University of Leicester

Programmable Electronics using FPGAs

Course Outline

- What is the next 10 weeks about?
- What does it mean to design a chip?
- FPGA platform
- Hands-on experience on training boards

This week

- Concurrency, Moore's Law, et al
- What is an FPGA?
- Our language and tool: VHDL and Quartus-II
- Example: compiling to hardware
- Lab work: build your own chip
- Housekeeping
- Assessment

Firstly: what is concurrency?

- **<u>Question:</u>** What do you think "concurrency" means?

Firstly: what is concurrency?

- **Answer:**

concurrent (adj.) 1. running together, going on side by side, occurring together. 2. cooperating, agreeing.

Because hardware is an ideal platform for supporting concurrency, in this course we will study concurrency (and programming patterns for concurrency) in some detail.

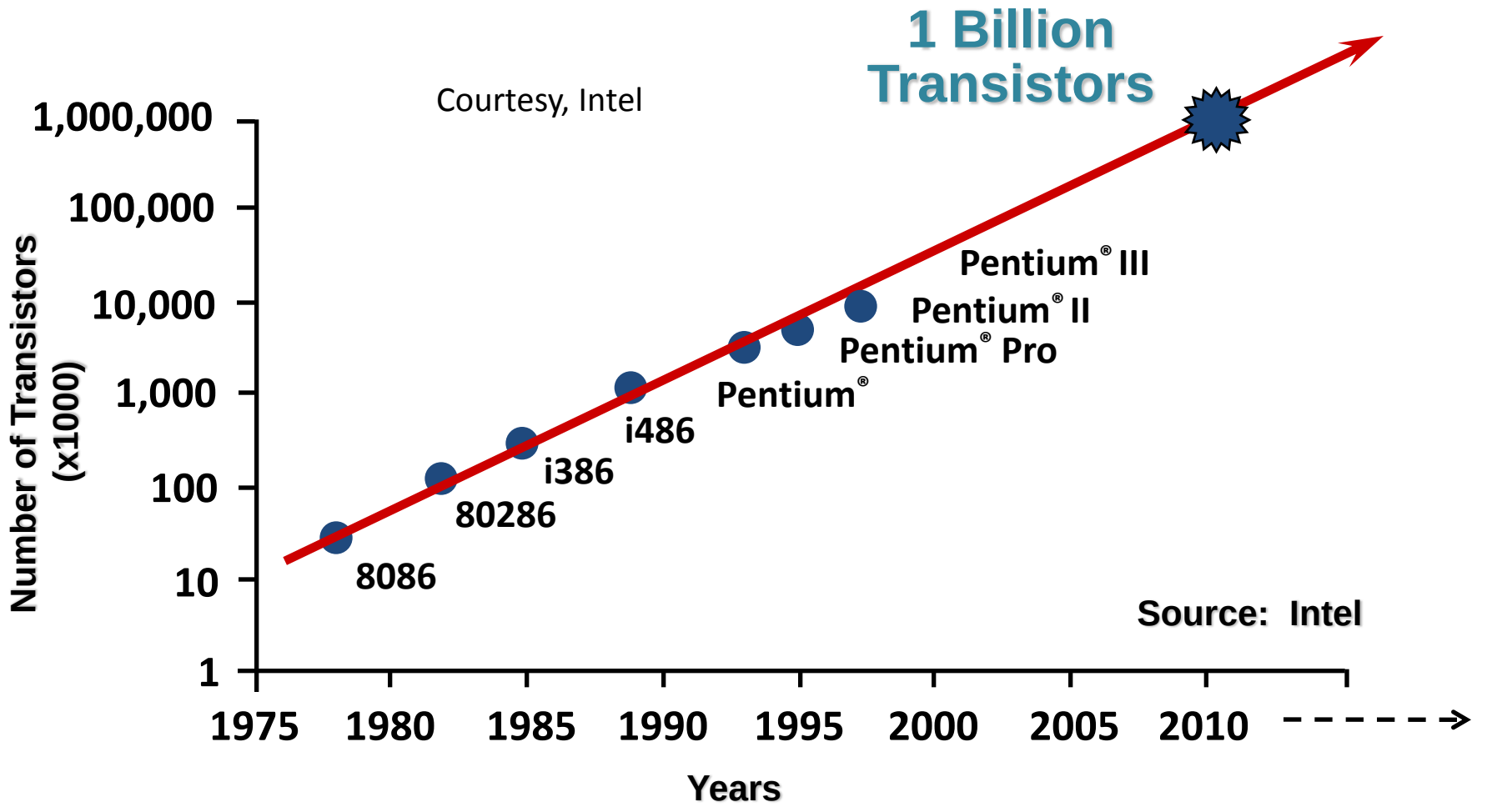
The aim of Programmable Electronics

- Know what reconfigurable hardware is, and it's relation to software and hardware systems;
- Appreciate (theoretical) issues in building and reasoning about practical, concurrent, communicating systems and the benefits that concurrency offers;
- Be able to develop and program inherently concurrent applications;
- Demonstrate competence with VHDL and Quartus-II;
- Apply these principles to the design, analysis and implementation of FPGA circuits.

