Turbo Decoder Using Contention-Free Interleaver and Parallel Architecture

Cheng-Chi Wong, Ming-Wei Lai, Chien-Ching Lin, Hsie-Chia Chang, and Chen-Yi Lee, *Member, IEEE*

***Abstract*—This paper introduces a turbo decoder that utilizes multiple soft-in/soft-out (SISO) decoders to decode one codeword. In addition, each SISO decoder is modified to allow simultaneous execution over multiple successive trellis stages. The design issues related to the architecture with parallel high-radix SISO decoders are discussed. First, a contention-free interleaver for the hybrid parallelism is presented to overcome the complicated collision problem as well as reduce interconnection network complexity. Second, two techniques for the high-speed add-compare-select (ACS) circuits are given to lessen area overhead of the SISO decoder. Third, a modification of the processing schedule is made for higher operating efficiency. Two designs with parallel architecture have been implemented. The first design with 32 SISO decoders, each of which processes 2 symbols per cycle, has 160 Mb/s and 0.22 nJ/b/iter after measurement. The second design uses 16 SISO decoders to deal with 4 symbols per cycle and achieves 100% efficiency, leading to 1000 Mb/s and 0.15 nJ/b/iter in post-layout simulation.**

***Index Terms*—Contention-free interleaver, parallel architecture, turbo codes.**

I. Introduction

TURBO code, also known as the parallel concatenated convolutional code, is impressive with the near Shannon limit performance [1]. A rate-1/3 turbo codeword is formed by the systematic data along with two parity checks, which are encoded from the information in original order and the information in interleaved order respectively. The constituent code structure and the interleaving method are critical to good error-correction capability since they allow an efficient iterative decoding process. Many standards such as IEEE 802.16e [2] and 3GPP [3] adopt turbo codes as their forward error correction (FEC) techniques [4]. However, the conventional turbo decoders in current applications are difficult to achieve higher than 100 Mb/s throughput.

Conventional turbo decoder consists mainly of one soft-in/soft-out (SISO) decoder and memories that store both received codewords and temporary decoding results. The SISO decoder utilizes the maximum *a posteriori probability* (MAP) algorithm [5] to calculate the log-likelihood ration (LLR) of each component code, and this algorithm is often approximated to Log-MAP or Max-Log-MAP algorithm in order to reduce implementation complexity [6]. The extrinsic information can be derived from the LLR, and it will be treated as the *a priori* estimation for the other component code. Such soft value calculation of each component code is named as a half-iteration, and two successive half-iterations form one complete iteration. In this paper, the half-iteration for original component codeword is called the normal decoding round, and that for interleaved component is called the permuted decoding round. The decoding flow alternates between these two decoding rounds iteratively until the iteration number reaches a threshold value; then the decisions are outputted at the final round. Both the processing time per round and the total round number are essential for the decoding speed. These issues can be resolved by either modifying decoder architecture [7] or using early stopping rules [8].

Exploiting parallel architecture is an intuitive method to enhance the decoding speed. There are three levels of parallelism: the turbo decoder level, the SISO decoder level, and the trellis stage level [7], [9]. In the turbo decoder level, multiple dedicated turbo decoders are used to decode different codeword blocks independently. In the SISO decoder level, multiple SISO decoders are responsible for the decoding process of single codeword block simultaneously. In the trellis stage level, the computation units inside the trellis-based decoder would process more than one trellis stage every clock cycle. Although the first level can be applied to any turbo decoder, it needs extra memories for multiple codewords and does not shorten the decoding latency. The other two levels have lower cost and less processing time for single codeword, yet they have difficulties in parallel data transmission. Consequently, each level has its advantage, and the design may utilize a hybrid parallelism rather than single parallel level to achieve higher throughput.

The SISO decoder level gets the most attention for its profits in recent years. In this parallel level, one memory module might be accessed by several SISO decoders at the same time, and such collision problem is the major design issue [7]. Spreading concurrent requests over several cycles [10] and storing data by specific rules [11] are two solutions. Both techniques can be compatible with conventional interleavers, but they require some hardware to deal with complicated data flow. For large blocks or high parallelism, the corresponding cost is considerable. Current studies solve the problem by designing the contention-free interleavers that allow instant access and trivial mapping for all sub-blocks [12]–[18]. These interleavers result in relatively lower overhead, and they also possess outstanding error-correcting capability. Furthermore, some contention-free interleavers relieve the complexity of interconnection between SISO decoders and memory modules [19], [20].

Manuscript received March 26, 2009; revised July 07, 2009. Current version published February 05, 2010. This paper was approved by Associate Editor Bevan Baas. This work was supported by the NSC, Taiwan, R.O.C., under contract NSC 97-2220-E-009-017.

The authors are with the Department of Electronics Engineering and Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan (e-mail: ccweng.ee93g@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/JSSC.2009.2038428

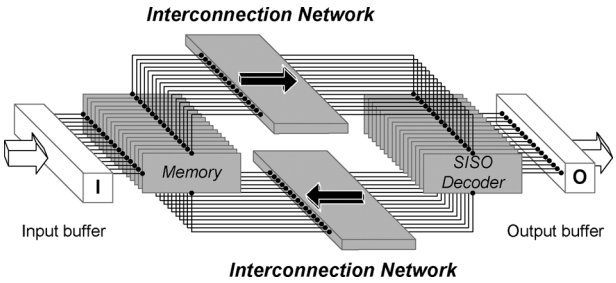


Fig. 1. Parallel turbo decoder architecture.

In addition to increasing the SISO decoder number, our work exploits parallel trellis stages to raise the throughput further. Because the data block will be divided into more segments by these two levels combination, the collision problem and the interconnection implementation become more complicated. Another significant problem is the operating efficiency, which is the ratio between the time for generating decisions and the total execution time of one decoding round [21]–[23]. The ratio is always less than 100% because of some preparatory operations for making decisions, and it might diminish the expected throughput improvement. In this paper, an interleaver for the hybrid parallelism is introduced for less interconnecting complexity; the *two-stage* and *relocation* techniques are proposed for lower area overhead; and the processing schedule is modified for higher operating efficiency.

This paper is organized as follows. Section II describes the design issues of parallel turbo decoder architecture. Section III introduces the interleaving method suitable for the implementation of parallel structure. Section IV illustrates the circuits and operations of add-compare-select (ACS) units for parallel trellis stage level. Section V discusses how to improve the operating efficiency of the SISO decoder. Section VI presents the simulation and implementation results of proposed designs; then Section VII concludes this paper.

II. Design Issues of Parallel Turbo Decoder Architecture

Fig. 1 shows the block diagram of the parallel turbo decoder architecture, in which the interconnection networks handle the data flows between multiple SISO decoders and memories. The throughput of such design can be expressed as follows [21]–[23]:

$$\text{Throughput} = \frac{P_S \times P_T \times F_C \times F_E}{N_R}. \tag{1}$$

The P_S, P_T, F_C, F_E, and N_R stand for the number of parallel SISO decoders, the number of parallel trellis stages, clock frequency, operating efficiency, and total decoding rounds, respectively. Increasing any factor in the numerator on (1) can improve the throughput, but it would cause the growth of the implementation overhead and the decrease of other factors. This section introduces the following design issues: collision problem, which is caused by P_S and P_T; critical path delay, which is relative to P_T and F_C; and operating efficiency, which is influenced by P_S.

