Digital System Design ELEC373/473

Hardware Description Languages (HDLs)

and coding styles

Overview of HDLs (Hardware Description Languages)

- Designers use HDLs
  1) To model circuits
  2) To compile the model to synthesize it into physical circuits.
  3) To simulate the model to verify the functionality and timing of the circuit. Designers need to write a testbench containing descriptions of the stimulus waveforms that are applied to the circuit.
- Today HDLs are widely used to design circuits of enormous size and complexity. HDLs use design hierarchy to support the top-down methodology.

Overview of HDLs

- Provides many descriptive styles
  - Structural
  - Register Transfer Level (RTL)
  - Behavioral
- HDL based designs are highly portable and independent of technology.
- HDLs are not like procedural software and cannot run on processors.
- HDLs are a convenient medium for integrating intellectual property (IP) from a variety of sources with a proprietary design.

Overview of HDLs

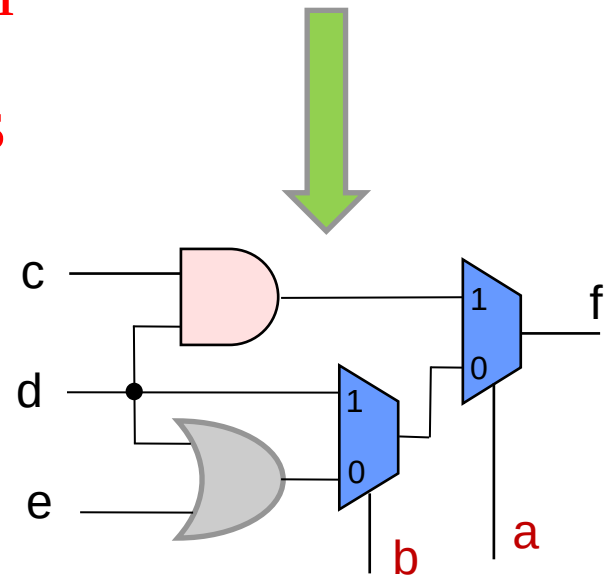
- HDLs reduces the time for the design cycle.
- Synthesis tools (e.g. Quartus) take a HDL model and generate the technology-specific netlist.
- Two main HDLs are used by industry:
  - Verilog (Verify Logic)
    - C-based
    - Driven by industry/academia
  - VHDL (Very High Speed Integrated Circuit HDL)
    - Ada based. Originally developed by an order of the USA DoD for documentation of digital systems.
    - Driven by defence/industry/academia

HDL vs. Software

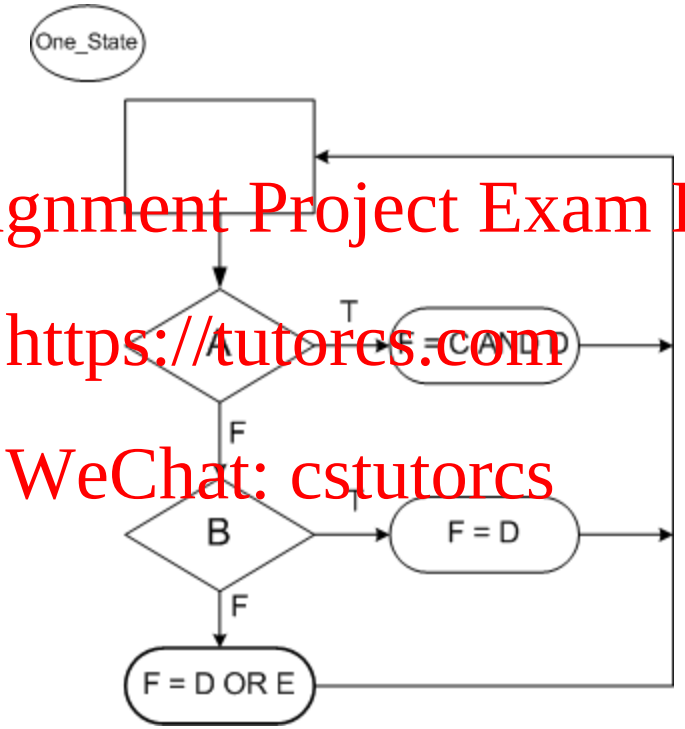
- In software programmes just one of the expressions is calculated depending on the values of a and b.

```
if(a) then f = c AND d;
  elsif(b) then f = d;
  else f = d OR e;
end if;
```


