Random Number Generation Math Game

ECE 271 Group Project

Edgar Solis

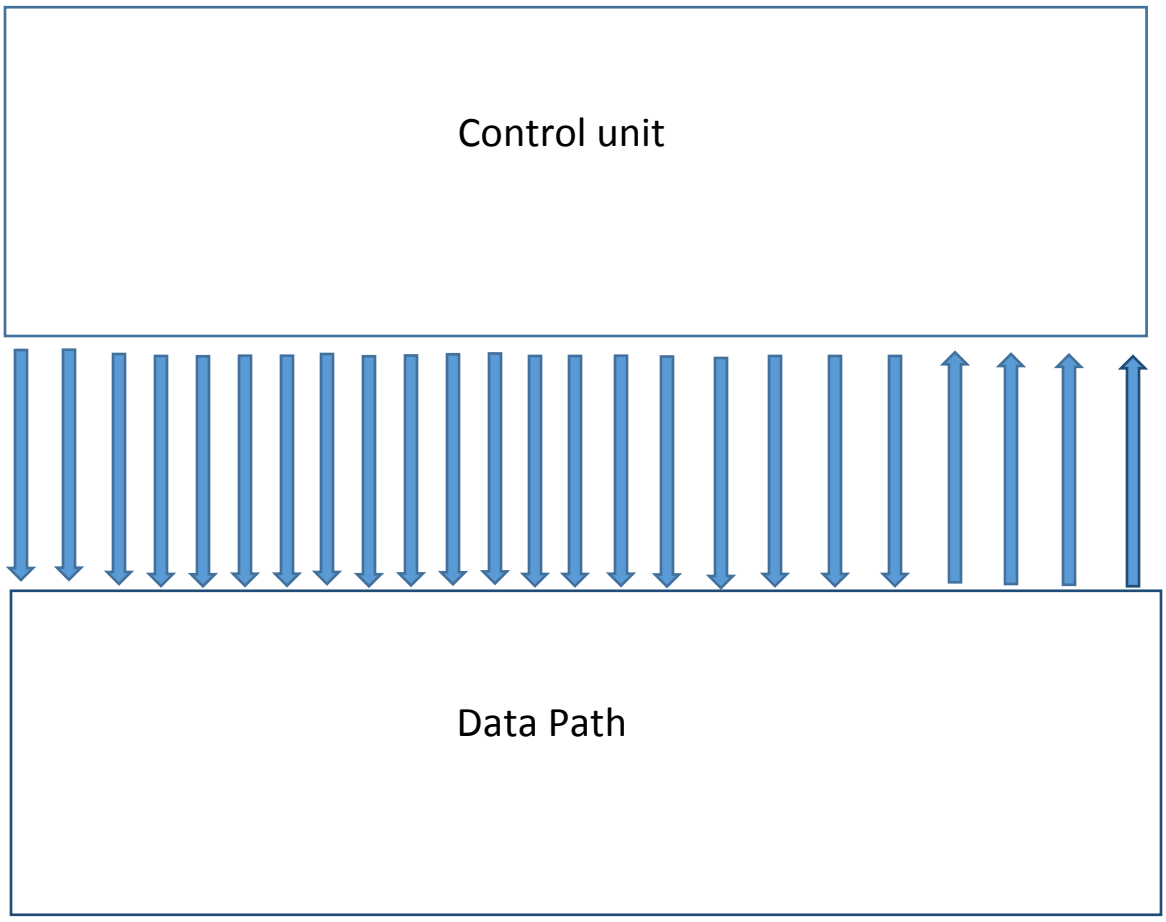
Jeffrey Clark

Justin Ross

December 9, 2016

1. Problem Specification

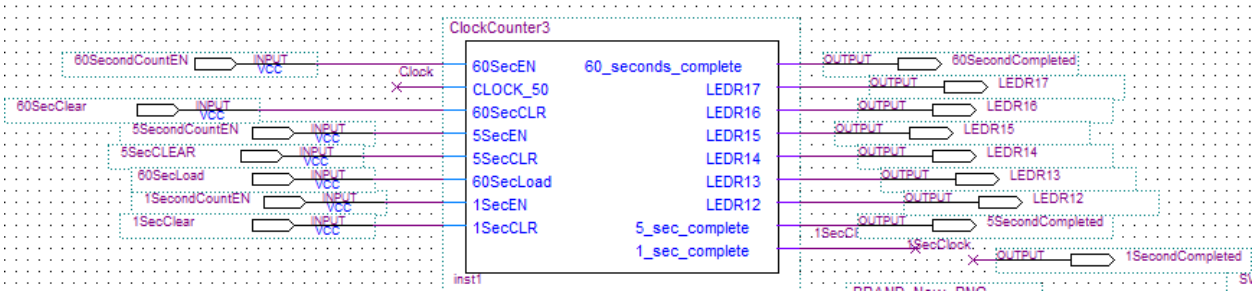
1) Break the design into a control unit and data path. Determine which of the primary inputs and outputs are fed into and read out from the Control Unit and which ones are fed into or read from the data path (this includes the RST and CLK inputs). Draw lower hierarchical level block diagram showing the inputs and outputs of the Control Unit and Data Path and justify each choice. You can leave the control and status connections between the Control Unit and Data Path unspecified for now.



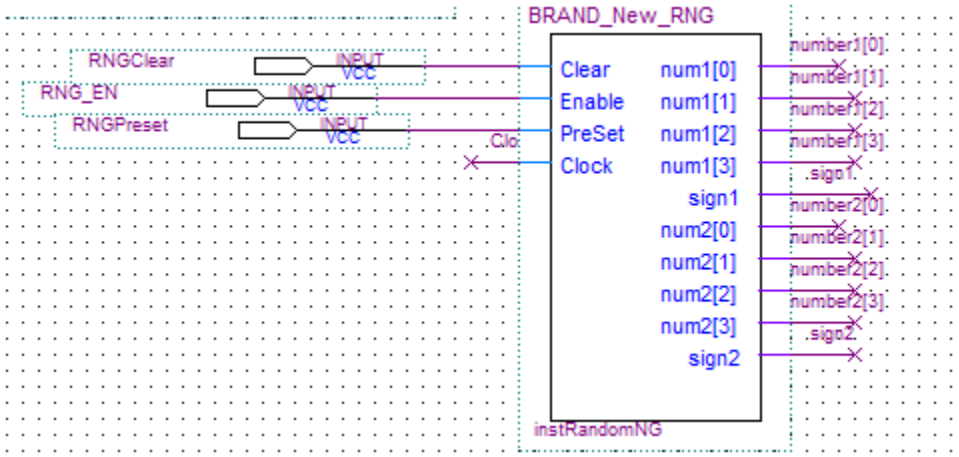
2) Determine the functionality of the Data Path and describe it.
The functionality of our data path is that the user has 60 seconds then the game operates by presenting two random single digit numbers, which could be positive, negative or zero. This number will be displayed on the seven segment displays, which are HEX7-HEX4. HEX 7 and HEX5 will hold the sign value for these numbers, and HEX6-HEX4 will hold the number value. Then these two numbers must be added by the adder and use SW5 and SW0, then the 2's complement must be taken and then press KEY 3 to indicate their answer is ready. After this a comparator will compare the users answer and if it is correct one point should be added to their score, the correct answer should be displayed in decimal value on HEX. Then the correct answer must be displayed in binary in LEDR, and then the letter "Y" displayed on HEX0 for 1 sec. Just like the correct answer the wrong answer will compare the users wrong answer with the correct

answer and do the same but instead will display a “n” if its incorrect on the HEX0 for 1 sec. There will also be a clock that will clock cycle that counts to 5 sec and if the user does not press KEY3 to submit their answer will be marked wrong and nothing will be added to their score. Then the correct answer will be displayed in binary by the LEDR lights. This procedure should keep repeating until the 60 seconds have been completed. The counting down of the 60 seconds will be shown in binary on the LEDR17-12. The binary score should be shown on LEDG4-LEDG0. Once the game is over all the LEDs should be cleared and the decimal value of score should be displayed on HEX1 and HEX0. Lastly another comparators will compare your score and if the score is less than 6 the display will read “uh-oh”, if the score is less than 13 but greater than or equal to 6 it will display “so-so”, if the score is less than 20 but greater than or equal to 13 the display will show “good”, Lastly if the score is greater than or = equal to 20 the display will show “yeah”.

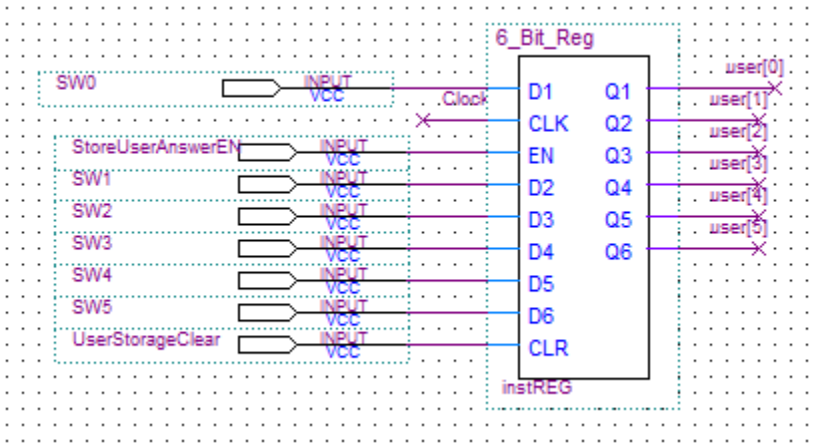
3) State and describe the different sub-modules of the design (e.g. 2’s complementor, 7-segment display controller, time counting module etc.). Show where each of these is determined from the problem specification.



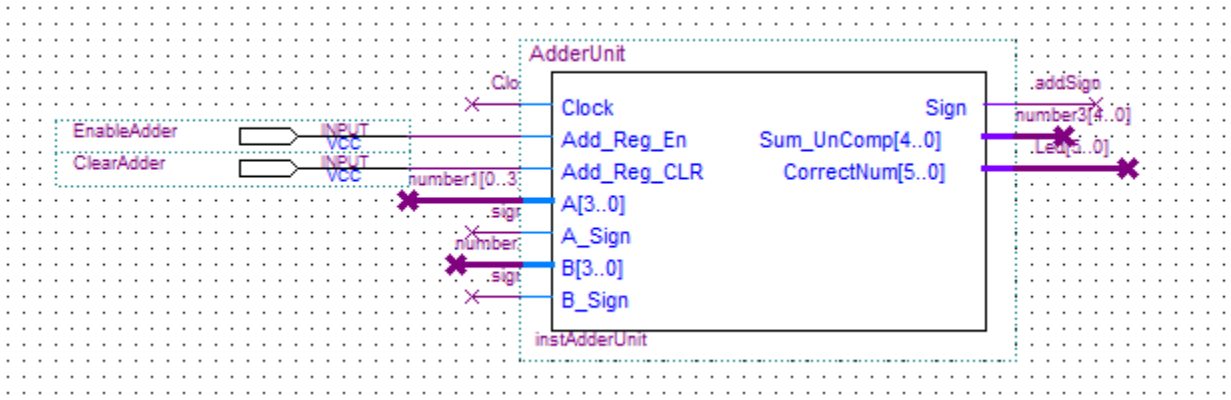
This is a Clock Counter, this is a collection of counters that will be activated when 60 seconds, 5 seconds, and 1 second all pass. The 60 seconds will determine when the game ends, the 5 seconds will determine if the player waited too long to answer the question, and 1 second will be how long the incorrect or correct screen is on. The LEDs are there to show how much time is left. They are hooked up to a count-down counter.



