No-Handshake Asynchronous Survivor Memory Unit for a Viterbi Decoder

Wei Shao
APT Group, School of Computer Science,
University of Manchester, Oxford Road, M13 9PL
Email: shaow@cs.man.ac.uk

Linda Brackenbury
APT Group, School of Computer Science,
University of Manchester, Oxford Road, M13 9PL
Email: lbrackenbury@cs.man.ac.uk

***Abstract*— The *Survivor Memory Unit* (SMU) is a vital part of a Viterbi decoder design. So far, classical implementations of SMU employ the register exchange or the trace back approaches. In the conventional trace back implementation, a read-write RAM architecture is generally adopted which requires a large size of memory. This gives the SMU design both area and power overhead. This paper presents a new no-handshake asynchronous approach to implement the trace back method. The SMU design based on this new architecture is a mixed synchronous and asynchronous circuit. Post-layout simulation results on a $.18\mu m$ process show the new architecture saves more than 84% of the power dissipated compare with a synchronised SMU design using a low power logic family and 30% compared with a handshaking asynchronous design.**

I. Introduction

Viterbi decoders are normally used in communications systems, such as GSM, satellite and cable digital television. In these systems convolution codes can be used to encode the data stream into a longer stream carrying redundant information. The Viterbi decoder is then used to decode the received data and eliminate errors caused by the noise and reconstruct the original signal [1].

In the decoder, the Survivor Memory Unit (SMU) is the final block. It keeps a history of the computations made on the input data stretching back over many time slots from the current slot. This history enables the SMU to determine and output the most likely data based on the estimated states of the encoder.

This paper is organized so that the Viterbi algorithm and trace back SMU designs are reviewed and discussed in the next section; this enables the implementation issue of the trace back process to be revealed. The third section of this paper describes the new SMU trace back architecture with the analysis of its timing. Finally, the last section gives results of the measured power and error correction performance on both a post-layout CMOS circuit and a FPGA.

II. Viterbi algorithm and trace back SMU designs

A. Convolutional Encoding

Digital data can be convolutionally encoded using k input bits using n modulo 2 adders to create n encoded digital bit streams which are transmitted; k is called the constraint length. Figure 1 shows a simple example of a $1/2$ rate convolutional encoder, with $k = 3$. Comprised with registers, an encoder

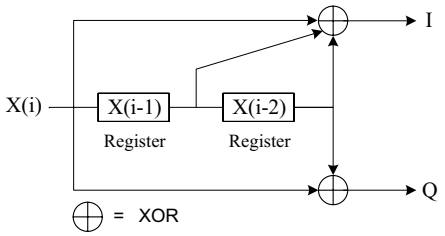


Fig. 1. A simple $1/2$ rate, constraint length 3 ($k = 3$) convolutional encoder.

with a constraint length of k has 2^{k-1} states. The encoding process of an convolutional encoder can be generalized and described as a states transitions process with a trellis diagram, shown in Figure 2. In Figure 2, the encoder states are indicated

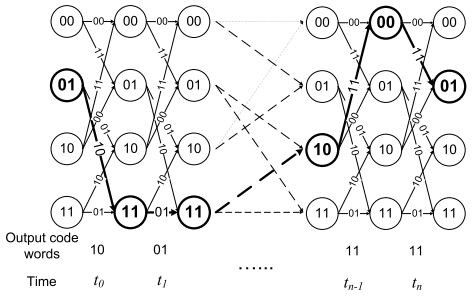


Fig. 2. The trellis diagram represents state transitions of a rate $1/2$, $k = 3$ convolutional encoder from time t_0 to time t_n.

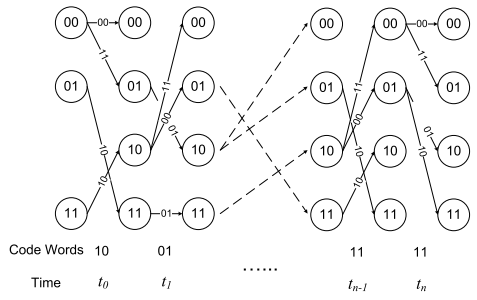
in circles where the states transitions and the correspondance encoder outputs are represented as the arrows from one state to another. Generally, if the encoder is at state j or $j + 2^{k-2}$ then it moves to state $2j$ for a '0' input and to state $2j + 1$ for a '1' input ($0 \leq j \leq 2^{k-2} - 1$).

