RTR532 2018 Spring Lecture 03

- **Design verification**
- **Finite State Machines**

Recap of previous lecture

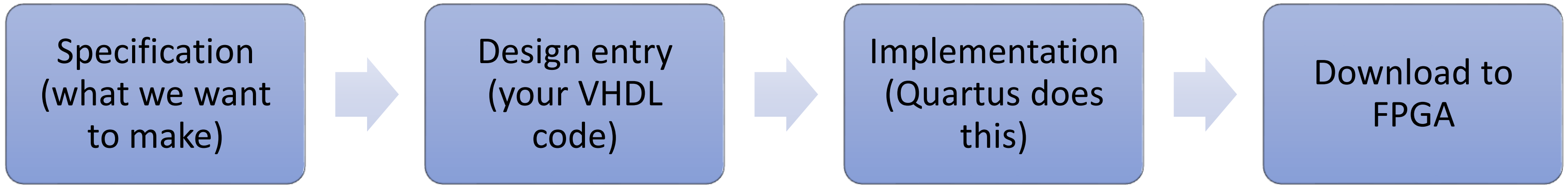
- Conditional assignments (2)
- Sequential assignments (2)

Final Project prospect

- Show your VHDL and FPGA skills
- Project ideas ?

Design verification

Verification definition


Specification (what we want to make)
Design entry (your VHDL code)
Implementation (Quartus does this)
Download to FPGA

But how can we know that the code works correctly ?

Verification – making sure your design works according to specification

What and how can we verify ?

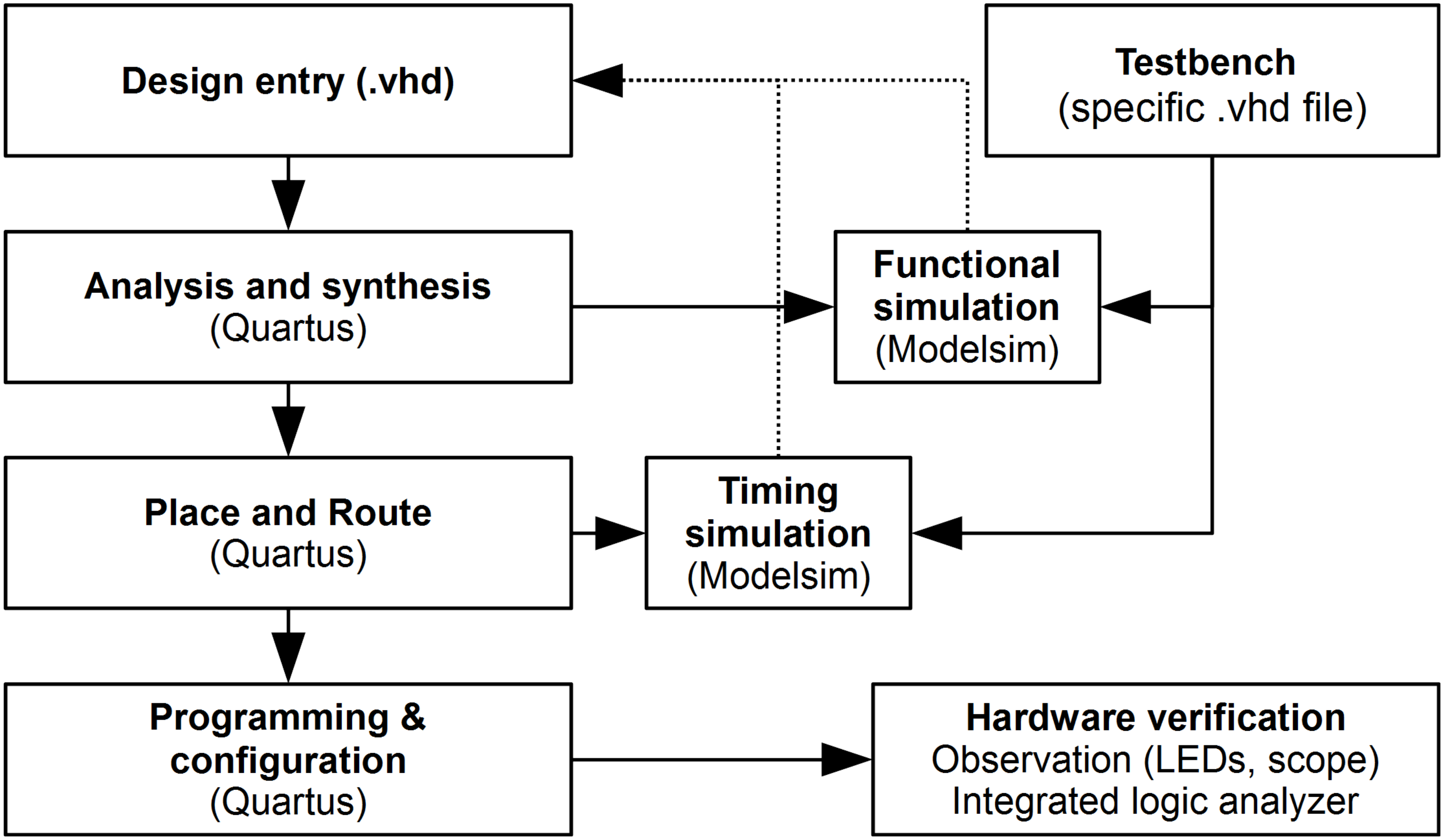
What

- **Functional verification** *does it function according to specification?*
- **Timing verification** *does the design work at the required speed?*

