COL 215 Lab Project First Progress Report

7up 7down

Made By: Sankalan Pal Chowdhury(2016CS10701) and
Shresth Tuli(2016CS10680)

CODE

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.CONV_STD_LOGIC_VECTOR;
use IEEE.NUMERIC_STD.ALL;


entity Project_7u7d is
  Port (
  clk : IN STD_LOGIC;
  bid1 : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
  bid2 : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
  win1 : INOUT STD_LOGIC;
  win2 : INOUT STD_LOGIC;
  bid1_invalid : INOUT STD_LOGIC;
  bid2_invalid : INOUT STD_LOGIC;
  bid1_invalid_25 : OUT STD_LOGIC;
  bid2_invalid_25 : OUT STD_LOGIC;
  bid1_invalid_high : OUT STD_LOGIC;
  bid2_invalid_high : OUT STD_LOGIC;
  cathode : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
  anode : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
  test : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
  init : IN STD_LOGIC:='0'
    );
end Project_7u7d;

architecture Behavioral of Project_7u7d is
    SIGNAL p1_bid : STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL p2_bid : STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL p1_money : STD_LOGIC_VECTOR(9 DOWNTO 0):="1111110000";
    SIGNAL p2_money : STD_LOGIC_VECTOR(9 DOWNTO 0):="1111110000";
    SIGNAL tax : STD_LOGIC_VECTOR(9 DOWNTO 0);
    SIGNAL ssd : STD_LOGIC_VECTOR(15 DOWNTO 0):="0000101100011101";
    SIGNAL random, random1 : STD_LOGIC_VECTOR(3 DOWNTO 0);
    SIGNAL cnt : std_logic_vector(5 downto 0):="101001";
    SIGNAL state : STD_LOGIC_VECTOR(2 DOWNTO 0):="000";
    SIGNAL counter, cycle : INTEGER:=0;
    SIGNAL one_bid_done : STD_LOGIC:='0';
    SIGNAL p1p2 : STD_LOGIC_VECTOR(1 DOWNTO 0);
begin

Display:
ENTITY work.lab4_seven_segment_display
port map(
    clk=>clk,
    cathode=>cathode,
    anode=>anode,
    b=>ssd
    );


process(clk)
begin
random_number_generator:
    if(clk='1' and clk'event) then
            if(init='1') then
            cnt(2 downto 0)<=std_logic_vector(unsigned(cnt(2 downto 0))+1);
                if(cnt(2 downto 0)="110") then
                    cnt(2 downto 0)<="001";
                    cnt(5 downto 3)<=std_logic_vector(unsigned(cnt(5 downto 3))+1);
                    if(cnt(5 downto 3)="110") then
                        cnt(5 downto 3)<="001";
                    end if;
                end if;
            else
                random1<=std_logic_vector(unsigned('0' & cnt(5 downto 3))+unsigned('0' & cnt(2
downto 0)));
            end if;
```

```
Control:
    --P1 bids
    if state = "000" then
        p1p2 <= "10";
        ssd <= "1011000111010001";
        p1_bid <= bid1;
        win1<='0';
        win2<='0';
        counter <= counter + 1;
        if(unsigned(p1_bid(6 DOWNTO 0))>unsigned(p1_money)/4) then bid1_invalid<='1';
bid1_invalid_25<='1'; else bid1_invalid<='0'; bid1_invalid_25<='0'; end if;
        if(unsigned(p1_bid(6 DOWNTO 0))>unsigned(p2_money)) then bid1_invalid<='1';
bid1_invalid_high<='1'; else bid1_invalid<='0'; bid1_invalid_high<='0'; end if;
        if(counter>100000000 and init='1' and bid1_invalid='0') then
            tax <= STD_LOGIC_VECTOR(unsigned(tax) + unsigned(p1_bid(6 DOWNTO 0))/4);
            p1_money <= STD_LOGIC_VECTOR(unsigned(p1_money) - unsigned(p1_bid(6 DOWNTO 0))/4);
            if(one_bid_done='0') then state <= "001"; one_bid_done <= '1'; counter<=0; else state
<= "010"; one_bid_done <= '0'; counter<=0; end if;
        end if;

