

B38DB: Digital Design and Programming

ALU Examples

Mustafa Suphi Erden

Heriot-Watt University
School of Engineering & Physical Sciences
Electrical, Electronic and Computer Engineering
Room: EM 2.01
Phone: 0131-4514159
E-mail: m.s.erden@hw.ac.uk

Two ALU Examples from the Tutorial Document

“detailed_solution_tutorial_04.pdf”

- Example 1:

Design an ALU (Arithmetic and Logical Unit) with two 8-bit inputs A and B, and control signals x, y, and z. The ALU should support the operations described in the Table below. Use an 8-bit adder and an arithmetic/logic extender. Treat these as blocks with no internal structural details required.

Inputs			Operation
x	y	z	
0	0	0	$S = A - B$
0	0	1	$S = A + B$
0	1	0	$S = A * 8$
0	1	1	$S = A / 8$
1	0	0	$S = A \text{ NAND } B$ (bitwise NAND)
1	0	1	$S = A \text{ XOR } B$ (bitwise XOR)
1	1	0	$S = \text{Reverse } A$ (bit reversal)
1	1	1	$S = \text{NOT } A$ (bitwise complement)

Two ALU Examples from the Tutorial Document

“detailed_solution_tutorial_04.pdf”

- Solution 1:

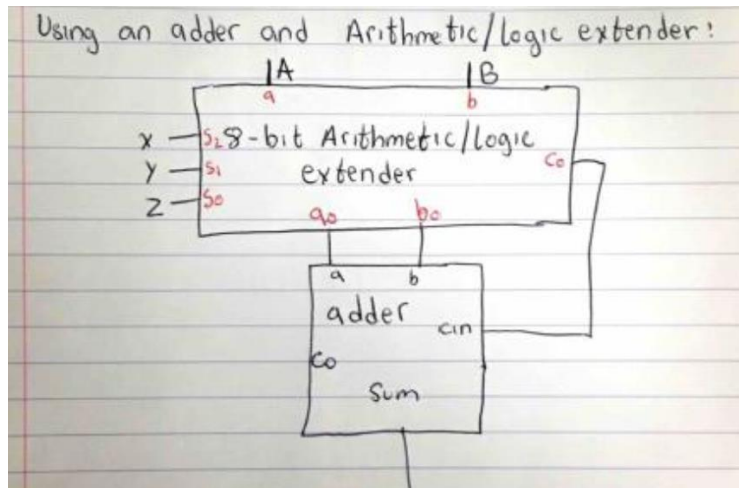
Inputs			Operation
x	y	z	
0	0	0	$S = A - B$
0	0	1	$S = A + B$
0	1	0	$S = A * 8$
0	1	1	$S = A / 8$
1	0	0	$S = A \text{ NAND } B$ (bitwise NAND)
1	0	1	$S = A \text{ XOR } B$ (bitwise XOR)
1	1	0	$S = \text{Reverse } A$ (bit reversal)
1	1	1	$S = \text{NOT } A$ (bitwise complement)

x y z	OPERATION
0 0 0	$A - B$
0 0 1	$A + B$
0 1 0	$A * 8$
0 1 1	$A \div 8$
1 0 0	$A \text{ NAND } B$
1 0 1	$A \text{ XOR } B$
1 1 0	REVERSE
1 1 1	NOT A

$A - B = A + B' + 1$

only functions related to adder

logical operations done outside adder



x y z	a0	b0	Co
0 0 0	a	b'	1
0 0 1	a	b	0
0 1 0	$a \ll 3$	0	0
0 1 1	$a \gg 3$	0	0
1 0 0	$a \text{ NAND } b$	0	0
1 0 1	$a \text{ XOR } b$	0	0
1 1 0	a reversed	0	0
1 1 1	NOT a	0	0

shifts a by 3 to left

shifts a by 3 to right

Two ALU Examples from the Tutorial Document

“detailed_solution_tutorial_04.pdf”

- Example 2:

Design a 3-bit Arithmetic Logic Unit (ALU) with two inputs (A, B), select signals (S1, S0), Output (O), and carry out (Cout). The ALU performs the following operations.

S1	S0	Operation
0	0	$F = A - B$
0	1	$F = A + B$
1	0	$F = A - B - 1$
1	1	$F = -A + B - 1$

Implement this ALU using a minimum number of 2-to-1 multiplexers, one single 3-bit adder (with a carry-in input), and a minimum number of additional gates. Overflow bit is NOT necessary to be considered.

Two ALU Examples from the Tutorial Document

“detailed_solution_tutorial_04.pdf”

- Solution 2:

S1	S0	Operation
0	0	$F = A - B$
0	1	$F = A + B$
1	0	$F = A - B - 1$
1	1	$F = -A + B - 1$

S1	S0	Operation
0	0	$A - B$
0	1	$A + B$
1	0	$A - B - 1$
1	1	$-A + B - 1$

For 00: $A - B = A + (-B)$
 $= A + B' + 1$

For 01: $A + B + 0$

For 10: $A - B - 1 = (A - B) - 1$
 $= A + B' + 1 - 1$
 $= A + B' + 0$

For 11: $-A + B - 1 = B - A - 1$
 $= B + (-A) - 1$
 $= B + A' + 1 - 1$
 $= A' + B + 0$

