An improved low power Viterbi decoder in system-on-chips

Rong Gao, Li-Xing Tang and Shan Cao†
School of Information and Electronics, Beijing Institute of Technology, Beijing, Haidian District, China
E-mail: caoshan@bit.edu.cn
www.bit.edu.cn

The (2,1,7) convolutional codes have become the standard encoding method of satellite communication system. Based on an optimized structure, a low power (2,1,7) Viterbi decoder with trace-back length of 32 is presented in this paper. New structure of add-compare-select(ACS) unit is exploited to reduce power consumption. Instead of using register-exchange(RE) and register banks, trace-back that consumes less power and the power-saving RAM are used. Our proposed Viterbi decoder consumes about 28.8k gates using Global Foundry 65 nm technology. The power consumption of our design is about 8.8mw at 250MHz.

Keywords: Viterbi decoder; add-compare-select(ACS); trace-back; low power.

1. Introduction

The importance of communication systems has greatly increased in the past few decades. However during the transmission in the channel, signal would be inevitably disturbed and distortions which lead to intolerable errors would occur. To eliminate the errors resulting from data transmission, error control coding methods like convolutional codes which can make full use of the relativity among all the bits has been widely used in modern communication systems. Viterbi decoding algorithm is considered to be the optimal decoding method of convolution codes. High-speed Viterbi decoder is despairingly needed in a SoC chip which needs to process data transmitted from satellites. However the implementation of high-speed Viterbi decoder needs large hardware overhead and power consumption. In order to meet the PPA (performance-power-area) requirements of the System-on-Chip (SoC), a performance and power trade-off is required for the design of Viterbi decoder.

There are several solutions for the implementation of low-power and high-speed Viterbi decoder. M-algorithm proposed by [1] and T-algorithm proposed by [2] both focus on reducing the hardware resource consumption during ACS process. Trace-back(TB) [3] and register-exchange(RE) [4] have also been proposed to achieve survivor path management and decoding output. In general,

T-algorithm is more commonly used in actual applications than M-algorithm which needs to conduct an additional sorting process. However T-algorithm spends a lot of hardware resources on searching for the optimal path metric(PM). Such measures increase settling time of ACS critical path heavily. TB method consumes less power and hardware resources than RE while RE holds much less latency. And in most recent works, using power-saving RAM instead of register banks can help reducing power consumption.

Considering the fact that power consumption is our main concern, an improved Viterbi decoder is employed in this paper. We modify some details of T-algorithm, new measures are taken during the implementation of ACS unit. Based on our proposed structure of the ACS unit, effective power reduction has been achieved without increasing hardware complexity. SMU(survivor memory unit) uses RAM instead of register banks to save power. And unlike commonly used k-pointer trace-back algorithm [5], fewer RAMs are needed in our design. According to simulation and synthesis results, not only can our design meet the demand of low power, but also has advantages on hardware implementation area and data throughput rate.

The rest of this paper is organized as follows. First, section II gives a brief review of Viterbi decoding algorithm. Section III presents the details of our proposed Viterbi decoder. Performance evaluation and synthesis results are reported in section IV. Conclusion is given in section V.

2. Preliminaries

2.1. *Viterbi decoding algorithm*

Viterbi decoding algorithm is a maximum likelihood decoding algorithm for convolutional codes. The process of decoding is usually presented by trellis diagram. A (2, 1, 3) convolutional codes, the received data stream of which is (11, 01, 01, 10, 01), is used as an example in Fig. 1. Here, the branch metric is marked in the circle. The arrow from the initial state ‘00’ with 1/11/2 means the former state is ‘00’, i.t., the current input data is ‘1’, the decoding output is ‘11’, the next state is ‘10’ and the branch metric is 2. The initial state of the decoding process is set to ‘00’ . Then, calculate the branch metric which is the accumulated hamming distance of received symbol and decoding outcome for each input data. When the decoding depth is reached, a trace-back procedure is conducted and this procedure needs to return to the original state from the last state with the minimum branch metric. The bold lines with reversed arrow in Fig. 1 represent the resulting path.

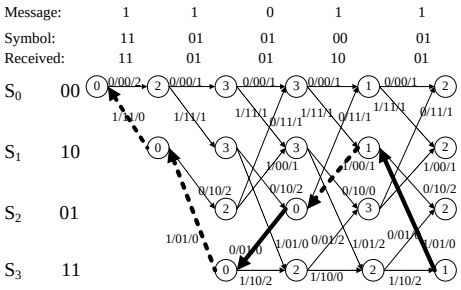

Fig. 1. Trellis diagram for (2, 1, 3) convolutional code

2.2. *The implementation of a typical Viterbi decoder*

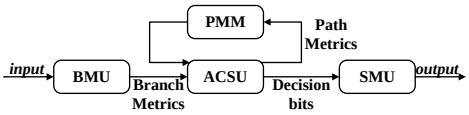

Fig. 2 A typical structure of Viterbi decoder

To complete Viterbi decoding operations, a typical structure of Viterbi decoder is shown in fig. 2. It consists of four modules: BMU(branch metric unit), ACSU, PMM(path metric memory) and SMU

The BMU is responsible for calculating the branch metric(BM) of each stage. The BM is the distance between the received symbol and the expected symbol. There are two decision methods in the Viterbi decoder, hard-decision and soft-decision. Soft decision holds better fault tolerance capability and higher hardware complexity.