   --P2 bids
   elsif state = "001" then
        p1p2 <= "01";
        ssd <= "1011000111010010";
        p2_bid <= bid2;
        win1<='0';
        win2<='0';
        counter <= counter + 1;
        if(unsigned(p2_bid(6 DOWNTO 0))>unsigned(p2_money)/4) then bid2_invalid<='1';
bid2_invalid_25<='1'; else bid2_invalid<='0'; bid2_invalid_25<='0'; end if;
        if(unsigned(p2_bid(6 DOWNTO 0))>unsigned(p1_money)) then bid2_invalid<='1';
bid2_invalid_high<='1'; else bid2_invalid<='0'; bid2_invalid_high<='0'; end if;
        if(counter>100000000 and init='1' and bid2_invalid='0') then
            tax <= STD_LOGIC_VECTOR(unsigned(tax) + unsigned(p2_bid(6 DOWNTO 0))/4);
            p2_money <= STD_LOGIC_VECTOR(unsigned(p2_money) - unsigned(p1_bid(6 DOWNTO 0))/4);
            if(one_bid_done='0') then state <= "000"; one_bid_done <= '1'; counter<=0; else state
<= "010"; one_bid_done <= '0'; counter<=0; end if;
        end if;

   --Take random number
   elsif state = "010" then
        if(init='0') then
            random <= random1;
            state <= "011";
        end if;

   --Calculations of money of each w.r.t random number
   elsif state = "011" then
        p1p2 <= "00";
        counter<=0;
        if(unsigned(random)>7 and p1_bid(7)='1' and p2_bid(7)='0') then
            p1_money <= STD_LOGIC_VECTOR(unsigned(p1_money)+unsigned(p1_bid(6 DOWNTO 0)));
            p2_money <= STD_LOGIC_VECTOR(unsigned(p2_money)-unsigned(p1_bid(6 DOWNTO 0)));
            win1<='1'; win2<='0';
        elsif(unsigned(random)>7 and p2_bid(7)='1' and p1_bid(7)='0') then
            p2_money <= STD_LOGIC_VECTOR(unsigned(p2_money)+unsigned(p2_bid(6 DOWNTO 0)));
            p1_money <= STD_LOGIC_VECTOR(unsigned(p1_money)-unsigned(p2_bid(6 DOWNTO 0)));
            win1<='0'; win2<='1';
        elsif(unsigned(random)>7 and p2_bid(7)='1' and p1_bid(7)='1') then
            p2_money <= STD_LOGIC_VECTOR(unsigned(p2_money)+unsigned(p2_bid(6 DOWNTO 0))-
unsigned(p1_bid(6 DOWNTO 0)));
            p1_money <= STD_LOGIC_VECTOR(unsigned(p1_money)+unsigned(p1_bid(6 DOWNTO 0))-
unsigned(p2_bid(6 DOWNTO 0)));
            win1<='1'; win2<='1';
        elsif(unsigned(random)<7 and p1_bid(7)='0' and p2_bid(7)='1') then
            p1_money <= STD_LOGIC_VECTOR(unsigned(p1_money)+unsigned(p1_bid(6 DOWNTO 0)));
            p2_money <= STD_LOGIC_VECTOR(unsigned(p2_money)-unsigned(p1_bid(6 DOWNTO 0)));
            win1<='1'; win2<='0';
        elsif(unsigned(random)<7 and p2_bid(7)='0' and p1_bid(7)='1') then
            p2_money <= STD_LOGIC_VECTOR(unsigned(p2_money)+unsigned(p2_bid(6 DOWNTO 0)));
```

