PAPER Special Section on Multiple-Valued Logic and VLSI Computing

Asynchronous Stochastic Decoding of LDPC Codes: Algorithm and Simulation Model

Naoya ONIZAWA[†a)], ***Member***, **Warren J. GROSS**[††b)], ***Nonmember***, **Takahiro HANYU**[†††c)], ***Member***, ***and*** **Vincent C. GAUDET**[††††d)], ***Nonmember***

SUMMARY Stochastic decoding provides ultra-low-complexity hardware for high-throughput parallel low-density parity-check (LDPC) decoders. Asynchronous stochastic decoding was proposed to demonstrate the possibility of low power dissipation and high throughput in stochastic decoders, but decoding might stop before convergence due to "lock-up", causing error floors that also occur in synchronous stochastic decoding. In this paper, we introduce a wire-delay dependent (WDD) scheduling algorithm for asynchronous stochastic decoding in order to reduce the error floors. Instead of assigning the same delay to all computation nodes in the previous work, different computation delay is assigned to each computation node depending on its wire length. The variation of update timing increases switching activities to decrease the possibility of the "lock-up", lowering the error floors. In addition, the WDD scheduling algorithm is simplified for the hardware implementation in order to eliminate time-averaging and multiplication functions used in the original WDD scheduling algorithm. BER performance using a regular (1024, 512) (3,6) LDPC code is simulated based on our timing model that has computation and wire delay estimated under ASPLA 90nm CMOS technology. It is demonstrated that the proposed asynchronous decoder achieves a 6.4-9.8× smaller latency than that of the synchronous decoder with a 0.25-0.3 dB coding gain.

key words: *forward error correction (FEC), stochastic computation, asynchronous circuits*

1. Introduction

Low-Density Parity-Check (LDPC) [1], [2] codes are very powerful forward-error-correcting codes and have been used in digital communication standards, such as WiMAX [3], WiFi [4], and 10GBASE-T [5]. LDPC decoding using the sum-product algorithm (SPA) or one of its variants, performs by iteratively passing *a posteriori* probability or log-likelihood ratio (LLR) soft-valued messages between two computation nodes on a factor graph [6].

In the hardware implementation based on the SPA and the min-sum algorithm (MSA), parallel decoder implementations tend to achieve high throughput, but the wiring complexity and large area consumption lead to an increase in the maximum wire length, which imposes an upper limit on achievable speed [7]–[10]. Stochastic decoding [11]–[22] has been proposed as an alternative implementation of LDPC decoders to provide very low-complexity hardware. Stochastic computation in essence represents a way of quantizing the amplitude of a signal onto the statistics of a random signal, rather than onto classical binary (e.g. 2's complement) values. This represents an evolution from binary or voltage/current amplitude signalling towards more modern time-based, and ultimately statistics-based signal processing. In stochastic decoding, probability messages represented by the statistics of a *Bernoulli* sequence are serially sent over an interleaver portion between two computation nodes that are simply designed using binary or even multiple-valued logic [17], [18] gates. As relatively large number of clock cycles are required compared with that of the MSA, high-speed clocking is required for high-throughput decoders, causing large power dissipation. For larger codes toward higher throughput demands, such as 40G- and 100GBASE-T, the wiring delay in the interleaver portion tends to have more variations due to the complexity of wiring. It causes very long interconnect, which limits the clock frequency of the synchronous stochastic decoders.

An asynchronous stochastic decoder has a possibility of implementing an energy-efficient high-throughput LDPC decoder due to the lack of clocking. An asynchronous scheduling algorithm and its hardware have been proposed for min-sum decoders [23], [24] and demonstrate high-throughput and low-power LDPC decoders by solving the clock-related problems, while maintaining BER performance [25], [26]. For stochastic decoders, we have presented an asynchronous scheduling algorithm, called a wire-delay independent (WDI) scheduling algorithm [19] that suffers from error floors that also occur in synchronous stochastic decoding. A wire-delay dependent (WDD) scheduling algorithm has been presented [20] in order to reduce the error floors and this paper is the extension. The WDD scheduling algorithm reduces the error floors, but it requires a complex function, such as time-averaging and multiplication functions, which increase the area and the delay time in hardware. In this extension, a simplified WDD (SWDD) scheduling algorithm is presented for the efficient hardware implementation. The SWDD scheduling algo-

Manuscript received November 28, 2013.
Manuscript revised April 9, 2014.

[†]The author is with the Frontier Research Institute for Interdisciplinary Sciences, Tohoku University, Sendai-shi, 980–8578 Japan.
[††]The author is with the Department of Electrical and Computer Engineering, McGill University, Montreal, Quebec, H3A 0E9, Canada.
[†††]The author is with the Research Institute of Electrical Communication, Tohoku University, Sendai-shi, 980–8577 Japan.
[††††]The author is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada.
a) E-mail: nonizawa@m.tohoku.ac.jp
b) E-mail: warren.gross@mcgill.ca
c) E-mail: hanyu@ngc.riec.tohoku.c.jp
d) E-mail: vgaudet@uwaterloo.ca

DOI: 10.1587/transinf.2013LOP0010

rithm exploits a simple addition and a comparison instead of using the complex functions while maintaining the BER performance.

The rest of the paper is organized as follows. A review of stochastic decoding is described in Sect. 2. Section 3 describes asynchronous stochastic decoding based on the WDI scheduling algorithm. Section 4 and Sect. 5 introduce the WDD and the SWDD scheduling algorithms, respectively. Section 6 evaluates BER performance using our simulator that includes computation and wire delay based on a 90nm CMOS technology. Finally, we conclude this paper in Sect. 7.

2. Review of Stochastic Decoding

2.1 Overview of LDPC Codes

Figure 1 shows a factor graph [6] that represents *n* variable and *m* parity-check nodes in an LDPC code. In an LDPC code, (*n*-*m*) information bits are encoded using a generator matrix $\boldsymbol{G}$, and the *n*-bit encoded data *x* containing *m* parity bits are transmitted over the channel, such that $\boldsymbol{GH}^T=0$ with operations typically (but not always) over GF(2), where $\boldsymbol{H}$ is a parity-check matrix. The received data transmitted with channel noises are often decoded using the sum-product algorithm (SPA) or one of its variants, such as the min-sum algorithm (MSA). LDPC decoding performs by iteratively passing *a posteriori* probability or log-likelihood ratio (LLR) messages along the edges of the factor graph between the variable and the check nodes. In the SPA, the outgoing probability in a two-input variable node, which has two inputs (*A* and *B*) and one output (*C*), is represented by

$$p(C) = \frac{p(A)p(B)}{p(A)p(B) + (1 - p(A))(1 - p(B))}. \tag{1}$$

Similarly, the outgoing probability in a two-input check node, which has two inputs (*A* and *B*) and one output (*C*), is represented by

$$p(C) = p(A)(1 - p(B)) + (1 - p(A))p(B). \tag{2}$$

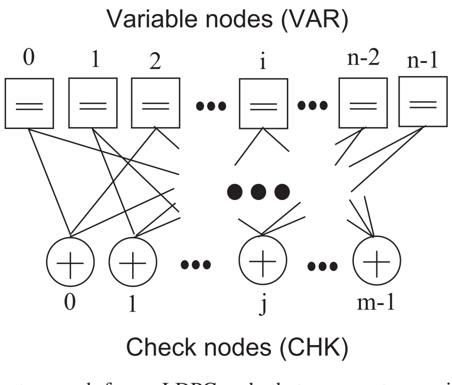

Fig. 1 Factor graph for an LDPC code that represents *n* variable nodes and *m* check nodes.

