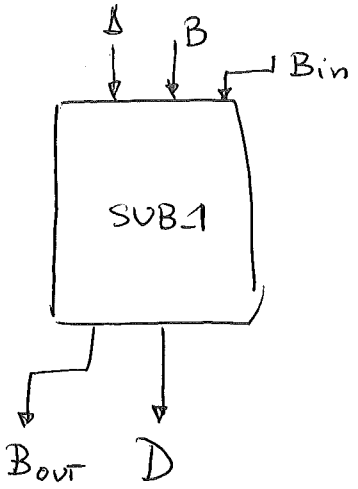
This is an example of some ideas on the solution



One-bit subtractor with borrow in and out

① Equations using only NAND gate

A	B	Bin	Bout	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$A - B \leftarrow Bin$$
$$B_{out} \leftarrow D$$

For example:

$$\begin{array}{r} 0 \\ -\ 0 \leftarrow 0 \\ \hline 0 \leftarrow 0 \end{array} \qquad \begin{array}{r} 0 \\ -\ 1 \leftarrow 1 \\ \hline 1 \leftarrow 0 \end{array} \qquad \begin{array}{r} 1 \\ -\ 0 \leftarrow 0 \\ \hline 0 \leftarrow 1 \end{array} \qquad \begin{array}{r} 1 \\ -\ 1 \leftarrow 1 \\ \hline 1 \leftarrow 1 \end{array}$$

Let's understand where the truth table came from applying some particular vectors

In this way, using blocks like this cascadable 1-bit substractor we can implement larger circuit like the SUB_8

$$\begin{array}{r} 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0 \\ -\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 1 \leftarrow 0 \\ \hline 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1 \end{array} \qquad \begin{array}{r} 238 \\ -\ 23 \\ \hline 215 \end{array}$$

After this initial discussion, we can plan the solution. In ① we are asked to obtain a gate-level circuit using only NAND (structural → equations)
In ② we have to develop the truth table itself in VHDL, so it's a behavioral description

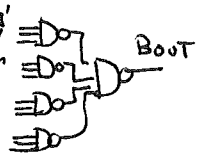
Let's write the canonical expression firstly:

$$B_{OUT} = f(A, B, B_{in}) = m_1 + m_2 + m_3 + m_7$$

$$B_{OUT} = \overset{0\,0\,1}{A'B'B_{in}} + \overset{0\,1\,0}{A'B\,B'_{in}} + \overset{0\,1\,1}{A'B\,B_{in}} + \overset{1\,1\,1}{A\,B\,B_{in}}$$

So, if we have to use only NAND: $(x+y)'' = (x'\cdot y')'$

$$B_{OUT} = \Big((A'B'B_{in})' \cdot (A'B\,B'_{in})' (A'\,B\,B_{in})' (A\,B\,B_{in})' \Big)'$$



You can do the same procedure with the other output D.
Or, instead, you may represent the function using the maxterms:

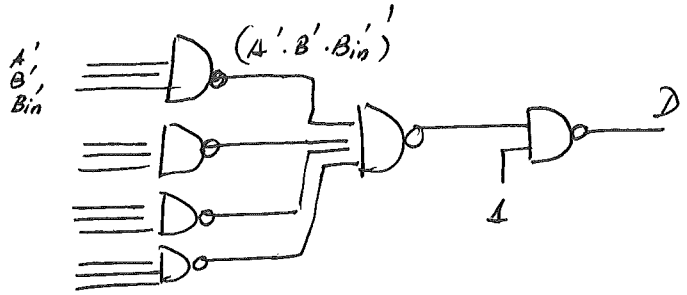
$$D = M_0 \cdot M_4 \cdot M_5 \cdot M_6$$

And, so, in order to have only NAND $\Rightarrow (x\cdot y)'' = ((x\cdot y)')'$

$$D = \overset{0\,0\,0}{(A+B+B_{in})} \cdot \overset{1\,0\,0}{(A'+B+B_{in})} \cdot \overset{1\,0\,1}{(A'+B+B'_{in})} \cdot \overset{1\,1\,0}{(A'+B'+B_{in})}$$

