

ECE 368, Fall 2005, Instructor: Prof. Shantanu Dutt

Midterm Exam: Mon, Nov. 14, Time: 2-3.15 pm

Exam Format: Open Book (only text book and one sheet of notes), Total Points: 100

Solutions

1. Regular Circuit Description using Generate Statements

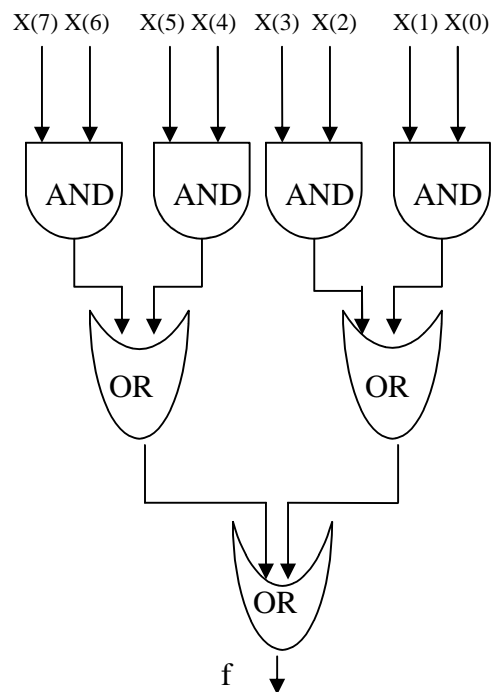
(a) You need to **structurally** describe in VHDL using the **generate** statements a $(n = 2^k)$ -input tree based AND-OR circuit for implementing a given SOP expression f with the following characteristics:

The SOP expression has 2^{k-1} product terms, and each product term has exactly 2 literals; n is the total number of literals. The literals are given as inputs to the circuit as a `std_logic` vector X with n elements. E.g. for $n = 4$, $f = ab + a'c$, and $X(0) = a$, $X(1) = b$, $X(2) = a'$, $X(3) = c$; for $n = 8$, $f = ac + a'b + bc + b'c'$ and $X(0) = a$, $X(1) = c$, $X(2) = a'$, $X(3) = b$, $X(4) = b$, $X(5) = c$, $X(6) = b'$, $X(7) = c'$. Note that, as shown in the figure below, the AND-OR tree should have 2-input AND gates only in the first level and 2-input OR gates in the rest of the levels. Also as shown in the figure, a gate at level $j > 1$ has inputs from the outputs of two gates at level $j - 1$.

You need to give both the entity and architecture descriptions of the circuit. Assume that you have the following entities for 2-input AND and OR gates that you can use for instantiating the required AND and OR gates in your code.

```
entity and_2 is
    generic (td: time := 2 ns); -- propagation delay of gate
    port (a, b : in std_logic; c: out std_logic);
end entity and_2;
entity or_2 is
    generic (td: time := 2 ns); -- propagation delay of gate
    port (a, b : in std_logic; c: out std_logic);
end entity or_2;
```

Note that in the entity description of the required circuit, k needs to be a generic parameter and either k and/or some function(s) of k need to be used in the architecture description. Also, use another generic parameter `gate_delay` of type `time` that you need to use as the delay of the AND/OR gate when instantiating them in your architecture. (40)



(b) Clearly derive the expression for the delay of the n -input AND-OR circuit above as a function of n and t_d (delay of a 2-input AND-OR gate). (10)

Solution:

a) The structural description in VHDL is given below:

```
entity tree_ckt is
    generic (k : natural, gate_delay : time := 2 ns);
    port (x : in std_logic_vector ( 2**k-1 downto 0);
          f : out std_logic);
end entity tree_ckt;

architecture struct of tree_ckt is
    -- component definitions of and_2 and or_2 entities
    type matrix is array (k-1 downto 0, 2**k-1 downto 0) of std_logic;
    signal temp : matrix;
begin
    outer_loop: for j in k-1 downto 0 generate
        inner_loop: for i in 0 to 2**j-1 generate
            and_gen: if j = k-1 then generate
                and_gate: entity work.and_2(behav)
                    generic map (gate_delay);
                    port map (X(2*i), X(2*i + 1), temp(j,i));
            end generate;
            or_gen: if j < k-1 then generate
                or_gate: entity work.or_2(behav)
                    generic map (gate_delay);
                    port map (temp(j+1, 2*i), temp(j+1, 2*i + 1), temp(j,i));
            end generate;
        end generate;
    end generate;
    f <= temp(0,0);
end architecture struct;
```

b) Delay = (# of levels in AND-OR tree) * td
= $\log_2(n) * td$

2. VHDL Process Design

Consider the following architecture-process description:

```
entity wierd is
    port (clk, a, b: input std_logic; c, d: output std_logic);
end entity wierd;
architecture wierder of wierd is
    signal w, x, y, z: std_logic;
begin
```

```
    x <= a xor b after 7 ns;
    y <= not(clk) and b after 8 ns;
```

```
    foo: process (sensitivity list) is
        signal u, v: std _logic;
        -- various variable descriptions
    begin
```

