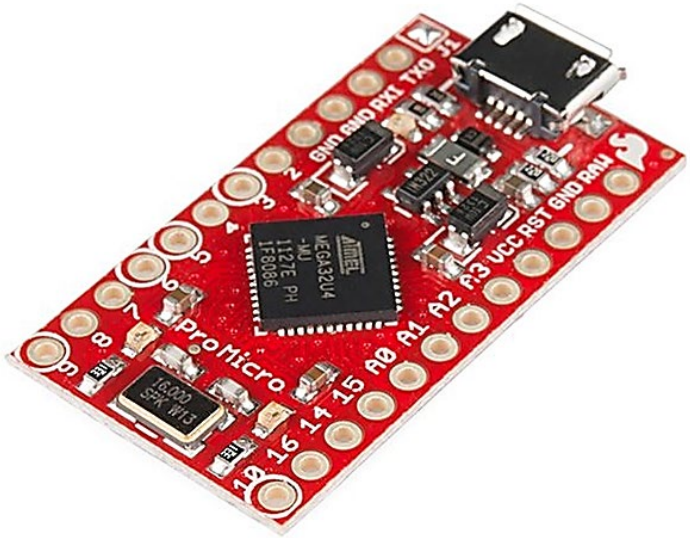
Electrical and Computer Engineering
Erik Jonsson School of Engineering & Computer Science
The University of Texas at Dallas
Dr. Tooraj Nikoubin



Project #1
(1) ALU
(2) Register File
(3) Multiplexors

EEDG/CE 6302
Microprocessors and Embedded Systems

Objectives:

- Designing and simulating synthesizable ALU (Arithmetic Logic Unit), Register File and required Multiplexors
- Getting more familiar with writing codes in hardware description languages (VHDL or Verilog)
- Writing testbench to test the written codes
- Assigning related ALU instructions to the previously specified Opcode

1. Introduction:

In this project, you will write HDL codes for other parts of the MCU, including: ALU, Register File and Multiplexors. Each of these modules has to be tested by a written testbench.

2. ALU

2.1. What is ALU?

Also called Function Unit or FU, ALU is one of the most important parts of your final design. As its name implies, it performs arithmetic and logic operations in the MCU. In additions, because of data path in our MCU project, it does have effects of many other instructions such as MOV or IN. Based on the previously given list of Opcodes, an ALU structure is different from one to another.

2.2. How it works?

In this project, ALU is a pure combinational module, i.e., it does not need clock signal. It only performs the operations which are dedicated by the instruction. There are five input and six output terminals:

(1) Function Select (FS): This terminal tells the ALU which operation has to be done. For example, “0110” may mean ADD, or “0001” may mean MOVA. Selection the pair of FS-operation combination is arbitrary. You can choose your own selection.
(2) Shift (SH): This terminal, specifies the number of shifts in the related instructions such as shift and rotate.
(3) A: The terminal of first operand into ALU
(4) B: The terminal of second operand into ALU
(5) Input_port: There are two input ports: First one is used to capture data with “IN” instruction directly from physical port of the FPGA. Second one is used to read the keyboard with “INK” instruction.
(6) F: The terminal which represents the result of ALU operation
(7) Negative (N): This status terminal reflects the sign of the result. It equals to the leftmost bit of F.
(8) Zero (Z): This status terminal shows that F is zero or not. If “F==0” then “Z=1”.
(9) Carry (C): This status terminal equals to 1 if the result sum of A and B has a carry. Note that generation of carry bit is regardless of ADD operation, i.e., executing any operation can potentially change the C bit. It may happen both for add and subtract operations. You have to consider both cases in your design.

(10) Overflow (V): This status terminal indicates overflow in the performing the operation. Note that, like C, executing any operation can potentially change the V bit.
(11) Data_branch (D): This status terminal is used for branch prediction that you will write the related code in the future. It is the XOR value of V and N bits.

