

ECE 368, Spring 2006, Instructor: Prof. Shantanu Dutt

Solutions to Midterm Exam

1. Regular Circuit Description using Generate Statements

(a) You need to **structurally** describe in VHDL using `generate` statements a tree-based circuit for realizing an SOP expressions with $(m = 2^k)$ product terms and n literals (complemented or uncomplemented forms of inout variables, i.e., x or x'). Each product term can have between 1 and 5 literals. Thus product terms for the SOP expression can range from x' (no AND gate needed here) to xy to $uvw'y'z$, where u, v, w, x, y, z are logic (not VHDL) variables. The n literals are available as the input array $X(n-1), \dots, X(0)$. The information about each of the m product terms is available as any array $prod(m-1), \dots, prods(0)$, where each $prod(i)$ is a record structure of type `prod_struct` declared as:

```
type prod_struct is record
lit_num: integer;
lit_list: array (0 to 4) of integer;
end record;
```

The meanings of the two fields in the record structure are as follows:

1. `lit_num` is the number of literals in the product term represented by this record structure.
2. `lit_list(0)` to `lit_list(lit_num-1)` are the integer indices of the literals in array X that are in this product term. If `lit_num = 3` and `lit_list(0)` to `lit_list(2) = 1 3 9`, then it means that the literals in the product term represented by this record structure are $X(1)$, $X(3)$ and $X(9)$. In other words, if $i < lit_num$, then $X(lit_list(i))$ is a literal in the product term represented by this record structure.

There are 4 types of AND gates available with 2 to 5 inputs called `and2`, `and3`, `and4`, and `and5`. The entity port description of an r -input AND gate has the first r signals as inputs and the last signal as an output. There is also a generic delay parameter.

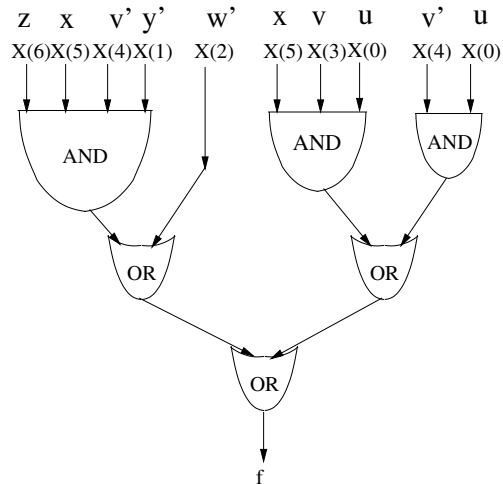
Also, 2-input OR gates `or2` are available. A sample entity description of a 3-input AND gates and the 2-input OR gates are given below.

```
entity and3 is
generic (td: time := 6 ns); -- propagation delay of gate
```

```
port (a, b, c : in std_logic; d: out std_logic);
end entity and3;
```

```
entity or2 is
generic (td: time := 4 ns); -- propagation delay of gate
port (a, b : in std_logic; c: out std_logic);
end entity or2;
```

The figure below shows the circuit structure that should be generated by your architecture code.



SOP tree for $n=7$ literals and $m=4$ product terms

The prod array for the input of this problem is:

prod(0) = {lit_num = 2, lit_list = 0 4 . . .}

prod(1) = {lit_num = 3, lit_list = 0 3 5 . .}

prod(3) = {lit_num = 1, lit_list = 2 . . .}

prod(4) = {lit_num = 4, lit_list = 1 4 5 6 .}

The tree implements the SOP expression

$uv' + uvx + w' + y'v'xz$, where

$X(0)=u, X(1)=v', X(2)=w', X(3)=v,$

$X(4)=y', X(5)=x, X(6)=z$

Give the structural VHDL code of the architecture for entity `SOP_tree_circuit` using nested `generate` statements to generate an SOP tree of the form shown in the above figure. **50**

