A High-Troughput Radix-4 Log-MAP Decoder With Low Complexity LLR Architecture

Hsiang-Tsung Chuang, Kai-Hsin Tseng and Wai-Chi Fang
Department of Electronics Engineering
National Chiao Tung University
Hsinchu, Taiwan, R.0.C
Mail:{zsc9411647.ee94g@nctu.edu.tw
kinesing@gmail.com
dr.wfang@gmail.com}

Abstract—**The throughput of turbo decoder is limited by the recursion architecture. In this paper, an improved radix-4 recursion architecture is presented. In order to decrease the critical path delay, a hybrid 4-inputs addition/subtraction structure is employed. Moreover, we present a modified trace-back architecture to decrease the hardware complexity of the log-likelihood ratios (LLR) architecture. The area of the proposed MAP decoder is 0.58 mm^2 on UMC 0.13μm standard cell technology and under the worst case a maximum throughput of 600 Mbps can be achieved.**

I. INTRODUCTION

Turbo code [1] was invented in 1993 by Berrou, Glavieux and Thitimajshima. It has outstanding error correction performance. Special features of turbo code are as follows: (1) Turbo encoder is composed of two parallel-concatenated recursive systematic convolutional code (RSC) with a large block size. (2) A pseudo random interleaver is used to re-permute the input sequence for the second RSC encoder. (3) Turbo decoder uses the maximum a posterior probability (MAP) algorithm. (4) The iterative technology is used. Those features make turbo decoder has a great ability for error correcting and almost near the Shannon capacity limit. Owing to its error correct ability, it has been applied widely for various communication standards (e.g., 3GPP [2], Consultative Committee for Space Data Systems (CCSDS) [3]).

However, in a hardware perspective, the throughput of turbo decoder is limited by the recursion architecture inherent with the MAP algorithm. In order to increase the throughput rate, radix-4 MAP decoder architecture could be used. In our design, an offset-add-compare-select (OACS) [4] architecture is employed to replace the conventional add-compare-select-offset (ACSO) architecture to decrease the critical path delay of the decoder. To further improve the throughput, we use a hybrid addition/subtraction structure to determine each two survivors.

Furthermore, due to the radix-4 organization, the hardware cost of the LLR architecture is substantially increased compared to the radix-2 counterpart. In this article, we present a trace-back architecture that can decrease the hardware cost of the radix-4 LLR architecture. The paper is organized as follows. Section II presents the MAP algorithm. Section III illustrates the proposed decoder structure, recursive and LLR architecture. The simulation and implementation result is described in Section IV. Eventually, the conclusion is given in Section V.

II. TURBO DECODING ALGORITHM

In this section, we only consider the MAP algorithm in logarithm domain which is called the log-MAP algorithm [5]. The log-MAP algorithm is designed to produce the LLR of each information bit μ_k at time k, as defined:

$$L(u_k) = \max_{(s',s)\in u_k=+1}{}^{*}\left[\alpha_{k-1}(s')+\gamma_k(s',s)+\beta_k(s)\right] - \max_{(s',s)\in u_k=-1}{}^{*}\left[\alpha_{k-1}(s')+\gamma_k(s',s)+\beta_k(s)\right] \quad (1)$$

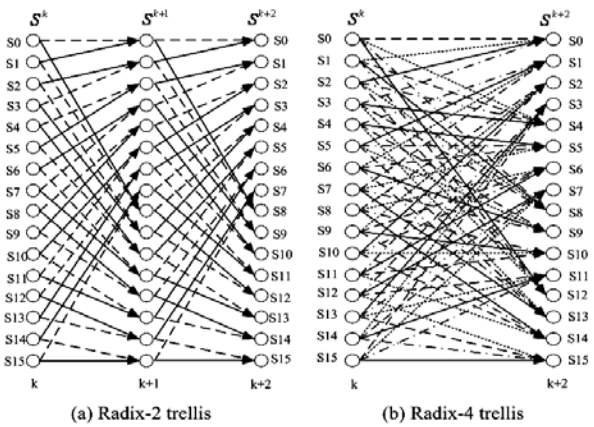

Figure 1. Trellis diagram for 16 state RSC encoder (a) radix-2 and (b) radix-4 trellis

