

ERROR-RESILIENT LOW-POWER VITERBI DECODERS

Rami A. Abdallah

*Coordinated Science Laboratory
1308 West Main Street, Urbana, IL 61801
University of Illinois at Urbana-Champaign*

REPORT DOCUMENTATION PAGE			Form Approved OMB NO. 0704-0188	
<small>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comment regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.</small>				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE November 2008		3. REPORT TYPE AND DATES COVERED
4. TITLE AND SUBTITLE Error-Resilient Low-Power Viterbi Decoders			5. FUNDING NUMBERS	
6. AUTHOR(S) Abdallah, Rami A.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Coordinated Science Laboratory University of Illinois 1308 W Main St Urbana, IL 61801			8. PERFORMING ORGANIZATION REPORT NUMBER UIIU-ENG-08-2218 DAC 110	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) GSRC University of California, Berkeley 545D Cory Hall Berkeley, CA 94729-1768			10. SPONSORING/MONITORING AGENCY REPORT NUMBER TI POB 6660'199, MS8649 Dallas, TX 75266-0199	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) 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 in absence of timing errors. In the rest, 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 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 concepts are employed to develop efficient estimators for error-correction. The first is based on reduced-precision redundancy and the second on state clustering. Power savings achieved in the presence of VOS and process variations can reach up to 71% and 62%, respectively, at a loss of 0.8 dB and 0.6 dB in coding gain in a IBM 130-nm CMOS process.				
14. SUBJECT TERMS Viterbi decoder; algorithmic noise-tolerance; process variations; voltage overscaling; error resiliency			15. NUMBER OF PAGES 70	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED		20. LIMITATION OF ABSTRACT UL

© 2008 Rami A. Abdallah

ERROR-RESILIENT LOW-POWER VITERBI DECODERS

BY

RAMI A. ABDALLAH

B.Eng., American University of Beirut, 2006

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2008

Urbana, Illinois

Adviser:

Professor Naresh R. Shanbhag

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 (AC-SUs) 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 rest, 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 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 concepts are employed to develop efficient estimators for error-correction. The first is based on *reduced-precision redundancy* and the second on *state clustering*. Power savings achieved in the presence of VOS and process variations can reach up to 71% and 62%, respectively, at a loss of 0.8 dB and 0.6 dB in coding gain in a IBM 130-nm CMOS process.

To my parents and siblings for their love and continual support.

ACKNOWLEDGMENTS

I would like to thank my adviser, Professor Naresh R. Shanbhag, for his continuous guidance, support, and invaluable comments. I also would like to thank my family and friends for their much appreciated encouragement.

TABLE OF CONTENTS

LIST OF TABLES	vii
LIST OF FIGURES	viii
LIST OF ABBREVIATIONS	x
CHAPTER 1 INTRODUCTION	1
1.1 Motivation	1
1.2 Background	3
1.2.1 Viterbi algorithm and architecture	3
1.2.2 Previous work on low power Viterbi decoders	12
1.2.3 Algorithmic noise-tolerance	14
1.3 Thesis Organization	16
CHAPTER 2 TIMING ERRORS AND ERROR RESILIENCY IN VITERBI DECODERS	18
2.1 Viterbi Decoder Implementation	18
2.2 Simulation Procedure under Timing Errors	20
2.2.1 Voltage overscaling	20
2.2.2 Process variations	21
2.3 Impact of Timing Errors	22
2.4 Algorithmic Noise-Tolerance in Recursive Architectures	24
2.4.1 Algorithmic noise-tolerance in general recursive architectures	26
2.4.2 Algorithmic noise-tolerance in ACSU	28
2.5 Analysis of Timing Errors Effect on BER	30
2.5.1 BER bound in absence of timing errors	30
2.5.2 BER bound under algorithmic noise tolerance	32
CHAPTER 3 LOW-POWER ERROR-RESILIENT ACSU ARCHITECTURES	35
3.1 Fast ACSU Architecture	35
3.1.1 Relaxed comparison	37
3.2 Redundancy-ANT ACSU Architecture	38
3.3 State-Clustered ANT ACSU Architecture	40
3.3.1 Estimation schemes	41
3.3.2 State clustered architecture design	42

3.4	Simulations and Results	45
3.4.1	Voltage overscaling	45
3.4.2	Process variations	47
3.4.3	Power savings	51
CHAPTER 4 CONCLUSIONS		54
REFERENCES		55

LIST OF TABLES

2.1	Probability of timing error (p_{vos}) at different supply voltage	34
3.1	The two cases where the relaxed comparator is in error	37
3.2	Complexity estimate of RA-ACSU	40
3.3	Complexity estimate of state clustering architectures (two states per cluster)	43
3.4	BER at 3σ slow process corner with WID variations at 2-dB SNR . .	50

LIST OF FIGURES

1.1	A WLAN receiver and the distribution of power consumption.	2
1.2	Distribution of frequency and leakage in microprocessors in a wafer. .	2
1.3	Convolution encoder with $r = 1/2$, $K = 3$, $g^0 = (1, 1, 1)$, and $g^1 = (1, 1, 0)$	5
1.4	Radix-2 trellis representation of rate $1/2$ 4-state CC.	6
1.5	General architecture of Viterbi decoders.	8
1.6	Detailed architecture of conventional ACSU with 8-bit PM and 4-bit BM.	9
1.7	Radix-4 trellis representation of rate $1/2$ 4-state CC.	10
1.8	Radix-4 ACSU for rate $1/2$ CC.	11
1.9	RAM based SMU for VDs.	12
1.10	Register exchange method for decision decoding in VDs assuming 4 states.	13
1.11	ANT for nonrecursive architectures. The shaded latch indicates the location of timing errors.	15
2.1	Convolution encoder with $r = 1/2$, $K = 8$, $g^0 = 247_8$, and $g^1 = 371_8$ (128 states).	19
2.2	Simulation procedure under (a) voltage overscaling and (b) process variations.	20
2.3	Delay distribution due to WID of 1.2-V ACSU at nominal and 3σ slow corner.	22
2.4	Decoding performance with ACSU subject to different sources of timing errors.	23
2.5	Path metric evolution under timing errors.	23
2.6	Decoding performance with ACSUs under WID variation at the 3σ slow corner.	25
2.7	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) introduction of additional latches using block interleaved pipelining (BIP).	25

2.8	Pipelined BIP architecture. (a) Block-processing architecture, (b) parallel architecture, (c) block-interleaved architecture, and (d) pipelined block-interleaved architecture.	27
2.9	Throughput and latency of (a) conventional ACSU, (b) retimed ACSU, (c) relaxed ACSU, and (d) fast ACSU (FACSU).	29
2.10	BER bounds with ANT-VD under timing errors.	34
3.1	FACSU architecture with delayed carry and relaxed comparator. (a) General architecture, (b) detailed architecture (PM selection Mux not included).	36
3.2	Decoding performance of FACSU. Errors are only caused by the relaxed comparison.	38
3.3	RA-ACSU: BIP and register retiming are used to prevent timing errors in the correction block.	39
3.4	State clustering for 128-state rate-1/2 code with 2-state cluster where states share the same set of previous states.	41
3.5	State clustering for 128-state rate-1/2 code with 4-state cluster where states share the same set of previous states.	42
3.6	State clustering with the introduction of a redundant stage.	43
3.7	State clustering architecture with two states X and Y. (a) SC-FACSU architecture, (b) SCS-FACSU architecture.	44
3.8	Delay of conventional ACSU and FACSU at different supply voltages.	46
3.9	BER of conventional ACSU, FACSU, and RA-ACSU at different supply voltages.	46
3.10	BER of a conventional ACSU, SC-FACSU, and SCS-FACSU at different supply voltages using radix-2 computation (2-state cluster).	47
3.11	BER of a conventional ACSU, SC-FACSU, and SCS-FACSU at different supply voltages using radix-4 computation (4-state cluster).	48
3.12	BER distribution at 3σ slow corner with WID variations and at SNR = 2 dB.	49
3.13	BER distribution with radix-2 computation at a 3σ slow corner with WID variations and SNR = 2 dB.	49
3.14	BER at 3σ slow corner with WID variations for RA-ACSU.	50
3.15	BER at 3σ slow corner with WID variations for state clustering.	51
3.16	Power consumption per ACSU under VOS and process variations.	52
3.17	Average power consumption per ACSU under VOS and process variations.	53

LIST OF ABBREVIATIONS

ACSU	add-compare-select unit
ANT	algorithmic noise tolerance
BER	bit error rate
BM	branch metric
BMU	branch metric unit
CC	convolution code
CMOS	complementary metal-oxide semiconductor
D2D	die-to-die
FACSU	fast add-compare-select unit
ISR	input subsampling replica
LSB	least significant bit
MLSE	maximum likelihood sequence estimation
MSB	most significant bit
PM	Path metric
RA-ACSU	redundancy ANT Add-compare-select unit
RPR	reduced precision redundancy
SC-FACSU	state clustered architecture with fast ACSU as estimator
SCS-ACSU	state clustered architecture with a redundant ACSU as estimator
SMU	survivor memory unit
SNR	signal-to-noise ratio

SRAM	static random access memory
VD	Viterbi decoder
VLSI	very large-scale integration
VOS	voltage overscaling
WID	within die
WLAN	wireless local area network

CHAPTER 1

INTRODUCTION

1.1 Motivation

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, an 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 as shown in Figure 1.1 [1]. This also constitutes 76% of the total digital processing complexity (in ops/s)[2].

Past works on VDs do not address the issue of energy-efficient robust VD design in the presence of process, voltage, and temperature (PVT) variations. These variations result in a wide distribution of error-free operating frequencies. This requires a reliance on worst-case design and leads to high power consumption. For example, Figure 1.2 [3] shows that there is around 30% variation in chip frequency and 20X variation in chip leakage in a 180 nm process [3]. This variation can reach 50 – 60% in 65-nm nodes and below according to the International Technology Roadmap for Semiconductors [4]. Circuit level techniques such as adaptive body bias (ABB) and adaptive supply voltage (ASV) [5] can be employed to tighten the delay distributions.

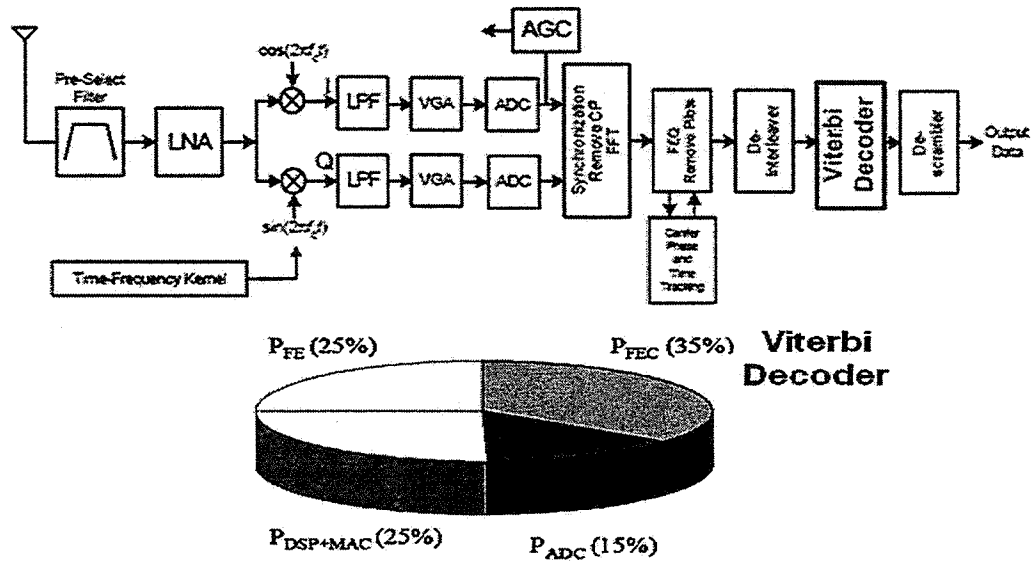


Figure 1.1 A WLAN receiver and the distribution of power consumption.

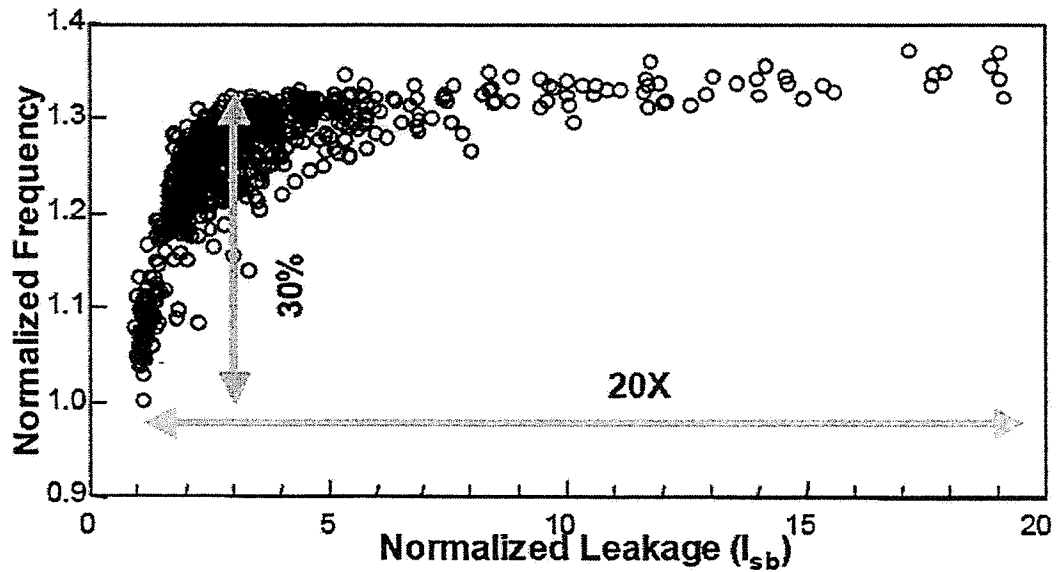


Figure 1.2 Distribution of frequency and leakage in microprocessors in a wafer.

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 [6], 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.

In this work, we study the resiliency of the conventional Viterbi algorithm and architecture to hardware errors induced by PVT variations. We propose new energy-efficient architectures based on the concept of error-resiliency to enable operation at nominal or reduced voltage while maintaining decoding performance. Next, we present a background on Viterbi decoding and ANT.

1.2 Background

1.2.1 Viterbi algorithm and architecture

The Viterbi algorithm, proposed by Viterbi in 1967, is an efficient procedure for solving maximum likelihood sequence estimation (MLSE) problems [7]. Decoding of convolution codes (CCs) is one. CC introduces memory dependency among information bits to improve error correction capability. The convolution encoder is a shift

register/state machine whose output bits are a combination of the input bits and the stored bits in the shift register. The CC is characterized by its rate (r), constraint length (K), and generator polynomials (g^i). The rate is expressed as the number of input bits divided by the number of output bits. The constraint length denotes the length of the code, i.e., how many stages the current output bits depend on. The generator polynomials express how each output bit is related to the previous bits. For example, Figure 1.3 shows a rate 1/2 CC encoder that generates two output bits (c_n^0, c_n^1) denoted by c_n at each time index n as a function of the input information bits (b_n) and the stored bits in the shift register (b_{n-1}, b_{n-2}). The constraint length is 3 since c_n depends on three stages (b_n, b_{n-1} , and b_{n-2}). The generator polynomials $g^0(z)$ and $g^1(z)$ for c_n^0 and c_n^1 are, respectively:

$$\begin{aligned} g^0(z) &= 1 + z + z^{-2} \\ g^1(z) &= 1 + z^{-1} \end{aligned} \tag{1.1}$$

The generator polynomials are usually denoted by the coefficient vector in binary or octal representation. For the CC under consideration, $g^0 = (1, 1, 1)$ and $g^1 = (1, 1, 0)$ in binary format or $g^0 = 4$ and $g^1 = 6$ in octal format. A closely related parameter to the constraint length is the memory length of the code denoted by m , which is the length of the shift register. The memory length determines how many states the encoder can have and is equal to 2^m . For our example, $m = 2$ and there are 4 possible states s_n for the encoder corresponding to the bits (b_{n-1}, b_{n-2}) in the shift register.

