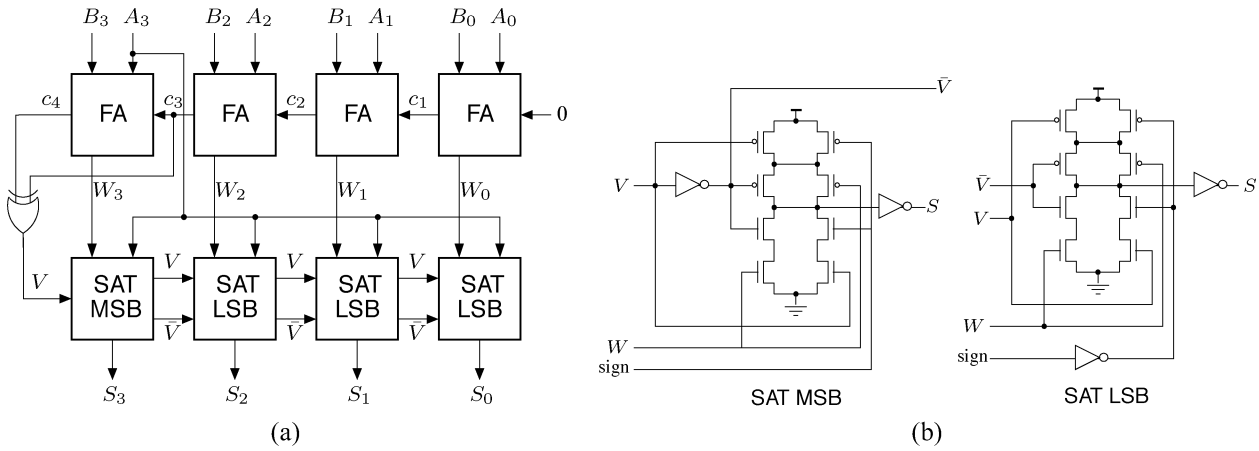

Fig. 9. Four-bit ripple-carry adder with saturation arithmetic. (a) Logic level implementation. (b) Transistor schematic of the saturation logic.

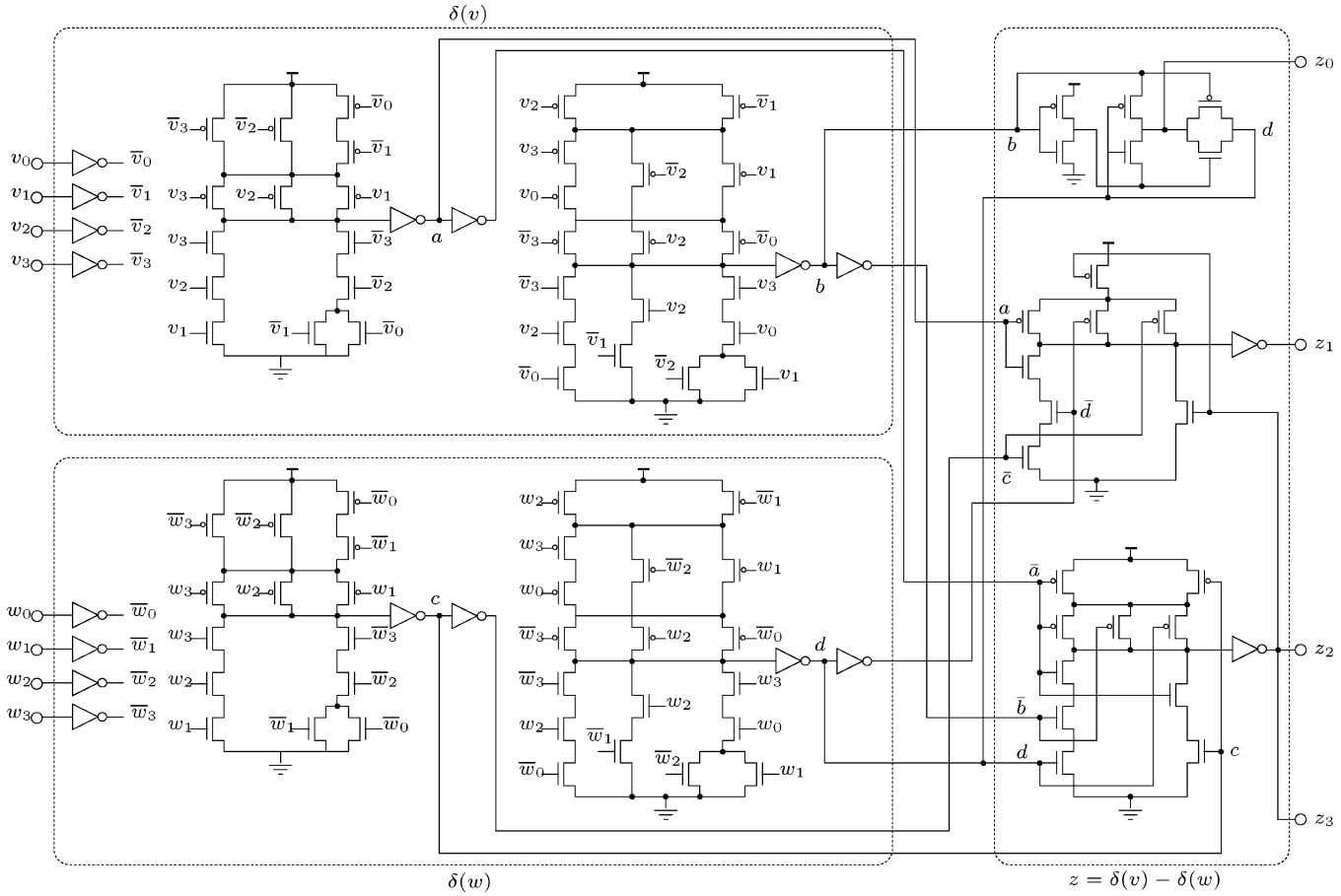

Fig. 10. Transistor-level schematic implementation of the correction factor $\delta(x - y) - \delta(x + y)$.

addition, the average interconnect length of the interconnection network is 1 mm compared to 3 mm in [12]. This allows the chip to run at approximately twice the clock speed (125 MHz), achieving a throughput of 640 Mb/s at 10 decoding iterations. The number of iterations can be adjusted depending on the channel SNR, leading to a throughput of 1.6 Gb/s at 2.6 dB (corresponding to four iterations). The chip requires on average 50% less iterations to converge (see [16]) compared to [12], which implements the TPMP algorithm. Finally, the chip decodes any mix of 2048-bit rate-1/2 (3,6)-AA-LDPC codes by programming the routing network.

