DAYANANDA SAGAR COLLEGE OF ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

Mini Project Work (18EC6ICMPR)

Presentation on

LDPC Decoding for FPGA Implementation using Min Sum Algorithm

By:

BATCH NO. - B6

1DS18EC091	Sudhamshu B N
1DS18EC093	Sumanth B P
1DS18EC094	Suprith N
1DS18EC075	Ramesh C S

Under the guidance of

Prof. R. Santosh Kumar

Assistant Professor, Dept. of ECE

CONTENTS

INTRODUCTION

Low Density Parity Check

Low-density

H is a sparse matrix (i.e. the number of ‘1’s is much lower than the number of '0's). It is the sparseness of H that guarantees the low computing complexity.

Parity-check

LDPC codes are represented by a parity-check matrix H, where H is a binary matrix that must satisfy cH’ = 0, where c is a codeword.

INTRODUCTION

Regular codes

The conditions to be satisfied in the construction of the parity-check matrix H of a binary regular LDPC code are:

• The corresponding parity-check matrix H should have a fixed column weight wc .

• The corresponding parity-check matrix H should have a fixed row weight wr .

• The number of "l"s between any two columns is no greater than 1.

• Both wc and wr should be small numbers compared to the code length n and the number of rows in H. Normally,

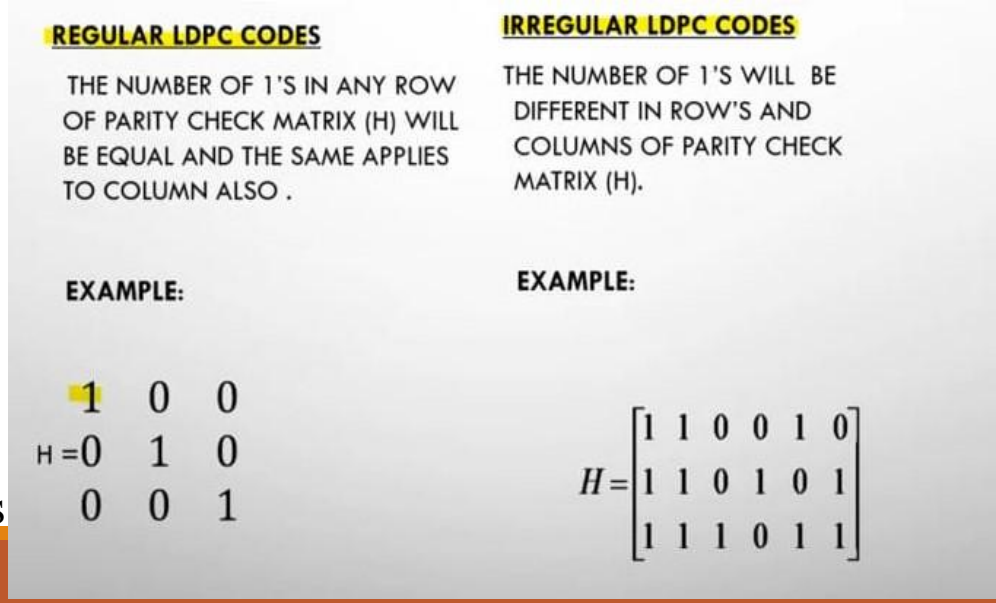
the code rate of LDPC codes is R = 1- (wc /wr).

Irregular codes

An irregular LDPC code has a parity-check matrix H that has a variable

wc or wr. In general, the bit error rate (BER)

performance of irregular LDPC codes is better than that of regular LDPC codes



INTRODUCTION

Encoding of LDPC Codes

Gauss Jordan Elimination

pT=B-AxT

Lower Triangular Based Encoding

Systematic Encoding

{c1, · · · , cN−M} = {u1, · · · , uN−M}

ci are recursively computed by using the lower-triangular shape:

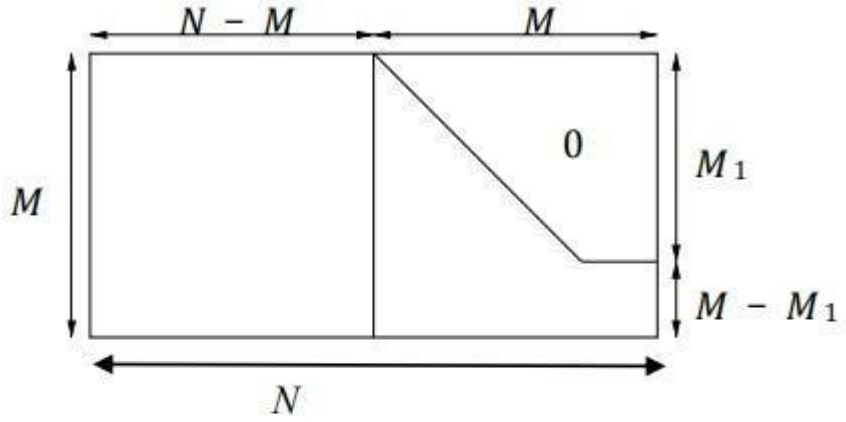
ci = −pci × (c1, · · · , ci−1)T, for i ϵ {N −M + 1, · · · ,N −M +M1}

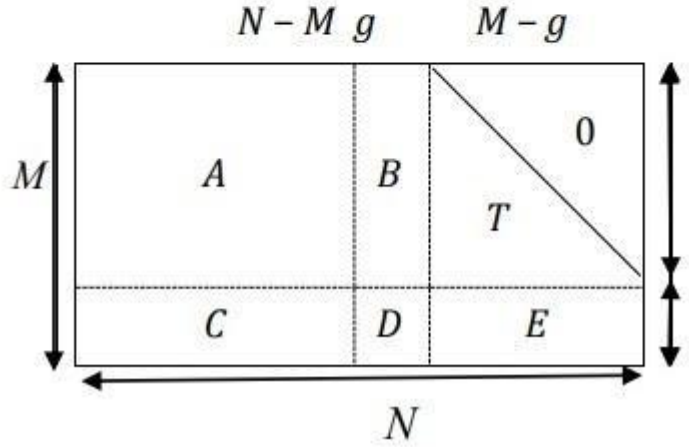
Other Techniques

Iterative Encoding

Low Density Generator Matrices

Cyclic Parity Matrices





cyclic parity technique

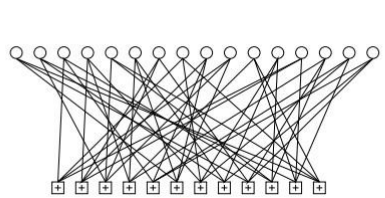
LITERATURE SURVEY

1. **Gallager Codes -Recent Results,David J.C. MacKay Cavendish Laboratory, Cambridge**

