IEEE 6th CAS Symp. on Emerging Technologies: Mobile and Wireless Comm.
Shanghai, China, May 31~June 2, 2004

Designing of Precomputational-based Low-Power Viterbi Decoder

Jing-ling Yang, Alfred, K.K. Wong
Department of Electrical and Electronic Engineering
The University of Hong Kong
jlyang, awong@eee.hku.hk

Abstract— This work addresses the low-power VLSI implementation of the Viterbi decoder (VD). A new precomputational scheme applied to the trellis butterflies calculation is presented. The proposed scheme is implemented in a 16-state, rate 1/3 VD. Gate-level power verification indicates that the proposed design reduces the power dissipated by the original trellis butterflies calculation by 42%.

I. INTRODUCTION

The Viterbi algorithm [1], which has been extensively applied to several decoding and estimation applications in communication and signal processing, was introduced in 1967 as a method for decoding convolutional code [2].

Convolutional encoding with Viterbi decoding is one of the most popular forward-error-correction methods for error correction in communication systems. In decoding, the VD attempts to reconstruct the action of the encoder based on the transmission of its outputs over a noisy channel.

The VD is comprised of three basic units - a branch metric unit (BMU), an add-compare-select unit (ACSU) and a trace back unit (TBU). The VD dissipates most of its power on ACSU and TBU.

The feedback loop can not be parallelized using a standard method, so the ACSU is normally considered to be the most critical part of the implementation of high speed Viterbi algorithms. Reference [3] and [4] showed that, despite the feedback loop, high speed can be achieved using a purely feed-forward parallel implementation that operates on the M-Step trellis. A high-speed VD that needs that must calculate more parallel trellis butterflies in parallel usually consumes more power.

Numerous techniques for reducing power dissipation on TBU have been proposed. Scarce state transition architecture [5] has been used to reduce the switching rate of the SMU. In [6], both system and circuit techniques have been proposed to reduce the power consumed by the SMU. In ACSU design for low power consumption, an adaptive VD [7] dynamically discards some states in the trellis that have high path metrics during the decoding, however, the extra controls required are too complex. Reference [8] and [9] introduce the low power design of large state VD.

This work presents a precomputation-based low power design scheme, which can also be applied to high-speed VD design, to perform the low-power trellis butterflies calculation. The proposed architecture is validated by implementating a 16-state, rate 1/3 VD. Designed to use a 0.13 um standard cell library, at a supplied voltage of 1.2V, the described decoder achieves a dissipations of 400uw at a throughput of 25MHZ, with only minor design modifications.

II. CALCULATION TRELLIS BUTTERFLIES

Fig.1 shows a basic diagram of a VD. The function of each block is described briefly below.

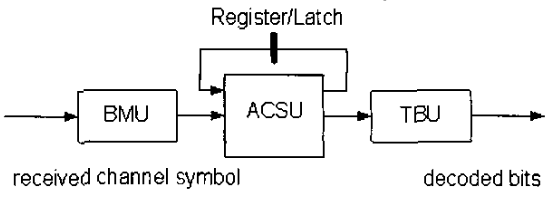

Fig. 1: Basic diagram of a Viterbi decoder

BMU generates the branch metrics, which measure the difference between the received symbol and the symbol that causes transitions between states in trellis.

ACSU is a collection of butterflies' calculations. Each butterfly calculates the path metrics of the two paths that connect two old statuses at time t to a new state at time t+1, and selects a smaller one as the new path metric of the new state.

The TBU traces the decision vector to generate the corrected output sequence. This can be done either by forward-processing the decision using the so-called register-exchange algorithm or by backward-tracing the previously stored decisions.

The quality of a VD design is primarily measured by applying four criteria - coding gain, throughput, area and power consumption. This work focuses on the power-efficiency implementation of the trellis butterfly calculation.

A. Trellis Butterfly

A trellis diagram is a simple means of visualizing the input and output sequences of a convolutional code [1] and [2]. The trellis diagram for a convolutional code can be subdivided into a number of basic modules, as shown in Fig. 2, in which, BM is a branch metric, S is the number of states, and $0 \le j \le S/2-1$. For efficient processing, the VD is implemented to compute two trellis butterflies in parallel.

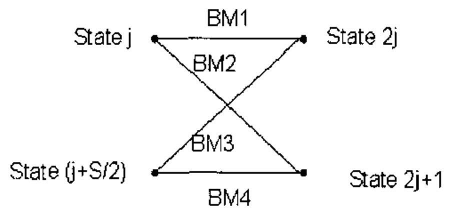

Fig.

