Hindawi Publishing Corporation
VLSI Design
Volume 2008, Article ID 109490, 14 pages
doi:10.1155/2008/109490

Research Article

An Energy-Efficient Multiwire Error Control Scheme for Reliable On-Chip Interconnects Using Hamming Product Codes

Bo Fu and Paul Ampadu

Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY 14627, USA

Correspondence should be addressed to Paul Ampadu, ampadu@ece.rochester.edu

Received 1 June 2008; Accepted 14 October 2008

Recommended by Adam Postula

We propose an energy-efficient error control scheme for on-chip interconnects capable of correcting a combination of multiple random and burst errors. The iterative decoding method, interleaver, using two-dimensional Hamming product codes and a simplified type-II hybrid ARQ, achieves several orders of magnitude improvement in residual flit-error rate for multiwire errors and up to 45% improvement in throughput in high noise environments. For a given system reliability requirement, the proposed error control scheme yields up to 50% energy improvement over other error correction schemes. The low overhead of our approach makes it suitable for implementation in on-chip interconnect switches.

1. INTRODUCTION

On-chip interconnect errors, exacerbated by very deep submicron (VDSM) technology [1], can be caused by supply voltage fluctuation, crosstalk, process variation, radiation, and electromagnetic interference (EMI) [2–6], among others. Reliability can be improved by introducing error control schemes [7–16] such as automatic repeat request (ARQ) and forward error correction (FEC) on the interconnect links. Multiwire errors, including multiple random errors and spatial burst errors (a burst error is defined as multiple adjacent wires being erroneous), become more common in VDSM technologies [4, 7, 15]; thus, previous work using conventional single-error correcting (SEC) codes (e.g., Hamming) [7, 9, 10], which cannot detect or correct multiwire errors, may be incapable of achieving on-chip communication reliability requirements.

Unfortunately, interconnect links have tight speed, area, and energy constraints, making the use of complex but powerful codes unsuitable. Product codes [17, 18], which can be easily implemented by concatenating simple component codes (e.g., Hamming codes), can be protected against both random and burst errors with a relatively small complexity. In this paper, we propose using two-dimensional Hamming product codes to address multiwire errors for on-chip interconnects. Also, an iterative decoding method is combined with a type-II hybrid ARQ scheme [19] to maintain energy efficiency. The remainder of the paper is organized as follows. Section 2 introduces related work on error control for on-chip interconnects. Section 3 describes the proposed error correction scheme. Implementation of the proposed scheme is shown in Section 4. Performance evaluation is discussed in Section 5. Further discussions and conclusions are presented in Section 6 and Section 7, respectively.

2. ERROR CONTROL SCHEMES FOR ON-CHIP INTERCONNECT

There are three classes of interconnect errors—permanent, intermittent, and transient [2]. In this work, we focus on multiple transient errors, which can be multiple random single errors or burst adjacent errors. On-chip communication typically uses one of three schemes for error recovery. (a) Automatic repeat request (ARQ) [7, 8, 16], wherein if the receiver detects errors, it requests that the transmitter resends the information. (b) In forward error correction (FEC) [9–12, 15], the receiver corrects errors without any retransmission requests. (c) In hybrid schemes (HARQ) [13, 14], the receiver corrects errors it can handle and requests retransmission when the errors exceed their error-correction capability. The various error control schemes have different strengths. In ARQ, the error detection codes

are easy to construct at a minor energy cost; however, retransmission reduces throughput (especially in a persistent noise environment) making it unsuitable for high performance applications. FEC can guarantee a certain throughput, but powerful error correction codes are more complex and consume large energy, a critical constraint for on-chip interconnects. Further, when errors exceed the code's error-correction capability, FEC cannot correct the errors and decoder failure occurs. HARQ combines FEC and ARQ to balance reliability, throughput, and energy consumption.

Previous work in this area has often focused on one- or two-bit error scenarios. In [7], cyclic redundancy check (CRC) codes are used to detect errors, and retransmission is used once errors are detected. Instead of CRC, Hamming codes are used to detect two single errors [7, 8] or to correct a single error [7, 9, 10]. In [10, 11], a duplicate-add-parity (DAP) code, in which the information is duplicated and an extra parity check bit added, is used to correct a single error. In [12], symbol error correcting codes are used to correct two-bit burst errors. In [13, 14], single-error correcting double-error detecting (SEC-DED) codes (e.g., extended Hamming) are used to perform HARQ. In order to improve error resilience against burst errors, interleaving can be used to split a wide bus into smaller groups and encode the groups separately [15, 16]. The outputs of these small groups can be further interleaved to reduce the probability of multiple errors occurring within the same group. In the case of multiple random and burst errors, previous approaches lose their effectiveness.

Product codes [17, 18], which can be easily realized by concatenating simple component codes, have a good protection capability against both random and burst errors. In this paper, we propose using product codes to address multiwire interconnect errors. Figure 1 shows the concept of a two-dimensional product code. Given two binary linear block codes $C_1(n_1, k_1)$ and $C_2(n_2, k_2)$, where the n_i's, k_i's represent widths of codeword and information, respectively, the product code C_{pc} can be expressed as

$$\begin{aligned} C_{\text{pc}}(N, K) &= C_1(n_1, k_1) \times C_2(n_2, k_2) \\ &= C_{\text{pc}}(n_2 \times n_1, k_2 \times k_1), \end{aligned} \tag{1}$$

where $N = n_2 \times n_1$ and $K = k_2 \times k_1$ are the product codeword width and message width, respectively. The codes can be obtained by arranging a K-bit input information in a $(k_2 \times k_1)$ matrix, encoding the k_2 rows using a row encoder (which appends $(n_1 - k_1)$ parity check bits to each row), and then encoding the n_1 columns by a column encoder (which appends $(n_2 - k_2)$ parity check bits to each column). Product codes can be constructed by a serial concatenation of simple component codes and a row-column block interleaver, in which the input sequence is written into the matrix row-wise and read out column-wise. Product codes have a larger Hamming distance compared to that of component codes. If the component codes C_1 and C_2 have minimum Hamming distances d_1 and d_2, respectively, then the minimum Hamming distance of the product code C_{pc} is the product $d_1 \times d_2$ [20], which greatly increases the error correction capability.

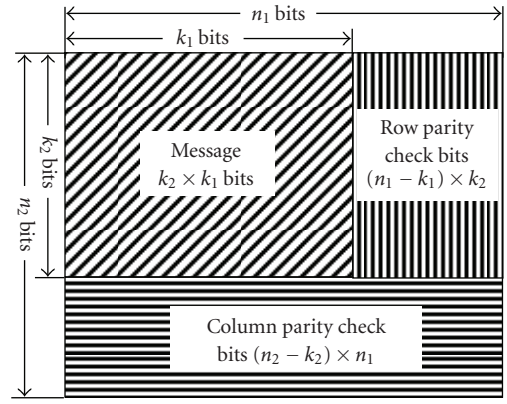


Figure 1: Concept of two-dimensional product codes.

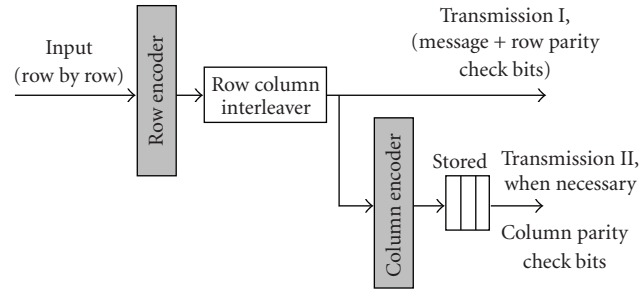


Figure 2: Proposed encoding process.

The simplest two-dimensional product codes are single-parity check (SPC) product codes, guaranteed to correct only one error by inverting the intersection bit in the erroneous row and column [20]. Multidimensional SPC product codes can be constructed to improve the error correction capability, but a more complex decoding process is required [21].

3. PROPOSED ERROR CONTROL SCHEME

In our approach, we use two-dimensional product codes, in which an SEC-DED (e.g., extended Hamming) code ($d_{\min} = 4$) is used for row encoding and a SEC (e.g., conventional Hamming) code ($d_{\min} = 3$) is used for column encoding. The proposed product code then has a total Hamming distance of $d_{\min} = 12$, able to correct $\lfloor (d_{\min} - 1)/2 \rfloor = 5$ random errors. Direct implementation of product codes results in low code rates (because of the large number of redundancy bits) and increased link energy consumption. In order to improve code rate and achieve energy efficiency, we use a modified type-II HARQ, in which redundancy bits are incrementally transmitted when necessary, combined with an iterative decoding method to process Hamming product codes. To the best of our knowledge, our group is the first to combine two-dimensional Hamming product codes using type-II HARQ with iterative decoding for on-chip interconnects. Specifically, a message is encoded and transmitted with its *row* parity check bits. The receiver uses the *row* parity check bits to correct any single random error and burst errors that are distributed in different rows. If the

receiver detects uncorrectable errors (e.g., two errors in a row), it instructs the transmitter to send the *column* check bits which are formed based on the original message. The proposed encoding process is shown in Figure 2. When the *column* parity check bits are received, they are used with the original message and the saved *row* parity check bits to perform iterative decoding of the codes. The effective code rate of the proposed error correction scheme can be expressed in

$$R_{\text{effective}} = \frac{\text{Information width}}{\text{Information width} + \text{Row parity bits} + A}, \quad (2)$$

where A denotes $P_{\text{retransmission}} \times$ Column parity bits and $P_{\text{retransmission}}$ is the probability of retransmission. Because retransmission occurs infrequently for on-chip interconnects, the effective code rate is increased using our proposed error correction scheme.

