ECE-111 Advanced Digital Term Project: Viterbi Decoder

Anne Lin | Christina Mai

Google Doc for easier navigation: ECE 111 Final Project Report

Viterbi Decoder

- **Define**
  - Decodes convolutional codes. A maximum likely-hood decoding algorithm
  - Implements a Viterbi algorithm using three blocks:
    - Branch metric calculation unit (BMU)
      - Calculates hamming distance metric (match between bits received and valid codeword) for each branch
    - Add-compare-select unit (ACS)
      - Updates state metrics every clock cycle
    - Survivor path decoding unit (Trace-back module operation (TBU))
      - Finds the shortest path through the trellis
      - Drives output value
- **Goal**
  - Complete the encoder.v as well as decoder.v and its submodules
- **Procedure**
  - **Add-Compare-Select**

path cost(i) = path metric(i) + branch metric(i)

valid_o	selection	path cost
0	0	0
0	1	0
1	0	path_cost_0
1	1	path_cost_1

valid_o = 0 if neither path valid

path_o_valid	path_1_valid	selection
0	0	0
0	1	1

1	0	0
1	1	*

*: 1 if path_cost_0 larger than path_cost_1; else 0

- **Branch Metric Computation Blocks**

(8 -- identical, except as noted below)

tmp00 = rx_pair[0]; tmp01 = rx_pair[1]
exception: for bmc 1,2,5, and 6, invert rx_pair[1] in the above expression

tmp10 = !tmp00 tmp 11 = !tmp01

path_0_bmc[1] = tmp00 & tmp01
path_0_bmc[0] = tmp00 ^ tmp01
same for path_1_bmc with tmp10 and tmp11

- **Encoder**

cstate	d_in = 0		d_in = 1	
	nstate	d_out_reg	nstate	d_out_reg
0	0	00	4	11
1	4	00	0	11
2	5	10	1	01
3	1	10	5	01
4	2	10	6	01
5	6	10	2	01
6	7	00	3	11
7	3	00	7	11

- **Trace Back Unit**

wr_en_reg = selection
d_o_reg = if (selection) d_in_1[pstate] else d_o_reg = 0

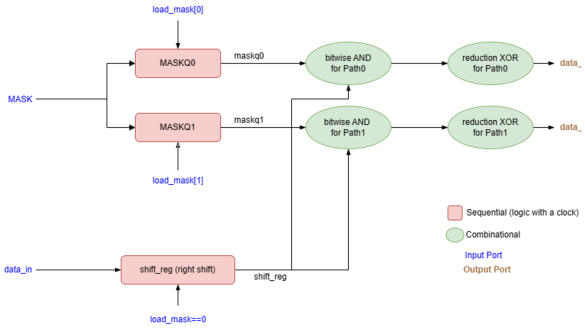
State Transition Table: nstate[i] (Next State) = f(pstate[i], selection, d_in_0[i], d_in_1[i]

pstate	selection=0	selection=0	selection=1	selection=1
	d_in_0[pstate] =0	d_in_0[pstate] =1	d_in_1[pstate] =0	d_in_1[pstate] =1
0	0	1	0	1
1	3	2	3	2
2	4	5	4	5
3	7	6	7	6
4	1	0	1	0
5	2	3	2	3
6	5	4	5	4
7	6	7	6	7

How does the Encoder work?:

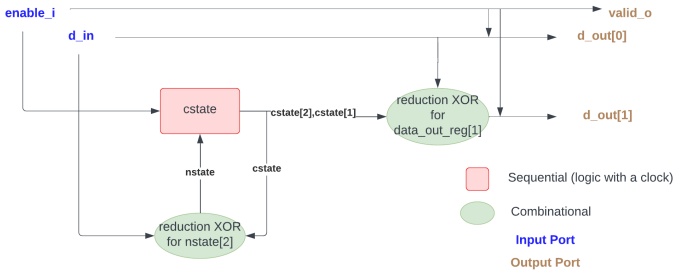
In comparison to HW 7's Encoder, the one used in this term project is different as shown below:

HW7: Rate ½ Convolutional Encoder



The encoder from HW 7 was non-systematic and non-recursive.

Term Project:



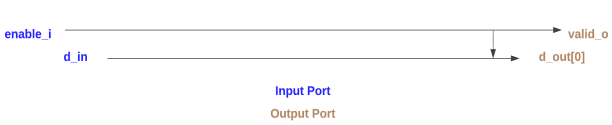
The encoder for the term project is systematic and recursive.

Reference: ECE 111 Discussion HW 7 by Naman Sehgal

The encoder **sequentially** computes "cstate", the current state, based on asynchronous "rst"(reset) and "enable_i". When "rst" or "enable_i" is low, cstate is loaded with 0 but in any other case, "cstate" would be loaded with "nstate" (next state).

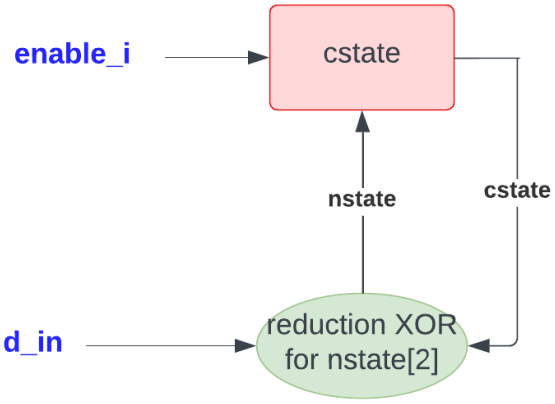
Combinationly, "valid_i," "nstate," and "d_out_reg" are computed. "Valid_i" is set equal to "enable_i". "nstate" is a right shift of "cstate" and then loaded with the reduction XOR of "d_in," "cstate[1]," and "cstate[0]". "d_out_reg[0]" is equal to "d_in", which makes the encoder systematic, and "d_out_reg[1]" is the reduction XOR of "d_in" "cstate[2]," and "cstate[1]". "

d_out" is **assigned** to "d_out_reg" when "enable_i" is high and to 0 when "enable_i" is 0.



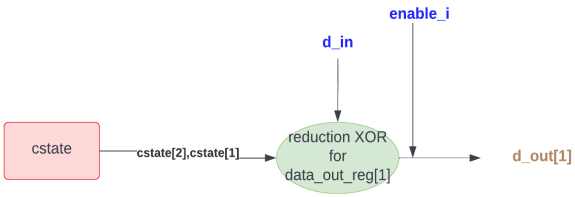
d_out[0] & valid_o

- Valid_o = enable_i
- d_out[0] = d_in if enable_i is 1
- dout[0] = din
    - Systematic since input is shown in the output



Recursion

- Description of cstate:
  - 3-bit wide register (sequential)
  - Has a reset (rst)
  - Value gets updated with ‘nstate’ when ‘enable_i’ is high
    - If ‘enable_i’ is low, ‘cstate’ is set to 0
- Description of nstate:
  - 3-bit wide register (combinational)
  - = {d_in ^ (cstate[1] ^ cstate[0]), cstate[2], cstate[1]}
    - Right shift with MSB loaded with reduction XOR (d_in,cstate[1],cstate[0])



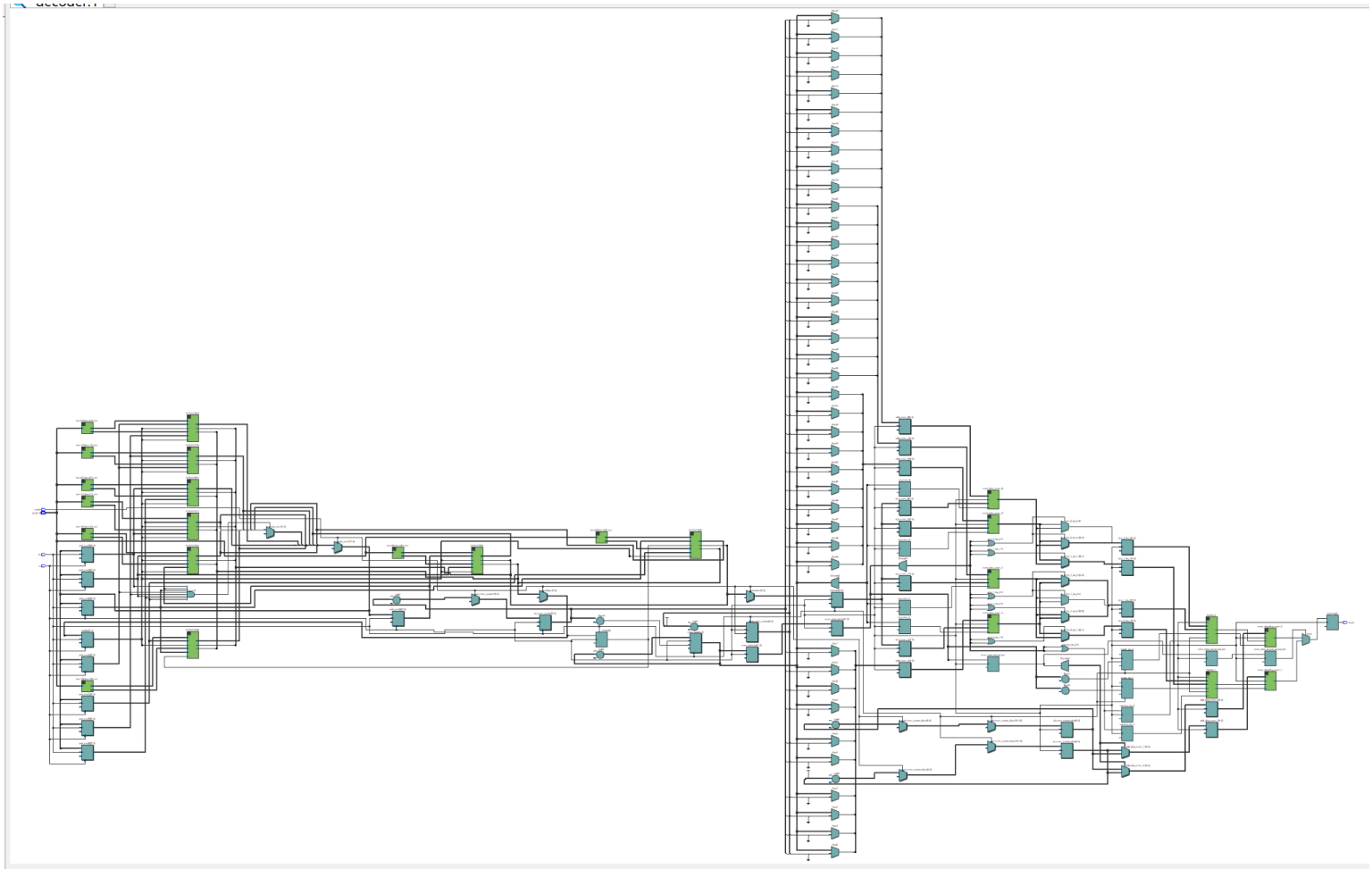
d_out[1]

- = d_in ^ (cstate[2]^cstate[1])
  - When enable_i is high
  - Reduction XOR (d_in, cstate[2],cstate[1])

