Asian Journal of Information Technology 15 (7): 1232-1242, 2016
ISSN: 1682-3915

Design and FPGA Implementation of Bit Level Pipelined Digit Serial VLSI Architecture for a Viterbi Decoder

T. Kalavathidevi and P. Sakthivel
Department of EIE, Kongu Engineering College, Perundurai, Tamil Nadu, India
Department of EEE, Velalar College of Engineering and Technology, Erode, Tamil Nadu, India

Abstract: This study addresses a systematic unfolding transformation technique to transform the conventional viterbi architecture to equivalent digit serial. The originality of the unfolding technique lies in the generation of functionally correct control circuits in digit serial architectures. Convolutional code is an essential Forward Error Correcting (FEC) code for many wireless communication systems. Viterbi decoder is an optimal algorithm for decoding a convolution code. Power dissipation is recognized as a critical parameter in modern Very Large Scale Integrated circuit (VLSI) design field. Viterbi decoder employed in digital wireless communication is complex and dissipates large power. The aim of the proposed method is to obtain high speed and low power Viterbi decoder using bit-level pipelined digit-serial architecture for various digit size and word length. In the digit-serial architecture N bits are processed per clock cycle and a word is processed per W/N clock cycles (W: word length, N: digit size). Bit-level pipelining technique is applied for each bit as well as for each block. Digit serial architecture and bit-level pipelining achieves high speed and low power. With this technique the viterbi decoder is designed for word length W = 8, 16, 32 and digit size N = 2, 4. The functionality is simulated and synthesized using Xilinx ISE 13.2i.

Key words: Power dissipation, digit size, Viterbi decoder, unfolding, bit level pipelining

INTRODUCTION

The Viterbi algorithm proposed by Viterbi (Forney, 1972; Forney, 1973) is widely used in digital communication. This method is efficient for the realization of maximum likelihood decoding of convolutional codes. The convolutional code (Dholakia, 1994) finds its impact in encoding, error correction and decoding applications.

Unfolding is a transformation technique that can be applied to a DSP blocks like adders, multipliers, FIR filters and in dynamic programming algorithms like Viterbi algorithm (Parhi, 1989; Parhi, 1987; Keshab and Parthi, 1999) to create a new blocks describing more than one iteration of the original blocks. More specifically, the unfolding a DSP program is done by the unfolding factor J which describes J consecutive iteration of the original program.

Unfolding algorithm can be applied for bit serial, digit-serial and word parallel processing architectures. In the bit-serial architecture, one bit is processed per clock cycle and a complete word is processed in W clock cycles. The speed of the Viterbi decoder is reduced and also it consumes more power. Figure 1 represents the bit-serial architecture in which a5, a4, a3, a2, a1, a0 are the inputs and b5, b4, b3, b2, b1, b0 are the outputs, a0 is processed in the first clock cycle to produce output b0, so each bit needs one clock cycle. Hence, totally six clock cycles are needed.

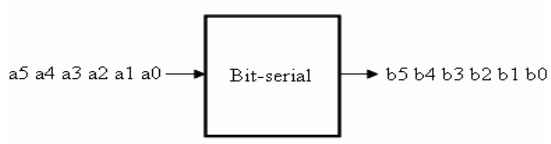

Fig. 1: Bit serial architecture

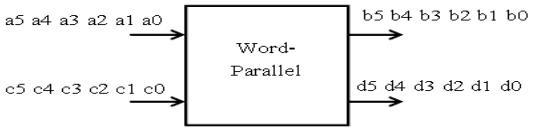

Fig. 2: Word parallel architecture

A direct application of unfolding transformation is to design parallel processing architecture from serial process architecture. At word level, that is the word-parallel structure can be designed from word-serial architecture. Figure 2 represents the word-parallel architecture in which a5, a4, a3, a2, a1, a0 and c5, c4, c3, c2, c1, c0 are the inputs and b5, b4, b3,

Corresponding Author: T. Kalavathidevi, Department of EIE, Kongu Engineering College, Perundurai, Tamil Nadu, India

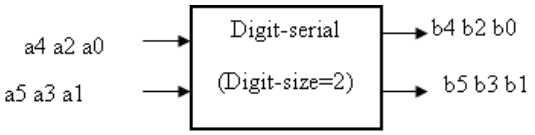

Fig. 3: Digit-serial architecture

b2, b1, b0 and d5, d4, d3, d2, d1, d0 are the outputs. Word parallel architecture processes J words per clock cycle.

When compared to the bit serial and word parallel design, the digit serial architecture needs only fewer latches in adder computation portions. Data conversion formats in digit-serial require only simple control circuits. For example a binary adder requires one third portion of latches for normal operations which can be reduced by digit-serial architectures. Digit serial systems are ideal for moderate speed applications.

Digit-serial architecture reduces the clock cycle; thereby it increases the speed and reduces power consumption. Figure 3 represents the digit-serial architecture in which a5, a4, a3, a2, a1, a0 are the inputs and b5, b4, b3, b2, b1, b0 are the outputs,a0 and a1 are processed gives outputs b0 and b1 in one clock cycle. Thus 2 bits are processed in one clock cycle, hence three clock cycles are needed to process 6 bits.

The research activities in the Viterbi decoder are categorized in two groups. The first group is based on altering the decoder architecture to obtain desired metrics while the second group utilized different circuit implementation techniques to optimize the related metric. The FPGA implementation of Viterbi decoders (Haene *et al*., 2004) for Multiple-Input Multiple-Output (MIMO) wireless communication systems with Bit-Interleaved Coded Modulation (BICM) and per-antenna coding is considered. The study describes how the recursive Add-Compare-Select (ACS) unit which constitutes the performance bottleneck of the circuit, can be pipelined to increase the throughput. As opposed to employing multiple parallel decoders, silicon area (resource utilization on the FPGA) is significantly reduced.

