Laboratory 1:

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 1

- Finish Pre Lab 1 at home.
- Students have to simulate all the exercises in Pre Lab 1 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

Laboratory 1:

ADDERS, SUBTRACTORS AND MULTIPLIERS

EXPERIMENT 1

Objective: Known how to program a system to add the value of an input A to itself repeatedly.

Requirement: The circuit in Figure 1, 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.

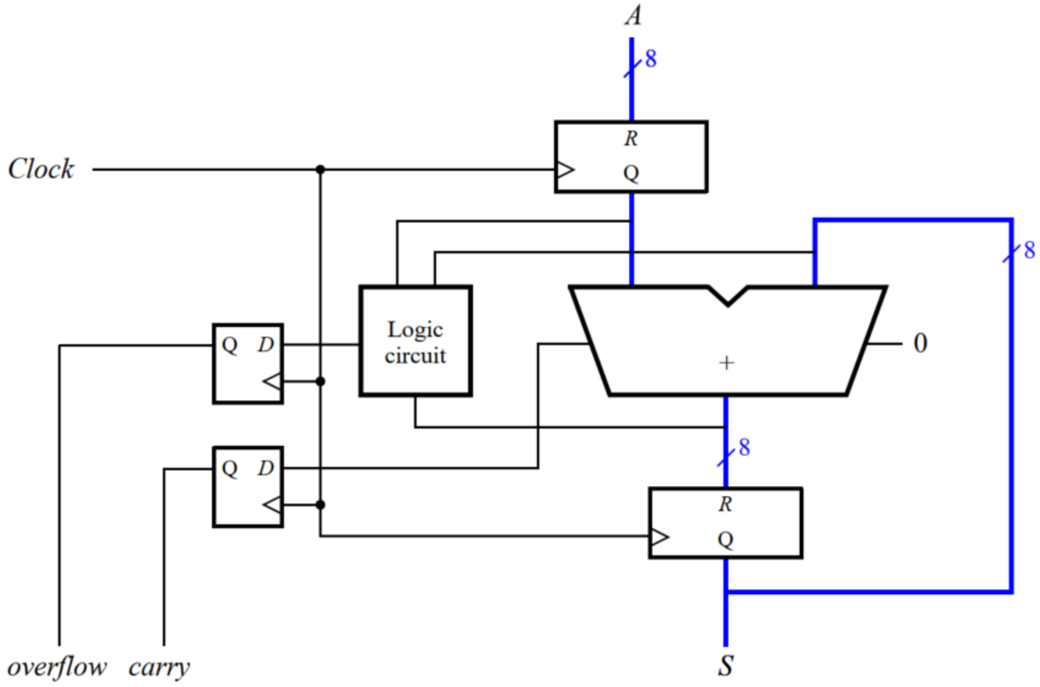


Figure 1: An eight-bit accumulator circuit.

Instruction:

