in the name of God

FPGA Project
Phase2
Encoding & Decoding of LAN 802.11a-1999
Sharif University of Technology

Name: Sajjad Akherati
id : 96101154

June 24, 2021

1 Introduction

In this part of the project, we are going to implement the encoder and decoder part of the standard LAN 802.11a-1999 for transmitter ans the receiver respectly. first, lets look at the Rate-dependent parameters table in the standard:

Data rate (Mbits/s)	Modulation	Coding rate (R)	Coded bits per subcarrier (N_{BPSC})	Coded bits per OFDM symbol (N_{CBPS})	Data bits per OFDM symbol (N_{DBPS})
6	BPSK	1/2	1	48	24
9	BPSK	3/4	1	48	36
12	QPSK	1/2	2	96	48
18	QPSK	3/4	2	96	72
24	16-QAM	1/2	4	192	96
36	16-QAM	3/4	4	192	144
48	64-QAM	2/3	6	288	192
54	64-QAM	3/4	6	288	216

Figure 1: rate dependent parameters

As we see in standard, the data rate of 6, 12, 24 MHz shall be suppurted and notice to their corresponding $R = \frac{1}{2}$ value, we don't need to consider implementation of puncturing in these modes. **Although i have implement them for both decoder and encoder in MATLAB.** So, in thid phase of the project was be to focuse on implementing the convlutional encoder and viterbi decoder for $R = \frac{1}{2}$.

2 Encoder

In this section i have designed a convlutional encoder with generator polynomials $g_0 = 133_g$ & $g_1 = 171_g$ with $R = \frac{1}{2}$ in verilog HDL.

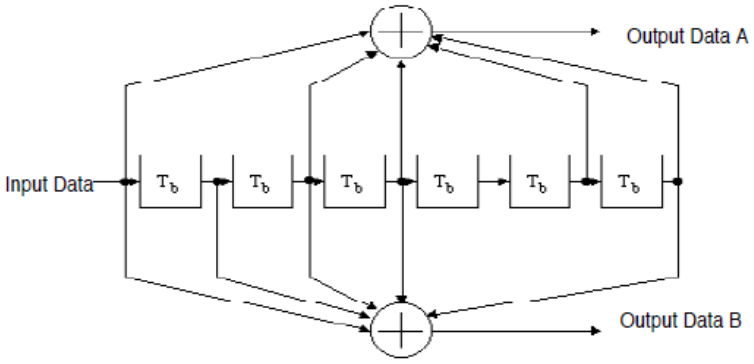

Figure 2: Convlutional Encoder ($k = 7$)

I also implement the convlutional encoder for $R = \frac{2}{3}$ & $R = \frac{3}{4}$ with enabled puncturing mood in verilog HDL and MATLAB. the verilog HDL code for this section can be found in *Verilog/ Encoder* directory named *convlutiona-encoder-puncturing*.
you can see the amount of recourse used in my design for encoder part in the following figures.

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	8	54,576	1%	
Number used as Flip Flops	8			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	5	27,288	1%	
Number used as logic	3	27,288	1%	
Number using O6 output only	3			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used as ROM	0			
Number of bonded IOBs	8	218	3%	
Number of RAMB16BWERs	0	116	0%	
Number of RAMB8BWERs	0	232	0%	
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%	
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%	
Number of BUFG/BUFGMUXs	1	16	6%	
Number used as BUFGs	1			
Number used as BUFGMUX	0			
Number of DCM/DCM_CLKGENs	0	8	0%	
Number of ILOGIC2/ISERDES2s	0	376	0%	
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	376	0%	
Number of OLOGIC2/OSERDES2s	0	376	0%	
Number of BSCANs	0	4	0%	
Number of BUFHs	0	256	0%	
Number used as Memory	0	6,408	0%	
Number used exclusively as route-thrus	2			
Number with same-slice register load	2			
Number with same-slice carry load	0			
Number with other load	0			
Number of occupied Slices	3	6,822	1%	
Number of MUXCYs used	0	13,644	0%	
Number of LUT Flip Flop pairs used	7			
Number with an unused Flip Flop	1	7	14%	
Number with an unused LUT	2	7	28%	
Number of fully used LUT-FF pairs	4	7	57%	
Number of unique control sets	2			
Number of slice register sites lost to control set restrictions	8	54,576	1%	
Number of BUFHs	0	256	0%	
Number of BUFPLLs	0	8	0%	
Number of BUFPLL_MCBs	0	4	0%	
Number of DSP48A1s	0	58	0%	
Number of ICAPs	0	1	0%	
Number of MCBs	0	2	0%	
Number of PCILOGICSEs	0	2	0%	
Number of PLL_ADVs	0	4	0%	
Number of PMVs	0	1	0%	
Number of STARTUPs	0	1	0%	
Number of SUSPEND_SYNCs	0	1	0%	
Average Fanout of Non-Clock Nets	2.69			

