COL215 Hardware Assignment 1 Report

By Disha Shiraskar & Srushti Junghare

In this assignment, we were asked to

• Design a combinational circuit that takes a single 4-bit hexadecimal or decimal digit input from the switches and produces a 7-bit output for the seven-segment display of Basys3 FPGA board.

• Extend the design to create a timing circuit that will drive all 4-displays for displaying 4-digits together.

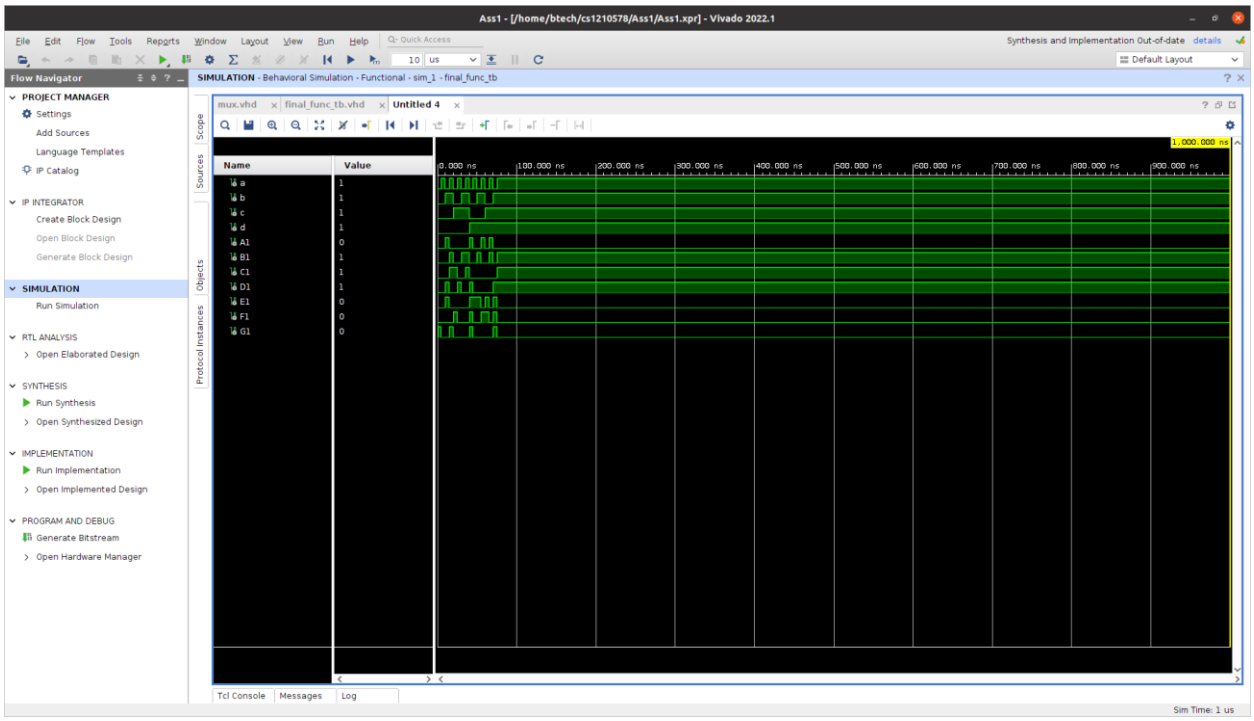
Our submission contains the following files:

- mux (Contains the main function)
  - final_clk (Contains Timing function)
    - Clock_divider (corresponds to selector s0 of mux)
    - new_clk (corresponds to selector s1 of mux)
  - seven_seg (Contains the seven segment decoder)

PART A : 7 Segment Decoder

The first part to be implemented was for displaying a 4-bit number on a one 7-segment display.
We first created a truth table containing 4 inputs and 7 outputs. The 4 input bits corresponded to the 4-bit binary form of decimal numbers 0 to 15. The 7 outputs corresponded to the seven segments of the display. We then deduced functions for each segment using k-map. We implemented all this in the behaviorial file, created a testbench and tested it on the Basys3 FGPA Board.