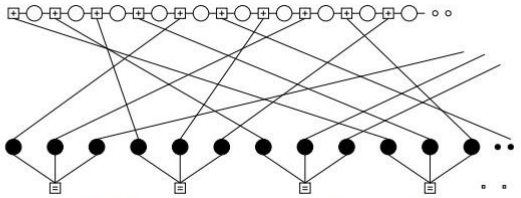
- This paper reviews low–density parity–check codes (Gallager codes).
- Also compare the ldpc codes with other codes like Turbo codes and Repeat-accumulate codes.
- Describing experiments on Gallager codes with small block lengths.
- Stopping rules for the decoding of sparse graph codes

2. **Efficient Encoding of Low-Density Parity-Check Codes,Thomas J. Richardson and Rüdiger L. Urbanke**

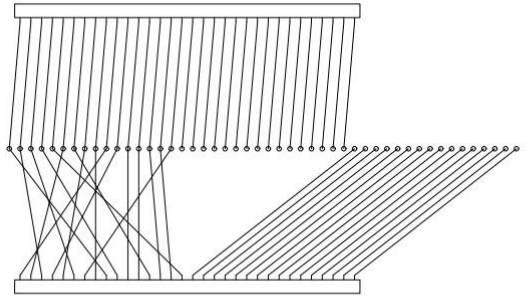
- In this paper, we consider the encoding problem for LDPC codes.
- This paper gave the idea of graphical representation of the regular ldpc codes.
- It also reviews the algorithms used in ldpc code encoding .



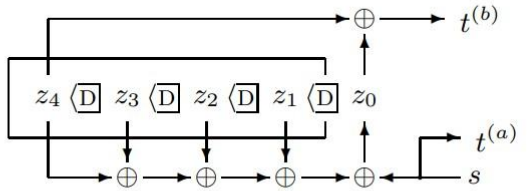
(a) Gallager code



(b) Repeat-accumulate code



(c1) Turbo code



(c2) $(21/37)_8$ recursive convolutional code

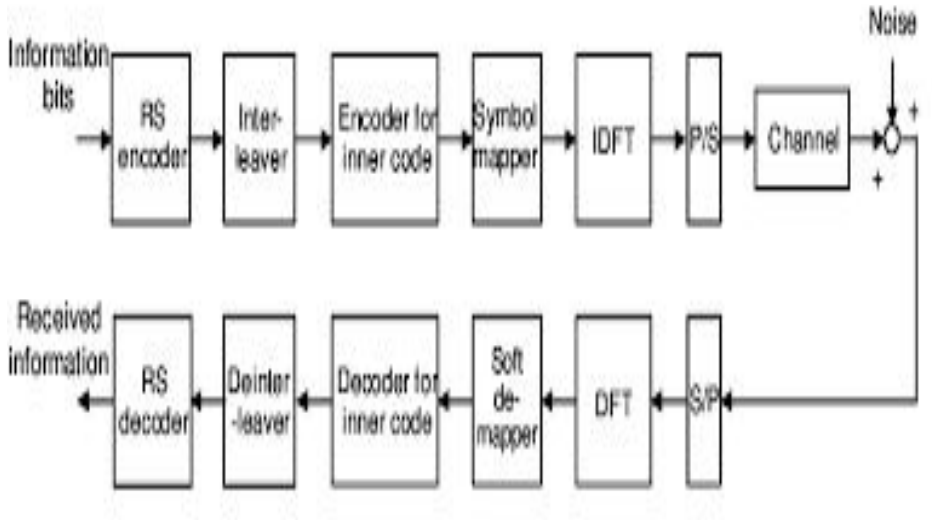
LITERATURE SURVEY

3. Parallel Decoding Architectures for Low Density Parity Check Codes, C.Howland and A. Blanksby

- This paper proposed a parallel architecture for decoding low density parity check codes .
- The feasibility of this architecture is demonstrated through implementation of 1024 bit,rate -½ ,soft decision parallel decoder.
- It also explains the message passing algorithm.

4.Low-Density Parity-Check Codes for Digital Subscriber Lines.E. Eleftheriou and S. Ölçer

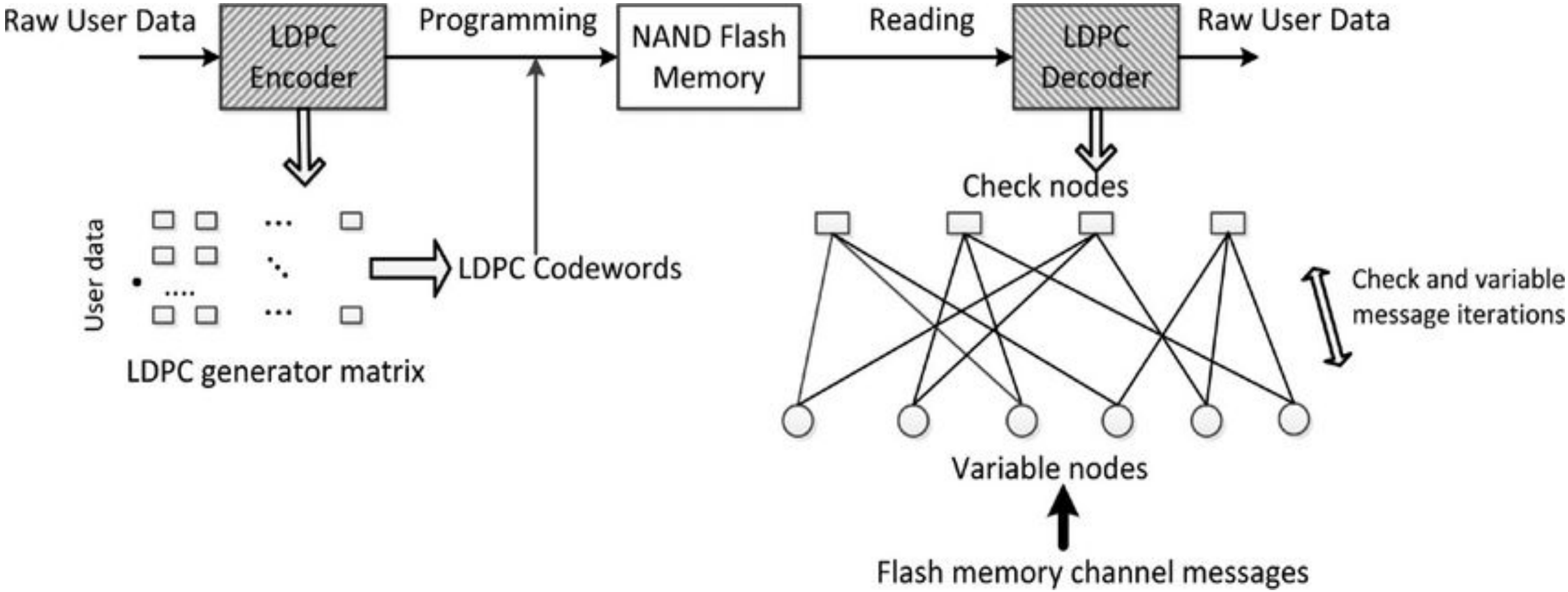
- The paper investigates the application of low-density parity-check (LDPC) codes to digital subscriber-line(DSL) transmission systems
- It also provides the bandwidth efficient LDPC -coded modulation.
- Implementation complexity is analyzed and compared with that of trellis-coded modulation as employed in current asymmetric DSL transceivers.



