ERROR-RESILIENT LOW-POWER VITERBI DECODERS VIA STATE CLUSTERING

Rami A. Abdallah and Naresh R. Shanbhag

Coordinated Science Laboratory/ECE Department
University of Illinois at Urbana-Champaign
1308 W Main St., Urbana, IL, USA, 61801.

ABSTRACT

Low-power Viterbi decoder (VD) architectures based on the principle of error-resiliency are presented in this paper. Power reduction in the add-compare-select units (ACSUs) of a VD is achieved by either overscaling the supply voltage (*voltage overscaling* (VOS)) or designing at the nominal process corner and supply voltage (*average-case design*). In either case, the data-dependent timing errors which occur whenever a critical path is excited, are corrected via the application of *algorithmic noise-tolerance* (ANT). The concept of *state clustering* is employed to develop efficient estimators for error-correction. Power savings achieved in the presence of VOS and process variations are 71% and 62%, respectively, at a loss of $0.8\,\mathrm{dB}$ and $0.6\,\mathrm{dB}$ in coding gain in a IBM 130nm CMOS process.

Index Terms— Viterbi Decoder, Algorithmic Noise Tolerance, Error Resiliency , Process Variations, Voltage overscaling.

1. INTRODUCTION

Viterbi decoders (VDs) in high data-rate applications have significant power consumption. For example, an IEEE 802.11a/g WLAN compliant VD consumes $35\,\%$ of the total receiver power [1] and constitutes $76\,\%$ of the total digital processing complexity (in ops/s) [2]. Hence, low-power VD architectures are of great interest and are in fact a well-studied subject. For example, energy-efficiency has been achieved by reducing the number of states [3] or the number of trellis paths [4, 5] at the expense of increased BER and/or reduced throughput.

None of the above referenced works addresses the issue of robustness in the presence of process, voltage, and temperature (PVT) variations. This requires a reliance on worst-case design and leads to high power consumption. The concept of error-resiliency offers an elegant solution to this problem and is considered a promising design philosophy for the nanoscale era. Error-resilient designs are implemented at the nominal PVT corner to save power. Logic errors, that may occur when the PVT corner is worse than nominal, are corrected via architectural and algorithmic techniques. The concept of error-resiliency in the presence of *voltage overscaling* (VOS) was first proposed in [6] and referred to as *algorithmic noise-tolerance* (ANT). More recently, a logic level technique [7] for combating timing errors due to VOS was presented.

In this paper, we present ANT-based energy-efficient and robust architectures for the add-compare-select unit (ACSU), a key computational kernel in the VD. Section 2 presents background material. Issues in designing error-resilient Viterbi architectures are discussed in section 3. Section 4 introduces idea of *state clustering* for error-resilient ACSUs. Simulation results demonstrating the BER performance and power savings under voltage and process variations are shown in section 5.

The authors acknowledge the support of Texas Instruments and the Gigascale System Research Center (GSRC), one of five research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation program.

2. BACKGROUND

2.1. Viterbi Algorithm

The Viterbi algorithm is an efficient prcedure for solving maximum likelihood sequence estimation (MLSE) problems such as decoding of convolutional codes. The encoding process can be represented by a time indexed trellis as shown in Fig. 1(a). The trellis has four encoder states with each branch being characterized by a branch metric (BM) that indicates the distance (usually Euclidean) between the received samples and the expected codeword. Each path through the trellis has a path metric (PM). The PM is inversely proportional to the log likelihood probability of that path. The Viterbi algorithm recursively finds the path with the minimum PM for each state. For the trellis in Fig. 1(a), there are two paths entering each state and the PMs for each state are updated as follows:

$$PM_k^{(n+1)} = \min\left(PM_i^{(n)} + BM_i^{(n)}, PM_j^{(n)} + BM_j^{(n)}\right) \quad (1)$$

where two path metrics ($PM_i^{(n)}$ and $PM_j^{(n)}$) and two branch metrics ($BM_i^{(n)}$ and $BM_j^{(n)}$) are employed to generate an updated value $PM_k^{(n+1)}$. Equation (1) is implemented in an ACSU. The trellis processing can be done one trellis section at a time (radix-2 in Fig. 1(a)) or two at a time (radix-4 in Fig. 1(b)) or more. This is referred to as higher radix computation. In general radix-k processing, where k is a power of 2, $log_2(k)$ trellis sections are processed in one time-step.

