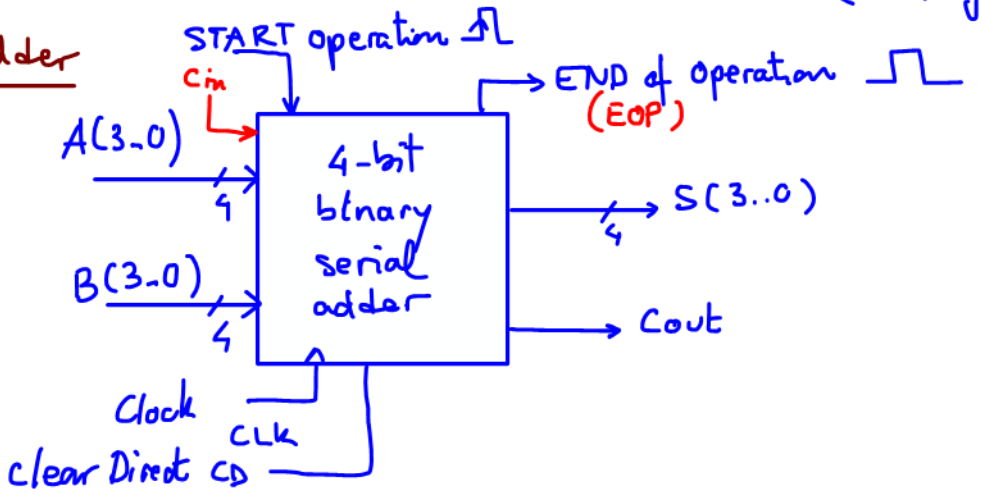
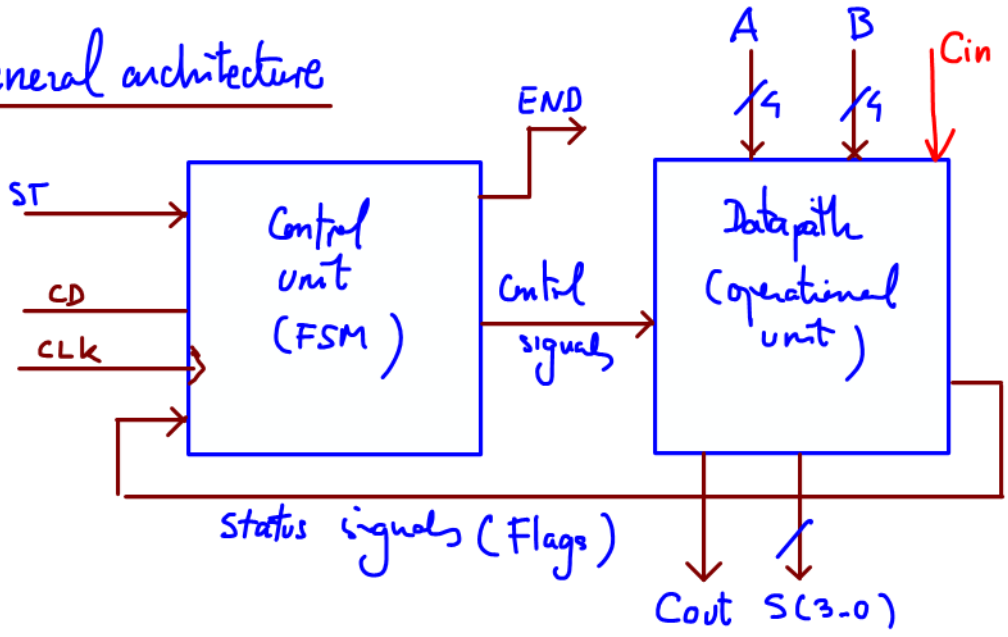
EX3 a dedicated processor / advanced digital system
a control circuit for a digital system / digital processor (data/signal)

A serial adder



It will be planned for 4 bit operations, but can be easily infered that it is enhanced in a simple way to perform binary additions for larger numbers

