Area Efficient of Max Log Map Algorithm Using SB/DB Methods for Turbo Decoder

S.Ashwini[1], P.Maniraj Kumar[2], Ms.S.Umayal[3], Dr.S.Sutha[4]
PG scholar (M.E., AE), [1,2,3]PSNACollege of Engineering and Technology, Dindigul[1]
[4]Dept of EEE, University College of Engineering,Panruti,Tamilnadu,India[4]
[1]ashwinis9128@gmail.com,[2]mani1376@yahoo.com, [3]umayal.16@gmail.com

Abstract_ In Turbo decoding the design of Log MAP algorithm is more complicated with the exponential sum. In this paper a mathematical model is designed with a jacobian logarithm which consists of a max function along with the exponential correction function. The complexity of the jacobian logarithm is also reduced by the Max Log Map algorithm by eliminating the correction term. The MAP decoder is recursive and complex, which makes high-speed implementations. The block-interleaved pipelining (BIP) as a new high-throughput technique for MAP decoders. An area-efficient symbol-based BIP MAP decoder architecture is proposed by combining BIP with the well-known look-ahead computation. Due to the powerful error correcting performance, the turbo codes have been adopted in many wireless communication standards. The optimum decoding procedure for turbo codes is the maximum a posteriori (MAP) algorithm. It is very difficult to design high-speed MAP Decoder, because of its recursive computations. The MAP decoding algorithm has equipped to output soft-decision bits that turbo decoding requires. We propose maximum a posteriori algorithm (MAP) which can support both single-binary (SB) and double-binary (DB) CTC decoding It gives a mathematical description of the MLMAP algorithm for Single Binary and Double Binary turbo decoders.

Keywords__ **Block interleaved pipelining, convolutional Turbo codes,Max log maximum a posteriori,Single Binary and Double Binary**

I. INDRODUCTION

The convolutional turbo code (CTC) has been widely adopted as one of forward error correcting (FEC) schemes of wireless standards to have a reliable transmission over noisy channels. A soft-input-soft-output (SISO) equalizer can be combined with a SISO decoder to form a turbo equalizer. Turbo decoders are composed of two or more constituent SISO decoders, which correspond to the component codes, employed in the transmitter, and an interconnection of these constituent the decoders through an interleaver/deinterleaver. The decoding Algorithm employed in the constituent decoders is the maximum *a posteriori* probability (MAP) algorithm. The MAP algorithm provides a reliability metric, known as the log-likelihood ratio (LLR), on the transmitted code symbols.

The LLR output is employed by other constituent decoders, which attempt to Improve their LLR estimates iteratively. However, the use of iterative the processing results in a large computational and storage complexity and hence high power dissipation in the receiver. Several high throughput VLSI design Architectures of turbo decoders have already appeared. These high throughput architectures can be classified into parallel processing look-ahead computation and algorithm reformulation approaches. Therefore, the low-power and high-throughput implementations for turbo decoders have recently been investigated for wireless and broadband applications. In this paper, we propose area-efficient processor designs of the maximum *a* posteriori algorithm (MAP) for multi Standard CTC schemes the proposed MAP processor can be the deterministic or reconfigurable FEC components.

In this paper, we propose a new critical path reduction technique referred to as *block-interleaved pipelining* (BIP), which Leads to pipelined ACS kernel with less area overhead compared to conventional high - throughput architectures. Turbo equalizer, the symbol - based decoder Architecture enables us to fold the operations of the SISO MAP equalizer and SISO MAP decoder on to the same hardware platform leading to savings in area. We show that the throughput of the symbol-based decoder architecture can be improved further by applying the BIP technique. The warm-up free method is used in practice for parallel processing, where the data frame is divided into sub blocks that are processed concurrently. The warm-up free method suffers from bit error rate (BER) performance over a small number of iterations, but can overcome the degradation after five iterations. Different techniques such as pipelining, parallel processing, and look-ahead technique have been proposed to increase the throughput of log-MAP decoder.

The MAP algorithm which is used in the turbo decoders is operated in the logarithmic domain in order to reduce the computational complexity [1]. The computation of log likelihood ratio with the values of state metrics is more complicated with log exponential sum calculation. The log exponential sum can be simplified by Jacobian algorithm by adding a correction term along with the max operator. The correction function calculation is critical because of the performance and complexity of the turbo decoder. So a lot of methods [2][3][4] are used to simplify the computation to satisfy the performance requirements.

