MÜHENDİSLİK FAKÜLTESİ
FACULTY OF ENGINEERING
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

DUMLUPINAR BULVARI 06800
ÇANKAYA ANKARA/TURKEY
T: +90 312 210 23 02
F: +90 312 210 23 04
ee@metu.edu.tr
www.eee.metu.edu.tr

EXPERIMENT 5. INTRODUCTION TO VERILOG

I. Introduction

I.I Objectives

In this experiment, you will get familiar with coding with Verilog HDL by designing and implementing basic modules that perform specific set of instructions. Therefore, before coming to lab you should learn and practice about behavioral and structural design with Verilog.

II. Preliminary Work

1) Clock enable signals are utilized to enable clocks periodically if the desired hardware operation is slower than the clock signals. Design and implement a clock enabler module similar to the tutorial that enables a given master clock to once in 2, 4 or 8 cycles. The module should have two inputs “clk” and “enabler” and one output “clk_enable”. You should be able to adjust the clock enable factor by changing the value of enabler pin. Block diagram of the module and set of instructions are given below for your convenience.

Valueofthedivisionpin	Result
00	No change in the frequency (“dividedclk” should have the same frequency with input “clk”)
01	Input clock should be enabled once in 2 clock signal
10	Input clock should be enabled once in 4 clock signal
11	Input clock should be enabled once in 8 clock signal

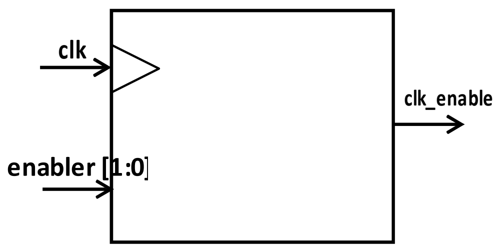


Fig. 1: Block diagram and instruction sets of the clock divider module.

Include your state diagram, Verilog code, simulation result and RTL schematic of the clock divider module in your preliminary work. You are free to choice input clock frequency in your simulations. Note that you can obtain RTL schematic of a design from **Tools > Netlist Viewers > RTL Viewer**

Note: For both part 2 and part 3 of the preliminary work, you will use hierarchical design. Therefore, if you feel unfamiliar with this topic, please learn about hierarchical design in Verilog before continue on part 2 and 3.

2) Design and implement a Verilog module that performs several operations on a given 4 bit data according to instructions taken from a serial port. The top module (consists of three submodules) should have three inputs "clk", "data" and serial instruction pin "ins" and one serial output pin "serial_out". The block diagram and instruction set of the top module are given below.

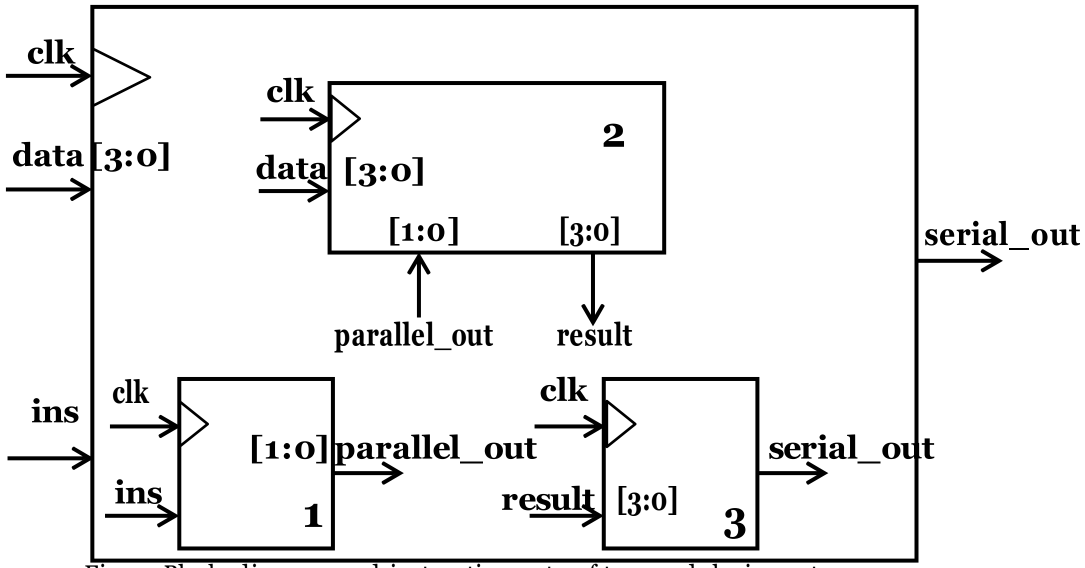


Fig. 2: Block diagram and instruction sets of top module in part 2.

Valuesread from ins pin	Value of the serial_out pin
00	Number of zeros in the given data (e.g. if data is "0010" the value of serial_out pin should be 0,0,1,1 (represents "3" zeros) respectively.)
01	Right shifted version of the data input by 1 bit
10	1's complement of the data input
11	Greatest number in a given set of data

As it is seen from the figure 2, you should design three submodules that are serial to parallel converter module (SPM) (1), logic module (LM) (2) and parallel to serial converter module (PSM) (3). SPM should convert serial instructions to parallel and provide them to LM module. LM should take the data value and according to instruction code, it will perform a set of specific operations defined above. Finally, the result should be sent out from the serial output pin by using PSM to serialize it.

