

Exercise 10.

Implementation and analysis of sequential networks

Requirements

- Digital design knowledge
- Verilog knowledge
- Basic Xilin ISE knowledge (Exercise 2.)
- Xilinx ChipScope (Logic analyzers and Xilinx ChipScope documentation)

Preparation for the measurement

This measurement is based on the earlier exercise “Digital devices basics”. You should scrutinize what you have learned before.

In this exercise, you will improve your previous verilog design and analyze the new hardware by means of a logic analyzer synthesized in the FPGA. Read the paper “Logic Analyzers and ChipScope” (Available on the homepage)!

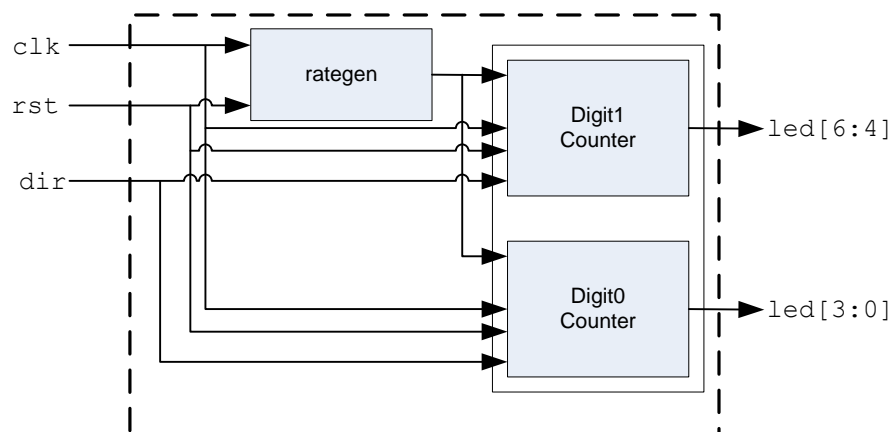
Check the measurement tasks and the test questions!

Measurement tasks

A piece of advice: if you want to analyze your design, the configuration and the connections of the logic analyzer must be setup based on *deliberate considerations made in advance*. All the posterior changes need extra time. The synthesis of the ChipScope needs quite long time, so double check every setting before starting it, and pick the synthesis time for the documentation of the previous measurement tasks!

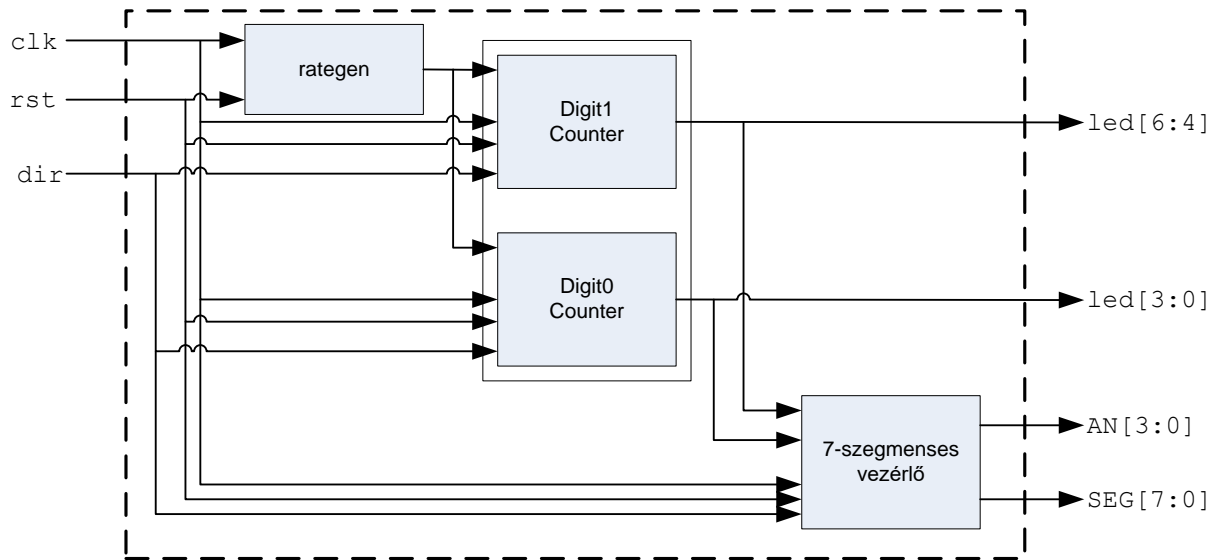
1. Improvement of the design: Counting on a 7 segment display

During a previous exercise, you have described a 2 digit BCD second counter.

