Joel D. Schipper

LECTURE #9: Adders, Comparators, and ALU's

EEL 3701: Digital Logic and Computer Systems

Based on lecture notes by Dr. Eric M. Schwartz

Adders:

-Adding 2-bits (2 input lines)

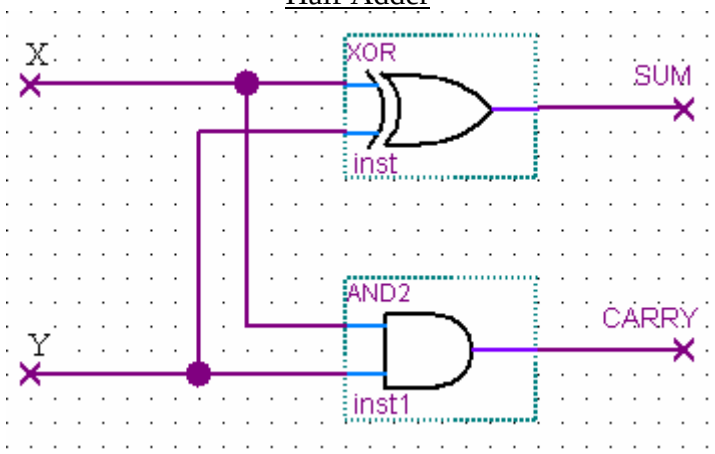
-4 possible combos: $0+0=0,\ 0+1=1,\ 1+0=1,\ 1+1=10$

-Requires 2 output bits: "Sum" and "Carry."

X	Y	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$Sum = \overline{X}Y + X\overline{Y} = X \oplus Y$, $Carry = XY$

Half Adder



-Half adders "carry out" but do not allow for "carry in" values.

Sum (S)

XY\C$_{in}$	0	1
00	0	1
01	1	0
11	0	1
10	1	0

$S = \overline{X}\ \overline{Y}C_{in} + \overline{X}Y\overline{C}_{in} + X\overline{Y}\ \overline{C}_{in} + XYC_{in}$

$S = (\overline{X}\ \overline{Y} + XY)C_{in} + (\overline{X}Y + X\overline{Y})\overline{C}_{in}$

$S = (\overline{X \oplus Y})C_{in} + (X \oplus Y)\overline{C}_{in}$

$S = X \oplus Y \oplus C_{in}$

C_{out}

XY\C$_{in}$	0	1
00	0	0
01	0	1
11	1	1
10	0	1

$C_{out} = XY + XC_{in} + YC_{in}$

$C_{out} = XY + X\overline{Y}C_{in} + \overline{X}YC_{in}$

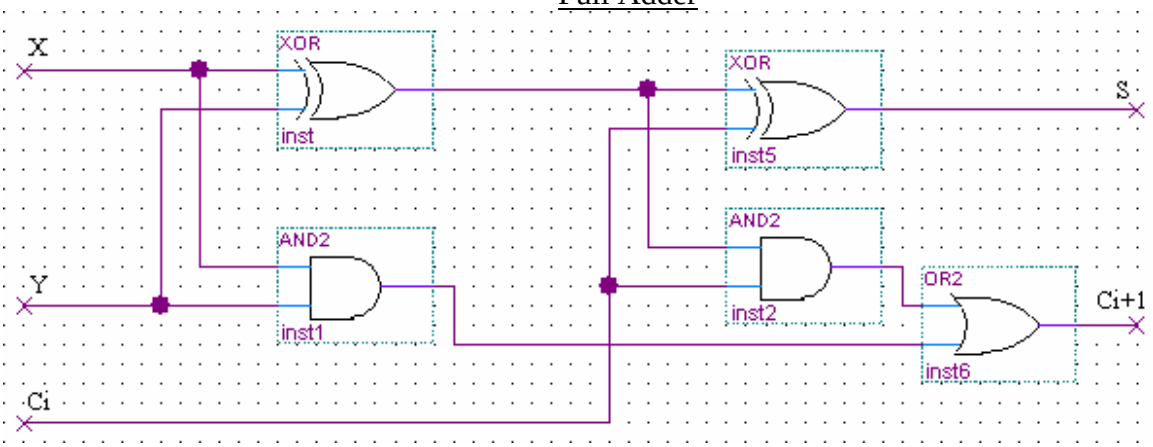
$C_{out} = XY + (X\overline{Y} + \overline{X}Y)C_{in}$

