# LOW-POWER PRE-DECODING BASED VITERBI DECODER FOR TAIL-BITING CONVOLUTIONAL CODES

*Rami A. Abdallah *, Seok-Jun Lee†, Manish Goel†, and Naresh R. Shanbhag**

*Coordinated Science Laboratory/ECE Department
University of Illinois at Urbana-Champaign, Urbana, IL
email: [rabdall3,shanbhag]@illinois.edu
† Communication and Medical Systems Laboratory
Digital Signal Processing Solutions R&D Center, Texas Instruments, Dallas, TX
email: [seokjun,goel]@ti.com

## ABSTRACT

Low-power and high-throughput Viterbi decoder (VD) for tail-biting convolutional codes is presented in this paper. First, a low complexity radix-4 VD with enhanced decoding features such as *end-state forcing* and *best-state trace back* is presented. Second, *simple pre-decoding* is proposed to decrease the runtime of VD, resulting in significant power saving. The design is implemented in 0.9 V TI 45-nm CMOS process at 100 MHz for *Long Term Evolution* (LTE) [1] as application. More than 90% power saving is achieved with predecoding at a throughput of 120 Mbps and 0.2 dB SNR loss for $10^{-5}$ frame error rate.

***Index Terms—*** Viterbi Decoder, Long Term Evolution, Add-Compare-Select, Trellis Decoding, Tail-Biting Convolutional Code.

## 1. INTRODUCTION

Wireless standards have been evolving to provide higher speed and better quality services for the users. This leads to an increase in the complexity and power of the user equipment (UE). Viterbi decoders (VDs) show good tolerance against channel errors and are employed across different communication standards such as *code division multiple access* (CDMA), *wireless local area network* (WLAN), *digital video broadcast* (DVB), and satellite communications. VDs are still going to be used in next generation broadband mobile wireless standards, in particular *Long Term Evolution* (LTE) [1], which is part of 3rd Generation Partnership Project (3GPP), and mobile Worldwide Interoperability for Microwave Access (WiMAX) [2]. In LTE and WiMAX, VD is used in so called control channels where critical information such as scheduling is transmitted to each UE. Due to the importance of control information, the control channels in LTE and WiMAX should be decoded quickly with increased resiliency to channel errors [3]. Therefore, VDs need to have high throughput and very low frame-error-rate (FER) and their power consumption is also a major concern to provide extended battery life in UE.

Low-power VD architectures has been of great interest and is in fact a well-studied subject. For example, energy-efficiency has been achieved by reducing the number of states [4] or the number of trellis paths [5, 6] at the expense of increased error rates and/or reduced throughput. However, low power VD architectures for tail-biting convolutional codes are underexplored.

In this paper, we present a low-power and high-throughput VD for tail-biting convolutional code which is employed in LTE and WiMAX control channels. Low complexity implementation and architectures are presented to increase decoding throughput and decrease power consumption. The techniques proposed in this paper can be generalized to VDs across different applications and are not limited to LTE and WiMAX.

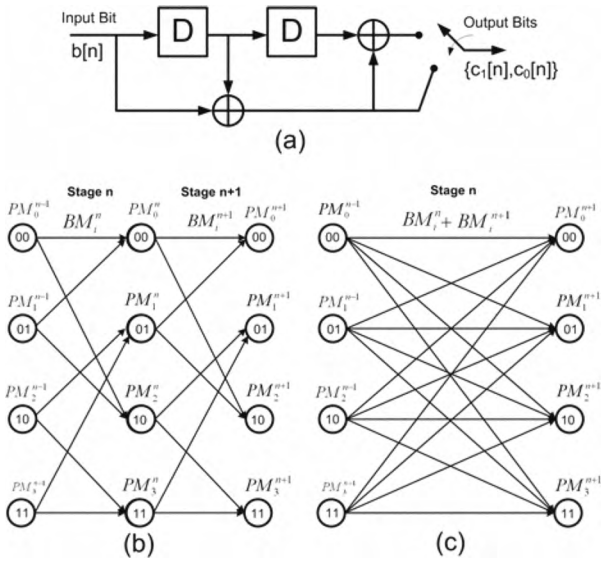



**Fig. 1**. A 4-state rate 1/2 convolutional code: (a) convolutional encoder (b) radix-2 trellis, and (c) a radix-4 trellis.

