The Improvement of ACSU of Viterbi Decoder for a Long Constraint Length with Soft-decision

M.H. Jing, Y.H. Chen, T.K. Truong *IEEE Fellow* and P.H. Huang

Department of Information Engineering
I-Shou University
Kaohsiung, Taiwan, R.O.C
mhjing@isu.edu.tw

Abstract-*The Viterbi algorithm is a method based on the maximum-likelihood principle to decode convolutional codes. This decoder is considered as an optimal method. In this decoder, there are two ways to enhance the error-correct capability, such as the use of soft-decision and the increasing of constraint length. However, both ways will also increase the system complexity. It has found that an Add-Compare-Select Unit (ACSU) in the decoder is the most complex module using soft-decision. The increasing rate in area of an ACSU module is also proportional to the constraint length. Therefore, the area reduction of ACSU is the most effective strategy to reducing the size of a Viterbi decoder. This system is aiming to reduce the size of the critical module using soft-decision with constraint length, Q=8. In this paper, a new architecture that is free from the absolute value comparison is proposed to reduce the hardware complexity from original ACSUs. So, the proposed ACSU can effectively reduce the hardware and the cost of a Viterbi decoder with larger constraint length.*

Keywords: Convolutional code, Viterbi decoder, Add-Compare-Select Unit, VLSI, Soft-decision.

1. Introduction

In modern communication systems, the convolutional code is widely used in wireless applications, such as cellular phone and satellite communication. Those decoders are usually implemented by a Viterbi algorithm. This Viterbi algorithm is proposed by [1] and then is considered as a maximum-likelihood decoding algorithm by Forney [2-3]. Since, the quality of communication has increasing requirement on noise immunization; the ways to enhance the error-correct capability of the convolutional code has two, such as increasing the constraint length and using soft-decision in decoder. However, both of these two ways will drive this Viterbi decoder even more complex.

The architecture of a Viterbi decoder has three modules, such as Branch Metric Unit (BMU), Add-Compare-Select Unit (ACSU), and Survivor Memory Unit (SMU). Among them, the ACSU is the most complex than other two modules and the increasing rate of the area of ACSU is proportioned to the constraint length. For instance, 8 sets of ACSUs are needed for the application of GSM system with the constraint length 5, and 128 ACSUs are needed for CDMA system with constraint length 9. Moreover, the data-width of ACSU is increasing by using soft-decision. Therefore, reducing the area of ACSU is the most effective strategy to design a Viterbi decoder to be less complex. In this paper, a new architecture of a modified Maged ACSU is proposed and it has further reduction from the designs.

The Viterbi algorithm is introduced in Section 2. In Section 3, the path metric normalization and the proposed ACSU are described. The comparison result is presented in Section 4. Finally, the conclusion is given in Section 5.

2. The Viterbi decoder

2.1. Background

The Viterbi algorithm based on a maximum-likelihood decoding method for convolutional code is used to find out the one having the minimum hamming distance in the decoding path. A decoding example of Viterbi algorithm is shown in Figure 1 which is a (2, 1), K=3 convolutional code. This convolutional encoder has four states in trellis diagram and the generator sequences are $g^{(0)} = (1,0,1)$ and $g^{(1)} = (1,1,1)$. In Figure 1, each branch has a label in the form of $x / y^{(0)} y^{(1)}$, where x is the input of state transition and $y^{(0)} y^{(1)}$ are the corresponding pair of output. If here is no noise in this example, the hamming distance of survivor path is zero.

