International Journal of Engineering Innovation & Research
Volume 1, Issue 6, ISSN: 2277 – 5668

Design & Implementation of Flexible Multi-Standard Turbo/LDPC Kernel

Abha Jain, Dr. Ashwani Singh

Abstract: **Modern wireless communication systems are designed to be multi-mode and use different algorithms to implement forward error correction decoding. Sharing of Datapath and Memories across different FEC families and also across different standards of one FEC family are of paramount importance. In this paper we analyze the hardware reuse of datapath across different possibilities of the implemented architecture. First a serial implementation scheme of Min-Sum algorithm for LDPC decoding is explored over Max-Log-Map based turbo decoder datapath architecture. The implementation results are presented, showing that different possibilities may at most give a reduction of around 14% in combined datapath area. Secondly, We propose a novel Parallel implementation scheme for Min-sum algorithm which fits very well with underline turbo decoding architecture, and show that the proposed scheme not only provides significant speed-up in terms of clock cycles for VN updates, but also results in a reduction of around 24.5% in combined datapath area.**

Keywords: **LDPC Codes, Turbo Codes, VLSI, Channel Decoder**

I. INTRODUCTION

Support of multi modes in emerging wireless standards has been pushing the need for flexible devices. Flexibility is defined in terms of ability of the hardware platform to adapt to the different algorithms and its reconfigurability towards different scale of an algorithm[1]. LDPC codes and turbo codes are among the known near Shannon limit codes that can achieve very low bit error rates for low signal-to-noise ratio (SNR) applications [2], [3]. A channel decoder architecture, which can operate as different individual decoders, supporting different FEC codes and their extensions has obvious advantage over conventional devices in terms of seamless switching across different modes and may also result in smaller area.

In recent years many research activities have emerged proposing multistandard ASIPs implementations in order to achieve flexible and high throughput parallel iterative decoding. In [4] a flexible and high performance ASIP model for turbo decoding was proposed which can be configured to sup- port all simple and double binary turbo codes up to eight states and in [5] a memory sharing across turbo and LDPC code ASIP was explored. In [6] author proposed a reconfigurable ASIP supporting different codes across different FEC families like Convolutional, Turbo and LDPC codes where datapath reuse across different FEC families has been used however at a macroscopic level of decoding structure.

In this paper, we first present an analysis on different posibilities of the datapath sharing across a Max-Log-Map based Turbo decoding kernel and Min-sum based LDPC decoding kernel capable of supporting different wireless standards. The Analysis explores the VLSI implementation complexity of a this joint datapath supporting different block length, code rates, constraint length and polynomials proposed in emerging wireless standards. The Datapath reuse is looked into at different hierarchy of platform design starting from basic ACS (add compare select logic) level. Secondly, we present a novel implementation scheme of min-sum algorithm, providing the designer with a higher reuse of underline turbo decoding infrastructure, as well as a faster check node processing. We believe such an analysis can be a basis for SoC designer for choosing a right fec kernel for designing high performance flexible decoders .

This paper is divided into VI sections. After a brief review of Multistandard FEC system and target FEC decoding algorithms in Section II, the general architecture of the FEC Kernel is discussed in Section III; the datapath reuse implementation results in case of serial implementation scheme of Min-Sum decoding over a turbo platform is presented in Section IV and then in Section V a new parallel implementation scheme for Min-Sum algorithm is proposed, along with its implementation complexity. Finally, Section VI provides the conclusion and future perspective of our work.

II. MULTISTANDARD FEC SYSTEM

The need to incorporate Turbo decoder (Single and Duo- binary), and LDPC decoder inside a single hardware demands a judicious sharing of logic and memory units. To formalize, this association can be split into two levels, logic sharing and Memory sharing. As different frame length of these codes in different standards, have big impact on memory sizes, it is logical to separate the standard dependent memory banks and common FEC kernel. Figure 1 shows the block diagram of such a generic decoder. An algorithm is partitioned into two segments: Control segment and the processing segment. FEC algorithms show an parallel processing segment that is easily map-able over an processing array. The control and the scheduling part is handled by the controller. In turbo decoding the processing steps are the basic Soft Input Soft Output (SISO) decoders which implements the BCJR algorithm [7]. The decoding of the binary and duo-binary Turbo codes is performed using the multi window Max-Log-Map version of BCJR algorithm [8].