2.2 Stochastic Decoding

Parallel LDPC decoders based on the MSA have been proposed [10], [24] instead of using the SPA [7], but wiring complexity of interleaver portions and relatively large area of the computation nodes are issues in hardware. In order to realize low-complexity decoders, stochastic decoders were first introduced in [11] and have been demonstrated to lead to simple, yet very high throughput hardware [14], [15] for LDPC codes. Stochastic decoding performs in the probabilistic domain, such as the SPA. The probability *p* is represented by a random sequence of bits that is a *Bernoulli* sequence and corresponds to the frequency of ones or zeros in the sequence. The probability can be represented by many different sequences of bits. For example, different sequences of bits (0011) and (1010) can be $p = 0.5$.

Figure 2 (a) shows a circuit diagram of a two-input stochastic variable node. Let $p(A) = \Pr(A(t)=1)$ and $p(B) = \Pr(B(t)=1)$ be the probabilities represented by the two input bit streams, and $p(C) = \Pr(C(t)=1)$ be the probability represented by the output bit stream. The circuit is designed based on a tracking forecast memory (TFM) method [15]. The TFM consists of a *r*-bit flip-flop (e.g. *r* is 5 to 8) to hold an estimate of the message probability that the precision can be increased with wider widths. It is possible that the width of the TFM can increase the number of iterations if the extra bits are used (hence slowing the convergence, but improving the BER). The TFM and another rerandomization unit, edge memory (EM) [14] generate an output bit stream randomly based on a probability stored in the memory when inputs are not the same. These units increase switching activities of messages to reduce the probability of "lock-up" that stops decoding before decoding convergence. The "lock-up", or also called "latching" occurs because a cycle in the graph, such as Fig. 1, causes a group of nodes to lock into a fixed state (see details in [12]). Large switching activities increase a possibility of breaking the fixed state, lowering the BER.

In the two-input stochastic variable node, Eq. (1) is implemented by the following rules:

$$C(t) = \begin{cases} r(t) & \text{if } U = 1 \\ r'(t) & \text{otherwise} \end{cases} \tag{3}$$

r(t) is an output of the two-input AND gate. *r*'(t) is a randomly selected bit based on a stored probability *P*(t) in the TFM. When U is 1, *P*(t) is updated by comparing *r(t)* with

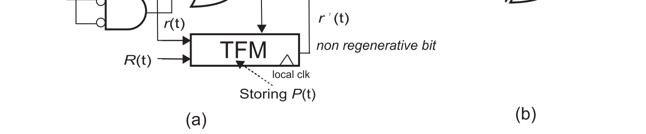

Fig. 2 Circuit diagrams in a stochastic decoder: (a) variable and (b) check nodes.

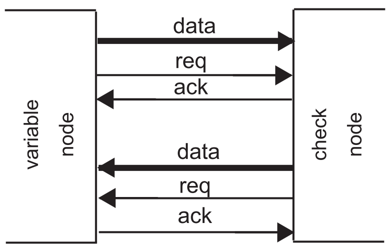

Fig. 3 Asynchronous data transmission between variable and check nodes, where local handshaking is performed using a request and an acknowledgment signal.

R(t) that is a random bit whose probability is 0.5. Figure 2 (b) shows a circuit diagram of a two-input stochastic check node. Equation (2) for the check node is implemented by the following rule:

$$C(t) = A(t) \bigoplus B(t). \tag{4}$$

Decoding performs by iteratively passing stochastic messages that are updated in variable and check nodes along the edges of the factor graph for a fixed number of cycles, known as decoding cycles [14], [15], or until a codeword is found.

3. Asynchronous Stochastic Decoding

3.1 Asynchronous Scheduling Algorithm

Stochastic decoders are usually implemented using synchronous circuits, where all the variable-node outputs are updated simultaneously using a global clock signal. However, the wiring delay via an interleaver tends to have variations from wire to wire due to the complexity in the chip implementation causing very long interconnects, which limits the clock frequency. We have previously proposed an asynchronous scheduling algorithm to alleviate this problem for the MSA [25], [26].

Asynchronous scheduling algorithms for stochastic decoding of LDPC codes have been proposed to tackle the wiring complexity, especially for clock distribution [19], [20]. In the scheduling algorithms, computation nodes do not need to wait for the all updated messages including from tardy long wires, but can proceed when ready as node computations are performed by using the most recent available messages rather than all updated messages at a global level. The transmission delays between the computation nodes are determined by each wire delay, and thus the decoding convergence and hence throughput is governed by the average (rather than worst-case) wire delays of the interleaver.

Figure 3 shows asynchronous data transmission between one variable node and one check node. Data (an stochastic stream) is transmitted using local control signals, such as a request and an acknowledge signal, labeled as req and ack, respectively. First, the request signal, req, is asserted to transmit data to the check node. Second, the check node detects the request signal and then receives the data. Third, the check node transmits the acknowledge signal, ack, to the variable node to inform that the data is received. Finally, the variable node detects the ack signal and then goes back to the first step to transmit the next data. In this way, the transmission delay via the interleaver is determined by local wiring delay characteristics as the asynchronous circuits are controlled by a request-acknowledge-based handshaking protocol,

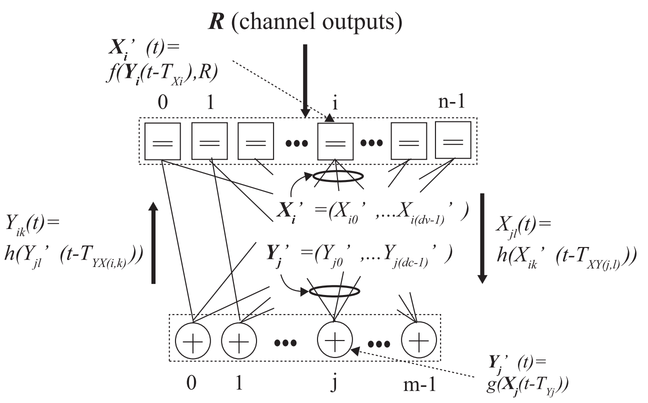
$\boldsymbol{R}$ (channel outputs)
$\boldsymbol{X}_i'(t)= f(\boldsymbol{Y}_i(t-T_{Xi}),R)$
0
1
i
n-1
$\boldsymbol{X}_i' =(X_{i0}', ...X_{i(dv-1)}')$
$\boldsymbol{Y}_j' =(Y_{j0}', ...Y_{j(dc-1)}')$
$Y_{ik}(t)= h(Y_{jl}'(t-T_{YX(i,k)}))$
$X_{jl}(t)= h(X_{ik}'(t-T_{XY(j,l)}))$
0
1
j
m-1
$\boldsymbol{Y}_j'(t)= g(\boldsymbol{X}_j(t-T_{Yj}))$

Fig. 4 Timing model for asynchronous stochastic decoding.

