

The University of Alabama in Huntsville
ECE Department
CPE 526 01
Midterm Exam Solution
Spring 2017

1. (18 points) (12 points) Write a VHDL function that has inputs **vect** of type **std_logic_vector**, **begin** of type natural and **end** of type natural. You may assume that **vect** has the form **std_logic_vector(vect'length-1 downto 0)**. The function extracts **sub** of the form **std_logic_vector(begin downto end)** from **vect**. Output an error if **begin** is less than **end**. Also output an error if **begin - end** is greater than **vect'length - 1**. (b)(6 points) Show an architecture that includes three calls to the function with the following properties. 1 - returns a value, 2 - triggers first error message 3 - triggers second error message.

```
library ieee;
use ieee.std_logic_1164.all;

package mine is
    function extract (vect : std_logic_vector; beginning : natural;
                     ending : natural) return std_logic_vector;
end mine;

package body mine is
    function extract (vect : std_logic_vector; beginning : natural;
                     ending : natural) return std_logic_vector is
        variable temp : std_logic_vector(beginning downto ending);
        variable copy : std_logic_vector(vect'length - 1 downto 0);
    begin
        copy := vect;
        assert (beginning >= ending)
            report "beginning is less than ending"
            severity note;
        assert (beginning - ending <= vect'length - 1)
            report "subvector is too large"
            severity note;
        temp := copy(beginning downto ending);
        return (temp);
    end extract;
end mine;

use work.mine.all;
library ieee;
use ieee.std_logic_1164.all;

entity try is
end try;

architecture try of try is
    signal this : std_logic_vector(0 to 5);
    signal that : std_logic_vector (18 downto 0);
begin
    process
    begin
        process
        begin
            this <= extract("00000111000000", 7, 2) after 10 ns;
            this <= transport extract("00000111000000", 2, 7) after 20 ns;
            that <= extract("00000111000000", 20, 2) after 30 ns;
            wait;
        end process;
    end process;
end try;
```

2.(7 points) Modify the following VHDL model by adding a parameter that sets the number of flip-flops in the counter. Also, add an input which is loaded with an asynchronous load input signal which is active low.

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity UPCOUNT is
    generic (N : integer);
    port ( CLOCK, RESETN, E LOADN: in std_logic;
          IN_VAL : in std_logic_vector(N-1 downto 0);
          Q      : out std_logic_vector (N-1 downto 0));
end UPCOUNT;

architecture BEHAVIOR of UPCOUNT is
    signal COUNT : std_logic_vector (N-1 downto 0);
begin
    process (CLOCK, RESETN, LOADN)
    begin
        if RESETN = '0' then
            COUNT <= (OTHERS => '0');
        elsif LOADN = '0' then
            COUNT <= IN_VAL;
        elsif (CLOCK'event and CLOCK = '1') then
            if E = '1' then
                COUNT <= COUNT + 1;
            else
                COUNT <= COUNT;
            end if;
        end if;
    end process;
    Q <= COUNT;
end BEHAVIOR;

```

3. (1 point) 'event is an example of a VHDL attribute.
4. (1 point) b Multiple Choice: Which of the following cannot occur outside a process?
(a) Signal Assignment (b) Variable Declaration (c) Signal Declaration
5. (1 point) A procedure is used when you have multiple return values.
6. (1 point) A(n) event occurs when a signal changes value.
7. (1 point). Bit is an example of a built-in enumerated type.

8. (14 points) Design a circuit using VHDL that resolves priority among eight active-low inputs $i(0) - i(7)$ where $i(0)$ has the highest priority. The circuit must produce active-high address outputs $a(2) - a(0)$ to indicate the number of the highest-priority asserted input. If at least one input is asserted, then an active-high **avalid** output should be asserted. If multiple outputs are asserted, an active-high output **mul** should be asserted.

Use the following entity:

```
entity priority is
  port (i : in std_logic_vector (0 to 7);
        avalid, mul : out std_logic;
        a : out std_logic_vector (2 downto 0));
end entity priority;
```

```
architecture behave of priority is
```

```
begin
  process (i)
    variable count : integer;
  begin
    count := 0;
    for j in i'range loop
      if (i(j) = '0') then
        count := count + 1;
      end if;
    end loop;
    if count > 1 then
      mul <= '1'; avalid <= '1';
    elsif count = 1 then
      avalid <= '1'; mul <= '0';
    else
      mul <= '0'; avalid <= '0';
    end if;
    if (i(0) = '0') then a <= "000";
    elsif (i(1) = '0') then a <= "001";
    elsif (i(2) = '0') then a <= "010";
    elsif (i(3) = '0') then a <= "011";
    elsif (i(4) = '0') then a <= "100";
    elsif (i(5) = '0') then a <= "101";
    elsif (i(6) = '0') then a <= "110";
    elsif (i(7) = '0') then a <= "111";
    else a <= "XXX";
    end if;
  end process;
end behave;
```

9. (13 points) Construct a 5 to 32 decoder with 3 to 8 decoders with enable. If necessary, configure a 3 to 8 decoder to represent any additional logic needed. The entities for the 3 to 8 and 5 to 32 decoders are as follows.