In Telecommuication a communications system is a collection of individual communications networks, transmission systems, relay stations, tributary stations, and data terminal equipment (DTE) usually capable of interconnection and interoperation to form an integrated whole. The components of a communications system serve a common purpose, are technically compatible, use common procedures, respond to controls, and operate in unison. Telecommunications is a method of communication

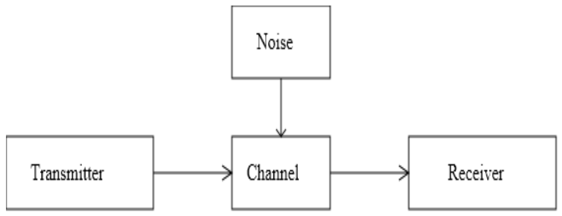


Fig 1:communication systems

In the modern communication systems, FEC (*Forward Error Correction*) is a very important module to protect data transmitted without error. Once the data encoded by the FEC encoding module, even the data is interfered by the noise in the transmitting channel, we can still correctly get the original data after we decode the received data by the FEC decoding module. Because of the rapid development of communication systems, the system requirement BER *(Bit Error Rate*) has became much higher.

II MAP DECODING

Architectures can be employed to implement the log-MAP algorithm however the trellis sweep over all observed symbols requires large memory in order to hold the forward and backward metrics until they are used in the LLR computation. Log-MAP decoding algorithm is derived via the property that the forward and backward metrics $\alpha_{k,}$ $\beta_{k.}$ The architecture has units for the computation of branch metrics (unit), one forward recursion (unit), two backward recursions and a buffer to store backward metrics, β_k and, β_k (unit), and the metric processing unit (MPU). The MAP algorithm provides a reliability metric, known as the log-likelihood ratio (LLR), on the transmitted code symbols. In general, the MAP is composed of branch metrics (γ), forward Recursion state metrics (α), backward recursion state metrics (, β), *a priori* LLR (^apr), *a posteriori* LLR (^apo).

The original derivation of the MAP algorithm was in the probability domain. The output of the algorithm is a sequence of decoded bits along with their reliabilities. This “soft” reliability information is generally described by the a posteriori probability (APP) P (u|y). For an estimate of bit u (- 1/ 1) having received symbol, we define the optimum soft output as

$$L(u) = \ln \frac{\mathrm{P(u = +1y)}}{\mathrm{P(u = -1y)}}$$

which is called the log-likelihood ratio (LLR). The LLR is a convenient measure, since it encapsulates both soft and hard bit information in one number. The sign of the number corresponds to the hard decision while the magnitude gives a reliability estimate. The original formulation of the MAP algorithm requires multiplication and exponentiation to calculate the required probabilities.

The MAP algorithm, in its native form, is challenging to implement because of the exponentiation and multiplication. If the algorithm is implemented in the logarithmic domain like the Viterbi algorithm, then the multiplications become additions and the exponentials disappear. Addition is transformed according to the rule .The additions are replaced using the Jacobi logarithm

$$\mathrm{MAX}^*(x.y) = \ln(e^x+e^y)$$
$$= \mathrm{MAX}\ (x.y) + \ln(1+e^{-x-y|})$$

Which is called the MAX* operation, to denote that it is essentially a maximum operator adjusted by a correction factor. The second term, a function of the single variable x-y, can be recalculated and stored in a small lookup table (LUT) [9]. The computational kernel of the MAP algorithm is the Add–MAX* operation, which is analogous, in terms of computation, to the Add–Compare–Select (ACS) operation in the Viterbi algorithm adjusted by an offset known as a correction factor. In what follows, we will refer to this kernel as ACSO (Add–Compare–Select–Offset).

The algorithm is based on the same trellis as the Viterbi algorithm. The algorithm is performed on a block of N received symbols which corresponds to a trellis with a finite number of stages N. We will choose the transmitted bit u_k from the set of {-1, +1}. Upon receiving the symbol y_k from the additive white Gaussian noise (AWGN) channel with noise variance σ^2 we calculate the branch metrics of the transition from state s’ to State s as

$$Gk(s', s) = \frac{-1}{2\sigma 2} \|y_k - C_k\ (s', s)\|^2$$

Where C_k(s’, s) is the expected symbol along the branch from State s’ to state s. The multiplication by -1/2 σ^2 can be done with either a multiplier or an LUT. Note that in the case of a turbo decoder which uses several iterations of the MAP algorithm, the multiplication by -1/2 σ^2 need only be done at the input to the first MAP algorithm.

