Exercise 10.

Implementation and analysis of sequential networks

Required knowledge

- Digital design knowledge
- Verilog knowledge
- Basic Xilin ISE knowledge (Exercise 3.)
- 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)!

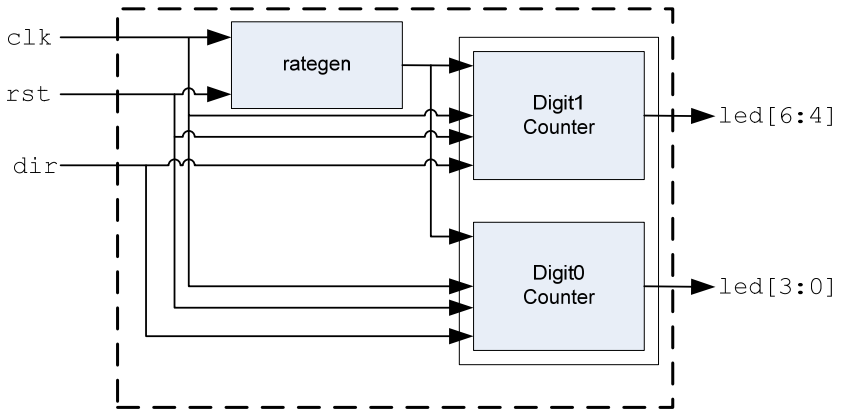
Check the measurement tasks and the test questions!

Measurement tasks

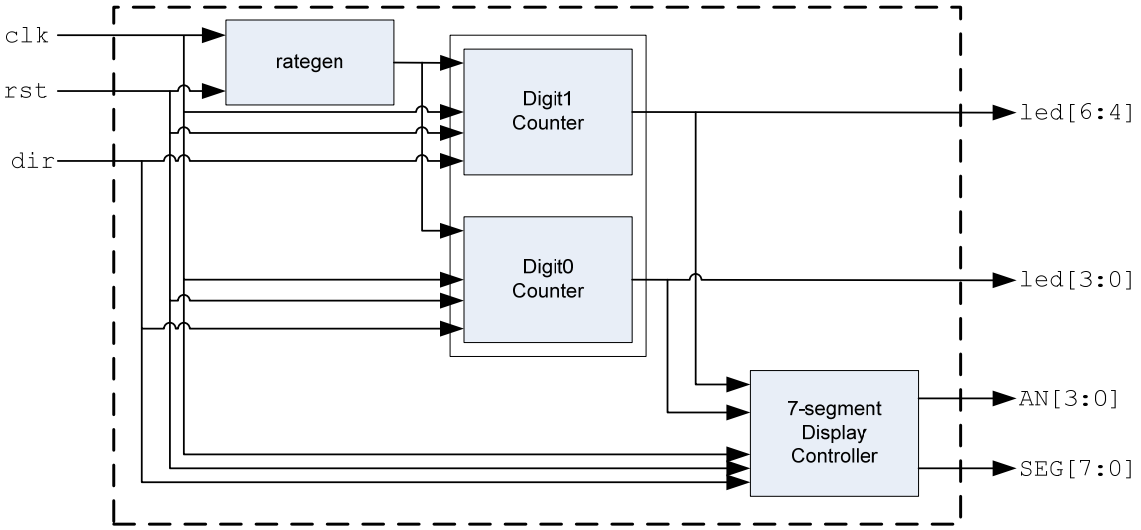
A piece of advice: if you want to analyze your design, the configuration and the connections of the logic analyzer must be setup based on *deliberate considerations made in advance*. All the posterior changes need extra time. The synthesis of the ChipScope needs quite long time, so double check every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

