VLSI DESIGN OF MULTI STANDARD TURBO DECODER FOR 3G AND BEYOND

Imran Ahmed, Tughrul Arslan

School of Electronics and Engineering. University of Edinburgh, King's Buildings Mayfield Rd, Edinburgh, EH9 3JL, UK
Institute for System Level Integration, The Alba Campus, The Alba Centre, Livingston, Scotland, EH54 7EG, UK

ABSTRACT

Turbo decoding architectures have greater error correcting capability than any other known code. Due to their excellent performance turbo codes have been employed in several transmission systems such as CDMA2000, WCDMA (UMTS), ADSL, IEEE 802.16 metropolitan networks etc. The computation kernel of the algorithm is very similar and we have exploited this commonality for a turbo decoder VLSI design suitable for deployment using platform based system on chip methodologies. Turbo and viterbi components of the unified array are also individually reconfigurable for different standards. This supports the 4G concept that user can be simultaneously connected to several access technologies (for example Wi-Fi, 3G, GSM etc) and can seamlessly move between them. A new normalization scheme for turbo decoding is presented to suit reconfigurable mappings. We have also shown dynamic reconfiguration methodology for a context switch between Turbo and Viterbi decoders which does not waste any clock cycles. The reconfigurable Turbo decoder fabric is implemented reusing components of Viterbi decoder on a 180 nm UMC process technology.

1. INTRODUCTION

In 1993, a parallel concatenated convolution code (PCCC) decoding scheme was proposed by Berrou et al., which consists of two SISO (soft input soft output) decoders concatenated through an interleaver - deinterleaver structure [1]. These component decoders are individually matched to corresponding encoders as shown in figure 1. The interleaver allows the low-weight code words produced by a single encoder to be transformed into high-weight code words for the overall encoder. This iterative decoding achieves transmission performance of a few tenths of a dB from Shannon limit when applied to BPSK transmission over channel with memory less noise.

The conventional VLSI implementation of a Map decoder (operating in Log Domain) involves complex multiplication, exponentials and logarithm computations. Suboptimal varieties of Map, Max-Log-Map, Linear Log Map, log Map [2-3] are usually used for VLSI implementations.

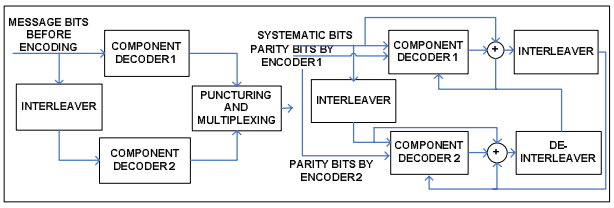

Figure 1. PCCC (Parallel Concatenated Convolution) Encoder and Decoder.

The aim of the paper is not to rigorously derive these algorithms but to identify critical issues related to a reconfigurable turbo decoder array with the aim to facilitate various viterbi decoding mappings. Our previous work [4] showed viterbi component details for the platform. This paper extends these concepts to reconfigurable turbo decoder domain. The block diagram of the communication platform is shown below

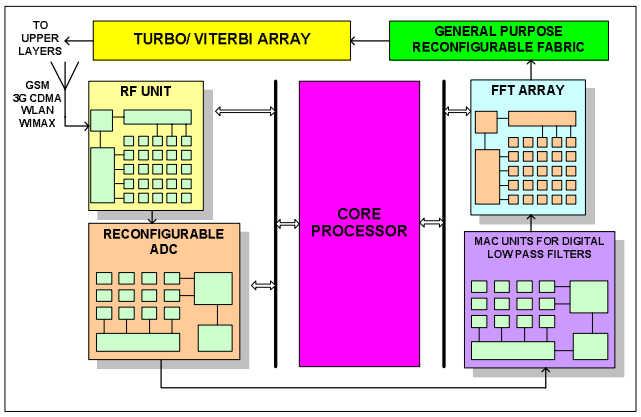

Figure 2. Platform showing Turbo/Viterbi Array.

2. OVERVIEW OF THE ALGORITHM

For coherence of representation and to show essential components of turbo decoders the Map algorithm is briefly described.

