LINKÖPING UNIVERSITY
Department of Electrical Engineering

TSIU03, SYSTEM DESIGN

LECTURE 4

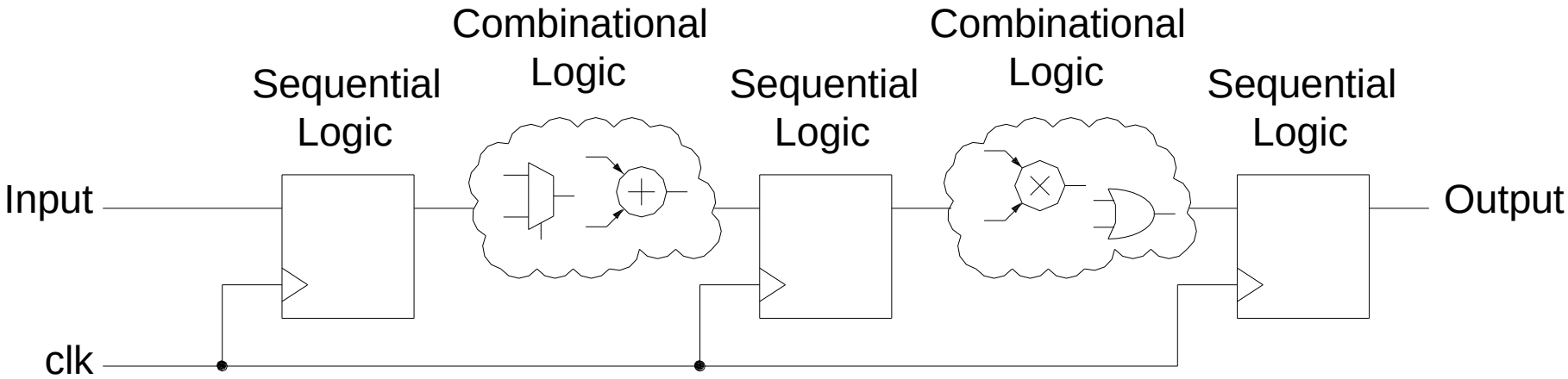
Mario Garrido Gálvez
mario.garrido.galvez@liu.se

TODAY

- Structure of a hardware system: combinational logic and sequential logic, clock,…
- Sequential logic: clock, clock cycle, clock frequency, reset, enable, initialization, register, shift register, counter, accumulator, timing diagram.
- VHDL description for sequential logic.
- Simulations and test bench.

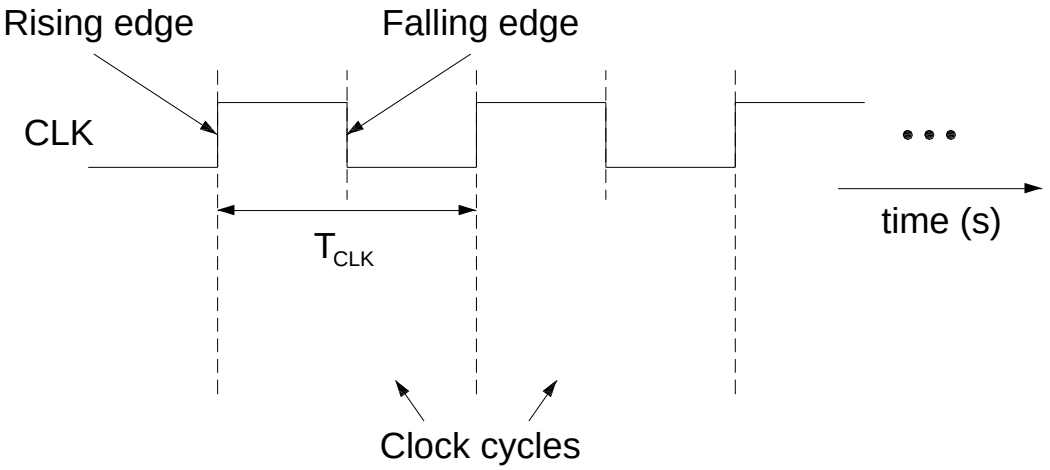
STRUCTURE OF A HW SYSTEM

- **Combinational logic**: Its output only depends on the current input.
- **Sequential logic:** Its output depends on present and previous values. They are memory elements.
- At each clock event, the output of the sequential logic is updated. This makes the combinational logic have new inputs, and the combinational logic makes the calculations for the new inputs.



