

TIME AND ENERGY EFFICIENT VITERBI DECODING USING FPGAS *

Jingzhao Ou, Viktor K. Prasanna

Department of Electrical Engineering
University of Southern California, Los Angeles, CA 90089

ABSTRACT

State-of-the-art FPGAs integrate multi-million gate configurable logic and heterogeneous hardware components. They are an attractive choice for implementing Viterbi decoders. As more emphasis is placed on time and energy performance, previous FPGA implementations of Viterbi decoders either fail to provide high data throughput or are not energy efficient. In this paper, we propose an architecture for implementing Viterbi decoders on FPGAs. Our architecture can provide various throughput and energy trade-offs. Considering the *throughput/energy* performance metric, experimental results show that our design achieves improvements up to 26.1% compared with the previous designs.

1. INTRODUCTION

Convolutional encoding with Viterbi decoding is a powerful method for forward error correction. It has been widely deployed in many wireless communication systems to improve the limited capacity of the communication channels. With the proliferation of portable and mobile devices as well as the ever increasing demands for high-speed data transmission, both time (e.g. throughput) and energy efficiency have become important performance metrics when implementing Viterbi decoders [9].

The rapid evolution of modern FPGAs have led to the inclusion of up to multi-million gate configurable logic and various precompiled hardware components (e.g. memory blocks, dedicated multipliers, etc.) on a single chip. Examples of these FPGA devices are Xilinx Spartan-3/Virtex-II/Virtex-II Pro [10] and Altera Stratix/Stratix-II [1]. These FPGA devices provide high computational performance, low power dissipation per computation, and reconfigurability. Therefore, they are an attractive choice for implementing various functions of telecommunication systems such as Viterbi decoding, adaptive beamforming, etc. [2].

Several researchers have implemented Viterbi decoders on FPGAs [6], [8] (See Section 2 for more details). However, these designs either fail to provide high throughput (requiring tens of clock cycles to generate each output symbol) or use a lot of registers and multiplexers and thus consume a lot of energy. In this paper, we propose a circular linear pipeline architecture based on the trace-back algorithm for implementing Viterbi decoders on FPGAs. The two major advantages provided by our design are: (1) *high throughput*: the tracing back, updating, and storing of the

input information sequences are accomplished concurrently within the processing elements (PEs) that constitute the circular linear pipeline. Our circular linear pipeline architecture overcomes the low throughput problem in the previous implementations of the trace-back algorithm. The degree of parallelism of our design is parametrized and is determined by the number of PEs employed; (2) *high energy efficiency*: First, by employing a trace-back algorithm, our design greatly reduces data movement compared with the register-exchange algorithm. Switching activity, a dominating factor that affects energy dissipation, is also reduced. Second, embedded memory blocks are used as the main storage. Embedded memory blocks dissipate much less power per bit data than that of slice-based memory blocks and flip-flop based registers.

The paper is organized as follows. Section 2 discusses related work. Our implementation of the Viterbi decoding algorithm is presented in Section 3. Experimental results are shown in Section 4. We conclude in Section 5.

2. RELATED WORK

As is discussed in [3], implementations of the Viterbi algorithm can be classified into two categories, register-exchange and trace-back, depending on how the information of the surviving paths is stored.

The *register-exchange* algorithm stores the surviving paths at each step of the decoding trellis directly. When the data comes in, each pair of surviving paths compete with each other in the next step of the decoding trellis. The complete surviving path is copied to the place where the discarded path is stored. Since the surviving paths are stored explicitly, when the truncation length is reached, the output symbol can be obtained immediately from the end of the surviving path with the greatest possibility of matching the input data sequence. An implementation of the register-exchange algorithm on FPGAs is proposed by Swaminathan *et. al.* [6]. They realized a suboptimal Viterbi decoding algorithm. In their design, copying of the surviving paths is realized using multiplexers and registers. The high memory bandwidth required by the concurrent copying of surviving paths prevents the use of embedded memory blocks even though embedded memory blocks are much more energy efficient than registers for storage. This is because access of the data stored at the memory blocks is restricted to two 32-pin I/O ports, which cannot provide such a high memory bandwidth. Another drawback is that all the multiplexers and registers are active during the decoding procedure, which consume a lot of power. Hence, while their de-

*THIS WORK IS SUPPORTED BY THE UNITED STATES NATIONAL SCIENCE FOUNDATION (NSF) UNDER AWARD NO. CCR-0311823.

