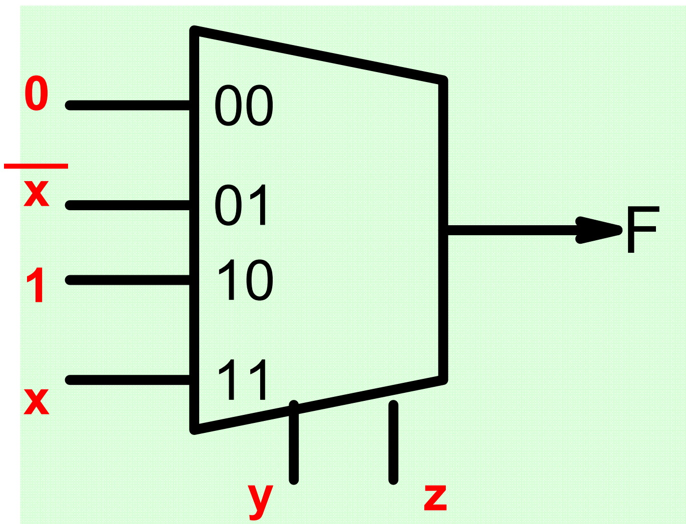
$$F(x,y,z)=\sum(1,2,6,7)$$

A 3 variable function can be implemented with a 4:1 mux with 2 select lines



x	y	z	F
0	0	0	0
1	0	0	0
0	0	1	1
1	0	1	0
0	1	0	1
1	1	0	1
0	1	1	0
1	1	1	1

F = 0 when yz = 00

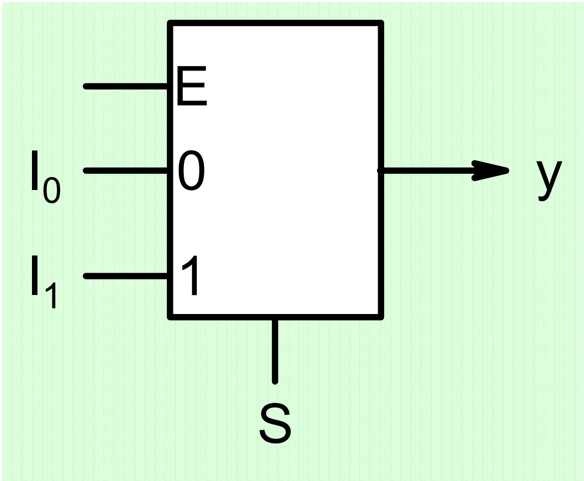
F = $\overline{x}$ when yz =01

F = 1 when yz = 10

F = x when yz = 11

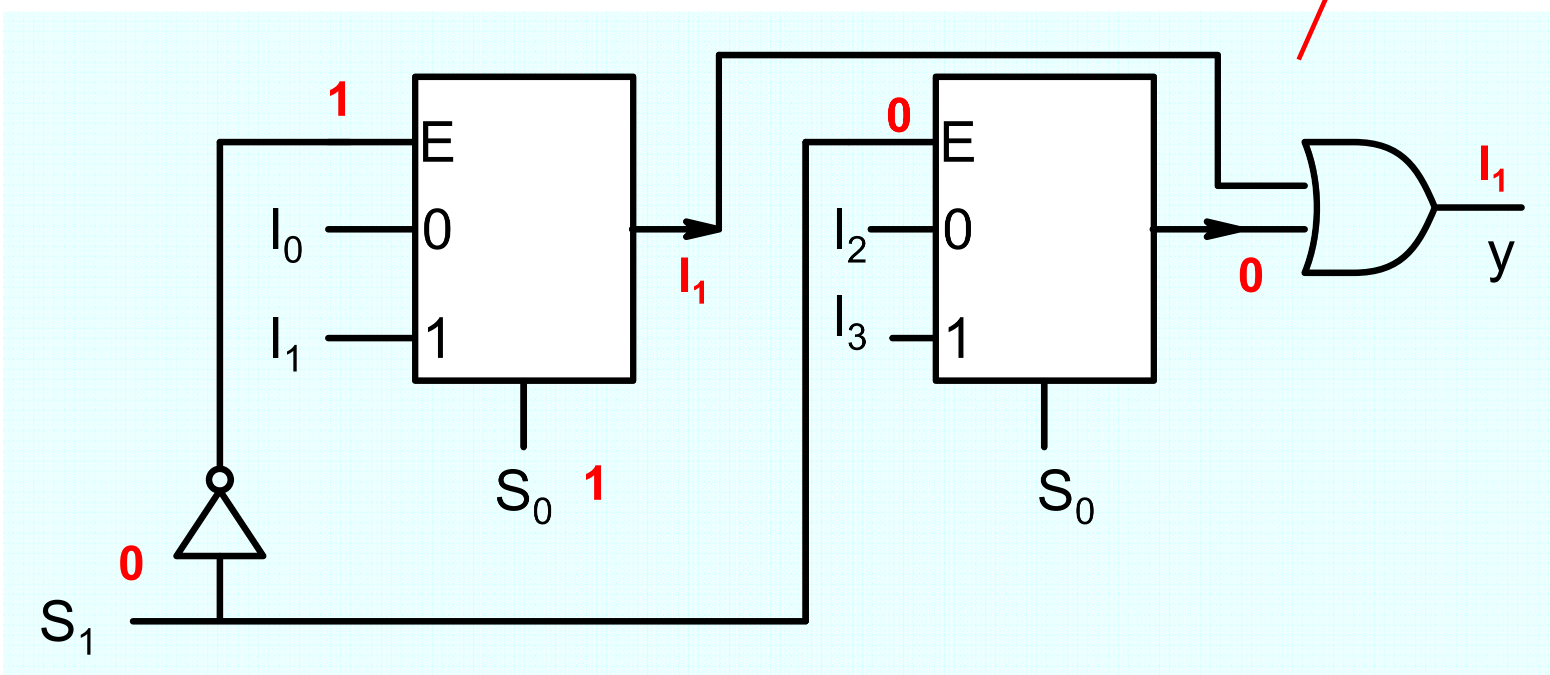
Mux is more efficient way of implementing combinational circuits as compared to decoders.