CLOCK

- The **clock** of a digital system is a periodical signal that changes alternatively between one and zero. The time in a digital system in measured in **clock cycles.**
- The **clock period (T_{CLK})** is the time in seconds between two consecutive clock cycles. The **clock frequency (f_{CLK})** is the inverse of the clock period and indicates the number of clock cycles per second, measured in Hz. Higher clock frequency means faster processing.
- All the digital system must be synchronized with the clock and updates every clock cycle. The sequential logic is activated only in the **rising edge** (alternatively the **falling edge**) of the clock.



$$f_{CLK} = \frac{1}{T_{CLK}}$$

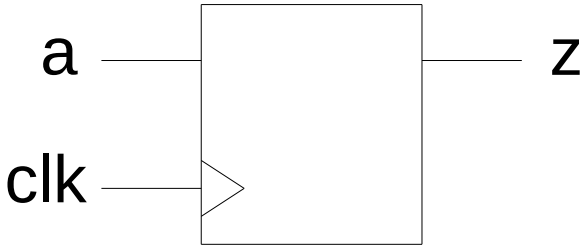
(DIGITAL) MUSIC

- The **clock** is like a conductor in an ochestra. There is only one and assures that everything is synchronized. With two conductors the orchestra cannot be synchronized. **Only one clock signal in the system!!**
- The conductor defines the tempo of the music. In a digital system, this tempo is given by the clock frequency (or the clock period).

- And the instruments? -> Sequential logic. The clock is only connected to the sequential logic and only connected as a clock, not as a normal input.

REGISTER

- n: discrete time unit.
- The register delays the input one clock cycle.
- Can be used to store data.
- If the input and output have one bit it is called D flip-flop.

