Integrated Circuit Design Lab 1

Part 1 Entity

A state machine is a behaviour model that consists of a fixed number of states, where the state destination is based on the inputs given to the state machine. In this section of the report I will explain how I used a finite state machine to solve the design problem of a Prime Number Checker (PNC). I had to determine whether a number between 0 and 63 is a prime number, to do this I had to exploit the mod operation in Vivado to determine whether there were divisors of the input number in question.

Initially to design the FSM, we need to create an entity that provides an external sight of the circuit it is here when we first declare our input and output ports. This is done by first assigning a name that is relevant to the design, we then must assign a mode to the port, in my case keywords **IN/OUT STD_LOGIC** and **IN/OUT STD_LOGIC_VECTOR** were used depending on whether the port created is a output or input as well as a vector. Since the Finite State Machine is a synchronous process our first input port **CLK** is declared and required for our process to change states on each positive clock edge. The next port in use is the **Input**, this port has 3 functions in the process, the main function uses positions **5 downto 0** to represent a number input between 0 and 63 that can be used as the number in question of being a prime number. The next position **6** is used as a User Guess input, where **HIGH** represents a guess of prime and **LOW** represents a guess of not prime, this allows the user to guess whether the number in question a prime number is or not. The last position **7** represents Reset function, this is used as an asynchronous process so that when the reset is **HIGH** it can force a change in the process at any time without the dependence of the **CLK**.

Next, I will go over the output ports declared in the entity, similar to the position **6** in **input**, port **UserGuess** is used as an indicator for the user's guess, this is in the form of the LED where the output is **HIGH** when the user believes the number is prime and **LOW** when the user believes the number is not prime. The next output port in use is the **PrimeLed**, after running the process this port determined whether the input number in question was Prime. The output was **HIGH** when the number was discovered to be prime and **LOW** when the number was not prime. The last port holds **GuessResult,** the value associated with this port was evaluated on whether the user's guess was correct in determining whether the number was prime or not. This value is determined after the process correctly identifies the number and is **HIGH** if the user's guess is correct and **LOW** if the user's guess is incorrect.

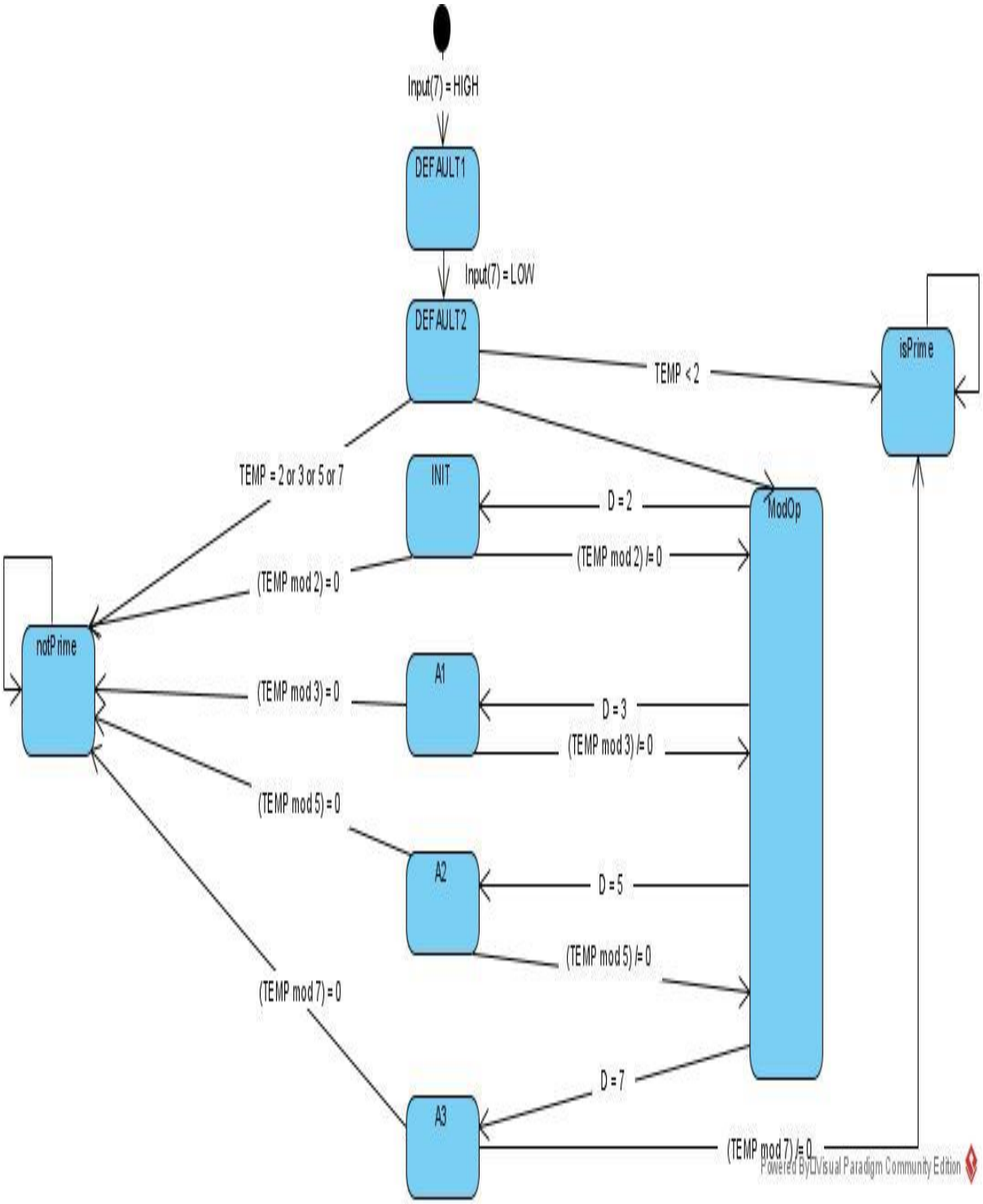
Part 1 Architecture

The VHDL state machine is described in the architecture of the VHDL code. We define a state type using an enumeration syntax that helps the compiler determine the values of the state types listed. To process the input information, the tasks are split into different states with relevant names, I used the states: **DEFAULT1,DEFAULT2, ModOp, INIT ,A1, A2, A3, isPrime, notPrime**. The 9 states listed made evaluating whether the input number was prime, along with the implementation of external events such as the Reset simpler to compute.

Upon beginning the behavioural before the core part of the state machine is written, we must implement a clock process that allows us to change state on a positive clock edge. **SYNC_PROC : PROCESS(CLK)**, we define our process using this line of code with **CLK** in the sensitivity list letting us know that the process is sensitive to the clock. On every positive clock edge we use **STATE <= NSTATE;** to map the value of the next state determined by the state machine to the current state value to ensure our process is synchronous. Although our process is synchronous, the reset signal we have implemented is asynchronous, so it is implemented in an if statement after we change the value of state to next state, this way the code is not run on every positive clock edge. In this reset if statement we assign the signal State to **DEFAULT1**, therefore forcing the state machine back to its default state.