The MAP algorithm gives, for each decoded bit u_K in step k, the probability that the bit was +1 or -1, Let $y_0^N = (y_0, y_1, y_2, ..., y_N)$ is the received distorted symbol sequence. Let S_K denote the state of the encoder at the time instant k. Assuming events after time index k are not influenced by observation y_o and bit u_k, if S_k is known, the log likelihood ratio value is calculated as:

1-4244-0630-7/07/$20.00 ©2007 IEEE.

$$L(u_k \mid y_o) = \log \frac{\Pr(u_k = +1 \mid y_o)}{\Pr(u_k = -1 \mid y_o)} = \log \frac{\sum_{(s`,s)=>u_k=+1} \alpha_{k-1}(s`).\gamma_k(s',s).\beta_k(s)}{\sum_{(s`,s)=>u_k=-1} \alpha_{k-1}(s`).\gamma_k(s',s).\beta_k(s)} \quad (1)$$

where α is the forward state metrics, β is the reverse state metrics and γ is the branch metrics.
The conventional VLSI implementation of a Map decoder (operating in Log Domain) involves complex multiplication, exponentials and logarithm computations. Suboptimal varieties of Map, Max-Log-Map, Linear Log Map, log Map [10,11] are usually used for VLSI implementations.

3. VLSI DESIGN

The block diagram of unified multi standard Turbo-Viterbi array is shown below and essential components of the array are explained in subsequent sections.

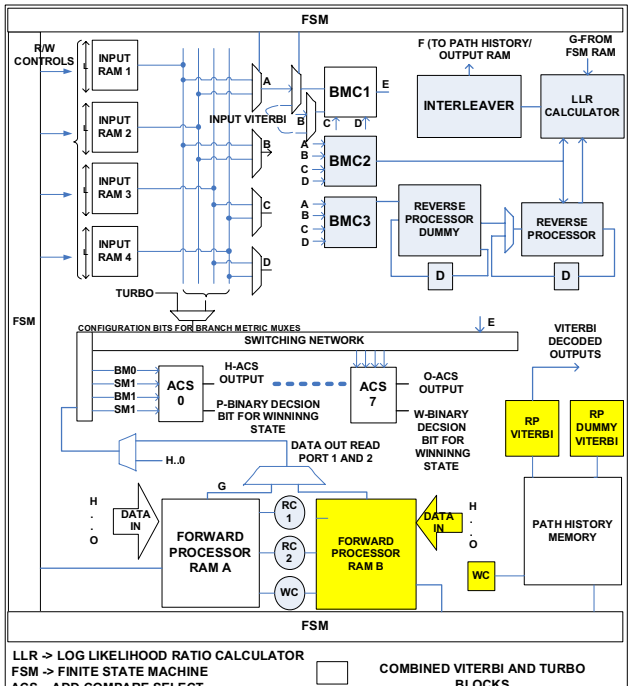

Figure 3. Block diagram

3.1 INPUT RAMS:

Input RAMs store input metrics for two window lengths (WLs). In viterbi mode the same input RAMs store the Branch Metrics configuration bits [4]. We have used Write-After-Read (WAR) RAMs to implement two memory architecture compared with three memory architecture proposed in [5]. We had shown in our previous work [4] that in viterbi mode the write and read operation on these RAMs is done without wasting any clock cycles resulting in dynamic context switch for multi standard viterbi mappings and continuous decoding operation for turbo mode.
Both viterbi and turbo decoders use forward and reverse state metrics processing. To improve the latency typically windowed versions of the algorithm are employed for VLSI implementations, largely known as sliding window BCJR algorithm [6]. The basic effect is that the equations will be applied separately to portions (window lengths- WLs) of the global block of data. In its simplistic form the algorithm uses two reverse processors Reverse Processor Dummy B2 and Reverse Processor B1 in parallel with on forward processor (shown by ACS0-ACS7 in figure 3). B2 can start cold in any state (initializing each state as equi – probable) but after a few iterations (equal to WL) the state metrics are as reliable as if the process had been started at the final known correct node of trellis. B2 initializes the start state of B1. The state machine controls the writing of input metrics and for Turbo mode it is shown in figure 4 for (WL =32).
The RAMs can be read and written in either forward or reverse direction by state machine using the 5 bit forward and reverse counters. This is shown by the scheduling diagram below

