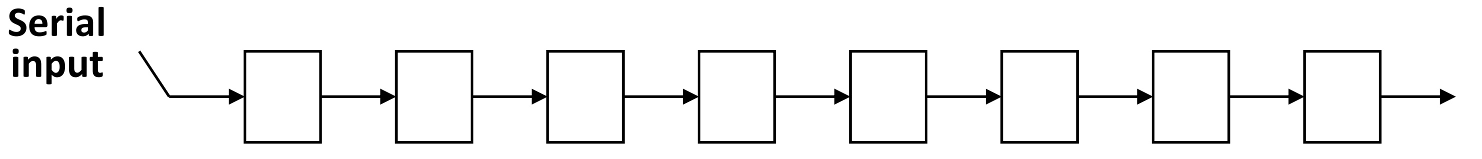
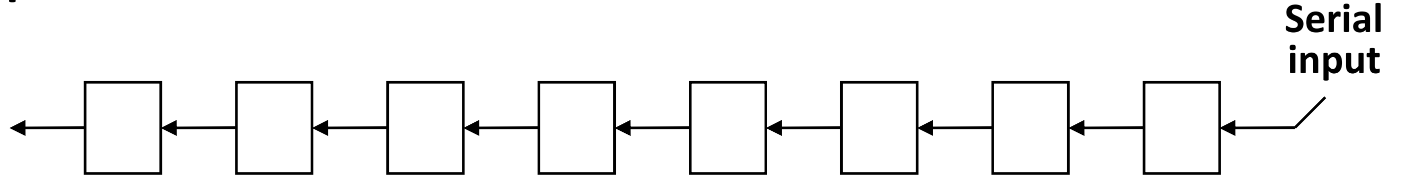
Shift Microoperations

- **There are three types of shifts**
  - ***Logical shift***
  - ***Circular shift***
  - ***Arithmetic shift***
- **What differentiates them is the information that goes into the serial input**

- **A right shift operation**



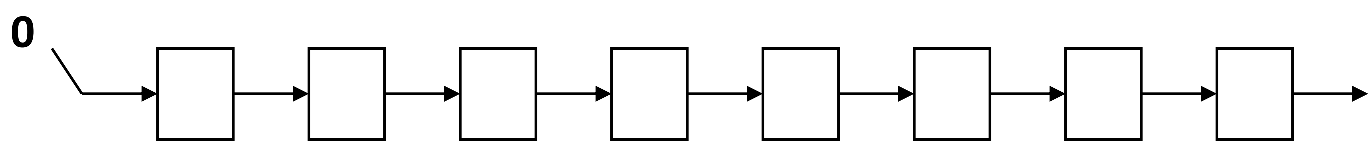
- **A left shift operation**



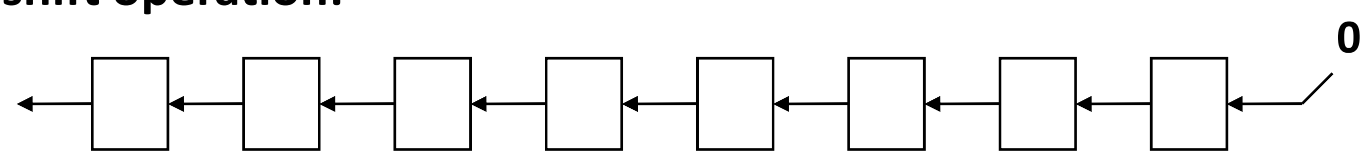
Logical Shift

- **In a logical shift the serial input to the shift is a 0.**

- **A right logical shift operation:**



- **A left logical shift operation:**

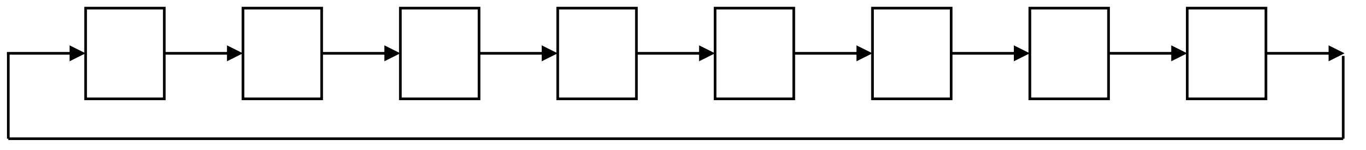


- **In a Register Transfer Language, the following notation is used**
  - ***shl*** **for a logical shift left**
  - ***shr*** **for a logical shift right**
  - **Examples:**
    - **R2 $\leftarrow$ *shr* R2**
    - **R3 $\leftarrow$ *shl* R3**

