



CSE211

**Computer Organization and
Design**

Lecture : 3

Tutorial: 1

Practical: 0

Credit: 4

Overview

- Register Transfer Language
- Register Transfer
- **Bus and Memory Transfers**
- Logic Micro-operations
- Shift Micro-operations
- Arithmetic Logic Shift Unit

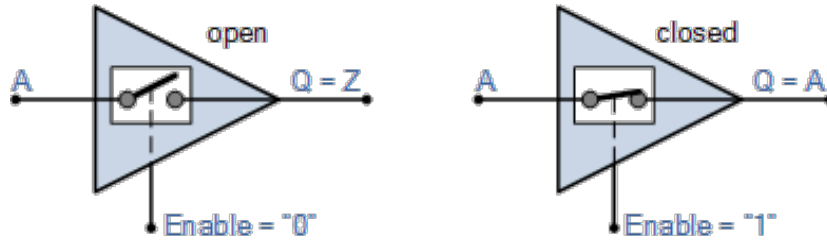
How much storage capacity does each stage in a shift register represent?

- a) One bit
- b) Two bits
- c) Four bits
- d) Eight bits

The bus used to connect the monitor to the CPU is

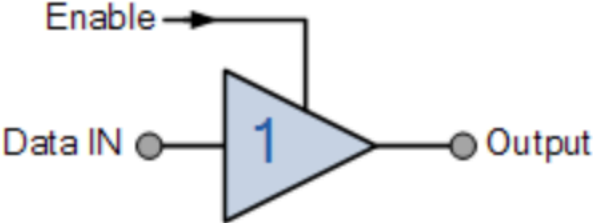
- a) PCI bus
- b) SCSI bus
- c) Memory bus
- d) Rambus

Tri-state Buffer Switch Equivalent



When activated into its third state it disables or turns “OFF” its output producing an open circuit condition that is neither at a logic “HIGH” or “LOW”, but instead gives an output state of very high impedance, **High-Z**, or more commonly Hi-Z. Then this type of device has two logic state inputs, “0” or a “1” but can produce three different output states, “0”, “1” or ” Hi-Z ” which is why it is called a “Tri” or “3-state” device.

Note that this third state is NOT equal to a logic level “0” or “1”, but is an high impedance state in which the buffers output is electrically disconnected from the rest of the circuit. As a result, no current is drawn from the supply.

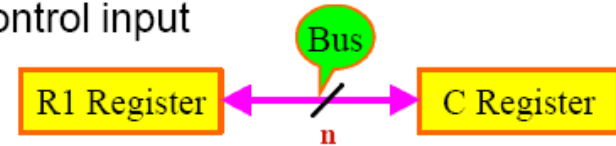
| Symbol | Truth Table | | |
|---|-------------|----|------|
|  <p data-bbox="510 768 761 805">Tri-state Buffer</p> | Enable | IN | OUT |
| | 0 | 0 | Hi-Z |
| | 0 | 1 | Hi-Z |
| | 1 | 0 | 0 |
| | 1 | 1 | 1 |
| Read as Output = Input if Enable is equal to "1" | | | |

Connecting Registers - Bus Transfer

◆ Bus Transfer

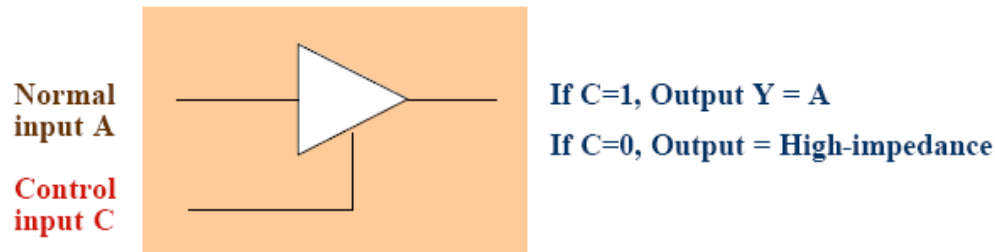
- The content of register C is placed on the bus, and the content of the bus is loaded into register R1 by activating its load control input

$$\left. \begin{array}{l} Bus \leftarrow C, R1 \leftarrow Bus \\ R1 \leftarrow C \end{array} \right\} =$$