$C_{out} = XY + (X \oplus Y)C_{in}$

Note: The first equation under C_{out} can be implemented directly with AND and OR gates.

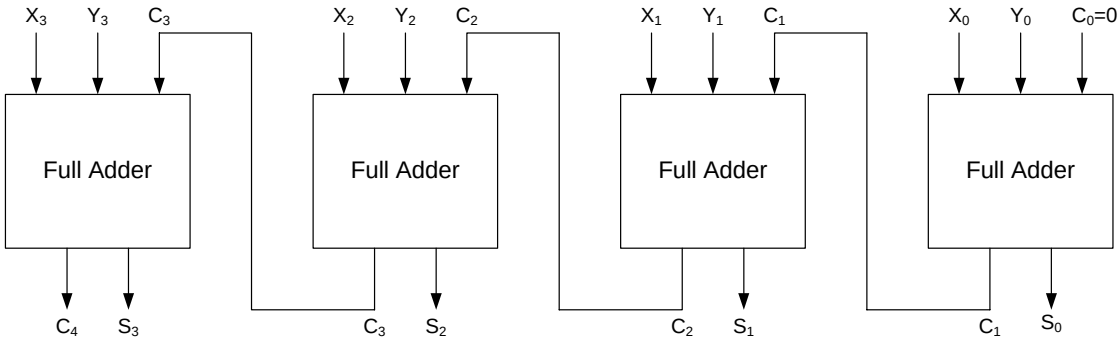
Note 2: The second equation under C_{out} can be constructed simply by not fully reducing the equation from the K-map.

Full Adder



Note: This is made from 2 half adders and an OR gate.
Recall: XOR's can be created from other logic gates.

-How do we add larger numbers?



Ripple-Carry 4-Bit Adder

-When adding 1111 to 0001 the carry takes a long time to propagate.
-Many adders have "Fast Carry" or "Look-Ahead Carry" to handle this.

Subtractors:

-Similar to adders but have a "borrow" bit rather than a "carry" bit.

Difference (D)

$JK\backslash B_i$	0	1
00	0	1
01	1	0
11	0	1
10	1	0

$D = J \oplus K \oplus B_i$

Adder: $Sum = X \oplus Y \oplus C_i$

Borrow (B_{i+1})

$JK\backslash B_i$	0	1
00	0	1
01	1	1
11	0	1
10	0	0

$B_{i+1} = \bar{J}K + \bar{J}B_i + KB_i$

Adder: $Carry = XY + XC_i + YC_i$

Letting $X = \bar{J}, Y = K, C_i = B_i$ and substituting into the adder equation:

$$Sum = \bar{J} \oplus K \oplus B_i \qquad Carry = \bar{J}K + \bar{J}B_i + KB_i = B_{i+1}$$

$$\overline{Sum} = J \oplus K \oplus B_i$$

$$\overline{Sum} = D$$

Aside: Proof that $\bar{Z} = X \oplus Y \leftrightarrow Z = \bar{X} \oplus Y$:

$$1: Z = \bar{X} \oplus Y$$

$$2: Z = \bar{X}\,\bar{Y} + XY$$

$$3: \bar{Z} = \overline{\bar{X}\,\bar{Y} + XY}$$

$$4: \bar{Z} = \overline{\bar{X}\,\bar{Y}} + \overline{XY}$$

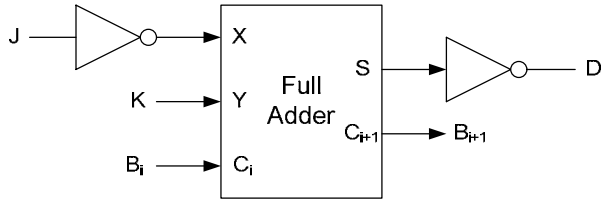
$$5: \bar{Z} = (X + Y)(\bar{X} + \bar{Y})$$