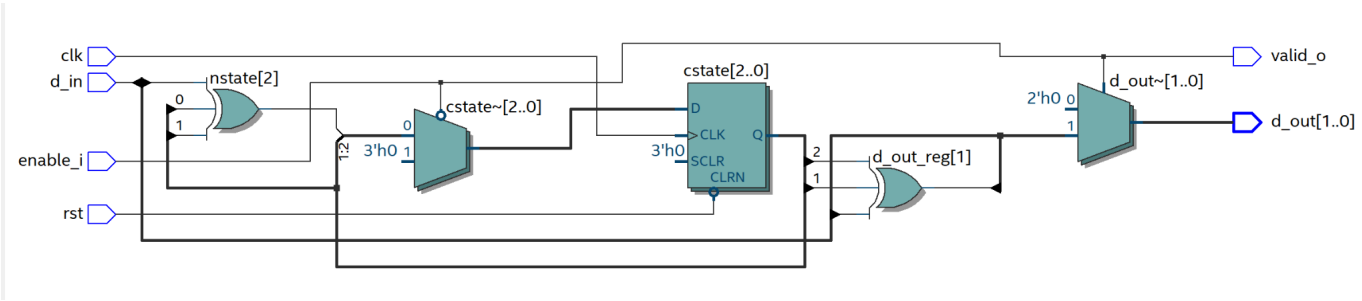
Synthesis

RTL Schematic

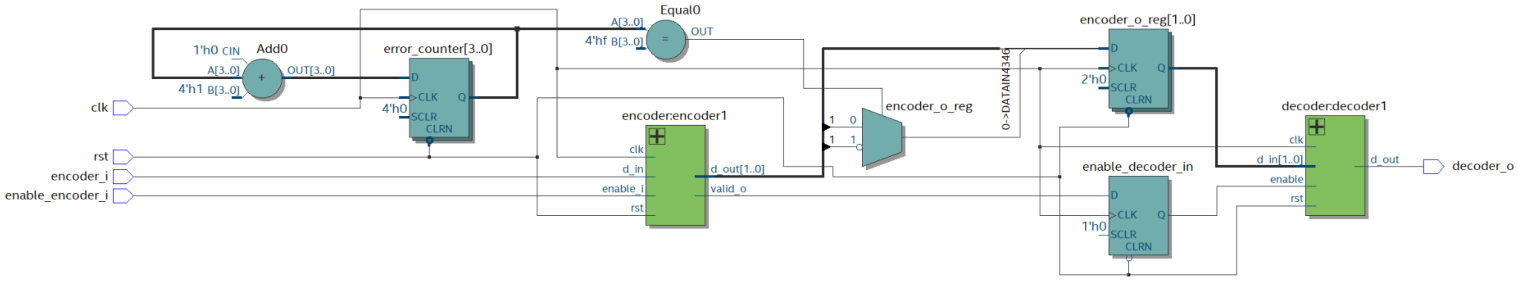
decoder.sv



encoder.sv

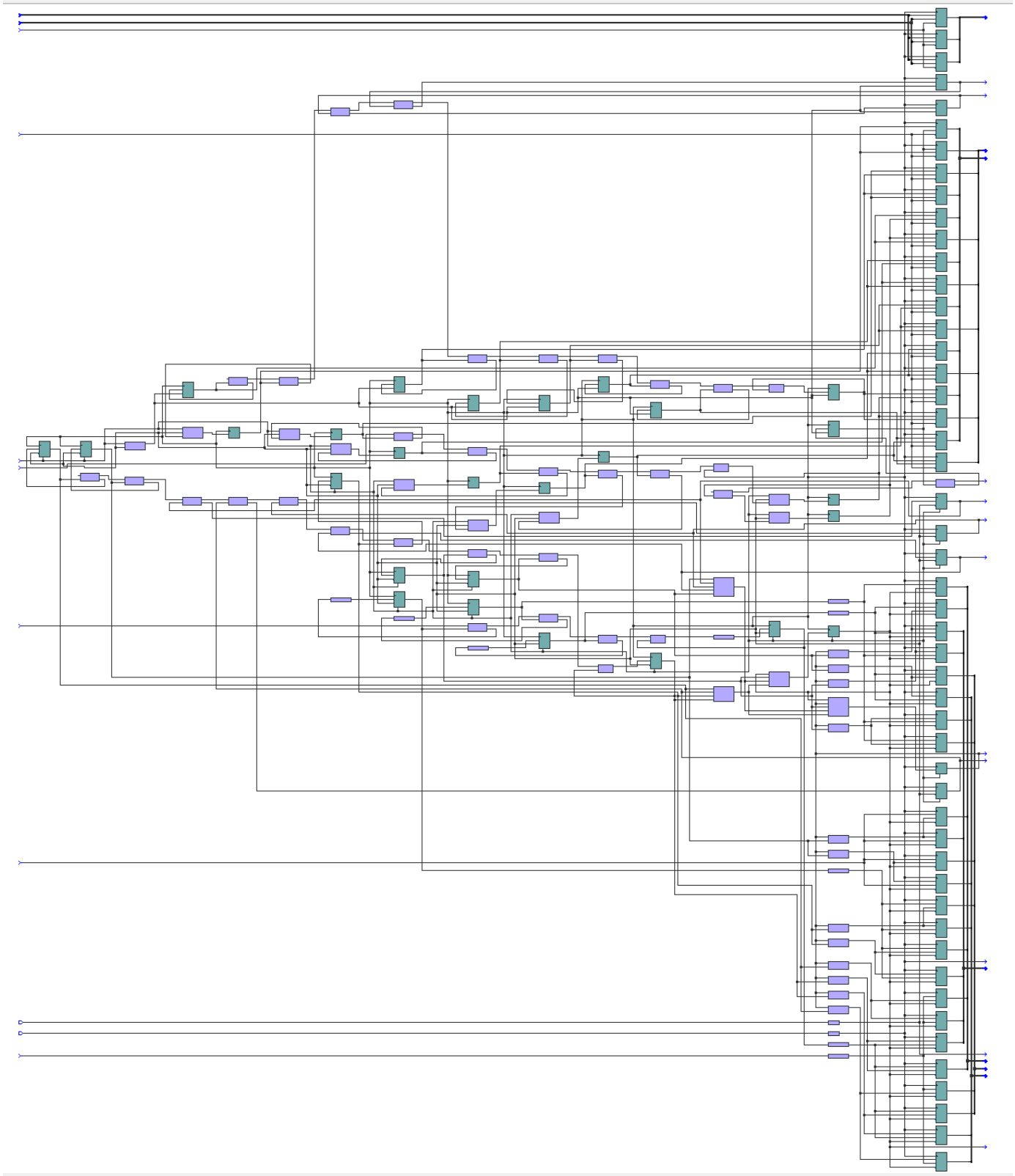


Viterbi_tx_rx.vs

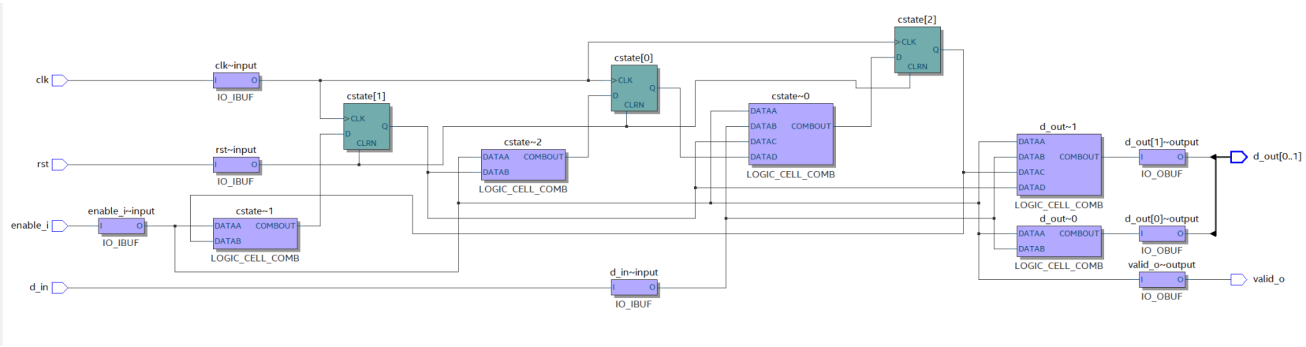


Post Mapping Schematic

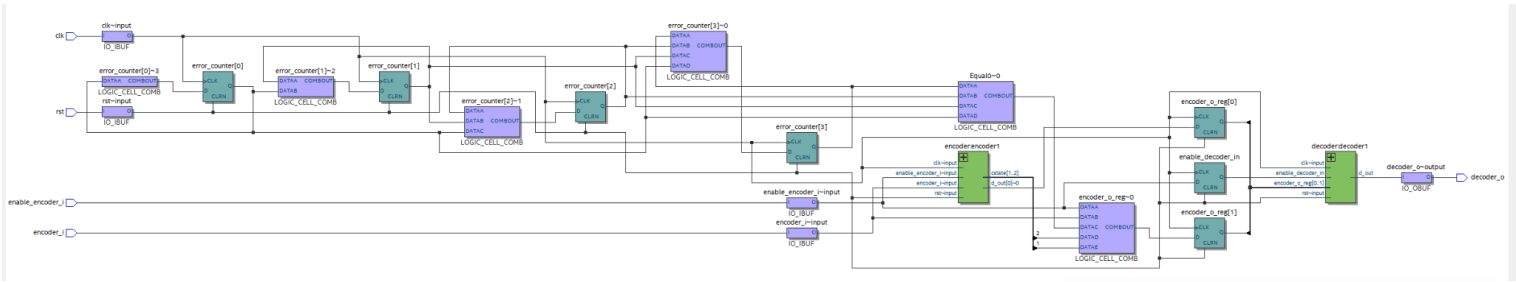
[decoder.sv]



encoder.sv



Viterbi_tx_rx.vs



Resource Usage

decoder.sv

- **Number of ALUT:** 357
- **Number of Functions:**
  - 2 (7 input functions)
  - 75 (6 input functions)
  - 20 (5 input functions)
  - 14 (4 input functions)
  - 246 (<= 3 input functions)
- **Number of Dedicated Logic Registers:** 239
- **I/O Pins:** 6
- **CLK pins:** 1

	Resource	Usage
1	ALUTs Used	357 / 36,100 (< 1 %)
1	-- Combinational ALUTs	357 / 36,100 (< 1 %)
2	-- Memory ALUTs	0 / 18,050 (0 %)
3	-- LUT_REGs	0 / 36,100 (0 %)
2	Dedicated logic registers	239 / 36,100 (< 1 %)
3		
4	Combinational ALUT ...by number of inputs	
1	-- 7 input functions	2
2	-- 6 input functions	75
3	-- 5 input functions	20
4	-- 4 input functions	14
5	-- <=3 input functions	246
5		
6	Combinational ALUTs by mode	
1	-- normal mode	187
2	-- extended LUT mode	2
3	-- arithmetic mode	168
4	-- shared arithmetic mode	0
7		
8	Logic utilization	481 / 36,100 (1 %)
1	-- Difficulty Clustering Design	Low
2	-- Combinational ALU...d in final Placement	449
1	-- Combinational with no register	210
2	-- Register only	92
3	-- Combinational with a register	147
3	-- Estimated pairs re...sters as design grows	-33
4	-- Estimated Combina...er pairs unavailable	65
1	-- Unavailable due to Memory LAB use	0
4	-- Estimated Combina...er pairs unavailable	65
1	-- Unavailable due to Memory LAB use	0
2	-- Unavailable due ... unpartnered 7 LUTs	2
3	-- Unavailable due ... unpartnered 6 LUTs	46
4	-- Unavailable due ... unpartnered 5 LUTs	0
5	-- Unavailable due t...ide signal conflicts	17
6	-- Unavailable due to LAB input limits	0
9		
10	Total registers*	239
1	-- Dedicated logic registers	239 / 36,100 (< 1 %)
2	-- I/O registers	0 / 2,232 (0 %)
3	-- LUT_REGs	0
11		
12	ALMs: partially or completely used	255 / 18,050 (1 %)
13		
14	Total LABs: partially or completely used	27 / 1,805 (1 %)
1	-- Logic LABs	27 / 27 (100 %)
2	-- Memory LABs	0 / 27 (0 %)
15		
16	Virtual pins	0
17	I/O pins	6 / 404 (1 %)
1	-- Clock pins	1 / 10 (10 %)
2	-- Dedicated input pins	0 / 28 (0 %)
18		
19	M9K blocks	6 / 319 (2 %)
20	Total MLAB memory bits	0
21	Total block memory bits	34,816 / 2...04 (1 %)
22	Total block memory implementation bits	55,296 / 2...04 (2 %)
23	DSP block 18-bit elements	0 / 232 (0 %)
24	PLLs	0 / 4 (0 %)
25	Global signals	2
1	-- Global clocks	2 / 16 (13 %)
2	-- Quadrant clocks	0 / 48 (0 %)
3	-- Periphery clocks	0 / 50 (0 %)
26	SERDES receivers	0 / 28 (0 %)
27	JTAGs	0 / 1 (0 %)
28	ASMI blocks	0 / 1 (0 %)
29	CRC blocks	0 / 1 (0 %)
30	Remote update blocks	0 / 1 (0 %)
31	Oscillator blocks	0 / 1 (0 %)
32	GXB Receiver channel PCSs	0 / 8 (0 %)
33	GXB Receiver channel PMAs	0 / 8 (0 %)
34	GXB Transmitter channel PCSs	0 / 8 (0 %)
35	GXB Transmitter channel PMAs	0 / 8 (0 %)
36	HSSI CMU PLLs	0 / 4 (0 %)
37	Impedance control blocks	0 / 3 (0 %)
38	Average interconnect usage (total/H/V)	0.1% / 0.2% / 0.1%
39	Peak interconnect usage (total/H/V)	2.2% / 2.4% / 1.8%
40	Maximum fan-out	245
41	Highest non-global fan-out	66
42	Total fan-out	2325
43	Average fan-out	3.34

encoder.sv

- **Number of ALUT:** 5
- **Number of Functions:**
    - 2 (4 input functions)
    - 3 (<= 3 input functions)
- **Number of Dedicated Logic Registers:** 3
- **I/O Pins:** 7
- **CLK pins:** 1

	Resource	Usage
1	ALUTs Used	5 / 36,100 (< 1 %)
1	-- Combinational ALUTs	5 / 36,100 (< 1 %)
2	-- Memory ALUTs	0 / 18,050 (0 %)
3	-- LUT_REGs	0 / 36,100 (0 %)
2	Dedicated logic registers	3 / 36,100 (< 1 %)
3		
4	Combinational ALUT ...by number of inputs	
1	-- 7 input functions	0
2	-- 6 input functions	0
3	-- 5 input functions	0
4	-- 4 input functions	2
5	-- <=3 input functions	3
5		
6	Combinational ALUTs by mode	
1	-- normal mode	5
2	-- extended LUT mode	0
3	-- arithmetic mode	0
4	-- shared arithmetic mode	0
7		
8	Logic utilization	5 / 36,100 (< 1 %)
1	-- Difficulty Clustering Design	Low
2	-- Combinational ALU...d in final Placement	5
1	-- Combinational with no register	2
2	-- Register only	0
3	-- Combinational with a register	3
3	-- Estimated pairs re...sters as design grows	0
4	-- Estimated Combina...er pairs unavailable	0
1	-- Unavailable due to Memory LAB use	0
4	-- Estimated Combina...er pairs unavailable	0
1	-- Unavailable due to Memory LAB use	0
2	-- Unavailable due ... unpartnered 7 LUTs	0
3	-- Unavailable due ... unpartnered 6 LUTs	0
4	-- Unavailable due ... unpartnered 5 LUTs	0
5	-- Unavailable due t...ide signal conflicts	0
6	-- Unavailable due to LAB input limits	0
9		
10	Total registers*	3
1	-- Dedicated logic registers	3 / 36,100 (< 1 %)
2	-- I/O registers	0 / 2,232 (0 %)
3	-- LUT_REGs	0
11		
12	ALMs: partially or completely used	4 / 18,050 (< 1 %)
13		
14	Total LABs: partially or completely used	1 / 1,805 (< 1 %)
1	-- Logic LABs	1 / 1 (100 %)
2	-- Memory LABs	0 / 1 (0 %)
15		
16	Virtual pins	0
17	I/O pins	7 / 404 (2 %)
1	-- Clock pins	1 / 10 (10 %)
2	-- Dedicated input pins	0 / 28 (0 %)
18		
19	M9K blocks	0 / 319 (0 %)
20	Total MLAB memory bits	0
21	Total block memory bits	0 / 2,939,904 (0 %)
22	Total block memory implementation bits	0 / 2,939,904 (0 %)
19	M9K blocks	0 / 319 (0 %)
20	Total MLAB memory bits	0
21	Total block memory bits	0 / 2,939,904 (0 %)
22	Total block memory implementation bits	0 / 2,939,904 (0 %)
23	DSP block 18-bit elements	0 / 232 (0 %)
24	PLLs	0 / 4 (0 %)
25	Global signals	2
1	-- Global clocks	2 / 16 (13 %)
2	-- Quadrant clocks	0 / 48 (0 %)
3	-- Periphery clocks	0 / 50 (0 %)
26	SERDES receivers	0 / 28 (0 %)
27	JTAGs	0 / 1 (0 %)
28	ASMI blocks	0 / 1 (0 %)
29	CRC blocks	0 / 1 (0 %)
30	Remote update blocks	0 / 1 (0 %)
31	Oscillator blocks	0 / 1 (0 %)
32	GXB Receiver channel PCSs	0 / 8 (0 %)
33	GXB Receiver channel PMAs	0 / 8 (0 %)
34	GXB Transmitter channel PCSs	0 / 8 (0 %)
35	GXB Transmitter channel PMAs	0 / 8 (0 %)
36	HSSI CMU PLLs	0 / 4 (0 %)
37	Impedance control blocks	0 / 3 (0 %)
38	Average interconnect usage (total/H/V)	0.0% / 0.0% / 0.0%
39	Peak interconnect usage (total/H/V)	0.0% / 0.0% / 0.0%
40	Maximum fan-out	6
41	Highest non-global fan-out	6
42	Total fan-out	36
43	Average fan-out	1.38

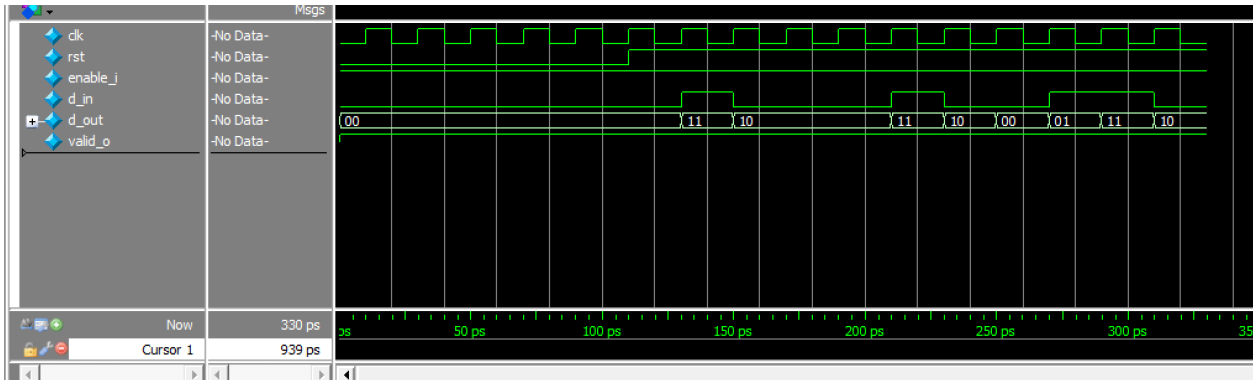
Viterbi_tx_rx.vs

- **Number of ALUT:** 367
- **Number of Functions:**
    - 2 (7 input functions)
    - 75 (6 input functions)
    - 21 (5 input functions)
    - 17 (4 input functions)
    - 152 (<= 3 input functions)
- **Number of Dedicated Logic Registers:** 249
- **I/O Pins:** 5
- **CLK pins:** 1