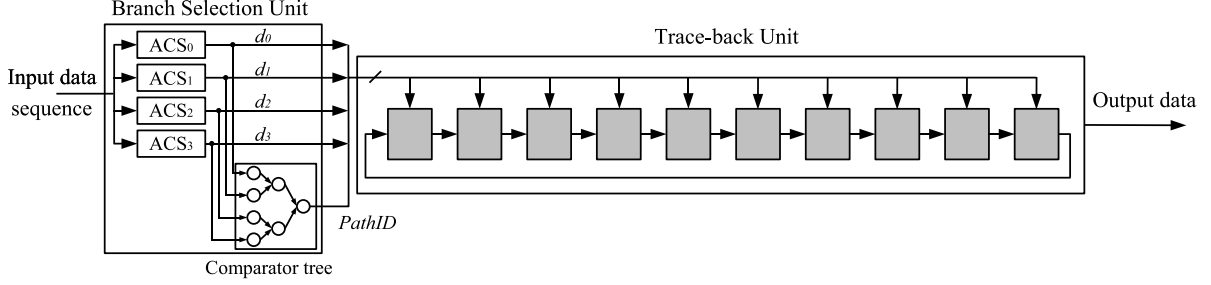


Fig. 1. Architecture of the proposed Viterbi decoder

sign provides high throughput, it is not suitable for energy-constrained systems.

In contrast, using techniques such as the one-bit function proposed in [8], the *trace-back* algorithm stores the transitions between two consecutive steps on the trellis. No path copying is required when a path is discarded. Since the path information is stored implicitly, when the truncation length is reached, a trace-back stage is required to trace back along the path transitions in order to identify the output symbol. An implementation of the trace-back algorithm can be found in a low-power architecture for Viterbi decoding proposed by Liang *et al.* [4]. Embedded memory blocks are used to improve energy efficiency. However, their design is pipelined. For environments with low SNR, a long truncation length is required and their decoder results in a very low data throughput.

Xilinx provides two implementations of Viterbi decoders on their FPGA devices [10]. The parallel implementation employs the register-exchange algorithm while the serial implementation employs the trace-back algorithm. Both versions suffer from either high energy dissipation or low throughput problem as described above.

Truong *et al.* [8] propose a fully pipelined VLSI architecture to implement Viterbi decoding using a trace-back algorithm. Their design achieves a high throughput of one output symbol per clock cycle. However, in their architecture, all the storage registers are active during the decoding process. This would result in a high memory I/O bandwidth requirement. Such requirement cannot be sustained by the embedded memory blocks on FPGAs. It would also result in high energy dissipation due to significantly high switching activity. Therefore, the design in [8] is not suitable for implementation on FPGAs.

3. OUR DESIGN

Our design of Viterbi decoder is based on the trace-back algorithm. The overall architecture of our design is shown in Figure 1. It consists of two major components: the branch selection unit and the trace-back unit. The *branch selection unit* calculates the partial costs of the paths traversing the decoding trellis, selects surviving paths, and identifies partial paths with lowest costs. The memory blocks within the *trace-back unit* store the selection results from the branch selection unit. Using the stored path selection information,

the *trace-back unit* traces back along the paths with lowest costs identified by the branch selection unit and generates output symbols. We employ a circular linear pipeline of PEs and move the data along the linear pipeline so that the trace-back operations can be performed concurrently within the PEs. Details of these two components are discussed in the following subsections.

3.1. Branch Selection Unit

Let K denote the constraint length of the convolutional code and TL denote the truncation length of Viterbi decoder. Then, there are 2^{K-1} surviving paths at each step of the decoding trellis and one output symbol is generated after tracing back TL steps on the trellis.

