Low Power State-Parallel Relaxed Adaptive Viterbi Decoder Design and Implementation

Fei Sun and Tong Zhang
Electrical, Computer and Systems Engineering Department
Rensselaer Polytechnic Institute, USA
Email: sunf@rpi.edu, tzhang@ecse.rpi.edu

***Abstract*— In this paper, we present an algorithm/architecture-level design solution for implementing state-parallel adaptive Viterbi decoders that, compared with their Viterbi counterparts, can achieve significant power savings and modest silicon area reduction, while maintaining almost the same decoding performance and throughput. The effectiveness of the proposed solution has been demonstrated using convolutional codes decoders as test vehicles, where Synopsys tools are used for synthesis, layout, and post-layout power estimation.**

I. Introduction

Adaptive Viterbi algorithm [1], [2], combining the Viterbi algorithm with the principle of *T*-algorithm [3], has a computational activity adaptive to the run-time signal to noise ratio (SNR) and a great potential of realizing significant power savings. As the main difference from the Viterbi algorithm, in the adaptive Viterbi algorithm, the winner path at each trellis state does not necessarily become a survivor path, i.e., only those whose path metrics are better than a global non-survivor purge limit will be feed to the next decoding depth as survivors. The non-survivor purge limit varies from one decoding depth to the next and depends on the metric of the overall best winner at each decoding depth. Due to the serial nature of the *search-the-best-winner* operation, the adaptive Viterbi algorithm cannot fit into a state-parallel decoder structure, hence its power-saving advantage cannot be leveraged by applications demanding very high throughput.

In this work, we developed a *relaxed* adaptive Viterbi decoder that eliminates the search-the-best-winner operation and directly fits into a state-parallel decoder structure. Compared with its state-parallel Viterbi counterpart, a state-parallel relaxed adaptive Viterbi decoder can realize significant power saving and modest silicon area reduction, while maintaining almost the same decoding performance and throughput. Using a 0.13μm CMOS standard cell library and Synopsys tools for synthesis, layout, and post-layout power estimation, we designed relaxed adaptive Viterbi decoders for rate-1/2 convolutional codes with 64, 128, and 256 states, respectively. When operating at 400Mbps, compared with their Viterbi counterparts, the relaxed adaptive Viterbi decoders realize up to 62.7% of power savings on the decoding computation[1], and up to 70.5% of overall decoder power savings if the register-exchange approach is used to implement the trace-back unit.

[1]The total power consumption of a convolutional code decoder contains two parts, including (1) power consumed by *decoding computation* including ACS (add-compare-select) computation, branch metric computation, etc., and (2) power consumed by *decoder output generation* that is carried out by a trace-back unit.

II. Relaxed Adaptive Viterbi Decoder

A. Decoding Algorithm

Fig. 1(a) shows the recursive decoding data flow diagram of an adaptive Viterbi algorithm, which adds two functional blocks, including the Best Winner Search and Non-Survivor Purge, into the original Viterbi algorithm. At each decoding depth, after each ACS unit determines a local winner, the Best Winner Search block finds the winner having the best (minimum) path metric, denoted as $\Gamma_B^{(n)}$, and the Non-Survivor Purge block deletes the local winners whose metric $\Gamma^{(n)} \geq \Gamma_B^{(n)} + T$ and feeds the others as survivors to the next decoding depth, where T is a fixed positive number. The value of $\Gamma_B^{(n)} + T$ is called non-survivor purge limit. Due to the serial essence of the search operation, the Best Winner Search block inevitably incurs a large delay in the recursive decoding datapath, which makes the adaptive Viterbi algorithm not suitable for a state-parallel decoder structure.

