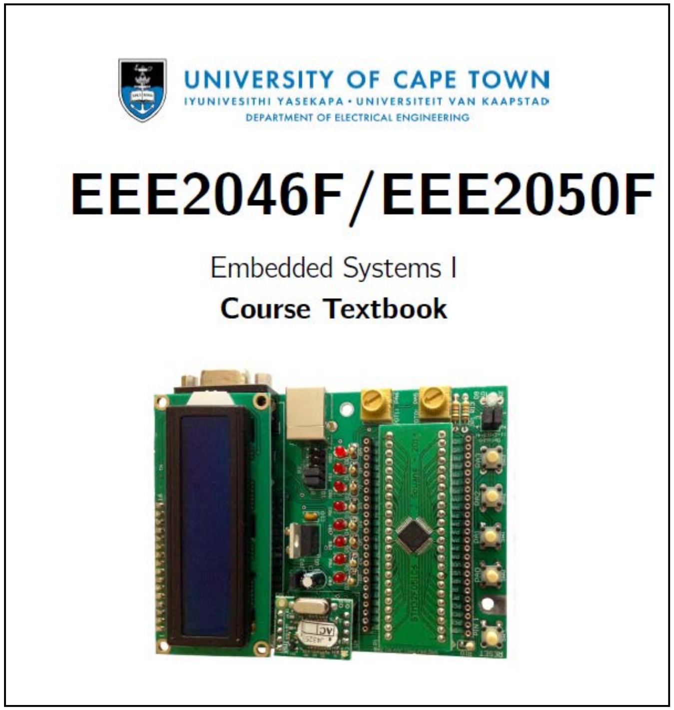
Digital Building Blocks

Arithmetic and Logical Operations using Digital Building Blocks

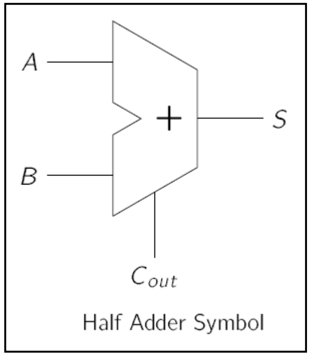
Chapter 4.1 and 4.2
“Digital Building Blocks” by R. Verrinder
EEE2046F/EEE2050F Embedded Systems I notes



Arithmetic and Logical Operations
Overview

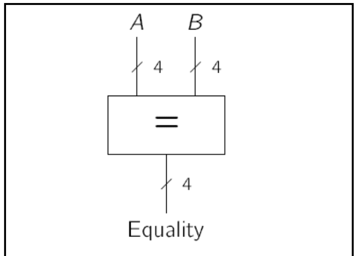
- Digital processors are used to perform arithmetic and logical operations
- Logical circuits can be used to implement these operations

- Arithmetic operations
  - Addition
  - Subtraction



Half Adder Symbol

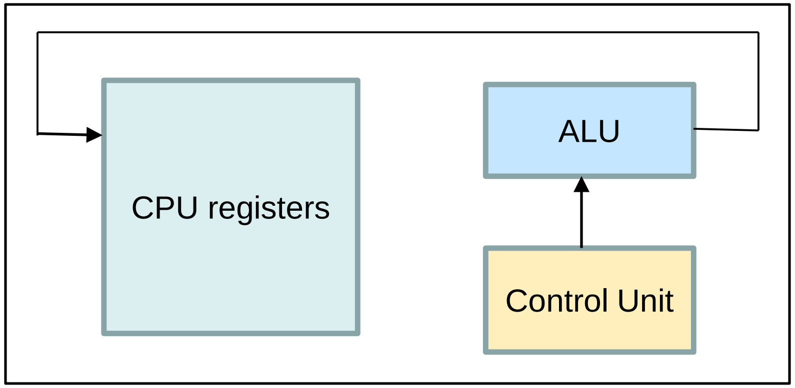
- Logical operations
  - Comparison: equality and magnitude comparators
  - Shifters and rotators: logical shift, arithmetic shift and rotators



4-bit Equality Comparator Symbol

Arithmetic and Logical Operations
Design approach: digital circuits

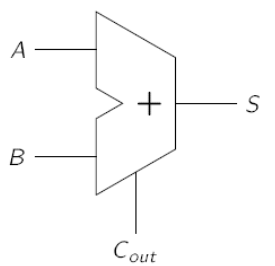
- In digital processors, a CPU is responsible for performing the arithmetic and logical operations
- Two approaches to design circuits for n-bit operations
  - Exhaustive approach: construct the truth table for all possible inputs and outputs. Then use combinational logic techniques to produce the circuit
  - Simple, extendable approach: construct combinational circuits for 2-bit arithmetic and logical operations. Thereafter, extend to n-bit circuits
- The simple, extendable approach is the preferred approach



