Digital circuit using HLS for a packet shifter

K.Vijaya Lakshmi

Duration : 06-06-2023 to 12-06-2023

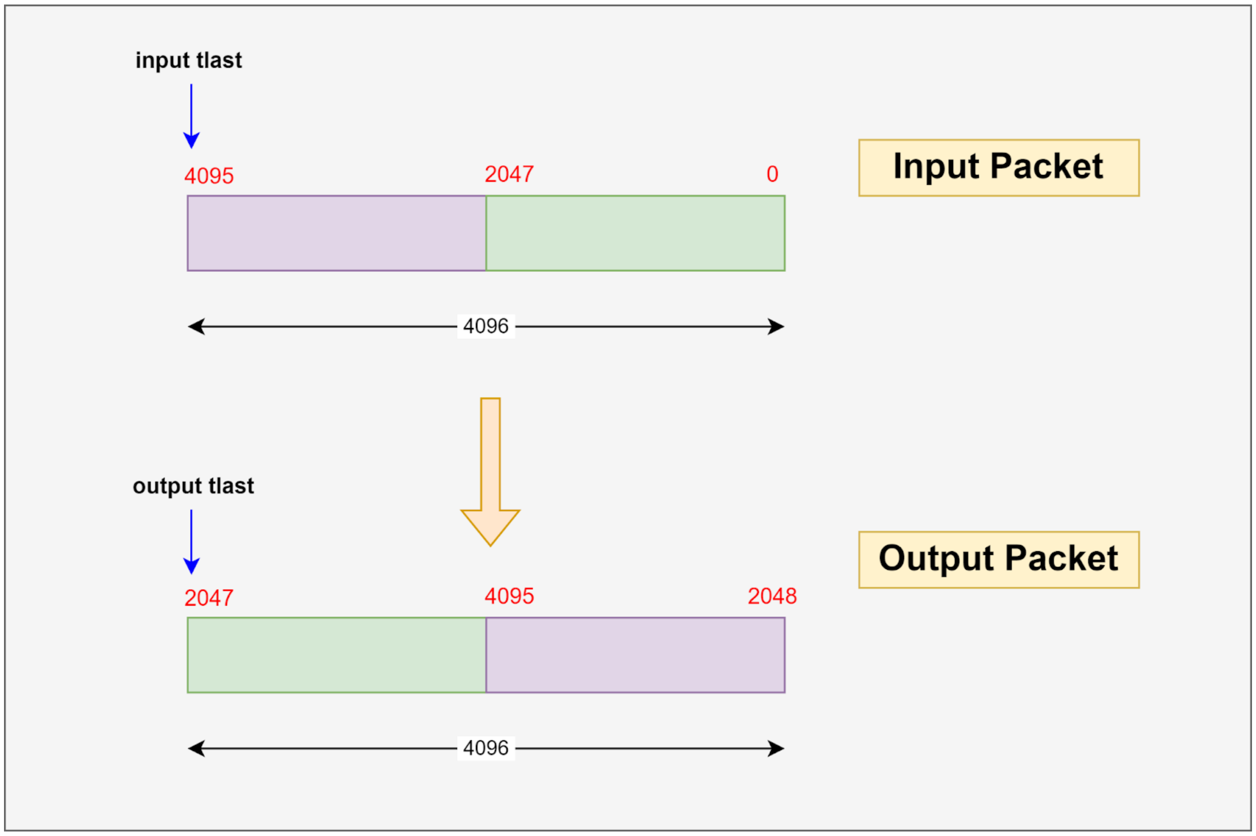
Table Of Content

Problem Statement:

Design a digital circuit using HLS for a packet shifter.

🎯 Requirements

- The circuit should produce an output packet from an input packet as illustrated in the below figure.



- The primary requirement is the output packet generation should be continuous without any idle clock cycles inbetween consecutive output packets, as the input packets stream will be continuous.
- But, Initial one time latency is acceptable.
- The input and output buses should use AXIS interface.
- Minimum Fmax required is 400MHz.

💼 Considerations

- Each packet length is 4096 bytes, and it will be received at a rate of 1 byte per clock cycle.

