Hacettepe University Computer Engineering Department

Experiment 4 - Combinational Circuits in Verilog

Due date: 26/11/2024, 23:59:59 **Via: https://submit.cs.hacettepe.edu.tr/**

BBM233 Digital Design Lab - 2024 Fall

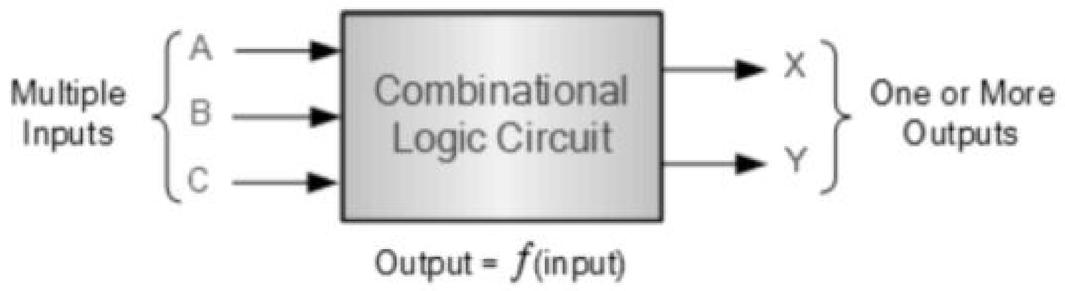
AIM

Designing and simulating combinational circuits in Verilog HDL.

BACKGROUND

Combinational Circuits

Combinational circuits are circuits whose outputs, at any instant of time, depend only on the present inputs (the combinational circuits do not use any memory elements). That is, the previous inputs or state of the circuit do not have any effect on its present state.



A combinational circuit performs a specific information-processing operation fully specified logically by a set of Boolean functions. The 'n' input variables come from an external source whereas the 'm' output variables go to an external destination. In many applications, the source or destination are storage registers.

Combinational Circuit Design Steps

To design a combinational circuit, start with the problem definition and use the following steps:

1. Identify the number of inputs and outputs of the circuit from the given specifications and assign them letter symbols.
2. Derive the truth table for each of the outputs based on their relationships to the inputs.
3. Simplify the Boolean function for each output (e.g. using Karnaugh Maps or Boolean algebra).
4. Construct the logic circuit diagram using Boolean functions obtained from the Step-3.

PLAGIARISM CONTROL NOTICE

Students must implement their solutions individually. All submissions will be submitted to a plagiarism check. Any submissions that show a high level of similarity will be graded with -100 pts and reported as plagiarism attempts to the ethics committee.

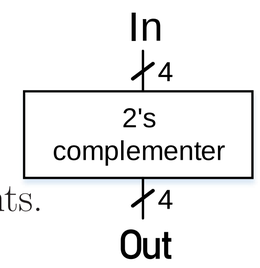
Part 0: 2's Complementer (25%)

The binary number system is one of the most popular number representation techniques used in digital systems, in which there are only two digits: 0 (off) and 1 (on). **Two's complement** is the way a computer uses to represent signed (positive, negative, and zero) integers. It is a mathematical operation to reversibly convert a positive binary number into a negative binary number with an equivalent (but negative) value. When the most significant bit is a one, the number is signed as negative. Two's complement is obtained by inverting (i.e. flipping) all bits, then adding a 1 to the inverted number.

Decimal	Signed-2's Complement
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000
- 0	—
- 1	1111
- 2	1110
- 3	1101
- 4	1100
- 5	1011
- 6	1010
- 7	1001
- 8	1000

2'S COMPLEMENTS

A **4-bit 2's complementer** circuit takes a 4-bit wide binary number as input, and produces a single 4-bit wide output that corresponds to its 2's complement. E.g., If the input is binary coded `3` (**`0011`**), the output is binary coded `-3` (**`1101`**). See the table above for more complements.

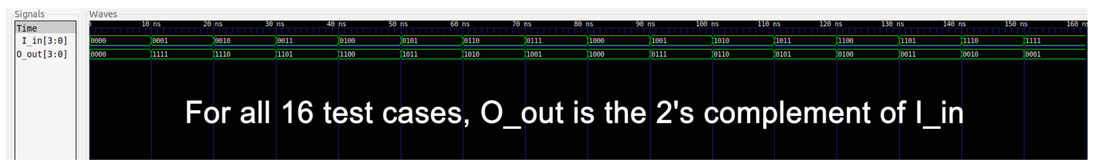


Experiment Steps

In this first part, you will design a **4-bit 2's complementer** and implement it in Verilog HDL.

