# Homework 1: Combinational Logic Circuits & Binary Arithmetic

**Due date: 10.04.2022 23:59**

*Res. Asst. Büşranur Bülbül Demir*
*bulbulb17@itu.edu.tr*

## Introduction

The aim in this experiment is to implement combinational logic circuits and Full adder using the Verilog. In this experiment, you are allowed to use '&' (Bitwise AND), '|' (Bitwise OR), '∼' (Bitwise NOT), and '{}' (Concatenate) operations. Other operations such as '^ ', '+', '-', '*', '/', '<<', and '>>' are forbidden. This experiment is an introduction to learn how to design and use a module. More complex experiments will be later experiments, so you can not use `always`, `parametric modules`, `switch case`, and `if else` etc. in this experiment. **You must simulate all modules you design.**

## Preliminary (7 pts)

1. For the function $F_1$ given in Equation 1, apply the following operations:
   a) Find its prime implicants using Karnaugh diagram.
   b) Find its prime implicants using Quine-McCluskey method.
   c) Create prime implicant chart and find the expression with the minimum cost with 2 units of cost for each variable and 1 unit of cost for complement of a variable.
   d) Design and draw the lowest cost expression using NOT, AND, and OR gates
   e) Design and draw the lowest cost expression using only NAND gates.
   f) Design and draw the lowest cost expression using a single 8:1 Multiplexer, AND, OR and NOT gates.

$$F_1(a,b,c,d) = \cup_1(0,2,6,7,8,10,11,15) + \cup_\varphi(4) \tag{1}$$

2. Design and draw the functions $F_2$ and $F_3$ given in Equations Equation 2 and Equation 3 using ONE single 3:8 decoder, 2-input OR gates (x′ represents the complement of x).

$$F_2(a, b, c) = a'bc + ab'c \tag{2}$$

$$F_3(a, b, c) = abc' + ab \tag{3}$$

3. 
   - Recall signed and unsigned addition for binary numbers in 2's complement notation.
   - Recall signed and unsigned subtraction for binary numbers in 2's complement notation.
   - Recall signed and unsigned subtraction for binary numbers in 2's complement notation.

## Experiment

## Part 1 (10 pts)

In this part, you are requested to implement AND, OR, NOT, XOR, NAND, 8:1 Multiplexer and 3:8 Decoder modules which you will use in the following experiments. You should write these "gates" as modules and use them later in your implementations. In other words, for example, when you need to apply an AND operation to two input wires, you should use the modules you have written before, rather than using operators like "&" or "|" directly. Following the implementation, please run simulations for various input combinations to validate your designs.

## Part 2 (7 pts)

Please implement the circuit that you have designed in Preliminary 1.d. section using NOT, AND, and OR modules in Verilog.

## Part 3 (7 pts)

Please implement the circuit that you have designed in Preliminary 1.e. section using only NAND modules in Verilog.

## Part 4 (7 pts)

Please implement the circuit that you have designed in Preliminary 1.f. section using a 8:1 multiplexer, AND, OR and NOT gates in Verilog.

## Part 5 (7 pts)

Please implement the circuit that you have designed in Preliminary 2 section using a 3:8 decoder and OR gates in Verilog.

## Part 6 (7 pts)

In this part, you should implement 1-Bit Half Adder module by using AND, OR, NOT, XOR modules which you designed in the first part. Then, you should simulate it for each different combination of input.

## Part 7 (7 pts)

In this part, you should implement 1-Bit Full Adder by using half adder and OR modules which you designed in the previous parts. Then, you should simulate it for each different combination of input.

## Part 8 (7 pts)

In this part, you should implement a 4-Bit Full Adder by using 1-Bit Full Adder modules which you designed in the previous part. Then, you should simulate it for '8+1', '2+7', '4+5', '11+10', '14+5', '15+9', '6+3', and '8+12' operations.

## Part 9 (7 pts)

In this part, you should implement an 8-Bit Full Adder by using 1-Bit Full Adder modules which you designed in the previous part. Then, you should simulate it for '29+5', '51+92', '17+28', '191+2', '200+95', '49+25', '78+255', and '43+59' operations.

## Part 10 (7 pts)

In this part, you should implement a 16-Bit Adder-Subtractor by using 8-Bit Full Adder and XOR modules which you designed in the previous part. Then, you should simulate it for '23+3', '21+75', '16800+16900', '69834+66500', '325+97', '44+190', '463+241', and '86+572' operations

## Part 11 (10 pts)

In this part, you should implement a module which calculates the B - 2A by using 16-Bit Adder-Subtractor, Adder, NOT, XOR, AND, and OR modules. Then, you should simulate it for 'A=32, B=7', 'A=21, B= 85', 'A=16, B=36', 'A=256, B=5', 'A=200,B=95', 'A=45, B=135', 'A=36, B=255', and 'A=25, B=65' inputs.

## Report (10 pts)

You should show your work of Preliminary study on the report in detail. You can use any tool for creating tables and circuit designs. You may attach them to the report as figures by properly referencing them in the text. Your report should also contain the results of your simulations for each module. If your implementations are not fully correct, discuss what the source of the errors might be in your report.

## Submission

- Please do not send any document via e-mail to one of the assistants.
- Your reports must be written with Latex format. Latex report template is available on Ninova. You can use any Latex editor whichever you want. If you upload your report without Latex file, you directly get 0 as your report grade. You should upload both .pdf and .tex files of your report.
- You should submit 2 separate ".v" files for your Verilog codes. One of them should contain the modules and other one should contain the simulation codes.
- It will be sufficient for one person from each group to upload the homework.
- Be aware of the deadline. Late submissions are not accepted.
- Please do not hesitate to contact me (bulbulb17@itu.edu.tr ) for any question.