15th European Signal Processing Conference (EUSIPCO 2007), Poznan, Poland, September 3-7, 2007, copyright by EURASIP

EFFICIENT PARALLEL MEMORY ORGANIZATION FOR TURBO DECODERS

Perttu Salmela, Ruirui Gu, Shuvra S. Bhattacharyya*, and Jarmo Takala*

Institute of Digital and Computer Systems
Tampere University of Technology
Tampere, Finland
{perttu.salmela, jarmo.takala}@tut.fi

*Department of Electrical and Computer Engineering and
Institute for Advanced Computer Studies
University of Maryland, College Park, MD, USA
{rgu, ssb}@eng.umd.edu

ABSTRACT

An efficient turbo decoder must access memory in parallel and with two different access patterns. It is shown that the problem of accessing memory both with sequential and interleaved access patterns is analogous to the graph coloring problem. The derivation proves that the obtained graph is bipartite and, therefore, only two memory banks are required in theory. For practical implementations, a system with four memory modules and a buffer is proposed. It is shown that modest buffer length is sufficient for 3GPP standard interleavers. There is no performance degradation in the proposed system and the address generation and memory interfaces are of modest complexity.

1. INTRODUCTION

Turbo codes [1] are applied in 3G telecommunication systems [2, 3] and, therefore, there is strong demand for efficient and economical implementations. Especially the memory requirements of turbo decoders are high due to the long block lengths. In addition, rapid decoding requires dual access to the memory. An obvious solution of applying dual port memory is uneconomical as it takes more chip area than a memory split into parallel accessible banks. Avoiding even single access conflict is crucial, since a conflict would require more complicated control logic capable of interrupting the decoding process. Parallel memory access of a turbo decoder is of high importance as indicated by two patent applications [4, 5].

In [6] and [4] a conflict free access scheme is developed. The methods are based on address generation and bank selection functions, which are derived from the interleaving patterns of the 3GPP standard. Both methods require six memory banks for conflict free accesses. In [6] also a structure with four memory banks is presented. With four banks only few access conflicts are present. As a drawback, the structures are specific to only one class of interleaver as there is a close connection of the interleaving patterns and bank selection. For the same reason, the structures depend on the additional information provided by the interleaver.

In [7] a conflict free mapping is derived with an iterative annealing procedure. The native block length of the algorithm is product of the number of parallel component decoders and the number of memory banks. Even if the reconfiguration is mandatory for varying interleaving patterns, no hardware implementation is presented for the annealing procedure.

In [8] graph coloring is used to find mappings. It uses more memory modules than [7], but a hardware architecture for the reconfiguration is presented. The reconfiguration takes $O(10K)$ clock cycles for K length code block [8]. For comparison, one conflict would take one additional clock cycle. Therefore, it can be more advantageous to suffer all the conflicts instead of reconfiguration in some cases. In addition, the address computations in [8] require division and modulus computations which are difficult to implement on hardware when the block length is not a power of two.

A different approach is applied in [9, 10, 5] where buffers are applied instead of deriving conflict free address generation and bank selection functions. In [9, 10, 5] high-speed decoding with several write accesses is assumed. For each writer there is one memory bank and for each bank there is a dedicated buffer. In [10] the buffered approach is developed further and the memories are organized in ring or chordal ring structures. The work is continued in [11] where a packet switched network-on-chip is applied and several network topologies are presented. To reduce sizes of queue buffers and to prevent overflows the network flow control is applied.

