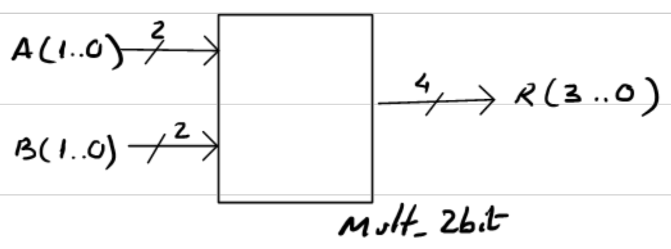
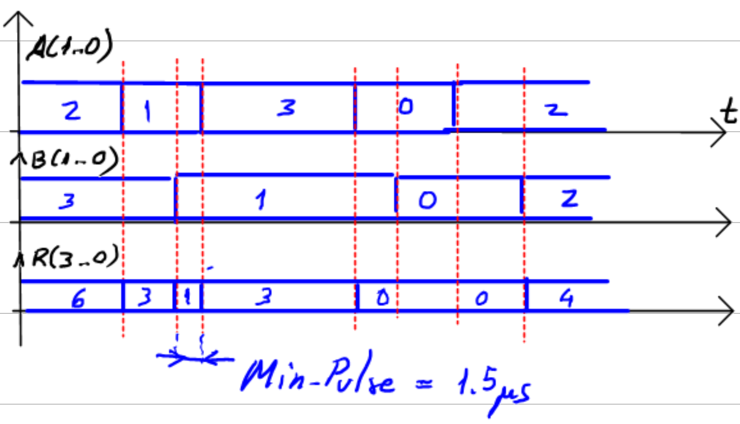
Problem 1

1. The truth table, the symbol and an example of timing diagram

	B_1	B_0	A_1	A_0	R_3	R_2	R_1	R_0
0	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
1	Ø	Ø	Ø	1	Ø	Ø	Ø	Ø
2	Ø	Ø	1	Ø	Ø	Ø	Ø	Ø
3	Ø	Ø	1	1	Ø	Ø	Ø	Ø
4	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø
5	Ø	1	Ø	1	Ø	Ø	Ø	1
6	Ø	1	1	Ø	Ø	Ø	1	Ø
7	Ø	1	1	1	Ø	Ø	1	1
8	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø
9	1	Ø	Ø	1	Ø	Ø	1	Ø
10	1	Ø	1	Ø	Ø	1	Ø	Ø
11	1	Ø	1	1	Ø	1	1	Ø
12	1	1	Ø	Ø	Ø	Ø	Ø	Ø
13	1	1	Ø	1	Ø	Ø	1	1
14	1	1	1	Ø	Ø	1	1	Ø
15	1	1	1	1	1	Ø	Ø	1

The circuit looks like a simple 2-bit multiplier with no extra signals for expansion.





When using the VHDL simulation tool, if we assume all the vectors with a Min_Pulse duration: $16 * \text{Min\_Pulse} = \underline{24\mu s}$ to test the complete table and be able to verify it.

