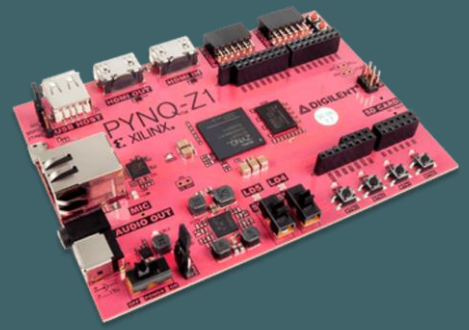
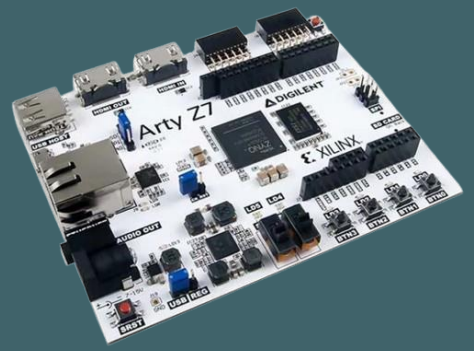
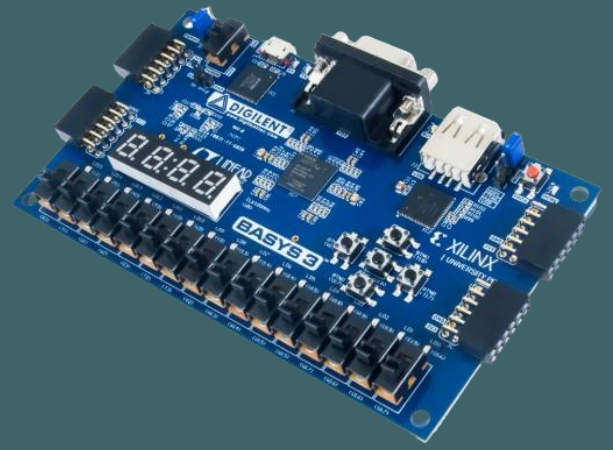






Intro to VHDL & Hardware Design

by Leeza Stetsenko the VP

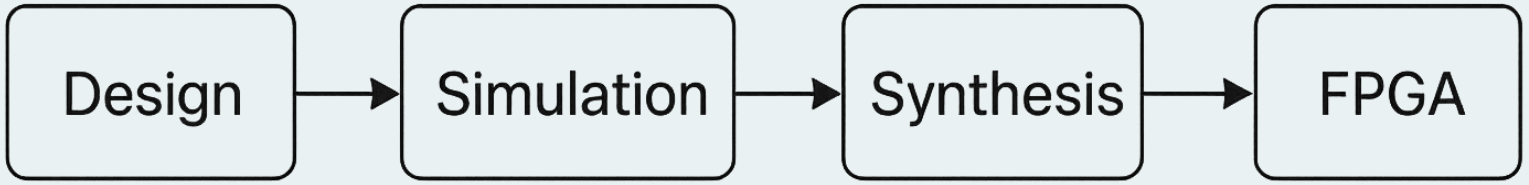
10/6/2025

Agenda

- Introduction to VHDL
- VHDL Design Structure
- Concurrent vs. Sequential

Motivation

- Very High Speed Integrated Circuit Hardware Description Language (VHDL)
- Allows for simulation, synthesis, and hardware implementation
- Key abstraction: RTL – Register-Transfer Level
- Used to describe registers, datapaths, FSMs, and control logic



VHDL vs Verilog

- Both used to describe digital hardware

VHDL	Verilog
Strongly typed, more explicit -> fewer synthesis mistakes	Loosely typed, faster for prototyping
Allows for multiple architectures per entity	Need to make new module for different functionality
Case insensitive	Case Sensitive
Non–blocking signal assignment: <= Variable / blocking assignment: :=	Nonblocking assignment: <= Blocking assignment: =

VHDL Design Structure

- Every VHDL file has architecture + entity
  - Entity -> defines I/O ports
  - Architecture -> defines internal behavior
- Multiple architectures can be written for the same entity

