Verilog Practice - Combinational Circuits

Week 9 - UNGRADED

BBM233 Digital Design Lab - 2024 Fall

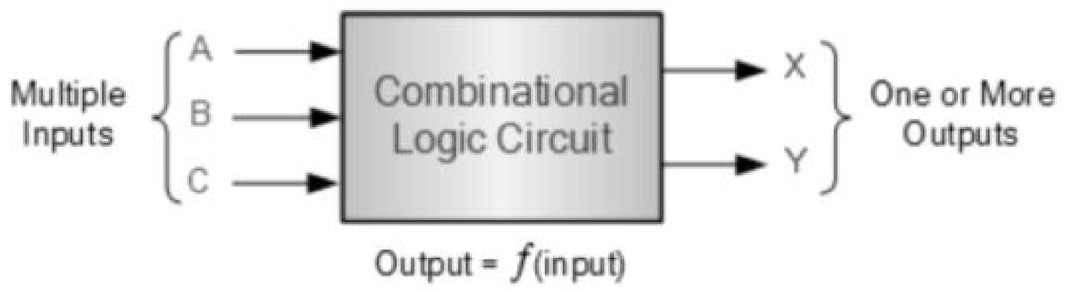
AIM

In this practice assignment you will practice 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.



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 specifications.
2. Derive the truth table for each of the outputs based on their relationships to the input.
3. Simplify the Boolean function for each output (e.g. using Karnaugh Maps or Boolean algebra).
4. Construct the circuit using Boolean functions obtained from the Step-3.

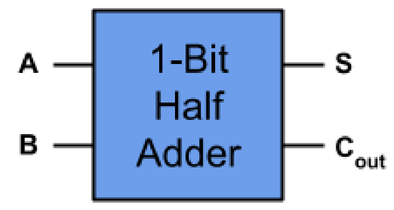
Grading Policy

Verilog codes: 100%

- Part 0 - Half Adder Implementation and Testbench: 20%
- Part 1 - Full Adder Implementation and Testbench: 20%
- Part 2 - Multiplier Implementation and Testbench: 30%
- Part 3 - 4-Bit RCA Implementation and Testbench: 30%

Part 0: 1-Bit Half Adder (30%)

A **Half Adder** is a fundamental combinational digital circuit designed to add two single-bit binary numbers A and B, and two outputs, named sum (S) and carry (C). The sum (S) represents the result of the addition, while the carry (C) signifies if an overflow occurred. Essentially, if adding A and B produces a value of 2 (binary '10'), the carry (C) is set to `1` and sum (S) is set to `0`.



1. The Half Adder has two 1-bit inputs: A and B.
2. It produces two 1-bit outputs: sum (S) and carry (C).

Inputs		Outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figure 1: Half Adder truth table.

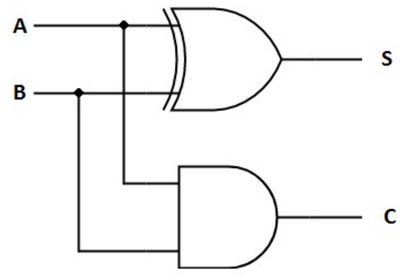


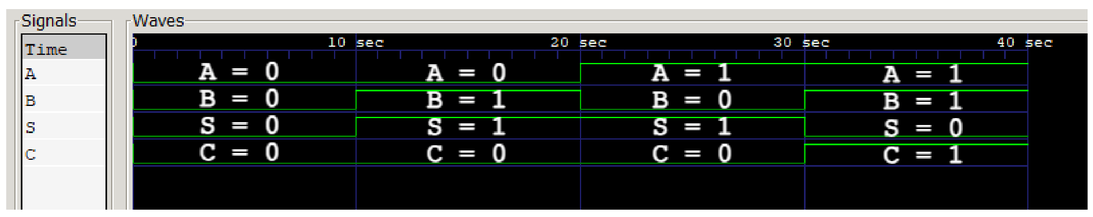
Figure 2: Half Adder circuit diagram.

Experiment Steps

In this first part, you will design a **1-bit half adder** and implement it in Verilog HDL.