In the next part of code we use **STATE_PROC: PROCESS(CLK)** to initialise the core part of our state machine. Like the **SYNC_PROC** process **STATE_PROC** includes **CLK** in its sensitivity list, to let us know that it is also sensitive to the clock. Upon beginning the process before our case statement the line of code **TEMP <= to_integer(unsigned(Input(5 downto 0)));** is used, this converts the 6 least significant bits associated with **Input** to an integer and assigns this value to the signal **TEMP,** this conversion is done so that we can exploit the **MOD** function without developing another function to complete this comparison.

Appendix Finite State Machine Diagram
(see text below for description)



Part 1 Finite State Machine Description

(STATE DEFAULT1)

The case statement begins here with the first state in question being the **DEFAULT1** state. This state is called upon whenever the reset(**Input(7)**) is actively **HIGH** and remains here until the reset(**Input(7)**) is set **LOW**. The input is sampled after the reset so **NSTATE** is set to **DEFAULT2** whilst it is in this state so that on a positive **CLK** edge the state is set to the initial state.

(STATE DEFAULT2)

The state **DEFAULT2,** is where **UserGuess** is initially determined via the input position 6**,** since we are using the MOD function to divide the number in question by the integers 2,3,5 and 7 as our prime number checking method. We check whether the number in question is 2,3,5 and 7 using an if statement(**IF(TEMP<=2)**), this will make sure these numbers are not mistaken for not being a prime number. If none of these cases are met, we set **NSTATE** to **ModOP** along with setting the signal **D** to integer 2, with the signal **D** representing the current divisor.

(STATE ModOp)

The state **ModOp,** is where the mod operation is completed on the signal **TEMP** for each of the divisors 2,3,5 and 7. We assign the remainder of this operation to the signal **R** to be used in later states. If statements are also used to compare the signal **D** with the divisors to determine what the value of **NSTATE** should be.

(STATE INIT)

The state **INIT,** is where we check the value of the remainder **R**. If found to be 0, we set our **LED4** to "0010" to represent our lowest common divisor along with setting our next state **NSTATE** to **notPrime**. The else case sets **NSTATE** to **ModOp** along with setting **D** to integer 3 to check for the next prime number case.

(STATE A1)

The state **A1,** is where we check the value of the remainder **R**. If found to be 0, we set our **LED4** to "0011" to represent our lowest common divisor along with setting our next state **NSTATE** to **notPrime**. The else case sets **NSTATE** to **ModOp** along with setting **D** to integer 5 to check for the next prime number case.

(STATE A2)

The state **A2,** is where we check the value of the remainder **R**. If found to be 0, we set our **LED4** to "0101" to represent our lowest common divisor along with setting our next state **NSTATE** to **notPrime**. The else case sets **NSTATE** to **ModOp** along with setting **D** to integer 7 to check for the next prime number case.

(STATE A3)

The state **A3,** is where we check the value of the remainder **R**. If found to be 0, we set our **LED4** to "0111" to represent our lowest common divisor along with setting our next state **NSTATE** to **notPrime.** The else case sets **NSTATE** to **isPrime**, to confirm the number in question is a prime number.

(STATE isPrime)

The state **isPrime**, is the destination state if the **TEMP** number is determined to be prime, it is here where we set **PrimeLed HIGH** and evaluate the **UserGuess** and determine whether the user was correct in their guess of the number being prime. If correct we will set the **GuessResult** led **HIGH,** if incorrect we set this led **LOW**.

(STATE notPrime)

The state **notPrime**, is the destination state if the **TEMP** number is determined to not be prime, it is here where we set **PrimeLed LOW** and evaluate the **UserGuess** and determine whether the user was correct in their guess of the number not being prime. If correct we will set the **GuessResult** led **HIGH,** if incorrect we set this led **LOW**.

Discussion

The design in question was able to accurately determine whether the input number was prime or not. I have also anticipated errors such as the code mistaking the integer 1 as a prime number in addition to mistaking the divisors 2,3,5 and 7 as not prime. Timing issues have been avoided using the different ports in the sensitivity list of the processes, as well as implementing multiple processes to split the tasks in the specification. A limitation in the code appears when we want to apply multiple inputs consecutively. With consecutive inputs we cannot implement a reset, in addition we cannot loop the code back to initial without it leading to errors in testing on the ELVIS development board, however this was not a requirement in the specification so the design function as intended.

Part 1 Testbench

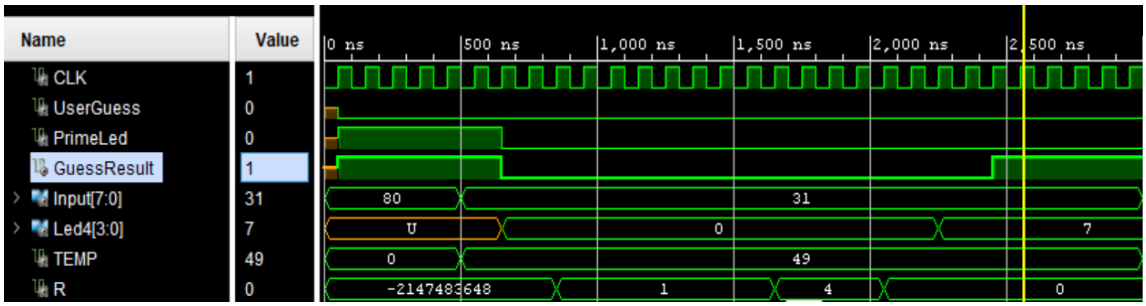
Initially to test the prime number checker, we must design a testbench that will simulate the test runs of the checker. The **Pnc_FSM** model is used as a component in the testbench as a link to the original **Pnc_FSM** design source. To map the component values to the signal values we have created port map is used, which allows us to use these port values in the process. before we begin simulating the input values, we must create a process **clk_proc** to simulate the positive and negative edges of the clock, this is due to our state machine being a synchronous process so **CLK** is needed to trigger the **state_proc** process in **Pnc_FSM** and initiate any state changes in the model.

The process **sim_proc** is used to simulate the input to the state machine. In the simulation, since we are implementing our code on the Elvis development board, in the final states isPrime and notPrime we do not loop back to the initial state so we do not have to worry about the timing of the inputs. To prevent clashing of input processes we use a wait time longer than necessary in all our inputs **2000ns** in comparison to our **CLK** wait time **50ns.**

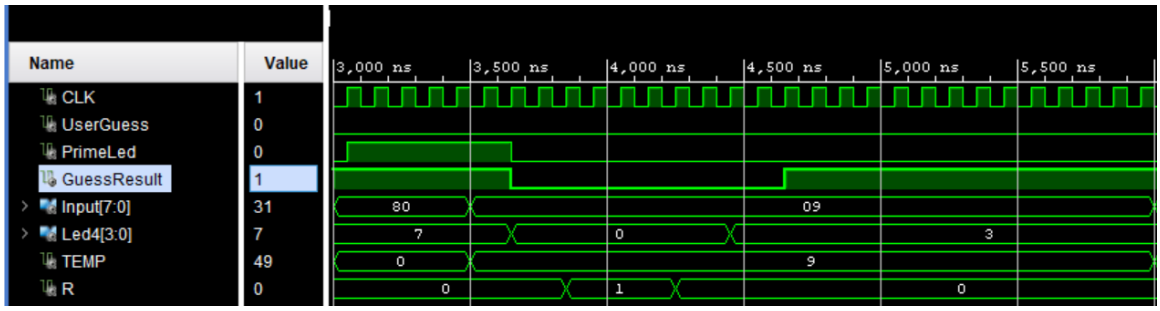
To test the possible cases we simulate the number cases where: TEMP equals our divisor, TEMP is less than 2, TEMP is a prime number, TEMP is not a prime number and our cases of UserGuess being HIGH and LOW to test our GuessResult output. Following the specification we also know that the reset(Input(7)) must be set high before each input, so we follow this in the simulation by using **Input <= "10000000"** before each input to simulate the reset.