A generic VD architecture, assuming 4-b BMs and 8-b PMs, is shown in Fig. 2. The PMs are quantized using two's complement representation in order to ensure correct operation in the presence of an overflow [8]. The branch metric unit (BMU) generates BMs for all the edges. The ACSU recursively computes the PM of each state according to (1). The survivor memory unit (SMU) keeps track of the survivor path of each state. A state-parallel ACSU is commonly employed for high data-rate applications where the PMs are computed in parallel. In this paper, we target the design of an ACSU for a 128-states rate-1/2 convolutional code. For the sake of brevity, in the following, time index n will not be listed. Instead, the precision of various signals will be referred to explicitly, e.g. $BM_i[3:0]$ is the BM added to $PM_i[7:0]$ in (1).

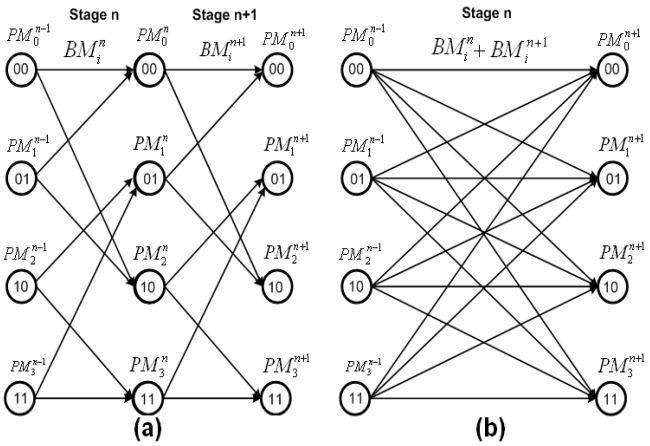

Fig. 1. Trellis of a 4-states rate 1/2 convolutional code: (a) radix-2 trellis, and (b) a radix-4 trellis.

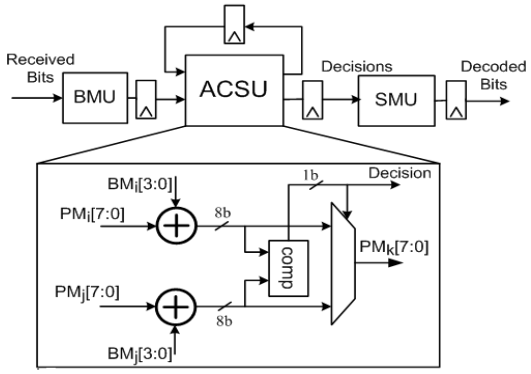

Fig. 2. A generic Viterbi decoder architecture.

2.2. Algorithmic Noise Tolerance

An ANT-based system (see Fig. 3) consists of a main block that computes correctly most of the time but makes PVT variation induced errors. For example, in VOS, the supply voltage is reduced below the critical value needed to avoid timing errors. Thus,

$$y_a[n] = y_o[n] + \eta[n], \tag{2}$$

where $y_a[n]$ is the main block output, $y_o[n]$ is the error-free output, and $\eta[n]$ is the signal representing timing errors due to PVT variations. These errors are corrected by an estimator which produces a statistical replica $y_e[n]$ of the error-free main block output $y_o[n]$. The estimator is designed using statistical signal processing techniques when the main block is a DSP kernel. The challenge in ANT-based systems is to discover a low-complexity estimator. Such an estimator can be designed to be error-free. A simple decision block detects and corrects errors in the main block output as follows:

$$\hat{y}\,[n] = \begin{cases} y_a\,[n] & if\ |y_a\,[n] - y_e\,[n]| < T_h \\ y_e\,[n] & otherwise \end{cases} \tag{3}$$

where T_h is a predefined threshold, and $\hat{y}[n]$ is the corrected final output shown in Fig. 3.

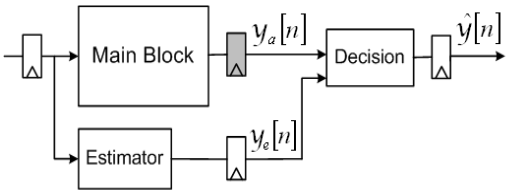

Fig. 3. ANT for non-recursive architectures. The shaded latch indicates the location of timing errors.

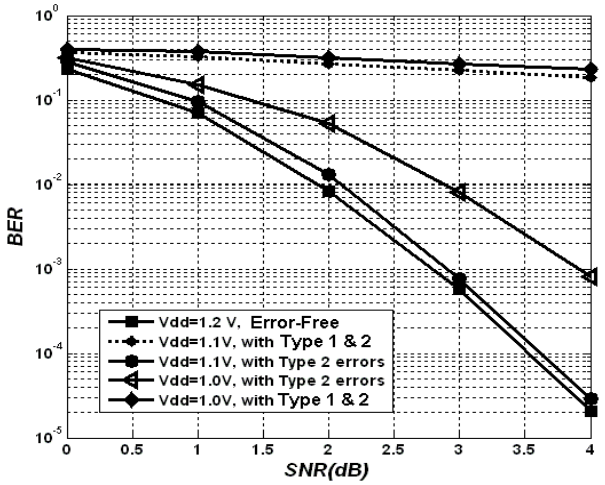

Fig. 4. BER with ACSU subject to different sources of timing errors.