The algorithm consists of three steps.

• Forward Recursion. The forward state metrics are recursively calculated and stored as

$$A_k(s) = \mathrm{MAX}^{*}{}_{s'}\ (A_k\text{-}1(s') + G_k(s', s))$$

Where K=1,.,., .N-1

The recursion is initialized by forcing the starting state to state 0 and setting
A_0 (0) =0
$A_0(s) = -\infty$, $S \neq 0$

• Backward Recursion. The backward state metrics are recursively calculated and stored as

$$B_{k-1}(s') = \mathrm{MAX}^*_s(B_k(s)+G_k(s',s))$$
Where k= N ...2.

The recursion is initialized by forcing the ending state to state 0 and setting
B_N (0) = 0
$B_N(s) = -\infty$, $s \neq 0$

The trellis termination condition requires the entire block to be received before the backward recursion can begin.

• Soft-Output Calculation. The soft output, which is called the LLR, for each symbol at time is calculated as

$$L\,(u_k) = \mathrm{MAX}_{(s',s)uk=+1}{*}(A_{k-1}(s')+G_k(s',s)+B_k(s)) - \mathrm{MAX}^{*}_{(s',s)uk=-1}(A_{k-1}(s')+G_k(s',s)+B_k(s))$$

Where the first term is over all branches with input label +1, and the second term is over all branches with input label -1. The MAP algorithm, as described, requires the entire message to be stored before decoding can start. If the blocks of data are large, or the received stream continuous, this restriction can be too stringent; "on-the-fly" decoding using a sliding-window technique has to be used. Similar to the Viterbi algorithm, we can start the backward recursion from the "all-zero vector" B^0 (i.e., all the components of B0 are equal to zero) with data {YK}, K from n down to n-L. Literations of the backward recursion allows us to reach a very good approximation of $g+B^{n-L}$ (where g is a positive additive factor).

This additive coefficient does not affect the value of the LLR. In the following, we will consider that after L cycles of backward recursion, the resulting state metric vector is the correct one. This property can be used in a hardware realization to start the effective decoding of the bits before the end of the message. The parameter L is called the convergence length. For on-the-fly decoding of non-systematic convolution codes as, five to ten times the constraint length was found to lead only to marginal signal-to-noise ratio (SNR) losses. For turbo decoders, due to the iterative structure of the computation, an increased value of L. might be required to avoid an error floor. A value of L=64 is reported for a recursive systematic code with a constraint length of five. In practice, the final value of L has to be determined via system simulation and analysis of the particular decoding system at hand.

III BLOCK INTERLEAVED PIPELINING TECHNIQUES

In this section, we present techniques for improving the throughput of recursive data-flow graphs. First, we review the existing techniques of parallel processing [25], [26] and look-ahead transform [39], [40]. Then, we present the BIP technique, which is one of the contributions of this paper. These techniques are applied to design high-throughput MAP decoder architectures

A. *parallel processing*

Pipelining or parallel processing becomes difficult for a recursive datapath. However, if the data is being processed in blocks and the processing satisfies the following two properties: 1) computation between blocks are independent and 2) computation within a block is recursive, then a block parallel processing architecture can be achieved.

Further, if a block can be segmented into computationally independent sub-blocks, parallel processing can be applied at the sub-block level leading to high-throughput architectures

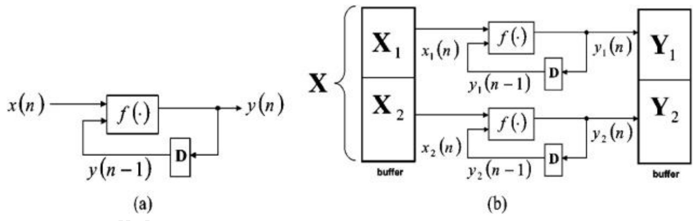


Fig: 2 parallel processing

B. *Blook ahead computation*

Another transform to achieve high-throughput for recursive data-flow graphs is look-ahead computation. Look-ahead leads to an increase in the number of symbols processed at each time step. Where two symbols are processed in one clock cycle

C.*Block interleaved pipelining*

To achieve an area-efficient implementation, we propose the block interleaved pipelining (BIP) technique. The BIP technique is applicable to general recursive data-flow graphs that satisfy the following two properties: 1) computation is block-based with computation between the blocks being independent and 2) operations within a block are recursive. Block computation can be partitioned into sub-block computation via algorithmic transforms while maintaining independence between sub-blocks, the BIP can be applied in a sub-block level.

