POLITECNICO DI TORINO
Repository ISTITUZIONALE

Hardware Design of a Low Complexity, Parallel Interleaver for WiMax Duo-Binary Turbo Decoding

Terms of use:

(Article begins on next page)

20 May 2024

Hardware design of a low complexity, parallel interleaver for WiMax duo-binary turbo decoding

Maurizio Martina, *Member IEEE*, Mario Nicola, Guido Masera, *Senior Member IEEE*

Abstract—A low complexity, parallel, collision-free interleaver architecture for the WiMax duo-binary turbo decoder is presented. The proposed architecture dynamically adapts to different block sizes and it features reduced complexity resorting to parallel circular shifting interleavers. Moreover, it sustains a peak throughput of nearly 90 Mb/s with a 200 MHz clock frequency, when synthesized on a 0.13 μm standard cell technology.

Index Terms—Parallel interleaver, hardware, VLSI

I. INTRODUCTION

Turbo codes are employed in several standards for wireless communications, such as WiMax [1]. When high transmission throughputs are required, parallel decoder architectures are needed to meet application speed constraints while keeping the clock frequency limited to few hundreds of MHz. A parallel turbo decoder is basically structured as M processing elements (PE) and M memories. Each PE plays the role of a Soft In Soft Out (SISO) module on a given window of data, whereas memories are used for exchanging extrinsic information among SISOs. Since WiMax resorts to a duo-binary turbo decoder [2], the encoder processes N_c couples of information bits $u = \{A, B\}$ with $A, B \in \{0, 1\}$, whereas the decoder works on N_c triplets of logarithmic likelihood ratios (LLRs) $\lambda[u] = (\lambda^{\overline{AB}}[u], \lambda^{A\overline{B}}[u], \lambda^{\overline{A}B}[u])$ where $\tilde{u} = \{\overline{A}, \overline{B}\}$ is taken as the reference symbol. The decoding process is iterative: in the in-order half iteration, the n-th SISO accesses only the n-th memory, whereas in the scrambled half iteration the n-th SISO reads from and writes to different memories. A collision occurs when two or more SISOs try to simultaneously access the same memory.

Parallel circular shifting interleavers are intrinsically collision-free [3] so they do not require further memory and logic to avoid collisions. In this letter, we propose a collision-free, low complexity, parallel architecture supporting the interleaving laws specified for the WiMax duo-binary turbo code [1]. Collisions are avoided by means of a variable parallelism architecture where M is chosen to grant that the resulting parallel interleaver is circular shifting. To the best of our knowledge this is the first work concerning the VLSI implementation of a collision-free parallel interleaver for the WiMax turbo decoder.

II. PROPOSED ARCHITECTURE

The permutation algorithm specified in [1] is structured in two steps. The first step switches $\lambda^{A\overline{B}}[u]$ and $\lambda^{\overline{A}B}[u]$ stored
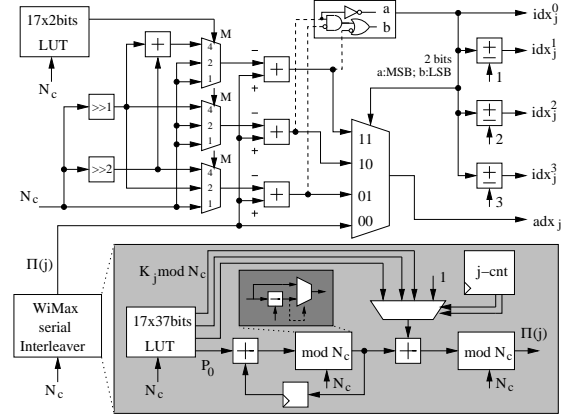
Figure 1.   WiMax parallel interleaver address generator architecture

at odd addresses leaving $\lambda^{\overline{A}B}[u]$ un-moved. The second step provides the interleaved address of the j-th triplet ($\lambda_j[u]$) as

$$\Pi(j) = (P_0 \cdot j + K_j) \bmod N_c \quad j = 0, 1, \ldots, N_c - 1 \quad (1)$$

