

Lecture 7: Review-Combinational Logic

CS311 Computer Organization

Soontae Kim

School of Computing

KAIST

Fall 2018

HW#1 & Project #1

- **Programming merge sort in MIPS assembly**
 - Due on Sept. 28 (Fri.)
- **Project#1: installing simplescalar simulator**
 - Due on Oct. 5 (Fri.)

1.1 Signals, Logic Operators, and Gates



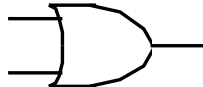

Name	NOT	AND	OR	XOR
Graphical symbol				
Operator sign and alternate(s)	x' $\neg X$ or \bar{X}	xy $x \wedge y$	$x \vee y$ $x + y$	$x \oplus y$ $x \neq y$
Output is 1 iff:	Input is 0	Both inputs are 1s	At least one input is 1	Inputs are not equal
Arithmetic expression	$1 - x$	$x \times y$ or xy	$x + y - xy$	$x + y - 2xy$

Figure 1 Some basic elements of digital logic circuits, with operator signs used in this book highlighted.

Variations in Gate Symbols

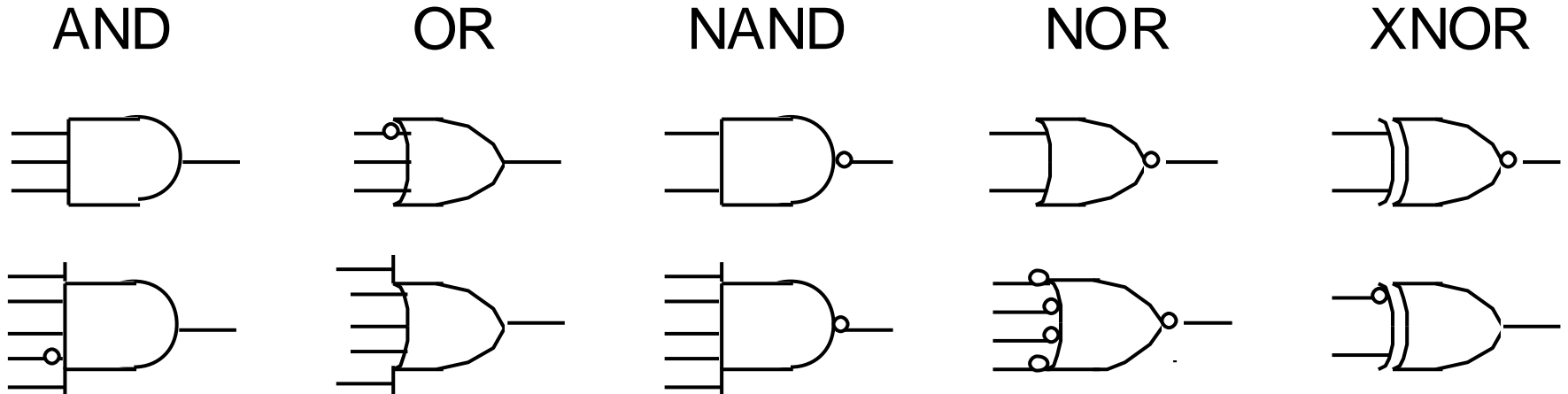
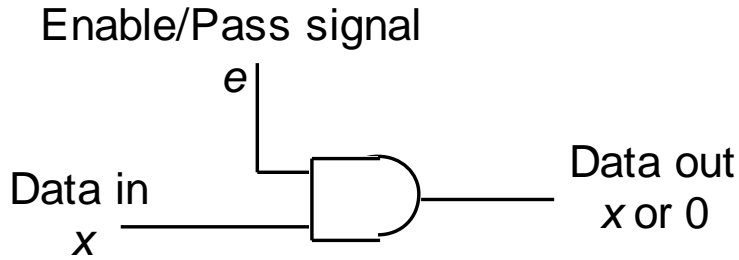
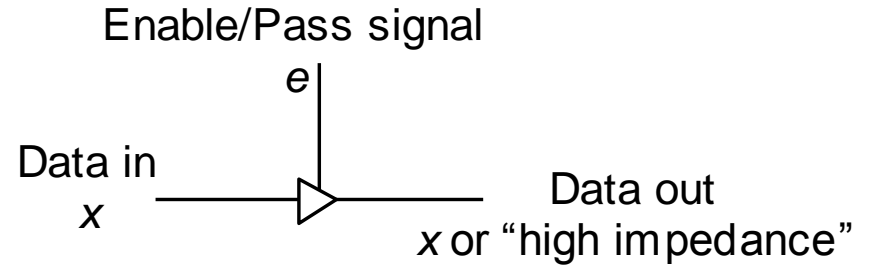


Figure 2 Gates with more than two inputs and/or with inverted signals at input or output.