How

- **Simulations**
  - Functional
  - timing
- **Hardware evaluation**
  - In-chip logic analyzer
  - Observing LED’s/LCD/data interface

Design flow with verification



Simulations

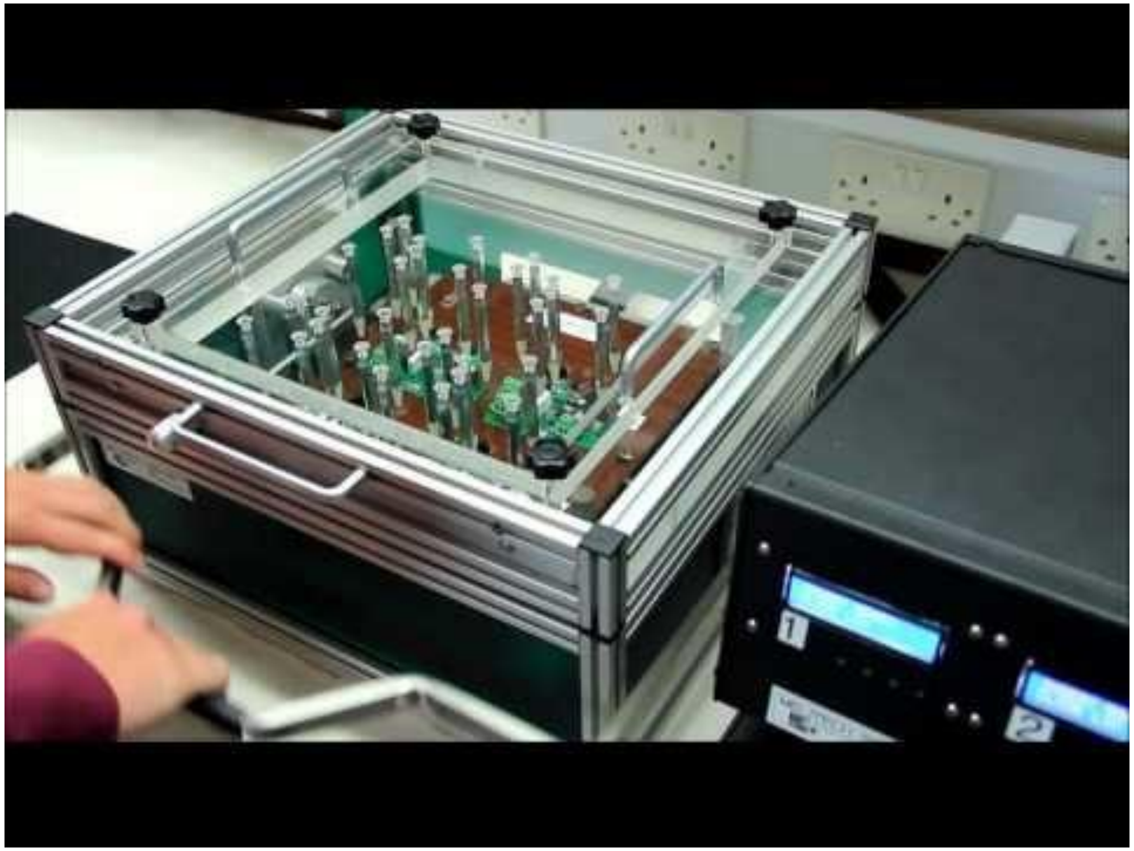
- Two simulation stages
  - **Functional simulation** - functionality
  - **Timing simulation** – functionality + delays

- You should ALWAYS do at least **functional simulation** before you download the FPGA code

Hardware evaluation

- Observing LEDs/LCD/data interfaces
  - Download fpga code and see what happens
  - Does not say much (optimized code?)
- In-chip logic analyzer
  - A specific IP module (VHDL block) – can be generated in Quartus
  - Acts as Logic Analyzer for required signals
  - See how signals change inside FPGA in real time

Testbench – test of PCB's



Testbench for VHDL

- We want to place our design in a testbed and
  - Apply stimulus
  - Check how outputs change
- Solution – Testbench vhdl file !
- In testbench you can:
  - Generate stimulus
  - Use non-synthesizeable signal type TIME
  - Use non-synthesizeable construct as WAIT, WAIT FOR, AFTER

Whiteboard example

- Draw entity to test (unit under test - UUT)
- Draw entity for a testbench to test UUT

Testbench (stimulus process, clock)

Stimulus process (wait only in Process):

```
stimulus_proc: process
begin
  wait for 10 ns;
  s <= "00";
  x <= "0000";
  wait for 10 ns;
  s <= "01";
  x <= "0110";
wait;
end process;
```

- WAIT Reference - http://www.ics.uci.edu/~jmoorkan/vhdlref/waits.html

Clock generation in testbench:

```
--<TIME> = xx {ps ns us s}
clk <= not clk after <TIME>;

--full period 20ns -> 50MHz
clk <= not clk after 10 ns;

--full period 1us -> 1 Khz
clk <= not clk after 500 ns;
```

- Quartus project example – testbench template
- Quartus allows to generate template testbench for your .vhd module!

