# ECE 476

# SPRING 2010

Lab:
Fatih GENÇ

Celal Zaim Çİl

1. Murat BÜK
2. Tuğçe BAKİ
3. Burcu YAKIŞIR
4. Eren AKÇA
5. Ahmet Çağrı ARLI
6. Çağatay DURMAZ

# ARITHMETIC CIRCUITS

05 March 2015

## INTEGERS

Always positive numbers = UNSIGNED

Numbers can be negative also = SIGNED

## Decimal numbers

radix = base = 10

$D = d_{n-1} d_{n-2} \cdots d_0$ DIGITS (n-tuple number)

$V(D) = d_{n-1} \times 10^{n-1} + d_{n-2} \times 10^{n-2} + \cdots + d_1 \times 10^1 + d_0 \times 10^0$ Value of the number

## BINARY NUMBERS

radix = base = 2

$B = b_{n-1}\, b_{n-2}\, b_{n-3} \cdots b_1\, b_0$

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

$$V(B) = \sum_{i=0}^{n-1} b_i \times 2^i$$

$(1\,1\,0\,1\,1)_2 \Rightarrow 2^4 + 2^3 + 0 \times 2^2 + 2^1 + 2^0 = 16 + 8 + 0 + 2 + 1$

$= 27_{10}$

$2^4\ 2^3\ 2^2\ 2^1\ 2^0$

---

Octal (radix 8)

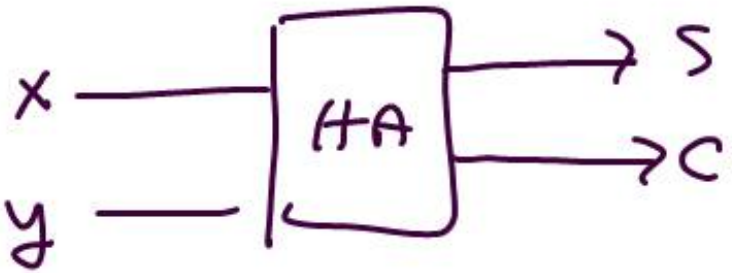
$(1\,0\,0\,1\,0\,1\,1\,0\,1\,1\,0\,0)_2 = (?)_8 = 4554_8$

4 5 5 4

Hexadecimal (radix 16) $= 96C_{16}$

# ADDITION of UNSIGNED NUMBERS

## HALF ADDER

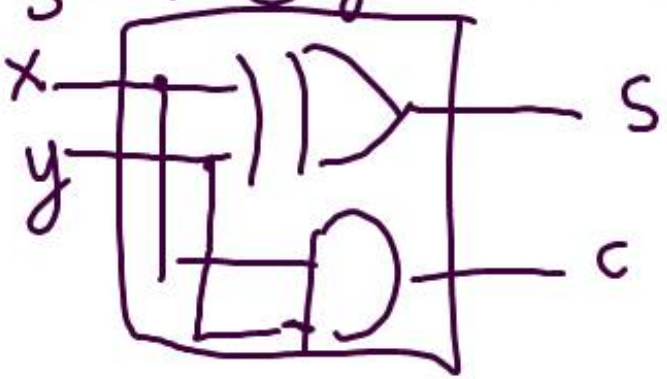



| x | y | S | C |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

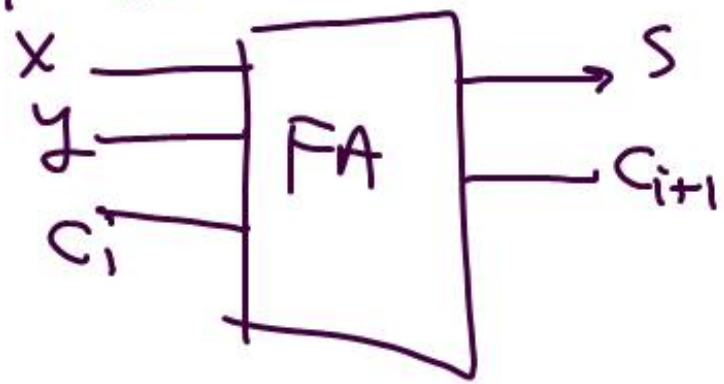
$$\begin{array}{r} 01 \\ +\ 01 \\ \hline 10 \end{array}$$

C S

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

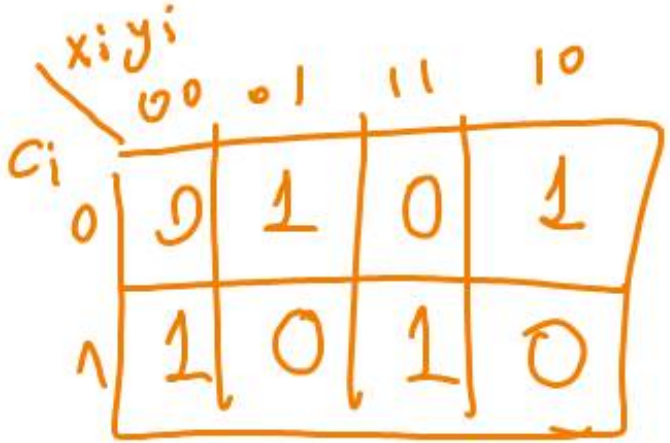



## FULL ADDER



# FULL ADDER

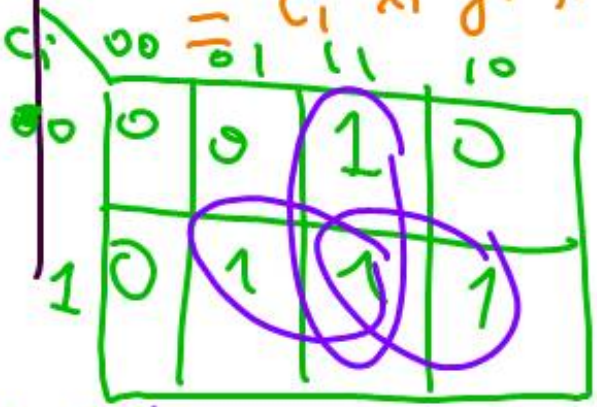
| $c_i$ $x_i$ $y_i$ | $S_i$ | $c_{i+1}$ |
|---|---|---|
| 0 0 0 | 0 | 0 |
| 0 0 1 | 1 | 0 |
| 0 1 0 | 1 | 0 |
| 0 1 1 | 0 | 1 |
| 1 0 0 | 1 | 0 |
| 1 0 1 | 0 | 1 |
| 1 1 0 | 0 | 1 |
| 1 1 1 | 1 | 1 |




$S_i$

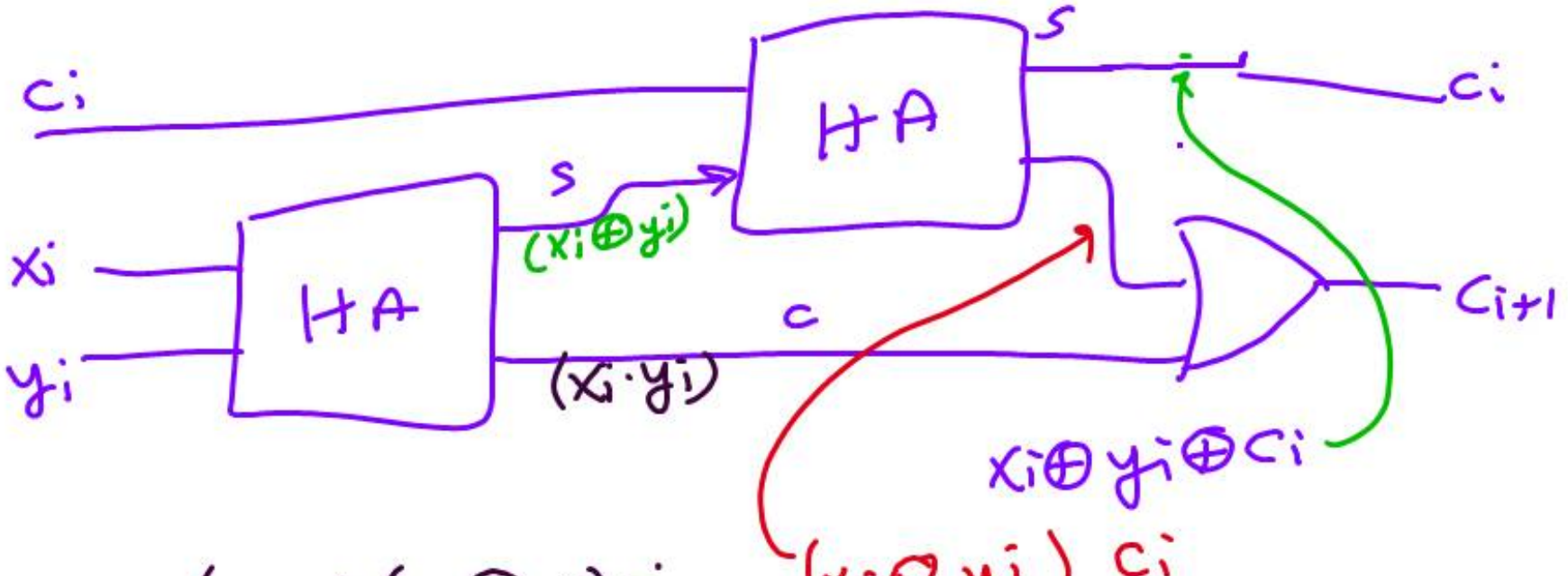
$$S_i = x_i \oplus y_i \oplus C_i$$

$$\bar{c_i}\,\bar{x_i}\,y_i + \bar{c_i}\,x_i\,\bar{y_i} + c_i\,\bar{x_i}\,\bar{y_i} + c_i x_i y_i$$