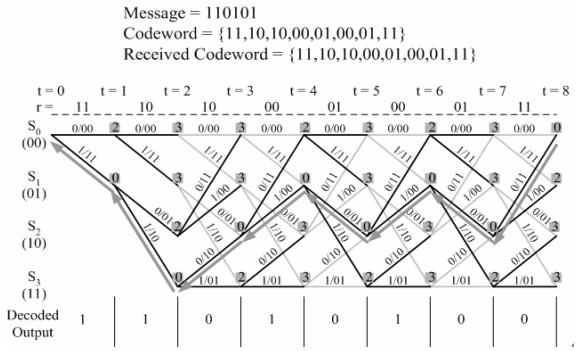

Figure 1. The example of a Viterbi algorithm

Based on the trellis diagram, the differences of all branch metrics with each received input are calculated firstly. The path metrics are accumulatively summed by such differences from all branch metrics to get the most possible path. The path with minimum summation means that this one has maximum-likelihood to the original or source state. The path link with such lowest difference is selected as the survivor path. The survivor paths are then updated step by step. Finally, the information is traced out along the survivor path in backwards. Therefore, the Viterbi decoder can be divided into three main processing units as shown in Figure 2, such as the BMU, ACSU, and SMU.

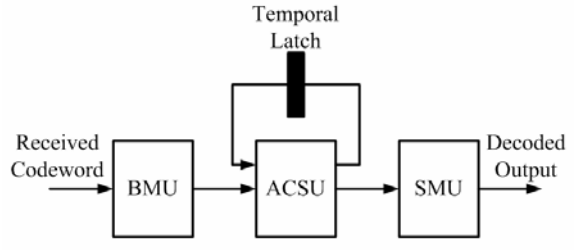

Figure 2. The architecture of Viterbi decoder

2.2. Path metric normalization

In soft-decision, the path metrics are accumulated by adding their differences. Therefore, the path metrics increase progressively during decoding process. The size of the path metrics is proportional to the decoding length of the Viterbi decoder. To solve the large size of the path metrics, the method of normalization is used [5]. The Fixed shift is used to prevent the values of path metrics exceed a maximum value so that the fixed bit-width of branch metric can be used for any decoding length and the size of the computing module is also reduced.

2.3. The Conventional ACSU

The architecture of conventional ACSU is using the butterfly diagram as shown in Figure 3. The algorithm of conventional ACSU is,

$$pm_{t+1}^{p} = \begin{cases} pm_t^i + bm_t^{(i,p)} & when \quad c_1 = 0 \\ pm_t^j + bm_t^{(i,q)} & when \quad c_1 = 1 \end{cases}, \qquad (1)$$

$$where \quad c_1 : pm_t^i + bm_t^{(i,p)} > pm_t^j + bm_t^{(i,q)}$$

$$pm_{t+1}^{q} = \begin{cases} pm_t^i + bm_t^{(i,q)} & when \quad c_2 = 0 \\ pm_t^j + bm_t^{(i,p)} & when \quad c_2 = 1 \end{cases}, \qquad (2)$$

$$where \quad c_2 : pm_t^i + bm_t^{(i,q)} > pm_t^j + bm_t^{(i,p)}.$$

In here, the c_1 and c_2 are Boolean value generated by checking the conditions of state transition

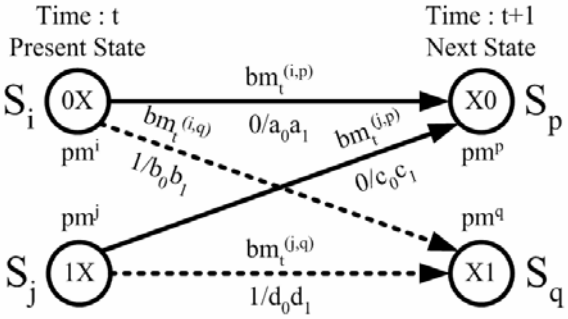

Figure 3. General butterfly diagram of (2, 1) convolutional code

According to [6], the parameters includes n, K, and Q, where n is the number of output bits of each set of code from the encoder, K is the constraint length of the convolutional code, and Q is the parameter of demodulating signal. When $Q = 2$, the Viterbi decoder has to decode by using hard-decision. The others, such as $Q = 4$ or 8, are the type of soft-decision. Some important parameters are stated here.

