CSE 675.02: Introduction to Computer Architecture

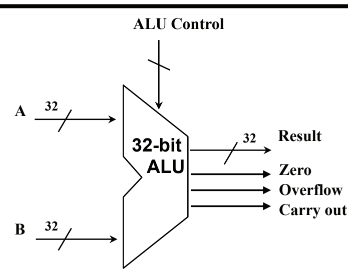
Arithmetic / Logic Unit – ALU Design

Presentation F

Reading Assignment: B5, 3.4

Slides by Gojko Babić

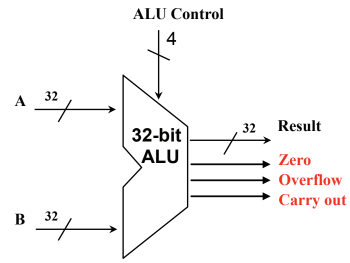
32-bit ALU



- Our ALU should be able to perform functions:
  - logical and function
  - logical or function
  - arithmetic add function
  - arithmetic subtract function
  - arithmetic slt (set-less-then) function
  - logical nor function
- ALU control lines define a function to be performed on A and B.

Functioning of 32-bit ALU

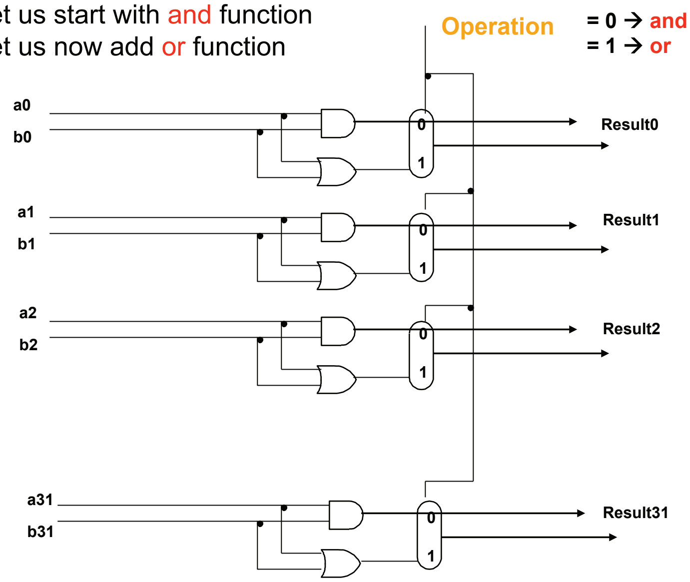
	ALU Control lines		
Function	**Ainvert**	**Binvert**	**Operation**
and	0	0	00
or	0	0	01
add	0	0	10
subtract	0	1	10
slt	0	1	11
nor	1	1	00



- Result lines provide result of the chosen function applied to values of A and B
- Since this ALU operates on 32-bit operands, it is called 32-bit ALU
- Zero output indicates if all Result lines have value 0
- Overflow indicates integer overflow of add and subtract functions; for unsigned integers, this overflow indicator does not provide any useful information
- Carry out indicates carry out and unsigned integer overflow

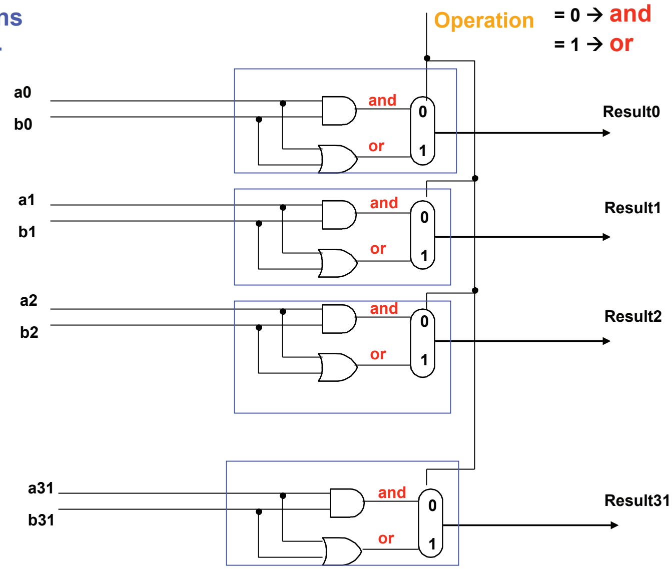
Designing 32-bit ALU: Beginning

1. Let us start with and function
2. Let us now add or function



Designing 32-bit ALU: Principles

- A number of functions are performed internally, but only one result is chosen for the output of ALU
- 32-bit ALU is built out of 32 identical 1-bit ALU's



