An Asynchronous Viterbi Decoder for Low-Power Applications

B. Javadi[1], M. Naderi[1], H. Pedram[1], A. Afzali-Kusha[2], and M.K. Akbari[1]

[1]Department of Computer Eng. and Information Technology,
Amirkabir University of Technology, Hafez Ave., Tehran, Iran
{javadi, naderi, pedram, akbari}@ce.aut.ac.ir

[2]IC Design Lab., Department of Electrical and Computer Eng., University of Tehran,
Kargar Ave., Tehran, Iran
afzali@ut.ac.ir

Abstract. This paper presents a robust and low-power Viterbi Decoder designed based on asynchronous architecture. The design is based upon Quasi Delay Insensitive (QDI) timing model which leads to a robust functionality for the decoder. To lower the power consumption of the decoder further, an optimization technique to reduce the power dissipation is applied to add-compare-select (ACS) unit of the decoder. The simulation results shows a 20% reduction in the power consumption for the asynchronous design compared to the synchronous design in 0.35μm CMOS technology with a power supply of 2.5V. The throughput for the circuit is 50 MS/s.

1 Introduction

Many of digital transmission and storage systems use Forward Error Correction (FEC) techniques for correcting errors occurring during the transmission, storage, or retrieval of data. Viterbi Algorithm [480,2] belongs to a large class of FEC known as convolution codes which are used in a wide range of applications such as wireless communications, digital mobile telephony, digital TV broadcast, CD-ROMs, and magnetic disks. The basic idea behind the Viterbi decoder is to maximize the correlation between received vector and the table of possible codewords while sequentially performing the opposite operation of the encoder. The Viterbi decoder operates by finding the maximum likelihood of the decoding sequence.

The quality of a Viterbi decoder design is mainly measured by three criteria: coding gain, throughput, and power dissipation [3]. High coding gain results in low data transfer error probability while high throughput is necessary for high-speed applications. The design of Viterbi decoders with high coding gain and throughput is made challenging by the need for a low power circuit implementation. This requirement mainly stems from the fact that the Viterbi decoders are often placed in communication systems running on batteries where the power reduction is a must. Several attempts for reducing the power dissipation in the Viterbi decoder have been

J.J. Chico and E. Macii (Eds.): PATMOS 2003, LNCS 2799, pp. 471–480, 2003.

reported in the literature [4,5,6,7,8,9]. These research activities may be categorized in two following groups. The first group is based on changing the decoder architecture to reduce the power while the second group utilizes different circuit implementation techniques to minimize the power dissipation. Among the methods in the first group, one can mention the alteration of the architecture for Add Compare Select (ACS) unit [4,5], changing the memory management method [6], and altering the trace-back circuit [7]. Examples in the second group include SPL (Single ended Pass transistor Logic) implementation [8], and the self-timed circuit implementation [9]. Except for the Viterbi decoder based on self-timed architecture, other designs are synchronous.

In this paper, we present an asynchronous design for a Viterbi decoder based on Quasi Delay Insensitive (QDI) timing model. In this model, no constrain on the delay of the circuit element exists except for the isochoric forks [10]. Another advantage of QDI for FEC applications is its robust functionality for a wide range of parameter values. This paper is organized as follows. In Section 2, the conventional Viterbi decoder is described while the proposed design of the decoder, asynchronous synthesis and optimization techniques used to reduce the power dissipation are presented in Section 3. Finally, Section 4 contains the results and the conclusion of the work.

2 Viterbi Algorithm

A brief description of the Viterbi algorithm is given in this section where a detailed one can be found in [2]. A small system is chosen to illustrate the Viterbi encoding and decoding process. The algorithm used by the Viterbi Decoder belongs to a class of algorithms known as convolution codes. The rate of the convolution coder is defined as the number of input bits to the output bits.

2.1 Encoder Operation

A typical encoder is depicted in Fig. 1 where the rate of encoding is 1/2, i.e. the system encodes 1 input bit to 2 output bits.

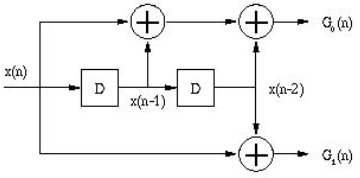

Fig. 1. Block diagram of the Viterbi Encoder.