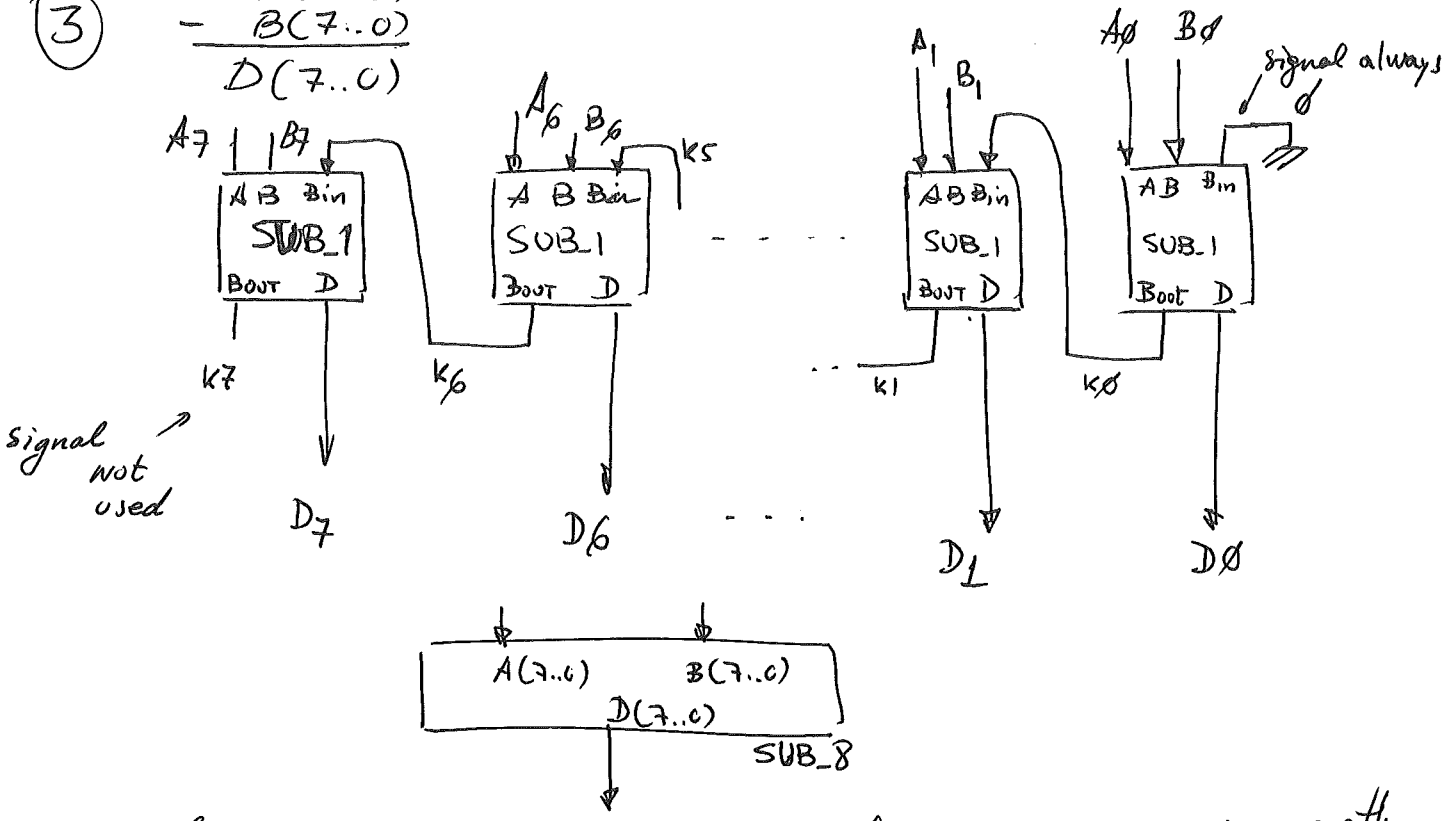
$$D = \Big((A+B+B_{in})'' \cdot (A'+B+B_{in})'' \cdot (A'+B+B'_{in})'' (A'+B'+B_{in})'' \Big)''$$

$$\Big(\big((A'\cdot B'\cdot B'_{in})' \cdot (A'\,B'\,B_{in})' (A'\,B\,B'_{in})' \cdot (A\,B\,B'_{in})' \big)' \Big)'$$



(3)

$$\begin{array}{r} A(7..0) \\ -\ B(7..0) \\ \hline D(7..0) \end{array}$$



The method, having designed first the SUB_1 is pretty much the same of the 8-bit binary adder

(4) A zero flag is generated when the result is 0. So, it is a special function like:

D7 D6 ... D1 D0	Z
0 0 0 0	1
0 0 0 1	0
⋮	⋮
1 1 ... 1 1	0

which has only one minterm

$$Z = f(D) = m_0 = D_7' \cdot D_6' \cdots D_0'$$

or: $(m_0)'' = (D_7 + D_6 + \cdots + D_0)'$

using a single NOR

Let's see which is the result when having "negative" numbers:

$$\begin{array}{r} 7 \\ -\ 9 \\ \hline -2 \end{array} \Rightarrow \begin{array}{r} 00000111 \\ -\ 00001001 \\ \hline 11111110 \end{array}$$

