International Journal of Engineering & Technology, 6 (4) (2017) 131-140

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi: 10.14419/ijet.v6i4.8038
Research paper

Implementation of a standard inner convolutional codec for DVB-T system using VHDL

Dina M. Hussein *, Abdelhalim Zekry, Said Baioumy, Fatma El-Newagy

Modern Academy for Engineering and Technology, Egypt Ain Shams University, Egypt
**Corresponding author E-mail: dinamohamed2110@gmail.com*

Abstract

Forward error correction (FEC) plays a vital role in digital communication systems. DVB-T system uses FEC as a channel coding technique to restore any data lost through transmission to the receiver. DVB-T system uses two levels of error protection. The first level is applied in the data transmitted by using a Reed-Solomon RS (204, 188) code followed by a convolutional interleaver. The other level of error protection is a punctured convolutional inner coding followed by an inner interleave in which the data sequence is rearranged again to minimize the influence of burst errors.
This paper describes the implementation of inner convolutional codec (Convolutional coder and Viterbi Decoder) and inner de/interleaving of a standard DVB-T system with a constrained length of 7 and a code rate of 2/3 using VHDL on virtex-6 FPGA xc6vlx240t. The designed channel convolutional encoder and Viterbi decoder follow European Standard ETSI EN 300 744 for digital terrestrial television. Verification of the design is accomplished by loop back and by comparison with the corresponding Xilinx core. Utilization and timing re-ports of the implemented device on Vertex 6 are included.

Keywords: *Channel Coding; Convolutional Coding (Inner Coding); DVB-T; Inner De/ Interleaver; Viterbi Decoder.*

1. Introduction

Digital video broadcasting terrestrial (DVB-T) is the name of the terrestrial transmission system developed by the DVB Project. DVB-T is running in many countries all over the world. [1]
In communication systems, errors correcting codes are used in detection and correction of the transmitted data errors. [2] Convolutional encoding with Viterbi decoding used in the DVB-T system is a powerful process for forward error correction.[2] Convolution code is a sort of error correcting code in which (k-bits) information symbol is to be encoded to (n-bits) data were (n>k). [3]
To decode the output data from the inner convolution encoder, a Viterbi decoder is used at the receiver. [4], [5] in 1967, Andrew J. Viterbi resolved and suggested the Viterbi decoding algorithm. [4] It is vastly used as a decoding way for convolutional codes. [4] This algorithm uses trellis structure, which is traced back for decoding the received information.[4] Also, it performs maximum likely hood detection on data that is coded by the convolutional encoder.[4] Viterbi decoding uses trellis diagram to get the output data that is most likely similar to the input sequence coming from the encoder. However, it reduces the complexity by using trellis structure. [4]
At market, almost Viterbi decoders are intellectual property (IP) core with an effective algorithm in the decoding of only one convolutional encoded sequence that is in some way expensive.[4] In addition, the costs for the convolutional encoder and Viterbi decoder are expensive for a specified design because of the patent issue.[4], [5] [5] Therefore, to obtain convolutional encoder and Viterbi decoder on a field programmable gate array (FPGA) board is so demanding.[5] This paper is concerned with designing and implementing a convolutional encoder and a Viterbi decoder with interleaver/de-interleaver of DVB-T system, that are essential blocks in digital communication systems, using FPGA technology and comparing their results with that of the IPcore built-in design on Xilinx ISE (Integrated Software Environment) design Suite 12.4 program.
There were many efforts to design and implement a convolutional encoder and Viterbi decoder with same constraint length(k=7) but without puncturing and with hard Viterbi decoder as in these papers [6], [7], [8], [9]. But this paper added the puncturing with code rate 2/3, soft Viterbi decoder and also their integration with inner interleaver and inner deinterleaver. Also, in this paper, a comparison is made with the decoder IPcore output, which implemented the whole system in Xilinx design suite ISE.
The rest of the paper is organized as follows: In section 2, the inner coding of the DVB-T system model is discussed. In section 3, the implementation of the proposed inner coding of the DVB-T system using “Xilinx ISE Design Suite 12.4” program is given. In section 4, VHDL modeling and output simulation results are shown. Also, soft Viterbi decoder output vs. that of the IPcore of the Viterbi decoder are shown. Finally; the conclusions are given in section 5.

2. Inner coding of DVB-T system model

In this section the Inner Convolutional Encoder/Interleaver and Inner Viterbi Decoder/Deinterleaver of the DVB-T system according to the European Standard ETSI EN 300 744 for digital terrestrial television are presented.[3]

2.1. DVB-T transmitter

The block diagram for a DVB-T transmitter is shown in Figure 1.

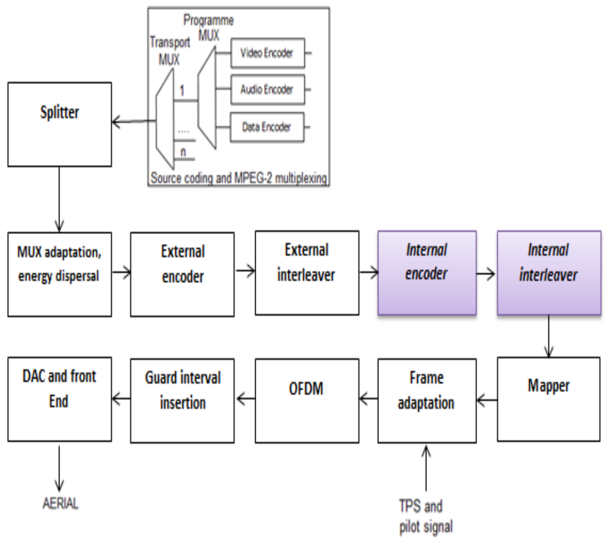


Fig. 1: Block Diagram of DVB-T Transmitter.

Compressed video, audio, and data streams are multiplexed into programme streams (PS), and then connected together into an MPEG-2. Transport stream (TS) connected to a splitter who can transmit two different TSs at the same time, send them to energy dispersal and MUX adaptation that decor relate the byte sequence. Then the signal goes to Reed-Solomon external encoder that allows the correction of up to a maximum of 8 wrong bytes for each 188 byte packet. The external interleaves rearranges the data sequence. The internal encoder produces a punctured convolutional code while the internal interleaver rearranges again the data sequence to reduce errors by using two separate interleaving processes, one operating on bits and another one operating on a group of bits. Then the bit sequence is mapped into a baseband modulated sequence using OFDM transmission. The transmitted signal is organized in frames by Frame adaptation. TPS and pilot signal are additional signals that are inserted in each block in order to simplify the reception of the signal being transmitted on the terrestrial radio channel. The sequence of blocks is modulated according to the OFDM technique. Guarding interval insertion is used to decrease receiver complexity. DAC and Front-end convert the digital signal into analog signal and then shifted to radio frequency (UHF) by the RF front-end.

2.1.1. Punctured internal encoder

An Inner error protection is accomplished by the convolutional encoding.[10] It represents the second level of protection in DVB-T system. [10] Its implementation is simple logic, as it consists of shift registers and XOR gates.[10] To reduce the number of bits transmitted, several puncturing schemes are used which delete selected bits from the encoder output.[10] The puncture rates are 2/3, 3/4, 5/6, and 7/8. [10] They can be implemented externally to the convolutional encoder.[10] This permits the freedom to change between the different puncture rates.[10]

2.1.1.1. Internal encoder (convolutional encoder)

The inner convolutional encoder consists of two signal paths and a 6 stage shift registers as shown in Figure 2, in which the input signal is mixed with the content of the shift register at certain tapping points.[3] The data stream input is divided into 3 data streams where the data route through the shift register affects both the upper and lower data streams by an Exclusive OR operation that lasts for 6 clock cycles.[3] This divides the information of one bit through 6 bits.[3] At particular points, there are EXOR gates in the upper data branch and the lower data branch that mix the data with the contents of the undelayed shift register, that provides at the output of the convolutional coder two steams of data, each of which shows the same data rate as that of the input signal.[3] In addition, the data stream was only provided with a particular memory extending over 6 clock cycles.[3] The total data rate at the output is then twice as high as the input data rate which corresponds to a code rate = 1/2. In which the two output data streams together carry 100% overhead, which is error protection. On the other hand, this reduces the available net data rate. This overhead, and that error protection, can be controlled by the puncturing unit shown in Table 1.[3].

