

A Novel VLSI Architecture For ACS Operation In Adaptive Viterbi Decoding

Yao.Gang,Tughrul.Arslan

School of Electrical and Engineering, University Of Edinburgh

Edinburgh,UK EH9,3JL

Email: {y.gang}@ed.ac.uk

Abstract—The trend in the wireless communication systems has indicated the need to dynamically adapt communication hardware architectures according to the run time physical environment. The adaptive Viterbi algorithm, which adaptively computes the survivor path according to the channel environment, reduces the computation of complexity and correspondingly power consumption greatly. In this paper, the authors present a novel VLSI architecture of Add-Compare-Select(ACS) unit, which is the key component of the adaptive Viterbi decoder. In the proposed scheme, the compare unit is simplified from the variable based to constant based by employing a simple transform to the Adaptive Viterbi Algorithm. Based on the simplicity of the constant comparison circuit, we develop a parallel approach to select N_{max} paths out of $2N_{max}$ possible paths. The proposed transformed architecture eliminates the need for path metric storage unit and reduces the complexity of the comparison operation. The proposed ACS architecture achieves 57%, 47% and 7% improvement in power, area and speed respectively compared to the conventional approach. Moreover, there is a 25% to 47% storage reduction in the Path Metric Unit(PMU).

Keywords: Viterbi Algorithm(VA), Adaptive Viterbi Algorithm(AVA), ACS

I. INTRODUCTION

Combining flexibility, performance and energy-efficiency is an important design aspect for the future wireless system [1]. The Viterbi decoder, which is the most commonly used forward error correction scheme in the wireless applications such as CDMA and WLAN systems, is embracing a great challenge to consume lower power while achieving considerable performance.

The computation complexity and storage requirement of the conventional Viterbi Algorithm (VA) increased exponentially to the constraint length of the convolutional encoder [2]. For instance, the CDMA standard IS-95 employs a convolutional encoder with 8 states. Hence, there are 256 states in the trellis graph. Considering the strict constraints on the area and the power in the mobile systems, it is quite difficult and complex to implement the conventional VA. The number of the states is the key to determine the complexity of the Viterbi Decoder. There has been some approaches refer to as the suboptimal Viterbi Algorithm (VA) in this research line [3] [4] [5]. The basic idea of the suboptimal VA follows the conventional VA except to reduce the visited state during trellis recursion while sustaining considerable decoding performance.

Chan's Adaptive Viterbi algorithm (AVA) [5] takes advantage of the other ones in terms of the resynchronization and error propagation issues.

In this paper, we focus on the design of the ACS unit, which is one of the key components of the adaptive Viterbi decoder. Up till now, Sriram's work [6] is the only one that described the VLSI architecture of AVA algorithm in literature. In [6], the minimum path metric are stored into the storage unit in the current iteration of the trellis recursion and are read out for the sake of discarding the paths in the next iteration. By applying a very simple transform to the rules of the AVA, we observed it is unnecessary to store the minimum path metric in every iteration and the variable based comparison circuit can be totally simplified to the constant based comparison. Moreover based on the simplified constant compare unit, we develop a parallel approach to select N_{max} possible paths from $2N_{max}$ paths with a great reduction in the design complexity.

The paper is organized as following: In section II, we briefly describe the AVA algorithm and previous approach by Sriram. Our proposed Architecture is described and discussed extensively in section III. In section IV, we present the area, energy and speed simulation result.

II. ALGORITHM DESCRIPTION

The well-known Viterbi Algorithm (VA) has been described in literature extensively. The VA examines all possible paths in the trellis graph and determines the most likely one. The AVA only keeps a number of the most likely states instead of the whole of 2^{K-1} states, where K is the constraint length of the convolutional encoder [2] [5]. The rest of the other states are all discarded. The selection is based on the likelihood or metric value of the paths, which for a hard decision decoder is the Hamming distance and a soft decision decoder is Euclidean distance. The rules of the selecting the survivor path is :

1) : Every surviving path at trellis level $l - 1$ is extended and its successors at level l are kept if their path metric are smaller or equal to $d_m + T$, where d_m is the minimum path metric of the surviving path at stage $l - 1$, T is discarding threshold configured by the user.

2) : The total number of survivor paths per trellis stage is up bounded to a fixed number: N_{max} , which is pre-set prior to the start of the communication.

