Name Key

EE3610 Practice Final Exam.

Part 1. Short Answer.

1. (3 pts) With the IEEE std_logic_1164 library, what will the state of a std_logic signal be if it is simultaneously driven with '0' and 'Z'?
   '0'

2. (3 pts) Describe how VHDL resolves signal conflicts like those in Question 1.
   The resolution function takes a vector of signals and returns the resolved signal value. When a signal is driven from multiple sources, VHDL calls the resolution function with a vector formed from the various sources

3. (4 pts) Represent 9,375 using 32-bit IEEE floating point (in hexadecimal or binary).
   $9375 = 10010010011111 = 1.001001001111 \times 2^{13}$ $13+127 = 140 = 10001100$

   0 1000 1100 0010 0100 1111 0000 0000 00 = 46127C00

4. (4 pts) What number is represented by C0D00000 in 32-bit IEEE floating point?
   $11000000\underbrace{1}_{129}1010\ldots$ $-1.101 \times 2^{129-127} = -1.101 \times 2^2 = -110.1$
   -6.5

5. (2 pts) (true/(false)) The 32-bit IEEE floating point representation of infinity is $FFFFFFFF_{16}$. 7FFFFFFF

6. (4 pts) Name two techniques that can be used to test sequential logic circuits
   (a) Distinguishing Sequences
   (b) Scan Path Testing

7. (2 pts) The code segment below is most likely to imply (dedicated/distributed) memory (circle one).
   ```
   if clk'event and clk='1' then
       data_out <= rom_array(address);
   end if;
   ```

8. (2 pts) ((true)/false) The original motivation for boundary scan testing was to test for circuit board defects (including soldering defects).

9. (3 pts) What is the acronym by which the IEEE/ANSI standard interface for boundary scan testing is commonly known?
   JTAG.

10. (2 pts) (true/(false)) One of the main advantages of a 1-hot state assignment is that the number of flip-flops is minimized.

11. (6 pts) The VHDL code below purports to implement the 4-bit linear feedback shift register shown in Figure 10-27 of the text. Identify 2 errors.

```
architecture behavioral of LFSR is
    type linear_feedback_shift_register is array (4 downto 1) of bit;
    signal Q: linear_feedback_shift_register;
    signal D1, Y: bit;
begin

    process(clk,rst)
    begin
        if rst='1' then
            Q <= "0000";
        elsif clk'event and clk='1' then
            for i in Q'left(1) to Q'right(1) loop
                if i = Q'left(1) then
                    Q(i) <= D1;
                else
                    Q(i) <= Q(i-1);
                end if;
            end loop;
        end if;
    end process;

    process (Q)
    begin
        if Q(1)='0' and Q(2)='0' and Q(3)='0' then
            Y <= '1';
        end if;
        D1 <= Q(1) xor Q(4) xor Y;
    end process;

end behavioral;
```

1 to 4

(Q, Y) or Make Y a variable

Creates a latch

else Y <= '0';

will be wrong Y unless

Extra Credit (3 pts): There is a third error in the code above that is particularly subtle. Find it.

Part 2. Problems.

12. In this problem, you are to design a VHDL behavioral model for an ALU that takes two 32-bit inputs A and B, and generates a result, Y, according the opcode and the function table below. (You may ignore carry for all operations, including addition and subtraction.) The entity part is provided on the top of the next page. (a) (8 pts) Write the architecture part. Be sure to add comments.

| opcode | Y |
|---|---|
| 000 | A |
| 001 | B |
| 010 | A+B |
| 011 | A – B |
| 100 | A and B |
| 101 | A or B |
| 110 | A rotated left by (the least significant 5 bits of) B |
| 110 | A rotated right by (the least significant 5 bits of) B |

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity alu32 is
    port ( opcode: in std_logic_vector(2 downto 0);      -- opcode as shown in table
           A, B: in std_logic_vector(31 downto 0);       -- 32-bit inputs
           Y: out: std_logic_vector(31 downto 0));       -- 32-bit output
end alu32;
```