Following are the snapshots of the simulation of above implementation:



PART B : Driving All Four Seven Segment Displays

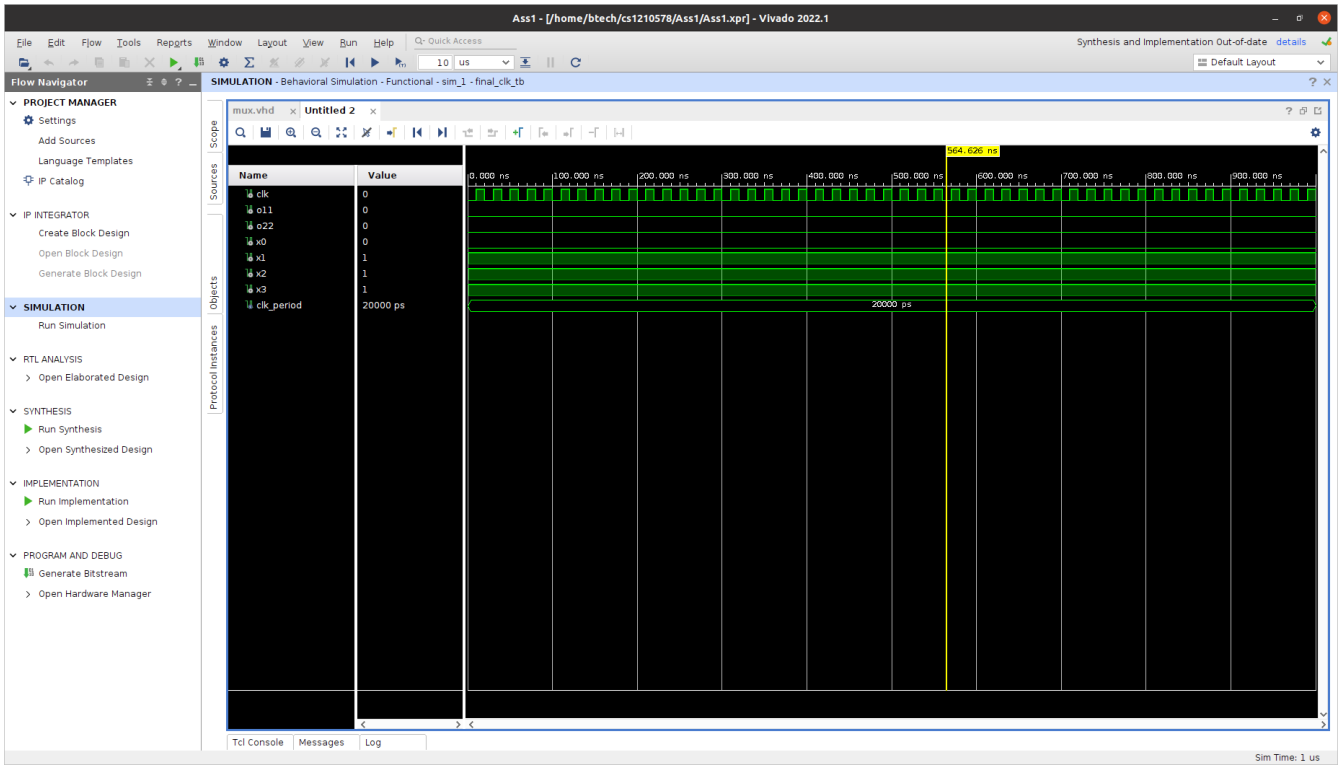
In the second part, we were asked to drive all four LED Displays by creating a 4:1 multiplexer module and timing circuit.
Since we are giving a 4-digit input and each digit has 4 bits in its binary form, the total no of inputs are 16.
We started by creating the timing circuit. The in-built clock of the basys3 board has a frequency of 100 MHz. This is equivalent to 10^{-8} seconds i.e 10 nanoseconds. We had to create a circuit that drives the anode signals and corresponding cathode patterns of each digit in a repeating, continuous

succession at an update rate that is faster than the human eye can detect. For this purpose, the refresh rate of the clock has to be reduced to 10 milliseconds i.e. to a frequency of 100Hz. If the frequency is decreased further a flicker will occur.

