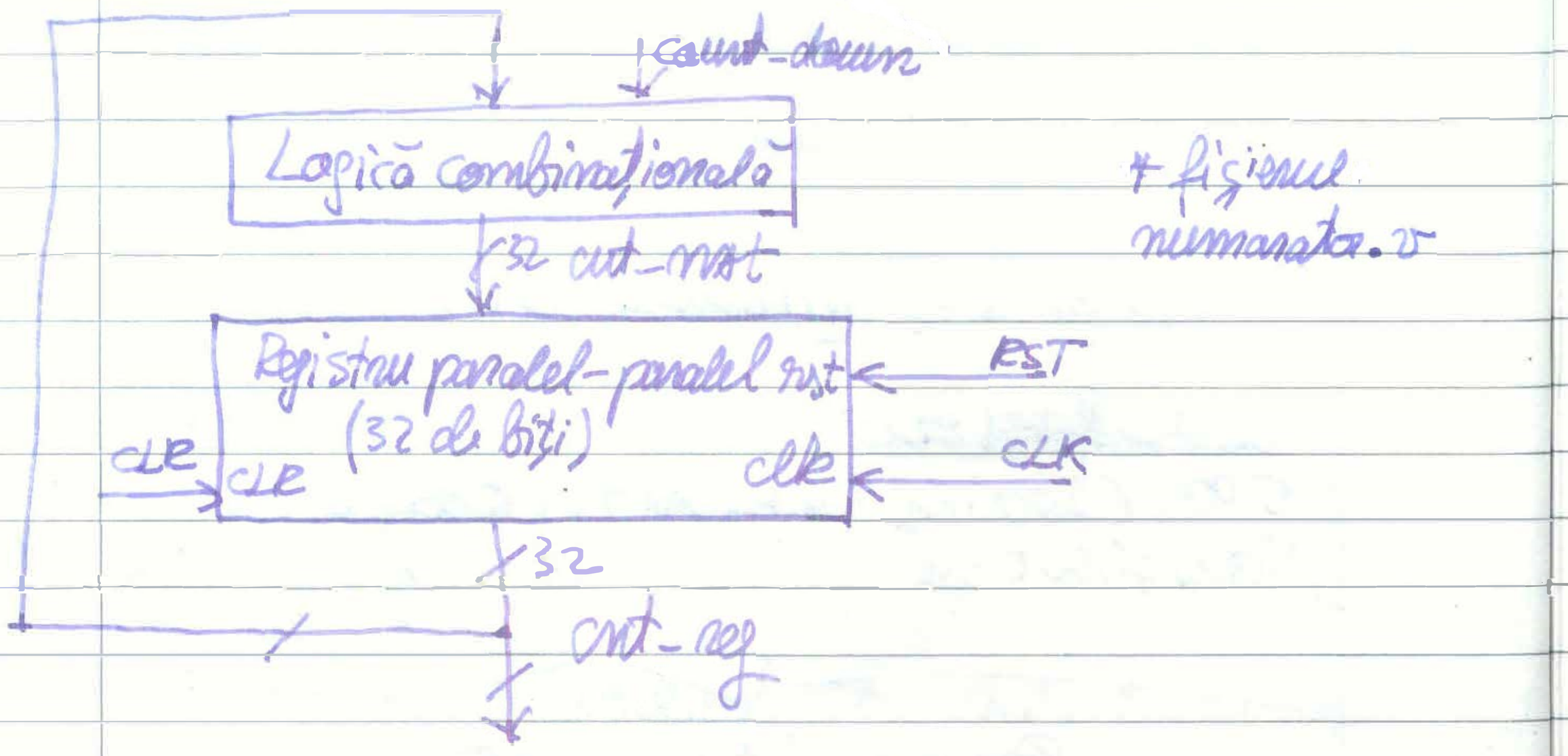
Assignment L

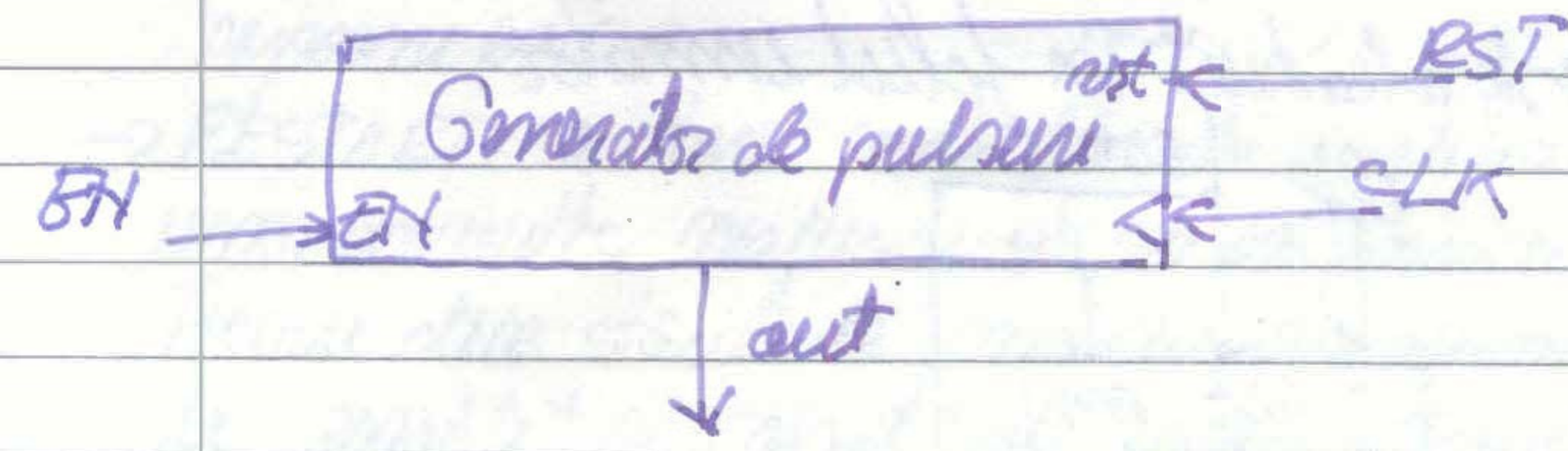
Vom folosi:
→ un numărător proiectat conform principiilor RTL:



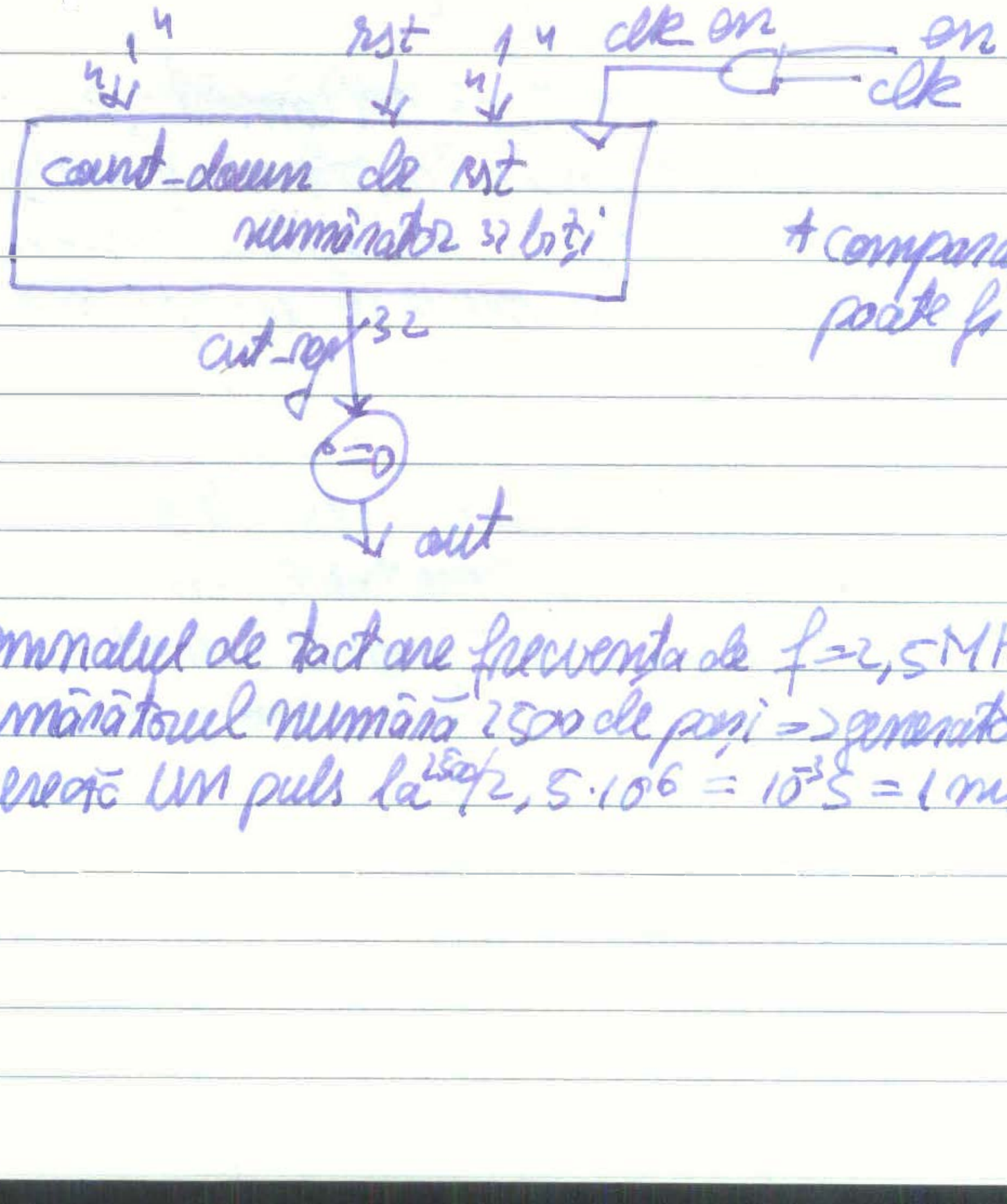
Descriere: → count_down: când este activ numără descrescător, iar când este inactiv numără crescător (activ pe '1')
→ RST: semnal de reset asincron (activ pe '0')
→ CLR: semnal de clear sincron (activ pe '0')
→ CLK: semnal de tact
→ registrul este construit din 32 de bistabile sincrone, comută la frontul crescător al semnalului de tact
→ în cazul nostru numărătorul are ca valoare de preîncărcare sau valoare limită superioară 2500 (numără 2500 de pași)

---> un generator de pulsuri, care este construit folosind circuitul anterior (numărătorul)

↑ fișierul: semnale.v



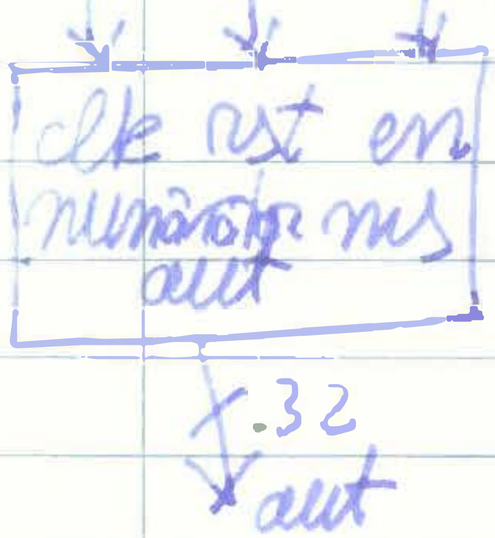
Semnale: → semnal de tact (CLK): oscilează pe frontul crescător
→ reset sincron (RST): activ în „0"
→ semnal de activare (EN): activ în „1"