General architecture

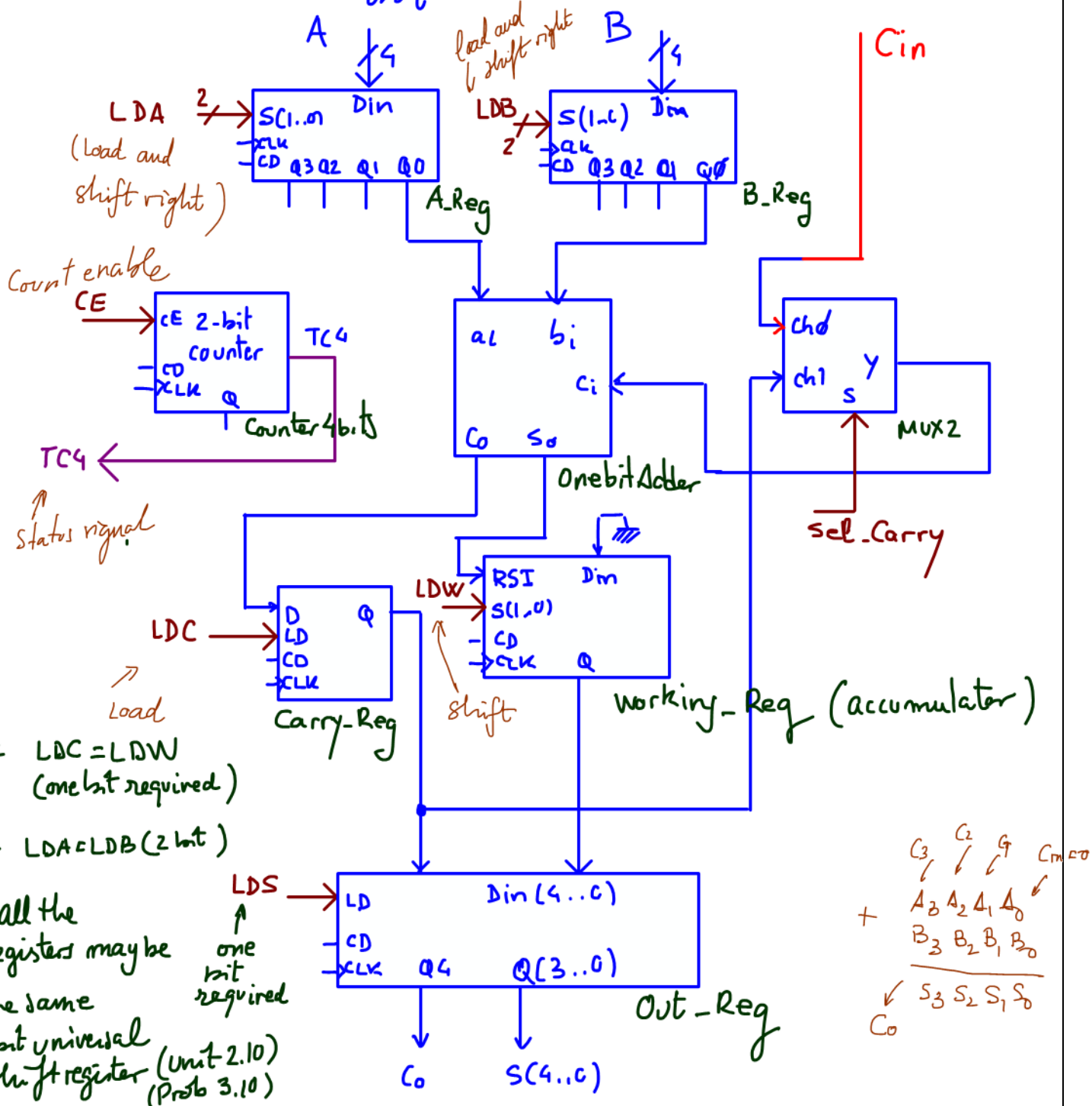


This is the "brain"
the control unit to "sequence"
the operation's algorithm
by means of a state diagram
or a microprogram