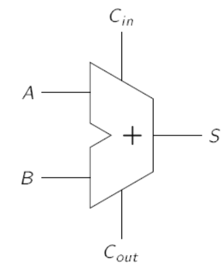
Simplified block diagram of generic CPU

Arithmetic and Logical Operations
Addition: introduction

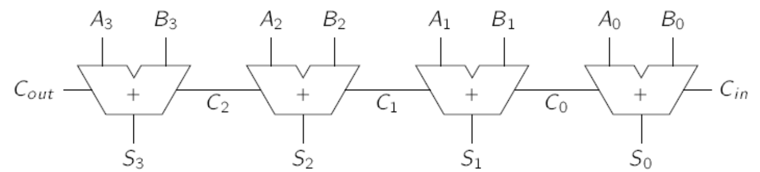
- Adders are used in digital circuits to perform binary addition
- Approach used to design a n-bit adder
  - 1-bit half adder: a ‘carry out’ output exists. No ‘carry in’ input
  - 1-bit full adder: possess both a ‘carry out’ and a ‘carry in’
  - Multi-bit adder: add n-bit binary numbers with a ‘carry out’ and ‘carry in’



Half Adder Symbol



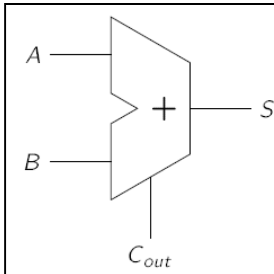
Full Adder Symbol



A 4-bit wide ripple-carry adder

Arithmetic and Logical Operations
Addition: 1-bit half adder

- A 1-bit half adder is a digital circuit used to add two 1-bit numbers: A + B
- Inputs
  - A: a 1-bit binary number
  - B: a 1-bit binary number
- Output
  - S: sum
  - C_{out}: carry out bit



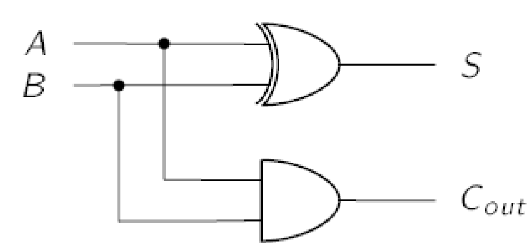
Half Adder Symbol

Input		Output	
A	B	S	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Half Adder truth table

Arithmetic and Logical Operations
Addition: 1-bit half adder

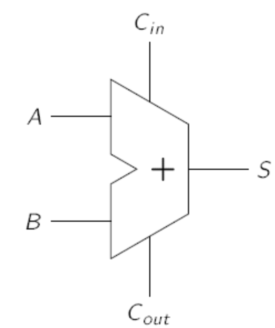
- The truth table can be used to write out the logical expression for the outputs:
  - $S = A \oplus B$
  - $C_{out} = A.B$
- The circuit diagram is made up of an XOR gate and a NAND gate
- Limitation: no 'carry in' input bit



The Half Adder Circuit Diagram.

Arithmetic and Logical Operations
Addition: 1-bit full adder

- A 1-bit full adder is a digital circuit used to add two 1-bit numbers, A + B, while incorporating a ‘carry in’ bit
- Inputs
  - A : a 1-bit binary number
  - B : a 1-bit binary number
  - C_{in}: a 1-bit carry in bit
- Output
  - S : sum (S = A + B + C_{in})
  - C_{out}: carry out bit



