Error-Resilient Low-Power Viterbi Decoder Architectures

Rami A. Abdallah, *Student Member, IEEE*, and Naresh R. Shanbhag, *Fellow, IEEE*

***Abstract*—Three low-power Viterbi decoder (VD) architectures are presented in this paper. In the first, limited decision errors are introduced in the add-compare-select units (ACSUs) of a VD to reduce their critical path delays so that they can be operated at lower supply voltages without incurring timing errors. Power savings in this design can reach 58% and 44% with a 0.15 dB coding loss under reduced voltage operation and process variations, respectively, with adaptive supply voltage and adaptive body biasing applied to avoid timing errors. In the other two designs, we permit data-dependent timing errors to occur whenever a critical path in the ACSU is excited. *Algorithmic noise-tolerance* (ANT) is then applied to correct for these errors. Power reduction in these schemes is achieved by either overscaling the supply voltage [*voltage overscaling* (VOS)] or designing at the nominal process corner and supply voltage (*average-case design*). Two techniques are proposed to develop efficient estimators for error-correction and achieving increased robustness to timing based errors. The first is based on *reduced-precision redundancy* and the second on *state clustering*. The first can achieve up to 40% and 25% power savings under VOS and process variations with loss in coding gain of 1.1 and 1.2 dB, respectively, in a 130-nm CMOS process. The second can achieve up to 71% and 62% power savings under VOS and process variations, respectively, at a loss in coding gain of 0.8 and 0.6 dB, respectively. Under process variations, the designs achieve 16-33X improvement in bit error-rate (BER) performance at a signal-to-noise ratio (SNR) of 2 dB.**

***Index Terms*—Algorithmic noise tolerance, error resiliency, process variations, viterbi decoder (VD), voltage overscaling.**

I. Introduction

VITERBI DECODERs (VDs) are widely employed in modern communication systems such as fourth generation (4G) mobile systems, wireless local area network (WLAN), code division multiple access (CDMA), satellite communication, digital video broadcast (DVB), and digital magnetic recording. The high data-rate over increasingly impaired channels results in an tremendous increase in the power consumption of VDs. For example, a IEEE 802.11a/g WLAN compliant VD has 128 states at a data rate of 54 Mb/s and consumes 35% of the total receiver power including the digital and analog front end [1]. This also constitutes 76% of the total digital processing complexity (in ops/s) [2].

Low-power VD is a well-studied subject. Power reduction in VDs has been achieved by either reducing the number of states (reduced-state sequence decoder) [3], the size of survivor memory [4], or the number of trellis paths (limited-search trellis) [5] at the expense of increased BER and/or reduced throughput. Other approaches include scarce state transition [6] where the most-likely path passes through the zero states most of the time allowing shorter survivor memories and efficient limited-search trellis decoding [7].

These past works do not address the issue of robustness in the presence of process, voltage, and temperature (PVT) variations. These variations result in a wide distribution of error-free operating frequencies requiring a reliance on worst-case design and hence, resulting in high power consumption. Circuit level techniques such as adaptive body bias (ABB) and adaptive supply voltage (ASV) [8] can be employed to tighten the delay distributions. However, these techniques result in increased power with respect to the nominal case and do not scale very well with process technology.

The present-day worst-case design philosophy leads to high power consumption while nominal case design results in a loss in yield. A design approach based on 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 process corner and nominal (or reduced) voltage to save power and the resulting logic errors are corrected via architectural and algorithmic techniques.

The concept of error-resiliency for reducing power was proposed in [9], where *voltage overscaling* (VOS) was employed to reduce power by scaling the supply voltage until data-dependent timing errors start to appear. These timing errors were then corrected via *algorithmic noise-tolerance* (ANT) [9] whereby the statistics of data and timing errors are exploited to achieve approximate error detection and correction. Recently, a VOS-based VD [10] was proposed whereby timing errors in critical bits were compensated for by providing timing guard-bands via controlled introduction of clock skew.

In this paper, we present energy-efficient architectures for the add-compare-select unit (ACSU), a key computational kernel in the VD. The convolutional code used is a rate-1/2 128-state code at a data rate of 590 Mb/s. The first architecture is based on permitting limited decision errors in order to decrease the critical path delay of the ACSU so that it can be operated at reduced supply voltages. Two other architectures employ ANT to compensate for timing errors induced by VOS and/or process

Manuscript received January 19, 2009; accepted June 02, 2009. First published June 30, 2009; current version published November 18, 2009. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Warren J. Gross. This work was supported by the Gigascale System Research Center (GSRC), one of five research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation program, Texas Instruments, Inc., and NSF grant CCF 0729092.

The authors are with the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana IL 61801 USA (e-mail: rabdall3@illinois.edu; shanbhag@illinois.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2009.2026078

variations. The application of ANT increases latency, which can be a problem for recursive architectures such as the ACSU. Hence, we propose the use of block-interleaved pipelining (BIP) [11] to increase memory elements in the design, and thereby absorb the increase in latency. The algorithmic properties of the ACSU and the Viterbi algorithm are then exploited to develop two robust, error-resilient techniques: *redundancy-based* and *state clustering*, allowing the ACSU to operate at lower voltages or nominal process corner. Preliminary results on these two schemes were presented in [12] and [13], respectively.

The remainder of the paper is organized as follows. Section II presents background material on Viterbi algorithm and ANT. Section III describes the impact of VOS and process variations on the ACSU, and discusses the challenges in applying ANT in recursive architectures. Section IV proposes three low-power VD architectures based on error resiliency. Simulation results demonstrating the BER performance, reliability, and power savings under VOS and process variations for the proposed architectures are shown in Section V. Finally, Section VI outlines future directions.

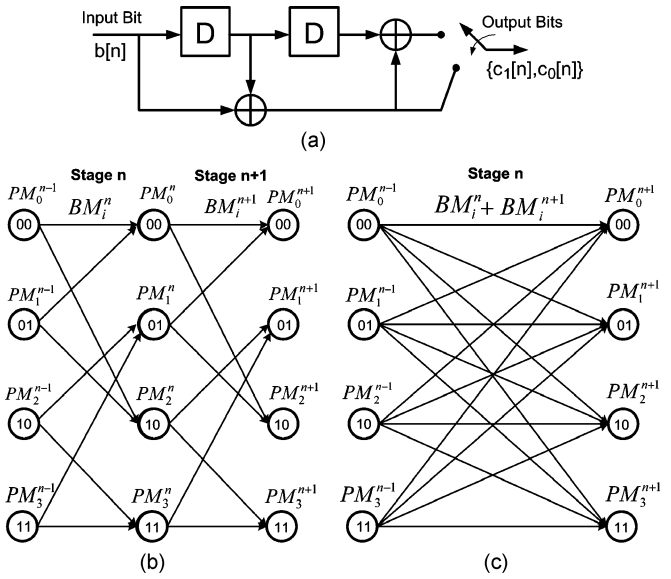

Fig. 1. A convolution code: (a) encoder. (b) Radix-2 trellis. (c) Radix-4 trellis.

II. Background

In this section, we present preliminaries of Viterbi algorithm and ANT. First, the Viterbi algorithm is introduced by describing the trellis structure and the general architecture of VD, followed by a description of ANT.

A. The Viterbi Algorithm

The Viterbi algorithm is an efficient procedure for solving maximum likelihood sequence estimation (MLSE) problems. Decoding of convolutional codes is one. The encoder [see Fig. 1(a)] for a rate 1/2 code generates two output codeword bits $(c_1[n], c_0[n])$ as a function of the input information bits $(b[n])$ and the encoder state (stored bits in the registers). The output bits are then transmitted over a noisy channel. The Viterbi algorithm estimates the most likely sequence of encoder state transitions given the received noisy samples. The encoder state evolution and the decoding process can be represented by the time-indexed trellis shown in Fig. 1(b). The trellis has four states representing the encoder state. Each state has two branches emanating from it. These represent possible transitions depending on the input bit $b[n]$ being a zero or a one. Each branch is characterized by a branch metric (BM) that indicates the distance (usually Euclidean) between the received samples and the ideal codeword $(c_1[n], c_0[n])$. Each path or a sequence of state transitions through the trellis has a path metric (PM) which is the sum of all BMs on that path. 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 (referred to as survivor path): At stage n and for each state, first the BMs are added to the PMs at stage $n-1$ for each incoming path from stage $n-1$ to form a new set of candidates for the PM at stage n, and then the path with the minimum PM among the new set of candidates is selected as the new survivor path for each state. This process is repeated at each stage using the PM of the survivor paths at previous stages. For the trellis in Fig. 1(b), there are two paths entering each state. The recursive update equation for the survivor PM at each state is given by

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

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 $PM_k^{(n+1)}$. The hardware unit that implements this update equation is referred to as the ACSU because it consists of an add (ADD) and a compare-select (CS) block. The regular trellis in Fig. 1(b) is called a radix-2 trellis. In higher radix processing, more than one section of the trellis is combined into a single section to increase decoding throughput. Fig. 1(c) shows one section of a radix-4 trellis where two stages of the radix-2 trellis in Fig. 1(b) are combined into a single stage. The radix-4 ACSU needs now to compare 4 updated PMs and select the minimum. In general, in radix-k processing, where k is a power of 2, $log(k)$ sections of radix-2 trellis are combined in a single section and the number of candidate paths in the ACSU is equal to k.