- Using the given truth table that represents each output bit given the two input bits A and B, obtain Boolean equations for the outputs sum (S) and carry (C), and implement the 1-bit Half Adder using **dataflow design approach** (use **assign** statements to implement each output signal. Save your file as **half_adder.v** (make sure your module is named the same!).
- Write a testbench and test for all 4 input combinations given in the truth table. Save your file as **half_adder_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

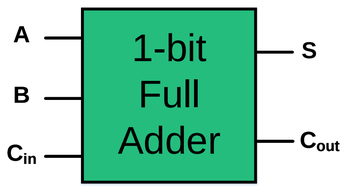
Verilog Practice - Combinational Circuits

Week 9 - UNGRADED

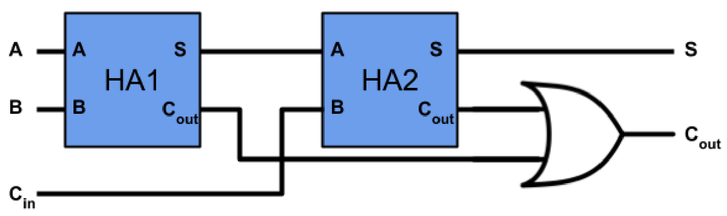
BBM233 Digital Design Lab - 2024 Fall

Part 1: 1-Bit Full Adder (30%)

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}. All ports are 1 bit wide.



1. The Full Adder has three 1-bit inputs: A, B, and carry-in (C_{in}).
2. It produces two 1-bit outputs: sum (S) and carry-out (C_{out}).
3. We can combine two Half Adders to get the Full Adder.
4. And use an OR gate to combine the carries from the two Half Adders.



Inputs			Outputs	
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figure 3: Full Adder circuit and truth table.

Experiment Steps

In this part, you will design a **1-bit full adder** using already designed half adder module, and implement it in Verilog HDL.

- Using the given circuit diagram above, implement the 1-bit Full Adder from two half adders and an OR gate, using **structural and gate-level design approach with explicit association**. Save your file as **full_adder.v** (make sure your module is named the same!).
- Write a testbench and test for all 8 input combinations given in the truth table. Save your file as **full_adder_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.

Part 2 - 3-Bit Multiplier (40%)

A 3-bit multiplier, also termed a 3x3 multiplier, is a circuit designed to accept two 3-bit numbers as inputs and yield a 6-bit number as its output, representing the product of the input numbers. Consider, for instance, the numbers $A = 5$ represented as $(101)_2$ and $B = 2$ represented as $(010)_2$ in binary. The output for this pair would be $P = A \times B = 10$, which in 6-bit binary is $(001010)_2$.

Inputs: Two 3-bit numbers: $A = A_2A_1A_0$ and $B = B_2B_1B_0$

Output: A 6-bit product: $P = A \times B = P_5P_4P_3P_2P_1P_0$

The design of this multiplier circuit incorporates half adders, full adders, and several other logic gates, as illustrated in the figure below.

