Survey of Stochastic Computation on Factor Graphs

Saeed Sharifi Tehrani, Shie Mannor and Warren J. Gross
Department of Electrical and Computer Engineering
McGill University
Montreal, Quebec, Canada H3A 2A7
Email: {sshari9, shie, wjgross}@ece.mcgill.ca

Abstract

Stochastic computation is a new alternative approach for iterative computation on factor graphs. In this approach, the information is represented by the statistics of the bit stream which results in simple high-speed hardware implementation of graph-based algorithms. Despite the first purpose of its invention (i.e., low-precision digital circuits), the stochastic representation has recently been shown to be able to provide near-optimal decoding performance for practical Low-Density Parity-Check (LDPC) codes, with respect to Sum-Product Algorithm (SPA). This paper provides a survey of stochastic methods for graph-based iterative decoding, the state-of-the-art and, their possible new applications.

1. Introduction

Stochastic arithmetic was introduced in the 1960's as a method to design low-precision digital circuits [6]. The important motivation for considering stochastic computation was the possibility of performing complex computations using only simple circuitry [4, 6]. In stochastic computation, probabilities are represented as streams of random digital bits using Bernoulli sequences in which the information is contained in the statistics of the bit stream. Using this representation, complex operations on probabilities such as multiplication and division are converted to operations on bits which can easily be manipulated using simple stochastic gates. This allows high clock rates for the stochastic computational elements while requiring low computation hardware area. In addition, due to the bit-serial nature of stochastic computation, communication between computational elements requires only one wire per signal. Also, compared to other computing techniques, stochastic computation is fault-tolerant and it can trade off computation accuracy and time without any change in hardware [4].

Stochastic computation has been considered for different applications such as implementation of artificial neural networks [4] and a real-time motor controller in [1]. Error control coding is a new recent application for stochastic computation [2, 5, 8, 9, 16, 17]. The simplicity of stochastic computation has made it attractive for the implementation of so-called "capacity approaching" decoders such as Turbo [3] and LDPC decoders [7] in which hardware complexity, interconnect wiring and routing congestion are major implementation problems. However, the early stochastic decoding methods could only work either on very simple short codes or for decoding some specific error-correcting codes on trellis graphs and hence, they were not successful for decoding state-of-the-art LDPC codes on factor graphs. Recently, a new stochastic decoding method is proposed in [16] which is able to provide a near-optimal performance for decoding practical LDPC codes with respect to floating-point SPA decoding. This method has indicated the viability of the stochastic approach for iterative decoding of the state-of-the-art LDPC codes and its potential for low-complexity high-throughput hardware decoding of error-correcting codes on factor graphs.

The rest of the paper is organized as follows. Sections 2 and 3 give an overview of stochastic representation and computation. Section 4 provides an overview of LDPC decoding. Section 5 describes the recent application of stochastic computation on factor graphs. Section 6 discusses the major problem of *latching* in stochastic decoding. Section 7 presents the new stochastic decoding method proposed in [16]. Finally, Section 8 discusses the possible perspective of stochastic computation.

2. Stochastic Representation

In stochastic computation, probabilities are encoded by a Bernoulli sequence as a random sequence of $\{a_i\}$

Proceedings of the 37th International Symposium on Multiple-Valued Logic (ISMVL'07)
0-7695-2831-7/07 $20.00 © 2007 IEEE

digital bits. Each bit in the sequence is equal to logic '1' with the probability to be encoded. As an example, a sequence of 10 bits with 6 bits equal to logic '1', represents the probability of 0.6. The encoding scheme is not unique thus different encoded stochastic sequences are possible for the same probability.

A common fallacy in interpreting the usefulness of the stochastic computation approach is that the precision provided by stochastic representation is very poor and this approach becomes very inefficient when high-precision is needed. For example, it might be claimed that to represent the probability of 0.001 only 10 bits is needed in a fixed-point representation but the equivalent stochastic representation needs at least 1000 bits. It is important to note that stochastic sequences are not necessarily "frames" of bits and they can be interpreted and used as "stochastic processes" where no framing is required. This is one of the main differences that distinguishes stochastic approach among other computing techniques in many applications. Stream representation is robust in the presence of noise and single bit faults [4] and more importantly, it facilities the relaxation techniques in applications and algorithms such as iterative decoding where self-correcting and/or decision elements are used. In other words, in such applications, computations rely on the flow of changes in the statistics of the stochastic streams rather than on the precision of discrete frames of bits. Therefore, the level of accuracy or precision might not be as important as the statistics of stream. For example, in such applications it might be important to know that if a probability is converging to 1, however, we might not care if the exact value of the probability at a given iteration is 0.900 or 0.901.

