

Short Problems

Problem 1 (2.5 points, estimated time 15 minutes)

Draw a block diagram corresponding to the given below VHDL code:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY swapmux IS
    PORT ( Data      : IN          STD_LOGIC_VECTOR(7 DOWNTO 0);
          Resetn, w  : IN          STD_LOGIC;
          Clock      : IN          STD_LOGIC;
          RinExt     : IN          STD_LOGIC_VECTOR(1 TO 3);
          BusWires   : BUFFER      STD_LOGIC_VECTOR(7 DOWNTO 0) );
END swapmux;

ARCHITECTURE Behavior OF swapmux IS

    COMPONENT regn IS
        GENERIC ( N : INTEGER := 8 );
        PORT ( R      : IN          STD_LOGIC_VECTOR(N-1 DOWNTO 0);
              Rin, Clock : IN      STD_LOGIC;
              Q      : OUT         STD_LOGIC_VECTOR(N-1 DOWNTO 0) );
    END COMPONENT regn;

    SIGNAL Rin, Q : STD_LOGIC_VECTOR(1 TO 3);
    SIGNAL S : STD_LOGIC_VECTOR(1 DOWNTO 0);
    SIGNAL R1, R2, R3 : STD_LOGIC_VECTOR(7 DOWNTO 0);

BEGIN

    Rin(1) <= RinExt(1) OR Q(3);
    Rin(2) <= RinExt(2) OR Q(2);
    Rin(3) <= RinExt(3) OR Q(1);

    reg1: regn PORT MAP ( BusWires, Rin(1), Clock, R1 );
    reg2: regn PORT MAP ( BusWires, Rin(2), Clock, R2 );
    reg3: regn PORT MAP ( BusWires, Rin(3), Clock, R3 );

    WITH Q SELECT
        S <= "00" WHEN "000",
            "10" WHEN "100",
            "01" WHEN "010",
            "11" WHEN OTHERS;

    WITH S SELECT
        BusWires <= Data WHEN "00",
            R1 WHEN "01",
            R2 WHEN "10",
            R3 WHEN OTHERS;

    PROCESS ( Resetn, Clock )
    BEGIN
        IF Resetn = '0' THEN
            Q <= (OTHERS => '0');
        ELSIF Clock'EVENT AND Clock = '1' THEN
```

```
Genbits: FOR i IN 3 DOWNT0 2 LOOP
    Q(i) <= Q(i-1);
END LOOP;
Q(1) <= w;
END IF ;
END PROCESS;
```

END Behavior;

Deliverable:

Hand-drawn block diagram.

Problem 2 (2.5 points, estimated time 15 minutes)

Using CAD tools available to you in the lab (including Constraint Editor) generate the UCF (User Constraint File) for the circuit described using VHDL code posted on the web at

http://ece.gmu.edu/courses/ECE449/exams_S04/Tu_section.htm

Assume that the circuit is to be implemented using the XSA-100 board, and the inputs to the circuit are provided using the following sources:

R[3:0] - inputs D3..D0 of the parallel port controlled using GXSPORT

L – the switch DIPSW1 of the XSA-100 board

w - pushbutton of the XSA-100 board

Clock – Master Clock on the board

An output from the circuit should appear in the form of a hexadecimal digit displayed using the seven segment display available on the board.

Deliverable:

Electronic version of the UCF file, to be stored in the subdirectory Problem2.

Problem 3 (2.5 points, estimated time 15 minutes)

Rewrite the following VHDL code using the **generate scheme for equations** (for-generate concurrent statement). The VHDL code is posted on the web at

http://ece.gmu.edu/courses/ECE449/exams_S04/Tu_section.htm

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity PROBLEM_3 is
    port(
        DATA_IN : in STD_LOGIC_VECTOR(15 downto 0);
        DATA_OUT : out STD_LOGIC
    );
end PROBLEM_3;

architecture PROBLEM_3_DATAFLOW of PROBLEM_3 is
    signal L3_0:std_logic;
    signal L3_1:std_logic;
    signal L3_2:std_logic;
    signal L3_3:std_logic;
    signal L3_4:std_logic;
    signal L3_5:std_logic;
    signal L3_6:std_logic;
    signal L3_7:std_logic;
    signal L2_0:std_logic;
    signal L2_1:std_logic;
    signal L2_2:std_logic;
    signal L2_3:std_logic;
    signal L1_0:std_logic;
    signal L1_1:std_logic;
    signal L0_0:std_logic;

begin

    L3_0<=DATA_IN(0) xor DATA_IN(1);
    L3_1<=DATA_IN(2) xor DATA_IN(3);
    L3_2<=DATA_IN(4) xor DATA_IN(5);
    L3_3<=DATA_IN(6) xor DATA_IN(7);
    L3_4<=DATA_IN(8) xor DATA_IN(9);
    L3_5<=DATA_IN(10) xor DATA_IN(11);
    L3_6<=DATA_IN(12) xor DATA_IN(13);
    L3_7<=DATA_IN(14) xor DATA_IN(15);

    L2_0<=L3_0 xor L3_1;
    L2_1<=L3_2 xor L3_3;
    L2_2<=L3_4 xor L3_5;
    L2_3<=L3_6 xor L3_7;

    L1_0<=L2_0 xor L2_1;
    L1_1<=L2_2 xor L2_3;

    L0_0<=L1_0 xor L1_1;

    DATA_OUT<=L0_0;

end PROBLEM_3_DATAFLOW;