B. Viterbi algorithm and decoder architecture

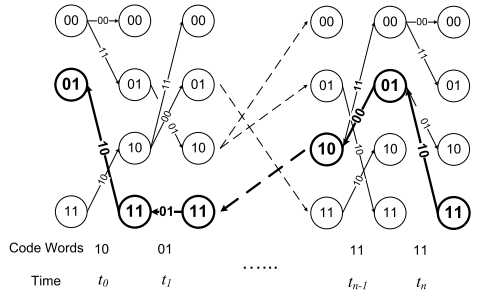
To decode the convolutional code, the Viterbi algorithm performs two stages of operations. The first stage of Viterbi decoding process involves branch metrics calculations and so called Add-Compare-Select (ACS) operations. With these operations, the Viterbi algorithm measures the likelihood of all the states transitions based on the code words so that only one of the two transitions into each state is selected as the survivor

1-4244-1378-8/07/$25.00 ©2007 IEEE.

branch at every time step. Figure 3(a) illustrates the results of this process as a trellis diagram, in comparison to Figure 2, and only the survived states transitions and the correspondence states are shown. The trace back process is basically a reverse



(a) Trellis of the survivor branches after ACS operations.



(b) Trace back the survivor trellis process of the Viterbi algorithm.

Fig. 3. Viterbi decoding process presented in trellis diagram.

of the decoding process, as indicated as the highlighted arrows in Figure 3(b). The trace back starts from a random state of time t_n and traces backward through the path formed by the history of the survivor branches. For example, the trace back in Figure 3(b) is started from state 11. Since paths that start from all states at the end of this trellis are all merged into state 10 at the second trace back step, there is only one unique states sequence can be obtained by the rest of the trace back steps. Therefore, this state sequence can be used to generate the decoded data. According to the 'rule of thumb' [2] [3], the decoding process needs to perform L steps of trace back where $L \geq 5k$ to guarantee the convergence, .

A Viterbi decoder normally comprises three blocks [1] as shown in Figure 4. The Branch Metric Unit (BMU) computes the branch metrics of the input symbols. The Path Metric Unit (PMU) performs ACS operations to obtain the survivor branch selection for each state. The trace back Survivor Memory Unit (SMU) keeps a history of these selections in a memory for a number of time steps and determines the decoded output from trace backs.

C. Trace back algorithm and SMU designs

The trace back algorithm can be generalised as a recursive updating process where the trace back recursion estimates the previous encoder state S_{n-1} according to the current state S_n

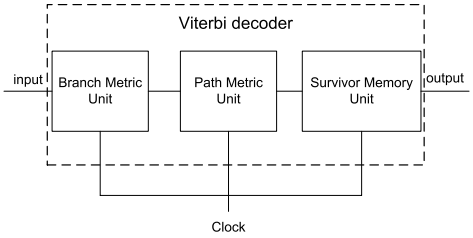


Fig. 4. Viterbi decoder block architecture

[4], which consists of m bits $(m = k - 1)$, where

$$S_{n-1} = S_n[m-2:0]d_n^S \quad (1)$$

for the common radix-2 trellis, where $S_n[m-2:0]$ represents the bottom m-1 bits of S_n. d_n^S is the one-bit survivor branch selection from the PMU and is read from the memory location addressed by state index S_n and time index n; the previous state S_{n-1} is obtained by simply removing the most significant bit, $S_n[m-1]$, of S_n and appending d_n^S as the least significant bit.

The most common synchronous trace back SMU design is the one-pointer architecture [5]. In this approach, a memory is used to store all branch selections and is partitioned into a write block, a merge block and a read block. During the decoding process, the new survivor branch selections are written to the write region while the previous survivor branch selections are traced and read from the merge and read block, simultaneously. In order to match the SMU in/out throughput, the trace back and read operations must complete before the write region is fully occupied. This normally results in a high frequency of memory read operations.

