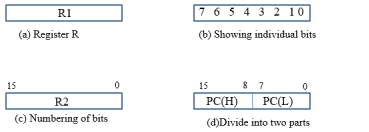
Chapter 4 - Register Transfers & Microoperations

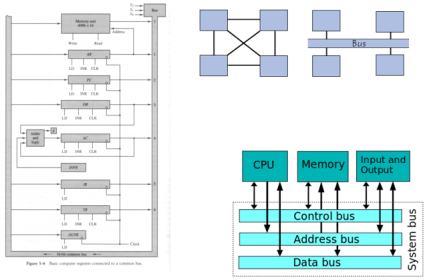
Microoperations: The operations executed on data stored in registers of a digital system (e.g., shift, count, clear, load, add).

RTL (register transfer language): describes the possible microoperations, specifies source and destination registers.

Notation: Registers denoted with capital letters and numbers e.g., PC, IR, R1



Why Bus is Required in a Computer System



Four Categories of microoperations

1. Register transfer
2. Arithmetic
3. Logic
4. Shift

All microoperations must have underlying hardware

In this chapter, we build the hardware to execute microoperations

Arithmetic microoperations

Symbolic Designation	Description
R3←R1+R2	Contents of R1 plus R2 transferred to R3
R3←R1-R2	Contents of R1 minus R2 transferred to R3
R2←$\overline{R2}$	Complement the contents of R2 (1's complement)
R2←$\overline{R2}$+1	2's complement the contents of R2 (negate)
R3←R1+$\overline{R2}$+1	R1 plus the 2's complement of R2 (subtraction)
R1←R1+1	Increment the contents of R1 by one
R1←R1-1	Decrement the contents of R1 by one

Logic Microops

Bit-wise logic operations on the contents of two registers.

Notation:
∧ - AND
∨ - OR
⊕ - XOR

Example

```
10101011   R1
10100011   R2
-----------------------------
10101011   R1 ∨R2
```

P+Q: R1←R2∨R3

Denotes OR of two control signals

denotes OR microop

P+Q: R1←R2+R3

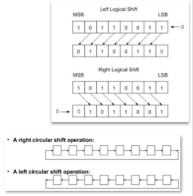
denotes arithmetic plus (addition)

Shift Microoperations

logical shift (shl/shr)
circular shift (cil/cir)
arithmetic shift (ashl/ashr)

ashl multiplies number by 2
ashr divides number by 2

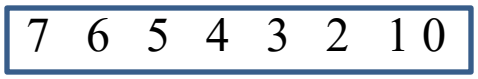
Symbolic designation	Description
R←shl R	Shift-left register R
R←shr R	Shift-right register R
R←cil R	Circular shift-left register R
R←cir R	Circular shift-right register R
R←ashl R	Arithmetic shift-left R
R←ashr R	Arithmetic shift-right R



Notation: Registers denoted with capital letters and numbers

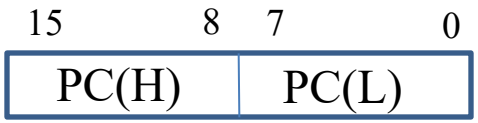
e.g., PC, IR, R1

(a) Register R



(b) Showing individual bits

(c) Numbering of bits



(d)Divide into two parts

Microperations:

Register load microop:

$R2 \leftarrow R1$

Contents of R1 placed into R2.
(Contents of R1 left unchanged)

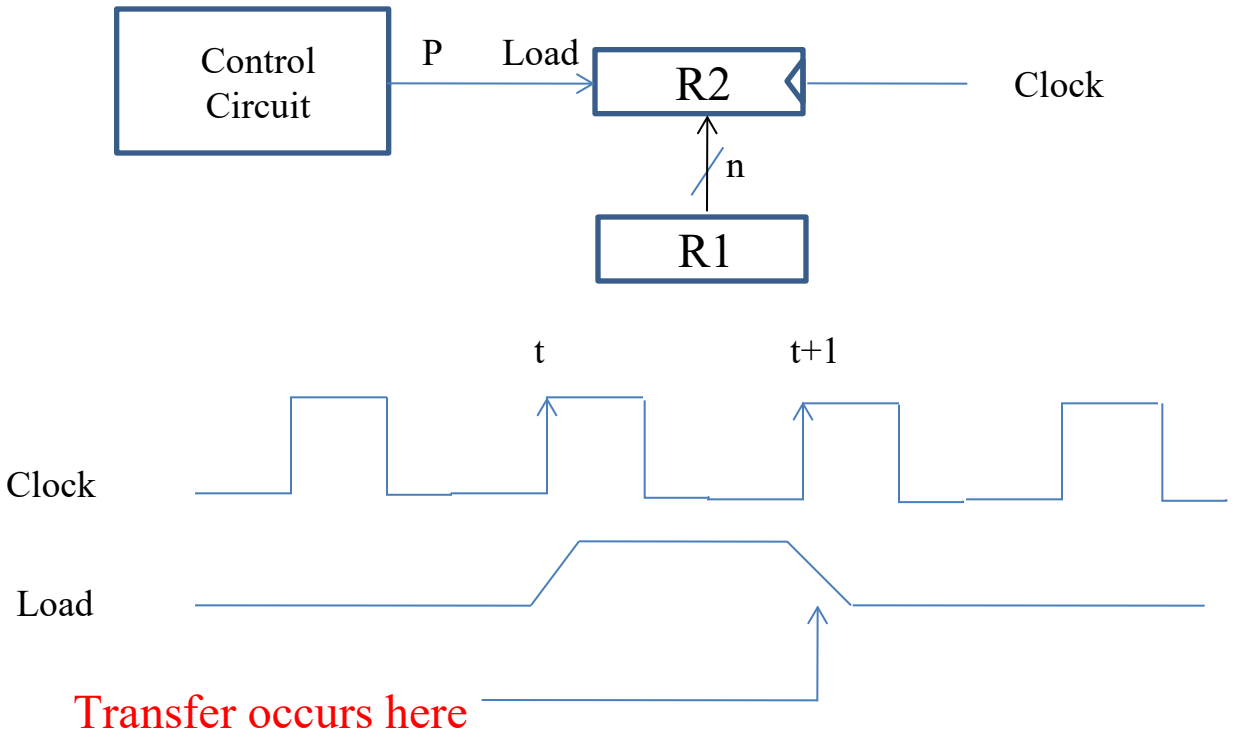
Control function microop:

$P: R2 \leftarrow R1$

If (P=1) then (R2←R1)

P is a Boolean variable known as a control signal.

Hardware construction of *P: R2 ← R1*



Concurrent operations can also be expressed in a single line of RTL by separating basic operations with a comma

e.g., $T: R2 \leftarrow R1, \quad R1 \leftarrow R3$

Basic Symbols for Register Transfers

Symbol	Description	Examples
Letters (and numerals)	*Denotes a register*	MAR, R2
Parentheses ( )	*Denotes a part of a register*	R2(0-7),R2(L)
Arrow ←	*Denotes transfer of information*	R2←R1
Comma ,	*Separates two microoperations*	R2←R1,R1←R2