The design supports a generic, ro-bust and configurable Viterbi decoder (Sun and Ding, 2012) with constraint length of 7, code rate of 1/2 and decoding depth of 36 symbols. Relevant simulation results using Verilog HDL language are verified based on a Xinlinx Virtex-II FPGA by ISE 7.1i. It is shown that the Viterbi decoder is capable of decoding (2, 1, 7) convolutional codes accurately with a throughput of 80 Mbps.

Retiming, pipelining and parallel to serial conversion techniques are used in the viterbi decoder to reduce area and optimize speed (Kim *et al*., 2009). The serial implementation of the viterbi decoder is pipelined and retimed to decrease critical path delay of the circuit and thus increases the throughput.

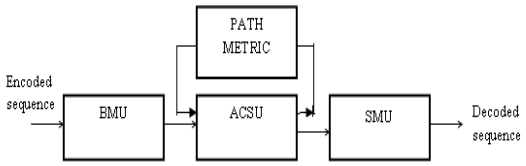

Fig. 4: Blocksof viterbi decoder

A practical method to design a parallel processing viterbi decoder was explored (Karim *et al*., 2011). The trace back decoder process parallel in order to improve the processing speed. Parallel processing reduces the processing time and increases the memory space. FPGA implementation is more advisable for parallel processing method to reduce the computation time delay. Trace back method does not hold good for low power circuits.

The shift-and-add multiplier generates the partial products sequentially and accumulates them successively as they are generated. This shift-accumulation approach was used to implement a digit-serial multiplier (Aksoy *et al*., 2011). The Digit-Serial Shift-and-Add Multiplier (DSAAM) contains two feedback loops the carry recursion in the accumulator and the carry loop in the carry propagating adder. The longest logic path of these two loops determines T_{pd}. This method reduces clock cycle and increase speed in a multiplier.

The conventional method takes (2, 1, 5) convolutional code (Zhang and Parhi, 2011) as an example and adopt a hard decision parallel viterbi decoding algorithm and implemented in Very High Speed Integrated Circuit Hardware Description Language (VHDL). Truncated decoding register exchange method was used which reduces the delay time and improves resource utilization.

Two bit-level pipelined ACS unit in Viterbi decoder to reduce the critical path based on 2-step look-ahead technique was addressed (Goo and Lee, 2008). This step look ahead technique has the problem of long latency which increases linearly with the number of states in Viterbi decoder. Thus, in order to reduce latency and to increase the speed of the Viterbi decoder, repeated iterations is performed at the same clock transition by unfolding algorithm. High speed with reduced power for the Viterbi decoder is achieved by digit serial technique (Landernas *et al*., 2004) which permits bit level pipelining.

Viterbi decoder: Viterbi decoder uses the Viterbi algorithm for decoding a bit stream that has been encoded using FEC based on a convolutional code. Trellis diagram is used to decode the input sequence. The Viterbi decoder is composed of three functional units which are given in Fig. 4:

- Branch Metric Unit (BMU): The BMU computes branch metrics by Hamming distance or Euclidean distance.
- Add Compare and Select Unit (ACSU): It selects a best trellis path based on current Branch Metric (BM) and previous state metric.
- Survivor Memory Unit (SMU): It generates the decoded bits by the best state sequence in the trellis.

MATERIALS AND METHODS

Block diagram of bit-level pipelined digit-serial architecture: Block diagram of proposed bit-level pipelined digit-serial architecture Viterbi decoder is shown in Figure 5. Branch Metric Unit calculates the hamming distance between the received and expected code length. Path Metric calculates the minimum path which requires addition, comparison and selection operations commonly called as ACS unit. Latch is included between adder and comparator, selector block. SMU unit stores the value of the received sequence of the minimum path metric.

Pipelining is a low power technique to reduce the power and latency. When a design is pipelined at M-level, then the critical path of the design is reduced to 1/M of its original length. M refers to the number of pipeline stages. Bit-level pipelining of the Viterbi decoder depends on the number of latches connected at the inputs and at the critical paths.

In bit level pipelining, input is given to the register in a bit by bit fashion, which is based on the clock. Hence four registers are placed at the four critical paths in the architecture. Representation of latch or register refers to the same. The registers are represented as REG1, REG2, REG3 and REG4. Shift registers are used to shift the content of the SMU. Input and BMU are latched by REG1. The BMU output is given to the adder of the ACS unit by REG2. In ACS unit REG3 is included between the adder and comparator. REG4 output is given to the SMU. In the programming concept, each register is associated with a clock, which is viewed in the Register Transfer Level (RTL) design. All the four clocks clkf, clks, clkm and clkr are synchronized by a common clock signal named clk, in such a way that the operations are performed during the positive and negative transitions as calculated by the switching instances. In bit-level pipelining clocking is done by edge triggered single phase clocking.

