H. W. --- Will discuss in the next class if you are not able to solve it.

A student staying in a hostel has to make up his mind about his dinner. If he has enough money (*M*) and at least three of his friends (*F*) also agree to go out for dinner, and it is not raining (R), he will have dinner with his friends in a restaurant outside the campus. If he is not able to go out, but at least three of his friends agree to join him (*J*), if the kind of food he wanted is available on home delivery (*K*) he will order home delivery of food from outside. But if the general feeling is that the food in the hostel mess is good on that day (*G*), he will have his dinner in the hostel mess. Let his decision be denoted by a 2-bit output D_1D_0:

R=1 not railing

$D_1 = 1$ home delivery

$D_0 = 1$ go out

$$D_0 = M \cdot F \cdot R \cdot \bar{G} \qquad D_1 = (\bar{M} + \bar{F} + \bar{R}) \cdot J \cdot K \cdot \bar{G}$$

$D_1 = 1$

$D_0 = 1$

$D_0 = D_1 = 1$ never happen

D_1D_0 = 00 ⇒ He eats in the hostel mess,

D_1D_0 = 01 ⇒ He goes out to have dinner in a restaurant, and

D_1D_0 = 10 ⇒ He orders food through home delivery.

practice problems

statement to logic

not there in Mano.

Binary Addition:

$X = x_4 x_3\ x_2\ x_1\ x_0 = 01111$

$Y = y_4 y_3\ y_2\ y_1\ y_0 = 01010$

Find:

Sum $S = \underline{s_4 s_3\ s_2\ s_1\ s_0}$ and
1 1 0 0 0

Carry $C = c_5, c_4, c_3, c_2$ and c_1
0 1 1 1 0

in hardware

carry forward

C	0 1 1 1 0
X	$c_5 \leftarrow$ 01111
Y	+01010
S	11001

$s_4 s_3 s_2 s_1 s_0$

$x_0 + y_0 \rightarrow c_i$

$1 \times 2^0 + 1 \times 2^0$

$= 2 \times 2^0$

$\rightarrow = 2^1$

$1 \times 2^1 + 0 \times 2^0$

HA (single bit) and FA (multi-bit)

$$c\oplus(x\oplus y) = (c\oplus x)\oplus y = c\oplus x\oplus y$$

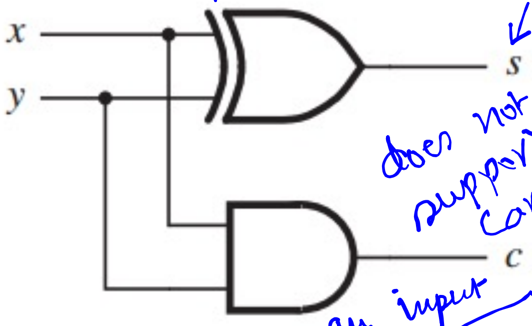
2 variables

$c_4\ c_3\ c_2\ c_1$?
$x_4\ x_3\ x_2\ x_1\ x_0$
$y_4\ y_3\ y_2\ y_1\ y_0$

x	y	Carry c	Sum s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

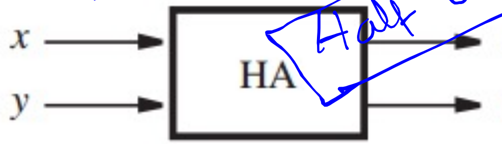
$x\oplus y$

$x\cdot y$



does not support carry input

no carry as an input



Half adder

Generated carries →	1 1 1 0		
$X = x_4x_3x_2x_1x_0$	0 1 1 1 1	$(15)_{10}$	
$+Y = y_4y_3y_2y_1y_0$	+ 0 1 0 1 0	$+(10)_{10}$	
$S = s_4s_3s_2s_1s_0$	1 1 0 0 1	$(25)_{10}$	