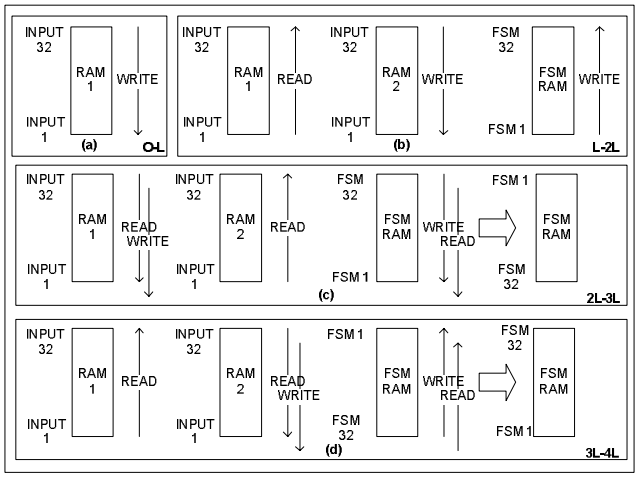

Figure 4. Read/Write FSM Control for RAMs

In Viterbi mode the write and read controls by FSM are much simpler and explained in [4]. For Turbo mode these are explained with the help of figure 4 and figure 5 below:

3.1.1 WINDOW LENGTH 0-L (FIGURE 4A):

Input metrics corresponding to first window length 0-L are written in RAM1. The last metric is saved in first memory

location and first metric in last memory location as shown in figure 4a.

3.1.2 TIME SLOT L-2L (FIGURE 4B):

Input metrics corresponding to second WL (L-2L) are written in RAM2. Reverse Processor Beta (RP2) uses these input values to calculate reverse state metrics (RSMs) Forward processor (FP) calculates forward state metrics (FSMs) by reading the RAM1 in reverse direction as shown in figure4b. Calculated FSMs are saved in FSM RAM in the reverse direction i.e., last state metric in first memory location and the first metric in last memory location.

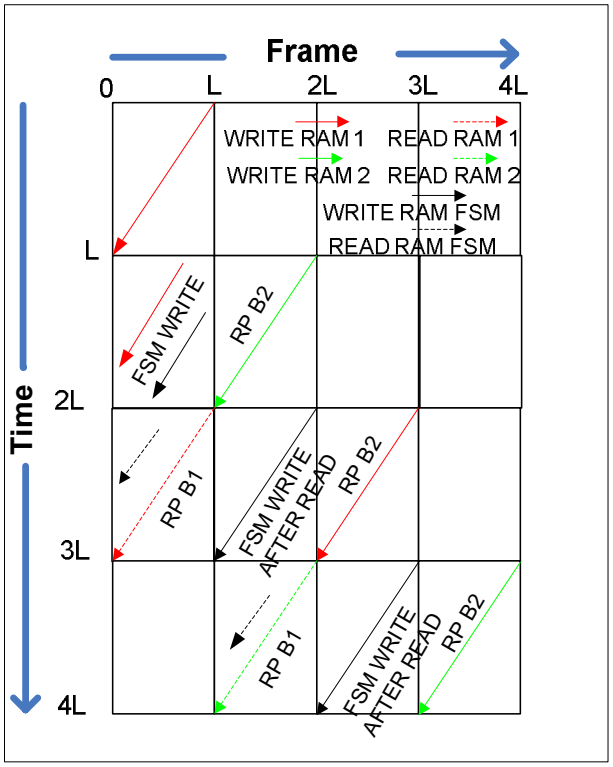

Figure 5.Scheduling diagram.

3.1.3 TIME SLOT 2L-3L (FIGURE 4C):

After the latency of the two WLs, the LLR values corresponding to WL '0-L' are calculated.
LLR calculates the decoded bits by reading FSM RAMs in forward direction as shown in figure 4c. Reverse Processor (RP1) is initialized by RP2. RAM1 is read in forward direction to provide input metrics (corresponding to WL 0-L) for RP1 calculations. FP calculation is now performed on WL L-2L, which is done by reading the RAM2 in reverse direction as shown in fig 4c. Calculated FSM values are saved in FSM RAM (Write-After-Read). Since FSM RAM was read in forward direction the write will also be performed in forward direction and first FSM value is saved in first memory location and last FSM value saved in last memory location.
RAM1 is read for RP2 calculations (corresponding to frame 2L-3L). The input metrics (for frame 2L-3L) are written on RAM1 after the old values are read by RP1. This is shown by solid red arrow in figure5.