Branch metric unit: The BMU computes branch metrics by Hamming distance. The branch metrics are calculated by comparing the received code symbol and the expected code symbol and count the number of differing bits. The branch metric unit consists of XOR gate and the ones counter is represented in Fig. 6. The

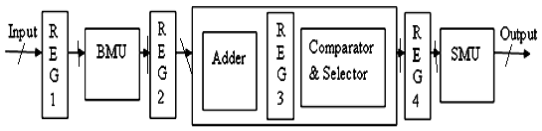

Fig. 5: Block diagram of proposed bit-level pipelined digit-serial architecture

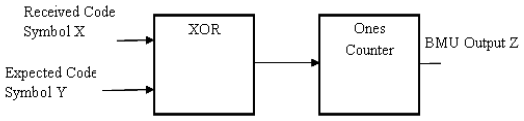

Fig. 6: General block diagram of branch metric unit

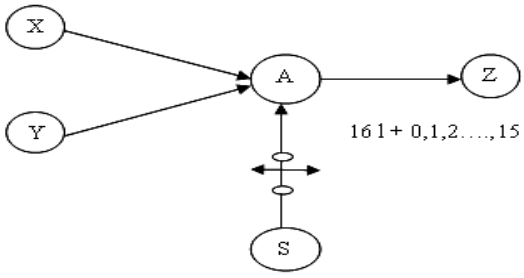

Fig. 7: DFG of XOR gate for word length W = 16

XOR gate compares the received code symbol with the expected code symbol and the counter count number of differing bits.

Data flow graph of XOR gate: The XOR gate in the branch metric unit is designed using digit-serial architecture using unfolding application. In order to explain the unfolding technique Data Flow Graph (DFG) of XOR gate is drawn. In the DFG, the nodes represent computations and the directed edges represent data paths and each edge has a nonnegative number of delays associated with it. Figure 7 represent the DFG of XOR gate. X and Y are the inputs and Y is the output of the XOR gate. The S is the switching instance ranges from 0, 1,..., 15.

Unfolded DFG of XOR gate: The DFG of XOR gate is unfolded by a factor J. For designing the unfolded DFG of XOR gate, we must consider switch. The process of unfolding edges with switches is first described and then applied to design digit-serial architecture from bit-serial architecture. Consider the edge U→V in switch shown in Fig. 8.To unfold this edge with unfolding factor J, two basic assumptions are made:

The word length W is a multiple of the unfolding factor J, i.e., W = W'J. All edges into and out of the

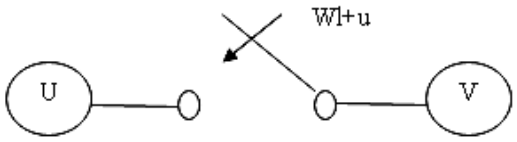

Fig. 8: Representation of a switch

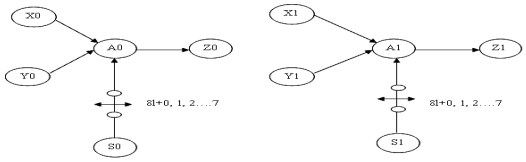

Fig. 9: Unfolded DFG of XOR Gate for W = 16 and unfolding factor J = 2

switch have no delays. With these two assumptions the edge can be unfolded using the following two steps. Write the switching instance as in Eq. 1:

$$wl + u = J\left(w'l + \lfloor u / j \rfloor\right) + (u\%J) \quad (1)$$

Draw an edge with no delays in the unfolded graph from the node $U_{u\%J}$ to the node $V_{u\%J}$ which is switched at time instance. The DFG of XOR gate is unfolded using above assumptions for word length W = 16 and unfolding factor J = 2. From assumptions W = W'J, W' = 16/2 = 8 and the edge has no delays. The switching instance is calculated using Eq. 1. First switching instance calculated as:

W= 16, u= 7, J= 2, W'= 8

16l+7 = 2(8l+ 7/2) + 7%2

16l+7 = 2(8l+3) + 1

Remaining switching instances16+0, 16l+1, 16l+2, ..., 16l+15 are calculated using above procedure. Sample of switching instances are given as follows:

16l+0 = 2(8l+0) + 0, 16l+8 = 2(8l+4) + 0

16l+1 = 2(8l+0) + 1, 16l+9 = 2(8l+4) + 1

Similarly, the calculation of switching instances 32l+0, 32l+1, 32l+2, ..., 32l+31 are done using the above procedure for W=32... A few switching instances are represented as:

32l+0 = 4(8l+0) + 0 32l+8 = 4(8l+2) + 0,

32l+16 = 4(8l+4) + 0 32l+24 = 4(8l+6) + 0,

32l+1 = 4(8l+0) + 1 32l+9 = 4(8l+2) + 1,

32l+17 = 4(8l+4) + 1 32l+25 = 4(8l+6) + 1&

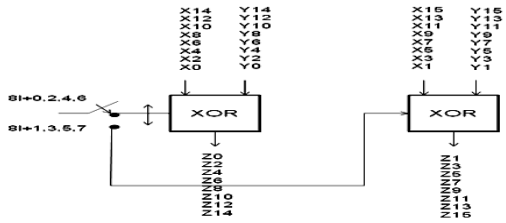

Fig. 10: Digit serial architecture of XOR gate

Table 1: Clock pulse required to produce output for various word length and digit size

Word length	Digit-size	Number of clock pulses
8	2	4
16	2	8
32	2	16
8	4	2
16	4	4
32	4	8

In the unfolded DFG, the edges are calculated using unfolding algorithm. Figure 9 shows the unfolded DFG of XOR gate. X and Y are the inputs and Y is the output. The type of operation to be performed on x,y at the switching instances So is given by A0..A1, etc.

Digit-serial architecture of XOR gate: For unfolding factor J = 2, word length W = 16 and digit size N = 2 the digit- serial architecture for XOR gate is shown in Fig. 10. The X and Y are the 16 bit inputs and Z is the output. First clock cycle Z_0, Z_1 obtained, second clock cycle Z_2, Z_3 obtained like this so on the entire output obtained in the eighth clock cycle.