Why Bus is Required in a Computer System

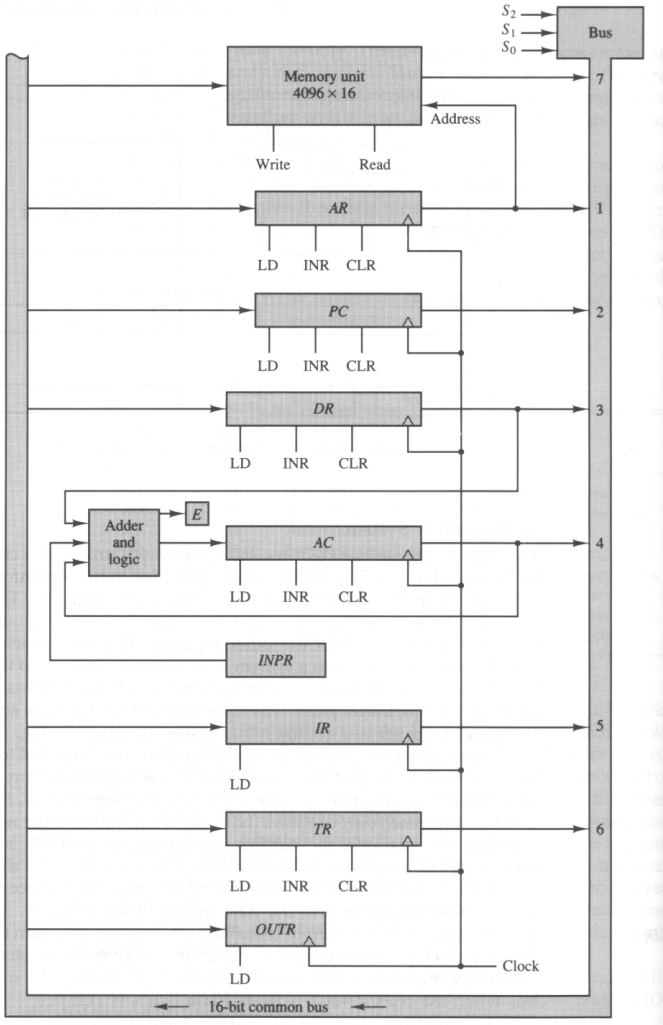
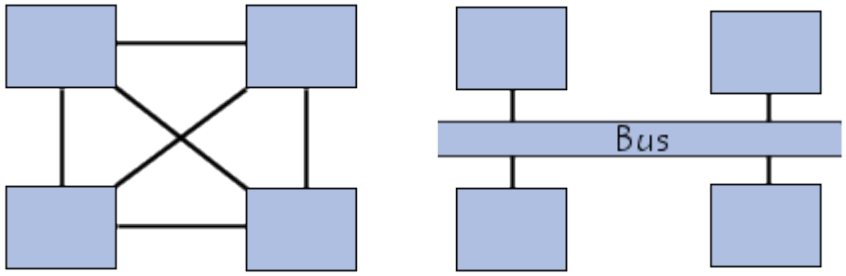
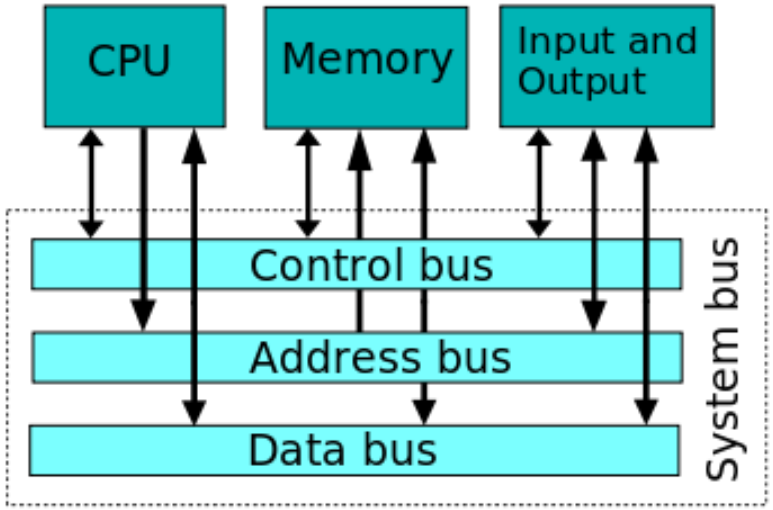


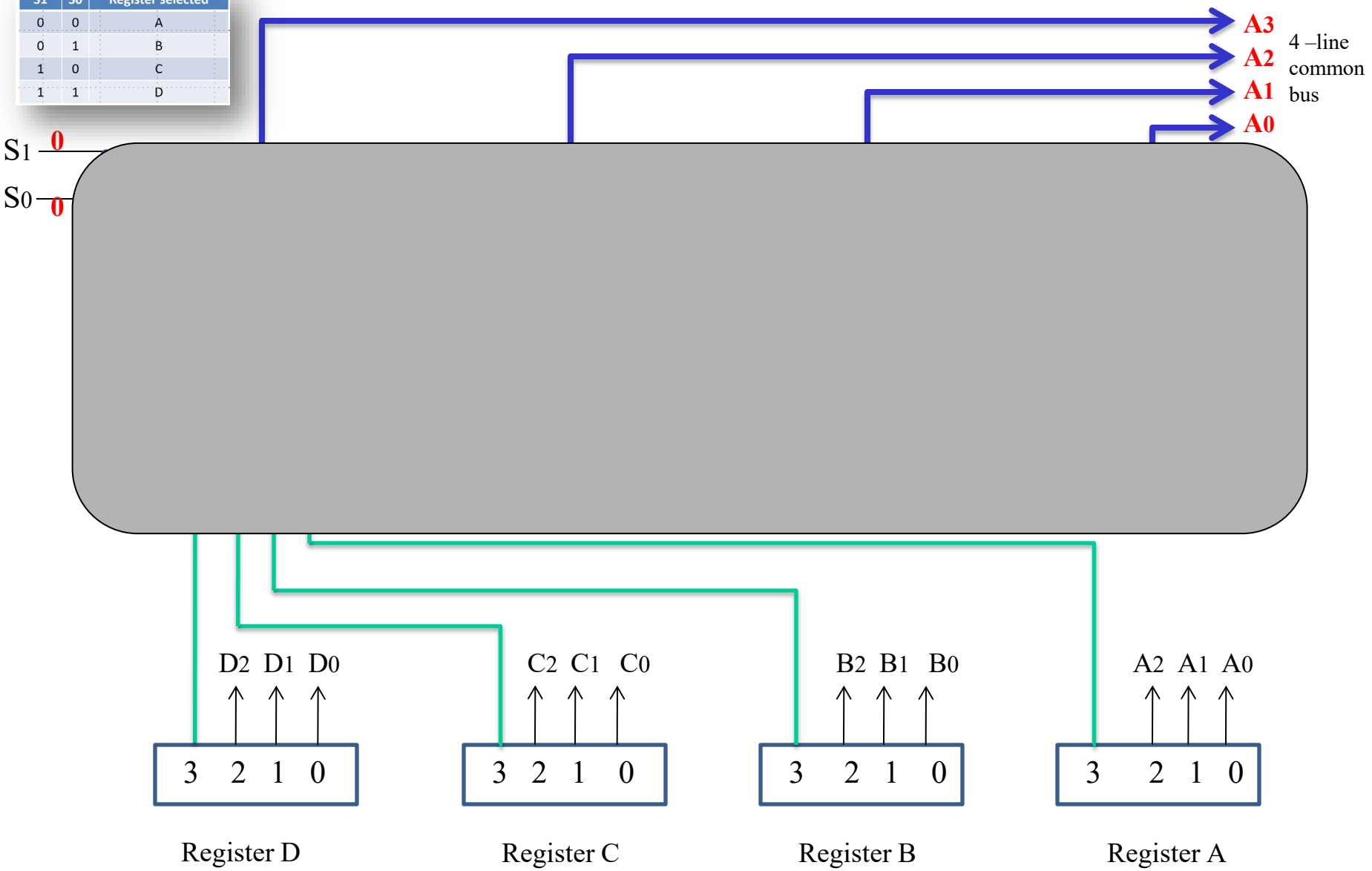
Figure 5-4 Basic computer registers connected to a common bus.





Bus System

S1	S0	Register selected
0	0	A
0	1	B
1	0	C
1	1	D



Q: How many MUXs needed to produce an *n*-line bus for *k* registers?

