Error-Resilient Low-Power Viterbi Decoders

Rami A. Abdallah and Naresh R. Shanbhag*
Coordinated Science Laboratory, University of Illinois at Urbana-Champaign
1308 W Main St., Urbana, IL, USA, 61801
E-mail:[rabdall3, shanbhag]@uiuc.edu

ABSTRACT

Two 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 in absence of timing errors. In the second one, we allow data-dependent timing errors which occur whenever a critical path in the ACSU is excited. *Algorithmic noise-tolerance* (ANT) is then applied at the level of the ACSU to correct for these errors. Power reduction in this design is achieved by either overscaling the supply voltage (*voltage overscaling* (VOS)) or designing at the nominal process corner and supply voltage (*average-case design*). Power savings in the first and second design are 58% and 40% at a coding loss of 0.15 dB and 1.1 dB respectively in a IBM 130nm CMOS process.

Categories and Subject Descriptors

B.8 [**Performance and Reliability**]: Miscellaneous

General Terms

Algorithms, Design, Performance, Reliability

1. INTRODUCTION

Viterbi decoders (VDs) are widely employed in modern communication systems. The high data-rate over increasingly impaired channels results in an tremendous increase in the power consumption. 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 [1] and constitutes 76 % of the total digital processing complexity (in ops/s)[2].

*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.

ISLPED'08, August 11–13, 2008, Bangalore, India..

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.

These past works do not address the issue of energy-efficient VD design in the presence of process and voltage variations. These variations result in a wide distribution of error-free operating frequencies. Circuit level techniques such as adaptive body bias (ABB) and adaptive supply voltage (ASV) [6] can be employed to tighten the delay distributions. 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. 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 was proposed in [7], 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) whereby the statistics of data and timing errors are exploited to achieve approximate error detection and correction. Recently, a VOS-based VD [8] 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 two energy-efficient architectures for the add-compare-select unit (ACSU), a key computational kernel in the VD. In the first design, limited decision errors are introduced to decrease the critical path of the ACSU and operate it at reduced supply voltages. In the second, ANT is employed to compensate for timing errors induced by VOS and/or process variations. The application of ANT increases the latency, which can be a problem for recursive architectures such as the ACSU. Hence, we propose the use of block-interleaved pipelining (BIP) [9] to reduce the ACSU critical path delay and absorb the increased latency.

The remainder of the paper is organized as follows. Section 2 describes the impact of VOS and process variations on the the bit-error rate (BER) of VD. Section 3 introduces a relaxed comparison and describes the design of a fast ACSU. Section 4 discusses ANT in recursive architecture as a generalization of the ACSU and presents a ANT-based ACSU design. Simulation results demonstrating BER performance

and power savings under VOS and process variations for the two designs are shown in section 5.

2. ALGORITHMIC IMPACT OF CIRCUIT NON-IDEALITIES

A generic VD architecture is presented in Fig. 1. The branch metric unit (BMU) calculates the distance (usually Euclidean) referred to as branch metric (BM), between the received sample and the transmitted symbol. The ACSU recursively computes the path metric (PM) of each state, which is the log-likelihood of the best (or survivor) path and hence the most likely transmitted sequence ending at that state. 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 a separate ACSU. In this paper, we target the design of an ACSU for a 128-states rate-1/2 convolutional code. The ACSU functionality is described by the following recursive equation:

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

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 $PM_k^{(n+1)}$.

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. Furthermore, 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 the presence of overflow. The add and compare operations in (1) are implemented by an LSB-first adder followed by an LSB-first comparator (subtractor).

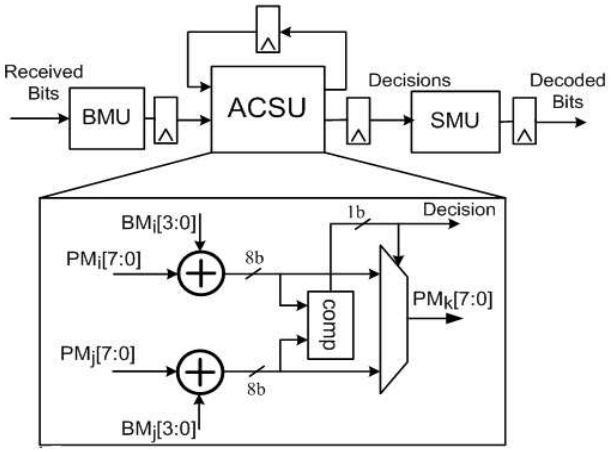

Figure 1: General architecture of Viterbi decoders.

