Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2009, Article ID 704174, 3 pages
doi:10.1155/2009/704174

Letter to the Editor

Comments on "Techniques and Architectures for Hazard-Free Semi-Parallel Decoding of LDPC Codes"

Kiran K. Gunnam,[1, 2] Gwan S. Choi,[2] and Mark B. Yeary[3]

[1] *Channel Architecture, Storage Peripherals Group, LSI Corporation, Milpitas, CA 95035, USA*
[2] *Department of ECE, Texas A&M University, College Station, TX 77843, USA*
[3] *Department of ECE, University of Oklahoma, Norman, OK 73019, USA*

Correspondence should be addressed to Kiran K. Gunnam, kgunnam@ieee.org

Received 7 December 2009; Accepted 7 December 2009

This is a comment article on the publication "Techniques and Architectures for Hazard-Free Semi-Parallel Decoding of LDPC Codes" Rovini et al. (2009). We mention that there has been similar work reported in the literature before, and the previous work has not been cited correctly, for example Gunnam et al. (2006, 2007). This brief note serves to clarify these issues.

The recent work by Rovini and others in [1] states that "Gunnam et al. describe in [10] a pipelined semi-parallel decoder for WLAN LDPC codes, but the authors do not mention the issue of the pipeline hazards; only, the need of properly scrambling the sequence of data in order to clear some memory conflicts is described." On the contrary, we gave detailed explanation of our decoder architecture, concepts of out-of-order processing in [2–6]. The proposed approach Unconstrained Output Processing (UoP) in [1] is similar to our approach outlined in [2–6]. So we would like to clarify more in this matter.

We describe in [2–6] a pipelined semi-parallel decoder for WLAN LDPC codes, that used scheduling of layered processing and out-of-order block processing to minimize the pipeline hazards and memory stall cycles. The following paragraph from [4, Page 1, Column 2], correctly describes our work: "This paper introduces the following concepts to LDPC decoder implementation: Block serial scheduling [5], value-reuse, scheduling of layered processing, out-of-order block processing, master-slave router, dynamic state. All these concepts are termed as on-the-fly computation as the core of these concepts is based on minimizing memory and re-computations by employing just-in-time scheduling." More detailed explanations and illustrations can be found in the presentations [5, 6] which were available on-line from October 2006 and May 2007, respectively.

Also [1] did not cite our work on layer reordering for optimizing the pipeline and memory accesses. In [3, Page 4, Column 2], last paragraph, we clearly mention that "It is possible to do the decoding using a different sequence of layers instead of processing the layers from 1 to j which is typically used to increase the parallelism such that it is possible to process two block rows simultaneously [4]. In this work, we use the concept of reordering of layers for increased parallelism as well as for low complexity memory implementation and also for inserting additional pipeline stages without incurring overhead."

Our proposal of out-of-order processing (OoP) for the layered decoding [2–6] is to process the circulants in a layer in out-of-order (not necessarily sequential) to remove the pipeline and memory conflicts. This includes the partial state processing and other related steps (Rold message generation, Q message generation, CNU partial state processing. that is, the processing step of finding Min1,Min2, Min1 ID), in out-of-order fashion and the processing of Rnew messages in out-of-order fashion. For instance, while processing the layer 2, the blocks/circulants which depend on layer 1 will be processed last to allow for the pipeline latency. Also Rnew selection is out-of-order (these messages will come from the most recently updated connected block), so that it can feed the data required for the PS processing of the second layer. A dependent circulant or connected circulant is the non-zero circulant that supplies the last updated information of P message to the specified nonzero circulant. The dependent layer is the layer which contains dependent circulant. So circulants in second layer will get the latest P update based on the Rnew messages from different connected circulant in different connected layers. Thus OoP for PS processing is across one layer (i.e., at any time the CNU partial state processing is concerned with starting and completing one layer; however, the order of the circulants processed in the layer is processed in out-of-order to satisfy the pipeline and memory constraints); OoP for Rnew message generation is across several layers. Also the P update (Q + Rnew), in [2, (9)] is computed on-the-fly along with reading of the Q message of the last updated circulant in the same block column from the Q memory and the Rnew message generation that is, at the precise moment when it is needed; this avoids the use of P memory and needs a single-port read and single-port write Q memory whose storage capacity is equal to the code length multiplied by the word length of Q message. The bandwidth of this memory measure in terms of number of Q messages is equal to the decoder parallelization [2–6]. Other decoder hardware architectures and implementations use both P memory and Q memory, use mirror memories, or use more complicated multiported memory. Illustrations for out-of-order processing were given in [5, 6].

We gave more explanation in [5]: "The decoder hardware architecture is proposed to support out-of-order processing to remove pipeline and memory accesses or to satisfy any other performance or hardware constraint. Remaining hardware architectures will not support out-of-order processing without further involving more logic and memory. For the above hardware decoder architecture, the optimization of decoder schedule belongs to the class of NP-complete problems. So there are several classic optimization algorithms such as dynamic programming that can be applied. We apply the following classic approach of optimal substructure."

