ALU & Comparator Design

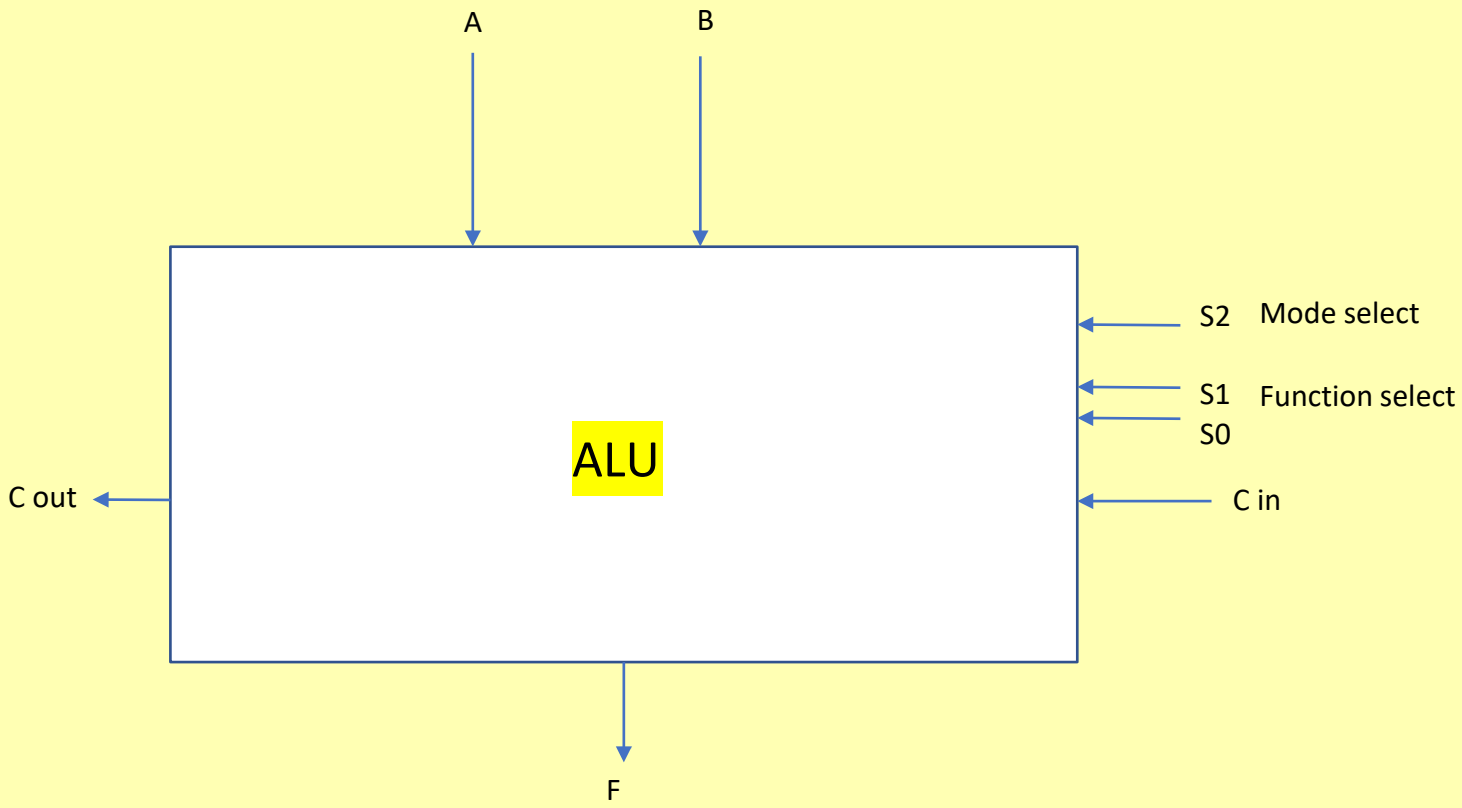
Ch-9.3 to 9.7 (Morris Mano)

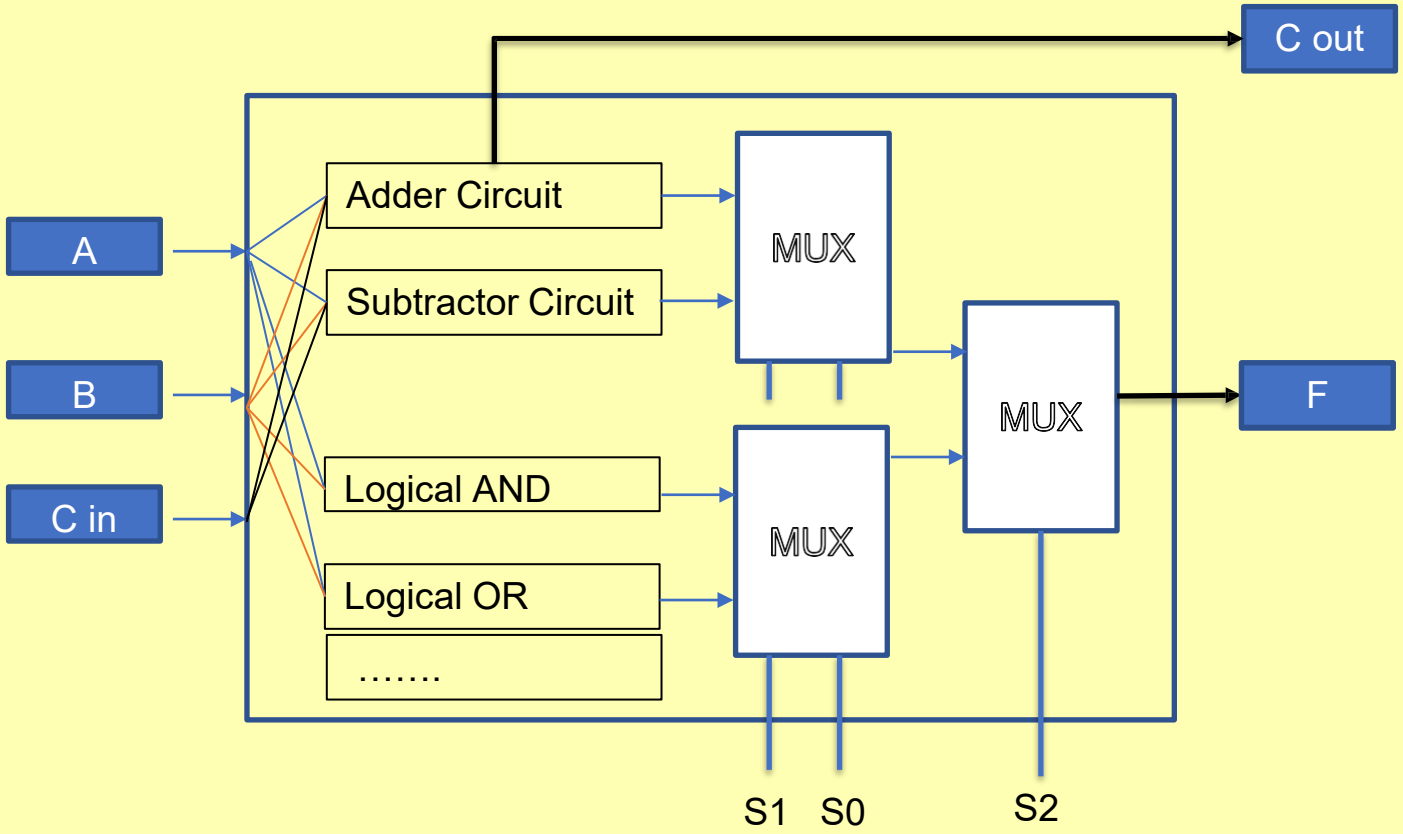
Courtesy To:

Lec Tasnim Ullah Shakib, MIST

Table of Contents

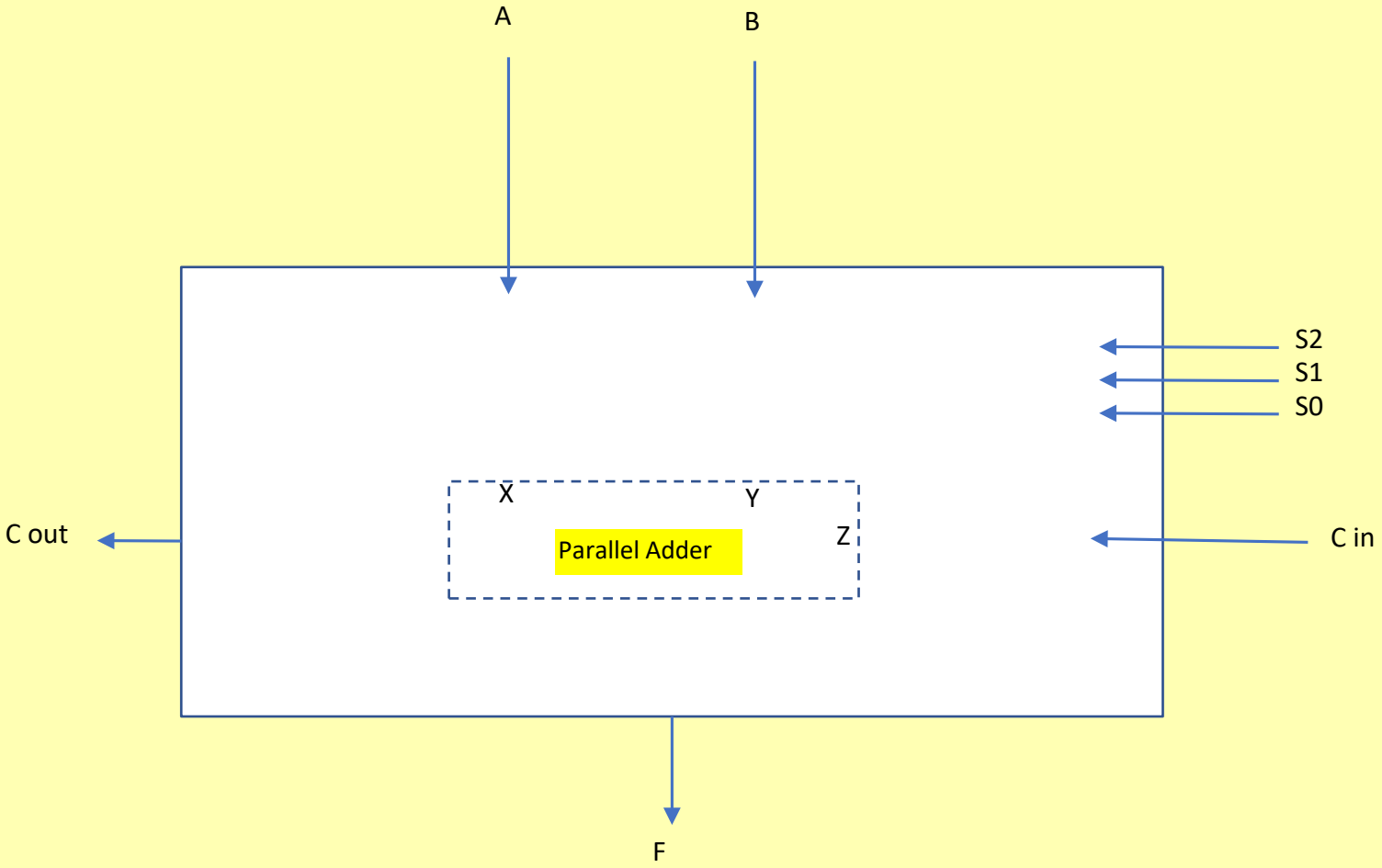
- Most CPU operations are performed by ALU.
- Control unit tells ALU which operation to perform.
- The result of the ALU operation is then stored into a register.
- ALU is the processor’s core component.
- To ensure a faster operation in the hardware, ALU only uses combinational logic gates for execution.