In this paper, an important result for parallel memory accesses in turbo decoders is derived as it is shown that the memory can be split into two banks to maintain conflict free accesses. The derivation is based on a graph coloring formulation and on construction of the graph that is to be colored. The result can be applied with systems relying on constant interleaving patterns. For systems applying varying interleaving patterns a practical memory structure with four banks is proposed. The developed structure applies trivial address generation and bank selection functions. Instead of resolving all the conflicts, the developed structure applies buffering to maintain uninterrupted memory accesses. It is shown that only modest buffer length is sufficient for 3GPP turbo codes. Contrary to previous buffered parallel access methods [9, 10, 5] our method relies on the asymmetric throughput rates of turbo decoder side and memory subsystem side. Instead of one memory bank per access, we apply a total of four banks to guarantee dual access with modest buffer length. Furthermore, instead of dedicated buffers, we apply a centralized buffer to balance buffer length requirements, which leads to even shorter buffer length. The results show that the proposed method outperforms [6] and [4] in terms of required number of memory banks. We also show the advantage of asymmetric throughput rates in comparison with equal number of parallel accesses and memory banks.

2. PROBLEM DESCRIPTION

In principle, the turbo decoder exploits a soft in soft out component decoder in an iterative process where extrinsic information generated on the previous half iteration is fed to the next half iteration. In practical implementation the data can reside always in sequential order in the memory and no de-

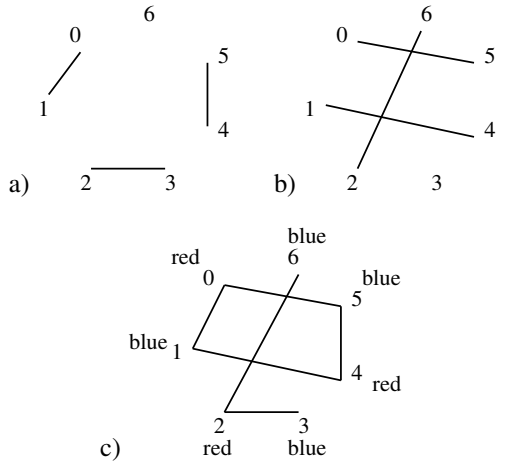


Figure 1: In graph presentation the connected vertices are accessed in parallel. (a) Sequential access pattern. (b) Interleaved access pattern. (c) Combined sequential and interleaved access patterns. The graph is colored with two colors.

interleaving is required. During the first half decoding iteration, it is both read and written in parallel with a sequential access pattern. During the second half iteration, it is both read and written in parallel with an interleaved access pattern. Thus, the order remains unchanged and de-interleaving is implicit. Avoiding explicit de-interleaving is important due to the high implementation complexity of de-interleaving functions. Between the read and write operations there is a constant distance proportional to the applied window length in sliding window algorithm [12]. The addresses, to which data is written, are not read during the same half iteration. Naturally, two parallel read and write operations (a total of four operations) can be substituted with two parallel read operations followed by two parallel write operations. So, the access scheme must provide such a mapping from addresses to parallel accessible memory banks that conflicts can be avoided or, alternatively, performance degradation due to the conflicts is avoided.

3. MEMORY ACCESS AS GRAPH COLORING PROBLEM

It is assumed that the distance between parallel read and write operations is one address unit. In practice, this is the same condition as if parallel read operations of adjacent addresses were followed by parallel write operations of adjacent addresses with arbitrary distance between read and write operations. Thus, a sequential access pattern $i = 0,1,2,3,\ldots,K-1$ has parallel access set consisting of pairs $(0,1),(2,3),\ldots,(K-2,K-1)$ with even block length K.

The memory access pattern is presented in a graph form so that each accessed location, i.e., original address, is denoted with a vertex of graph. When two addresses are accessed in parallel there is an edge between them in the graph. For the sequential parallel access set $(0,1),(2,3),(4,5),(6)$, such a graph is shown in Figure 1(a). Since the block length, K, is odd, there is one unconnected vertex. With interleaving $\pi_i = 1,4,0,5,2,6,3$ and parallel access set $(1,4),(0,5),(2,6),(3)$, a similar graph is shown in Figure 1(b).

