2026 Digital IC Design

Homework 1: 4-bit Ripple Carry Adder

1. Introduction:

This homework focuses on designing digital modules to implement a 4-bit Ripple Carry Adder (RCA) circuit using **hierarchical design**. You are requested to design a 1-bit Half Adder (HA) module, which performs binary addition of two 1-bit inputs and outputs the sum and carry. The Half Adder is then used to construct a Full Adder (FA), which includes two Half Adders to incorporate the carry-in bit.

The Full Adder is further used to construct a 4-bit Ripple Carry Adder, which computes the sum of two 4-bit unsigned binary numbers along with an input carry. The circuit should be implemented by hierarchical module composition: starting from **Half Adder → Full Adder → RCA**.

You are encouraged to verify the correctness of each level using comprehensive testbenches. The overall architecture should follow the hierarchical design as shown in the provided diagram.

1.1 1-bit Half Adder

The Half Adder circuit used in this architecture is shown in Figure 1, and its specification and I/O interface are listed in Table I. The Half Adder takes two 1-bit binary inputs and computes a 1-bit sum and a 1-bit carry as outputs.

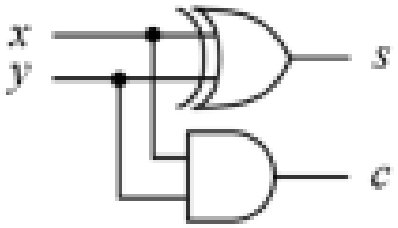

Fig 1. Architecture of the Half Adder

Table I. Specification and I/O interface of Half Adder

Signal Name	I/O	Width	Description
x	I	1	First 1-bit input operand
y	I	1	Second 1-bit input operand
s	O	1	The sum of the input x and y
c	O	1	A carry when both x and y are 1

1.2 1-bit Full Adder

The architecture of the Full Adder module for this homework is shown in Figure 2, and its specification and I/O interface is listed in Table II. The Full Adder module takes three 1-bit binary inputs—two operands and one carry-in—and produces a 1-bit sum and a 1-bit carry-out as outputs.

In this homework, you must construct the Full Adder circuit using your 1-bit Half Adder module in a hierarchical design.

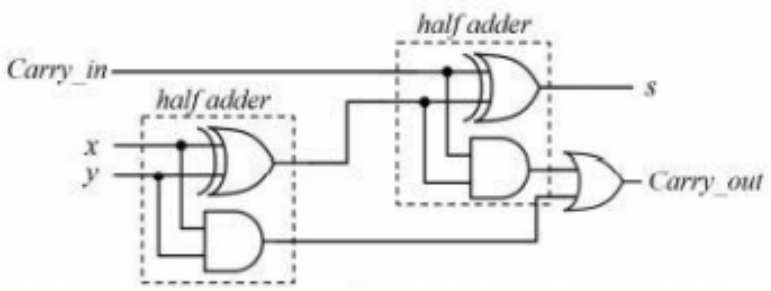

Fig 2. Architecture of the Full Adder

Table II. Specification and I/O interface of Full Adder

Signal Name	I/O	Width	Description
x	I	1	First 1-bit input operand
y	I	1	Second 1-bit input operand
c_in	I	1	Carry-in input from the previous adder stage
s	O	1	1-bit sum result of the addition
c_out	O	1	Carry-out to be passed to the next adder stage

1.3 4-bit Ripple Carry Adder

The architecture of the Ripple Carry Adder (RCA) module for this homework is shown in Figure 3, and its specification and I/O interface is listed in Table III. The RCA module takes two 4-bit binary inputs and a 1-bit carry-in, and produces a 4-bit sum along with a 1-bit carry-out as outputs.

In this homework, you must construct the RCA circuit using your previously designed 1-bit Full Adder modules in a hierarchical and cascaded structure.