Mux. expansion

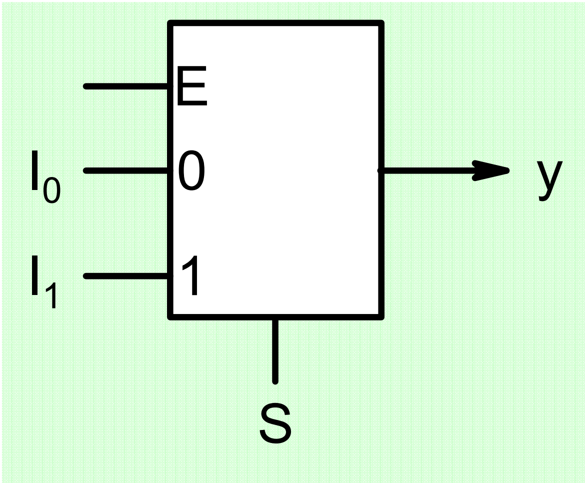


E	S	y
0	x	0
1	0	I_0
1	1	I_1

S_1	S_0	y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

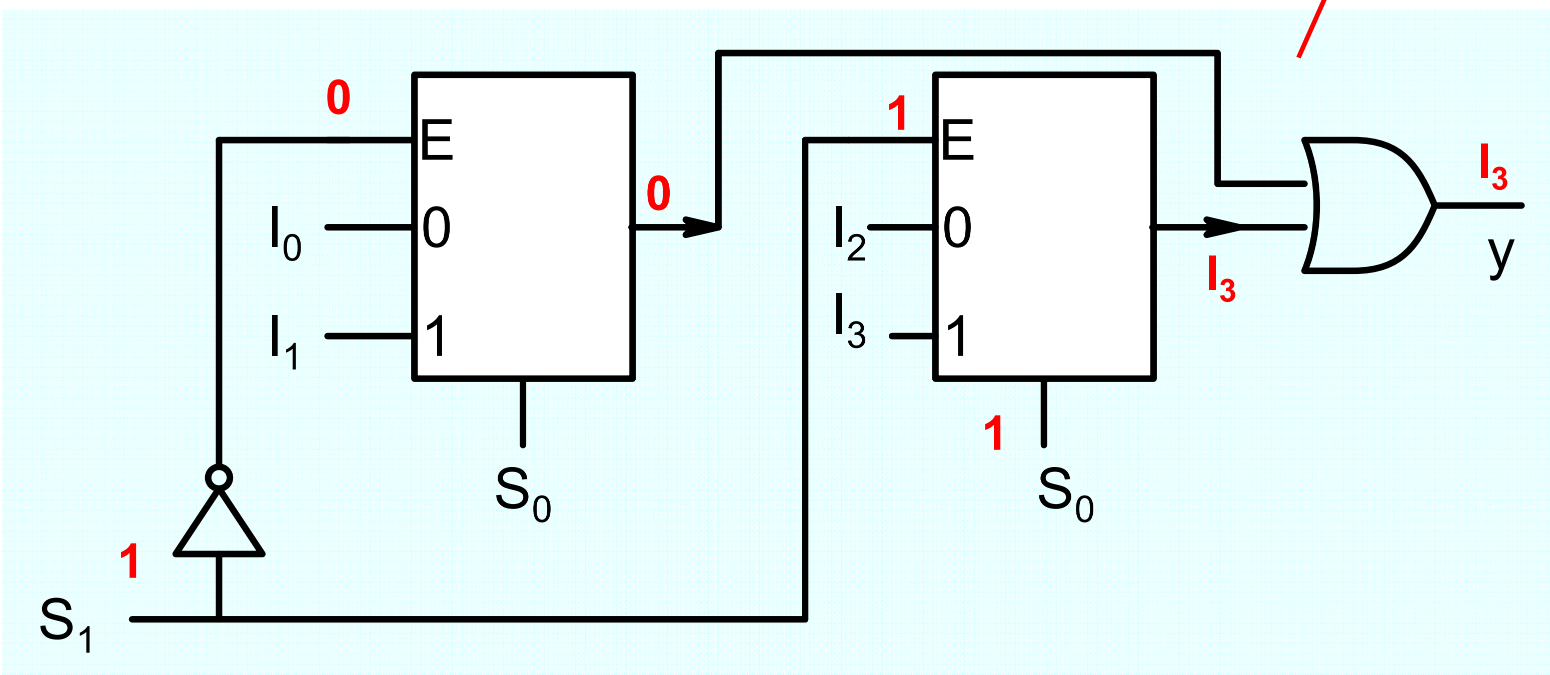


Mux. expansion

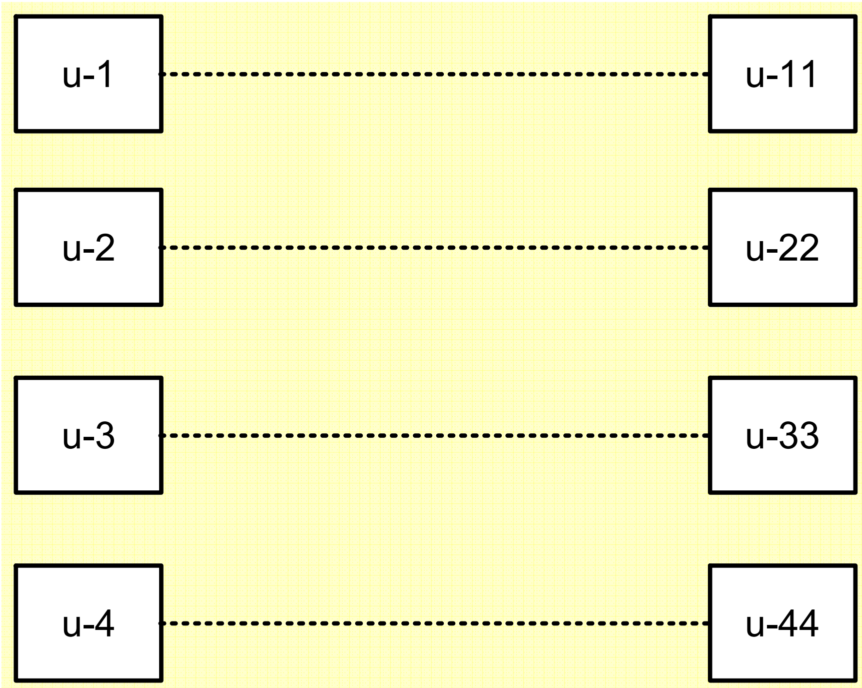


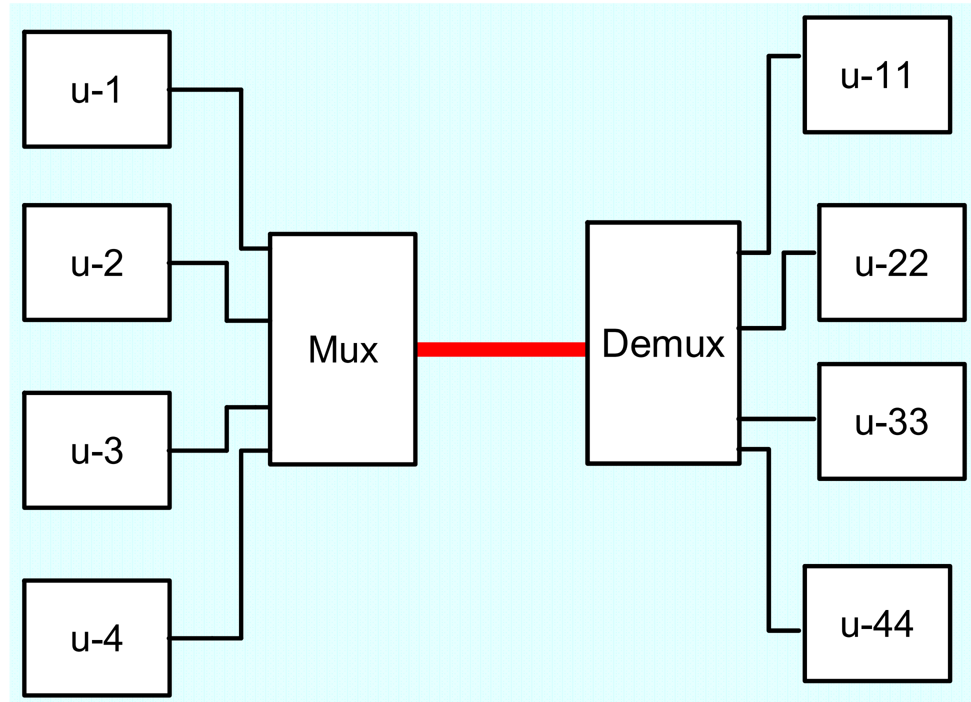
E	S	y
0	x	0
1	0	I_0
1	1	I_1

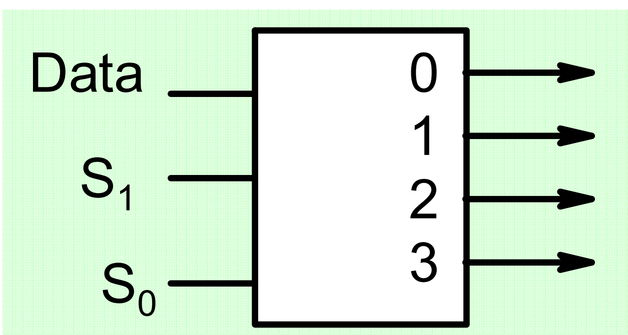
S_1	S_0	y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



