Journal of Signal Processing
Volume 2 Issue 3

An Approach for Effective Design Space Exploration of Hard-Decision Viterbi Decoder: Algorithm and VLSI Implementation

T. Usha Rani, S. Iswariya
St. Peters Engineering College, India
E-mail: usharani@stpetershyd.com

Abstract

Viterbi algorithmic rule is usually used as a cryptography technique for convolutional codes, bit detection technique, Trellis in storage devices. The design space for VLSI implementation of Viterbi decoders is massive, involving selections of turnout, latency, area and power. Even for a set of parameters like constraint length, encoder polynomials and trace-back depth, the task of de-signing a Viterbi decoder is kind of troublesome and needs important effort. Sometimes, as a result of incomplete style area exploration or incorrect analysis, a suboptimal style is chosen. This work analyzes the planning complexness by applying most of the identified VLSI implementation techniques for hard-decision Viterbi cryptography to a distinct set of code parameters. The conclusions square measure supported real styles that actual synthesis and layouts were obtained. In authors' read, as a result of the depth lined, it is the foremost comprehensive analysis of the subject revealed to this point.

Keywords: *Viterbi algorithm, VLSI design, survey*

INTRODUCTION

Viterbi decoder is one among the foremost wide used components in digital communications and storage devices. Although, its VLSI implementation is studied full over the last decades, still each new style starts with style area exploration. This could be partly explained by the very fact that the planning area is giant. Additionally, the improvement criteria and also the style figures continue dynamical with the advancement in CMOS technology and style tools. Totally different style aspects of the Viterbi decoder are studied in an exceedingly variety of analysis papers. However, most researchers consider one specific part of the planning (e.g., branch unit of measurement, add compare choose unit, path metrics unit or survival memory unit). A scientific and comprehensive analysis summarizing and characterizing as several of

the trade-offs and implementation techniques as attainable is missing. This contribution presents such a survey, providing designers with clear guide-lines and references to search out the simplest answer for each specific case.

The scope of the paper is proscribed to the "classical" Viterbi algorithmic rule. Trellis decryption techniques like MAP-decoding, T- and M- algorithms don't seem to be mentioned well. Additionally, almost all of the studied styles optimizations, though established by synthesis for a particular target CMOS technology, square measure technology freelance, therefore, the conclusions ought to stay valid for consecutive two to three solid-state device generations. For this reason, semiconductor unit and interconnect level enhancements do not seem to be thought of during this paper. We have a tendency to considering T-Algorithm has planned in 2 variations, the relaxed adaptive VD, that suggests mistreatment a calculable optimum path metric, rather than finding the $64000 one every cycle and, therefore, the limited-search parallel state VD supported scarce state transition (SST). Once applied to high rate convolutional codes, the relaxed adaptive VD suffers a severe degradation of bit-error-rate (BER) performance because of the inherent drifting error between the calculable optimum path metric and therefore the correct one.

We assume that the reader is familiar with convolutional codes and basic implementation techniques of the Viterbi algorithm, so we limit the discussion of these subjects in the following sections to a minimum. A good summary of references to both the topics can be found. All area and timing results presented in this paper are based on the standard cell 90-nm CMOS technology.

GENERAL STRUCTURE OF THE VITERBI ALGORITHM

Viterbi decoding rule is that the preferred methodology to de-code convolutional error correcting codes. In a very convolutional encoder, an input bitstream is versed a register. Input bits square measure combined exploitation the binary single bit addition (XOR) with many outputs of the register cells. Ensuing output bitstreams represent the encoded input bitstream. Generally speaking, each input bit is encoded exploitation output bits, therefore, the committal to writing rate is outlined as (or if input bits square measure used). The constraint length of the code is outlined because the length of the register and one. Finally, generator polynomials outline, that bits within the input stream have to be compelled to be

supplemental to make the output. An encoder is totally represented by polynomials of degree or less. Figure 1 shows an example of a straightforward convolutional encoder at the side of the corresponding parameters.

As can be easily recognized, a convolutional encoder forms a finite-state machine (FSM), whose state is described by the contents of the shift register. Every new input bit processed by the encoder leads to a state transition. One widely used presentation form of these state transitions is the so-called trellis diagram. An example of such a diagram is shown in Figure 2. Note that for the state encoding, B_{t-1} represents the least significant bit and B_{t-2} the most significant bit respectively. A transition in the trellis diagram is called a branch. For a binary convolutional code, every state in the trellis has two incoming branches representing the transmission of one and zero, respectively

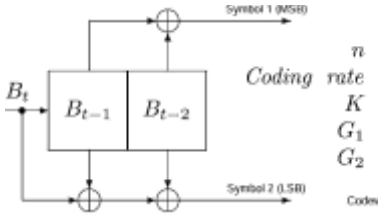

Fig. 1: *Convolutional Encoder (Index Denotes Timing in Clock Cycles).*

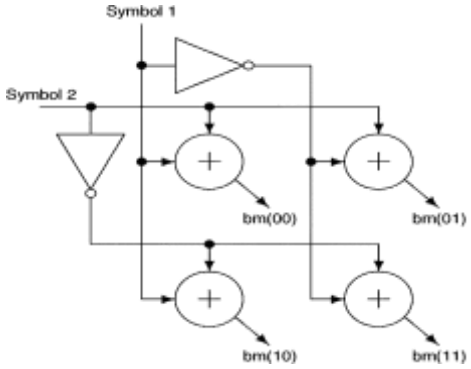

Fig. 2: *Trellis Diagram.*

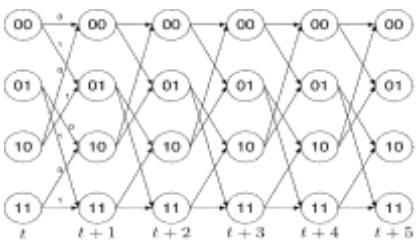

Fig. 3: *Radix-2 BMU.*

Every state transition within the encoder leads to one codeword being made. When the transmission over a loud channel, the code words might get corrupted. Viterbi decoder reconstructs the initial input sequence of the encoder by hard the foremost probable sequence of the state transitions. This can be done by tracing the trellis in a very reverse manner whereas observing the sequence of incoming code words. Each codeword is made as results of a mix of a precise input bit with specific encoder state. This means, that the probability of the state transitions are often calculated notwithstanding received code words square measure corrupted. By scrutiny the probability values, some transitions are often erased like a shot, thereby removing unwanted house. Once the state transition sequence is set, the reconstruction of the transmitted bit sequence is trivial.

At the highest level, Viterbi Decoder consists of 3 units: the branch metric (BMU), the trail metric (PMU), and also the survivor memory unit (SMU). The BMU calculates the distances from the received (noisy) symbols to all or any legal codewords. Just in case of the encoder diagrammatical in Figure 1 the sole legal code-words are "00", "01", "10", and "11". The live calculated by the BMU is the overacting distance of the laborious input decryption or the Manhattan/Euclidean distance in case of the soft input decryption (e.g., each incoming image is represented exploitation many bits). The PMU accumulates the distances of the one codeword metrics created by the BMU for each state. Below the idea that zero or one was transmitted, corresponding branch metrics are accessorial to the antecedently hold on path metrics that are initialized with zero values. The ensuing worths are compared with one another and also the smaller value is chosen and hold on because the new path metric for every state. In parallel, the corresponding bit decision (zero or one) is transferred to the SMU while the inverse decision is discarded. The structure of the add-compare-select circuit which performs the operations described above will be discussed in detail later.

Finally, the SMU stores the bit decisions produced by the PMU for a certain defined number of clock cycles (referred as traceback depth, TBD) and processes them in a reverse manner called backtracking. Starting from a random state, all state transitions in the trellis will merge to the same state after TBD (or less) clock cycles. From this point on, the decoded output sequence can be reconstructed. Coding rate $1/n$, constraint length K, traceback depth TBD, and the number of bits representing each input value (referred as softbits or input bit width) are the key parameters influencing the performance and design of the Viterbi decoder. In the

following sections, the affects of changing these parameters are discussed for every decoder unit separately.

BRANCH METRIC UNIT (BMU)

Design Space

BMU is usually the smallest unit of Viterbi decoder. Its complexness will increase exponentially with n (reciprocal of the writing rate) and additionally with the quantity of samples processed by decoder per clock cycle (radix issue, e.g., radix-2 corresponds to 1 sample per clock cycle). The complexness will increase linearly with softbits. So, the realm and outturn of the BMU is fully delineate by these 2 factors. As BMU isn't the crucial block in terms of space or outturn, its style appearance quite easy. The version hard the playing distance for writing rate 1/2 bestowed in Figure 3 performs utterly in terms of each space and outturn. A BMU hard square euclidian or Manhattan distance is slightly additional advanced however is simply mapped to an array of adders and subtracters likewise.