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

$C_{i+1}$




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

$$= x_i y_i + (\bar{x_i} y_i + x_i \bar{y_i}) c_i$$

$$= x_i y_i + c_i \bar{x_i} y_i + c_i x_i \bar{y_i}$$

$$= y_i (x_i + c_i \bar{x_i}) + c_i x_i \bar{y_i}$$

$$= y_i (x_i + c_i)(\underbrace{x_i + \bar{x_i}}_{1})$$

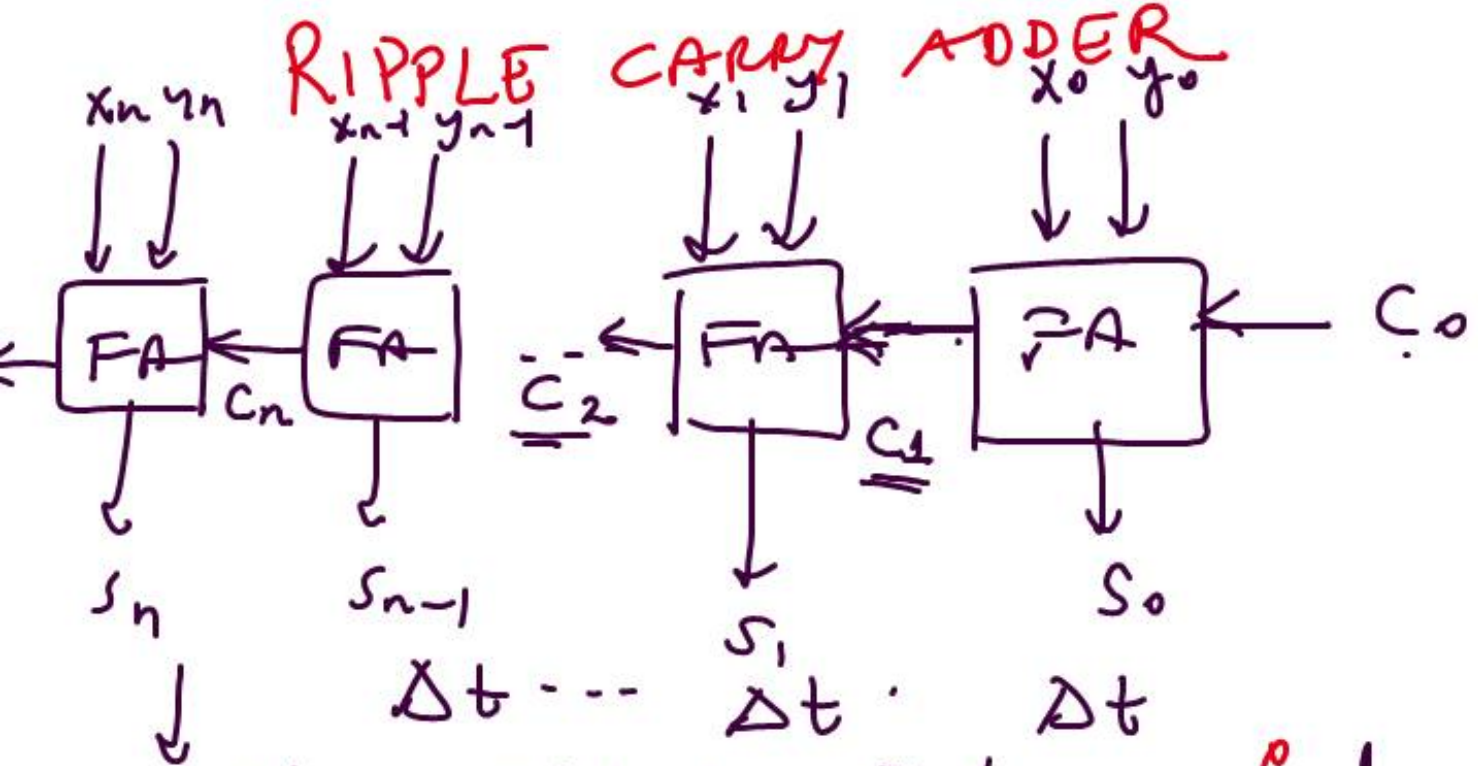
$$= y_i x_i + y_i c_i + c_i x_i \bar{y_i}$$

$$= y_i x_i + c_i (y_i + x_i y_i)$$

$$= y_i x_i + c_i \big( (y_i + x_i)(\underbrace{y_i + \bar{y_i}}_{1}) \big)$$

$$= y_i x_i + c_i y_i + c_i x_i$$

# RIPPLE CARRY ADDER




$n \cdot \Delta t$ need to be waited.