2.1 Impact of Timing Errors

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 ACSU at different supply voltages using HSPICE in an IBM 130nm CMOS process. An RTL simulation of the VD is then carried out

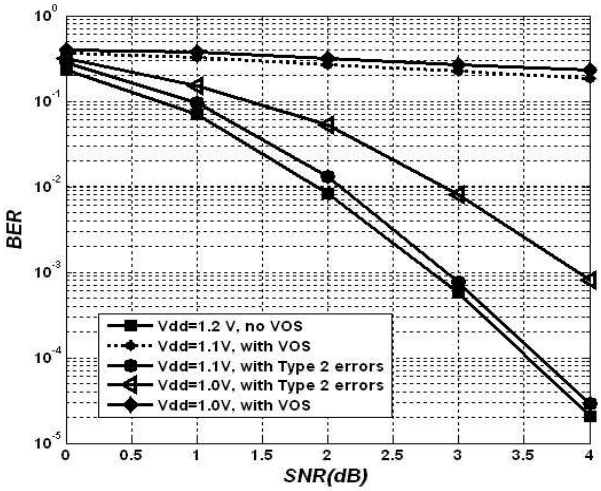

Figure 2: BER with ACSU subject to different sources of timing errors.

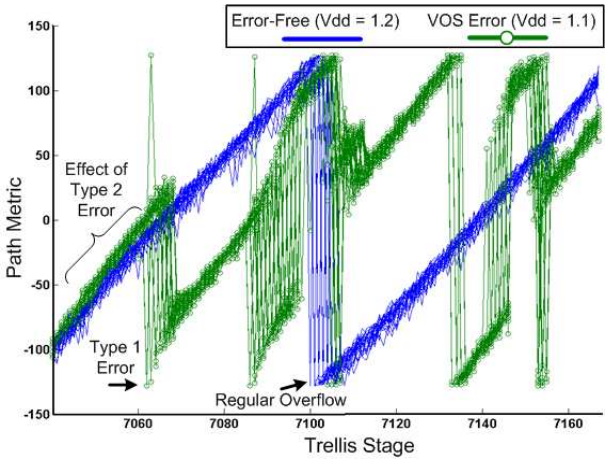

Figure 3: Path metric evolution under timing errors.

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 chosen to meet the timing constraints of 590 Mb/s at 1.2 V. An additive white Gaussian channel with binary phase-shift keying (BPSK) modulation is considered. Figure 2 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 loss in BER can be understood by observing the evolution of the PMs over time as shown in Fig. 3. One can classify timing errors into two types:

- Type 1 errors result in the computation and the selection of an incorrect PM. This event can occur when the MSBs in the adder fail to compute correctly and the incorrect PM gets selected. In such a situation, the BER is impacted severely because an incorrect PM value is propagated across multiple trellis stages leading to many incorrect decisions. The effect of Type 1 error can be observed by the abrupt shift in PMs in Fig. 3.

- Type 2 errors occur because of timing errors in the compare-select block so that the incorrect (wrong) PM gets selected. The incorrect PM is close to the correct PM when there are no errors in the adder, and it is a better estimate when there are adder errors. Thus, the impact of Type 2 errors on the BER is not as great as in the case of Type 1 errors as seen in Fig. 2 where BER curves with Type 2 errors only are shown. Clearly, a higher frequency Type 2 errors can also have a severe impact on the BER as decision errors occur more regularly on chosen paths. The effect of Type 2 errors can be observed by the slight increase in the PM's evolution rate in Fig. 3.

2.2 Impact of Process Variations

Process variations are classified into within-die (WID) and die-to-die (D2D) variations. WID variations consist of variations between different devices on the same chip. This is caused by geometric or layout dependent variations and random dopant fluctuations. On the other hand, D2D variations include variations between chips on different wafers and lots and are caused by fluctuations in process conditions such as temperature, equipment properties, and wafer placement. Timing errors induced by process variations severely affects decoder performance.

Delay distribution of various gates found in the ACSU were obtained via Monte Carlo simulations in an 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 ACSU. These instances are sampled and simulated at the RTL level in order to generate the BER curves. The clock frequency is determined by the data-rate and hence is kept fixed. The impact on BER due to timing errors induced by process variations in the ACSU is very severe as can be seen in Fig. 4.

