Digital Logic Design

Final project report

PREPARED FOR

Dr. Mahdiani

Mr. Mosayebi

PREPARED BY

Mohsen Karbalaei Amini

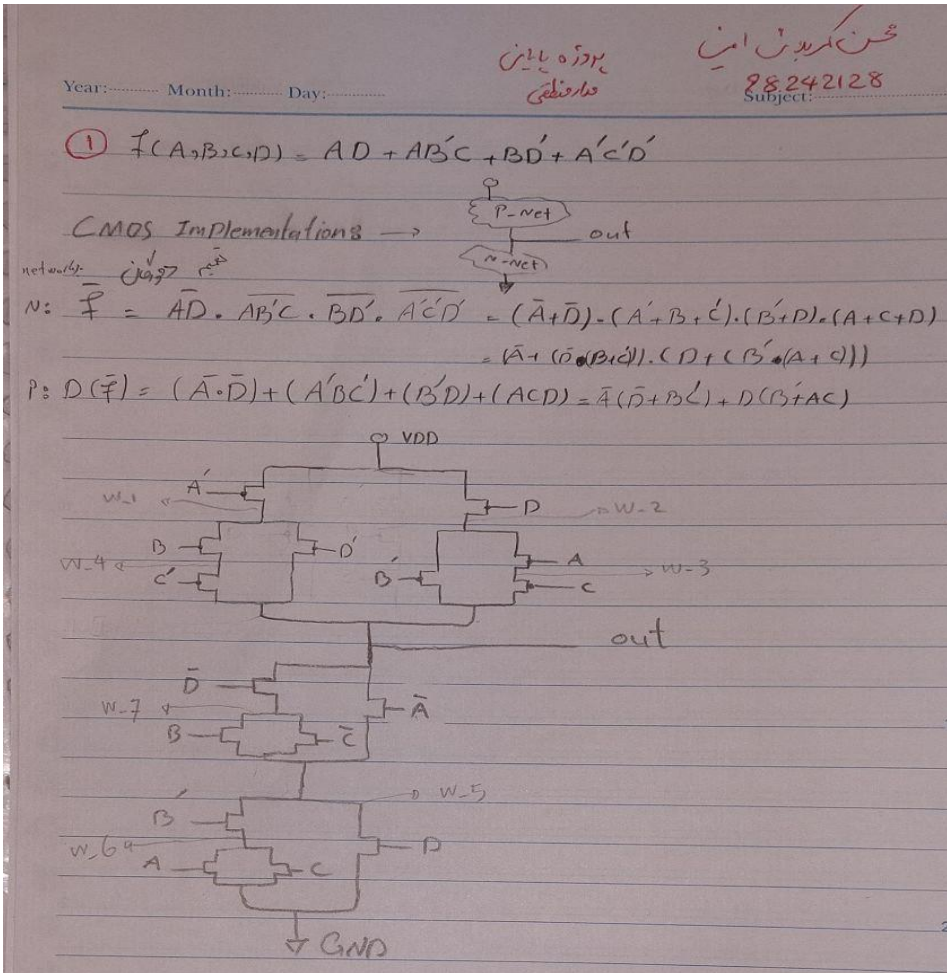
Student number: 98242128

Jan 25, 2023

EXECUTIVE SUMMARY

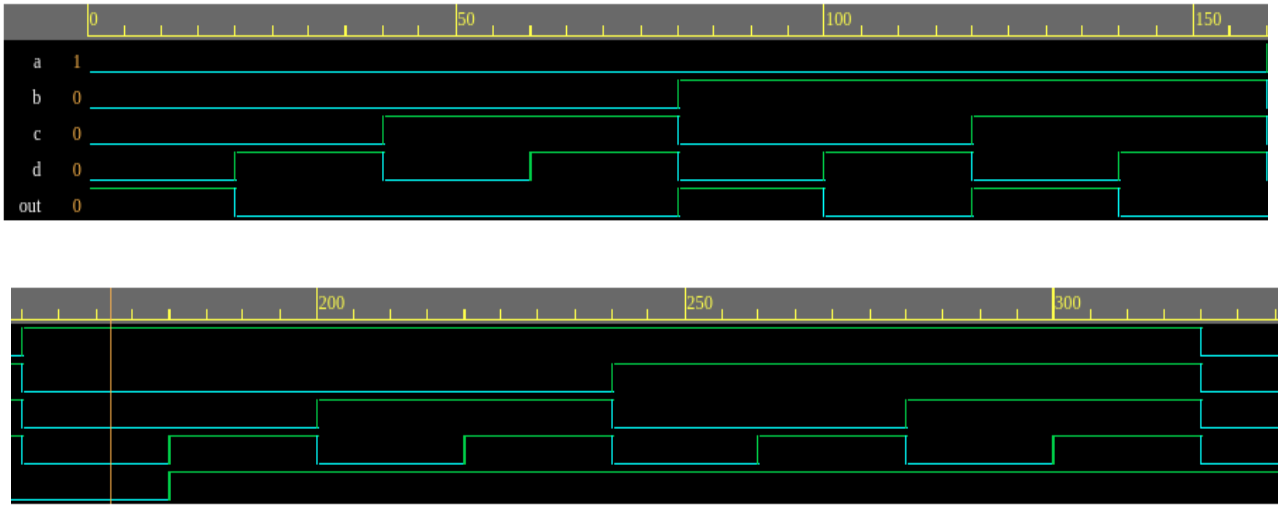
1. Transistor-level Boolean function

To implement this function, I made use of the **CMOS** method to understand the **relations** between transistors, in order to start coding in a **structural** abstraction. Here You can get a detailed view of what i have done in that area:



The wires, PMOS and NMOS transistors used in the code are presented.