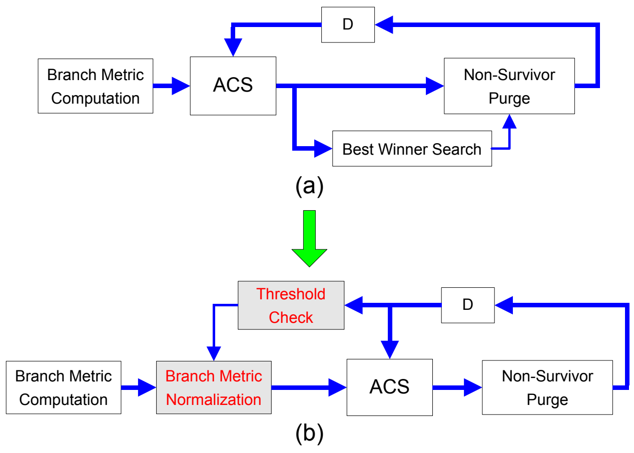

Fig. 1. The recursive decoding data flow diagrams of (a) original adaptive Viterbi algorithm and (b) proposed relaxed adaptive Viterbi algorithm.

In this work, we developed a method to eliminate the search-the-best-winner operation and hence enable the high-throughput state-parallel adaptive Viterbi decoder implementation. The basic idea is to *dynamically normalize* the branch metrics in such a way that the metric of the overall best winner

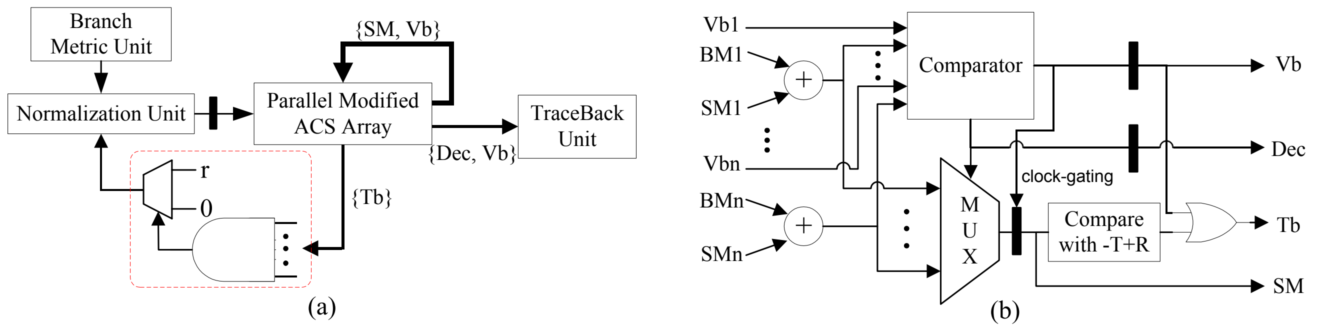

Fig. 2. Architectures of (a) the state-parallel relaxed adaptive Viterbi decoder, and (b) a modified ACS unit.

is almost always very close to $-T$, which means the non-survivor purge limit is almost always very close to zero. As a relaxation, we simply fix the non-survivor purge limit as zero at each decoding depth, which directly eliminates the search-the-best-winner throughput bottleneck in the recursive decoding datapath. The resulted algorithm is referred to as relaxed adaptive Viterbi algorithm. The branch metric dynamic normalization is realized by the two shaded functional blocks as shown in Fig.1(b), which are described as follows:

- *Threshold Check*: It checks whether at least one survivor has a metric less than $-T + R$, where R is a positive number that is much less than T. If yes, it outputs a zero, otherwise, it outputs r, where r is a positive number that is less than or equal to R.
- *Branch Metric Normalization*: At each depth, it finds the best (minimum) branch metric, denoted as $BM_B^{(n)}$. Given the input $d^{(n)}$ from the Threshold Check, which is either zero or r, it subtracts $BM_B^{(n)} + d^{(n)}$ from each branch metric.

If at least one survivor has a metric less than $-T + R$ (i.e., the metric of the best survivor is very close to $-T$ since R is much less than T), the normalized branch metrics will be non-negative with the minimum value of zero, and the path metrics will monotonically increase. If non of the survivors has a metric less than $-T + R$ (i.e., the metric of the best survivor is not very close to $-T$), we bias the branch metric normalization by r to push the path metrics toward $-T$. With appropriate selection of R and r, we can dynamically adjust the best metric almost always very close $-T$.