3.1.4 TIME SLOT 3L-4L (FIGURE 4D):

This slot provides the LLR decoded outputs for second WL L-2L. LLR calculator calculates the decoded bits by reading FSM RAMs in reverse direction as shown in figure 4d. RAM2 is read in forward direction for RP1 calculation (for window length L-2L). RAM1 is read in reverse direction for FP calculation (for WL, 2L-3L). This is shown in fig 4d. Calculated FSM values are saved in FSM RAM after read operation. Since FSM RAM was read in reverse direction the write will also be performed in reverse direction and last FSM is saved in first memory location and first FSM saved in last.
RAM2 is read to calculate RP2 values (corresponding to frame 3L-4L).
The cycle repeats after this where time slot 4L-5L is similar to time slot 2L-3L and time slot 5L-6L is similar to time slot 3L-4L.

4. BRANCH METRICS CALCULATOR (BMC)

Branch metrics are computed by computing the Euclidian distance of the soft input metrics. One of the key points in the implementation of channel decoders is the fixed point representation for all quantities involved in the decoding algorithm. The finite precision of input metrics directly affects the capacity of input buffers. The input metrics are represented in 4Q2 signed two's complement format which provides an acceptable trade-off between error correcting performance and area. In Viterbi mode only BMC1 is used. In this mode the input metrics to BMC1 are provided directly bypassing the input memory arrangement (used in Turbo mode) by using multiplexer as shown in figure 3. In the turbo mode BMC1 calculates Branch metrics for FP, BMC2 for RP2 and BMC3 for RP1.
BMC1 and BMC2 are connected to input RAMS 1 and 2 by using multiplexers which are controlled by finite state machine (FSM) as per the scheduling algorithm explained in section 3 and table 1 below:
BMC2 which provides branch metrics for RP2 is directly connected to input metrics.

Time Slot	BMC1(FP)	BMC3(RP1)
0-L	*RAM2*	*RAM1*
L-2L	RAM1	*RAM2*
2L-3L	RAM2	RAM1
3L-4L	RAM1	RAM2

4L-5L	RAM2	RAM1

Table 1. Input RAMs connections to BMC blocks.

5. FORWARD AND REVERSE PROCESSOR CALCULATION:

The main kernel of the Turbo-Viterbi algorithm is ADD-COMPARE-SELECT (ACS) operation which is preformed by FP, RP1 and RP2 blocks. These blocks have similar designs and are shown in figure 6. There are 8 parallel ACS blocks and hence 8 states can be processed in parallel. Therefore for ADSL (generator matix [1,17_{octal}/15_{octal}]), Metropolitan Area Network IEEE 802.16 (Tail biting Circular Convolution codes), DVB-RCS (similar to 802.16-Duo Binary), and 3GPP turbo mappings on the array will work in fully parallel schemes. Fully parallel architectures assign one ACS for each state to meet the performance constraints on speed and latency. In Viterbi mode however since number of states (N) are higher (256 states for 3GPP) therefore P(P=8) ACS are used to process N (up to 256) states[4]. Similarly CCSDS (Consultative Committee for Space Data Systems) turbo decoder family has 16 states which will be decoded 8 states at a time is a similar fashion as GSM Viterbi mappings on Viterbi array as was explained in our previous work in [4].

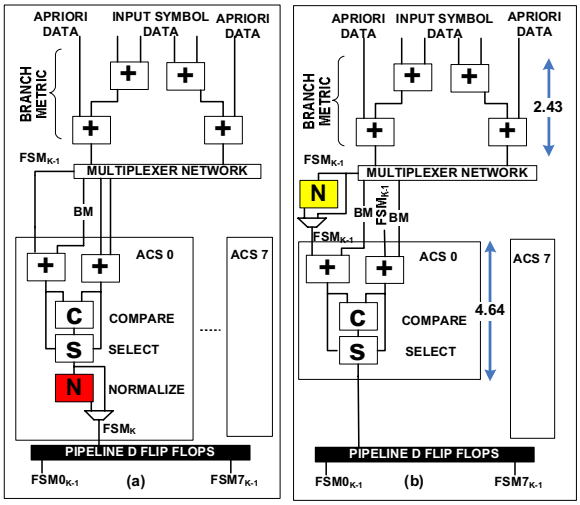

Figure 6. Normalization scheme and BM, FSM units

5.1 NORMALIZATION / SATURATION:

We have adopted a new normalization scheme to support the mappings that do not use state parallel architectures. These include all viterbi mappings and for all turbo mappings on the array that has greater than 8 states (for example decoders for CCSDS telemetry operations).