Table 1: *Design Space.*

value	meaning	
000	strongest	0
001	relatively strong	0
010	relatively weak	0
011	weakest	0
100	weakest	1
101	relatively weak	1
110	relatively strong	1
111	strongest	1

Branch Metric Precision

BM should be able to hold the maximum possible difference (distance) between ideal and received symbols. For hard input (softbits $=1$) , the Hamming distance between a coded word and its complement is the maximum possible BM (i.e., the distance between codewords consisting of all 0's and all 1's). In such case, the maximum distance is equal to the symbol length n. So, BM precision is given by:

(1)

$$\mathrm{BM}_{\mathrm{precision}} = \lfloor \log_2 n \rfloor + 1.$$

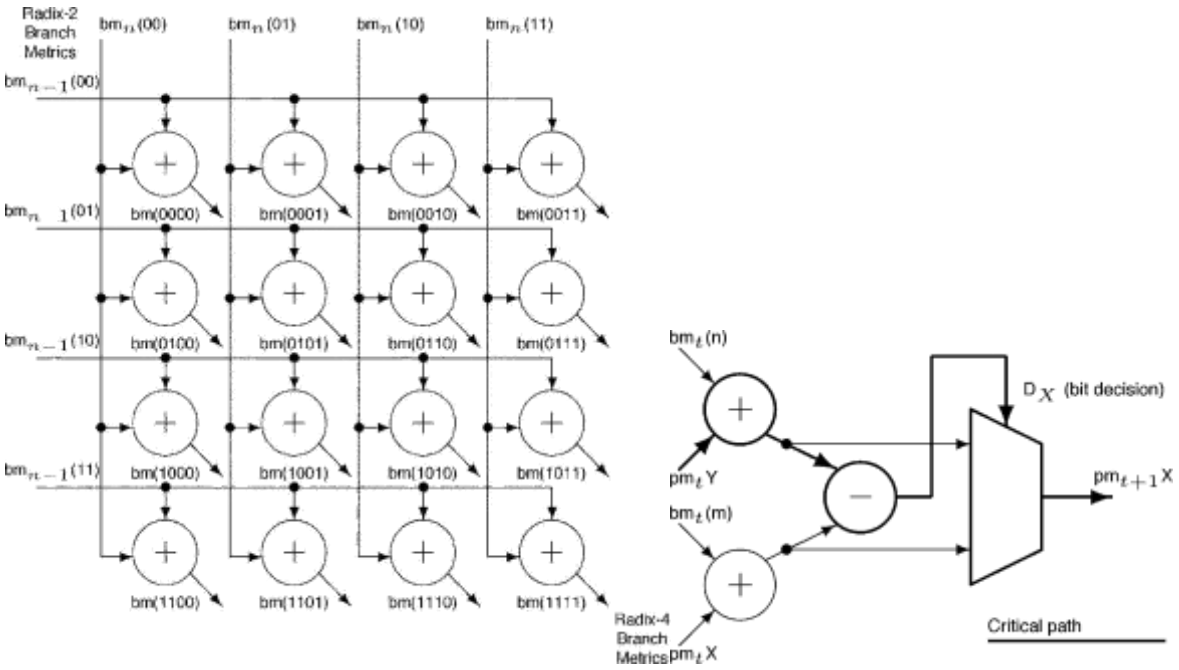

Fig. 5: *Radix-2 ACS.* ***Fig. 4:*** *Radix-4 BMU.*

In general, radix-8 Viterbi decoder designs are very uncommon due to their overall complexity. For radix factors beyond 2, combined with soft input de-coding, BMU can also be implemented as a set of lookup tables or cascaded combination of adders and lookup tables. Depending on the cell library and the technology used, such implementations may occupy less silicon area than the straight forward approach. Still, taking into account the moderate area of the BMU, these area savings are almost invisible at the top level. In soft input decoding, each received symbol is represented by more than one bit (softbits >1) . In this case, the maximum possible branch metric is calculated as distance

(2)
$$\mathrm{BM_{BW}} = \lfloor \log_2(2^{\mathrm{soft\ bits}-1}) \times n \rfloor + 1.$$

Radix-2 BMU

The hardware cost of BMU from Figure 3 in terms of areas of basic components can be expressed as follows:

(3)
$$\begin{aligned}\mathrm{BMU2_{area}} = 2^n \times (&\mathrm{HA_{area}} \\ &+ \mathrm{FA_{area}} \times (\mathrm{softbits} - 1)) \\ &+ \mathrm{softbits} \times n \times \mathrm{INV_{area}}\end{aligned}$$

where $HA_{area}FA_{area}INV_{area}$, denote the area of a half adder, full adder, and inverter, respectively.

Radix-4 BMU

An implementation of radix-4 BMU (processing two code-words per clock cycle) is shown in Figure 4. The underlying radix-2 BMs are added to form radix-4 BMs [11]. For Radix-4 BMU, the area cost is expressed as follows:

$$\begin{aligned}&\mathrm{BMU4}_{\mathrm{area}}\\&\quad=2^{2\times n}\times(\mathrm{HA}_{\mathrm{area}}+\mathrm{FA}_{\mathrm{area}}\times\mathrm{softbits})+2\\&\qquad\times2^{n}\times(\mathrm{HA}_{\mathrm{area}}+\mathrm{FA}_{\mathrm{area}}\times(\mathrm{softbits}-1)\\&\qquad+n\times\mathrm{softbits}\times\mathrm{INV}_{\mathrm{area}}).\end{aligned}\tag{4}$$

BMs are generated assuming that every possible combination of n bits is a valid encoder output. But as value n beyond 4 is seldom used and radix is limited to factor 4, the area of the BMU is negligible compared to other units. For radix-8 designs, BMU will require considerably more area and also become slower.

PATH METRIC UNIT (PMU)

Design Space

PMU may be a crucial block each in terms of space and output. The key downside of the PMU style is that the algorithmic nature of the add-compare-select (ACS) operation (path metrics calculated within the previous clock cycle area unit employed in the present clock cycle as Figure 5 shows). Improvement techniques of PMU in Viterbi decoder implementation area unit shown in Figure 6. So, as to extend the output or to scale back the realm, optimizations is introduced at recursive, word or bit level. To get a awfully high output, correspondence at recursive level is exploited. By recursive transformations, the Viterbi decipherment is regenerate to a strictly feed forward computation. This enables freelance process of input blocks. The recursive parallel block process strategies shall come through unlimited concurrency by freelance block decipherment of input stream. These techniques lead to quite high space figures. However, as technological advancements area unit creating the devices shrink, they're obtaining additional engaging. Still, for a selected case, if needed output is achieved by utilizing word or bit level improvement techniques, there is no specific ought to use recursive transformations.

Word level optimizations work on folding (serialization) or unfolding (parallelization) the ACS recursion loop. In the folding technique, the same ACS is shared among a certain set of states. This technique trades off throughput for area. This is an area efficient approach for low throughput decoders, though in case of folding, routing of the PMs becomes quite complex. With unfolding, two or more trellis stages are processed in a single recursion (this is called look ahead). If look ahead is short then area penalty is not high. Radix-4 lookahead (i.e., processing two bits at a time) is a commonly used technique to increase decoder's throughput.

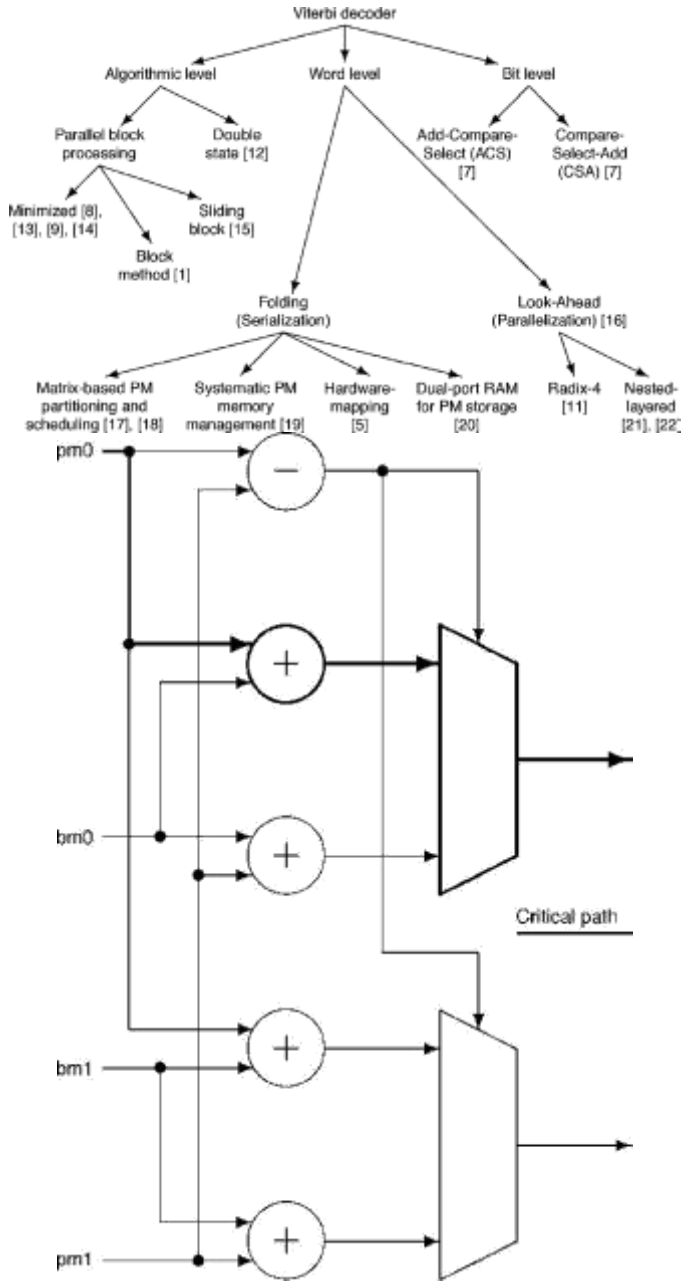

Fig. 6: *Design Space for the VLSI Implementation of the Viterbi Algorithm.*

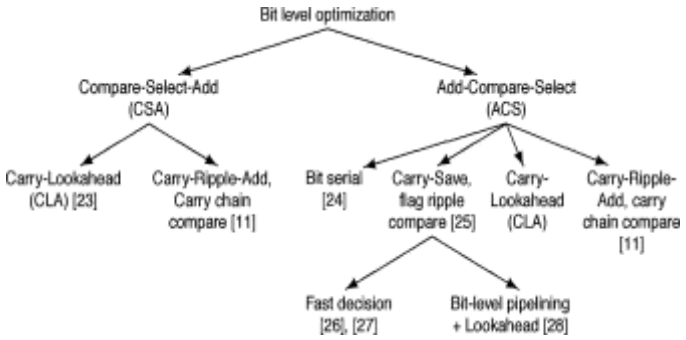

Fig. 7: *Bit-Level Optimization Techniques.*

Bit level optimizations can be used to speed up the ACS operation itself. A conventional fully parallel implementation of the Viterbi These techniques differ from word level techniques by having a decoder implies using one radix-2 ACS per state of the trellis, lower area penalty. Fig. 7 shows the various bit level optimiza processing one stage of trellis at a time. For decoders with very tion techniques. Compare-select-add (CSA) is a retimed high throughput or very low area constraints, some optimization of the ACS operation as shown in Figure 8. It computes all techniques at system level appear attractive. These techniquespossible paths originating from a node and based on decision it achieve area figures lower or throughput figures higher, than retains the minimum PM path. Thus, addition and comparison the conventional fully parallel implementation of the algorithm can be processed in parallel. Further, addition and comparison operation can be imple-
mented using different arithmetics. Carry-look-ahead (CLA) technique [23] uses carry look ahead adders for addition and comparison (subtraction) operation. Ripple-add carry-chain compare exploits bit level parallelism between addition and comparison operations. In this technique, subtraction is used. In this paper only bit level optimizations are discussed in for comparison. Both addition and subtraction operations start depth, since analysis of the algorithmic and word level tech with the least significant bit (LSB). Thus, as soon as LSB of the niques concerning their effects on area and throughput is rather sum is computed, it can be used for comparison (see Figure 9). straightforward. In the carry save approach, instead of propagating the carry from LSB to the most significant bit (MSB), carry generated B. Path Metric Precision during each bit addition is saved. The comparison starts with The register temporarily storing path metrics should be wide MSB, taking into account the saved carry bits. This makes the enough to avoid overflow errors in the PMU operations. Modulo comparison operation quite complex. The advantage, however, arithmetic is usually used for