The maximum branch metric is

$$\lambda_{\max} = n(Q-1). \qquad (3)$$

The bit-width of branch metric is

$$bm_{bits} = \log_2(Q) + 1. \qquad (4)$$

The maximum dynamic range of path metric is

$$\Delta_{\max} = \lambda_{\max}(K-1). \qquad (5)$$

The bit-width of path metric is

$$pm_{bits} = \lceil \log_2(\Delta_{\max} + \lambda_{\max}) \rceil. \qquad (6)$$

For instance, a convolutional code wth (2, 1), $K = 3$, $Q = 8$ has the parameters as shown in Table 1. According to those descriptions, the bit-width of path metric and branch metric is 6 and 4, respectively. The $Q = 8$ means that Viterbi decoder uses soft-decision and the architecture of the conventional ACSU is shown in Figure 4. The d_{t+1}^{p}

and d_{t+1}^{q} are the results of comparison which are stored in SMU. In Figure 4, four 6-4 bits (6 bits and 4 bits) adders and two 6-bits (6 bits and 6 bits) comparators are needed.

Table 1. The parameters of (2, 1), $K=3$, $Q=8$ convolutional code

Maximum branch metric	14
Bit-width of branch metric	4
Maximum dynamic range of path metric	28
Bit-width of path metric	6

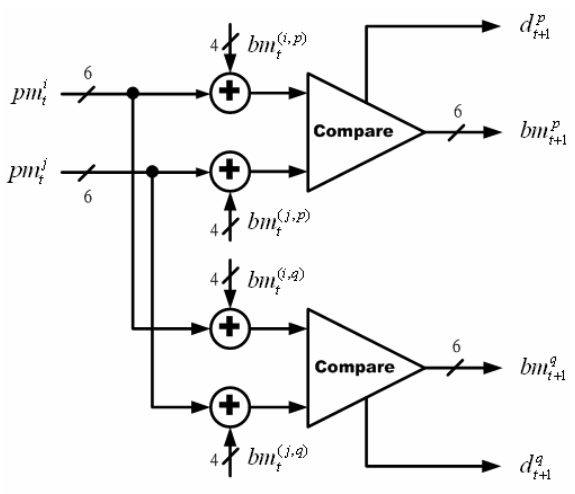

Figure 4. The conventional ACSU for (2, 1), $K=3$, $Q=8$ convolutional code

3. The Design of ACSUs

The ACSU can be implemented by using three different methods, such as conventional, rearranged, and Maged's ACSU [4]. Initially, the conventional ACSU is implemented by a butterfly diagram as shown in Figure 4. Using this diagram, it compares the summations of path metrics to select the survived path with lowest value. The advantage of this method is easy to implement, but the disadvantage is the need of more components. The rearranged ACSU is using a modified butterfly diagram. Their comparisons are on the differences of metrics to select the survived path with smaller number of adders. The Maged's ACSU uses just one absolution value comparator to compare the differences so that it uses less components than previous ones.

3.1 The rearranged ACSU

From Eq (1) and (2), $bm_t^{(i,p)}$ is moved from left hand side to the right side and pm_t^j is moved in reverse direction. We get,

$$pm_{t+1}^{p}=\begin{cases} pm_t^i+bm_t^{(i,p)} & when\ \ c_1=0 \\ pm_t^j+bm_t^{(i,q)} & when\ \ c_1=1 \end{cases}, \quad (7)$$
$$where\ \ c_1: pm_t^i-pm_t^j>bm_t^{(i,q)}-bm_t^{(i,p)}.$$

With the same movement for $bm_t^{(i,q)}$ and pm_t^j, we get,

$$pm_{t+1}^{q}=\begin{cases} pm_t^i+bm_t^{(i,q)} & when\ \ c_2=0 \\ pm_t^j+bm_t^{(i,p)} & when\ \ c_2=1 \end{cases}, \quad (8)$$
$$where\ \ c_2: pm_t^i-pm_t^j>bm_t^{(i,p)}-bm_t^{(i,q)}.$$

According to Eq. (7) and (8), the path metrics are firstly subtracted, and then compared by the Δbm_t^{q-p} and Δbm_t^{p-q} in the branch metrics supplied by the BMU. The associate path metric and branch metric are chosen by the compared result, and the new path metric is generated by adding the chosen path metric and branch metric. This architecture is called rearranged ACSU and shown as Figure 5.