◆ Three-State Bus Buffers

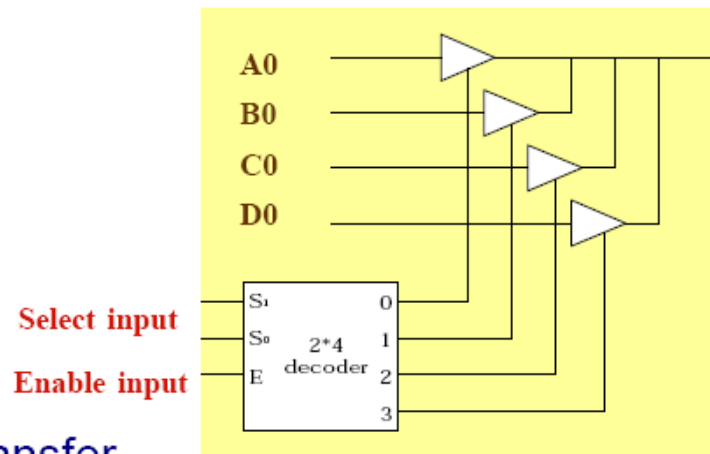
- A bus system can be constructed with **three-state gates** *instead of multiplexers*
- Tri-State : 0, 1, High-impedance (**Open circuit**)
- Buffer
 - » A device designed to be inserted between other devices to match impedance, to prevent mixed interactions, and **to supply additional drive or relay capability**
 - » Buffer types are classified as inverting or noninverting
- Tri-state buffer gate : Fig. 4-4
 - » When control input = 1 : The output is enabled (output Y = input A)
 - » When control input = 0 : The output is disabled (output Y = high-impedance)



Connecting Registers - Bus Transfer

◆ The construction of a bus system with tri-state buffer : *Fig.*

- The outputs of four buffer are connected together to form a single bus line (Tri-state buffer)
- No more than one buffer may be in the active state at any given time (2 X 4 Decoder)
- To construct a common bus for 4 register with 4 bit : Fig.



AR: Address Reg.
DR: Data Reg.
M : Memory Word(Data)

READ : $DR \leftarrow M[AR]$

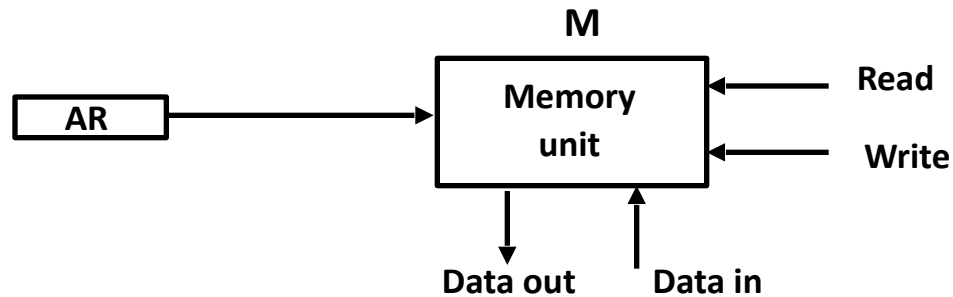
WRITE : $M[AR] \leftarrow R1$

◆ Memory Transfer

- Memory read : A transfer information into DR from the memory word M selected by the address in AR
- Memory Write : A transfer information from R1 into the memory word M selected by the address in AR

Memory Transfer

Memory is usually accessed in computer systems by putting the desired address in a special register, the Memory Address Register (MAR, or AR)



