L 6
(curs 12 - S12)

Outline

- Clasificare
- Organizare memorii
- Creștere latime de banda/spatiu de adrese
- Ciclu de scriere
- Ciclu de citire
- Stivă și coada

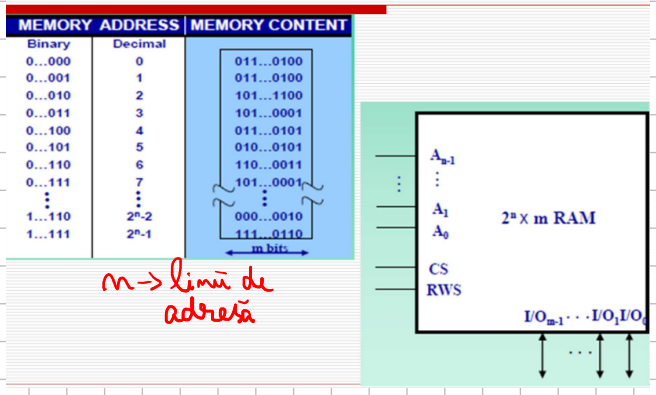
Scriere și Citire		Non-volatile scriere și citire	ROM (numai citire)
Acces random	Acces Non-random	EPROM EEPROM FLASH	Măști programate
SRAM DRAM	FIFO LIFO		

- Din punct de vedere al modului de adresare:
  - prin ADRESĂ
  - prin CONȚINUT (ex. mem. Cache L1)

Clasificare memorii

- Metrici:
  - **Densitatea memoriei** (număr biți/μm^2) și **capacitate**
  - **Timp de acces** (timpul necesar unei op. de scriere sau citire) și **throughput**
  - **Consumul de putere**

Random Access Memory (RAM)
Memorie cu acces aleator



$2^n \cdot m$ — n: nr. linii de adresă; m: nr. biți pe care este stocată linia de memorie; RAM

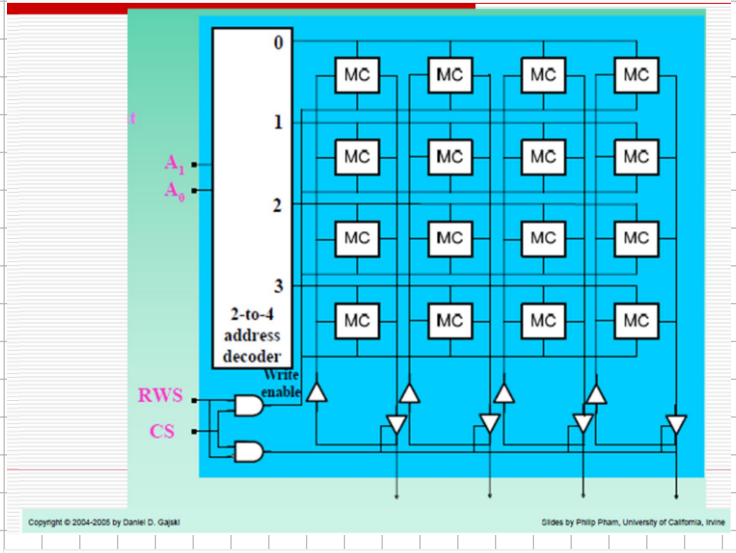
- De câte linii de adrese avem nevoie pentru a accesa o memorie de 1kbit?
- De câte linii de adrese avem nevoie pentru a accesa o memorie de 64kbit?

$1024 \cdot 1 = 2^{10}$ biți $= 2^m \cdot m$

$2^{10} \cdot 64 = 64$ kbit $= 2^{16} = 2^m \cdot m$

$m = ?$

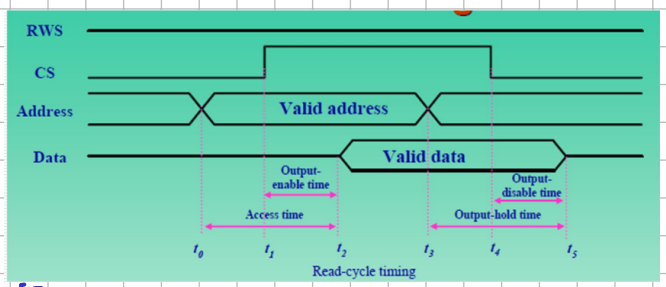
RAM: organizare



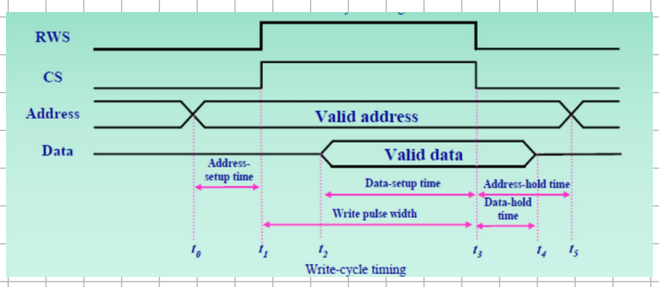
SRAM vs DRAM

- SRAM memorează starea câtă vreme e alimentat. DRAM-ul are nevoie de alimentare + refresh (controler mai complex)
- DRAM are densitate mai mare (1 tranzistor /celulă) în comparație cu SRAM-ul (4-6 pt. cross-cupled inverters), dar:
  - Are nevoie de ciclu refresh.
  - Citirea e distructiva, deci trebuie rescrisa informația imediat după
- FPGA-urile folosesc tehnologie SRAM - BRAM