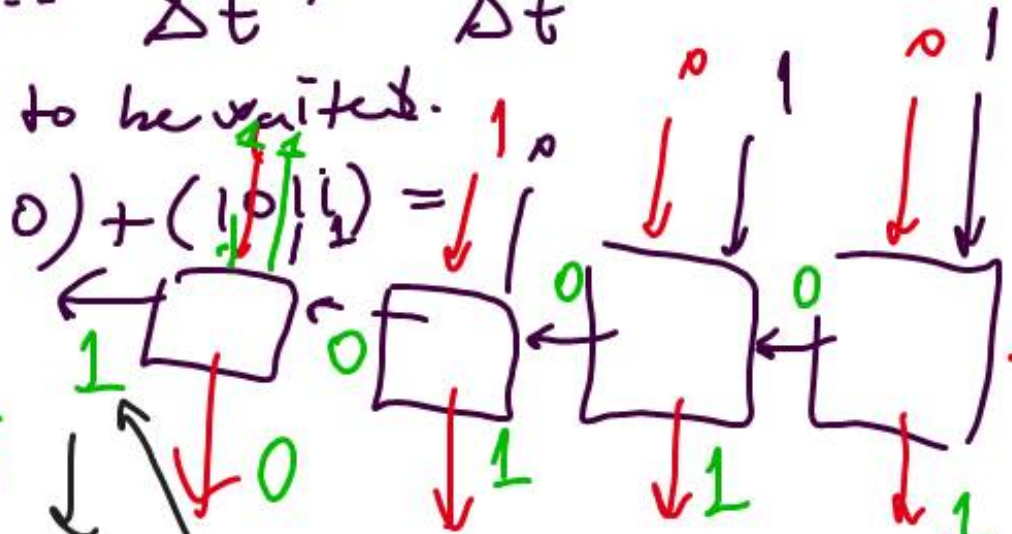
Ripple

$(1100) + (1011) =$

1100 = 12

1011 = 11

23




$= 1\ 0\ 1\ 1\ 1$

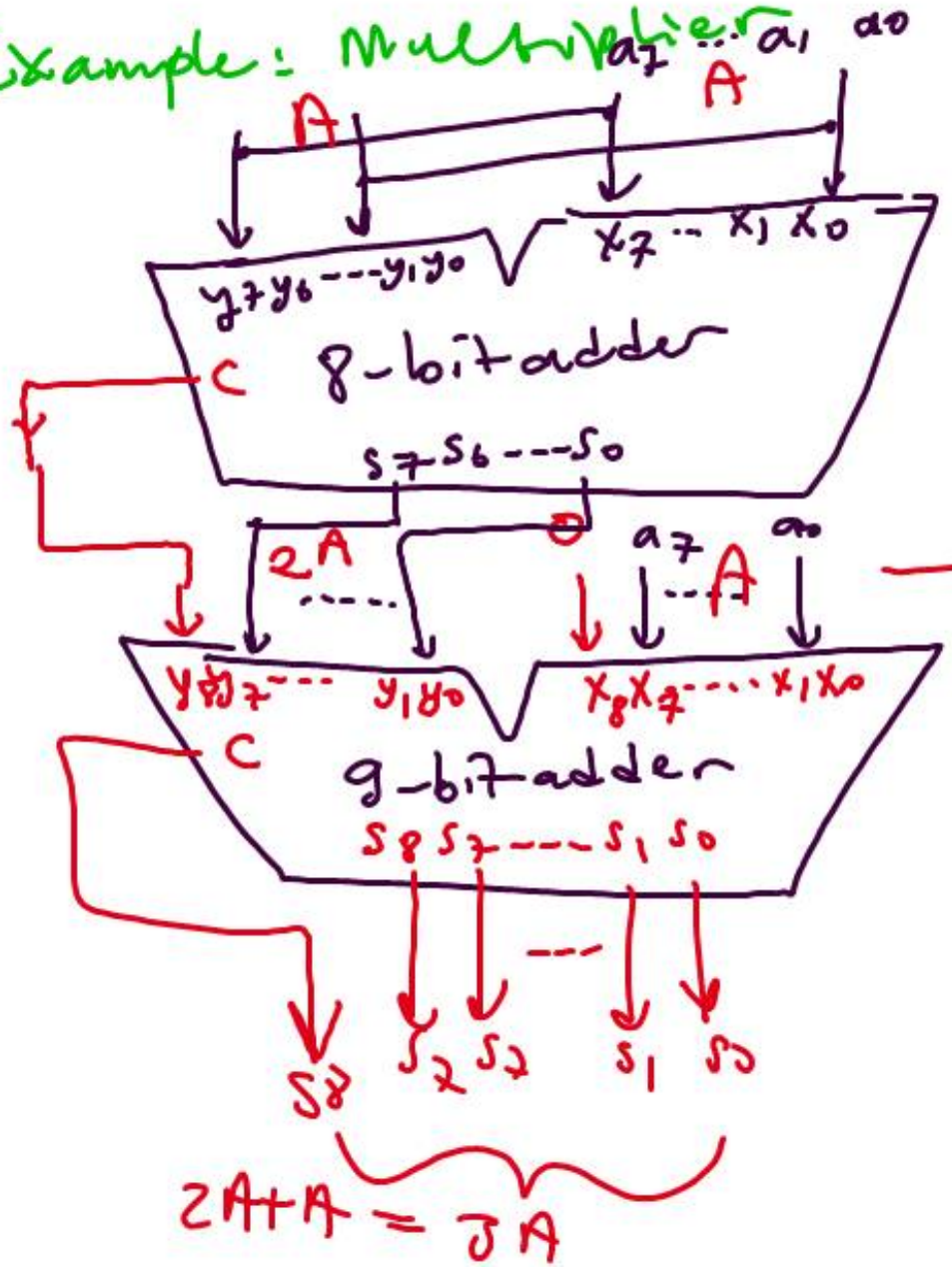
$16 + 4 + 2 + 1$

$= 23_{10}$

Wait = $4 \times \Delta t$ to obtain

Example: Multiplier

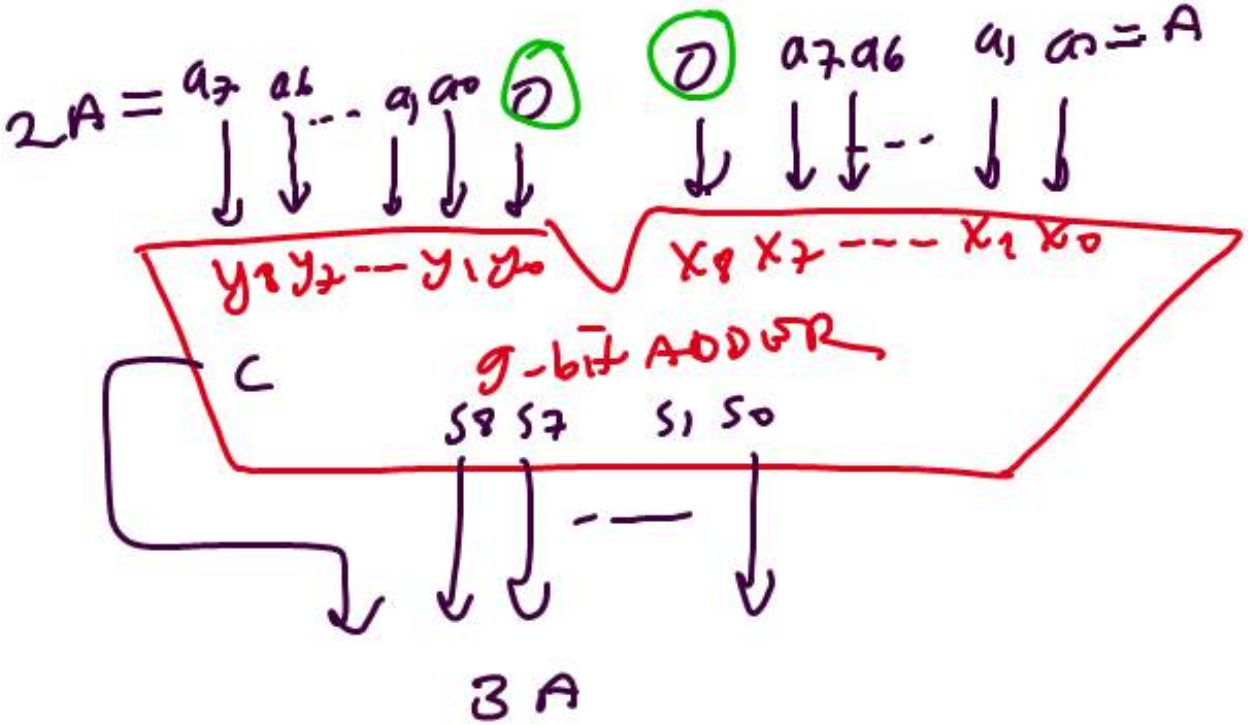
$A = a_7 a_6 \cdots a_1 a_0$




$0A = A$

$A \times 3 = 3A$

multiplication by summation




$0011 = 3_{10}$

$00110 = 6_{10}$

$01100 = 12_{10}$

$= a_7 a_6 a_5 \dots a_1 a_0 0$ } = Multiply by 2

shifting 1 bit left.
and adding 0 from right

# SIGNED NUMBERS

1) Sign and magnitude  MSB ⇒ sign

0 101
+ 5 = +5

1 101 = −5
− 5

2) One's complement

0 111 = +7$_{10}$ ⟶ 1000 = −7
+ 1

1 111 → 0000
= −0  +0

problem! 2 zeroes.

3) 2's complement

0000 → one zero!

(0 101) ⇒ 2's comp: 1010
+5  + 1
1011
= −5

## 4-bit numbers:

| | |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| −8 | 1000 |
| −7 | 1001 |
| −6 | 1010 |
| −5 | 1011 |
| −4 | 1100 |
| −3 | 1101 |
| −2 | 1110 |
| −1 | 1111 |

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

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

$n=4$

$2^{4-1}-1 = 8-1 = 7$

$-2^{n-1} = -2^3 = -8$

$1000 = 0111 + 1$

$0111 + 1 = 1000$

$-8$

$1101 \rightarrow 0010 + 1 = 0011$

$-3$

$1001 \rightarrow 0110 + 1 = 0111$

$-7$

# OVERFLOW

10 Mar 2010

In 2's complement numbers OVERFLOW may occur (in addition & subtraction)

Signs same!

$$\begin{array}{r} +4 \\ +5 \\ \hline +9 \end{array}$$

Signs same!

$$\begin{array}{r} -6 \\ -5 \\ \hline -11 \end{array}$$

$$\begin{array}{r} +5 \\ -2 \\ \hline +3 \end{array}$$

$$\begin{array}{r} -5 \\ +4 \\ \hline -1 \end{array}$$

cannot be represented by 4 bits! $\Rightarrow$ OVERFLOW

4 bits to represent: $\{0,1,2,3,4,5,6,7$ $-1,-2,-3,-4,-5,-6,-7,-8\}$ in 2's complement representation

n-bits $\rightarrow$ $2^{n-1}-1$ positive VALUES, $2^{n-1}$ negative numbers VALUES

+4
+5
+9

0100
+0101
1001
(-7) X

0110
1
0111

$C_1 = 0$
$C_2 = 0$
$C_3 = 1$
$C_4 = 0$

Have different values!

OVERFLOW!

Overflow = $C_3 \oplus C_4$

-6
+-5
-11

1010
+1011
10101

-6 = 0110
1001
1
1010

-5 ⇒ 1011
0101
1010
1
1011

+5 X

$C_3 = 0$
$C_4 = 1$

Overflow = $C_3 \oplus C_4 = 1$

$$\begin{array}{r} +5 \\ +\ -3 \\ \hline +2 \end{array} \qquad \begin{array}{r} 0101 \\ +\ 1101 \\ \hline 10010 \end{array}$$

$-3 = 0011$

$1100$

$+\ \ 1$

$1101$

$+2$ ✓

$C_3 = 1$

$C_4 = 1$

$C_4 \oplus C_3 = 0$

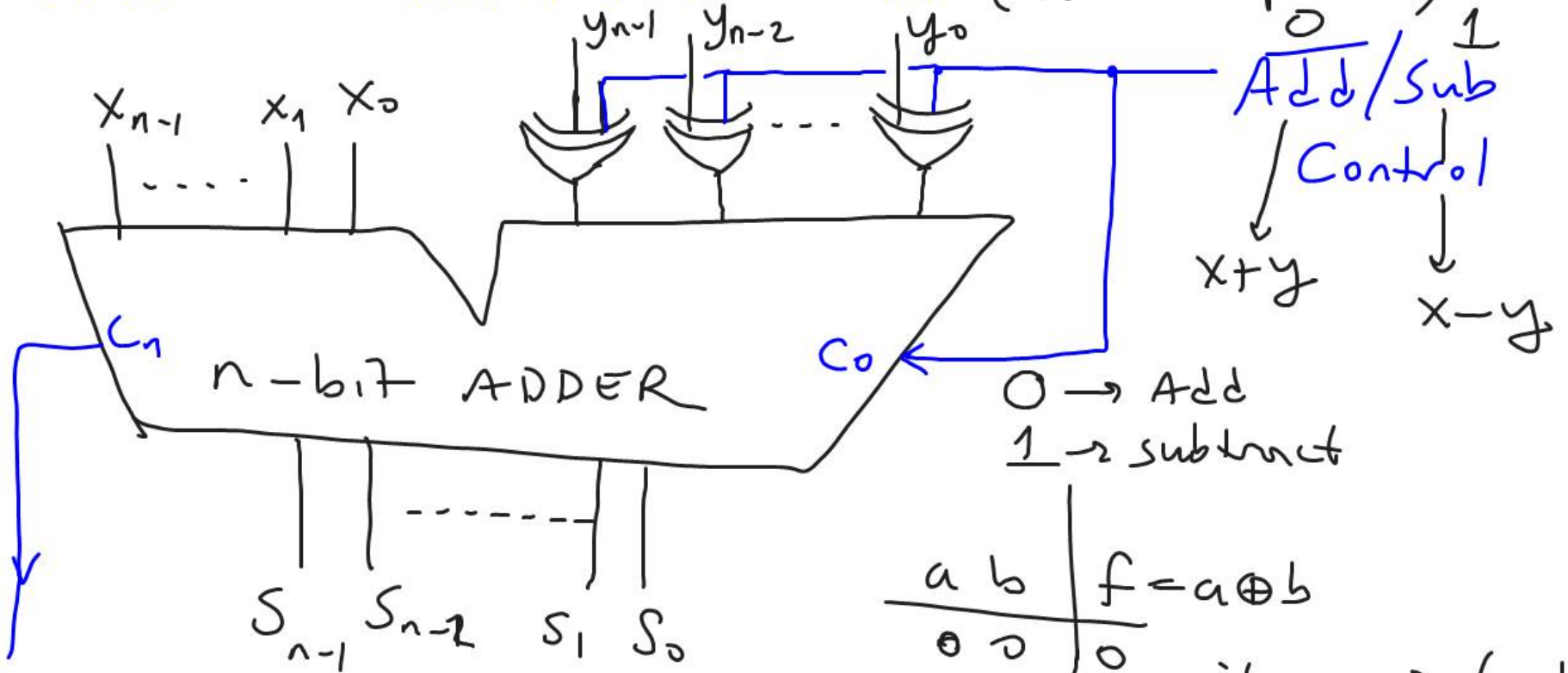
No overflow ✓

## In General:

For n-bits 2's complement addition, subtraction Overflow is detected by:

$$V = C_n \oplus C_{n-1}$$

# ADDER & SUBTRACTOR UNIT (2's complement)




0 → Add

1 → subtract

| a | b | f = a ⊕ b |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

if a = 0 ⇒ f = b

if a = 1 ⇒ f = $\bar{b}$

$X + (-y) = X - y$

# FAST ADDERS (Look Ahead Carry Adder)

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

$$C_{i+1} = x_i y_i + (x_i + y_i) c_i$$

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

$g_i$ = generation function $= x_i y_i$

$p_i$ = propagation function $= x_i + y_i$

$$c_i = g_{i-1} + p_{i-1} c_{i-1}$$

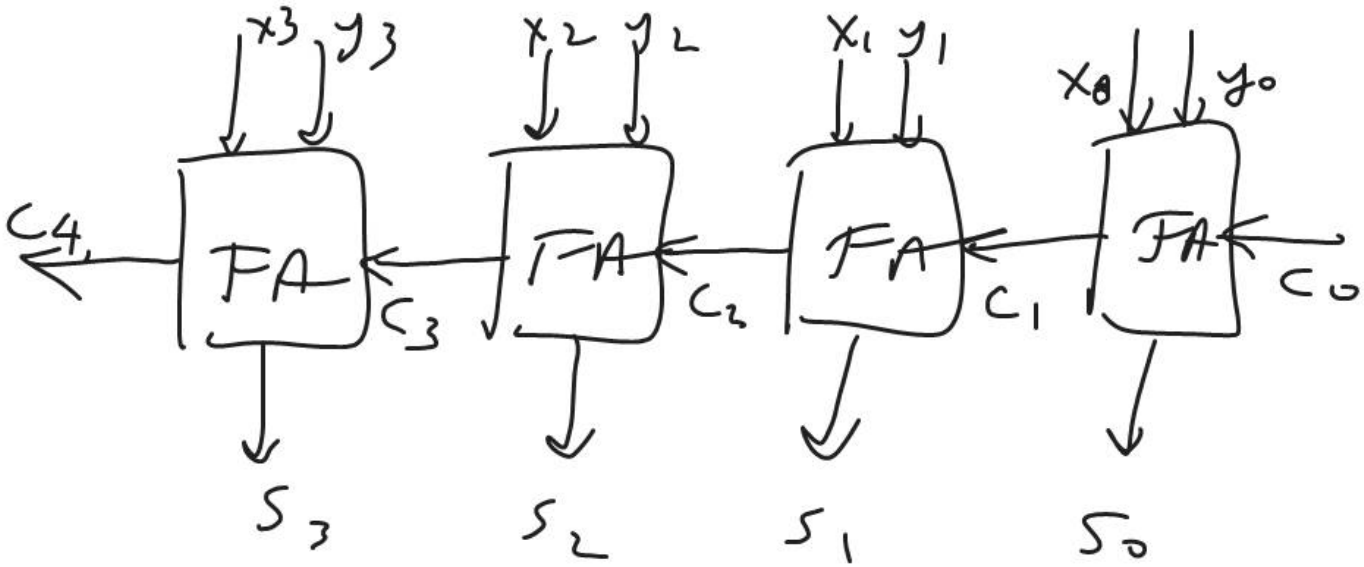
$$C_{i+1} = g_i + p_i (g_{i-1} + p_{i-1} c_{i-1})$$

