High-Efficiency Processing Schedule for Parallel Turbo Decoders Using QPP Interleaver

Cheng-Chi Wong and Hsie-Chia Chang

***Abstract*—This paper presents a high-efficiency parallel architecture for a turbo decoder using a quadratic permutation polynomial (QPP) interleaver. Conventionally, two half-iterations for different component codewords alternate during the decoding flow. Due to the initialization calculation and pipeline delays in every half-iteration, the functional units in turbo decoders will be idle for several cycles. This inactive period will degrade throughput, especially for small blocks or high parallelism. To resolve this issue, we impose several constraints on the QPP interleaver and rearrange the processing schedule; then the following half-iteration can be executed before the completion of the current half-iteration. Thus, it can eliminate the idle cycles and increase the efficiency of functional units. Based on this modified schedule with 100% efficiency, a parallel turbo decoder which contains 32 radix-2^4 SISO decoders is implemented with 90 nm technology to achieve 1.4 Gb/s while decoding size-4096 blocks for 8 iterations.**

***Index Terms*—Parallel turbo decoder and quadratic permutation polynomial (QPP) interleaver.**

I. Introduction

THE turbo code, which can achieve near Shannon limit performance via iterative decoding process, is an impressive forward error correction technique [1]. It is typically constructed with two constituent convolutional codes and one interleaver. The corresponding codeword includes the systematic data along with two parity checks, which are encoded from the information in original sequence and in permuted sequence, respectively. During the decoding procedure, we need one soft-in/soft-out (SISO) decoder to calculate soft output values and several memories to store received codewords and temporary results. The SISO decoder can utilize the maximum *a posteriori probability* (MAP) algorithm [2] to obtain the log-likelihood ratio (LLR) and extrinsic information of each constituent convolutional code. The LLR is used to make a decision, while the extrinsic information is treated as the *a priori* probability estimation for the other constituent code. Such soft value computation of each constituent code is called one half-iteration, and two successive half-iterations make a complete iteration. The decoding flow alternates between these components until certain stopping criteria are satisfied.

The interleaver to generate the permuted sequence is crucial to the error-correcting capability, but its pseudo random property complicates parallel execution and restricts throughput enhancement. To solve this problem, many contention-free interleavers which allow multiple SISO decoders to process every codeword under trivial memory mapping are proposed [3]. The quadratic permutation polynomial (QPP) interleaver is such an interleaver and is adopted by 3GPP LTE standard due to its simple formula and outstanding performance [4], [5]. For a size-N block, the QPP interleaving formula is

$$F(x) = f_1 x + f_2 x^2 \pmod{N}, \tag{1}$$

where x is the original address, and $F(x)$ stands for the interleaved address. The interleaving parameters, f_1 and f_2, are determined by the block size [4]. In general, there will be many valid pairs of f_1 and f_2. After a careful search of these parameters, we can get the contention-free interleavers with superior error-correcting capability [4].

For practical turbo decoders, the processing schedule is also an important design issue. The percentage of time used for generating outputs within each half-iteration, denoted as *operating efficiency*, will affect the throughput calculation [6]. This factor is mostly dependent on block size, and its value is low for decoding small blocks. A parallel architecture with P SISO decoders will divide every received size-N block into P size-(N/P) subblocks. However, the overall speedup will be less than the expected P because a shortened subblock size leads to lower operating efficiency. Several methods have been proposed to improve this factor. The modified schedule in [7]–[9] has shorter latency and higher efficiency at the expense of extra storage elements. The work in [10] restricts the interleaving rule so that the decoder can carry out partial processes of the current half-iteration and partial processes of the following half-iteration concurrently. Based on these two ideas, a high-efficiency schedule and special QPP interleavers are presented. Moreover, a design that uses 32 radix-2^4 SISO decoders, each of which can complete the executions of 4 trellis stages per cycle, to decode the rate-$1/3$ and size-4096 block is implemented. With the proposed approaches, its operating efficiency can be enhanced from 66.7% to 100%.

The paper is organized as follows. Section II introduces the relation between processing schedule and operating efficiency. Section III illustrates how to achieve high operating efficiency

Manuscript received June 01, 2010; revised October 03, 2010; accepted November 05, 2010. Date of publication January 20, 2011; date of current version May 27, 2011. This work was supported in part by the NCTU-MTK Research Center and in part by the National Science Council (NSC), Taiwan, under Contract NSC 99-2220-E-009-033. This paper was recommended by Associate Editor V. Gaudet.

The authors are with the Department of Electronics Engineering and Institute of Electrics, National Chiao Tung University, 300 Hsinchu, Taiwan (e-mail: hcchang@mail.nctu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2010.2097690

1549-8328/$26.00 © 2011 IEEE

by imposing constraints on the QPP interleaver. Section IV discusses the proposed schedule and the corresponding SISO decoder. Section V gives the simulation and implementation results; and Section VI concludes this paper.

II. Processing Schedule and Operating Efficiency

A practical SISO decoder usually adopts the sliding window method to execute the path metric and LLR for less overhead [11]. Fig. 1(a) shows the typical processing schedule within two half-iterations [12], where W_i stands for the ith window. In each window, the dummy backward path metric β_d, forward path metric α, backward path metric β, and LLR are computed successively. Each half-iteration can be divided into four partitions: δ_a is the pipeline delay for accessing data from memories to SISO decoders; τ_a is the interval for initial metric calculation between the first input and the first output; τ_b is the time for deriving all LLR's and decisions; and δ_b is the pipeline delay for writing extrinsic information back to memories. The following half-iteration can not start until the processes of the current half-iteration finish. Fig. 1(b) shows when the major operations of every window are executed. In every half-iteration, it takes τ_b cycles to generate the decoding results; so we can define the operating efficiency of one SISO decoder as

$$\eta = \frac{\tau_b}{\delta_a + \delta_b + \tau_a + \tau_b}. \tag{2}$$

All of δ_a, δ_b, τ_a, and τ_b are determined by decoder architecture and block size. Both δ_a and δ_b are constant because they only comprise memory access time and pipeline delay time. Here we represent the window length as L and use a radix-2^ν SISO decoder to process ν successive trellis stages within one cycle. From Fig. 1(a), τ_a includes $2L/\nu$ cycles and few pipeline delays. When L and ν are fixed, τ_a is almost the same in any block. By contrast, τ_b is in proportion to the window number. If the SISO decoder has to process $K\ (= N/L)$ windows, then τ_b requires $(K \times L)/\nu$ cycles. Therefore, the η of the typical schedule can be expressed as

$$\frac{K \times L/\nu}{(K+2) \times L/\nu + \Delta},$$

where Δ is the summation of δ_a, δ_b, and the constant part of τ_a. It is obvious that smaller K results in lower η, especially when L/ν is close to Δ. Based on our design characteristics, we assume that Δ consumes 8 cycles ($\delta_a = 4$ and $\delta_b = 2$) and that L/ν equals 8 cycles ($L = 32$ and $\nu = 4$). These assumptions are helpful in getting the η's numerical value in the following discussion.