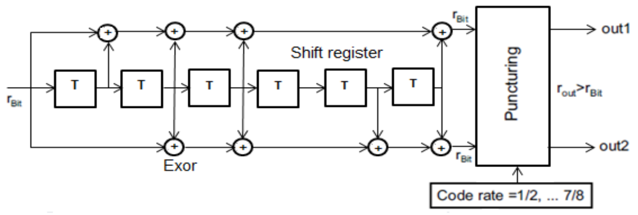


Fig. 2: Convolutional Coder in DVB-T system [3].

2.1.1.2. Puncturing

The data rate can be decreased by choosing omitted bits that are made by an arrangement called the puncturing pattern, that is recognized by both the transmitter and the receiver which easy the variation of the code rate between 1/2 and 7/8.[3] 1/2 means no puncturing, that is maximum error protection, and 7/8 means minimum error protection and a maximum net data rate.[3] At the receiver, these bits are filled with ‘Don’t Care’ bits and are handled like errors in Viterbi decoding and then rebuilt.[3] In the case of DVB-T, these two streams are combined together to compose a common data stream by accessing the upper and lower punctured data stream X and Y, respectively.[3]
We use code rate=2/3 which means that for every 2 input bits there are only 3 output bits instead of 4 output bits as seen in Table 1

Table 1: Puncturing Pattern [11]

Code Rates r	Puncturing pattern	Transmitted sequence (after parallel-to-serial conversion)
½	X:1 Y:1	X1 Y1
2/3	X:10 Y:11	X1 Y1 Y2
¾	X:101 Y:110	X1 Y1 Y2 X3
5/6	X:10101 Y:11010	X1 Y1 Y2 X3 Y4 X5
7/8	X:1000101 Y:1111010	X1 Y1 Y2 Y3 Y4 X5 Y6 X7

2.1.2.. Internal interleaver

It is the last process of these error protection techniques. It consists of bit-wise interleaver and symbol interleaver as shown in Figure 3. [10] The aim of the interleaver is to vary the symbols sequence to distribute any errors that might be introduced through the transmission channel. [6]

2.1.2.1. Bit-wise interleaving

It is carried out only on the useful data.[11] Data is demultiplexed into v sub-streams, where v = 2 for QPSK, v=4 for 16-QAM, and v=6 for 64-QAM respectively.[7], [8]

2.1.2.2. Symbol interleaver

It maps v bit words to the active carriers per OFDM symbol. For 2k mode, 126 data from bit interleaver are grouped into 12 to give vector length 1512 while for 8k mode these 126 data are grouped into 48 to give vector length 6048. [7], [8] 2k mode is the one used in this design implementation.

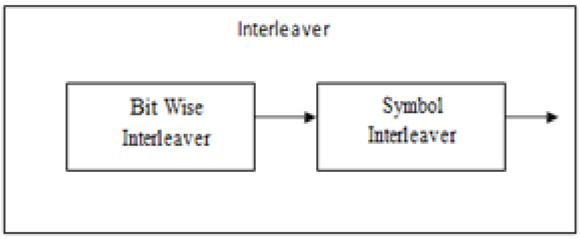


Fig. 3: DVB-T Internal Interleaver DVB-T Receiver.

2.2. DVB-T Receiver

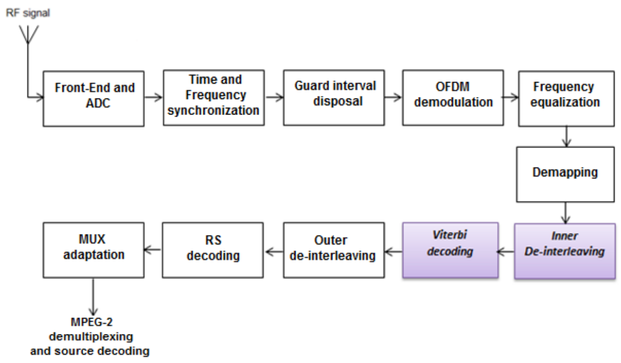


Fig. 4: Block Diagram of DVB-T Receiver.

2.2.1. Inner deinterleaver

It consists of symbol deinterleaver followed by bit-wise deinterleaver. It rearranges again the bits scrambled by interleaver.

2.2.2. Viterbi decoder (inner decoder)

Its constraint length is 7, traceback length=84 and uses polynomials 171 (octal) = 1111001 (binary) and 133 (octal) = 1011011 (binary).
To deal with the puncture rates, as given by the standard, puncturing is executed outside the Viterbi decoder, and instead of the missing symbols, null symbols is inserted with an erase input to denote the position of these null symbols. [10]
We use soft coding format (with soft width=3) as a decoder input to perform a better bit error rate (BER), as it gives each bit a confidence value ranging from a maximum confidence zero (000) to a maximum confidence one(111).[10]
DVB-T receiver is shown in Figure 4 where the data path is reversed relative to that of the transmitter. Inner deinterleaver and Viterbi decoder are explained in the following sub-sections.

3. Proposed system implementation

In this section, we introduce the implementation of the convolutional encoder, inner interleaver, inner deinterleaver, and Viterbi decoder of the DVB-T system as depicted in Figure 5(a),5(b) using “Xilinx ISE Design Suite 12.4” program.

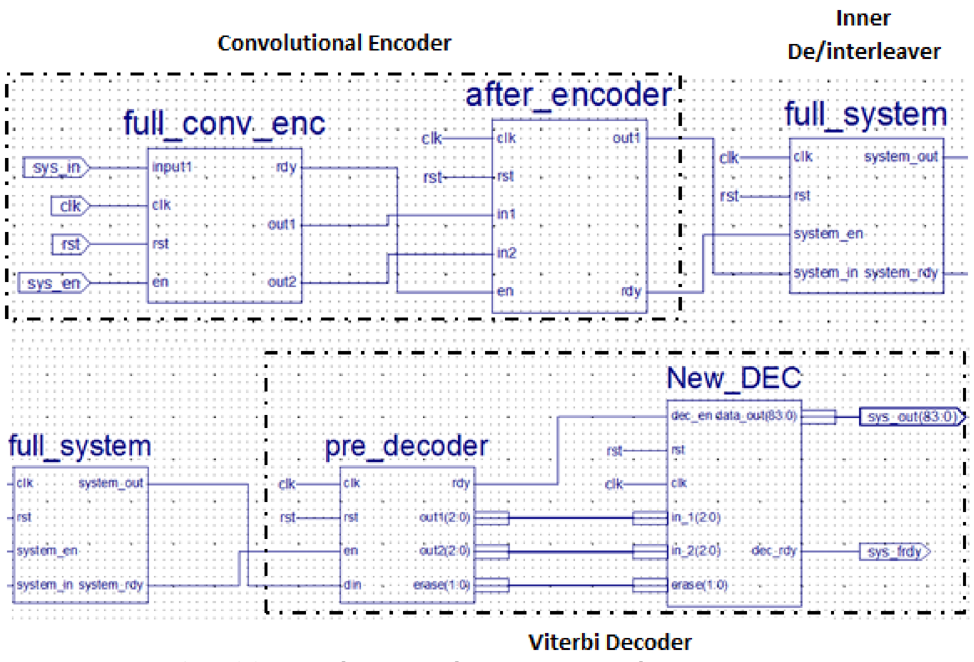


Fig. 5: (A) Proposed System Implementation Using Xilinx ISE Program.