where $K_j = 1$ when $j \bmod 4 = 0$, $K_j = 1 + N_c/2 + P_1$ when $j \bmod 4 = 1$, $K_j = 1 + P_2$ when $j \bmod 4 = 2$, $K_j = 1 + N_c/2 + P_3$ when $j \bmod 4 = 3$; P_0, P_1, P_2, P_3 are constants that depend only on the number of couples N_c [1]. Since the two steps can be swapped, the first step can be performed on the fly using $\Pi(j)$ least significant bit ($LSB[\Pi(j)]$) as a selector. The implementation of (1) can be derived as follows: if $x \in [0, 2 \cdot N_c - 1]$, $x \bmod N_c$ can be implemented by means of a subtracter and a multiplexer. Unfortunately, $P_0 \cdot j + K_j$ is not granted to belong to $[0, 2 \cdot N_c - 1]$. As a consequence, several $x \bmod N_c$ blocks ought to be cascaded to obtain $\Pi(j)$. However, (1) can be rewritten as

$$\Pi(j) = \{[(P_0 \cdot j) \bmod N_c] + (K_j \bmod N_c)\} \bmod N_c \quad (2)$$

Given that a small Look-Up-Table (LUT) is employed to store P_0 and the $K_j \bmod N_c$ terms, (2) can be implemented by two parts as depicted in Fig. 1 (shaded gray box). The first part accumulates P_0 to implement the $P_0 \cdot j$ term and the $\bmod N_c$ block produces the correct modulo N_c result. Since j is a counter ($j - cnt$ in Fig. 1), $(P_0 \cdot j) \bmod N_c$ is generated in one clock cycle adding P_0 to $[P_0 \cdot (j - 1)] \bmod N_c$ and performing the modulo operation. The second part employs the two least significant bits of the $j - cnt$ counter to select the proper $K_j \bmod N_c$ value, which is added to the $(P_0 \cdot j) \bmod N_c$ term. A further modulo N_c operation is performed at the output. Since in this architecture both the first and the second part work on data belonging to $[0, 2 \cdot N_c - 1]$, all the $\bmod N_c$ operations are implemented by means of a subtracter and a multiplexer (dark-shaded gray box in Fig. 1).

In the WiMax HUMAN-OFDM profile for 10 MHz channelization [1] the worst case downlink throughput is $\hat{T}_{dl} \simeq 65$ Mb/s. The decoder throughput can be estimated as the number of decoded bits ($2N_c$) over the time required to perform the decoding operations:

$$T = \frac{2N_c \cdot f_{clk}}{2I(\frac{N_c}{M} + SISO_l)} \qquad (3)$$

where $2I$ is the number of half iterations, f_{clk} is the clock frequency and $SISO_l$ is the SISO latency. We adopt a sliding window based approach where boundary metrics are inherited from one iteration to the next one as proposed in [4]. This allows to obtain $SISO_l = 2W$ (W is the window size). Assuming $W=32$ [5], $I=8$ and $f_{clk}=200$ MHz, we estimate the throughput of the decoder for the 17 possible values of N_c [1]. As shown in Fig. 2, $M=3$ allows to achieve $\hat{T}_{dl}$ (horizontal solid line) only for $N_c \geq 1440$, whereas with $M=4$ it can be reached for $N_c > 500$ (i.e. $N_c \geq 960$, the next specified size).
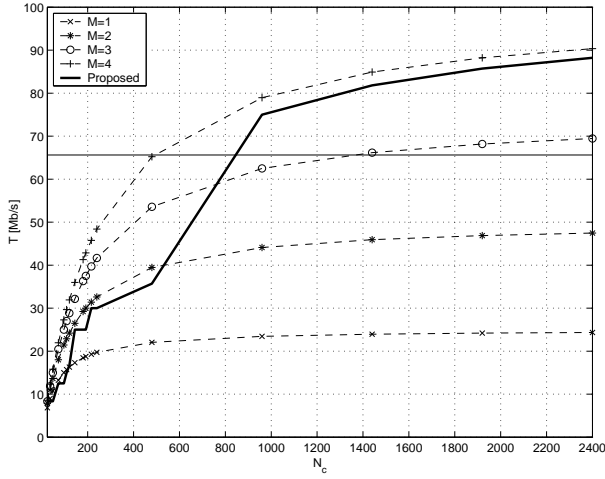

Figure 2. Parallel decoder throughput as a function of N_c for different parallelism degree values M ($W=32$ for dashed lines, W as in Table I for solid curve)