Here the information "data" is processed (operated)
⇒ Datapath

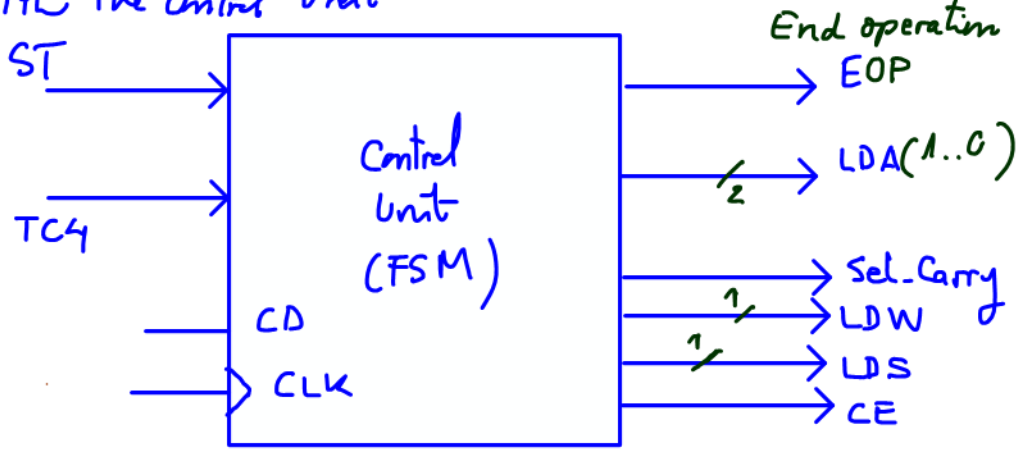
2. Planning

Let's design firstly the datapath component, and secondly the control unit

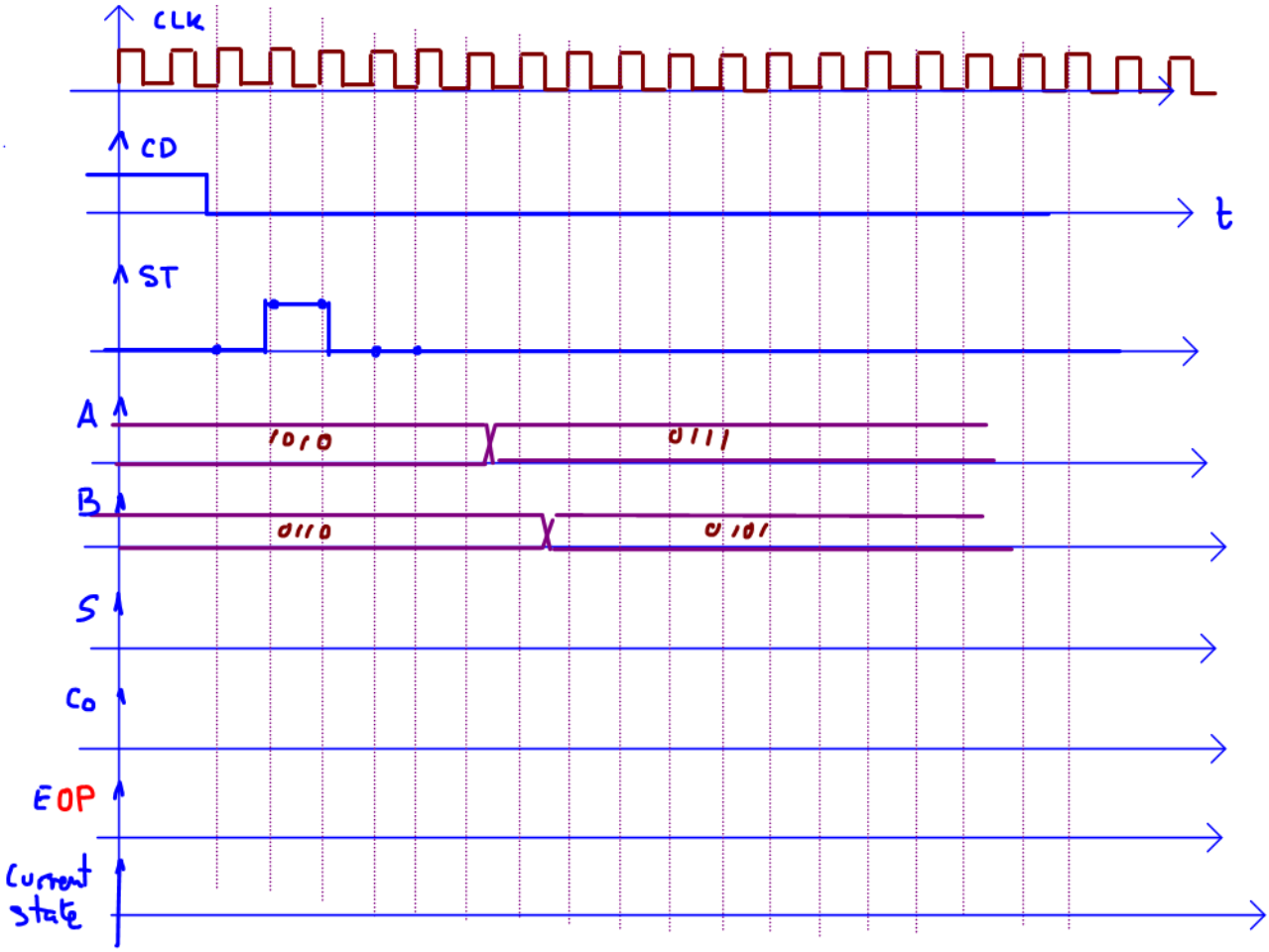


* Resources: 1) 1, 4 or 5-bit data and shift registers (can we use all the time the same?) ✓ Yes!
2) one bit adder
3) MUX2 to init and propagate the carry
↳ or a MUX4 used as a MUX2 !!