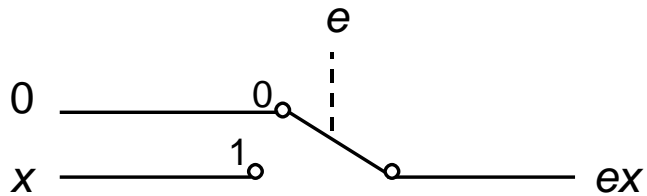
Gates as Control Elements



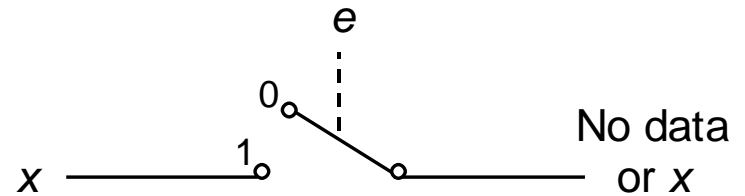
(a) AND gate for controlled transfer



(b) Tristate buffer



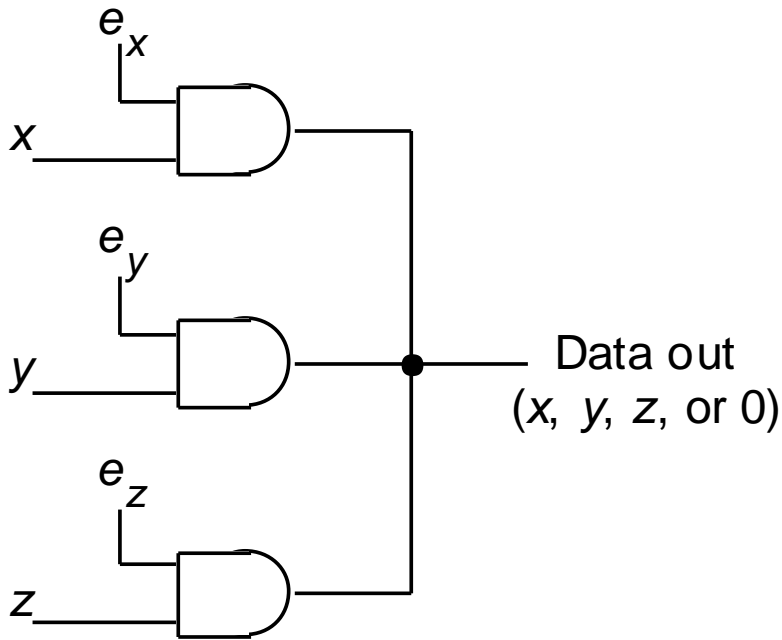
(c) Model for AND switch.



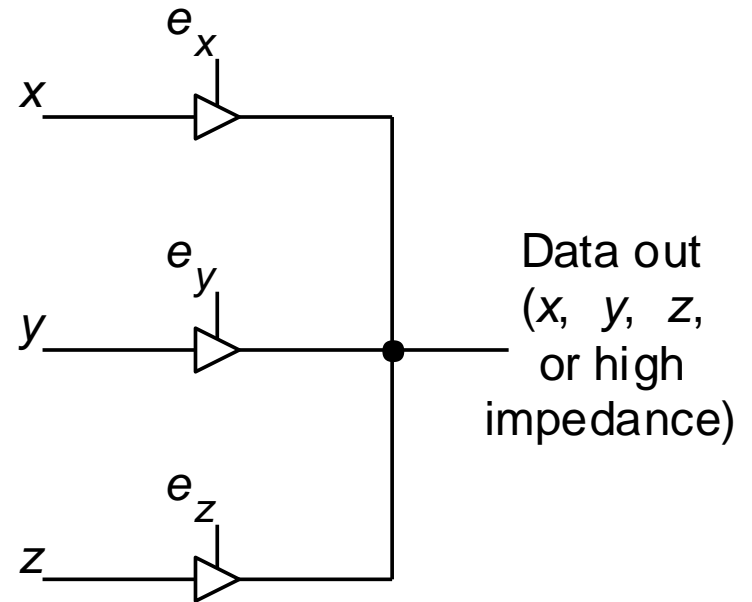
(d) Model for tristate buffer.

Figure 3 An AND gate and a tristate buffer act as controlled switches or valves. An inverting buffer is logically the same as a NOT gate.

Wired OR and Bus Connections



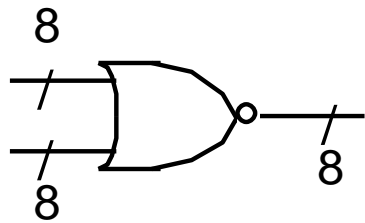
(a) Wired OR of product terms



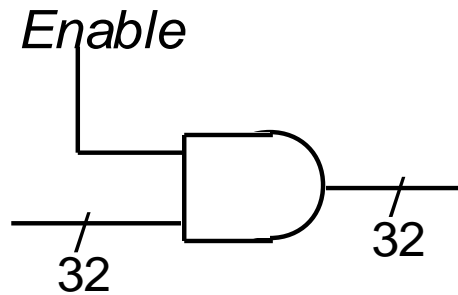
(b) Wired OR of tristate outputs

Figure 4 Wired OR allows tying together of several controlled signals.

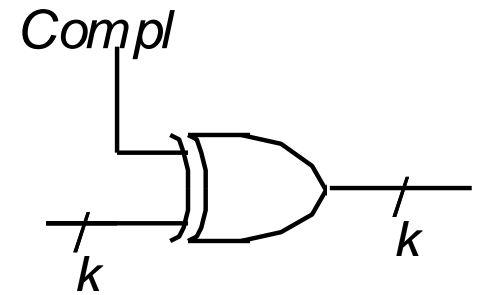
Control/Data Signals and Signal Bundles



(a) 8 NOR gates



(b) 32 AND gates



(c) k XOR gates

Figure 5 Arrays of logic gates represented by a single gate symbol.

1.2 Boolean Functions and Expressions

Ways of specifying a logic function

- Truth table: 2^n row, “don’t-care” in input or output
- Logic expression: $w' (x \vee y \vee z)$, product-of-sums, sum-of-products, equivalent expressions
- Word statement: Alarm will sound if the door is opened while the security system is engaged, or when the smoke detector is triggered
- Logic circuit diagram

Manipulating Logic Expressions

Table 1.2 Laws (basic identities) of Boolean algebra.