	Resource	Usage
1	ALUTs Used	367 / 36,100 (1 %)
1	-- Combinational ALUTs	367 / 36,100 (1 %)
2	-- Memory ALUTs	0 / 18,050 (0 %)
3	-- LUT_REGs	0 / 36,100 (0 %)
2	Dedicated logic registers	249 / 36,100 (< 1 %)
3		
4	Combinational ALUT ...by number of inputs	
1	-- 7 input functions	2
2	-- 6 input functions	75
3	-- 5 input functions	21
4	-- 4 input functions	17
5	-- <=3 input functions	252
5		
6	Combinational ALUTs by mode	
1	-- normal mode	197
2	-- extended LUT mode	2
3	-- arithmetic mode	168
4	-- shared arithmetic mode	0
7		
8	Logic utilization	476 / 36,100 (1 %)
1	-- Difficulty Clustering Design	Low
2	-- Combinational ALU...d in final Placement	469
1	-- Combinational with no register	220
2	-- Register only	102
3	-- Combinational with a register	147
3	-- Estimated pairs re...sters as design grows	-42
4	-- Estimated Combina...er pairs unavailable	49
1	-- Unavailable due to Memory LAB use	0
4	-- Estimated Combina...er pairs unavailable	49
1	-- Unavailable due to Memory LAB use	0
2	-- Unavailable due ... unpartnered 7 LUTs	1
3	-- Unavailable due ... unpartnered 6 LUTs	47
4	-- Unavailable due ... unpartnered 5 LUTs	1
5	-- Unavailable due t...ide signal conflicts	0
6	-- Unavailable due to LAB input limits	0
9		
10	Total registers*	249
1	-- Dedicated logic registers	249 / 36,100 (< 1 %)
2	-- I/O registers	0 / 2,232 (0 %)
3	-- LUT_REGs	0
11		
12	ALMs: partially or completely used	276 / 18,050 (2 %)
13		
14	Total LABs: partially or completely used	30 / 1,805 (2 %)
1	-- Logic LABs	30 / 30 (100 %)
2	-- Memory LABs	0 / 30 (0 %)
15		
16	Virtual pins	0
17	I/O pins	5 / 404 (1 %)
1	-- Clock pins	1 / 10 (10 %)
2	-- Dedicated input pins	0 / 28 (0 %)
18		
19	M9K blocks	6 / 319 (2 %)
20	Total MLAB memory bits	0
21	Total block memory bits	34,816 / 2...04 (1 %)
22	Total block memory implementation bits	55,296 / 2...04 (2 %)
19	M9K blocks	6 / 319 (2 %)
20	Total MLAB memory bits	0
21	Total block memory bits	34,816 / 2...04 (1 %)
22	Total block memory implementation bits	55,296 / 2...04 (2 %)
23	DSP block 18-bit elements	0 / 232 (0 %)
24	PLLs	0 / 4 (0 %)
25	Global signals	2
1	-- Global clocks	2 / 16 (13 %)
2	-- Quadrant clocks	0 / 48 (0 %)
3	-- Periphery clocks	0 / 50 (0 %)
26	SERDES receivers	0 / 28 (0 %)
27	JTAGs	0 / 1 (0 %)
28	ASMI blocks	0 / 1 (0 %)
29	CRC blocks	0 / 1 (0 %)
30	Remote update blocks	0 / 1 (0 %)
31	Oscillator blocks	0 / 1 (0 %)
32	GXB Receiver channel PCSs	0 / 8 (0 %)
33	GXB Receiver channel PMAs	0 / 8 (0 %)
34	GXB Transmitter channel PCSs	0 / 8 (0 %)
35	GXB Transmitter channel PMAs	0 / 8 (0 %)
36	HSSI CMU PLLs	0 / 4 (0 %)
37	Impedance control blocks	0 / 3 (0 %)
38	Average interconnect usage (total/H/V)	0.2% / 0.2% / 0.1%
39	Peak interconnect usage (total/H/V)	1.5% / 1.9% / 1.1%
40	Maximum fan-out	255
41	Highest non-global fan-out	66
42	Total fan-out	2395
43	Average fan-out	3.30

Simulation Results (Modelsim)

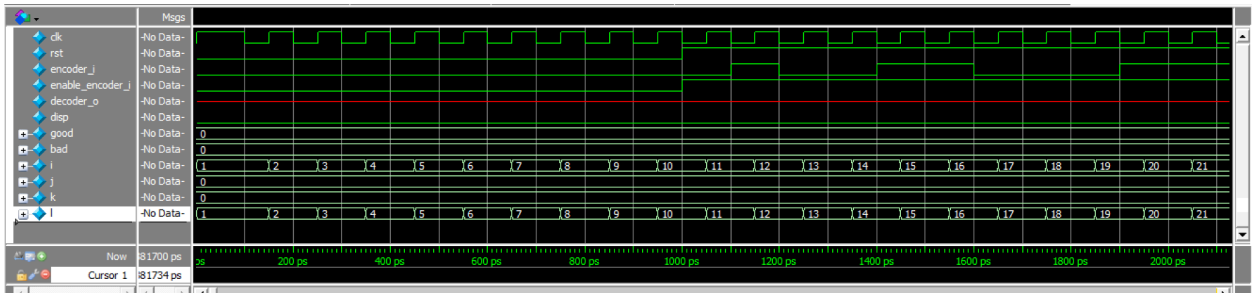
Encoder_tb.sv simulation:

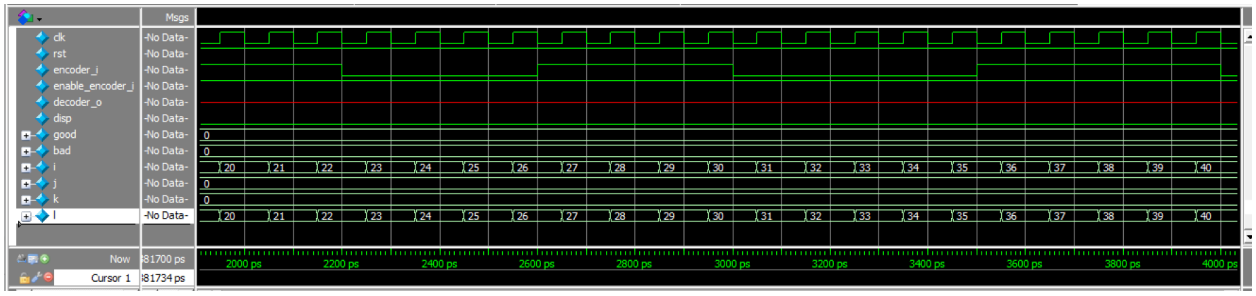


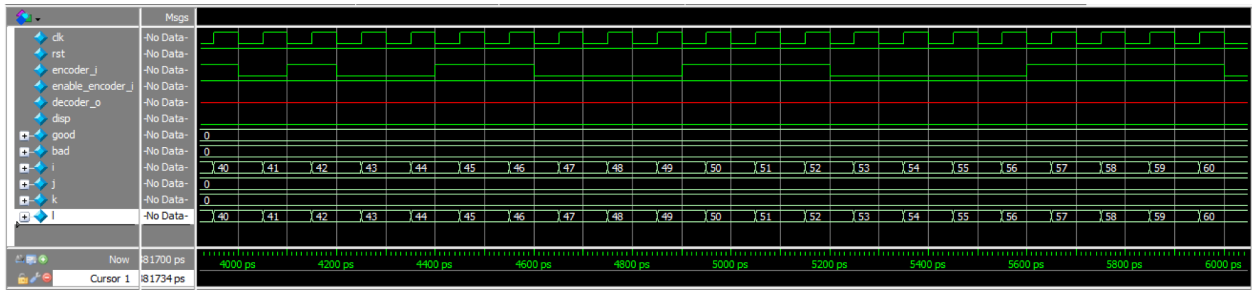

```
VSIM 10> run -all
# ** Note: $stop    : C:/Users/ITSloaner.DESKTOP-F1ESCSS.000/Downloads/vitassign/encoder_tb.sv(62)
#    Time: 330 ps  Iteration: 0  Instance: /encoder_tb
# Break in Module encoder_tb at C:/Users/ITSloaner.DESKTOP-F1ESCSS.000/Downloads/vitassign/encoder_tb.sv line 62
```

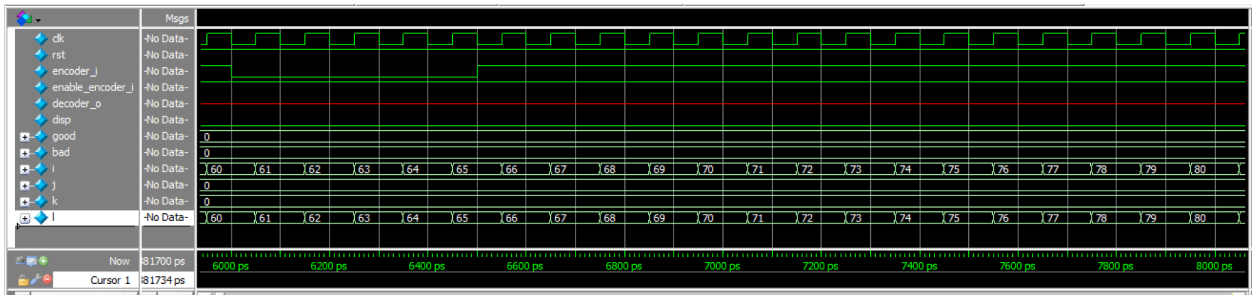
Viterbi_tx_rx_tb.sv simulation:

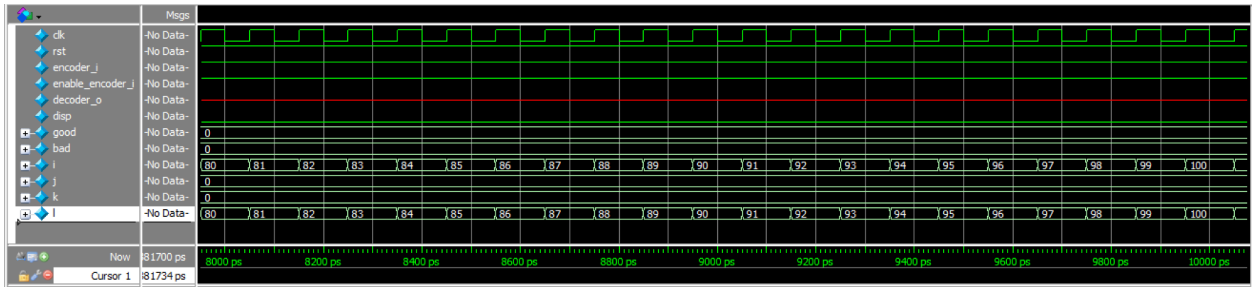
```
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# good =         256, bad =           0
# ** Note: $stop    : C:/Users/ITSloaner.DESKTOP-F1ESCSS.000/Downloads/vitassign/viterbi_tx_rx_tb.sv(371)
#    Time: 1381700 ps  Iteration: 0  Instance: /viterbi_tx_rx_tb
# Break in Module viterbi_tx_rx_tb at C:/Users/ITSloaner.DESKTOP-F1ESCSS.000/Downloads/vitassign/viterbi_tx_rx_tb.sv line 371
```

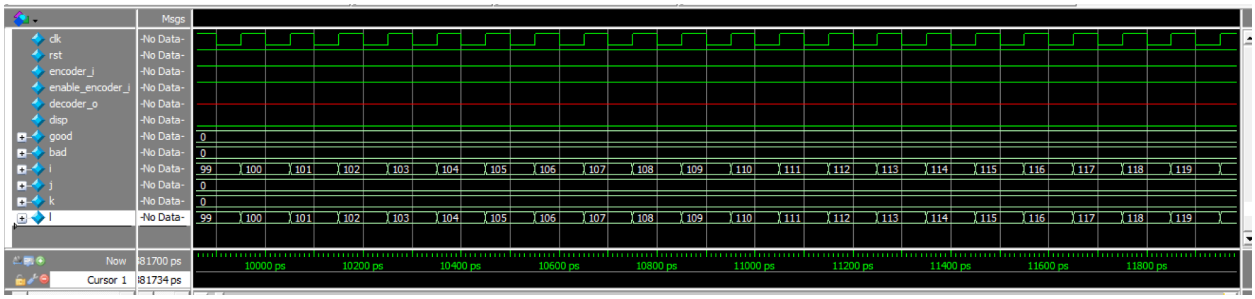


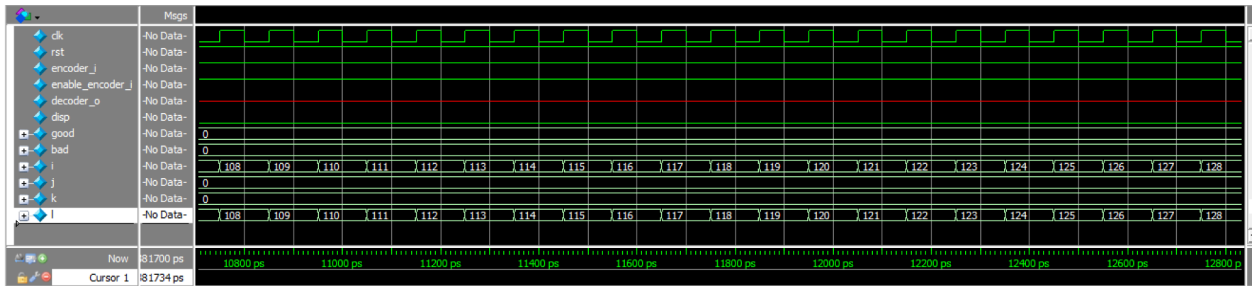


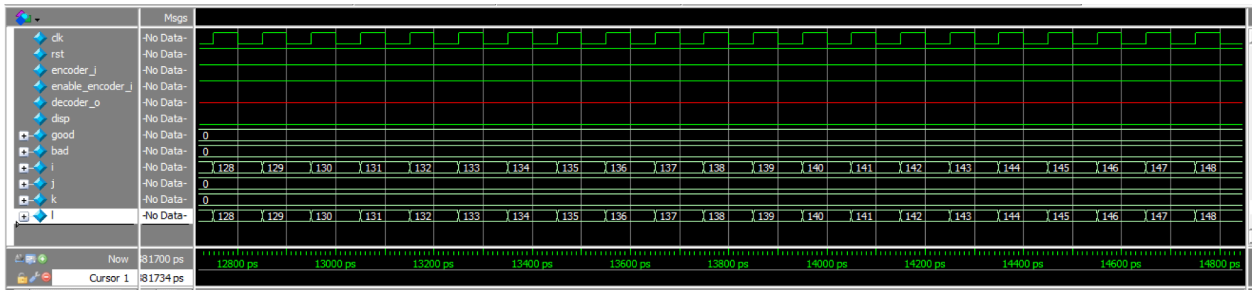

Msgs
clk
rst
encoder_i
enable_encoder_i
decoder_o
disp
good
bad
i
j
k
l
-No Data-
0
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
Now
81700 ps
Cursor 1
81734 ps
4000 ps 4200 ps 4400 ps 4600 ps 4800 ps 5000 ps 5200 ps 5400 ps 5600 ps 5800 ps 6000 ps