The simplest strategy of product code decoding is the two-step row-column (or column-row) decoding algorithm [20]. In this algorithm, the received matrix is first decoded row-by-row using a row decoder; the resulting row-decoded matrix is then decoded column-by-column using a column decoder. Unfortunately, this decoding method fails to correct a rectangular four-error pattern, such as $d_{1,1}$, $d_{1,3}$, $d_{3,1}$, and $d_{3,3}$ shown in Figure 3. More powerful soft-input soft-output (SISO) decoding processes can be implemented to decode product codes using the Chase algorithm [18], at the cost of increased latency and power. Moreover, generating soft-input information for on-chip interconnects introduces extra complexity overhead.

Compared to two-step row-column decoding, our method properly addresses the rectangular four-error pattern problem by recording behavior of the row and column decoders using row and column status vectors. Instead of only passing coded data between the row and column decoders, the row and column status vectors are passed between stages and used to help make decoding decisions. The realization of the row and column status vectors can be described by two rules, shown in Figure 3. As shown, two status bits are used to record the decoding behavior for each *row*, and one status bit is used for the *column* status. An example of the status vector implementation is also shown in Figure 3. The row and column status vectors are first initialized to all zeroes. The details of the algorithm are as follows.

Step 1. Row decoding of the received encoded matrix. If the error is correctable, the error bit indicated by the syndrome is flipped. The corresponding row status vector position is set according to the mapping in Figure 3.

Step 2. Column decoding of the updated matrix. First, the individual syndromes of each column are calculated in parallel. If a column syndrome is nonzero, there are two possible scenarios, depending on the error position indicated by that syndrome (e_{syn}) and the row status vector.

(a) If the row state corresponding to e_{syn} is not "00" (e.g., if an error occurs in $d_{3,3}$ and the third row's state is "10"), e_{syn} is flipped.

(b) If the corresponding row state value is "00", e_{syn} may be incorrect (e.g., if there is a double error in the column, as shown in Figure 3). If three or more syndromes in the column decoders have the same value, then e_{syn} is flipped. If only one or two syndromes are the same, no correction is performed and the column status vectors are set to "1" to be used in the next step.

Step 3. Row decoding the matrix after changes from Step 2. The syndrome for each row is recalculated. If there are still two errors in one row, the column status vector is used to indicate which columns need to be corrected. If only one error remains in the row, the row syndrome will be used to do the correction.

Figure 4 illustrates how the algorithm is used to decode a rectangular four-error pattern. In Step 2, because only two column syndromes are the same and the corresponding row state is zero, the column syndromes are not used, and "1"s are recorded in those column states. In Step 3, each row still detects two errors, so the column status vector is used to indicate which positions need to be fixed. In this way, the system is able to correct rectangular error patterns.

A comprehensive simulation of all possible error patterns consisting of five random errors or fewer was performed, verifying that the proposed iterative decoding method operates correctly.

4. IMPLEMENTATION OF THE PROPOSED SCHEME

In this section, we present realization of the proposed error correction scheme and iterative decoding of Hamming product codes.

4.1. Block diagram of proposed error correction scheme

Figure 5 shows a block diagram of the transceiver design used in our proposed error correction scheme. In the transmitter, the input message is encoded using a two-dimensional Hamming product code, which is realized by serially concatenating row and column encoders. The original message is first transmitted with row parity check bits; column parity check bits are stored into a transmitter buffer. The column parity check bits are kept in the transmitter buffer until an acknowledgement/negative acknowledgement (ACK/NACK) signal indicating the status of the previous transmission is received. If an NACK signal is received, the stored column parity check bits are sent to the receiver. Triple modular redundancy is implemented to protect the ACK/NACK signal against errors. The transmitter's operation can be described using the flow chart of Figure 6(a).

Figure 5(b) shows a block diagram of the receiver. The proposed three-stage iterative decoding is separated into two parts. The first stage row decoding is always performed, and the last two stages are only performed when necessary, as shown in Figure 5(b). The coded information is first deinterleaved and presented to the row decoder. The row decoder

Rule 1: row status vector

State value	Condition
00	No error
01	Correctable
10	Detectable but uncorrectable

Rule 2: column status vector

State value	Condition
1	Two column syndromes are the same and meet one of the following conditions: 1) More than a single "10" in row status vector 2) The row, where error occurs, has a state "00"
0	Other cases

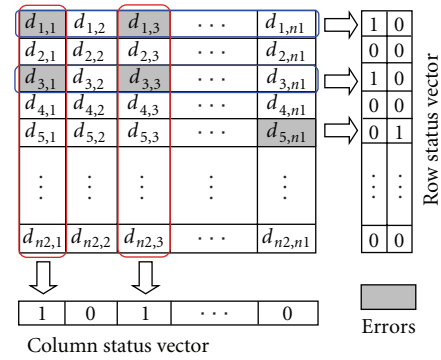


Figure 3: Realization of status vectors.

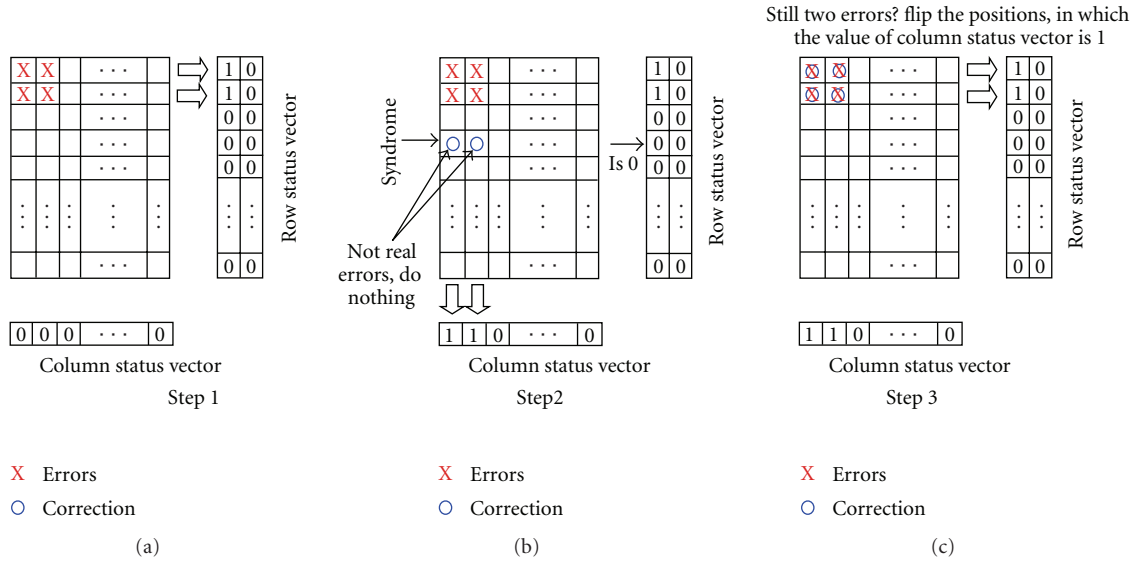


Figure 4: Decoding of rectangular four-error pattern using the proposed iterative decoding scheme.

corrects single errors occurring in different rows. Once a transmission is completed, the receiver sends an ACK/NACK. In order to simplify the hardware implementation, only one retransmission is allowed. When the receiver detects that the errors are uncorrectable, it saves the row decoded message with row parity check bits in a buffer and uses the NACK signal to instruct the transmitter to send the column parity check bits, which are formed based on the original message. When these column parity-check bits are received, they are combined with the previously received row parity-check bits to perform the last two stages of iterative decoding. The receiver's operation can be described using the flow chart of Figure 6(b).