To satisfy the throughput requirement and to avoid high cost inter-SISO communication structures, a parallel collision-free interleaver is advisable. According to [3], a circular shifting interleaver is defined as $\Pi(j) = (a \cdot j + r) \bmod N$ where N is the block size, $r < N$ is an offset and $a < N$ is a step size that is relatively prime to N. Comparing this definition with (1), it is clear that the WiMax interleaver could be circular shifting with $a = P_0$, $r = K_j$ and $N = N_c$. As detailed in [6], parallel collision-free, circular shifting interleavers are obtained imposing

$$\Pi\left(j + k \cdot \frac{N_c}{M}\right) = \left[\Pi(j) \pm k \cdot \frac{N_c}{M}\right] \bmod N_c \qquad (4)$$

with $j = 0, 1, \ldots \frac{N_c}{M} - 1$ and $k = 0, 1, \ldots M-1$. Substituting (1) into (4) we obtain the conditions required to ensure that the WiMax interleaver is collision-free for a given parallelism degree M. Given that $\frac{N_c}{M} \in \mathbb{N}$, the first condition is defined by (5). Further conditions must be added depending on $\frac{N_c}{M}$ parity: if $\frac{N_c}{M} \bmod 4 \neq 0$ the conditions required are (5) and

(6). Finally, if also $\frac{N_c}{M} \bmod 2 = 1$, (5), (6) and (7) must be simultaneously satisfied:

$$(P_0 \mp 1) \bmod M = 0 \qquad (5)$$

$$P_2 \bmod N_c = 0 \qquad |P_3 - P_1| \bmod N_c = 0 \qquad (6)$$

$$F(P_1, P_3, |P_2 - P_1|, |P_3 - P_1|) = [0\ 0\ 0\ 0] \qquad (7)$$

where $F(x_0, \ldots, x_3) = [f(x_0) \ \ldots \ f(x_3)]$ with $f(x) = (2x + N_c) \bmod 2N_c$.

Let's introduce $\mathcal{I}$ as the set of the 17 possible N_c values specified by the WiMax standard [1]. Given $N_c \in \mathcal{I}$ and the corresponding P_0, P_1, P_2, P_3, we find which $M \in \{2, 3, 4\}$ grants to obtain parallel and collision-free interleavers. It is worth pointing out that all the possible P_0 specified in [1] satisfy (5) with $M \in \{2, 3, 4\}$. As a consequence, all the configurations where $\frac{N_c}{M} \bmod 4 = 0$ with $M \in \{2, 3, 4\}$ and $N_c \in \mathcal{I}$ correspond to a parallel collision-free interleaver. This is the case of $M=3$ that leads to parallel collision-free interleavers for every $N_c \in \mathcal{I}$.

When $M=2$, we have $\frac{N_c}{M} \bmod 4 \neq 0$ for $N_c \in \mathcal{I}' = \{36, 108, 180\}$. In this case $\frac{N_c}{M} \bmod 2 = 0$, so that only (6) must be checked. The only value $N_c \in \mathcal{I}'$ not satisfying (6) is $N_c=108$, which leads to collisions.

When $M=4$, we have $\frac{N_c}{M} \bmod 4 \neq 0$ for $N_c \in \mathcal{I}'' = \{24, 72, 120, 216\}$ and $N_c \in \mathcal{I}''' = \{36, 108, 180\}$, namely $\frac{N_c}{M} \bmod 2 = 0$ with $N_c \in \mathcal{I}''$ and $\frac{N_c}{M} \bmod 2 = 1$ with $N_c \in \mathcal{I}'''$. Since (6) is verified with $M=4$ and $N_c \in \mathcal{I}''$, these configurations lead to parallel collision-free interleavers. On the other hand, when $M=4$ and $N_c \in \mathcal{I}'''$ both (6) and (7) must be satisfied to obtain parallel collision-free interleavers. The only $N_c \in \mathcal{I}'''$ leading to collisions is $N_c=108$, that satisfies neither (6) nor (7).

