LINKÖPING UNIVERSITY
Department of Electrical Engineering

TSIU03, SYSTEM DESIGN

LECTURE 2

Kent Palmkvist
Kent.Palmkvist@liu.se

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

Linköping, 2021

TODAY

- From 1bit to several bits.
  - Review of binary number representations.
  - Word length.
- Numbers and binary representations in VHDL:
  - Package std_logic_1164 (std_logic and std_logic_vector)
  - Assignments of bits and concatenation.
  - Package numeric_std (signed and unsigned).
  - Integer.
- Complete hardware design process.
- Lab tools and devices.

BINARY NUMBERS

- How much is this binary number in decimal?

1001

a) 9
b) 5
c) -1
d) -7

BINARY NUMER REPRESENTATIONS

- Given a binary number $x_{n-1}x_{n-2}\ldots x_1x_0$, its value in decimal depends on the representation that is used for the number. The most common representations are:
  - Unsigned:

    $$x = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \ldots + x_1 2^1 + x_0 2^0$$

    Range: $[0, \quad 2^n - 1]$
  - 2's complement:

    $$x = -x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \ldots + x_1 2^1 + x_0 2^0$$

    Range: $[-2^{n-1}, \quad 2^{n-1} - 1]$
  - Sign and magnitude:

    $$x = \pm(x_{n-2}2^{n-2} + \ldots + x_1 2^1 + x_0 2^0)$$ x_{n-1} : sign bit

    Range: $[-2^{n-1}+1, \quad 2^{n-1} - 1]$

SIGN CHANGE

- 2’s complement: replace ones by zeros and zeros by ones and add one:

010010 ≡ 18	101110 ≡ -18
101101	010001
101110 ≡ -18	010010 ≡ 18

- Sign and magnitude: change the sign bit:

000101 ≡ 3

100101 ≡ -3

HOW MUCH…?

- How much is in decimal the binary number 11010011 if it is represented as:

a) Unsigned.

b) 2’s complement.

c) Sign and magnitude.

MORE QUESTIONS

- Which is the maximum value that can be represented using 8 bits in 2’s complement?
- And the minimum?
- If 011110 is an unsigned number, which 2’s complement number represents the same decimal value?
- If 011110 is an unsigned number, which number in sign and magnitude representation represents the same decimal value?

AND EVEN MORE QUESTIONS

- If 100001 is 2’s complement, can we find an unsigned number that represents the same decimal number? Why/Why not?
- If 100001 is unsigned, can we find a 2’s complement number that represents the same decimal number? Why? Which is the number?
- How many bits do we need to represent the decimal number 500 as a 2’s complement?
- And as unsigned?

WORD LENGTH

- The word length of a binary representation is the number of bits that it has.
- Most significant bit (MSB): the bit in the first position. It is called most significant because it has the highest weight.
- Least significant bit (LSB): the bit in the last position. It is called less significant because it has the lowest weigth.
- Examples:

  - The word length of 00101 is 5, its MSB is 0 and the LSB is 1.

  - The number 7 represented in 2's complement and word length 4 is 0111, and with word length 8, it is 00000111.

PACKAGE std_logic_1164

- Include the types **std_logic** and **std_logic_vector**.
- Both **std_logic** and **std_logic_vector** are binary representations.
- **std_logic** is for one bit and **std_logic_vector** for several bits.
- Values that an **std_logic** can take:

Value	Meaning	Synthesizable
'0'	Logic Value 0	Yes
'1'	Logic Value 1	Yes
'Z'	High Impedance	Yes
'U'	Unknown	No
'X'	Forcing Unknown	**No**

- Further explanation in [R6.3.1]

std_logic_vector

- A **std_logic_vector** is an array with several bits where each of them is an **std_logic**:

```
type std_logic_vector is array (natural range < >) of
std_logic;
```