B. State-Parallel Decoder Architecture

The above relaxed adaptive Viterbi algorithm can be directly mapped onto a state-parallel decoder hardware architecture, as illustrated in Fig. 2(a), that is very similar to that of a state-parallel Viterbi decoder. Notice that the search-the-best-branch-metric operation in the normalization unit will not lead to a throughput bottleneck because (i) the number of branch metrics is typically very small, and (ii) it locates outside the recursive datapath and hence can be directly pipelined if necessary.

The Threshold Check functional block in the above relaxed Viterbi algorithm is realized in a distributed manner, i.e., the compare-with-($-T + R$) is realized by each individual modified ACS unit, and the pass/fail decisions from all the modified ACS units are AND together to determine whether a zero or r should be sent to the normalization unit, as illustrated in Fig. 2(a). The architecture of a modified ACS unit is shown in Fig. 2(b), which generates four outputs, including (1) survivor path metric SM, (2) decision bits Dec, (3) validity bit Vb (Vb=0 indicates that a survivor is generated from the present trellis state), and (4) threshold check result Tb (Tb=0 indicates that the corresponding path metric is less than $-T + R$). As demonstrated in the implementation examples described in the next section, compared with its state-parallel Viterbi decoder counterpart, a state-parallel relaxed adaptive Viterbi decoder can achieve:

- *Significant power saving*: The power saving is gained from both decoding computation and decoder output generation:
  1) The power saving on decoding computation is realized by clock-gating the path metric output SM of non-survivors (as shown in Fig. 2(b)), which leads to a largely reduced switching activity in the subsequent ACS computation due to the significantly reduced number of survivors.
  2) The power saving on decoder output generation depends on the implementation of trace-back unit: (a) If we use the register-exchange approach, we can simply clock-gate the registers associated with non-survivors; (b) If we use the memory-based approach, we may force the decision bits of non-survivors to be a constant to reduce the switching activity in memory storage (i.e., very likely the constant will be written into the same memory location successively). Notice that, although memory-based approach tends to consume less power, in order to match the high throughput of state-parallel decoding computation, it requires complex memory structure design and/or multi-frequency/phase clock signals, and may incur certain decoding performance degradation.
- *Modest silicon area reduction*: Contrary to the first impression that the relaxed adaptive Viterbi decoder may occupy larger area due to the extra functional blocks, the