The output bits are transmitted over a noisy channel. The received noisy sample or codeword corresponding to (c_n^0, c_n^1) at time n is denoted by r_n . Assuming N symbols are transmitted and knowing the initial state and structure of the encoder, the problem of estimating the information bit sequence $\underline{\mathbf{b}} = [b_0, b_1, \dots, b_{N-1}]$ is the

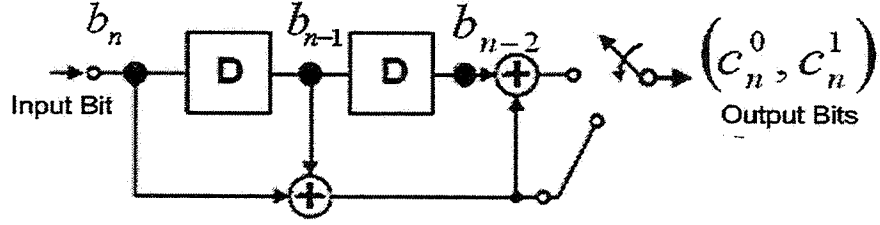


Figure 1.3 Convolution encoder with $r = 1/2$, $K = 3$, $g^0 = (1, 1, 1)$, and $g^1 = (1, 1, 0)$.

same as estimating the transmitted sequence $\underline{\mathbf{c}} = [(c_0^0, c_0^1), (c_1^0, c_1^1), \dots, (c_{N-1}^0, c_{N-1}^1)] = [c_0, c_1, \dots, c_{N-1}]$ or estimating the sequence of state transitions $\underline{\mathbf{s}} = [s_0, s_1, \dots, s_{N-1}]$.

Maximum likelihood symbol-by-symbol detection will estimate the information bit at time n by just using the single received sample at the corresponding time index. On the other hand, an MLSE uses the received sequence of N samples to estimate the transmitted sequence of bits and thus exploits the memory dependency in the CC. Assuming a zero mean additive white Gaussian noise (AWGN) channel with standard deviation σ and N symbols being transmitted and denoting the received sequence of samples by $\underline{\mathbf{r}} = [r_0, r_1, \dots, r_{N-1}]$, MLSE is given by

$$\begin{aligned} \hat{\underline{\mathbf{b}}} &= \arg \max_{\underline{\mathbf{c}}} P(\underline{\mathbf{r}} | \underline{\mathbf{c}}) = \arg \max_{\underline{\mathbf{c}}} \prod_{i=0}^{N-1} P(r_i | c_i) \\ &= \arg \max_{\underline{\mathbf{c}}} \prod_{i=0}^{N-1} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2} \|r_i - c_i\|^2} \end{aligned} \quad (1.2)$$

Taking the logarithm and factoring out common terms,

$$\Rightarrow \hat{\underline{\mathbf{b}}} \propto \arg \min_{\underline{\mathbf{c}}} \sum_{i=0}^{N-1} \|r_i - c_i\|^2 \quad (1.3)$$

The complexity of a brute force MLSE is exponential in the length of the decoding sequence N since according to (1.3) it will search for the minimum metric by iterating

through all possibilities of the codeword sequence \underline{c} . The Viterbi algorithm is an iterative efficient MLSE algorithm that limits the search space, leading to a complexity that is linear in N . The encoding or decoding process can be represented by a time indexed trellis as shown in Figure 1.4.

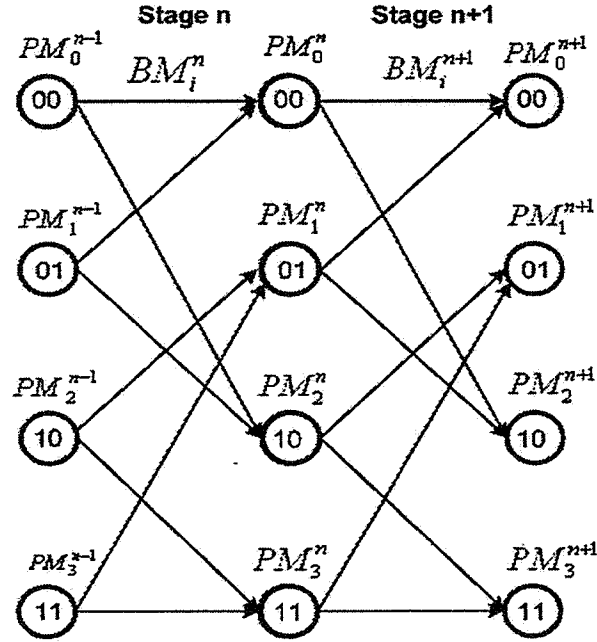


Figure 1.4 Radix-2 trellis representation of rate 1/2 4-state CC.

The trellis has 4 states representing the encoder states. Each state has two branches emanating from it. These represent possible transitions depending on the input bit $b[n]$ being a 0 or a 1. Each branch is characterized by a branch metric (BM) that indicates the distance between the received samples and the expected codeword. Under AWGN and using (1.3), $BM = \|r_i - c_i\|^2$. 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. A brute force MLSE will find the PMs of all paths in the trellis up to time index $N - 1$ and select the path with minimum PM. The Viterbi algorithm, on the other hand, limits the number of paths to consider at each time index n by discarding paths that do not fall on the overall best path. This is done recursively

by keeping for each state only the path reaching it with the minimum PM (referred to as survivor path for each state): 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 Figure 1.4, there are two paths entering each state. The recursive update equation for each state is given by

$$PM_k^{(n+1)} = \min \left(PM_i^{(n)} + BM_i^{(n)}, PM_j^{(n)} + BM_j^{(n)} \right) \quad (1.4)$$

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. After finding the best path in the trellis at time N , a trace-back operation is performed along the best path to decode the information bits from the sequence of state transitions. 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.

A generic VD architecture, assuming 4-b BMs and 8-b PMs, is presented in Figure 1.5. The main modules are the branch metric unit (BMU), the ACSU, and survivor memory unit (SMU).

The branch metric unit (BMU) generates BMs for all the edges and is shared among all states. The number of different BMs to compute is equal to the number of possible code symbols c . The BM expression can be simplified by factoring out common terms or using simpler distance metrics than the Euclidean, such as Hamming distance.

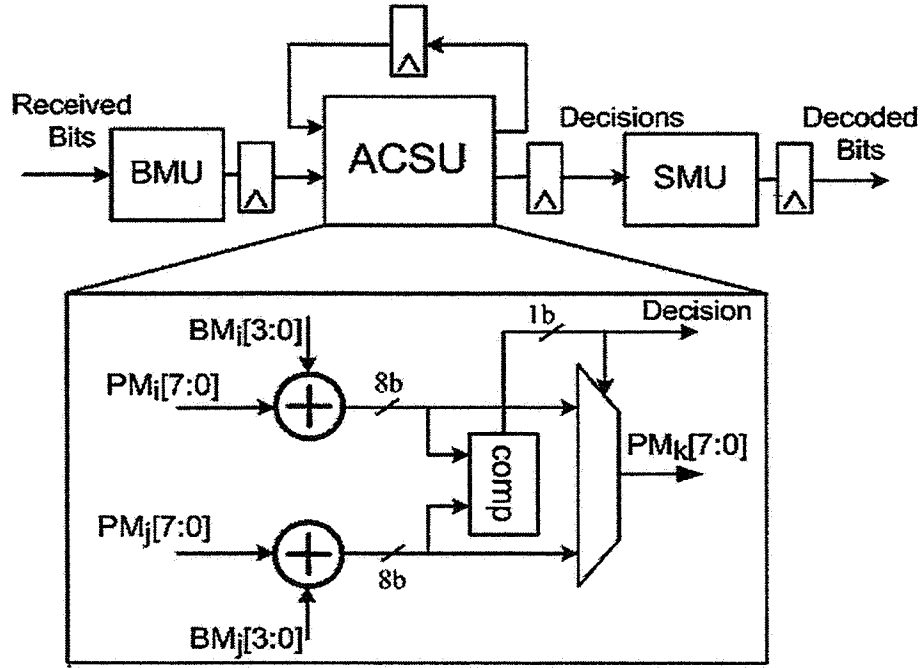


Figure 1.5 General architecture of Viterbi decoders.

The ACSU recursively computes the PM of each state according to (1.4) and is the main computational kernel in VD. A state-parallel ACSU is commonly employed for high data-rate applications where the PM of each state is updated in a separate ACSU. The detailed conventional architecture of an ACSU is shown in Figure 1.6 where a least significant bit (LSB) first addition followed by an LSB first comparison is applied. The critical path delay is dominated by the carry ripple across an 8-bit adder.

The feedback structure of the ACSU makes it the bottle-neck for high throughput applications since no low overhead pipelining can be applied. Different architectures have been proposed to break this bottle-neck or increase throughput. In [8], bit-level pipelining is proposed where pipelining registers are introduced at the bit level by using carry-save adders and comparators. In [9], most-significant-bit (MSB) first addition and selection is proposed to prevent carry propagation. Similarly in [10], an improved MSB-first ACSU is proposed. Although these architectures achieve high

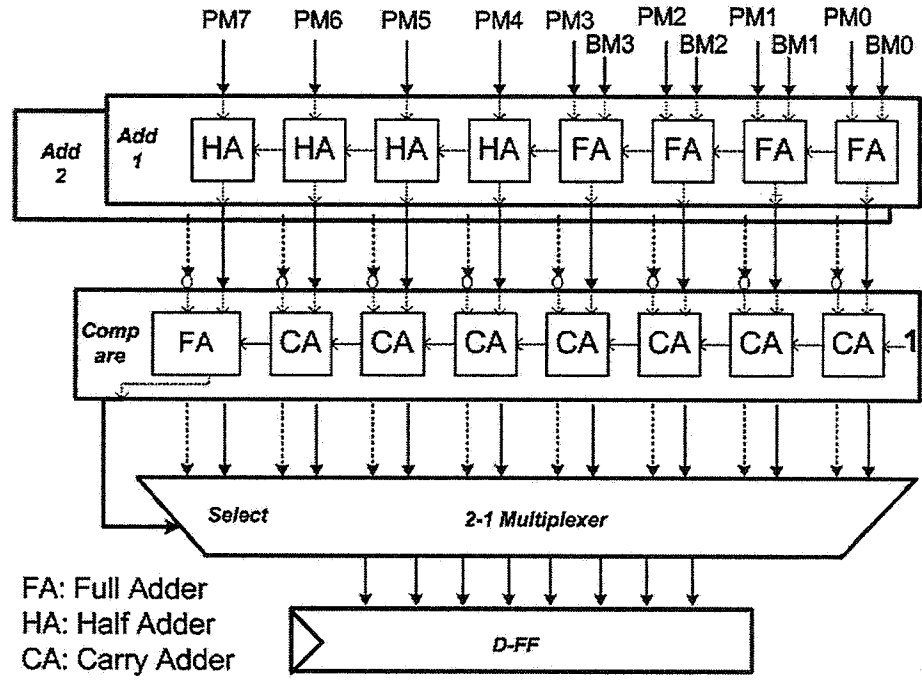


Figure 1.6 Detailed architecture of conventional ACSU with 8-bit PM and 4-bit BM.

throughput, the overhead induced, especially at the comparison stage, is usually large, leading to increased area and power consumption. Another alternative for increased throughput is higher radix processing [11]. The regular trellis in Figure 1.4 is called a radix-2 trellis. Under higher radix processing, more than one section of the trellis is combined into a single section to increase decoding throughput. Figure 1.7 shows one section of a radix-4 trellis where two stages of the radix-2 trellis in Figure 1.4 are combined into a single stage. The radix-4 ACSU needs now to compare 4 updated PMs and selects the minimum. By processing two stages at the same time the throughput is increased by a factor of 2, but the problem is that the ACSU critical path delay is increased since more computations are needed. In [12], six comparisons are done in parallel for radix-4 ACSU to maintain a critical path delay comparable to a radix-2 ACSU as shown in Figure 1.8. In general 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 PMs to compute and compare in the ACSU increases by a factor of $\log(k)$.

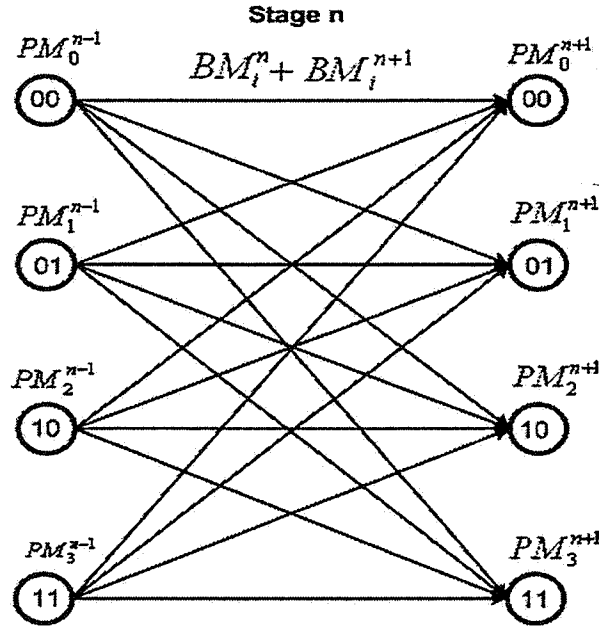


Figure 1.7 Radix-4 trellis representation of rate 1/2 4-state CC.

The survivor memory unit (SMU) keeps track of the survivor path of each state and traces back the best path. The design of the SMU depends on the path convergence property of the Viterbi algorithm which says that the survivor paths of all states will tend to converge to the same path at $L = 5K$ stages backwards. Therefore, instead of comparing the PMs of all states to find the best path to start decoding, trace-back can start from any state up to L stages back and then start decoding. The general structure of an SMU is shown in Figure 1.9. In each column, decisions from the ACSUs corresponding to all states are written. Three main operations take place in the SMU:

- Write where new decision vectors are written.
- Trace-back where trace-back starts from any state up to L stages back.
- Read/decode where decisions are read and information bits are decoded.

The memory is implemented in a circular buffer format; i.e., when the writing operation reaches the end of the memory it starts writing decisions at its starting address.

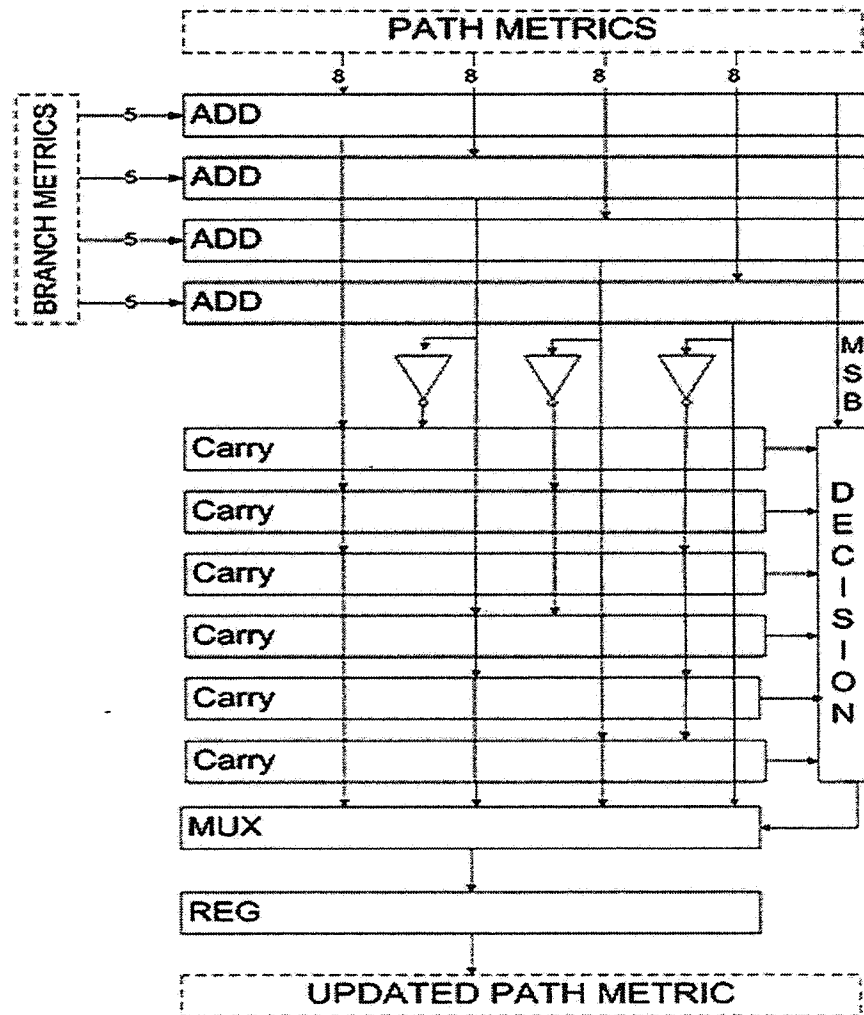


Figure 1.8 Radix-4 ACSU for rate 1/2 CC.