- Definition of **std_logic_vector**:

```
signal a: std_logic_vector (3 downto 0);
signal b: std_logic_vector (7 downto 0);
```

- We use **downto** to define the range of bits. For n bits, the range goes from n-1 downto 0. Thus, in the example a has 4 bits and b 8.
- We can select bits from an **std_logic_vector** and assign them to other signals. They must have the same word length!!

```
b(5 downto 4) <= a(2 downto 1);
c <= a(3); -- here c must be an std_logic and –- is used
       -- to add comments in the code.
```


ASSIGNING BITS

- Which of these statements are right or under which conditions are they correct?

```
z <= a;
z(2) <= a(3);
z <= a(3);
z(2 downto 1) <= a;
z(4 downto 2) <= "000";
z(6 downto 4) <= a(3 downto 0);
z(4 downto 6) <= a(0 downto 2);
z(3 downto 2) <= '1';
z(3 downto 2) <= (others => '1');
z <= (others => '0');
z <= (3 => '1', others => '0');
```

Note that ' ' is used to assign fixed values to an std_logic and " " to and std_logic_vector.

ASIGNING BITS

- Which of these statements are right or under which conditions are they right?

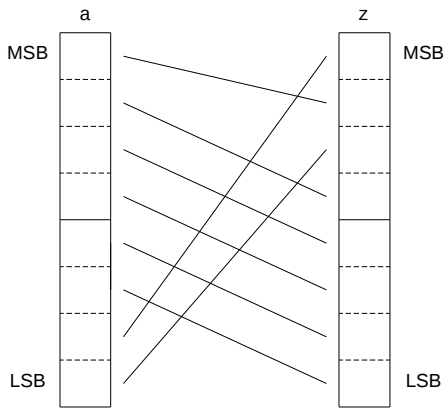
```
z <= a;     If z and a have the same word length.
z(2) <= a(3);  Ok!
z <= a(3); If z only has one bit (is an std_logic).
z(2 downto 1) <= a;    a must have two bits.
z(4 downto 2) <= "000";         Ok!
z(6 downto 4) <= a(3 downto 0);     Wrong: different word length.
z(4 downto 6) <= a(0 downto 2);     Wrong: 4 < 6 and 0 < 2.
z(3 downto 2) <= '1';           Wrong: different word length.
z(3 downto 2) <= (others => '1');   Ok! New command: others
z <= (others => '0');           Ok! Sets all the bits to '0'.
z <= (3 => '1', others => '0');     Ok!
```


MORE EXAMPLES

- Given the signals:

```
signal a: std_logic_vector (7 downto 0);
signal z: std_logic_vector (7 downto 0);
```

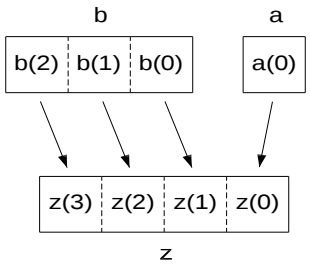
write the VHDL code that is needed to obtain z from a according to:



&

- The operator & concatenates two vectors (**std_logic** or **std_logic_vector**).
- Example:

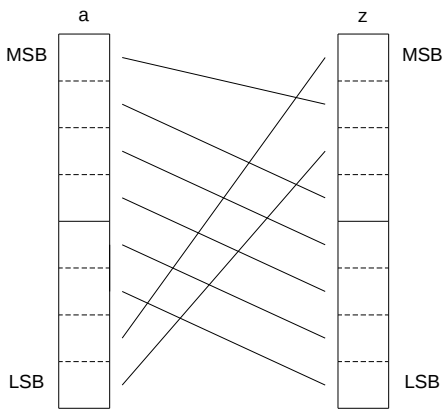
```
  …
  signal a: std_logic;
  signal b: std_logic_vector (2 downto 0);
  signal z: std_logic_vector (3 downto 0);

begin

  z <= b & a;
  …
```



- The sizes of the vectors have to match!

