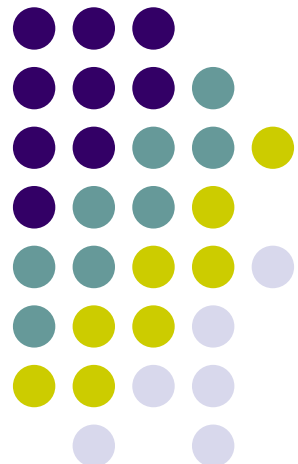
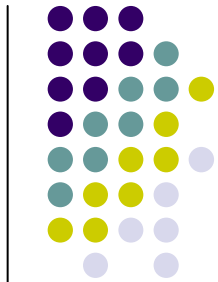
EE-4313
COMPUTER ENGINEERING DESIGN PROJECT - I

Prof. Ramesh Karri
rkarri@duke.poly.edu
Office: LC001, phone #: 4011
TA: Shyam Mantravadi
smantr01@utopia.poly.edu

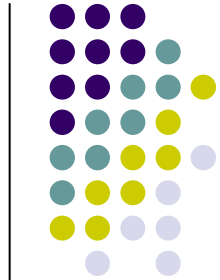


EE-4313: overview



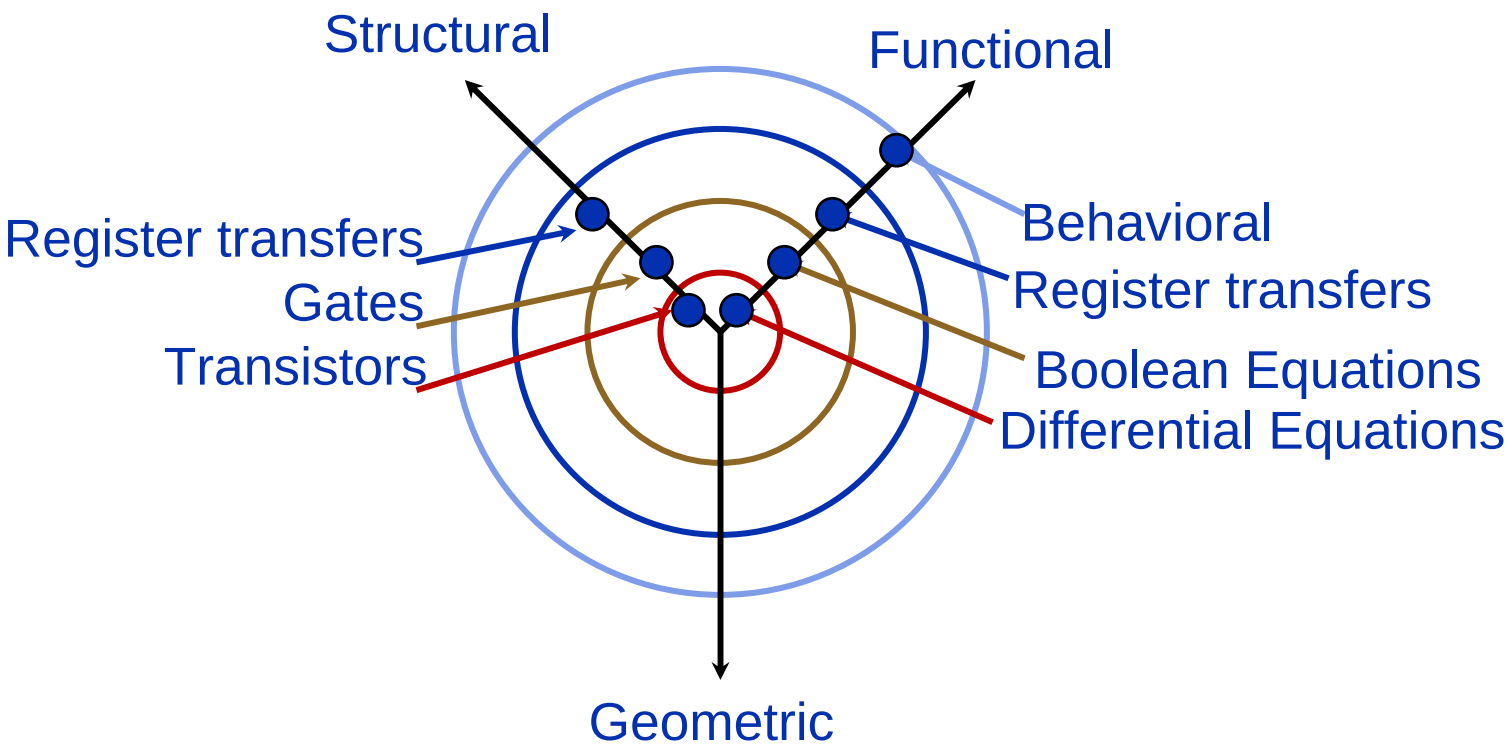
- Grading policy
  - Quizzes (at the beginning of lecture) 10%
  - Exam 1 : 20%
  - Exam 2 : 20%
  - Lab : 54 %
    - VHDL code for RC5
    - Simulation for RC5 (**Meaningful** simulation snapshots)
    - Possibly downloading designs onto FPGA and verify them
- Office hours
  - Tues, Wed, Thur 11:00 AM to -12:00 PM.

Introduction

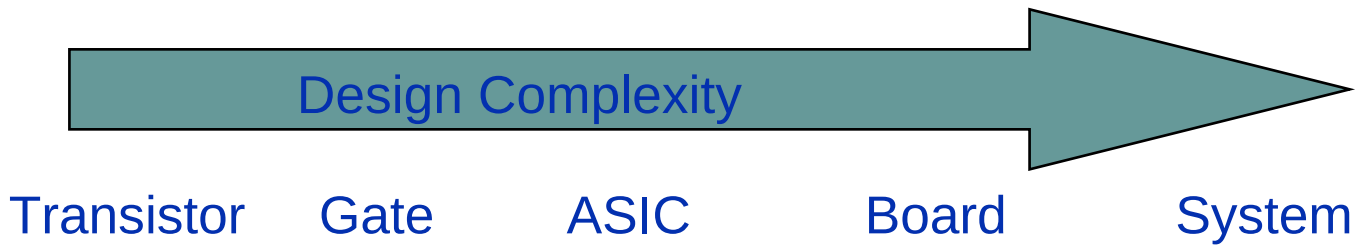


- WHAT
  - VHSIC Hardware Description Language
  - V=Very High Speed Integrated Circuit
  - HDL=Describes behavior, data flow, structure
- WHY
  - Shrinking design cycle
  - Simulation and formal verification
  - Synthesis
  - Reliable design process (eliminate design errors)
  - Documentation and requirements specification
- HOW
  - Focus on Synthesizable VHDL Models
  - Verify behavior and timing using various CAD tools

The Y-Chart

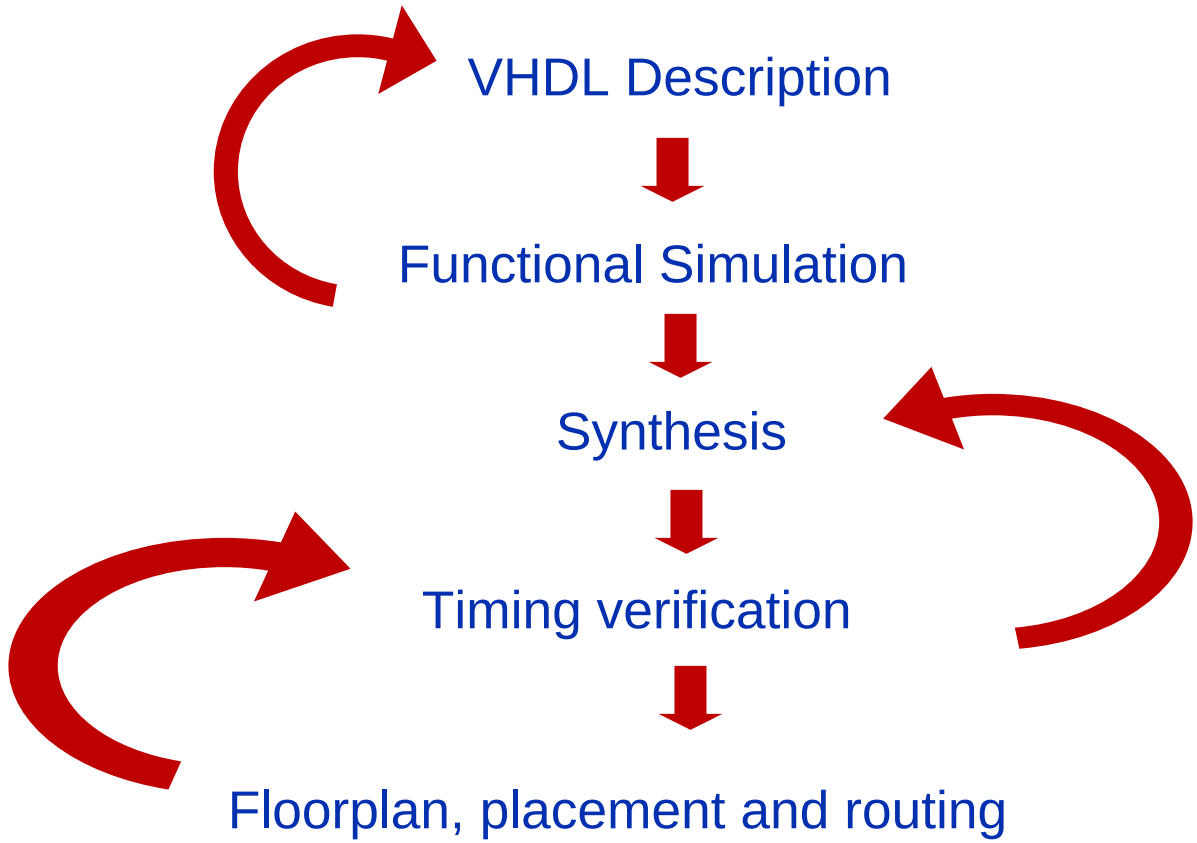

Structural
Functional
Register transfers
Gates
Transistors
Behavioral
Register transfers
Boolean Equations
Differential Equations
Geometric