3. ERROR RESILIENCY IN VITERBI DECODERS

3.1. Impact of Timing Errors on VD BER Performance

We first characterized the worst case delays of basic gates employed in the ACSU at various supply voltages using HSPICE in an IBM 130nm CMOS process. An RTL simulation of the VD is then carried out with individual gate delays obtained from the circuit level characterization mentioned above. The clock frequency is chosen to meet a data rate of $590\,\mathrm{Mb/s}$ at $1.2\,\mathrm{V}$. An additive white Gaussian channel with binary phase-shift keying (BPSK) modulation is assumed. Figure 4 shows the BER curves obtained by reducing the supply voltage from $1.2\,\mathrm{V}$ to $1.1\,\mathrm{V}$ and $1\,\mathrm{V}$. Note the large increase in BER due to VOS induced timing errors.

Timing errors in ACSU can occur during the add or the compare step in the ACSU. Thus, one can classify timing errors into two types:

- Type 1 errors result when the MSBs in the BM adder fail to compute, and the resulting incorrect PM gets selected. In this case, the BER is impacted severely because the PM value due to MSB errors may flip from being large and positive to small and negative. Since VD searches for the smallest PM, the incorrect PM is propagated across multiple trellis stages leading to multiple incorrect decisions.
- Type 2 errors occur because of timing errors in the compare-select block. This results in the wrong (larger) PM getting selected. In practice, the competing PMs for a given state are similar when there are no adder errors. Thus, Type 2 errors are benign as compared to Type 1 errors, as seen in Fig. 4

at Vdd = 1.1V. A higher frequency of Type 2 errors can also have a large impact on the BER as decision errors occur more regularly on chosen paths (see Fig. 4 at Vdd = 1.0V with Type 2 errors only).

3.2. Algorithmic Noise-Tolerance in Recursive Architectures

The recursive architectures present two main challenges for the application of error-resiliency. First, recursive architectures suffer from *error propagation* where the residual error at time instant n can impact future outputs (see Fig. 5(a)). Thus, highly effective estimators are needed to keep the residual error within bounds. Second, the introduction of the decision block increases the critical path delay (see Fig. 5(a)) thereby generating timing violations in the decision block which are hard to correct because of its non-linear nature. In addition, an ACSU is intrinsically non-linear and thus making it hard to design efficient estimators.

Retiming can help by placing the latch at the output of a block (such as C in Fig. 5(b)) with easily to correct errors, or before the decision block to avoid decision block errors as shown in Fig. 5(c). Furthermore, one can employ low-complexity pipelining techniques such as *block interleaved pipelining* (BIP) [9] to remove the decision block from the critical path all together as shown in Fig. 5(d).

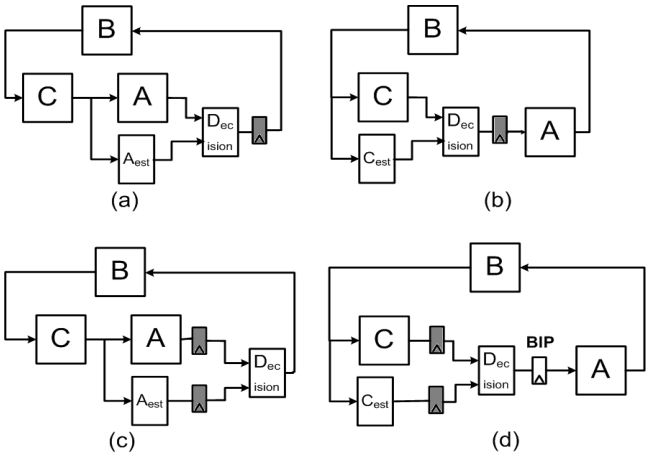

Fig. 5. ANT for recursive architectures with shaded latches indicating the location of timing errors: a) timing errors impact hard to correct decision block, b) retiming to ease error-correction, c) retiming to prevent errors in decision block, and d) block interleaved pipelining (BIP) to eliminate decision block from the critical path.