- Fill out a truth table that represents each output bit given the four input bits. Obtain Boolean equations for the outputs Out_3 through Out_0 from the truth table and implement the 4-bit 2's complementer using **dataflow design approach** (use **assign** statements to implement each output signal: Out_3-Out_0). Save your file as **two_s_complement.v** (make sure your module is named the same!).
- Write a testbench and test for all 16 input combinations given in the truth table. Save your file as **two_s_complement_tb.v**
- **You MUST download and use these starter code files before you start working! Do NOT change the I/O port names, order, or width!**

Make sure to obtain a similar waveform that shows the correctness of your design.



Hacettepe University Computer Engineering Department

Experiment 4 - Combinational Circuits in Verilog

Due date: 26/11/2024, 23:59:59 **Via: https://submit.cs.hacettepe.edu.tr/**

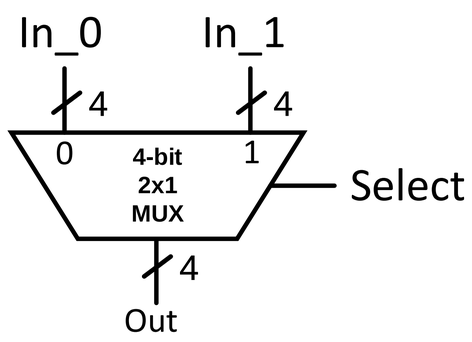
BBM233 Digital Design Lab - 2024 Fall

Part 1 - Multiplexer (20%)

A multiplexer (MUX) is a combinational logic circuit designed to switch one of several input lines through to a single common output line by the application of a control signal. A MUX has a maximum of 2^n data inputs. One of the inputs is connected to the output based on the value of the selection line(s). There will be 2^n possible combinations of 1s and 0s since there are 'n' selection lines.

4-bit 2-to-1 MUX

In a 4-bit 2-to-1 MUX, only one out of two 4-bit wide inputs is selected as the output based on a 1-bit select signal.



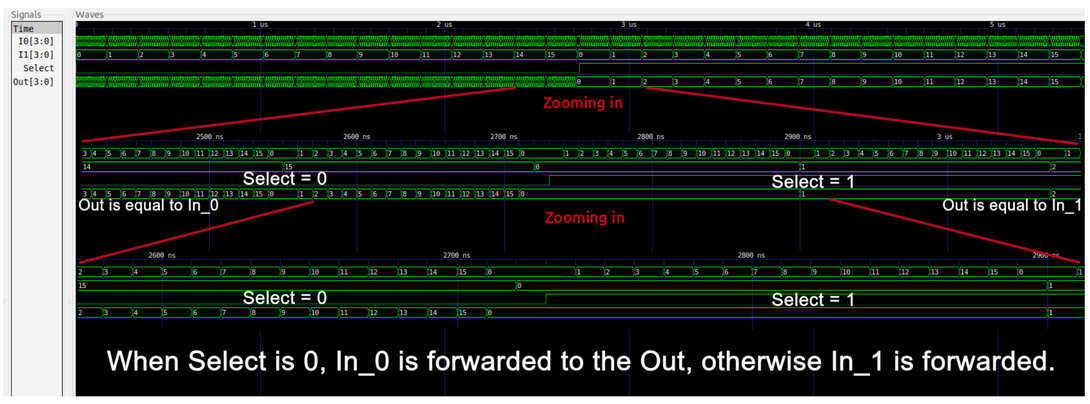
Select	Out
0	In_0
1	In_1

Experiment Steps

In this second part, you will design a **4-bit 2-to-1 MUX** and implement it in Verilog HDL.