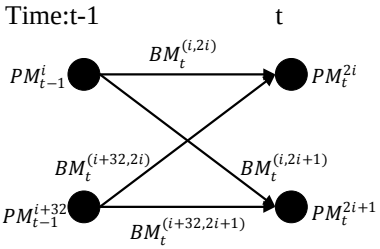

Fig. 3 Papilionaceous arithmetic for (2, 1, 7) convolutional code

ACSU which is the kernel unit of the Viterbi decoder adds the BM to the PM, compares the PM and selects the survivor path for each state. Butterfly unit is commonly used in the design of ACSU. Fig. 3 is the papilionaceous arithmetic for convolutional code. Every state has two paths pointing to next stage. There are also two paths entering every state in next stage. Path metric of states is calculated as follows:

$$PM_t^{2i} = \min(PM_{t-1}^{i} + BM_t^{(i,2i)}, PM_{t-1}^{i+32} + BM_t^{(i+32,2i)})$$

$$PM_t^{2i+1} = \min(PM_{t-1}^{i} + BM_t^{(i,2i+1)}, PM_{t-1}^{i+32} + BM_t^{(i+32,2i+1)}) \quad (1)$$

The BM values are found nonnegative, so PMs continue to increase by accumulating the BMs. Then overflow error has the potential to happen because PM can only be implemented by limited length registers in real hardware circuits. Therefore, normalization methods must be considered. According to [6], there are several methods such as reset metric normalization, variable shit metric normalization and fixed shift metric normalization. All these methods have its own advantages and disadvantages.

There are two kinds of structure for ACS: serial and full parallel. For Serial ACS, PMM(path metric memory) is needed to store path metrics so that ACS can calculate the PM one by one. For full parallel ACS, all PM can be calculated immediately, PMM is not needed and even has disadvantage on decoding speed.

SMU stores the decision bits and generates the decoded outputs. Theoretically, only when the transmission vector has been received completely, the Viterbi decoder gets started to decode and find the maximum likelihood path. After the maximum likelihood path has been found, decoding results are able to output. For huge amounts of data during the actual communications, the receive sequence is very long and huge storage space are needed in the decoder. Truncation decoding algorithm is used to solve such problem.

3. The Proposed Low-power Viterbi Decoder

The proposed Viterbi decoder consists of three modules: BMU, ACSU and SMU. PMM is not needed since full parallel ACSU is used. ACSU contains 64 parallel modified arithmetic units. Each unit handles a corresponding state, supports to delete path and can stop calculation when a path is deleted. Besides a new method of trace-back is applied in this paper. The implementation of each module is described as follows.

3.1. *BMU*

6--level quantitative soft decision is employed in this paper, '0' in the received symbols is expanded to '000' and '1' expanded to '111', which represents the most plausible transmission. Then the range of branch metric expands to 0~14. To reduce the calculation complexity, Manhattan distance instead of Euclidean distance is used. Assuming the received vector $r = \{r_0, r_1\}$, BM is calculated as follows:

$$\begin{aligned} BM_{00} &= |000\ xor\ r_0| + |000\ xor\ r_1| \\ BM_{01} &= |000\ xor\ r_0| + |111\ xor\ r_1| \\ BM_{10} &= |111\ xor\ r_0| + |000\ xor\ r_1| \\ BM_{11} &= |111\ xor\ r_0| + |111\ xor\ r_1| \end{aligned} \quad (2)$$

BM_{00}, BM_{01}, BM_{10}, BM_{11} represent the branch metric whose expected symbol is ‘00’ , ‘01’, ‘10’ and ‘11’ respectively. For (2, 1, 7) convolutional code, all states have only 4 expected output symbols as ‘00’, ‘01’, ‘10’ and ‘11’. So only 4 branch metric calculators are needed.