```
architecture behavioral of alu32 is
    signal S, T, U: unsigned (31 downto 0);
    signal n: integer range 0 to 31
begin
    S <= unsigned(A); T <= unsigned(B); -- unsigned for convenience
    n <= To_integer(T(4 downto 0)); -- rotate amount - integer
    U <= S when opcode = "000" else          -- Y = A
         T when opcode = "001" else          -- Y = B
         S+T when opcode = "010" else        -- Y = A+B
         S-T when opcode = "011" else        -- Y = A-B
         S and T when opcode = "100" else    -- Y = A and B
         S or T when opcode = "101" else     -- Y = A or B
         S(31-n downto 0) & S(31 downto 32-n) when opcode = "110" else
         S(n-1 downto 0) & S(31 downto n); -- Y = A ror B

    Y <= std_logic_vector(U)
```

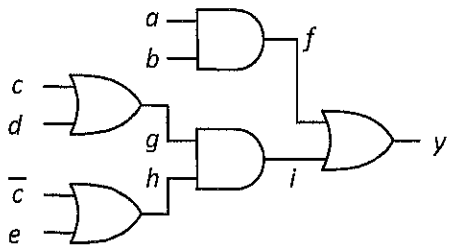
(b) (3 pts) What is the risk of synthesizing an ALU using a behavioral model like this?

We are forced to accept the synthesizer's choice of adder or subtractor

(c) (3 pts) How could you mitigate the risk that you described in part (b)?

