Pre Laboratory 3:
ADDERS, SUBTRACTORS AND MULTIPLIERS

OBJECTIVES

➢ The purpose of this lab is to learn how to connect simple input (switches) and output devices (LEDs and 7-segment) to an FPGA chip and implement a circuit that uses these devices.

➢ Examine arithmetic circuits that add, subtract, and multiply numbers.

PREPARATION FOR LAB 3

➢ Students have to simulate all the exercises in Pre Lab 3 at home. All results (codes, waveform, RTL viewer, …) have to be captured and submitted to instructors prior to the lab session. *If not, students will not participate in the lab and be considered absent this session.*

REFERENCE

1. Intel FPGA training

Pre Laboratory 3:

ADDERS, SUBTRACTORS AND MULTIPLIERS

EXERCISE 1:

Objective: Known how to program an adder using '+' sign in VHDL.

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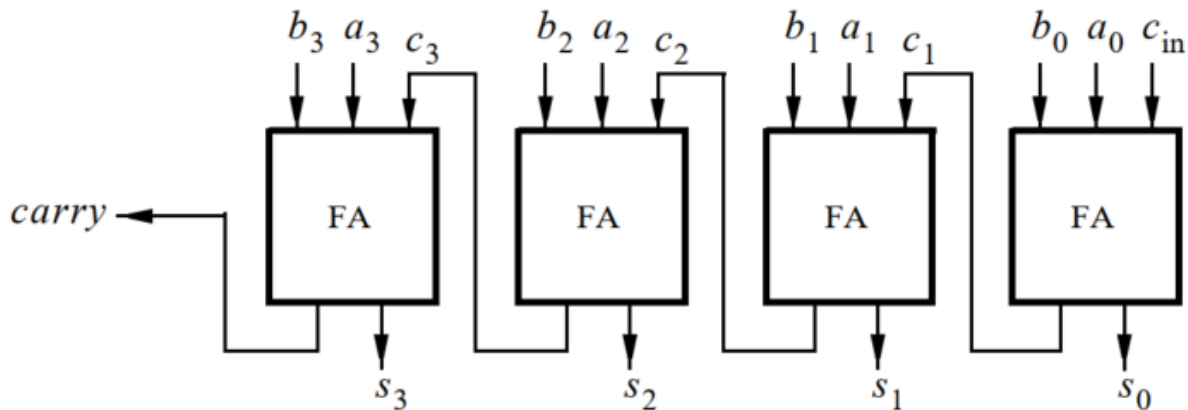

Figure 1: A four-bit ripple carry adder.

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

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_signed.all;
. . .
SIGNAL sum : STD_LOGIC_VECTOR(n-1 DOWNTO 0);
. . .
sum <= A + B;
```

Modify the code above to implement an 8-bit carry adder.

Using waveform tool, test the functionality of your design by inputting various data values and observing the generated outputs.

Pre Laboratory 3:

ADDERS, SUBTRACTORS AND MULTIPLIERS

Check: Your report has to show two results:

> ➢ The waveform to prove the circuit works correctly.

> ➢ The result of RTL viewer.

ADDERS, SUBTRACTORS AND MULTIPLIERS

EXERCISE 2

__Objective:__ Known how to program a multiplier.

__Requirement:__ Figure 2a gives an example of paper-and-pencil multiplication $P = A \times B$, where $A = 11$ and $B = 12$.

		a_3	a_2	a_1	a_0
	$\times$	b_3	b_2	b_1	b_0

```
        1 1                1 0 1 1                          a₃   a₂   a₁   a₀
     x  1 2             x  1 1 0 0                       x  b₃   b₂   b₁   b₀
     ───────           ───────────
        2 2                0 0 0 0                      a₃b₀ a₂b₀ a₁b₀ a₀b₀
      1 1                0 0 0 0                   a₃b₁ a₂b₁ a₁b₁ a₀b₁
     ───────          1 0 1 1                a₃b₂ a₂b₂ a₁b₂ a₀b₂
      1 3 2          1 0 1 1            a₃b₃ a₂b₃ a₁b₃ a₀b₃
                    ─────────────    ──────────────────────────────────
                    1 0 0 0 0 1 0 0   P₇  P₆   P₅   P₄   P₃   P₂   P₁   P₀