- In HDL all hardware for computing the logic are created during synthesis.
- Multiplexers are used to select the correct output.
- Learn to understand how descriptions are translated to hardware. “think hardware”



ASM for previous example



- Only combinational logic so just has a single state

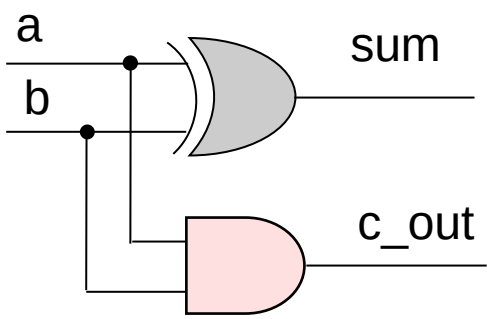
Verliog Module

- In Verilog, a circuit is a **module**.
- Three types of ports: **input**, **output**, **inout** (bidirectional)
- A signal is attached to every port.
- Specify multiple bits using range, e.g. a[7:0] or b[3:7]
- Single bit if range is not specified.

module name *ports names*

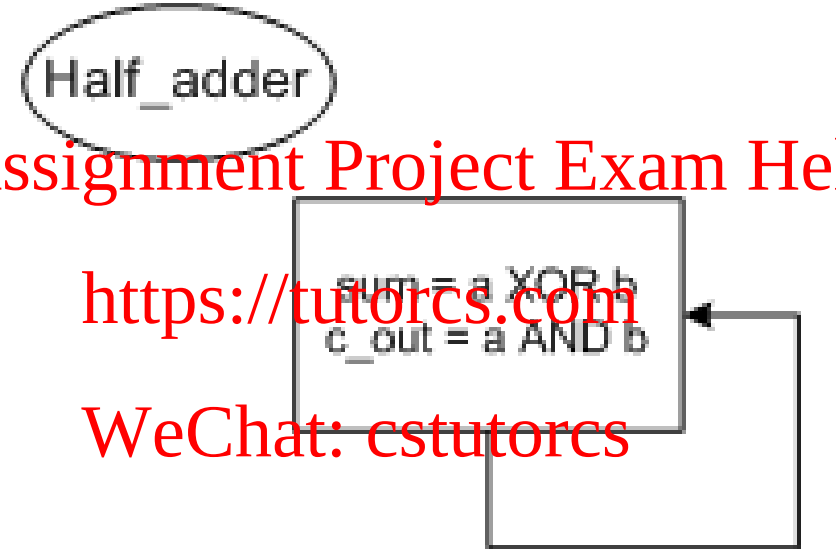
```
module Add_half ( sum, c_out, a ,b);
    input       a, b;
    output      c_out, sum;
    xor         (sum, a, b);
    and         (c_out, a, b);
endmodule
```

Ports types



Half Adder

ASM for half_adder



Alternative module declaration (introduced in the Verilog 2001 standard)

```
module Add_half ( output sum, output c_out, input a, input b);
      xor              (sum, a, b);
      and              (c_out, a, b);
endmodule
```

Module Styles

- Modules can be specified in different ways
  1) **Structural**: connect primitives and modules
  2) **RTL** (Register Transfer Level) :
     - use continuous assignments
  3) **Behavioral**: use **initial** and **always** blocks
     - Note that “initial” is primarily for simulation rather than for synthesis.
- A single module can use more than one method.