this purpose [29]. PM bitwidth is that the addition and comparison operations can be pipelined determined as follows: at bit level to reduce the critical path. Due to the complex compare operation there is a significant area penalty. In the bit serial approach the addition and comparison operations are carried out in serial manner. Where is given as (6)

K M
P

$$\mathrm{PM_{BW}} = \lceil \log_2 B + \log_2(2 \times 2 \times (K-1)) \rceil$$

B

N N

$$B = (2^{\mathrm{softbits}} - 1) \times n.$$

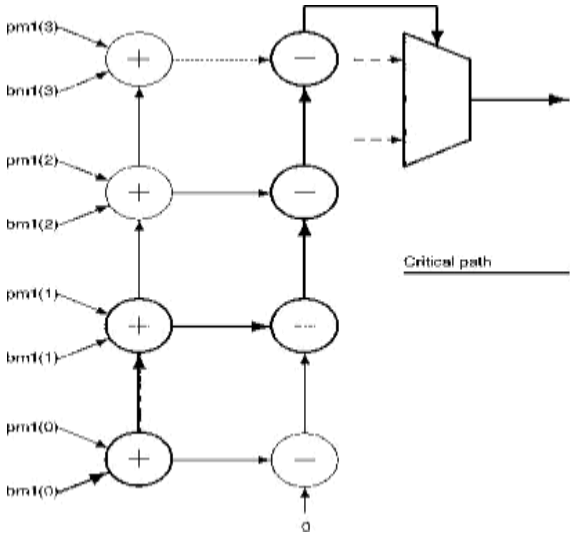

Fig. 9: *Parallelism between Addition and Comparison Operations in Ripple Adder Configuration (pm1(x) and bm1(x) Denotes the th bit in* the Numerical Representation of the Corresponding Metric, Respectively).

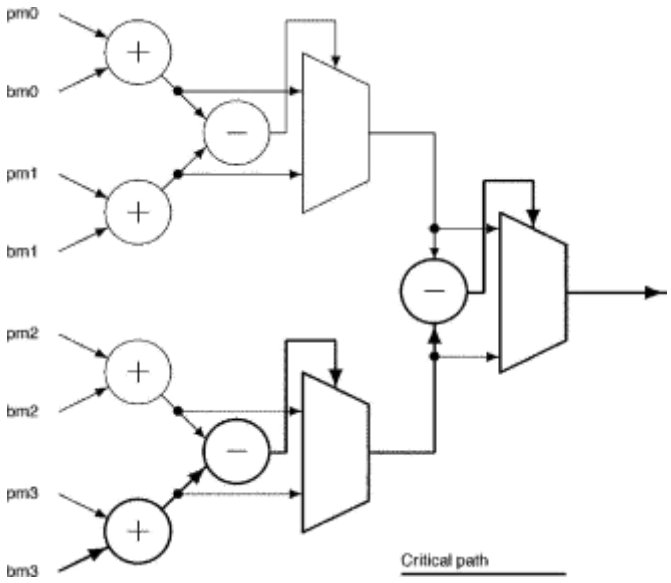

Fig. 10: *Radix-4 ACS with Hierarchical Comparisons.*

to the synthesis tool) or by instantiating specific arithmetic circuits manually such as ripple carry or carry look ahead (CLA) adders and subtracters.

As PMU is a fairly regular unit where a basic module (ACS) is instantiated several times (e.g., 2^{K-1} in case of the fully parallel approach), it is especially important to optimize this module. For N bit addition, the delay of CLA implemented in a logarithmic configuration [23] can be expressed as

:

Table 1: *Throughput and Complexity Comparison of Algorithmic and Word Level Optimization Techniques.*

Technique	Through-put	Area complexity
Sliding Block [15]	M	$M + 2 \times TBD$
Nested Lookahead [22]	M	$\frac{M}{K} \times \sum_{i=2}^{K} 2^i + (2^{K-1})^2 \times \left(\frac{M}{K} - 1\right) + 2^{K-1} \times \frac{M}{K} + 2^{K-1} \times \left(\frac{M}{K} - 1\right)$
Folding [17], [18]	$\frac{P}{2^{K-1}}$	$\approx \frac{P}{2^{K-1}}$(of PMU)
Radix-4 [28]	1.7	≈ 2 (of PMU)

(7)
$$\mathrm{CLA}_{\mathrm{delay}} = T_{\mathrm{XOR}} + (T_{\mathrm{AND}} + T_{\mathrm{OR}}) \times \log_2 N + T_{\mathrm{FA}}$$

where delays of logic components such as XOR, AND, OR, and full adder are denoted as $T_{\mathrm{XOR}} T_{\mathrm{AND}} T_{\mathrm{OR}}$, and T_{FA} respec-tively. Similarly, delay through ripple carry adder can be ex-pressed as a function of half-adder and full adder delays

(8)
$$\mathrm{RCA}_{\mathrm{delay}} = T_{\mathrm{HA}} + (N - 1) \times T_{\mathrm{FA}}.$$

As indicated in [23] the efficiency of CLA is better than ripple only for large values of N, e.g., N>8.

Figure 5 highlights the critical path in radix-2 ACS. When implemented in CLA configuration, each circle represents an addition/subtraction with precision PMbw. Assuming no par-allelism between addition and comparison operations, critical path delay can be expressed as

Choice Among Various PMU Implementations

As mentioned before, there are several options to implement the ACS operation. Radix-2 ACS (see Figure 5) is a basic design with lowest area and throughput figures. Other techniques such as CSA (see Fig. 8) and Radix-4 (see Figure 10) provide a higher throughput than radix-2 ACS at the cost of higher area as will be discussed in this section later. Note that the architecture of the adders and subtracters for the ACS implementation is an orthogonal design choice. More specifically, the addition and subtraction operations can be described either in a behavioral fashion (i.e., giving the freedom to choose a particular implementation

(9)
$$\times \log_2 \mathrm{PM_{BW}} + T_{\mathrm{FA}}) + T_{\mathrm{MUX}}.$$

For ACS based on ripple-carry adders, critical path delay can be expressed as

(10)
$$\mathrm{RCA\_ACS_{delay}} = (\mathrm{PM_{BW}} + 1) \times T_{\mathrm{FA}} + T_{\mathrm{MUX}}.$$

With CLA implementation, we can observe a logarithmic in-crease in critical path delay with path metric precision rather than a linear one as in ripple implementation. As mentione earlier, by using ripple configuration in radix-2 ACS, bit level parallelism between addition and comparison can be exploited (see Figure 9). Therefore, the delay of addition and comparison op-erations is just one full adder delay more than the PMbw bits ripple carry addition. While using CLA for addition operation in radix-2 ACS the parallelism between addition and comparison operations cannot be fully exploited. Thus, addition and com-parison operations require nearly twice the delay of a PMbw bits CLA addition. Therefore, using CLA for addition in radix-2 ACS for low PMbw values will not improve the critical path delay significantly. The advantage of using CLA becomes ap-parent when the number of precision bits increases. The critical
path delay for CLA is $\approx\lceil 2\times\log_2 \mathrm{PM_{BW}}\rceil$ while for ripple
carry, it is $\approx \mathrm{PM_{BW}}$.

Figure 8 highlights the critical path of a CSA implementation. If the adders of Figure 8 are implemented using the CLA technique, delay can be expressed as:

(11)
$$\begin{aligned}\mathrm{CLA\_CSA_{delay}} &= \mathrm{CLA_{delay}} + T_{\mathrm{MUX}} \\ &= (T_{\mathrm{XOR}} + (T_{\mathrm{AND}} + T_{\mathrm{OR}}) \\ &\quad \times \log_2 \mathrm{PM_{BW}} + T_{\mathrm{FA}}) + T_{\mathrm{MUX}}.\end{aligned}$$

While in case of a ripple carry implementation for the same adders, the delay can be expressed as:

(12)
$$\mathrm{RCA\_CSA_{delay}} = \mathrm{PM_{BW}} \times T_{\mathrm{FA}} + T_{\mathrm{MUX}}.$$

In case of CSA, the critical path delay using ripple carry adders is $\approx \mathrm{PM_{BW}}$, while using CLA, it is $\approx \log_2 \mathrm{PM_{BW}}$. Here, the advantage of using CLA becomes apparent. Thus, for higher values of PMbw, CSA using CLA adders and subtracters is a better design choice to achieve high

throughput.