Figure 3: Recourse Used By Encoder

3 Decoder

I this part of the project, our goal is to decode the received data by a viterbi decoder and remove the noise that has been occured on data when it has been transmitted.
the decoding is done using a viterbi decoder; you can see the structure of a viterbi decoder in the following figure:

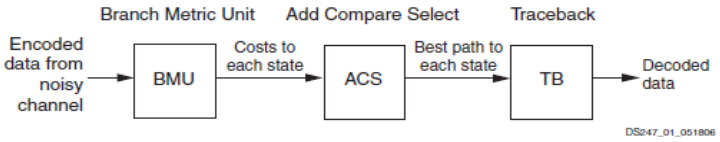

Figure 4: Viterbi Decoder Block Diagram

The viterbi decoder uses a dynamic programming approach to decode data.
In BMU, we calculate the costs to each state and in the ACS we choose the best path for each state. we do this procedure for a detremined depth and at the end using the TB, we found the closes decoded data to the main data. in this project, i have implemented the viterbi decoder for depth = 36 and also depth = 48; i decoded the datas for depth = 48, because the length of a frame is a multiple of 48. the decoding in puncturing mood for $R = \frac{2}{3}$ and $R = \frac{3}{4}$ has been implemented in MATLAB, too.

Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	3,476	54,576	6%	
Number used as Flip Flops	3,476			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	4,309	27,288	15%	
Number used as logic	3,701	27,288	13%	
Number using O6 output only	2,863			
Number using O5 output only	640			
Number using O5 and O6	198			
Number used as ROM	0			
Number used as Memory	416	6,408	6%	
Number used as Dual Port RAM	0			
Number of fully used LUT-FF pairs	2,419	4,825	50%	
Number of unique control sets	11			
Number of slice register sites lost to control set restrictions	28	54,576	1%	
Number of bonded IOBs	5	218	2%	
Number of RAMB16BWERs	0	116	0%	
Number of RAMB8BWERs	0	232	0%	
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%	
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%	
Number of BUFG/BUFGMUXs	1	16	6%	
Number used as BUFGs	1			
Number used as BUFGMUX	0			
Number of DCM/DCM_CLKGENs	0	8	0%	
Number of ILOGIC2/ISERDES2s	0	376	0%	
Number used as Single Port RAM	0			
Number used as Shift Register	416			
Number using O6 output only	0			
Number using O5 output only	0			
Number using O5 and O6	416			
Number used exclusively as route-thrus	192			
Number with same-slice register load	64			
Number with same-slice carry load	128			
Number with other load	0			
Number of occupied Slices	1,458	6,822	21%	
Number of MUXCYs used	1,024	13,644	7%	
Number of LUT Flip Flop pairs used	4,825			
Number with an unused Flip Flop	1,890	4,825	39%	
Number with an unused LUT	516	4,825	10%	
Number of OLOGIC2/OSERDES2s	0	376	0%	
Number of BSCANs	0	4	0%	
Number of BUFHs	0	256	0%	
Number of BUFPLLs	0	8	0%	
Number of BUFPLL_MCBs	0	4	0%	
Number of DSP48A1s	0	58	0%	
Number of ICAPs	0	1	0%	
Number of MCBs	0	2	0%	
Number of PCILOGICSEs	0	2	0%	
Number of PLL_ADVs	0	4	0%	
Number of PMVs	0	1	0%	
Number of STARTUPs	0	1	0%	
Number of SUSPEND_SYNCs	0	1	0%	
Average Fanout of Non-Clock Nets	3.47			

Figure 5: Recourse Used By Decoder

At the end i announce some features about designed decoder:

1. this decoder hsa a latency of 75 clock cycle. it means that when we receive the first bit of the frame, it will be decode in 75 clock cycle later.
2. the BUM and ACS blcok in my system are fully parallel.

3. the TB in my system is serial.