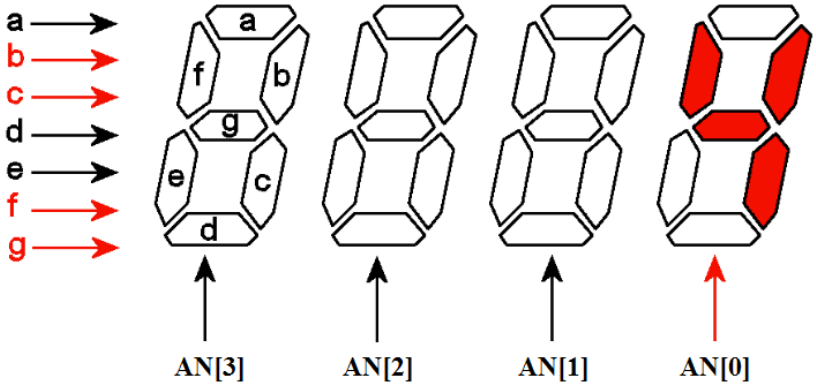
During a previous exercise, you have described a 2 digit BCD second counter.



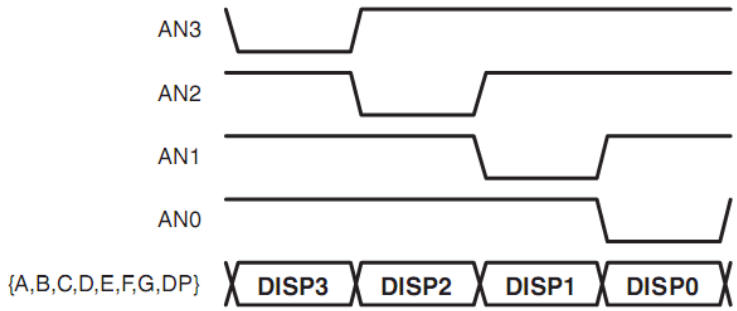
Now, you should add a 7-segment display controller as a new module.



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.



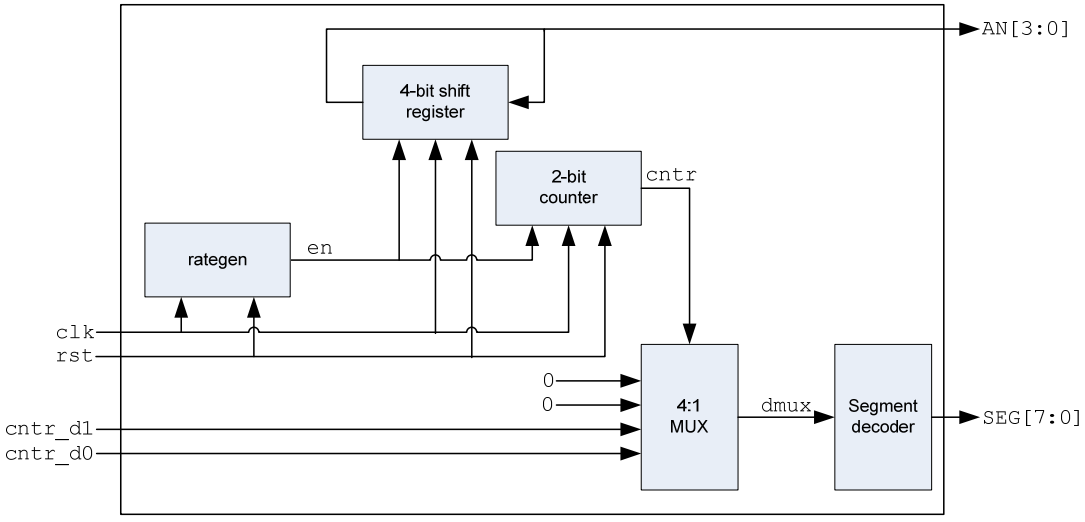
To give the digits enough time to light up but avoid blinking, **the AN frequency should be in the order of kHz.** This must be generated with a *rate generator* having the same structure as the one used before in the second counter module.

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. 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 chatode output to the 7-segment-display.

After coding the module, check syntax!