where s is the state of the trellis diagram in Figure 1 (a), and the forward recursion metric $\alpha_k(s)$, the branch metric $\gamma_k(s', s)$ and the backward recursion metric $\beta_k(s)$ in logarithm domain are defined:

$$\alpha_k(s) = \max_{s'}{}^{*}(\gamma_k(s',s)+\alpha_{k-1}(s')) \quad (2)$$

$$\beta_k(s) = \max_{s''}{}^*(\gamma_{k+1}(s,s'') + \beta_{k+1}(s'')) \tag{3}$$

$$\gamma_k(s',s) = \frac{1}{2}\cdot\left(u_k \cdot L_a(u_k) + L_c \cdot \left(y_k^s \cdot x_k^s + y_k^p \cdot x_k^p\right)\right) \tag{4}$$

where the term $L_C = 4E_s/N_0$ is called channel reliability. x_k^s and x_k^p are the outputs from the two RSC encoders, respectively. After BPSK modulation and transfer through the channel, y_k^s and y_k^p are the received symbols of x_k^s and x_k^p. Furthermore, the Jacobian function is defined in (5), where the term $lut(x_1, x_2)$ is the correct term of the function. In hardware design, it is mostly implemented by a look-up-table (LUT).

$$\begin{aligned} \ln(e^{x_1} + e^{x_2}) &\triangleq \max^*(x_1, x_2) \\ &= \max(x_1, x_2) + \ln(1 + e^{-|x_1 - x_2|}) \\ &= \max(x_1, x_2) + lut(x_1, x_2) \end{aligned} \tag{5}$$

The principle of the radix-4 MAP algorithm is derived by ignoring the even time recursive state metrics as shown in Figure 1(b). That is, the radix-4 MAP decoder calculates two bits per clock cycle, whereas the radix-2 MAP decoder calculates one bit per clock cycle. Then the forward recursion metric and the backward recursion metric can be represented by the following equations:

$$\alpha_k(s) = \max_{(s',s'')}{}^*[\gamma_k(s',s) + \gamma_{k-1}(s'',s') + \alpha_{k-2}(s'')] \tag{6}$$

$$\beta_{k-2}(s'') = \max_{(s,s')}{}^*[\gamma_{k-1}(s'',s') + \gamma_k(s',s) + \beta_k(s)] \tag{7}$$

Finally, the LLR can be written as:

$$\begin{aligned} L(u_k) = &\max_{(s'',s',s)\in u_k=+1}{}^*\{\alpha_{k-2}(s'') + \gamma_{k-1}(s'',s') + \gamma_k(s',s) + \beta_k(s)\} \\ &- \max_{(s'',s',s)\in u_k=-1}{}^*\{\alpha_{k-2}(s'') + \gamma_{k-1}(s'',s') + \gamma_k(s',s) + \beta_k(s)\} \end{aligned} \tag{8}$$

III. MAP Decoder Architecture

The block diagram of the radix-4 MAP decoder is shown in Figure 2. During the decoding process, the soft-input symbols are written to the two-port RAM, and are read by the ACS α or β block to calculate the branch and state metrics. The state metrics computed from the ACS α block, are stored in "Alpha RAM", and are later fetched by the LLR unit for LLR calculation when the ACS β block state metrics become available. In order to decrease the latency and memory, the dummy β ACS block fetches the soft-input symbols directly.