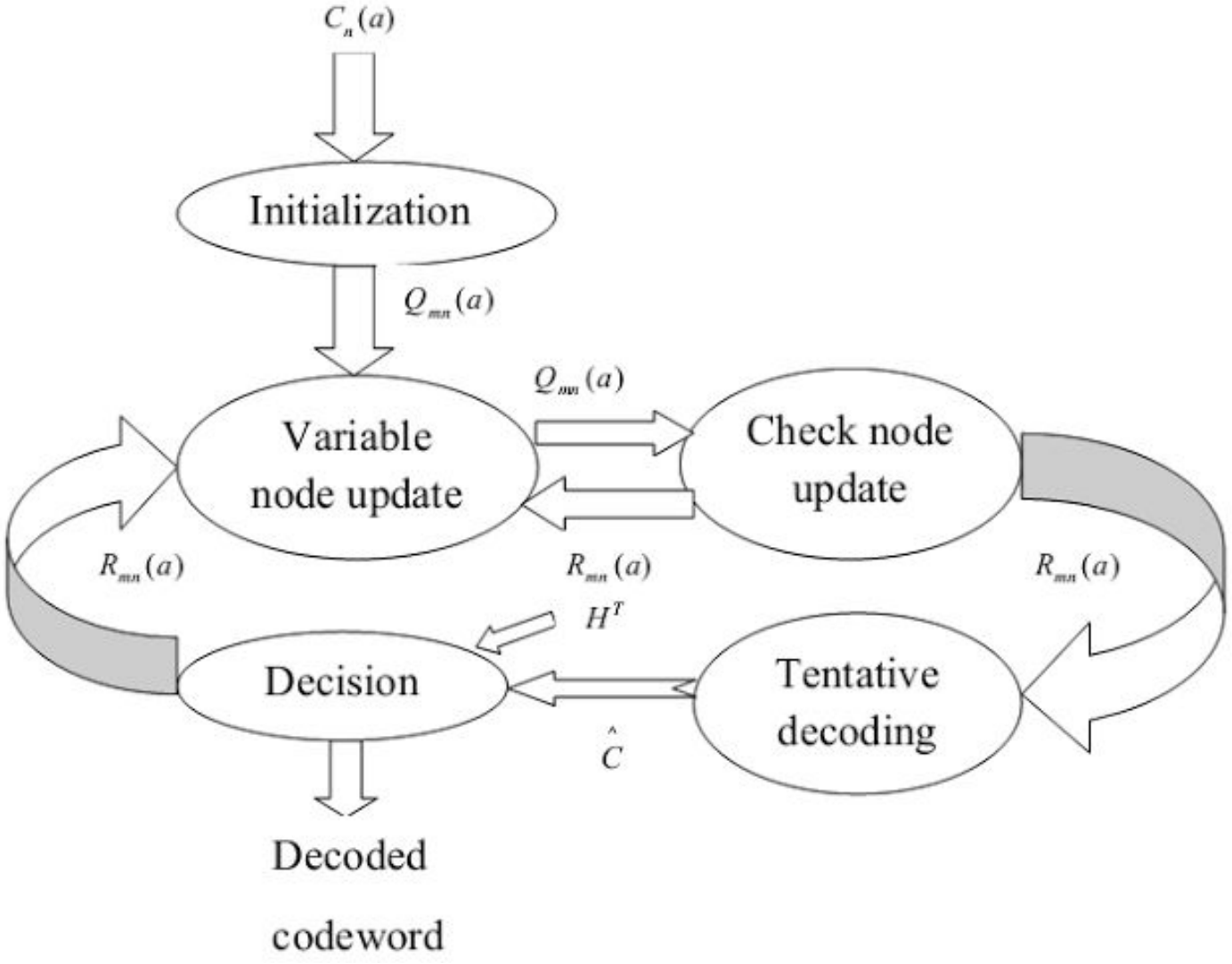
Methodology

- Use the min-sum algorithm with llr decoder algorithm
- Write Verilog code for the hardware implementation
- Use ModelSim to compile, simulate and view the waveforms.
- Use Quartus Prime for RTL Simulation
- Use DESim for DE-10 lite fpga implementation simulation
- Compare BER of Hard Decision Algorithm and Min Sum Algorithm using Matlab

BLOCK DIAGRAM



Min-Sum Algorithm



STEP 1 Initialization: Set $i = 0$ and the maximum number of iterations to I_{max}. Set $q_{vc} = y_v$.

STEP 2 Check node update: at check node $0 \leq c \leq \gamma m - 1$, for $0 \leq v < \rho m - 1$, compute $\sigma_{cv}^{(i)}$ by

$$\sigma_{cv} = \alpha \cdot \min_{v' \in \mathcal{N}(c) \backslash v} |q_{cv'}| \cdot \prod_{v' \in \mathcal{N}(c) \backslash v} \text{sign}\,(q_{cv'})$$

STEP 3 Variable node update: At variable node $0 \leq v < \rho m - 1$, for $0 \leq c \leq \gamma m - 1$, compute q_{vc} and z_v by

