A NOVEL VLSI ARCHITECTURE FOR TURBO DECODERS

Pallavi M.T[1], Mrs. Bhanu Rekha.K[2]
[1]M.Tech, [2]Asst. Professor, ECE Dept, Sir. MVIT, Bangalore, India

ABSTRACT: Before designing any circuits our main aim is to consider the trade-off parameters i, e power, speed and area. Everyday there is an increase in the number of users of mobile wireless services which demands high data transmission rates. The one of the wireless communication standard LTE (Long Term Evolution) currently offering 300Mbps aims to achieve high data rate in terms of Gbps. But the coding technique adopted by LTE is turbo code which involves iterative decoding procedure that results in low decoding throughput. It is necessary to incorporate a decoder which uses minimum number of components and thus uses less area that helps in high coding gain. In this paper an efficient VLSI architecture has been proposed for turbo decoders that uses add-compare-select unit in the architecture.

Keywords: LTE, Turbo decode, Add-compare-select

I. INTRODUCTION

As we have seen there is tremendous increase in the number of wireless communication users there is a need to satisfy the requirements of customers without partiality. It depends on the rate at which data is transmitted and delivered to the user. Hence it is a major challenge among network and broadband companies. Data delivery depends on the rate at which it is decoded. Currently the wireless communication is using Turbo codes as their coding scheme. At the decoder Log-MAP algorithm is used which involves complex computation making VLSI implementation very difficult. In this paper we discuss how this complex computation can be reduced by implementing simple structures and thus reducing area which in turn increases area.

II. TURBO DECODER

Turbo decoder consists of parallel concatenation of two decoders that employ different types of decoder algorithms. One such algorithm is maximum a posteriori (MAP) algorithm. There are different versions of these algorithm. The area and power consumed by the decoder varies based on the algorithm used. The decoder structure is shown figure 1. The output of encoder is fed into communication channel as shown in figure. In the channel the data gets corrupted due to disturbance and hence noise gets added. The noisy data is fed into decoder. The noisy data first enters input buffer from which the same data is distributed among two decoders and the selection of which decoder to operate is in the hands of control unit. The control unit starts the iteration by selecting the decoder 1. At the decoder the LLR value is calculated corresponding to each bit based on the algorithm used. Once the operation completes it notifies the

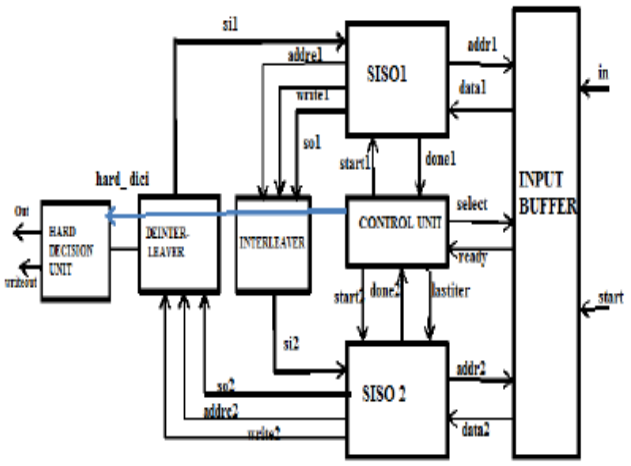

Fig 1: Turbo Decoder

control unit about completion. Then the second decoder is started using data from input buffer and output of first decoder after passing through the interleaver. Similarly the output of second decoder after passing through the deinterleaver and data input from input buffer are the inputs for the decoder 1. During the iterations forward, backward and branch metrics are calculated. The output is taken from the deinterleaver after certain number of iterations. The final output will be a LLR value which will be decoded as 1 if negative and 0 if positive.

