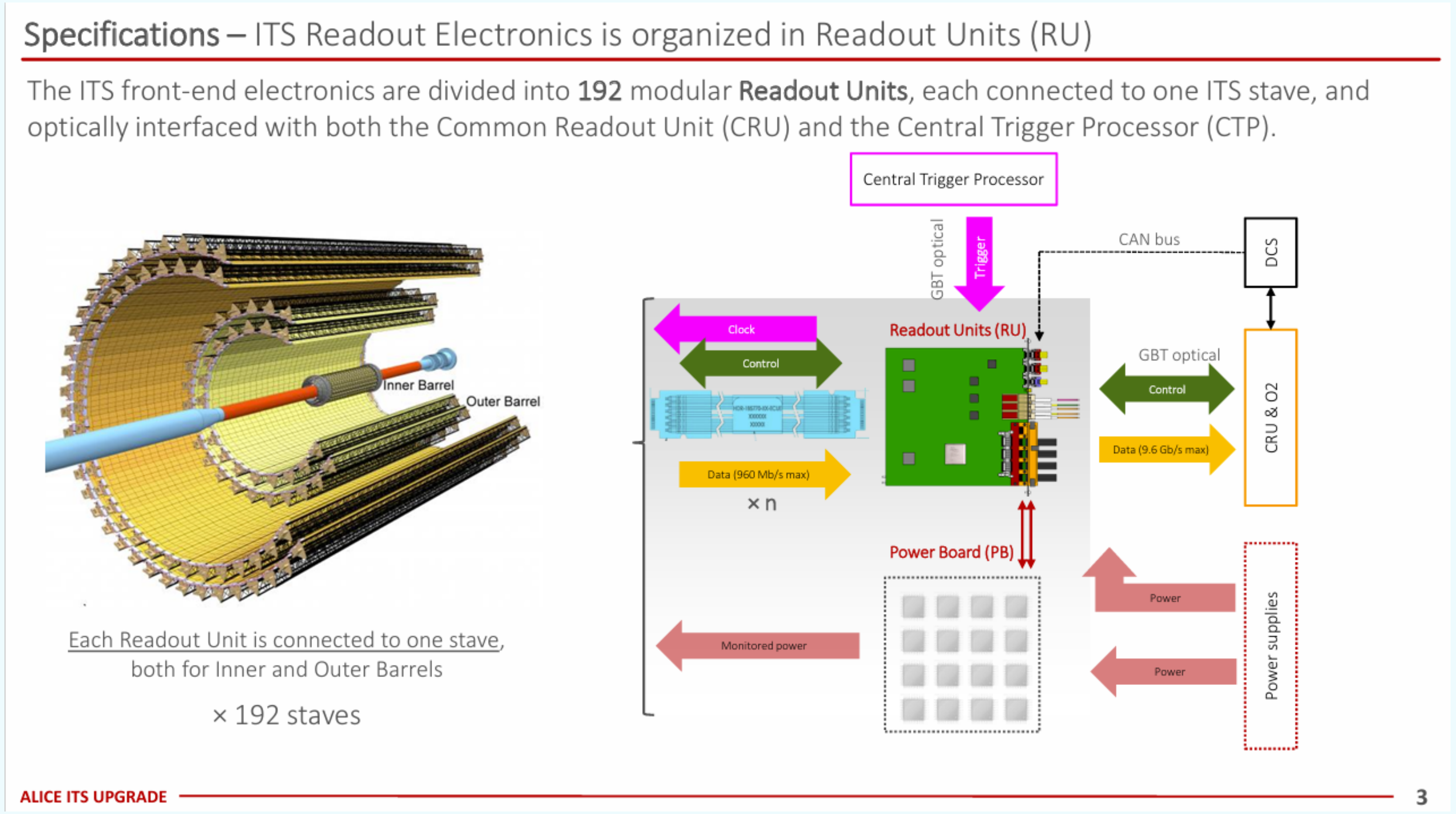
Digital design with VHDL

Lecture week on next generation particle detectors

10th – 14th June 2024, Bergen Norway

Johan Alme (johan.alme@uib.no)

ITS2 Readout



Specifications – ITS Readout Electronics is organized in Readout Units (RU)

The ITS front-end electronics are divided into **192** modular **Readout Units**, each connected to one ITS stave, and optically interfaced with both the Common Readout Unit (CRU) and the Central Trigger Processor (CTP).

Each Readout Unit is connected to one stave, both for Inner and Outer Barrels

× 192 staves

ALICE ITS UPGRADE 3

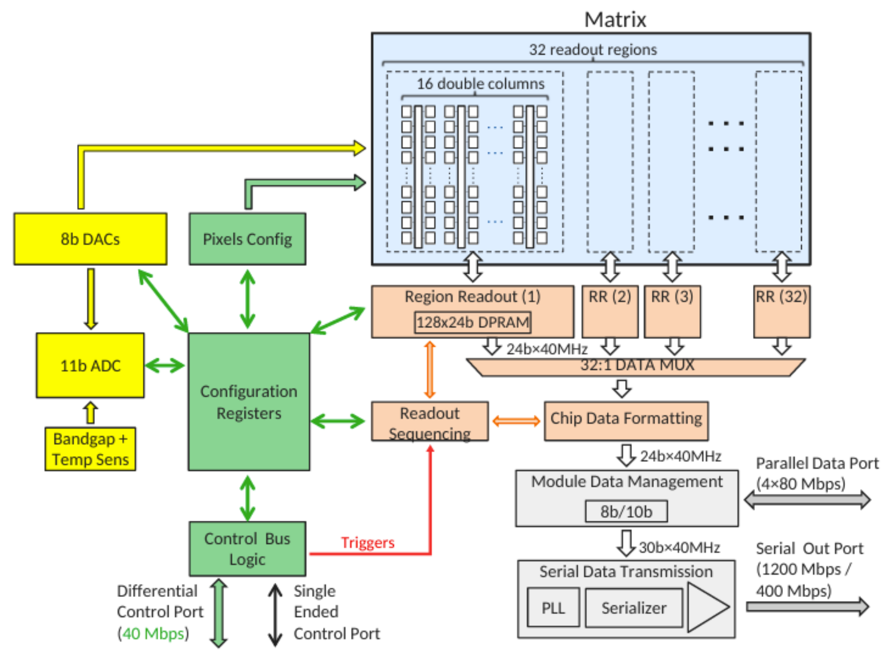
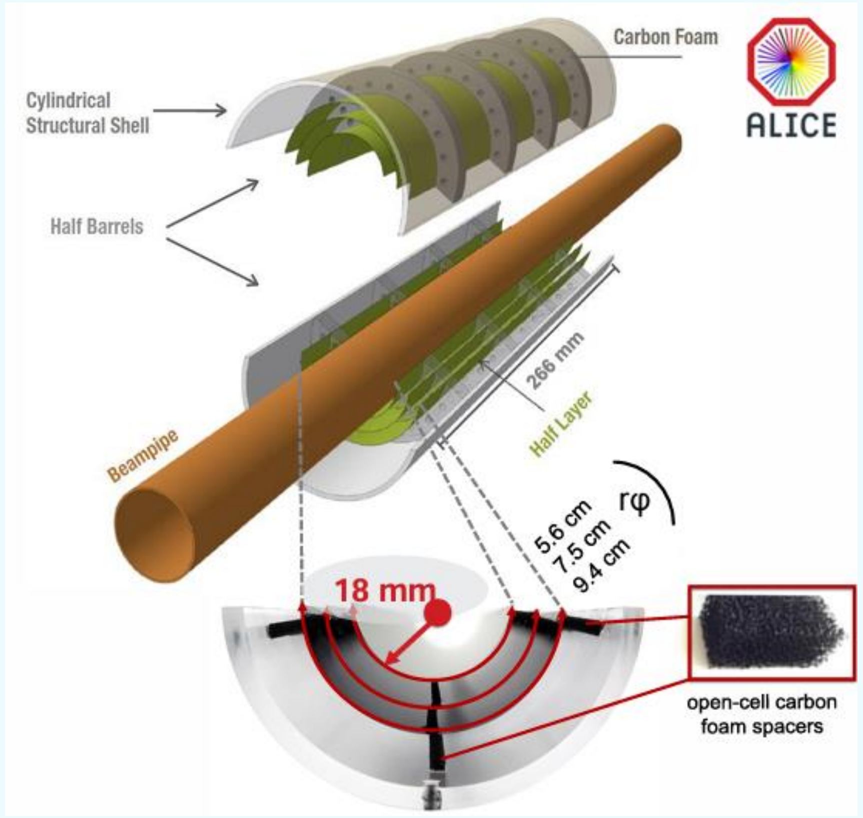


