

Design and VLSI architecture of non-polynomial based low probability of error (P_b) Viterbi decoder

C Arun^{1*} and V Rajamani²

¹Department of Information Technology, Sri Venkateswara College of Engineering, Chennai 602 105, India

²Centre for Research and Development, Department of Electronics and Communication Engineering, PSNA College of Engineering and Technology, Dindigul 624 622, India

Received 14 July 2008; revised 27 November 2008; accepted 03 December 2008

This paper presents implementation of a new non-polynomial approach to design a high throughput with reduced bit error probability Viterbi decoder. Increase in d_{free} has been achieved by proposed non-polynomial convolutional coding method. A decoder system (code rate $k/n=1/6$, constrain length $K=4$) has been implemented on Xilinx VERTEX-E. Performance of Viterbi decoder with proposed method has been improved from 27% to 75%. High speed (60.299 Mbps) and low bit error rate (BER) are achieved for Viterbi decoder. Proposed Viterbi decoder provides satisfactory probability of error (P_b) performance and high operating speed under conditions including AWGN, co-channel interference and adjacent channel interference environments.

Keywords: Add compare select (ACS), Branch metric unit (BMU), Free distance, Low bit error rate, Non-polynomial approach, Trace back unit (TBU), Viterbi algorithm

Introduction

Viterbi decoders are used to decode convolutional coding, which has been used in deep space communications as well as wireless communications. A wireless cellular standard for CDMA (code division multiple access), IS-95, employs convolutional coding. Viterbi algorithm works well for less-complex codes, indicated by constraint length K . For decoding more powerful codes with large constraint lengths, adaptive Viterbi algorithm (AVA) has been developed¹. For many signal-processing systems², it is possible to exploit variations in signals to vary computation and memory requirements.

For high-speed communication, Chan¹ implemented an adaptive Viterbi decoder, which discards some states (in trellis) with high path metrics dynamically during decoding process^{3,4}, suggesting the use of a scarce state transition (SST) scheme. The scheme employs a simple pre-decoder followed by a pre-encoder to minimize signal transitions at the input of a conventional Viterbi decoder, which leads to lower bit error rate (BER). Kang *et al*⁵ studied designing a low probability of error (P_b)

Viterbi decoder for IS-95 CDMA system. Lin *et al*⁶ implemented a soft decision decoder for 8-level (3 bit), 16 level (4 bit) and 32 level (5-bit) quantization. Increasing quantization level improves error probability and reduces signal-to-noise ratio but has increased hardware complexity. To address BER requirements of WLAN and Broadband applications, 3-bit soft decision Viterbi decoder was implemented as a reconfigurable VHDL macrocell.

Per-survivor processing (PSP) Viterbi decoder⁷, which performs noncoherent sequence detection for Bluetooth, simplifies overall hardware architecture of decoder. At a typical target BER (10^{-3}) for Bluetooth devices⁸, hardware implementation of PSP decoder requires signal-to-noise ratio (SNR) (10 dB); therefore, it achieves a performance gain of >6 dB compared to simpler discriminator detectors⁹. Despite being limited by speed performance of FPGA, maximum decoding rate of PSP Viterbi decoder (20 Mbits/s) far exceeds 1 Mbits/s, required by bluetooth standard⁸.

This paper presents a low probability of error Viterbi decoder suitable for future wireless communication applications.

* Author for correspondence
E-mail: carun@svce.ac.in

Polynomial Based Convolutional Encoder

Among encoder representation (generator polynomial, state diagram, trellis diagram), state diagram is chosen as it shows a more compact structure in describing possible states and outputs of encoder. For encoder, generator polynomials are $g_1(x)$, $g_2(x)$, ..., $g_6(x)$, as

$$g_1(x) = 1 + X + X^2 + X^3 \quad \dots(1)$$

$$g_2(x) = 1 + X + X^2 + X^3 \quad \dots(2)$$

$$g_3(x) = 1 + X^2 + X^3 \quad \dots(3)$$

$$g_4(x) = 1 + X^2 + X^3 \quad \dots(4)$$

$$g_5(x) = 1 + X + X^3 \quad \dots(5)$$

$$g_6(x) = 1 + X + X^3 \quad \dots(6)$$

Lower order terms in polynomial correspond to input stage. Output sequence is given as

$$n_i(x) = m(x) g_j(X), \quad \dots(7)$$

where, $i, j = 1, 2, \dots, 6$.

Convolutional encoder requires a number of 0 bits to be appended to the end of input of data sequence for clearing or flushing encoding shift register of data bits, called redundant or parity bits. A convolutional encoder belongs to the class of finite state machines. 'Finite' refers that there are only a finite number of unique states that machine can encounter¹⁰. In state diagram (Fig. 1), circles represent states of encoder and paths between states represent output branch word (BW) resulting from such transitions. The 6 bits in transition branch denotes output sequence and dotted lines show transition triggered by 1 and solid lines show transition triggered by 0.

