Chapter 4

Digital Logic Revision (by Dr. Tamer Mostafa)

ALU + Register File

Lecture 5

Dr. Karim Emara

Agenda

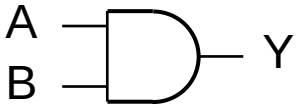
- Combinational Circuits
- Sequential Circuits
- CPU Design
- Basic CPU Elements
  - ALU
  - Register File

Logic Blocks

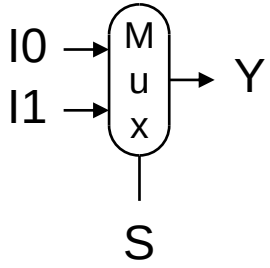
- A logic block has a number of binary inputs and produces a number of binary outputs – the simplest logic block is composed of a few transistors
- A logic block is termed *combinational* if the output is only a function of the inputs
- A logic block is termed *sequential* if the block has some internal memory (state) that also influences the output
- A basic logic block is termed a *gate* (AND, OR, NOT, etc.)

Combinational Elements

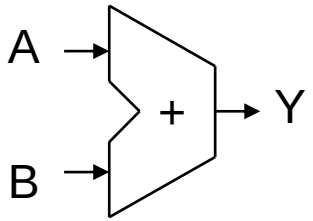
- AND-gate
  - Y = A & B



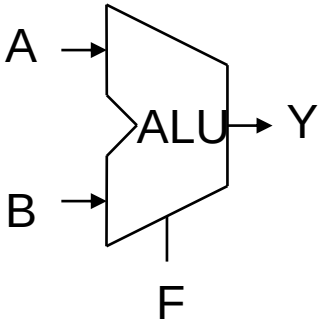
- Multiplexer
  - Y = S ? I1 : I0



- Adder
  - Y = A + B



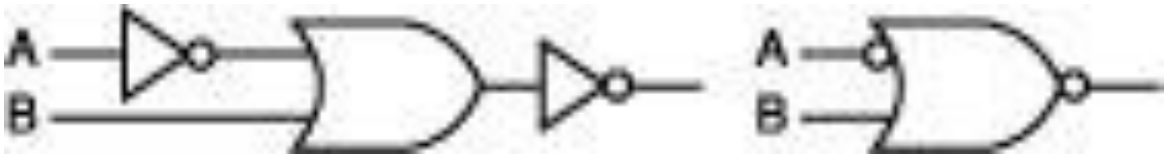
- Arithmetic/Logic Unit
  - Y = F(A, B)



Pictorial Representations

AND OR NOT

What logic function is this?



Boolean Algebra Rules

- Identity law : $A + 0 = A \quad ; \quad A \cdot 1 = A$
- Zero and One laws : $A + 1 = 1 \quad ; \quad A \cdot 0 = 0$
- Inverse laws : $A \cdot \overline{A} = 0 \quad ; \quad A + \overline{A} = 1$
- Commutative laws : $A + B = B + A \quad ; \quad A \cdot B = B \cdot A$
- Associative laws : $A + (B + C) = (A + B) + C$
  $A \cdot (B \cdot C) = (A \cdot B) \cdot C$
- Distributive laws : $A \cdot (B + C) = (A \cdot B) + (A \cdot C)$
  $A + (B \cdot C) = (A + B) \cdot (A + C)$

DeMorgan's Laws

- $\overline{A + B} = \overline{A} \cdot \overline{B}$

- $\overline{A \cdot B} = \overline{A} + \overline{B}$

Truth Table

- A truth table defines the outputs of a logic block for each set of inputs
- Consider a block with 3 inputs A, B, C and an output E that is true only if *exactly* 2 inputs are true

A	B	C	E

Truth Table

- A truth table defines the outputs of a logic block for each set of inputs
- Consider a block with 3 inputs A, B, C and an output E that is true only if *exactly* 2 inputs are true

A	B	C	E
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Can be compressed by only representing cases that have an output of 1

Sum of Products

- Can represent any logic block with the AND, OR, NOT operators
  - Draw the truth table
  - For each true output, represent the corresponding inputs as a product
  - The final equation is a sum of these products

A	B	C	E
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

$$(A \cdot B \cdot \overline{C}) + (A \cdot C \cdot \overline{B}) + (C \cdot B \cdot \overline{A})$$

- Can also use “product of sums”
- Any equation can be implemented with an array of ANDs, followed by an array of ORs

Boolean Equation

- Consider the logic block that has an output E that is true only if exactly two of the three inputs A, B, C are true

Multiple correct equations:

Two must be true, but all three cannot be true:

$$E = ((A \cdot B) + (B \cdot C) + (A \cdot C)) \cdot \overline{(A \cdot B \cdot C)}$$

Identify the three cases where it is true:

$$E = (A \cdot B \cdot \overline{C}) + (A \cdot C \cdot \overline{B}) + (C \cdot B \cdot \overline{A})$$

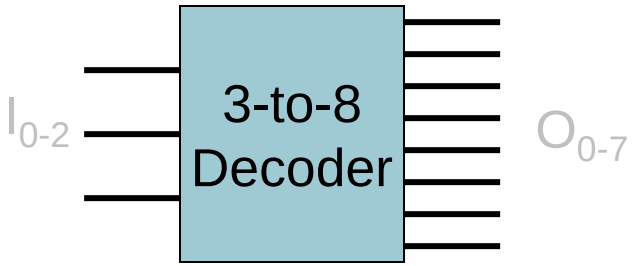
NAND and NOR

- NAND : NOT of AND : A nand B = $\overline{A \cdot B}$
- NOR : NOT of OR : A nor B = $\overline{A + B}$
- NAND and NOR are *universal gates*, i.e., they can be used to construct any complex logical function