III. PROPOSED METHODOLOGY

The modified BCJR algorithm is called MAP algorithm. It has different forms like constant log MAP, max log MAP and Log MAP algorithm. The notations used are u is the input sequence, γ branch metric, β backward state metric, α forward state metric and sk state of encoder at time k. The log likelihood ratio is calculated as follows,

$$\mathrm{LLR}(u_k) = \log\left(\frac{\sum_{u_k=+1}\tilde{\alpha}_{k-1}(s')\tilde{\beta}_k(s)\tilde{\gamma}_k(s',s)}{\sum_{u_k=-1}\tilde{\alpha}_{k-1}(s')\tilde{\beta}_k(s)\tilde{\gamma}_k(s',s)}\right)$$

The metrics used in this algorithm are as follows,

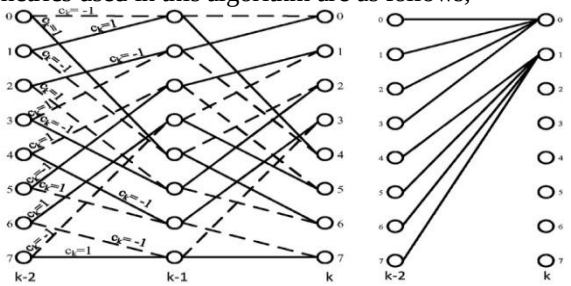

Fig 2: Radix 2 trellis structure and partial radix 4 trellis structure

$$\tilde{\alpha}_k(s) = \sum_{s'} \tilde{\gamma}_k(s', s)\tilde{\alpha}_{k-1}(s')$$

$$\tilde{\beta}_{k-1}(s') = \sum_{s} \tilde{\gamma}_k(s', s)\tilde{\beta}_k(s)$$

$$\tilde{\gamma}_k(s', s) = \exp\left[\frac{1}{2}L_e(u_k)u_k + \frac{1}{2}L_c X_k^s u_k + \frac{1}{2}L_c X_k^p c_k\right]$$

Where Xsk and Xpk are received soft inputs corresponding to systematic and parity bits of encoder, Lc and Le are channel reliability and extrinsic information bits. The main complexity arises due to calculation of α and β recursion units. Also we can note that it is not necessary to calculate every branch metric for all branches in ACS because few of them are same depending on the trellis structure. Based on previous metrics the new metrics are updated. One important thing to note is the processing speed can be further increased by N times by clubbing N stages of trellis into one higher order radix stage. Hence we are making use of radix 4 ACS unit built using two radix 2 ACS units.

Using the below max operation for metrics calculation,

$$\begin{aligned}\widehat{\max}(A, B) &= \log(\exp(A) + \exp(B)) \\ &= \max(A, B) + f(A, B),\end{aligned}$$

Where f(A,B) is

$$f(A, B) = \log(1 + \exp(|A - B|)),$$

Considering two nodes of trellis structure

$$\begin{aligned}\beta_{k-2}(0) = \max*\{&\beta_k(0) - \gamma_k(1), \beta_k(4) - \gamma_k(4), \\ &\beta_k(2) + \gamma_k(3), \beta_k(6) + \gamma_k(2)\}\end{aligned}$$

$$\begin{aligned}\beta_{k-2}(1) = \max*\{&\beta_k(0) + \gamma_k(4), \beta_k(4) + \gamma_k(1), \\ &\beta_k(2) - \gamma_k(2), \beta_k(6) - \gamma_k(3)\}\end{aligned}$$

Rewriting the above equations as follows

$$\beta_{k-2}(0) = \max*\{A, B\}$$

$$\beta_{k-2}(1) = \max*\{C, D\}$$

where A, B, C, and D can be written as

$$A = \max*\{\beta_k(0) - \gamma_k(1), \beta_k(4) - \gamma_k(4)\}$$

$$B = \max*\{\beta_k(2) + \gamma_k(3), \beta_k(6) + \gamma_k(2)\}$$

$$C = \max*\{\beta_k(0) + \gamma_k(4), \beta_k(4) + \gamma_k(1)\}$$

$$D = \max*\{\beta_k(2) - \gamma_k(2), \beta_k(6) - \gamma_k(3)\}.$$

First values of A & B are calculated using radix 2 and another radix 2 architecture is used to achieve βk-2(0) as shown in figure. Now we can observe that the distances between the input values A and C are equal, similarly for B and D. Hence the LUT and comparator units can be omitted for the calculation of C and D values. Thus leading to a novel architecture as shown in figure 3. This holds good for calculating other node values also.

