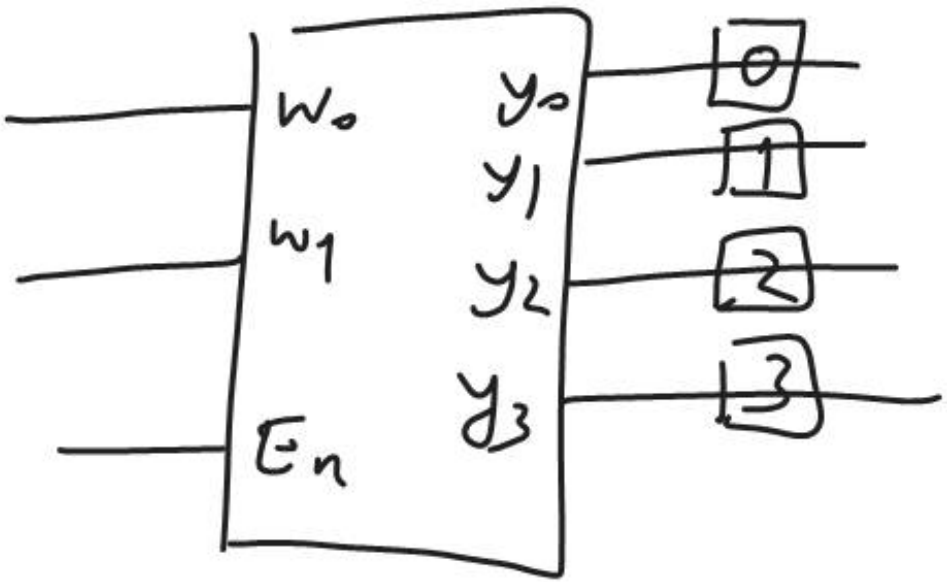
ex a 2-to-4 binary decoder 03.03.2011

(C)



E_n	w_1	w_0	y_0	y_1	y_2	y_3
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
0	X	X	0	0	0	0

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY DEC2TO4 IS
PORT( W: IN STD_LOGIC_VECTOR (1 DOWNTO 0);
      En: IN STD_LOGIC;
```

```
    Y : OUT STD_LOGIC_VECTOR(0 TO 3));
END DEC2TO4;
ARCHITECTURE Behavior OF DEC2TO4 IS
SIGNAL EnW : STD_LOGIC_VECTOR (2 DOWNTO 0);
BEGIN
EnW <= En & W;        -- Concatenation
WITH EnW SELECT
   Y <= "1000" WHEN "100",
        "0100" WHEN "101",
        "0010" WITH "110",
        "0001" WITH "111",
        "0000" WITH OTHERS;
END Behavior;
```

En W1 W0

ARITHMETIC CIRCUITS

$+$ → PLUS (Arithmetic)

$+$ → OR (Logic)

$\cdot$ → X (Arithmetic)

$\cdot$ → AND (Logic)

—— ⇒ subtraction (arithmetic)

INTEGERS → UNSIGNED

INTEGERS → SIGNED

$D = d_{n-1}\, d_{n-2} \cdots d_1\, d_0$ ⇒ digits - decimal

326 → a - 3 digit decimal number.