Common Logic Blocks – Decoder

Takes in N inputs and activates one of 2^N outputs

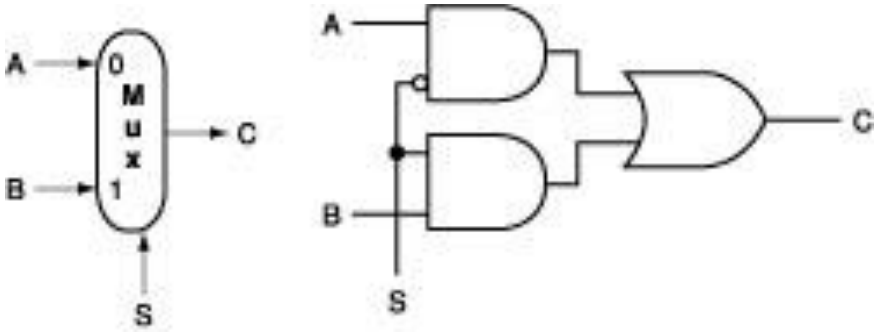
I_0	I_1	I_2	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1



Common Logic Blocks – Multiplexor

- Multiplexor or selector: one of N inputs is reflected on the output depending on the value of the $\log_2 N$ selector bits

2-input mux



Adder Algorithm

	1	0	0	1
	0	1	0	1
Sum	1	1	1	0
Carry	0	0	0	1

Truth Table for the above operations:

A	B	Cin	Sum	Cout
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Adder Algorithm

	1	0	0	1
	0	1	0	1
Sum	1	1	1	0
Carry	0	0	0	1

Truth Table for the above operations:

A	B	Cin	Sum	Cout
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

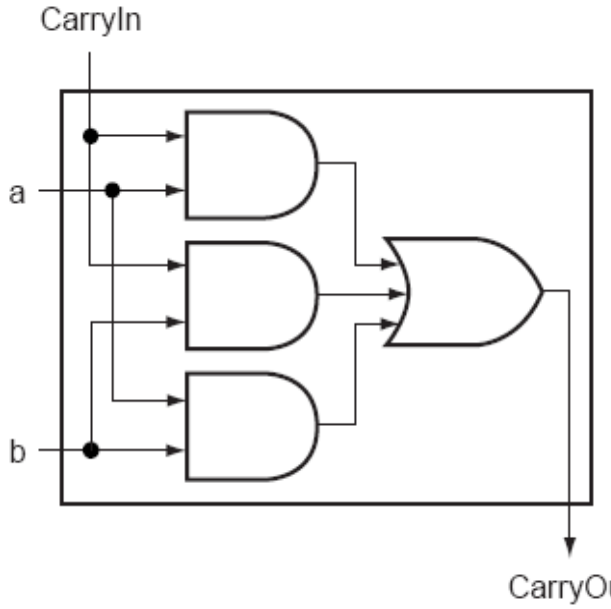
Equations:

$$\text{Sum} = Cin \cdot \overline{A} \cdot \overline{B} + B \cdot \overline{Cin} \cdot \overline{A} + A \cdot \overline{Cin} \cdot \overline{B} + A \cdot B \cdot Cin$$

$$\text{Cout} = A \cdot B \cdot Cin + A \cdot B \cdot \overline{Cin} + A \cdot Cin \cdot \overline{B} + B \cdot Cin \cdot \overline{A}$$

$$= A \cdot B + A \cdot Cin + B \cdot Cin$$

Carry Out Logic



Equations:

Sum = Cin . $\overline{A}$. $\overline{B}$ +
B . $\overline{Cin}$. $\overline{A}$ +
A . $\overline{Cin}$. $\overline{B}$ +
A . B . Cin

Cout = A . B . Cin +
A . B . $\overline{Cin}$ +
A . Cin . $\overline{B}$ +
B . Cin . $\overline{A}$
= A . B +
A . Cin +
B . Cin

FIGURE B.5.5 Adder hardware for the carry out signal. The rest of the adder hardware is the logic for the Sum output given in the equation on page B-28.

