Majority Circuit

Signature and Grading Sheet

Group #:_____ Name(s):_____ .

Signature

Section 4.5: _____.

Grading

- Section 4.1: Truth table and logic expressions (20 points):_____.
  Attach a page.
- Section 4.2: VHDL code (25 points):_____.
  Attach code printout (with proper header and comment)
- Section 4.3 Simulation (25 points):_____.
  Attach simulation timing diagram printout.
- Section 4.4 (10 points):_____.
  Attach one-page report printout with # of LEs circled.
- Section 4.5 (20 points):_____.

Total points: _____.

Experiment
Majority Circuit

1 Purpose

To design a combinational circuit with basic logical operators

2 Reading

- Chapter 3 of *Embedded SoPC Design with Nios II Processor and VHDL Examples.*

3 Project specification

We want to design a circuit that counts 4 votes and displays the results. Only VHDL logical operators (i.e., **and**, **or**, **not**, and **xor**) can be used in VHDL code and no process is allowed.

- input:
  - `v`: 4-bit inputs representing 4 votes, with 1 for yes and 0 for no..
- output
  - `fail`: 1-bit output. It is asserted when the motion fails (i.e., less than two 1's).
  - `tie`: 1-bit output. It is asserted when the vote is a tie (i.e., two 1's and two 0's).
  - `pass`: 1-bit output. It is asserted when there is a majority (i.e., three or four 1's).

4 Design Procedures

4.1 Truth table

(a) Derive the truth table for this circuit.

(b) Derive the logic expressions in SOP (sum-of-product) format.

4.2 VHDL design

The entity declaration of this design is shown below:

```
entity maj is
    port(
         v: in std_logic_vector(3 downto 0);
         fail, tie, pass: out std_logic
    );
END maj;
```

Derive the architecture body with only VHDL logical operators to design circuit. You should follow the guidelines when writing VHDL code:

- Include a file header as in the example VHDL file.
- Add proper spaces and blank lines to enhance readability
- Add proper comments in your VHDL code.
- Use meaningful names for signals.
- Use the exact entity declaration shown above.

A sample VHDL file is attached in Appendix.

4.3 Simulation

Compile the design and perform simulation to verify its operation. The input and output signals should be clearly arranged and represented in proper format so that the simulation result can be easily understood. All 16 input combinations should be included. A good timing diagram is shown in Appendix.

4.4 Circuit size

(a) Look at the compiling report and determine the number of LEs used in circuit.

4.5 Implementation and testing

(a) Perform pin assignment as follows:
- `v`: 4 slide switches
- `fail, tie, pass`: 3 LEDs

(b) Re-synthesize the circuit again.

(c) Check in the pin assignment in fitter report.

(d) Download the configuration file to DE1 board.

(e) Use the switches and LEDs to verify the operation of physical circuit.

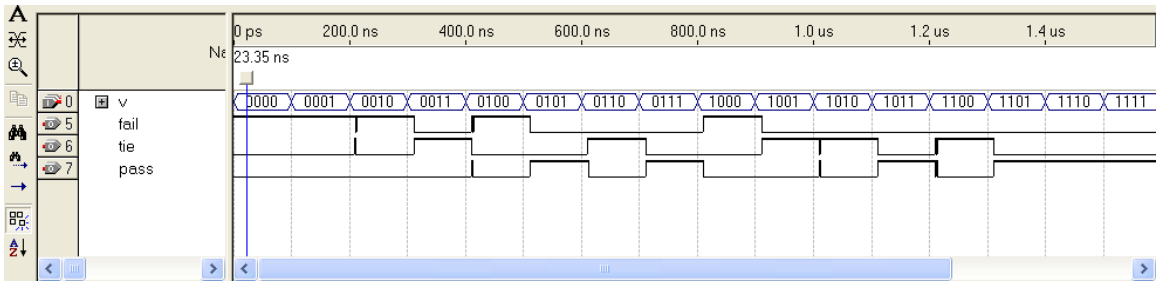
(f) Demonstrate the circuit to instructor and get signature.

5 Appendix

5.1 Verification and Simulation

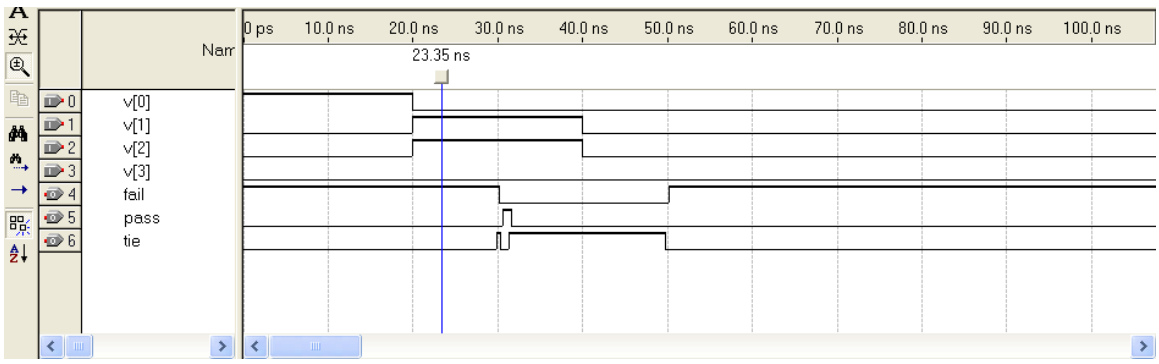
Both of your schematic design and VHDL description need to be verified by simulation. In Quartus waveform editor, you can group signal as a "bus" and show them in different format (binary, octal, hex or decimal). You should choose proper format to make the timing diagram clear and comprehensible.

Example of a good and a poor timing diagram are shown bellow. The circuit simulated is identical to the one of this project. Following is a good timing diagram:



Note that we can easily identify the error (the "0101" pattern).

A poor timing diagram is shown below. The signal format is difficult to comprehend, the time scale is small and the number of test vectors is not sufficient.



5.2 A sample documented VHDL file

```
--************************************************************
--
--    Author: p chu
--
--    File: sum3.vhd
--    Design units:
--          ENTITY sum3
--          ARCHITECTURE prim_op_arch
--    Purpose: count # of 1s from inputs
--          Inputs:  a,b,c
--          Outputs: s: 2-bit sum
--
--    Library/Package:
--          ieee.std_logic_1164: to use std_logic
--
--    Software/Version:
--          Simulated by: Altera Quartus v6.0
--          Synthesized by: Altera Quartus v6.0
--
--    Revision History
--          Version 1.0:
--          Date: 9/1/2006
--          Comments: Original
--
--************************************************************

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY sum3 IS
      PORT(
            a,b,c: IN STD_LOGIC;
            s: OUT STD_LOGIC_VECTOR(1 DOWNTO 0));
END sum3;

-- architecture using primitive logic operators
ARCHITECTURE prim_op_arch OF sum3 IS
      SIGNAL tmp_sig: STD_LOGIC;
BEGIN
      -- the boolean equation is essentially for a adder
      s(1) <= (a AND b) OR (a AND c) OR (b AND c) ;
      -- tmp is just used for demo purpose
      tmp_sig <= a XOR b;
      s(0) <= tmp_sig XOR c;
END prim_op_arch;
```