

## 3. measurement

# Digital design

### Introduction

The complexity of digital devices is increasing, with applications integrating hundreds of thousands or even millions of gates into a single IC package becoming very common. These circuits can no longer be designed using traditional paper-and-pencil methods and *computer-aided design (CAD)* is required. A later measurement - measurement 10 and 11 - will deal entirely with designing in this way. To prepare for those measurements, this measurement provides an introduction to a CAD system and the *Verilog hardware description language*.

### The purpose of the measurement

The basic aim of the measurement is to provide the knowledge necessary for the later digital measurements of the subject, to *introduce* the modern design and testing tools and methods included in them, to introduce the profession-specific knowledge about programmable logic networks, computer-aided design of digital networks, tools and methods for the verification of logic networks and functional units, which is *not included* in the previous studies *and necessary for the measurements*.

### Theoretical basis of measurement

#### References, recommended reading for preparation

In electronic form on the subject website

- [1] Introduction to using the ISE system
- [2] Verilog brochure slides
- [3] Developer panel documentation

### Tasks to prepare

*As preparation for the measurement at home, do the following independently!*

1. Review what you learned in Digital Design [2] !
2. Read the *recommended literature to help you prepare!*
3. Read and think about the *Measurement exercises!*

## Measurement tasks

This laboratory exercise is a "workshop" type of guided measurement exercise, and thus differs in its "delivery" from later laboratory exercises. Accordingly, this "measurement guide" is also in a different form.

### 1. Xilinx ISE introduction (guided measurement)

Follow **the tasks described in the ISE introduction as** you go along with the measurer. This document is available on the subject website.

**Before programming the FPGA IC, it is necessary to check that the routing of the outputs is correct.** This is because if a leg is programmed as an output to which an I/O device on the meter panel is also wired, **the meter panel may fail.**

### 2. ADDITIONAL TASK: Extend the functionality

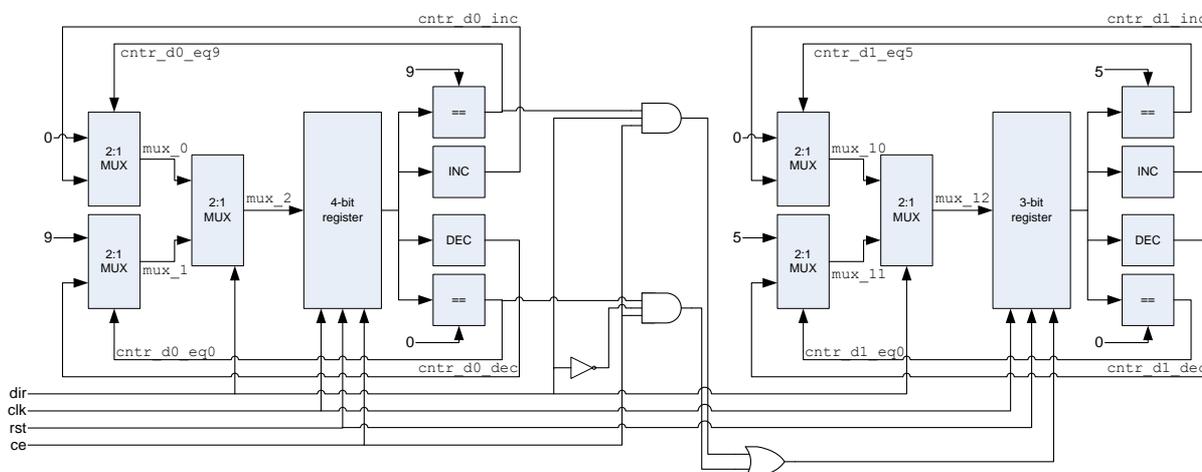
Modify the design to implement a two-digit BCD (binary coded decimal) seconds counter. That is: the lower digit can take values between 0...9, while the upper digit can take values between 0...5.

Consider the operation of the upper digit, which is quite similar to the lower digit (it is also an enableable up/down counter). The lower digit counts in all cases when the external enable signal is active. In contrast, the upper digit only needs to change when the lower digit has reached its final value (which is 9 for counting up and 0 for counting down) and the external enable signal is active. That is, the upper digit enable signal is '1' if:

- is counting up ( $dir='1'$ ), the lower digit is 9 ( $cntr\_d0\_eq9='1'$ ) and the enable signal is active ( $ce='1'$ ) OR
- count down ( $dir='0'$ ), the lower digit is 0 ( $cntr\_d0\_eq0='1'$ ) and the enable signal is active ( $ce='1'$ )

Another difference compared to the lower digit is that the numerator's final value is 5 instead of 9, which can be represented in 3 bits.

After the above considerations, the block diagram is as follows.



Steps to change the counter:

- Declare the signals needed for the second counter!

- Give the functional description!
- Increase the size of the output variable ( $q$ ) of the *count\_sec* module from 4 to 7 bits. Fold the lower 4 bits with the lower digit and the upper 3 bits with the upper digit.
- Check the syntactical correctness of the modified Verilog code!
- Use simulation to check the correct operation! Since the ports of the *count\_sec* module have changed (increased  $q$  output signal width), this change must be made in the testbench file.

Modify the top-level module:

- Modify the top-level module (*wpbvtop1*) to meet the requirements of the new counter (7-bit value to display).
- Modify the pinout! Display the 3 bits of the upper digit on LED6, LED5, LED4 of the panel. In other words, connect the  $q[6:0]$  output port  $q[6:4]$  bits to the corresponding pins!
- Implement the modified design, then download the configuration file to the FPGA and verify the operation.

### 3. Test questions

1. Give the module declaration of the Verilog module with the following ports. Inputs: scalar clk, scalar en; outputs: 8-bit dout.
2. Declare a 4-bit data signal of type wire, and continuously give it the hexadecimal value 0x42 in binary.
3. Declare three variables of type 8-bit signed wire (res, op\_a, op\_b). Drive res with the sum of op\_a and op\_b.
4. Declare a 16-bit and an 8-bit variable of type wire (d16, d8). For the 16-bit variable, pass the extended value of the sign of the 8-bit variable (the 8-bit variable can contain two complement values).
5. Declare one 8-bit variable of type reg (res) and two 8-bit variables of type wire (op\_a, op\_b). Specify the Verilog code that implements combinational logic where res is the sum of op\_a and op\_b.
6. Give the Verilog code for a 4:1 multiplexer. Input signals: selection signal (sel), data inputs (in0, in1, in2, in3); output: r. Declare the input signals as wires (data are 1 bit) and the output as described.
7. Give the Verilog code of an the 1-bit D FF with an asynchronous reset (rst), set (set), enable (en). Declare both output and input signals (the latter as 1-bit wires).
8. Give the Verilog code of a testbench that generates a 10 MHz clock. The name of the clock signal should be clk, declare it.
9. Give the Verilog code for an 8-bit, resettable, up-counter module. Inputs: clock (clk), reset (rst); output: counter value (cntr).
10. Give the Verilog code of an 8-bit left shift shift-register module. Inputs: clock signal (clk), serial data input (din); output: shift register value (shr).