Hardware Programming HWP01 2020-2021

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
  - Signals and variables
- 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

Signals and Variables in VHDL (1/2)

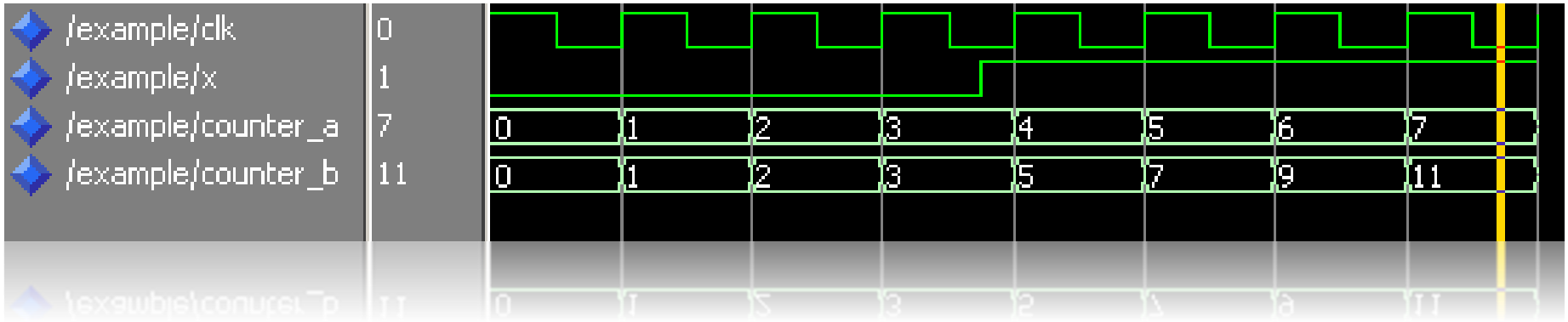
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity example is
        port (
                        clk:    in      std_logic;
                        x:      in      std_logic;

                        counter_a:      out integer range 0 to 15;
                        counter_b:      out integer range 0 to 15;
                );
end example;
```

Signals and Variables in VHDL (2/2)

