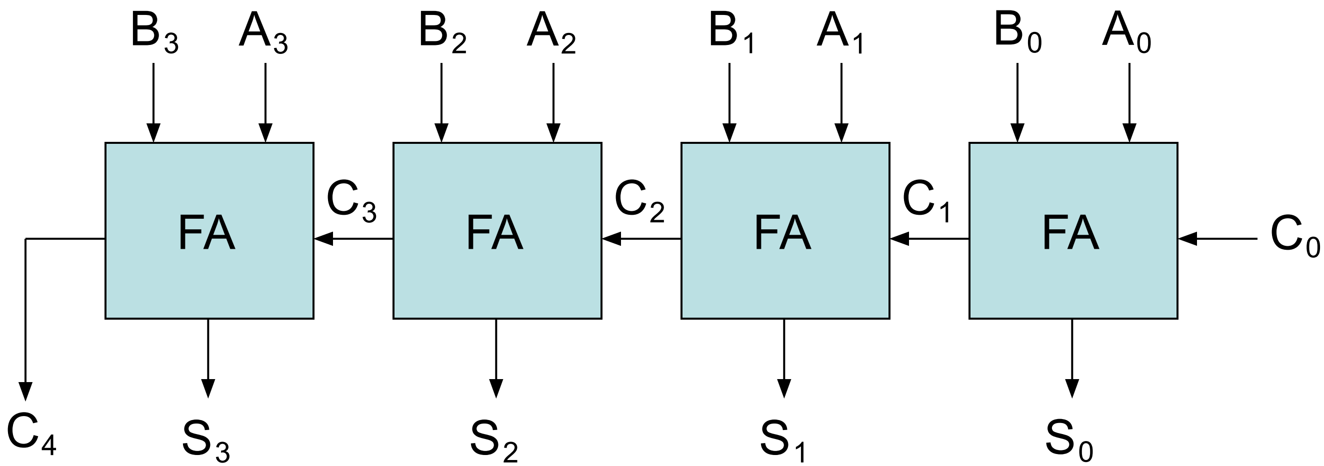
EE204: Computer Architecture

Review: Design of ALU

- Today we are going to review the basic design of ALU
- The microoperations most often encountered in digital computers are classified into four categories:
  - Register transfer microoperations
  - Arithmetic/Logic microoperations
  - Logic microoperations
  - Shift microoperations

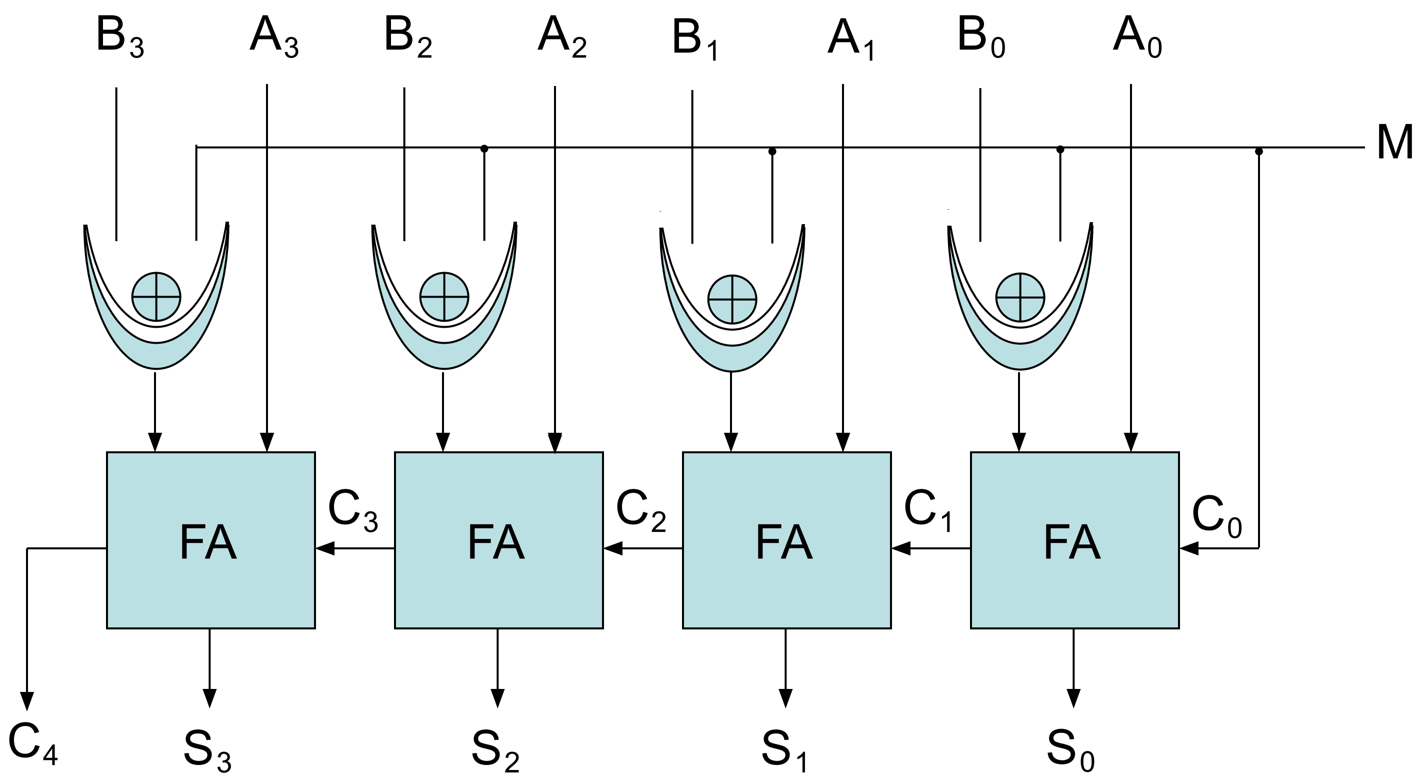
4-4 Arithmetic Microoperations
Binary Adder