3.3. Algorithmic Noise-Tolerance in ACSU

The ACSU is implemented using a LSB-first BM adder followed by a LSB-first comparator (subtractor) (see Fig. 2). The two operations proceed in parallel so that the delay is dominated by a single 8-b adder (see Fig. 6(a)). A low-complexity estimator can be designed for the BM adder as it is a linear block. Determining an effective estimator for the comparator is hard as it is a non-linear block. We retime the feedback register across the comparator as illustrated in Fig. 6(b) to avoid timing errors at its output. However, this step doubles the critical path delay since the comparator involves a selection step that cannot be completed in parallel with the adder. Alternatively, since a limited number of Type-2 errors can be tolerated, we propose to use a relaxed comparator (see Fig. 6(c)) to avoid timing-based decision errors in all ACSUs, where the comparison is done in two parallel steps with reduced delay at the expense of a small loss in decoding performance. In addition, we introduce a fast ACSU (**FACSU**) (see Fig. 6(d)) using the relaxed comparator where error-free operation is feasible at lower supply voltages due to reduced critical path delay. In this architecture, the BM adder and comparator steps are implemented in a parallel manner. The **FACSU** will be used as an estimator for the ANT-based scheme proposed later and is discussed next.

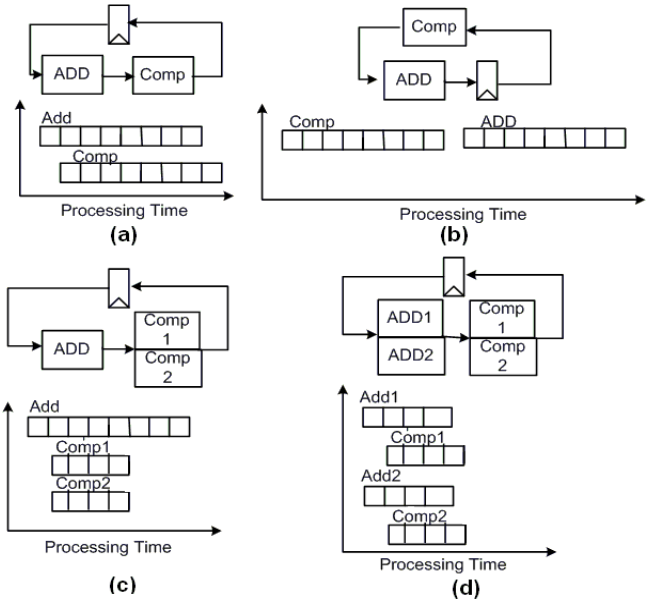

Fig. 6. Throughput and latency. (a) conventional ACSU, (b) retimed ACSU, (c) relaxed ACSU, and (d) fast ACSU (**FACSU**).

3.4. Fast ACSU Architecture

The **FACSU** shown in Fig. 7 employs the concepts of *delayed carry* and *relaxed comparator*. Here, the computation is partitioned into an LSB and an MSB section with the delay of each section being approximately half of the original ACSU delay. Recall that the BMs and the PMs are quantized to 4-b and 8-b, respectively. During BM addition, a concept similar to carry-save addition is applied, i.e., the carry from the LSB to the MSB section is saved and processed only in the next clock cycle. Therefore, the PM is a 9-b value with the additional bit representing the delayed carry. This reduces the BM adder delay by a factor of 2. The **FACSU** uses a relaxed comparator as explained next.

3.4.1. Relaxed Comparison

The relaxed comparator is partitioned into an MSB and an LSB comparator. The MSB comparator is a 4-b comparator that compares the outputs of the MSB adders ($PM_i[7:4] + C_i(4)$ and $PM_j[7:4] + C_i(4)$). The LSB comparator is a 5-b comparator that compares the 5-b result of adding the BMs to the 4-b LSBs of the input PMs ($PM_i[3:0]$ and $PM_j[3:0]$) including the output carries. The LSB comparator decision is considered only when the MSB part of the updated PMs are equal. The relaxed comparator has

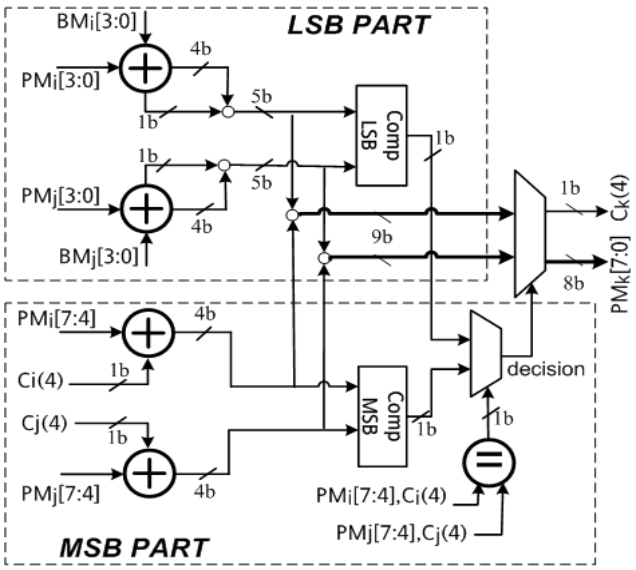

Fig. 7. **FACSU** with delayed carry and relaxed comparator.