```
architecture example of example is
        signal a: integer range 0 to 15 := 0;
begin
        process (clk)
                variable b: integer range 0 to 15 := 0;
        begin
                if rising_edge(clk) then
                        a <= a + 1;
                        b := b + 1;
                        if x = '1' then
                                a <= a + 1;
                                b := b + 1;
                        end if;
                end if;
                counter_b <= b;
        end process;
        counter_a <= a;
end example;
```

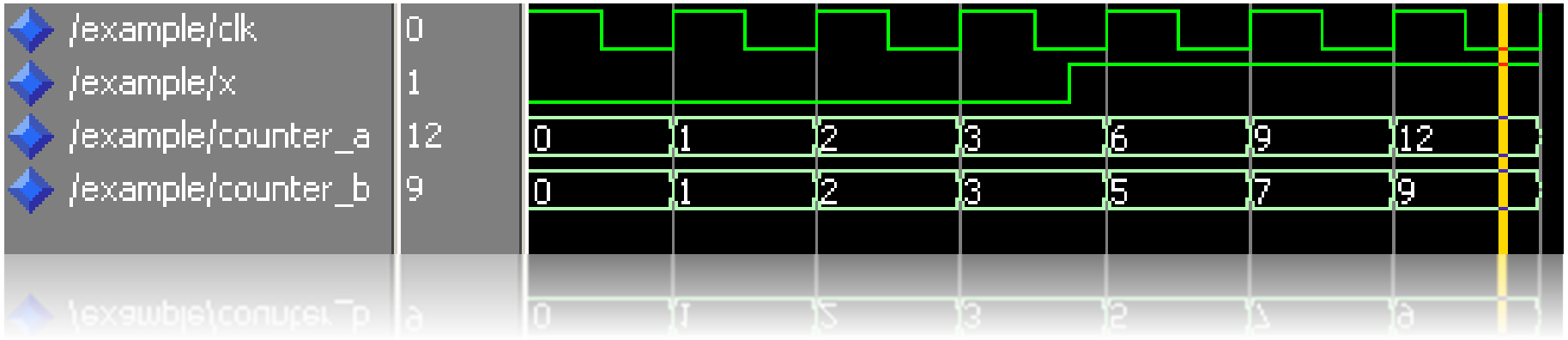
Variable can only be used inside the process!

Example



- When x is high
  - The variable counter is counted up twice
  - The signal counter is counted up only once
    - Only the last assignment is effective for a signal

Last signal assignment is only used

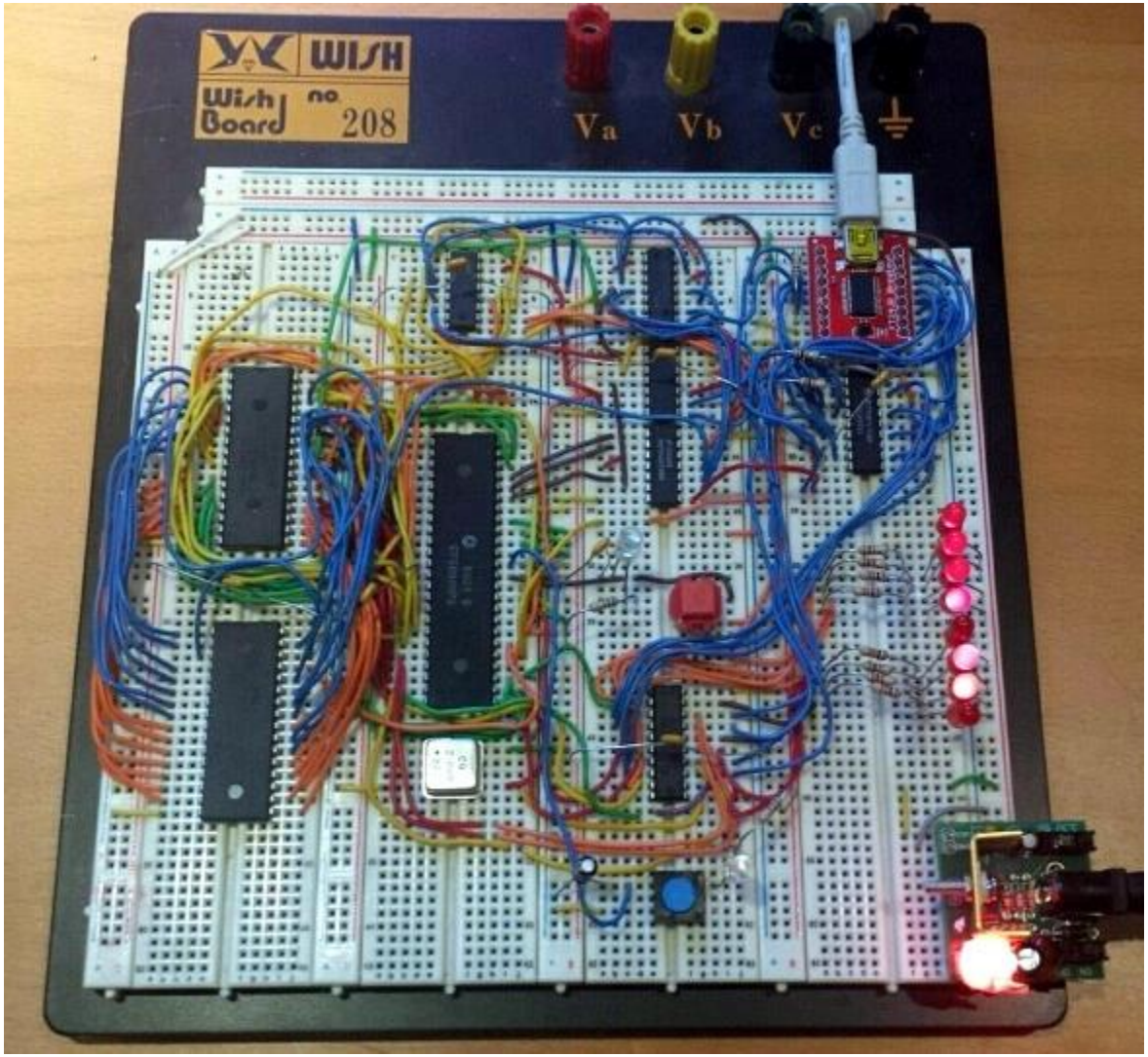


- Proof:
  - change a <= a + 1
  - to a <= a + 3
- Conclusion
  - Variables are useful when you have to do multiple assignments inside a process

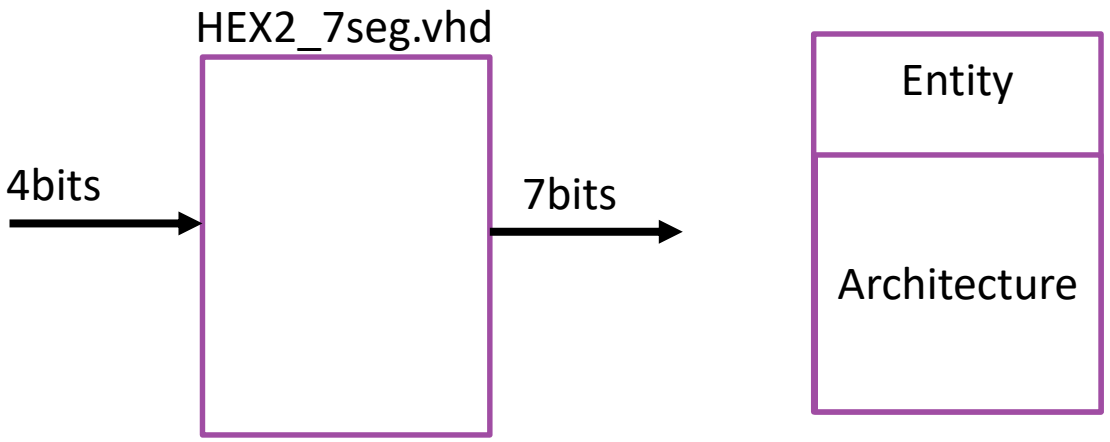
```
if rising_edge(clk) then
        a <= a + 1;
        b := b + 1;
        if x = '1' then
                a <= a + 3;
                b := b + 1;