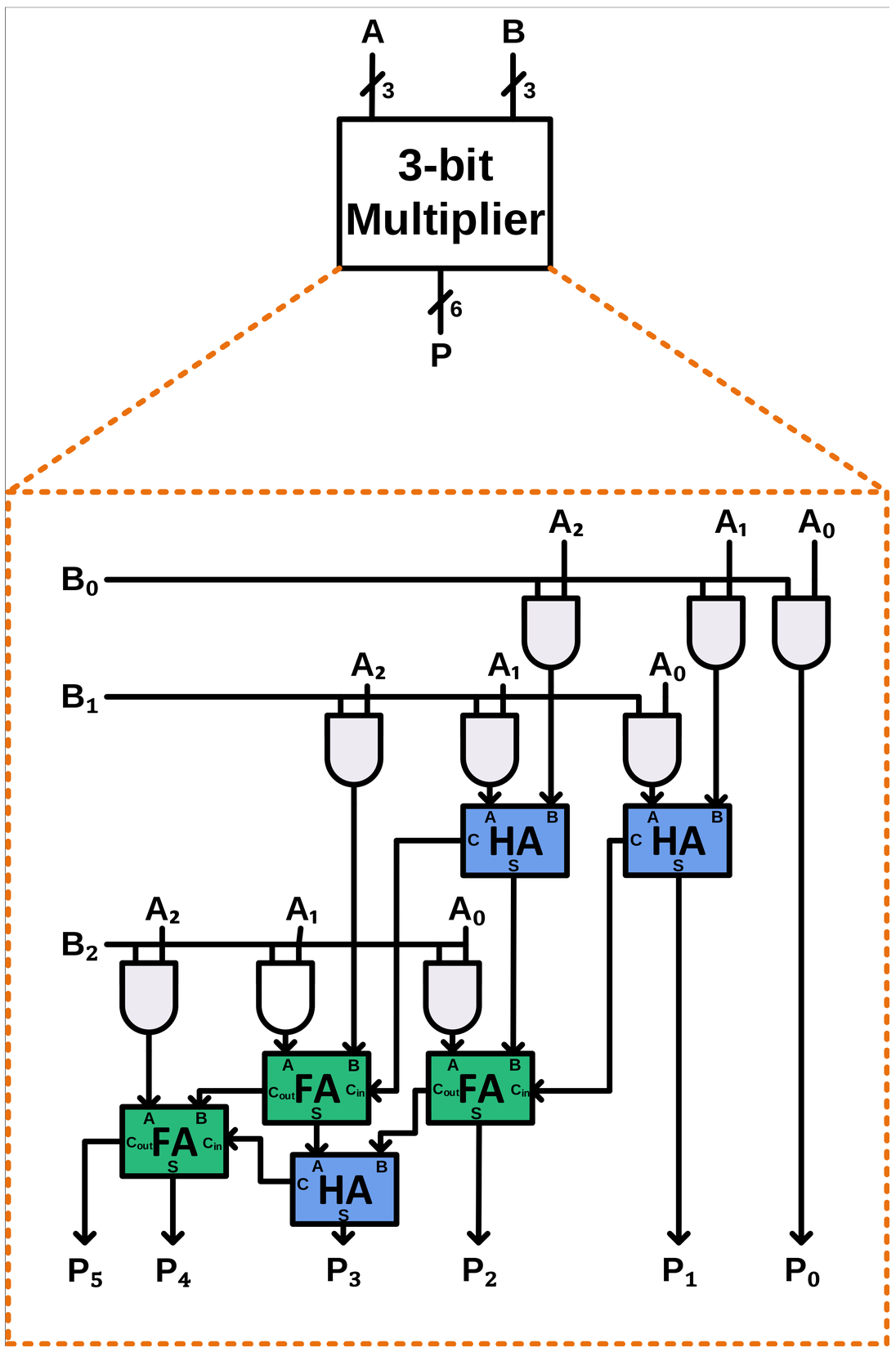


Figure 4: 3-bit multiplier circuit diagram.

Hacettepe University Computer Engineering Department

Verilog Practice - Combinational Circuits

Week 9 - UNGRADED

BBM233 Digital Design Lab - 2024 Fall

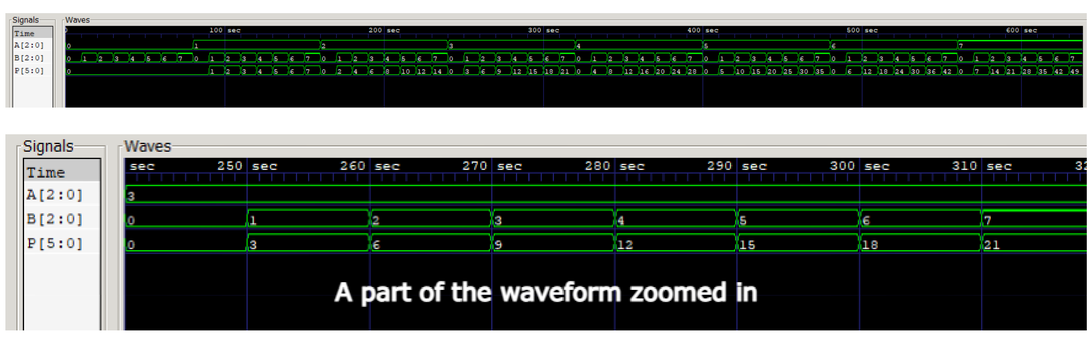
Part 2 - 3-Bit Multiplier (cont'd)

