A Block-Based Parallel Decoding Architecture for Convolutional Codes

Chengyi Su, Yu Zhang, Changyong Pan, Xiaofeng Wan
State Key Laboratory on Microwave and Digital Communications
Tsinghua National Laboratory for Information Science and Technology
Department of Electronic Engineering
Tsinghua University
Beijing, China
E-mail: zhang-yu@mail.tsinghua.edu.cn

***Abstract*—This paper delivers a block-based parallel convolutional decoding architecture in which several Viterbi decoders work concurrently to decode consecutive code blocks. Each code block contains a preamble and a postamble which are duplicate data from neighbor blocks. Preamble and postamble are beneficial to the continuity and correctness of decoding output. Simulation results demonstrate that this architecture has a negligible coding-gain loss, compared with the conventional Viterbi decoder. An FPGA implementation of this architecture achieves a throughput up to 1.2Gbps.**

Keywords-Convolutional Codes; Parallel Decoding; Viterbi algorithm;FPGA

I. INTRODUCTION

Viterbi algorithm [1] is an optimal convolutional decoding algorithm from the maximal likelihood decoding point of view. However, the feedback loop in the add-compare-select (ACS) unit in the Viterbi decoder imposes the bottleneck on the decoding speed [2]. Parallel architecture has been widely studied and proposed to improve the decoder throughput. Block-based decoding approaches like state initialization and interleaving have relatively simple architectures, but their usages are limited to certain coding schemes [3]. M-step lookahead algorithm and slice-block algorithm could be applied to any coding scheme, at the price of an exponential growth in hardware area [3, 4]. The highest throughput of a (2, 1, 7) Viterbi decoder was about 1Gbps on ASIC [5], or 510Mbps on FPGA [6] so far.

The basic limitation on the parallelism of convolutional codes is its memory effect. The simplest method is to decompose the code sequence into blocks of length L_B which can be processed in parallel using N convolutional Viterbi decoders. These N decoders are named sub-decoders in the following text to avoid confusion. Such architecture multiplies the throughput and complexity both by N times. The difficulty of this method is that the initial state metrics of a block is unknown until its previous block has been processed [3]. Without the knowledge of the initial state metrics, the decoding of a block should undergo an initial synchronization stage in which the paths gradually merge into one correct path. Initial synchronization stage takes a length of 4 or 5 times of l_C, where l_C is the constraint length of the code. Besides, the decoding output in initial synchronization stage is subject to error [1].

CHINACOM 2010, August 25-27, Beijing, China

DOI 10.4108/chinacom.2010.77

A partition scheme was proposed in [7] that a segment of S branches from precedent block is appended to the beginning of a block for the initial synchronization stage. Theoretical analysis was given to prove that the error probability due to this partition scheme is negligible compared with the affect of path memory truncation. The choices of S for certain codes were verified by simulation. Nevertheless, the architecture in [7] introduces an unnecessary output delay. The area efficiency and practical performance under noise were not exploited.

In this paper a novel partition scheme is developed based on [7]. A preamble and a postamble are appended to a block to increase the continuity and correctness of decoding. The preamble of the n-th block is a sequence of duplicate branches from the valid data segment of the (n-1)th block. Preamble helps a sub-decoder to transition to a required initial state before the decoding of valid data segment starts. The postamble of the n-th block is a sequence of duplicate branches from the valid data segment of the (n+1)th block. Postamble is a guard interval between two consecutive blocks in a sub-decoder. A proper proportion of preamble and postamble in a block could decrease the area overhead and bit error rate (BER) to the minimum level. Simulation of this architecture shows that the coding-gain loss by this parallel architecture is almost negligible. Punctured codes are also taken into consideration. This architecture is implemented on an FPGA chip as part of a concatenated decoder based on CCSDS recommendation [8].

The organization of this paper is as follows. Section II covers the architecture of the whole decoder. Section III depicts the format of a block. In Section IV the theoretical analysis of this partition method is discussed. Simulation and implementation results are provided in Section V and VI, respectively.