State metrics (FSMs and RSMs) are accumulated within a block as they are recursively computed for sliding window ACS computations. To avoid overflow metrics normalization is usually employed as shown in figure 6a. We have adopted a very efficient normalization scheme where at each time instant we check if any of the state metrics is greater than 2^{q-2}, then a fixed value 2^{q-2} is subtracted from all state metrics. This is shown by normalization (N) block shown in figure 6. The block comprises of a subtractor that subtracts a fixed value (2^{q-2}) from state metrics and a multiplexer that selects the subtracted value if the normalization has to be employed. The multiplexer select signal is provided by each ACS block and in case of state serial architecture mappings (states >8) the select signal is provided after all the states are processed. In figure 6a the normalized FSMs were saved in the FSM RAM, this new scheme the normalization is applied after reading the state metrics from FSM RAM. The critical path delay of Branch Metric and State Metrics component is shown in figure 6 with blue arrows. Note that this adjustment keeps the critical path still exactly the same, however now the same Processor blocks can be used for decoders with states greater than 8.

6. LLR CALCULATION:

As shown in figure above LLR block require the values of forward, backward state metrics and branch metrics. It consists of two identical blocks (block A) calculating the LLR of bit 0 and bit1 respectively. The maximum calculated value of LLR1 and LLR0 is subtracted to find the final LLR output value. The sign of a posteriori value gives the value of decoded bit 1 or 0. LLR block is used in turbo mode only and is disabled in viterbi mode. The LLR block is pipelined to reduce the critical path delay. The position of pipeline registers is shown by dotted line. Insertion of this pipeline reduces LLR components bottleneck on critical path (delay in ns shown in blue arrows). However ACS still remains in critical path and cannot be further pipelined due to recursive nature of mapped algorithms.

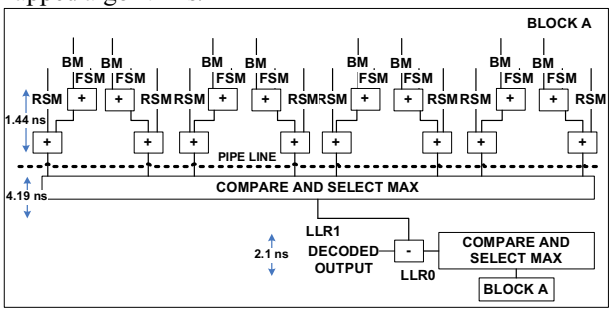

Figure 7 LLR Computation Unit.

7. RECONFIGURABLE INTERCONNECT

The reconfiguration topology for viterbi mappings were explained in our previous work [4]. We have used a fully flexible trellis processing for Turbo decoding as well. This

allows mappings of decoder with any generator polynomials. Each branch metrics and FSM connection to ACS block is done through a multiplexer. For example for Rate ½, there are four possible branch metrics that can be connected to each BM branch of ACS block. Similarly for 8 states, there will be 8 possible ACS values that can be fed back to each FSM branch of ACS (refer figure 6 for these connections). These flexible connections are provided through multiplexer network as shown in figure 1. The multiplexer network is therefore a multiplexer bank providing 4x1 and 8x1 multiplexer connections for each BM and FSM branch of ACS operation of Forward and Reverse processors. Viterbi blocks in the array are shown in white in figure 1 and these are clocked down by using an active clocking gating strategy throughout the chip.

8. INTERLEAVER:

One challenge in the design of turbo decoders is the length of the interleaver. The near Shannon performance of turbo codes is directly linked with the length of the interleaver. 3GPP defines an interleaver of the order greater than 5 thousand bits. Interleavers are usually implemented storing the interleaved address patterns in LUTs or ROMs. This storage will amount to interleaver memories equivalent to frame length (for example 5114x6 bits for 3GPP). This is a major overhead on area and power and we have addressed this in our previous work [7]. We have shown performance improvements by an alternative memory less implementation of 3GPP S-Random Interleaver.