In order to illustrate how the AVA works, an example using a code rate $R = 1/2$, constraint length $K = 3$ is given in Fig1. The Threshold T is set to 1 and N_{max} is set to 3 respectively. Initially at $t = 0$, we set the d_m equal to 0 and the decoder states equal to 00. The received sequence is

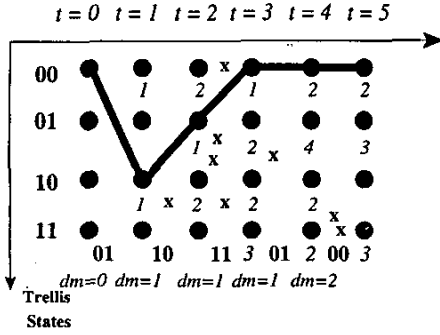


Fig. 1. Trellis Graph Of Adaptive Viterbi Decoding

TABLE I
OPTIMAL ARCHITECTURE PARAMETER OF N_{max} AND T

K	N_{max}	T
4	4	14
7	8	17
9	9	19

{01, 10, 11, 01, 00}. The symbol X represents the discarded path and bold line represents the final decision path by the AVA algorithm. It can be seen that at each trellis stage, the number of the survivor states is smaller than the VA (2^{K-1}) and gets the same decision paths as the VA.

It is obvious that the number of the T and N_{max} influences the number of the survivor paths and correspondingly the decoding performance Bit Error Rate (BER). If the T and N_{max} are set big enough, the behaviour of the AVA algorithm will be exactly the same as the VA algorithm but with a big hardware overhead. Consequently, there is a trade off between the performance and complexity. [5] [6] have described a selection strategy for determining the optimal value of N_{max} and T . In this paper, we will use their result as illustrated in TABLE I as parameter for AVA architecture.

Sriram's work [6] is the only hardware approach to implement the AVA mentioned above. Details of the approach are showed in Fig 2. In the ACS unit, the path metrics for all potential next state paths, which is denoted by d_i are computed by addition of branch metric BM_i from Branch Metric Unit (BMU) unit and corresponding path metric PM_i stored in Path Metric Unit (PMU). Comparators are then used to compare the new path metrics to the sum of $d_m + T$ to identify path metrics with excessive cost. Those exceeding the threshold path are discarded. The path metric of the survivor state is stored in the Path Metric Unit (PMU). At the same time, the minimum-value surviving path metric among all path metrics, d_m is determined for the preceding trellis stage and stored in the d_m storage unit.

III. PROPOSED ARCHITECTURE

A. Algorithm Transform

Let l represents the time step during the trellis recursion i.e. $l \in N = \{1, 2, 3, \dots\}$. And $s, k \in S$ represent the states

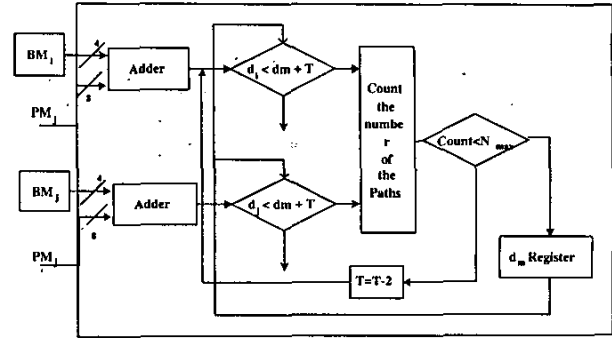


Fig. 2. Conventional ACS Architecture Of Adaptive Viterbi Decoder

of the trellis graph and s is the predecessor state of k . And the path metric of state k in time step l can be denoted as $d_k(l)$ and the minimum path metric at time step $l-1$ can be simply denoted as d_m . So the rule 1) of the AVA algorithm can be expressed as

$$d_k(l) \leq d_m + T \quad (1)$$

As the state s is the predecessor of the k , and we let p denote the branch metric refer to state transition s to k :

$$d_k(l) = d_s(l-1) + p \quad (2)$$

While the d_m is the minimum path metric of the time step $l-1$,

$$q = d_s(l-1) - d_m \geq 0 \quad (3)$$

Plug (2) and (3) into (1), AVA equation (1) can be expressed as

$$q + p \leq T \quad (4)$$

Where q can be seen as the rescaled version of the path metric of the trellis graph at time step $l-1$, and p is branch metric calculated by the Branch Metric Unit (BMU).

B. Constant Comparison

Compared to Sriram's work based on the form of AVA equation (1), we propose an architecture based on the form of AVA equation(4). In our approach, the threshold selection is based on the rescaled version of path metric, which is denoted by q . At every stage of the trellis recursion, the current stage path metric is compared to the preset constant T . As shown in Fig 3, in form of eq(4), we reduce one adder and the d_m storage overhead and more importantly, change the compare select operation from the variable based to constant based just as illustrated in Fig 3. We know the variable based comparator can be implemented by the adder and exerts a heavy burden on the critical path and power consumption of the ACS unit. While the constant based comparison can be implemented very efficiently. Three type basic comparison cell units are designed in UMC 0.18 μ m standard cell technology. TABLE II summarizes the comparison of the result of compare function