- Infinite stream of such packets will be fed to the circuit without any IDLE clock cycles in between consecutive input packets.
- The LSB byte of a packet will be received first.
- A packet end indicated by the TLAST signal in both input and output.

🚚 Deliverables

- HDL code implementation of the circuit.
- RTL simulation results.
- Images of the waveforms.
- Resource utilization report analysis.
- Timing report which contains the latency, Interval, throughput and the Fmax of the circuit.
- Documentation detailing the design choices, implementation steps, and performance evaluation

Solution Approach

- Based on the given input and output format mentioned in the assignment, I written half of the input stream to a buffer(array) and the remaining half of the input packet was directly given to the output stream .
- So if input packet contains 8 bytes of data , for example { 1,2,3,4,5,6,7,8} then output should be {5,6,7,8 , 1,2,3,4 } , so we first store 4 elements to an array and remaining is given to output stream and after this we read the input stored in the buffer(array) .
- Initial one time latency is because we are writing half of the input data to an array , so it will happen and we have to make sure input and output is continuous .

Code Implementation

- I implemented the code using switch case statements , that can be understood by the following FSM (Finite State Machine) .

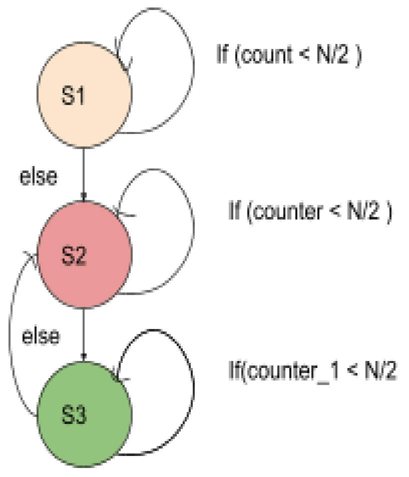


Figure 1 : Steps in code implementation

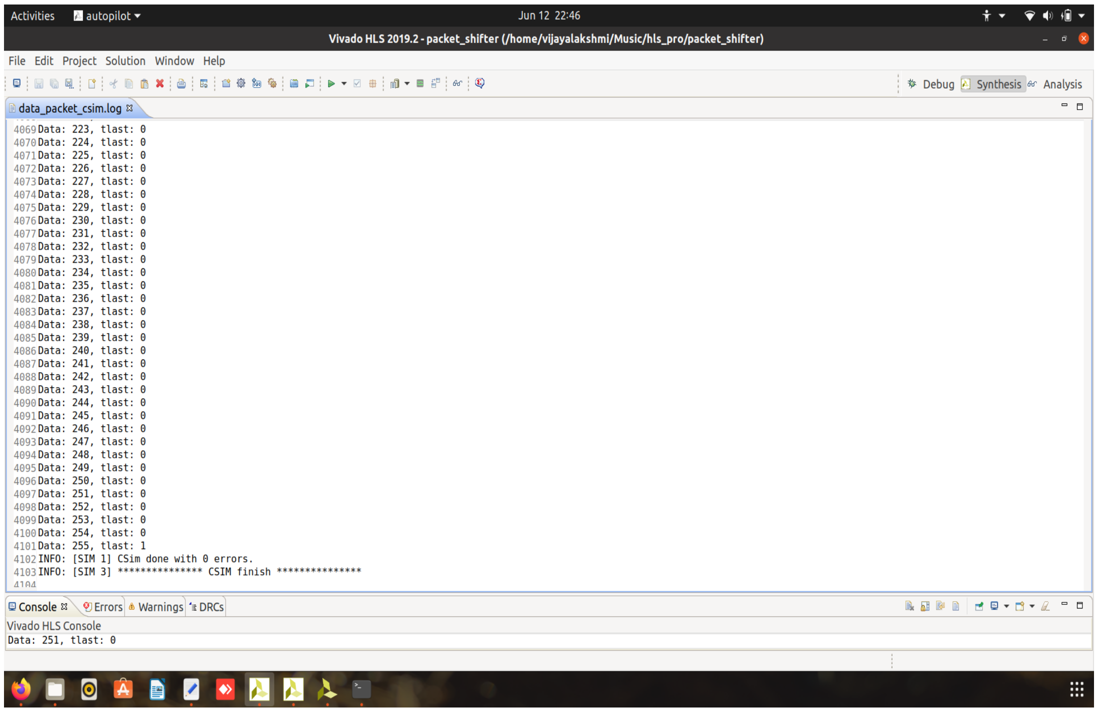
S1 State : In the S1 state I store half of the input stream(each packet containing N bytes of data) to a buffer of size N/2 . It works based on the counter ,if the condition fails it goes to S2 state .