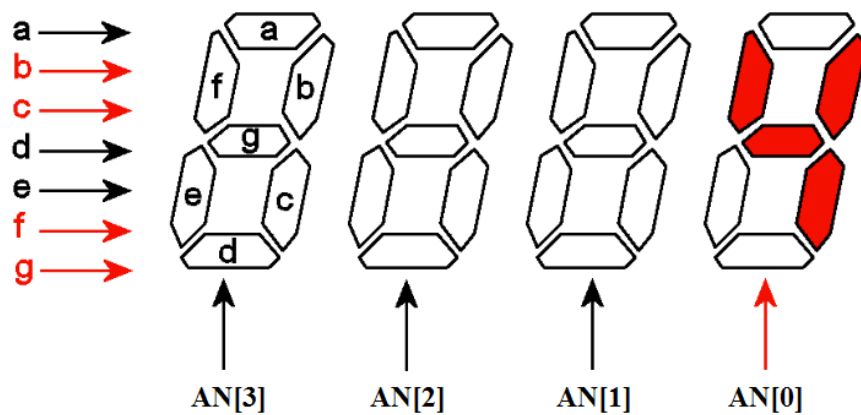


Laboratory exercises 10.

Now, you should add a 7-segment display controller as a new module.

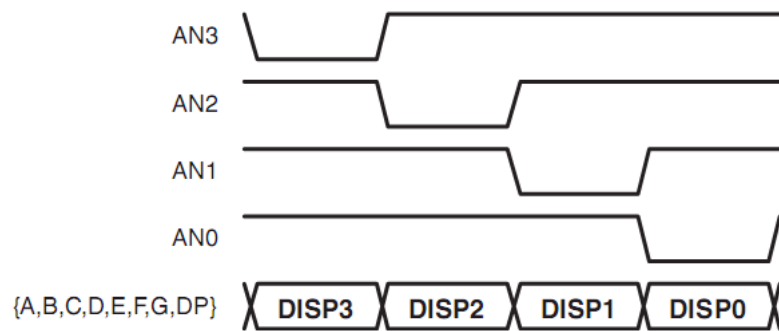


The 7-segment display mounted on the development board can be controlled with time-division multiplexing. The a,-g, segment signals off all the 4 digits are connected to common FPGA pins (i.e. the 4 a, segment is connected to the same pin, the b, segments to another one etc.). A certain digit is selected by its low-active anode (AN) signal.



The segment signals represent always the value for the currently selected *single* digit, but the selection changes so quickly, that the result is the same, as all the digits would be controlled separately. The waveform of this time-division multiplexing is depicted below.

Exercise 10.



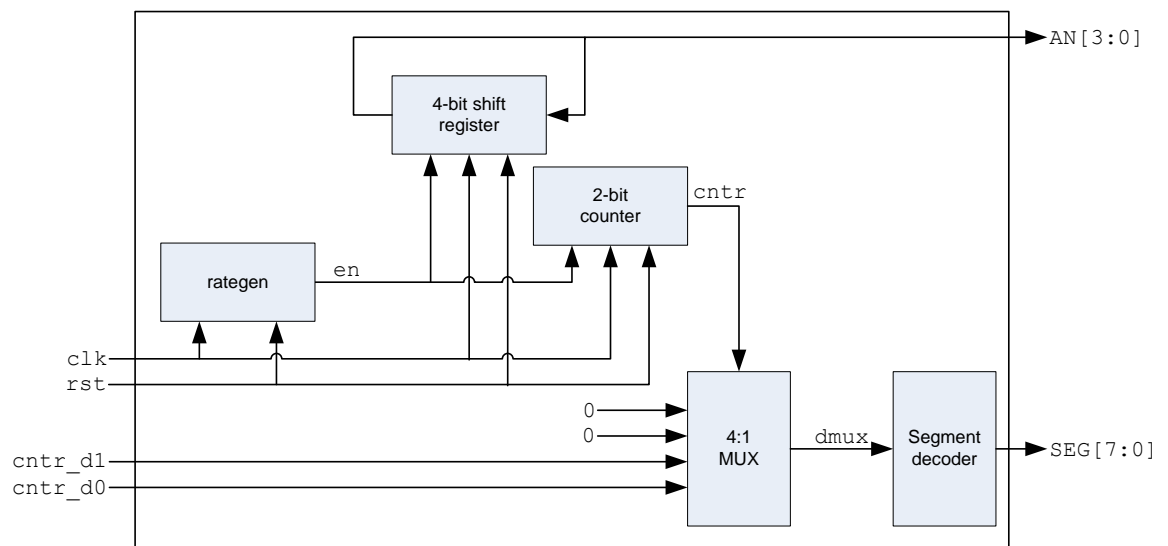
To give the digits enough time to light up but avoid blinking, **the AN frequency should be in the order of kHz**. This must be generated with a rate generator having the same structure as the one used before in the second counter module.

The easiest way to generate the AN signal is to use a looped back shift register. (The rate generator should be connected to its enable input.). The initial value after reset can be „1110”, and the whole cyclic sequence with left shift is 1110 → 1101 → 1011 → 0111 → 1110...

The BCD value to display for the currently selected digit can be selected by a 4:1 multiplexer. The appropriate multiplexer input can be addressed with a 2-bit counter (counting synchronously with the AN shifter).

The 4 bit BCD value must be “translated” to the 8 bit (7 segments, decimal points) SEG signal. This segment decoder is already coded in your source skeleton file.