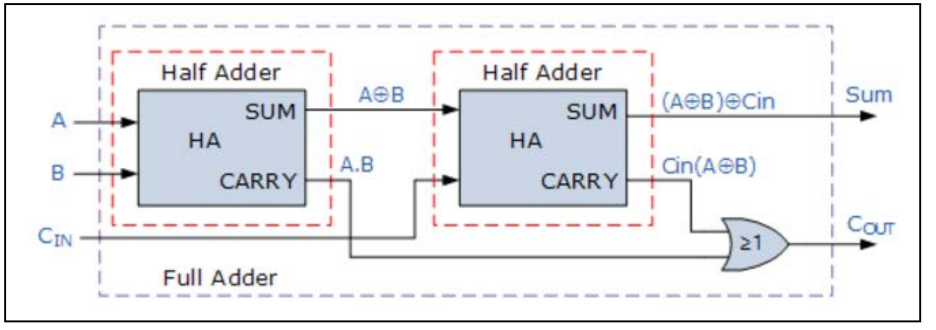
Full Adder Symbol

Input			Output	
C_{in}	A	B	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

Full Adder truth table

Arithmetic and Logical Operations
Addition: 1-bit full adder

- A 1-bit full adder can be constructed by connecting two half adders together



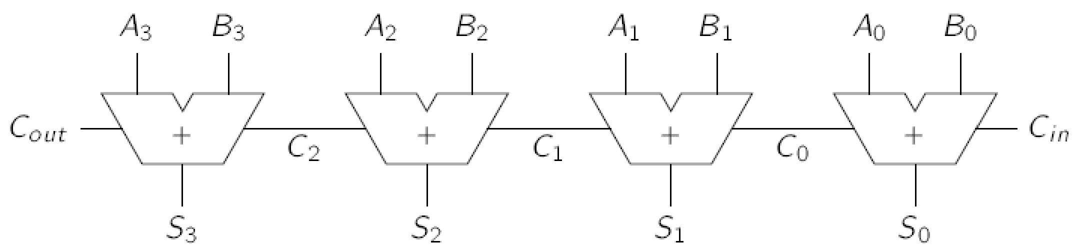
Circuit diagram of a 1-bit full adder (taken from [1])

[1] http://www.electronics-tutorials.ws/combination/comb_7.html

Arithmetic and Logical Operations

Addition: n-bit full adder

- A n-bit full adder performs binary addition of two n-bit numbers
- This adder architecture is known as a ripple-carry adder
- Properties of a ripple-carry adder
  - Speed = n * speed of a 1-bit full adder
- Faster adder architecture: carry-look ahead adder

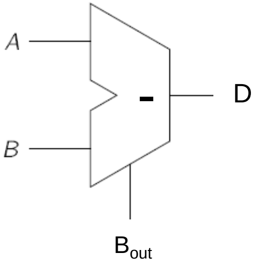


A 4-bit wide ripple-carry adder

Arithmetic and Logical Operations

Subtraction: 1-bit half subtractor

- A 1-bit half subtract or is a digital circuit used to subtract two 1-bit numbers, A and B, to output D = A - B
- Inputs
  - A: a 1-bit binary number
  - B: a 1-bit binary number
- Output
  - D: difference output
  - B_{out}: borrow out bit



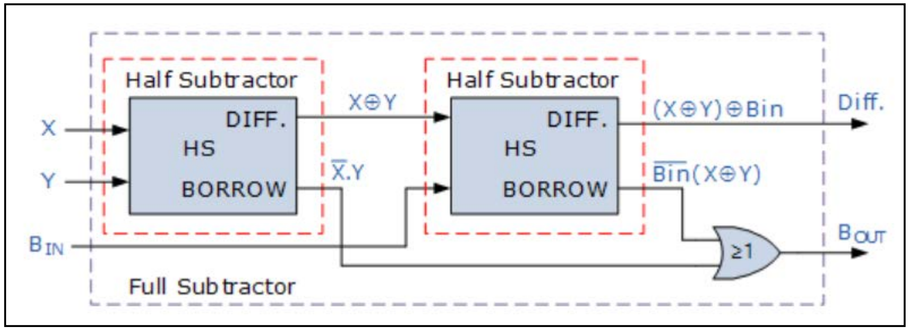
Half subtractor symbol

Input		Output	
A	B	D	B_{out}
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Half subtractor truth table

Arithmetic and Logical Operations
Subtraction: 1-bit full subtractor

- A 1-bit full subtractor is a digital circuit used to subtract two 1-bit numbers, X and Y, while incorporating a 'borrow in' input B_{in}
- Inputs
  - X : a 1-bit binary number
  - Y : a 1-bit binary number
  - B_{in}: a 1-bit borrow in bit
- Output
  - Diff: output (Diff = X – Y – B_{in})
  - B_{out}: borrow out bit



Circuit diagram of a 1-bit full subtractor (taken from [2])

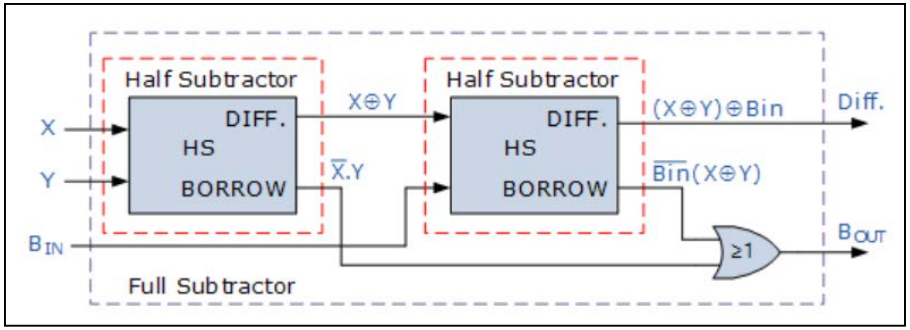
[1] http://www.electronics-tutorials.ws/combination/binary-subtractor.html

