IMPLEMENTATION OF HIGH THROUGHPUT SOFT OUTPUT VITERBI DECODERS

Engling Yeo, Stephanie Augsburger, Wm. Rhett Davis, Borivoje Nikolić

University of California, Berkeley, CA 94720

ABSTRACT

The architectural considerations for VLSI implementations of soft output Viterbi decoders are presented. Structural transformation of the add-compare-select structures provides high throughput with small area overhead. Modifications to the survivor memory unit and a comparison between the register exchange and memory traceback methods are highlighted. A 4mm^2 demonstration chip, consisting of two parallel, 8-state, 7-bit soft output Viterbi decoders, has been implemented in 0.18μm CMOS technology, and decodes at 500Mb/s with 1.8V supply. These decoders are used with Turbo codes, which have been demonstrated to achieve information rates close to the Shannon limit.

1. INTRODUCTION

This work discusses architectures and implementations of decoders applying the Soft Output Viterbi Algorithm (SOVA) [1]. These decoders can be employed as soft-input-soft-output (SISO) decoders for a Turbo coded system in magnetic recording [2], where high throughputs are required. Figure. 1 shows an example that comprises a serial concatenation of an 8-state Octal(13) convolutional encoder, with an enhanced partial response class-4 (EPR4) channel.

Each SOVA decoder outputs sign-magnitude values that consist of one decoded bit (hard output) and a multi-bit unsigned log-likelihood ratio (soft output). The soft output is based on the difference in path metric between the two most-likely (ML) paths that trace back to complementary bit decisions.

The arithmetic computation of the SOVA decoder will be discussed in Section 2. Section 3 highlights the micro-architectural analysis of the add-compare-select (ACS) structures, which are the traditionally speed-bottleneck of Viterbi decoder designs. Continuing the emphasis on high throughput rates, Section 4 and 5 describe the use of deeply pipelined mechanisms for the traceback, equivalence detecting, and comparison of competing path metrics. Finally, Section 6 discusses the system architecture of a physical implementation, as well as the design flow, testing methodologies and measurement results.

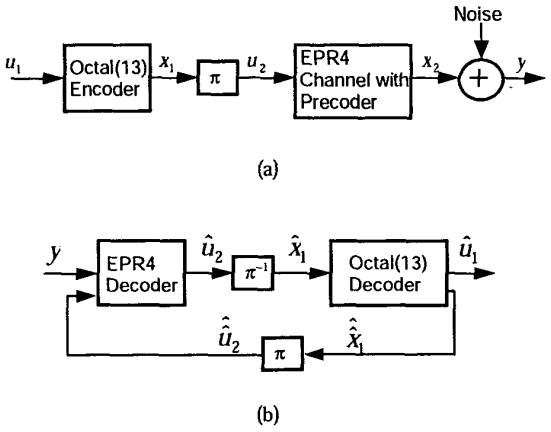

Figure. 1 Serial Turbo (a) encoder and (b) decoder with blocks separated by interleavers/ deinterleavers (π /π^{-1}).

2. SOVA DECODER

The SOVA decoder outputs the log-likelihood of a correctly decoded bit. This value is approximated by the difference between the path metrics of the two most-likely (ML) paths, α and β, that trace back to complementary bit decisions, $\hat{x}$ and $\bar{\hat{x}}$, respectively.

Similar to the Viterbi decoder, a bit decision, $\hat{x}$, is always based on the ML path, α. Since α and β provide the most-likely paths to all possible outcomes of each received bit, the probability of an erroneous decision can be approximated in (1). This assumes that the path metrics, M_α and M_β, of these two ML paths dominate over all other possible paths. The log-likelihood of a correct output by the SOVA decoder is developed in (2).

The SOVA decoder implementation uses a two-stage procedure [5], [7] to determine the two ML paths. The right section of Figure 2 shows that α is determined using the regular Viterbi algorithm with an L-step traceback. Traditional Viterbi decoders employ add-compare-select (ACS) structures to perform path selection at each iteration of the algorithm. The SOVA decoder further requires that each ACS outputs the difference in path metrics between the pair of competing paths.