Figure 10 highlights the critical path through a radix-4 ACS. It can be observed that hierarchical comparisons cause major delay in this design. Alternative version compares each sum pair-wise and then encodes the results of the comparison, as shown in Figure 11. Depending on the PMbw value, the critical path through the radix-4 ACS becomes shorter in this case. En-coder and MUX can also be combined to form an integrated selection unit. The delay of Radix-4 ACS while using CLA for addition and comparison operation can be expressed as:

$$\begin{aligned}\mathrm{CLA\_}R4_{\mathrm{delay}} &= 2 \times \mathrm{CLA}_{\mathrm{delay}} + T_{\mathrm{encoder}} + T_{\mathrm{MUX}} \\ &= 2 \times (T_{\mathrm{XOR}} + (T_{\mathrm{AND}} + T_{\mathrm{OR}}) \\ &\quad \times \log_2 \mathrm{PM}_{\mathrm{BW}} + T_{\mathrm{FA}}) \\ &\quad + T_{\mathrm{encoder}} + T_{\mathrm{MUX}}.\end{aligned} \tag{13}$$

The delay of Radix-4 ACS using ripple carry technique for addition operation can be expressed as

(14)

$$\mathrm{RCA\_}R4_{\mathrm{delay}} = (\mathrm{PM}_{\mathrm{BW}} + 1)T_{\mathrm{FA}} + T_{\mathrm{encoder}} + T_{\mathrm{MUX}}.$$

As radix-4 ACS is larger than radix-2 ACS or CSA, a variety of configurations are possible in its implementation. Area and throughput requirements determine the choice of a certain configuration. For radix-4, the critical path delay is ≈2×log2PMbw for CLA and ≈PMbw for ripple carry adder. As compared to the delay of radix-2 ACS, there is an overhead of an encoder. For the cascade comparison as shown

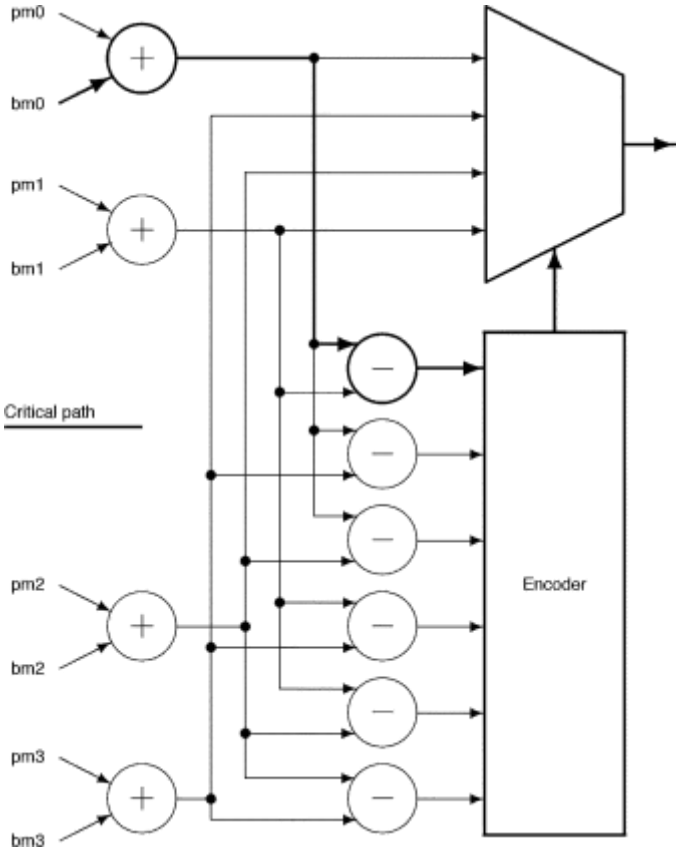

Fig. 11: *Radix-4 ACS using Parallel Comparisons.*

in Figure 10 the delay will be ≈3xlo$_2$PMbw. For low values of K and softbits, PMbw is low. Using CLA for lower number of bits does not provide any throughput benefit. Processing Re-write Suggestions Done (Unique Article) To compare the styles in terms of space and logic delay and to see the best implementation of every style, radix-2 ACS and CSA still as radix-4 ACS units were enforced and synthesized mistreatment the higher than mentioned techniques. Different implementations of the styles were synthesized with a relaxed temporal order constraint to be ready to see the result of the architectural decisions. Figures 12 and 13 show the plots of space and demanding path delay of various PMUs as obtained from synthesis experiments. For sure, the standard radix-2 ACS could be a better option in terms of space, on condition that the turnout requirement is met. As an example, the gain in space for the CLA implementation will be up to 50%.

Advantage of the CLA becomes visible just for higher values. As an example, mistreatment carry-lookahead within the CSA technique, achieves sixty eight higher turnout at a value of sixty two higher space compared to a radix-2 ACS with ripple carry adders. It is mentioned

that CSA can do double the maximum amount turnout as ACS [7]. However, from Figure 13, it is clear that CSA does not have an essential path [*fr1] that of ACS in any implementation. This distinction may be explained by the very fact that the info return from the prelayout analysis for one ACS/CSA operation and glued parameter set, whereby Figure 13is predicated on post-layout analysis of the whole PMU for several totally different parameter values in [7]. Contrary to the widespread opinion, we conclude that the throughput advantage of CSA over ACS is much less than factor 2 compared to almost doubled area

cost.

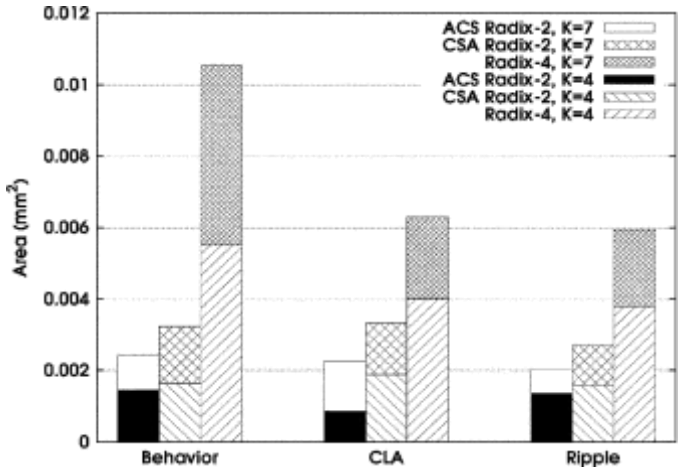

Table 2: *Area and Timing Comparison of Bit-Serial and Behavioral Description of ACS.*

	$K = 4$		$K = 9$	
	BS	Behavior	BS	Behavior
Area (mm^2)	0.006	0.007	0.248	0.448
Critical Path Delay (ns)	1.92	1.36	1.92	2.21
Throughput (Mbps)	57.87	735	57.87	452

Fig. 12: *Area Figures for Various ACS Implementation Techniques (Works with , and works with , and).*

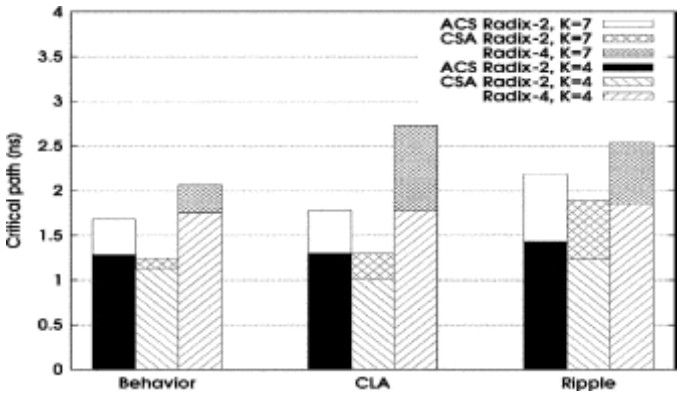

Fig.13: *Critical Path Delay for Various ACS Implementation Techniques (Works with ,*

and works with

, and).

Behavioral Versus Structural Synthesis

A normally united issue in VLSI style is that the correct alternative of lipoprotein description and committal to writing vogue. It is been ascertained that activity description for combinative logic and its structural description manufacture completely different synthesis results. Structural description of combinative logic was particularly common at the days once synthesis tools were not intelligent enough. Now a-days, the tools are becoming a lot of economical, so activity description has become an appropriate possibility. This can be quite evident from the graphs in Figures 12 and 13. As shown within the Figures, the activity description of ACS operation has comparable delay and space because the structural one. The rationale is that behavioral description offers the synthesis tools the liberty to optimally opt for logic for implementation. For instance, given certain temporal order constraints, the tool opts for CLA for addition and comparison operations throughout synthesis of the ACS unit, although an activity model is employed.

Bit-Serial (BS) ACS

A BS design for ACS has been proposed in [24] for a high constraint length, low throughput Viterbi decoder. This design uses some circuit level optimizations and custom designed FIFOs and aims to reduce power and area. The bit serial approach uses 1-bit adder over PMuw number of cycles to implement PMuw bit addition (or subtraction). Considerable area saving is expected if PMuw is large. Referring to (5) and (6), the PMuw value is high for high values of K and *softbits*. To compare bit-serial approach with other designs, a corresponding architecture was implemented with a standard cell library. The area and delay comparisons are shown in Table II. Area reduction nearly by a factor of two is seen for BS for $K=9$. In case of $K=4$, the area advantage is negligible. BM precision and PM precision determine the number of 1-bit adders (area for addition operation) in bit-serial approach and consequently the area advantage in using bit-serial over bit-parallel. Thus, bit-serial approach has an advantage in cases where the throughput requirements are met while the constraint length is large. Note that two different clock frequencies are required for a PMU built upon bit-serial ACS, since path metrics unit needs several clock cycles to process a single input symbol while other units typically run at one symbol per clock cycle rate.

Critical Factors in PMU Design

PMU contains an electric circuit. Therefore, in general, output can't be enlarged by pipelining (though as already mentioned, some look-ahead techniques enable loop unrolling and pipelining, however, are extraordinarily costly in terms of space and power consumption). Thus, PMU may be a bottleneck in achieving high output for the Viterbi decoder. At identical time, it occupies a considerable space considering the complete style. The most critical design parameters for the PMU are the input bitwidth, i.e., softbits and the number of bits to process per clock cycle (radix-2 versus radix-4). These parameters will be discussed in the following subsections in more detail.