A
B
ALU
S2
Mode select
S1
Function select
S0
C out
C in
F

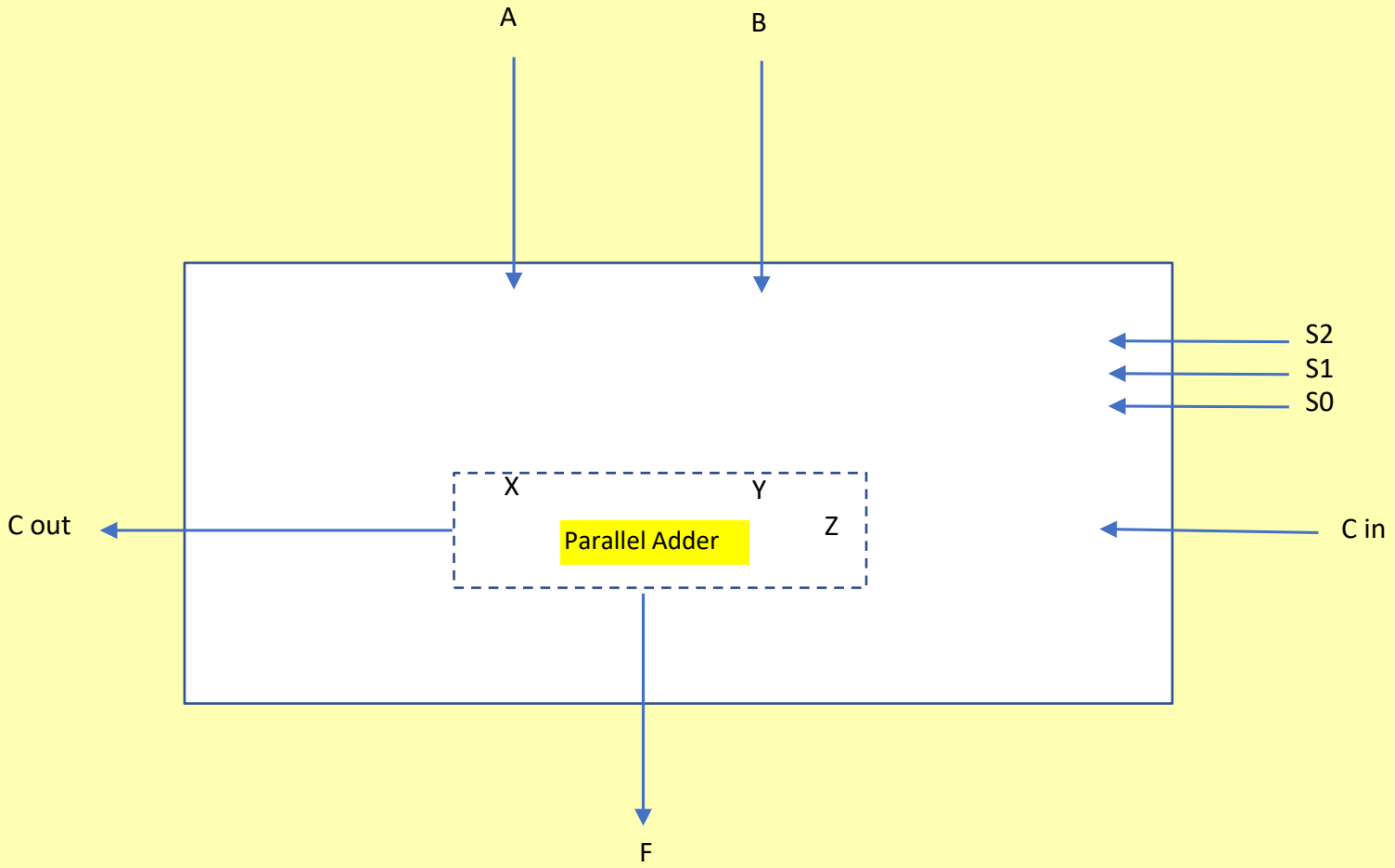


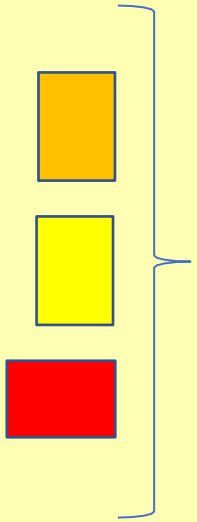
Inefficient Design: Design each of the arithmetic circuit and logic circuit separately

Use multiplexer and selection pins to connect them into a single output

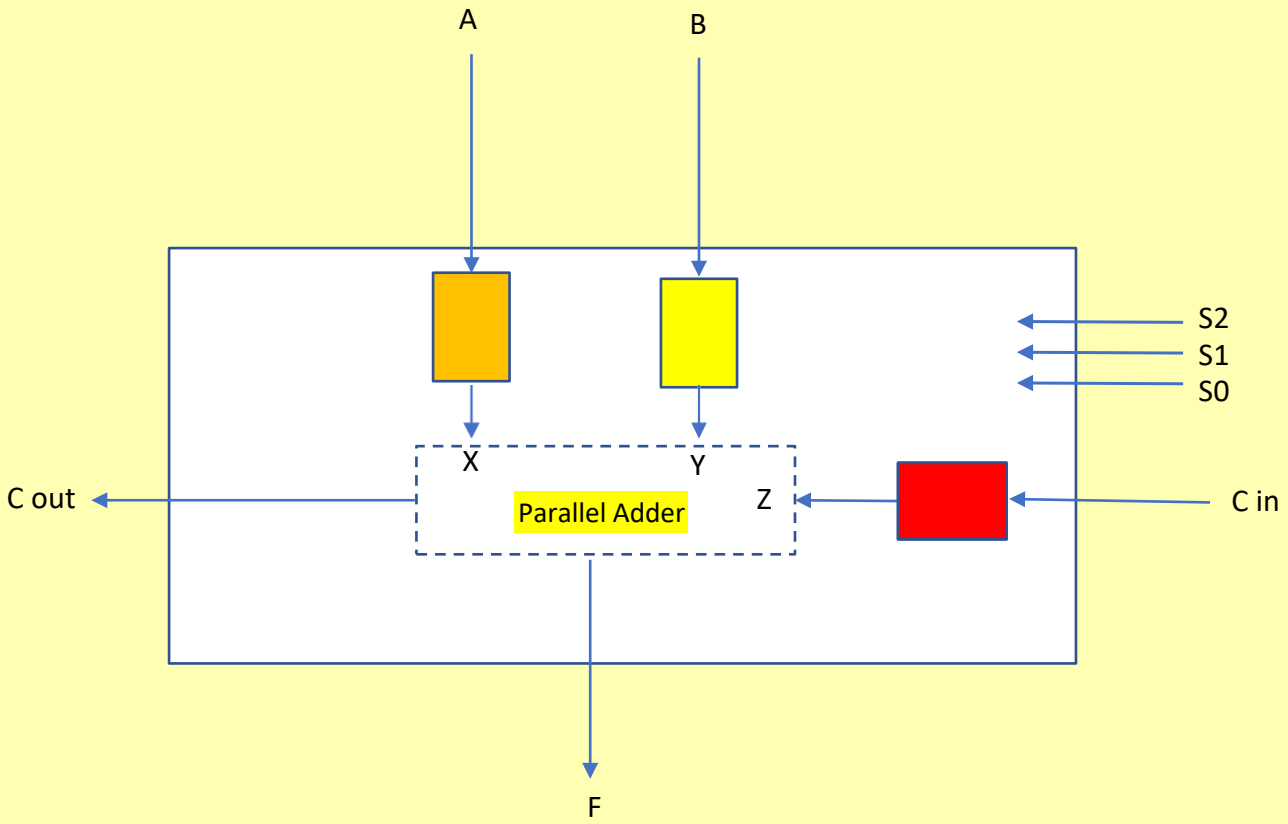


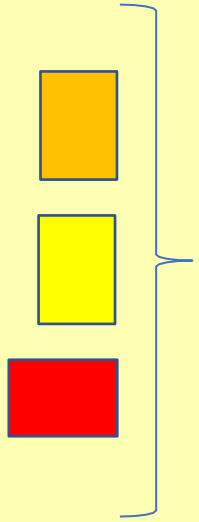
Instead, we design the ALU using Parallel Adder