Our goal is to implement an energy-efficient high-throughput stochastic LDPC decoder based on asynchronous scheduling. A BER simulation is required before decoder implementation to validate decoding and scheduling algorithms. In a synchronous stochastic decoder, all computation blocks operate at the same time, thus the BER simulation is straightforward to implement. However, in the asynchronous stochastic decoder, computation blocks and data-transmission blocks operate at different speed based on each individual delay. To validate the asynchronous scheduling algorithms, we need a complex timing model that considers all delay information.

3.2 Wire-Delay Independent (WDI) Scheduling

In this subsection, we describe a timing model for BER simulations based on an asynchronous scheduling algorithm called a wire-delay independent (WDI) scheduling [19]. Figure 4 shows a timing model for the asynchronous scheduling algorithms. Suppose that incoming and outgoing messages to and from check (CHK) nodes are represented by vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively, where these vectors have length n and m, respectively. The operations of variable (VAR) and CHK nodes are represented by

$$\begin{cases} \text{VAR} : \boldsymbol{X} = f(\boldsymbol{Y}, \boldsymbol{R}), \\ \text{CHK} : \boldsymbol{Y} = g(\boldsymbol{X}), \end{cases} \tag{5}$$

where f and g represent the VAR and the CHK operations, respectively. $\boldsymbol{R}$ is a received channel output whose vector has length n. The operations f and g correspond to Eqs. (1) and (2) in probabilistic domain, respectively. The hardware implementations of f and g are described in Fig. 2 (a) and (b), respectively.

Table 1 Delay time that determines update timing in synchronous, asynchronous and clockless stochastic decoding.

	Synchronous	Asynchronous (WDI)	Asynchronous (WDD)	Clockless
VAR	T_{Sync}	T_{VAR}	$\alpha * avg(T_{YX_{(i,0)}}, \dots, T_{YX_{(i,(dv-1))}})$	-
CHK	T_{Sync}	T_{CHK}	$\alpha * avg(T_{XY_{(j,0)}}, \dots, T_{XY_{(j,(dc-1))}})$	-
Interleaver	T_{Sync}	$min(T_{XY}) - max(T_{XY})$	$min(T_{XY}) - max(T_{XY})$	-

The timing model includes three blocks that are VAR, CHK, and data transmission (DT) between VAR and CHK. These three blocks operate at different timing i.e. asynchronously. The output of each VAR operation, $\boldsymbol{X'_i}$ $(0 \leq i < n)$ with the update timing is given by:

$$\boldsymbol{X'_i(t)} = \begin{cases} f(\boldsymbol{Y_i}(t - T_{Xi}), R_i), & \text{if } t = rT_{Xi} \\ \text{hold.} & \text{otherwise} \end{cases} \quad (6)$$

$\boldsymbol{X_i}$ is composed of X_{ik} and $\boldsymbol{Y_i}$ is composed of $Y_{ik}(0 \leq k < dv)$, where dv is the number of inputs from CHK at each VAR. $\boldsymbol{T_X}$ represents the VAR computation delay time, which is represented by a vector $(T_{X0}, \dots, T_{X(n-1)})$ and r is an integer value. Each VAR is updated using a local control signal that can be generated by a ring oscillator. $\boldsymbol{X'_i}$ is updated if the simulation time is equal to rT_{Xi}. Otherwise, the output holds the previous value.

The output of each CHK operation, $\boldsymbol{Y'_j}$ $(0 \leq j < m)$ with the update timing is given by:

$$\boldsymbol{Y'_j(t)} = \begin{cases} g(\boldsymbol{X_j}(t - T_{Yj})), & \text{if } t = rT_{Yj} \\ \text{hold.} & \text{otherwise} \end{cases} \quad (7)$$

$\boldsymbol{Y_j}$ is composed of Y_{jl} and $\boldsymbol{X_j}$ is composed of $X_{jl}(0 \leq l < dc)$, where dc is the number of inputs at each CHK. $\boldsymbol{T_Y}$ represents the CHK computation delay time, which is represented by a vector $(T_{Y0}, \dots, T_{Y(m-1)})$.

The output of each DT operation from VAR to CHk, X_{jl} and the output from CHK to VAR, Y_{ik} with the update timing are given by:

$$X_{jl}(t) = \begin{cases} h(X'_{ik}(t - T_{XY(j,l)})), & \text{if } t = rT_{XY(j,l)} \\ \text{hold,} & \text{otherwise} \end{cases} \quad (8)$$

$$Y_{ik}(t) = \begin{cases} h(Y'_{jl}(t - T_{YX(i,k)})), & \text{if } t = rT_{YX(i,k)} \\ \text{hold,} & \text{otherwise} \end{cases} \quad (9)$$

where $T_{XY(j,l)}$ and $T_{YX(i,k)}$ represent data-transmission delay time from the k-th output of the i-th VAR to the l-th input of the j-th CHK and from the l-th output of the j-th CHK to the k-th input of the i-th VAR, respectively. Suppose that $T_{XY(j,l)}$ and $T_{YX(i,k)}$ are the same delay time. h is a mapping function that copies an input to an output in a module.

4. Wire-Delay Dependent (WDD) Scheduling for Asynchronous Stochastic Decoding

4.1 Error Floors in WDI Scheduling

In [19], the asynchronous stochastic LDPC decoder based on the WDI scheduling algorithm achieves up to 7.37× improvement of throughput estimated in the simulation model compared to that of the synchronous stochastic decoder. However, it causes error floors due to the "lock-up", such as the synchronous stochastic decoders. We have also proposed clockless stochastic decoding that is another type of asynchronous stochastic decoding [21]. It is not governed by any global and local control signals and hence computation nodes operate when incoming messages sent along wires are changed. Unlike WDI asynchronous stochastic decoding, clockless stochastic decoding reduces error floors and achieves comparable BER performance to the SPA. The difference between WDI asynchronous and clockless stochastic decoding is the update timing in the computation nodes.

