Laboratory Exercise 6

Adders, Subtractors, and Multipliers

The purpose of this exercise is to examine arithmetic circuits that add, subtract, and multiply numbers. Each circuit will be described in Verilog and implemented on an Altera DE1 board.

Part I

Consider again the four-bit ripple-carry adder circuit used in lab exercise 2; its diagram is reproduced in Figure 1.

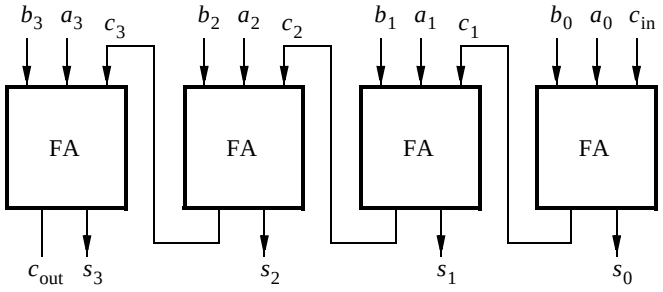

Figure 1: A four-bit ripple carry adder.

This circuit can be implemented using a '+' sign in Verilog. For example, the following code fragment adds *n*-bit numbers A and B to produce outputs sum and $carry$:

```
wire [n-1:0] sum;
wire carry;
...
assign {carry, sum} = A + B;
```

Use this construct to implement a circuit shown in Figure 2.

Design and compile your circuit with Quartus II software, download it onto a DE1 board, and test its operation as follows:

1. Create a new Quartus II project. Select as the target chip the Cyclone II EP2C20F484C7, which is the FPGA chip on the Altera DE1 board. Select Cyclone II EP2C20F484C7 device to implement the designed circuit on the DE1 board.
2. Write Verilog code that describes the circuit in Figure 2.
3. Connect input A to switches SW_{7-0}, and use KEY_0 as an active-low asynchronous reset and KEY_1 as a manual clock input. The sum output should be displayed on red $LEDR_{7-0}$ lights and the carry-out should be displayed on the red $LEDR_0$ light.
4. Assign the pins on the FPGA to connect to the switches and 7-segment displays by importing the *DE1_pin_assignments.qsf* file.
5. Compile your design and use timing simulation to verify the correct operation of the circuit. Once the simulation works properly, download the circuit onto the DE1 board and test it by using different values of A. Be sure to check that the *Overflow* output works correctly.

6. Open the Quartus II Compilation Report and examine the results reported by the Timing Analyzer. What is the maximum operation frequency, f_{max}, of your circuit? What is the longest path in the circuit in terms of delay?

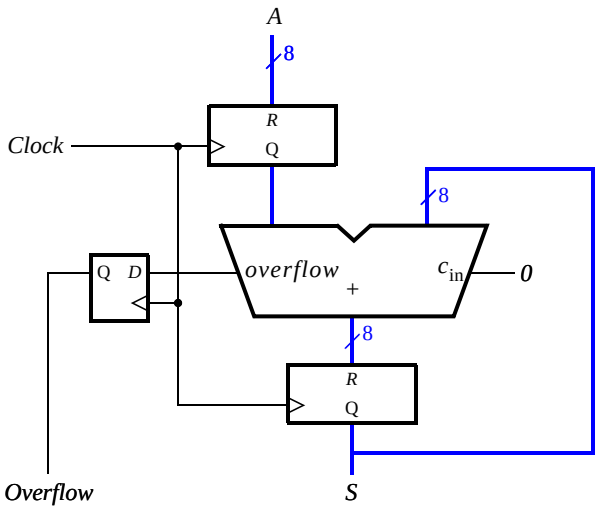

Figure 2: An eight-bit accumulator circuit.

Part II

Extend the circuit from Part I to be able to both add and subtract numbers. To do so, add an *add_sub* input to your circuit. When *add_sub* is 1, your circuit should subtract A from S, and add A and S as in Part I otherwise.

Part III

Figure 3a gives an example of paper-and-pencil multiplication $P = A \times B$, where $A = 11$ and $B = 12$.