A
B
S2
S1
S0
X
Y
Z
Parallel Adder
C out
C in
F

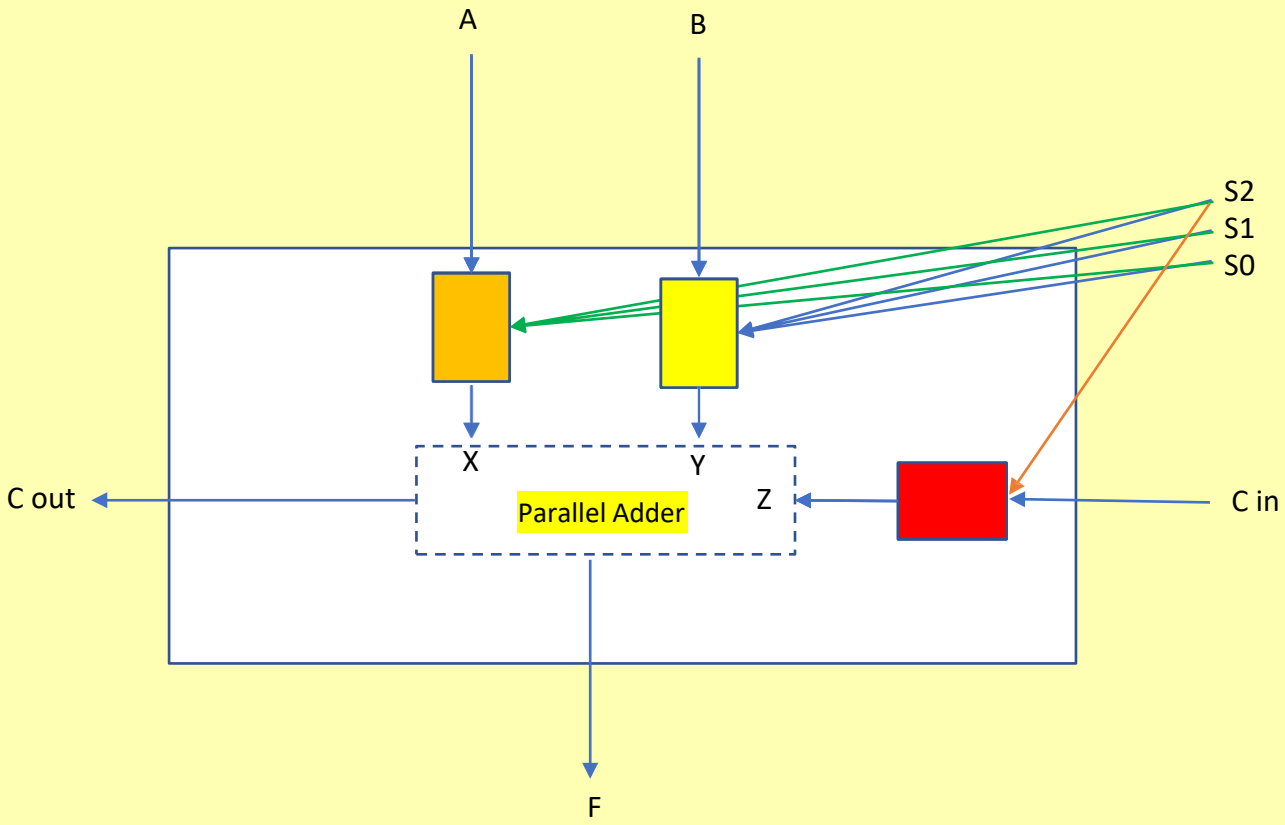


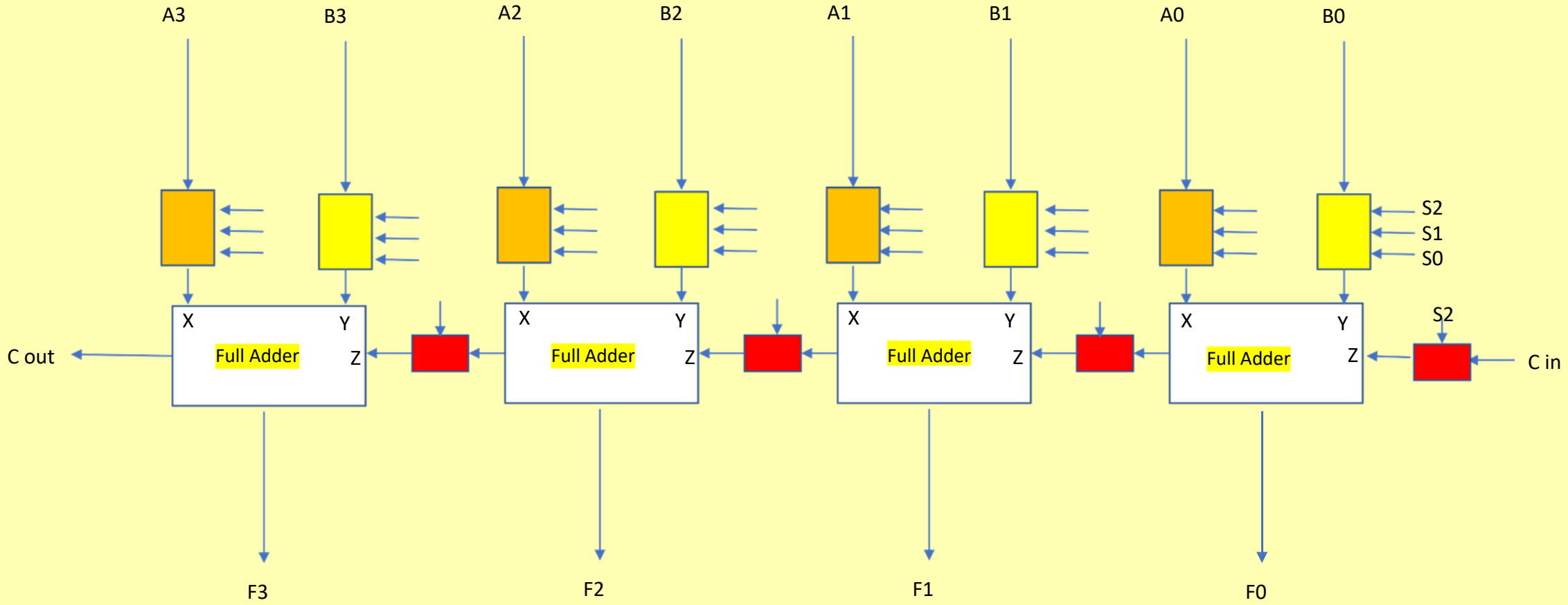
Combinational logic

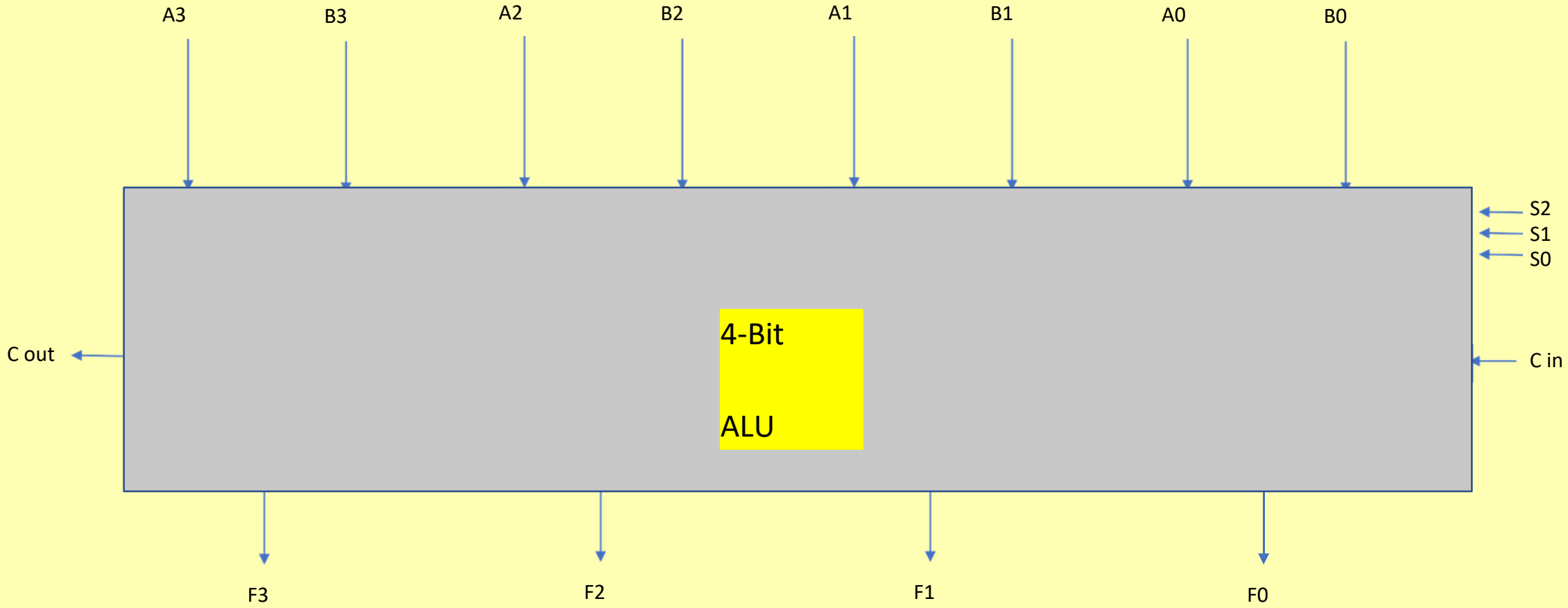




Combinational logic




A3
B3
A2
B2
A1
B1
A0
B0
S2
S1
S0
X
Y
Z
Full Adder
C out
C in
F3
F2
F1
F0


A3
B3
A2
B2
A1
B1
A0
B0
S2
S1
S0
C out
4-Bit
ALU
C in
F3
F2
F1
F0

Design of Arithmetic Unit

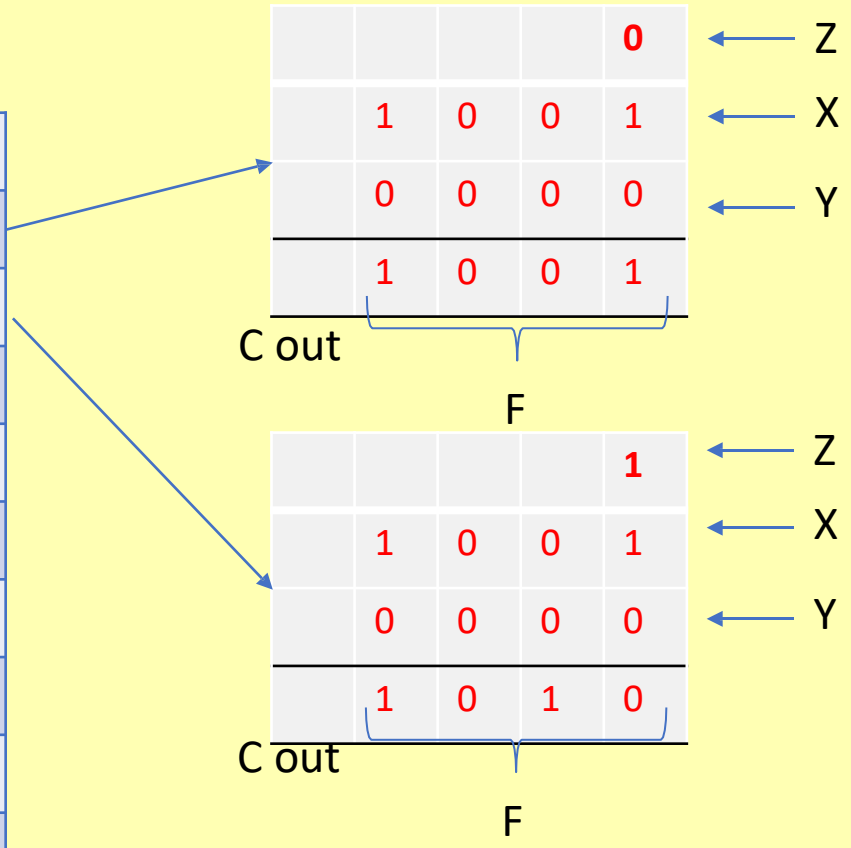
			1 bit	N bits	N bits	
	S1	**S0**	**C in**	**X**	**Y**	**F**

For 8 arithmetic operations, 3 selection inputs are necessary.

