EFFICIENT INTERLEAVER MEMORY ARCHITECTURES FOR SERIAL TURBO DECODING

Zhongfeng Wang[1*] and Keshab Parhi[2]

1. Nation Semiconductor Corporation
1351 South Sunset Street
Longmont, CO 80501
Tel: 303-774-5316
Email: zwang@ia.nsc.com

2. Dept. of Electrical and Computer Engineering
University of Minnesota
Tel: 612-626-4116
Email: parhi@ece.umn.edu

ABSTRACT

A practical turbo decoder is usually implemented with a serial decoding architecture for low complexity, where the extrinsic information symbols are stored in the so-called interleaver memory for the next decoding. Either a dual-port (or two ping-pong memories) or a single-port memory can be employed for this memory. The first approach achieves twice the throughput as the second one while spending approximately twice the hardware on the interleaver memory. In this work, two novel architectures are proposed for the interleaver memory design. Both proposed architectures work for any type of random interleavers. Compared with the traditional single-port approach, twice the throughput can be obtained with less than 1% area overhead when applied in third generation CDMA systems. On the other hand, more than 25% area of an entire turbo decoder can be saved compared with the traditional dual-port solution.

Keywords: VLSI, serial architecture, low complexity, turbo code, interleaver, memory.

1. INTRODUCTION

Turbo code [1], since its invention, has found many applications. Especially, it has been decided to use turbo code in the third generation (3G) CDMA (code division multiple accesses) systems [2][3]. Turbo code usually works with large block sizes. It is generally true that the larger block size, the better the performance. For 3G CDMA systems, the maximum turbo block size is more than 20,000 (bits) [3]. Due to the requirement of large memory, serial decoding architectures are widely used in practice for turbo decoders, especially in wireless applications. For example, the turbo decoder products provided by Texas Instrument (C6416), Altera, Icoding, Small World Comm., Amphion, Adlante Technology, et al, all used serial decoding architectures. A typical serial turbo decoder architecture is shown in Figure 1. It has only one soft-input soft-output (SISO) decoder, which works in a time-multiplexed way. Each iteration consists of two decoding phases, i.e., *the sequential decoding phase (or simply Phase A)*, in which the data are processed in sequential order, and *the interleaved decoding phase (or Phase B)*, in which data are processed in an interleaved order. Both maximum *a posterior* probability (MAP) algorithm and soft output Viterbi algorithm (SOVA) can be adopted for the SISO decoder [5].

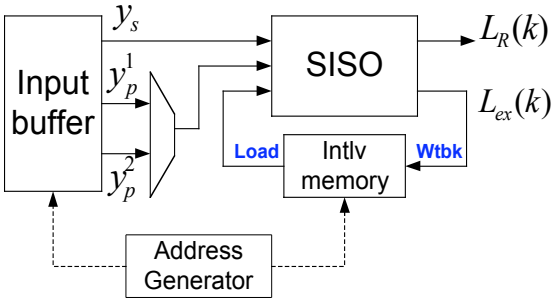

Figure 1, A serial turbo decoder architecture

As can been seen from Figure 1, a serial turbo decoder consists of two memories: one is used to store the received soft symbols, namely the input buffer, the other is used to store the extrinsic information [1], denoted as the inter-leaver memory. The SISO decoder takes soft inputs (including the received systematic bit y_s and the

* Partial work was completed when the author was with Morphics Technology Inc., California.

received parity bit y_p^1 or y_p^2) from the input buffer and the *a priori* information from the interleaver memory. It outputs the log likelihood ratio, $L_R(k)$, and the extrinsic information, $L_{ex}(k)$, for the k-th information bit in the decoding sequence. The extrinsic information is sent back as the new *a priori* information for next decoding. The interleaving and de-interleaving processes are completed in an efficient way. Basically the data are loaded according to the current decoding sequence. After processing, the data are written back to the original places. For example, in Phase B, we load the data from the interleaver memory in an interleaved order. After processing, we write them back to the original places in the same order.