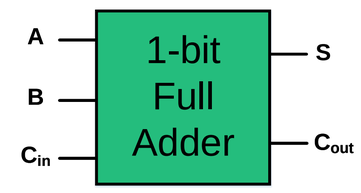
- Using the design specifications given in the truth table, implement a 4-bit 2-to-1 MUX using **any design approach you want** (A question to think about: is it possible to use an approach with which we can have a single line of code implementation of the MUX?). Save your file as **four_bit_2x1_mux.v**
- Write a testbench and test for all possible input combinations. Save your file as **four_bit_2x1_mux_tb.v**
- **You MUST download and use these starter code files before you start working! Do NOT change the I/O port names, order, or width!**

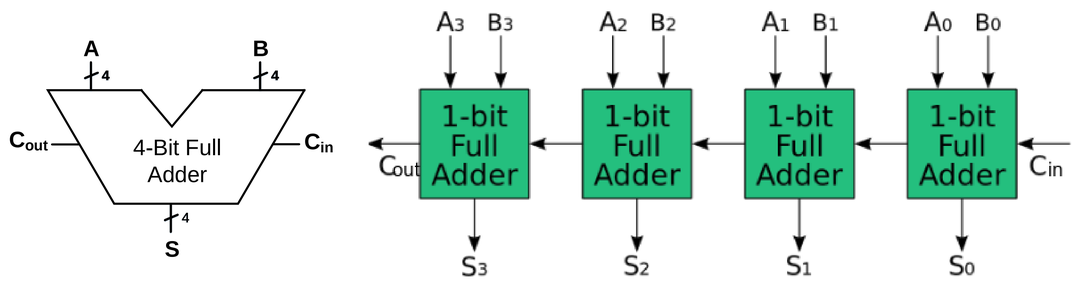
Make sure to obtain a similar waveform which shows the correctness of your design. You should prove that the correct signal is chosen for all possible select signal inputs.



Part 2 - 4-Bit Full Adder (25%)

A full adder is a combinational logic circuit that forms the arithmetic sum of three binary numbers. A full adder consists of three inputs and two outputs. Two of the input variables denoted by A and B represent the two numbers to be added. The remaining input variable C_{in} represents the carry from the previous summation. The two outputs of the logic circuit consist of the summation result and output carry C_{out}.





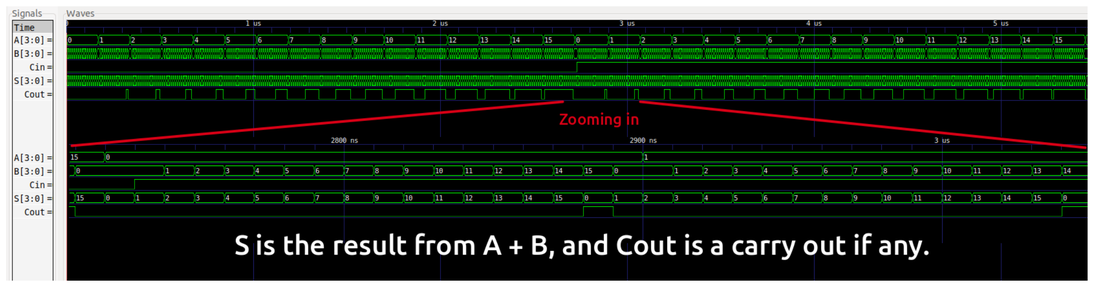
A 4-Bit Ripple Carry Adder can be implemented by instantiating four 1-Bit Full Adder modules as shown in the figure above.

Experiment Steps

In this third part, you will design a **4-bit full RCA adder** by using 4 single-bit full adders and implement it in Verilog HDL (check the practice example from the last week - you are allowed to use the solution from the slides for this part):

1. Implement a single-bit full adder using **data-flow** design approach. Save your module as **full_adder.v**. Don't forget to write a testbench and test your module before moving on to the next step.
2. Using 4 single-bit full adders, implement a 4-bit full RCA adder using **structural design approach** with **explicit association**. Save your module as **four_bit_rca.v**.
3. Test your modules by writing an appropriate testbenches **four_bit_rca_tb.v** and **full_adder_tb.v** for all possible input cases.
4. **You MUST download and use these starter code files before you start working! Do NOT change the I/O port names!**

Make sure to obtain a similar waveform that shows the correctness of your design.