```

entity dec3to8 is
  port (x : in std_logic_vector (2 downto 0);
        en : std_logic;
        y : out std_logic_vector (7 downto 0));
end entity dec3to8;

architecture behave of dec3to8 is
  ...
end behave;

entity dec5to32 is
  port (x : in std_logic_vector (4 downto 0);
        en : std_logic;
        y : out std_logic_vector (31 downto 0));
end entity dec5to32;

architecture structure of dec5to32 is
  signal temp : std_logic_vector (7 downto 0);
begin
  U0 : entity dec3to8
    port map (x(2) => '0', x(1) => x(4), x(0) => x(3), y => temp, en => en);
  U1 : entity dec3to8(behave)
    port map (x => x(2 downto 0), en => temp(3), y => y(31 downto 24));
  U2 : entity dec3to8(behave)
    port map (x => x(2 downto 0), en => temp(2), y => y(23 downto 16));
  U3 : entity dec3to8(behave)
    port map (x => x(2 downto 0), en => temp(1), y => y(15 downto 8));
  U4 : entity dec3to8(behave)
    port map (x => x(2 downto 0), en => temp(0), y => y(7 downto 0));
end structure;

```

10. (18 points) Given the following VHDL, indicate all transactions and events. Give the values of A, B, C, D, E, and F each time a change occurs. Carry this out until no further change occurs.

```

entity prob is
  port (D : out bit);
end prob;

architecture PROB of PROB is
  signal A, B, C, E, F : bit;
begin
  process
    A <= '1' after 5 ns;
    wait;
  end process;
  P1: process (D, C)
  begin
    B <= D after 2 ns;
    E <= C after 7 ns;
  end process P1;
  C1: C <= transport A or E
    after 6 ns;
  P2: process (C, E)
  begin
    F <= (C and E) after 4 ns;
  end process P2;
  C2: D <= A xor B xor C after 1 ns;
end PROB;

```

Time	A	B	C	D	E	F
0 ns	0	0	0	0	0	0
5 ns	1	0	0	0	0	0
6 ns	1	0	0	1	0	0
8 ns	1	1	0	1	0	0
9 ns	1	1	0	0	0	0
11 ns	1	0	1	0	0	0
18 ns	1	0	1	0	1	0
22 ns	1	0	1	0	1	1

Time	Event	Processes Triggered	Scheduled Transactions	Event?
5 ns	A → 1	C1	C (1, 11 ns)	Yes
		C2	D (1, 6 ns)	Yes
6 ns	D → 1	P1	B (1, 8 ns)	Yes
		P1	E (0, 13 ns)	No
8 ns	B → 1	C2	D (0, 9 ns)	Yes
9 ns	D → 0	P1	B (0, 11 ns)	Yes
		P1	E (0, 16 ns)	No
11 ns	C → 1, B → 0	P1	B (0, 13 ns)	No
		P1	E (1, 18 ns)	Yes
		P2	F (0, 15 ns)	No
		C2	D (0, 12 ns)	No
18 ns	E → 1	P2	F (1, 22 ns)	Yes
		C1	C (1, 24 ns)	No
22 ns	F → 1	None		

Scheduling Rules	Transport	Inertial
New before existing	Overwrite existing	Overwrite existing
New after existing	Append new	If $v_{\text{new}} = v_{\text{existing}}$, append new Elsif $t_{\text{new}} - t_{\text{existing}} >$ reject append new Else overwrite existing

11. (15 points) Draw the state diagram for the following state machine. You may omit reset from all the arcs.