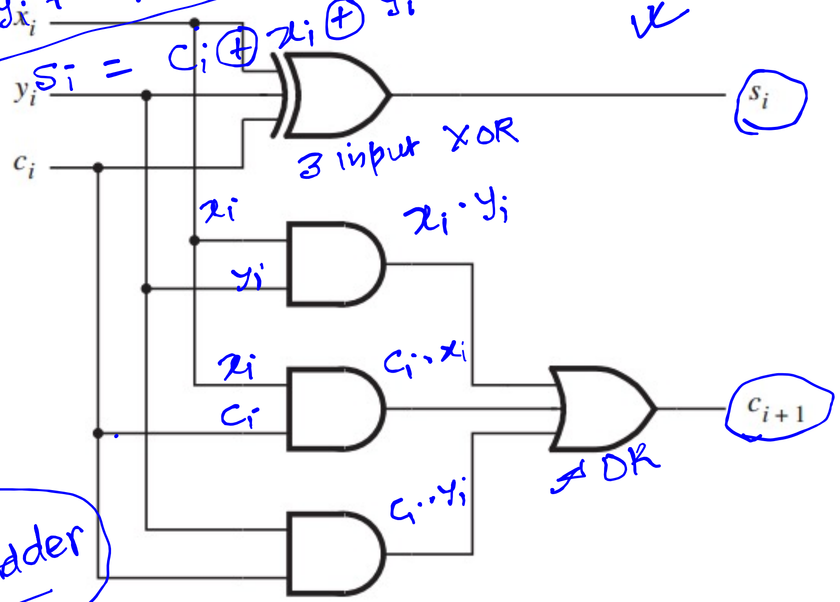
...	c_{i+1}	c_i	...		
...	...	x_i	...		
...	...	y_i	...		
...	...	s_i	...		

minimised SOP

$x_iy_i + x_ic_i + y_ic_i = c_{i+1}$

$s_i = c_i\oplus x_i\oplus y_i$

c_i	x_i	y_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Full Adder

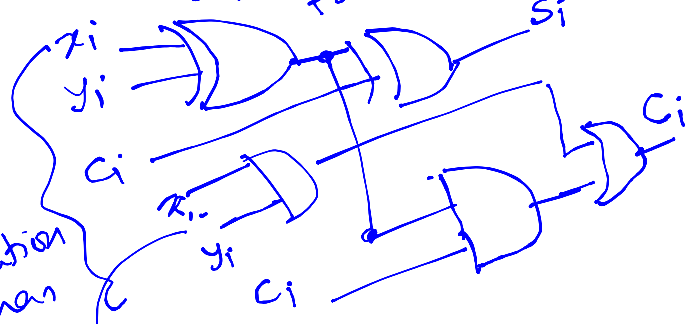
HA (single bit) and FA (multi-bit):

$$c_{i+1} = \bar{c}_i \cdot x_i \cdot y_i + c_i \cdot \bar{x}_i \cdot y_i + c_i \cdot x_i \cdot \bar{y}_i + c_i \cdot x_i \cdot y_i \longrightarrow \text{CSOP}$$

less power
faster

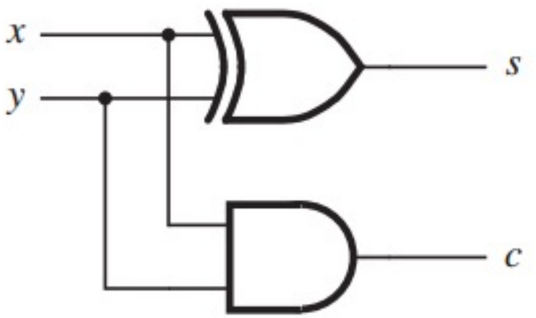
$$c_{i+1} = x_i \cdot y_i + c_i \cdot (\bar{x}_i \cdot y_i + x_i \cdot \bar{y}_i)$$

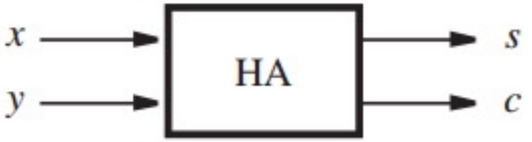
$$c_{i+1} = x_i \cdot y_i + c_i \cdot (x_i \oplus y_i)$$