4 Results

In this part, to see the result of implemented design I simulated an assumed data, in MATLAB and ISIM and compared them:

1. first I coded the scrambled frame from phase1 in both MATLAB and HDL.
2. then I add some noise on data in a simulated serial link in MATLAB.
3. then I decoded the frame in MATLAB and verilog and compared the results.

you can refer to the following figures for more details:

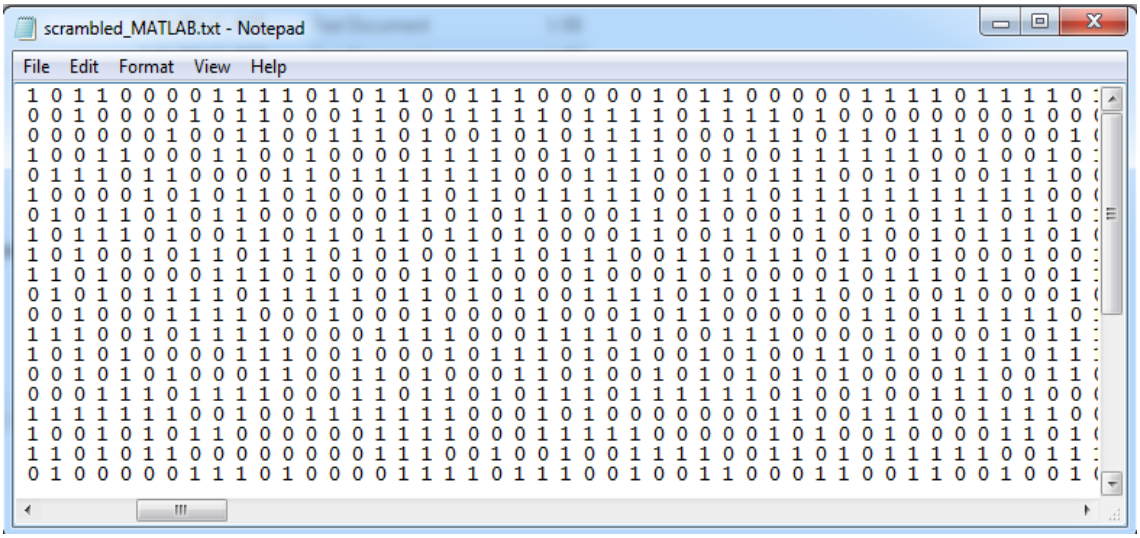