Proposed System Design

Free Distance

In state diagram, d_{free} is minimum hamming distance between any two different paths of any length l , where paths begin in same state (i) and end in same state (j), where i need not to be equal to j . Minimum distance is an important metric because it inherently gives error correcting power of code word (CW). Value of d_{free} gives a measure of how many bits can be "flipped" in order for received CW not to be a given CW n subspace C . Daut¹¹ derived a simple upper bound minimum free distance of a rate $1/n$ convolutional code as

$$d_{\text{free}} \leq \min_{l \geq 1} \left[\frac{2^{l-1}}{2^l - 1} (K + l - 1) n \right] \quad \dots(8)$$

where $\lfloor x \rfloor$, largest integer contained in x ; K , constraint length; mb , m burst of errors of length b ; n , number of encoded output bits. d_{free} increases if either constraint length increases or code rate r decreases. These factors affect system performance and implementation of Viterbi decoder.

Fig. 2 shows determination of free distance for $r=1/6$ and $K=4$ and labeling each branch with its hamming distance from all-zeros CW instead of its BW symbols. A path, at a distance 20 from all-zero paths, departs from all zero paths at time t_1 and merges at time t_5 . Similarly, there are two paths at distance 20; one departs at time t_1 and merges at t_6 and other departs at time t_1 and merges at t_7 and so on. From dashed and solid lines (Fig. 2), input bits for distance 20 (t_1 - t_5) are 1000; it differs in only one bit from all-zero path. Similarly, input bits for distance 20 (t_1 - t_6) and 20 (t_1 - t_7) are 11000 and 111000, which differ in position 2 and 3 from all-zero paths. For $d_{\text{free}} = 20$, total number of channel error bits that can be corrected is 9. By following non-polynomial structure, number of errors that can be corrected gets increased > 9 .

Probability of Error

Probability of error can be determined as

$$P_B \leq Q \left(\sqrt{2d_f \frac{E_c}{N_0}} \right) \exp \left(d_f \frac{E_c}{N_0} \right) \frac{dT(D, L, N)}{dN} \Big|_{N=1, L=1, D=\exp(-E_c/N_0)} \quad \dots(9)$$

where, d_f free distance; E_c/N_0 , channel symbol energy to noise power spectral density; $Q(x)$, complementary

error function; $\frac{dT(D, L, N)}{dN}$, transfer function; $E_c/N_0 = r E_b/N_0$, r , code rate; E_b/N_0 , information bit energy to noise power spectral density. Low probability of error can be achieved by increasing free distance.

There are two error probabilities associated with convolutional codes, first event (P_e) and bit error (P_b). P_e is the probability that an error begins at a particular time. P_b is the average number of bit errors in decoded sequence. For hard-decision decoding, P_e and P_b are defined as