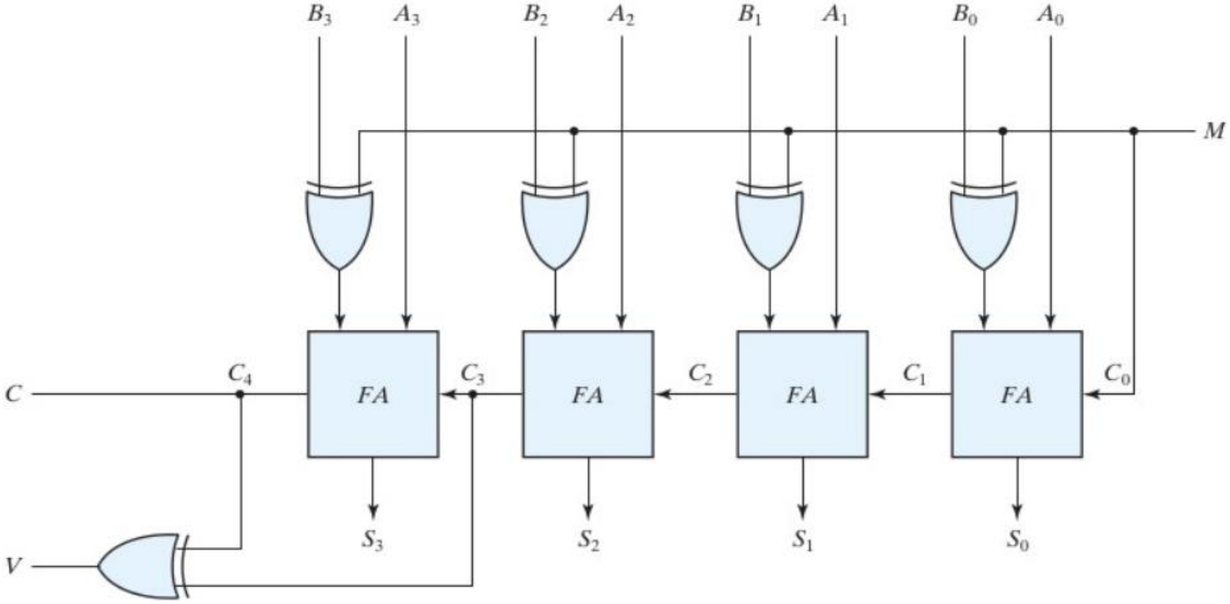
- Write System Verilog code to describe 4-bit adder circuit.
- There are two ways to detect overflow for 2's complement:

  - Compare inputs and outputs: The inputs and outputs always have the same sign. If not, the overflow must be set.

  - Compare carry bits: The circuit for checking overflow of 4-bit adder is shown below.

Laboratory 1:

ADDERS, SUBTRACTORS AND MULTIPLIERS



In the figure, V is the overflow flag.

- Write System Verilog code to describe the circuit given in Figure 1.
- Connect input A to switches SW7−0, use KEY0 as an active-low asynchronous reset, and use KEY1 as a manual clock input. The sum from the adder should be displayed on the red lights LEDR7−0, the registered carry signal should be displayed on LEDR8, and the registered overflow signal should be displayed on LEDR9. Show the registered values of A and S as hexadecimal numbers on the 7-segment displays HEX3−2 and HEX1 − 0.

Check: Your report has to show two results:

- The waveform to prove the circuit works correctly. You must show all the cases of overflow flag and carry flag.
- The result of RTL viewer.

Laboratory 1:

ADDERS, SUBTRACTORS AND MULTIPLIERS

EXPERIMENT 2

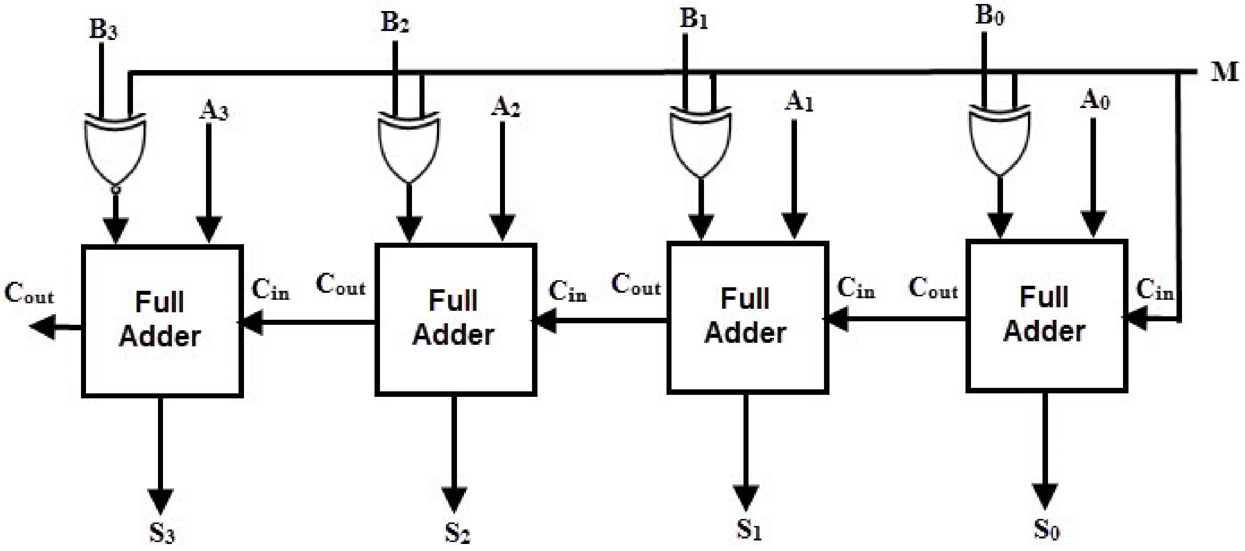
Objective: Known how to program a system to add or subtract the value of an input A to itself repeatedly.