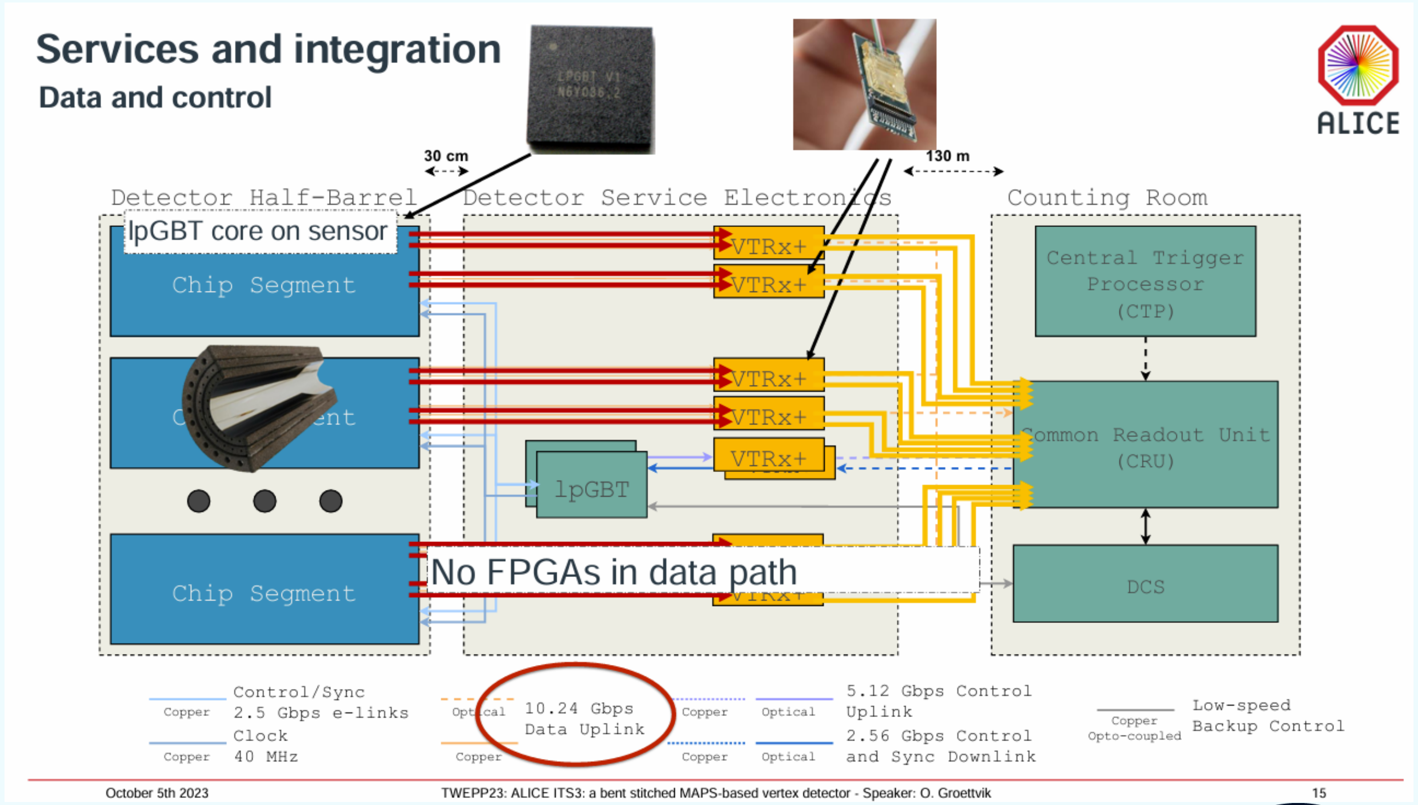
Figure 2.1: ALPIDE chip block diagram.

ALPIDEmanual_v0_3.pdf

WP10_PRR_RU_Overview_v1.pdf (cern.ch)

