International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.3, May-June 2012 pp-716-721 ISSN: 2249-6645

Design of Efficient Parallel Interleaver through OPMM (Optimized Memory Address remapping)

E. Priya
ME-VLSI Design,
Dept of ECE Easwari Engineering College,
Chennai, India.

S. Sudha
Dept of ECE
Easwari Engineering College
Chennai, India.

Abstract: **This work presents some mathematical models and collision free exchange rules for a parallel interleaver, using an optimized memory addresses remapping (OPMM) method that enables a classic interleaver to be exchanged for a parallel interleaver with efficiently. Both analytic and experimental results demonstrate that the rate of annealing and collision free results achieved using the OPMM approach is much faster than that achieved using the traditional Memory Address Remapping (MM) method.**

Index Terms: **collision-free, memory address remapping, parallel interleaver, turbo decoder** .

I. INTRODUCTION

INTERLEAVING is scrambling the processing order of the data inside a block to break up neighborhood-relations. Interleaving is a process of rearranging the ordering of a symbol sequence. The interleaver in turbo decoder is used to permute the input symbols such that the constituent decoders are operating on the same set of input symbols, but in interleaved (permuted) order. It is used in many channel coding schemes and also essential for the communications performance of turbo-codes. It is an advanced technique used by high-end motherboards /chipsets to improve memory performance [1].

Interleaving is a process of rearranging the ordering of a symbol sequence. The interleavers is a one to one mapping function that maps a sequences of t bits into another sequences of t bit. Interleavers are widely used for a vast range of communications applications. In Fig.1, M SISOs in SISO1s write symbols to memory modules named MEM_1, MEM_2, … MEM_M; since each memory module is written only once at a time, no collision exists. Similarly SISOs in SISO2s read symbols from memory modules MEM_1, MEM_2,…MEM_M; Since memory module MEM_1 is read two times at a time, collision happens.

The problem of collision is currently being solved in several ways. Memory arbitration technique was presented in [2]-[4]. To avoid losing clock cycles using online generated parallel interleavers were proposed in [5]-[7].

These patterns is based on the parallel level *M*, making it particularly difficult to optimize and performance differences for the same turbo decoder but with different numbers of parallel blocks. MM is an effective way to solve the problem. This approach preserves the interleaving pattern, so there is no performance loss both in error-correcting quality and decoding time.

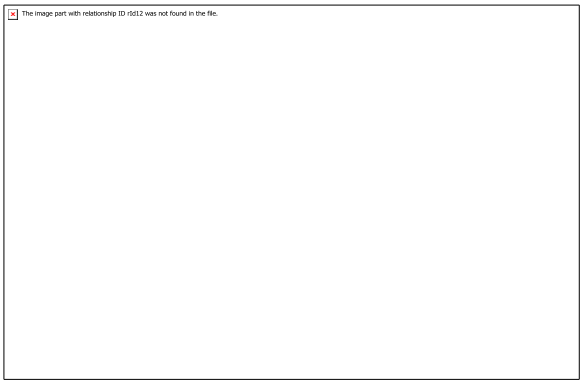


Fig. 1.Problem of collision.

The remainder of this paper is organized as follows. Section II introduces the terminologies and definitions used in this paper. Section III defines and mathematical models in parallel interleaving. Section IV explains collision-free exchange rules and the proposed OPMM method. Section V presents an example. Section VI presents experimental results. Section VII concludes the work.

II. TERMINOLOGIES AND NOTATIONS

The notations used in this papers are,

L Length of symbol sequence.

M Parallel level of a turbo decoder, is a divisor of *L*. ***N*** Word length of a memory module, .***N=L/M.***

i Matrix row index, $i \varepsilon [1,2,\dots N]$,i_1 and i_2 are instances of i .

j Matrix column index,$j \varepsilon [1,2,\dots,M]$,*j1* and *j2* are instances of j.

(i,j) Matrix index with value equivalent to $(i-1)M+j$.

g Memory module or SISO module number, $g \varepsilon [1,2,\dots,M]$, *g1* and *g2* are instances of g .

t Memory access time,$t \varepsilon [1,2,\dots N]$,*t1* and*t2* are instances of t .

A. Symbol Index

The symbol index of either SISO1s or SISO2s is defined as $(i-1)M+j$, where i denotes output or input time, and j is the number of the SISO module in SISO1s (or SISO2s). For instance, the symbol output from SISO1_2 at time 5 has the Symbol index $(5-1)M+2$, and symbol

input to SISO2_2 at time 5 also has the symbol index $(5-1)M+2$.