In this paper, we target the design of an ACSU for a 128-state rate-1/2 convolutional code. A generic VD architecture is presented in Fig. 2. 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 PM of each state is computed in parallel.

B. Algorithmic Noise Tolerance

Traditionally, architectures are designed to operate correctly, i.e., meet all timing specifications, at the worst-case PVT corner. ANT maintains circuit operation under timing errors achieving increased robustness to PVT variation and significant power savings.

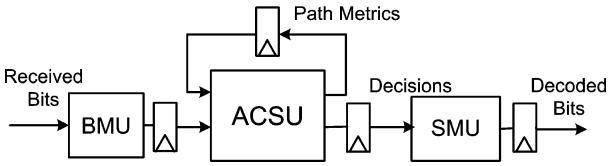

Fig. 2. A generic VD architecture.

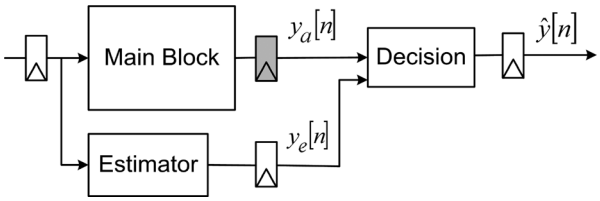

Fig. 3. ANT for a nonrecursive architecture. The shaded latch indicates the location of timing errors.

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, the main block may be subject to a supply voltage that is lower than the critical value needed to avoid timing errors. Thus, the output of the main block can be expressed as

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

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 reduced supply voltage. These errors are corrected by an estimator that produces a statistical replica $y_e[n]$ of the error-free main block output $y_o[n]$.

The design of an ANT-based system depends on the data correlation, system architecture, and statistical signal processing techniques. The challenge in ANT-based systems is to discover a low-complexity estimator with a much smaller critical path delay. This ensures that the estimator output is error-free even though the main block may exhibit intermittent timing errors. A number of estimation techniques have been proposed in the past. These include linear prediction [9], adaptive error cancellation [15], reduced precision redundancy (RPR), and input subsampling replica (ISR). In each technique, correlation in the signals or cross-correlation between signals and the error is exploited to generate an estimate of the correct output.

Error detection exploits the fact that in a least-significant bit (LSB) first computation, timing errors in the main block output occur in the most significant bits (MSBs). Thus, a large deviation between the main block output and the estimated output will be observed during an error event. A simple decision block can be used to detect and correct errors in the main block output as follows:

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

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

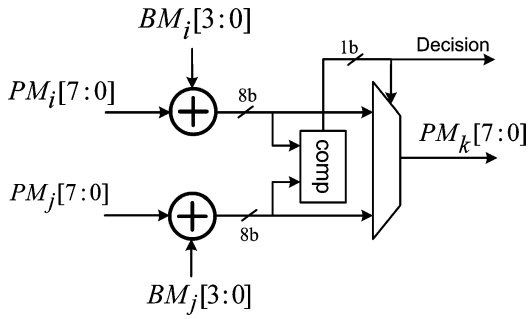

Fig. 4. The ACSU architecture.

III. Timing Errors and Error Resiliency in VD

In this section, we study the resiliency of the Viterbi algorithm to timing errors and challenges in applying ANT to VD architecture. Two main sources of errors will be studied: VOS and process variation. First, we study the effect of timing errors on the decoding performance of the VD. Next, we propose architectural techniques to apply ANT for general recursive architectures first, and then to the specific case of an ACSU.

A. Simulation Setup

We target the design of an ACSU for a 128-state rate-1/2 convolutional code. A detailed ACSU architecture, assuming 4-b BMs and 8-b PMs, is presented in Fig. 4. A state-parallel ACSU is commonly employed for high data-rate applications where the PM of each state is computed in parallel. For the sake of brevity, in the following, the time index n will not be listed. Instead, the precision of various signals will be referred to explicitly. In our design, the BMs and the PMs are quantized to 4-b and 8-b, respectively, e.g., $BM_i[3:0]$ is the BM added to $PM_i[7:0]$ in (1). The PMs are quantized using two's complement representation in order to ensure correct operation in presence of overflow [14].

In order to evaluate the impact of timing errors on the algorithmic behavior of the VD, we characterized the worst case delays of basic gates employed in the ACSU at different supply voltages using HSPICE in a IBM 130-nm CMOS process. An RTL-level simulation of the VD is then carried out with individual gate delays obtained from the circuit level characterization mentioned above so that the BER under different supply voltages can be obtained. The clock frequency is fixed at 590 MHz and is chosen to meet the timing constraints at 1.2 V in order to support a data rate of 590 Mb/s. An additive white Gaussian noise (AWGN) channel, with a channel SNR of less than 4 dB and binary phase-shift keying (BPSK) modulation is considered. The AWGN model and the SNR range for the outer (encoder to decoder) channel are typical for 802.11a application.

Process variations are classified into within-die (WID) and die-to-die (D2D) variations [16]. WID variations consist of variations between different devices on the same chip. D2D variations include variations between chips on different wafers and lots. These variations cause a large distribution in the delay profile of the gates.

In order to evaluate the effect of process variations on the BER, the delay distributions of various gates found in the ACSU

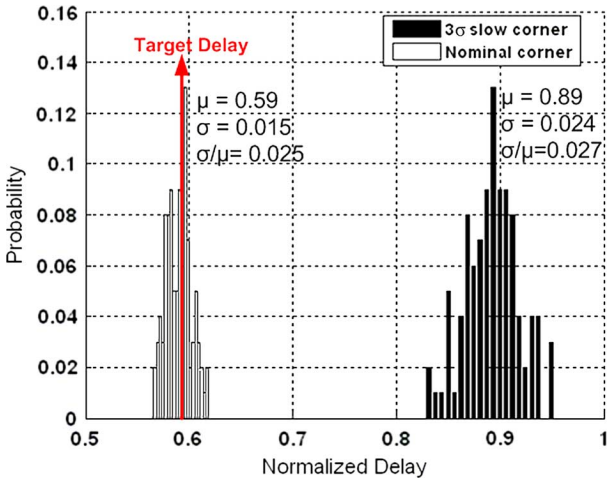

Fig. 5. Delay distribution of 1.2-V ACSU due to WID at nominal and 3σ slow corner.

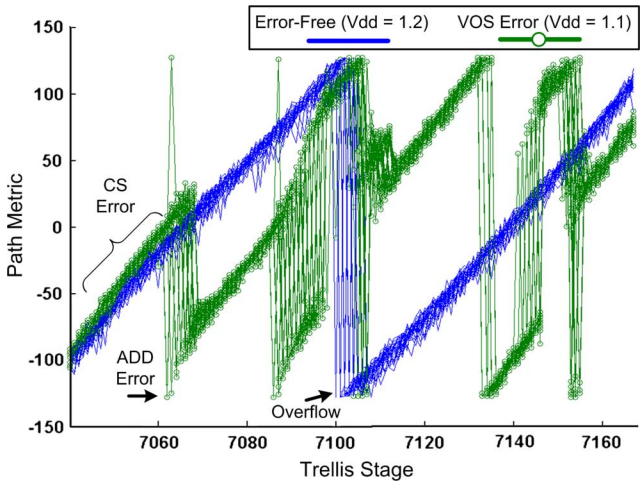

Fig. 6. Path metric evolution under timing errors.

were obtained via Monte Carlo simulations in an IBM 130-nm CMOS process at the 3σ slow process corner in the presence of WID variations. These delay distributions are sampled and used to obtain different RTL instances of the ACSU. For example, Fig. 5 shows the delay distributions of the ACSU due to WID variations at a supply voltage of 1.2 V at nominal and 3σ slow process corners using the extracted basic gate delays. The different ACSUs are sampled and simulated at the RTL level in order to generate different BER curves at the desired process corner. The clock frequency is determined by the data-rate and hence is kept fixed in all simulations.