The graph coloring problem involves mapping colors to vertices of graphs in such a way that adjacent vertices, i.e., vertices connected with edges, do not have the same color. For graphs in Figure 1(a) and (b) such coloring can be found easily, since each vertex has at maximum one adjacent vertex. A combined graph in Figure 1(c) presents the both the sequential and interleaved access patterns. In addition, a coloring of vertices is marked in Figure 1(c). Thus, if the colors represent memory banks, both sequential and interleaved access patterns are possible, since adjacent vertices, i.e., vertices which are accessed in parallel, have different colors.

A bipartite graph is a graph whose vertices can be partitioned into two disjoint sets such that no vertices in the same set are adjacent. A graph is bipartite if and only if it is a two-colorable graph. Especially, all trees are bipartite and a graph is bipartite if and only if all its cycles are of even length [13]. Thus, to prove that two memory banks are enough for conflict-free sequential and interleaved memory access patterns it is sufficient to show that the combined graph of both access patterns cannot have cycles of odd length.

The proof is based on building the combined graph, G_C. We initialize it with the graph of the sequential access pattern, $G_C = G_S$, and insert edges of the interleaved access pattern graph, G_π, to G_C one by one. No vertices need to be inserted since both graphs contain the same vertices. The degree of vertex $d_g(v)$ gives the number of edges connected with v in graph G_g, $g \in \{S,\pi,C\}$. In the combined graph it is always less than or equal to sum of degrees in both graphs, i.e., $d_C(v) \leq d_S(v) + d_\pi(v) \leq 2$. If the block length is odd G_S includes an unconnected vertex u, $d_S(u) = 0$. If it is connected vertex in G_π, then $d_C(u) = 1$. If it is unconnected also in G_π then $d_C(u) = 0$. Thus, $d_C(u) \leq 1$.

In the beginning, there are no cycles in G_C and the length of all the paths equals one, i.e., paths have odd length. Arbitrary edges, e, from G_π are inserted to G_C, which can modify G_C in some of the following ways:

1. e connects two odd length paths, so the length of the new path is odd.
2. e connects an odd length path to the unconnected vertex, u. This results in an even length path, $p_{u,0}$. Since there can be at maximum one u, there can be only one even length path.
3. e connects the even length path, $p_{u,j}$, to an odd length path, which lengthens the even length path, $p_{u,j}$, to $p_{u,j+1}$.
4. e connects two ends of the odd length path which results in even length cycle.

There are three more clarifying observations:

1. e cannot be connected to any vertex, v_m, in the middle of any path. Otherwise $d_C(v_m) > 2$, which is a contradiction.
2. e cannot connect two ends of the even path and result in odd cycle. One end of the even path is always u and $d_C(u) \leq 1$. So, u cannot be included in any cycle.
3. if e is included already in G_C, the previous state remains unchanged.

Thus, all the cycles in the combined graph, G_C, have even length and, therefore, the graph is bipartite and it can be colored with two colors. So, there exists a conflict-free memory bank mapping for sequential and interleaved access patterns and only two banks are required. Even if the graph coloring can be too complex to be computed on the fly, the existence