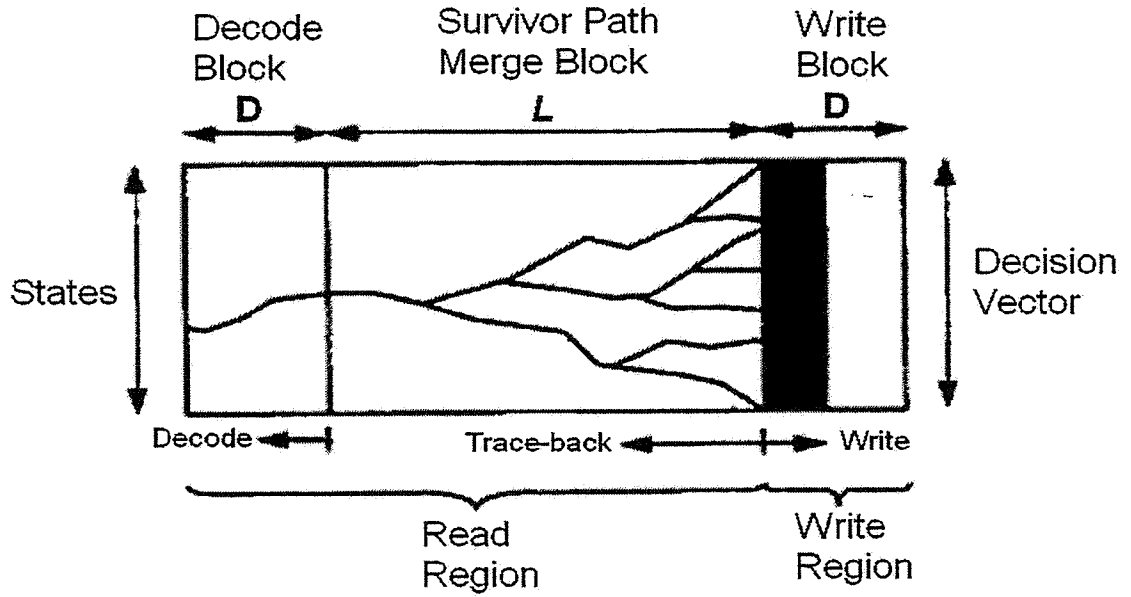


Figure 1.9 RAM based SMU for VDs.

The number of columns of the SMU and read/write rate are designed to ensure that new decisions are not written over decisions that have not been decoded yet, as discussed in [13].

Another form of SMU is the register exchange method where information bits are stored in a set of shift registers and multiplexed according to the decisions from the ACSUs (see Figure 1.10). The memory span of the registers is L stages before reading the decision due to the path converge property. The register exchange method is fast but consumes much more power than the RAM based SMU due to the continuous register content switching. This makes the RAM based SMU a more widely employed option across different applications than the register exchange method.

1.2.2 Previous work on low power Viterbi decoders

Low power Viterbi decoders are a well studied subject in literature. Power reduction by scarce state transition (SST) decoding was proposed in [14]. The idea is to perform an initial hard decision symbol-by-symbol decoding on the received samples and then

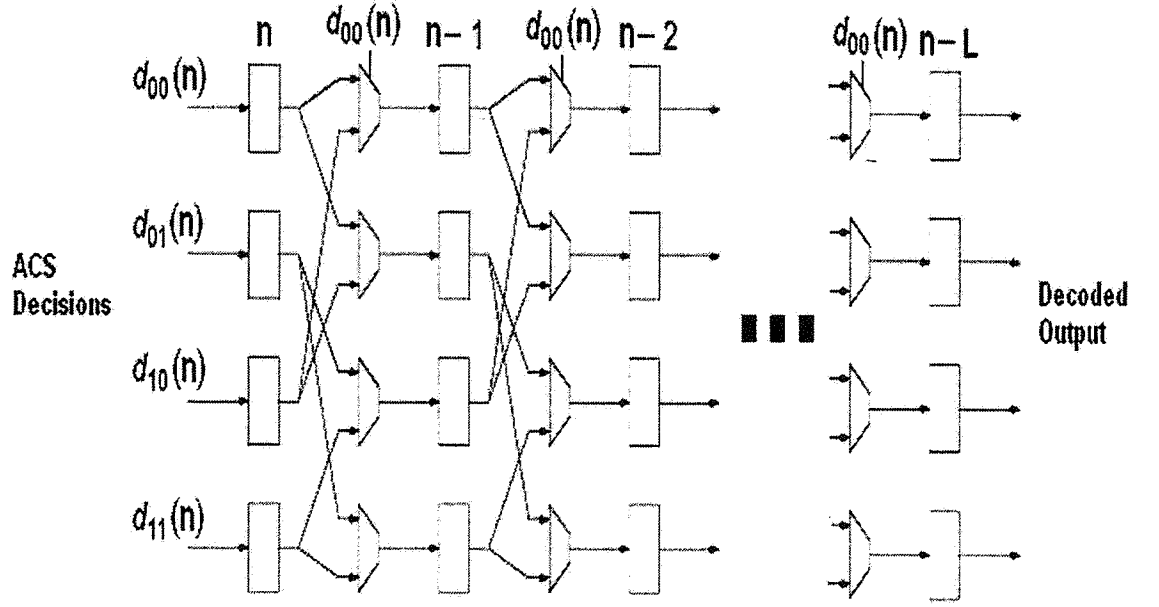


Figure 1.10 Register exchange method for decision decoding in VDs assuming 4 states.

subtract this decision from the received samples to obtain a conventionally coded error sequence \underline{e} that encodes the difference between the received samples and the hard decision based samples. The VD is then applied on \underline{e} to decode the errors in the hard decision step. Therefore, in SST decoding the VD operates on the errors that were added by the channel and were not captured by the hard-decision decoding step. Thus, most of the time the path through the zero states is the optimum path since the probability of an error appearing is relatively small, especially at high signal-to-noise ratio (SNR). Simulations in [14] show that the PM in all the ACSUs is small, which reduces switching activity, and shorter SMU can be used since most of the time the best path is known. Reduced state sequence detection (RSSD) is another low power VD technique that was proposed in [15]. In RSSD, the trellis states are grouped into a smaller number of representative states so that VD operates on a smaller number of survivor paths. Adaptive VDs, which are sometimes referred to as limited-search trellis decoders, present another alternative for reduced power decoders. The idea

here is to limit the number of survivor paths to keep at each stage. Two main algorithms are proposed in this area: the M-algorithm [16] and the T-algorithm [17]. The M-algorithm keeps the best M paths at each stage where M is certainly less than the number of states. The T-algorithm retains only paths that are within a threshold T from the best path. Different VLSI implementations for these types of limited-search decoders are presented in [18], [19], [20]. The threshold T or the number of paths M is chosen small enough to overcome the overhead associated with comparing and sorting the paths before pruning, and this may greatly affect BER. In [21], the overhead of searching for the most likely path at each stage can be removed by using SST since the zero state will most of the time hold the best path. A fourth technique for power saving is SMU buffering where memory locality during trace-back is exploited to reduce memory access [22]. Recently a new low power ACSU for VDs was proposed in [23] by introducing clock-skew aware scheduling. By realizing the importance of each bit in the ACSU, clock-skew is introduced to provide more time for the important bits to finish. All these past works do not address the issue of energy-efficient robust VD design in presence of hardware errors induced by PVT variations, which will be the subject of this work.

1.2.3 Algorithmic noise-tolerance

Traditionally, circuits are designed to be error free at the worst-case PVT variations. They are operated at a critical supply voltage so that the timing constraints imposed by the critical path delays are met. ANT maintains circuit operation under timing errors and achieves significant power savings despite the overhead.

An ANT-based system (see Figure 1.11) 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

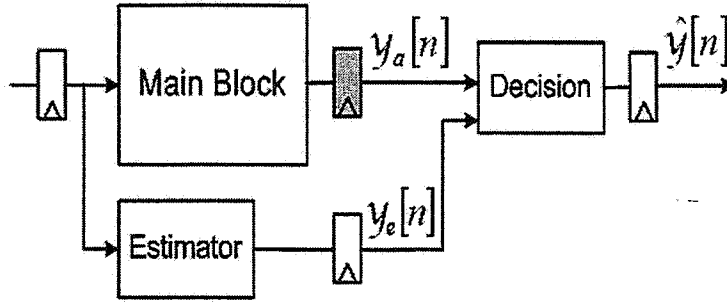


Figure 1.11 ANT for nonrecursive architectures. The shaded latch indicates the location of timing errors.

errors. Thus,

$$y_a[n] = y_o[n] + \eta[n] \quad (1.5)$$

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 reduced critical path delay. This ensures that the estimator output is error-free at all times while the main block is suffering from intermittent timing errors. Estimation techniques include linear and adaptive prediction [6], [24], reduced precision redundancy (RPR) [25], and input subsampling replica (ISR) [26]. Under prediction, the correlations among signals are exploited to provide a reduced complexity estimate for the output or to adaptively cancel the error component $\eta[n]$. In RPR, the estimator has the same architecture as the main block but with reduced precision. The precision of the estimator is carefully chosen to balance the loss in performance due to estimation error and the critical path delay of the estimator block. In ISR, the estimator operates on a reduced set

of samples from the main block output to provide a close estimate to the main block output and is commonly applicable to applications that require averaging.

In a least-significant bits (LSBs) first computation, timing errors in the main block output occur in the most significant bits (MSBs) of the result, thus, a large deviation between the main block output and the estimated output will be observed when an error occurs. A simple decision block can be used to detect and correct errors in the main block output as follows:

$$\hat{y}[n] = \begin{cases} y_a[n] & \text{if } |y_a[n] - y_e[n]| < T_h \\ y_e[n] & \text{otherwise} \end{cases} \quad (1.6)$$

where T_h is a predefined threshold, and $\hat{y}[n]$ is the corrected final output shown in Figure 1.11.

1.3 Thesis Organization

This chapter discussed the increased impact of PVT variations on the future of CMOS technology, the high power consumption of VDs, and the need for error-resilient designs. Background on Viterbi algorithms and architecture, and a literature review of existing low power VD design were presented. ANT and the different devised estimation schemes were also introduced.

Chapter 2 presents the architectural implementation of the selected VD and the followed simulation procedure for process variations and voltage overscaling in this work. The resiliency of the Viterbi algorithm and the conventional architecture to timing errors introduced by process variations and voltage overscaling is then analyzed and studied. Finally, challenges and solutions to applying error-resiliency to the Viterbi decoder and recursive architectures in general are proposed.

Chapter 3 proposes three low-power designs for the add-compare-select units (AC-

SUs), the main computational kernel in the VD. Simulations of the decoding performance for the novel architectures and the achieved power savings under voltage overscaling and process variations are presented.

Chapter 4 concludes the study of error-resiliency in VDs and outlines future directions.

CHAPTER 2

TIMING ERRORS AND ERROR RESILIENCY IN VITERBI DECODERS

In this chapter, we study the resiliency of the Viterbi algorithm to timing errors and challenges of applying ANT. Two main sources of errors will be used: voltage over-scaling and process variation. First, the VD designed in this work and the simulation procedure under timing errors are discussed. After that, we study the effect of timing errors following the explained simulation procedure on the decoding performance. We propose architectural techniques to apply ANT for general recursive architecture and for the ACSUs of the VD in particular. Finally, the effect of timing errors with ANT on the BER performance of the VD under study is analyzed.

2.1 Viterbi Decoder Implementation

The CC used in this work is chosen to provide enough protection against channel errors and is used in most modern communication systems. The CC is a rate $1/2$ with constraint length $K = 8$ and generator polynomials $g^0 = 247$ and $g^1 = 371$ in octal. The encoder is shown in Figure 2.1. The VD for this code is therefore a 128-state decoder and the required throughput is designed to be 590 Mb/s to be inline with most of state-of-the-art requirements. The nominal decoder is designed at 1.2-V IBM 130-nm CMOS process.

An additive white Gaussian channel with binary phase-shift keying (BPSK) modulation is considered. The received samples are quantized using three bits. The Euclidean distance measure for the BM under AWGN is reduced to a linear distance

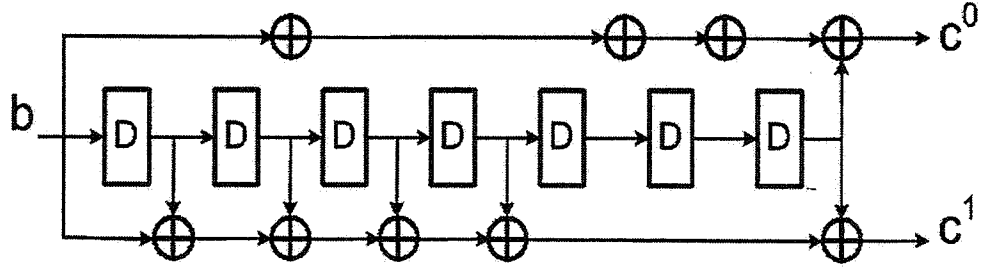


Figure 2.1 Convolution encoder with $r = 1/2$, $K = 8$, $g^0 = 247_8$, and $g^1 = 371_8$ (128 states).

measure under BPSK by factoring out common terms as discussed in [27]. Therefore with rate $1/2$, i.e, 2 symbols per codeword, the BM is a 4-bit number. Instead of rescaling the PM of all states after a specific number of stages in the trellis, the PM precision is chosen based on the modular property of two's complement arithmetic that ensures the correctness of the add and compare operations in presence of PM overflow [28]. In two's complement, the final result in a series of successive arithmetic operations is correct inspite of overflow in intermediate stages as long as there is enough precision to represent the final result. The ACSU consists of two stages: addition and subtraction (comparison). Therefore, the result of comparison is correct if there is enough precision to represent the difference between merging PMs. In the trellis, any two paths at any time index are guaranteed to pass through the same state after at most K stages. Thus, the maximum path difference ΔPM is bounded as follows:

$$|\Delta PM| \leq BM_{\max} K \quad (2.1)$$

Representing ΔPM in two's complement requires $\lceil BM_{\max} K \rceil + 1$. For the VD decoder under consideration and the 4-bit BM, the required PM precision is 8 bits. With higher radix processing, the precision of BM and PM will increase. For example in radix 4, two radix-2 BMs are combined to form a new 5-bit BM and the required precision for PM is 9 bits. Fixed point C and RTL level simulation are carried out to

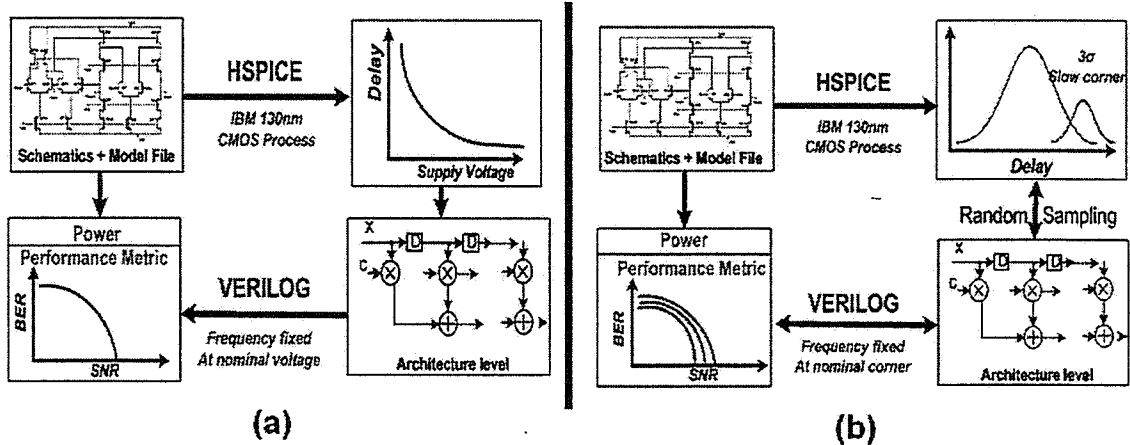


Figure 2.2 Simulation procedure under (a) voltage overscaling and (b) process variations.

simulate the decoder performance (BER vs. SNR). The simulation procedure under timing errors induced by PVT variations is discussed next.

2.2 Simulation Procedure under Timing Errors

Throughout this work, timing errors will be introduced at the level of the ACSUs in the VD. RTL-level simulation is used to evaluate the performance of the VD under timing errors. IBM 130-nm CMOS process is used. The simulation procedure is illustrated in Figure 2.2 and is explained next with errors induced by two sources: voltage and/or process variations.