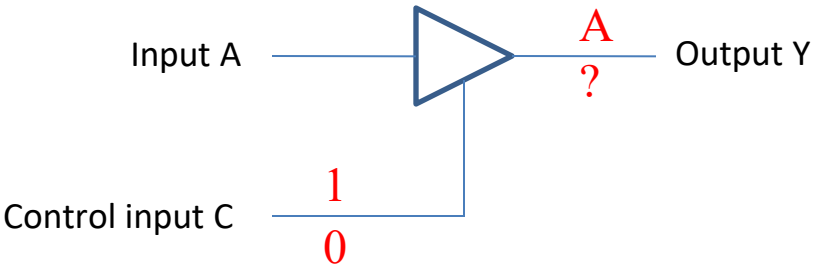
A: *n* MUXs, where each MUX is $k \times 1$.

Representing data transfer through Bus

Bus for information transfer between registers:
BUS ← C, R1 ← BUS

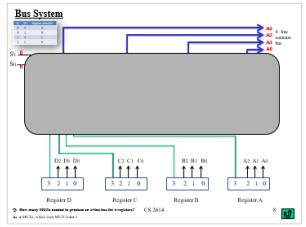
If we know the bus exists:
R1 ← C

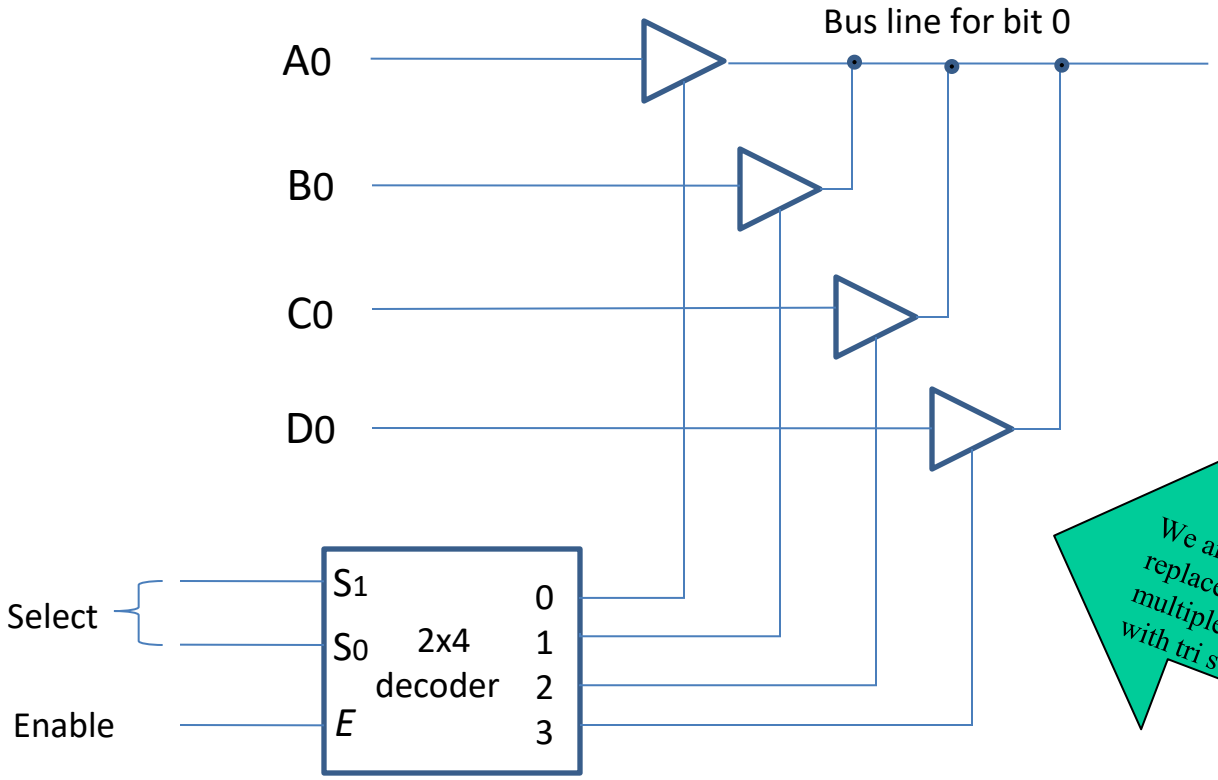
Three (Tri-) State Buffers



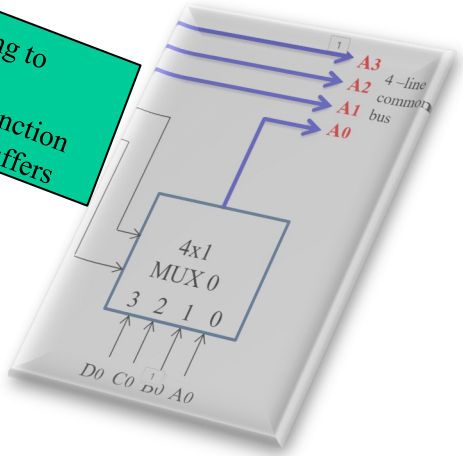
Output Y=A if C=1
High-impedance if C=0

Building a Bus with tri-state buffers

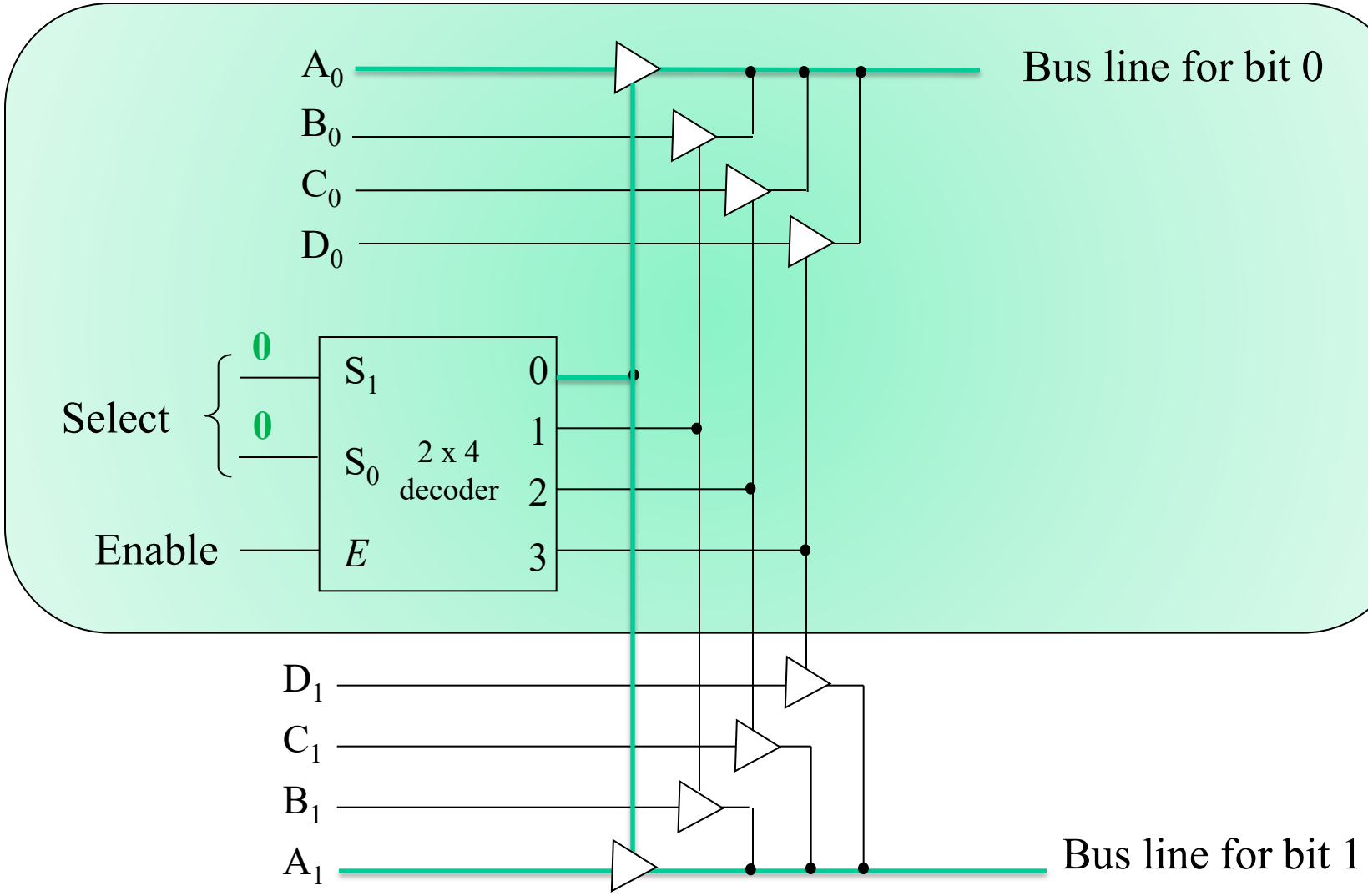




We are trying to replace the multiplexer function with tri state buffers

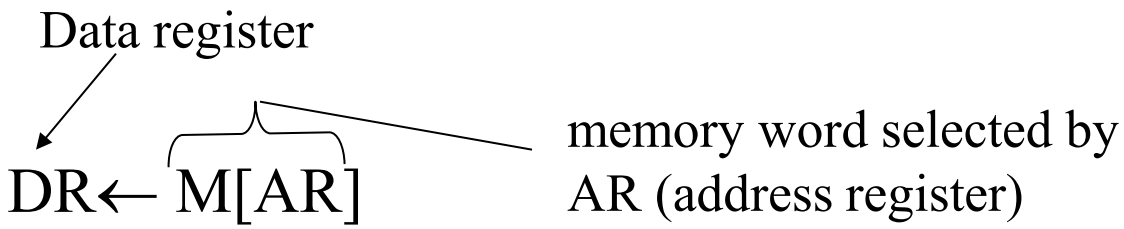


Q: How many decoders needed for an n-line bus?
A: Still just one!


A0
B0
C0
D0
Bus line for bit 0
0
0
Select
S1
S0
2 x 4
decoder
0
1
2
3
Enable
E
D1
C1
B1
A1
Bus line for bit 1

Memory Transfers

Read:



Write:

M[AR]← DR

Four Categories of microoperations