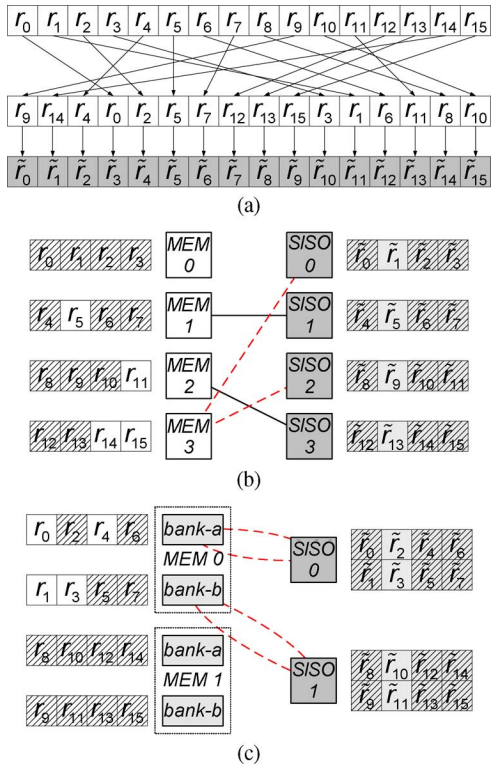


Fig. 2. An example of the collision problem. (a) Interleaving rules. (b) Collision in the parallel SISO decoder level. (c) Collision in the hybrid of parallel SISO decoder level and parallel trellis stage level.

A. Collision Problem

In the parallel SISO decoder architecture, the original size-N block is divided into P_S size-N/P_S sub-blocks. The data inside each sub-block are further divided into P_T groups when parallel trellis stages are applied. We assume that N is a multiple of P_S, and N/P_S is a multiple of P_T in this paper. It is common to store each of P_S sub-blocks in one P_T-bank memory module individually. Such arrangement promises that every SISO decoder accesses data from the same memory module in normal decoding rounds, but it might encounter the collision problem in permuted decoding rounds as either P_S or P_T is larger than 1. Fig. 2 illustrates a collision example of size-16 block. According to the mapping rule in Fig. 2(a), the sequence $(r_0, \cdots, r_{15})$ is reordered and then labeled as $(\widetilde{r}_0, \cdots, \widetilde{r}_{15})$. For the design with $P_S = 4$ and $P_T = 1$, four sub-blocks are stored in separate memory modules in the nature order. Fig. 2(b) indicates the **MEM 3** has trouble with the simultaneous requests for $\widetilde{r}_1$ and $\widetilde{r}_9$. For the design with $P_S = 2$ and $P_T = 2$, two sub-blocks are stored in separate two-bank memory modules, each of which consists one bank for odd-addressed data and one bank for even-addressed data. Such arrangement allows one SISO decoder to access 2 successive data from 2 different banks without contention in normal decoding rounds. However, it complicates the collision situation in permuted decoding rounds. As shown in Fig. 2(c), both **bank-0** and **bank-1** of **MEM 0** have multiple requests when the two SISO decoders access $(\widetilde{r}_2, \widetilde{r}_3)$ and $(\widetilde{r}_{10}, \widetilde{r}_{11})$. Because the problem gets worse under high parallelism, it is inefficient to solve the problem by enlarging the storage bandwidth or using sophisticated control system.

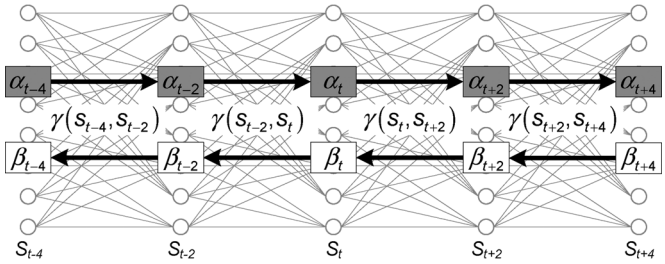


Fig. 3. Trellis diagram of original structure.

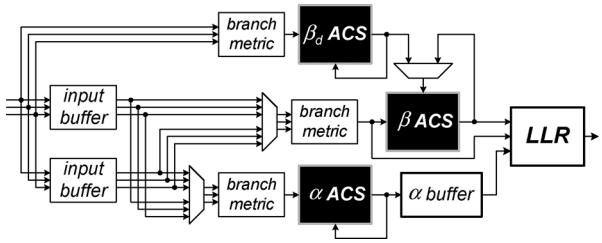


Fig. 4. Conventional architecture of the SISO decoder.

B. Critical Path Delay

The SISO decoder performs the calculations of branch metric γ, forward path metric α, backward path metric β, and LLR. Fig. 3 shows the trellis diagram of the convolutional code along with γ's, α's, and β's. Although the trellis diagram of traditional component code is radix-2 structure, the diagram is modified to the radix-4 scheme by merging two successive trellis stages for consistency with later discussions. From the Max-Log-MAP algorithm, the SISO decoder computes α by (2), β by (3), the log likelihood (LL) by (4), and LLR by (5).

$$\alpha_t(s_t) = \max_{\forall s_{t-2}} \{\alpha_{t-2}(s_{t-2}) + \gamma(s_{t-2}, s_t)\} \quad (2)$$

$$\beta_t(s_t) = \max_{\forall s_{t+2}} \{\beta_{t+2}(s_{t+2}) + \gamma(s_t, s_{t+2})\} \quad (3)$$

$$LL_{u_t}^{u_{t+1}} = \max_{s_t \to s_{t+2}} \{\alpha_t(s_t) + \gamma(s_t, s_{t+2}) + \beta_{t+2}(s_{t+2})\} \quad (4)$$

$$LLR(u_t) = \max_{\forall u_{t+1}} \left\{LL_{u_t=0}^{u_{t+1}}\right\} - \max_{\forall u_{t+1}} \left\{LL_{u_t=1}^{u_{t+1}}\right\} \quad (5)$$

The s_{t-2} and s_{t+2} denote those states connecting to s_t at the $(t-2)$-th and $(t+2)$-th trellis stages respectively, and the $LL_{u_t}^{u_{t+1}}$ stands for log likelihood value of two successive bits. The SISO decoder calculates the γ's at each time instant, adds them to their corresponding path metrics, and then finds the maximum among all summations to update the path metric of each state. Such recursive ACS operation leads to the speed bottleneck of the trellis-based decoding process. The α at first stage and the β at the last stage can be determined by proper initialization or termination. After getting the γ's, α's, and β's, the SISO decoder starts to compute the LL value and LLR.

Since the data dependency of successive trellis stages makes intuitive pipelining technique difficult to insert registers within the ACS circuit, most researches improve the critical path by modifying the ACS circuit. The design in [24] shifts the normalization circuit, and the design in [25] applies the double state technique. However, it is a challenge to provide a stable clock signal with high frequency. Exploiting parallel trellis stage is another way to higher throughput of SISO decoder. The strategy uses the radix-2^{P_T} structure to deal with P_T consecutive symbols per cycle at the expense of hardware.

C. Operating Efficiency

The practical SISO decoder usually applies the sliding window method for less storage and shorter latency [26]. This method does a dummy β_d calculation to provide reliable initialization for β of each window. Fig. 4 shows the typical architecture, which consists of branch metric units, ACS units, LLR unit, and buffers. The input data inside each window are transmitted in descending order rather than in ascending order, so the decoder requires less memory usage to buffer input and α [27]. Fig. 5(a) indicates when the main computations for each window is performed during one decoding round, where W_i stands for the data in the i-th window, and K is total window number. The tail-biting technique calculates the α in W_{K-1} first to provide more robust α initialization of W_0 [28], and then it computes β_d, α, β, and LLR in turn for every window, from W_0 to W_{K-1}. Each of the β_d, α, and β computations of a window takes t_W cycles. It needs several cycles to get the LLR, extrinsic information, and decision after calculating β. Here we assume the latency to get these decoding results and write them back to memory is t_W cycles, and one decoding round takes $\Delta T = (K+4) \times t_W$ cycles.

Fig. 5(b) shows the active periods of the constituent units. The β_d-ACS, β-ACS, and LLR unit operate for $K \times t_W$ cycles, whereas the α-ACS operates for $(K+1) \times t_W$ cycles. Table I lists their operating efficiencies derived from dividing the active period by total processing time. Since the LLR calculation involves making hard decisions, the efficiency of LLR unit is the F_E for throughput calculation. Table I also shows the operating efficiencies of $K = 128$ and $K = 4$. If we use single SISO decoder to process a large block with 128 windows, F_E is close to 100%. After exploiting 32 parallel SISO decoders, K is 4 for each sub-block, and F_E is reduced to about 50%. It is obvious that lower operating efficiency is a side effect of high parallelism.