Memory Read

- To read a value from a location in memory and load it into a register, the register transfer language notation looks like this:

$$R1 \leftarrow M[MAR]$$

- This causes the following to occur
 1. The contents of the MAR get sent to the memory address lines
 2. A Read (= 1) gets sent to the memory unit
 3. The contents of the specified address are put on the memory's output data lines
 4. These get sent over the bus to be loaded into register R1

Memory Write

- To write a value from a register to a location in memory looks like this in register transfer language:

$$M[MAR] \leftarrow R1$$

- This causes the following to occur
 1. The contents of the MAR get sent to the memory address lines
 2. A Write (= 1) gets sent to the memory unit
 3. The values in register R1 get sent over the bus to the data input lines of the memory
 4. The values get loaded into the specified address in the memory

SUMMARY OF R. TRANSFER MICROOPERATIONS

$A \leftarrow B$

1. Transfer content of reg. B into reg. A

$A \leftarrow \text{constant}$

3. Transfer a binary constant into reg. A

$ABUS \leftarrow R1, R2 \leftarrow ABUS$

4. Transfer content of R1 into bus A and, at the same time, transfer content of bus A into R2

AR

5. Address register

DR

6. Data register

$M[R]$

7. Memory word specified by reg. R

M

8. Equivalent to $M[AR]$

$DR \leftarrow M$

9. Memory *read* operation: transfers content of memory word specified by AR into DR

$M \leftarrow DR$

10. Memory *write* operation: transfers content of DR into memory word specified by AR

MICROOPERATIONS

Computer system microoperations are of four types:

- **Register transfer microoperations**
- **Arithmetic microoperations**
- **Logic microoperations**
- **Shift microoperations**

Arithmetic MICROOPERATIONS

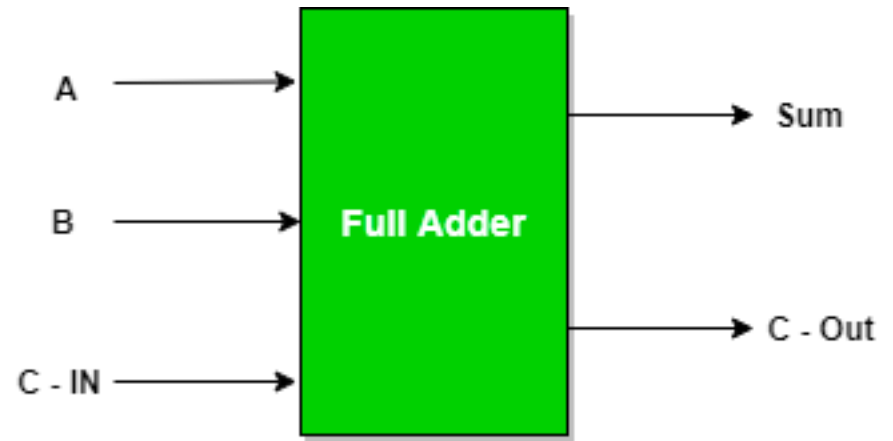
- **The basic arithmetic microoperations are**
 - Addition
 - Subtraction
 - Increment
 - Decrement

- **The additional arithmetic microoperations are**
 - Add with carry
 - Subtract with borrow
 - Transfer/Load
 - etc. ...

Summary of Typical Arithmetic Micro-Operations

| | |
|------------------------------|--|
| $R3 \leftarrow R1 + R2$ | Contents of R1 plus R2 transferred to R3 |
| $R3 \leftarrow R1 - R2$ | Contents of R1 minus R2 transferred to R3 |
| $R2 \leftarrow R2'$ | Complement the contents of R2 |
| $R2 \leftarrow R2' + 1$ | 2's complement the contents of R2 (negate) |
| $R3 \leftarrow R1 + R2' + 1$ | subtraction |
| $R1 \leftarrow R1 + 1$ | Increment |
| $R1 \leftarrow R1 - 1$ | Decrement |

Full Adder is the adder which adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S which is SUM.