Softbits

Number of bits per symbol, i.e., softbits influences not only the area and delay of the PMU but also the perfor-mance of the Viterbi decoder in terms of Bit Error Rate (BER). Figure 14 shows the dependency of BER on the Eb/No (energy per bit/noise) ratio for additive white Gaussian noise (AWGN) channel with $K=3$. As seen in the Figure, there is a significant gain in BER when moving from hard-input to 4 bits per symbol. Further increase has almost invisible gains (indeed, the curves for softbits= 5 and 6 are almost identical). Note that with channel models other than AWGN, increasing the input

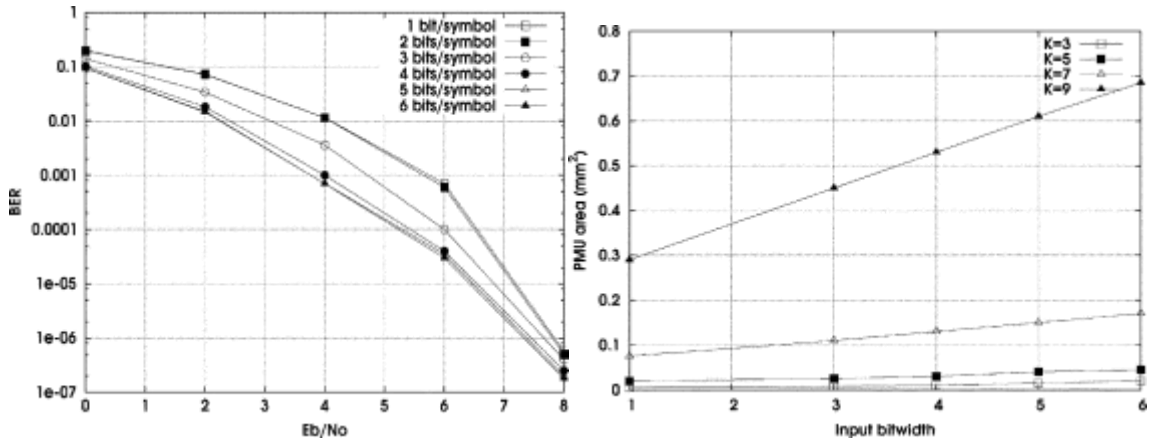

Fig. 14: *BER versuswith Varying Input Bitwidth.*

Fig. 15: *Critical Path Delay versus Input Bitwidth for Various Constraint Lengths (Radix-2).*

(Radix-2).

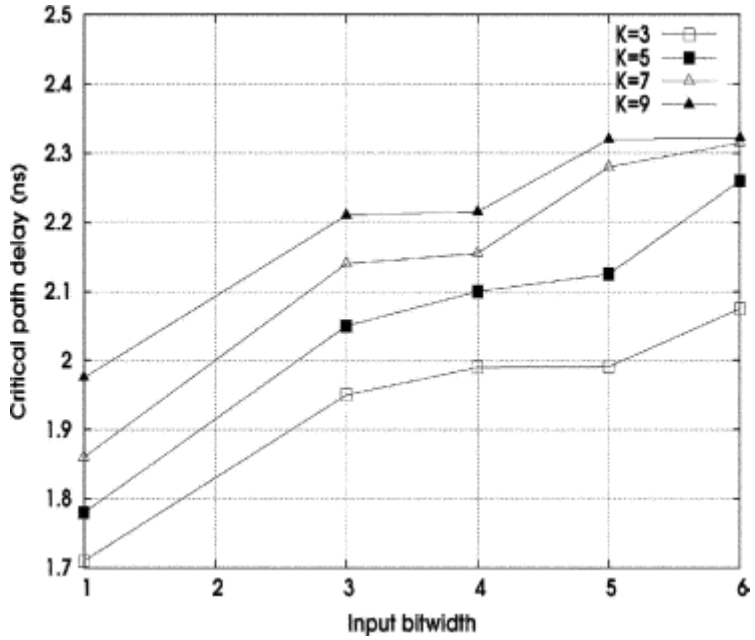

Fig. 16: *PMU Area versus Input Bitwidth for Various Constraint Lengths(Radix-2).*

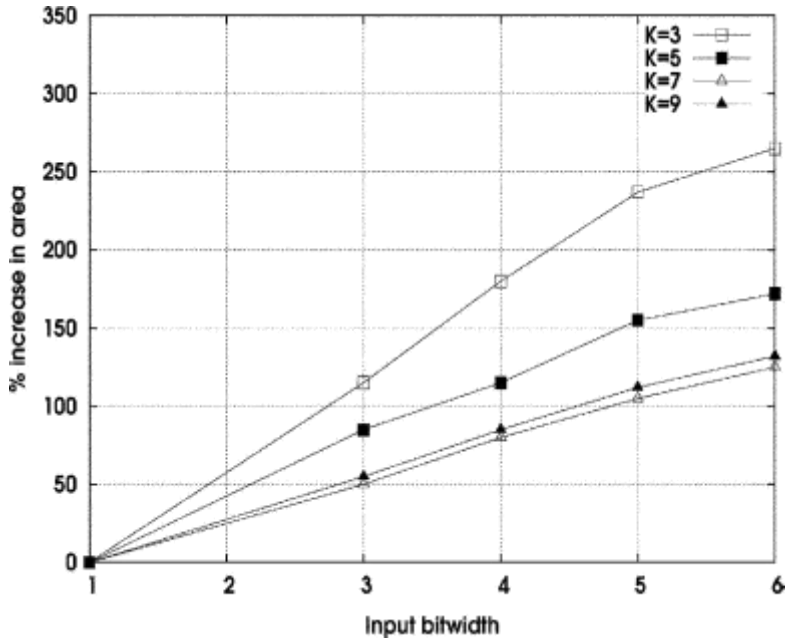

Fig. 17: *Increase in Area versus Input Bitwidth for Various Constraint Lengths (Radix-2).*

bitwidth beyond 4 bits still may make sense. However, the designer must always confine mind, that increasing the softbits parameter causes a rise in branch metric and path metric exactitude (and area) and in most cases decreases the outturn as illustrated by Figures 15–17 for radix-2 decoders with completely different values (conventional ACS, activity description). Figure 15 shows the affect of input bitwidth on the critical path delay for various K. As expected, in majority of the cases there is a linear increase in the critical path delay. Figure 16 displays the variation in space for the PMU with in-creasing softbits for varied. A transparent linear increase in space is visible all told cases in addition, since BM and PM precisions increase linearly with the input bitwidth. This is often as a result of the amount of full adders, multiplexers, and registers grow at a similar rate. Consequently, the graph in Figure 16 could be a nice illustration of what one would expect from basic theoretical analysis.

The increase in area for lower values of K is not very apparent in Figure 16. That is why Figure 17 was created to show the percentage increase in area over the area of PMU for hard-input decoder. It also shows that that there is a significant area penalty when increasing the softbits for all values of K. From (5) and (6), one can conclude that PM precision increases with the logarithm of K. At the same time, it increases linearly with input bitwidth, thus the influence of softbits on PM precision is more pronounced for lower values of K. This explains the highest rel-ative increase in area for the PMU for $K=3$.

Radix-2 versus Radix-4

Figure 18 shows the throughput achievable by using radix-2 and radix-4 decoder implementations. These results are supported timing-driven synthesis to realize the best output figures attainable. The ensuing throughputs are calculated by the synthesis tool, taking under consideration all clock connected problems like clock skew, uncertainty, insertion delay etc., before layout. To ascertain the back-end affects, variety of layouts were created additionally. The results indicate that on the average, the deviation between pre- and post-layout crucial path delay is a smaller amount than 5-hitter. When using the radix-4 technique, two output bits are produced per clock cycle. Therefore, the next turnout will be achieved even though the decoder is running at a lower clock frequency. Clock frequency becomes crucial, particularly in deep sub micron vary. Higher clock frequency results in cross speak problems and better power consumption besides decreasing signal responsibility. That is why

radix-4 is probably an appropriate selection in deep submicron domain, particularly if high turnout (e.g., 700 Mb/s) is targeted while not pushing the sting on the operative clock frequency.

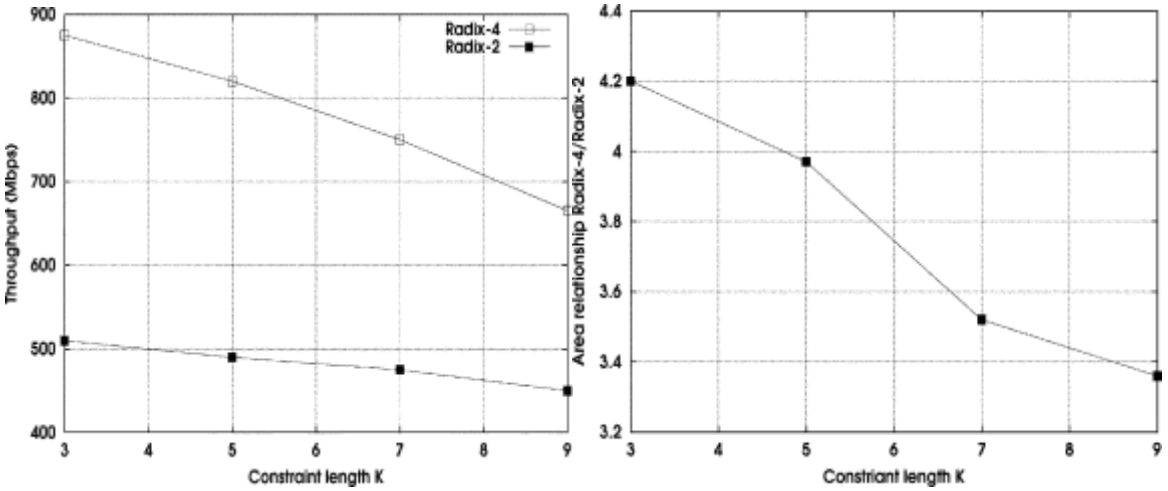

Fig. 18: *Maximum throughput versus Constraint Length for Radix-2 and Radix-4.*

Fig. 19: *PMU Area versus Constraint Length for Radix-2 and Radix-4.*

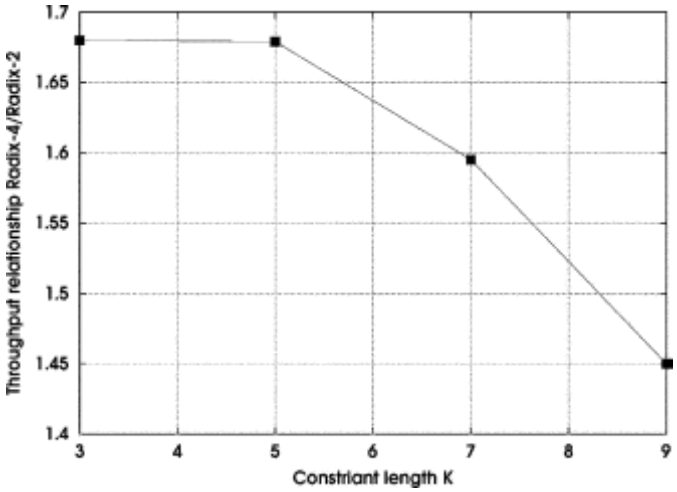

Fig. 20: *Area Ratio between Radix-4 and Radix-2 versus (Timing Optimized Synthesis).*

