

Embedded Systems 1

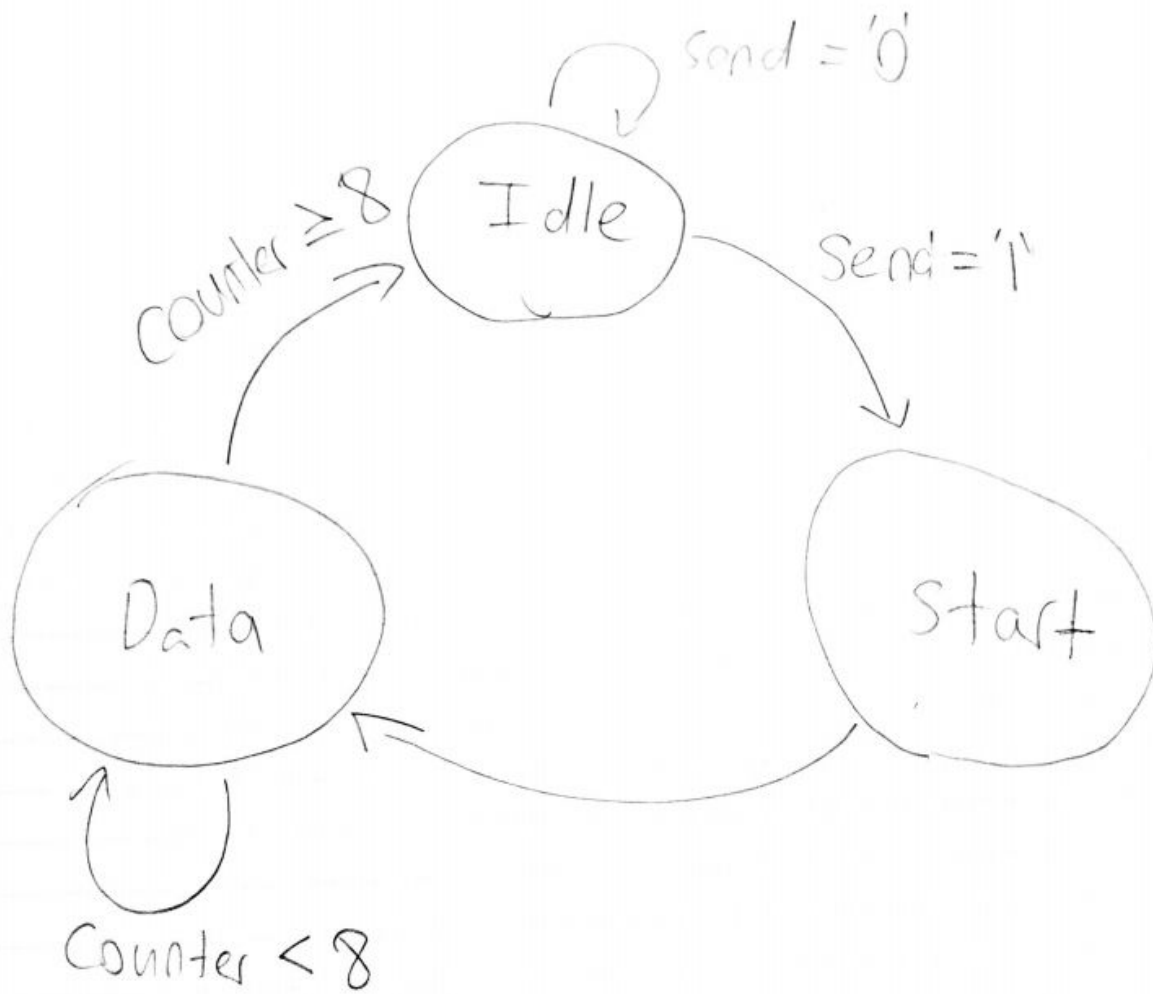
Lab Report 3

Gavin McKim

3/28/2019

Pre Lab:

Pre Lab 3



Purpose:

The main purpose of the lab was to introduce the idea of Finite State Machines(FSM). FSM are computational models that utilize “states”. The program performs certain actions depending on the state it is currently in. The state changes on clock ticks when certain conditions are met. The concept of FSM was demonstrated by creating a basic Universal Asynchronous Receiver Transmitter(UART). A UART uses FSM to transmit data between two machines.