From [13], the parallel design using P radix-2^ν SISO decoders with clock frequency $\mathcal{F}$, efficiency η, and iteration number $\mathcal{I}$ can achieve the throughput as

$$\frac{P \times \nu \times \mathcal{F} \times \eta}{2 \times \mathcal{I}}. \tag{3}$$

Using large ν and large P is a common method to enhance throughput. However, the increment of the two factors will cause negative effects on $\mathcal{F}$ and η. The choice of high ν indicates the extension of the critical path and a decrease of $\mathcal{F}$. Also, each SISO decoder processes K/P windows in the parallel architecture, so we need to replace K with K/P while computing η. The overall speedup with respect to P is influenced by the change of η. For example, a block with $K = 128$ has 97.7% operating efficiency. After using 32 parallel decoders, the η of each subblock is reduced to 57.1% $(K/P = 4)$. The actual speedup is about 18 times rather than the expected 32 times. Due to the declining η, parallel processing has only limited success.

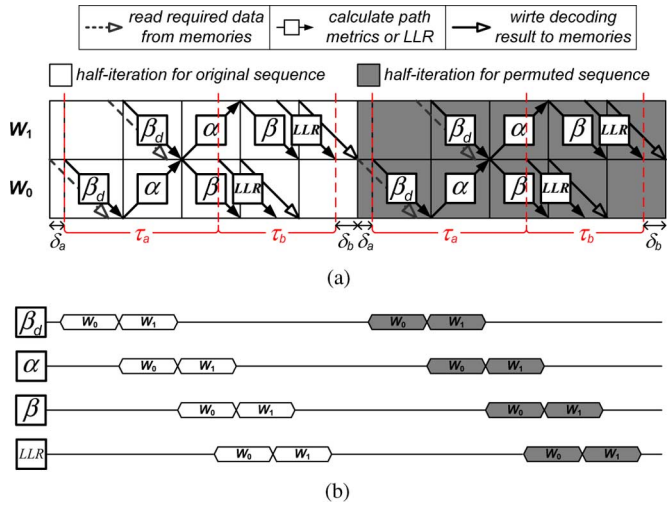

Fig. 1. Conventional processing schedule with two windows. (a) Decoding flow for two half-iterations. (b) Active periods of main components.

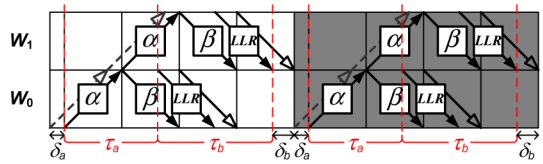

Fig. 2. Modified schedule with two windows ($\eta = 50\%$).

From (2), δ_a, δ_b, and τ_a dominate the operating efficiency η. A trivial solution for raising η is to shorten their execution time. Instead of the dummy calculation, the works in [7]–[9] utilize the boundary of α and β from the previous iteration to initialize α and β in the current iteration. Fig. 2 shows the modified processing schedule. This initialization approach can reduce L/ν cycles from τ_a, so the general expression of η becomes

$$\frac{(K/P) \times (L/\nu)}{(K/P+1) \times (L/\nu) + \Delta}.$$

Here we take the above example with $K = 128$ and $P = 32$ again; the η of each SISO decoder with $K/P = 4$ is improved from 57.1% to 66.7%, so the speedup is about 21 times. This schedule with smaller τ_a can reduce the damage of the declining η in a highly parallel architecture.

III. QPP Interleaver Constraints for Overlapping Half-Iterations

The operating efficiency for small blocks is far from 100% in normal processing schedules like Fig. 1 or 2. The major reason of this disappointing result is the necessary execution time (δ_a, δ_b, and τ_a). Reducing them to zero within each individual half-iteration is difficult, but starting the succeeding half-iteration in advance can achieve the same effect. The design could compute the LLR of one component code and the path metric of

another component code at the same time [10]. For simplicity, this method is called *overlapping half-iterations*. However, the turbo decoder would encounter several problems. The first one is the shortage of reliable *a priori* probability estimation for the new half-iteration, which might cause a large performance loss. The other problem is memory conflict; this would arise when one memory address is accessed by writing the output of the unfinished half-iteration and reading the input for the new half-iteration simultaneously. To avoid the above problems, the corresponding interleaver must guarantee no mapping between the processed data of different constituent codes involved in the overlapping region.

The turbo decoder needs appropriate interleaving rules and processing schedules for successful overlapping half-iterations. In our approach, all original data are divided into two groups according to their window indexes. All permuted data are also classified into two window groups. The second step is mapping one window group in the original sequence exactly onto one window group in the permuted sequence. This rule implies that the remaining data of both sequences will be correlated with each other. We then take advantage of the processing schedule in [7]–[9], where all windows can be processed in arbitrary order by utilizing previous α's and β's. By arranging the execution order of all windows properly, one window group in the original sequence and its uncorrelated group in the permuted sequence can be decoded at the same time. Thus, the decoding process with overlapping half-iterations can be accomplished. In this section, two types of interleaver constraints will be introduced. As the parameters meet either of them, the interleaver can possess the mentioned characteristic. In addition, a turbo code with such a restricted interleaving rule can obtain comparable performance as a conventional turbo code.

A. Interleaver Constraints for Overlapping Half-Iterations

First of all, a proper address expression is necessary in the proposed strategy. The x in (1) is replaced with $(sL+j)$, indicating the jth symbol in the sth window. With the above substitution, the interleaving address is rewritten as index q_j in the Q_sth window

$$\begin{aligned} F(sL+j) &= f_1 sL + f_2 s^2 L^2 + 2f_2 jsL + f_1 j + f_2 j^2 \\ &\triangleq Q_s L + q_j \pmod N. \end{aligned} \tag{4}$$

Note that $0 \le s, Q_s < N/L$ and $0 \le j, q_j < L$. The Q_s is determined by s and j, whereas q_j only depends on j [4]. We can divide both sides in (4) by L and get Q_s as (5)

$$Q_s = f_1 s + f_2 s^2 L + 2f_2 js + \left\lfloor \frac{f_1 j + f_2 j^2}{L} \right\rfloor \quad \left(\mathrm{mod} \frac{N}{L}\right). \tag{5}$$

