CPE462 - VHDL: Simulation and Synthesis - Fall'11

Prof. Nuno Alves (nalves@wne.edu), College of Engineering

Homework Assignment #7

Due Date: Monday, October 10th 2011

1.a)

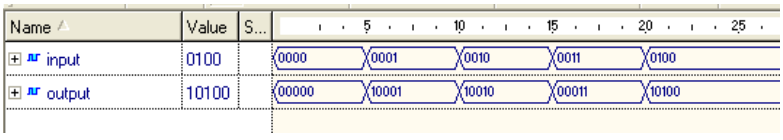
```
entity parity_gen is
         -- this entity will be generic with n=7
        generic (n: integer := 7);
        -- the input signal is defined at compilation
        -- where n is an integer defined before
        port (input : in bit_vector (n-1 downto 0);
        -- output signal is also dynamic
        output : out bit_vector(n downto 0));
end     entity;

-- entering the architecture part of the code
architecture myarch of parity_gen is
-- there are no internal signals defined here
begin
        -- entering process whose instructions are executed sequentially
        -- inside it. this process will be entered whenever input
        -- changes
        process(input)
        -- defining a bit variable
        variable temp1 : bit;
        -- defining a bit_vector variable whose size is the same as
        -- the the range of the output
        variable temp2 : bit_vector (output'range);
        begin
                -- initializing the temp variable to zero
                temp1:='0';
                -- loop that starts at i, and ends up at n-1
                for i in input'range loop
                        -- XOR each individual element with the content
                        -- of the temp variable, and store its result in
                        -- there
                        temp1 := temp1 XOR input(i);
                        -- copy each input element to a temporary variable
                        temp2(i) := input(i);
                end loop;
                temp2(output'high):=temp1; -- sets the parity as LSB bit
                output<=temp2; -- assigns the temporary value to the output
        end process;
end architecture;
```

1.b) Main program:

```
entity parity_gen is
        generic (n: integer := 4);
        port (input : in bit_vector (n-1 downto 0);
        output : out bit_vector(n downto 0));
end     entity;

architecture myarch of parity_gen is
begin
        process(input)
        variable temp1 : bit; variable temp2 : bit_vector (output'range);
        begin
                temp1:='0';
                for i in input'range loop
                        temp1 := temp1 XOR input(i); temp2(i) := input(i);
                end loop;
                temp2(output'high):=temp1;
                output<=temp2;
        end process;
end architecture;
```

Test-bench:

```
entity testparity is
end;

architecture bench of testparity is
  constant n : positive := 4;
  component parity_gen
        generic (n: integer := n);
        port (input : in bit_vector (n-1 downto 0);
        output : out bit_vector(n downto 0));
  end component;

  signal input  : bit_vector (n-1 downto 0);
  signal output : bit_vector(n downto 0);

begin

 input   <= "0000"
 , "0001" after 5 ns
 , "0010" after 10 ns
 , "0011" after 15 ns
 , "0100" after 20 ns ;


 m: parity_gen  generic map(n) port map (input,output);

end bench;
```



2.

```
SIGNAL a : BIT := '1';
SIGNAL b : BIT_VECTOR (3 DOWNTO 0) := "1100";
SIGNAL c : BIT_VECTOR (3 DOWNTO 0) := "0010";
SIGNAL d : BIT_VECTOR (7 DOWNTO 0);
SIGNAL e : INTEGER RANGE 0 TO 255;
SIGNAL f : INTEGER RANGE -128 TO 127;
```

a)

```
x1 <= a & c;          ->    x1 <= ________
x2 <= c & b;          ->    x2 <= ________
x3 <= b XOR c;        ->    x3 <= ________
x4 <= a NOR b(3);     ->    x4 <= ________
x5 <= b sll 2;        ->    x5 <= ________
x6 <= b sla 2;        ->    x6 <= ________
x7 <= b rol 2;        ->    x7 <= ________
x8 <= a AND NOT b(0) AND NOT c(1);     ->    x8 <= ________
d <= (5=>'0', OTHERS=>'1');            ->    d <= ________
```

x1 <= "11100";

x2 <= "00101100";

x3 <= "1110";

x4 <= '0';

x5 <= "0000";

x6 <="0000"; * sla means shift left arithmetic (fill right vacated bits with rightmost bit)

x7 <= "0011"; * rol mean rotate left (circular)

x8 <= '0';

d<="11011111";

b)

```
c'LOW              ->    0
d'HIGH             ->    7
c'LEFT             ->    3
d'RIGHT            ->    0
c'RANGE            ->    (3 downto 0)
d'LENGTH           ->    8
c'REVERSE_RANGE    ->    (0 to 3)
```