0-7803-7587-4/02/$17.00 ©2002 IEEE

$$P_{err} = \frac{\exp(-M_\beta)}{\exp(-M_\alpha)+\exp(-M_\beta)} = \frac{1}{1+\exp(\Delta)} \quad ; \quad \Delta = M_\beta - M_\alpha \tag{1}$$

$$\log\left[P\left(\frac{CorrectDecision}{WrongDecision}\right)\right] = \log\left(\frac{1-P_{err}}{P_{err}}\right) = \Delta = M_\beta - M_\alpha \tag{2}$$

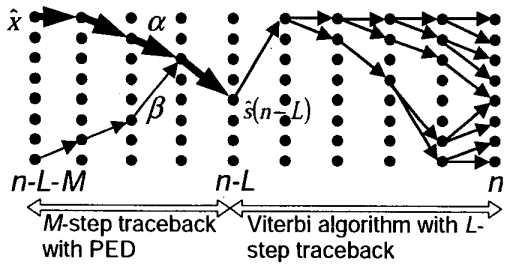

Figure 2. Two-stage traceback in a SOVA decoder to determine the two ML paths, α and β.

A second-stage traceback uses a path-equivalence detector (PED) to determine the equivalence between each pair of competing decisions obtained through a k-step traceback from nodes along the ML path, $k \in [1,2,\ldots,M]$. A reliability measure unit (RMU) is further employed to resolve the minimum difference in competing path metrics. The result is the next-ML path, β, which reflects the complementary bit decision, $\bar{\hat{x}}$.

3. ADD-COMPARE-SELECT STRCUTURES

The throughputs of SOVA decoders have traditionally been limited by the implementation of the add-compare-select (ACS) structure due to a single-step recursion that prevents pipelining. Previous high throughput implementations of the Viterbi decoder, [6], [8], resorted to unrolling of the ACS loop in order to achieve high throughputs. These methods increase the critical path delay, but improve the overall throughput.

A more effective architectural modification is to perform retiming and transformation of the ACS unit, [3] [4]. Figure 3 shows the transition trellis of an example 2-state Viterbi or SOVA decoder. Let $sm_i(n)$ represent the running metric for state i, and $bm_{ij}(n)$ be the branch metric of a corresponding transition from state i to state j. The time instance is denoted by n. The critical path of the traditional ACS computation outlined in (3) goes through the sequential execution of two additions, a comparison and a selection.

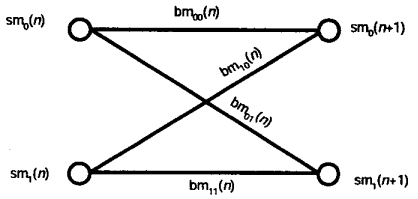

Figure 3. Transition trellis of a 2-state SOVA decoder.

ACS0:

$$sm_0(n+1) = \min\{sm_0(n)+bm_{00}(n) \;,\; sm_1(n)+bm_{10}(n)\}$$

ACS1: (3)

$$sm_0(n+1) = \min\{sm_0(n)+bm_{01}(n) \;,\; sm_1(n)+bm_{11}(n)\}$$

Without loss of generality, the operations in the ACS can be retimed, or delayed by a third of a cycle, as shown in Figure 4. The operations are reordered as: a comparison between the two sums, followed by selection of the appropriate minimum value, and addition with the corresponding branch metrics. The resulting structure is known as a Compare-Select-Add (CSA) unit.

The CSA structure is further transformed by moving the add operations before the select operation, as shown in Figure 5, resulting in the parallel execution of the compare and add operations. This modification decreases the critical path delay at the cost of doubling the number of adders and multiplexers.

An exploration was performed to compare the transformed CSA architecture with traditional Radix-2 and loop unrolled Radix-4 ACS architectures. The simulations were performed with high-threshold cells and low supply for evaluation of performance for low-speed, low-power applications, but the relative numbers will be applicable to the high-speed, low-threshold cells used in the SOVA decoder silicon implementation. Compared to the Radix-2 ACS, a CSA structure provides a speed improvement of 34% at the expense of a 22% increase in overall area. The Radix-4 ACS achieves a further throughput increase of 5%, but area is more than doubled. These results are listed in Table 1.

The heavy area penalty of the Radix-4 ACS is due to the increased interconnect of the higher radix. In addition, the remaining components of the SOVA decoder, namely the survivor path selection and reliability measure units, have to be adjusted for the doubled symbol rate. Hence, the CSA structure is an effective solution for high throughput SOVA decoder implementations. The adders in the transformed CSAs have flat input data-arrival profiles, and permit datapath synthesis to achieve the fastest adder implementation.