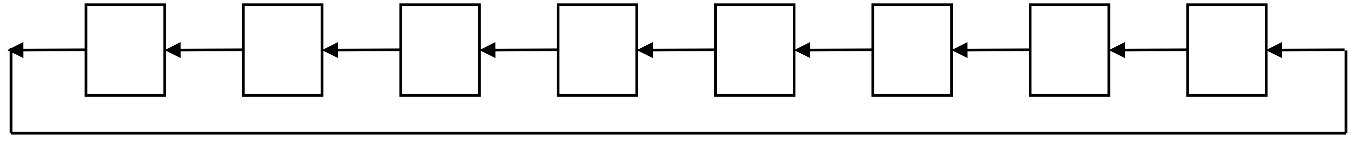
Circular Shift

- **In a circular shift the serial input is the bit that is shifted out of the other end of the register.**

- **A right circular shift operation:**



- **A left circular shift operation:**

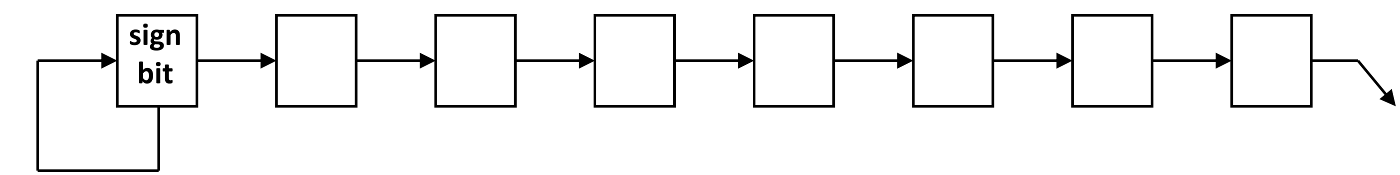


- **In a RTL, the following notation is used**
  - ***cil*** **for a circular shift left**
  - ***cir*** **for a circular shift right**
  - **Examples:**
    - **R2 ← *cir* R2**
    - **R3 ← *cil* R3**

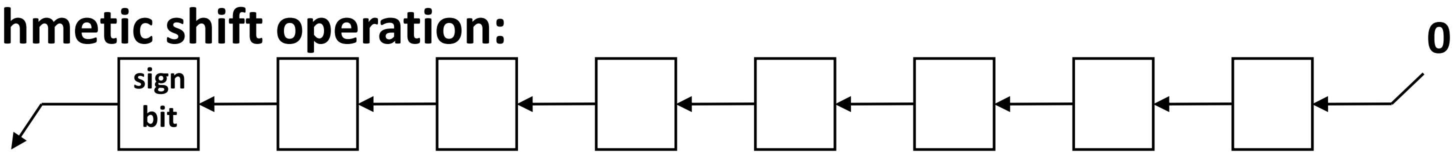
Arithmetic Shift

- **An arithmetic shift is meant for signed binary numbers (integer)**
- **An arithmetic left shift multiplies a signed number by two**
- **An arithmetic right shift divides a signed number by two**
- **Sign bit : 0 for positive and 1 for negative**
- **The main distinction of an arithmetic shift is that it must keep the sign of the number the same as it performs the multiplication or division**