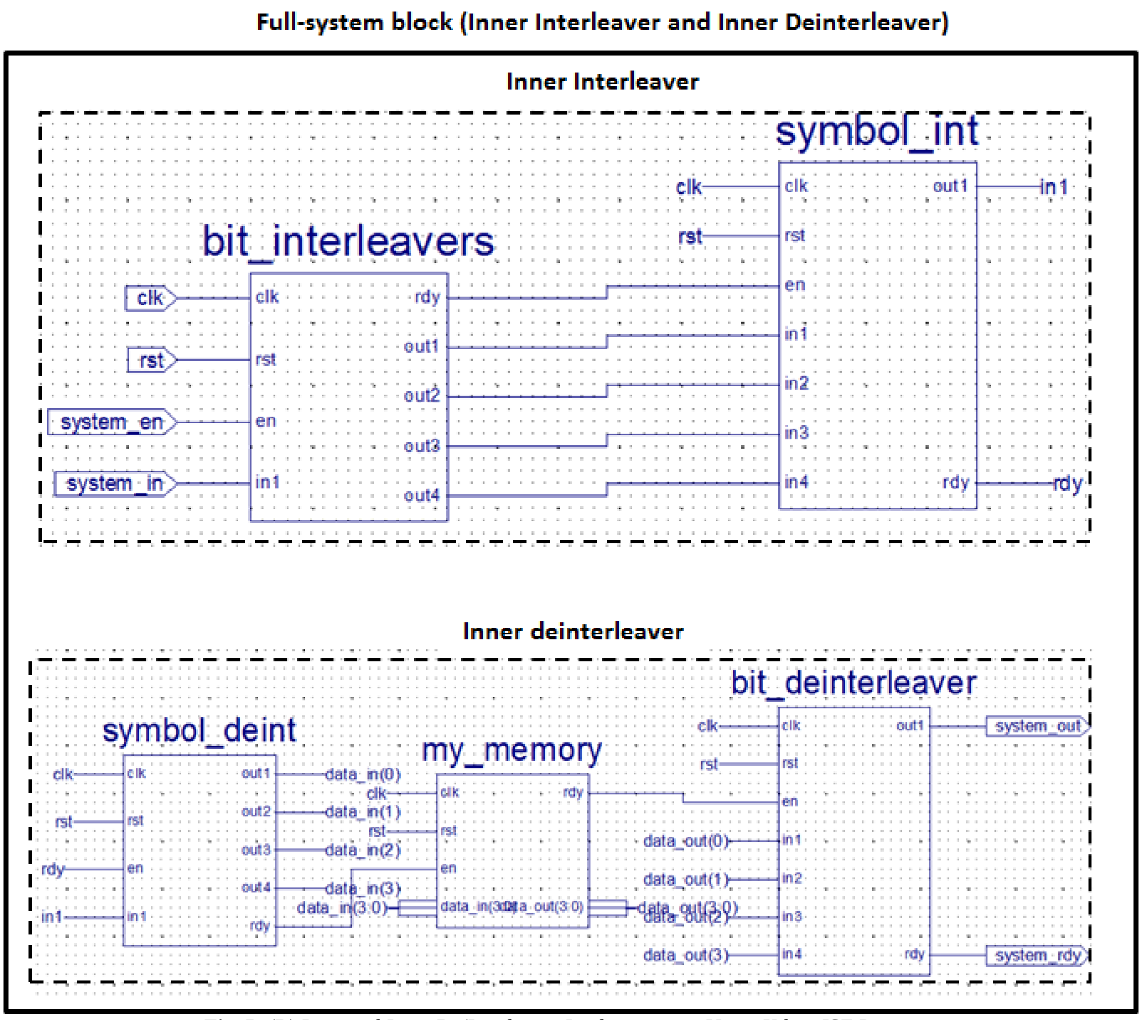


Fig. 5: (B) Proposed Inner De/Interleaver Implementation Using Xilinx ISE Program.

3.1. Inner convolutional encoder and puncturing

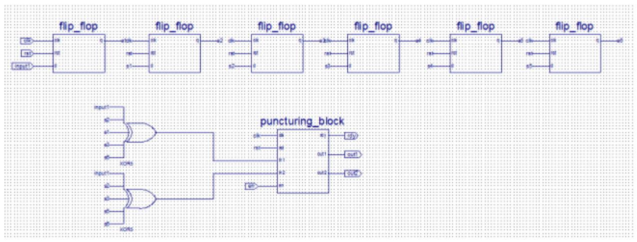


Fig. 6: DVB-T Inner Convolutional Encoder.

Inner Convolutional encoder: The encoder consists of 6 shift registers flip flops stages connected by XOR gates as shown in Figure 6 according to the generator polynomials:
G1=171 (octal) for X o/p = 1111001 (binary), and
G2=131 (octal) for Y o/p = 1011011 (binary).
Its memory= 2^6=64 and has 2 o/p data streams, each with the same data rate as the input data rate. Then we enter the output of the gates to the puncturing block.
The puncturing block: It takes the 2 parallel outputs of the encoder, omits some bits according to the pattern shown in Figure 7. Using a code rate =2/3 (means that for every 2 i/p bits we have 3 o/p bits rather than 4 bits and output 2 parallel 252 bits each. This protocol is agreed with our Decoder.

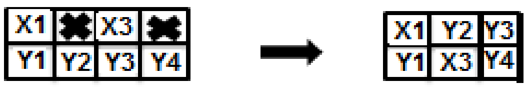


Fig. 7: [2/3] Punctured Code Rate Pattern.

After puncturing, the 504 parallel bits output from the puncturing block are converted into a serial data to feed them to the interleaver.

3.2. Inner interleaver and deinterleaver

This ‘Full system’ block implemented in figure 5 consists of inner interleaver and inner deinterleaver.

3.2.1. Inner interleaver

Inner interleaver is designed as Bit-wise-interleaver and Symbol-interleaver.

3.2.1.1. Bit-wise interleaver

It consists of 4 shift registers 126 bits each. These bits will be interleaved and delivered in 4 parallel branches in 126 cycles such that the output of the interleaver can be expressed by the (out1, out2, out3, out4) according to the equations:
outputI0 [w] = inputI0 [w]
outputI1 [w] = inputI1 [(w+63) mod 126]
outputI2 [w] = inputI2 [(w+105) mod 126]
outputI3 [w] = inputI3 [(w+42) mod 126]
Then go to “symbol interleaver”.

3.2.1.2. Symbol interleaver

Its input = 378 bits (three times of 126 bits on each one of the 4 parallel inputs) .It takes this data and interleave them serially. It is pipelined with the bit interleaver. Its output 504x3=1512 bits serially.
The system is back to back system so the data goes to the symbol deinterleaver

3.2.2. Inner deinterleaver

3.2.2.1. Symbol deinterleaver

It deinterleaves the data according to the mapping of the symbol interleaver and outputs 378 bits on the 4 outputs three times.

3.2.2.2. My memory

Acts as a shift register that makes delay to the data in which it takes 378 bits on 4 parallel input pins and divides them to 126 bits on 4 output pins.

3.2.2.3. Bit-wise deinterleaver

The bit interleaver needs 126 bits only as input on each input bit of the 4 parallel inputs. It deinterleaves the data and sends 504 bits serial to the decoder.

3.3. Viterbi decoder

The steps of Decoding are:

1) Calculate the error for a transition
2) Compare 2 transitions
3) Decide transition & Record error
4) Repeat (1),(2)&(3) 84 times
5) Trace back the path chosen corresponding to least error which is the best state

The implemented decoder is designed by some blocks as shown in Figure 8 to maintain our requirements. These blocks are named: simple block, my add small, add compare, my ram, get min, get min2, ram total and last block.

Pre-decoder: as the decoder needs 2 parallel inputs, it converts serial input data to two parallel data and changes it from hard to soft representation with every bit is presented by 3 bits. Also it tells the decoder the flag of the puncturing so as to be suited with the encoder, its output is 126 bits.