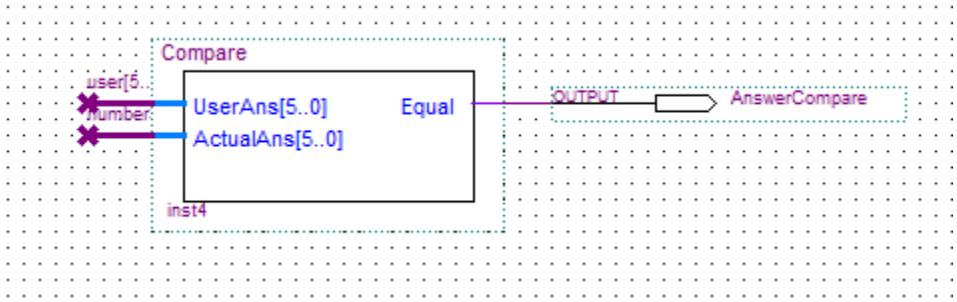
This is the random number generator; it will produce the numbers that are being added together.



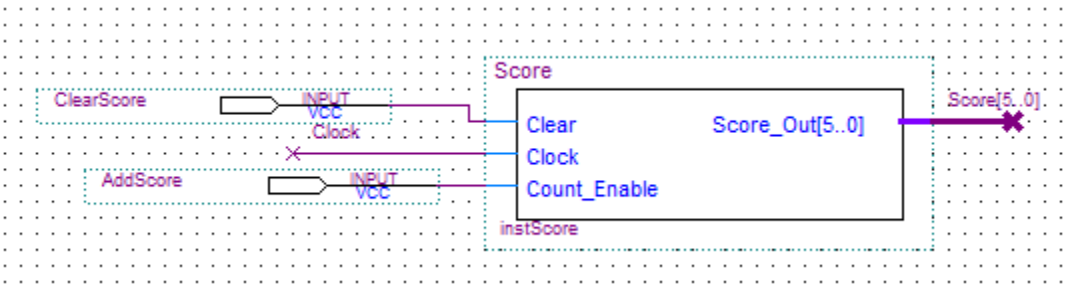
This 6-Bit Register is used to store the answer that the user puts in. The value stored will be used to compare the users answer to the actual answer. When it's the same the user's score will go up.



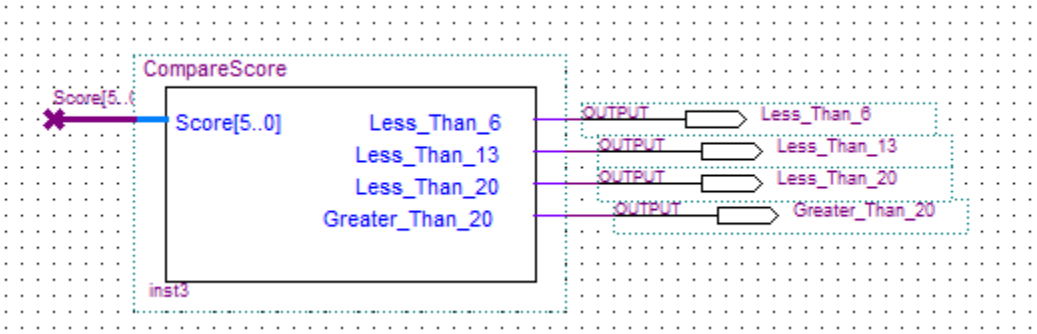
The Adder Unit will be used to add the two rand numbers together. This will produce a 2's complemented answer that will be used for comparison, and an uncomplemented answer that will be used in the display of the correct answer.



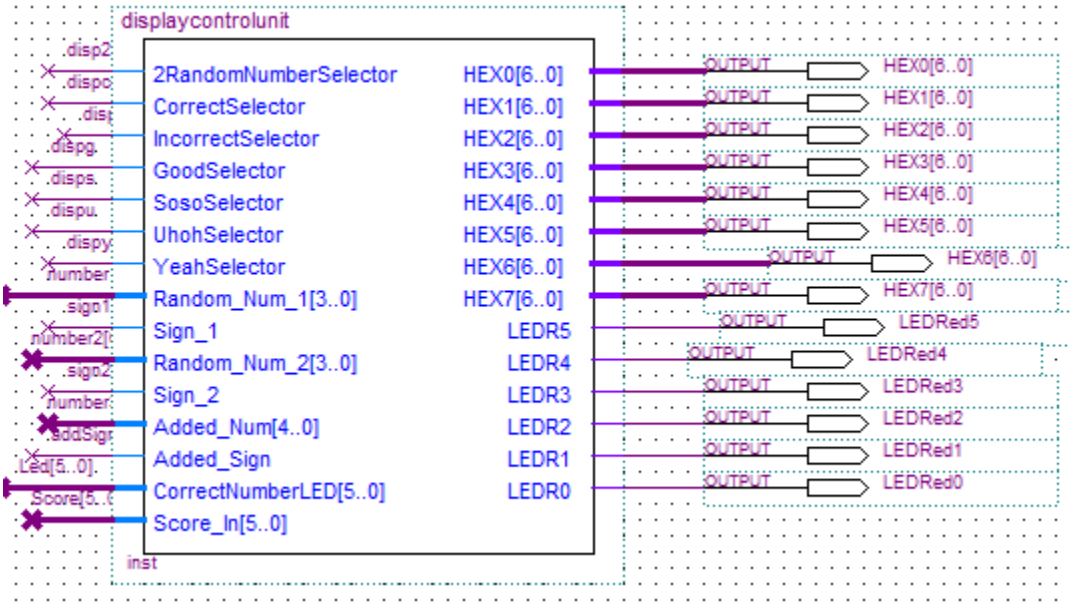
The comparator will compare the user's answer against the actual answer from the adder unit. If they're the same the output will go high.



The Score module is a counter that will only count if the enable is high. There is a state dedicated to adding the score that only lasts one clock cycle, so it will only add one.



This Compare Score component will take the score and compare it against 6, 13, and 20. These will determine which end game display will be shown. if the score is less than 6 the display will read “uh-oh”, if the score is less than 13 but greater than or equal to 6 it will display “so-so”, if the score is less than 20 but greater than or equal to 13 the display will show “good”, Lastly if the score is greater than or = equal to 20 the display will show “yeah”.



The display control unit is a giant unit that is a collection of multiplexors. It takes in 8 different selector bits that will determine which seven segment display will show. It will also show the correct binary expression of the answer when the correct or incorrect display are on. So, for example, if 2RandomNumberSelector is high, it will show the two random numbers and the rest of the displays are blank.

2. REQUIREMENT SPECIFICATION:

Show how you determined what components are necessary to realize the design

Answer the following question in this section:

1) Determine what components (e.g. counters, registers, muxes, decoders etc.) will comprise the Data Path and justify each choice referring to the Problem Specification.

Answer:

1. Frequency Divider - The intent was to take the 50 MHz clock and divide down to a 1 Hz clock which would give us the 1 second that we need for our 3 counters in order to tell the state machine when to change states. This was tested in multiple different ways and it did in fact put out 1 sec.

2. Counters - 60 sec. 5 sec, and 1 sec. counters where used to track the time elapsed for the game, the answer time allowed, and when 1 sec has elapsed.

3. Registers - we needed for a lot of things. (to hold the answer from the user, also needed one to hold the values coming from the random number generator, in the adder to hold the answer

4. Muxes - These were used on all of our display functions (the random numbers, the correct number of answers, the incorrect and correct answers, as well as displaying the (yeah, good soso, and uhoh,).

5. Decade counter - 2 were used in the creation of the random number generator this allows the user to get only numbers from -9 to 9.

6. SSD - These were used on all of our display functions (the random numbers, the correct number of answers, as well as displaying the (yeah, good soso, and uhoh) words for how many total correct answers there were.

7. Adders – to add the numbers generated by the LFSR

8. Comparators – to compare the users answer against the stored answer we used 4, one for score less than 6, one for a score 6<13, one for 13<20, and one for 20 and above

9. 2's complimentor – used in the adder unit in order to add negative numbers appropriately, we also used a reverse complimentor to get the correct number out of the adder unit.

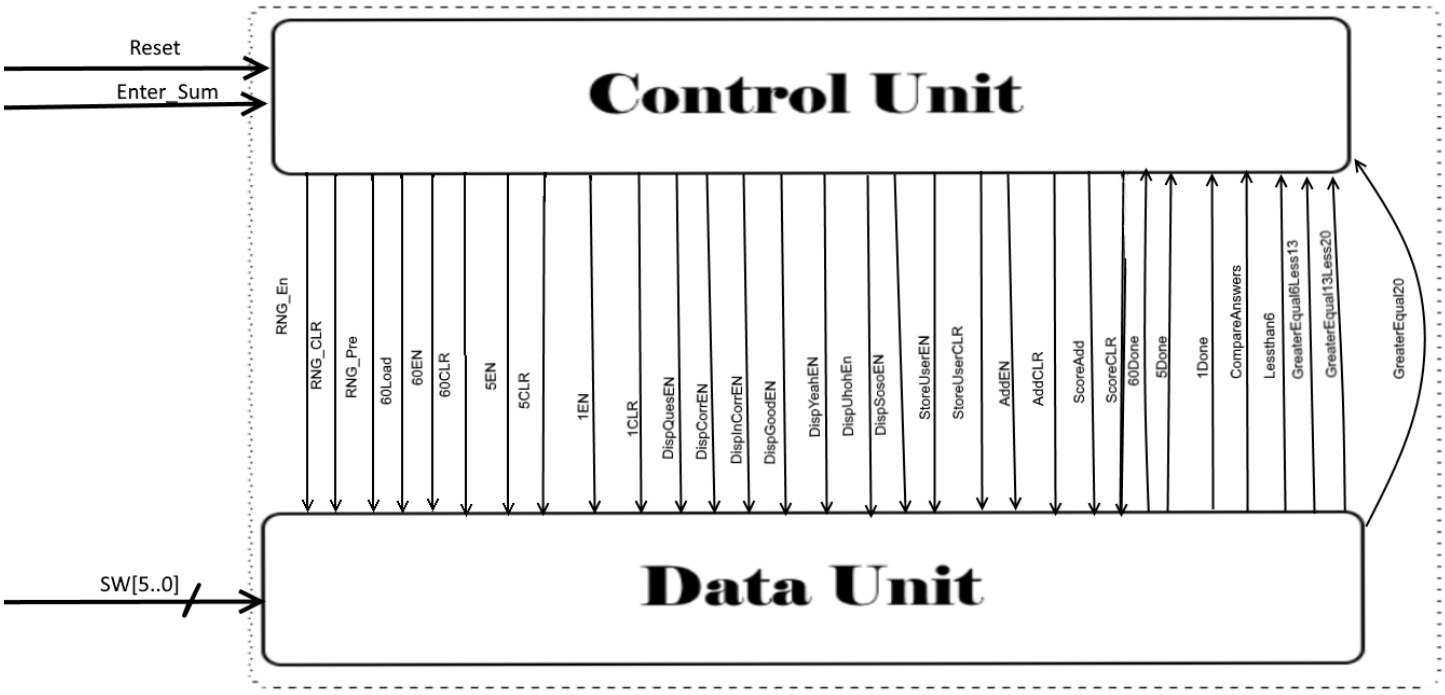
10. State Machine – Was used to control the entire game which consisted of the states(Not Ready, Play, Add, New Number, Correct, Incorrect, Game Over, and Add Score).