B. Impact of VOS Timing Errors

Timing errors in ACSU can occur in the add or the compare-select block of ACSU in Fig. 4. Thus, one can classify timing errors into two types:

- ADD errors result in the computation and the selection of an incorrect PM. This event occurs when the MSBs in the BM adder (see Fig. 4) fail to compute correctly and the incorrect PM gets selected. In such a situation, 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. The effect of ADD errors can be observed by the abrupt shift due to sign change in PMs in Fig. 6, which shows the PM evolution for all states across multiple trellis stages.
- CS errors occur when the candidate PMs are computed correctly but, because of timing errors in the compare-select block, an incorrect (larger) PM is selected. The incorrect PM is close to the correct PM when there are no errors in the BM adder. Thus, CS errors are benign as compared to ADD errors. The effect of CS errors can be observed by the slight increase in the PM's evolution rate in Fig. 6 because sometimes a slightly larger PM is being selected.

Fig. 7 shows the BER curves obtained by reducing the supply voltage from 1.2 V to 1.1 V and 1 V. Note the large increase in BER due to VOS induced timing errors. The effect of CS errors on BER is less than that of ADD errors as seen in Fig. 7 at $V_{dd} = 1.1$ V, where BER curves with CS errors only are shown. Clearly, higher frequency CS errors can also have a large impact on the BER as decision errors occur more regularly on chosen paths (see Fig. 7 at $V_{dd} = 1.0$ V with CS errors only).

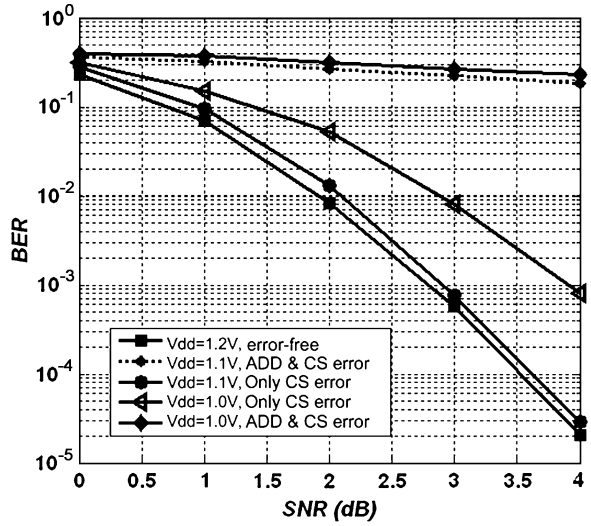

Fig. 7. Decoding performance with ACSU subject to different sources of timing errors.

C. Impact of Timing Errors Due to Process Variations

The impact on BER due to timing errors induced by process variations in the ACSU is very severe as can be seen in Fig. 8. There is a 4 orders-of-magnitude increase in the BER at a typical input SNR of 4 dB. As the ACSU is a recursive architecture, we next discuss issues and propose solutions for error resiliency in recursive systems in general, and the ACSU, in particular.

D. Algorithmic Noise-Tolerance in General Recursive Architectures

Fig. 9(a) shows a generic recursive architecture with three computational blocks $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$. Timing errors occurs at the output of $\mathbf{A}$. That is why an estimator (A_{est}) and a decision

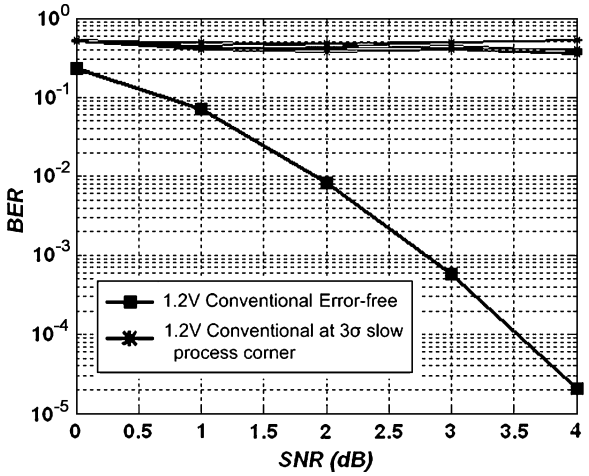

Fig. 8. Decoding performance with ACSUs under WID variations at the 3σ slow corner.

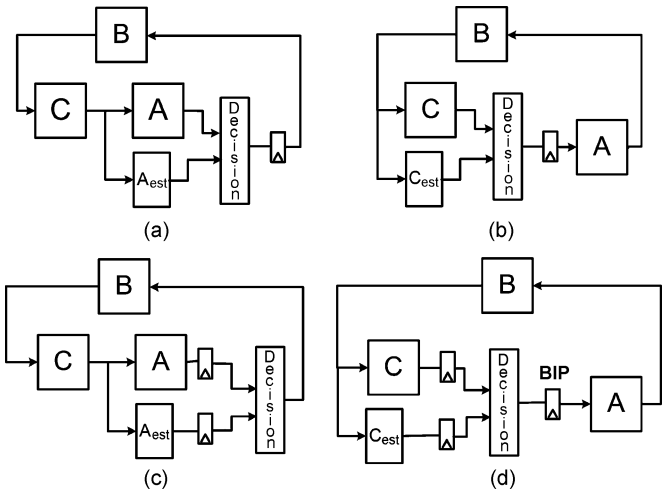

Fig. 9. ANT for recursive architectures with shaded latches indicating the location of timing errors: (a) timing errors impact hard-to-correct block $\mathbf{A}$. (b) Retiming to ease error-correction. (c) Retiming to prevent errors in decision block. (d) Introduction of additional latches using block interleaved pipelining (BIP).

block are added. Recursive architectures present two key challenges for the application of error-resiliency. First, recursive architectures suffer from *error propagation* where the error at time instant n can impact future outputs [see Fig. 9(a)], whereas in feedforward systems, the error affects the output at one time instance. Thus, highly effective estimators are needed for recursive architectures to keep the residual error within bounds. Second, the introduction of the decision block increases the critical path delay [see Fig. 9(a)], thereby generating timing violations in the decision block that are hard to correct because of its nonlinear nature. Feedforward architectures avoid this problem by introducing a pipelining register before the decision block as shown in Fig. 3. Pipelining registers cannot be introduced arbitrarily in recursive systems as that would change the functionality. Next, we present various schemes to alleviate decision block errors in general recursive architectures and in ACSUs.

Retiming [17] refers to the moving of latches inside a data-flow graph. Hence, one can retime the feedback register to place it at the output of a block whose errors can be easily corrected. Fig. 9(b) shows that the latch is retimed to the output of block $\mathbf{C}$, which is assumed to generate easily correctable errors. Also, retiming can be employed to ensure that the decision block computes correctly as shown in Fig. 9(c). Retiming in this manner will result in errors at the output of block $\mathbf{C}$. This is acceptable because an estimator is employed to correct the errors. However, the error frequency and error magnitude at the output will be greater than an equivalent nonrecursive architecture due to the increased critical path delay.

Look-ahead pipelining [17] can also be employed to introduce additional pipelining delays into the feedback loop in order to modulate the error distribution at the main block output. However, conventional look-ahead pipelining comes with hardware overhead. Instead, for VD applications, one can employ a low-complexity pipelining technique referred to as *block interleaved pipelining* (BIP) [11]. Thus, we can now apply BIP followed by retiming to remove the decision block from the critical path as shown in Fig. 9(d). BIP can be applied as long as the input can be processed in a block-based and block-independent manner. In streaming applications, we induce block-independence by introducing a known sequence (zeros) at block boundaries to force a specific shared state (all zeros) between the blocks. In fact, recent standards such as WiMax [18] and Long Term evolution (LTE) [19] employ tail-biting convolutional codes, which require block processing anyway. Thus, the BIP technique can be readily applied in such applications without additional storage. Also, BIP enhances the throughput by the same factor. Thus, as in case of pipelining, one can reduce the power overhead by scaling the supply voltage. In the rest of this paper, we focus only on power savings in the ACSU.