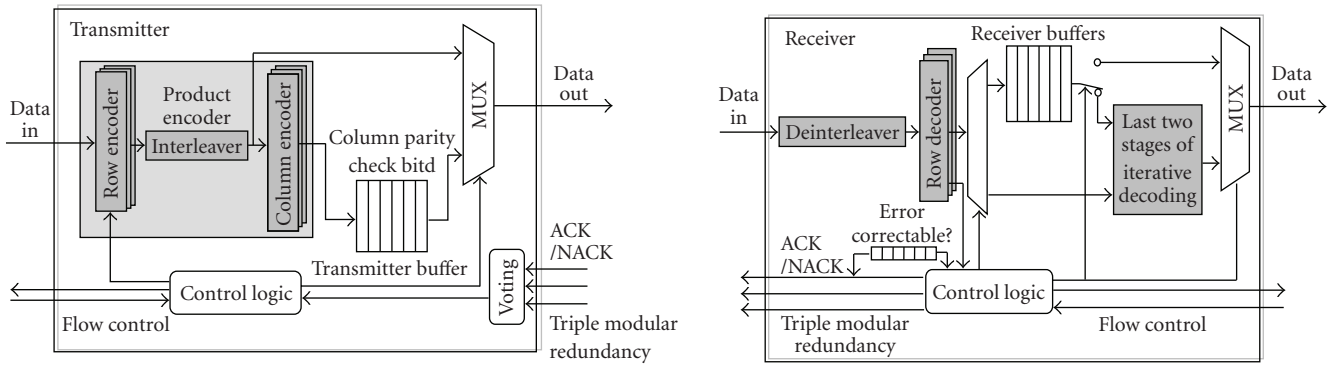
4.2. Transmitter design

Figure 7(a) shows hardware implementation of the transmitter. Assume that a K-bit input information is used to construct a $C_{pc}(n_1 \times n_2, k_1 \times k_2)$ product code. The input message is separated into k_2 groups with group size k_1 bits (each group constitutes a single row). In order to minimize the encoding latency, multiple row encoders are implemented. Each group is encoded using a row encoder and generates n_1 bits as outputs. All $k_2 \times n_1$ outputs of the row encoders are fed into a row column interleaver, which is implemented by hardwire direct connection with the mapping relation in

$$i_{\text{output}} = k_2 * \bmod (i_{\text{input}}, n_1) + \left\lfloor \frac{i_{\text{input}}}{n_1} \right\rfloor, \tag{3}$$

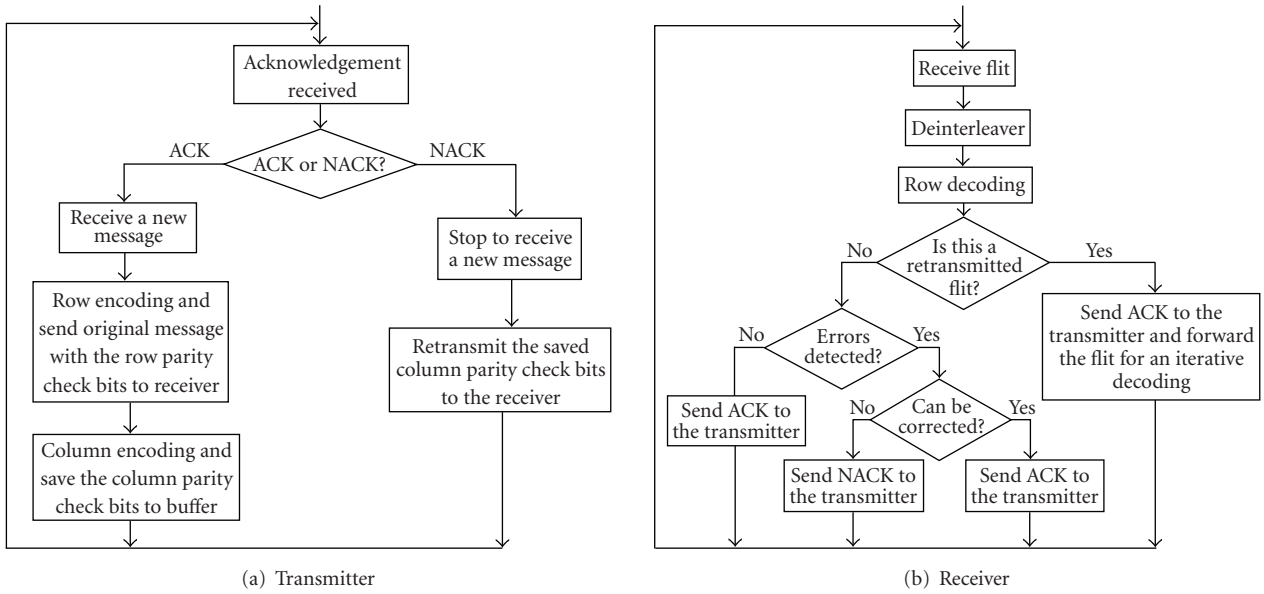
where $0 \le i_{\text{input}}, i_{\text{output}} \le k_2 \times n_1 - 1$. Figure 7(b) demonstrates an example of the mapping relation for a 64-bit input information encoded using $C_{pc}(7 \times 22, 4 \times 16)$. The 64-bit input information is split into four identical groups. Each group is encoded with extended Hamming (22,16) code and the outputs of these encoders are interleaved. The inputs are named $d_1, \ldots, d_{64}$ and the check bits are named $p_1, \ldots, p_{24}$. Check bits $p_1, \ldots, p_6$ are calculated from the first group; check bits $p_7, \ldots, p_{12}$ are calculated from the second group, and so on. The interleaver outputs are shown in Figure 7(b).

The interleaved row encoder outputs are transmitted to the receiver and fed into n_1 column encoders at the same time. The total $(n_2 - k_2) \times n_1$ parity check bits of



(a) Transmitter (b) Receiver

Figure 5: Block diagram of proposed transceiver.



(a) Transmitter (b) Receiver

Figure 6: Flow chart of proposed transceiver operation.

the n_1 column encoders are saved into a transmitter buffer. MUXes are used to select the outputs of row encoders or the saved column parity-check bits using control signals generated according to the ACK/NACK signal. The required bus width to transmit the encoded information is the maximum value between the first transmission of $k_2 \times n_1$ bits and retransmission of $(n_2 - k_2) \times n_1$ column parity check bits. In the proposed method, retransmission bits $(n_2 - k_2) \times n_1$ are always less than the first $k_2 \times n_1$ transmission bits; zero padding is used in the retransmission, as shown in Figure 7(a).

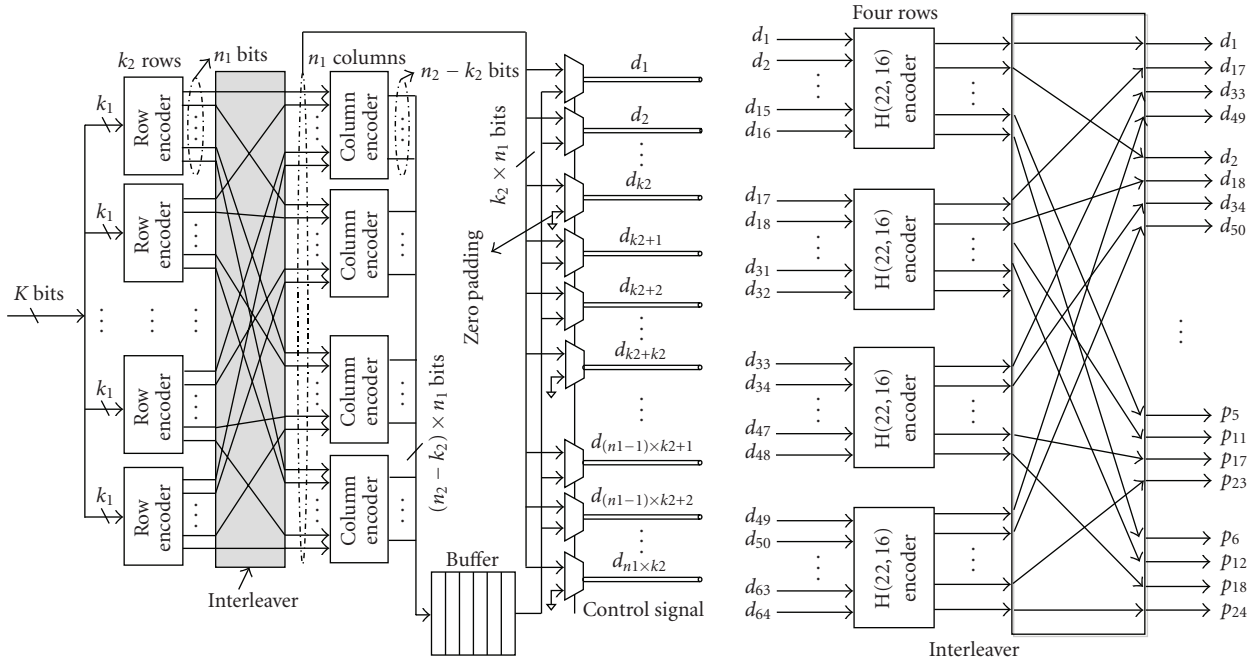
4.3. Receiver design

4.3.1. Block diagram of receiver

Figure 8(a) shows the block diagram of the proposed receiver. In the proposed receiver, the coded information is first deinterleaved then presented to the row decoder. The syndromes of each row are calculated for the received data. Because of our use of SEC-DED codes, a single error can be corrected or two errors can be detected for each row. When all errors are correctable, the receiver sends back an ACK signal to the transmitter and saves the decoded message into a decoding buffer. When the receiver detects two errors in any row, it saves the erroneous message in the decoding buffer and sends an NACK signal to instruct the transmitter to transmit the column parity check bits, which are formed based on the original message. When the column parity check bits are received, they are used with the saved row redundancy information and original information to perform an iterative decoding of product codes. Figure 8(b) shows the implementation of the row decoder.