What about the tools?

- An FPGA is a Field Programmable Gate Array – it is the hardware that we will use on this course;
- VHDL is a hardware language for FPGA design;
- Quartus-II is the tool will use to simulate (develop) the design and implement the logic on FPGA hardware.

Moore's law



Microprocessor transistor counts double every 2 years

Integrated circuits (ICs)

- Today digital IC technology is used in:
  - Microprocessors
  - Memory chips
  - Application Specific Integrated Circuits (ASICs)
  - Programmable Logic Devices (CPLD/FPGA)

What is an FPGA?

- Field Programmable Gate Array (FPGA)
  - Semiconductor device containing programmable logic blocks with programmable interconnects.
  - These features allow us to implement digital logic consisting of millions of logic gates.
  - It is a fully reconfigurable device – sometimes called "reconfigurable hardware".
  - Advantages include quick time to market, simple upgrades, lower unit production, and blurring software/hardware distinctions.
  - Disadvantages include high power consumption, slower clock rates and expensive for low production systems.

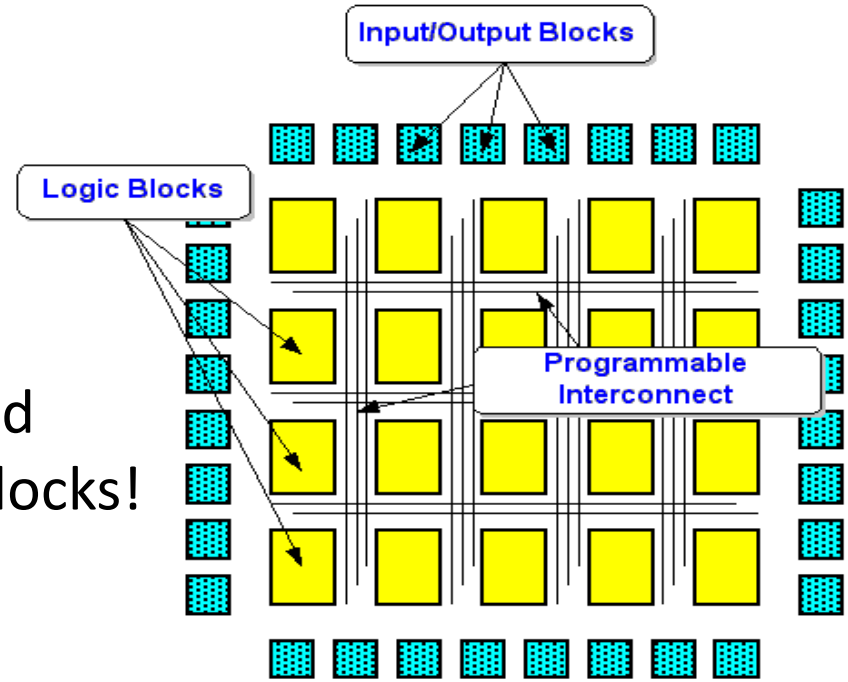
Where are FPGAs used?

- Traditional ASIC domains, such as:
  - Digital signal processing, Vision, Cryptography (especially real-time) and Solid state devices.

- Now starting to encroach on traditional software domain – Hardware/Software Co-Design

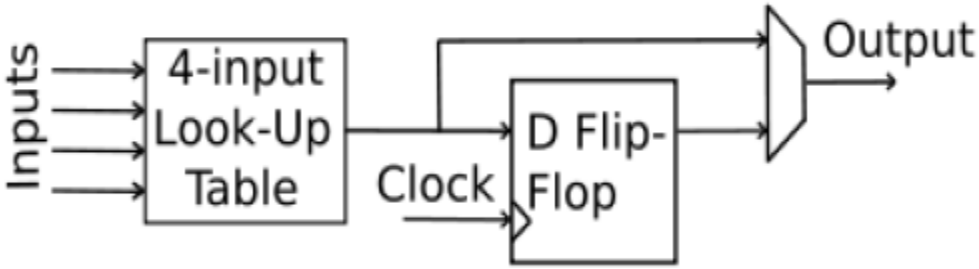
FPGA – Generic architecture

- Overview
  - An FPGA consists of three programmable blocks;
    - Logic blocks
    - Input/Output (IO) blocks
    - Routing blocks
  - By using specific languages and tools we can program these blocks!



