Lab Notebook

FPGA Capstone Project

## Academic Integrity

*By signing below you acknowledge the CU Honor Code, "On my honor, as a University of Colorado Boulder student I have neither given nor received unauthorized assistance" applies to this assignment.*

Signed: Din Carmon

# Module 1

## Setup

Author: Din Carmon

Date: 8/5/22

Procedure/Description of Test:

Observations:

* On Connecting the board to the computer the following happens:
  + 7-Seg display go from 0 to F in hex format with a ~0.5 sec display for a digit
  + Led light pattern is: even light, uneven light, all lights and repeat
  + D4 (5V led) and conf led are on.

Data:

Images/Drawings:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי  
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

Results:

|  |  |
| --- | --- |
| DE10\_LITE\_Small | |
| Fmax | 300.48MHz |
| Logic Utilization % | <1% |
| # Flip-Flops | * 18 in spi\_ee\_config * 2 in reset delay * 1 for Cont * 3 in oLED * **24 FF in total** |
|  |  |

|  |  |
| --- | --- |
| DE10\_LITE\_Default | |
| Fmax | 96.1MHz |
| Logic Utilization % | 1% |
| # Flip-Flops | * 18 in spi\_ee\_config * 2 in reset delay * 1 for Cont * 3 in oLED * 1 in VGA\_Audio\_PLL * 7 in VGA\_OSD\_RAM * 13 in VGA\_Controller * **45 FF in total** |
|  |  |

Questions:

1. Record your observations of the board behavior once the FPGA is programmed. Does it behave as you might expect?

* Yes. The Default version takes more logic elements due to more peripherals.
* In the default version pulling SW0 up makes the board respond to the accelerometer. Lights are like an air bubble in water
* Connecting the VGA outputs a picture.

Conclusions:

Lessons Learned (What did you learn?):

* How to program an FPGA
* How to have fun!!

## Part 1

Author: Din Carmon

Date: 11.5.22

Procedure/Description of Test:

* HEX0, HEX1 as a 7-SEG display of SW3-0 and SW7-4 binary value respectively

Observations:

* The display matches our desired application
* A binary value bigger than 9 output a 7-seg display of 0

Data:

Images/Drawings:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

Results:

|  |  |
| --- | --- |
| Fmax | No constraints were written |
| Logic Utilization % | <1% |
| # Flip-Flops | 0 |
|  |  |

Questions:

1. Based on your observations of the board behavior once the FPGA is programmed, does it behave as you might expect?

* Yes

1. Explain the reason for the number of flip-flops used in the design.

* In principle FF are needed for memory. However there is no clock / memory in our design.

Conclusions:

Lessons Learned (What did you learn?):

* Functions in VHDL
* Basic process rules in VHDL

## Part 2

Date: 11.5.22

Procedure/Description of Test:

* SW3-0 translate to 2-digit binary representation.

Observations:

* As designed

Data:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

Images/Drawings:

Results:

|  |  |
| --- | --- |
| Fmax | No constraints were written |
| Logic Utilization % | <1% |
| # Flip-Flops | 0 |
| With V = 0, z |  |
| With V = 0, A |  |
| With V = F, z |  |
| With V = F, A |  |
|  |  |

Questions:

1. Based on your observations of the board behavior once the FPGA is programmed, does it behave as you might expect?

* YES,

1. Does this design use more or less logic than the design in Part 1? Why?

Less logic. Before it used 15. Now 9 logic elements

Conclusions:

Lessons Learned (What did you learn?):

* If else in VHDL

## Part 3

Date: 12.5.22

Procedure/Description of Test:

* Showing in LEDR an adder output

Observations:

* Works as desired

Data:

Images/Drawings:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

Results:

|  |  |
| --- | --- |
| Fmax | No constraints were written |
| Logic Utilization % | <1% |
| #Flip-Flops | 0 |
|  |  |

Questions:

1. Based on your observations of the board behavior once the FPGA is programmed, does it behave as you might expect?

* YES

Conclusions:

Lessons Learned (What did you learn?):

* Nothing new really…

## Part 4

Date: 12.5.22

Procedure/Description of Test:

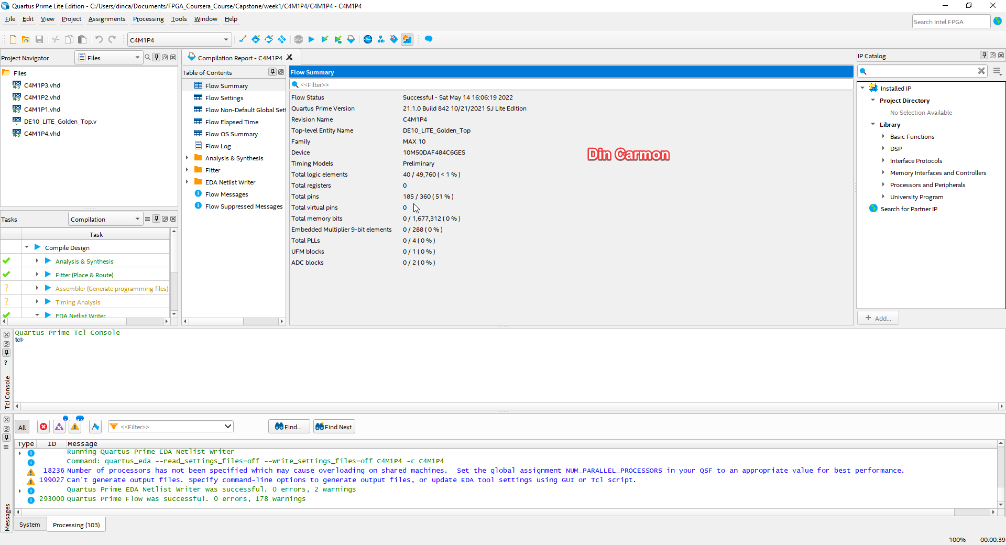
* Combination of all the last parts

Observations:

* As expected

Data:

Images/Drawings:



Results:

|  |  |
| --- | --- |
| # Logic Cells | 40 |
| Logic Utilization % | <1% |
|  |  |
|  |  |

Questions:

1. Based on your observations of the board behavior once the FPGA is programmed, does it behave as you might expect?

* Yes

Conclusions:

Lessons Learned (What did you learn?):

* Using perused modules is awesome

## Part 5

Date: 14.5.22

Procedure/Description of Test:

* Combination of all the last parts

Observations:

Data:

Images/Drawings:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

Results:

|  |  |
| --- | --- |
| # Logic Cells | 40 |
| Logic Utilization % | <1% |
|  |  |
|  |  |

Questions:

1. Based on your observations of the board behavior once the FPGA is programmed, does it behave as you might expect?

* Yes!

1. How does this compare to the number of Logic Cells in Part 4?

* Same! The compiler is smart.

Conclusions:

Lessons Learned (What did you learn?):

* You can trust the compiler to create the required architecture.

# Module 2

## PWM

Author: Din Carmon

Date: 28.5.2022

Procedure/Description of Test:

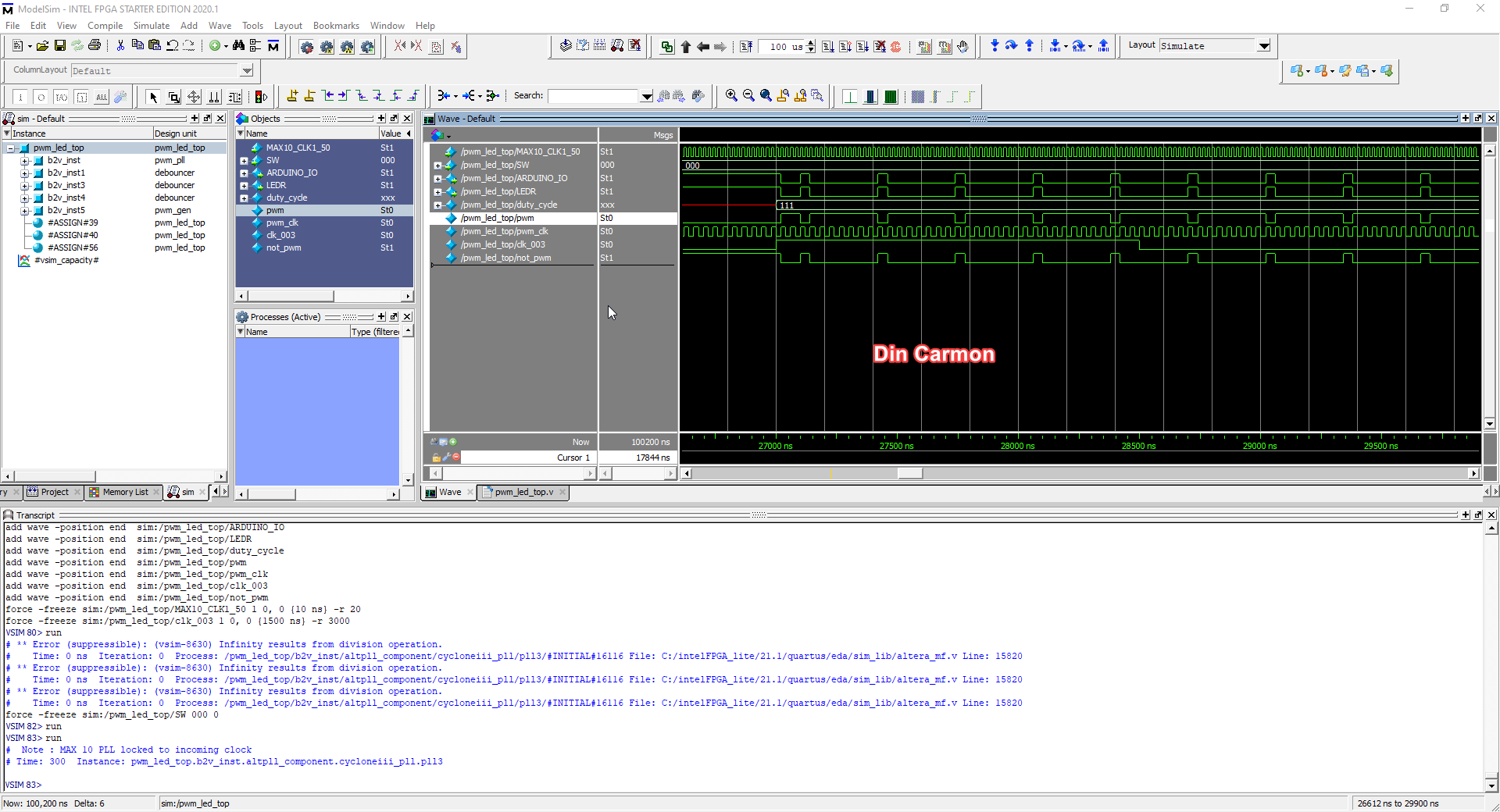
* This projcets creates a PWM clock at a rate of 25/8MHz with 8 levels of duty cycle. The duty cycle is chosen by client through SW2-0.
* Main pages:
  + Pwm\_led\_top.bsf – a design diagram we used to connect the modules
  + Pwm\_led\_top.v – Verilog files created from the last page,
  + Debouncer.v – a popular solution for getting a balanced an unnoisy SW reading.
  + Pwm\_gen.v – this module receives a clock and a duty cycle represented in binary and creates a clock with this duty cycle
  + Pwm\_pll.qip – an ip core we generated for getting the clock frequencies we need.