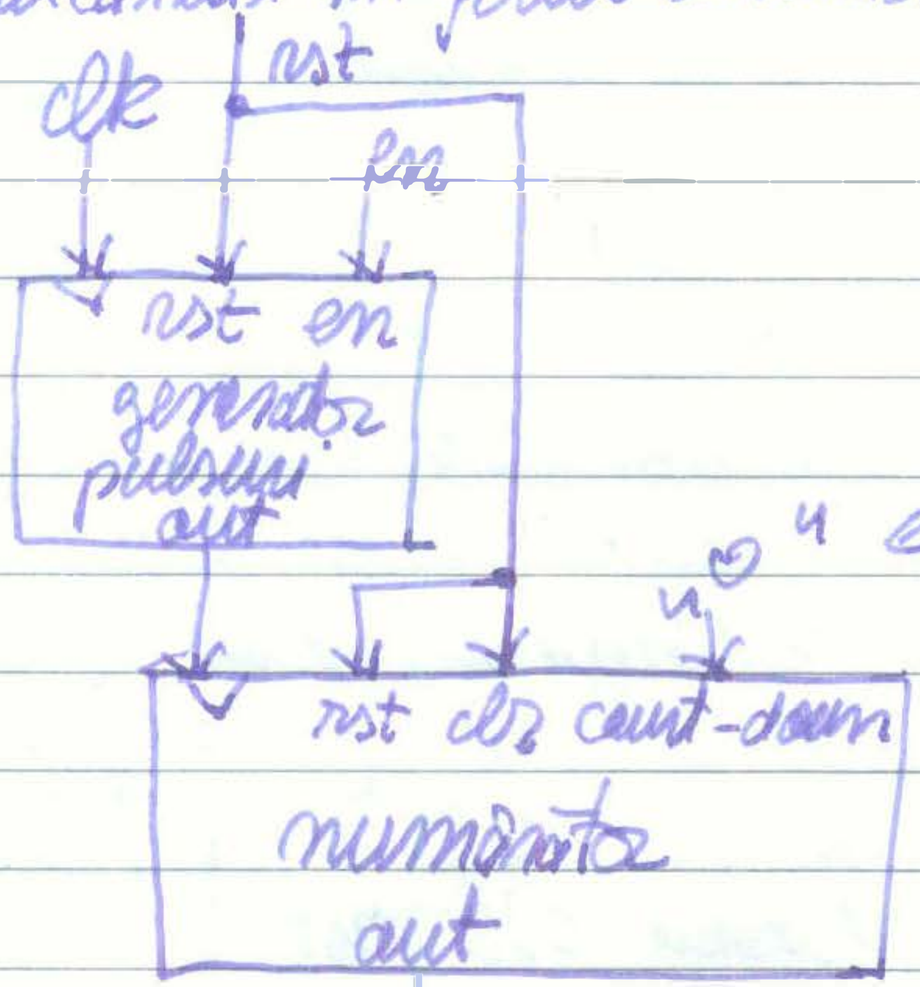
↑ comparatorul (=0) poate fi o poartă NOR.

→ semnalul de tact are frecvența de $f = 2,5 MHz$, deci un numărătorul numără 2500 de pași => generatorul de pulsuri generează un puls la $\frac{2500}{2,5 \cdot 10^6} = 10^{-3} s = 1 ms$.