Fig. 2: Trellis Butterfly

These trellis butterflies depict the transitions between two old states at trellis stage i and two new states at trellis stage i+1.

Each stage of a trellis diagram has 2(k-1) trellis butterflies, where K is the constraint length of the convolutional code. To visualize this feature of a trellis diagram, consider one stage of the trellis diagram for a convolutional code with a constraint length of K=3. See Fig. 3.

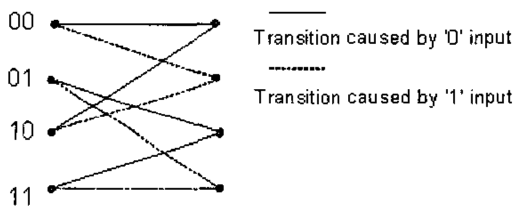
00
01
10
11
Transition caused by '0' input
Transition caused by '1' input

Fig.3: One Stage of a Trellis Diagram for K=3 and R=1/n

The VD computes the trellis butterflies by using those basic function units called ACSj, 0<=j<=2(k-1)-1. See Fig.3.

Computing a trellis butterfly includes the following steps:

1. Reading the path metrics and survivor paths of states j and j+s/2 at stage i.
2. Computing the path metrics of states 2j and 2j+1 at stage i+1.
3. Comparing the two path metrics and selecting the path with the smaller one.
4. Storing the updated path metrics and survivor paths.

B. ACSU Implementation

Fig. 4 shows a conventional VLSI architecture for implementing the ACSU butterfly unit of A VD. See Fig.2. The core structure, called ACSj, is confined within the dotted rectangle. The ACSU runs recursively. The new path metrics of the current recursion will be the old metrics of the sequent recursion.

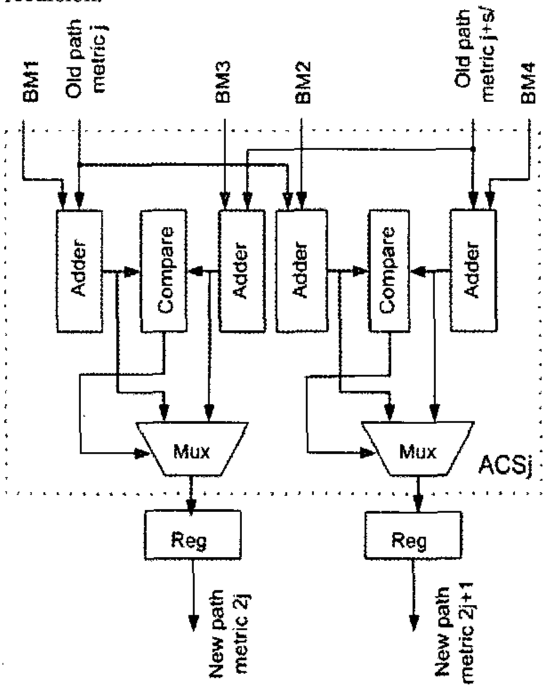

Fig. 4: Architecture of Conventional ACSU

In ACSU implementation, see Fig. 4, two competing paths arrive at each state in each cycle. PM_{i+1}^{2j}, PM_{i+1}^{2j+1} represent the survivor path metric of state 2j, 2j+1 at time i+1 respectively, PM_i^j and $PM_i^{j+S/2}$ are old path metric of state j and j+S/2 at time i respectively. The new path metrics of states 2j and 2j+1 at i+1 are calculated in two butterflies which are defined in following equations.

$$PM_{i+1}^{2j} = \min(PM_i^j + BM1, PM_i^{j+S/2} + BM3) \quad (1)$$

$$PM_{i+1}^{2j+1} = \min(PM_i^j + BM2, PM_i^{j+S/2} + BM4) \quad (2)$$

For a 16-state, rate 1/3, VD, 32 additions and 16 compare-and-select operations have to be done for every decoded bit. The number of operation is significant in relation to the number of operations associated with other unit, such as the BMU and SMU. For the high speed VD with parallel structure, the number is even large. The power consumed by the ACSU must be minimized to reduce the power consumed by VD.

III. Precomputation Scheme

Precomputation logic, first proposed in [10], is optimized to trade area for power in a synchronous digital circuit. The principle of precomputation logic is to identify logical conditions at some inputs to combination logic that do not affect output. Since those input values do not affect the output, the input transitions can be disabled to reduce the switching activities.

