Datapath Reuse in a Multi-Standard FEC Kernel

Abha Jain,
Department of Electronics,
Radharaman Institute Of Technology & Science, Bhopal, India.
Email: abha_jain23@yahoo.co.in

Dr. Ashwani Singh
Helium Infotech Private Limited
Ghaziabad, India
Email: singh.ashwani@gmail.com

***Abstract*—Sharing of datapath and memories across different forward error correction (FEC) decoder implementations are important in flexible wireless communication system design. In this paper, we explore datapath reuse possibilities across some important FEC families like convolutional, turbo and low density parity check (LDPC) codes. At first, design of a reduced complexity trellis network for shuffling the updated state metrics in Viterbi and Turbo decoding kernel is presented. Our design results in complexity reduction of upto 70 % compared to trellis networks implemented in state of art multi standard FEC kernels. Furthermore, the hardware reuse potential of different existing implementation schemes for check node processing in LDPC decoding is explored over such a FEC kernel architecture. In the last part of the paper, we propose a novel parallel implementation approach (”Tree-Way”) for check node processing, which fits very well with underline FEC Kernel architecture. Implementation results are presented showing that the proposed scheme provides significant speed-up in terms of required clock cycles without significant increase in combined datapath area compared to existing approaches.**

***Index Terms*—LDPC codes, Turbo Codes, Min-Sum Algorithm, Max-Log-Map Algorithm, VLSI Decoder Architecture.**

I. Introduction

Support of multi modes in emerging wireless standards has been pushing the need for flexible devices. Among the different functionalities of such standards, one of the most demanding operations is channel decoding. Convolutional codes (CC), turbo codes and low-density parity-check codes (LDPC) are established channel coding schemes. However, multiple variations of these algorithms are employed in standards. Every standard uses a different configuration of an algorithm which makes it unique to that standard. For example, the turbo coding algorithm used in UMTS employs a different polynomial, block size, and coding rate from that used in WiMAX. This translates to a further increase in complexity of a flexible channel decoding platform.

In last few years many research activities have emerged proposing implementations in order to achieve flexible and high throughput decoder architectures for supporting the flexibility across different FEC code families. Approaches combining the turbo and Viterbi decoding have been reported in [1], [2] etc. A unified decoder architecture for LDPC and turbo codes has been presented in [3] where multi-mode decoding is achieved by employing a flexible add-compare-select (FACS) unit. By representing LDPC codes as parallel concatenated single parity check (PCSPC) codes, authors have tried to efficiently reuse the turbo decoding infrastructure for LDPC decoding functions.

In the later years when focus was shifted towards application specific instruction-set processor (ASIP) architectures, authors in [4] presented the FlexiTreP ASIP family which supports trellis based channel codes. In [5] a memory sharing across turbo and LDPC code in an application specific processor (ASIP) was explored. The FlexiTreP core is merged with an LDPC ASIP core into a single ASIP, named FlexiChaP, which is capable to support CC, turbo codes, and structured LDPC codes. It was shown that because of the efficient memory sharing across turbo and LDPC decoding, area increase in FlexiChap is very small compared to FlexiTrep ASIP. However sharing of datapath logic area (which is about 25% of the complete ASIP) across these two families were considered insignificant. Very recently in [6] authors proposed the application-specific instruction programmable architecture addressing in a unified way the emerging turbo and LPDC coding requirements of 3GPP-LTE, IEEE802.11n, IEEE802.16(e) and DVB-S2/T2 where datapath and memory reuse across different FEC families has been used.

From the analysis of the state of the art in hardware reuse in flexible channel decoder design two important points can be gathered. First, memory sharing across different codes definitely is the bigger player in overall reuse scenario and several works present efficient technique for memory reuse. However, datapath logic which could be up to 25% of the whole decoder complexity do present an interesting area of exploration for hardware reuse, mostly because of a multiplicity of ways to implement different FEC decoding algorithms. Secondly, it can be seen that a comprehensive analysis of datapath reuse providing the designer with the choice to decide on the different possible ways of implementing basic FEC computation units is still lacking in existing works. Motivated by these observations, in this paper we present some new results in datapath hardware reuse across different FEC code families.