ITS3 Readout



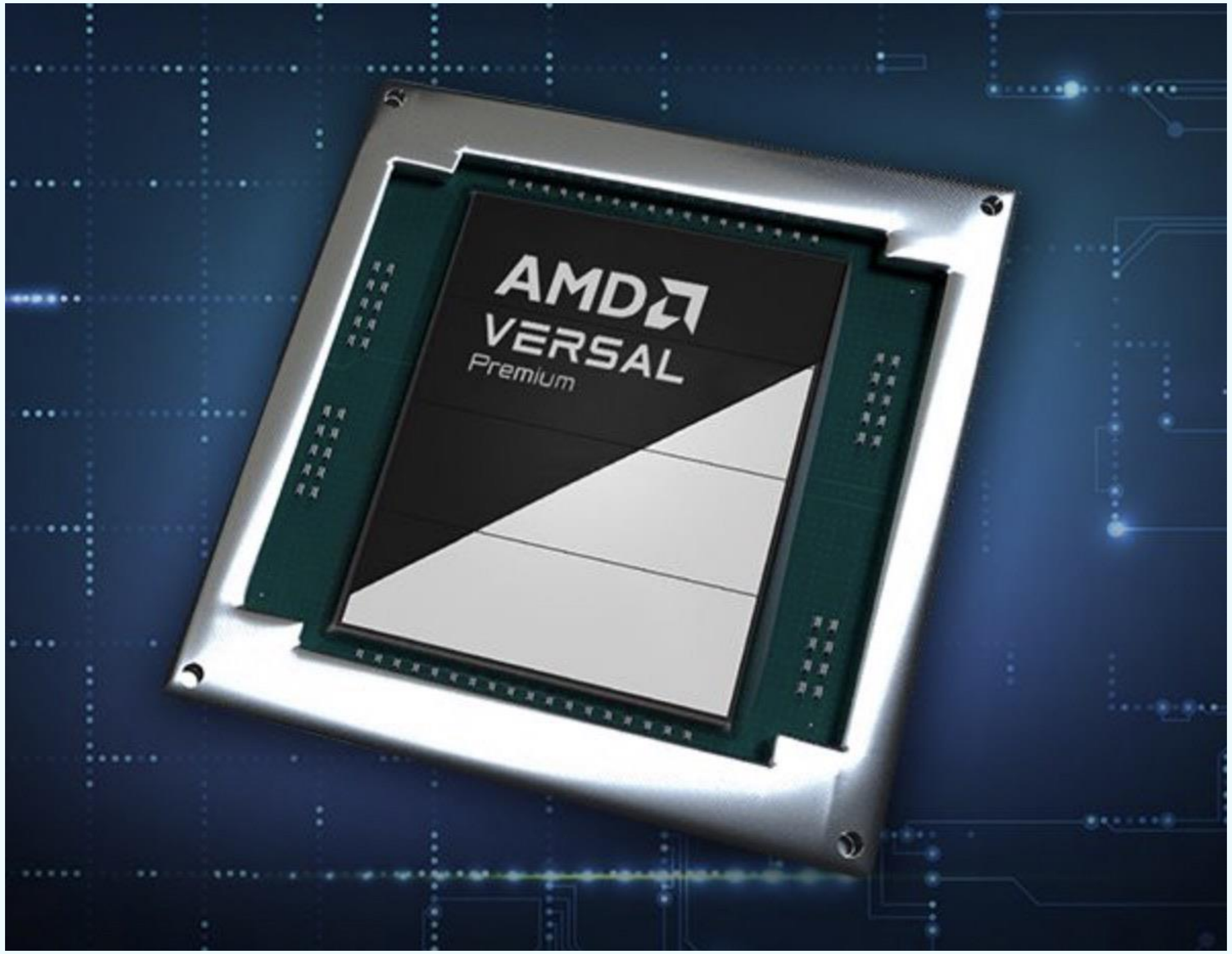

Services and integration

Data and control

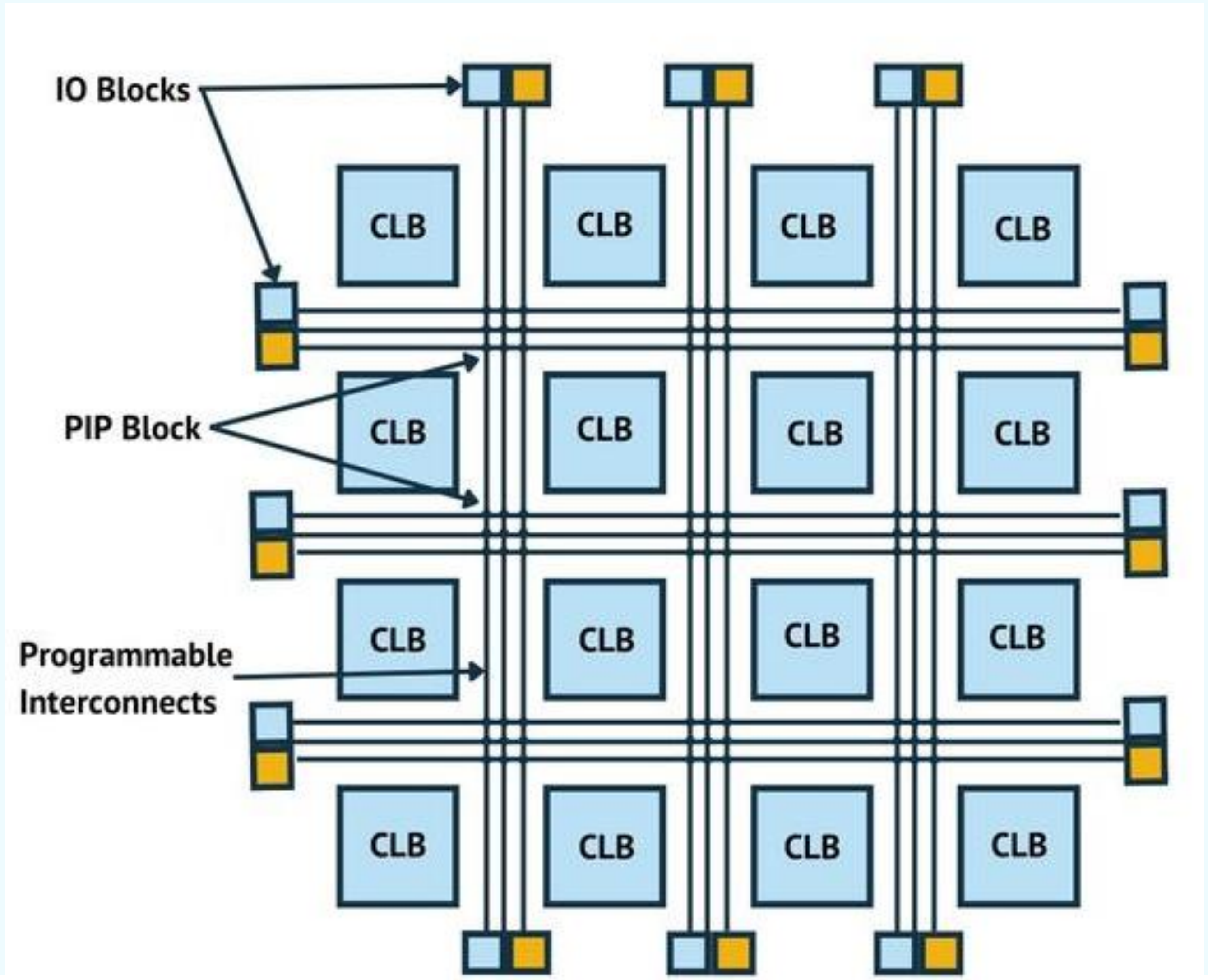
October 5th 2023 TWEPP23: ALICE ITS3: a bent stitched MAPS-based vertex detector - Speaker: O. Groettvik 15

[2023_10_05-TWEPP23-ITS3 - final.pdf (cern.ch)]