Simple block: It calculates the error of one bit; each one is responsible for 1 state. Every 2 simple blocks are connected to my add small.

My add small: It adds the errors coming from 2 simple blocks for one transition (e0 & e1); where e0 is the sum of 1st bit and 2nd bit errors coming from simple block while e1 is that of 3rd bit and 4th bit. Every 2 my add small is connected to add compare.

Add compare: It adds c0 + e0 and c1+e1, compare them, get the smallest one and record its state where c0 & c1 are sum of errors of the path.

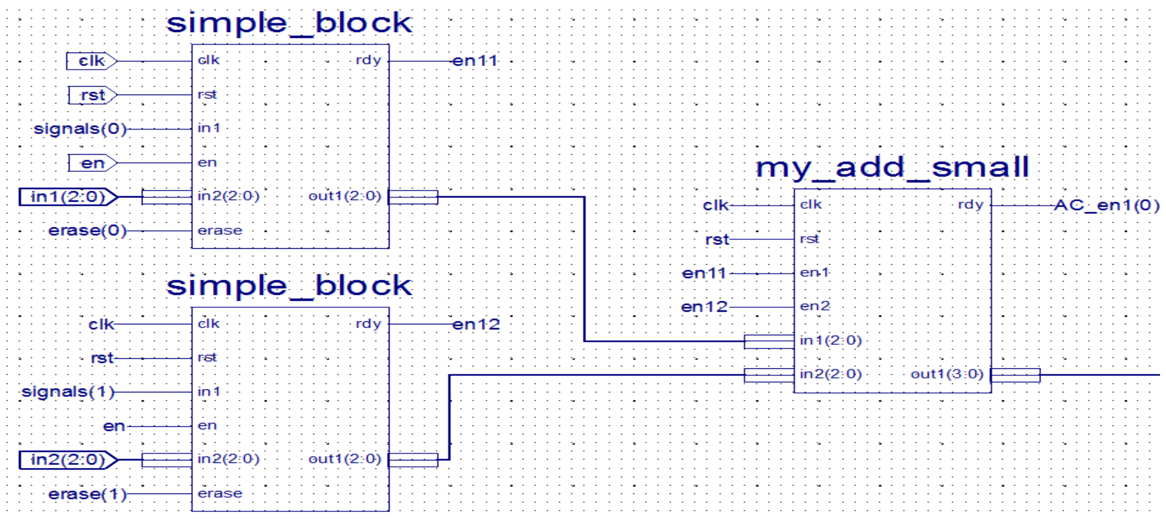
My ram: It records all the states coming from add compare block and save them in a register.

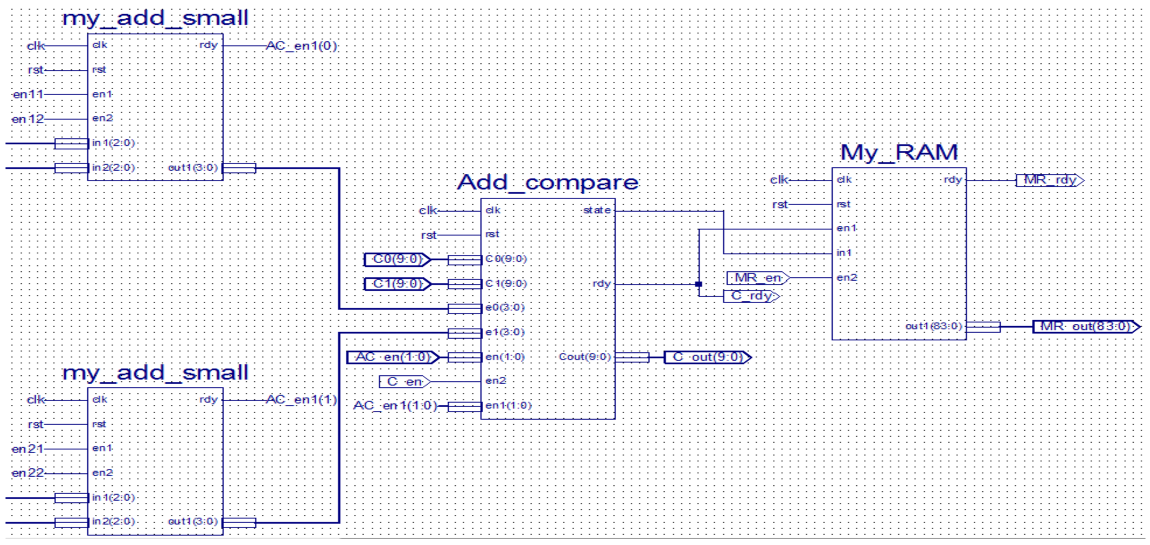
Get min: It gets its input from 8 states (8 commutative errors), gets their minimum by comparing them and determines their best state. There are 8 blocks of Get min to get 8 minimums out of 64 bits.

Get min2: It gets its input from the 8 states (8 outputs) of Get min blocks, and delivers the index of the minimum one and send it to Last block.

Ram total: It takes the data from My Ram blocks, records them and then gives them to Last Block when it needs them.

Last block: receives the data from ram total and the minimum index from Get min and outputs the decoded data in 84 bits





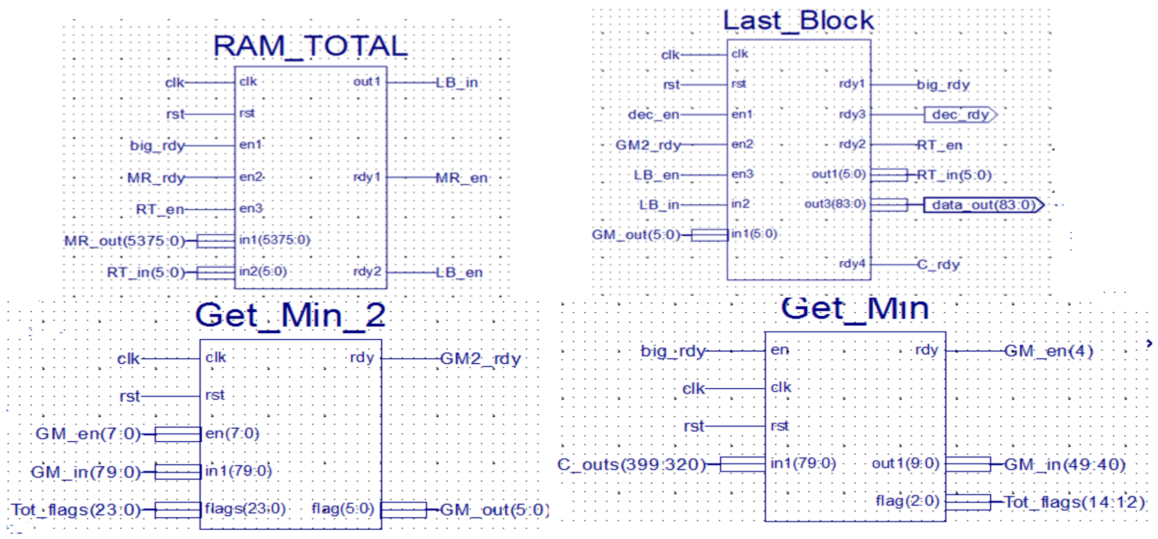


Fig. 8: Soft Viterbi Decoder Implementation.

4. VHDL modeling and simulation results

4.1. System modeling and simulation

4.1.1. The integrated system model

The designed RTL model of the integrated inner codec is depicted in Figure 9. It consists of the model for the convolutional encoder, the after encoder block that acts as a parallel to serial block, the full system block that consists of inner interleaver and inner deinterleaver),the pre-decoder block that acts as serial to parallel block ,and the viterbi decoder. The pin assignment of all blocks is indicated in the same Figure 9. The detailed operation and the resulting waveforms for all the implemented functional blocks are presented in the following section.