3. Stochastic Computation

The comparator shown in Figure 1 can be used to convert probabilities to stochastic streams [1, 6]. In this figure, X and R are W-bit wide inputs. R is a random number with uniform distribution which varies in each clock cycle. The output bit of the comparator is '1' when $X > R$. Therefore, the probability of having an output equal to '1' is $X/2^W$. The following subsections describe main stochastic operations.

3.1. Multiplication

Let $P_a = \Pr(a_i = 1)$ and $P_b = \Pr(b_i = 1)$ be the probabilities to be multiplied. The outcome, P_c, can be computed by an AND gate shown in Figure 2. Similarly, other boolean operations (*e.g.*, NOT, XOR *etc.*) can be used to implement different operations on probabilities.

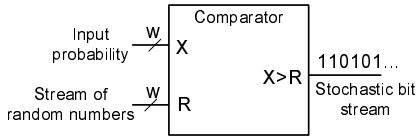

Figure 1. Probability to stochastic stream conversion.

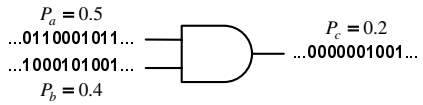

Figure 2. Stochastic multiplication.

3.2. Division

Consider the JK flip-flop shown in Figure 3 with input sequences of $\{a_i\}$ and $\{b_i\}$ representing the probabilities of P_a and P_b, respectively. The output bit c_i is equal to '1' with the probability of P_c and is equal to '0' with the probability of $1 - P_c$. A random output transition from '1' to '0' occurs with probability of $(1-P_c)P_a$ and the reverse transition occurs with the probability of P_cP_b. Since the expected occurrence of random transition in both directions must be equal, we have

$$P_cP_b = (1 - P_c)P_a \rightarrow P_c = P_a/(P_a + P_b). \quad (1)$$

The operation of (1) is an approximation to P_a/P_b, if $P_a << P_b$. There exist other stochastic division methods with more precision [6]. However, as will be discussed in Sections III and IV, (1) matches the variable node operation in SPA.

3.3. Addition and Subtraction

Stochastic addition and subtraction are not as straightforward as multiplication and division. This is because they are not closed operations on the probability interval of $[0, 1]$. Therefore, these operations should be combined with a scaling operation to ensure the range of $[0, 1]$ for the outcome [4]. Addition with scaling is performed as $P_c = \sum S_l P(A_l)$, where $P(A_l) = P_{a_l} = \Pr(a_{l_i} = 1)$ and S_l is the probability of selecting given input A_l such that $\sum S_l = 1$. The outcome is the scaled sum of the input probabilities. This operation can be implemented in hardware using a multiplexer as shown in Figure 4, where RS refers to the random selection supplied by (pseudo) random number generators. Generating RS is straightforward when the N is a power of two. For the case where N is not a power of two, it is possible to increase N by padding '0' signals to input with the cost of sub-optimality of calculation [4].

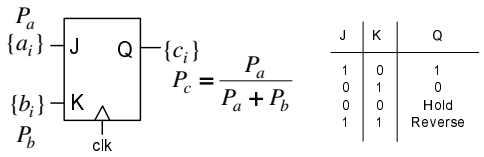

Figure 3. Stochastic division.

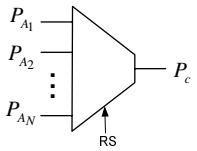

Figure 4. Stochastic (scaled) addition [4].

4. LDPC Decoding and Factor Graphs

LDPC codes [7] are a class of linear block codes which the set of all codewords, $x \in C$, spans the null space of a sparse parity-check matrix H *i.e.*, $Hx = 0$. LDPC codes are decoded by means of SPA. These codes are shown to be capable of error-correcting performance close to the Shannon limit [15]. Graphical representations such as Bayesian networks and later-developed representations like *factor graphs* [12] are unifying frameworks for LDPC decoding algorithms. Figure 5 shows a factor graph for a $(n = 12, k = 3)$ LDPC code. The nodes of a factor graph are separated into two distinctive sets: variable and check nodes. Each edge in the graph connects a node from the two sets.