3. Design Specification

Describe you design in terms of your:

a) Data Unit:

i) Determine which control signals are required from the Control Unit to the Data Path and which status signals will be output from the Data Path. Justify them



Signals from the Control Unit to the Data Unit

RNG_En- Will come from the Control Unit to the Data Unit, this will enable the register in the Random Number Generator to save, this stored value will be the numbers displayed and then added.

RNG_CLR- This is the signal from the Control Unit to clear the Random Number Generator's register.

RNG_Pre- The LSFR is composed of D-Flip Flops, those can take a high or low preset, each of the flip flops are on the same input. In order to generate Random Numbers, the preset must be high.

60Load- In this design, there was a sixty second count down on LED's. This signal from the Control Unit's Reset State will enable the load on a count-down counter to be enabled. The binary representation of 60 will be loaded into that counter and the count-down Leds should display.

60EN- This signal will enable the sixty second count. There are two counters, one will count up to sixty and when it's count is done, it will be used as a status signal from the Data Path to the Control Unit. The 60EN signal is also used in order to activate the previously mentioned count-down counter.

60CLR- This signal will reset the two counters that have been mentioned previously.

5EN- This will enable the 5 second clock counter to begin counting.

5CLR- This will clear the 5 second clock counter.

1EN- This will enable the 1 second clock counter to enable.

1CLR – This will clear the 1 second clock counter.

DispQuesEN- This control signal will be sent to the Display Control Unit in the Data Unit to enable the two random numbers to be displayed on the Hex's.

DispCorrEN- When this signal is sent from the Control Unit to the Data Unit, the two random numbers, the sum of those numbers, and the letter "y" to be displayed on the Hex's.

DispInCorrEN- This control signal is sent to the Data Unit, the two random numbers, the sum of those numbers, and the letter "n" to be displayed on the Hex's.

DispGoodEN- When this control signal is high, the Display Control Unit will select the Hex's to display the score that the user earned, and the word "good". This signal should only go high when the score is greater than or equal to 13 but still less than 20.

DispYeahEN- When this control signal is high, the Display Control Unit will select the Hex's to display the score that the user earned, and the word "yeah". This signal should only go high when the score is greater than or equal 20.

DispUhohEN- When this control signal is high, the Display Control Unit will select the Hex's to display the score that the user earned, and the word "uhoh". This signal should only go high when the score is less than 6.

DispSosoEN- When this control signal is high, the Display Control Unit will select the Hex's to display the score that the user earned, and the word "soso". This signal should only go high when the score is greater than or equal to 6 but still less than 13.

StoreUserEN- This signal will enable the register that will store what the user inputs for the added number.

StoreUserCLR- When this signal goes to the data path, the register that stores the users answer will be cleared.

AddEN- This signal will enable the register after the adder to store the added value. This value will be used to compare against what the user inputs, this comparator will determine if the score increases or not.

AddCLR- When this signal is high, it will clear the register right after the adder.

ScoreAdd- This will signal the counter that keeps the score to begin counting for how ever long this signal is high. This was set in the State machine on a state that only lasts one clock cycle.

ScoreCLR- This signal from the Control Unit will clear the counter that keeps track of the score.

Signals from the Data Unit to the Control Unit

60Done- This signal will be sent to the Control Unit when the sixty second counter reaches 60 seconds. When this is high the State machine should go directly to the Game Over State.

5Done- This signal will be sent to the Control Unit when the five second counter reaches 5 seconds. When this is high the State machine should go directly to the Incorrect State.

1Done- This signal will be sent to the Control Unit when the one second counter reaches 1 second. When this is high the State machine should go from the Correct or Incorrect State to the NewNumber State which will generate new numbers.

CompareAnswers- This signal will determine if the added number and the user answer match. If they match, the signal will be high, and this will indicate the State Machine to go to the Correct State. When this signal is low, it will indicate that the State Machine go to the Incorrect State.

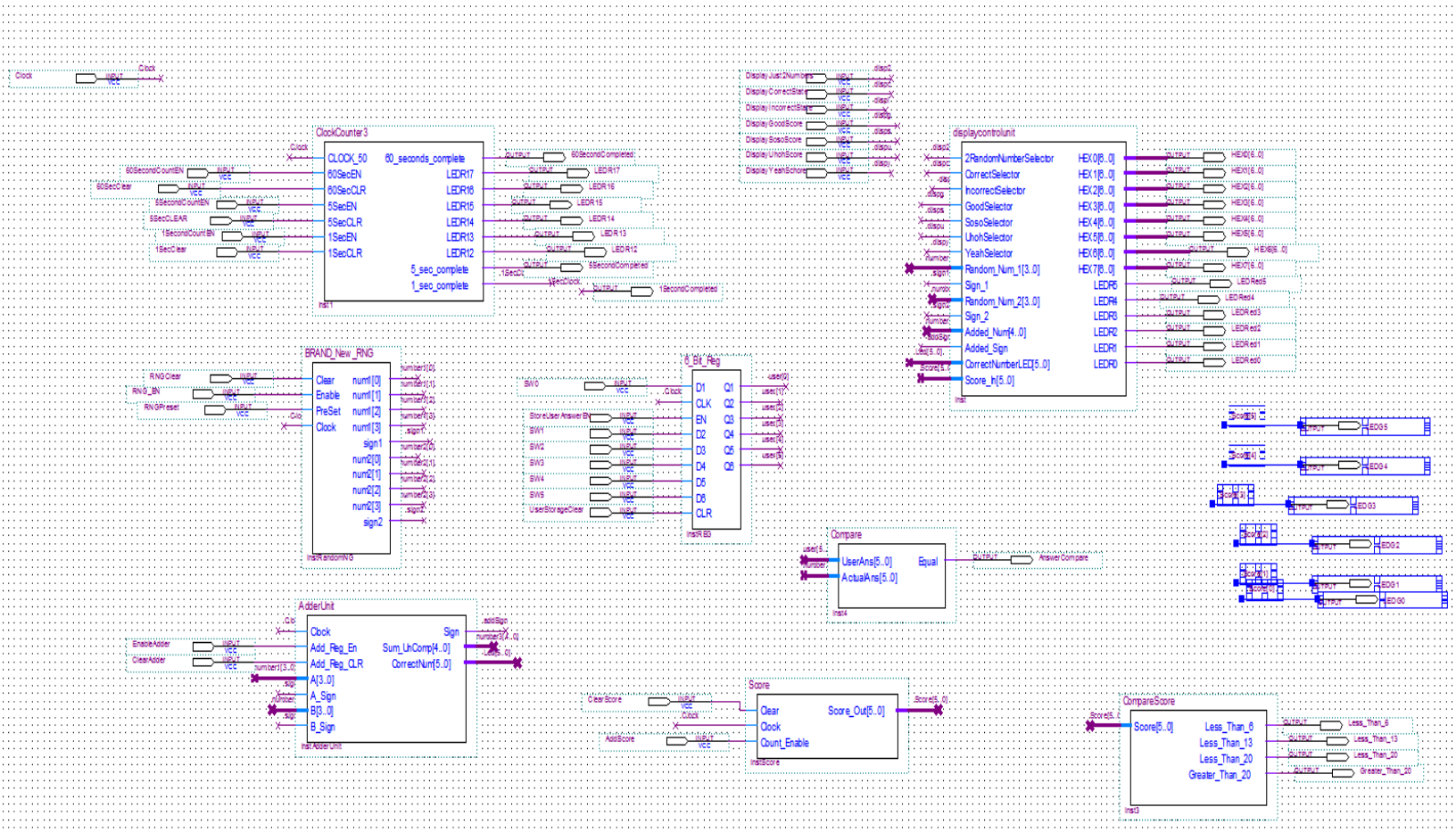
LessThan6- This will come from the score comparator, when the score is less than 6, this will go high. That will cause the Control Unit to select the respective Game Over display.

GreaterEqual6Less13- This will come from the score comparator, when the score is greater than or equal to six and still less than thirteen, this will go high. That will cause the Control Unit to select the respective Game Over display.

GreaterEqual13Less20- This will come from the score comparator, when the score is greater than or equal to thirteen and still less than twenty, this will go high. That will cause the Control Unit to select the respective Game Over display.

GreaterThan20- This will come from the score comparator, when the score is greater than 20, this will go high. That will cause the Control Unit to select the respective Game Over display.

ii) Draw a block diagram showing the internal components of the Data Path and the signals between them. Also show on this diagram any Control signals from the CU, status signals to the CU and Primary inputs and outputs of the Data Path



The Clock Counter takes the 60EN, 60CLR, 60LOAD, 5EN, ECLR, 1EN, 1CLR, and clock inputs, and then outputs 60Done, 5Done, 1Done, and then the countdown to some LED's. The inputs will tell the clock which counters to enable, load, or clear, and the outputs will go to the state machine.

The Random Number Generator (RNG) takes in the RNGEN, RNGPRE, RNGCLR signals that will enable the register, clear the register, or preset the LSFR respectively. It outputs two 4 bit random numbers each with a sign bit. Each of those numbers will go to the Adder Unit, as well as the Display Control Unit.

The Adder Unit will take in the AddEN, AddCLR, and the two random numbers and their sign bit from the RNG. The AddEN signal will enable the register after the adder, whereas the AddCLR signal will clear the register. This will output two 6 bit numbers, one is the 2's complemented added number, and the other is the un-2's complemented added number. The 2's complemented one will be displayed on LED's and it will also go to the Compare Unit. The Un 2's complemented number will go to the Display Control Unit.

The 6-Bit Register will take in StoreUserEN, StoreUserCLR, and whatever 6-bit number the user inputs using the switches. StoreUserEN will enable this register, and StoreUserCLR will clear this register. The output is the same number that was input when the enable was high, but it will go to the Compare Unit.

The Compare Unit takes in the previously mentioned complemented number and the users answer. It compares them and if they are equal, the AnswerCompare output goes high. This will indicate to the state machine which state it should choose, Correct or Incorrect.

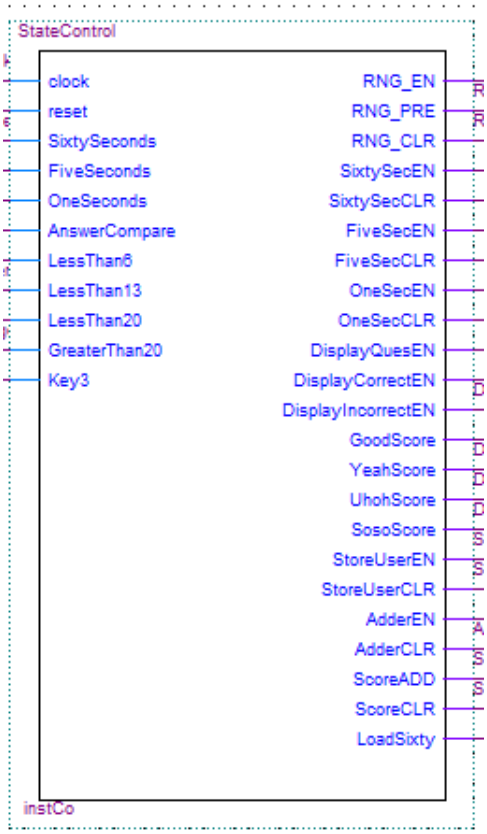
The Display Control Unit, this takes in the DispGoodEN, DispYeahEN, DispUhohEN, DispSosoEN, DispCorrEN, DispIncorrEN,, DispQuesEN, the two randomly generated numbers, and the uncomplemented added number. Each of the inputs from the Control Unit will select which of the Hex's will go high. The Random numbers and the uncomplemented added number is important because the two numbers that are being added together actually need to be displayed and the answer also needs to be displayed. This will output each of the 8 Hex's and it will output the LED's of the binary code for the correct answer.

