International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

Design of Single and Multimode Channel Decoders for Mobile Wireless Communication

T. Merlin Leo[1], Dr. T. Latha[2]

[1]Department of ECE, Sun College of Engineering and Technology, Kanyakumari, Tamilnadu, India

[2]Department of ECE, St.Xaviers Catholic College of Engg, Chunkankadai, Kanyakumari, Tamilnadu, India

Abstract: ***This paper presents a study into the performance, reliability and costs of channel decoders for different channel codes such as Convolutional _turbo, Viterbi, Reed Solomon and LDPC codes. A framework for multi-code forward error correction (FEC) architectures is identified to implement four dedicated decoders for convolutional_turbo, Reed Solomon and LDPC codes as well as a single decoder capable of decoding all the four. Performance estimations for all the four architectures are presented to provide a clear and fair comparison between single-mode and multi-mode decoders.***

Keywords: Channel Decoder, LDPC, Turbo Decoder, Reed Solomon Convolutional, Viterbi Decoder, Multi-Standard

1. Introduction

In the present mobile communication systems, more than one type of channel coding algorithm is used due to common standardization . WIMAX has gained a wide popularity due to the growing interest and diffusion of broadband wireless access systems. In order to be flexible and reliable WIMAX adopts several different channel codes, namely convolutional-codes (CC), convolutional-turbocodes(CTC), block-turbo-codes (BTC) and low-density-parity-check (LDPC) codes, that are able to cope with different channel conditions and application needs.Hence it becomes necessary to accommodate for all the required coding types and parameter sets. The main concern here is the decoding part of the signal processing chain, as it is computationally complex than the encoders. FEC decoders are traditionally implemented as specialized IP blocks for different code types. Recently more efforts are taken to merge decoder architectures for different decoding algorithms.. This type of approach gives the system designer the option to reduce the number of IP blocks in the System-on-Chip (SoC) .A question that has not been answered yet is just how efficient it is to merge decoders and if it would be advantageous to keep dedicated decoders over a multi-mode solution. In general, it is difficult to compare architectures since they are often implemented on different technologies with different design goals in mind, making it hard to accurately assess them. Most authors just mention logic sizes of the combined decoders and do not do a comparison to single-mode decoders. To more deeply explore the effects of merging data paths for all the four cases of viterbi ,CTC,RS and LDPC decoding, we will use the multi-mode ASIP decoder presented in [7] as a reference and starting point. It is programmable to decode each of these code types and is going to serve as a basis to implement four separate ”single-mode” decoders, each of which is dedicated to viterbi, CTC,RS or LDPC decoding, respectively. All the four single-mode decoders shall follow the same ASIP approach and use the same synthesis parameters to allow a fair evaluation of the multi-mode decoder. Since all conditions are kept constant any differences in logic size can directly be attributed to the changes in architecture design, which makes it easy to actually judge the effects of these design changes.

2. Merging Decoding Algorithms

This section deals with the different approaches that can be taken to combine the functionality of several channel decoders:

A. Memory sharing

This is the most high-level approach to merge decoders. The logic is left mostly unchanged andthe separate decoder instances are simply connected to the same memory banks and I/O or system interfaces. This approach is not too hard to implement and is able to provide good logic area savings in cases where the storage requirements are high. But the results do depend on the application and savings for scenarios with shorter codeword or in environments where there exists an abundance of memories such as an FPGA with a lot of built-in memory blocks. An example for an implementation employing memory sharing can be found in [4].

B. Functional unit merging

If we step one hierarchical level down and look at the functional units (FUs - An FU in our case is used to refer to a building block of a decoder that performs a part of the decoding algorithm, e.g. branch metric computation or traceback) inside a decoder, there is a lot of redundancy that can be exploited, when trying to merge two decoders. First, we identify that in both data paths, FU1 has similar input and output connections, so it makes sense to merge the two FU1 instances to reuse the connection to memory and any output buffers that might exist. This means the functionality of path A FU1 and path B FU1 are put into one FU and made accessible by a selector switch while the number of in/out ports are kept as they were. Now, in the resulting data path we were able to save one connection (between FU1 and memory). Usually, such a FU will have registers for the output ports due to timing or data routing concerns. They can also be shared. Identifying whole groups of FUs that can be utilized in this way can lead to even bigger savings.

C. Algorithmic operation merging

Even more low-level is the approach of merging the processing of algorithm kernels such as Add-Compare-

Select (ACS). Here, similar operations in the target algorithms are identified and used to design acommon basic processing unit. This is of course limited to such algorithms that share suitable operations. For a detailed example of merging ACS and a trellis-based LDPC decoding algorithm, see [3].