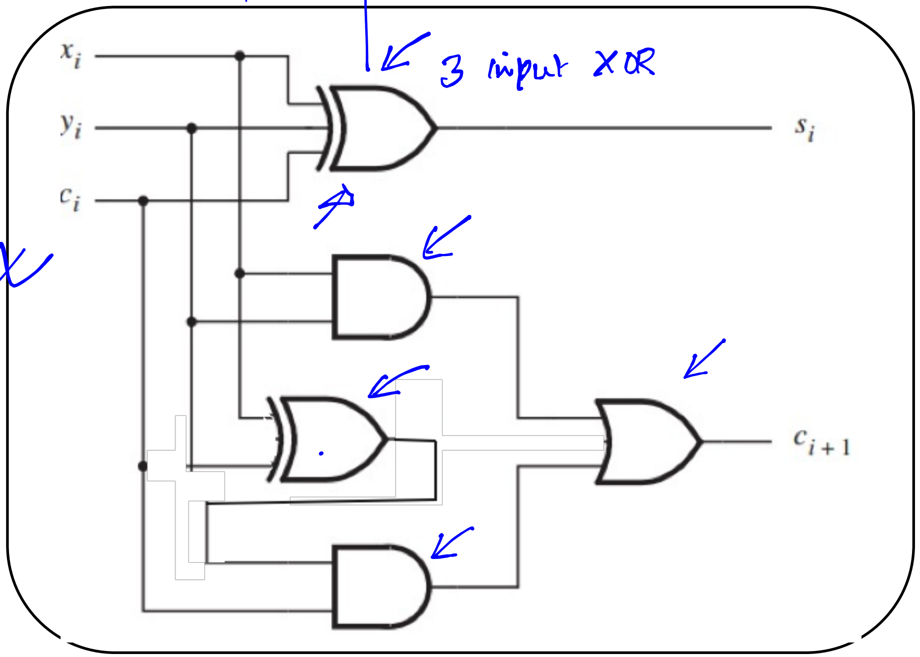
Better realization than

x	y	Carry c	Sum s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0





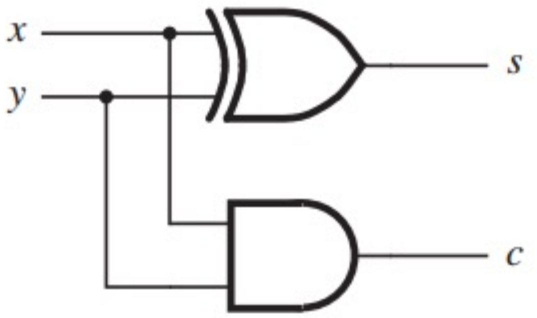
c_i	x_i	y_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