The q_j is the remainder of this division

$$q_j = f_1 j + f_2 j^2 \pmod L.$$

The interleaver design involves classification methods and mapping rules, and only s and Q_s are under consideration. All windows are categorized according to s and Q_s modulo 2; then even s's and odd s's must be mapped onto different groups of Q_s's after interleaving. Three basic restrictions are set in this paper: $2L \mid N$, $2 \nmid f_1$, and $2 \mid f_2$, where the notation "$a \mid b$" indicates a divides b; otherwise "$a \nmid b$." The first restriction, $2L \mid N$, promises that K is an even number. As $2 \nmid f_1$ and $2 \mid f_2$, it is trivial that $f_1 s \equiv s \pmod 2$, $f_2 s^2 L \equiv 0 \pmod 2$, and $2f_2 js \equiv 0 \pmod 2$. With these properties, (5) leads to

$$Q_s \equiv s + \left\lfloor \frac{f_1 j + f_2 j^2}{L} \right\rfloor \pmod 2. \tag{6}$$

The term within the floor function determines the relation between s and Q_s. It is essential that, for all possible j's, the results of the floor function modulo 2 are the same; they are all 0's or all 1's, and then either of $Q_s \equiv s \pmod 2$ or $Q_s \not\equiv s \pmod 2$ can hold. For this purpose, we must find special f_1 and f_2. The following constraints on these parameters can be used for $Q_s \equiv s \pmod 2$.

Proposition 1: If f_1, f_2, and L can satisfy (7a), (7b), and (7c), then Q_s and s can be congruent modulo 2

$$\begin{cases} L \mid (f_1 - 1) & \text{(7a)} \\ L \mid f_2 & \text{(7b)} \\ (f_1 - 1)/L \equiv f_2/L \pmod 2. & \text{(7c)} \end{cases}$$

Proof:

a) The floor function in (6) can be rewritten as

$$\left\lfloor \frac{f_1 j + f_2 j^2}{L} \right\rfloor = \left\lfloor \frac{(f_1 - 1)j}{L} + \frac{f_2 j^2}{L} + \frac{j}{L} \right\rfloor.$$

b) With (7a) and (7b), both $(f_1 - 1)j/L$ and $f_2 j^2/L$ are integers. If j is an odd number, $(f_1 - 1)j/L$ and $f_2 j^2/L$ are congruent modulo 2 after applying (7c); otherwise, they are both even integers. In either case, these constraints make their summation always be an even integer. They then can be moved out of the floor function

$$\frac{(f_1 - 1)j}{L} + \frac{f_2 j^2}{L} + \left\lfloor \frac{j}{L} \right\rfloor.$$

c) Due to $0 \le j < L$, the range of j/L is $0 \le j/L < 1$, and $\lfloor j/L \rfloor$ will be eliminated.

d) As a result, the last term in (6) is divisible by 2 and can be removed; and $Q_s \equiv s \pmod 2$ holds true for $s = 0$ to $(N/L - 1)$ and $j = 0$ to $(L - 1)$. ■

In the proof, decomposing (6), checking the summation of $(f_1 - 1)j/L$ and $f_2 j^2/L$, and removing $\lfloor j/L \rfloor$ are critical steps. If we replace (7a) by $(f_1 - i)j/L$ with another integer i, the decomposition will generate $\lfloor ij/L \rfloor$, and the elimination will fail. Without (7c), an odd number j might result in an odd summation of $(f_1 - 1)j/L$ and $f_2 j^2/L$, and it would destroy the consistency of $Q_s \equiv s \pmod 2$ for all j's. This set of constraints lets all even s's map to even Q_s's and odd s's map to odd Q_s's. The window length L is an important factor in finding suitable interleaver parameters. This search process can start by setting $(f_1, f_2) = (L+1, L)$ or $(f_1, f_2) = (2L+1, 2L)$. By adding multiples of $2L$ to f_1 and f_2, we can get many usable (f_1, f_2): $(L + 1 + 2Li_1, L + 2Li_2)$ and $(2L + 1 + 2Li_1, 2L + 2Li_2)$, where i_1 and i_2 are arbitrary integers. The final step is to verify the corresponding performance of the turbo code with these parameters.

B. Variant Constraints for Overlapping Half-Iterations

If the tail-biting technique is applied [14], we can have more choices of the interleaving parameters for overlapping half-iterations. Another constraint set that promises $Q_s \not\equiv s \pmod 2$ in an alternative way will be used for this method. Since the initial and ending states are the same, the whole trellis can be regarded as a loop. The decoding procedure for the original sequence, $(r_0, r_1, \ldots, r_{N-1})$, could be rotated as $(r_\rho, \ldots, r_{N-1}, r_0, \ldots, r_{\rho-1})$. Similarly, the permuted sequence also could be changed from $(\tilde{r}_0, \tilde{r}_1, \ldots, \tilde{r}_{N-1})$ to $(\tilde{r}_\rho, \ldots, \tilde{r}_{N-1}, \tilde{r}_0, \ldots, \tilde{r}_{\rho-1})$. The cyclic shift with offset ρ reorganizes the window classification and allows for different constraints.

The value of ρ relies on a feature of the interleaving rules. For our variant constraints on the QPP interleaver, ρ must be 1. Before presenting another proposition, we have to redefine the indexes for the rotated sequence. Here $(sL + j - 1)$ and $(Q_sL + q_j - 1)$ are expressed as $(s'L + j')$ and $(Q'_sL + q'_j)$, respectively. The relation between the modified indexes and the original indexes can be further derived as follows:

$$\begin{aligned}
&j' = j - 1 \pmod L,\\
&s' = s + \left\lfloor \frac{j-1}{L} \right\rfloor\\
&\quad = \begin{cases} s-1 & \left(\bmod \frac{N}{L}\right) \quad \text{if } j = 0\\ s & \left(\bmod \frac{N}{L}\right) \quad \text{if } j \neq 0, \end{cases}\\
&q'_j = q_j - 1 \pmod L,\\
&\qquad\qquad \text{and}\\
&Q'_s = Q_s + \left\lfloor \frac{q_j - 1}{L} \right\rfloor\\
&\quad = \begin{cases} Q_s - 1 & \left(\bmod \frac{N}{L}\right) \quad \text{if } q_j = 0\\ Q_s & \left(\bmod \frac{N}{L}\right) \quad \text{if } q_j \neq 0. \end{cases}
\end{aligned}$$

Each of the $\lfloor (q_j - 1)/L \rfloor$ and $\lfloor (j-1)/L \rfloor$ has two possible values: -1 and 0. Besides, in (4), $j = 0$ implies

$$q_j|_{j=0} = F(sL + 0) \pmod L = 0.$$

Hence, the two terms are equivalent to each other

$$\begin{aligned}
\left\lfloor \frac{q_j - 1}{L} \right\rfloor &= \left\lfloor \frac{j-1}{L} \right\rfloor\\
&= \begin{cases} -1 & \text{if } j = 0 (q_j = 0)\\ 0 & \text{if } j \neq 0 (q_j \neq 0). \end{cases}
\end{aligned} \tag{8}$$

With the modified indexes, the interleaving parameters for $Q'_s \not\equiv s \pmod 2$ can be proven as well as for $Q_s \not\equiv s' \pmod 2$.