3.2. *ACSU*

Full parallel 64 basic arithmetic units are employed in this paper. Each unit handles a corresponding state, so that power can be saved and the data throughput rate is increased. Combined with the existing normalization methods, a new method is proposed aiming at complete the normalization with power reduction. First of all, the bit width of PM is associated with the hardware complexity of ACS. According to modified T-algorithm proposed in [7], if the minimum PM is subtracted at each stage, PM width can be limited to 5 bits, and the subtracted PM which is larger than 17 for (2,1,7) convolutional code can be deleted. To save the hardware resource spent on searching for the minimum PM, searching operation can be replaced with subtracting a constant. 15 is used as the constant in this paper and PM width is expanded to 6 bits to avoid overflow error. Then we can delete paths whose PM is larger than 17 after the subtraction. After the comparison with the threshold, a flag signal is generated to indicate whether the PM is deleted. The basic structure of normalization block is shown in Fig.5. Buffers are inserted in the normalization block to avoid glitch.

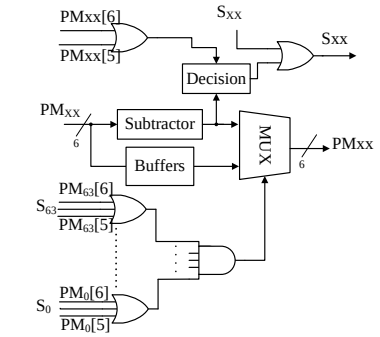

Fig.4 diagram of normalization block

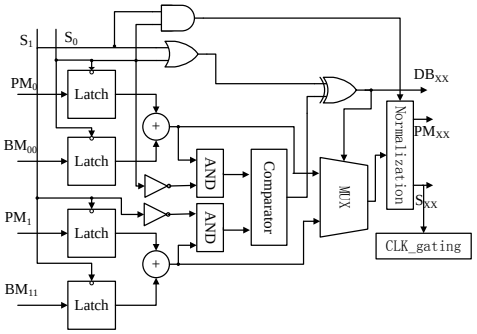

Fig.5 block diagram of basic arithmetic unit

After a path is deleted, corresponding logic in ACS of the modified T-algorithm is still active. To reduce the switching activity of ACSU, the flag signal generated in normalization block indicating whether the path is deleted can be useful. A new structure of basic arithmetic unit in ACSU is shown in fig.6. S_x is the flag signal generated in normalization block and DB_{xx} is the decision bit indicating which path is selected. We cannot use S signal to gate the basic arithmetic unit if one branch path has been deleted while the other is active. So S signal is used with a latch to maintain inputs to the adder and with an AND

cell to mask inputs to the comparator. Also S signal is used to gate registers between current state and next state. Thus after one path is deleted, the switching activity of its corresponding logic cells in ACSU is reduced effectively. To avoid timing violation, S signal must be the earliest input to latch.

Hardware complexity increased by latches and AND cells in ACSU is offset by the reduction of bit width. The hardware complexity hasn't been increased effectively.

3.3. *SMU*

A modified trace-back method is applied in this paper. For truncated decoding algorithm, truncated length is called decoding depth δ. When δ is big enough such as δ=(4~5)K (K represents the constraint length), such truncation has little disadvantage on decoding performance. However, the decoding latency from input to output and storage space increase with δ. So, δ=32 is employed in this paper. The block diagram of SMU is shown in Fig.6.

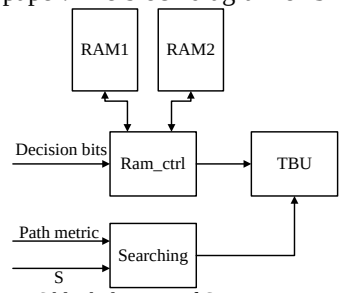

Fig.6 block diagram of SMU

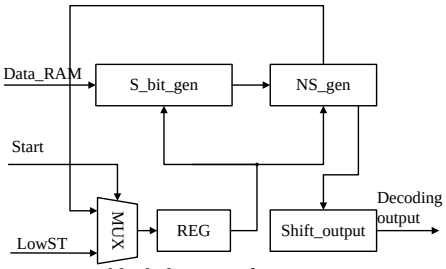

Fig. 7 block diagram of TBU

RAM control unit controls the access to RAMs. There are two RAMs in our design and both of them are 32x64 bits. At the beginning, ACSU starts running and generating decision bits. These bits are sent to SMU and wrote to RAM1. When truncation length is reached, RAM1 is full. Then RAM control unit switches write enable to RAM2 and keeps writing decision bits. Also control unit activates read enable of RAM1, and decision bits start being read to trace-back unit(TBU). At the same time, search unit searches for the minimum PM and sends it to the TBU. To finish the trace back and output process, TBU needs 32 clock cycles. When RAM2 is full, control unit starts to write RAM1 and read RAM2. Then the whole system is on a pipeline running state.

