LINKÖPING UNIVERSITY
Department of Electrical Engineering

TSIU03, SYSTEM DESIGN

LECTURE 7

Kent Palmkvist
Kent.Palmkvist@liu.se

Slides by: Mario Garrido Gálvez (mario.garrido.galvez@liu.se)

Linköping, 2021

TODAY

- Processes: order of execution inside a process, processes for combinational logic, case statement, for… loop, taking care of the sensitivity list, …
- Lab 4

WHAT DOES THIS CIRCUIT DO?

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

- As usual, to know what it does you only need to draw the circuit.

… AND THIS ONE?

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

… AND THIS ONE?

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


PROCESS EXECUTION

- A process is evaluated **only** when there is a change of one of the signals in the sensitivity list.
- Only in that moment, the entire process is evaluated.
- Output signals are assigned the last value that they are given during the process evaluation.
- If no value is assigned to an output of a process during an execution of the process, then the output keeps its previous value. Thus, it is important to define which is the value of the output for each possible combination of inputs. Otherwise we may have unexpected behaviors.

WHAT DOES THIS CIRCUIT DO?

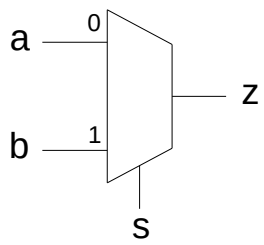
```
process (s, a, b)
begin
  if s = '0' then
    z <= a;
  else
    z <= b;
  end if;
end process;
```

- Is it a combinational circuit or a sequential one?

WHICH IS BETTER?

- Among these two VHDL descriptions of the same circuit, which one do you prefer?

```
process (s, a, b)
begin
  if s = '0' then
    z <= a;
  else
    z <= b;
  end if;
end process;
```




```
z <= a when s = '0' else b;
```

PROCESS FOR COMBINATIONAL LOGIC

- We can use processes for describing combinational logic, not only sequential one.
- In most cases, however, the description of a combinational circuit using processes is more complicated and more difficult to understand than the normal description. Thus, **try not to use processes for combinational logic unless it is strictly necessary.**
- We also have to be careful. What happens here?:

```
process (s)
begin
  if s = '0' then
    z <= a;
  else
    z <= b;
  end if;
end process;
```

(example of bad VHDL)

OR EVEN WORSE…

- What does this circuit do?

```
process (a)
begin
  z <= a;
  if s = '0' then
    z <= b;
  end if;
end process;
```

(example of bad VHDL: difficult to understand the circuit and bad behavior)

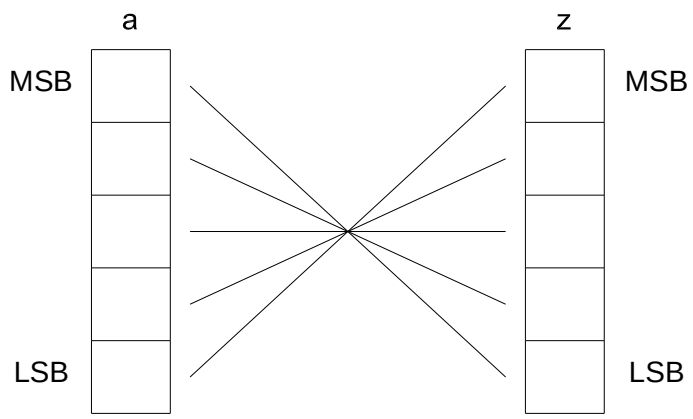
- …probably this is not the circuit that you want to implement.

WHICH CIRCUITS CAN WE USE TO...?

FUNCTION					
Calculate logic functions	X				
Calculate mathematical operations			X		
Delay a signal				X	
Store signals				X	
Compare signals	(X)		X		
Detect a transition in a signal	X			X	
Count	(X)		X	X	
Select among several signals		X			
Transform serial-parallel or parallel-serial		(X)		X	
Store constant values					X
Create a state machines	X			X	

HOW CAN WE DESCRIBE…

- Let’s see if we remeber how to assign bits…
- How can we describe the next circuit?



FOR … LOOP

- First option: normal assignments:

```
z(4) <= a(0);
z(3) <= a(1);
z(2) <= a(2);
z(1) <= a(3);
z(0) <= a(4);
```

- Second option: **for…loop** statement:

```
process (a)
begin
  for i in 0 to 4 loop
    z(i) <= a(4-i);
  end loop;
end process;
```

- Which description would you prefer if **a** and **z** have 32 bits?

CASE STATEMENT

- Only inside processes:

```
case <signal> is
  when <value1> =>
    <output> <= <input1/constant1>;
  when <value2> =>
    <output> <= <input2/constant2>;
  when others =>
    <output> <= <inputN/constantN>;
end case;
```

- It is important to **define all the values that the control signal can have**, or use “others” to cover those cases that are not assigned explicitly. Note that not defined cases keep the previous value, which may lead to unexpected behaviors of the circuit.

CASE STATEMENT: EXAMPLE 1

- Case statement for sequential logic:

```
process(clk) --IMP: clk: the only triggering signal
begin
  if rising_edge (clk)
    case s is
      when '0' => z <= a;
      when '1' => z <= b;
    end case;
  end if;
end process
```

- Note that in sequential logic, the sensitivity list of the process only needs to include the clk and reset (sequential logic does not need to change under other circumstances).

CASE STATEMENT: EXAMPLE 2

- Case statement for combinational logic:

```
process(s, a, b)
begin
  case s is
    when '0' => z <= a;
    when '1' => z <= b;
  end case;
end process
```

- Note that s, a and b need to be in the sensitivity list.
- Do you recognize this circuit?

LAB 4

- Control the audio codec used to input/output analog waveforms
  - Get audio signal into FPGA, modify, send back out
- Audio codec communicates with FPGA in serial form
  - Compare with lab 2, keyboard data recieved in serial form
- Input signal in stereo, can reuse parts of the design
  - Use hierarchy (block diagram) to use the same design in two situations

LAB 4

- Control the audio codec used to input/output analog waveforms
  - Get audio signal into FPGA, modify, send back out
- Input signal in stereo, can reuse parts of the design
  - Use hierarchy (block diagram) to use the same design in two situations
- Audio codec communicates with FPGA in serial form
  - Compare with lab 2, keyboard data recieved in serial form
  - Processing inside FPGA expect parallel form (vector)
    - Left data in, Left data out, Right data in, Right data out, lrsel
    - lrsel change indicate new input value

LAB 4

- Hierarchical design
  - Ctrl block to create control signals and clocks
    - Implemented as a counter + decoding/selection of bits
  - Channel_Mod block to convert between serial and parallell form
    - Shift registers only active when block is selected
    - Only 16 of 32 input bits should be saved in register

CHECKLIST FOR LECTURE 7

- VHDL processes:
  - order of execution inside a process.
  - taking care of the sensitivity list.
  - processes for combinational circuits.
  - for…loop.
  - case statement.
- Lab 4

AT HOME

- Review the checklist for lecture 7 and check that you understand all the concepts and you know how to use them.
- Start prepare for lab 4