IV. RESULTS

The verilog code for proposed turbo decoder is simulated using Modelsim SE 6.4C and synthesized using Xilinx 13.2 tool. The summary shows that the area required is less compared previous architecture. Here we have considered viterbi decoder.

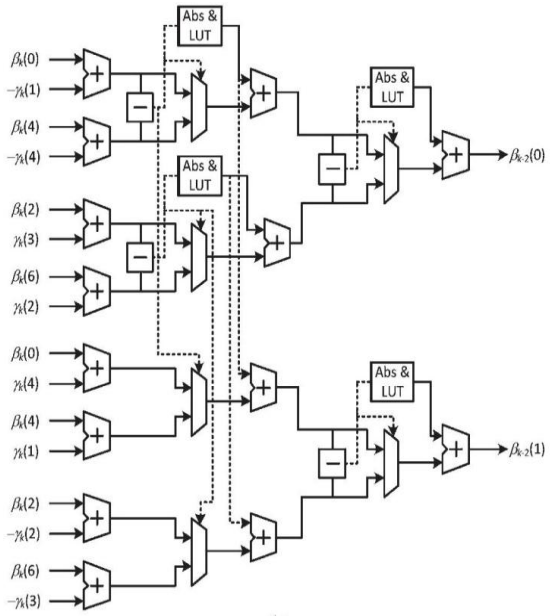

Fig 3: Proposed radix 4 ACS architecture for two concurrent metrics computation

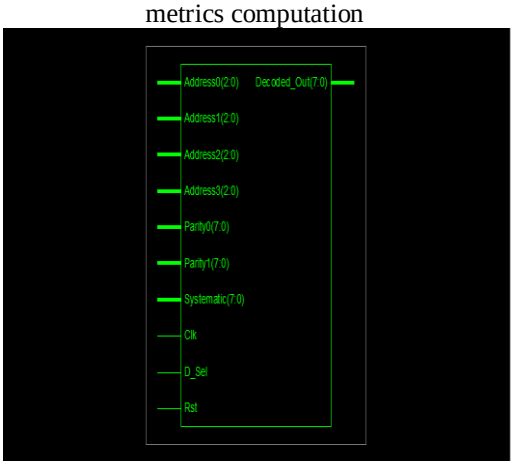

Fig 4: Schematic of proposed turbo decoder

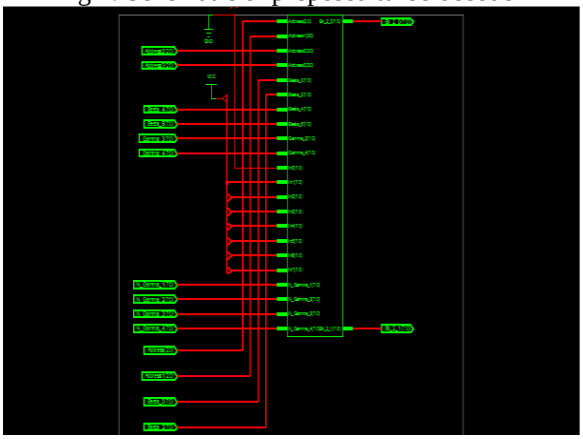

Fig 5: RTL schematic of proposed ACS unit

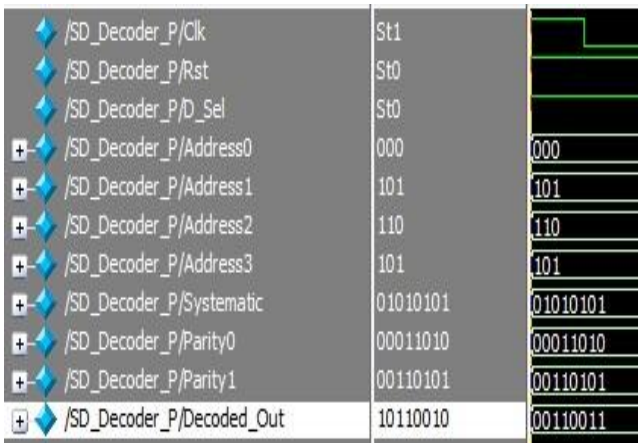

Fig 7: Simulation results of proposed turbo decoder

V. CONCLUSION

Turbo decoding is simulated in modelsim 6.4 and implemented Xilinx. An ACS unit is incorporated in the decoder along with only one comparator and LUT for calculation of two nodes values of trellis. Thus the usage of minimized number of components to decrease the area can be seen from the device utilization summary tabulated in table 1.

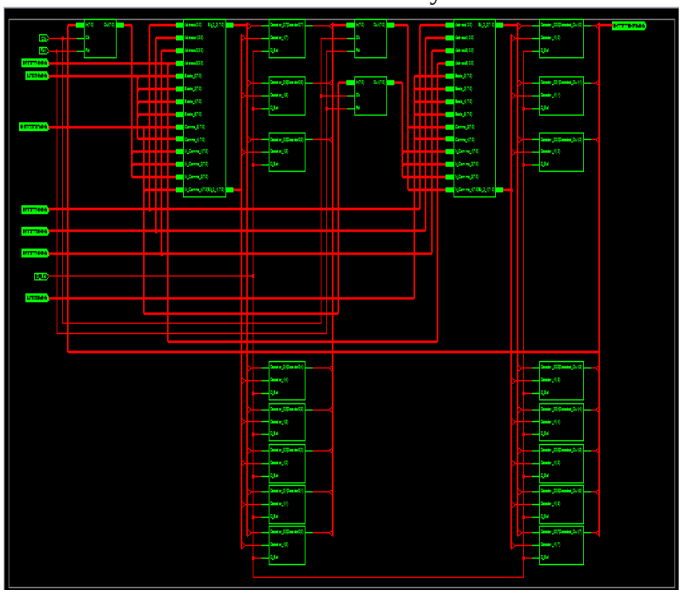
Fig 6: RTL schematic of proposed Turbo decoder

Logic utilization	Proposed	conventional	Available
Number of Slice flipflops	24	185	3840
Number of 4input LUT's	97	437	3840
Number of slices containing only related logic	57	239	239

Table 1: Comparsion of proposed and conventional turbo decoder

