TSIU03, SYSTEM DESIGN

How to Describe a HW Circuit

Sometimes it is difficult for students to describe a hardware circuit. This document shows how to do it in order to present all the relevant information and in a way that it is easy for the reader to understand.

The explanation considers the system as a hierarchy of blocks and follows a top-down approach. First, the system is described at a high level. Then, each block is explained in detail.

System Level

At the system level, what is important is to provide a general idea of the system, so that the reader understands how the system works. In order to achieve this, it is important to show which are the main blocks of the system, how they process the signals, and some hardware details on the functionality of some important signals.

1) Drawing of the system:
    a. A drawing of the system is a good way to introduce it. The drawing should be complete and at the same time concise. Include only the main signals.

2) Signal processing perspective (when you explain it, refer to the figure):
    a. Describe the functionality of the entire system.
    b. Identify the blocks and describe the functionality of each block, i.e., which part of the total functionality (mathematical if it is the case) is calculated by each of the blocks (show the equations).
    c. Describe the interconnections between the block. Which relevant information is transmitted?

3) Hardware details:
    a. Explain the functionality of the main signals and how the interaction between the block happens.

4) Alternative solutions:
    a. A hardware circuit can be implemented in many different ways. If you have thought of different implementations, mention them and explain why you preferred the one that you propose.

5) Justify that it works according to the requirements:
    a. Sometimes, it is obvious that the system meets the requirements. If it is not the case, you have to justify why the system meets the requirements.

Example

In the next text, you have the system level description of a door entry system. **As an exercise**, find in the text and mark in different colors the information that is related to: 1) The introduction of the blocks, 2) The functionality of the blocks, 3) The signals that interconnect the blocks, 4) The functionality of the signals that interconnect the blocks, 5) The interactions between the blocks, 6) The interaction between the blocks and the external signals.