2.2.1 Voltage overscaling

Under VOS, the voltage is reduced beyond a critical design voltage (1.2 V in our design) so that timing errors start to appear. To evaluate the effect of timing errors and the decoder performance at different supply voltages, first HSPICE is used to characterize the worst case delays of basic gates employed in the ACSU (1-bit adder, and-gate, or-gate, inverter, etc.) at different supply voltages in the 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 in all these simulations and is chosen to meet the timing constraints at 1.2 V, resulting in a throughput of 590 Mb/s. HSPICE is also used to estimate the power consumption in the conventional and the proposed ACSU under different supply voltages.

2.2.2 Process variations

Process variations are classified into within-die (WID) and die-to-die (D2D) variations [29]. 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. These variations cause a large distribution in the delay profile of the gates.

Delay distributions of various gates found in the ACSU were obtained via Monte Carlo simulations in an IBM 130-nm CMOS process at the 3σ slow process corner with WID variations enabled. These delay distributions are sampled to obtain different instances of the ACSU. For example, Figure 2.3 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 of 590 Mb/s at the nominal 1.2 V and hence is kept fixed in all simulations.

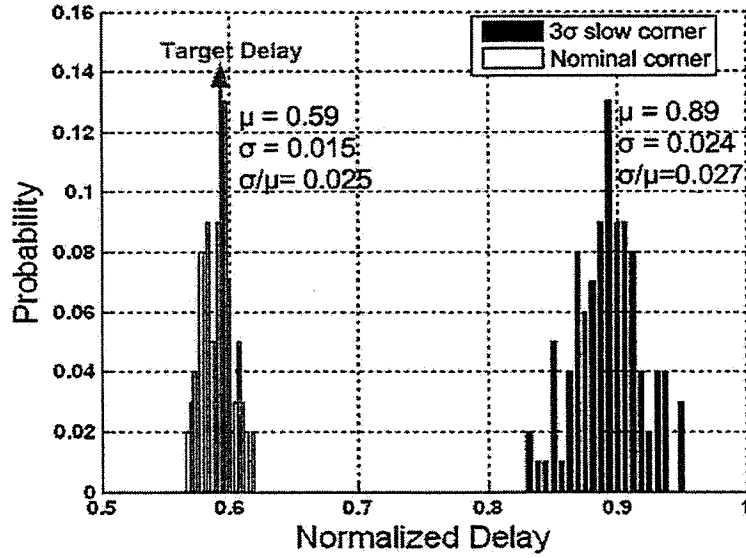


Figure 2.3 Delay distribution due to WID of 1.2-V ACSU at nominal and 3σ slow corner.

2.3 Impact of Timing Errors

Having explained the simulation procedure under VOS and process variations, Figure 2.4 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.

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

- Type 1 errors result in the computation and the selection of an incorrect PM. This event occurs when the MSBs in the BM adder 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 Type 1 error can be observed by the abrupt shift due to sign change in PMs in Figure 2.5, which shows the PM evolution for all states across multiple trellis stages.

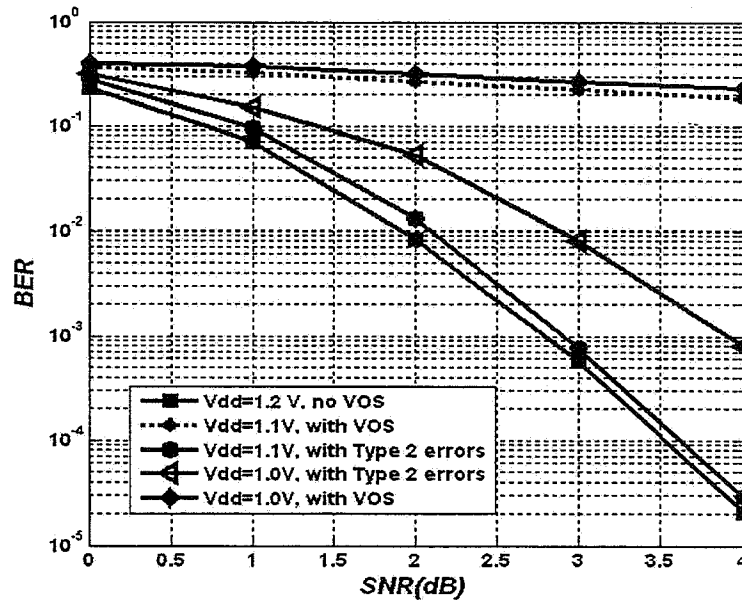


Figure 2.4 Decoding performance with ACSU subject to different sources of timing errors.

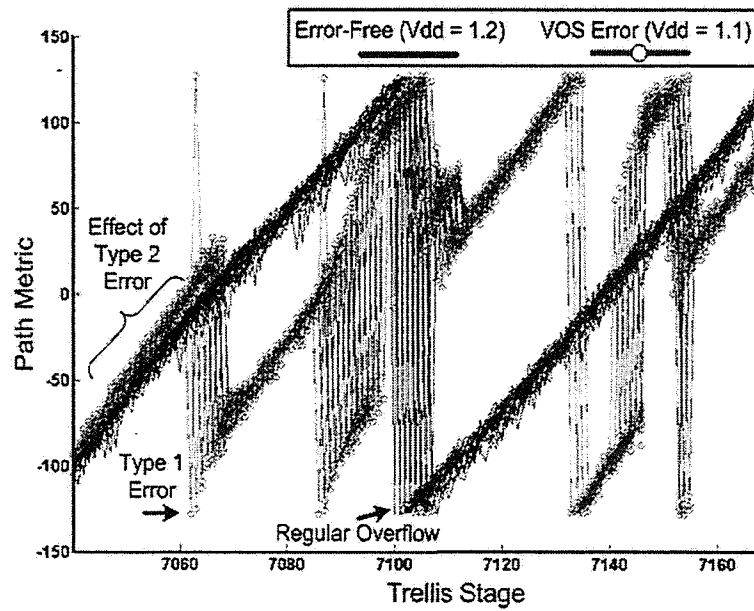


Figure 2.5 Path metric evolution under timing errors.

- Type 2 errors occur because of timing errors in the compare-select block so that the wrong (larger) PM gets selected. The wrong PM is close to the correct path metric when there are no errors in the BM adder, and it is a better estimate when there are adder errors (Type 1 errors). Thus, Type 2 errors are benign as compared to Type 1 errors, as seen in Figure 2.4 at $V_{dd} = 1.1$ V, where BER curves with Type 2 errors only are shown. Clearly, higher frequency Type 2 errors can also have a large impact on the BER as decision errors occur more regularly on chosen paths (see Figure 2.4 at $V_{dd} = 1.0$ V with Type 2 errors only). The effect of Type 2 errors can be observed by the slight increase in the PM's evolution rate in Figure 2.5 because sometimes a slightly larger PM is being selected.

The impact on BER due to timing errors induced by process variations in the ACSU is very severe as well, as can be seen in Figure 2.6. Therefore, there is a need to employ error resiliency for the ACSU. Because it is a recursive architecture, next we discuss issues and propose solutions for error resiliency in recursive systems in general and the ACSU in particular.

2.4 Algorithmic Noise-Tolerance in Recursive Architectures

Recursive architectures present two main challenges for the application of error-resiliency. First, recursive architectures suffer from *error propagation* where the estimation error at time instant n can impact future outputs (see Figure. 2.7(a)), whereas in feed-forward systems the estimation error affects only 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 at the output of the main block increases the critical path delay (see Figure. 2.7(a)), thereby gen-

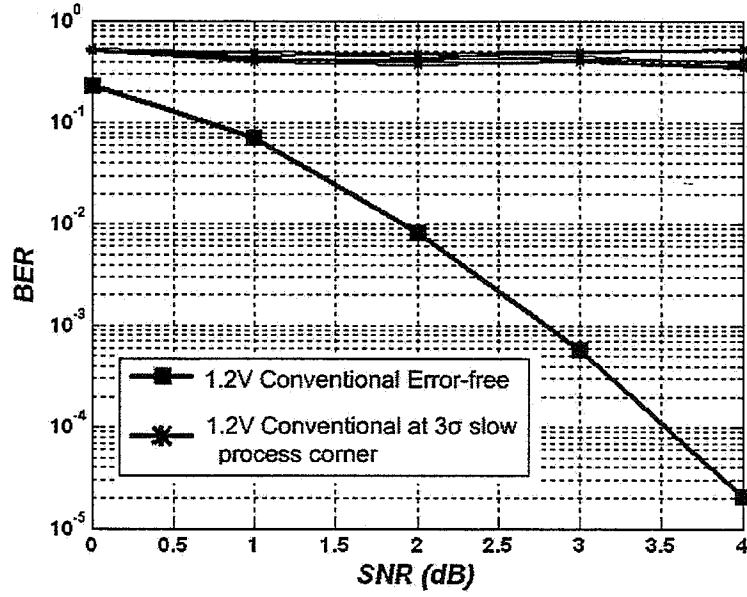


Figure 2.6 Decoding performance with ACSUs under WID variation at the 3σ slow corner.

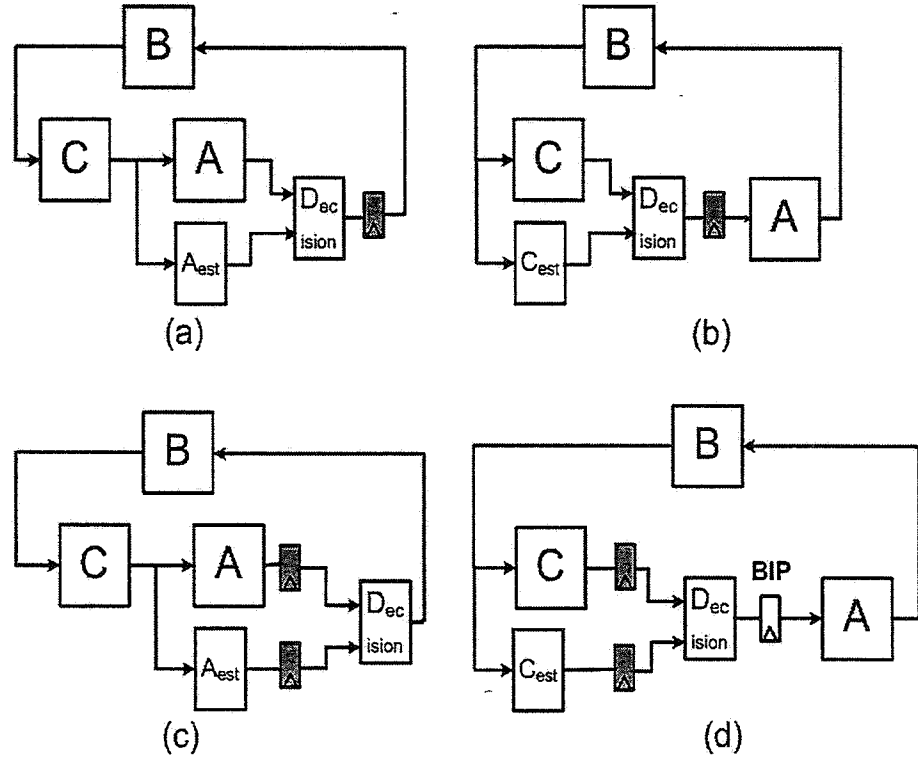


Figure 2.7 ANT for recursive architectures with shaded latches indicating the location of timing errors. (a) Timing errors impact hard-to-correct decision block, (b) retiming to ease error-correction, (c) retiming to prevent errors in decision block, and (d) introduction of additional latches using block interleaved pipelining (BIP).

erating timing violations in the decision block that are hard to correct because of its nonlinear nature. In feed-forward architectures, a pipelining register is introduced before the decision block as shown in Figure 1.11. This cannot be applied in recursive systems due to the feedback path at the output of the decision block. Next, we present different schemes to alleviate decision block errors in general recursive architectures and in ACSUs.

2.4.1 Algorithmic noise-tolerance in general recursive architectures

An aspect of the flexibility of recursive architecture is that one 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 2.7(b), where it may be 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 2.7(c). This scenario is acceptable because the estimator can be employed, as in the case of nonrecursive architectures, to generate the correct result. However, the error frequency and error magnitude at the output will be larger than a corresponding nonrecursive architecture due to the increased critical path delay.

Alternatively, look-ahead pipelining [10] can 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 additional hardware expense. Instead, for VD applications, one can employ a low-complexity pipelining technique referred to as *block interleaved pipelining* (BIP) [30].

BIP can be applied as long as the input can be processed in a block-based, block-independent manner. Assuming a recursive function f , the computations can be processed in parallel in two separate hardware units as shown in Figure 2.8(b) [30] if the processing of the two input blocks, X_1 and X_2 , is independent. Multiplexing, or

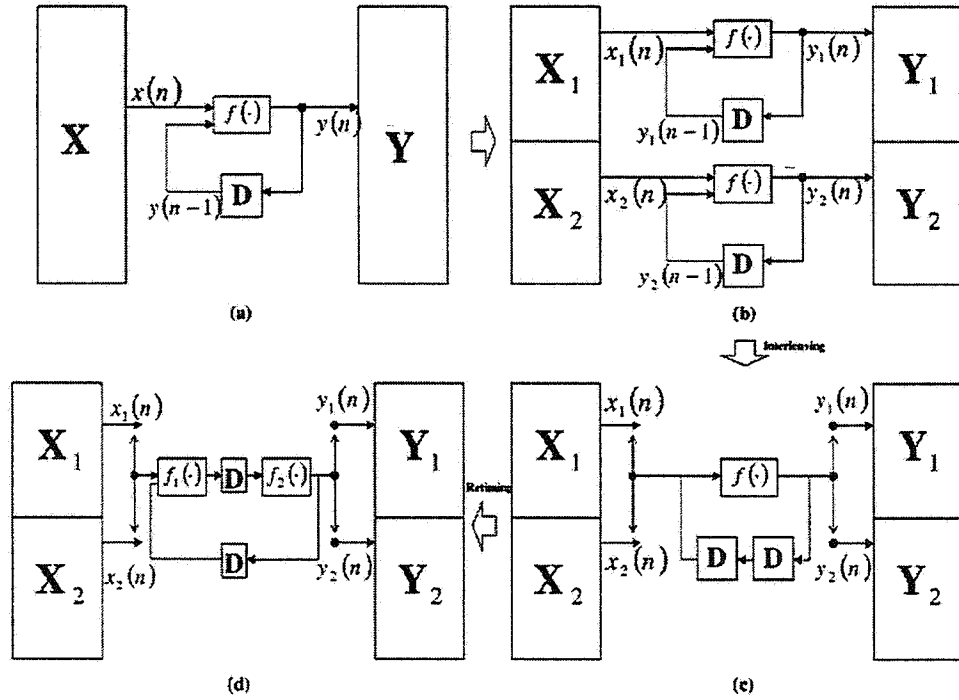


Figure 2.8 Pipelined BIP architecture. (a) Block-processing architecture, (b) parallel architecture, (c) block-interleaved architecture, and (d) pipelined block-interleaved architecture.

folding the two computations into a single hardware unit, will introduce an additional delay register in the feedback path (see Figure 2.8(c)). The additional register can be used to decrease the critical path delay of the hardware unit by retiming (see Figure 2.8(d)). In recursive ANT architectures, the retimed register will be used to provide enough time for the ANT decision block to finish computations at reduced supply voltages as shown in Figure 2.7(d), where BIP followed by retiming is applied. 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 shared state (zero) among the blocks.

2.4.2 Algorithmic noise-tolerance in ACSU

The ACSU is implemented using an LSB-first BM adder followed by an LSB-first comparator (subtractor) (see Figure 1.6). The two operations proceed in parallel so that the delay is dominated by a single 8-bit adder as illustrated in Figure 2.9(a). Errors in the ACSU can occur in the BM addition step or the compare-select step. A low-complexity estimator can be designed for the BM adder under ANT as it is a linear block. Determining an effective estimator for the comparator is hard as it is a nonlinear block. In addition to that, designing a low overhead ANT decision block to detect errors in the comparator is difficult because the output is a 1-bit or 2-bit signal. Therefore, in our proposed ANT schemes for ACSU we try to avoid timing errors in the comparator. One way to do that is to retime the feedback register across the comparator as illustrated in Figure 2.9(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 (decision-based errors) can be tolerated as explained in Section 2.3, we propose in the next chapter to use a relaxed comparator 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 (see Figure 2.9(c)). In addition, we introduce a fast ACSU architecture (FACSU) using the relaxed comparator where error-free operation is feasible at lower supply voltages due to reduced critical path delay (see Figure 2.9(d)). In this architecture, the BM adder and comparator steps are implemented in a parallel manner. The FACSU presents by itself a low power solution for VD applications and will also be used as an estimator for some of the ANT-based schemes proposed in the next chapter.