a critical path delay equal to that of a 5-bit adder delay though it will make errors once in a while as discussed next.

The error in splitting the comparator to an MSB and LSB part is that we are ignoring the propagation of carries into the MSB parts of the updated PMs and which will be done during the next clock cycle. Therefore, the relaxed comparator makes incorrect decisions only when the MSB parts of the updated PMs differ by unity and the propagated (saved) carries from the LSB part makes them equal in the next clock cycle. Under this setting, the relaxed comparator decision is based on the the 4-b MSBs of the updated PMs before the carry propagation. However, the correct decision depends on the 4-b LSBs of the updated PMs since the propagated carries will make the 4-b MSBs of the updated PMs equal. These wrong decisions have a small effect on decoding performance because the updated PMs will differ only in their 4-b LSBs. Thus, the two paths under consideration have a close likelihood probability as their PM difference is less than 16. Simulation results in section 5 shows only a 0.15 dB loss in coding gain due to relaxed comparator.

4. PM ESTIMATION USING STATE CLUSTERING

We cluster the ACSUs that have the same set of previous PMs into a single cluster. In this way, the updated PMs (the selected PMs after BM addition) for all ACSUs in each group will be close to each other since they share the same input PMs and may only differ in the added BMs. Therefore, we can use a single estimator to correct for the wrong PMs in each cluster. Estimator design will be discussed in the next subsection. In 128 states rate-1/2 trellis, 7 bits are needed to represent each state and each state has two possible transitions depending on the input information bit being a zero or a one. Therefore, the 128 states can be partitioned into 64 disjoint 2-state clusters of the form shown in Fig. 8 and 64 PM estimators are needed in total (One for each cluster).

The number of ACSUs per cluster can be increased and consequently the number of estimators needed in the VD can be decreased by employing radix-k processing where each state will have k possible transitions. For example, the 128 states in a rate-1/2 radix-4

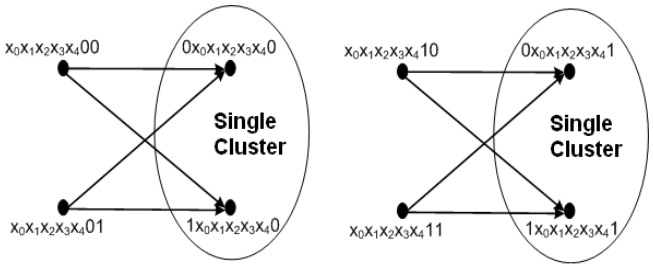

Fig. 8. State clustering for 128-states rate-1/2 code with 2 states per cluster.

trellis can be divided into 32 disjoint 4-state clusters requiring 32 PM estimators.

4.1. PM Estimation Techniques

Two possible PM estimation schemes are presented to correct for BM addition errors inside each cluster. In the first, the **FACSU** is employed for one of the states in the cluster to provide a PM estimate for the other error-prone ACSUs in the cluster. This technique is referred to as **SC-FACSU**. In the second, a redundant state is added to each cluster and is referred to as **SCS-FACSU**. A **FACSU** is used for this redundant state in order for it to operate error-free at lower supply voltages. The redundant state shares the same set of previous states as the others in the cluster (see Fig. 9). The codeword or BM for the branch, joining any state X to the redundant state, is chosen to be equidistant from the codewords on the branches joining state X to the rest of the states in the cluster. Therefore, the estimated PM will be closer to the correct PM than in the previous scheme. In higher radix processing, it may not be possible to have codewords equidistant from the rest of codewords in certain clusters. In this case, codewords for the incoming branches of the redundant state are assigned to be as equidistant as possible.

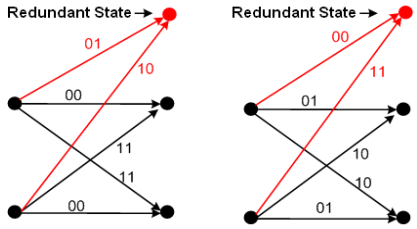

Fig. 9. State clustering using a redundant state.