Agenda

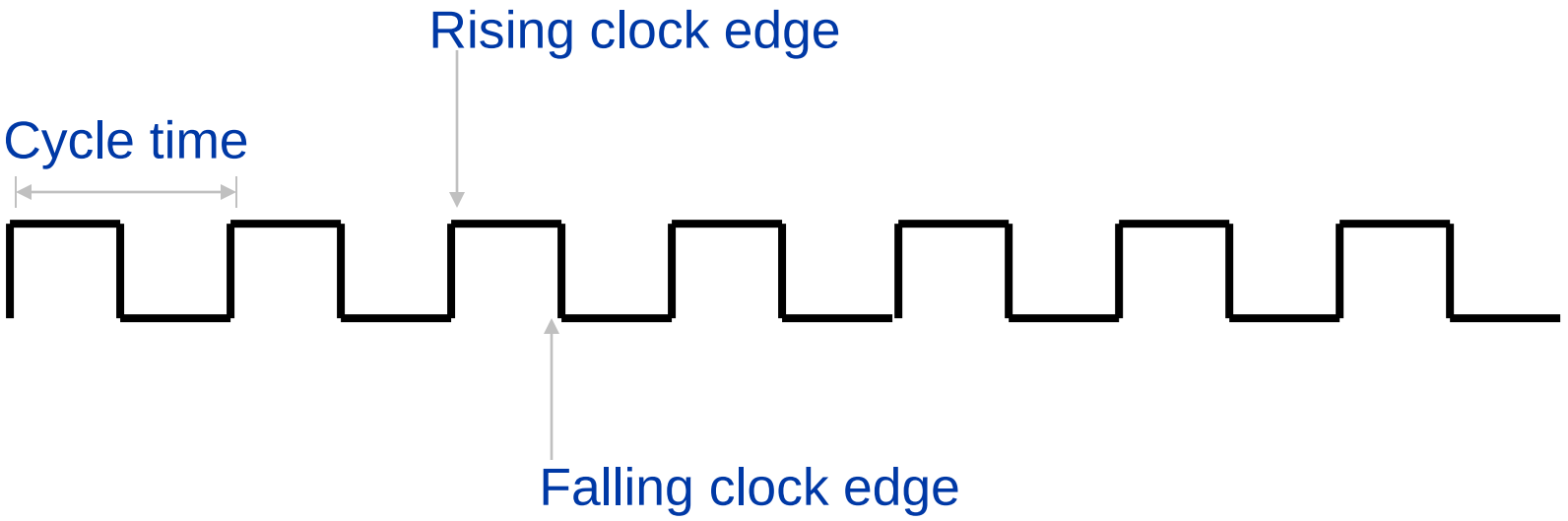
- Combinational Circuits
- Sequential Circuits
- CPU Design
- Basic CPU Elements
  - ALU
  - Register File

Clocks

- A microprocessor is composed of many different circuits that are operating simultaneously – if each circuit X takes in inputs at time TI_X, takes time TE_X to execute the logic, and produces outputs at time TO_X, imagine the complications in co-ordinating the tasks of every circuit

- A major school of thought (used in most processors built today): all circuits on the chip share a clock signal (a square wave) that tells every circuit when to accept inputs, how much time they have to execute the logic, and when they must produce outputs

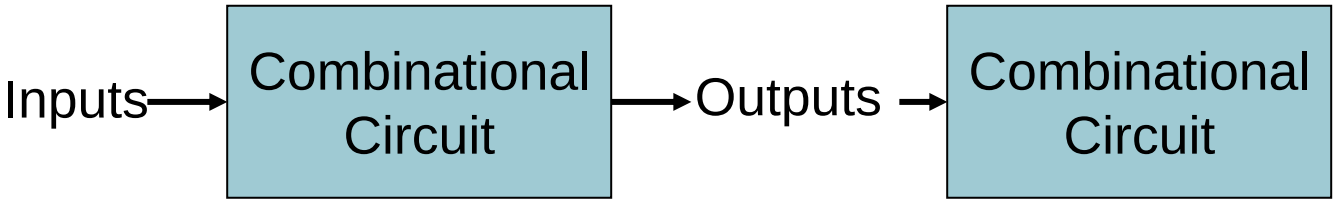
Clock Terminology



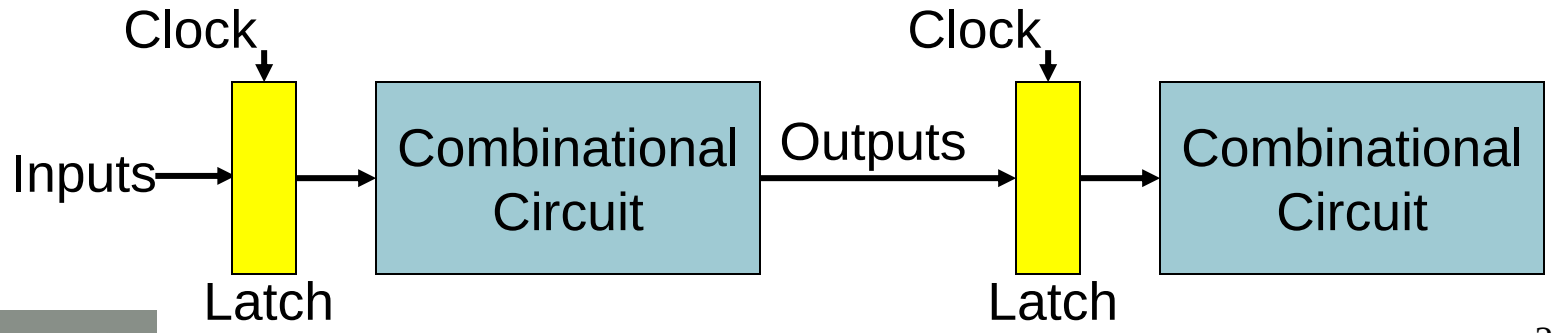
$$4\ \text{GHz} = \text{clock speed} = \frac{1}{\text{cycle time}} = \frac{1}{250\ \text{ps}}.$$

Sequential Circuits

- Until now, circuits were combinational – when inputs change, the outputs change after a while (time = logic delay thru circuit)