While considering that (N, K) be a binary LDPC code described by a sparse parity check matrix H of size MxN, where M=N-K is the check nodes corresponding to the parity checks of a bipartite graph and N is the bit nodes corresponding to encoded bits. For the standards in

the scope of our work the matrix is designed such a way that every block column in a layer has zero or one non zero element. Each bit node is connected to d_c check nodes and each check node is connected to d_v bit nodes considering a regular code.

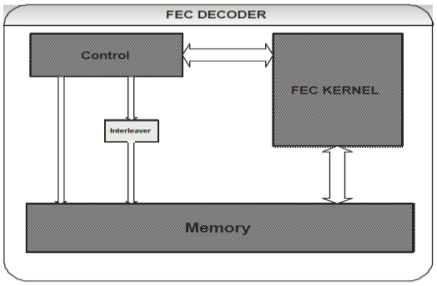
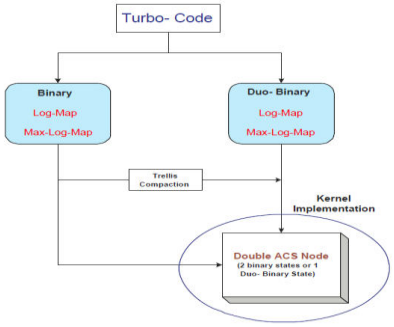
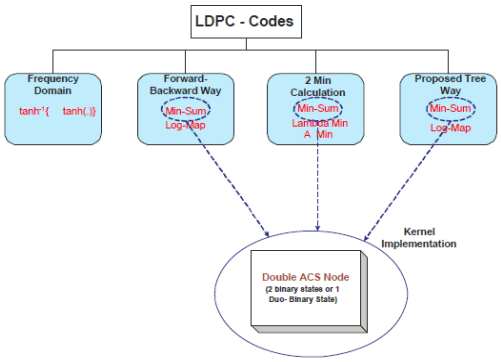


Fig. 1. Multistandard FEC Decoder.



(a) Basic processing unit for proposed Min-Sum Algorithm



(b) Basic processing unit for Min- Sum and Turbo processing

Fig. 2. Processing elements for Min-Sum and Max-Log-Map Processing

The Sum Product algorithm consists of an iterative process involving check nodes and bit nodes updates. Two phase decoding has recently given way to the so called layered or shuffled decoding[9][10]. The key point of which is the overlap of the traditional VN and CN phase, as a result of which each iteration can be viewed as the concatenation of several layered sub iterations. After the layered sub iteration reliability of the bit in the received codeword is immediately available to the next layer, which will work on this new information. This results in approximately two times faster convergence of the algorithm. In our implementation we use the Min-Sum algorithm for CN update process [11].