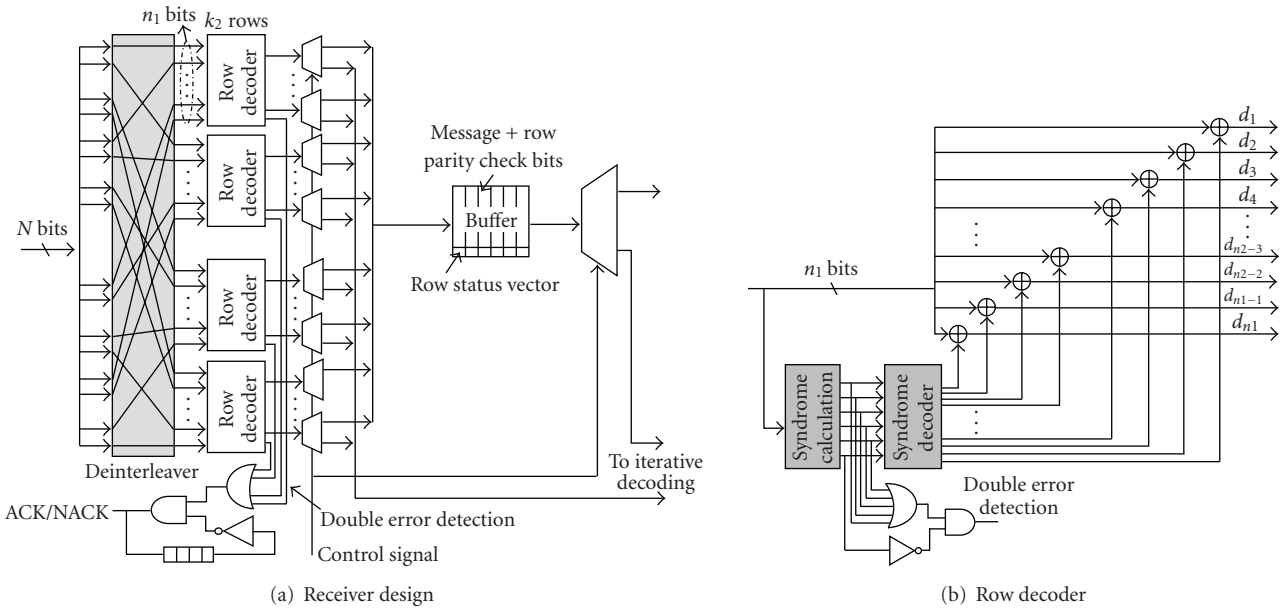
4.3.2. Iterative decoding

In the proposed receiver design, if a retransmission is required, the total row and column parity check bits are



(a) Transmitter design (b) Interleaving for 64-bit input information

Figure 7: Implementation of the proposed transmitter design and interleaver mapping algorithm.



(a) Receiver design (b) Row decoder

Figure 8: Implementation of proposed receiver and row decoder.

used for iterative decoding of Hamming product codes. The block diagram of the proposed three-stage iterative decoding is described in Figure 9. The errors are corrected using the syndrome and row and column status vectors together.

Figure 10(a) shows the block diagram of the column decoder used in the proposed iterative decoder. Unlike a conventional Hamming decoder, where error correction depends only on syndrome values, the error correction in the proposed method uses syndrome values, row status vector, and the relation among the syndromes of all column decoders. In the column decoder, the syndrome values are needed to determine whether more than two column syndromes are the same. This is realized by comparing the error vectors instead of syndromes directly. The column syndromes are first decoded into error vectors. Because only

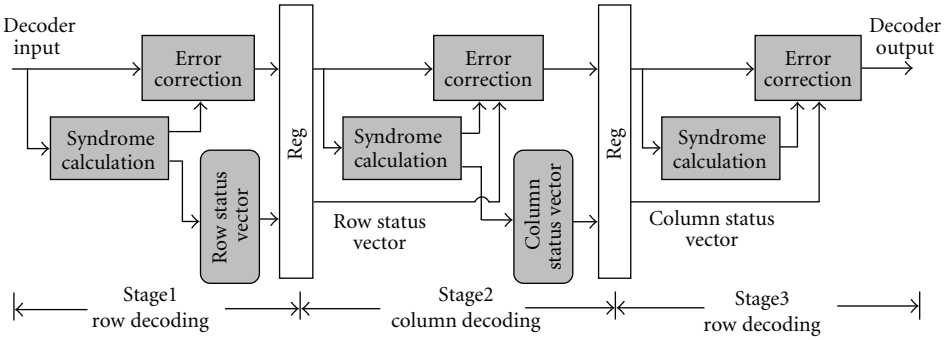
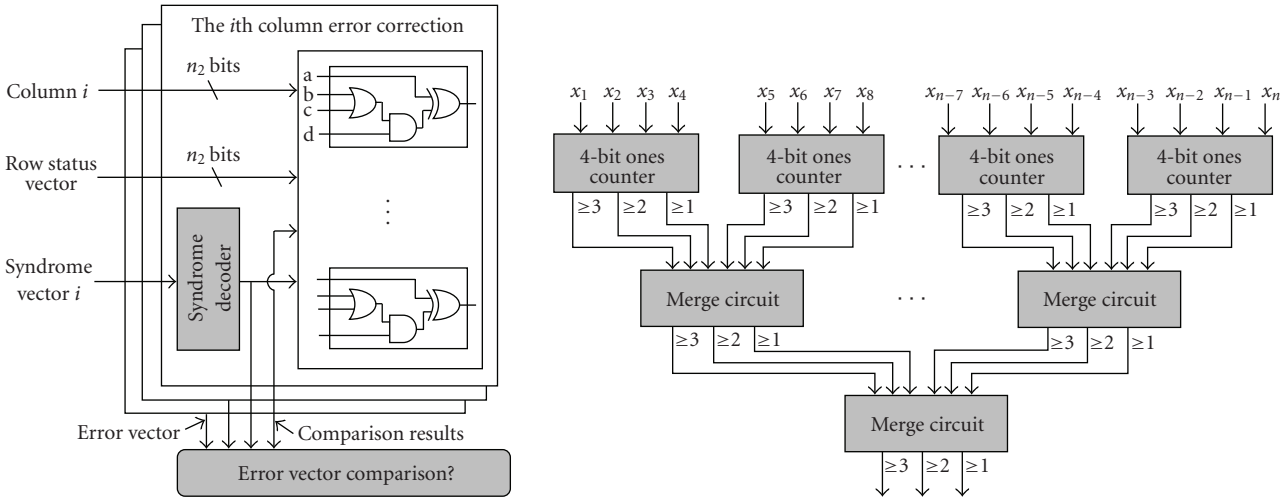


Figure 9: Proposed iterative decoding method of two-dimensional Hamming product codes.



(a) Column decoder

(b) Combinational ones counter

Figure 10: Implementation of the column decoder in the proposed iterative decoding method.

one bit is "1" in the error vector, error vectors are considered to be the same only if they have "1" in the same position. Thus, the comparison is converted into judging whether more than two "1"s exist in the same position. This can be implemented by a ones counter, shown in Figure 10(b). The combinational ones counter can be implemented by two-level logic. The first one is multiple four-bit ones counter, which will generate the ones number for each four-bit input. The outputs of the four-bit ones counter will be combined together using a multilevel merge circuits. Figure 11(a) shows the implementation of the four-bit ones counter, which is realized by modifying cellular threshold circuits in [22]. For every four-bit input, the ones counter circuit generates three outputs indicating the number of ones greater than or equal to 1, 2, and 3. The number of ones is equal to the number of error vectors having the same value. Two four-bit ones counters can be combined together using a merge circuit, shown in Figure 11(b). The merge circuit combines two four-bit ones counters and still generates three outputs indicating the number of ones greater than or equal to 1, 2, and 3. Multiple level merge circuits can be used in a tree structure.

5. EVALUATION OF THE PROPOSED ERROR CONTROL SCHEME

The performance of the proposed coding scheme was evaluated in terms of complexity, reliability, throughput, power, and energy consumption, for a 64-bit input message arranged into a 4×16 two-dimensional matrix. Each row was encoded using an extended H(22,16) Hamming code, obtained by shortening an H(32,26) code. Each column was encoded using an H(7,4) Hamming code. An 88-bit bus equal to the outputs of row encoders was used to transmit the encoded information. The proposed error correction scheme was developed and verified in Verilog HDL. To measure power consumption, the Verilog HDL code was mapped to gate level schematic. The gate level schematic was simulated in Cadence Spectre using predictive technology model (PTM) CMOS 45 nm technology [23]. The wire dimension was estimated for 45 nm technology using simple 1/S scaling rule [24]; the values are shown in Table 1 [25]. Registers were applied between encoder and links, and also between links and decoder to improve the system clock frequency by allowing the encoder, interconnect links, and decoder to operate in a pipelined manner.

Table 1: Parameters used for link model.

Width (μm)	Space (μm)	Thickness (μm)	Height (μm)	Dielectric constant
0.31	0.31	0.83	0.14	2.1

Table 2: Bus widths, delay, and equivalent gate number of different error control schemes.