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity mux_2x1 is
5      port(
6          in0    : in  std_logic;
7          in1    : in  std_logic;
8          sel    : in  std_logic;
9          output : out std_logic);
10 end mux_2x1;
11
12 architecture case_statement of mux_2x1 is
13 begin
14     process(in0, in1, sel)
15     begin
16         case sel is
17             when '0' =>
18                 output <= in0;
19             when others =>
20                 output <= in1;
21         end case;
22     end process;
23 end case_statement;
```

Data Types and Operators

- Common libraries
  - ieee.std_logic_1164.all
  - ieee.numeric_std.all
- Important data types
  - std_logic, std_logic_vector, signed, unsigned, bit, Boolean, integer
- Special Logic values
  - ‘0’, ‘1’, ‘Z’, ‘X’, ‘-’
- Arithmetic requires casting with unsigned() or signed()
  - Eg. sum <= std_logic_vector(unsigned(a) + unsigned(b))

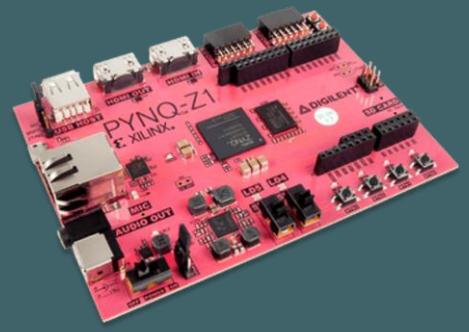
Concurrent vs Sequential Statements

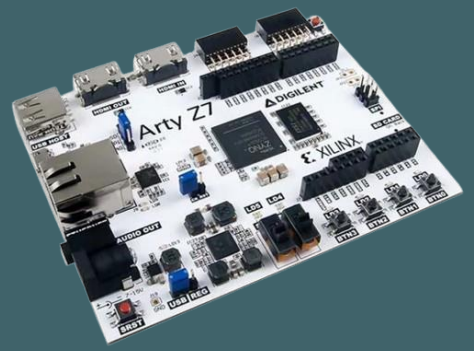
- Concurrent
  - Outside of a process
  - Executes simultaneously
  - Eg. with select, when else, direct assignments
- Sequential
  - Inside of a process
  - Executes in sequential order
  - Eg. If, case, loop
- Process triggers on changes in sensitivity list

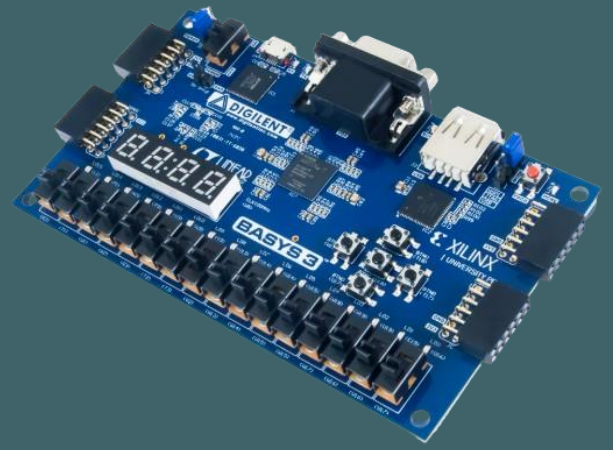
```
