ECSE 325

Lab 5 Report

Furkan Ercan, Arash Ardakani

Group 04

Alex Hale, Anna Bieber

260672475, 260678856

April 16th , 2018

Introduction

The goal of this lab is to learn how to use a combination of VHDL descriptions, Qsys components, and C logic to implement bridging between the Hard Processor System (HPS) and the FPGA of the DE1-SOC board. Functions will be implemented to map the inputs, switches and pushbuttons on the board, to the outputs, 7-segment displays and LEDs.

Design

After creating and initializing the project with the details for the HPS and memory on the DE1-SOC board, Parallel I/O components are made for each of the inputs and outputs. The bit widths are set individually according to the needs of each component: 8 bits for each 7-segment display, 1 bit for each switch, 1 bit for each pushbutton, and 1 bit for each LED. The declared components are connected to the clock and reset signals, and to each other, in the Qsys GUI, as seen in Figure 1. External connections are exported for the input and output components. VHDL design files are generated, ending the use of Qsys for the time being. Next, the generated TCL Script File is used to perform pin mappings for the memory connected to the HPS, and the provided QSF file is used to assign the remaining pin mappings.

The provided C code in *test_leds.c* is loaded to the board and executed to verify that the timer counts up on the 7-segment displays and the LEDs illuminate when their corresponding slider switches are activated, demonstrating that the component connections were made correctly.

Moving back into component creation, the provided *qsys_lab_custom_component.vhd* and *qsys_lab_function.vhd* files are loaded into the project. In *qsys_lab_function.vhd*, the skeleton is filled in to multiply two 16-bit numbers and output the result (Figure 2). In Qsys, a custom component type is created to make use of *qsys_lab_custom_component.vhd*, then a component of that type is added to the system and connected to the other components as required. VHDL design files are generated to reflect these changes.

Finally, the functionality of the newly-added multiplier is tested with a new C logic file, *test_multiplication.c* (Figure 3). The user first defines a 10-bit binary number using the slider switches, then saves that number by pressing one of the pushbuttons. The user then defines another 10-bit binary number using the slider switches. After another pushbutton press, the two numbers are multiplied, and the result is displayed in hexadecimal on the 7-segment displays. After another pushbutton press, the numbers and 7-segment displays are cleared, and another multiplication can be started.