S2 State : In the S2 state I'm giving the other half of the array directly to the output stream . If bypassing completes in this state it goes to S3 state .

S3 State : In the S3 state , I'm writing the stored elements in the S1 state to the output and storing another half of the data from another packet to the array .If this completes it goes to S2 state (process continues between S2 and S3 state).

HLS Reports Analysis

Simulation Result

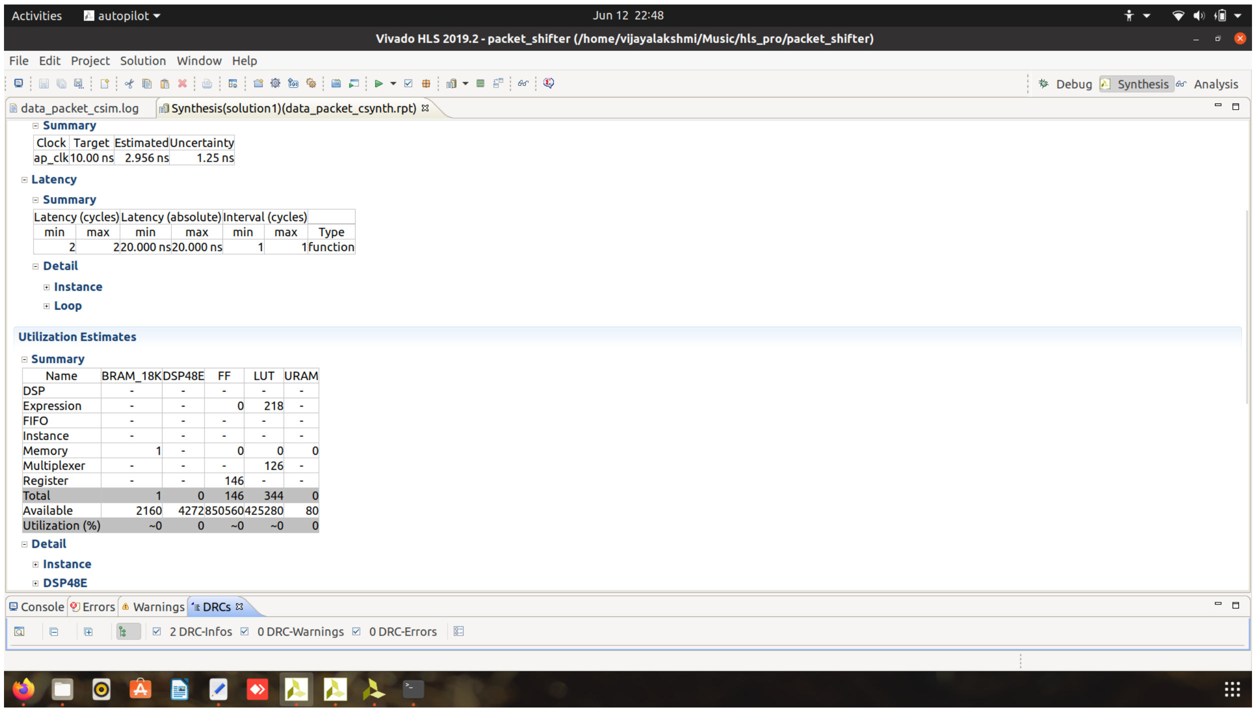

Activities
autopilot
Jun 12 22:46
Vivado HLS 2019.2 - packet_shifter (/home/vijayalakshmi/Music/hls_pro/packet_shifter)
File Edit Project Solution Window Help
Debug
Synthesis
Analysis
data_packet_csim.log
Console
Errors
Warnings
DRCs
Vivado HLS Console

```
Data: 223, tlast: 0
Data: 224, tlast: 0
Data: 225, tlast: 0
Data: 226, tlast: 0
Data: 227, tlast: 0
Data: 228, tlast: 0
Data: 229, tlast: 0
Data: 230, tlast: 0
Data: 231, tlast: 0
Data: 232, tlast: 0
Data: 233, tlast: 0
Data: 234, tlast: 0
Data: 235, tlast: 0
Data: 236, tlast: 0
Data: 237, tlast: 0
Data: 238, tlast: 0
Data: 239, tlast: 0
Data: 240, tlast: 0
Data: 241, tlast: 0
Data: 242, tlast: 0
Data: 243, tlast: 0
Data: 244, tlast: 0
Data: 245, tlast: 0
Data: 246, tlast: 0
Data: 247, tlast: 0
Data: 248, tlast: 0
Data: 249, tlast: 0
Data: 250, tlast: 0
Data: 251, tlast: 0
Data: 252, tlast: 0
Data: 253, tlast: 0
Data: 254, tlast: 0
Data: 255, tlast: 1
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