2   --Mux Architecure with Sequential statements
3   architecture if_statement of mux_2x1 is
4   begin
5       process(in0, in1, sel)
6       begin
7           if (sel = '0') then
8               output <= in0;
9           else
10              output <= in1;
11          end if;
12      end process;
13  end if_statement;
14
15  --Mux architecure with concurrent statements
16  architecture when_else of mux_2x1 is
17  begin
18      output <= in0 when sel = '0' else in1;
19
20  end when_else;
```

Combinational Logic Examples

- All inputs must be in sensitivity list
- Outputs must be assigned on all paths -> avoids latches
- Implement logic in multiple ways
  - If statement
  - Case statement
  - When else
  - With select



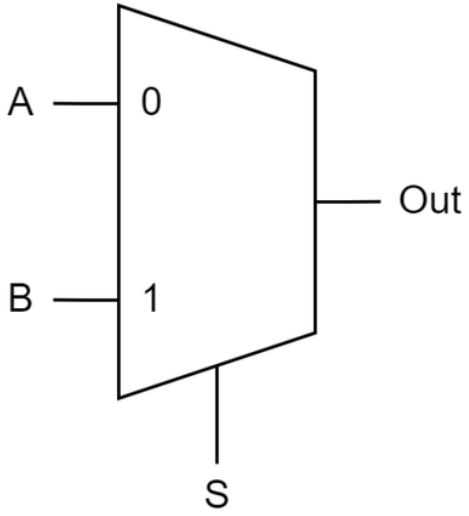




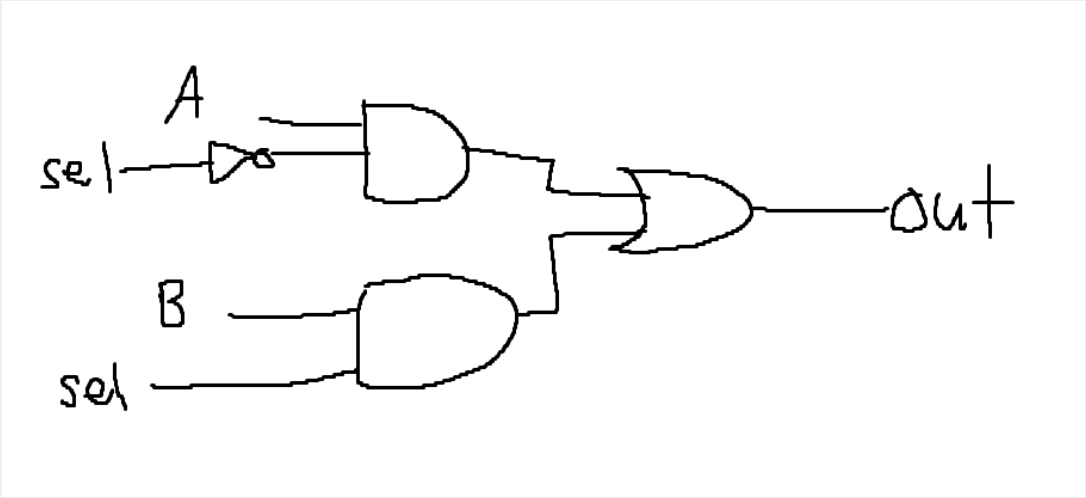
Combinational Logic Live Demo

by Antonio Marrero-Acosta
10/23/2025

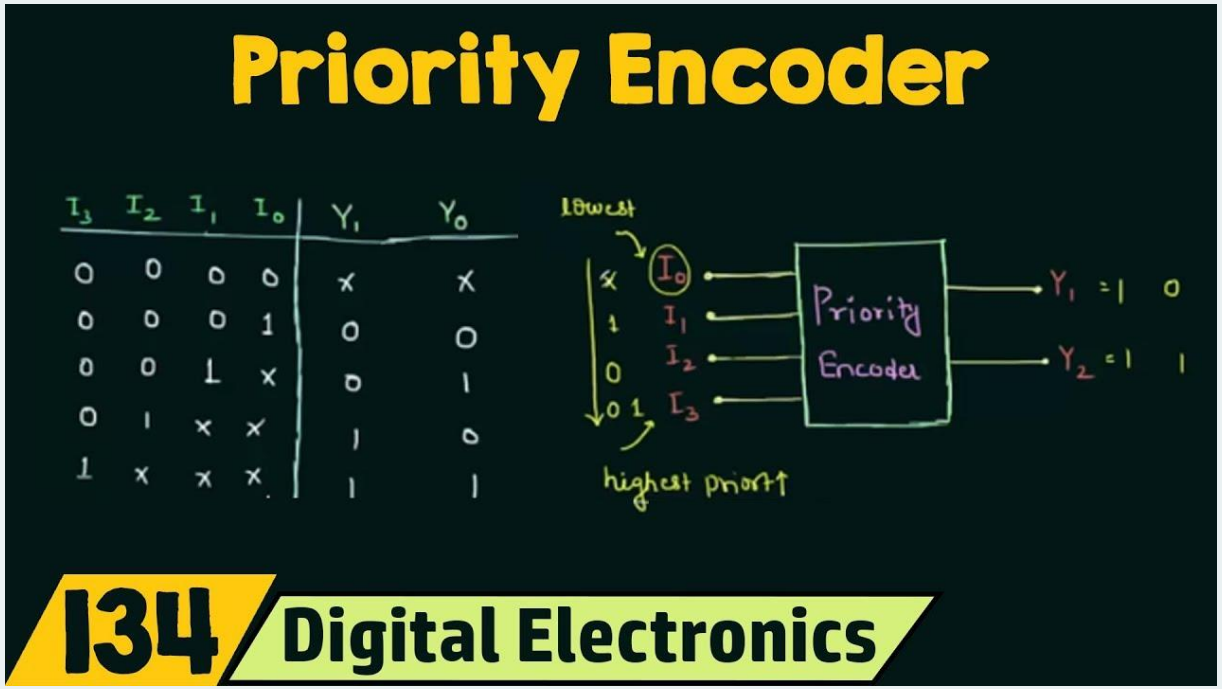
Multiplexer



S	A	B	Out
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



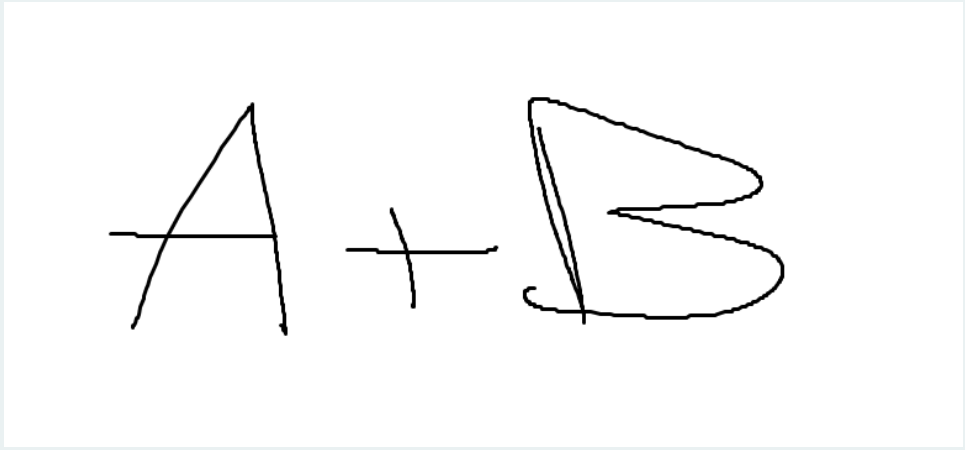
Priority Encoder

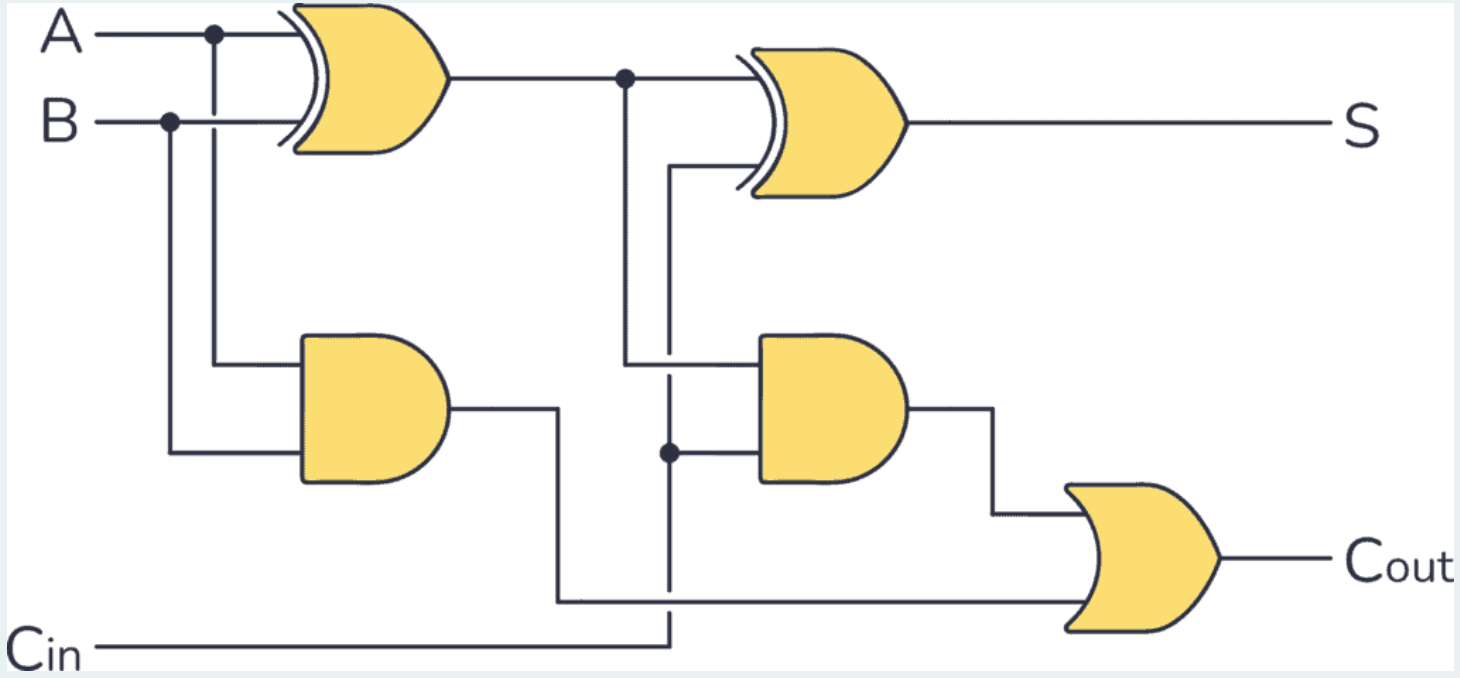


https://youtube.com/playlist?list=PLBInK6fEyqRjMH3mWf6kwqiTbT798eAOm&si=Ats4DwNcolX47nsB

Neso Academy Digital Electronics Playlist (for refresher on logic gates and Digital Logic)

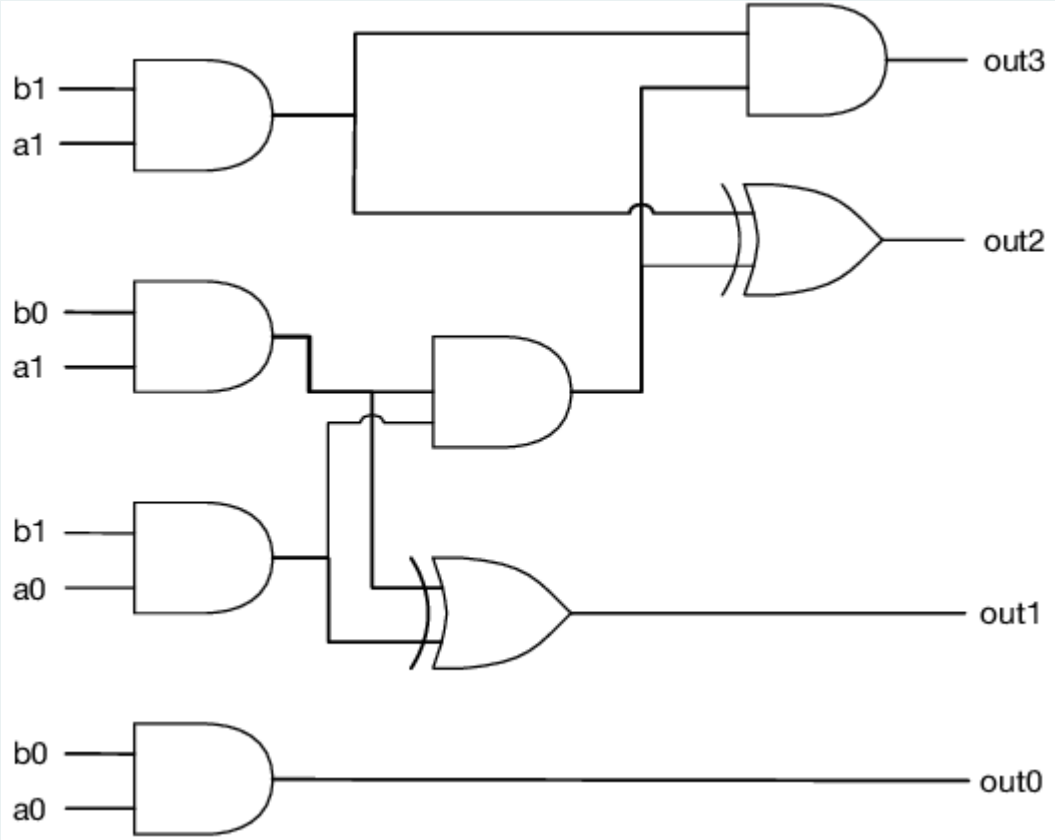
Adder