| Inputs | | | Outputs | |
|--------|---|--------|---------|---------|
| A | B | C - IN | Sum | 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 |

Binary Adder

◆ 4-bit Binary Adder : *Fig. 4-6*

- Full adder = 2-bits sum + previous carry
- Binary adder = the arithmetic sum of two binary numbers of any length
- c_0 (input carry), c_4 (output carry)

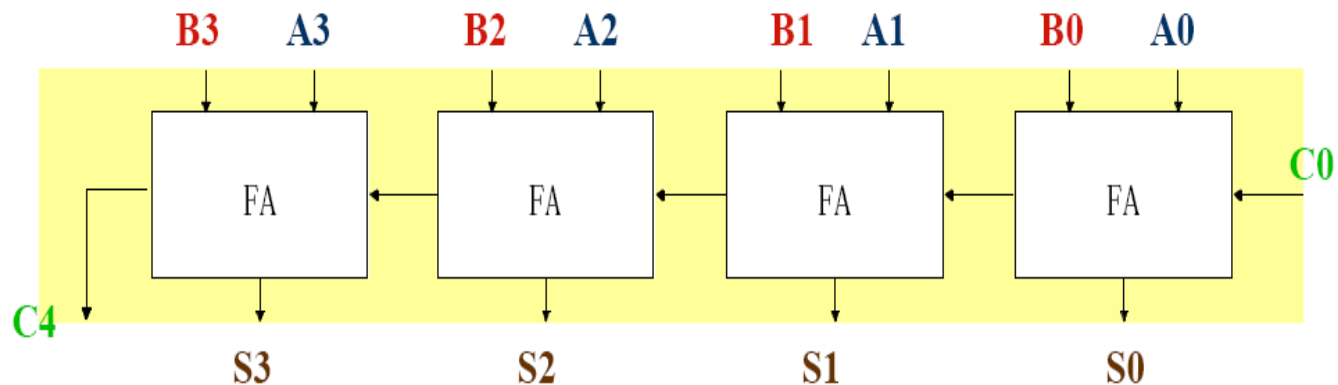
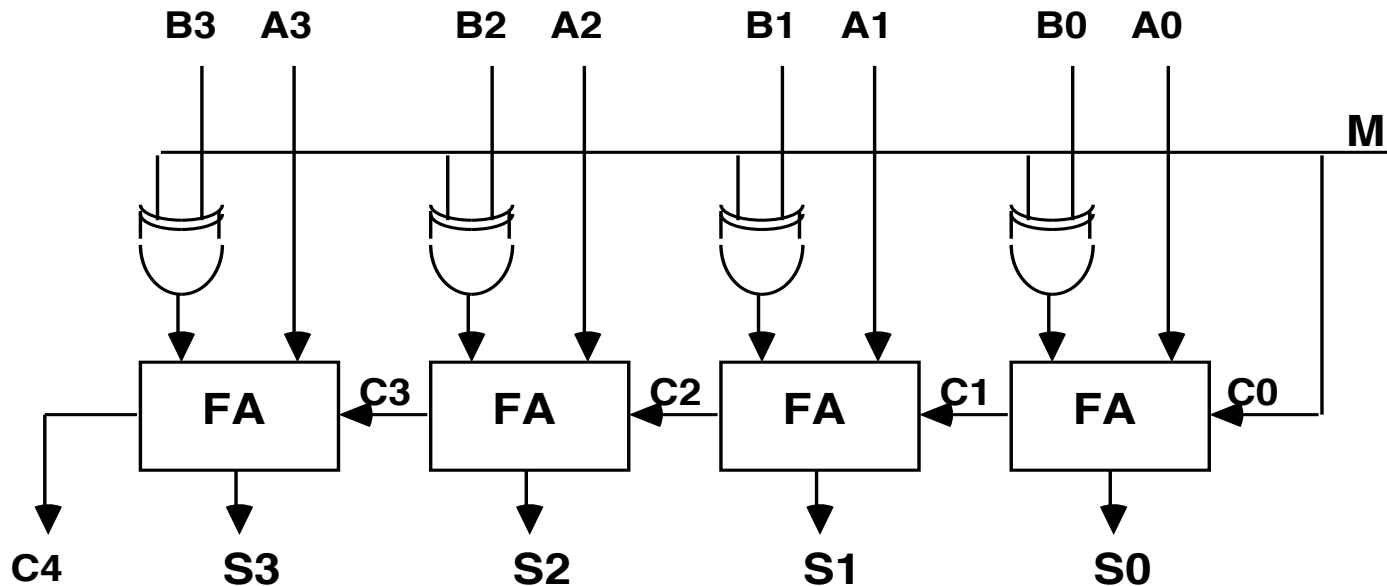