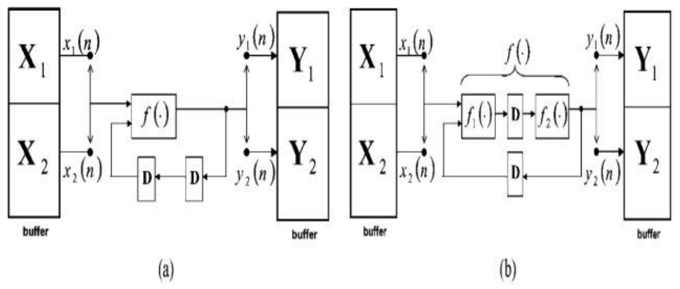


Fig: 3 BIP Architecture

VI. INTRODUCTION TO VARIOUS MAP ALGORITHMS

A. *MAP Algorithm*

MAP decoders are superior with respect to communications performance and for that reason preferred in advanced implementations. The MAP decoding algorithm is a recursive technique that computes the Log-Likelihood Ratio (LLR) of each bit based on the entire observed data block of length K. The MAP algorithm is used to calculate LogLikelihood Ratio (LLR) (Λ (D_k)) of the a posteriori probability (APP) of D_k = 1 to D_k = 0.

The state of the encoder is represented by S_k at time k. The state transition is from k-1 to k. The LLR value can be given by equation 1.

$$\Lambda(D_k) = \ln \frac{\sum_{S_k} \sum_{S_{k-1}} \gamma_1(y_k, S_{k-1}, S_k)\alpha_{k-1}(S_{k-1})\beta_k S_k}{\sum_{S_k} \sum_{S_{k-1}} \gamma_0(y_k, S_{k-1}, S_k)\alpha_{k-1}(S_{k-1})\beta_k S_k}$$

(1)

The forward state metric α_k can be expressed as

$$\alpha(S_k) = \frac{\sum_{S_{k-1}} \sum_{i=0}^{1} \gamma_i (y_k, S_{k-1}, S_k) \alpha_{k-1}(S_{k-1})}{\sum_{S_{k-1}} \sum_{i=0}^{1} \gamma_i (y_k, S_{k-1}, S_k) \alpha_{k-1}(S_{k-1})} \quad (2)$$

and the backward state metric β_k can be expressed as

$$\beta(S_k) = \frac{\sum_{S_{k+1}} \sum_{i=0}^{1} \gamma_i (y_{k+1}, S_k, S_{k+1}) \beta_{k+1}(S_{k+1})}{\sum_{S_k} \sum_{S_{k+1}} \sum_{i=0}^{1} \gamma_i (y_{k+1}, S_k, S_{k+1}) \alpha_k(S_k)} \quad (3)$$

The branch transition probabilities γ_i can be calculated by the equation (4).

$$\gamma_i((y_{sk}, y_{pk}), S_{k-1}, S_k) = e^{\sqrt{x_{sk}(L_e + (x_{sk}) + L_c(y_{sk}) + L_c y_{sk} x_{pk}}} \quad (4)$$

The channel reliability value $L_c = 2 / \sigma^2$ with σ^2 is the noise variance and L_e is the extrinsic information which gives a priori information.

B. *Log MAP Algorithm*

The implementation of MAP decoders in VLSI is more complex because of the recursive calculations. As pointed out in [10], it is mandatory to implement the MAP algorithm in the logarithmic domain (Log-MAP) to avoid numerical problems without degrading the decoding performance. So, this complexity in the MAP algorithm can be reduced by Log MAP algorithm. To avoid complicated calculations in the MAP algorithm, the entire MAP algorithm is calculated in Logarithmic domain. So the basic equations in the MAP algorithm (2), (3) and (4) are converted into log domain which reduces the number of strong operators such as multiplications and adds the weaker operations such as additions used.