E. Algorithmic Noise-Tolerance in ACSU

The ACSU is usually implemented using an LSB-first BM adder (ADD) followed by an LSB-first CS block using carry ripple adders. These two operations execute in parallel so that the delay is dominated by a single 8-bit adder as illustrated in Fig. 10(a). Errors in the ACSU can occur in the ADD or the CS block. A low-complexity estimator can be designed for the ADD block as it implements a linear operation. Determining an effective estimator for the CS block is hard as it is nonlinear and, in addition, designing a low-overhead ANT decision block to detect errors in the CS is hard because the output is a 1 or 2-bit signal. Therefore, in the proposed ANT techniques for ACSU, we choose to avoid timing errors in the comparator. One approach is to retime such that the feedback register appears at the input of the CS block as illustrated in Fig. 10(b). However, doing so doubles the critical path delay: the comparator involves a selection step so that ADD block needs to wait for the CS output before it starts processing unlike in Fig. 10(a) where CS block can start processing in parallel with the ADD block. Instead of retiming, since a limited number of CS errors (decision-based errors) can be tolerated as explained in Section III-B, we propose in the next section to use a relaxed CS (RCS) block to avoid timing-based decision errors in all ACSUs. The RCS block computes using two low-precision reduced-delay CS blocks operating in parallel at the expense of a small loss in decoding perfor-

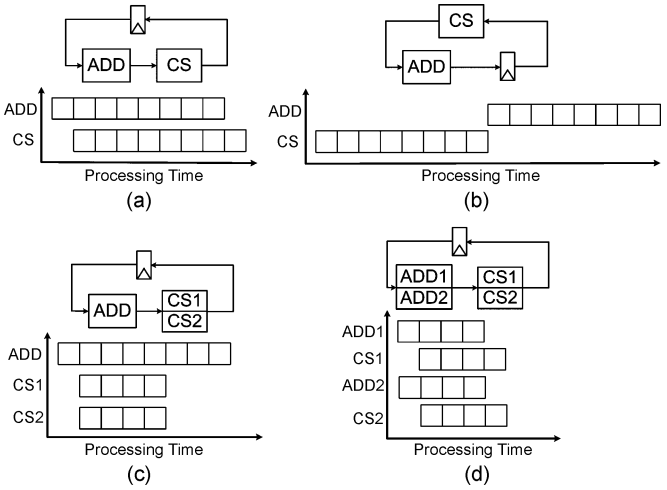

Fig. 10. Throughput and latency of: (a) conventional ACSU. (b) Retimed ACSU. (c) Relaxed ACSU. (d) Fast ACSU (**FACSU**).

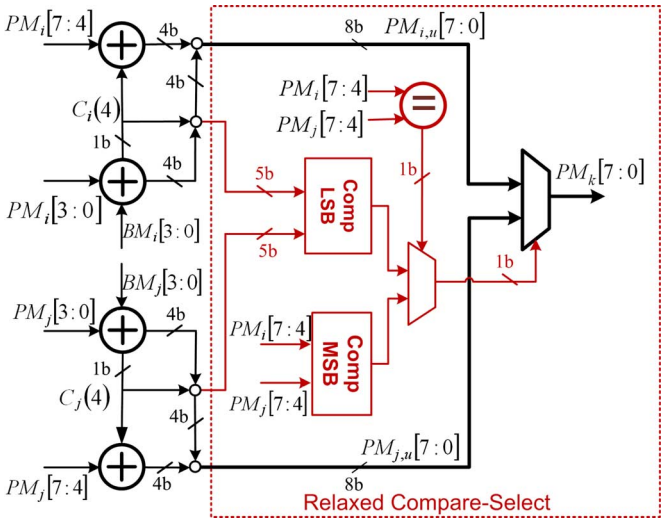

Fig. 11. An ACSU with RCS block.

mance [see Fig. 10(c)]. Furthermore, we decompose the ADD block into two parallel reduced-delay ADD blocks. Combining this with the RCS block, we obtain the fast ACSU (**FACSU**) based VD architecture in Fig. 10(d). In the **FACSU**, the reduced-delay ADD and CS blocks compute in parallel so that it can operate error-free at lower supply voltages due to reduced critical path delay. The **FACSU** itself is low-power but not error-resilient and thus can be employed as an estimator in the ANT-based techniques proposed in Section IV.

IV. Low-Power ACSU Architectures

In this section, we present two novel low-power error-resilient ACSU architectures. In addition, we also present the **FACSU**, which consumes lower power than a conventional ACSU, but is not error-resilient. We first present the RCS block which is employed in all proposed architectures and avoids timing errors in the comparison step of the ACSU.

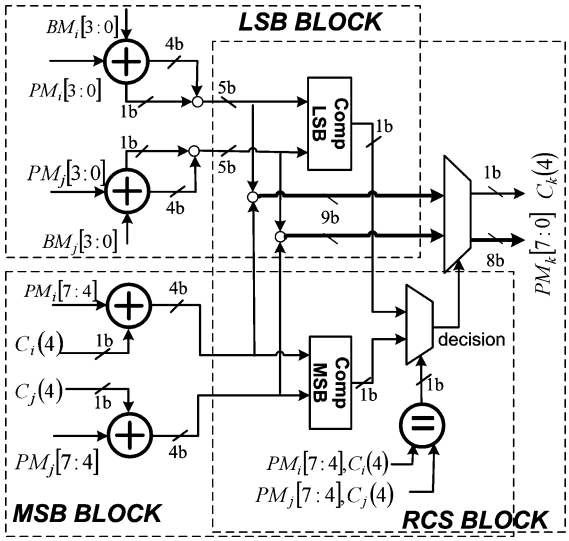

Fig. 12. The **FACSU** architecture with delayed carry and RCS block.

A. RCS Block

The RCS block reduces the critical path delay of the comparison step at the expense of limited decision errors. It is partitioned into an MSB and an LSB comparator followed by a selection mux as shown in Fig. 11. Recall that the BMs and the PMs are quantized to 4-b and 8-b, respectively. The MSB comparator is a 4-b comparator that compares the MSBs of input PMs ($PM_i[7:4]$ and $PM_j[7:4]$). 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 ($C_i(4)$ and $C_j(4)$). The LSB comparator decision is considered only when the MSBs of PMs ($PM_i[7:4]$ and $PM_j[7:4]$) are equal. The relaxed comparator has a critical path delay equal to that of a 5-bit adder delay. The RCS block will make errors once in a while. Table I lists the scenarios when this event occurs. The RCS block makes incorrect decisions only when the MSBs of the PMs differ by unity and the propagated carries ($C_i(4)$ and $C_j(4)$) from the LSB parts to the MSB parts make them equal, as illustrated in cases A and B of Table I. The RCS decision is based on the 4-b MSBs of the PMs before carry propagation. However, the correct decision depends on the 4-b LSBs of the updated PMs ($PM_{i,u}[3:0]$ and $PM_{j,u}[3:0]$) since the propagated carries will make the 4-b MSBs of the updated PMs ($PM_{i,u}[7:4]$ and $PM_{j,u}[7:4]$) equal. These wrong decisions have a small effect on decoding performance because the chosen PMs 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. In the simulation section, we show that there is only a 0.15-dB loss in coding gain due to the use of the RCS block.

B. The FACSU Architecture

The **FACSU** architecture permits limited errors to occur in order to reduce the critical path delay. Low-power operation is achieved by scaling down the supply voltage in order to exploit the increased timing slack. The **FACSU** employs a RCS block and *delayed carry*. The architecture of the **FACSU** is shown in Fig. 12, where the computation is partitioned into an LSB and an

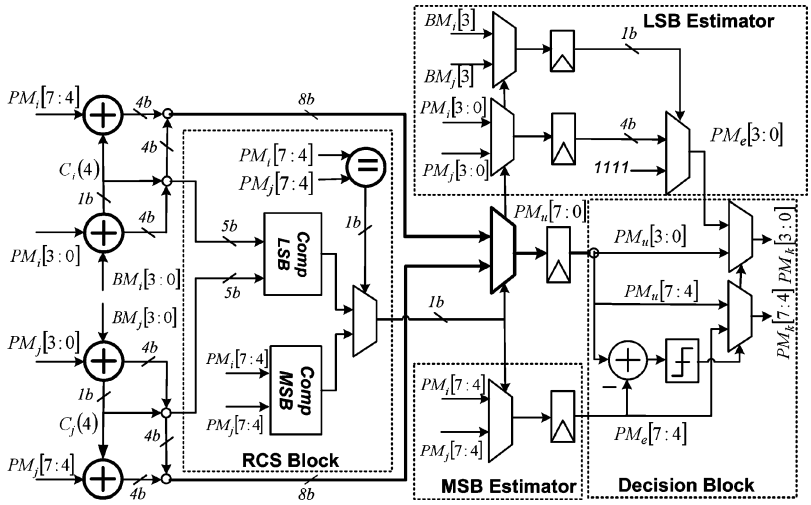

Fig. 13. The **RA-ACSU**: BIP and register retiming are used to introduce latches.