III. Contention-Free Interleaver Design

The interleaver design in this paper originates from the inter-block permutation interleaver in [13], and now it takes both the hybrid parallelism and implementation issue into account. Actually, many contention-free interleaver can be regarded as a combination of the interleaving operation within one sub-block and the interchange operation among all sub-blocks. The practical design needs an address generator for each memory module to perform the intra-block permutation and an interconnection network to complete the parallel data transmission. If the original P_T successive data could be one-to-one mapped onto the P_T distinct groups after interleaving, the design can support parallel trellis stage level. The intra-block permutation is the key to achieve this objective. For parallel SISO decoder level, the interconnection could be either the hierarchical memory structures [21], [22] or the network topologies suitable for connecting multiple sources to multiple destinations [19], [20]. The inter-block permutation plays an important role in reducing the storage for buffering data and lessening cycles for transmitting data.

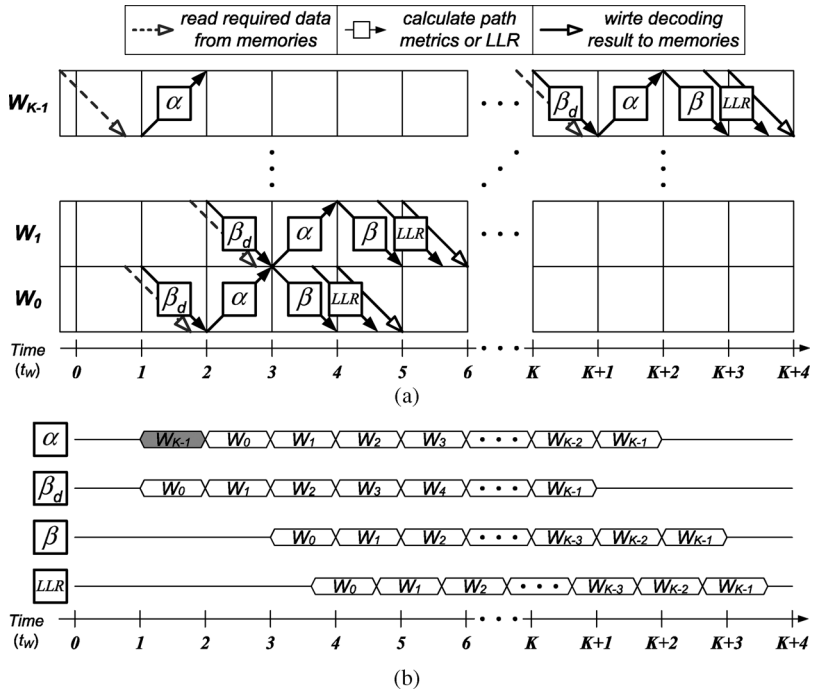


Fig. 5. Processes in the conventional architecture. (a) Processing schedule of one SISO decoder for every decoding round. (b) Corresponding active periods of main components.

TABLE I
OPERATING EFFICIENCY

Main Components of SISO Decoders	Active Period in Each Round	Efficiency		
		K	$K = 128$	$K = 4$
α-ACS	$(K+1) \times t_W$	$\frac{K+1}{K+4}$	97.7%	62.5%
β_d-ACS	$K \times t_W$	$\frac{K}{K+4}$	97.0%	50.0%
β-ACS				
LLR unit				

Before introducing the methodologies for the intra-block and inter-block permutations, the following example illustrates the interleaving steps. In the design with P_S size-N/P_S sub-blocks, the n-th symbol in the m-th sub-block is labeled as r_n^m. An example with one size-16 block and $P_S = 4$ is given in Fig. 6. Fig. 6(a) demonstrates that the first step is reordering the data in each sub-block from $(r_0^m, r_1^m, r_2^m, r_3^m)$ to $(r_2^m, r_3^m, r_0^m, r_1^m)$ for $m = 0 \sim 3$. The intra-block permutation must be constrained to make $\widetilde{r}_n^m$ and its nearby data from different banks so that the collision problem for $P_T \geq 2$ can be prevented. Fig 6(b) shows that the second step is swapping the symbols with the same index n with each other. Thus, it establishes a mapping between the original r_n^m's and the interleaved $\widetilde{r}_n^m$'s. Both the $\{r_n^0, r_n^1, r_n^2, r_n^3\}$ and $\{\widetilde{r}_n^0, \widetilde{r}_n^1, \widetilde{r}_n^2, \widetilde{r}_n^3\}$ can be accessed by four parallel SISO decoders concurrently without contention.

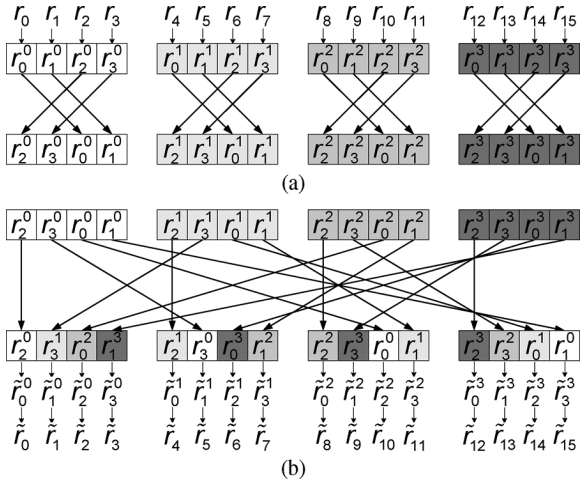


Fig. 6. An example of the contention-free interleaver with 4 sub-blocks. (a) Intra-block permutation. (b) Inter-block permutation.

A. Double Prime Method in Intra-Block Permutation

In intra-block permutation, only the sequence index n inside each sub-block is concerned. The prime interleaver $\pi(\cdot)$ in (6) whose factor ϵ must be relatively prime to N/P_S is one possible solution for parallel trellis stage level [29].

$$\pi(n) = (n \times \epsilon + \delta) \quad \left(\mathrm{mod} \frac{N}{P_S}\right). \tag{6}$$

The offset δ is another essential factor in randomness. By assigning appropriate parameters, the indexes n and $(n+i)$ can satisfy (7) after interleaving.

$$\pi(n) \not\equiv \pi(n+i) \pmod{P_T},\ 0 \leq i < P_T. \tag{7}$$

We could use padding bits to let the sub-block length be a multiple of P_T, and the desired parameters for (7) could be determined easily. For example, the ϵ will be an odd number with $P_T = 2$, and the $\pi(n)$ and $\pi(n+1)$ are not congruent modulo 2. Based on the prime interleaver, an alternative method that supports higher P_T but has fewer constraints is given in (8).

$$\pi(n) = \begin{cases} 2\left(\lfloor \frac{n}{2} \rfloor \times \epsilon\right) + 1 & \left(\mathrm{mod} \frac{N}{2P_S}\right), \quad n \text{ is odd} \\ 2\left(\lfloor \frac{n}{2} \rfloor \times \epsilon + \delta\right) & \left(\mathrm{mod} \frac{N}{2P_S}\right), \quad n \text{ is even} \end{cases} \tag{8}$$

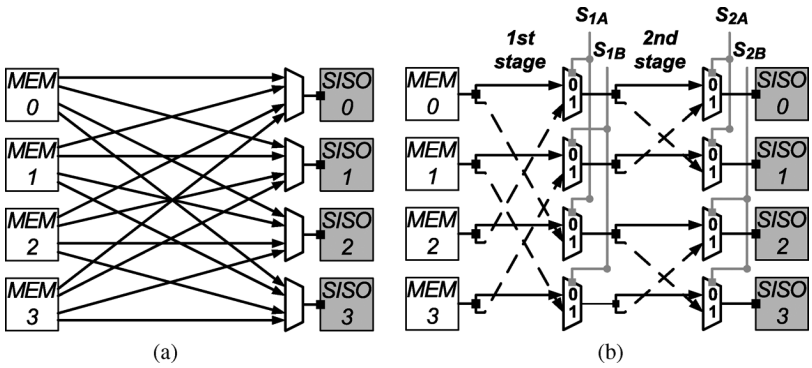


Fig. 7. Networks for connecting 4 SISO decoders and 4 memory modules. (a) Fully-connected network. (b) Butterfly network.

