LINKÖPING UNIVERSITY
Department of Electrical Engineering

TSIU03, SYSTEM DESIGN

LECTURE 8

Kent Palmkvist
Kent.Palmkvist@liu.se

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

Linköping, 2021

TODAY

- Reminder 2's complement vs sign magnitude
- Designing a Hardware System.
- Debugging Strategies.

From Lecture 2: Binary numbers

- 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.
  - Unsigned (VHDL datatype 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 (VHDL datatype signed):

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

    Range: $[-2^{n-1}, \quad 2^{n-1} - 1]$
  - Sign and magnitude (NOT a datatype used in VHDL):

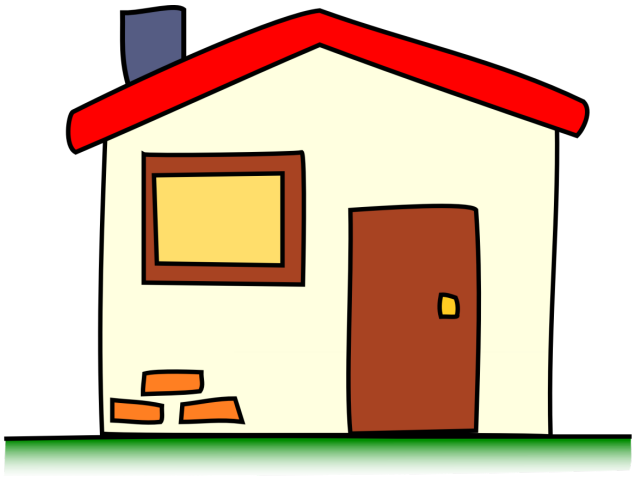
    $x = \pm(x_{n-2}2^{n-2} + \ldots + x_1 2^1 + x_0 2^0)$ x_{n-1} : sign bit (not included in magnitude)

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

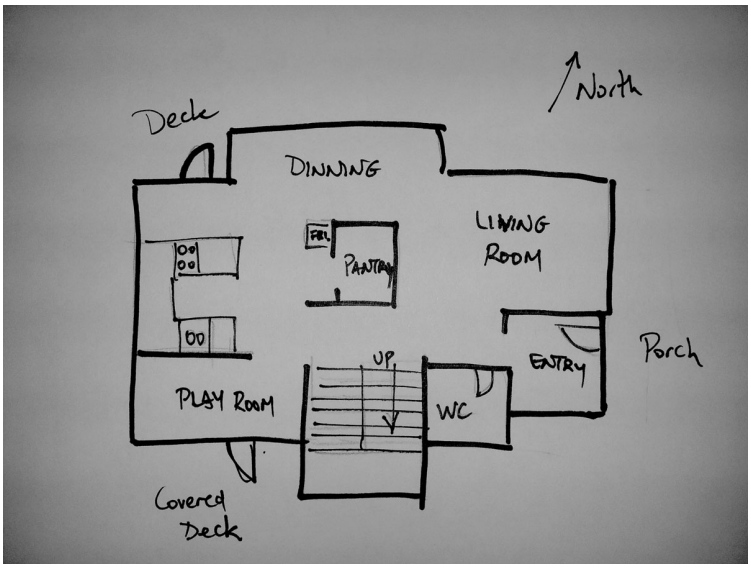
HOW TO DO A DESIGN

- We want to build a house. What to do first?

1) Decide the size of the windows.
2) Decide the color of the walls.
3) Build the roof.
4) Build the floor.



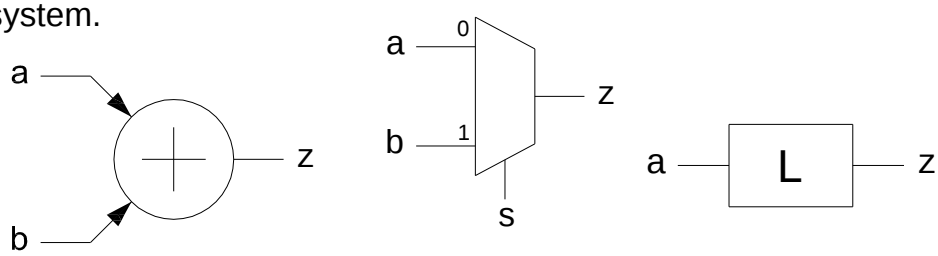
FIRST: DO A DRAFT!!!