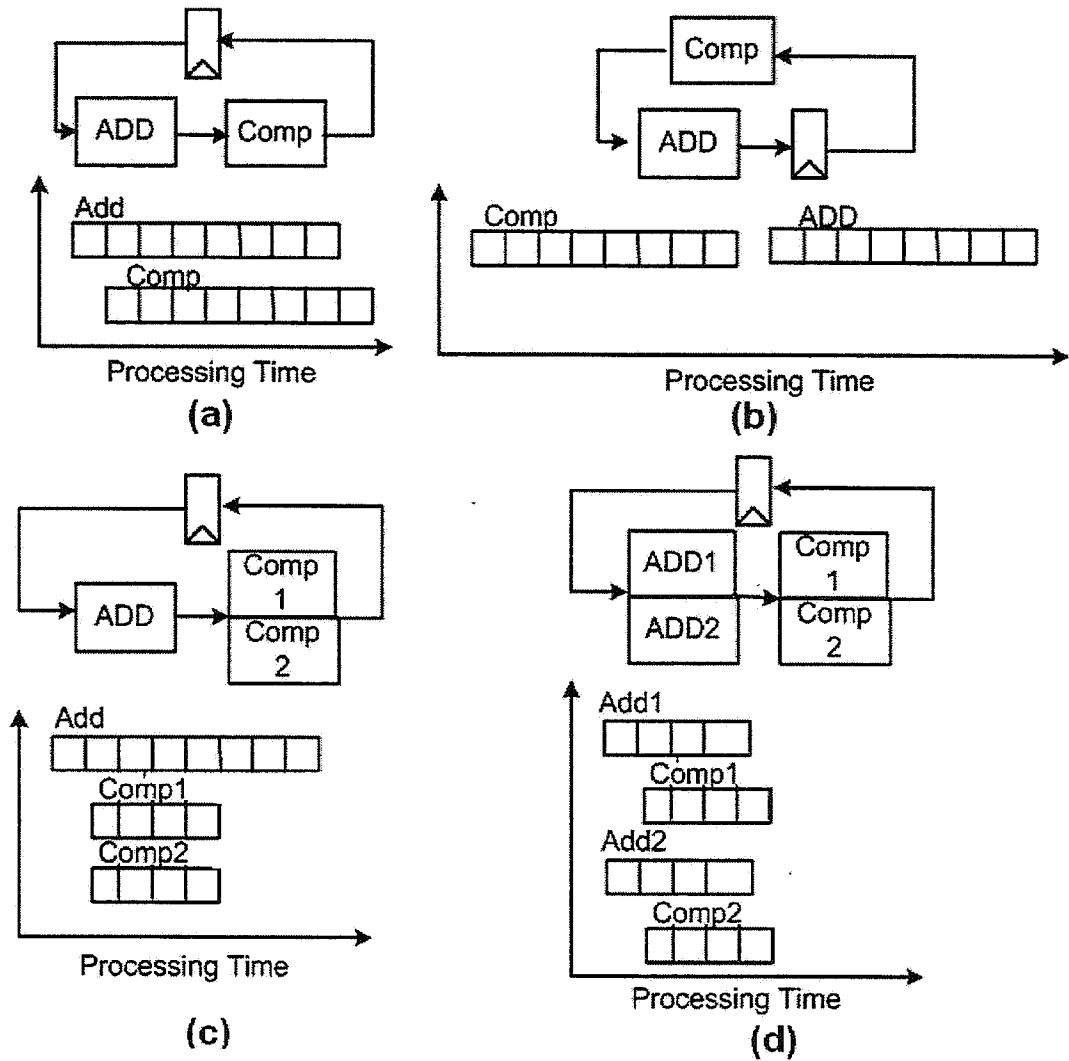


Figure 2.9 Throughput and latency of (a) conventional ACSU, (b) retimed ACSU, (c) relaxed ACSU, and (d) fast ACSU (FACSU).

2.5 Analysis of Timing Errors Effect on BER

In this section we analyze the effect of timing errors on the BER performance of the VD with ANT. First, we assume that there are no timing errors and derive a bound on BER based on derivation similar to [31]. Then we assume that ANT is applied to the ACSU of the VD and evaluate the effect of timing and estimation error on the derived bounds.

2.5.1 BER bound in absence of timing errors

By linearity of the convolution code, we assume that the all-zero sequence is sent and find the probability that another sequence is decoded. The probability of error P_b for CC is derived using the first event error probability p_e , which is the probability that another path ($P_{th_i} | i \neq 0$) is selected for the first time when compared to the all-zero path (P_{th_0}) given that the all-zero sequence (S_0) is sent. Defining the event E_0 as the event when S_0 is sent and summing the BMs over the considered paths, p_e is given by

$$p_e = p \left(\sum_{j \in P_{th_i} | i \neq 0} BM_j \leq \sum_{k \in P_{th_0}} BM_k \middle| E_0 \right) \quad (2.2)$$

In this work we consider BPSK modulation with alphabet $(+A, -A)$, AWGN noise $\mathcal{N}(0, \sigma^2)$, and rate 1/2 CC; from Section 1.2.1 the BM can be written as

$$BM_j = \|r_j - c_j\|^2 = \sum_{i=0}^1 (r_j^i - c_j^i)^2 \quad (2.3)$$

Factoring out the constant terms and dividing by $2A$,

$$BM_j \propto \sum_{i=0}^1 -\frac{r_j^i c_j^i}{A} = \sum_{i=0}^1 -r_j^i \text{sign}(c_j^i) \quad (2.4)$$

Given event E_0 , then r_j^0 and r_j^1 are $\mathcal{N}(-A, \sigma^2)$. Assuming paths P_{th_0} and P_{th_i} differ in d coded bits and using (2.4), then (2.2) is expressed as

$$p_{e_d} = \left(\sum_{\substack{t=1 \\ i \in \{0,1\}}}^d -2r_t^i \leq 0 \middle| E_0 \right) = Q \left(\frac{2dA}{\sqrt{4d\sigma^2}} \right) \quad (2.5)$$

where the Q -function is defined as

$$Q(\alpha) = \frac{1}{\sqrt{2\pi}} \int_{\alpha}^{\infty} e^{-\frac{x^2}{2}} dx \quad (2.6)$$

Let a_d be the number of paths that merge with P_{th_0} and differ from it in d coded bits; then the overall error probability P_e is upper-bounded as follows:

$$P_e \leq \sum_{d=d_{free}}^{\infty} a_d p_{e_d} \quad (2.7)$$

The above error probability indicates the probability that a path different from the zero path is selected, and d_{free} is called the minimum free distance of the CC and refers to the minimum distance between the all-zero path and any other path in the trellis. We are interested in the errors in the information bits which are decoded when the wrong path is selected. Define f_d to be the number of information bits that differ between the zero path and the other path. Then the average bit error probability P_b is bounded as follows:

$$P_b \leq \sum_{d=d_{free}}^{\infty} f_d a_d p_{e_d} = \sum_{d=d_{free}}^{\infty} f_d a_d Q \left(\frac{2dA}{\sqrt{4d\sigma^2}} \right) = \sum_{d=d_{free}}^{\infty} f_d a_d Q \left(\sqrt{\gamma d} \right) \quad (2.8)$$

where γ is the signal-to-noise ratio.

For CC, d , a_d , and f_d can be obtained from the state transfer equation which relates the state variables based on the properties of the CC and the trellis connection as discussed in [31].

2.5.2 BER bound under algorithmic noise tolerance

The above derivation is based on the fact that the hardware is error-free. With timing errors introduced and estimation errors due to ANT, the BM is shifted to a smaller value, making the path on which timing errors occur more probable. Therefore, we can assume the received codeword bit corresponding to a zero ($r^{(0)}$), which is $\mathcal{N}(-A, \sigma^2)$, is shifted by αA when a timing error occurs and the received codeword bit corresponding to a one ($r^{(1)}$), which is $\mathcal{N}(A, \sigma^2)$, is shifted by $-\beta A$ when a timing error occurs where α and β are greater than zero. We further assume that α is greater than β under timing errors when we derive the first event error probability with the assumption that the zero path is sent. In other words, we assume that the timing errors affect the zeros codeword bits more than the ones codeword bits, making the path different from the zero path more probable.

Assume the following:

- The length of path $P_{th_i}|i \neq 0$ (or number of stages) before it merges with the zero path P_{th_0} is $L/2$ so that the number of codeword bits on this path is L since a rate $1/2$ CC is considered.
- The two paths $P_{th_i}|i \neq 0$ and P_{th_0} differ in d symbols.
- k_0 timing errors occur on P_{th_0} with probability p_{k_0} .
- k_i timing errors occur on P_{th_i} with probability p_{k_i} .
- Timing errors occur uniformly in all L bits.

We expect k_0 errors in the zeros codeword bits on P_{th_0} , $\frac{L-d}{L}k_i$ errors in the ones codeword bits on P_{th_i} , and $\frac{d}{L}k_i$ errors in the zeros codeword bits on P_{th_i} . Thus, we can express the first event error probability in (2.2) with timing errors under ANT as follows:

$$p_{e_d} = Q \left(\left(-k_0\alpha + \frac{d}{L}k_i\beta - \frac{L-d}{L}k_i\alpha + 2d \right) \frac{A}{\sqrt{4d\sigma^2}} \right) p_{k_0}p_{k_i} \quad (2.9)$$

The path length L can be obtained from the CC state transfer function. To find p_{k_0} and p_{k_i} , we assume that the timing error is a binomial variable with occurrence probability of p_{vos} that depends on the supply voltage used in the ANT system and thus:

$$p_{k_i} = \binom{L_d}{k_i} (1 - p_{vos})^{L_d - k_i} p_{vos}^{k_i} \quad (2.10)$$

Following the derivation of (2.8) but using (2.9), we obtain the following average bit error rate probability under timing errors:

$$P_b = \sum_{d=d_{free}}^{\infty} a_d f_d \sum_{k_0=0}^L \sum_{k_1=0}^L Q \left(\left\{ -k_0\alpha + \frac{d}{L}k_i\beta - \frac{L-d}{L}k_i\alpha + 2d \right\} \sqrt{\frac{\gamma}{4d}} \right) p_{k_0} p_{k_1} \quad (2.11)$$

In absence of timing errors, we have $p_{vos} = 0$ resulting in $k_0 = 0$ and $k_i = 0$, and (2.11) will reduce to (2.8). The state transfer function for the $K = 8$, rate=1/2, $g^0 = 247$, and $g^1 = 371$ CC in this work was obtained following the procedure outlined in [31] to characterize d , a_d , f_d , and L_d . A closed form solution was not found when L_d is included, so we assume $L_d = \lceil 1.5d \rceil$. RTL simulation with individual gate delay characterization from HSPICE is used to estimate p_{vos} at different supply voltages shown in Table 2.1. With ANT estimation error we assume that the residual error is approximately equal to the BM value. Thus, we choose $\alpha = 1$ and $\beta = 0.9$. Figure 2.10 shows the bounds on BER expressed in 2.11. With supply voltage reduced up to 0.9 V, there is around 0.5 dB to 1.8 dB loss in coding gain and this loss increases dramatically as supply voltage is reduced further.

Table 2.1 Probability of timing error (p_{vos}) at different supply voltage

Supply Voltage (V_{dd})	p_{vos}
1.2V	0
1.1V	0.005
1.0V	0.024
0.9V	0.077
0.8V	0.32
0.7V	0.62

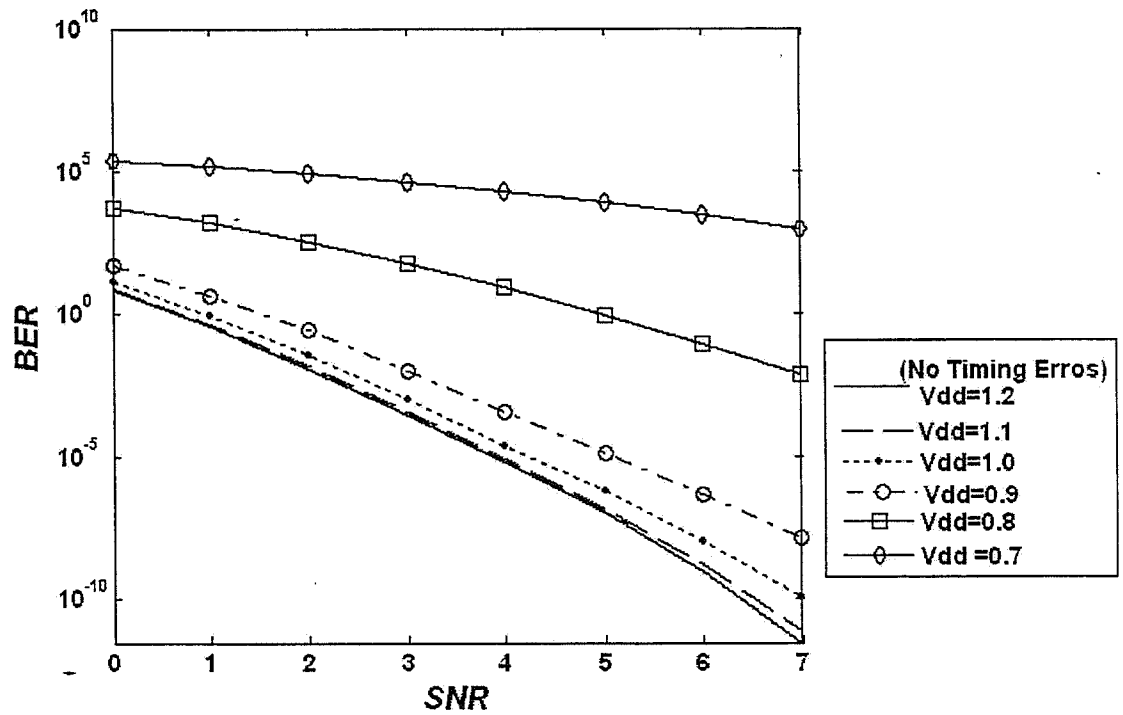


Figure 2.10 BER bounds with ANT-VD under timing errors.

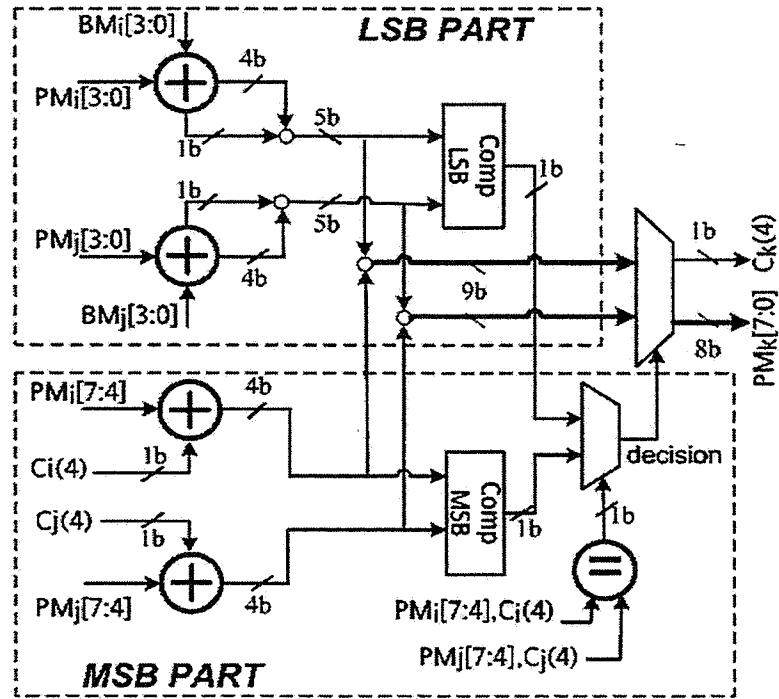
CHAPTER 3

LOW-POWER ERROR-RESILIENT ACSU ARCHITECTURES

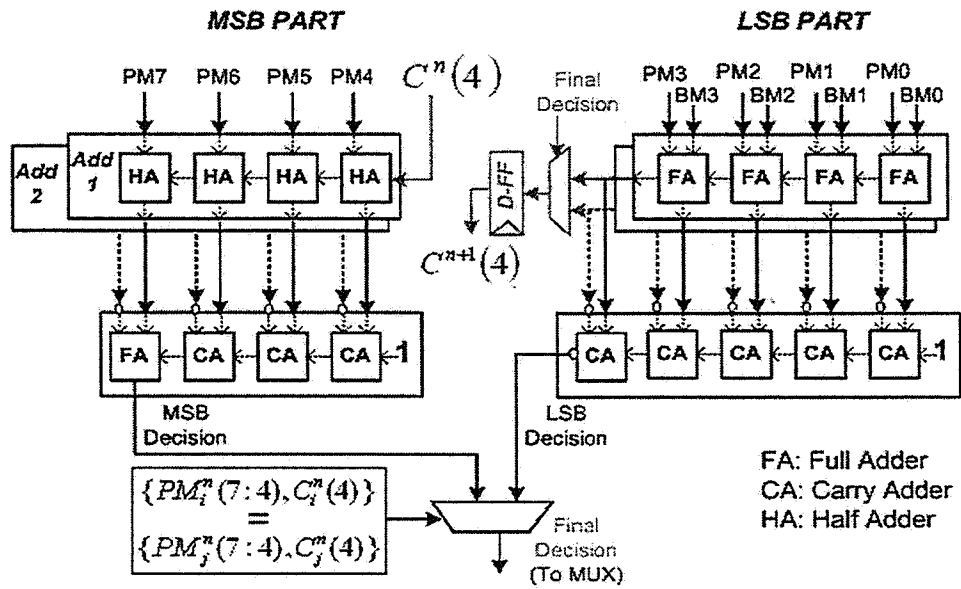
Having analyzed the architectural issues in designing error-resiliency for general recursive systems and particularly ACSUs, we present three novel ACSU architectures for low power operation and/or PVT variations. The first is a fast architecture with a relaxed comparison and is referred to as FACSU. Power savings can be achieved by operating at lower supply voltages due to smaller critical path delay. Although the FACSU is a low power solution, it cannot handle timing errors. In the second and third, two ANT schemes are proposed to allow the ACSU to handle timing errors and still save power. Simulations showing BER performance and the achieved power savings for all proposed architectures are also presented.