Arithmetic and Logical Operations
Subtraction: 1-bit full subtractor

- The truth table of a 1-bit full subtractor: Diff = $X - Y - B_{in}$

Truth Table				
B-in	Y	X	Diff.	B-out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	1
1	1	1	1	1

Truth table of a 1-bit full subtractor (taken from [2])



Circuit diagram of a 1-bit full subtractor (taken from [2])

[2] http://www.electronics-tutorials.ws/combination/binary-subtractor.html

Arithmetic and Logical Operations
Subtraction: binary numbers

- Subtraction of two binary numbers: A – B
- Another approach to perform subtraction
  - Step 1: find the 2's complement of B, which can be expressed as $\overline{B}$ + 1
  - Step 2: add A and the 2's complement of B: A + $\overline{B}$ + 1

A	B	A – B	$\overline{B}$	A + $\overline{B}$ + 1
0	0	0	1	0
0	1	1	0	1
1	0	1	1	1
1	1	0	0	0

Arithmetic and Logical Operations
Subtraction: binary numbers

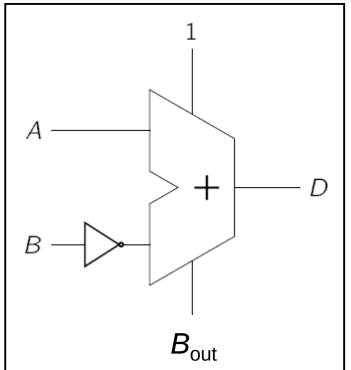
- Subtraction of two binary numbers: A – B
- Another approach to perform subtraction
  - Step 1: find the 2's complement of B, which can be expressed as $\overline{B}$ + 1
  - Step 2: add A and the 2's complement of B: A + $\overline{B}$ + 1

A	B	A – B	$\overline{B}$	A + $\overline{B}$ + 1
0	0	0	1	0
0	1	1	0	1
1	0	1	1	1
1	1	0	0	0

$\therefore$ A – B = A + $\overline{B}$ + 1

Arithmetic and Logical Operations
Subtraction: 1-bit full subtractor

- A 1-bit full adder can be used to perform subtraction: A – B
  - C_{in} = 1
  - Invert B



A Modified Full Adder

Input		Output	
A	B	D	B_{out}
0	0	0	1
0	1	1	$\boxed{0}$
1	0	1	1
1	1	0	1

1-bit subtractor truth table

a '0' represents a borrow from a higher significant bit

a '1' represents a non-borrow from a higher significant bit

- Observe: A – B = A + (-B)

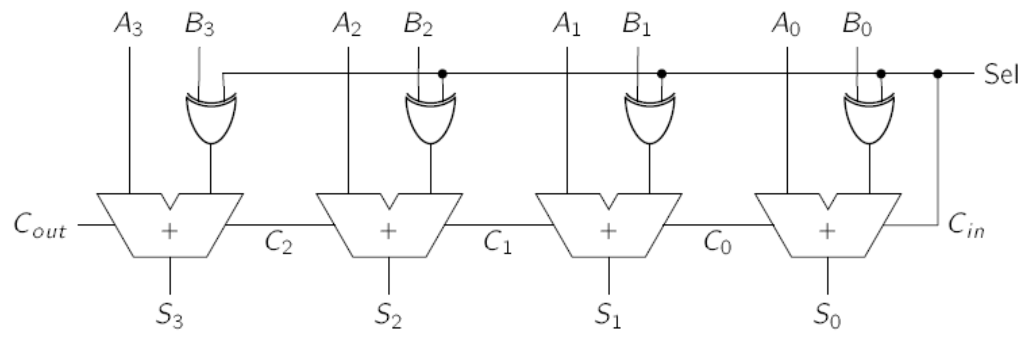
$$= A + (\overline{B} + 1)$$

Arithmetic and Logical Operations

Subtractor: n-bit full subtractor

- A n-bit full adder can be modified to perform the subtraction of two n-bit binary numbers: A – B
- Use Sel line to configure the circuit to be either a n-bit adder or a n-bit subtractor
  - Sel = 0: A + B
  - Sel = 1: A + $\overline{B}$ + 1

$$\text{Sel} \oplus B = B \quad (\text{Sel} = 0)$$
$$\text{Sel} \oplus B = \overline{B} \quad (\text{Sel} = 1)$$

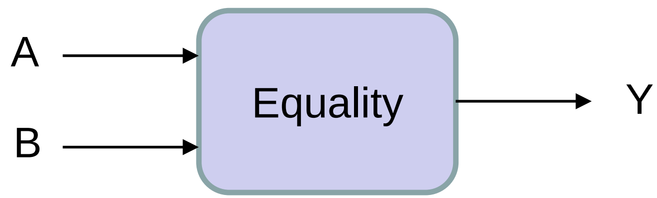


A 4-bit adder-subtractor.