This paper is divided into VI sections. A design space of flexible channel decoder implementation is explored in section II highlighting the problem subset dealt in this work. A general architecture of the turbo decoding kernel and its reuse for implementing Viterbi decoding algorithm is presented in Section III with special focus on reduced complexity trellis networks for shuffling the updated state metrics. Section IV presents a short description of different check node architectures used for LDPC decoding and the datapath reuse scenarios in case of their mapping on turbo decoding kernel. In Section V a new ”Tree Way” approach for check node update in

LDPC decoding is proposed, along with comparison of ASIC synthesis results for different check node architectures mapped onto turbo decoding kernel. Finally, Section VI concludes the paper.

II. Flexible FEC decoder design space

The implementation of complex (iterative) channel decoding systems became essential to reach the performances now required in term of quality of transmission. Dedicated hardware architectures (i.e. ASIC) implementing parts of these systems are already been dealt with in many existing works. However, the requirements of: very low error rate, very high throughput and increased flexibility of the implementation, create the need for adequate multiprocessor architectures. In this context, Multi-Processor System-on-Chip (MPSoC) architectures are being widely investigated in recent years. A MPSoC based flexible decoder design space is shown in Figure 1. Here application specific processing element design capable of supporting different channel decoding algorithms and their different scales constitute challenging subset of flexible channel decoder architecture design problem (shown in Figure 1). It can be seen that, other than the different codes to be supported, the system specification requirement of the required decoder throughput is one of the important factors to keep in mind while designing the processing element. Furthermore due to the iterative exchange of data between processing elements each processor working on the same data block has to communicate with each other, yielding only limited locality. A communication network has to support the communication demands of the different processing elements without degrading the throughput of the overall system, which is the other important subset of the flexible decoder design problem space as seen in Figure 1.

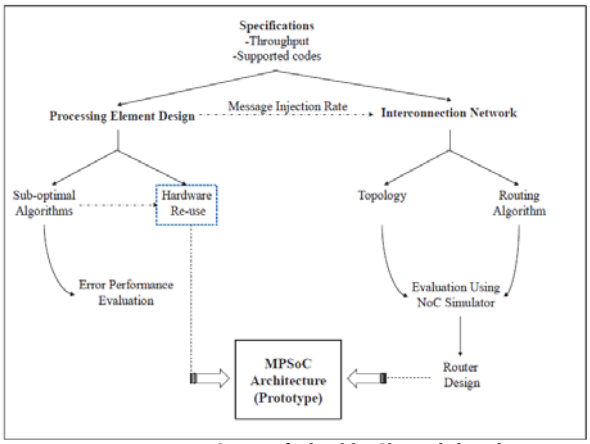

Fig. 1. Design Space of Flexible Channel decoder

The need to achieve high throughput pushes the demand for sub-optimal algorithms design of related code encoding and decoding procedures for low latency implementation, which however need to be efficiently validated for their error performance robustness. The other problem which needs to be explored while designing a flexible processing element is the reuse of hardware resources (datapath, memories) across different decoding algorithm implementations (highlighted by dotted box in Figure 1). The work in this paper focuses on this problem set, more specifically on the datapath reuse part as reuse of memory components has already been studied across several standards, some new results are presented in the following sections.

