13 | Combinatorial Logic Components, ALU

Combinational circuits: Gates and components connected with wires and/or busses
Component: Combinational circuit, generally shown as a block diagram
The others in the book are unmentioned otherwise.

Sign extend component: Expands a bus (group of parallel wires), preserving the sign, giving the binary number a two's-complement (represent the same #)

- Basically copies bit 15 of input to bits 16-31 in the output

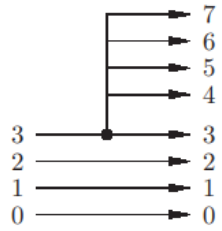


Figure 6.24: A sign-extend component for which the input is a 4-bit bus, and the output is an 8-bit bus, preserving the sign of the number

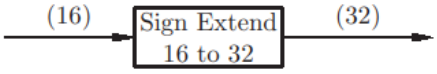


Figure 6.25: Block diagram of a 16-bit to 32-bit sign extend component

Multiplexer component: Has several input buses and one output bus; the width of the output bus is the same as each input bus; has control signals

- Control signals select one of input busses to be placed on the output bus (doesn't change the input)

- Example: If the control systems are 0010, then input bus number 2 is connected to the output bus (because 0010 is the binary value of decimal number 2)

- ***m* * *n*** multiplexer would have *m* input buses (width of each input bus and the output bus is *n* bits)
- Requires *c* control signals, $m = 2^c$ (due to 2 bit binary no.)
- Can be used to solve contradictions

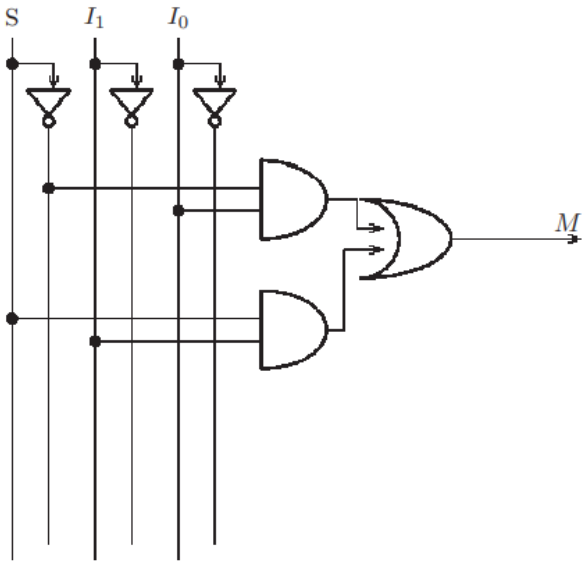


Figure 6.38: Logic diagram for a 2x1 multiplexer

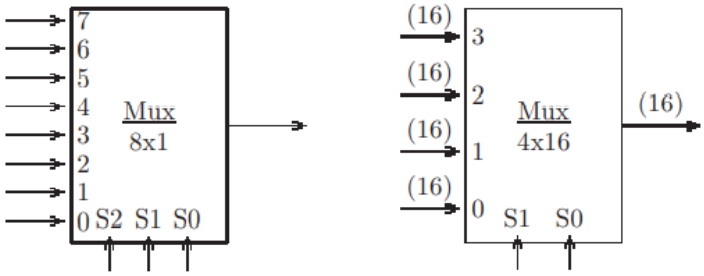


Figure 6.39: Block diagrams of an 8x1 multiplexer (3 control inputs), left, and a 4x16 multiplexer (2 control inputs), right

The above bottom left diagram shows 8 inputs (where each input is 1 bit). 2^3 is 8, so we have 3 control signals; S2, S1, and S0. (S stands for select here). If the three select inputs were 110, that would be 6, and thus input 6 would be chosen. (MUX is the abbreviation of multiplexer.)

The diagram next to that, each bus has 16 bits. You can see there are 4 inputs; thus, we need 2^2 = 4 inputs. Thus, 2 control signals (S1 and S0). If the two select inputs from the control signals were 01, input bus 1 would be chosen for the output.