Table 1. Energy Efficiency of ACS & CSA Architectures

	Throughput @ 1.2V	Relative Area
Radix-2 ACS	82 Msym/s	1.00
CSA	116 Msym/s	1.22
Radix-4 ACS	122 Msym/s	2.70

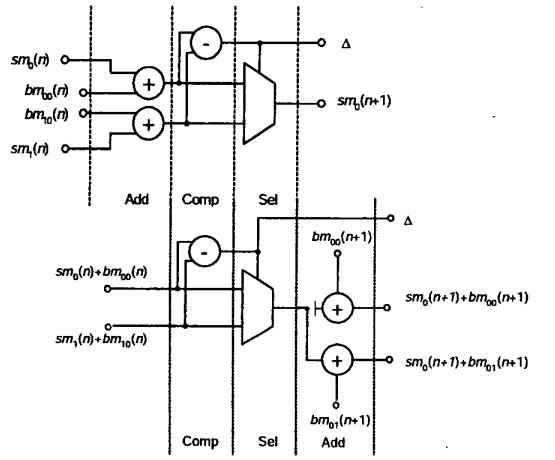

Figure 4. Retiming of ACS units to construct Compare-Select-Add (CSA) units.

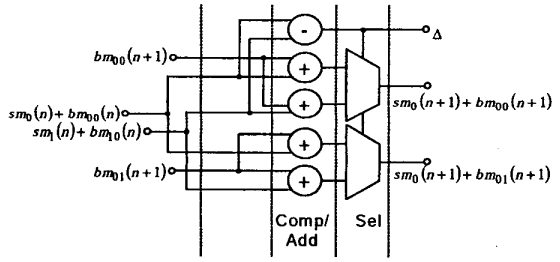

Figure 5. Transformation of CSA structure moves add operations before the select operation.

4. SURVIVOR PATH STRUCTURES

The two ML paths are determined by a dual-traceback function. A survivor memory unit (SMU) is cascaded with a combination of path-equivalence detector (PED) and reliability measure unit (RMU). The SMU and PED have similar functions. Both essentially examine a list of competing paths by retracing a history of decisions and path metric differences. Previous implementations of the SOVA decoder used either the register exchange method [5] or memory traceback [7] methods. These methods are reviewed and compared, followed by detailed description of the structure of the PED and RMU.

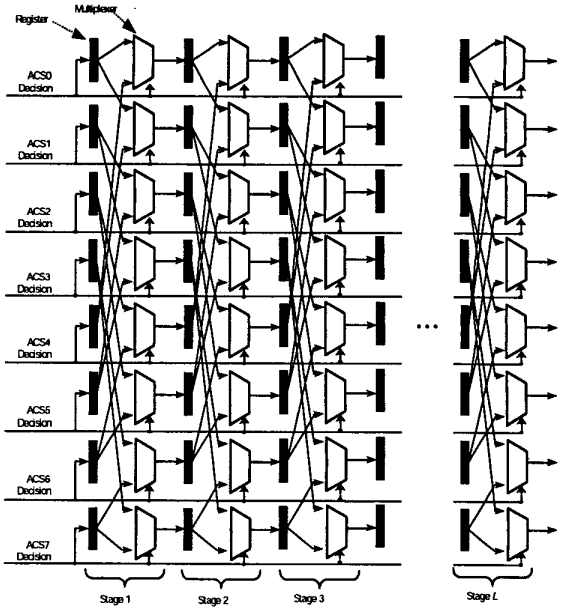

Figure 6. 8-state register exchange for SMU implementation.

4.1. Register exchange method

A register exchange consists of a two-dimensional array of 1-bit registers and multiplexers as shown in Figure 6. The connection between registers in successive stages mimics the trellis representation of the convolutional code.

The registers are controlled by a global clock signal (not shown), whose frequency is equal to the throughput of the Viterbi decoder. A binary path decision from each of the eight ACSs selects the outputs of a row of multiplexers. The decisions are also input to the register exchange pipeline.

At each clock cycle, a multiplexer located at row i and column k $\{i \in [1,2,\ldots,8],\ k \in [1,2,\ldots,L]\}$ outputs a bit decision corresponding to a traceback of length k, originating from state i. This bit is stored in a register and will be input to a multiplexer at column k+1 in the following clock cycle.

4.2. Memory traceback method

The memory traceback method simply writes a vector of path decisions from the ACS recursions into RAM at each iteration of the Viterbi algorithm. After an initial startup delay, the decisions are retraced by reading the stored decisions in a reversed direction. Due to the non-causality of this method, the memory traceback method faces imminent pipeline stalls.