1. Register transfer
2. Arithmetic
3. Logic
4. Shift

All microoperations must have underlying hardware

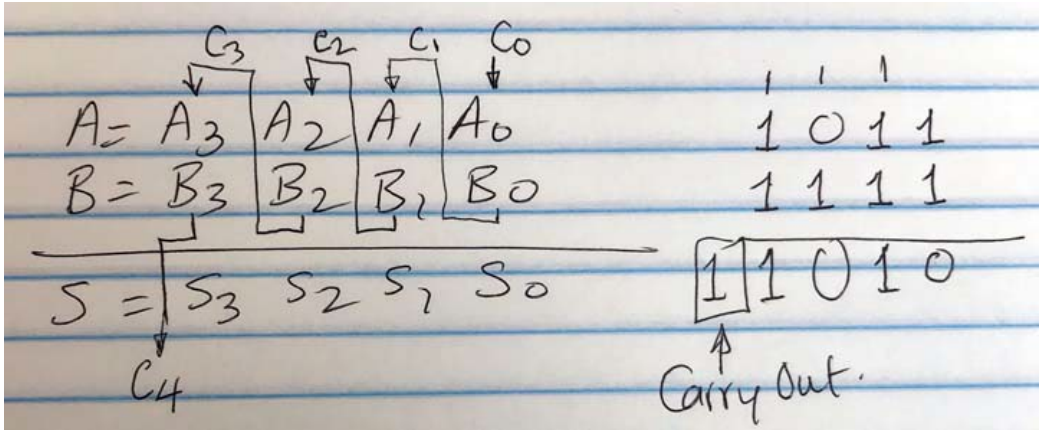
In this chapter, we build the hardware to execute microoperations

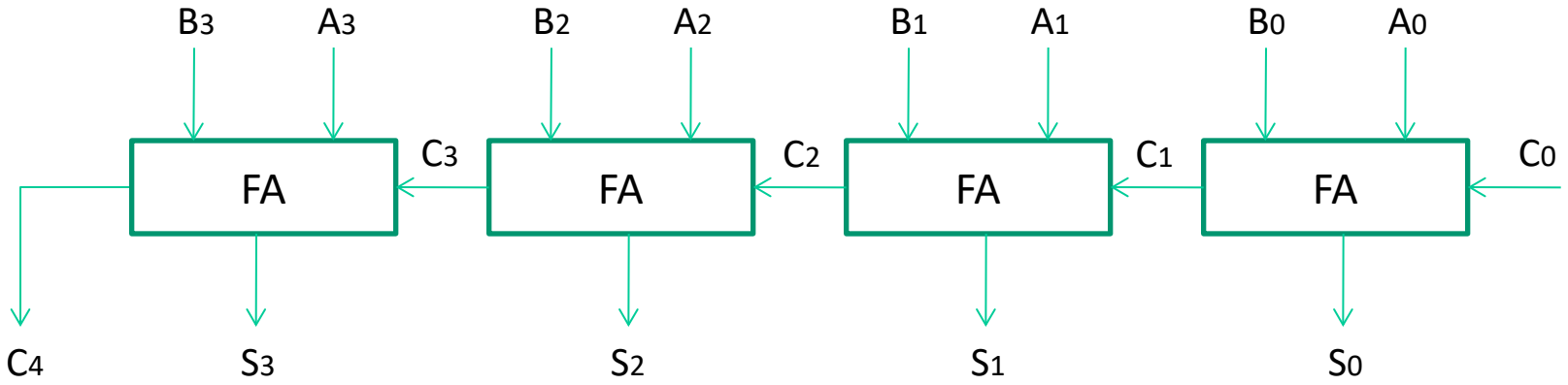
Arithmetic microoperations

Symbolic Designation	Description
R3←R1+R2	Contents of R1 plus R2 transferred to R3
R3←R1-R2	Contents of R1 minus R2 transferred to R3
R2←$\overline{R2}$	Complement the contents of R2 (1's complement)
R2←$\overline{R2}$+1	2's complement the contents of R2 (negate)
R3←R1+$\overline{R2}$+1	R1 plus the 2's complement of R2 (subtraction)
R1←R1+1	Increment the contents of R1 by one
R1←R1-1	Decrement the contents of R1 by one

4-bit adder circuit

Addition using pencil and paper





Above implements $S \leftarrow A + B$

What about $S \leftarrow A - B$ or $S \leftarrow A + \bar{B} + 1$?