2. Canonical forms to represent logic functions

$$R_2 = g(B,A) = \sum_4 m(10, 11, 14)$$

$$R_1 = f(B,A) = \prod_4 M(0,1,2,3,4,5,8,10,12,15)$$

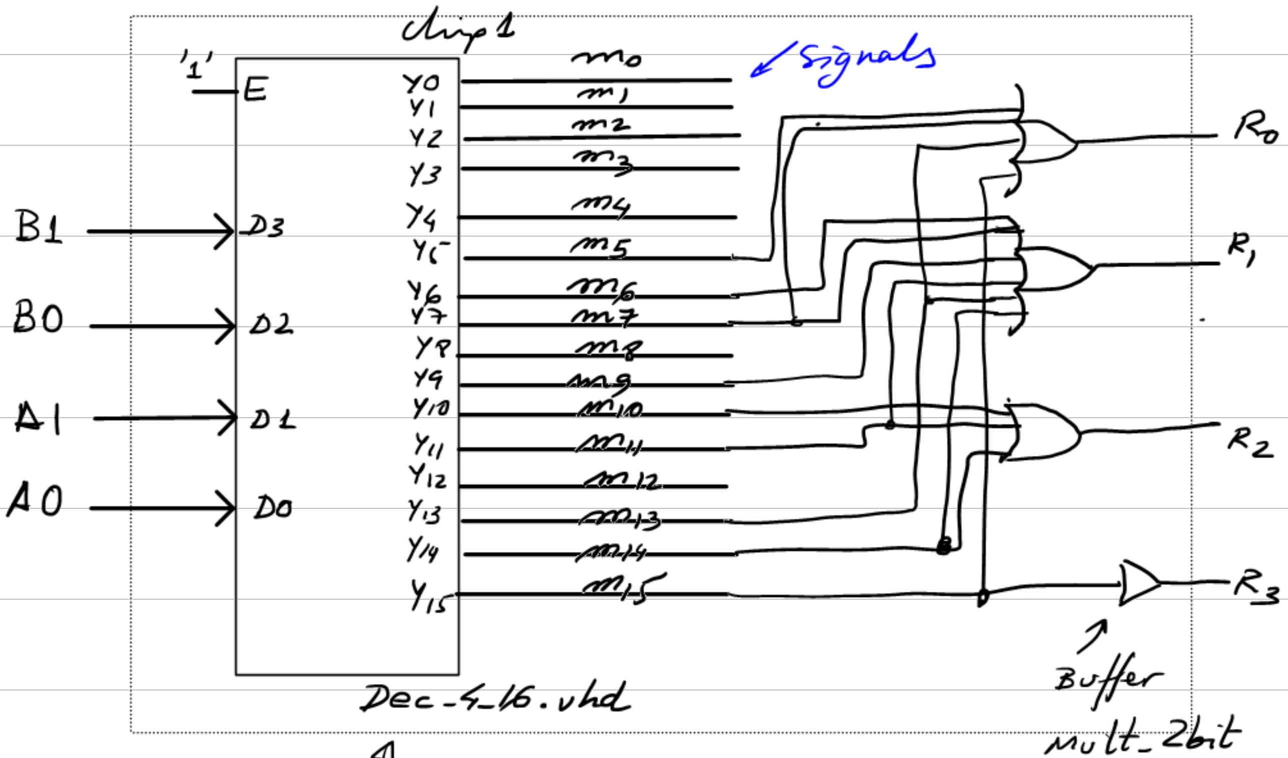
For instance $\rightarrow$ $m_{10} = B_1 \cdot B_0' \cdot A_1 \cdot A_0'$

$\searrow$ 1Ø1Ø

$M_4 = (B_1 + B_0' + A_1 + A_0)$

$\searrow$ Ø1ØØ

3. Method of decoders



chip 1
'1'
E
B1
B0
A1
A0
D3
D2
D1
D0
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7 Y8 Y9 Y10 Y11 Y12 Y13 Y14 Y15
m0 m1 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 m13 m14 m15
signals
R0
R1
R2
R3
Buffer
Dec_4_16.vhd
Mult_2bit

Internally, it can be built using other components of the same kind, like Dec_3_8, or it can be a flat circuit.
Let's suppose it is a flat circuit

The decoder is a circuit which generates all the minterms for the inputs D(3..0), so that we simply have to OR some of them for implementing functions

$R_3 = m_{15}$
$R_2 = \Sigma m(10, 11, 14)$
$R_1 = \Sigma m(6, 7, 9, 11, 13, 14)$
$R_0 = \Sigma m(5, 7, 13, 15)$

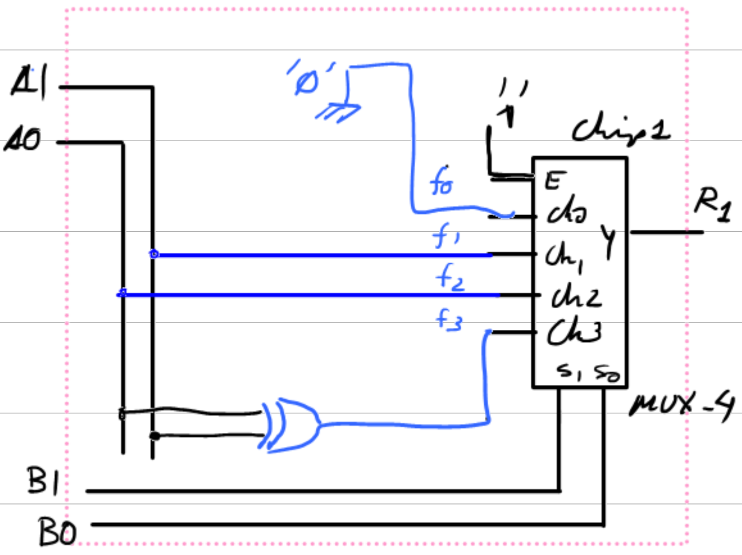
Thus, if the component Dec_4_16 is a flat circuit, the design will consist of 2 VHDL source files.