FPGA – Basic logic block

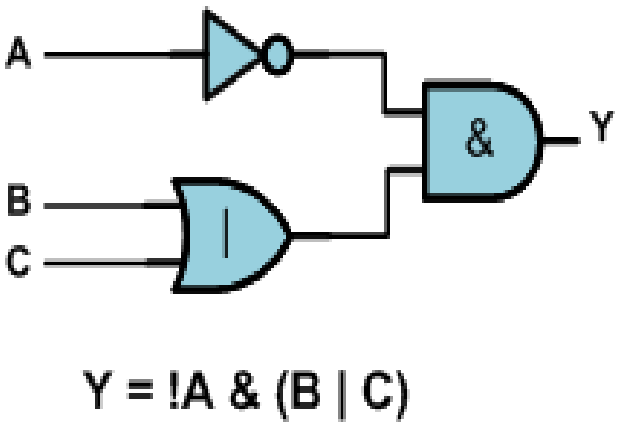
- How does “reconfigurable” work?
  - The basic logic block contains a Lookup Table (LUT) and a Flip-Flop (FF);



  - For combinatorial logic a lookup table is programmed with a logical function;
  - The flip-flop serves as a latch;
  - Modern FPGAs also have other resources such as dedicated memory, processor cores and arithmetic blocks.

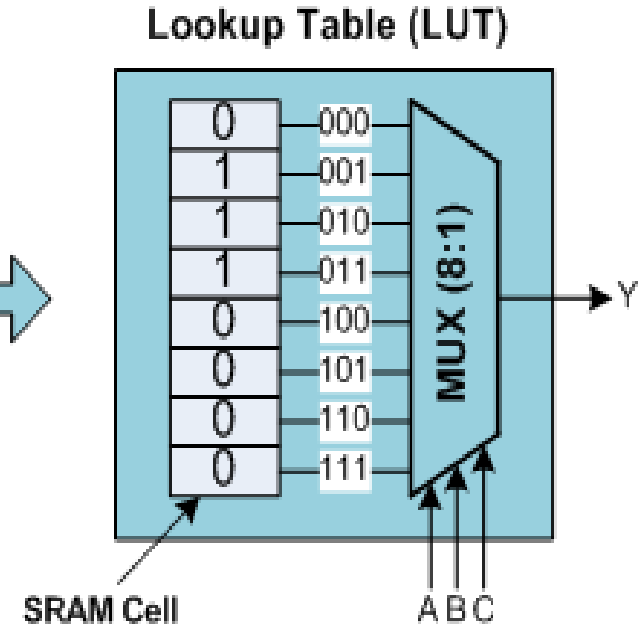
An example – Combinatorial logic on FPGA

- Configuring a LUT:
  - *N*-LUT consists of 2^N SRAM cells which connect with a 2^N:1 Multiplexer (MUX).



Truth Table

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0



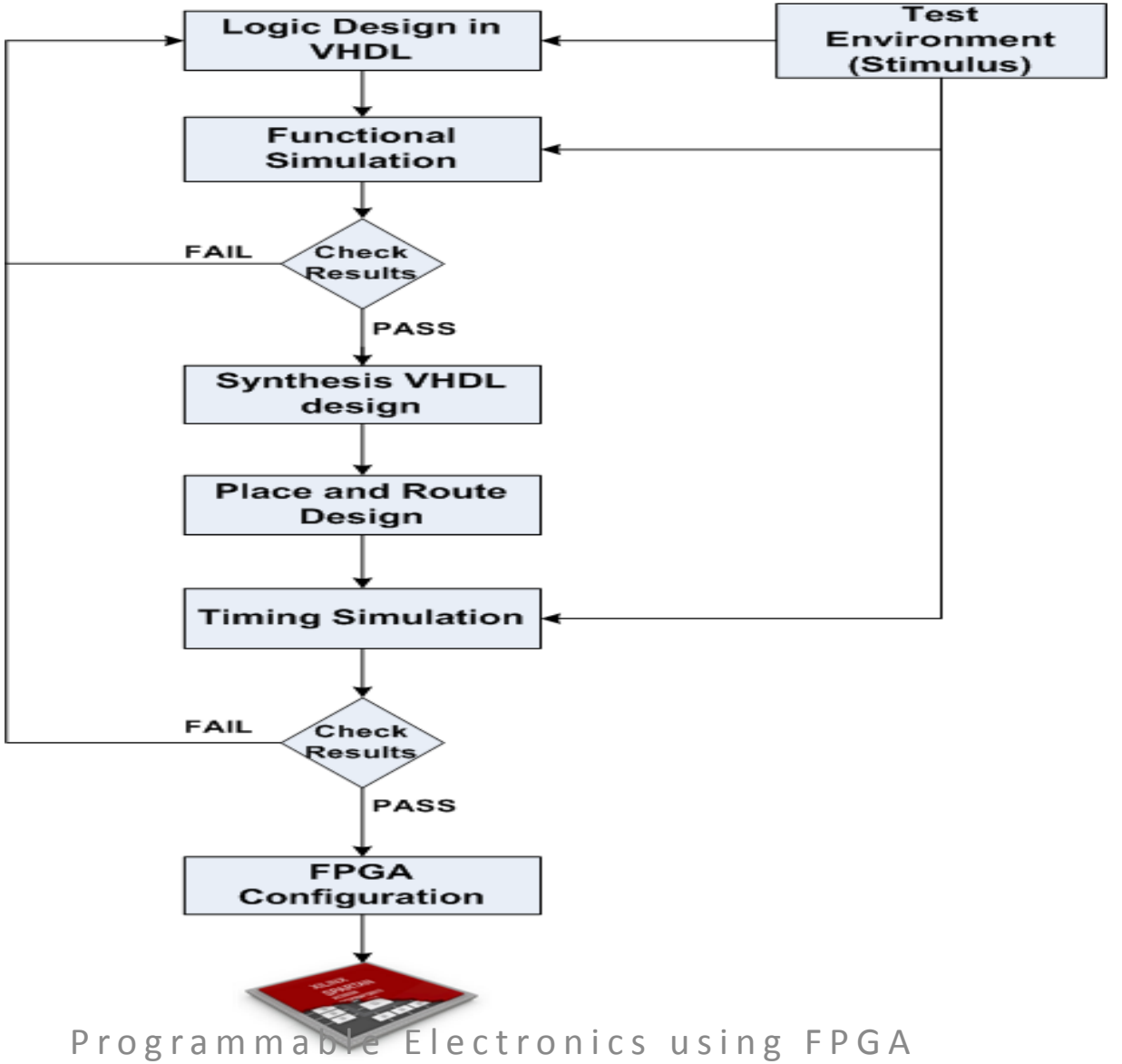
What is VHDL?