On the contrary to a synchronised design [1], the timing is individually scheduled in an asynchronous system and multiple trace backs are performed concurrently and continuously controlled by handshakes. However the major implementation issue of the asynchronous design is the overhead of handshake logic that consumes extra power.

III. The new SMU design

In order to take the advatages of asynchronous trace back and also to avoid its implementation overhead, we propose an mixed synchronous and asynchronous trace back SMU design.

A. The new SMU architecture

Figure 5 illustrates the new SMU top level architecture. As shown in Figure 5, the design consists of 3 major blocks: the Survivor Branch selections Memory, the Trace Back Path, the Decoding block. The memory and the Decoding block are synchronised to the global clock whereas the Trace Back Path runs mainly asynchronously. The memory block is implemented as L slots latches rather than RAM and is selectively loaded from the system clock. Therefore, the stored selections are all output straightaway and synchronously to the Trace Back Path

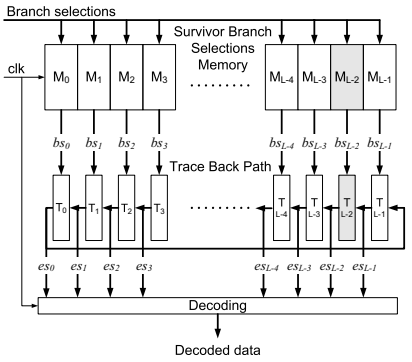


Fig. 5. The new SMU architecture, which consists of three major blocks. The Survivor Branch selections Memory and the Decoding block are synchronised by the clk while the Trace Back Path is running asynchronously.

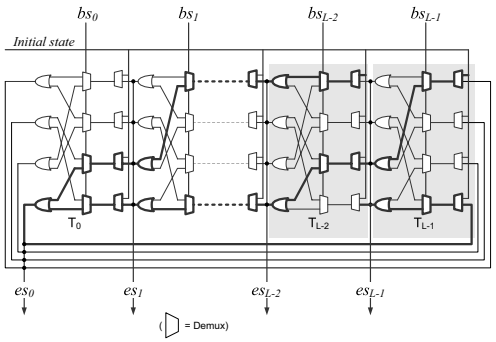


Fig. 6. The structure of the Trace Back Path design for a rate $1/2$ and $k = 3$ decoder which is a direct implementation of the trellis structure.

when being updated. This avoids the repeated memory read operations usually performed in a conventional SMU.

The major novelty of this new SMU design is in the Trace Back Path design. The encoder state in the Trace Back Path is encoded as $1 - of - 2^{k-1}$ bits data other than the k-bit index in a normal SMU. For example, the states 00, 01, 10, and 11 of an $k = 3$ encoder are represented by 4-bit data 0001, 0010, 0100, and 1000 respectively. With this state encoding scheme, the Trace Back Path is implemented as the trellis structure shown in Figure 2 and, therefore, a trace back is carried out as an asynchronous logic '1' signal propagation. In the Trace Back Path when a memory slot M_{L-2}, for instance, is updated at time t, its correspondence trace back stage, T_{L-2}, will be initiated with a new random encoder state, es_{L-2}, which is passed to the previous stage T_{L-3} to update its encoder state, es_{L-3}. This updating process is then propagated asynchronously as a trace back through the rest of the Trace Back Path. In the next clock cycle $t + 1$, the memory slot M_{L-1} will be updated by new branch selections and the trace back stage T_{L-1} also initiate a new trace back which is then running asynchronously thereafter and concurrently with the previous trace back.

In the design trace backs are started at each clock cycle in the increment trace back units from left to right in Figure 5. Since they will eventually converge before reaching the beginning of the trace back path where the estimated encoder state is captured, it is not necessary to use state holding devices in the trace back path.