- **A right arithmetic shift operation:**

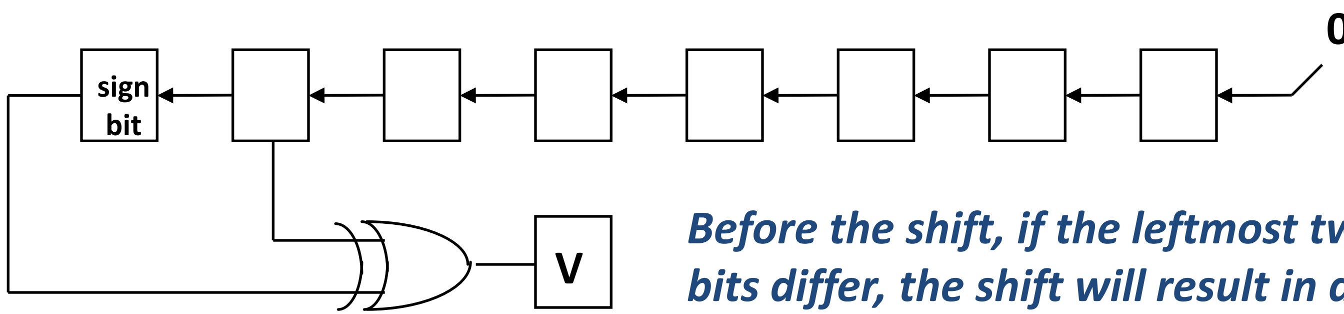


- **A left arithmetic shift operation:**



Arithmetic Shift

- **An left arithmetic shift operation must be checked for the overflow**



Before the shift, if the leftmost two bits differ, the shift will result in an overflow

- **In a RTL, the following notation is used**
  - ***ashl*** **for an arithmetic shift left**
  - ***ashr*** **for an arithmetic shift right**
  - **Examples:**
    - » **R2 ← *ashr* R2**
    - » **R3 ← *ashl* R3**