III. A FEC Kernel for Turbo and Viterbi decoding

In turbo decoding the processing steps are the basic Soft Input Soft Output (SISO) decoders which implement the BCJR algorithm [7]. Decoding of the binary and duo-binary turbo codes is performed using either Log-Map or the Max-Log-Map version of BCJR algorithm. In the Log-MAP algorithm, the function used to compute the forward (α), backward (β) and log-likelihood ratio (LLR) output metric is given as:

$$f(a, b) = max(a, b) + \ln(1 + e^{-|a-b|}). \quad (1)$$

On the other hand Max-Log-MAP algorithm is a simpler sub-optimal version of Log-MAP and could be obtained simply by discarding the correction factor in equation (1). Fig. 2 depicts the architecture of a multi standard turbo kernel which is quite similar to the one proposed in [8]. We have followed multiprocessor approach to implement reconfigurable FEC kernel where each processor performs state computation operation. Moreover, within a particular type of channel coding, the code rates and polynomials are also variable, this translates to a need of high flexibilty inside the trellis kernel. The kernel consists of 8 processing units named Double-ACS (D-ACS) as the basic building block of Max-Log-MAP algorithm implementation. Each of these units contains 4 adders and 3 Compare Select (CS) elements as shown in Fig. 4.(a) in next section. The designed kernel has two operating modes. In *mode1*, it performs all the forward and backward state metric computation and the results are stored in State Metric (SM) memory banks, while in *mode2* it can also perform certain stages in ACS operations of LLR computation calculation (partial LLR). The architecture can process binary turbo codes directly or reusing duo-binary trellis, thanks to trellis compaction [9]. One D-ACS unit is assigned to each state (2 states for binary codes) and the interconnection between these units is established by means of a network (ACS Network) that maps multiple trellis diagrams for different turbo codes. Each D-ACS unit caters to 4 trellis transitions; hence the maximum trellis transition parallelism is 32, which very well supports the requirement of current standards as shown in Table I and Table II. For a more efficient reuse of the computing resources, for lower states code (4,8) forward and backward recursions are performed in parallel.

By analyzing the concepts and the architectures of the convolutional and the turbo code decoders, some circuits sharing techniques are applied to merge the main functions into one decoder. Until now some approaches combining the turbo and Viterbi decoding have been reported in [2] and [10].

TABLE I
FEC KERNEL PARAMETERS IN DIFFERENT STANDARDS FOR CC CODING

Standard	Code Rate	States	Throughput (Mbps)
UMTS	1/2, 1/3	256	0.064
CDMA 2000	1/2, 1/3	256	0.038
DVB	1/2	64	32
DAB	1/4	64	1.1
WiLAN	1/2	64	54
GSM	1/2	16	0.0096

TABLE II
FEC KERNEL PARAMETERS IN DIFFERENT STANDARDS FOR TURBO CODING

Standard	Code Type	States	Throughput (Mbps)
WiMAX 802.16e	Duo-Binary	8	70
DVB-RCS	Duo-Binary	8	31
UMTS	Binary	4	2.3
CDMA 2000	Binary	8	2
3GPP2	Binary	16	-

The fundamental fact utilised in these works about the datapath sharing between turbo and Viterbi decoding is that both Viterbi and Max-Log-Map algorithms work with add-compare-select (ACS) operations. In our kernel implementation, for Viterbi decoding case, 2 states are mapped to one D-ACS unit. The platform is capable of handling up to 16 states CC as fully parallel architecture, for higher numbers of states (e.g. 32-256) trellis operations are mapped to one D-ACS following the approach mentioned in [11]. Reuse of this kernel for LDPC decoding functionality will be presented in the next sections.

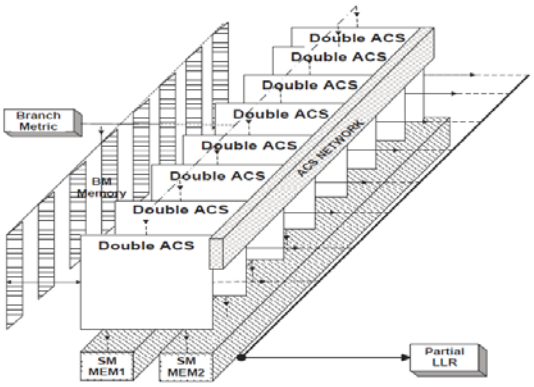

Fig. 2. Multistandard Turbo Kernel.