4-bit binary adder
(connection of FAs)

4-4 Arithmetic Microoperations

Binary Adder-Subtractor



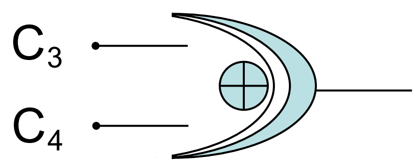
4-bit adder-subtractor

4-4 Arithmetic Microoperations
Binary Adder-Subtractor

- For unsigned numbers, this gives A – B if A≥B or the 2's complement of (B – A) if A < B

  (example: 3 – 5 = -2= 1110)
- For signed numbers, the result is A – B provided that there is no overflow. (example : -3 – 5= -8) 1101

$$
\begin{array}{r}
1011 \; + \\
\hline
1000
\end{array}
$$



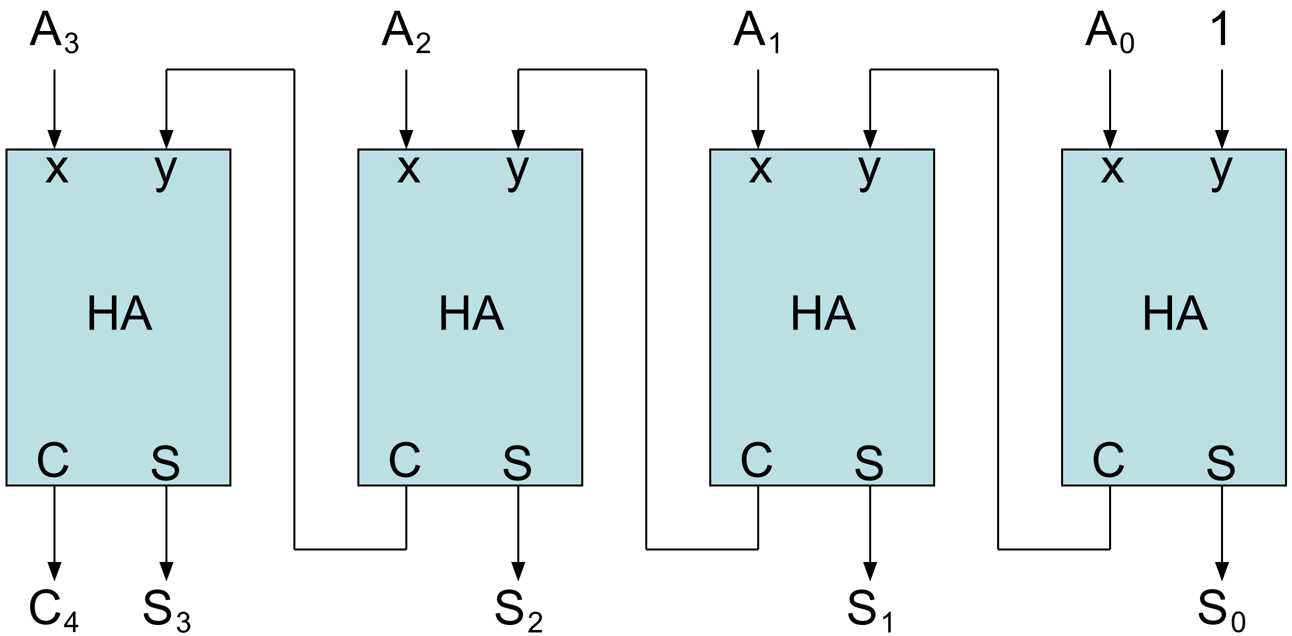
$$= V \begin{cases} 1, \text{ if overflow} \\ 0, \text{ if no overflow} \end{cases}$$

Overflow detector for signed numbers

4-4 Arithmetic Microoperations
Binary Adder-Subtractor cont.

- What is the range of unsigned numbers that can be represented in 4 bits?
- What is the range of signed numbers that can be represented in 4 bits?
- Repeat for n-bit?!

4-4 Arithmetic Microoperations
Binary Incrementer



4-bit Binary Incrementer

4-4 Arithmetic Microoperations
Binary Incrementer

- Binary Incrementer can also be implemented using a counter
- A binary decrementer can be implemented by adding 1111 to the desired register each time!