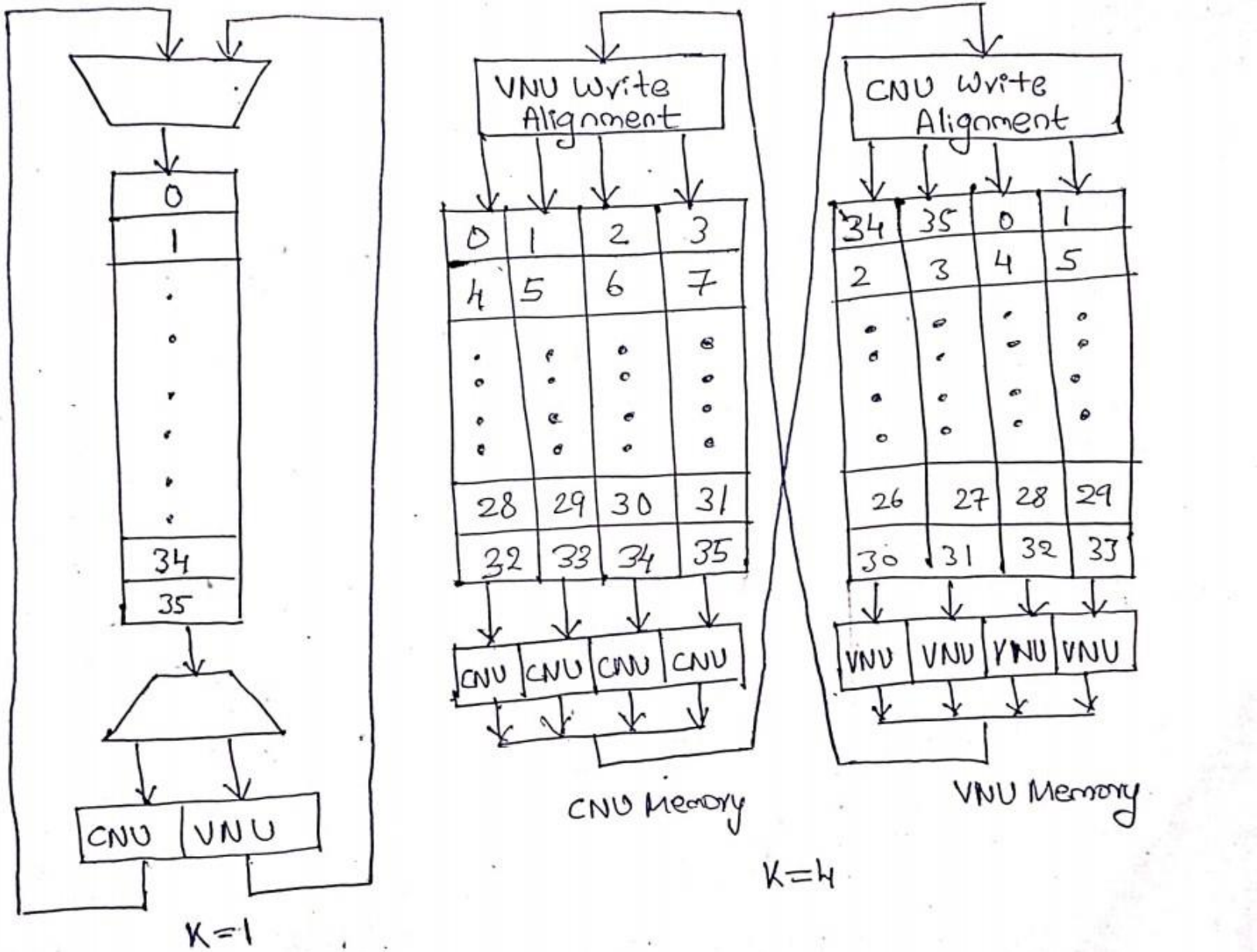
$$q_{vc} = y_v + \sum_{c' \in \mathcal{M}(v) \backslash c} \sigma_{c'v}$$

$$z_v = y_v + \sum_{c \in \mathcal{M}(v)} \sigma_{cv}$$

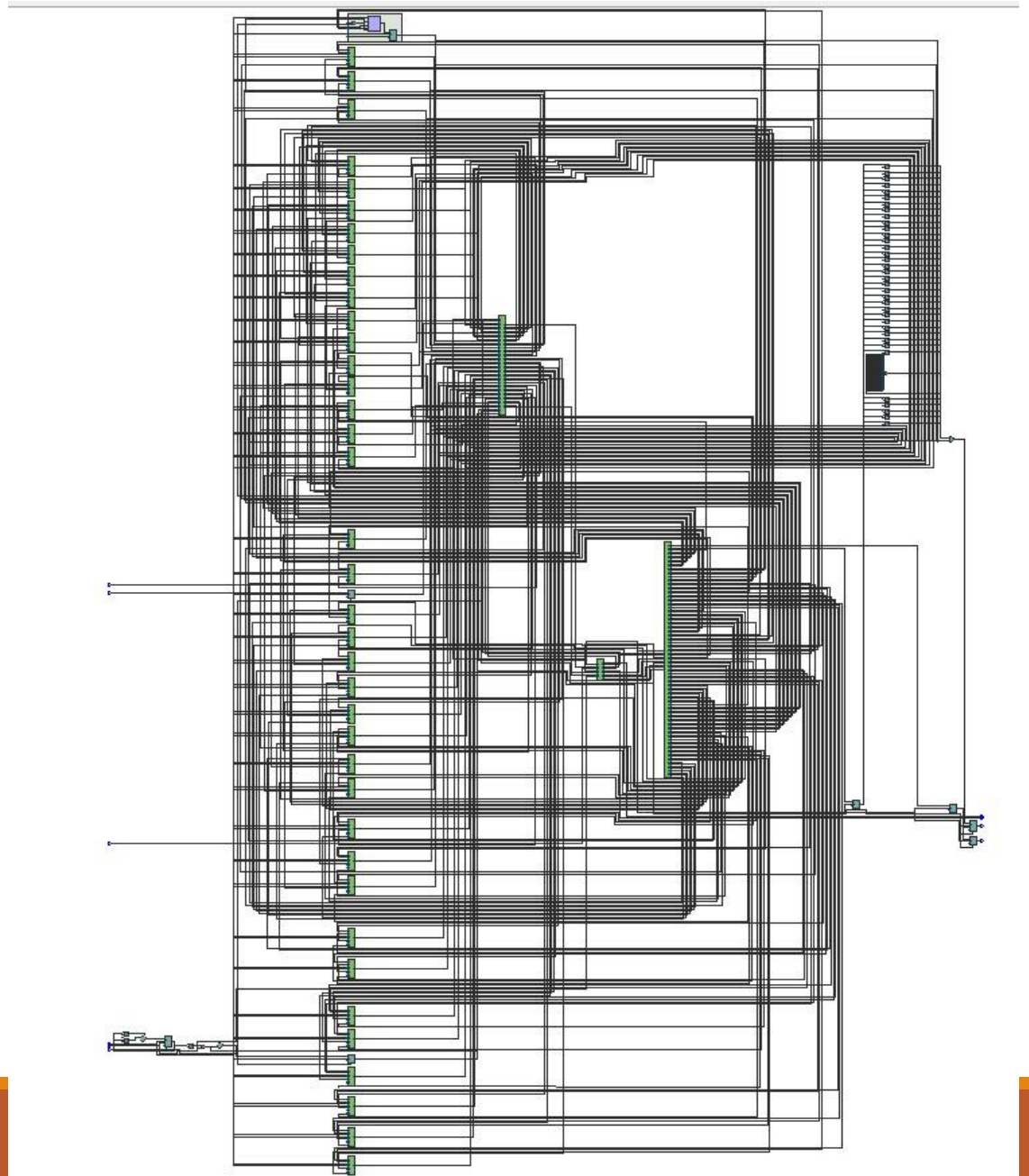
STEP 4 Tentative decode: If $\text{sign}\,(\mathbf{z}) \cdot \mathbf{H}^T = 0$ or I_{max} is reached, go to (5). Otherwise, $i \leftarrow i + 1$ and go to (2).

STEP 5 Termination: Take $\text{sign}\,(\mathbf{z})$ as the decoded codeword and stop the decoding process.