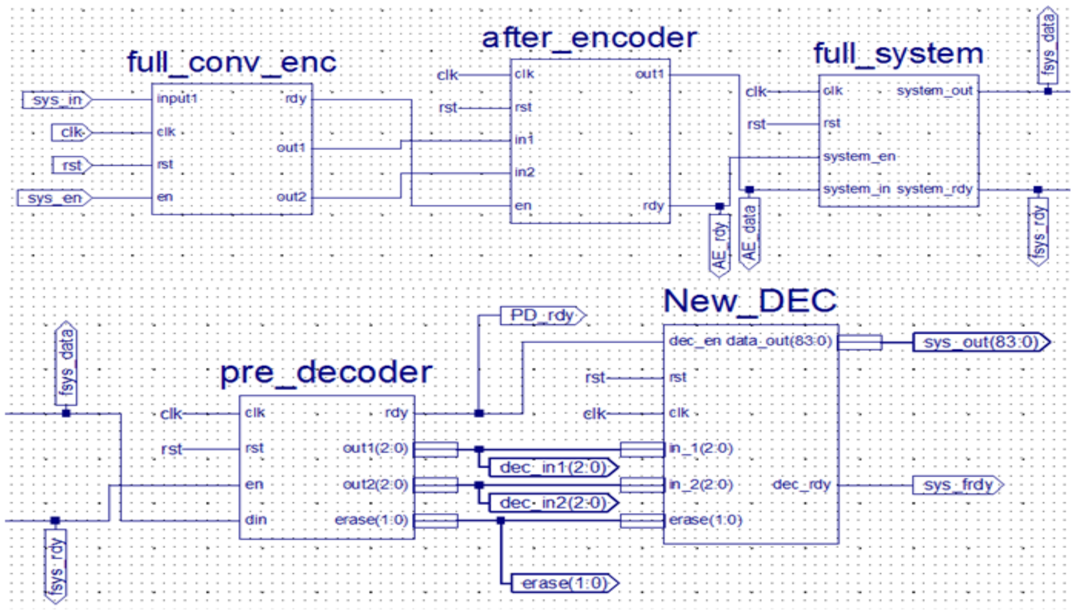
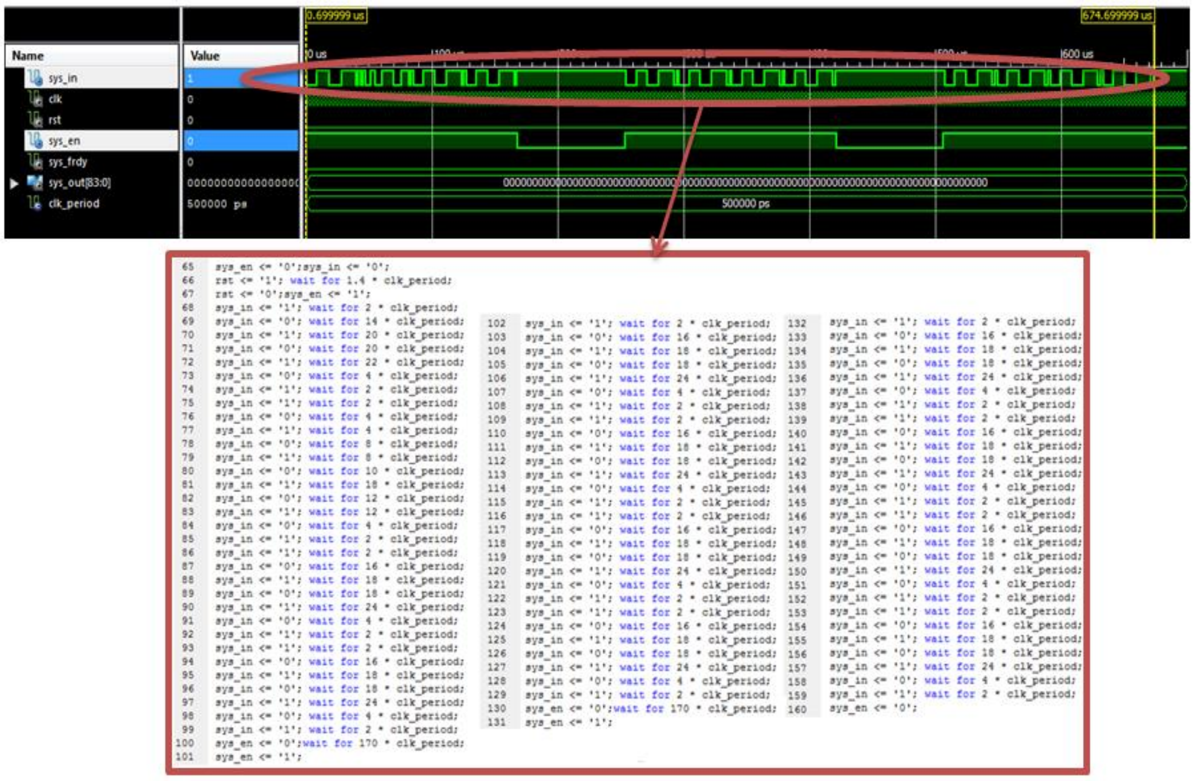


Fig. 9: Overall Proposed System Integration Using Xilinx Suite.

4.1.2. Convolutional encoder and soft viterbi decoder waveforms

At rising edge, reset =0 and the enable of the system pin named “sys-en”=1, the data goes through the input of the system named “sys-in” according to the test-bench values as shown in Figure 10 which are 336bits and wait 170bits by changing “sys-en” from ‘1’ to ‘0’ and then “sys-en” will be 1 again to take 336bis again followed by 170bits waiting and then 336bits again.