Name of law	OR version	AND version
Identity	$x \vee 0 = x$	$x 1 = x$
One/Zero	$x \vee 1 = 1$	$x 0 = 0$
Idempotent	$x \vee x = x$	$x x = x$
Inverse	$x \vee x' = 1$	$x x' = 0$
Commutative	$x \vee y = y \vee x$	$x y = y x$
Associative	$(x \vee y) \vee z = x \vee (y \vee z)$	$(x y) z = x (y z)$
Distributive	$x \vee (y z) = (x \vee y) (x \vee z)$	$x (y \vee z) = (x y) \vee (x z)$
DeMorgan's	$(x \vee y)' = x' y'$	$(x y)' = x' \vee y'$

Proving the Equivalence of Logic Expressions

Example 1.1

- Truth-table method: Exhaustive verification

- Arithmetic substitution

$$x \vee y = x + y - xy$$

$$x \oplus y = x + y - 2xy$$

Example: $x \oplus y \equiv? x'y \vee xy'$

$$x + y - 2xy \equiv? (1-x)y + x(1-y) - (1-x)y x(1-y)$$

- Case analysis: two cases, $x = 0$ or $x = 1$
- Logic expression manipulation

1.3 Designing Gate Networks

- AND-OR, NAND-NAND, OR-AND, NOR-NOR
- Logic optimization: cost, speed, power dissipation

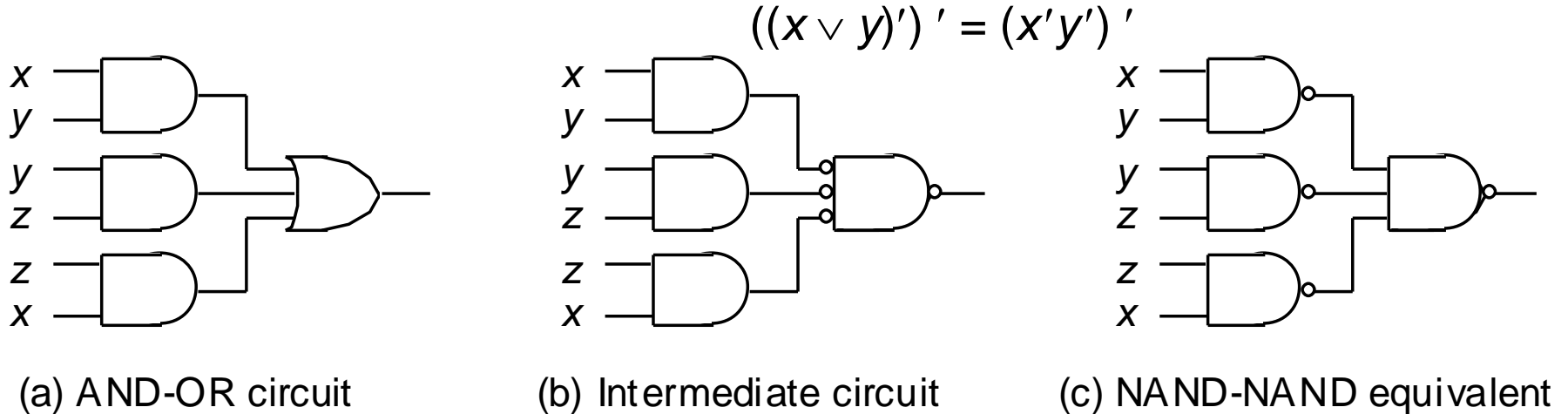
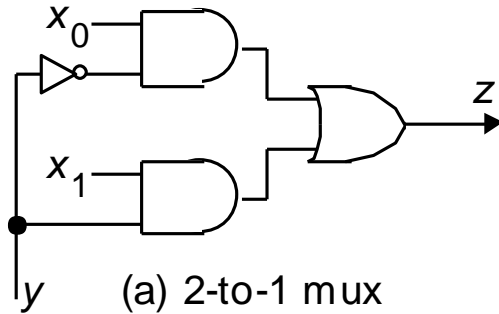


Figure 6 A two-level AND-OR circuit and two equivalent circuits.

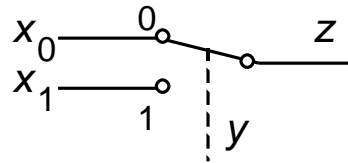
1.4 Useful Combinational Parts

- High-level building blocks
- Much like prefab parts used in building a house
- Here we cover three useful parts:
multiplexers, decoders/demultiplexers, encoders