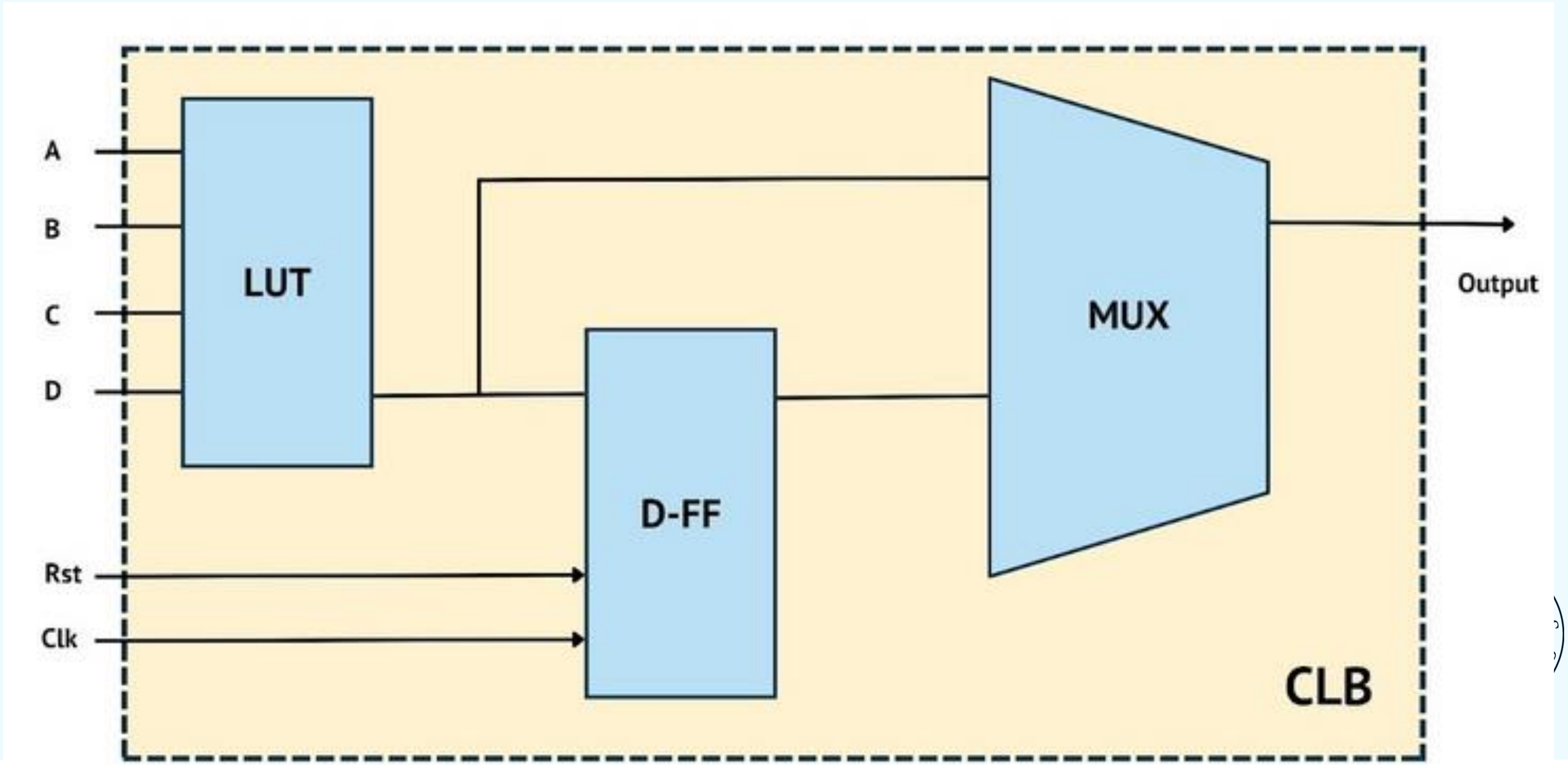
FPGAs Field Programmable Gate Arrays



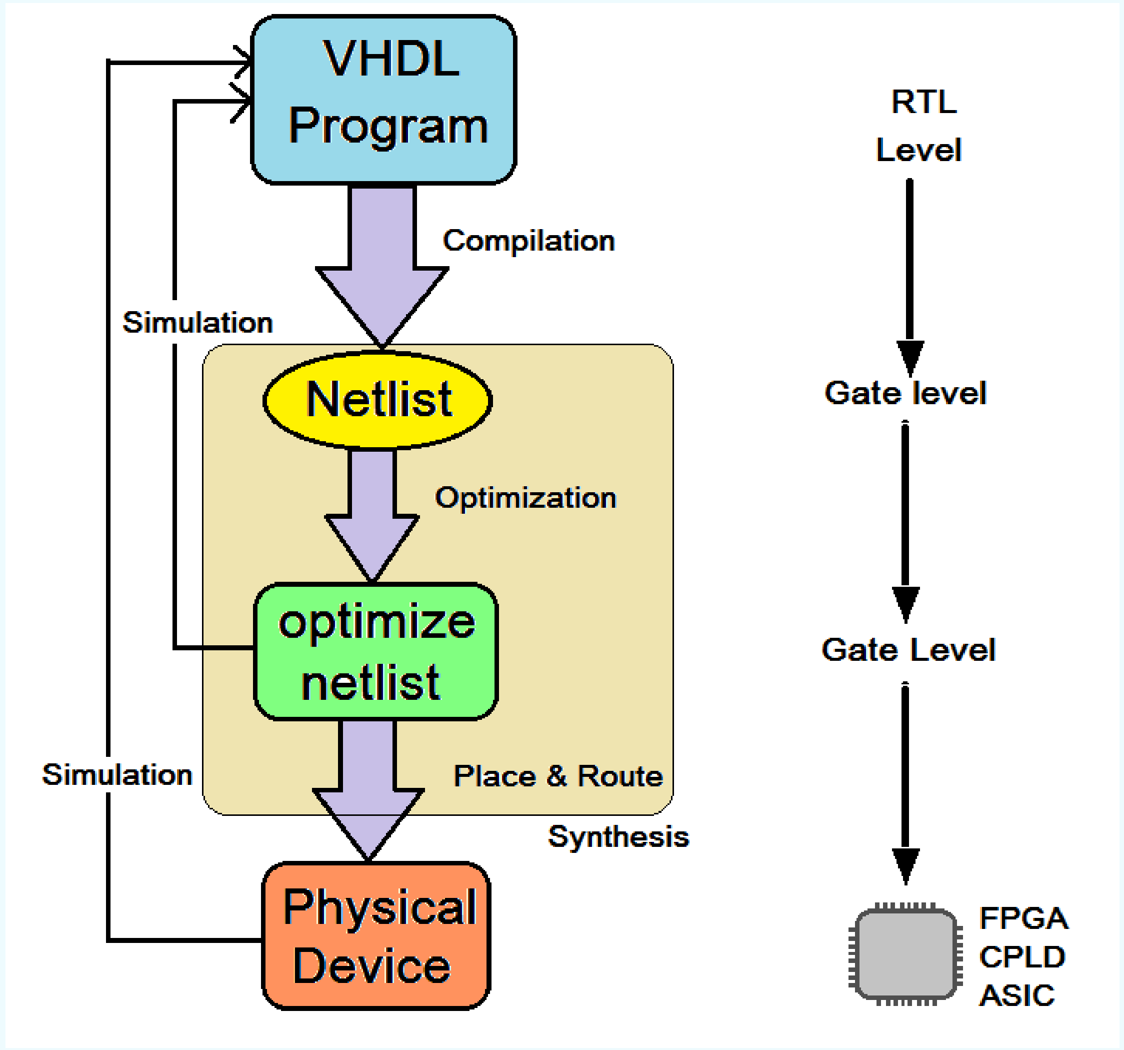
FPGAs Field Programmable Gate Arrays

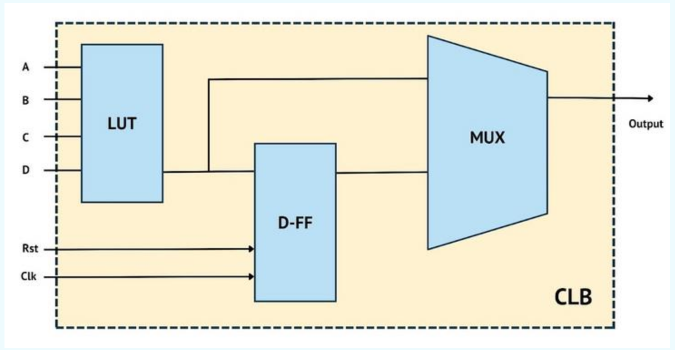


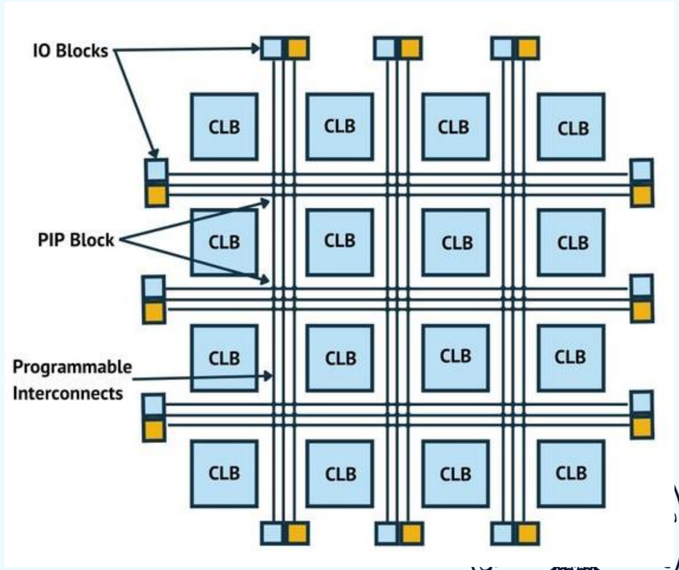
FPGAs Field Programmable Gate Arrays