B. Memory Address

The memory address of *M* memory modules is defined as$(i-1)M+g$, where *i* denotes the position of the memory cell, g is the number of the memory module. For example, the 5's position of MEM_3 has the address$(i-1)M+3$.

C. Time Tile

The time tile represents a time, and defines all of the memory write/read operations of all the memory modules at that time.

D. Memory Tile

The memory tile represents a number of a memory module, and defines memory write/read operations of that memory module at all times.

E. Collision

In a time tile, if more than one attempt is made to access the memory module g, then a g-collision happens. The g –collision number equals (number of attempted-accesses -1).

F. Missing

In a time tile, if memory module g is not accessed, then a g-miss occurs.

III. PARALLEL INTERLEAVING

A Parallel Interleaver, in Fig. 2, PI plays a key role in the performance of the parallel decoder. It comprises the first interleaving stage (FIS) and the second interleaving stage (SIS). The FIS, which can hold up to *m*×*d* metrics, permutes the *m* metrics coming simultaneously from m SISOs (FIS depth *d* is termed also as FIS delay in the following).

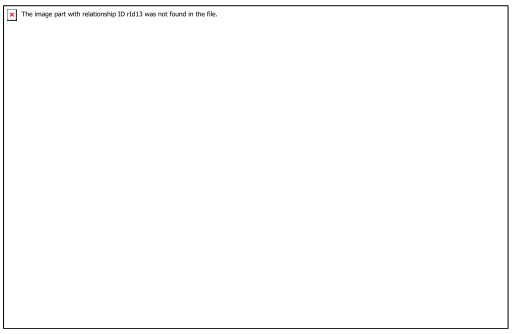


Fig. 2.Parallel interleaver

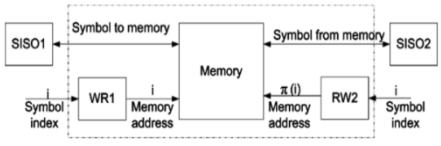


Fig.3.A Turbo decoder

A. Mathematical Model in Parallel Interleaving

The function of the interleaver defines how symbols from SISO1s are written to memory modules and how symbols that are input to SISO2s are read from memory modules.

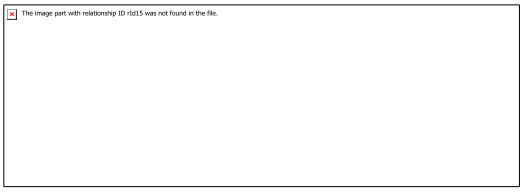
1) Write Matrix: $w\_s^T_{N\times M}$ describes how symbols from SISO1s are stored in memory modules.

In write matrix $w\_s^T_{N\times M}$, *(i,j)* represent memory address;$w\_s(i,j)$ represents symbol index, $w^t(i,j)$ represents the time tile of $w\_s(i,j)$.

$w^t(i,j)$ is derived from $w_{s(i,j)}$, $w^t(i,j)$ is the integer quotient of $(w_{s(i,j)}-1)/M+1$.

Combine $w\_s(i,j)$ and $w^t(i,j)$, a matrix element called $w\_s^t(i,j)$ can be created.$W\_S^T_{N\times M}$ is represented as

In write matrix $w\_s^T_{N\times M}$,column *j* is memory tile *j*, $w^t(i,j)$.



2) Read Matrix: $R\_A^G_{N\times M}$ describes hoe symbols are read to SISOs from memory modules.

In read matrix $R\_A^G_{N\times M}$,*(i,j)*represents symbol index,$r\_a(i,j)$ represents memory address ;$r^g(i,j)$ represents the memory tile of $r\_a(i,j)$, $r^g(i,j)$ is the integer reminder of $r_{a(i,j)}/N$. Combine $r\_a(i,j)$ and $r^g(i,j)$, a matrix element called $r\_a^g(i,j)$ can be created. Read matrix $R\_A^G_{N\times M}$ is presented as

$$R\_A^G_{N\times M}=\begin{vmatrix} r\_a^g(1,1) & , & r\_a^g(1,j) & , & r\_a^g(1,M) \\ r\_a^g(2,1) & , & r\_a^g(2,j) & , & r\_a^g(2,M) \\ , & , & , & , & , \\ r\_a^g(i,1) & , & r\_a^g(i,j) & , & r\_a^g(i,M) \\ , & , & , & , & , \\ r\_a^g(N,1) & , & r\_a^g(N,j) & , & r\_a^g(N,M) \end{vmatrix}$$

In read matrix $R\_A^G_{N\times M}$, row *i* is time tile *i*, $r^g(i,j)$ is memory tile $r^g(i,j)$.