Designing the Adder

- 32-bit adder is built out of 32 1-bit adders

1-bit Adder

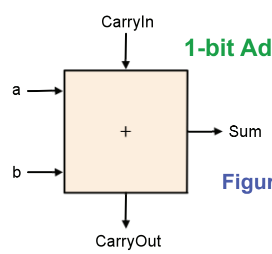


Figure B.5.2

1-bit Adder Truth Table

Input			Output	
a	b	Carry In	Sum	Carry 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

Figure B.5.3

From the truth table and after minimization, we can have this design for CarryOut

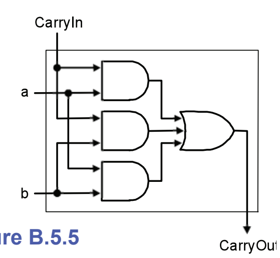
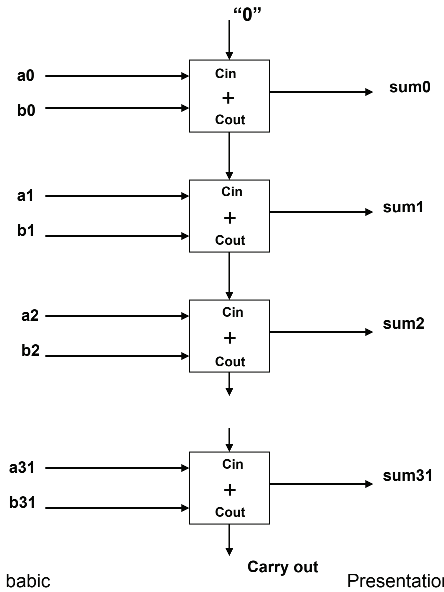


Figure B.5.5

32-bit Adder



This is a ripple carry adder.

The key to speeding up addition is determining carry out in the higher order bits sooner.
Result: **Carry look-ahead adder**.

32-bit ALU With 3 Functions

1-bit ALU

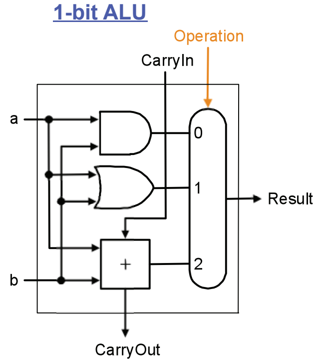
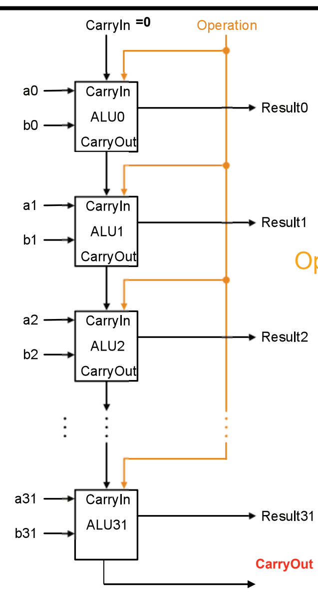