Arithmetic and Logical Operations
Comparators: overview

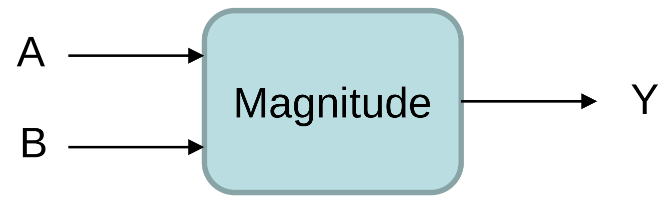
- A comparator is a digital circuit that compares two inputs: A and B.

- Let's look at two types of comparators
  - Equality comparator: output Y is HIGH when inputs are equal: A == B
  - Magnitude comparator: output Y is HIGH when A < B



Inputs		Output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Truth table for the equality comparator



Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	0
1	1	0

Truth table for the magnitude comparator

Arithmetic and Logical Operations

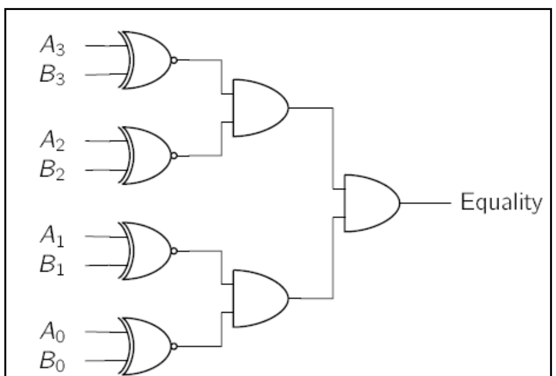
Comparators: equality comparator

- The truth table of a 1-bit equality comparator can be used to write out the Boolean expression of the output Y:

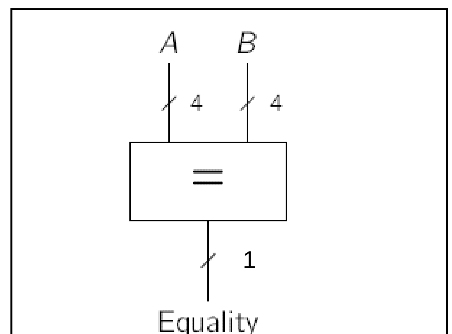
$$\begin{aligned} Y &= \bar{A}\bar{B} + AB \\ &= \overline{(A \oplus B)} \end{aligned}$$

XNOR gate

- A n-bit equality comparator is constructed by connecting N 1-bit equality comparators to a multi-input AND gate



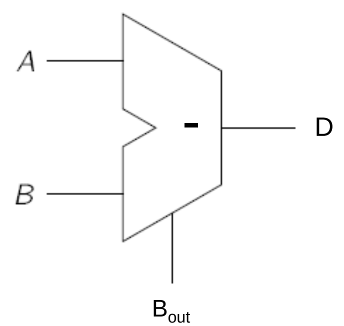
4-bit Equality Comparator



4-bit Equality Comparator Symbol

Arithmetic and Logical Operations
Comparators: magnitude comparator

- A magnitude comparator operation can be performed using a subtractor unit:
  - $B_{out} = 1$: $A < B$
  - $B_{out} = 0$: $A \geq B$



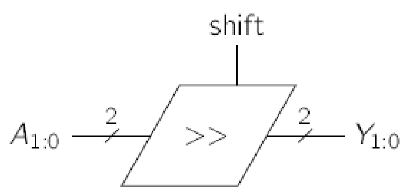
Half subtractor symbol

Input		Output	
A	B	D	B_{out}
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

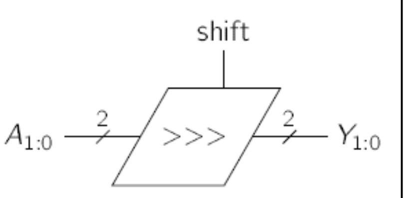
Half subtractor truth table

Arithmetic and Logical Operations
Shifters and Rotators: overview

- Shifters and rotators are logical operations that manipulate the position of bits in a binary string
- The operations shift bits one by one into or out of the MSB or LSB of a binary number
- We will cover two types of shifters and rotators:
  - Logical shift: used for division and multiplication by 2. Assumes unsigned numbers
  - Arithmetic shift: used for division and multiplication by 2. Assumes signed numbers
  - Rotation: circular shift operation