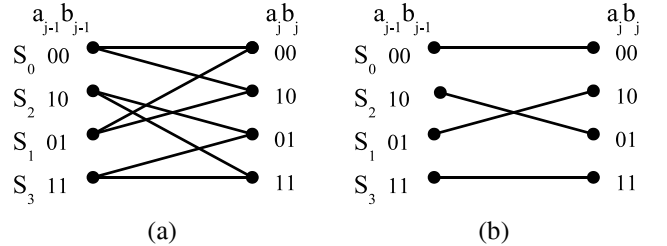


Fig. 2. (a) Possible connections between two consecutive steps on the trellis; (b) one selection path result for $(d_0, d_1, d_2, d_3) = (0, 1, 0, 1)$

Since the branch selection unit is relatively simple, we fully parallelize its operations in order to achieve a high data throughput. The branch selection unit is clock gated so that it dissipates negligible amount of energy when there is no input data. There are 2^{K-1} *Add-Compare-Select* units, ACS_k , $0 \leq k \leq 2^{K-1} - 1$. Each ACS unit is responsible for selecting one of the 2^{K-1} surviving paths and calculating its cost $pCst$. The path selection results, d_k , $0 \leq k \leq 2^{K-1} - 1$, is represented using the one-bit function proposed in [8]. For example, we consider the possible connections between two consecutive steps on the decoding trellis as shown in Figure 2(a). For each pair of paths merging into a node on the right, only one of them is selected. $d_k = 0$ if the path coming from above is selected; $d_k = 1$ if the path coming from below is selected. The selection result shown in Figure 2(b) is represented as

$(d_0, d_1, d_2, d_3) = (0, 1, 0, 1)$. Finally, the costs of the surviving paths are sent to the comparator tree where the path with the lowest cost is identified.

The comparator tree is fully-pipelined and is composed of $2^K - 1$ comparator modules. The architecture of these comparator modules is shown in Figure 3. Each module accepts the partial path IDs (pID_1 and pID_2) and the costs ($pCst_1$ and $pCst_2$) of the two input paths and outputs the partial path ID (pID_3) and cost ($pCst_3$) of the path with the lower cost.

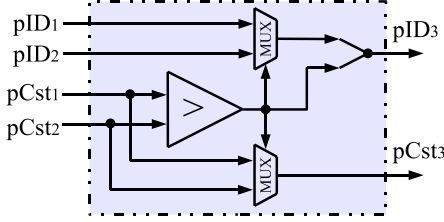


Fig. 3. The comparator module

3.2. Trace-back Unit

As shown in Figure 1, the trace-back unit consists of a circular linear pipeline of N_p processing elements (PEs), PE_i , $0 \leq i \leq N_p - 1$. N_p is a user-defined parameter and determines the throughput of the Viterbi decoder.

The architecture of a PE is shown in Figure 4. The path selection information from the branch selection unit is stored at the survivor memory. Instead of the centralized architecture adopted by the previous designs, we implement the survivor memory in a distributed manner. Each PE has its own survivor memory implemented using the embedded memory blocks (BRAMs) on Xilinx FPGAs. These BRAMs provide two I/O ports (port A and port B) that enable independent shared access to a single memory space. The data from the branch selection unit comes into the survivor memory through port A of the BRAMs and is made available to the trace-back circuits through port B. Let L denote the trace-back depth of the PEs. The memory space of each PE is divided into two parts, each of which stores the trace-back information of L consecutive steps on the decoding trellis. These two parts of memory space are used alternately to provide trace-back information to the trace-back circuit through port B and to store the data from the branch selection unit through port A.

The lower part of Figure 4 shows the trace-back circuit when $K = 3$. $y_{i,j}(a_{i,j}b_{i,j})$ are the path selection information from the survivor memory. *Regs* are registers that store $(a_{i,j}b_{i,j})$, the status of trace-back path i at step j . According to the definition of one-bit function in [8], the state of trace-back path i at step $j - 1$ can be obtained using the following equations.