III. Circuit Design

This section describes the circuit styles employed in designing the SISO engine, bi-directional network, flip-flops, and memory blocks of the decoder. Global clock and power distribution are then discussed.

A. Logic Style

The combinational logic in the SISO engine was designed using static CMOS circuit style with the pMOS and nMOS transistors ratioed as 2.5:1. Fig. 8 shows the logic schematic of the

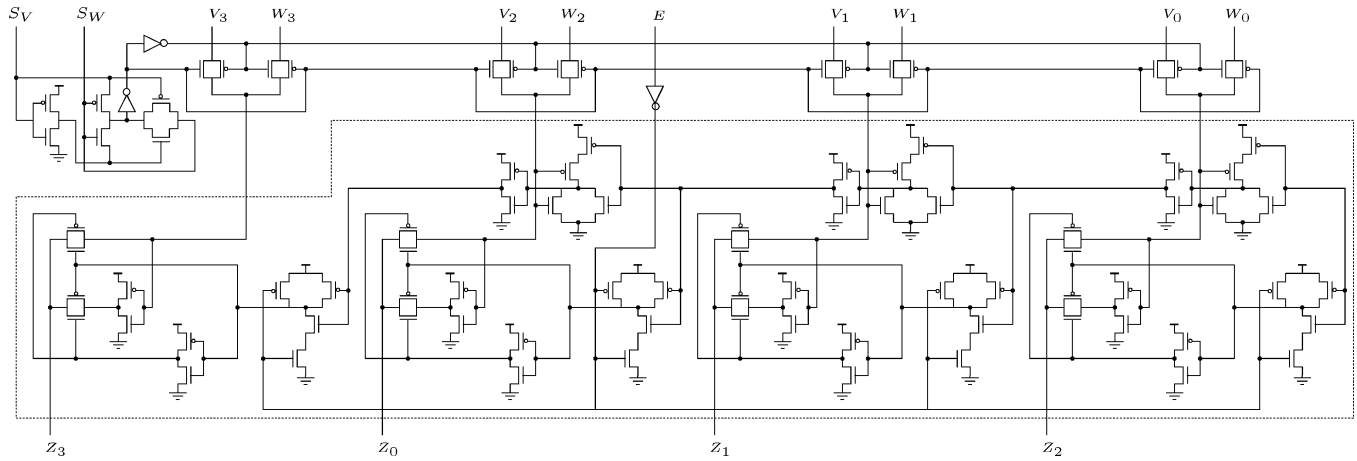

Fig. 11. Transistor schematic of the max-selector and sign-corrector circuit.

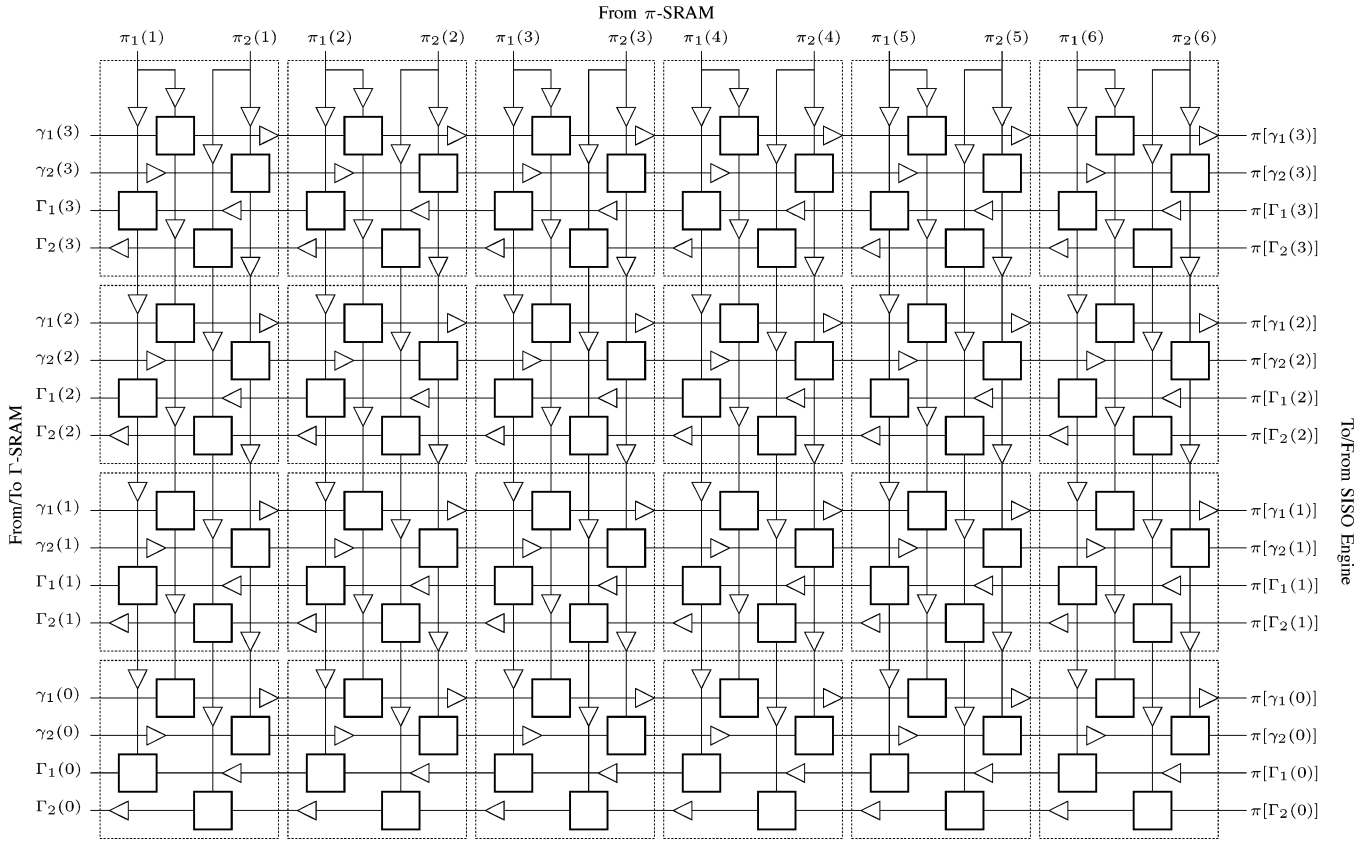

Fig. 12. Illustration of placement of network cells and buffer insertion in the bi-directional network for signal enhancement and delay optimization.