Section 2 presents background material on VDs. Section 3, presents the design of a low-complexity radix-4 VD for LTE with advanced features support. Section 4 introduces the pre-decoding based VD to reduce power consumption. Section 5 shows simulation results demonstrating FER performance and power savings in 0.9 V TI 45-nm CMOS process.

## 2. BACKGROUND ON VITERBI DECODER

The Viterbi algorithm is an efficient procedure for solving maximum likelihood sequence estimation (MLSE) problems such as decoding of convolutional codes. Figure 1(a) shows a rate 1/2 convolutional encoder that generates two output bits $(c_0[n], c_1[n])$ at each time

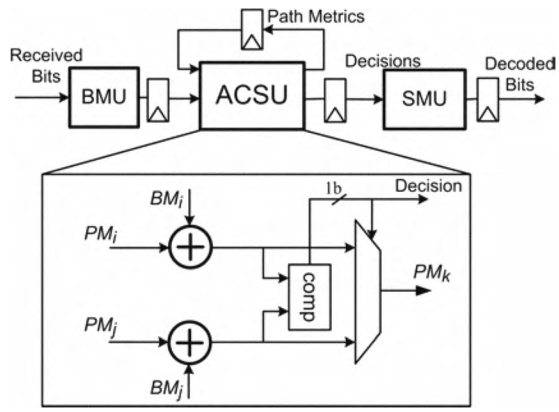



**Fig. 2**. A generic VD architecture.

index $n$ as a function of the input information bits ($b[n]$) and the stored bits in the shift register ($b[n-1], b[n-2]$), where constraint length ($K$) is 3. The output bits are then transmitted over a noisy channel. The Viterbi algorithm estimates the maximum likelihood (ML) sequence of encoder states transitions given the received noisy samples.

The encoding process can be represented by a time indexed trellis as shown in Fig. 1(b). The trellis has four 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. Each path through the trellis has a path metric (PM). The PM is inversely proportional to the log likelihood probability of that path. The Viterbi algorithm recursively finds the path with the minimum PM for each state in case that the distance between the received codeword and the expected codeword is employed as a metric. For the trellis in Fig. 1(b), there are two paths entering each state and the PMs for each state are updated as follows:

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

where two path metrics ($PM_i^{(n)}$ and $PM_j^{(n)}$) and two branch metrics ($BM_i^{(n)}$ and $BM_j^{(n)}$) are employed to generate an updated value $PM_k^{(n+1)}$. Equation (1) is implemented in an add-compare-select (ACS).

A generic VD architecture is shown in Fig. 2. The branch metric unit (BMU) generates BMs for all the transitions. The ACS unit (ACSU) consists of multiple ACSs that recursively update the PM of each state according to (1). The survivor memory unit (SMU) keeps track of the survivor path of each state by storing the ACSU decisions.

Due to the feedback loop in the ACSU, the VD operating frequency is limited to the critical path delay in the ACSU. Increasing the throughput is done by processing more than one trellis section in a single clock cycle or what is termed as higher radix processing [7]. Fig. 1(b) is a radix-2 trellis where one section is processed at a time while Fig. 1(c) is a radix-4 trellis where two sections are combined in a single section to be processed at a time. In general radix-$k$ processing, where $k$ is a power of 2, $log_2(k)$ trellis sections are processed in one time-step.

In streaming applications, where the length of the received sequence is large, the VD has low probability of error. On the other hand, in applications where data transmission is done in small frames, multiple Viterbi decoding iterations are performed on the same frame to achieve low FER. In order to allow multiple iterations on the same frame, the starting and end states of the convolutional encoder need to be identical. Two approaches: *zero-terminating* and *tail-biting*, are usually employed. In *zero-terminating*, zeros are added at the end of each frame to force the same starting and end state. However, this decreases the coding rate. In *tail-biting* [1, 2], the encoder shift-register is initialized to the last bits of the frame. Here, the coding rate is maintained but the starting and end state is unknown and the VD has to do more work to estimate this state.

## 3. LOW-COMPLEXITY HIGH-THROUGHPUT VITERBI DECODER FOR TAIL-BITING CONVOLUTIONAL CODE

In this section, several enhanced features for decoding of tail-biting convolutional code are presented by taking LTE control channels as an example. In LTE, a tail-biting convolutional code with constraint length ($K$) 7, i.e. $2^{K-1} = 64$ states, and mother code rate of 1/3 is used in the control channels, specifically the Physical Broadcast Channel (PBCH) and Physical Downlink Control Channel (PDCCH) [1]. The generators for the three codeword bits are $(133, 171, 165)$ in octal. We design VD for control channels and it can be easily reconfigured to handle PDCCH and PBCH channels since the frame length can be configurable.