USING &

- Can you now describe this circuit in one line of VHDL code?



BACK TO THE BEGINNING

- Which decimal number does this **std_logic_vector** represent?

```
…
  signal a: std_logic_vector (3 downto 0);
begin
  a <= "1001";
…
```

a) 9
b) -1
c) -7
d) ?

numeric_std

- Used to represent binary numbers in VHDL.
- The type **unsigned** is used to represent unsigned numbers:

```
type unsigned is array (natural range < >) of std_logic;
signal a: unsigned (3 downto 0);
```

- The type **signed** is used to represent numbers in 2's complement.

```
type signed is array (natural range < >) of std_logic;
signal b,c,z: signed (3 downto 0);
```

- **signed** and **unsigned** represent numbers (**std_logic_vector** does not) and, therefore, it is possible to use comparison operators with them (which is not allowed for **std_logic_vectors**):

```
z <= a when a > b else b;
```

- By the way, which mathematical function does this line of code implement? Can you draw a circuit that calculates this function?

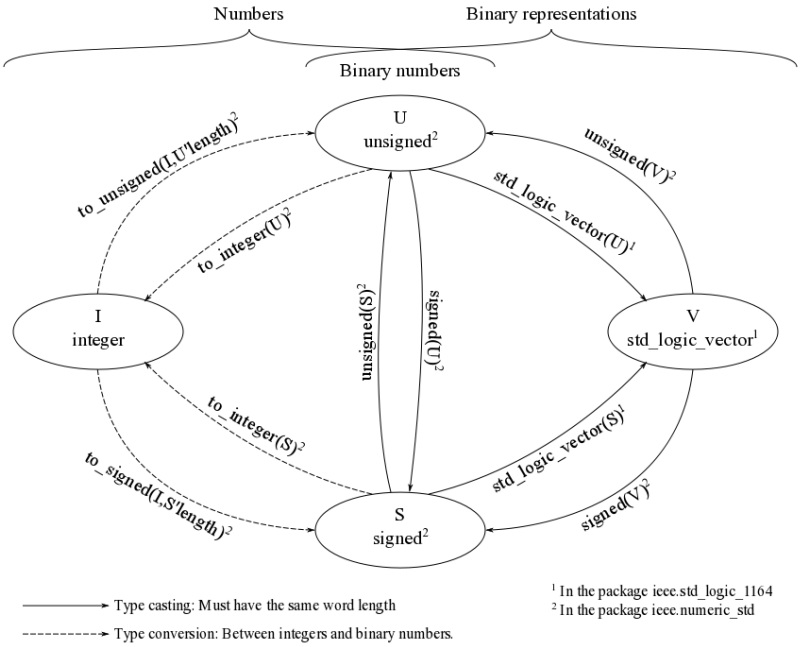
INTEGERS

- The type **integer** is built-in in VHDL, i.e., it comes by default and no library is needed to use it.

```
type integer is range -214783648 to +2147483647
```

- Its range corresponds to 32-bit numbers in 2's complement.
- Integers should be used carefully in VHDL, only in very specific cases. Note that by using **std_logic_vector**, **unsigned** and **signed** we have knowledge about the values of the bits and the ranges, so we can control the operations that we do with them and we can know which circuit is described. Using integers we lose this knowledge. This may lead into circuits that do not behave as expected or take much more resources than needed. We will discuss when to use integers in the next lecture.