4.1. Sum-Product Algorithm

SPA is an iterative algorithm for LDPC decoding. SPA is a form of Pearl's belief propagation [14] that uses *message passing* over the edges of factor graphs. For good decoding performance, it is important that the length of *cycles* in the factor graph are as long as possible. Short cycles such as the length-4 cycle (namely 4-cycle) highlighted in Figure 5, correlate messages and degrade the performance of SPA. SPA computes the Log-Likelihood Ratio (LLR) of the received sequence and makes a decision by comparing this LLR to the threshold value. In a Binary Phase Shift Keying (BPSK) transmission (± 1) over an Additive White Gaussian Noise (AWGN) channel, the channel LLR value of the i-th sample in the received block $(i = 1, ..., n)$ is defined as $L_i = \log(\frac{\Pr(x_i=+1|y_i)}{\Pr(x_i=-1|y_i)}) = \frac{4y_i}{N_0}$, where N_0 is the AWGN power spectral density and, $x_i \in \{-1, +1\}$ and $y_i \in R$ denote the i-th sample of the transmitted and received block, respectively. Let $P_{i\to j} \in [0, 1]$ be the probability message from a variable node v_i with $d_v = 3$

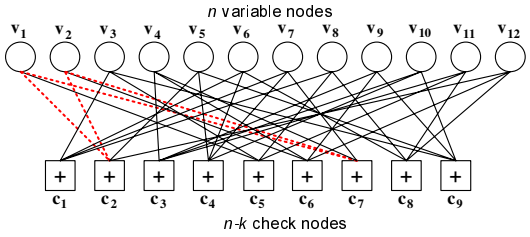

Figure 5. Factor graph of a (12,3) LDPC Code. Dashed lines show a 4-cycle.

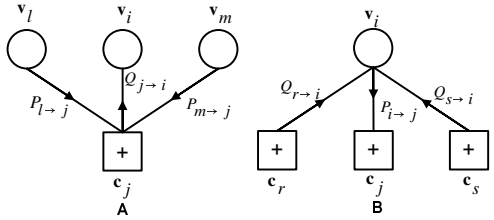

Figure 6. A) extrinsic and B) intrinsic probability messages in SPA.

to a check node c_j $(j = 1, ..., n-k)$ with $d_c = 3$. Let $Q_{j\to i} \in [0, 1]$ be the probability message from c_j to v_i. Also, let $\{\mathrm{v}_i, \mathrm{v}_l, \mathrm{v}_m\}$ be the set of variable nodes connected to c_j and, $\{\mathrm{c}_j, \mathrm{c}_r, \mathrm{c}_s\}$ be the set of check nodes connected to v_i[1]. SPA steps can be described as follows.

1) For v_i, convert L_i to the initialization probability as $P^i_{\text{init}} = e^{L_i}/(e^{L_i} + 1)$. v_i sends $P_{i\to j} = P^i_{\text{init}}$ to c_j.

2) c_j sends $Q_{j\to i}$ to v_i as (see Figure 6(A))

$$Q_{j\to i} = P_{l\to j}(1 - P_{m\to j}) + P_{m\to j}(1 - P_{l\to j}). \quad (2)$$

3) v_i sends $P_{i\to j}$ to c_j as (see Figure 6(B))

$$P_{i\to j} = \frac{Q_{r\to i}Q_{s\to i}}{Q_{r\to i}Q_{s\to i} + (1 - Q_{r\to i})(1 - Q_{s\to i})}. \quad (3)$$

4) Stop decoding once the estimated codeword, $\hat{x}$, satisfies $H\hat{x} = 0$ or a fixed number of iterations has been completed. Otherwise, return to step 2.

5. Stochastic Decoding on Factor Graphs

The stochastic representation of probabilities in the code factor graph results in low-complexity bit-serial parity-check and variable nodes. Let $P_a = \Pr(a_i = 1)$

[1]Note that higher degree nodes can be converted to subgraphs containing only degree three nodes [12].

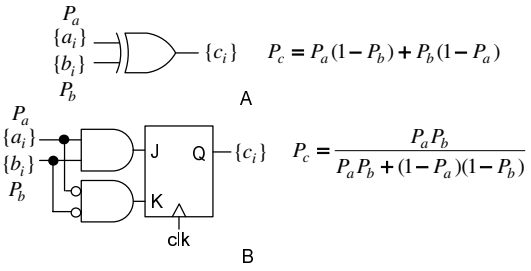

Figure 7. A) Parity-check and B) variable nodes in stochastic decoding.

