Lab 5: 4-bit ALU and BCD

Goals

- The objective of this lab is to implement a 4-bit ALU using the previous knowledge and labs we have used so far.
- The goal is that the student is familiarized with how an ALU works and sub-block integration.

Introduction

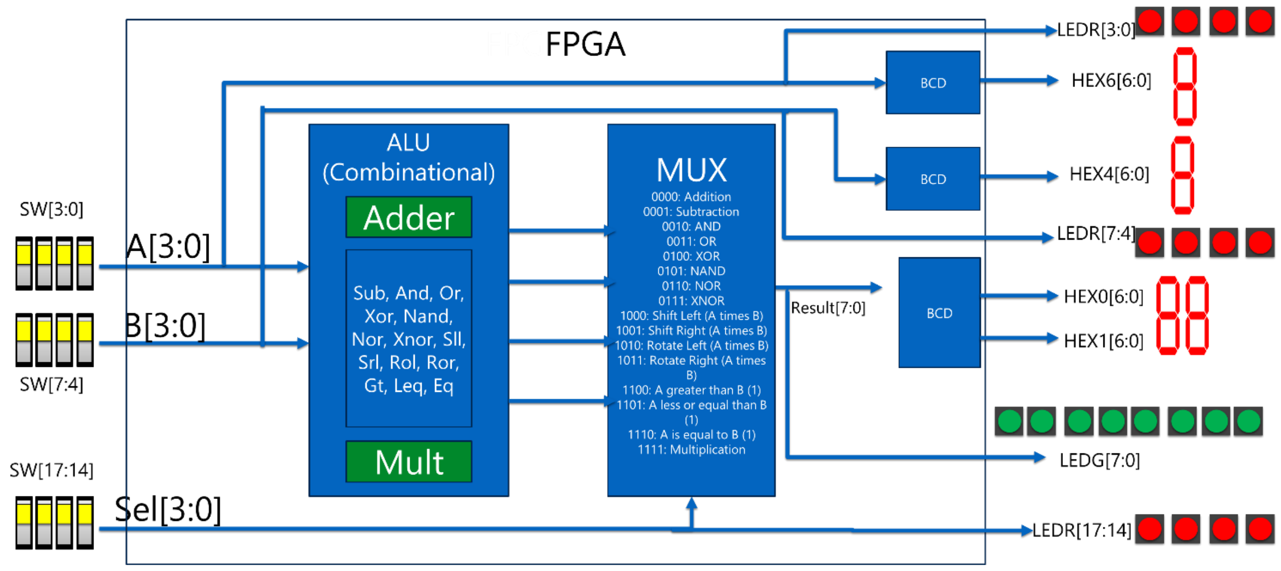
An Arithmetic Logic Unit (ALU) is a fundamental digital circuit within a computer's central processing unit (CPU) responsible for performing arithmetic and logical operations on data.

Lab description

Design and implement a 4-bit ALU using the previous labs.

Step 1.

Create a 4-bit ALU using the following diagram as a base:



You will utilize the lab 4 for the multiplier and the lab 2 for the binary adder, the other operations will be added to the ALU as procedural with operators as needed. You can also opt to make the ALU purely behavioral (via a procedural block and case for example).

The ALU will implement all the following operations and will output based on sel signal:

- 0000: Addition
- 0001: Subtraction
- 0010: AND
- 0011: OR
- 0100: XOR

- 0101: NAND
- 0110: NOR
- 0111: XNOR
- 1000: Shift Left (A times B)
- 1001: Shift Right (A times B)
- 1010: Rotate Left (A times B)
- 1011: Rotate Right (A times B)
- 1100: A greater than B (if condition is met, output will be 1)
- 1101: A less or equal than B (if condition is met, output will be 1)
- 1110: A is equal to B (if condition is met, output will be 1)
- 1111: Multiplication

Step 2.

Once you have the implementation you will create a testbench to test all operations and then implement it in FPGA and test combinations of the values to test out all the operations.

After you have the implementation of the FPGA, instance it on a top module to add the double dabble BCD circuit you did, for 4 bits the max value is 15 and for 8 bits is 255, so you will be able to fit that into the FPGA without problem and observe base-10 outputs.

You can also opt to use the 7segment decoder expanded to display HEX values instead from 0 to F on the 7segment, and it will mean you only need one 7segment to display inputs of 4-bits (0 to F) or for 8-bits (0 to FF) for the result, the choice is up to you. Here's an example of the pinout:

- Use the following **switches** for input A: **SW[3] to SW[0]**
- Use the following **switches** for input B: **SW[7] to SW[4]**
- Use the following **switches** for input SEL: **SW[17] to SW[14]**
- **For double dabble (BCD) + 7segment**
  - Use the following **7-segment** display for decimal input A: **HEX[7]**
  - Use the following **7-segment** display for decimal input A: **HEX[6]**
  - Use the following **7-segment** display for decimal input B: **HEX[5]**
  - Use the following **7-segment** display for decimal input B: **HEX[4]**
  - Use the following **7-segment** display for hexadecimal output result: **HEX[2] to HEX[0]**
- **For 7segment and HEX output from 0 to F**
  - Use the following **7-segment** display for decimal input A: **HEX[6]**
  - Use the following **7-segment** display for decimal input B: **HEX[4]**
  - Use the following **7-segment** display for hexadecimal output result: **HEX[1] to HEX[0]**
- Use the following green **leds** for display of the output in binary format: **LEDG[7] to LEDG[0]**
- Use the following reg **leds** for display binary format of input A: **LEDR[3] to LEDR[0]**
- Use the following reg **leds** for display binary format of input B: **LEDR[7] to LEDR[4]**

- Use the following reg **leds** for display binary format of input SEL: **LEDR[17] to LEDR[14]**

You can also implement both versions (double-dabble BCD and 7segment + 7segment with hexadecimal decodification) and add another switch to switch from hexadecimal to decimal for extra points.

Submission

Implementation description

Generate a document providing a detailed description of the implementation. This should include an explanation of design decisions, challenges encountered, and how they were overcome. Include diagrams, schematics, tables, or equations used in the design development.

Video:

Upload a short video to the Teams group, recording the operation of the implemented circuit on the FPGA and providing a brief description of its functionality.