4-4 Arithmetic Microoperations

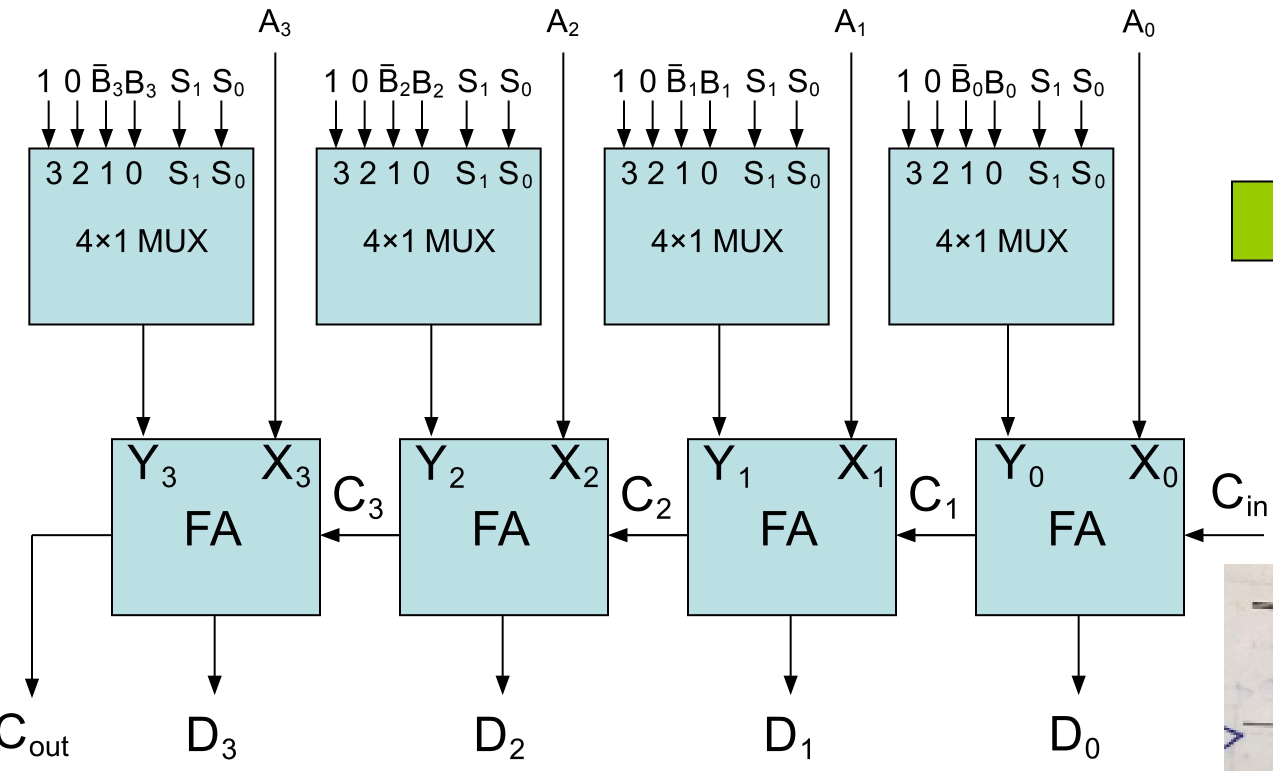
Arithmetic Circuit

- This circuit performs seven distinct arithmetic operations and the basic component of it is the parallel adder

Symbolic designation
$R3 \leftarrow R1 + R2$
$R3 \leftarrow R1 - R2$
$R2 \leftarrow \overline{R2}$
$R2 \leftarrow \overline{R2} + 1$
$R3 \leftarrow R1 + \overline{R2} + 1$
$R1 \leftarrow R1 + 1$
$R1 \leftarrow R1 - 1$

4-4 Arithmetic Microoperations

Arithmetic Circuit cont.



4-bit Arithmetic Circuit

Figure A

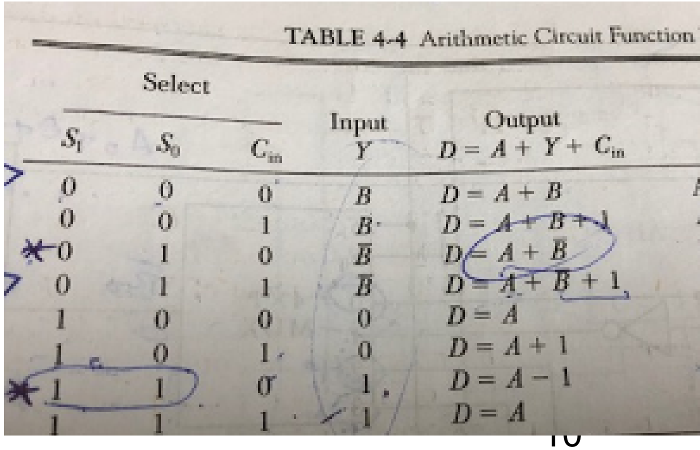


TABLE 4-4 Arithmetic Circuit Function

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

4-5 Logic Microoperations Hardware Implementation

- The hardware implementation of logic microoperations requires that logic gates be inserted for each bit or pair of bits in the registers to perform the required logic function
- Most computers use only four (AND, OR, XOR, and NOT) from which all others can be derived.