In general, solutions employ some variation of the k-pointer trace-back architecture [10]. An example may make use of k-1 parallel read pointers that access as many independent banks of memories, while a write pointer simultaneously stores the decisions from the ACS recursions into a k^{th} memory bank. A recent implementation [7] uses a single bank of multi-ported DRAM in 1.2μm CMOS process. This design was particularly limited by the speed of the memory. The effective throughput rate of the decoder is eight times slower than the clock of the memory in order to permit multiple accesses in the traceback pipeline.

4.3. Comparison of survivor path methods

This section compares the area, power, and throughput implications of the two survivor path structures.

The memory traceback method permits the use of very compact SRAM that provides significant area advantage. The area of a typical 6T memory cell in 0.18μm technology is about $10\mu m^2$. Address decoders and word/bit line drivers may increase this area by another 50%, due to the small dimensions of the required memory. Nevertheless, it still compares favorably to the use of flip-flops in the register exchange method, since each flip-flop occupies about $50\mu m^2$.

The memory traceback method is also the choice structure for low power implementations. The bit decisions are stored in static memory locations, as opposed to being constantly moved through a pipeline of flip-flops, which is required by the register exchange method.

On the other hand, the register exchange is an appropriate structure for high throughput implementations. The critical path of a single stage in the register exchange architecture is easily less than 1ns; it consists only of a flip-flop (register) and a 2:1 multiplexer. Comparatively, a single-ported SRAM with less than 512 bits typically has a 5ns cycle period. The overall memory bandwidth is further reduced by additional I/O ports, which are required by the memory traceback architecture to address the pipeline stall problem.

As the emphasis of this work is on high throughput implementation, the register exchange method was selected for implementation of the survivor path structure.

5. PATH EQUIVALENCE DETECTOR (PED) AND RELIABILITY MEASURE UNIT (SMU)

Since the two data inputs to each multiplexer reflect the competing bit decisions, a test for their equivalence could be performed by addition of an XOR gate at each multiplexer location (Figure 7). The ensuing Boolean outputs, $EQ_{i,j}(n)$, indicate the equivalence between the two competing decisions obtained through a j-step traceback from state i.

From ACS_i, the difference between the two path metrics, $\Delta_i(n)$, arriving at time n, state i, is retained in FIFO buffers; $i \in \{1,2,...,8\}$. The output from the SMU selects $\Delta_i(n)$ and $EQ_{i,j}(n)$, which correspond to the values along the ML path, as inputs to the RMU.

The RMU consists of comparators and multiplexers in a pipeline that selects the minimum $\Delta(n)$ along the ML path. It is initialized with the maximum binary representation of the reliability measure, "111111", symbolized as "∞" in Figure 9. Based on the EQ input, each pipelined section outputs one of the following:

EQ = 1: Reliability measure from the previous step

EQ = 0: Min{Δ_i, previous reliability measure}

Compared with a Viterbi decoder implementation, the total size of the SMU and PED is approximately doubled ($L = M$). The RMU overhead consists of M copies of 1 register, 2 multiplexers and a 2-input comparator. The latency through the SOVA decoder is $L + M$. The additional latency remains insignificant compared to the overall latency in the Turbo-SOVA system, which is dominated by the latency through the interleavers.

6. PHYSICAL DESIGN FLOW AND TESTING

Using an automated design flow for direct mapping of signal processing algorithms into integrated circuits [8], two SOVA decoders were implemented in silicon. The automated flow was further enhanced for high-speed design through customization of the clock tree to achieve low clock skews.

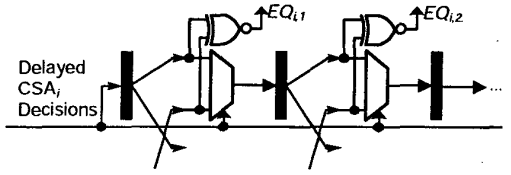

Figure 7. State-slice of register exchange used in the PED

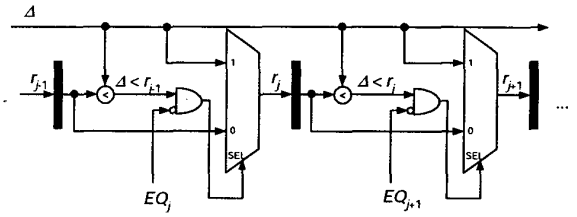

Figure 8. Pipelined section of reliability measure unit (RMU).