DeMultiplexer

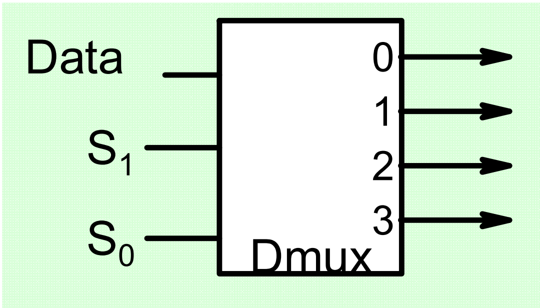


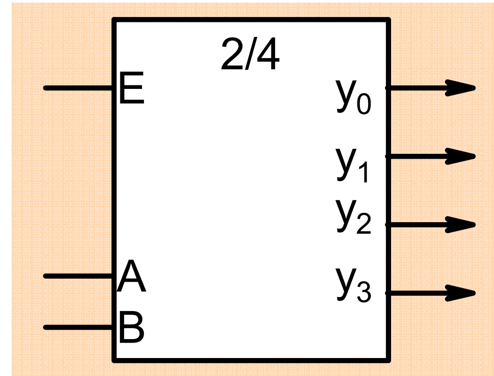




S_1	S_0	y_0	y_1	y_2	y_3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

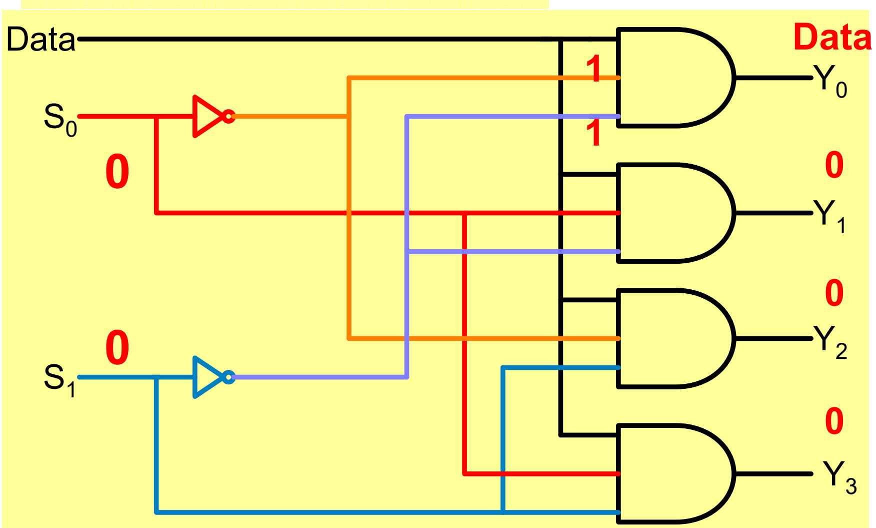
Demultiplexer is very much like a decoder

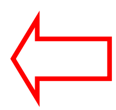


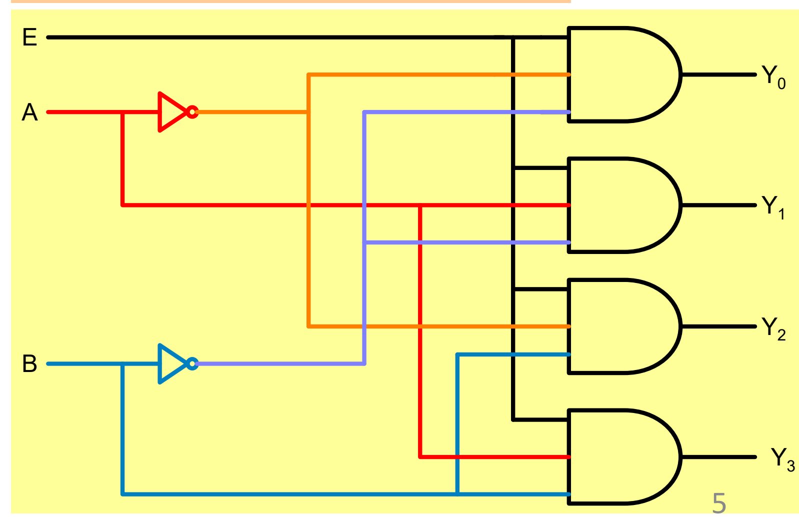


S_1	S_0	y_0	y_1	y_2	y_3
0	0	I_0	0	0	0
0	1	0	I_1	0	0
1	0	0	0	I_2	0
1	1	0	0	0	I_3

E	B	A	Y_0	Y_1	Y_2	Y_3
0	x	x	0	0	0	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	0	1







Comparator

$$A = A_3 A_2 A_1 A_0$$

$$x_i = A_i . B_i + \overline{A_i} . \overline{B_i} \qquad \text{for i} = 0,1,2,3$$

$$B = B_3 B_2 B_1 B_0$$

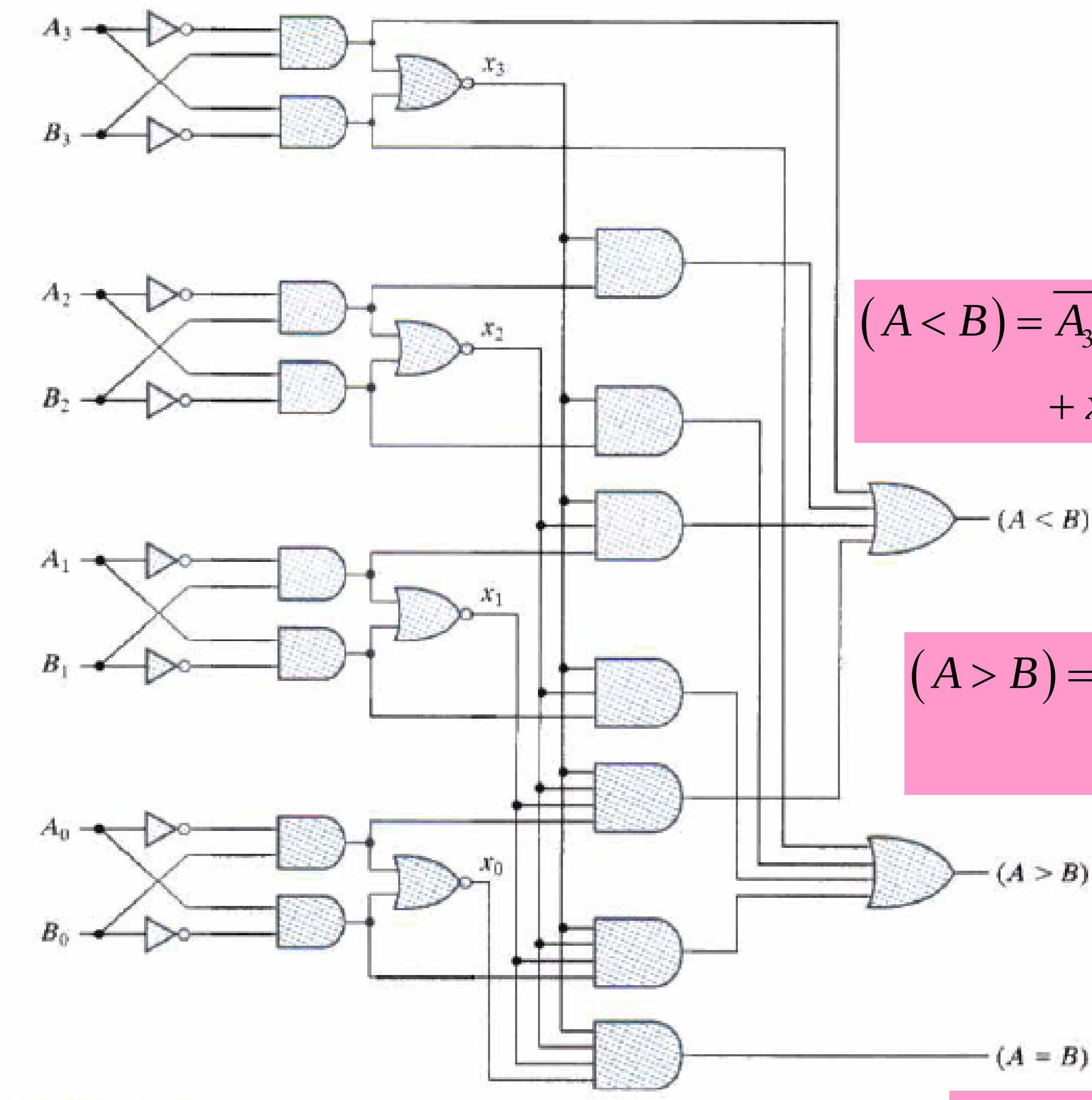
where x_i = 1 only if the pair of bits in position i are equal (i.e., if both are 1 or both are: 0).