The input buffer is generally indispensable. Regarding the interleaver memory, two approaches have been used in the past. With the first approach, a dual-port memory or two ping-pong memories are employed to complete one Read and one Write operations required to process each information bit within one cycle. In the second approach, one single-port memory is used while spending two clock cycles to complete the required two operations. Due to the less complexity, the latter approach is more popular in practice. In this work, two novel architectures are proposed for the interleaver memory design. Either proposed architecture can complete the two operations within one cycle while using almost the same hardware as the case of using one single-port memory. Compared with traditional designs, the proposed approaches can either save more than 25% area of an entire turbo decoder or achieve twice the throughput with less than 1% area overhead when applied in 3G CDMA systems.

2. EFFICIENT INTERLEAVER ARCHITECTURES

The basic idea is to partition one single-port memory into N ($N \geq 2$) segments and use M ($M \geq 2$) small buffers to assist writing the output of extrinsic information into the inter-leaver memory, where N does not have to be the same as M. For design simplicity, both N and M are normally chosen to be a power of 2. N is better chosen to be a multiple of M. In this work, we will consider only two cases (a) N=2, M=2 and (b) N=4, M=2.

Memory can be partitioned in various ways. An easy and effective way is to partition the memory according to a number of least significant bits (**lsb**'s) of the memory address. In case of N=2, a piece of memory can be partitioned into two segments according to the least significant bit. For the resultant two memory segments, one contains data with even addresses and the other contains data with odd addresses.

For the two memory segments, it is possible, in principle, to support two data accesses within one cycle. However, in practice, it is generally impossible to find a partitioning scheme which ensures that the target data (for Read and Write operations) are always located in two different segments at each cycle because of interleaving. In fact, practical systems are usually required to support a variety of turbo block sizes. In those cases, the interleave patterns vary significantly.

The problem of data access conflicts can be solved in two different ways. One is to design specialized interleavers. The other is to employ memory arbitration techniques. Readers interested in designing special interleaver structures to avoid the data access conflict are referred to [6][7][8]. The details, however, are not presented in this paper.

In practice, turbo interleaver structures are often fixed, e.g., in 3G CDMA systems, turbo interleave patterns are specified in the standards. It is impractical to change the standards. In these cases, to solve the problem of data access conflicts, we propose to use efficient memory arbitration schemes. The key point of the memory arbitration is to give a high priority to Read operations while giving a low priority to Write operations. For convenience of later discussion, we consider a typical partitioning scheme. We partition an inter-leaver memory into two segments, one contains even addressed data, denoted as **Seg-1** and the other one contains data with odd addresses, denoted as **Seg-2**. We use two small data & address buffers, denoted as **Buf-1** and **Buf-2**. We propose two kinds of architectures as shown in Figure 2 and Figure 3 respectively, where control states {0, +1} and {0, -1} denote the data flow to/from the memory segment. In other words, {0, +1} means that one or zero datum may be written into the memory segment at each cycle. {0, -1} represents that the memory segment outputs either one or zero datum at each cycle.

At each cycle, the SISO decoder obtains the required datum (the *a priori* information), which could be located in either interleaver memory segment. Without loss of generality, we assume the SISO decoder obtains the required *a priori* information from **Seg-1** at time index **k**. At the same cycle, the SISO decoder outputs the extrinsic information for an information bit, whose corresponding soft inputs were loaded at time index **k-D**, where **D** denotes the sliding window length. Here we assume the sliding window approach [10] is employed in turbo decoding.

The detailed functions for both types of architectures in various cases are described in Table 1. The small buffers in Type-A architecture work as FIFOs (first-in first-out). There are four different cases: (1) no data coming in or out, (2) only data coming in while no data out, (3) only data coming out while no data coming in and (4) there are data coming in and data coming out. The VLSI implementation for these FIFOs is trivial. In general a dual-port memory can be used to synthesize a FIFO.

With regard to Type-B architecture, the small buffers work as simplified FIFOs because there is no Case 4 discussed above. So one single-port memory can be used to implement each simplified FIFO. Compared with Type-A architecture, Type-B architecture requires some extra MUXes (multiplexer) while saving a significant amount of area in buffers. Thus Type-B architecture is more efficient in area. It should also be mentioned that, when there is no read operations being performed (when all branch metrics are computed), the remaining data in **Buf-1** and **Buf-2** can be written into **Seg-1** and **Seg-2** simultaneously once the buffers are not empty, which reduces the overall decoding latency.