The output z_0, z_1 is obtained at the switching instance 8l+0, z_2, z_3 at the switching instance of 8l+1, z_4, z_5 at the switching instance of 8l+2 and z_6, z_7 at the switching instance of 8l+3. So, within the eight cycles the sixteen outputs are obtained but in normal XOR gate require 16 cycles for 16 outputs. BMU unit is implemented in VHDL. To ensure the switching transition clock signal is given for each module. For the BMU operation Clkx is assigned. During the negative transition of the clock pulse the transitions 1, 3, 5, 7 occur. At the positive edge 0, 2, 4, 6 occur. The counter counts the number of one's from the output of the XOR gate at each switching instance. Number of difference in the input bits is counted by the counter. A Variable j is assigned for counter operation. This variable gets incremented for every difference in transition. End of the fourth clock cycles the number of ones at the output of the exor gate is obtained. Thus, the output of the BMU unit is given to the adder of the ACS unit. The switching instants and clock pulse required are

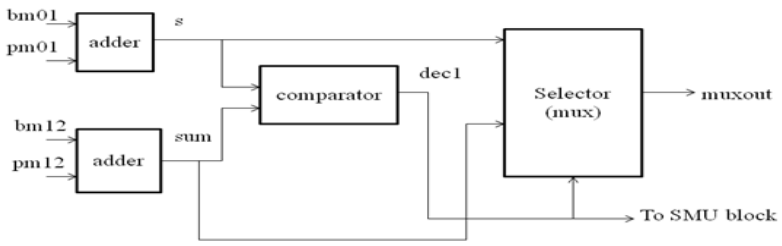

Fig. 11: Block diagram of ACS unit

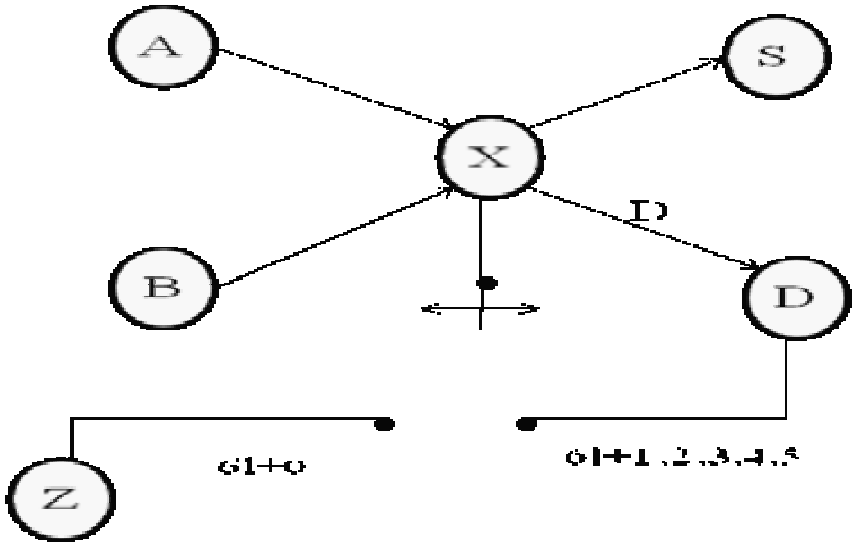

Fig. 12: DFG of full adder for W = 6

given in Table 1. It reflects that the number of clock pulses get reduced when the word length increases.

Add-compare-select unit: The second step in the Viterbi decoding algorithm is the ACS unit which is the heart of the process and dictates the performance of the decoder. Figure 11 shows the block diagram of ACS unit of the Viterbi Decoder. The two adders compute the partial path metric (s, sum) of each branch, the comparator compares the two partial path metrics produce decision (dec1) and the selector selects (muxout) an appropriate branch. The new path metric updates the state metric of state and decision is given to SMU block.

Data flow graph of full adder: In ACS unit the Full adder is used for adder section which is designed using digit-serial technique with Unfolding algorithm. Data Flow Graph of full adder is shown in Fig. 12. A, B are inputs, S represents output, dummy node D is between X and switch due to delay between them, Z is initial carry (i.e., 0), 6l+0, 1, 2, 3, 4, 5 are switching instances.

Unfolded DFG of full adder: Unfolded DFG of full adder is constructed using unfolding algorithm shown in Fig. 13. Switching instance for the full adder is calculated with the formula: Word length W = 6, unfolded factor J = 2 W' = 3 u = 0-5:

$$6l+0 = 2(3l+0)+0,\ 6l+3 = 2(3l+1)+1$$
$$6l+1 = 2(3l+0)+1,\ 6l+4 = 2(3l+2)+0$$
$$6l+2 = 2(3l+1)+0,\ 6l+5 = 2(3l+2)+1$$

Switching instance for W = 6, u = 3, J = 2, W' = 3 is calculated as follows:

$$W = 6,\ u = 3,\ J = 2,\ W' = 3$$
$$6l+3 = 2(3l+3/2) + 3\%4$$
$$6l+3 = 2(3l+1) + 1$$