$$P_e < T(D, L, N) \Big|_D = \sqrt{4p(1-p)}, L = 1, N = 1 \quad \dots(10)$$

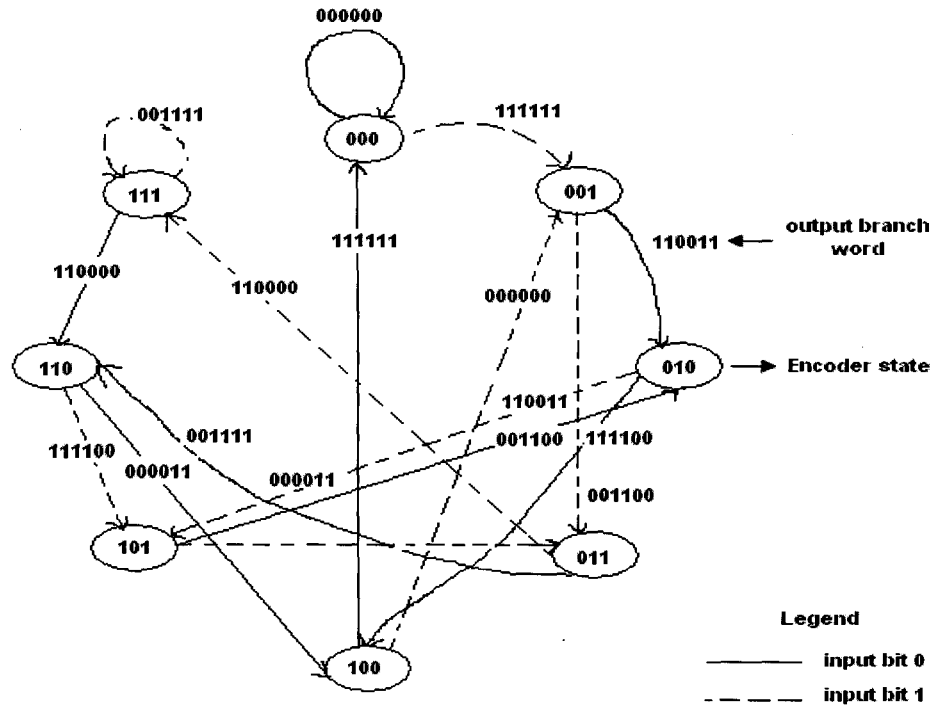
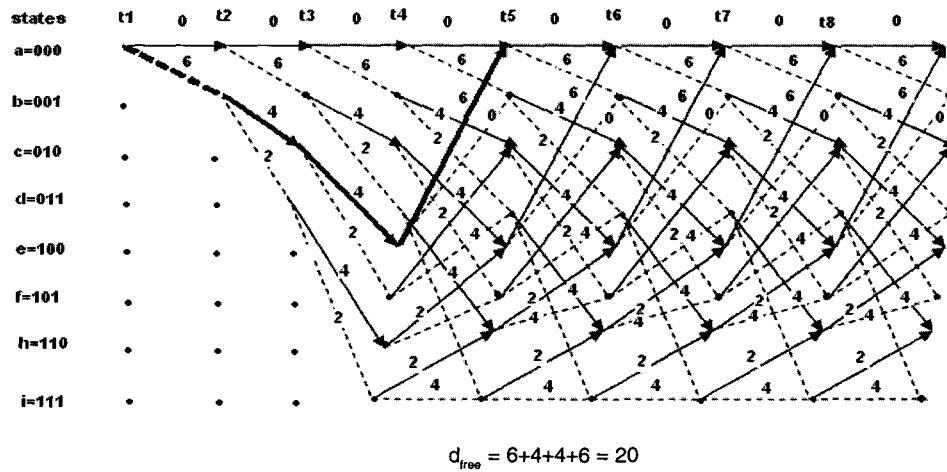


Fig. 1—State diagram of polynomial based conventional encoder with K=4, r=1/6

Fig. 2—Polynomial based Trellis diagram with $d_{\text{free}} = 20$

$$P_b < \frac{dT(D, L, N)}{dN} \Big|_{D = \sqrt{4P(1-P)}, L=1, N=1} \dots (11)$$

where, $p=Q\left(\sqrt{\frac{2rE_b}{N_0}}\right)$ is probability of channel symbol error.

Complementary error function is given by $Q(x)$ as

$$Q(x) = \int_x^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} du \dots (12)$$

Table 1—Non-polynomial based convolutional encoder output transition table with d-free=21 and d-free=24

Current state	Non-polynomial based without repetition output symbols		Non-polynomial based with repetition output symbols	
	Input = 0	Input = 1	Input = 0	Input = 1
000	111000	111111	000011	111111
001	111110	100010	111111	010100
010	110111	110000	111111	001100
011	110011	100000	001000	110100
100	011111	010010	111111	100001
101	011000	010011	001001	000110
110	010100	110010	101010	010110
111	010001	111101	011110	100010

For $x > 3$, it can be approximated as

$$Q(x) \approx \frac{1}{x\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \quad \dots(13)$$

Error correcting capability of code depends on minimum distance. Since a convolutional code is a linear code, minimum distance is just the distance between each of CW sequences and all 0's sequence. Error correcting capability of code has been computed as

$$t = \frac{d_{\text{free}} - 1}{2} \quad \dots(14)$$

where, t is maximum number of errors that can be corrected.

Design of Proposed Non-Polynomial Based Structure

Encoding is accomplished through the addition of redundant bits to transmit information symbols. Redundant bits make decoders capable to correct transmission errors. Convolutional codes¹² form a set of popular error-correction codes. The d_{free} and P_b are calculated based on polynomial approach and non-polynomial approach for a constant code rate for different constraint lengths. The d_{free} of convolutional coder is increased to a high value by non-polynomial method. By increasing free distance, number of errors that can be corrected is increased and BER is reduced. The d_{free} can be increased by changing branch words of encoder with repetition values and without repetition values.

Without Repetition of Branch Word

State diagram values ($r=1/6$ and $K=4$) consisting of BW (Table 1), which are not repeated as in the case of original polynomial structure, are defined randomly with adjacent BWs having difference of three bits. So, this is called as non-polynomial based structure. By selecting like this, d_{free} is increased up to 21 and BER is reduced to some extent compared to the previous polynomial structure. Output transition occurs whenever input bit is 1 or 0. Catastrophic error can be avoided by giving non-zero weight to branch word of closed loop.