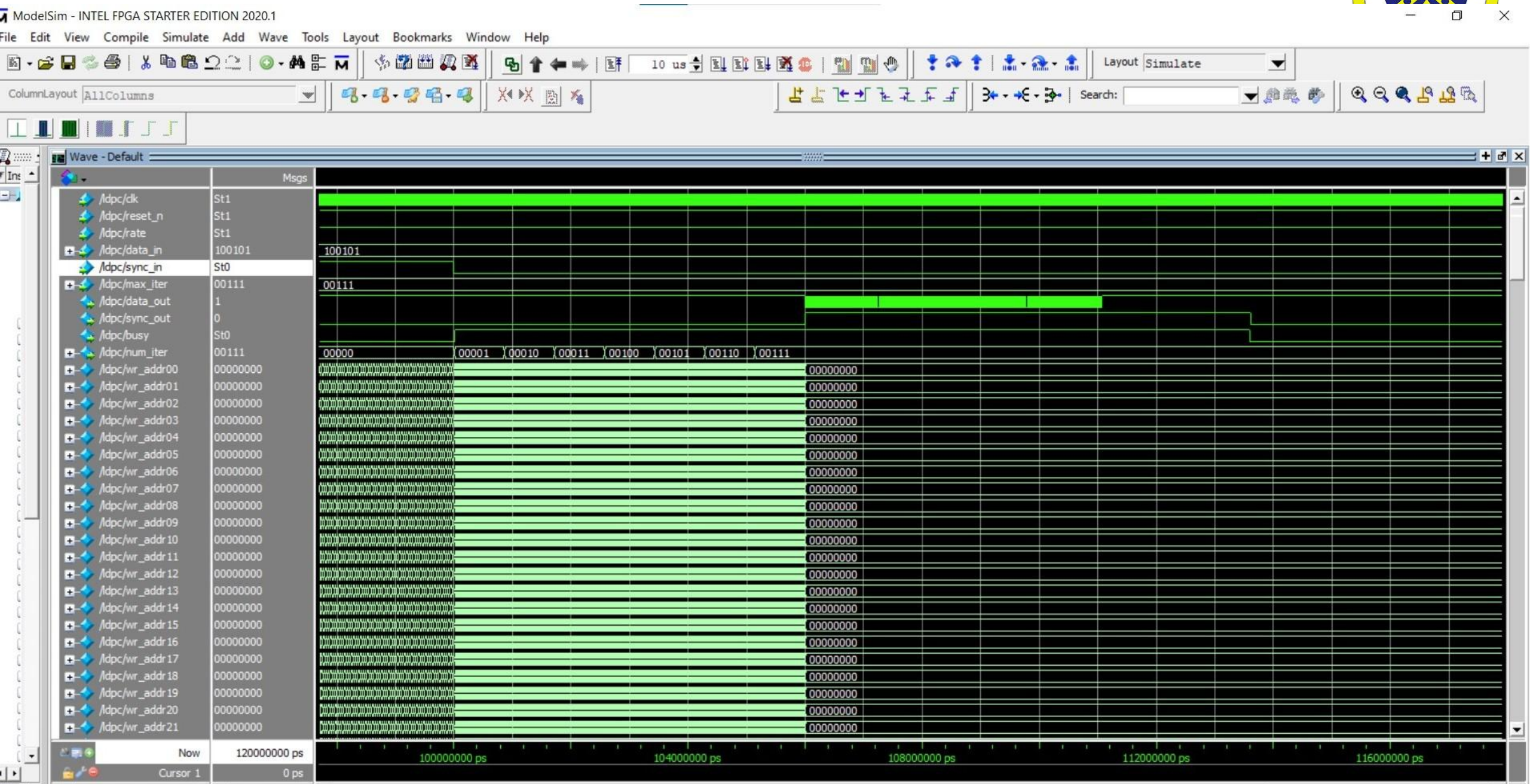
Message Packing and Alignment



RESULTS

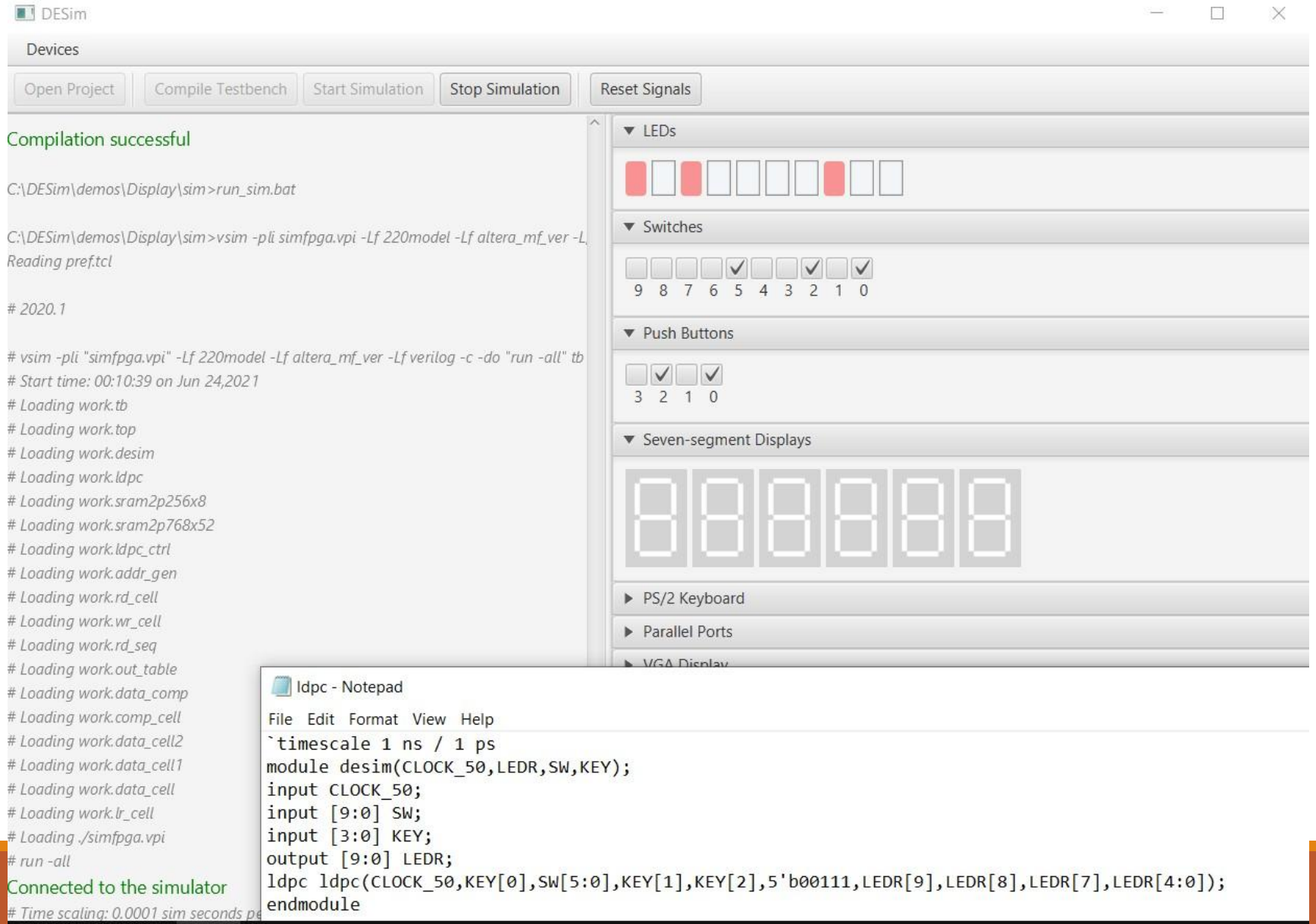


RESULTS

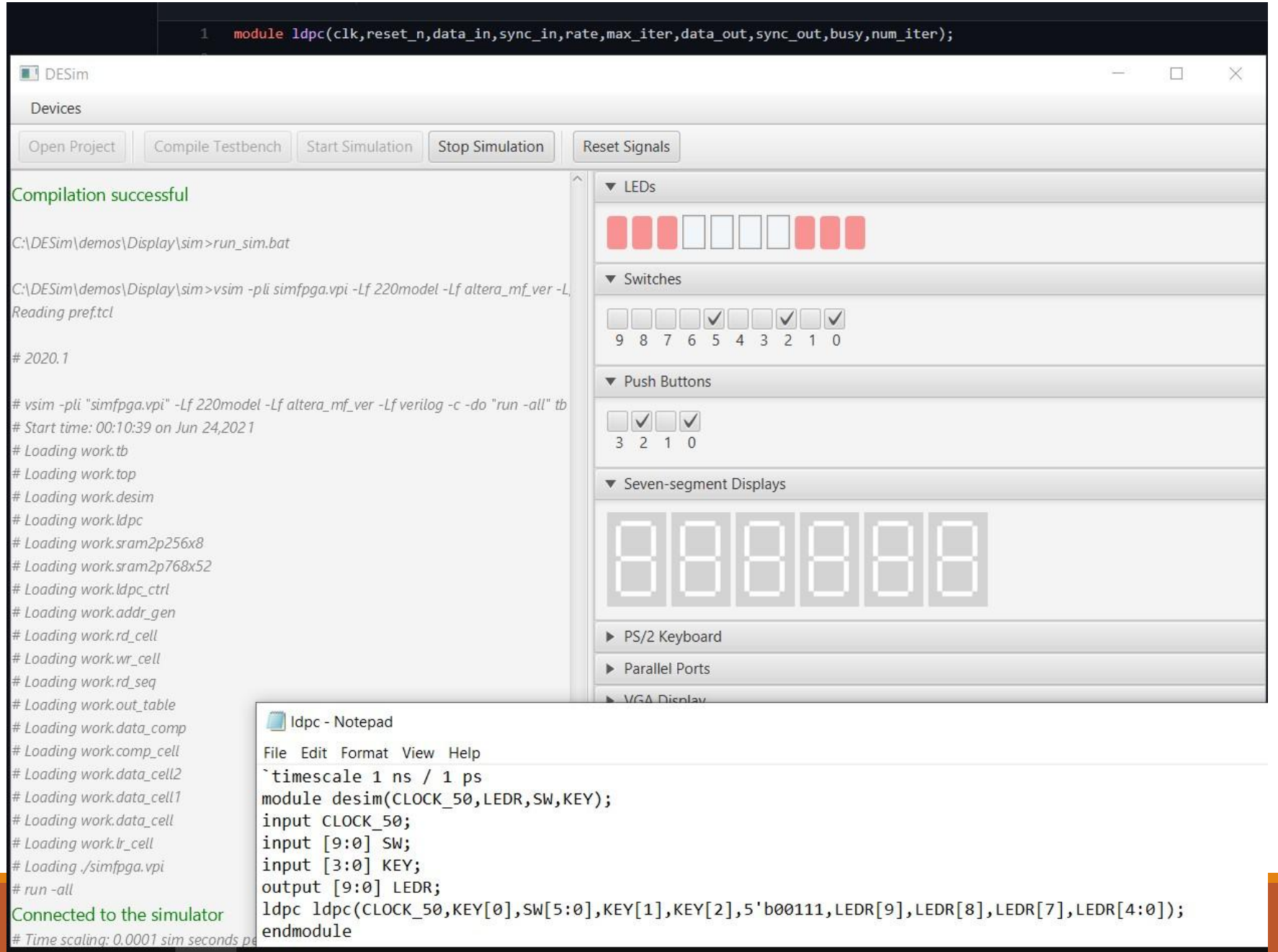

ModelSim - INTEL FPGA STARTER EDITION 2020.1
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Wave - Default
Msgs
/ldpc/clk St1
/ldpc/reset_n St1
/ldpc/rate St1
/ldpc/data_in 100101
/ldpc/sync_in St0
/ldpc/max_iter 00111
/ldpc/data_out 1
/ldpc/sync_out 0
/ldpc/busy St0
/ldpc/num_iter 00111
/ldpc/wr_addr00 00000000
/ldpc/wr_addr01 00000000
/ldpc/wr_addr02 00000000
/ldpc/wr_addr03 00000000
/ldpc/wr_addr04 00000000
/ldpc/wr_addr05 00000000
/ldpc/wr_addr06 00000000
/ldpc/wr_addr07 00000000
/ldpc/wr_addr08 00000000
/ldpc/wr_addr09 00000000
/ldpc/wr_addr10 00000000
/ldpc/wr_addr11 00000000
/ldpc/wr_addr12 00000000
/ldpc/wr_addr13 00000000
/ldpc/wr_addr14 00000000
/ldpc/wr_addr15 00000000
/ldpc/wr_addr16 00000000
/ldpc/wr_addr17 00000000
/ldpc/wr_addr18 00000000
/ldpc/wr_addr19 00000000
/ldpc/wr_addr20 00000000
/ldpc/wr_addr21 00000000
100101
00111
00000
00001
00010
00011
00100
00101
00110
00111
00000000
Now 120000000 ps
Cursor 1 0 ps
100000000 ps
104000000 ps
108000000 ps
112000000 ps
116000000 ps