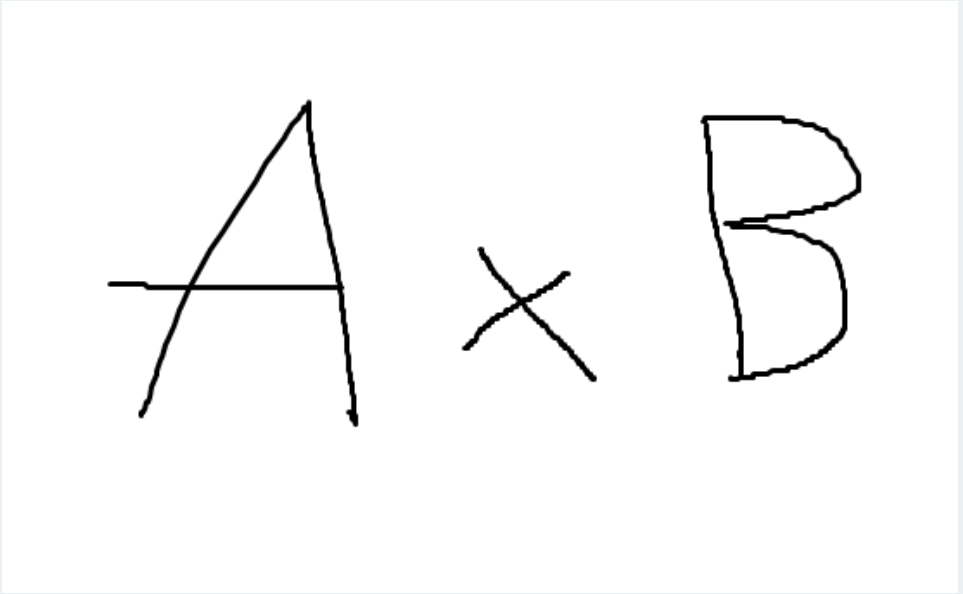


Full Adder

Multiplier



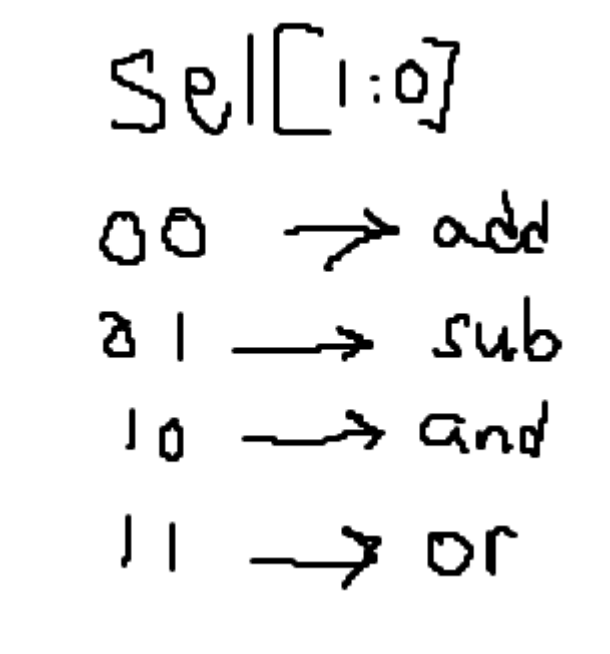
2-bit multiplier



ALU

- Too long to do here
- Left as an exercise to the reader
- Testbench will be provided if people want to do it on their own time

```
entity alu1 is
    generic (
        WIDTH : positive := 8);
    port (
        in0    : in  std_logic_vector(WIDTH-1 downto 0);
        in1    : in  std_logic_vector(WIDTH-1 downto 0);
        sel    : in  std_logic_vector(1 downto 0);
        neg    : out std_logic;
        pos    : out std_logic;
        zero   : out std_logic;
        output : out std_logic_vector(WIDTH-1 downto 0));
end alu1;
```



Thanks!