Q-function block used in the SISO modules shown in Fig. 6(a). The 4-bit ripple carry adders in the Q-function block perform saturation addition and subtraction on 4-bit two's complement numbers. The logic level implementation of an adder and the transistor schematic of the saturation logic are shown in Fig. 9.

The δ-blocks in Fig. 8 perform the operation $\delta(u) = \max(5/8 - |u|/4, 0)$. The output of the lower δ-block is subtracted from the upper block to form the correction factor $\delta(x-y) - \delta(x+y)$. Fig. 10 shows the circuit implementing the correction factor $\delta(x-y)-\delta(x+y)$. This correction factor is then added to the maximum of $(x-y)$ and $(x+y)$ after sign correction. Fig. 11 shows the transistor schematic of the max-selector and sign-corrector circuit.

The bi-directional network was implemented using transmission gates with minimum transistor sizing for nMOS and PMOS. Appropriate buffering between network layers was employed for signal enhancement and delay optimization as shown in Fig. 12. Control signals coming from π-SRAM are routed vertically on 6×4 buses. Data signals from Γ-SRAM to the SISO decoder are routed horizontally from left to right on 4×2 buses, while data signals from the SISO engine back to Γ-SRAM and routed horizontally from right to left.

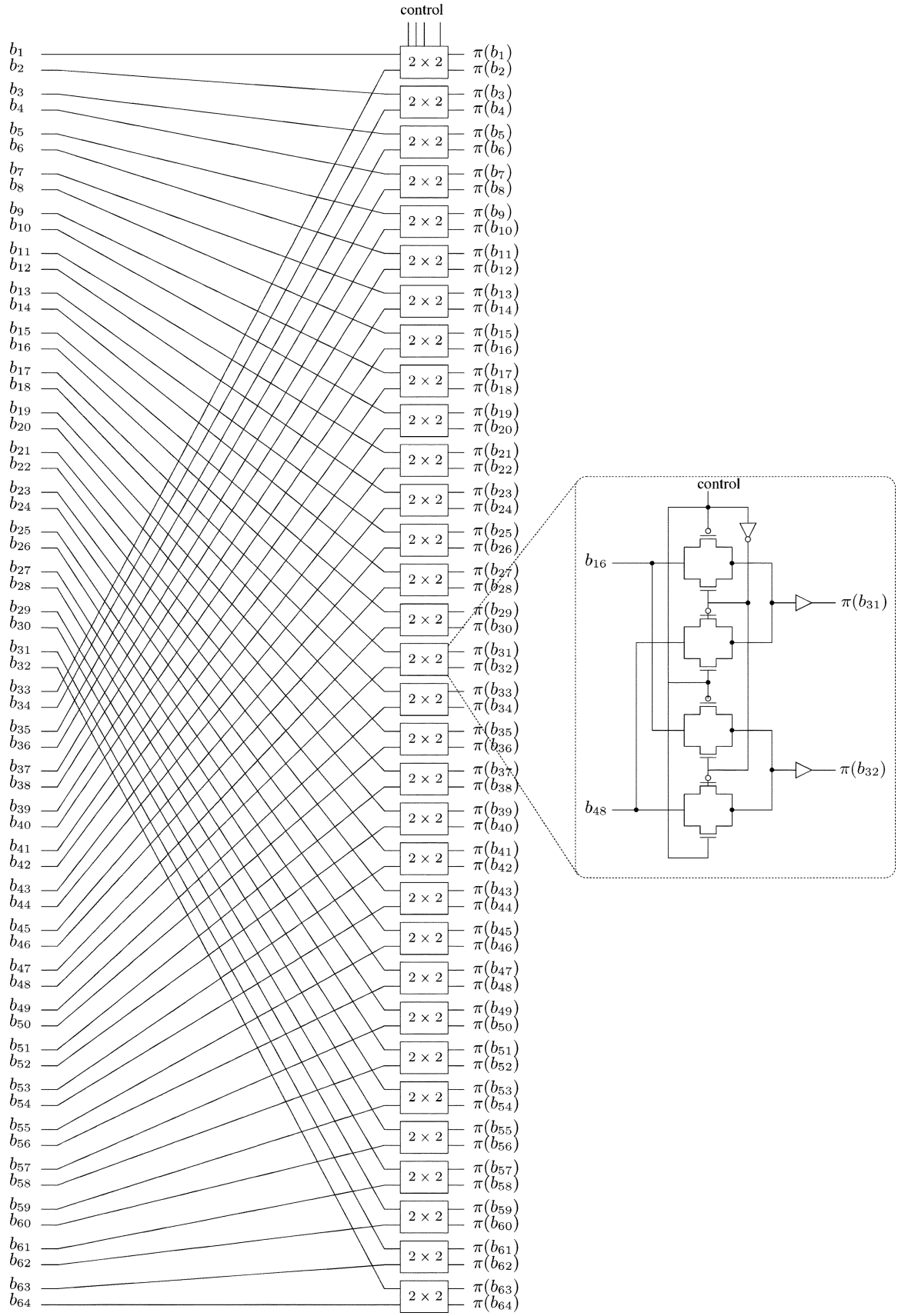

Fig. 13. Circuit implementation of a 64 × 64 network. The transistor schematic of the 2 × 2 switches is shown on the right.

Fig. 13 shows the circuit implementation of a 64 × 64 network. The network performs the shuffle-exchange operation on the 64 inputs using 32 2 × 2 switches. The transistor schematic of a 2 × 2 is shown on the right in Fig. 13.