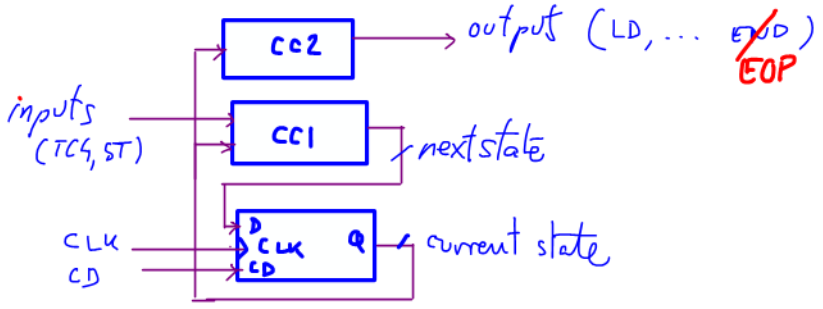
Once all the control lines for the datapath are set, we can proceed with the control unit



Before defining a state diagram, better to solve an example timing diagram



The FSM architecture



Let's infer an example of state to sequence the operation. However, it is preferable to analyse how an example operation will be carried out in this datapath

$A = 1010$ ($A_3 A_2 A_1 A_0$) (10)

$B = 0111$ ($B_3 B_2 B_1 B_0$) (7)

17

$S = 0001$, $C_{in} = 0$ → $C_{out} = 1$

(A) Idle

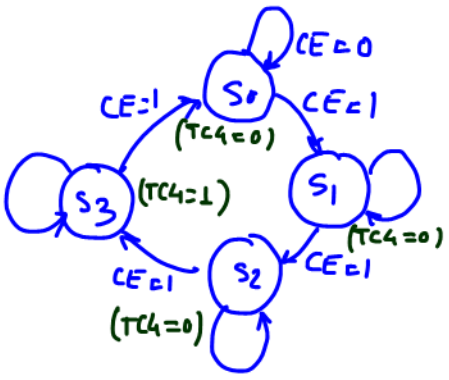
(B) Load operands

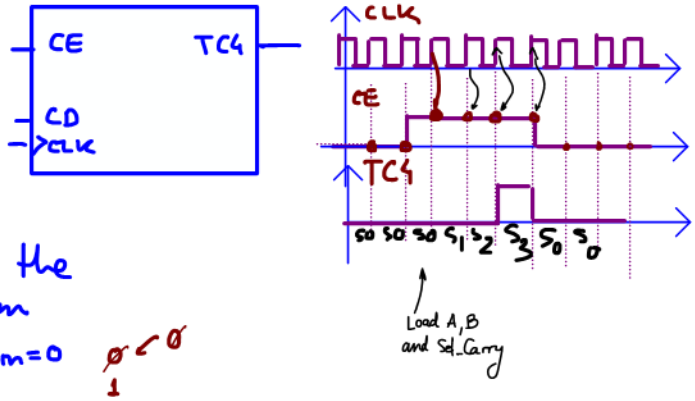
A_Reg = 1010

B_Reg = 0111

Sel_Carry = 0 (let's begin with $C_{in} = 0$)

CE = 1 ⇒ It will imply to run a 4-state counter, so that TC4 = 1 when Q = 11





(C) The combinational circuit has the result of the first addition

$A_0 + B_0$ ← $C_{in} = 0$, S_0, C_{out}

0 ← 0, 1, 0 ← 1

So, we have to 'save' the partial result of S_0 and C_{out} and make possible that the carry be available for the next addition

LDW = '01' (Right shift)

RSI = S_0 → [S_0 | x | x | x]

C_0 → LDC = 1 → [0] (C_{out}) (Load Carry)

Sel_Carry = 1 (so that it is available as C_{in} for the next operation thanks to the MUX2)