- FPGA programming language - VHDL
  - **V**ery **H**igh Speed Integrated Circuit Hardware **D**escription **L**anguage;
  - It is not a procedural language such as C, BASIC and Assembly etc.
  - VHDL supports concurrency;
  - VHDL is a strongly typed language where most of its syntax is driven from Ada;
  - VHDL-2008 is the latest version (available as IEEE standard 1076-2008, supported by most commercial tools).

Altera DE0 Development Board

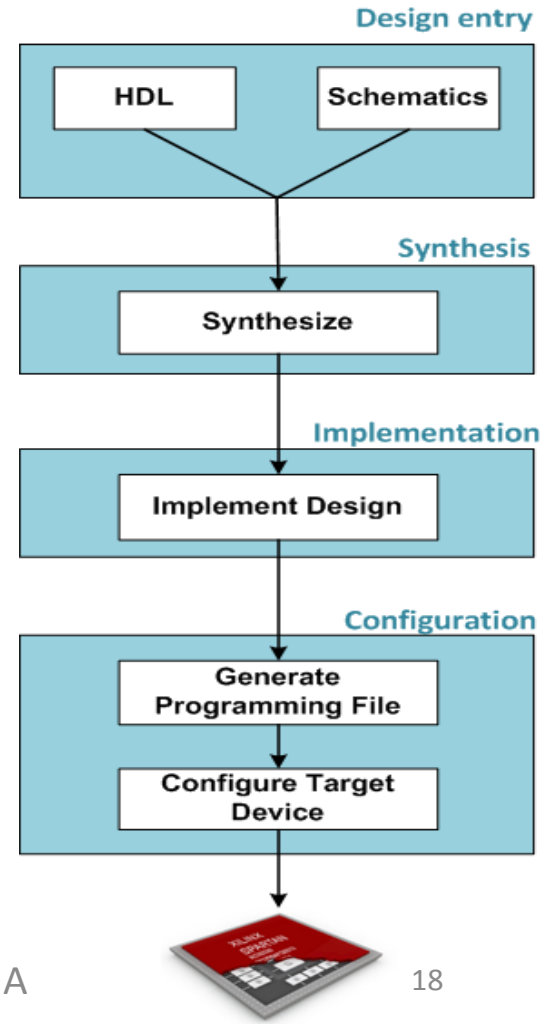
- We use this kit for experiments in LAB.
- Features include flash memory, SDRAM, SD memory card slot, mouse/keyboard port, VGA port and 7-segment displays. We will concentrate on programming the FPGA.

FPGA design flow using VHDL



Using Quartus-II: the basics

- Quartus-II is a Integrated Software Environment for Altera FPGAs;
- Used for creating design files, simulation, synthesis, implementation and configuration;
- Basic design flow:
  - **Design entry** – HDL or Schematics
  - **Synthesis** – Convert design into netlist
  - **Implementation** – Translate, Map and Place & Routing for specific FPGA
  - **Configuration** – Program FPGA



