
# Hardware Programming HWP01 2022-2023

capturing a
FPGA
design with
VHDL

# Planning: theory

- First week
  - Introduction digital systems
  - Structured digital Design
  - RTL
- Second week
  - Introduction VHDL
  - Code structure and data types
  - Design verification
- Third week
  - Combinational versus sequential design
  - Concurrent and sequential code
- **Fourth week**
  - Components
  - Generics
- Fifth week
  - Introduction to state machines
  - Designing state machines
  - Advanced VHDL design

# Agenda

- **Discussion of previous week**
- Signals versus variables

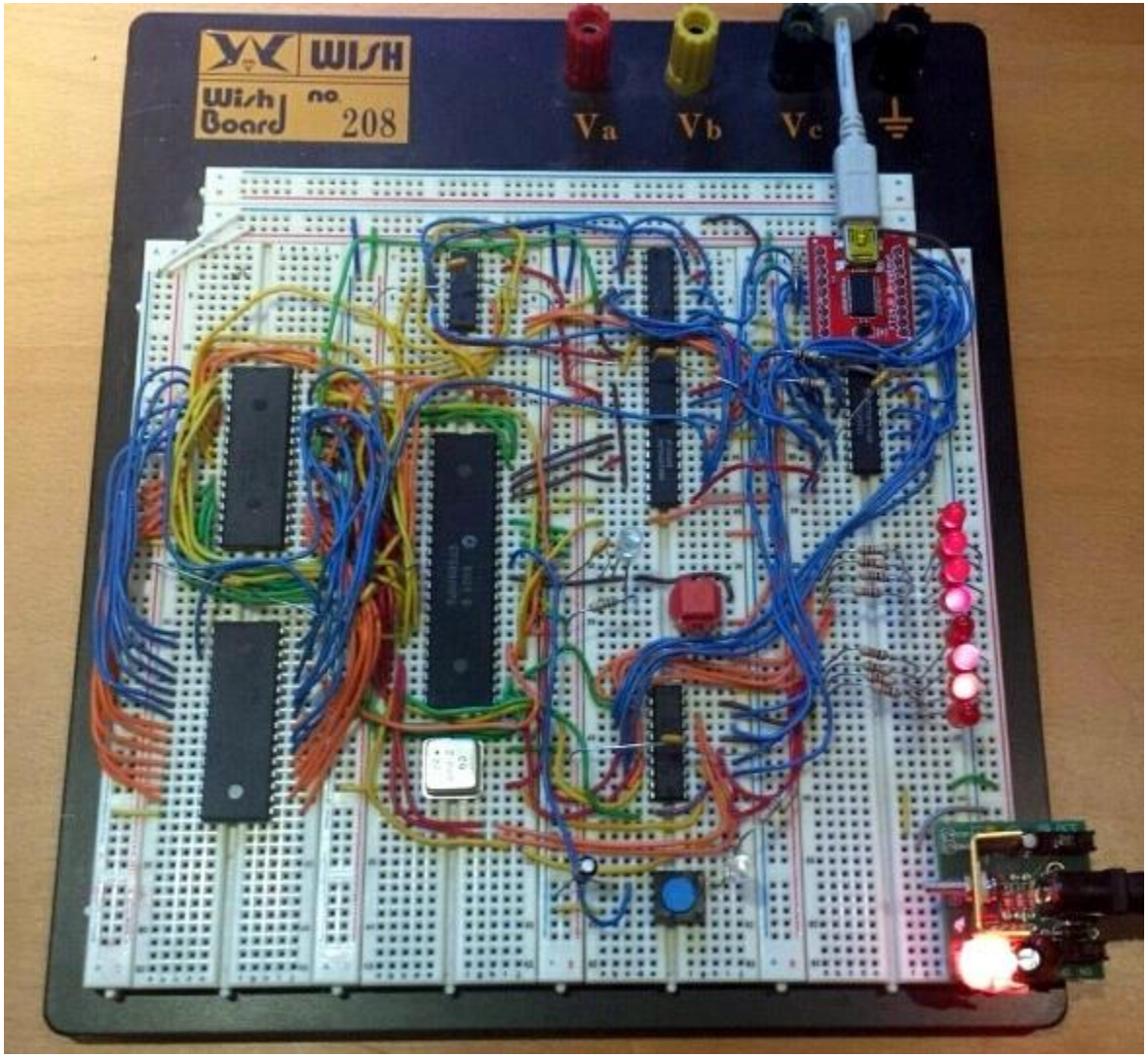
# Signals versus variables

- **SIGNAL** properties:
  - Can *ONLY* be declared outside a **PROCESS** but can be used within a **PROCESS**
  - Within sequential code the signal is not 'updated immediately' (at the end of the **PROCESS**)
  - Only a *single* assignment is allowed to a signal in the whole code (multiple assignments in **PROCESSES** are fine, but only the last one will be effective!)
- **VARIABLE** properties:
  - Can *ONLY* be declared inside a **PROCESS**
  - Is 'updated immediately' and can be used in the next line of code
  - *Multiple* assignments are not a problem

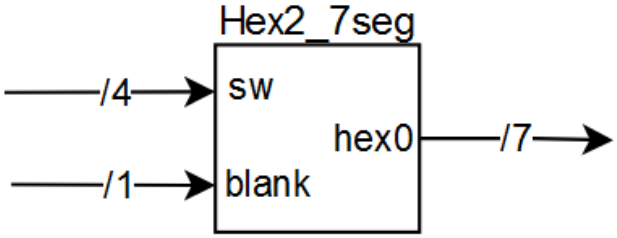
# Agenda

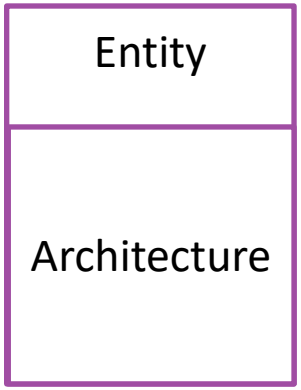
- Discussion of previous week
- Signals versus variables