2-bit right logical shifter symbol



2-bit right arithmetic shifter symbol

Arithmetic and Logical Operations

Shifters and Rotators: Logical Shift

- A logical shift operation shifts binary bits either to the left or to the right and the empty bits are replaced by zeros
- Example: A = 1 1 1 0 1 0 1 1
  - Logical 1-bit right shift, A >> 1 gives 0 1 1 1 0 1 0 1
  - Logical 1-bit left shift, A << 1 gives 1 1 0 1 0 1 1 0

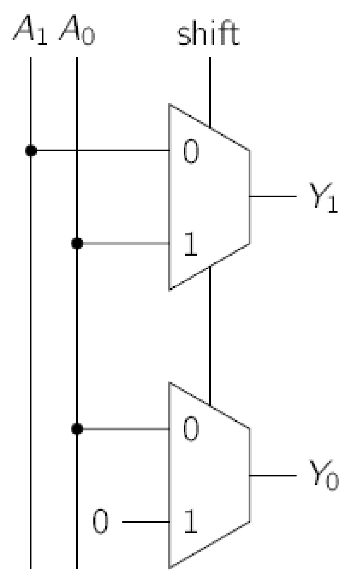
- A n-bit left logical shift is used for fast multiplication by 2^n
- A n-bit right logical shift is used for fast division by 2^n
- Unsigned binary numbers are assumed
- Example
  - Let A = 8: 0 0 0 0 1 0 0 0 (decimal value of 8)
  - A << 1: 0 0 0 1 0 0 0 0 (decimal value of 16) x 2
  - A >> 1: 0 0 0 0 0 1 0 0 (decimal value of 4) ÷ 2

Arithmetic and Logical Operations
Shifters and Rotators: Logical Shift

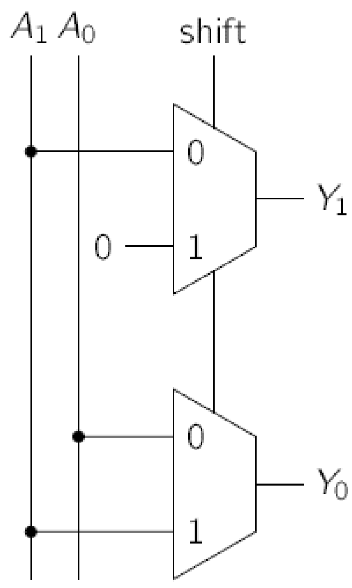
- A 2-bit logical shifter can be constructed using two 1-bit multiplexers

shift = 0
- $Y_1 = A_1$
- $Y_0 = A_0$

shift = 1
- $Y_1 = A_0$
- $Y_0 = 0$



2-bit left logic shifter



2-bit right logic shifter

shift = 0
- $Y_1 = A_1$
- $Y_0 = A_0$

shift = 1
- $Y_1 = 0$
- $Y_0 = A_1$

Arithmetic and Logical Operations
Shifters and Rotators: Arithmetic Shift

- There are two types of arithmetic shift operations:
  - Arithmetic right shift: bits are shifted to the right and empty bits are replaced by the original MSB
  - Arithmetic left shift: bits are shifted to the left and empty bits are replaced by zeros. An arithmetic left shift is equivalent to a logical left shift

- Example: A = 1 1 1 0 1 0 1 1
  - Arithmetic 1-bit right shift, A >>> 1 gives 1 1 1 1 0 1 0 1
  - Arithmetic 1-bit left shift, A <<< 1 gives 1 1 0 1 0 1 1 0

Arithmetic and Logical Operations
Shifters and Rotators: Arithmetic Shift

- Where are arithmetic shifters used?
- A n-bit right arithmetic shift is used for fast division by 2^n
  - Signed binary numbers are assumed
- A n-bit left arithmetic shift is used for fast multiplication by 2^n
  - Signed binary numbers are assumed
- Example
  - Let A = -56 : 1 1 0 0 1 0 0 0 (decimal value of -56)
  - A >>> 1: 1 1 1 0 0 1 0 0 (decimal value of -28) ÷ 2
  - A <<< 1: 1 0 0 1 0 0 0 0 (decimal value of -112) x 2