II. DECODER ARCHITECTURE

Fig. 1 depicts the proposed parallel architecture. It consists of a data splitter, N conventional convolutional sub-decoders and a data combiner. Data splitter decomposes the codes sequence into blocks with equivalent size and distributes these

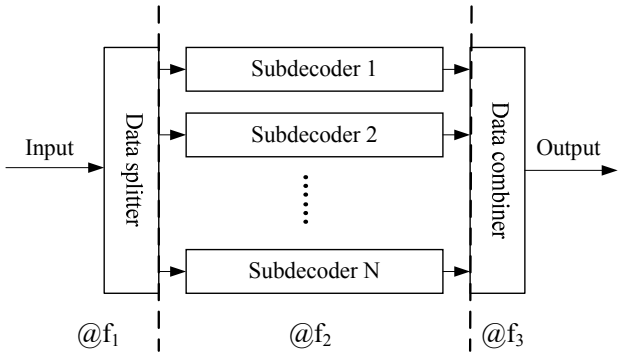

Figure 1. The proposed parallel architecture

blocks to the N parallel sub-decoders in order. Sub-decoders work concurrently and deliver the decoding output into data combiner. Data combiner assembles the decoding output block by block to retrieve a continuous sequence. To facilitate clock management and maintain an acceptable timing condition, the whole system is partitioned into three clock domains. Parallel-to-serial conversion and serial-to-parallel conversion are performed to the input and output of the sub-decoders to slow down the frequency of the first time domain and the third time domain. Asynchronous FIFOs are utilized as buffers for clock domain crossing.

A. Data Splitter

Data Splitter works at the first clock domain. The frequency of this clock domain, f_1, depends on the throughput of the system, the width of input data in symbol and the code rate. Code sequences are disassembled into blocks here, following the format described in Section III. Symbol synchronization, branch synchronization or puncture pattern synchronization should be processed here before the partition of data. Sub-decoder feedback is provided for the try-and-error search to dissolve these synchronization issues.

B. Sub-decoders

The sub-decoders instantiated in this architecture are conventional Viterbi decoders [9] operating at frequency f_2. The basic code is (2, 1, 7), with punctured codes at the rate of 2/3, 3/4, 5/6, 7/8, as recommended by CCSDS [8]. The traceback depth l_T of sub-decoder could be adjusted with the punctured rate. These N sub-decoders work concurrently. A sub-decoder processing the n-th block will continue to process the (n+N)th block later. The decoding outputs of preamble and postamble are discarded. A serial-to-parallel conversion is performed at the output end of the sub-decoder to slow down the clock frequency of data combiner. Setting the output width of sub-decoder as the symbol width of an outer code is a good choice from a systematical perspective.

C. Data Combiner

Data Combiner fetches data from sub-decoders in order, one block at a time, to assemble a seamless sequence output. Parameters of block format should be selected to assure the decoding result of a block is the multiple of data bus width under any code rate.

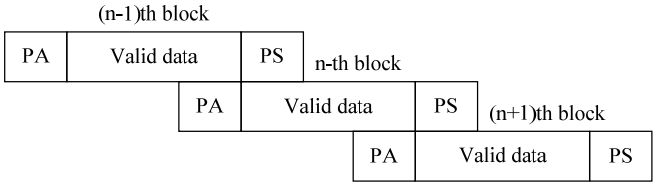

Figure 2. The proposed block format

III. BLOCK FORMAT

Fig. 2 depicts the proposed block format and the relation between adjacent blocks. Every block contains three segments, the preamble (PA), the valid data segment and the postamble (PS). The lengths of these segments in branches are l_1, l_2 and l_3, respectively. Neighbor blocks share part of data, but there is no overlap or gap between their valid data segments.