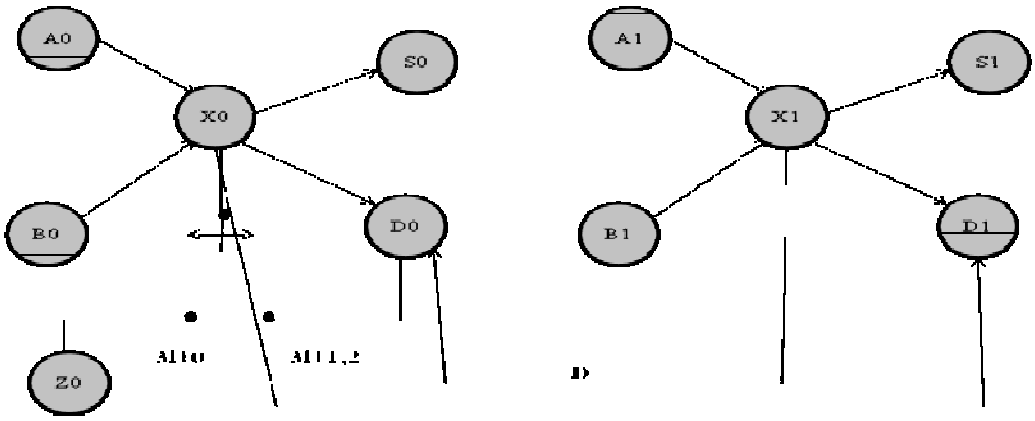

Fig. 13: Unfolded DFG of full adder for W = 6, J = 2

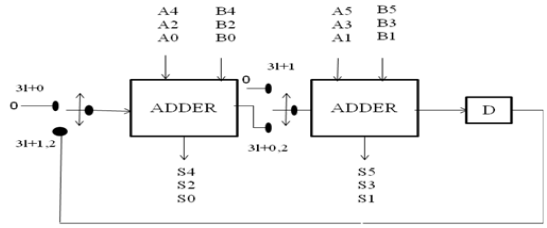

Fig. 14: Digit serial architecture of full adder for W = 6, N = 2

Table 2: Inputs and outputs at the corresponding switching instance of full adder

Switching Instance	Input	Output
3l+0	A0,B0,A1,B1	S0,S1
3l+1	A2,B2,A3,B3	S2,S3
3l+2	A4,B4,A5,B5	S4,S5

Table 2 shows inputs and outputs at the corresponding switching instance of full adder. First switching instance 6l+0 in which inputs A0, B0, A1, B1 are processed produce output S0, S1. In the third switching instance entire outputs S0, S1, S2, S3, S4 are obtained.

Digit-serial architecture of full adder: For unfolding factor J = 2, word length W = 6 and digit size N = 2 the digit-serial architecture of full adder is shown in Fig. 14. A and B are the 6 bit inputs and S is the output. In the first clock cycle S0, S1 are obtained, in the second clock cycle S2, S3 are obtained and in the third cycle S4, S5 are obtained. This digit-serial processing is applied to both adders (upper and lower branch adder) in this ACS unit of the Viterbi decoder. In Fig. 14 A, B represents current branch metric (bm), previous path metric (pm) and S is partial path metric.

In the third switching instance, the output of the adder is given to the comparator which compares the two adder output produce decision. The comparator output is given to the survivor path recording block. Simultaneously, the adder output is given to the selector which selects the shortest path metric value between the outputs of the two adders. In this method the 2:1 multiplexer is used for selector, the control signal of the selector is obtained from the decision of the comparator.

Survivor memory unit: The third step in the Viterbi decoding is the Survivor Memory Unit. Three approaches are often used to record survivor branches, Trace Back (TB), Register Exchange (RE) and Modified Register Exchange (Dib and Elmasry, 2004) Method (MREM). The RE approach assigns a register to each state. The register records the decoded output sequence along the path starting from the initial state to the final state. The RE technique is acceptable for trellises with only a small number of states whereas the TB approach is acceptable for trellises with a large number of states. Therefore, the TB Method has been widely investigated and implemented. The drawback in trace back is that all the paths of the states have to be traced either forward or backward which involve more transitions and switching activity. On considering the VLSI implementation RE Method is better. RE Method is updated as MREM. In MREM, there is no need for checking all the paths. Here the shift registers for the minimum value input alone is maintained to be active during the entire process of operation. The SMU was designed as 4x4 shift register using D-flip flop. The length of the shift register depends on the length of the convolution encoder. In the SMU for a constraint length of K = 3, there will be 2^{k-1} shift registers for each state.

Table 3: Comparison of trace back, register exchange and modified register exchange method

Parameters	Trace back (existing)	Register exchange (existing)	Modified register exchange (proposed)
Net switching power	72.0345 µW	40.52 µW	4.5761 µW
Total dynamic power	209.7302 µW	180.23	4.5761 µW
Combinational area	181.7500	150	77.00
Non combinational area	176.00	125	133.00
Total cell area	357.7500	275	210.000

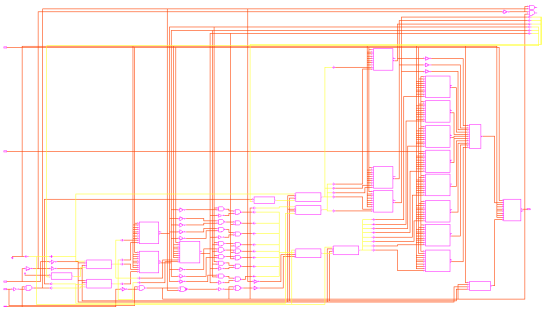

Fig. 15: Register transfer level of modified register ex-change method

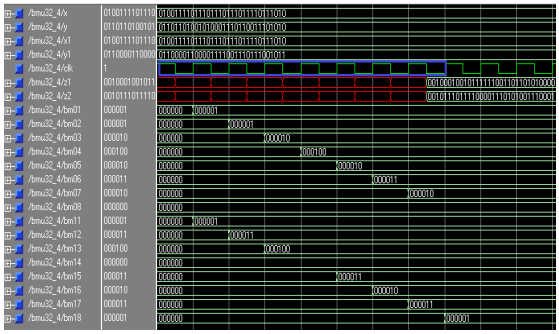

Fig. 16: Branch metric unit output waveform for W = 32, N = 4

Performance comparison of the three methods of SMU in Table 3 shows that the Modified Register Exchange Method permits the design of compact memory unit with significant area and power reduction. Figure 15 shows the Register Transfer Level of Modified Register Ex-change (MRE) Method. It depicts that the interconnection of various gates and registers involved in SMU architecture.