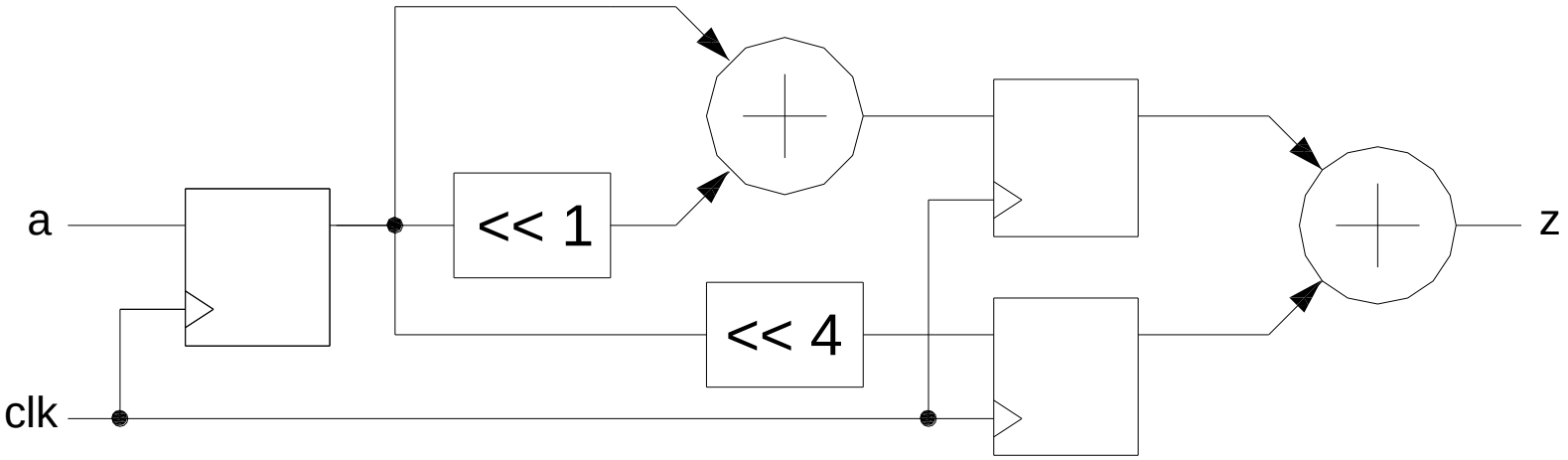


$$z[n] = a[n-1]$$

```
architecture rtl of reg is
…
begin
…
process (clk)
begin
  if rising_edge (clk) then
    z <= a;
  end if;
end process;
…
```

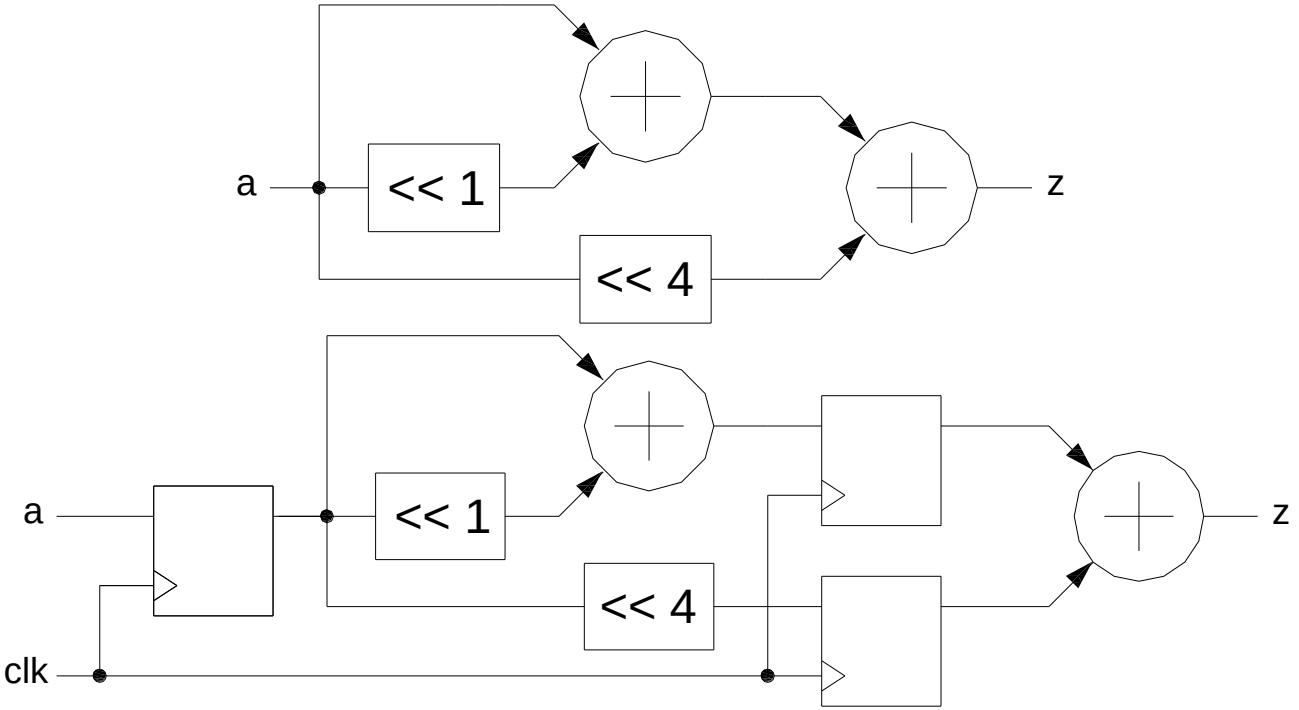
- What is the difference with respect to a wire in the VHDL code?
- Ref: [A4.1.1]

WHAT DOES THIS CIRCUIT DO?



WHAT IS THE DIFFERENCE?