Hacettepe University Computer Engineering Department

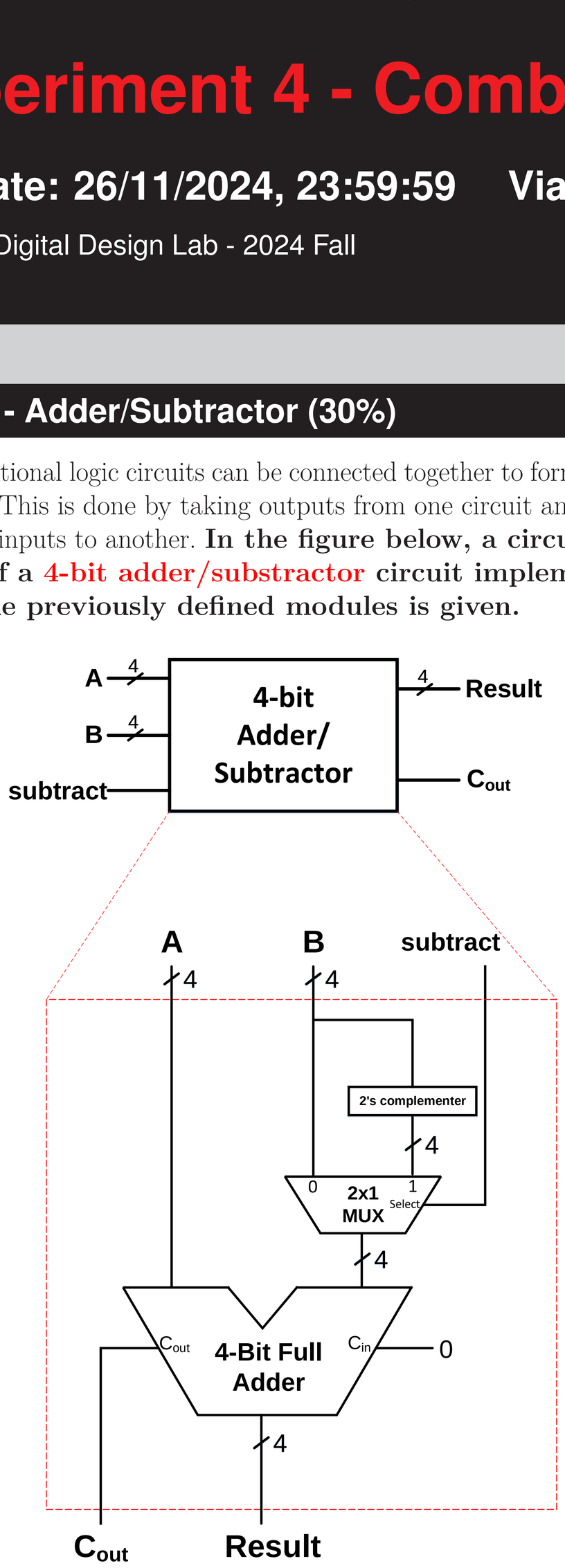
Experiment 4 - Combinational Circuits in Verilog

Due date: 26/11/2024, 23:59:59 **Via: https://submit.cs.hacettepe.edu.tr/**

BBM233 Digital Design Lab - 2024 Fall

Part 3 - Adder/Subtractor (30%)

Combinational logic circuits can be connected together to form larger circuits. This is done by taking outputs from one circuit and using them as inputs to another. **In the figure below, a circuit diagram of a 4-bit adder/substractor circuit implemented with the previously defined modules is given.**



A **4-bit adder/substractor** circuit has **two 4-bit inputs** `A` and `B`, and a **1-bit input** `subtract`. `A` and `B` are the numbers that will be either added or subtracted, while `subtract` is the mode signal: when `subtract` is **1 (HIGH)**, subtraction should be performed (**`Result = A - B`**), whereas when `subtract` is **0 (LOW)**, addition should be performed (**`Result = A + B`**).

This circuit has two outputs: one 4-bit output `Result` and **one 1-bit output** `Cout`. `Result` will output the result of the desired computation on the input numbers `A` and `B` (either their sum or difference), whereas `Cout` will output any carry-out resulting from the calculation.

This circuit shows us that it is possible to implement a subtractor using an adder. The idea is to obtain `A - B` by actually calculating `A + (-B)`, where `-B` is obtained by 2's complementing `B`. The multiplexer serves as the selector circuit to pass either `B` or `-B` to the adder based on the selected mode of operation via the signal `subtract`.