The Score Unit will take in the ScoreAdd, ScoreCLR, and the clock inputs. ScoreAdd will enable the counter to count, whereas ScoreCLR will clear the score. The counter is set to count clock cycles, so a state was made that output ScoreAdd and that only lasted one clock cycle in order to get an accurate score. This unit output a 6 bit number representing the score.

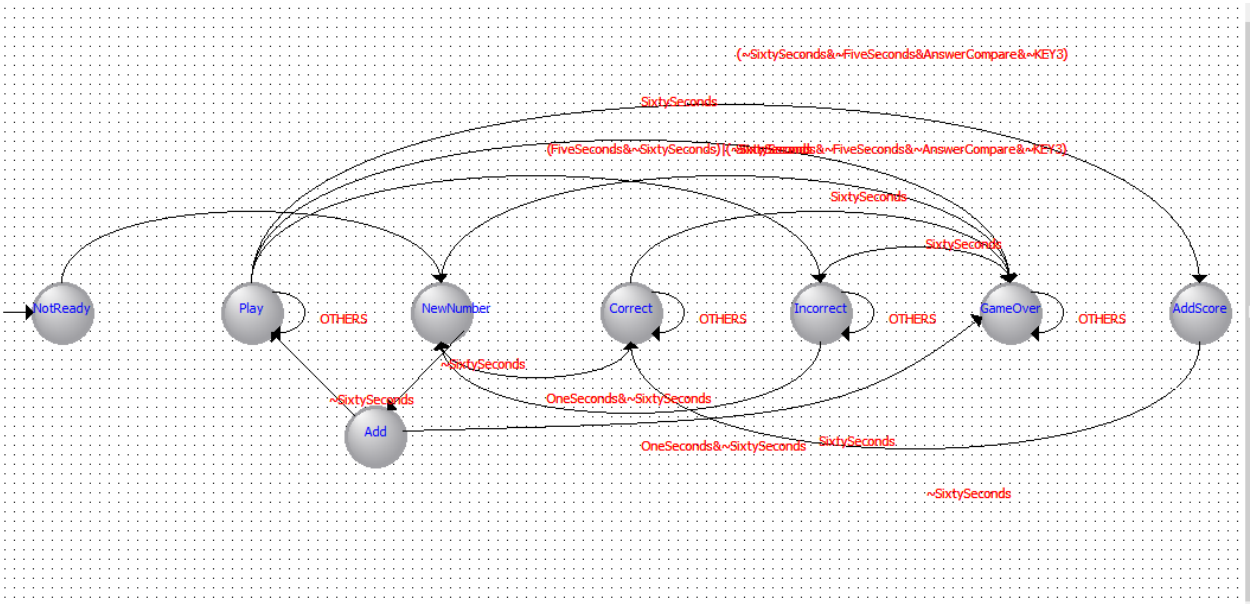
And the Score Comparator took in that 6 bit number and would output LessThan6, GreaterEqual6Less13, GreaterEqual13Less20, GreaterEqual20. It will compare the score and will choose which output to put high depending on how many points the user earned. These outputs are sent to the Control Unit to decide which Game Over display to choose

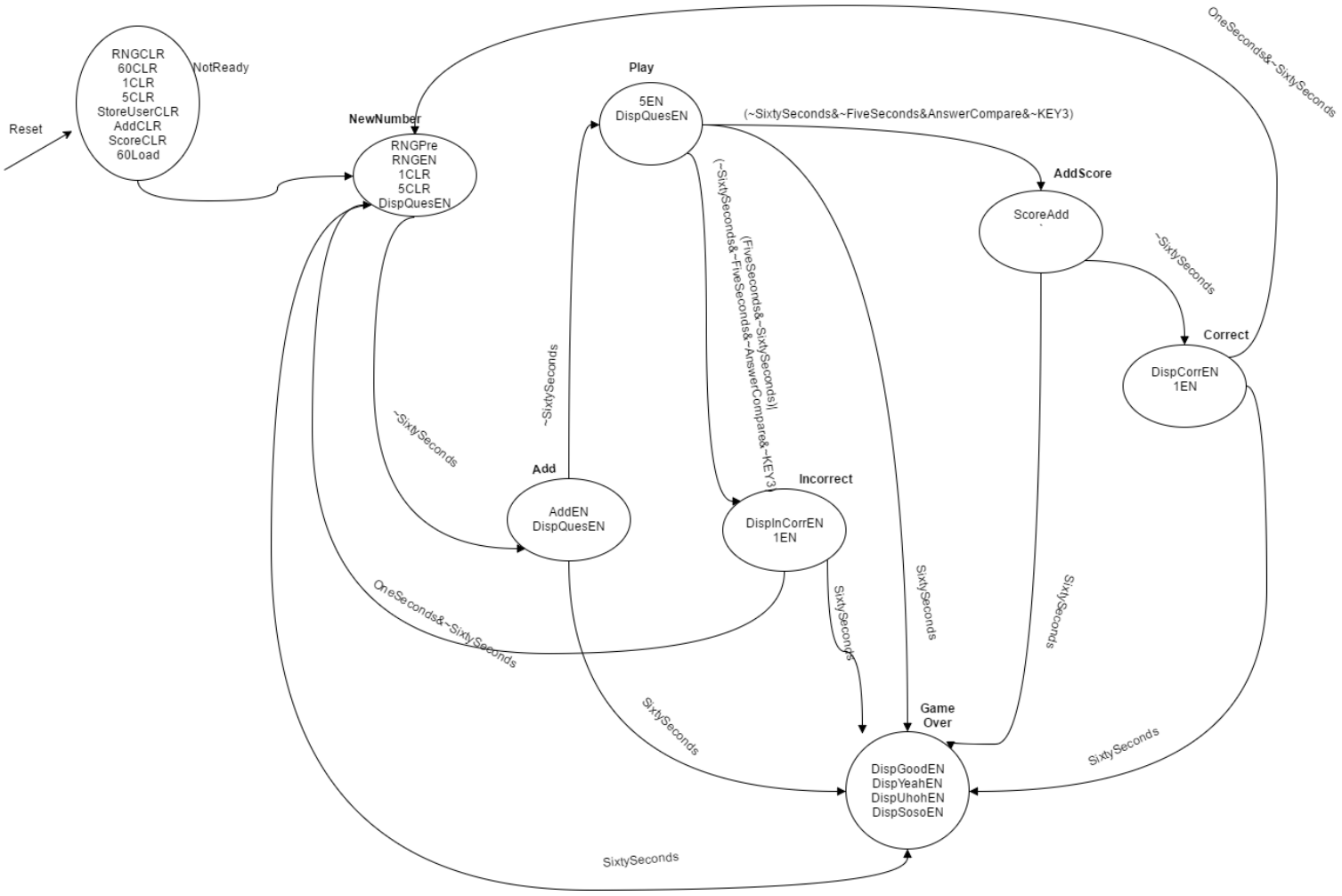
b) Control Unit:

i. Draw a block diagram of the Control Unit showing its primary inputs and outputs as well as its control outputs to the Data Path and status inputs from the Data Path



ii. Design a Moore FSM to act as the Control Unit. Draw the state diagram labeling states, transition conditions and outputs for each state. Explain how it will work.





This will begin in the NotReady state which will clear all of the registers and load sixty into the count-down counter. Then on the next clock cycle, the game begins. The NewNumber State will generate a new number and store it, it will display the question, but it will most likely be the previous question, this will most likely go un-noticed by most people, as this will only be up for 1/50,000,000th of a second. Then, so long as sixty seconds haven't passed, the transition will be to the Add state. This will enable the add register and will store the sum of the two stored random numbers. This state will display the correct question. The Adder State will then move to the play state, given that sixty seconds haven't passed. The play state will show the question, and then it enables the five second counter to begin. If the user doesn't answer within those 5 seconds, or the user enters the wrong input, (which is determined by the external button press of KEY3) the Play State will transistion to the incorrect state. The incorrect State will show the "incorrect" display. This display is just the two random numbers, the correct sum, and letter n after the correct sum. This state will also begin the 1 second counter, once that finishes, it will be redirected to the NewNumber State.

But, if the user answers before 5 seconds is up and inputs the correct answer, the Play state will transition to the AddScore state, which will display the "correct" display. This display is just the two random numbers, the correct sum, and letter y after the correct sum. The AddScore state also sets ScoreAdd to high, which will enable the score counter. This state is only active for 1 clock cycle, so the next state is the Correct state. This state continues the "correct" display and will begin a 1 sec counter. After that one second is up, the Correct state will transition to NewNumber.

But if at any point the sixty second timer is up, the state machine will transition to the Game Over state. This state will output one of the score displays depending on the signal from the Data Unit. If the score is less than 6, the Uhoh display will be enabled. If it is greater than or equal to 6 but still less than 13, the Soso display will be enabled. If the score is greater than or equal to 13 but is less than 20, the Good display will enable. And if the score is greater than or equal to 20, the Yeah display will enable. Each display is just the respective word written on the hexes along will the score that the user earned.

4. DISCUSSION

Discuss your experience designing for this project, how well your team worked together and what portions of the project work well, work less well, don't work or can be improved.

Answer:

Our team worked very well together. The team was able to develop and manage the project very quickly and divided the project amongst the group members. As individuals we were able to complete our assigned programming pieces in a short period of time, (Jeff did the Clock, Edgar did the LFSR & RNG, and Justin did the Display Unit, between Justin and Jeff the team created all the other smaller pieces like the adders, and comparators etc.) which allowed us to bring the pieces together to create the larger data unit and state machines; however we

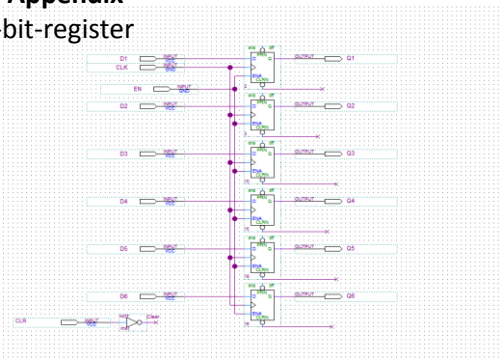
started to run into issues with the different pieces of the program not communicating effectively.

As a team we began the troubleshooting process, which at first started out with each team member methodically reworking our own pieces against the larger data unit. The troubleshooting process then migrated to eventually using one Data Unit and trying to change all the pieces at the same time while working within one Data Unit, which didn't work so well, it even created some communication problems as one person would continue to troubleshoot and the group's members weren't aware of the current status of the project this created a problem of being able to effectively track all the changes that were made. We were able as a team to mitigate this problem and all get back on the same page.