B. The design of Trace Back Path

The trellis structure of the Trace Back Path is illustrated in Figure 6. As Figure 6 indicates, the Trace Back Path comprises only simple combinational circuits, i.e. OR gates, multiplexers and demultiplexers. There is no handshake logic to control the asynchronous trace back signals. For example, corresponding to the example in Figure 5, when the memory slot M_{L-2} is updated the new branch selections bs_{L-2} will change the connections of the demultiplexers in the trace back stage T_{L-2} in Figure 6; while the multiplexers of this stage is selecting the initial encoder state, e.g. state 0001. Therefore, the logic '1' signal is passed through the top multiplexer, the demultiplexers and the OR gate of this stage and then travels asynchronously as a new trace back on the rest of the connected path, which is highlighted in Figure 6. As the new trace back is started, the encoder state es_{L-1} produced by stage T_{L-1} is selected and synchronised by the Decoding block to yield decoded data. Then in the next clock cycle, a new trace back will also be started from this stage and goes through a different path. However, since they will eventually merged by the OR gate of stage T_0, as shown in Figure 6, the new trace back will not affect the result of the last trace back and, thus, the estimated encoder state es_0 can be easily synchronised and used to produce the decoded data.

This Trace Back Path architecture provides 3 major features to the new SMU design. Firstly, this Trace Back Path allows multiple asynchronous trace backs running concurrently, thus the number of memory slots can be minimised. Secondly, in this architecture trace backs are merged by OR gates which is a simple but efficient way of implementing the path convergence. With this approach, a trace back propagation stops once it is 'OR'ed with an existing trace back resulting in no further transitions occurring at the OR gate output. This minimises the power dissipated in trace backs. Thirdly, a trace back in the new architecture is implemented as a 1-bit logic '1' signal propagation without any clock or handshake logic. Therefore, the number of circuits transitions caused by the trace back process is also minimised so that the power dissipation can be significantly reduced.

C. Timing skew analysis

1) Negative timing skew: All paths trace back simultaneously and because of element tolerance and differences in wire length, there will be a variation in time between the arrival of trace back decisions at the oldest timeslot. If large enough, this could cause incorrect data to be determined for clocking into the output flip flop. This can be refered to as negative timing skew and is illustrated in Figure 7. The solid line from state S_1 represents the global winner propagation with time and the dotted line represents all the other 'loser' states. The slower

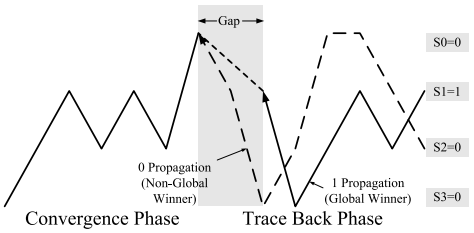


Fig. 7. Trace back gap caused by timing skew.

propagation of the winner means that the loser '0' states can combine with a zero on the winner path from the previous timeslot to indicate a period where no winner is indicated; this is indicated by the shaded region in Figure 7. Were this to happen at the time the decoded data is clocked into the output flip flop then incorrect data would be decoded and output.

The gap caused by the negative timing skew is determined by the delay differences of different paths in the Trace Back Path at each stage. The period ρ of a gap is upper bounded by

$$\rho \leq \sum_{n=0}^{L-1} (d^n_{max} - d^n_{min}) \tag{2}$$

where d^n_{max} and d^n_{min} are the maximum and minimum delays of stage n. Therefore, the possible error caused by the negative timing skew can always be avoided by shifting the clock edge of the synchronisation flip flop out of the invalid data gap. Thus, as long as the ρ is less than a clock cycle, the valid decoding operations of the Trace Back Path are guaranteed.

IV. Test Results

The Viterbi decoder design with this new SMU architecture is implemented in both CMOS circuitry and on a FPGA. Power simulations are used to estimate the power figures of the CMOS and FPGA implementations whereas the BER performance is measured from the in circuit tests on FPGA.