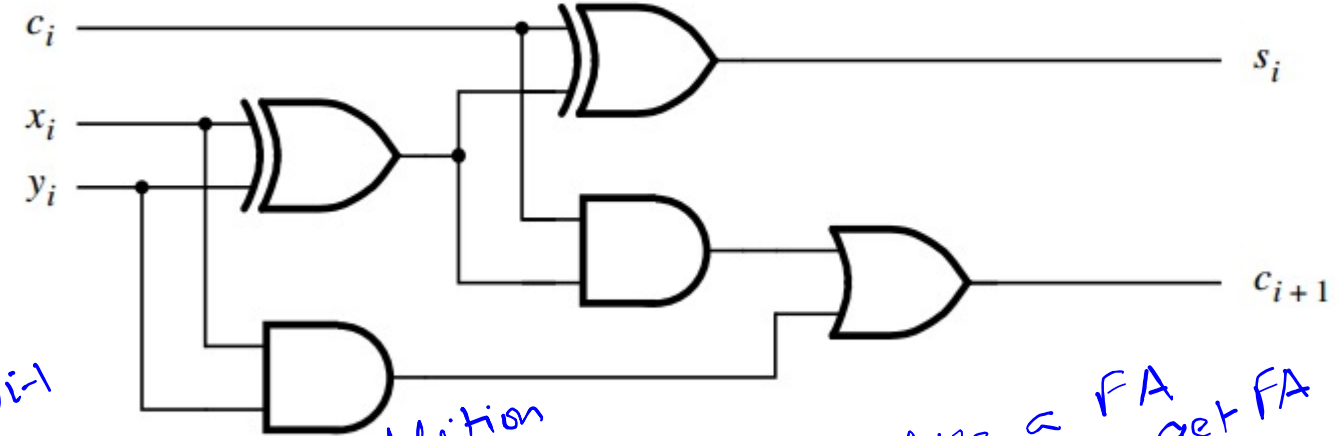


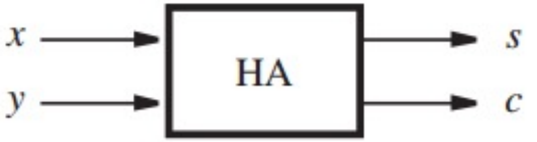
HA (single bit) and FA (multi-bit)