Appendix Testbench Simulation capture

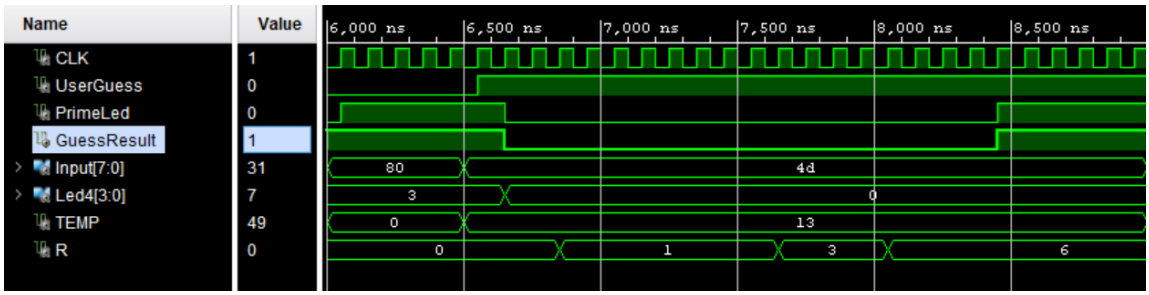
First Case where TEMP = 7 which is one of the divisors and a prime number



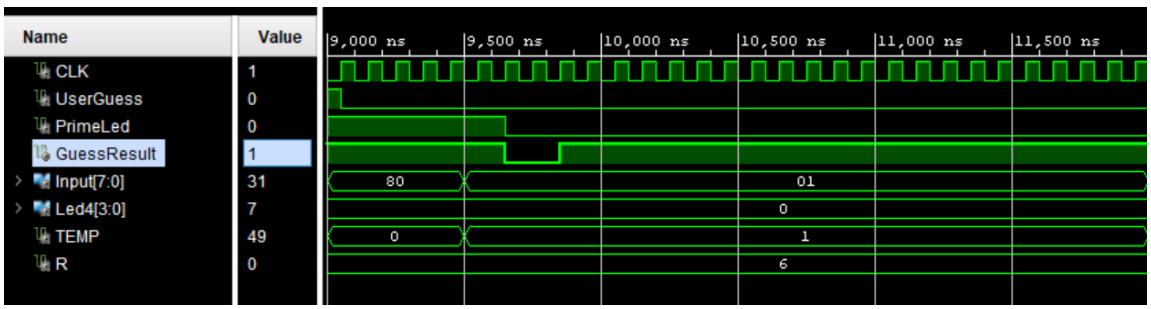
Second Case where TEMP = 9 which is not a prime number where we guessed correctly



Third Case where TEMP = 13 which is a prime number which we have guessed correctly.



Fourth Case where TEMP = 1 which is not a prime number which we have guessed correctly.



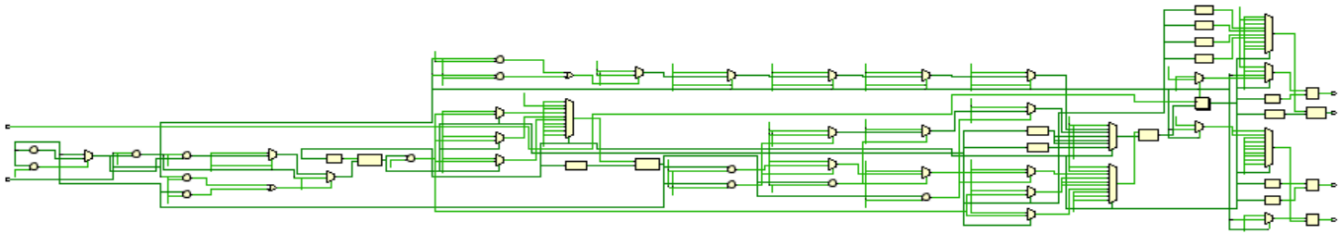
Part 2 RTL and Synthesis analysis

The VHDL synthesis tool is used to take high level descriptions such as VHDL code, and convert it into low level descriptions that the FPGA(Field Programmable Gate Arrays) can understand. FPGA's consists of a 2D array of flip flops and logic blocks, in VHDL we use RTL analysis to turn our synthesized code into a design implementation. Tools such as partitioning, floor planning and placement is used to split the circuit into various blocks with systems in place to minimise the chip usage area. Before synthesis we must ensure our code in synthesizable, statements such as wait that were previously used in simulation cannot be used since the FPGA's has no internal timers and can only function with clock inputs. A similar practical application comes with ASIC(Application Specific Integrated Circuit) which is similar to FPGA but is not reprogrammable, since the circuits are permenantly drawn into silicon.

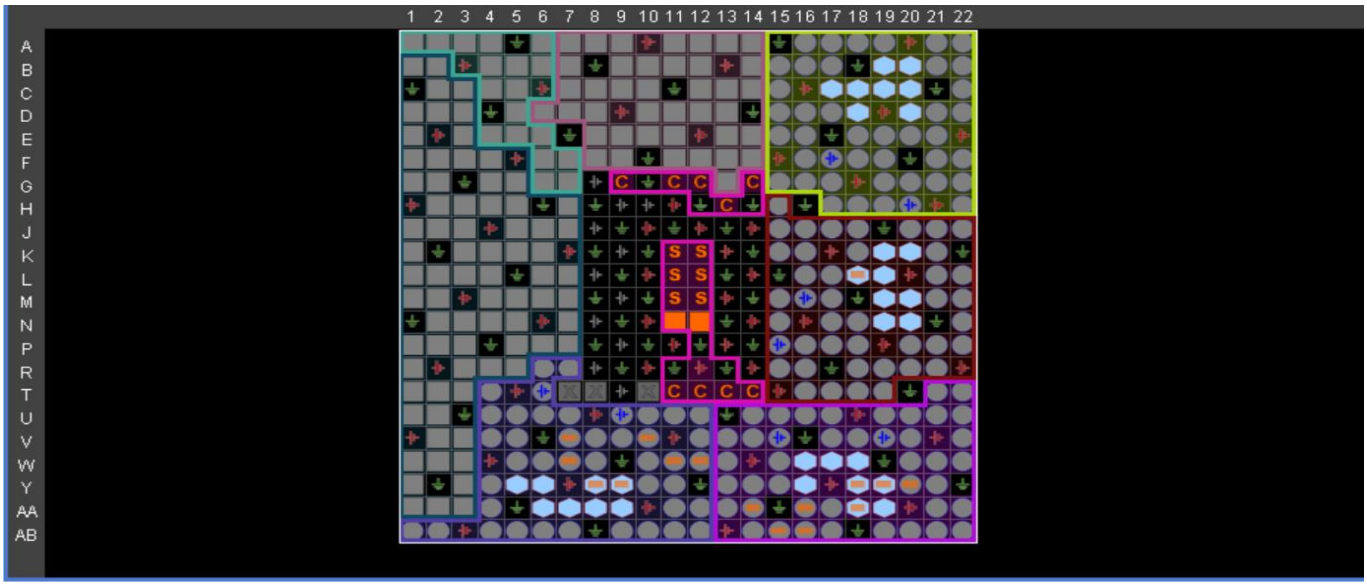
Since the clock is the only concept of time available in the FPGA's, we see the importance of our sequential logic processes which require a clock to look at the previous states/values. In our case we use two processes that operate concurrently that is clocked using sequential logic, in these processes sequential statements such as IF THEN ELSE and CASE statements are used. This shows the importance of sequential logic in achieving the intention behind the finite state machines in use.