→ un numărător de milisecunde alcătuit din cele două circuite descrise anterior: un numărător respectiv un generator de pulsuri (care va juca rolul generatorului semnalului de tact cu perioada de T=1ms)



este alcătuit din felul următor:



* fișierul: ms.v

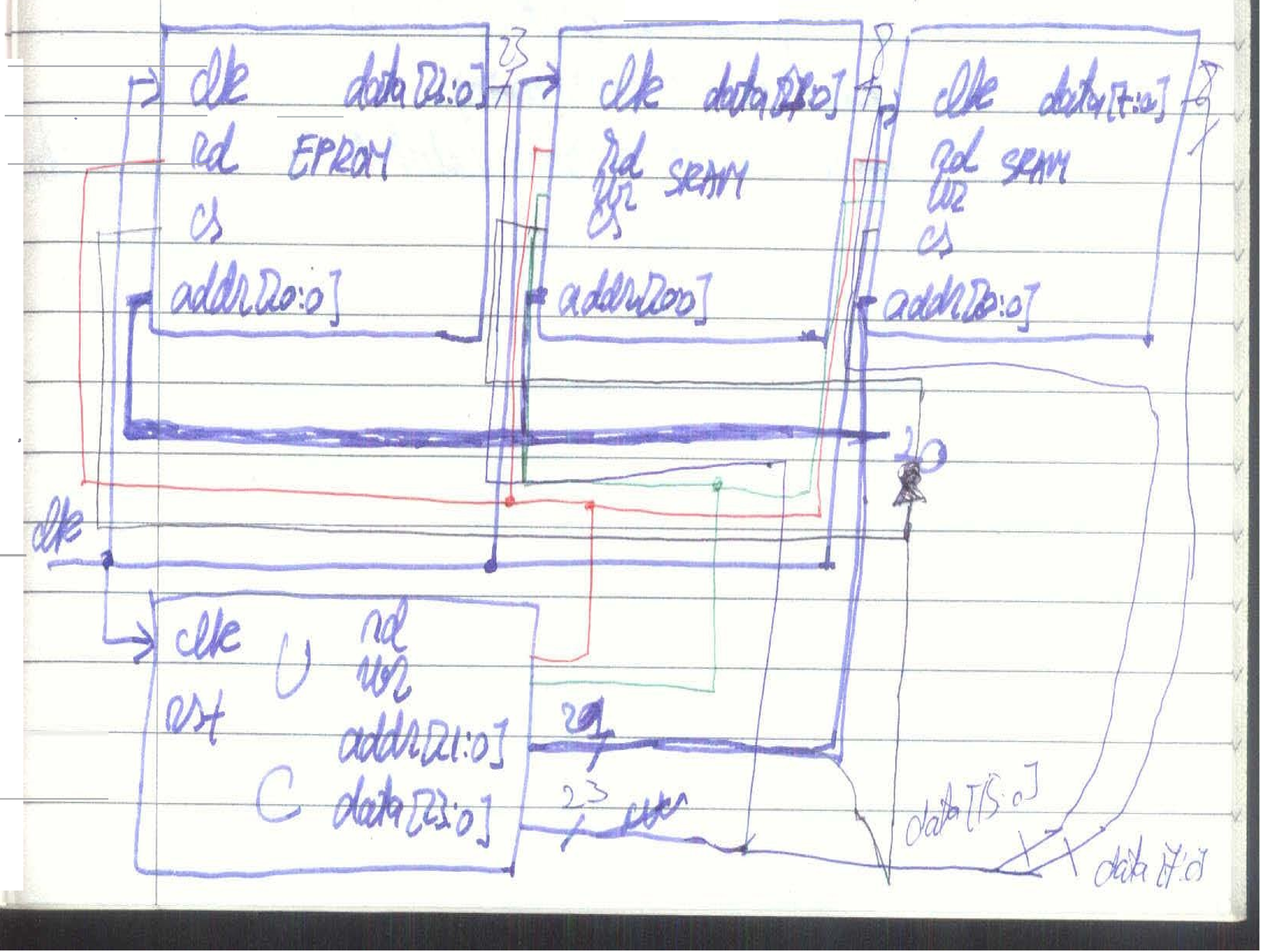
→ circuitului anterior îi vom atașa un convertor BCD, acesta va avea câte câte 4 biți (reprezentare BCD) pentru cifrele: miilor, sutelor, zecilor și unităților. Presupunem că vom număra pe mai mulți de 10 biți (milisecunde).