Requirement: Extend the circuit from Experiment 1 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 register *S*, and when *add_sub* is 0 your circuit should add *A* to register *S* as in experiment 1.

Instruction:

- There are two ways to implement that exercise:

  - If the 4-bit adder circuit is constructed from four instances of the FA, using the example 4-bit circuit:



  - If the 4-bit adder entity is written by using a '+' sign in System Verilog, you need to modify the circuit using IF_ ELSE statement.

- Connect input A to switches SW7−0, input *add_sub* to switch 9. Use KEY0 as an active-low asynchronous reset, and use KEY1 as a manual clock input. The sum from the adder should be displayed on the red lights LEDR7−0, the registered carry signal should be displayed on LEDR8, and the registered overflow signal should be displayed on LEDR9.

Show the registered values of A and S as hexadecimal numbers on the 7-segment displays HEX3−2 and HEX1 − 0.

Check: Your report has to show two results:

- The waveform to prove the circuit works correctly. You must show all the cases of overflow flag and carry flag.
- The result of RTL viewer.

Laboratory 1:

ADDERS, SUBTRACTORS AND MULTIPLIERS

EXPERIMENT 3

Objective: Known how to program 8x8 multiplier circuit with registered inputs and outputs.

Requirement: The circuit in Figure 2 shows an 8 x 8 multiplier circuit with registered inputs and outputs. Write System Verilog code to implement this system.