The final outcome of the attached Verilog codes, can be represented in the below diagram of waves, which is the result of testing **all the possible** inputs.

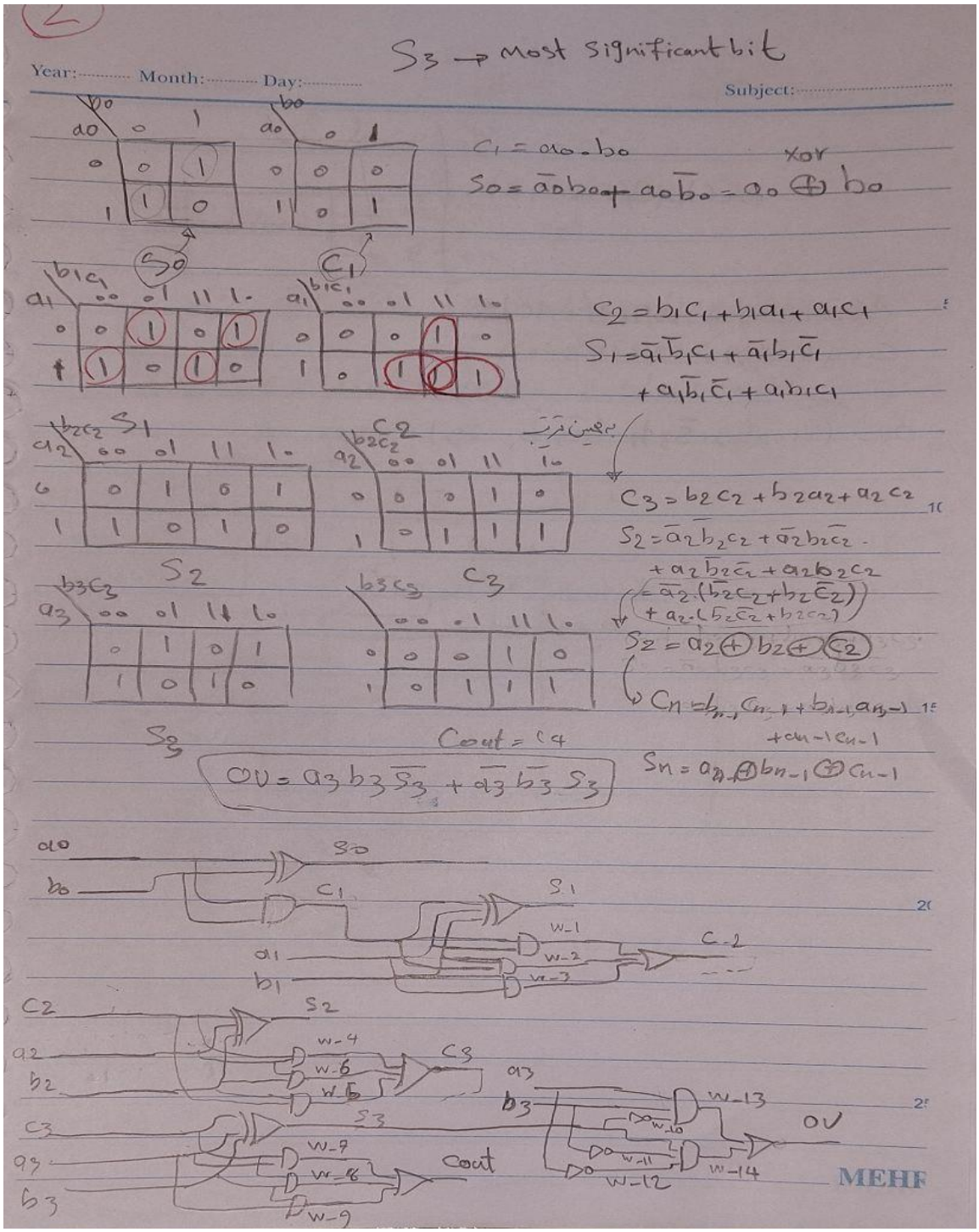


2. Gate-level 4-bit signed adder with overflow detector

To implement this circuit I needed to take advantage of **Karnaugh maps** and using **minterms**, define what each bit of the output (s[3:0]) consists of (e.g **what SOP makes s[1]**).

I also defined **carry bits** which would generate after each **one-to-one bit operation** and **take action in the next sum** operation. And ofcourse the **last** carry bit is represented as an **output** of the circuit.

The following picture is the view of the Karnaugh maps used and the circuits that would be built upon those maps.