In order to encode bit $x(n)$ from the input stream, this encoder creates two bits, namely $G_0(n)$ and $G_1(n)$ using the last 3 input bits, i.e. $x(n\text{-}1)$, $x(n\text{-}2)$, and $x(n)$. The encoding bits are produced from the following equations.

$$G_0(n) = x(n) + x(n\text{-}1) + x(n\text{-}2) \bmod 2 \quad (1)$$

$$G_1(n) = x(n) + x(n\text{-}2) \bmod 2 \quad (2)$$

The number of bits used for encoding one bit is called *constraint length* (k) (in this case $k = 3$), and the equations that describe the encoding are called *generator polynomials*.

2.2 Decoder Operation

At the destination, the decoder utilizes the *trellis diagram* to decode the received stream by finding the sequence with the maximum likelihood [1,2]. Fig. 2 shows the trellis diagram of the encoder in Fig. 1. Each node in the trellis diagram denotes one of the four potential pairs ($x(n\text{-}1)$, $x(n\text{-}2)$) of the last two decoded bits. The trellis can be seen as a flow-control diagram where each node represents a state and transitions happen depending on the input stream. From any node we can make a transition to one of two other nodes corresponding to receiving a 0 or a 1 as bit $x(n)$ at the input. The way the trellis diagram is constructed depends on the constraint length but not on the generator polynomials. The two numbers shown on every transition in the figure are the results of the above two generator polynomials. Each *stage*, which is associated with one decoded bit and two encoded bits, comprised of 2^{k-1} nodes.

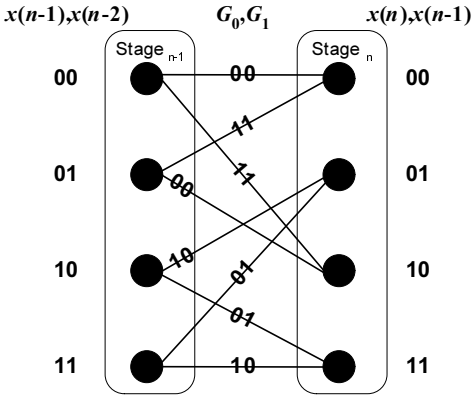

Fig. 2. Two stages of the trellis diagram.

There will be three distinct steps for the implementation of the decoding algorithm. The first step is called *branch metric* calculation where the received data symbols are compared to the ideal outputs of the encoder from the transmitter. The second step, called *path metric* computation where the path metrics of a stage is calculated by adding the branch metrics associated with the received symbol to the path metrics from the previous stage of the trellis diagram. The final step is *trace-back* process, when the survivor path, the path with the highest likelihood, and the output data are identified.

In the first step, the branch metric for each symbol of the input sequence is generated by calculating the Hamming distance between the input symbol and the expected symbol for each connection of the trellis diagram. The Hamming distance is the number of bits which two codewords are different. It can be realized by XORs and an adder. In the second step, the path metric is calculated. The calculations performed in this step, constitute a major part of the arithmetic operations performed in a Viterbi decoder. In this step, the decoder makes use of *Add-Compare-Select* (ACS) module as shown in Fig. 3. An ACS unit has 4 inputs, namely, two branch metrics (BM_1 and BM_2) and two path metrics (PM_{IN1} and PM_{IN2}), and two outputs, namely, the new path metric (PM_{OUT}) and the *Decision Bit*. The decision bit is the most important information generated by the ACS. It indicates that which sum between an input path metric and a branch metric generates the smallest result and was selected as output path metric or *local winner*. All decision bits generated at one stage are saved in a trace-back memory and later used for the trace-back operation.

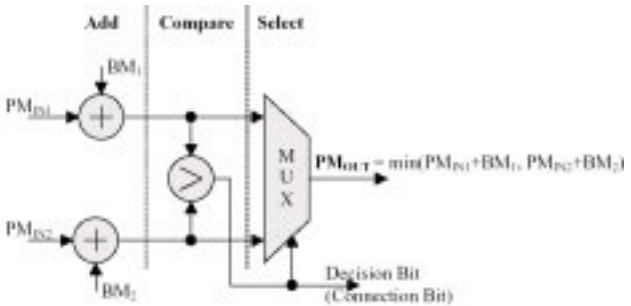

Fig. 3. Add-Compare-Select unit [8].