$$6: \bar{Z} = X\bar{X} + X\bar{Y} + \bar{X}Y + Y\bar{Y}$$

$$7: \bar{Z} = X\bar{Y} + \bar{X}Y$$

$$8: \bar{Z} = X \oplus Y$$

Therefore, we can use a full adder to create a full subtractor:

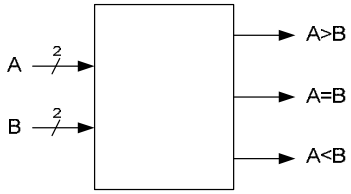


Full Subtractor

Magnitude Comparators:

-Input: Two binary numbers (A and B).

-Output: Three signals indicating A>B, A=B, and A<B.



2-Bit Magnitude Comparator

A_1	A_0	B_1	B_0	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

A>B

$A_1A_0 \backslash B_1B_0$	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

$$(A > B) = A_1\overline{B}_1 + A_0\overline{B}_1\overline{B}_0 + A_1A_0\overline{B}_0$$

A=B

$A_1A_0 \backslash B_1B_0$	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

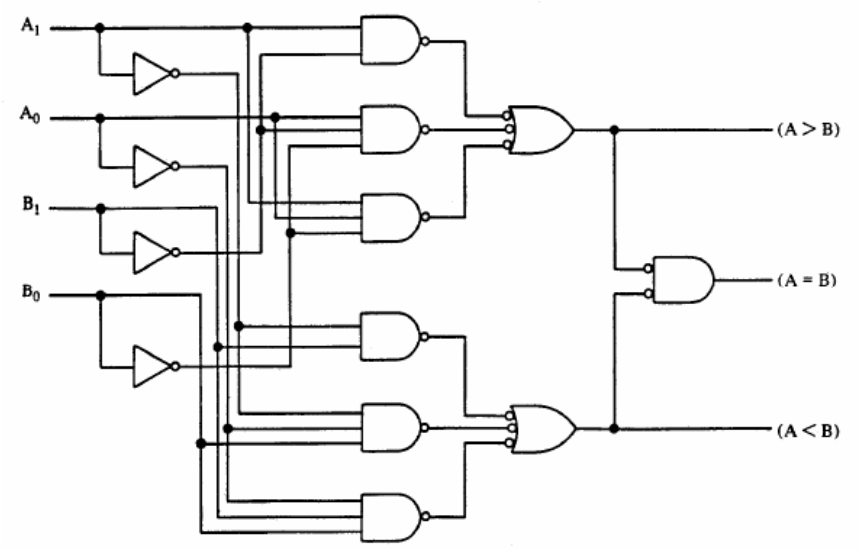
$$(A = B) = \overline{A}_1\overline{A}_0\overline{B}_1\overline{B}_0 + \overline{A}_1A_0\overline{B}_1B_0 + A_1\overline{A}_0B_1\overline{B}_0 + A_1A_0B_1B_0$$

Note: This equation is not reducible.