4.2 WDD Scheduling

Despite of the good BER performance of clockless stochastic decoding, it is hard to implement the clockless stochastic decoder due to the lack of the hardware implementation [22]. Unlike the clockless stochastic decoder, the hardware implementation of the asynchronous circuits have been presented [25]–[27]. A wire-delay dependent (WDD) scheduling algorithm is the extension of the WDI scheduling algorithm in order to lower the error floors. The scheduling algorithms are summarized in Table 1.

The VAR operation in the WDD scheduling algorithm is the same as that in Eq. (6), but the update timing ($\boldsymbol{T_X}$) is different. In the WDI scheduling algorithm, as all VARs are updated based on the same cycle delay, the update timing (T_{X_i}) at each node is fixed to T_{VAR} and is defined by

$$T_{X_i} = T_{VAR}. \quad (10)$$

In the WDD scheduling algorithm, update timing at each VAR is determined by averaging delay time of data transmission from CHKs over wires that connect to the VAR. The i-th variable node has dv wires on which data-transmission delay time is $T_{YX_{(i,k)}}$. The assignment of different update timing at each VAR makes a similar characteristic of updating in clockless stochastic decoding that updates the output depending on just input-message changes. The update timing in the WDD scheduling algorithm is given by:

$$T_{X_i} = \alpha \times avg(T_{YX_{(i,0)}}, \dots, T_{YX_{(i,(dv-1))}}), \quad (11)$$

where α is a variable.

The CHK operation in the WDD scheduling algorithm is the same as that in Eq. (7). In the WDI scheduling algorithm, as all CHKs are updated based on the same cycle delay, T_{Y_j} at each node is defined by

$$T_{Y_j} = T_{CHK}. \quad (12)$$

In the WDD scheduling algorithm, update timing at each CHK is determined by averaging delay time of data transmission from VARs over wires that connect to its CHK. The j-th check node has dc wires on which data-transmission delay is $T_{XY_{(j,l)}}$. T_{Y_j} in the WDD scheduling algorithm is given by:

$$T_{Y_j} = \alpha \times avg(T_{XY_{(j,0)}}, \ldots, T_{XY_{(j,(dc-1))}}). \tag{13}$$

The DT operation is the same in both WDI and WDD scheduling algorithms.

The update timing of the WDD scheduling algorithm is different at each computation node while that of the WDI scheduling algorithm is the same at all computation nodes shown in Table 1. The WDD update timing is similar to that of the clockless scheduling algorithm as the updated timing depends on the data-transmission delay [21], [22]. The difference between the WDD and the clockless scheduling algorithms is the data transmission. The data transmission in the WDD is controlled by local handshaking between VARs and CHKs, but it is not in the clockless scheduling algorithm.

4.3 Combination of WDI and WDD Scheduling Algorithms

We also introduce another scheduling algorithm based on the combination of the WDI and the WDD. In the scheduling algorithm, at first, decoding performs based on the WDI or the WDD until a specific decoding time, and then it performs based on the WDD or the WDI. The algorithm would make more switching activities of messages than the WDD and reduce the probability of error floors. Suppose that T_{DCT} is total decoding time. It is represented by

$$T_{DCTI} = \gamma \times T_{DCT}, \tag{14}$$

$$T_{DCTD} = (1 - \gamma) \times T_{DCT}, \tag{15}$$

where T_{DCTI} and T_{DCTD} are decoding time for the WDI and the WDD, respectively.

5. Simplified WDD (SWDD) Scheduling

In the WDD scheduling algorithm, a different timing update at each computation node is realized using the average data-transmission delay information that related to the computation nodes. However, a time-averaging circuit and a multiplier are required in hardware for the WDD scheduling algorithm. The hardware tends to be complex, which increases the area and the delay time at each computation node. Hence, we simplify the WDD scheduling algorithm for the efficient hardware implementation of the asynchronous stochastic decoder.

In the simplified WDD (SWDD) scheduling algorithm, the time-average function and the multiplication are removed. The update timing is determined by comparing the number of transmitted data with a threshold value. The number of transmitted data is counted using asynchronous request and acknowledge signals. The number of transmitted data in the i-th VAR ($N_{VARi}(t)$) is given by:

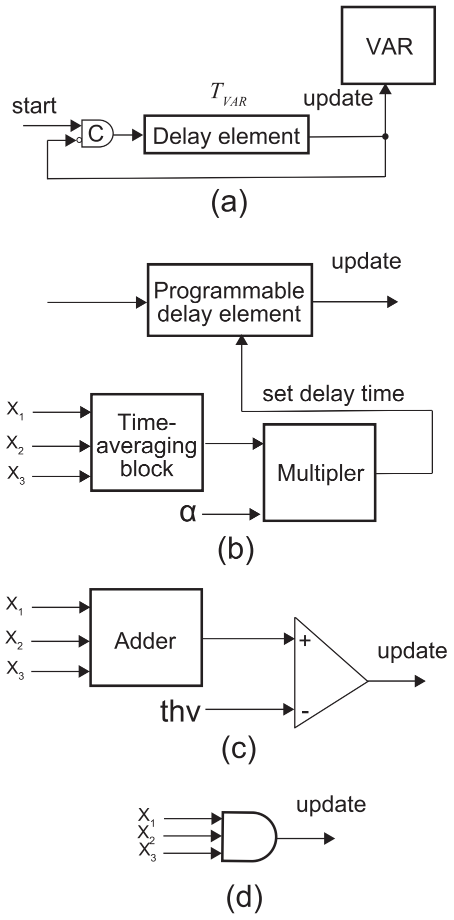

Fig. 5 Hardware architectures of a controller for a three-input VAR based on : (a) WDI scheduling, (b) WDD scheduling, (c) SWDD scheduling, and (d) SWDD scheduling if *thv* is *dv*.

$$N_{VARi}(t) = \begin{cases} N_{VARi}(t) + 1, & \text{if } t = rT_{YX_{(i,k)}} \\ 0, & \text{else if } N_{VARi}(t) \geq thv \\ \text{hold}, & \text{otherwise} \end{cases} \tag{16}$$

where thv ($1 \leq thv \leq dv$) is the threshold value in the VAR. Let $\boldsymbol{N_{VAR}}(t)$ be the number of transmitted data represented by a vector $(N_{VAR0}(t), \ldots, N_{VAR(n-1)}(t))$. The output value of each VAR is updated as follows:

$$\boldsymbol{X_i'(t)} = \begin{cases} f(\boldsymbol{Y_i}(t - T_{Xi}), R), & \text{if } N_{VARi}(t) \geq thv \\ \text{hold}, & \text{otherwise} \end{cases} \tag{17}$$

where T_{Xi} can be a different value at each variable node depending on the data-transmission delay to the variable node.