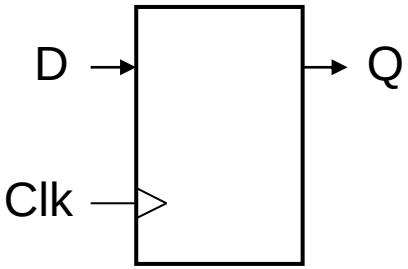


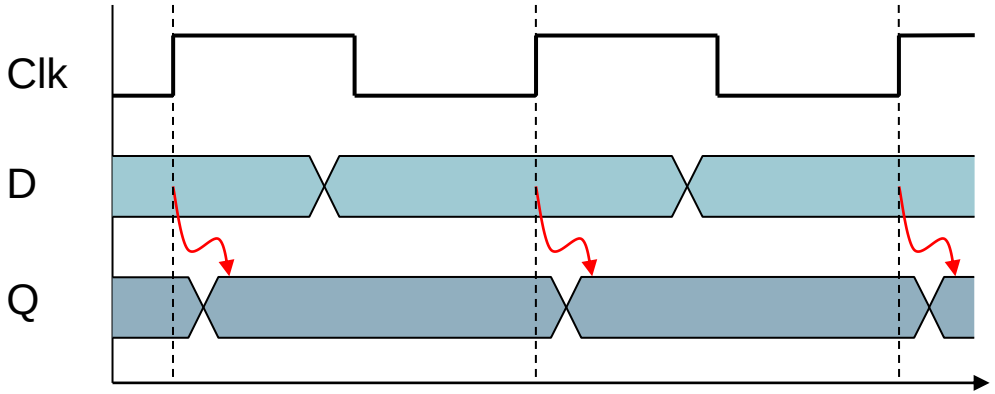
- We want the clock to act like a start and stop signal – a “latch” is a storage device that stores its inputs at a rising clock edge and this storage will not change until the next rising clock edge



Sequential Elements

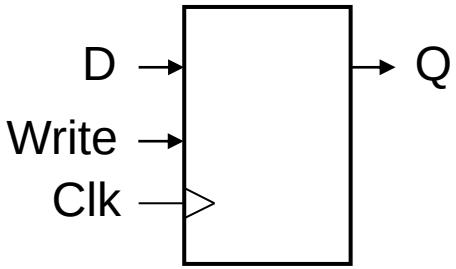
- Register: stores data in a circuit
  - Uses a clock signal to determine when to update the stored value
  - Edge-triggered: update when Clk changes from 0 to 1

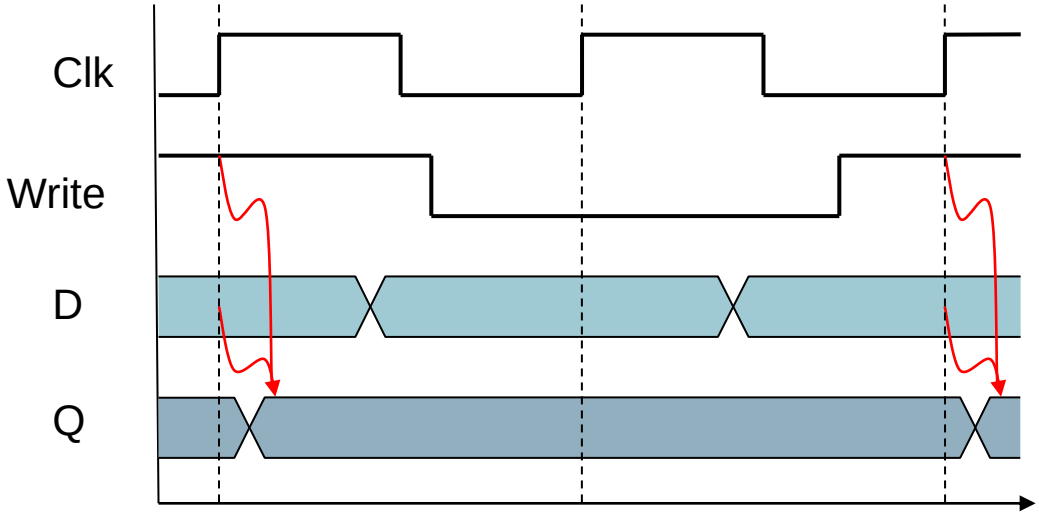




Sequential Elements

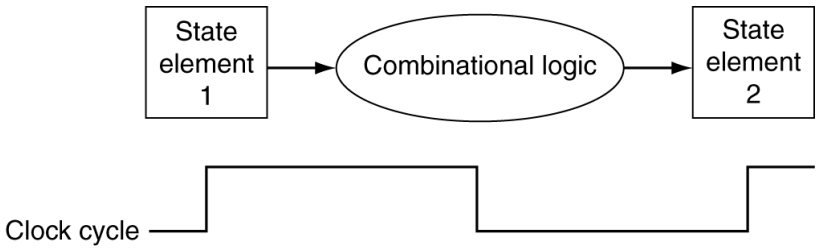
- Register with **write control**
  - Only updates on clock edge when write control input is 1
  - Used when stored value is required later