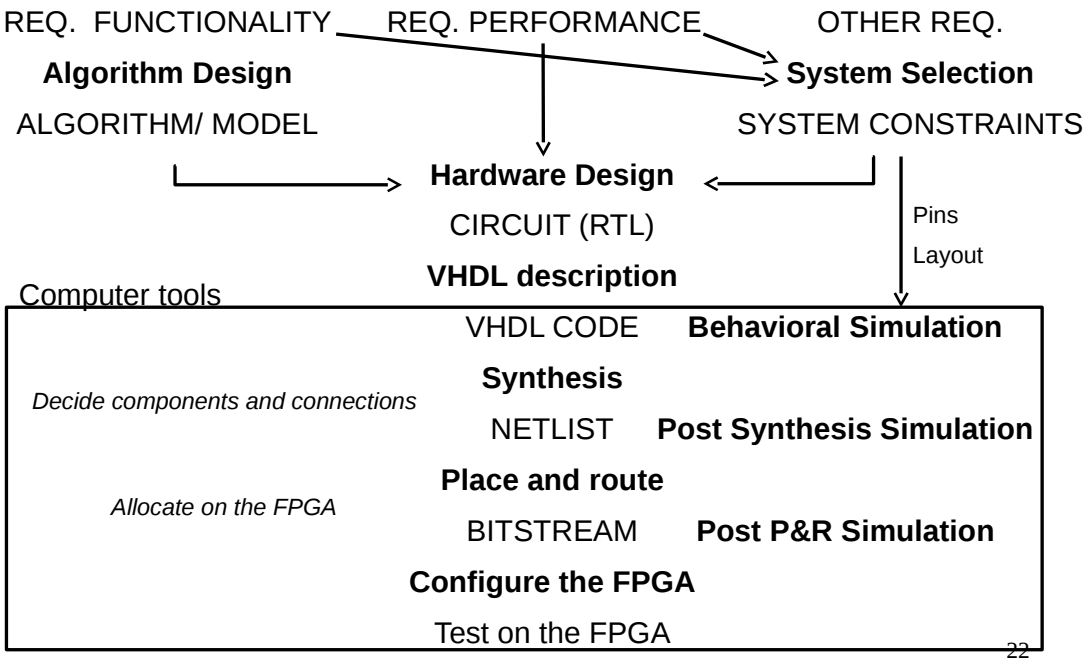
TYPE CASTING AND CONVERSION



EXAMPLES

```
signal a: std_logic_vector (6 downto 0);
signal b: unsigned (6 downto 0);
signal c: integer;
signal d: signed (8 downto 0);

begin

b <= unsigned (a);
c <= to_integer (b);
d <= to_signed (c, 9);
```


COMPLETE HW DESIGN PROCESS



LAB TOOLS AND DEVICES

- Altera Quartus: tool for analysis and synthesis of HDL designs (from the VHDL CODE to the configuration of the FPGA). If you want to try it at home, you can get a Quartus free license (select the 13.0 version to match lab setup):

https://www.altera.com/downloads/download-center.html

- ModelSim: Advanced tool for simulations.
- DE2-115: Development board that includes the FPGA.

REMOTE USE OF LAB

- Tools run on windows 10. The software is only installed in MUXEN3, MUXEN4, GRINDEN and TRANSISTORN lab (all at ISY department). You will have physical access to MUXEN4 and TRANSISTORN.
- Remote control of windows machines is possible. The university support rdp protocol control of some machines.
- IMPORTANT!: If you use rdp you lock the machine from other students. You MUST therefore check the schedule of the lab and verify that it is not booked before connecting.
- More information at https://rdpklienter.edu.liu.se

CHECKLIST FOR LECTURE 2

- Binary number representations: unsigned, signed, sign and magnitude, range, word length, MSB, LSB.
- VHDL language: std_logic_1164, std_logic_vector , '0', '1', 'Z', 'U', 'X', numeric_std, signed, unsigned, integer, downto, &, others, ' ', " ", --, <=, casting, conversion.
- Hardware design process: simulation, synthesis, netlist, place and route, bit stream.
- Tools and devices: Altera Quartus, ModelSim, DE2-115.

AT HOME

- Register for the course if you have not done it yet.
- Review the checklist for lecture 2 and check that you understand all the concepts and you know how to use them.
- Finish the Assignment 1. It has to be submitted in Lisam before the beginning of lecture 5.