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

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

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

# 4-bit Adder




$$C_4 = X_3Y_3 + (X_3+Y_3)\cdot C_3$$

$g_3$ $p_3$ $C_3$

$$C_4 = g_3 + p_3 C_3$$

$$C_4 = g_3 + p_3(g_2 + p_2 C_2)$$

$$C_4 = g_3 + p_3 g_2 + p_3 p_2 C_2$$

$$= g_3 + p_3 g_2 + p_3 p_2 (g_1 + p_1 C_1)$$

$$= g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 C_1$$

$$= g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 (g_0 + p_0 C_0)$$

$$C_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_0 p_1 C_0$$

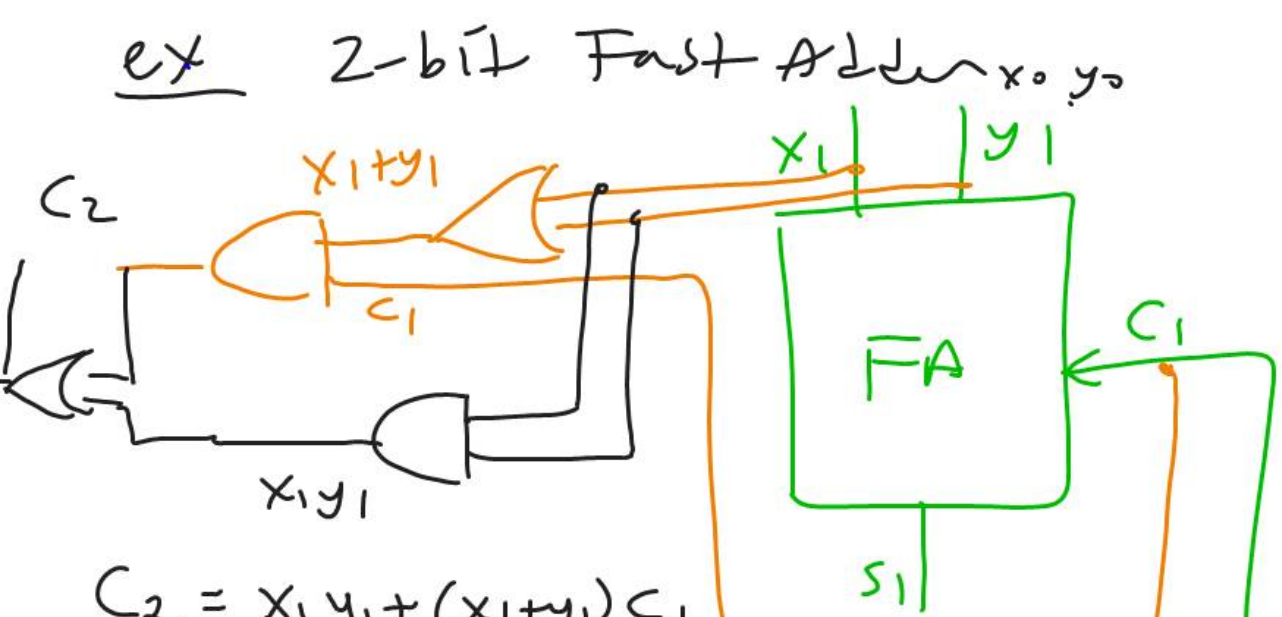
$g_3 = X_3Y_3$ $p_3 = (X_3+Y_3)$

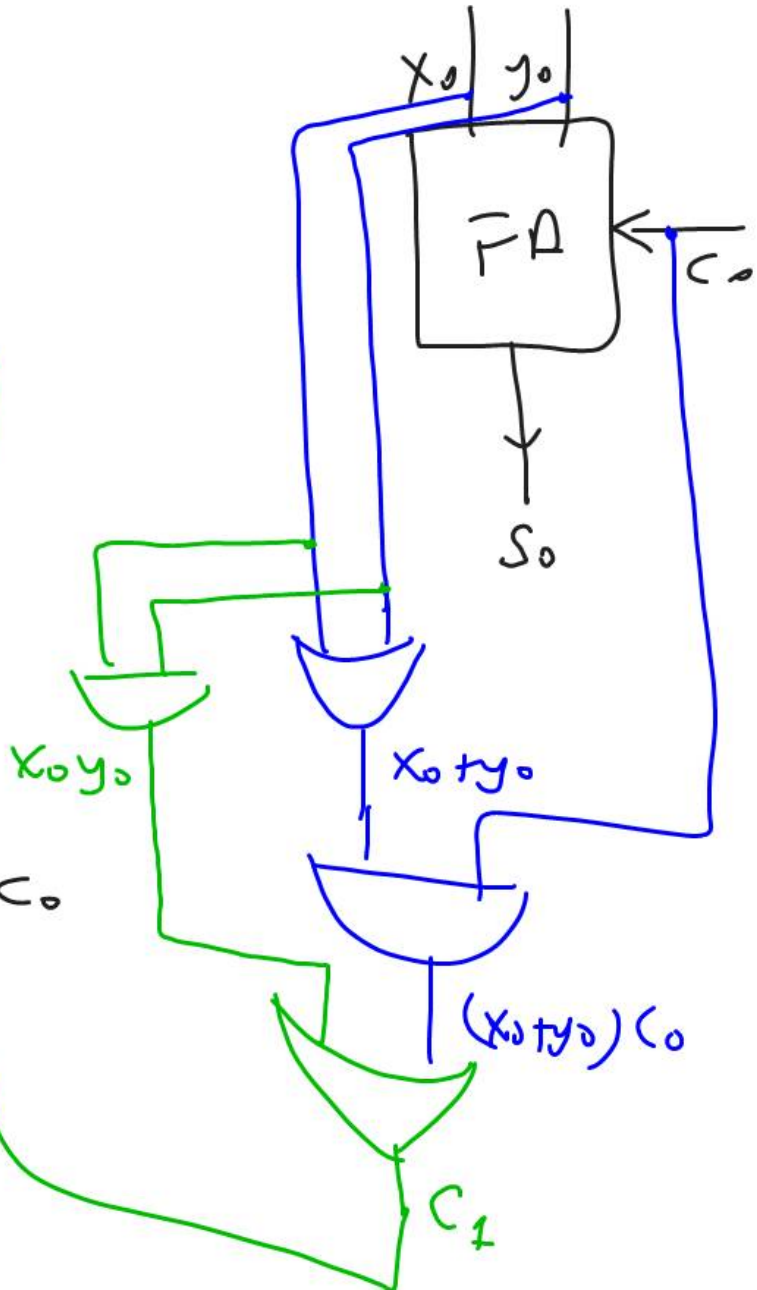
$g_2 = X_2Y_2$ $p_2 = (X_2+Y_2)$

$g_1 = X_1Y_1$ $p_1 = (X_1+Y_1)$

$g_0 = X_0Y_0$ $p_0 = (X_0+Y_0)$

ex 2-bit Fast Adder




$$C_2 = x_1y_1 + (x_1+y_1)C_1$$

$$C_1 = x_0y_0 + (x_0+y_0)C_0$$

---

$$C_2 = x_1y_1 + (x_1+y_1)(x_0y_0) + (x_1+y_1)(x_0+y_0)C_0$$

$$C_1 = x_0y_0 + (x_0+y_0)C_0$$

2 hours in INT-3 Classroom. 11 Mar 2020

→ Design of Arithmetic circuit

→ Multibit signals

→ Arithmetic Assignment statements

→ n-bit full adder → unsigned numbers

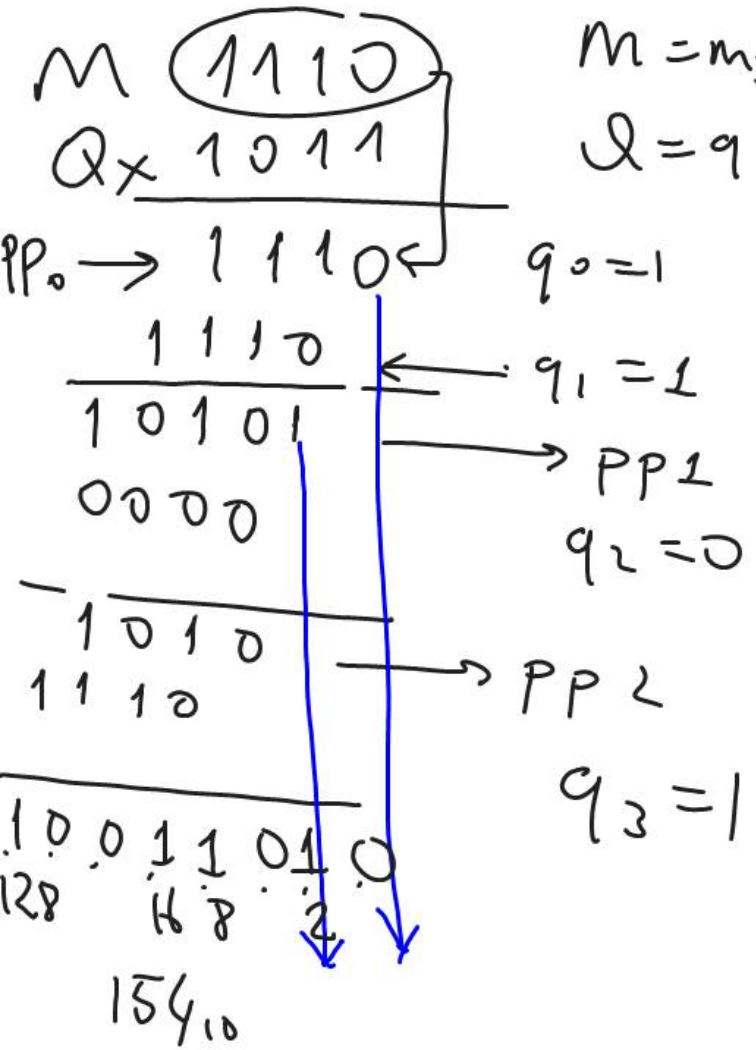
→ signed numbers

→ Multiplications

16-bit FA

17 mar 2010

©




$M = m_3\, m_2\, m_1\, m_0$

$Q = q_3\, q_2\, q_1\, q_0$

$P = M \times Q$ (4-bit)

$P = p_7\, p_6\, p_5\, p_4\, p_3\, p_2\, p_1\, p_0$

(8-bit)