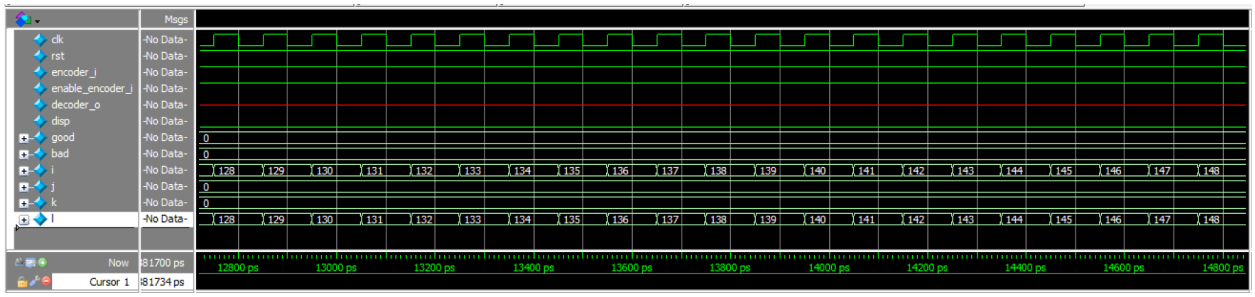

Msgs
clk
rst
encoder_i
enable_encoder_i
decoder_o
disp
good
bad
i
j
k
l
-No Data-
0
60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
Now
81700 ps
Cursor 1
81734 ps
6000 ps 6200 ps 6400 ps 6600 ps 6800 ps 7000 ps 7200 ps 7400 ps 7600 ps 7800 ps 8000 ps

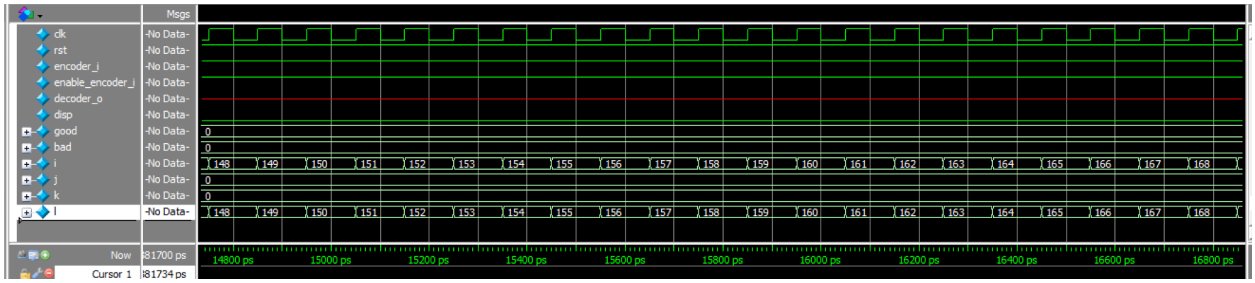

Msgs
clk
rst
encoder_i
enable_encoder_i
decoder_o
disp
good
bad
i
j
k
l
-No Data-
0
80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
Now
81700 ps
Cursor 1
81734 ps
8000 ps 8200 ps 8400 ps 8600 ps 8800 ps 9000 ps 9200 ps 9400 ps 9600 ps 9800 ps 10000 ps

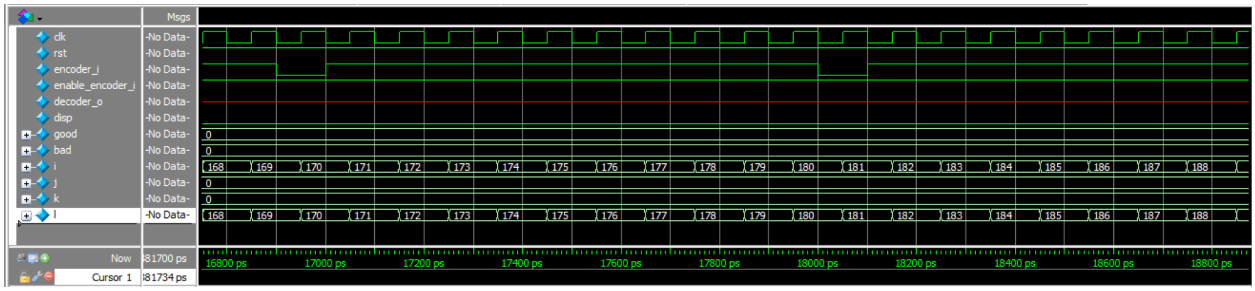

Msgs
clk
rst
encoder_i
enable_encoder_i
decoder_o
disp
good
bad
i
j
k
l
-No Data-
0
99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119
Now
81700 ps
Cursor 1
81734 ps
10000 ps 10200 ps 10400 ps 10600 ps 10800 ps 11000 ps 11200 ps 11400 ps 11600 ps 11800 ps

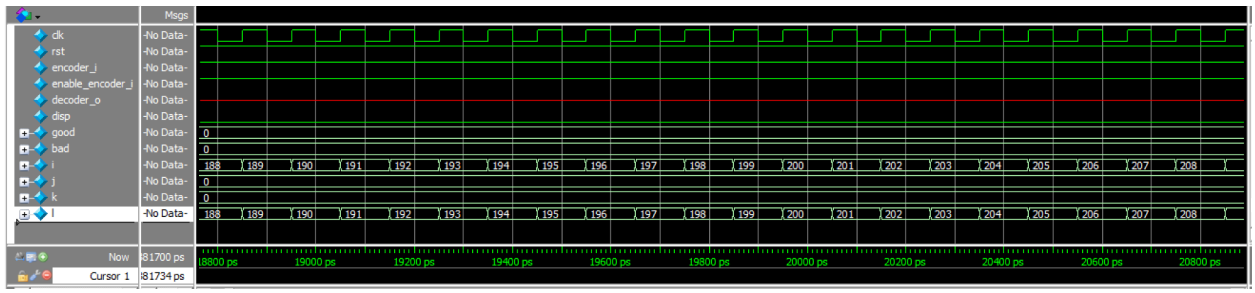

Msgs
clk
rst
encoder_i
enable_encoder_i
decoder_o
disp
good
bad
i
j
k
l
-No Data-
0
108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128
Now
81700 ps
Cursor 1
81734 ps
10800 ps 11000 ps 11200 ps 11400 ps 11600 ps 11800 ps 12000 ps 12200 ps 12400 ps 12600 ps 12800 ps

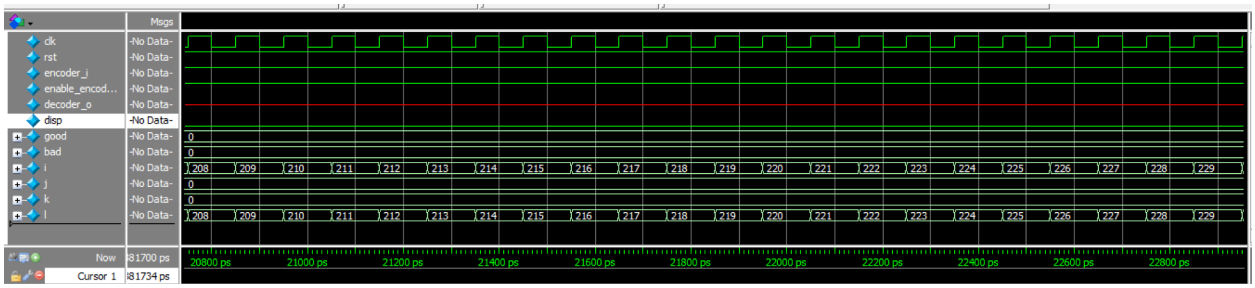

Msgs
clk
rst
encoder_i
enable_encoder_i
decoder_o
disp
good
bad
i
j
k
l
-No Data-
Now
81700 ps
Cursor 1
81734 ps
12800 ps
13000 ps
13200 ps
13400 ps
13600 ps
13800 ps
14000 ps
14200 ps
14400 ps
14600 ps
14800 ps

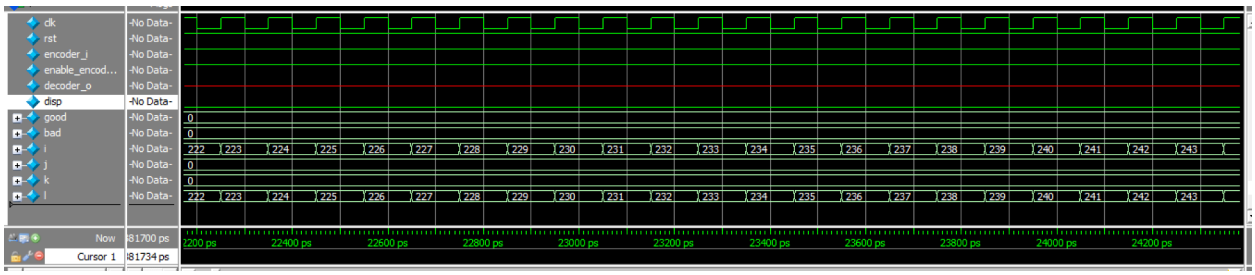

Msgs
clk
rst
encoder_i
enable_encoder_i
decoder_o
disp
good
bad
i
j
k
l
-No Data-
Now
81700 ps
Cursor 1
81734 ps
12800 ps
13000 ps
13200 ps
13400 ps
13600 ps
13800 ps
14000 ps
14200 ps
14400 ps
14600 ps
14800 ps

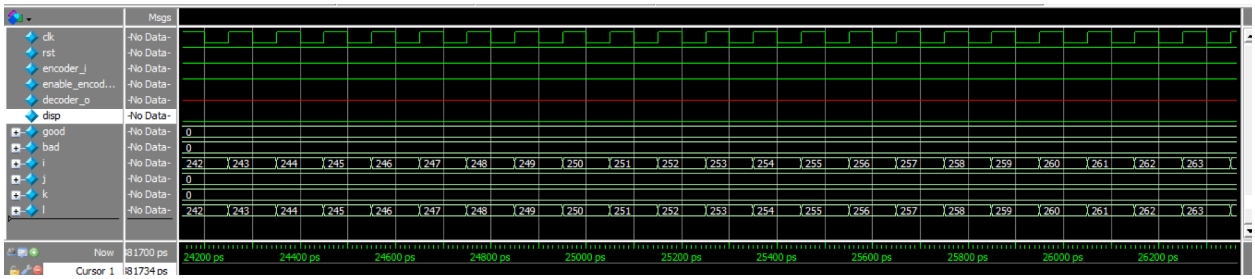

Msgs
clk
rst
encoder_i
enable_encoder_i
decoder_o
disp
good
bad
i
j
k
l
-No Data-
Now
81700 ps
Cursor 1
81734 ps
14800 ps
15000 ps
15200 ps
15400 ps
15600 ps
15800 ps
16000 ps
16200 ps
16400 ps
16600 ps
16800 ps

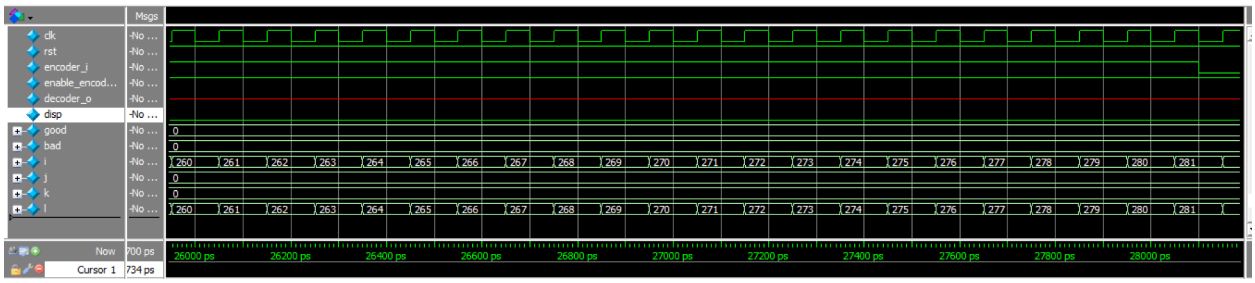

Msgs
clk
rst
encoder_i
enable_encoder_i
decoder_o
disp
good
bad
i
j
k
l
-No Data-
Now
81700 ps
Cursor 1
81734 ps
16800 ps
17000 ps
17200 ps
17400 ps
17600 ps
17800 ps
18000 ps
18200 ps
18400 ps
18600 ps
18800 ps

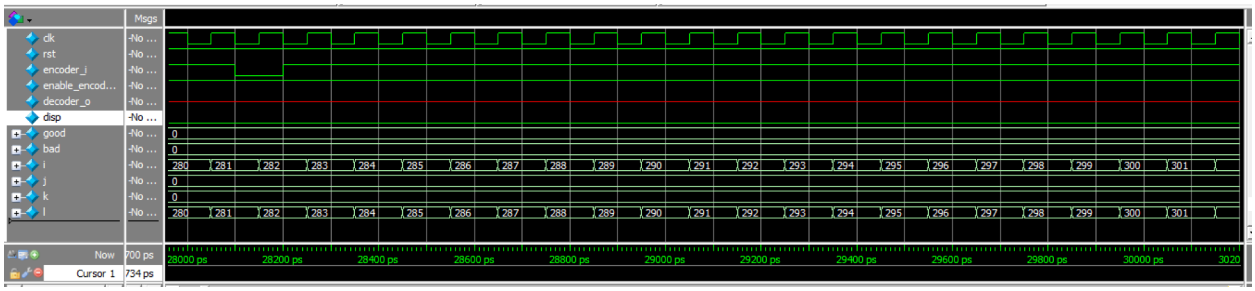

Msgs
clk
rst
encoder_i
enable_encoder_i
decoder_o
disp
good
bad
i
j
k
l
-No Data-
Now
81700 ps
Cursor 1
81734 ps
18800 ps
19000 ps
19200 ps
19400 ps
19600 ps
19800 ps
20000 ps
20200 ps
20400 ps
20600 ps
20800 ps