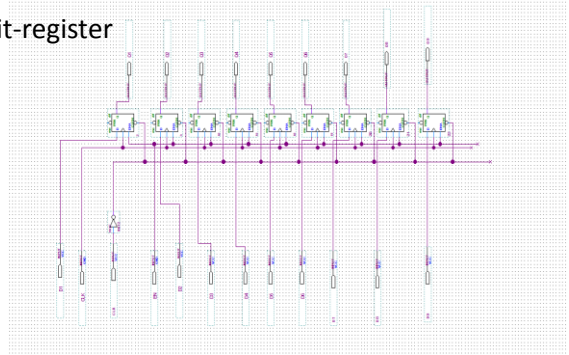
Overall this project was a lot of fun and challenging, we were able to get our project to almost work, I think with some more time we would have been able to work out the bugs on our own and make the project work successfully. We were sure that there was probably a lot of ways that our programming could have been improved upon. There are however many ways that this project could have been programmed.

5. Appendix

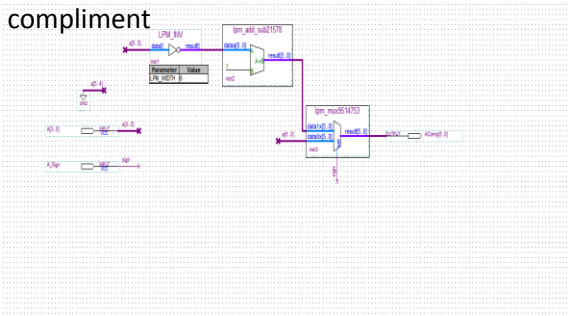
6-bit-register



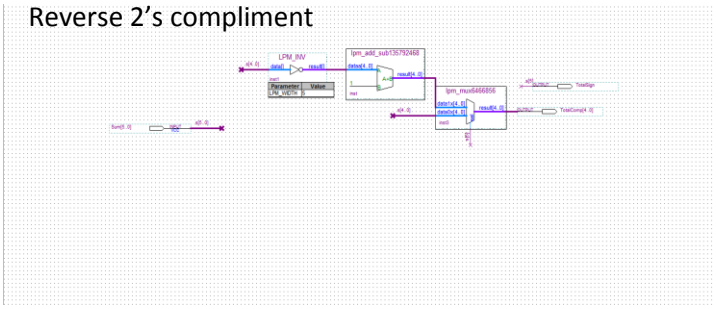
10-bit-register



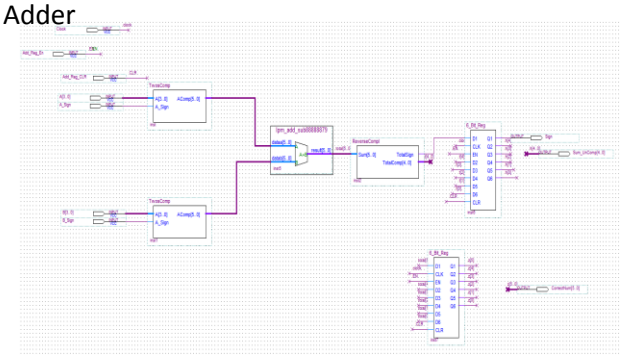
2's compliment



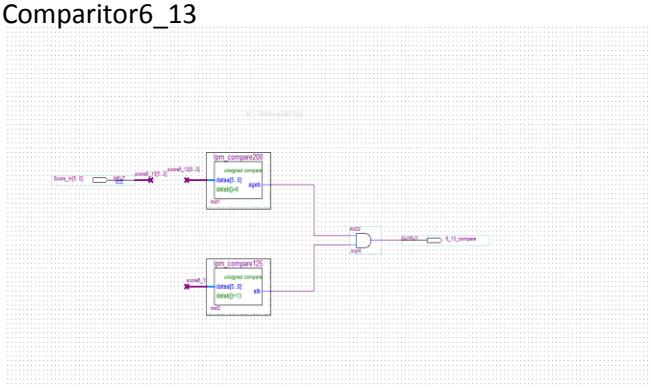
Reverse 2's compliment



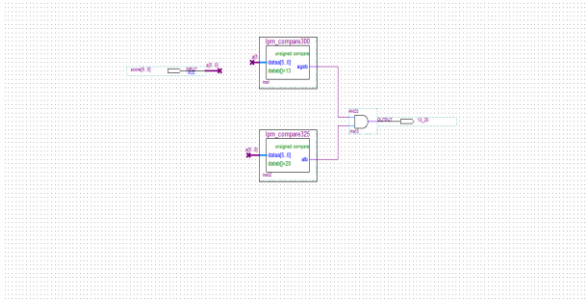
Adder



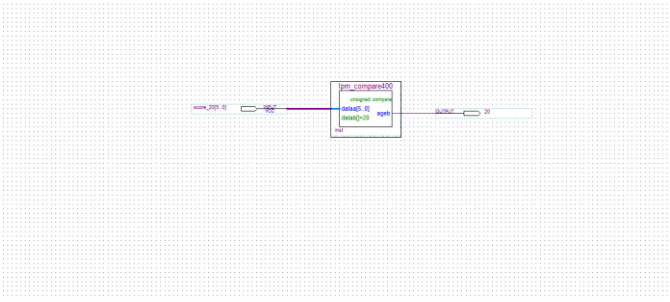
Comparitor6_13



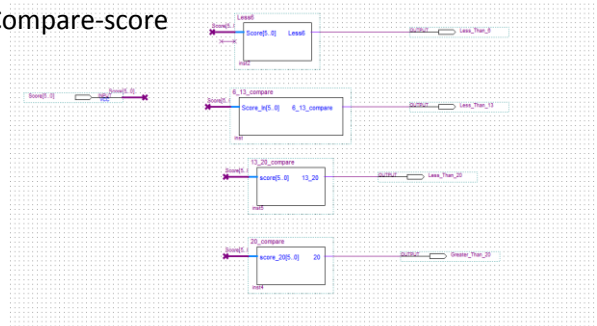
Comparitor13_20



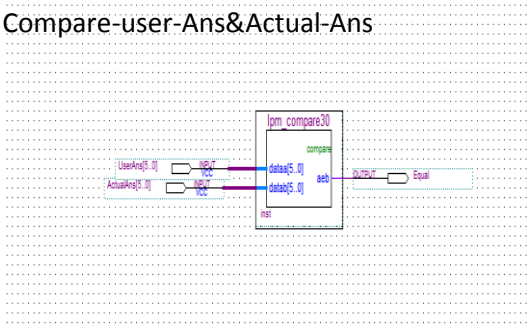
Comparitor_20



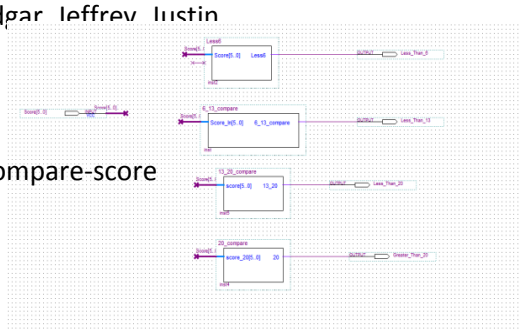
Compare-score



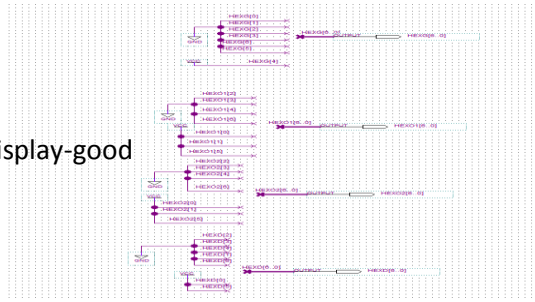
Compare-user-Ans&Actual-Ans



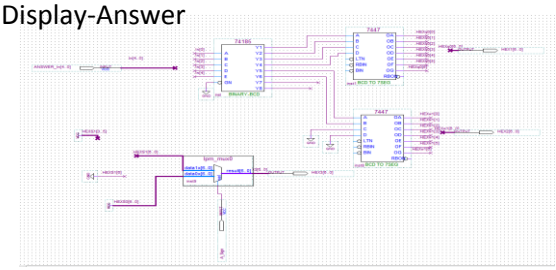
Compare-score



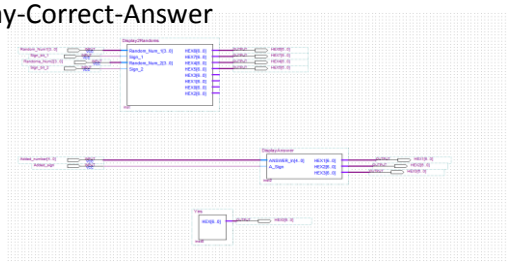
Display-good



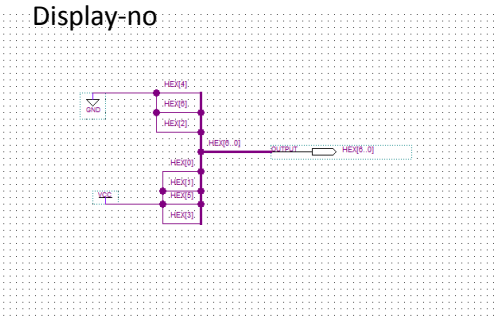
Display-Answer



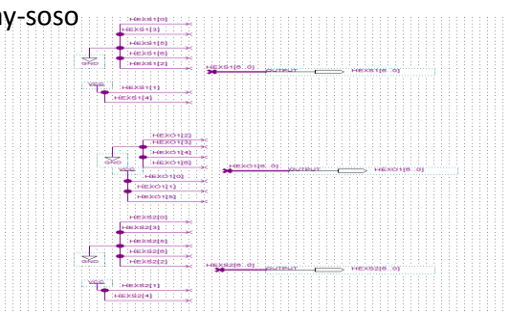
Display-Correct-Answer



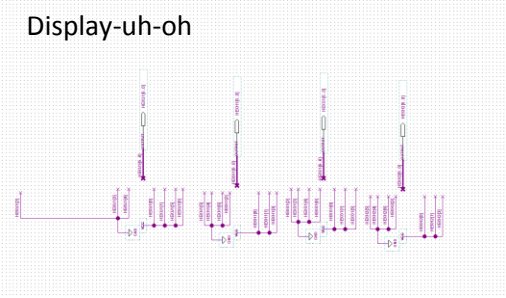
Display-no



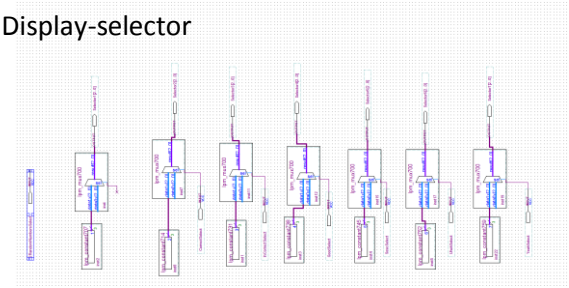
Display-soso



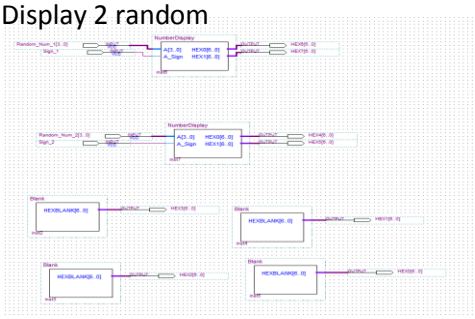
Display-uh-oh



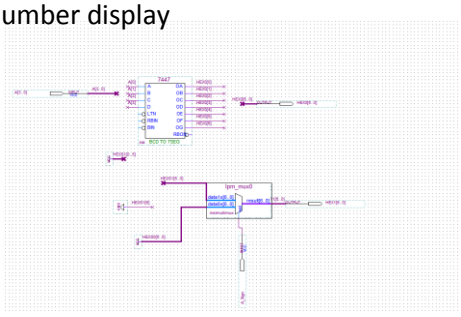
Display-selector



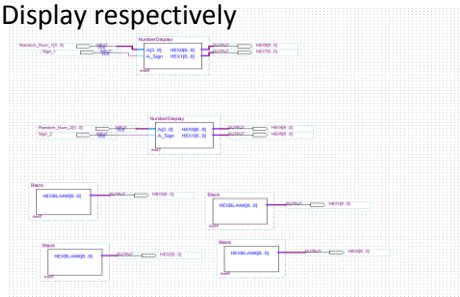
Display 2 random



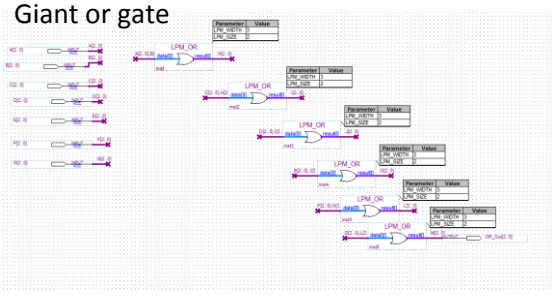
number display



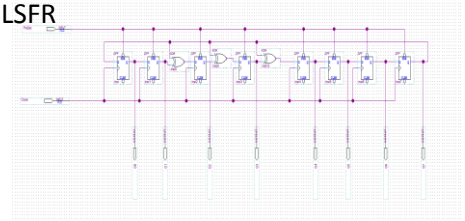
Display respectively



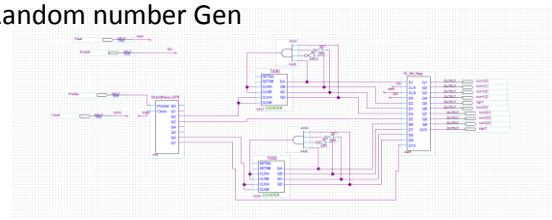
Giant or gate



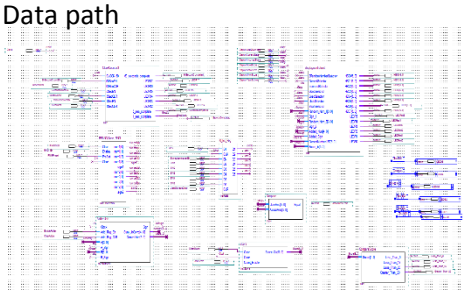
LSFR



Random number Gen

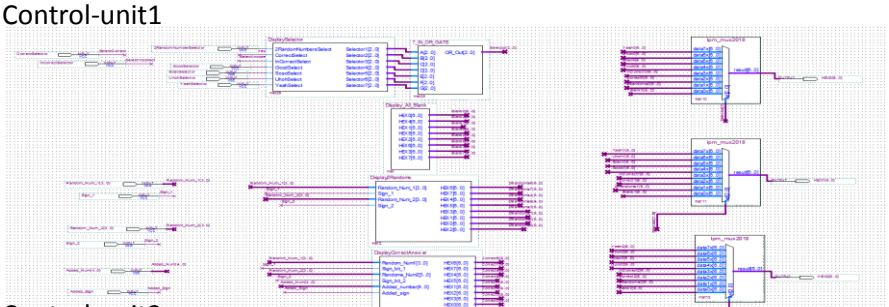


Data path

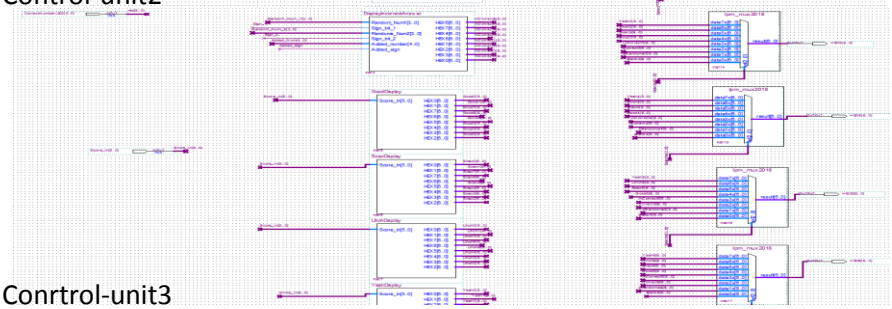


Clock

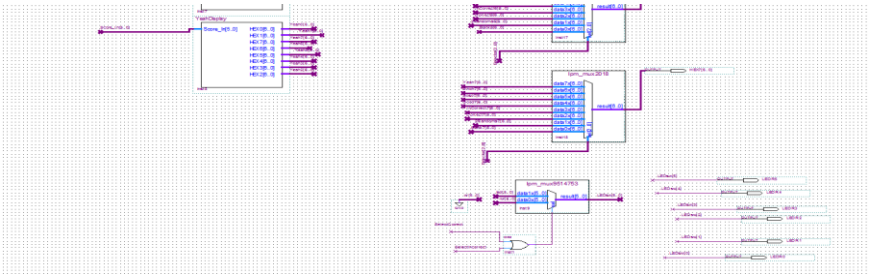
Control-unit1



Control-unit2



Conrtrol-unit3



DE2 File

Altera recommends that you do not modify this file. This
file is updated automatically by the Quartus II software
and any changes you make may be lost or overwritten.

```
set_global_assignment -name DEVICE EP2C35F672C6
set_global_assignment -name FAMILY "Cyclone II"

set_location_assignment PIN_N25 -to SW0
set_location_assignment PIN_N26 -to SW1
set_location_assignment PIN_P25 -to SW2
set_location_assignment PIN_AE14 -to SW3
set_location_assignment PIN_AF14 -to SW4
set_location_assignment PIN_AD13 -to SW5
set_location_assignment PIN_AC13 -to SW6
set_location_assignment PIN_C13 -to SW7
set_location_assignment PIN_B13 -to SW8
set_location_assignment PIN_A13 -to SW9
set_location_assignment PIN_N1 -to SW10
set_location_assignment PIN_P1 -to SW11
set_location_assignment PIN_P2 -to SW12
set_location_assignment PIN_T7 -to SW13
set_location_assignment PIN_U3 -to SW14
set_location_assignment PIN_U4 -to SW15
set_location_assignment PIN_V1 -to SW16
set_location_assignment PIN_V2 -to SW17
set_location_assignment PIN_T6 -to DRAM_ADDR[0]
set_location_assignment PIN_V4 -to DRAM_ADDR[1]
set_location_assignment PIN_V3 -to DRAM_ADDR[2]
set_location_assignment PIN_W2 -to DRAM_ADDR[3]
set_location_assignment PIN_W1 -to DRAM_ADDR[4]
set_location_assignment PIN_U6 -to DRAM_ADDR[5]
set_location_assignment PIN_U7 -to DRAM_ADDR[6]
set_location_assignment PIN_U5 -to DRAM_ADDR[7]
set_location_assignment PIN_W4 -to DRAM_ADDR[8]
set_location_assignment PIN_W3 -to DRAM_ADDR[9]
```

```
set_location_assignment PIN_Y1 -to DRAM_ADDR[10]
set_location_assignment PIN_V5 -to DRAM_ADDR[11]
set_location_assignment PIN_AE2 -to DRAM_BA_0
set_location_assignment PIN_AE3 -to DRAM_BA_1
set_location_assignment PIN_AB3 -to DRAM_CAS_N
set_location_assignment PIN_AA6 -to DRAM_CKE
set_location_assignment PIN_AA7 -to DRAM_CLK
set_location_assignment PIN_AC3 -to DRAM_CS_N
set_location_assignment PIN_V6 -to DRAM_DQ[0]
set_location_assignment PIN_AA2 -to DRAM_DQ[1]
set_location_assignment PIN_AA1 -to DRAM_DQ[2]
set_location_assignment PIN_Y3 -to DRAM_DQ[3]
set_location_assignment PIN_Y4 -to DRAM_DQ[4]
set_location_assignment PIN_R8 -to DRAM_DQ[5]
set_location_assignment PIN_T8 -to DRAM_DQ[6]
set_location_assignment PIN_V7 -to DRAM_DQ[7]
set_location_assignment PIN_W6 -to DRAM_DQ[8]
set_location_assignment PIN_AB2 -to DRAM_DQ[9]
set_location_assignment PIN_AB1 -to DRAM_DQ[10]
set_location_assignment PIN_AA4 -to DRAM_DQ[11]
set_location_assignment PIN_AA3 -to DRAM_DQ[12]
set_location_assignment PIN_AC2 -to DRAM_DQ[13]
set_location_assignment PIN_AC1 -to DRAM_DQ[14]
set_location_assignment PIN_AA5 -to DRAM_DQ[15]
set_location_assignment PIN_AD2 -to DRAM_LDQM
set_location_assignment PIN_Y5 -to DRAM_UDQM
set_location_assignment PIN_AB4 -to DRAM_RAS_N
set_location_assignment PIN_AD3 -to DRAM_WE_N
set_location_assignment PIN_AC18 -to FL_ADDR[0]
set_location_assignment PIN_AB18 -to FL_ADDR[1]
set_location_assignment PIN_AE19 -to FL_ADDR[2]
set_location_assignment PIN_AF19 -to FL_ADDR[3]
set_location_assignment PIN_AE18 -to FL_ADDR[4]
set_location_assignment PIN_AF18 -to FL_ADDR[5]
set_location_assignment PIN_Y16 -to FL_ADDR[6]
set_location_assignment PIN_AA16 -to FL_ADDR[7]
set_location_assignment PIN_AD17 -to FL_ADDR[8]
set_location_assignment PIN_AC17 -to FL_ADDR[9]
set_location_assignment PIN_AE17 -to FL_ADDR[10]
set_location_assignment PIN_AF17 -to FL_ADDR[11]
set_location_assignment PIN_W16 -to FL_ADDR[12]
set_location_assignment PIN_W15 -to FL_ADDR[13]
set_location_assignment PIN_AC16 -to FL_ADDR[14]
set_location_assignment PIN_AD16 -to FL_ADDR[15]
set_location_assignment PIN_AE16 -to FL_ADDR[16]
set_location_assignment PIN_AC15 -to FL_ADDR[17]