Appendix RTL and Synthesis Schematic

RTL schematic



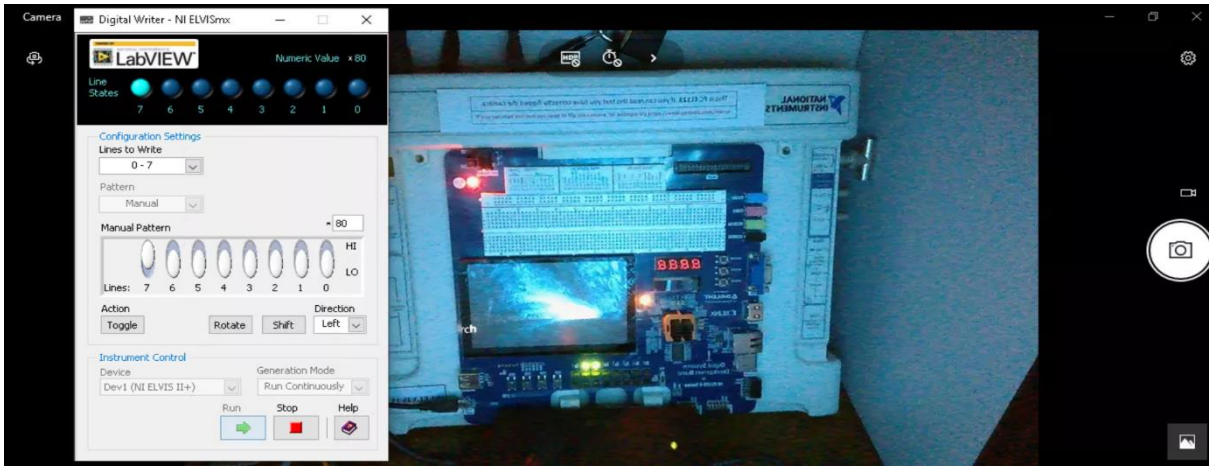
Synthesis Design



Part 3 Implementation evaluation with captures

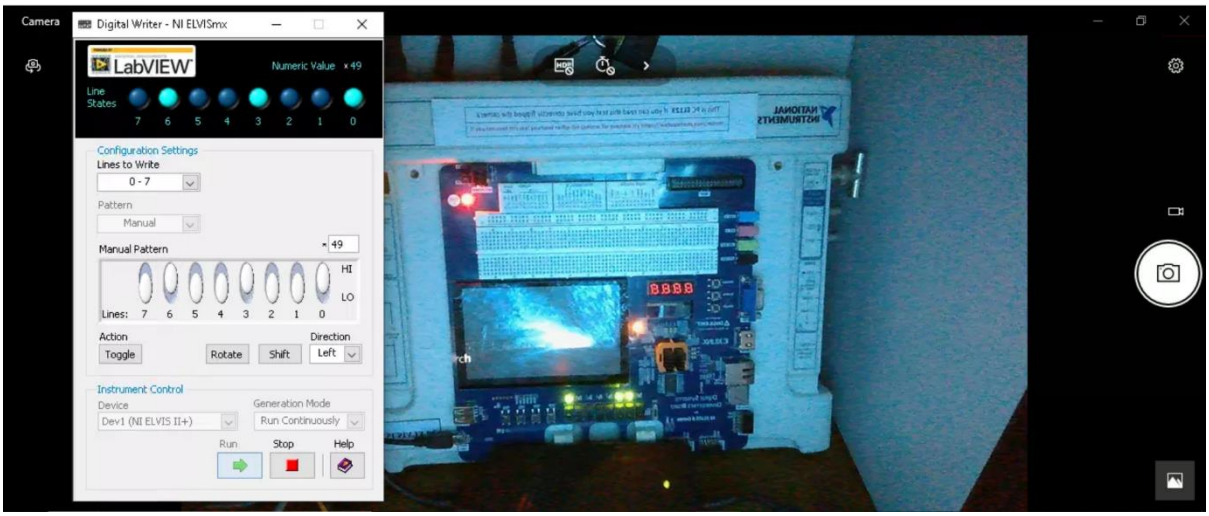
In the implementation of the development kit, the prerequisites were to update and uncomment necessary pins in the master constraint file. This is necessary to link the Pnc_FSM ports to the ELVIS development board, to visualise the output ports of Pnc_FSM we use the LED pins and to show the input ports we use the Elvis port pins. Lastly, we must uncomment the Clock signal since the process is synchronous. To test the implementation virtually, RGATE was used to access the ELVIS development board. Upon generating the bitstream, to test the input cases the NI Launcher Digital Writer tool was used to simulate the 7-bit input. In the captures below we see how the outputs LED's correspond to the different inputs

Reset implementation with initial case GuessResult and PrimeLed set HIGH

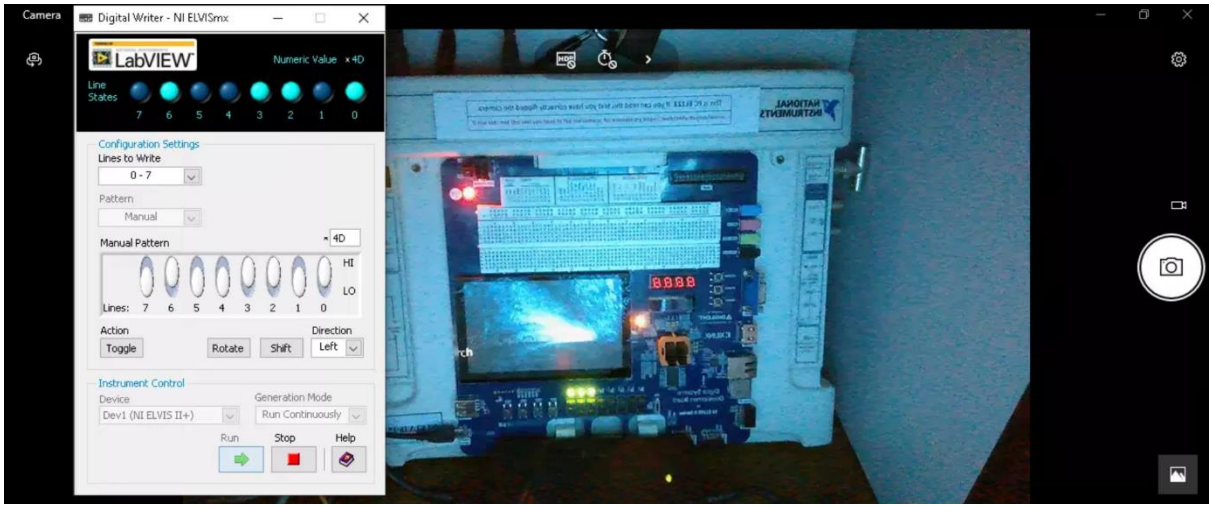


the input integer 9 with Lowest common divisor 3 shown, GuessResult also set HIGH

the input integer 9 with lowest common divisor 3 shown, UserGuess set HIGH to test GuessResult functionality



the input integer 13, with UserGuess, PrimeLed and GuessResult set HIGH shown



Appendix – Synthesis Design Report

#---

Vivado v2017.2 (64-bit)