Include state diagrams, verilog codes, simulation results and RTL schematics of all submodules and the top module you have designed in your preliminary work. Your simulation results should show that all of the instructions work simultaneously for a given set of data. You can use any clock frequency in your simulations.

3) Design and implement a Verilog module that continuously generates random numbers between 0 and 9 and displays them on seven segment display of the FPGA. For this purpose, you should design and integrate two submodules that are random number generator module (RNGM) (1) and seven segment display controller module (SSDCM) (2). The top module should have one input "clk" and one output "display". RNGM should generate random numbers between 0 and 9 and provide these numbers to SSDCM. SSDCM should convert these numbers to a data format so seven segment display can properly display them. The block diagram of top module is given below.

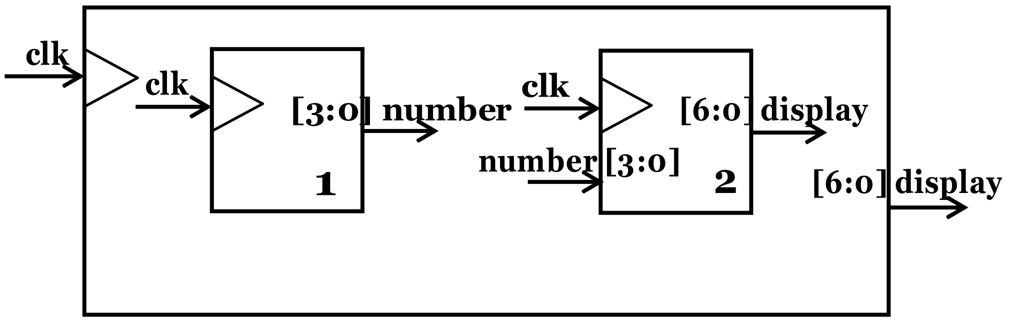
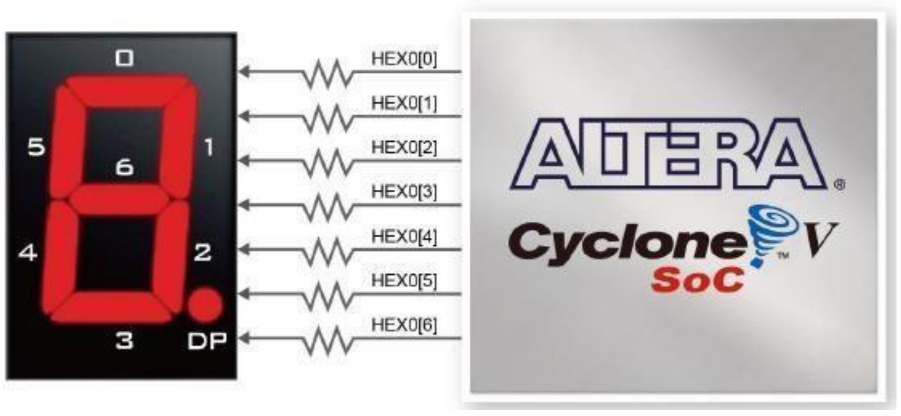


Fig. 3: Block diagram of top module in part 3.

You are not allowed to use "$random" function of the Verilog (This function is not synthesizable and has no hardware implementation). Instead, you should use your own algorithm to generate random numbers. Although the generated numbers will not be truly random but *pseudo* random you can still search for interesting algorithms on Internet and implement them in Verilog. Any approaches are welcomed however; you should clearly explain your algorithm. After you have generated a random number with RNGM, it should be properly displayed via "display" pin by using SSDCM. Since we have one digit number (0-9), using first digit of the seven segment display will be sufficient. More detailed information about seven segment display is provided below.



Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
HEX0[0]	PIN_AE26	Seven Segment Digit 0[0]	3.3V
HEX0[1]	PIN_AE27	Seven Segment Digit 0[1]	3.3V
HEX0[2]	PIN_AE28	Seven Segment Digit 0[2]	3.3V
HEX0[3]	PIN_AG27	Seven Segment Digit 0[3]	3.3V
HEX0[4]	PIN_AF28	Seven Segment Digit 0[4]	3.3V
HEX0[5]	PIN_AG28	Seven Segment Digit 0[5]	3.3V
HEX0[6]	PIN_AH28	Seven Segment Digit 0[6]	3.3V

Fig. 4: Connections between seven segment display and pin assignment for the first seven segment digit on FPGA board.

Note that seven segment displays on the FPGA board are active low. Hence, for example; if you want to display the number "1" on the seven segment display, onl part 1 and 2 should be ON and therefore only 2nd and 3rd bits of the "display" pin should be LOW (display pin should be equal to "1111001").

Include state diagrams, Verilog codes, simulation results and RTL schematics of all submodules and the top module you have designed in your preliminary work. You are free to use any clock frequency for the simulation purposes. However, in practice, it will be better to design your module to be worked at a frequency (e.g. 1 Hz) where different random numbers displayed on seven segment display can be recognized with eye. Please note that the clock frequency of the FPGA we use is 50 MHz. Therefore, you may prefer to design a clock enabler that enables the clock once in 1 second.