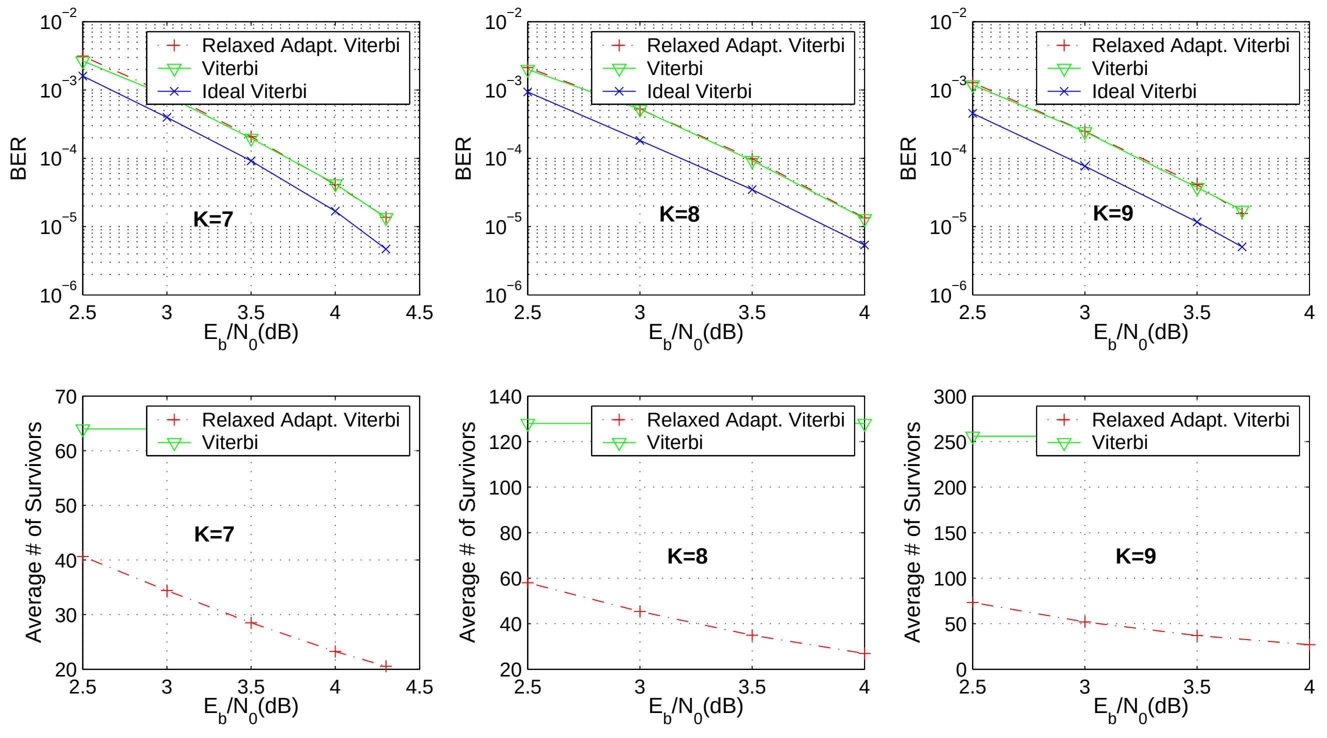

Fig. 3. Simulated BER and average number of survivors.

branch metric normalization can help to reduce the finite word-length of the path metrics, leading to an overall modestly reduced silicon area (as demonstrated in the next section).

Moreover, as demonstrated in the next section, the relaxed adaptive Viterbi decoders can achieve almost the same decoding performance (with appropriate selection of T, R, and r) and decoding throughput, compared with their Viterbi counterparts. The reason for the latter can be briefly explained as follows:

1) The reduced finite word-length of path metric can compensate the latency overhead incurred by the slightly more complex operation in the ACS computation;
2) The threshold check operation will not incur a throughput bottleneck because the involved operation is very simple and it can be directly pipelined if necessary (in the design examples described later, the overall decoder critical path always lies in ACS computation).

Finally, we note that, due to their very similar architectures, the circuit-level design/optimization techniques ever developed for state-parallel Viterbi decoders can be equally applied to state-parallel relaxed adaptive Viterbi decoders.

III. Design Examples

For the purpose of demonstration, we designed state-parallel relaxed adaptive Viterbi decoders for rate-1/2 convolutional codes with the constraint lengths K of 9, 8 and 7 (corresponding to the trellises with 256, 128 and 64 states, respectively). The code generators are (561, 753) for K=9, (247, 371) for K=8, and (133, 171) for K=7. For comparison, we also designed the state-parallel Viterbi counterparts. The trace-back units are realized as register-exchange with majority vote (as discussed in [4], the majority vote can be slightly more power-efficient for register-exchange trace-back unit). We use a 0.13μm CMOS standard cell library and Synopsys tool sets are used for synthesis (Design Compiler), layout (Astro), and post-layout power estimation (Prime Power). The design parameters are outlined in Table I. We note that the branch metric normalization in relaxed adaptive Viterbi decoders help to reduce the finite word-length of the path metrics.

TABLE I
Design Parameters

	Viterbi	Relaxed Adaptive Viterbi
Decision Length	55 (K=9), 46 (K=8), 40 (K=7)	
Soft Input	3 bits	
Path metric[2]	8 bits	6 bits
Others	N/A	$T = 24$, $R = r = 3$

[2]In the Viterbi decoder, the computation of path metric is based on modulo arithmetic, where the configuration of 3-bit soft input and 8-bit path metric is widely used in open literature, e.g., see [5].