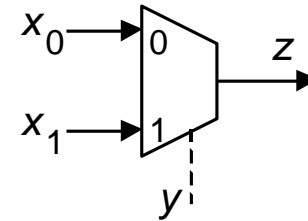
Multiplexers



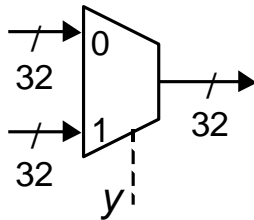
(a) 2-to-1 mux



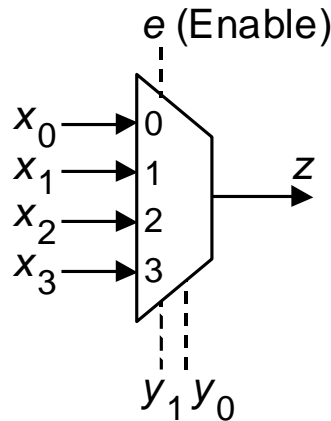
(b) Switch view



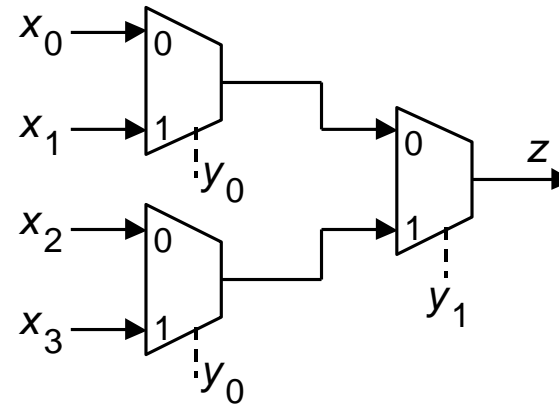
(c) Mux symbol



(d) Mux array



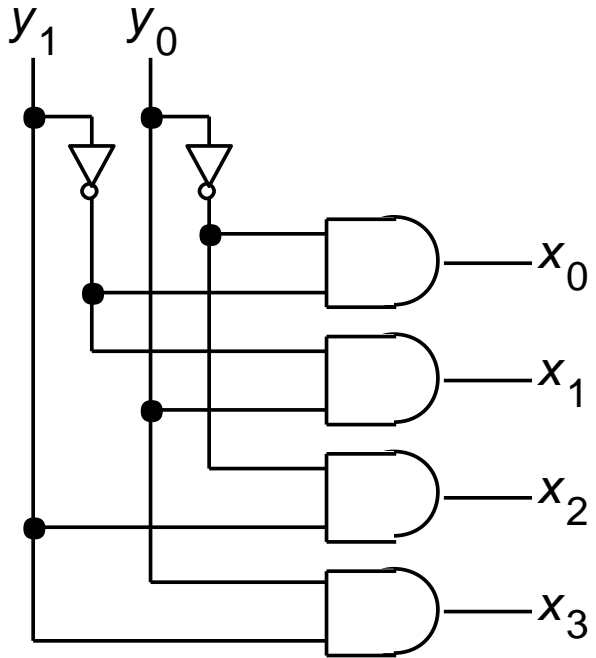
(e) 4-to-1 mux with enable



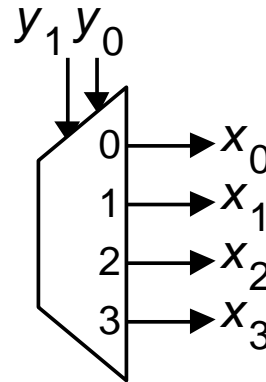
(e) 4-to-1 mux design

Figure 7 Multiplexer (mux), or selector, allows one of several inputs to be selected and routed to output depending on the binary value of a set of selection or address signals provided to it.

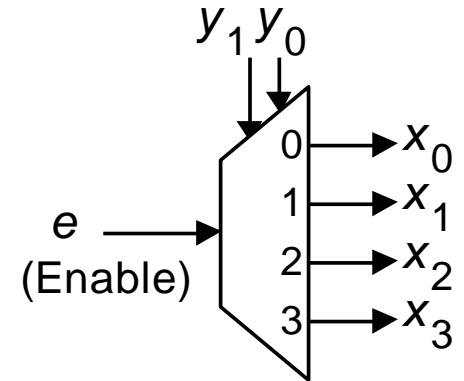
Decoders/Demultiplexers



(a) 2-to-4 decoder



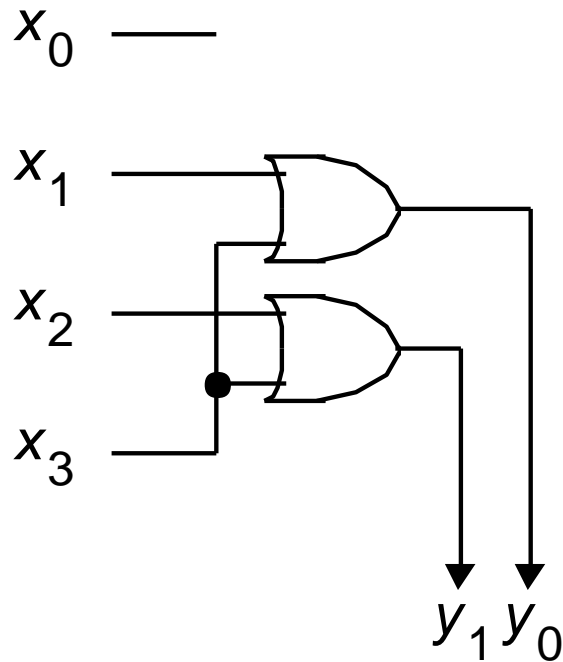
(b) Decoder symbol



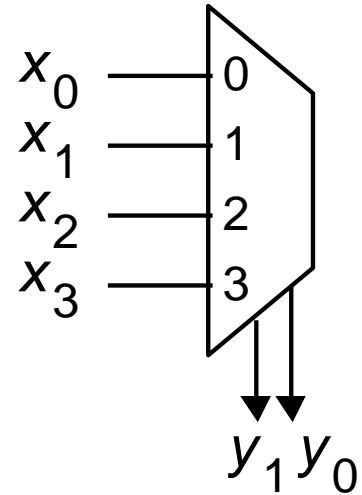
(c) Demultiplexer, or decoder with “enable”

Figure 8 A decoder allows the selection of one of 2^a options using an a -bit address as input. A demultiplexer (demux) is a decoder that only selects an output if its enable signal is asserted.