- **An arithmetic left shift multiplies a signed number by**

A) 4

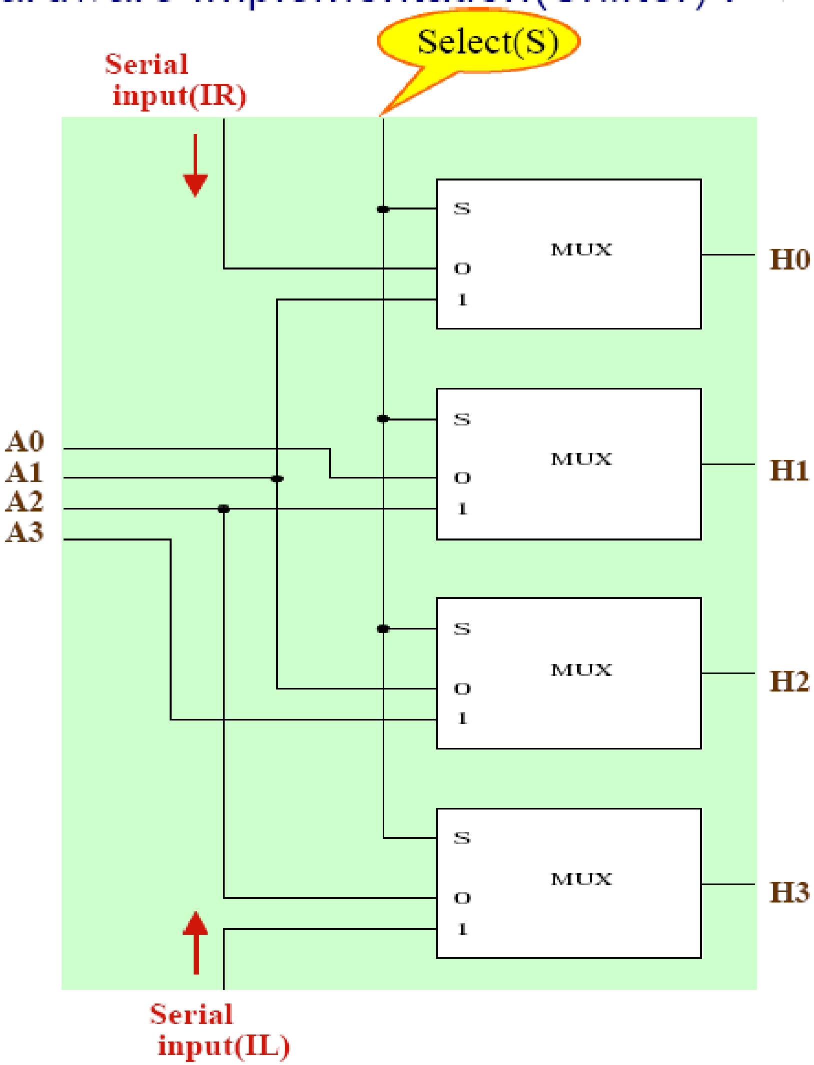
B) 8

C) 2

D) 16

Hardware Implementation of Shift Microoperation

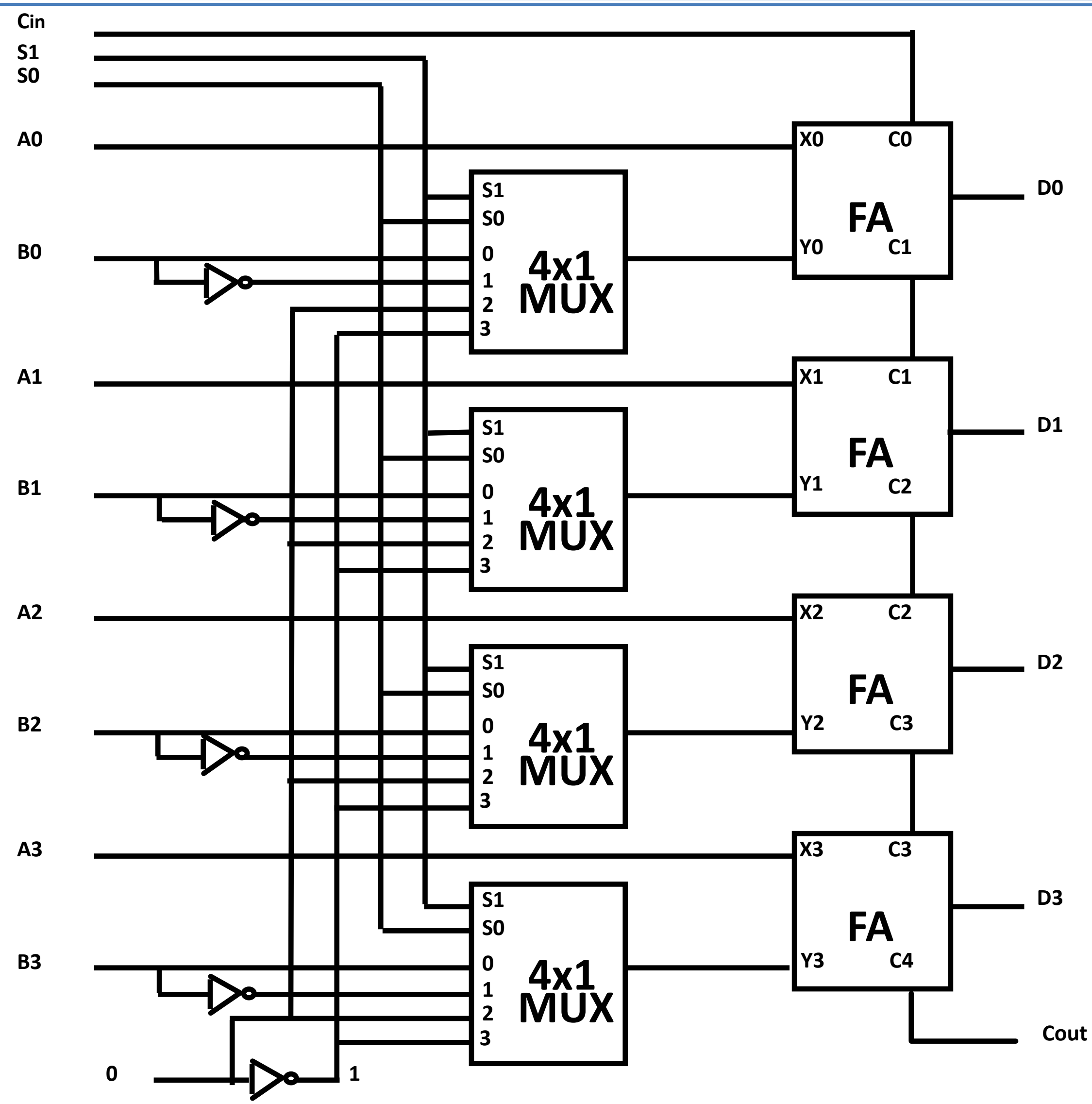
- Hardware Implementation(Shifter) :



Function Table

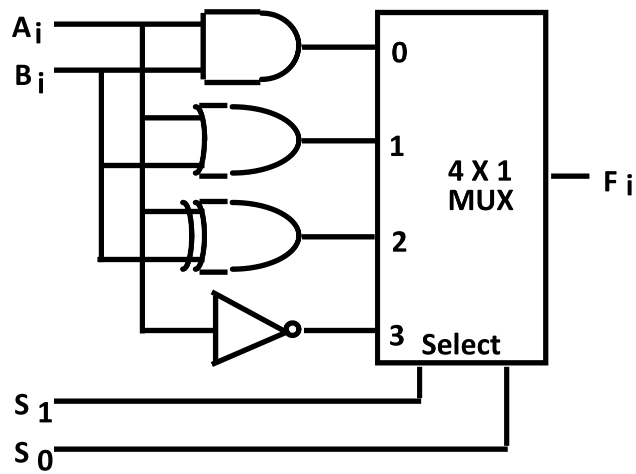
Select	output			
S	H0	H1	H2	H3
0	IR	A0	A1	A2
1	A1	A2	A3	IL