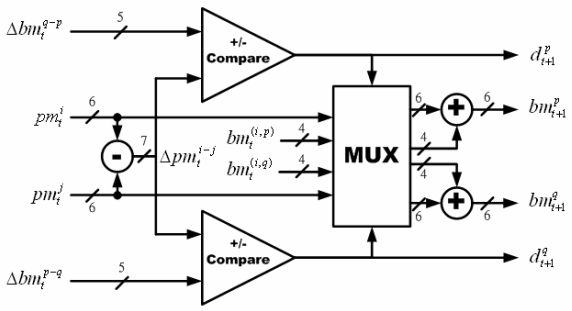

Figure 5. The rearranged ACSU for decoding (2, 1), $K=3$, $Q=8$ convolutional code

3.2 The Maged's ACSU

The algorithm of Maged's ACSU is,

$$pm_{t+1}^{p}=\begin{cases} pm_t^j+bm_t^{(i,q)} & when\ \ c=0,\ \Delta bm_t^{q-p}\ \ is\ \ negative \\ pm_t^i+bm_t^{(i,p)} & when\ \ c=0,\ \Delta bm_t^{q-p}\ \ is\ \ positive \\ pm_t^i+bm_t^{(i,p)} & when\ \ c=1,\ \Delta pm_t^{i-j}\ \ is\ \ negative \\ pm_t^j+bm_t^{(i,q)} & when\ \ c=1,\ \Delta pm_t^{i-j}\ \ is\ \ positive \end{cases} \quad (9)$$

$$pm_{t+1}^{q}=\begin{cases} pm_t^i+bm_t^{(i,p)} & when\ \ c=0,\ \Delta bm_t^{q-p}\ \ is\ \ negative \\ pm_t^j+bm_t^{(i,q)} & when\ \ c=0,\ \Delta bm_t^{q-p}\ \ is\ \ positive \\ pm_t^i+bm_t^{(i,p)} & when\ \ c=1,\ \Delta pm_t^{i-j}\ \ is\ \ negative \\ pm_t^j+bm_t^{(i,q)} & when\ \ c=1,\ \Delta pm_t^{i-j}\ \ is\ \ positive, \end{cases} \quad (10)$$
$$where\ \ c\ \ :\ |\Delta pm_t^{i-j}|>|\Delta bm_t^{q-p}|.$$

According to this algorithm, the path metrics are firstly subtracted, and then the comparator of absolute value can be used to simplify the architecture of previous one as shown in Figure 6.

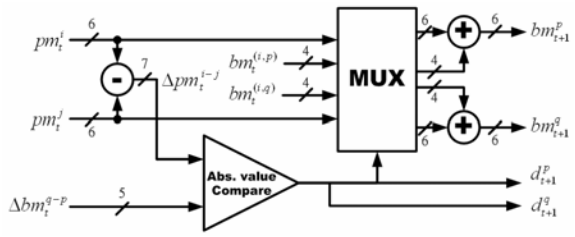

Figure 6. The Maged's ACSU for decoding $(2,\ 1),\ K=3,\ Q=8$ **convolutional code**

In Figure 6, the value of Δbm_t^{q-p} is transformed from 2's complement to sign-magnitude by a Multiplexer in BMU. The Δbm_t^{q-p} and Δbm_t^{p-q} have the same magnitude, but with different signs ($\Delta bm_t^{q-p} = -\Delta bm_t^{p-q}$). The Maged's ACSU needs two 6-4 bits (6 bits and 4 bits) adders, one 6-bits (6 bits and 6 bits) subtracter, and one 7-5 bits (7 bits and 5 bits) absolute-value comparator. The choice of path metrics and branch metrics are summarized on Table 2.