$$a_{i,j-1} = b_{i,j} \quad (1)$$

$$b_{i,j-1} = y_{i,j}(a_{i,j}b_{i,j}) \quad (2)$$

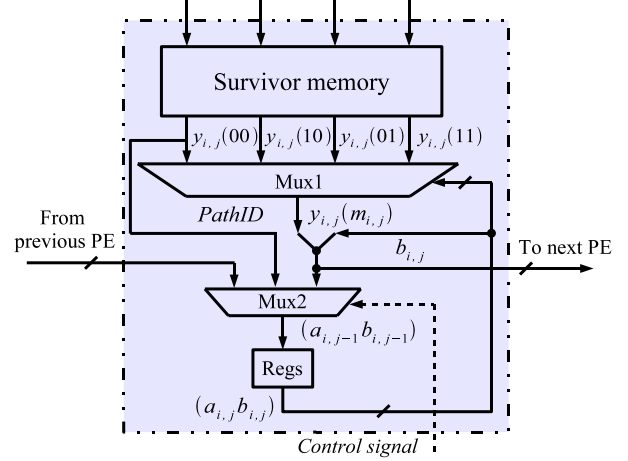


Fig. 4. PE_i in the circular linear pipeline

Equation 2 is implemented using multiplexer MUX_1 in Figure 4. The data required by the concurrent trace-back processes is stored in a distributed manner among the PEs. Multiplexer MUX_2 is responsible for moving the trace-back processes along the circular linear pipeline so that these processes can be executed in the PEs where the required trace-back data is stored.

Figure 5 illustrates the data movement in the PEs when the parameters of the decoders are set as $K = 2$, $N_p = 3$, and $L = TL = 10$. We assume that the trace-back information for steps 1 to 10 and steps 31 to 40 is stored at PE_1 before time 1. Then, during time 1 to 200, PE_1 initiates the trace-back processes that generate output symbols 22 to 31. Note that the trace-back processes may be moved to the next PEs along the circular linear pipeline in order to obtain the required data from the survivor memory of those PEs. For example, in the trace-back of $32 \rightarrow 23$, $32 \rightarrow 31$ is performed in PE_1 initially. Since the path selection information of step 30 to 23 are stored in PE_3 , multiplexer MUX_2 in Figure 4 moves the remaining trace-back process of $31 \rightarrow 23$ to be performed in PE_3 . In the mean time, the trace-back data for step 61 to 70 comes in from the branch selection unit and is stored at the place where the trace-back data for step

Time	PE1		PE2		PE3		Output
	input	trace-back	input	trace-back	input	trace-back	
1-20	(61-70)	31→22		41→32		51→42	22,32,42
21-40	↓	32→23	(71-80)	42→33		52→43	23,33,43
41-60	(1-10)	33→24	↓	43→34	(81-90)	53→44	24,34,44
...		...	(11-20)
161-180		39→30		49→40	(21-30)	59→50	30,40,50
181-200		40→31		50→41		60→51	31,41,51
201-220	(91-100)	61→52		71→62		81→72	52,62,72
241-260	↓	62→53	(101-110)	72→63		82→73	53,63,73
261-320	(31-40)	63→54	↓	73→64	(111-120)	83→74	54,64,74
...		...	(41-50)
361-380		69→60		79→70	(51-60)	89→80	60,70,80
381-400		70→61		80→71		90→81	61,71,81
...	

Fig. 5. Concurrent decoding processes within the PEs

1 to 10 is originally stored. The data in PE_2 and PE_3 is stored and updated in a similar manner as in PE_1 .

3.3. Performance Analysis

Tracing back one step on the trellis is performed within one clock cycle. Thus, each PE generates one output symbol every TL clock cycles. Since the decoding process occurs concurrently in all the N_p PEs, the throughput of our decoder is N_p/TL output symbol per clock cycle. Constrained by the branch selection unit, the maximum throughput is equal to or less than one output symbol per clock cycle, which poses limitations on N_p that $N_p \leq TL$.

4. EXPERIMENTAL RESULTS

Parameterized designs of our Viterbi decoder were described in VHDL. Synplify Pro 7.2 [7] was used for synthesis. ISE 5.2.03 [10] was used for implementation. ModelSim 5.7 [5] was used for simulation. Our target device was Xilinx Virtex-II Pro. The energy values were obtained by using XPower [10] to analyze the simulation files from ModelSim which record the switching activities of each logic and wire on the device.

We consider two optimal 1/2-rate convolutional codes in octal format: (57, 65) for $K = 6$ and (357, 233) for $K = 8$. Truncation length TL is set as $10 \cdot K$. Also, we set the operating frequency of the decoders to be 100 MHz and the trace-back length L of the PEs to be 256. The throughput and energy performance of the two Viterbi decoders configured with different numbers of PEs are shown Figure 6.