FPGAs Field Programmable Gate Arrays

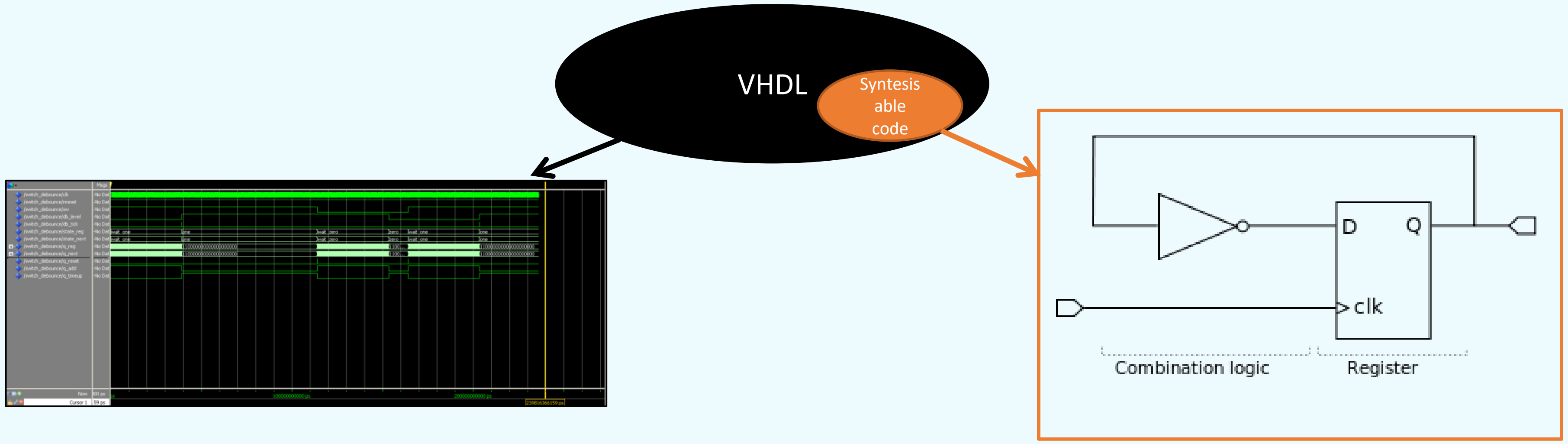






What is VHDL?

- VHDL is:
  - A high level design description language
  - A model that will either be used to synthesize HW or just used as a simulation model
    - Only a subset of the language can be used for synthesis