III. TURBO/LDPC KERNEL

Figure 3 depicts the zoom in view of the multistandard kernel is capable of handling upto 16 states binary turbo codes and 8 states duo-binary turbo codes. Each Double ACS/ CNP unit caters to 4 Trellis transitions. Hence the maximum trellis transition parallelism is 32, which very well supports the requirements of current and emerging standards. One Double ACS / CNP unit is assigned to each state (2 states for binary codes) and the interconnection (ACS Network) between these units is established according to trellis diagram of the code. The most critical factor that affects the processing efficiency is the communication scheme of moving data between Double ACS / CNP units. Since the reconfigurable platform supports different communication standards, multiple trellis states with different communication requirements needs to be supported. In case of LDPC decoding mode 8 check node process can be processed in parallel each of them implementing Min- Sum algorithm in sequential fashion. Total number of Check Node processed by this unit depends on the size of SM memory banks. Given the size of these memory has depth W (Window Size in case of turbo decoding), maximum check node processed by the kernel is 8×W. The Kernel could be reused for Partial LLR (log Likelihood Ratio)computation (to perform certain stages of ACS operations of LLR computation). For Lower states code(4,8) alpha and beta recursions are performed in parallel. Though we are not dealing with memory sharing across LDPC and Turbo cases in this work a little example of possible memory reuse could be seen in the figure 3. The calculated minimum and second minimum values for the LDPC algorithm are stored in the memory banks used for storing the state metric values during the forward and backword recursions. The branch metric memory banks of the Turbo decoding case is reused for storing the min index, and sign of minimum information in the LDPC case.

FEC kernel. In our implementation the platform consists of 8 processing units called as Double ACS / CNP unit. The

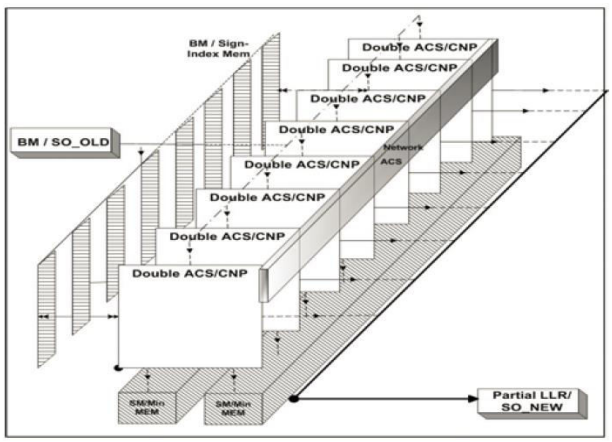


Fig. 3. Multistandard Turbo/LDPC Kernel.

A. ACS Network

The interconnection between Double ACS / CNP units is established according to trellis diagram of the code. Different permutation of interconnection are possible for different number of trellis states *m* and for forward (A_k) and backward (B_k) states computation for any information value *i* as shown in Fig 4. For the forward recursion network layout is organized as debrujin graph . Such a layout for state 4,8,16 binary turbo code is implemented, similarly the layout derived for backward recursion is also implemented. Further more for duo binary turbo code 8 state forward and backward trellis interconnection network is also implemented. Our manual mapping approach results in common interconnection between different scales of the code implemented. As mentioned in previous section the Kernel could be reused for Partial LLR computation as shown in Fig 4 , such a reuse necessitates extra mapping of permutations on the ACS network element of the kernel. However taking into account forward and backward trellis computations for all the codes supported in current standards and possibility of reuse as partial LLR computation, the designed network still occupies 60 % less area in terms of multiplexers used ,compared to fully interconnected network as designed in [4].

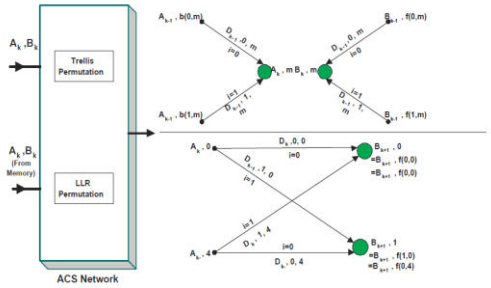


Fig. 4. ACS Network supporting Trellis and LLR permutations

B. Double-ACS / Check Node Processor

A functional block diagram of Double-ACS / Check Node unit is shown in Fig 5. This unit performs all the forward and backward state metric computation and Partial LLR calculation in case of Turbo codes. In case of LDPC codes Soft output based check node processing is implemented [12], [13]. The turbo decoding part of the Double-ACS / Check Node unit acts as a Min finder to compute two minimum value needed for check node processing. Each Double-ACS/Check Node unit contains 4 adders and 3 compare select (CS) logic.