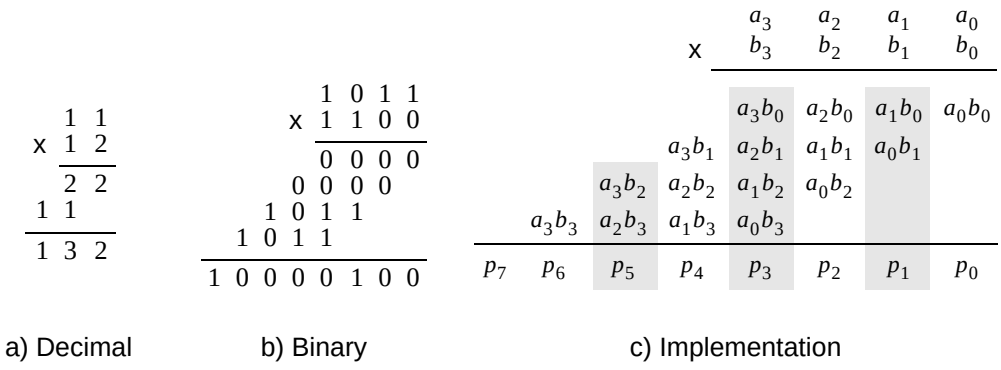

Figure 3: Multiplication of binary numbers.

We compute $P = A \times B$ as an addition of summands. The first summand is equal to A times the ones digit of B. The second summand is A times the tens digit of B, shifted one position to the left. We add the two summands to form the product $P = 132$.

Part b of the figure shows the same example using four-bit binary numbers. To compute $P = A \times B$, we first form summands by multiplying A by each digit of B. Since each digit of B is either 1 or 0, the summands are either shifted versions of A or 0000. Figure 3c shows how each summand can be formed by using the Boolean AND operation of A with the appropriate bit of B.

A four-bit circuit that implements $P = A \times B$ is illustrated in Figure 4. Because of its regular structure, this type of multiplier circuit is called an *array multiplier*. The shaded areas correspond to the shaded columns in Figure 3*c*. In each row of the multiplier AND gates are used to produce the summands, and full adder modules are used to generate the required sums.

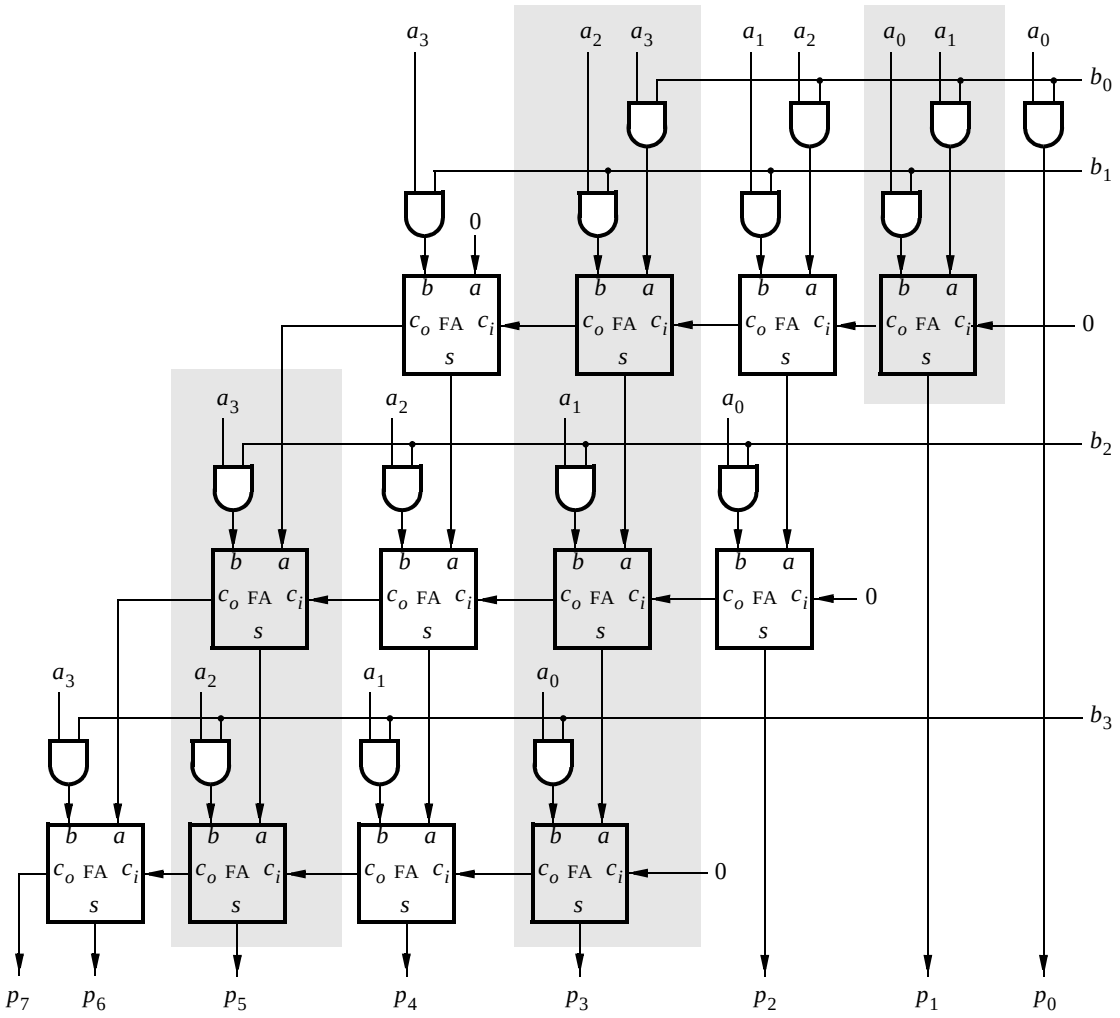

Figure 4: An array multiplier circuit.