SW Build 1909853 on Thu Jun 15 18:39:09 MDT 2017

IP Build 1909766 on Thu Jun 15 19:58:00 MDT 2017

Start of session at: Mon Mar 29 21:01:12 2021

Process ID: 30308

Current directory: C:/Users/Nader/Downloads/FINALICD/FINALICD.runs/synth_1

Command line: vivado.exe -log Pnc_FSM.vds -product Vivado -mode batch -messageDb vivado.pb -notrace -source Pnc_FSM.tcl

Log file: C:/Users/Nader/Downloads/FINALICD/FINALICD.runs/synth_1/Pnc_FSM.vds

Journal file: C:/Users/Nader/Downloads/FINALICD/FINALICD.runs/synth_1\vivado.jou

#---

source Pnc_FSM.tcl -notrace

Command: synth_design -top Pnc_FSM -part xc7z020clg484-1

Starting synth_design

Attempting to get a license for feature 'Synthesis' and/or device 'xc7z020-clg484'

INFO: [Common 17-349] Got license for feature 'Synthesis' and/or device 'xc7z020-clg484'

INFO: Launching helper process for spawning children vivado processes

INFO: Helper process launched with PID 26488

Starting RTL Elaboration : Time (s): cpu = 00:00:05 ; elapsed = 00:00:05 . Memory (MB): peak = 307.539 ; gain = 77.297

INFO: [Synth 8-638] synthesizing module 'Pnc_FSM' [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:48]

WARNING: [Synth 8-614] signal 'TEMP' is read in the process but is not in the sensitivity list [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:71]

WARNING: [Synth 8-614] signal 'D' is read in the process but is not in the sensitivity list [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:71]

WARNING: [Synth 8-614] signal 'R' is read in the process but is not in the sensitivity list [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:71]

INFO: [Synth 8-256] done synthesizing module 'Pnc_FSM' (1#1) [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:48]

Finished RTL Elaboration : Time (s): cpu = 00:00:06 ; elapsed = 00:00:06 . Memory (MB): peak = 346.867 ; gain = 116.625

Report Check Netlist:

+------+------------------+-------+---------+-------+------------------+

| |Item |Errors |Warnings |Status |Description |

+------+------------------+-------+---------+-------+------------------+

|1 |multi_driven_nets | 0| 0|Passed |Multi driven nets |

+------+------------------+-------+---------+-------+------------------+

Finished RTL Optimization Phase 1 : Time (s): cpu = 00:00:06 ; elapsed = 00:00:06 . Memory (MB): peak = 346.867 ; gain = 116.625

INFO: [Device 21-403] Loading part xc7z020clg484-1

INFO: [Project 1-570] Preparing netlist for logic optimization

Processing XDC Constraints

Initializing timing engine

Parsing XDC File [C:/Users/Nader/Downloads/DSDB_Master.xdc]

Finished Parsing XDC File [C:/Users/Nader/Downloads/DSDB_Master.xdc]

INFO: [Project 1-236] Implementation specific constraints were found while reading constraint file [C:/Users/Nader/Downloads/DSDB_Master.xdc]. These constraints will be ignored for synthesis but will be used in implementation. Impacted constraints are listed in the file [.Xil/Pnc_FSM_propImpl.xdc].

Resolution: To avoid this warning, move constraints listed in [.Xil/Pnc_FSM_propImpl.xdc] to another XDC file and exclude this new file from synthesis with the used_in_synthesis property (File Properties dialog in GUI) and re-run elaboration/synthesis.

Completed Processing XDC Constraints

INFO: [Project 1-111] Unisim Transformation Summary:

No Unisim elements were transformed.

Constraint Validation Runtime : Time (s): cpu = 00:00:00 ; elapsed = 00:00:00.004 . Memory (MB): peak = 656.285 ; gain = 0.000

Finished Constraint Validation : Time (s): cpu = 00:00:19 ; elapsed = 00:00:23 . Memory (MB): peak = 656.285 ; gain = 426.043

Start Loading Part and Timing Information

Loading part: xc7z020clg484-1

Finished Loading Part and Timing Information : Time (s): cpu = 00:00:19 ; elapsed = 00:00:23 . Memory (MB): peak = 656.285 ; gain = 426.043

Start Applying 'set_property' XDC Constraints

Finished applying 'set_property' XDC Constraints : Time (s): cpu = 00:00:19 ; elapsed = 00:00:23 . Memory (MB): peak = 656.285 ; gain = 426.043

INFO: [Synth 8-802] inferred FSM for state register 'STATE_reg' in module 'Pnc_FSM'

WARNING: [Synth 8-6014] Unused sequential element STATE_reg was removed. [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:62]

WARNING: [Synth 8-6014] Unused sequential element NSTATE_reg was removed. [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:64]

INFO: [Synth 8-5545] ROM "Led4" won't be mapped to RAM because address size (32) is larger than maximum supported(25)

INFO: [Synth 8-5545] ROM "D" won't be mapped to RAM because address size (32) is larger than maximum supported(25)

INFO: [Synth 8-5545] ROM "R1" won't be mapped to RAM because address size (32) is larger than maximum supported(25)

INFO: [Synth 8-5545] ROM "NSTATE" won't be mapped to RAM because address size (32) is larger than maximum supported(25)

INFO: [Synth 8-5545] ROM "NSTATE" won't be mapped to RAM because address size (32) is larger than maximum supported(25)

INFO: [Synth 8-5545] ROM "NSTATE" won't be mapped to RAM because address size (32) is larger than maximum supported(25)

INFO: [Synth 8-5545] ROM "NSTATE" won't be mapped to RAM because address size (32) is larger than maximum supported(25)

INFO: [Synth 8-5544] ROM "PrimeLed" won't be mapped to Block RAM because address size (4) smaller than threshold (5)

INFO: [Synth 8-5544] ROM "R" won't be mapped to Block RAM because address size (4) smaller than threshold (5)

INFO: [Synth 8-5545] ROM "NSTATE" won't be mapped to RAM because address size (32) is larger than maximum supported(25)

INFO: [Synth 8-5545] ROM "NSTATE" won't be mapped to RAM because address size (32) is larger than maximum supported(25)

INFO: [Synth 8-5545] ROM "NSTATE" won't be mapped to RAM because address size (32) is larger than maximum supported(25)

INFO: [Synth 8-5545] ROM "NSTATE" won't be mapped to RAM because address size (32) is larger than maximum supported(25)

INFO: [Synth 8-5544] ROM "NSTATE" won't be mapped to Block RAM because address size (1) smaller than threshold (5)

INFO: [Synth 8-5544] ROM "NSTATE" won't be mapped to Block RAM because address size (1) smaller than threshold (5)