Table 1, the function details of both architectures (assuming the datum to be loaded is located in Seg-1)

	Functions of Type-A architecture	Functions of Type-B architecture
Case I: The datum is to be written into **Seg-1**.	(1), Lex and associated address are sent to **Buf-1**. (2), If **Buf-2** is not empty, the stored extrinsic information at the top of the buffer is sent to **Seg-2** by using its associated address.	(1), Lex and associated address are sent to **Buf-1**. (2), If **Buf-2** is not empty, the stored extrinsic information at the top of the buffer is sent to **Seg-2** by using its associated address.
Case II: The datum is to be written into **Seg-2**.	(1), Lex and associated address are sent to Buf-2. (2), If **Buf-2** is not empty, the stored extrinsic information at the top of the buffer is sent to **Seg-2** by using its associated address.	(1), Lex is sent to **Seg-2** by using its associated address.

3. SIMULATION RESULTS

If the interleaver memory is partitioned into the even addressed segment and the odd addressed segment and the Type-B architecture is used for the memory arbitration, from cycle-accurate simulations we find that the smallest size for both buffers is 42 words when applied in CDMA2000 systems (the turbo block size ranges from 378 to 20730)[3], where the MAP algorithm and the sliding window approach are adopted for turbo decoding and the sliding window size is chosen to be 32.

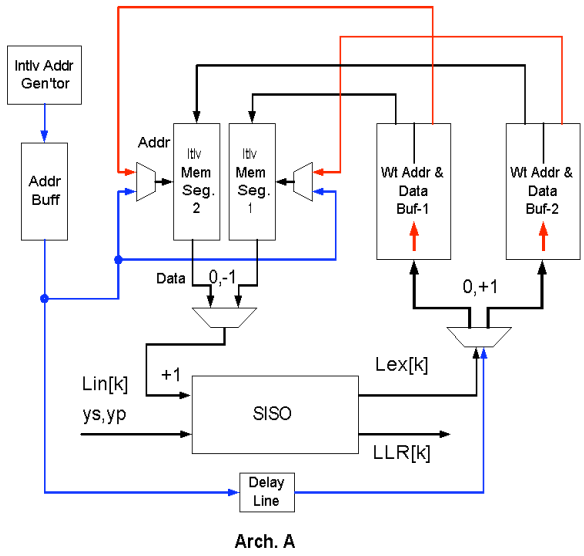

Figure 2, Type-A architecture for the efficient interleaver memory design

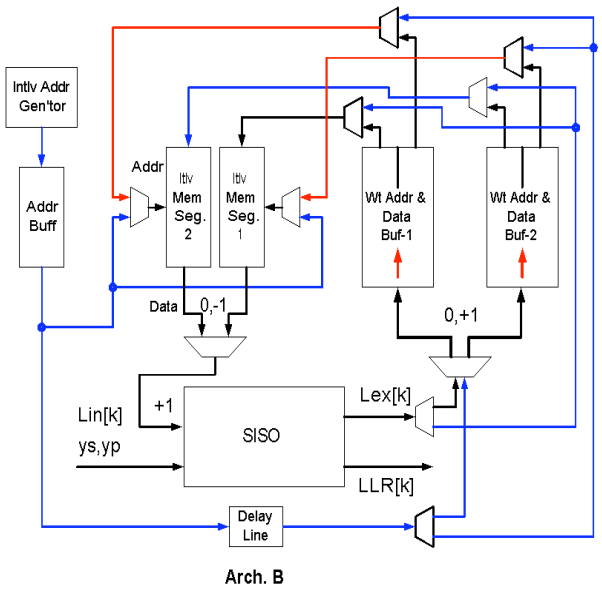

Figure 3, Type-B architecture for the efficient interleaver memory design

In the worst case we need one extra clock cycle to finish the write operations compared with using a dual-port memory to store the extrinsic information. For WCDMA (3GPP) systems, the turbo block size ranges from 40 to 5114. From cycle accurate simulations we find the smallest sizes for **Buf-1** and **Buf-2** are respectively 40 and 42. A few extra processing cycles are introduced in some cases. However, the overall decoding latency is increased by less than 1% for any turbo block size. If the memory is partitioned into four segments and two small buffers are used as before, the requirement for the buffer size will be reduced by 50%.