```
Data: 251, tlast: 0
```

Synthesis Report

- The design utilized 344 LUT's and 146 FF's and 1 BRAM .


Activities
autopilot
Jun 12 22:48
Vivado HLS 2019.2 - packet_shifter (/home/vijayalakshmi/Music/hls_pro/packet_shifter)
File Edit Project Solution Window Help
Debug
Synthesis
Analysis
data_packet_csim.log
Synthesis(solution1)(data_packet_csynth.rpt)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	2.956 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
2	2	20.000 ns	20.000 ns	1	1	function

Detail

Instance

Loop

Utilization Estimates

Summary

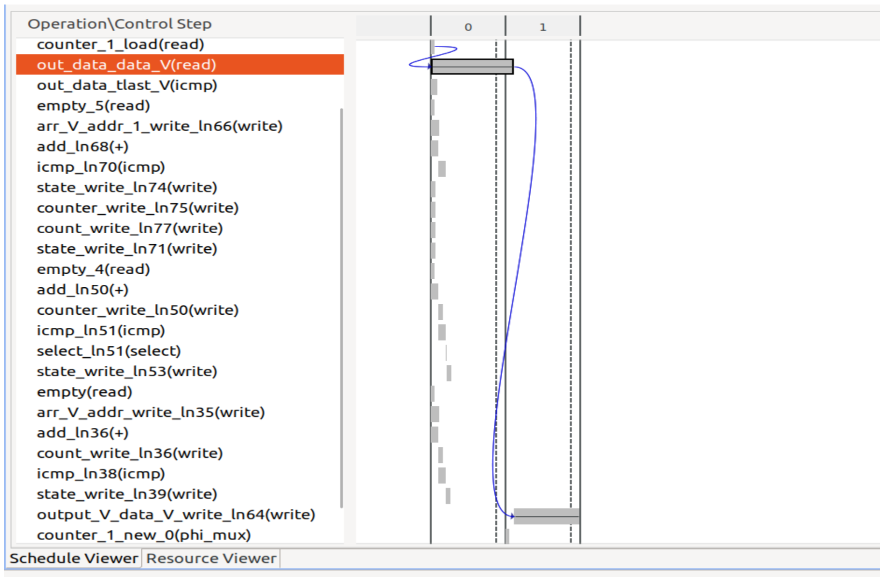
Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	218	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	1	-	0	0	0
Multiplexer	-	-	-	126	-
Register	-	-	146	-	-
Total	1	0	146	344	0
Available	2160	4272	850560	425280	80
Utilization (%)	~0	0	~0	~0	0

Detail

Instance

DSP48E

