in the name of God

FPGA Project

Phase3

Data Interleaving & Data Deinteleaving of LAN 802.11a-1999

Sharif University of Technology

Name: Sajjad Akherati
id : 96101154

July 9, 2021

1 Introduction

In this phase of the project, we are going to implement data interleaver for transmitter side and data deinterlraver for receiver side of LAN 802.11a-1999.
for more explanation, data interleaving is done due to convlutional decoder and the excisting errors on the physical layers in data networks. the error in this communications are type burst errors means that the convlutional encoder can not decode data if this burst error be too much. this is because that the convlutional encoder is a memorized coder that mens every output from the current input is dependent to this input and the past ones and if some bits of data that are continuos together become noisy (burst noise) the viterbi decoder is not able to decode data. by the trick of data interleaving, we can pass this problem and use the perfomance of the convlutional encoder and decoder in our system.

2 Data Interleaver (TX Side)

All encoded data bits shall be interleaved by a block interleaver with a block size corresponding to the number of bits in a single OFDM symbol, NCBPS. The interleaver is defined by a two-step permutation. The first permutation ensures that adjacent coded bits are mapped onto nonadjacent subcarriers. The second ensures that adjacent coded bits are mapped alternately onto less and more significant bits of the constellation and, thereby, long runs of low reliability (LSB) bits are avoided.
We shall denote by k the index of the coded bit before the first permutation; i shall be the index after the first and before the second permutation, and j shall be the index after the second permutation, just prior to modulation mapping. The first permutation is defined by the rule:

$$i = \frac{N_{CBPS}}{16}(k\%16) + [\frac{k}{16}] \qquad k = 0, 1, \ldots, N_{CBPS} - 1$$

the second permution is defined by the rule:

$$j = s \times [\frac{i}{s}] + (i + N_{CBPS} - [\frac{16 \times i}{N_{CBPS}}])\%s \qquad i = 0, 1, \ldots, N_{CBPS} - 1$$

and the parameter s is defined by the following rule:

$$s = \max(\frac{N_{BPSC}}{2}, 1)$$

notice to the rate depenedent parameters table we obtaine that N_{CBPS} is equal to 48, 96, 192 when rate is 6, 12, 24 MHz and also the parameter N_{BPSC} is equal to 1, 2, 4 means that s is equal to 1, 1, 2; notice to the above approches and using some simplification on above formulas to implement in a hardware system, i used some FIFO and counter to implement this part of the project:

Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	415	54,576	1%	
Number used as Flip Flops	415			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	1,789	27,288	6%	
Number used as logic	1,789	27,288	6%	
Number using O6 output only	1,769			
Number using O5 output only	0			
Number using O5 and O6	20			
Number used as ROM	0			
Number used as Memory	0	6,408	0%	
Number of occupied Slices	623	6,822	9%	
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 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 MUXCYs used	16	13,644	1%	
Number of LUT Flip Flop pairs used	1,789			
Number with an unused Flip Flop	1,378	1,789	77%	
Number with an unused LUT	0	1,789	0%	
Number of fully used LUT-FF pairs	411	1,789	22%	
Number of unique control sets	7			
Number of slice register sites lost to control set restrictions	25	54,576	1%	
Number of bonded IOBs	12	218	5%	
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 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	6.58			

Figure 1: Recourse Used By Data Interleaver

3 Data Deinterleaver (RX Side)

The deinterleaver, which performs the inverse relation, is also defined by two permutations.
Here we shall denote by j the index of the original received bit before the first permutation; i shall be the index after the first and before the second permutation, and k shall be the index after the second permutation, just prior to delivering the coded bits to the convolutional (Viterbi) decoder.
the first permution is defined by the rule:

$$i = s \times [\frac{j}{s}] + (j + [\frac{16 \times j}{N_{CBPS}}])\%s \qquad j = 0, 1, \ldots, N_{CBPS} - 1$$

s is the same as in data interleaver. the second permution is defined by the rule:

$$k = 16 \times i - (N_{CBPS} - 1)[\frac{16 \times i}{N_{CBPS}}] \qquad i = 0, 1, \ldots, N_{CBPS} - 1$$

As i said earlier, i don't use the exactly above formulas to implement desired hardware for interleave and deinterleaver. i checked out their functionalities and notice to what these formulas do on a frame of data, i found that i can implement the interleaver and its corresponding deinterleaver using some counters, some SRL or FIFO's, and piplining approches approches. so my design doesn't need to calulate any floor or product or division using DSP cores that waste the time and recources of the system. in the following figure, you can see the number of the recources used by my design:

Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	434	54,576	1%	
Number used as Flip Flops	434			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	718	27,288	2%	
Number used as logic	718	27,288	2%	
Number using O6 output only	706			
Number using O5 output only	0			
Number using O5 and O6	12			
Number used as ROM	0			
Number used as Memory	0	6,408	0%	
Number of occupied Slices	256	6,822	3%	
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 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 MUXCYs used	8	13,644	1%	
Number of LUT Flip Flop pairs used	725			
Number with an unused Flip Flop	292	725	40%	
Number with an unused LUT	7	725	1%	
Number of fully used LUT-FF pairs	426	725	58%	
Number of unique control sets	11			
Number of slice register sites lost to control set restrictions	30	54,576	1%	
Number of bonded IOBs	9	218	4%	
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 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	6.10			

Figure 2: Recourse Used By Data Deinterleaver

4 Results

in this part, i interleaved a frame was coded in previous phase in both MATLAB and Verilog and then compared their results. i also did it for deinterleaving modules. you can see the results of running them in MATLAB and Verilog in the following figures:

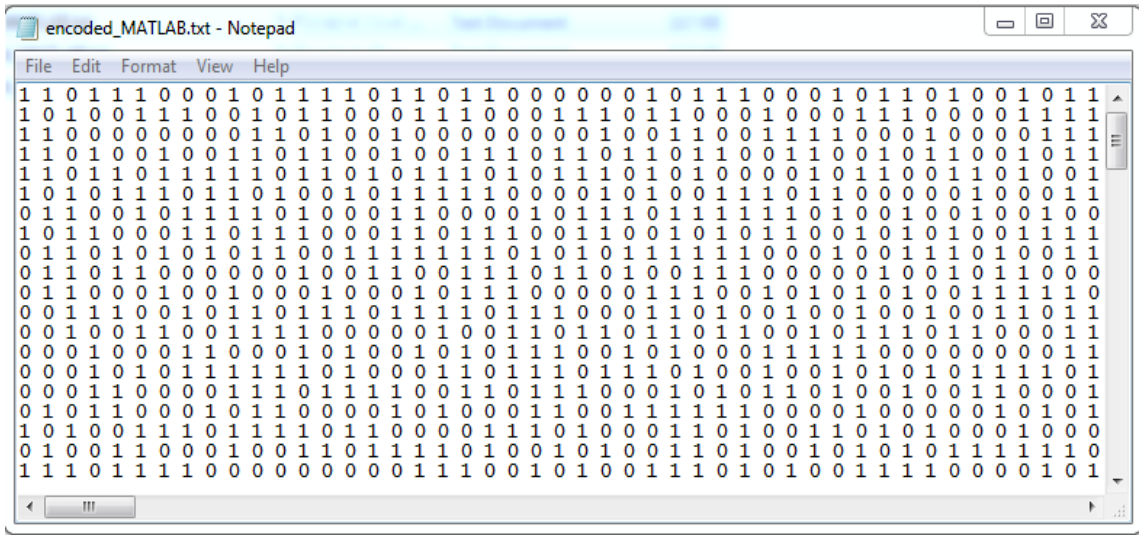

Figure 3: The Main Encoded Frame

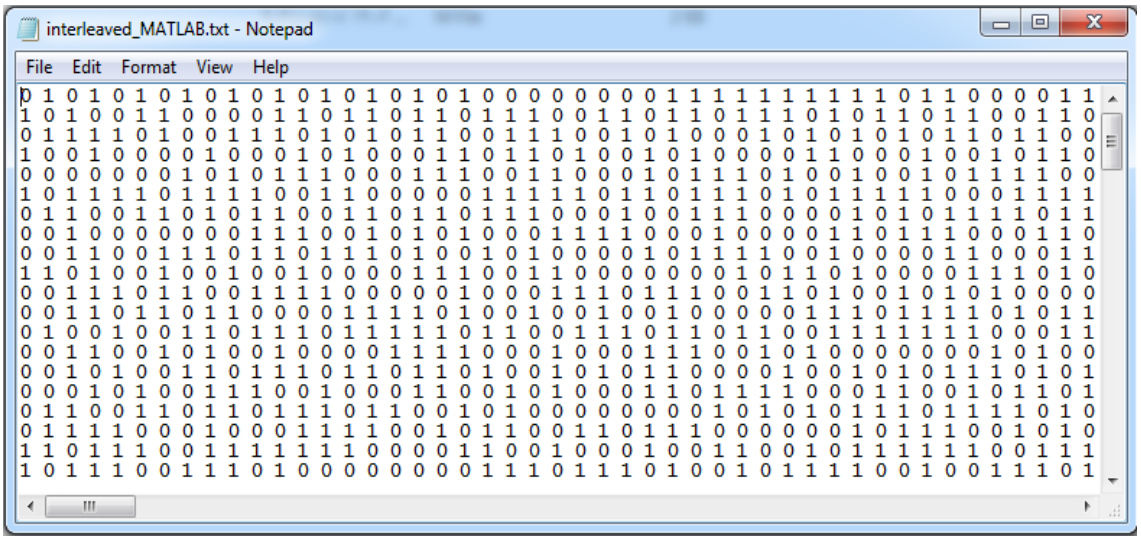

Figure 4: Interleaved Frame MATLAB

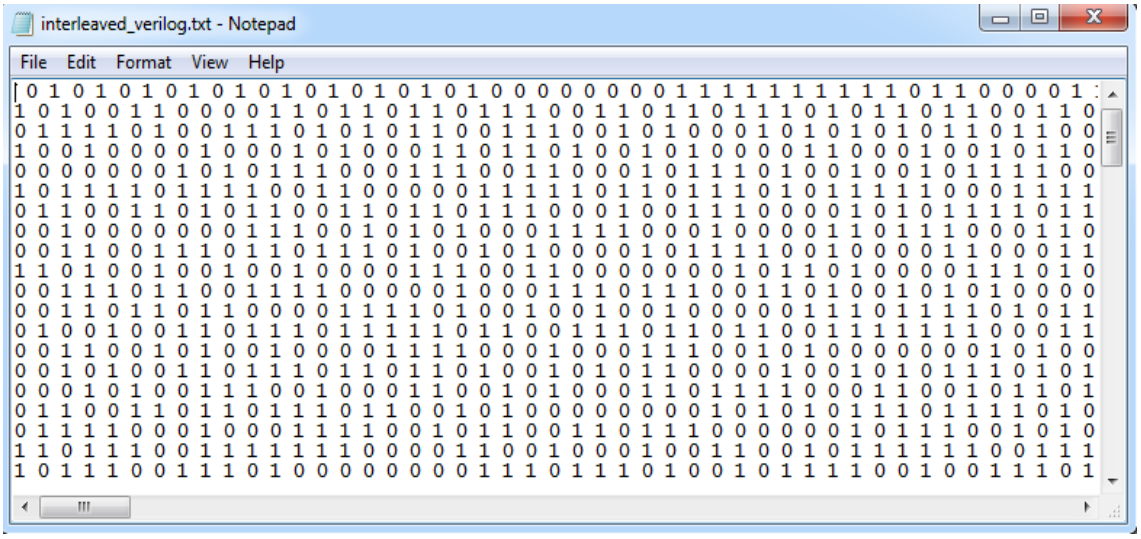

Figure 5: Interleaved Frame Verilog

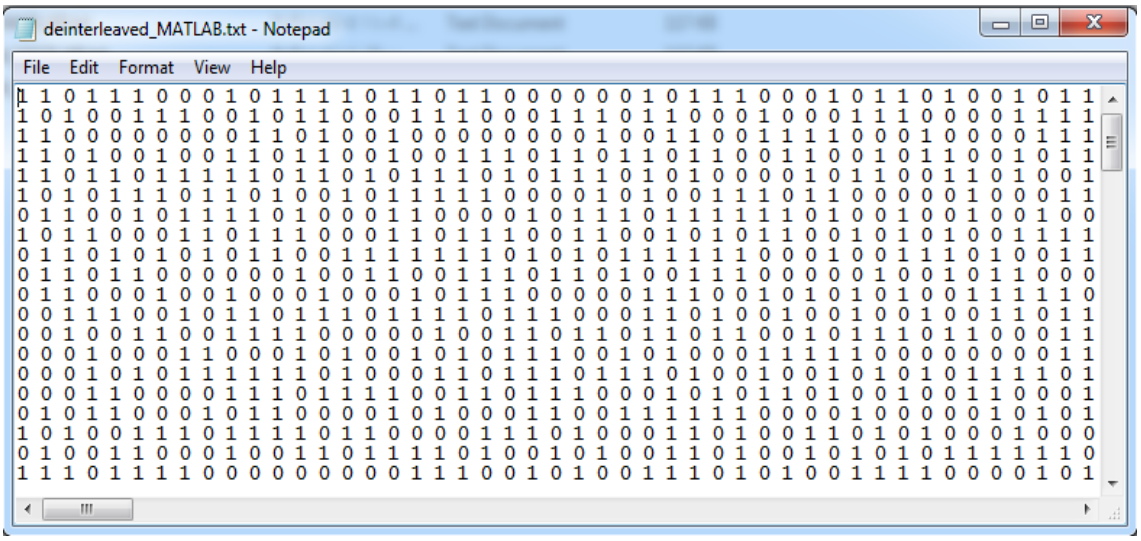

Figure 6: Deinterleaved Frame MATLAB

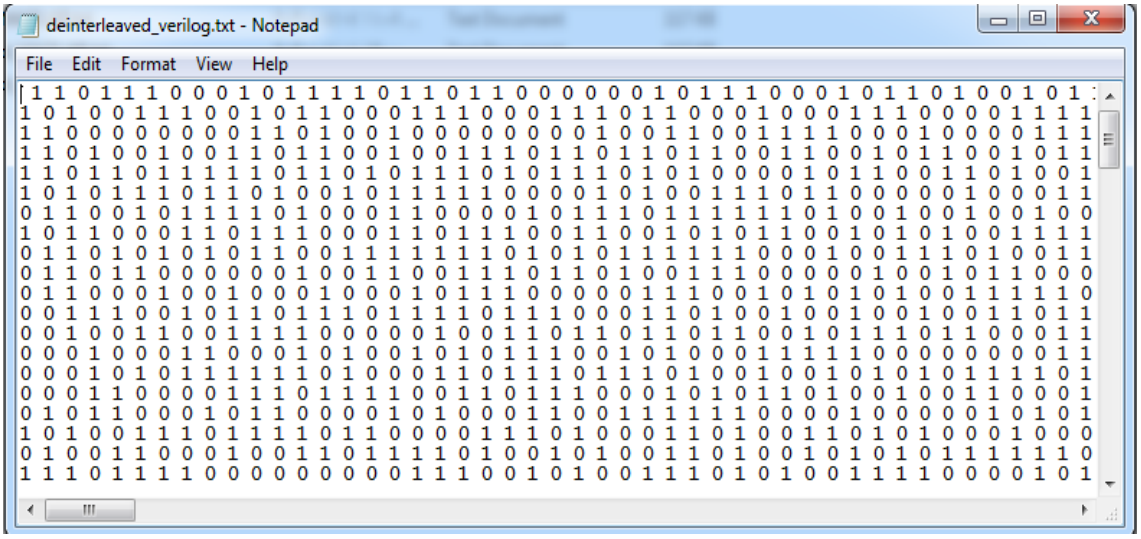

Figure 7: Deinterleaved Frame Verilog

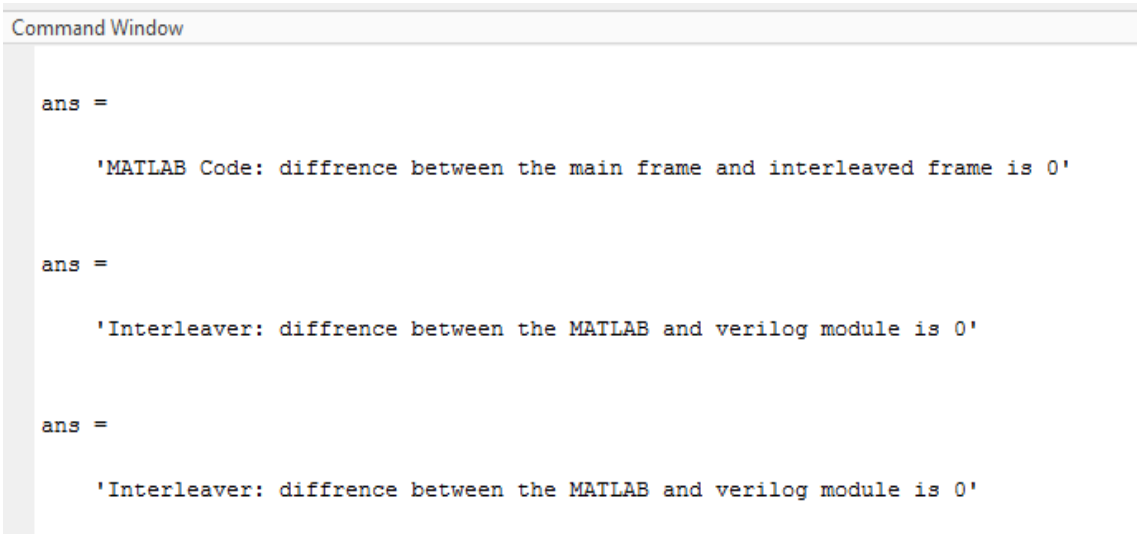

Figure 8: Compare the Result of MATLAB and Verilog in MATLAB

By the above figure, it is clear that the result from the MATLAB and Verilog code are matched together.