VHDL Modeling Capability



- Develop Gate, Block and ASIC level VHDL models

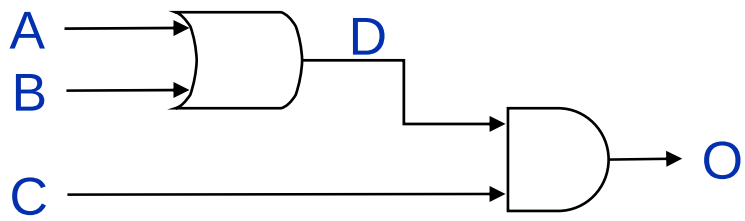
Top-down VHDL-based design methodology



Describing a Digital Design in VHDL

- External Interface (Entity declaration)
  - Describes ports of a module
    - Input-output
    - Width
    - Type
- Internal Implementation (Architecture)
  - Describes
    - Structure
    - Functionality

Entity Declaration



entity name **port name** **port direction**

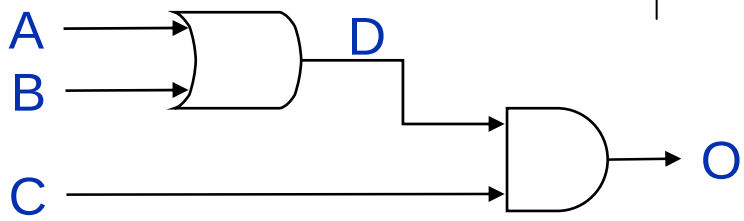
```
entity or_and is
    port ( a, b, c: in bit;
                o: out bit );
end entity or_and;
```

port type

Architecture Body

entity name

```
architecture Do_it of or_and is
  signal D: bit;
  begin
    D <= A or B;
    O <= D and C;
end Do_it;
```




```
architecture Do_it of or_and is
 signal D: bit;
  begin
   O <= D and C;
   D <= A or B;
end Do_it;
```

Concurrent Statements

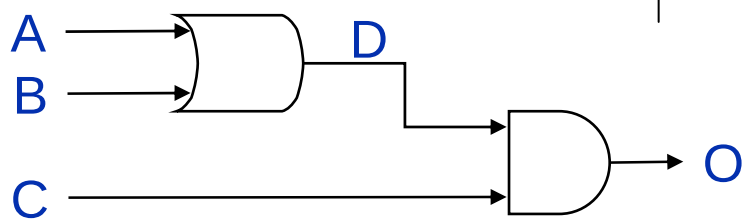
- Can be synthesized and simulated
- Order of statements is not important

OR-AND Gate VHDL Model

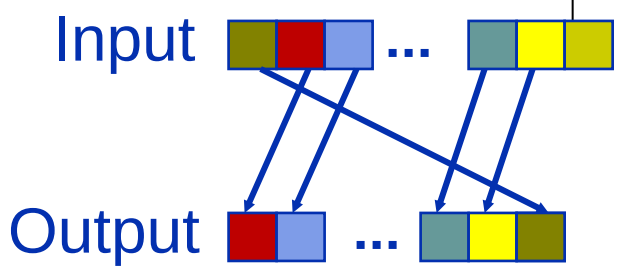
```
library IEEE;
use IEEE.std_logic_1164.all;

entity or_and is
  port ( a, b, c: in bit;
         o: out bit );
end entity or_and;

architecture Do_it of or_and is
signal D: bit;
begin
    D <= A or B;
    O <= D and C;
end Do_it;
```



Circular left shift by One



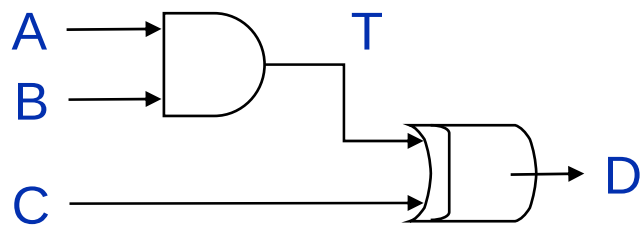

```
library IEEE;
use IEEE.std_logic_1164.all;

entity Circular_shift_left_1 is
port (
  In28 : in std_logic_vector(27 downto 0);
  Out28 : out std_logic_vector(27 downto 0));
end Circular_shift_left_1;

architecture Do_it of Circular_shift_left_1 is
  begin
     Out28(27 downto 1) <= In28(26 downto 0);
     Out28(0) <= In28(27);
  end Do_it;
```

NANDXOR Model 1

```
Entity NANDXOR is
port (
        A,B, C :in bit; D: out bit);
end NANDXOR;
architecture RTL of NANDXOR is
signal T : bit;
begin
   p0 : T <= A nand B after 2 ns;
   p1 : process (T, C)
     begin
          D <= T xor C after 3 ns;
     end process p1;
end RTL;
```



Simulation of NANDXOR

--NANDXOR

--force file

force A 0 0;

force A 1 20;

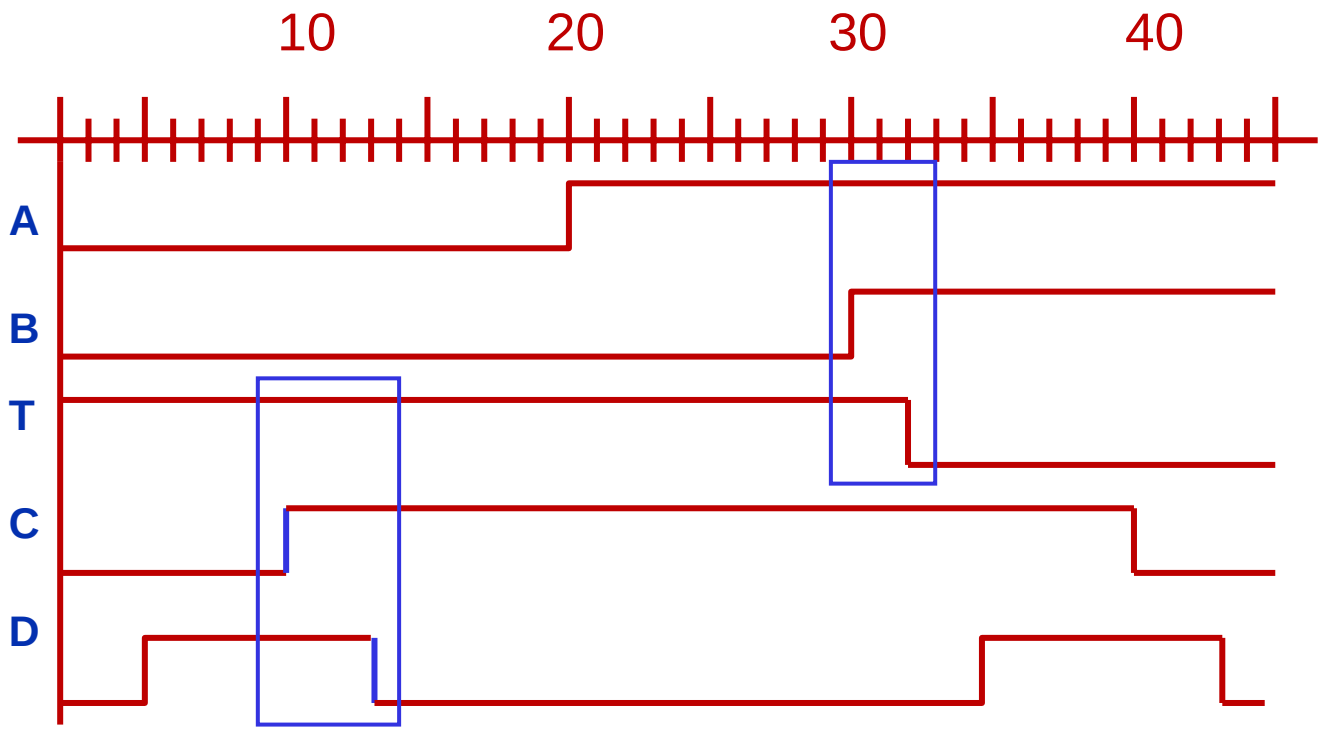
force B 0 0;

force B 1 30;

force C 0 0;

force C 1 10;