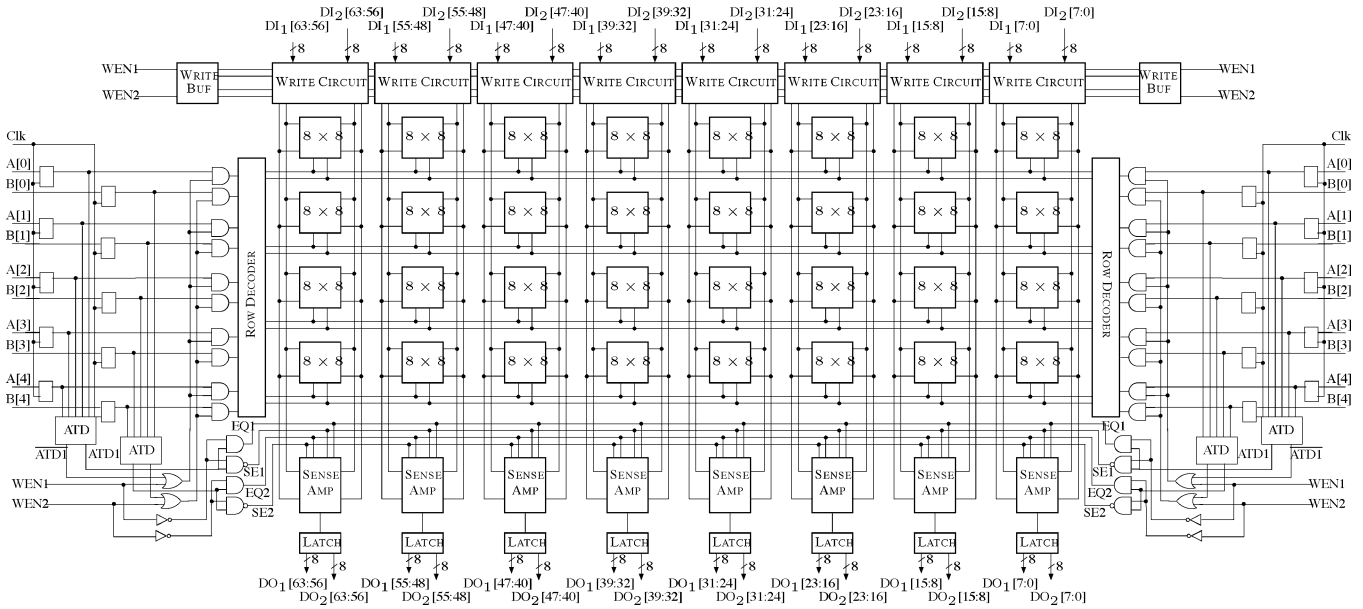

Fig. 14. Architecture of the 32×64 Γ-SRAM module.

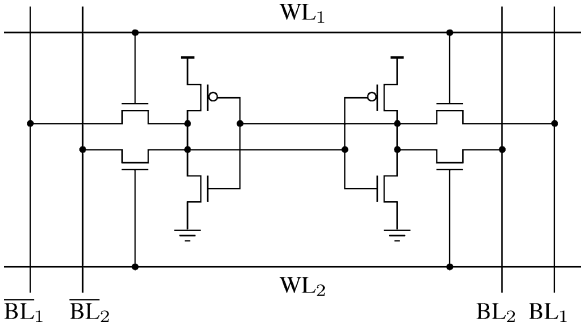

Fig. 15. Transistor schematic of a dual-port SRAM cell.

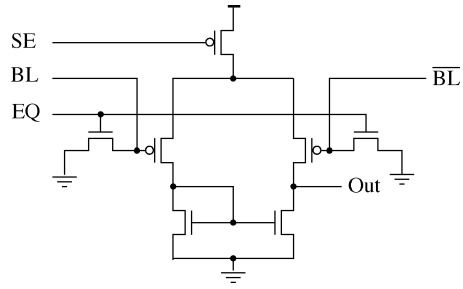

Fig. 17. Transistor schematic of the sense amplifier with sense-enable and equalize signals.

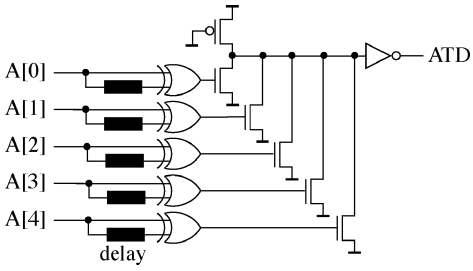

Fig. 16. Address transition detection circuit.

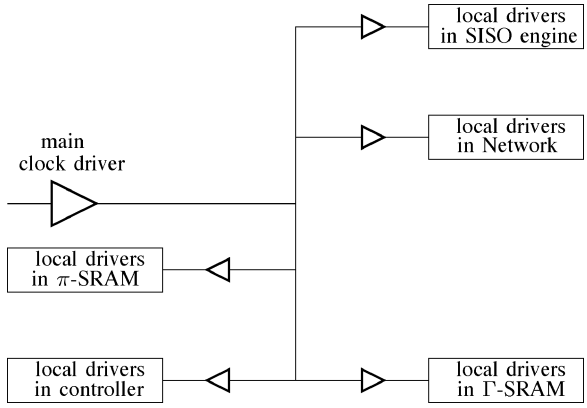

Fig. 18. Clock distribution network using distributed buffering.

B. Flip-Flops and Memory

All flip-flops in the pipeline registers and IO scan registers are positive edge-triggered flip-flops using a single-phase clocking strategy. They were implemented using standard static CMOS master-slave latches.

The various memory blocks of the TDMP decoder were custom designed to meet the required word-lengths and aspect ratios. The single-port $\mathbf{H}$-SRAM and λ-SRAM blocks were designed using standard 6-transistor SRAM cells, while the dual-port Γ-SRAM and π-SRAM blocks were designed using eight-transistor SRAM cells. All memory blocks are self-timed where an address transition on the address lines initiates a read operation. To deliver maximum bandwidth to the appropriate functional blocks, a complete word line is read/written per a read/write memory operation (and hence there is no need for column decoders).

Fig. 14 illustrates the design of the 32×64 Γ-SRAM module. The SRAM cells are placed in a 4-by-8 array of 8×8 cells. To minimize the row-access delay, two row decoders are used to drive the word lines in both directions.

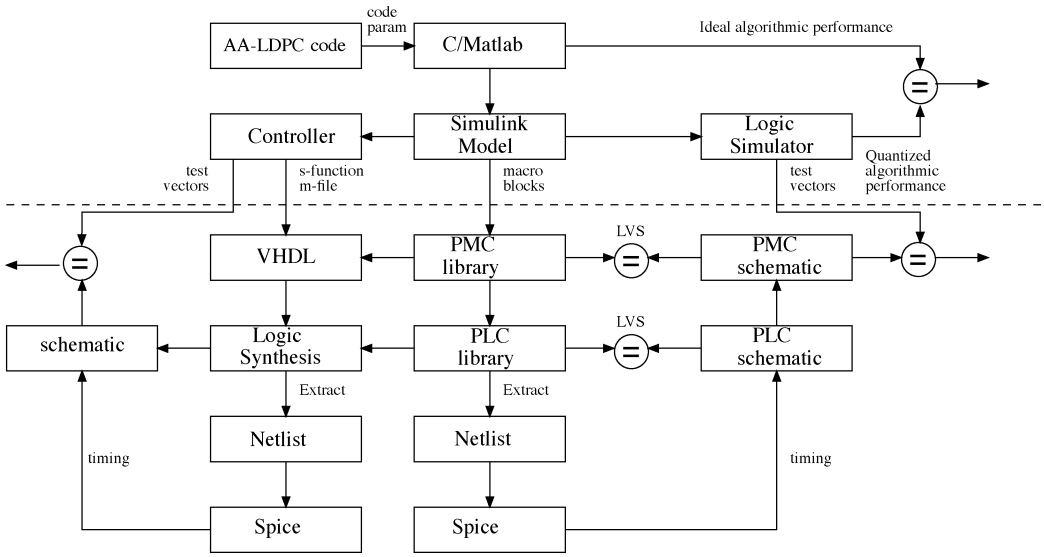

Fig. 19. Hierarchical chip verification model.