       (a) Decimal          (b) Binary                 (c) Implementation
```
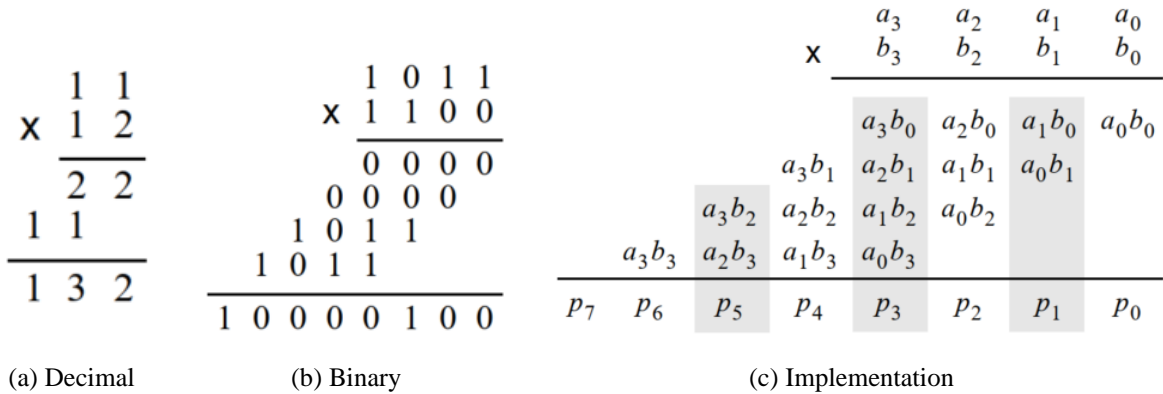
__Figure 2:__ Multiplication of binary numbers.

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 2c 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 3. 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 2c. 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.
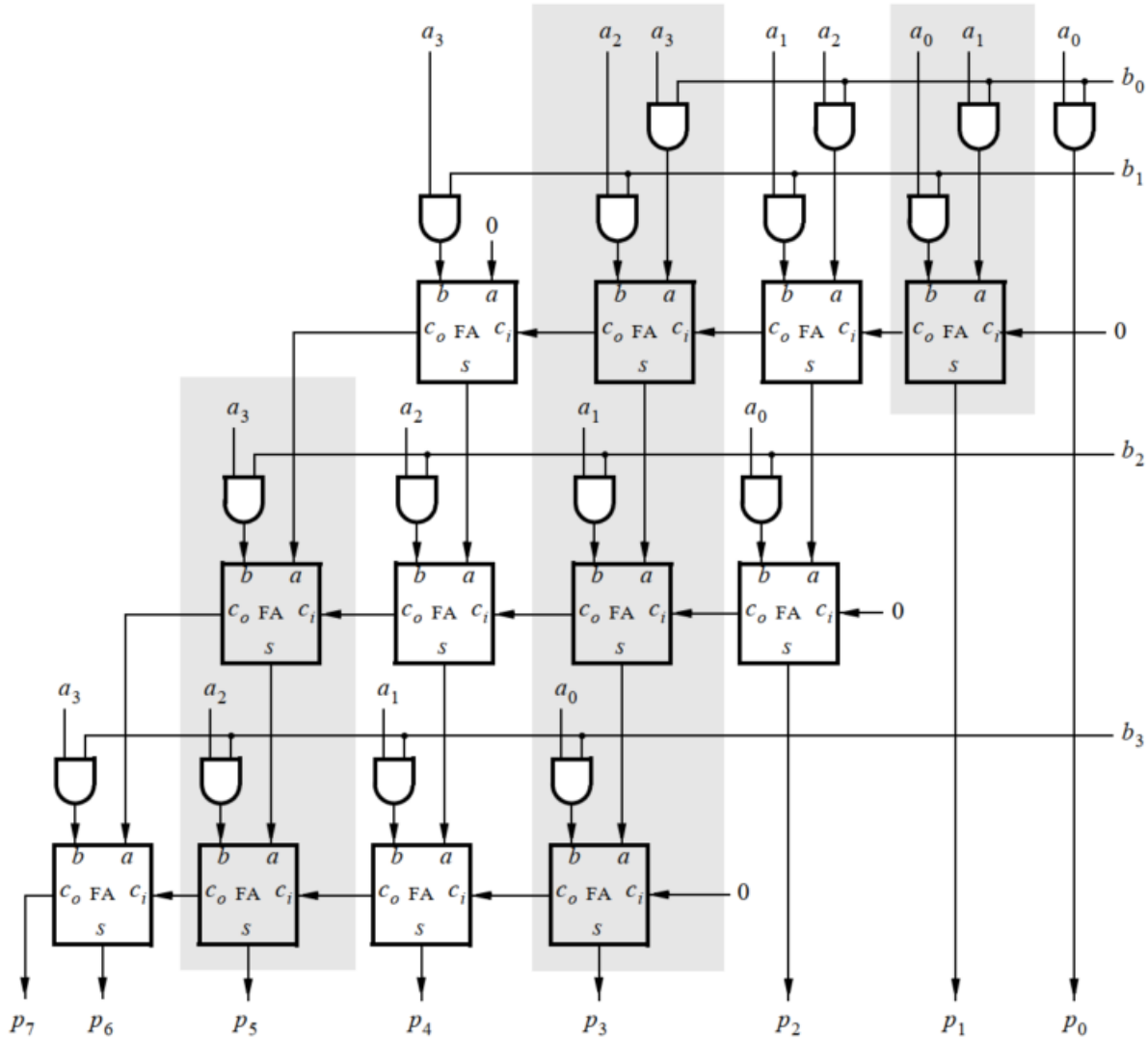
Figure 3: An array multiplier circuit.

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 4. Each *n*-bit adder adds a shifted version of *A* for a given row and the *partial product* 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 4.
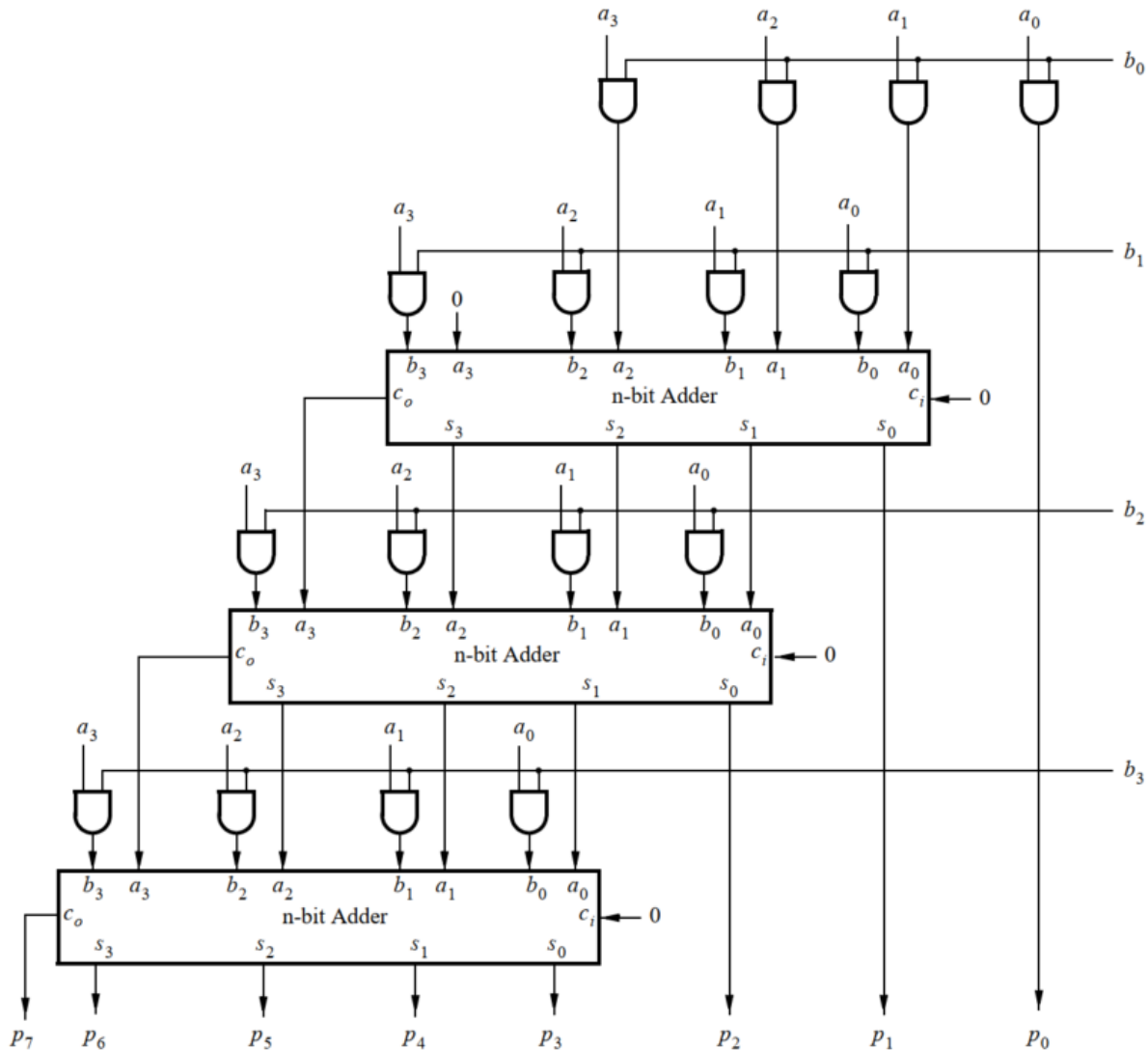
Figure 4: An array multiplier implemented using n-bit adders.

Use this approach, write VHDL code to implement **an 8 x 8 multiplier circuit**.

Instruction:

 ➢ Generate a VHDL file with the code above and include it in the project.

 ➢ Compile the code. Use the Quartus RTL Viewer tool to examine the gate-level circuit produced from the code, then verify that the latch is implemented as shown in Figure 3 and 4.

Pre Laboratory 3:

ADDERS, SUBTRACTORS AND MULTIPLIERS

➢ Simulate the behavior of your VHDL code by using the simulation feature provided in the Quartus software. Test the functionality of your design by inputting various data values and observing the generated outputs.

Check: Your report has to show two results:

➢ The waveform to prove the circuit works correctly.

➢ The result of RTL viewer.