After partitioning each sub-block into two groups by the index n modulo 2, the odd-indexed data and the even-indexed data are permuted separately. The idea can be generalized to more partitions for various P_T's. The basic double prime method is suitable for our designs with $P_T = 2$ and $P_T = 4$. Both the interleaver and de-interleaver can be expressed in (8) with different parameters. The performance of such strategy with well-searched parameters is similar to the 3GPP turbo code [30].

B. Network-Oriented Approach in Inter-Block Permutation

Since the fully-connected network can provide any arbitrary interconnection, it is a trivial solution for parallel architectures. Fig. 7(a) shows the fully-connected network with $P_S = 4$. Such network can support various contention-free interleavers by assigning proper control signals to these P_S-to-1 multiplexers. However, it has some difficulties in implementing the network in high parallelism. As P_S increases, the area overhead of multiplexers increases rapidly, and the routing congestion becomes more severe. The network complexity depends on both the parallelism and the characteristic of interleaver. If we can design the inter-block permutation based on a simpler network, interconnecting patterns can be constrained so that the network complexity can be alleviated.

Our design exploits the multi-stage network that consists primarily of 2-to-1 multiplexers to transmit concurrent P_S sub-blocks. The multi-stage network must be constructed according to the following principle. The output port of every multiplexer or every memory module must connect to the first input port of one multiplexer and to the second input port of another multiplexer in next stage. We let the multiplexers with common input source share the same 1-bit control signal. Thus, each of the P_S data can be sent to next stage via exactly one of the two paths. The inter-block permutation will follow the behavior of the network, and a systematic interconnection will facilitate the implementation.

Fig. 7(b) demonstrates the multi-stage structure with $P_S = 4$, where the butterfly network topology is utilized. This network can swap m-th data with the $(m + 2^{((\log_2 P_S)-j)})$-th data in the j-th stage for $j = 1 \sim \log_2 P_S$. For example, the m-th data and the $(m + 2^0)$-th can be exchanged in the second stage of the $P_S = 4$ network. It gives a low-complexity solution without performance degradation [30]. The external control signals can be determined in advance and stored in a small look-up table.

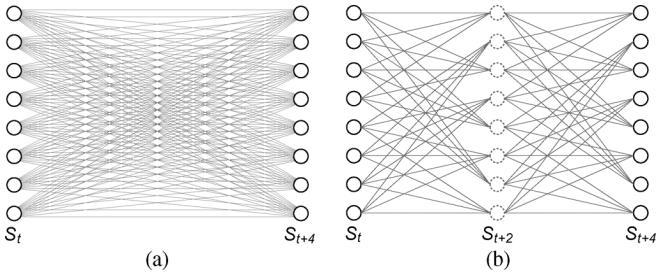


Fig. 8. (a) Radix-16 trellis diagram. (b) Radix-4 × 4 trellis diagram.

Using periodic assignment on these control signals can reduce the table size. With the help of this approach, the parallel design not only takes lower routing effort but also avoids complex control circuits.

IV. High-Speed ACS Circuits

Because high-radix ACS circuits for all α's and β's will contribute large hardware cost, a two-stage technique is proposed firstly to achieve high throughput with adequate area overhead; then a relocation technique is further applied to reduce the critical path delay. After applying these approaches, a high-throughput and area-efficient ACS architecture is derived.

A. Two-Stage Technique

The branch number per trellis stage determines the ACS execution amount of a SISO decoder. From a rough estimate of the area overhead, the original radix-2 trellis with N_S states has $2 \cdot N_S$ branches in each stage. After merging P_T successive stages to raise the throughput, the branch number in a radix-2^{P_T} trellis grows to $2^{P_T} \cdot N_S$. The two-stage technique decomposes the radix-2^{P_T} trellis into two radix-$2^{P_T/2}$, symbolized to radix-$2^{P_T/2} \times 2^{P_T/2}$, trellis stages. There are $2 \cdot 2^{P_T/2} \cdot N_S$ branches now. The two 8-state trellis diagrams in Fig. 8 are derived from 4 consecutive radix-2 trellis stages. The conventional radix-16 scheme has 128 branches in each stage, whereas the radix-4 × 4 scheme has 64 branches. Due to the decrease in branch number, the area of ACS unit will be lessened.

Fig. 9 shows the corresponding ACS structures in the radix-2^{P_T} and radix-$2^{P_T/2} \times 2^{P_T/2}$ trellis diagrams. These

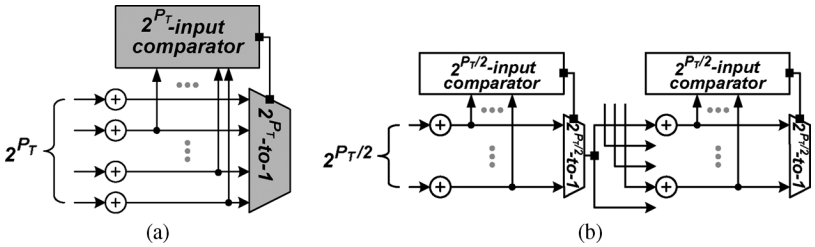


Fig. 9. (a) Radix-2^{P_T} ACS structure. (b) Radix-$2^{P_T/2} \times 2^{P_T/2}$ ACS structure.

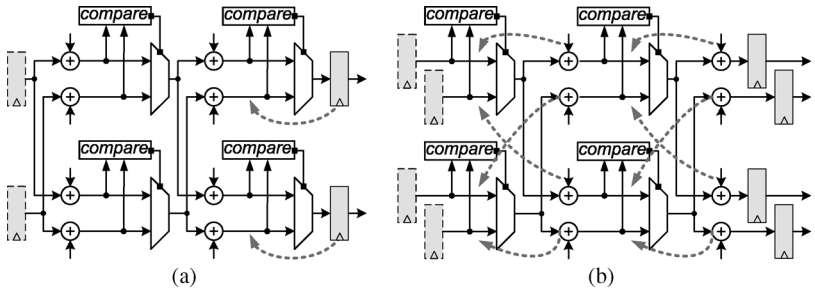


Fig. 10. Relocated procedure of radix-2 × 2 ACS structure. (a) Retiming of registers. (b) Relocation of adders.

TABLE II
AREA OF ACS UNITS WITH $P_T = 2$ AND $P_T = 4$

ACS unit	Area (gates count)	
	radix-2^{P_T}	radix-$2^{\frac{P_T}{2}} \times 2^{\frac{P_T}{2}}$
$P_T = 1$	3.2k	—
$P_T = 2$	11.9k	8.4k
$P_T = 4$	47.8k	21.3k

two high-radix structures have different amounts and types of circuits. The radix-2^{P_T} ACS unit consists of $2^{P_T} \cdot N_S$ adders, N_S comparators, and N_S multiplexers; both the comparators and multiplexers are 2^{P_T}-input components. The radix-$2^{P_T/2} \times 2^{P_T/2}$ ACS unit has $2 \cdot 2^{P_T/2} \cdot N_S$ adders, $2N_S$ comparators, and $2N_S$ multiplexers; the last two are $2^{P_T/2}$-input components. In general, one 2^{P_T}-input comparator (multiplexer) is larger than two $2^{P_T/2}$-input comparators (multiplexers). Thus, the ACS unit with two-stage technique occupies less area than the other one. Table II lists the synthesized area of various ACS units with 8 states and 9-bit metric quantization. Here we impose the same timing constraints on those units with the same P_T. In every ACS unit, the path metric registers requires about 0.4 k gates count. Only the combinational circuits are affected by P_T, and they dominates the area of ACS units. As the P_T increases, the difference between radix-2^{P_T} and radix-$2^{P_T/2} \times 2^{P_T/2}$ ACS units becomes more significant.