# ARRAY MULTIPLIER

$PP0 = M$ AND $Q$

$= m_3q_0 \quad m_2q_0 \quad m_1q_0 \quad m_0q_0$ — M AND $q_0$

$0 \quad PP0_3 \quad PP0_2 \quad PP0_1 \quad PP0_0$

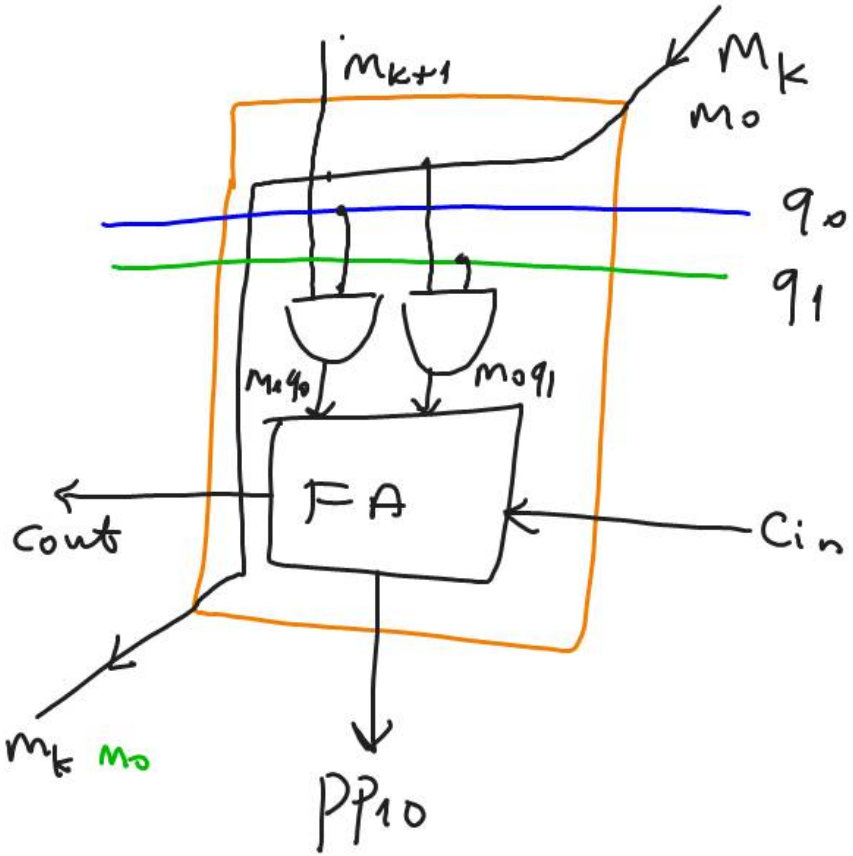
$0 \quad m_3q_0 \quad m_2q_0 \quad m_1q_0$

$+ \quad m_3q_1 \quad m_2q_1 \quad m_1q_1 \quad m_0q_1 \quad 0$ ← M AND $q_1$ and shift 1 bit LEFT

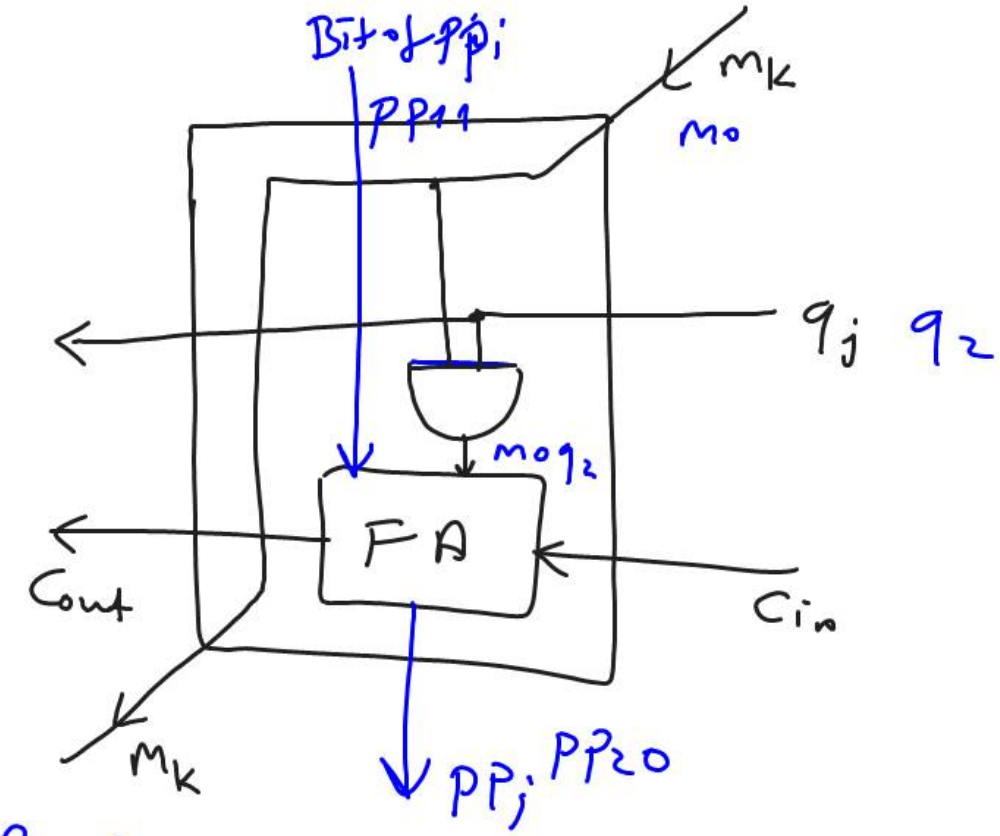
$PP1 \quad 0 \quad PP_{13} \quad PP_{12} \quad PP_{11} \quad PP_{10} \quad PP_{00}$

$+ \quad m_3q_2 \quad m_2q_2 \quad m_1q_2 \quad m_0q_2$

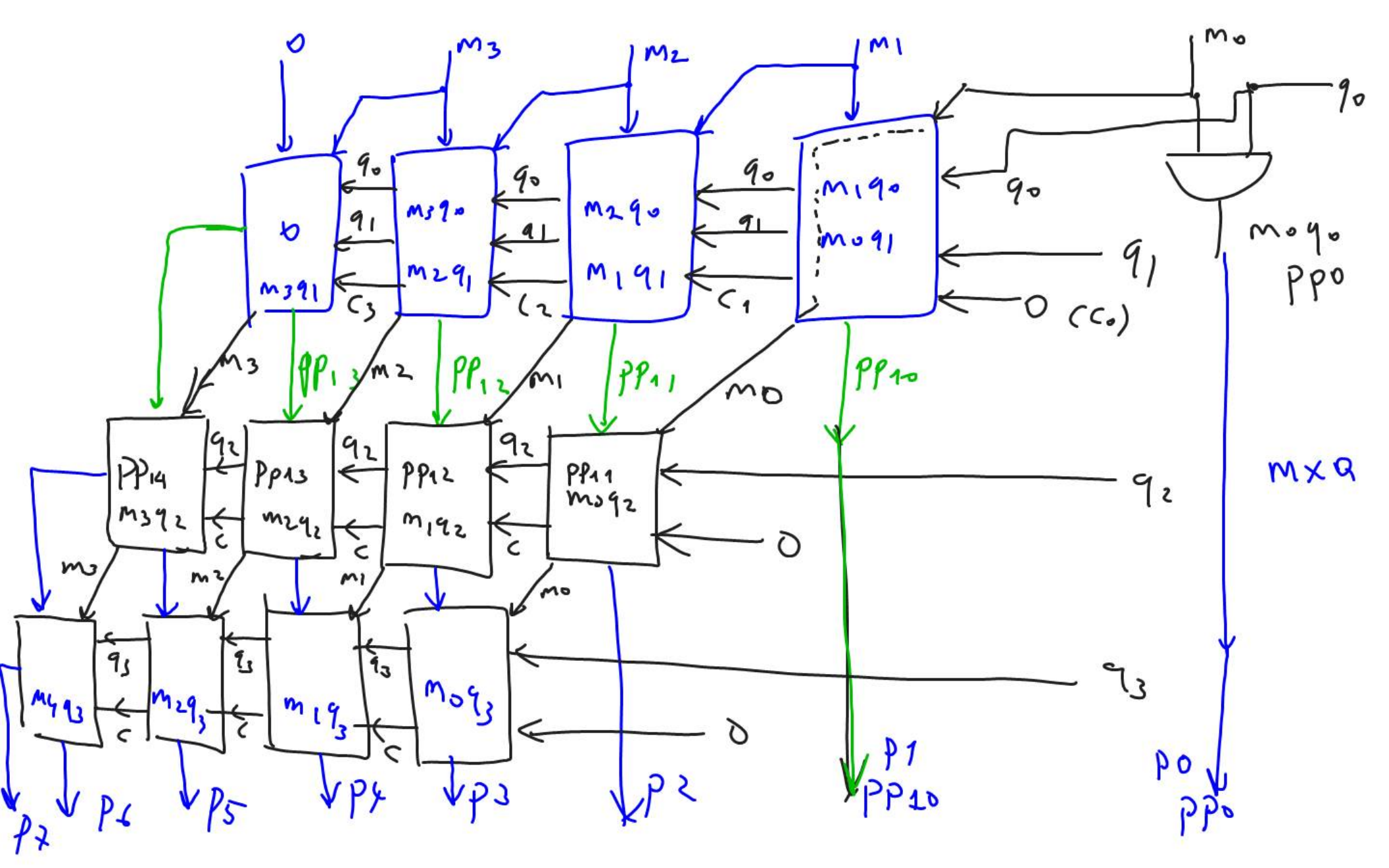
$PP2 \quad PP_{23} \quad PP_{22} \quad PP_{21} \quad PP_{20}$




1st Stage
Building Block

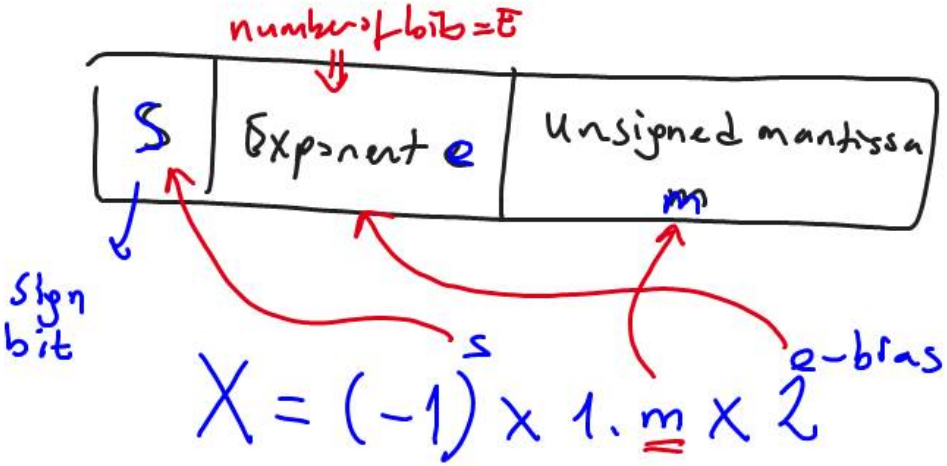



Building block for the other stages (2, 3, ...)