$$s_i = (x_i \oplus y_i \oplus c_i)$$

$$c_{i+1} = x_i \cdot y_i + c_i \cdot (x_i \oplus y_i)$$







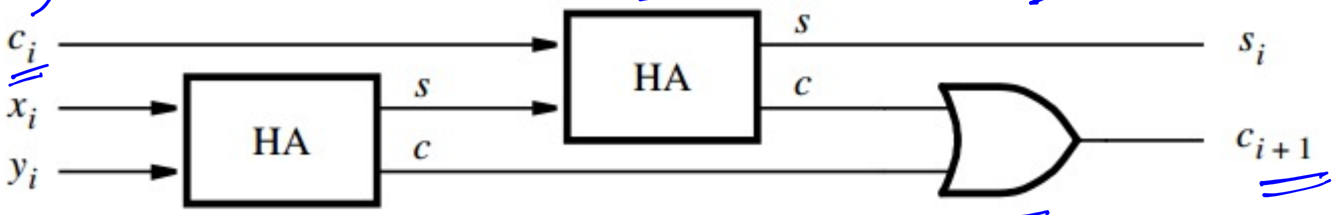
designed

$c_i = x_{i-1} \cdot y_{i-1}$

$x_{i-1} \oplus y_{i-1}$

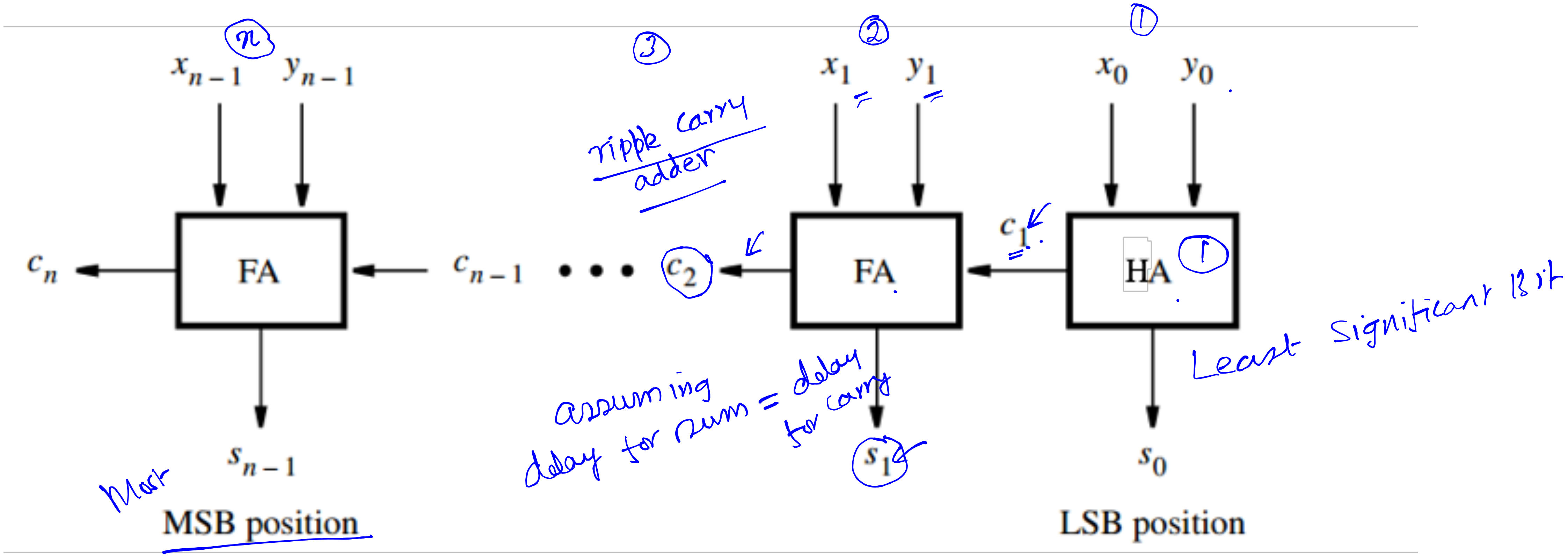
logic → addition

We realize a FA
use HA to get FA



This approach minimizes the number of ICs needed to implement the circuit, and it reduces the wiring complexity substantially.

4-bit FA Block diagram



This is called a Ripple Carry Adder. There are other architectures available for an adder and will be a topic in your Computer Organization course next sem.

H.W. Draw a timing diagram for a 3-bit full adder.

unable to draw

We will look at it in the next class, after you try it. If you try and find it difficult, we will discuss it, else we will spend a little time over what does the timing diagram tell us. If you are not keen on doing the H.W. we will not bother about it at all.

4 bit adder

$x_3 x_2 x_1 x_0$

$y_3 y_2 y_1 y_0$

3 bit adder

$x_2 x_1 x_0$

$y_2 y_1 y_0$

$W_1 W_0$

$x_1 x_0$

$y_1 y_0$

$z_1 z_0$

addition of 4 numbers

display solution in the next class

"length"

1 MHz → 1 nsec fast

1 μsec

1 GHz. slow

1 nsec.

Unsigned Numbers (Issues)

- Negative Numbers
- Subtraction when the result is negative
- Separate Circuits for addition and subtraction

Borrow 2^1+2^1

same ckt. but adjust input

Logic used

diff. bet. y and x

diff. x and y

$y - x$

$y+(-x)$

x	y	D_{yx}	B_{yx}
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	0

Borrow

Half Subtractor

Sign

$x - y$

$x+(-y)$

x	y	D_{xy}	B_{xy}
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Borrow

$D_{yx} = x \oplus y\ ;\ \ B_{yx} = x \cdot \bar{y}$

$D_{xy} = x \oplus y\ ;\ \ B_{xy} = \bar{x} \cdot y$

Signed Numbers: Addition/ Subtraction:

Decimal	Signed Magnitude
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000
−0	1000
−1	1001
−2	1010
−3	1011
−4	1100
−5	1101
−6	1110
−7	1111
−8	—

1. If both the numbers have the same sign, drop the sign bit, add the two numbers and reinsert the sign bit.
2. If one is positive and the other is negative, remove the sign bit, subtract the ~~larger~~ number from the ~~smaller~~ number and then reintroduce the sign of the number with larger magnitude. ---- A very laborious process, hence not preferred.

+4 familiar −4 negative representation

sign bit 0 → positive 1 → negative

8 sign

better representation

8 bit → 1 bit for sign, 7 bit number convention

4 bit → 1 sign bit + 3 bit number

signed numbers 01000 → 5 bits, 4 bits

(0) Compare sign, store sign bits; (1) add (2) reinsert sign

4 bit 0→7

32 bit 31

3 bit 1 1 1 2^2 2 2

exists. not preferred.

256 bit → 64 bit comp. $(2^{32}-1)$

Check smaller magnitude; larger; for magnitude

Computer organization 13423700 800 compare

(0) store accuracy sign bits (1) Compare magnitude (2) subtract (3) Reinsert sign bit

1342×10^{9999} 1342 9999 zeros map it to binary 100000000 9999999 13423789

2^{31} 66 32 16 15 power 2^{4} bigger

13421 13422 13423 13420

6 (2 bits) 4 range $(2^8)^4$ $(256)^4$ approximation