Management and analysis of physics datasets, Part. 1

Fourth Laboratory

Antonio Bergnoli bergnoli@pd.infn.it

22/11/2021

Laboratory Introduction

Goals

- Recap of the previous lectures
- Synchronous logic design
- inference in VHDL and State machines

new VHDL constructs used in this lecture

1. **type** definition
2. other uses of **wait**
3. **if then else**

Selection of Internet resources

Web resources

1. https : //www.edaplayground.com/
2. https : //vhdlwhiz.com/
3. https : //surf-vhdl.com/

Free Books and Handbooks

4. VHDL Handbook **old but still good!!**
5. Free Range VHDL

Synchronous Logic design

Synchronous circuits

Neglecting temporary the *multi clock* systems, we assume the following statements as true:

1. **there is a single clock in the system**
2. **we consider only one *sensitive edge* (rising or falling)**
3. **all the activity of the signals in the circuit occurs during the rising edge of the clock**

In other words: only during a rising edge of the clock the *state* the system could change

The RTL design

RTL stands for **R**egister **T**rasfer **L**evel

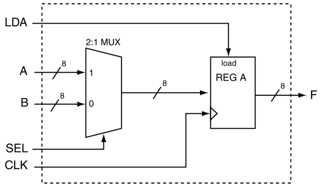


Figure 1: rtl example 1

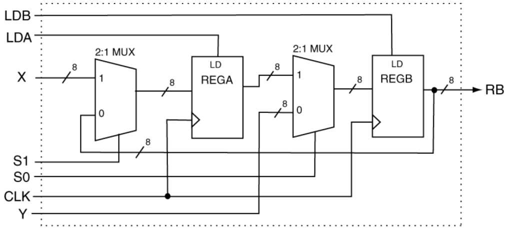
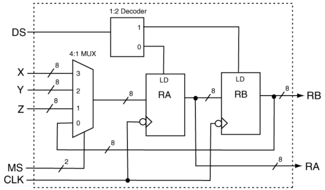


Figure 3: rtl example 3



Synchronizing the input

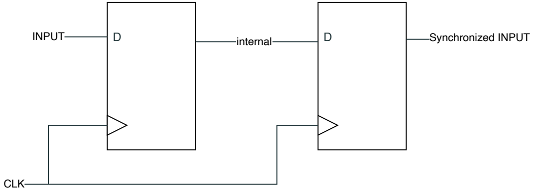


Figure 4: double flop

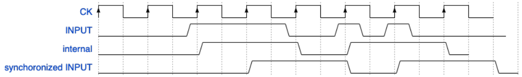


Figure 5: double flop wave

Why two flip flop and not one? (see next lectures, *metastability*)

Exercise

Model the input synchronizer and test it

1. write a VHDL source code for the synchronizer
2. write a testbench to validate the input synchronizer

Inference in VHDL

What Inference means

Inference informal definition

Inference is the process of transforming a high level description into a lower level representation (e.g: *rtl* level or *gate* level)

From “Free Range VHDL”

(...) Modeling digital circuits with VHDL is a form of modern digital design distinct from schematic-based approaches. The programmer writes a loose description of what the final logic circuit should do and a language compiler, in this case called a synthesizer, attempts to **infer** what the actual final physical logic circuit should be. Novice programmers are not always able to convince the synthesizer to implement something that seems very clear in their minds. (...)

State machines

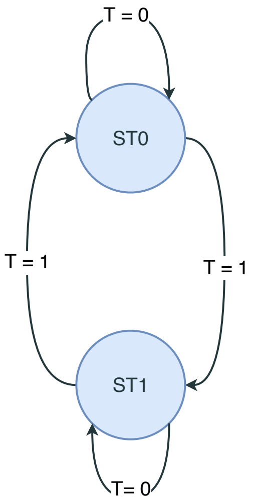


Figure 6: toggle state machine

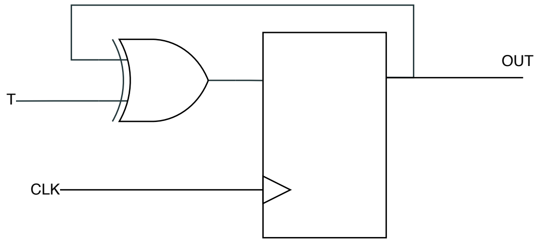


Figure 7: toggle flip flip

State machines source code (taken from “Free Range VHDL” book)

Toggle flip flop as state machine

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;            -- entity
entity my_fsm1 is
  port (TOG_EN, CLK, CLR : in  std_logic;
        Z1               : out std_logic);
end my_fsm1;
```

```vhdl
architecture fsm2 of my_fsm1 is
  type state_type is (ST0, ST1);
  signal state : state_type;
begin
  sync_proc : process(CLK)
  begin
    if (rising_edge(CLK)) then
      if (CLR = '1') then
        state <= ST0;
        Z1    <= '0';                   -- pre-assign
      else
        case state is
          when ST0 =>  -- items regarding state ST0 Z1 <= '0'; -- Moore output
            Z1                           <= '0';  -- pre-assign
            if (TOG_EN = '1') then state <= ST1;
            end if;
          when ST1 =>                   -- items regarding state ST1
            Z1                           <= '1';  -- Moore output
            if (TOG_EN = '1') then state <= ST0;
            end if;
          when others =>                -- the catch-all condition
            Z1    <= '0';               -- arbitrary; it should never
            state <= ST0;               -- make it to these two statements
        end case;
      end if;
    end if;
  end process sync_proc;
end fsm1;
```

Homework

Suggested exercises

- build another state machine and try to model it in VHDL;
- build a testbench for it.