B. Relocation Technique

This two-stage method can save lots of hardware, but it will lead to longer critical path delay. The reason for inferior clock frequency is that the radix-$2^{P_T/2} \times 2^{P_T/2}$ ACS unit comprises two additions while the radix-2^{P_T} ACS unit comprises only one addition. For time-driven designs, the following technique can be applied to shorten the path delay.

The ACS circuit could not execute compare-select operations until additions are completed, so we have to break such data dependency to shorten the path delay. The relocation technique is utilized to eliminate the dependency by changing the hardware position. It can overcome the disadvantage of two-stage technique at the expense of extra hardware. An example of relocating a radix-2 × 2 ACS is illustrated in Fig. 10. The first step shown in Fig. 10(a) is retiming of registers. After moving and duplicating the registers ahead of the comparing circuits, the computation order is rearranged from ACS to compare-select-add (CSA). The second step shown in Fig. 10(b) is relocation of adders. By moving and duplicating the adders ahead of the multiplexers, the compare-select and addition could execute parallelly.

Fig. 11(a) shows the relocated radix-2 × 2 ACS circuit, and the critical path becomes two successive compare-select operations. Fig. 11(b) shows the variations of path delay among three high-radix architectures, where A, $C_{2^{P_T/2}}(C_{2^{P_T}})$, and $S_{2^{P_T/2}}(S_{2^{P_T}})$ stand for the execution time for addition, $2^{P_T/2}(2^{P_T})$-input comparison, and $2^{P_T/2}(2^{P_T})$-input selection circuits respectively. The relocated radix-$2^{P_T/2} \times 2^{P_T/2}$ structure has the shortest path delay among three structures, but the duplicated hardware makes its area larger than the original radix-$2^{P_T/2} \times 2^{P_T/2}$ structure. Although the area overhead due to relocation technique would diminish the advantage of two-stage technique, the combination of these two methods remains an area-efficient solution to achieve high throughput.

C. Modified ACS Operation

The original registers in ACS unit store the maximum among all summations of the previous path metrics and their corresponding branch metrics. Since the two-stage technique combines two successive path metric calculations and the relocation technique keeps all candidates rather than the maximum, some modifications are required to update these registers [31]. The

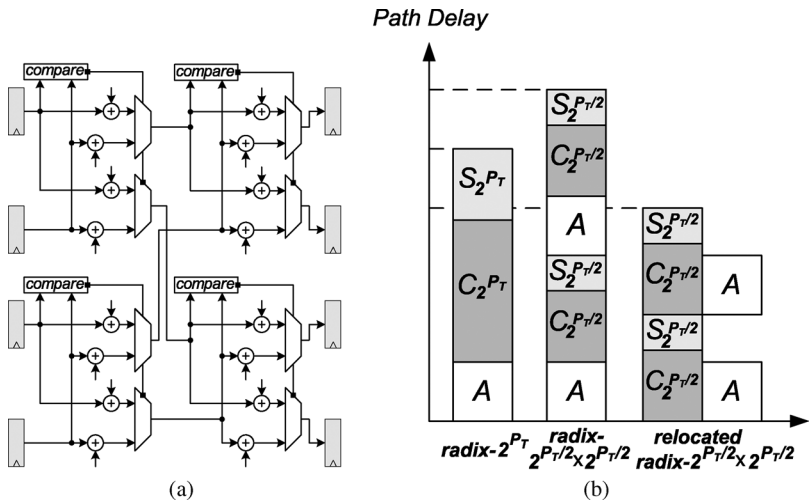


Fig. 11. (a) Relocated radix-2 × 2 ACS unit. (b) Path delay comparison.

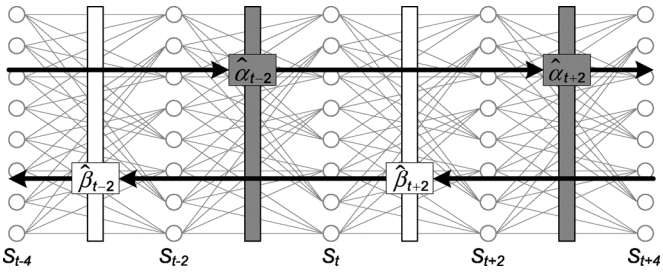


Fig. 12. Trellis diagram of relocated two-stage structure.

$\hat{\alpha}$ in (9) and $\hat{\beta}$ in (10) show the relations between modified and original path metrics. The log likelihood (LL) calculation in (11) is also affected by such adjustment.

$$\begin{aligned}\hat{\alpha}_{t-2}(\hat{s}_{t-2}) &= \alpha_{t-2}(s_{t-2}) + \gamma(s_{t-2}, s_t) \\ &= \max_{s_{t-4}} \{\alpha_{t-4}(s_{t-4}) + \gamma(s_{t-4}, s_{t-2})\} \\ &\quad + \gamma(s_{t-2}, s_t) \qquad (9)\\ \hat{\beta}_{t+2}(\hat{s}_{t+2}) &= \beta_{t+2}(s_{t+2}) + \gamma(s_t, s_{t+2}) \\ &= \max_{s_{t+4}} \{\beta_{t+4}(s_{t+4}) + \gamma(s_{t+2}, s_{t+4})\} \\ &\quad + \gamma_t(s_t, s_{t+1}) \qquad (10)\\ LL(u_{t-2}^{t+1}) &= \max_{s_{t-2} \to s_{t+2}} \{\alpha_{t-2}(s_{t-2}) + \gamma(s_{t-2}, s_t) \\ &\quad + \gamma(s_t, s_{t+2}) + \beta_{t+1}(s_{t+1})\} \qquad (11)\\ &= \max_{\widehat{S}} \left\{\hat{\alpha}_{t-2}(\hat{s}_{t-2}) + \hat{\beta}_{t+2}(\hat{s}_{t+2})\right\} \qquad (12)\end{aligned}$$

Note the operands in (11) are replaced with $\hat{\alpha}$ and $\hat{\beta}$, and $\widehat{S}$ is used to represent all possible summations of these two terms. Fig. 12 shows the corresponding change from the trellis in Fig. 3. Originally, the amount of registers for either forward or backward path metric is equivalent to the state number. The relocation technique will increase the register number. The vertical lines in Fig. 12 indicate the register locations of the relocated path metrics.

V. High-Efficiency Processing Schedule

The schedule in Fig. 5(a) indicates that it spends $(K+4)\times t_W$ cycles on K windows every decoding round. Since the duration of generating decisions is $K \times t_W$, the F_E is $K/(K+4)$. If the inactive cycles of the main components can be decreased, the F_E can be increased due to smaller denominator. Except the α-ACS, the idle period of any other functional unit takes $4 \times t_W$ cycles. The reason for such inconsistency is the dummy α calculation in the last window. Therefore, we initialize forward path metric with the boundary α's from previous iteration rather than the dummy calculation in current iteration [32]. It shortens the processing time at the expense of registers for previous α's. Any of the functional units spends $K \times t_W$ cycles on K windows, and one round takes $\Delta T = (K+3) \times t_W$ cycles. The F_E becomes $K/(K+3)$ now. The consistent process of each window will benefit the interlaced schedule, which will be introduced in next subsection to raise F_E up to 100%.