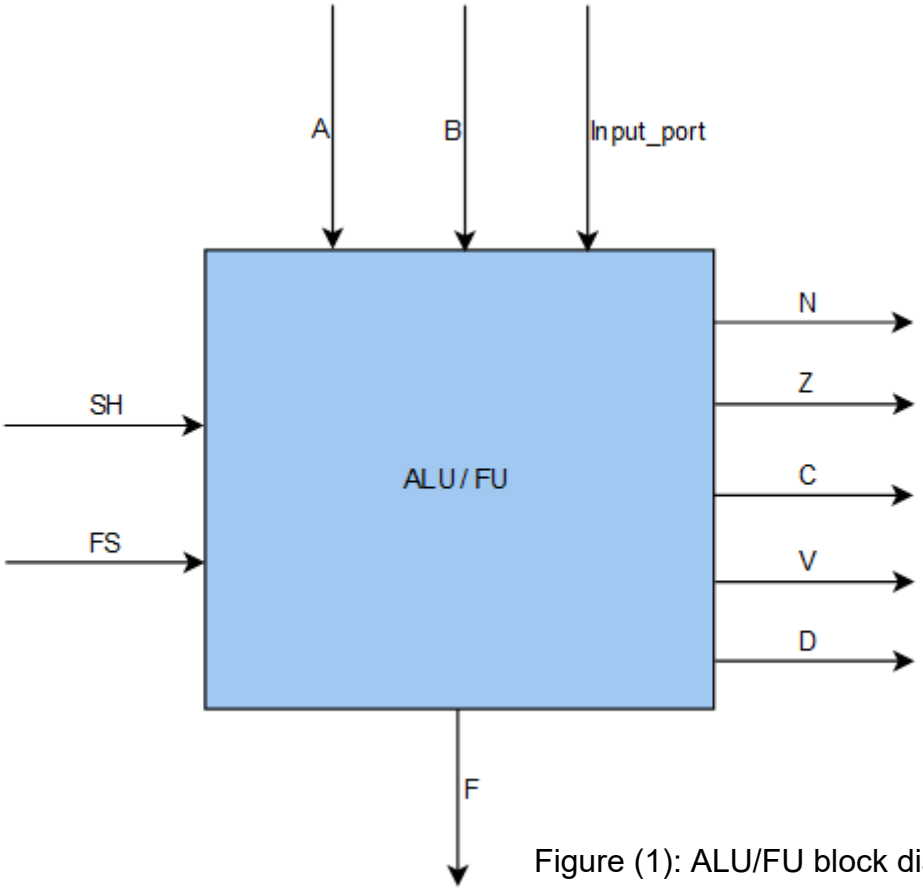


Figure (1): ALU/FU block diagram

2.3. Implementation:

Following table shows the width of each input and each output of the ALU/FU. Note that this module does not have a clock and combinational. Thus, your design must not contain any latch or flip-flop.

	Bit width
A, B	8
Input_port_data	8
Input_port_kb	8
FS	4
SH	3
F	8
C, Z, V, N, D	1

3. Register FIle:

The MCU has a Register File of 8×8-bit registers named from R0 to R7. R0 is always zero and permanently tied to “00000000”. Other registers can be read or written.