Fig. 15 shows the transistor schematic of a dual-port SRAM cell designed using two cross-coupled inverters and four access transistors. The address transition detection (ATD) circuit is shown in Fig. 16. An asserted ATD signal triggers an equalize signal to drain all bit lines, and enables the row decoders to activate the appropriate word line. The sense amplifiers (shown in Fig. 17) are then used to read out the contents of the SRAM cells.

C. Clock and Power Distribution

The clock was distributed using a distributed clock-tree approach as shown in Fig. 18. A maximum simulated clock skew of 100 ps was observed. The clock tree was routed using Metal-6 layer.

The power grid was designed using Metal-1 and Metal-2 layers. To meet metal migration requirements and reduce switching noise, several power and ground pads were provided to the chip. In addition, the power and ground pads of the IO ring were separated from core power and ground pads.

IV. Implementation and Measured Results

The TDMP decoder chip was fabricated in a 0.18-μm 1.8-V six-metal layer CMOS process. The implementation was based on a parameterized core-based design methodology for mapping signal processing/communications algorithms into hardwired reusable cores. The tape-out to the silicon foundry was in GDS-II format.

The design flow is based on a hierarchical parameterized cell layout library that accommodates the main building blocks of the TDMP decoder core [20]. A high-level system model of the decoder based on these building blocks was developed in Matlab and Simulink, and used to generate test vectors for verification and later in chip testing. Fig. 19 shows a block diagram of the hierarchical verification model. Low-level SPICE simulations of all building blocks (including the controllers) were matched with simulation results from the high-level system model. The high-level model also facilitated detailed quantization analysis to determine the optimal tradeoff between BER performance and chip area in sizing the decoder internal data paths.

The chip was tested by simulating all-zero codewords as well as randomly generated noisy codewords under nominal conditions of 25 °C and 1.8 V. The test vectors were generated using the high-level system model with a pre-computed noise component calibrated to specific SNRs. The test vectors were applied to the chip via a logic analyzer, and the test chip outputs were recorded and compared with the ideal outputs.

The chip operates in six phases to decode a frame. These phases correspond respectively to resetting the chip, programming the base matrix, programming the permutation matrices, reading in a new frame, decoding a frame, and reading out a decoded frame. During testing of these phases, a fault was detected in phase 2 which is responsible for programming the bi-directional networks. The fault has to do with the synchronization between filling the input scan buffer to π-SRAM and writing to π-SRAM from the buffer. The operation of this phase was tested using the final pattern written to π-SRAM and the power dissipation was recorded. The remaining phases were fully operational. Note that the operation of the six phases are independent, with each phase having its own local controller.

A bit-accurate model of the decoder chip was developed to simulate BER performance. Each block of the decoder was modeled at the bit-level and tested separately. The models were then integrated together and tested. All simulations and measured results assume an additive white Gaussian noise channel with BPSK modulation.

Fig. 20 shows the simulated BER performance of the TDMP decoder based on the bit-accurate model. The code that was tested is a rate-1/2 length 2048-bit AA-LDPC code (third curve from the bottom in Fig. 20). Also shown in the figure is the ideal BER of a length 2048 LDPC code using the TDMP algorithm with unquantized messages (second curve from the bottom), the BER using Gallager's decoding algorithm using the Q-function for message computations (fourth curve from the bottom), and the frame error rate (FER) curves for the TDMP and Gallager's TPMP algorithm. For comparison, the performance of a rate-1/2

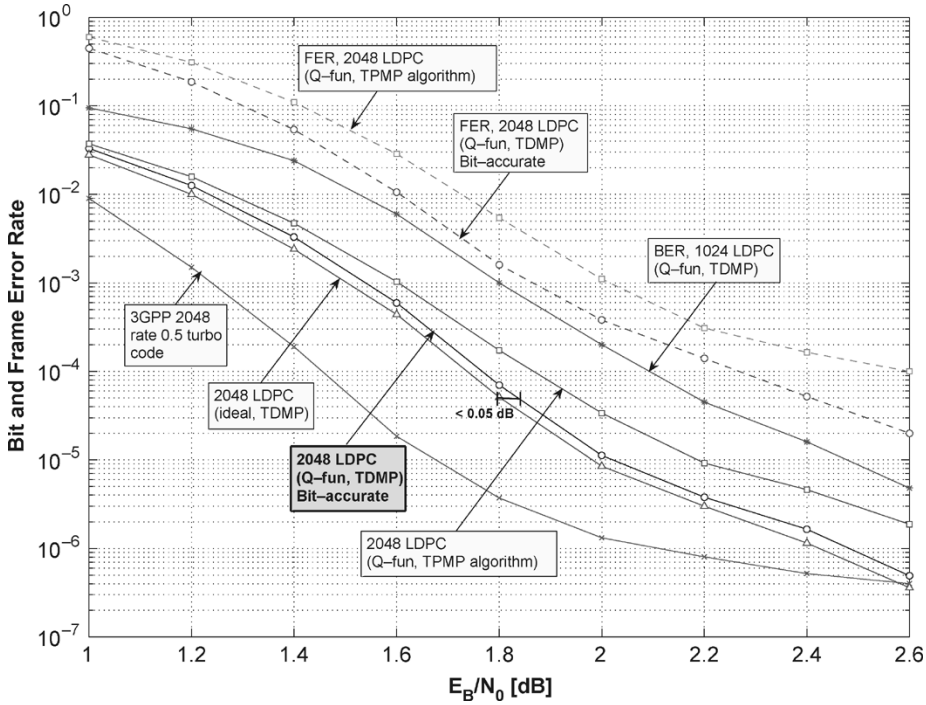

Fig. 20. BER and FER results illustrating the performance of a 2048-bit AA-LDPC code based on a *bit-accurate* simulation model of the decoder chip. Also shown in the figure is the BER and FER of the 2048-bit code using Gallager's TPMP algorithm, ideal BER of the same code assuming unquantized messages, BER of a rate-1/2 1024-bit AA-LDPC code using the TDMP algorithm with the Q-function, and BER of a rate-1/2 2048-bit 3GPP turbo code.