$V(D)$ = value of the decimal number:

$$d_{n-1} \times 10^{n-1} + d_{n-2} \times 10^{n-2} + \cdots + d_1 \times 10^1 + d_0 \times 10^0$$

d_{n-1}: Digit; 10^{n-1}: base-10 (radix 10)

A binary number $= b_{n-1} \cdot b_{n-2} \cdots b_1 b_0$

$$V(B) = \sum_{i=0}^{n-1} b_i \cdot 2^i = b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \cdots + b_1 \cdot 2 + b_0 \cdot 2^0$$

5 bits, $n = 5$

ex $(1\,0\,0\,1\,1)_2 \Longrightarrow V = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2 + 1 \times 2^0$

$V = 2^4 + 2 + 1 = 19_{10}$

$b_4\ b_3\ b_2\ b_1\ b_0$

ADDITION OF UNSIGNED NUMBERS

$$\begin{array}{cc} & x \\ + & y \\ \hline c & s \end{array}$$

c: Carry

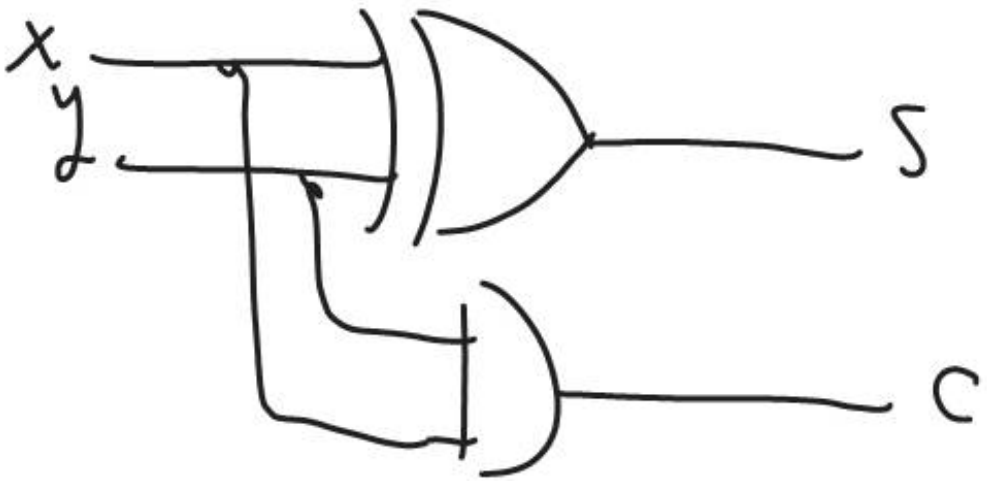
s: Sum

x	y	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$1+1=10$

HALF ADDER

$S = x \oplus y \qquad C = x \cdot y$



FULL ADDER

carry in → C_i | carry out → C_{i+1} | sum → S_i

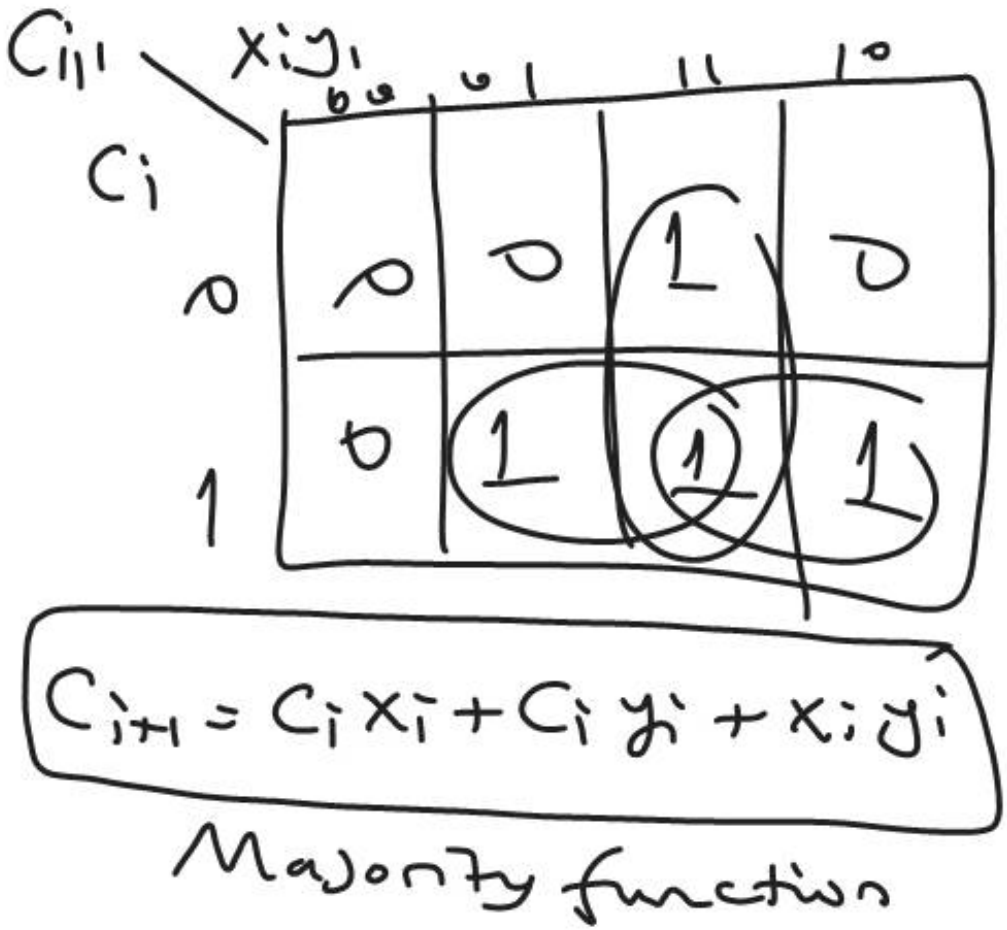
C_i	x_i	y_i	C_{i+1}	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