Coding scheme	Bus width	Delay	Equivalent gates
Hamming (71,64)	71	0.53 ns	2.6 kgates
ARQ (CRC-5)	69	0.47 ns	2.8 kgates
Hybrid ARQ (extended Hamming (72,64))	72	0.57 ns	4.7 kgates
The proposed scheme	88	0.61 ns	11.9 kgates

Simulation results were compared to forward error correction (FEC) schemes using Hamming code H(71,64), ARQ schemes using standardized CRC-5 with generator polynomial $x^5 + x^2 + 1$, and HARQ using extended Hamming code H(72,64). In order to improve throughput, Go-back-N retransmission policy [26] was applied to the ARQ and HARQ schemes. Table 2 shows the comparative results for different error control schemes.

5.1. Delay and complexity

Table 2 shows the delay of different error correction schemes. The Hamming encoder was implemented as a simple XOR tree. Instead of using linear feedback shift registers to generate check bits for CRC codes, a parallel implementation method [27] was employed to reduce the large latency of CRC codes at a minor cost of complexity. The decoder delay, typically much larger than encoder delay, is reported in Table 2. As expected, the decoder delay of the ARQ scheme using CRC-5 is the smallest, compared to other error control schemes, because only syndrome is calculated and no error correction is needed in this scheme. In the proposed method, the iterative decoding process was implemented using a three-stage pipelined architecture. The worst delay occurs in the column decoding stage. Compared to Hamming H(71,64) and HARQ, the delay of our proposed error control scheme increases by about 15% and 10%, respectively, primarily because of error pattern comparison overhead.

Table 2 also shows the complexity of different error control schemes in term of equivalent two-input NAND gate count. In go-back-N retransmission policy, N flits will be retransmitted if an NACK signal is received. Thus, a transmitter buffer is needed to store these N flits in ARQ and HARQ schemes. The number N is dependent on the round trip transmission delay. In our simulation, $N = 4$. Besides the transmitter buffer, a receiver buffer was needed in our method to store the row decoded message for iterative decoding, when necessary. The result shows that FEC scheme using H(71,64) has the least equivalent gate count complexity because the encoder and syndrome calculation circuits were implemented as simple XOR trees; also no buffers were needed in this scheme. In the proposed method, the part of the receiver buffer which stores the original message is shared with the routing buffer used for routing and flow control purpose [28]. This greatly reduces the buffer size required. The equivalent gate count of our proposed method increases about 2.5 times compared to that of HARQ scheme, because of overhead associated with iterative decoder.

5.2. Reliability

On-chip communication errors can be attributed to voltage perturbations induced by noise from many sources. A simple model proposed in [29] assumes that an error occurs at a certain probability for a single wire when a transition occurs. The error probability of a single wire can be modeled by (see [29])

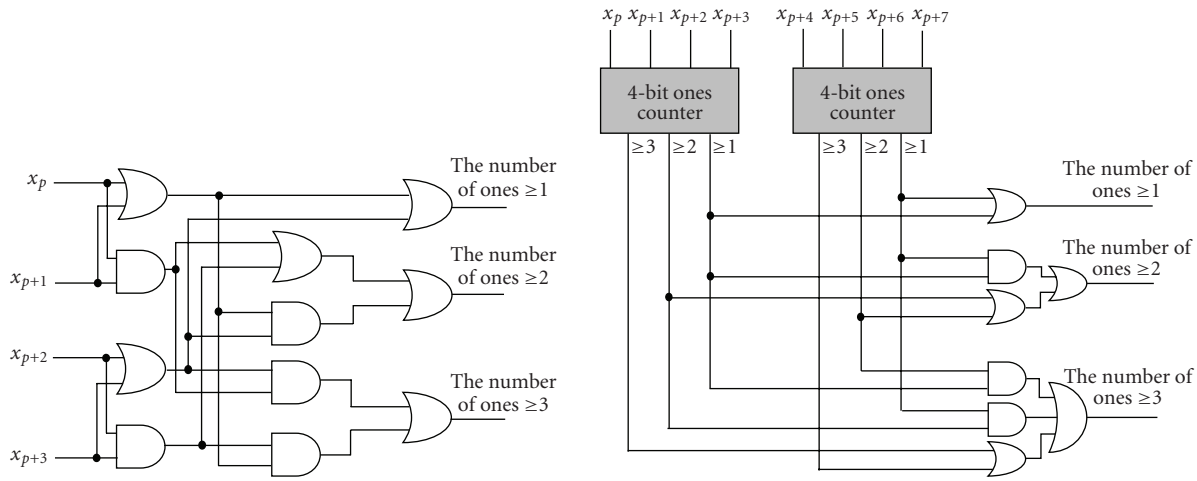
$$\varepsilon = Q\left(\frac{V_{\text{swing}}}{2\sigma_N}\right) = \int_{V_{\text{swing}}/2\sigma_N}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-y^2/2} dy, \tag{4}$$

where V_{swing} is the link swing voltage and σ_N is the standard deviation of the noise voltage, which is assumed to be a normal distribution. The model in (4) assumes that the probability of error in each wire is independent. As technology scales, the probability of one event causing spatial burst errors increases. The model can be extended to account for burst errors by assuming that a fault affects its neighboring wires with a certain probability P_n. The probability can be obtained by simulating data transmissions across an interconnect link and counting the number of burst errors caused by coupling noise [15].

The residual flit-error rate, which is the probability of decoding failure or error, was used to evaluate the reliability of different error control schemes. Hamming codes can only correct one error at a time and if more than one error occurs in a codeword it will lead to uncorrected errors. In ARQ and HARQ scheme, an encoded message is accepted by the receiver only if it either contains no errors or contains an undetectable error pattern. The residual flit-error rate of ARQ and HARQ scheme can be expressed in (see [19])

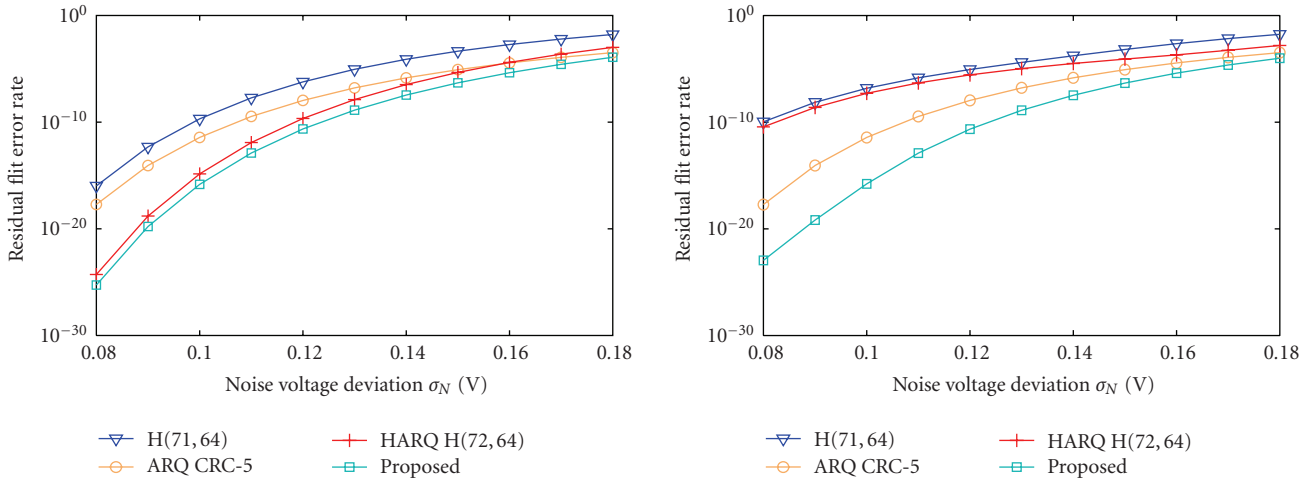
$$\text{RFER}_{\text{ARQ}} = P_e + P_d \cdot P_e + P_d^2 \cdot P_e + \cdots, \tag{5}$$

where P_e is the probability of an undetectable error pattern and includes the decoding error probability in an HARQ scheme, and P_d is the probability that an error can be detected and is not correctable. In the experiments,



(a) Four-bit ones counter (b) Merge circuit

Figure 11: Implementation of combinational ones counter.



(a) Multiple independent errors (b) The combination of multiple independent errors and burst errors

Figure 12: Residual flit-error rate comparison of different error control schemes.

a comprehensive simulation with different error patterns was performed to decide the P_e and P_d values.

Figure 12 shows the residual flit-error rate of different error control schemes as a function of noise voltage deviation. Two noise scenarios were considered: multiple independent errors and a combination of multiple random and burst errors. A supply voltage of 1 V was assumed in model (4). Figure 12(a) shows the residual flit-error rate of different error control schemes with only multiple independent errors. The simulation results show that the proposed coding scheme achieves several order reduction in residual flit error rate compared to H(71,64) and ARQ CRC-5 scheme because the undetectable error probability P_e of the proposed method is much smaller than that of H(71,64) and CRC-5. The proposed method can detect any two random errors and most combinations of three random errors. The HARQ H(72.64) scheme achieves performance close to the proposed method when only multiple independent errors were considered because H(72,64) can also detect two random errors. Figure 12(b) shows the residual flit-error rate of different error control schemes when a combination of multiple random and burst errors was considered. Up to five-bit errors were considered. The results show that the residual flit-error rate of H(71,64) and HARQ decreases significantly because the burst error correction or detection capability of Hamming and extended Hamming codes is poor. The residual flit-error rates of the ARQ CRC-5 scheme and the proposed method are almost the same compared to the case of multiple random errors because both can detect or correct up to five burst errors.