PDCCH carries scheduling assignments and other control parameters that inform to each UE if the UE is the intended receiver of the data in the data channel. PDCCH information must be decoded quickly so that the user knows the subsequent steps or whether it has to stay in sleeping mode [3]. This demands high-throughput decoding. Thus, in our proposed LTE VD design ACSU consists of state-parallel ACSs, where the PMs for each state are updated in parallel, and uses radix-4 architecture, where two stages of the trellis are processed in a single clock cycle, to ensure high throughput.

### 3.1. BMU and ACSU Architectures

For the mother code rate 1/3 convolutional code, each bit of the three received codeword bits is quantized to 4 bits resulting in a 6-bit BM per a trellis section. Two trellis stages are combined under radix-4 as shown in Fig. 1(c) and the BM precision is 7-bit. The BMU is based on Hamming distance to generate 64 different BMs for ACSU. Two's complement modular arithmetic is used in the ACSU to ensure correct operation in presence of PM overflow [8]. Thus, with 64-states radix-4 trellis the PM precision must be 10-bit.

Two approaches are commonly used to design a radix-4 ACS where 4 PMs need to be updated and compared to find the minimum across the 4 candidates (see Fig. 3(a) and (b)). The first approach, referred to as *2-stage-compare*, uses two stages of comparisons and the second, referred to as *1-stage-compare*, uses a single stage with 6 parallel comparators [9]. The adders and the comparators are based on two's complement least significant bit (LSB) first addition and subtraction respectively. That is why the comparator can proceed in parallel with the adder when the comparator is after the adder. Therefore, the critical path delay is dominated by two carry ripples in the first approach and by a single carry ripple in the second approach. The delay and complexity estimates are reported in Table 1. *1-stage-compare* delay is around two times less than *2-stage-compare* delay. However, *1-stage-compare* has much higher complexity than *2-stage-compare* which results in higher power consumption.

To avoid increased complexity and power consumption in radix-4 while maintaining throughput, we cascade two radix-2 ACSUs in