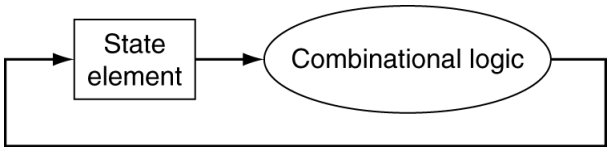




Clocking Methodology

- Combinational logic transforms data during clock cycles
  - Between clock edges
  - Input from state elements, output to state element
  - Longest delay determines clock period





Agenda

- Combinational Circuits
- Sequential Circuits
- CPU Design
- Basic CPU Elements
  - ALU
  - Register File

CPU Design

- CPU performance factors
  - Instruction count
    - Determined by ISA and compiler
  - CPI and Cycle time
    - Determined by CPU hardware
- We will examine two MIPS implementations
  - A simplified version
  - A more realistic pipelined version
- Simple subset, shows most aspects
  - Memory reference: `lw`, `sw`
  - Arithmetic/logical: `add`, `sub`, `and`, `or`, `slt`
  - Control transfer: `beq`, `j`

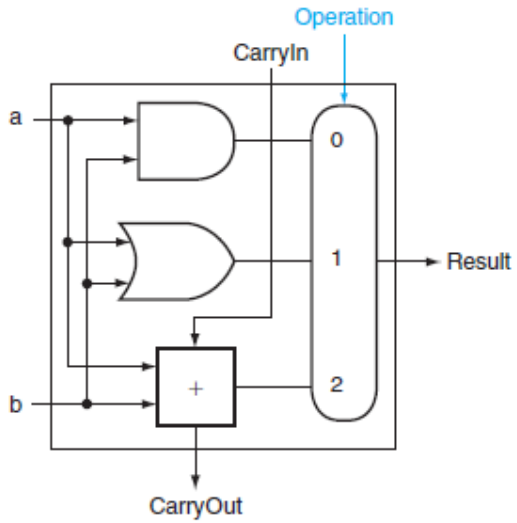
Agenda

- Combinational Circuits
- Sequential Circuits
- CPU Design
- Basic CPU Elements
  - ALU
  - Register File

Arithmetic Logic Unit (ALU)

- ALU should be able to perform all transformation on the register values to execute different instructions (ADD, AND, OR, SUB, …etc.)

ALU – version 1



1-bit ALU: AND, OR, ADD

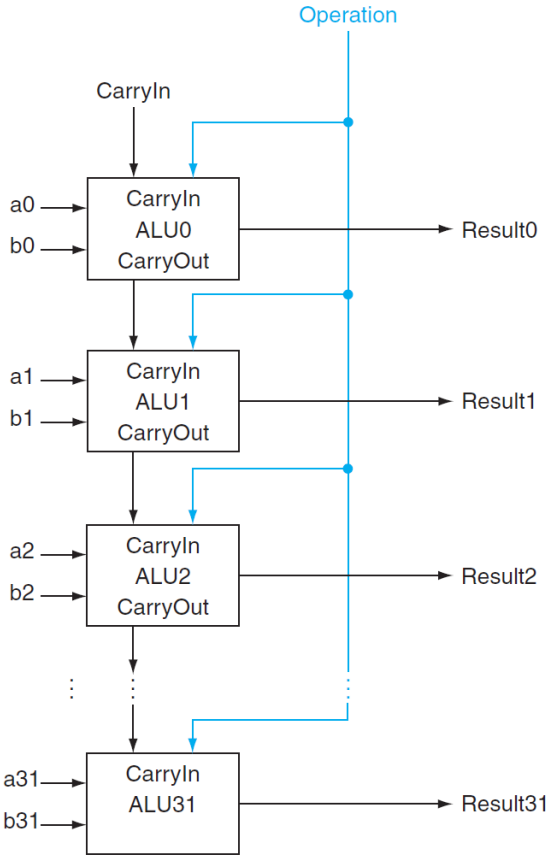
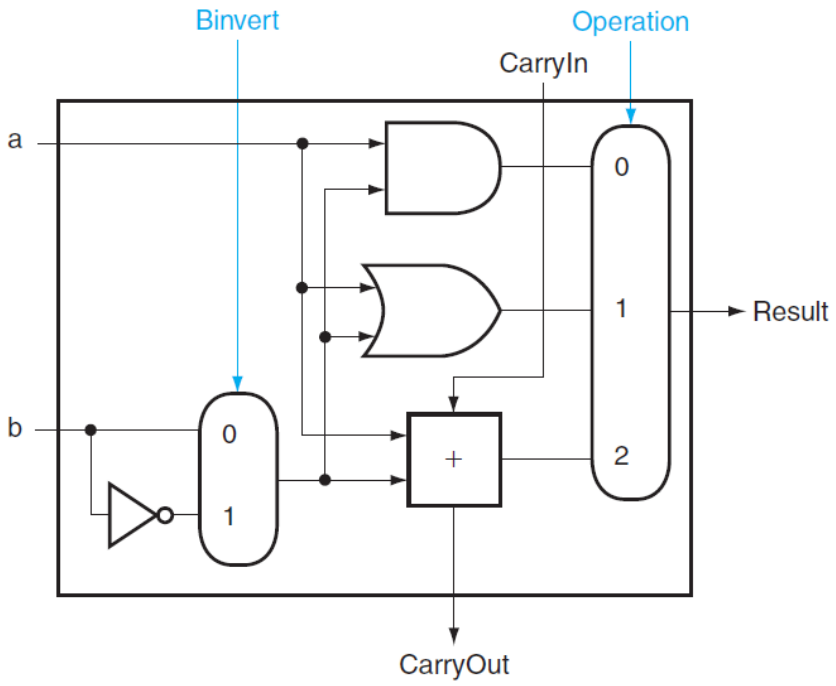
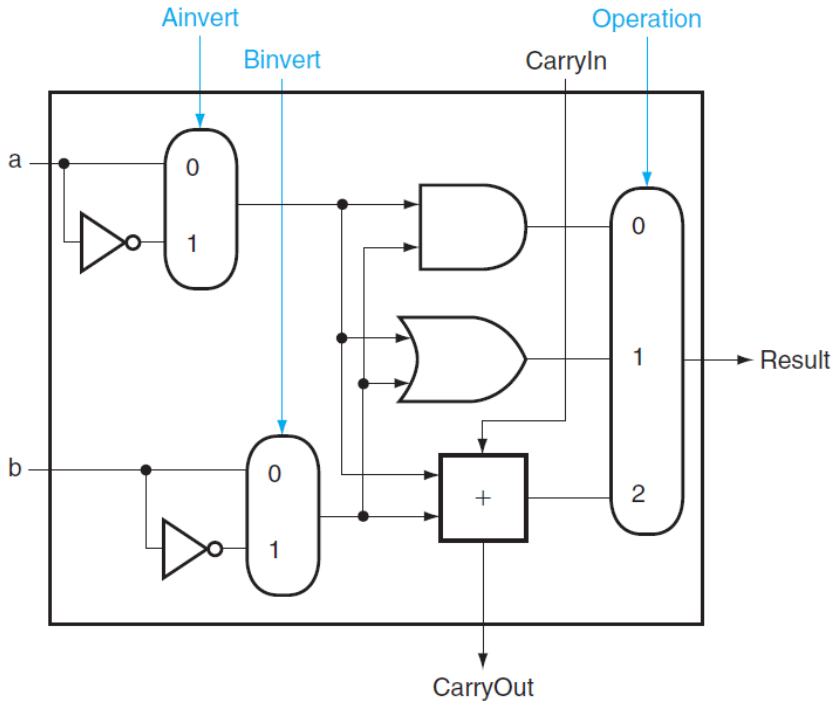