Full adder:

3 inputs: x, y, c_{in} (each of these is 1 bit)
2 outputs: s (for sum), c_{out} (c stands here for carry-in, carry-out) (additionally, each has 1 bit)

Meanwhile, below is a truth table defining the sum and carry outputs of a full adder. Just thinking of adding the three bits, producing binary numbers as a result (for example, for the last row, 1 + 1 + 1 = 3 = 011 in binary).

x	y	c_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

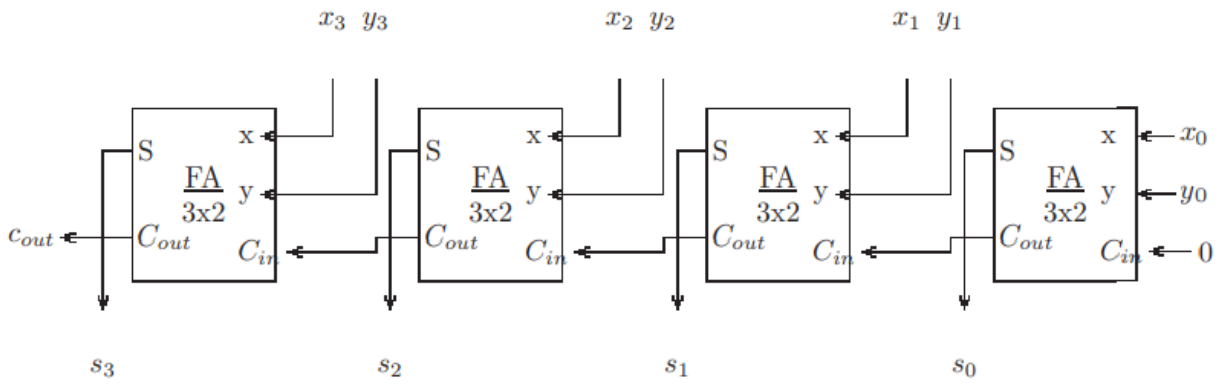
It is used to build a binary adder. The formula for the sum, *s*, is the exclusive or of the three inputs: $s = x \oplus y \oplus c_{in}$. The formula for c_{out} is that if any two of the outputs are true, we get a 1 (the + represents logical or): $c_{out} = xy + c_{in}x + c_{in}y$.

Also, you would need 32 full adders to implement MIPS add instruction.

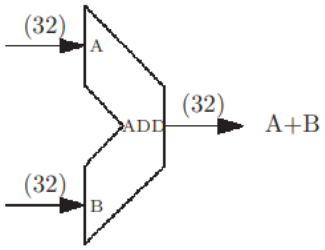
Binary adder:

A component which can add two *n*-bit words. It requires *n* full adders, one for each bit position.

- Example: 4-bit binary adder requires 4 full adders (shown below, read right to left)
- The carry out from each stage is the carry in to the next (like in ordinary addition)



The following is a block diagram of a 32-bit adder (+ means addition here). Most books will draw this like a wedge shaped object or rectangle. (this example doesn't show the c.out)



ALU stands for *arithmetic logic unit*. It's part of the CPU that does computations, responsible for arithmetic (like add, sub) and logical operations (like andi, or, nor) . It can also be used to:

- Determine whether a conditional branch is to be taken
- Computing an effective memory address (with an explicit memory address, we have register contents + a displacement, which implies an addition needing to take place)

The inputs are two busses (A and B) and four control inputs (which determine the operation). The outputs are one bus (result of the operation) and Z output signal (output bus is all 0s, AKA whether you get a 0 result coming out of the ALU).

Let's look at the function table for the ALU (note: if means IFFs, according to slides; the OR means inclusive or; unspecified meaning something to do with two's complement; NOR is one's complement of the result):

Operation Select	Data Out	Z Out
0000	A AND B	1 if A AND B is all 0's
0001	A OR B	1 if A OR B is all 0's
0010	A + B	1 if A + B is all 0's
0110	A - B	1 if A - B is all 0's
0111	[unspecified]	1 if A < B
1100	A NOR B	1 if A NOR B is all 0's

Figure 6.63: Function table for the MIPS ALU. Each of the inputs, A and B, is a 32-bit bus. Z is a 1-bit output, indicating a zero result

For operation 0111, the operation is compared using two's complement, so check the highest-order bit of each number (leftmost). If it's 1, the number is negative. And negative is always less than positive.

Let's look at a block diagram:

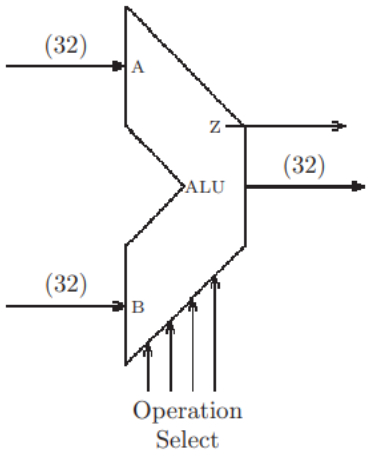


Figure 6.64: Block diagram of an ALU for the MIPS processor

This is a 32-bit ALU. A and B are both 32-bit inputs, and the output is also 32-bits. The Z output is a single output (the operation select has 4 single bits coming in). This is what we want in our MIPS processor since the word size in our instruction set is 32 bits.