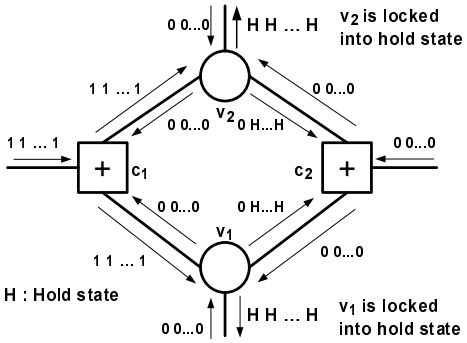

Figure 8. Latching within a 4-cycle.

and $P_b = \Pr(b_i = 1)$ be the probability of two input bits, a_i and b_i, in a $d_c = 3$ check node. The output probability P_c is computed as

$$P_c = P_a(1 - P_b) + P_b(1 - P_a). \quad (4)$$

Similarly, the equality function in a $d_v = 3$ variable node for inputs P_a and P_b is

$$P_c = P_a P_b / (P_a P_b + (1 - P_a)(1 - P_b)). \quad (5)$$

Figure 7 shows the equivalent hardware structures for (4) and (5) [8]. Note that the variable node in Figure 7(B) is in *hold state* (*i.e.*, $c_i = c_{i-1}$), if the two input bits are not equal ($a_i \neq b_i$). In addition to simple variable and check node structures, stochastic computation also reduces the routing congestion problem. Because only one bit (per direction) is required to represent an edge between a parity-check and a variable node. This implies that in a decoding round, the stochastic decoding proceeds by the variable and check nodes exchanging a bit (per direction) along each edge in the factor graph. We refer to these decoding rounds as *Decoding Cycles* (DCs) to highlight that they do not directly correspond to the iterations in SPA.

The variable and check node structures shown in Figure 7 are used in [2] and [8] for decoding a (7,4) Hamming and a (16,8) LDPC code, respectively. In [8], the decoder had about 0.15 dB decoding performance loss at a Bit Error Rate (BER) of 10^{-4} with respect to SPA decoding. Unfortunately, due to the high decoding performance loss, this method cannot directly be used to decode longer (practical) codes. Several research works have been conducted to improve the stochastic decoding methods to be able to decode practical codes. These efforts are discussed in the next section.

6. Latching and Rerandomization

A major difficulty observed in stochastic decoding is the sensitivity to the level of random switching activity (bit transition) for proper decoding operation [16, 17]. This problem is specially crucial when the code factor graph has cycles. The *latching* problem refers to the case where cycles in the graph correlate messages such that a group of nodes are locked into a fixed state which is solely maintained by the correlated messages [5, 16]. The latching problem is particulary acute in practical LDPC decoders [16, 17]. This problem can be worse at high Signal-to-Noise-Ratios (SNRs) because the received LLRs are large so that the corresponding probabilities approach 0 (or 1). In this situation, bits in stochastic sequences are mostly '0' (or '1'), making random switching events very rare for proper decoding operation [16, 17]. Figure 8 illustrates the latching of two variable nodes, v_1 and v_2, into a fixed state (hold state in this example) for several DCs, which is forced by a 4-cycle in the absence of enough switching activity. The latching problem causes huge performance loss for decoding practical LDPC code. To circumvent this problem, different methods are suggested in the literature which are discussed in the following sections.

6.1. Supernodes

The idea of *supernodes* is proposed in [5]. Supernodes are special structures which reduce the latching problem by regenerating uncorrelated messages based on the probabilities of incoming stochastic messages. A supernode tabulates incoming messages in histograms to estimate their probabilities and regenerates new uncorrelated stochastic messages. There exists different structures for supernodes with different complexity. Also, supernodes are used in different positions in the decoder. In [5], they are used as a special variable node which can be placed in critical part of the graph (*e.g.*, where short cycles exist), however, in [9] they are placed between variable and check nodes of the decoder. Figure 9(A) shows the structure of supernodes used in [5] for trellis decoding of a (256,121) product Turbo code.

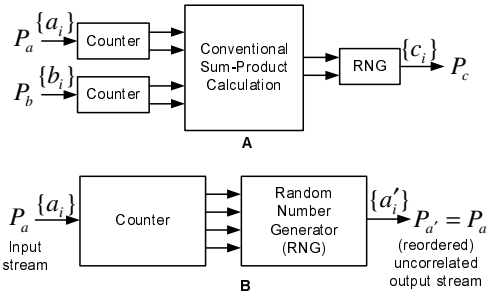

Figure 9. Structure of supernode A) used in [5] and B) used in [9] .