RESULTS AND DISCUSSION

Bit level pipelined digit serial viterbi decoder is designed using VHDL and implementation of the design is done in Virtex. The architecture is verified for code rate of ½, ¼ for various constraint lengths. Bit level pipelining of the Viterbi decoder depends on the number of digit size. Registers are placed at the four critical paths in the architecture. Here latch or register refer the same. The registers are represented as reg 1, 2, 3 and shift registers to shift content of the SMU. In the VHDL programming, for each registers a clock is associated to it. Reg1 is placed between the input and BMU. Reg 2 is placed between the BMU and ACS unit. The Reg3 is between Adder and comparator. All the four clocks are synchronized in such a way that the operations are performed during the positive and negative transitions as calculated by the switching instances (Zengliang and Cheng, 2011). When implementing the digit serial Viterbi decoders with bit level pipelining increases the system throughput with reduced power consumption. In bit level pipelining throughput depends on the configurable logic cells, register computation delay, register clock time and interconnection between adjacent registers. Based on these parameters the system throughput is obtained from the synthesis results.

Digit-serial branch metric unit output waveform for W = 32, N = 4: Figure 16 shows Branch Metric Output waveform for W = 32, N = 4. x = x1 = 0100111101110 1110111011110111010 are represented as received sequence and y = 0110110100101000111011001110 1010, y1 = 0110000110000111100111011100 1011 are represented as expected sequence. The output of xor gate are represented as z1 = 0010001001011111100110110101 0000, z2 = 0010111011110000111010100111 0001. Each four bits of z1, z2 are obtained in single clock cycle; hence eight clock cycles are required to process 32 bits. Counter output of BMU are represented as bm01 = 000001, bm02 = 000001, bm03 = 000010, bm04 = 000100, bm05 = 000010, bm06 = 000011, bm07 = 000010, bm08 = 000000, bm11 = 000001, bm12 = 000011, bm13 = 000100, bm14 = 000000, bm15 = 000011, bm16 = 000010, bm17 = 000011, bm18 = 000001.

Add compare select unit

Digit-serial architecture of acs for W = 4, N = 2: Figure 17 shows single state ACSU output waveform for W = 4, N = 2. The current upper and lower branch metric are represented as bm01 = 0100, bm12 = 0011. Previous path metric values are pm01 = 0101, pm12 = 0001 and partial path metric are s = 01001, sum = 00100. First two bits of s and sum are obtained in first clock cycle and next three bits are obtained in second clock cycle. Comparator