(D) shift A, shift B ⇒ Sel_Carry = 1

$A_1 + B_1$ ← C_0, S_1, C_1

1 + 1 ← 0, 0, 1 (C_1, S_1)

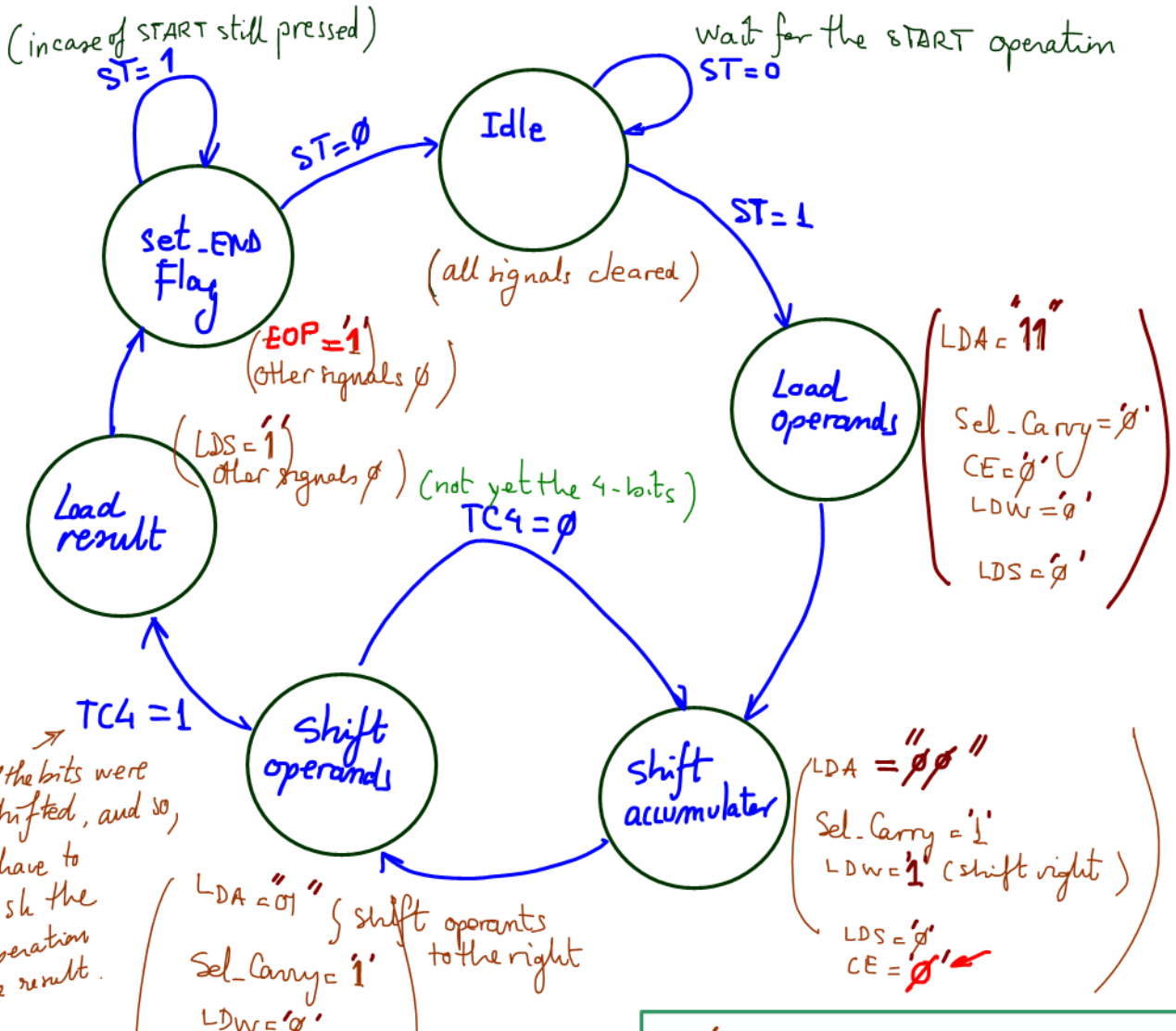
So, now, we have to repeat (C) and (D) until all the bit are shifted and added. This is why we have the 2-bit counter. After 3 turns, TC4 = 1 and the operation is finished. The only task now is to load the final result register of 5 bits and signal the END flag to indicate that the operation have finished, and a new one can be carried out.