FIGURE B.5.7 A 32-bit ALU constructed from 32 1-bit ALUs. CarryOut of the less significant bit is connected to the CarryIn of the more significant bit. This organization is called ripple carry.

ALU – versions 2 and 3



1-bit ALU: AND, OR, ADD, SUB

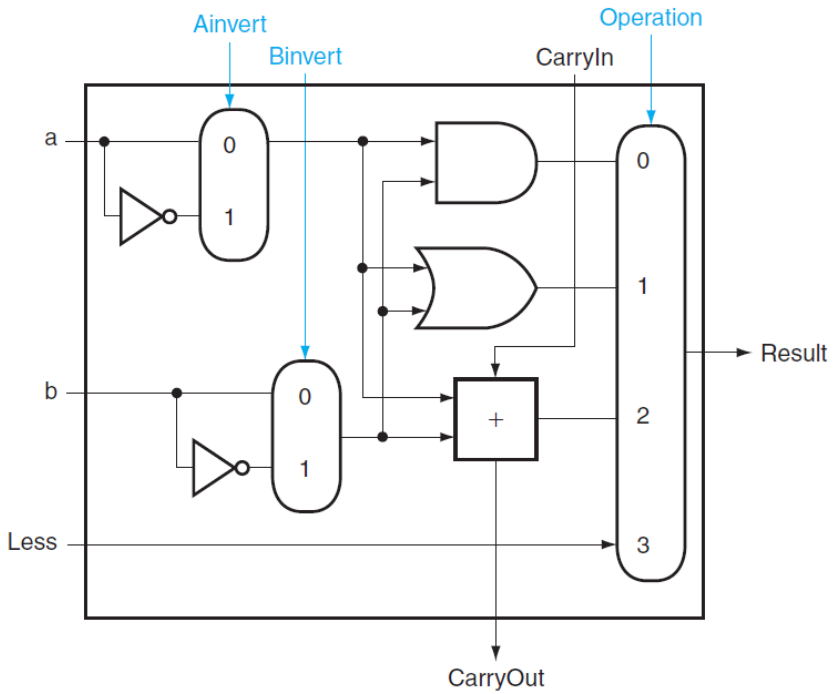


1-bit ALU: AND, OR, ADD, SUB, NOR

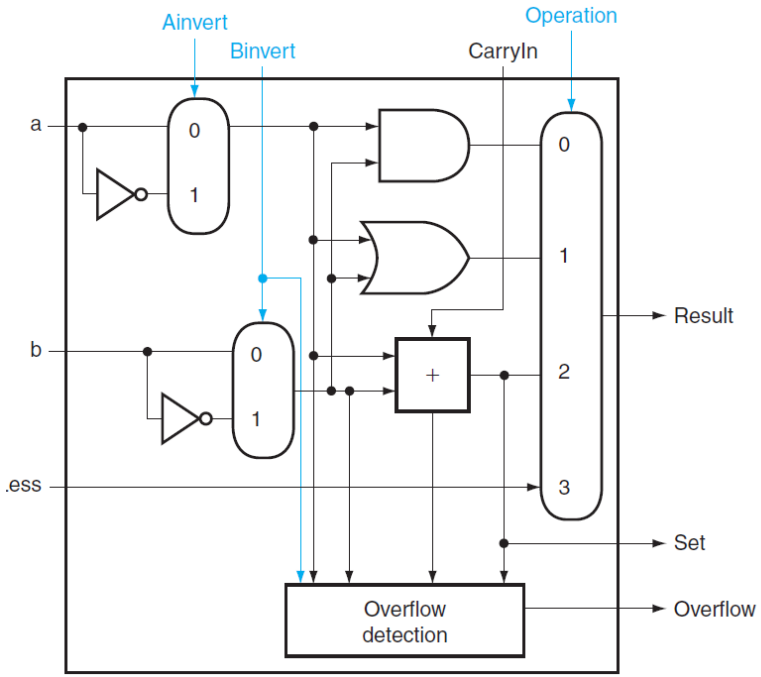
a NOR b = NOT (a OR b) = a' and b'

ALU – Final Version

- Support slt instruction,
  - if a < b => 000…01, otherwise => 0
  - Remember if a < b => SF != OF after a – b