Observations:

* In modelsim we saw a correct behaviour. Changing SW2-0 changes accordingly the pulse received in the output.
* On board test shows difference in LED intensity according to pulse. As a hardware engineer, my guess is that the LED anode holds some capacity average voltage determined by charge to discharge time ration inflicted by the PWM.

Data:

Images/Drawings:



תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

Results:

|  |  |
| --- | --- |
| Fmax | 548.55MHz – for duty cycle clk  602.41MHz – for debouncer clock |
| % Logic Utilization | 33 / 49760  (<1%) |
| Total registers | 31 |
|  |  |

Questions:

1. Did the LED change brightness depending on the setting of the first 3 switches?

* Yes, On board test shows difference in LED intensity according to pulse. As a hardware engineer, my guess is that the LED anode holds some capacity average voltage determined by charge to discharge time ration inflicted by the PWM.

Conclusions:

Lessons Learned (What did you learn?):

* Using the pll ip core
* Simulating ip core of altera in modelsim
* Creating a PWM signal
* PWM signal influence over a led

## ADC

Author: Din Carmon

Date: 28.5.2022

Procedure/Description of Test:

* Connecting 2 pins as described.
* Switching SW2-0 changes output on 7seg display + led intensity

Observations:

* Switching SW2-0 changes output on 7seg display + led intensity

Module explanations:

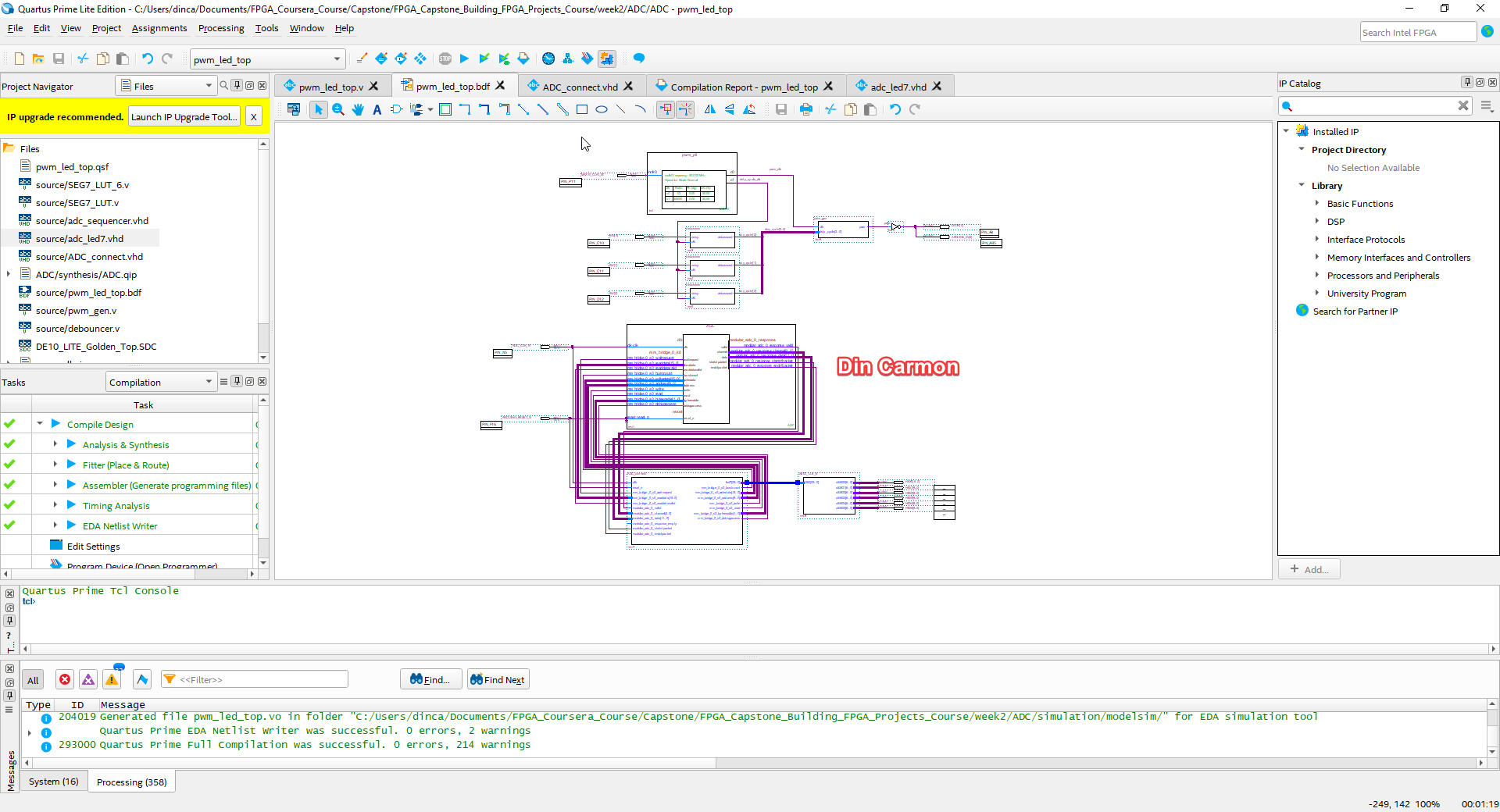
* Pwm\_led\_top.bdf – a design diagram we used to connect the modules
  + Debouncer.v – a popular solution for getting a balanced an unnoisy SW reading.
  + Pwm\_gen.v – this module receives a clock and a duty cycle represented in binary and creates a clock with this duty cycle
  + Pwm\_pll.qip – an ip core we generated for getting the clock frequencies we need for the PWM subproject of this project
  + ADC.qip – A “module” created in Platform Designer. Its application is configuring the ADC ip core with its desired configuration and signals and allowing access and managment to its data by Avalon Stream and Avalon MM from JTAG Debug or from another configurable master.
  + ADC\_Connect – This module is intended for connecting between the ADC output and the LED input. It is not a basic signal match. It includes the following:
    - Adc\_sequencer.vhd - The purpose of this module is to get the ADC running, by writing to the command-status register.
    - Adc\_led7.vhd – intended for finding the stream start of pocket and delivering the ADC output to the 7-SEG array
      * SEG7\_LUT\_6 – receives all the data and output its to all the 7-SEG displays
        + SEG7\_LUT – for each 7\_SEG display

Data:

Images/Drawings:

תמונה שמכילה אלקטרוניקה

התיאור נוצר באופן אוטומטי



Results:

|  |  |
| --- | --- |
| Fmax |  |
| %Logic Utilization | 1102 / 49760 (2%) |
|  |  |
|  |  |

Questions:

1. Does the board behave as you expected?

* Yes. At first I expected to see the voltage as is. But than I remembered that the ADC output a 12bit output which is a ratio of the total 2.5V.
* In addition, sometimes the 7-SEG output is not fully understandable. Some of its led light with low intensity. This is probably due to the voltage not being constant. Thereby we probably measure the ADC too fast and thus the human eye cant distinguish between the two close digits.

1. Is this a good voltmeter as is?

* At first look it seems it does the trick. However it does have a limitation of 2.5V max. Its error rate should be checked according to the ADC datasheet, and so on.

3. What could you change in either the board hardware or FPGA logic to make it perform better?

* Changing the LED output in slower rate, allowing a proper display of the leds
* Doing the bonus mission and outputting a proper decimal view of the voltage

Conclusions:

Lessons Learned (What did you learn?):

* Using the ADC toolkit
* Configuring an ADC ip core
* Using platform designer, especially with Avalon bridge, and buses
* Reading others Verilog files and understanding them