0
m3
m2
m1
m0
q0
q1
q2
q3
0
m3q0
m2q0
m1q0
m0q0
m3q1
m2q1
m1q1
m0q1
c3
c2
c1
c0 (c0)
PP0
PP13
PP12
PP11
PP10
m3q2
m2q2
m1q2
m0q2
PP14
m4q3
m3q3
m2q3
m1q3
m0q3
MXQ
P7
P6
P5
P4
P3
P2
P1
PP10
P0
PP0

# FLOATING-POINT NUMBERS x FIXED-POINT NUMBERS

A floating point number:




$$X = (-1)^{s} \times 1.\underline{m} \times 2^{e-bias}$$

$$bias = 2^{E-1} - 1$$

## Example:

Determine the decimal value 9.25 in a 12-bit Custom floating-point format

$A(1, 6, 5)$ $\quad 1+6+5 = 12$ bit

- 1 → sign bit
- 6 → E = 6 bit
- 5 → m, 5-bit

$$bias = 2^{6-1} - 1 = 2^{5} - 1 = 31$$

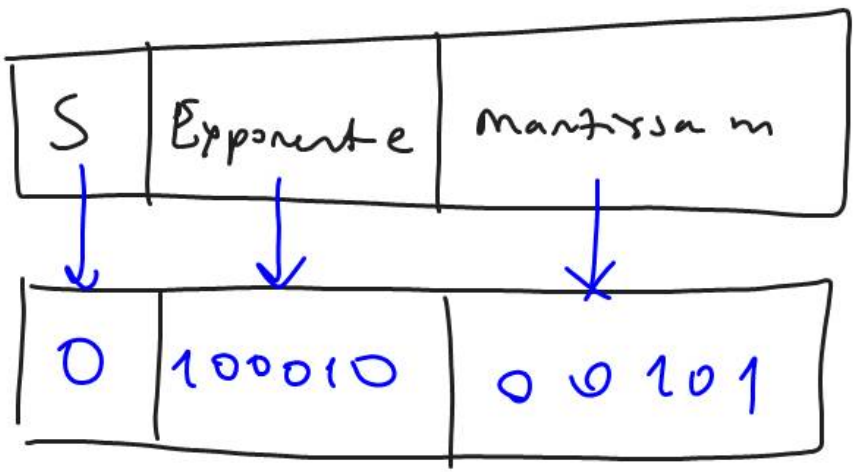
$9.25_{10} = 1001.01_2 = 1.00101 \times 2^3$