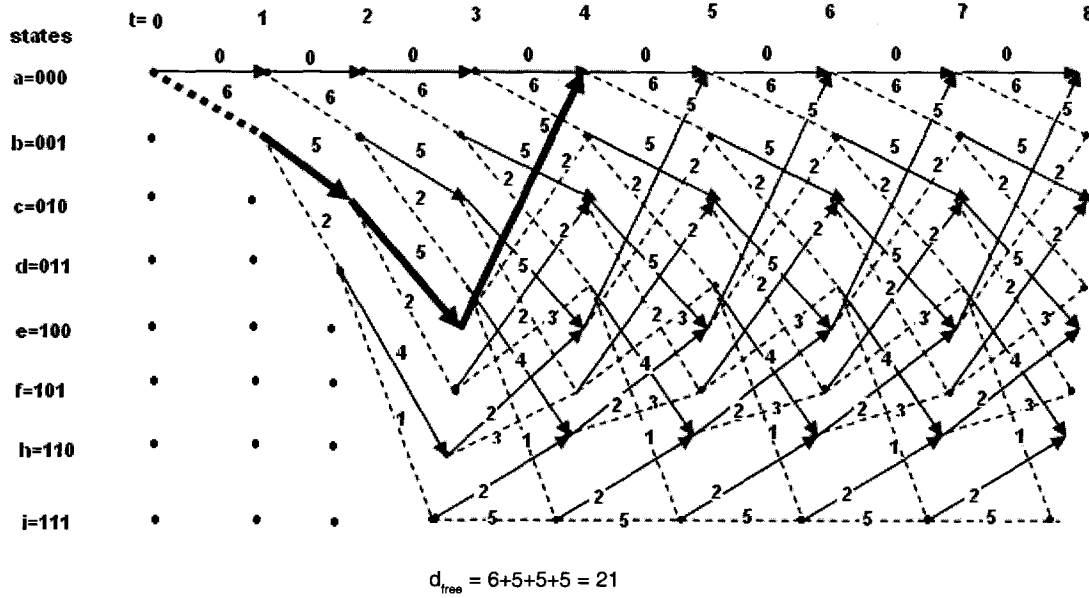
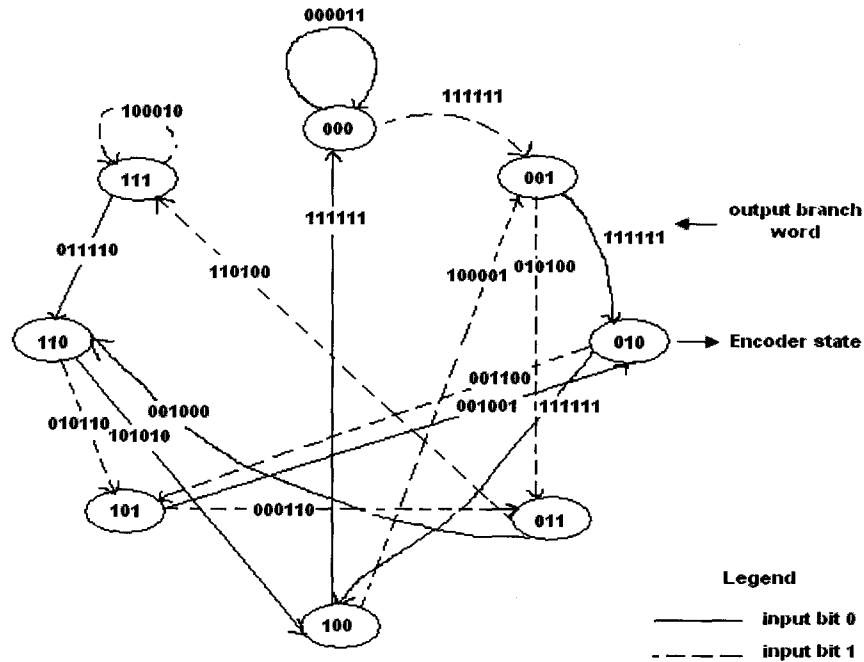
Since free distance depends on hamming weights of 2, 3, 5 and 9th branches of trellis, by keeping these values to a maximum, d_{free} can be increased. By this way, one gets a d_{free} of 21 (Fig. 3) and total number of errors that can be corrected gets increased to 10 using Eq. (14).

With Repetition of Branch Word

State diagram ($r=1/6$ and $K=4$) consisting of BW, is repeated more than twice such that d_{free} gets increased up to 24 (Fig. 4). These values are defined randomly with adjacent BWs having difference of 4 bits. Since d_{free} depends on hamming weights of 2, 3, 5 and 9th branches of trellis and maximum value is 6 for all branches, by keeping this value to all these branches, d_{free} can be increased to a maximum. By this way, one gets a d_{free} of 24 (Fig. 5) and total number of errors that can be corrected gets increased up to $t=12$ using Eq. (14). BER is reduced to maximum extent compared to previous polynomial structure and BW without repetition using this method.