Assuming these convolutional codes are modulated by BPSK (binary phase shift keying) and transmitted over an AWGN (additive white Gaussian noise) channel, Fig. 3 shows the simulated BER (bit error rate) and average number of survivors of the decoders. In each case, we also show the performance of ideal Viterbi decoding (i.e., floating point precision and infinite decision length). The relaxed adaptive

Viterbi decoders can achieve almost the same decoding performance as their Viterbi counterparts, while incurring much less number of survivors.

During synthesis and layout, we set the target throughput as 400Mbps (with the power supply of 1.1V) and the results show that all the decoders can meet this target with similar timing slack. The post-layout silicon area and power estimation (when the decoders run at 400Mbps) results are listed in Tables II, III and IV for K=7, 8, and 9, respectively. They clearly show the effectiveness of the proposed relaxed adaptive Viterbi decoders, where the power-saving efficiency improves as the constraint length increases.

TABLE II

POST-LAYOUT AREA AND POWER ESTIMATION FOR K=7

		Viterbi	Relaxed Adaptive Viterbi	
Core Area (mm^2)		0.204	0.196 (-3.9%)	
Power (mW) @ 400Mbps	SNR	4dB	3dB	4dB
	Decoding Computation	48.1	27.3 (-43.2%)	25.2 (-47.7%)
	Output Generation (reg. exchange)	57.4	56.4	45.0
	Total	105.5	83.7 (-20.6%)	70.2 (-33.5%)

[3]The number in parenthesis represents the percentage of decrease against the Viterbi decoder.

TABLE III

POST-LAYOUT AREA AND POWER ESTIMATION FOR K=8

		Viterbi	Relaxed Adaptive Viterbi	
Core Area (mm^2)		0.463	0.424 (-8.4%)	
Power (mW) @ 400Mbps	SNR	4dB	3dB	4dB
	Decoding Computation	103.2	49.6 (-51.9%)	43.5 (-57.9%)
	Output Generation (reg. exchange)	123.7	76.7	53.1
	Total	226.9	126.3 (-44.3%)	96.6 (-57.4%)

TABLE IV

POST-LAYOUT AREA AND POWER ESTIMATION FOR K=9

		Viterbi	Relaxed Adaptive Viterbi	
Core Area (mm^2)		1.10	1.02 (-7.3%)	
Power (mW) @ 400Mbps	SNR	4dB	3dB	4dB
	Decoding Computation	221.5	94.8 (-57.2%)	82.6 (-62.7%)
	Output Generation (reg. exchange)	344.4	139.1	84.1
	Total	565.9	233.9 (-58.7%)	166.7 (-70.5%)

IV. CONCLUSIONS

This paper presents the algorithm design and VLSI implementation of state-parallel relaxed adaptive Viterbi decoders for high-throughput and low-power trellis decoding. Supported with detailed synthesis, layout, and post-layout power estimation results, the key feature that distinguishes this design solution from the existing work is that it can realize significant power saving and modest silicon area reduction without degradation on the decoding performance and throughput, compared with state-parallel Viterbi decoders.

REFERENCES

[1] R. Henning and C. Chakrabarti, "An approach for adaptively approximating the Viterbi algorithm to reduce power consumption while decoding convolutional codes," *Transactions on Signal Processing*, vol. 52, pp. 1443–1451, May 2004.

[2] M.-H. Chan, W.-T Lee, M.-C. Lin, and L.-G. Chen, "IC design of an adaptive Viterbi decoder," *IEEE Transactions on Consumer Electronics*, vol. 42, pp. 52–62, Feb. 1996.

[3] S. J. Simmons, "Breadth-first trellis decoding with adaptive effort," *IEEE Transactions on Communications*, vol. 38, pp. 3–12, Jan. 1990.

[4] M. Petrov A.M. Obeid, A. Garcia and M. Glesner, "A multi-path high speed viterbi decoder," in *Proceedings of the 2003 10th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Dec 2003, pp. 1160–1163.

[5] Y.-N. Chang, H. Suzuki, and K. K. Parhi, "A 2-Mb/s 256-state 10-mW rate-1/3 Viterbi decoder," *IEEE Journal of Solid-State Circuits*, vol. 35, pp. 826–834, June 2000.