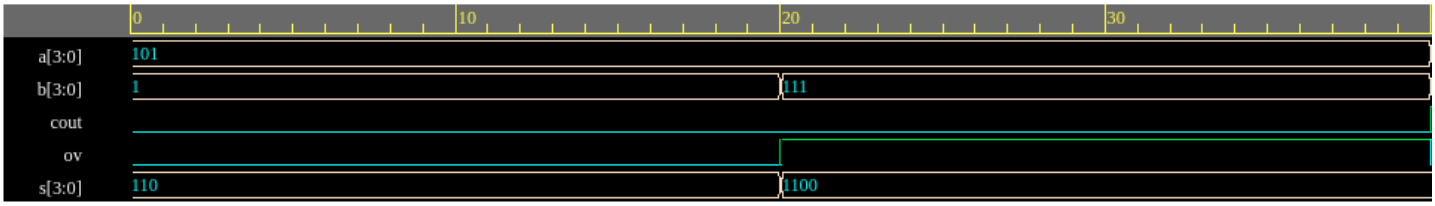


The gates and the wires they output, and also the **equations** to make those circuits, are represented above.

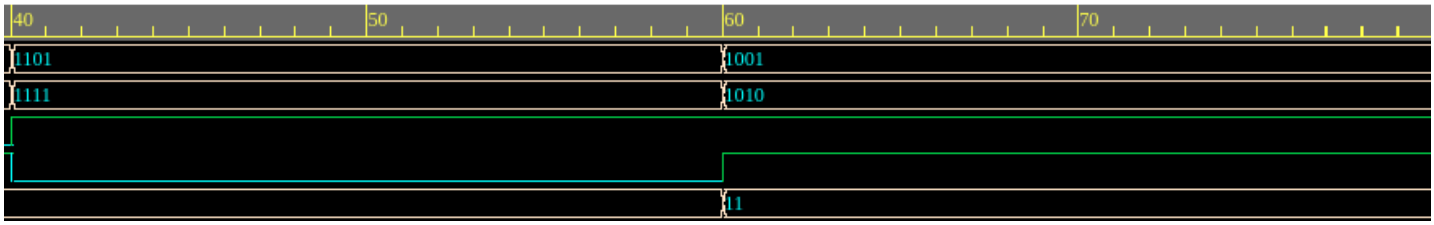
There was also an interesting point in this process which was that using **simplification** we could understand that all the "**s bits**" are generated from "**xor**" gates.

The result of the asked tests are illustrated in the below diagram:

1. Result of the **positive** inputs (**with/without overflow**)



2. Result of the **negative** inputs (**with/without overflow**)



3. 6-bit ALU with 4 different operations

To implement this circuit in Verilog, I needed to use the **dataflow coding style**. Which means making use of different operators to reach our goal. Using "**?:**" operator, I could "**assign**" the output to different operations.

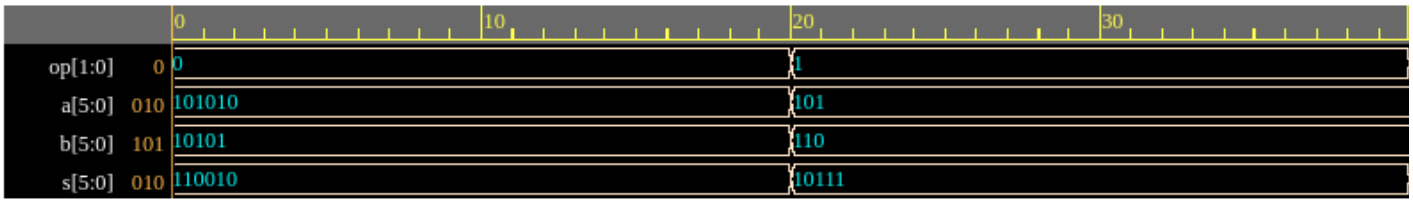
I also made use of a **temporary wire** to make it easier to get absolute value (**operation 3**).

There's not as much writing as others before for this one, but the usual calculations on binary to see if the code works fine.

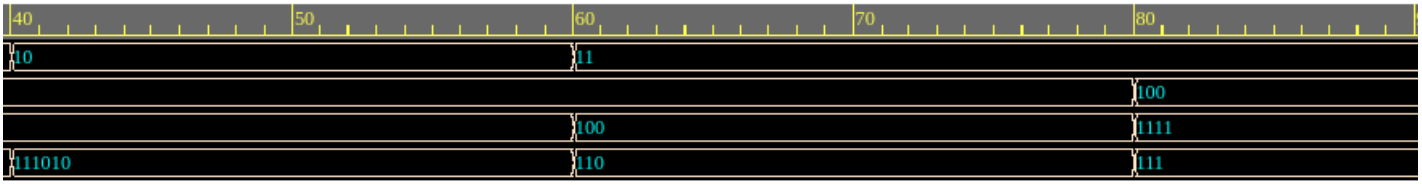
I prepared some tests which are present in the attached file regarding this item.

The final outcome of the test bench:

1. Operations 0 and 1



2. Operations 1 and 2(for two inputs)



NOTE: the pictures used in this document are also attached in the zip file so you could get a better view if these ones are not visible enough.