$$(A = B) = x_3 x_2 x_1 x_0$$

all x_i *variable must be equal to 1*

$$(A > B) = A_3 \overline{B_3} + x_3 A_2 \overline{B_2} + x_3 x_2 A_1 \overline{B_1} + x_3 x_2 x_1 A_0 \overline{B_0}$$

$$(A < B) = \overline{A_3} B_3 + x_3 \overline{A_2} B_2 + x_3 x_2 \overline{A_1} B_1 + x_3 x_2 x_1 \overline{A_0} B_0$$

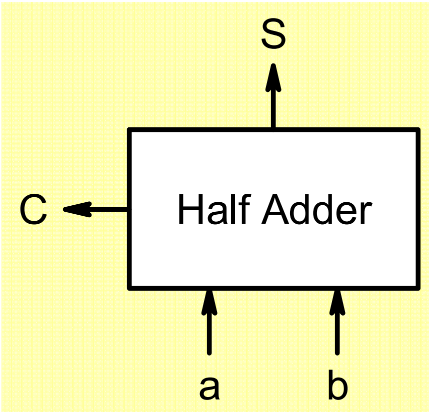


$$(A < B) = \overline{A_3}B_3 + x_3\overline{A_2}B_2 + x_3x_2\overline{A_1}B_1 + x_3x_2x_1\overline{A_0}B_0$$

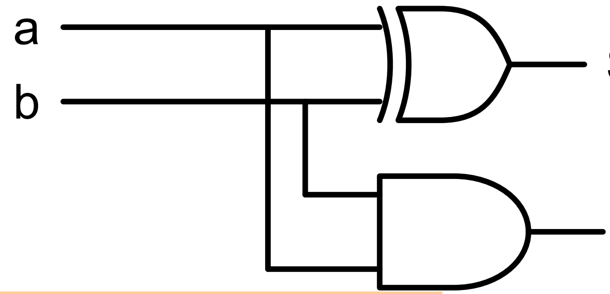
$$(A > B) = A_3\overline{B_3} + x_3A_2\overline{B_2} + x_3x_2A_1\overline{B_1} + x_3x_2x_1A_0\overline{B_0}$$

$$(A = B) = x_3x_2x_1x_0$$

Adder



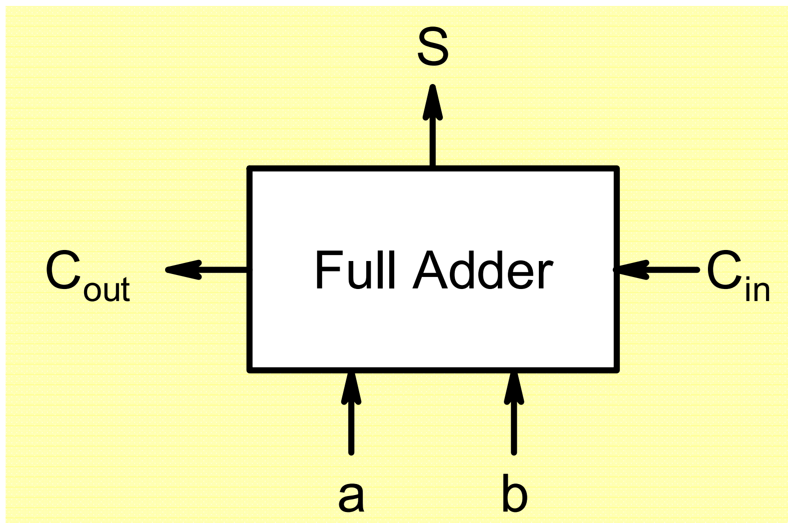
a	b	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



$$S = \overline{a}.b + a.\overline{b}\,; C = a.b$$

a	b	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

```
 1
 1 1 1
 1 1 0
---------
1 1 0 1
```



$$S = \overline{a}.\overline{b}.c_{in} + \overline{a}.b.\overline{c_{in}} + a.\overline{b}.\overline{c_{in}} + a.b.c_{in}\,;$$

$$C_{out} = \overline{a}.b.c_{in} + a.\overline{b}.c_{in} + a.b.\overline{c_{in}} + a.b.c_{in}$$