So, $\alpha_k (S_k) = \ln \alpha_k(S_k)$.

$$\overset{\Lambda}{\alpha}(S_k) = \ln(\sum_{S_{k-1}} \sum_{i=0}^{1} e^{\ln(\gamma_i (y_{sk}, y_{pk}) S_{k-1}, S_k) + \ln \alpha_{k-1}(S_{k-1})}) - \sum_{S_k} \sum_{S_{k-1}} e^{\ln(\gamma_i (y_{sk}, y_{pk}) S_{k-1}, S_k) + \ln \alpha_{k-1}(S_{k-1})} \quad (5)$$

$$\overset{\Lambda}{\alpha}(S_k) = \ln(\sum_{S_{k-1}} \sum_{i=0}^{1} e^{\ln(\gamma_i (y_{sk}, y_{pk}) S_{k-1}, S_k) + \ln \alpha_{k-1}(S_{k-1})}) - \sum_{S_k} \sum_{S_{k-1}} e^{\ln(\gamma_i (y_{sk}, y_{pk}) S_{k-1}, S_k) + \ln \alpha_{k-1}(S_{k-1})} \quad (6)$$

The above equation (5),(6) and (7) can be simplified into general form, which is given by the equation (8).

$$F(x_1, x_2, x_3 \ldots x_N) = \ln(\sum_{i=1}^{n} e^{x_i}) \quad (7)$$

The jacobian logarithm for two variable expressions is given by

$$\max{}^*(x_1, x_2) = \ln(e^{x1} + e^{x2}) = \max(x_1, x_2) + \ln(1 + e^{-|x_1 - x_2|})$$
$$= \max(x_1, x_2) + C(x) \quad (8$$

The calculation of the correction term C(x) in (9) is more complicated in Log MAP algorithm. The same form of log exponential sum occurs in the calculation of backward state metrics (β_k) , and LLR Λ (D_k).So, to improve the performance of the Log Map Algorithm, a simple method of implementation of correction function must be required. Various methods of correction implantation is discussed in [6][7][8][9]. The easiest method of finding out the correction function is presented in [10].But this requires more memory to store the table sequences.

C. *Max Log MAP Algorithm*

The max log map algorithm introduces an approximation $\max{}^*(x_1,x_2) \approx \max(x_1,x_2)$. This Max log map algorithm simply omitting the correction term C(x).But the performance of the Log Map algorithm is dropped by 10% compared to Log Map algorithm. Even though the Max Log Map algorithm is least complex, it gives worst BER. So, in Max Log Map algorithm, a simple implementation of correction function is required to improve the performance of the turbo decoders.

D. *Different Algorithms for Correction Approximation*

In the Linear Log Map algorithm proposed in [8], the MacLaurin Series expansion is used and it is observed that the correction function is approximately zero.

$$C(x) \approx \max(0, \ln 2 - \frac{1}{2} x) \quad (9)$$

This approximation offers a good result than the Constant Log Map Algorithm presented in [9] where C(x) =

$$\begin{cases} \frac{3}{8}, -2 \le x \le 2 \\ 0, otherwise \end{cases} \quad (10)$$

An accurate method of approximation proposed in [10] is Multistep Log Map Algorithm.

$$C(x) \approx \frac{\ln 2}{2^{[x+0.5]}} \quad (11)$$

This method of correction is more accurate. This method of algorithm employs with shift registers storing the ln(2).For fast computation , a speed registers are required for this algorithm. Figure 3 shows the graphical comparison of various Approximations to the correction function.

V.HIGH THROUGHPUT MAP DECODER ARCHITECTURE

A.*BIP Architecture*

Following the approach described in Section III-C, the BIP architecture is obtained by processing M sub-blocks of size M\N. For simplicity, we refer to this architecture as a BIP architecture even though the interleaving is done at the sub-block level and we assume that the block-size and the sub-blocksize is a multiple of the warm-up period . the BIP architecture, where the α,β ,γ andMPUs are pipelined in order to reduce the critical path delay(N)and delay-line blocks are required to compute sub-blocks(M) in a block-interleaved order. Theα and β units have the folded
block-interleaved ACS architecture shown in Fig. , wherethe block-interleaving factor is M. The BIP architecture processes trellis sections K,K+(N\M),K+2(N\M)….K+(M-1)(N\M) in consecutive clock cycles

Thus, retiming can be employed to pipeline the critical path by levels in order to achieve speed-up by afactor of . The price we pay is an increase in the pipeliningregisters. Hence, the BIP architecture will consume less areathan the parallel architecture.

We apply an -level look-ahead transform to theα and β recursions.Doing so enables us to compute α and β the and metricsby merging trellis sections and to decode bits per each clockcycle [40]. We refer to this approach as *symbol-based* MAP decoding. For simplicity, a 2-bit symbol decoding, where two trellis sections are grouped is described next. Extensionto the multi-bit symbol case is straightforward.