FPGA Design example: Shift Register

1. Draw the circuit (block) diagram of shift register on paper including the inputs and outputs;

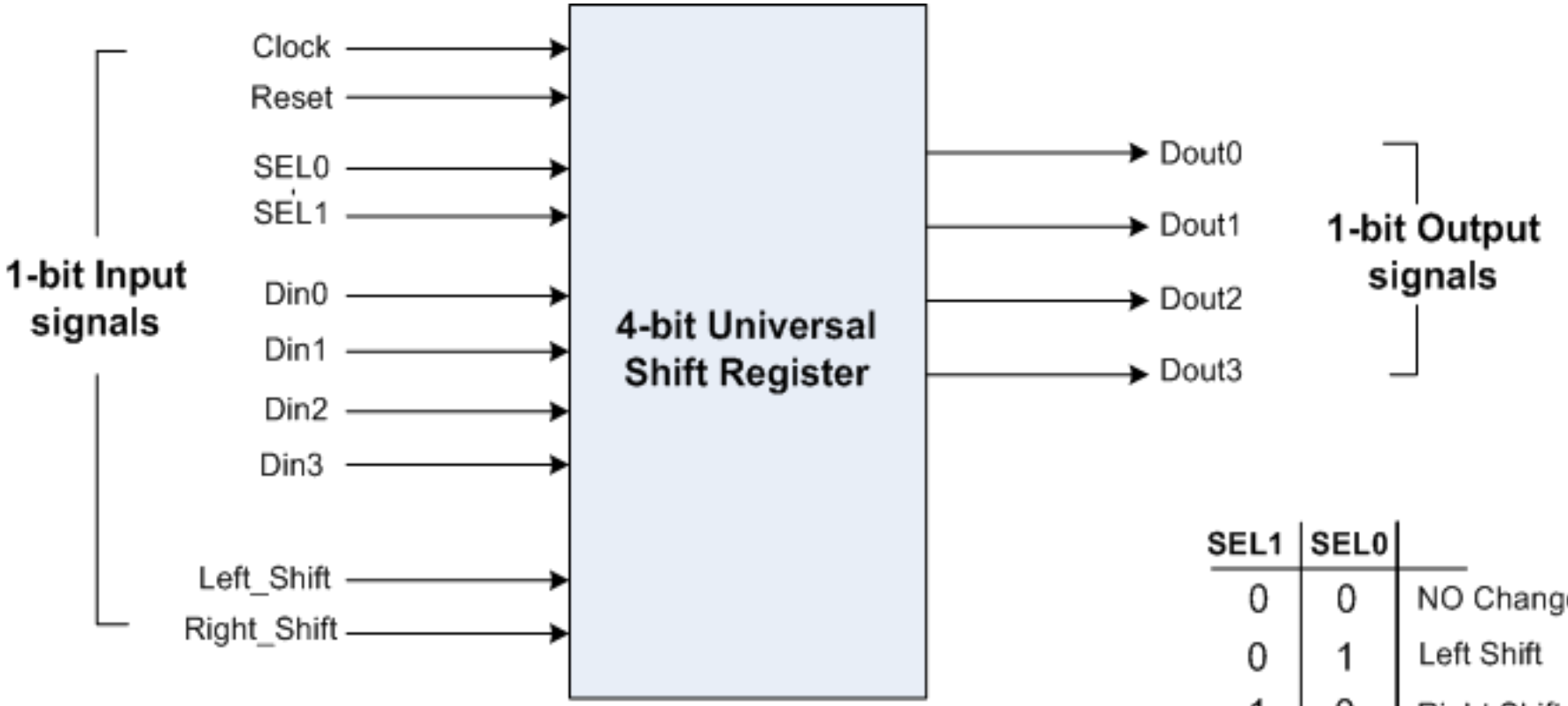
2. Develop RTL (Register Transfer Level) design in VHDL and create a test bench;

3. Check the functional simulation using the test bench;

4.
   I. Synthesize to create gate netlist;
   II. Implement the gate netlist for a specific FPGA fabric;
   III. Generate programming file.

5. Download this file into the FPGA and watch it work!

Step-1: Draw the circuit diagram



SEL1	SEL0	
0	0	NO Change
0	1	Left Shift
1	0	Right Shift
1	1	Load

Step-2: Design and testing in VHDL

RTL Design

```
1
2  library IEEE;
3  use IEEE.std_logic_1164.all;
4  use IEEE.numeric_std.all;
5
6  --// This is the TOP VHDL file which implements a 4-bit Universal Shift Register
7  entity UniShiftReg is
8    port (
9      Reset  : in std_logic;   -- One bit input port
10     Clock  : in std_logic;   -- One bit input port
11     SIL    : in std_logic;   -- One bit input port
12     SIR    : in std_logic;   -- One bit input port
13     Sel    : in std_logic_vector(1 downto 0);      -- Two bit input port
14     Din    : in std_logic_vector(3 downto 0);      -- 4-bits data input port
15     Dout   : out std_logic_vector(3 downto 0));    -- 4-bits data output port
16 end UniShiftReg;
17 --// End of entity
18
19 --// This is an architecture of 4-bits universal shift register
20 architecture RTL of UniShiftReg is
21   signal unishftreg4 : unsigned(3 downto 0); -- Inte
22 begin
23   -- Main process block which implment the universal
24   process(Reset, Clock)
25   begin
26     if Reset = '0' then
27       unishftreg4 <= (others => '0');
28     elsif rising_edge(Clock) then
29       if Sel = "11" then
30         unishftreg4 <= unsigned(Din);
31       elsif Sel = "01" then
```

