Laboratory Exercise 10

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 Intel FPGA DE10-Lite, DE0-CV, DE1-SoC, or DE2-115 board.

Part I

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

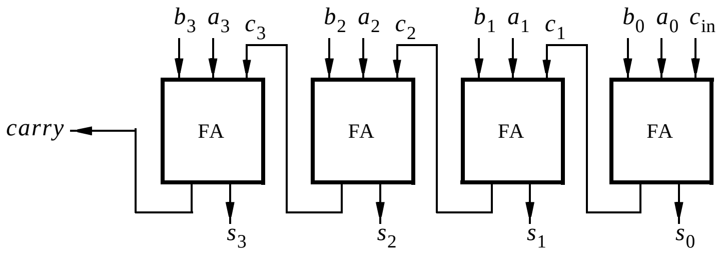

Figure 1: A four-bit ripple carry adder.

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. This circuit, which is often called an *accumulator*, is used to add the value of an input A to itself repeatedly. The circuit includes a carry out from the adder, as well as an *overflow* output signal. If the input A is considered as a 2's-complement number, then *overflow* should be set to 1 in the case where the output *sum* produced does not represent a correct 2's-complement result.

Perform the following steps:

1. Create a new Quartus project. Write Verilog code that describes the circuit in Figure 2.
2. Connect input A to switches SW_{7-0}, use KEY_0 as an active-low asynchronous reset, and use KEY_1 as a manual clock input. The sum from the adder should be displayed on the red lights $LEDR_{7-0}$, the registered carry signal should be displayed on $LEDR_8$, and the registered *overflow* signal should be displayed on $LEDR_9$. Show the registered values of A and S as hexadecimal numbers on the 7-segment displays HEX3 − 2 and HEX1 − 0.
3. Make the necessary pin assignments needed to implement the circuit on your DE-series board, and compile the circuit.
4. Use timing simulation to verify the correct operation of the circuit. Once the simulation works properly, download the circuit onto your DE-series board and test it by using different values of A. Be sure to check that the *overflow* output works correctly.

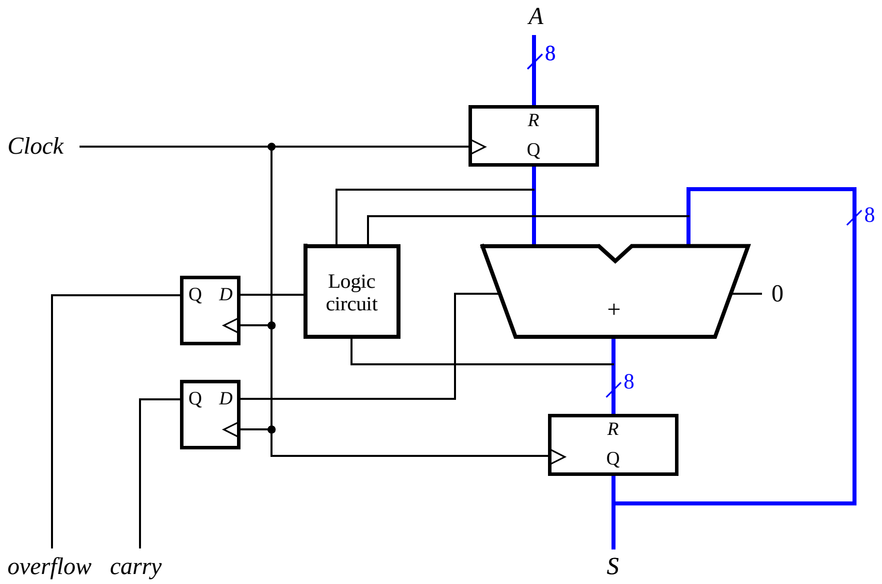

Figure 2: An eight-bit accumulator circuit.

Part II

Extend the circuit from Part I to be able to both add and subtract numbers. To do so, introduce an *add_sub* input to your circuit. When *add_sub* is 1, your circuit should subtract A from S, and when *add_sub* is 0 your circuit should add A to S as in Part I.

Part III

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

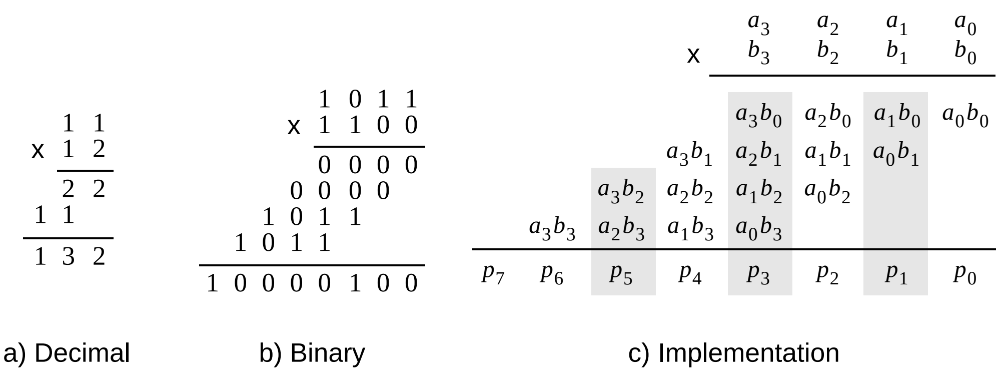

Figure 3: Multiplication of binary numbers.