As discussed in Section 3.3, the throughput of our Viterbi decoders increases linearly with the number of PEs, N_p . The energy dissipation for decoding one bit data also increases due to the costs for sending input data to each PE and moving the trace-back processes along the linear pipeline. When $N_p = 1$, our designs lead to the serial implementations of the trace-back algorithm discussed in Section 2. Thus, designs with $N_p = 1$ are used as the baseline to illustrate the performance improvement. We consider *throughput/energy* as the performance metric. Designs with $N_p = 32$ achieve maximum performance improvements of 26.1% when $K = 6$ and 18.6% when $K = 8$ compared with the baseline designs with $N_p = 1$.

Note that placing and routing the designs become more complicated with more PEs. Designs with more than 32 PEs are unable to meet the 100 MHz timing constraint and thus are not shown in Figure 6.

5. CONCLUSION

An architecture for implementing high throughput and energy efficient Viterbi decoders on FPGAs is proposed. The effectiveness of our design is shown through the analysis of the architecture as well as low-level experimental results.

The use of multiple clock domains as in the Xilinx design of Viterbi decoder [10] can also be applied to our design to further improve throughput and energy efficiency. Besides, our architecture can be used to derive time and energy efficient Turbo code decoders. The iterative decoding style

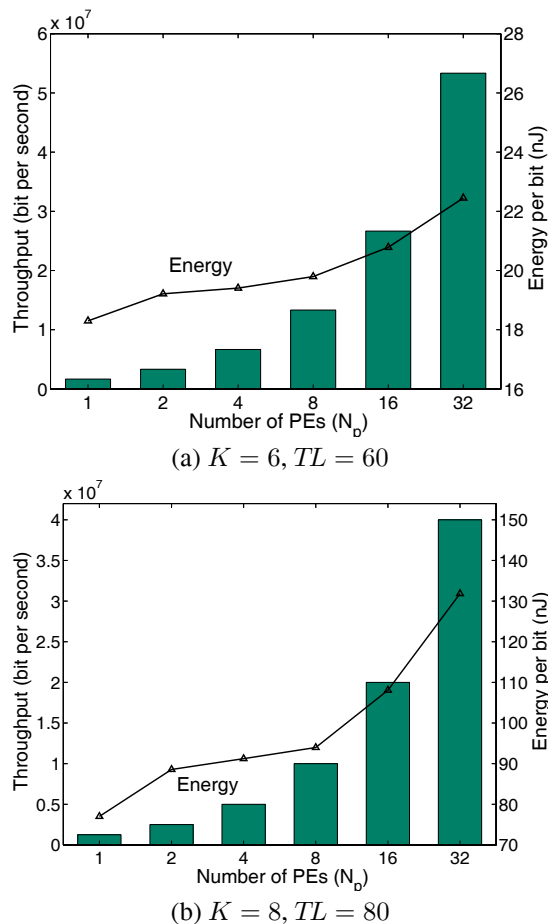


Fig. 6. Performance of the proposed Viterbi decoder

of Turbo code decoder results in a low data throughput for a serial trace-back based design while its long truncation length and back-updating causes high energy dissipation for a register-exchange based design.

6. REFERENCES

- [1] Altera, Inc., <http://www.altera.com>.
- [2] C. Dick, "The Platform FPGA: Enabling the Software Radio," *Software Defined Radio Tech. Conf. (SDR)*, 2002.
- [3] G. Forney, "The Viterbi Algorithm," *Proc. IEEE*, 1973.
- [4] J. Liang, "Development and Verification of System-On-a-Chip Communication Architecture," *Ph.D. Thesis*, Univ. of Mass., 2004.
- [5] Mentor Graphics, Inc., <http://www.mentor.com>.
- [6] S. Swaminathan, R. Tessier, D. Goeckel, and W. Burleson, "A Dynamically Reconfigurable Adaptive Viterbi Decoder," *ACM FPGA*, 2002.
- [7] Synplicity, Inc., <http://www.synplicity.com>.
- [8] T. Truong, M.-T. Shih, Irving S. Reed, E. Satorius, "A VLSI Design for A Trace-back Viterbi Decoder," *IEEE Trans. Comm.*, March 1992.
- [9] Walter Tuttlebee, "Software Defined Radio : Enabling Technology," *John Wiley & Sons, Inc.*, 2002.
- [10] Xilinx, Inc., <http://www.xilinx.com>.