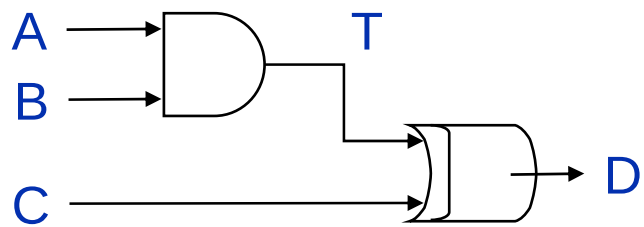
force C 0 40;



NANDXOR Model 2

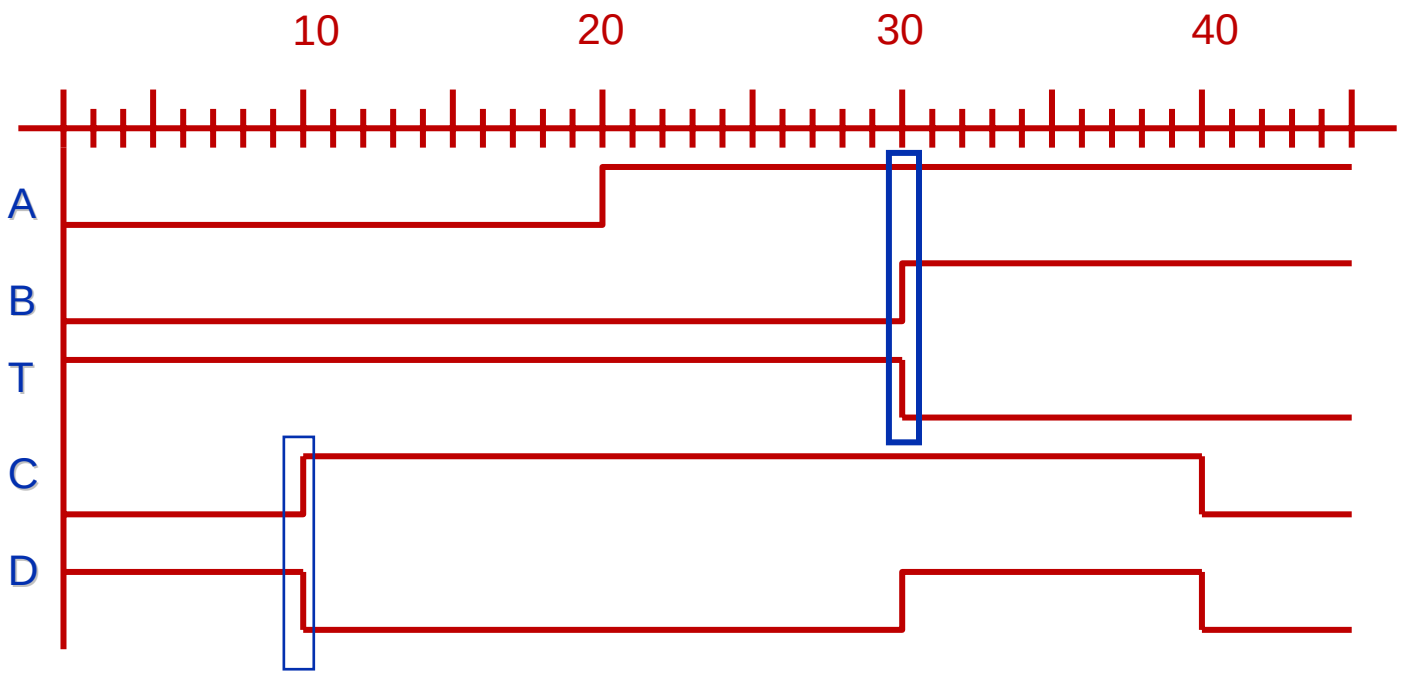
```
Entity NANDXOR is
  port (
          A,B, C: in bit; D: out bit);
 end NANDXOR;


architecture DELTA of NANDXOR is
signal T : bit;
begin
      p0 : T <= A nand B;
      p1 : process (T, C)
      begin
            D<= T xor C;
      end process p1;
end DELTA;
```

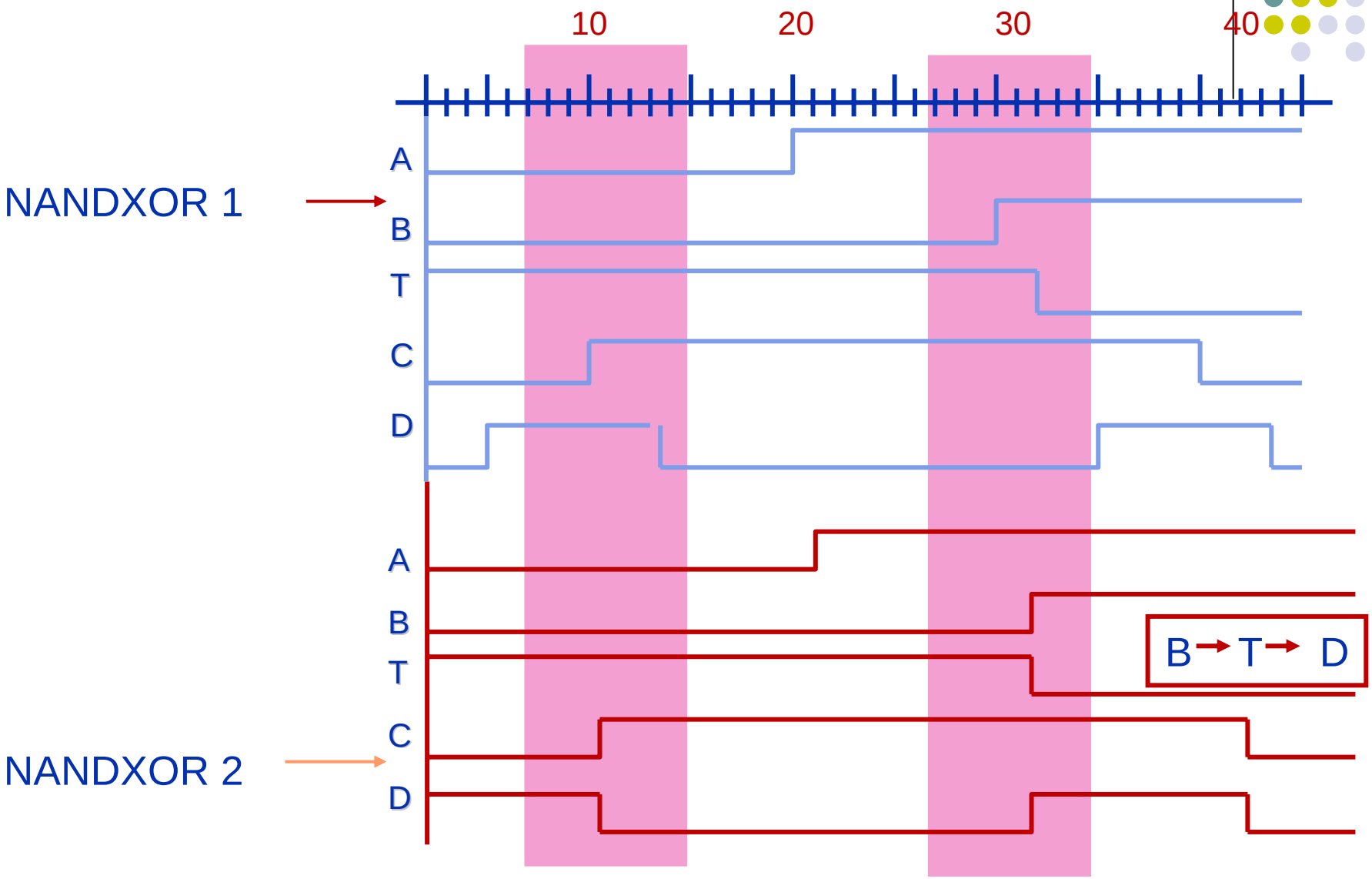


Simulation of NANDXOR 2

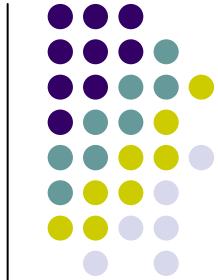
```
--NANDXOR2
--force file
force A 0 0;
force A 1 20;
force B 0 0;
force B 1 30;
force C 0 0;
force C 1 10;
force C 0 40;
```