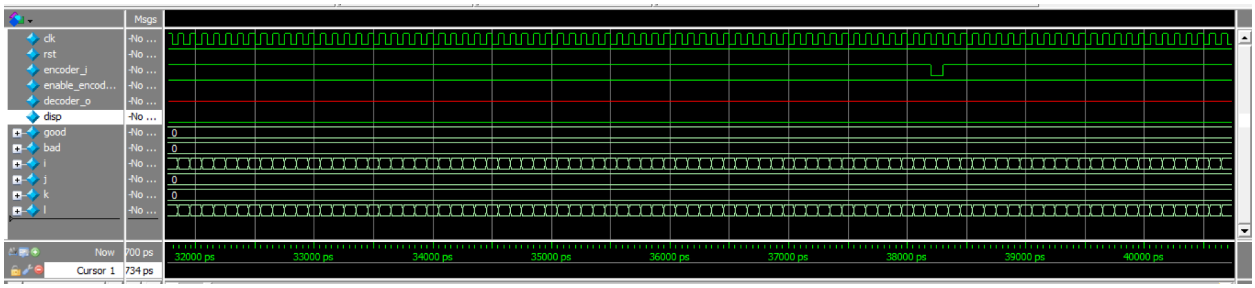

Msgs
clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
-No Data-
0
208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229
Now
81700 ps
Cursor 1
81734 ps
20800 ps 21000 ps 21200 ps 21400 ps 21600 ps 21800 ps 22000 ps 22200 ps 22400 ps 22600 ps 22800 ps

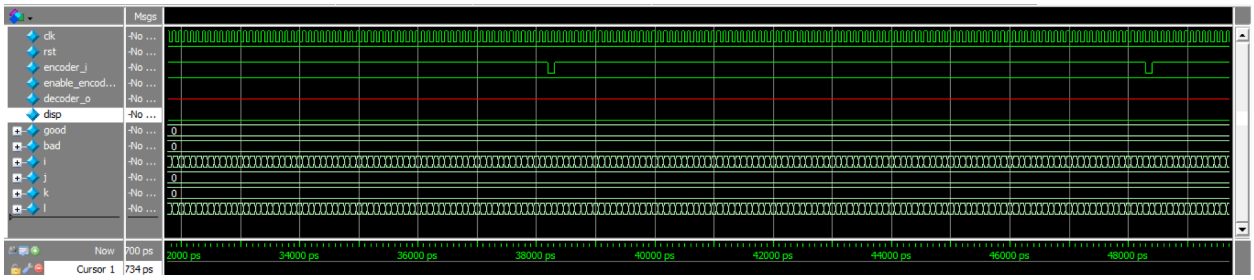

clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
-No Data-
0
222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243
Now
81700 ps
Cursor 1
81734 ps
200 ps 22400 ps 22600 ps 22800 ps 23000 ps 23200 ps 23400 ps 23600 ps 23800 ps 24000 ps 24200 ps

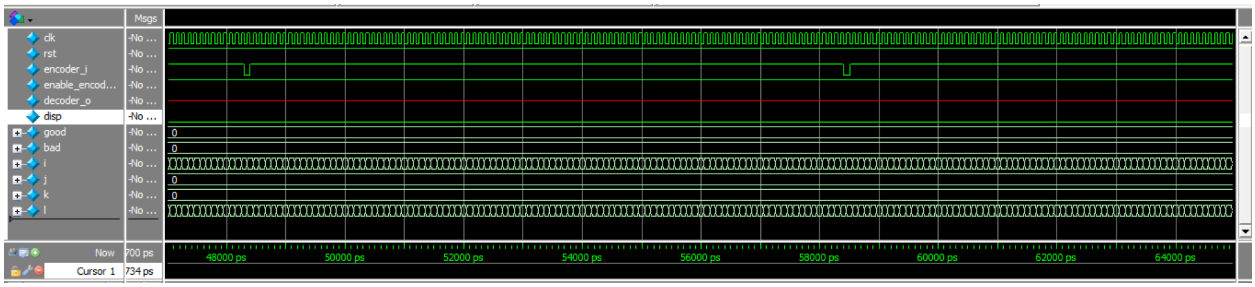

clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
-No Data-
0
242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263
Now
81700 ps
Cursor 1
81734 ps
24200 ps 24400 ps 24600 ps 24800 ps 25000 ps 25200 ps 25400 ps 25600 ps 25800 ps 26000 ps 26200 ps

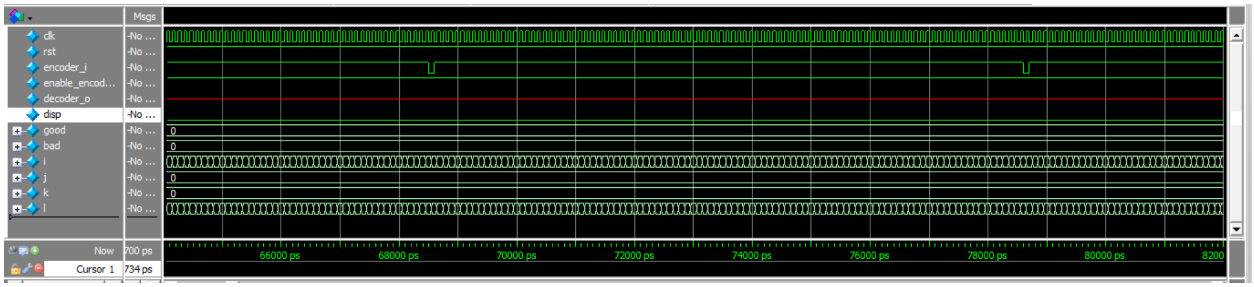

clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
-No ...
0
260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281
Now
700 ps
Cursor 1
734 ps
26000 ps 26200 ps 26400 ps 26600 ps 26800 ps 27000 ps 27200 ps 27400 ps 27600 ps 27800 ps 28000 ps

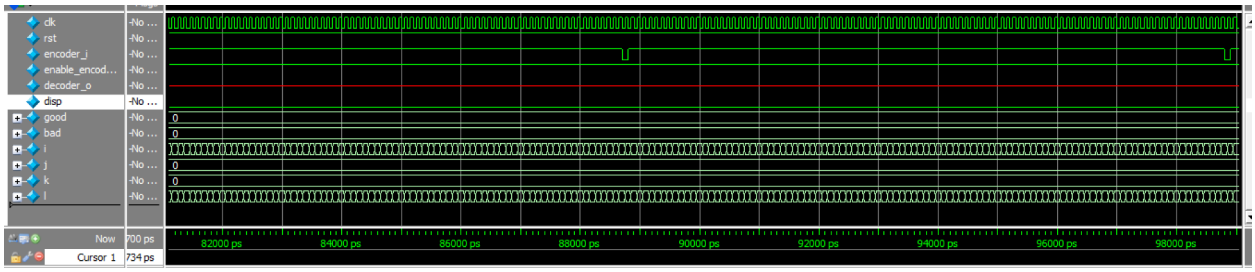

clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
-No ...
0
280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301
Now
700 ps
Cursor 1
734 ps
28000 ps 28200 ps 28400 ps 28600 ps 28800 ps 29000 ps 29200 ps 29400 ps 29600 ps 29800 ps 30000 ps

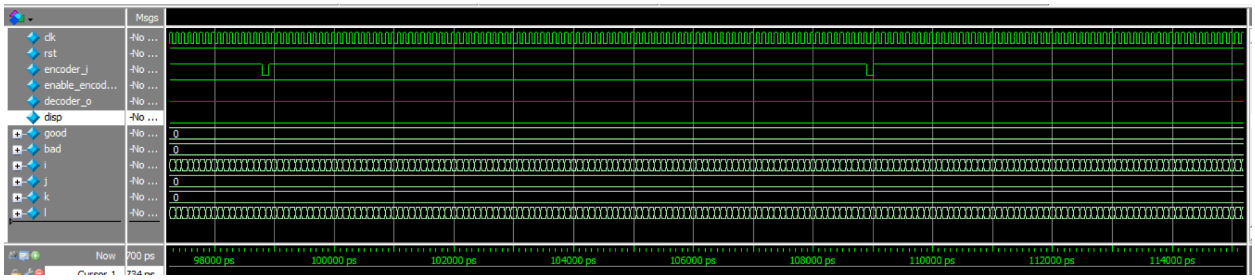

Msgs
clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
Now
700 ps
Cursor 1
734 ps
32000 ps
33000 ps
34000 ps
35000 ps
36000 ps
37000 ps
38000 ps
39000 ps
40000 ps

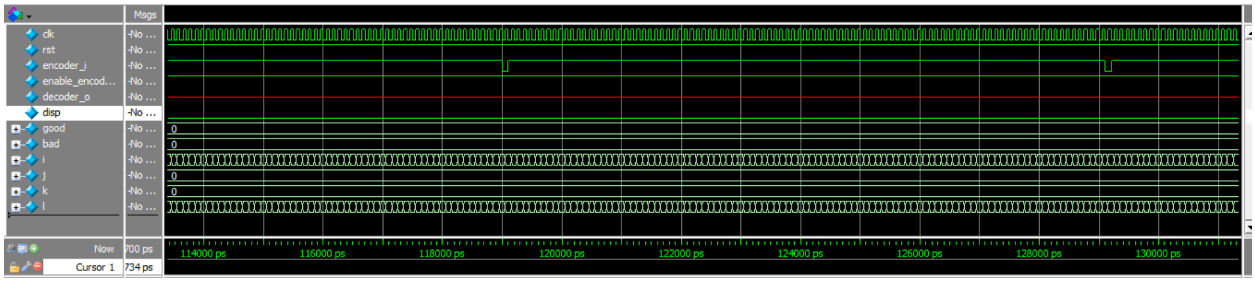

Msgs
clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
Now
700 ps
Cursor 1
734 ps
34000 ps
36000 ps
38000 ps
40000 ps
42000 ps
44000 ps
46000 ps
48000 ps

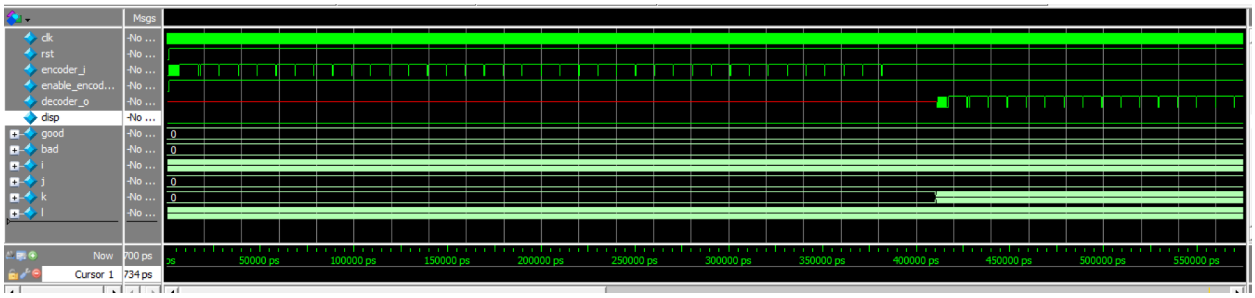

Msgs
clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
Now
700 ps
Cursor 1
734 ps
48000 ps
50000 ps
52000 ps
54000 ps
56000 ps
58000 ps
60000 ps
62000 ps
64000 ps

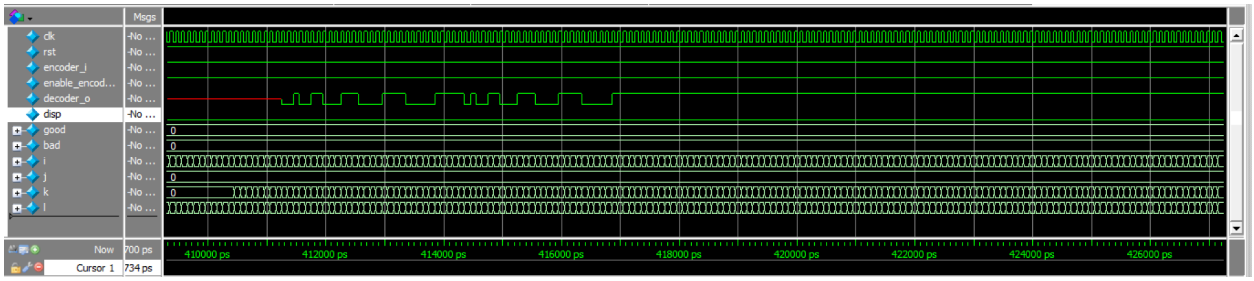

Msgs
clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
Now
700 ps
Cursor 1
734 ps
66000 ps
68000 ps
70000 ps
72000 ps
74000 ps
76000 ps
78000 ps
80000 ps

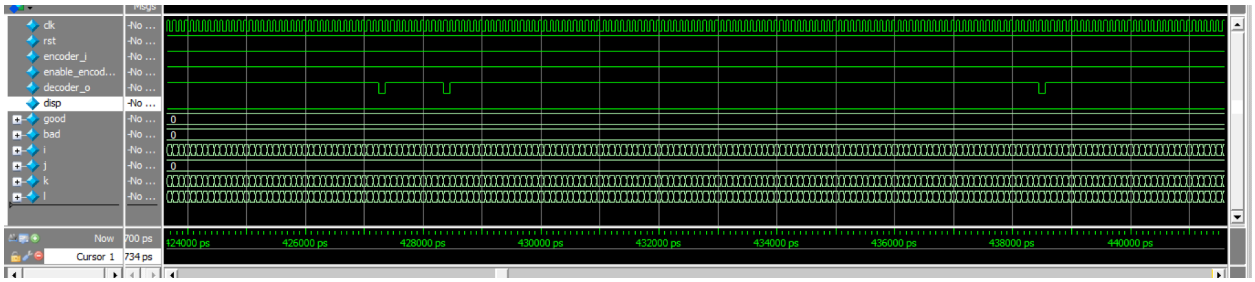

Msgs
clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
Now
700 ps
Cursor 1
734 ps
82000 ps
84000 ps
86000 ps
88000 ps
90000 ps
92000 ps
94000 ps
96000 ps
98000 ps

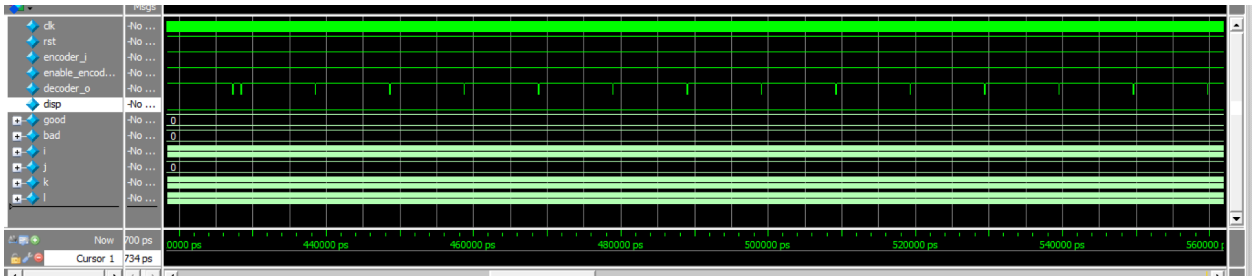

Msgs
clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
Now
700 ps
98000 ps
100000 ps
102000 ps
104000 ps
106000 ps
108000 ps
110000 ps
112000 ps
114000 ps