4. Method of multiplexers using a MUX_4

Here, the good idea is to copy again the truth table so that we can inspect it and section it in order to see which functions has to be connected to the channels

		B_1	B_0	A_1	A_0	R_3	R_2	R_1	R_0
ch0	0	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
	1	Ø	Ø	Ø	1	Ø	Ø	Ø	Ø
	2	Ø	Ø	1	Ø	Ø	Ø	Ø	Ø
	3	Ø	Ø	1	1	Ø	Ø	Ø	Ø
ch1	4	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø
	5	Ø	1	Ø	1	Ø	Ø	Ø	1
	6	Ø	1	1	Ø	Ø	Ø	1	Ø
	7	Ø	1	1	1	Ø	Ø	1	1
ch2	8	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø
	9	1	Ø	Ø	1	Ø	Ø	1	Ø
	10	1	Ø	1	Ø	Ø	1	Ø	Ø
	11	1	Ø	1	1	Ø	1	1	Ø
ch3	12	1	1	Ø	Ø	Ø	Ø	Ø	Ø
	13	1	1	Ø	1	Ø	Ø	1	1
	14	1	1	1	Ø	Ø	1	1	Ø
	15	1	1	1	1	1	Ø	Ø	1

Channel select



Combinational circuit

The partial functions are:

$f_0 = f(A_1, A_0) = $ 'Ø' allways to Ø

$f_1 = f(A_1, A_0) = A_1$ a buffer of the input port

$f_2 = f(A_1, A_0) = A_0$

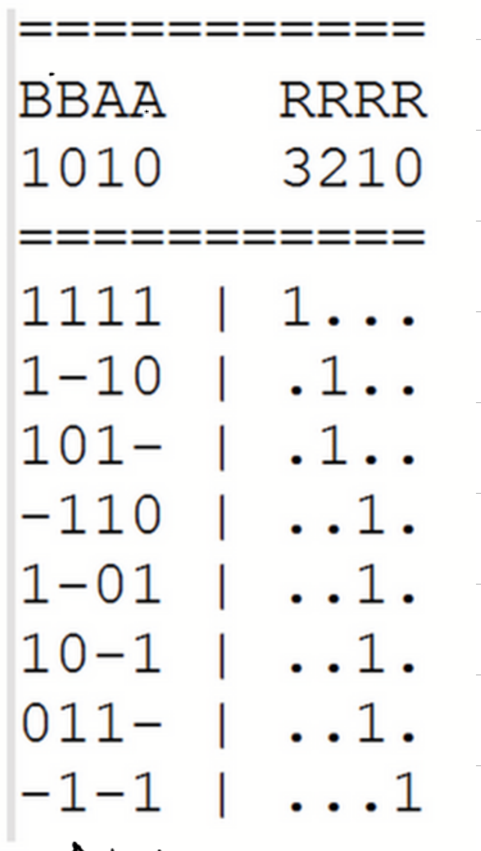
$f_3 = f(A_1, A_0) = A_1 \oplus A_0$

(or $A_1' A_0 + A_1 A_0'$)

(or $(A_1 + A_0)(A_1' + A_0')$)

In the same way, we can implement the other outputs. a MUX is required to solve an output, so, to implement the complete Mult-2bit 4 MUX_4 and some logic gates are required.

5. Flat circuit using logic gates and minimised equations



```
===========
BBAA   RRRR
1010   3210
===========
1111 | 1...
1-10 | .1..
101- | .1..
-110 | ..1.
1-01 | ..1.
10-1 | ..1.
011- | ..1.
-1-1 | ...1
```