Step 1. "We will try different layer schedules (j! i.e., j factorial of j if there are j layers). For simplicity, we will try only a subset of possible sequences so as to have more spread between the original layers."

Step 2. "Given a layer schedule or a re-ordered H matrix, we will optimize the processing schedule of each layer. For this, we use the classic approach of optimal substructure that is, the solution to a given optimization problem can be obtained by the combination of optimal solutions to its sub problems. So first we optimize the processing order to minimize the pipeline conflicts. Then we optimize the resulting processing order to minimize the memory conflicts. So for each layer schedule, we are measuring the number of stall cycles (our cost function)."

Step 3. "We choose a layer schedule which minimizes the cost function that is meets the requirements with less stall cycles due to pipeline conflicts and memory conflicts and also minimizes the memory accesses (such as FS memory accesses to minimize the number of ports needed and to save the access power and to minimize the more muxing requirement and any interface memory access requirements)."

Also we would like to mention how we calculate the architecture efficiencies: we mention in [2], "Here, all calculations for the decoded throughput are based on an average of 5 decoding iteration to achieve frame error rate of 10e-4, while $it_{\max}$ is set to 15." If we are considering the actual system throughput, then we should consider how many maximum iterations the system can run and what is the additional overhead from LLR/Q memory and hard decision memory statistical buffering and loading and unloading times. We have close to 1.5 iteration overhead due to statistical buffering. So mixing the average number of iterations with actual system throughput to calculate the decoder core architecture efficiency is not a fair metric. In our works [2–6], for the decoders based on one-circulant processing, the number of clock cycles for decoding of each block/circulant is 1. Note that [2] and [3] designs are similar and have the pipeline depth of 5. The Clock Cycles Per Iteration (CCI) for most of the IEEE 802.11n and IEEE 802.16e H matrices after reordering of layers and out-of-order processing and data forwarding and speculative computations is the number of blocks in H matrix. The only exception is the rate 5/6 matrix of IEEE 802.16e and IEEE 802.11n. For 802.16e 5/6 matrix, the CCI is 87 clock cycles to process 80 blocks. For 802.11n 5/6 matrix, the CCI is 85 clock cycles to process 79 blocks. We gave the worst case for CCI as total number of blocks in H matrix + 2 cycle overhead per each layer in [3, Page 6, Column 1, Line 23–28]. So if we were to report the Architecture Efficiency = CCI/Ideal CCI, the architecture efficiency for our decoders would be 1 for all the codes except 2 cases. For 802.16e 5/6 matrix, this number would be 1.0875. For 802.11n 5/6 matrix, the number would be 1.0759. Even if we use the above worst case number reported in [3] for all the codes even though it is not necessary, then the architecture efficiency number would vary from 1.0759 to 1.29 for 802.11n codes and 1.0875 to 1.3158 for 802.16e codes.

Also our work covers more aspects. We can apply OoP for PS processing across multiple layers. While waiting for the data from the currently processed layer 1, we can start processing the independent circulants in next layer 2 that will not depend on current layer 1 and also the circulants in layer 3 that will not depend on layer 1 and layer 2. In [5], "also we will sequence the operations in layer such that we process the block first that has dependent data available for the longest time. This naturally leads us to true out-of-order processing across several layers. In practice we won't do out-of-order partial state processing involving more than 2 layers."

References

[1] M. Rovini, G. Gentile, F. Rossi, and L. Fanucci, "Techniques and architectures for hazard-free semi-parallel decoding of LDPC codes," *EURASIP Journal on Embedded Systems*, vol. 2009, Article ID 723465, 15 pages, 2009.

[2] K. Gunnam, G. Choi, W. Wang, and M. Yeary, "Multi-rate layered decoder architecture for block LDPC codes of the

IEEE 802.11n wireless standard," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '07)*, pp. 1645–1648, New Orleans, La, USA, May 2007.

[3] K. K. Gunnam, G. S. Choi, M. B. Yeary, and M. Atiquzzaman, "VLSI architectures for layered decoding for irregular LDPC codes of WiMax," in *Proceedings of the IEEE International Conference on Communications (ICC '07)*, pp. 4542–4547, Glasgow, UK, June 2007.

[4] K. K. Gunnam, G. S. Choi, W. Wang, E. Kim, and M. B. Yeary, "Decoding of quasi-cyclic LDPC codes using an on-the-fly computation," in *Proceedings of the 4th Asilomar Conference on Signals, Systems and Computers*, pp. 1192–1199, October-November 2006.

[5] K. Gunnam, *Area and energy efficient VLSI architectures for low density parity-check decoders using an on-the-fly computation*, Ph.D. presentation, Texas A&M University, College Station, Tex, USA, October 2006, http://dropzone.tamu.edu/~kirang/10112006.pdf.

[6] K. Gunnam, G. Choi, W. Wang, and M. Yeary, "Multi-rate layered decoder architecture for block LDPC codes of the IEEE 802.11n wireless standard," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '07)*, pp. 1645–1648, New Orleans, La, USA, May 2007.