However, especially for higher K values, area penalty of radix-4 compared to radix-2 can become quite significant as Figure 19 shows. Alternatively, radix-4 technique can be used to achieve higher throughput at a higher area cost.

Figures 20 and 21 show the relative increase in space and outturn for radix-4 PMU compared to radix-2 PMU for various values of K. Since the logic of radix-4 ACS is additional

advanced compared to radix-2 ACS, the utmost clock frequency it will run at is lower. This means that just in case of AN optimisation towards highest possible outturn, a radix-4 ACS isn't specifically an element of two quicker than a radix-2 ACS jointly would expect within the case of synthesis beneath relaxed temporal order constraints. Still depending on constraint length, throughput can be increased by

40–70% (see Figure 21) provided that the PMU area penalty of a factor 3.4–4.2 (see Figure 20) can be afforded. Especially when radix-2 designs cannot meet the throughput requirements, this factor is a reasonable price to pay. Note that for K_3, comparable results were reported in [7]. Since the PMU is only one part of the Viterbi decoder, the overall penalty paid in terms of silicon area should be scaled accordingly (the area breakdown of a whole Viterbi decoder for K_7 will be presented below).

Figures 22–25 summarize throughput and area tradeoffs in the radix-2 and radix-4 PMU design for different constraint lengths and input bitwidths. To prove our findings on the PMU design, radix-2 and radix-4 Viterbi decoders for K_7softbits$_3$, and TBD$_128$

were implemented. The trace-back depth selection was done based on the simulations for a wireless fading channel under very noisy conditions (the value results from a baseband design project carried out by one of the authors). As known from lit-erature, in case of AWGN channel, lower values for TBD, e.g., 5–6 K, would suffice. Processing Re-write Suggestions Done (Unique Article) The synthesis was ran for worst-case military conditions. The post-layout netlists were simulated with temporal arrangement to see the most attainable output. Two cases documented as strained and relaxed synthesis were thought of to urge additional objective figures. Strained synthesis runs with tightest temporal arrangement constraints settings doable to get a de-coder with most output. Relaxed synthesis sets token temporal arrangement constraints to urge the bottom space figure. The results area unit summarized in Table 3 that conjointly shows the general space of the Viterbi decoder. supported these figures, radix-4-based Viterbi decoder achieves fifty fifth and 100 pc higher output at an element of 2 and 1.8 larger space for temporal arrangement driven and relaxed synthesis severally.

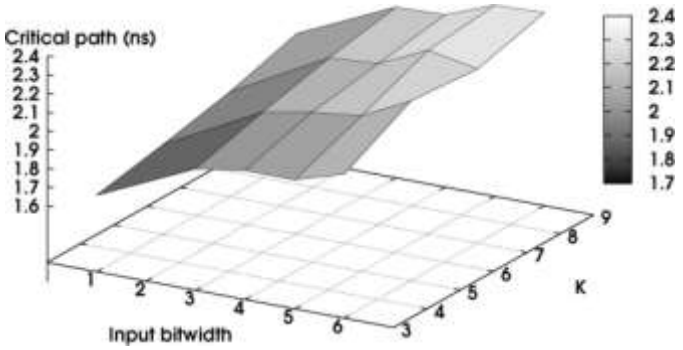

Fig. 21: *Throughput Ratio between Radix-4 and Radix-2 versus Constraint Length.*

Table 3: *Design Comparison of the Constrained and Relaxed Synthesis of Radix-2 and Radix-4 Decoder.*

	Radix-2		Radix-4	
	Constrained	Relaxed	Constrained	Relaxed
Delay (ns)	2.18	14.17	2.72	14.17
Clock (MHz)	458	70	367	70
Throughput (Mbps)	460	70	714	141
Area (mm^2)	0.279	0.165	0.568	0.311

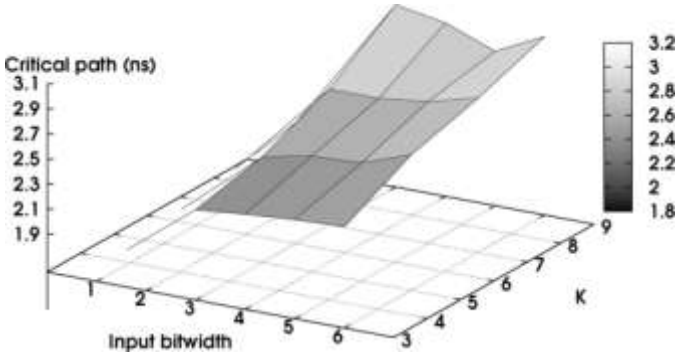

Fig. 22: *PMU Delay for Different Values of and Input Bitwidth (Radix-2).*

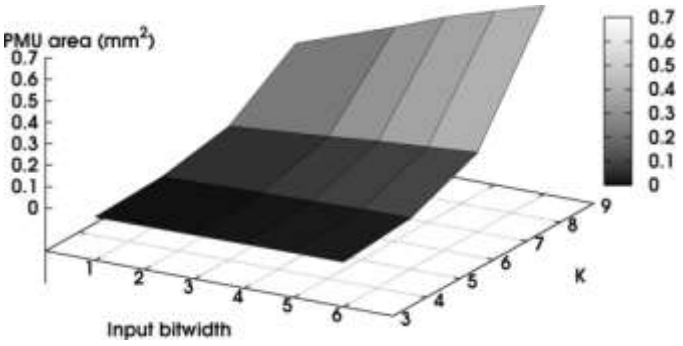

Fig. 23: *PMU Delay for Different Values of and Input Bitwidth (Radix-4).*

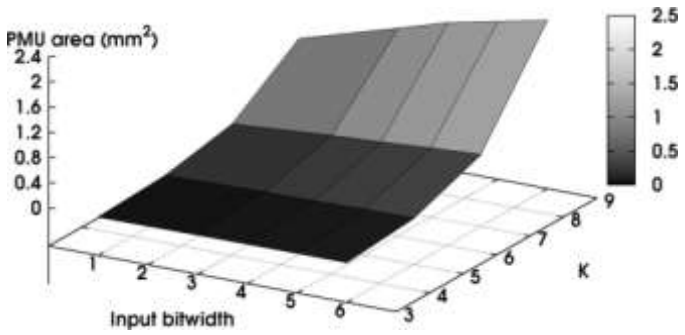

Fig. 24: *PMU Area for Different Values of and Input Bitwidth (Radix-2).*

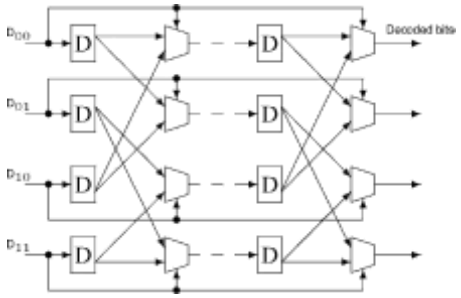

Fig. 25: *PMU Area for Different Values of and Input Bitwidth (Radix-4).*

Thus, area penalty is affordable compared to the increase in throughput. Table 4shows the detailed area breakdown for all sub-units of the Viterbi decoders for timing-driven and relaxed synthesis. It is clearly visible that in case of highly constrained synthesis, the worst area penalty occurs in the PMU. Due to the critical path in the PMU, timing constraint synthesis requires more hardware-intensive transformations. The BMU, trace-back logic, and the LIFO are much less critical in terms of delay and can meet the constraints without significant area penalty. During the timing-driven (constrained) synthesis, the radix-2 PMU becomes twice as large and the radix-4 more than 3 times larger in area compared to the relaxed case. In general, the radix-4 PMU leaves the synthesis tool somewhat more freedom for timing optimization since its logic is deeper and more complex than radix-2.

A factor that favors the radix-4 ACS is that it allows achieving the same throughput at a lower clock frequency. This helps the other modules of the Viterbi decoder such as the survivor memory unit (SMU). The reason is that the trace-back based SMU design requires one or more SRAM(s) and the use of fast SRAM results in high area and power consumption. The design of the SMU unit is discussed in the next section. Note that power dissipation aspects of the PMU have not been discussed so far. As it will be shown later, the power consumption of the PMU is a comparatively small fraction of the overall power

budget for a Viterbi decoder. That is why, for the design of the ACS architecture minimizing power consumption should not play a dominating role. For low-power decoders, general optimization techniques to reduce power dissipation like clock gating can be applied to any of the PMU architectures introduced in this section [30, 31].