B_1 B_0 A_1 A_0

If the minimased output table format comes from a SoP

$R3 = B_1 \cdot B_0 \cdot A_1 \cdot A_0$

$R2 = B_1 \cdot A_1 \cdot A_0' + B_1 \cdot B_0' \cdot A_1$

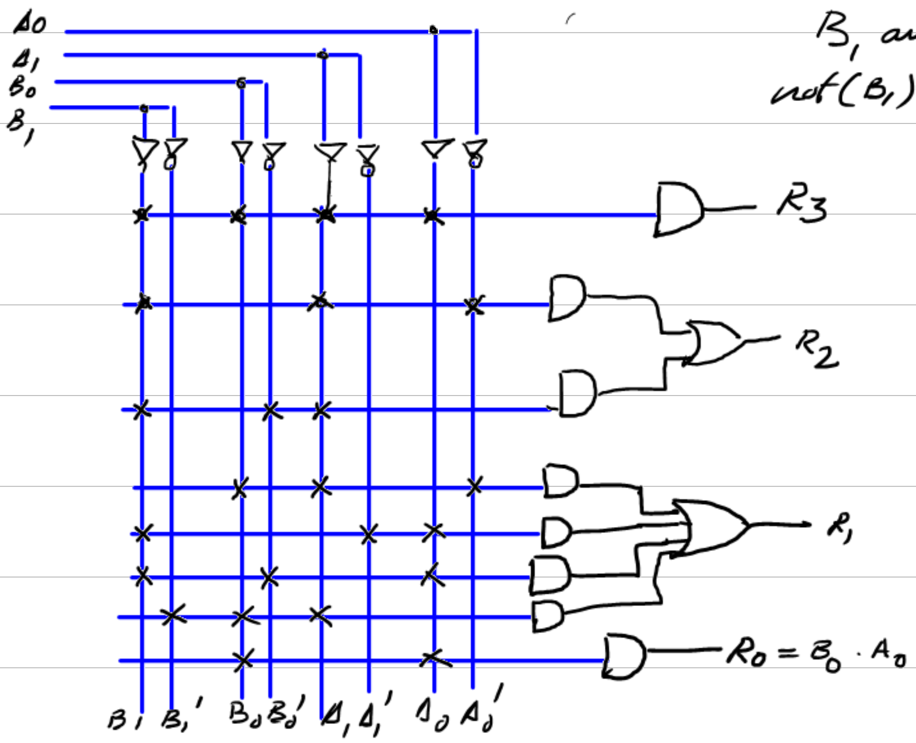
$R_1 = B_0 \cdot A_1 \cdot A_0' + B_1 \cdot A_1' A_0 + B_1 B_0' A_0 + B_1' B_0 \cdot A_1$

$R_0 = B_0 \cdot A_0$

in VHDL

R_0 <= B1 and A_0 ;

R_3 <= B0 and A_1 and not(A_0) or
B1 and not(A_1) and A_0 or
B_1 and not(B_0) and A_0 or
not(B_1) and B0 and A_1 ;



6. Using only-NOR to solve digital circuits

From the Minilog output table:

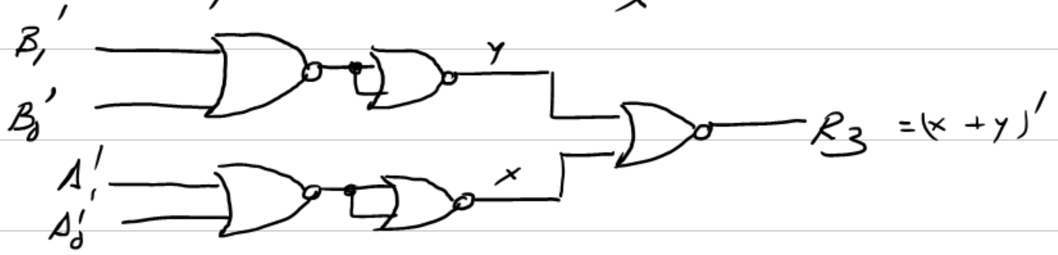
$$R3 = B_1 \cdot B_0 \cdot A_1 \cdot A_0 = (B_1 \cdot B_0 \cdot A_1 \cdot A_0)''$$

$$R_3 = (B_1' + B_0' + A_1' + A_0')'$$

↗

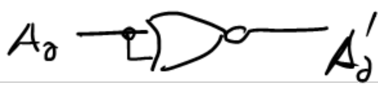
NOR of 4 inputs that can be implemented using NOR of 2 inputs in this way:

$$\left(\underbrace{(B_1' + B_0')''}_{Y} + \underbrace{(A_1' + A_0')''}_{X}\right)' = R_3$$