4-5 Logic Microoperations
Hardware Implementation $^{\text{cont.}}$

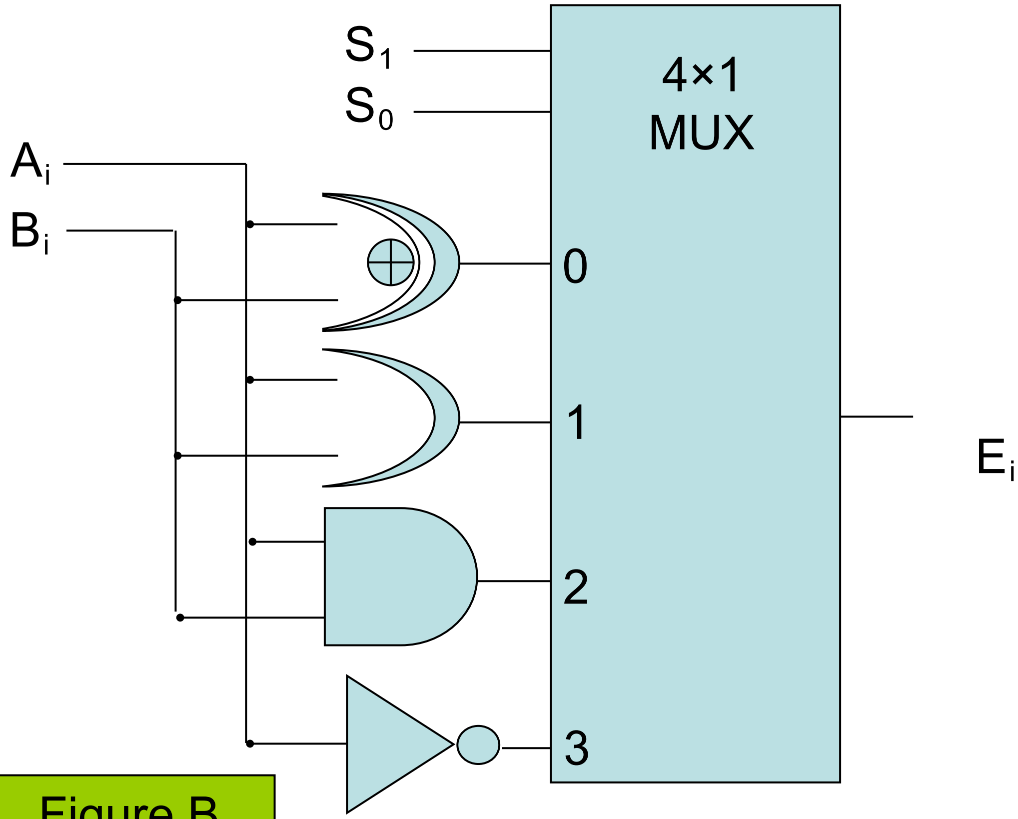


Figure B

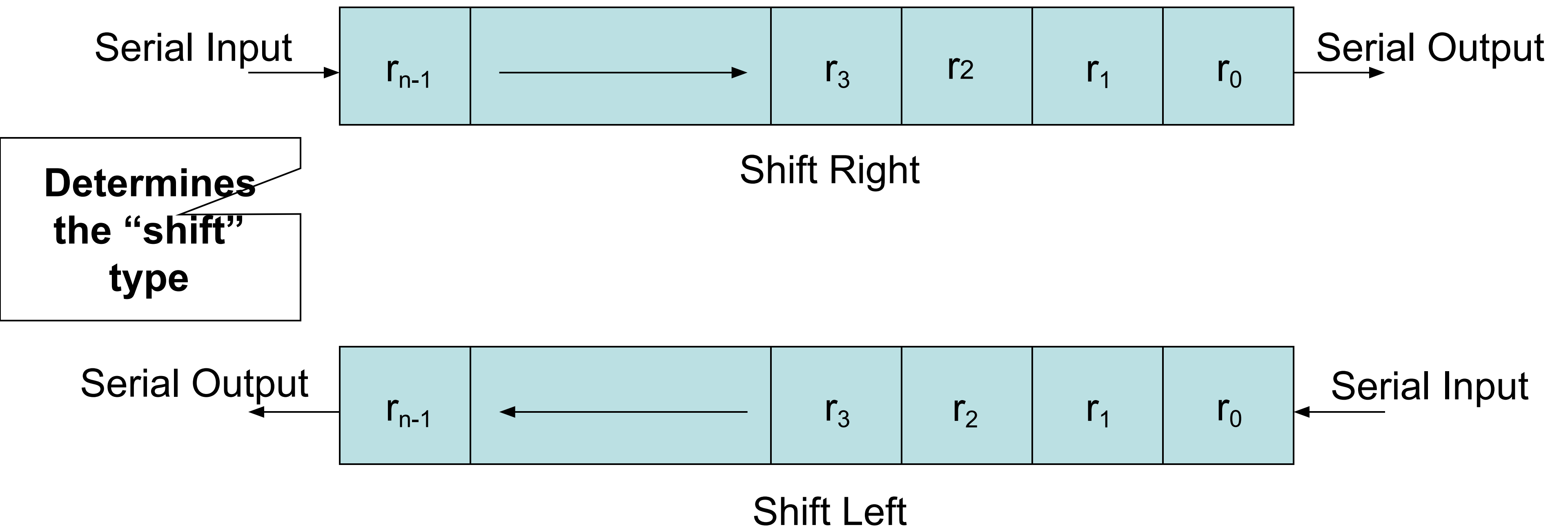
S_1	S_0	Output	Operation
0	0	$E = A \oplus B$	XOR
0	1	$E = A \vee B$	OR
1	0	$E = A \wedge B$	AND
1	1	$E = \overline{A}$	Complement