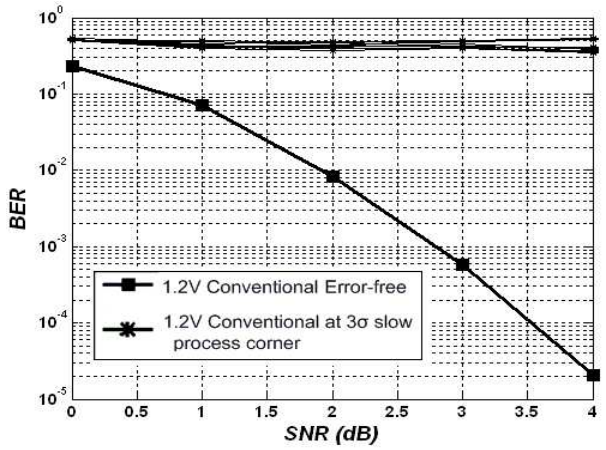

Figure 4: BER with ACSU under process variations.

3. FAST ACSU ARCHITECTURE

The critical path delay in the ACSU in Fig. 1 is dominated by the BM addition and the comparison stage as both of these computations proceed in an LSB-first manner. In what follows, a carry-save like addition and a relaxed comparator are used to decrease the delay of the BM addition and comparison stage respectively resulting in an ACSU with a reduced delay and is referred to as **FAST-ACSU**. The architecture of the **FAST-ACSU** is shown in Fig. 5. The computation in the **FAST-ACSU** is separated into an LSB and MSB part. The delay of each part is around half the original ACSU delay. Recall that the BMs and the PMs are quantized to 4-b and 8-b, respectively. During BM addition, a similar concept to carry-save addition is applied: instead of propagating the carry from the LSB part to the MSB part as in regular ACSU, the carry is saved to the second clock cycle. Therefore, instead of the PM being an 8 bit value, now it has an additional bit representing the delayed carry. This reduces the BM addition delay to around half. During comparison stage, the computation is similarly split into an LSB and MSB part while introducing limited decision errors as explained in the following relaxed comparison.

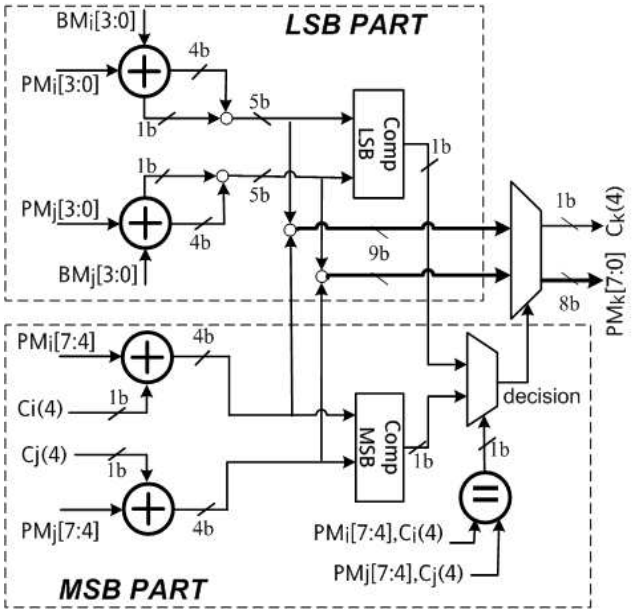

Figure 5: FAST-ACSU architecture with delayed carry and relaxed comparator.

3.1 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 part adders ($PM_i[7:4] + C_i(4)$ and $PM_j[7:4] + C_j(4)$). 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 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 as illustrated in cases A and B table 1. 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 for **FAST-ACSU** in section 5 shows only a 0.15 dB loss in coding gain.

4. ALGORITHMIC NOISE-TOLERANCE IN ACSU

In this section, we allow the ACSU to commit intermittent timing errors and we apply ANT to correct for them. However, the ACSU has a recursive structure and hence presents special challenges for the application of error-resiliency.

4.1 Algorithmic Noise-Tolerance in Recursive Architectures

An ANT-based system (see Fig. 6) includes a main block that computes correctly most of the time but makes intermittent VOS or process variation induced 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 VOS or process 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 estimator can be designed to be error-free because its complexity is much lower than that of the main block. 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. 6. Note that the application of error-resiliency via ANT increases the latency of the output $\hat{y}[n]$. In addition, the residual error $e[n] = \hat{y}[n] - y_o[n]$ will be present in the output.

Application of ANT in recursive architecture faces two main challenges. First, unlike non-recursive architectures, recursive architectures suffer from *error propagation* where the residual error at time instant n can impact future outputs (see Fig. 7(a)). Thus, more effective estimators need to be developed in order to keep the residual error within bounds.