After delay equal to 1482.5us as shown in Figure 12 in which the ready pin of the receiver that is called “sys-frdy” is high, allowing the output to appear at the receiver’s o/p pin that is called “sys-out” which is 84 bits as shown in Figure 11 in Hexadecimal where in Figure 11(a) the output is ‘c3fffff00000fffff0003’which is equal to ‘(MSB)11000011111111111111111111110000000000000000000011111111111111111111000000000000000011(LSB)’in binary, in Figure 11(b) the output is ‘c3ff003ffff003fc03c3’ which is equal to ‘(MSB)11000011111111111100000000000011111111111111100000000000111111110000000011110000 11(LSB)’in binary and in Figure 11(c) the output is ‘c3fffffc0000ffffc0003’ which is equal to ‘(MSB)11000011111111111111111111111100000000000000000111111111111111111000000000000000011(LSB)’. This output (which is read from LSB to MSB) is found to be as the input data to the encoder via the test bench in Figure 10.



```
sys_en <= '0';sys_in <= '0';
rst <= '1'; wait for 1.4 * clk_period;
rst <= '0';sys_en <= '1';
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '0'; wait for 14 * clk_period;
sys_in <= '1'; wait for 20 * clk_period;
sys_in <= '0'; wait for 20 * clk_period;
sys_in <= '1'; wait for 22 * clk_period;
sys_in <= '0'; wait for 4 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '0'; wait for 4 * clk_period;
sys_in <= '1'; wait for 4 * clk_period;
sys_in <= '0'; wait for 8 * clk_period;
sys_in <= '1'; wait for 8 * clk_period;
sys_in <= '0'; wait for 10 * clk_period;
sys_in <= '1'; wait for 18 * clk_period;
sys_in <= '0'; wait for 12 * clk_period;
sys_in <= '1'; wait for 12 * clk_period;
sys_in <= '0'; wait for 4 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '0'; wait for 16 * clk_period;
sys_in <= '1'; wait for 18 * clk_period;
sys_in <= '0'; wait for 18 * clk_period;
sys_in <= '1'; wait for 24 * clk_period;
sys_in <= '0'; wait for 4 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '0'; wait for 16 * clk_period;
sys_in <= '1'; wait for 18 * clk_period;
sys_in <= '0'; wait for 18 * clk_period;
sys_in <= '1'; wait for 24 * clk_period;
sys_in <= '0'; wait for 4 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_en <= '0';wait for 170 * clk_period;
sys_en <= '1';
```

```
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '0'; wait for 16 * clk_period;
sys_in <= '1'; wait for 18 * clk_period;
sys_in <= '0'; wait for 18 * clk_period;
sys_in <= '1'; wait for 24 * clk_period;
sys_in <= '0'; wait for 4 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '0'; wait for 16 * clk_period;
sys_in <= '1'; wait for 18 * clk_period;
sys_in <= '0'; wait for 18 * clk_period;
sys_in <= '1'; wait for 24 * clk_period;
sys_in <= '0'; wait for 4 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '0'; wait for 16 * clk_period;
sys_in <= '1'; wait for 18 * clk_period;
sys_in <= '0'; wait for 18 * clk_period;
sys_in <= '1'; wait for 24 * clk_period;
sys_in <= '0'; wait for 4 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '0'; wait for 16 * clk_period;
sys_in <= '1'; wait for 18 * clk_period;
sys_in <= '0'; wait for 18 * clk_period;
sys_in <= '1'; wait for 24 * clk_period;
sys_in <= '0'; wait for 4 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_en <= '0';wait for 170 * clk_period;
sys_en <= '1';
```

```
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '0'; wait for 16 * clk_period;
sys_in <= '1'; wait for 18 * clk_period;
sys_in <= '0'; wait for 18 * clk_period;
sys_in <= '1'; wait for 24 * clk_period;
sys_in <= '0'; wait for 4 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '0'; wait for 16 * clk_period;
sys_in <= '1'; wait for 18 * clk_period;
sys_in <= '0'; wait for 18 * clk_period;
sys_in <= '1'; wait for 24 * clk_period;
sys_in <= '0'; wait for 4 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '0'; wait for 16 * clk_period;
sys_in <= '1'; wait for 18 * clk_period;
sys_in <= '0'; wait for 18 * clk_period;
sys_in <= '1'; wait for 24 * clk_period;
sys_in <= '0'; wait for 4 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_in <= '0'; wait for 16 * clk_period;
sys_in <= '1'; wait for 18 * clk_period;
sys_in <= '0'; wait for 18 * clk_period;
sys_in <= '1'; wait for 24 * clk_period;
sys_in <= '0'; wait for 4 * clk_period;
sys_in <= '1'; wait for 2 * clk_period;
sys_en <= '0';
```

Fig. 10: Random Inputs Given to the Convolutional Encoder Via Testbench.

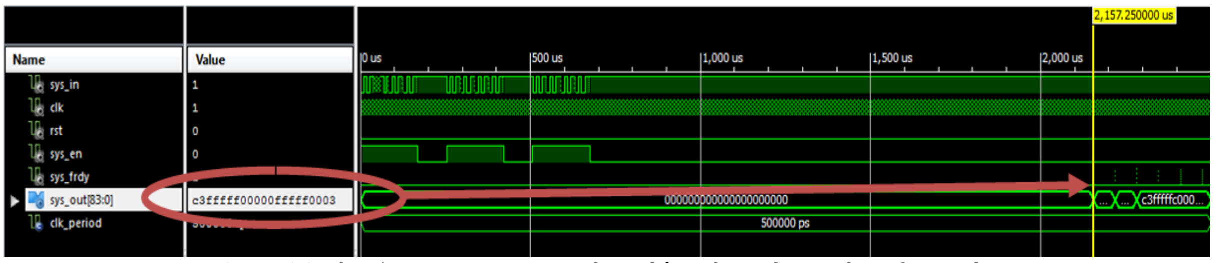


Fig. 11: (A): The 1st 84 Bits Output in Hexadecimal from the Implemented Viterbi Decoder.

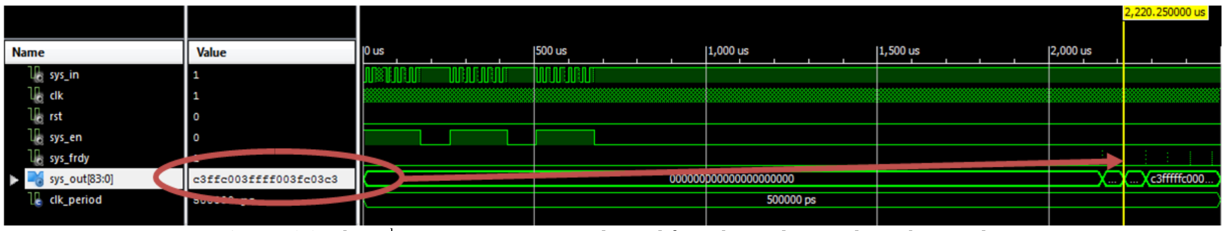


Fig. 11: (B): The 2nd 84 Bits Output in Hexadecimal from the Implemented Viterbi Decoder.

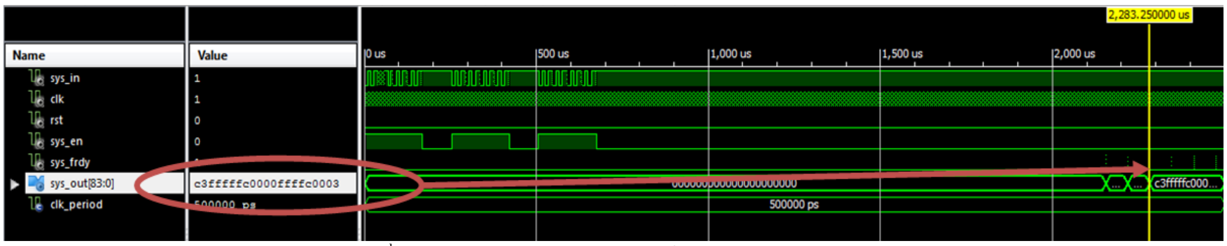


Fig. 11: (C): The 3rd 84 Bits Output in Hexadecimal from the Implemented Viterbi Decoder.
Fig. 11: (A), (B) and (C) Output of the implemented system with the test-bench given for the first 252 bits input.

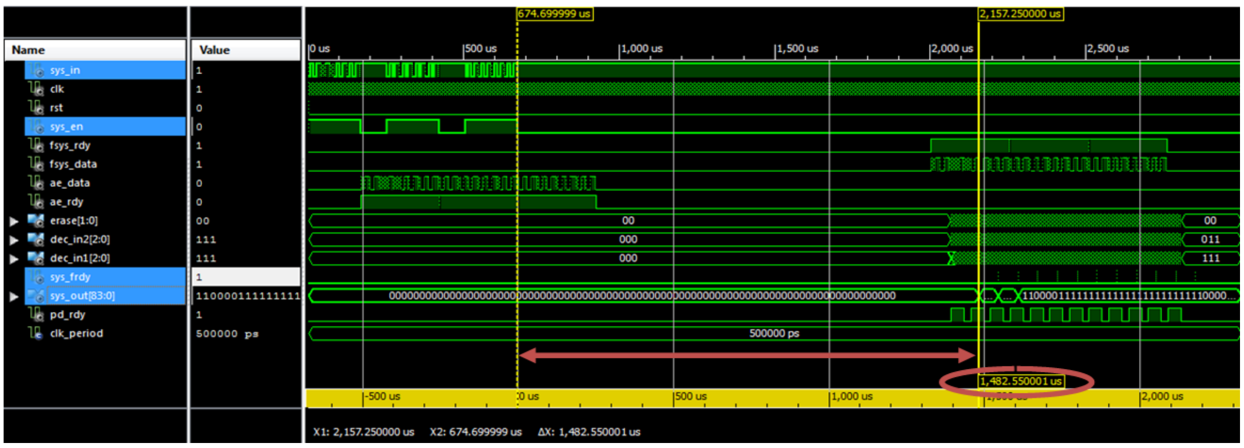


Fig. 12: The Delay between Input and Output.