B. Terminologies Used in OPMM Method

1) Memory Element Exchange Pair: $(g1|t1,g2|t2)$ exchange memory elements *g1*and *g2* between two rows (time tiles) *t1* and *t2* in read matrix.

2) Time Element Exchange Pair:$(t1|g1,t2|g2)$ exchange time elements *t1*and *t2*between two columns (memory tiles)*g1* and *g2* in write matrix.

3) Selected Memory Element Exchange pair:
$(g1|t1,g2|t2)$ exchange memory elements and between

two rows (time tiles) *t1*and *t2* to reduce one or two collisions in read matrix.

4) Selected Address Exchange Pair: $(r\_a^g(i1,j1), r\_a^g(i2,j2))$ exchange two addresses in the read matrix to eliminate one or two collisions.

5) Updated Sequence Exchange Pair: $(w\_s^t(i_1,j_1), w\_s^t(j_1,j_2))$ exchanges two symbol sequences in write matrix to update the selected address exchange happened in read matrix.

6) Middle Sequence Exchange Pair: $(w\_s^t(i_{11},j_{11}), w\_s^t(i_{22},j_{22}))$ exchange two symbol sequences in write matrix to remove the collisions caused by the updated sequence exchange.

7) Updated Address Pair: $(r\_a^g(i_1^a,j_1^a), r\_a^g(i_2^a,j_2^a))$ exchange two addresses in read matrix to update the middle sequence exchange happened in write matrix.

IV. PROPOSED OPMM METHOD

Collision-free conditions applying to both write and read matrices can also be stated as follows: the *N p*ositions of each memory module are written and/or read at *N* different times.

A. Collision-Free Exchange Rules:

1) Exchange Rule 1: Each selected address exchange in read matrix must be updated in write matrix, and each middle sequence exchange in write matrix must update in read matrix. Each selected address exchange in read matrix should be updated in write matrix to ensure the interleaving pattern is maintained. An address exchange in read matrix is updated in write matrix by an updated symbol index exchange.

2) Exchange Rule 2: An updated sequence exchange in write matrix is collision-free if the two exchanged *M* symbol sequences are in same time tile or same column (memory tile).

In write matrix, because *N* symbol sequences in same time tile are written to memory modules at same time, and symbol sequences in same column (memory tile) are saved in same memory module, therefore any exchange that occurs in the same time tile or the same column (memory tile) is collision-free.

3) Exchange Rule 3: If an updated sequence exchange in write matrix is not in the same time tile or column (memory tile), then a middle sequence exchange in write matrix should be added to remove the collisions introduced by the updated sequence exchange.

4) Exchange Rule 4: An updated address exchange in read matrix is collision-free if two exchanged memory addresses are in same memory tile or row (time tile).

In read matrix, because *M* addresses in same row (time tile) are written to memory modules at same time, and *N* positions in same memory tile are saved in same memory module, therefore any exchange that occurs in the same row (time tile) or the same memory tile is collision-free.

B. OPMM Procedure

For a given interleaving pattern, the collision-free parallel interleaver with parallel level can be generated from the classic interleaver through the following steps.

1) Initialization: Generate write matrix $W\_S_{N,M}^T$ and read matrices$R_A_{N,M}^G$, based on given interleaving pattern.

2) Check Collision: Check whether read matrix meet constraint (4). If no collision exists, then finish, otherwise go to OPMM step.

3) OPMM: If a memory element exchange pair $(g1|t1, g2|t2)$ in read matrix satisfies the conditions that row (time tile) *t1* has *g1* -collision, *g2*-missing and row (time tile) *t2* has *g1*-collision, *g2*-missing, then this exchange is called a type 1 selected exchange. A type 1 selected exchange can remove two collisions. If a memory element exchange pair $(g1|t1, g2|t2)$ satisfies conditions that row (time tile) *t1* has *g1*-collision and row(time tile) *t2* has *g2*-collision, *g1*-missing, then this exchange is called type 2 selected exchange. Repeat exchange cycle: identify a selected memory element $(g1|t1, g2|t2)$ exchange pair in matrix; choose a selected address exchange pair; find its updated sequence exchange pair in the write matrix. Since the updated sequence exchange pair does not meet exchange rule 2, a middle sequence exchange pair is added to the write matrix.

