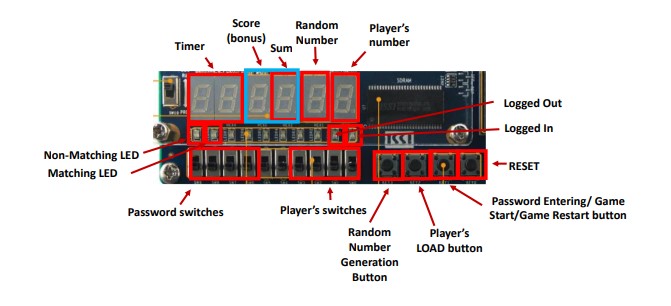
**FPGA Based Mental Math Binary Game**

**Introduction:**

Mental Math Binary Game is a system that can be used by a player to test himself/herself in their arithmetic addition skills. The player is checked for a password so he/she can log onto the board and start playing the game. In this game, a random number from 0 to 15 is displayed on the second right most 7-Segment display when a button is pressed. The player should observe the number displayed on the 7-segment display and should mentally convert the number observed into the binary version, then identify the matching number so that the sum of the two numbers is 1111 in binary. Player should then enter the matching number using Player’s set of toggle switches and then press the designated button. The number should be observed on the right most 7-segment display. The sum of the two numbers will be computed by the FPGA and displayed on the third rightmost 7-segment display device. If the sum matches 1111, the second leftmost LED will light up otherwise the leftmost LED will light up. There is a timer of 99 seconds in the game, this will start down counting from 99 to 00 once the player logged into the game and press the game start button. The player can check his/her skills as many times as possible within the time limit.

 **System Architecture Design and Simulation Results:**

**System Architecture:**

Diagram, schematic

Description automatically generated

The above figure shows how different modules are connected and which module is getting a signal as input from the hardware parts(buttons, toggle switches) and giving a signal as output to the hardware parts(LEDs, 7-Segment displays) on the FPGA board. In all the modules that are having clk signal, it is 50MHz frequency clock, and rst is the reset (active low signal).

**Decoder:**

A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. Here, a 4:16 decoder is used for converting 4bit binary coded decimal (BCD) to 7 segment display numbers(7bit).

Chart, box and whisker chart

Description automatically generated

The Decoder module takes **4bit input** and gives **7bit output** in this application. There are 5 decoder modules in Lab3- one for the random number display, one for the player entered digit display, two for the timer display, one for the sum of the player entered number and the random number to be displayed.

Diagram

Description automatically generated

**Decoder Simulation Results:**

**Graphical user interface, application

Description automatically generated**

Above waveform shows the output of a decoder that takes 4bit as input and converts it into 7segment display. Here the 7-segment is active low signal i.e., if the signal is low (0), the display line is high. In the above waveform, we can observe that the input is 1111 (F) and the output is displayed as 0111000 i.e., 0,4,5,6 lines are on and the remaining are off in the seven segments.

**Adder:**

The adder module has two inputs-one from the player input and the other from the random number generator. This module adds the two input values and gives the sum as output. Both the inputs and output are **4bit** each.

Diagram

Description automatically generated

**Adder Simulation Results:**

**Graphical user interface

Description automatically generated**

We gave 1000 and 0011 as inputs to the adder, it gives the output 1011, We gave 1000 and 1001 as inputs to the adder, it should give output as 10001 but it gave output as 0001 as we defined input as 4bit. We gave 1000 and 0111 as inputs to the adder, it gives the output 1111(Sum Matching Case).

**Verification:**

The Verification module has one input (Sum) of **4bits** coming from the Adder module. It checks the input with 1111 and gives the output accordingly. The output Matched is high if the given Sum matches to 1111 otherwise the output Unmatched is high. The outputs are connected to the LEDs on the FPGA board. Both the outputs are **single bit**.

Diagram

Description automatically generated

**Random\_Number Generator:**

The random number generator module generates a random number based on the time we hold the button. Internally, this module has a counter module which will start counting when we press the button and when we release the button, it gives the count at that moment as the random number. That’s how the random number generation depends on the time we hold the button.

This module has 3 inputs- clock(clk), reset(rst) and RNG\_Gen(coming from the access controller) and one output Random\_Number. RNG\_Gen and Random\_Number are 4bits each.

Diagram

Description automatically generated

Counter:

Counter module has 3 inputs- clock(clk), reset(rst) and count and one output count\_out. Count and count\_out are 4bits each. This counter module is the internal module for Random number generator module. This module counts from 0000 to 1111 and repeats again when the button is being pressed. LFSR based counter is used to build the Random number generator.

Diagram

Description automatically generated

**Random\_Number Simulation Results:**

**Graphical user interface

Description automatically generated**

The above waveform shows that a random number(Random\_Number) is generated whenever the button(RNG\_Gen) is pressed.

**Load Register:**

The load register is where the player inputs are stored and when the button (load) is pressed(activated) the player entered number is displayed on the 7-Segment display. This load register module has 4 inputs-clock(clk), reset(rst), input from the player toggle switches(D\_In), a signal to activate coming from the access controller(load). Input from the player is a 4bit input and the load is a single bit input. The output is the D\_In (i.e., D\_Out=D\_In) when the button is pressed.

Diagram

Description automatically generated

**Load Register Simulation Results:**

**Graphical user interface

Description automatically generated**

The waveform shows how the load signal controls the input. Whenever the load signal is high the output is generated from the input.

**GC\_AC :**

GC\_AC module checks whether the password is correct or not and it is responsible for controlling the load register, random number generator and digit timer modules. In other words, it can be said to be the activation and game control module. This module has 3 internal modules namely, ROM\_Authentication module, ROM\_PSWD module and Game\_Controller module. Game\_Controller module has the inputs coming from the buttons that control the random number generator module, load register module. It reconfigures the timer module with digits when the game is logged in by setting the correct password and the game start button is pressed. The ROM\_PSWD module and the ROM\_Authentication module are for the Password storage and for checking the user entered password with the stored ROM value.