4.2. Error-resilient Architecture using State Clustering

The relaxed comparator is employed in all ACSUs to avoid timing errors in the compare-select block. Architectures for **SC-FACSU** and **SCS-FACSU** based on radix 2 (two states X and Y per cluster) are presented in Fig. 10 (a) and (b) respectively. In **SC-FACSU**, state X uses **FACSU** architecture and is error free. State Y uses regular ACSU with a relaxed comparison. ANT-decision block corrects the output of State Y by using the PM estimate provided by state X. Timing errors in Y will occur in the MSBs of the PM due to the LSB-first BM addition resulting in a large deviation from the estimate. Therefore, ANT-decision block can detect an error by computing the

difference between the MSBs of the PMs of state Y ($PM_{ye}[7:4]$) and its estimate from X ($PM_x[7:4]$). If the absolute value of this difference is greater than unity, then an error is detected and the output PM of state Y (PM_y) is set to the estimated PM (PM_x), otherwise it is set to the output PM, PM_{ye}. In **SCS-FACSU**, the estimate is provided by the redundant state. Similar architectures can be designed for higher radix computation. Complexity estimate shows 22% and 90% increase in gate complexity for **SC-FACSU** and **SCS-FACSU** respectively over conventional ACSU with 2-state clusters.

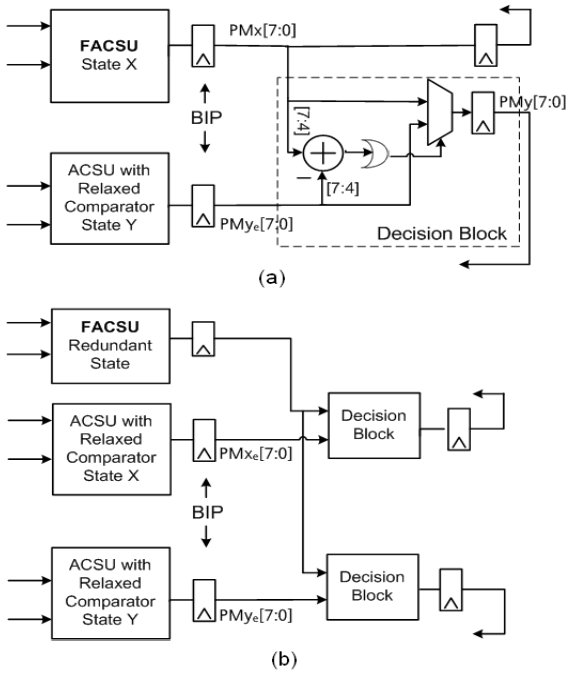

Fig. 10. State clustered architectures with two states X and Y: (a) **SC-FACSU** architecture, and (b) **SCS-FACSU** architecture.

5. SIMULATIONS AND RESULTS

This section describes the BER performance and power savings achieved by **SC-FACSU** and **SCS-FACSU** over a conventional ACSU.

5.1. Voltage Overscaling

The simulation set-up for obtaining results under VOS was described in section 3.1. Figure 11 shows the BER curves obtained by reducing the supply voltage from $1.2\,\mathrm{V}$ to $0.8\,\mathrm{V}$ with 2-state clusters. As the supply voltage is reduced from $1.2\,\mathrm{V}$ to $1.1\,\mathrm{V}$, the conventional ACSU BER increases dramatically. At $0.9\,\mathrm{V}$, **SC-ACSU** and **SCS-ACSU** show only $0.8\,\mathrm{dB}$ and $0.35\,\mathrm{dB}$ loss in the coding gain respectively at a BER of 3×10^{-4} with approximate 667X and 2000X BER improvement over conventional ACSU. Note that at $1.2V$ **SC-ACSU** shows only a loss of $0.15\,\mathrm{dB}$ due to relaxed comparison errors. Reduction of voltage beyond $0.9\,\mathrm{V}$ leads to a noticeably degraded performance (see the BER curve for $V_{dd} = 0.8\,\mathrm{V}$) due to increased frequency of VOS errors. Similarly, Fig. 12 shows the BER curves with 4-state clusters. **SC-FACSU** and **SCS-FACSU** show only a loss of $0.7\,\mathrm{dB}$ and $0.25\,\mathrm{dB}$ in coding gain at the same BER with 956X and 2442X BER improvement.