This is for one bit i

4-6 Shift Microoperations

- Used for serial transfer of data
- Also used in conjunction with arithmetic, logic, and other data-processing operations
- The contents of the register can be shifted to the left or to the right
- As being shifted, the first flip-flop receives its binary information from the serial input
- Three types of shift: Logical, Circular, and Arithmetic

4-6 Shift Microoperations cont.



**Note that the bit ri is the bit at position (i) of the register

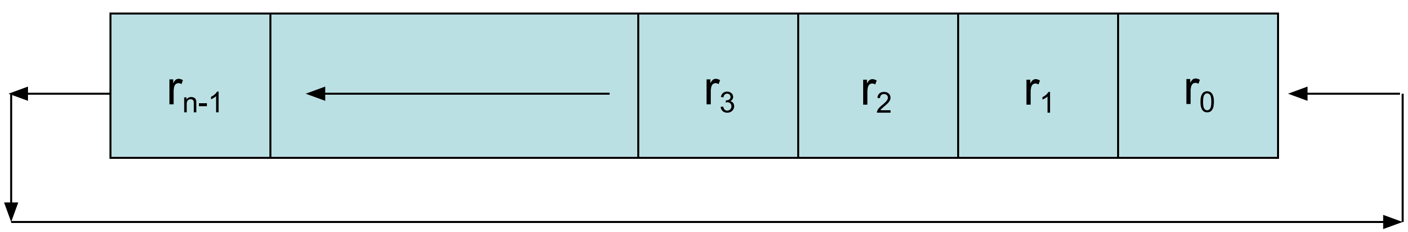
4-6 Shift Microoperations:
Circular Shifts (Rotate Operation)

- Circulates the bits of the register around the two ends without loss of information
- Circular Shift Right: $R1 \leftarrow cir\ R1$