Architecture of Viterbi Decoder

Viterbi algorithm¹² determines a minimum distance path with regard to hamming distances applied to each

Fig. 3—Trellis diagram for non-polynomial based without repetition of branch word ($d_{\text{free}}=21$)Fig. 4—State diagram of non-polynomial based encoder with repetition of branch word ($d_{\text{free}}=24$)

received symbol. A limiting factor in Viterbi decoder implementations is the need to preserve candidate paths at all 2^{K-1} trellis states for each received symbol. This requirement leads to an exponential growth in the amount of computation performed and amount of path storage retained as constraint length K grows. Most

hardware implementations of Viterbi algorithm¹³ are split into three parts: i) branch metric generators (BMG); ii) add-compare-select (ACS) units; and iii) survivor memory unit. A BMG unit determines hamming distances between received and expected symbols. An ACS unit determines path costs and identifies lowest-

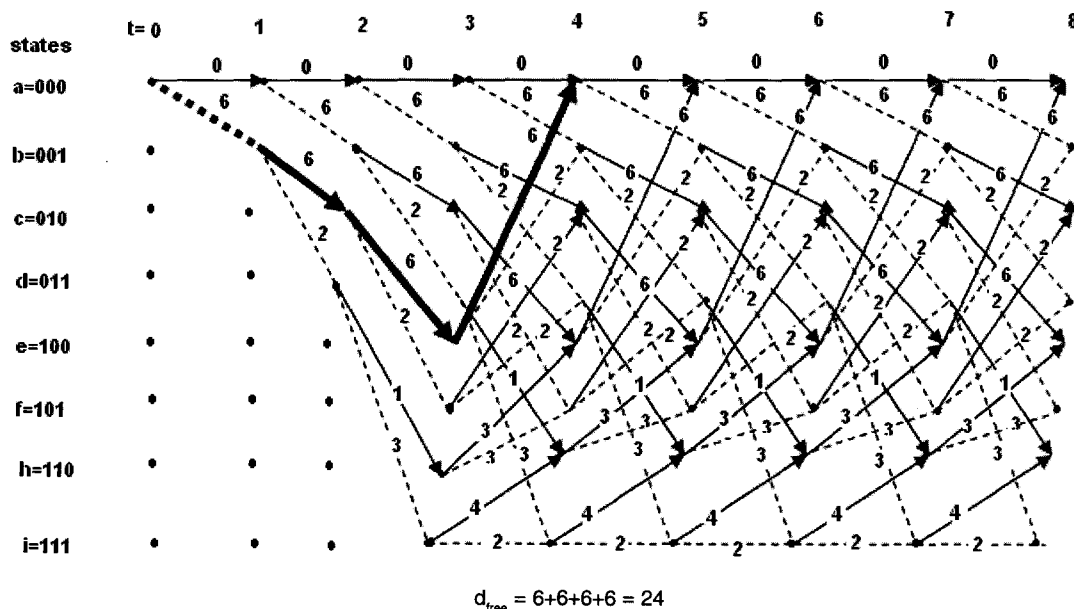


Fig. 5—Trellis diagram for non-polynomial based without repetition of branch word ($d_{free}=24$)

cost paths. Survivor memory stores lowest cost bit-sequence paths based on decisions made by ACS units. Hardware implementation of algorithm demonstrates benefit of Viterbi algorithm. This architecture takes advantage of parallelization and specialization of hardware for specific constraint length and decoder hardware to changing channel noise characteristics (Fig. 6).

Branch Metric Unit (BMU)

Branch metrics are difference values between received code symbol and corresponding modified BWs from encoder trellis. Encoder BWs are code symbols that would be expected to come from encoder output as a result of the state transitions. Hamming distance $d(X, Y)$ between X and Y is defined as equal the number of differing elements. Maximum value of hamming distance corresponds to the number of bits in symbol. Minimum value is zero, when received symbol and encoder symbol are equal. BMG determines difference between received n -bit (6) value and 2^n possible expected values. A total of $2^{2(K-1)}$ branch metrics has determined by BMG.

Add-Compare-Select (ACS) unit

Critical-path of a traditional ACS computation extends through sequential execution of two parallel additions, a comparison and a selection. Let $smi(n)$ represent path metric for state i , and $bmij(n)$, branch metric of a corresponding transition from state i to state

j , with time step denoted by n . Then, an example of ACS recursion corresponding to state 0 is shown as

$$s_0(n+1) = \min \begin{cases} s_0(n) + b_{00}(n) \\ s_0(n) + b_{10}(n) \end{cases} \quad \dots(15)$$

Structure of ACS unit consists of two-input adders and a multiplexer. A comparison between two sums of branches is followed by selection of appropriate minimum value, and finally, addition of two corresponding branch metrics and structure of ACS unit (Fig. 7).