# Module 3

## NIOS II Hardware Design

Author: Din Carmon

Date: 23.7.2022

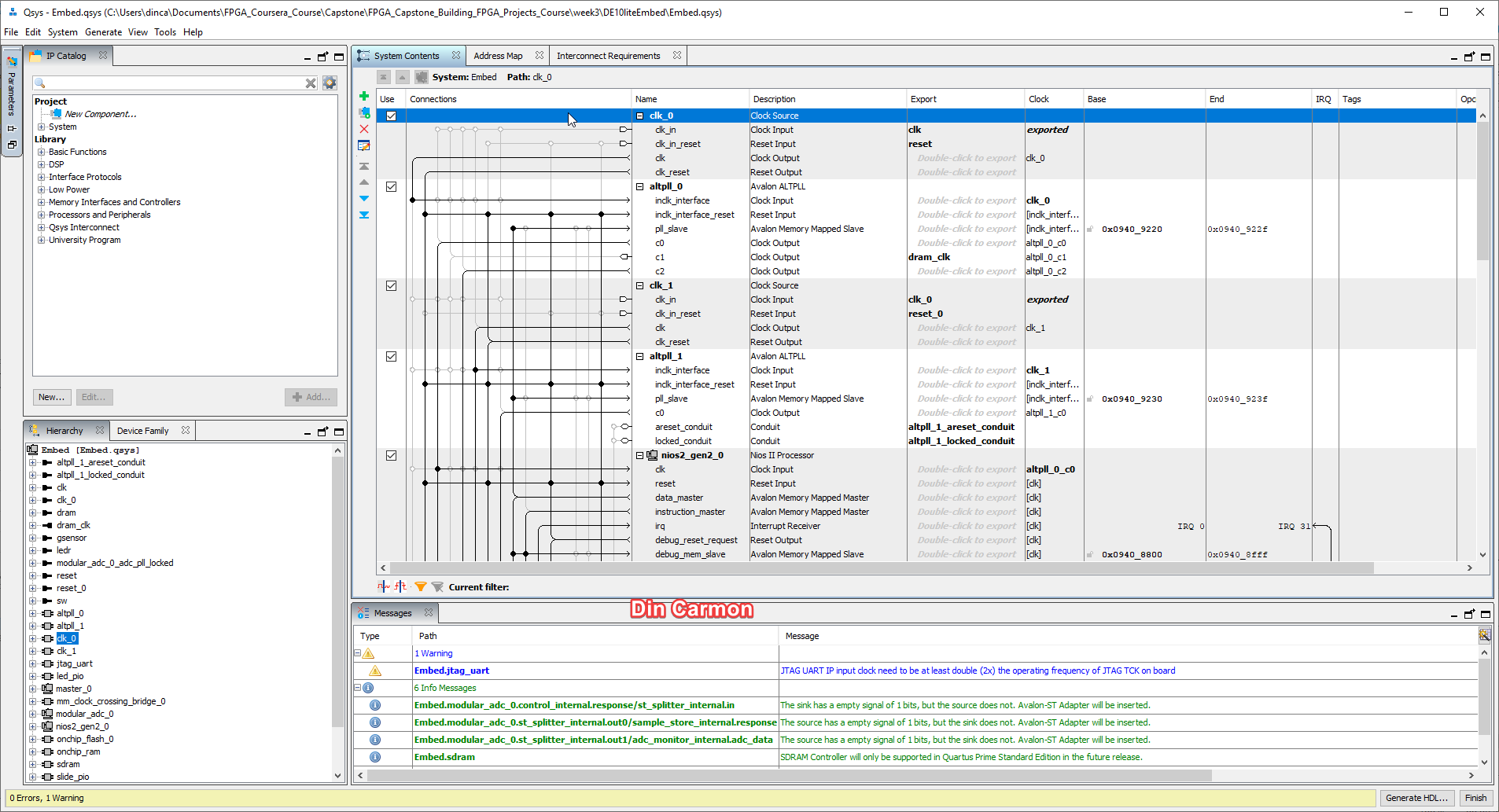
Procedure/Description of Test:

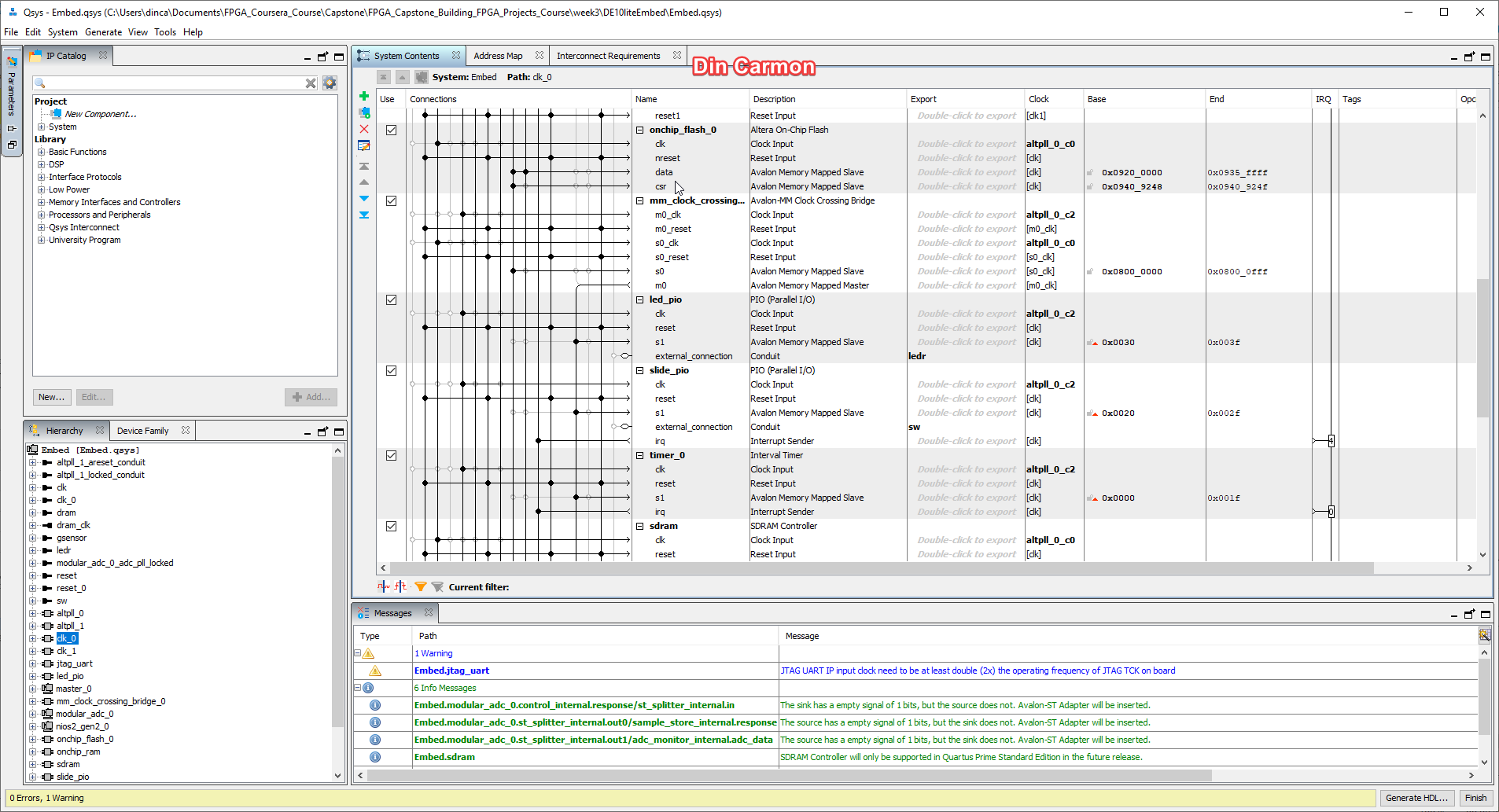
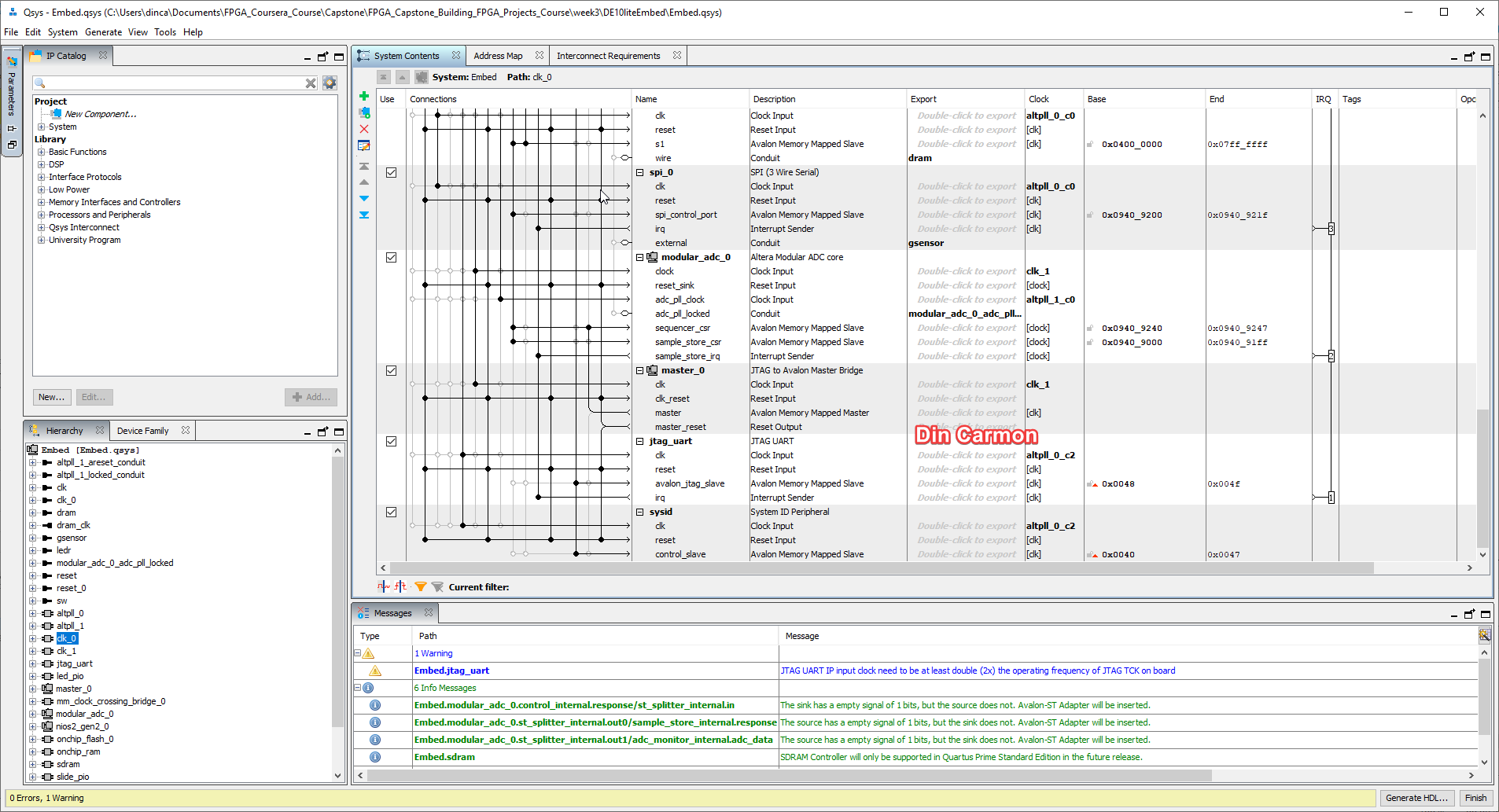
Observations:

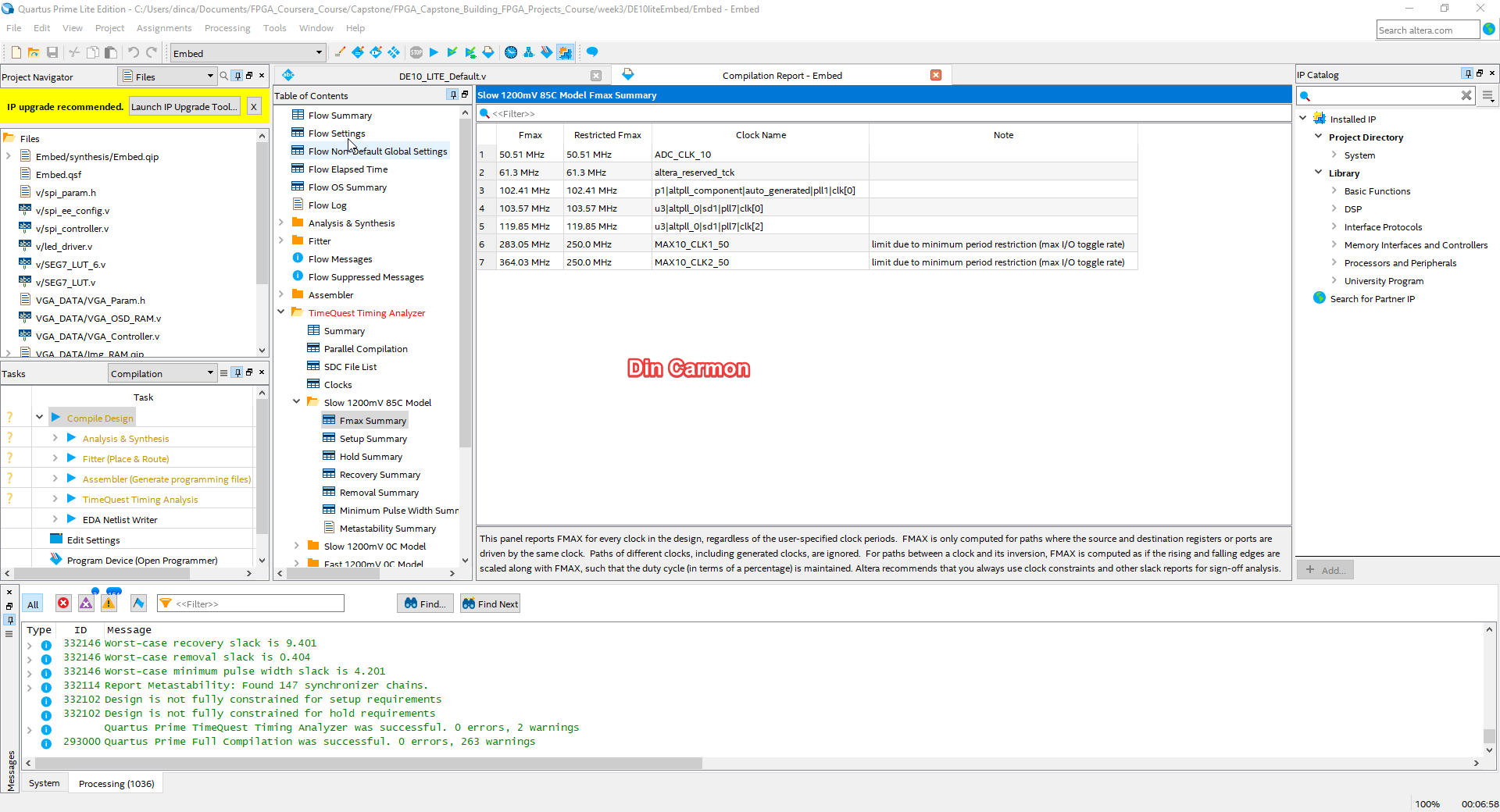
* Followed steps of guide exactly

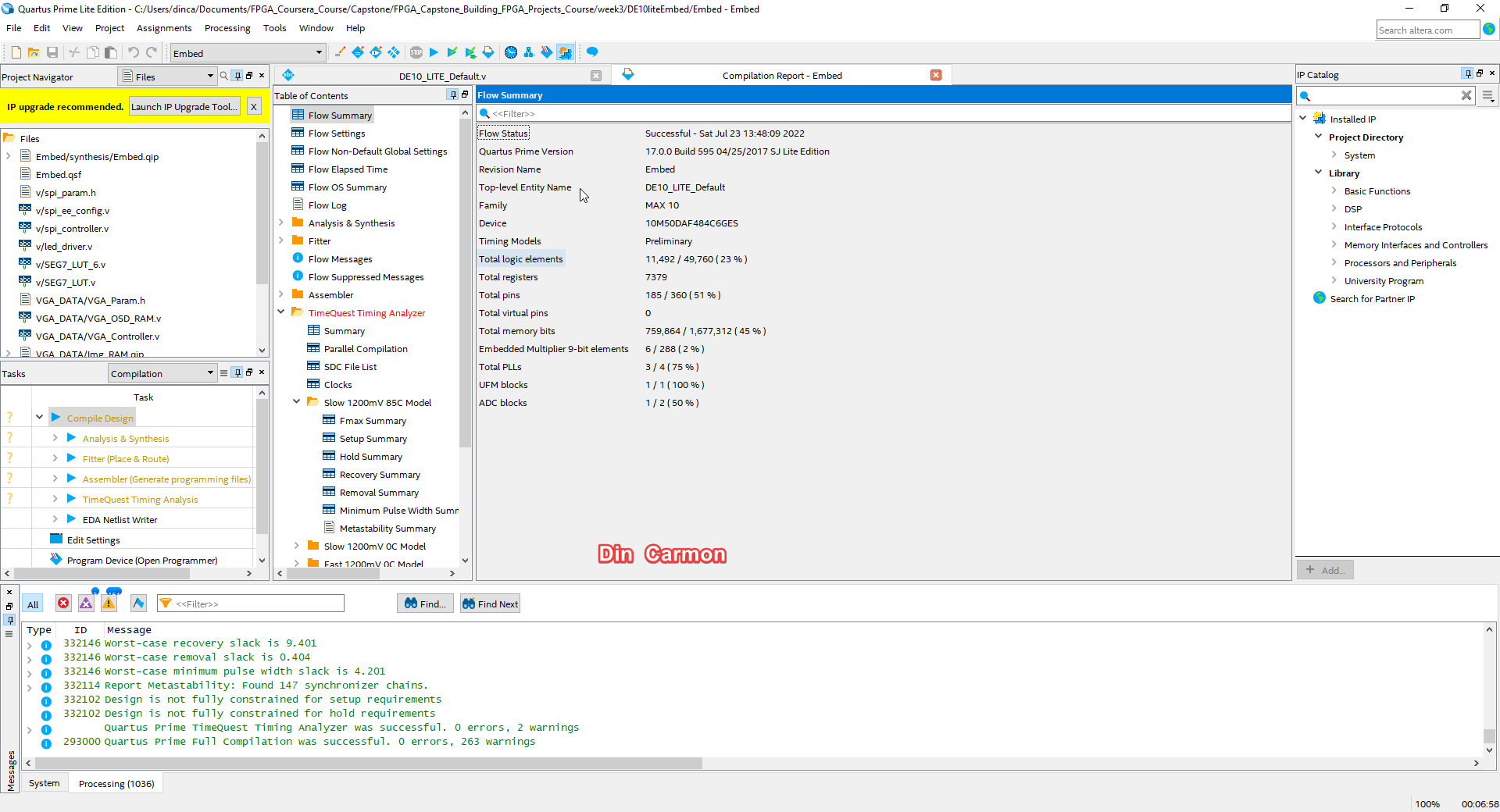
Data:

Images/Drawings:









Results:

|  |  |
| --- | --- |
| Fmax |  |
| Logic Utilization |  |
|  |  |
|  |  |

Questions:

1. What is the purpose of the Avalon Memory-Mapped Clock-Crossing Bridge?

Answer:

As stated in the guide:

“We will place several “slow” peripherals in a separate clock domain from the Nios II processor. With the bridge, a single clock crossing bridge is built into the system for all of the slow peripherals. Peripherals often have requirements to work with a different clock, creating a bridge to them is a general technique to handle the different clock domains. A bridge takes data, addressing and control signals on the Avalon bus, and translates them to signals needed by the peripheral so that data can be exchanged between the devices.”

Conclusions:

Lessons Learned (What did you learn?):

* Creating a full QSys system with nios2.

# Module 4

## NIOS II Software Design and System Test

Author:

Date:

Procedure/Description of Test:

Observations:

Data:

Images/Drawings:

Results:

|  |  |
| --- | --- |
| Fmax |  |
| Logic Utilization |  |
|  |  |
|  |  |

Questions:

1. Is the control of the 10 LEDs implemented in hardware or in software?
2. Is the control of the 7-segment LEDs done by hardware or by software?
3. How much memory is required to run your Nios II program? Can you fit it into the onchip RAM if you redesign the onchip RAM block?
4. Your Nios II processor is running at what clock speed? How much faster can it run in your MAX10 design?

Conclusions:

Lessons Learned (What did you learn?):