Table 1: The two cases where the relaxed comparator is in error.

	Case A	Case B
$PM_i\,(7:4) - PM_j\,(7:4)$	1	-1
Saved carry i	0	1
Saved carry j	1	0
$PM_{i,updated}\,(7:4) - PM_{j,updated}\,(7:4)$	0	0

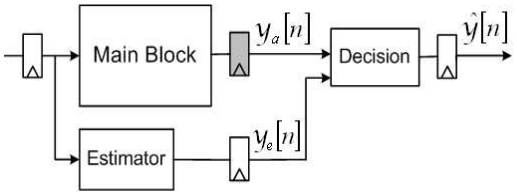

Figure 6: ANT for non-recursive architectures. The shaded latch indicates the location of timing errors.

Second, the increase in the critical path delay due to the decision block is harder to compensate for. Figure 7(a) shows that the critical path passes through the main block (A) and decision block. Thus, timing violations will impact the decision block first resulting in a severe adverse impact on the algorithmic performance. A flexibility in recursive architecture is that you can retime the feedback register to a place where it is easy to estimate the output and correct the timing errors as shown in Figure 7(b) where it maybe easier to correct errors at the output of block C than at the output of Block A. Also retiming can be used to ensure that the decision block computes correctly at the expense of errors in the main block output as shown in Figure 7(c). This scenario is acceptable because the estimator can be employed, as in case of non-recursive architectures, to generate the correct result. However, the error frequency and error magnitude will be larger than a corresponding non-recursive architecture. For VD applications, one can employ a low-complexity pipelining technique referred to as *block interleaved pipelining* (BIP) [9]. BIP can be applied as long as the input can be processed in a block-based, block-independent manner. Applying BIP followed by retiming can be employed to remove the decision block from the critical path as shown in Fig. 7(d). In order to induce block-independency for enabling the application of BIP in the VD, we introduce a known sequence (zeros) at block boundaries to force a specific state (zero) while causing a small loss in data rate.

4.2 ANT-based ACSU Architecture

Timing violations induced in the comparator stage of the ACSU are hard to correct for due to the nonlinearity of the computation under consideration. We use the relaxed comparator, which was already presented in section 3.1, to avoid timing errors due to voltage or process variations in the comparison stage. We use BIP to provide enough time for the ANT correction to take take place.

The architecture of the ANT-based ACSU is shown in Fig. 8 and is referred to as **ANT-ACSU**. The only source of timing errors in this architecture is the BM addition stage which consists of two 8-b adders. In order to correct for these errors, the **ANT-ACSU** incorporates an MSBs and an LSBs estimator of the updated PM ($PM_u[7:0]$). The MSBs estimator employs the MSBs of the input PMs ($PM_i[7:4]$ and $PM_j[7:4]$) as an estimate ($PM_{MSB,e}[3:0]$) of the MSBs of the updated PMs. The LSBs estimator generates an estimate ($PM_{LSB,e}[3:0]$) for the 4 LSBs of the updated PM. Based on the decision of the relaxed comparator, the LSBs estimator chooses the appropriate BM. If it is greater than or equal to 8, $PM_{LSB,e}[3:0]$ is set to all ones. Otherwise, $PM_{LSB,e}[3:0]$ is set equal to the LSBs of the input PM corresponding to the chosen BM.

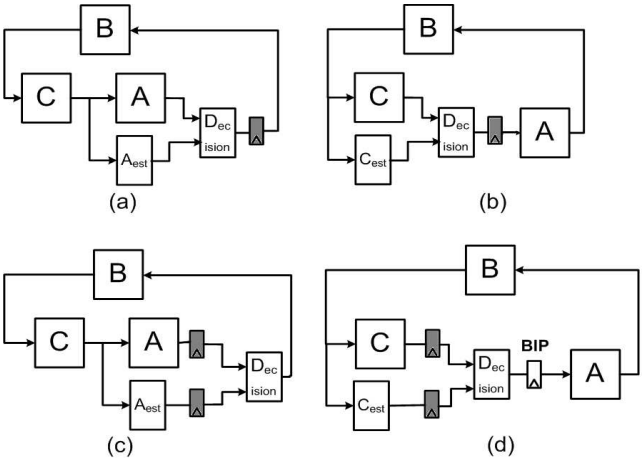

Figure 7: ANT for recursive architectures. The shaded latches indicate the location of timing errors. a) timing errors impact hard-to-correct decision block, b) retiming to ease error-correction, c) retiming to prevent errors in decision block, and d) introduction of additional latches using block interleaved pipelining (BIP).