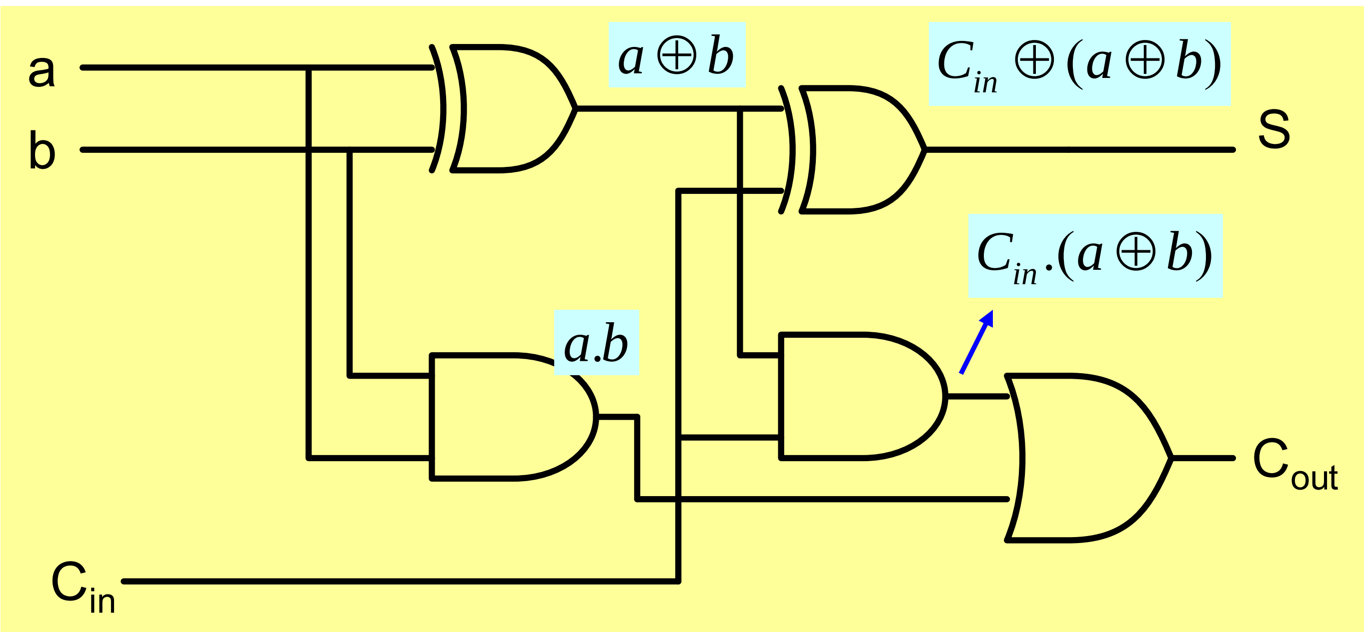
$$S = \bar{a}.\bar{b}.c_{in} + \bar{a}.b.\overline{c_{in}} + a.\bar{b}.\overline{c_{in}} + a.b.c_{in}$$

$$S = C_{in} \oplus (a \oplus b)$$

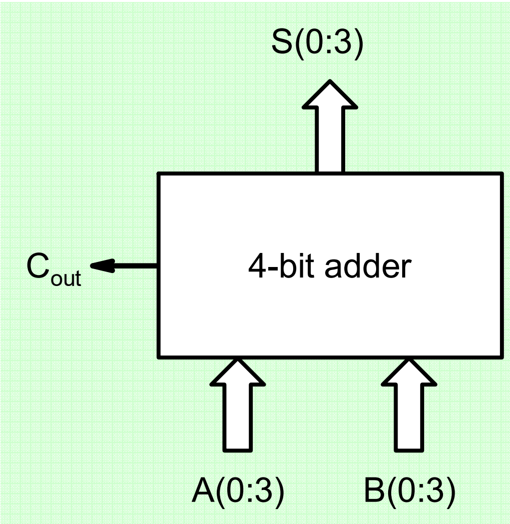
$$C_{out} = \bar{a}.b.C_{in} + a.\bar{b}.C_{in} + a.b.\overline{C_{in}} + a.b.C_{in}$$