Proposition 2: If f_1, f_2, and L can satisfy (9a), (9b), and (9c), then $Q'_s \not\equiv s \pmod 2$ and $Q_s \not\equiv s' \pmod 2$ are true.

$$\begin{cases} L \mid (f_1 + 1) & \text{(9a)}\\ L \mid f_2 & \text{(9b)}\\ (f_1 + 1)/L \equiv f_2/L \pmod 2. & \text{(9c)} \end{cases}$$

Proof:

a) By substituting Q'_s for Q_s in (6), we get

$$Q'_s \equiv s + \left\lfloor \frac{q_j - 1}{L} \right\rfloor + \left\lfloor \frac{f_1 j + f_2 j^2}{L} \right\rfloor \pmod 2. \tag{10}$$

b) The second floor function in (10) can be rewritten as

$$\left\lfloor \frac{(f_1+1)j}{L} + \frac{f_2 j^2}{L} - \frac{j}{L} \right\rfloor.$$

Due to (9a), (9b), and (9c), the summation of the first two terms is an even integer and can be moved out of the floor function. This outcome lets (10) change to

$$\begin{aligned}
Q'_s &\equiv s + \left\lfloor \frac{q_j - 1}{L} \right\rfloor + \frac{(f_1+1)j}{L} + \frac{f_2 j^2}{L} + \left\lfloor -\frac{j}{L} \right\rfloor\\
&\equiv s + \left\lfloor \frac{q_j - 1}{L} \right\rfloor + \left\lfloor -\frac{j}{L} \right\rfloor \pmod 2.
\end{aligned} \tag{11}$$

c) The $\lfloor (q_j - 1)/L \rfloor$ term is given in (8). If j is 0, $\lfloor -j/L \rfloor$ is 0; otherwise, its value is -1. Therefore, (11) can be further expressed as $Q'_s \equiv (s-1) \pmod 2$ in (12)

$$Q'_s \equiv \begin{cases} s - 1 + 0 & \pmod 2 \quad \text{if } j = 0\\ s + 0 - 1 & \pmod 2 \quad \text{if } j \neq 0 \end{cases}. \tag{12}$$

d) The relation of Q_s and s' can be found by substituting s' for s in (6) and simplifying the equation with the proposed constraints and mentioned properties

$$\begin{aligned}
Q_s &\equiv s' - \left\lfloor \frac{j-1}{L} \right\rfloor + \left\lfloor \frac{f_1 j + f_2 j^2}{L} \right\rfloor\\
&\equiv s' - \left\lfloor \frac{j-1}{L} \right\rfloor + \left\lfloor -\frac{j}{L} \right\rfloor \pmod 2.
\end{aligned} \tag{13}$$

Then, the value of $\lfloor (j-1)/L \rfloor$ and $\lfloor -j/L \rfloor$ are assigned

$$Q_s \equiv \begin{cases} s' + 1 + 0 & \pmod 2 \quad \text{if } j = 0\\ s' - 0 - 1 & \pmod 2 \quad \text{if } j \neq 0 \end{cases}. \tag{14}$$

e) Finally, (12) and (14) indicate that both $Q'_s \not\equiv s \pmod 2$ and $Q_s \not\equiv s' \pmod 2$ hold true. ■

The constraints set is similar to the preceding one, but it uses $(f_1 + 1)$ to replace $(f_1 - 1)$. It will generate $\lfloor -j/L \rfloor$ by the decomposition as in (11) or (13). Although this term has two possible values, the substitution of s' or Q'_s can offer a complementary number and get a constant result. With (9a), (9b), and (9c), the required property for the overlapping process in the circular trellis structure is allowed. Here we take the example of size-16 blocks with $L = 4$. We represent the original and the permuted sequences as $(r_0, \ldots, r_{15})$ and $(\tilde{r}_1, \ldots, \tilde{r}_{15}, \tilde{r}_0)$, respectively. As the interleaver meets

$$\begin{cases} 4 \mid (f_1 + 1)\\ 4 \mid f_2\\ (f_1 + 1)/4 \equiv f_2/4 \pmod 2, \end{cases}$$

the two windows, $\{r_0, r_1, r_2, r_3\}$ and $\{r_8, r_9, r_{10}, r_{11}\}$ $(s = 0, 2)$, are mapped to $\{\tilde{r}_5, \tilde{r}_6, \tilde{r}_7, \tilde{r}_8\}$ and $\{\tilde{r}_{13}, \tilde{r}_{14}, \tilde{r}_{15}, \tilde{r}_0\}$

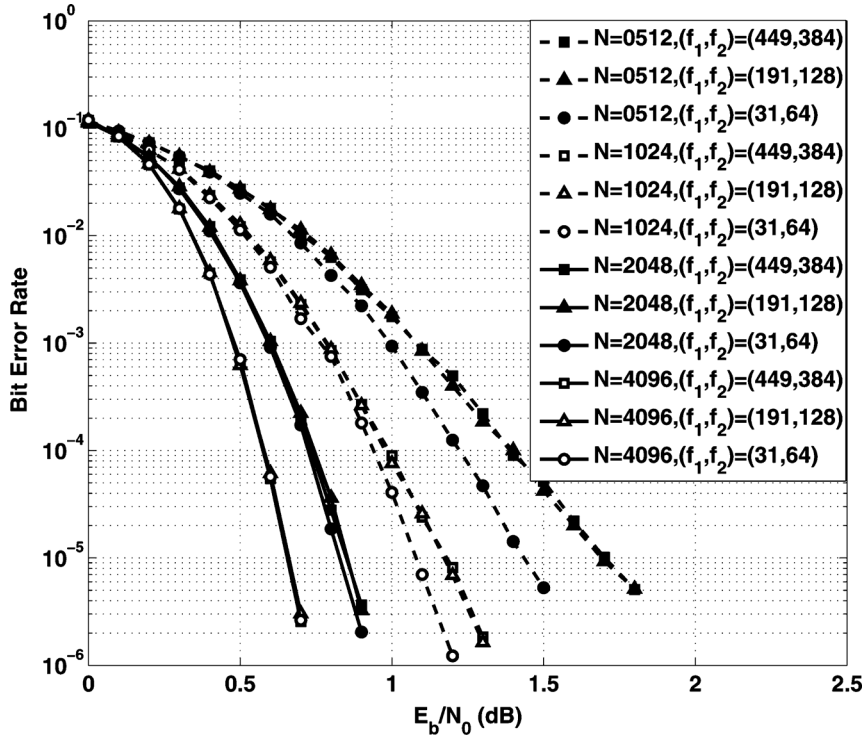

Fig. 3. Performance of rate-$1/3$ turbo codes in an AWGN channel and for BPSK modulation: floating point simulation with Max-Log MAP algorithm, 8 iterations, various block sizes, and three sets of (f_1, f_2).