Encoders



(a) 4-to-2 encoder

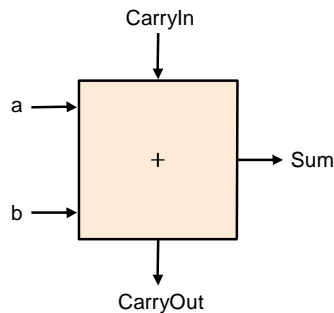


(b) Encoder symbol

Figure 9 A 2^a -to- a encoder outputs an a -bit binary number equal to the index of the single 1 among its 2^a inputs.

An ALU (arithmetic logic unit)

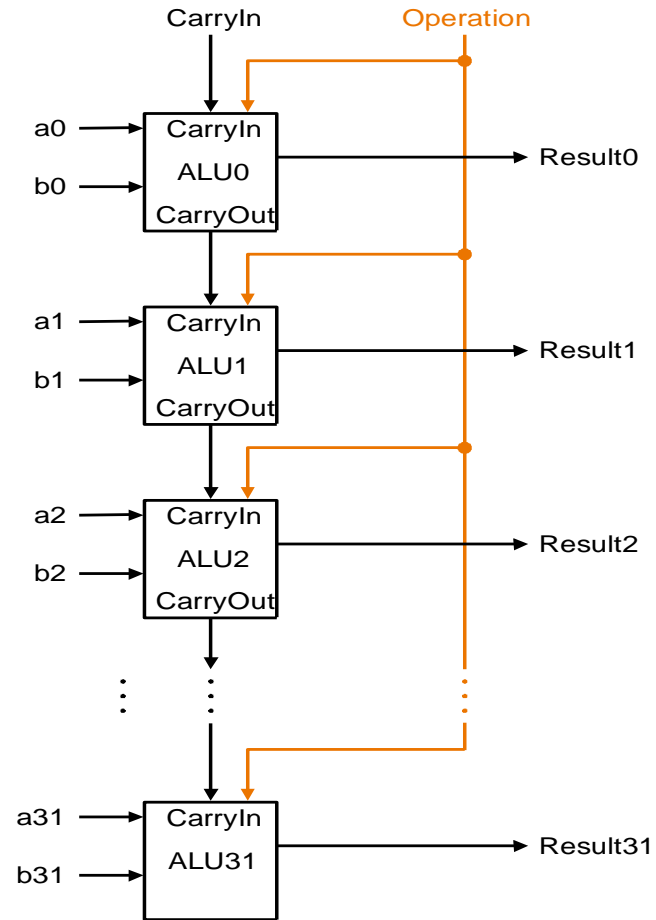
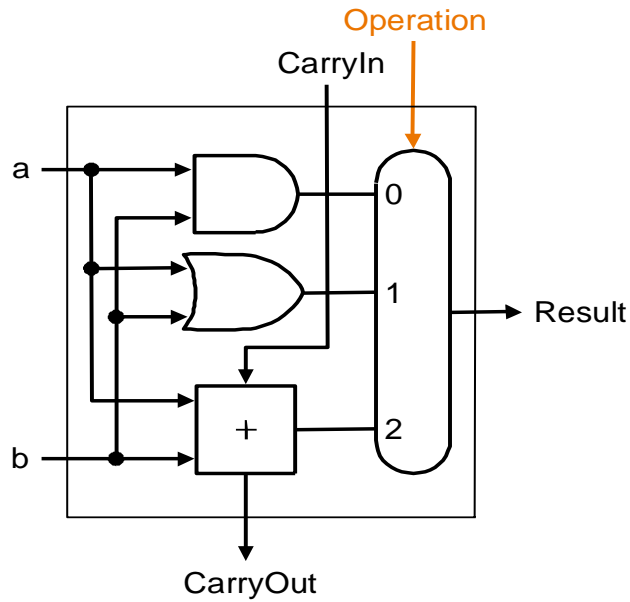
- **Not easy to decide the “best” way to build something**
 - Don't want too many inputs to a single gate
 - Don't want to have to go through too many gates
 - for our purposes, ease of comprehension is important
- **Let's look at a 1-bit ALU for addition:**



$$c_{out} = a b + a c_{in} + b c_{in}$$
$$sum = a \text{ xor } b \text{ xor } c_{in}$$

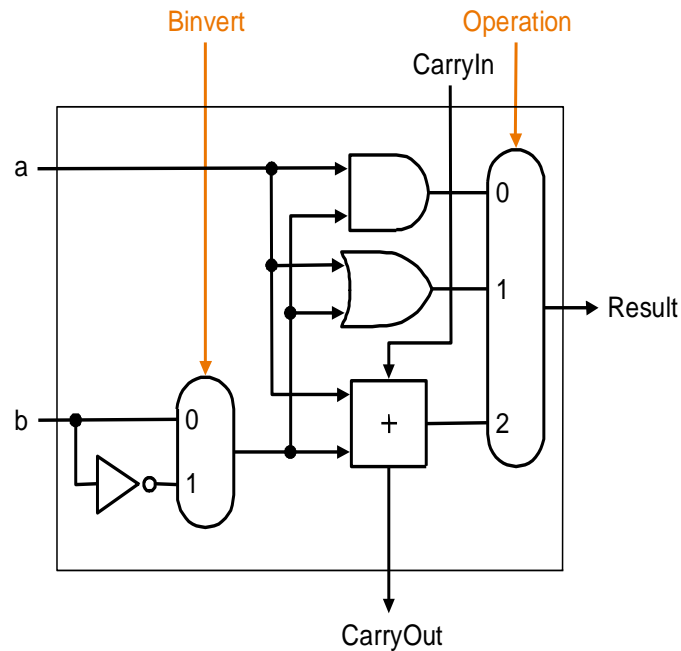
- **How could we build a 1-bit ALU for “add”, “and”, and “or”?**
- **How could we build a 32-bit ALU?**