A. CMOS implementation results

The new SMU operates at frequencies 45MHz and 100MHz; and uses a 0.18 micron technology and a 1.8V supply voltage. The layout has been automatically generated using a commercial tool from logic schematics comprising elements from an in-house library of conventional CMOS logic circuits. This approach results in random delays in the trace back path. Table I summarizes the characteristic of the SMU layout. The design has been tested by running Nanosim post-layout simulations. In the post-layout tests, three different size ($5k$, $10k$ and $50k$) of random data patterns were generated and added with white Gaussian noise according to the signal noise ratio in decibels. The resulting output bit error rate (BER) and averaged power consumptions for different signal to noise ratios of code rate $1/2$ and is given in Table II. The throughput is 45Mbit/sec in these simulations which is equal

TABLE I

Characteristics of the new SMU core

Throughput	$45Mbits/s$ and $100Mbits/s$
Rate	1/2 (or punctured $2/3$ to $7/8$)
Trace back length	64
Core size	$1.05 \times 1.05 = 1.10mm^2$
Transistors	$241K$
Technology	$.18\mu m$ standard cell

to the targeted throughput of the reference designs in [1] and [6]. It can be seen that the power increases only relatively

TABLE II

New SMU BER and power consumption in decoding $1/2$ codes

S/N **in** (dB)	**Input** BER	**Output** BER	**Normal VB** BER	**Power** (mW)
0.4	0.07335	0.1524	0.12	6.65
1.7	0.04234	0.0293	0.032	6.31
2.6	0.02834	$< 4.85e-5$	7.43e-3	6.24
3.5	0.01904	$< 2.31e-5$	1.34e-4	6.17

slowly with increasing input BER. This suggests that trace backs in the new SMU consume only a small portion of the overall SMU power.

The average power consumption and area of this new SMU design is compared in Table III with low power Viterbi decoder designs from [6] and [1], which are implemented with single-ended pass-transistor logic (SPL) and asynchronous logic respectively. Since the SPL design is implemented with the .35 micron CMOS technology and a 3.3V supply voltage, its power and area are scaled down by factors of 8 and 4 respectively. The power results have all been scaled (where required) to a 45Mb/s data rate for a $.18\mu m$ process running from 1.8V.

TABLE III

The new SMU power consumption comparing with other low power SMU designs at 1.8V and 180nm.

	New SMU	**SPL[6]**	**Asynch[1]**
Decoder Area (mm^2)	N/A	1.64	1.96
SMU Area (mm^2)	1.10	1.04	~ 0.98
No. of States	64	64	64
Avg. Power (mW) of Decoder @45Mb/s	N/A	62.5	~ 18
Avg. Power (mW) of SMU @45Mb/s	6.31	39.38	~ 9

According to the comparison, the new SMU architecture only uses 16% of the power dissipated in the low power design using SPL logic. In the asynchronous design from

[1], the overall decoder power consumption is also much less than the synchronous SPL design from [6]. Comparing to the power figure from the asynchronous design, the new SMU architecture can provide 29.8% power reduction. So, the new SMU is the most power efficient design amongst these three.

B. FPGA implementation results

Since the post-layout power simulation is extremely computation intensive, it does not provide BERs that are accurate enough at low noise levels. The design has therefore been transferred to a $90nm$ FPGA. Two Viterbi decoders, $R = 1/2$, $k = 7$ and $R = 1/2$, $k = 3$, using this new SMU architecture were implemented and tested on a Virtex4 XC4VSX35 FPGA board. The test framework includes a reference Viterbi decoder IP core from Xilinx which is compatible with many common standards, e.g.DVB, 3GPP2, and IEEE802.16. The power figure is estimated by the Xpower tool from Xilinx. The size of the post place and route design is larger than the Xilinx IP core, as shown in Table IV. It shows the Viterbi decoder with

TABLE IV
NUMBER OF SLICES THE VITERBI DECODER OCCUPIED WITH THE NEW SMU ARCHITECTURE

	RAM Blocks	Slices
VD with new SMU(k=3)	0	284
Xilinx IP(k=3)	2	224
VD with new SMU(k=7)	1	3,686
Xilinx IP(k=7)	5	2,423

the new SMU is 21% and 34% larger than the Xilinx Viterbi decoder of constraint lengths 3 and 7, respectively. All of the designs can operate at a frequency of 100MHz.

