**Slot Machine**

*Project made by Șerban Dan-Andrei*

*Year I, computer science student*

* Specification of the project

A slot machine with a 1x3 grid is simulated on the FPGA board, using 3 seven-segments displays. Instead of the symbols, due to limitations, the 7 segment will display digits from 1 to 7. The random number generator for each digit will be simulated by a fast-counter that will go through the digits from 1 to 7, at different frequencies, for each digit.

Content

1. Block scheme
2. Block diagram
3. Components used
4. Block components
5. Notations and explications
6. Usage on the FPGA board

1. Block Scheme

Black box

Play\_ btn

 Bet(1 downto 0)

 sel

 clk

 Outp(4 downto 0)

 Seg(6 downto 0)

 An(3 downto 0)

Inputs:

Clk- internal clock signal

Play\_btn-simulates a button that spins the numbers

Sel – selection for the 7seg to choose between the sum and the spun numbers

Bet- the value of the value bet

Outputs:

Seg, An – used for the 7segment display

Outp – LED’s that signal a winning hand

1. Block Diagram

Timing module

Counter 1

Counter 2

Counter 3

 7 segment display

mux

Mux

Bcd to hex

Money ram

comparator

Win output

Win calculator

Error block

Play\_btn

Bet

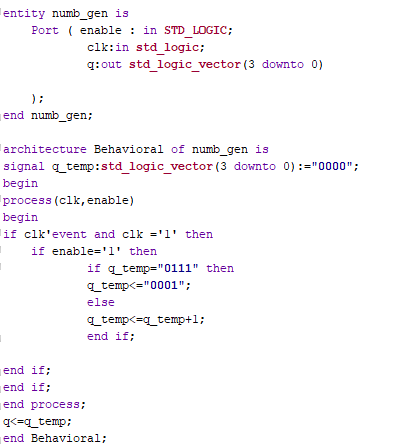
Sel

Clk

1. Components used

For each component, the main part of the vhdl code will also be given.

1. *Counter modulo 7*

As the name suggests, one of the basic components that is used in the project is a counter modulo 7 with the data path on 4 bits. This counter is used for the 3 “random number generators”. The counters used in the project use different frequencies in order to simulate the appearance of different numbers. On top of that, each counter also has an enable signal

1. *Frequency divider*

The frequency divider is used to divide the frequency of the in-board clock. In the project, 2 of these dividers are used, one to divide the main clock and another one to divide the divided clock into different frequencies.

A screenshot of a computer program

Description automatically generated

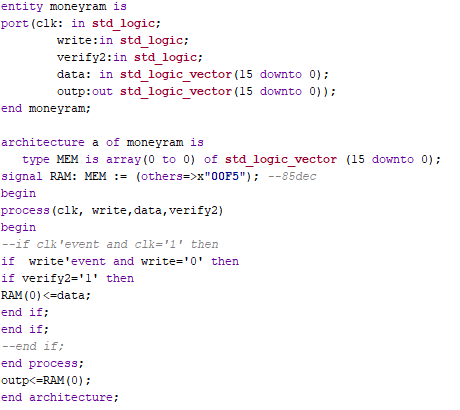
1. *7Segment Decoder*

A screenshot of a computer code

Description automatically generatedThe 7-segment decoder is a component that transforms a 4bit input number into its corresponding 7-bit number that is going to be used for the 7-segment. The used decoder is modified from a standard one in the idea that if the number is 14, then an “E” will be visible and if the number is 15 then no cathode is light up.

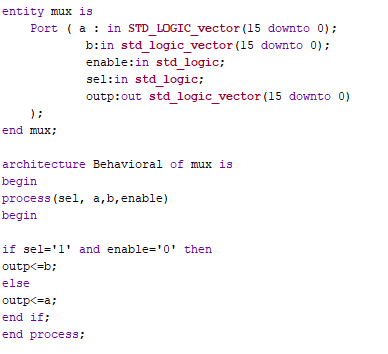
1. *Ram block*

The Ram is consisting of only one memory module of 16-bits which holds the “sum” which is at a given moment into the “machine”. It functions based on the play\_btn signal and also has an enable which is active only when the sum which is going to be bet is available in it.