Trace Back Unit

Trace back is area efficient and is better than REA (Register Exchange Array). Storage can be implemented as RAM and is called path memory. After at least 1 branches have been processed, trellis connections are recalled in reverse order and path is traced back through trellis diagram. Since 1 bits must be traced back before bits the path can be considered reliable as output bits, this technique requires more than one branch to be traced back for every time unit. For high-speed decoder, implement a number of parallel processes. Once state update is done, path memory for all states must be updated. Path memory is the set of sequences of decision bits that lead to each state at current time. Path for each new state is the path of its predecessor state with the decision bit for new state appended to its end. Rather

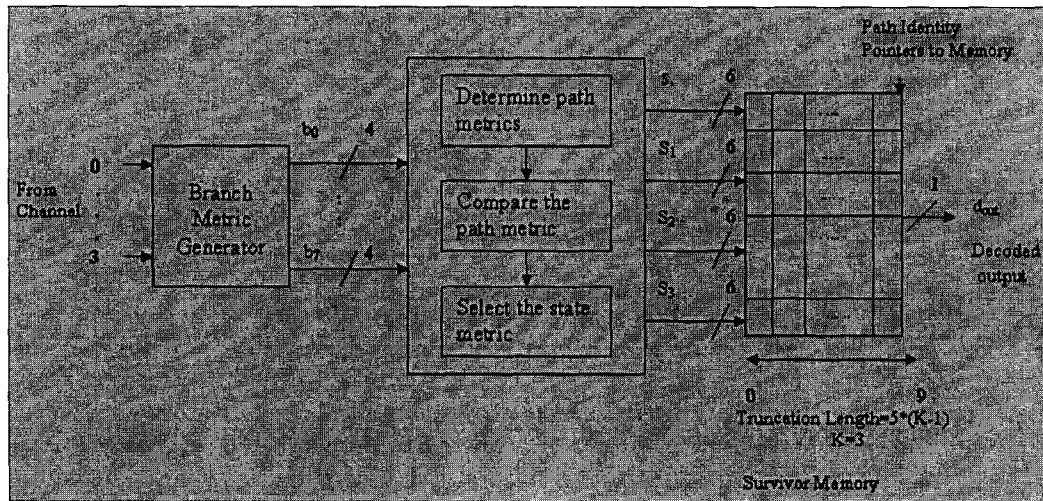


Fig. 6—Architecture of Viterbi decoder based on non-polynomial approach

than moving entire paths from one memory location to another, a more efficient technique is to store all decisions for all states into a RAM as decisions are made. RAM is organized as a rectangular page where each column contains all decisions made for a given bit time, and moving back one column along a row is equivalent to backing up a bit time. State forms row address and bit time forms column address.

A set of ACS operations is completed. Resulting decisions are written into appropriate column of RAM. At the end of a state update cycle, path memory contains all decisions from current bit time as well as all decisions from previous N bit times. To determine path that led to a given state, as the row address and the current bit time as the column address. Left shifting the old row address and adding the bit just read as the LSB then forms a new row address. Column address is decremented and another bit is read. This is repeated until the trace back has gone far back enough so that the path can be assumed to have converged. At that point the data bits on the path are output as corrected data, but in reverse order this data is stored in last in first out (LIFO) stack, which outputs data in correct sequence.

Path converges most rapidly if trace back starts at the state with best (lowest) metric. Usually, trace back is limited to about 4 to 10 constraint lengths, at which point degradation from a theoretical design is only one or two tenths of a dB. Higher code rates generally require more path memory depth than low rates. One decoded bit can be carried out from each trace back cycle. Fewer operations are required to output multiple bits per trace

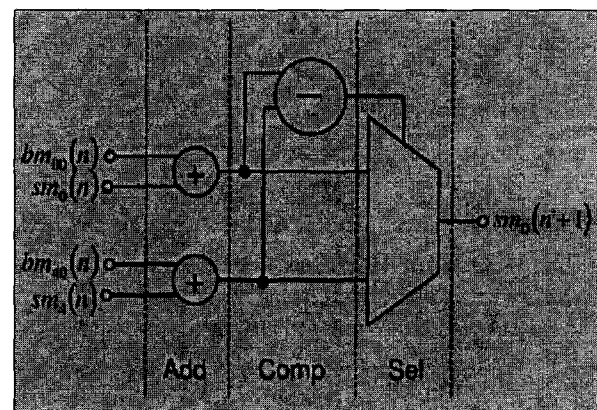
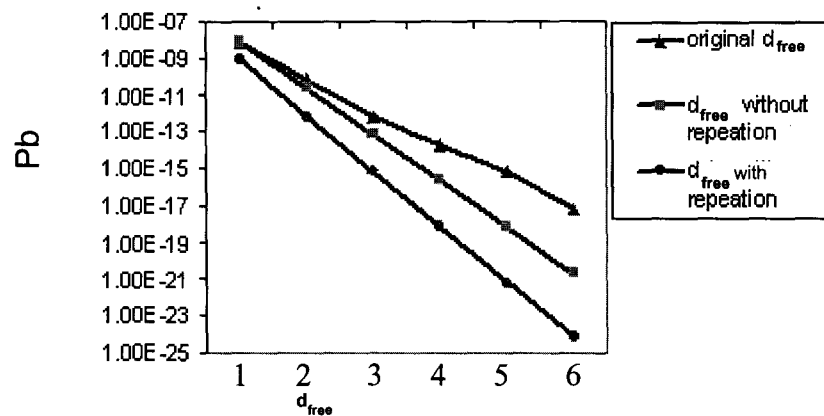


Fig. 7—Internal Structure of ACS

back, but performance is slightly worse than recovering each bit from a path traced all the way to end of the path memory.