Supernodes in this trellis decoder [5] are used instead of variable nodes shown in Figure 7. These supernodes were *packetized* in a sense that they are using the conventional SPA calculation (*i.e.*, (3)) after a time-step to calculate the probabilities of the new outgoing messages and regenerate new stochastic messages. A reduced-complexity version of these supernodes is suggested in [17] which only uses Random Number Generators (RNGs), fixed-point addition operation and counters. Figure 9(B) shows another structure of supernodes suggested in [5]. In this structure the input messages are fed directly to a counter to tally the number of ones for a given number of samples. This count is then used to generate new probabilities. This structure is used in [9] for hardware implementation of a (16,8) LDPC decoder where supernodes were placed between variable and check nodes of the decoder to reorder and break the correlation between stochastic messages.

6.2. Scaling Channel LLRs

As mentioned before, the latching problem can be worse at high SNRs due to the lack of switching activity. The idea of scaling channel LLRs is based on scaling received LLRs to increase switching activity in the decoder. This idea is used in [17] for stochastic decoding of a (16,11) Hamming code. In this method, every block of the channel LLRs are scaled to a maximum value to ensure the same level of switching activity for each block. *Noise-Dependant Scaling* (NDS) is another scaling method used in [16]. In this method, the channel LLRs are scaled by a scaling factor which is proportional to the operating SNR. Because the scaling factor is proportional to the noise level, it ensures a similar level of switching activity for different ranges of SNRs. The scaled LLR, L'_i, for the i-th symbol, y_i, is calculated as $L'_i = (\alpha N_0/Y)L_i = 4\alpha y_i/Y$, where Y is a fixed maximum value of the received symbols and, α is a constant factor ($0 < \alpha < Y$) in which its value can be chosen based on the BER performance [16].

6.3. Rerandomization by Edge Memories

Edge Memories (EMs) are proposed in [16]. EMs are M-bit shift registers introduced at each edge of the code factor graph. Each EM is updated in accordance with (5) only when the variable node is not in hold state for that edge. If a hold state happens, a bit is randomly chosen[2] from the EM of the edge and passed as the outgoing bit. Using this updating scheme the random switching activity of stochastic messages in the code graph is increased because the chance of locking into a fixed state is reduced. This is because in a hold state, a bit is randomly chosen from those previous outputs of the variable node which are not produced in a hold state. EMs rely on the basic idea of stochastic computation on streams of bits and hence they do not use the conventional SPA calculation and packetized bit sequences.

7. The New Stochastic Decoding Method

The new stochastic LDPC decoding method proposed in [16] uses both NDS and EMs. The NDS method is usually sufficient for decoding short codes [16], however, both NDS and EMs are essential for decoding relatively longer LDPC codes. In this method, after applying NDS to channel LLRs, one bit of corresponding stochastic probability streams is passed to each variable node during each DC. Variable nodes and EMs operate as described in Section 6.3. The check nodes perform the parity-check equation and pass their messages to the variable nodes. Each variable node computes its output at the end of each DC and passes it to an up/down counter which is decremented in case of a '0' output and incremented in case of a '1'. The sign bit of the up/down counter indicates the hard decision in each DC. The decoder operations for a fixed number of DCs unless the stopping criterion, $H\hat{x} = 0$, is satisfied sooner.

Figure 10 shows the BER performance of this method for a regular (1024,512) LDPC code[3]. This code has $d_v = 3$ variable and $d_c = 6$ check nodes. An $M = 50$ and $\alpha = 3$ are used for EMs and NDS[4], respectively, with maximum 60K DCs. As shown, compared to SPA with floating-point implementation, this method provides a near-optimal performance for the (1024,512) code. The SNR loss at the BER of 10^{-5} is less than 0.05

[2]This can be done by generating random addresses for EMs.
[3]In this figure, SNR=E_b/N_0 (dB) and E_b is the average bit energy.
[4]An $\alpha = 3$ provided the best BER performance.