Solution:

```
entity SOP_tree_circuit is
generic (n, k: natural); -- there are n literals and m = 2^k product terms
port (X: in std_logic_vector(n-1 downto 0);
prod: in array (2**k-1 downto 0) of prod_struct;
f: out std_logic -- circuit output);
end entity SOP_tree_circuit;

architecture struct of SOP_tree is
type matrix is array (k downto 0, 2**k-1 downto 0) of std_logic;
signal wire : matrix;
begin
outer_loop: for j in k downto 0 generate
inner_loop: for i in 0 to 2**j-1 generate
and_gen: if j = k then generate

no_and_gen: if prod(i).lit_num = 1 then generate
```

```

wire(k, i) <= X(prod(i).lit_list(0) );
end generate; and2_gen: if prod(i).lit_num = 2 then generate

and_gate2: entity work.and2(behav)
generic map (4 ns);
port map (X(prod(i).lit_list(0)), X(prod(i).lit_list(1)), wire(k,i));
end generate;
and3_gen: if prod(i).lit_num = 3 then generate

and_gate3: entity work.and3(behav)
generic map (6 ns);
port map (X(prod(i).lit_list(0)), X(prod(i).lit_list(1)), X(prod(i).lit_list(2)), wire(k,i));
end generate;
and4_gen: if prod(i).lit_num = 4 then generate

and_gate4: entity work.and4(behav)
generic map (8 ns);
port map (X(prod(i).lit_list(0)), X(prod(i).lit_list(1)), X(prod(i).lit_list(2)), X(prod(i).lit_list(3)), wire(k,i));
end generate;
and5_gen: if prod(i).lit_num = 5 then generate

and_gate5: entity work.and5(behav)
generic map (10 ns);
port map (X(prod(i).lit_list(0)), X(prod(i).lit_list(1)), X(prod(i).lit_list(2)), X(prod(i).lit_list(3)), X(prod(i).lit_list(4)),
wire(k,i));
end generate;
end generate;

or_gen: if j < k-1 then generate

or_gate: entity work.or2(behav)
generic map (4 ns);
port map (wire(j+1, 2*i), wire(j+1, 2*i + 1), wire(j,i));
end generate;
end generate;

end generate;
f <= wire(0,0);

end architecture struct;

```

2. Behavioral VHDL

Give the entity and behavioral architecture description of an n -to- 2^n decoder with an enable signal en . If $en = 1$, the decoder functions normally, and if $en = 0$, all decoder outputs are 0. Besides en , the decoder has an n -bit `std_logic_vector` input and has a (2^n) -bit `std_logic_vector` output. Specify a delay of $n \times t_d$ ns for each output of the decoder irrespective of whether the output is set to 0 or to 1; n, t_d should be generic parameters. If you need to you can use functions `conv_to_std_logic_vector(integer_inp, k)` which converts integer value “integer_inp” to a k -bit `std_logic_vector` and `conv_to_integer(std_logic_vector_inp, k)` which converts a k -bit `std_logic_vector` “std_logic_vector_inp” to an unsigned integer value. 25

Solutions:

```
entity decoder is
generic(n: natural; td: time := 2 ns);
port(X: in std_logic_vector(n-1 downto 0); en: in std_logic; f: out std_logic_vector(2**n-1 downto 0));
end entity decoder;
```

```
architecture behav of decoder is
begin
foo: process is

variable inp : integer;
begin
inp := conv_to_integer(X, n);
for i in 0 to 2**n -1 loop
if i=inp then f(i) <= '1' and en after n*td;
else f(i) <= '0' after n*td;
endif;
end loop;
end process foo;
end architecture behav;
```

3. VHDL: Signals and Variables

Consider the following architecture description.

```
architecture XYZ of ABC is
signal a, b, c, d: integer := 0;
signal clk: bit := '0';
begin

clock: process is
begin
clk <= '0','1' after 5 ns;
wait for 10 ns ;
end process clock;

foo: process is
variable x, y, z: integer := 1;
begin
-- point 1
a <= b**2 + x;
x := 3*y + b**2;
b <= x**2 + y after 4 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;
--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 5 ns—half the clock period).

25

iter 1, point 1	time	a	b	c	d	x	y	z
iter 1, point 2	time	a	b	x				
iter 1, point 3	time	a	b	x				
iter 1, point 4	time	b	c	y				
iter 1, point 5	time	b	c	d				
iter 2, point 1	time	a	b	c	d	x	y	z
iter 2, point 2	time	a	b	x				