Figure 6: The Main Scrambled Frame

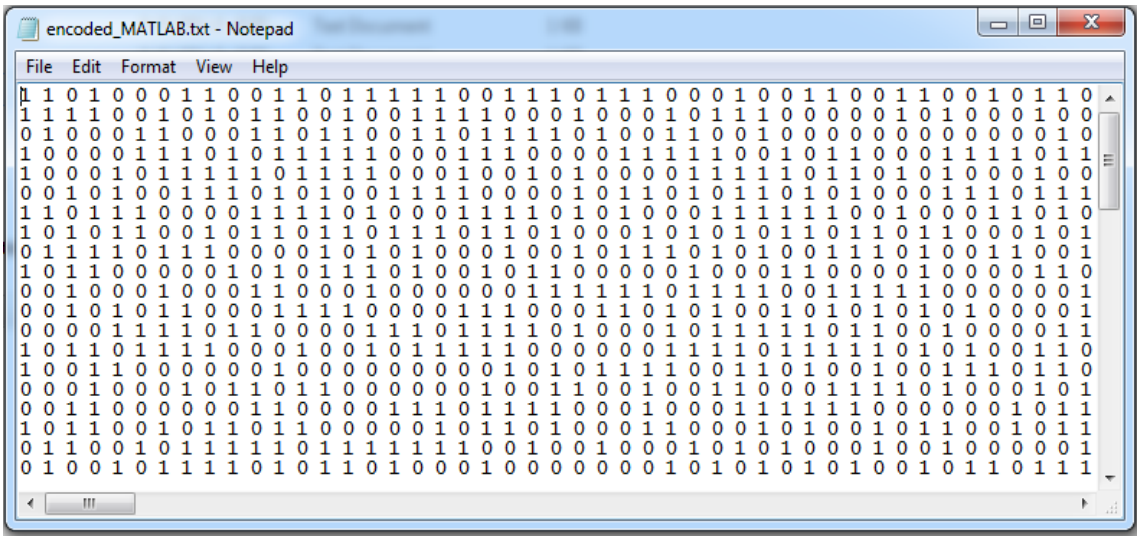

Figure 7: Encoded Frame MATLAB

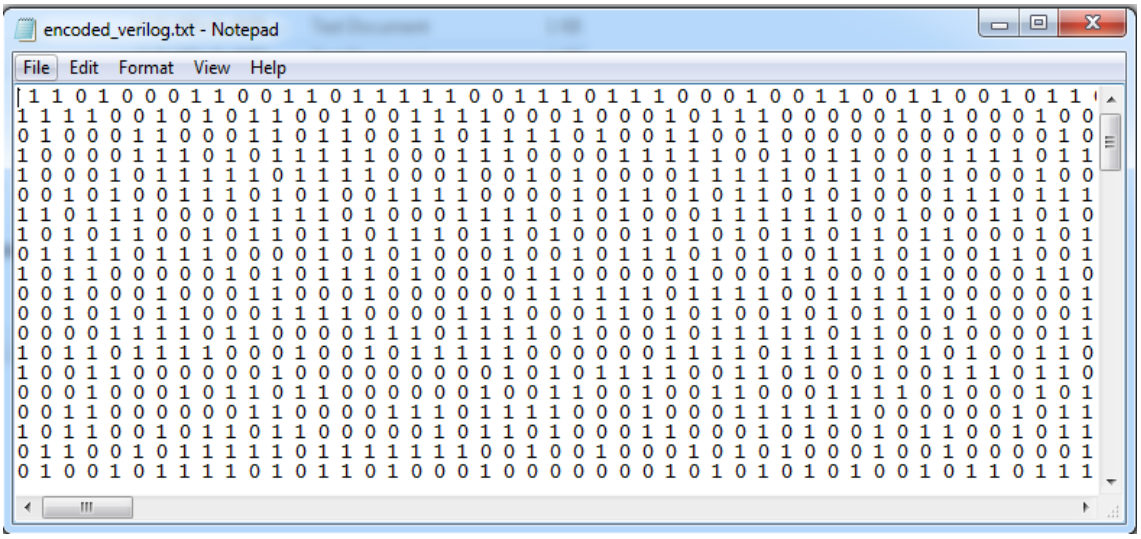

Figure 8: Encoded Frame Verilog

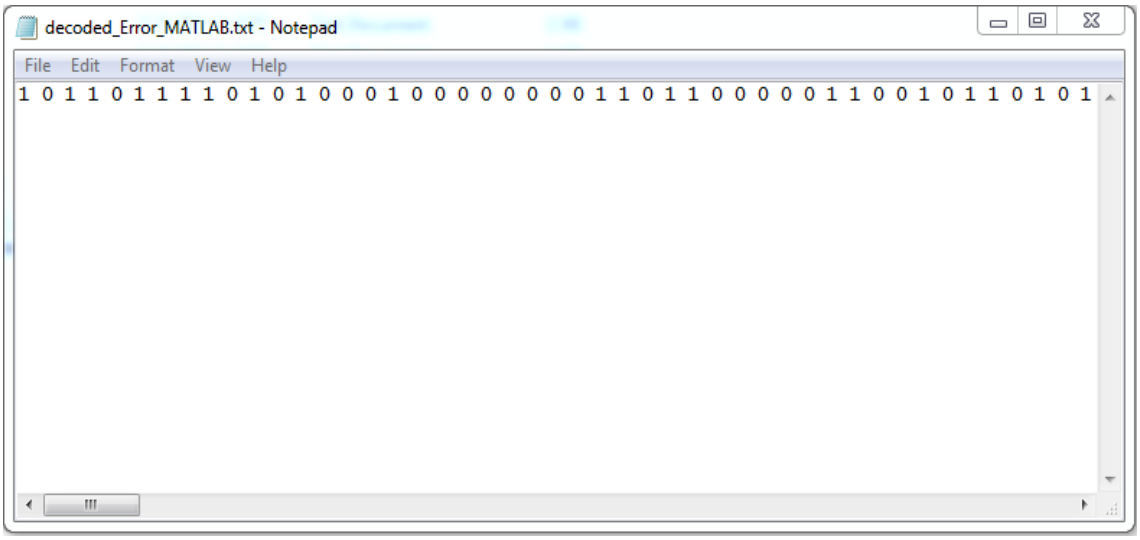

Figure 9: Decoded Frame MATLAB

for mor explanation about the above figure, my MATLAB code is serial in BMU and ACS and is not as fast as the verilog one. so i did the decoding of 480 bits from the main frame ad the decoding has not be done completely.