- **Timing Circuit: How did we slow the inbuilt clock?**

We made use of counter 'prescale' which started counting from zero and kept on incrementing it by 1 whenever we approach a rising edge of the clock (when the waveform of inbuilt clock goes from 0 to 1), until we reach a certain value. Upon reaching that value, it resets itself to 0 again and the same process repeats itself.
Now, we wish to slow down the clock to 10 milliseconds I.e. to a frequency of 100 Hz . This means that if we combine every 10^6 cycles of the inbuilt clock we would get an output waveform which has a time period of 10 milliseconds. This would mean that for $10^6/2 = 5 \times 10^5$ cycles the original clock of the board would be at zero and for 5×10^5 cycles, it would be at 1. So, the value up to which we would keep incrementing the counter 'prescale' until it reaches the value 5×10^5.
Hence, the output waveform of such a circuit will that will have a time period of 10 milliseconds which is within the allowed range from 0 to 16 milliseconds.
In the assignment, we have created two clocks, 'clock divider' and 'new_clk' whose outputs are fed as inputs to the selector lines s0 and s1 of the 4 × 1 multiplexer respectively. For this reason (since all 4 combinations of s0 and s1 to be possible I.e. 00,01,11,10), we have purposely set the time period of the new clock equal to half the time period of the clock divider.
These two clocks have been combined in the file final_clk which takes the inbuilt clock of basys board as input and generates 2 buffers (since they are outputs and also are used to calculate x0,x1,x2,x3) o11 and o22 (which then become selector lines) and additional 4 outputs x0,x1,x2 and x3 corresponding to the anode pin to be chosen depending upon values of o11 and o22. For eg , when o11 =o22 =0 , then x0 = 0, x1= 1,x2=1 and x3=1, thereby activating the rightmost anode pin. Similarly arguments for activating other anode pins.

Following is the snapshot of the simulation of final_clk (waveform is not completely visible since the value of prescale is high):



Multiplexer:

Finally, our 4 x 1 Multiplexer takes 4 inputs each of 4 bits , so total 16 bit input(in sets of 4 each). The selector lines s0 and s1 of the Multiplexer are fed values of the waveforms of clock divider and new clk respectively. It will generate one output of 4 bits which will then be fed to the 7 segment decoder. The Multiplexer is functioned in such a way that when the selector lines are 00 (I.e x3x2x1x0 : 1110) the rightmost anode is activated, 01 (I.e x3x2x1x0 : 1101) the second right most anode is activated, 11 (I.e x3x2x1x0 : 1011) the second anode from left is activated and 10 (I.e x3x2x1x0 : 0111) the first anode from left is activated. Following this, the Multiplexer outputs the 4 bits corresponding to the anode chosen.
The values of the selector lines s0, s1 keep changing in a cyclic manner from 00 to 01 to 11 to 10 and back to 00. Each of the above four values remains active for 10/4 =2.5 milliseconds. Therefore, the refresh rate of our circuit is 2.5 milliseconds.