WARNING: [Synth 8-327] inferring latch for variable 'UserGuess_reg' [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:83]

WARNING: [Synth 8-6014] Unused sequential element STATE_reg was removed. [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:62]

WARNING: [Synth 8-327] inferring latch for variable 'NSTATE_reg' [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:64]

WARNING: [Synth 8-6014] Unused sequential element STATE_reg was removed. [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:62]

WARNING: [Synth 8-327] inferring latch for variable 'NSTATE_reg' [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:64]

State	New Encoding	Previous Encoding
default1	000000001	0000
default2	000000010	0001
modop	000000100	0010
init	000001000	0011
a1	000010000	0100

```
          a2 |              000100000 |                 0101
          a3 |              001000000 |                 0110
     notprime |              010000000 |                 1000
     isprime |              100000000 |                 0111
---------------------------------------------------------------------------------------------
```

INFO: [Synth 8-3354] encoded FSM with state register 'STATE_reg' using encoding 'one-hot' in module 'Pnc_FSM'

WARNING: [Synth 8-6014] Unused sequential element NSTATE_reg was removed. [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:64]

WARNING: [Synth 8-6014] Unused sequential element STATE_reg was removed. [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:62]

WARNING: [Synth 8-327] inferring latch for variable 'FSM_onehot_NSTATE_reg' [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:64]

WARNING: [Synth 8-327] inferring latch for variable 'PrimeLed_reg' [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:80]

WARNING: [Synth 8-327] inferring latch for variable 'GuessResult_reg' [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:81]

WARNING: [Synth 8-327] inferring latch for variable 'Led4_reg' [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:98]

WARNING: [Synth 8-327] inferring latch for variable 'D_reg' [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:103]

WARNING: [Synth 8-327] inferring latch for variable 'R_reg' [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:131]

Finished RTL Optimization Phase 2 : Time (s): cpu = 00:00:20 ; elapsed = 00:00:24 . Memory (MB): peak = 656.285 ; gain = 426.043

Report RTL Partitions:

```
+-+--------------+------------+----------+
| |RTL Partition |Replication |Instances |
+-+--------------+------------+----------+
+-+--------------+------------+----------+
```

Start RTL Component Statistics

Detailed RTL Component Info :

+---Adders :

3 Input 32 Bit Adders := 1

2 Input 31 Bit Adders := 1

+---Muxes :

9 Input 32 Bit Muxes := 1

2 Input 32 Bit Muxes := 2

2 Input 31 Bit Muxes := 1

22 Input 9 Bit Muxes := 1

9 Input 4 Bit Muxes := 2

5 Input 1 Bit Muxes := 1

2 Input 1 Bit Muxes := 13

9 Input 1 Bit Muxes := 8

Finished RTL Component Statistics

Start RTL Hierarchical Component Statistics

Hierarchical RTL Component report

Module Pnc_FSM

Detailed RTL Component Info :

+---Adders :

3 Input 32 Bit Adders := 1

2 Input 31 Bit Adders := 1

+---Muxes :

9 Input 32 Bit Muxes := 1

2 Input 32 Bit Muxes := 2

2 Input 31 Bit Muxes := 1

22 Input 9 Bit Muxes := 1

9 Input 4 Bit Muxes := 2

5 Input 1 Bit Muxes := 1

2 Input 1 Bit Muxes := 13

9 Input 1 Bit Muxes := 8

Finished RTL Hierarchical Component Statistics

Start Part Resource Summary

Part Resources:

DSPs: 220 (col length:60)

BRAMs: 280 (col length: RAMB18 60 RAMB36 30)

Finished Part Resource Summary

Start Cross Boundary and Area Optimization

INFO: [Synth 8-5545] ROM "NSTATE" won't be mapped to RAM because address size (32) is larger than maximum supported(25)

INFO: [Synth 8-5545] ROM "Led4" won't be mapped to RAM because address size (32) is larger than maximum supported(25)

INFO: [Synth 8-5545] ROM "D" won't be mapped to RAM because address size (32) is larger than maximum supported(25)

INFO: [Synth 8-5545] ROM "R1" won't be mapped to RAM because address size (32) is larger than maximum supported(25)

INFO: [Synth 8-3886] merging instance 'D_reg[27]' (LD) to 'D_reg[31]'

INFO: [Synth 8-3886] merging instance 'D_reg[28]' (LD) to 'D_reg[31]'

INFO: [Synth 8-3886] merging instance 'D_reg[29]' (LD) to 'D_reg[31]'

INFO: [Synth 8-3886] merging instance 'D_reg[30]' (LD) to 'D_reg[31]'

INFO: [Synth 8-3886] merging instance 'D_reg[31]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[3]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[4]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[5]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[6]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[7]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[8]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[9]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[10]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[11]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[12]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[13]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[14]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[15]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[16]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[17]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[18]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[19]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[20]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[21]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[22]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[23]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[24]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3886] merging instance 'D_reg[25]' (LD) to 'D_reg[26]'

INFO: [Synth 8-3333] propagating constant 0 across sequential element (\D_reg[26])

INFO: [Synth 8-3333] propagating constant 0 across sequential element (\Led4_reg[3])

INFO: [Synth 8-3333] propagating constant 0 across sequential element (\R_reg[31])

WARNING: [Synth 8-3332] Sequential element (Led4_reg[3]) is unused and will be removed from module Pnc_FSM.

WARNING: [Synth 8-3332] Sequential element (D_reg[26]) is unused and will be removed from module Pnc_FSM.

WARNING: [Synth 8-3332] Sequential element (R_reg[31]) is unused and will be removed from module Pnc_FSM.

Finished Cross Boundary and Area Optimization : Time (s): cpu = 00:00:22 ; elapsed = 00:00:27 . Memory (MB): peak = 656.285 ; gain = 426.043

Report RTL Partitions:

+-+--------------+------------+----------+

| |RTL Partition |Replication |Instances |

+-+--------------+------------+----------+

+-+--------------+------------+----------+

Start Applying XDC Timing Constraints

Finished Applying XDC Timing Constraints : Time (s): cpu = 00:00:36 ; elapsed = 00:00:42 . Memory (MB): peak = 656.285 ; gain = 426.043

Start Timing Optimization

Finished Timing Optimization : Time (s): cpu = 00:00:36 ; elapsed = 00:00:42 . Memory (MB): peak = 656.285 ; gain = 426.043

Report RTL Partitions:

+-+--------------+------------+----------+

| |RTL Partition |Replication |Instances |

+-+--------------+------------+----------+

+-+--------------+------------+----------+

Start Technology Mapping

Finished Technology Mapping : Time (s): cpu = 00:00:36 ; elapsed = 00:00:42 . Memory (MB): peak = 660.863 ; gain = 430.621

Report RTL Partitions:

+-+--------------+------------+----------+

```
| |RTL Partition |Replication |Instances |
+-+--------------+------------+----------+
+-+--------------+------------+----------+
```

Start IO Insertion

Start Flattening Before IO Insertion

Finished Flattening Before IO Insertion

Start Final Netlist Cleanup

Finished Final Netlist Cleanup

WARNING: [Synth 8-5396] Clock pin G has keep related attribute (keep/mark_debug/dont_touch) which could create extra logic on its net [C:/Users/Nader/Downloads/FINALICD/FINALICD.srcs/sources_1/new/ANDGATE.vhd:131]