TABLE I
PROPERTIES OF VARIOUS N AND (f_1, f_2)

(f_1, f_2)	Spread Factor ‖ Minimum Distance		
	$(449, 384)$	$(191, 128)$	$(31, 64)$
$N = 0512$	16 ‖ 31	16 ‖ 36	32 ‖ 33
$N = 1024$	32 ‖ 38	32 ‖ 38	32 ‖ 43
$N = 2048$	64 ‖ 44	64 ‖ 44	32 ‖ 44
$N = 4096$	64 ‖ 44	64 ‖ 44	32 ‖ 44

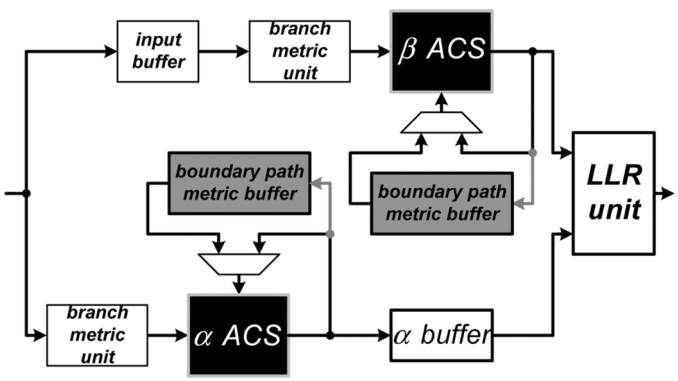

Fig. 4. Architecture of the SISO decoder using previous path metrics.

$(Q'_s = 1, 3)$. The other windows in the original sequence, $\{r_4, r_5, r_6, r_7\}$ and $\{r_{12}, r_{13}, r_{14}, r_{15}\}$ $(s = 1, 3)$, are mapped to $\{\tilde{r}_1, \tilde{r}_2, \tilde{r}_3, \tilde{r}_4\}$ and $\{\tilde{r}_9, \tilde{r}_{10}, \tilde{r}_{11}, \tilde{r}_{12}\}$ $(Q'_s = 0, 2)$. During the search for proper parameters, (f_1, f_2) could be $(L-1, L)$ or $(2L-1, 2L)$ at first. The subsequent flow is to examine the performance of the turbo code while (f_1, f_2) equals $(L-1+2Li_1, L+2Li_2)$ or $(2L-1+2Li_1, 2L+2Li_2)$ with any integers i_1 and i_2.

TABLE II
AREA OF MAIN COMPONENTS IN DIFFERENT RADIX-2^ν SISO DECODERS

Processing schedule	Equivalent gates count ($L = 32$, $K/P = 4$, 5-bit input, and 8-bit metric)			
	Conventional (α,β,β_d)		Modified (α,β)	
	Radix-2^1	Radix-2^4	Radix-2^1	Radix-2^4
Branch metric units	0.25k×3	2.45k×3	0.25k×2	2.45k×2
ACS circuits	2.67k×3	9.32k×3	2.83k×2	9.97k×2
Input buffers	8.35k		4.41k	
Boundary storage	2.57k		7.48k	
Summation	19.7k	46.2k	18.1k	36.7k

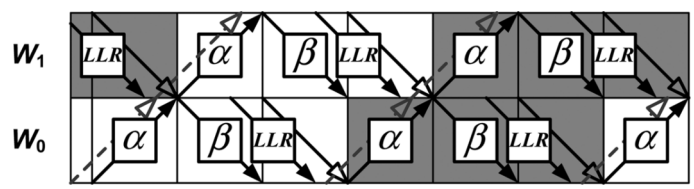

Fig. 5. Overlapping half-iterations with two windows ($\eta = 66.7\%$).

C. Performance of Turbo Code With Constrained Parameters

Fig. 3 shows the performance of turbo codes with different interleaver parameters for $N = 512$, 1024, 2048, and 4096. The simulation utilizes the turbo code generator polynomial in the 3GPP LTE standard and the tail-biting method [5], [14]. Here we apply a typical processing schedule with $L = 32$ [12]. For each block size, $(f_1, f_2) = (449, 384)$ can meet (7a), (7b), and (7c); $(f_1, f_2) = (191, 128)$ can meet (9a), (9b), and (9c); and $(f_1, f_2) = (31, 64)$ are the parameters defined in [5]. The selection of these parameters mainly depends on their spread factors, which can filter inferior interleavers quickly [4], [15], [16]. Both $(f_1, f_2) = (449, 384)$ and $(f_1, f_2) = (191, 128)$ have higher