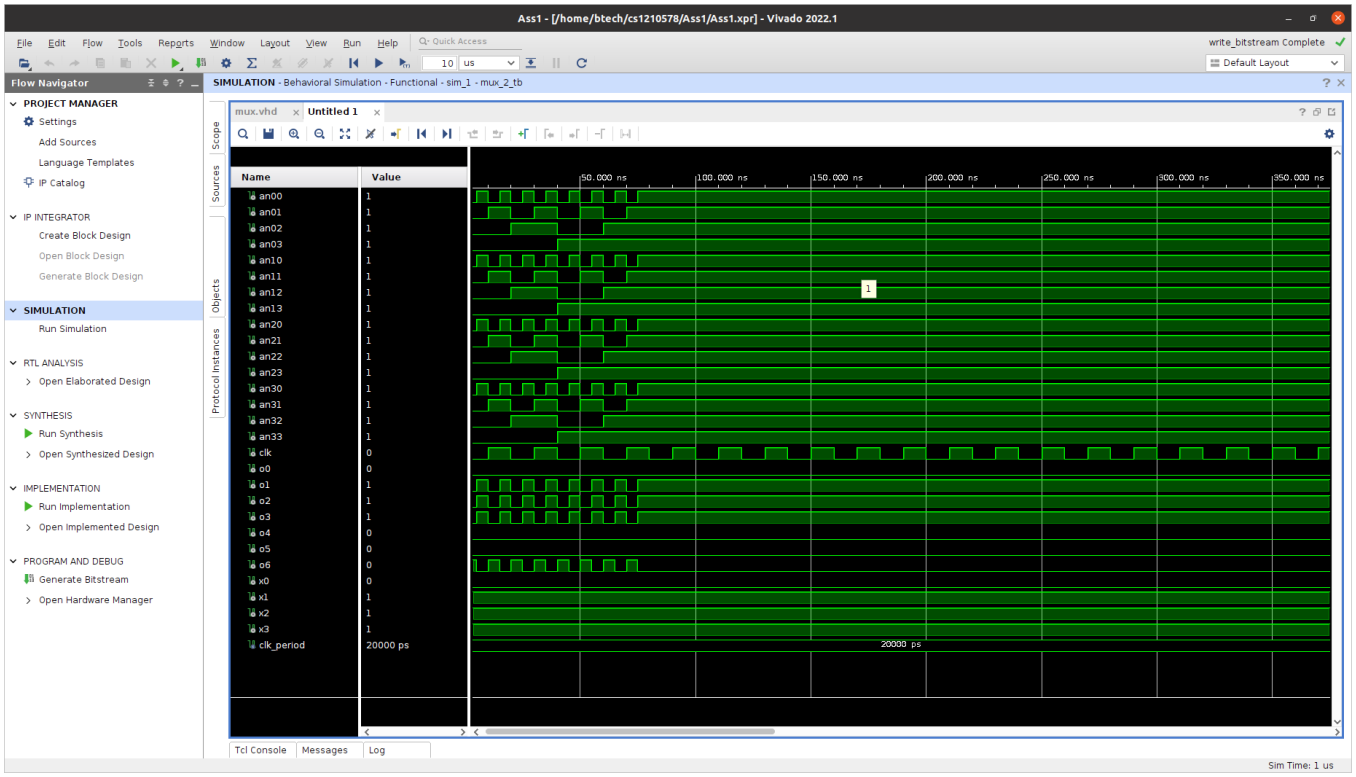
7 Segment Decoder:

This 4 bit output of the Multiplexer is then fed as an input to the 7 segment decoder implemented in part 1 of the assignment. The 7 segment decoder decodes the 4 bit input and generates a 7 outputs , thereby lighting up the LEDs corresponding to the digit to be displayed.