3.1 Fast ACSU Architecture

The FACSU employs the concepts of *delayed carry* and *relaxed comparator* to decrease the delay of the BM addition and comparison stage, respectively, resulting in an ACSU with a reduced delay. The architecture of the FACSU is shown in Figure 3.1. Here, the computation is partitioned into an LSB and an MSB section with the delay of each section being approximately half the original ACSU delay. Recall that the BMs and the PMs are quantized to 4-b and 8-b, respectively. During BM addition stage, 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-bit value with the additional bit representing the delayed carry. This reduces



(a)



(b)

Figure 3.1 FACSU architecture with delayed carry and relaxed comparator. (a) General architecture, (b) detailed architecture (PM selection Mux not included).

Table 3.1 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

the BM adder delay by a factor of 2. In the comparison stage, FACSU uses a relaxed comparator to split the computations into two parts, similar to the BM addition stage. The relaxed comparator is explained next.

3.1.1 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 BM 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 parts 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 addition of BM to PM is always correct and errors only occur in the comparator because we are splitting it into MSB and LSB parts and ignoring the propagation of carry between the two due to BM addition, 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 make them equal in the next clock cycle, as illustrated in cases A and B of Table 3.1. Under this setting, the relaxed comparator decision is based on

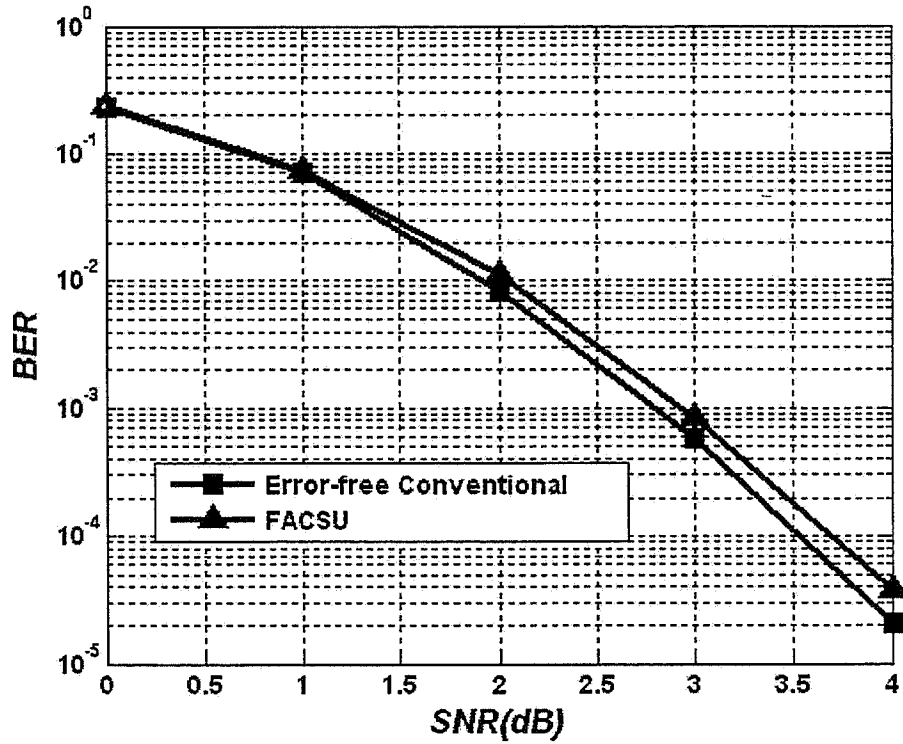


Figure 3.2 Decoding performance of FACSU. Errors are only caused by the relaxed comparison.

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 after carry propagation 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. BER simulation in Figure 3.2 shows only a 0.15-dB loss in coding gain due to the relaxed comparator.

3.2 Redundancy-ANT ACSU Architecture

Timing violations induced in the comparator stage of the ACSU are hard to correct for, as discussed in Section 2.4.2, due to the nonlinearity of the computation under

consideration. We use the relaxed comparator to avoid timing errors due to voltage or process variations in the comparison stage at the expense of limited-decision errors. We use BIP to provide enough time for the ANT correction block to finish computation.

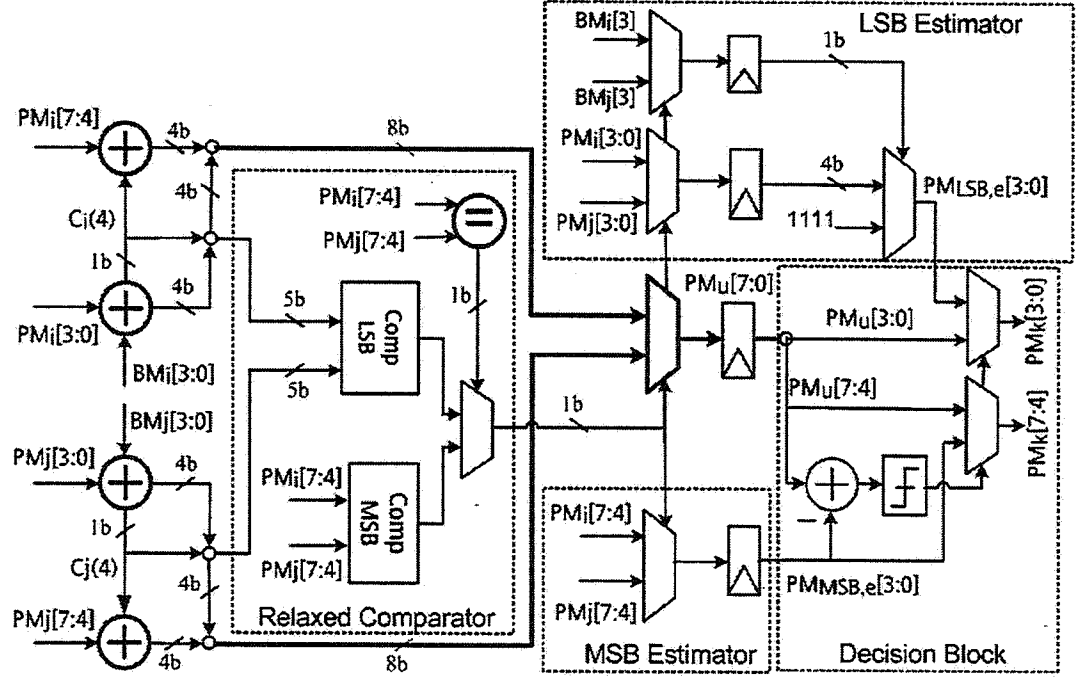


Figure 3.3 RA-ACSU: BIP and register retiming are used to prevent timing errors in the correction block.

The architecture of the redundancy-ANT ACSU is shown in Figure 3.3 and is referred to as RA-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 RA-ACSU incorporates an MSB and an LSB estimator of the updated PM ($PM_u[7:0]$). The MSB 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 LSB estimator is incorporated to decrease the estimation error, which is necessary to alleviate error propagation due to ACSU feedback path. The LSB 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 LSB 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 LSB of the input PM corresponding to the chosen BM, i.e., either $PM_i[3 : 0]$ or $PM_j[3 : 0]$.

The decision block computes the difference between the MSBs of the updated PM ($PM_u[7 : 4]$) and its estimates ($PM_{MSB,e}[3 : 0]$). 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; otherwise, the output PM is set to the updated PM. Table 3.2 indicates that there is 68% increase in gate complexity in the RA-ACSU when compared to the conventional ACSU. In spite of this overhead, the simulation section will show that power savings can still be achieved since the whole circuit is operated at reduced supply voltage.

Table 3.2 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

3.3 State-Clustered ANT ACSU Architecture

In RA-ACSU, computational redundancy within the ACSU is induced to provide an estimate of the output. In the new architecture, state-clustered ACSU, we exploit the spatial computational redundancy among the different ACSUs in the trellis. We benefit from the structure of the trellis to provide an estimate of the erroneous PM. 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 the ACSUs

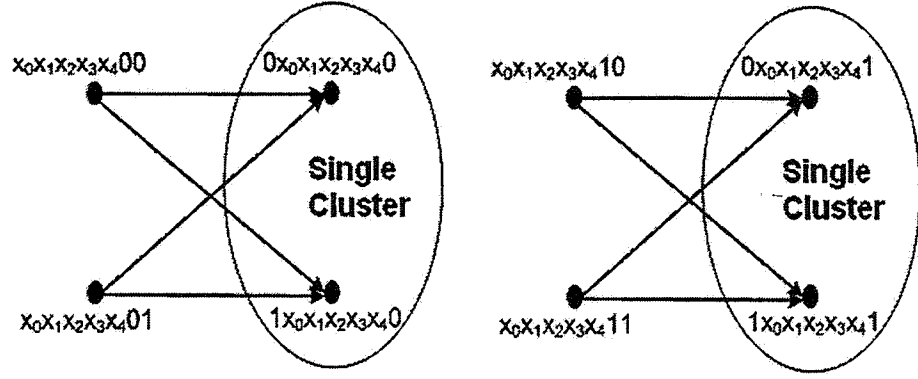


Figure 3.4 State clustering for 128-state rate-1/2 code with 2-state cluster where states share the same set of previous states.

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 in each cluster to correct for the wrong PMs. For example, in a 128-states rate-1/2 trellis, 7 bits are needed to represent each state and the trellis is partitioned into 64 disjoint 2-state clusters of the form shown in Figure 3.4. In all ACSUs, we still use a relaxed comparison to avoid timing based errors in the comparators.

In order to increase the number of ACSUs per cluster and consequently decrease the number of estimators needed, we can use radix- k processing where each state will have k transitions resulting in a k -state cluster. For example, the 128-state rate-1/2 radix-4 trellis can be divided into 32 disjoint 4-state clusters of the form shown in Figure 3.5.

3.3.1 Estimation schemes

Two estimation schemes are presented to correct for BM addition errors under state clustering. In the first, the FACSU is used for one of the states in the group to provide an estimate for the erroneous ACSUs in the same group since the FACSU can operate correctly under reduced voltage. This scheme is referred to as SC-FACSU. In the second, a redundant state is added to each cluster and is referred to as SCS-

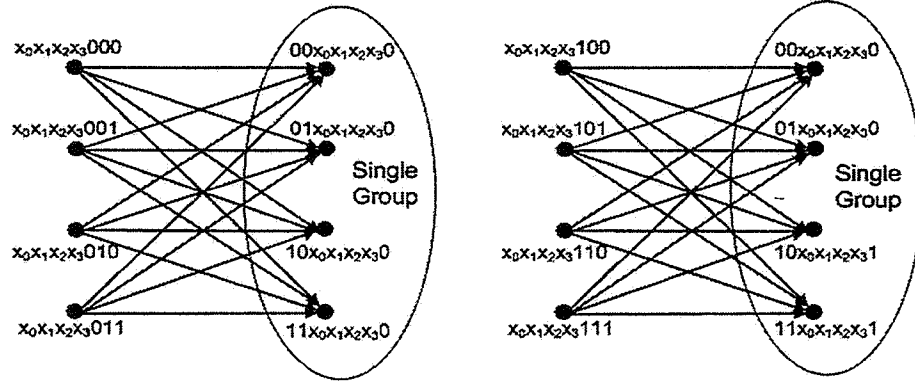


Figure 3.5 State clustering for 128-state rate-1/2 code with 4-state cluster where states share the same set of previous states.

FACSU. A FACSU is used for this state to operate error-free at low supply voltages. The additional state shares the same set of previous states as the rest of the states in the cluster (see Figure 3.6). The codeword (or BM) for the branch joining state X to the additional state is chosen to be equidistant from the codewords on the branches joining state X to the rest of the states in the cluster. For example, in the first cluster in Figure 3.6, the codewords chosen are 01 and 10, which are equidistant from 00 and 11. Therefore, the estimated PM will be closer to the correct PM than the previous scheme (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. In all ACSUs in the two schemes, a relaxed comparator is used to avoid timing-based decision errors.

3.3.2 State clustered architecture design

The general architectures for SC-FACSU and SCS-FACSU based on radix 2 (two states X and Y per cluster) are presented in Figure 3.7 (a) and (b). In SC-FACSU, state X uses FACSU architecture and is error free. State Y uses regular ACSU with a relaxed comparison. The ANT decision block corrects the output of state Y

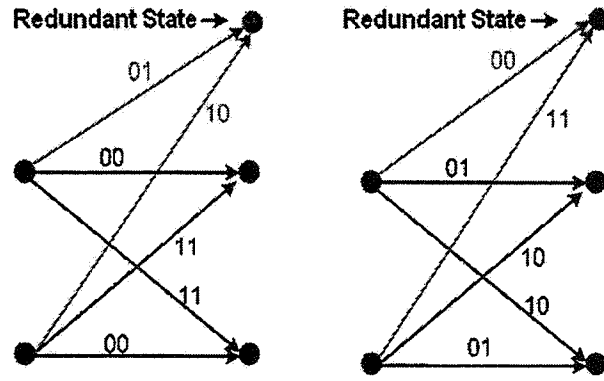


Figure 3.6 State clustering with the introduction of a redundant stage.

Table 3.3 Complexity estimate of state clustering architectures (two states per cluster)

Modules	2-input AND	2-input OR	Inverter	Transistors
Conventional	326	184	182	2404
SC-ACSU	395	226	219	2922
SCS-FACSU	618	353	343	4570

by using the estimate provided by state X. It computes the difference between the MSBs of the PM of state Y ($PM_{ye}[7 : 4]$) and the provided estimate from state 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 output PM of state Y (PM_y) is set to the estimated PM (PM_X); otherwise, it is set to the output PM of state Y (PM_{ye}). In SCS-FACSU, the estimate is provided by the redundant state. Similar architectures apply for higher radix clustering where we have more ACSUs per cluster. SC-FACSU and SCS-FACSU exhibit, in Table 3.3, 22% and 90% respective increase in gate complexity over conventional architecture with 2-state clusters. This overhead in complexity will decrease as the number of states per cluster is increased since a single estimate is used for larger number of states.