TABLE I
THE TWO CASES WHERE THE RCS BLOCK MAKES ERROR

	Case A	Case B
$PM_i\,(7:4) - PM_j\,(7:4)$	1	-1
Propagated carry i $(C_i(4))$	0	1
Propagated carry j $(C_j(4))$	1	0
$PM_{i,u}\,(7:4) - PM_{j,u}\,(7:4)$	0	0

TABLE II
COMPLEXITY ESTIMATE OF **RA-ACSU**

Modules	2-input AND	2-input OR	Inverter	Transistors
Conventional ACSU	163	92	91	1202
Main **RA-ACSU**	176	102	101	1314
Correction Overhead	99	53	51	710
RA-ACSU	275	155	152	2024

MSB block with the delay of each block being approximately half the original ACSU delay. During BM addition stage, 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-bit value with the additional bit representing the delayed carry. This reduces the BM adder delay by a factor of 2. In the comparison stage, **FACSU** uses a RCS block whose operation was described in Section IV-A.

C. The Redundancy-ANT ACSU (**RA-ACSU**) Architecture

The **RA-ACSU** architecture is designed to detect and correct timing errors in the ADD block, i.e. ADD errors. Limited decision errors in the CS block are permitted, i.e., left uncorrected, via the use of an RCS block. Further, we use BIP to provide sufficient time for the ANT correction block to complete computation.

The only source of timing errors in the **RA-ACSU** architecture in Fig. 13 is in the ADD block. In order to correct for ADD errors, the **RA-ACSU** incorporates an MSB and an LSB estimator of the updated PM $(PM_u[7:0])$. The MSB estimator selects between the MSBs of the input PMs, $PM_i[7:4]$ and $PM_j[7:4]$, to generate an estimate $(PM_e[7:4])$ of the MSBs of the updated PM $(PM_u[7:0])$. The LSB estimator generates an estimate $PM_e[3:0]$ for the 4 LSBs of the updated PM $(PM_u[3:0])$. Based on the decision of the RCS block, the LSB estimator first determines if the chosen BM is greater than or equal to 8, i.e, $BM[3] = 1$. If it is, then $PM_e[3:0]$ is set to all ones since the BM is considered large. Otherwise, $PM_e[3:0]$ is set equal to the LSBs of the input PM corresponding to the chosen BM, i.e., either $PM_i[3:0]$ or $PM_j[3:0]$. In other words, we assume that the BM value in this case is zero.

The decision block computes the difference between the MSBs of the updated PM $PM_u[7:4]$ and its estimate $PM_e[7:4]$. If the value of this difference is greater than unity, then an error is detected. In such a case, the output PM is set to the estimated PM $PM_e[7:0]$; otherwise, the output PM is set to the updated PM. Table II indicates that there is 68% increase in gate complexity in the **RA-ACSU** when compared to the conventional ACSU. We will show in Section V that **RA-ACSU** saves power in spite of this overhead.

D. State-Clustered ANT ACSU (**SC-ACSU**) Architectures

In state-clustered ACSU, we exploit the structure of the trellis to provide an estimate of an erroneous PM. We cluster the ACSUs that have the same set of previous states into a single cluster. In this way, the updated PMs for all the ACSUs in each group will be close to each other since they share the same input PMs and differ only in the added BMs. Therefore, we can use a single estimator in each cluster to correct for the erroneous PMs. Estimation schemes will be discussed in next subsection. For example, in a 128-state rate-1/2 trellis, 7 bits are needed to represent each state and each state has two possible transitions depending on the input bit being a zero or a one. Therefore, the trellis can be partitioned into 64 disjoint 2-state clusters of the form shown in Fig. 14 and 64 estimators are needed in total - one for each cluster.

In order to increase the number of ACSUs per cluster and consequently decrease the number of estimators, we can employ radix-k processing where each state will have k transitions resulting in a k-state cluster. For example, the 128 states in a rate-1/2 radix-4 trellis can be partitioned into 32 disjoint 4-state

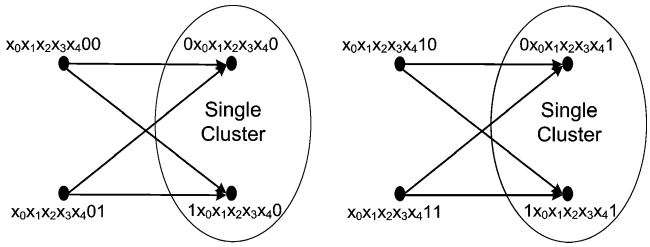

Fig. 14. State clustering for 128-state rate-1/2 code with 2 states per cluster where states share the same set of previous states.

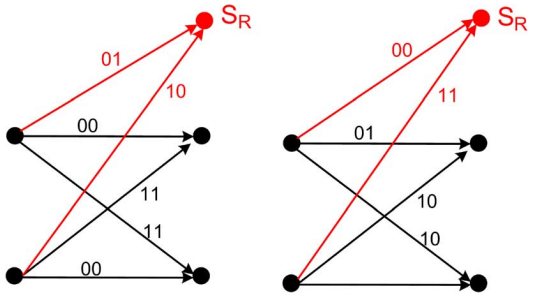

Fig. 15. State Clustering with a redundant state S_R.

clusters requiring 32 estimators. A number of **SC-ACSU** architectures can be designed by employing various PM estimators.

1) Estimation Techniques: Two possible estimation techniques are presented to correct for ADD block errors under state clustering. In the first, the **FACSU** is used to compute the PM for one of the states in each cluster. The **FACSU** is expected to operate correctly under reduced voltage and thus provide a good estimate for the erroneous PMs in the cluster. This scheme is referred to as **SC-FACSU**. In the second, the **FACSU** is used to compute the PM of a redundant state S_R which is added to each cluster. This technique is referred to as **SCS-FACSU**. The redundant state S_R has the same set of previous states as the rest of the states in the cluster (see Fig. 15). The codeword (or BM) for any branch, joining any state S_X to the redundant state S_R, is chosen to be equidistant from the codewords on the branches joining state S_X to the rest of the states in the cluster. For example, in the first cluster in Fig. 15, the codewords on branches feeding into S_R are chosen as 01 and 10, which are equidistant from 00 and 11. This constraint forces the estimated PM to be closer to the correct PM than in the case of **SC-FACSU**. With higher-radix state clustering, in certain clusters it is impossible to have codewords equidistant from the rest of the codewords, so codewords for the redundant branches are assigned to be as equidistant as possible. Both schemes employ the RCS block in order to avoid timing-based decision errors.

2) Architectures: The architectures for **SC-FACSU** and **SCS-FACSU** based on radix-2 are presented in Fig. 16(a) and (b), respectively. In **SC-FACSU**, state S_X uses the **FACSU** architecture and is error free. State S_Y uses the regular ACSU but with RCS. The decision block in Fig. 16(a) corrects the output of S_Y by using the estimate provided by **FACSU** (S_X). It computes the difference between the MSBs

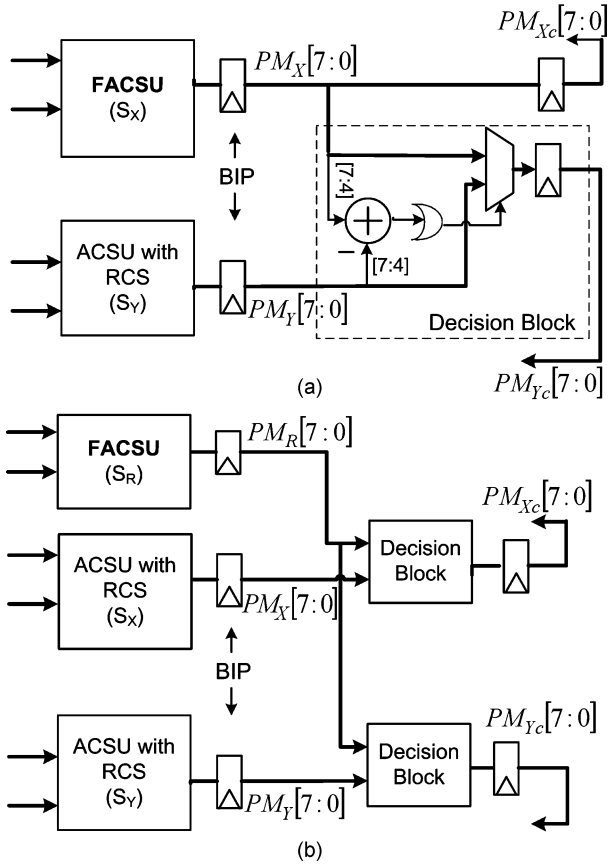

Fig. 16. State clustering architecture with two states X and Y: (a) the **SC-FACSU**, and (b) the **SCS-FACSU** architecture.