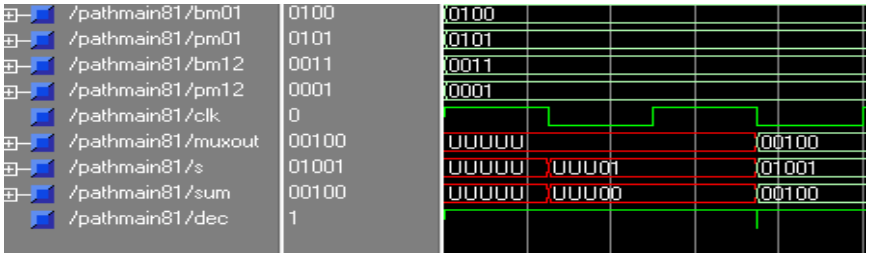

Fig. 17: Single state add compare select unit output waveform for W = 4, N = 2

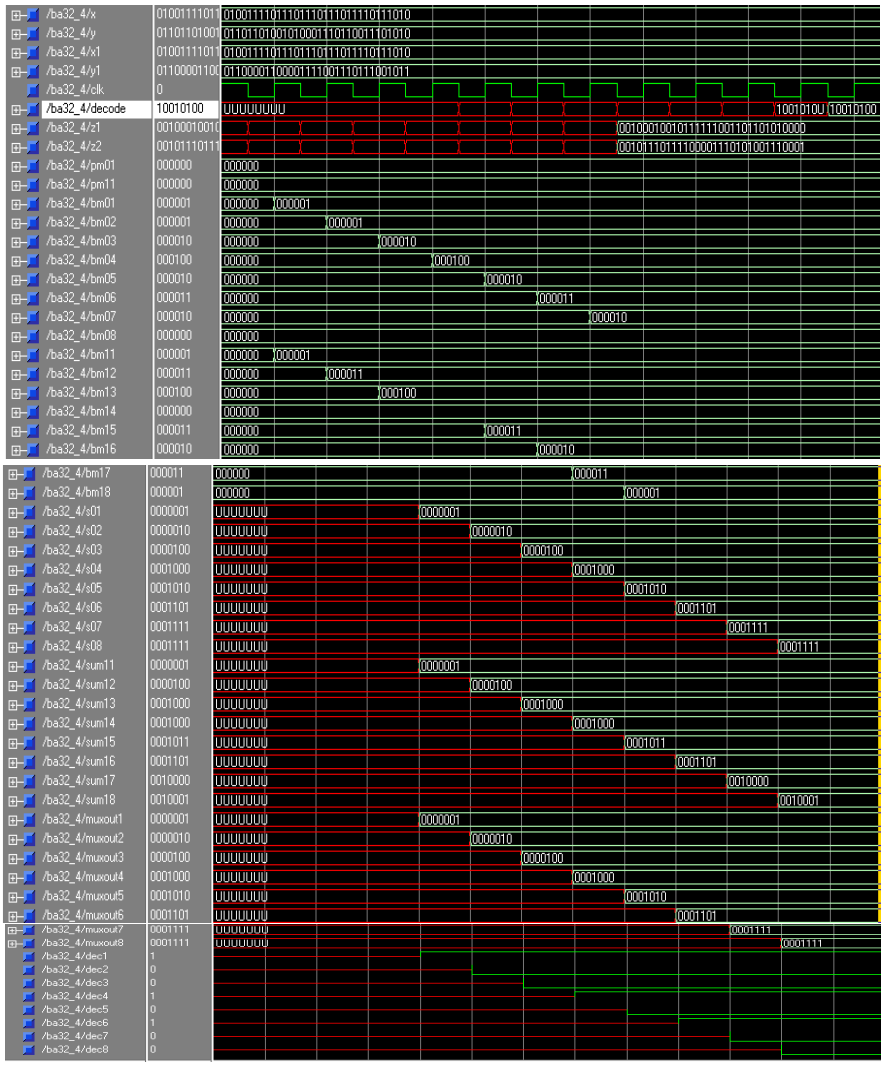

Fig. 18: Output waveform of digit-serial viterbi decoder for W = 32

compares s and sum, sum is smaller than s so comparator produce decision dec = 1.Based on this decision selector select sum and produce muxout = 00100, it is updated in state metric. Decision is given to the SMU to store decoded bit as 1 in the shift register.

Digit-serial viterbi decoder for W = 32: Figure 18 illustrates digit-serial Viterbi decoder for W = 32 and code rate = 1/4. In Fig. 18, x = x1 = 01001111011101110 111011110111010 are represented as received sequence and y = 011011010010100011101100111010 10, y1 = 01100

Table 4: Device utilization summary of digit-serial and bit-serial viterbi decoder

Logic utilization	Digit-serial for W = 8	Digit-serial for W = 32	Bit-serial for W=8	Bit-serial for W = 32	Existing method
Total number of slice register	358	850	274	576	316
Number 4 input LUTs	246	653	235	679	510
Number of occupied slices	286	623	246	527	-
Total number of 4 input LUTs	527	946	454	910	-
Number used as logic	246	653	235	679	-
IOB latches	2	8	2	8	8
Number of GCLKs	1	8	1	8	1
Total equivalent gate count for design	6,522	13,755	5,108	10,281	21504

Table 5: Performance comparison of designed viterbi decoder

Designed architectures	Maximum operating frequency (MHZ)	Power consumption (mw)
Bit-serial Viterbi decoder for W = 8	161.368	55
Bit-Level pipelined digit-serial Viterbi decoder for W = 8; N = 4	193.14	37
Bit-serial viterbi decoder for W = 32	162.787	92
Bit-level pipelined digit-serial Viterbi decoder for W = 32; N = 4	186.501	56

Table 6 Power consumption comparison of viterbi decoder

Architecture	Power consumption (mw)
Bit-level pipelined digit-serial architecture for viterbi decoder W = 8 N = 4.	37
Bit-level pipelined digit-serial architecture for viterbi decoder W = 32 N = 4	56
Two bit-level pipelined (ACSU) Viterbi decoder (Santhi *et al*., 2008, 2009)	78
Asynchronous pipelined Viterbi decoder	112
Synchronous pipelined Viterbi decoder	114

00110000111100111010111001011 are represented as expected sequence. The output of xor gate are represented as z1 = 00100010010111111001101101010000, z2 = 00101110111100001110101001110001 Upper branch metric are bm01, bm02, bm03, bm04, b05, bm06, bm07, bm08 and lower branch metric are bm11, bm12, bm13, bm14, bm15, bm16, bm17, bm18. Initial path metric pm01, pm11 are 000000. Further states are processed in the above manner and the decode sequence are stored in the register. Decoded sequence stored in SMU register for given input are decode = 10010100.

Synthesis report: The proposed bit-level pipelined digit-serial Viterbi decoder is simulated using Model sim and synthesized using Xilinx FPGA. The synthesis report in Table 4 suggests that there is an increase in gate count when the wordlength increases with a reduction in delay and logic transformations. Device utilization summary of the proposed viterbi decoder is compared with the existing reference where the decoder was designed for 802.11a for OFDM systems (Kang and Willson, 1998). Compared to the general Viterbi decoder, this design can ef-fectively decrease the 11% of chip logic elements, reduce 5% of power consumption and increase the encoder and de-coder working performance in the hardware implementation.

Table 5 shows the performance comparison of proposed and existing Viterbi decoder. Viterbi decoder is designed for bit serial architectures in order to perform the comparison, it is found that the critical path of the decoder is in ACS unit as it comprises of two-two input adder, comparator and a multiplexer. The designed architectures are for wordlength W = 8, 32 with increase in digit size. Thus, the number of operations is increased with reduced computation time. Proposed method increase clock frequency and decrease power consumption upto 20%.

Table 6 explains the power consumption of existing and proposed method. The proposed method is compared with the designed bit serial architecture and the methods from the literature survey, it reduces power consumption about 50% in asynchronous and synchronous pipelined architecture and 10% in two bit-level pipelined (ACSU) Viterbi decoder.

In this study, we proposed an efficient digit-serial viterbi decoder. From the viterbi algorithm the decoder hardware was obtained. Then the concept of digit and bit level technique is applied with the help of Data Flow graph and the switching instances are calculated as indicated in the materials and methods. The proposed multiplier was modelled in VHDL and simulated to verify its functionality. After verifying the proposed decoder's functionality, we compared the performance of the decoder with the previous research at three levels. The device utilization summary of the designed blocks are compared with reference. Performance of the proposed architectures is compared among themselves. Finally, the

power consumption of the proposed work is compared with references (Santhi *et al*., 2008, 2009). The observations of the proposed architecture are: 1) it does not restrict the choice of word and digit size.and 2) it can be pipelined at the bit level. In addition, the computational delay time of the proposed architecture is significantly less than previously proposed architectures from the comparison (Kang and Willson, 1998). Furthermore, since the decoder has the features of regularity, modularity, low power consumption it is well suited to VLSI implementation requiring moderate sample rates and where area is critical.