The distribution of data is illustrated in (1) - (3).

$$\begin{cases} d_{valid}(m,i,j) = d_{in}(k) \\ i = \lfloor (k-1)/l_2 \rfloor \bmod N + 1, i = 1,2,\ldots,N \\ j = (k-1) \bmod (l_2) + 1, j = 1,2,\ldots,l_2 \\ k = ((m-1)N + i - 1)l_2 + j \end{cases} \tag{1}$$

where $d_{in}(k)$ is the k-th branch in the input sequence, $d_{valid}(m, i, j)$ is the j-th branch in the data valid segment of the m-th block processed by the i-th sub-decoder. For the n-th block in Fig. 2, n=(m-1)N+i.

$$\begin{cases} d_{PA}(1,1,j) = 0, j = 1,2,\ldots,l_1 \\ d_{PA}(m,1,j) = d_{valid}(m-1,N,l_2 - l_1 + j), \\ j = 1,2,\ldots,l_1, m > 1 \\ d_{PA}(m,i,j) = d_{valid}(m,i-1,l_2 - l_1 + j), \\ i = 2,3,\ldots,N, j = 1,2,\ldots,l_1, m \geq 1 \end{cases} \tag{2}$$

where $d_{PA}(m, i, j)$ is the j-th branch in the preamble of the m-th block processed by the i-th sub-decoder. The preamble of the n-th block is the last l_1 branches of the data valid segment of the (n-1)th block.

$$\begin{cases} d_{PS}(m,i,j) = d_{valid}(m,i+1,j), \\ i = 1,2,\ldots,N-1, j = 1,2,\ldots,l_3 \\ d_{PS}(m,N,j) = d_{valid}(m+1,1,j), \\ j = 1,2,\ldots,l_3 \end{cases} \tag{3}$$

$d_{PS}(m, i, j)$ is the j-th branch in the postamble of the m-th block processed by the i-th sub-decoder. The postamble of the n-th block is the first l_3 branches of the data valid segment of the (n+1)th block.

To appreciate the benefits of a preamble, the initial synchronization of a Viterbi decoder should be reviewed. As a Viterbi decoder starts to work, its initial values of all the state metrics are set to zeros. The decoder will merge to the correct path gradually, and finally recover after a span of decoding errors when the valid path dominates. This stage will take a few constraint lengths. For punctured codes or codes under noise it should take a larger length. The affect of incorrect initial state metrics on decoder is negligible after this stage.

For a sub-decoder in the proposed architecture, the code sequence input is discontinued at the boundaries between blocks. Therefore the sub-decoder shall undergo an initial synchronization at the beginning of every block. Preamble is used as training sequence to help the sub-decoder merge to the correct path before decoding the valid data segment. The decoding outputs of preamble, which are subject to error, should be discarded. But that of valid data segments are generally reliable and reserved. The length of preamble l_1, as mentioned above, is about 4~5 times of l_C.

Postamble works as a guard interval between blocks. For a sub-decoder working in a continuous stream mode, it traces back a length of data before making a decision and delivering output. Without a postamble, the head of (n+N)th block is used for the decoding of the tail of the n-th block. This would cause errors at the end of each block. A postamble l_3 no shorter than the traceback depth l_T will prevent this kind of error. In a practical system, l_T is 4~5 times of l_C for original code, and up to 15 times of l_C for highly punctured codes [10].

While l_1 and l_3 depend on l_C and the code rate, l_2 is free to scale. As overheads, preamble and postamble decrease the area efficiency and throughput of the system. A large l_2 helps to improve efficiency and throughput. But a trade-off between efficiency and area should be considered here because a larger l_2 means longer blocks which require more buffers before and after decoding. When the ratio of $l_2/(l_1+l_3)$ is 20, the overhead is cut down to 5%. Meanwhile the total memory utilization is within an acceptable range.

IV. Theoretical Analysis

In [1, 7] initial synchronization error probability P_{IS} is bounded by

$$P_{IS}(l_1) \le \exp[-Vl_1E(R)] \tag{4}$$

where R=b/V bit /symbol is the code rate and

$$E(R) = \max_{0\le\rho\le1}[E_0(\rho) - \rho R] \tag{5}$$

is Gallager's function.