```
            p1_money <= STD_LOGIC_VECTOR(unsigned(p1_money)-unsigned(p2_bid(6 DOWNTO 0)));
            win1<='0'; win2<='1';
        elsif(unsigned(random)<7 and p2_bid(7)='0' and p1_bid(7)='0') then
            p2_money <= STD_LOGIC_VECTOR(unsigned(p2_money)+unsigned(p2_bid(6 DOWNTO 0))-
unsigned(p1_bid(6 DOWNTO 0)));
            p1_money <= STD_LOGIC_VECTOR(unsigned(p1_money)+unsigned(p1_bid(6 DOWNTO 0))-
unsigned(p2_bid(6 DOWNTO 0)));
            win1<='1'; win2<='1';
        elsif(unsigned(random)=7 and unsigned(p1_bid)>unsigned(p2_bid)) then
            p1_money <= STD_LOGIC_VECTOR(unsigned(p1_money)+unsigned(p1_bid(6 DOWNTO 0)));
            p2_money <= STD_LOGIC_VECTOR(unsigned(p2_money)-unsigned(p1_bid(6 DOWNTO 0)));
            win1<='1'; win2<='0';
        elsif(unsigned(random)=7 and unsigned(p2_bid)>unsigned(p1_bid)) then
            p2_money <= STD_LOGIC_VECTOR(unsigned(p2_money)+unsigned(p2_bid(6 DOWNTO 0)));
            p1_money <= STD_LOGIC_VECTOR(unsigned(p1_money)-unsigned(p2_bid(6 DOWNTO 0)));
            win1<='0'; win2<='1';
        end if;
        state <= "100";

    --Display
    elsif state = "100" then
        counter <= counter+1;
        if(counter<200000000) then --Display random number
            ssd(15 DOWNTO 8) <= "00000000";
            ssd(7 DOWNTO 4) <= STD_LOGIC_VECTOR(unsigned(random)/10);
            ssd(3 DOWNTO 0) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(random)) -
10*to_integer(unsigned(random)/10),4);
        elsif(counter<400000000) then --Display who won
            if(win1='1' and win2='1') then ssd<="0000000100000010";
            elsif(win1='0' and win2='0') then ssd<="0000000000000000";
            elsif(win1='1' and win2='0') then ssd<="0000000000000001";
            elsif(win1='0' and win2='0') then ssd<="0000000000000010";
            end if;
        elsif(counter<600000000) then --Display player 1 money
            ssd(15 DOWNTO 12) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(p1_money)/1000),4);
            ssd(11 DOWNTO 8) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(p1_money)/100) -
10*to_integer(unsigned(p1_money)/1000),4);
            ssd(7 DOWNTO 4) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(p1_money)/10) -
10*to_integer(unsigned(p1_money)/100),4);
            ssd(3 DOWNTO 0) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(p1_money)) -
10*to_integer(unsigned(p1_money)/10),4);
        elsif(counter<800000000) then --Display player 2 money
            ssd(15 DOWNTO 12) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(p2_money)/1000),4);
            ssd(11 DOWNTO 8) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(p2_money)/100) -
10*to_integer(unsigned(p2_money)/1000),4);
            ssd(7 DOWNTO 4) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(p2_money)/10) -
10*to_integer(unsigned(p2_money)/100),4);
            ssd(3 DOWNTO 0) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(p2_money)) -
10*to_integer(unsigned(p2_money)/10),4);
        elsif(counter<1000000000) then --Display tax
            ssd(15 DOWNTO 12) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(tax)/1000),4);
            ssd(11 DOWNTO 8) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(tax)/100) -
10*to_integer(unsigned(tax)/1000),4);
            ssd(7 DOWNTO 4) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(tax)/10) -
10*to_integer(unsigned(tax)/100),4);
            ssd(3 DOWNTO 0) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(tax)) -
10*to_integer(unsigned(tax)/10),4);
        elsif(counter=1000000000) then
            if(cycle mod 2 = 1) then state<="000"; else state<="001"; end if; cycle<=cycle+1;
            if(p1_money = "0000000000") then
                p1_money <= STD_LOGIC_VECTOR(unsigned(p1_money) + unsigned(tax));
                state<="101"; counter<=0; end if;
            if(p1_money = "0000000000") then
                p2_money <= STD_LOGIC_VECTOR(unsigned(p2_money) + unsigned(tax));
                state<="101"; counter<=0; end if;
        end if;

    --Display Winner and reset for new game
    elsif state = "101" then
        counter <= counter+1;
        if(p1_money = "0000000000") then
```