Simulation



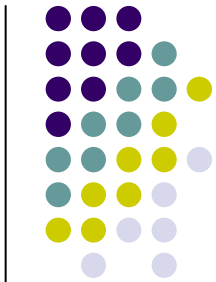
VHDL Data Objects



- Constant
  - Cannot be updated
  - Can be declared inside or outside a process

- Variable
  - Can be initialized
  - Updated as soon as a variable assignment stmt is executed
  - “:=“ is used for variable assignment
  - Used for local storage in processes, procedures and functions
- Signal
  - Models a physical wire
  - If not initialized, default value is the left-most value of its type
  - Ports are signals
  - “<=“ is used for signal assignment

VHDL Basic Data Types



- Some Predefined VHDL Data Types
  - Bit: ‘0’ or ‘1’
  - Boolean: FALSE or TRUE
  - Integer: range $-(2^{-31}-1)$ to $+(2^{+31}-1)$

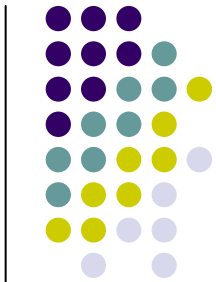
- Enumerated Data Types
  - Example 1:

```
type color is (red, green, yellow);
signal A: color;
A <= green;
```

  Some Attributes
  - color’left is red
  - color’right is yellow
  - Example 2:

```
type state_type is (s0, s1, s2, s3);
signal state: state_type := s1;
```

VHDL Record Data Types



```
constant length: integer := 8;
subtype byte_vector is bit_vector (length-1 downto 0);
type byte_and_ix is record
        byte: byte_vector;
        ix: integer range 0 to length;
end record;

signal x, y, z: byte_and_ix;
```

Write (read) to a record by field name

```
        x.byte <="11001100";
        x.ix <= 2;
```

Write (read) to a record by record name

```
        y <=x;
```

VHDL Array Data Type

- Unconstrained Arrays

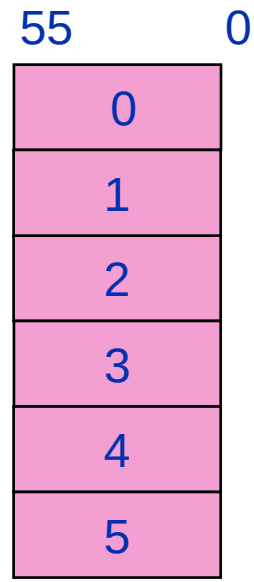
  type std_logic_vector is array (natural range<>) of std_logic;
  type bit_vector is array (integer range <>) of bit;
  variable sample_vector: bit_vector (2 downto -3);

- Constrained Arrays

  subtype skey56set is std_logic_vector(55 downto 0);
  type key_56_tbl is array (0 to 5) of skey56set;

- Array Attributes

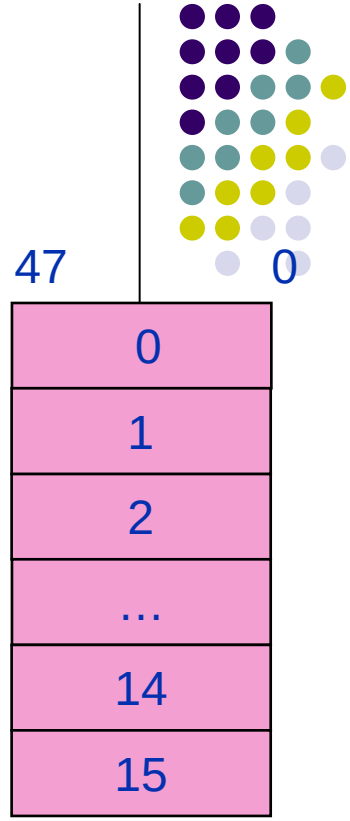
  sample_vector'left is 2 sample_vector'right is -3
  sample_vector'high is 2 sample_vector'low is -3
  sample_vector'length is 5 sample_vector'range is (2 downto -3)
  sample_vector'reverse_range is (-3 to 2)



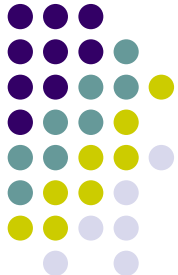
VHDL Package

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

package DEAL128PACK is
        subtype Nkeyi is std_logic_vector(47 downto 0);
        type fixed_key_tbl is array(0 to 15) of NKeyi;
end DEAL128PACK;
```



Array Attributes

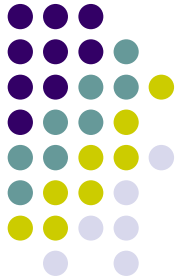


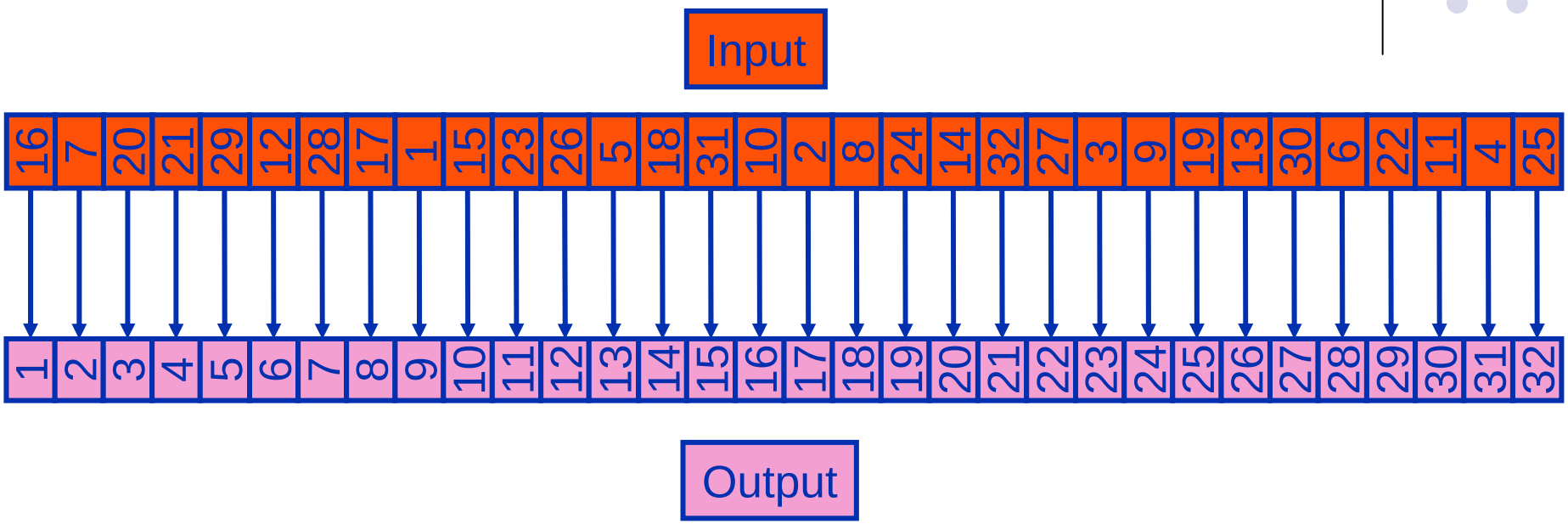
```
library IEEE; use IEEE.std_logic_1164.all;
entity P_box is port (
        In32 : in std_logic_vector(31 downto 0);
        Out32 : out std_logic_vector(31  downto 0));
end P_box;

architecture Do_it of P_box is
type In32array_type is array (0 to 31) of integer;
constant In32table : In32array_type := In32array_type '(
        15, 6,19,20, 28, 11, 27, 16, 0,14, 22, 25 4,17, 30, 9,
        1, 7, 23,13, 31, 26, 2, 8,18,12, 29, 5, 21,10, 3,24);
begin
        process (In32)
        begin
                for i in In32table'Range loop
                        Out32(i) <= In32(In32table(i));
                end loop;
        end process
end Do_it;
```

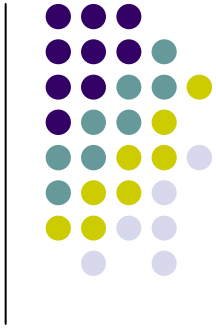
Works for any sized table

Permutation Box





Data Dependent Rotation



```
LIBRARY IEEE;
USE                    IEEE.STD_LOGIC_1164.ALL;
ENTITY   leftrotate IS
  PORT (a: in STD_LOGIC_VECTOR(31 DOWNTO 0);
                b: in STD_LOGIC_VECTOR(31 DOWNTO 0);
                o: out STD_LOGIC_VECTOR(31 DOWNTO 0));
END leftrotate;