A. Interlaced Processing Schedule

Overlapping the normal round and the permuted round of each codeword is another way to achieve higher F_E, but it might cause performance loss due to unreliable *a priori* estimation. In order to keep the error-correcting capability, an alternative method that overlaps the processes of multiple codewords is applied. We let the SISO decoder decode several codewords concurrently by utilizing the idle processing time of each independent codeword. When the functional units finish the process at any decoding round of original codeword, these units will be occupied by the process of other codewords. The interlaced schedule can increase the F_E and is harmless to performance. Such concept can be regarded as the *pseudo parallel* turbo decoder level. The overhead is the storage for extra received codeword, extrinsic information, and decisions.

Fig. 13(a) illustrates the partial procedure based on the above-mentioned methods. Here one SISO decoder deals with codeword $\mathbf{A}$ and codeword $\mathbf{B}$, each of which comprises 4 windows. Fig. 13(b) gives the corresponding active periods of main functional units. All functional units are fully utilized so that F_E is 100%. Nevertheless, the interlaced schedule may extend

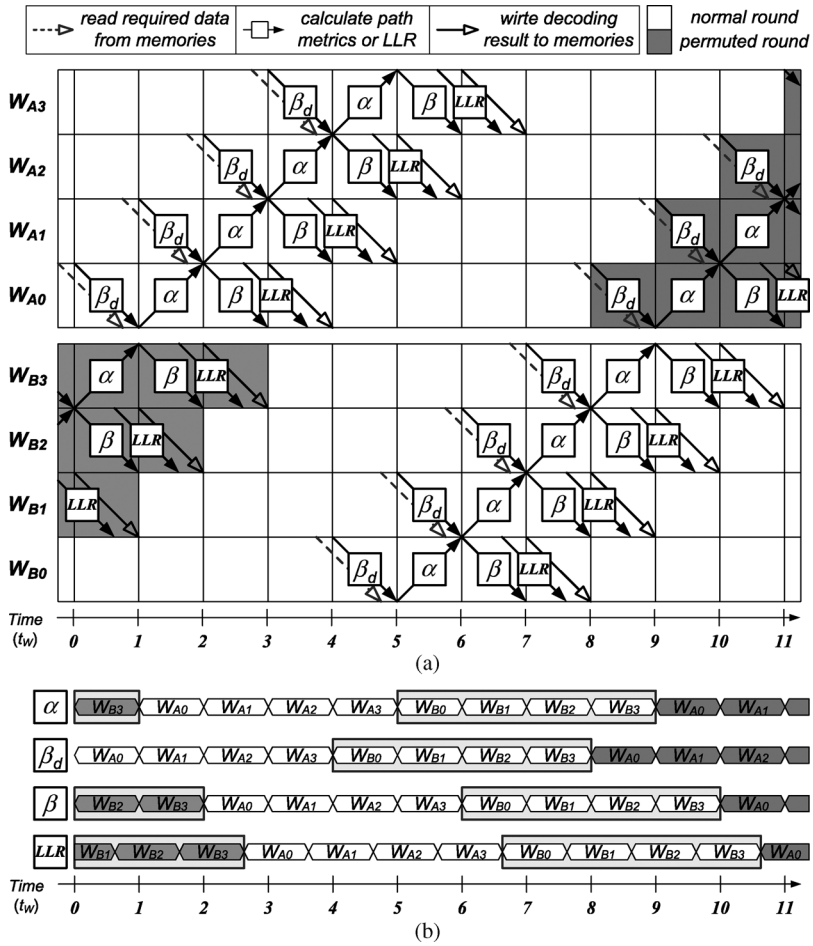


Fig. 13. Processes of independent codeword $\mathbf{A}$ and codeword $\mathbf{B}$. (a) Partial processing schedule of one SISO decoder. (b) Corresponding active periods of main components.

the necessary time for one round. As shown in Fig. 13(a), the permuted round of codeword $\mathbf{A}$ is delayed because the required components are occupied by codeword $\mathbf{B}$. The interlaced schedule for double codewords makes the ΔT extend to $8 \times t_W$ cycles. For $K = 4$, the throughput is increased due to higher F_E, but the overall latency $N_R \times \Delta T$ also increases.

B. Influence in the Small Sub-Block

Table III compares F_E and ΔT between the original schedule and the interlaced schedule with double codewords. The interlaced schedule is applied to improve F_E of small sub-blocks in parallel architecture, so this comparison considers window number $K = 1 \sim 8$ only. This table reveals that the data block size is an significant factor for F_E and ΔT in the proposed schedule. For $K < 3$, the F_E's are still less than 100%, and the ΔT's remain the same. Interlacing more than two blocks with the same K is a possible solution for this condition. For $K = 3$, it has a excellent trade-off between the 100% F_E and unchanged ΔT. For $K > 3$, the F_E's in the proposed schedule are raised to 100% with growing ΔT. Interlacing one block whose $K > 3$ and one block whose $K < 3$ can achieve high F_E with fewer increment of ΔT. Although two successive codewords always have the same length, dividing each codeword into several sub-blocks with unequal size could support this method.

TABLE III
COMPARISON BETWEEN ORIGINAL AND PROPOSED SCHEDULES

	Original Schedule (Single Codeword)		Proposed Schedule (Double Codewords)	
K	E_F	ΔT	E_F	ΔT
1	25.0%	$4t_W$	50.0%	$4t_W$
2	40.0%	$5t_W$	75.0%	$5t_W$
3	50.0%	$6t_W$	100%	$6t_W$
4	57.1%	$7t_W$	100%	$8t_W$
5	62.5%	$8t_W$	100%	$10t_W$
6	66.7%	$9t_W$	100%	$12t_W$
7	70.0%	$10t_W$	100%	$14t_W$
8	72.7%	$11t_W$	100%	$16t_W$

VI. SIMULATION AND IMPLEMENTATION RESULTS

Table IV lists the specifications of our proposed turbo decoders with parallel architecture. Design-II is a modified version from Design-I. Both designs use the code polynomial in the 3GPP standard [3] and support up to 4096 block size. The sliding window length is set to 32, and the iteration number is fixed at 8. These data widths and interleaver parameters are determined via fixed point simulation. By translating the decimal

TABLE IV
SPECIFICATIONS OF PROPOSED TURBO DECODERS

	Design-I	Design-II
Code Polynomial	$[1\ \frac{1+D+D^3}{1+D^2+D^3}]$	
Code Rate	$1/2$	$1/3$
Block Size	$128, 256, 512,$ $1024, 2048, 4096$	$256, 512,$ $1024, 2048, 4096$
Window Length	32	
Iterations	8 (16 rounds)	
Data Width	Received Symbol: $6(3.3)$ Extrinsic Information: $6(4.2)$ Path Metric: $8(6.2)$	
Double Prime Method	$(\epsilon, \delta) = (15, 23)$	
Inter-Block Permutation (32 Periodic Sequences)	$08, 19, 12, 18, 17, 14, 29, 05,$ $10, 21, 06, 23, 03, 26, 20, 22,$ $30, 04, 25, 15, 00, 02, 11, 09,$ $28, 24, 31, 27, 01, 13, 16, 07.$	
P_S	32	16
P_T	2	4
F_E	50%	100%

inter-block permutation parameters into binary expressions, the control signals for the multi-stage butterfly network can be derived. All multiplexers in the same stage are controlled by the same 1-bit signal for less storage requirement. Moreover, we apply the Max-Log MAP algorithm with 0.75 scaling factor for extrinsic information. Due to their distinct methods and individual area constraints, other design factors of the two designs are different, including code rate, supportable block size, P_S, P_T, and F_E.

Design-I combines the parallel SISO decoder level and the parallel trellis stage level. This design is implemented with 0.13 μm technology, and it consists of 2.67 M logic gates. To reduce the memory size for received codeword and I/O pin number, it is made into a rate-1/2 code by puncturing. There are 32 radix-2^2 SISO decoders in this design, and each SISO decoder is responsible for one 128-bit sub-block. By forcing the most significant bits of inter-block permutation parameters to zero, it can support the following block sizes: 128, 256, 512, 1024, 2048, and 4096. This design utilizes the relocated radix-2 $\times$ 2 ACS units, and the clock rate is 265 MHz according to the post-layout simulation. The operating efficiency is 50% because of the conventional processing schedule. In the SISO decoder with $P_T = 2$, each main component spends 16 cycles processing one window, so each decoding round takes 8 $\times$ 16 cycles.