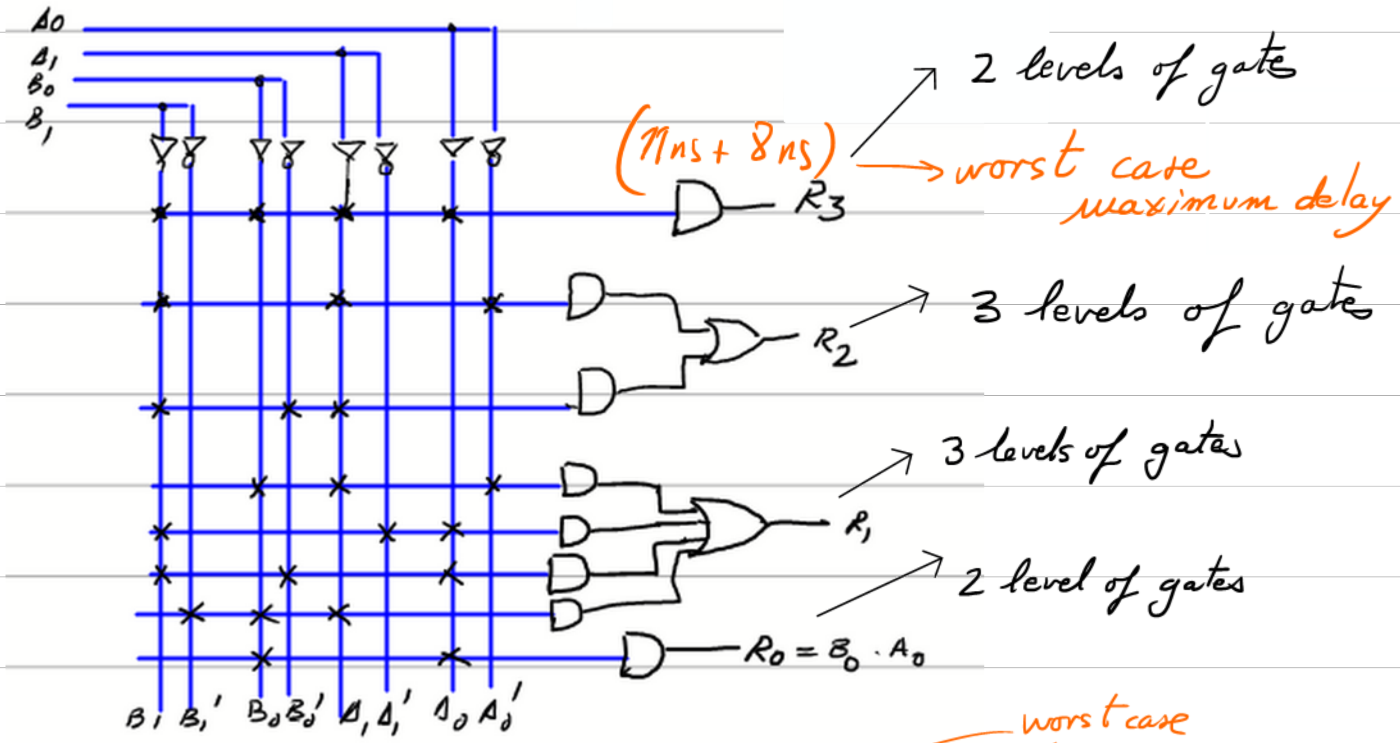


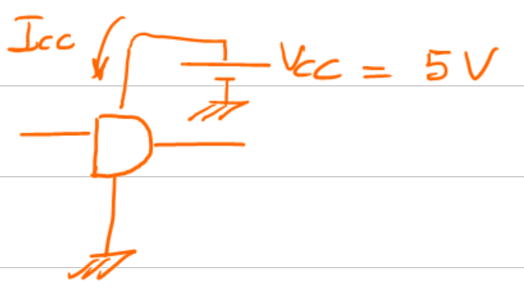
↗

The NOT can be implemented also using NOR-2



7. Calculating the maximum speed of computing and the circuit's power consumption





worst case

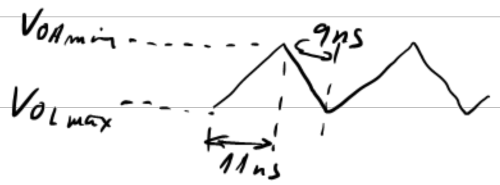
$$\overline{I_{CC}} = \frac{I_{CCH} + I_{CCL}}{2} = \frac{1.1\,mA + 4.2\,mA}{2}$$

$$\overline{I_{CC}} = 2{,}65\,mA$$

$$\text{Power consumption} = 18 \text{ gates} \times \left(5V \cdot 2.65mA\right)_{\text{gate}} =$$

$$P_c = 18 \times 13{,}25\,mW = \underline{238{,}5\ mW}$$

$$\text{Maximum speed of processing is} \leqslant \frac{1}{\underbrace{(3 \text{ levels of gates})(11ns + 8ns)}_{t_{P\,max}}} = \frac{1}{57ns} = \underline{17{,}54MHz}$$



This is the worst case output waveform. For example R2, when switching '1' and '0' at maximum speed.