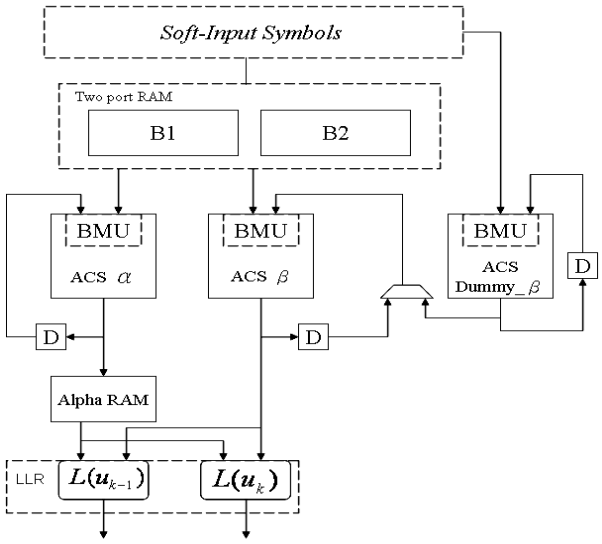

Figure 2. Block diagram of radix-4 MAP decoder

A. Sliding Window and Memory optimization

In a traditional sliding window [5], four two port memory banks are needed. Besides, the latency of the traditional sliding window is at least 4L, where L is the window size. In our architecture, the function for each memory-bank is illustrated in Figure 3 (a). The red-bias block represents a store of the soft-input symbols to the memory bank, and the blue-dotted block represents a read of the soft-input symbols to compute the forward state metrics α; the green-bias block represents first a read of the previous soft-input symbols to compute the backward state metrics β followed by an immediate store of the next input received symbols. A detailed dynamic description is also illustrated in Figure 3 (b). The green solid arrow in Figure 3 (b) represents calculation of the backward state metrics β. Furthermore, the dummy-β is calculated directly from the input symbols without the use of any memory-banks. Based on the above reason, two memory banks are enough for our sliding window method. Once we have the forward and backward state metrics, the soft output calculator is employed to decode the LLR out. Therefore, the latency of our proposed SW method is only about 2L.

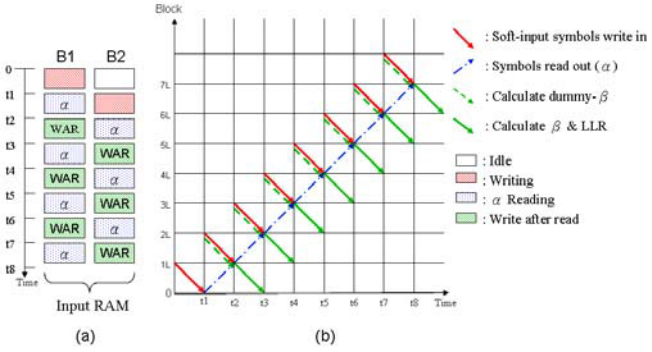

Figure 3. Space and time relationship of (a) memory-bank management (b) dynamic dataflow of sliding window

B. ACS Architecture

In the following, we discuss the radix-4 architecture. In equation (6), (7), the Jacobian function (i.e. *max*(w, x, y, z)*) has four inputs. If we directly implement this function, the critical path delay will be twice that of the radix-2 recursion unit [8], therefore, it can not increase the throughout. An approximation method can be implemented in [6], but result in a little performance loss as shown in Figure 4 and equation (9).

$$\begin{aligned} &\max^*(w,x,y,z) = \max^*\big(\max^*(w,x),\max^*(y,z)\big) \\ &\simeq \max^*\big(\max(w,x),\max(y,z)\big) \\ &= \max\big(\max(w,x),\max(y,z)\big) + lut\big(\max(w,x),\max(y,z)\big) \end{aligned} \tag{9}$$

