

Sir Ming Leo
 07/24/2000

+12 1. (12 points) (3 points) ⁺³ Write an entity for a 4-bit comparator which has 2 4-bit inputs and outputs a '1' if all of the bits match and a '0' otherwise. (9 points) ⁺⁹ Use concurrent signal assignments to model this comparator.

```
entity compare is
    port (a, b: bit_vector(3 downto 0); check: out bit);
end compare;
```

```
Architecture concurrent of compare is
    signal temp0, temp1, temp2, temp3: bit;
begin
```

```
check <= not (temp0 or temp1 or temp2 or temp3);
```

```
temp0 <= a(0) xor b(0);
temp1 <= a(1) xor b(1);
temp2 <= a(2) xor b(2);
temp3 <= a(3) xor b(3);
```

```
end concurrent;
```

sb
 Architecture concurrent compare is
 begin
 check <= '1' when A=B
 else '0';
 end concurrent;

x1 2. (1 point) A package is a collection of commonly used data types and subprograms used in a design.

x1 3. (1 point) A variable assignment is identified by the symbol :=.

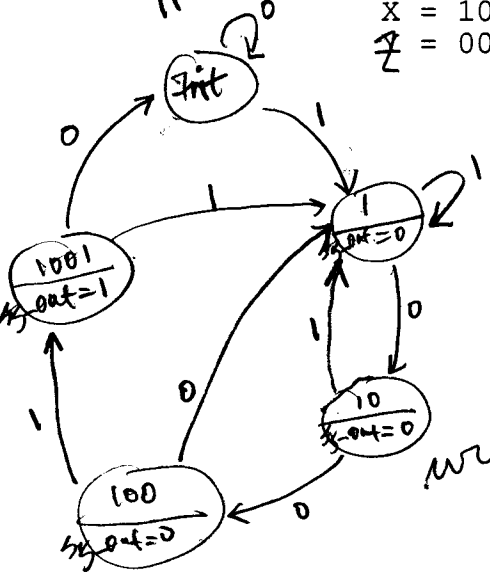
+84. (8 points) For the following declarations, are the given assignments valid?

```
SIGNAL count : BIT_VECTOR (1 TO 9);
ALIAS sign : BIT IS count (1);
ALIAS msd : BIT_VECTOR (1 TO 4) IS count (2 TO 5);
ALIAS lsd : BIT_VECTOR (1 TO 4) IS count (6 TO 9);
```

- (a) ~~no~~ sign <= "10";
- (b) ~~yes~~ msd <= "1001";
- (c) ~~yes~~ count <= "110010000";
- (d) ~~no~~ lsd <= sign;

5. (15 points) Design a resetting Moore machine that detects the number 9₁₀ encoded in binary, 1001, and excess-3 1100. Sample input (X) and output (Z) sequences are given below.

X = 100110001100...
Z = 000100000001...



this recognizes overlapping sequences

wrong for 1100

entity detect9 is
port (sig-in; clk, reset; in bit,
sig-out; out bit);
end detect9;

architecture state-machine of detect9 is
type state-type is (Init, one, onezero,
onezerozero, onezerozeroone);
signal state : state-type;

```
begin
process (clk)
begin
if (clk='1' and clk'event) then
if (reset='1') then
state <= Init;
end if;
end if;
```

what about the output?

```
if (state = Init) then
if (sig-in = '1') then
state <= one;
else
state <= Init;
end if;
elsif (state = one) then
if (sig-in = '0') then
state <= onezero;
else
state <= one;
end if;
elsif (state = onezero) then
if (sig-in = '0') then
state <= onezerozero;
else
state <= one;
end if;
elsif (state = onezerozero) then
if (sig-in = '1') then
state <= onezerozeroone;
else
state <= one;
end if;
elsif (state = onezerozeroone) then
if (sig-in = '0') then
state <= Init;
else
state <= one;
end if;
end if;
```

end if;
end process

6. (1 point) An entity X, when used in another entity Y, becomes a component for the entity Y.

7. (2 points) For the following function call, which function will be called?

VARIABLE a, b : INTEGER;
b := decrement (a);

- (a) FUNCTION decrement (x : INTEGER) RETURN INTEGER;
- (b) FUNCTION decrement (x : REAL) RETURN REAL;

continued

with state select
sig-out <= '0' when Init;
'0' when one;
'0' when onezero;
'0' when onezerozero;
'1' when onezerozeroone;

OK

end state-machine;

+10

8. (10 points) Draw the state diagram for the following state machine.