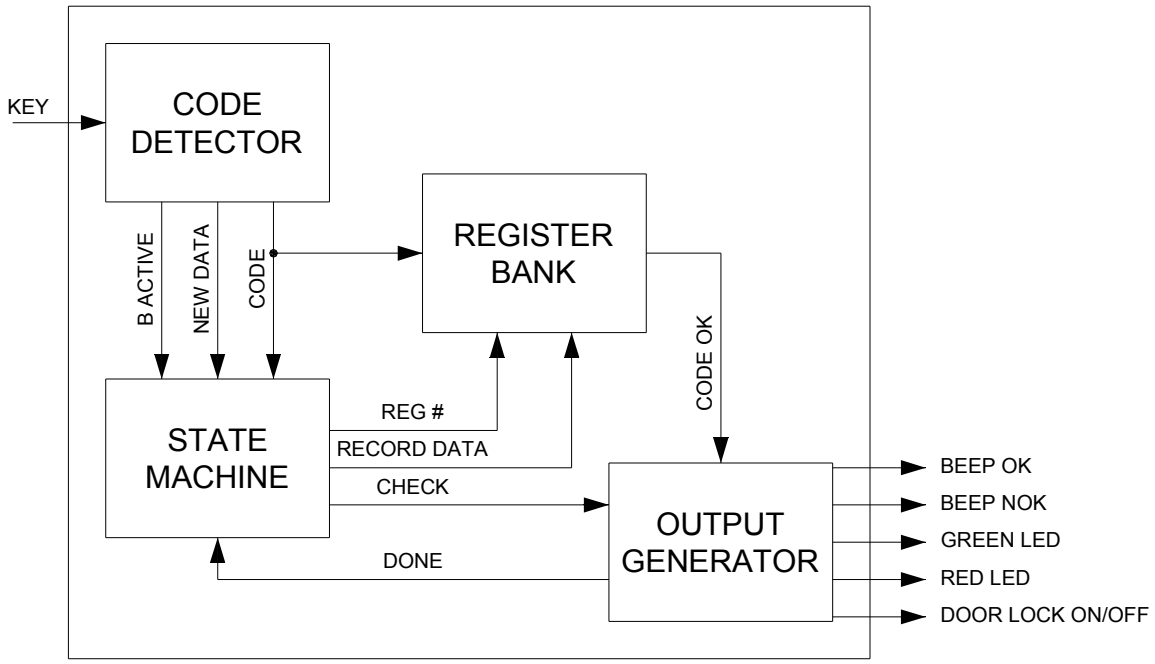

Figure 1: Door Entry System

Description of the functionality of the entire system: Figure 1 shows the circuit used for a door entry system. The system receives keys pressed on a key pad (numbers 0…9 and letter B). A user can input a code that consists of the letter B followed by four numbers. If these four numbers coincide with a secret key, the system opens the door (DOOR LOCK ON), turns on a GREEN LED and produces a BEEP OK sound for 3 seconds. If the code is wrong, the system turns on a RED LED and produces a BEEP NOK sound for 3 seconds.

Blocks, their functionality, their interconnections and HW details: The system consists of four main blocks. The CODE DETECTOR receives the keys pressed in the key pad and decodes them. When the CODE DETECTOR receives a new key, it activates a signal NEW DATA. If the pressed key is the letter B, it also activates B ACTIVE. If the key pressed is a number, that number is sent in CODE together with the NEW DATA activation.

The door entry system includes a STATE MACHINE to control the system. The STATE MACHINE is in idle state until it receives a new key. First, it waits for the letter B and then for the four numbers from the CODE DETECTOR. When a new number is received, the CODE is stored in the REGISTER BANK. This is done by activating the signal RECORD DATA and, at the same time, indicating with REG # (register number) the register in which the CODE must be stored. Each of the four pressed numbers is stored in a register, and the value of REG # is equal to their order of arrival.

The REGISTER BANK stores the CODEs for the four pressed numbers and generates a signal CODE OK when the four numbers are equal to the digits of the secret code.

Finally, the OUTPUT GENERATOR generates the output signals. The OUTPUT GENERATOR waits until it receives a pulse in the signal CHECK from the STATE MACHINE. When a CHECK is issued, the OUTPUT GENERATOR verifies if CODE OK = 1. In that case it opens the door, turns on the GREEN LED and produces the sound BEEP OK for 3 seconds. If CODE OK = 0, then the RED LED and BEEP NOK are activated for 3 seconds instead. After 3 seconds, all the outputs are turned off again and the system returns to the idle state. To do this, the OUTPUT GENERATOR sends a pulse in a signal DONE to the STATE MACHINE to indicate that it has finished and the STATE MACHINE can return to idle.

Block Level

Each block is explained independently now. At this level, we add more details that were not included at the system level with the purpose of not complicating the explanation.

If a block is complex, it can be described in the same way as is done at the system level, by dividing it into sub-blocks. If the block is not complex, you can explain it as follows.

The description of the blocks is in part similar to the description at the system level. It includes:

1) Drawing of the system:
    a. A drawing of the internal architecture of the block is a good way to introduce it. The drawing should be complete and at the same time concise. For the drawing, use the hardware circuits that we already know. Do not go to the level of gates and adders if it is not needed. For instance, we know how a counter looks like. Then, you can draw a block that says COUNTER instead of drawing an adder and a register. This provides more clarity to the figure.

2) Signal processing perspective (when you explain it, refer to the figure):
    a. Identify the inputs and outputs and which is their format or how they should look like. For instance, the CODE DETECTOR in Fig. 1 receives data in series from the input KEY. In this case, you have to describe the format/protocol in which the data is transmitted.
    b. Describe the functionality of the block (show the mathematical equations if it the circuit calculates a mathematical function).
    c. Justify how this functionality is achieved by the circuit.

3) Hardware details:
    a. Take into account the timing of the circuit (if it is complicated, you can draw a timing diagram).
    b. Take into account the word length of the signals, as well as overflow effects, if it is relevant for the description. For instance, if you have a counter you can say the number of bits that it uses and justify why you choose that number of bits.
    c. Explain other hardware details that may not be obvious for the reader.