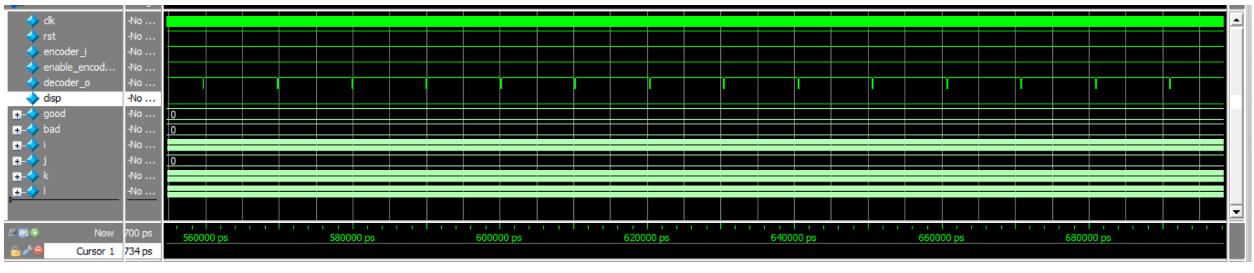

Msgs
clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
Now
Cursor 1
114000 ps
116000 ps
118000 ps
120000 ps
122000 ps
124000 ps
126000 ps
128000 ps
130000 ps

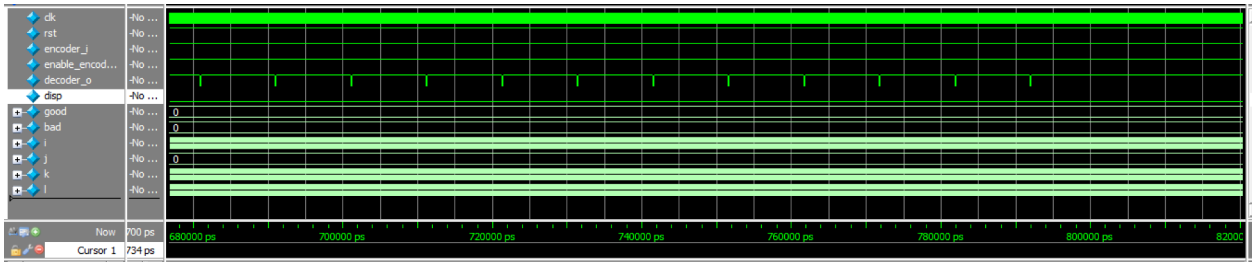

Msgs
clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
Now
Cursor 1
50000 ps
100000 ps
150000 ps
200000 ps
250000 ps
300000 ps
350000 ps
400000 ps
450000 ps
500000 ps
550000 ps

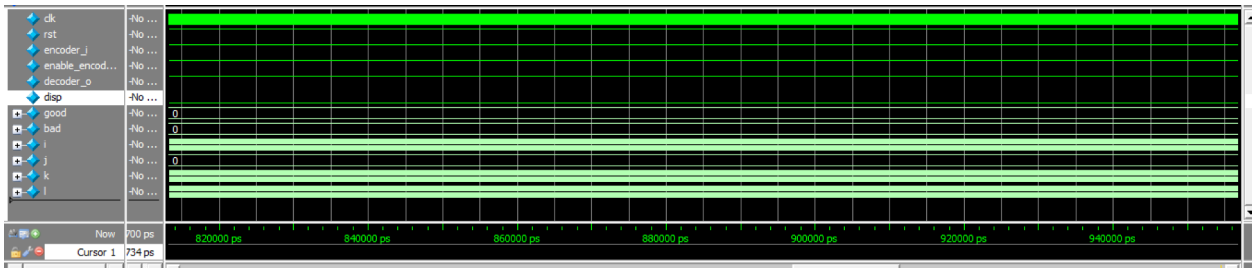

Msgs
clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
Now
Cursor 1
410000 ps
412000 ps
414000 ps
416000 ps
418000 ps
420000 ps
422000 ps
424000 ps
426000 ps

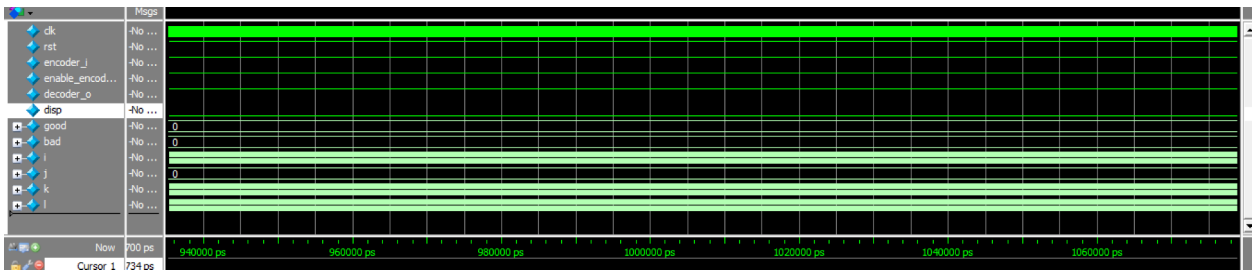

clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
Now
Cursor 1
424000 ps
426000 ps
428000 ps
430000 ps
432000 ps
434000 ps
436000 ps
438000 ps
440000 ps

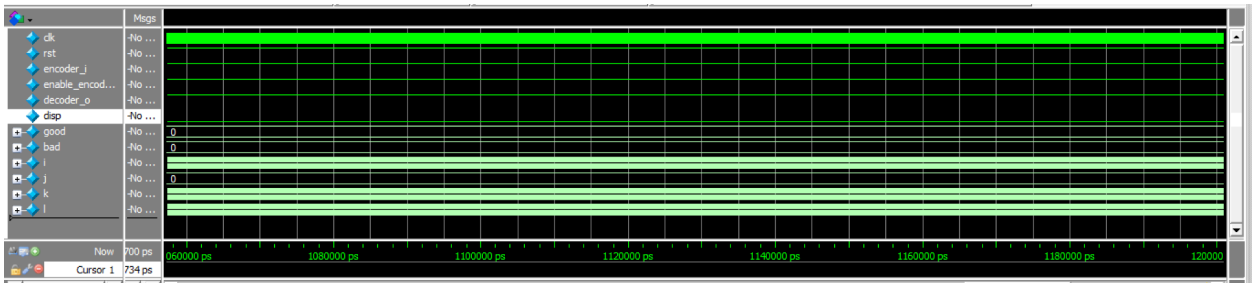

clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
Now
Cursor 1
440000 ps
460000 ps
480000 ps
500000 ps
520000 ps
540000 ps
560000

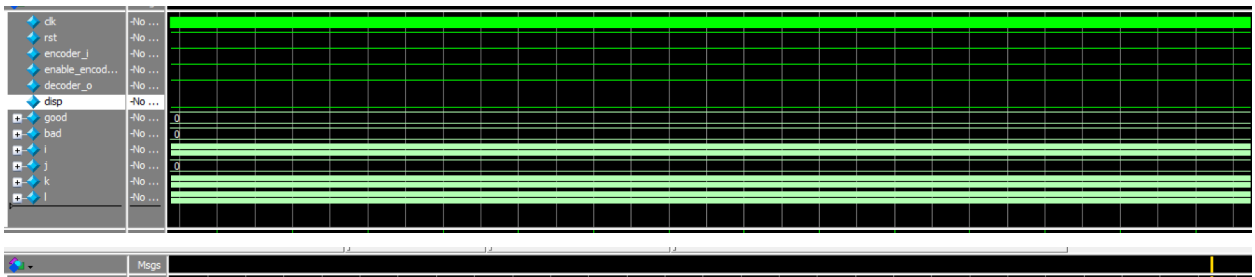

clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
Now
Cursor 1
560000 ps
580000 ps
600000 ps
620000 ps
640000 ps
660000 ps
680000 ps

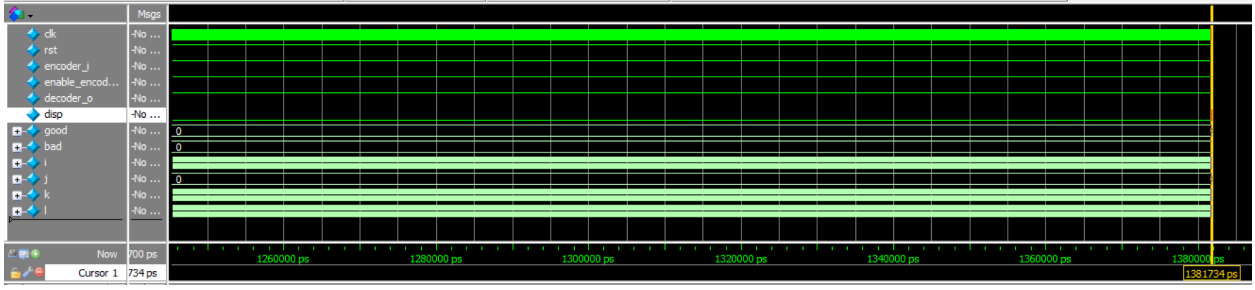

clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
Now
Cursor 1
680000 ps
700000 ps
720000 ps
740000 ps
760000 ps
780000 ps
800000 ps


clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
Now
Cursor 1
820000 ps
840000 ps
860000 ps
880000 ps
900000 ps
920000 ps
940000 ps


Msgs
clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
Now
Cursor 1
940000 ps
960000 ps
980000 ps
1000000 ps
1020000 ps
1040000 ps
1060000 ps


Msgs
clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
Now
Cursor 1
1060000 ps
1080000 ps
1100000 ps
1120000 ps
1140000 ps
1160000 ps
1180000 ps


clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l


Msgs
clk
rst
encoder_i
enable_encod...
decoder_o
disp
good
bad
i
j
k
l
Now
Cursor 1
1260000 ps
1280000 ps
1300000 ps
1320000 ps
1340000 ps
1360000 ps
1380000 ps
1381734 ps

Transcript

```
# Top level modules:
#         viterbi_tx_rx_tb
# End time: 22:59:04 on Mar 23,2023, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
vsim work.viterbi_tx_rx_tb
# vsim work.viterbi_tx_rx_tb
# Start time: 22:59:09 on Mar 23,2023
# Loading sv_std.std
# Loading work.viterbi_tx_rx_tb
# Loading work.viterbi_tx_rx
# Loading work.encoder
# Loading work.decoder
# Loading work.bmc000
# Loading work.bmc001
# Loading work.bmc010
# Loading work.bmc011
# Loading work.bmc100
# Loading work.bmc101
# Loading work.bmc110
# Loading work.bmc111
# Loading work.ACS
# Loading work.mem
# Loading work.tbu
# Loading work.mem_disp
add wave -position insertpoint sim:/viterbi_tx_rx_tb/*
# ** Warning: (vsim-WLF-5000) WLF file currently in use: vsim.wlf
#           File in use by: ITSloaner  Hostname: DESKTOP-F1ESCSS  ProcessID: 8832
#           Attempting to use alternate WLF file "./wlftesmf0j".
# ** Warning: (vsim-WLF-5001) Could not open WLF file: vsim.wlf
#           Using alternate file: ./wlftesmf0j

run -all
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 1, out = 1
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 1, out = 1
```

```
# yaa! in = 1, out = 1
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 0, out = 0
# yaa! in = 1, out = 1
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
# yaa! in = 0, out = 0
```

```
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
```

```
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
```

```
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 0, out = 0
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 0, out = 0
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
```

```
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# yaa! in = 1, out = 1
# good =         256, bad =           0
# ** Note: $stop    :
C:/Users/ITSloaner.DESKTOP-F1ESCSS.000/Downloads/vitassign/viterbi_tx_rx_tb.sv(371)
#    Time: 1381700 ps  Iteration: 0  Instance: /viterbi_tx_rx_tb
# Break in Module viterbi_tx_rx_tb at
C:/Users/ITSloaner.DESKTOP-F1ESCSS.000/Downloads/vitassign/viterbi_tx_rx_tb.sv line
371
```

SystemVerilog Code

Design (Assignment)

decoder.sv

```
`include "ACS.sv"
`include "bmc000.sv"
`include "bmc001.sv"
`include "bmc010.sv"
`include "bmc011.sv"
`include "bmc100.sv"
`include "bmc101.sv"
`include "bmc110.sv"
`include "bmc111.sv"
`include "mem_1x1024.sv"
`include "mem_8x1024.sv"
`include "tbu.sv"

module decoder
(
  input             clk,
  input             rst,
  input             enable,
  input [1:0]       d_in,
  output logic      d_out);

//   logic             decoder_o_reg;

//bmc module signals
  wire  [1:0]       bmc000_path_0_bmc;
  wire  [1:0]       bmc001_path_0_bmc;
  wire  [1:0]       bmc010_path_0_bmc;
  wire  [1:0]       bmc011_path_0_bmc;
  wire  [1:0]       bmc100_path_0_bmc;
  wire  [1:0]       bmc101_path_0_bmc;
  wire  [1:0]       bmc110_path_0_bmc;
  wire  [1:0]       bmc111_path_0_bmc;

  wire  [1:0]       bmc000_path_1_bmc;
  wire  [1:0]       bmc001_path_1_bmc;
```

```
  wire  [1:0]       bmc010_path_1_bmc;
  wire  [1:0]       bmc011_path_1_bmc;
  wire  [1:0]       bmc100_path_1_bmc;
  wire  [1:0]       bmc101_path_1_bmc;
  wire  [1:0]       bmc110_path_1_bmc;
  wire  [1:0]       bmc111_path_1_bmc;

//ACS modules signals
  logic   [7:0]       validity;
  logic   [7:0]       selection;
  logic   [7:0]       path_cost   [8];
  wire    [7:0]       validity_nets;
  wire    [7:0]       selection_nets;

  wire              ACS000_selection;
  wire              ACS001_selection;
  wire              ACS010_selection;
  wire              ACS011_selection;
  wire              ACS100_selection;
  wire              ACS101_selection;
  wire              ACS110_selection;
  wire              ACS111_selection;

  wire              ACS000_valid_o;
  wire              ACS001_valid_o;
  wire              ACS010_valid_o;
  wire              ACS011_valid_o;
  wire              ACS100_valid_o;
  wire              ACS101_valid_o;
  wire              ACS110_valid_o;
  wire              ACS111_valid_o;

  wire  [7:0]       ACS000_path_cost;
  wire  [7:0]       ACS001_path_cost;
  wire  [7:0]       ACS010_path_cost;
  wire  [7:0]       ACS011_path_cost;
  wire  [7:0]       ACS100_path_cost;
  wire  [7:0]       ACS101_path_cost;
  wire  [7:0]       ACS110_path_cost;
  wire  [7:0]       ACS111_path_cost;
```

```
//Trelis memory write operation
  logic   [1:0]       mem_bank;
  logic   [1:0]       mem_bank_buf;
  logic   [1:0]       mem_bank_buf_buf;
  logic               mem_bank_buf_buf_buf;
  logic               mem_bank_buf_buf_buf_buf;
  logic               mem_bank_buf_buf_buf_buf_buf;
  logic   [9:0]       wr_mem_counter;
  logic   [9:0]       rd_mem_counter;

  logic   [9:0]       addr_mem_A;
  logic   [9:0]       addr_mem_B;
  logic   [9:0]       addr_mem_C;
  logic   [9:0]       addr_mem_D;

  logic               wr_mem_A;
  logic               wr_mem_B;
  logic               wr_mem_C;
  logic               wr_mem_D;

  logic   [7:0]       d_in_mem_A;
  logic   [7:0]       d_in_mem_B;
  logic   [7:0]       d_in_mem_C;
  logic   [7:0]       d_in_mem_D;

  wire    [7:0]       d_o_mem_A;
  wire    [7:0]       d_o_mem_B;
  wire    [7:0]       d_o_mem_C;
  wire    [7:0]       d_o_mem_D;

//Trace back module signals
  logic               selection_tbu_0;
  logic               selection_tbu_1;

  logic   [7:0]       d_in_0_tbu_0;
  logic   [7:0]       d_in_1_tbu_0;
  logic   [7:0]       d_in_0_tbu_1;
  logic   [7:0]       d_in_1_tbu_1;

  wire                d_o_tbu_0;
  wire                d_o_tbu_1;
```

```
  logic               enable_tbu_0;
  logic               enable_tbu_1;

//Display memory operations
  wire                wr_disp_mem_0;
  wire                wr_disp_mem_1;

  wire                d_in_disp_mem_0;
  wire                d_in_disp_mem_1;

  logic   [9:0]       wr_mem_counter_disp;
  logic   [9:0]       rd_mem_counter_disp;

  logic   [9:0]       addr_disp_mem_0;
  logic   [9:0]       addr_disp_mem_1;

//Branch matrc calculation modules

  bmc000   bmc000_inst(d_in,bmc000_path_0_bmc,bmc000_path_1_bmc);
/*  similarly for bmc001 through 111
*/
  bmc001   bmc001_inst(d_in,bmc001_path_0_bmc,bmc001_path_1_bmc);
  bmc010   bmc010_inst(d_in,bmc010_path_0_bmc,bmc010_path_1_bmc);
  bmc011   bmc011_inst(d_in,bmc011_path_0_bmc,bmc011_path_1_bmc);
  bmc100   bmc100_inst(d_in,bmc100_path_0_bmc,bmc100_path_1_bmc);
  bmc101   bmc101_inst(d_in,bmc101_path_0_bmc,bmc101_path_1_bmc);
  bmc110   bmc110_inst(d_in,bmc110_path_0_bmc,bmc110_path_1_bmc);
  bmc111   bmc111_inst(d_in,bmc111_path_0_bmc,bmc111_path_1_bmc);


//Add Compare Select Modules
  ACS
ACS000(validity[0],validity[1],bmc000_path_0_bmc,bmc000_path_1_bmc,path_cost[0],path
_cost[1],ACS000_selection,ACS000_valid_o,ACS000_path_cost);
  ACS
ACS001(validity[3],validity[2],bmc001_path_0_bmc,bmc001_path_1_bmc,path_cost[3],path
_cost[2],ACS001_selection,ACS001_valid_o,ACS001_path_cost);
  ACS
ACS010(validity[4],validity[5],bmc010_path_0_bmc,bmc010_path_1_bmc,path_cost[4],path
_cost[5],ACS010_selection,ACS010_valid_o,ACS010_path_cost);
```

```
  ACS
ACS011(validity[7],validity[6],bmc011_path_0_bmc,bmc011_path_1_bmc,path_cost[7],path
_cost[6],ACS011_selection,ACS011_valid_o,ACS011_path_cost);
  ACS
ACS100(validity[1],validity[0],bmc100_path_0_bmc,bmc100_path_1_bmc,path_cost[1],path
_cost[0],ACS100_selection,ACS100_valid_o,ACS100_path_cost);
  ACS
ACS101(validity[2],validity[3],bmc101_path_0_bmc,bmc101_path_1_bmc,path_cost[2],path
_cost[3],ACS101_selection,ACS101_valid_o,ACS101_path_cost);
  ACS
ACS110(validity[5],validity[4],bmc110_path_0_bmc,bmc110_path_1_bmc,path_cost[5],path
_cost[4],ACS110_selection,ACS110_valid_o,ACS110_path_cost);
  ACS
ACS111(validity[6],validity[7],bmc111_path_0_bmc,bmc111_path_1_bmc,path_cost[6],path
_cost[7],ACS111_selection,ACS111_valid_o,ACS111_path_cost);

  assign selection_nets  =
{ACS111_selection,ACS110_selection,ACS101_selection,ACS100_selection,

ACS011_selection,ACS010_selection,ACS001_selection,ACS000_selection};
  assign validity_nets    =
{ACS111_valid_o,ACS110_valid_o,ACS101_valid_o,ACS100_valid_o,

ACS011_valid_o,ACS010_valid_o,ACS001_valid_o,ACS000_valid_o};


  always @ (posedge clk, negedge rst) begin
     if(!rst)  begin
        validity          <= 8'b00000001;
        selection         <= 8'b00000000;
/* clear all 8 path costs
        path_cost[i]      <= 8'd0;
*/
       path_cost[0]      <= 8'd0;
        path_cost[1]      <= 8'd0;
        path_cost[2]      <= 8'd0;
        path_cost[3]      <= 8'd0;
        path_cost[4]      <= 8'd0;
        path_cost[5]      <= 8'd0;
        path_cost[6]      <= 8'd0;
        path_cost[7]      <= 8'd0;
```

```
    end
    else if(!enable)   begin
       validity          <= 8'b00000001;
       selection         <= 8'b00000000;
/* clear all 8 path costs
       path_cost[i]      <= 8'd0;
*/
       path_cost[0]      <= 8'd0;
       path_cost[1]      <= 8'd0;
       path_cost[2]      <= 8'd0;
       path_cost[3]      <= 8'd0;
       path_cost[4]      <= 8'd0;
       path_cost[5]      <= 8'd0;
       path_cost[6]      <= 8'd0;
       path_cost[7]      <= 8'd0;
    end
    else if( path_cost[0][7] && path_cost[1][7] && path_cost[2][7] &&
path_cost[3][7] &&
           path_cost[4][7] && path_cost[5][7] && path_cost[6][7] && path_cost[7][7]
)
    begin

       validity          <= validity_nets;
       selection         <= selection_nets;

       path_cost[0]      <= 8'b01111111 & ACS000_path_cost;
/*  likewise for path_cost[1:7] and ACS001:111_path_cost
*/
       path_cost[1]      <= 8'b01111111 & ACS001_path_cost;
       path_cost[2]      <= 8'b01111111 & ACS010_path_cost;
       path_cost[3]      <= 8'b01111111 & ACS011_path_cost;
       path_cost[4]      <= 8'b01111111 & ACS100_path_cost;
       path_cost[5]      <= 8'b01111111 & ACS101_path_cost;
       path_cost[6]      <= 8'b01111111 & ACS110_path_cost;
       path_cost[7]      <= 8'b01111111 & ACS111_path_cost;

    end
    else   begin
       validity          <= validity_nets;
       selection         <= selection_nets;
```

```
        path_cost[0]      <= ACS000_path_cost;
/* likewise for 1:7
*/
        path_cost[1]      <= ACS001_path_cost;
        path_cost[2]      <= ACS010_path_cost;
        path_cost[3]      <= ACS011_path_cost;
        path_cost[4]      <= ACS100_path_cost;
        path_cost[5]      <= ACS101_path_cost;
        path_cost[6]      <= ACS110_path_cost;
        path_cost[7]      <= ACS111_path_cost;
     end
  end

  always @ (posedge clk, negedge rst) begin
     if(!rst)
        wr_mem_counter <= 10'd0;
     else if(!enable)
        wr_mem_counter <= 10'd0;
     else
        wr_mem_counter <= wr_mem_counter + 10'd1;
  end

  always @ (posedge clk, negedge rst) begin
     if(!rst)
        rd_mem_counter <= 10'b1111111111; // -1   how do you handle this in 10 bit
binary?
     else if(enable)
        rd_mem_counter <= rd_mem_counter - 10'd1;
  end

  always @ (posedge clk, negedge rst)
     if(!rst)
        mem_bank <= 2'b00;
     else begin
        if(wr_mem_counter==10'b1111111111)
              mem_bank <= mem_bank + 2'b01;
     end

  always @ (posedge clk)    begin
     d_in_mem_A  <= selection;
     d_in_mem_B  <= selection;
```

```
    d_in_mem_C  <= selection;
    d_in_mem_D  <= selection;
  end

  always @ (posedge clk)     begin
    case(mem_bank)
       2'b00:          begin
          addr_mem_A        <= wr_mem_counter;
          addr_mem_B        <= rd_mem_counter;
          addr_mem_C        <= 10'd0;
          addr_mem_D        <= rd_mem_counter;

          wr_mem_A          <= 1'b1;
/* other wr_mems = 0
  */        wr_mem_B           <= 1'b0;
          wr_mem_C           <= 1'b0;
          wr_mem_D           <= 1'b0;
       end
       2'b01:          begin
          addr_mem_A        <= rd_mem_counter;
          addr_mem_B        <= wr_mem_counter;
          addr_mem_C        <= rd_mem_counter;
          addr_mem_D        <= 10'd0;

          wr_mem_B          <= 1'b1;
/* other wr_mems = 0
*/
          wr_mem_A           <= 1'b0;
          wr_mem_C           <= 1'b0;
          wr_mem_D           <= 1'b0;
       end
       2'b10:    begin
          addr_mem_A        <= 10'd0;
          addr_mem_B        <= rd_mem_counter;
          addr_mem_C        <= wr_mem_counter;
          addr_mem_D        <= rd_mem_counter;

          wr_mem_C       <= 1'b1;
/* other wr_mems = 0
*/
          wr_mem_A           <= 1'b0;
```

```
           wr_mem_B            <= 1'b0;
           wr_mem_D            <= 1'b0;
        end
        2'b11:     begin
           addr_mem_A         <= rd_mem_counter;
           addr_mem_B         <= 10'd0;
           addr_mem_C         <= rd_mem_counter;
           addr_mem_D         <= wr_mem_counter;

           wr_mem_D       <= 1'b1;
/* other wr_mems = 0
*/
           wr_mem_B            <= 1'b0;
           wr_mem_C            <= 1'b0;
           wr_mem_A            <= 1'b0;
        end
     endcase
 end

//Trelis memory module instantiation

  mem   trelis_mem_A
  (
     .clk,
     .wr(wr_mem_A),
     .addr(addr_mem_A),
     .d_i(d_in_mem_A),
     .d_o(d_o_mem_A)
  );
/* likewise for trelis_memB, C, D
*/
  mem   trelis_mem_B
  (
     .clk,
     .wr(wr_mem_B),
     .addr(addr_mem_B),
     .d_i(d_in_mem_B),
     .d_o(d_o_mem_B)
  );

  mem   trelis_mem_C
```

```
  (
     .clk,
     .wr(wr_mem_C),
     .addr(addr_mem_C),
     .d_i(d_in_mem_C),
     .d_o(d_o_mem_C)
  );

  mem   trelis_mem_D
  (
     .clk,
     .wr(wr_mem_D),
     .addr(addr_mem_D),
     .d_i(d_in_mem_D),
     .d_o(d_o_mem_D)
  );
//Trace back module operation

  always @(posedge clk)
     mem_bank_buf   <= mem_bank;

  always @(posedge clk)
     mem_bank_buf_buf   <= mem_bank_buf;

  always @ (posedge clk, negedge rst)
     if(!rst)
           enable_tbu_0   <= 1'b0;
     else begin
        if(mem_bank_buf_buf==2'b10)
           enable_tbu_0   <= 1'b1;
        else
           enable_tbu_0   <= enable_tbu_0;
     end

  always @ (posedge clk, negedge rst)
     if(!rst)
           enable_tbu_1   <= 1'b0;
     else begin
        if(mem_bank_buf_buf==2'b11)
           enable_tbu_1   <= 1'b1;
        else
```

```
            enable_tbu_1    <= enable_tbu_1;
    end

always @ (posedge clk)
    case(mem_bank_buf_buf)
        2'b00:      begin
            d_in_0_tbu_0    <= d_o_mem_D;
            d_in_1_tbu_0    <= d_o_mem_C;

            d_in_0_tbu_1    <= d_o_mem_C;
            d_in_1_tbu_1    <= d_o_mem_B;

            selection_tbu_0<= 1'b0;
            selection_tbu_1<= 1'b1;

        end
        2'b01:      begin
            d_in_0_tbu_0    <= d_o_mem_D;
            d_in_1_tbu_0    <= d_o_mem_C;

            d_in_0_tbu_1    <= d_o_mem_A;
            d_in_1_tbu_1    <= d_o_mem_D;

            selection_tbu_0<= 1'b1;
            selection_tbu_1<= 1'b0;
        end
        2'b10:      begin
            d_in_0_tbu_0    <= d_o_mem_B;
            d_in_1_tbu_0    <= d_o_mem_A;

            d_in_0_tbu_1    <= d_o_mem_A;
            d_in_1_tbu_1    <= d_o_mem_D;

            selection_tbu_0<= 1'b0;
            selection_tbu_1<= 1'b1;
        end
        2'b11:      begin
            d_in_0_tbu_0    <= d_o_mem_B;
            d_in_1_tbu_0    <= d_o_mem_A;

            d_in_0_tbu_1    <= d_o_mem_C;
```

```
          d_in_1_tbu_1   <= d_o_mem_B;

          selection_tbu_0<= 1'b1;
          selection_tbu_1<= 1'b0;
       end
    endcase

//Trace-Back modules instantiation

  tbu tbu_0   (
    .clk,
    .rst,
    .enable(enable_tbu_0),
    .selection(selection_tbu_0),
    .d_in_0(d_in_0_tbu_0),
    .d_in_1(d_in_1_tbu_0),
    .d_o(d_o_tbu_0),
    .wr_en(wr_disp_mem_0)
  );

/* analogous for tbu_1
*/
  tbu tbu_1   (
    .clk,
    .rst,
    .enable(enable_tbu_1),
    .selection(selection_tbu_1),
    .d_in_0(d_in_0_tbu_1),
    .d_in_1(d_in_1_tbu_1),
    .d_o(d_o_tbu_1),
    .wr_en(wr_disp_mem_1)
  );
//Display Memory modules Instantioation

  assign   d_in_disp_mem_0   =  d_o_tbu_0;
  assign   d_in_disp_mem_1   =  d_o_tbu_1;

 mem_disp   disp_mem_0
 (
    .clk              ,
    .wr(wr_disp_mem_0),
```

```
    .addr(addr_disp_mem_0),
    .d_i(d_in_disp_mem_0),
    .d_o(d_o_disp_mem_0)
 );
/* analogous for disp_mem_1
*/
 mem_disp   disp_mem_1
 (
    .clk              ,
    .wr(wr_disp_mem_1),
    .addr(addr_disp_mem_1),
    .d_i(d_in_disp_mem_1),
    .d_o(d_o_disp_mem_1)
 );
// Display memory module operation
 always @ (posedge clk)
    mem_bank_buf_buf_buf <= mem_bank_buf_buf[0];

 always @ (posedge clk)
    if(!rst)
       wr_mem_counter_disp  <= 10'b0000000010;
    else if(!enable)
       wr_mem_counter_disp  <= 10'b0000000010;
    else
       wr_mem_counter_disp  <= wr_mem_counter_disp - 10'd1;

 always @ (posedge clk)
    if(!rst)
       rd_mem_counter_disp  <= 10'b1111111101;
    else if(!enable)
       rd_mem_counter_disp  <= 10'b1111111101;
    else
       rd_mem_counter_disp  <= rd_mem_counter_disp + 10'd1;

 always @ (posedge clk)
    case(mem_bank_buf_buf_buf)
       1'b0:
       begin
          addr_disp_mem_0   <= rd_mem_counter_disp;
          addr_disp_mem_1   <= wr_mem_counter_disp;
       end
```

```
        1'b1: //swap rd and wr
        begin
           addr_disp_mem_0   <= wr_mem_counter_disp;
           addr_disp_mem_1   <= rd_mem_counter_disp;
        end
     endcase

  always @ (posedge clk) begin
     mem_bank_buf_buf_buf_buf     <= mem_bank_buf_buf_buf;
     mem_bank_buf_buf_buf_buf_buf <= mem_bank_buf_buf_buf_buf;
  end

  always @ (posedge clk) begin
/*  d_out = d_o_disp_mem_i
   i = mem_bank_buf_buf_buf_buf_buf
*/
    d_out <= (mem_bank_buf_buf_buf_buf_buf)? d_o_disp_mem_1:d_o_disp_mem_0;
 end


endmodule
```

ACS.sv

```
module ACS                              // add-compare-select
(
  input       path_0_valid,
  input       path_1_valid,
  input [1:0] path_0_bmc,             // branch metric computation
  input [1:0] path_1_bmc,
  input [7:0] path_0_pmc,           // path metric computation
  input [7:0] path_1_pmc,

  output logic        selection,
  output logic        valid_o,
  output      [7:0] path_cost);

  wire  [7:0] path_cost_0;            // branch metric + path metric
  wire  [7:0] path_cost_1;
```

```
/* Fill in the guts per ACS instructions
*/
  assign valid_o = (path_0_valid || path_1_valid) ? 1:0;

  assign selection = ((!path_0_valid && path_1_valid) || (path_0_valid &&
path_1_valid && (path_cost_0 > path_cost_1)))? 1:0;

  assign path_cost_0 = path_0_pmc + path_0_bmc;
  assign path_cost_1 = path_1_pmc + path_1_bmc;
   assign path_cost = (!valid_o)? 0: (selection)? path_cost_1:path_cost_0;

endmodule
```

tbu.sv

```
module tbu
(
  input       clk,
  input       rst,
  input       enable,
  input       selection,
  input [7:0] d_in_0,
  input [7:0] d_in_1,
  output logic  d_o,
  output logic  wr_en);

  logic         d_o_reg;
  logic         wr_en_reg;

  logic   [2:0] pstate;
  logic   [2:0] nstate;

  logic         selection_buf;

  always @(posedge clk)    begin
     selection_buf  <= selection;
     wr_en          <= wr_en_reg;
     d_o            <= d_o_reg;
  end
```

```
  always @(posedge clk, negedge rst) begin
     if(!rst)
        pstate   <= 3'b000;
     else if(!enable)
        pstate   <= 3'b000;
     else if(selection_buf && !selection)
        pstate   <= 3'b000;
     else
        pstate   <= nstate;
  end

/*  combinational logic drives:
wr_en_reg, d_o_reg, nstate (next state)
from selection, d_in_1[pstate], d_in_0[pstate]
See assignment text for details
*/
  always_comb begin
     wr_en_reg = selection;
     d_o_reg = (selection) ? d_in_1[pstate]:0;

     if (!selection) begin
       nstate = {pstate[1], pstate[0], d_in_0[pstate] ^ pstate[2] ^ pstate[0]};
     end
     else if (selection) begin
       nstate = {pstate[1], pstate[0], d_in_1[pstate] ^ pstate[2] ^ pstate[0]};
     end
    else
      nstate = pstate;
  end

endmodule
```

Encoder.sv

```
// figure out what this encoder does -- differs a bit from Homework 7
module encoder                    // use this one
(  input             clk,
  input             rst,
  input             enable_i,
  input             d_in,
```

```
  output logic      valid_o,
  output      [1:0] d_out);

  logic         [2:0] cstate;
  logic         [2:0] nstate;

  logic         [1:0] d_out_reg;

  assign   d_out    =  (enable_i)? d_out_reg:2'b00;

  always_comb begin
     valid_o  =   enable_i;
// fill in the guts
     nstate = {d_in ^ (cstate[1] ^ cstate[0]), cstate[2], cstate[1]};
     d_out_reg = {d_in ^ (cstate[2]^cstate[1]), d_in};
  end

  always @ (posedge clk,negedge rst)   begin
//      $display("data in=%d state=%b%b%b data
out=%b%b",d_in,reg_1,reg_2,reg_3,d_out_reg[1],d_out_reg[0]);
     if(!rst)
        cstate   <= 3'b000;
     else if(!enable_i)
        cstate   <= 3'b000;
     else
        cstate   <= nstate;
  end

endmodule
```

Bmc000.sv

```
module bmc000              // branch metric computation
(
  input     [1:0] rx_pair,
  output    [1:0] path_0_bmc,
  output    [1:0] path_1_bmc);

  logic tm00, tm10, tm01, tm11;
```

```
  assign tmp00 = rx_pair[0];
  assign tmp01 = rx_pair[1];

  assign tmp10 = !tmp00;
  assign tmp11  = !tmp01;

  assign path_0_bmc[1] = tmp00 & tmp01;
  assign path_0_bmc[0] = tmp00 ^ tmp01;

  assign path_1_bmc[1] = tmp10 & tmp11;
  assign path_1_bmc[0] = tmp10 ^ tmp11;

endmodule
```

Bmc001.sv

```
module bmc001
(
  input    [1:0] rx_pair,
  output   [1:0] path_0_bmc,
  output   [1:0] path_1_bmc);

  logic tm00, tm10, tm01, tm11;

  assign tmp00 = rx_pair[0];
  assign tmp01 = !rx_pair[1];

  assign tmp10 = !tmp00;
  assign tmp11  = !tmp01;

  assign path_0_bmc[1] = tmp00 & tmp01;
  assign path_0_bmc[0] = tmp00 ^ tmp01;

  assign path_1_bmc[1] = tmp10 & tmp11;
  assign path_1_bmc[0] = tmp10 ^ tmp11;

endmodule
```

Bmc010.sv

```
module bmc010
(
  input    [1:0] rx_pair,
  output   [1:0] path_0_bmc,
  output   [1:0] path_1_bmc);

  logic tm00, tm10, tm01, tm11;

  assign tmp00 = rx_pair[0];
  assign tmp01 = !rx_pair[1];

  assign tmp10 = !tmp00;
  assign tmp11  = !tmp01;

  assign path_0_bmc[1] = tmp00 & tmp01;
  assign path_0_bmc[0] = tmp00 ^ tmp01;

  assign path_1_bmc[1] = tmp10 & tmp11;
  assign path_1_bmc[0] = tmp10 ^ tmp11;

endmodule
```

Bmc011.sv

```
module bmc011
(
  input    [1:0] rx_pair,
  output   [1:0] path_0_bmc,
  output   [1:0] path_1_bmc);

  logic tm00, tm10, tm01, tm11;

  assign tmp00 = rx_pair[0];
  assign tmp01 = rx_pair[1];

  assign tmp10 = !tmp00;
  assign tmp11  = !tmp01;

  assign path_0_bmc[1] = tmp00 & tmp01;
```

```
  assign path_0_bmc[0] = tmp00 ^ tmp01;

  assign path_1_bmc[1] = tmp10 & tmp11;
  assign path_1_bmc[0] = tmp10 ^ tmp11;
endmodule
```

Bmc100.sv

```
module bmc100
(
  input    [1:0] rx_pair,
  output   [1:0] path_0_bmc,
  output   [1:0] path_1_bmc);

  logic tm00, tm10, tm01, tm11;

  assign tmp00 = rx_pair[0];
  assign tmp01 = rx_pair[1];

  assign tmp10 = !tmp00;
  assign tmp11  = !tmp01;

  assign path_0_bmc[1] = tmp00 & tmp01;
  assign path_0_bmc[0] = tmp00 ^ tmp01;

  assign path_1_bmc[1] = tmp10 & tmp11;
  assign path_1_bmc[0] = tmp10 ^ tmp11;
endmodule
```

Bmc101.sv

```
module bmc101
 (
  input    [1:0] rx_pair,
  output   [1:0] path_0_bmc,
  output   [1:0] path_1_bmc);

  logic tm00, tm10, tm01, tm11;
```

```
  assign tmp00 = rx_pair[0];
  assign tmp01 = !rx_pair[1];

  assign tmp10 = !tmp00;
  assign tmp11  = !tmp01;

  assign path_0_bmc[1] = tmp00 & tmp01;
  assign path_0_bmc[0] = tmp00 ^ tmp01;

  assign path_1_bmc[1] = tmp10 & tmp11;
  assign path_1_bmc[0] = tmp10 ^ tmp11;

endmodule
```

Bmc110.sv

```
module bmc110
(
  input    [1:0] rx_pair,
  output   [1:0] path_0_bmc,
  output   [1:0] path_1_bmc);

  logic tm00, tm10, tm01, tm11;

  assign tmp00 = rx_pair[0];
  assign tmp01 = !rx_pair[1];

  assign tmp10 = !tmp00;
  assign tmp11  = !tmp01;

  assign path_0_bmc[1] = tmp00 & tmp01;
  assign path_0_bmc[0] = tmp00 ^ tmp01;

  assign path_1_bmc[1] = tmp10 & tmp11;
  assign path_1_bmc[0] = tmp10 ^ tmp11;

endmodule
```

Bmc111.sv

```
module bmc111 (
  input    [1:0] rx_pair,
  output   [1:0] path_0_bmc,
  output   [1:0] path_1_bmc);

  logic tm00, tm10, tm01, tm11;

  assign tmp00 = rx_pair[0];
  assign tmp01 = rx_pair[1];

  assign tmp10 = !tmp00;
  assign tmp11  = !tmp01;

  assign path_0_bmc[1] = tmp00 & tmp01;
  assign path_0_bmc[0] = tmp00 ^ tmp01;

  assign path_1_bmc[1] = tmp10 & tmp11;
  assign path_1_bmc[0] = tmp10 ^ tmp11;

endmodule
```

Design (Given)

viterbi_tx_rx.sv

```
module viterbi_tx_rx(
  input    clk,
  input    rst,
  input    encoder_i,
  input    enable_encoder_i,
  output   decoder_o);

  wire  [1:0] encoder_o;

  logic   [3:0] error_counter;
  logic   [1:0] encoder_o_reg;
```

```
  logic       enable_decoder_in;
  wire        valid_encoder_o;

  always @ (posedge clk, negedge rst)
     if(!rst) begin
        error_counter  <= 4'd0;
        encoder_o_reg  <= 2'b00;
        enable_decoder_in <= 1'b0;
     end
     else     begin
        enable_decoder_in <= valid_encoder_o;
        encoder_o_reg  <= 2'b00;
        error_counter  <= error_counter + 4'd1;
        if(error_counter==4'b1111)
           encoder_o_reg  <= {~encoder_o[1],encoder_o[0]};  // inject one bad bit
out of every 32
        else
           encoder_o_reg  <= {encoder_o[1],encoder_o[0]};
     end

// insert your convolutional encoder here
// change port names and module name as necessary/desired
  encoder encoder1       (
     .clk,
     .rst,
     .enable_i(enable_encoder_i),
     .d_in(encoder_i),
     .valid_o(valid_encoder_o),
     .d_out(encoder_o)   );

// insert your term project code here
  decoder decoder1       (
     .clk,
     .rst,
     .enable(enable_decoder_in),
     .d_in(encoder_o_reg),
     .d_out(decoder_o)   );

endmodule
```

mem_1x1024.sv

```
module mem_disp      (
  input          clk,
  input          wr,
  input [9:0]    addr,
  input          d_i,
  output logic     d_o);

  logic            mem   [1024];

  always @ (posedge clk) begin
     if(wr)
        mem[addr]   <= d_i;
     d_o  <=  mem[addr];
 end
endmodule
```

mem_8x1024.sv

```
module mem              (
  input                  clk,
  input                  wr,
  input         [9:0]    addr,
  input         [7:0]    d_i,
  output logic  [7:0]    d_o);

  logic         [7:0]    mem   [1024];

  always @ (posedge clk) begin
     if(wr)
        mem[addr]   <= d_i;
     d_o  <=  mem[addr];
  end
endmodule
```

Testbenches

viterbi_tx_rx_tb.sv

```
// test bench
module viterbi_tx_rx_tb();
  bit clk;
  bit rst;
  bit encoder_i;        // original data
  bit enc_i_hist[2048];    // history thereof
  bit enable_encoder_i;
  wire decoder_o;          // decoded data
  bit dec_o_hist[2048];    // history thereof
  bit disp;
  int good, bad;           // scoreboard

// this module contains conv. encode and Vit decode
  viterbi_tx_rx vtr(
     .clk,
     .rst,
     .encoder_i,         // original data
     .enable_encoder_i,
     .decoder_o     );      // decoded data

  always begin
     #50   clk    = 1;
     #50  clk     = 0;
  end
  int i, j, k, l;

  always @(posedge clk) begin
   enc_i_hist[i] <= encoder_i;
   i <= i+1;
   l <= l+1;
  end
  initial begin
   #410400;
   forever @(posedge clk) begin
     dec_o_hist[k] <= decoder_o;
     k<=k+1;
    end
  end
```

```
initial begin
    clk      =  1'b1;
    #1000
    rst       =  1'b1;
    enable_encoder_i  =  1'b1;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b1;
```

```
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b1;

    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
```

```
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b1;
    #100
    encoder_i=  1'b1;
    #100
```

```
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;
    #1000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;


    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;


    #10000
    encoder_i=  1'b0;
    #100
```

```
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;


    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
```

```
    encoder_i=  1'b1;


    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
```

```
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;


    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
```

```
    encoder_i=  1'b1;
    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #10000
    encoder_i=  1'b0;
    #100
    encoder_i=  1'b1;

    #1000000
    for(j=0; j<256; j=j+1)      // checker & scoreboard
      if(enc_i_hist[j]==dec_o_hist[j]) begin
        $displayb("yaa! in = %b, out = %b",enc_i_hist[j],dec_o_hist[j]);
        good++;
    end
    else begin
        $displayb("boo! in = %b, out = %b",enc_i_hist[j],dec_o_hist[j]);
        bad++;
    end
```

```
    $display("good = %d, bad = %d",good,bad);
    disp = 1;

     $stop;
  end

endmodule
```

encoder_tb.sv

```
module encoder_tb;

  bit         clk;
  bit         rst;
  bit         enable_i=1;
  bit         d_in;
  wire  [1:0] d_out;
  wire        valid_o;

  encoder DUT    (
     .clk,
     .rst,
    .enable_i,
     .d_in,
     .d_out,
    .valid_o   );

  always begin
     #10   clk  =  1'b1;
     #10   clk  = 1'b0;
  end

  initial  begin

     #110
     rst   =  1'b1;
     d_in  =  1'b0;
```

```
    #20
    d_in  =  1'b1;

    #20
    d_in  =  1'b0;

    #20
    d_in  =  1'b0;

    #20
    d_in  =  1'b0;

    #20
    d_in  =  1'b1;

    #20
    d_in  =  1'b0;

    #20
    d_in  =  1'b0;

    #20
    d_in  =  1'b1;

    #20
    d_in  =  1'b1;

    #20
    d_in  =  1'b0;

    #20
    d_in  =  1'b0;

    $stop;

  end
endmodule
```