Figure 13 shows the residual flit-error rate of different error control schemes, when burst errors of more than five bits were considered. The results show that the residual flit-error rate of both CRC-5 and the proposed method decreases

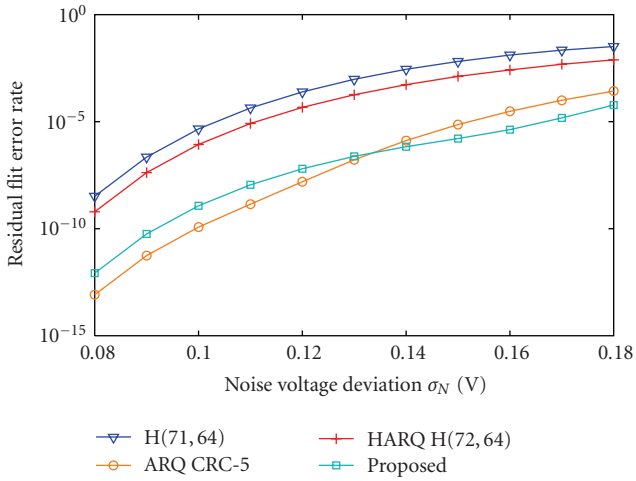


Figure 13: Residual flit-error rate for up to seven-bit burst error modeled.

greatly because those larger bursts exceed the burst error correction or detection capability. The proposed method can still achieve a two-order reduction in flit-error rate compared to HARQ H(72,64) scheme.

5.3. Throughput

Another main concern in on-chip communication is the throughput. In our simulations, go-back-N retransmission policy was applied to improve the throughput of ARQ and HARQ schemes. Compared to stop-and-wait retransmission, the transmitter in go-back-N does not wait for an ACK signal after sending a flit. When an NACK is received, the transmitter resends N flits including the erroneous flit and the succeeding flits that were transmitted during the round-trip delay. Go-back-N achieves efficient bus usage at the cost of hardware complexity. The average clock cycles to successfully transmit a flit in go-back-N can be expressed in (see [19])

$$\begin{aligned} T_{\text{go-back-}N} &= 1 \cdot P_c + (N+1) \cdot P_c \cdot (1-P_c) \\ &\quad + (2N+1) \cdot P_c \cdot (1-P_c)^2 + \cdots, \end{aligned} \tag{6}$$

where P_c is the probability that a received flit contains no error and N is the round-trip delay, which was four cycles in our simulation setup. In our method, retransmission was limited to only one additional flit; thus, the maximum clock cycles to successfully transmit a flit in our method was two.

The throughput of the different error control schemes is compared in Figure 14. The throughput was normalized to the throughput in the case of no errors occurring. Figure 14(a) shows the throughput when only multiple independent errors were included. Figure 14(b) shows the throughput when a combination of multiple random and burst errors was considered. As shown, each scheme achieves nearly the same throughput at low noise environments (small σ_N). As σ_N increases, the throughput decreases in all cases except for the H(71,64) system, which achieves the same throughput independent of the noise condition at the cost of reliability. The ARQ scheme achieves the lowest throughput because retransmission is the only way for it to correct errors. As the noise environment gets worse, the overhead for retransmission increases. Compared to HARQ H(72,64) scheme, the proposed method achieves better throughput because more errors can be corrected during the first transmission. The throughput of the ARQ scheme and the HARQ scheme decreases by about 8% more when a combination of multiple random and burst errors was considered. The effects of burst errors on the throughput of the proposed method are small. The results show that the proposed method achieves 45% and 10% improvement in the throughput under high noise conditions compared to ARQ and HARQ scheme, respectfully, when multiple independent and burst errors are considered.

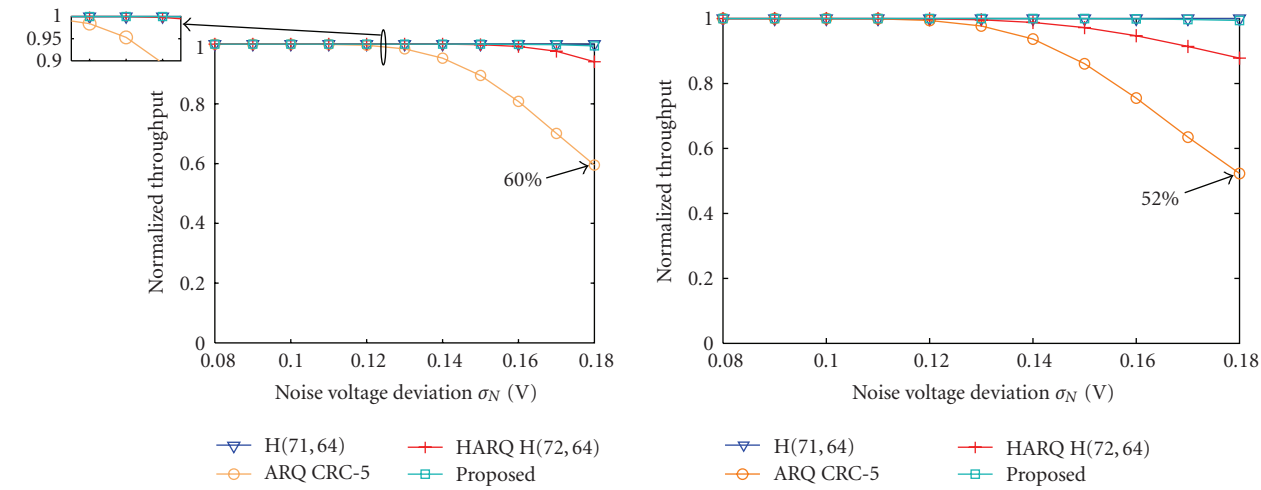
5.4. Power and energy consumption

Figure 15 shows the codec power consumption of different error control schemes. The codec power was measured using the 45 nm predictive technology model technology [23] at a supply voltage of 1 V. The clock frequency was 1 GHz. The codec power consumption is the sum of the encoder and decoder power. The results show that Hamming code H(71,64) consumes the smallest codec power because of the simple XOR tree implementation and no buffers needed in the transmitter. ARQ CRC-5 scheme consumes the smallest decoder power because only the syndrome is calculated and no error correction is needed. In comparison to HARQ, the proposed coding scheme consumes about twice the codec power due to the overhead of the iterative decoder.

The link power PW_L is related to interconnect capacitance C_L, the wire transition probability α, the link width W_L, the link swing voltage V_{swing}, and clock frequency f_{clk}. PW_L can be expressed in

$$PW_L \propto C_L \cdot \alpha \cdot W_L \cdot V_{\text{swing}}^2 \cdot f_{\text{clk}}, \tag{7}$$

where α is assumed to be 0.5 in the simulation and W_L depends on the error control scheme. The link swing voltage V_{swing} is decided by the reliability requirement according to (4). For a given reliability requirement, the error control schemes with low error correction capability need a higher link swing voltage compared to that of the error control schemes with higher error correction capability [7, 10, 14, 29, 30]. Figure 16 shows the link swing voltage of different error control schemes as a function of noise voltage deviation for a given reliability requirement. In the simulations, the residual flit-error probability is assumed to be less than 10^{-20} [10]. The proposed method has the highest error correction capability compared to other error control schemes. To achieve this reliability requirement, the link swing voltages of the proposed method are about 80% and 63% compared to those of the ARQ CRC-5 scheme and the HARQ H(72,64) scheme, respectively. The selection of the appropriate link swing voltage can be a design-time decision. Also, a voltage converter can be used to dynamically select the proper link swing voltage based on a link quality monitor [8]. The energy



(a) Multiple independent errors

(b) A combination of multiple independent errors and burst errors

Figure 14: Throughput comparison of different error control schemes.

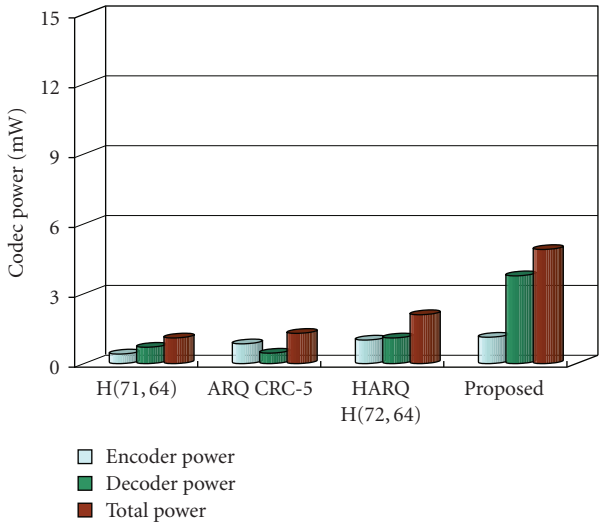


Figure 15: Codec power consumption comparison.