3. Multi-Mode Decoder Architecture

In [7] we have a multi-core multi-mode decoder for convolutional, turbo viterbi Reed Solomon and LDPC codes. Each core of the decoder can be independently programmed to decode either CCs,CTCs or LDPC codes. In fig. 3 we see an overview schematic of one such decoder core showing the control path and instruction memory (IMEM) as well as the four parallel processing elements (PEs) and respective data memories connected through an interconnect network. As control scheme we apply a very long instruction word (VLIW) approach, where every FU is programmable with an own function instruction word within the VLIW. The PEs work in parallel in a SIMD fashion.The data path merging for this architecture was mostly performed on the FU level. Inside a PE, there are several FUs .Most of these are used for more than one algorithm. The Gamma unit, for example, calculates branch metrics in CC/CTC mode and is used for subtracting extrinsic information from channel log-likelihood ratio (LLR) values in LDPC mode. One PE consists of enough computation blocks to be able to process 16 trellis states in parallel when forward-recursion with ACS is used, as in the Viterbi algorithm. Alternatively, it can process 8 states forward/backward in parallel using the BCJR algorithm. As another option, a PE can process 8 LPDC check node updates in parallel, where connected edge values are loaded sequentially, one per cycle. LDPC decoding is implemented as layered decoding based on the offset Min-Sum algorithm [8]. Reed Solomon (RS) codes are a sort of non-binary cyclic codes. Then based on the Proposed RiBC algorithm, we can achieve high-speed, throughput and improved error correcting capability than Hard Decision Decoding (HDD) design with less area. The original burst error correcting algorithm consists of inversion operation and some computational steps with long data dependency and long data path. To resolve these problems we reformulate BC algorithm to the proposed reformulated inverse free burst error correction algorithm. The Reformulated inverse free burst error correction algorithm is a kind of list decoding algorithm .In RiBC algorithm eight polynomials are updated simultaneously in each iteration. The architecture was tested with assembly test programs for the four decoding algorithms, which resulted in a throughput of 86 Mbps at a clock frequency of 200 MHz for a (171,133)-CC with rate 0.5, a throughput of 19 Mbps for the LTE turbo code with rate 1/3 and 128 information bits ,a throughput of 55 Mbps for the Wimax rate 3/4 LDPC code with 576 information bits and a throughput of 75Mbps for (7, 3) Reed Solomon code.

4. Derived Decoder Architectures

Starting with the architecture from section three, we modified the design such as to ”derive” decoder architectures that are dedicated to only decoding one type of code and are optimized with this goal in mind. The overall control structure remains the same, so that the resulting decoders are still programmable with assembly programs and these programs are executed in the same way as on the original architecture. Because of this, the decoding performance will be completely equivalent between the specialized decoders and the multi-mode decoder, which allows us an exact comparison.

A. Viterbi Decoder

The Viterbi decoder has the simplest data path of all decoder realizations .It implements the well-known Viterbi algorithm. The interconnect only has to perform the simple multiplexing of input values to the PEs. In the Gamma FU the branch metric calculation is executed and in the Alphabeta FU the state metric calculation. The trellis connection network feeds back the state metrics into Alphabeta FU. Note that it is connected to the other PEs as well, allowing to process 4x16=64 trellis states in parallel. Other active parts are the survivor memory LMEM and the traceback FU, of which there is only one in the decoder. It collects the survivor information from all four PEs to produce the hard-output values.

B. Turbo Decoder

For turbo decoding, the BCJR algorithm [9] is used, with parallel forward and backward recursions. This means that in the turbo decoder many FUs perform similar functions as they do in the Viterbi decoder. Branch metric and state metric computation in Gamma and Alphabeta FUs with trellis connection feedback is almost identical. Different is the function of the interconnect network, it is implemented as crossbar over all four PEs to allow the random memory access necessary for interleaving. The Lambda FU generates the LLRs from forward and backward metrics. LMEM is used as intermediate storage for state metrics. In the DALU FU the new extrinsic values are generated from the LLRs.

C. LDPC Decoder

LDPC decoding uses a layered offset-min-sum algorithm [8], so most central FUs are used differently than for turbo decoding. A big difference is the interconnect operation. The 4x4 crossbar is coupled with a barrel shifter stage to provide the circular matrix permutations that are required for aligning the message vectors associated with the submatrices of a parity-check matrix to the PEs. The Gamma FU is used to subtract extrinsic value from the current bit value. In the Alphabeta FU the minima search and LLR accumulation operations of the min-sum algorithm are performed. The new extrinsic value is recombined with the bit value in the DALU FU. LMEM is again used as intermediate storage.