The most critical factor that affects the processing efficiency is the communication scheme of moving data between D-ACS units. Since the reconfigurable platform supports different communication standards, multiple trellis states with different communication requirements need to be supported. Furthermore as in *mode2* platform is reused for partial LLR computation, such a reuse necessitates extra connections on the ACS network element of the kernel which is detailed in following sub section.

A. ACS Network Reuse

As mentioned in previous sub-section, interconnection between D-ACS units is established according to trellis diagram of the code. As shown in Table I and Table II, in mobile and wireless communication systems the constraint length of convolutional codes varies typically between (5-9) implying states to be supported as 16-256. Convolutional codes are used as component codes for turbo codes used in emerging wireless standard. Here the constraint length typically varies between 3-5, implying states to be supported as 4-16. In addition to this, WiMax (802.16e) and DVB standards use duo-binary turbo codes where each state in the trellis has four incoming and four outgoing branches. Since the reconfigurable platform supports different communication standards, multiple trellis states with different communication requirements need to be supported. The simplest solution to support this flexibility would be a fully interconnected network as designed in [8], which however would occupy significant area. For our kernel design interconnection between 16 outputs of kernel which are fed back to the 32 inputs based on trellis structure, a fully interconnected network will require (32x15=480) 2-1 multiplexers. For a more judicious use of hardware resource we propose to map different possible communication scenarios resulting from the need to support different standards; such scenarios are classified in Table III. As the next step we

TABLE III
ACS INTERCONNECTION SCENARIOS

No.	Interconnection Scenario
1	16 and higher State CC
2	8 state Duo-Binary turbo forward trellis
3	8 state Duo-Binary turbo backward trellis
4	16 state Binary turbo forward trellis
5	16 state Binary turbo Backward trellis
6	8 state Binary turbo forward/backward trellis
7	4 state Binary turbo forward/backward trellis
8	16 state Binary turbo LLR computation
9	8 state Duo-Binary turbo LLR computation
10	16 state Binary turbo LLR computation
11	4 state Binary turbo LLR computation

map these 11 scenarios over a single ACS network acting as a common interconnection across all supported codes: this approach provides a significant hardware reduction as summarized in Table IV, where complexities are expressed in terms of 2:1 multiplexers. Taking into account forward and backward trellis computations for all the codes supported in standards under consideration and possibility of reuse of kernel for partial LLR computation, the designed network still occupies 70 % less area in terms of multiplexers used as compared to fully interconnected network as designed in [8].

TABLE IV
COMPLEXITY OF ACS NETWORK USING CLASSIFICATION OF TABLE III

Unit	mode 1	mode 2	State of Art ([10])
MUX (2-1)	84	133	480

IV. TURBO FEC KERNEL REUSE FOR LDPC DECODING

A LDPC decoder is defined by its parity check matrix H of M rows by N columns. Each column in H is associated with one bit of the codeword or Variable Node (VN), and each row corresponds to a parity check equation or Check Node

(CN). LDPC codes are decoded in an iterative way by using the sum-product algorithm or belief propagation algorithm involving CN and VN updates [12]. However this two phase decoding has recently given way to the so called layered or shuffled decoding [13], [14] which results in approximately two times faster convergence of the algorithm. For highly efficient decoder implementations it is furthermore necessary to use suboptimal check node approximations of low complexity.