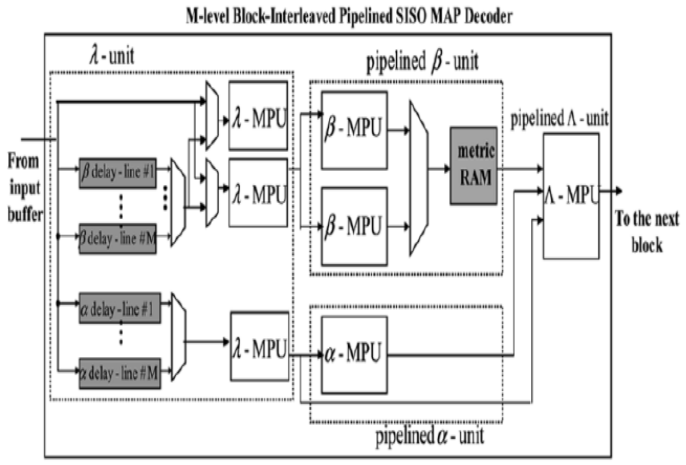


Fig: 4 M Level BIP Architecture

B. .*Pipelined Metric Architecture*

In order to compute the LLR values forward (α) and backward (β) states, and branch metric (γ) values of all states are required. Figure 4 shows the metric pipelined architecture unit. This proposed architecture consists of three adders, a subtractor, a comparator, one Selection MUX, Trellis MUX and logic. Since, the add-compare select operation is more complex than Trellis MUX, a pipelined architecture proposed as shown in Figure

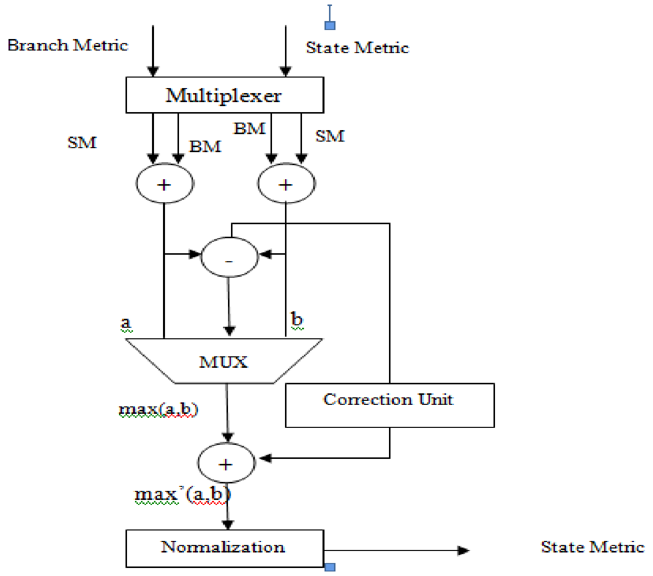


Fig: 5 pipelined architecture

VI. PROPOSED LLR UNIT

A.*LLR (LOG-LIKELIHOOD RATIO*)

To achieve a high hardware usage of the dual-mode LAPO, the *a-posteriori* LLR are rewritten. By rewriting MAX operation $\phi s_{(z')}$ is defined as

$$\Phi_{s(00)} = \mathrm{MAX}_{S_{k-2},S_k,u_{k-1}=0,u_k=0}\,(\alpha_{k-2}(S_{k-2})+\gamma_k(S_{k-2},S_k)+\beta_k(S_k))$$
$$\Phi_{s(01)} = \mathrm{MAX}_{S_{k-2},S_k,u_{k-1}=0,u_k=1}\,(\alpha_{k-2}(S_{k-2})+\gamma_k(S_{k-2},S_k)+\beta_k(S_k))$$
$$\Phi_{s(10)} = \mathrm{MAX}_{S_{k-2},S_k,u_{k-1}=1,u_k=0}\,(\alpha_{k-2}(S_{k-2})+\gamma_k(S_{k-2},S_k)+\beta_k(S_k))$$
$$\Phi_{s(11)} = \mathrm{MAX}_{S_{k-2},S_k,u_{k-1}=0,u_k=1}\,(\alpha_{k-2}(S_{k-2})+\gamma_k(S_{k-2},S_k)+\beta_k(S_k))$$