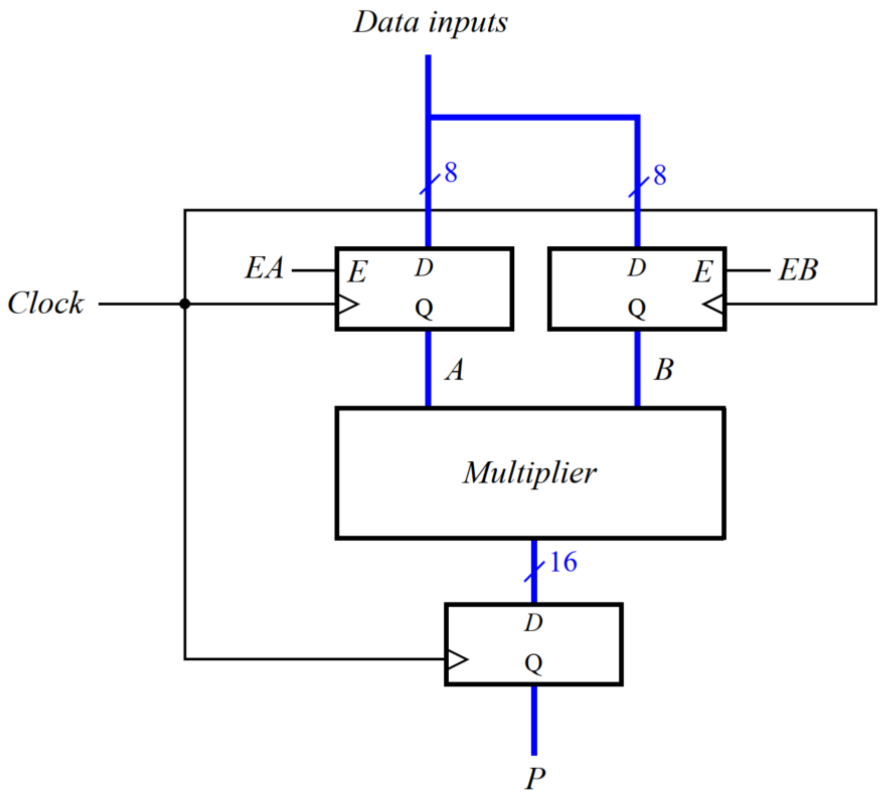


Figure 2: A registered multiplier circuit.

Instruction:

- In exercise 2, pre lab 3, you have write the multiplier. Write System Verilog module that instantiates the instances of this multiplier and three registers to describe the circuit given in Figure 2.
- Use switches SW7−0 to provide the data inputs to the circuit. Use SW9 as the enable signal EA for register A, and use SW8 as the enable for register B. When SW9 = 1 display the contents of register A on the red lights LEDR, and display the contents of register B on

these lights when SW8 = 1. Use KEY0 as a synchronous reset input, and use KEY1 as a manual clock signal. Show the product P = A×B as a hexadecimal number on the 7-segment displays HEX3-0.

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

Check: Your report has to show two results:

- The waveform to prove the circuit works correctly.
- The result of RTL viewer.

Laboratory 1:

ADDERS, SUBTRACTORS AND MULTIPLIERS

EXPERIMENT 4

Objective: Known how to program ALU circuit with registered inputs and outputs.

Requirement: The circuit in Figure 3 shows an ALU circuit with registered inputs and outputs. The ALU can implement add, subtract and multiply depending on *Sel* input which is shown in Table 1. Write System Verilog code to implement this system.

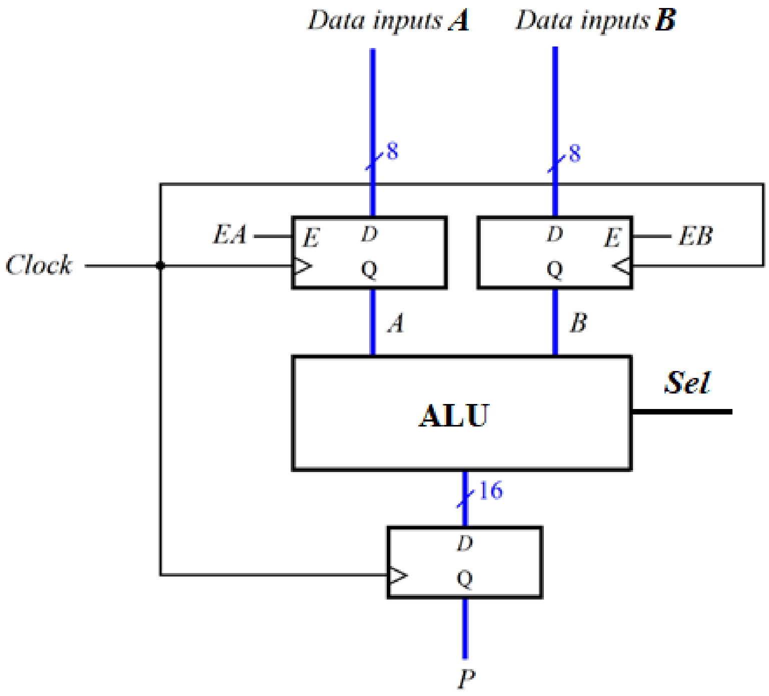


Figure 3: A registered ALU circuit.

Sel Input	ALU operator
000	Add
001	Subtract
010	Multiply

Table 1: Operation of ALU circuit.

Instruction: This circuit has a lot of inputs (22 inputs) then we can not implement this circuit on DE10 standard. Only test the functionality of your design by inputting various data values and observing the generated outputs.

Laboratory 1:

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.