Design-II involves the hybrid of the SISO decoder level, the trellis stage level, and the pseudo turbo decoder level. This design consists of 2.66 M gates count while implemented with 90 nm technology. There are 16 radix-2^4 SISO decoders in this design, and each SISO decoder is responsible for one 256-bit sub-block. It can support the following block sizes: 256, 512, 1024, 2048, and 4096. All ACS units are modified to the radix-4 $\times$ 4 structure by two-stage technique. Due to the advanced technology, this design can operate at 250 MHz even without relocation technique. The interlaced schedule is applied so that the

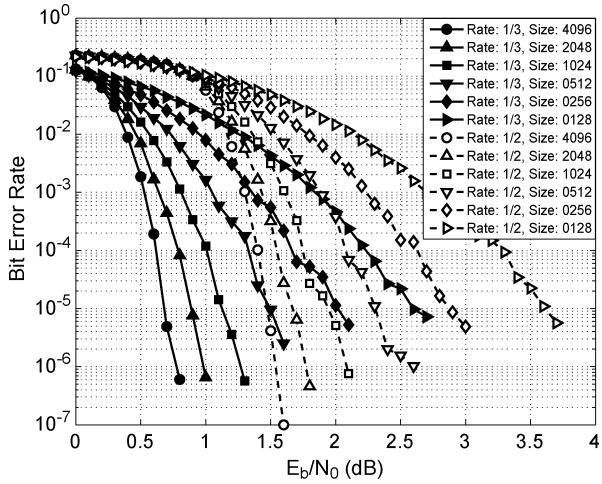
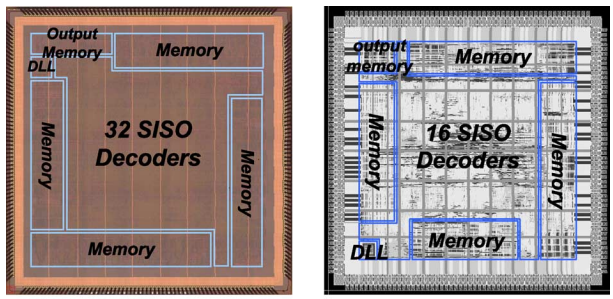


Fig. 14. Performance of the proposed turbo codes: rate-1/3 and rate-1/2; various block sizes; 8 states, 8 iterations, fixed point, and Max-Log MAP algorithm.



(a) (b)

Fig. 15. (a) Chip photo of the Design-I. (b) Layout plot of the Design-II.

operating efficiency achieves 100%. The total memory modules for double codewords cost about 920 k gates count. If the design uses original schedule for single codeword, the storage is 520 k gates count. In the SISO decoder with $P_T = 4$, each main component spends 8 cycles processing one window, so each decoding round takes 8 $\times$ 8 cycles.

Fig. 14 presents the corresponding bit error rate (BER) performance of the two proposed designs in additive white Gaussian noise (AWGN) channel. For the 4096-bit block, the performance loss caused by puncture is about 0.7 dB when BER is 10^{-5}. From (1) and the parameters in Table IV, the throughput of the two designs is $2F_C$ and $4F_C$ respectively. After post-layout simulation, Design-I is expected to achieve 530 Mb/s, and Design-II is expected to achieve 1000 Mb/s.

Fig. 15(a) shows the die photo of Design-I, and Fig. 15(b) shows the layout photo of Design-II. The parallel SISO decoders occupy the major area of both designs. A delay lock loop (DLL) circuit is used to generate internal clock source as four times the external frequency. However, we bypass the DLL mode of Design-I because of its malfunction. The measurement result indicates that the power consumption of Design-I is 275 mW with 1.32 V supply at 160 Mb/s, and the energy efficiency is 0.22 nJ/b/iter. Table V lists the comparison of the proposed decoders with three published works [33]–[35]. In the parallel architecture, the speedup in throughput will be larger than the

TABLE V
COMPARISON OF DIFFERENT DESIGNS

	Design-I	Design-II	[33]	[34]	[35]
Maximum Size	4096	4096	5114	384	2048
Iterations	8	8	6	4.43	5
Technology (μm)	0.13	0.09	0.18	0.18	0.13
Core Area (mm^2)	17.81	9.3	14.5	7.16	10
Frequency (MHz)	80 (265[a])	250[b]	145	160	352
Throughput (Mb/s)	160 (530[a])	1000[b]	24	71.7	352
Power (mW)	275 (508[a])	1158[b]	1450	N/A	2464
Energy Efficiency (nJ/b/iter)	0.22 (0.12[a])	0.15[b]	10.0	2.19	1.4

[a] Post-layout simulation at 1.2 V.
[b] Post-layout simulation at 1.0 V.

growth in power, implying that our designs achieve the best energy efficiency among available solutions.

VII. CONCLUSION

In this paper, a turbo decoder architecture utilizing both parallel SISO decoder level and parallel trellis stage level is presented. We introduce an interleaver that allows contention-free property in the hybrid parallelism. The interleaver design also alleviates the complexity of the interconnection network between multiple SISO decoders and memory modules. In addition, the two-stage and relocation techniques are exploited to raise the throughput of each SISO decoder with less area overhead. Finally, an interlaced processing schedule is proposed to improve the operating efficiency for small sub-block. The implementation results indicate the proposed designs have not only high throughput but also better energy efficiency.

ACKNOWLEDGMENT

The authors thank the Chip Implementation Center and UMC for for their support. They also appreciate Prof. Yu T. Su and Dr. Yan-Xiu Zheng for their assistance in inter-block permutation interleaver design.