A. Proposed Precomputation Logic

Fig. 5 shows the precomputation architecture of a 16-state, rate 1/3 VD. Due to the nature of ACSj, there are some conditions under which the output is independent of some of the values of the input registers and latches. For a code rate 1/3 VD, the maximum number of branch metrics is three, when the difference between the old path metric of the two competing paths exceeds three, the selected path can be determined independent of the input data of another path. Under such precomputation condition referred as G(X) in Fig. 5, we can disable register load of these registers and latches to prevent unnecessary switching, thereby conserving power. The ACSU is correctly computed because it receives all required value from the remaining register.

Comparing Fig 4 and Fig. 5 shows that the extra logic added for the precomputation is latch and G(X). Since the number of branch metric is usually smaller, latches do not cause much hardware cost. For example, an R=1/3 code has a maximum branch metric of three, so two-bit latches suffice to control a branch metric. To be efficient, the selection logic G(X) must also be simple. The following section introduces a simple G(X) design.

B. Precomputation Conditions

To generate the load-disable signal to the unusable registers and latches, a precomputation function G(X) is required to detect the condition under which ACSU is independent of the unusable registers and latches. For ACSU, an obvious G(X) is:

$$\left| \mathrm{PM}_i^{j} - \mathrm{PM}_i^{j+S/2} \right| > 3$$

When $\mathrm{G}_u(\mathrm{X}) = \mathrm{PM}_i^{j} - \mathrm{PM}_i^{j+S/2} > 3$, the selected path is from state j+S/2, new path metrics are calculated using Eqs (3) and (4) without the inputs of PM_i^{j}, BM1 and BM3.

$$\mathrm{PM}_{i+1}^{2j} = \mathrm{PM}_i^{j+S/2} + \mathrm{BM3} \qquad (3)$$

$$\mathrm{PM}_{i+1}^{2j+1} = \mathrm{PM}_i^{j+S/2} + \mathrm{BM4} \qquad (4)$$

When $\mathrm{G}_l(\mathrm{X}) = \mathrm{PM}_i^{j+S/2} - \mathrm{PM}_i^{j} > 3$, the selected path is from state j, new path metrics are calculated using Eqs (5), (6) without the inputs of $\mathrm{PM}_i^{j+S/2}$, BM2 and BM4.

$$\mathrm{PM}_{i+1}^{2j} = \mathrm{PM}_i^{j} + \mathrm{BM1} \qquad (5)$$

$$\mathrm{PM}_{i+1}^{2j+1} = \mathrm{PM}_i^{j} + \mathrm{BM2} \qquad (6)$$

When $\left| \mathrm{PM}_i^{j} - \mathrm{PM}_i^{j+S/2} \right| \leq 3$, no data can be disabled because all signals are required to compute the out put of ACSU.

If PM_i^{j} and $\mathrm{PM}_i^{j+S/2}$ use a 6 bit path metrics collided A(a5, a4, a3, a2, a1, a0) and B(b5, b4, b3, b2, b1, b0) respectively, then the precomputation conditions G_u (X) and G_l (X) can be calculated by using the following logic expressions.

$$\mathrm{G}_u(\mathrm{X}) = a_5 \overline{b_5} \cdot \overline{b_4 b_3 b_2 b_1}$$

$$\mathrm{G}_l(\mathrm{X}) = b_5 \overline{a_5} \cdot \overline{a_4 a_3 a_2 a_1}$$

According to the G(X), if the probabilities of obtaining a zero and a one are equal, then the probability that $\left| \mathrm{PM}_i^{j} - \mathrm{PM}_i^{j+S/2} \right| > 3$ is a 47% (30/64), under which condition almost half of the path metric calculation is disabled. Also when the load disable signal is asserted, the comparator performs few switching activities because some of its input data are not switched.

IV. Implementation and Results

A 16-state, rate 1/3 VD is designed, using the proposed precomputation logic. The conventional and the proposed VDs are coded in VHDL and synthesized with Synopsys using the TCMS 0.13 um technology library. Fig. 6 shows the results of a gate-level. Here INP refers to original data input to the convolutional encoder, EOP is the output of the encoder, and SOUT is the output of the VD.

The power consumptions of the two architectures are estimated using a gate-level power simulator based on a real delay model. A 1.2V supply and 25MHZ operating frequency are assumed. The results indicate that the proposed ACSU design has an increased by 3% larger area, a 1% lower speed and a 42% lower power than the conventional ACSU design.

V. Conclusions

Low-power architectures for the ACSU of VD were presented. A 3% increase in area, 1% increase in speed and 42% reduction in power consumption were obtained using the proposed architecture.

References

[1] G.D. Forney, Jr., "The Viterbi Algorithm," Proc. IEEE. Vol.61, pp.268-278, Mar. 1973.

[2] A.J. Viterbi, "Error Bounds for Convolutional Codes and Asymptotically Optimum Decoding Algorithms," IEEE Trans. Inform. Theory, vol. IT-13, pp.260-269, April, 1967.