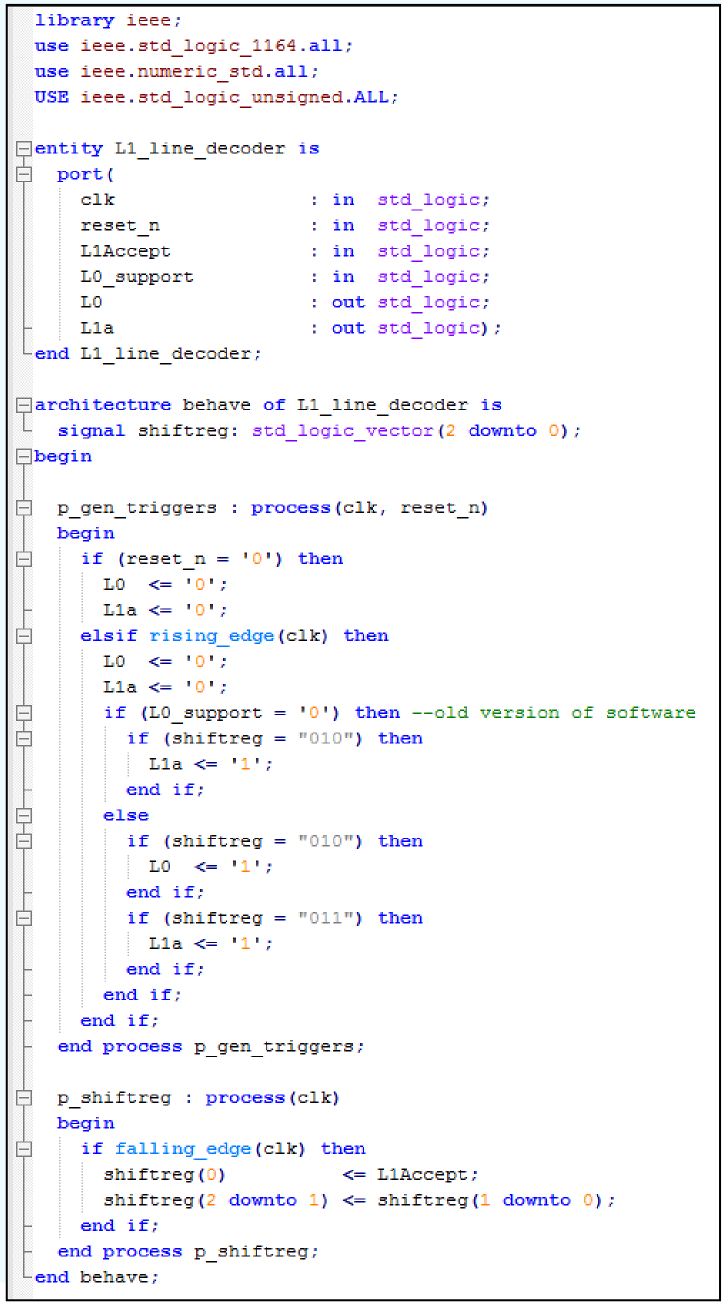


Default VHDL model

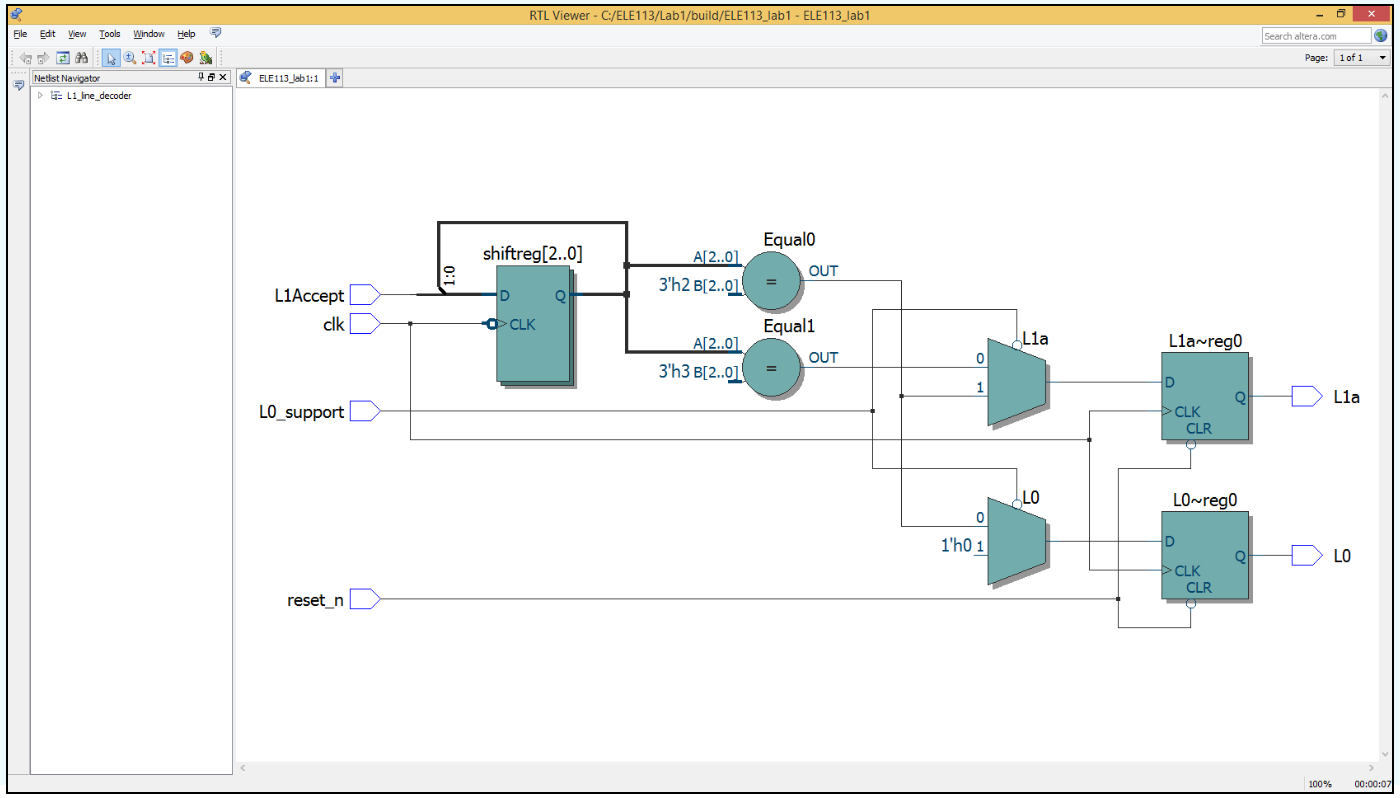
Libraries &
Packages headers

Interface definition
(input and outputs)

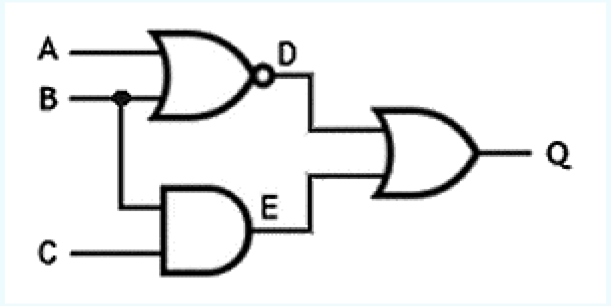
Functional/behavioural
implementation



```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
USE ieee.std_logic_unsigned.ALL;

entity L1_line_decoder is
  port(
    clk                 : in  std_logic;
    reset_n             : in  std_logic;
    L1Accept            : in  std_logic;
    L0_support          : in  std_logic;
    L0                  : out std_logic;
    L1a                 : out std_logic);
end L1_line_decoder;

architecture behave of L1_line_decoder is
  signal shiftreg: std_logic_vector(2 downto 0);
begin

  p_gen_triggers : process(clk, reset_n)
  begin
    if (reset_n = '0') then
      L0  <= '0';
      L1a <= '0';
    elsif rising_edge(clk) then
      L0  <= '0';
      L1a <= '0';
      if (L0_support = '0') then --old version of software
        if (shiftreg = "010") then
          L1a <= '1';
        end if;
      else
        if (shiftreg = "010") then
          L0  <= '1';
        end if;
        if (shiftreg = "011") then
          L1a <= '1';
        end if;
      end if;
    end if;
  end process p_gen_triggers;

  p_shiftreg : process(clk)
  begin
    if falling_edge(clk) then
      shiftreg(0)           <= L1Accept;
      shiftreg(2 downto 1) <= shiftreg(1 downto 0);
    end if;
  end process p_shiftreg;
end behave;
```



Concurrency



- THINK HARDWARE:
  - In real life the value @ **E** is always the result of **B and C**
  - Whenever **B** and/or **C** changes – **E** will change accordingly!
  - Similarly the value @ **D** changes when **A** and/or **B** changes
  - It might happen that the value @ **D** and @ **E** changes at the same time → *Concurrency*
- The following assignment statements are *concurrent* – they can be written in *any order*

```
D <= A NOR B;
E <= B AND C;
Q <= D OR E;
```