TABLE III
COMPLEXITY ESTIMATE OF STATE CLUSTERING ARCHITECTURES

Modules	2-input AND	2-input OR	Inverter	Transistors	Overhead
2-State Cluster					
Conventional	326	184	182	2404	
SC-ACSU	395	226	219	2922	22%
SCS-FACSU	618	353	343	4570	90%
4-State Cluster					
Conventional	652	368	364	4808	
SC-ACSU	859	506	457	6374	32%
SCS-FACSU	1091	604	562	7904	64%

of the PM of $S_Y(PM_Y[7:4])$ and that of $S_X(PM_X[7:4])$. If the absolute value of this difference is greater than unity, then an error is detected. In such a case, the corrected PM of $S_Y(PM_{Yc}[7:0])$ is set to $PM_X[7:0]$. Otherwise, it is set to the output PM of $S_Y(PM_Y[7:0])$. In the **SCS-FACSU** (see Fig. 16(b)), the estimate is provided by the redundant state $S_R(PM_R[7:0])$. This PM is used to correct the error prone PMs of state $X(PM_X[7:0])$ and state $Y(PM_Y[7:0])$. Architectures for higher radix clustering can similarly be derived. **SC-FACSU** and **SCS-FACSU** have 22% and 90% increase in gate complexity, respectively over conventional architecture, as shown in Table III. The complexity overhead for **SCS-FACSU** decreases as the number of states per cluster is increased since a single estimate is used for larger number of states.

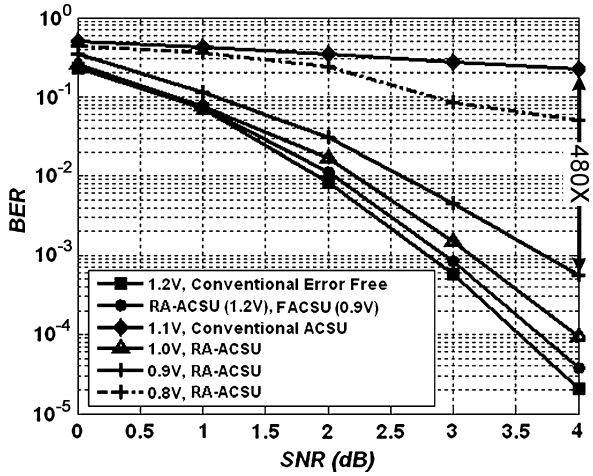

Fig. 17. BER of conventional ACSU, **FACSU**, and **RA-ACSU** at different supply voltages.

V. Simulations and Results

This section describes the BER performance and power savings achieved by the various proposed architectures (**FACSU**, **RA-ACSU**, **SC-FACSU**, and **SCS-FACSU**) under voltage overscaling and process variations, and compares it with the conventional ACSU. The simulation procedure has already been described in Section III-A.

A. Voltage Overscaling Results

HSPICE simulation in an IBM 130-nm CMOS process shows that the **FACSU** can run at 0.9 V in absence of timing errors while maintaining the same throughput as a conventional 1.2-V ACSU. The only source of errors in this case are the CS errors introduced by the RCS block. Fig. 17 shows the BER curves for the conventional, **FACSU**, and **RA-ACSU** at various supply voltages. As indicated in Section IV-B, **FACSU** suffers only from a 0.15 dB loss in the coding gain at its nominal supply voltage (0.9 V). Similar performance is obtained for **RA-ACSU** at a nominal supply voltage of 1.2 V since the only errors under this setting are due to the RCS block. As the supply voltage is reduced from 1.2 V to 1.1 V, the conventional ACSU BER increases dramatically reaching four orders-of-magnitude at an SNR of 4 dB. The **FACSU** will also show a large increase in BER as timing errors start to occur at voltages beyond its nominal voltage (0.9 V) since it is not resilient to timing errors.

The **RA-ACSU** shows 1 dB loss in the coding gain at a BER of 10^{-3} under VOS errors with supply voltage equals to 0.9 V. Reduction of voltage beyond 0.9 V leads to a noticeably degraded performance of **RA-ACSU** (see the BER curve for $V_{dd} = 0.8\,\mathrm{V}$) due to increased frequency of VOS errors as well as errors in the estimator and decision blocks. Thus, **RA-ACSU** is resilient to VOS errors allowing power savings with small loss in coding performance. Note that coding loss due to error resiliency is smaller at low SNR than higher SNR since estimation errors in our techniques start to dominate channel errors as SNR is increased.

Fig. 18 shows the BER curves for state clustered architectures obtained by reducing the supply voltage from 1.2 V to 0.8 V with two and four states per cluster in (a) and (b) respectively. In Fig. 18(a), at 0.9 V, **SC-FACSU** and **SCS-FACSU** show only 0.8 dB and 0.35 dB loss in the coding gain, respectively, at a BER of 3×10^{-4}, with approximately 667X and 2000X BER reduction over conventional ACSU at a SNR of 4 dB. At 1.2 V, **SC-FACSU** shows only a loss of 0.15 dB due to RCS block errors. Reduction of voltage beyond 0.9 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, in Fig. 18(b), where we have four states per cluster, **SC-FACSU** and **SCS-FACSU** show only a loss of 0.7 dB and 0.25 dB in coding gain at the same BER, with 956X and 2442X BER reduction at a SNR of 4 dB. Thus, higher radix computations improve the effectiveness of state clustering architectures.

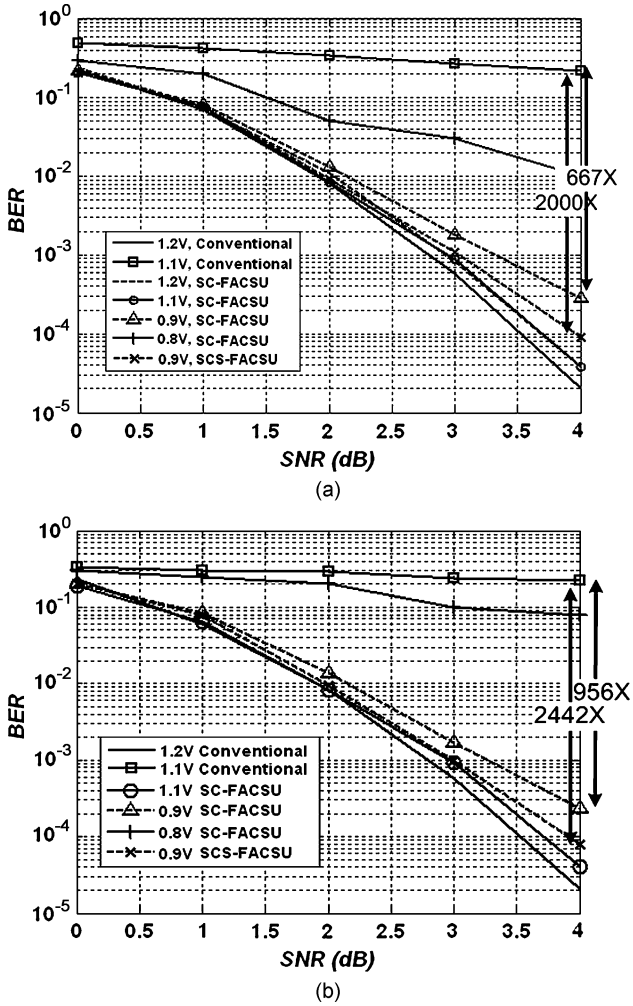

Fig. 18. BER of conventional ACSU, **SC-FACSU**, and **SCS-FACSU** at different supply voltages: (a) using radix-2 computations and (b) using radix-4 computations.

B. Process Variations Results

Figs. 19(a) and (b) show the BER distribution due to process variations at an SNR of 2 dB for conventional ACSU, **FACSU**, **RA-ACSU** and **SC-FACSU** with two states per group. Conventional ACSU suffers from a large increase in BER under process variations. ASV and/or ABB need to be applied to keep its delay fluctuations at the 3σ slow process corner within the operating

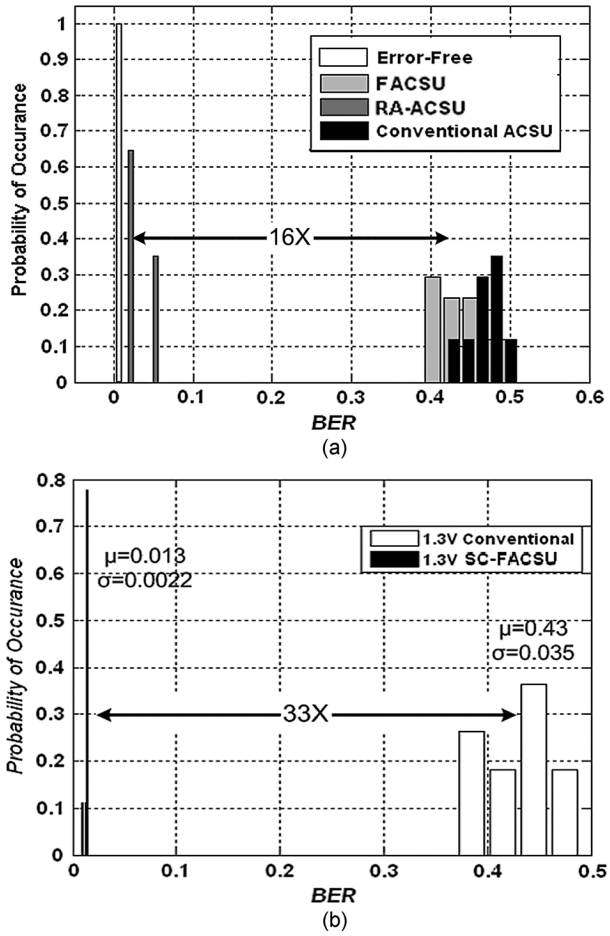

Fig. 19. BER distribution at 3σ slow corner with WID variations and $\mathrm{SNR} = 2$ dB: (a) conventional, **FACSU**, and **RA-ACSU**, and (b) **SC-ACSU**.