Monte Carlo simulations were used to get the BER results shown in Figure 8. It shows the measured BERs from the

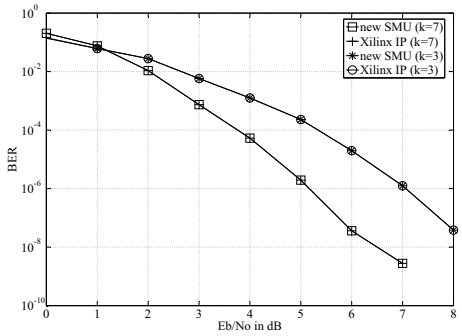


Fig. 8. BER performance from FPGA tests.

Viterbi decoders with the new SMU match the standard Viterbi decoders BERs. There are only small differences at high noise levels due to the difference between the trace back speeds in the new SMU and the Xilinx Viterbi decoder.

The power simulations are run for 5k symbols at an $Eb/No = 0dB$ with a 100MHz clock. The results are listed in Table V. These power figures indicate that for small constraint length, such as $k = 3$, the proposed new SMU saves 28% dynamic power of a standard SMU implementation on the FPGA. However, for $k = 7$, the saving becomes only 4%. This is due to the exponential increase of trellis wire connections. For a small constraint length, the increased power dissipation of these extra connections is relatively small. However, as the number of the connections increases exponentially with the constraint length, the power overhead becomes significant and reduces the power saving of this new SMU.

TABLE V
DYNAMIC POWER CONSUMPTION OF THE VITERBI DECODER WITH THE NEW SMU IMPLEMENTED ON FPGA

	Trace back length	Power(mW)
VD with new SMU(k=3)	48	20.3
Xilinx IP(k=3)	48	28.2
VD with new SMU(k=7)	48	135.6
Xilinx IP(k=7)	48	146.2

V. CONCLUSION

In this paper, a new power efficient SMU architecture has been proposed. In this new design, data is entered into the SMU synchronously while the tracing back to reconstruct the encoder states history is decoupled from the data entry and free runs asynchronously. The trace back logic is entirely combinational thus avoiding the complex control and handshakes normally associated with asynchronous design. These measures give the new design an equivalent bit error performance but at lower power than other SMUs.

Although the trace back may cause an error due to the negative timing skew, it can be fixed by shifting the clock edge of the synchronisation flip-flop. This is confirmed by the measured BERs from the FPGA tests. This indicates that to achieve an efficient SMU design without handshakes is not without problems and that careful timing analysis and simulation are required at the working frequencies and on the targeted process.

On CMOS circuitry, the new SMU architecture is more power efficient compared with a low power design using SPL as it reduces the power dissipated in the SMU by a factor of 6. Thus, this proposed architecture is believed to be suitable for low power CMOS trace back SMU implementations.

ACKNOWLEDGMENT

The authors would like to thank UK EPSRC for funding this work and the project of designing a low power consumption Viterbi decoder.

REFERENCES

[1] P. Riocreux, L. Brackenbury, M. Cumpstey, and S. Furber, "A low-power self-timed Viterbi decoder," in *Proc. Async*, Salt Lake City, Utah, USA, Mar. 2001, pp. 15–24.
[2] G. C. Clark and J. B. Cain, *Error-Corection Coding for Digital Communications*. New York: Plenum Press, 1981.
[3] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*. New York: Wiley-IEEE Press, 1999.
[4] C. Lee, "A Viterbi decoder with efficient memory management," *ETRI Journal*, vol. 26, pp. 21–26, Feb. 2004.

[5] C. M. Radar, "Memory management in a Viterbi decoder," *IEEE Trans. Commun.*, vol. 29, pp. 1399–1401, Sept. 1981.
[6] M. C. Munteanu, "Low power design of integrated circuits," MPhil thesis, The Department of Electronic and Electrical Engineering, The University of Sheffield, Sheffield, Aug. 2000.