The Min finder block does the major logical processing in Double-ACS/Check Node unit. As Shown in Fig 5 input multiplexers choose the data for processing in turbo or LDPC cases respectively. In case of binary turbo case only two CS blocks are active processing a turbo code trellis butterfly, while for duo-binary turbo case this PU processes one state using the three CS unit. For the LDPC case the Min finder resources are fully reused for calculation of two min values (used in MS algorithm for CN updates generation), only two CS blocks are active and decision output of first CS block is further used for second minimum calculation. All the inputs are processed sequentially. Following pseudo code shows the behavioral algorithm of the Min finder in LDPC case.

```
IF {Magnitude < MIN}
        MIN = Magnitude
        MIN1 = MIN0
ELSE
        IF {Magnitude < MIN1}
                MIN1 = Magnitude
        ENDIF
ENDIF
```

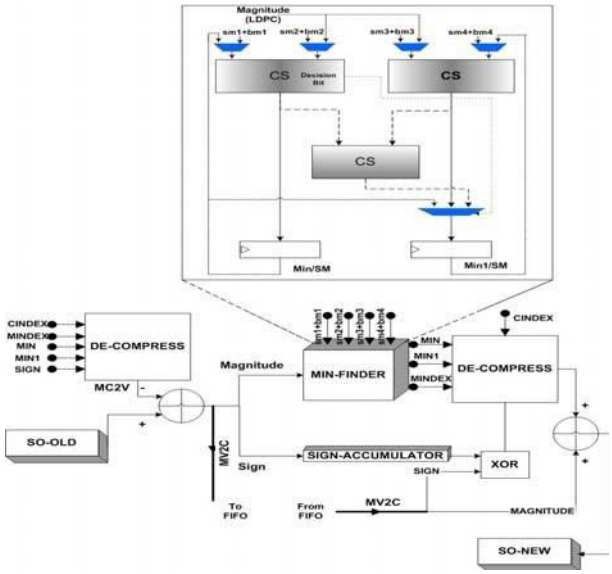


Fig. 5. Double-ACS / Check Node Processor

However apart from Min Finder block LDPC case requires computations like : compress unit to obtain MC2V for VN to check message computation, Two's compliment to Signed magnitude conversion, min index updating and product of sign calculation, compress unit to obtain MC2V for CN to VN update. Compress block uses Equality check operation across CINDEX and MINDEX and a multiplexer for choosing MIN or MIN1 based on the output of equality check operation. Finally the sign bit is added and fed to Signed magnitude to Two's compliment conversion block to generate MC2V message.

C. Re-use Scenario and Implementation results

The different architecture scenarios were compiled for a Virtex XCV1000-6 device. The idea is to evaluate vlsi complexity of reusing the data path at different implementation level. Internal data width for both Turbo and LDPC datapath implementation is chosen at 8 bits. Different implementation scenarios are categorized into following cases:

CASE1: Turbo double ACS unit was implemented and synthesized (as it is the basic process in forward (A k), backward(Bk) states and LLR computations.

CASE2: LDPC Min Finder unit was implemented and synthesized.

CASE3: Turbo double ACS unit hardware reused as Min Finder.

CASE4: Check Node Processor was implemented and synthesized.
CASE5: Complete Check Node Processor with Min finder block reused as Turbo Double CS was implemented and synthesized.
CASE6: Complete Check Node Processor with Min finder block reused as Turbo Double CS also the two adders in CNP and 2 adders in decompress units reused for SM+BM addition in turbo case.

Table I Synthesis Result Comparison.

Scenario	H/W [Gates]	reused H/W [Gates]	Change(%)
CASE1	891		
CASE2	392		
CASE3	1283	1106	−13.7
CASE4	834		
CASE5	1725	1560	−9.6
CASE6	1725	1881	9

The results are presented in Table I. It can be seen that the reuse scenario at very microscopic level of CASE 1 gives a reduction of around 14% in area compared to separate dedicated datapath of turbo and LDPC case. A further effort on resources sharing at the higher level results lesser area gain and in some cases an increase in area possibly because of increase in the control logic for handling the different cases.