```
            if counter < 200000000 then ssd <= "0010001000100010"; end if;
            if counter < 400000000 then
                ssd(15 DOWNTO 12) <=
CONV_STD_LOGIC_VECTOR(to_integer(unsigned(p1_money)/1000),4);
                ssd(11 DOWNTO 8) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(p1_money)/100) -
10*to_integer(unsigned(p1_money)/1000),4);
                ssd(7 DOWNTO 4) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(p1_money)/10) -
10*to_integer(unsigned(p1_money)/100),4);
                ssd(3 DOWNTO 0) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(p1_money)) -
10*to_integer(unsigned(p1_money)/10),4);
            end if;
            if counter < 600000000 then ssd <= "0000000000000000"; end if;
            if counter = 600000000 then
                ssd <= "1011000111010001";
                state <= "000"; counter<=0; cycle<=0;
                p1_money<="1111110000";
                p2_money<="1111110000";
            end if;
        elsif(p2_money = "0000000000") then
            if counter < 200000000 then ssd <= "0001000100010001"; end if;
            if counter < 400000000 then
                ssd(15 DOWNTO 12) <=
CONV_STD_LOGIC_VECTOR(to_integer(unsigned(p1_money)/1000),4);
                ssd(11 DOWNTO 8) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(p1_money)/100) -
10*to_integer(unsigned(p1_money)/1000),4);
                ssd(7 DOWNTO 4) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(p1_money)/10) -
10*to_integer(unsigned(p1_money)/100),4);
                ssd(3 DOWNTO 0) <= CONV_STD_LOGIC_VECTOR(to_integer(unsigned(p1_money)) -
10*to_integer(unsigned(p1_money)/10),4);
            end if;
            if counter < 600000000 then ssd <= "0000000000000000"; end if;
            if counter = 600000000 then
                ssd <= "1011000111010001";
                state <= "000"; counter<=0; cycle<=0;
                p1_money<="1111110000";
                p2_money<="1111110000";
            end if;
        end if;
    end if;

    end if;


END PROCESS;

end Behavioral;
```

XDC

```
## This file is a general .xdc for the Basys3 rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level signal
names in the project

# Clock signal
#Bank = 34, Pin name = ,                   Sch name = CLK100MHZ
#       set_property PACKAGE_PIN W5 [get_ports CLK]
#       set_property IOSTANDARD LVCMOS33 [get_ports CLK]
#       create_clock -period 10.000 -name sys_clk_pin -waveform {0.000 5.000} -add [get_ports
CLK]

create_clock -period 10.000 -name sys_clk_pin -waveform {0.000 5.000} -add [get_ports clk]
# Switches
set_property PACKAGE_PIN R2 [get_ports bid1[7]]
set_property IOSTANDARD LVCMOS33 [get_ports bid1[7]]
set_property PACKAGE_PIN T1 [get_ports bid1[6]]
set_property IOSTANDARD LVCMOS33 [get_ports bid1[6]]
set_property PACKAGE_PIN U1 [get_ports bid1[5]]
set_property IOSTANDARD LVCMOS33 [get_ports bid1[5]]
set_property PACKAGE_PIN W2 [get_ports bid1[4]]
set_property IOSTANDARD LVCMOS33 [get_ports bid1[4]]
set_property PACKAGE_PIN R3 [get_ports bid1[3]]
set_property IOSTANDARD LVCMOS33 [get_ports bid1[3]]
set_property PACKAGE_PIN T2 [get_ports bid1[2]]
set_property IOSTANDARD LVCMOS33 [get_ports bid1[2]]
set_property PACKAGE_PIN T3 [get_ports bid1[1]]
set_property IOSTANDARD LVCMOS33 [get_ports bid1[1]]
set_property PACKAGE_PIN V2 [get_ports bid1[0]]
set_property IOSTANDARD LVCMOS33 [get_ports bid1[0]]
set_property PACKAGE_PIN W13 [get_ports bid2[7]]
set_property IOSTANDARD LVCMOS33 [get_ports bid2[7]]
set_property PACKAGE_PIN W14 [get_ports bid2[6]]
set_property IOSTANDARD LVCMOS33 [get_ports bid2[6]]
set_property PACKAGE_PIN V15 [get_ports bid2[5]]
set_property IOSTANDARD LVCMOS33 [get_ports bid2[5]]
set_property PACKAGE_PIN W15 [get_ports bid2[4]]
set_property IOSTANDARD LVCMOS33 [get_ports bid2[4]]
set_property PACKAGE_PIN W17 [get_ports bid2[3]]
set_property IOSTANDARD LVCMOS33 [get_ports bid2[3]]
set_property PACKAGE_PIN W16 [get_ports bid2[2]]
set_property IOSTANDARD LVCMOS33 [get_ports bid2[2]]
set_property PACKAGE_PIN V16 [get_ports bid2[1]]
set_property IOSTANDARD LVCMOS33 [get_ports bid2[1]]
set_property PACKAGE_PIN V17 [get_ports bid2[0]]
set_property IOSTANDARD LVCMOS33 [get_ports bid2[0]]