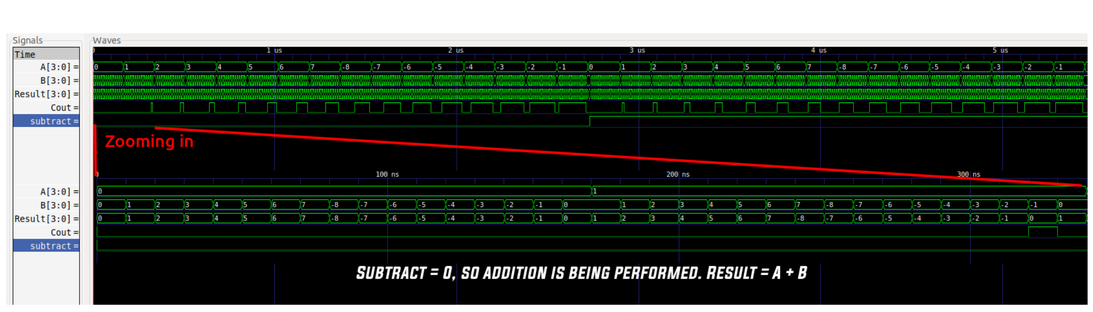
Part 3 - Adder/Subtractor (cont'd)

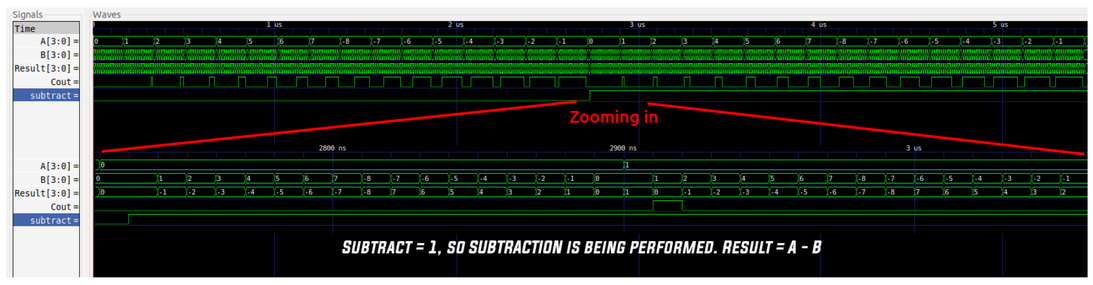
Experiment Steps

Implement the **4-bit adder/substractor** circuit in Verilog using the circuit diagram on the left and the following steps:

1. Implement your module in Verilog while looking at the connections in the diagram on the left. Use **structural design approach and explicit association** to connect the relevant ports to each other. The module must be named as **four_bit_adder_subtractor.v**.
2. Test it by writing an appropriate testbench **four_bit_adder_subtractor_tb.v** for all possible input cases.
3. **You MUST download and use these starter code files before you start working! Do NOT change the I/O port names, order, or bit width!**

Make sure to obtain a similar waveform that shows the correctness of your design.





Automatic grading [VERY IMPORTANT!]

Your submissions will be graded using an automatic grading script. They **will not be** white-box tested. So, you are expected to **match the given wave forms 100%**, to receive full-credit.

You MUST download and use these starter code files before you start working! Do NOT change the I/O ports (names, order, bit width, etc.)!

Hacettepe University Computer Engineering Department

Experiment 4 - Combinational Circuits in Verilog

Due date: 26/11/2024, 23:59:59 **Via: https://submit.cs.hacettepe.edu.tr/**

BBM233 Digital Design Lab - 2024 Fall

Testing your codes before submission

You can test your codes via our Tur[6]Bo Grader before the submission to see which tests you are passing. **Note that this is used for testing purposes only**, and you still have to submit your full codes and report via https://submit.cs.hacettepe.edu.tr/ to get graded.

Submission via submit.cs

Submissions will be accepted via https://submit.cs.hacettepe.edu.tr/. **Note that the deadline for submission of the report and all codes is 26/11/2024, 23:59:59.**

Your submission will include the Verilog codes and a report PDF and it must be in the following format to be accepted:

- b<studentID>.zip
  - two_s_complement.v
  - two_s_complement_tb.v
  - four_bit_2x1_mux.v
  - four_bit_2x1_mux_tb.v
  - full_adder.v
  - full_adder_tb.v
  - four_bit_rca.v
  - four_bit_rca_tb.v
  - four_bit_adder_subtractor.v
  - four_bit_adder_subtractor_tb.v

Note that only ZIP archives are accepted!

Do not use any ready code from the internet as it will be considered as plagiarism if more than one student has the same solution even if they found the solutions individually.

Grading

Verilog codes: 100%

- Part 0: 25%
- Part 1: 20%
- Part 2: 25%
- Part 3: 30%