4-bit adder-subtractor

Four Categories of microoperations

1. Register transfer
2. Arithmetic
3. Logic
4. Shift

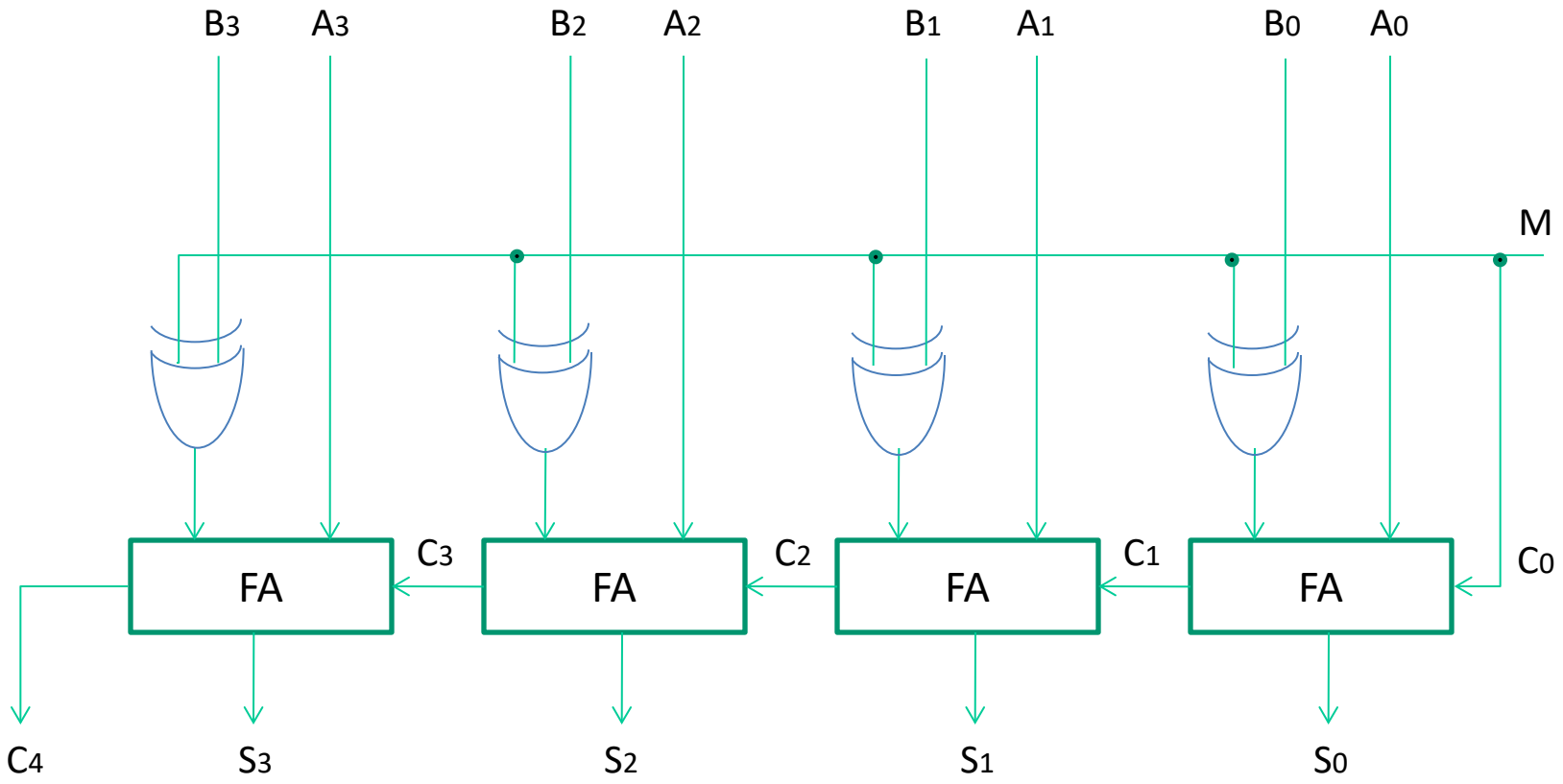
All microoperations must have underlying hardware

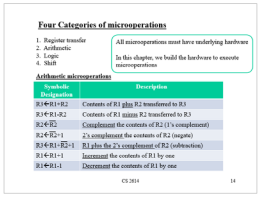
In this chapter, we build the hardware to execute microoperations

Arithmetic microoperations

Symbolic Designation	Description
R3←R1+R2	Contents of R1 plus R2 transferred to R3
R3←R1-R2	Contents of R1 minus R2 transferred to R3
R2←$\overline{R2}$	Complement the contents of R2 (1's complement)
R2←$\overline{R2}$+1	2's complement the contents of R2 (negate)
R3←R1+$\overline{R2}$+1	R1 plus the 2's complement of R2 (subtraction)
R1←R1+1	Increment the contents of R1 by one
R1←R1-1	Decrement the contents of R1 by one

CS 2614 14





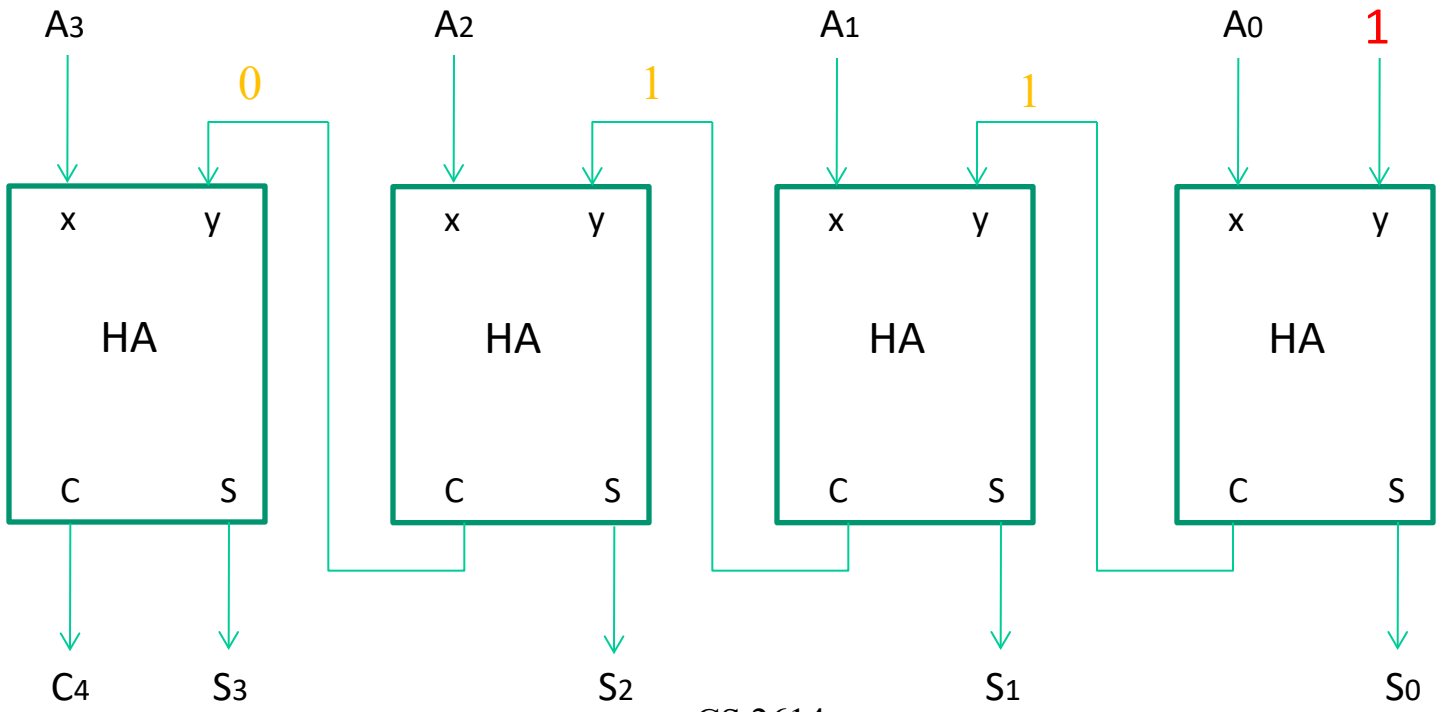
Binary Incrementer

A counter can be used as an incrementor.

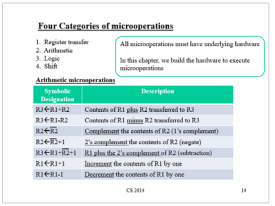
If a combinational circuit has to be used as incrementer, a number of half adders can be connected in sequence.