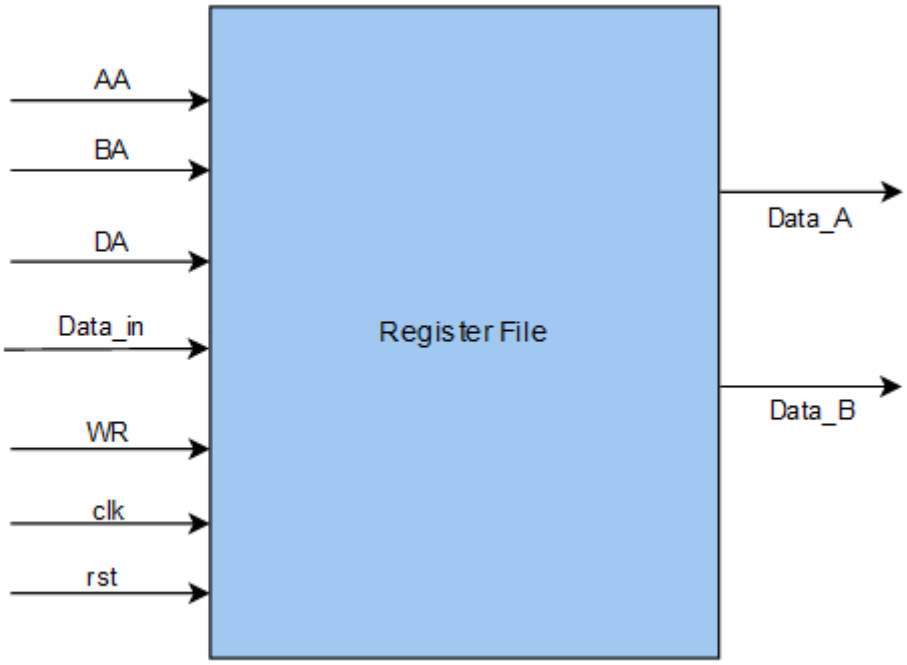


Figure (2): Register File block diagram

Following inputs and outputs are used in Register File:

	Bit width	Meaning
AA	3	Select a register to put data to Data_A output
BA	4	Select a register to put data to Data_B output
DA	3	Data in read from Data_in port and saved to the register to which DA is pointed
Data_in	8	Input data to be saved to a register
WR	1	If “WR==1”, the data will be written from Data_in to a specified register. If “WR==0”, data can only be read from Register File.
clk	1	clock
rst	1	Reset signal. If activated, all registers should be reset to zero
Data_A	8	The read data from R[AA]
Data_B	8	The read data from R[BA]

Note that R0 cannot be written. Your code should avoid this case by defining appropriate logic.

4. Multiplexors:

Your MCU requires multiple multiplexors (or Mux). Following multiplexors are available in the MCU:

	MUXA	MUXB	MUXC	MUXD
Number of input ports	2	2	4	3
Bit width of input ports	8	8	8	8
Bit width of output port	8	8	8	8

4.2. Implementation:

MUXA, MUXB, MUXC and MUXD are simple multiplexors. You just need to write the required HDL code. Please note that like ALU, a multiplexor is a combinational object. Thus, that this module does not have a clock and purely combinational, and your design must not contain any latch or flip-flop.

5. Project requirements:

- Write a code in an HDL for ALU, Register Files and Multiplexors.
- Write a testbench for each code. Verify and show both your coding works correctly using waveforms. Your testbench should include all conditions that may happen in normal operation of each module.
- Take a screenshot of the report of Xilinx ISE (or any other HDL synthesizers) which shows your ALU/FU and multiplexors do not contain any latch or flip-flop and insert the screenshot in your report.
- For testing multiplexors, just prepare testbench for MUXD.
- Each student should submit a report no more than 4 pages (excluding cover page, TOC, appendix, etc).
- Do not include your codes in the report. Submit your HDL codes along with their testbenches separately.
- Answer to the questions at the end of this manual in the report.

6. Questions:

1. There are two completely different ways to write ALU in HDLs: (1) using if-then-else template or (2) using switch-case (VHDL and Verilog) or with-select (only in VHDL) templates; and both are functionally correct. Why do you have to prevent using if-then-else in ALU coding? Explain briefly in 3 to 5 lines.
2. What does overflow I arithmetic operation mean? When does it happen? When do you use each of different overflows in your ALU design? Explain briefly in 2 to 3 lines.
3. Which template do you use for multiplexors? If-then-else or switch-case? Why?