After repeating steps 1 and 2 for the desired depth of the trellis (determined by the memory and code gain requirements), step 3 will start for a back-trace path and find the *maximum likelihood* path with the best metric. To start this operating we need the state with the minimum path metric. This state is known as the *global winner*. The trace-back operation starts from the global winner state and trace the memory based on the local winner in the active state of each stage. The block diagram of the Viterbi decoder is shown in Fig. 4.

The global winner detection is not needed in each stage of the metric computation. It could be done based on the depth of the memory and the trace-back unit. Because the path metrics are ascending integer numbers they may overflow and in such situation the global winner detection may find a false winner. The false global winner will cause a malfunctioning of the trace-back unit leading to an incorrect output. The essential operation that should be added to the standard global winner detection is the *overflow prevention*. At the end of the trace-back step, the survivor path will be determined and the output data is reconstructed according to the trellis diagram.

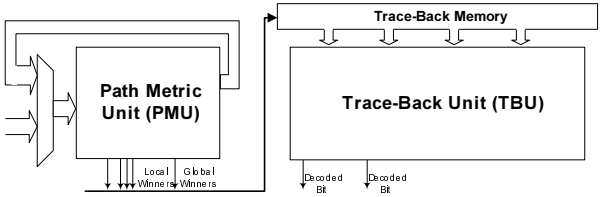

Fig. 4. Block Diagram of the Viterbi Decoder.

2.3 Decoder Structure

Five stages of the trellis diagram in the Viterbi decoder is shown Fig. 5 where the input bits are received at the top and the decoder gives the output decoded bits at the bottom. Each block in this figure is an ACS unit. The dashed lines correspond to two-way communications, i.e., the left state sends the path metric to the right state, while the right state sends a signal to the left state during the backtracking process.

Instead of a direct implementation of the diagram shown in Fig. 5, a more cost effective solution could be to implement with a smaller number of stages and reuse them in every cycle. The output metric of the last stage can be stored in a register to be used by the first stage in the next cycle. The comparison results of the states can also be held in the resisters that are used by the back-trace logic.

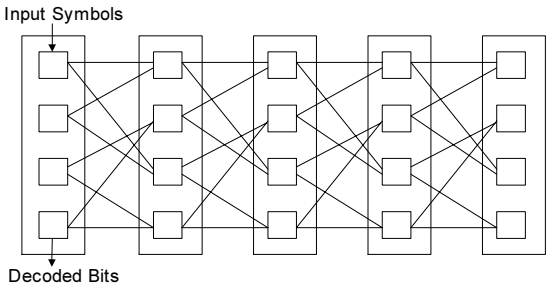

Fig. 5. Stages of the Viterbi decoder (k=3). Each block is an ACS unit.

The path metric unit could be implemented only in one stage. To obtain higher throughputs, one could utilize more stages to compute the metrics for many input symbols in each cycle of circuit operation. The number of bits for the metrics is an important parameter where a longer metric could lead to a higher decoding performance at a higher implementation cost. The last parameters are the depth of the trace-back unit and the number of decoded bits in each trace-back step. These parameters, which determine the Viterbi decoder, architecture will change the speed and the decoding performance of the circuit.

3 Decoder Implementation

In this section, we describe the architecture, the synthesis, and the optimization of the Viterbi decoder design in this work. Deciding on the parameters that are explained at end of previous section we find our Viterbi decoder architecture. In our design, the number of stages for the path metric computation is two while the depth of the trace-back section is four and two bits are decoded in each stage. Fig. 6 and fig. 7 contains the detailed designs of the Path Metric Unit and the Trace-Back Unit of fig. 4. The path metrics are 8-bit wide while the branch metrics are 2-bit wide.