$$C_{out} = C_{in}(a.\bar{b} + \bar{a}.b) + a.b = C_{in}.(a \oplus b) + a.b$$

a	b	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

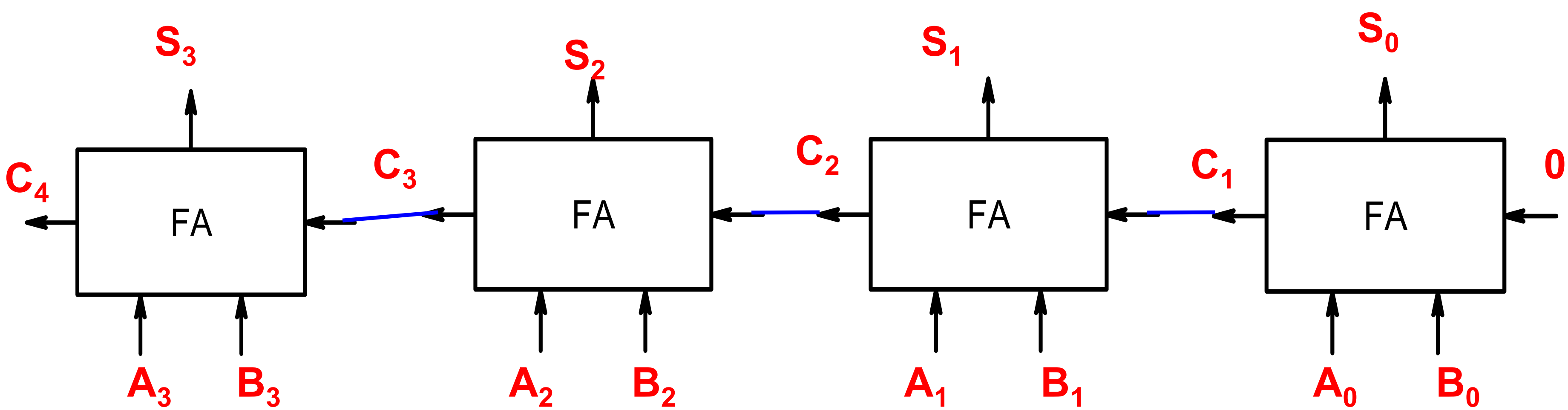


4-bit Adder



$A_3A_2A_1A_0$	$B_3B_2B_1B_0$	$S_3S_2S_1S_0$	C_{out}
0000	0000	0000	1
0000	0001	0001	0
0001	0000	0001	0
⋮	⋮	⋮	⋮

$$
\begin{array}{ccccc}
 & C_3 & C_2 & C_1 & \\
 & A_3 & A_2 & A_1 & A_0 \\
 & B_3 & B_2 & B_1 & B_0 \\
\hline
C_4 & S_3 & S_2 & S_1 & S_0 \\
\hline
\end{array}
$$