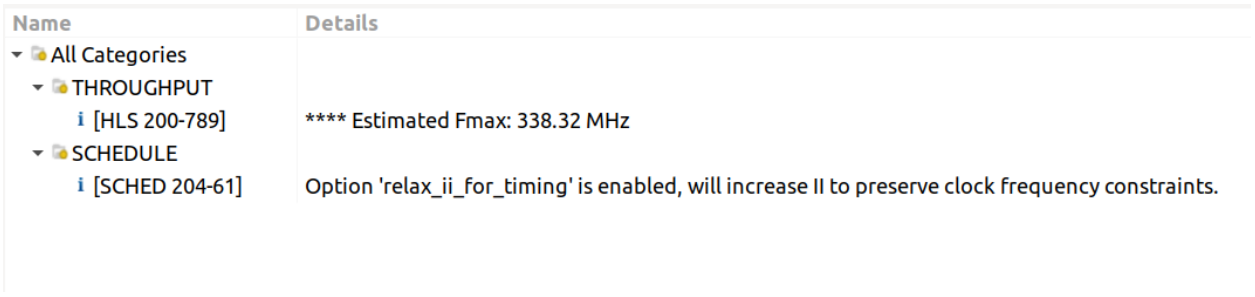
- In the synthesis report we can see that the design utilized one BRAM (Single Port BRAM) , where in one clock cycle it reads the data and in the next clock it writes the data , in the below diagram we can see that .Here estimated time is 2.956 ns so Fmax is 338.32 Mhz



Operation\Control Step
counter_1_load(read)
out_data_data_V(read)
out_data_tlast_V(icmp)
empty_5(read)
arr_V_addr_1_write_ln66(write)
add_ln68(+)
icmp_ln70(icmp)
state_write_ln74(write)
counter_write_ln75(write)
count_write_ln77(write)
state_write_ln71(write)
empty_4(read)
add_ln50(+)
counter_write_ln50(write)
icmp_ln51(icmp)
select_ln51(select)
state_write_ln53(write)
empty(read)
arr_V_addr_write_ln35(write)
add_ln36(+)
count_write_ln36(write)
icmp_ln38(icmp)
state_write_ln39(write)
output_V_data_V_write_ln64(write)
counter_1_new_0(phi_mux)