C in is one of the selectors. S1 and S0 are the other two.

Design of Arithmetic Unit

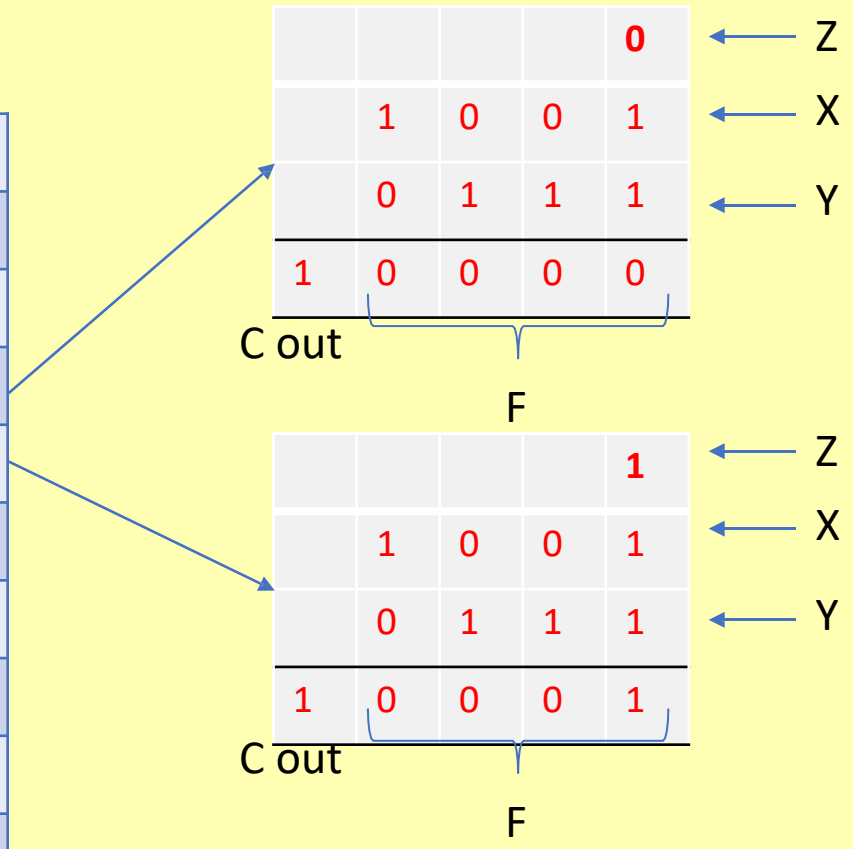
			1 bit	N bits	N bits	
	S1	**S0**	**C in**	**X**	**Y**	**F**
			0	A	0	A, Transfer A
			1	A	0	A+1, Increment A



For the arithmetic unit, assume that Z = C in.

Design of Arithmetic Unit

			1 bit	N bits	N bits	
	S1	**S0**	**C in**	**X**	**Y**	**F**
			0	A	0	A, Transfer A
			1	A	0	A+1, Increment A
			0	A	B	A+B, Addition
			1	A	B	A+B+1, Addition with Carry



Design of Arithmetic Unit

			1 bit	N bits	N bits	
	S1	**S0**	**C in**	**X**	**Y**	**F**
			0	A	0	A, Transfer A
			1	A	0	A+1, Increment A
			0	A	B	A+B, Addition
			1	A	B	A+B+1, Addition with Carry
			0	A	B’	A+B’, 1s complement Subtraction, Subtraction with borrow, A-B-1
			1	A	B’	A+B’+1, 2s complement Subtraction, Subtraction, A-B

1s complement of B = B’

2s complement of B = B’+1
Subtraction stands for A-B
A-B stands for A+B’+1
A+B’+1 = A-B
Thus, A+B’ = A-B-1 = Sub w Borrow

Design of Arithmetic Unit

			1 bit	N bits	N bits	
	S1	**S0**	**C in**	**X**	**Y**	**F**
			0	A	0	A, Transfer A
			1	A	0	A+1, Increment A
			0	A	B	A+B, Addition
			1	A	B	A+B+1, Addition with Carry
			0	A	B’	A+B’, 1s complement Subtraction, Subtraction with borrow, A-B-1
			1	A	B’	A+B’+1, 2s complement Subtraction, Subtraction, A-B
			0	A	all 1	A-1, Decrement A
			1	A	all 1	A, Transfer with carry

all 1 = 2^n -1
A + all 1 + 0 = A + (2^n -1)
= (A-1) + 2^n
A-1 causes the decrement
While, 2^n causes the carry out

all 1 = 2^n -1
A + all 1 +1 = A + (2^n -1) +1
= (A-1) + 2^n -1
= A + 2^n
A is the transfer operation
While, 2^n causes the carry out

Design of Arithmetic Unit

			1 bit	N bits	N bits	
	S1	**S0**	**C in**	**X**	**Y**	**F**
	0	0	0	A	0	A, Transfer A
	0	0	1	A	0	A+1, Increment A
			0	A	B	A+B, Addition
			1	A	B	A+B+1, Addition with Carry
			0	A	B’	A+B’, 1s complement Subtraction, Subtraction with borrow, A-B-1
			1	A	B’	A+B’+1, 2s complement Subtraction, Subtraction, A-B
			0	A	all 1	A-1, Decrement A
			1	A	all 1	A, Transfer with carry

Design of Arithmetic Unit

			1 bit	N bits	N bits	
	S1	**S0**	**C in**	**X**	**Y**	**F**
	0	0	0	A	0	A, Transfer A
	0	0	1	A	0	A+1, Increment A
	0	1	0	A	B	A+B, Addition
	0	1	1	A	B	A+B+1, Addition with Carry
			0	A	B’	A+B’, 1s complement Subtraction, Subtraction with borrow, A-B-1
			1	A	B’	A+B’+1, 2s complement Subtraction, Subtraction, A-B
			0	A	all 1	A-1, Decrement A
			1	A	all 1	A, Transfer with carry

Design of Arithmetic Unit

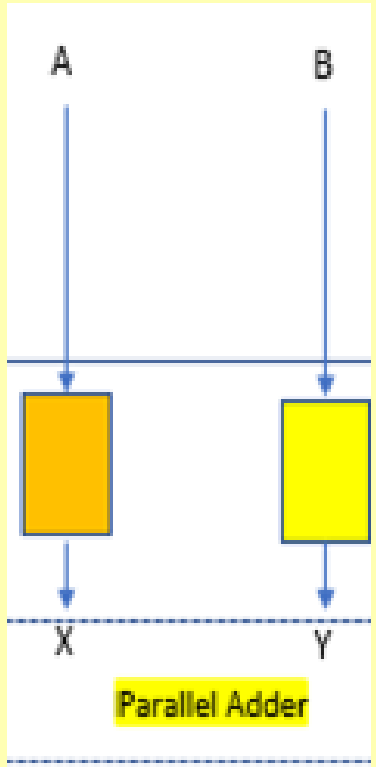
			1 bit	N bits	N bits	
	S1	**S0**	**C in**	**X**	**Y**	**F**
	0	0	0	A	0	A, Transfer A
	0	0	1	A	0	A+1, Increment A
	0	1	0	A	B	A+B, Addition
	0	1	1	A	B	A+B+1, Addition with Carry
	1	0	0	A	B’	A+B’, 1s complement Subtraction, Subtraction with borrow, A-B-1
	1	0	1	A	B’	A+B’+1, 2s complement Subtraction, Subtraction, A-B
			0	A	all 1	A-1, Decrement A
			1	A	all 1	A, Transfer with carry

Design of Arithmetic Unit

			1 bit	N bits	N bits	
	S1	**S0**	**C in**	**X**	**Y**	**F**
	0	0	0	A	0	A, Transfer A
	0	0	1	A	0	A+1, Increment A
	0	1	0	A	B	A+B, Addition
	0	1	1	A	B	A+B+1, Addition with Carry
	1	0	0	A	B’	A+B’, 1s complement Subtraction, Subtraction with borrow, A-B-1
	1	0	1	A	B’	A+B’+1, 2s complement Subtraction, Subtraction, A-B
	1	1	0	A	all 1	A-1, Decrement A
	1	1	1	A	all 1	A, Transfer with carry

Design of Arithmetic Unit

			1 bit	N bits	N bits	
	S1	**S0**	**C in**	**X**	**Y**	**F**
	0	0	0	A	0	A, Transfer A
	0	0	1	A	0	A+1, Increment A
	0	1	0	A	B	A+B, Addition
	0	1	1	A	B	A+B+1, Addition with Carry
	1	0	0	A	B'	A+B', 1s complement Subtraction, Subtraction with borrow, A-B-1
	1	0	1	A	B'	A+B'+1, 2s complement Subtraction, Subtraction, A-B
	1	1	0	A	all 1	A-1, Decrement A
	1	1	1	A	all 1	A, Transfer with carry



The equations would then be,

X=A

Y=0 when S1S0=00
B when S1S0=01
B'when S1S0=10
1 when S1S0=11

Thus,

Y=0S1'S0' + BS1'S0 + B'S1S0' + 1S1S0
=BS1'S0 + B'S1S0' + S1S0
=BS1'S0 + B'S1S0' + (B+B')S1S0
=BS1'S0 + B'S1S0' + BS1S0 + B'S1S0
=BS0(S1'+S1) + B'S1(S0'+S0)

Y=BS0 + B'S1

We did not consider S2 selection pin in these equations as we only designed the arithmetic unit so far.