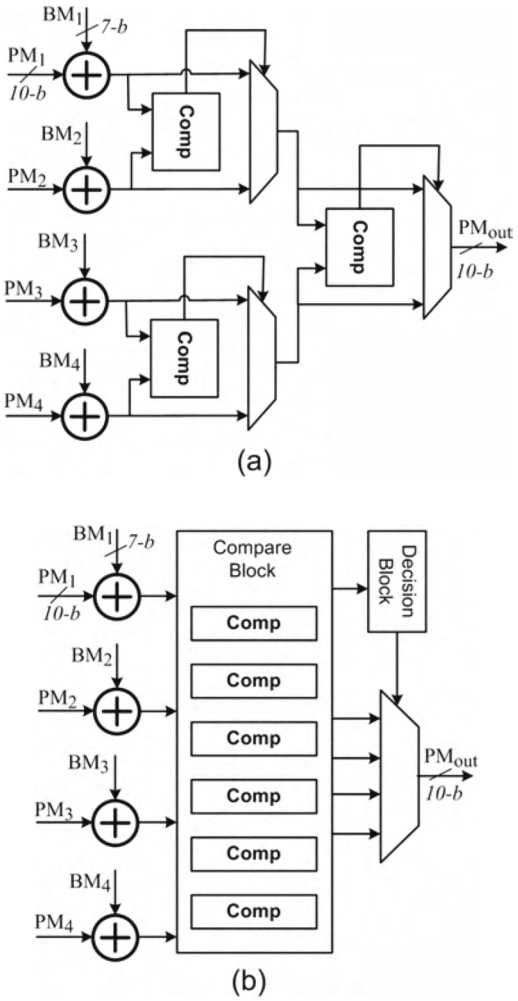



**Fig. 3**. Conventional architectures of radix-4 ACS: (a) *2-stage-compare* and (b) *1-stage-compare*.

series and we refer to this proposed approach as *Cascaded-radix-2*. The ACSU for a radix-2 trellis is duplicated so that the first set of ACSs processes the first trellis stage and passes its results to the next set of ACSs to process the next trellis stage in the same clock cycle. For example, in the trellis of Fig. 1(b) and (c), *Cascaded-radix-2* uses 8 radix-2 ACSs based on trellis (b) while *2-stage-compare* and *1-stage-compare* use 4 radix-4 ACSs each. In *Cascaded-radix-2*, the BM and PM precisions are the same as radix-2 processing and equal to 6-bit and 9-bit respectively for LTE. *Cascaded-radix-2* has delay slightly smaller than *2-stage-compare* and shows around 25% complexity savings as reported in Table 1. Another benefit of *Cascaded-radix-2* over the two approaches is the decrease in the complexity of the BMU since it only needs to duplicate the BMU of a radix-2 trellis.

### 3.2. Enhanced Feature Support for Tail-Biting Convolutional Codes

In this study, a codeword have 40 information bits or 120 codeword bits [1]. It is encoded using the tail-biting convolutional code described in previous section. The proposed VD performs two iterations on one codeword as described next to achieve very low FER. The decoding process is illustrated in Fig. 4 where we illustrate a 4-state trellis instead of 64-state trellis for simplicity. Since the starting state is unknown, the VD starts with equal probability for all 64 states, i.e blind startup where all PMs are initialized to zero, and performs the first decoding iteration on the received frame. After the end of the first iteration, the states will have different probabilities of occurrence which is reflected in their different PM values that are going to be used in the following iteration. Continuing through the second iteration, we know by the tail-biting property of the convolutional code that the end state (ES) for each path in the trellis is the same as its starting state (SS). Therefore, we force the ES for each path in the trellis during the second iteration to be the same as its SS at the beginning of the second iteration to improve FER performance. This is achieved by two architectural changes in the ACSU:

- ***Starting state storage:***
  we need to maintain the SS for each path in the trellis starting from the second iteration. At the ACSU architectural level, each path is characterized now by a PM value and a SS. An additional 6-bit register is used in each ACS to store the SS. For example in the $k^{th}$ ACS in Fig. 5, the register value $SS_k$ gets updated by the SS of the path selected among the competing paths inside the ACS, i.e. either $SS_i$ or $SS_j$ during the second iteration.

**Table 1**. Delay and complexity of radix-4 ACS. $HA$: 1-b half adder, $FA$: 1-b full adder, $INV$: 1-b inverter, $t_{FA}$: delay of 1-b full adder, $t_{mux}$: delay of a 2-input Mux, $t_{inv}$: delay of an inverter, and $t_{DEC}$: delay of the decision block.

| Architecture | (BM,PM) Precision | Delay | Complexity |
|---|---|---|---|
| *2-stage-compare* | $(7, 10)$ | $21t_{FA} + 2t_{mux} + t_{inv}$ | *40 FA + 30 HA + 30 INV + 30 MUX* |
| *1-stage-compare* | $(7, 10)$ | $11t_{FA} + 4t_{mux} + t_{inv} + t_{DEC}$ | *40 FA + 60 HA + 60 INV + 30 MUX + DEC* |
| *Cascaded-radix-2* (proposed) | $(6, 9)$ | $20t_{FA} + 2t_{mux} + t_{inv}$ | $2 \times$ (*18 FA + 9 HA + 9 INV + 9 MUX*) |

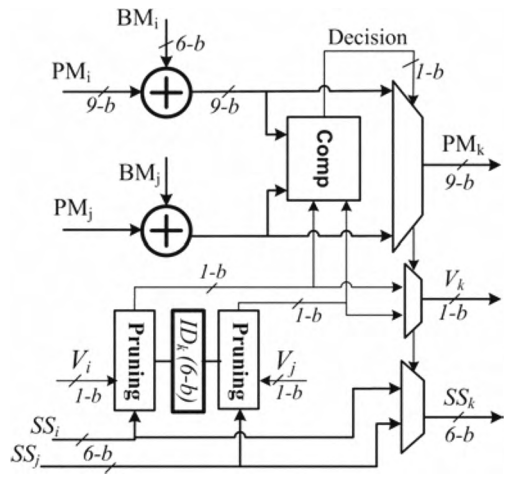



**Fig. 5**. An ACS for tail-biting code with end state forcing. The ACSU is duplicated in the proposed *Cascaded-radix-2* to enable radix-4 processing.