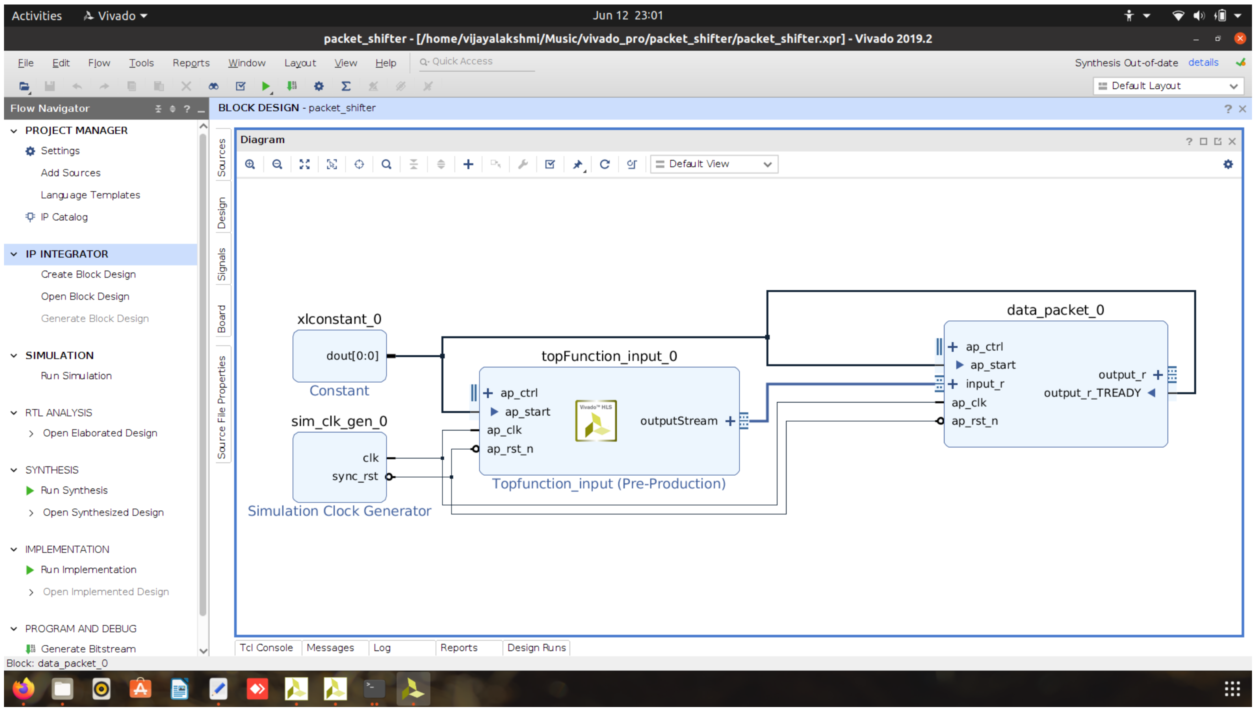
Fmax

- The estimated Fmax for the design is 338.32 MHz .

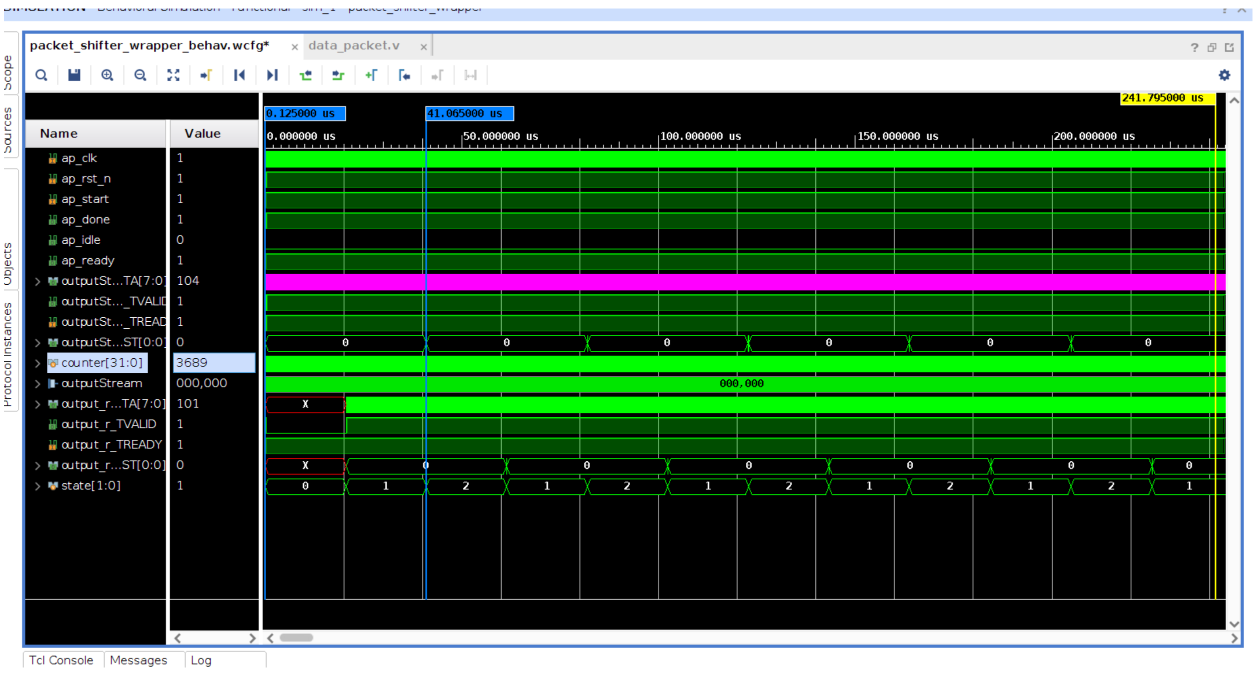


Name	Details
All Categories	
THROUGHPUT	
[HLS 200-789]	**** Estimated Fmax: 338.32 MHz
SCHEDULE	
[SCHED 204-61]	Option 'relax_ii_for_timing' is enabled, will increase II to preserve clock frequency constraints.

Vivado Block Diagram



RTL simulation



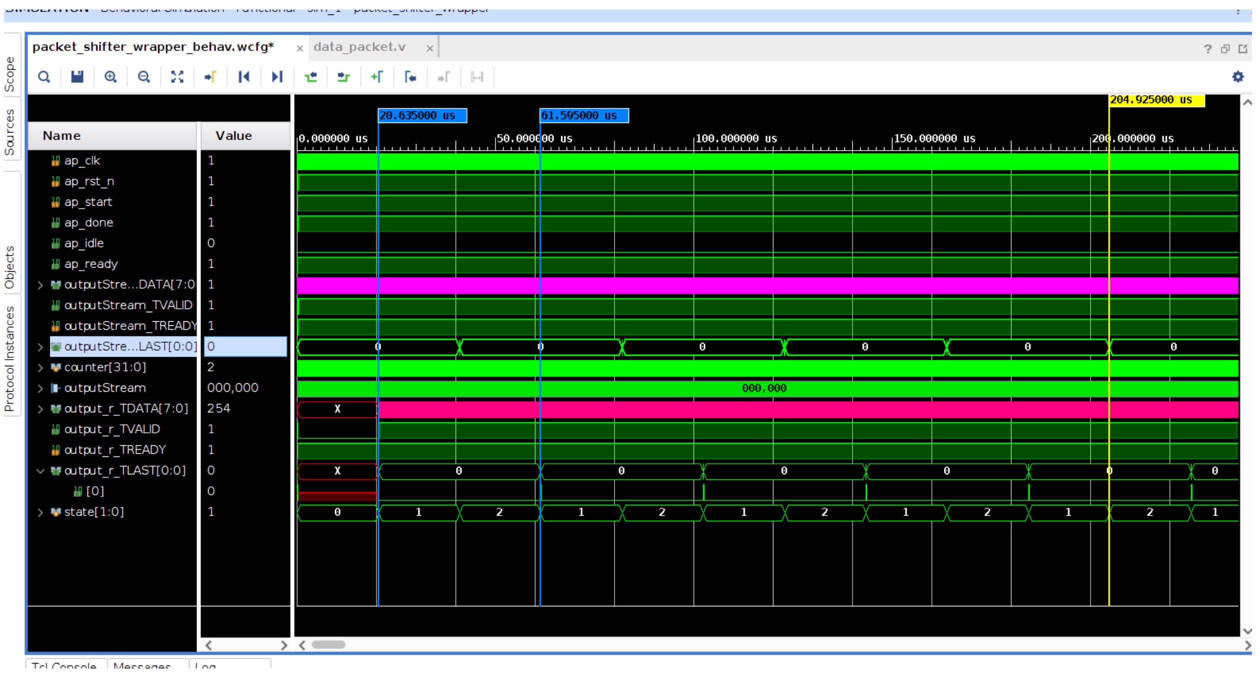
packet_shifter_wrapper_behav.wcfg* × data_packet.v ×

Name	Value
ap_clk	1
ap_rst_n	1
ap_start	1
ap_done	1
ap_idle	0
ap_ready	1
outputSt...TA[7:0]	104
outputSt..._TVALID	1
outputSt..._TREAD	1
outputSt...ST[0:0]	0
counter[31:0]	3689
outputStream	000,000
output_r...TA[7:0]	101
output_r_TVALID	1
output_r_TREADY	1
output_r...ST[0:0]	0
state[1:0]	1

0.125000 us
41.065000 us
241.795000 us
0.000000 us
50.000000 us
100.000000 us
150.000000 us
200.000000 us

Tcl Console Messages Log

(one packet of input data between the blue markers . → input color : Magenta)



packet_shifter_wrapper_behav.wcfg* × data_packet.v ×

Name	Value
ap_clk	1
ap_rst_n	1
ap_start	1
ap_done	1
ap_idle	0
ap_ready	1
outputStre...DATA[7:0]	1
outputStream_TVALID	1
outputStream_TREADY	1
outputStre...LAST[0:0]	0
counter[31:0]	2
outputStream	000,000
output_r_TDATA[7:0]	254
output_r_TVALID	1
output_r_TREADY	1
output_r_TLAST[0:0]	0
[0]	0
state[1:0]	1

20.635000 us
61.595000 us
204.925000 us
0.000000 us
50.000000 us
100.000000 us
150.000000 us
200.000000 us

Tcl Console Messages Log

(one packet of output data between the blue markers . output color –light purple)

Conclusion

- In this assignment I generated the output packet continually without any idle clock cycles in between consecutive output packets as input is also continuous .
- I got the Fmax as 338.32 Mhz .