[3] G. Fettweis, L. Thiele, G. Meyr, "Algorithm Transformation for Unlimited Parallelism", *1990*.

[4] Herbert Dawid, Gerhard Fettweis, and Heinrich Meyr, "A CMOS IC for GB Viterbi Decoding: System Design and VLSI Implementation", IEEE Trans on Very Large Scale Intergration (VLSI) Systems, vol. 4, no. 1, Mar, 1996.

[5] K. Sekiet. al., "Very Low Power Consumption Viterbi Decoder LSLC Employing the SST (Scarce State Transition) Scheme for Multimedia Mobile Communications", in IEE electronics Letters, Vol. 30, No. 8, pp.637-639, April, 1991.

[6] Kang and A. N. Wilson Jr., "Low Power Viterbi Decoder for CDMA Mobile terminals," in IEEE Journal of Solid-State Circuits, vol.33, no.3, pp.473-482, March, 1998.

[7] M-H Chan, W-T Lee, M-C Lin and L-G Chen, "IC Design of an Adaptive Viterbi Decoder," IEEE Trans. On Consumer Electronics, vol. 42, pp. 52-61, Feb. 1996.

[8] Xun Liu, Marios C. Papaefthymiou, "Design of a High-Throughput Low Power IS95 Viterbi Decoder," DAC02.

[9] Tobias Gemmeke, Michael Gansen, and Tobias G. Noll, "Implementation of Scalable Power and Area Efficient High-Throughput Viterbi Decoders," in IEEE Journal of Solid-State Circuits, vol.37, no.7, pp.941-948, July, 2002.

[10] Alidina, M., Monteiro, J., Devadas, S., Ghosh, A., Papaefthymiou, M., "Precomputation-based sequential logic optimization for low power," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, , Volume: 2 Issue: 4 , Dec.1994 Page(s): 426 -436

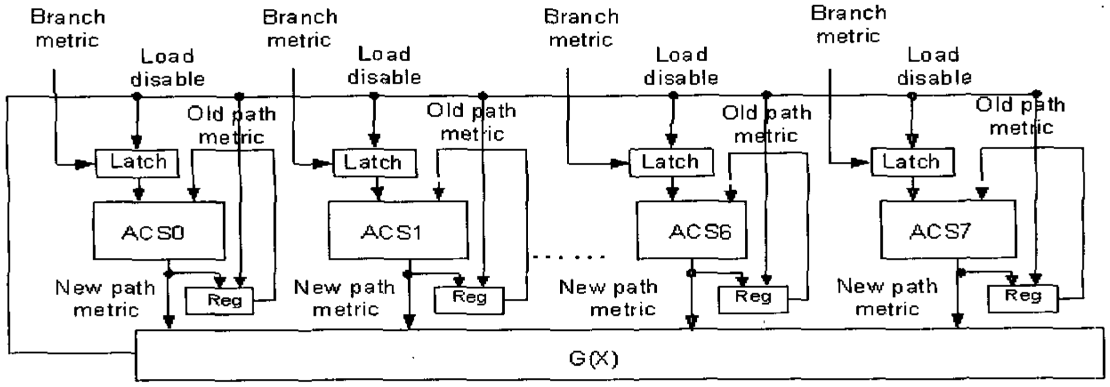

Fig. 5: Structure of proposed Precomputation Scheme

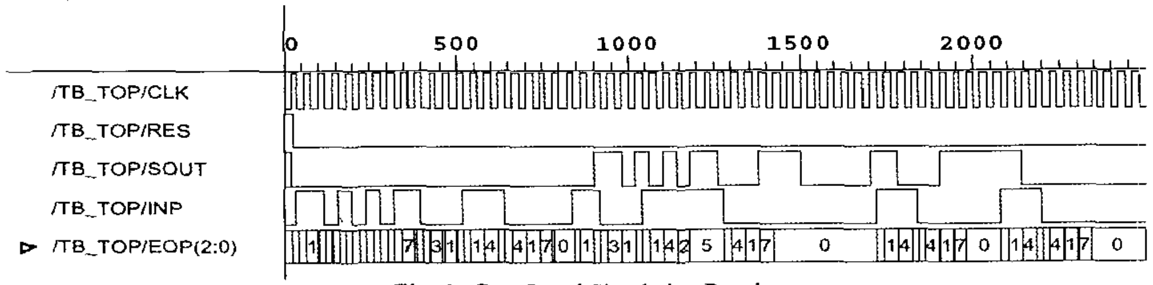

Fig. 6: Gate-Level Simulation Results