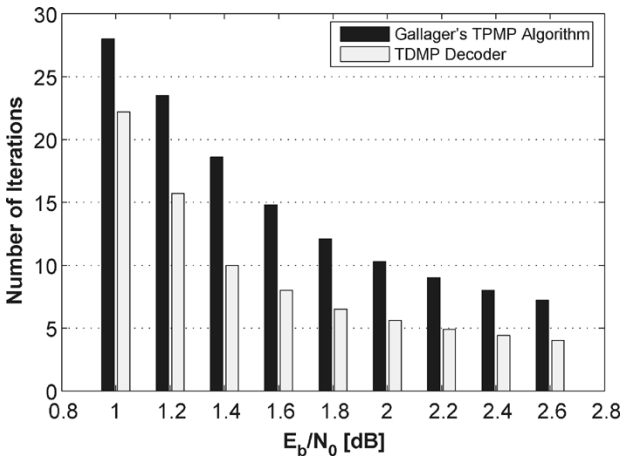

Fig. 21. Comparison of the convergence rate of the TDMP decoder versus Gallager's algorithm as a function of SNR.

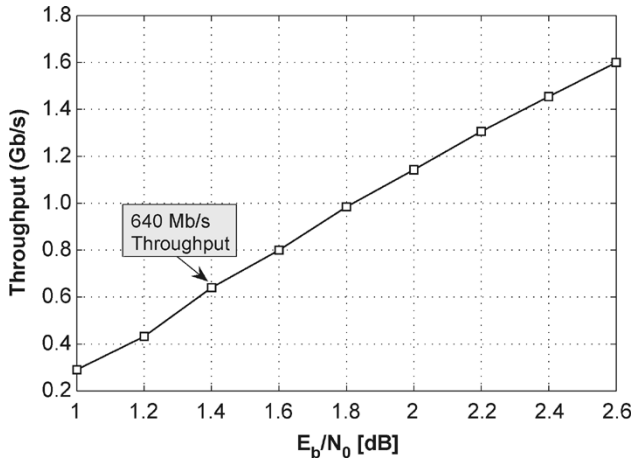

Fig. 22. Average throughput versus SNR at 1.8 V and 125 MHz.

1024-bit LDPC code comparable to [12], and the performance of a rate-1/2 2048-bit 3GPP turbo code are also shown. Decoding of the LDPC codes is performed for a maximum of 32 iterations, or when the codeword has 0 syndrome (above 1.4 dB less than 10 iterations are needed [see Fig. 21]). The proposed LDPC decoder attains more than an order of magnitude improvement in coding gain over the 1024-bit LDPC code, and results in a degradation of <0.05 dB at BER 5×10^{-5} using the Q-function approximation compared to the ideal performance. Finally, the TDMP decoder achieves higher coding gain compared to the TPMP algorithm as shown in the figure.

Fig. 21 compares the average number of iterations of the TDMP decoder versus Gallager's TPMP algorithm as a function of SNR. The TDMP decoder converges in less than half the number of iterations beyond 1.6 dB.

The throughput of the chip can be tuned by controlling the number of decoding iterations to be performed for a given value of SNR based on pre-computed BER curves and average number of iterations required for convergence (unlike [12] where the throughput is fixed by the architecture). Fig. 22 plots the average throughput attained as a function of SNR. The throughput ranges between 278 Mb/s at 1.0 dB up to 1.6 Gb/s at 2.6 dB for a supply of 1.8 V and clock frequency of 125 MHz. At 1.4 dB, the throughput attained is 640 Mb/s.

Fig. 23 shows the measured power dissipation of the chip operating at clock frequencies of 50, 75, 100, and 125 MHz across the six phases of a decoding operation, including programming

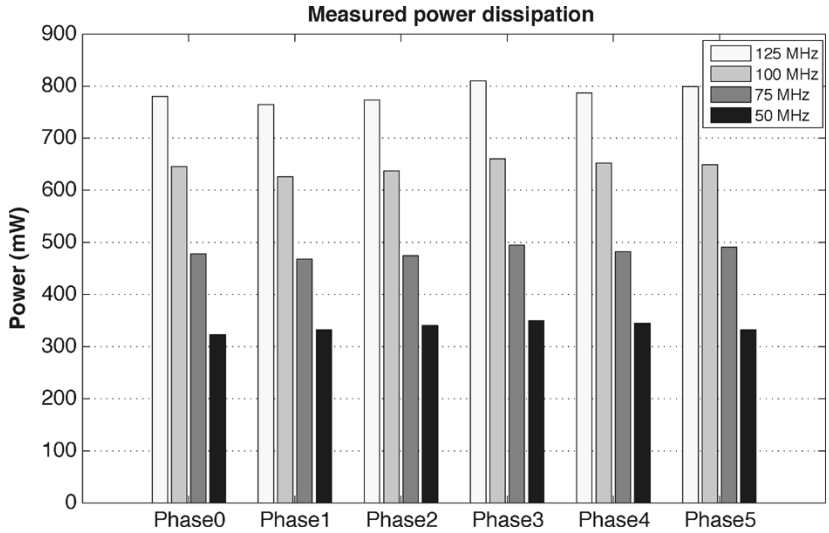

Fig. 23. Measured power consumption of the TDMP decoder chip at 50, 75, 100, and 125 MHz at 1.4 dB.