Building a 32 bit ALU



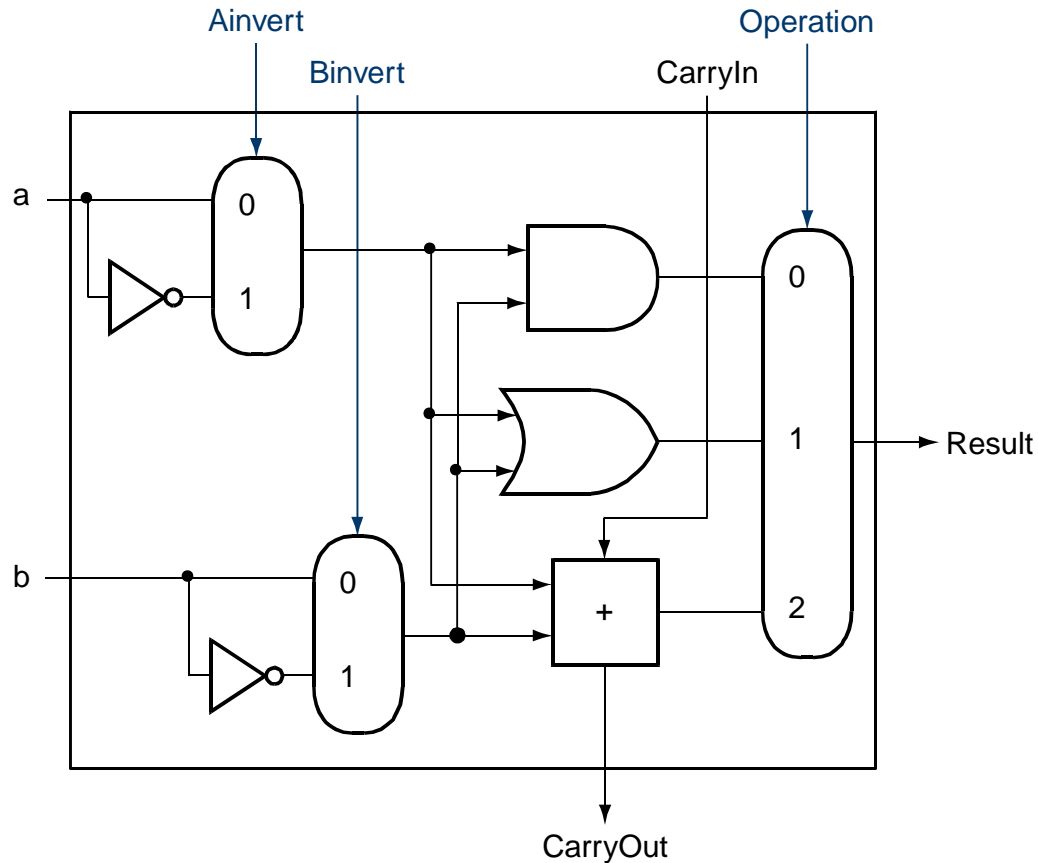
What about subtraction ($a - b$) ?

- **Two's complement approach: just negate b and add.**
- **How do we negate?**
- **A very clever solution:**