4) End Condition: Repeat OPMM until read matrix is collision-free. Then new collision-free write and read matrices are acquired, based on them the content of writing and reading ROMs, e.g., input and output signals of block and *WR1_1,WR1_2,…WR1_M* and *WR2_1,WR2_2,…WR2_M* can then be easily created.

V. EXAMPLES

In this section, an example is used to demonstrate how the exchanges rules are used in the process of OPMM.

Example: an interleaving pattern with parallel level *M=3*

$$G=\begin{pmatrix}1 & 7 & 4\\ 3 & 5 & 6\\ 2 & 8 & 9\end{pmatrix}$$

In interleaving matrix G, matrix index is the sequence number of write process and matrix value is the sequence number of read process.

A. Initialization

After initialization, the original write matrix $W\_S^T_{3\times3}$ and read matrix $R\_A^G_{3\times3}$ are

$$W\_S^T_{3\times3}=\begin{pmatrix}1^1 & 2^1 & 3^1\\ 4^2 & 5^2 & 6^2\\ 7^3 & 8^3 & 9^3\end{pmatrix}$$

$$R\_A^G_{3\times3}=\begin{pmatrix}1^1 & 7^1 & 4^1\\ 3^3 & 5^2 & 6^3\\ 2^2 & 8^2 & 9^3\end{pmatrix}$$

B. Check Collision

Check read matrix for collision. In this example, all three rows have collisions.

C. Exchange Cycle 0

According to exchange rule 2, no collisions will be introduced. Before exchange

$$W\_S^T_{3\times3}=\begin{pmatrix}1^1 & (2^1) & (3^1)\\ 4^2 & 5^2 & (6^2)\\ 7^3 & (8^3) & 9^3\end{pmatrix}$$

$$R\_A^G_{3\times3}=\begin{pmatrix}1^1 & 7^1 & 4^1\\ (3^3) & 5^2 & (6^3)\\ (2^2) & (8^2) & 9^3\end{pmatrix}$$

After exchange

$$W\_S^T_{3\times3}=\begin{pmatrix}1^1 & 2^1 & 3^1\\ 4^2 & 5^2 & 6^2\\ 7^3 & 8^3 & 9^3\end{pmatrix}$$

$$R\_A^G_{3\times3}=\begin{pmatrix}1^1 & 7^1 & 4^1\\ 2^2 & 5^2 & 8^2\\ 3^3 & 6^3 & 9^3\end{pmatrix}$$

D. Exchange Cycle 1

According to exchange rule 2, no collisions will be introduced. Before exchange

$$W\_S^T_{3\times3}=\begin{pmatrix}1^1 & (2^1) & 3^1\\ 4^2 & 5^2 & 6^2\\ (7^3) & 8^3 & 9^3\end{pmatrix}$$

$$R\_A^G_{3\times3}=\begin{pmatrix}1^1 & (7^1) & 4^1\\ (2^2) & 5^2 & 8^2\\ 3^3 & 6^3 & 9^3\end{pmatrix}$$

After exchange

$$W\_S^T_{3\times3}=\begin{pmatrix}1^1 & 2^1 & 3^1\\ 4^2 & 5^2 & 6^2\\ 7^3 & 8^3 & 9^3\end{pmatrix}$$

$$R\_A^G_{3\times3}=\begin{pmatrix}1^1 & 2^2 & 4^1\\ 7^1 & 5^2 & 8^2\\ 3^3 & 6^3 & 9^3\end{pmatrix}$$

E. Exchange Cycle 2

According to exchange rule 4, no new collisions are introduced.

$$W\_S^T_{3\times3}=\begin{pmatrix}1^1 & 2^1 & ((3^1))\\ ((4^2)) & 5^2 & 6^2\\ 7^3 & ((8^3)) & ((9^3))\end{pmatrix}$$

$$R\_A^G_{3\times3}=\begin{pmatrix}1^1 & 2^2 & ((4^1))\\ 7^1 & 5^2 & ((8^2))\\ ((3^3)) & 6^3 & ((9^3))\end{pmatrix}$$

After exchange

$$W\_S^T_{3\times3}=\begin{pmatrix}1^1 & 2^1 & 3^1\\ 4^2 & 5^2 & 6^2\\ 7^3 & 8^3 & 9^3\end{pmatrix}$$

$$R\_A^G_{3\times3}=\begin{pmatrix}1^1 & 2^2 & 9^3\\ 7^1 & 5^2 & 7^3\\ 8^2 & 6^3 & 4^1\end{pmatrix}$$

The interleaving pattern derived from the remapped write and read matrices are as follows:

$$G^{OPMM}=\begin{pmatrix}1 & 7 & 4\\ 3 & 5 & 6\\ 2 & 8 & 9\end{pmatrix}$$

The matrix G^{OPMM} is confirmed to maintain the original interleaving pattern.