1. We Can Rebuild Him

Theory:

The first part of the lab involved completing the UART. The lab provided the final design of the UART and the receiver aspect of it. The first part had us create the transmitter side of the UART. This transmitter is triggered when the “line” in the UART goes low. When that happens the transmitter sends an 8-bit character one bit at a time. It does this using FSM. When the transmitter is triggered it goes in to a “start” state where it prepares to send the character. Then it transitions into the “data” state where it sends the character one bit at a time. When run with the provided test bench, it should show the correct hexadecimal values in charRec.

VHDL Code:

```

1
2 library IEEE;
3 use IEEE.STD_LOGIC_1164.ALL;
4 use IEEE.NUMERIC_STD.ALL;
5
6 -- Uncomment the following library declaration if instantiating
7 -- any Xilinx leaf cells in this code.
8 --library UNISIM;
9 --use UNISIM.VComponents.all;
10
11 entity uart_tx is
12 port (
13     clk , en , send , rst : in std_logic;
14     char : in std_logic_vector (7 downto 0);
15     ready , tx : out std_logic );
16 end uart_tx ;
17
18 architecture Behavioral of uart_tx is
19
20     -- state type enumeration and state variable
21     type state is (idle, start, data);
22     signal curr : state := idle;
23
24     signal temp : std_logic_vector (7 downto 0);
25     signal counter : std_logic_vector (3 downto 0);
26 begin
27     process(clk) begin
28         if rising_edge(clk) then
29             if(rst = '1') then
30                 temp <= (others => '0');
31                 counter <= (others => '0');
32                 curr <= idle;
33                 tx <= '1';
34                 ready <= '1';
35             end if;
36
37
38
39             if(en = '1') then
40                 case curr is
41
42                     when idle =>
43                         if(send = '1') then
44                             temp <= char;
45                             curr <= start;
46                             ready <= '0';
47                         else
48                             tx <= '1';
49                             ready <= '1';
50                         end if;
51
52                     when start =>
53                         tx <= '0';
54                         counter <= "0000";
55                         curr <= data;

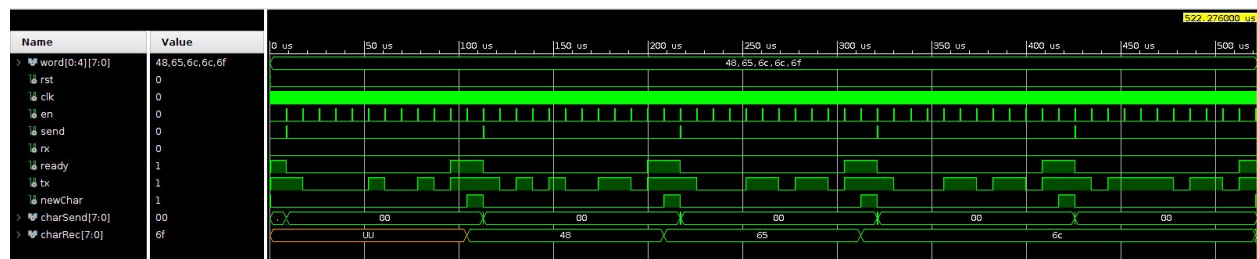
```

```

56 :
57 :
58 :         when data =>
59 :             if(unsigned(counter) < 8) then
60 :                 tx <= temp(to_integer(unsigned(counter)));
61 :                 counter <= std_logic_vector(unsigned(counter) + 1);
62 :
63 :             else
64 :                 tx <= '1';
65 :                 ready <= '1';
66 :                 curr <= idle;
67 :             end if;
68 :
69 :         end case;
70 :
71 :     end if;
72 : end if;
73 :
74 : end process;
75 :
76 : end Behavioral;
77 :

```

Simulation



2. We Can Rebuild Him

Theory

For the next part of the lab we create a program to test sending the computer information from the board. This program holds a four character array that contains my RUID(gqm4). The sender program sends this RUID utilizing a single FSM. It defaults in the idle state and when the button is pressed while the system is ready, it outputs one character of the RUID. Then it cycles through a few “busy” states. When the button is unpressed it goes from the last busy state back to idle. When it reaches the end of the RUID it starts over. This was the implemented in a top level design along with a clock divider, debouncers, and the uart. This top level design would be implemented onto a zybo board along with a PMOD JB. After using the console to connect the devices, the Zybo’s buttons should control the program. When one button is pressed it should send only one character of my RUID until reaching the end at which point it starts over. If the reset button is pressed it should reset back to the first character in my RUID.