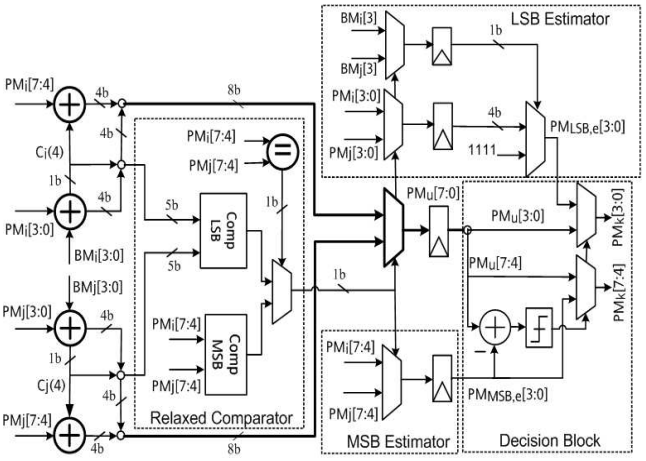

Figure 8: ANT-ACSU: BIP and register retiming are used to prevent timing errors in the correction block.

The decision block computes the difference between the MSBs of the updated PM ($PM_u[7:4]$) and $PM_{MSB,e}[3:0]$. If the absolute 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, otherwise the output PM is set to the updated PM. Table 2 indicates that there is 68% increase in gate complexity in the **ANT-ACSU** when compared to the conventional ACSU.

Table 2: Complexity estimate of ANT-ACSU.

Modules	2-input AND	2-input OR	Inverter	Transistors
Conventional ACSU	163	92	91	1202
ANT-ACSU	275	155	152	2024

5. SIMULATIONS AND RESULTS

This section describes the BER and power savings achieved by **FAST-ACSU** and **ANT-ACSU** over a conventional ACSU. The simulation set-ups under VOS and process variations are already described in sections 2.1 and 2.2.

HSPICE simulation in an IBM 130nm CMOS process shows that the **FAST-ACSU** 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 is decision errors introduced by the relaxed comparator. Figure 9 shows BER results for different ACSUs under different voltages. **FAST-ACSU** suffers only from a 0.15 dB loss in the coding gain at its nominal supply voltage (0.9 V), i.e in absence of VOS errors. A similar performance is obtained for **ANT-ACSU** at a nominal supply voltage of 1.2 V since only errors under this setting are due to the relaxed comparator. As the supply voltage is reduced from 1.2 V to 1.1 V, the conventional ACSU BER increases dramatically. The **ANT-ACSU** shows only 1.1 dB loss in the coding gain at a BER of 10^{-3} under VOS errors. Reduction of voltage beyond 0.9 V leads to a noticeably degraded performance of **ANT-ACSU** (see the BER curve for $V_{dd} = 0.8$ V) due to increased frequency of VOS errors. Note that, a **FAST-ACSU** will also show a large increase in BER as timing errors start to occur at voltages beyond its nominal voltage (0.9 V).

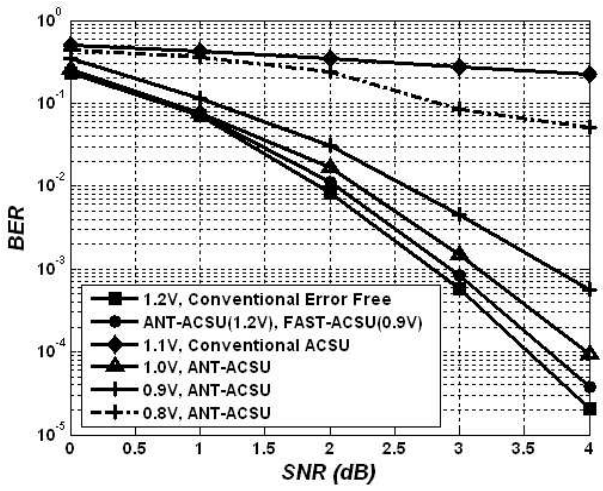

Figure 9: BER of conventional ACSU, FAST-ACSU, and ANT-ACSU at different supply voltages.