The same

- Circular Shift Left: $R2 \leftarrow cil\ R2$

The same



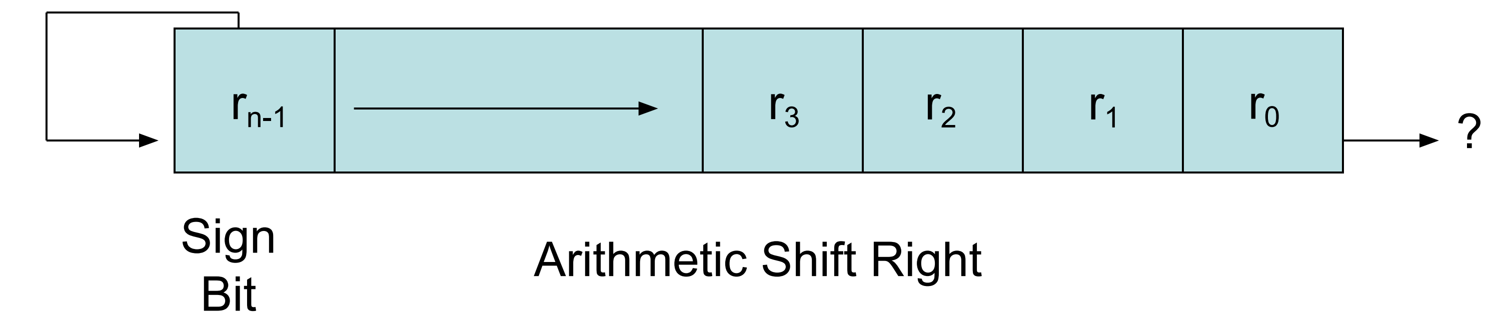
Circular Shift Left

4-6 Shift Microoperations
Arithmetic Shifts

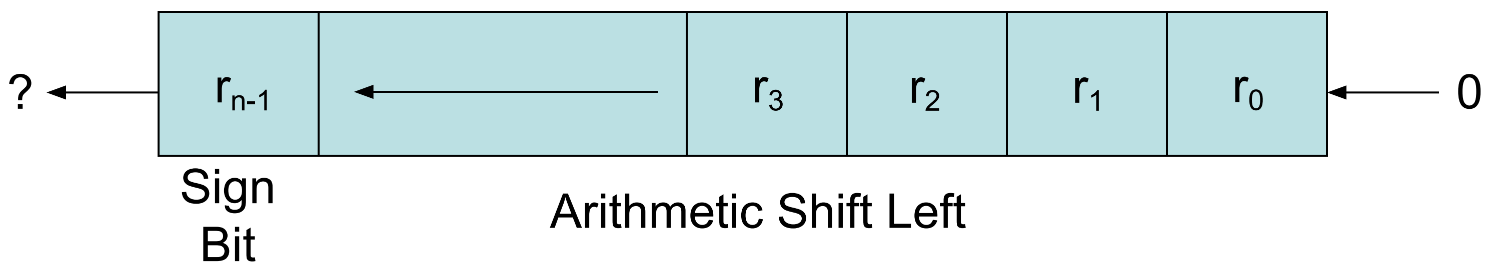
- Shifts a signed binary number to the left or right
- An arithmetic shift-left multiplies a signed binary number by 2: ashl (00100): 01000
- An arithmetic shift-right divides the number by 2

  ashr (00100) : 00010
- An overflow may occur in arithmetic shift-left, and occurs when the sign bit is changed (sign reversal)

4-6 Shift Microoperations
Arithmetic Shifts cont.



Arithmetic Shift Right

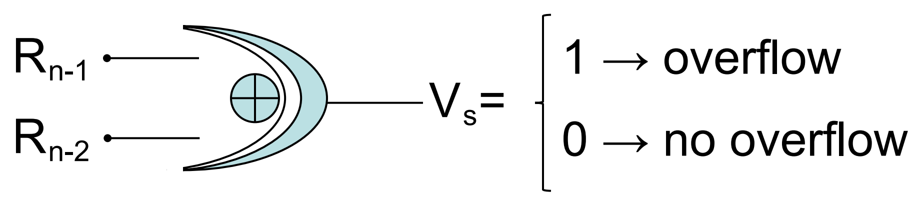


Arithmetic Shift Left

4-6 Shift Microoperations
Arithmetic Shifts cont.

- An overflow flip-flop V_s can be used to detect an arithmetic shift-left overflow