```

entity STATE_MACHINE is
    port (CLK, RESET, START, FULL, EMPTY, ZERO: in std_logic;
          HOT, COLD, DRAIN, TURN : out std_logic);
end STATE_MACHINE;

architecture SYNTH of STATE_MACHINE is
    type STATE_TYPE is (IDLE, FILL_1, WASH, DRAIN_1, SPIN_1,
                        FILL_2, RINSE, DRAIN_2, SPIN_2);
    signal STATE : STATE_TYPE;
begin
    process(CLK, RESET, START, FULL, EMPTY)
    begin
        if (RESET = '1') then
            STATE <= IDLE;
        elsif (CLK'event and CLK = '1') then
            HOT <= '0'; COLD <= '0'; DRAIN <= '0'; TURN <= '0';
            case STATE is
                when IDLE => if (START = '1') then
                    STATE <= FILL_1;
                else
                    STATE <= IDLE;
                end if;
                when FILL_1 => if (FULL = '1') then
                    STATE <= WASH; TURN <= '1';
                else
                    STATE <= FILL_1; HOT <= '1';
                end if;
                when WASH => if (ZERO = '1') then
                    STATE <= DRAIN_1;
                else
                    TURN <= '1'; STATE <= WASH;
                end if;
                when DRAIN_1 => if (EMPTY = '1') then
                    TURN <= '1'; STATE <= SPIN_1;
                else
                    DRAIN <= '1'; STATE <= DRAIN_1;
                end if;
                when SPIN_1 => if (ZERO = '1') then
                    STATE <= FILL_2;
                else
                    STATE <= SPIN_1; DRAIN <= '1'; TURN <= '1';
                end if;
            end case;
        end if;
    end process;
end architecture SYNTH;

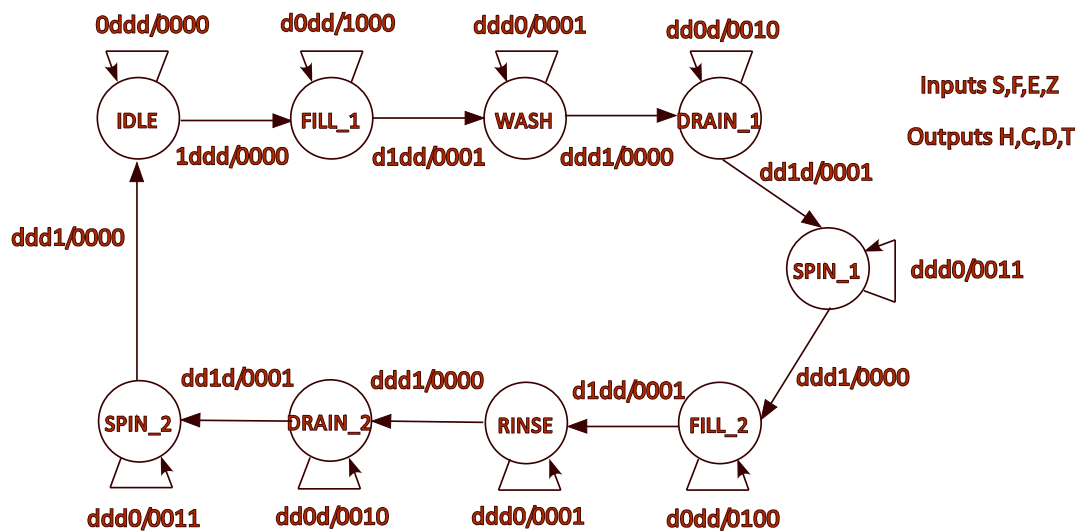
```

```

when FILL_2 => if (FULL = '1') then
    TURN <= '1'; STATE <= RINSE;
else
    STATE <= FILL_2; COLD <= '1';
end if;
when RINSE => if (ZERO = '1') then
    STATE <= DRAIN_2;
else
    TURN <= '1'; STATE <= RINSE;
end if;
when DRAIN_2 => if (EMPTY = '1') then
    TURN <= '1'; STATE <= SPIN_2;
else
    STATE <= DRAIN_2; DRAIN <= '1';
end if;
when SPIN_2 => if (ZERO = '1') then
    STATE <= IDLE;
else
    STATE <= SPIN_2; TURN <= '1'; DRAIN <= '1';
end if;

end case;
end if;
end process;
end SYNTH;

```



12. (10 points) Write a VHDL entity (3 points) and architecture (7 points) of a synthesizable T flip-flop with synchronous reset that operates on the falling edge of the clock.

```
entity t_ff is
  port (t, clk, r : in std_logic;
        q, qb : out std_logic);
end entity;

architecture synth of t_ff is
begin
  process (clk)
  begin
    if (clk'event and clk = '0') then
      if r = '1' then
        q <= '0';
        qb <= '1';
      elsif (t = '1') then
        q <= not q;
        qb <= not qb;
      end if;
    end if;
  end process;
end synth;
```