set_location_assignment PIN_AB15 -to FL_ADDR[18]
set_location_assignment PIN_AA15 -to FL_ADDR[19]
set_location_assignment PIN_Y15 -to FL_ADDR[20]
set_location_assignment PIN_Y14 -to FL_ADDR[21]
set_location_assignment PIN_V17 -to FL_CE_N
set_location_assignment PIN_W17 -to FL_OE_N
set_location_assignment PIN_AD19 -to FL_DQ[0]
```

```
set_location_assignment PIN_AC19 -to FL_DQ[1]
set_location_assignment PIN_AF20 -to FL_DQ[2]
set_location_assignment PIN_AE20 -to FL_DQ[3]
set_location_assignment PIN_AB20 -to FL_DQ[4]
set_location_assignment PIN_AC20 -to FL_DQ[5]
set_location_assignment PIN_AF21 -to FL_DQ[6]
set_location_assignment PIN_AE21 -to FL_DQ[7]
set_location_assignment PIN_AA18 -to FL_RST_N
set_location_assignment PIN_AA17 -to FL_WE_N
set_location_assignment PIN_AF10 -to HEX0[0]
set_location_assignment PIN_AB12 -to HEX0[1]
set_location_assignment PIN_AC12 -to HEX0[2]
set_location_assignment PIN_AD11 -to HEX0[3]
set_location_assignment PIN_AE11 -to HEX0[4]
set_location_assignment PIN_V14 -to HEX0[5]
set_location_assignment PIN_V13 -to HEX0[6]
set_location_assignment PIN_V20 -to HEX1[0]
set_location_assignment PIN_V21 -to HEX1[1]
set_location_assignment PIN_W21 -to HEX1[2]
set_location_assignment PIN_Y22 -to HEX1[3]
set_location_assignment PIN_AA24 -to HEX1[4]
set_location_assignment PIN_AA23 -to HEX1[5]
set_location_assignment PIN_AB24 -to HEX1[6]
set_location_assignment PIN_AB23 -to HEX2[0]
set_location_assignment PIN_V22 -to HEX2[1]
set_location_assignment PIN_AC25 -to HEX2[2]
set_location_assignment PIN_AC26 -to HEX2[3]
set_location_assignment PIN_AB26 -to HEX2[4]
set_location_assignment PIN_AB25 -to HEX2[5]
set_location_assignment PIN_Y24 -to HEX2[6]
set_location_assignment PIN_Y23 -to HEX3[0]
set_location_assignment PIN_AA25 -to HEX3[1]
set_location_assignment PIN_AA26 -to HEX3[2]
set_location_assignment PIN_Y26 -to HEX3[3]
set_location_assignment PIN_Y25 -to HEX3[4]
set_location_assignment PIN_U22 -to HEX3[5]
set_location_assignment PIN_W24 -to HEX3[6]
set_location_assignment PIN_U9 -to HEX4[0]
set_location_assignment PIN_U1 -to HEX4[1]
set_location_assignment PIN_U2 -to HEX4[2]
set_location_assignment PIN_T4 -to HEX4[3]
set_location_assignment PIN_R7 -to HEX4[4]
set_location_assignment PIN_R6 -to HEX4[5]
set_location_assignment PIN_T3 -to HEX4[6]
set_location_assignment PIN_T2 -to HEX5[0]
set_location_assignment PIN_P6 -to HEX5[1]
set_location_assignment PIN_P7 -to HEX5[2]
set_location_assignment PIN_T9 -to HEX5[3]
set_location_assignment PIN_R5 -to HEX5[4]
set_location_assignment PIN_R4 -to HEX5[5]
set_location_assignment PIN_R3 -to HEX5[6]
set_location_assignment PIN_R2 -to HEX6[0]
set_location_assignment PIN_P4 -to HEX6[1]
```

```
set_location_assignment PIN_P3 -to HEX6[2]
set_location_assignment PIN_M2 -to HEX6[3]
set_location_assignment PIN_M3 -to HEX6[4]
set_location_assignment PIN_M5 -to HEX6[5]
set_location_assignment PIN_M4 -to HEX6[6]
set_location_assignment PIN_L3 -to HEX7[0]
set_location_assignment PIN_L2 -to HEX7[1]
set_location_assignment PIN_L9 -to HEX7[2]
set_location_assignment PIN_L6 -to HEX7[3]
set_location_assignment PIN_L7 -to HEX7[4]
set_location_assignment PIN_P9 -to HEX7[5]
set_location_assignment PIN_N9 -to HEX7[6]
set_location_assignment PIN_G26 -to KEY0
set_location_assignment PIN_N23 -to KEY1
set_location_assignment PIN_P23 -to KEY2
set_location_assignment PIN_W26 -to KEY3
set_location_assignment PIN_AE23 -to LEDR[0]
set_location_assignment PIN_AF23 -to LEDR[1]
set_location_assignment PIN_AB21 -to LEDR[2]
set_location_assignment PIN_AC22 -to LEDR[3]
set_location_assignment PIN_AD22 -to LEDR[4]
set_location_assignment PIN_AD23 -to LEDR[5]
set_location_assignment PIN_AD21 -to LEDR[6]
set_location_assignment PIN_AC21 -to LEDR[7]
set_location_assignment PIN_AA14 -to LEDR[8]
set_location_assignment PIN_Y13 -to LEDR[9]
set_location_assignment PIN_AA13 -to LEDR[10]
set_location_assignment PIN_AC14 -to LEDR[11]
set_location_assignment PIN_AD15 -to LEDR[12]
set_location_assignment PIN_AE15 -to LEDR[13]
set_location_assignment PIN_AF13 -to LEDR[14]
set_location_assignment PIN_AE13 -to LEDR[15]
set_location_assignment PIN_AE12 -to LEDR[16]
set_location_assignment PIN_AD12 -to LEDR[17]
set_location_assignment PIN_AE22 -to LEDG[0]
set_location_assignment PIN_AF22 -to LEDG[1]
set_location_assignment PIN_W19 -to LEDG[2]
set_location_assignment PIN_V18 -to LEDG[3]
set_location_assignment PIN_U18 -to LEDG[4]
set_location_assignment PIN_U17 -to LEDG[5]
set_location_assignment PIN_AA20 -to LEDG[6]
set_location_assignment PIN_Y18 -to LEDG[7]
set_location_assignment PIN_Y12 -to LEDG[8]
set_location_assignment PIN_D13 -to CLOCK_27
set_location_assignment PIN_N2 -to CLOCK_50
set_location_assignment PIN_P26 -to EXT_CLOCK
set_location_assignment PIN_D26 -to PS2_CLK
set_location_assignment PIN_C24 -to PS2_DAT
set_location_assignment PIN_C25 -to UART_RXD
set_location_assignment PIN_B25 -to UART_TXD
set_location_assignment PIN_K4 -to LCD_RW
set_location_assignment PIN_K3 -to LCD_EN
set_location_assignment PIN_K1 -to LCD_RS
```

```
set_location_assignment PIN_J1 -to LCD_DATA[0]
set_location_assignment PIN_J2 -to LCD_DATA[1]
set_location_assignment PIN_H1 -to LCD_DATA[2]
set_location_assignment PIN_H2 -to LCD_DATA[3]
set_location_assignment PIN_J4 -to LCD_DATA[4]
set_location_assignment PIN_J3 -to LCD_DATA[5]
set_location_assignment PIN_H4 -to LCD_DATA[6]
set_location_assignment PIN_H3 -to LCD_DATA[7]
set_location_assignment PIN_L4 -to LCD_ON
set_location_assignment PIN_K2 -to LCD_BLON
set_location_assignment PIN_AE4 -to SRAM_ADDR[0]
set_location_assignment PIN_AF4 -to SRAM_ADDR[1]
set_location_assignment PIN_AC5 -to SRAM_ADDR[2]