```

Deliverable:

Electronic version of the VHDL file, to be stored in the subdirectory Problem 3.

Problem 4 (2.5 points, estimated time 15 minutes)

Draw the state diagram corresponding to the following VHDL code:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity FSM_MIDTERM is
    port (
        CLOCK: in STD_LOGIC;
        INPUT: in STD_LOGIC;
        RESET: in STD_LOGIC;
        OUTPUT: out STD_LOGIC_VECTOR (2 downto 0));
end;

architecture FSM_MIDTERM_OPERATION of FSM_MIDTERM is

    type Sreg0_type is (S1, S2, S3, S4);
    signal Sreg0: Sreg0_type;

    attribute syn_state_machine: boolean;
    attribute syn_state_machine of Sreg0: signal is true;

    attribute syn_preserve: boolean;
    attribute syn_preserve of FSM_MIDTERM_OPERATION: architecture is false;

begin

    Sreg0_machine: process (CLOCK, reset)
    begin
        if RESET='1' then
            Sreg0 <= S1;
            OUTPUT <= "000";
        elsif CLOCK'event and CLOCK = '1' then
            case Sreg0 is
                when S1 =>
                    OUTPUT <= "000";
                    if INPUT='1' then
                        Sreg0 <= S2;
                    elsif INPUT='0' then
                        Sreg0 <= S1;
                    end if;
                when S2 =>
                    OUTPUT <= "010";
                    if INPUT='1' then
                        Sreg0 <= S4;
                    elsif INPUT='0' then
                        Sreg0 <= S3;
                    end if;
                when S3 =>
                    OUTPUT <= "100";
                    if INPUT='0' then
                        Sreg0 <= S1;
                    elsif INPUT='1' then
                        Sreg0 <= S4;
                    end if;
            end case;
        end if;
    end process;
end;
```

```
        when S4 =>
            OUTPUT <= "101";
            if INPUT='1' then
                Sreg0 <= S4;
            elsif INPUT='0' then
                Sreg0 <= S3;
            end if;
        when others =>
            null;
    end case;
end if;
end process;