Finished IO Insertion : Time (s): cpu = 00:00:37 ; elapsed = 00:00:43 . Memory (MB): peak = 660.863 ; gain = 430.621

Report Check Netlist:

```
+------+-----------------+-------+---------+-------+-----------------+
|      |Item             |Errors |Warnings |Status |Description      |
+------+-----------------+-------+---------+-------+-----------------+
|1     |multi_driven_nets |     0|       0|Passed |Multi driven nets |
+------+-----------------+-------+---------+-------+-----------------+
```

Start Renaming Generated Instances

Finished Renaming Generated Instances : Time (s): cpu = 00:00:37 ; elapsed = 00:00:43 . Memory (MB): peak = 660.863 ; gain = 430.621

Report RTL Partitions:

+-+--------------+------------+----------+

| |RTL Partition |Replication |Instances |

+-+--------------+------------+----------+

+-+--------------+------------+----------+

Start Rebuilding User Hierarchy

Finished Rebuilding User Hierarchy : Time (s): cpu = 00:00:37 ; elapsed = 00:00:43 . Memory (MB): peak = 660.863 ; gain = 430.621

Start Renaming Generated Ports

Finished Renaming Generated Ports : Time (s): cpu = 00:00:37 ; elapsed = 00:00:43 . Memory (MB): peak = 660.863 ; gain = 430.621

Start Handling Custom Attributes

Finished Handling Custom Attributes : Time (s): cpu = 00:00:37 ; elapsed = 00:00:43 . Memory (MB): peak = 660.863 ; gain = 430.621

Start Renaming Generated Nets

Finished Renaming Generated Nets : Time (s): cpu = 00:00:37 ; elapsed = 00:00:43 . Memory (MB): peak = 660.863 ; gain = 430.621

Start Writing Synthesis Report

Report BlackBoxes:

+-+--------------+----------+

| |BlackBox name |Instances |

+-+--------------+----------+

+-+--------------+----------+

Report Cell Usage:

+------+-------+------+

| |Cell |Count |

+------+-------+------+

|1 |BUFG | 1|

|2 |CARRY4 | 64|

|3 |LUT1 | 12|

|4 |LUT2 | 82|

|5 |LUT3 | 34|

|6 |LUT4 | 157|

|7 |LUT5 | 24|

|8 |LUT6 | 63|

|9 |FDCE | 8|

|10 |FDPE | 1|

|11 |LD | 49|

|12 |IBUF | 9|

|13 |OBUF | 7|

+------+-------+------+

Report Instance Areas:

+------+---------+-------+------+

| |Instance |Module |Cells |

+------+---------+-------+------+

|1 |top | | 511|

+------+---------+-------+------+

Finished Writing Synthesis Report : Time (s): cpu = 00:00:37 ; elapsed = 00:00:43 . Memory (MB): peak = 660.863 ; gain = 430.621

Synthesis finished with 0 errors, 0 critical warnings and 19 warnings.

Synthesis Optimization Runtime : Time (s): cpu = 00:00:23 ; elapsed = 00:00:31 . Memory (MB): peak = 660.863 ; gain = 121.203

Synthesis Optimization Complete : Time (s): cpu = 00:00:37 ; elapsed = 00:00:43 . Memory (MB): peak = 660.863 ; gain = 430.621

INFO: [Project 1-571] Translating synthesized netlist

INFO: [Netlist 29-17] Analyzing 122 Unisim elements for replacement

INFO: [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds

INFO: [Project 1-570] Preparing netlist for logic optimization

INFO: [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).

INFO: [Project 1-111] Unisim Transformation Summary:

A total of 49 instances were transformed.

LD => LDCE: 49 instances

65 Infos, 22 Warnings, 0 Critical Warnings and 0 Errors encountered.

synth_design completed successfully

synth_design: Time (s): cpu = 00:00:38 ; elapsed = 00:00:47 . Memory (MB): peak = 662.281 ; gain = 439.809

INFO: [Common 17-1381] The checkpoint 'C:/Users/Nader/Downloads/FINALICD/FINALICD.runs/synth_1/Pnc_FSM.dcp' has been generated.

report_utilization: Time (s): cpu = 00:00:00 ; elapsed = 00:00:00.043 . Memory (MB): peak = 662.281 ; gain = 0.000

INFO: [Common 17-206] Exiting Vivado at Mon Mar 29 21:02:08 2021...

Appendix – VHDL State machine code

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 10.02.2021 14:44:29
-- Design Name:
-- Module Name: ANDGATE - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
```

```
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Pnc_FSM is
   Port (
        CLK : in STD_LOGIC;
        Input : in STD_LOGIC_VECTOR(7 downto 0);
        UserGuess: out STD_LOGIC;
        PrimeLed:  out STD_LOGIC;
        GuessResult : out STD_LOGIC;
        Led4 : out STD_logic_vector(3 downto 0)
        );
end Pnc_FSM;

architecture FSM of Pnc_FSM is
type state_type is (DEFAULT1,DEFAULT2, ModOp, INIT ,A1, A2, A3, isPrime, notPrime); --states associated with the finite state machine
SIGNAL STATE : STATE_TYPE;
SIGNAL NSTATE : STATE_TYPE;
SIGNAL TEMP : INTEGER;
SIGNAL R : INTEGER;
SIGNAL D : INTEGER := 2;
begin

    SYNC_PROC : PROCESS(CLK, Input(7)) --process that synchronises the state machine process
     BEGIN

        IF (Input(7) = '1') THEN  --checks if RESET is HIGH
```

```
        STATE <= DEFAULT1;
      ELSIF (CLK'EVENT AND CLK = '1') THEN  --sets current state to next state value
        STATE <= NSTATE;
      END IF;



  END PROCESS SYNC_PROC;

  STATE_PROC: PROCESS(STATE, Input)
   BEGIN

    TEMP <= to_integer(unsigned(Input(5 downto 0))); --converts binary input to integer for mod
operation

    CASE STATE IS

    WHEN DEFAULT1 =>  --initial state when reset is set HIGH
        NSTATE <= DEFAULT2;
        PrimeLed <= '1';
        GuessResult <= '1';
        if(input(6) = '1') then -- Users prime guess is evaluated
         UserGuess <= '1';
        end if;
        if(input(6) = '0') then
         UserGuess <= '0';
        end if;

    WHEN DEFAULT2 =>   --checks for error input cases
        if(input(6) = '1') then
         UserGuess <= '1';
```

```
end if;

if(input(6) = '0') then
 UserGuess <= '0';
end if;

Led4 <= "0000";
PrimeLed <= '0';
GuessResult <= '0';

NSTATE <= ModOp;
D <= 2;

if (TEMP = 1 or TEMP = 0) then
 NSTATE <= notPrime;
 Led4 <= "0000";
END IF;

if (TEMP = 2) then
 NSTATE <= isPrime;
 Led4 <= "0000";
END IF;

if (TEMP = 3) then
 NSTATE <= isPrime;
 Led4 <= "0000";
END IF;

if (TEMP = 5) then
 NSTATE <= isPrime;
 Led4 <= "0000";
```

```
    END IF;

    if (TEMP = 7) then
     NSTATE <= isPrime;
     Led4 <= "0000";
    END IF;