TABLE IV
BER AT 3σ SLOW PROCESS CORNER WITH WID VARIATIONS AT 2 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

clock frequency. Similarly, **FACSU** shows large degradation in performance due to timing errors caused by process variations. On the other hand, **RA-ACSU** (see Fig. 19(a)) and **SC-FACSU** (see Fig. 19(b)) can operate at the 3σ slow process corner with 16X and 33X average BER reduction if the voltage is increased from 1.2 V to 1.3 V. Performance under process variations at 2 dB SNR for **SC-FACSU** and **SCS-FACSU** with various radix processing is shown in Table IV. We see that **SCS-FACSU** shows more resiliency to process variations than **SC-FACSU**. Similar conclusion applies as we go into higher radix processing (compare radix-2 to radix-4 results in Table IV).

Fig. 20(a) and (b) show the BER curves under process variations for conventional ACSU, **FACSU**, **RA-ACSU**, and **SC-FACSU**. **RA-ACSU** exhibits a coding loss of 0.4 dB in approximately 66% of the cases, while the rest experience a 1.2 dB loss. The conventional ACSU and **FACSU** show a highly degraded BER at all SNRs. For **SC-FACSU** with 2-state cluster

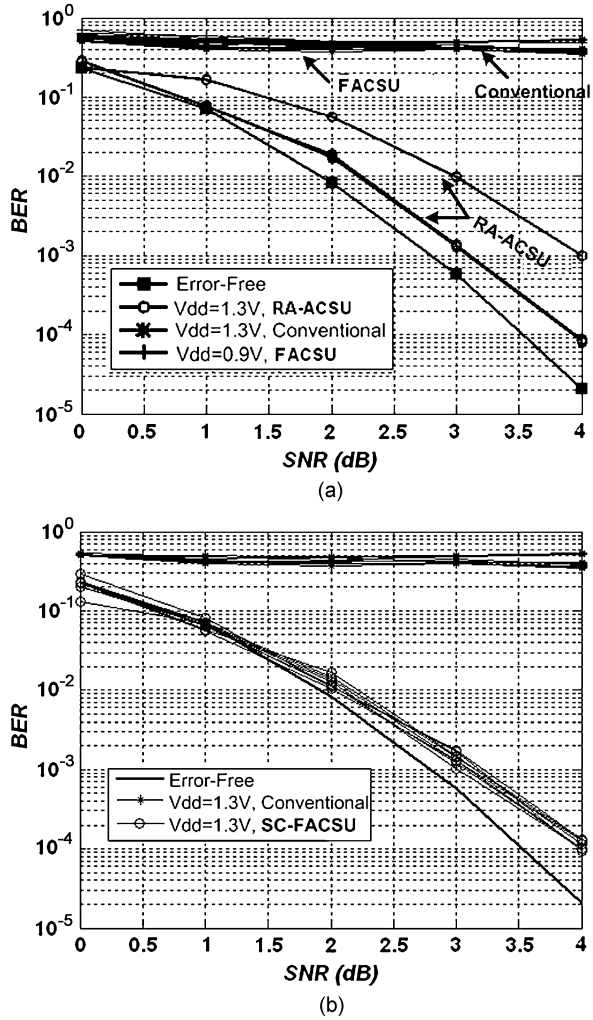

Fig. 20. BER at 3σ slow corner with WID variations: (a) conventional ACSU, **FACSU**, **RA-ACSU**, and (b) **SC-FACSU**.

in Fig. 20(b), there is only a 0.6 dB loss in the average coding gain at a BER of 3×10^{-4} using **SC-FACSU**. Thus, **RA-ACSU** and **SC-FACSU** are resilient to process variations at all SNRs while conventional ACSU, and **FACSU** are highly sensitive to process variations.

C. Power Savings

HSPICE simulations are carried out to estimate the ACSU power consumption shown in Fig. 21(a) and (b). An input test vector of size 50 is used and clock frequency is fixed to meet the target data rate. At the nominal process corner, **RA-ACSU** including correction overhead achieves 40% power savings under VOS while a **FACSU** exhibits 58% power. With 2-state clusters, **SC-FACSU** and **SCS-FACSU** achieve power savings of 71% and 42%, respectively, and a power savings of 56% and 41%, respectively, with 4-state clusters.

As seen in Fig. 19(a), at the 3σ slow process corner, the conventional and **FACSU** architectures breakdown in terms of BER under WID variations. Thus, ASV and ABB are applied to the conventional ACSU and **FACSU** to enable these architectures to operate at the target data rate. Under this setting, the 1.3-V **RA-ACSU** exhibits 25% power savings over the conventional

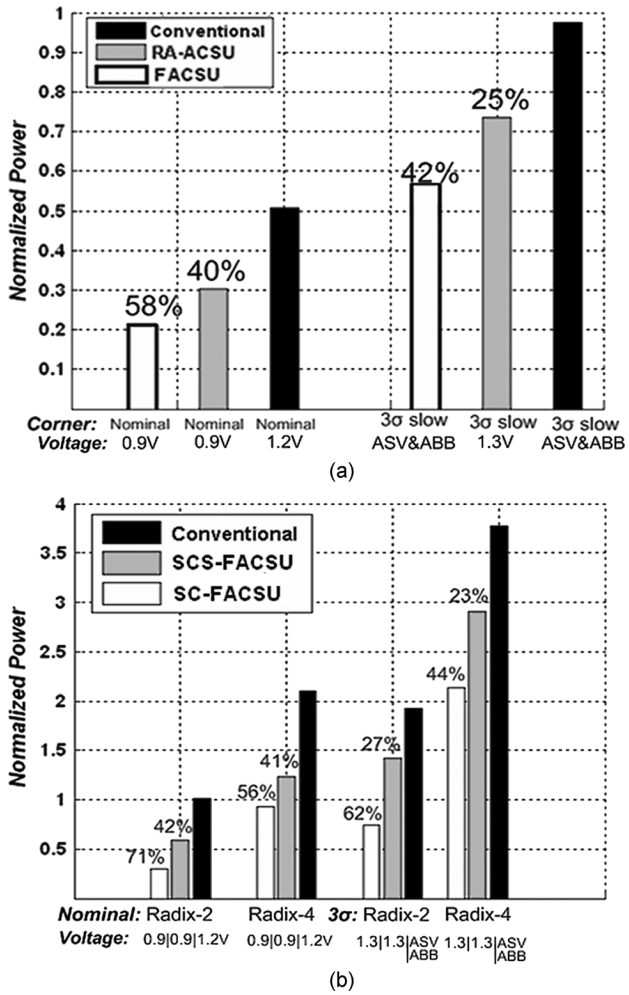

Fig. 21. Power consumption per ACSU under VOS and process variations for a $\mathrm{BER} = 10^{-3}$: (a) conventional, **RA-ACSU**, and **FACSU** architectures, and (b) conventional, **SC-FACSU**, and **SCS-FACSU** architectures.

ACSU whereas the **FACSU** achieves 42%. Under process variations, the power savings achieved by **SC-FACSU** and **SCS-FACSU** are 62% and 27%, respectively, with 2-state cluster, and 44% and 23%, respectively, with 4-state cluster. Note that **SCS-FACSU** provides a non-trivial reduction in power (23% for radix-4) in spite of the large (65% see Table III) overhead due to the combined effect of voltage overscaling and reduction in average activity factor. The activity factor in **SCS-FACSU** doesn't increase in direct proportion to the gate-count when compared to the conventional architecture. This is because the activity of the error detection block depends only upon the MSBs of the PMs which do not change as rapidly as the LSBs. In addition, the estimator output is selected only when a timing error occurs, which is a relatively rare event.