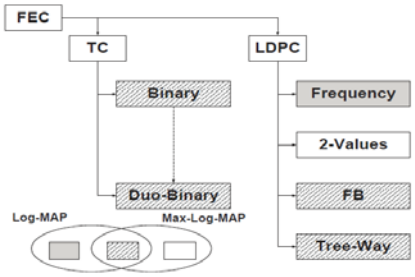

Fig. 3. State Metric (turbo) and Check Node (LDPC) processing implementations

In Fig. 3 the right half shows different ways in which the core check node computations can be performed. Frequency domain computation kernel [14] involves addition operation and look up table (LUT) and is far away from turbo kernel's ACS operations, thus it will not be considered in this work. The forward backward (FB) way is similar to turbo processing on a 2 state trellis and could be performed either using min-sum (which is similar as Max-Log-Map from implementation perspective) or the Log-Map [14] algorithm. Method known as 2-values calculation (or 2MIN) exploits the fact that in min-sum decoding, out of all CN LLRs of a CN only two magnitudes are of interest, since only the minimum (MIN1) and the second minimum magnitude (MIN2) are used to produce LLRs for connected VNs. It is the natural way to compress data in memory and normally results in significant memory saving in CN kernel [15]. There are other similar approaches in this class, like λ-min [16] or average min-sum [17]. The proposed new category of implementation named "Tree-Way" approach will be detailed in the next section.

While reusing the turbo kernel for LDPC decoding utilizing either FB way or 2 value computation approach, 8 serial check node computation can be processed in parallel, each of them mapped onto separate D-ACS unit. Fig. 4.(a) shows the D-ACS unit with only turbo functionality (basic blocks in Fig. 2), in which Level 2 CS is active only when duo-binary or "trellis compacted" turbo-code is used, otherwise there are two separate datapaths (shown by the bold lines), handling of these configurations results in the use of LEVEL 1 MUX. Inputs NETOUT correspond to updated state metric values routed through ACS network. Incorporating LDPC functionality in the turbo kernel results in increased complexity of the basic D-ACS unit. As an example, Fig. 4.(b) shows a D-ACS unit with both turbo and LDPC (2-value) functionalities, for LDPC decoding it is used only for MIN1 and MIN2 calculation part of the check node processing. Complete details of the implementation aspects of the 2 min check node architecture is beyond the scope of this work, interested readers are referred to [15].

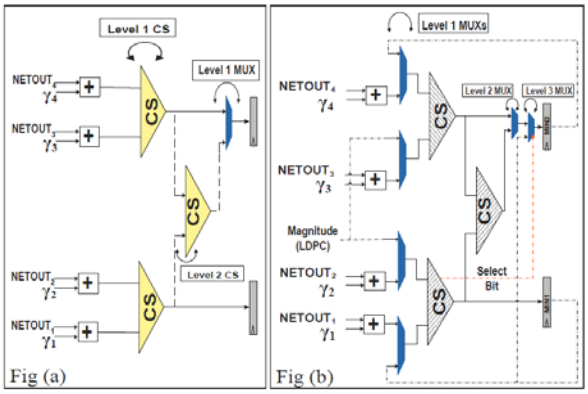

Fig. 4. (a) D-ACS unit with Turbo decoding functionality. (b) D-ACS unit with Turbo/LDPC (2 MIN) functionalities.

The dataflow for implementing the min finder algorithm is highlighted by the dotted lines in Fig. 4.(b). Three CS units are completely reused for min finder function. It can be seen that extra Level 1 and 3 multiplexer units are introduced in turbo D-ACS while supporting the LDPC decoding functionality.

V. Tree-Way Implementation Scheme

In this section we present a parallel implementation scheme of check node updates, developed keeping in mind the trellis computation kernel of turbo decoding. Fig 5 shows a simple graphical representation of the approach. For check node

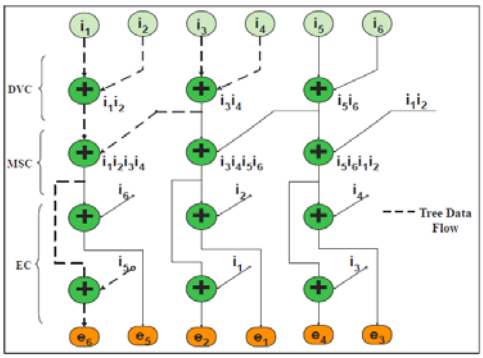

Fig. 5. Parallel Tree-Way Calculation of VN extrinsics

degree d_c = 6, each VN (i_1,i_2, ...i_6) is represented as a leaf node and tree is traversed performing min calculation at branch nodes until VN extrinsics are derived at the root nodes (e_1,e_2, ...e_6). One of the key contributions of this paper is that for a parallel check node architecture we generalize the tree network connectivity and the data flow for any value of d_c and present a fairly simple control mechanism for it. For the sake of architecture uniformity odd d_c values are considered as their even counterpart with extra VN intrinsic value initialized at $+\infty$ (i.e. d_c=d_c if d_c is even; else d_c=d_c+*1*).