ARCHITECTURE rtl OF leftrotate IS
SIGNAL ab_xor: STD_LOGIC_VECTOR(31 DOWNTO 0);
BEGIN
ab_xor <= a XOR b;
```

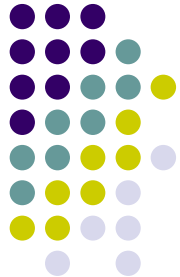
O=(A XOR B) <<<B

```
WITH b(4 DOWNTO 0) SELECT
O<=  ab_xor(30 DOWNTO 0) & ab_xor(31) WHEN "00001",
       ab_xor(29 DOWNTO 0) & ab_xor(31 DOWNTO 30) WHEN "00010",
       ab_xor(28 DOWNTO 0) & ab_xor(31 DOWNTO 29) WHEN "00011",
       ab_xor(27 DOWNTO 0) & ab_xor(31 DOWNTO 28) WHEN "00100",
       ab_xor(26 DOWNTO 0) & ab_xor(31 DOWNTO 27) WHEN "00101",
       ab_xor(25 DOWNTO 0) & ab_xor(31 DOWNTO 26) WHEN "00110",
       ab_xor(24 DOWNTO 0) & ab_xor(31 DOWNTO 25) WHEN "00111",
       ab_xor(23 DOWNTO 0) & ab_xor(31 DOWNTO 24) WHEN "01000",
       ab_xor(22 DOWNTO 0) & ab_xor(31 DOWNTO 23) WHEN "01001",
       ab_xor(21 DOWNTO 0) & ab_xor(31 DOWNTO 22) WHEN "01010",
       ab_xor(20 DOWNTO 0) & ab_xor(31 DOWNTO 21) WHEN "01011",
       ab_xor(19 DOWNTO 0) & ab_xor(31 DOWNTO 20) WHEN "01100",
       ab_xor(18 DOWNTO 0) & ab_xor(31 DOWNTO 19) WHEN "01101",
       ab_xor(17 DOWNTO 0) & ab_xor(31 DOWNTO 18) WHEN "01110",
       ab_xor(16 DOWNTO 0) & ab_xor(31 DOWNTO 17) WHEN "01111",
       ab_xor(15 DOWNTO 0) & ab_xor(31 DOWNTO 16) WHEN "10000",
```

O =(A XOR B) <<<B

```
        ab_xor(14 DOWNTO 0) & ab_xor(31 DOWNTO 15) WHEN "10001",
        ab_xor(13 DOWNTO 0) & ab_xor(31 DOWNTO 14) WHEN "10010",
        ab_xor(12 DOWNTO 0) & ab_xor(31 DOWNTO 13) WHEN "10011",
        ab_xor(11 DOWNTO 0) & ab_xor(31 DOWNTO 12) WHEN "10100",
        ab_xor(10 DOWNTO 0) & ab_xor(31 DOWNTO 11) WHEN "10101",
        ab_xor(9 DOWNTO 0) & ab_xor(31 DOWNTO 10) WHEN "10110",
        ab_xor(8 DOWNTO 0) & ab_xor(31 DOWNTO 9) WHEN "10111",
        ab_xor(7 DOWNTO 0) & ab_xor(31 DOWNTO 8) WHEN "11000",
        ab_xor(6 DOWNTO 0) & ab_xor(31 DOWNTO 7) WHEN "11001",
        ab_xor(5 DOWNTO 0) & ab_xor(31 DOWNTO 6) WHEN "11010",
        ab_xor(4 DOWNTO 0) & ab_xor(31 DOWNTO 5) WHEN "11011",
        ab_xor(3 DOWNTO 0) & ab_xor(31 DOWNTO 4) WHEN "11100",
        ab_xor(2 DOWNTO 0) & ab_xor(31 DOWNTO 3) WHEN "11101",
        ab_xor(1 DOWNTO 0) & ab_xor(31 DOWNTO 2) WHEN "11110",
        ab_xor(0) & ab_xor(31 DOWNTO 1) WHEN "11111",
        ab_xor WHEN OTHERS;
END rtl;
```

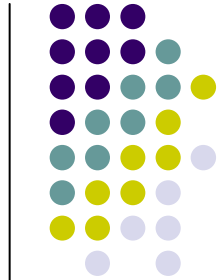
Two Approaches



```
WITH b(4 DOWNTO 0) SELECT
   ab_rot<=ab_xor(30 DOWNTO 0) & ab_xor(31) WHEN “00001”,
           ab_xor(29 DOWNTO 0) & ab_xor(31 DOWNTO 30) WHEN “00010”,
           ……
           ab_xor(0) & ab_xor(31 DOWNTO 1) WHEN “11111”,
           ab_xor WHEN OTHERS;
```

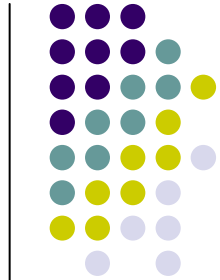
```
PROCESS(b, ab_xor)
BEGIN
   CASE b(4 DOWNTO 0) IS
      WHEN “00001”=>     ab_rot<= ab_xor(30 DOWNTO 0) & ab_xor(31) ;
      WHEN “00010”=>     ab_rot<=ab_xor(29 DOWNTO 0) & ab_xor(31
                                    DOWNTO 30);
          ……
      WHEN OTHERS=>      ab_rot<=ab_xor;
   END CASE;
END PROCESS;
```

Exercise 1

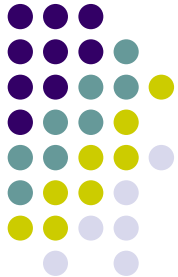


- Simulate Left Rotate VHDL Model
  - Functional Simulation using modelsim
- Synthesize the Design
  - Perform Timing Simulation
- Draw and Discuss the Synthesized Circuit
- Understand use of “select” VHDL statement

Functional Simulation



- Create project folder
  - C:\ee4313\proj1
- Save your VHDL code in project folder
  - leftrotate.vhd
- Start ModelSim

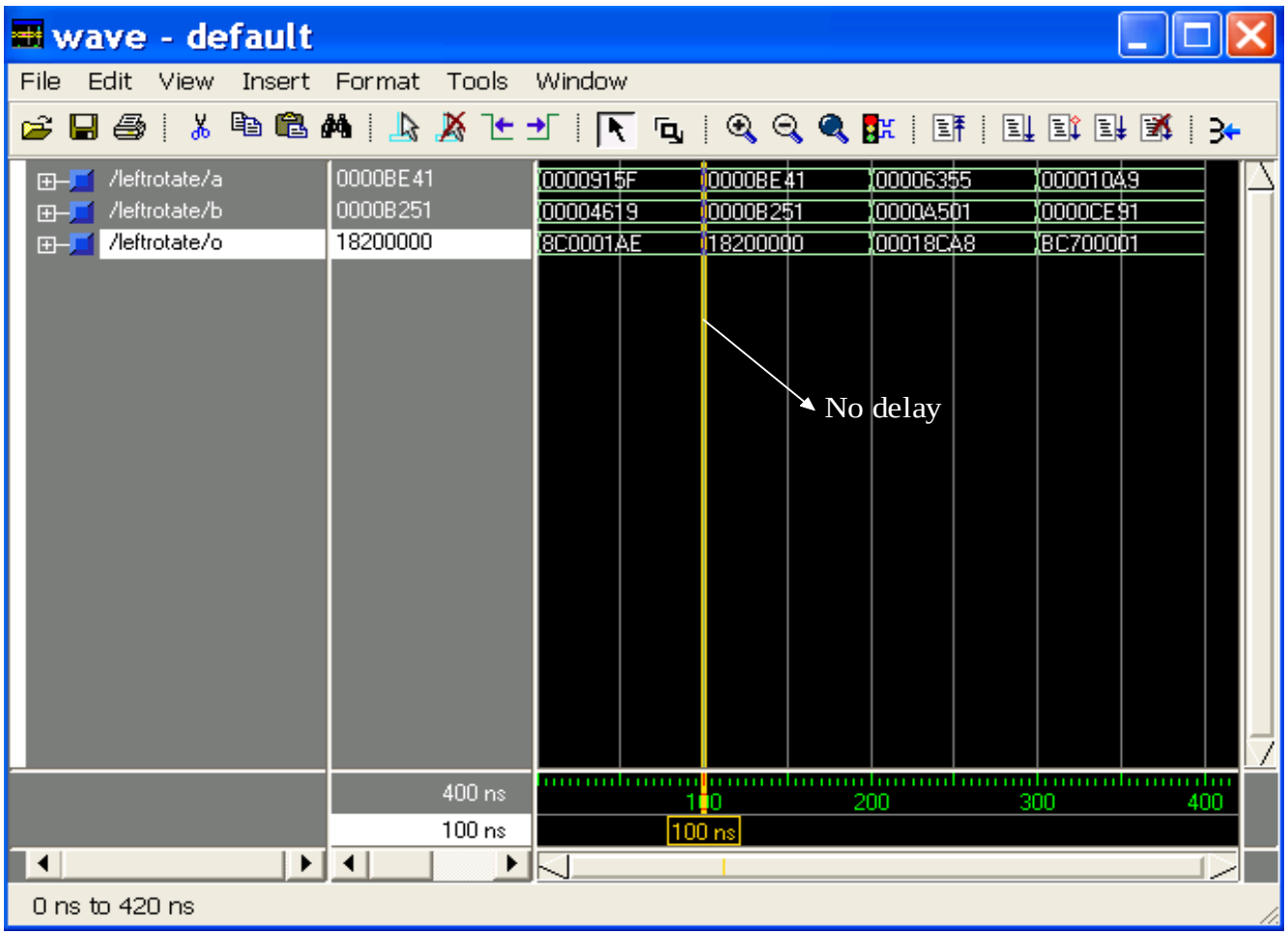


Functional Simulation

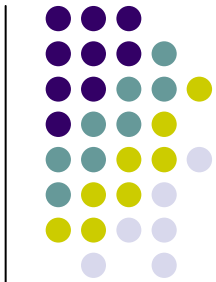
- In ModelSim console window, type following commands in sequence