A<B

$A_1A_0 \backslash B_1B_0$	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

$$(A < B) = \overline{A}_1B_1 + \overline{A}_1\overline{A}_0B_0 + \overline{A}_0B_1B_0$$



2-Bit Magnitude Comparator

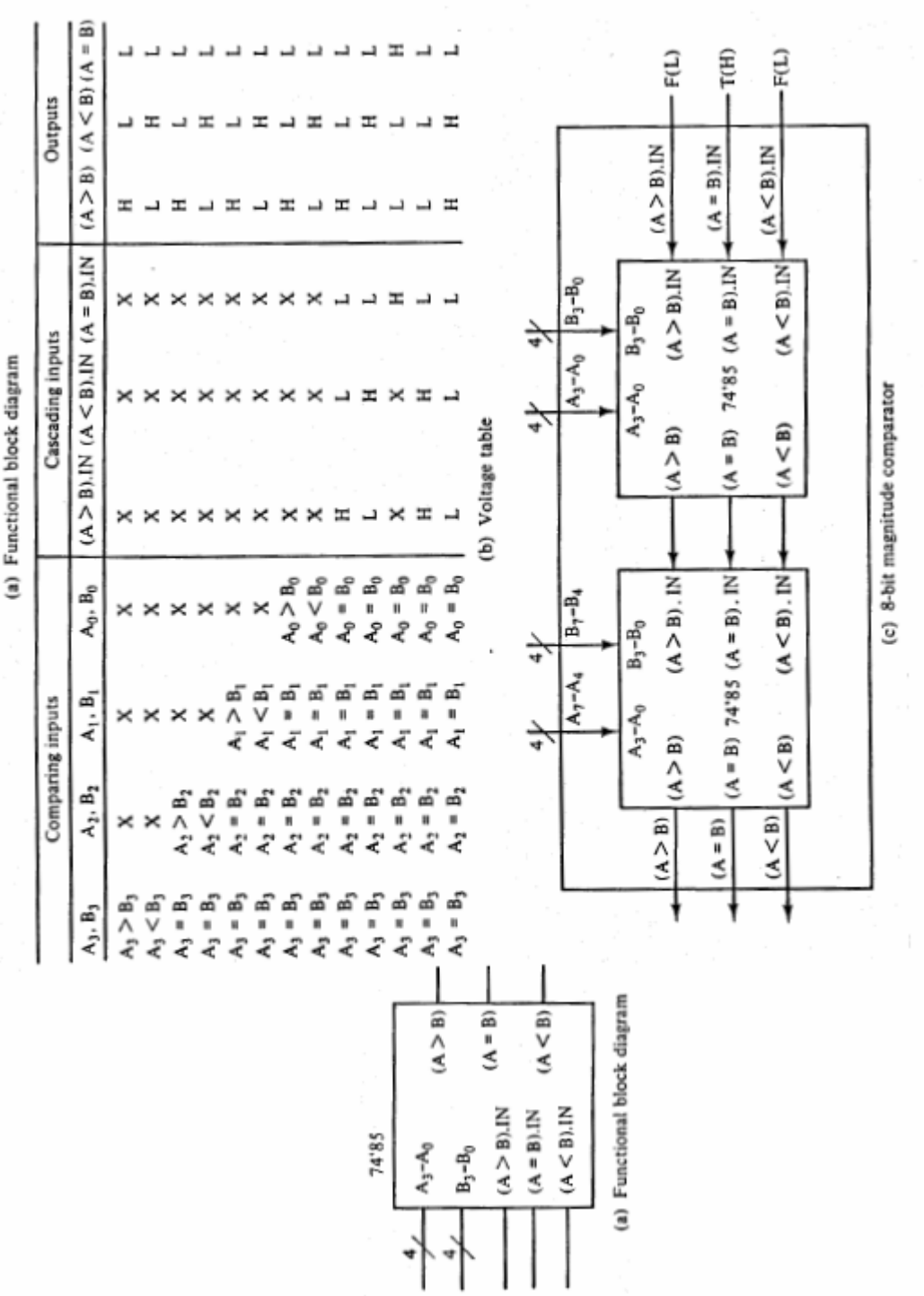
(Figure 4.7 from *Fundamentals of Computer Engineering: Logic Design and Microprocessors* by Lam, O'Malley, and Arroyo)

-Comparing 4-bit numbers:

-Requires a $2^4 \times 2^4 = 256$ square K-map

-Can use two 2-bit comparators and basic logic

-Cascading comparators exist so that you can easily expand comparator size



(a) Functional block diagram

Comparing inputs				Cascading inputs			Outputs		
A_3, B_3	A_2, B_2	A_1, B_1	A_0, B_0	$(A > B).IN$	$(A < B).IN$	$(A = B).IN$	$(A > B)$	$(A < B)$	$(A = B)$
$A_3 > B_3$	X	X	X	X	X	X	H	L	L
$A_3 < B_3$	X	X	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 > B_2$	X	X	X	X	X	H	L	L
$A_3 = B_3$	$A_2 < B_2$	X	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	X	X	X	X	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	X	X	X	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	H	L	L	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	H	L	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	X	X	H	L	L	H
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	H	H	L	L	L	H
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	L	L	H	H	L

(b) Voltage table

(c) 8-bit magnitude comparator

(a) Functional block diagram

4-Bit Cascading Magnitude Comparator (74’85)

(Figure 4.8 from *Fundamentals of Computer Engineering: Logic Design and Microprocessors* by Lam, O’Malley, and Arroyo)

-A 32-bit processor would require 8 of these

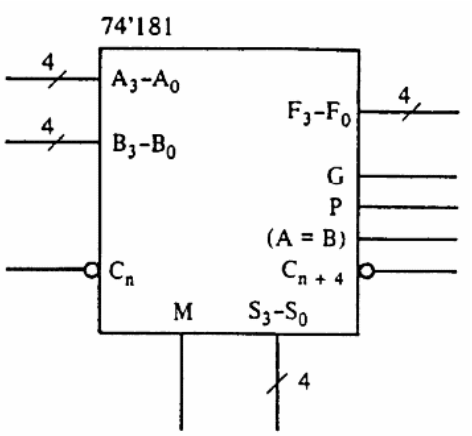
Arithmetic Logic Units (ALU's):

-ALU's are combinational logic circuits that perform basic calculations including:

1) Arithmetic operations (addition, subtraction, etc.)
2) Logic operations (OR, AND, complement, etc.).

-ALU's are LSI because they can perform many MSI fuctions.

-ALU's are at the core of CPU's (Central Processing Units)



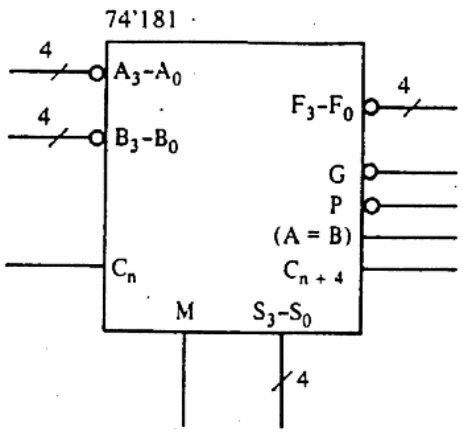
(a) Active-high view

Selection				Active-high data		
				M = H logic functions	M = L; Arithmetic operations	
S3	S2	S1	S0		C_n = H (no carry)	C_n = L (with carry)
L	L	L	L	F = $\overline{A}$	F = A	F = A PLUS 1
L	L	L	H	F = $\overline{A + B}$	F = A + B	F = (A + B) PLUS 1
L	L	H	L	F = $\overline{A}$B	F = A + $\overline{B}$	F = (A + $\overline{B}$) PLUS 1
L	L	H	H	F = 0	F = MINUS 1 (2's COMP)	F = ZERO
L	H	L	L	F = $\overline{AB}$	F = A PLUS A$\overline{B}$	F = A PLUS A$\overline{B}$ PLUS 1
L	H	L	H	F = $\overline{B}$	F = (A + B) PLUS A$\overline{B}$	F = (A + B) PLUS A$\overline{B}$ PLUS 1
L	H	H	L	F = A⊕B	F = A MINUS B MINUS 1	F = A MINUS B
L	H	H	H	F = A$\overline{B}$	F = A$\overline{B}$ MINUS 1	F = A$\overline{B}$
H	L	L	L	F = $\overline{A}$ + B	F = A PLUS AB	F = A PLUS AB PLUS 1
H	L	L	H	F = $\overline{A⊕B}$	F = A PLUS B	F = A PLUS B PLUS 1
H	L	H	L	F = B	F = (A + $\overline{B}$) PLUS AB	F = (A + $\overline{B}$) PLUS AB PLUS 1
H	L	H	H	F = AB	F = AB MINUS 1	F = AB
H	H	L	L	F = 1	F = A PLUS A	F = A PLUS A PLUS 1
H	H	L	H	F = A + $\overline{B}$	F = (A + B) PLUS A	A = (A + B) PLUS A PLUS 1
H	H	H	L	F = A + B	F = (A + $\overline{B}$) PLUS A	F = (A + $\overline{B}$) PLUS A PLUS 1
H	H	H	H	F = A	F = A MINUS 1	F = A