9. RESULTS

The design is synthesized using Synopsys Design Compiler for 0.18 microns CMOS UMC cell library and the chip layout is done on Silicon ensemble. Post layout power figures are taken from Synopsys Design Power by capturing the toggle activity of each node and then back annotating this in the circuit. Synopsys designware SRAMs were used for Forward Processor RAMs. Virtual Silicon 2K x 8 synchronous (separate read and write port) macro RAMs were used for Output/Path history memory consuming 110 uW/MHz/Port.

The overall results are summarized in table 2 below.

Technology	UMC 0.18 microns standard cell CMOS
Supported code rates	½,1/3 Turbo, ½,1/3,1/4,1/5 Viterbi
Representation	Signed fixed point
Constraint length	Max 4 in parallel mode and max 9 (256 states) for state sequential (8 states at a time)
Generator Polynomial	Flexible for both turbo and viterbi
Survivor (Trace back length)	Up to 6 times the constraint length
Decision level	4 bit soft decision
A posteriori estimation	6 bit soft decision.
Supply Voltage	1.8V
Interleaver	Memory less-supports frames upto 5114 (3GPP)
Max Operating Frequency	84 MHz
Max Throughput Turbo @ 6 iterations	14 Mbits / sec (overall for 6 iterations)
Latency	2 window lengths
Total area	1.67mm^2 (without output RAMs) 2.88mm^2 (with output RAMs)
Power @ 20 MHz(Turbo mode)	78.54 mW

Table 2. Results

The components of turbo decoder array contributing in the critical path delay (without LLR pipeline) are : Input Rams, BMC and LLR. The total delay for the critical path will be 15.01 ns. The insertion of LLR pipeline changes the critical path and the components in the path are: Input Rams, BMC and FP. The overall path delay will be 11.92 ns. This can be further improved if CLA adders are used instead of Full Adders in BMC and FP blocks. The critical path and the delay of the individual components is shown in figure 8 below:

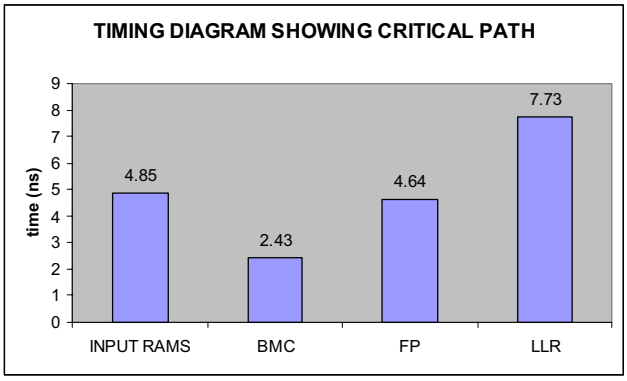

Figure 8. Timing diagram

The overall area and power results of individual components can be compared in figure 9.

The turbo decoder array is compared with the following reconfigurable categories: ASIP (application specific instruction set processor), implementations on general purpose processors, implementations on FPGAs and ASICs. In ASIP flexibility is provided by the use of embedded processors specifically targeted to the decoding application. ASIP being software controlled is broader in domains of reconfigurability and hence more flexible than our design. However power, area and speed figures are much lower. For example in [8] there would be 8 XiRisc processors needed in order to achieve a through put of

2Mbps. They are also required to run in parallel on successive blocks of data.

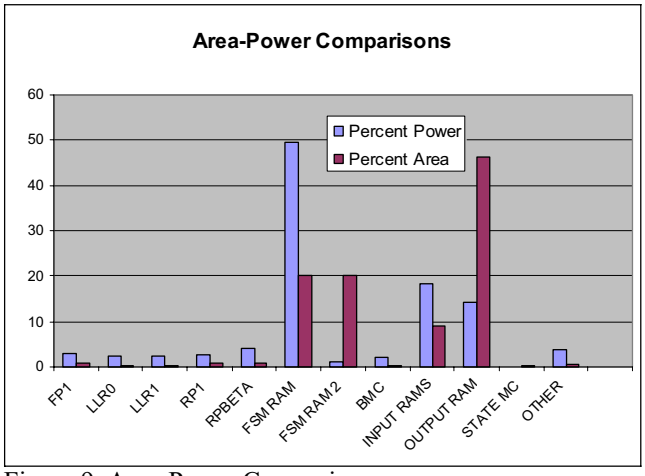

Figure 9. Area-Power Comparisons.