Experiment Steps

In this third part, you will design a **3-bit multiplier** by using 3 half adders, 3 full adders, and additional logic gates as necessary, and implement it in Verilog HDL using the given circuit diagram:

1. Implement a 3-bit multiplier using a combination of **structural and gate-level** design approaches. Save your module as **multiplier.v**.
2. Test your module by writing an appropriate testbench **multiplier_tb.v** for all possible input cases (**64 possible input cases in total**).
3. **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.



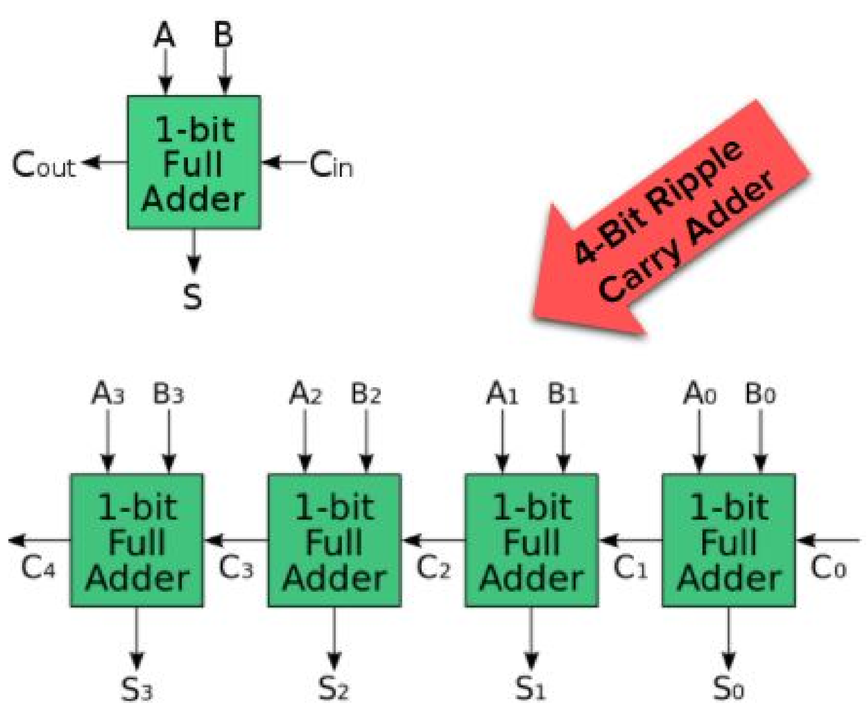
Automatic grading [VERY IMPORTANT!]

Your submissions will be graded using an automatic grading script. There **will not be** any manual grading. So, you are expected to **match the given waveforms 100%**, to receive the 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.)!

You must 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 via https://submit.cs.hacettepe.edu.tr/ to get graded.

Part 3: 4-Bit Ripple Carry Adder



Implement a 4-Bit Ripple Carry Adder in Verilog in the following steps:

1. Implement a 1-Bit Full Adder using **behavioral dataflow design approach**. Fill in the truth table, find the corresponding functions for Sum and Carry_out, write a Verilog module **full_adder.v**.
2. Test it by writing a testbench **full_adder_tb.v** for all possible input cases.
3. Implement a 4-Bit Ripple Carry Adder by instantiating your 1-Bit Full Adder module as many times as necessary (refer to the figure above for circuit structure). Use **structural design approach and explicit association**. Name your module as **four_bit_RCA.v**
4. Test it by writing an appropriate testbench **four_bit_RCA_tb.v** for all possible input cases.

Solutions can be found in Verilog Tutorial Slides.

Submission format

Your submission must be in the following format to be accepted:

- b<studentID>.zip
  - half_adder.v
  - half_adder_tb.v
  - full_adder.v
  - full_adder_tb.v
  - multiplier.v
  - multiplier_tb.v
  - four_bit_RCA.v
  - four_bit_RCA_tb.v