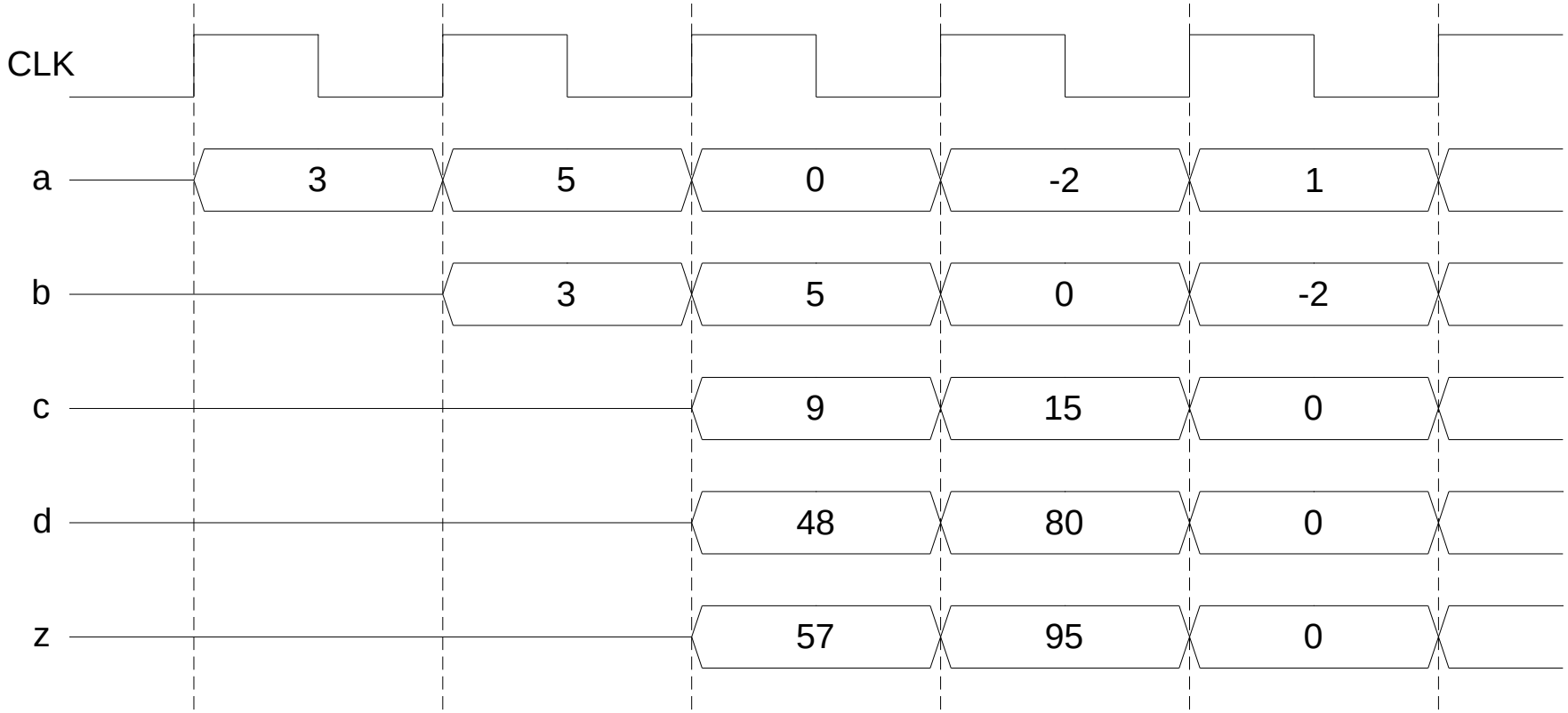
- Which is the difference between these two circuits?

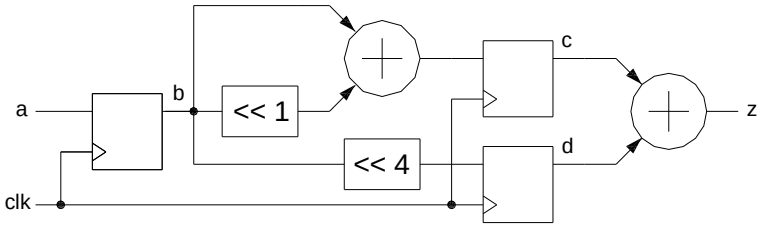


- Can we use any of these circuits to multiply by 19?
- Which mathematical function is calculated by each of them?