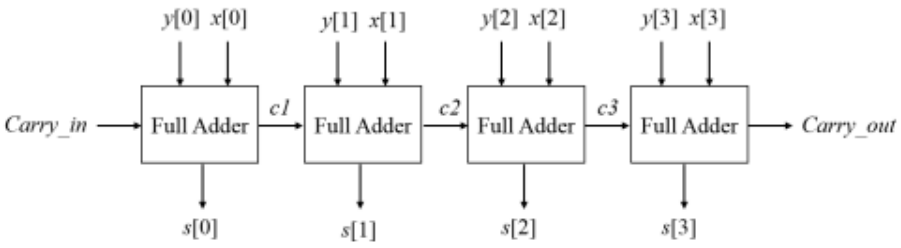

Fig 3. Architecture of the RCA

Table III. Specification and I/O interface of RCA

Signal Name	I/O	Width	Description
x	I	4	First 4-bit input operand
y	I	4	Second 4-bit input operand
c_in	I	1	Carry-in input, typically applied to the least significant bit (LSB)
s	O	4	4-bit sum result of the addition

c_out	O	1	Final carry-out generated from the most significant bit (MSB)

1.4 File Description

File Name	Description
HA.v	The module of 1-bit Half Adder
FA.v	The module of 1-bit Full Adder
RCA.v	The module of 4-bit Ripple Carry Adder
testfixture.sv	Testbench file. This file **is not allowed** to be modified

2. Scoring:

2.1 1-bit Half Adder [20%]

The result should be generated correctly, and you will get the following message in ModelSim simulation.

```
# === Testing Half Adder ===
# PASS: #0 data is correct
# PASS: #1 data is correct
# PASS: #2 data is correct
# PASS: #3 data is correct
# PASS: #4 data is correct
# PASS: #5 data is correct
# PASS: #6 data is correct
# PASS: #7 data is correct
# >>> [HA] TEST PASSED
```

Fig. 4 Simulation result for Half Adder

2.2 1-bit Full Adder [30%]

The result should be generated correctly, and you will get the following message in ModelSim simulation. **Please construct the Full Adder circuit with your Half Adder modules. Otherwise, you can just get half of the points.**

```
# === Testing Full Adder ===
# PASS: #0 data is correct
# PASS: #1 data is correct
# PASS: #2 data is correct
# PASS: #3 data is correct
# PASS: #4 data is correct
# PASS: #5 data is correct
# PASS: #6 data is correct
# PASS: #7 data is correct
# >>> [FA] TEST PASSED
```

Fig. 5 Simulation result for Full Adder

2.3 4-bit Ripple Carry Adder [50%]

The result should be generated correctly, and you will get the following message in ModelSim simulation. **Please construct the Ripple Carry Adder circuit with your Full Adder modules. Otherwise, you can just get half of the points.**

```
# === Testing Ripple Carry Adder ===
# PASS: #0 data is correct
# PASS: #1 data is correct
# PASS: #2 data is correct
# PASS: #3 data is correct
# PASS: #4 data is correct
# PASS: #5 data is correct
# PASS: #6 data is correct
# PASS: #7 data is correct
# PASS: #8 data is correct
# PASS: #9 data is correct
# >>> [RCA] TEST PASSED
```

Fig. 6 Simulation result for RCA

3. Submission

3.1. File Submission

You should classify your files into two directories and compress them to .zip format. The naming rule is StudentID.zip. If your file is not named according to the naming rule, you will lose five points.

	RTL
*.v	All of your Verilog RTL code
	Documentary
Report.pdf	The report file of your design (in pdf).

Fig. 7 File hierarchy

StudentID.zip
- StudentID
  - RTL
    - *.v (All of your Verilog RTL code)
  - Documentary
    - Report.pdf

3.2. Report File

Please follow the spec of report. You are asked to describe how the circuit is designed as detailed as possible.

3.3. Notes

a. Please submit your .zip file to folder HW1 in moodle.
   Deadline: 2025/10/05 23:55
b. Late submission will only be accepted within a week and will result in a penalty of 5 points per day.
c. TA email: p76135038@gs.ncku.edu.tw