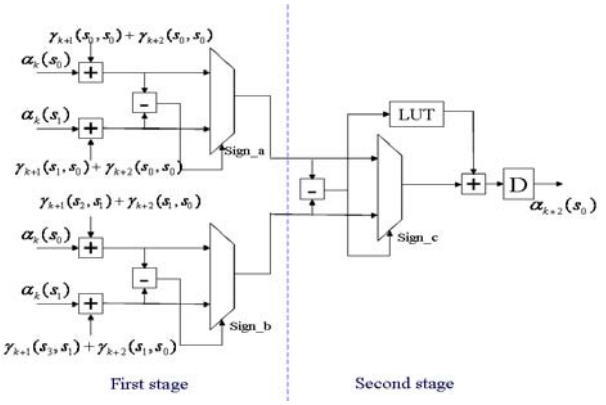

Figure 4. Radix-4 recursion unit was proposed in [6]

In order to further increase the throughput, we use the OACS [4] architecture as shown in Figure 5 and the computation for the Jacobian function can be expressed as [7]:

$$\begin{aligned} &\max^*\big(w,x,y,z\big) \simeq \max\big(\max(w,x),\max(y,z)\big) \\ &+ lut(\max(w,x),\max(y,z)) + sel(lut\big(w,x\big),lut\big(y,z\big)\big) \end{aligned} \tag{10}$$

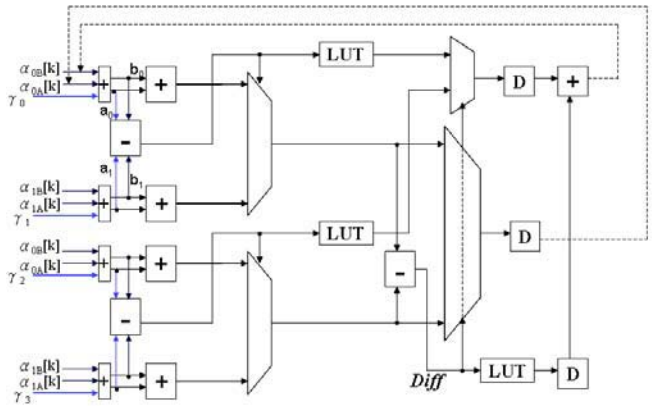

Figure 5. Proposed Radix-4 recursion unit

where the *sel* function outputs lut(w, x) when max(w, x) is larger than max(y, z), and lut(y, z) otherwise.

First, we introduce a one stage carry-save structure described in [8] to reduce a three-number addition to a two-number addition (i.e. the small addition block shows in Figure 5). Second, in order to further increase the throughput, a hybrid 4-inputs addition/subtraction structure is illustrated in Figure 6. Due to the subtraction is computed after the one stage carry-save adder instead of the second adder, the throughput is increased clearly. Since the delay from the LUT block is less than the delay of the adder, the critical path delay of our design is around three times the delay of a 9-bits adder.

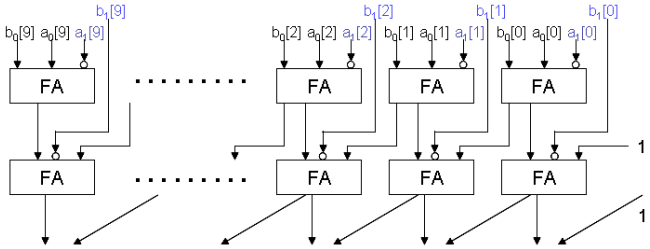

Figure 6. hybrid 4-inputs addition/subtraction structure

C. LLR Unit

In a radix-4 decoder the LLR unit in Figure 7 (a) has double the hardware complexity compared to that of a radix-2 decoder. This is because in a radix-4 LLR architecture, each $\beta_{k+1}(s')$ nodes needs to be calculated from two $\beta_{k+2}(s'')$ nodes and their corresponding backward paths as shown in left diagram of Figure 7 (b), which is not required in a radix-2 LLR unit. Thus, in a radix-4 trellis as shown in Figure 7 (b) right-hand side, the number of paths in a single stage is doubled compared to that of a radix-2 trellis.

