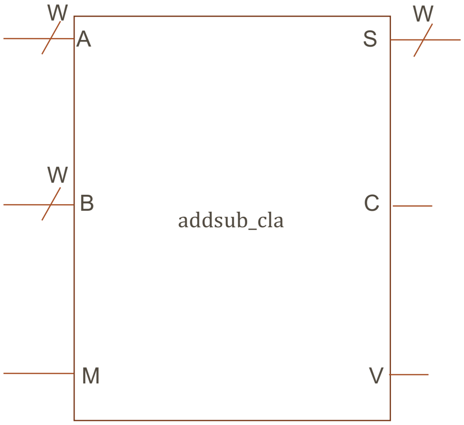
Seung Jun Baek



1 Outline

In this assignment, you are asked to design a verilog module for a binary adder-subtractor. The main arithmetic operation, that is addition and subtraction, must be implemented using a **carry-lookahead adder**. The number of bits for the addition or subtraction must be configurable, that is, you must use verilog `parameter` keyword in order for a parameterized implementation of the adder-subtractor. You need to design two modules, `addsub_cla` and `cla_gen`. The following is the specification for the modules.

2 Specification

2.1 module `addsub_cla`

Module `addsub_cla` is a parameterized module which performs addition or subtraction, based on carry-lookahead mechanism. The input and output signals are given as follows:

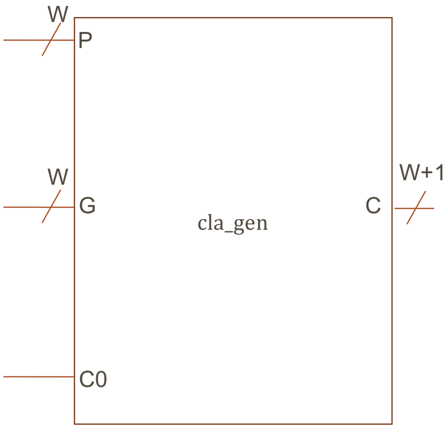
- input `A,B`. The signals are W-bit bus $(W \geq 1)$, and W is the configurable parameter. These are 2's complement (signed) representation of binary numbers to add or subtract.
- output `S`. This signal is a W-bit bus. This is the result of addition/subtraction, and is also a 2's complement (signed) binary number.

- output C. This 1-bit output signal represents the carry generated at the most significant bit after addition/subtraction.
- input M. This 1-bit control signal determines whether addition or subtraction is performed. If $M = 0$, the output S corresponds to $A + B$. If $M = 1$, the output S corresponds to $A - B$.
- output V. This 1-bit signal indicates whether an overflow has occurred as a result of addition/subtraction. $V = 0$ means normal, and $V = 1$ means an overflow.

The following are the requirements of this module: **IF YOU DO NOT FOLLOW THE REQUIREMENTS, YOU WILL NOT GET FULL CREDIT, AND MAY EVEN GET ZERO POINTS.**

1. When you use parameter for the bit width of the input/output numbers, the name of the parameter MUST be W.
2. Your module MUST instantiate a module for carry-lookahead generator. The name of the module must be cla_gen (explained in the following section).
3. The name of the instance of the cla_gen module in your addsub_cla module MUST have the following name: CLAGEN. The name is case sensitive, that is, you must use uppercase letters (capital letters) for your module name. **Summary: your addsub_cla must instantiate a module called cla_gen, and the name of the instance must be CLAGEN.**

2.2 module cla_gen



Module cla_gen is a parameterized module implementing a carry-lookahead generator. The function of carry-lookahead generator is given by Fig. 4-12 of the textbook, and also in lecture slides. The input and output signals are given as follows:

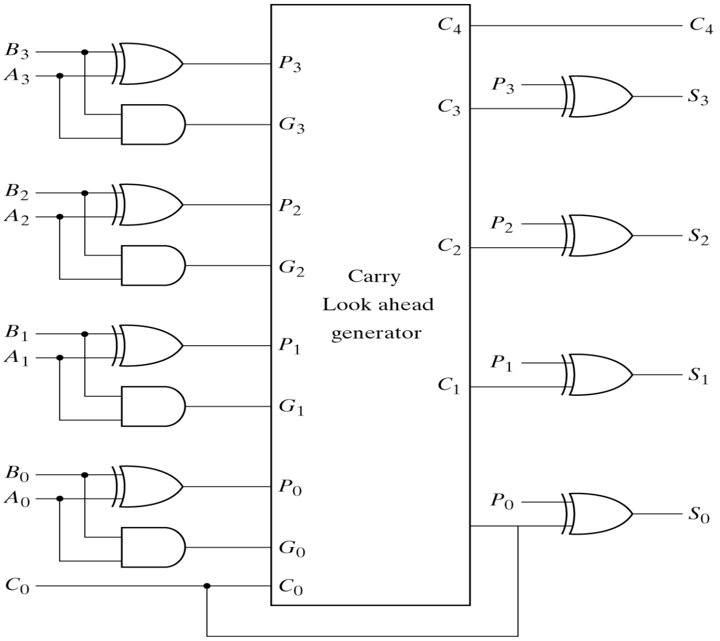


Fig. 4-12 4-Bit Adder with Carry Lookahead

Figure 1: Fig 4-12 from textbook

- input `P,G`. These signals are W-bit bus which correspond to `P` and `G` signals in the carry-lookahead generator. W is the configurable parameter.
- input `C0`. This 1-bit input signal which represents the input carry C_0 of the carry-lookahead generator.
- output `C`. This is $(W + 1)$-bit bus which generates the carries $C_0, C_1, \ldots, C_W$. Note that `C[0]` is simply equal to input `C0`.

The following are the requirements of this module: **IF YOU DO NOT FOLLOW THE REQUIREMENTS, YOU WILL NOT GET FULL CREDIT, AND MAY EVEN GET ZERO POINTS.**

1. When you use `parameter` for the bit width of the input/output signals, the name of the parameter MUST be `W`.

3 What to submit

- You must submit a file, named `adder.v`, and the file must contain the implementation of two modules `cla_gen` and `addsub_cla`.

- Upload your file at Blackboard before deadline (no late submission accepted).

4 Comments

- You don't have to find the minimum (optimal) design of your module.
- You may implement as many submodules as you like, if necessary. In that case, all the modules must be contained within the `adder.v` file.
- Make sure you follow the requirements in Section 2.1 and 2.2. Also your input/output signal names must exactly match the instruction in Section 2.1 and 2.2.

5 How to test your module

In the blackboard, I have uploaded `cla_main.v` so that you can test your module. The file contains `main` module. The `main` module instantiates `addsub_cla` module, and feeds the test input signals `a, b, sub_not_add` to the module. The test results can be monitored using `gtkwave` tool by looking at `sum, carry, overflow` signals.

You can run the following in your command line to compile and simulate the source files.