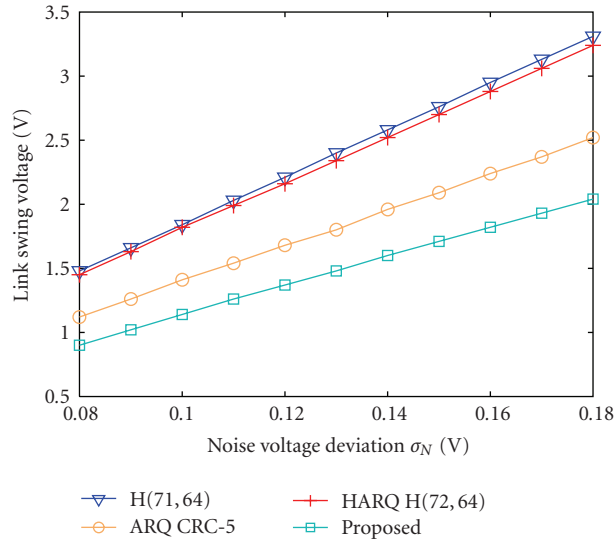
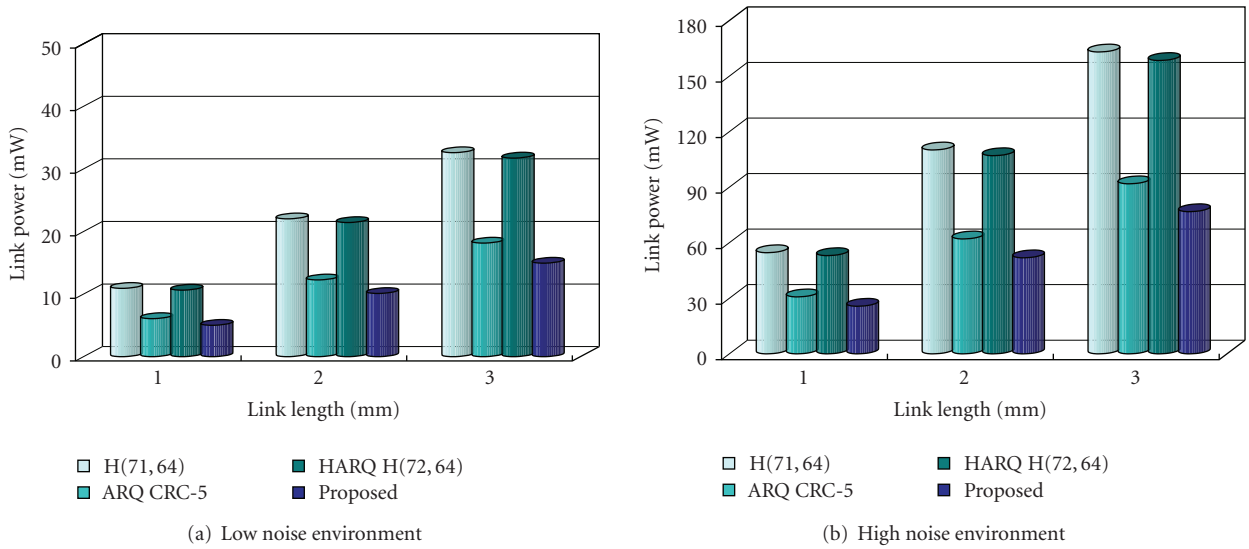


Figure 16: Link swing voltage for different error control schemes.

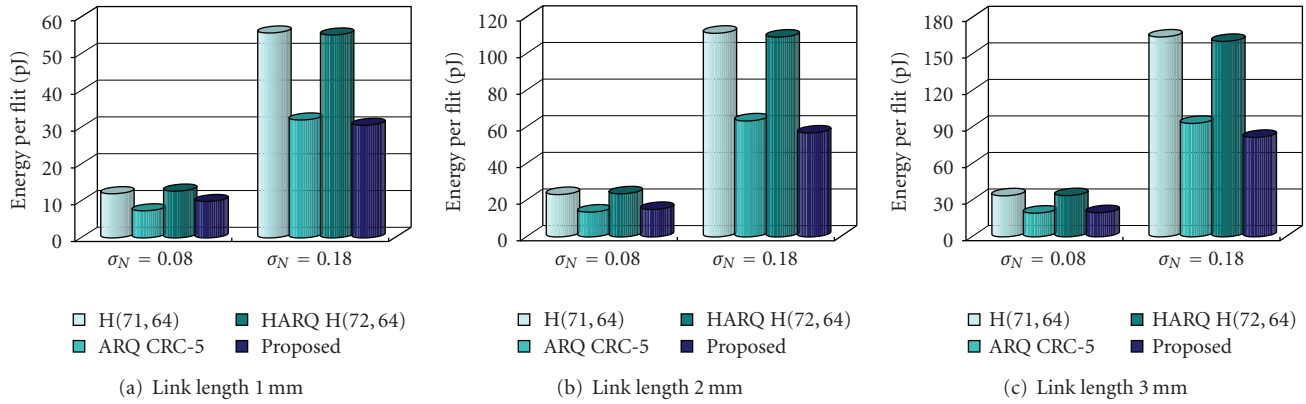
cost of the voltage converter can be neglected because of the high conversion efficiency [30].

In network-on-chip (NoC) architecture, the link length is the distance between two switches, which is decided by the tile block size. In mesh- or torus-shaped NoC design, the links between two switches are generally a few millimeters long wires [28, 31–33]. For example, in Intel's 80-tile NoC architecture [31], each tile has an area of 2 mm × 1.5 mm and 2 mm link length is used. In [32], each tile has an area of 3 mm × 3 mm and 3 mm link length is used. In our experiments, three link lengths, 1 mm, 2 mm, and 3 mm, were examined. The corresponding link resistance and capacitance were calculated using the method in [23]. The link resistance and capacitance are 86 Ω and 218 fF for the 1 mm link, 171 Ω and 436 fF for the 2 mm link, and 256 Ω and 653 fF for the 3 mm link, respectively. The clock frequency is 1 GHz and the power consumption is measured using Cadence Spectre. Figure 17 compares the link power of different error control schemes for a given reliability requirement. The simulation was performed for a low noise environment ($\sigma_N = 0.08$) and a high noise environment ($\sigma_N = 0.18$). The link swing voltages V_{swing} of different error control schemes were obtained by performing a simulation for each noise environment. For the same reliability requirement, the proposed method requires the lowest link swing voltage and thus, the smallest link power of the compared schemes. Figure 17 shows that the proposed method consumes the least link power at each link length and noise environment when the same residual flit-error rate ($\leq 10^{-20}$) is required. More link power is consumed at higher noise environments because higher link swing voltages are needed to achieve the same reliability, as shown in Figure 16.



(a) Low noise environment (b) High noise environment

Figure 17: Link power comparison of different error control schemes for different noise environments.



(a) Link length 1 mm (b) Link length 2 mm (c) Link length 3 mm

Figure 18: Energy comparison of different error control schemes for different link lengths.

Link power is much larger than the codec power from Figure 15 and can dominate the total power consumption of error control schemes [34]. As link length increases, link power consumption increases because of the increased link resistance and capacitance. The link power of the proposed method is about 83% and 48% compared to the link power of the ARQ CRC-5 scheme and the link power of the HARQ H(72,64) scheme, respectively.

The average energy to successfully transmit one flit, E_{avg}, is used as the metric to measure the energy consumption. E_{avg} can be expressed in

$$E_{\text{avg}} = (E_{e1} + E_{l1} + E_{d1}) + P_d(E_{e2} + E_{l2} + E_{d2}), \quad (8)$$

where E_{e1}, E_{l1}, and E_{d1} are the energy consumption of the encoder, links, and the decoder in the first transmission. E_{e2}, E_{l2}, and E_{d2} are the extra energy consumption of the encoder, links, the decoder, and buffers when a retransmission is required. P_d is the probability that an error can be detected and is not correctable. P_d is equal to zero for FEC schemes. Compared to the HARQ scheme and the proposed method, the ARQ CRC-5 scheme has a larger value of P_d, which increases of the retransmission energy consumption.

For a given reliability requirement, the proposed method consumes the largest codec energy but the least link energy. We evaluated whether such a link energy reduction was beneficial in terms of average energy consumption E_{avg}. Figure 18 compares E_{avg} for different error control schemes at the same reliability requirement. The simulation was performed at different noise environments for link lengths of 1 mm, 2 mm, and 3 mm. The simulation results show that the ARQ CRC-5 scheme achieves the least average energy consumption E_{avg} at low noise environment ($\sigma_N = 0.08$) because of the relatively smaller codec energy and link energy consumption. As the noise voltage deviation increases, the average energy consumption of the ARQ CRC-5 scheme increases faster than the average energy consumption of the proposed method because the ARQ CRC-5 scheme has larger link energy consumption. At the higher noise environment ($\sigma_N = 0.18$), the proposed method yields the least energy

consumption because the smaller link energy consumption counterbalances the larger codec energy consumption. In that environment, the proposed method consumes 5% and 43% less energy compared to that of the ARQ CRC-5 scheme and that of the HARQ H(72,64) scheme, respectively, for a link length of 1 mm. As the link length increases, the proposed method can benefit more from the smallest link energy consumption. When the link length is 3 mm, the energy consumption of the proposed method is about 13% and 50% less than that of the ARQ scheme and HARQ scheme, respectively.