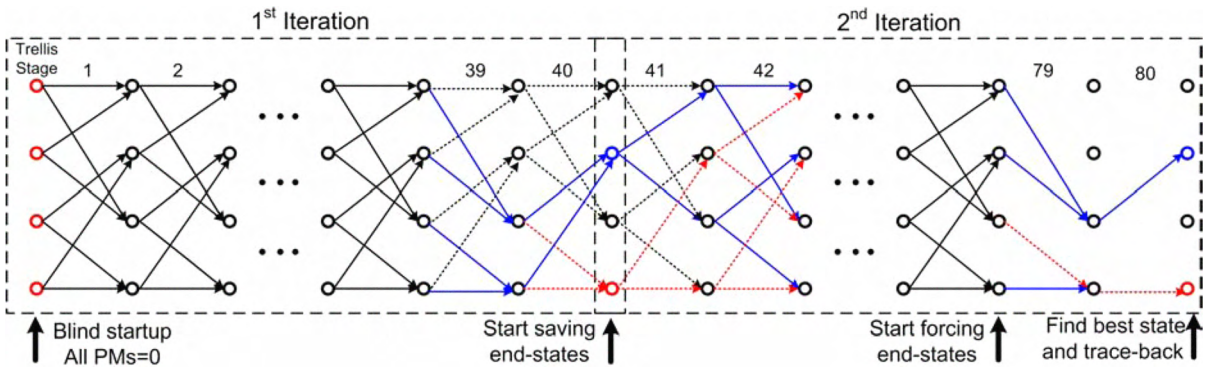



**Fig. 4**. Decoding process for 40-bit codeword. Two Viterbi decoding iterations are performed.

- ***End state forcing:***
  The ES for each path is the same as its SS. For each codeword, we start forcing the ES for each path at trellis stage 75. Note that in Fig. 4, end state forcing starts at stage 79 since 4 states are assumed instead of 64 states. End state forcing is implemented in ACS using a path validation signal to prune the trellis paths that do not lead to an ES that is equal to the stored SS in each path. This signal is denoted as $V$ in Fig. 5. Disregarding a valid path from the set of possible transitions or pruning it is done by the pruning block in Fig. 5. Depending on the trellis stage being $75, 76, ..., 80$, a bit in the SS of the path is selected and compared to the corresponding bit in its next state, i.e. the 6-bit ACS index which is hardwired in each ACS and is denoted in the pruning block in Fig. 5 as $ID_k$. If the bits are the same then the path is valid else it is pruned. The pruning decision is passed to the PM compare block to be considered during comparison stage. If the path is invalid, then the other one is selected. If both are invalid, then the output path is also invalid.

### 3.3. Trace-Back Architecture

Once the second decoding iteration is done, the next step is to trace-back the ACSU decisions stored in the survivor memory unit (SMU). To improve FER performance, we start tracing-back from the best state which is the state with the minimum PM. This increases the complexity of the design since 63 comparators and multiplexors are needed to find the minimum across 64 PMs. In this work, we propose to reuse the ACSU when second decoding iteration is done while zeroing out all BMs to search for the minimum path. When BMs are zeros, ACS behaves similar to a compare-select. The trace-back phase for each PDCCH frame is outlined as follows and is illustrated in Fig. 6:

1. ***Searching for best state:*** after second iteration, three clock cycles are additionally needed by the radix-4 *Cascaded-radix-2* ACSU to search the minimum metric state across 64 PMs while the ACSU decisions are stored in the SMU. During these clock cycles, all BMs are set to zero and a specific set of ACSs in the ACSU is activated each clock cycle to behave like compare-select and pass the minimum. The only drawback of ACSU reuse over a dedicated minimum-search unit is the loss of 3 clock cycles.
2. ***Tracing-back:*** to reach the best state at the end of the second iteration, additional three clock cycles are also needed to trace-back the decisions stored in SMU when ACSU finishes searching the best-state. Once the best state is reached, simulations show that the decisions are more reliable in trellis stages 25 to 64. Thus, we need to trace-back without decoding information bits from stage 80 to 65.
3. ***Frame Decoding:*** from stage 64 to 25, decisions are traced back and the information bits are decoded. Last-input first-output (LIFO) buffers are employed to correct the order of the decoded information bits.