In our design, we introduce a modified trace-back technique in order to address this drawback. According to the trace-back [9] method, we use state metric $\beta_k(s)$ to derive the two paths as shown in Figure 7 (c) and:

$$path0 = \beta_k(s_2) - lut\big[(\beta_{k+1}(s_1)+\gamma_{k+1}(s_2,s_1)),(\beta_{k+1}(s_3)+\gamma_{k+1}(s_2,s_3))\big] \tag{11}$$

$$path1 = \beta_k(s_2) - lut\big[(\beta_{k+1}(s_1)+\gamma_{k+1}(s_2,s_1)),(\beta_{k+1}(s_3)+\gamma_{k+1}(s_2,s_3))\big] - Diff \tag{12}$$

The two paths in Figure 7 (b) correspond to the orange arrows and nodes, and have values $\beta_{k+1}(s_1) + \gamma_{k+1}(s_2, s_1)$ and $\beta_{k+1}(s_3) + \gamma_{k+1}(s_2, s_3)$, respectively. Then the survivor0 and survivor1 are decided by the sign bit of Diff; If sign bit of Diff is 1, survivor0 assumes the value of path1 and survivor1 the value of path0, otherwise survivor0 assumes the value of path0 and survivor1 that of path1. As mentioned above, one adder is enough to add the two inputs (i.e. survivor and $\alpha_k(s)$) instead of the two required in the traditional radix-4 LLR unit. Thus, due to the modified trace-back technique, the hardware cost of the proposed architecture in Figure 7 (d) is greatly reduced compared to the traditional radix-4 LLR unit.

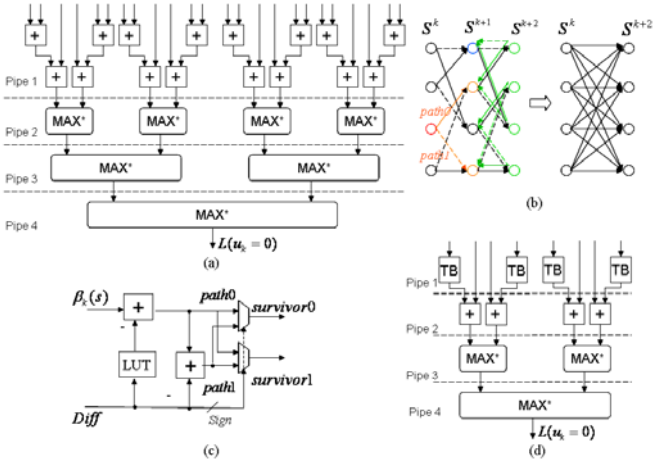

Figure 7. (a) traditional radix-4 LLR unit (b) radix-2 and radix-4 trellis diagrams (c) modified trace-back unit (d) proposed LLR architecture with TB (trace-back) unit

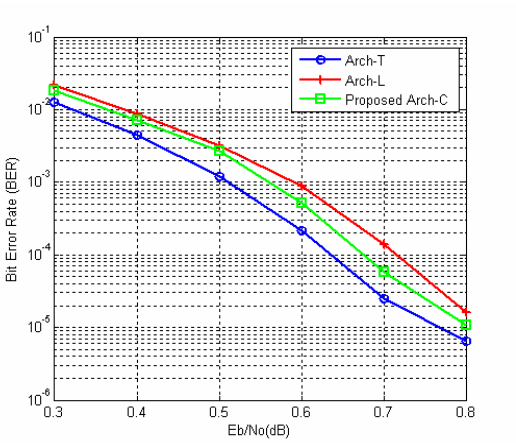

Figure 8. Performance comparison among those three architectures

IV. Simulation and Implementation Results

We synthesize four kinds of recursion units using UMC 0.13µm standard cell library and compare their timing and area results. The synthesis result is shown in Table 1, where *(q, f)* denotes a quantization scheme that uses *q* bits in total and *f* bits to represent the fractional part. Arch-T denotes the traditional radix-2 ACSO architecture; Arch-L the radix-4 architecture shown in Figure 5; Arch-W the radix-4 architecture proposed by Wang [8]; and Arch-C the proposed architecture, having the highest throughput among all recursion units.