Results

To validate non-generating polynomial Viterbi decoder and evaluate its decoding performance in terms of probability of error, decoding delay, speed and resource usage, by using VHDL language, a synthesizable core of decoder has been developed and implemented on Xilinx VERTEX-E FPGA device. Performance measures for channel decoding are low BER and bandwidth efficiency, which is determined by code word chosen and type of modulation used. SNR has determined BER performance. Constraint length and d_{free} values have improved SNR of Viterbi decoder. In non-polynomial Viterbi decoder, d_{free} is higher than

Fig. 8—Free distance (d_{free}) versus probability of error (P_b)Table 2—Comparison of d_{free} and P_b for polynomial and non-polynomial approach with constant code rate 1/6

Constraint length K	Conventional polynomial method		Proposed non-polynomial method without repetition of branch word		Proposed non-polynomial method with repetition of branch word	
	d_{free}	P_b	d_{free}	P_b	d_{free}	P_b
3	16	7.06E-09	16	7.06E-09	18	7.16E-10
4	20	6.87E-11	21	2.25E-11	24	6.90E-13
5	24	6.60E-13	26	6.90E-14	30	6.95E-16
6	27	2.00E-14	31	2.12E-16	36	6.85E-19
7	30	6.90E-16	36	6.65E-19	42	7.36E-22
8	34	6.62E-18	41	2.16E-21	48	7.20E-25

Table 3—Comparison of d_{free} for polynomial and non-polynomial approach with constant constraint length 4

Code rate r	Conventional polynomial method d_{free}	Proposed non-polynomial method without repetition of branch word, d_{free}	Proposed non-polynomial method with repetition of branch word, d_{free}
1/2	6	5	8
1/3	10	9	12
1/4	13	13	16
1/5	16	17	20
1/6	20	21	24
1/7	23	25	28
1/8	26	29	32

conventional polynomial approach. Increase in d_{free} tends to increase SNR and thus improves BER. Here, a BER of 6.90×10^{-13} is achieved for a constraint length of 4, code rate 1/6 and d_{free} of 24. One can also demonstrate overall decoding rate of 60.299 Mbps by reconfiguring decoder unit. This performance of Viterbi decoder system in terms of probability of error based on non-polynomial approach has been improved to 75 % for detecting and correcting error than reported results¹³.

For a constant code rate (1/6) and for different

constraint lengths, performance is better when BWs are repeated since free distance gets increased compared to other two methods (Table 2). For a constant constraint length ($K=4$) and for different code rates, performance is better when BWs are repeated since free distance gets increased compared to other two methods (Table 3). For a constant code rate (1/6) and for different number of states, coding gain for original free distance and increased free distance with repetition and without repetition of BWs are shown in Table 4. Coding gain

Table 4—Code gain comparison for proposed methods

Constraint length	Number of states	Conventional method coding gain	Proposed method code gain without repetition	Proposed method code gain with repetition
3	4	4.25	4.25	4.7
4	8	5.22	5.44	6.02
5	16	6.02	6.36	6.98
6	32	6.53	7.13	7.78
7	64	6.98	7.78	8.45
8	128	7.53	8.34	9.03

Table 5—Comparison of P_b with different E_b/N_0 and constant code rate $r = 1/6$

E_b/N_0 (Signal to noise ratio)	Conventional polynomial method P_b	Proposed non-polynomial method without repetition of branch word P_b	Proposed non-polynomial method with repetition of branch word P_b
4.5	7.34E-6	3.34E-6	3.22E-7
5.5	2.47E-8	9.64E-9	5.76E-10
6.5	2.35E-10	7.88E-11	2.77E-12
7.5	4.34E-12	1.26E-12	2.78E-14
8.5	1.03E-13	2.3E-14	3.25E-16

can be calculated using Eq. 10 for all three methods (Table 4).

Code gain increases when number of states increases. Also, when BWs are repeated, code gain is maximum. For a constant code rate (1/6) and for different E_b/N_0 ratio, probability of error for original free distance and increased free distance with repetition and without repetition of BWs are shown in Table 5. E_b/N_0 (SNR) increases with P_b decreasing. Also, when BWs are repeated, P_b gets decreased compared to other two methods.