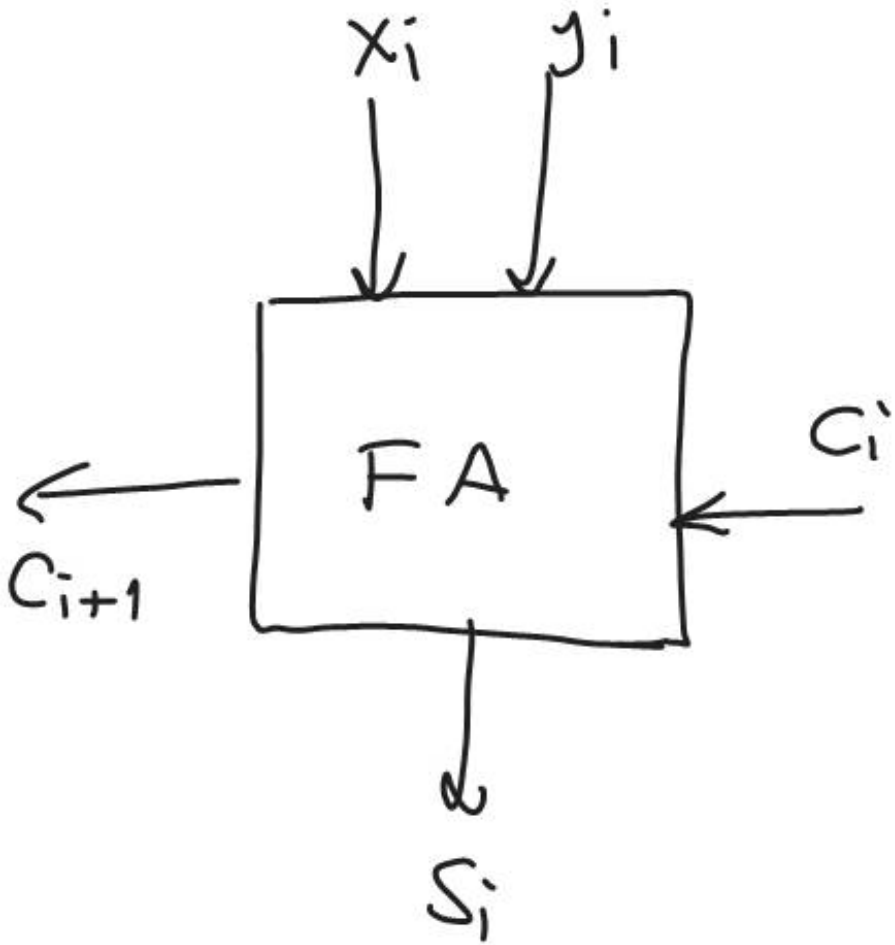
S_i \ x_iy_i	00	01	11	10
C_i = 0	0	1	0	1
C_i = 1	1	0	1	0

$$S_i = x_i \oplus y_i \oplus C_i$$

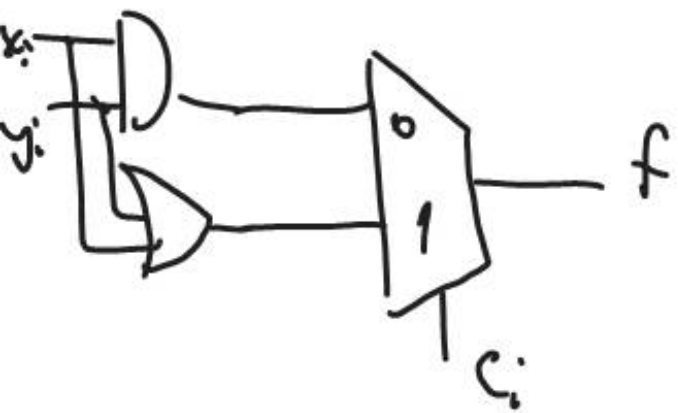
$$= (x_i \oplus y_i) \oplus C_i$$

$$= (x_i \oplus y_i)\, C_i + (x_i y_i)\, \overline{C_i}$$

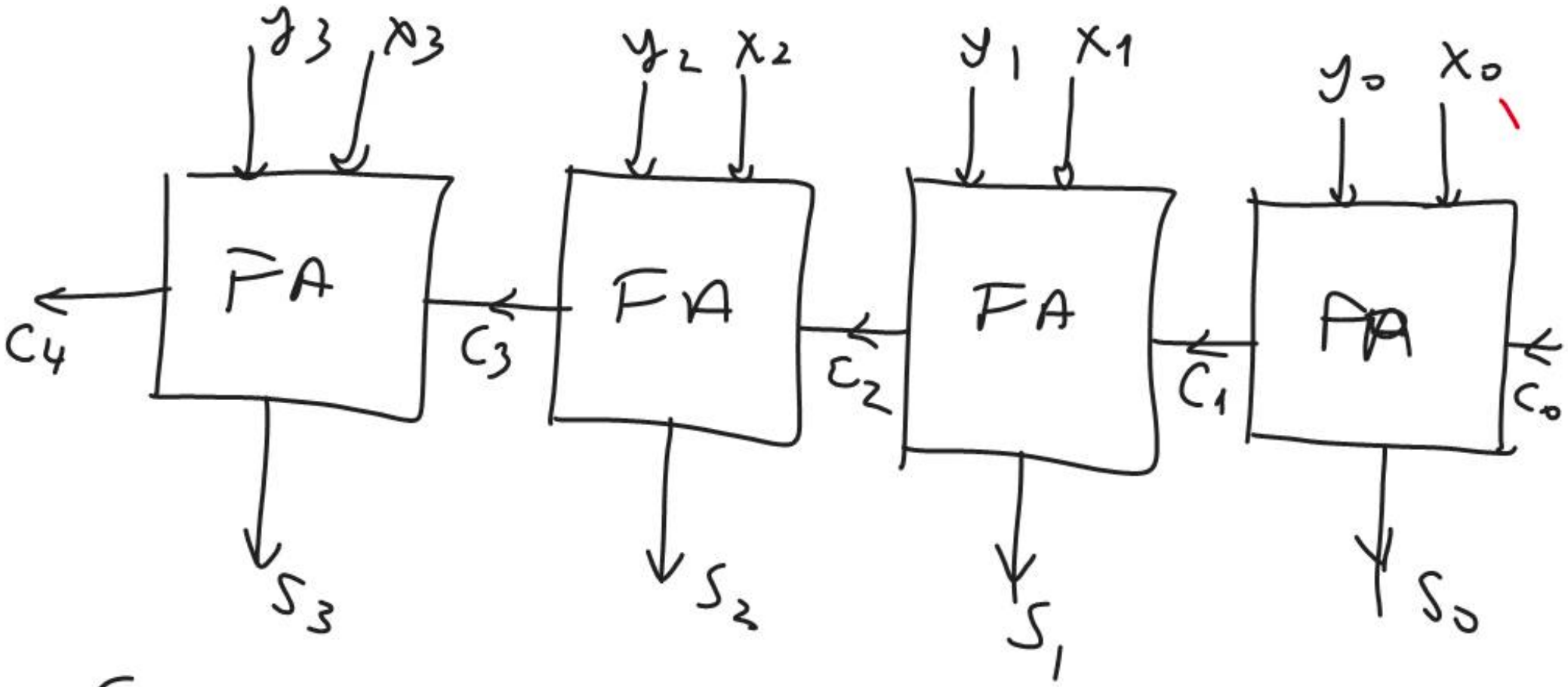




$$C_{i+1} = \overline{C_i}\, x_i y_i + C_i (x_i + y_i)$$



A Ripple Carry Adder (4-bit)



Carry is propagated, and the summation at each FA cell should wait for the carry generated from the previous FA cell.