IV. Parallel Min-Sum Scheme

Motivated by the results in previous section we propose a parallel implementation scheme of check node update, the implementation scheme is fine tuned keeping in mind the BCJR computation kernel of Turbo decoding. The effort is to map check node processing across the kernel parallely. As seen in Fig 1 we have 8 double acs units i.e. 16 acs units. we can use the resource for processing a check node with maximum dc(check node degree) of 32. This takes care of almost all the current wireless standards. However proposed scheme is very generic and can be easily extended for any value of dc. Check node with dc degree are mapped over 8 double- acs units calculating the extrinsic VN in parallel. For the sake of architecture uniformity odd d c are considered as their even counterpart with extra SO initialized at +∞ (i.e. d c'=dc if dc is even; else dc'=dc+1). Other than sign accumulation which is performed by separate XOR tree(not shown in Fig 6), in the proposed scheme VN extrinsic calculation consists of three category of stages: Direct VN comparison (DVC) stage, Multiple Shuffled comparison (MSC) stages and two Extrinsic Calculation(EC) stage. The first and the last category of stages take 1 and 2 clock cycles respectively for all values of d c . The total clock cycles Ncc required to calculate extrinsic VN values (e1, e2,e3...etc) for check node degree d c'can be calculated offline as following:

X=0;
For (i=dc'-2; i>=2; i--)
{
X+= (log2 i);
i= i-2^X;
}
Ncc=X+2;

A. Direct VN comparison (DVC) stage

The dc old SO values are fed to the first stage. As seen in Fig 6 for dc '=8 the old SO values (i1,i2, ...i8) are fed to two parallel double acs units (i.e. 4 CS units). in case of dc=7 , i8 is initialized as +∞. for all values of dc there is only one direct comparison stage. The output of each stage is passed on to next stage as well as stored in state metric memory for use in later stages. Operations in this stage are performed in one clock cycle.

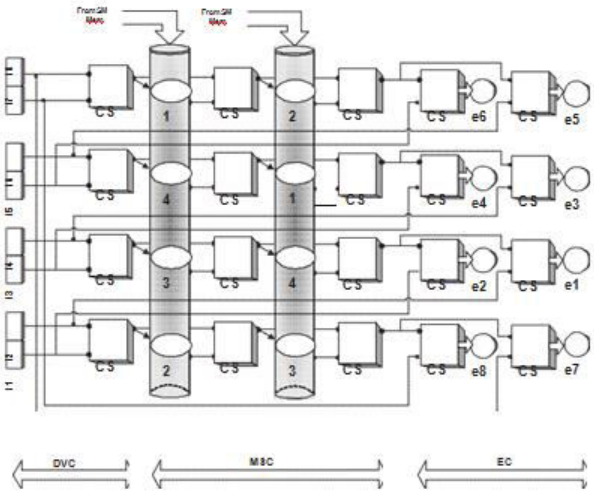


Fig. 6. Proposed Parallel Min Sum Calculation Scheme

B. Multiple Shuffled comparison (MSC) stage

Shuffled comparison stage could be compared to the trellis computation stages in BCJR algorithm for turbo decoding. Input to the CS block is the shuffled output from the previous stages. The shuffle network implements a circular shifting permutation, which can be easily mapped on to acs network of Fig 3 without significant hardware cost. The rotational shift depends on d_c' and the substage of shuffled comparison stage. There are multiple shuffled stages depending on d_c' and equal to N_{cc}-3. For different values of d_c' Table II provides the information on N_{cc} values and shift associated with each shuffled stage. These permutations are stored in similar way as trellis configuration in case of turbo decoding for different codes.