The only statemets involving signals in this process are:

- wait until clk='1' and clk'event and clk'last_value='0';
- v <= u xor w after 7 ns;
- z <= v nand x after 4 ns;
- c <= not(z) after 2 ns;
- Jumbo := y nor b; -- jumbo is a variable

```
end process foo;
```

```
    boo: process (sensitivity list) is
        -- various variable descriptions
    begin
```

The only statemets involving signals in this process are:

- wait until clk='0' and clk'event and clk'last_value='1';
- d <= a xor z;
- timbuktoo := x and a after 4 ns; --timbuktoo is a variable
- w <= x xnor timbuktoo after 7 ns;

```
end process boo;
```

```
end architecture wierder;
```

- (a) Will process `foo` execute correctly if it does not have a sensitivity list or the corresponding `wait on` statement at the end of the process? Why or why not? (5)
- (b) What are the drawbacks of not having a sensitivity list in a process or the corresponding `wait on` statement at the end of the process? (5)
- (c) Determine the sensitivity list of processes `foo` and `boo` so that they execute correctly but also as efficiently as possible (i.e., as little time taken for simulating it as possible). The key to determining the sensitivity list of a given process is to allow all necessary iterations of the process but avoid unnecessary ones. (15)

Solution:

- a) `foo` will execute correctly if it does not have a sensitivity list or the corresponding “wait on” statement. The reason is that without the above sensitivity list/wait statement, `foo` will iterate infinitely (or as long as the simulation is allowed to proceed). Thus any new signal value that is supposed to appear in the iterations after the current one will appear and the signal/variable assignment statements with these new values on the R.H.S (inputs) will be executed so that new values are computed for the output signals/variables of these statements and so forth. Thus, starting with initial input signals and internal variables values, all signal/variable assignment values will propagate through all signal/variable assignment statements until the final correct outputs of the process are produced.
- b) Drawbacks of not having a sensitivity list or corresponding “wait on” statement is that the process will iterate as many times as possible (within the given simulation time window) irrespective of whether any signal or variable value at the input of its assignment statements has changed. If there were no changes to any of these signal/variable values, the assignment statements need not be simulated. But without the sensitivity list or “wait on” statements all the statements will be executed leading to needless computations and loading of the CPU.
- c) Sensitivity list of any process should only have those external signals that are inputs (R.H.S) of its assignment or wait statements and those internal signals that appear as inputs as well as outputs of at least one assignment or wait statements. Thus Sensitivity list of process `foo` should be (`clk`, `x`, `y`, `u`, `w`, `v`, `z`, `b`) and sensitivity list of process `boo` should be (`clk`, `a`, `z`, `x`).

3. VHDL Signals and Variables

Consider the following architecture description:

```
architecture XYZ of ABC is
signal a, b, c, d: integer := 1;
signal clk: bit := '0';
begin
clock: process is
begin
    clk = '0','1' after 7 ns;
    wait for 14 ns ;
end process clock;
foo: process is
variable x, y, z: integer := 2;
begin
    -- point 1
    a = 2 * b + 3 * x after 1 ns;
    x := 2 * y + b ** 2;
    b = x ** 2 + y after 5 ns;
    --point 2
    wait for 3 ns;
    --point 3
    c = a + b;
    y := 2 * a + c;
    --point 4
    wait until clk'event and clk='1';
    d = a + b;
    y := 4 * y + d;
    --point 5
    wait on clk;
end process foo;
end architecture XYZ;
```

Determine the simulation time, and values of the signals and variables at the following points and iterations given in the table below. Assume that simulation begins at time 0 and that the first value of the clock is '0' (and thus that the first transition on the clock will be a positive edge and will occur at time 7 ns---half the clock period).

(25)

Solution:

Iteration-1, point 1	Time	a	b	c	d	x	y	z
	0 ns	1	1	1	1	2	2	2
Iteration-1, point 2	Time	a	b	x				
	0 ns	1	1	5				
Iteration-1, point 3	Time	a	b	x				
	3 ns	8	1	5				
Iteration-1, point 4	time	b	c	y				
	3 ns	1	1	17				
Iteration-1, point 5	time	b	c	d				
	7 ns	27	9	1				
Iteration-2, point 1	time	a	b	c	d	x	y	z
	14 ns	8	27	9	35	5	17	2
Iteration-2, point 2	time	a	b	x				
	14 ns	8	27	763				