VHDL Code

Sender

```

1
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use IEEE.NUMERIC_STD.ALL;
5
6
7  entity sender is
8      Port ( rst, clk, en, btn, ready : in std_logic;
9             send : out std_logic;
10            char : out std_logic_vector(7 downto 0));
11  end sender;
12
13  architecture Behavioral of sender is
14      type my_array is array (0 to 3) of std_logic_vector(7 downto 0);
15      type state is (idle, busyA, busyB, busyC);
16
17      signal NETID : my_array;
18      signal curr : state := idle;
19      signal i : std_logic_vector(3 downto 0) := (others => '0');
20
21
22
23  begin
24      NETID(3) <= x"67";
25      NETID(2) <= x"71";
26      NETID(1) <= x"6D";
27      NETID(0) <= x"34";
28
29
30  process(clk) begin
31  if rising_edge(clk) and en = '1' then
32
33  if(rst = '1') then
34      send <= '0';
35      char <= (others => '0');
36      i <= (others => '0');
37      curr <= idle;
38
39  else
40  case curr is
41  when idle =>
42  if(ready = '1' and btn = '1') then
43  if (unsigned(i) < 4) then
44      send <= '1';
45      char <= NETID(to_integer(unsigned(i)));
46      i <= std_logic_vector( unsigned(i) + 1 );
47      curr <= busyA;
48  else
49      i <= (others => '0');
50  end if;
51  end if;
52
53  when busyA =>
54  curr <= busyB;
55
56  when busyB =>
57  send <= '0';
58  curr <= busyC;
59
60  when busyC =>
61  if(ready = '1' and btn = '0') then
62  curr <= idle;

```

```
63 :         else
64 :             curr <= busyC;
65 ⊞         end if;
66 :
67 ⊞ end case;
68 ⊞ end if;
69 ⊞ end if;
70 ⊞ end process;
71 :
72 :
73 :
74 ⊞ end Behavioral;
```

Clock Divider


```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  -- Uncomment the following library declaration if using
6  -- arithmetic functions with Signed or Unsigned values
7  --use IEEE.NUMERIC_STD.ALL;
8
9  -- Uncomment the following library declaration if instantiating
10 -- any Xilinx leaf cells in this code.
11 --library UNISIM;
12 --use UNISIM.VComponents.all;
13
14 entity clock_div is
15 port (
16     clk_in : in std_logic;
17     div : out std_logic
18 );
19 end clock_div;
20
21 architecture cnt of clock_div is
22     signal count : std_logic_vector (25 downto 0) := (others => '0');
23 begin
24     process(clk_in) begin
25         if rising_edge(clk_in) then
26             count <= std_logic_vector( unsigned(count) + 1);
27
28             if ( unsigned(count) = 1085) then
29                 div <= '1';
30                 count <= (others => '0');
31
32             else
33                 div <= '0';
34             end if;
35
36         end if;
37     end process;
38
39 end cnt;
40
41 library ieee;
42 use ieee.std_logic_1164.all;
43 use ieee.numeric_std.all;
44
45 entity divider is
46     Port ( clk : in std_logic;
47           Q : out std_logic);
48 end divider;

```

```

40
41 library ieee;
42 use ieee.std_logic_1164.all;
43 use ieee.numeric_std.all;
44
45 entity divider is
46     Port ( clk : in std_logic;
47           Q : out std_logic);
48 end divider;
49
50 architecture Behavioral of divider is
51
52 component clock_div is
53 port (
54     clk_in : in std_logic;
55     div : out std_logic
56 );
57 end component;
58
59 signal D, CE : std_logic := '0';
60
61 begin
62
63 di: clock_div
64 port map ( clk_in => clk,
65           div => CE);
66
67 reg: process(clk)
68 begin
69     if (rising_edge(clk)) then
70         if(CE = '1') then
71             D <= not D;
72         end if;
73     end if;
74 end process;
75     Q <= D;
76
77 end Behavioral;
78 !