CONCLUSION

The proposed method explains high speed and low power Viterbi decoder using bit-level pipelined digit-serial architecture for various digit size and word length. Viterbi decoder is designed with code rate k/n = ¼, constraint length K = 3, word length W = 8, 32 and digit size N = 2, 4. The functionality is simulated and verified using Modelsim and synthesized using Xilinx FPGA Virtex. The power consumption of the proposed method for wordlength W = 8, 32 is 37mw and 56mw with maximum clock frequency of 186.501 and 193.14MHz.

REFERENCES

Aksoy, L., C. Lazzari, E. Costa, P. Flores and J. Monteiro, 2011. Optimization of area in digit-serial multiple constant multiplications at gate-level. Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), May 15-18, 2011, IEEE, Rio de Janeiro, Brazil, ISBN: 978-1-4244-9473-6, pp: 2737-2740.

Dholakia, A., 2012. Introduction to Convolutional Codes With Applications. Vol. 275, Springer Science & Business Media, North Carolina, USA., ISBN: 978-1-4613-6168-8, Pages: 208.

Dib, D.A.E. and M.I. Elmasry, 2004. Modified register-exchange viterbi decoder for low-power wireless communications. Circuits Syst. I Regul. Pap. IEEE. Trans., 51: 371-378.

Forney, J.G.D., 1972. Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference. Inf. Theory IEEE. Trans., 18: 363-378.

Forney, J.G.D., 1973. The viterbi algorithm. Proc. IEEE., 61: 268-278.

Goo, Y.J. and H. Lee, 2008. Two bit-level pipelined viterbi decoder for high-performance UWB applications. Proceedings of the IEEE International Symposium on Circuits and Systems ISCAS, May 18-21, 2008, IEEE, Seattle, Washington, ISBN: 978-1-4244-1683-7, pp: 1012-1015.

Haene, S., A. Burg, D. Perels, P. Luethi and N. Felber *et al*., 2005. FPGA implementation of Viterbi decoders for MIMO-BICM. Proceedings of the the Thirty-Ninth Asilomar Conference on Signals, Systems and Computers, October 28, 2005, IEEE, Pacific Grove, California, ISBN: 1-4244-0131-3, pp: 734-738.

Kang, I. and J.A.N. Willson, 1998. Low-power viterbi decoder for CDMA mobile terminals. Solid-State Circuits IEEE. J., 33: 473-482.

Karim, M.U., M.U.K. Khan and Y.M. Khawaja, 2011. An area reduced, speed optimized implementation of Viterbi decoder. Proceedings of the International Conference on Computer Networks and Information Technology (ICCNIT), July 11-13, 2011, IEEE, Abbottabad, Pakistan, ISBN: 978-1-61284-940-9, pp: 93-98.

Kim, J., S. Yoshizawa and Y. Miyanaga, 2009. Design of variable wordlength viterbi decoder in BICM-OFDM systems. Proceedings of the 9th International Symposium on Communications and Information Technology ISCIT, September 28-30, 2009, IEEE, Icheon, South Korea, ISBN: 978-1-4244-4521-9, pp: 849-852.

Landernas, K., J. Holmberg and O. Gustafsson, 2004. Implementation of bit-level pipelined digit-serial multipliers. Proceedings of the 6th Symposium on Nordic Signal Processing NORSIG, June 9-11, 2004, Malardalen University, Espoo, Finland, pp: 125-128.

Parhi, K.K. and D.G. Messerschmitt, 1987. Concurrent cellular VLSI adaptive filter architectures. Circuits Syst. IEEE. Trans., 34: 1141-1151.

Parhi, K.K., 1989. Look-ahead in dynamic programming and quantizer loops. Proceedings of the IEEE International Symposium on Circuits and Systems, May 8-11, 1989, IEEE, Portland, Oregon, pp: 1382-1387.

Parhi, K.K., 1999. VLSI Digital Signal Processing Systems: Design and Implementation. John Wiley and Sons, New Jersey, USA., ISBN: 9780471241867, Pages: 808.

Santhi, M., G. Lakshminarayanan and S.V. Varadhan, 2008. FPGA based asynchronous pipelined viterbi decoder using two phase bundled-data protocol. Proceedings of the International Conference on SoC Design ISOCC'08, November 24-25, 2008, IEEE, Busan, South Korea, ISBN: 978-1-4244-2598-3, pp: 314-317.

Santhi, M., G. Lakshminarayanan, R. Sundaram and N. Balachander, 2009. Synchronous pipelined two-stage radix-4 200Mbps MB-OFDM UWB viterbi decoder on FPGA. Proceedings of the International Conference on SoC Design Conference (ISOCC), November 23-24, 2009, IEEE, Busan, South Korea, pp: 468-471.

Sun, Y. and Z. Ding, 2012. FPGA Design and Implementation of a Convolutional Encoder and a Viterbi Decoder Based on 802.11 a for OFDM. Wirel. Eng. Technol., 3: 125-131.

Zengliang, Z. and X. Cheng, 2011. Design method of viterbi decoding of convolutional code based on VHDL. Proceedings of the 2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN), May 27-29, 2011, IEEE, Xian, China, ISBN: 978-1-61284-485-5, pp:178-181.

Zhang, X. and K.K. Parhi, 2004. High-Speed VLSI architectures for the AES algorithm. IEEE Trans. Very Large Scale Integr. Syst., 12: 957-967.