TABLE I
SUMMARY OF THE AA-LDPC CODE PARAMETERS

Block length	2048
Parity-check equations	1024
Code rate	8/16,9/16,$\cdots$,14/16
Row degree	6
Max. column degree	3
Parity-check matrix size	1024×2048
Base parity-check matrix size	16×32
Size of permutation submatrices	64
Decoding algorithm	TDMP

TABLE II
SUMMARY OF CHIP PARAMETERS

Datapath	4×4-bit buses
Frequency	125 MHz
Throughput	640 Mbps @ 10 iter.
Area	$14.3\,\mathrm{mm}^2$
Power	787 mW @ 1.4 dB
Energy efficiency	123 pJ/Bit/Iter.
Logic gates	220 K
Total memory	51,680 bits
IO pins	76
Technology	$0.18\,\mu\mathrm{m}$, 1.8 V TSMC CMOS

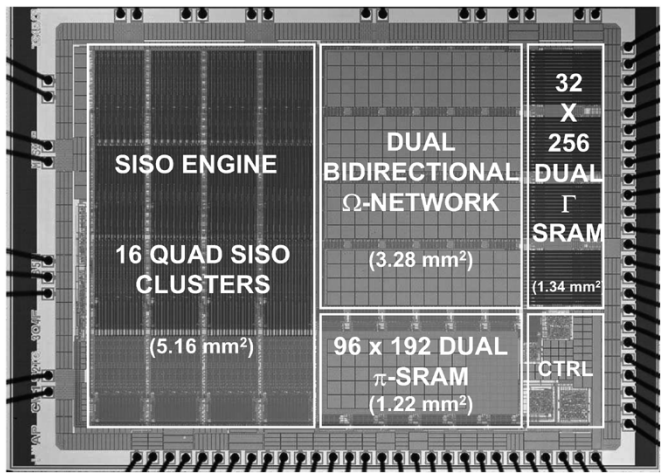

Fig. 24. Die micrograph of the LDPC decoder chip.

of the chip. At 1.4 dB, the average power consumption of the chip operating at 1.8 V and 125 MHz is 787 mW, resulting in an energy consumption per decoded bit per iteration of 123 pJ. The power measurements were performed at the highest clock frequency permitted by the testing equipment.

The decoder chip integrates 220K logic gates and 51 680 bits of memory, and occupies an area of 14.3 mm^2 including the I/O ring, drivers and pads. Other parameters are summarized in Tables I and II. The die micrograph of the chip is shown in Fig. 24. Table III compares the TDMP decoder with state-of-the-art LDPC decoder of [12] and the turbo decoders of [21], [22]. In [23], an 8088-bit rate-1/2 partly parallel LDPC decoder was simulated and synthesized in 0.11-μm CMOS. The reported throughput is 188 Mb/s for 25 iterations and parallelism factor of 24, with a total of 407K memory gates. When normalized to the same code length, the TDMP decoder attains a throughput of 256 Mb/s and requires only 207 488 bits of memory.

V. CONCLUSION

A 14.3-mm^2 code-programmable and code-rate tunable decoder chip for 2048-bit AA-LDPC codes has been presented. The chip dissipates 787 mW of power at 1.4 dB when operated at 125 MHz and 1.8 V, and attains a throughput of 640 Mb/s at 10 decoding iterations. This performance is achieved by implementing the memory-efficient TDMP decoding algorithm which has a faster convergence rate than Gallager's two-phase algorithm.

The interconnection network realizing the Tanner graph of an LDPC code constitutes a throughput bottleneck for designing parallel LDPC decoders for long codes. AA-LDPC code design methodology is an attractive partly parallel decoder design

TABLE III
COMPARISON OF THE TDMP DECODER WITH EXISTING LDPC AND TURBO DECODERS

	TDMP LDPC Decoder	Parallel LDPC Decoder [12]	3GPP-HSDA Turbo Decoder [21]	Parallel Turbo Codec [22]
Throughput	640 Mb/s	1.0 Gb/s	24 Mb/s	75.6 Mb/s
Area	$14.3\,\mathrm{mm}^2$	$52.5\,\mathrm{mm}^2$	$14.5\,\mathrm{mm}^2$	$14.7\,\mathrm{mm}^2$
Memory	51,680 bits	34,816 bits	460,800 bits	36,864 bits
Interleaver	$3.28\,\mathrm{mm}^2$-Network	$26.25\,\mathrm{mm}^2$-Network	192,512-Bit RAM	RAM (size N/A)
Avg. Interconnect Len.	1 mm	3 mm	—	—
Power	787 mW	690 mW	1,450 mW	—
Frequency	125 MHz	64 MHz	145 MHz	170.9 MHz
Energy Efficiency	123 pJ/Bit/Iter.	10.9 pJ/Bit/Iter.	10.0 nJ/Bit/Iter.	2.72 nJ/Bit/Iter.
Block Length	2048	1024	5114	432 (max)
Code Rate	$\frac{8}{16}:\frac{1}{16}:\frac{14}{16}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{3},\frac{1}{2},\frac{2}{3},\frac{3}{4}$
Tunable Code Rate	Yes	No	No	Yes
Code Programmable	Yes	No	No	No
CMOS Technology	0.18 μm, 1.8 V	0.16 μm, 1.5 V	0.18 μm, 1.8 V	0.18 μm, 1.8 V

alternative that decouples the architectural dependence of the decoder from the code properties by transforming the decoder design into a turbo decoding problem while guaranteeing high algorithmic performance and high throughput. The architecture-aware structure of AA-LDPC codes guarantees an efficient implementation of the interconnection network using simple programmable multi-stage networks, and a memory organization that can sustain the required bandwidth as demonstrated by this work. The complexity of this network scales sub-linearly with the code length, allowing for an efficient implementation of high-speed TDMP decoders for long LDPC codes. AA-LDPC codes are attractive candidates for emerging communications standards because of their powerful error-correction capability and their scalable, area-efficient and high-throughput decoder characteristics.

ACKNOWLEDGMENT

The authors would like to thank Dr. Seok-Jun Lee for his assistance during the testing phase of the chip, and Dr. Makram M. Mansour for his support in building the parameterized leaf-cell layout library.