SURVIVOR SEQUENCE UPDATE AND STORAGE UNIT

Design Space

The survivor sequence update and storage unit (briefly called survivor memory unit, SMU) receives decisions from the PMU and produces the decoded sequence. Figure 26 shows the design space for the SMU implementation. The implementation techniques for this unit can be divided in two classes: standard and adaptive. TBD determines the number of memory access operations required by the SMU. The value of the TBD parameter depends on the transmission channel conditions.

Table 4: *Area Comparison (in millimeters squared) between Constrained and Relaxed Synthesis of Radix-2 and Radix-4 Decoder.*

	Radix-2		Radix-4	
	Constrained	Relaxed	Constrained	Relaxed
BMU	0.00065	0.00065	0.02091	0.01620
PMU	0.11389	0.06413	0.38043	0.12845
Trace Back	0.01592	0.01579	0.02643	0.02637
LIFO	0.00377	0.00382	0.00413	0.00418

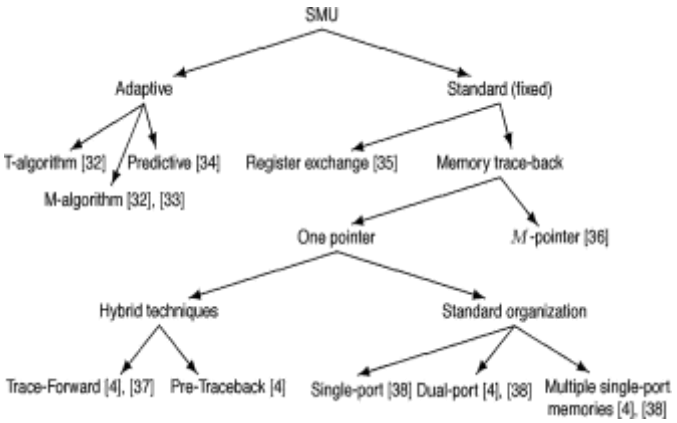

Fig. 27: *Simple Register Exchange Network corresponding to the Code Intro-Duced in Figure 1.*

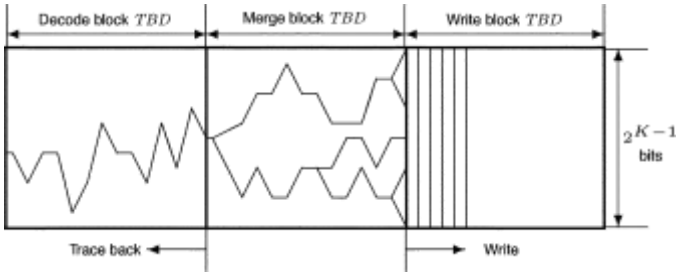

Fig. 26: *Design Space for SMU.* ***Fig. 28:*** *General Trace-Back Scheme [11].*

There are a unit variety of adaptational techniques mentioned within the literature. though all of them need extra hardware to be further to the SMU, on the average power dissipation of the unit may be reduced because of a lower range of memory accesses. The adaptational techniques referred to as M-algorithm and T-algorithm area unit mentioned, that reduces the quantity of states to think about by eliminating the states with overlarge path metric, essentially dynamically adopting the quantity of ACS to calculate and trace-back methods to store [32, 33]. This concept is more developed as adaptational approximation by adding a pruning unit to the decoder's core [39. The pruning unit limits the quantity of trace-back methods to follow, once more aiming at power consumption reduction. Moreover, Associate in Nursing SMU with dynamic prediction com- bined with trace-forward technique (see below) is introduced for Viterbi decoders utilized in wireless applications, that reduces the quantity of operation by quite seventieth in certain cases [34]. It is quite difficult to give general guidelines for the usage of registers in the RE technique, the power consumption is signif- adaptive techniques in the SMU, since their positive affects are icantly reduced. very channel/application dependent (as mentioned in all refer- TB is split in three major operations. These are write, merge, ences). For some designs, a very noisy channel can even lead to and decode as shown in Figure 28. If a survivor path is traced back some data loss in the decoder, so additional hardware is required for TBD stages, then the state at which all paths merge can be de- for recovery [32]. The only way to get figures for a specific determined. Symbols traced back beyond this state can be used as sign are experiments and simulation with accurate channel char decoded output. In each cycle 1 column of decisions is written acteristics in every single application case. Unfortunately, all into memory and columns are read for trace-back and de-references known to the authors only include simulation data for code. Pointers are used to keep track of the column read. The 1-pointer technique uses a single pointer multiple times to read

M columns. Multiple pointers are used in M-pointer technique for reading M-columns. The M-pointer technique requires in-efficient and expensive multi-port or multi-bank memory, so it is rather unsuitable for a VLSI implementation.

Trace forward (TF) technique is used to eliminate the merge stage (i.e., the buffer referred as the "merge block" in Figure 28) which estimates the starting state in decode operation [4]. This technique uses a set of registers that store the encoded state. In every clock cycle, the decision bits coming from the PMU are used to update these registers with new states corresponding to the previous trellis stage. After TBD clock cycles, all registers are expected to converge to the same state, which is at the same time the starting state for the decode operation. A refined version of this technique is introduced in [37].

Pre-traceback (PTB) is a technique used to reduce memory access frequency [4]. Instead of writing decisions at each stage processed by the PMU, m-stages are pre-traced and written as one composite decision. Thus, only one column of write operations is required for every m trellis stages processed by the PMU. The pre-trace is implemented by a RE network of typically 3 or 4 stages.

Note that especially for small values of TBD, some trace-back techniques like trace-forward can show lower BER performance than the conventional trace-back [40]. All techniques mentioned above involve several accesses to the memory in the same clock cycle. This implies some design decisions to be made with respect to the memory architecture. Memory can be organized as multi-port or single port with in-creased access rate. In case the access rate is too high, several single port memories can be interleaved in a ping-pong manner. Particular memory architecture strongly depends on the actual design case (e.g., availability of the multi-port memory, max-imum clock speed and width of the memory, etc.).

Throughput and area figures for the SMU can be calculated quite easily. For RE and TB, the critical path delay is determined by the flipflop toggle rate and SRAM access time respectively. The area for both increases linearly with TBD and exponentially with K

(15) $$\mathrm{SMU}_{\mathrm{Area}} \approx A \times \mathrm{TBD} \times B \times 2^{(K-1)}$$

where A is the number of trace-back memory blocks (4 with straightforward single-port-memory approach, 3 with folded read-write access, 2 if trace-forward is used, 1 for RE), B is an area of a flip-flop plus multiplexer for RE or an area of an SRAM bit for TB respectively.

As can be seen, the area and critical path delay of the SMU are fixed by the technology. The most important decision for the designer is the architecture choice between RE and TB which is mainly driven by power consumption. Therefore, the further analysis is completely dedicated to the power dissipation as-pects.

Implementation

Simulations of power dissipation were ran for a radix-2 $K=7$ Viterbi decoder with TBD of 128. The middle column of Table 5 shows the corresponding power dissipation profile. As

Table 5: *Power Dissipation Break-Down for Radix-2, Viterbi Decoder with rRunning at 500 mhz with 2-level ptb and at 167 mhz using Register Exchange.*

SMU Subunit	PTB Power (mW)	RE Power (mW)
BMU	0.143	0.036
PMU	59.690	10.190
SMU	117.000	42.040
LIFO	0.827	—
Rest (clock, pads etc.)	26.640	6.484
Total	204.300	58.750

Table 6:*Cell Area and Power Dissipation of Different Parts of the Trace-Back SMU.*

Unit	Area (mm^2)	Power (mW)
Trace-Forward	0.0113	3.441
Pre-Trace-Back	0.0025	2.175
TOP Memories	0.1440	110.900
Address Generation	0.0006	0.087
Read Mux	0.0011	0.178
Reset Logic	0.0003	0.137
Total	0.1599	117.000

Table 7: *Design Choices for Decision Memory.*

Optimization Criteria	Critical Units
Throughput	PMU
Latency	SMU
Power	SMU
Area	SMU, PMU

the decision memory in the SMU is synthesized using a stan-dard cell library, it has a high power consumption (110.9 mW compared to 6.1 mW for the rest of the SMU). Therefore for low power designs, the SMU should be optimized e.g., using low-power low-leakage SRAMs.

Such SRAM modules are available with power dissipation as low as 0.0324 mW/MHz. Using these memories in the Viterbi decoder from Table V would bring the power dissipation down to 22.3 mW at 250 MHz access frequency. In addition, it can be observed from Table VI that PTB occupy only 1.7% of the SMU area and dissipates only 1.9% of the total SMU power. This makes PTB an attractive solution to reduce the access frequency of the decision memory which enables the usage of low power high density SRAMs operating at a lower frequency. The right column of Table V shows power dissipation profile for the Viterbi decoder using the RE technique for the SMU. In this case, 72% of the total power is dissipated in the SMU. Compared to the TB implementation which costs 57% even with non-optimal standard cell based memories, this is quite high. As discussed above, the size of the RE network is determined by K and TBD. Therefore, we can conclude that for high values of K and TBD, RE is not a suitable design choice in terms of power.

As already discussed, for decoders running at very high speed, SRAMs of matching clock frequency may not be

Table 8: *Criteria for Optimization.*

Criterion	Memory Architecture
$f_p < \frac{f_m}{3}$	1 single-port memory+TF (optional)
$\frac{f_m}{3} < f_p < \frac{f_m}{2}$	1 single-port memory+TF
$\frac{f_m}{2} < f_p < f_m$	2 single-port memories+TF ($K > 7$) or 1 single-port memory+TF+ 2-level PTB ($K \leq 7$)
$f_m < f_p < 2 \times f_m$	2 N-bit single-port memories+TF+2–4-level PTB ($K > 7$) or 2 $\times$ N-bit 1 single-port memory+TF+4-level PTB ($K \leq 7$)

Table 9:*Summary of the Charts, Figures, and Tables.*

Design Aspect	Reference
Design space	Fig. 6 (top view), 7 (PMU), and 26 (SMU)
Algorithmic and word-level optimization techniques	Table I
PMU area and delay for different adder architectures	Fig. 12 and 13, Table II
PMU area and delay as a function of *softbits*	Fig. 15, 16, and 17
Comparison between radix-2 and radix-4 PMU	Fig. 18, 19, 20, and 21
softbits and K vs. critical path and area for radix-2 PMU	Fig. 22 and 24 (3D-plots)
softbits and K vs. critical path and area for radix-4 PMU	Fig. 23 and 25 (3D-plots)
Radix-2 vs. radix-4 comparison for a complete decoder	Table III and IV
SMU power and area figures	Table V and VI
Guidelines for SMU memory architecture choice	Table VIII

available. The memory architecture depends on the frequency at which the PMU is running and the maximum access fre-quency of the SRAMs. Table VIII summarizes the guidelines for the memory architecture choice depending on the clock frequencies. f_m denotes the maximum access frequency for the SRAMs. Operation frequency of the PMU is denoted by f_r, which implies the frequency at which the decision columns are generated.