Table II Clock Cycle Requirement And Shift Permutation

dc	CC	Shifts Permutations	dc	CC	Shift Permutations
5,6	4	1	19, 20	7	1, 3, 5, 9
7,8	5	1, 3	21, 22	7	1, 3, 5, 9
9,10	5	1, 3	23, 24	8	1, 3, 5, 9, 11

11,12	6	1, 3, 5	25, 26	7	1, 3, 5, 9
13,14	6	1, 3, 5	27, 28	8	1, 3, 5, 9, 13
15,16	7	1, 3, 5, 7	29, 30	8	1, 3, 5, 9, 13
17,18	6	1, 3, 5	31, 32	9	1, 3, 5, 9, 13, 15

As can be seen from Fig 6 the input to the shuffle network is either from the immediate previous stage output or earlier stage output stored in the SM memories. No matter what is the source to shuffle network input the shift value remains the same for a given stage. However, for a given stage the address to be accessed in SM memory depends on d_c' value and can be derived using relatively simple control. Fig 7 shows the memory access pattern for d_c' upto 32. The vertical axis represent the time in clock cycles. Vertical binary level 10 represent DVC stage output, while 100 and 1000 represent first two MSC stage outputs stored in SM memory. The bottom horizontal row shows the binary representation of value d_c'-2. For a given d_c' the address can be derived directly from corresponding binary value in the bottom most layer. For example for d_c'=16, corresponding binary representation of d_c-2 = 1110 i.e. 1000+100+10, thus address for memory access during the shuffle stage correspond to value at binary level of 100 and 10.

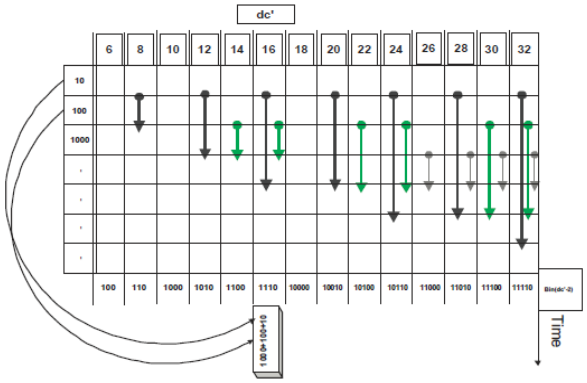


Fig. 7. Memory Access during Shuffle Stage

C. Extrinsic Calculation (EC) stage

The last two stages for any value of d c' are extrinsic calculation stage for the VNs i1,i2...i . As seen in Fig 6 input to these stages is the output from the last MSC stage and shifted SO old values of VN. This shift is circular over dc' and equal to 1 and 2. The two stages are processed in 2 clock cycles.

D. Implementation Results

The different architecture scenarios were compiled for a Virtex-IV XC4VFX140-11 device. It can be seen that the basic processing unit (LDPC PU) for proposed Min-Sum Algorithm has very comparable area requirement to double ACS for turbo codes. A combined unit handling both the processing (TURBO-LDPC PU) results in an area saving of 12.5 %. However as we go up into the hierarchy of the implementation we see that a combined Turbo LDPC Kernel (TL Kernel) results into a saving of 24.5 % in area compared to two independent datapath for LDPC and Turbo codes.

Table III Synthesis Result Comparison For Proposed Implementation

Scenario	H/W [Gates]	reused H/W [Gates]	Change(%)
Turbo Double ACS	891		
LDPCPU	851		
TURBO-LDPC PU	1742	1523	−12.5
TURBO-KERNEL	16843		
LDPC-KERNEL	12166		
TL-KERNEL	29009	21899	−24.5

V. CONCLUSION

We presented a VLSI complexity analysis of datapath sharing across two FEC code families viz. Turbo and LDPC. Various implementation possibilities of Min-Sum based LDPC decoding over a Turbo BCJR core was explored. The implementation results are presented, showing that different possibilities may at most give a reduction of around 14% in combined datapath area. In addition to this, we proposed a new scheme of parallel implementation for Min-sum algorithm which fits very well with underline turbo BCJR core and results in a reduction of around 24.5% in combined datapath area. The proposed scheme is efficient in terms of clock cycles required for VN update and results in a high throughput kernel. The future work will deal with exploring the memory reuse possibilities for these two fec families.