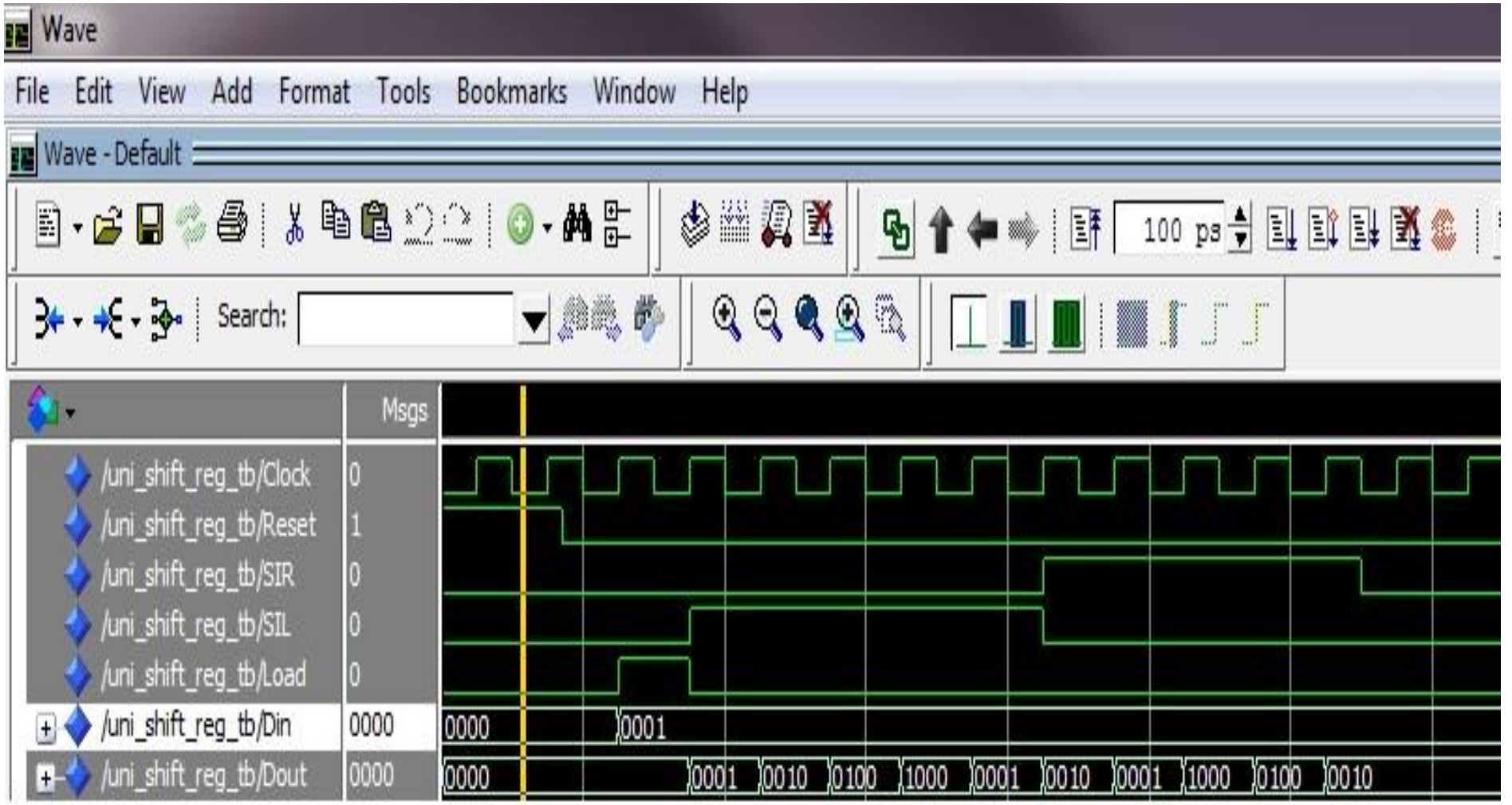
Design Summary (running) | UniShiftReg.vhd