1. *Mux with enable with data path on 16 bits*

The Multiplexor is on a data path of 16 bits and has 2 inputs with a one-bit selection. On top of that it also has an enable signal. The multiplexor has on its outputs the corresponding input based on the selection and if the enable is on.



1. *Mux without enable with data path on 16 bits*

Similar to the previous component, it functions exactly the same, but the only difference is that there is no enable signal present.

1. *4-bit Comparators*

A screenshot of a computer code

Description automatically generatedThe 4-bit comparator is a component that has the data path on 4 bits and signals if the 2 input numbers are equal or not and also if the numbers are different from the number 14, which will further be explained.

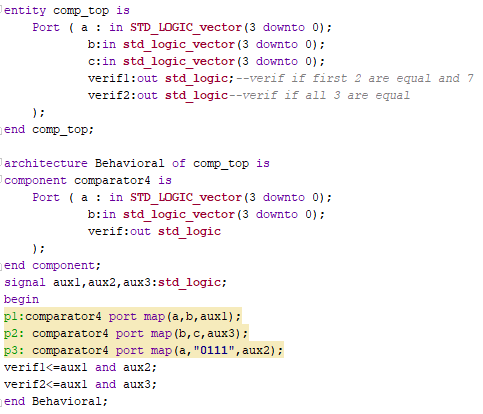
1. Block Components

The block components are either made with the previous components or they are specially made to give the desired design for such a project.

1. *Timing module*

The timing module is basically composed of the 2 frequency dividers which were described earlier. Firstly the first frequency divider is used to divide overall the 50 MHZ frequency of the board’s clock. The divided frequency is then divided by 2 and by 3. Those frequencies are needed in order to obtain different speeds of the random number generators made with the counters modulo 7. On top of that the component has the enables for the 3 counters. Thus the inputs of this block are the play\_btn signal and the internal clock and the outputs are 3 frequencies for the 3 random number generators and their enable signals

1. *Comparator*

The comparator block is used to compare the 3 output numbers after stopping the counters which generate the pseudo-random numbers. Thus, the inputs of the block are the 3 generated numbers and the outputs are 2 verification signals which signal to other blocks if there are equal numbers or not. 2 verification signals are needed in order to check if all the numbers are equal or only 2 of them which are specifically “7”. In the constituation of the block come 3 comparators with the data path on 4 bits which are mapped. The first comparator compares the 2 first numbers, the second one the 2nd and 3rd numbers and one final comparator which compares the first number with the number “7”. Then, using an AND gate, the output verification signals are properly given.

1. *Win output block*

The Winning output block is used to signal the user if the slot landed on a winning combination. The block takes as input the clock which is used for the last number of the slot, the verification signals from the comparator block, one of the numbers of the randomly generated numbers and an enable signal so that the component doesn’t work whilst the numbers are “generated”. The output of the block is a 5-bit number that will only take the values 0,1,2,4,8,16. The idea of this block is the number will be outputted on 5 LEDs which will continuously light up from right to left to signal the user it won.

The block works with multiple if-clauses that transform the number from 0 to 16 in order.

The block works only when the enable is ‘1’, which is when the play\_btn is set on low. The number will go through its phases only if one of the verification signals is ‘1’.

A screenshot of a computer code

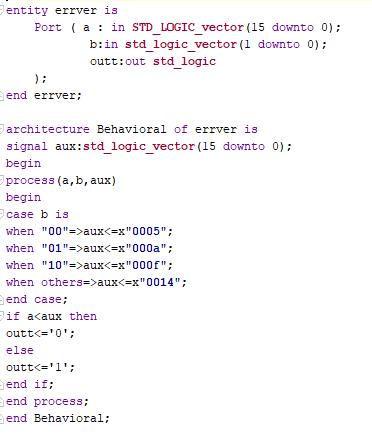
Description automatically generated

1. *Win calculator block*

The winning calculator block is used to calculate the sum, which is needed to be given to the user, based on the winning combination, the bet amount and the money which there is in the RAM block. The inputs of this block are the verification signals, the bet signals, which are input signals and a 16-bit signal which represents the sum of the ram block. The output of this block is the sum, given those signals, either lower or higher, based on the numbers which were generated. The idea of the block is to change the sum which will then be written on the ram block. It will change it followingly: if no verification is active, the sum betted is subtracted, or if there is a verification signal set on ‘1’ then a certain value is awarded, based on the bet amount again.