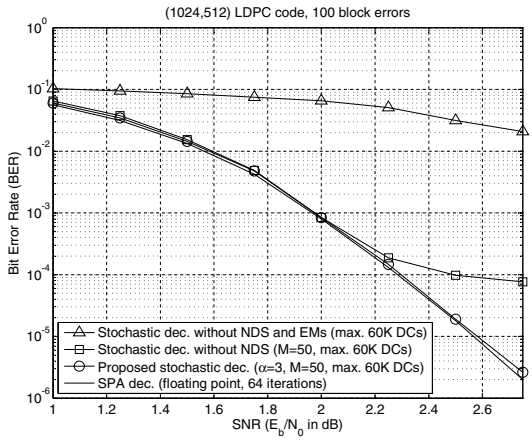

Figure 10. Simulation results [16].

dB. To demonstrate the individual performance contribution of NDS and EMs, Figure 10 also shows results for decoding with EMs but without NDS as well as results for decoding without NDS and EMs. The observed average DCs per block are much less than the maximum DCs. For example, the average DCs at the BER of 10^{-5} is about 6K DCs. It should be noted that DCs are not equivalent to the iterations in SPA and because of the simple structure of a stochastic decoder, it is more feasible to have a fully-parallel stochastic decoder with a clock frequency much higher than that of a SPA decoder.

8. New Applications

As suggested in [17], reliable computation in nano-scale systems might be a new application for stochastic computation and decoders. Nano devices provide signal energy that can be even below the thermal noise energy at room temperature. Several methods for reliable computation using unreliable nanoelectronic components have been proposed [10, 11] in which many of them are similar to stochastic operations. Stochastic gates might have a simpler implementation using inherently noisy devices. As an example, a simple structure for stochastic variable nodes is suggested in [17] using nano-scale MOS devices in which variable operation is accomplished by circuit random (noisy) behavior.

Beside the implementation and performance advantages of stochastic computation, theoretical aspects of stochastic representation of probabilities and their impact on the dynamics of computation are also interesting. An interesting application might be quantum computing [13]. Stochastic computation is performed on streams of stochastic bits. Equivalent representation of stochastic bits using *qubits* and expressing the decoding operation using quantum logic gates might open doors for several research topics on the applications of quantum computation for iterative decoding. Another interesting research possibility is to consider stochastic approach for other message passing algorithms (instead of SPA) for using inference on graphical models such as Bayesian networks in real-time and control applications.

References

[1] A. Dinu et al. Stochastic implementation of motor controllers. In *Proc. of the IEEE Int. Symp. on Industrial Electronics*, pages 639–644, July 2002.

[2] A. Rapley et al. Stochastic iterative decoding on factor graphs. In *Proc. 3rd Int. Symp. on Turbo Codes and Related Topics*, pages 507–510, Brest, France, Sept. 2003.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding: Turbo codes. In *Proc. IEEE Int. Conf. on Communications*, pages 1064–1070, May 1993.

[4] B. Brown and H. Card. Stochastic neural computation I: Computational elements. *IEEE Trans. on Computers*, 50(9):891–905, Sept. 2001.

[5] C. Winstead et al. Stochastic iterative decoders. In *Proc. IEEE Int. Symp. on Information Theory*, pages 1116–1120, Sept. 2005.

[6] B. Gaines. *Advances in Information Systems Science*, chapter 2, pages 37–172. Plenum, New York, 1969.

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

[8] V. Gaudet and A. Rapley. Iterative decoding using stochastic computation. *Electron. Lett.*, 39(3):299–301, Feb. 2003.

[9] W. J. Gross, V. Gaudet, and A. Milner. Stochastic implementation of LDPC decoders. In *Proc. 39th Asilomar Conf. on Signals, Systems, and Computers*, Nov. 2005.

[10] J. Han and P. Jonker. A system architecture solution for unreliable nanoelectronic devices. *IEEE Trans. on Nanotechnology*, 1(4):201–208, Dec. 2002.

[11] K. Nepal et al. Designing logic circuits for probabilistic computation in the presence of noise. In *the Design Automation Conference*, June 2005.

[12] F. Kschischang, B. Frey, and H. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory*, 47(2):498–519, Feb 2001.

[13] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge U. Press, 2000.

[14] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Kaufman, 1988.

[15] T. J. Richardson and R. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Trans. Inform. Theory*, 47:599–618, Feb. 2001.

[16] S. Sharifi Tehrani, W. J. Gross, and S. Mannor. Stochastic decoding of LDPC codes. *IEEE Communications Letters*, 10(10):716–718, Oct. 2006.

[17] C. Winstead. Error-control decoders and probabilistic computation. In *Tohoku Univ. 3rd SOIM-COE Conf.*, Sendai, Japan, Oct. 2005.