Arithmetic and Logical Operations

Shifters and Rotators: Arithmetic Shift

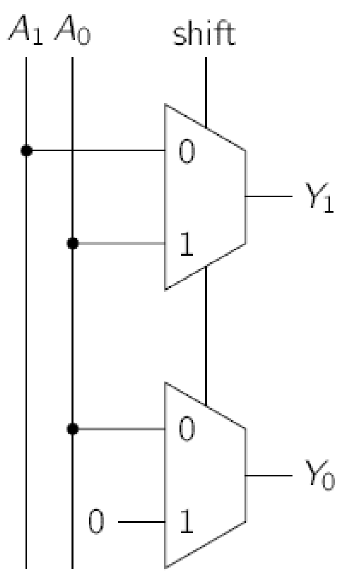
- A 2-bit arithmetic shifter can be constructed using two 1-bit multiplexers

shift = 0

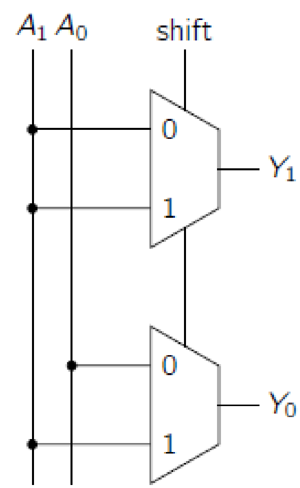
- $Y_1 = A_1$
- $Y_0 = A_0$

shift = 1

- $Y_1 = A_0$
- $Y_0 = 0$



2-bit left arithmetic shifter



(a) 2-bit right arithmetic shifter circuit

2-bit right arithmetic shifter

shift = 0

- $Y_1 = A_1$
- $Y_0 = A_0$

shift = 1

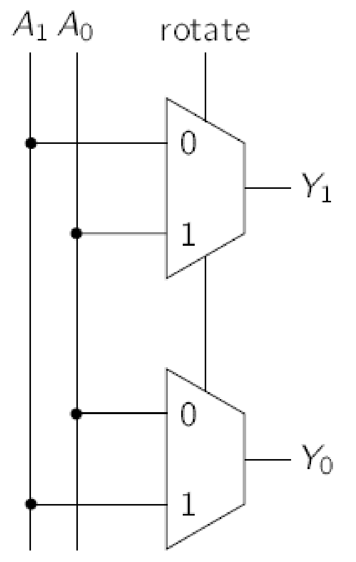
- $Y_1 = A_1$
- $Y_0 = A_1$

Arithmetic and Logical Operations
Shifters and Rotators: Rotators

- A rotator also performs shift operations and empty bits are replaced by the bits that were pushed out
- A rotator performs a circular shift operation
- Example
  - Let A = 1 1 1 0 1 0 1 0
  - Left rotate A: 1 1 0 1 0 1 0 1
  - Right rotate A: 0 1 1 1 0 1 0 1

Arithmetic and Logical Operations
Shifters and Rotators: Rotators

- A 2-bit rotator can be constructed using two 1-bit multiplexers



2-bit right rotator circuit

shift = 0
- $Y_1 = A_1$
- $Y_0 = A_0$

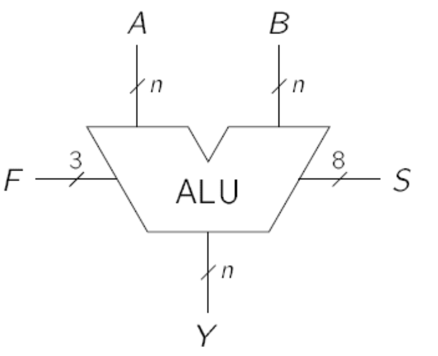
shift = 1
- $Y_1 = A_0$
- $Y_0 = A_1$

Arithmetic and Logical Operations
Arithmetic Logic Unit: overview

- An Arithmetic Logic Unit (ALU) is the heart of a CPU that is responsible for performing a variety of arithmetic, logical and bit shifting operations

- Inputs
  - A: n-bit binary number
  - B: n-bit binary number
  - F: control bus that specifies the opcode

- Outputs
  - Y: n-bit output binary number
  - S: status output bits



A n-bit Arithmetic Logic Unit.

Arithmetic and Logical Operations
Arithmetic Logic Unit: opcodes

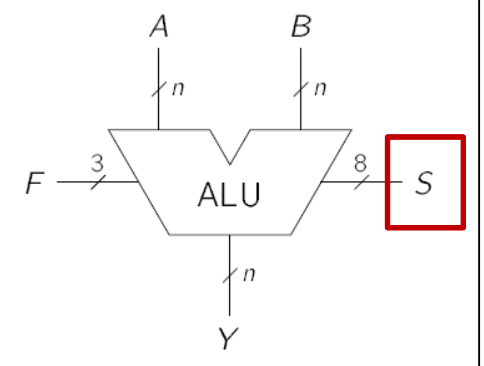
- An opcode specifies the type of operation that must be done on inputs A and B
- An example of opcode values and operations

Opcode operations.