4.1.3. Inner interleaver and inner deinterleaver waveforms

Figure13(a) presents the output of the encoder which is the input to the inner interleaver that is assigned in pin called “ae_data” with its ready pin “ae_rdy” also it presents the output of the inner deinterleaver which is input to the decoder that is assigned in pin called “fsys_data” with its ready pin called “fsys_rdy”. They are shown separately in Figure 13(b) and Figure 13(c).

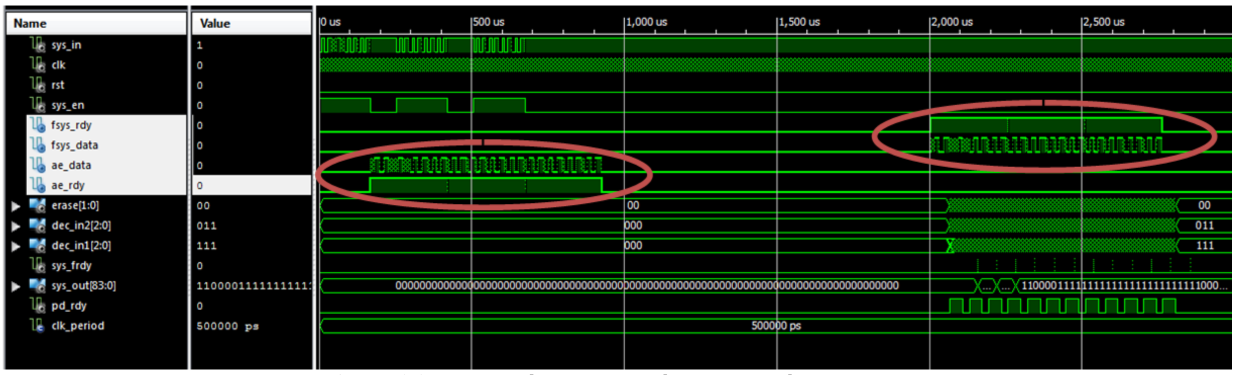


Fig. 13: (A): Inner Interleaver Input and Inner Deinterleaver Output.

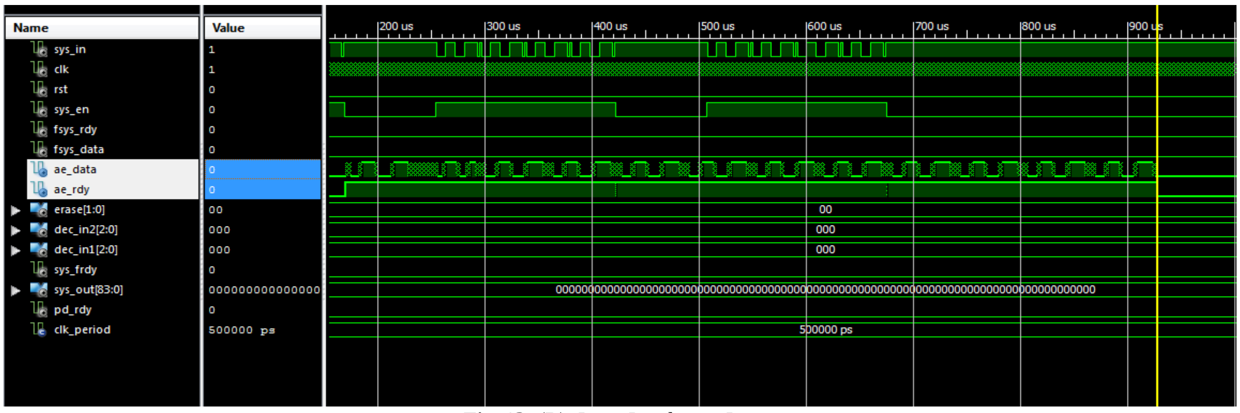


Fig. 13: (B): Inner Interleaver Input.

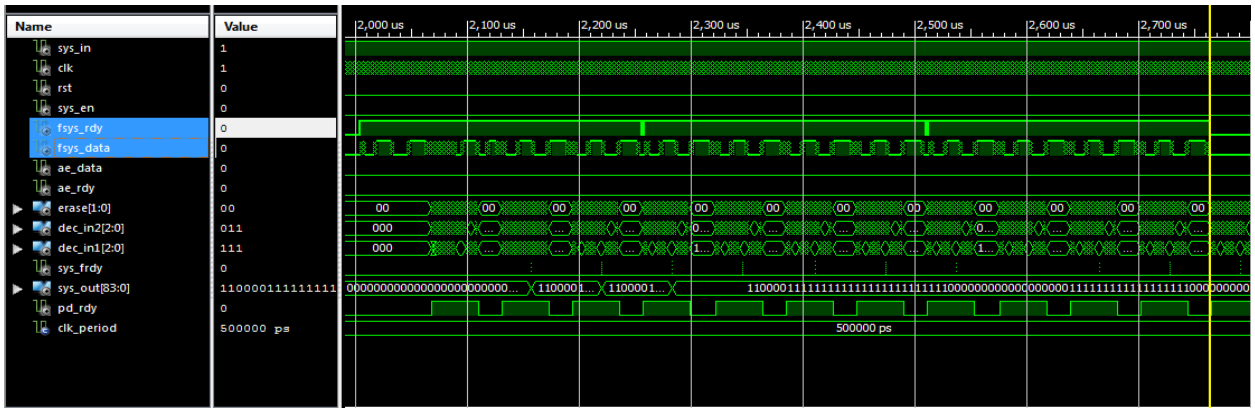


Fig. 13: (C): Inner Deinterleaver Output.

As seen in Figures 13(b) and Figure 13(c), the input of the inner interleaver is the same as the output of the inner deinterleaver

4.1.4. Proposed soft viterbi decoder vs Xilinx IPcore soft viterbi decoder waveforms

The result of the proposed soft viterbi decoder output data is shown in Figure 14 at the pin called "data_out".It takes 89.75us as a processing time to start showing this data. This result is compared by Xilinx IPcore built-in soft viterbi decoder having the same input througth the same test-bench given to the implemented decoder and the same parameters, that results in the same output as shown in Figure15 at pin called "d-out" but after a processing time equals to 182.25us.
By comparing these results the processing time of the proposed soft viterbi decoder of DVB-T sysem is improved.

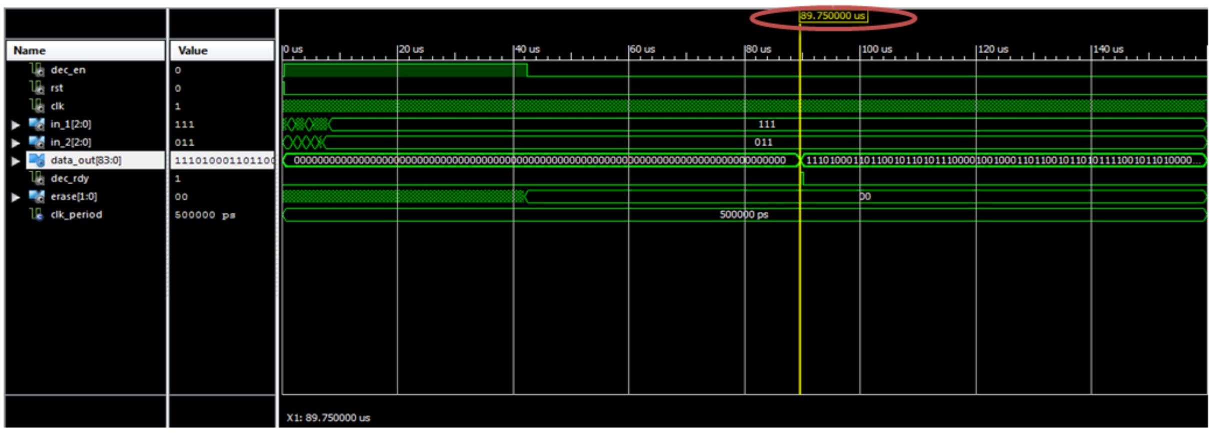


Fig. 14: Proposed Soft Viterbi Decoder of DVB-T System Output Result.