Table 2. The choice of path metric and branch metric for Maged's ACSU

$\lvert \Delta pm_t^{i-j} \rvert > \lvert \Delta bm_t^{p-q} \rvert$	Δpm_t^{i-j} positive	Δbm_t^{q-p} positive	pm_{t+1}^{p}	pm_{t+1}^{q}
0	0	0	$pm_t^j + bm_t^{(i,q)}$	$pm_t^i + bm_t^{(i,q)}$
0	0	1	$pm_t^i + bm_t^{(i,p)}$	$pm_t^j + bm_t^{(i,p)}$
0	1	0	$pm_t^j + bm_t^{(i,q)}$	$pm_t^i + bm_t^{(i,q)}$
0	1	1	$pm_t^i + bm_t^{(i,p)}$	$pm_t^j + bm_t^{(i,p)}$
1	0	0	$pm_t^i + bm_t^{(i,p)}$	$pm_t^i + bm_t^{(i,q)}$
1	0	1	$pm_t^i + bm_t^{(i,p)}$	$pm_t^i + bm_t^{(i,q)}$
1	1	0	$pm_t^j + bm_t^{(i,q)}$	$pm_t^j + bm_t^{(i,p)}$
1	1	1	$pm_t^j + bm_t^{(i,q)}$	$pm_t^j + bm_t^{(i,p)}$

3.2 The proposed ACSU

In Maged's ACSU, the difference of path metric may be positive or negative. Therefore, a comparator for absolute value is needed to check $\lvert \Delta pm_t^{i-j} \rvert > \lvert \Delta bm_t^{q-p} \rvert$ so that an absolute value comparator is needed. On the other hand, if the format of Δpm_t^{i-j} and Δbm_t^{q-p} is sign-magnitude, the absolute value comparator is not necessary. When the format of Δpm_t^{i-j} and Δbm_t^{q-p} are sign-magnitude, it is only to compare the magnitude part of Δpm_t^{i-j} and Δbm_t^{q-p}, the result of $\lvert \Delta pm_t^{i-j} \rvert > \lvert \Delta bm_t^{q-p} \rvert$ can be easily generated without using the absolute value comparator listed in Table 3. Referring to Maged's ACSU, this proposed ACSU uses different component and method to implement the ACSUs. In order to transform the format of difference of path metrics from 2's complement to sign-magnitude, the path metrics are firstly compared, and then exchanged if necessary. Because of the value of a bigger path metric subtracts smaller one, the result of subtraction is always positive. The magnitude of difference of path metric and branch metric are compared by a simple comparator. The associate path metric and branch metric are chosen by the results of compare and sign of difference of path metric and branch metric, and the new path metric is generated by the summation of the chosen path metric and branch metric. The architecture of proposed ACSU with soft-decision is then shown in Figure 7.

Table 3. The operations for checks of $\lvert \Delta pm_t^{i-j} \rvert > \lvert \Delta bm_t^{q-p} \rvert$ **using sing- magnitude form.**

Δpm_t^{i-j} 's sign	Δbm_t^{q-p} 's sign	Checking $\lvert \Delta pm_t^{i-j} \rvert > \lvert \Delta bm_t^{q-p} \rvert$ operation
Positive	Positive	Direct compare the magnitude of Δpm_t^{i-j} and Δbm_t^{q-p}
Negative	Negative	
Positive	Negative	
Negative	Positive	

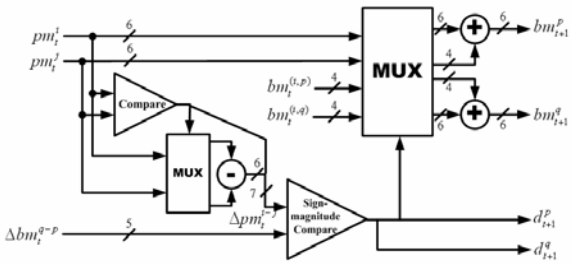

Figure 7. The proposed ACSU for decoding $(2,\ 1),\ K=3,\ Q=8$ **convolutional code**