Opcode	Operation
000	A **AND** B
001	A **OR** B
010	$A+B$
011	unused
100	A **AND** $\overline{B}$
101	A **OR** $\overline{B}$
110	A-B
111	**SET LESS THAN** $(A < B)$

Arithmetic and Logical Operations
Arithmetic Logic Unit: status bits

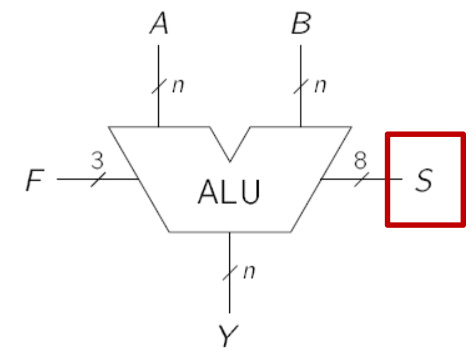
- An status bits S provide information about the last output Y
- Typical flags
  - Carry/Borrow (C):
    - For addition, X = A + B, the C flag is set when there is a carry out of the MSB of the output X. Both A and B are interpreted as unsigned numbers.
    - For subtraction, X = A – B, the C flag is set when there a borrow from the higher significant bit has not taken place. Both A and B are interpreted as unsigned numbers.
  - Zero (Z): Z is set when the last operation resulted in a zero. Typically used to test equality (A == B)



A n-bit Arithmetic Logic Unit.

Arithmetic and Logical Operations
Arithmetic Logic Unit: status bits

- Typical flags
  - Negative or less than (N): N is set when the result of the last operation resulted in a negative value or the MSB is set. The result is interpreted as a signed number
    - If the last operation was an addition operation, N is set when the result is greater than $(2^{n-1} - 1)$ and A and B are n-bit signed numbers
    - If the last operation was a subtraction, A - B. N is set when A < B, where both A and B are unsigned numbers
  - Overflow (V): assumes both A and B are n-bit signed numbers and the last operation caused an overflow: outside of the range $[-(2^{n-1})\quad(2^{n-1} - 1)]$. As a result, the 2's complement result of the last output value will be represented incorrectly.



A n-bit Arithmetic Logic Unit.

Arithmetic and Logical Operations

Arithmetic Logic Unit: Question 1

Design an arithmetic circuit for an ALU with the following instruction set. The opcodes for the instruction set are provided in the table below.

		OPCODES AND OPERATIONS	
F_1	F_0	$C_{in} = 0$	$C_{in} = 1$
0	0	A	$A + 1$
0	1	$A + B$	$A + B + 1$
1	0	$A + \overline{B}$	$A - B$
1	1	$A - 1$	A

Only design the system for the single-bit per input case. You have the following components available to you: 1-bit wide full adders, 2-to-1 multiplexors, AND, OR, NOT, NOR, NAND and XOR gates, decoders etc.

Arithmetic and Logical Operations

Arithmetic Logic Unit: Question 2

Design an arithmetic circuit for an ALU with the following instruction set. The opcodes for the instruction set are provided in the table below.

OPCODES AND OPERATIONS

F_1	F_0	$C_{in} = 0$	$C_{in} = 1$
0	0	$A + B$	$A - B$
0	1	$\overline{A} + B$	$B - A$
1	0	$A - 1$	$A + 1$
1	1	$\overline{A}$	$\overline{A} + 1$

Only design the system for the single-bit case. You have the following components available to you: 1-bit full adders, multi-bit wide multiplexors, AND, OR, NOT, NOR, NAND and XOR gates, decoders etc.

Arithmetic and Logical Operations

Arithmetic Logic Unit: Question 3

Design an arithmetic circuit for the ALU above, which has a 3-bit instruction set. The opcodes in the instruction set are provided in the table below. Only design the system for the 1-bit case. You have the following components available to you: 1-bit full adders, multi-bit wide multiplexors, AND, OR, NOT, NOR, NAND and XOR gates, decoders etc.

Opcode			**Operation**
F_2	F_1	F_0	F
0	0	0	$A+B$
0	0	1	$A-B$
0	1	0	$\overline{A}+B$
0	1	1	$B-A$
1	0	0	$A-1$
1	0	1	$A+1$
1	1	0	$\overline{A}$
1	1	1	$\overline{A}+1$