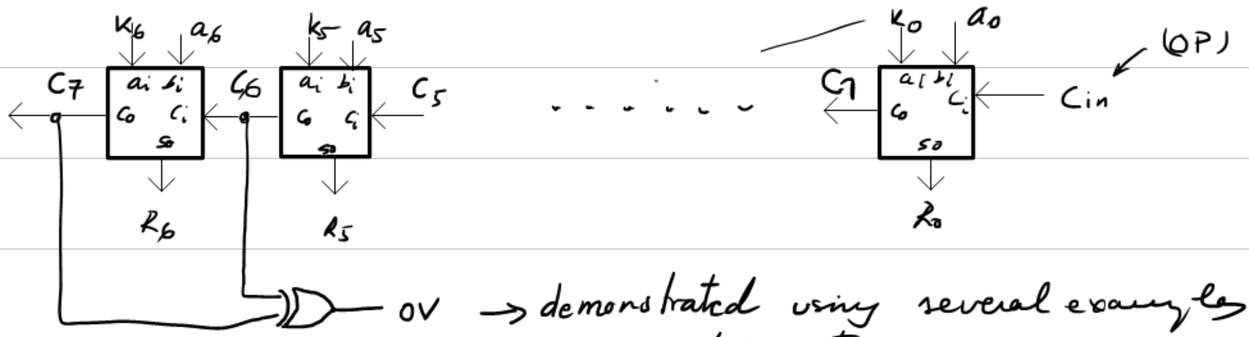
Problem 2

1. The design of a 7-bit adder/subtracter was studied in P4

2. Using 2C and 7 bit the range is $-2^6 \leq Q \leq +2^6-1$

$$-64 \leq Q \leq +\underline{63}$$

The overflow flag can be deduced as the XOR logic function between the last two carries $\left(C_6 \oplus C_7 = OV \right)$ of the 7-bit internal arithmetic adder.



→ demonstrated using several examples in P4 class notes.

3. Let's try several operations:

a) $A = (+39)_{10}$ $\quad B = (\boxed{1}ØØ1Ø1Ø)_{2C}$ ⇒ $\boxed{Ø}11Ø1Ø1$ $+1$; $\boxed{Ø}11Ø11Ø$ → +54

OP = Ø $\quad$ −54

→ The result is −15 and there is no overflow $\quad$ OV = Ø $\quad$ Z = Ø

$C_6 = Ø$

```
  + Ø 1 Ø Ø 1 1 1
    1 Ø Ø 1 Ø 1 Ø
    -------------
    1 1 1 Ø Ø Ø 1
```

$C_7 = Ø$ $\quad$ −15 ⇒ OK!

→ $\boxed{1}ØØ111Ø$ $+1$; $\boxed{Ø}ØØ1111$ → +15

b) $(ØØ1Ø11Ø)_{2C} - (-55)_{10}$ ⇒ A = +22 $\quad$ B = −55 $\quad$ OP = 1 $\quad$ R = A − B

→ The result is +77 → out of range → OV = 1

The computer will solve the subtraction in this way $\quad R = A - B = A + 2C(B)$

+55 = $\boxed{Ø}11Ø111$

```
  1ØØ1ØØØ
       +1
  -------
  1ØØ1ØØ1
```

$C_6 = 1$

```
  Ø Ø 1 Ø 1 1 Ø
  Ø 1 1 Ø 1 1 1
  -------------
  1 Ø Ø 1 1 Ø 1
```

$C_7 = Ø$ $\quad$ → wrong result!

−51 ⇒ OV = 1 $\quad$ → Ø11ØØ1Ø +1 ; Ø11ØØ11 $\quad$ 32+16+3 → +51

c) $(+18)_{10}$ OP = ∅ $B = (\boxed{1}1∅111∅)_{2C} \rightarrow -18$

$\downarrow$

$\boxed{∅}∅1∅∅1∅$

$\downarrow$ $\boxed{∅}∅1∅∅∅1$

$+1$

$\boxed{∅}∅1∅∅1∅ \rightarrow 18$

The result is $R = A + B = 0$

$OV = 0$

$Z = 1$

$C_6 = 1$

Let's see:

$$\begin{array}{rccccccc} + & \boxed{∅} & ∅ & 1 & ∅ & ∅ & 1 & ∅ \\ & \boxed{1} & 1 & ∅ & 1 & 1 & 1 & ∅ \\ \hline C_7 = 1 & \boxed{∅} & ∅ & ∅ & ∅ & ∅ & ∅ & ∅ \end{array} \rightarrow R = 0 \checkmark$$

d) $(-31)_{10} - (\boxed{∅}∅1∅11∅)_{2C} \rightarrow +22$

$\boxed{1}1∅1∅∅1$

$+1$

$\boxed{1}1∅1∅1∅$

$\downarrow$ OP = 1 $\rightarrow R = A + 2C(B)$

+31 $\boxed{∅}∅11111$

$11∅∅∅∅∅$

$+1$

$\rightarrow -31$ $\boxed{1}1∅∅∅∅1$

The result is $-31 - (+22) = -53$

$OV = ∅$

$Z = ∅$

$C_6 = 1$

$$\begin{array}{rccccccc} & \boxed{1} & 1 & ∅ & ∅ & ∅ & ∅ & 1 \\ + & 1 & 1 & ∅ & 1 & ∅ & 1 & ∅ \\ \hline C_7 = 1 & \boxed{1} & ∅ & ∅ & 1 & ∅ & 1 & 1 \end{array}$$

$\swarrow$ −53 OK!