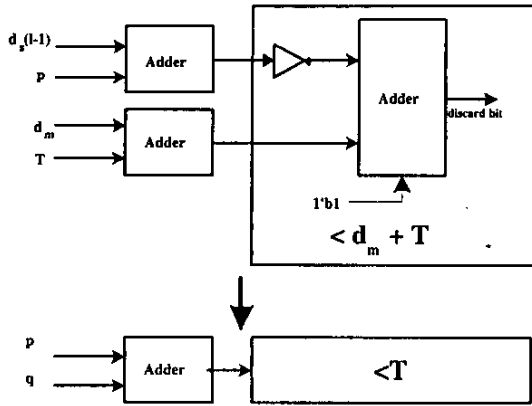


Fig. 3. Algorithm Transform Of AVA Algorithm

TABLE II
COMPARISON OF HARDWARE OVERHEAD OF VARIABLE BASED
COMPARISON AND CONSTANT BASED COMPARISON ($T = 20$)

	Area	Power	Delay
$a < T$	1	1	1
$a < b$	8.34	2.03	1.88
$a < b + T$	9.87	2.18	2.47

$a < T$, $a < b$ and $a < b + T$, where $T = 20$ is a typical constant used in AVA algorithm and a and b are both 5 bits wide vectors. Here we normalize the value of the area, power and speed. It can be found that significant saving can be obtained by the constant based comparison in area, and power consumption simultaneously. The improvement of the speed is not so significant because in conventional approach, the sum of $d_m + T$ can be calculated in parallel with the sum of the path metric.

It can be also noted that as we do not store the minimum path metric, we still need to calculate the minimum path metric on the current stage of the trellis, and in order to calculate the value of q , an additional subtraction operation is needed. It shall be shown that this kind of computation contains no redundancy at all.

It can also be found that the path metric will increase as the receiving channel message increases so the path metric will overflow under certain circumstances. Two methods are known to address this problem: modulo logic and rescale technique [7]. In [7], the author concluded that at a penalty of increase in path metric storage, modulo logic in 2's complement system is superior to the rescale logic because the residual delay introduced by the determination of minimum metric. And in the Viterbi decoders that have been in the literature including Sriram's work, all adopt the modulo logic. While in AVA, the calculation of the d_m is necessary and the residual overhead of calculating is inherent with AVA. What is more important is that the subtraction operation in our proposed transform corresponds to the same procedure in rescale technique, and the q represents the normalization vector [9]. So the width of the path metric can be reduced. In a

TABLE III
COMPARISON OF WIDTH FOR THE PATH METRIC UNIT

K(Constraint Length)	4	6	7	9
Path Metric Width (Proposed)	3	4	5	5
Path Metric Width (Conventional)	4	8	8	9

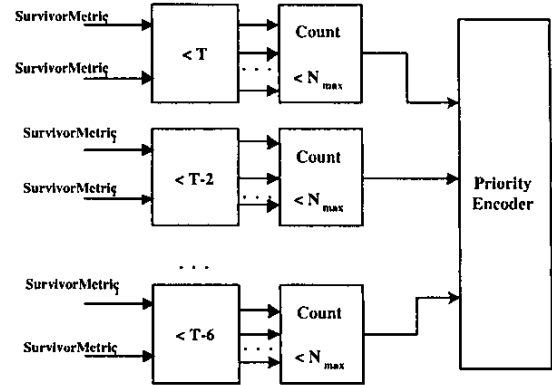


Fig. 4. Block Diagram Of Parallel Sift Architecture

$R = 1/2$, $K = 9$ Viterbi decoder with 3 bits soft quantization, the width of path metric employing modulo logic is 8 [8]. While according to our approach, only 7 bits path memory storage is needed in the first appearance. Moreover, as the compare select operation is based on the constant T , which has an optimal value of 26 according to [5], we only need 5 bits to store the path metric because all the paths with the metric bigger than T will be discarded. The simplified path metric reduces the complexity of the add of the path metric and the determination of the d_m further. In TABLE III, we briefly calculate the path metric width requirement of our proposed approach and the conventional one and found that a storage efficiency of [25%, 47%] can be achieved under different hardware specifications. In summary, compared to the conventional approach, the rescale operation introduces slight hardware overhead while reduces the complexity of the comparison unit and path metric storage unit efficiently.