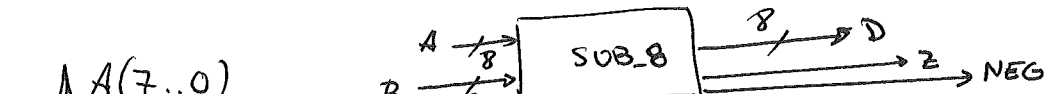
(1) $B_{OUT_7} = 1$

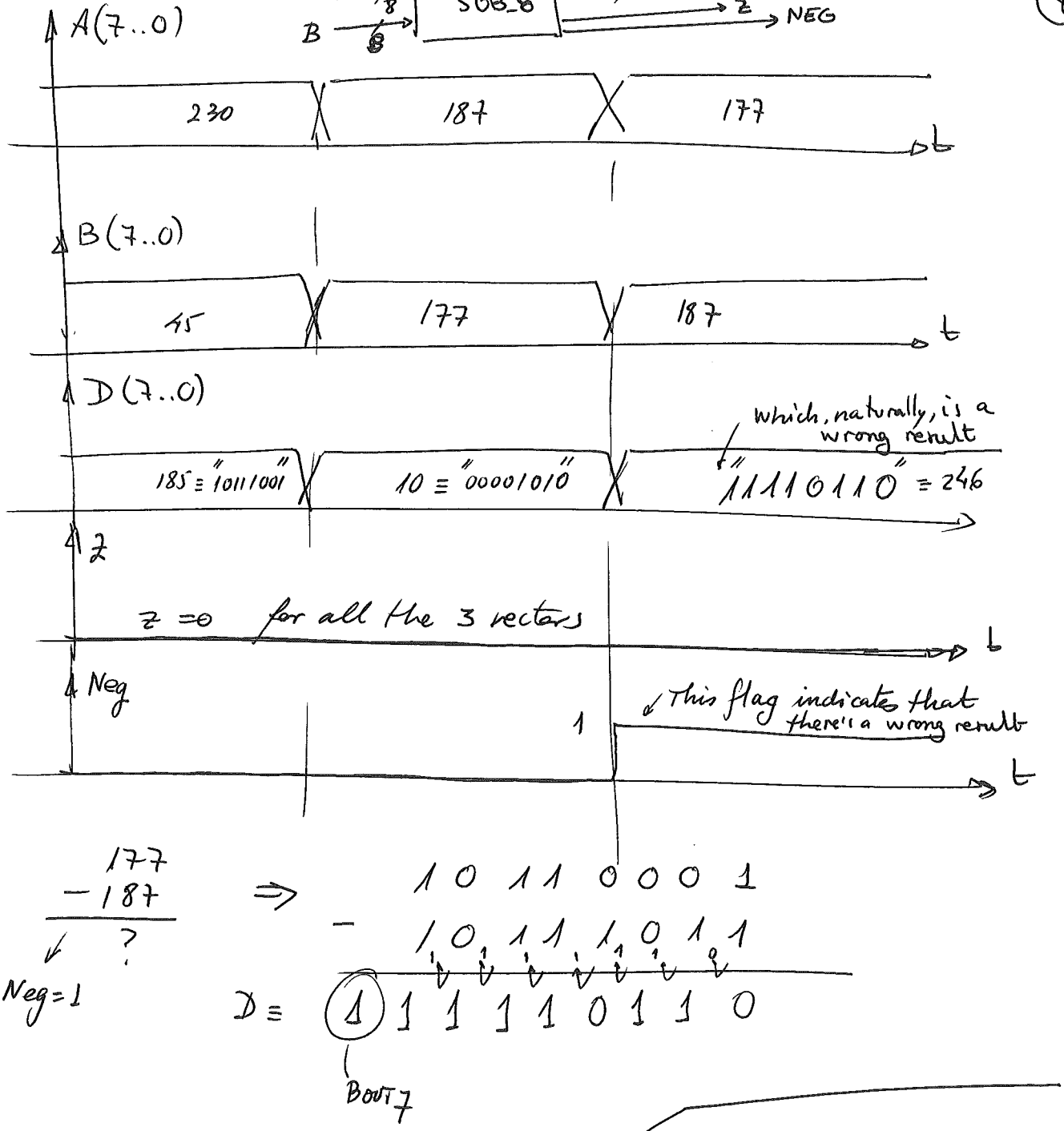
?

$$\begin{array}{r} 13 \\ -15 \end{array} \quad \begin{array}{r} 00001101 \\ -\ 00001111 \\ \hline 11111110 \end{array}$$

(1)

So, $\Rightarrow$ $Neg = B_{OUT_7}$ may be used as a flag.





$$\begin{array}{r} 177 \\ -187 \\ \hline \end{array} \Rightarrow \begin{array}{r} 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1 \\ -\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1 \\ \hline \end{array}$$

$Neg = 1$

$D \equiv (1)\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0$

$BOUT_7$