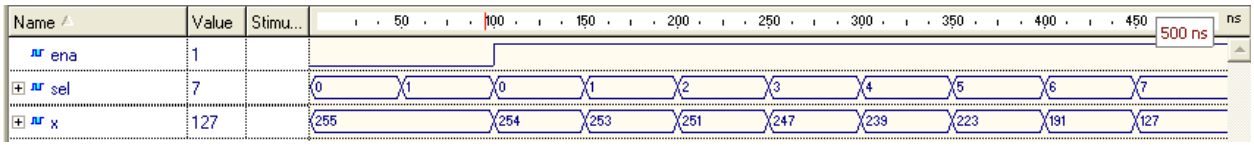
c)

`b(0) AND a`	• legal since both signals have the same data-type and dimensions
`a + d(7)`	• illegal arithmetic with bit datatype
`NOT b XNOR c`	• legal, logic operators amongst same sized bit_vectors
`c + d`	• illegal, arithmetic with bit_vector
`e - f`	•legal, arithmetic with integers is valid
`IF (b<c) ...`	• legal, comparing the same data types is okay
`IF (b>=a) ...`	• illegal, we are comparing two signals with different sizes
`IF (f/=e) ...`	• legal, valid comparison amongst same data types
`IF (e>d) ...`	• illegal, comparing different datatypes
`b sra 1`	• legal, shifting a bit_vector
`c srl -2`	• legal, shifting a bit_vector
`f ror 3`	• illegal, shifting a integer
`e*3`	• legal, multiplying a integer
`5**5`	• legal exponential operation
`f/4`	• legal, division with integers
`e/3`	• legal, division with integers
`d <= c`	• illegal, there is a size mismatch
`d(6 DOWNTO 3) := b`	• legal, size and data-type matches
`e <= d`	• illegal, operation with two different data-types
`f := 100`	• legal assignment as 100 is within the valid range

3.a) Main code:

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

entity decoder is
        port (ena: in std_logic;
        sel: in std_logic_vector (2 downto 0);
        x : out std_logic_vector (7 downto 0)
        );
end decoder;

architecture generic_decoder of decoder is
begin
        process (ena,sel)
        variable temp1 : std_logic_vector(x'high downto 0);
        variable temp2 : integer range 0 to x'high;

        begin
                temp1:= (OTHERS => '1');
                temp2:=0;
                if (ena='1') then
                        for i in sel'range loop
                                if (sel(i)='1') then
                                  temp2:=2*temp2+1;
                                else
                                        temp2:=2*temp2;
                                end if;
                        end loop;
                        temp1(temp2):='0';
                        end if;
                        x<=temp1;
        end process;

end architecture;
```

Output-waveform generated by the test-bench on the next page:



Test-bench:

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

entity testdecoder is
end;

architecture bench of testdecoder is
  component decoder
        port (ena: in std_logic;
        sel: in std_logic_vector (2 downto 0);
        x : out std_logic_vector (7 downto 0)
        );
  end component;

  signal ena  : std_logic;
  signal sel  : std_logic_vector (2 downto 0);
  signal x    : std_logic_vector (7 downto 0);

begin

 ena   <= '0', '1' after 100ns;
 sel   <= "000"                --0
 , "001" after 50ns            --1
 , "000" after 100ns   --0
 , "001" after 150ns   --1
 , "010" after 200ns   --2
 , "011" after 250ns   --3
 , "100" after 300ns   --4
 , "101" after 350ns   --5
 , "110" after 400ns   --6
 , "111" after 450ns;  --7

m: decoder port map (ena,sel,x);

end bench;
```

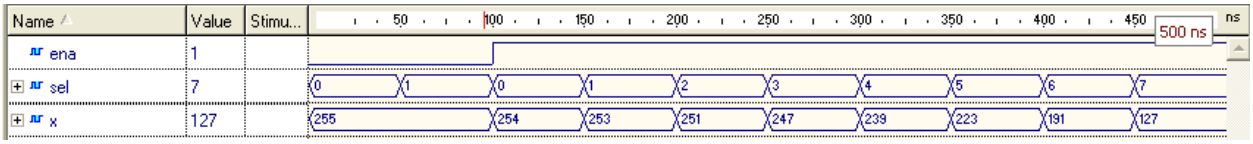
3.b) Main program:

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

entity decoder is
        generic( n : integer := 3);
        port (ena: in std_logic;
        sel: in std_logic_vector (n-1 downto 0);
        x : out std_logic_vector ((2**n)-1 downto 0)
        );
end decoder;

architecture generic_decoder of decoder is
begin
        process (ena,sel)
        variable temp1 : std_logic_vector(x'high downto 0);
        variable temp2 : integer range 0 to x'high;

        begin
                temp1:= (OTHERS => '1');
                temp2:=0;
                if (ena='1') then
                        for i in sel'range loop
                                if (sel(i)='1') then
                                  temp2:=2*temp2+1;
                                else
                                        temp2:=2*temp2;
                                end if;
                        end loop;
                        temp1(temp2):='0';
                        end if;
                        x<=temp1;
        end process;

end architecture;
```

Test-bench when `generic( n : integer := 3);`

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

entity testdecoder is
end;

architecture bench of testdecoder is
constant n : positive := 3;
component decoder
        generic (n: integer := n);
        port (ena: in std_logic;
        sel: in std_logic_vector (n-1 downto 0);
        x : out std_logic_vector ((2**n)-1 downto 0)
        );
  end component;

  signal ena  : std_logic;
  signal sel  : std_logic_vector (n-1 downto 0);
  signal x    : std_logic_vector ((2**n)-1 downto 0);

begin

 ena   <= '0', '1' after 100ns;
 sel   <= "000"                --0
 , "001" after 50ns            --1
 , "000" after 100ns   --0
 , "001" after 150ns   --1
 , "010" after 200ns   --2
 , "011" after 250ns   --3
 , "100" after 300ns   --4
 , "101" after 350ns   --5
 , "110" after 400ns   --6
 , "111" after 450ns;  --7

 m: decoder generic map(n) port map (ena,sel,x);

end bench;
```

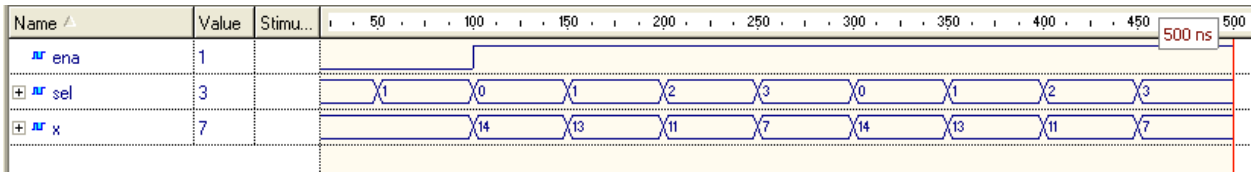
Waveform when `generic( n : integer := 3);`



Test-bench when `generic( n : integer := 2);`

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

entity testdecoder is
end;

architecture bench of testdecoder is
constant n : positive := 2;
component decoder
        generic (n: integer := n);
        port (ena: in std_logic;
        sel: in std_logic_vector (n-1 downto 0);
        x : out std_logic_vector ((2**n)-1 downto 0)
        );
  end component;

  signal ena  : std_logic;
  signal sel  : std_logic_vector (n-1 downto 0);
  signal x    : std_logic_vector ((2**n)-1 downto 0);

begin

 ena   <= '0', '1' after 100ns;
 sel   <= "00"                --0
 , "01" after 50ns            --1
 , "00" after 100ns    --0
 , "01" after 150ns    --1
 , "10" after 200ns    --2
 , "11" after 250ns    --3
 , "00" after 300ns    --0
 , "01" after 350ns    --1
 , "10" after 400ns    --2
 , "11" after 450ns;   --3

 m: decoder generic map(n) port map (ena,sel,x);