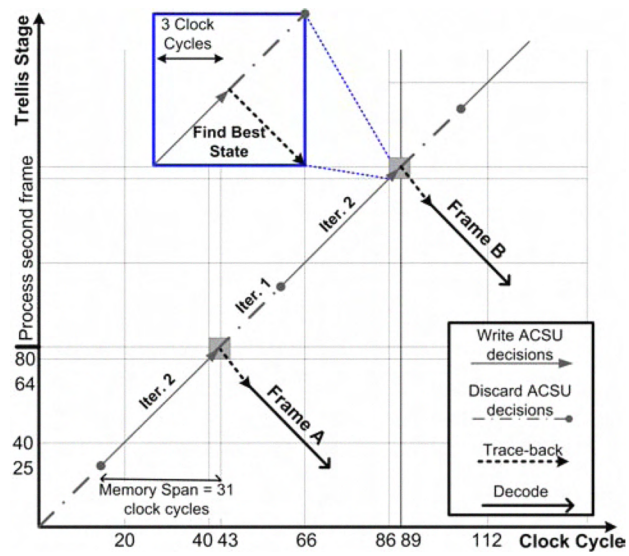



**Fig. 6**. Memory management for 40-information-bit codeword frame decoding with radix-4 processing and ACSU reuse for best-state search.

The selected path history from the 64 *Cascaded-radix-2* ACSU consists of 128 bits and it is stored in SMU each clock cycle. SMU needs to store decision vectors from clock cycle 13 to 43 so its length needs to be equal to 31 for 40-information-bit codeword (see Fig. 6). In addition to that, it consists of two banks to allow back-to-back frame processing so that while tracing-back the first frame, the second frame is being processed by ACSU. Therefore, the VD in this paper can decode 40 bits every 43 clock cycles.

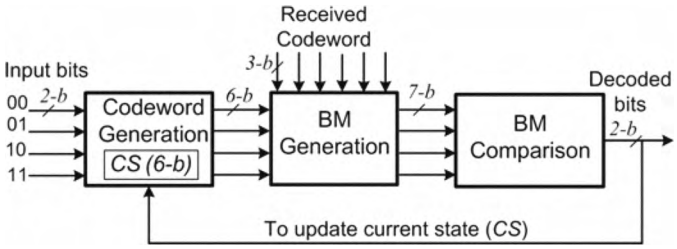



**Fig. 7**. Simple predecoder (SPD) for convolutional codes with two bits decoding.

## 4. LOW-POWER PRE-DECODING BASED VITERBI DECODER

In previous section, a low-complexity high-throughput VD with enhanced decoding features is introduced for tail-biting convolutional code. In this section, we propose a new way to reduce the power consumed by the designed VD. The main idea is that under operating channel conditions the full decoding power of the VD is not needed all the time and a simple decoder may be enough to decode the received frame correctly [10]. Therefore, if we know when the simple decoder is able to decode the frame correctly, turning off the VD will save decoding power. The predecoding based VD operates as follows:

1. Decode the frame using a simple predecoder (SPD).
2. Detect if the decoded frame is in error. If it is in error, proceed to step 3. If it is not, go back to 1 and process the next frame.
3. Decode the frame using VD.

### 4.1. Design of Simple Predecoder and Failure Detection

The SPD design is based on symbol-by-symbol detection and shown in Fig.7. It maintains only a single path through the radix-4 trellis and outputs the decoded bits at each stage directly. Knowing the current state (CS) and the four BMs corresponding to the possible transmitted bits 00, 01, 10, or 11, SPD finds the minimum BM to decode bits and update its CS.

For tail-biting code, the issue is that the starting state is unknown. We use the VD for 3 clock cycles to update the PMs for all 64 states and then reuse the ACSU as described in previous section for another 6 clock cycles to find the best one and trace-back to the best state. Therefore, SPD can decode a 40-information-bit codeword frame every 29 clock cycles.

There are two types of errors in detecting when the SPD fails: *miss detection* and *false alarm*. Miss detection occurs when we assume that SPD correctly decoded the frame while it did not. False alarm occurs when we assume that SPD did not decode the frame correctly while it did. In designing SPD failure detection schemes, the probability of miss detection is of more concern than the probability of false alarm since miss detection directly affects FER and may cause an error floor while false alarm affects the throughput and power savings. Three SPD failure detection schemes are proposed and they were optimized to decrease miss detection for a 40-information-bit codeword frame using simulations . We assume that the SPD fails if any of the three schemes detects a failure. The detection schemes are:

1. ***Tail-biting property:*** If last 6 decoded bits in a frame do not match with the starting state, then SPD is detected as "fail".