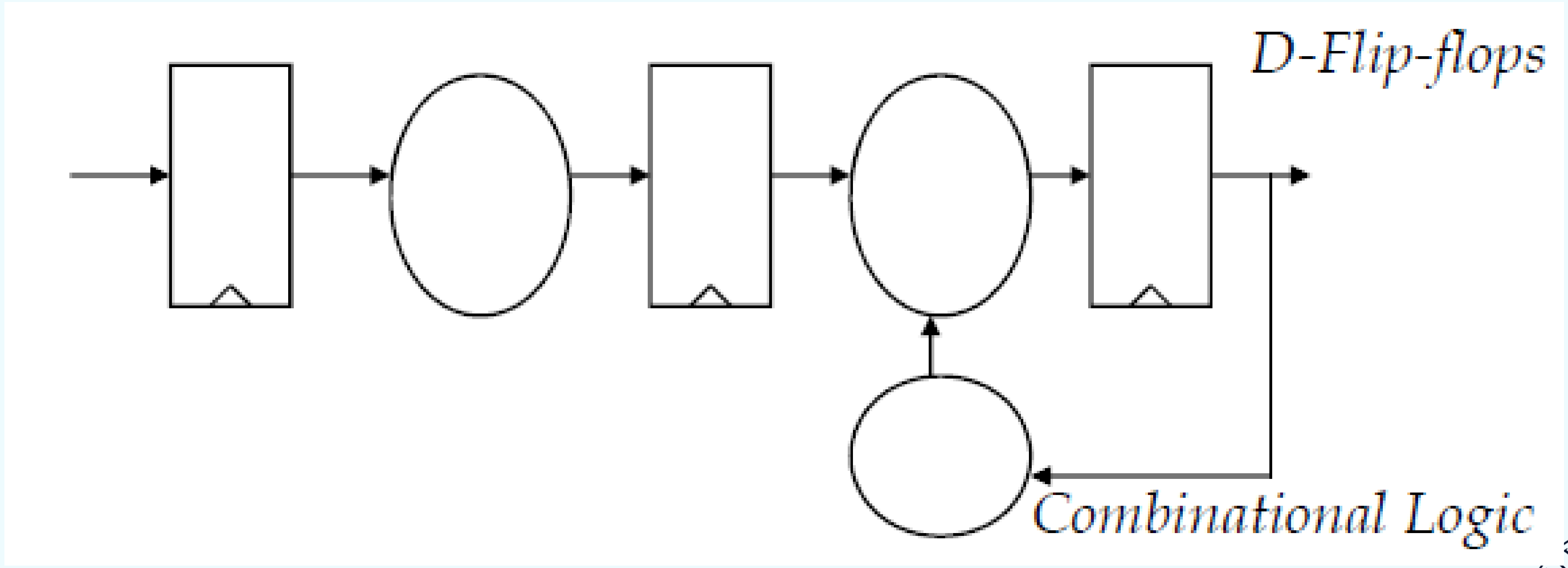
==

```
Q <= D OR E;
D <= A NOR B;
E <= B AND C;
```

==

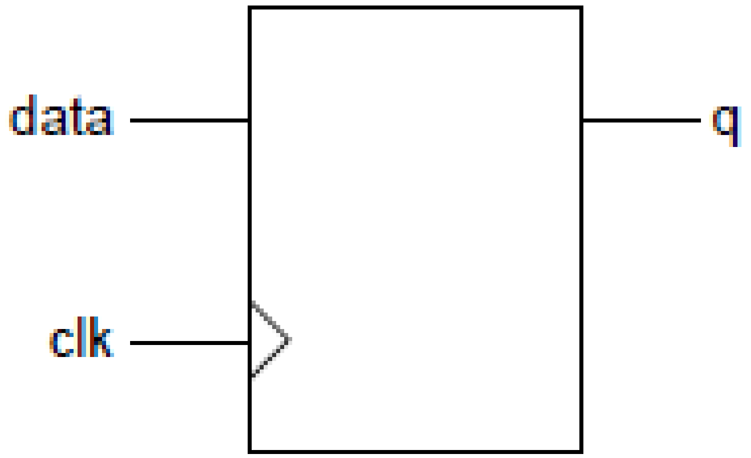
```
Q <= (A NOR B) OR (B AND C);
```

Digital design – Register Transfer Level (RTL)



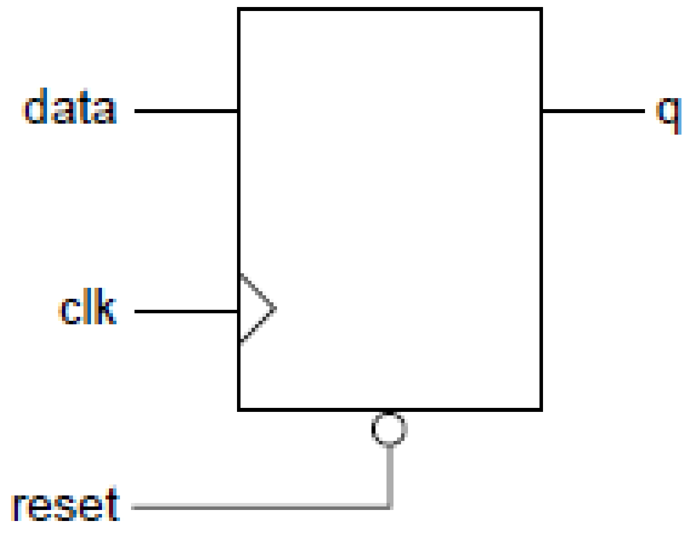
Typical VHDL constructs (1): D flip-flop

```
d_ff : process(clk)
begin
  if rising_edge(clk) then
    q <= data;
  end if;
end process d_ff;
```



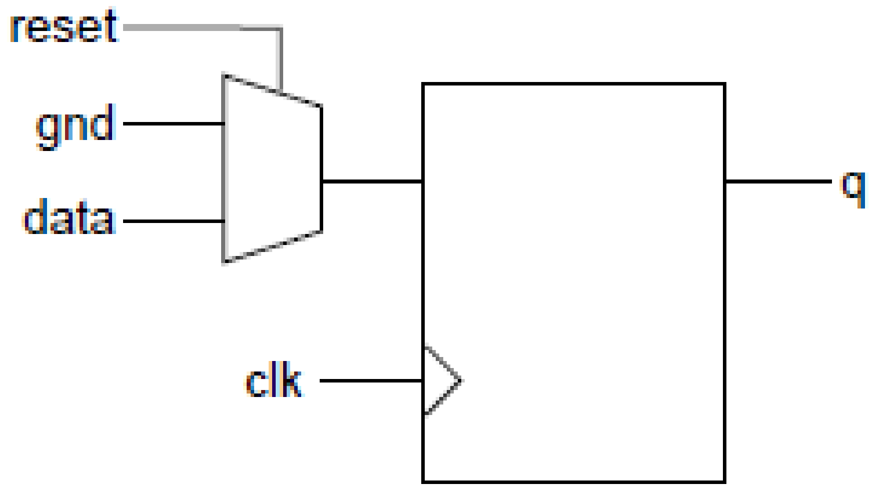
Typical VHDL constructs (2): D flip-flop with asynchronous reset

```
d_ff : process(clk, reset)
begin
  if (reset = ‘0') then
    q <= '0';
  elsif rising_edge(clk) then
    q <= data;
  end if;
end process d_ff;
```