Although **FACSU** demonstrates a large power savings, it is not resilient to timing errors induced by VOS or process variations as was shown in Fig. 20. 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 the cluster size increases, leading to smaller number of estimators.

VI. Conclusions and Future Work

This paper introduces error-resilient architectures to decrease power consumption in VDs. Three low-power schemes for ACSU have been presented that enable a tradeoff between low-power and reliability. Simulations shows significant power savings and increased robustness to process variations. In practice, the extent of process variations is not known before hand or may change as the process matures. Thus, adaptive versions of the proposed error-resiliency techniques will need to be implemented, whereby at start-up, a power-up calibration block can provide known inputs and compare with expected outputs of the error-resilient block, in order to gauge the raw error-rates and hence the impact of process, temperature and voltage. Then, sufficient computational resources can be assigned in order to bring out the desired level of system level reliability, e.g., BER in case of a VD.

Error-resiliency, as shown in this paper, provides increased robustness to transistor-level variations making it a promising design for future nanoscale processes. Modern day CAD tools and design methodologies do not support the design of error-resilient systems. New metrics for verification and test need to be defined, that comprehend the statistical attributes of such systems. This is a challenge the CAD community has just begun to address.

Error-resiliency has been applied to the VD logic. The survivor memory unit also constitutes a significant component of the VD. Recent results in [20] show that error-resiliency can also be applied to memory. Thus, error resiliency in the survivor memory is a good candidate for further investigation. Future work can also be directed toward the design of more efficient estimators for recursive architecture to combat error propagation. Error-resiliency in other types of forward-error control decoders such as turbo and LDPC decoders also presents a topic for future research as they are being widely employed in different applications with more stringent decoding requirements leading to increased complexity and power consumption.

Acknowledgment

The authors would like to thank M. Goel and S.-J. Lee from Texas Instruments, Inc., for helpful discussions.

References

[1] B. Bougard *et al.*, "Energy-scalability enhancement of wireless local area network transceivers," in *Proc. IEEE Workshop on Signal Process. Adv. Wireless Commun.*, Lisboa, Portugal, Jul. 2004.

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

[3] J. B. Anderson and E. Offer, "Reduced-state sequence detection with convolutional codes," *IEEE Trans. Inf. Theory*, vol. 40, no. 3, pp. 965–972, May 1994.

[4] Y. Chang, H. Suzuki, and K. K. Parhi, "A 2-Mb/s 256-state 10-mW rate-1/3 Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 35, no. 6, pp. 826–835, Jun. 2000.

[5] F. Sun and T. Zhang, "Parallel high-throughput limited search trellis decoder VLSI design," *IEEE Trans. VLSI Syst.*, vol. 13, no. 9, pp. 1013–1022, Sep. 2005.

[6] S. Kubota, S. Kato, and T. Ishitani, "Novel Viterbi decoder VLSI implementation and its performance," *IEEE Trans. Commun.*, vol. 41, no. 8, pp. 1170–1178, Aug. 1993.

[7] J. Jin and C. Tsui, "A low power Viterbi decoder implementation using scarce state transition and path pruning scheme for high throughput wireless applications," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Oct. 2006, pp. 406–411.
[8] 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. VLSI Syst.*, vol. 11, no. 5, pp. 888–899, Oct. 2003.
[9] R. Hegde and N. R. Shanbhag, "Soft digital signal processing," *IEEE Trans. VLSI*, vol. 9, no. 6, pp. 813–823, Dec. 2001.
[10] Y. Liu, T. Zhang, and J. Hu, "Low-power trellis decoder with over-scaled supply voltage," in *Proc. IEEE Workshop on Signal Process. Syst. Design Implement.*, Oct. 2006, pp. 205–208.
[11] S. Lee, N. Shanbhag, and A. Singer, "Area-efficient, high-throughput MAP decoder architectures," *IEEE Trans. VLSI Syst.*, vol. 13, no. 8, pp. 921–933, Aug. 2005.
[12] R. A. Abdallah and N. R. Shanbhag, "Error-resilient low-power Viterbi decoders," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2008.
[13] R. A. Abdallah and N. R. Shanbhag, "Error-resilient low-power Viterbi decoders via state clustering," in *Proc. IEEE Workshop on Signal Process. Syst. (SiPS)*, Oct. 2008.
[14] A. P. Hekstra, "An alternative to metric rescaling in Viterbi decoders," *IEEE Trans. Commun.*, vol. 37, no. 11, pp. 1220–1222, Nov. 1989.
[15] L. Wang and N. R. Shanbhag, "Low-power filtering via adaptive error-cancellation," *IEEE Trans. Signal Process.*, vol. 51, no. 2, pp. 575–583, Feb. 2003.
[16] K. Bowman, S. Duvall, and J. Meindl, "Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration," *IEEE Trans. Signal Process.*, vol. 37, no. 2, pp. 183–190, Feb. 2002.
[17] K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York: Wiley, 1999.
[18] WiMAX Forum Mobile System Profile ver. 1.1.0, July 2006.
[19] 3GPP TS 36.212, Evolved Universal Terrestrial Radio Access (E-UTRA): Mutiplexing and Channel Coding ver. 8.4.0, Sep. 2008.
[20] F. J. Kurdahi *et al.*, "System-level SRAM yield enhancemet," in *Proc. Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2006, pp. 179–184.

Rami A. Abdallah (S'06) received the B.Eng. degree with highest distinction from the American University of Beirut (AUB), Beirut, Lebanon, in 2006 and the M.Sc. degree from the University of Illinois at Urbana Champaign (UIUC) in 2008, all in electrical and computer engineering.

He is currently pursuing the Ph.D. degree at UIUC. Since August 2006, he has been a Research Assistant with the Coordinated Science Laboratory (CSL), UIUC. During summers of 2007, 2008, and 2009, he was with Texas Instruments, Inc., Dallas, with the Digital Signal Processing Solutions R&D center where he was involved in the design of On-chip DC-DC conversion and communication receivers for Long Term Evolution (LTE) and WiMAX. His research interests are in the design of integrated circuit (IC) and systems for communications, digital signal processing, and general purpose computing.

Dr. Abdallah was on the Dean's honor list at AUB from 2002 to 2006, and was selected among World's top students to participate in the Research Science Institute at the Massachusetts Institute of Technology (MIT), Cambridge, in 2001. He received the Hariri Foundation Silver Medal in 2002, the Scientific Development Association Scholarship in 2005 and 2006, the Charli S. Korban award in 2006, and the HKN Honor Society Scholarship award in 2009.

Naresh R. Shanbhag (F'06) received the Ph.D. degree from the University of Minnesota, Minneapolis, in 1993 in electrical engineering.

From 1993 to 1995, he was with AT&T Bell Laboratories, Murray Hill, NJ, where he was the lead chip architect for AT&T's 51.84 Mb/s transceiver chips over twisted-pair wiring for Asynchronous Transfer Mode (ATM)-LAN and very high-speed digital subscriber line (VDSL) chip-sets. Since August 1995, he is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana Champaign, and the Coordinated Science Laboratory, where he is presently a Professor. His research interests are in the design of integrated circuits and systems for communications including low-power/high-performance VLSI architectures for error-control coding, equalization, as well as integrated circuit design. He has numerous publications in this area and holds five U.S. patents. He is also a coauthor of the research monograph "Pipelined Adaptive Digital Filters " (Boston, MA: Kluwer Academic, 1994). In 2000, he co-founded and served as the Chief Technology Officer of Intersymbol Communications, Inc., a venture-funded fabless semiconductor start-up that provides mixed-signal ICs for electronic dispersion compensation of OC-192 optical links. In 2007, Intersymbol Communications, Inc., was acquired by Finisar Corporation, Inc.

Dr. Shanbhag received the 2006 IEEE JOURNAL OF SOLID-STATE CIRCUITS Best Paper Award, the 2001 IEEE TRANSACTIONS ON VLSI Best Paper Award, the 1999 IEEE Leon K. Kirchmayer Best Paper Award, the 1999 Xerox Faculty Award, the Distinguished Lecturership from the IEEE Circuits and Systems Society in 1997, the National Science Foundation CAREER Award in 1996, and the 1994 Darlington Best Paper Award from the IEEE Circuits and Systems Society. He served as an Associate Editor for the IEEE TRANSACTION ON CIRCUITS AND SYSTEMS: PART II (1997–1999) and the IEEE TRANSACTIONS ON VLSI (1999–2002), respectively. He is currently serving on the Technical Program Committees of major international conferences such as the International Solid-State Circuits Conference (ISSCC), the International Conference on Computer-Aided Design (ICCAD), the International Symposium on Low-Power Design (ISLPED), the International Conference on Acoustics, Speech and Signal Processing (ICASSP), and others.