Figure 4-6. 4-bit binary adder

Binary Adder-Subtractor

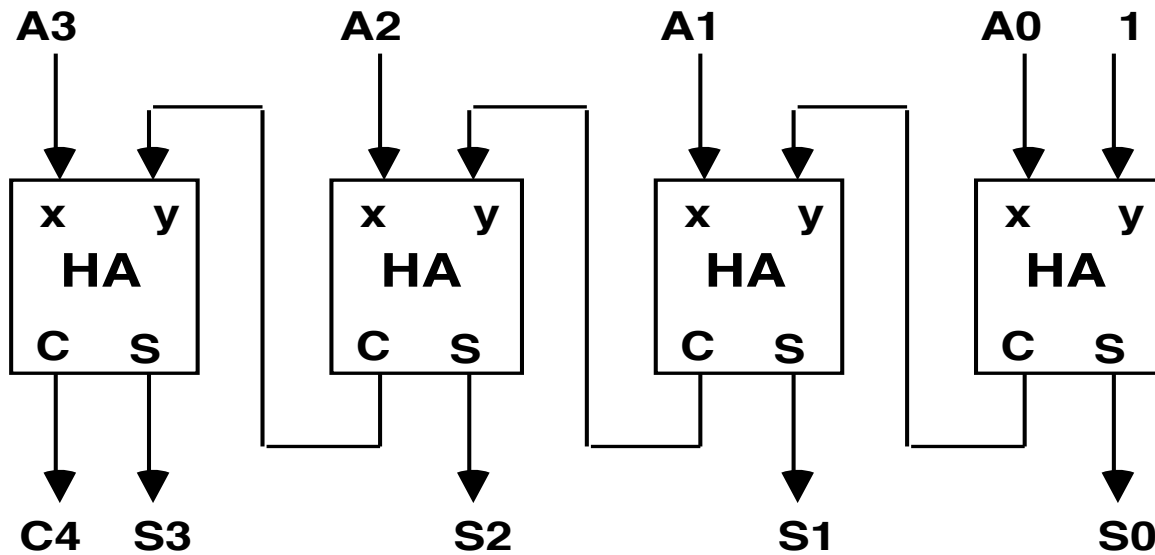
Binary Adder-Subtractor



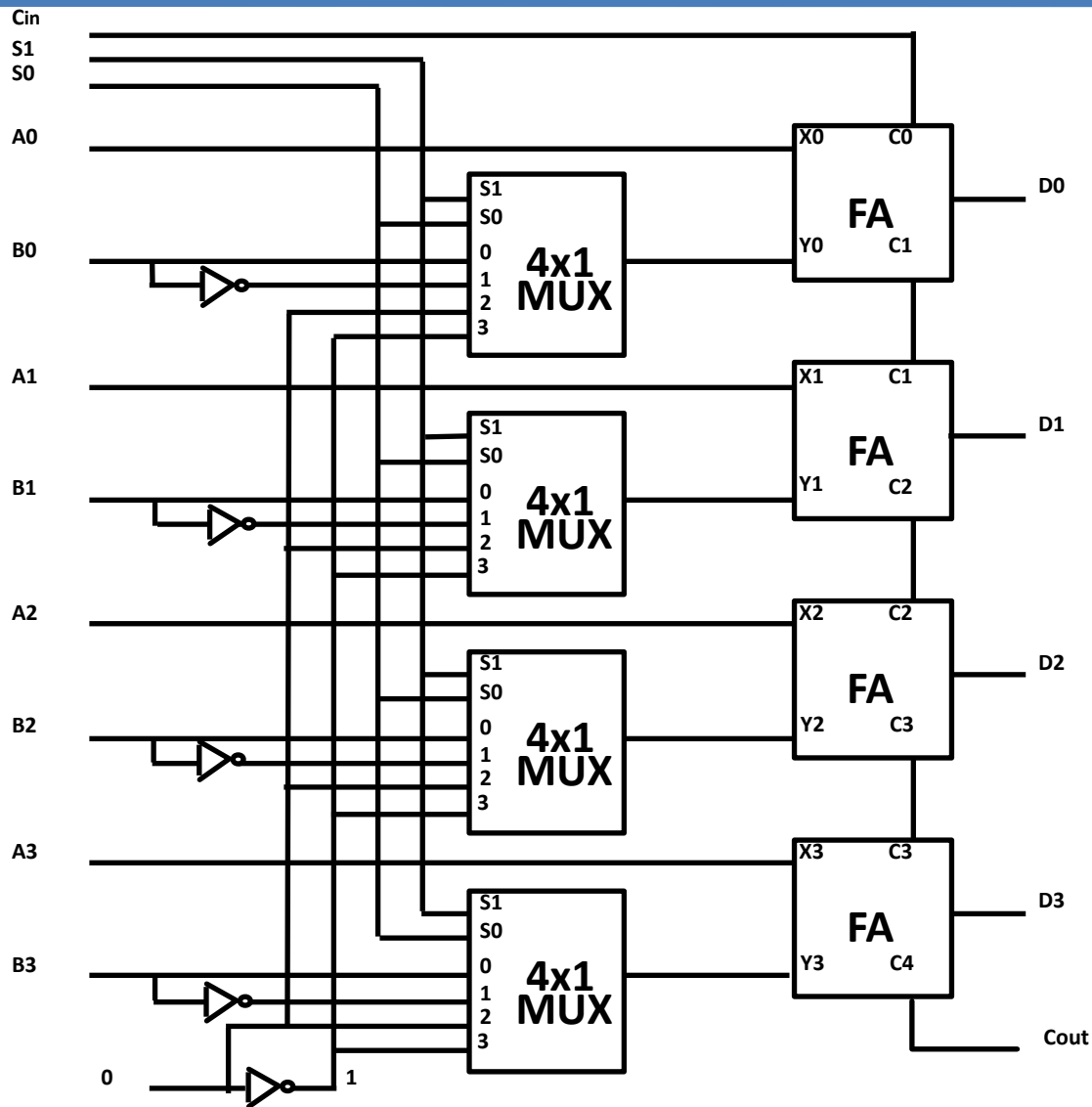
- Mode input M controls the operation
 - M=0 ---- adder
 - M=1 ---- subtractor

Binary Incrementer

Binary Incrementer



Arithmetic Circuits



| Select | | Input | Output | |
|--------|-------|----------|--------|------------|
| S_1 | S_0 | C_{in} | Y | |
| 0 | 0 | 0 | B | $D=A+B$ |
| 0 | 0 | 1 | B | $D=A+B+1$ |
| 0 | 1 | 0 | B' | $D=A+B'$ |
| 0 | 1 | 1 | B' | $D=A+B'+1$ |
| 1 | 0 | 0 | 0 | $D=A$ |
| 1 | 0 | 1 | 0 | $D=A+1$ |
| 1 | 1 | 0 | 1 | $D=A-1$ |
| 1 | 1 | 1 | 1 | $D=A$ |