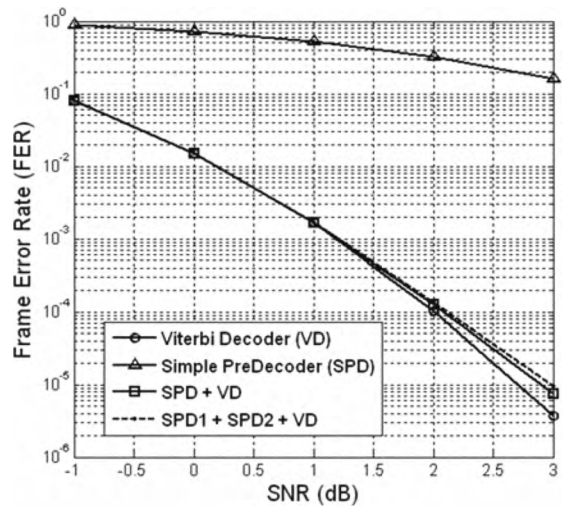



**Fig. 8**. FER performance of proposed predecoding schemes for 40-information-bit codeword frame.

**Table 2**. Normalized complexity and power for different blocks in 0.9 V TI 45-nm CMOS process.

| Component | VD | BMU | ACSU | SMU | SPD | Failure Detection |
|---|---|---|---|---|---|---|
| Complexity | 1 | 0.018 | 0.502 | 0.48 | 0.011 | 0.012 |
| Power | 1 | 0.015 | 0.405 | 0.58 | 0.01 | 0.01 |

2. ***Branch metric reliability:*** The BM value represents the distance between the received codeword and the expected codeword and can be used as a reliability measure. The larger the BMs we obtain, the less likely the SPD is on the right path in the trellis. Therefore, if the number of trellis sections with large BM values ($> 20$) is greater than 4, then SPD is detected as "fail".
3. ***Branch metric difference:*** The SPD is comparing 4 BMs in each clock cycle. If the BM values get closer to each other, the more likely the SPD will fail. Therefore, if the difference between the two best BMs in any stage is less than 12 in more than any 7 stages, then SPD is detected as "fail".

To further decrease power and turn-off the VD for longer time, we also propose another predecoding based VD scheme where we use two SPDs (SPD1 and SPD2) before the VD. Decoding priority is as follows: SPD1, SPD2, and then VD, i.e., first we run SPD1. if it fails, we run SPD2. If SPD2 also fails, we run the VD. The SPDs are designed the same as above but one proceeds in a forward manner across the trellis while the other proceeds in a backward manner. This ensures that the two SPDs fail independently.

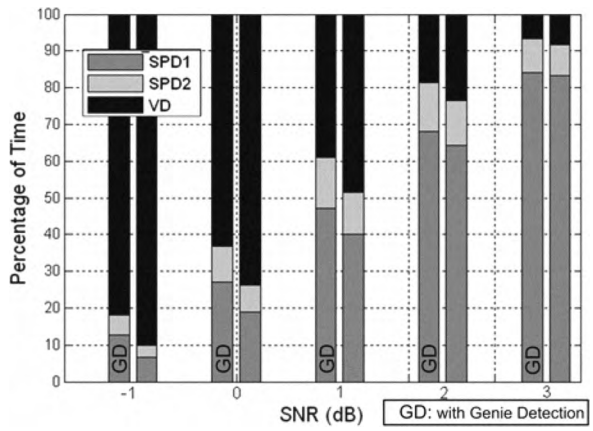
## 5. SIMULATIONS AND RESULTS

This section describes the FER performance and power savings achieved by the proposed schemes for 40-bit codeword decoding. The proposed VD is implemented and synthesized in 0.9 V TI 45-nm CMOS process at 100 MHz clock frequency. Table 2 shows the complexity and power breakup across the different blocks in the proposed VDs. Complexity and power is dominated by ACSU and

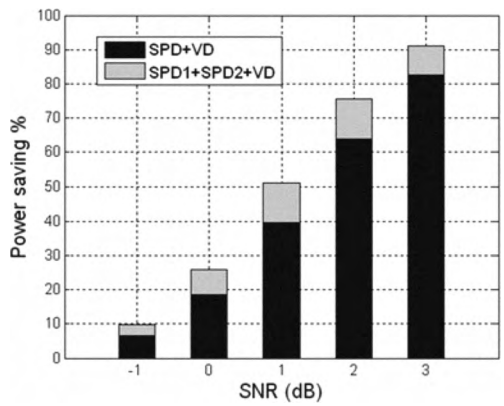
SMU. The SPD with failure detection logic constitutes only around 1% of the overall VD.