C. Parallel Sifting

If the number of the survivor paths that satisfy the VA rule exceeds the number of N_{max} , we must sift N_{max} paths out of them. Under the worst condition, we need to sift N_{max} out of $2N_{max}$ paths. It is obvious that we can achieve the goal via a sorting operation just as proposed in M algorithm indicated [9] but with great hardware complexity and corresponding high energy consumption. And in Sriram's approach, he achieves this by making feedback adjustments to the parameter T . If the number of paths that survive the threshold is less than N_{max} , T is iteratively reduced by 2 for the current trellis stage until the number of the paths surviving the threshold condition is less than N_{max} . In this approach, the worst case iteration must be satisfied in order to achieve constant decoding latency and as there is feedback path, the ACS unit must work under the higher frequency than the other parts such as BMU unit

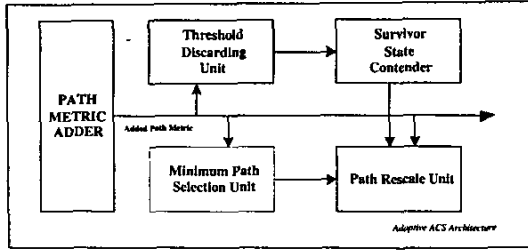


Fig. 5. Block Diagram Of Proposed Adaptive ACS Architecture

which make it very difficult to implement in the standard cell based design due to the complexity of the clocking tree schema. Based on the low overhead of the constant based hardware version of the compare scheme, we propose to adopt a parallel approach to tackle this problem. As shown in Fig 4, in our proposed scheme, the added path metric (the number is up bounded by $2N_{max}$) are compared parallel to a series of constant $T, T-2, \dots, M$, where M corresponding to the worst case iteration and simulation showed $T-6$ is enough for the correct sifting operation. The number of the path that satisfy the threshold condition is counted parallel and are fed into to a Priority Encoder. Through Priority Encoder, we select the N_{max} survivor paths.

IV. RESULT

We implemented our novel approach in a fully synthesizable Verilog models as described in Fig 5 with 0.18um UMC standard cell library. The specification of the AVA decoder is $R = 1/2, K = 7$ with 3 bit soft decision. Synopsis tool is employed for synthesis and characterization. Power analysis is performed by Synopsis Design Power tool. These results are summarized in TABLE IV. 57%, 47% and 7% improvement are achieved in power efficiency, area and speed respectively. Here we do not take into the consideration of the Branch Metric Unit (BMU) and Survivor Memory Unit (SMU) because the two components in different approaches are the same. It can be noted that although the introduction of the rescale logic is an overhead compared to conventional approach, there is still an overall significant improvement in area and power and modest improvement in speed. Compared to the TABLE II, improved power saving can be contributed to the reduced width of the path metric vector.

TABLE IV
COMPARISON OF PROPOSED AND CONVENTIONAL AVA ARCHITECTURE

	Area	Power	Delay
Proposed Architecture	1	1	1
Conventional Architecture	1.87	2.28	1.07

V. CONCLUSION

An efficient architecture of ACS unit of AVA decoder based on the transformed AVA equation is presented. The proposed scheme simplifies the compare operation from variable based

to constant based through a simple transform to the AVA rule. Compared to conventional approach, 57%, 47% and 7% savings are achieved simultaneously in power, area and speed. The proposed architecture also achieves up to 47% storage efficiency reduction in the Path Metric Storage Unit (PMU). The architecture with reduced hardware complexity is very attractive for the wireless application with high constraint length such as CDMA and WLAN systems.

REFERENCES

- [1] Jan.M.Rabaey, "Wireless Beyond the Third generation - Facing the Energy Challenge," *ISLPED '01 Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, Monterey, pp.1-3, Aug. 2001.
- [2] A.J.Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. on Commun.*, vol.COM-19, pp.751-771, Oct. 1971.
- [3] Stanley.J.Simmons, "Breadth-first trellis decoding with adaptive effort," *IEEE Trans. on Commun.*, vol 38, pp.3-12, Jan. 1990.
- [4] J.B.Anderson and Elke Offer, "Reduced-State Sequence Detection with Convolutional Codes," *IEEE Trans. Inform. Theory*, vol.40, pp.965-972, May. 1994.
- [5] Francois Chan and David Haccoun, "Adaptive Viterbi Decoding of Convolution Codes over Memoryless Channels," *IEEE Trans. on Comm.*, vol.45, pp.1389-1400, Nov. 1997.
- [6] Sriram Swaminathan, Russell Tessier and Dennis Goeckel etc. "A Dynamically Reconfigurable Adaptive Viterbi Decoder," *FPGA '02, International Symposium on Field Programmable Gate Arrays archive Proceedings of the 2002*, pp.227-236, Feb. 2002.
- [7] A.P.Hekstra, "An alternative to metric rescaling in Viterbi decoders," *IEEE Trans. on Commun.*, vol.37, no.11, pp.1220-1222, Nov. 1989.
- [8] Peter J.Black and Teresa H.Meng, "A 140-Mb/s 32-State, Radix-4 Viterbi Decoder," *IEEE J. Solid-State Circuits.*, vol.27, pp.1877-1885, Dec. 1993.
- [9] Peter A.Bengough and Stanley J.Simmons, "Sorting Based VLSI Architectures for the M-algorithm and T-algorithm Trellis Decoders," *IEEE Trans. on Commun.*, vol.43, pp.514-521, Feb./Mar./Apr. 1995.