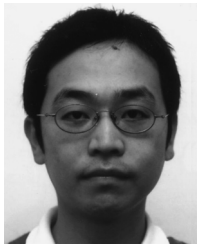
REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *IEEE Proc. Int. Conf. Communications*, May 1993, pp. 1064–1070.
[2] *Local and Metropolitan Area Network Part 16: Air Interface for Fixed Broadband Wireless Access Systems*, IEEE Std. 802.16e/D11, 2005.
[3] *Technical Specification Group Radio (3GPP) Access Network; Multiplexing and Channel Coding (FDD)*, 3GPP Std. TS 25.212, 2007, Rev. 8.0.0.
[4] K. Gracie and M.-H. Hamon, "Turbo and turbo-like codes: Principles and applications in telecommunications," *Proc. IEEE*, vol. 95, no. 6, pp. 1228–1254, Jun. 2007.
[5] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, no. IT-20, pp. 284–287, Mar. 1974.
[6] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal decoding algorithm," in *IEEE Proc. Int. Conf. Communications*, 1995, pp. 1009–1013.
[7] 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.
[8] R. Y. Shao, S. Lin, and P. C. Fossorier, "Two simple stopping criteria for turbo decoding," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1117–1120, Aug. 1999.
[9] O. Muller, A. Baghdadi, and M. Jezequel, "Exploring parallel processing levels for convolutional turbo decoding," in *2nd Information and Communication Technologies*, Apr. 2006, pp. 2353–2358.
[10] M. J. Thul, N. Wehn, and L. P. Rao, "Enabling high-speed turbo decoding through concurrent interleaving," in *IEEE Proc. Int. Symp. Circuits and Systems*, May 2002, pp. 26–29.
[11] A. Tarable, S. Benedetto, and G. Montorsi, "Mapping interleaving laws to parallel turbo and LDPC decoder architecture," *IEEE Trans. Inf. Theory*, vol. 50, no. 9, pp. 2002–2009, Sep. 2004.
[12] A. Giulietti, L. V. der Perre, and M. Strum, "Parallel turbo coding interleavers: Avoiding collisions in accesses to storage elements," *Elec. Lett.*, vol. 38, no. 5, pp. 232–234, Feb. 2002.
[13] Y. X. Zheng and Y. T. Su, "A new interleaver design and its application to turbo codes," in *Proc. IEEE Vehicular Technology Conf.*, Sep. 2002, vol. 3, pp. 1437–1441.
[14] D. Gnaedig, E. Boutillon, M. Jezequel, V. C. Gaudet, and P. G. Gulak, "On multiple slice turbo code," in *Proc. 3rd Int. Symp. Turbo Codes and Related Topics*, Sep. 2003, pp. 343–346.
[15] A. Abbasfar and K. Yao, "Interleaver design for high speed turbo decoders," in *IEEE Wireless Communications and Networking Conf.*, Mar. 2004, pp. 1611–1615.
[16] L. Dinoi and S. Benedetto, "Variable-size interleaver design for parallel turbo decoder architectures," in *IEEE Global Telecommunications Conf.*, Nov. 2004, pp. 3108–3112.
[17] Z. He, S. Roy, and P. Fortier, "High speed and low power design of parallel turbo decoder," in *IEEE Proc. Int. Symp. Circuits and Systems*, 2005, pp. 6018–6021.
[18] 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.
[19] G. Prescher, T. Gemmeke, and T. Noll, "A parameterizable low-power high-throughput turbo decoder," in *IEEE Int. Acoustics, Speech, and Signal Processing Conf.*, 2005, pp. V/25–V/28.
[20] H. Moussa, O. Muller, A. Baghdadi, and M. Jezequel, "Butterfly and bene-based on-chip communication networks for multiprocessor turbo decoding," in *Design, Automation and Test in Europe Conference and Exhibition*, Apr. 2007, pp. 1–6.
[21] R. Dobkin, M. Peleg, and R. Ginosar, "Parallel VLSI architecure for MAP turbo decoder," in *IEEE Int. Symp. Personal, Indoor and Mobile Radio Communications*, Sep. 2002, pp. 15–18.
[22] R. Dobkin, M. Peleg, and R. Ginosar, "Parallel interleaver design and VLSI architecture for low-latency MAP turbo decoders," *IEEE Trans. VLSI Syst.*, vol. 13, no. 4, pp. 427–438, Apr. 2005.
[23] D. Gnaedig, E. Boutillon, J. Tousch, and M. Jezequel, "Towards an optimal parallel decoding of turbo codes," in *Proc. 4th Int Symp. Turbo Codes Related Topics*, Apr. 2006.
[24] J. H. Han, A. T. Erdogan, and T. Arslan, "High speed Max-Log-MAP turbo SISO decoder implementation using branch metric normalization," in *IEEE Computer Society Annual Symp. VLSI*, 2005, pp. 173–178.
[25] I. Lee and J. L. Sonntag, "A new architecture for the fast Viterbi algorithm," *IEEE Trans. Commun.*, vol. 51, no. 10, pp. 1624–1628, Oct. 2003.
[26] S. A. Barbulescu, "Iterative Decoding of Turbo Codes and Other Concatenated Codes," Ph.D. dissertation, Univ. South Australia, , 1996.
[27] G. Masera, M. Mazza, G. Piccinini, F. Viglione, and M. Zamboni, "Architectural strategies for low-power VLSI turbo decoders," *IEEE Trans. VLSI Syst.*, vol. 10, no. 3, pp. 279–285, Jun. 2002.
[28] C. Weiß, C. Bettstetter, S. Riedel, and D. J. Costello, "Turbo decoding with tail-biting trellises," in *IEEE Proc. URSI Int. Symp. Signals, Systems, and Electronics*, Oct. 1998, pp. 343–348.
[29] S. Crozier, J. Lodge, P. Guinand, and A. Hunt, "Performance of turbo codes with relative prime and golden interleaving strategies," in *6th Int. Mobile Satellite Conf.*, Jun. 1999, pp. 268–275.
[30] C. C. Wong, C. H. Tang, M. W. Lai, Y. X. Zheng, C. C. Lin, H. C. Chang, C. Y. Lee, and Y. T. Su, "A 0.22 nJ/b/iter 0.13 μm turbo decoder chip using inter-block permutation interleaver," in *IEEE Custom Integrated Circuits Conference*, Sep. 2007, pp. 273–276.
[31] C. L. Chen, "High-Speed Viterbi Decoder Based on the Two-Dimensional ACS Structure," Master's thesis, National Chiao-Tung University, 2005.
[32] 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.

[33] M. Bickerstaff, L. Davis, C. Thomas, D. Garrett, and C. Nicol, "A 24 Mb/s radix-4 logMAP turbo decoder for 3GPP-HSDPA mobile wireless," in *IEEE Int. Solid-State Circuit Conf.*, Feb. 2003, pp. 151–484.
[34] B. Bougard, A. Giulietti, V. Derudder, J. Willem, S. Dupont, F. Catthoor, L. Hollevoet, L. V. der Perre, H. D. Man, and R. Lauwereins, "A scalable 8.7 nj/bit 75.6 Mb/s parallel concatenated convolutional (turbo-)codec," in *IEEE Int. Solid-State Circuit Conf.*, Feb. 2003, pp. 152–484.
[35] P. Urard, L. Paumier, M. Viollet, E. Lantreibecq, H. Michel, S. Muroor, and B. Gupta, "A generic 350 Mb/s turbo-codec based on a 16-states SISO decoder," in *IEEE Int. Solid-State Circuit Conf.*, Feb. 2004, pp. 424–536.

Cheng-Chi Wong received the B.S. degree in electrical engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2004. He is currently pursuing the Ph.D. degree in the Institute of Electronics, National Chiao Tung University.

His research interests include algorithm and VLSI architecture of error control codes and wireless communication system.



Ming-Wei Lai received the B.S. degree in electrical engineering from the National Tsing Hua University, Hsinchu, Taiwan, in 2002, and the M.S. degree in electronics engineering from the National Chiao Tung University, Hsinchu, Taiwan, in 2007.

He is currently working for MediaTek Corp., Hsinchu, Taiwan. His research interests include channel coding theory and architecture.

Chien-Ching Lin received the B.S. degree in electrical engineering from National Tsing Hua University, Hsinchu, Taiwan, in 2001, and the Ph.D. degree in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2006.

From 2007 to 2008, he was a postdoctoral researcher in the Department of Electronics Engineering at National Chiao Tung University. In February 2008, he joined Ambarella Taiwan Ltd. where he is an engineer working on the design of multimedia systems. His research interests include coding theory, VLSI architectures and integrated circuit design for communications and signal processing.

Hsie-Chia Chang received the B.S. and M.S. and the Ph.D. degrees in electronics 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 Corporation, 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.

Chen-Yi Lee (M'01) received the B.S. degree from National Chiao Tung University, Hsinchu, Taiwan, in 1982, and the M.S. and Ph.D. degrees from Katholieke University Leuven (KUL), Belgium, in 1986 and 1990, respectively, all in electrical engineering.

From 1986 to 1990, he was with IMEC/VSDM, working in the area of architecture synthesis for DSP. In February 1991, he joined the faculty of the Electronics Engineering Department, National Chiao Tung University, Hsinchu, Taiwan, where he is currently a Professor and Dean of Research and Development Office. His research interests mainly include VLSI algorithms and architectures for high-throughput DSP applications. He is also active in various aspects of high-speed networking, system-on-chip design technology, very low power designs, and multimedia signal processing. In these areas, he has published more than 180 papers and holds decades of patents.

Dr. Lee served as the Director of Chip Implementation Center (CIC), an organization for IC design promotion in Taiwan (2000–2003), and the microelectronics program coordinator of Engineering Division under National Science Council of Taiwan (2003–2005). He was the former IEEE CAS Taipei Chapter Chair.