REFERENCES

[1] Polydoros, A., "Algorithmic aspects of radio flexibility," Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on , vol., no., pp.1-5, 15-18 Sept. 2008

[2] R. G. Gallager. Low-Density Parity-Check Codes, IEEE Transactions on Information Theory, Jan. 1962, pp. 21-28.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima, Near Shannon limit error- correcting coding and decoding: Turbo-codes, in Proc. ICC93, Geneve, Switzerland, May 1993, pp. 1064-1070.

[4] Olivier Muller, Amer Baghdadi, Michel Jzquel. ASIP-Based Multiprocessor SoC Design for Simple and Double Binary Turbo Decoding, in Proc. 2006 Design, Automation and Test in Europe (DATE 06), Munich, Germany, Mar. 2006.

[5] Alles, M., Vogt, T., Wehn, N., "Flexi ChaP: A reconfigurable ASIP for convolutional, turbo, and LDPC code decoding," Turbo Codes and Related Topics, 2008 5th International Symposium on , vol., no., pp.84-89, 1-5 Sept. 2008

[6] Naessens, F.; Bougard, B.; Bressinck, S.; Hollevoet, L.; Raghavan, P.; Van der Perre, L.; Catthoor, F., "A unified instruction set programmable architecture for multi-standard advanced forward error correction," Signal Processing Systems,2008. SiPS 2008. IEEE Workshop on , vol., no., pp.31-36, 8-10 Oct. 2008

[7] 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. 284-287, Mar. 1974.

[8] A. Giulietti, L. van der Perre, and A. Strum, "Parallel turbo coding interleavers: avoiding collisions in accesses to storage elements", Elec. Lett. , vol. 38, no. 5, pp. 232-234, Feb. 2002.

[9] D. Hocevar, A reduced complexity decoder architecture via layered decoding of LDPC codes, in Signal Processing Systems, 2004. SIPS 2004. IEEE Workshop on, (Austin, USA), pp. 107112, Oct. 2004.

[10] M. Mansour and N. Shanbhag, Low-power VLSI decoder architectures for LDPC codes, in Low Power Electronics and Design, 2002. ISLPED02. Proceedings of the 2002 International Symposium on, (Monterey, USA), pp. 284-289, Aug. 2002.

[11] M. Fossorier, M. Mihaljevic, and H. Imai, Reduced complexity iterative decoding of low-density parity check codes based on belief propagation, IEEE Transactions on communications, vol. 47, pp. 673-680, May 1999.

[12] T. Brack, M. Alles, F. Kienle, and N. Wehn, A synthesizable IP core for WIMAX 802.16e LDPC code decoding, in Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium on,(Helsinki, Finland), pp. 1-5, Sept. 2006.

[13] M. Rovini, F. Rossi, P. Ciao, N. LInsalata, and L. Fanucci, Layered decoding of non-layered LDPC codes, in Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006. 9th EUROMICRO Conference on, (Dubrovnick, Croatia), pp. 537-544, Sept. 2006.

AUTHORS PROFILE

Abha jain received the B.Tech degree in Electronics & Communication (RGPV BHOPAL) in 2004 and the M.Tech. Degree in Digital communication (RGPV BHOPAL) in 2010,and working as an Asst. Professor in Electronics & Communication Dept. at Radharaman Institute Of Technology & Science, Bhopal.

Dr. Ashwani Singh received dual Ph.D. degree from Politecnico di Torino, Torino Italy and University de Bretagne Sud, Lorient France. He holds a Master Degree from Saint' Anna School of Advanced Studies, Pisa Italy and Bachelor in Engineering from NIT Bhopal, India. His research and development experience includes over 8 years in the Industry and Academia in India and different European countries. He has authored/co-authored several refereed journal/conference papers.