- Architects design houses… we design circuits.

DESIGNING A HARDWARE SYSTEM

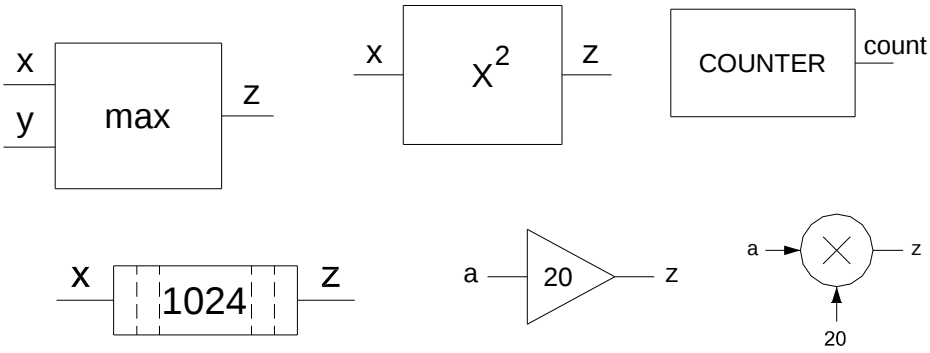
- We have already learn which are our “LEGO” pieces to build a system.



- At the system level (high in the hierarchy) we can forget the details (word length, overflow, specific values of the signals, detailed drawing of the circuit, etc.).
- Now the questions are:
  - Which circuits do we need to create our system?
  - How to interconnect these circuits / which information is sent among them?

USING BLOCKS

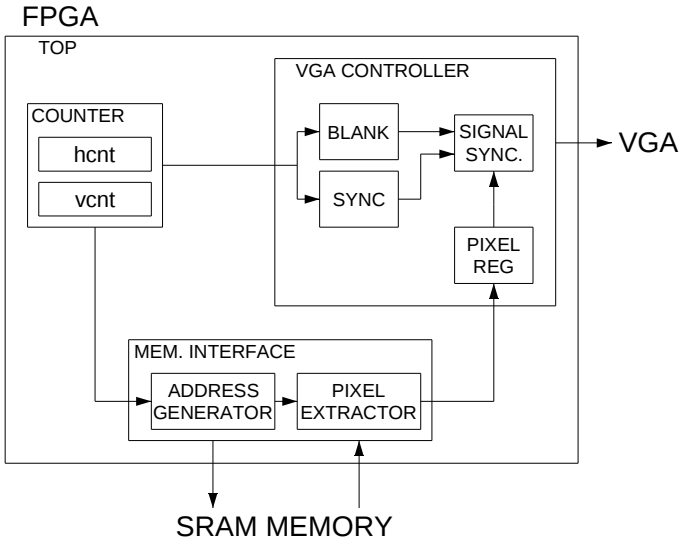
- You can use any block for your system as long as you know how to implement it internally in hardware.



- Or you can use a new block, but then you have to explain how to implement it.

EXAMPLE 1

- Do you remember how we described Lab 3 with blocks?
- You understand the system, right?



EXAMPLE 2: DOOR ENTRY SYSTEM

- How can we implement the circuit for the door of an appartment?
- Problem definition (requirements):
  - To open the door we have to press “B” followed by a code with 4 digits.
  - If the code is right, for 3 seconds the door will be open, a beepOK will sound and a green led will light.
  - If the code is wrong, for 3 seconds we will hear a beepNOK sound and a red led will light.
- First question: Which of the circuits that we have studied may we need? Why?

HOW TO DESIGN A HW SYSTEM

- In the project you have to face the design of a big hardware system.
- Some advice from previous students:
  - “I have learned that it helps a lot to draw pictures to understand the design. And then try to implement it. I do not work to do the implementation first. That is always a bad idea from now on.”
  - “You divide it into modules and solve each module. Go from high complexity to lower complexity by dividing the problem into subproblems.”
  - “Make small modules and combine them into bigger ones.”
  - “I have learnt that it is a lot easier when you plan the system. It is really important to plan which modules to use and which signals go to which module.”