Adding a NOR function

- **Can also choose to invert a. How do we get “a NOR b” ?**

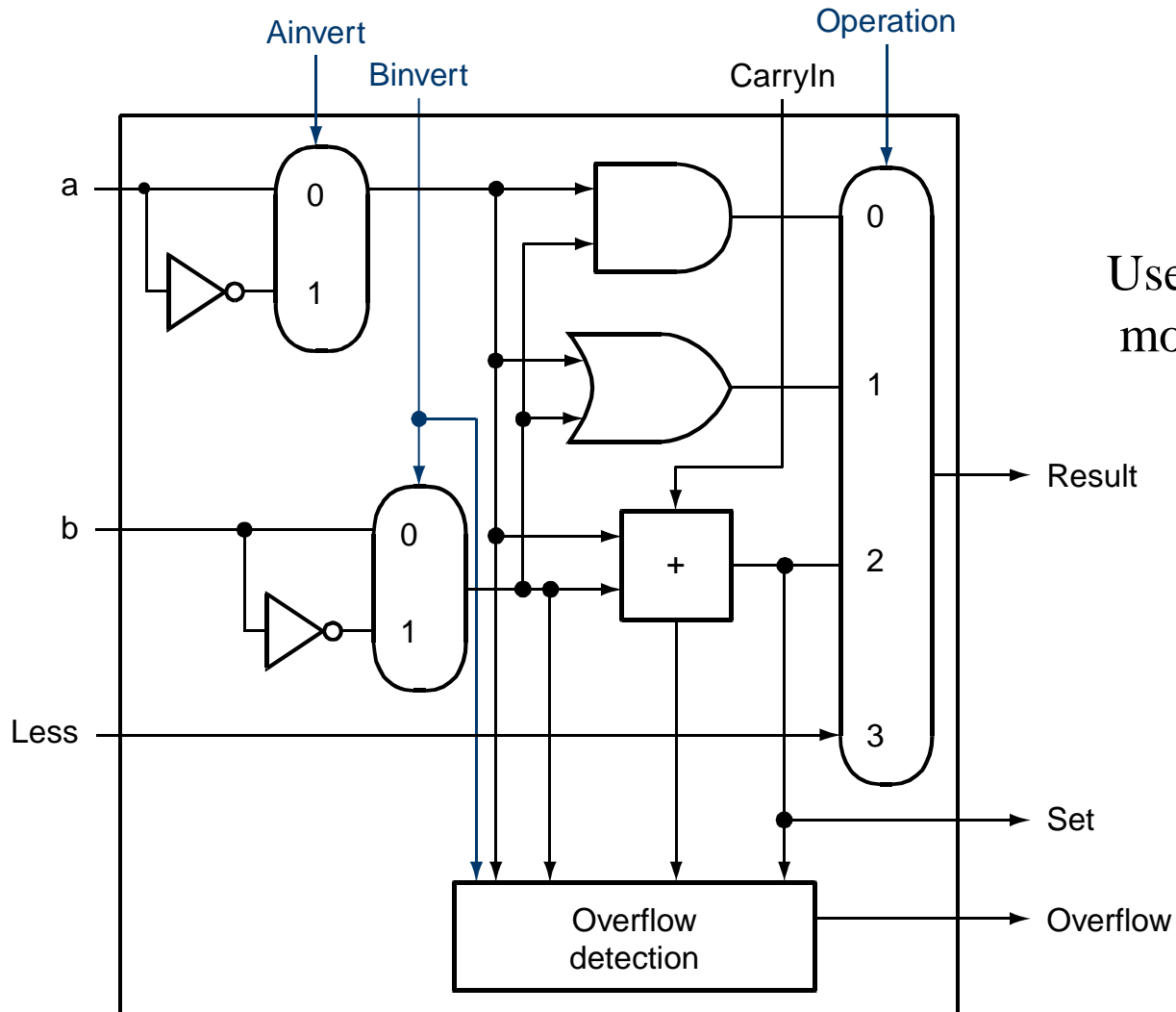


Tailoring the ALU to the MIPS

- **Need to support the set-on-less-than instruction (slt)**
 - remember: slt is an arithmetic instruction
 - produces a 1 if $rs < rt$ and 0 otherwise
 - use subtraction: $(a-b) < 0$ implies $a < b$
- **Need to support test for equality (beq)**
 - use subtraction: $(a-b) = 0$ implies $a = b$