Test bench

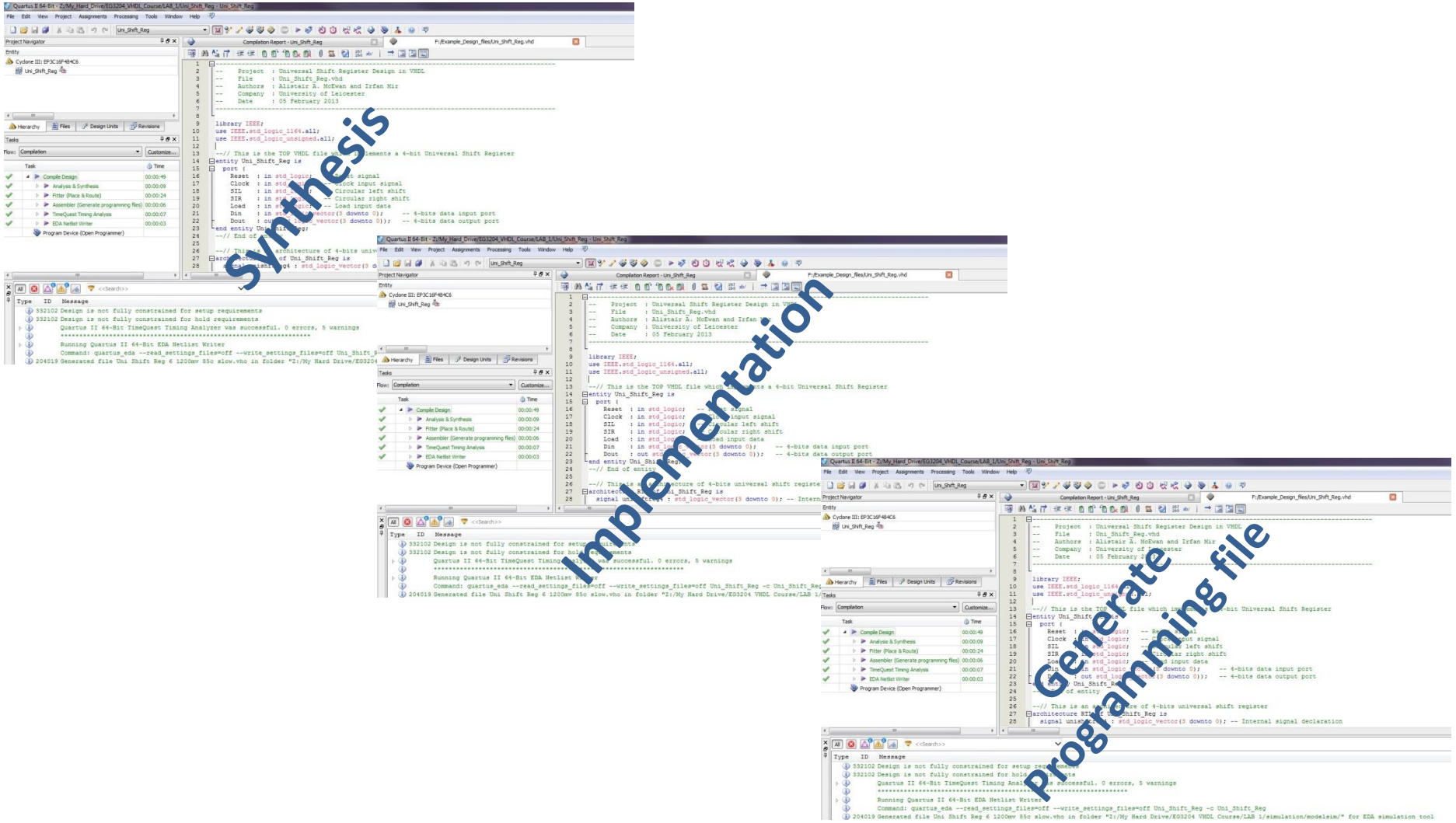
```
1  library ieee;
2  USE ieee.std_logic_1164.ALL;
3  USE ieee.std_logic_unsigned.all;
4  USE ieee.numeric_std.ALL;
5
6  --// This is the TOP VHDL Test Bench file for testing a 4-bit Universal Shift Register
7  entity UniShiftReg_TB IS
8  end UniShiftReg_TB;
9
10 -- Architecture block of an entity where the main testing part is designed
11 architecture TB of UniShiftReg_TB is
12
13    -- Component Declaration of 4-bit Universal Shift Register --
14    Component UniShiftReg
15    port(
16       Reset  : in std_logic;
17       Clock  : in std_logic;
18       SIL    : in std_logic;
19       SIR    : in std_logic;
20       Sel    : in std_logic_vector(1 downto 0);
21       Din    : in std_logic_vector(3 downto 0);
22       Dout   : out std_logic_vector(3 downto 0)
23         );
24    end Component;
25
26    -- Initialize input signals --
27    signal Clock  :  std_logic := '0';
28    signal Reset  :  std_logic := '0';
29    signal SIR    :  std_logic := '0';
30    signal SIL    :  std_logic := '0';
31    signal Sel    :  std_logic_vector(1 downto 0) := (others=>'0');
```

Design Summary (Programming File Generated) | UniShiftReg.vhd | UniShiftReg_TBench.vhd*