```
cd {c:/ee4313/proj1}
vlib work                    -- create work library
vcom leftrotate.vhd          -- “compile” your design (the vhdl model)
vsim leftrotate              -- simulate your design
view wave                    -- start waveform viewer
add wave a b o               -- add input/output signals in the waveform viewer
-- assign input signals (use hexadecimal, time unit ns)
force a 16#915F 0, 16#BE41 100, 16#6355 200, 16#10A9 300
force b 16#4619 0, 16#B251 100, 16#A501 200, 16#CE91 300
force 400                    -- simulate 400ns
```

Functional Simulation



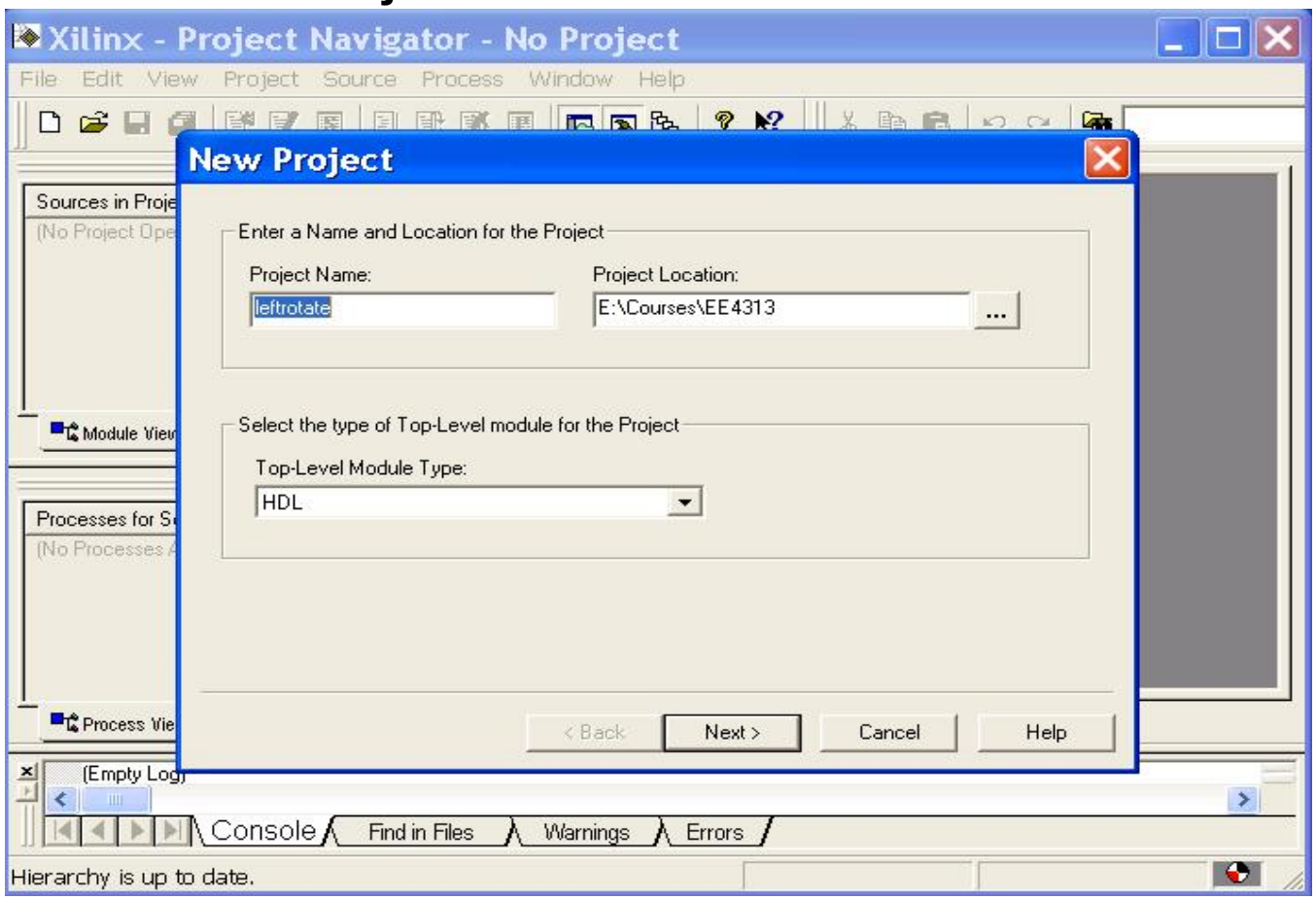
FPGA Synthesis and Implementation



- Start Xilinx Project Navigator

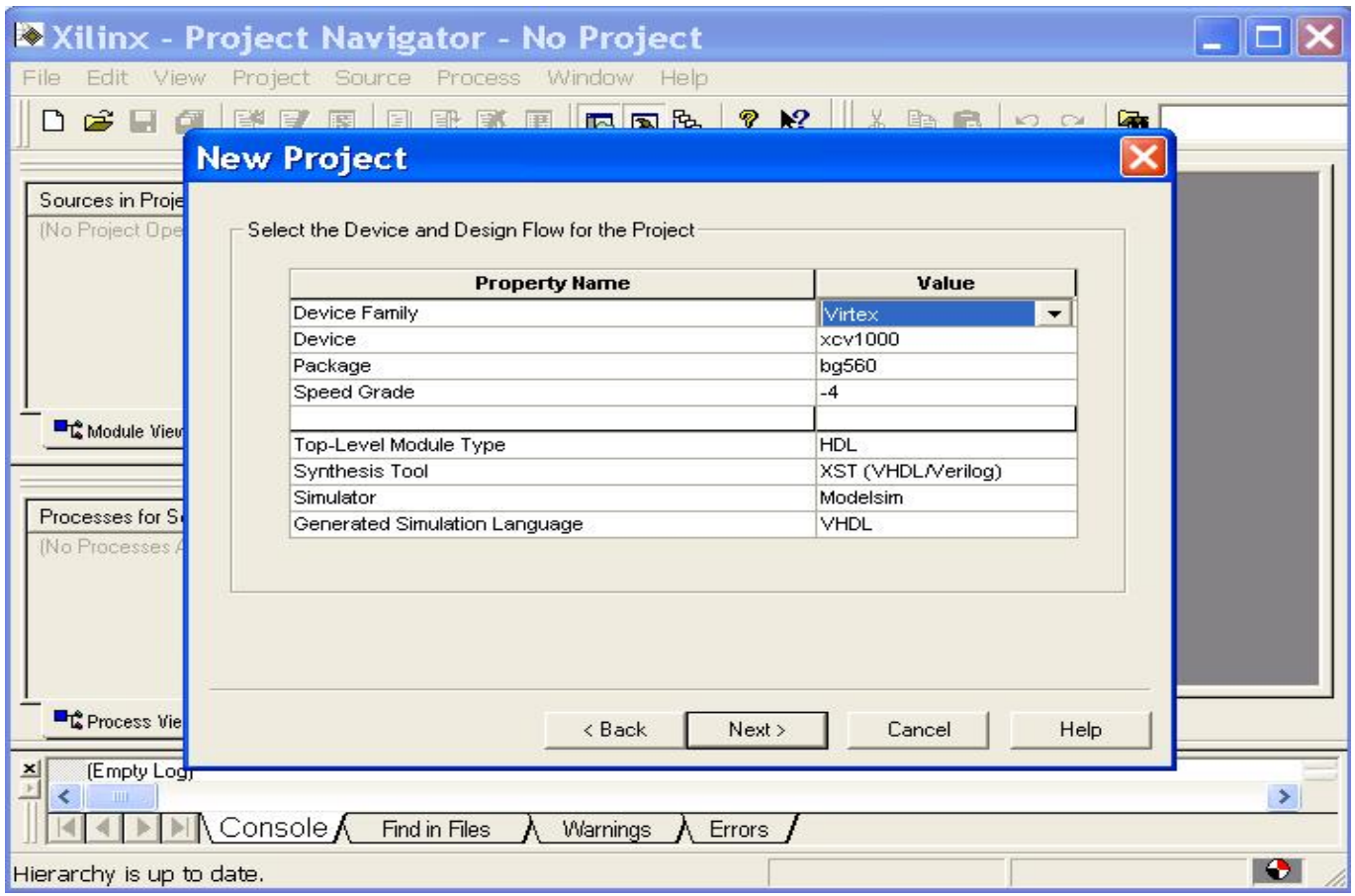
FPGA Synthesis

- File->New Project…