set_location_assignment PIN_AC6 -to SRAM_ADDR[3]
set_location_assignment PIN_AD4 -to SRAM_ADDR[4]
set_location_assignment PIN_AD5 -to SRAM_ADDR[5]
set_location_assignment PIN_AE5 -to SRAM_ADDR[6]
set_location_assignment PIN_AF5 -to SRAM_ADDR[7]
set_location_assignment PIN_AD6 -to SRAM_ADDR[8]
set_location_assignment PIN_AD7 -to SRAM_ADDR[9]
set_location_assignment PIN_V10 -to SRAM_ADDR[10]
set_location_assignment PIN_V9 -to SRAM_ADDR[11]
set_location_assignment PIN_AC7 -to SRAM_ADDR[12]
set_location_assignment PIN_W8 -to SRAM_ADDR[13]
set_location_assignment PIN_W10 -to SRAM_ADDR[14]
set_location_assignment PIN_Y10 -to SRAM_ADDR[15]
set_location_assignment PIN_AB8 -to SRAM_ADDR[16]
set_location_assignment PIN_AC8 -to SRAM_ADDR[17]
set_location_assignment PIN_AD8 -to SRAM_DQ[0]
set_location_assignment PIN_AE6 -to SRAM_DQ[1]
set_location_assignment PIN_AF6 -to SRAM_DQ[2]
set_location_assignment PIN_AA9 -to SRAM_DQ[3]
set_location_assignment PIN_AA10 -to SRAM_DQ[4]
set_location_assignment PIN_AB10 -to SRAM_DQ[5]
set_location_assignment PIN_AA11 -to SRAM_DQ[6]
set_location_assignment PIN_Y11 -to SRAM_DQ[7]
set_location_assignment PIN_AE7 -to SRAM_DQ[8]
set_location_assignment PIN_AF7 -to SRAM_DQ[9]
set_location_assignment PIN_AE8 -to SRAM_DQ[10]
set_location_assignment PIN_AF8 -to SRAM_DQ[11]
set_location_assignment PIN_W11 -to SRAM_DQ[12]
set_location_assignment PIN_W12 -to SRAM_DQ[13]
set_location_assignment PIN_AC9 -to SRAM_DQ[14]
set_location_assignment PIN_AC10 -to SRAM_DQ[15]
set_location_assignment PIN_AE10 -to SRAM_WE_N
set_location_assignment PIN_AD10 -to SRAM_OE_N
set_location_assignment PIN_AF9 -to SRAM_UB_N
set_location_assignment PIN_AE9 -to SRAM_LB_N
set_location_assignment PIN_AC11 -to SRAM_CE_N
set_location_assignment PIN_K7 -to OTG_ADDR[0]
set_location_assignment PIN_F2 -to OTG_ADDR[1]
set_location_assignment PIN_F1 -to OTG_CS_N
set_location_assignment PIN_G2 -to OTG_RD_N
```

```
set_location_assignment PIN_G1 -to OTG_WR_N
set_location_assignment PIN_G5 -to OTG_RST_N
set_location_assignment PIN_F4 -to OTG_DATA[0]
set_location_assignment PIN_D2 -to OTG_DATA[1]
set_location_assignment PIN_D1 -to OTG_DATA[2]
set_location_assignment PIN_F7 -to OTG_DATA[3]
set_location_assignment PIN_J5 -to OTG_DATA[4]
set_location_assignment PIN_J8 -to OTG_DATA[5]
set_location_assignment PIN_J7 -to OTG_DATA[6]
set_location_assignment PIN_H6 -to OTG_DATA[7]
set_location_assignment PIN_E2 -to OTG_DATA[8]
set_location_assignment PIN_E1 -to OTG_DATA[9]
set_location_assignment PIN_K6 -to OTG_DATA[10]
set_location_assignment PIN_K5 -to OTG_DATA[11]
set_location_assignment PIN_G4 -to OTG_DATA[12]
set_location_assignment PIN_G3 -to OTG_DATA[13]
set_location_assignment PIN_J6 -to OTG_DATA[14]
set_location_assignment PIN_K8 -to OTG_DATA[15]
set_location_assignment PIN_B3 -to OTG_INT0
set_location_assignment PIN_C3 -to OTG_INT1
set_location_assignment PIN_C2 -to OTG_DACK0_N
set_location_assignment PIN_B2 -to OTG_DACK1_N
set_location_assignment PIN_F6 -to OTG_DREQ0
set_location_assignment PIN_E5 -to OTG_DREQ1
set_location_assignment PIN_F3 -to OTG_FSPEED
set_location_assignment PIN_G6 -to OTG_LSPEED
set_location_assignment PIN_B14 -to TDI
set_location_assignment PIN_A14 -to TCS
set_location_assignment PIN_D14 -to TCK
set_location_assignment PIN_F14 -to TDO
set_location_assignment PIN_C4 -to TD_RESET
set_location_assignment PIN_C8 -to VGA_R[0]
set_location_assignment PIN_F10 -to VGA_R[1]
set_location_assignment PIN_G10 -to VGA_R[2]
set_location_assignment PIN_D9 -to VGA_R[3]
set_location_assignment PIN_C9 -to VGA_R[4]
set_location_assignment PIN_A8 -to VGA_R[5]
set_location_assignment PIN_H11 -to VGA_R[6]
set_location_assignment PIN_H12 -to VGA_R[7]
set_location_assignment PIN_F11 -to VGA_R[8]
set_location_assignment PIN_E10 -to VGA_R[9]
set_location_assignment PIN_B9 -to VGA_G[0]
set_location_assignment PIN_A9 -to VGA_G[1]
set_location_assignment PIN_C10 -to VGA_G[2]
set_location_assignment PIN_D10 -to VGA_G[3]
set_location_assignment PIN_B10 -to VGA_G[4]
set_location_assignment PIN_A10 -to VGA_G[5]
set_location_assignment PIN_G11 -to VGA_G[6]
set_location_assignment PIN_D11 -to VGA_G[7]
set_location_assignment PIN_E12 -to VGA_G[8]
set_location_assignment PIN_D12 -to VGA_G[9]
set_location_assignment PIN_J13 -to VGA_B[0]
set_location_assignment PIN_J14 -to VGA_B[1]
```

```
set_location_assignment PIN_F12 -to VGA_B[2]
set_location_assignment PIN_G12 -to VGA_B[3]
set_location_assignment PIN_J10 -to VGA_B[4]
set_location_assignment PIN_J11 -to VGA_B[5]
set_location_assignment PIN_C11 -to VGA_B[6]
set_location_assignment PIN_B11 -to VGA_B[7]
set_location_assignment PIN_C12 -to VGA_B[8]
set_location_assignment PIN_B12 -to VGA_B[9]
set_location_assignment PIN_B8 -to VGA_CLK
set_location_assignment PIN_D6 -to VGA_BLANK
set_location_assignment PIN_A7 -to VGA_HS
set_location_assignment PIN_D8 -to VGA_VS
set_location_assignment PIN_B7 -to VGA_SYNC
set_location_assignment PIN_A6 -to I2C_SCLK
set_location_assignment PIN_B6 -to I2C_SDAT
set_location_assignment PIN_J9 -to TD_DATA[0]
set_location_assignment PIN_E8 -to TD_DATA[1]
set_location_assignment PIN_H8 -to TD_DATA[2]
set_location_assignment PIN_H10 -to TD_DATA[3]
set_location_assignment PIN_G9 -to TD_DATA[4]
set_location_assignment PIN_F9 -to TD_DATA[5]
set_location_assignment PIN_D7 -to TD_DATA[6]
set_location_assignment PIN_C7 -to TD_DATA[7]
set_location_assignment PIN_D5 -to TD_HS
set_location_assignment PIN_K9 -to TD_VS
set_location_assignment PIN_C5 -to AUD_ADCLRCK
set_location_assignment PIN_B5 -to AUD_ADCDAT
set_location_assignment PIN_C6 -to AUD_DACLRCK
set_location_assignment PIN_A4 -to AUD_DACDAT
set_location_assignment PIN_A5 -to AUD_XCK