RESULTS

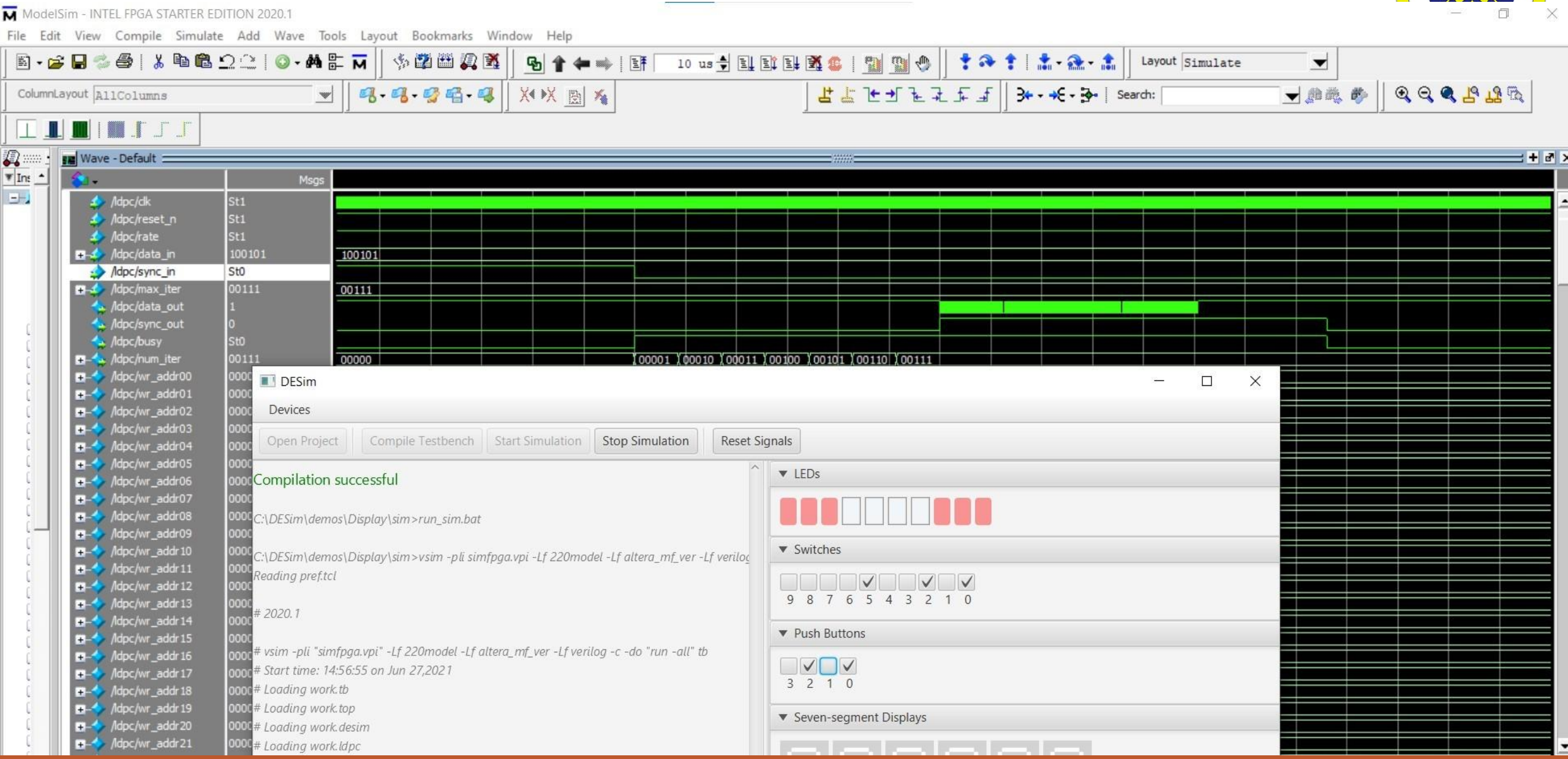
```
1  module ldpc(clk,reset_n,data_in,sync_in,rate,max_iter,data_out,sync_out,busy,num_iter);
```



RESULTS


module ldpc(clk,reset_n,data_in,sync_in,rate,max_iter,data_out,sync_out,busy,num_iter);
DESim
Devices
Open Project
Compile Testbench
Start Simulation
Stop Simulation
Reset Signals
Compilation successful
C:\DESim\demos\Display\sim>run_sim.bat
C:\DESim\demos\Display\sim>vsim -pli simfpga.vpi -Lf 220model -Lf altera_mf_ver -L
Reading pref.tcl
2020.1
vsim -pli "simfpga.vpi" -Lf 220model -Lf altera_mf_ver -Lf verilog -c -do "run -all" tb
Start time: 00:10:39 on Jun 24,2021
Loading work.tb
Loading work.top
Loading work.desim
Loading work.ldpc
Loading work.sram2p256x8
Loading work.sram2p768x52
Loading work.ldpc_ctrl
Loading work.addr_gen
Loading work.rd_cell
Loading work.wr_cell
Loading work.rd_seq
Loading work.out_table
Loading work.data_comp
Loading work.comp_cell
Loading work.data_cell2
Loading work.data_cell1
Loading work.data_cell
Loading work.lr_cell
Loading ./simfpga.vpi
run -all
Connected to the simulator
LEDs
Switches
9 8 7 6 5 4 3 2 1 0
Push Buttons
3 2 1 0
Seven-segment Displays
PS/2 Keyboard
Parallel Ports
ldpc - Notepad
File Edit Format View Help
`timescale 1 ns / 1 ps
module desim(CLOCK_50,LEDR,SW,KEY);
input CLOCK_50;
input [9:0] SW;
input [3:0] KEY;
output [9:0] LEDR;
ldpc ldpc(CLOCK_50,KEY[0],SW[5:0],KEY[1],KEY[2],5'b00111,LEDR[9],LEDR[8],LEDR[7],LEDR[4:0]);
endmodule

RESULTS


ModelSim - INTEL FPGA STARTER EDITION 2020.1
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Wave - Default
Msgs
/ldpc/clk St1
/ldpc/reset_n St1
/ldpc/rate St1
/ldpc/data_in 100101
/ldpc/sync_in St0
/ldpc/max_iter 00111
/ldpc/data_out 1
/ldpc/sync_out 0
/ldpc/busy St0
/ldpc/num_iter 00111
100101
00111
00000
00001 00010 00011 00100 00101 00110 00111
DESim
Devices
Open Project
Compile Testbench
Start Simulation
Stop Simulation
Reset Signals
Compilation successful
C:\DESim\demos\Display\sim>run_sim.bat
C:\DESim\demos\Display\sim>vsim -pli simfpga.vpi -Lf 220model -Lf altera_mf_ver -Lf verilog
Reading pref.tcl
2020.1
vsim -pli "simfpga.vpi" -Lf 220model -Lf altera_mf_ver -Lf verilog -c -do "run -all" tb
Start time: 14:56:55 on Jun 27,2021
Loading work.tb
Loading work.top
Loading work.desim
Loading work.ldpc
LEDs
Switches
9 8 7 6 5 4 3 2 1 0
Push Buttons
3 2 1 0
Seven-segment Displays