Figure 10 shows BER distribution due to process variations at 2 dB signal-to-noise ratio (SNR). Conventional ACSU suffers from a large increase in BER under process variations. ASV and ABB need to be applied to keep its delay fluctuations at the 3σ slow process corner within the operating clock frequency. Similarly, **FAST-ACSU** shows large degradation in performance due to timing errors caused by process variations. On the other hand, **ANT-ACSU** can operate at the 3σ slow process corner with a small loss in performance if voltage is slightly increased from 1.2 V to 1.3 V. Figure 11 shows the BER curves for various SNRs under process variations. The **ANT-ACSU** exhibits a coding

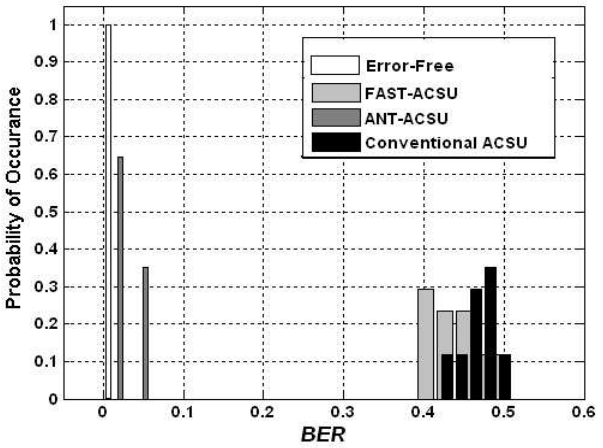

Figure 10: BER distribution at 3sigma slow corner with WID variations and at an SNR=2 dB.

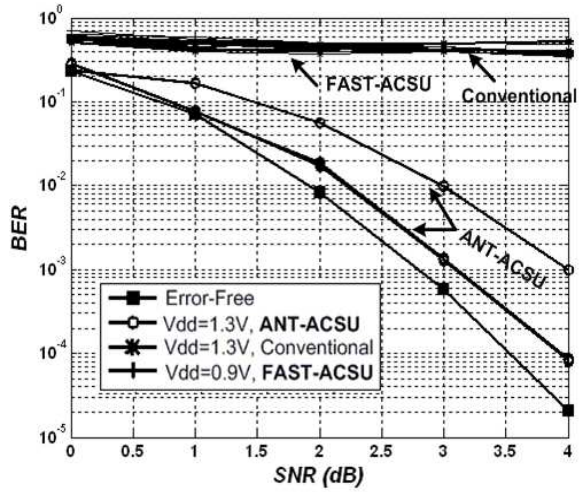

Figure 11: BER at 3σ slow corner with WID variations .

loss of 0.4 dB in approximately 66% of the cases, while the rest experience a 1.2 dB loss. The conventional ACSU and **FAST-ACSU** show a highly degraded BER at all SNRs.

HSPICE simulations are carried out to estimate the ACSU power consumption shown in Fig. 12. An input testvector of size 50 is used and clock frequency is fixed to meet the target data rate. At the nominal process corner, **ANT-ACSU** including correction overhead achieves 40% power savings under VOS while a **FAST-ACSU** exhibits 58% power savings. At the 3σ slow process corner, ASV and ABB are applied to the conventional ACSU and **FAST-ACSU** to operate at the target data rate. Under this setting, the 1.3 V **ANT-ACSU** exhibits 25% power savings over the conventional ACSU where as the **FAST-ACSU** achieves 42%. In terms of power savings, **FAST-ACSU** shows the largest savings. However, it is not resilient to timing errors induced by VOS or process variations. On the other hand, **ANT-ACSU** is more resilient to timing errors but exhibits lesser power savings due to correction overhead.

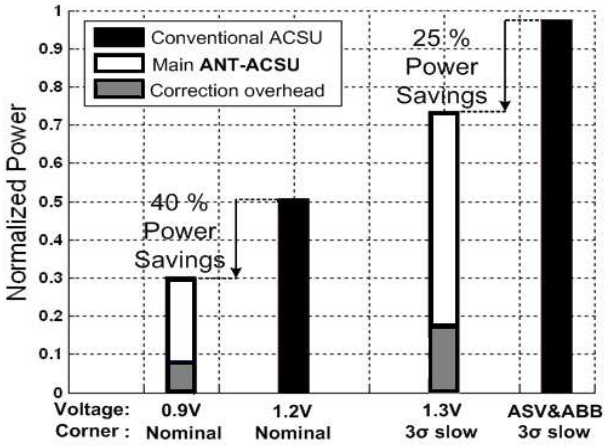

Figure 12: Power consumption under VOS and process variations.

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. 965Ű972, 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, pp. 826-835, June, 2000.

[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] 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.

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

[8] 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.

[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.