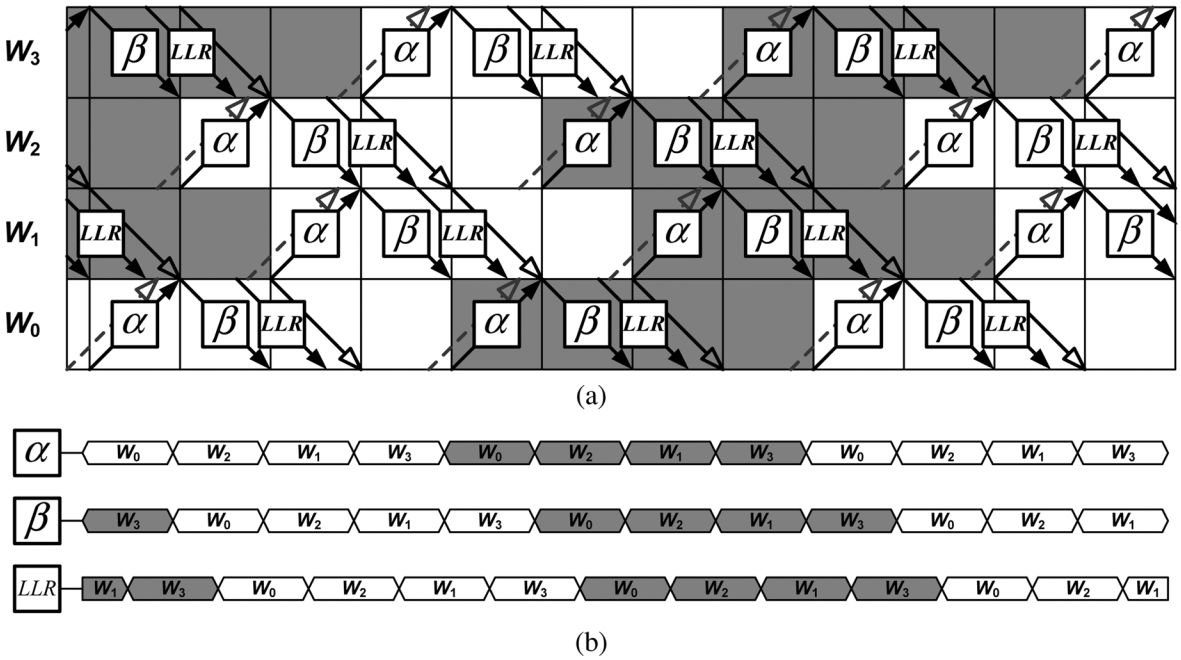

Fig. 6. Overlapping half-iterations with four windows ($\eta = 100\%$). (a) Decoding flow for several half-iterations. (b) Active periods of main components.

spread factors than most interleavers under their respective constraints. Table I shows the properties of these interleavers, including spread factor and minimum distance [17]. Note that these data are helpful for removing interleavers that result in very poor performance. An accurate estimation of performance must consider their distance spectra. At 10^{-5} error rate, the performance loss caused by both restricted interleavers is about 0.3 dB for $N = 512$ and 0.1 dB for $N = 1024$, while the loss becomes insignificant for $N = 2048$ or $N = 4096$. If the window length is 16, the specification parameters $(f_1, f_2) = (31, 64)$ can also satisfy the second proposition. The corresponding decoder could support overlapping half-iterations and get better operating efficiency. However, the short window might reduce error-correcting capability. In comparison to $L = 32$ for $N = 4096$, the code with $L = 16$ has 0.1 dB loss at 10^{-5} error rate. To obtain both benefits of performance and operating efficiency, sufficient window length plays a dominant role in the search of parameters.

IV. Architecture and Schedule of the Proposed SISO Decoder

Both the architecture and the schedule are adjusted to support overlapping half-iterations. The hardware cost of one SISO decoder is greatly affected by the design characteristics and decoding flow. Each radix-2^4 SISO decoder in our design deals with one size-128 subblock. The subblock is divided into four windows with $L = 32$. Since the modified architecture allows an unusual execution order of all windows, the overlapping process becomes likely to work. In the following discussion about the proposed schedule, we only take $Q_s \equiv s \pmod 2$ into account. For the other condition with a tail-biting trellis structure, the corresponding schedule can be derived by rotating either the original sequence or the permuted sequence.

A. Modified SISO Decoder Architecture

Fig. 4 illustrates the block diagram of the modified decoder. Its main components include branch metric units, add-compare-select (ACS) circuits, the LLR unit, and buffers. Unlike the architectures in [12], the circuits for computing β_d are unnecessary, and the amount of input buffers are reduced. However, it needs extra storage elements for previous boundary metrics to initialize the path metrics of each window. If the processed blocks are small, the modified SISO decoder requires smaller buffers, which might reduce hardware cost compared to a conventional SISO decoder. Furthermore, the exploitation of a high-radix structure will make a larger difference between the two SISO decoders since the area of combinational circuits grows rapidly, but the boundary metric storage is unaffected.

Table II gives the synthesis results of the SISO decoder with conventional and modified schedules. The components with significant difference are listed here. The conventional SISO decoder also needs some boundary storage for appropriate initialization in the parallel architecture. When the SISO decoder is designed for $K/P = 4$ and $L = 32$, the circuits with the modified schedule save about 1.5 k and 9.5 k gates count in radix-2^1 and radix-2^4 structures, respectively. Consequently, the parallel architecture using this radix-2^4 SISO decoder for size-128 subblock can benefit area overhead.

B. Proposed High-Efficiency Scheduling

The decoding flow within each half-iteration should be arranged before overlapping two consecutive half-iterations. The guideline for this arrangement is the prevention of mapping correlation among the original and the permuted sequences in the overlapping region. With the relation between s and Q_s, the execution order can be determined. According to the classification method, we let the SISO decoder first handle all the windows

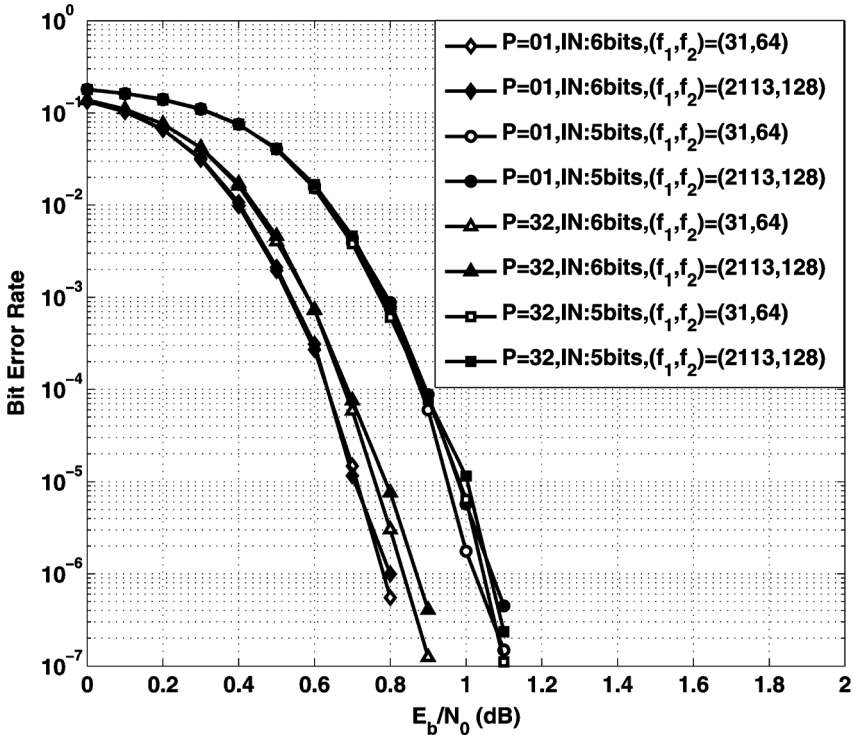

Fig. 7. Performance of rate-1/3 turbo codes in an AWGN channel and for BPSK modulation: fixed point simulation for size-4096 blocks with Max-Log MAP algorithm, $L = 32$, and 8 iterations.

whose indexes are identical in modulo-2 calculation. Thus, the remaining windows in the current half-iteration will belong to the same group. When we choose the uncorrelated windows as the initially processed data in the following half-iteration, a high-efficiency decoding process free from conflict is feasible.