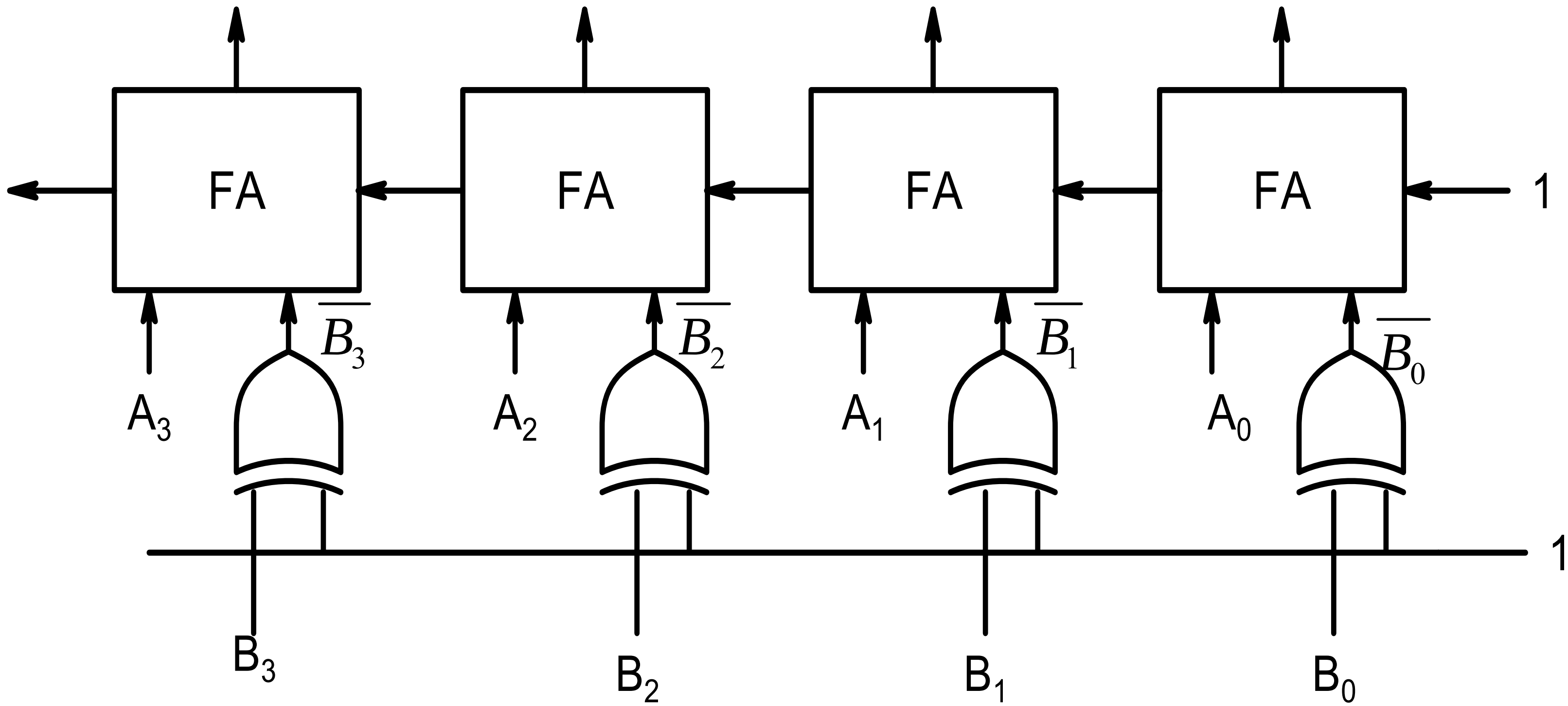


Subtraction

A - B = A + 2's complement of B

A - B = A + 1's complement of B+1

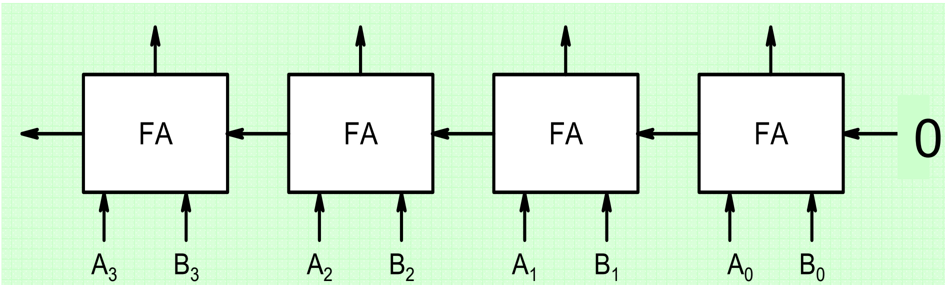
$$A - B = A + \overline{B} + 1$$

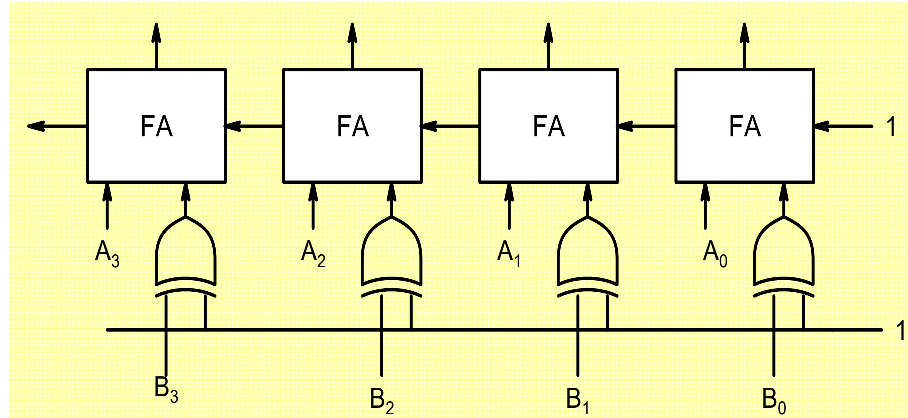


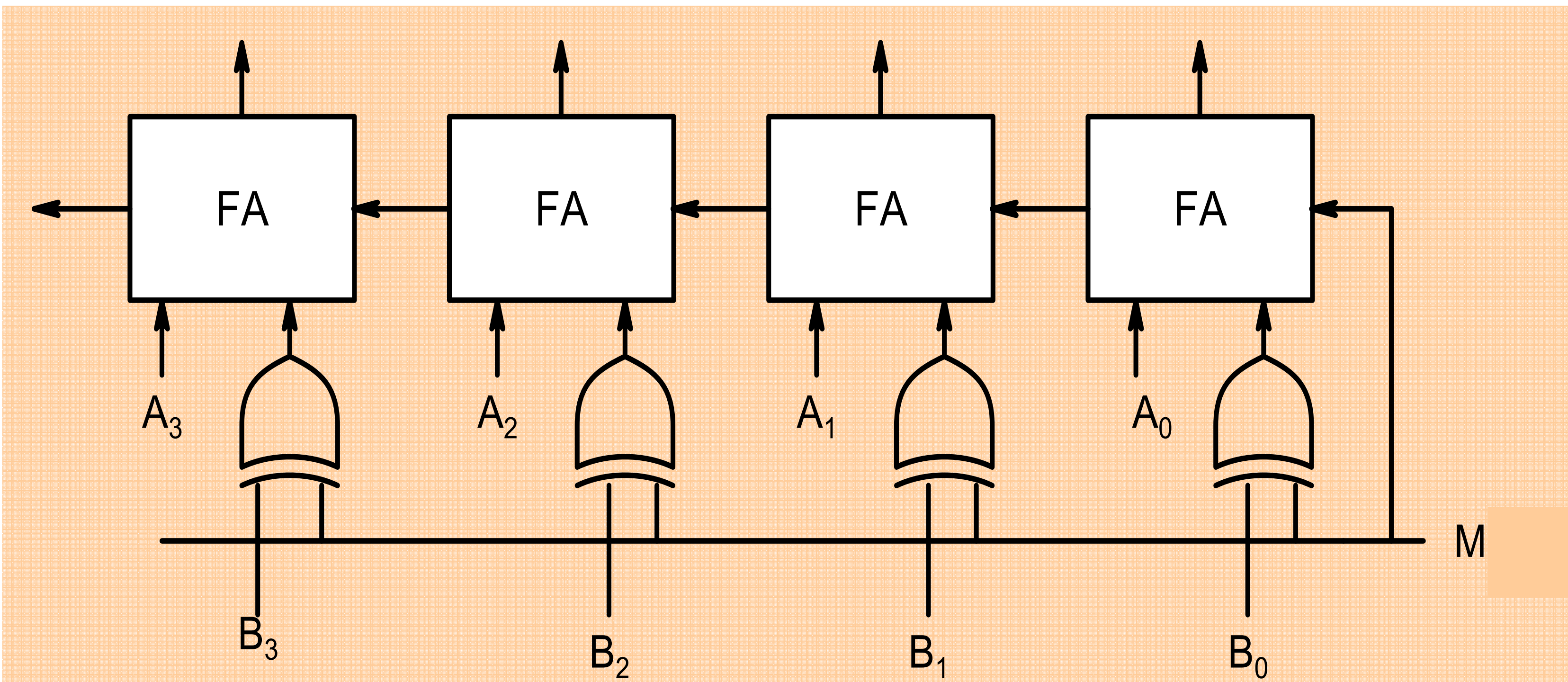
$$B_0 \oplus 1 = B_0.\overline{1} + \overline{B_0}.1 = \overline{B_0}$$

One needs add a circuit for predicting errors resulting from overflow

Adder/Subtractor







$$B_0 \oplus 0 = B_0.\overline{0} + \overline{B_0}.0 = B_0$$

$$B_0 \oplus 1 = B_0.\overline{1} + \overline{B_0}.1 = \overline{B_0}$$

$M = 0$ for Adder

M=1 for Subtractor