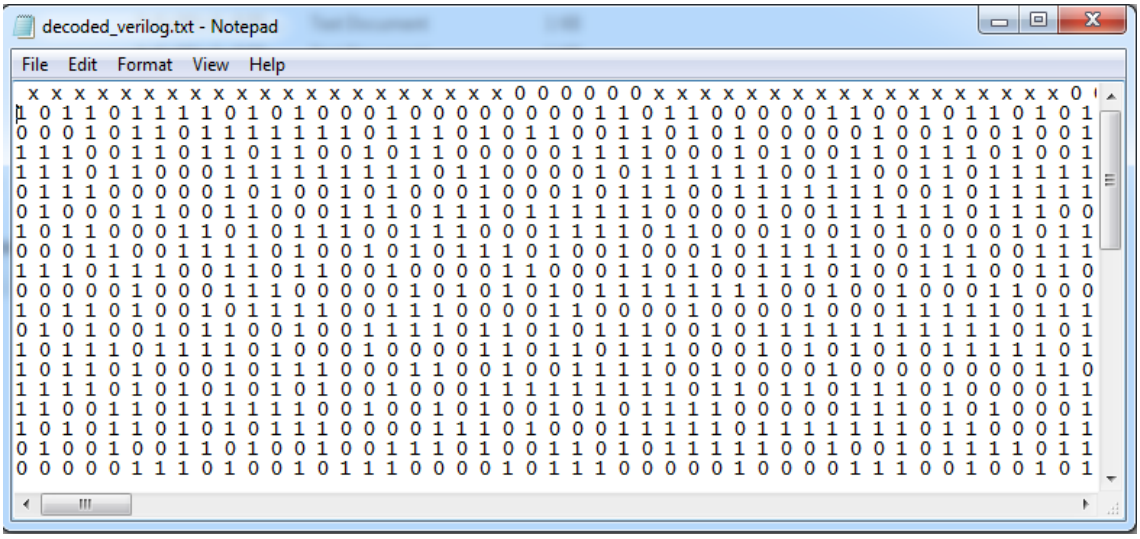

Figure 10: Decoded Frame Verilog

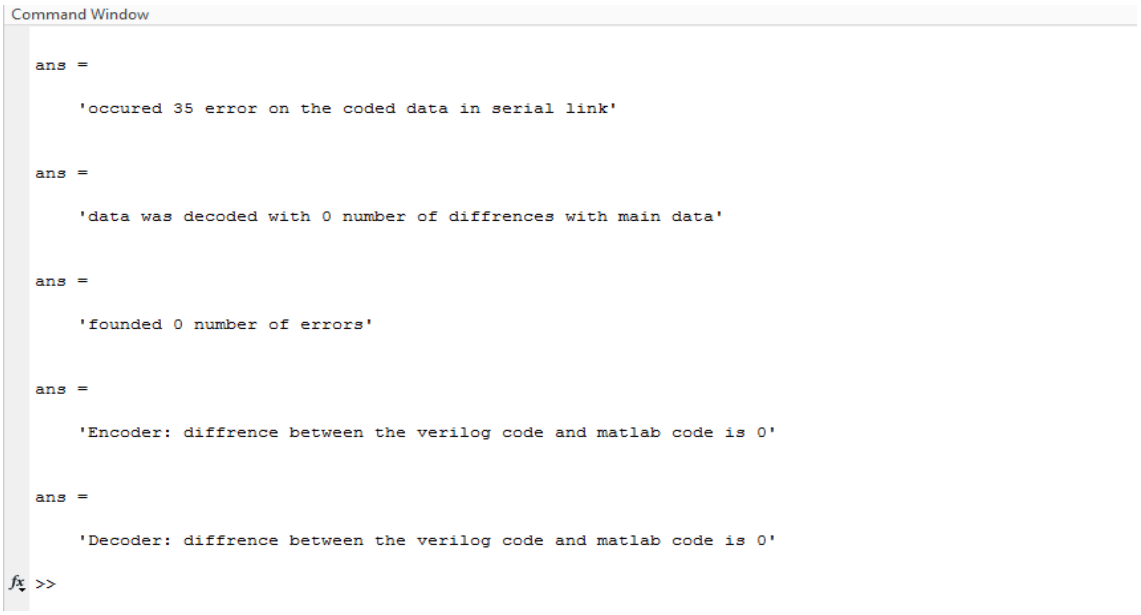

Figure 11: Comparison

from the above figure, we can see that 35 number of error in the frame has been detected and corrected.