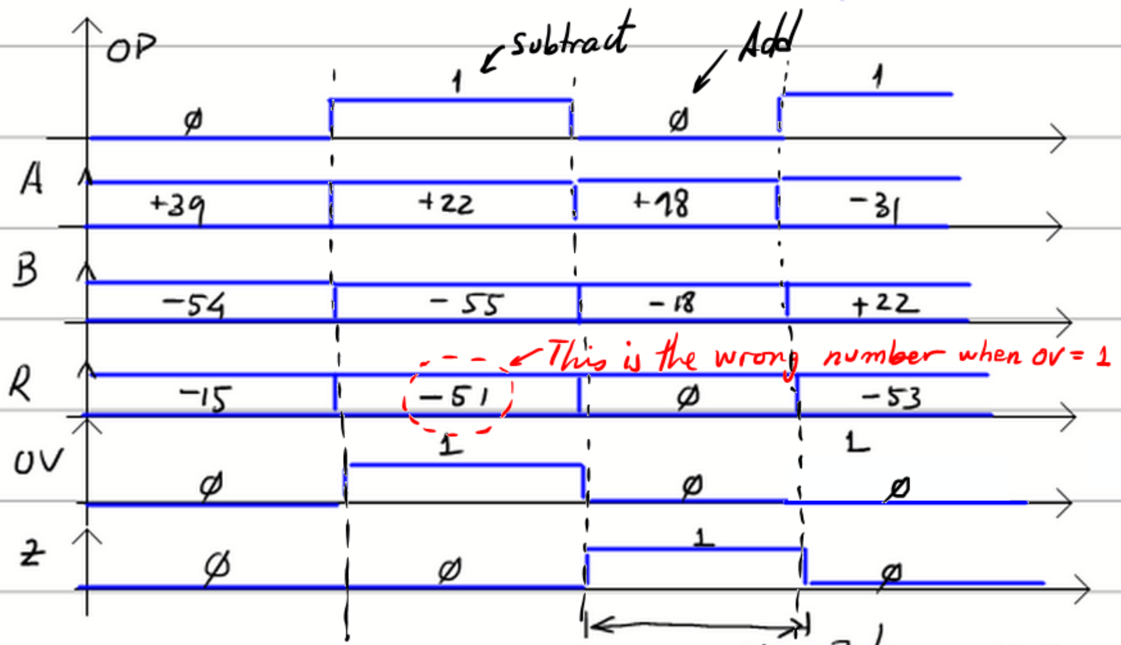
Negative $\Rightarrow$ $∅11∅1∅∅$

$+1$

$\boxed{∅}11∅1∅1 \rightarrow +53$

$32+16+5$

4. Timing diagram and VHDL test bench



Min_Pulse = 10.5 μs

There are 15 inputs, so $\to$ run for $\to 2^{15} \cdot 10.5\,\mu s = \underline{345\,ms}$

A+B+OP ↑

To translate the stimulus vectors to VHDL →

for instance:

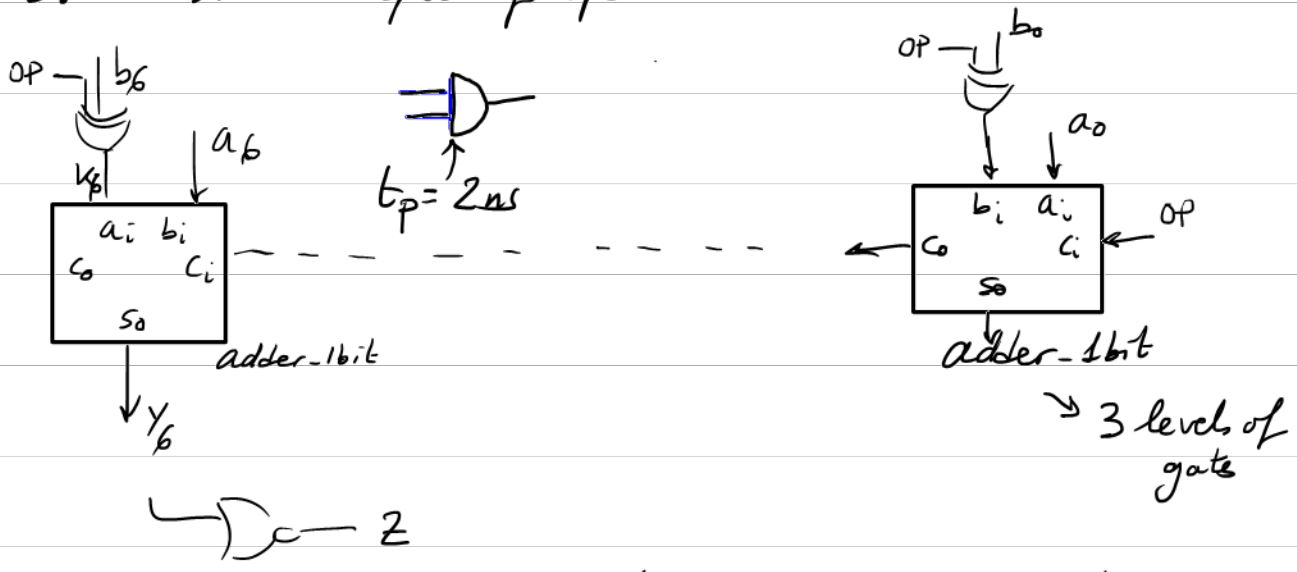
```
A <= "Ø1ØØ111";
B <= "1ØØ1Ø1Ø";
OP <= 'Ø';
wait for Min_Pulse;
```