Figure 5 shows hardware architectures of a controller for a three-input VAR. In the WDI scheduling algorithm, the controller can be designed using a delay element and a C-element shown in Fig. 5 (a). The C-element is a typical storage component for asynchronous circuits. The output is the same value of the inputs if the two inputs are the same. Otherwise, the output holds the previous value. The delay time is set to T_{VAR}. In the WDD scheduling algorithm, the controller would be very complex because of using the time-averaging function block and the multiplier shown in Fig. 5 (b). In the time-averaging function block, dv delay elements are required to detect the data-transmission delay time. In addition, an adder and a divider are necessary to average the data-transmission delay time. After the multiplication of α and the output of the time-averaging block, the output of the multiplier sets the delay time of the programmable delay element. In the SWDD scheduling algorithm, the controller can be designed using an adder and a comparator based on Eqs. (16) and (17) shown in Fig. 5 (c). The adder includes three 1-bit inputs and hence it can be designed using a full adder. The comparator includes two 2-bit inputs and one 1-bit output when dv is 3. If thv is fixed to dv, the controller can be designed using just an AND gate as shown in Fig. 5 (d).

The CHK operation is also described, such as the VAR operation. The number of transmitted data in the j-th CHK ($N_{CHKj}(t)$) is given by:

$$N_{CHKj}(t) = \begin{cases} N_{CHKj}(t) + 1, & \text{if } t = rT_{XY(j,l)} \\ 0, & \text{else if } N_{CHKj}(t) \geq thc \\ \text{hold}, & \text{otherwise} \end{cases} \tag{18}$$

where thc ($1 \leq thc \leq dc$) is a threshold value in the CHK. Let $\boldsymbol{N_{CHK}(t)}$ be the number of transmitted data represented by a vector $(N_{CHK0}(t), \ldots, N_{VAR(m-1)}(t))$. The output value of the CHK is updated as follows:

$$\boldsymbol{Y_j}'(t) = \begin{cases} g(\boldsymbol{X_j}(t - T_{Yj})), & \text{if } N_{CHKj}(t) \geq thc \\ \text{hold}, & \text{otherwise} \end{cases} \tag{19}$$

where T_{Yj} can be a different value at each check node depending on the data-transmission delay to the check node.

6. Evaluation

We use our BER simulator for asynchronous stochastic decoding. A regular (1024, 512) (3,6) LDPC code is used for evaluation of BER performance, where dv is 3 and dc is 6. Suppose that the stochastic decoders are implemented based on ASPLA 90nm CMOS technology. We assume that data-transmission delay distribution is symmetric truncated Gaussian from 100 to 1000 ps, where the variance σ is set to 223 ps. Suppose that the data-tranmission delay consists of wire delay and gate delay related to the data transmission, such as buffers. The VAR computation delay T_{VAR} for the WDI scheduling algorithm is 380 ps. Also, assume that the CHK computation delay T_{CHK} for the WDI is 250 ps. Those values estimated by HSPICE simulation are minimum values that achieve good BER performance [19]. In the stochastic variable node, the memory size of the TFM is set to 5 bits and the type of the TFM is a reduced complexity architecture [15]. Noise dependent scaling is set to 1.5 [14], [15]. The BER results are obtained with 100 frame errors

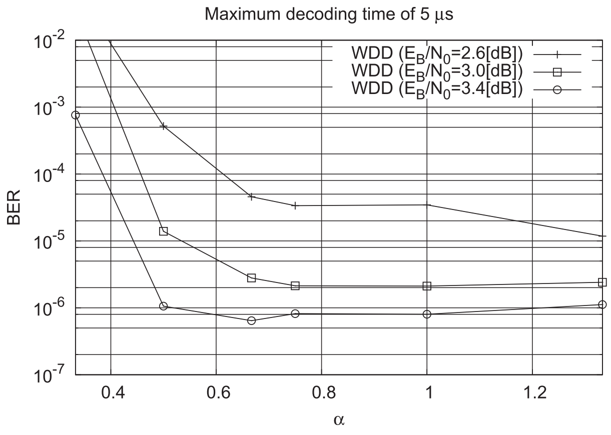

Fig. 6 Simulated BER results of a regular (1024,512) LDPC code depending on α in the WDD scheduling algorithm.

under binary-phase shift-keying (BPSK) modulation on an additive white Gaussian noise (AWGN) channel. Decoding terminates when the decoding latency reaches a maximum decoding cycle (DC) count for synchronous stochastic decoding, a maximum decoding time for asynchronous stochastic decoding, or a maximum number of iterations for the SPA with the syndrome checking as an early stopping method. BER performance is evaluated on a 2.6 GHz Opteron 6282 SE server.

6.1 BER on Asynchronous Stochastic Decoding

Figure 6 shows BER performance depending on α based on the WDD scheduling algorithm. The maximum decoding time T_{DCT} is 5 μs. In a region of small α, small T_X causes unnecessary updating of rerandomization units, such as EMs or TFMs in the VARs and hence induces bad BER performance. This explanation is described in detail in [19]. The minimum BERs are obtained by different α depending on E_b/N_0. At E_b/N_0 = 3.4 [dB], α=2/3 is the value to achieve the minimum BER.

Figure 7 shows BER performance depending on γ based on the combination of the WDI and the WDD scheduling algorithms. In this simulation, at first, the WDI is used until T_{DCTI}, then the WDD is used until T_{DCT}. α is set to 2/3. $\gamma = 0$ indicates that decoding performs based on only the WDD. $\gamma = 1$ indicates that decoding performs based on only the WDI. At E_b/N_0 = 2.6 [dB], the combination of the WDI and the WDD is not effective to reduce BER. In contrast, at E_b/N_0 = 3.0 and 3.4 [dB], γ = 1/10 is the value to achieve the minimum BERs.

Figure 8 shows BER performance depending on γ based on the combination of the WDD and the WDI scheduling algorithms. The order of using the two algorithms is contrary to the previous one. At first, the WDD is used until T_{DCTD}, then the WDI is used until T_{DCT}. Similar to the previous results, at E_b/N_0 = 2.6 [dB], the combination of the WDI and the WDD updating causes worse BER than that using only the WDI updating. At E_b/N_0 = 3.0 and 3.4 [dB], γ = 1/10 is the value to achieve the minimum BERs.