Employing the overlapping half-iterations can raise the operating efficiency, but the maximal improvement will be dominated by subblock size and hardware resources. Since the execution of one window group ($K/2P$ windows) takes $(\delta_a + \delta_b + \tau_a + \tau_b/2)$ cycles, the subsequent half-iteration can start at least $\tau_b/2$ cycles prior to that in the normal schedule. For the overlapping process, the equivalent cycle number of one half-iteration is shortened to $(\delta_a + \delta_b + \tau_a + \tau_b/2)$, and this variation leads to

$$\eta = \frac{\tau_b}{\delta_a + \delta_b + \tau_a + \tau_b/2} = \frac{(K/P) \times (L/\nu)}{(K/2P + 1) \times (L/\nu) + \Delta}. \quad (15)$$

Note that η can exceed 100% if $\tau_b/2 \geq (\delta_a + \delta_b + \tau_a)$. However, the above condition implies a demand for double the number of functional units. If extra overhead is disallowed, then 100 % will be an upper bound of η. Two fundamental examples of the proposed schedule are demonstrated here. In Fig. 5, the subblock contains only two windows; this is the minimum window number K/P due to the basic restriction, $2L \mid N$. The LLR of original-ordered W_1 and the α of permuted-ordered W_0 are computed simultaneously. Compared to Fig. 2, the equivalent η increases from 50% to 66.7%. Fig. 6 shows the schedule with $K/P = 4$ and activities of the main component circuits. The execution order of windows in either half-iteration is W_0, W_2, W_1, and W_3. After deriving the extrinsic information of W_0 and W_2, the SISO decoder continues the unfinished execution of W_1 and W_3. Meanwhile, the process of W_0 and W_2 in different-ordered sequence are activated immediately. It is a remarkable fact that the equivalence, $\tau_b/2 = (\delta_a + \delta_b + \tau_a)$, holds in this case, and η grows rapidly from 66.7% to 100%. When the limited resources set the ceiling of η to 100 %, the proposed schedule with $K/P = 4$ can achieve the optimum improvement.

V. Simulation and Implementation Results

Fig. 7 shows the fixed-point simulation results with various combinations of (f_1, f_2), parallelism, and data width. The constituent convolutional code with generator matrix

$$\left[1 \frac{1 + D + D^3}{1 + D^2 + D^3}\right]$$

is exploited, along with the tail-biting method in [14]. The data width of path metrics is 9 bits for 6-bit input and 8 bits for 5-bit input. Under the same parallelism and data width, the performance with $(f_1, f_2) = (2113, 128)$, which can meet (7a), (7b), and (7c), is similar to that with specification parameters $(f_1, f_2) = (31, 64)$ [5]. In contrast to a design with a single SISO decoder, using 32 parallel SISO decoders will cause less than 0.1 dB performance loss. On the other hand, shorter data width leads to about 0.2 dB loss. If there is sufficient design area, the employment of 6-bit input is preferable; otherwise, we choose another quantization with moderate performance.

Our design with 32 radix-2^4 SISO decoders can process rate-$1/3$ codewords with $N = 4096$ and $L = 32$. It utilizes

TABLE III
CHIP SUMMARY AND COMPARISON

	Proposed*	[18]*	[19]*	[20]	[21]	[22]
Technology	90 nm	65 nm	130 nm	90 nm	130 nm	130 nm
Voltage	0.9 V	N/A	1.2 V	1.0 V	1.2 V	1.2 V
Max. Block Size	4096	6144	480	6144	6144	6144
Max. Parallelism	32	32	10	8	8	8
SISO decoder	radix-2^4	radix-2^2	radix-2^2	radix-2^1	radix-2^2	radix-2^2
Max. Iteration	8	6	4	8	8	5.5
Operating Efficiency	100 %	66.7 %	46.2 %	94.4 %	74.4 %	89.0 %
Area (mm^2)	9.61	N/A	6.66 ($3.20^{\dagger}$)	2.10	10.7 ($5.14^{\dagger}$)	3.57 ($1.71^{\dagger}$)
Frequency (MHz)	175	200	100	275	250	302
Throughput (Mb/s)	1400	711	115	130	186	391
Power (W)	1.356	N/A	0.197	0.219	N/A	0.789
Energy Efficiency (nJ/(bit·iteration))	0.12	N/A	0.43 ($0.12^{\sharp}$)	0.21 ($0.17^{\ddagger}$)	0.61 ($0.16^{\sharp}$)	0.37 ($0.10^{\sharp}$)

* The proposed work and [19] give post-layout simulation results; [18] gives synthesis results.
† Normalization factor is $0.48\ (=(90\text{ nm}/130\text{ nm})^2)$.
‡ Normalization factor is $0.81\ (=(0.9\text{ V}/1.0\text{ V})^2)$.
♯ Normalization factor is $0.27\ (=(90\text{ nm}/130\text{ nm})^2\times(0.9\text{ V}/1.2\text{ V})^2)$.

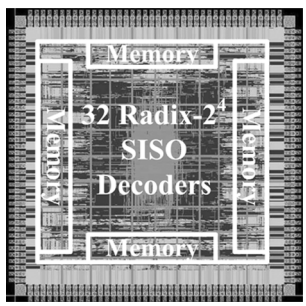

Fig. 8. Layout graph of the proposed design.

a barrel-shift network for lower routing effort [20] and a *two-stage* technique for less overhead [23]. Considering the available area, the quantized data with 5-bit codewords and 8-bit metrics are adopted. In this design, we can configure (f_1, f_2) and select the appropriate processing schedule. If (f_1, f_2) can satisfy the proposed constraints, the turbo decoder can operate with either overlapping process ($\eta = 100\%$) or normal process ($\eta = 66.7\%$); for the other (f_1, f_2), only normal process ($\eta = 66.7\%$) is permitted. Fig. 8 shows the layout graph implemented with 90 nm technology. The post-layout simulation indicates that the core area is 9.61 mm^2 with 85.75% core utilization. Its equivalent gates count is 2833 k. The maximal frequency is 175 MHz; so this design can reach 1.4 Gb/s with $\eta = 100\%$ and 933 Mb/s with $\eta = 66.7\%$ while executing eight iterations. The power consumption of the two modes is 1.356 W and 0.994 W respectively. Table III gives a comparison of several parallel decoders; it lists simulation results in [18], [19] and measurement results in [20]–[22]. The operating efficiency of [18]–[22] is calculated by dividing the throughput by $(P \times \nu \times \mathcal{F}/(2 \times \mathcal{I}))$. We can further find their execution intervals in one half-iteration via (2) and estimate their η's with respect to various subblock sizes. They will suffer from low operating efficiency as the subblock size is small. Dissimilar to those designs, even though N/P is merely 128 in our decoder, all functional units can be fully utilized during the decoding flow with the help of specific parameters and overlapping half-iterations.