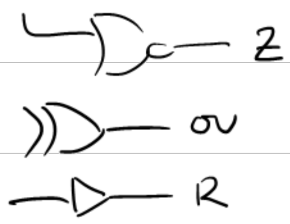
etc...

So, it is necessary a test time of 345 ms in case of liking to check and verify all the input vectors

→ This "mechanical" process can be carried out automatically using VHDL test bench features like ASSERT

5. Maximum speed of operation





The total number of levels of gate (assuming a ripple carry 7-bit adder) is

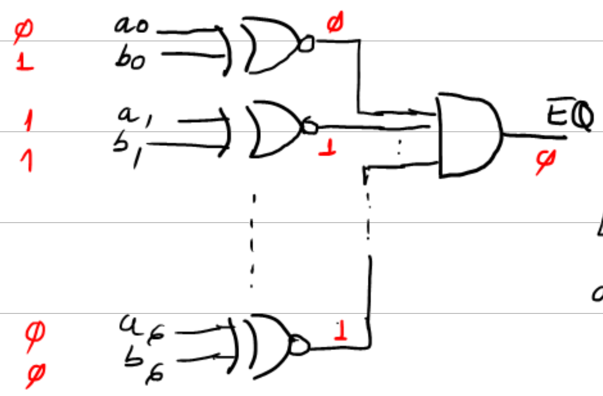
$$\text{XOR} + \underset{\uparrow \text{ 3 levels of gates}}{(\text{adder\_1bit})} \cdot 7 + \left\{ \begin{matrix} \text{XOR} \\ \text{OV} \end{matrix} \right.$$

$$t_p + 7 \cdot (3 \cdot t_p) + t_p = \underset{\uparrow 2ns}{23\, t_p} \doteq 46ns$$

$$f_{max} \leq \frac{1}{46ns} = \underline{21,74\ MHz}$$

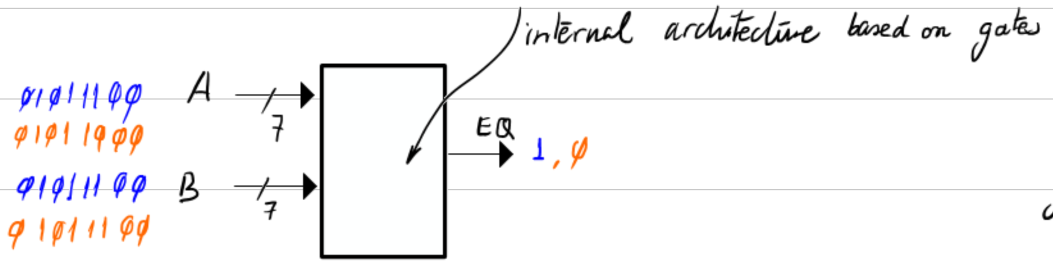
≈ 21,7 millions of operations per second

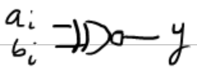
6. The EQ flag detector can be implemented using NXOR



a_i	b_i	y
Ø	Ø	1
1	Ø	Ø
Ø	1	Ø
1	1	1

EQ = 1 only when all the A bits are the same as all the B bits





$$y = a_i' \cdot b_i' + a_i b_i$$

$$y = (a_i \oplus b_i)'$$

$$y = (a_i' + b_i) \cdot (a_i + b_i')$$