6. DISCUSSION

In the proposed method, the encoded message is separated into two transmissions. The reliability of the proposed method depends on both the error detection capability in the first transmission and error correction capability for the iterative decoding method. In the first transmission, the error patterns with single errors in different rows are corrected and the error patterns with two errors in a row are detected. The proposed method is capable of detecting 75% of all random independent five-error patterns and 100% of error patterns consisting of two burst errors of up to three bits each (e.g., one three-bit burst error and another single-bit random error) in the first transmission. The iterative decoding algorithm can correct up to five-bit errors once the row and column parity check bits are received. Also, our method can correct permanent errors that are distributed in different rows. More complex codes can be used to increase the reliability of on-chip communication. Our primary concern is energy efficiency; combining error correction with retransmission is a good approach that balances energy and reliability. The codec area overhead is relatively small when compared to the millions of transistors integrated in a system-on-chip (SoC) [35].

7. CONCLUSION

In this paper, we presented an error control scheme combining Hamming product codes with a simplified type-II hybrid ARQ for on-chip interconnects. The efficient combination of powerful product codes with retransmission shows a good balance between the reliability and energy efficiency in error scenarios where a combination of multiple random and burst errors is considered. Moreover, an efficient iterative decoding method of Hamming product codes is proposed. The proposed decoding algorithm is easily realized in a three-stage pipelined architecture by modifying the conventional row-column decoding algorithm, with a small increase in delay and complexity.

The performance of the proposed method was evaluated in terms of reliability, throughput, and energy consumption. Several orders of reduction in residual flit-error rate can be achieved using the proposed method when multiple errors are considered. Compared to an ARQ scheme using CRC codes and an HARQ scheme using extended Hamming codes, the proposed method achieves about 45% and 10% improvement in throughput, respectively, in high noise environments. The high reliability of the proposed method can permit a reduction in the link swing voltage and consequently, reduction in communication energy. The decreased link energy counterbalances the overhead of codec energy. For a given reliability requirement, the proposed error control scheme can achieve up to 50% reduction in energy consumption compared to other error correction schemes in high noise environments.

REFERENCES

[1] International Technology Roadmap for Semiconductors, 2006, http://public.itrs.net.

[2] C. Constantinescu, "Trends and challenges in VLSI circuit reliability," *IEEE Micro*, vol. 23, no. 4, pp. 14–19, 2003.

[3] M. Lajolo, M. S. Reorda, and M. Violante, "Early evaluation of bus interconnects dependability for system-on-chip designs," in *Proceedings of the 14th IEEE International Conference on VLSI Design (VLSID '01)*, pp. 371–376, Bangalore, India, January 2001.

[4] G. De Micheli and L. Benini, *Networks on Chips: Technology and Tools*, Elsevier, Amsterdam, The Netherlands, 2006.

[5] A. Maheshwari, W. Burleson, and R. Tessier, "Trading off transient fault tolerance and power consumption in deep submicron (DSM) VLSI circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 3, pp. 299–311, 2004.

[6] F. Caignet, S. Delmas-Bendhia, and E. Sicard, "The challenge of signal integrity in deep-submicrometer CMOS technology," *Proceedings of the IEEE*, vol. 89, no. 4, pp. 556–573, 2001.

[7] D. Bertozzi, L. Benini, and G. De Micheli, "Error control schemes for on-chip communication links: the energy-reliability tradeoff," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 6, pp. 818–831, 2005.

[8] L. Li, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "Adaptive error protection for energy efficiency," in *Proceedings of International Conference on Computer Aided Design (ICCAD '03)*, pp. 2–7, San Jose, Calif, USA, November 2003.

[9] D. Rossi, A. K. Nieuwland, A. Katoch, and C. Metra, "Exploiting ECC redundancy to minimize crosstalk impact," *IEEE Design and Test of Computers*, vol. 22, no. 1, pp. 59–70, 2005.

[10] S. R. Sridhara and N. R. Shanbhag, "Coding for system-on-chip networks: a unified framework," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 6, pp. 655–667, 2005.

[11] P. P. Pande, A. Ganguly, B. Feero, B. Belzer, and C. Grecu, "Design of low power & reliable networks on chip through joint crosstalk avoidance and forward error correction coding," in *Proceedings of the 21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT '06)*, pp. 466–476, Arlington, Va, USA, October 2006.

[12] D. Rossi, P. Angelini, and C. Metra, "Configurable error control scheme for NoC signal integrity," in *Proceedings of the 13th IEEE International On-Line Testing Symposium (IOLTS '07)*, pp. 43–48, Crete, Greece, July 2007.

[13] S. Murali, T. Theocharides, N. Vijaykrishnan, M. J. Irwin, L. Benini, and G. De Micheli, "Analysis of error recovery schemes for networks on chips," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 434–442, 2005.

[14] A. Ejlali, B.M. Al-Hashimi, P. Rosinger, and S. G. Miremadi, "Joint consideration of fault-tolerance, energy-efficiency and performance in on-chip networks," in *Proceedings of the*

Conference on Design, Automation and Test in Europe (DATE '07), pp. 1647–1652, Nice, France, April 2007.

[15] H. Zimmer and A. Jantsch, "A fault model notation and error-control scheme for switch-to-switch buses in a network-on-chip," in *Proceedings of the International Workshop on Hardware/Software Codesign (CODES/ISSS '03)*, pp. 188–193, Newport Beach, Calif, USA, October 2003.

[16] T. Lehtonen, P. Liljeberg, and J. Plosila, "Online reconfigurable self-timed links for fault tolerant NoC," *VLSI Design*, vol. 2007, Article ID 94676, 13 pages, 2007.

[17] R. M. Pyndiah, "Near-optimum decoding of product codes: block turbo codes," *IEEE Transactions on Communications*, vol. 46, no. 8, pp. 1003–1010, 1998.

[18] F. Chiaraluce and R. Garello, "Extended Hamming product codes analytical performance evaluation for low error rate applications," *IEEE Transactions on Wireless Communications*, vol. 3, no. 6, pp. 2353–2361, 2004.

[19] S. Lin, D. J. Costello Jr., and M. J. Miller, "Automatic-repeat-request error-control schemes," *IEEE Communications Magazine*, vol. 22, no. 12, pp. 5–17, 1984.

[20] S. Lin and D. J. Costello, *Error Control Coding*, Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2004.

[21] L. Ping, S. Chan, and K. L. Yeung, "Iterative decoding of multi-dimensional concatenated single parity check codes," in *Proceedings of the IEEE International Conference on Communications (ICC '98)*, vol. 1, pp. 131–135, Atlanta, Ga, USA, June 1998.

[22] S. M. Reddy and J. R. Wilson, "Easily testable cellular realizations for the (exactly p)-out-of-n and (p or more)-out-of-n logic functions," *IEEE Transactions on Computers*, vol. 23, no. 1, pp. 98–100, 1974.

[23] Arizona State University, Predictive Technology Model, http://www.eas.asu.edu/~ptm.

[24] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Design Perspective*, Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

[25] X. Sheng, I. Benito, and W. Burleson, "Thermal impacts on NoC interconnects," in *Proceedings of the 1st International Symposium on Networks-on-Chip (NOCS '07)*, p. 220, Princeton, NJ, USA, May 2007.

[26] D. Bertozzi and L. Benini, "Xpipes: a network-on-chip architecture for gigascale systems-on-chip," *IEEE Circuits and Systems Magazine*, vol. 4, no. 2, pp. 18–31, 2004.

[27] M.-D. Shieh, M.-H. Sheu, C.-H. Chen, and H.-F. Lo, "A systematic approach for parallel CRC computations," *Journal of Information Science and Engineering*, vol. 17, no. 3, pp. 445–461, 2001.

[28] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 1025–1040, 2005.

[29] R. Hegde and N. R. Shanbhag, "Toward achieving energy efficiency in presence of deep submicron noise," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 4, pp. 379–391, 2000.

[30] F. Worm, P. Ienne, P. Thiran, and G. De Micheli, "A robust self-calibrating transmission scheme for on-chip networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 1, pp. 126–139, 2005.

[31] S. R. Vangal, J. Howard, G. Ruhl, et al., "An 80-tile sub-100-W teraFLOPS processor in 65-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 29–41, 2008.

[32] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 684–689, Las Vegas, Nev, USA, June 2001.

[33] J. S. Kim, M. B. Taylor, J. Miller, and D. Wentzlaff, "Energy characterization of a tiled architecture processor with on-chip networks," in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED '03)*, pp. 424–427, Seoul, Korea, August 2003.

[34] A. Jantsch, R. Lauter, and A. Vitkowski, "Power analysis of link level and end-to-end data protection in networks on chip," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '05)*, vol. 2, pp. 1770–1773, Kobe, Japan, May 2005.

[35] K. Sakamura, "Future SOC possibilities," *IEEE Micro*, vol. 22, no. 5, pp. 7–9, 2002.