set_property PACKAGE_PIN U18 [get_ports init]
set_property IOSTANDARD LVCMOS33 [get_ports init]


set_property PACKAGE_PIN W5 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]


# LEDs

set_property PACKAGE_PIN W18 [get_ports test[1]]
set_property IOSTANDARD LVCMOS33 [get_ports test[1]]
set_property PACKAGE_PIN V19 [get_ports test[0]]
set_property IOSTANDARD LVCMOS33 [get_ports test[0]]

set_property PACKAGE_PIN U2 [get_ports anode[0]]
set_property IOSTANDARD LVCMOS33 [get_ports anode[0]]
set_property PACKAGE_PIN U4 [get_ports anode[1]]
set_property IOSTANDARD LVCMOS33 [get_ports anode[1]]
```

```
set_property PACKAGE_PIN V4 [get_ports anode[2]]
set_property IOSTANDARD LVCMOS33 [get_ports anode[2]]
set_property PACKAGE_PIN W4 [get_ports anode[3]]
set_property IOSTANDARD LVCMOS33 [get_ports anode[3]]

set_property PACKAGE_PIN W7 [get_ports cathode[0]]
set_property IOSTANDARD LVCMOS33 [get_ports cathode[0]]
set_property PACKAGE_PIN W6 [get_ports cathode[1]]
set_property IOSTANDARD LVCMOS33 [get_ports cathode[1]]
set_property PACKAGE_PIN U8 [get_ports cathode[2]]
set_property IOSTANDARD LVCMOS33 [get_ports cathode[2]]
set_property PACKAGE_PIN V8 [get_ports cathode[3]]
set_property IOSTANDARD LVCMOS33 [get_ports cathode[3]]
set_property PACKAGE_PIN U5 [get_ports cathode[4]]
set_property IOSTANDARD LVCMOS33 [get_ports cathode[4]]
set_property PACKAGE_PIN V5 [get_ports cathode[5]]
set_property IOSTANDARD LVCMOS33 [get_ports cathode[5]]
set_property PACKAGE_PIN U7 [get_ports cathode[6]]
set_property IOSTANDARD LVCMOS33 [get_ports cathode[6]]

set_property PACKAGE_PIN L1 [get_ports bid1_invalid]
set_property IOSTANDARD LVCMOS33 [get_ports bid1_invalid]
set_property PACKAGE_PIN P1 [get_ports bid1_invalid_25]
set_property IOSTANDARD LVCMOS33 [get_ports bid1_invalid_25]
set_property PACKAGE_PIN N3 [get_ports bid1_invalid_high]
set_property IOSTANDARD LVCMOS33 [get_ports bid1_invalid_high]
set_property PACKAGE_PIN V13 [get_ports win1]
set_property IOSTANDARD LVCMOS33 [get_ports win1]

set_property PACKAGE_PIN V14 [get_ports bid2_invalid]
set_property IOSTANDARD LVCMOS33 [get_ports bid2_invalid]
set_property PACKAGE_PIN U19 [get_ports bid2_invalid_25]
set_property IOSTANDARD LVCMOS33 [get_ports bid2_invalid_25]
set_property PACKAGE_PIN E19 [get_ports bid2_invalid_high]
set_property IOSTANDARD LVCMOS33 [get_ports bid2_invalid_high]
set_property PACKAGE_PIN U16 [get_ports win2]
set_property IOSTANDARD LVCMOS33 [get_ports win2]