We also simulate three types of turbo decoders under MATLAB for BER comparison. Figure 8 shows the BER performance of a code rate 1/3, 16 states, and frame size of 1784 bits on CCSDS standard. The number of total iteration is eight. The MATLAB simulations were operating under the assumption of AWGN channel and BPSK modulation. We could see that the traditional radix-2 architecture has the best performance due to least approximation, and the other two approximation architecture: Arch-L and Arch-C resulting to about 0.1 and 0.05dB performance loss, respectively. Table 2 shows the area and power consumption of two LLR architectures operating at 200MHz. (note: only $L(u_k)$ computation uses the modified TB architecture, while the $L(u_{k+1})$ computation uses a traditional LLR architecture) Compared to a traditional radix-4 LLR unit, the hardware cost of the proposed LLR unit is lower by around 19%.

TABLE I. Comparison of Four Recursion Architectures

	Timing (ns)	Relative Area	Relative throughput	Quantization
Arch-T	3.00	1	1	Receiver input: (5, 2)
Arch-L	4.02	1.55	1.49	Extrinsic information: (6, 2)
Arch-W	3.67	1.70	1.64	Branch metric: (7, 2)
Arch-C	3.18	2.01	1.89	State metric: (9, 2)

TABLE II. The Area and Power Comparison of a SISO Decoder for CCSDS Standard. (UMC 0.13 μm process, @200MHz)

	Area (Gate count)	Power (mw) @200MHz
Traditional radix-4 LLR	53.15k	1.042
Proposed LLR	43.19k	0.822

V. Conclusions

In this paper, a high-throughput radix-4 recursion architecture is presented. In order to decrease the critical path delay, Arch-C uses a hybrid addition/subtraction structure. Moreover, due to the trace-back method, the hardware complexity of the proposed LLR unit is lower than traditional radix-4 LLR unit by around 19%. After static timing analysis and post layout simulation, the area of our MAP decoder is 0.58 mm^2 and under the worst case a maximum throughput of 600 Mbps can be achieved.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. IEEE Int. Conf. Communications*, 1993, pp. 1064–1070

[2] 3GPP Specifications. 3rd generation partnership project. [Online]. Available: http:// www.3GPP.org

[3] J. B. Berner and K. S. Andrews, "Deep Space Network Turbo Decoder Implementation," in *Proc. IEEE Aerospace Conf.*, vol. 3, 10-17 March 2001, pp. 1149-1157.

[4] E. Boutillon,W.J. Gross and P.G. Gulak, "VLSI architectures for the MAP algorithm," *IEEE Transactions on Communications*, vol. 51(2), pp. 175 - 185, Feb. 2003.

[5] A. J. Viterbi, "An intuitive justification of the MAP decoder for convolutional codes," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 260–264, Feb. 1998.

[6] M. Bicherstaff, L. Davis, C. Thomas, D. Garrett, and C. Nicol, "A 24Mb/s Radix-4 LogMAP Turbo Decoder for 3GPP-HSDPA Mobile Wireless," in *IEEE ISSCC Dig. Tech. Papers*, 2003, pp. 150 – 151.

[7] C. Zhang, X, Wang, F. Ye and J. Ren, "A 400Mb/s Radix-4 MAP Decoder with Fast Recursion Architecture", in *IEEE ICACT,* pp.1339-1342, Feb. 17-20, 2008

[8] Z. Wang, "High-speed recursion architecture for MAP-based Turbo decoders", in *IEEE Trans. VLSI Syst,* vol 14, No. 4, pp. 470-474, April 2007

[9] T.-H. Tsai, C.-H. Lin, and A.-Y. Wu, "A memory-reduced log-MAP kernel for turbo decoder," in *Prof. IEEE ISCAS*, 2005, pp. 1032-1035.