CONCLUSION

In this paper, a comprehensive analysis of the Viterbi decoder design space is presented. Table 7 summarizes the importance of the different subunits of the decoder depending on the optimization criteria (the BMU is not mentioned, since its influence on area, power and throughput is negligible). As shown in Table 7, the PMU is critical for throughput while the SMU is critical for latency and power consumption. The most significant contributions of the paper can be summarized as follows:

- Detailed presentation of the design space for the hard-deci-sion Viterbi decoder and each of its subunits;
- Comparison of different adder architectures for the PMU implementation (ripple-carry, carry-lookahead, and bit-se-rial);
- Quantitative comparison of behavioral and structural HDL coding styles;
- Quantitative comparison of different ACS architectures; in particular, it has been shown that the general assumption of CSA being better than ACS and the radix-4 technique being inefficient in terms of area does not hold for some cases;
- Analysis of the influence of timing constraints (relaxed vs. constrained synthesis) on the design choices;
- Detailed analysis of the impact of constraint length and input bitwidth on area and throughput;
- Detailed analysis of different SMU implementation tech-niques and their impact on power consumption based on post-layout simulation.

For a better overview of the material, Table IX summarizes the figures, charts, and tables related to different design aspects to help the reader in finding the proper guidelines for choosing the Viterbi decoder design. So far, this work discusses most of the known VLSI implementation techniques for the hard-decision Viterbi algorithm in standard cell CMOS

technology and carefully analyzes the tradeoffs and dependencies between different design decisions. To the best of authors' knowledge, this is the most comprehensive analysis of hard-decision Viterbi algorithm VLSI implementation based on actual designs including post-layout experiments published so far.

ACKNOWLEDGMENT

The authors would like to thank Dr. N. Engin for valuable dis-cussions on hardware-software tradeoffs in Viterbi implementa-tion and A. Vaassen for supporting the layout experiments.

REFERENCES

1. H. D. Lin and D. G. Messerschmit, "Algorithms and architectures for concurrent Viterbi decoding," in *Proc. IEEE Int. Conf. Commun.*, Jun. 1989, pp. 836–840.
2. G. Fettweis and H. Meyr, "High rate Viterbi processor: A systolic array solution," *IEEE J. Sel. Areas Commun.*, vol. 8, no. 8, pp. 1520–1534, Oct. 1990.
3. G. Feygin, G. Gulak, and F. Pollar, "Survivor sequence memory man-agement in Viterbi decoder," in *Proc. IEEE Int. Sym. Circuits Syst.*, Jun. 1991, vol. 5, pp. 2967–2970.
4. P. J. Black and T. H.-Y. Meng, "Hybrid survivor path architectures for Viterbi decoders," in *Proc. Int. Conf. Acoust. Speech, Signal Process.*, 1993, vol. 1, pp. 433–436.
5. M. Boo, F. Arguello, J. D. Bruguera, R. Doallo, and E. L. Zapata, "High-performance VLSI architecture for the Viterbi algorithm," *IEEE Trans. Commun.*, vol. 45, no. 2, pp. 168–176, Feb. 1997.
6. K. K. Parhi, "An improved pipelined MSB-first add-compare select unit structure for Viterbi decoders," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 3, pp. 504–511, Mar. 2004.
7. Y. Engling, S. A. Augsburger, W. R. Davis, and B. Nicolic, "A 500-mb/s soft-output Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 38, no. 7, pp. 1234–1241, Jul. 2005.
8. G. Fettweis and H. Meyr, "High speed parallel Viterbi decoding: Al-gorithm and VLSI architecture," *IEEE Commun. Mag.*, vol. 29, no. 5, pp. 46–55, May 1991.
9. H. M. H. Dawid and G. Fettweis, "A CMOS IC for gb/s Viterbi de-coding: System design and VLSI implementation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 4, no. 1, pp. 17–31, Mar. 1996.
10. R. D. Wesel, *Wiley Encyclopedia of Telecommunications*. New York: Wiley, 2003, ch. Convolutional Codes.

11. P. J. Black and T. H. Y. Meng, "A 140-mb/s, 32-state, radix-4 Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 27, no. 12, pp. 1877–1885, Dec. 1992.
12. I. Lee and J. L. Sonntag, "A new architecture for the fast Viterbi al-gorithm communications," *IEEE Trans. Commun.*, vol. 51, no. 10, pp. 1624–1628, Oct. 2003.
13. G. Fettweis, H. Dawid, and H. Meyr, "Minimized method Viterbi de-coding: 600 Mb/s per chip," in *Proc. GLOBECOM 90*, Dec. 1990, vol. 3, pp. 1712–1716.
14. G. Fettweis and H. Meyr, "Cascaded feedforward architectures for par-allel Viterbi decoding," in *Proc. IEEE Int. Sym. Circuits Syst.*, May 1990, pp. 978–981.
15. 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, no. 6, pp. 797–805, Jun. 1997.
16. G. Fettweis and H. Meyr, "Parallel Viterbi decoding by breaking the compare-select feedback bottleneck," *IEEE Trans. Commun.*, vol. 37, no. 8, pp. 785–790, Aug. 1989.
17. C. B. Shung, H.-D. Ling, R. Cypher, P. H. Siegel, and H. K. Thapar, "Area-efficient architectures for the Viterbi algorithm part I: Theory," *IEEE Trans. Commun.*, vol. 41, no. 4, pp. 636–644, Apr. 1993.
18. C. B. Shung, H.-D. Ling, R. Cypher, P. H. Siegel, and H. K. Thapar, "Area-efficient architectures for the Viterbi algorithm part II: Applica-tions," *IEEE Trans. Commun.*, vol. 41, no. 5, pp. 802–807, May 1993.
19. T. Jarvinen, P. Salmela, T. Sipila, and J. Takala, "Systematic approach for path metric access in Viterbi decoders," *IEEE Trans. Commun.*, vol. 53, no. 5, pp. 755–759, May 2005.
20. M. Benaissa and Y. Zhu, "A novel high-speed configurable Viterbi de-coder for broadband access," *J. Appl. Signal Process. EURASIP JASP*, no. 13, pp. 1317–1327, 2003.
21. J. J. Kong and K. K. Parhi, "K-nested layered look-ahead method and architectures for high throughput Viterbi decoder," in *Proc. IEEE Work-shop Signal Process. Syst.*, Aug. 2003, pp. 99–104.
22. J. J. Kong and K. K. Parhi, "Low-latency architectures for high-throughput rate Viterbi decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 6, pp. 642–651, Jun. 2004.
23. V. Madisetti, *VLSI Digital Signal Processors: An Introduction to Rapid Prototyping and Digital Synthesis*. Newton, MA: Butter-worth-Heinemann, Jun. 1995.
24. Y. N. Chang, H. Suzuki, and K. K. Parhi, "A 2-Mb/s 256-state 10-mw rate-1/3 Viterbi

decoder," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 35, no. 6, pp. 826–834, Jun. 2000.

25. G. Fettweis and H. Meyr, "A 100 mbit/s Viterbi decoder chip: Novel architecture and its realization," in *Proc. IEEE Int. Conf. Commun.*, Apr. 1990, vol. 3, pp. 463–467.
26. V. S. Gierenz, O. Weiss, T. G. Noll, I. Carew, J. Ashley, and R. Karabed, "A 550 Mb/s radix-4 bit-level pipelined 16-state 0.25-m CMOS Viterbi decoder," in *Proc. IEEE Int. Conf. ASAP*, 2000, pp. 195–201.
27. T. Gemmeke, M. Gansen, and T. G. Noll, "Implementation of scalable power and area efficient high-throughput Viterbi decoders," *IEEE J. Solid-State Circuits*, vol. 37, no. 7, pp. 941–948, Jul. 2002.
28. A. K. Yeung and J. M. Rabaey, "210 Mb/s radix-4 bit-level pipelined Viterbi decoder," in *Proc. IEEE Int. Solid-State Circuit Conf.*, Feb. 1995, pp. 88–89.
29. A. Hekstra, "An alternative to metric rescaling in Viterbi decoders," *IEEE Trans. Commun.*, vol. 37, no. 11, pp. 1220–1222, Nov. 1989.
30. S. Ranpara, "On a viterbi decoder design for low power dissipation," Master's thesis, Bradley Dept. Elect. Comput. Eng., Virginia Polytech. Inst. State Univ., Blacksburg, Apr. 1999.
31. I. Bogdan, M. Munteanu, P. A. Ivey, N. L. Seed, and N. Powell, "Power reduction techniques for a viterbi decoder implementation," in *Proc. 3rd Int. Workshop (ESDLPD)*, Rapello, Italy, Jul. 2000, pp. 5–9.
32. M. H. Chang, W. T. Lee, M. C. Lin, and L. G. Chen, "IC design of an adaptive Viterbi decoder," *IEEE Trans. Consum. Electron.*, vol. 42, no. 1, pp. 52–62, Feb. 1996.
33. E. Boutillon and L. Gonzalez, "Trace back technques adapted to the surviving memory managementin the m algorithm," in *Proc. Int. Conf. (ICASSP)*, Istanbul, Turkey, Jun. 2000, pp. 3366–3369.