In this work a parallel, collision-free interleaver is obtained selecting M as a function of N_c and in particular $M \neq 2$ and $M \neq 4$ when $N_c=108$. Since the resulting interleaver is a parallel circular shifting interleaver, we can write $\left[\Pi(j) \pm k \cdot \frac{N_c}{M}\right] \bmod N_c = \mathrm{idx}_j^k \cdot \frac{N_c}{M} + \mathrm{adx}_j$, where $\mathrm{adx}_j = \Pi(j) \bmod \frac{N_c}{M}$, $\mathrm{idx}_j^k = (\mathrm{idx}_j^0 \pm k) \bmod M$ and $\mathrm{idx}_j^0 = \Pi(j) - \mathrm{adx}_j$. This results in a simplified address generation with all SISOs simultaneously accessing the same location (adx_j) of different memories, where idx_j^k is the memory accessed by SISO-k at time j during a scrambled half iteration. Thus, the parallel interleaver is obtained cascading a simple block to the serial interleaver architecture. This block extracts adx_j and idx_j^k from the first N_c/M values of $\Pi(j)$:

$$\mathrm{adx}_j = \begin{cases} \Pi(j) & \text{if } \Pi(j) \in [0, \frac{N_c}{M} - 1] \\ \Pi(j) - \frac{N_c}{M} & \text{if } \Pi(j) \in [\frac{N_c}{M}, 2\frac{N_c}{M} - 1] \\ \ldots \\ \Pi(j) - (M-1)\frac{N_c}{M} & \text{if } \Pi(j) \in [(M-1)\frac{N_c}{M}, N_c - 1] \end{cases} \qquad (8)$$

whose straightforward implementation needs to calculate N_c/M and to allocate $M - 2$ multipliers, $M - 1$ subtracters, an M-ways multiplexer and few logic for selecting the proper adx_j value. The N_c/M division can be simplified by choosing the possible M values as powers of two. Thus, in the proposed throughput/parallelism scalable decoder, we employ: $M=1$ when $N_c \leq 108$, $M=2$ when $120 \leq N_c \leq 480$ and $M=4$

Table I
PARALLELISM DEGREE (M), WINDOW SIZE (W), NUMBER OF WINDOWS PER SISO (N_W^M) AND THROUGHPUT (T) IN MB/S ACHIEVED BY THE WIMAX TURBO DECODER PARALLEL ARCHITECTURE FOR THE 17 N_c VALUES

N_c	24	36	48	72	96	108	120	144	180
M	1	1	1	1	1	1	2	2	2
W	24	36	48	36	48	36	60	36	45
N_W^M	1	1	1	2	2	3	1	2	2
T	8.3	8.3	8.3	12.5	12.5	15	16.7	25	25

N_c	192	216	240	480	960	1440	1920	2400
M	2	2	2	2	4	4	4	4
W	48	36	40	48	40	40	40	40
N_W^M	2	3	3	5	6	9	12	15
T	25	30	30	35.7	75	81.8	85.7	88.2

Figure 3.   WiMax parallel turbo decoder architecture

when $960 \leq N_c \leq 2400$. As it can be inferred from Fig. 1, multiplications are avoided resorting to simple shift operations ($x >> i = x/2^i$). The sign of the subtractions (dashed lines in Fig. 1) allows not only to select the correct adx_j but also to find idx_j^0. Then, the idx_j^k values are generated with $M-1$ modulo M adders/subtracters. Moreover, the proposed decoder grants that N_c/M is even, so that $LSB[\text{adx}_j]$ is used to switch $\lambda^{A\overline{B}}[u]$ and $\lambda^{\overline{A}B}[u]$ stored at odd addresses (see the *switch couple* signal in Fig. 3). We also impose $N_c/(M \cdot W) \in \mathbb{N}$, which implies that each SISO processes the same number (N_W^M) of windows in a data frame. Of course, proper W and M values must be selected for each frame size, as reported in Table I, where N_W^M and the resulting throughput T, obtained from post synthesis simulations with Modelsim (bold line in Fig. 2), are also given. This solution requires the component decoder to support different sliding window sizes and N_W^M values. However, resorting to the SISO architecture detailed in [7], different W and N_W^M values are supported with a negligible complexity overhead using two programmable counters in the SISOs control unit.

III. RESULTS AND CONCLUSIONS

The architecture detailed in the previous paragraphs simultaneously produces M addresses per cycle and is employed to implement the interleaver reading part. Since idx_j^k identifies the memory accessed by SISO-k at time j, the parallel interleaver architecture ought to signal to the memory which SISO is requiring the data. This operation is accomplished by a 4×4 crossbar switch (radx-switch) controlled by idx_j^k with 2 bit wide fixed inputs, as shown in Fig. 3. When the idx^k memory (EI-MEM idx^k) is read, it sends back the corresponding $\lambda[u]$ triplet to SISO-k, through a 4×4 crossbar switch (rdata-switch). This crossbar switch is controlled by the output of the radx-switch.