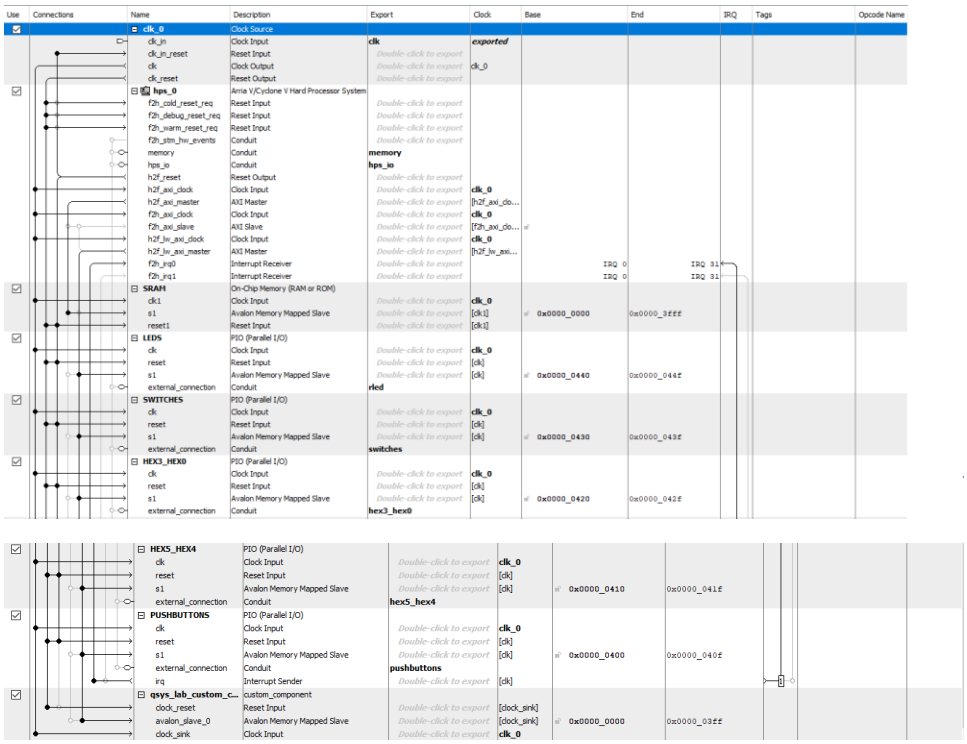

Figure 1 – Qsys GUI Connections

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3  USE ieee.numeric_std.all;
4
5  ENTITY qsys_lab_function IS
6  PORT ( clock, resetn : IN STD_LOGIC;
7     read, write, chipselect : IN STD_LOGIC;
8     address : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
9     in_data : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
10    out_data : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
11 );
12 END qsys_lab_function ;
13
14 ARCHITECTURE Behavior OF qsys_lab_function IS
15 SIGNAL  input1, input2: std_logic_vector(31 DOWNTO 0);
16 signal routput: unsigned(31 downto 0);
17
18 BEGIN
19    process(clock,resetn)
20    begin
21    if(resetn='0') then
22          input1<= (others=>'0');
23          input2<= (others=>'0');
24          routput<=(others=>'0');
25    elsif(rising_edge(clock)) then
26
27        if (write = '1') then
28
29          if (address(0) = '1') then
30             input1 <= in_data;
31          elsif(address(0)='0') then
32             input2<=in_data;
33       end if;
34
35       elsif(read='1') then
36          routput<=unsigned(input1(15 downto 0))*unsigned(input2(15 downto 0));
37
38       end if;
39
40    end if;
41    end process;
42    out_data<= std_logic_vector(routput);
43
44 END Behavior;
45
```

Figure 2 – VHDL Multiplication Function

```c
// test multiplication program
int main(void) {
    volatile int * led = (int *) 0xFF200040; // red LED address
    volatile int * hex3_hex0 = (int *) 0xFF200020; // 7-segment 0 to 3 address
    volatile int * hex5_hex4 = (int *) 0xFF200010; // 7-segment 4 to 5 address
    volatile int * switchptr = (int *) 0xFF200030; // SW slider switch address
    volatile int * pushbutton_ptr = (int *) 0xFF200040  // pushbutton address
    int switch_value;
    int state_machine_status = 0;

    int a = 0;
    int b = 0;

    while(1) {
        while(*(pushbutton_ptr) == 0) {
            // wait for pushbutton press, setting LEDs according to switches
            switch_value = *(switchptr);
            *(led) = switch_value;
        }

        if (state_machine_status == 0) {
            a = *(switchptr);   // read status of switches

            state_machine_status = 1;
        } else if (state_machine_status == 1) {
            b = *(switchptr);   // read status of switches
            int c = a * b;

            *(hex3_hex0) = pack_hex(hex_to_7segment((0x00F000 & c) / (pow(2.0, 12.0))), hex_to_7segment((0x000F00 & c) / (pow(2.0, 8.0))),
                hex_to_7segment((0x0000F0 & c) / (pow(2.0, 4.0))), hex_to_7segment(0x00000F & c));
            *(hex5_hex4) = pack_hex(0,0,hex_to_7segment((0xF00000 & c) / (pow(2.0, 20.0))), hex_to_7segment((0x0F0000 & c) / (pow(2.0, 16.0))));

            state_machine_status = 2;
        } else if (state_machine_status == 2) {
            a = 0;  // clear registers
            b = 0;

            // set 7-segment displays to 0
            *(hex3_hex0) = pack_hex(hex_to_7segment(0), hex_to_7segment(0), hex_to_7segment(0), hex_to_7segment(0));
            *(hex5_hex4) = pack_hex(0,0,hex_to_7segment(0), hex_to_7segment(0));
            state_machine_status = 0;
        }
    }
}
```

Figure 3 – Multiplication Test Code

Conclusion

This lab taught the use of a combination of VHDL descriptions, Qsys components, and C logic to implement bridging between the Hard Processor System (HPS) and the FPGA of the DE1-SOC board. Functions were implemented to map the inputs (switches and pushbuttons on the board), to the outputs (7-segment displays and LEDs).