VI. EXPERIMENTS

The OPMM scheme accelerates the remapping process in two ways: The proposed OPMM presents the collision free output

$$\mathrm{G}=\begin{pmatrix}5^2 & 2^2 & 9^3\\ 7^1 & 1^1 & 8^2\\ 3^3 & 6^3 & 4^1\end{pmatrix}$$

With the 4 exchange cycles. These collision full read matrix is given to the MM method and got 9 exchange cycle. When compare with MM Method time period also increased from 2.627ns to2.644ns.

1) Optimizing the selected address exchange pairs to maximize the number of OPMM steps be finished in a single remapping cycle.

2) In repeat remapping cycles, all the selected address exchange pairs are varied in two memory tiles instead *M* memory tiles, thus shortening the number of remapping cycles. Experimental results indicate that the OPMM method has a much shorter CPU calculation time than the traditional MM method.

VII. RESULTS

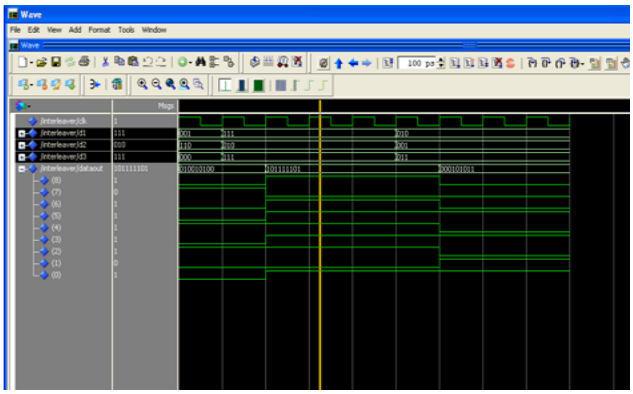


Fig 4.Interleaved Order

In Fig 4,it shows the interleaved order of the input values. MM and OPMM Method contains the Swap logic and Inter Leaving Methods used. These Modules are shown in Fig 5,6 and Fig 7.

Fig 8,9 shows MM Method and OPMM method.