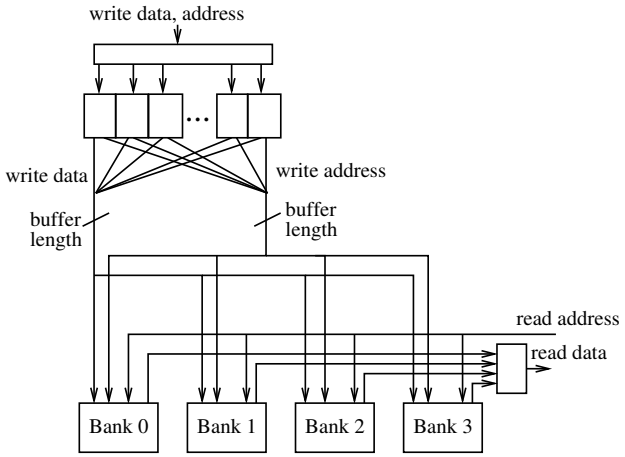


Figure 2: High-level description of the proposed memory structure with buffered write operations.

```
process memory_interface(id) begin
   if bank(read_addr) = id then
      data_out := memory[ag(read_addr)]
   elif bank(abuff[0]) = id then
      memory[ag(abuff[0])] := dbuff[0]
   elif bank(abuff[1]) = id then
      memory[ag(abuff[1])] := dbuff[1]
   ...
   elif bank(abuff[N-1]) = id then
      memory[ag(abuff[N-1])] := dbuff[N-1]
   end
end

function bank(address) begin
   return address mod 4
end
function ag(address) begin
   return address >> 2
end
```

Figure 3: Functional description of the memory bank interface. Buffer length is `N`. Data in buffer is referred with `dbuff[]` and addresses with `abuff[]`.

of a two bank solution is important for any system applying long block length and a constant interleaving pattern.

4. BALANCING CONFLICTS WITH BUFFERED WRITES

In the 3GPP standard, different interleaving patterns are specified for all the block lengths $K = 40, 41, \ldots, 5114$ and, therefore, the memory bank mapping should be computed on the fly instead of statically. Even if graph coloring results in minimum number of memory banks, the computation takes too many clock cycles. To meet practical demands of on the fly mapping, a simpler memory bank mapping is required for 3GPP turbo decoders.

4.1 Structure

Instead of trying to solve all the conflicts with complex memory bank mapping and address generation like in [6, 4, 7, 8], our approach in this paper is to use a very simple memory bank selection function and to maintain a constant throughput on the component decoder side in spite of conflicting accesses on the memory bank side. In [6, 4], a total of six memory banks are required for conflict free memory access. In this study, only four banks are suggested. There will be conflicts, but they do not interrupt the decoding and degrade the performance.

The proposed method is based on buffering the conflicting write accesses. A high-level block diagram is shown in Figure 2. The bank selection function is a simple modulo operation of the address and the number of banks. So, with four banks, the bank selection and address generation are carried out by simply hardwiring the bits to the new positions.

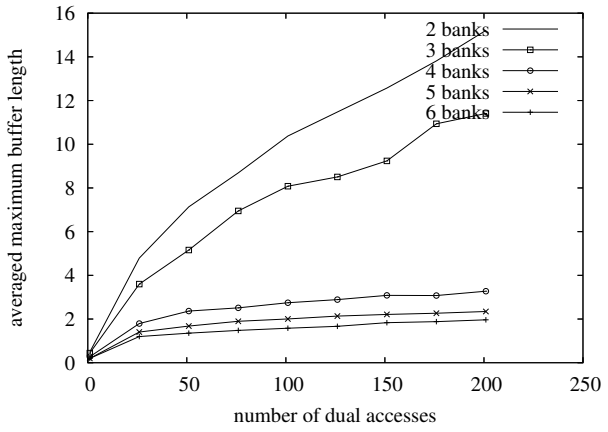
In principle the proposed system in Figure 2 gives the highest priority for memory reads. They are always served to allow continuous decoding. On the contrary, write operations are inserted to the buffer. All the memory banks that do not serve the read operation are free to serve write operations in the buffer. The described functionality of the memory bank interface is shown in Figure 3. The proposed buffer must be able to be read and written in a random access manner and in parallel by all the memory bank interfaces. Thus, it must be implemented with registers. However, the length of the buffer is modest as will be shown later on. The buffer length is obtained by simulating the system with all the targeted interleaver patterns and selecting the minimum buffer length which does not cause an overflow.