****

1. *Error block*

The role of the Error Block is to signal to the output that the sum which is wanted to be bet is not available in the memory, basically the user no longer has the sum needed to bet. The input of the block is the sum from the money ram and also the bet signals, which indicate the sum which can be bet. The output of the block is a verification signal that states if the numbers will further generate or not on the next spin.

1. *Hex to bcd block*

The Hexadecimal to BCD block is a crucial block in showing the user how much money is in the account, based on a switch signal. The idea of this block is to use a known algorithm which is widely used in this kind of conversion. The fundamental aspect is that it shift lefts the number given in binary and then checks a certain digit if its over or under 4, in which case a ‘3’ is not added, if the number is lower than ‘4’. The block basically contains registers and shift registers that are used to simulate this algorithm in a physical way.

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated

1. *7 segment block*

The last block of the project is the seven-segment display block which uses the 7segment decoder and using a clocked process, it goes through the 4 available anodes of the board.

The input number of the block is a number given by a multiplexor which chooses between the spinning numbers and the sum of the account, given the low number of 7segments.

A screenshot of a computer program

Description automatically generated

1. Notations and explications

Now that all the components and the blocks were described, a brief explanation of the way the components are connected is given.

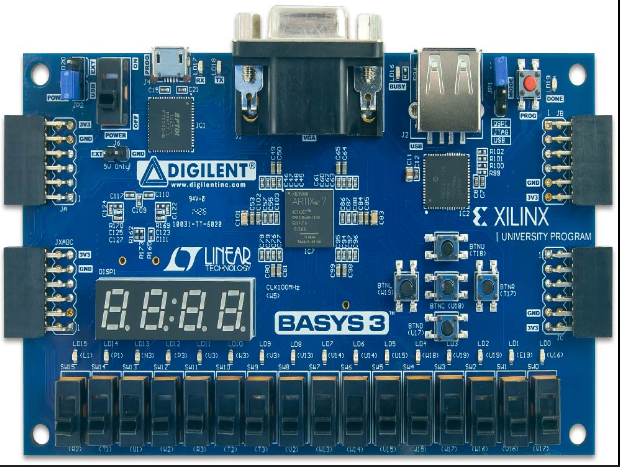
A screenshot of a computer code

Description automatically generated

In the above image, all the components are ported into a top module. A signal “not\_play” represents the inverted play\_btn which is further used in some of the components.

Now, the way the components are connected together, they work based on the block diagram.The timing module sends the frequencies to the 3 counters which generate the numbers. A 16-bit number “number\_to\_show” will contain all the numbers. The upper part of this number is set to “1111” in order to be decoded to nothing to be shown on the 7segment. The ram contains the “moneyout” signal which is the amount of money in the memory and the “moneywl” which represents the signal that holds the value of the win-calculator block after a loose or win. The moneyout is also used in the win-computations block. The comparator uses the 3 generated numbers “q\_out1/2/3” and the “verif” signal that tells further blocks if a winning combination was received. The binary to bcd block holds only the value in the memory for easier reading of the user. The error block holds the money in the memory, the bet input and the verification output for the other components. The Mux components are used:mux for choosing either between generating the numbers or showing the usm of the account based on the “sel” input and the mux2 which either sends further the numbers or an error number; “badnr” which is basically “EEE” to the input of the first mux. By combining all these, the desired behaviour is given.

1. Usage on the FPGA board

For the simulation of the project, a Basys 3 FPGA is going to be used. Further, an image with all the inputs and outputs is going to be given.

Play Button

Bet signal

LEDs that signal a win

Selection signal

7segment displays

The user starts the slot machine by moving up the play button switch, then on the 7segment display, the random numbers are generated. After, the user can choose when to depress the switch in order to obtain the generated numbers at a certain time. The user can also choose between bet amount with the 2 bet switches. The user also has the selection switch which can be used only after a spin, (=when the play button is set down) in order to see the amount of money left. The LEDs will continuously light up when a winning combination is stopped on.

1. Conclusions and further developments

Now, the project was presented, step by step using the FPGA board, but it can be further made using physical components to better simulate the idea behind this type of slot. More segments can be added and overall, the project can be ported to a bigger slot machine. On top of that, the numbers can be ported onto a screen which could be connected to the board and then transformed into symbols to better simulate a more complex slot machine.