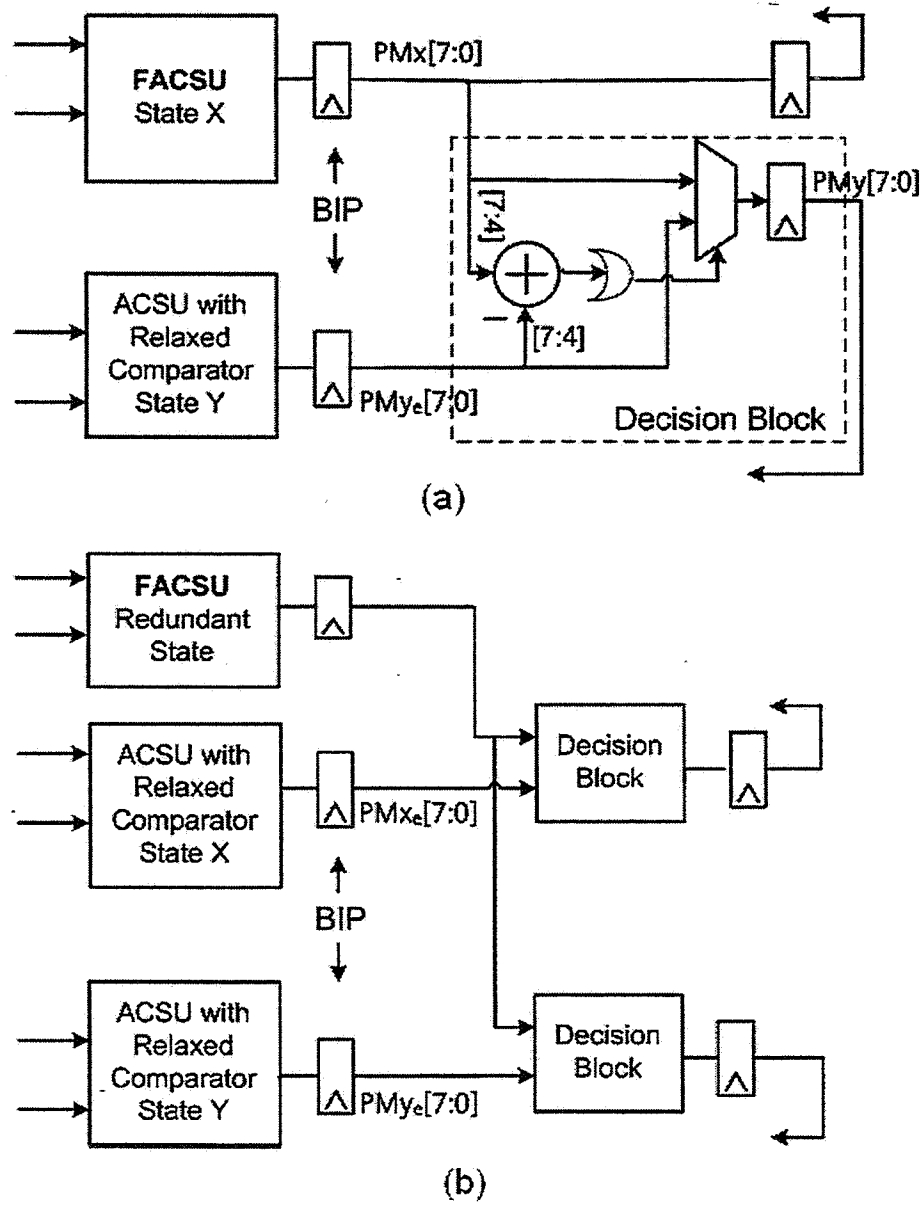


Figure 3.7 State clustering architecture with two states X and Y. (a) SC-FACSU architecture, (b) SCS-FACSU architecture.

3.4 Simulations and Results

This section describes the BER performance and power savings achieved by the different architectures proposed - FACSU, RA-ACSU, SC-FACSU, and SCS-FACSU - under voltage overscaling and process variations, and compares it with conventional ACSU. The simulation procedure has already been discussed in Section 2.2.

3.4.1 Voltage overscaling

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 delay of FACSU at different voltages using HSPICE is shown in Figure 3.8. The only source of errors in this case is decision errors introduced by the relaxed comparator. As indicated in Section 3.1, FACSU 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 BER performance is obtained for RA-ACSU at a nominal supply voltage of 1.2 V since only errors under this setting are due to the relaxed comparator and this is shown in Figure 3.9. As the supply voltage is reduced from 1.2 V to 1.1 V, the conventional ACSU BER increases dramatically. The RA-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 RA-ACSU (see the BER curve for $V_{dd} = 0.8$ V) due to increased frequency of VOS errors as well as errors in the estimator and decision blocks. Note that a 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.

Figure 3.10 shows the BER curves for state clustering obtained by reducing the supply voltage from 1.2 V to 0.8 V with two states per cluster. At 0.9 V, SC-FACSU

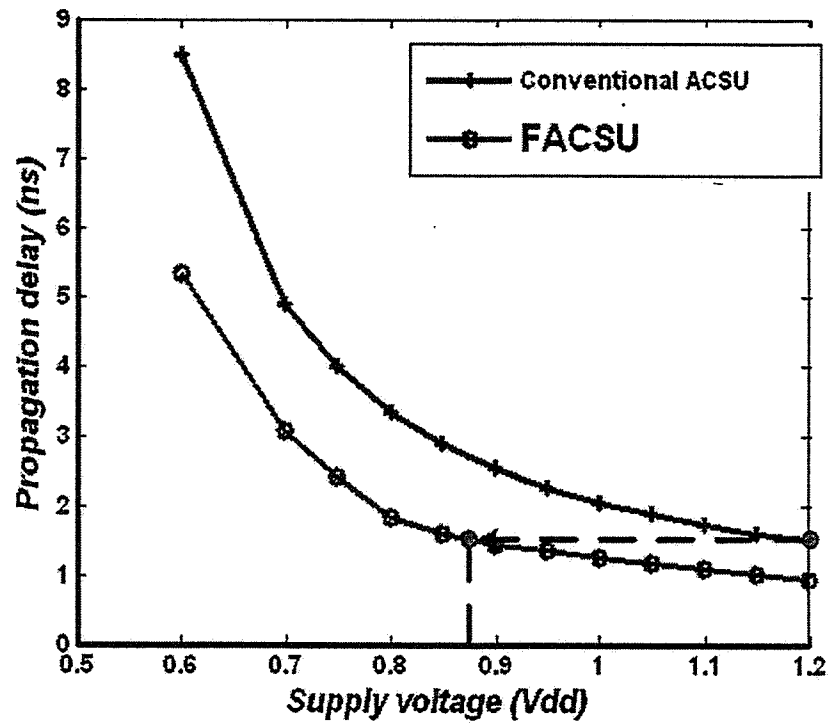


Figure 3.8 Delay of conventional ACSU and FACSU at different supply voltages.

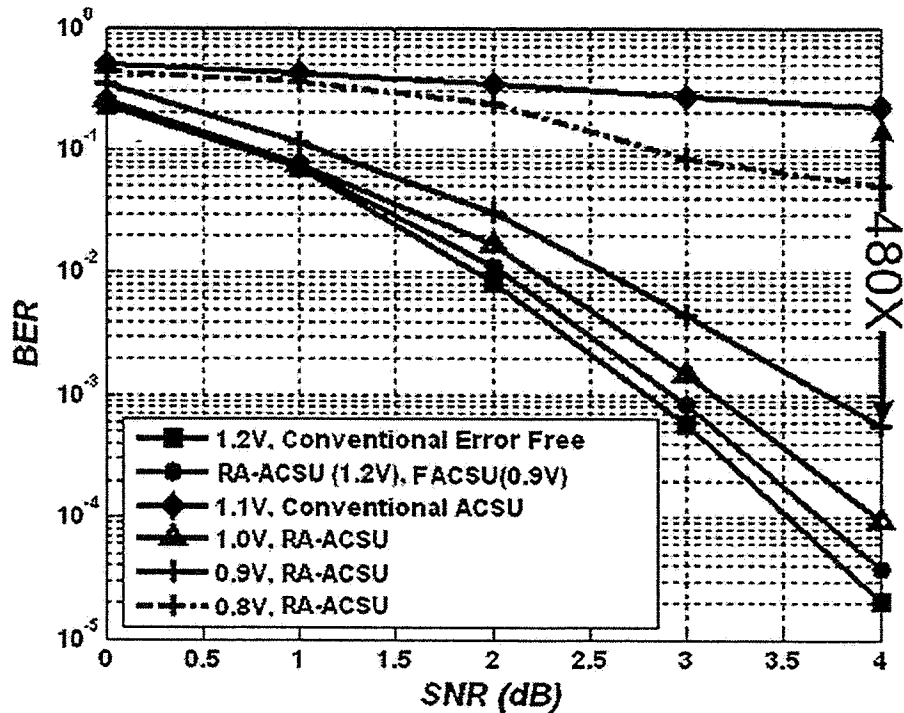


Figure 3.9 BER of conventional ACSU, FACSU, and RA-ACSU at different supply voltages.

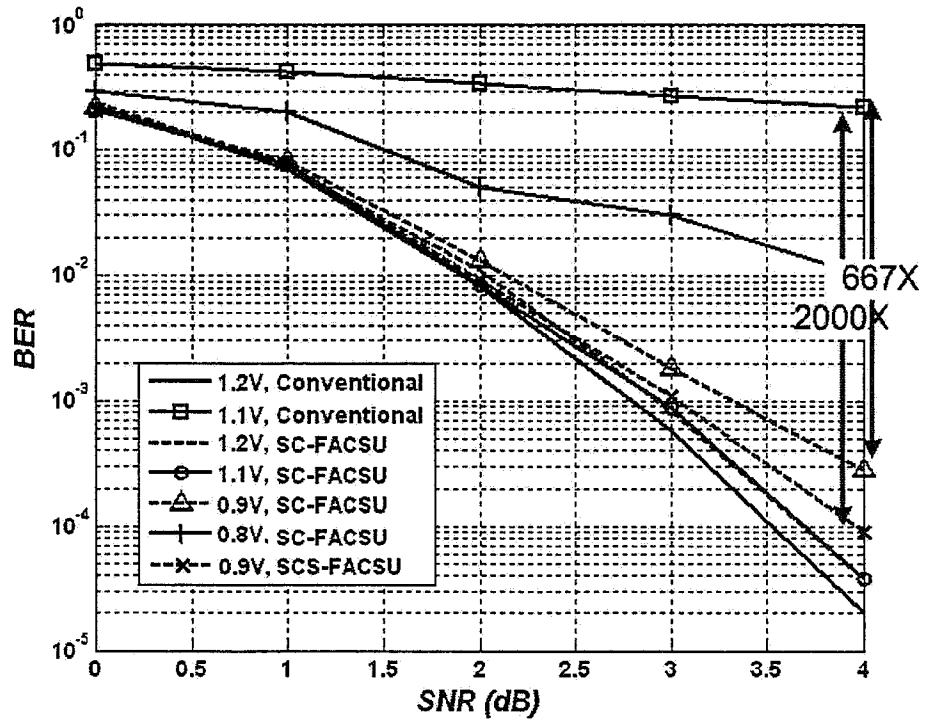


Figure 3.10 BER of a conventional ACSU, SC-FACSU, and SCS-FACSU at different supply voltages using radix-2 computation (2-state cluster).

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 approximate 667X and 2000X BER improvement over conventional ACSU. At 1.2 V SC-FACSU shows only a loss of 0.15 dB due to relaxed comparison errors. Reduction of voltage beyond 0.9 V leads to a noticeably degraded performance (see the BER curve for $V_{dd} = 0.8$ V) due to increased frequency of VOS errors. Similarly, Figure 3.11 shows the BER curves with 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 $V_{dd} = 0.9$ V with 956X and 2442X BER improvement.

3.4.2 Process variations

Figures 3.12 and 3.13 show BER distribution due to process variations at 2-dB signal-to-noise ratio (SNR) for conventional ACSU, FACSU, RA-ACSU, and SC-FACSU with two states per group. Conventional ACSU suffers from a large increase in BER

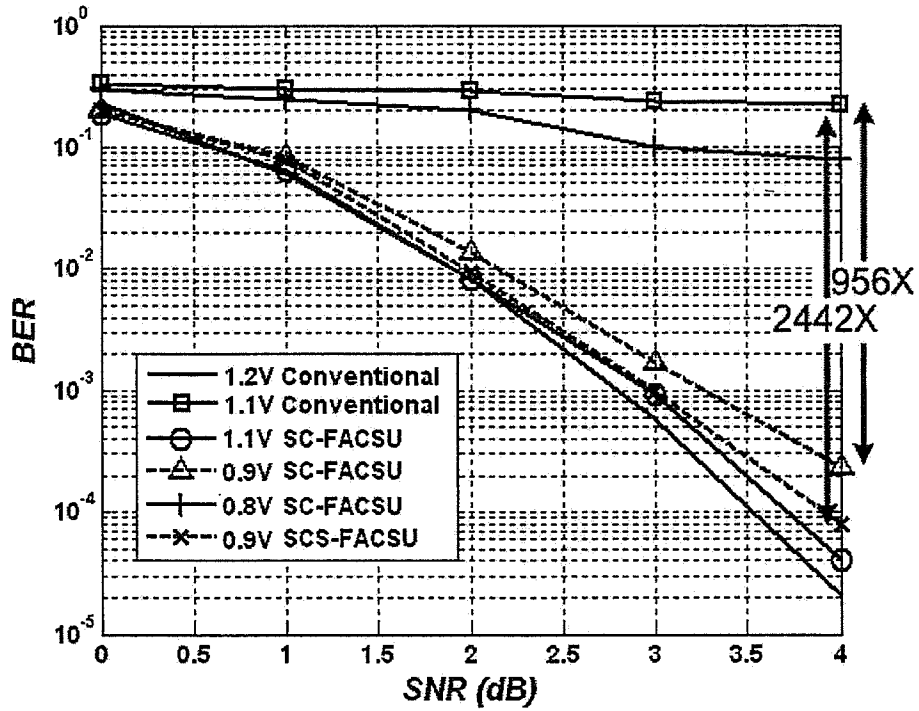


Figure 3.11 BER of a conventional ACSU, SC-FACSU, and SCS-FACSU at different supply voltages using radix-4 computation (4-state cluster).

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 clock frequency. Similarly, FACSU shows large degradation in performance due to timing errors caused by process variations. On the other hand, RA-ACSU and SC-FACSU can operate at the 3σ slow process corner with 16X and 33X average BER improvement if voltage is slightly increased from 1.2 V to 1.3 V. Performance under process variations at 2-dB SNR for SC-FACSU and SCS-FACSU with different radix processing is shown in Table 3.4.

Figure 3.14 shows the BER curves for various SNRs under process variations for conventional ACSU, RA-ACSU, and 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 in Figure 3.15, there is only a 0.6-dB loss in the average coding gain at a BER of 3×10^{-4} using SC-FACSU.

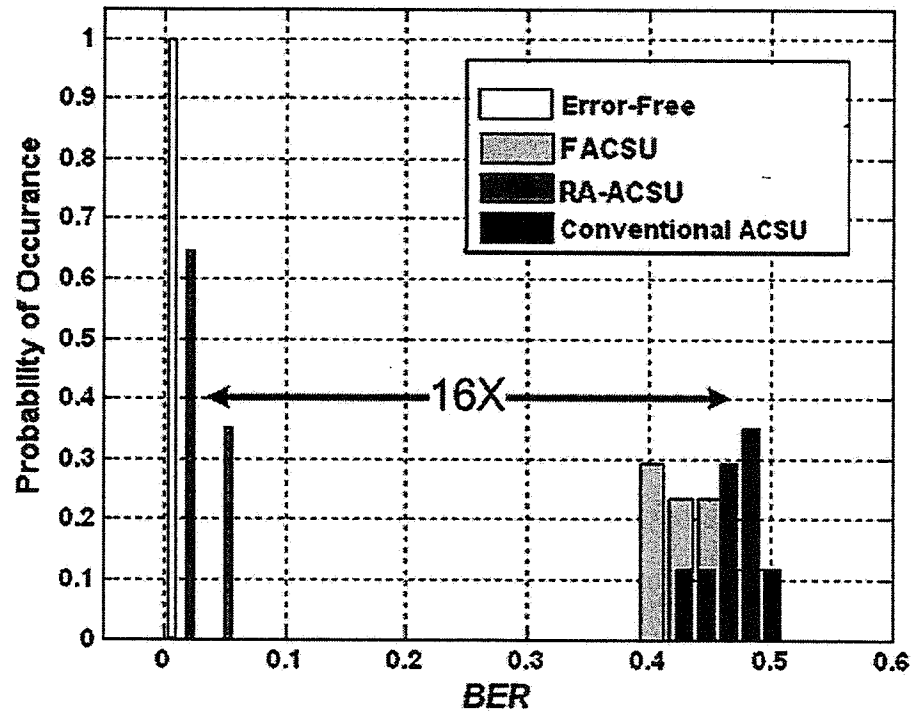


Figure 3.12 BER distribution at 3σ slow corner with WID variations and at SNR = 2 dB.