        end if;
end if;
```

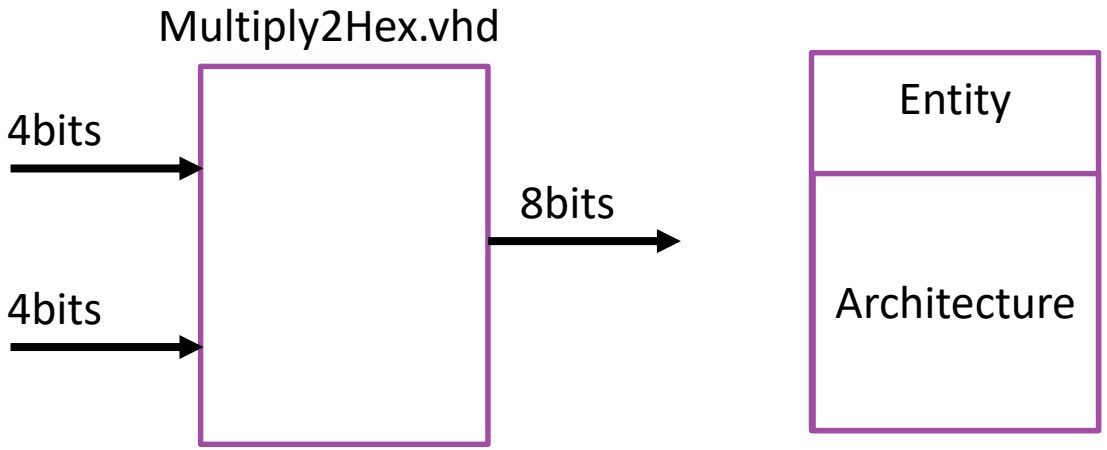
Formally:
<= is a concurrent statement!
:= is a sequential statement!

Putting it together

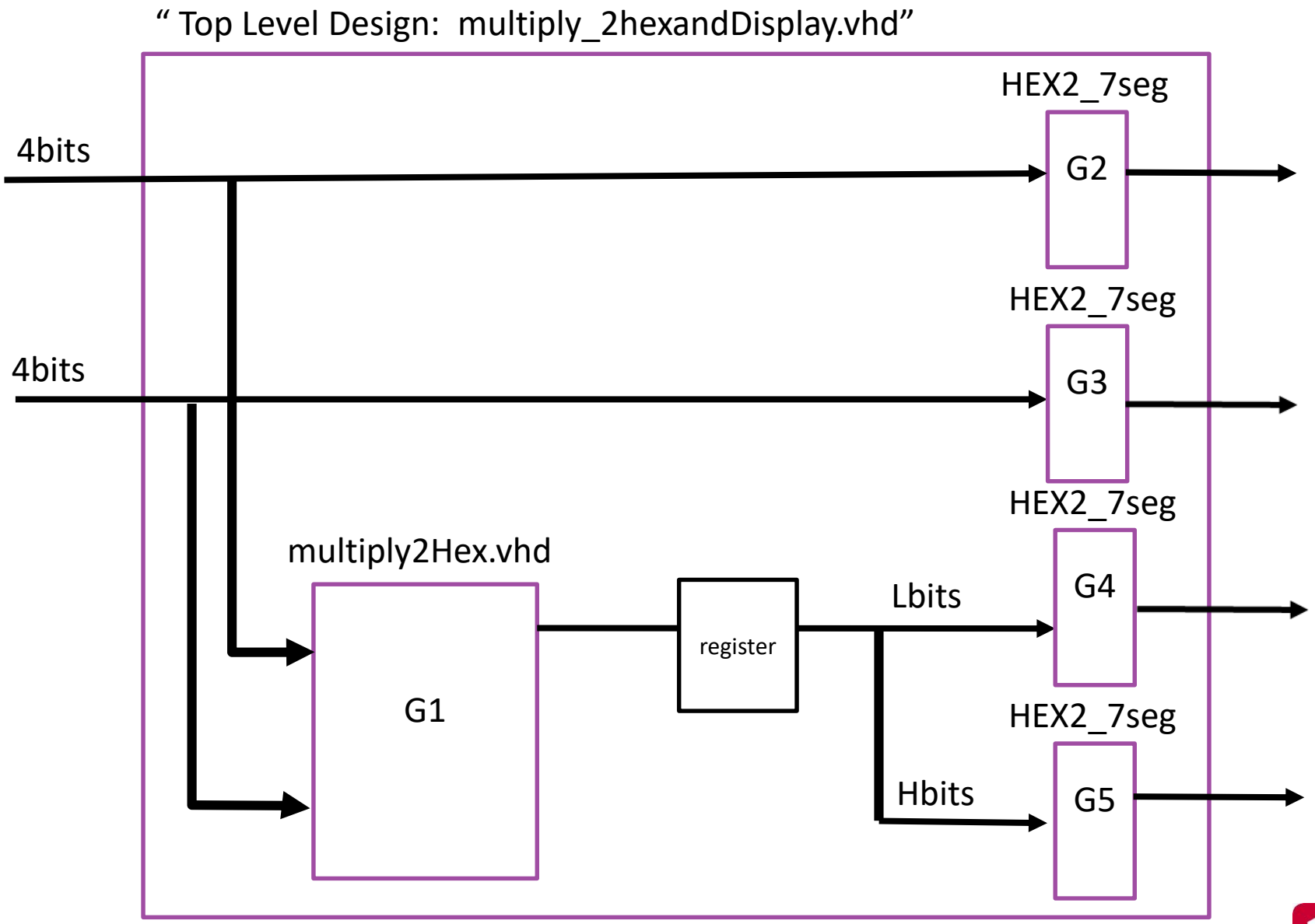


Create (separately) validated components





Add components to your top level design



ADD components and wire them together

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

ENTITY mult_2HexandDisplay is
  port
   (HexA,HexB: IN STD_LOGIC_VECTOR(3 downto 0);
segA,segB, Lout,Hout :OUT STD_LOGIC_VECTOR(6 downto 0));
   end;

ARCHITECTURE structure of mult_2Hexvals is

    component multiply2hex
      port (Hex1in,Hex2in: in STD_LOGIC_VECTOR(3 downto 0);
              res8: out STD_LOGIC_VECTOR(7 downto 0));
      end component;

    component hex2_sevenseg
      port (hex1: in STD_LOGIC_VECTOR(3 downto 0);
            sevseg: out STD_LOGIC_VECTOR(6 downto 0));
      end component;
```

```
signal resold, resnew : STD_LOGIC_VECTOR(7 downto 0);

Begin
 G1:  multiply2hex    port map (HexA,HexB,resold);
 G2: hex2_7seg port map (HexA,SegA);
 G3: hex2_7seg port map (HexB,SegB);
 G4: hex2_7seg port map (resnew(3 downto 0),Lout);
 G5: hex2_7seg port map (resnew(7 downto 4),Hout);

 resnew <= resold;   -- concurrent signal assignment

End ARCHITECTURE;
```

Components

Instantiation

- The architecture structure of mult_2Hexvals makes instances of multiply2hex AND hex2_7seg through the component instantiations at the bottom of the architecture (G1,G2,G3,G4, G5). The component names (multiply2hex AND hex2_7seg) are references to design entities defined elsewhere in the library. The instance labels (multiply2hex AND hex2_7seg) identify two specific instances of the components, and are mandatory.

Port Maps

- The ports in a component declaration must match the ports in the entity declaration one-for-one. The component declaration defines the names, order, mode and types of the ports to be used when the component is instanced in the architecture body. A component is declared once within any architecture, but may be instanced any number of times.

Association

- Signals in an architecture are associated with ports on a component using a port map. In effect, a port map makes an electrical connection between “pieces of wire” in an architecture (signals) and pins on a component (ports). The same signal may be associated with several ports - this is the way to define interconnections between components.

Summary

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