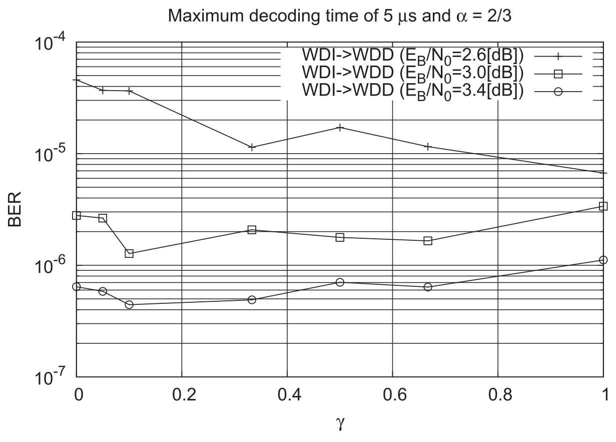

Fig. 7 Simulated BER results of a regular (1024,512) LDPC code depending on γ in the combination of the WDI and the WDD scheduling algorithms.

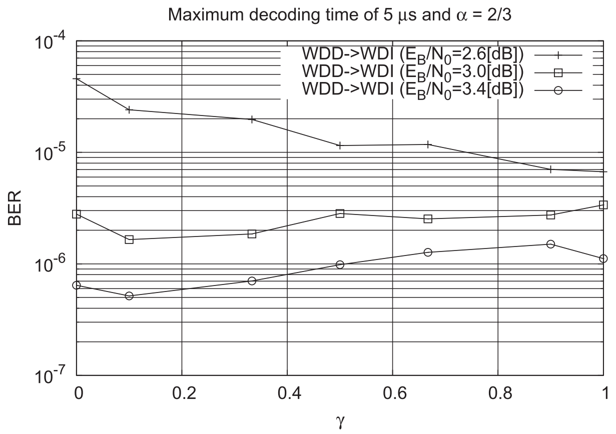

Fig. 8 Simulated BER results of a regular (1024,512) LDPC code depending on γ in the combination of the WDD and the WDI scheduling algorithms.

From these results in Figs. 7 and 8, the combination of the WDI and the WDD is effective to reduce BERs at a high SNR region, but the BER performance is not affected by the order of the usage of these two algorithms.

Figure 9 shows BER performance based on the SWDD scheduling algorithm. When *thv* is equal to 1, decoding is not properly performed. The small-*thv* effect on BER is similar to when small α is used in the WDI scheduling. Other than *thv*=1, decoding is correctly performed. When *thv* is 3 and *thc* is 6, two controllers in each VAR and CHK can be designed using AND gates.

Figure 10 shows the BER performance based on asynchronous stochastic decoding. The optimum parameters to achieve the minimum BERs at the high SNR region are selected for all the algorithms. For the SWDD scheduling algorithm, one additional parameter (*thv*=3 and *thc*=6) is selected, where the hardware is simply designed as shown in Fig. 5. The WDD and the SWDD scheduling algorithms achieve better BER performance than the WDI scheduling algorithm. The BER of the SWDD is superior to that of

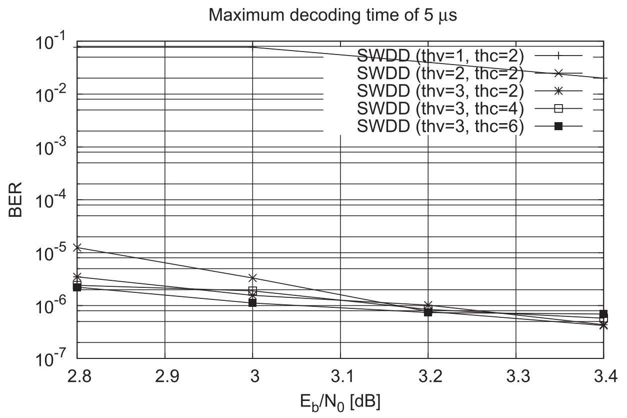

Fig. 9 Simulated BER results of a regular (1024,512) LDPC code based on the SWDD scheduling algorithm.

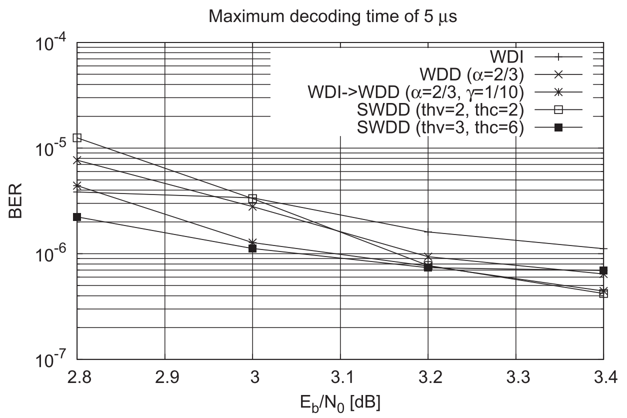

Fig. 10 Simulated BER results of a regular (1024,512) LDPC code based on asynchronous stochastic decoding.

the WDD and is similar to that of the combination of the WDI and the WDD at the high SNR region. In addition, the SWDD requires less complex hardware than that based on the WDD and the combination of the WDI and the WDD.

6.2 Comparisons and Discussions

We compare BERs with those based on the sum-product algorithm (SPA), synchronous stochastic decoding, and clockless stochastic decoding [21], [22] shown in Fig. 11. The BER based on the SPA is evaluated using a floating-point simulation with 32 iterations. In synchronous stochastic decoding, two maximum decoding counts (MaxDC) are set to 5k and 1.6k. Suppose that the clock frequency of the synchronous stochastic decoders is 333 MHz under ASPLA 90nm CMOS technology. MaxDC=5k is the maximum decoding time of 15 μs and MaxDC=1.6k is that of 5 μs. The BER performance at MaxDC=5k is worse than that of the SWDD scheduling algorithm with requiring three times larger maximum decoding time, causing larger input and output buffers to the decoder [15].

Table 2 shows decoding latency in synchronous and asynchronous stochastic decoding. All decoders use the

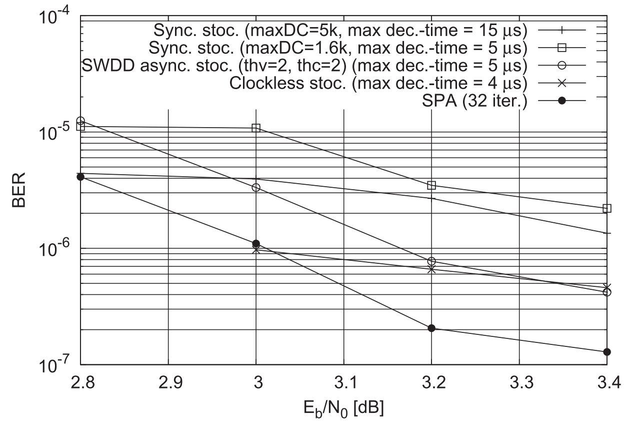

Fig. 11 Simulated BER results of a regular (1024,512) LDPC code.

Table 2 Decoding latency at E_b/N_0 = 3.4 dB using a regular (1024,512) LDPC code.