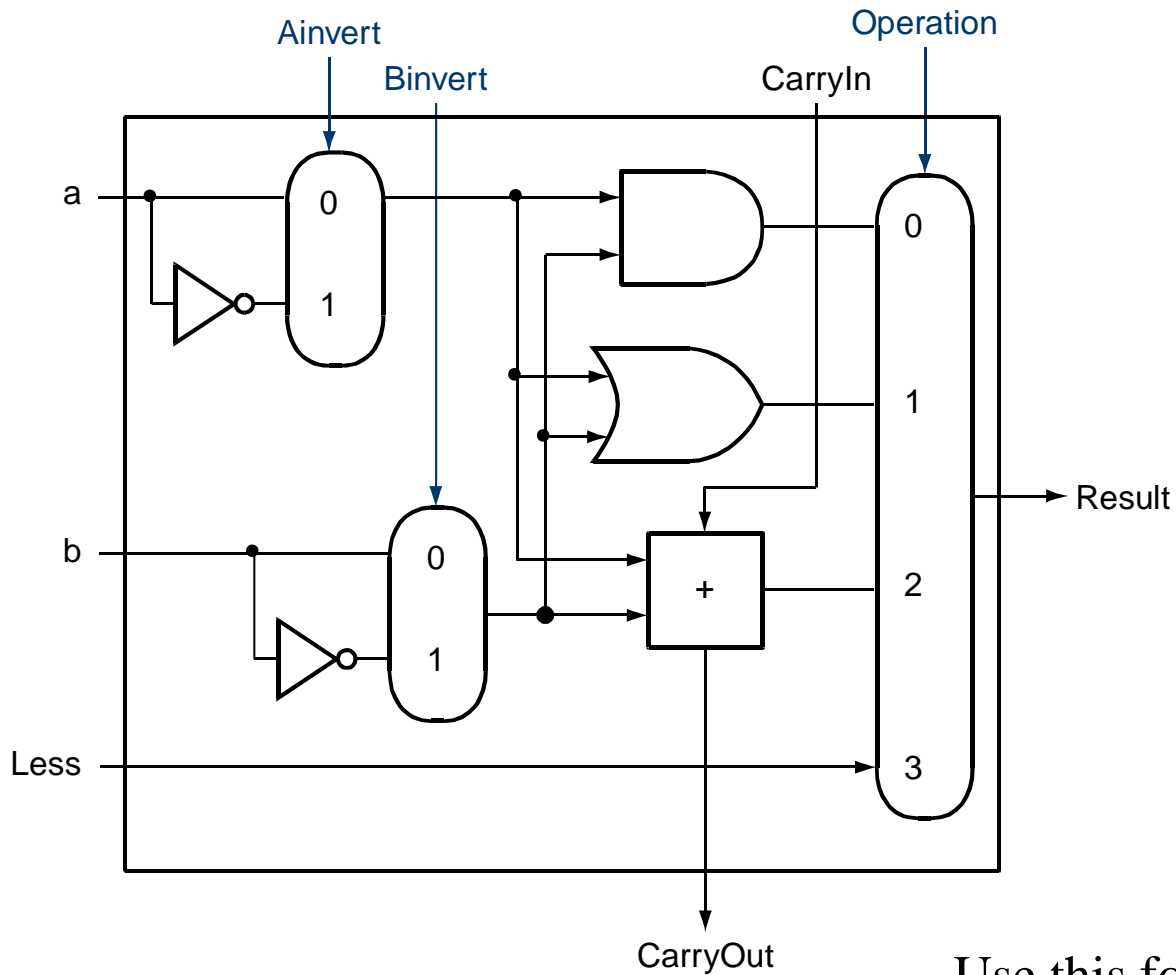
Supporting slt

- **Can we figure out the idea?**



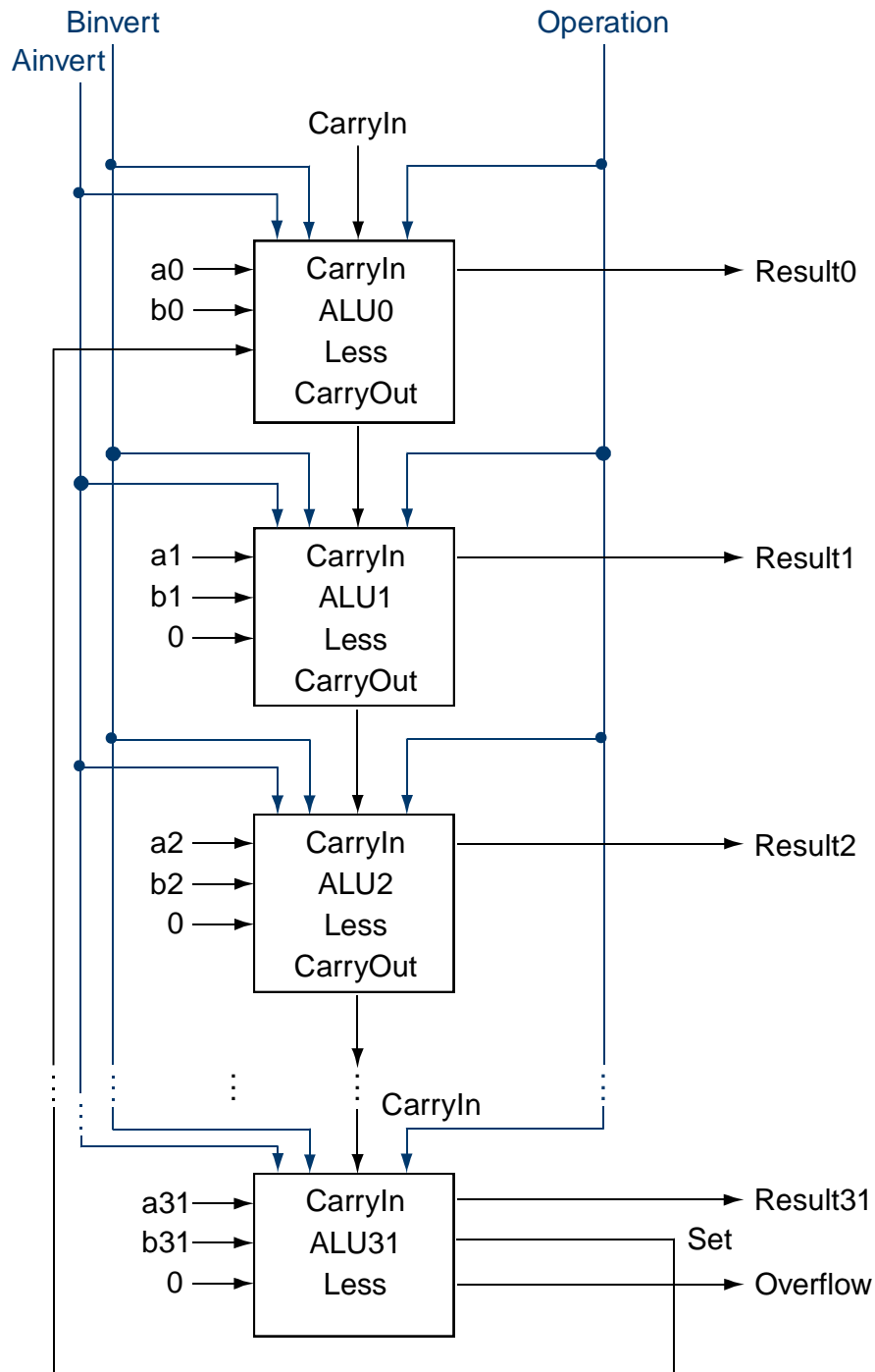
Use this ALU for
most significant bit

Supporting slt (cont'd)



Use this for all other bits

Supporting slt (cont'd)



Conclusion

- **We can build an ALU to support the MIPS instruction set**
 - key idea: use multiplexor to select the output we want
 - we can efficiently perform subtraction using two's complement
 - we can replicate a 1-bit ALU to produce a 32-bit ALU
- **Important points about hardware**
 - all of the gates are always working
 - the speed of a gate is affected by the number of inputs to the gate
 - the speed of a circuit is affected by the number of gates in series (on the “critical path” or the “deepest level of logic”)