4.2 Asymmetric Throughput Rates

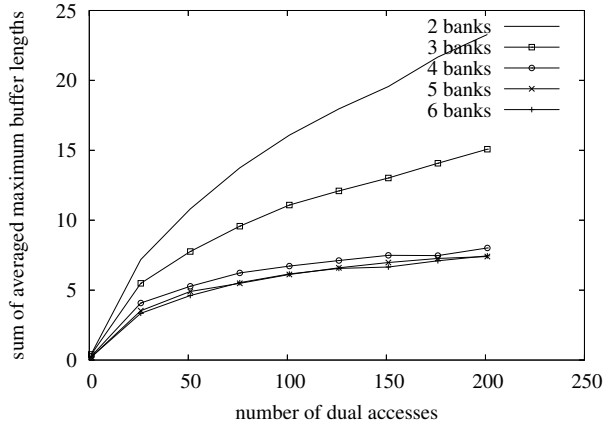
The ability of the proposed method to perform without performance degradation is based on the asymmetric throughput rates and throughput capability between the decoder side and memory bank side. The decoder produces memory accesses at a constant rate, two accesses per clock cycle, i.e., one read and one write operation. On the contrary, the memory system is capable of maximum throughput directly proportional to the number of banks. In [9], the average rate of accesses per bank is one per clock cycle. In our approach, the average rate of accesses per bank is less than one per clock cycle.

Buffered accesses are presented in [9, 10, 5], but there are certain crucial differences. In the aforementioned studies only the write operations are considered. A decoder with separate read memory for extrinsic information is targeted. However, several parallel write operations are assumed, which results in similar types of conflicting accesses as in our problem statement. A more significant difference is the ratio of memory banks to the number of parallel accesses. In [9, 10, 5], there is only one memory bank per writer. In addition, there is a dedicated buffer per memory.

The effect of the ratio of parallel accesses to the number of banks is exemplified in Figure 4(a) where random dual accesses are generated for systems with 2 to 6 memory banks and the required buffer length is reported. When the access rate is less than the capacity of memory subsystem, there are free clock cycles, when the accesses can be emptied from the buffer and the required buffer length does not increase rapidly. On the contrary, if the access rate is higher than memory throughput, the buffer fills continuously.



a)



b)

Figure 4: Required buffer lengths with random access patterns. (a) centralized shared buffer. (b) dedicated buffers.

4.3 Centralized Buffer

In principle, the buffer balances memory accesses. Balancing is targeted also with single shared buffer instead of dedicated buffers for each memory bank. If there are dedicated buffers for memory banks their length must match the maximum requirement. However, the length of combined buffer is less than sum of dedicated buffers. This is natural, since only one buffer can be filled at a time if dedicated buffers are used. In Figure 4(b), such dedicated buffers are assumed and the sum of required buffer lengths of all the memory banks is reported. When compared to the proposed method with a single shared buffer in Figure 4(a) savings in buffer length can be noticed. In Figure 4(a) and (b) several simulations with random access patterns are run to obtain averages of the maximum buffer lengths.

5. RESULTS

The proposed method is applied with interleaving patterns of the 3GPP standard and the required buffer lengths are reported in Table 1. The buffer length is the minimum required buffer size to avoid the overflows with all the 3GPP interleaver patterns with $K = 2557, \ldots, 5114$. For practical implementation only block lengths greater than 2556 are interesting. Since the maximum block length is 5114, it is the amount of available memory in four banks. If the block length is less than 2557, banks can be organized as dedicated

Table 1: Overflow free buffer length of the proposed method with 3GPP interleaving patterns. Block length $K = 2557 \ldots 5114$.

R/W distance	buffer length
32	10
48	11
64	15
96	10
128	10

Table 2: Area costs (in gates) of the proposed memory banking structures. Clock frequency f=100 MHz.

Word length	6	8	10	12
Buffer (length=10)	1478.75	1622.75	1754.75	1888.75
Memory interface	353.50	389.50	397.75	418.50

read and write memories and after every half iteration, the roles of the banks are interchanged. The distance between read and write operations is varied in Table 1. The distance depends on the window length and schedule of forward and backward metric computations in the sliding window algorithm. The results in Table 1 indicate clearly that modest buffer length in the range 10–15 is sufficient for the parallel memory access of the 3GPP compliant turbo decoder.