TIMING DIAGRAM

- A timing diagram is used to show the evolution of the signals in time.





REGISTER WITH RESET & ENABLE

```
process (clk, reset)
begin
  if reset = '1' then
    z <= (others => '0');
  elsif rising_edge (clk) then
    if enable = '1' then
      z <= a;
    end if;
  end if;
end process;
```

- Reset: sets the signals to their initial value.
- Enable: enables the computations of the circuit.

PROCESS

```
process (<sensitivity_list>) -- triggers for the process
<variable_declaration> -- only if the process has variables
begin
  if reset = '1' then     -- asynchronous reset (also:  ='0')
    <initialization>      -- initialize signals and variables
  elsif rising_edge (clk) then
    if enable = '1' then -- only if the enable is included
      <statements>        -- behavior of the circuit
    end if;
  end if;
end process;
```

WHAT DOES THIS CIRCUIT DO?

```
process (clk, reset)
begin
  if reset = '1' then
    z <= (others => '0');
  elsif rising_edge (clk) then
    if s = '1' then
       z <= a;
    else
       z <= b;
    end if;
  end if;
end process;
```

- Can you draw the circuit?
- Can you describe the circuit in a different way?

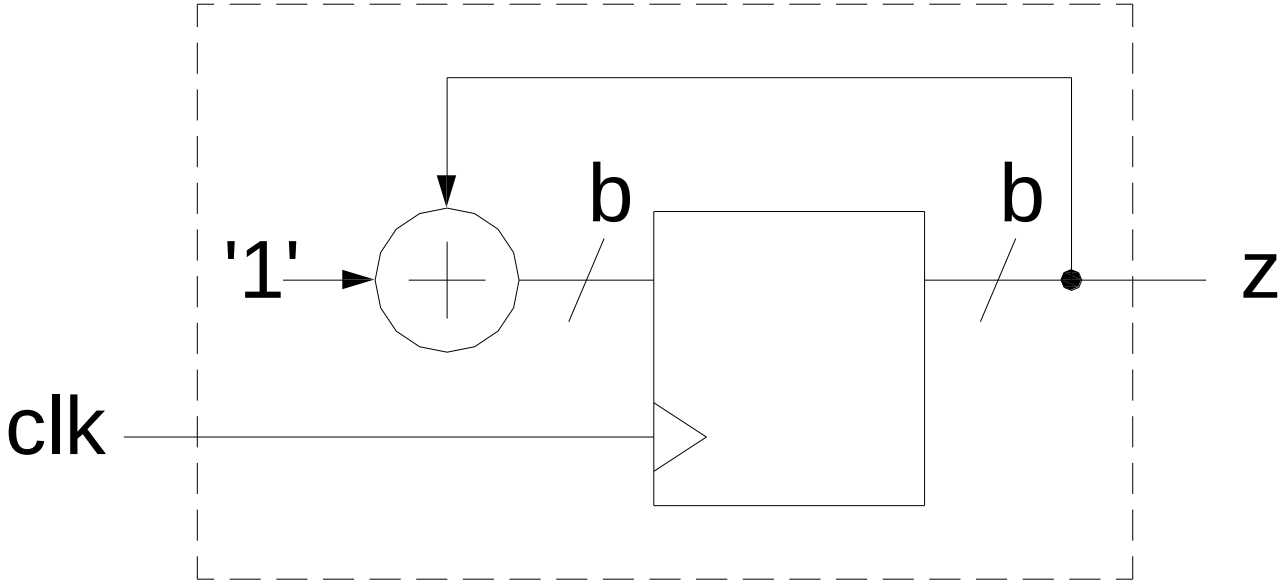
OTHER OPTION

- In this case, in the description we separate the combinational and the sequential parts of the circuit:

```
process (clk, reset)
begin
  if reset = '1' then
    z <= (others => '0');
  elsif rising_edge (clk) then
      z <= p;
  end if;
end process;


p <= a when s = '1' else b;
```