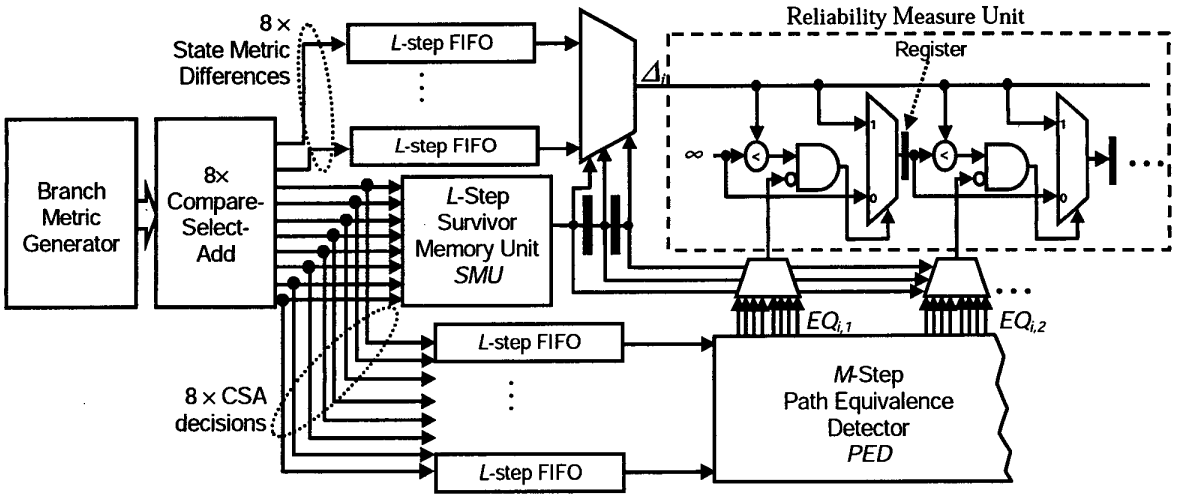

Figure 9. System architecture of 8-state SOVA decoder.

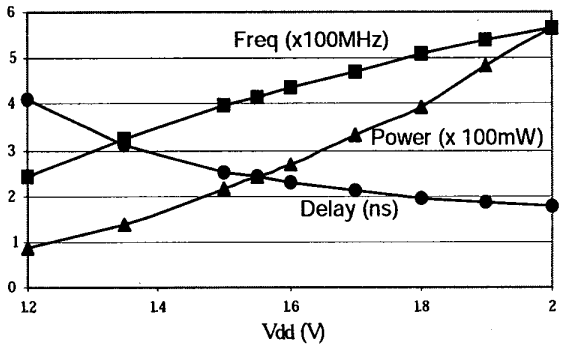

Figure 10. Performance of SOVA_EPR4 decoder.

Both decoders have the same architecture, but are matched to different generator polynomials; the *SOVA_EPR4* decoder is matched to an Enhanced Partial Response Class 4 (EPR4) channel with a $(1\oplus D)^{-1}$ precoder while the *SOVA_13* decoder is matched to an Octal(13) generator. The equivalent generator polynomials are $\frac{1}{1\oplus D}(1+D-D^2-D^3)$ and $1\oplus D^2\oplus D^3$ respectively. The system architecture of this implementation is shown in Figure 9. The high throughput requirement is provided by the realization of the CSA and register exchange structures as described in Section 3 and 4 respectively. The branch metric generator, compare-select-add (CSA), and L-step survivor memory unit (SMU) form the building blocks of a conventional Viterbi decoder. There are eight copies of the CSA in accordance to the targeted eight-state convolutional codes. The register exchange implements traceback with $L = M = 15$, which is five times the constraint length of the convolutional code.

The required wordlength of each SOVA decoder was ascertained by comparing the performance difference between floating-point computation and several fixed-point types. 7-bit sign-magnitude signals were necessary to provide less than 0.1dB degradation of required signal-to-noise ratio at a bit-error rate of 10^{-5}. The number of Turbo-iterations simulated was restricted to five [2], in line with feasibility constraints of a high throughput pipelined decoder system implementation,.

This design identifies the ACS recursion as the throughput bottleneck of the decoder design, and implements retiming and transformation for a shorter critical path. This places more stringent requirements on the SMU and PED realizations, which are met only through the use of register exchange structures similar to [11]. This work also achieves two orders of magnitude advantage over [7] in terms of throughput rate, at the cost of an order of magnitude difference in power dissipation. As mentioned, the latter is due to the continuous movement of data through rows of shift registers and FIFO buffers, as well as the increased parallel CSA activities.