REFERENCES

[1] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.
[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: turbo codes," in *IEEE Int. Conf. Commun.*, 1993, pp. 1064–1070.
[3] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.
[4] D. J. C. MacKay and M. C. Davey, "Evaluation of Gallager codes for short block length and high rate applications," in *Codes, Systems and Graphical Models*, B. Marcus and J. Rosenthal, Eds., 2000, vol. 123, IMA Volumes in Mathematics and Its Applications, pp. 113–130.
[5] Digital Video Broadcasting (DVB-S2) Via Satellite [Online]. Available: http://www.dvb.org
[6] B. Bangerter *et al.*, "High-throughput wireless LAN air interface," *Intel Technol. J.*, vol. 7, no. 3, Aug. 2003. [Online.] Available: http://www.intel.com/technology/itj.
[7] Y. Rashi *et al.*. LDPC: Efficient Alternative FEC for the TFI-OFDM PHY Proposal. IEEE 802.15 Working Group for WPAN. [Online]. Available: http://grouper.ieee.org/groups/802/15/pub
[8] J. Fan. LDPC: Efficient Alternative FEC for the TFI-OFDM PHY Proposal. IEEE 802.20 Mobile Broadband Wireless Access (MBWA). [Online]. Available: http://grouper.ieee.org/groups/802/20/Contribs
[9] H. Song *et al.*, "Low density parity check (LDPC) code concatenated with generalized partial response (GPR) target for high-capacity magneto-optic (MO) recording channels," *Jpn. J. Appl. Phys.*, vol. 41, pp. 1749–1752, Mar. 2002.
[10] B. Vasic *et al.*, "Low-density parity-check codes and iterative decoding for long-haul optical communication systems," *J. Lightw. Technol.*, vol. 21, no. 2, pp. 438–446, Feb. 2003.
[11] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 5, pp. 533–547, Sep. 1981.
[12] C. Howland and A. Blanksby, "Parallel decoding architectures for low density parity check codes," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Sydney, Australia, May 2001, pp. 742–745.
[13] E. Yeo *et al.*, "VLSI architectures for iterative decoders in magnetic recording channels," *IEEE Trans. Magn.*, vol. 37, no. 2, pp. 748–755, Mar. 2001.
[14] M. M. Mansour and N. R. Shanbhag, "Architecture-aware low-density parity-check codes," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, vol. 2, Bangkok, Thailand, May 2003, pp. 57–60.
[15] ——, "On the architecture-aware structure of LDPC codes from generalized Ramanujan graphs and their decoder architectures," in *Proc. 37th Annu. Conf. Inf. Sciences Syst. (CISS'03)*, Baltimore, MD, Mar. 2003, pp. 215–220.
[16] ——, "Turbo decoder architectures for low-density parity-check codes," in *Proc. IEEE GLOBECOM*, Nov. 2002, pp. 1383–1388.
[17] M. M. Mansour, M. M. Mansour, and N. R. Shanbhag, "Design methodology for high-throughput memory-efficient programmable decoder cores for architecture-aware low-density parity-check codes," in *Proc. IEEE Workshop on Signal Process. Syst. (SiPS'03)*, Seoul, Korea, Aug. 2003, pp. 159–164.
[18] 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.
[19] ——, "Low-power VLSI decoder architectures for LDPC codes," in *Proc. ISLPED*, Monterey, CA, Aug. 2002, pp. 284–289.
[20] M. M. Mansour, M. M. Mansour, and A. Mehrotra, "Parameterized macrocells with accurate delay models for core-based designs," in *Proc. IEEE Int. Symp. Qual. Electron. Design (ISQED'03)*, San Jose, CA, Mar. 2003, pp. 319–324.
[21] M. Bikerstaff *et al.*, "A 24 Mb/s radix-4 LogMAP turbo decoder for 3GPP-HSDPA mobile wireless," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, 2003, pp. 150–151.
[22] M. Bougard *et al.*, "A scalable 8.7 nJ/bit 75.6 Mb/s parallel concatenated convolutional (turbo-) codec," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, 2003, pp. 152–153.
[23] Y. Chen and D. Hocevar, "A FPGA and ASIC implementation of rate 1/2, 8088-b irregular low density parity check decoder," in *Proc. IEEE GLOBECOM*, San Francisco, CA, Dec. 2003, pp. 113–117.

Mohammad M. Mansour (S'98–M'03) received the B.E. degree with distinction and the M.E. degree, both in computer and communications engineering, from the American University of Beirut (AUB), Beirut, Lebanon, in 1996 and 1998, respectively, and the M.S. degree in mathematics and the Ph.D. degree in electrical engineering from the University of Illinois at Urbana-Champaign (UIUC), in August, 2002, and May, 2003, respectively.

He is currently an Assistant Professor of electrical engineering with the Department of Electrical and Computer Engineering at AUB. From 1998 to 2003, he was a Research Assistant at the Coordinated Science Laboratory (CSL) at UIUC. In 1997, he was a Research Assistant at the Department of Electrical and Computer Engineering at AUB, and in 1996, he was a Teaching Assistant in the same department. During the summer of 2000, he worked at National Semiconductor Corporation, San Francisco, CA, with the wireless research group. His research interests are VLSI architectures and integrated circuit design for communications and coding theory applications, digital signal processing systems, and general-purpose computing systems.

From 1992 to 1996, Dr. Mansour was on the Dean's honor list at AUB. He received the Harriri Foundation award twice in 1996 and 1998, the Charli S. Korban award twice in 1996 and 1998, the Makhzoumi Foundation Award in 1998, and the Phi Kappa Phi Honor Society award twice in 2000 and 2001. His Ph.D. dissertation was nominated for the ACM Best Dissertation Award by UIUC in 2003.

Naresh R. Shanbhag (M'88–SM'98–F'06) received the B.Tech. degree from the Indian Institute of Technology, New Delhi, India, in 1988, the M.S. degree from the Wright State University, Dayton, OH, in 1990, and the Ph.D. degree from the University of Minnesota, Minneapolis, 1993, all in electrical engineering.

From July 1993 to August 1995, he worked at AT&T Bell Laboratories, Murray Hill, NJ, where he was responsible for the development of VLSI algorithms, architectures and implementation of broadband data communications transceivers. In particular, he was the lead chip architect for AT&T's 51.84-Mb/s transceiver chips over twisted-pair wiring for Asynchronous Transfer Mode (ATM)-LAN and broadband access chip-sets. Since August 1995, he has been with the Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, where he is presently a Professor and the Director of the Illinois Center for Integrated Microsystems. At the University of Illinois, he founded the VLSI Information Processing Systems (ViPS) Group, whose charter is to explore issues related to low-power, high-performance, and reliable integrated circuit implementations of broadband communications and digital signal processing systems. He has published numerous journal articles, book chapters, and conference publications in this area and holds three U.S. patents. He is also a coauthor of the research monograph *Pipelined Adaptive Digital Filters* (Kluwer, 1994).

Dr. Shanbhag received the 2001 IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS Best Paper Award, the 1999 IEEE Leon K. Kirchmayer Best Paper Award, the 1999 Xerox Faculty Award, the National Science Foundation CAREER Award in 1996, and the 1994 Darlington Best Paper Award from the IEEE Circuits and Systems society. From July 1997 to 2001, he was a Distinguished Lecturer for the IEEE Circuits and Systems Society. From 1997 to 1999, he served as an Associate Editor for the IEEE TRANSACTION ON CIRCUITS AND SYSTEMS–PART II: ANALOG AND DIGITAL SIGNAL PROCESSING and the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS.