PROJECT REPORT

KOMATREDDY VIKRAM KUMAR(MT2021516)

PRAFFUL CHOUDHARY(MT2021523)

FPGA ACCELERATED COMPRESSION AND DECOMPRESSION

BLOCK DIAGRAM:

1)COMPRESSION

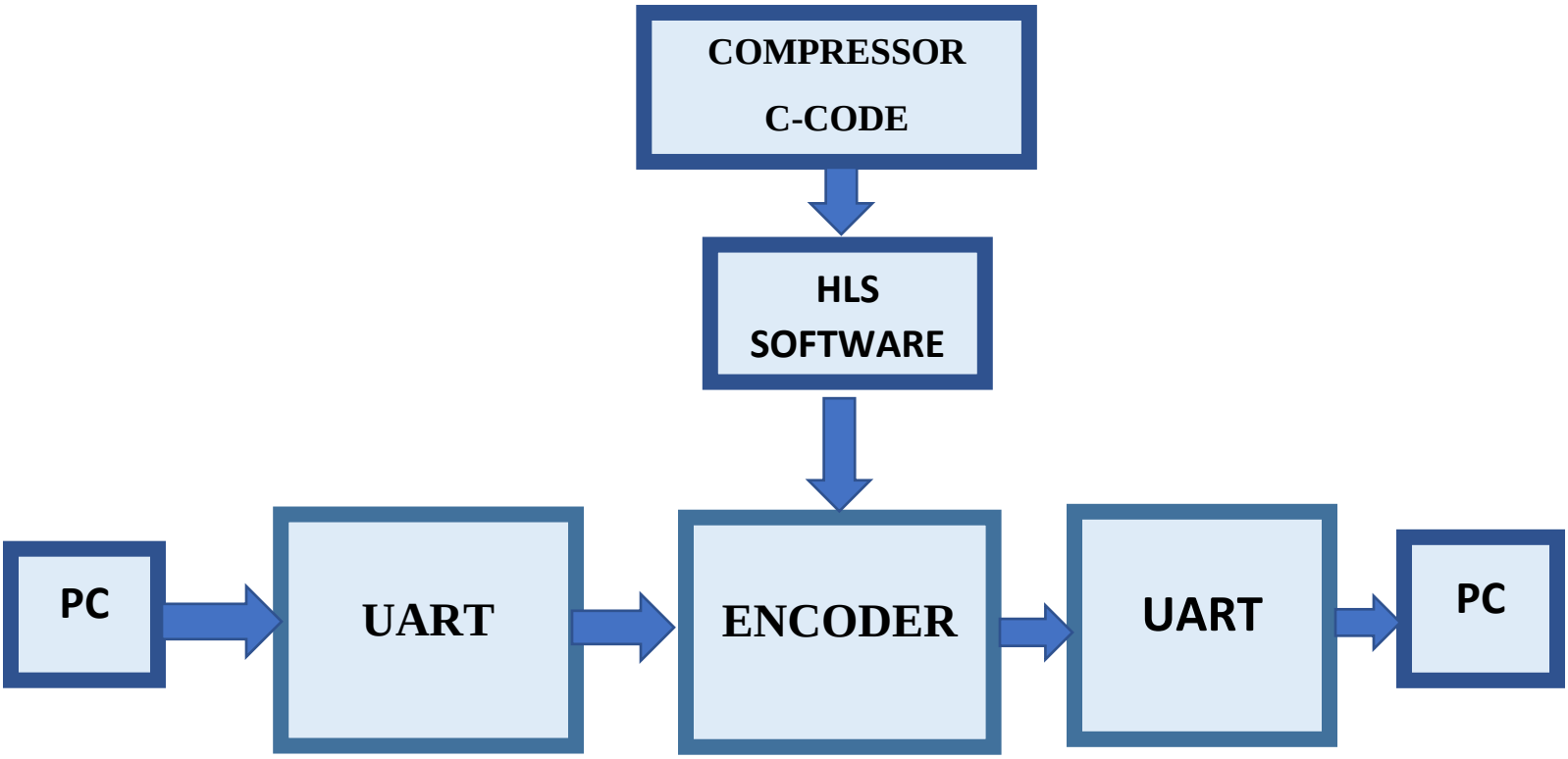


Figure1: Block Diagram for Compression Step

Steps Followed:

1. The First step of our Project is to write the C code for the compression of the text file. The C code is basically converting the text file characters into sequence of binary numbers.
2. Then we use the HLS Tool which help us to convert this C Code into RTL.
3. We have created the Compressor synthesized RTL code.
4. We are going to use the UART communication between Basys 3 Board and computer terminal.
5. Input for Compressor module is a text file which contains number of characters.
6. These characters in the text file are sent individually to the FPGA board through UART.
7. Once all the characters in the text file reaches FPGA, the compressor/encoder module is activated by giving start=1.
8. When encoding process is done, the module make ap_done signal as one. From then all the encoded bits must be sent back to computer from FPGA again through UART.

2)DECOMPRESSION

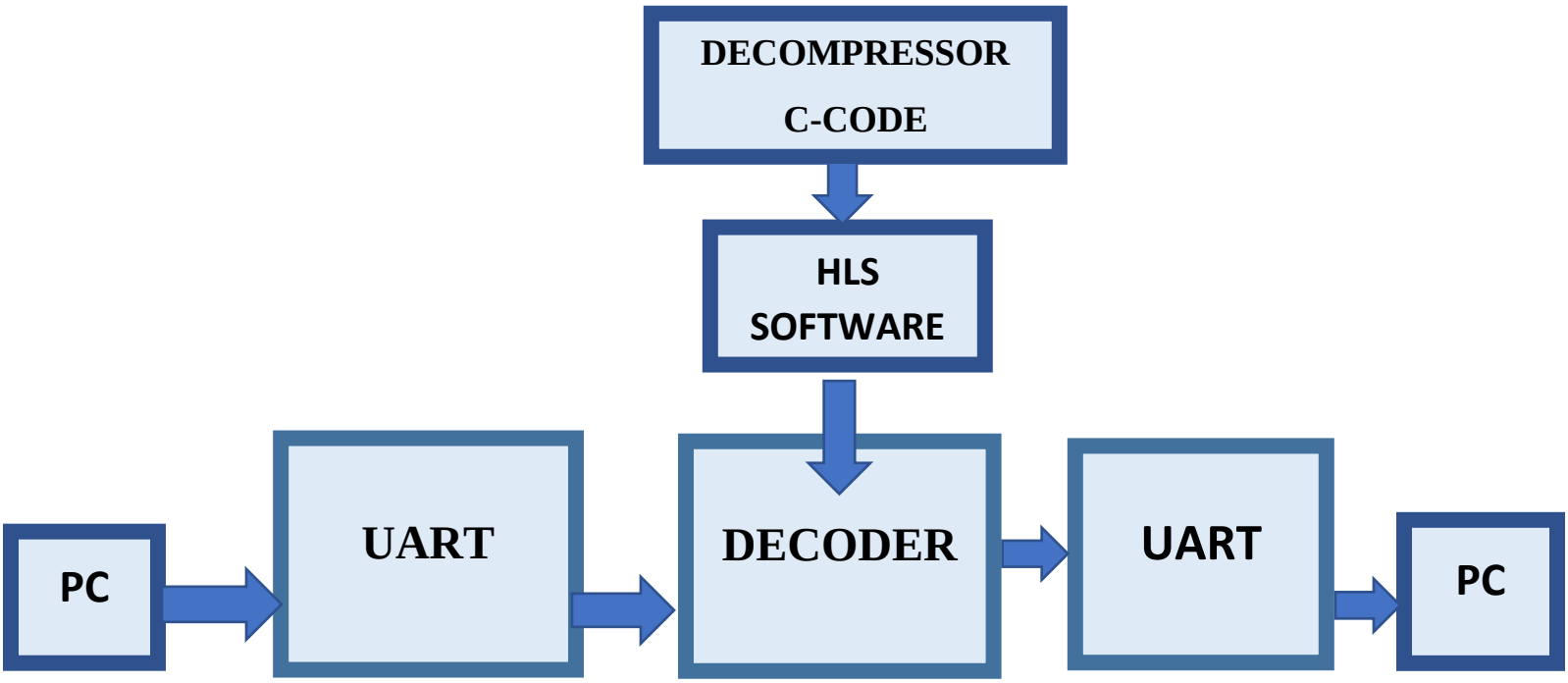


Figure2: Block Diagram for Decompression step

Steps Followed:

1. The First step of our Project is to write the C code for the decompression of the text file. The C code is basically converting the encoded data in the previous stage into actual input data.
2. Then we use the HLS Tool which help us to convert this C Code into RTL.
3. We have created the decompressor synthesized RTL code.
4. We are going to use the UART communication between Basys 3 Board and computer terminal.
5. Input for decompressor module is a text file which contains encoded integer values of the characters provided as input in the first stage.
6. These integers in the text file are sent individually to the FPGA board through UART.
7. Once all the data in the text file reaches FPGA, the decompressor/decoder module is activated by giving start=1.
8. When decoding process is done, the module make ap_done signal as one. From then all the encoded bits must be sent back to computer from FPGA again through UART.

SIMULATION RESULTS:

Steps:

1. We have extracted all the RTL codes generated in HLS and created another Vivado project.

2. Wrote a testbench providing inputs to the RTL codes extracted and checked the functionality of the design.

3. We observed that the desired encoding and decoding are happening. And the simulation waveforms are attached below.

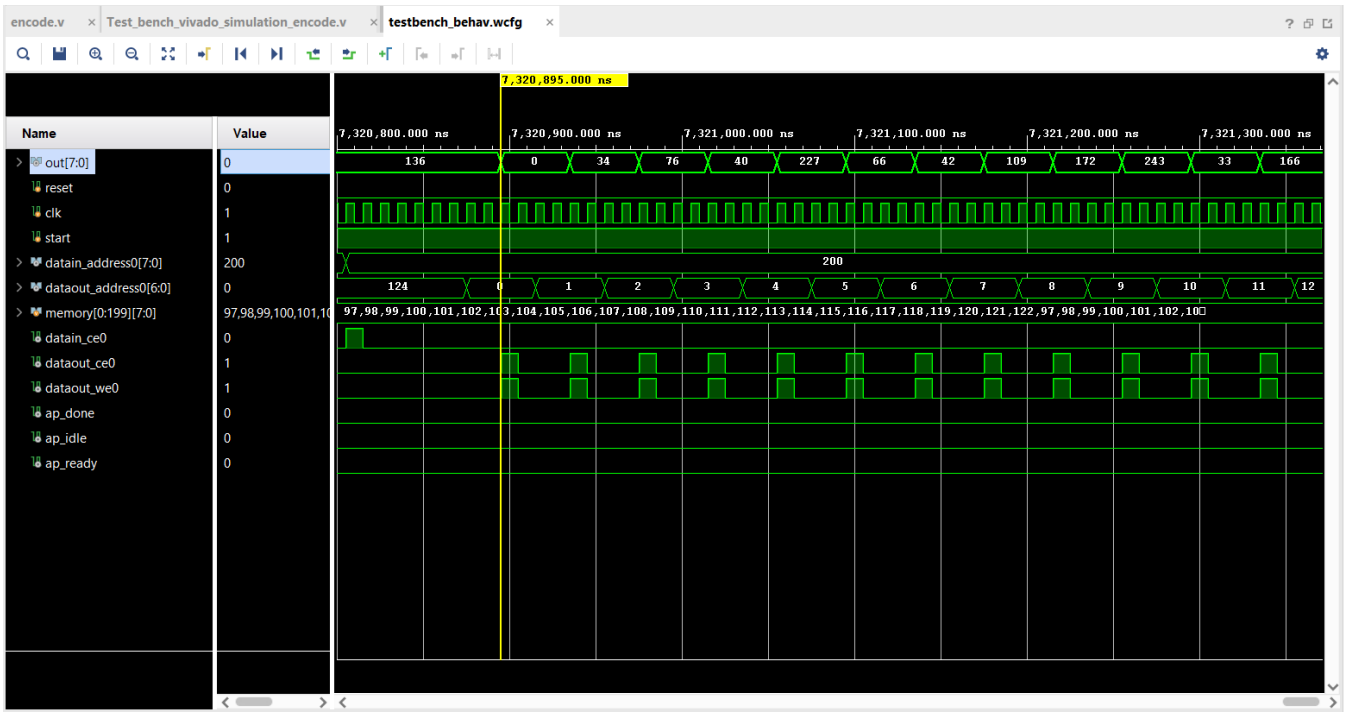


Figure3: Encoder Waveform after Simulation Step

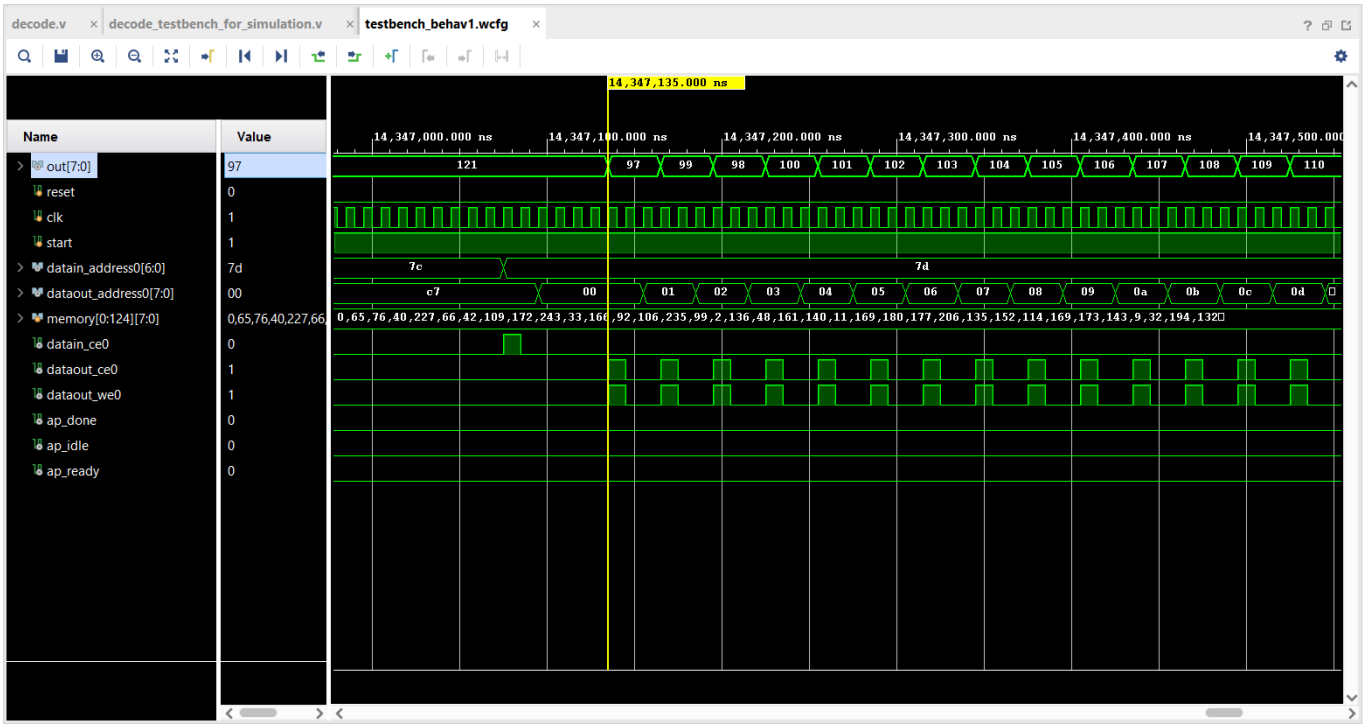


Figure4: Decoder Waveform after Simulation Step

IMPLEMENTATION:

Let us look at the schematic structures obtained after the implementation stage:

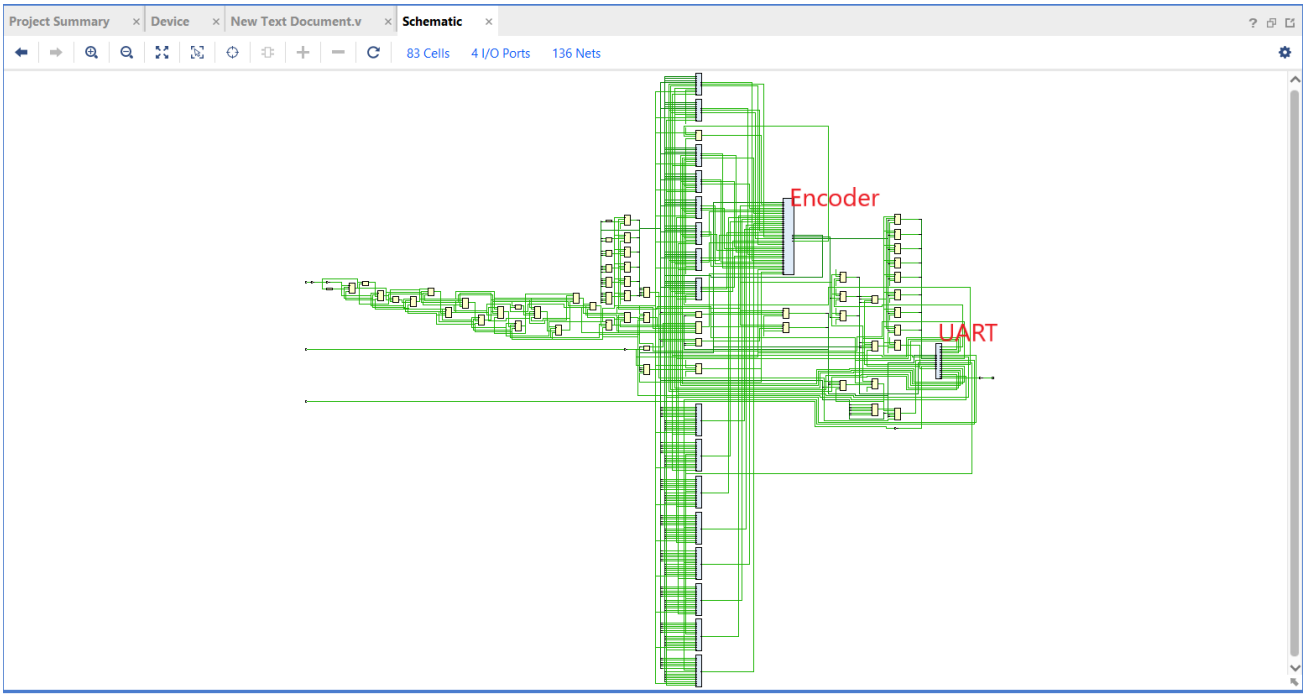


Figure5: Schematic of encoder after implementation

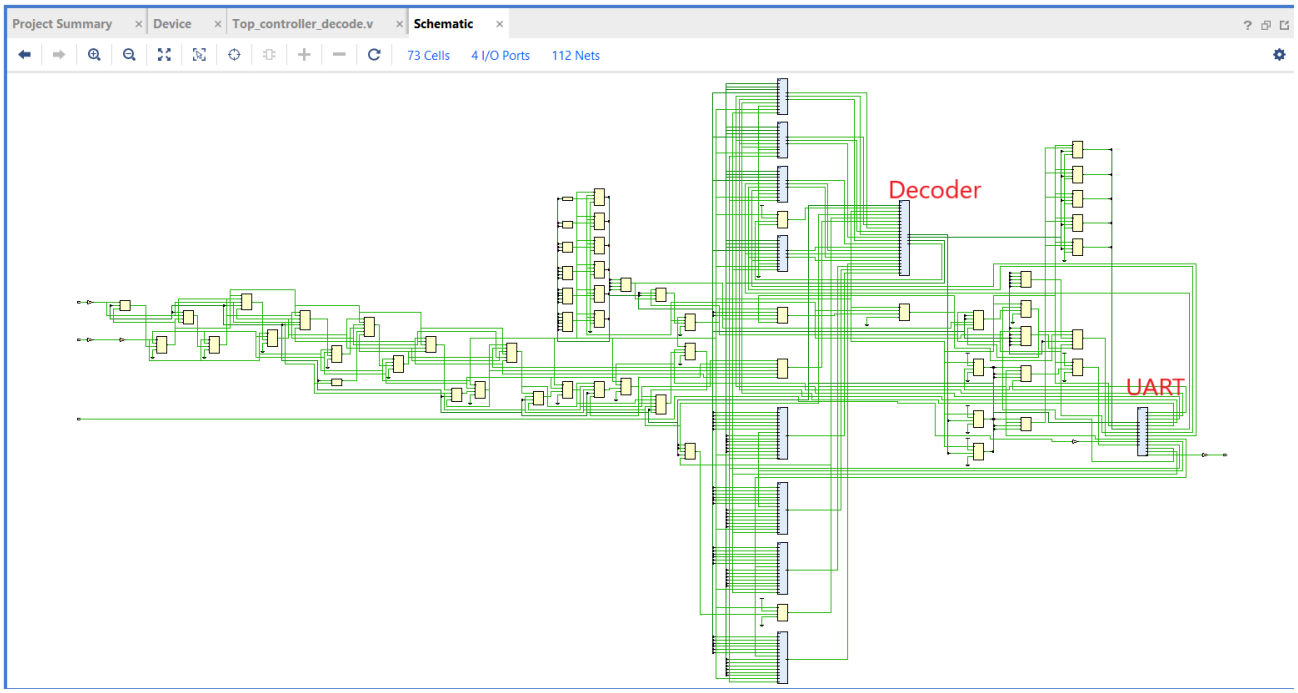
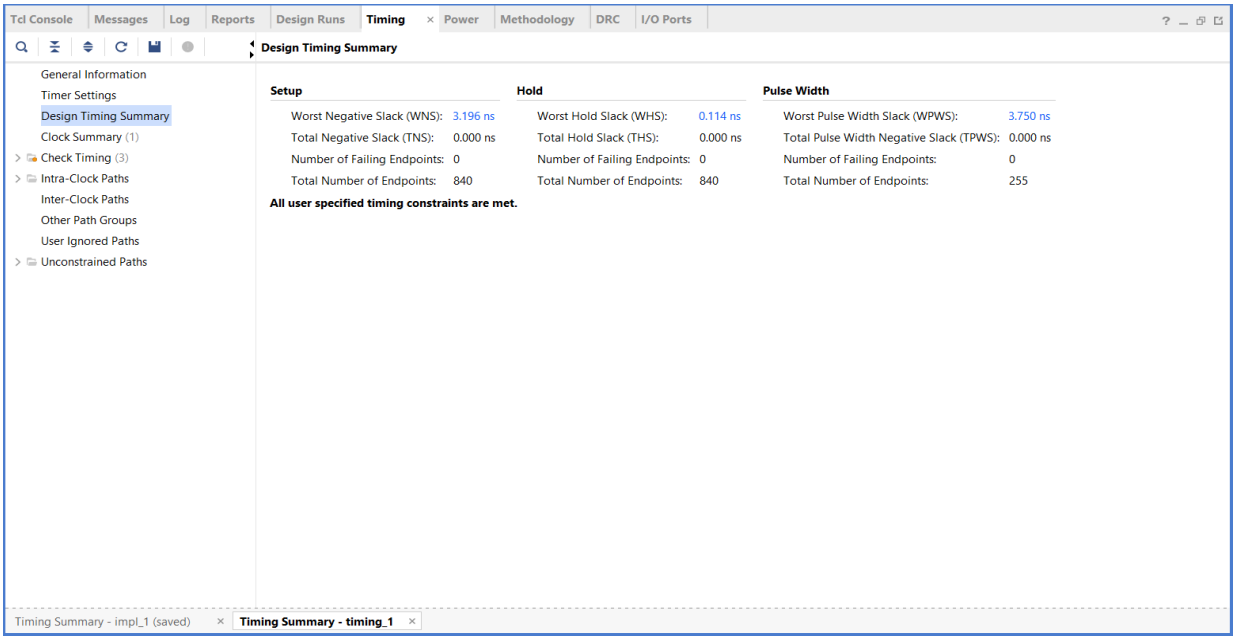


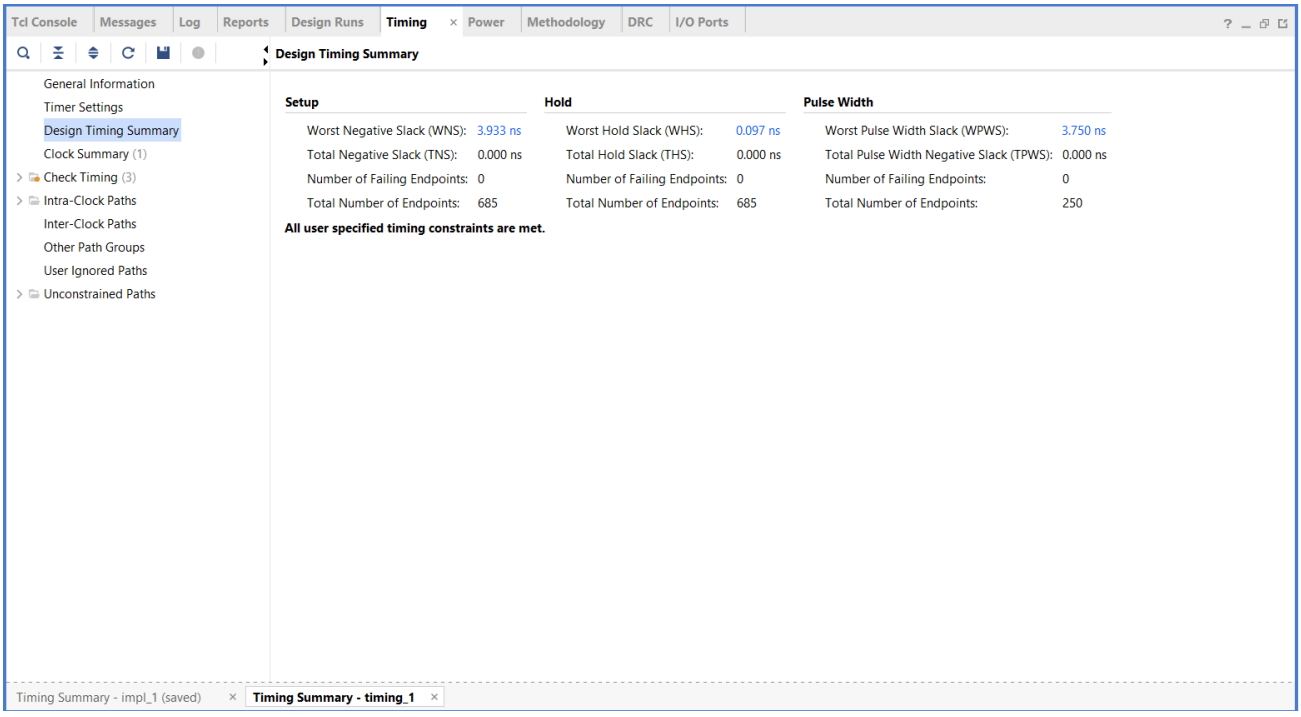
Figure6: Schematic of decoder after implementation.

Timing Reports of each Design are also attached here:

1. Encoder:

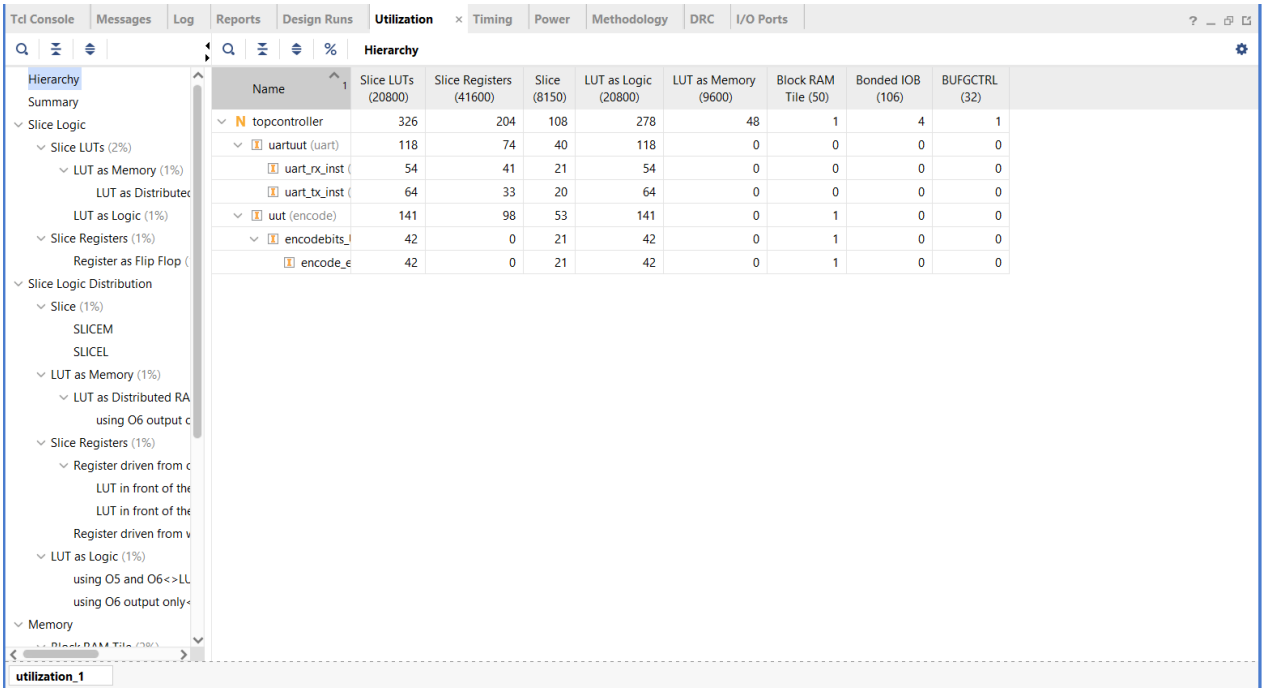


2. Decoder



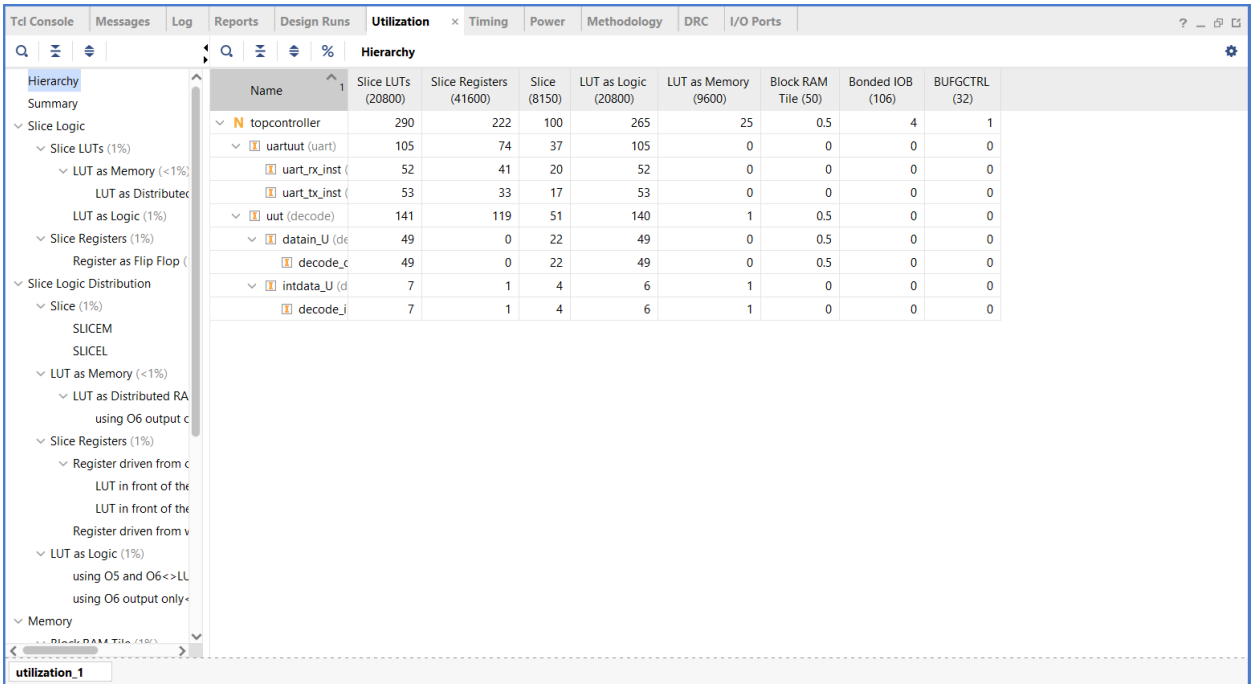
Utilization Reports of each design:

1. Encoder



Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT as Memory (9600)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
topcontroller	326	204	108	278	48	1	4	1
uartuut (uart)	118	74	40	118	0	0	0	0
uart_rx_inst (	54	41	21	54	0	0	0	0
uart_tx_inst (	64	33	20	64	0	0	0	0
uut (encode)	141	98	53	141	0	1	0	0
encodebits_I	42	0	21	42	0	1	0	0
encode_e	42	0	21	42	0	1	0	0

2. Decoder:



Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT as Memory (9600)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
topcontroller	290	222	100	265	25	0.5	4	1
uartuut (uart)	105	74	37	105	0	0	0	0
uart_rx_inst (	52	41	20	52	0	0	0	0
uart_tx_inst (	53	33	17	53	0	0	0	0
uut (decode)	141	119	51	140	1	0.5	0	0
datain_U (de	49	0	22	49	0	0.5	0	0
decode_c	49	0	22	49	0	0.5	0	0
intdata_U (d	7	1	4	6	1	0	0	0
decode_i	7	1	4	6	1	0	0	0

Steps followed in implementation in Vivado:

1. We have simulated a python code in which the input characters are read from a file and provided to the FPGA through UART.

2. Once the input characters reach the FPGA, the encoding/decoding process gets started.

3. After the process is done, all the data will be sent to the pc from FPGA through UART.

Results:

We are successful in sending the data from the input file in pc to the FPGA through UART. But we are unable to get the output from UART. Address of the output was not getting incremented and we are getting only one output values displayed in the output file.

```
data:%d 118
184
data:%d 119
144
data:%d 120
74
data:%d 121
27
data:%d 122
235
data:%d 123
124
data:%d 124
136
data received from FPGA (data %d): 0
114
data received from FPGA (data %d): 1
114
data received from FPGA (data %d): 2
114
data received from FPGA (data %d): 3
114
data received from FPGA (data %d): 4
114
data received from FPGA (data %d): 5
114
data received from FPGA (data %d): 6
114
```

----------THANKYOU----------