# Others (BITSTREAM, CONFIG)
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.SPI_BUSWIDTH 4 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]

set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]

set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]
```

CODE DESCRIPTION

The code is divided into 3 parts:

1. Random Number Generator
2. Control
3. Display

Display

The Seven Segment Display has been imported as a component from the code developed for Lab 4. The *ssd* signal used in the architecture is corresponds to the 16-bit input for the Lab 4 Module. The other ports including *cathode*, *anode* and *clk* (clock) have been mapped with this component.

Inside the Process (sensitive on *clk*), there are two parts: Random Number Generator and Control.

Random Number Generator

The random number generator relies on the noise in the time for which the *init* button is pressed. Since this would be manually done, the resolution cannot be more than ~100Hz(refer to this). If measured to the accuracy of 10^{-8} seconds, we can assume the last few significant figures to be random and uniformly distributed. We use two counters, both reseting at 6 to 1. One of these get incremented at every clock cycle(when *init* is high) and the other gets incremented only when the first goes from 6 to 1. These two independently simulate two dices. When *init* becomes low, the sum of the numbers on the two dices gets fed to the signal *random1.* In this way, *random1* gets a new (random) value whenever there is a pulse on *init*. When needed, it is stored into another signal *random* and used.

Control

In the Control part of the code most of the ASM functionalities have been achieved. The overall Control block has been divided into 5 states (the buffer states in ASM have been combined for ease of code, but may be separated later; if required).

1. State 1:

   The *state* signal has been used to indicates the state of the system. The first state corresponding to *state* = “000”. The signal *p1p2* shows which one of the two players needs to bid. As per the ASM the player who bids first alternates between the two players. “10” corresponds to player 1 bidding first and “01” corresponds to player 2 bidding. In state 1, the signal *p1_bid* stores the bid amount of player 1. The 7 least significant bits of *bid1* represent value of bid from 0 to 127 and *bid(7)* = *p1_bid(7)* represents if the bid is for >7 or <7.

Also, *win1* and *wins2* are two signals which store the winning player who won the current bid. They have to be '0' till the bidding process is not complete. *Bid1_invalid*, *bid1_invalid_25* and *bid1_invalid_high* are three signals which show if the bid set by player 1 is valid or not. *Bid1_invalid_25* is '1' if the player does not have 25% of the bid value in his purse, *bid1_invalid_high* is '1' if the bid value is higher than the money in other player's purse and *bid1_invalid* is '1' in either of the two cases. The bid is not accepted if the *bid1_invalid* is high.

When the player 1 has done bidding and bid is valid and he presses the *init* button then 25% of bid value is deducted from his purse *p1_money* (1008 initially) and added to *tax*. The signal *one_bid_done* is to store if player 1 had done first or player 2. If player 1 is first then *one_bid_done* is '0' so goes to state 2 , otherwise if *one_bid_done* is '1' then player 1 is the second bidder hence goes to state 3. The *counter* for timing is reset to 0.

2. State 2:
   This state is for player 2 bidding similar to that of player 1 as in state 1, but here the *p1p2* signal is "01" and not "10".

3. State 3:
   Stores the random generated for comparisons.

4. State 4:
   This state calculates the money in each of the two purses p1_money and p2_money as per the Game Description (in Design Document 1) and also shows the player winning the current bid.

5. State 5:
   This state displays the following sequentially each for 2 seconds:
   - Random Number 2 to 12
   - Which player win the current bid: 1 or 2 or both or none
   - Money left with Player 1
   - Money left with player 2
   - Tax

6. State 6:
   This state displays the winner for 2 seconds then his/her money then resets all signals and starts new game. At start the first bidding player is player 1.