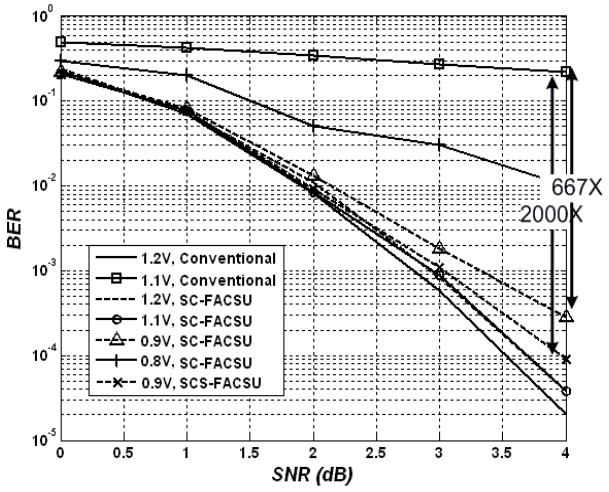

Fig. 11. BER of a conventional ACSU, **SC-FACSU**, and **SCS-FACSU** at different supply voltages using radix-2 computation. Note that curves for $1.2V$ **SC-FACSU** and $1.1V$ **SC-FACSU** show small deviations.

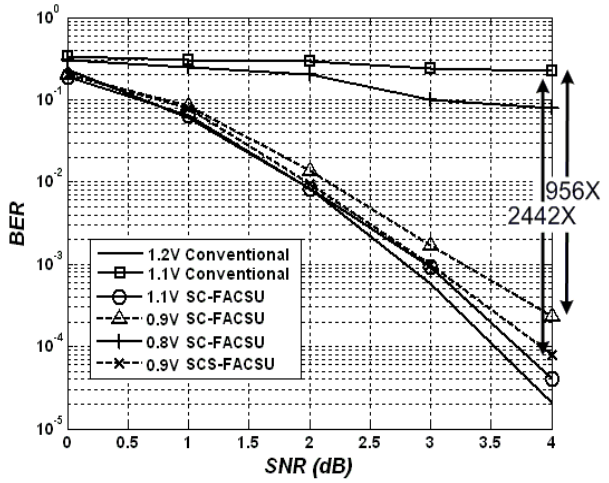

Fig. 12. BER of a conventional ACSU, **SC-FACSU**, and **SCS-FACSU** at different supply voltages using radix-4 computation.

5.2. Process Variations

Process variations are classified into within-die (WID) and die-to-die (D2D) variations. Delay distribution of various gates found in the ACSU were obtained via Monte Carlo simulations in IBM 130nm CMOS process at the 3σ slow process corner with WID variations enabled. These delay distributions are sampled to obtain different instances of the ACSUs. These instances are randomly sampled and simulated at the RTL level to generate different BER curves. The clock frequency is determined by the data-rate and hence is kept fixed at the nominal process corner.

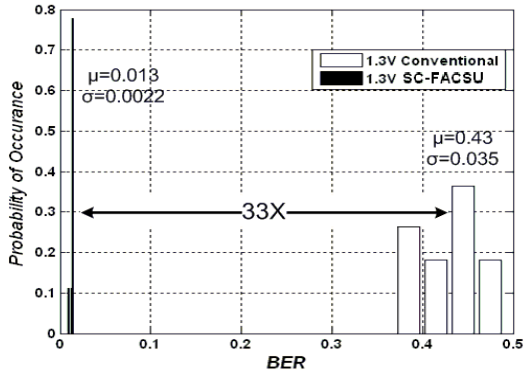

Fig. 13. BER distribution with radix-2 computation at a 3σ slow corner with WID variations and SNR=$2\,\mathrm{dB}$.

Figure 13 shows BER distribution due to process variations at $2\,\mathrm{dB}$ signal-to-noise ratio (SNR) for **SC-FACSU** with two states per group. Conventional ACSU suffers from a large increase in BER under process variations. Adaptive supply voltage and/or adaptive body bias [10] need to be applied to keep its delay fluctuations at the 3σ slow process corner within the operating clock frequency. On the other hand, **SC-FACSU** can operate at the 3σ slow process corner with 33X average BER improvement if voltage is slightly increased from $1.2\,\mathrm{V}$ to $1.3\,\mathrm{V}$. BER simulations under process variations show only a $0.6\,\mathrm{dB}$ loss in the average coding gain at a BER of 3×10^{-4} using **SC-FACSU**. BER statistics under process variations at $2\,\mathrm{dB}$ SNR for **SC-FACSU** and **SCS-FACSU** with different radix processing are reported in Table 1.