Where $Z'=(u_{k-1},u_k)$for the radix-4 SB MAP decoding. Then, $A_{apo,k-1}(u_{k-1})$ and $A_{apo,k}(u_k)$ are computed by

$$A_{apo,k-1}(u_{k-1}) = \mathrm{MAX}(\phi_{s(10)},\phi_{s(11)}) - \mathrm{MAX}(\phi_{s(00)},\phi_{s(01)})$$
$$A_{apo,k}(u_k) = \mathrm{MAX}(\phi_{s(01)},\phi_{s(11)}) - \mathrm{MAX}(\phi_{s(00)},\phi_{s(10)})$$

By rewriting MAX operations $\phi d_{(z)}$ is defined as

$$\Phi_{d(00)} = \mathrm{MAX}_{S_{k-1},S_k,u_k=00}\,(\alpha_{k-1}(S_{k-1})+\gamma_k(S_{k-1},S_k)+\beta_k(S_k))$$
$$\Phi_{d(01)} = \mathrm{MAX}_{S_{k-1},S_k,u_k=01}\,(\alpha_{k-1}(S_{k-1})+\gamma_k(S_{k-1},S_k)+\beta_k(S_k))$$
$$\Phi_{d(10)} = \mathrm{MAX}_{S_{k-1},S_k,u_k=10}\,(\alpha_{k-1}(S_{k-1})+\gamma_k(S_{k-1},S_k)+\beta_k(S_k))$$
$$\Phi_{d(11)} = \mathrm{MAX}_{S_{k-1},S_k,u_k=11}\,(\alpha_{k-1}(S_{k-1})+\gamma_k(S_{k-1},S_k)+\beta_k(S_k))$$

Where $z=u_k$ for the radix-4 SB MAP decoding.
Then, Aapo, k $^{(z)}$ is computed by

$$A_{apo,k}^{(00)}(u_k)=0$$
$$A_{apo,k}^{(01)}(u_k)= \Phi_{d(01)} - \Phi_{d(00)}$$
$$A_{apo,k}^{(10)}(u_k)= \Phi_{d(10)} - \Phi_{d(00)}$$
$$A_{apo,k}^{(11)}(u_k)= \Phi_{d(11)} - \Phi_{d(00)}$$

Based on the observation in the MAX operation of $\phi_{s(z')}$ and $\phi_{d(z)}$ are the same. Therefore, the computational hardware of the dual-mode LAPO can be shared. Shared. Fig. 12 shows the block diagram of the dual-mode LAPO. The MAX $^{(z)}$ module is composed of 3-level binary-tree-structured compare select units (CSUs). The path LL $^{(z)}$ is the output of the shared MAX operations of $\phi s_{(z')}$ and $\phi_{d(z)}$.

The LLR calculator performs three subtractions and two subtractions and four MAX operations . Note that a MAX operation is implemented by a CSU. Thus, the subtracted in a CSU can be the subtraction of $A_{apo,k}{}^{(z)}$. The shared hardware design of the dual-mode LLR calculator is shown in fig. 13 and fig 14. The signal "Mode" is used to configure the dual-mode LLR calculator. When "Mode" is active low, the dual-mode LLR calculator is in SB mode.

In order to compute the LLR values forward (α) and backward (β) states, and branch metric (γ) values of all states are required. Figure shows the metric pipelined architecture unit. This proposed architecture consists of three adders, a subtractor, a comparator, one Selection MUX, Trellis MUX and logic. Since, the add-compare select operation is more complex than Trellis MUX, a pipelined architecture proposed as shown in Figure