WHEN ModOp  =>  --mod operation of input TEMP is completed
    R <= TEMP mod D;
    if(D <= 2) then
     NSTATE <= INIT;
    elsif(D <= 3) then
     NSTATE <= A1;
    elsif(D <= 5) then
     NSTATE <= A2;
    elsif(D <= 7) then
     NSTATE <= A3;
    END IF;

WHEN INIT => --evaluates case for divisor = 2

    if (R = 0) then
     Led4 <= "0010";
     NSTATE <= notPrime;
    ELSIF (R /= 0) then
     D <= 3;
     NSTATE <= ModOp;
    END IF;
```

```
WHEN A1 => --evaluates case for divisor = 3
    IF (R = 0) then
     Led4 <= "0011";
     NSTATE <= notPrime;
    ELSIF (R /= 0) then
     D <= 5;
     NSTATE <= ModOp;
    END IF;

WHEN A2 => --evaluates case for divisor = 5
    if (R = 0) then
     Led4 <= "0101";
     NSTATE <= notPrime;
    ELSIF (R /= 0) then
     D <= 7;
     NSTATE <= ModOp;
    END IF;

WHEN A3 => --evaluates case for divisor = 7
    IF (R = 0) then
     Led4 <= "0111";
     NSTATE <= notPrime;
    ELSIF (R /= 0) then
     NSTATE <= isPrime;
    END IF;
```

```
        WHEN isPrime => --prime number successfully
            if(Input(6) = '1') then --correct guess
             PrimeLed <= '1';
             GuessResult <= '1';
            else                --incorrect guess
             GuessResult <= '0';
             PrimeLed <= '1';
            end if;


        WHEN notPrime => --not prime successfully detected
            if(Input(6) = '1') then --incorrect guess
             GuessResult <= '0';
             PrimeLed <= '0';
            else                --correct guess
             Primeled <= '0';
             GuessResult <= '1';
            end if;

        WHEN others =>

        END CASE;


    END PROCESS STATE_PROC;

END FSM;
```

Appendix – VHDL Test Bench Code

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 06.03.2021 17:50:15
-- Design Name:
-- Module Name: PNC_TB - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
```

```
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity PNC_TB is
end PNC_TB;

architecture Behavioral of PNC_TB is

component Pnc_FSM is
   Port (
        CLK : in STD_LOGIC;
        Input : in STD_LOGIC_VECTOR(7 downto 0);
        UserGuess: out STD_LOGIC;
        PrimeLed:  out STD_LOGIC;
        GuessResult : out STD_LOGIC;
        Led4 : out STD_logic_vector(3 downto 0)
        );
end component;
signal CLK, UserGuess, PrimeLed, GuessResult: std_logic;
signal Input : std_logic_vector(7 downto 0);
signal Led4 : std_logic_vector(3 downto 0);

begin
UUT : Pnc_FSM port map(CLK => CLK, Input => Input, UserGuess => UserGuess, PrimeLed =>
PrimeLed, GuessResult => GuessResult, Led4 => Led4);
clk_proc: process --process simulating the clock cycle
begin
 CLK <= '0';
 wait for 50 ns;
 CLK <= '1';
```

```
 wait for 50 ns;
end process clk_proc;

sim_proc: process --process simulating resets and different input combinations
begin

Input <= "10000000"; wait for 2000ns; --reset HIGH
Input <= "00001101"; wait for 2000ns; --input number 13 with user guessing not prime
Input <= "10000000"; wait for 2000ns; --reset HIGH
Input <= "00110001"; wait for 2000ns; --input number 49 with user guessing not prime
Input <= "10000000"; wait for 2000ns; --reset HIGH
Input <= "00001111"; wait for 2000ns; --input number 15 with user guessing not prime
Input <= "10000000"; wait for 2000ns; --reset HIGH
Input <= "01010000"; wait for 2000ns; --input number 16 with user guessing prime
Input <= "10000000"; wait for 2000ns; --reset HIGH
Input <= "00010011"; wait for 2000ns; --input number 19 with user guessing not prime


end process sim_proc;
end Behavioral;
```

Appendix Master Constraint File

```
## Clock Signal

set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 } [get_ports { CLK }];
#IO_L12P_T1_MRCC_34 Sch=sysclk_125mhz


## LEDs

set_property -dict { PACKAGE_PIN Y9    IOSTANDARD LVCMOS33 } [get_ports {UserGuess}];
#IO_L12P_T1_MRCC_13 Sch=led[0]

set_property -dict { PACKAGE_PIN Y8    IOSTANDARD LVCMOS33 } [get_ports { PrimeLed }];
#IO_L12N_T1_MRCC_13 Sch=led[1]

set_property -dict { PACKAGE_PIN V7    IOSTANDARD LVCMOS33 } [get_ports { GuessResult }];
#IO_L23P_T3_13 Sch=led[2]

set_property -dict { PACKAGE_PIN W7    IOSTANDARD LVCMOS33 } [get_ports { Led4[0] }];
#IO_L23N_T3_13 Sch=led[3]

set_property -dict { PACKAGE_PIN V10   IOSTANDARD LVCMOS33 } [get_ports { Led4[1] }];
#IO_L1P_T0_13 Sch=led[4]

set_property -dict { PACKAGE_PIN W12   IOSTANDARD LVCMOS33 } [get_ports { Led4[2] }];
#IO_L4N_T0_13 Sch=led[5]

set_property -dict { PACKAGE_PIN W11   IOSTANDARD LVCMOS33 } [get_ports { Led4[3] }];
#IO_L3P_T0_DQS_13 Sch=led[6]


## Elvis Port

set_property -dict { PACKAGE_PIN Y20   IOSTANDARD LVCMOS33 } [get_ports { Input[0]  }];
#IO_L9P_T1_DQS_33 Sch=pbc_dio[0]

set_property -dict { PACKAGE_PIN AA16  IOSTANDARD LVCMOS33 } [get_ports { Input[1]  }];
#IO_L18P_T2_33 Sch=pbc_dio[1]

set_property -dict { PACKAGE_PIN Y19   IOSTANDARD LVCMOS33 } [get_ports { Input[2]  }];
#IO_L11P_T1_SRCC_33 Sch=pbc_dio[2]

set_property -dict { PACKAGE_PIN AB16  IOSTANDARD LVCMOS33 } [get_ports { Input[3]  }];
#IO_L18N_T2_33 Sch=pbc_dio[3]

set_property -dict { PACKAGE_PIN AA18  IOSTANDARD LVCMOS33 } [get_ports { Input[4]  }];
#IO_L12N_T1_MRCC_33 Sch=pbc_dio[4]

set_property -dict { PACKAGE_PIN AB15  IOSTANDARD LVCMOS33 } [get_ports { Input[5]  }];
#IO_L24N_T3_33 Sch=pbc_dio[5]
```

```
set_property -dict { PACKAGE_PIN Y18   IOSTANDARD LVCMOS33 } [get_ports { Input[6]  }];
#IO_L12P_T1_MRCC_33 Sch=pbc_dio[6]

set_property -dict { PACKAGE_PIN AA14  IOSTANDARD LVCMOS33 } [get_ports { Input[7]  }];
#IO_L22N_T3_33 Sch=pbc_dio[7]
```