4. The Implementation and Result

As mentioned in section 3, the complexities of ACSU are affected by the bit-width of path metric and branch metric. According to Eq. (7), Eq. (8), Eq. (9), and Eq. (10), the data-width of branch metric is affected by Q, and the data-width of path metric is affected by n, K, and Q. The number of ACSU is affected by K. The synthesis results of different ACSUs and the data-width of bm and pm in difference K and same Q are listed in Table 4. In this

proposed design, the data-width of all parameters is reduced to a fixed number by a shift operation, firstly. In this system, only two simple comparators are applied so that the complexity of ACSU has been reduced. In the implementation, this design uses VHDL by using design tool Synopsys with the 0.18-*um* UMC standard cell library. The complexity of the proposed ACSU is presented in Table 4.

5. The Conclusion

According to the result on Table 4, the proposed ACSU is the smallest one. This proposed ACSU has average reduction for 10% of original ACSU as shown in Figure 8. Therefore, the proposed ACSU can effectively reduce the complexity and also decrease the cost. As a result, the designer has more freedom to design a Viterbi decoder for a highly reliable communication using a larger constraint length with soft-decision.

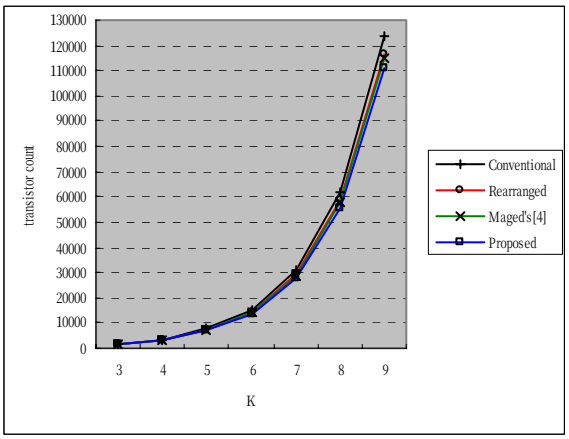

Figure 8. The complexity of ACSUs in (2, 1), Q=8 convolutional code

References

[1] Viterbi, A.J., "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, IT-13, pp.260-269, April, 1967.

[2] Forney, G. D., "The Viterbi Algorithm," *Proc. IEEE*, Vol. 61, pp.268-278, March, 1973.

[3] Forney, G. D., "Convolutional Codes II: Maximum Likelihood Decoding," *Information and Control*, Vol. 25, pp.222-266, July, 1974.

[4] Ghoneima, M., Sharaf, K., Ragai, H.F., El-Halim Zekry, "Low power units for the Viterbi decoder," *the 43rd IEEE Midwest Symposium on Circuits and Systems*, Vol. 1 pp.412-415, August, 2000

[5] Shung, C.B., Siegel, P.H., Ungerboeck, G., Thapar, H.K., "VLSI architectures for metric normalization in the Viterbi algorithm," *IEEE International Conference on Communications*, Vol.4, pp.1723-1728, April, 1990

[6] Black, P.J., Meng, T.H., "A 140-Mbs, 32-state, radix-4 Viterbi decoder," *IEEE Journal of Solid-State Circuits, Issue: 12*, Vol. 27, pp.1877–1885, Dec. 1992

Table 4 The synthesis result of different ACSUs for Q = 8 in (2, 1) convolutional code

Q	K	*bm*	*pm*	Number of ACSU	Conventional	Rearranged	Maged's[4]	Proposed
		(bit-width)			(transistor count)			
8	3	4	6	2	1710	1598	1600	1558
	4	4	6	4	3420	3196	3199	3116
	5	4	7	8	7712	7304	7195	6913
	6	4	7	16	15424	14608	14389	13825
	7	4	7	32	30848	29216	28781	27650
	8	4	7	64	61696	58432	57559	55299
	9	4	7	128	123392	116864	115119	110598