set_location_assignment PIN_B4 -to AUD_BCLK
set_location_assignment PIN_D17 -to ENET_DATA[0]
set_location_assignment PIN_C17 -to ENET_DATA[1]
set_location_assignment PIN_B18 -to ENET_DATA[2]
set_location_assignment PIN_A18 -to ENET_DATA[3]
set_location_assignment PIN_B17 -to ENET_DATA[4]
set_location_assignment PIN_A17 -to ENET_DATA[5]
set_location_assignment PIN_B16 -to ENET_DATA[6]
set_location_assignment PIN_B15 -to ENET_DATA[7]
set_location_assignment PIN_B20 -to ENET_DATA[8]
set_location_assignment PIN_A20 -to ENET_DATA[9]
set_location_assignment PIN_C19 -to ENET_DATA[10]
set_location_assignment PIN_D19 -to ENET_DATA[11]
set_location_assignment PIN_B19 -to ENET_DATA[12]
set_location_assignment PIN_A19 -to ENET_DATA[13]
set_location_assignment PIN_E18 -to ENET_DATA[14]
set_location_assignment PIN_D18 -to ENET_DATA[15]
set_location_assignment PIN_B24 -to ENET_CLK
set_location_assignment PIN_A21 -to ENET_CMD
set_location_assignment PIN_A23 -to ENET_CS_N
set_location_assignment PIN_B21 -to ENET_INT
set_location_assignment PIN_A22 -to ENET_RD_N
set_location_assignment PIN_B22 -to ENET_WR_N
```

```
set_location_assignment PIN_B23 -to ENET_RST_N
set_location_assignment PIN_AE24 -to IRDA_TXD
set_location_assignment PIN_AE25 -to IRDA_RXD
set_location_assignment PIN_AD24 -to SD_DAT
set_location_assignment PIN_AC23 -to SD_DAT3
set_location_assignment PIN_Y21 -to SD_CMD
set_location_assignment PIN_AD25 -to SD_CLK
set_location_assignment PIN_D25 -to GPIO_0[0]
set_location_assignment PIN_J22 -to GPIO_0[1]
set_location_assignment PIN_E26 -to GPIO_0[2]
set_location_assignment PIN_E25 -to GPIO_0[3]
set_location_assignment PIN_F24 -to GPIO_0[4]
set_location_assignment PIN_F23 -to GPIO_0[5]
set_location_assignment PIN_J21 -to GPIO_0[6]
set_location_assignment PIN_J20 -to GPIO_0[7]
set_location_assignment PIN_F25 -to GPIO_0[8]
set_location_assignment PIN_F26 -to GPIO_0[9]
set_location_assignment PIN_N18 -to GPIO_0[10]
set_location_assignment PIN_P18 -to GPIO_0[11]
set_location_assignment PIN_G23 -to GPIO_0[12]
set_location_assignment PIN_G24 -to GPIO_0[13]
set_location_assignment PIN_K22 -to GPIO_0[14]
set_location_assignment PIN_G25 -to GPIO_0[15]
set_location_assignment PIN_H23 -to GPIO_0[16]
set_location_assignment PIN_H24 -to GPIO_0[17]
set_location_assignment PIN_J23 -to GPIO_0[18]
set_location_assignment PIN_J24 -to GPIO_0[19]
set_location_assignment PIN_H25 -to GPIO_0[20]
set_location_assignment PIN_H26 -to GPIO_0[21]
set_location_assignment PIN_H19 -to GPIO_0[22]
set_location_assignment PIN_K18 -to GPIO_0[23]
set_location_assignment PIN_K19 -to GPIO_0[24]
set_location_assignment PIN_K21 -to GPIO_0[25]
set_location_assignment PIN_K23 -to GPIO_0[26]
set_location_assignment PIN_K24 -to GPIO_0[27]
set_location_assignment PIN_L21 -to GPIO_0[28]
set_location_assignment PIN_L20 -to GPIO_0[29]
set_location_assignment PIN_J25 -to GPIO_0[30]
set_location_assignment PIN_J26 -to GPIO_0[31]
set_location_assignment PIN_L23 -to GPIO_0[32]
set_location_assignment PIN_L24 -to GPIO_0[33]
set_location_assignment PIN_L25 -to GPIO_0[34]
set_location_assignment PIN_L19 -to GPIO_0[35]
set_location_assignment PIN_K25 -to GPIO_1[0]
set_location_assignment PIN_K26 -to GPIO_1[1]
set_location_assignment PIN_M22 -to GPIO_1[2]
set_location_assignment PIN_M23 -to GPIO_1[3]
set_location_assignment PIN_M19 -to GPIO_1[4]
set_location_assignment PIN_M20 -to GPIO_1[5]
set_location_assignment PIN_N20 -to GPIO_1[6]
set_location_assignment PIN_M21 -to GPIO_1[7]
set_location_assignment PIN_M24 -to GPIO_1[8]
set_location_assignment PIN_M25 -to GPIO_1[9]
```

```
set_location_assignment PIN_N24 -to GPIO_1[10]
set_location_assignment PIN_P24 -to GPIO_1[11]
set_location_assignment PIN_R25 -to GPIO_1[12]
set_location_assignment PIN_R24 -to GPIO_1[13]
set_location_assignment PIN_R20 -to GPIO_1[14]
set_location_assignment PIN_T22 -to GPIO_1[15]
set_location_assignment PIN_T23 -to GPIO_1[16]
set_location_assignment PIN_T24 -to GPIO_1[17]
set_location_assignment PIN_T25 -to GPIO_1[18]
set_location_assignment PIN_T18 -to GPIO_1[19]
set_location_assignment PIN_T21 -to GPIO_1[20]
set_location_assignment PIN_T20 -to GPIO_1[21]
set_location_assignment PIN_U26 -to GPIO_1[22]
set_location_assignment PIN_U25 -to GPIO_1[23]
set_location_assignment PIN_U23 -to GPIO_1[24]
set_location_assignment PIN_U24 -to GPIO_1[25]
set_location_assignment PIN_R19 -to GPIO_1[26]
set_location_assignment PIN_T19 -to GPIO_1[27]
set_location_assignment PIN_U20 -to GPIO_1[28]
set_location_assignment PIN_U21 -to GPIO_1[29]
set_location_assignment PIN_V26 -to GPIO_1[30]
set_location_assignment PIN_V25 -to GPIO_1[31]
set_location_assignment PIN_V24 -to GPIO_1[32]
set_location_assignment PIN_V23 -to GPIO_1[33]
set_location_assignment PIN_W25 -to GPIO_1[34]
set_location_assignment PIN_W23 -to GPIO_1[35]

set_global_assignment -name PARTITION_NETLIST_TYPE SOURCE -section_id Top

set_global_assignment -name IGNORE_CLOCK_SETTINGS ON
set_global_assignment -name FMAX_REQUIREMENT "50 MHz"
set_global_assignment -name PARTITION_COLOR 2147039 -section_id Top
set_global_assignment -name LL_ROOT_REGION ON -section_id "Root Region"
set_global_assignment -name LL_MEMBER_STATE LOCKED -section_id "Root Region"

set_instance_assignment -name FAST_INPUT_REGISTER ON -to *
set_instance_assignment -name FAST_OUTPUT_REGISTER ON -to *
set_instance_assignment -name TSU_REQUIREMENT "10 ns" -from * -to *
set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to *
set_instance_assignment -name CURRENT_STRENGTH_NEW "MAXIMUM CURRENT" -to I2C_SCLK
set_instance_assignment -name CURRENT_STRENGTH_NEW "MAXIMUM CURRENT" -to I2C_SDAT

set_instance_assignment -name PARTITION_HIERARCHY root_partition -to | -section_id Top
```