HW DESIGN STRATEGIES

- How can we face the design of a large system?
- There are two main strategies:
  - Top-down design: consider the entire system and divide it into modules with specific functionality, e.g., control, memory interface and VGA controller. We can continue dividing these modules hierarchically into smaller functional blocks.
  - Bottom-up design: combine modules to create a bigger module with another functionality. For instance, we can combine an adder and a register in order to obtain an accumulator. The bottom-up design leads to new modules that can be reused in the future.
- In practice, both strategies are combined. We apply a top-down design until we reach a level in which we know how to provide the expected functionality based on a bottom-up strategy.

DESIGNING A HW SYSTEM (1)

- Step 1: General analysis of the system and partition it:
  a) Decide which are the modules of your system and their functionality.
  b) Define the connections among the modules.
  c) Extract the requirements of each module: the functionality, which inputs and outputs are needed, how data are received and sent (continuous flow, one sample every several clock cycles, etc),…
- As a result you will have a general view of the system, the modules that constitute the system and the requirements that the system and each module must fulfill.

DESIGNING A HW SYSTEM (2)

- Step 2: Design the individual modules:
  a) Decide which circuit you are going to use in order to meet the expected requirements.
  b) Apply possible simplifications of the circuit, e.g., to multiply by 4 just add two bits instead of using a multiplier.
  c) Take into account possible overflow and truncation effects.
- Step 3: Describe the circuit in VHDL:
  a) Follow the templates for the different circuits.
  b) Add comments to parts of the code that may be difficult to understand later, e.g.,

```
z <= resize(a & "000“,11) + resize(a & '0‘,11) –- z = 10a
```

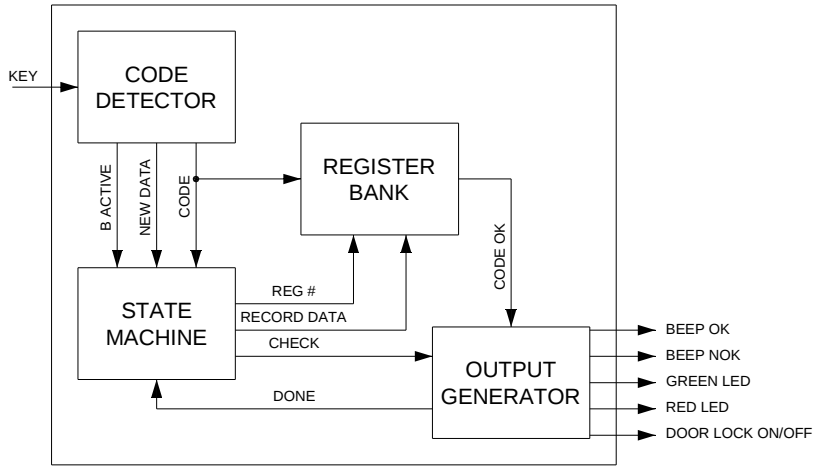
  c) Check that the syntax is correct and the code is synthesizable.

DESIGNING A HW SYSTEM (3)

- Step 4: Verify the functionality:
  a) Use simulations. Check not only the typical cases, but also and specially extreme/limit/worse cases.
  b) Configure the FPGA with the design and check if it performs as expected.
  c) You can use other devices on the development board for verification purposes, such as the LEDs.
- If everything went well and the system works as expected, congratulations!!
- Otherwise, you have to debug…