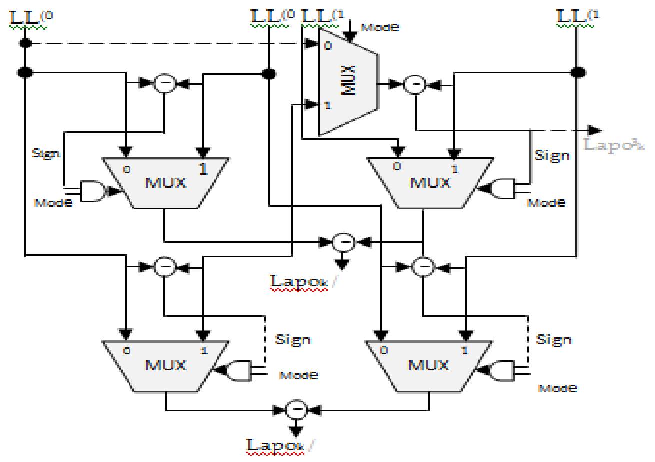


Fig:6 SB\DB MAP LLR calculation of dual mode
a)Single Binary mode

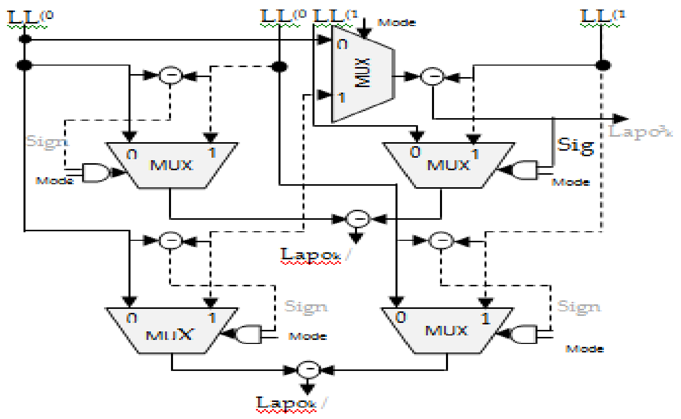
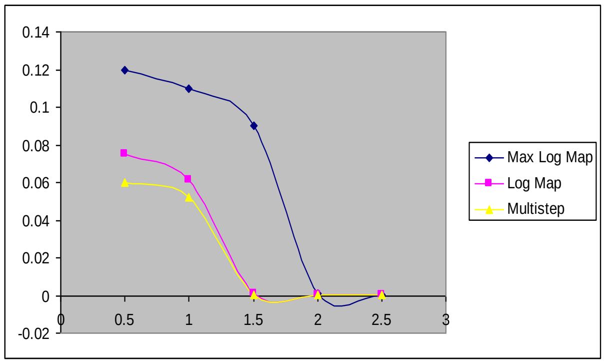


Fig: 7(b), double binary mode

VII. CONCLUSION

The optimum performance of Log Map Algorithm involves complex operations. By neglecting the correction function in the Max Log Map algorithm gives the capacitive loss and hence the correction term must be added which gives the optimal Log Map algorithm. The shared dual mode MAP decoder achieves low computational costs and low storages. This MAP processor achieves high throughput rates and high area efficiency.



REFERENCES

[i] *C. Berrou A Glavieux, and P. Thitimajshima "Near Shannon limit error-correcting coding And decoding Turbo codes," in Proc IEEEInt. Conf Communications, Geneva, Switzerland, May 1993pp, 1064–1070*

[ii] *S. Benedetto et al., "A soft-input soft-output Maximum a posterior r (MAP) module to Decode parallel and serial concatenated codes,"Jet Propulsion Lab,Pasadena,CA,TDA Progress Rep. 42-127, 1996*

[iii] *R. G. Gallager, Low-Density Parity-Check Codes. Cambridge, MA: MIT Press, 1963.*

[iv] *C. Douillard, M. Jezequel, C. Berrou, A. Picart, P. Didier, and A. Glavieux, "Iterative correction of intersymbol interference:Turbo equalization," Eur. Trans. Telecommun., vol. 6, pp. 507–511, Sep.–Oct. 1995*

[v] *Z.Wu and J. M. Cioffi, "Turbo decision aided equalization for magnetic recording channels," in Proc . Global Tele communication Conf., 1999, pp. 733–738.*

[vi] *Z. Wang, H. Suzuki, and K. K. Parhi, "VLSI Implementation issues of turbo decoder design for wireless applications," Proc. IEEE Signal Processing Systems (SiPS): Design and Implementation, pp. 503–512, Oct. 1999.*

[vii] *Z. Wang, Z. Chi, and K. Parhi,"Area efficientHigh – speed decoding schemes for turbo Decoders ," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 10, no. 6, pp. 902– 912, Dec. 2002.*

[viii] *B. Bougard, A. Ciulietti, L. V. d. Perre, and F.Catthoor, "A class ofpower efficient VLSI architectures for high speed turbo-decoding," inProc. IEEE Global Telecommunication Conf., vol. 1, 2002, pp. 553–549.*

[xi] *A. Giulietti et al., "Parallel turbo code interleavers: Avoiding collisions in accesses to storage elements," Electron. Lett., vol. 38, no. 5, pp. 232–234, Feb. 2002.*

[x] *T. Miyauchi, K. Yamamoto, T.Yokokawa,"High- performance programmable SISOdecoder VLSI implementation for decoding turbocodes," in Proc. IEEE Global Tele communications Conf., vol. 1, 2001 pp. 305–309.*