Experiment 4. Hierarchical Design

Masud ul Hasan · Aiman El-Maleh · Ahmad Khayyat – Version 151, 3 August 2015

Table of Contents

1. Objectives

- Get *more* familiar with FPGAs and design tools.
- Learn about symbols and hierarchical design.
- Build reusable half adders and full adders.
- Build a 4-bit adder using full adders.
- Model the half adder, full adder, and 4-bit adder using the Verilog hardware description language (HDL).

2. Materials Required

- An FPGA prototyping board.
- Design and simulation software tools.

3. Background

You have encounterd the half adder circuit in the previous experiment. For review purposes, below are the half adder circuit schematic diagram and its truth table.

Table 1. Half Adder Truth Table

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

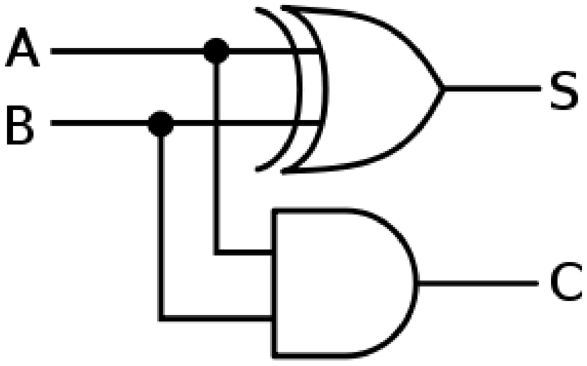

Figure 1. Half Adder Circuit

A *full adder*, as you also have seen in the previous experiment, is a circuit that adds three bits. The Logic Diagram of a Full Adder figure below shows one way to build a full adder. Note that you can use two half adders to build a full adder, in which case you will also need one 2-input OR gate.

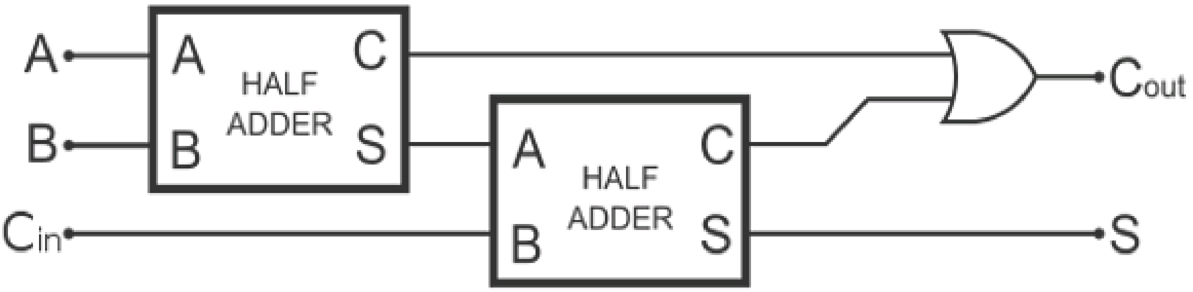

Figure 2. Logic Diagram of a Full Adder

Reusing digital components, such as the half adder being used twice to build a full adder in the Logic Diagram of a Full Adder figure, is a common technique in digital design. As such, there are facilities in digital design tools and hardware description languages for packaging a design as a symbol, or a module, and then instantiating it as many times as you need.

In this experiment, you will create a half adder symbol, then use it to build a full adder. Afterwards, you will create a symbol for the full adder, and use it to build an *n*-bit adder, which is a circuit that adds two *n*-bit numbers.

In an *n*-bit adder, a full adder can be used to add a bit of the first number with the corresponding bit of the second number. Because adding two bits may generate a carry bit that must be added to the bits at the next more significant position, a third input is needed at each position to use the carry that may be generated from the previous position. Therefore, each position requires a 3-input adder, or a full adder, to accommodate the carry bit from the previous position, which is known as the *carry-in*, and will generate a carry bit that must be added to the bits of the next position, which is known as the *carry-out*. That is, the *carry-in* of each full adder is the *carry-out* of the previous full-adder. This type of adder is called a *ripple carry adder (RCA)*.

In this experiment, you will build a 4-bit binary adder using four full adders symbols that you will have created previously. The logic diagram of the 4-bit adder is shown in the Logic Diagram of a 4-Bit Adder figure.