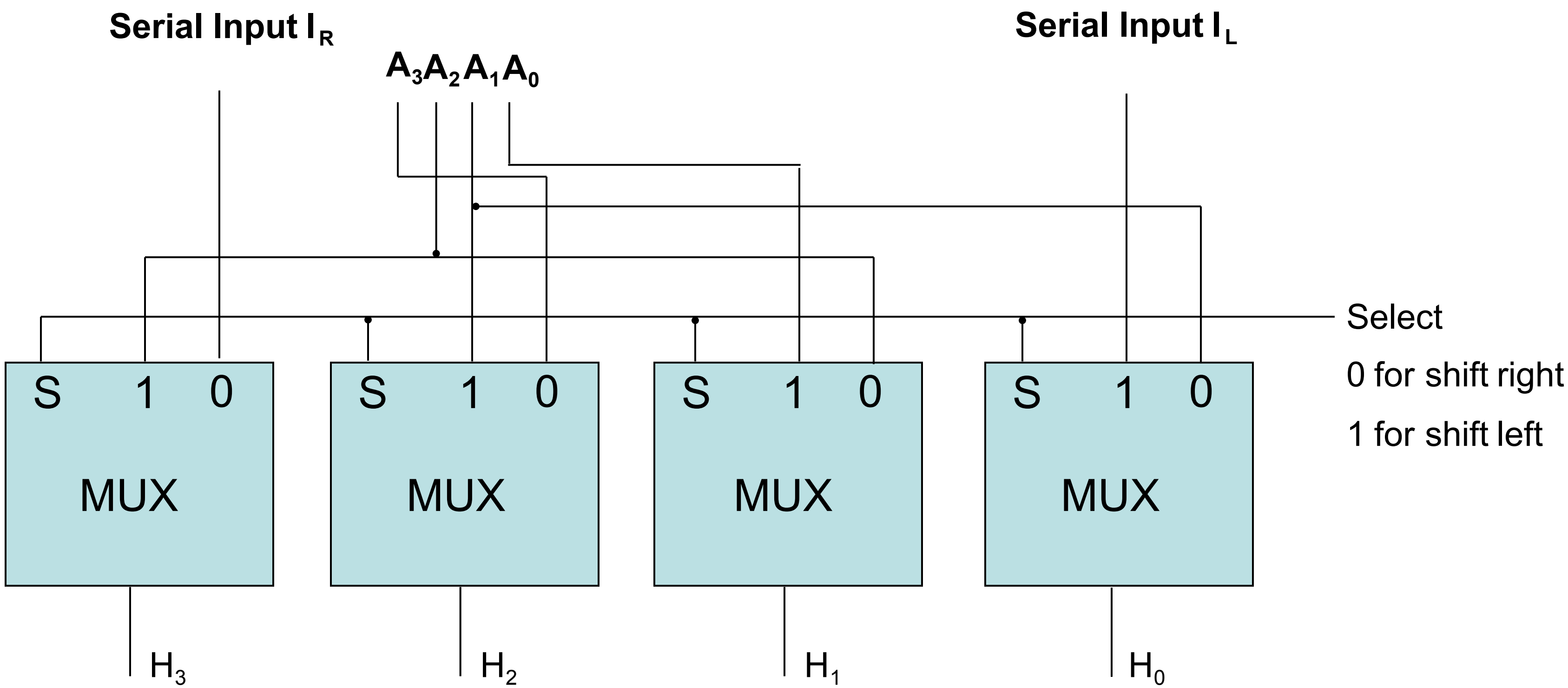
$$V_s = R_{n-1} \oplus R_{n-2}$$



4-6 Shift Microoperations cont.

- Example: Assume R1=11001110, then:
  - Arithmetic shift right once : R1 = 11100111
  - Arithmetic shift right twice : R1 = 11110011
  - Arithmetic shift left once : R1 = 10011100
  - Arithmetic shift left twice : R1 = 00111000
  - Logical shift right once : R1 = 01100111
  - Logical shift left once : R1 = 10011100
  - Circular shift right once : R1 = 01100111
  - Circular shift left once : R1 = 10011101

4-6 Shift Microoperations Hardware Implementation cont.



4-bit Combinational Circuit Shifter

Putting things Together
4-7 Arithmetic Logic Shift Unit

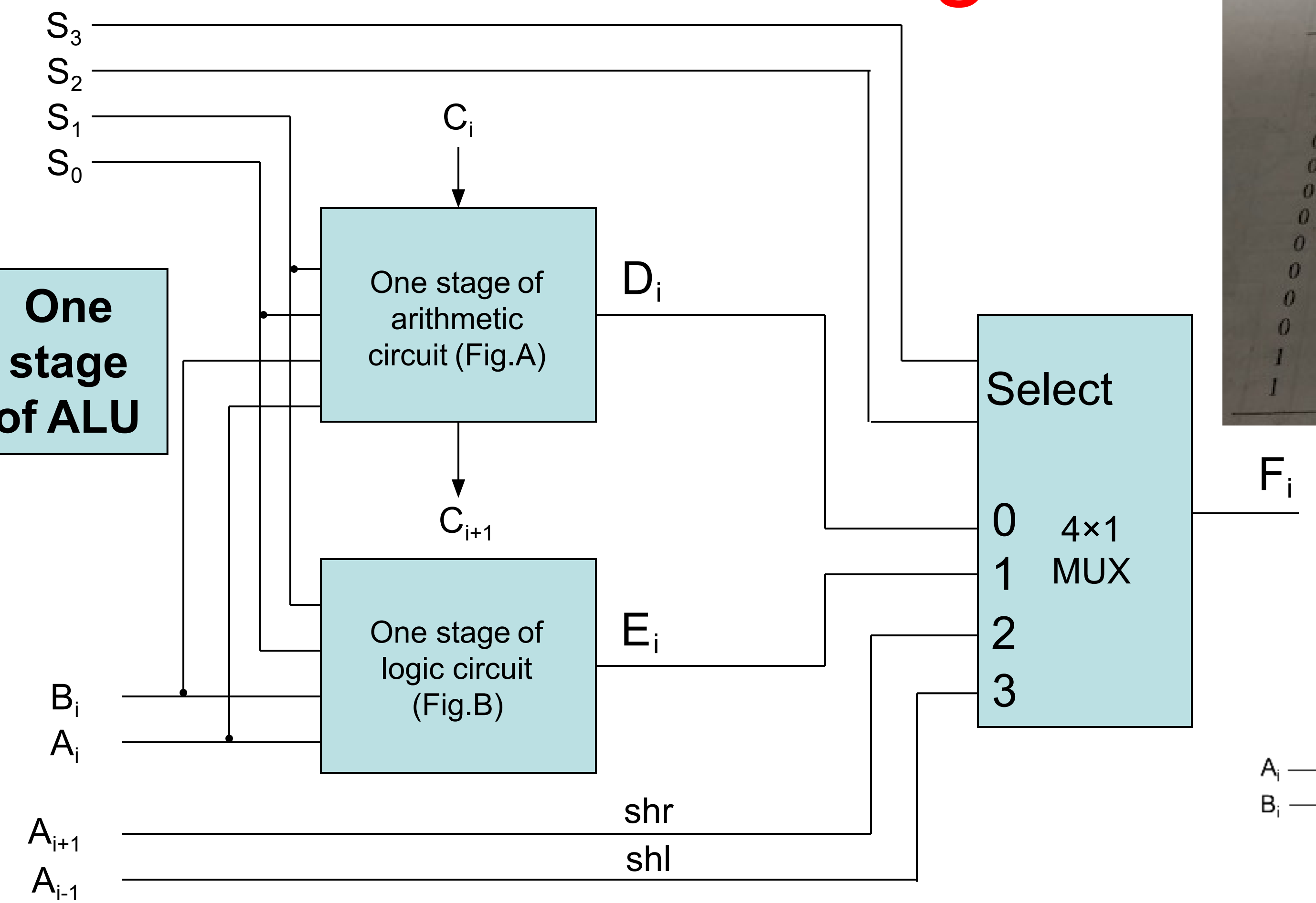
- Instead of having individual registers performing the microoperations directly, computer systems employ a number of storage registers connected to a common operational unit called an Arithmetic Logic Unit (**ALU**)