Look ahead carry generation:

$$C_{i+1} = x_i \cdot y_i + c_i x_i + c_i y_i$$

$$C_{i+1} = \underbrace{x_i \cdot y_i}_{\text{generation function } g_i} + c_i \underbrace{(x_i + y_i)}_{\text{propagation function } = p_i}$$

$$C_{i+1} = g_i + p_i c_i$$

$$C_{i+1} = g_i + p_i (g_{i-1} + p_{i-1} c_{i-1})$$

$$C_{i+1} = g_i + p_i g_{i-1} + p_i p_{i-1} C_{i-1}$$

$$C_{i+1} = g_i + p_i g_{i-1} + p_i p_{i-1} (g_{i-2} + p_{i-2} C_{i-2})$$

$$C_{i+1} = g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + p_i p_{i-1} p_{i-2} C_{i-2}$$

In the end each carry bit can be obtained as a function of only x_i, y_i and c_0.

This means, knowing x_i, y_i and c_0, all carry bits can be generated instantly by a logic circuit for each. This increases the complexity, but all sums and carry bits are instantly generated and with today's technology it's not a big deal.

SIGNED NUMBERS

(1) Sign and magnitude → $+5_{10} = +101_2$

$-5 = -101$

(2) 1's complement

(3) 2's complement

1's COMPLEMENT

n-bit negative number K

$$K = (2^n - 1) - P$$

where P is the positive equivalent of K

ex $-5_{10} = (15_{10} - P)$ $\quad -5_{10} = (1111)_2 - (\underbrace{0101}_{+5})$

positive equivalent of -5

if $n = 4$

$K = (2^4 - 1) - P$

$K = 15 - P$

$-5_{10} = (1111)_2 - (\underbrace{0101}_{+5}) = 1010_2$

K P

$$\begin{array}{r} 1111 \\ -\ 0101 \\ \hline 1010 \end{array}$$

4-bit ($n=4$)

$+5 \Rightarrow 0101$

$-5 \Rightarrow 1010$ (take each bit's complement to obtain the 1's complement)

First bit (LeftMostBit = LMB) is the sign bit.
LMB = 1 → negative
LMB = 0 → positive

Positive		negative	
0000	0	1111	-0
0001	1	1110	-1
0010	2	1101	-2
0011	3	1100	-3
0100	4	1011	-4
0101	5	1010	-5
0110	6	1001	-6
0111	7	1000	-7

It has 2 ZEROES!

$0000 = 0_{10}$

$1111 = -0_{10}$

positive $0 - (2^{n-1} - 1)$

negative $0 - (2^{n-1} - 1)$

③ 2's COMPLEMENT SIGNED NUMBERS

$$K = 2^n - P$$

K is a negative number in 2's complement system.

P → positive equivalent of K

ex Decimal, 2-digit $n = 2$

$$K = 10^n - P$$

n → base

$65 \Rightarrow -35$

$$-35 = 10^2 - (35) = 100 - 35 = 65$$

In binary integer signed numbers

$A_2 = 10011 \Rightarrow$ 2's complement of A = $01100 + 1 = 01101$

$n = 4$

$1111 + 1 = 0000$

positive	negative
0000	1111 → -1_{10}
0001	1110 — -2
0010	1101 — -3
0011	1100 — -4
0100	1011 — -5
0101	1010 — -6
0110	1001 — -7
0111	1000 — -8

sign bit

$2^{n-1} - 1$ positive

2^{n-1} negatives