- Help in \rtr532-2018\resources\
  1) step by step how to **adding_testbench_to_quartus_project_2017-03-07.txt**
  2) «clean template»- **testbench_template.vhd**

Design verification more info

- http://users.wpi.edu/~rjduck/VHDL%20module8%20a.pdf
- https://www.xilinx.com/support/documentation/application_notes/xapp199.pdf

Finite State Machines (**FSM**)

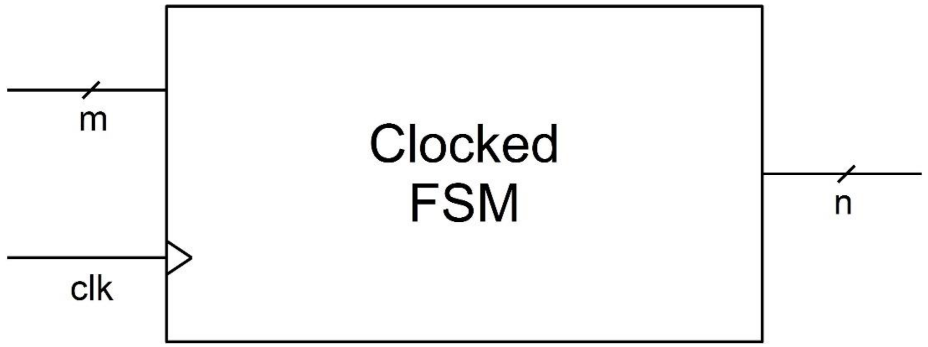
What is a finite state machine?

- Output depends on
  - Inputs
  - And previous state!

- FSM allows us to create **sequential** logic system

- Finite number of states

FSM definition



- k states $(S_1, S_2, \dots, S_k)$
- m inputs $(I_1, I_2, \dots, I_m)$
- n outputs $(O_1, O_2, \dots, O_n)$
- State transition: $S' = f(S, I)$
- Outputs based on state: $O = f(S)$
  (optionally also simultaneously on inputs)

Applications of FSM

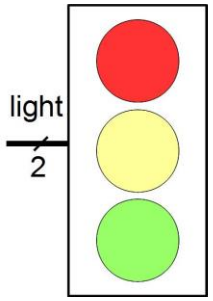
- Decision tree
- Sequence of events to happen

- Each state requires state transition (to what state we go from this state?!)
- Multiple FSM styles
  - **Single process**, two process, three process
  - Mealy machine, Moore machine.

Whiteboard example

- **FSM Decision tree**
- Binary stream signal input
- 1 bit signal output
- Output high level when 0110 binary sequence is detected in binary data stream
- Draw FSM state diagram

Whiteboard example



Traffic light controller FSM (event control)

Task: form a traffic light control signal:

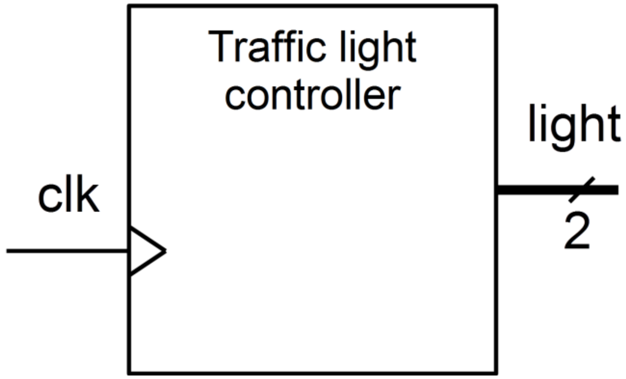
1. *Light* signal mapping: red = «00», yellow = «01», green = «10»
2. Required color timing:

   Red – 30sec

   Yellow – 5sec

   Green – 30sec

   Yellow – 5sec
3. Clock frequency = 1Hz



FSM with VHDL

- We start with Single-Process FSM
- Process + case
- Each case when => requires state transition info

- Output signals can be assigned either
  - in the state process
  - not in state process (z <= ‘1’ when state = «abc» else ‘0’);

- Quartus example
  - Binary stream capture
  - Traffic light

FSM more info

- https://www.youtube.com/watch?v=LM6Rm7mPcxg
- http://web.engr.oregonstate.edu/~traylor/ece474/vhdl_lectures/essential_vhdl_pdfs/essential_vhdl107-127.pdf
- http://quartushelp.altera.com/15.0/mergedProjects/hdl/vhdl/vhdl_pro_state_machines.htm

Lecture Summary

- Remember from this lecture:
    - Design verification
    - Finite State Machines – when to use

To do till next lecture:

- Problem set 03 (submit by 2018-03-22)
- Generate ideas for final project