In the end of a half iteration, there are no parallel read accesses but only write accesses for the last samples and the utilization of the buffer cannot increase. If the buffer is not emptied during this phase, extra clock cycles are spent to empty the buffer. The experimented cases in Table 1 do not require such extra cycles, i.e., the buffer is empty when the decoder issues the last write operation.

The area costs of the proposed structure for parallel memory access are shown in Table 2. The costs of the buffer and memory interface are separated. For four memory banks, four interfaces are required. The word length is varied in Table 2 but the address width is constant 13 bits, since the maximum block length is 5114 in 3GPP systems. The complexity of memory interface is relatively low, since it does not require complex arithmetics and the buffer length is short.

6. CONCLUSIONS

In this study, efficient parallel memory access for turbo decoders was addressed. With the aid of a graph coloring formulation it was shown that, in theory, only two memory banks are required to enable parallel sequential or interleaved access patterns. For practical 3GPP compliant implementations, in which interleaving patterns are not statically known, a structure with four memory banks was proposed. In comparison to existing methods the main differences in our approach are the number of banks, ratio of number of banks to parallel accesses, and sharing of the buffer. Analysis of these differences shows advantages of low complexity and ability to maintain uninterrupted decoding without extra delay.

REFERENCES

[1] C. Berrou, A. Glaviex, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *IEEE Int. Conf. Commun.*, vol. 2, Geneva, Switzerland, May 1993, pp. 1064–1070.

[2] *3GPP TS 25.212; Multiplexing and Channel coding (FDD)*, 3rd Generation Partnership Project Std. 4.3.0, Dec. 2001.

[3] *3GPP2 C.S0002-C Physical Layer Standard for cdma2000 Spread Spectrum Systems*, 3rd Generation Partnership Project 2 Std. 1.0, May. 2002.

[4] D.-S. Shiu and I. Yao, "Buffer architecture for a turbo decoder," International Patent Application WO 02/093 755 A1, Nov., 2002.

[5] F. Berens, M. J. Thul, F. Gilber, and N. Wehn, "Electronic device avoiding write access conflicts in interleaving, inparticular optimized concurrent interleaving architecture for high throughput turbo-decoding," European Patent Application EP 1 401 108 A1, Mar., 2004.

[6] P. Salmela, T. Järvinen, T. Sipilä, and J. Takala, "Parallel memory access in turbo decoders," in *Int. Symp. on Personal, Indoor, and Mobile Radio Communications*, Beijing, China, Sep. 2003, 2157–2161.

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

[8] X. He, H. Luo, and H. Zhang, "A novel storage scheme for parallel turbo decoder," in *IEEE Vehicular Technol. Conf.*, Dallas, TX, USA, Sep. 2005, 1950–1954.

[9] M. J. Thul, N. Wehn, and L. P. Rao, "Enabling high-speed turbo-decoding through concurrent interleaving," in *IEEE Intern. Symp. on Circuits and Systems*, vol. 1, Phoenix, USA, May. 2002, pp. 897–900.

[10] M. J. Thul, F. Gilbert, T. Vogt, G. Kreiselmaier, and N. Wehn, "A scalable system architecture for high-throughput turbo decoders," *Journal of VLSI Signal Processing*, vol. 39, no. 1–2, pp. 63–77, 2005.

[11] C. Neeb, M. J. Thul, and N. Wehn, "Network-on-chip-centric approach to interleaving in high throughput channel decoders," in *IEEE Int. Symp. on Circuits and Systems*, Kobe, Japan, May 2005, 1766–1769.

[12] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE J. Select. Areas Commun.*, vol. 16, no. 2, pp. 260–264, Feb. 1998.

[13] S. Skiena, *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Boston, MA, USA: Addison-Wesley, 1990.