```

Top Level Design

```

1
2
3 library IEEE;
4 use IEEE.STD_LOGIC_1164.ALL;
5 use IEEE.NUMERIC_STD.ALL;
6
7
8 entity top_level is
9     Port (txd, clk : in std_logic;
10         btn : in std_logic_vector(1 downto 0);
11         rxd : out std_logic;
12         cts : out std_logic := '0';
13         rts : out std_logic := '0' );
14 end top_level;
15
16 architecture Behavioral of top_level is
17
18     component debouncer is
19         Port ( clkd, bt : in std_logic;
20             debounce: out std_logic);
21     end component;
22
23     component divider is
24         Port ( clk : in std_logic;
25             Q : out std_logic);
26     end component;
27
28     component sender is
29         Port ( rst, clk, en, btn, ready : in std_logic;
30             send : out std_logic;
31             char : out std_logic_vector(7 downto 0));
32     end component;
33
34     component uart is
35     port (
36         clk , en , send , rx , rst : in std_logic ;
37         charSend : in std_logic_vector (7 downto 0) ;
38         ready , tx , newChar : out std_logic ;
39         charRec : out std_logic_vector (7 downto 0)
40     ) ;
41 end component;
42
43 signal dbnc : std_logic_vector(1 downto 0);
44 signal c : std_logic_vector(7 downto 0);
45 signal s, r, div : std_logic;
46
47 begin
48
49     u1: debouncer
50     port map ( bt => btn(0),
51         clkd => clk,
52         debounce => dbnc(0));
53
54     u2: debouncer
55     port map ( bt => btn(1),
56         clkd => clk,
57         debounce => dbnc(1));
58
59     u3: divider
60     port map ( Q => div,
61         clk => clk );
62

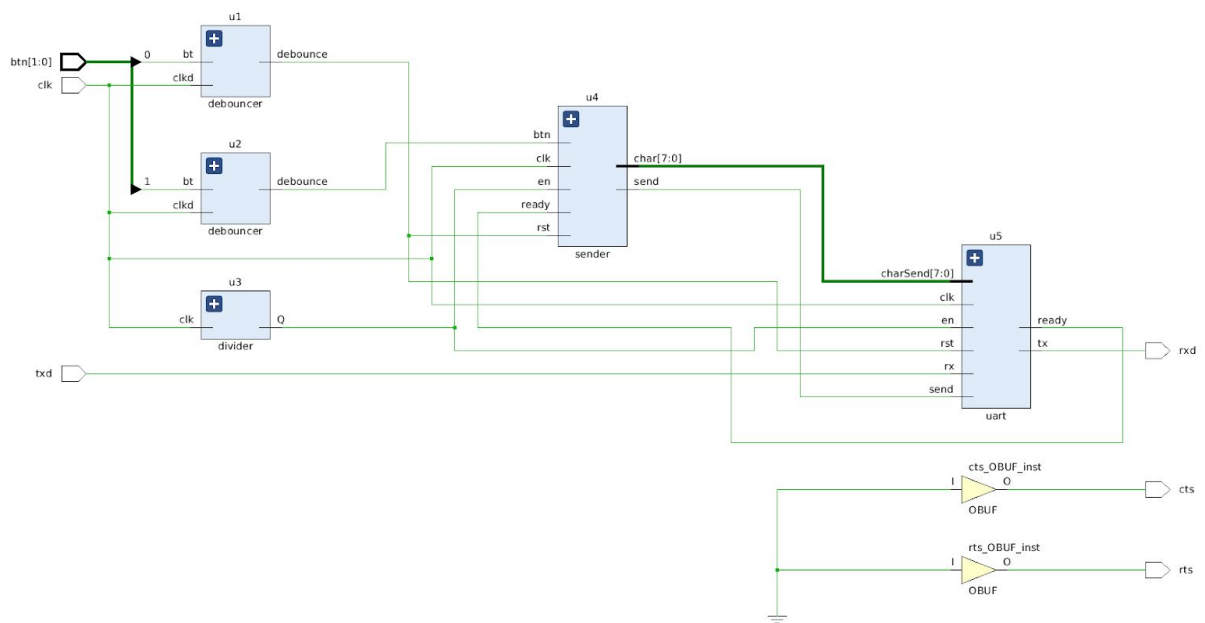
```

```

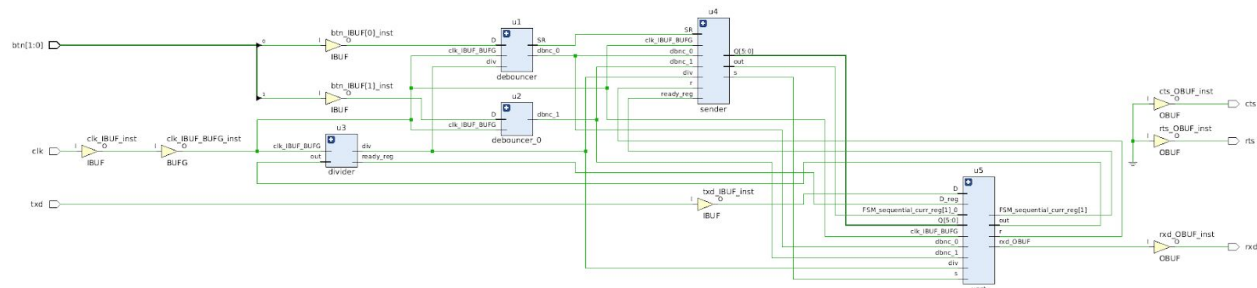
63 u4: sender
64 port map ( btn => dbnc(1),
65             clk => clk,
66             en => div,
67             ready => r,
68             rst => dbnc(0),
69             send => s,
70             char => c);
71
72 u5: uart
73 port map ( charSend => c,
74             clk => clk,
75             en => div,
76             rst => dbnc(0),
77             rx => txd,
78             send => s,
79             tx => rxd,
80             ready => r );
81 end Behavioral;
82