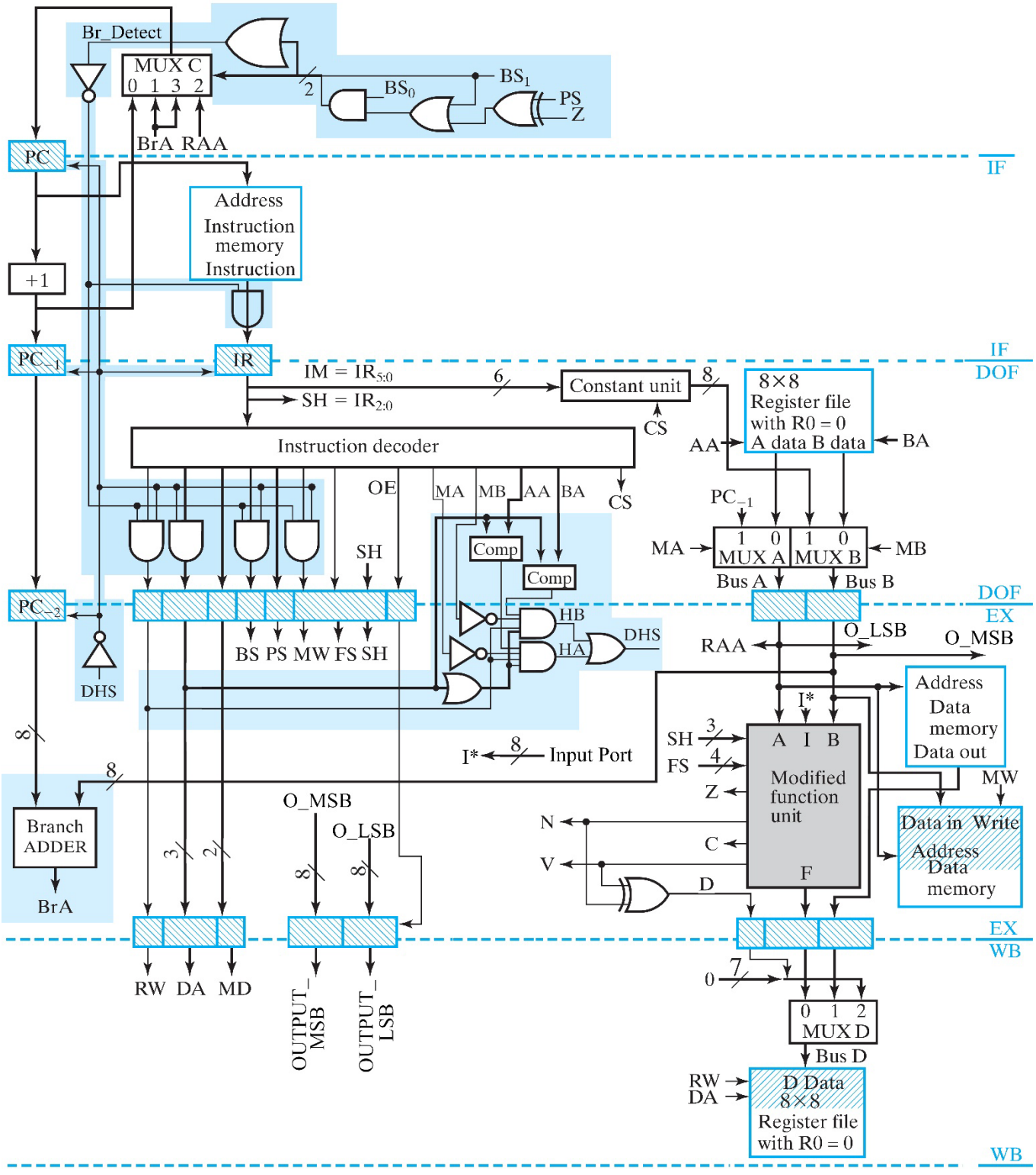


Figure (3): Block diagram of the MCU

Reference:

Mano, M. M. AND Kime, C. R. AND Martin M. "*Logic and Computer Design Fundamentals*", 5th ed. Hoboken, NJ: Pearson Higher Education Inc., 2015, pp 585-615.