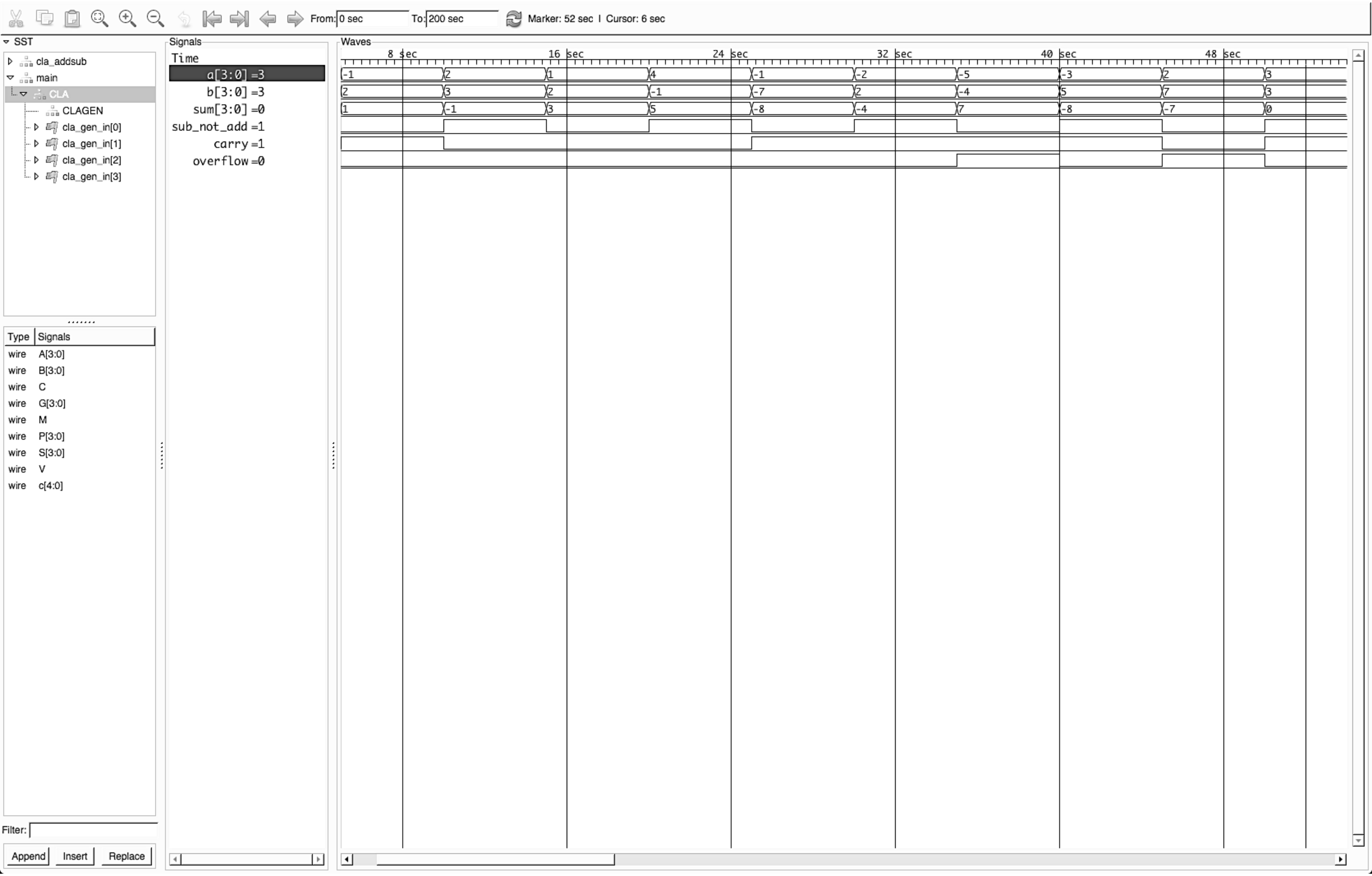
- `iverilog -o h2.out adder.v cla_main.v`
- `vvp h2.out`
- `gtkwave h2_output.vcd`

A screenshot is attached at the end of this document. In that example, the bit width of the adder is set to 4.

6 Grading

- 5 points (full) if your module works correctly, that is,
  1. if `addsub_cla` is correctly designed. For example, in the main module, `sum, carry, overflow` must be produced correctly, given the input test vector.
  2. if `cla_gen` is correctly designed. Given the test input, the input signals `P, G, C0` to the module and output signals `C` from the module must have proper values.
  3. if your modules can be instantiated with arbitrary number of input/output bits. For example, we will test if your module can be instantiated as 16-bit adder or 32-bit adder, etc.

The rest of case is 0 points, i.e., if you do not submit (or late), or if your file does not compile correctly, or produces wrong results.



Type	Signals
wire	A[3:0]
wire	B[3:0]
wire	C
wire	G[3:0]
wire	M
wire	P[3:0]
wire	S[3:0]
wire	V
wire	c[4:0]

Signals

```
Time
a[3:0] =3
b[3:0] =3
sum[3:0] =0
sub_not_add =1
carry =1
overflow =0
```