Table 1. BER at 3σ slow process corner with WID variations at $2\,\mathrm{dB}$ SNR

	Radix-2		Radix-4	
	μ	σ	μ	σ
Conventional ACSU	0.43	0.035	0.45	0.032
SC-FACSU	0.013	0.0022	0.012	0.0020
SCS-FACSU	0.011	0.0020	0.009	0.0019

5.3. Power Savings

In 3GPP applications, ACSUs constitute around 42% of the total decoder power [11]. In this work, power reduction was achieved at the level of the ACSUs only. To estimate this saving, HSPICE simulations are carried out using a single ACSU with an input test vector of size 50 and clock frequency fixed to meet the target data rate. Results are reported in Fig. 14. At the nominal process corner with VOS and with 2-state clusters, **SC-FACSU** and **SCS-FACSU** achieve power savings of 71% and 42%, respectively, and power savings of 56% and 41%, respectively, with 4-state clusters. Under process variations, the power savings achieved by **SC-FACSU** and **SCS-FACSU** are 62% and 27%, respectively, with 2-state clusters, and 44% and 23% with 4-state clusters. We see that **SCS-FACSU** has better performance than **SC-FACSU** but consumes more power due to the redundant ACSU. We expect the power savings to increase with higher radix processing for **SCS-FACSU** since estimation overhead due to redundant state will decrease as the cluster size increases.

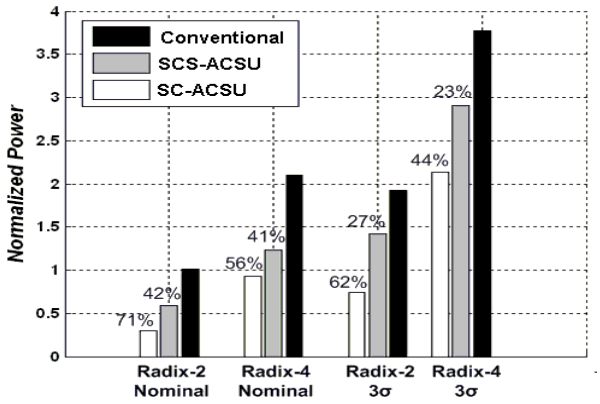

Fig. 14. Average power consumption per trellis state under VOS and process variations.

6. REFERENCES

[1] B. Bougard, et al.," Energy-scalability enhancement of wireless local area network transceivers," in *Proc. of the IEEE Workshop on Signal Processing Advances in Wireless Communication*, Lisboa, Portugal, July 2004.

[2] K. Masselos, S. Blionas, and T. Rautio," Reconfigurability requirements of wireless communication systems," in *Proc. of the IEEE Workshop on Heterogeneous Reconfigurable Systems on Chip*, Hamburg, Germany, April 2002.

[3] J. B. Anderson and E. Offer," Reduced-state sequence detection with convolutional codes," *IEEE Trans. Information Theory*, vol. 40,no 3., pp. 965972, May 1994.

[4] F. Sun and T. Zhang," Parallel high-throughput limited search trellis decoder VLSI design," *IEEE Trans. on VLSI Systems*, vol. 13, no. 9, pp. 1013-1022, September 2005.

[5] J. Jin and C. Tsui," A low power Viterbi decoder implementation using scarce state transition and path pruning scheme for high throughput wireless applications," *International Symposium on Low Power Electronics and Design*, pp. 406-411, 2006.

[6] R. Hegde, and N. R. Shanbhag," Soft digital signal processing," *IEEE Trans. on VLSI*, vol. 9, pp. 813-823, December 2001.

[7] Y. Liu, T. Zhang, and J. Hu," Low-power trellis decoder with overscaled supply voltage," in *IEEE Workshop on Signal Processing Systems Design and Implementation*, pp. 205-208, October 2006.

[8] A. P. Hekstra," An alternative to metric rescaling in Viterbi decoders," *IEEE Trans. Commun.*, vol. 37, pp. 12201222, November 1989.

[9] S. Lee, N. Shanbhag, and A. Singer,"Area-efficient, high-throughput MAP decoder architectures," *IEEE Trans. on VLSI Systems*, vol. 13, no. 8, pp. 921-933, August 2005.

[10] T. Chen and S. Naffziger," Comparison of adaptive body bias (ABB) and adaptive supply voltage (ASV) for improving delay and leakage under the presence of process variation," *IEEE Trans. on VLSI Systems*, vol. 11, no. 5, pp. 888-899, October 2003.

[11] C. C. Lin, Y. H. Shih, H. C. Chang, and C. Y. Lee,"A low power Turbo/Viterbi decoder for 3GPP2 applications," *IEEE Trans. on VLSI Systems*, vol. 14, no. 4, pp. 426-430, April 2006.