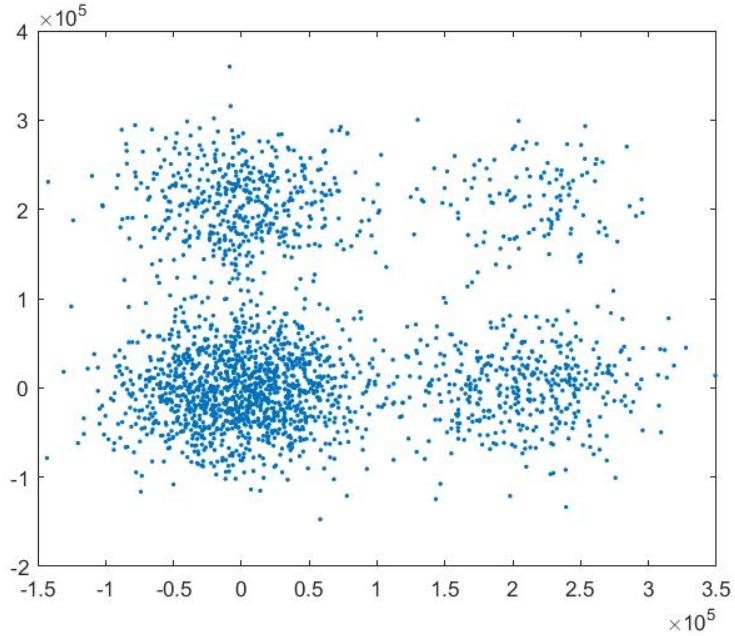
RESULTS

ldpc_encode.m | ldpc_decode.m

```
10 -    clc;
11 -    format compact;
12 -    clear all;
13 -    coderate = 0.5;
14 -    mode = '16qm';
15      % modulate = 'bpsk' 'qpsk' '16qm'
16 -    SNR = 7.7;
17
18 -    if coderate == 0.5
19 -        X=randi(9216*coderate,1);
20          %X=zeros(9216*coderate,1);
21 -        load data\G.mat
22 -        PC= mod((X*X),2);
23 -        MSG(1:4608)=PC';
24 -        MSG(4608+1:4608+4608)=X;
25          %col_order=[0 1 2 3 4 5 9 10 11 13 15 16 17 19 20 22 23 24 26 27 31 32 33 34 36 37 38 39 41 4
26 -        load data\col_order.mat
27 -        for i=1:9216
28 -            C(col_order(i)+1)=MSG(i);
29 -        end
30 -    end
31
32 -    if coderate == 0.75
33 -        X=randint(9216*coderate,1);
```

Command Window

```
Hard Decision BER: 8104
Min-Sum Alogorithm BER: 7752
```



APPLICATIONS/FUTURE SCOPE

3GPP - 5G NR

ETSI - DVB-S2X, DVB-S2, DVB-T2, DVB-T2-Lite, DVB-C2, GMR-1

IEEE - IEEE 802.3 (10 GBASE-T), IEEE 802.11 (WiFi), IEEE 802.15.3c (60 GHz PHY),IEEE 802.16 (Mobile WiMAX), IEEE 802.22 (WRAN)

Others - ATSC 3.0, DOCSIS 3.1, CCSDS, CMMB, DTMB(DMB-T/H),ITU-TG.hn (G.9960),WiMedia 1.5 UWB

Drafts - DVB-NGH, IEEE 802.11ac (WiFi), IEEE 802.11ad (WiGig)

CONCLUSION

- Understand advancements in the field of Linear Block Codes, LDPC Codes Basics, LDPC Codes and its rateless relatives, types of LDPC Encoding and Decoding etc.
- DESim is used for the FPGA implementation
- Which is an emulator of DE-10 lite FPGA Kit by “FPGAcademy”
- RTL simulation is done using Quartus Prime and RTL
- Verification of DESim output using ModelSim.
- BER for Hard Decision Algorithm and Min-Sum Algorithm are compared where Min Sum has low bit error rate.
- Waveform for modulations is obtained using ldpc encoding- decoding simulation using Matlab.

REFERENCES

- Near Shannon Limit Performance of Low Density Parity Check Codes - David J.C. MacKay, Radford M. Neal
- Gallager Codes – Recent Results David J. C. MacKay
- Comprehensive Algorithmic Review and Analysis of LDPC Codes, Waheed Ullah, University of the Witwatersrand, Abid Yahya, Universiti Malaysia Perlis
- Reduced Complexity Iterative Decoding of Low-Density Parity Check Codes Based on Belief Propagation Marc P. C. Fossorier, Member, IEEE, Miodrag Mihaljevic, and Hideki Imai, ´ Fellow, IEEE
- Implementation of Near Shannon Limit Error-Correcting Codes Using Reconfigurable Hardware, Benjamin Levine , R Reed Taylor , Herman Schmit
- Parallel Decoding Architectures for Low Density Parity Check Codes, C.Howland and A. Blanksby
- A 220mW 1Gb/s 1024-Bit Rate-1/2 Low Density Parity Check Decoder , Chris Howland and Ansrew Blanksby, High Speed Communications VLSI Research Department, Agere Systems, Holmdel NJ 07733
- Joint Code and Decoder Design for Implementation Oriented (3, k)- regular LDPC Codes, Tong Zhang and Keshab K. Parhi, Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, USA
- Low-Density Parity-Check Codes and Their Rateless Relatives,Nicholas Bonello, Sheng Chen and Lajos Hanzo

REFERENCES

- Performance Analysis and Code Optimization of Low Density Parity-Check Codes on Rayleigh Fading Channels Jilei Hou, Student Member, IEEE, Paul H. Siegel, Fellow, IEEE, and Laurence B. Milstein, Fellow, IEEE
- Performance Comparison of Layered Space Time Codes Ka Leong Lo, Slavica Marinkovic, Zhuo Chen and Branka Vucetic The School of Electrical and Information Engineering, The University of Sydney, NSW 2006, Australia
- Low-Density Parity-Check (LDPC) Coded OFDM Systems with M-PSK, Hisashi Futaki, Tomoaki Ohtsuki, Graduate School of Science and Technology, Tokyo University of Science, Faculty of Science and TEchnology, Tokyo University of Science
- Reduced-Complexity Decoding of Q-ary LDPC Codes for Magnetic Recording Hongxin Song, Member, IEEE, and J. R. Cruz, Fellow, IEEE
- Efficient Encoding of Low-Density Parity-Check Codes Thomas J. Richardson and Rüdiger L. Urbanke
- High Throughput Low-Density Parity-Check Decoder Architectures, Engling Yeo, Payam Pakzad, Borivoje Nikolić, and Venkat Anantharam Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720-1770
- VLSI Implementation-Oriented (3,k)-Regular Low-Density Parity-Check Codes, Tong Xhang and Keshab K.Parhi,Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, USA
- Low Density Parity Check Codes over GF(q), Matthew C. Davey and David J. C MacKay, Cavendish Laboratory, Cambridge CB3 OHE, England

THANK YOU