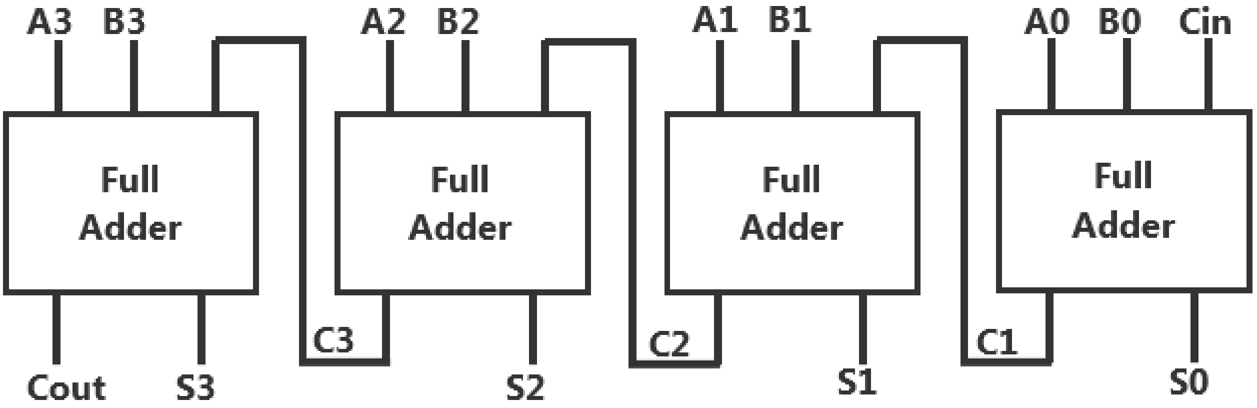

Figure 3. Logic Diagram of a 4-Bit Adder

4. Tasks

In this experiment, you will use half adders to build a full adder, and use full adders to build a 4-bit binary adder. You will implement these circuits both using schematic entry and using the Verilog hardware description language.

4.1. Create a Half Adder Symbol

In this task, you will use the `half_adder` circuit that you created in the previous experiment to define a new `half_adder` symbol that you can easily reuse.

This task is highly dependent on the design software you are using. Typically, the design software suite will include a tool that facilitates the creation of custom symbols based on a user-created design.

> *Xilinx ISE Schematic Editor*
> The Xilinx ISE design software suite includes a *schematic editor* tool that can typically be launched through the *Tools* menu.

Typically, such a tool would prompt the user for the following options:

- Design format: whether the design of the symbol is to be specified using a schematic drawing or a hardware description language. For this experiment, select the schematic option.
- Design file: the file that implements the design of the new symbol. To create a half adder symbol, use the `half_adder` circuit that you created in the previous experiment.
- Symbol name: the name of the new symbol. For the half adder symbol, use the name `half_adder`.
- Symbol pins: the inputs and outputs of the symbol. For example, a half adder has two inputs and two outputs.

> *Xilinx ISE Symbol Library*
> In Xilinx ISE, when you create a new symbol, it becomes available for use in schematic designs through the symbol library. You will find your new symbol in a library named after your project.

4.2. Build a Full Adder using Half Adder Symbols

1. Create a new schematic design file, named `full_adder`.
2. Implements a full adder using two half adders based on the schematic shown in the Logic Diagram of a Full Adder figure. Use the half adder symbol that you created in the previous task.
3. Define the inputs and outputs using I/O markers
   - Name the inputs `a`, `b`, and `c_in`.
   - Name the outputs `s` and `c_out`.
4. Verify that your full adder works correctly through simulation.

4.3. Create a Full Adder Symbol

Create a new symbol named `full_adder` the same way you created your `half_adder` symbol.

4.4. Build a 4-Bit Adder Using Full Adder Symbols

1. Create a new schematic design file named `four_bit_adder`.
2. Implement a 4-bit adder using four full adders based on the schematic shown in the Logic Diagram of a 4-Bit Adder figure. Use the full adder symbol that you created in the previous task.
3. Define the inputs and outputs.
   - Name the first operand inputs `a0` (the least significant bit), `a1`, `a2`, and `a3` (the most significant bit).
   - Similarly, name the second operand inputs `b0`, `b1`, `b2`, and `b3`.
   - Name the outputs `s0`, `s1`, `s2`, `s3`, and `c_out` (the most significant output bit).
4. Connect the first carry-in bit, which is the `c_in` input of the least significant full adder, to ground (gnd) to make it permanently zero.
5. Verify that your 4-bit adder works correctly through simulation.
6. Verify that your 4-bit adder works correctly on your FGPA board.
   - Choose the switches and LEDs that you will use to control the inputs and observe the outputs.
   - Find out the pin numbers of your chosen switches and LEDs and assign them to the input and output ports of your design.
   - Program the FPGA
   - Verify that your circuit is working correctly on hardware.

4.5. Model the Half Adder, Full Adder, and 4-Bit Adder in Verilog

1. Model the half adder using the following Verilog model:

```
module half_adder (output c, s, input a, b);
    xor (s, a, b);
    and (c, a, b);
endmodule
```

2. Model the full adder using the following Verilog model:

```
module full_adder (output c_out, s, input a, b, c_in);
    half_adder h1 (c1, s1, a , b);
    half_adder h2 (c2, s , s1, c_in);
    or (c_out, c1, c2);
endmodule
```

3. Verify the correct functionality of the full adder through simulation.
4. Model the 4-bit adder shown in the Logic Diagram of a 4-Bit Adder figure using Verilog.
5. Verify the correct functionality of your 4-bit adder Verilog model using simulation.

5. Grading Sheet

Task	Points
Build and verify a half adder circuit	15
Build a full adder using half adder symbols	15

Task	Points
Build a 4-bit adder using full adder symbols	15
Model and verify the half adder, full adder, and 4-bit adder in Verilog	30
Lab notebook and discussion	25

Version 151
Last updated 2016-02-06 02:38:45 AST