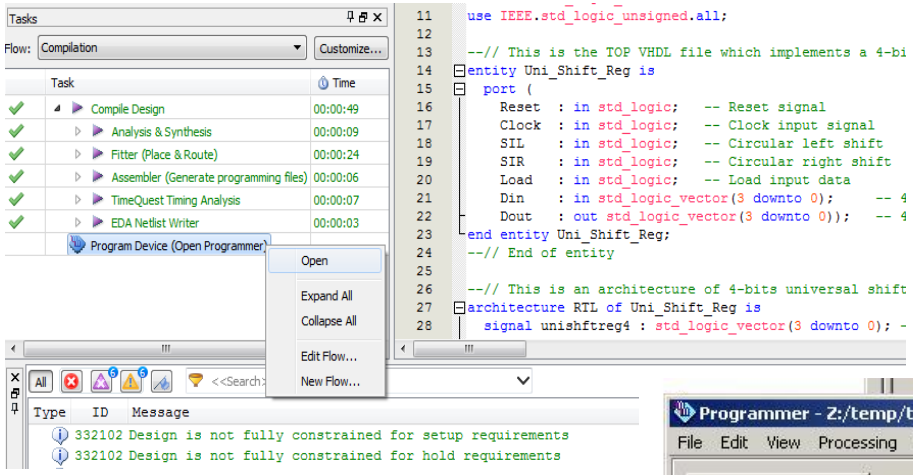
Step-3: Functional Simulation



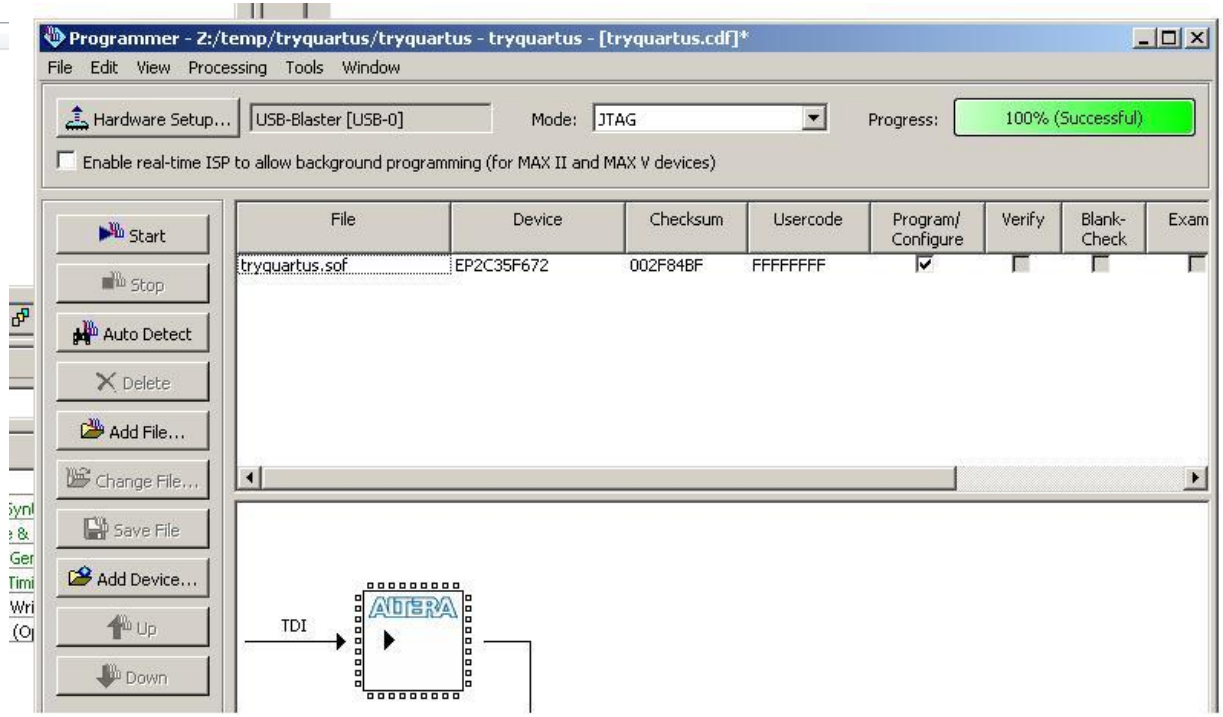
Step-4: VHDL Design Mapping on FPGA



Step-5: Configure Target Device



- Run the Program Device (Open Programmer)



- Add bit file (***.sof**) from the **output_files** directory and then click on **start** to program the FPGA.

Program

Lab work: build and configure a design

- In next week's lab class, you will load, and compile a given test application using Quartus-II, and load the application onto the test board.
- The aim is to gain familiarity with using the toolkit, and loading configurations onto the FPGA board.
- You should normally attempt the lab work individually!
- Attendance at labs is compulsory!
- Attendance at lectures is compulsory!
- If you fail to attend, you will struggle to pass the course!

Housekeeping

- You must attend the same lab session every week).

- Boards are normally made available for each person depending on numbers.

- Lecture notes and lab exercise sheets shall be posted on blackboard weekly. Look for the module space (EG3204) on blackboard!

Assessment

- There will be two assessed lab exercises later in the module.