	Sync.	Async. (WDI)	Async. (SWDD) (*thv*=2, *thc*=2)
Latency [ns]	225	35	23

same maximum decoding time of 5 μs. The latency of synchronous stochastic decoding is large because the clock frequency is restricted by the worst-case latency due to long wires in the interleaver portion. The latencies of asynchronous stochastic decoding have 6.4x-9.8x smaller latencies than that of synchronous stochastic decoding. The WDI scheduling algorithm needs to assign relatively large computation delay time on the VAR as small computation delay time updates too much in the VAR, causing poor BER performance [19]. The WDD and the SWDD can assign smaller computation delay time than the WDI depending on data-transmission delay time. As a result, the SWDD scheduling algorithm achieves a 1.4x smaller latency than that of the WDI scheduling algorithm.

Compared with clockless stochastic decoding at the maximum decoding time of 4 μs, asynchronous stochastic decoding based on the SWDD scheduling algorithm achieves almost the same BER performance at a high SNR region with a 25% increase of the maximum decoding time. In terms of the hardware implementation, asynchrornous stochastic decoding has some benefits compared with clockless stochastic decoding. Currently, we are developing an implementation of clockless stochastic decoders [22], but there are still some issues. The issue concerned is metastability that stores wrong data in registers because there is not any global and local clock signals. That would change the probability wrongly in the EM or the TFM, causing poor BER performance. Other issue is that there is no design flow for clockless circuits that have lots of loops without registers, which would be difficult to implement and test the clockless stochastic decoder. In contrast, asynchronous stochastic decoding can be implemented using asynchronous circuits [25], [26] that operate based on the request-and-acknowledgement handshaking. The handshaking process guarantees the timing of storing data in the EM or the TFM, which maintains good BER performance. Using the SWDD scheduling algorithm, asynchronous control circuits would be small.

7. Conclusions

In this paper, we have presented asynchronous scheduling algorithms for stochastic decoding of LDPC codes. Asynchronous stochastic decoding has the possibility of implementing an energy-efficient high-throughput LDPC decoder due to the lack of a global clock signal that releases from the worst-case delay restriction and reduces the power dissipation of clocking. The previous WDI scheduling algorithm suffers from error floors on BER performance that occur in synchronous stochastic decoding. The WDD scheduling algorithm assigns different update timing at each computation node depending on the data-transfer delay time between the nodes. The variation of the update timing increases the switching activity in stochastic LDPC decoders, which reduces the error floors. As a result, asynchronous stochastic decoding based on the WDD and the simplified WDD (SWDD) scheduling algorithms achieves 6.4-9.8× smaller latency than that of synchronous stochastic decoding with a 0.25-0.3 coding gain. In addition, the SWDD scheduling algorithm eliminates some complex functions that are a time-averaging circuit and a multiplier used in the WDD scheduling algorithm. In future prospects, an energy-efficient high-throughput LDPC decoder can be implemented based on the SWDD scheduling algorithm.

Acknowledgements

This research was supported by JST, CREST and JSPS KAKENHI Grant Number 26700003. This simulation is supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc.

References

[1] R. Gallager, "Low-density parity-check codes," IRE Trans. Inf. Theory, vol.8, no.1, pp.21–28, Jan. 1962.
[2] D. MacKay, "Good error-correcting codes based on very sparse matrices," Proc. IEEE International Symposium on Information Theory, p.113, June/July 1997.
[3] IEEE 802.16 Working Group [Online] Available: www.ieee802.org/16.
[4] IEEE 802.11n Working Group [Online] Available: www.ieee802.org/11.
[5] IEEE P802.3an 10GBASE-T Task Force [Online] Available: www.ieee802.org/3/an.
[6] F. Kschischang, B. Frey, and H.A. Loeliger, "Factor graphs and the sum-product algorithm," IEEE Trans. Inf. Theory, vol.47, no.2, pp.498–519, Feb. 2001.
[7] A. Blanksby and C. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder," IEEE J. Solid-State Circuits, vol.37, no.3, pp.404–412, March 2002.
[8] A. Darabiha, A. Chan Carusone, and F. Kschischang, "Power reduction techniques for LDPC decoders," IEEE J. Solid-State Circuits, vol.43, no.8, pp.1835–1845, Aug. 2008.

[9] Z. Zhang, V. Anantharam, M. Wainwright, and B. Nikolic, "An efficient 10GBASE-T Ethernet LDPC decoder design with low error floors," IEEE J. Solid-State Circuits, vol.45, no.4, pp.843–855, April 2010.

[10] T. Mohsenin, D. Truong, and B. Baas, "A low-complexity message-passing algorithm for reduced routing congestion in LDPC decoders," IEEE Trans. Circuits Syst. I: Regular Papers, vol.57, no.5, pp.1048–1061, May 2010.

[11] V. Gaudet and A. Rapley, "Iterative decoding using stochastic computation," Electron. Lett., vol.39, no.3, pp.299–301, Feb. 2003.

[12] S. Sharifi Tehrani, W. Gross, and S. Mannor, "Stochastic decoding of LDPC codes," IEEE Commun. Lett., vol.10, no.10, pp.716–718, Oct. 2006.

[13] S. Tehrani, S. Mannor, and W. Gross, "Survey of stochastic computation on factor graphs," 37th International Symposium on Multiple-Valued Logic, p.54, May 2007.

[14] S. Sharifi Tehrani, S. Mannor, and W. Gross, "Fully parallel stochastic LDPC decoders," IEEE Trans. Signal Process., vol.56, no.11, pp.5692–5703, Nov. 2008.

[15] S. Sharifi Tehrani, A. Naderi, G.A. Kamendje, S. Hemati, S. Mannor, and W. Gross, "Majority-based tracking forecast memories for stochastic LDPC decoding," IEEE Trans. Signal Process., vol.58, no.9, pp.4883–4896, Sept. 2010.

[16] V.C. Gaudet and W.J. Gross, "Switching activity in stochastic decoders," 40th IEEE International Symposium on Multiple-Valued Logic, pp.167–172, May 2010.

[17] C. Winstead, V. Gaudet, A. Rapley, and C. Schlegel, "Stochastic iterative decoders," International Symposium on Information Theory, 2005. ISIT 2005, pp.1116–1120, Sept. 2005.

[18] G. Sarkis, S. Mannor, and W. Gross, "Stochastic decoding of LDPC codes over GF(q)," IEEE International Conference on Communications, 2009, pp.1–5, June 2009.

[19] N. Onizawa, V. Gaudet, T. Hanyu, and W. Gross, "Asynchronous stochastic decoding of low-density parity-check codes," IEEE 42nd International Symposium on Multiple-Valued Logic (ISMVL), pp.92–97, May 2012.