(E) Load Result_Register LDS = 1

(F) → Set the bit END = 1 to indicate with a flag that the operation is finished (EOP)

(wrong name!!! ⇒ End of operation)

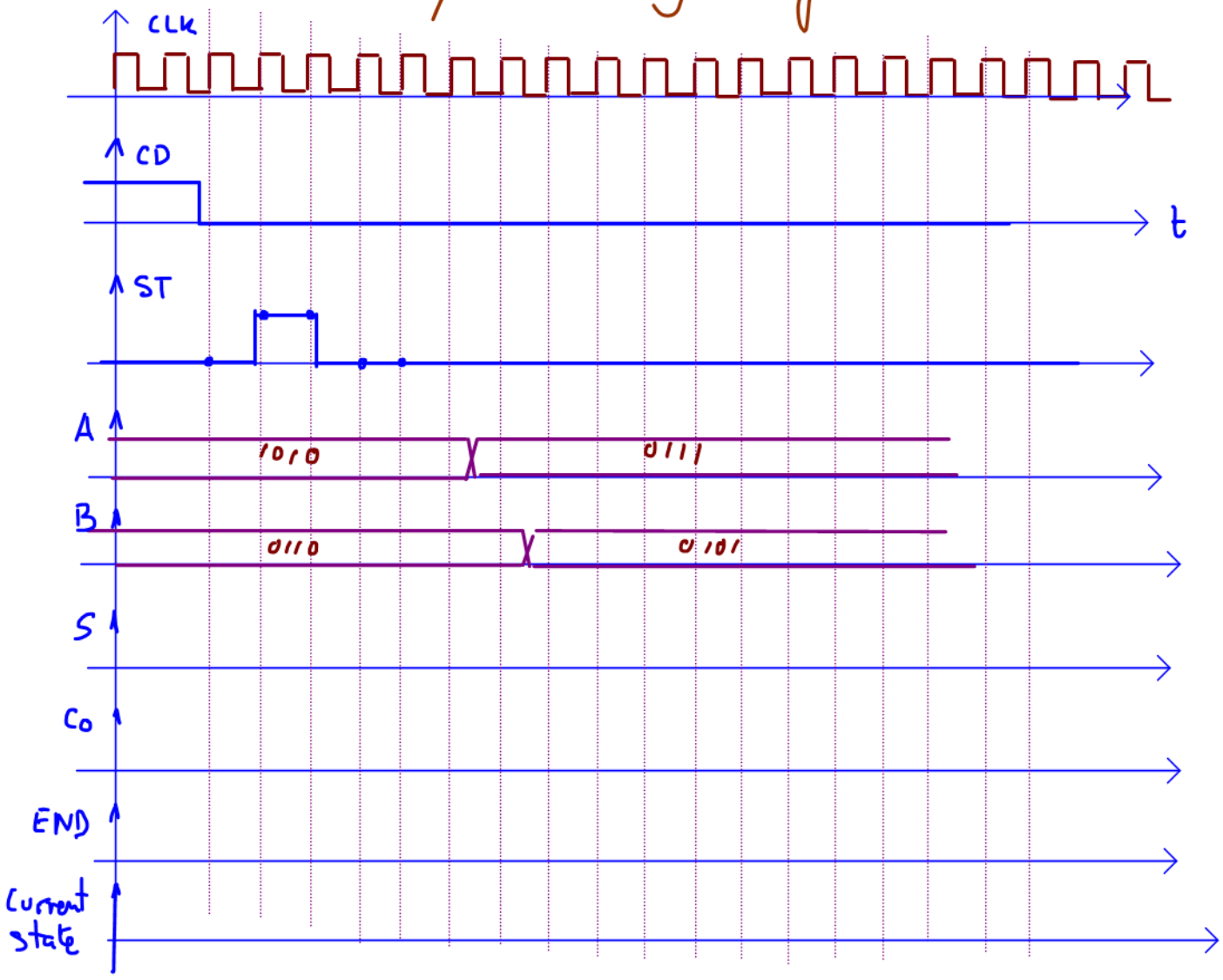
(Which can also be done in (E))



Now, the system can be developed and tested, so that "bugs" can be found and corrected

* Which is the operation speed of this system?

* Test the state diagram trying to complete this example timing diagram:



⇒ Estudy this notes and make your own, so that you really comprehends the way the system works and how it is engineered.

⇒ There are other examples like this dedicated processor on the web to perform different operations and processes with digital informations (Chapter III)