Perform the following steps to implement the array multiplier circuit:

1. Create a new Quartus II project to implement the desired circuit on the Altera DE1 board.
2. Generate the required Verilog file, include it in your project, and compile the circuit.
3. Use functional simulation to verify your design.
4. Augment your design to use switches SW_{11-8} to represent the number A and switches SW_{3-0} to represent B. The hexadecimal values of A and B are to be displayed on the 7-segment displays *HEX6* and *HEX4*, respectively. The result $P = A \times B$ is to be displayed on *HEX1* and *HEX0*.
5. Assign the pins on the FPGA to connect to the switches and 7-segment displays by importing the *DE1_pin_assignments.qsf* file.

6. Recompile the circuit and download it into the FPGA chip.

7. Test the functionality of your circuit by toggling the switches and observing the 7-segment displays.

Part IV

In Part III, an array multiplier was implemented using full adder modules. At a higher level, a row of full adders functions as an n-bit adder and the array multiplier circuit can be represented as shown in Figure 5.

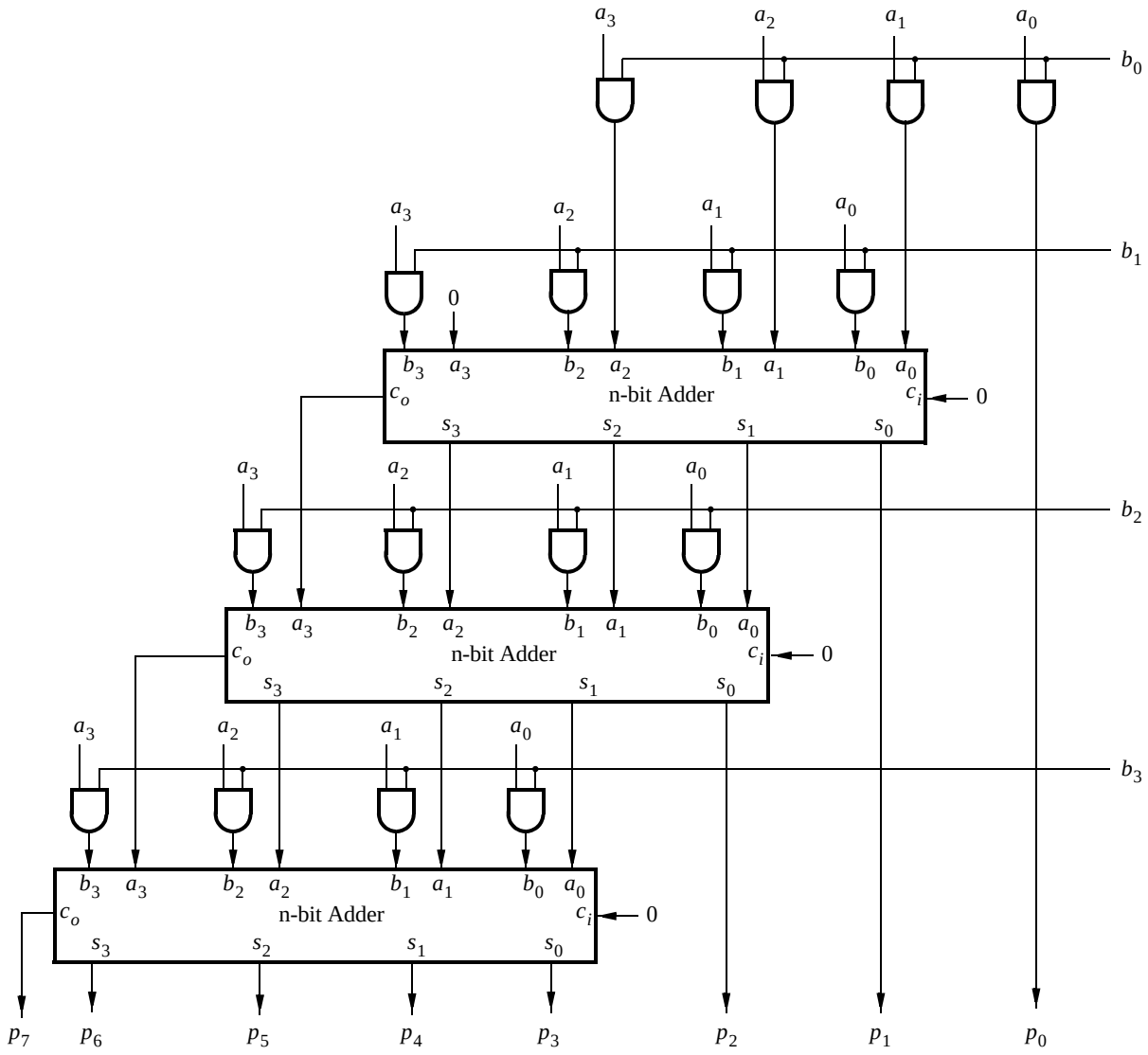

Figure 5: An array multiplier implemented using n-bit adders.

Each n-bit adder adds a shifted version of A for a given row and the partial sum of the row above. Abstracting the multiplier circuit as a sequence of additions allows us to build larger multipliers. The multiplier should consist of n-bit adders arranged in a structure shown in Figure 5. Use this approach to implement a 5x5 multiplier circuit with registered inputs and outputs, as shown in Figure 6.

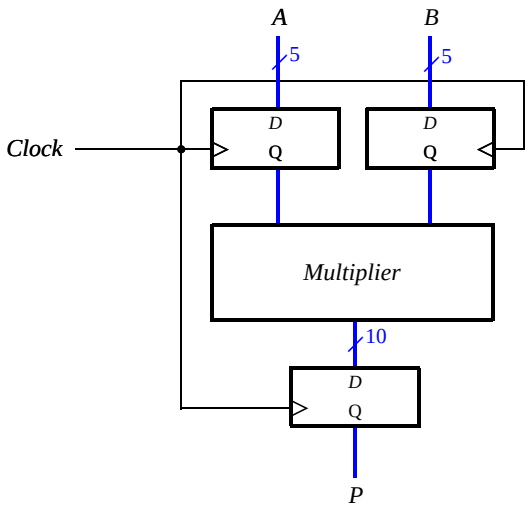

Figure 6: A registered multiplier circuit.

Perform the following steps:

1. Create a new Quartus II project.