Design of Arithmetic Unit

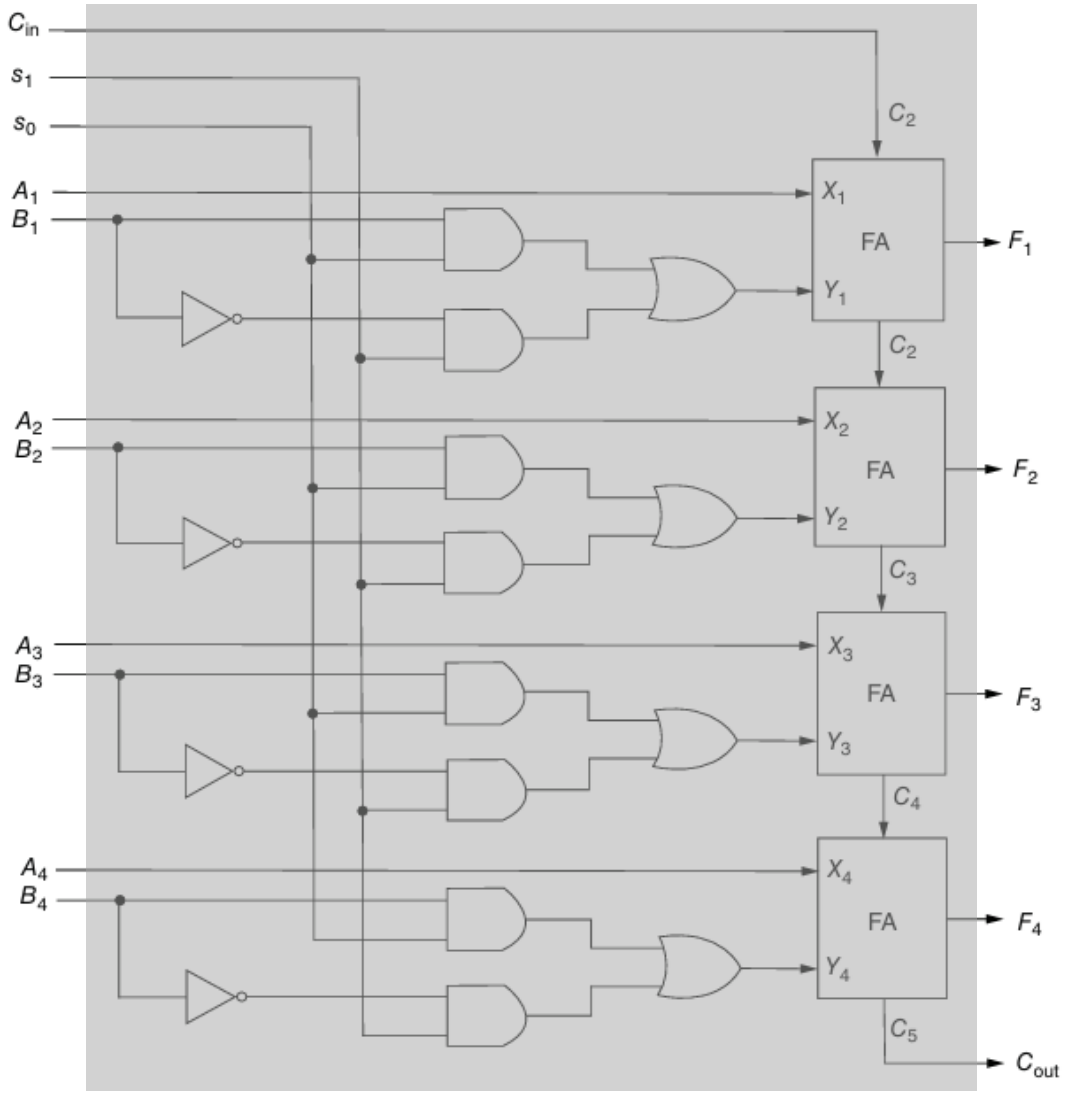
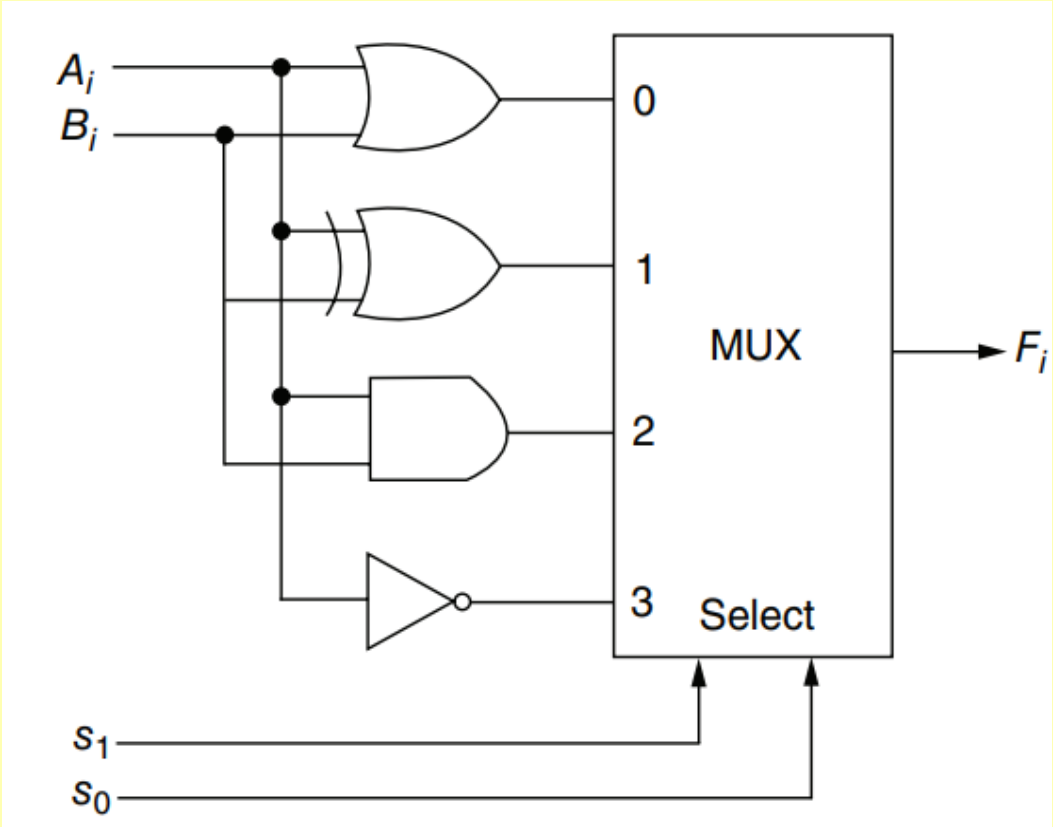


Figure 9-8 Logic diagram of arithmetic circuit

Design of Logical Unit



This is an inefficient design.
A more efficient ALU can be obtained if we investigate the possibility of generating logic operations in an already available arithmetic circuit.

Design of Logical Unit

			1 bit	N bits	N bits	
	S2	**S1**	**S0**	**C in**	**X**	**Y**	**F**
Arithmetic when S2=0	0	0	0	0	A	0	A, Transfer A
	0	0	0	1	A	0	A+1, Increment A
	0	0	1	0	A	B	A+B, Addition
	0	0	1	1	A	B	A+B+1, Addition with Carry
	0	1	0	0	A	B’	A+B’, 1s complement Subtraction, Subtraction with borrow, A-B-1
	0	1	0	1	A	B’	A+B’+1, 2s complement Subtraction, Subtraction, A-B
	0	1	1	0	A	all 1	A-1, Decrement A
	0	1	1	1	A	all 1	A, Transfer with carry
Logical when S2=1	1			X			
	1			X			
	1			X			
	1			X			

We introduce S2 for arithmetic and logical operations.

During logical operations, we would want to inhibit the carry in, Cin into the adder circuit. As, a carry would effect the result of the logical operation.

Now that we are inhibiting the Cin, So, Cin is DON’T CARE (X). Hence, we cannot have 8 logical operations, but 4 logical operations instead.

Design of Logical Unit

			1 bit	N bits	N bits	
	S2	**S1**	**S0**	**C in**	**X**	**Y**	**F**
Arithmetic when S2=0	0	0	0	0	A	0	A, Transfer A
	0	0	0	1	A	0	A+1, Increment A
	0	0	1	0	A	B	A+B, Addition
	0	0	1	1	A	B	A+B+1, Addition with Carry
	0	1	0	0	A	B’	A+B’, 1s complement Subtraction, Subtraction with borrow, A-B-1
	0	1	0	1	A	B’	A+B’+1, 2s complement Subtraction, Subtraction, A-B
	0	1	1	0	A	all 1	A-1, Decrement A
	0	1	1	1	A	all 1	A, Transfer with carry
Logical when S2=1	1	0	0	X	A	0	
	1			X			
	1			X			
	1			X			

Recall that,
X=A
Y=BS0 + B’S1

Design of Logical Unit

			1 bit	N bits	N bits	
	S2	**S1**	**S0**	**C in**	**X**	**Y**	**F**
Arithmetic when S2=0	0	0	0	0	A	0	A, Transfer A
	0	0	0	1	A	0	A+1, Increment A
	0	0	1	0	A	B	A+B, Addition
	0	0	1	1	A	B	A+B+1, Addition with Carry
	0	1	0	0	A	B’	A+B’, 1s complement Subtraction, Subtraction with borrow, A-B-1
	0	1	0	1	A	B’	A+B’+1, 2s complement Subtraction, Subtraction, A-B
	0	1	1	0	A	all 1	A-1, Decrement A
	0	1	1	1	A	all 1	A, Transfer with carry
Logical when S2=1	1	0	0	X	A	0	
	1	0	1	X	A	B	
	1			X			
	1			X			

Recall that,
X=A
Y=BS0 + B’S1

Design of Logical Unit

			1 bit	N bits	N bits	
	S2	**S1**	**S0**	**C in**	**X**	**Y**	**F**
Arithmetic when S2=0	0	0	0	0	A	0	A, Transfer A
	0	0	0	1	A	0	A+1, Increment A
	0	0	1	0	A	B	A+B, Addition
	0	0	1	1	A	B	A+B+1, Addition with Carry
	0	1	0	0	A	B’	A+B’, 1s complement Subtraction, Subtraction with borrow, A-B-1
	0	1	0	1	A	B’	A+B’+1, 2s complement Subtraction, Subtraction, A-B
	0	1	1	0	A	all 1	A-1, Decrement A
	0	1	1	1	A	all 1	A, Transfer with carry
Logical when S2=1	1	0	0	X	A	0	
	1	0	1	X	A	B	
	1	1	0	X	A	B’	
	1			X			

Recall that,
X=A
Y=BS0 + B’S1

Design of Logical Unit

			1 bit	N bits	N bits	
	S2	**S1**	**S0**	**C in**	**X**	**Y**	**F**
Arithmetic when S2=0	0	0	0	0	A	0	A, Transfer A
	0	0	0	1	A	0	A+1, Increment A
	0	0	1	0	A	B	A+B, Addition
	0	0	1	1	A	B	A+B+1, Addition with Carry
	0	1	0	0	A	B’	A+B’, 1s complement Subtraction, Subtraction with borrow, A-B-1
	0	1	0	1	A	B’	A+B’+1, 2s complement Subtraction, Subtraction, A-B
	0	1	1	0	A	all 1	A-1, Decrement A
	0	1	1	1	A	all 1	A, Transfer with carry
Logical when S2=1	1	0	0	X	A	0	
	1	0	1	X	A	B	
	1	1	0	X	A	B’	
	1	1	1	X	A	all 1	