Fig 6 shows the architectural mapping of different stages of VN extrinsic calculation for proposed "Tree-Way" scheme, other than sign accumulation which is performed by separate *XOR* tree (not shown in figure), VN extrinsic calculation consists of following stages:

A. Direct VN Comparison (DVC) stage

As seen in Fig 6 for d'_c=8, the intrinsic values (i_1,i_2, ...i_8) are fed parallely to two D-ACS units that are configured as 4 CS units. For all values of d_c there is only one direct comparison stage. The output of DVC and each subsequent stage is passed on to next stage through ACS network (NETOUT) as well as are stored in SM memory (see Fig 2) for use in later stages.

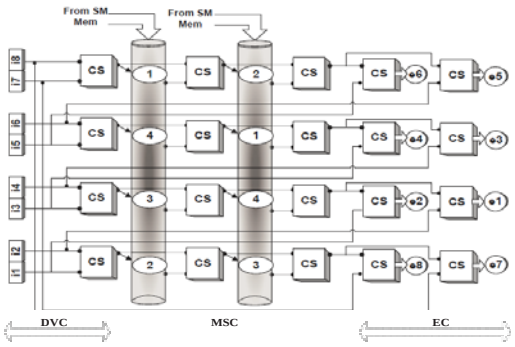

Fig. 6. Generalization of Tree-Way Scheme

B. Multiple Shuffled Comparison (MSC) stage

Shuffled comparison stage could be compared to the trellis computation stages in BCJR algorithm for turbo decoding. The shuffle network implements a circular shifting permutation, which can be easily mapped on to acs network without significant hardware cost. The rotational shift depends on d_c and the substage of the shuffled comparison stage. There are multiple shuffled stages depending on d_c and equal to $N_{cc} - 3$ where N_{cc} is the required number of clock cycles for check node update. For different values of d_c Table V 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. It should be noted that, FB and 2MIN approaches being inherently serial, take longer time for check node updates, while "Tree-Way" approach parallizes the check node computation and results in lesser clock cycles. For example, if d_c=21, 7 cycles are required in proposed method instead of 21 in a serial approach, which means 67% of saved cycles.

As can be seen from Fig 6 the input to the shuffle network is either the output from the immediate previous stage or output of a much earlier stage stored in the state metric memories. Irrespective of the input source of the shuffle network, the shift associated with a stage is fixed. For retrieval of outputs stored in SM memory from previous stages, a simple mechanism is formulated. Output values at each stage are assigned a binary label e.g. '10' represent DVC stage output, while '100' and '1000' represent first two MSC stage outputs and so on. For a given stage, the address to be accessed in SM memory depends on d_c value and can be derived using a relatively simple control based on binary representation of the value d_c-2. 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 corresponds to values at labels of '100' and '10'.

TABLE V
CLOCK CYCLE REQUIREMENT AND SHIFT PERMUTATION.

d_c	N_{CC}	Permutation	d_c	N_{CC}	Permutation
5,6	4	1	19, 20	7	1, 2, 4, 8
7,8	5	1, 2	21, 22	7	1, 2, 4, 8
9,10	5	1, 2	23, 24	8	1, 2, 4, 8, 10
11,12	6	1, 2, 4	25, 26	7	1, 2, 4, 8
13,14	6	1, 2, 4	27, 28	8	1, 2, 4, 8, 12
15,16	7	1, 2, 4, 6	29, 30	8	1, 2, 4, 8, 12
17,18	6	1, 2, 4	31, 32	9	1, 2, 4, 8, 12, 14