The scheme for the 4-digit BCD to 7-segment display decoder is depicted below. In our design, we will use only the lower digits to display the second-counter. The upper two digits should constantly show 0.



1.1. Project creation

Download the skeleton files from the web and create a new ISE project using them. You find the following files in the ZIP:

- wpbevtop1: The top module for the project instantiating all the sub-modules except the 7-segment counter.

- *rategen*: 1 Hz rate generator for second counter..
- *count_sec*: 2-digit second counter. (There is nothing to modify)
- *cntrl_7seg*: skeleton for the 7-segment display controller.
- *counter_pins.ucf*: pin-out for the FPGA. (There is nothing to modify)

1.2. Implement the 7-segment display controller

Use the skeleton file and the schematic and description given above! Do not create any extra module! (The single given skeleton file can contain multiple always blocks and assignments.) The provided UCF file also contains all necessary pin associations, there is no need to modify it.

The *cntrl_7seg* module has following ports:

- *clk*: 16 MHz system clock
- *rst*: external reset
- *din0*, *din1*: 4 bit BCD data inputs, coding the values to display on the lower digits.
- *AN*: 4 bit anode output to the 7-segment-display.
- *SEG*: 8-bit cathode output to the 7-segment-display.

After coding the module, check syntax!

1.3. Instantiation

Instantiate the 7-segment controller in the top-level module! Generate the bit-file, download it, and evaluate whether the new design works fine!

2. “Knight Rider” effect

Create the Verilog description of the specified below.

The system implements a bi-directional chaser ("Knight Rider effect") using 8 LEDs on the development panel. The initial state is "0000001", after which the light LED moves to the left, approximately every second. When the final value of "1000000" is reached, the light changes direction and moves to the right until the state "0000001" is reached, at which point it starts to move to the left again.

Exercise 10.

The per second enable signal is provided by the existing ratgen module, so the block to be designed will have an enabling input.

Select one of the following solutions and implement it.

2.1 Implementation with state machine (1)

The system to be implemented has $2^8 - 2$ state machines: the values "00000001" and "10000000" occur only once, the others occur twice. A possible solution is to encode the states as the same as the output value, so that the state register is directly the output of the state machine.

In Verilog the description can be easily solved with a larger case structure, see Verilog PPT.

2.2 Implementation with state machine (2)

You may notice that the state transitions in the state machine are very simple, you always go from a given state to a given next state unconditionally. It is actually a counter that counts in the range 0...13, the counter itself is the state register. And the output logic is a decoder, which produces the bit pattern to be displayed on the LEDs (e.g. "0000" → "00000001") from the given binary value.

2.3 Implementation functional elements

You can implement the design using functional elements, if you notice that it is really a bidirectional shift register. In addition to the shift register, we will need a direction register to define the direction of the shifting. Take care when should you modify the direction register: when the end value of a given direction is reached, the shift must already be in the other direction, so the direction modification must be done in the previous state.

Test questions

1. Specify the waveform for the control (AN and DIG signals) of a time-division multiplexed, 4-digit, 7-segment display. How often should the AN signal be changed? What is the effect if this signal changes very slowly (~1 sec)?
2. Create 3-input and/or gates by instantiating 2-input and/or gates (and2/or2). Ports of the instantiated modules are: i0, i1 inputs, o output.
3. Write the Verilog code of a 4-bit rotating shift register. The initial value of the register should be 4'b1000, shifting to the left. The inputs of the module are: clock (clk), reset (rst); the output is the current state of the shift register (shr).
4. Write the Verilog code of a 4-bit rotating shift register. The initial value of the register should be 4'b1000, shifting to the right. The inputs to the module are clock (clk), reset (rst); the output is the current state of the shift register (shr).
5. Write the Verilog source code of an enable signal generator module (ratgen) that generates a sequence of enable pulses of one clock pulse length at a frequency of 800 kHz

Laboratory exercises 10.

using a 16 MHz system clock. The inputs of the module are: clock (clk), reset (rst); output: enable signal (en).

6. Write the Verilog source code of an enable signal generator module (rategen) that generates a series of enable pulses of 1 kHz frequency and 1 clock length using a 50 MHz system clock. Module inputs: clock (clk), reset (rst); output: enable signal (en).
7. Write the Verilog code of a 1 digit up-counter BCD counter. The reset state of the counter shall be 0. The inputs of the module are: clock (clk), reset (rst); the output is the current state of the counter (cnt).
8. Write the Verilog code of a 1 digit down-counter BCD counter. The reset state of the counter shall be 0. The inputs of the module are: clock (clk), reset (rst); the output is the current state of the counter (cnt).
9. Write the Verilog Test Fixture code fragment (including signal declarations) that provides a 10 MHz clock signal for the module under test.
10. The following Verilog code is given. Complete the timing diagram.

```
reg [7:0] cnt=0;
reg en; reg c_dl;
always @ (posedge clk)
begin
    cnt <= cnt + 1;
    c_dl <= cnt[7];
    en <= ~cnt[7] & c_dl;
end

reg [3:0] shr=4'b1110;
always @ (posedge clk)
    shr <= {shr[0], shr[3:1]};
```