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 3c. 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.

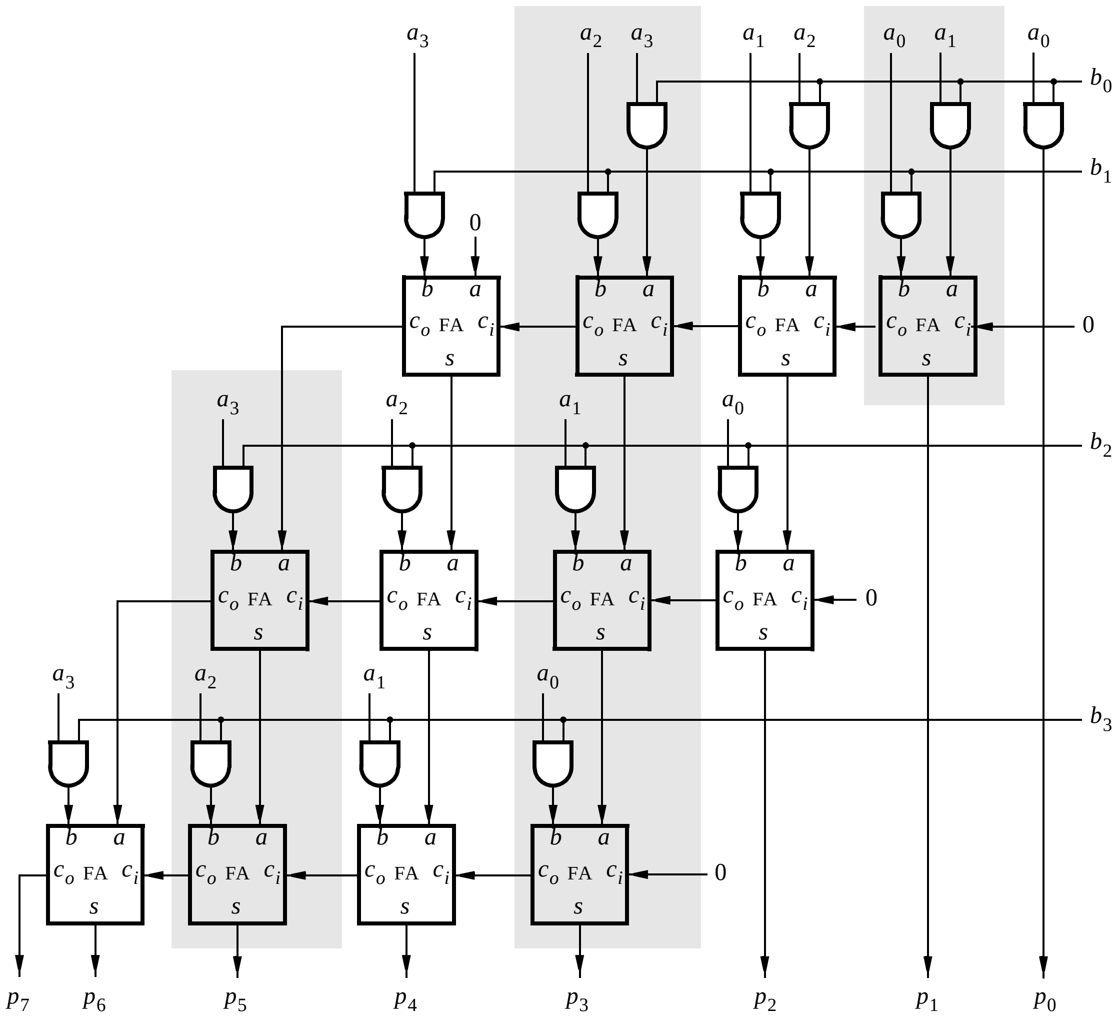

Figure 4: An array multiplier circuit.

Perform the following steps to implement the array multiplier circuit:

1. Create a new Quartus project.
2. Generate the required Verilog file. Use switches SW_{7-4} 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 $HEX2$ and $HEX0$, respectively. The result $P = A \times B$ is to be displayed on $HEX5-4$.
3. Make the necessary pin assignments needed to implement the circuit on your DE-series board, and compile the circuit.
4. Use simulation to verify your design.
5. Download your circuit onto your DE-series board and test its functionality.