```

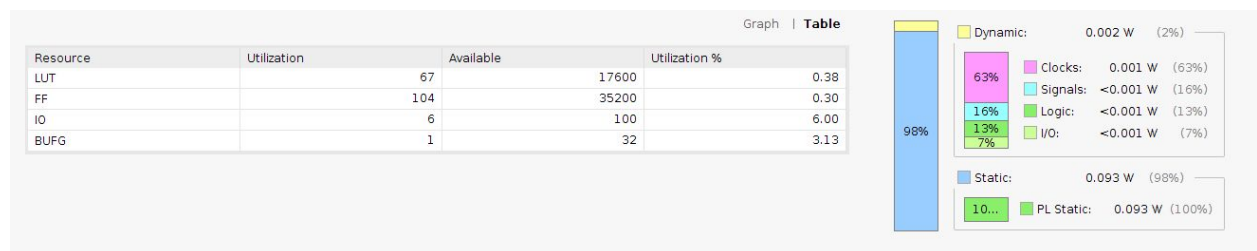
Elaborated Schematic



Synthesis Schematic



On-Chip Power and Utilization



Constraint File

```

1  ## This file is a general .xdc for the ZYBO Rev B board
2  ## To use it in a project:
3  ## - uncomment the lines corresponding to used pins
4  ## - rename the used signals according to the project
5
6
7  ##Clock signal
8  set_property -dict { PACKAGE_PIN L16  IOSTANDARD LVCMOS33 } [get_ports { clk }]; #IO_L11P_T1_SRCC_35 Sch=sysclk
9  create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports { clk }];
10
11  ##Buttons
12  set_property -dict { PACKAGE_PIN R18  IOSTANDARD LVCMOS33 } [get_ports { btn[0] }]; #IO_L20N_T3_34 Sch=BTN0
13  set_property -dict { PACKAGE_PIN P16  IOSTANDARD LVCMOS33 } [get_ports { btn[1] }]; #IO_L24N_T3_34 Sch=BTN1
14  #set_property -dict { PACKAGE_PIN V16  IOSTANDARD LVCMOS33 } [get_ports { btn[2] }]; #IO_L18P_T2_34 Sch=BTN2
15  #set_property -dict { PACKAGE_PIN Y16  IOSTANDARD LVCMOS33 } [get_ports { btn[3] }]; #IO_L7P_T1_34 Sch=BTN3
16
17
18  ##Pmod Header JB
19  set_property -dict { PACKAGE_PIN T20  IOSTANDARD LVCMOS33 } [get_ports { rts }]; #IO_L15P_T2_DQS_34 Sch=JB1_p
20  set_property -dict { PACKAGE_PIN U20  IOSTANDARD LVCMOS33 } [get_ports { rxd }]; #IO_L15N_T2_DQS_34 Sch=JB1_N
21  set_property -dict { PACKAGE_PIN V20  IOSTANDARD LVCMOS33 } [get_ports { txd }]; #IO_L16P_T2_34 Sch=JB2_P
22  set_property -dict { PACKAGE_PIN W20  IOSTANDARD LVCMOS33 } [get_ports { cts }]; #IO_L16N_T2_34 Sch=JB2_N
23

```

Discussion

The lab was very useful in understanding Finite State Machines. This system was useful for understanding how FSM's can be utilized and how they work. It also gave me an idea of how UART's transfer information bit by bit. It also helped understand how to use a PMOD and write a constraint file for it. I feel as I completely understand FSM and how they work. However, there is still some confusion with UART's. For example I created the extra credit echo program, but was unsure how it fit into the top level design. Also I am still uncomfortable with the debugging process. I had a bug with my top level implementation and it took me a very long time to figure out what was wrong.