Diagram, schematic

Description automatically generated

Timer\_reconfig and Timer\_enable signals goes from Game\_Controller module to the Digit\_Timer module and Time\_out signal comes from Digit\_Timer module to the Game\_Controller module. RNG\_Gen\_In signal is from the button that is pressed to generate a random number to the Game\_Controller module and RNG\_Gen\_Out is from the Game\_Controller module to the RNG\_Gen module. Load\_P1\_In signal is from the button that is pressed to load the player input to the Game\_Controller module and Load\_P1\_Out is from the Game\_Controller module to the Load\_Register module. That’s how the Game\_Controller module controls the game flow.

The password checking is done by the ROM\_Autherization module and it generates the Logged\_In and Logged\_Out signals accordingly as the password is correctly entered or not.

**ROM\_PSWD:**

This module is used to store passwords. This is the ROM generated by Quartus prime. ROM is a device that contains a set of words. It has two dimensions-(i)Width and (ii)Depth. It is only Read out; it is preprogrammed before we even start the usage of the system. It cannot be changed in content.

**ROM\_Authentication:**

ROM\_Authentication module is used to check the user entered password with the prestored password in the ROM\_PSWD.

**Digit Timer:**

Digit\_Timer module is responsible for the timer in the game. In this game, we have time of 99 seconds for the player to play the game.

Timer specifications: Two digits are used for timer control. After the reset, the timer should display and remain ‘00’ seconds. After passing the password check, the timer will display ‘99’ seconds. Before the timer starts, no game can be played (pushing the random number generation button or the player’s load button should not load any new numbers to the display). Pushing the game start/restart (same as the password entering button) once will start the timer. After the timer starts (before it expires), the game can be played. Once the timer reaches ‘00’ second, all game functions are disabled. Pushing the game start/restart button will load ‘99’ seconds to the timer. Pushing the game start/restart button once more will start a new game.

Diagram, schematic

Description automatically generated

**Digit\_Timer Simulations Results:**

**Graphical user interface

Description automatically generated**

**Graphical user interface

Description automatically generated**

**Graphical user interface

Description automatically generated**

**Digit:**

Digit module is the base for the digit timer. We use two digit modules to create Digit\_Timer for the game, one digit module for the 1’s digit in the timer and one more digit module for the ten’s digit. Digit module has 5 inputs and 3 outputs. Inputs to this module include clock(clk), reset(rst), reconfig which are for starting the module and BDN, NBUP are for the communication of the 1Sec\_Timer module and 10’s digit timer module respectively with 1’s digit module. Outputs include BUP to communicate with 10’s digit that it needed a borrow, and NBDN is to the access controller saying that time is out. DigitCount is the output, timer digit being displayed.

A picture containing diagram

Description automatically generated

**OneSec\_Timer:**

1Sec\_Timer module is the internal module for the digit timer module. It indicates the digit timer to decrease whenever there is 1 second time out. We built the OneSec\_Timer module using a One\_ms timer, counting 100 times to get 100ms time and then counting it 10 times to get the 1 second time out. The below figure shows how the OneSec\_Timer is built.

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

**OneSec\_Timer Simulation Results:**

**Graphical user interface

Description automatically generated**

The above waveform shows how the OneSec\_Timer works. Whenever 1ms timer gives timeout, 100ms timer starts counting, similarly when 100ms timer gives timeout, 1s timer starts counting.

**One\_ms\_Timer:**

One\_ms\_Timer module starts counting from 0 to 49,999 as we have 50MHz clock on our FPGA board. Once the count reaches the terminal value it gives the timeout signal. In this, a 16bit LFSR counter is used to generate 1ms time\_out.

Diagram

Description automatically generated

**Graphical user interface

Description automatically generated**

**Button Shaper:**

Whenever the button is pressed the button shaper module loads the access controller module with the signal to load the load registers. The button shaper module has clock(clk), reset(rst) and the button(B\_in) as inputs and the signal(B\_out) to the access controller as output.

Diagram

Description automatically generated

**Button Shaper Simulation Results:**

Graphical user interface

Description automatically generated

The above waveform shows how the button shaper module works. When the button is pressed the module gives the single pulse output.

**FPGA Board Testing Results:**

When the FPGA board is powered on, game logged\_out condition

**A picture containing text

Description automatically generated**

When reset button is pressed, game logged\_out condition

**A picture containing text

Description automatically generated**

The player entered the password correct, the board is logged in and the timer displays 99

**A picture containing text

Description automatically generated**

When Game Start button is pressed, the timer starts counting

**A picture containing text

Description automatically generated**

The random number generator button is pressed, a random number is generated.

**A picture containing text

Description automatically generated**

Player entered a number so that the sum of the random number and player entered number matches 1111 and the matched led on.

**A picture containing text

Description automatically generated**

Player entered a number so that the sum of the random number and player entered number does not matche 1111 and the unmatched led on.

**Text

Description automatically generated with medium confidence**

Time\_out

**A picture containing text

Description automatically generated**

Again game restart button is pressed

A picture containing text

Description automatically generated

The game start button is again pressed

**A picture containing text, electronics

Description automatically generated**

**Video link:** **https://drive.google.com/file/d/1I08PdF5ONRHEaqjGtONNN2ieBJPHndps/view?usp=share\_link**

**Conclusion:**

Successfully, built mental math binary game on FPGA with features- Password to login into the board, logged in and logged out indication, Timer display, Matching and Non-Matching sum indication. **The Random number generator module is built by using the LFSR counter.**