Arithmetic Circuits



Select			Input	Output
S1	S0	C_{in}	Y	$D=A+Y+C_{in}$
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

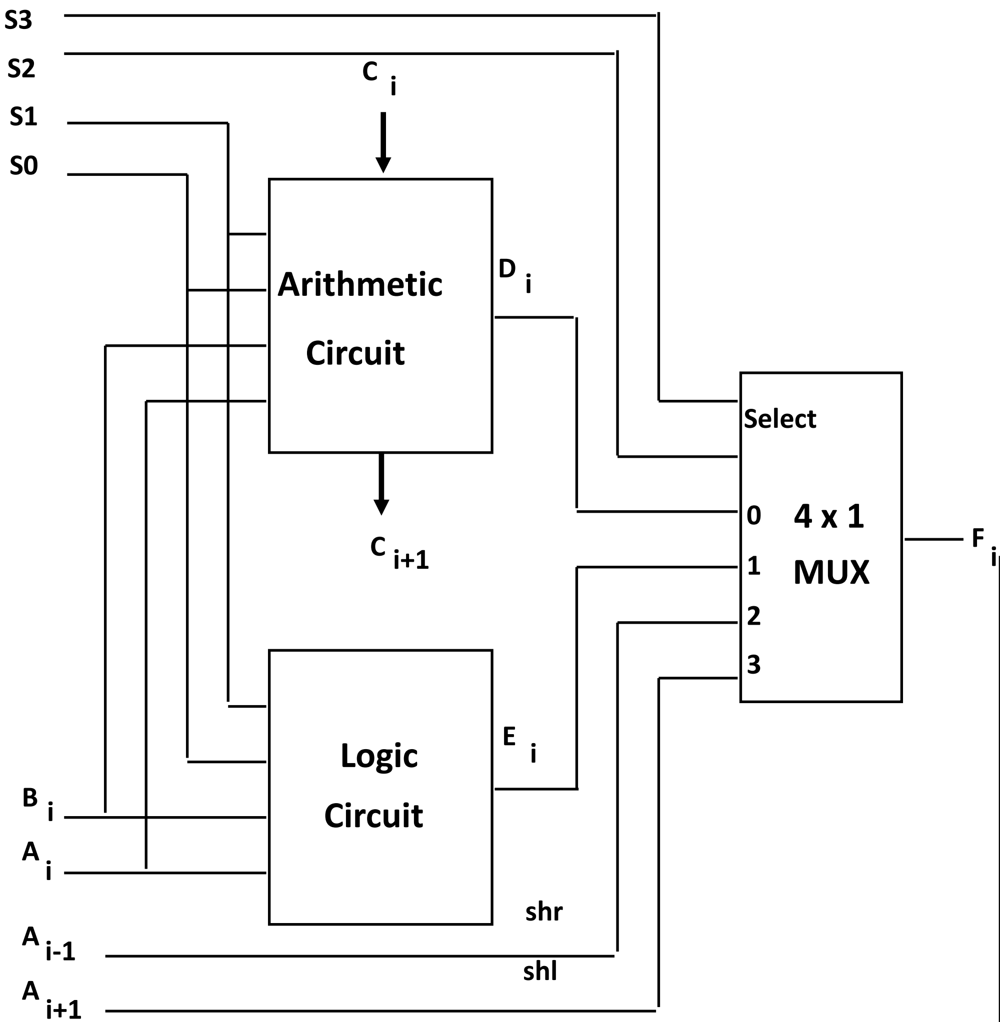
Hardware Implementation



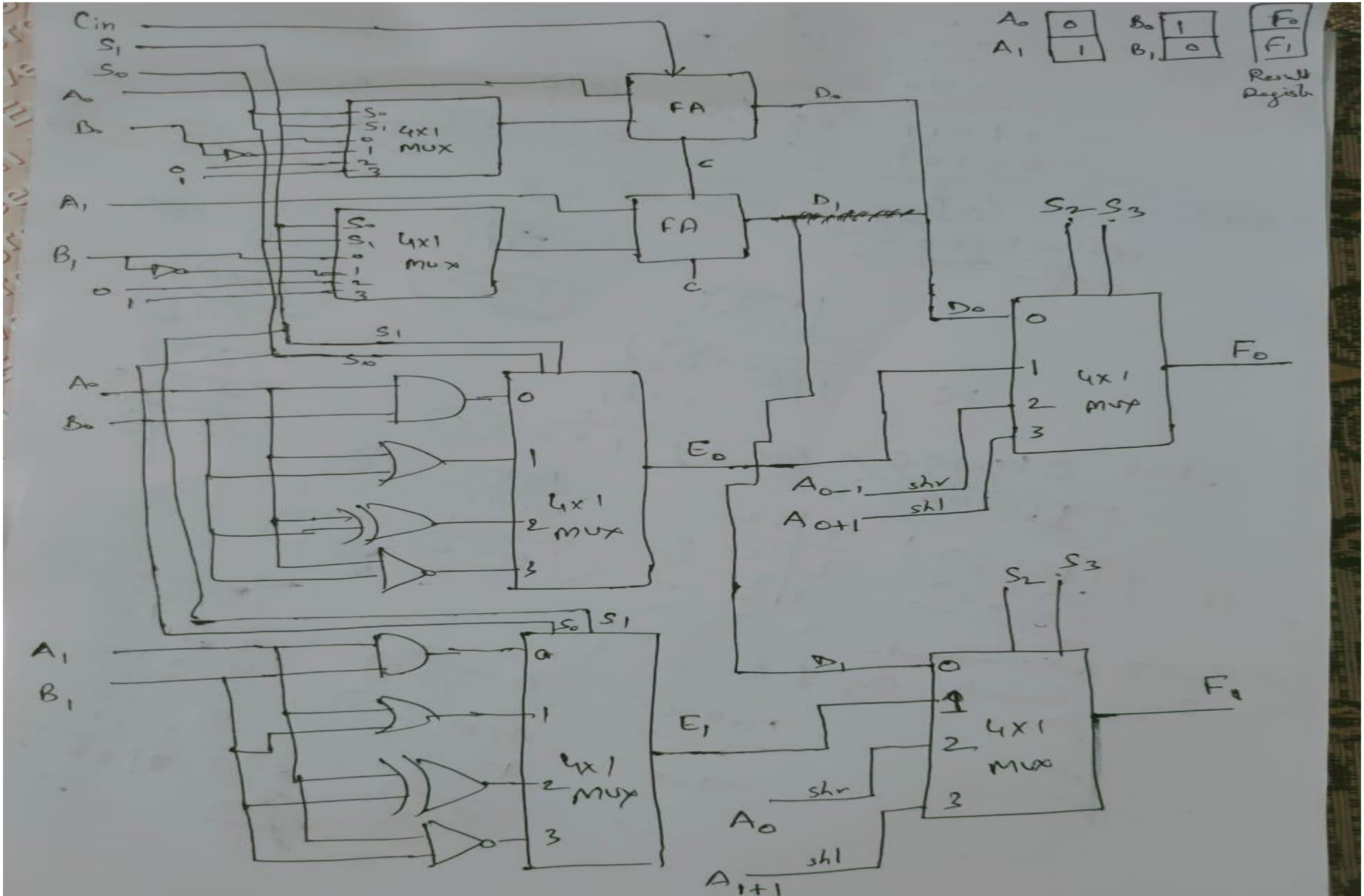
Function table

S_1 S_0	Output	μ-operation
0 0	$F = A \wedge B$	AND
0 1	$F = A \vee B$	OR
1 0	$F = A \oplus B$	XOR
1 1	$F = A'$	Complement

Arithmetic Logic and Shift Unit



S3	S2	S1	S0	Cin	Operation
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 + B’
0	0	1	0	1	F = A + B’+ 1
0	0	1	1	0	F = A - 1
0	0	1	1	1	F = A
0	1	0	0	X	F = A $\wedge$ B
0	1	0	1	X	F = A $\vee$ B
0	1	1	0	X	F = A $\oplus$ B
0	1	1	1	X	F = A’
1	0	X	X	X	F = shr A
1	1	X	X	X	F = shl A



2-Bit Arithmetic Logic Shift Unit

- In context of arithmetic shift left operation, which of the following Gate is used to check the overflow?

A) OR

B) XOR

C) XNOR

D) NOR