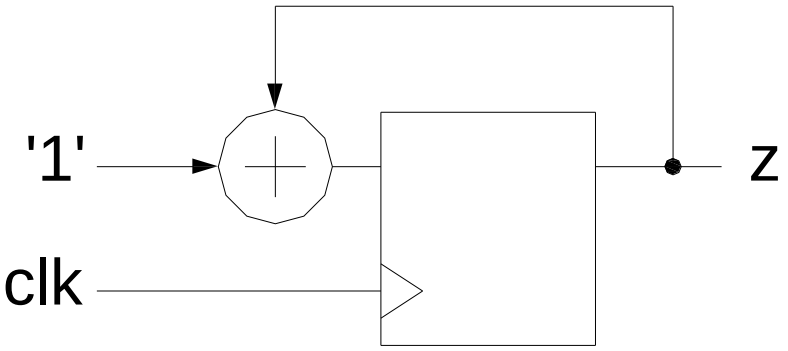
WHAT DOES THIS CIRCUIT DO?



- What happens with the overflow?

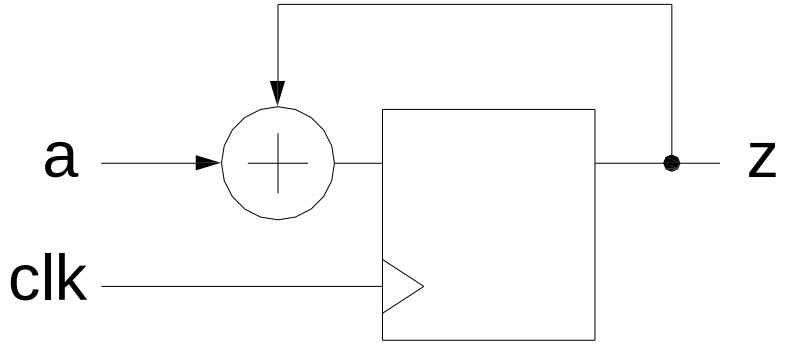
COUNTER & ACCUMULATOR

COUNTER



$$z[n] = n$$

ACCUMULATOR



$$z[n] = \sum^{n} a[n]$$

- Be very careful with the overflow.
- If not controlled, the counter is periodical.
- Which is the difference of VHDL code of these circuits with respect to the VHDL code of a register?
- Ref: [A4.2].

HOW CAN WE INITIALIZE…?

- How can we initialize the signals y and z?

```
signal a, b, y, z: std_logic;
…
begin
…
process (clk)
begin
if rising_edge (clk) then
        z <= a AND b;
  end if;
end process;
y <= a OR b;
```

SIGNAL INITIALIZATION

- We can only initialize the values of signals that store information.
- Some registers need to be initialized. Otherwise, the initial value may be generated arbitrarily, which may lead to unexpected behaviors.
- Other registers do not need to be initialized. They will be updated once the circuit starts to compute data.
- Initialization is done by using the reset signal.
- Signals that do not store information can not be initialized!

SHIFT REGISTER

- Register in which the bits shift position.
- Can be used as a delay of L clock cycles.

```
process (clk)
begin
if rising_edge (clk) then
        z <= sr (L-2);
        sr (L-2 downto 1) <= sr (L-3 downto 0);
        sr (0) <= a;
  end if;
end process;
```

- How can we shift the bits in the other direction?
- Ref: [A4.1.1]

SIMULATIONS

- We use Model Sim.
- Commands / tricks that you should know so far from the labs: zoom in the simulation, use cursors and measure the time, add dividers, change the order and the color of the signals, combine signals, change signal representation (radix).
- The simulation shows exactly what you will see when you load your system to the board. If the circuit in the simulation does not calculate the expected function, you know that it will NOT work on the board.
- We configure the simulation using:
  - VHDL test benches.
  - Scripts.
  - Runing commands manually in ModelSim.