Figure 8 shows the FER for VD, SPD, VD with single SPD, and VD with two SPDs. The channel is assumed to be additive white Gaussian noise channel. At typical channel FERs ranging from $10^{-2}$ to $10^{-4}$, the three schemes behave almost identically. At higher FERs, miss detection probability starts to show slightly degraded performance for predecoding based schemes. For example, at FER of $10^{-5}$, 0.25 dB and 0.3 dB loss in SNR is observed for VD with a single SPD and two SPDs respectively. The FER of SPD ranges from 0.9 to 0.15 as SNR is increased which results in longer power-down time for VD and more power savings as discussed next.

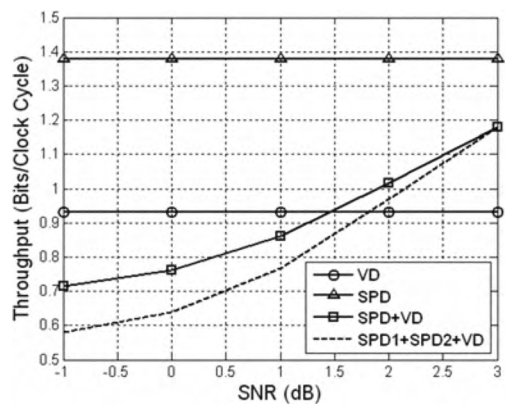
Figure 9(a) shows the runtime of predecoding based VD with single and two SPDs. Genie detection (GD) in the figure corresponds to the runtime if we were able to perfectly know when SPD1 and SPD2 will fail. The SPD failure detection technique proposed in this paper behaves close to the GD and improves at higher SNR as the effect of channel noise becomes limited and the BMs become better indicators of the reliability of the trellis path chosen by the SPD. As expected also, as SNR increases the VD runtime will decrease and most of the decoding will be performed by the SPDs only. Power saving in Fig. 9(b) varies from 25% to 75% for typical SNRs between 0 dB and 3 dB. At higher SNRs, power saving reaches more than 90%. Another factor to consider is the variable throughput with predecoding since SPD needs 29 clock cycles to decode a PDCCH frame while VD needs 43 clock cycles and sometimes we are running the two decoders. Figure 9(c) shows the throughput obtained for the different setups at different SNRs. For SNRs greater than 1.5 dB, VD with a single SPD starts to deliver throughput higher than VD. To increase the throughput of the predecoding based schemes at lower SNRs, SPD can process more than 2 trellis sections per clock cycle at the expense of small increase in complexity overhead.




(a)




(b)




(c)

**Fig. 9**. Proposed predecoding schemes for VD: (a) runtime, (b) power, and (c) throughput.

## 6. REFERENCES


[1] 3GPP TS 36.212, Evolved Universal Terrestrial Radio Access (E-UTRA): Mutiplexing and Channel coding", version 8.4.0, September 2008.

[2] WiMAX Forum Mobile System Profile, version 1.1.0, July 2006.

[3] S. Lee, M. Goel, Y. Zhu, J. Ren, and Y. Sun, "Forward Error Correction Decoding for WiMAX and 3GPP LTE Modems," *Asilomar Conference on Signals, Systems, and Computers*, 2008.

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

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

[6] 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.

[7] G. Fettweis and H. Meyr, " Parallel Viterbi algorithm implementation: Breaking the ACS-bottleneck," *IEEE Trans. Commun.*, vol. 37, pp. 785−790, August 1989.

[8] A. P. Hekstra," An alternative to metric rescaling in Viterbi decoders," *IEEE Trans. Comm.*, vol. 37, pp. 1220−1222, November 1989.

[9] P. Black and T. Meng," A 140 Mb/s 32-state radix-4 Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 27, pp. 18771885, December 1992.

[10] W. Shao and L. Brackenbury," Pre-processing of Convolutional Codes for Reducing Decoding Power Consumption," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Las Vegas, Nevada, 2008.