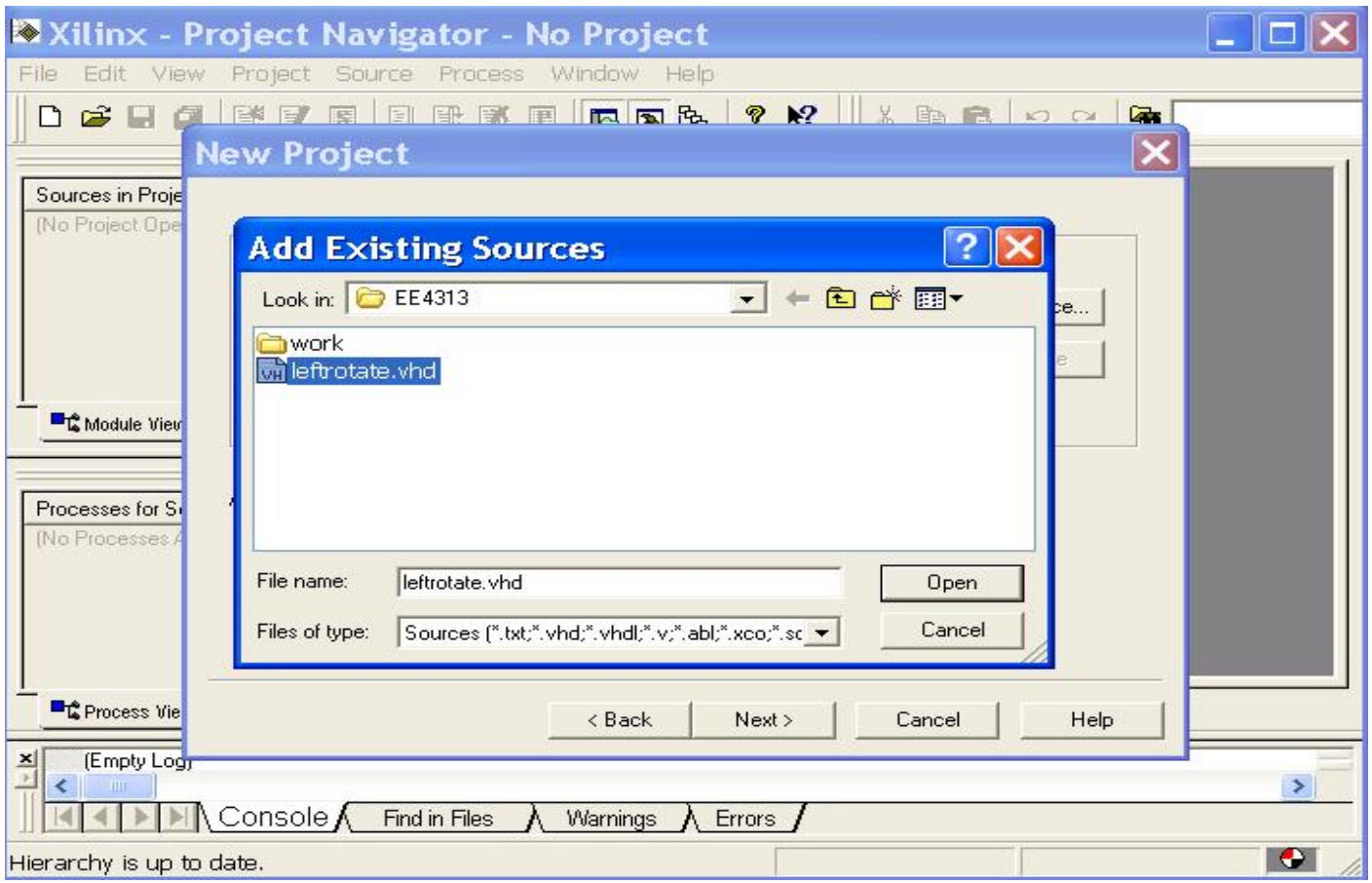
FPGA Synthesis

- Next>



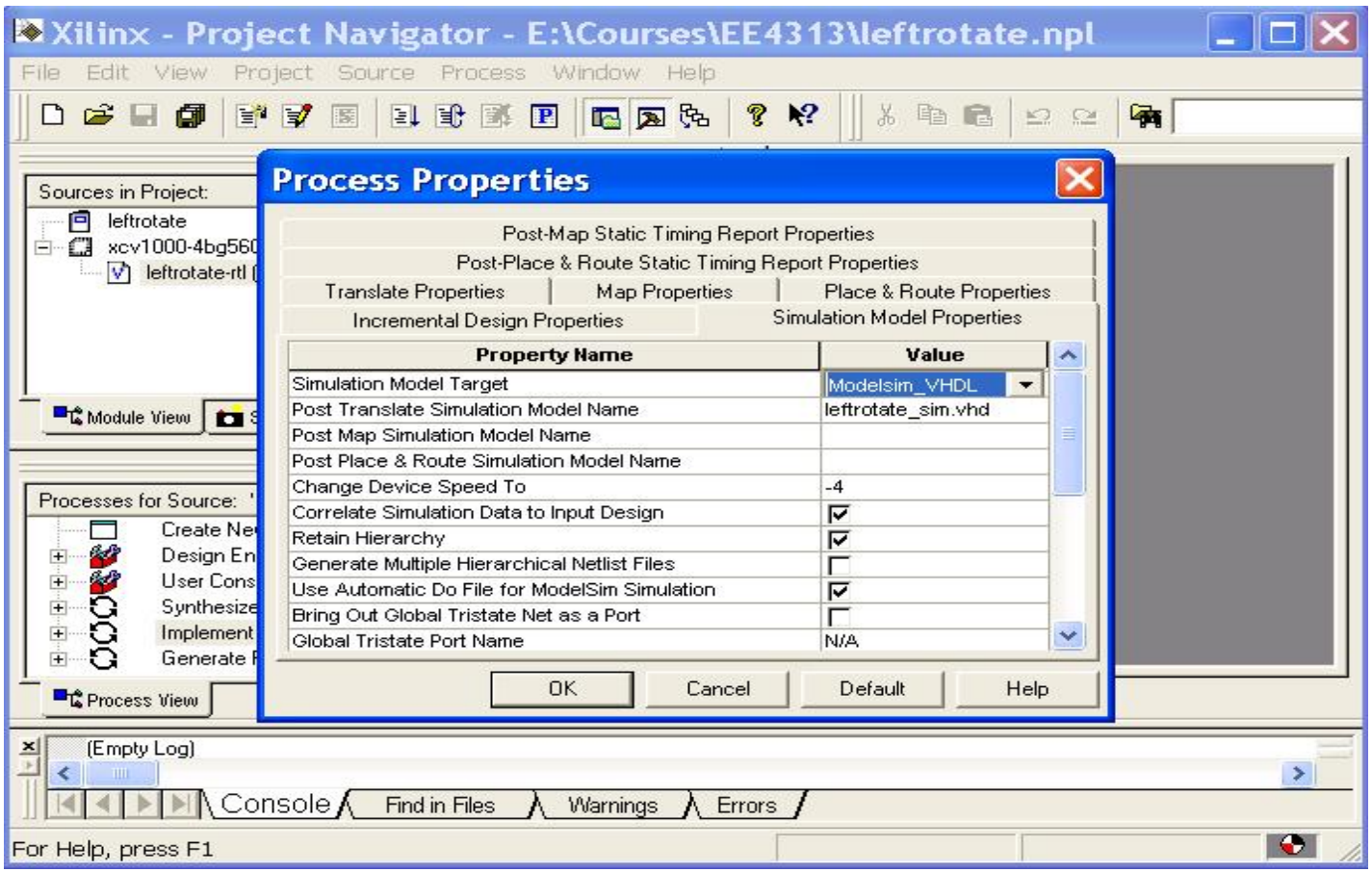
FPGA Synthesis

- Next>

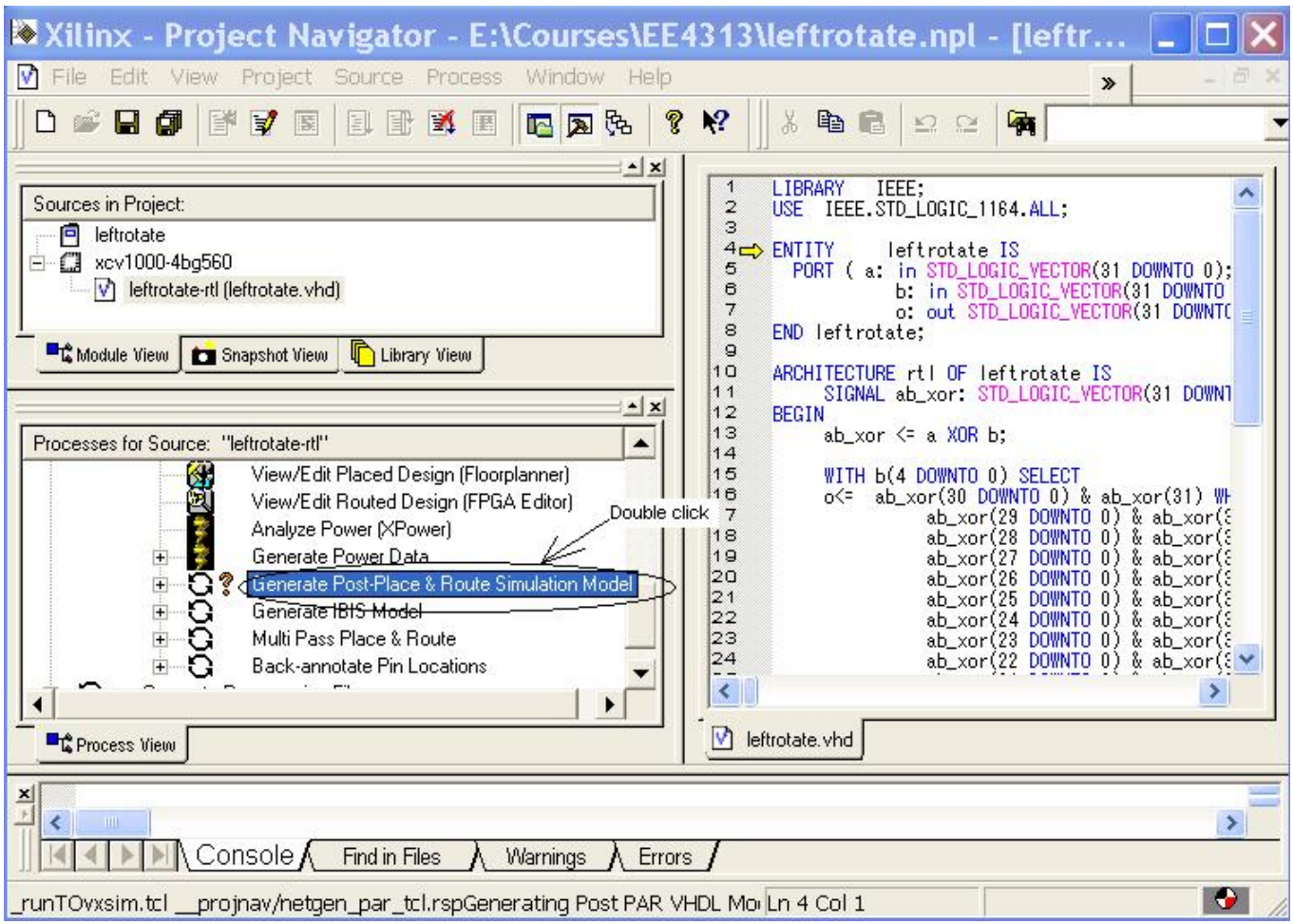


FPGA Synthesis

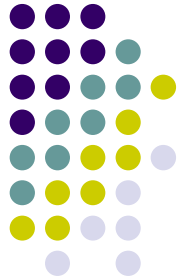
- Next>Simulation Model Properties



FPGA Synthesis



Timing Simulation



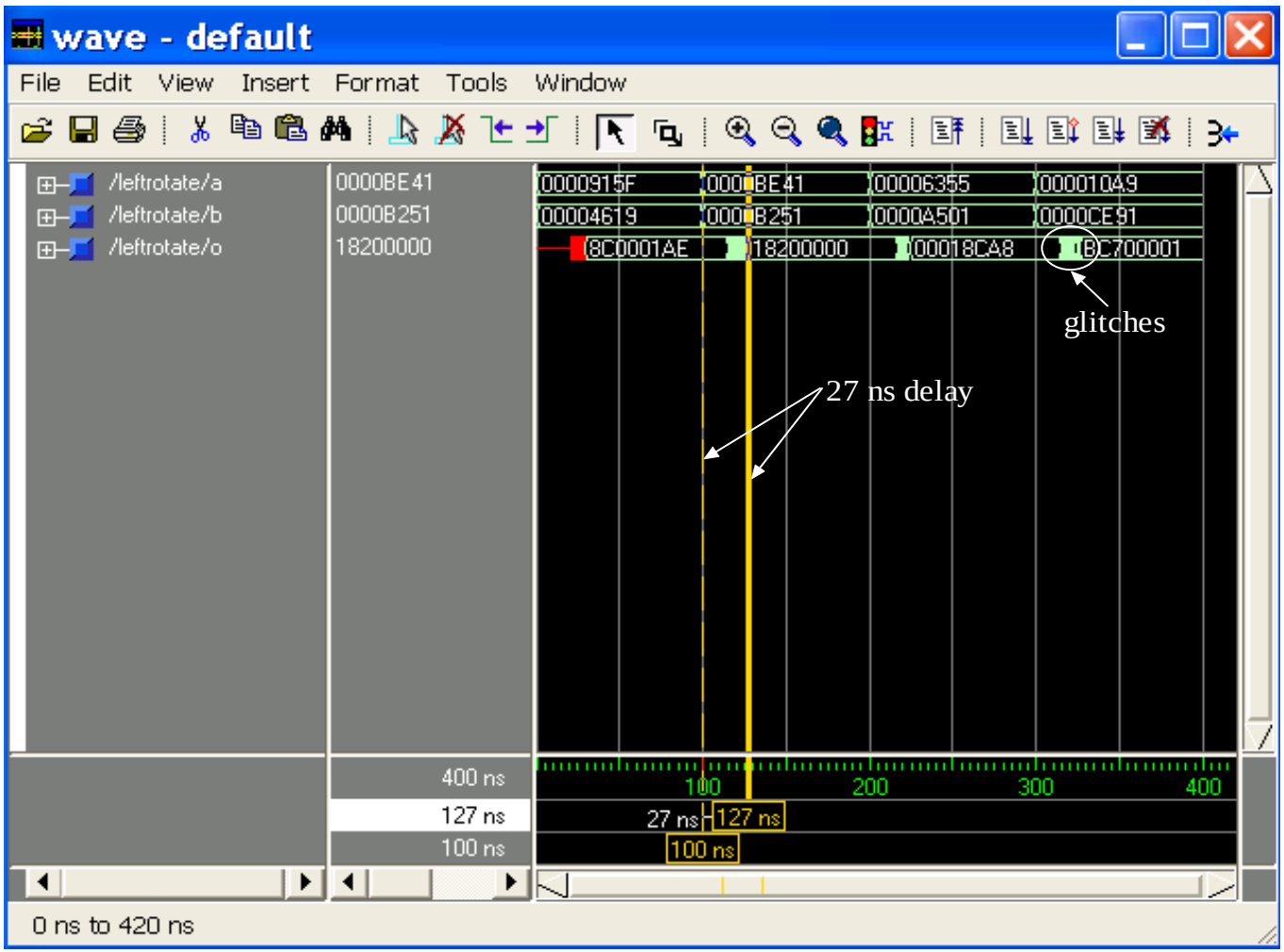
- Make sure timing simulation models are successfully generated
  - leftrotate_sim.vhd and leftrotate_sim.sdf
  - sdf stands for “standard delay format”
- Start ModelSim

Timing Simulation

- In ModelSim console window, type in following commands in sequence

```
cd {c:/ee4313/proj1}
vlib work                          -- create a work library
-- map simprim to pre-compiled standard library
Vmap simprim c:/modeltech_5.6d/xilinx/simprim
vcom leftrotate_sim.vhd     -- “compile” simulation model
-- run timing simulation with delay information (leftrotate_sim.sdf)
vsim -sdftyp /=E:/Courses/EE4313/leftrotate_sim.sdf work.leftrotate
view wave                          -- start waveform viewer
add wave a b o                     -- add input/out signals in the waveform viewer
-- assign input signals (use hexadecimal, time unit ns)
force a 16#915F 0, 16#BE41 100, 16#6355 200, 16#10A9 300
force b 16#4619 0, 16#B251 100, 16#A501 200, 16#CE91 300
force 400                          -- simulate 400ns
```

Timing Simulation



Exercise 2

- Modify the previous model to implement O=(A XOR B) >>>B (right rotate)
  - Simulate the Model
  - Synthesize the Design
  - Draw and Explain the Synthesized Design