2. Write the required Verilog file, include it in your project, and compile the circuit.
3. Use functional simulation to verify your design.
4. Augment your design to use switches SW_{9-5} to represent the number A and switches SW_{4-0} to represent B. The result $P = A \times B$ is to be displayed on *HEX3-0*.
5. Assign the pins on the FPGA to connect to the switches and 7-segment displays.
6. Recompile the circuit and download it into the FPGA chip.
7. Test the functionality of your design by toggling the switches and observing the 7-segment displays.
8. How large is the circuit in terms of the number of logic elements?
9. What is the f_{max} for this circuit?

Part V

Part IV showed how to implement multiplication $A \times B$ as a sequence of additions, by accumulating the shifted versions of A one row at a time. Another way to implement this circuit is to perform addition using an adder tree.

An adder tree is a method of adding several numbers together in a parallel fashion. This idea is illustrated in Figure 7. In the figure, numbers A, B, C, D, E, F, G, and H are added together in parallel. The addition $A + B$ happens simultaneously with $C + D$, $E + F$ and $G + H$. The result of these operations are then added in parallel again, until the final sum P is computed.

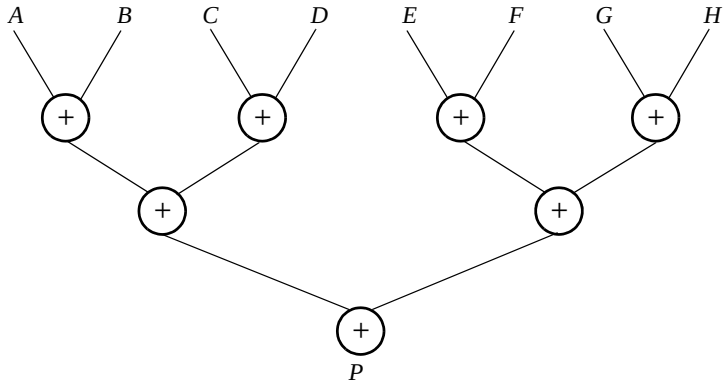

Figure 7: An example of adding 8 numbers using an adder tree.

In this part you are to implement a 5x5 array multiplier that computes $P = A \times B$. Use an adder tree structure to implement operations shown in Figure 5. Inputs A and B, as well as the output P should be registered as in Part IV. What is the f_{max} for this circuit?

Preparation

The recommended preparation for this laboratory exercise includes Verilog code for Parts I through V.