VI. CONCLUSION

This paper highlights the importance of operating efficiency in parallel turbo decoders. To improve the inefficient execution caused by high parallelism, a modified processing schedule using previous path metrics instead of dummy metric calculations is exploited. This schedule with a shorter latency and higher efficiency can also allow the SISO decoder to deal with all windows in an arbitrary order. With the flexibility in execution order, the constraints over the QPP interleaver and the corresponding decoding flow are introduced to support overlapping half-iterations. Moreover, these methods are applied to a parallel turbo decoder containing 32 radix-2^4 SISO decoder. From the postlayout simulation with 90 nm technology, the proposed design can achieve 100% operating efficiency and 1.4 Gb/s throughput.

ACKNOWLEDGMENT

The authors thank the Chip Implementation Center, UMC, and NCTU Si2 Lab for their assistance.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE Int. Conf. Commun.*, May 1993, pp. 1064–1070.
[2] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. IT-20, pp. 284–287, Mar. 1974.
[3] E. Boutillon, C. Douillard, and G. Montorsi, "Iterative decoding of concatenated convolutional codes: Implementation issues," *Proc. IEEE*, vol. 95, no. 6, pp. 1201–1227, Jun. 2007.
[4] O. Y. Takeshita, "On maximum contention-free interleavers and permutation polynomials over integer rings," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 1249–1253, Mar. 2006.
[5] *Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access; Multiplexing and Channel Coding (Release 8)*, 3GPP Std. TS 36.212, Rev. 8.5.0, Dec. 2008.

[6] R. Dobkin, M. Peleg, and R. Ginosar, "Parallel interleaver design and VLSI architecture for low-latency MAP turbo decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 4, pp. 427–438, Apr. 2005.
[7] A. Dingninou, F. Rafaoui, and C. Berrou, "Organisation de la mémorie dans un turbo décodeur utilisant l'algorithme sub-MAP," in *Proc. GRETSI*, Sep. 1999, pp. 71–74.
[8] J. Dielissen and J. Huisken, "State vector reduction for initialization of sliding windows MAP," in *Proc. 2nd Int. Symp. Turbo Codes Related Topics*, Sep. 2000, pp. 387–390.
[9] S. Yoon and Y. Bar-Ness, "A parallel MAP algorithm for low latency turbo decoding," *IEEE Commun. Lett.*, vol. 6, no. 7, pp. 288–290, Jul. 2002.
[10] D. Gnaedig, E. Boutillon, J. Tousch, and M. Jézéquel, "Towards an optimal parallel decoding of turbo codes," presented at the 4th Int. Symp. Turbo Codes Related Topics, Munich, Germany, Apr. 2006.
[11] S. A. Barbulescu, "Iterative decoding of turbo codes and other concatenated codes," Ph.D. Dissertation, Univ. South Australia, Adelaide, Australia, 1996.
[12] G. Masera, M. Mazza, G. Piccinini, F. Viglione, and M. Zamboni, "Architectural strategies for low-power VLSI turbo decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 10, no. 3, pp. 279–285, Jun. 2002.
[13] R. Dobkin, M. Peleg, and R. Ginosar, "Parallel VLSI architecure for MAP turbo decoder," in *Proc. IEEE Int. Symp. Personal, Indoor Mobile Radio Commun.*, Sep. 2002, pp. 15–18.
[14] C. Weiβ, C. Bettstetter, S. Riedel, and D. J. Costello, "Turbo decoding with tail-biting trellises," in *Proc. IEEE URSI Int. Symp. Signals, Syst., Electron.*, Oct. 1998, pp. 343–348.
[15] S. Dolinar and D. Divsalar, Weight distribution of turbo codes using random and nonrandom permutations Jet Propulsion Lab., TDA Progress Report 42-122, Aug. 1995.
[16] S. Crozier, "New high-spread high-distance interleavers for turbo-codes," in *Proc. 20th Bienn. Symp. Commun.*, May 2000, pp. 3–7.
[17] S. Crozier, P. Guinand, and A. Hunt, "Estimating the minimum distance of turbo-codes using double and triple impulse methods," *IEEE Commun. Lett.*, vol. 9, no. 7, pp. 631–633, Jul. 2005.
[18] Y. Sun, Y. Zhu, M. Goel, and J. R. Cavallaro, "Configurable and scalable high throughput turbo decoder architecture for multiple 4G wireless standard," in *Proc. IEEE Int. Conf. Appl.-Specific Syst., Archit., Processors*, Jul. 2008, pp. 209–214.
[19] C.-H. Lin, C.-Y. Chen, A.-Y. Wu, and T.-H. Tsai, "Low-power memory reduced traceback MAP decoding for double-binary convolutional turbo decoder," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 5, pp. 1005–1016, May 2009.
[20] C.-C. Wong and H.-C. Chang, "Reconfigurable turbo decoder with parallel architecture for 3GPP LTE system," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 57, no. 7, pp. 566–570, Jul. 2010.
[21] J.-H. Kim and I.-C. Park, "A unified parallel radix-4 turbo decoder for mobile WiMAX and 3GPP-LTE," in *Proc. IEEE Custom Integr. Circuits Conf.*, Sep. 2009, pp. 487–490.
[22] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang, "A 380 mb/s 3.57 mm^2 3GPP-LTE turbo decoder ASIC in 0.13 μm CMOS," in *Proc. IEEE Int. Solid-State Circuit Conf.*, Feb. 2010, pp. 274–276.
[23] C.-H. Tang, C.-C. Wong, C.-L. Chen, C.-C. Lin, and H.-C. Chang, "A 952 Mb/s Max-Log MAP decoder chip using radix-4 $\times$ 4 ACS architecture," in *Proc. IEEE Asian Solid-State Circuits Conf.*, Nov. 2006, pp. 79–82.

Cheng-Chi Wong received the B.S. and Ph.D. degrees in electrical engineering from National Chiao Tung University, Hsinchu Taiwan, in 2004 and 2010, respectively.

His research interests include algorithms and VLSI architecture of error correction codes.

Hsie-Chia Chang received the B.S., M.S., and Ph.D. degrees in electrical engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1995, 1997, and 2002, respectively.

From 2002 to 2003, he was with OSP/DE1 in MediaTek Corp., working in the area of decoding architectures for Combo single chip. In February 2003, he joined the faculty of the Electronics Engineering Department, National Chiao Tung University, where he is currently an associate professor from August 2007. His research interests include algorithms and VLSI architectures in signal processing, especially for error control codes and crypto-systems. Recently, he also committed himself to joint source/channel coding schemes and multi-Gb/s chip implementation for wireless communications.