REFERENCES

[1] Arash Ardakani and Mahdi Shabany, "A novel area efficient VLSI architecture for recursion computation in LTE turbo decoders", IEEE Trans. On circuits and systems, vol. 62, no. 6, jun 2015.

[2] L. Li, R. Maunder, B. Al-Hashimi, and L. Hanzo, "A low-complexity turbo decoder architecture for energy-efficient wireless sensor networks," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 21, no. 1, pp. 14–22, Jan. 2013.

[3] C. Studer, S. Fateh, C. Benkeser, and Q. Huang, "Implementation tradeoffs of soft input soft-output MAP decoders for convolutional codes," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 59, no. 11, pp. 2774–2783, Nov. 2012.

[4] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang, "Design and implementation of a parallel turbo-decoder ASIC for 3GPP-LTE," IEEE J. Solid-State Circuits, vol. 46, no. 1, pp. 8–17, Jan. 2011.

[5] C.-C. Wong, M.-W. Lai, C.-C. Lin, H.-C. Chang, and C.-Y. Lee, "Turbo decoder using contention-free interleaver and parallel architecture," IEEE J. Solid-State Circuits, vol. 45, no. 2, pp. 422–432, Feb. 2010.

[6] S. Papaharalabos, P. Mathiopoulos, G. Masera, and M. Martina, "On optimal and near-optimal turbo decoding using generalized max operator," IEEE Commun. Let., vol. 13, no. 7, pp. 522–524, Jul. 2009.