At this point it is interesting to evaluate the hardware complexity of the MSC shuffled network. From the perspective of reusing the ACS-Network of Turbo/Viterbi kernel for LDPC decoding the permutations in Table V are mapped over the interconnection matrix for Turbo/Viterbi which is based on classification of Table III. The resultant interconnection network still occupies 50 % less area in terms of network as designed in [8] and supports turbo, Viterbi and LDPC decoding functionalities. Although there is some additional complexity due to control logic compare to the optimized shuffle network without "Tree-way" approach, the hardware reuse is nevertheless significant.

C. Extrinsic Calculation (EC) stage

The last two stages for any value of d_c are extrinsic calculation stage. As seen in Fig 6 input to these stages is the output from the last MSC stage and the shifted *VN* intrinsic values. This shift is circular over d'_c and equal to 1 and 2 for EC stage 1 and 2 respectively.

D. D-ACS unit implementation and ASIC Synthesis Results

Fig. 7 shows both turbo and LDPC (Tree-way) functionalities incorporated in D-ACS unit. It can be seen that, this flexibility comes at the cost of introduction of multiplexers and logic gate layers in the architecture. Level 1 AND gates and Level 3 MUX are driven by the choice of FEC decoding algorithms (i.e. Turbo-binary/Turbo-duo binary/ LDPC), while level 2 MUXs are supporting inputs from different stages of "Tree-Way" implementation.

First half of the Table VI shows the comparison of ASIC synthesis results for D-ACS unit when different check node architectures are mapped onto turbo decoding infrastructure. At the D-ACS unit level FB approach is the most promising as it shows a datapath area saving of 17.2 % compared to sum of two dedicated architectures for turbo and LDPC decoding. On the other hand lower half of the table shows the comparison of synthesis results at the higher level of hierarchy of FEC kernel. It can be seen that though LDPC kernel with "Tree-way" approach occupies larger area than its FB and 2MIN

TABLE VI
SYNTHESIS COMPARISON RESULTS. (LOGIC GATES IN 0.13μm TECHNOLOGY AT 300MHZ)

Cost of FEC D-ACS Unit H/W									
Independent Architectures				Combined Architectures					
TC	LDPC			Sum of Two Architectures			Shared Architecture		
	FB	2MIN	Tree	FB	2MIN	Tree	FB	2MIN	Tree
405	309	396	342	714	801	747	591 (-17.2%)	687 (-14.2%)	625 (-16.0%)
Cost of FEC Kernel H/W									
TC	LDPC			Sum of Two Architectures			Shared Architecture		
	FB	2MIN	Tree	FB	2MIN	Tree	FB	2MIN	Tree
7691	2472	3168	4803	10163	10859	12494	8628 (-15.1%)	9396 (-13.4%)	8901 (-28.7%)

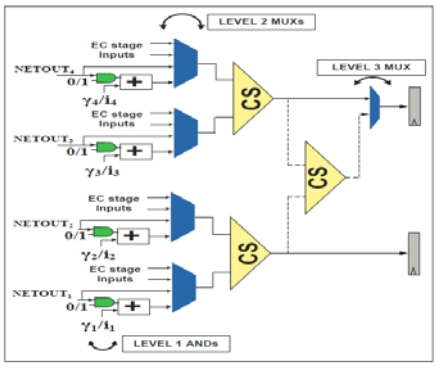

Fig. 7. D-ACS unit with Turbo/LDPC (Tree-way) functionalities.

counterparts, the complexity of its shared kernel architecture remains almost the same as that of FB and 2MIN, thanks to the reuse of turbo ACS network.