Since each SISO outputs its data in reverse order, during the reading operation idx_j^k and adx_j are stored into a LIFO; idx_j^k and adx_j are read from the LIFO during the writing operation to configure a 4×4 crossbar switch (wdata-switch).

The proposed parallel interleaver has been described in VHDL and synthesized on a 0.13 μm standard cell technology with Synopsys Design Compiler. Power consumption values have been obtained with Modelsim and Synopsys Power Compiler. The 17×37 bits LUT, synthesized as combinational logic, require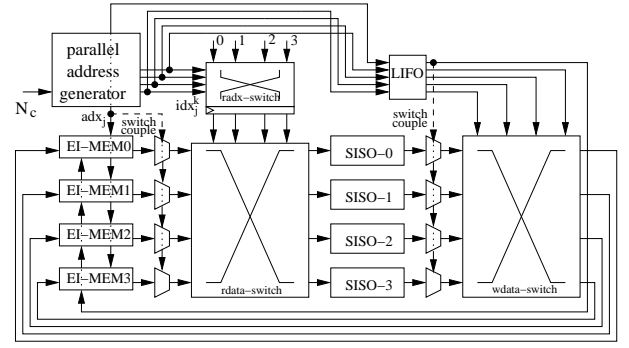s 330 equivalent gates. Thus, the serial interleaver is made of only 1340 equivalent gates and the total address generator needs 1474 equivalent gates, where 24 equivalent gates are used to implement the 17×2 bits LUT devoted to obtain M from N_c. Further 50 and 577 equivalent gates are required respectively by the radx-switch and the data-switch (either rdata-switch or wdata-switch), where 8 bits are employed for each LLR of the $\lambda[u]$ triplet. The complete parallel interleaver architecture needs about 2200 equivalent gates for the reading part and about 600 equivalent gates for the writing part, leading to a total complexity of about 2800 equivalent gates and an average power consumption of about 1.3 mW with a 200 MHz clock frequency (memories not included). Similar complexity results were obtained on FPGA: the complete parallel interleaver architecture requires 876 LUTs and 103 Flip-Flops for a 200 MHz clock frequency on a Xilinx Virtex 5 X5VLX30 with Xilinx Ise. The complete decoder requires 133.6 kbits [7] of memory, where 57.6 kbits are devoted to the EI-MEM and 2.2 kbits to the LIFO.

A parallel decoder made of M SISOs requires to simultaneously produce M addresses per cycle. Thus, for a fair comparison, we consider a parallel interleaver obtained using four instances of the single address per cycle interleaver in [5]: this solution requires about 4880 equivalent gates (1220 equivalent gates for each address generator) not including the switches and the memories, with an average power consumption of about 3.2 mW (0.8 mW for each address generator) at 200 MHz. As it can be observed the proposed interleaver, is more than 50% simpler than placing four instances of [5].

REFERENCES

[1] "IEEE Std 802.16, part 16: air interface for fixed broadband wireless access systems," Oct. 2004.
[2] C. Berrou et al. "The advantages of non-binary turbo codes," in *IEEE Inf. Theory Workshop*, 2001, pp. 61–63.
[3] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations," *TDA Progress Report*, vol. 42-122, pp. 56–65, Aug 1995.
[4] C. Zhan et al. "An efficient decoder scheme for double binary circular turbo codes," in *IEEE ICASSP*, 2006, pp. 229–232.
[5] J. H. Kim and I. C. Park, "Double-binary circular turbo decoding based on border metric encoding," *IEEE Trans. on Circuits and Systems II*, vol. 55, no. 1, pp. 79–83, Jan 2008.
[6] J. Kwak and K. Lee, "Design of dividable interleaver for parallel decoding in turbo codes," *IET Electronics Letters*, vol. 38, no. 22, pp. 1362–1364, Oct 2002.
[7] M. Martina, M. Nicola, and G. Masera, "A flexible UMTS-WiMax turbo decoder architecture," *IEEE Trans. on Circuits and Systems II*, vol. 55, no. 4, pp. 369–373, Apr 2008.