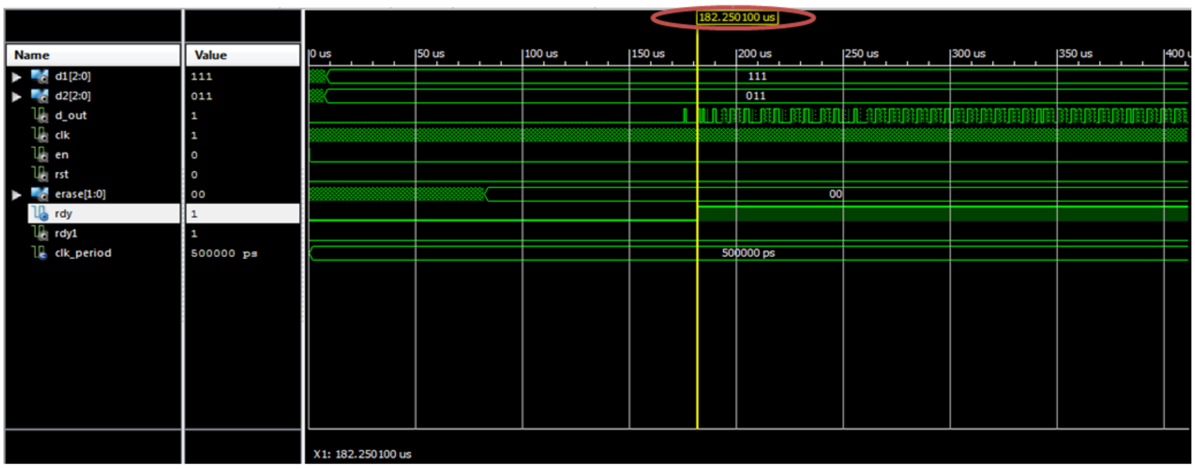


Fig. 15: Xilinx Built-in Soft Viterbi Decoder Ipcore Results.

4.1.5. Logic design summary

The logic design summary is one of the important characteristics of the design that indicates the utilization of the available logic blocks in the used FPGA chip. DVB-T convolutional codec and interleaver are implemented in VHDL to encode and decode symbols for reliable communication. The device utilization summary of the whole inner coded system is listed in Table (2).the device utilization summary for the implemented soft Viterbi decoder is listed in Table (3) while that of the IPcore Viterbi decoder is listed in Table (4) for purpose of comparison. The whole design is implemented using virtex-6 FPGA xc6vlx240t kit.

Table 2: Device Utilization Summary for the Whole System

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of slice registers	32899	301440	10%
Number of slice LUTs	15202	150720	10%
Number of fully used LUT-FF pairs	8887	39214	22%
Number of bounded IOBs	102	400	25%
Number of BUFG/BUFGCTRLs	2	32	6%

Table 3: Device Utilization Summary for the Implemented Soft Viterbi Decoder

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of slice registers	19820	301440	6%
Number of slice LUTs	7550	150720	5%
Number of fully used LUT-FF pairs	2044	25326	8%
Number of bounded IOBs	96	400	24%
Number of BUFG/BUFGCTRLs	2	32	6%

Table 4: Device Utilization Summary for the Ipcore Viterbi Decoder

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of slice registers	1689	301440	0%
Number of slice LUTs	2361	150720	1%
Number of fully used LUT-FF pairs	876	3174	27%
Number of bounded IOBs	14	400	3%
Number of Block RAM/FIFO	2	416	0%
Number of BUFG/BUFGCTRLs	1	32	3%

4.1.6. Timing summary

The timing report in Table (5) containing the timing summary of the design that completes the characterization of the logic design. Here, the DVB-T system inner codec that consists of puncturing convolutional codec and interleaver are implemented in VHDL. The timing summary for our soft Viterbi decoder, IPcore soft Viterbi decoder and that of the whole implemented system, synthesized on Virtex-6 (xc6vlx240t) are shown in Table (5). It is clear that our soft Viterbi decoder has appreciably lower speed than that of the IPcore soft Viterbi decoder. This is an expected result since the lower speed of our design is on the cost of less hardware.

Table 5: Timing Summary for the Implemented Soft Viterbi Decoder, Ipcore Soft Viterbi Decoder, and the Whole System

Time summary	our soft viterbi decoder	IPcore soft viterbi decoder	The whole system
Speed grade	-3	-3	-3
Minimum period	8.277ns	2.415ns	8.277ns
Maximum frequency	120.824MHz	414.14MHz	120.82224MHz

5. Conclusion

In this paper inner convolutional encoder with puncturing, inner interleaver, inner deinterleaver and soft Viterbi decoder with a constraint length of 7 and a code rate puncturing of 2/3 are presented. The designs are functionally verified using Xilinx ISE Design Suite 12.4 environments. Then the code is synthesized on Virtex-6(xc6vlx240t) FPGA.The synthesis results show that the FPGA implementation can run with frequency up to 120.824MHz. The speed of our design is much smaller than that of the IP core while it consumes much less area of the FPGA chip. So, in our design, the speed is traded for a leass area.

References

[1] U. W. E. Ladebusch and C. A. Liss, "Terrestrial DVB (DVB-T): A Broadcast Technology for Stationary Portable and Mobile Use," vol. 94, no. 1, pp. 183–193, 2006.

[2] V. Nostrand, T. T. Kadota, U. Grenander, J. R. Ragazzini, J. H. Laning, and R. H. Battin, "$ b (t) - ml (t) 1 dt Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," pp. 260–269, 1967.

[3] W.Fischer, Digital Video and Audio Broadcasting Technology. A Practical Engineering Guide_3rd ed _Springer. 2010. https://doi.org/10.1007/978-3-642-11612-4.

[4] B. K. Sudharani, B. Dhananjay, J. Praveen, and R. R. A, "Efficient Convolutional Adaptive Viterbi Encoder and Decoder Using RTL Design," vol. 3, no. 2, pp. 101–105, 2015.

[5] K. Rajendar and K. Bapayya, "FPGA Implementation of Efficient Viterbi Decoder for Multi-Carrier Systems," vol. 4, pp. 52–59, 2014.

[6] Y. Sun, "FPGA Design and Implementation of a Convolutional Encoder and a Viterbi Decoder Based on 802.11a for OFDM," Wirel. Eng. Technol., vol. 3, no. 3, pp. 125–131, 2012. https://doi.org/10.4236/wet.2012.33019.

[7] Y. M. Sandesh and K. Rambabu, "Implementation of Convolution Encoder and Viterbi Decoder for Constraint Length 7 and Bit Rate 1 / 2," Int. J. Eng. Res. Appl., vol. 3, no. 6, pp. 42–46, 2013.

[8] S. Mishra and R. R. Tripathi, "VDHL Implementation of Viterbi Algorithm for Decoding of Convolutional Code," Proc. - 2015 Int. Conf. Comput. Intell. Commun. Networks, CICN 2015, no. 1111001, pp. 1367–1370, 2016.

[9] V. Kavinilavu, S. Salivahanan, V. S. K. Bhaaskaran, S. Sakthikumaran, B. Brindha, and C. Vinoth, "Implementation of convolutional encoder and Viterbi decoder using Verilog HDL," ICECT 2011 - 2011 3rd Int. Conf. Electron. Comput. Technol., vol. 1, pp. 297–300, 2011. https://doi.org/10.1109/ICECTECH.2011.5941609.

[10] R. Green, "Forward Error Correction in Digital Television Broadcast," vol. 270, pp. 1–25, 2008.

[11] E. B. Union, "En 300 744," vol. 2, pp. 1–47, 1997.

[12] M. Elsharief, "Implementing a Standard DVB-T System using MATLAB Simulink," vol. 98, no. 5, pp. 27–32, 2014.