Add structural models to compute A+B and A-B

13. Consider the circuit below:




(a) (12 pts) Find a set of test vectors that will test every stuck-at fault. Note: treat $\bar{c}$ as its own variable, but one that is constrained (in the test vectors) to be the complement of *c*. (e.g. use $\bar{c}0$ to denote $\bar{c}$ stuck-at-0.)

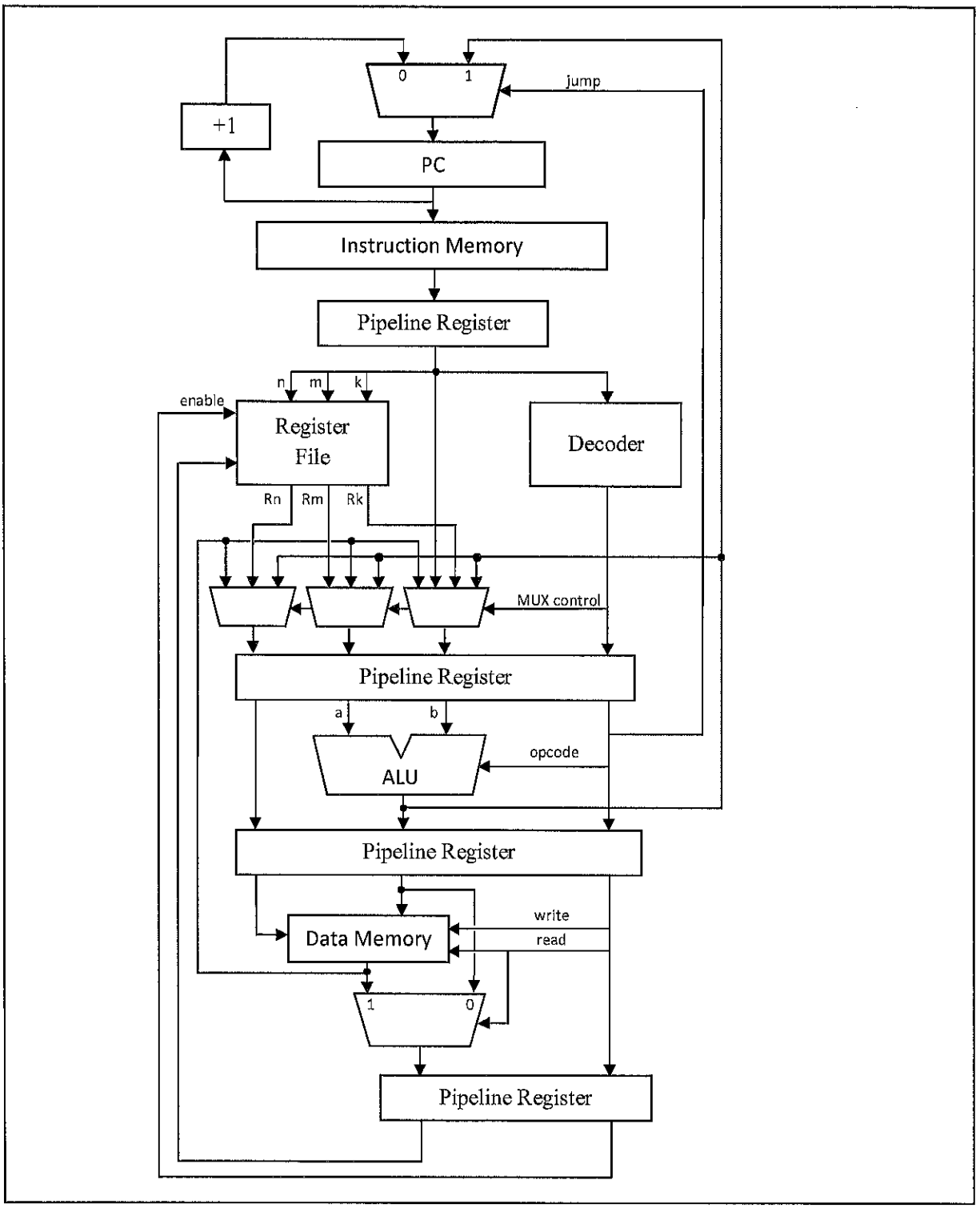
| *a* | *b* | *c* | *d* | *e* | *y* | Fault(s) |
|---|---|---|---|---|---|---|
| 1 | 1 | X | 0 | 0 | 1 | a0, b0, f0 |
| 0 | X | 1 | 0 | 1 | 1 | c0, e0, g0, h0, i0 |
| 0 | X | 0 | 1 | 0 | 1 | d0, $\bar{c}$0, g0, h0, i0 |
| 0 | 1 | 0 | 0 | 1 | 0 | a1, c1, d1, g1, i1 |
| 1 | 0 | 1 | 1 | 0 | 0 | b1, $\bar{c}$1, e1, h1, i1 |
| | | | | | | |

(b) (8 pts) Write a test bench to test every vector you found in (a). TO GET FULL CREDIT YOU MUST (i) use constant arrays for your input and output vectors, and (ii) check the output of the circuit and generate an error message if it is incorrect.