And the average path memory truncation error probability P_T is also bound by

$$P_T(l_T) \le \exp[-Vl_TE(R)] \tag{6}$$

The error probability in a valid data segment P_{valid} is

$$\begin{aligned} P_{valid}(l_1,l_2) &= \frac{1}{l_2}\sum_{i=l_1}^{l_1+l_2-1} P_{IS}(i) \\ &\le \frac{\exp[-Vl_1E(R)]}{(1-\exp[-VE(R)])l_2} \end{aligned} \tag{7}$$

For l_2=20l_1 and l_1=l_7=5l_C, P_{valid} is constrained by a bound 2 orders lower than P_T. More theoretical analysis is difficult to carry on, but it could be safe to say that with proper l_1 and l_2, the effect of initial synchronization on each block is negligible compared with that of path memory truncation. And For $l_3 \geqslant l_T$, the partition will cause no error at the end of a valid data segment.

V. Simulation Results

Computer simulation was performed to exploit the degradation of performance of the proposed architecture under different signal-to-noise ratio (SNR). The simulation used Binary phase shift keying (BPSK) in additive white Gaussian noise (AWGN) channel. Punctured codes were also simulated. Comparison of conventional decoder and the proposed architecture on the code rate of 1/2 and 3/4 are demonstrated in Fig. 3 and Fig. 4. For 1/2 code rate, l_1=l_3=l_7=6l_C, l_2=40l_1; for 3/4 code rate, l_1=l_3=l_7=8l_C, l_2=30l_1. In another word, the preamble, postamble and traceback depth are larger for 3/4 code, while the valid data segment is equivalent for two codes. Theoretical curve for 1/2 code and unquantized curve for 3/4 code are plotted for reference.

The curves of conventional decoder and the proposed architecture are very close in Fig. 3 and Fig. 4. Actually the BER of the proposed architecture is about 5% higher than that of the conventional decoder for 1/2 code. The BER difference ratio of 3/4 code ranges from 5% to 10% at different SNR. This slight coding-gain degradation is about 0.01~0.02dB in E_b/N_0.