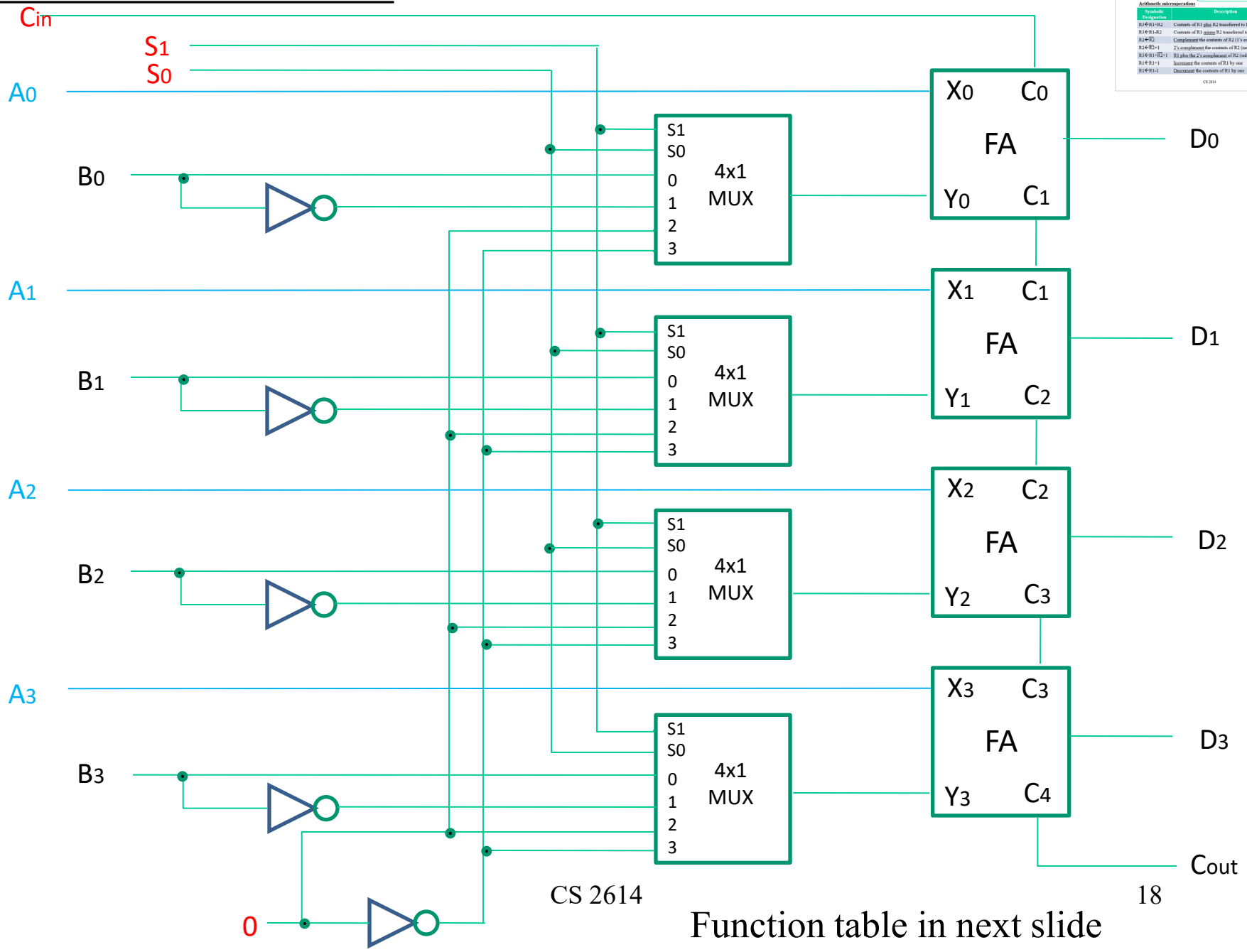
Example

```
0  1  1        <= Carry (y)
1  0  1  1     <=     A  (x)
+           1  <=        (y)
-----------------------------
1  1  0  0     <= A+1
```



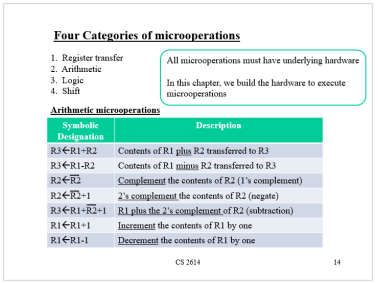
4-bit arithmetic circuit





Function table in next slide

Function table of arithmetic circuit



Four Categories of microoperations

1. Register transfer
2. Arithmetic
3. Logic
4. Shift

All microoperations must have underlying hardware

In this chapter, we build the hardware to execute microoperations

Arithmetic microoperations

Symbolic Designation	Description
R3←R1+R2	Contents of R1 plus R2 transferred to R3
R3←R1-R2	Contents of R1 minus R2 transferred to R3
R2←$\overline{R2}$	Complement the contents of R2 (1's complement)
R2←$\overline{R2}$+1	2's complement the contents of R2 (negate)
R3←R1+$\overline{R2}$+1	R1 plus the 2's complement of R2 (subtraction)
R1←R1+1	Increment the contents of R1 by one
R1←R1-1	Decrement the contents of R1 by one

CS 2614 14

Select			Input	Output	Microoperation
S1	S0	Cin	Y	D=A+Y+Cin	
0	0	0	B	D=A+B	Add
0	0	1	B	D=A+B+1	Add with carry
0	1	0	$\overline{B}$	D=A+$\overline{B}$	Subtract with borrow
0	1	1	$\overline{B}$	D=A+$\overline{B}$+1	subtract
1	0	0	0	D=A	Transfer A
1	0	1	0	D=A+1	Increment A
1	1	0	1	D=A-1	Decrement A
1	1	1	1	D=A	Transfer A

Logic Microops

Bit-wise logic operations on the contents of two registers.

Notation:

$\wedge$ - AND
$\vee$ - OR
$\oplus$ - XOR

Example

```
1 0 1 0 1 0 1 1     R1
1 0 1 0 0 0 1 1     R2
-------------------------------
1 0 1 0 1 0 1 1     R1 ∨ R2
```

P+Q: R1←R2∨R3

Denotes OR of two control signals

denotes OR microop

P+Q: R1←R2+R3

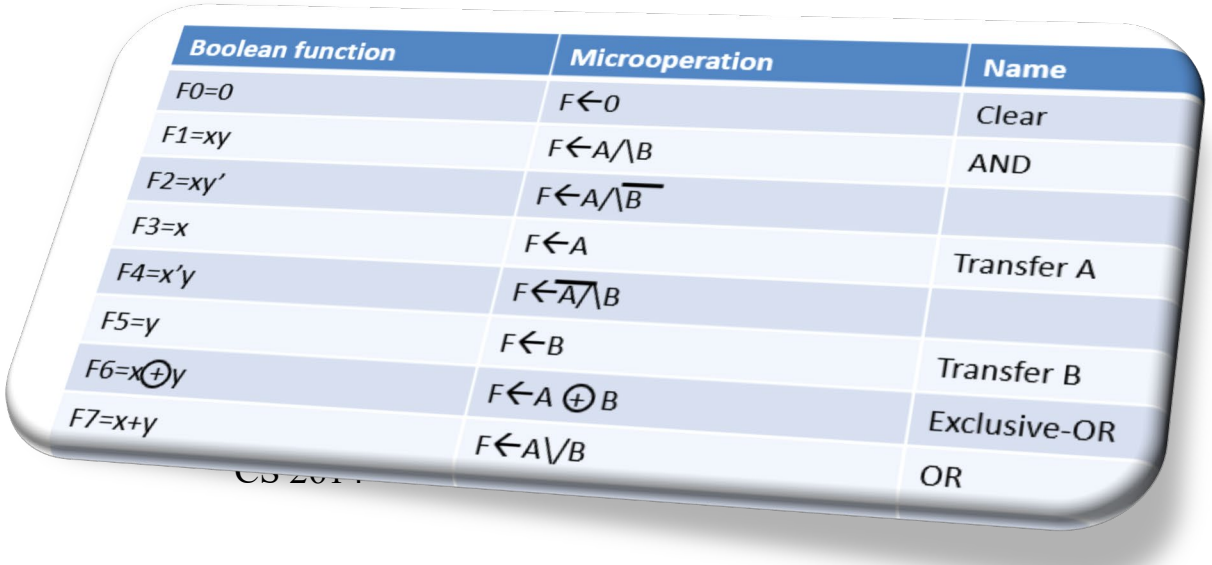
denotes arithmetic plus (addition)