D. Reed-Solomon (RS) coder

Reed-Solomon (RS) codes are concerned with the detection and correction of errors in symbols. RS codes are widely used for correcting the errors in storage and communication systems [10]. During transmission error may happen for a number of reasons e.g. (scratches on CD, radio frequency interference with mobile phone reception, noise etc.) At the receiving side, the decoder detects and corrects a limited predetermined number of errors occurred during transmission. When we are dealing with the RS codes as forward error correction, the errors are generated in transmission procedure are divided into burst errors, random

errors and erasures[11]. In this brief, a low-complexity high speed RS encoding and decoding architecture will improve the overall system performance significantly. In this a low complexity reformulated inverse free burst-error correcting (RiBC) algorithm is developed for practical applications. Then, based on the proposed reformulated inverse free burst error correcting algorithm, a unified VLSI architecture that is capable of correcting random errors, as well as burst errors and erasures, is firstly presented for multi-mode decoding requirements. It will be shown that, being the first RS Encoder and then decoder owning enhanced burst-error correcting capability, it can achieve significantly better burst error correcting capability than hard-decision decoding (HDD)

5. Derived Decoding Algorithm

Decoder proposed in this paper is based on the offset BP-based algorithm. The offset BP-based algorithm is a modified version of BP-Based algorithm , which has a tolerable performance loss compared to BP algorithm. We define some symbols first.

$L\ (c_i\ )$ is the intrinsic message, $L\ (\ r_{ji}\ )$ is the message computed by the check node j , and sent to the variable node i . $L\ (\ q_{ij}\ )$ is the message computed by the variable node i , and sent to the check node j . Definition of a_{ij} , b_{ij} are as follows:

$$a_{ij} = sign\ (\ L\ (\ q_{ij}\ ))$$
$$b_{\ ij} = abs\ (\ L\ (\ q_{\ ij}\ ))$$

Offset BP-based algorithm
Step 1) Initialization
Using the intrinsic messages initialize decoder.

$$L\ (\ q_{ij}\ ) = L\ (\ c_i\ )$$

Step 2) Check-Node Update(CNU)
Check nodes update log-likelihood ratio (LLR) in the following equation. *offset* is a value which can be computed by density evolution theory.

Step 3) Variable-Node Update(VNU)
Variable nodes update LLR in the following equation.

$$L(q_{ij}\ ) = L(c_i\ ) + \sum_{j'\in C_i \setminus j} L(r_{j'i}\ )$$

Also calculate

A. Input Buffers
Original LLR data are transferred to the input buffers, which consist of an array of RAMs. The number of RAMs is N , which is the maximum of column of supported base matrices. The depth of each RAM is Z , which is the expansion factor.

B. An Array of Node RAMs
The number of node RAMs is X , which is the maximum of nonzero submatrices of supported base matrices. The results of each CNU update or VNU update are written into these node RAMs for the next update.

C. Control Unit (CU)
Control Unit (CU) is the most important part of decoder. According to value stored in mode register, it configures other modules to adapt different modes. Main functions are as follows:

1) Controlling input state machine to adjust input numbers.
2) Reconfiguring three routers.(input buffers and node RAM, node RAM and CNU, node RAM and VNU)
3) Selecting the number of node RAMs which need to be updated, the number of CNU circuits and the number of VNU circuits.
4) Controlling output state machine to adjust output numbers.

$$L(Q_i\ ) = L(c_i\ ) + \sum_{j'\in C} L(r_{ji}\ )$$

if $L(Q\ ) > 0$, the result of hard-decision is 0,otherwise the result is 1.And if all the parity check constraints are satisfied or the maximum of iterations is reached, decoding algorithm turn to Step 4),otherwise turn to Step 2).

Step 4) Output
Output decoded data.