* fișierul: bcd.v

→ circuitul anterior poate fi comandat de un automat Moore (în cazul nostru) cu următoarele stări:
START, PAUSE, STOP, RESTART, având intrările:
nop, start, pause, stop.
(00) (01) (10) (11)

->avem două tipuri de memorie: EPROM (în cazul nostru va stoca instrucțiunile și va putea fi doar citită, existând în fișierul eprom.v un ciclu inițial de scriere) și SRAM (care va conține datele, implementarea fiind în fișierul sram.v).
->EPROM: are semnale de tact, read, chip select, address și data, mai concret: este activă la frontul crescător al semnalului de tact operația read: memoria situează pe magistrala de date conținutul de la adresa de memorie address, altfel o situează în impedanță ridicată, cuvântul are dimensiunea de 16 biți, iar dimensiunea memoriei este de 2^{21} (are 21 de biți de adresă)
->SRAM: are semnal de tact, al cărui front crescător declanșează operațiile, semnal de read care atunci când este activ face ca memoria să situeze pe magistrala de date (de doar 8 biți aici) cuvântul stocat la adresa address (pe 21 de biți), iar în cazul în care read este inactiv, dacă write este activ, magistrala de date devine în impedanță ridicată pentru a putea primi date pe aceasta, care vor fi scrise la adresa de memorie address.

-> unitate centrală care dispune de semnale de read, write, tact, reset asincron, magistrală de date (24 de biți) și magistrală de adrese (22 de biți), precum și semnale de comunicare cu top module-ul ce conține automatul Moore, alu-in pentru oferirea intrărilor automatului, alu-out, pentru oferirea primirea ieșirilor acestuia automat, dar și alu-clk pentru semnalul de tact al automatului.
->decodificarea memoriilor este trivială, completă, LSB-ul de pe magistrala de adrese va intra în CS-urile memoriilor, când A0 este activ va fi selectată memoria de cod (EPROM), în cest caz cea de date (vom folosi 2 în paralel, întrucât ieșirea automatului este pe 16 biți - 4 cifre BCD - iar magistrala de date a fiecărei memorii SRAM este de doar 8 biți).
->unitatea centrală este de tip von Neumann (non-pipeline) având 5 stagii, care vor fi descrise mai amănunțit ulterior.


clk
rd
cs
addr[20:0]
data[23:0]
EPROM
clk
rd
cs
addr[20:0]
SRAM
clk
rd
cs
addr[20:0]
SRAM
clk
rst
rd
wr
addr[21:0]
data[23:0]
20
23
data[15:0]
data[7:0]

Stările unității centrale:

IF: AR ← PC.1 ; pentru decodificare adăugăm acel „1"
    WR ← 0 ; dezactivăm scrierea, în cazul în care a fost activată
ID: PC ← PC+1 ; incrementăm numărătorul de program
    RD ← 1 ; negăm o citire
DT: DR ← data_bus ; preluăm datele
Ex: alu_clk ← 1 ; activăm tactul pentru automat
    IR ← DR[23:2] ; instrucțiunea reprezintă primii 2 biți
                  ; de date, restul fiind rezervați
; pentru adresa la care să fie depozitat rezultatul în cazul
; instrucțiunilor de stand „ sau „ pause "
    RD ← 0 ; nu „vom mai citi"
WB: if (IR = pause sau IR = stop)
        AR ← DR[21:0] ; preluăm adresa din SRAM la care să
        DR ← alu_out ; scriem rezultatul ALU (automatului)
        WR ← 1 ; vom activa scrierea
    alu_clk ← 0 ; s-a încheiat un ciclu al automatului

→ ciclurile de mai sus se repetă, fiecare ciclu durând o perioadă de tact
→ am folosit regiștrele: PC pe 21 de biți (numărătorul de program), AR pe 22 de biți (regiștrul de adrese), DR pe 24 de biți (regiștrul de date), IR-C regiștrul de instrucțiuni, semnale de RD (read) și WR (write).
→ alu_out putem spune că joacă rolul de regiștru acumulator.