Sixteen Logic Microoperations

Boolean function	Microoperation	Name
F0=0	F←0	Clear
F1=xy	F←A/\B	AND
F2=xy’	F←A/\$\overline{B}$	
F3=x	F←A	Transfer A
F4=x’y	F←$\overline{A}$/\B	
F5=y	F←B	Transfer B
F6=x⊕y	F←A ⊕ B	Exclusive-OR
F7=x+y	F←A\/B	OR
F8=(x+y)’	F←$\overline{A \backslash/ B}$	NOR
F9=(x⊕y)’	F←$\overline{A \oplus B}$	Exclusive NOR
F10=y’	F←$\overline{B}$	Complement B
F11=x+y’	F←A\/$\overline{B}$	
F12=x’	F←$\overline{A}$	Complement of A
F13=x’+y	F←$\overline{A}$\/B	
F14=(xy)’	F←$\overline{A/\backslash B}$	NAND
F15=1	F←all 1’s	Set to all 1’s

Truth Tables for 16 functions of Two Variables

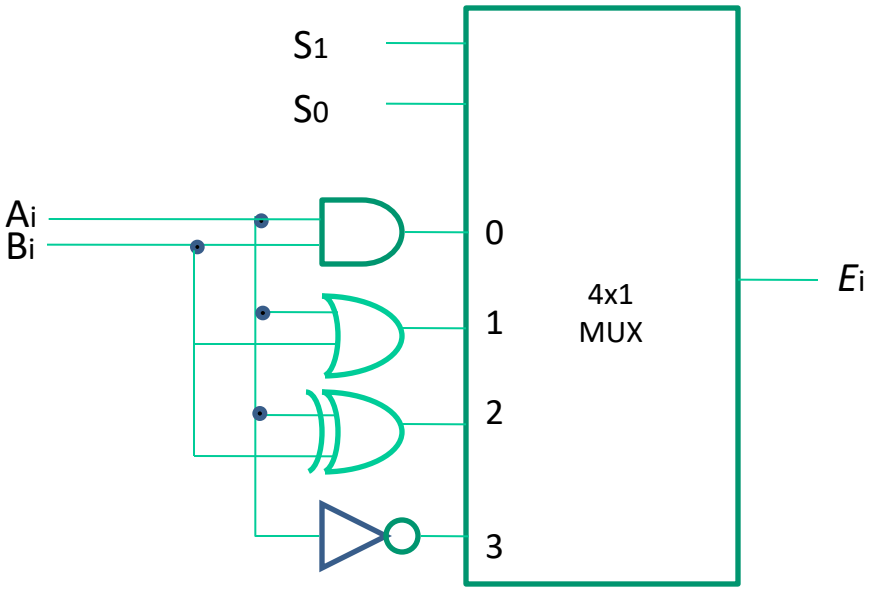
Inputs		Function Outputs															
X	Y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1



Most computers implement only four functions: AND, OR, COMPLEMENT and XOR. One stage is shown below.

S_1	S_0	Output	Operation
0	0	E=A/\B	AND
0	1	E=A\/B	OR
1	0	E=A ⊕ B	XOR
1	1	$E=\overline{A}$	Complement

(a) Function table



(b) Logic diagram

Applications of Logic Operations

Selective Set

1 0 1 0 A before

1 1 0 0 B

---------- OR (selectively *turn on* bits)

1 1 1 0

Selective Complement

1 0 1 0 A before

1 1 0 0 B

---------- XOR (selectively invert bits)

0 1 1 0

Selective Clear

1 0 1 0 A before

0 0 1 1 B

---------- AND (selectively *turn off* bits)

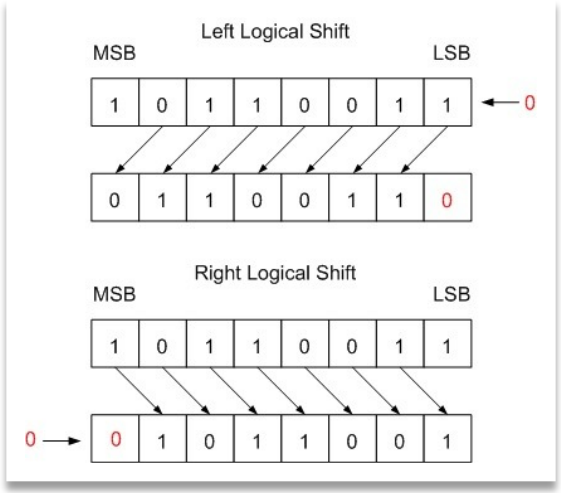
0 0 1 0

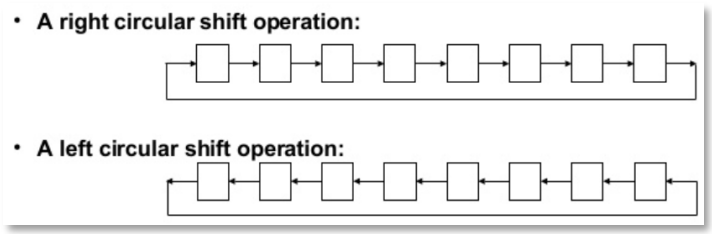
Shift Microoperations

logical shift (shl/shr)
circular shift (cil/cir)
arithmetic shift (ashl/ashr)

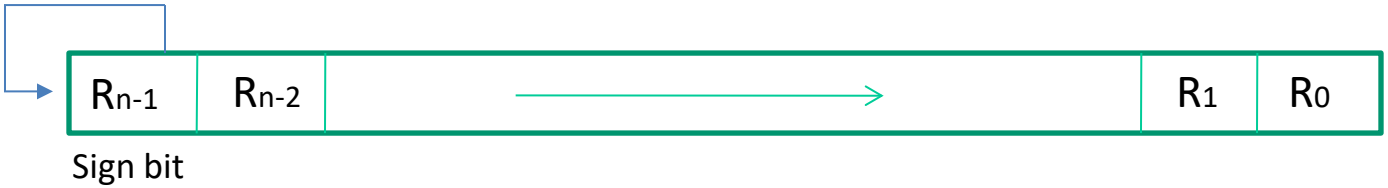
ashl multiplies number by 2
ashr divides number by 2

Symbolic designation	Description
R←shl R	Shift-left register R
R←shr R	Shift-right register R
R←cil R	Circular shift-left register R
R←cir R	Circular shift-right register R
R←ashl R	Arithmetic shift-left R
R←ashr R	Arithmetic shift-right R