Next we compute the net savings. Assume the soft inputs are quantized as 4 bits/symbol and the extrinsic information is quantized as 6bits/sumbol as in [9]. For CDMA2000 systems, the maximal block size is approximately 20K, the lowest code rate is _. The input buffer requires 4*4*20K=320K bits, and the interleaver memory consumes 6*20K=120K bits. The memory requirement for the two small buffers is 21*2*(6+14)=840bits< 1Kbits, where 14 bits are used to record the memory address as the **lsb** is not required to store because of the specific memory partitioning. It can be seen that the overhead of the small buffers is less than 0.5 % of the total area of a single-port interleaver memory. The net saving compared with using the dual-port memory is approximately 120K/ (120K+320K) = 27%. The similar computation can be performed with WCDMA systems. The net saving is about 6*5K /(6*5K+3*4*5K) = 30K/90K = 30%. In summary, the net saving with the proposed architecture is significant in 3G CDMA systems. Compared with the traditional single-port approach, the proposed architectures can achieve twice the throughput with an overhead of less than 0.5% of the total hardware of an entire turbo decoder when applied in 3G CDMA systems.

It should be mentioned that the control circuitry with either proposed architecture is trivial. Only the **lsb** of the memory address for the datum to be loaded and that for the datum to be written are required to control the data flow.

Although the above examples are focused on 3G wireless communication systems, the proposed interleaver architectures are clearly applicable to generic turbo decoders. The basic reason is that it is always possible to find an efficient memory partition scheme to ensure the data load to each memory segment is balanced on average so that the length of each buffer can be very small.

4. CONCLUSIONS

The proposed efficient interleaver architectures can either save a significant amount of memory compared with using a dual-port memory or achieve twice the throughput with negligible overhead compared with using a single-port memory. The proposed architectures are applicable to any turbo decoders once the block size is relatively large (e.g., >200). If the interleave pattern is specially designed, there could be no overhead.

REFERENCES

[1] C. Berrou, A. Clavieux and P. Thitimajshia, "Near Shannon limit error correcting coding and decoding: turbo codes", ICC'93, pp 1064-70.

[2] 3rd Generation Partnership Project (3GPP), "Technical specification group radio access network, multiplexing and channel coding (TS 25.212 version 3.0.0)", http://www.3gpp.org.

[3] 3rd Generation Partnership Project 2 (3GPP2), http://www.3gpp2.org.

[4] H. Suzuki, Z. Wang and K. K. Parhi, "A K=3, 2Mbps Low Power Turbo Decoder for 3rd Generation W-CDMA Systems", IEEE Custom Integrated Circuits Conference (CICC), 2000, pp 39-42.

[5] P. Robertson, E. Villeburn and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms in the log domain", in IEEE ICC'95, vol. 2, pp 1009-1013, Seattle, 1995.

[6] Z. Wang, "Low complexity, high performance turbo decoder design", Ph.D. dissertation, Chapter 5, Univ. of Minnesota, Aug. 2000.

[7] A. Giulietti, L. Perre and M. strum, "Parallel turbo coding interleavers: avoiding collisions in accesses to storage elements", Electronics Letters, Feb. 2002, vol. 38, No. 5, pp 232-34.

[8] M. Thul, N. When and L. P. Rao, "Enabling high-speed turbo-decoding through concurrent interleaving", ISCAS 2002, pp I- 897~890, vol 1.

[9] Z. Wang, H. Suzuki and K. K. Parhi, "VLSI implementation issues of Turbo decoder design for wireless applications", IEEE Workshop on Signal Processing Systems, Design and Implementation (SiPS'99), pp. 503-512, Oct. 1999.

[10] A. J. Viterbi, "An intuitive justification of the MAP decoder for convolutional codes", IEEE Journal on Selected Areas in Communications, vol.16, pp. 260-264, February 1998.