The functionality of the chip (Figure 11) has been verified with 1.8V supply at 25°C. Throughput rates above 500 Mb/s were achieved and power dissipation was 400mW. The speed characterization was performed using a clock tree with a built-in delay line. The speed and power performance for one of the SOVA decoders is plotted in Figure 10. The power measurements were performed at the highest frequencies permitted by the supply voltage. Table 3 summarizes the characteristics of the decoders.

Table 3. Summary of results

Decoder Type	SOVA_EPR4	SOVA_13
Number of States	8	8
Transistor Count	164K	174K
Core Area	1mm x 0.5mm	1mm x 0.5mm
Speed	500Mb/s	500Mb/s
Avg. Power (400MHz with random inputs)	395mW	400mW

Figure 11. Die micrograph.

7. CONCLUSION

The design of 500MHz soft-output Viterbi decoders has been described. The 4mm^2 chip consists of two SOVA decoders matched to different generator polynomials. They can be employed as a soft-input-soft-output (SISO) decoders for Turbo code systems. This implementation extends the register exchange method of a Viterbi decoder with a path equivalence detector (PED) and a reliability measure unit (RMU) in order to determine the next-most-likely path, and the difference in path metric between the two most-likely paths. The use of deeply-pipelined, datapath-intensive structures permit high throughput without the requirement for specialized RAM blocks with complex multiple pointer mechanisms or multiple read/write ports.

Architectural retiming and transformation of the add-compare-select structures with modification of the register exchange allow a high throughput with small area overhead, compared to previous loop-unrolling methods. In addition to magnetic recording applications, the SOVA decoder is also appropriate for Turbo-coded forward error correction applications in wireless, wireline, and optical communication systems.

8. ACKNOWLEDGEMENTS

The authors thank T. Smilkstein, P. Pakzad and V. Anantharam of UC Berkeley for their technical assistance, and ST Microelectronics for fabrication of the test chip. Texas Instruments supported this work under the UC MICRO program.

9. REFERENCES

[1] J. Hagenauer and P. Hoecher, "A Viterbi algorithm with soft-decision outputs and its applications," *Proc. IEEE GLOBECOM*, pp. 47.11-47.17, Dallas, TX, Nov 1989.

[2] E. Yeo, P. Pakzad, B. Nikolic, and V. Anantharam, "VLSI architectures for iterative decoders in magnetic recording channels," *IEEE Trans. Magnetics*, vol.37, no.2, pp.748-55, Mar. 2001.

[3] G Fettweis, R. Karabed, P.H. Siegel, and H.K. Thapar, "Reduced-complexity Viterbi detector architectures for partial response signaling," *Proc. IEEE GLOBECOM*, pp.559-63, Singapore, Nov 1995.

[4] I. Lee and J. L. Sonntag, "A new architecture for the fast Viterbi algorithm," *Proc. IEEE GLOBECOM*, pp.1664-8, San Francisco, CA, Nov. 2000.

[5] O. J. Joeressen and H. Meyr, "A 40 Mb/s soft-output Viterbi decoder," *IEEE Journ. Of Solid-States Circuits*, vol.30, no.7, pp.812-18, Jul. 1995.

[6] Yeung, and J. M. Rabaey, "A 210 Mb/s radix-4 bit-level pipelined Viterbi decoder," *Proc. IEEE ISSCC*, San Francisco, CA, USA, pp.88-9, 344, 440, Feb 1995.

[7] D. Garrett and M. Stan, "Low power architecture of the soft-output Viterbi algorithm", *Proc. IEEE ISLPED*, Monterey, CA, pp.262-7, Aug. 1998.

[8] P. Black and T. Meng, "A 1-Gb/s, four-state, sliding block Viterbi decoder," *IEEE Journ. Of Solid-States Circuits*, vol.32, no.6, pp.797-805, Jun 1997.

[9] W. R. Davis, N. Zhang, K. Camera, D. Markovic, T. Smilkstein, M. J. Ammer, E. Yeo, S. Augsburger, B. Nikolic, and R. W. Brodersen, "An Automated Design Flow for High-Throughput Low-Power Dedicated Signal Processing Systems," *IEEE Journ. Of Solid-States Circuits*, vol. 37, no. 3, pp.420-431, Mar 2002.

[10] G. Feygin and P. Gulak, "Architectural tradeoffs for survivor sequence memory management in Viterbi decoders," *IEEE Trans. on Comms*, vol.41, (no.3), pp.425-9, Mar 1993.

[11] O. Joeressen, M. Vaupel, and H. Meyr, "High-speed VLSI architectures for soft-output Viterbi decoding", *Proc. IEEE ICASAP*, Berkeley, CA, pp.373-84, Aug 1992.