EXAMPLE: DOOR ENTRY SYSTEM

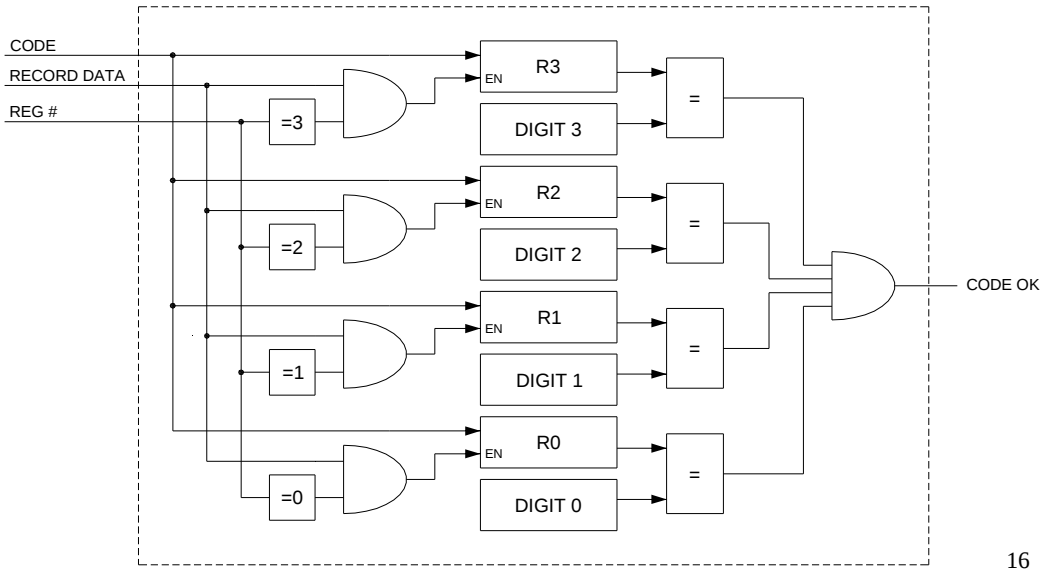
- Let´s continue with the example…



- Note that this is just one solution, but there are many other alternatives

DETAIL OF A SUBMODULE

- The register bank is used to compare the code introduced by the user and the code that opens the door.



DEBUGGING

- It does not work!!! What to do???
- DO NOT:
  - Get blocked.
  - Try one's luck: Change parts of the code to see if hopefully it works… you may be doing changes forever…
- Most of the problems in hardware design are easy to fix, but they may be difficult to find! First find the problem, then fix it.

HOW DID YOU SOLVE PROBLEMS?

This is what previous students did:

- "We solved them by a lot of trial and fail. It was often after you solve a problem you realize what was wrong, and it was so simple."
- "I had strange errors and I had to google to understand them."
- "I got a tip from the supervisor that it was the synchronization that was the problem. When I knew where the problem was, it was quite easy to fix it."
- "Sending certain values to each module to see if the module was working."
- "We carefully went through all the connections and how they were supposed to interact."
- "We just tried to isolate the problem as much as we could, and we used LEDs to see what part of the code was a fault."

DEBUGGING STRATEGIES

- To facilitate and avoid debugging:
  - **Do a good design of the circuit.** A good design requires less circuits and it is much easier to describe in VHDL.
  - Write a clear VHDL code by using the templates and adding comments. This makes the code much simple to debug.
- Once the circuit has been described in VHDL:
  - First think: “Where can the problem be?” and “which can be the reason for the problem?”
  - Think of the system as a group of blocks and check the functionality of each block independently.
  - Check that the circuit that you have designed actually calculates the function that you expect (also check overflow and timing).
  - Simulations are an excellent tool to check if the behavior of the circuit is the expected one.

MORE DEBUGGING STRATEGIES

- Check that you have described the circuit properly. Are all the connections in your description the same as those in the circuit?
- It does not work as before! Check if you are using the same files. Check if you have changed the initial conditions or if it is possible that the initial conditions have changed. Load the previous version.
- It does not work at all! Are you resetting the system or part of it? Is the clock connected? Have you compiled the last version of your system in Quartus?
- Repeat the programming steps.

MORE DEBUGGING STRATEGIES

- Use components of the board, such as LEDs, to represent the value of certain signals.
- Understand the expected behavior. For instance, if a signal is zero during a clock cycle every second, you will not see it with a LED.
- The simulation and the code do not match! Have you zoomed enough or too much? Are you only showing the last clock cycles in the simulation where the signals do not change?
- There are many red signals in the simulation. As conflicts between signals propagate, sort out the signals in ModelSim from the inputs to the output and check which is the first signal that is corrupted.

SIMULATIONS

- In big systems, you will run the same simulation several times. In this case:
  - Create a good test bench and/or script for the simulation.
  - Take your time to prepare the signals in ModelSim (add all the signals that you may need, remove unnecessary ones, organize them in the wave, add dividers, change colors, change radix, and save the layout in a .do file, etc). Remember that you can save the configuration of the WAVE in a .do file.
- This will save you a lot of time!!

CHECKLIST FOR LECTURE 8

- How to design a hardware system.
- Debugging strategies.

AT HOME

- Review the checklist for lecture 8 and check that you understand all the concepts and you know how to use them.