Supported Operations

TABLE 4-8 Function

Operation select					Operation	Function
S_3	S_2	S_1	S_0	C_{in}		
0	0	0	0	0	$F = A$	Transfer A
0	0	0	0	1	$F = A + 1$	Increment A
0	0	0	1	0	$F = A + B$	Addition
0	0	0	1	1	$F = A + B + 1$	Add with carry
0	0	1	0	0	$F = A + \overline{B}$	Subtract with borrow
0	0	1	0	1	$F = A + \overline{B} + 1$	Subtraction
0	0	1	1	0	$F = A - 1$	Decrement A
0	0	1	1	1	$F = A$	Transfer A
0	1	0	0	×	$F = A \wedge B$	AND
0	1	0	1	×	$F = A \vee B$	OR
0	1	1	0	×	$F = A \oplus B$	XOR
0	1	1	1	×	$F = \overline{A}$	Complement A
1	0	×	×	×	$F = \text{shr } A$	Shift right A into F
1	1	×	×	×	$F = \text{shl } A$	Shift left A into F

4-7 Arithmetic Logic Sh



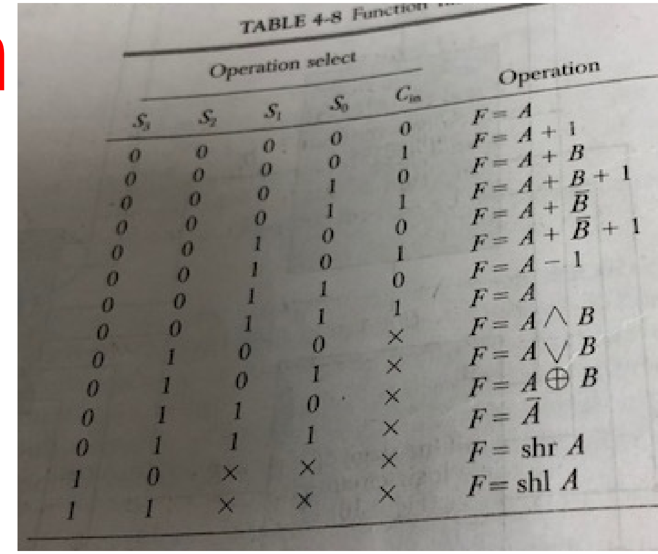
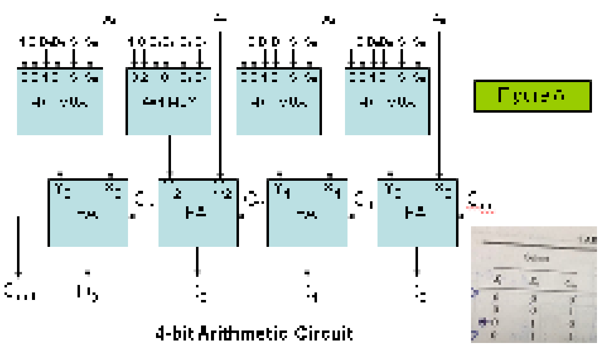


TABLE 4-8 Function ...

Operation select					Operation
S_3	S_2	S_1	S_0	C_{in}	
0	0	0	0	0	$F = A$
0	0	0	0	1	$F = A + 1$
0	0	0	1	0	$F = A + B$
0	0	0	1	1	$F = A + B + 1$
0	0	1	0	0	$F = A + \overline{B}$
0	0	1	0	1	$F = A + \overline{B} + 1$
0	0	1	1	0	$F = A - 1$
0	0	1	1	1	$F = A$
0	1	0	0	×	$F = A \wedge B$
0	1	0	1	×	$F = A \vee B$
0	1	1	0	×	$F = A \oplus B$
0	1	1	1	×	$F = \overline{A}$
1	0	×	×	×	$F = \text{shr } A$
1	1	×	×	×	$F = \text{shl } A$



4-bit Arithmetic Circuit

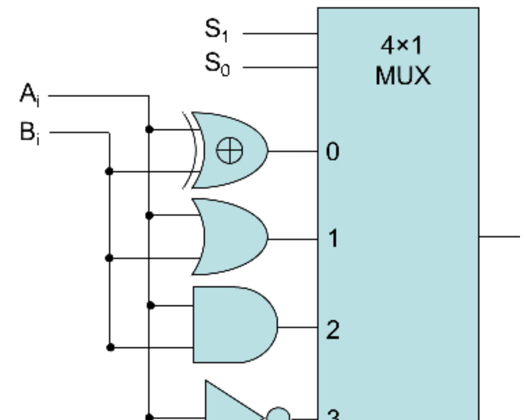


Figure B