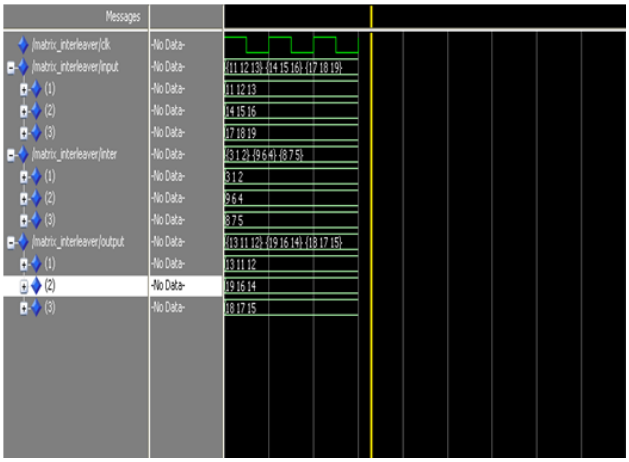


Fig 5.Interleaved Matrix

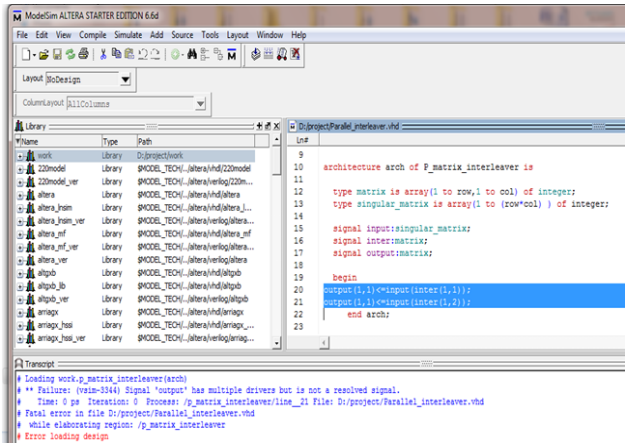


Fig 6.Memory Collision

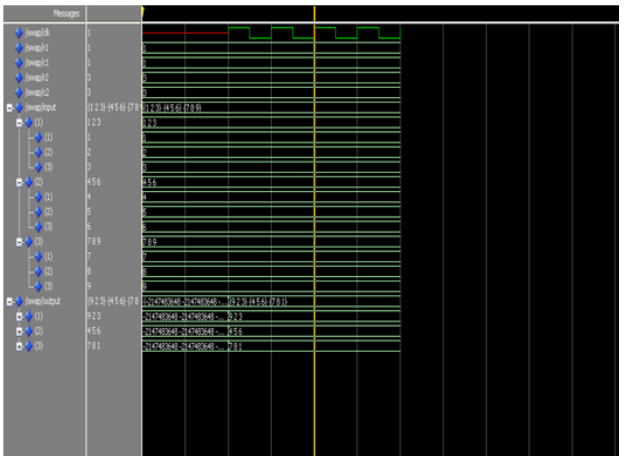


Fig 7.Swap Process

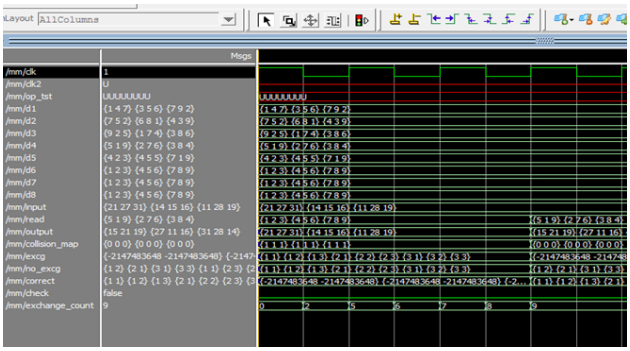


Fig 8.MM Method

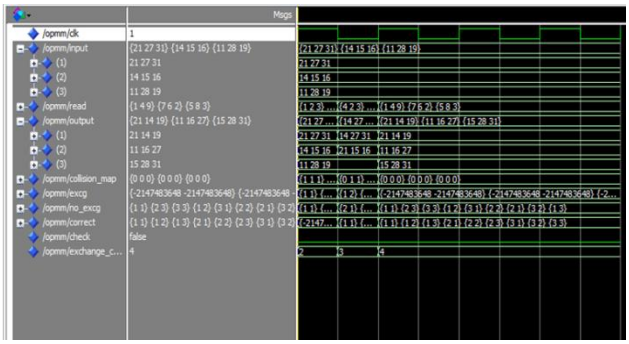


Fig 9.OPMM Method

REFERENCES

[1] Jing-ling Yang,, " Parallel Interleavers Through Optimized Memory Address Remapping," in *Proc. IEEE Int. Conf.* VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS., VOL. 18, NO. 6, JUNE 2010.

[2] M. J. Thul, F. Gibert, and N.Wehn, "Optimized current interleaving architecture for high-throughput turbo-decoding," in *Proc. 9th Int. Conf. Electron., Circuits Syst.*, Sep. 2002, vol. 3, pp. 1099–1102.

[3] M. J. Thul, F. Gilbert, and N.Wehn, "Concurrent interleaving architectures for high-throughput channels coding," in *Proc. IEEE Int. Conf.Acoust., Speech, Signal Process.*, Apr. 2003, vol. 2, pp. II–613–II-616.

[4] Z. Wang and K. Parhi, "Efficient interleaver memory architectures for serial turbo decoding," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Apr. 2003, vol. 2, pp. II-629–II-632.

[5] A. Giulietti, L. v. d. Perre, and A. Strum, "Parallel turbo coding interleavers: Avoiding collisions in accesses to storage elements," *Electron.Lett.*, vol. 38, no. 5, pp. 232–234, Feb. 2002.

[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. 427–437, Apr. 2005.

[7] Z. He, P. Fortier, and S. Roy, "Highly-parallel decoding architectures for convolutional turbo codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 11, pp. 1147–1151, Oct. 2006.

[8] A. Tarable and S. Benedetto, "Mapping interleaving laws to parallel turbo decoder architectures," *IEEE Commun. Lett.*, vol. 8, no. 3, pp. 162–164, Mar. 2004.

[9] A. Tarable and S. Benedetto, "Mapping interleaving laws to parallel turbo and LDPC decoder architectures," *IEEE Trans. Inf. Theory*, vol. 50, no. 9, pp. 2002–2009, Sep. 2004.