VI. CONCLUSION

In this paper we explored the design space of flexible multi-standard FEC decoder platform. We presented a VLSI complexity analysis of datapath sharing across FEC code families viz. convolutional, turbo and LDPC for such a hardware platform. A reduced complexity network was designed to realize trellis structures for different turbo and convolutional codes used in various wireless communication systems. Furthermore, implementation results of various possibilities of check node architectures in LDPC decoding reusing the Turbo Max-Log-MAP core were presented. The results are useful for designers to make early architectural choices while designing a multi-standard FEC kernel. In addition to this, to best of our knowledge, the paper presented the first parallel implementation of check node computations using Min-sum algorithm for LDPC decoding, which is optimized for maximun reuse of turbo decoding kernel (-28.7 % less area compared to two independent turbo and LDPC kernel) and is efficient in terms of clock cycles required for check node computations.

REFERENCES

[1] M. A. Bickerstaff, D. Garrett, T. Prokop, C. Thomas, B. Widdup, G. Zhou, and L. M. Davis, "A Unified Turbo/Viterbi Channel Decoder for 3GPP Mobile Wireless in 0.18-mm CMOS", IEEE Journal of Solid-State Circuits, pp. 1555-1564, Nov. 2002.

[2] G. Kreiselmaier, T. Vogt, and N. Wehn, "Combined Turbo and Convolutional Decoder Architecture for UMTS Wireless Applications", Proceedings of DATE, pp. 192-197, Feb. 2004.

[3] Y. Sun and J. R. Cavallaro, "Unified decoder architecture for LDPC/turbo codes", IEEE Workshop on Signal Processing Systems, pp. 13-18, Oct. 2008.

[4] T. Vogt and N. Wehn, "A Reconfigurable Application Specific Instruction Set Processor for Convolutional and Turbo Decoding in a SDR Environment," in Proc. of DATE'08, Germany, pp. 38-43, Mar. 2008..

[5] M. Alles, T. Vogt and N. Wehn, "FlexiChaP: A reconfigurable ASIP for convolutional, turbo, and LDPC code decoding," Turbo Codes and Related Topics, 2008 5th International Symposium on , pp.84-89, 1-5 Sept. 2008.

[6] F. Naessens, B. Bougard, S. Bressinck, L. Hollevoet, P. Raghavan, L. Van der Perre, F. Catthoor, "A unified instruction set programmable architecture for multi-standard advanced forward error correction," IEEE Workshop on Signal Processing Systems, pp.31-36, 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. 284287, Mar. 1974.

[8] O. Muller, A. Baghdadi, M. Jeźeq́uel. "ASIP-Based Multiprocessor SoC Design for Simple and Double Binary Turbo Decoding", in Proc. of DATE'06), Germany, pp. 1330-1335, Mar. 2006.

[9] P. Black and T. Meng, "A 140-mb/s, 32-state, radix-4 Viterbi decoder", IEEE Commun. Lett., vol. 27, no. 12, pp. 1877-1885, Dec. 1992.

[10] K. Huang, F. Li, P. Shen, and A. Wu, "VLSI design of dual-mode Viterbi/turbo decoder for 3GPP", Proceedings of International Symposium on Circuits and Systems,pp. 773-776, May 2004.

[11] B. Min, "Architecture VLSI pour le decodeur de Viterbi", Thesis: Telecom Paris , June 1991.

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

[13] D. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," IEEE Workshop on Signal Processing Systems, USA, pp. 107-112, Oct. 2004.

[14] M. Mansour and N. Shanbhag, "Low-power VLSI decoder architectures for LDPC codes," in Proc. of the 2002 International Symposium on Low Power Electronics and Design, USA, pp. 284-289, Aug. 2002.

[15] 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.

[16] T. Bhatt, V. Sundaramurthy, V. Stolpman, and D. McCain, "Pipelined block-serial decoder architecture for structured LDPC codes," in Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing , vol. 4, France, p. IV, May 2006.

[17] N. Axvig, D. Dreher, K. Morrison, E. Psota, L.C. Peŕez and J.L. Walker "Average Min-Sum Decoding of LDPC Codes" in Proc. of International Symposium on Turbo Codes and Related Topics, Switzerland, pp. 356-361, Sept. 2008.

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 Universite 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.