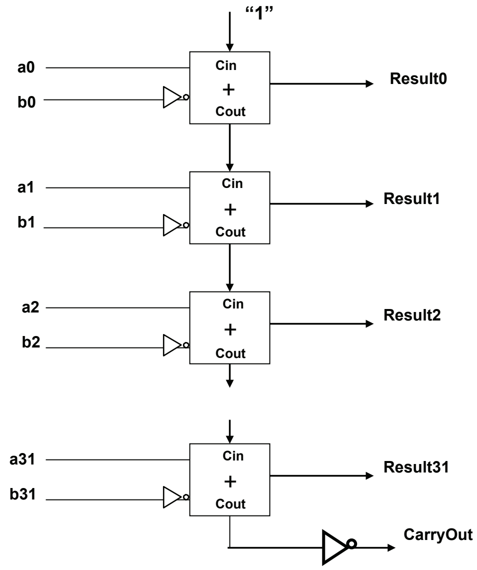
Figure B.5.6



Operation = 00 → **and**
= 01 → **or**
= 10 → **add**

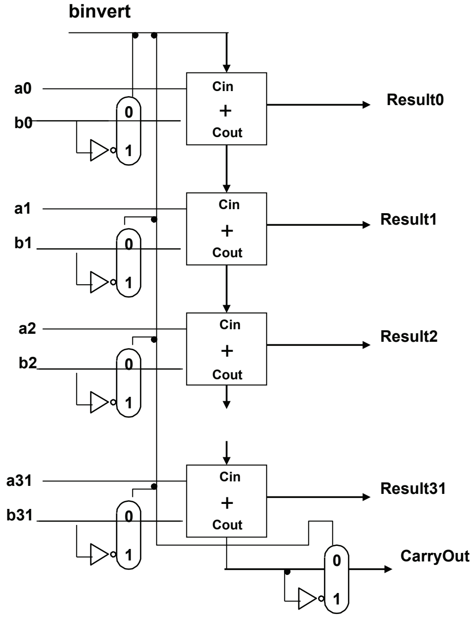
Figure B.5.7
+ carry out

32-bit Subtractor



$$A - B = A + (-B)$$
$$= A + \overline{B} + 1$$

32-bit Adder / Subtractor



Binvert = 0 → addition
= 1 → subtraction

32-bit ALU With 4 Functions

1-bit ALU

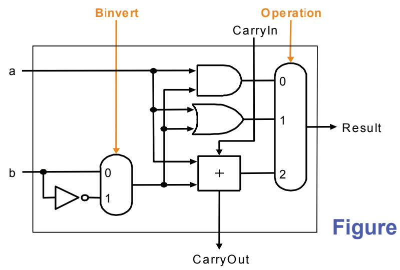
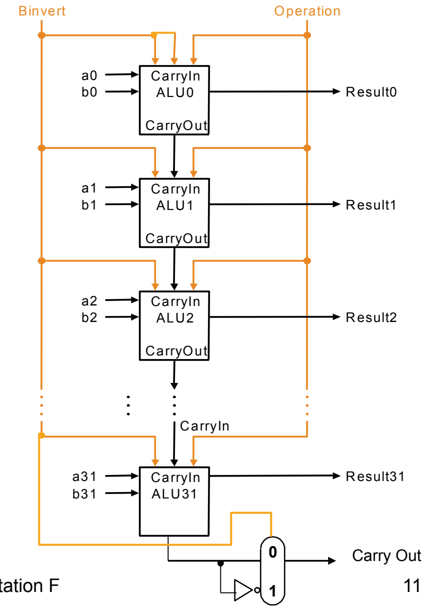


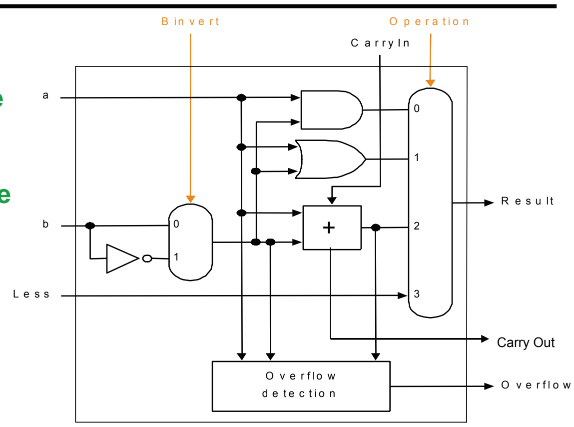
Figure B.5.8



	Control lines	
Function	**Binvert** (1 line)	**Operation** (2 lines)
and	0	00
or	0	01
add	0	10
subtract	1	10

2's Complement Overflow

- 2's complement overflow happens:
  - if a sum of two positive numbers results in a negative number
  - if a sum of two negative numbers results in a positive number

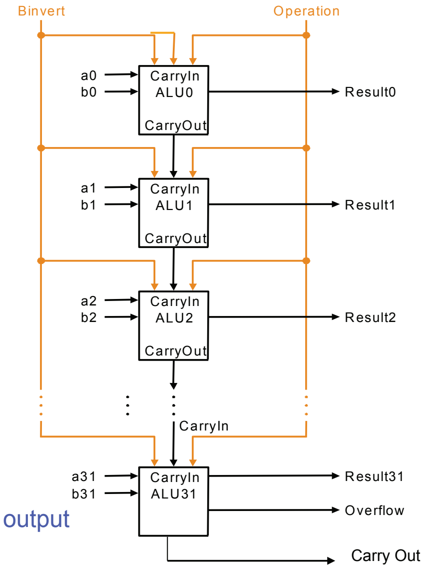


1-bit ALU for the most significant bit

Other 1-bit ALUs, i.e. non-most significant bit ALUs, are not affected.

32-bit ALU With 4 Functions and Overflow

Function	Control lines: Binvert (1 line)	Control lines: Operation (2 lines)
and	0	00
or	0	01
add	0	10
subtract	1	10



Missing: slt & nor functions and Zero output