Recall that,
X=A
Y=BS0 + B’S1

Design of Logical Unit

			1 bit	N bits	N bits	
	S2	**S1**	**S0**	**C in**	**X**	**Y**	**F**
Arithmetic when S2=0	0	0	0	0	A	0	A, Transfer A
	0	0	0	1	A	0	A+1, Increment A
	0	0	1	0	A	B	A+B, Addition
	0	0	1	1	A	B	A+B+1, Addition with Carry
	0	1	0	0	A	B’	A+B’, 1s complement Subtraction, Subtraction with borrow, A-B-1
	0	1	0	1	A	B’	A+B’+1, 2s complement Subtraction, Subtraction, A-B
	0	1	1	0	A	all 1	A-1, Decrement A
	0	1	1	1	A	all 1	A, Transfer with carry
Logical when S2=1	1	0	0	X	A	0	
	1	0	1	X	A	B	
	1	1	0	X	A	B’	
	1	1	1	X	A	all 1	

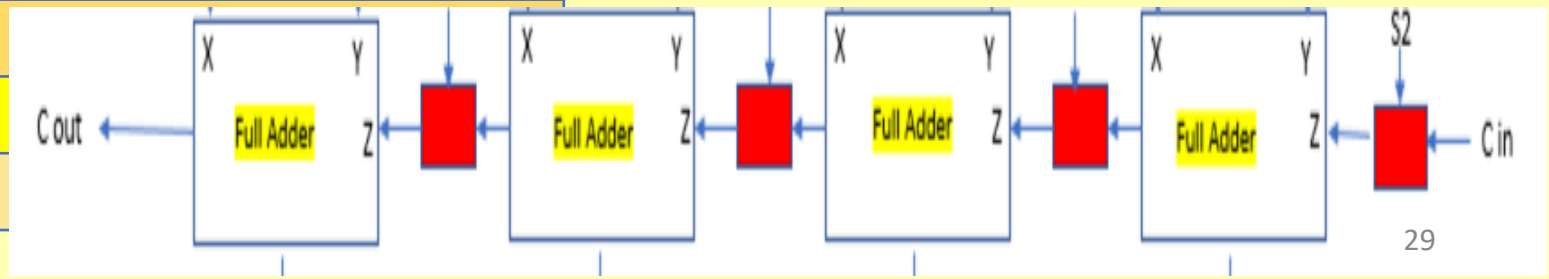
For arithmetic operations (***S2=0***),

Z = Cin.

And for logical operations (***S2=1***), whatever be the value of input carry, ***Z=0***.

Thus, **Z = S2’.Cin + S2.0**

Z = S2’.Cin



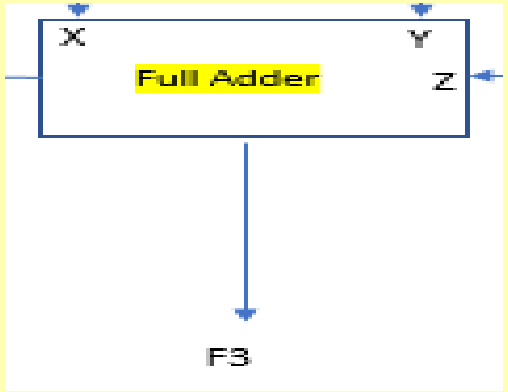
Design of Logical Unit

			1 bit	N bits	N bits	
	S2	**S1**	**S0**	**C in**	**X**	**Y**	**F**
Arithmetic when S2=0	0	0	0	0	A	0	A, Transfer A
	0	0	0	1	A	0	A+1, Increment A
	0	0	1	0	A	B	A+B, Addition
	0	0	1	1	A	B	A+B+1, Addition with Carry
	0	1	0	0	A	B’	A+B’, 1s complement Subtraction, Subtraction with borrow, A-B-1
	0	1	0	1	A	B’	A+B’+1, 2s complement Subtraction, Subtraction, A-B
	0	1	1	0	A	all 1	A-1, Decrement A
	0	1	1	1	A	all 1	A, Transfer with carry
Logical when S2=1	1	0	0	X	A	0	
	1	0	1	X	A	B	
	1	1	0	X	A	B’	
	1	1	1	X	A	all 1	

X=A

Y=BS0 + B’S1

Z = S2’.Cin



Full Adder Output Sum Equation,

F = X⊕Y⊕Z

= X⊕Y⊕0 [Z=0 when S2=0]

= X⊕Y

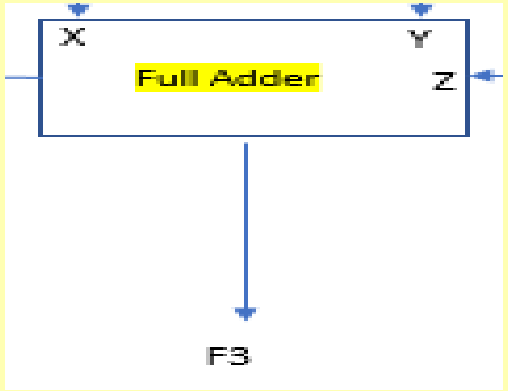
Design of Logical Unit

			1 bit	N bits	N bits	
	S2	S1	S0	C in	X	Y	F
Arithmetic when S2=0	0	0	0	0	A	0	A, Transfer A
	0	0	0	1	A	0	A+1, Increment A
	0	0	1	0	A	B	A+B, Addition
	0	0	1	1	A	B	A+B+1, Addition with Carry
	0	1	0	0	A	B’	A+B’, 1s complement Subtraction, Subtraction with borrow, A-B-1
	0	1	0	1	A	B’	A+B’+1, 2s complement Subtraction, Subtraction, A-B
	0	1	1	0	A	all 1	A-1, Decrement A
	0	1	1	1	A	all 1	A, Transfer with carry
Logical when S2=1	1	0	0	X	A	0	A
	1	0	1	X	A	B	A XOR B
	1	1	0	X	A	B’	A XNOR B
	1	1	1	X	A	all 1	NOT A

X=A

Y=BS0 + B’S1

Z = S2’.Cin



F = X⊕Y = A⊕0 = A

F = X⊕Y = A⊕B

F = X⊕Y = A⊕B’ = AB+A’B’ =

A⊙B F = X⊕Y = A⊕1 = A.0+A’.1

= A’

Design of Logical Unit

			1 bit	N bits	N bits	
	S2	**S1**	**S0**	**C in**	**X**	**Y**	**F**
Arithmetic when S2=0	0	0	0	0	A	0	A, Transfer A
	0	0	0	1	A	0	A+1, Increment A
	0	0	1	0	A	B	A+B, Addition
	0	0	1	1	A	B	A+B+1, Addition with Carry
	0	1	0	0	A	B’	A+B’, 1s complement Subtraction, Subtraction with borrow, A-B-1
	0	1	0	1	A	B’	A+B’+1, 2s complement Subtraction, Subtraction, A-B
	0	1	1	0	A	all 1	A-1, Decrement A
	0	1	1	1	A	all 1	A, Transfer with carry
Logical when S2=1	1	0	0	X	A	0	A **BUT I WANT A OR B**
	1	0	1	X	A	B	A XOR B
	1	1	0	X	A	B’	A XNOR B **BUT I WANT A AND B**
	1	1	1	X	A	all 1	NOT A

Design of Logical Unit

			1 bit	N bits	N bits	
S2	**S1**	**S0**	**C in**	**X**	**Y**	**F**
0	0	0	0	A+K	0	A, Transfer A
0	0	0	1	A+K	0	A+1, Increment A
0	0	1	0	A+K	B	A+B, Addition
0	0	1	1	A+K	B	A+B+1, Addition with Carry
0	1	0	0	A+K	B’	A+B’, 1s complement Subtraction, Subtraction with borrow, A-B-1
0	1	0	1	A+K	B’	A+B’+1, 2s complement Subtraction, Subtraction, A-B
0	1	1	0	A+K	all 1	A-1, Decrement A
0	1	1	1	A+K	all 1	A, Transfer with carry
1	0	0	X	A+K	0	A **BUT I WANT A OR B**
1	0	1	X	A+K	B	A XOR B
1	1	0	X	A+K	B’	A XNOR B **BUT I WANT A AND B**
1	1	1	X	A+K	all 1	NOT A

Arithmetic when S2=0

Logical when S2=1

We would try to place such values in K, so that,

F = X⊕Y⊕Z

= (A+K) ⊕ Y ⊕ Z

gives us the desired result for some operations, while the result of other operations remains unchanged.

Design of Logical Unit

			1 bit	N bits	N bits	
S2	**S1**	**S0**	**C in**	**X**	**Y**	**F**
0	0	0	0	A+**0**	0	A, Transfer A
0	0	0	1	A+**0**	0	A+1, Increment A
0	0	1	0	A+**0**	B	A+B, Addition
0	0	1	1	A+**0**	B	A+B+1, Addition with Carry
0	1	0	0	A+**0**	B’	A+B’, 1s complement Subtraction, Subtraction with borrow, A-B-1
0	1	0	1	A+**0**	B’	A+B’+1, 2s complement Subtraction, Subtraction, A-B
0	1	1	0	A+**0**	all 1	A-1, Decrement A
0	1	1	1	A+**0**	all 1	A, Transfer with carry
1	0	0	X	A+**B**	0	**A OR B**
1	0	1	X	A+**0**	B	A XOR B
1	1	0	X	A+**B’**	B’	**A AND B**
1	1	1	X	A+**0**	all 1	NOT A

Arithmetic when S2=0

Logical when S2=1

F = X⊕Y⊕Z
= (A+0)⊕Y⊕C in
= A⊕Y⊕C in
Thus we OR 0 with A so that the result remains unchanged.