TBU is a key part of SMU, the main function of which is to trace back decision bits read from RAM. The block diagram of TBU is shown in fig.7. When data from RAMs starts being sent to TBU, RAM control unit would set start signal to '1'. LowST which is the state with minimum PM calculated by search unit would be sent to REG as current state. After this, start signal is set to

'0' until data from another RAM arrives. At the rest of the process cycle, MUX outputs the next_state from NS_gen to the REG as current state. Then S_bit_gen can pick the decision bit of current state and send it to NS_gen. According to decision bit and current state, NS_gen generates the next state to MUX and decoding result to shift. For every 32 clock cycles, the shift_output reverses the order and outputs the decoding result.

4. Implementation Results

Using top-down method, we implement the design with VHDL. Functional simulation demonstrates the validation of proposed Viterbi decoder. After the simulation, it was synthesized using Synopsys Design Compiler and GF 65nm technology. During the synthesis, low power methods such as clock gating has been used. Power consumption could be simulated via Synopsys power compiler and VCD file produced by Synopsys VCS. The comparison with existing works which are also for (2,1,7) convolutional code is shown in Table I.

Table 1. Comparison of various Viterbi decoder

	Technology	δ	Max frequency(MHz)	Cell area(mm^2)	Gates	Power(mW)
[8]	TSMC 180nm	42	132	4.30	--	196.3 (132MHz)
[9]	TSMC 180nm	64	--	3.06	49k	68(72MHz)
[10]	TSMC 180nm	36	--	--	28.6k	19.6(100MHz)
Ours	TSMC 180 nm	32	125	2.3	28.8k	18.4mw(125MHz)
ours	GF 65nm	32	466	0.55	28.8k	8.8(250MHz)

Since the comparison result between GF 65nm technology and 180nm technology isn't very intuitive, we conduct a synthesis using 180nm technology for comparison. According to the result shown in Table 1, effective power sonsumption has been achieved without increasing area.

5. Conclusion

An improved Viterbi decoder is presented in this paper. Both the modified ACS unit and new trace-back method are employed in the proposed Viterbi decoder. The modified ACS has applied a new normalization block to delete path whose PM is big enough and generate a flag signal to latch corresponding switching activities. The trace-back method employed in this paper replace the trace-back reading operation by searching for the minimum PM. The storage space and

RAM-read operations which consumes a lot of power is therefore reduced. According to the implementation results, the proposed Viterbi decoder consumes 8.8mW when works at 250MHz when using 65nm technology. All these features demonstrate that the proposed design is more suitable for future low power applications.

References

1. C.F. Lin and J.B. Anderson, M-algorithm decoding of channel convolutional codes. *Princeton Conf. Info. Sci. Syst.* (Princeton NJ, Mar. 1986).
2. S.J. Simmons, Breadth-first trellis decoding with adaptive effort, *IEEE Trans. Commun.*, vol. 38, no. 1, pp. 3–12, Jan. 1990.
3. T.K. Truong, M.T. Shih, I. S. Reed, E. H. Satorius, A VLSI design for a trace-back Viterbi decoder, *IEEE Trans. Commun.*, vol. 40, no. 3, pp.616–624, Mar. 1992.
4. D.A. El-Dib and M.I. Elmasry, Modified register-exchange Viterbi decoder for low-power wireless communications, *IEEE Trans. On Circuits and Systems I*, vol. 51, no. 2, pp. 371–378, Feb. 2004.
5. G. Feygin, and P.G. Gulak, Archigectural tradeoffs for survivor sequence memory management in Vitebi decoders, *IEEE Trans. on Comm.*, vol. 41, no. 3, pp. 425-429, Mar. 1993.
6. Kelvin Yi-Tse Lai. A high-speed low-power pipelined Viterbi decoder: breaking the ACS-bottleneck, *Proceedings of 2010 International Conference on Green Circuits and Systems*, Shanghai, China, 2010:334 - 337.
7. M.H. Chan, W.T. Lee, M.C. Lin, L.G. Chen. IC design of an adaptive Viterbi decoder. *IEEE Trans. Consumer Electronics*, 42 (1996), pp. 52–62
8. Z. Xu, J. Ren, X. Wang, and F. Ye, Implementation of folded sliding block Viterbi decoders for MB-OFDM UWB communication system, *IEEE Int. Symp. on Circuits and Syst.*, 2007, pp. 2574–2577.
9. C. C. Lin, Y. H. Shih, H. C. Chang, C.Y. Lee, Design of a power-reduction Viterbi Decoder of WLAN Application, *IEEE Trans. Circuit and Systems I*, vol. 52, pp. 1148-1156, 2005.
10. C.J. Chen, C. Yu, M.H. Yen, P.A. Hsiung, and S.J. Chen, Design of a low power Viterbi decoder for wireless communication applications, *Proc. IEEE Int. Symp. on Consumer Electronics*, Jun. 2010, pp. 1-4