[20] N. Onizawa, W. Gross, T. Hanyu, and V. Gaudet, "Lowering error floors in stochastic decoding of LDPC codes based on wire-delay dependent asynchronous updating," IEEE 43rd International Symposium on Multiple-Valued Logic (ISMVL), pp.254–259, 2013.

[21] N. Onizawa, W. Gross, T. Hanyu, and V. Gaudet, "Clockless stochastic decoding of low-density parity-check codes," 2012 IEEE Workshop on Signal Processing Systems, pp.143–148, Oct. 2012.

[22] N. Onizawa, W. Gross, T. Hanyu, and V. Gaudet, "Clockless stochastic decoding of low-density parity-check codes: architecture and simulation model," J. Signal Process. Syst., pp.1–10, 2013.

[23] N. Onizawa, T. Ikeda, T. Hanyu, and V. Gaudet, "3.2-Gb/s 1024-b rate-1/2 LDPC decoder chip using a flooding-type update-schedule algorithm," 50th Midwest Symposium on Circuits and Systems, pp.217–220, Aug. 2007.

[24] N. Onizawa, T. Hanyu, and V. Gaudet, "Design of high-throughput fully parallel LDPC decoders based on wire partitioning," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.18, no.3, pp.482–489, March 2010.

[25] N. Onizawa, V. Gaudet, and T. Hanyu, "High-throughput bit-serial LDPC decoder LSI based on multiple-valued asynchronous interleaving," IEICE Trans. Electron., vol.E92-C, no.6, pp.867–874, June 2009.

[26] N. Onizawa, V. Gaudet, and T. Hanyu, "Low-energy asynchronous interleaver for clockless fully parallel LDPC decoding," IEEE Trans. Circuits Syst. I: Regular Papers, vol.58, no.8, pp.1933–1943, Aug. 2011.

[27] K. Christensen, P. Jensen, P. Korger, and J. Sparso, "The design of an asynchronous TinyRISCTM TR4101 microprocessor core," Proc. 1998 Fourth International Symposium on Advanced Research in Asynchronous Circuits and Systems, 1998, pp.108–119, March/April 1998.

Naoya Onizawa received the B.E., M.E. and D.E. degrees in Electrical and Communication Engineering from Tohoku University, Japan, in 2004, 2006 and 2009, respectively. He is currently an Assistant Professor in the Frontier Research Institute for Interdisciplinary Sciences at Tohoku University, Japan. He was a postdoctoral fellow at Tohoku University from 2009 to 2011 and at University of Waterloo, Canada in 2011 and at McGill University, Canada from 2011 to 2013. His main interests and activities are in the energy-efficient VLSI design based on asynchronous circuits and multiple-valued circuits, and their applications, such as LDPC decoders, associative memories, and Network-on-Chips. He received the Best Paper Award in IEEE Computer Society Annual Symposium on VLSI in 2010. Dr. Onizawa is a Member of the IEEE.

Warren J. Gross received the B.A.Sc. degree in electrical engineering from the University of Waterloo, Waterloo, Ontario, Canada, in 1996, and the M.A.Sc. and Ph.D. degrees from the University of Toronto, Toronto, Ontario, Canada, in 1999 and 2003, respectively. Currently, he is an Associate Professor with the Department of Electrical and Computer Engineering, McGill University, Montreal, Quebec, Canada. His research interests are in the design and implementation of signal processing systems and custom computer architectures. Dr. Gross is currently Vice-Chair of the IEEE Signal Processing Society Technical Committee on Design and Implementation of Signal Processing Systems and serves as Associate Editor for the IEEE Transactions on Signal Processing. He has served as Technical Program Co-Chair of the IEEE Workshop on Signal Processing Systems (SiPS 2012) and as Chair of the IEEE ICC 2012 Workshop on Emerging Data Storage Technologies. Dr. Gross is a Senior Member of the IEEE and a licensed Professional Engineer in the Province of Ontario.

Takahiro Hanyu received the B.E., M.E. and D.E. degrees in Electronic Engineering from Tohoku University, Sendai, Japan, in 1984, 1986 and 1989, respectively. He is currently a Professor in the Research Institute of Electrical Communication, Tohoku University. His general research interests include nonvolatile logic circuits and their applications to ultra-low-power and/or PVT-variation-free VLSI processors, and multiple-valued current-mode circuit and its application to power-aware asynchronous Network-on-Chip systems. He received the Sakai Memorial Award from the Information Processing Society of Japan in 2000, the Judge's Special Award at the 9th LSI Design of the Year from the Semiconductor Industry News of Japan in 2002, the Special Feature Award at the University LSI Design Contest from ASP-DAC in 2007, the APEX Paper Award of Japan Society of Applied Physics in 2009, the Excellent Paper Award of IEICE, Japan, in 2010, Ichikawa Academic Award in 2010, the Best Paper Award of IEEE ISVLSI 2010, and the Paper Award of SSDM 2012. Dr. Hanyu is a Senior Member of the IEEE.

Vincent C. Gaudet received the B.Sc. in Computer Engineering from the University of Manitoba in 1995, the M.A.Sc. from the University of Toronto in 1997, and the Ph.D. from the University of Toronto in 2003. Since 2010 he has been an Associate Professor in the Department of Electrical and Computer Engineering at the University of Waterloo. From 2002 to 2010, he was with the University of Alberta, first as Assistant Professor, and later as Associate Professor. In 2002, he was a Research Associate at the Ecole Nationale Supérieure des Télécommunications de Bretagne (now Télécom-Bretagne) in Brest, France. In 2008-2009 he was on sabbatical leave at Northeastern University in Boston, MA, and at Tohoku University in Sendai, Japan. His research interests focus on information-processing microsystems, more specifically on energy-efficient circuits for graph-based decoding of error-control codes such as Turbo and Low-Density Parity-Check (LDPC) codes. He also has interests in biomedical circuits and systems. He has taught several courses in analog and digital microelectronic circuit design, at the undergraduate and graduate levels. He currently serves as the Undergraduate Advisor for the Electrical Engineering program at the University of Waterloo, and will begin a term as Associate Chair for Undergraduate Studies in September 2013. Prof. Gaudet is a Senior Member of the IEEE and is licensed as a Professional Engineer in Ontario. He serves as Technical Editor for the IEEE International Solid-State Circuits Conference and as Vice-Chair of the IEEE Computer Society Technical Committee on Multiple-Valued Logic. In 2012, he served as the Program Chair for the 2012 IEEE International Symposium on Multiple-Valued Logic in Victoria, BC. He was the recipient of the 2009 Petro Canada Young Innovator Award. He has previously served on the NSERC Scholarships and Fellowships Committee 176 (2005-2007), and as Associate Editor for the IEEE Transactions on Circuits and Systems Parts I (2010-2012) and II (2008-2009).