74'181 Arithmetic Logic Unit

(Figure 6.1a, b from *Fundamentals of Computer Engineering: Logic Design and Microprocessors* by Lam, O'Malley, and Arroyo)

Note: Some of these functions may not be very useful. They just happen to be the resulting output of the ALU. Perhaps some of these outputs were treated as "Don't Cares" during design and implementation of this circuit.



(c) Active-low view

Selection				Active-low data		
				M = H logic functions	M = L; Arithmetic operations	
S3	S2	S1	S0		C_n = L (no carry)	C_n = H (with carry)
L	L	L	L	F = $\overline{A}$	F = A MINUS 1	F = A
L	L	L	H	F = $\overline{AB}$	F = AB MINUS 1	F = AB
L	L	H	L	F = $\overline{A}$ + B	F = A$\overline{B}$ MINUS 1	F = A$\overline{B}$
L	L	H	H	F = 1	F = MINUS 1 (2's COMP)	F = ZERO
L	H	L	L	F = $\overline{A + B}$	F = A PLUS (A + $\overline{B}$)	F = A PLUS (A + $\overline{B}$) PLUS 1
L	H	L	H	F = B	F = AB PLUS (A + $\overline{B}$)	F = AB PLUS (A + $\overline{B}$) PLUS 1
L	H	H	L	F = $\overline{A \oplus B}$	F = A MINUS B MINUS 1	F = A MINUS B
L	H	H	H	F = A + $\overline{B}$	F = A + $\overline{B}$	F = (A + $\overline{B}$) PLUS 1
H	L	L	L	F = $\overline{A}$B	F = A PLUS (A + B)	F = A PLUS (A + B) PLUS 1
H	L	L	H	F = A $\oplus$ B	F = A PLUS B	F = A PLUS B PLUS 1
H	L	H	L	F = B	F = A$\overline{B}$ PLUS (A + B)	F = A$\overline{B}$ PLUS (A + B) PLUS 1
H	L	H	H	F = A + B	F = (A + B)	F = (A + B) PLUS 1
H	H	L	L	F = 0	F = A PLUS A	F = A PLUS A PLUS 1
H	H	L	H	F = A$\overline{B}$	F = AB PLUS A	F = AB PLUS A PLUS 1
H	H	H	L	F = AB	F = A$\overline{B}$ PLUS A	F = A$\overline{B}$ PLUS A PLUS 1
H	H	H	H	F = A	F = A	F = A PLUS 1

74'181 Arithmetic Logic Unit

(Figure 6.1c, d from *Fundamentals of Computer Engineering: Logic Design and Microprocessors* by Lam, O'Malley, and Arroyo)

If the ALU receives an instruction to complement A, the system must:

1) Connect register A to the correct ALU input
2) Send the correct control signals to the ALU
   (i.e. M = H, S_3 = L, S_2 = L, S_1 = L, S_0 = L)
3) Delay long enough for ALU to process the result properly
4) Load the ALU output into register A