$2^3\ 2^2\ 2^1\ 2^0\ 2^{-1}\ 2^{-2}$

$8+0+0+1=9 \qquad 0+\frac{1}{4}=0.25$

$m$

$9.25_{10} =$




$X = (-1)^S \times 1.m \times 2^{e-bias}$

$m = 00101$

5-bit

$2^{e-bias} = 2^3$

$e = 3 + bias$

$e = 3 + 31 = 34_{10} = 100010_2$

e

6-bit

$X = (-1)^0 \times 1.00101 \times 2^3$

$= 9.25_{10}$

example buck conversion

---

18.03.2010
2 hours

COMBINATIONAL CIRCUITS

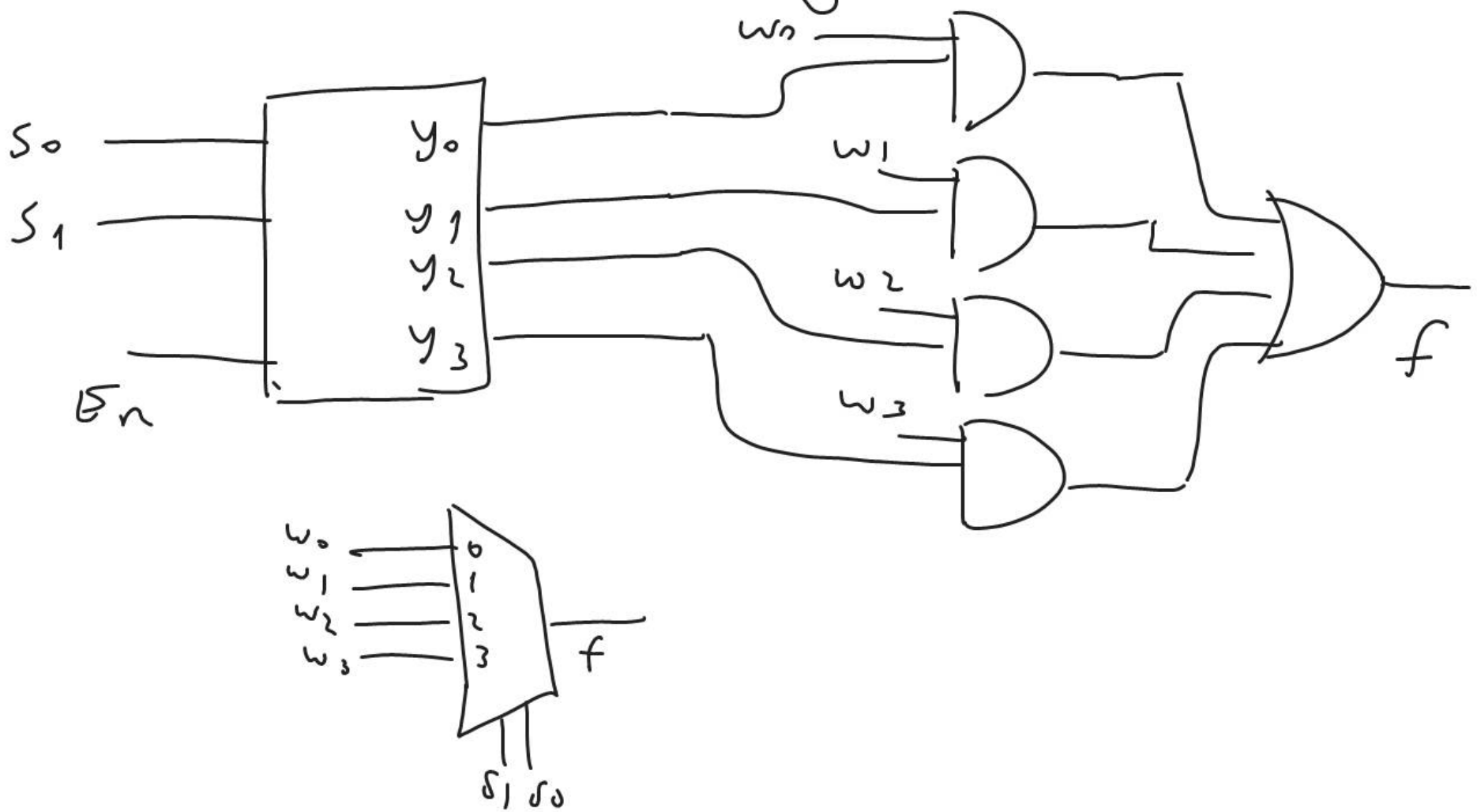
→ BCD adder

→ Decoder

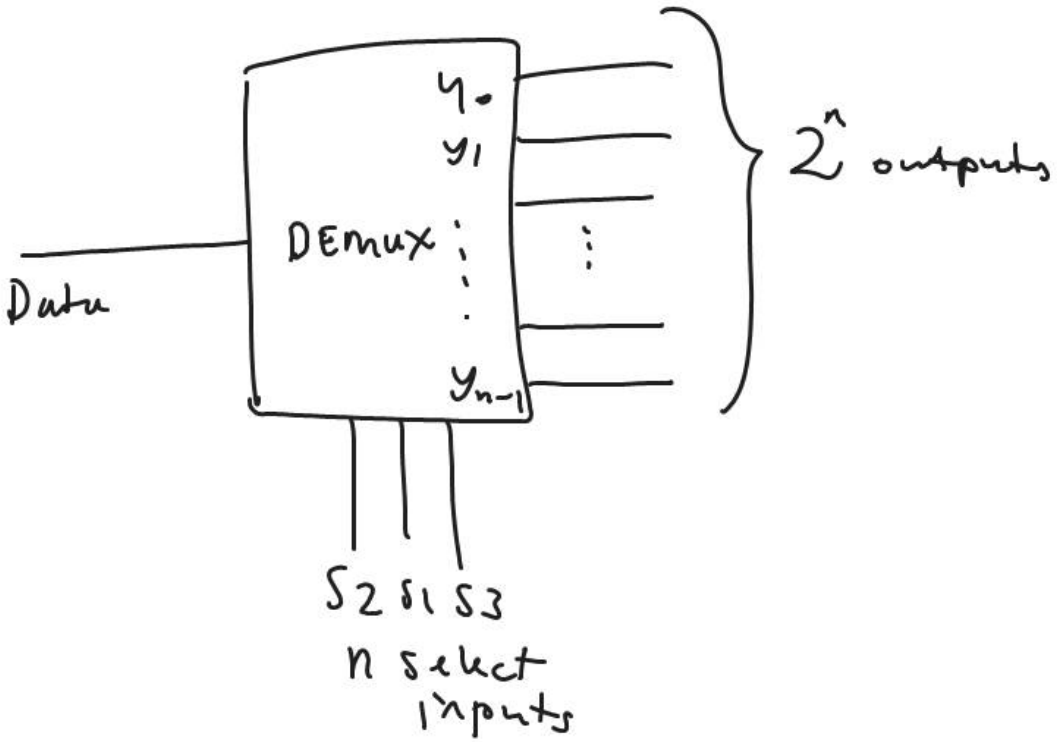
→

---

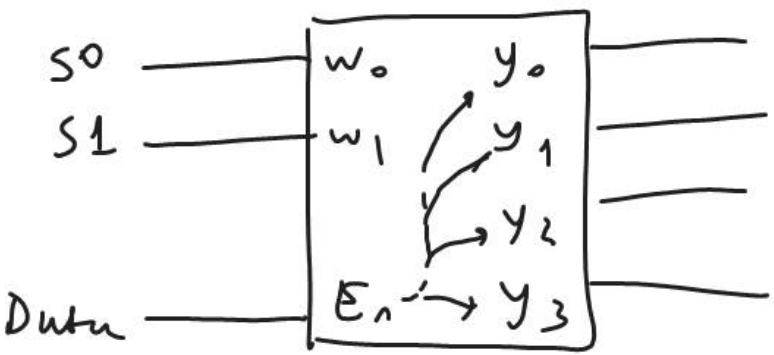
# A 4-to-1 Mux using a decoder

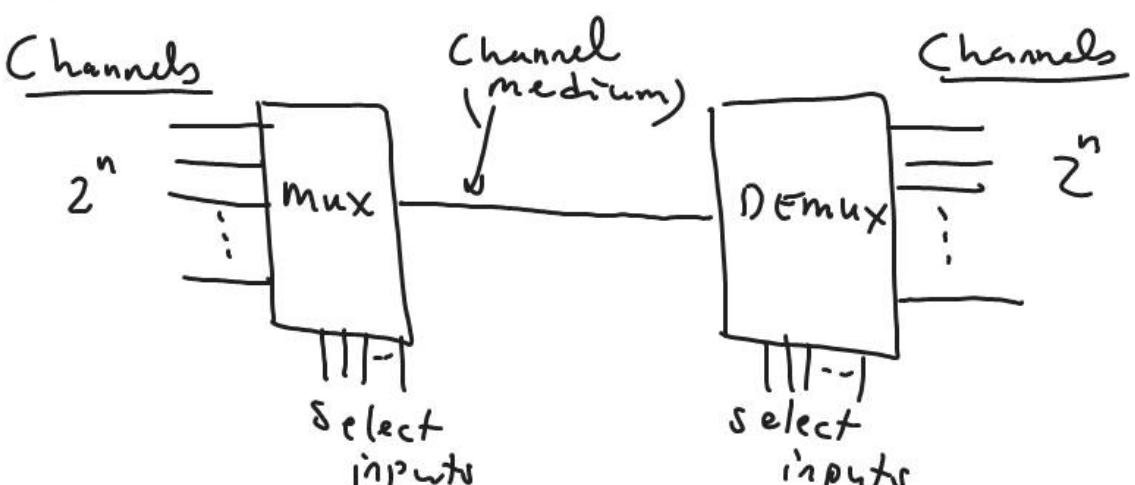


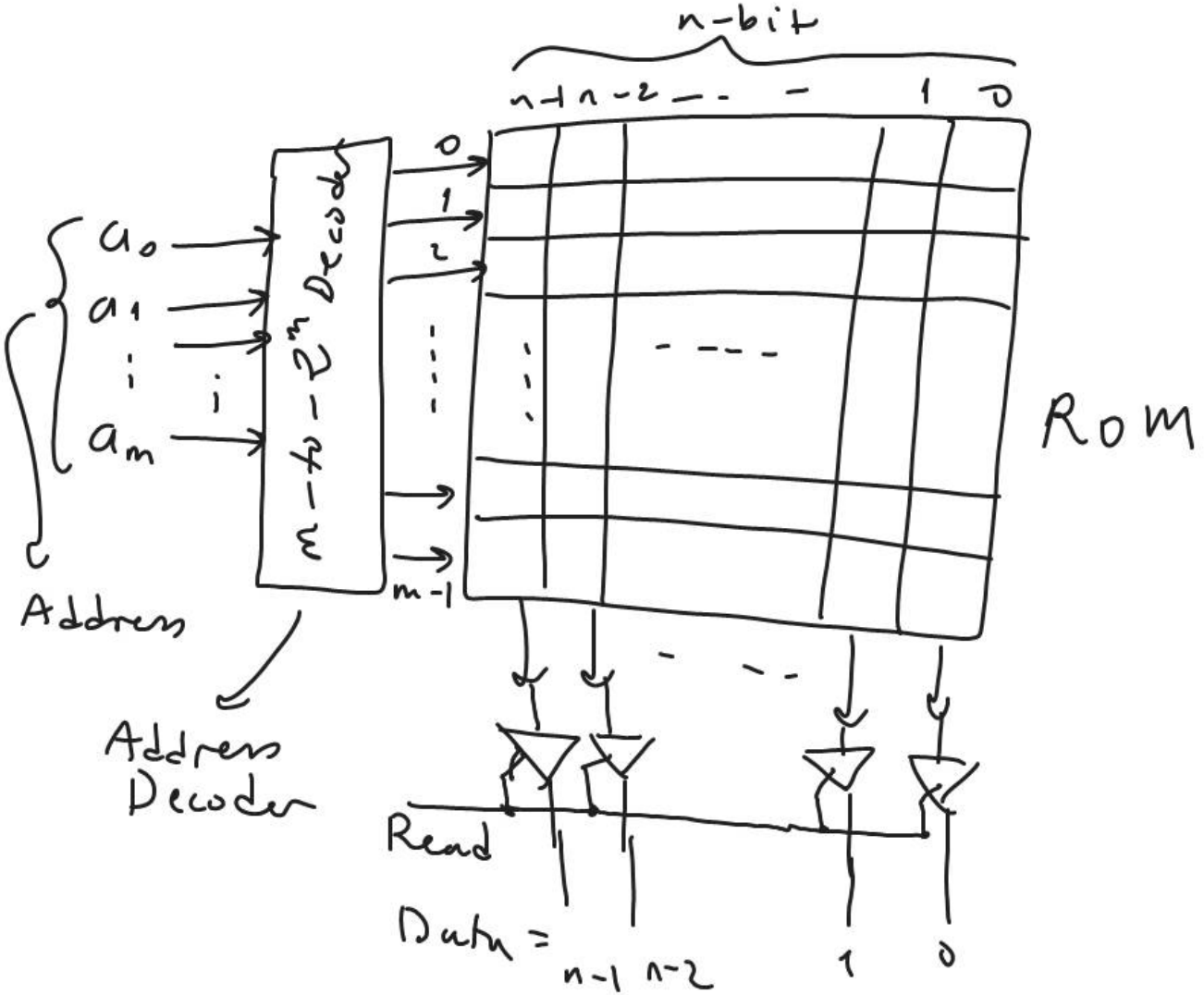
# DEMULTIPLEXERS



# USING the DECODERS as DEMUXs:

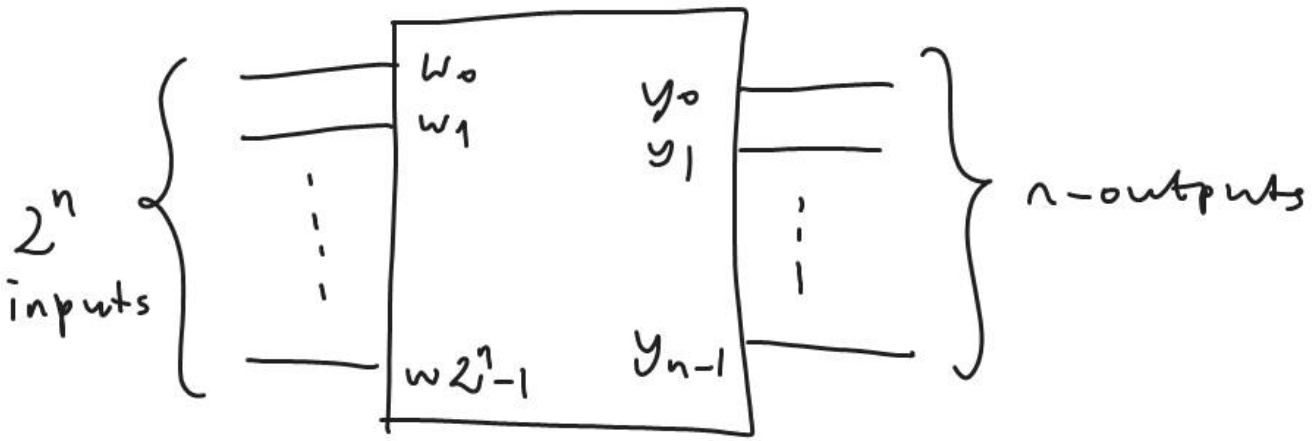





A $2^m \times n$ ROM block

# ENCODERS




(opposite of decoder)

A 4-to-2 Encoder

| $w_3$ | $w_2$ | $w_1$ | $w_0$ | $y_1$ | $y_0$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | N/A | |

→ compression
→ using less memory location

# Priority Encoders:

| | $w_3$ | $w_2$ | $w_1$ | $w_0$ | $y_1$ | $y_0$ | $z$ |
|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | d | d | 0 |
| $i_0$ | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| $i_1$ | 0 | 0 | 1 | X | 0 | 1 | 1 |
| $i_2$ | 0 | 1 | X | X | 1 | 0 | 1 |
| $i_3$ | 1 | X | X | X | 1 | 1 | 1 |

X → irrelevant (anlamsız) / ilgisiz

d → don't care

Intermediate signals:

$$i_0 = \bar{w_3}\,\bar{w_2}\,\bar{w_1}\,w_0$$

$$i_1 = \bar{w_3}\,\bar{w_2}\,w_1$$

$$i_2 = \bar{w_3}\,w_2$$

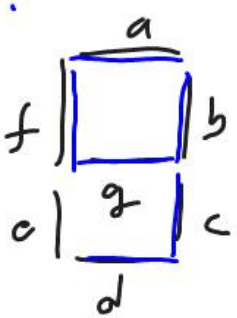
$$i_3 = w_3$$

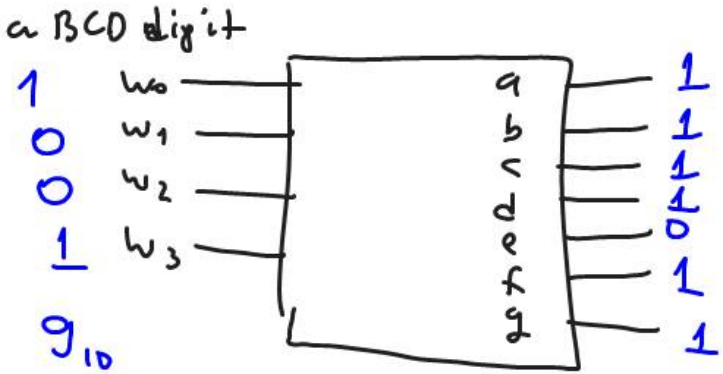
$$y_0 = i_1 + i_3$$

$$y_1 = i_2 + i_3$$

$$z = i_0 + i_1 + i_2 + i_3$$

## BCD to 7-segment ~~encoder~~ decoder ?



# ARITHMETIC COMPARISON CIRCUITS

A, B, two unsigned numbers of n-bit each

The comparator outputs:

A eq B (A = B)

A gt B (A > B)

A lt B (A < B)

ex: n = 4

$A = a_3\, a_2\, a_1\, a_0$

$B = b_3\, b_2\, b_1\, b_0$

intermediate signals:

$i_3, i_2, i_1, i_0$

A = B when $a_k = b_k$ for $i = 0$ to $n$

31 March 2010

25 Mar 2010 (2 hours)

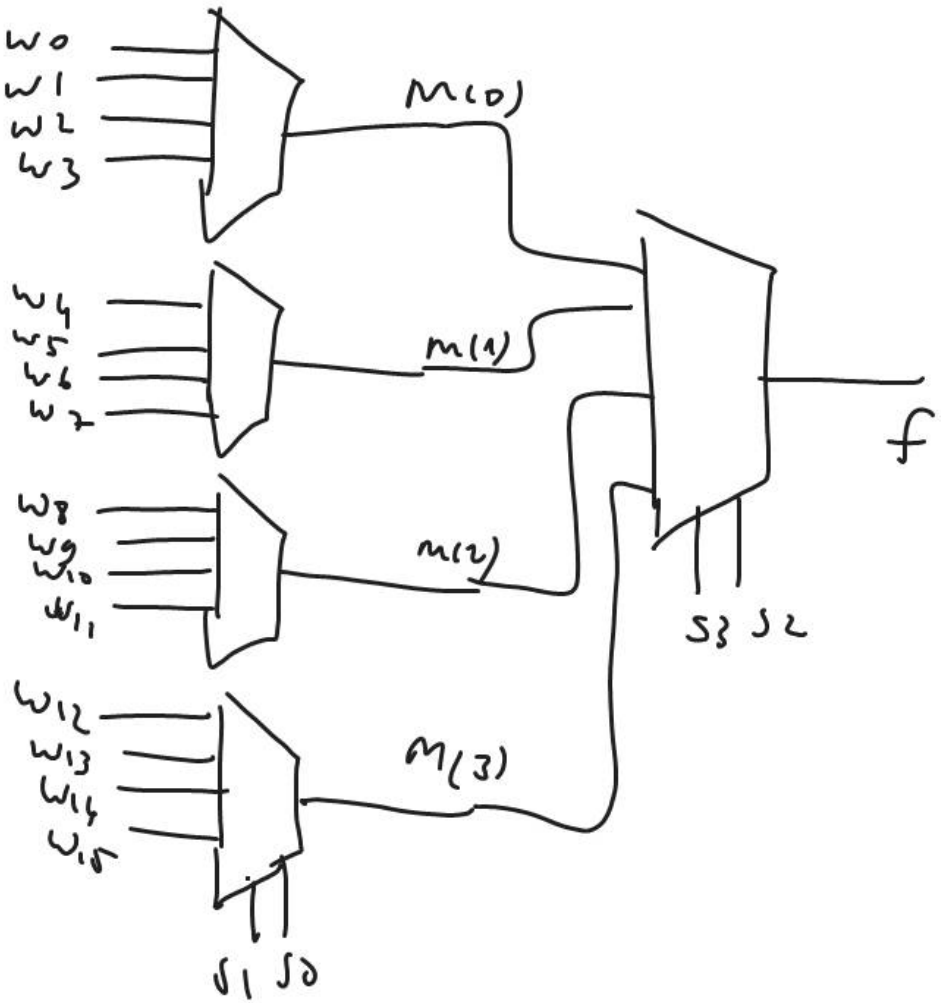
→ Arithmetic comparison circuits

→ Assignment Statements

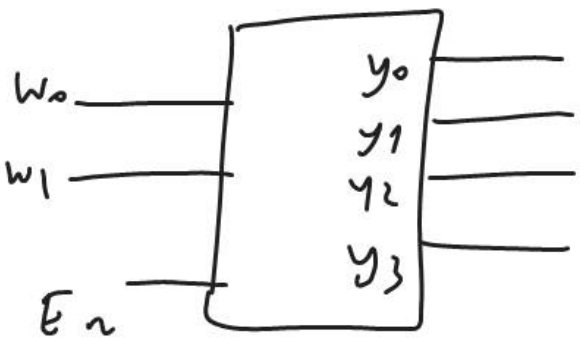
→→ Selected

→→→ 4-to-1 mux | 2-to-1 mux

An 16-to-1 mux with 5 4-to-1 mux:



# 2-to-4 Binary Decoder




| En | W1 | W0 | y0 | y1 | y2 | y3 |
|---|---|---|---|---|---|---|
| 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 |

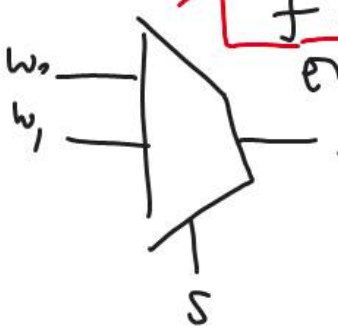
d : don't care

X = irrelevant (not applicable)

```
ENTITY dec2to4 IS
PORT( W: IN STD_LOGIC_VECTOR
      (1 DOWNTO 0);
      En: IN STD_LOGIC;
      y: OUT STD_LOGIC_VECTOR
      (1 TO 3));
END dec2to4;
ARCHITECTURE Behavior of dec2to4 IS
  SIGNAL EnW: STD_LOGIC_VECTOR
  (2 DOWN TO 0);
BEGIN
  EnW <= En & W;
  WITH EnW SELECT
  y <= "1000" WHEN "100",
       "0100" WHEN "101",
       "0010" WHEN "110",
       "0001" WHEN "111",
       "0000" WHEN OTHERS;
End Behavior;
```

CONDITIONAL SIGNAL ASSIGNMENT

```
ENTITY mux2to1 IS
PORT(W0,W1, S : IN STD_LOGIC;
     f        : OUT STD_LOGIC);
END mux2to1;
ARCHITECTURE Behavior OF 2to1 mux IS
BEGIN
  f <= W0 WHEN S='0' ELSE W1;
END Behavior;
```




2-to-1 mux

Both concurrent assignments statements.

Recall: with SELECT

```
ARCHITECTURE -------
BEGIN
WITH S SELECT
  f <= W0 WHEN '0';
       W1 WHEN OTHERS;
END Behavior;
```

# A 4-to-1 PRIORITY ENCODER

ENTITY Priority IS

| | $w_3$ | $w_2$ | $w_1$ | $w_0$ | $y_1$ | $y_0$ | $z$ |
|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | d | d | 0 |
| $i_0$ | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| $i_1$ | 0 | 0 | 1 | X | 0 | (1) | 1 |
| $i_2$ | 0 | 1 | X | X | (1) | 0 | 1 |
| $i_3$ | 1 | X | X | X | (1) | (1) | 1 |

X = irrelevant

d = don't care

$y_1 = i_2 + i_3$

$y_0 = i_1 + i_3$

$z = i_0 + i_1 + i_2 + i_3$

$i_0 = \overline{w_3}\,\overline{w_2}\,\overline{w_1}\,w_0$

$i_1 = \overline{w_3}\,\overline{w_2}\,w_1$

$i_2 = \overline{w_3}\,w_2$

$i_3 = w_3$

```
PORT( w : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
      y : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
      z : OUT STD_LOGIC);
END Priority;
ARCHITECTURE Behavior OF Priority IS
BEGIN
  y <= "11" WHEN w(3)='1' ELSE
       "10" WHEN w(2)='1' ELSE
       "01" WHEN w(1)='1' ELSE
       "00";
  z <= '0' WHEN w="0000" ELSE '1';
END Behavior;
```

CONDITIONAL Statements

# Priority encoder with SELECT STATEMENT

```
ARCHITECTURE ----
BEGIN
  WITH W SELECT
  y <= "00" WHEN "0001";
       "01" WHEN "0010";
       "01" WHEN "0011";
       "10" WHEN "0100";
       "10" WHEN "0101";
       "10" WHEN "0110";
       "10" WHEN "0111";
       "11" WHEN OTHERS;
```

```
WITH W SELECT
  Z <= '0' WHEN "0000"
       '1' WHEN OTHERS;
END Behavior;
```

# 01 April 2010 2 hours

- Implementations of Comparators
- GENERATE statement

→ For next (FOR GENERATE)
→ Conditional (IF GENERATE)

- SEQUENTIAL ASSIGNMENT STATEMENTS
  - If then else
  - CASE statement
  - PROCESS statement

Ex: priority encoder

+ 5 - 9 April 2010 Exam Week

14 April 2010

# Priority Encoder with IF-THEN

```
ARCHITECTURE Behavior OF Priority IS
 BEGIN
 PROCESS(w)
 BEGIN
   y <= "00";
 IF w(1)='1' THEN y <= "01"; END IF;
 IF w(2)='1' THEN y <= "10"; END IF;
 IF w(3)='1' THEN y <= "11"; END IF;
  z <= '1';
 IF w="0000" THEN z <= '0'; END IF;
 END PROCESS;
 END BEHAVIOR;
```

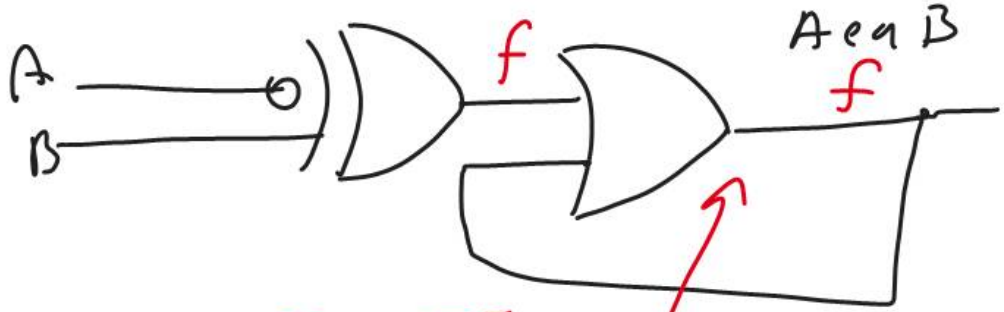
within the PROCESS the values assigned to the parameters may change. But the latest value is the valid one.

# Example: Comparator

```
ENTITY Compare1 IS
PORT( A,B : IN  STD_LOGIC;
      AeqB : OUT  " );
END Compare1
ARCHITECTURE Behavior OF Compare1 IS
 BEGIN
  PROCESS(A,B)
  BEGIN
   AeqB <= '0';
   IF A=B THEN
    AeqB <= '1';
   END IF;
 END PROCESS;
END Behavior;
```

← The DEFAULT ASSIGNMENT STATEMENT

# IMPLIED MEMORY




```
ENTITY Implied IS
PORT ( A, B : IN STD_LOGIC;
       AeqB : OUT  " " );
END Implied;
ARCHITECTURE Behavior OF Implied IS
 BEGIN
  PROCESS (A, B)
  BEGIN
   IF A=B THEN
    AeqB <= '1';
   END IF;
  END PROCESS;
 END Behavior;
```

IMPLIED MEMORY!! MISTAKE/ERROR

$f = \bar{A} \oplus B = \bar{A}B + \bar{A}\bar{B}$

$f = AB + \bar{A}\bar{B}$

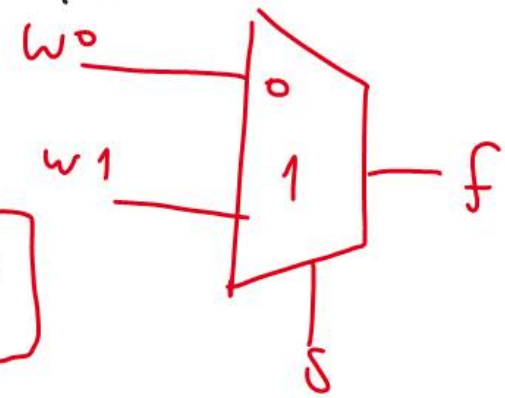
| A | B | f |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Value f will be kept forever!

# CASE STATEMENT

ex an 2-to-1 mux

```
ARCHITECTURE Behavior OF mux2 IS
BEGIN
  PROCESS (W1, W0, S)
  BEGIN
    CASE S IS
      WHEN '0' =>
        f <= W0;
      WHEN OTHERS =>
        f <= W1;
    END CASE;
  END PROCESS;
END Behavior;
```




## RECALL:

(1) CONDITIONAL

```
f <= W0 WHEN S='0'
     ELSE W1;
```

(2) WITH S SELECT

```
WITH S SELECT
  f <= W0 WHEN '0';
       W1 WHEN OTHERS;
```

(3) IF-THEN-ELSE

```
PROCESS (W0, W1, S)
BEGIN
  IF S='0' THEN f <= W0;
  ELSE f <= W1;
  END IF;
END PROCESS;
```