TEST BENCH

- A test bench is a VHDL file that generates stimuli for a circuit and receives the outputs of the circuit with the purpose of testing its behavior.
- It is created as a top of the file that we want to test.
- As the test bench is only used for simulation purposes and is never configured on the FPGA, it can contain **non-synthesizable VHDL**.
- It allows for generating any type of test vectors, e.g., create an input test signal that is a sinusoid.
- It can also import test data from a file and write the outputs of the system to another file. In this way we can generate test vectors with another program such as Matlab, run the simulations in ModelSim and, then, anaylze the results with Matlab again. For input/output values from/to a file, we need to use the package **textio**:

```
library std;
use std.textio.all;
```

TEST BENCH EXAMPLE

```
...
  constant clk_period : time := 10 ns; -- Clock period.
begin
  rstn <= '1', '0' after 10 ns, '1' after 25 ns; -- Reset.
  -- Generation of the clock.
  clk_process :process
  begin
    clk   <= '0';
    wait for clk_period/2;
    clk   <= '1';
    wait for clk_period/2;
  end process;
  -- Instantiation of the component to simulate.
  ctrBlock: entity work.controlBlock
    port map( rstn => rstn, clk => clk, counter => counter);
...
```

SCRIPTS FOR MODELSIM

- All these commands can be done manually in ModelSim. The script just runs them automatically:

```
vlib work                  Creates a design library.
vcom *.vhd                 Compile all the .vhd files.
vsim work.Sound            Simulate the file Sound.
add wave sim:/Sound/*      Adds signals to the wave.
        Initial values for the inputs of the simulated file:
force -freeze sim:/sound/clk 1 0, 0 {10 ns} -r 20ns
force -freeze sim:/sound/rstn 0 0, 1 {30 ns}
force -freeze sim:/sound/adcdat 0 0
run 1ms             Run the simulation for a certain time.
wave zoom full      Adjust the zoom of the wave.
```

- Note that we always choose which commands we run manually and which ones with a script.

VHDL TEST BENCH + SCRIPTS

- The best way to do prepare a simulation is to combine the advantages of a test bench in VHDL and the use of scripts.
- VHDL test bench:

  - Better for generation of input signals (clock, inputs, etc.). For instance, we can generate a sinusoid in the test bench to test our circuit. Imagine how it would be to do it with a script…

- Script (.do file):

  -Good for specific ModelSim commands (vlib, vcom, run,…). Also possible to run them manually.

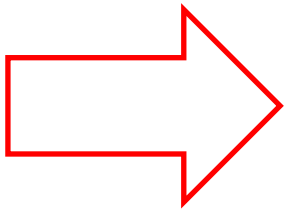
  - Very useful to save the wave format. File -> Save Format. This saves a .do file with the entire layout of the simulation, so that you do not have to create it again.

GRADE FOR THE ASSIGNMENTS

Assignment Points

(0 to 10)

Grade Conversion



Grade

(1 to 5)

Pass = 3

- Numerical grade because:

  - International students prefer a numerical grade to report to their university (a “pass” without numerical grade is usually recognized as the minimum grade to pass).

  - It is less strict to calculate the average than to have to pass all the individual assignments (you can pass even if you are not lucky with some assignments).

GRADE FOR THE ASSIGNMENTS

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity GradeConversion is
   port (AssignmentPoints: in unsigned (3 downto 0);
         Grade : out unsigned (2 downto 0);
end GradeConversion;
architecture arch of GradeConversion is
   signal x: unsigned (3 downto 0);
begin
   x <= AssignmentPoints + 1;
   Grade <= x(3 downto 1);
 end arch;
```

CHECKLIST FOR LECTURE 4

- Sequential logic: clock, clock cycle, clock frequency, clock period, reset, enable, initialization, register, shift register, counter, accumulator, timing diagram.
- VHDL: process, clock signal, reset signal, enable signal, initialization, sensitivity list, rising_edge, if statement.
- Simulations: VHDL test bench, script, simulation tricks.

AT HOME

- Review the checklist for lecture 4 and check that you understand all the concepts and you know how to use them.