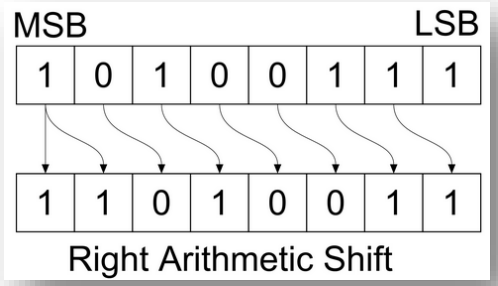




Arithmetic shift-right



- Sign bit in R_{n-1} remains unchanged
- R_{n-1} is copied into R_{n-2}

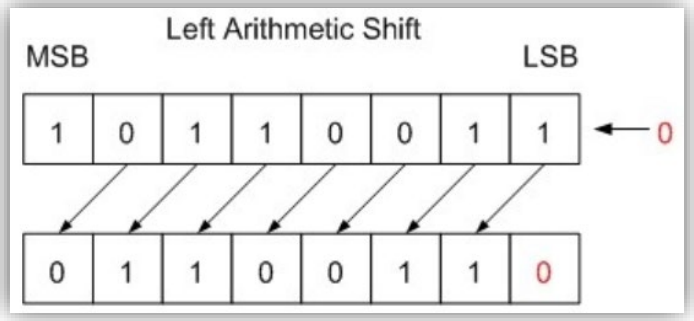


EXAMPLE

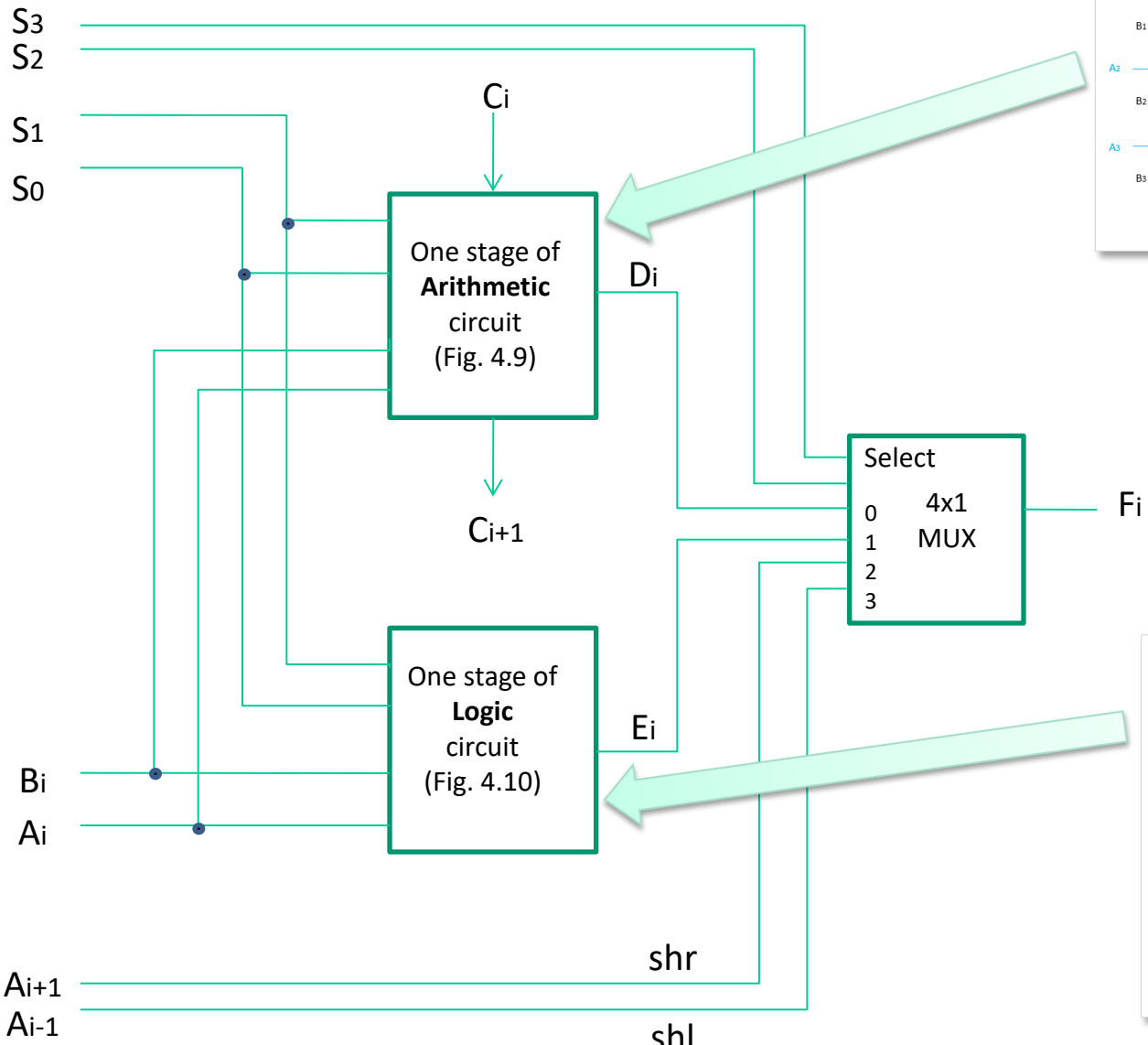
1110 becomes	1111
(-2)	(-1)
1010 becomes	1101
(-6)	(-3)

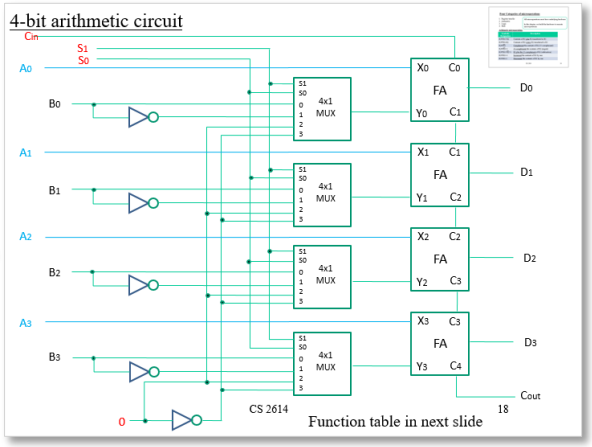
Arithmetic shift-left

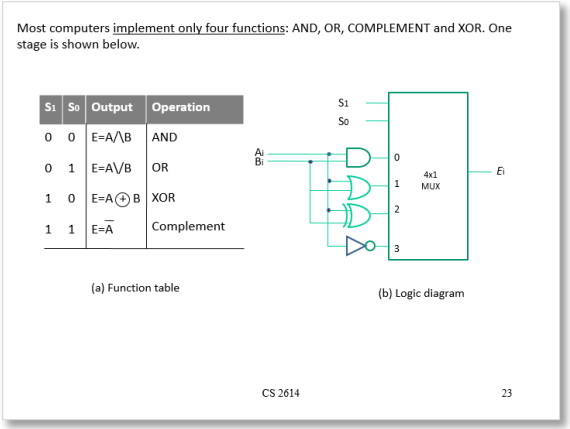
- Insert 0 into R_0 and shift all bits to the left
- Overflow occurs if R_{n-1} is not equal to R_{n-2}



Arithmetic Logic Shift Unit (One stage shown)







Function Table for Arithmetic Logic Shift Unit

Operation select					Operation	Function
S_3	S_2	S_1	S_0	C_{in}		
0	0	0	0	0	F=A+B	Add
0	0	0	0	1	F=A+B+1	Add with carry
0	0	0	1	0	F=A+B’	Subtract with borrow
0	0	0	1	1	F=A+B’+1	Subtract
0	0	1	0	0	F=A	Transfer A
0	0	1	0	1	F=A+1	Increment A
0	0	1	1	0	F=A-1	Decrement A
0	0	1	1	1	F=A	Transfer A
0	1	0	0	X	F=A/\ B	AND
0	1	0	1	X	F=A\/B	OR
0	1	1	0	X	F=A⊕B	XOR
0	1	1	1	X	F=A’	Complement A
1	0	X	X	X	F= shr A	Shift right A into F
1	1	x	x	x	F= shl A	Shift left A into F