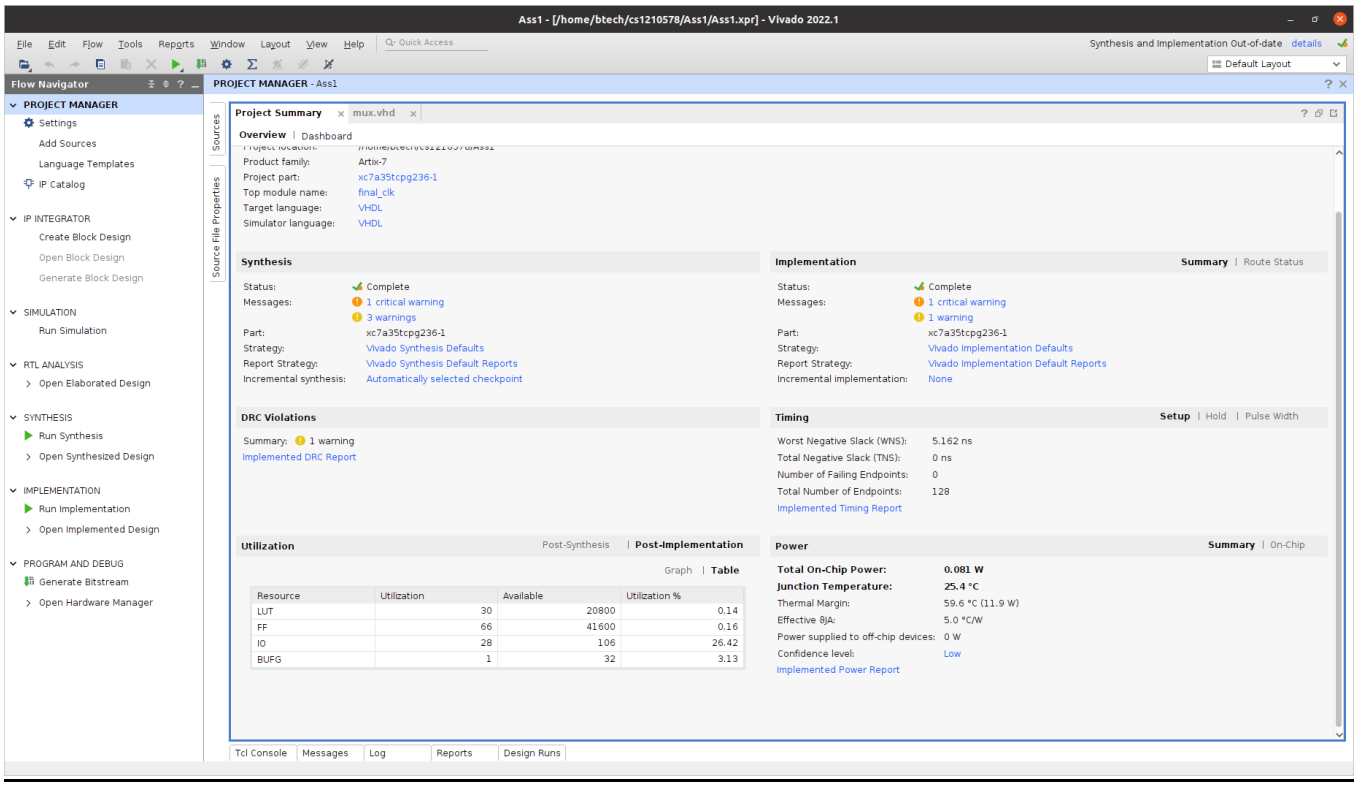
Summary:

In the file mux of our assignment, we have combined the seven segment decoder, multiplexer and the timing circuit. Therefore, it takes total 16 inputs (and one input is inbuilt clock) and generates 7 outputs (of seven segment decoder) and another 4 outputs for activating the correct anode pin. The 4 outputs are directly given as arguments in the constraint file.