4) Practical example:
    a. You can present an example to clarify how the circuit works.

5) Alternative solutions:
    a. A hardware circuit can be implemented in many different ways. If you have thought of different implementations, mention them and explain why you preferred the one that you propose.

6) Justify that it works according to the requirements:
    a. Sometimes, it is obvious that the system meets the requirements. If it is not the case, you have to justify why the system meets the requirements.

Example

The next circuit calculates the maximum of two numbers. **As an exercise**, find the previous information in the following description.

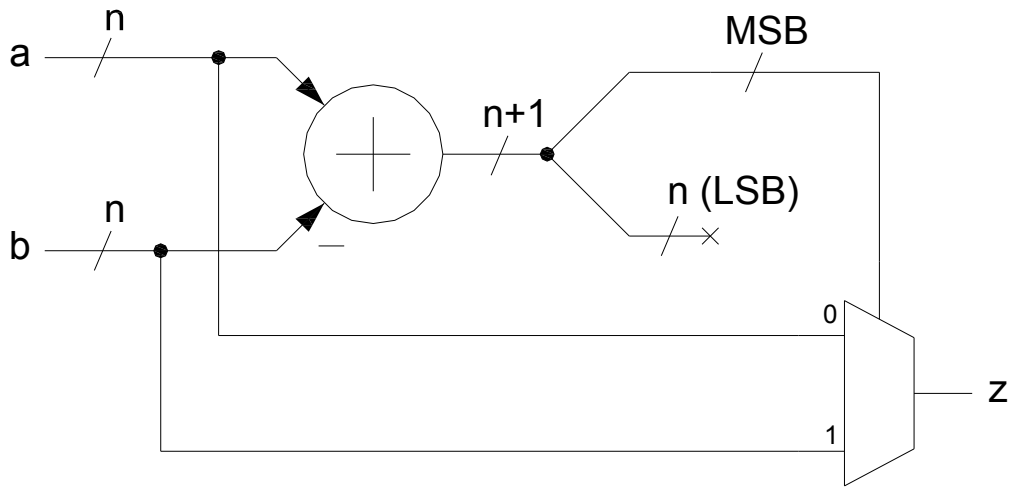

Figure 2: Circuit to calculate the maximum of two numbers, *a* and *b*.

The circuit in Fig. 2 is used to calculate the maximum of two input numbers, *a* and *b*, being the output:

$$z = \max(a, b)$$

The inputs are provided in 2's complement and their word length is *n* bits. In order to determine which number is larger, both numbers are subtracted ($a - b$). The MSB of the subtraction directly indicates which is the bigger one. If *a* is larger, the MSB will be 0, and if *b* is larger, the MSB will be 1. This bit is directly used as the input of a multiplexer to select the larger number and send it to the output *z*.

For instance, if the number of input bits is $n = 4$ and the inputs are $a = 2 \equiv 0010$ and $b = 4 \equiv 0100$, the result of the subtraction is $-2 \equiv 11110$ and the MSB is 1. Therefore, *b* is be selected, which is the

maximum of *a* and *b*. Conversely, if $a = 4 \equiv 0100$ and $b = 2 \equiv 0010$, the result of the subtraction is $2 \equiv 00010$ and the MSB is 0. Thus, *a* is selected, which is the maximum.

It is important to note that the output of the subtracter has one more bit than the input. This is important to guarantee that the MSB actually indicate which is the largest number. For instance, if $a = 7 \equiv 0111$ and $b = -8 \equiv 1000$, then the result of the subtraction will be $15 \equiv 01111$. Here the MSB is 0, which indicates that *a* is larger. However, if the result of the subtraction would have 4 bits as the inputs, then the results would be 1111 and the MSB would be 1. This would select *b* as the maximum, which is not the case.