F = X⊕Y = (A+B)⊕0 = A+B = A OR B

F = X⊕Y = (A+0)⊕B = A⊕B

F = X⊕Y = (A+B’)⊕B’
= AB+B’B+ABB’ = AB = A AND B

F = X⊕Y = (A+0)⊕1
= A⊕1 = A.0+A’.1 = A’

Design of Logical Unit

			1 bit	N bits	N bits	
S2	**S1**	**S0**	**C in**	**X**	**Y**	**F**
0	0	0	0	A+**0**	0	A, Transfer A
0	0	0	1	A+**0**	0	A+1, Increment A
0	0	1	0	A+**0**	B	A+B, Addition
0	0	1	1	A+**0**	B	A+B+1, Addition with Carry
0	1	0	0	A+**0**	B’	A+B’, 1s complement Subtraction, Subtraction with borrow, A-B-1
0	1	0	1	A+**0**	B’	A+B’+1, 2s complement Subtraction, Subtraction, A-B
0	1	1	0	A+**0**	all 1	A-1, Decrement A
0	1	1	1	A+**0**	all 1	A, Transfer with carry
1	0	0	X	A+**B**	0	**A OR B**
1	0	1	X	A+**0**	B	A XOR B
1	1	0	X	A+**B’**	B’	**A AND B**
1	1	1	X	A+**0**	all 1	NOT A

Arithmetic when S2=0

Logical when S2=1

So, K=B when S2S1S0=100
and B’ when S2S1S0=110

Thus, modifying the previous equation of X,

X = A + S2S1’S0’B + S2S1S0’B’

Final ALU Design

		1 bit	N bits	N bits			
	S2	S1	S0	C in	X	Y	F
Arithmetic when S2=0	0	0	0	0	A+**0**	0	A, Transfer A
	0	0	0	1	A+**0**	0	A+1, Increment A
	0	0	1	0	A+**0**	B	A+B, Addition
	0	0	1	1	A+**0**	B	A+B+1, Addition with Carry
	0	1	0	0	A+**0**	B'	A+B', 1s complement Subtraction, Subtraction with borrow, A-B-1
	0	1	0	1	A+**0**	B'	A+B'+1, 2s complement Subtraction, Subtraction, A-B
	0	1	1	0	A+**0**	all 1	A-1, Decrement A
	0	1	1	1	A+**0**	all 1	A, Transfer with carry
Logical when S2=1	1	0	0	X	A+**B**	0	**A OR B**
	1	0	1	X	A+**0**	B	A XOR B
	1	1	0	X	A+**B'**	B'	**A AND B**
	1	1	1	X	A+**0**	all 1	NOT A

Function Table of ALU

X = A + S2S1'S0'B + S2S1S0'B'
Y=BS0 + B'S1
Z = S2'.Cin

Equations

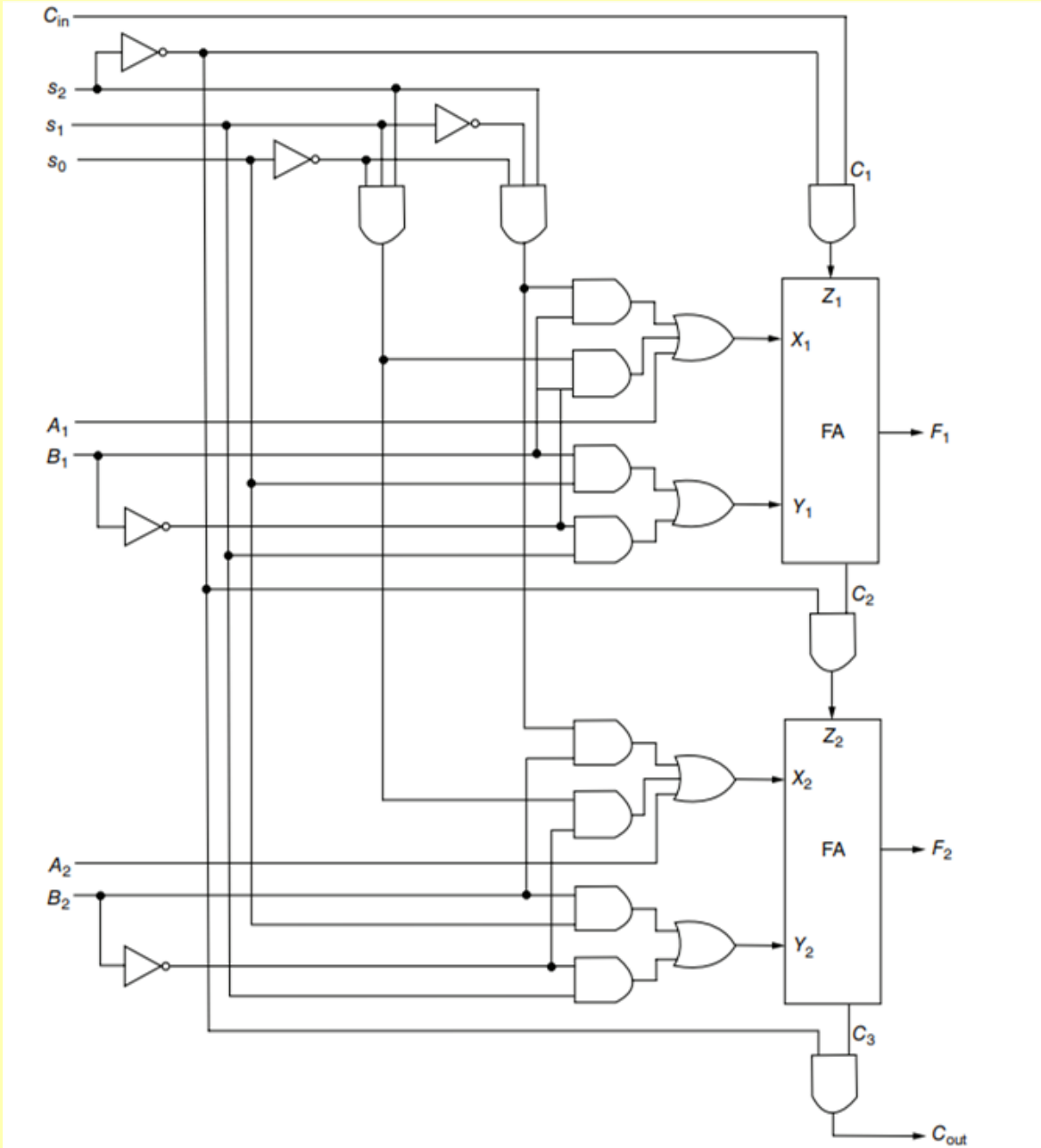


Figure 9-13 Logic diagram of arithmetic logic unit (ALU)

(For 2 bits)

Generating all possible basic logical operations

Or-ing K with A, F = X⊕Y⊕Z

= (A+K)⊕Y⊕0

= (A+K)⊕Y

16 combinations of 0, B, B', 1
gives us different logical operation results

F = (A+K)⊕Y
= (A+B')⊕B
= A'B+AB'+B'
= A'B+(A+1)B'
= A'B+(A'+1)B'
= A'B+A'B'+B'
= A'(B+B')+B'
= A'+B'
= (A.B)'
= A NAND B

	K	Y	F = (A+K)⊕Y
A	0	0	A
	0	B	A XOR B
	0	B'	A XNOR B
	0	1	NOT A
	1	0	All 1
	1	B	NOT B
	1	B'	B
	1	1	All 0
	B	0	A OR B
	B	B	
	B	B'	
	B	1	A NOR B
	B'	0	
	B'	B	A NAND B
	B'	B'	A AND B
	B'	1	

Generating all possible basic logical operations

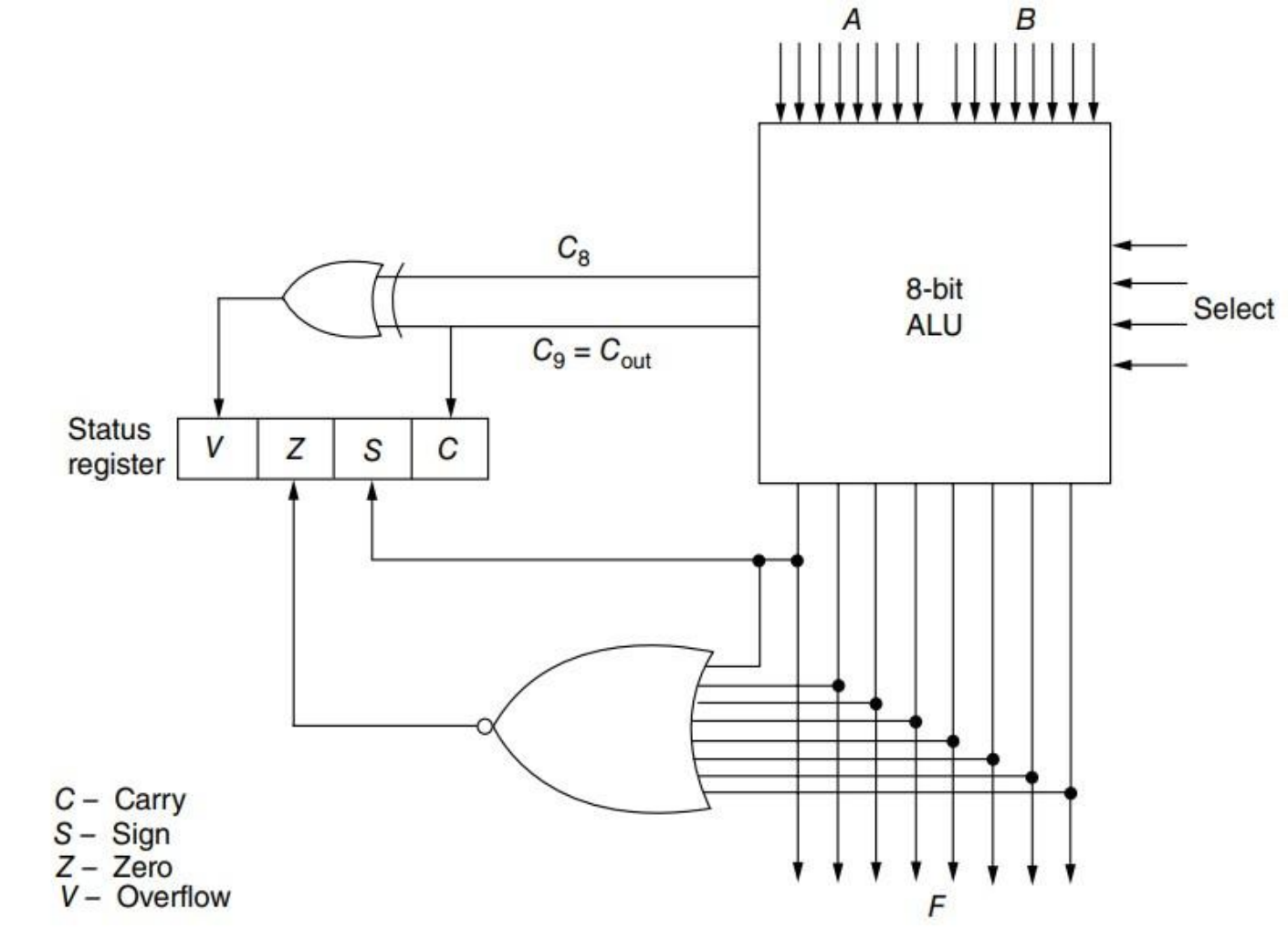
Similarly
Or-ing K with B, F = X⊕Y⊕Z
= X⊕(B+K)⊕0
= X⊕(B+K)