Moore state machine
well, I am not sure!

```

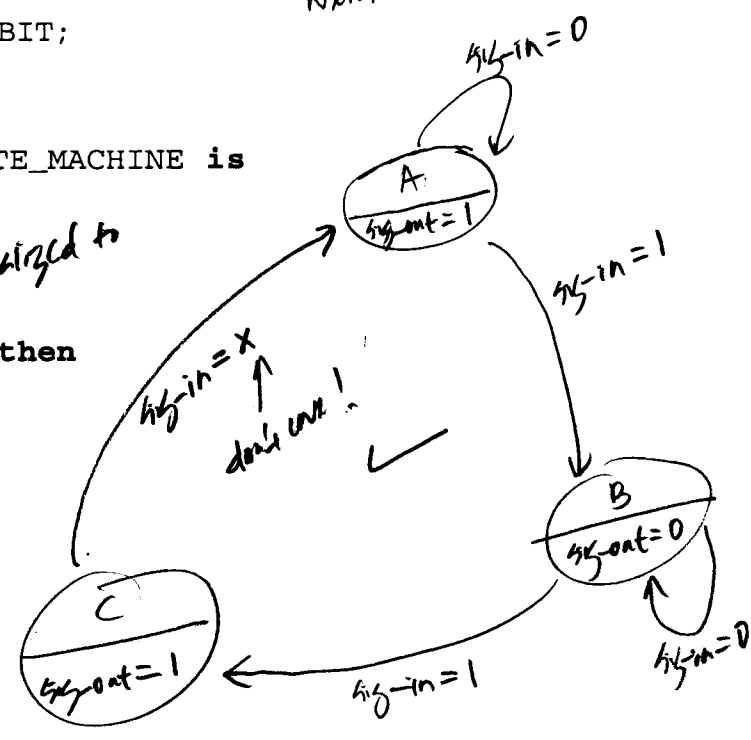
entity STATE_MACHINE is
  port (SIG_IN : in BIT; CLK : in BIT;
        SIG_OUT : out BIT);
end STATE_MACHINE;

architecture STATE_MACHINE of STATE_MACHINE is
  type STATE_TYPE is (A, B, C);
  signal STATE : STATE_TYPE := A;
begin
  process (CLK)
  begin
    if (CLK = '1' and CLK'EVENT) then
      if (STATE = A) then
        if (SIG_IN = '1') then
          STATE <= B;
        end if;
      elsif (STATE = B) then
        if (SIG_IN = '1') then
          STATE <= C;
        end if;
      elsif (STATE = C) then
        STATE <= A;
        end if;
      end if;
    end process

    with STATE select
      SIG_OUT <= '1' when A,
                '0' when B,
                '1' when C;
  end STATE_MACHINE;
  
```

that's why it is moore

state machine initialized to state A!



+5

9. (5 points) Write a process statement that will have the same effect as the following concurrent signal assignment statement.

```

with Z select
  A <= transport X and Y when 0, X or Y when 1, not X when others;
  
```

```

process (Z, X, Y)
begin
  if (Z = '0') then
    A <= transport (X and Y);
  elsif (Z = '1') then
    A <= transport (X or Y);
  else
    A <= transport (not(X));
  end if;
end process;
  
```

+15 (3)

15 10. (15 points) (a) (5 points) Create entity declarations and architecture body pairs for:

- a. and2
- c. not

15 (b) (5 points) Create a package with component declarations for the entities defined previously.
 15 (c) (5 points) Implement the equation $f = ab'$ using the components in the package created.

```

1a) entity and2 is
  port (a, b : in bit; c : out bit);
end and2;

architecture behave of and2 is
begin
  c <= a and b;
end behave;
  
```

```

entity inv is
  port (a : in bit; c : out bit);
end inv;

architecture behave of inv is
begin
  c <= not a;
end behave;
  
```

```

1b) package mypack is
  component and2
    port (a, b : in bit; c : out bit);
  end component;
  component inv
    port (a : in bit; c : out bit);
  end component;
end mypack;
  
```

```

c) use work.mypack.all;
library ieee;
use ieee.std_logic_1164.all;

entity function is
  port (a, b : in bit; f : out bit);
end function;

Architecture conc of function is
  signal temp0 : bit;
begin
  u1 : inv port map (b, temp0);
  u2 : and2 port map (temp0, a, f);
end conc;
  
```

} not needed for bit

11. (10 points) Specify type declarations for the following data types.

3 a. (3 points) A MONTH_OF_YEAR enumeration data type.

Type MONTH_OF_YEAR is (Jan, Feb, Mar, Apr, May, June, Jul, Aug, Sept, Oct, Nov, Dec);

3 b. (2 points) A data type DAY_OF_YEAR that can have integer values in the range from 1 to 366.

3 Type DAY_OF_YEAR is integer range 1 to 366;

3 c. (2 points) An ascending range data type INC_8 with integer values from 0 to 7.

Type INC_8 is integer range 0 to 7;

3 d. (3 points) A 16-bit descending-index register composite data type, REGISTER_8_BIT_ASCENDING, with index values from the type INC_8 declared above, and component values of type BIT.

Type REGISTER_8_BIT_ASCENDING is array (INC_8) of bit;

Subtype

+4 12. (4 points) Write declarations for the following constants, variables, or signals. Refer to the data types defined in Problem 11.

+1 a. (1 point) A constant MID_YEAR that has value JUNE of type MONTH_OF_YEAR.

constant MID_YEAR : ~~MONTH_OF_YEAR~~ := JUNE;

+1 b. (1 point) A signal HOLIDAY of type DAY_OF_YEAR.

signal HOLIDAY : ~~DAY_OF_YEAR~~;

+2 c. (2 points) A variable NUMBER which is initialized to the 8-bit representation for (-1) in twos complement notation of type REGISTER_8_BIT_ASCENDING.

variable NUMBER : ~~REGISTER_8_BIT_ASCENDING~~ := "11111111";

+13 13. (15 points) Write a VHDL procedure that counts the number of 1's, 0's and Z's in a std_logic_vector. The procedure must be general, i.e., it must accept an input parameter of arbitrary length.

procedure one-zero-z (a: in std_logic_vector; one, zero, z: out integer) is

variable temp-one, temp-zero, temp-z: integer;

begin

temp-one := 0;

temp-zero := 0;

temp-z := 0;

for i in a'range loop

if (a(i) = '0') then

temp-zero := temp-zero + 1;

elsif (a(i) = '1') then

temp-one := temp-one + 1;

elsif (a(i) = 'Z') then

temp-z := temp-z + 1;

end if;

end loop;

one := temp-one;

zero := temp-zero;

z := temp-z;

end one-zero-z;

:= not <=

+1 14. (1 point) All statements inside of a subprogram are sequential.

+1 15. (1 point) A process is executed whenever signals in the sensitivity list
being stimulated (N signal waits on. arrived!)
has an event

in things
in function
and procedure
are default to
variable.