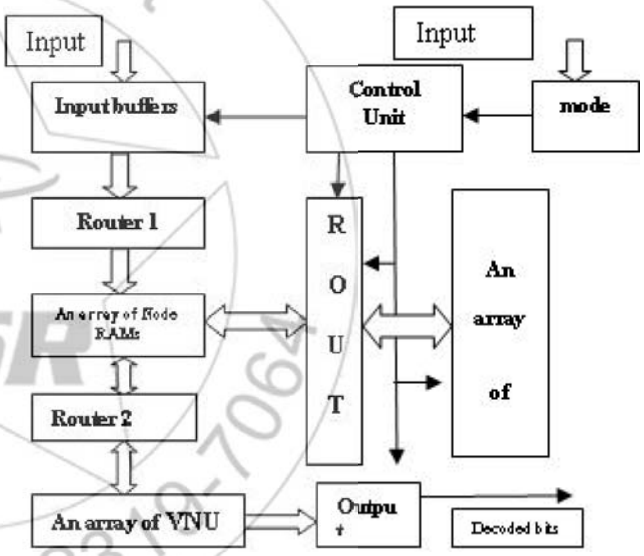

Figure 1: Block diagram of proposed decoder

D. CNU Circuit
Firstly, it finds out the minimum, the second smallest value and mark the position of the minimum. After modifying by offset value, the output port in the position of the minimum exports the second smallest value and the output ports in the rest position export the minimum. Finally, sign bit is attached to the output data. The pipeline technique is adopted to achieve higher throughput. The number of CNU input ports is maximal check nodes degree. To support different check nodes degree, the unused input of CNU is imported as the maximum value in fixed-point representation to produce the calculated data when the number of input data is less than the hardware input ports.

E. VNU Circuit
The number of VNU input ports is set to maximal variable nodes degree. And the unused pin of VNU is imported as zero value to produce the correct calculated data when the number of input data is less than the hardware input ports.

F. Output Buffers

The design of output buffers is similar to the design of interleaver. The VNU circuits output *N* bit hard-decision values of LLR at a time. And then the decoded information bits are written into output buffers by row. Once one frame data are stored in the output buffers, the decoded bits now are read from output buffers by column. After previous described procedure, the decoding process is accomplished.

6. Conclusion

In this paper, a multi-mode QC-LDPC decoder design method and relative architecture are proposed. This multi-mode QC-LDPC decoder is base on a partially parallel architecture and the offset BP-based algorithm is employed. Any type of QC- LDPC codes with the same expansion factor can be supported and at least 3 different QC-LDPC codes can be configured into this unique architecture. There is however a heavy penalty on the power consumption of the multi-mode decoder, which leads to the conclusion that it only makes sense to employ a multi-mode solution in an environment that is not completely focused on low-power. An obvious example would be a base station in mobile communications.

References

[1] F. Naessens, B. Bougard, S. Bressinck, L. Hollevoet, P. Raghavan, L. van der Perre, and F. Catthoor, "A unified instruction set programmable architecture for multistandard advanced forward error correction," in *Proc.IEEE Workshop on Signal Processing Systems (SiPS'08)*, Washington, DC, Oct. 2008, pp. 31–36.

[2] G. Gentile, M. Rovini, and L. Fanucci, "A multi-standard flexible turbo/LDPC decoder via ASIC design," in *Proc. International Symposium on Turbo Codes and Iterative Information Processing (Turbo Coding'10)*, Brest, France, sep 2010.

[3] Y. Sun and J.R. Cavallaro, "Unified decoder architecture for ldpc/turbo codes," in *Proc. IEEE Workshop on Signal Processing Systems (SiPS'08)*, Washington, DC, Oct. 2008, pp. 13–18.

[4] M. Alles, T. Vogt, and N. Wehn, "FlexiChaP: A reconfigurable ASIP for convolutional, turbo, and LDPCcode decoding," in *Proc. Int. Symposium on Turbo Coding (TURBO CODING'08)*, Lausanne, Switzerland, Sept. 2008, pp. 13–18.

[5] F.M. Li, C.H. Lin, and A. Wu, "Unified convolutional/ turbo decoder design using tile-based timing analysis of va/map kernel," *IEEE Trans. VLSI*, vol. 16, no. 10, pp. 1358–1371, Oct. 2008.

[6] J.R. Cavallaro and M. Vaya, "Viturbo: A Reconfigurable Architecture For Viterbi and Turbo Decoding," in *Proc. Int. Conf. on Acoustics, Speech and Sig. Proc. (ICASSP'03)*, Hong Kong, Apr. 2003, pp. 497–18.

[7] S. Kunze, T. Kobori, E. Matuˇs, and G. Fettweis, "A "Multi-User" Approach Towards A Channel Decoder For Convolutional, Turbo And Ldpc Codes," in *Proc. Int'l Workshop on Signal Processing Systems (SIPS'10)*, San Jose, CA, USA, Oct. 2010.

[8] J. Chen, A. Dholakia, E. Eleftheriou, M.P.C. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[9] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 248–287, Mar. 1974.

[10] B.Sklar, " Digital Communication, Fundamental and Application" Prentice Hall, Upper Saddle River, 2001.p.1104.

[11] S.B.Wicker and V.K. Bhargava, eds "Reed-Solomon codes and their applications" New york: IEEE press 1994.