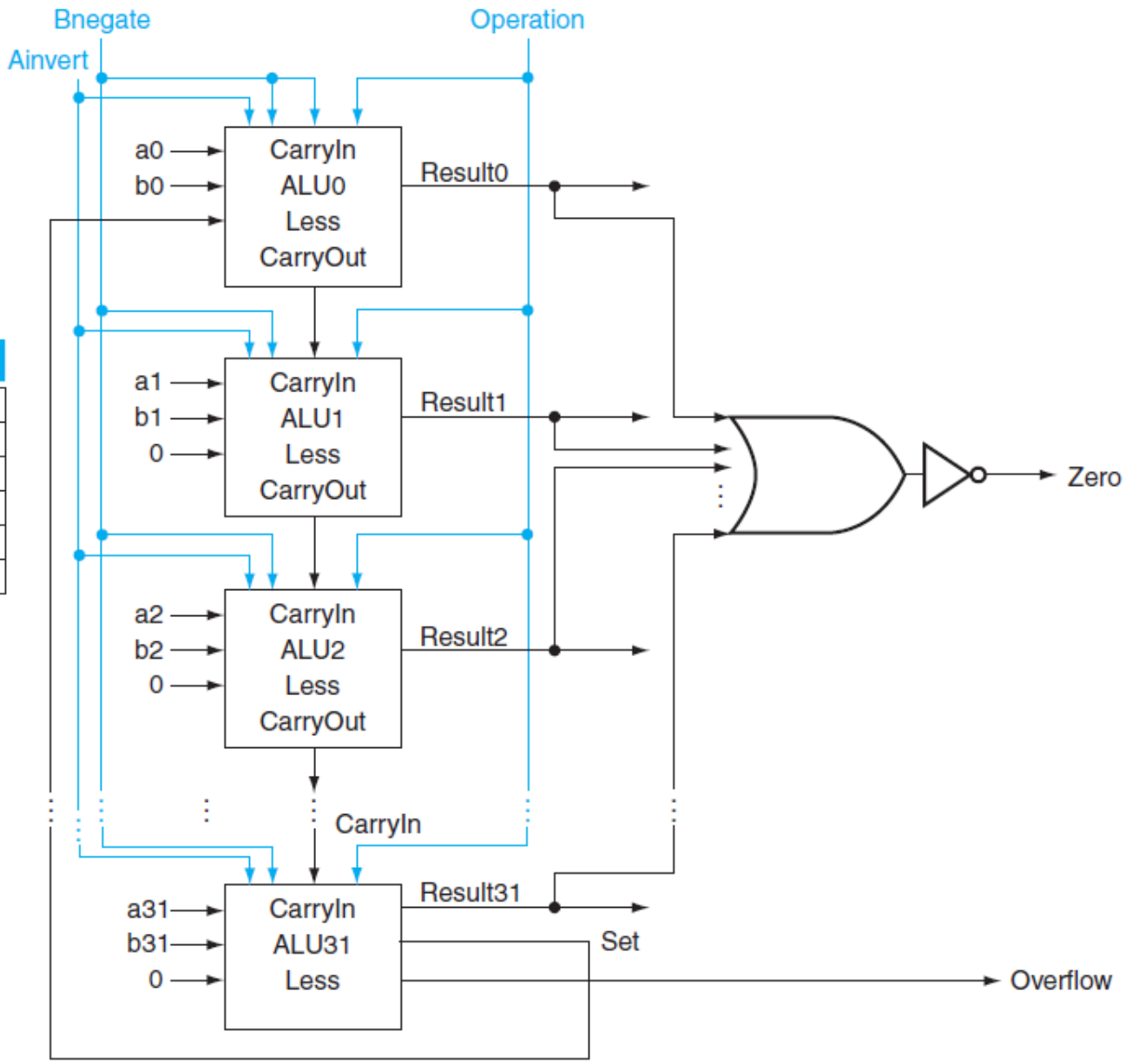
Bit 0 through 30



Bit 31

Complete ALU

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

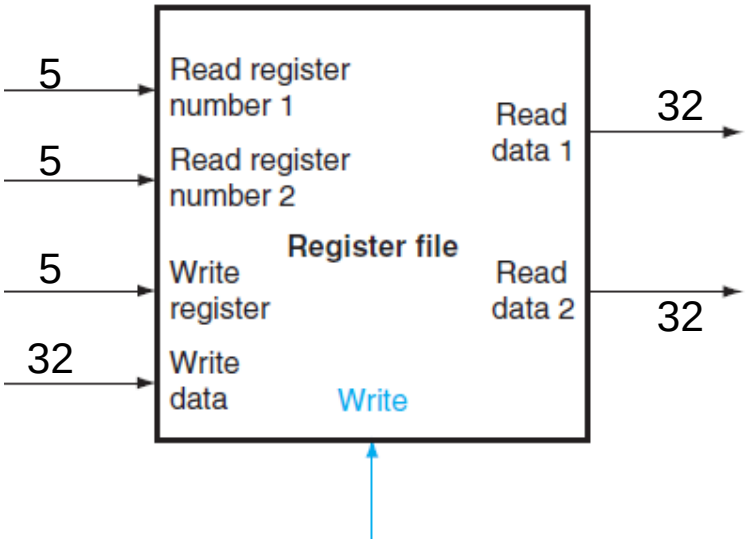


Agenda

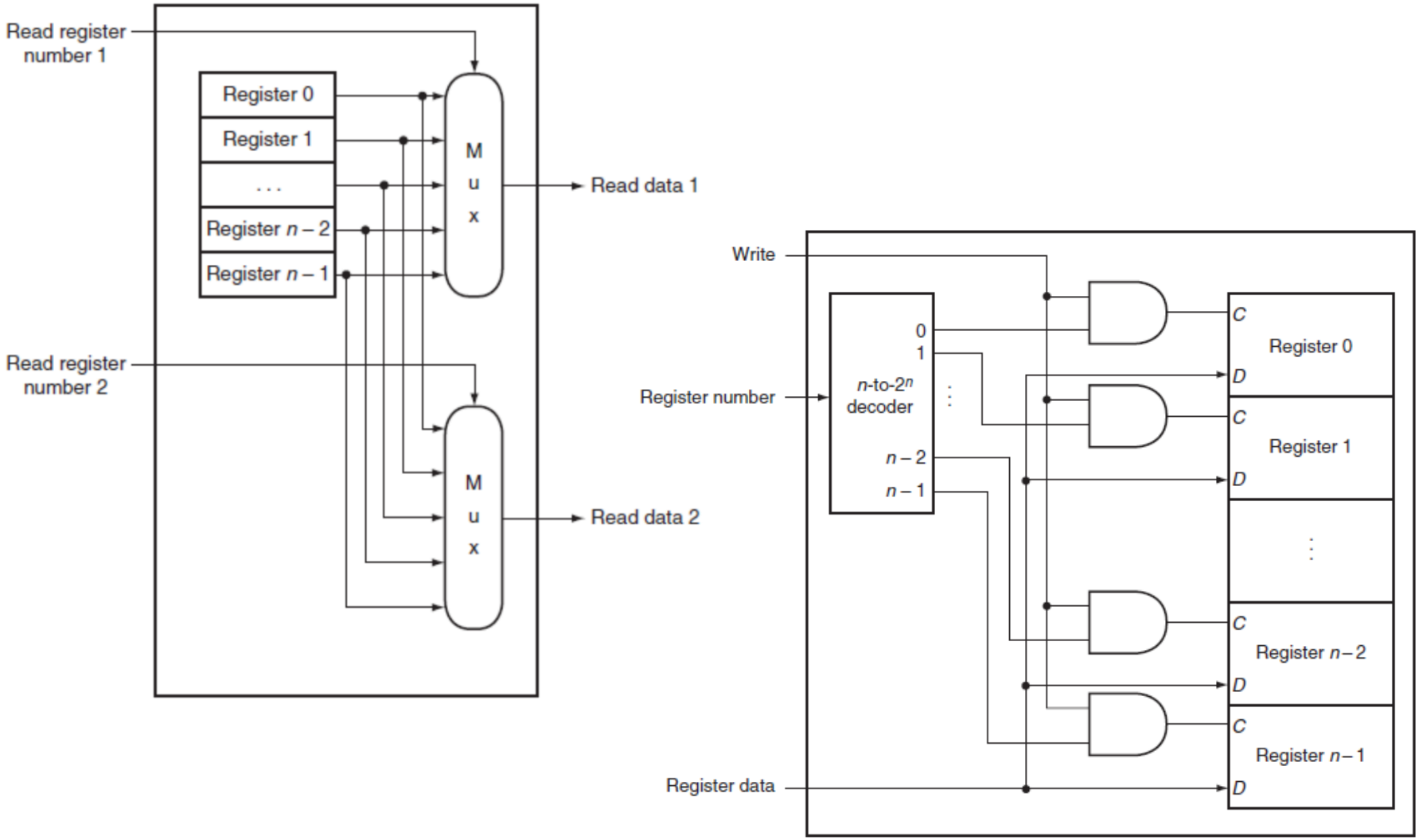
- Combinational Circuits
- Sequential Circuits
- CPU Design
- Basic CPU Elements
  - ALU
  - Register File

Register File

- The 32 registers are arranged as register file
- Using this register file, any two registers can be selected and a third can be written



Register File



Reading

- Appendix B.5, B.8
- Chapter 4: 4.1 - 4.3