An alternative solution to the calculation of the maximum is to compare the numbers bit by bit. If the first number is 0 in one of them and 1 in the other one, then the first one is bigger. If the MSBs of both inputs are the same, then the rest of the bits are compared starting from the second MSB. This solution is also efficient to implement in terms of hardware resources. However, this solution requires numerous comparisons. This makes it is more complicated to design and implement in VHDL. Therefore, we have preferred the proposed solution described before.

Language Used in the Description

When you describe a hardware circuit you have to include all the information and be concise at the same time. The lack of details will provide an incomplete explanation. Conversely, too much information will confuse the reader, who will not know what is important and what is not.

Look at the explanation of the system in Fig. 1. You can observe that the explanation is brief, but includes all the relevant information. This is what we are looking for.

To write a good explanation you can follow the next suggestions:

1) Relate and reflect.
2) Give the meaning / functionality of the signals. Think why they are needed: ”We use the signal to…, because…”
3) Answer to the question HOW.
4) Answer to the question WHY.
5) Present a problem -> Give the solution. // What is the circuit expected to do // How we make it happen.
6) Make an observation -> Give the reason.

If you want to explain VHDL code,

1) Identify which digital circuit is implemented in each part of the code.
2) Explain why you use the code that you have written.

<u>Example</u>

Next you have two explanations of the task in lab 2. The first explanation is incomplete because it does not relate neither reflects. It just describes the circuit. This is not enough. The second explanation is a complete explanation. Note that it relates all the circuit elements and provides reasons for them and for the value of the signals. When something is presented, it is next said, why

how, which are the implications, etc. **As an exercise**, comment the last paragraph of Description 2.

Description 1: Bad/Incomplete description

This is the clock of the system, which change every 20 ns. This is the reset signal. Here we register the inputs and here we detect the falling edge. The *detect_fall* signal is connected to the enable of the shift register. The shift register shifts the scan code, and the scan code of the shift register is sent to the 7-segment display.

Description 2: Good/Complete description

First we register the inputs. We do this because[1] in a digital circuit all the signals must be synchronized with the clock. Then, we need to[2] find when we get new valid data from PS2_DAT. This happens when there is a transition of PS2_CLK from 1 to 0. In order to[3] detect this transition we delay the signal PS2_CLK[4] one clock cycle. If we do this[5], we will have a value 1 in the delayed signal and 0 in the input one only when a transition from 1 to 0 occurs. We use a logic gate to detect this case and generate a pulse, called *detect fall*, that indicates that[6] data from PS2_DAT is valid and we can read it.

Then[7], we use a shift register[8] to store[9] the input values from PS2_DAT. We use detect fall as an enable of the shift register, so that[10] new data are only input when they are valid. Otherwise[11], the shift register would input values all the time and the information would be shifted out. The format for PS2_DAT is LSB first. Due to this, we do a shift to the right[12] in the shift register, so that[13] the LSB of the data ends in the LSB position and the MSB in the MSB position. Otherwise[14], with a shift to the left the code would be inverted in the in the register.

Data are only loaded to the register when we detect a transition of PS2_CLK from 1 to 0. And this only happens when a key from the keyboard is transmitted. If there is no transmission, PS2_CLK will be 1 and the information in the shift register will be stable. This is why we can see the number in the 7-segment display. We can also notice that the value of the 7-segment display flickers when we press a button until it gets stable. This happens because the bits in the shift register are moving, and the display reflects the intermediate values in the shift register during this time.

1: Says WHY.
2: Problem definition.
3: Explanation of the solution to the problem.
4: HOW you solve the problem.
5: Justify the solution (WHY this solution works).
6: Explanation of the functionality/meaning of the signals.
7: New hardware block: We start a new paragraph for clarity.
8: We introduce a new component.
9: We define the functionality of the new component (what it is used for).
10: Says WHY.
11: Reflection. Says WHY.
12: HOW it works.
13: Says WHY.
14: Reflection. Says WHY.