16 combinations of 0, A, A', 1
gives us different logical operation results

F = X⊕(B+K)
= A⊕(B+A')
= A'B+A'+AB'
= A'(B+1)+AB'
= A'(B'+1)+AB'
= A'B'+A'+AB'
= B'(A'+A')+A'
= B'+A'
= (B.A)'
= B NAND A

	K	X	F = X⊕(B+K)
B	0	0	B
	0	A	B XOR A
	0	A'	B XNOR A
	0	1	NOT B
	1	0	All 1
	1	A	NOT A
	1	A'	A
	1	1	All 0
	A	0	B OR A
	A	A	
	A	A'	
	A	1	B NOR A
	A'	0	
	A'	A	B NAND A
	A'	A'	B AND A
	A'	1	

Status or Flag Registers



- Carry can be detected when the Cout is 1.
- Sign is 1 when MSB is 1 in signed ALU operation.
- Overflow occurs when the carry-out of the 2 MSB are different.
- Zero Flag is set when all the bits of the result are 0. i.e.
  Z=F1'F2'…F7'F8'
  Z=(F1+F2+…+F7+F8)'

Interpreting Flag Registers & Comparator Design

Status bits after the subtraction of unsigned numbers ($A - B$)

Relation	Condition of Status Bits	Boolean Function
A = B	Z=1	Z
A ≠ B	Z=0	Z’
A >= B	C=1	C
A < B	C=0	C’
A > B	C=1 and Z=0	CZ’
A <= B	C=0 or Z=1	C’+Z

Opposite

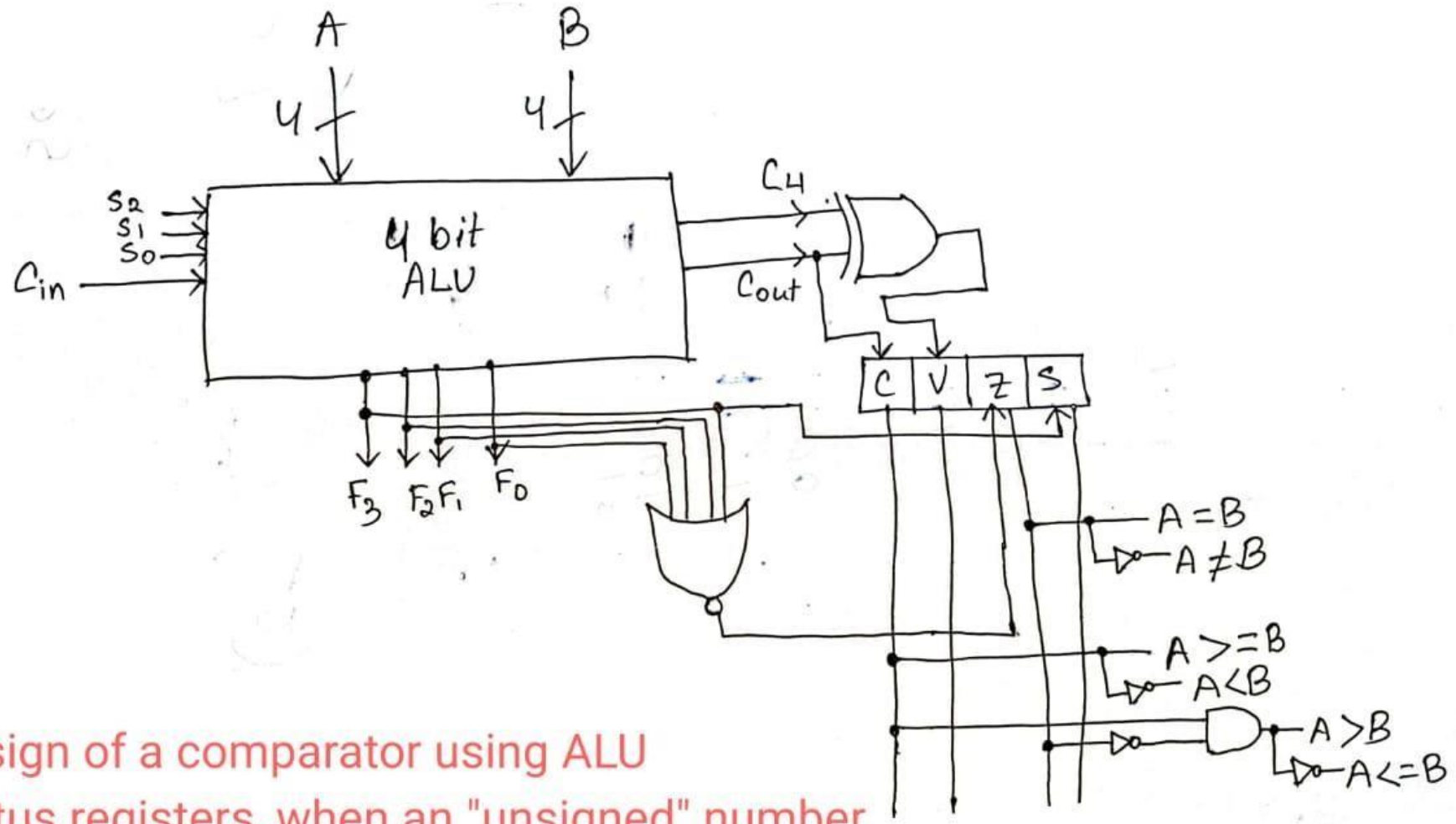
Opposite

Opposite

In case of unsigned operation, we are not concerned with the sign flag and overflow flag.

There is no signed overflow in unsigned operation.

Interpreting Flag Registers & Comparator Design



Design of a comparator using ALU status registers, when an "unsigned" number subtraction A-B is performed.

Interpreting Flag Registers & Comparator Design

Status bits after the subtraction of sign-2's complement numbers ($A - B$)

Relation	Condition of Status Bits	Boolean Function
A = B	Z=1	Z
A ≠ B	Z=0	Z'
A >= B	(S=0 and V=0) or (S=1 and V=1)	S'V' + SV = S⊙V
A < B	(S=1 and V=0) or (S=0 and V=1)	SV' + S'V = S⊕V
A > B	Z=0 and ((S=0 and V=0) or (S=1 and V=1))	Z'(S⊙V)
A <= B	Z=1 or ((S=1 and V=0) or (S=0 and V=1))	Z' + (S⊕V)

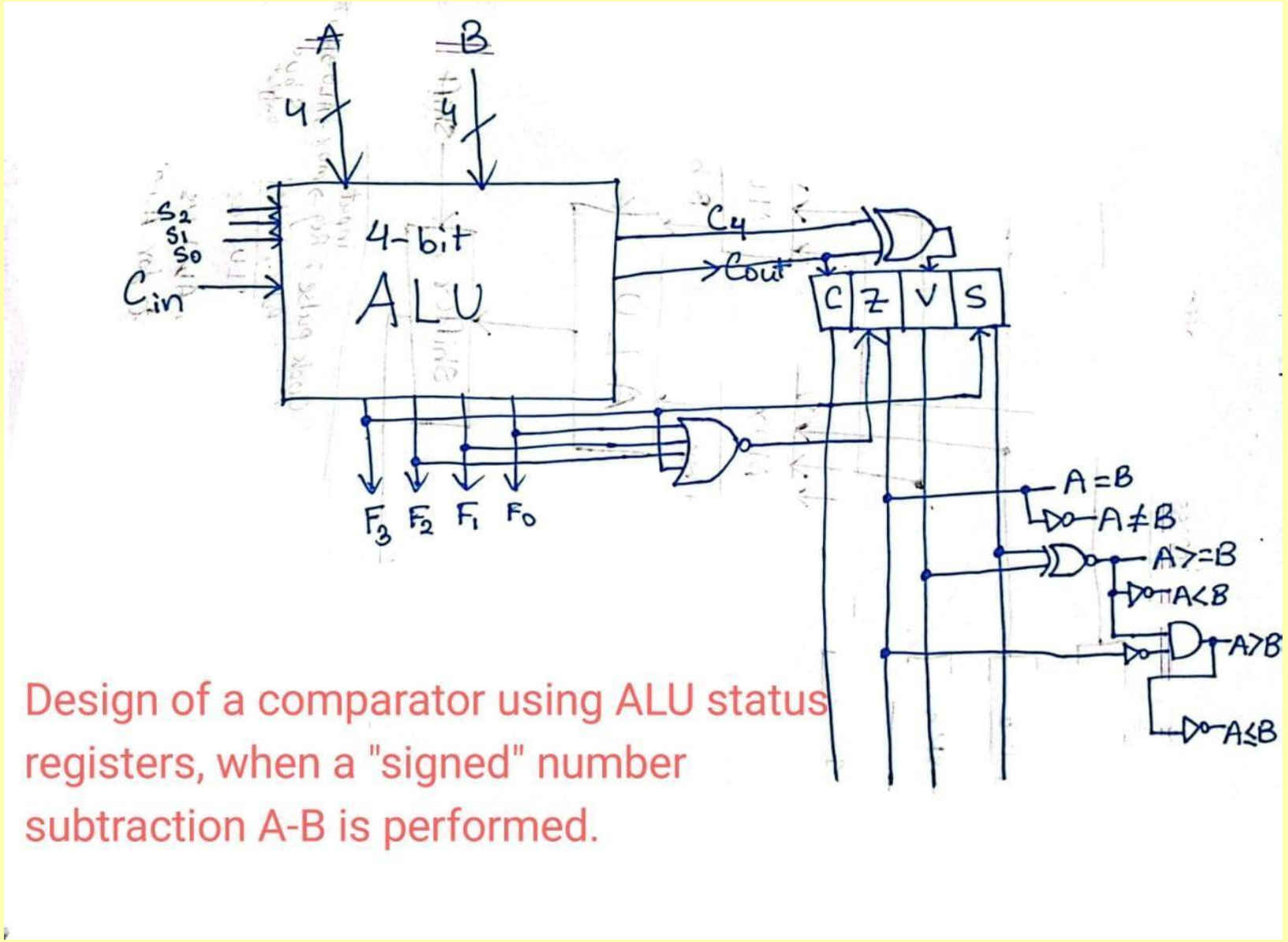
Opposite

Opposite

Opposite

In case of signed operation, we are not concerned with the carry flag, as a carry out cannot tell anything about the signed result of an operation.
Recall that, signed subtraction can eventually be converted to signed addition.

Interpreting Flag Registers & Comparator Design



Design of a comparator using ALU status registers, when a "signed" number subtraction A-B is performed.

Thank You