```
entity testbench is
end testbench;

architecture behavioral of testbench is
    component foo is
        port (a, b, c, d, e: in bit; y: out bit);
    end foo;
    signal a, b, c, d, e, f, y: bit;
    type vt is array (1 to 5) of bitvector (1 to 6);
    constant v: vt := ("110001", "001011", "000101", "010010", "101100");
begin
    uut: foo(a,b,c,d,e,y);
    process
    begin
        for i in v'range loop
            a <= v(i)(1);
            b <= v(i)(2);
            c <= v(i)(3);
            d <= v(i)(4);
            e <= v(i)(5);
            wait for 10 ns;
            assert y = v(i)(6) report "wrong-o, bucko";
        end loop;
    end process;
end behavioral;
```

Use the 5-stage processor architecture depicted in the figure below for Problem 14.



14. Consider the program below:

```
add r1,r1,r3          ; r1 ← r1 + r3
load r2,4(r1)         ; load r2 with the contents of data memory address [r1+4]
and r2,r2,#255        ; r2 ← r2 and 011111111₂
store r2,8(r1)        ; store r2 at address [r1+8] in data memory
```

(a) (3 pts) Assume a "write-through" register file and the forwarding paths shown in the diagram. Which of these instructions (if any) will stall the pipeline?

and r2,r2,#255

(b) (6 pts) Assume all hazards are detected in the decode stage (i.e. an instruction remains in the decode stage until it is safe for it to proceed.) If the first instruction (add r1,r1,r3) is in the fetch stage in cycle 1, complete the table below to show the pipeline stage for each instruction on a given cycle. You may abbreviate a stage with its first letter. (e.g. Write F instead of Fetch).

| Instruction | Cycle 1 | Cycle 2 | Cycle 3 | Cycle 4 | Cycle 5 | Cycle 6 | Cycle 7 | Cycle 8 | Cycle 9 |
|---|---|---|---|---|---|---|---|---|---|
| add r1,r1,r3 | F | D | E | M | W | | | | |
| load r2,4(r1) | - | F | D | E | M | W | | | |
| and r2,r2,#255 | - | | F | D | D | E | M | W | |
| store r2,8(r1) | - | | | F | F | D | E | M | W |

(c) (4 pts) The inputs of the ALU in the diagram are labeled a and b. Assume that initially R1 = $1234_{16}$, R3 = $2220_{16}$ and the data memory at address $3458_{16}$ is $89AB_{16}$. State which instruction is in the *execute* stage and what the value of a and b are during each clock cycle given below. (Note: if a pipeline bubble is in the execution stage, write X in each column.)

| Cycle | Instruction | a | b |
|---|---|---|---|
| 3 | add r1,r1,r3 | 1234 | 2220 |
| 4 | load r2,4(R1) | 3454 | 0004 |
| 5 | X | X | X |
| 6 | and r2,r2,#255 | 89AB | 00FF |
| 7 | store r2,8(r1) | 3454 | 0008 |

1234
2220
3454

(d) (4 pts) What value is stored to memory by the last instruction in the program (store r2,8(r1)), and at what address is it stored?

Value: 00AB

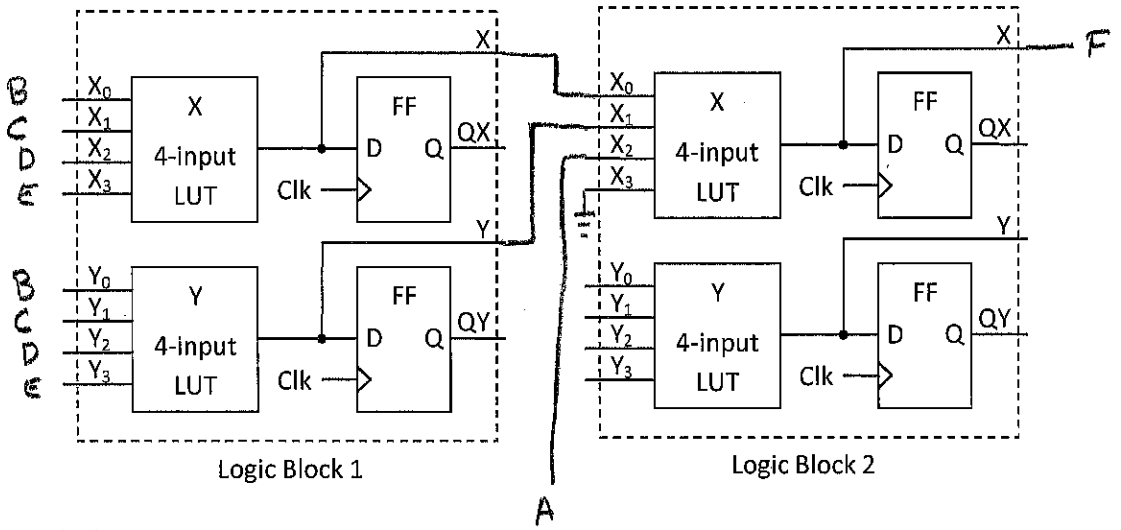
Address: 345C

15. (a) (4 pts) Use Shannon's Decomposition to expand the equation below about the variable *A*.

$$F = A\bar{B}C\bar{E} + \bar{A}CD\bar{E} + AB\bar{C}\bar{D}E + \bar{B}\bar{C}DE$$

F = $A(\bar{B}C\bar{E} + B\bar{C}\bar{D}E + \bar{B}\bar{C}DE) + \bar{A}(CD\bar{E} + \bar{B}\bar{C}DE)$

(b) (10 pts) Use your results from (a) to implement this equation with the logic blocks below. Label inputs A-E, label output F, and draw in the necessary interconnects. Then give the logic function for each block. (Hint. You may not need all the LUTs.)




Logic Block 1:

X = $\bar{X}_0 X_1 \bar{X}_3 + X_0 \bar{X}_1 \bar{X}_2 X_3 + \bar{X}_0 \bar{X}_1 X_2 X_3$

Y = $Y_1 Y_2 \bar{Y}_3 + \bar{Y}_0 \bar{Y}_1 Y_2 Y_3$

Logic Block 2:

X = $X_0 X_2 + X_1 \bar{X}_2$

Y = Not Used