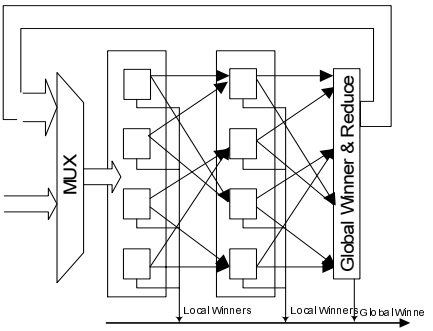

Fig. 6. Path Metric Unit.

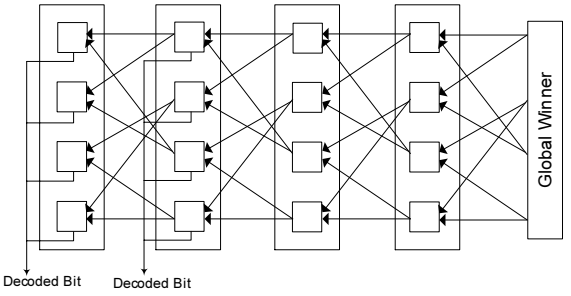

Fig. 7. Trace Back Unit.

3.1 Asynchronous Synthesis

For the implementation of the decoder, we have used the asynchronous design style. In general, the asynchronous implementation has a greater potential for low power applications, although the synthesis tools are less availability for this design style. In our design, an asynchronous ripple carry adder is utilized in the ACS unit. Since the numbers of bits for the branch metric and the path metric are different, the adder only adds the required bits. The asynchronous adder consumes less power compared to the synchronous counterpart due to the fact that the metrics are small numbers, in most cases, and, hence, a large part of the circuit is not active. Another advantage of using this style for the decoder is that in the trace-back only one unit within each stage is

required to be active, which lead to lower power dissipation in the asynchronous implementation.

Asynchronous design mythologies are classified based on the timing model being used and may be categorized to bounded-delay and delay insensitive models [12]. These two main categories have many sub-models. The bounded-delay models have timing consideration similar to the synchronous design techniques with this difference that the clock is not used. The delay insensitive models use handshaking for the synchronization of the data transfer. The implementation of pure delay-insensitive circuits is very complex and has a significant overhead. To reduce the complexity and the overhead, one needs to tolerate some timing constrains. One of the most powerful with minimum timing constrain models is Quasi Delay Insensitive (QDI) method that was introduced by A. Martin [10].

We have used QDI model and Martin synthesis method in design of this circuit. Martin synthesis method converts a description of circuit in CSP (Communicating Sequential Process) directly to CMOS instead of converting it to a gate-level description as many synchronous synthesis tools do. For this research, some of the synthesis steps were synthesized using the tools developed by the asynchronous design group of [13] while the rest were synthesized manually.

Here, the design was based on the dual-rail encoding for data transfer and the four-phase handshaking protocol for the communications between the circuit elements. Many of the functional blocks are based on PCFB (Pre-Charge Full Buffer) template [14]. With these blocks, there is no need for buffers because they have an internal buffer.

3.2 ACS Optimization

As mentioned before, a large part of the power is consumed in the ACS units and, therefore, the optimization of these units can reduce the total power consumption of the circuit. Each ACS consists of two add and one compare operations. Considering a block of two ACS neighbors in butterfly architecture [15], each block consists of four 8-bit to 2-bit adder and two 8-bit comparators/selectors where the path metric has 8 bits while the branch metric has 2 bits. In [4], a re-arrangement of the butterfly operations is proposed to reduce the number of operations. This architecture is shown in fig. 8 where in this figure, BM and PM are Branch Metric and Path Metric, respectively Sa and Sb indicate inputs and S0, and S1 shows two possible path for 0 and 1 as output. his re-arrangement reduces the circuit to two 2-bit subtractors, one 8-bit subtractor, two 8 to 3-bit comparators/selectors, and two 8 to 2-bit adders, hence, a considerable reduction in the power. Except for the comparators, all the components are implemented using QDI method.