1.3. 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. Using logic analyzer

To easily learn the usage of logic analyzer, modify the original design: *edit the rategen module and speed-up the counter*. (So, the whole counting cycle can be analyzed with a small state memory.)

Using an external analyzer, one must guarantee access to the appropriate digital signals and connect them to the PODs. (If you want to evaluate internal signals, they previously must be connected to FPGA pins, thus the modification of the Verilog code is needed.) In this lab, the analyzer itself is implemented in the FPGA, so only a JTAG connection is needed to read the state memory.

To create the appropriate ChipScope configuration, we should overview the analysis task. All the design considerations are given below, but you should understand them! Read this part, synthesize the ChipScope, download the new bit-file and do the tasks!

2.1. Simple trigger condition: the counter value is 15h, dir=1.

Dir and the whole counter must be connected to a trigger port (TRIG0)

2.2. One-level trigger condition: Direction changes while the lower-digit value is in range 5h : 9h.

Direction change detection needs a TRIG0 match unit with the ability of edge-triggering.

Don’t care-s are not allowed, if a range is checked in a match unit. Practically, for the range check the lower digits of the counter should be connected to a separated trigger port (TRIG1) with in-range match unit.

2.3. 3 step sequential trigger condition: the lower digit is 4, then direction changes, then the lower digit is 4 again.

2.4. Check, whether the reset button or the direction-switch is bouncing!

Trigger on rst/dir change? Is any bouncing time detectable this way?

Demonstrate the bouncing with a waveform!

What is the problem with bouncing? How can you de-bounce a button?

Knowing the analyzer tasks and using the paper about the analyzer, add a ChipScope definition file (cs_cntr.cdc) to the project! Use the following settings! A feladatok

- Trigger Parameters:

  TRIG0: 9 bit, #1 Basic w/edges MatchUnit,

  TRIG1: 4 bit, #1 Range MatchUnit

  Enable Trigger Sequencer (Max. 3 level)

  Enable Storage Qualification

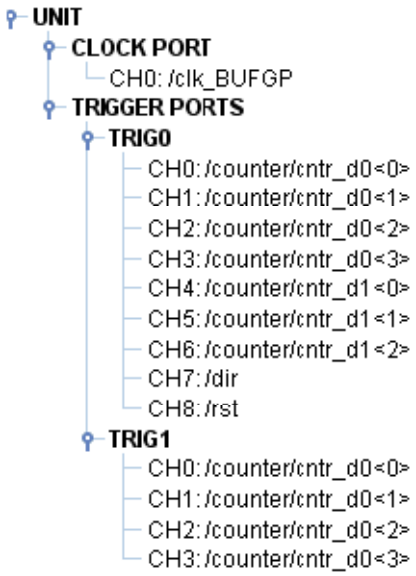
- Capture Parameters:

  Sample on Rising Clock Edge

  Data Same As Trigger, Include TRIG0

  Data Depth: 1024 Samples

- Net Connections: as depicted below:



Double check the settings. Finalize the log about the previous tasks during the synthesis! After downloading the new bit file, set up the trigger conditions described before!

Test questions

1. What is the difference between the State and Timing Modes of the logic analyzer? Can ChipScope be used in both modes?
2. What is the purpose of trigger signals and trigger conditions of logic analyzers?
3. The states of the sequential network have to be determined. Which mode has to be chosen and what is the source of sampling signal in this case?

4. The propagation time (Td) of a network has to be determined with the logic analyzer. Which mode has to be chosen and what is the source and frequency of the sampling signal in this case?
5. A periodic square wave is analyzed with a logic analyzer. The frequency of the signal is between 5 kHz - 10 kHz and the duty cycle is between 20% - 50%. What is the minimum value of the sampling frequency if the time span of L and H values have to be derived with at most 5% uncertainty? Assume that only 1 period is captured with the analyzer and the uncertainty of the internal clock is negligible.
6. Why is the usage of multiple clock sources disallowed in a Verilog design?