end bench;
```

Waveform when `generic( n : integer := 2);`



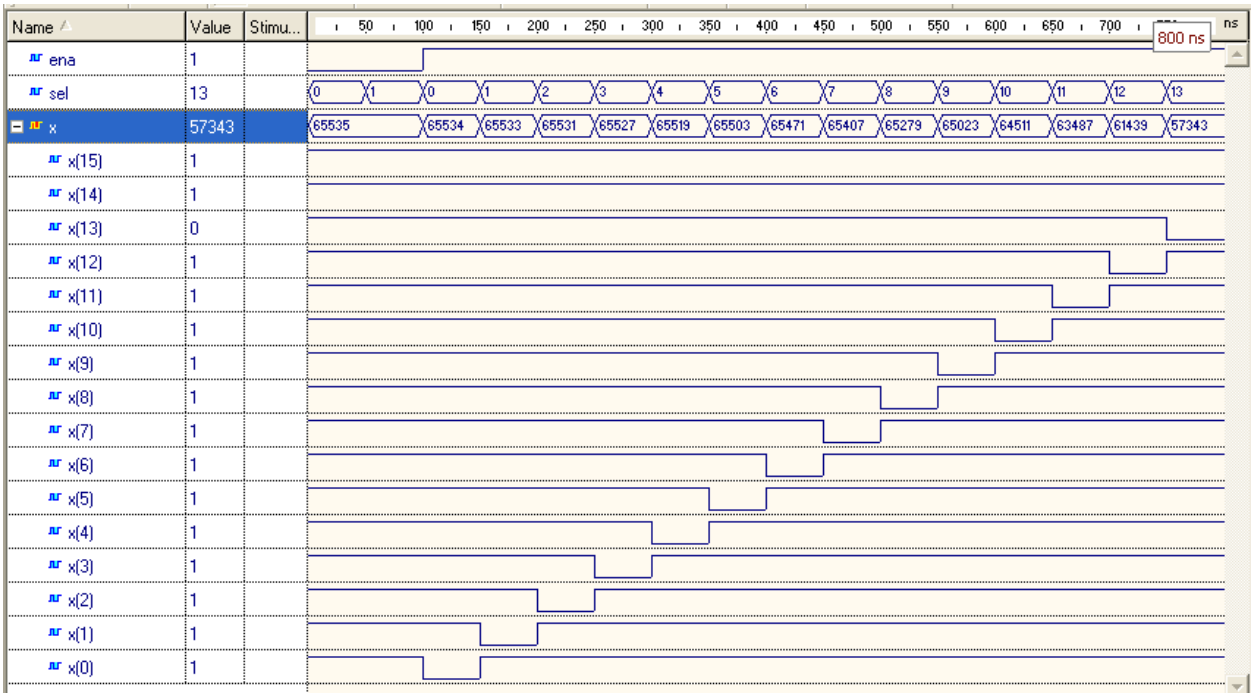
3.c) Main program:

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

entity decoder is
        generic( n : integer := 4);
        port (ena: in std_logic;
        sel: in integer range (2**n)-1 downto 0;
        x : out std_logic_vector ((2**n)-1 downto 0)
        );
end decoder;

architecture generic_decoder of decoder is
begin
        process (ena,sel)
        variable temp1 : std_logic_vector(x'high downto 0);

        begin
                temp1:= (OTHERS => '1');
                if (ena='1') then
                        temp1(sel):='0';
                end if;
                x<=temp1;
        end process;

end architecture;
```

Test-bench for `generic( n : integer := 4):`

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

entity testdecoder is
end;

architecture bench of testdecoder is
constant n : positive := 4;
component decoder
        generic (n: integer := n);
        port (ena: in std_logic;
        sel: in integer range (2**n)-1 downto 0;
        x : out std_logic_vector ((2**n)-1 downto 0)
        );
  end component;

  signal ena  : std_logic;
  signal sel: integer range (2**n)-1 downto 0;
  signal x    : std_logic_vector ((2**n)-1 downto 0);

begin

 ena   <= '0'           , '1' after 100ns;
 sel   <= 0 , 1 after 50ns
        , 0 after 100ns  , 1 after 150ns
        , 2 after 200ns  , 3 after 250ns
        , 4 after 300ns  , 5 after 350ns
        , 6 after 400ns  , 7 after 450ns
        , 8 after 500ns  , 9 after 550ns
        , 10 after 600ns , 11 after 650ns
        , 12 after 700ns , 13 after 750ns;

 m: decoder generic map(n) port map (ena,sel,x);

end bench;
```

Waveform for `generic( n : integer := 4):`



4.

List of used operators

- `:= , <= , XOR`

List of used attributes

- input'range

List of used generics

- generic (n: integer := 7)