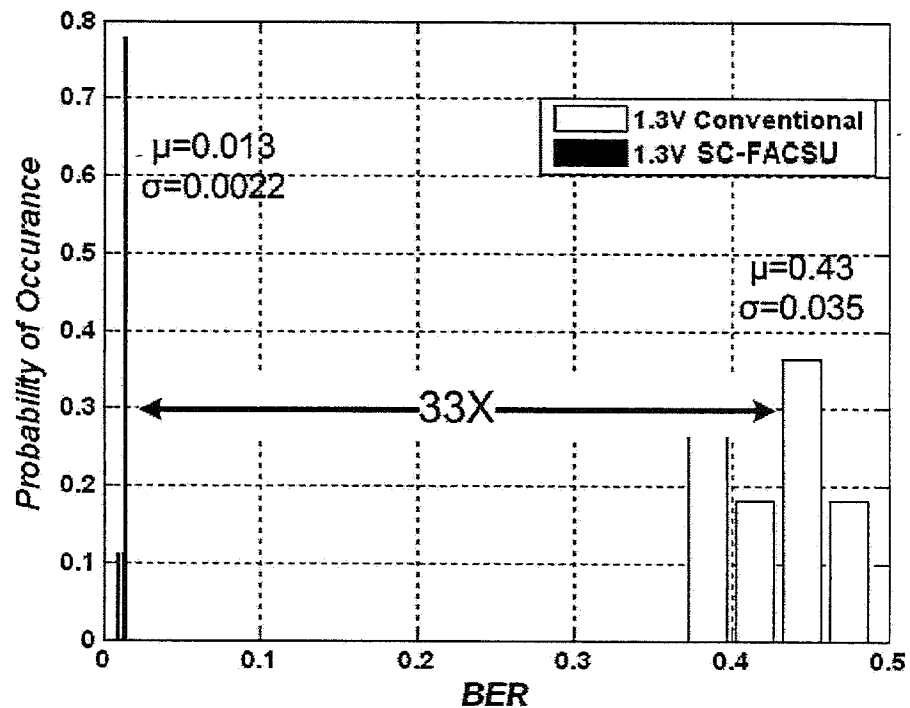


Figure 3.13 BER distribution with radix-2 computation at a 3σ slow corner with WID variations and SNR = 2 dB.

Table 3.4 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

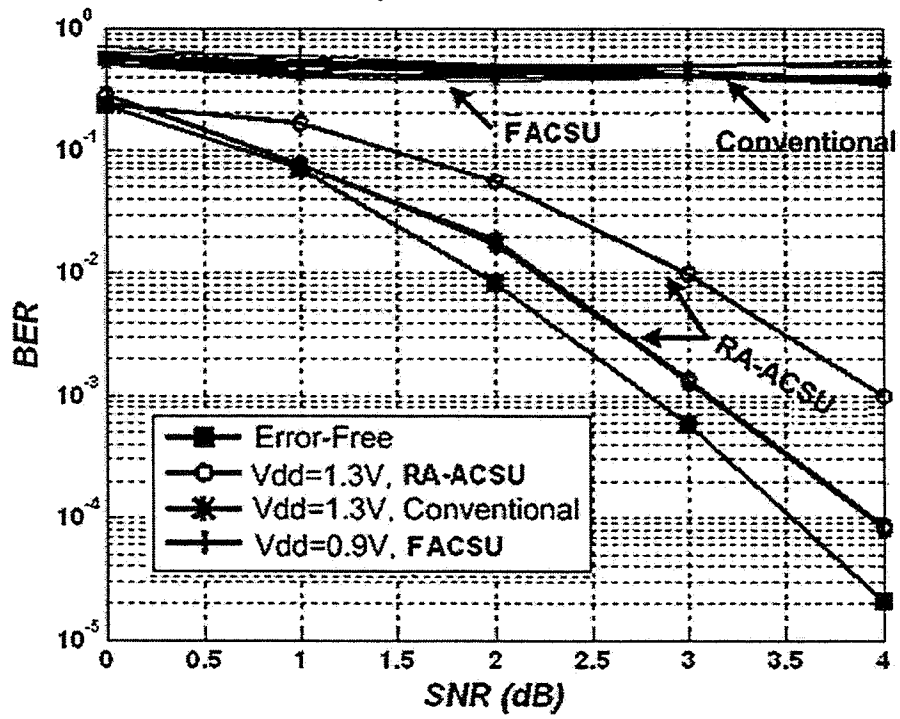


Figure 3.14 BER at 3σ slow corner with WID variations for RA-ACSU.

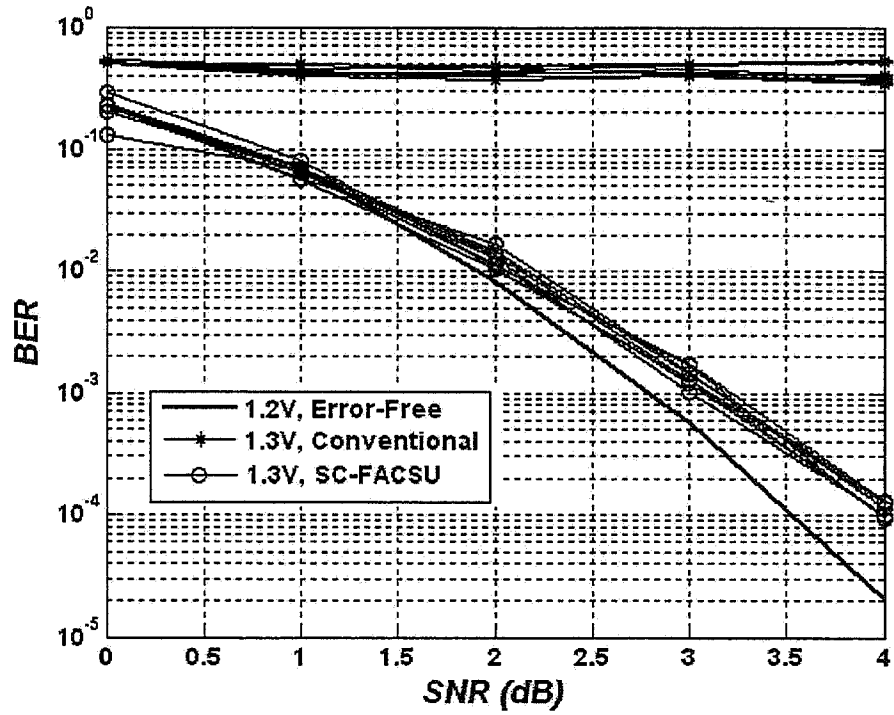


Figure 3.15 BER at 3σ slow corner with WID variations for state clustering.

3.4.3 Power savings

HSPICE simulations are carried out to estimate the ACSU power consumption in Figures 3.16 and 3.17. 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 savings. With 2-state cluster, SC-FACSU and SCS-FACSU achieve power savings of 71% and 42%, respectively, and a power savings of 56% and 41%, respectively, with 4 states/cluster.

At the 3σ slow process corner, ASV and ABB are applied to the conventional ACSU and FACSU to operate at the target data rate. Under this setting, the 1.3-V RA-ACSU exhibits 25% power savings over the conventional ACSU whereas the FACSU achieves 44%. 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% with 4-state cluster. Although FACSU shows a large power savings, it is not resilient to timing errors induced by VOS or process variations. 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.

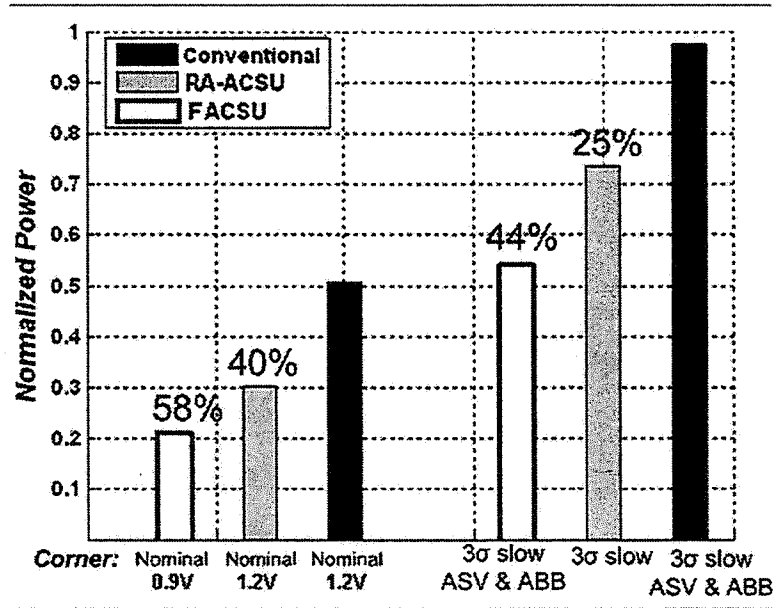


Figure 3.16 Power consumption per ACSU under VOS and process variations.

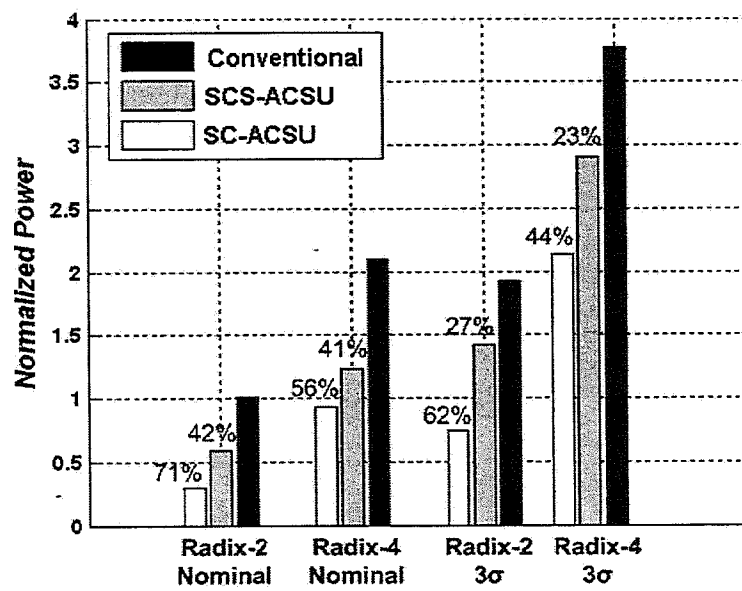


Figure 3.17 Average power consumption per ACSU under VOS and process variations.

CHAPTER 4

CONCLUSIONS

In this thesis, we introduced error resilient frameworks to decrease power consumption in add-compare-select units of a Viterbi decoder by operating at supply voltages beyond conventional ones. Error resiliency saves power by enabling nominal-case design instead of worst-case design methodology under process, voltage, and temperature variations. As technology scaling continues, the benefits from reduced energy and delay are mitigated by the increase of such variations, making error resiliency an elegant solution to continue reaping the benefits of scaling. Three low-power schemes for add-compare-select units have been proposed that present a trade-off between low power and reliability in terms of bit-error-rate performance of the decoder.

Future work can be directed toward the design of more efficient estimators for recursive architecture to combat error propagation. The survivor memory unit of Viterbi decoders is another place that consumes a significant portion of decoder power and where error resiliency can be applied. Error resiliency in other types of forward-error control decoders such as Turbo and LDPC decoders also presents a topic for future investigation as they are being widely employed in different applications with more stringent decoding requirements leading to increased complexity and power consumption.

REFERENCES

- [1] B. Bougard et al., "Energy-scalability enhancement of wireless local area network transceivers," in *IEEE Workshop on Signal Processing Advances in Wireless Communication*, July 2004, pp. 449–453.
- [2] K. Masselos, S. Blionas, and T. Rautio, "Reconfigurability requirements of wireless communication systems," in *IEEE Workshop on Heterogeneous Reconfigurable Systems on Chip*, Hamburg, Germany, April 2002.
- [3] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Design Automation Conference (DAC)*, 2003, pp. 338–342.
- [4] International Technology Roadmap for Semiconductors, "Executive summary," 2007. [Online]. Available: <http://www.itrs.net/reports.html>.
- [5] 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, Oct. 2003.
- [6] R. Hegde and N. R. Shanbhag, "Soft digital signal processing," *IEEE Trans. on VLSI*, vol. 9, no. 6, pp. 813–823, Dec. 2001.
- [7] J. Forney, "The Viterbi algorithm," *Proc. of the IEEE*, vol. 61, no. 3, pp. 268–278, March 1973.
- [8] G. Fettweis and H. Meyr, "High-speed parallel viterbi decoding: Algorithm and vlsi-architecture," *EEE Comm. Mag.*, pp. 46–55, May 1991.
- [9] A. Yeung and J. Rabaey, "A 210mb/s radix-4 bit-level pipelined viterbi decoder," in *IEEE International Solid-State Circuits Conference*, Feb. 1995, pp. 88–89.
- [10] K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York, NY: John Wiley & Sons, 1999.
- [11] G. Fettweis and H. Meyr, "High-rate viterbi processor: A systolic array solution," *IEEE J. Select. Areas Commun.*, vol. 8, pp. 46–55, Oct. 1990.

- [12] P. J. Black and T. H. Meng, "A 140-mb/s, 32-state, radix-4 viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 27, pp. 1877–1885, Dec. 1992.
- [13] —, "Hybrid survivor path architectures for viterbi decoders," *IEEE J. Solid-State Circuits*, vol. 27, pp. 1877–1885, Dec. 1992.
- [14] S. Kubota, S. Kato, and T. Ishitani, "Novel viterbi decoder implementation and its performance," *IEEE Trans. Comm.*, vol. COM-41, no. 8, pp. 1170–1178, 1993.
- [15] J. Anderson and E. Offer, "Reduced-state sequence detection with convolutional codes," *IEEE Trans. Information Theory*, vol. 40, no. 3, pp. 965–972, May 1994.
- [16] J. B. Anderson, "Limited search trellis decoding of convolutional codes," *IEEE Trans. Information Theory*, vol. 35, no. 5, pp. 944–955, Sep. 1989.
- [17] A. J. Simmons, "Breadth-first trellis decoding with adaptive effort," *IEEE Trans. Comm.*, vol. 38, no. 1, pp. 3–12, Jan. 1990.
- [18] M. Chan, W. Lee, M. Lin, and L. Chen, "IC design of an adaptive viterbi decoder," *IEEE Trans. Consum. Electron.*, vol. 42, pp. 52–62, Feb. 1996.
- [19] J. B. Anderson, "An approach for adaptively approximating the viterbi algorithm to reduce power consumption while decoding convolutional codes," *IEEE Trans. Signal Proc.*, vol. 52, no. 5, pp. 1443–1451, May 2004.
- [20] F. Sun and T. Zhong, "Parallel high-throughput limited search trellis decoder vlsi design," *IEEE Trans. VLSI Systems*, vol. 13, no. 9, pp. 1013–1022, Sept. 2005.
- [21] 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 *International Symposium on Low Power Electronics and Design*, 2006, pp. 406–411.
- [22] C. Lin, Y. H. Shih, H. C. Chang, C. Y. Lee, "Design of power-reduction viterbi decoder for WLAN applications," *IEEE Trans. Circuits and Systems I*, vol. 52, no. 6, pp. 1148–1156, June 2005.
- [23] 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*, Oct. 2006, pp. 205–208.
- [24] L. Wang and N. R. Shanbhag, "Low-power filtering via adaptive error-cancellation," *IEEE Trans. Signal Proc.*, vol. 51, no. 2, pp. 575–583, Feb. 2003.
- [25] B. Shim and N. R. Shanbhag, "Low-power digital signal processing via reduced-precision redundancy," *IEEE Trans. on VLSI Systems*, vol. 12, no. 5, pp. 497–510, May 2004.

- [26] G. Varatkar and N. R. Shanbhag, "Energy-efficient motion estimation using error-tolerance," in *International Symposium on Low Power Electronics Design (ISLPED)*, October 2006, pp. 338–342.
- [27] H. L. Lou, "Linear distances as branch metrics for viterbi decoding of trellis codes," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 6, June 2000, pp. 3267–3270.
- [28] A. Hekstra, "An alternative to metric rescaling in viterbi decoders," *IEEE Trans. Comm.*, vol. 37, no. 11, pp. 1220–1222, Nov. 1989.
- [29] 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 Proc.*, vol. 37, no. 2, pp. 90–183, Feb. 2002.
- [30] 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, Aug. 2005.
- [31] J. G. Proakis, *Digital Communications*, 4th ed. New York, NY: McGraw-Hill, 2001.