end FSM_MIDTERM_OPERATION;
```

Deliverable:

Hand-drawn block diagram.

Hands-on Design Problem (25 points, estimated time 1 hour)

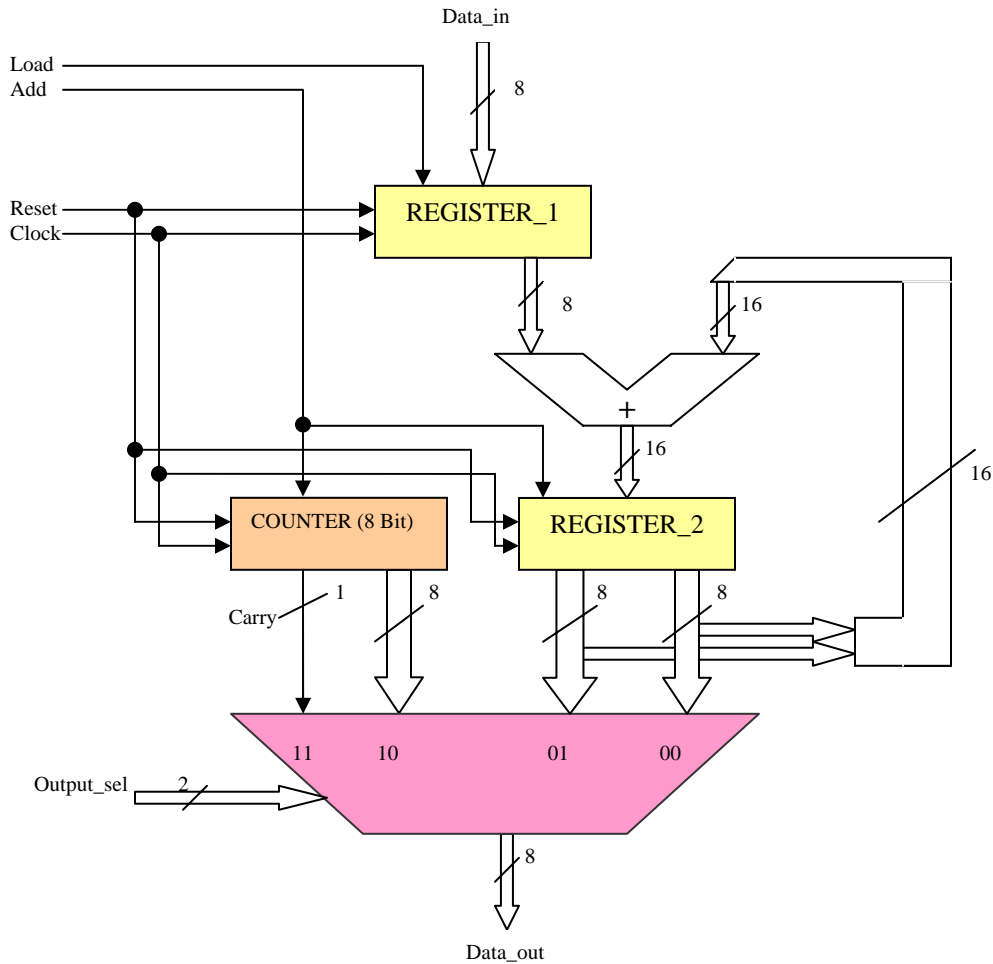
Circuit description

Design and write an RTL VHDL code for an 8-bit adder accumulator that is capable of calculating the sum

$$S = D_1 + D_2 + \dots + D_{N-1} + D_N$$

for $N \leq 256$.

The block diagram of the circuit is shown in the figure below, and its inputs/outputs are specified in the table below.



8-bit data words D_i are fed to the circuit using the data input *Data_in*. A 16-bit sum is accumulated in the register *REGISTER_2*. *COUNTER* contains the number of data words added up to the given point in time.

<i>Signal</i>	<i>Width (Bits)</i>	<i>Mode</i>	<i>Function</i>
<i>Data_in</i>	8	IN	Data Input
<i>Load</i>	1	IN	When HIGH, REGISTER_1 is loaded with Data_in on the rising edge of Clock
<i>Clock</i>	1	IN	Master Clock
<i>Reset</i>	1	IN	When HIGH, REGISTERS and the COUNTER is reset to a known state (all '0's)
<i>Add</i>	1	IN	When HIGH, REGISTER_2 is loaded with output of the ADDER (shown with the '+' sign) on the rising edge of Clock
<i>Output_sel</i>	2	IN	Selects among 4 inputs to the multiplexer
<i>Data_out</i>	8	OUT	Data Output

Design Requirements

The combinational portion of the circuit should be described using the dataflow VHDL code, and the sequential portion of the circuit should be described using the synthesizable behavioral code. Your code should infer a circuit that requires a minimum amount of FPGA resources. The target clock frequency should be 50 MHz.

Tasks

Perform the following tasks:

1. Write and debug a VHDL code of your main circuit.
2. Write a testbench verifying the operation of your main circuit.
3. Perform functional simulation of your circuit and use it to debug your VHDL code.
4. Synthesize your main circuit using Synplify Pro in the GUI mode. Save the RTL Netlist diagram.
5. Implement your main circuit using Xilinx ISE.
6. Perform post-synthesis and timing simulations of your circuit using Active-HDL. Check if post-synthesis simulation matches functional simulation. Based on the circuit block diagram and timing simulation, determine the most critical path in your circuit and its length.

Deliverables (electronic files to be written to the zip disk)

1. VHDL code of your main circuit fulfilling the requirements specified in the *Design Requirements* section above - 10 pts
2. VHDL code of your testbench – 5 points
3. RTL Netlist of your main circuit – 2 points
4. Simulation waveforms from the post-synthesis simulation proving the correct operation of your circuit – 2 points
5. Simulation waveforms from the timing simulation demonstrating the delay of its critical path – 2 points
6. FPGA resource utilization – 2 points
7. Minimum clock period of your circuit – 2 points.