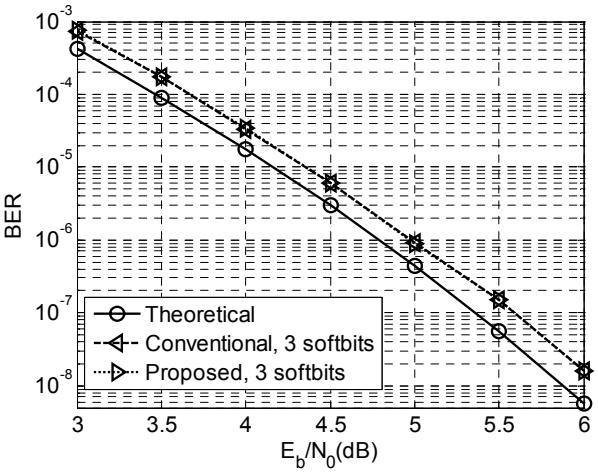

Figure 3. BER of 1/2 code

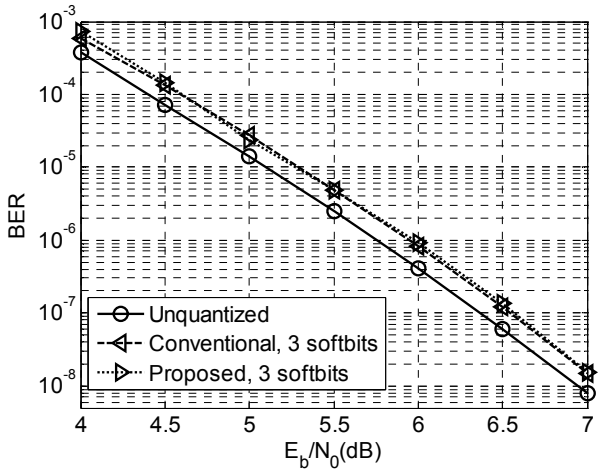

Figure 4. BER of 3/4 code

VI. IMPLEMENTATION

A high-speed parallel decoder as proposed is implemented on Altera FPGA Stratix II EP2S90F1020C3. The resource utilization is listed in Table. I. The maximum frequencies of three clock domains are list in Table. II. The number of sub-decoders N=6, with 3 soft bits and maximum constraint length l_T=15l_C, for each sub-decoder. The implementation shows that for a large $l_2/(l_1+l_3)$, this architecture is almost linear in complexity. For 1/2 code, code sequence input through a bus width of 4 branches at 300MHz. Decoding output is delivered as byte sequence at 150MHz. A throughput up to 1.2 Gbps is achieved. A throughput about 1.1Gbps is available for highly punctured codes like 7/8 codes. High rate codes require longer preamble and postamble, decreasing the area efficiency and hence throughput.

TABLE I. RESOURCE UTILIZATION

Logic Utilization	21%
Combinational ALUTs	13,976/72,768(19%)
Dedicated logic registers	7,288/72,768(10%)
Total block memory bits	425,496/4,520,448(9%)

TABLE II. MAXIMUM FREQUENCY OF CLOCK DOMAIN

Clock Domain	**F_{MAX}(MHz)**
1st: Data splitter	327.0
2nd: Sub-decoders	220.1
3rd: Data combiner	224.5

VII. CONCLUSION

In this paper a block-based parallel convolutional decoding architecture is introduced. A specific block partition format for this architecture is also developed. Compared with other parallel decoding algorithms, this architecture exceed in scalability, generality and compatibility. The complexity of this architecture is linear to the throughput. Area permitting, arbitrary folds of throughput can be achieved. It imposes no constraint on the code rates or the encoding scheme. This architecture is independent of the details of sub-decoder, providing a simple expansion method for any present algorithm. Simulation results demonstrated that the coding-gain loss caused by the initial synchronization in each block is negligible.

ACKNOWLEDGMENT

All the work in this paper is supported by Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT)

REFERENCES

[1] A. J. Viterbi and J. K. Omura, “Principles of digital communication and coding,” New York:McGraw-Hill, 1979.

[2] J. Tang and K. K. Parhi, “Viterbi decoder for high-speed ultra-wideband communication system,” ICASSP 2005, vol.5, pp.37-40, 18-23 March 2005.

[3] H. D. Lin and D. G. Messerschmitt, ”Algorithms and architectures for concurrent Viterbi decoding,” ICC’89, vol. 2, pp.836-840, 11-14 June 1989.

[4] P. J. Black and T.H. -Y. Meng, “A 1-Gb/s, four-state, sliding block Viterbi decoder,” IEEE J. Solid-State Circuits, vol. 32, pp.797-805, June 1997.

[5] Mark A. Anders, Sanu K. Mathew, Steven K. Hsu, Ram K. Krishnamurthy, Shekhar Borkar, “A 1.9 Gb/s 358 mW 16256 State Reconfigurable Viterbi Accelerator in 90 nm CMOS,” IEEE Journal of Solid State Circuits, Vol. 43, No. 1, pp214-222. January 2008.

[6] Jiuling Tang, “Design and FPGA Implementation of a Viterbi Decoder: A Case Study Using SystemVerilog and Co-Simulation,” Signal Processing and Information Technology (ISSPIT), 2009 IEEE International Symposium on, Ajman. Feb. 2009.

[7] Y.F. Zhang and P. Csillag, “Parallel architecture for high-speed Viterbi decoding of convolutional codes,” Electronics Letters, vol. 35, No.14, pp. 887–888, April 1989.

[8] Recommendation for space data system standards-TM synchronization and channel coding, CCSDS 131.0-b-1, Blue Book September 2003.

[9] J. A. Heller and I. M. Jacobs, “Viterbi decoding for satellite and space communication,” IEEE Trans. Commun. Technol., vol. COM-19, No.5, pp. 835-848, October 1971.

[10] H. A. Bustamante, I. Kang, C. Nguyen and R.E. Peile, “Stanford telecom VLSI design of a convolutional decoder,” MILCOM'89, vol.1, pp.171-178, 15-18 October 1989.