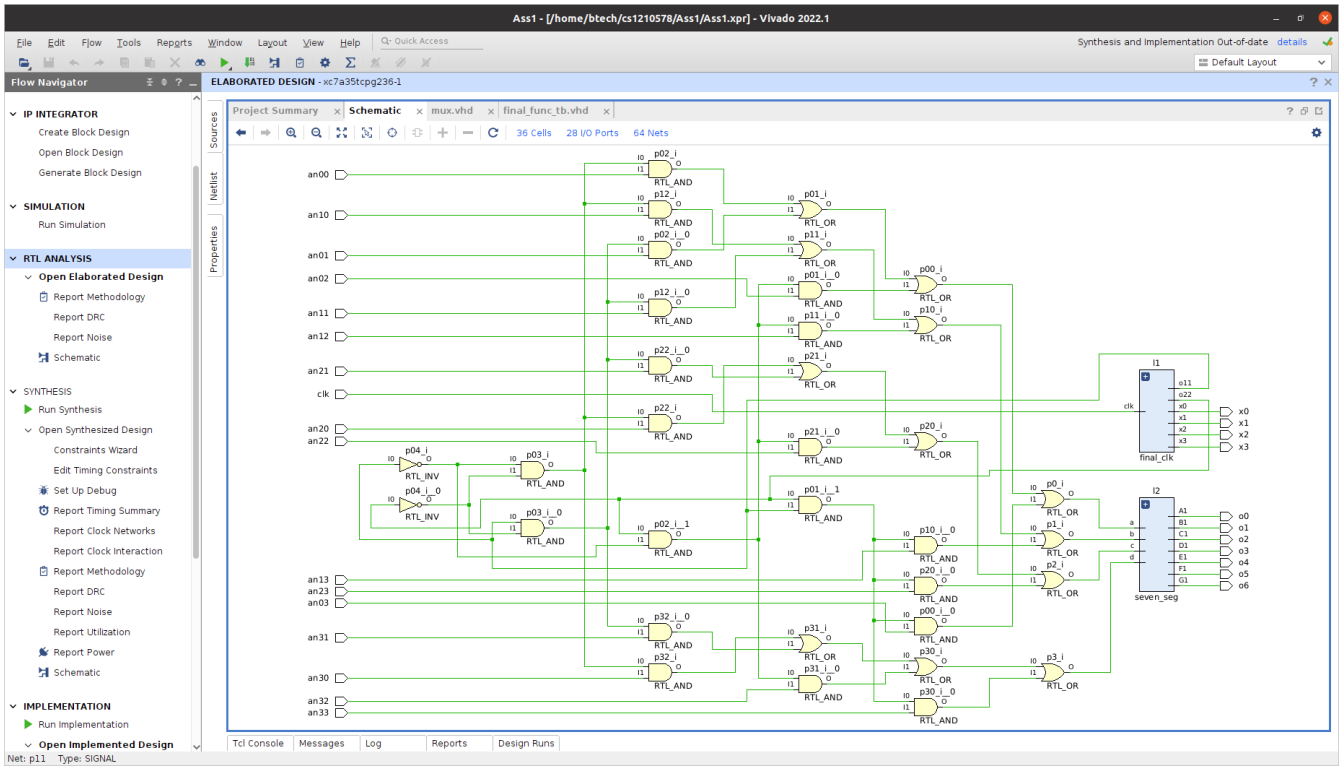
Following is the snapshot of simulation of file mux:



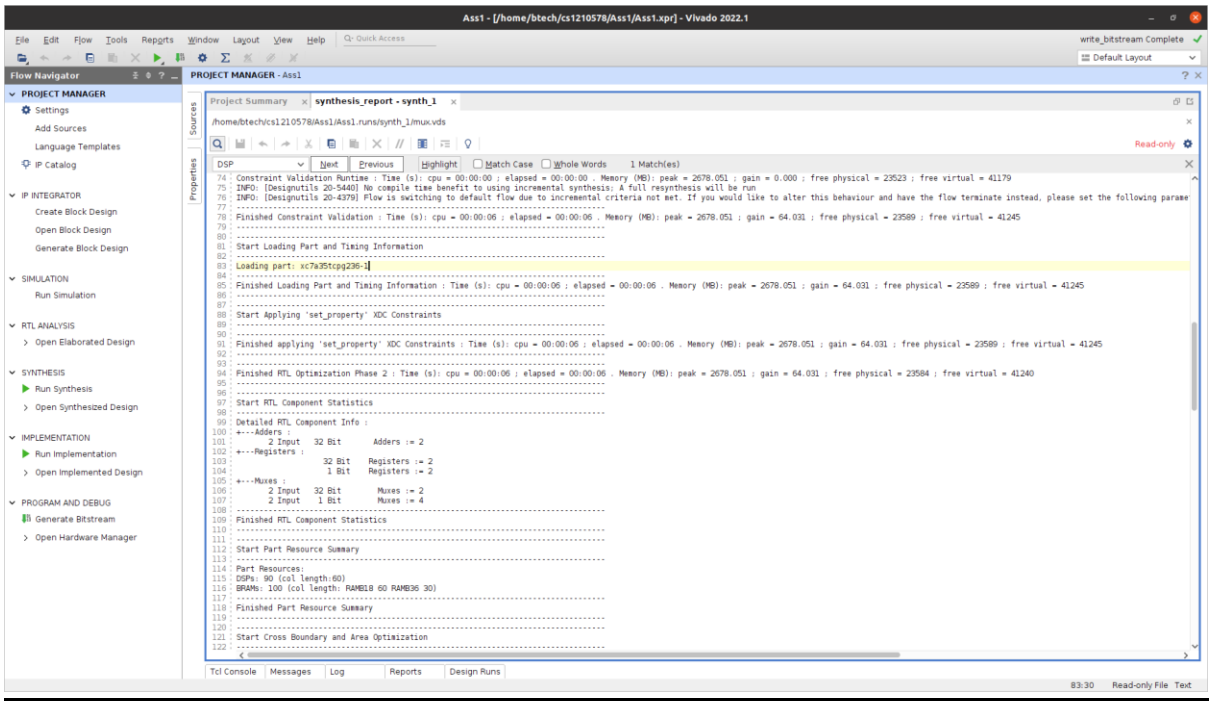
Synthesis Report:



Elaborated Design:



BRAM:


Ass1 - [/home/btech/cs1210578/Ass1/Ass1.xpr] - Vivado 2022.1
File Edit Flow Tools Reports Window Layout View Help
write_bitstream Complete
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
PROJECT MANAGER - Ass1
Project Summary
synthesis_report - synth_1
/home/btech/cs1210578/Ass1/Ass1.runs/synth_1/mux.vds
Read-only
DSP
Next
Previous
Highlight
Match Case
Whole Words
1 Match(es)
Constraint Validation Runtime : Time (s): cpu = 00:00:00 ; elapsed = 00:00:00 . Memory (MB): peak = 2678.051 ; gain = 0.000 ; free physical = 23523 ; free virtual = 41179
INFO: [Designutils 20-5440] No compile time benefit to using incremental synthesis; A full resynthesis will be run
INFO: [Designutils 20-4379] Flow is switching to default flow due to incremental criteria not met. If you would like to alter this behaviour and have the flow terminate instead, please set the following parame
Finished Constraint Validation : Time (s): cpu = 00:00:06 ; elapsed = 00:00:06 . Memory (MB): peak = 2678.051 ; gain = 64.031 ; free physical = 23589 ; free virtual = 41245
Start Loading Part and Timing Information
Loading part: xc7a35tcpg236-1
Finished Loading Part and Timing Information : Time (s): cpu = 00:00:06 ; elapsed = 00:00:06 . Memory (MB): peak = 2678.051 ; gain = 64.031 ; free physical = 23589 ; free virtual = 41245
Start Applying 'set_property' XDC Constraints
Finished applying 'set_property' XDC Constraints : Time (s): cpu = 00:00:06 ; elapsed = 00:00:06 . Memory (MB): peak = 2678.051 ; gain = 64.031 ; free physical = 23589 ; free virtual = 41245
Finished RTL Optimization Phase 2 : Time (s): cpu = 00:00:06 ; elapsed = 00:00:06 . Memory (MB): peak = 2678.051 ; gain = 64.031 ; free physical = 23584 ; free virtual = 41240
Start RTL Component Statistics
Detailed RTL Component Info :
+---Adders :
2 Input 32 Bit Adders := 2
+---Registers :
32 Bit Registers := 2
1 Bit Registers := 2
+---Muxes :
2 Input 32 Bit Muxes := 2
2 Input 1 Bit Muxes := 4
Finished RTL Component Statistics
Start Part Resource Summary
Part Resources:
DSPs: 90 (col length:60)
BRAMs: 100 (col length: RAMB18 60 RAMB36 30)
Finished Part Resource Summary
Start Cross Boundary and Area Optimization
Tcl Console
Messages
Log
Reports
Design Runs
83:30
Read-only File Text