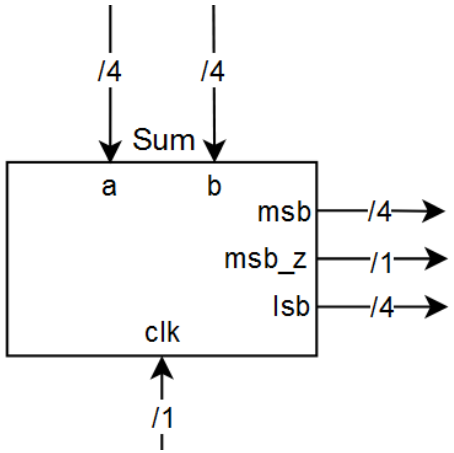
# Putting it together

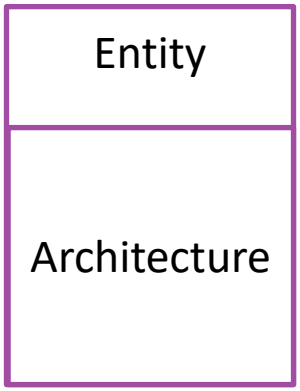


# Create (separately) validated components



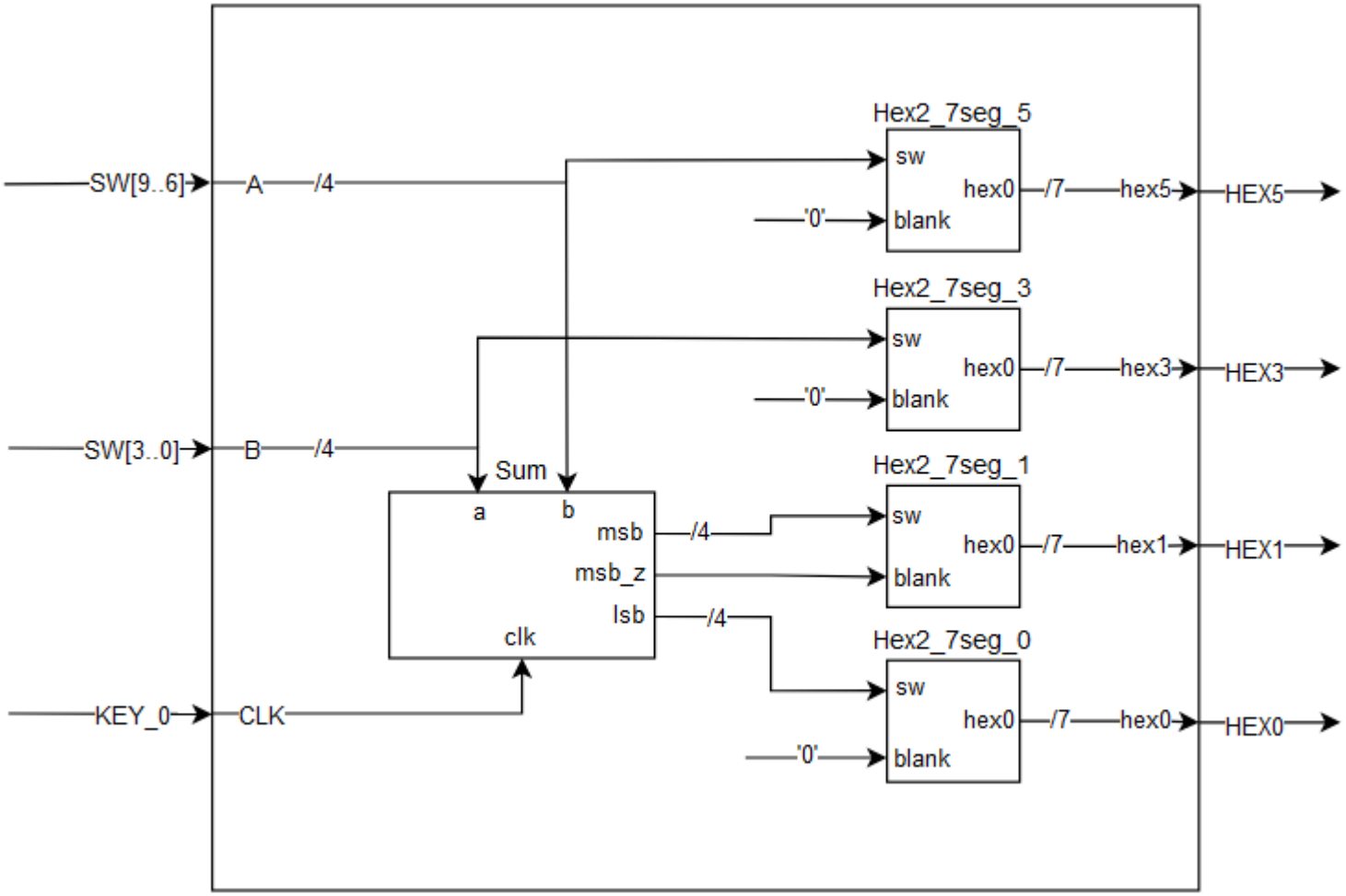



# Add components to your top level design

" Top Level Design:  multiply_2hexandDisplay.vhd"



# Instantiating components

```
begin
    hex5decoder: seven_segment_decoder port map(
        sw => A, blank => '0', hex0 => hex5
    );
    hex3decoder: seven_segment_decoder port map(
        sw => B, blank => '0', hex0 => hex3
    );
    hex1decoder: seven_segment_decoder port map(
        sw => sum_msn, blank => blank_msn, hex0 => hex1
    );
    hex0decoder: seven_segment_decoder port map(
        sw => sum_lsn, blank => '0', hex0 => hex0
    );
```

```
begin
    dut: assignment3
    port map(A => SW(9 downto 6), B => SW(3 downto 0), clk => not KEY(0), hex0 => HEX0, hex1 => HEX1, hex3 => HEX3, hex5 => HEX5);
    -- Extra: connect the LEDR outputs to the SW inputs
    LEDR <= SW;
end architecture;
```

# Generic statements

- Generic values are used for declaring global constant in a component
- What is the use of generics?
- For more information on generics and what else they are capable of, see CH 6.7

# Voorbeeld Generics

- With value

```
entity add_compare_cell is
   generic (
      NUM_BITS: natural := 16)
   port (
      a, b: in std_logic_vector(NUM_BITS-1 downto 0);
      comp: out std_logic;
      sum: out std_logic_vector(NUM_BITS downto 0));
end entity;
```

- Without value

```
generic (
   type bus_type;
   BUS_WIDTH: natural := 32;
   LAST_ADDRESS: natural);
```

# Summary

- Remember the differences between signals and variables
- Components are pre-validated VHDL library elements used to reduce development time of new products.