Conclusions

A novel Viterbi algorithm was designed and implemented using non polynomial approach at Xilinx VERTEX-E FPGA for achieving low bit error rate and high throughput rate. Larger d_{free} have chosen to achieve the best performance of the system. An attempt has been made to study the effect of d_{free} on constraint length and code rate. Proposed method gives maximum throughput at encoder section (423.93 Mbps) and decoder section (60.299 Mbps) of architecture. Probability of error for constraint length 4 and code rate 1/6 is 6.90×10^{-13} . Overall performance of Viterbi decoder architecture for proposed

method is 75% than existing value of 25%. Proposed Viterbi decoder provides satisfactory probability of error (P_b) performance and high operating speed under AWGN, co-channel interference and adjacent channel interference environments.

References

- 1 Chan F & Haccoun D, Adaptive Viterbi decoding of convolutional codes over memory less channels, *IEEE Trans Commun*, **45** (1997), 1389-1400.
- 2 Tessier R & Burleson W, Reconfigurable computing and digital signal processing: A survey, *J VLSI Signal Process*, **28** (2001) 7-27.
- 3 Seki K, Kubota S, Mizoguchi M & Kato S, Very low power consumption Viterbi decoder LSIC employing SST (scarce state transition) scheme for multimedia mobile communications, *IEEE Electron Lett*, **30** (1994) 637-639.
- 4 Lang L, C Y Tsui & R S Cheng, Low power soft output viterbi decoder scheme for turbo code decoding, in *Proc IEEE Int Symp Circuits and Systems*, ISCAS-97 (Hong Kong) 1997, 1369-1372.
- 5 Kang I & Willson A N, Low-power Viterbi decoder for CDMA mobile terminals, *Solid-state circuits, IEEE J*, **3** (1998) 473-482.
- 6 Lin C C, Wu C C & Lee C Y, A low power and high speed Viterbi decoder chip for WLAN application," in *Proc IEEE Int European Solid State Circuits Conf, ESSCIRC'03* (Estoril, Portugal) 2003, 723-726.

- 7 Shirley A, Shahriar M, Lutz L & Robert S, Per-survivor processing Viterbi decoder for Bluetooth applications, in *IEEE Int Symp on Circuits and Systems (ISCAS 2006)* (Island of Kos, Greece) (2006) 2601-2604.
- 8 B S I G (SIG), "Bluetooth standard core version 2.0 + EDR," November 2004.
- 9 Lampe L, R Schober & M. Jain, Noncoherent sequence detection for Bluetooth systems, *IEEE J Select Areas Commun*, **23** (2005) 1718-1727.
- 10 Gemmeke T, Gansen M & Noll T, Implementation of scalable power and area efficient high-throughput Viterbi decoders, *IEEE J Solid-State Circuits*, **37** (2002) 941-948.
- 11 Daut D, Modestino J & Wismer L, New short constraint length convolutional code constructions for selected rational rates, *IEEE Trans Inform Theory*, **28** (1982) 794-801.
- 12 Proakis J, Digital Communications (McGraw-Hill, New York, NY) 1995, 340-500.
- 13 Boo M, Arguello F, Bruguera J D, Doallo R, Zapata & Javier E L, High performance VLSI architecture for the Viterbi algorithm, *IEEE Trans Commun*, **45** (1997) 168-176.
- 14 Chadha K & Cavallaro J R, A reconfigurable Viterbi decoder architecture, in *Proc 35th Asilomar Conf on Signals, Systems and Computers* (Pacific Grove, California USA) 2001, 66-71.
- 15 Engling Y, Augsburger S A, Davis W R Z & Nikolic B, A 500Mb/s soft output Viterbi decoder, *IEEE J Solid State Circuits*, **38** (2003) 1234-1241.
- 16 Gang Y, Arshlan T & Erdogan A T, An efficient pre-traceback approach for Viterbi decoders, *IEEE Trans Commun*, **6** (2005) 5441-5444.
- 17 Kinniment D J, Bystrov A & Yakovlev A V, Synchronization circuit performance, *IEEE J Solid-State Circuits*, **37** (2002) 202-209.
- 18 Lee C, A Viterbi decoder with efficient memory management, *ETRI J*, **26** (2004) 21-26.
- 19 Lin M B, New path history management circuits for Viterbi decoders, *IEEE Trans Commun*, **48** (2000) 1605-1608.
- 20 Parhi K K, Suzuki H H & Chang Y-N, A 2-Mb/s 256-state 10-mW rate-1/3 Viterbi decoder, *IEEE J Solid-State Circuits*, **35** (2000) 826-834.
- 21 Parhi K K, An improved pipelined MSB-first add compare select unit structure for Viterbi decoders, *IEEE Trans Circuits and Systems I: Fundamental Theory and Applications*, **51** (2004) 504-511.