Table3 below lists some Turbo decoder implementations on general purpose processors and also lists the maximum through put possible. Results are worse than ASIP however flexibility will be higher.

Processor	Clock Speed	Throughput possible	Ref
Motorola 56603 DSP	Not quoted	48.6 kbps/iteration	[9]
ST120	200MHz	540 kbps/iteration	[9]
Intel Pentium III	933 MHz	262 kbps/iteration	[10]
DSP SP-5 SIMD	Not quoted	227kbps/iteration	[11]

Table 3. Throughput quoted for general processors.

Dedicated implementation on general reconfigurable logic for example FPGAs can achieve higher throughput however consume higher power than ASIC implementation.. For example, in [12] implementation on Xilinx Vitex XCV300E (almost 50% resource utilization) consumes 695mW(25 Mhz) for 1Mbps.

A more exact ASIC comparison of our IP can be made with the work in [14], where we have achieved similar area and power figures, however the reported unified array targets 3GPP standard only. Our design is more flexible as it can target multiple standards both on viterbi and turbo mode.

10. CONCLUSION

We have presented a unified FEC IP solution that can be used either independently or as a Co – Processor for increased flexibility. Decoder consumes 78.5 mW occupying 2.824 mm^2. We have shown the benefits of domain specific reconfigurable platform in terms of area, power and speed as compared to more general purpose Processor or FPGA based solutions.

11. REFERENCES

[1] C.Berrou, et al. “Near Shannon limit error-correcting coding and decoding: Turbo-Codes,” In Proc. ICC `93, Pages 1064-1070, Geneva, Switzerland, May 1993.

[2] P. Robertson, E. Villerbrun and P. Hoeher, “ A comparison of optimal and sub-optimal MAP decoding algorithm operating in the log domain,” in porc. ICC’95, pp. 1009-1013

[3] Shahram Talakoub et al., “ A Linear Log-MAP Algorithm for Turbo Decoding and Turbo Equalization”, *IEEE Conference WiMob’2005*, vol. 1, pp. 182–186, Aug. 2005.

[4] I Ahmed, T Arslan, “ A Reconfigurable Viterbi decoder for a communication platform”. IEEE FPL

[5] Masera et al, “VLSI architectures for turbo codes”; IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Sept. 1999 Page(s):369 - 379

[6] S. Benedetto et al, “Soft-output decoding algorithms for continuous decoding of parallel concatenated convolutional codes”, Proceedings of ICC’96, Dallas, Texas, June 1996.

[7] I Ahmed, T Arslan,“ A low energy VLSI design of Random Block Interleaver for 3GPP Turbo Decoding”. ISCAAS 2006

[8] A. La Rosa, et al, “ Implementation of a UMTS turbo-decoder on a dynamcially reconfigurable platform”, Design, Automation and Test in Europe, Volume 2, 16-20 Feb. 2004 pp. 1218-1223 Vol 2.

[9] H. Michel, A.Worm et al, “Hardware/Software trade-offs for advanced 3G channel coding”, in Proc. Design, Automation, Test Eur. Conf., Mar 2002, pp. 396-401.

[10] M.C. Valenti and J.Sun,“ The UMTS turbo code and efficient decoder implementation suitable for software-defined radios,” Int. J. Wireless Inform. Netwoks, vol. 8, no. 4, pp. 203-216, 2001.

[11] J. harrison, “Implementation of a 3GPP turbo decoder on a programmable DSP core,” Commun. Design Conf., San Jose, CA, Oct. 2001.

[12] S. Sharm et al, “ A simplified and efficient implementation of FPGA-based turbo decoder” Proceedings of the 2003 IEEE Intl. Conf. on Perf,Computing and Communications”, 9-11 April 2003 pp. 207-213.

[13] Xiao-Jun et al, “Design and implementation of a turbo decoder for 3G W-CDMA systems” Consumer Electronics, IEEE Transaction, Volume 48, Issue: 2, May 2002 pp. 284-291.

[14] Mark A. Bickerstaff et. al., “ A Unified Turbo/Viterbi Channel Decoder for 3GPP mobile wireless in 0.18-um CMOS”, IEEE J. Solid-State Circuits, vol 37, no. 11, pp. 1555-1564, Nov. 2002.