The asynchronous implementation of the comparators using QDI model leads to large area and power consumption. This is due to the fact that they are active almost all the times and all the input bits are used in this operation. This gives rise to a rather large power and area overhead for the normal dual-rail coding in QDI design, and, therefore, the two comparators are implemented using Micro-Pipeline method [16], which has a lower overhead. The Micro-Pipeline has a normal function covered by a delay-insensitive asynchronous control circuit.

4 Results and Discussion

We used a C++ program to simulate the behavior of the design and to measure the coding gain (decoding performance) of the decoder. After designing the circuit, the data flow was simulated using Verilog. The Martin method was used to synthesize the decoder to switch level description [10]. The description was converted to CMOS circuit implementation using a 0.35μm technology. Each block was simulated independently using SPICE to obtain its timing behavior and power consumption. Finally, the entire circuit was simulated in SPICE.

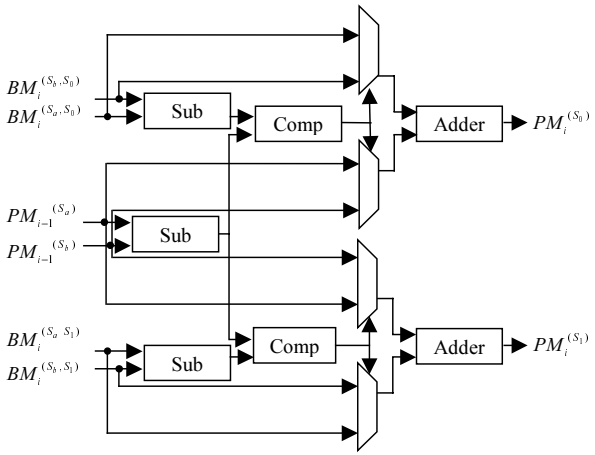

Fig. 8. Re-arranged butterfly ACS [5].

Table 1 shows the estimated power consumption for the circuit by adding all the blocks power and the power consumption from simulation. The results show that the power consumption is half of the worst-case estimation. So, at the average half of the circuit blocks are inactive during normal functionality of the circuit.

Table 1. SPICE simulation results

Simulation	Power (mW)	
	V_{dd}=3.3V	V_{dd}=2.5V
Worst-Case Estimation (Block Simulation)	0.285	0.175
Total Simulation	0.166	0.085

As the table 1 shows about half of the power is consumed in ACS units. Optimizing the power of ACS units can reduce the total power consumption of the circuit. The reduction in the power consumption achieved for the rearranging the ACS structure in the proposed model compared to the same change in the conventional butterfly ACS is given in Table 2.

The simulation results show that the asynchronous design has more potential for low-power applications. This is due to the in the asynchronous implementation, the blocks are only active when needed and so the activity factor is lower for asynchronous circuit.
Also, in [4] by adding a glitch reduction logic, a reduction of 7% in the power dissipation was obtained while the latency was not changed at a cost of 3% increase in the area compared to the ACS unit introduced in Fig. 8. Since the QDI timing model is hazard-free model [10], there is no need to the glitch reduction logic circuit.

Table 2. Re-arranged butterfly ACS architecture results

Architecture	Technology	V_{dd}	Optimization
Synchronous butterfly ACS [5]	0.8 µm	5 V	30.2%
Asynchronous butterfly ACS [This work]	0.35 µm	2.5V	42%

Table 3 shows a comparison of the Viterbi decoders of this work and some other low-power Viterbi decoders reported in the literature. The power consumption in the case of the optimized decoder with a power supply of 2.5V is lowest among the decoders of Table 3. This circuit could operate up to 50 MS/s.

Table 3. Comparison of Viterbi decoder designs.

Design	Technology	V_{dd} (V)	Power (mW)
Synchronous Reference [9]	0.35µm	n/a	203
Systolic Array [7]	0.5 µm	3.3	380
SPL [8]	0.35 µm	2.5	88
Self-Timed [9]	0.35 µm	n/a	9.2
This work	0.35 µm	3.3	166
This work	0.35 µm	2.5	85
This work-Optimized ACS	0.35 µm	3.3	109
This work-Optimized ACS	0.35 µm	2.5	62

5 Summary and Conclusion

The design and implementation of a low-power asynchronous Viterbi decoder was presented. The design was based upon Quasi Delay Insensitive (QDI) timing model which can be used for robust circuits. It was shown the ACS unit consumes about half of the power of the decoder and, hence, to reduce the power, an optimization technique was applied to this unit. The simulation results of the optimized asynchronous decoder showed a 20% reduction in the power consumption compared to the synchronous design in 0.35µm CMOS technology with a power supply of 2.5V. Therefore, the asynchronous Viterbi decoder could be a good candidate for low-power applications.