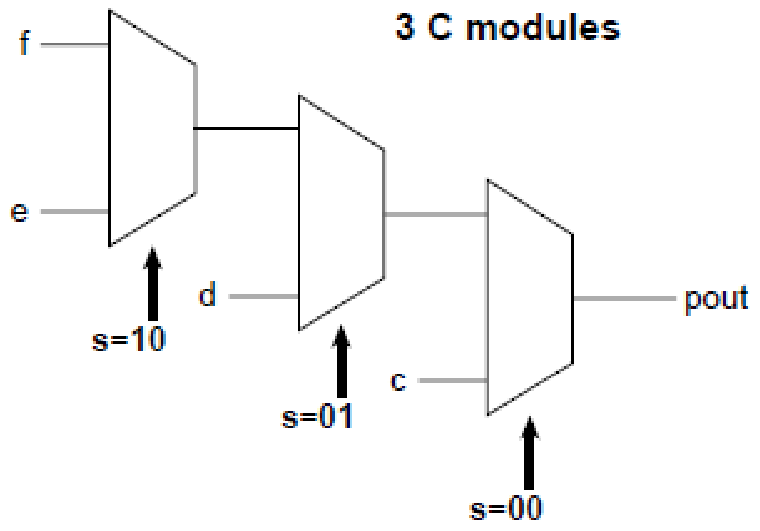
Typical VHDL constructs (3): D flip-flop with synchronous reset

```
d_ff : process(clk)
begin
  if rising_edge(clk) then
    if (reset = '0') then
      q <= '0';
    else
      q <= data;
    end if;
  end if;
end process d_ff;
```



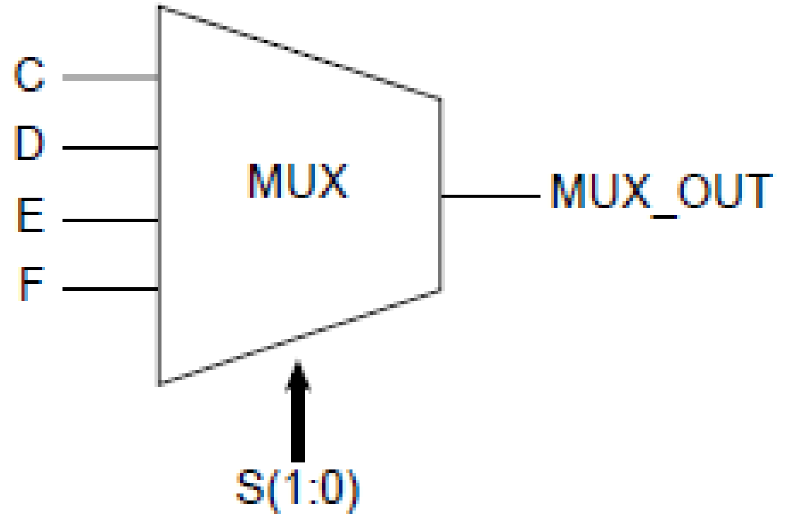
Typical VHDL constructs (4): Priority encoder

```
priority_encoder: process (a, c, d, e, f)
begin
  if (s = "00") then
    pout <= c;
  elsif (s = "01") then
    pout <= d;
  else
    pout <= f;
  end if;
end process priority_encoder;
```



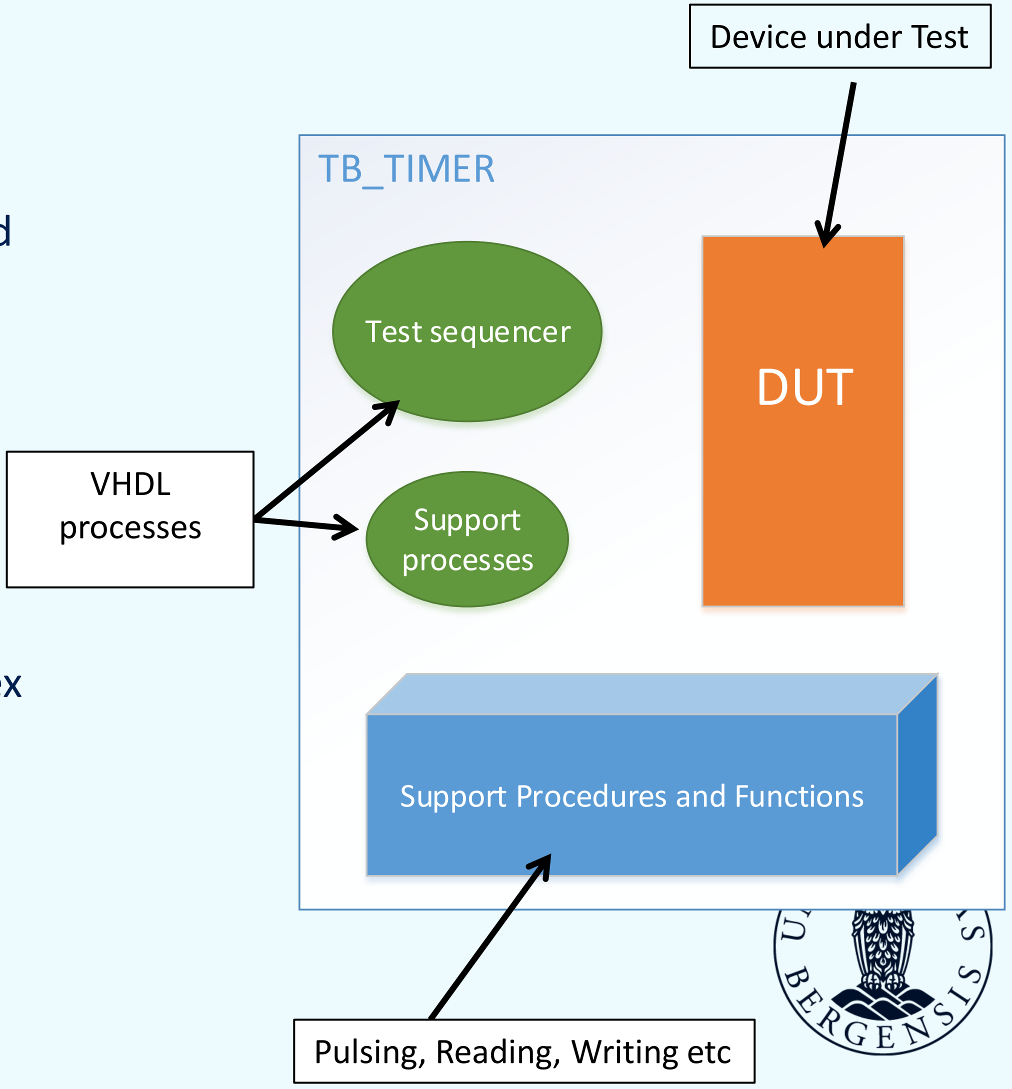
Typical VHDL constructs (5): Multiplexer

```
mux : process(s, c, d, e, f)
begin
  case s is:
    when "00"   => muX_out <= c;
    when "01"   => muX_out <= d;
    when "10"   => muX_out <= e;
    when others => muX_out <= f;
  end case;
end process mux;
```



VHDL for design… and VERIFICATION

- A verification specification is always needed, but
  - The verification spec. should not be too extensive/detailed
  - Required details could be added in a later spec. iteration
- A separate verification specification document is normally not needed
- Turn the verification spec into testcases
- We will use a simplistic verification model here. Various complex testbench models exist
  - UVVM
  - UVM
  - COCOTB …



Sources

- This assignment:

https://universityofbergen-my.sharepoint.com/:f:/g/personal/johan_alme_uib_no/EklesnHWAXtPivKrNOX7fFkBO8xOUjZeSWtmw0Awt_R2Aw?e=lOuxGv

Password: IRTG2024

- FYS4220 (UiO): https://fys4220.github.io/