Acknowledgement. The authors wish to thank Dr. Babak Sadeghian for his useful remarks and discussions and Dr. Mehdi Sedighi for his helps about SPICE simulations and power estimation methods.

References

1. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, in *IEEE Trans. Information Theory*, vol. 13, pp. 260–269, April 1967.
2. G. Davis and Jr. Forney. The Viterbi algorithm, *Proceedings of IEEE*, vol. 61, no. 3, pp. 268–278, Mar. 1973.
3. Xun Liu and M. C. Papaefthymiou. Design of a High-Throughput Low-Power IS95 Viterbi Decoder, in *Proc. of Design Automation Conf. (DAC)*, pp. 263–268, June 2002.
4. J.H. Ryu, S.C. Kim, J.D. Cho, and H.W. Park, and Y. H. Chang. A New Lower Power Viterbi Decoder Architecture With Glitch Reduction, in *Proc. of the First IEEE Asia-Pacific Conf. on ASICs,* pp. 83–86, Aug. 1999.
5. Chi-Ying Tsui, R.S.K. Cheng, and C. Ling. Low power ACS unit design for the Viterbi decoder, in *Proc. of the IEEE Symp. on Circuits and Systems*, pp. 137–140, 1999.
6. K. Hu, M.D. Caldwell. A Viterbi Decoder Memory Management System Using Forward Traceback and All- Path Traceback, in *Proc. of Int. Conf. on Consumer Electronics*, pp. 68–69, 1999.
7. D. Ahmadian, M.N. Azarmanesh, and Kh. Hadidi. VLSI Design of a Viterbi Decoder using Systolic Array for Trace-Back operation in 0.5u µm CMOS, in *Proc. of 10th Iranian Conf. on Elec. Eng.*, Tabriz, Iran, pp. 524–532, May 2002.
8. I. Bogdan, M. Mumunteanu, P.A. Ivey, N.L. Seed, and N. Powell. Power Reduction Techniques for a Viterbi Decoder Implementation, *ESPLD 2000 (European Low Power Initiative for Electronic System Design) Third International Workshop*, Rapallo, Italy, ISBN 90-5326-036-6, pp 28–48, July 2000
9. P.A. Riocereux, E.M. Brackenbury, M. Cumpstey, and S.B. Fruber. A Low-Power Self-Timed Viterbi Decoder, in *Proc. 7th International Symp. on Asynchronous Circuits and Systems,* pp. 15–24, 2001.
10. A. J. Martin. Programming in VLSI: From Communicating Processes to Delay-Insensitive Circuits, in *UT Year of Programming Institute on Concurrent Programming,* C. A. R. Hoare, Ed. MA: Addison-Wesley, pp. 1-64, 1989
11. S. Ranpara and Ha Dong Sam. A low-power Viterbi decoder design for wireless communications applications, in *Proc. 12th Ann. IEEE Int. AISC/SOC Conf.,* pp. 377-381, Sep. 1999.
12. Scott Hauck. Asynchronous Design Methodologies, *Proceedings of the IEEE*, vol. 83, no. 1, pp. 69-93, Jan. 1995.
13. http://ce.aut.ac.ir/async.
14. A.J. Martin. Asynchronous Data paths and the Design of an Asynchronous Adder, *Formal Methods in System Design*, vol. 1, no. 1, pp. 119-137, July 1992.
15. I. Kang and A. N. Willson. A 0.24mW, 14.4 kbps, r=1/2, k=9 Viterbi Decoder, in *the Proc. of IEEE CICC*, pp. 603-606, 1997.
16. I. E. Sutherland. Micropipelines, *Communications of the ACM*, vol. 32, no. 6, pp. 720-738, Jun. 1989.