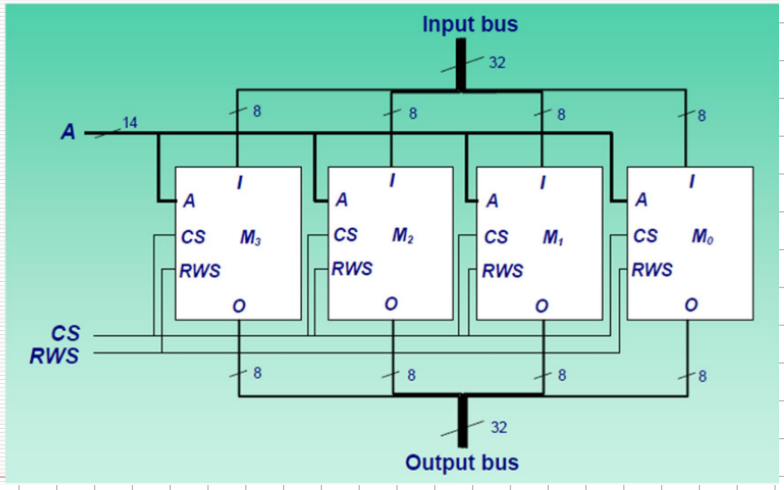
Ciclu RAM citire



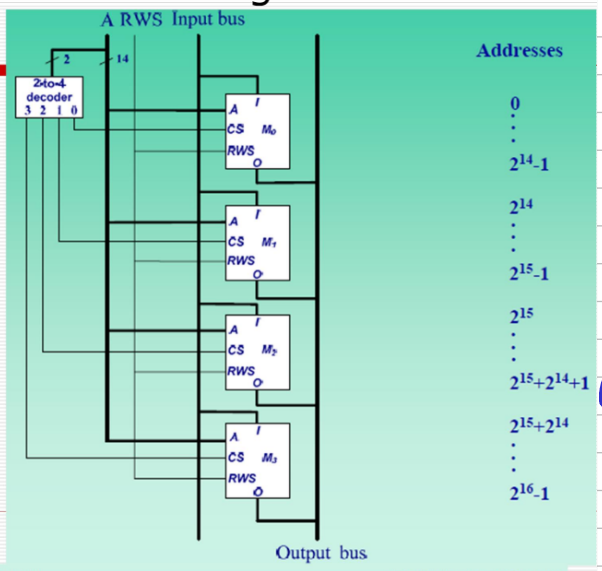
Ciclu RAM scriere



16K x 32 RAM design with 16K x 8 RAMs

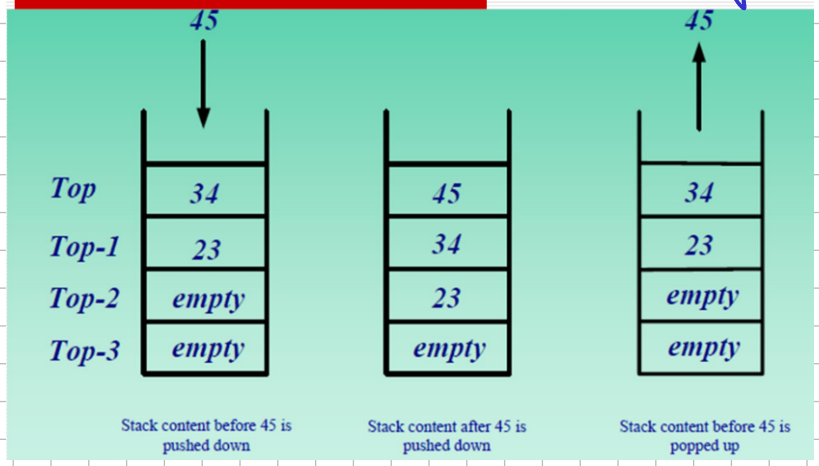


64K x 8 RAM design with 16K x 8 RAMs



$(2^{15}+2^{14}-1)$

Stiva (LIFO) last in → first out



- 4 word, m bit push-down stack cu:
  - m input lines (IN),
  - m output lines (OUT),
- semnale de control:
  - push/pop :
    - 0 data este adaugata in stiva,
    - 1 pentru scoaterea datei din stiva
  - Enable: permite operarea stivei
  - Semnale de stare (Empty si Full)

Stiva 4 cuv.

Push/Pop	Enable	Operations
X	0	No change
0	1	Push
1	1	Pop

Operation table

Push/Pop Enable		Shift register controls S_1	S_0	Counter controls D	E
X	0	0	0	X	0
0	1	1	1	0	1
1	1	1	0	1	1

Control table

Counter outputs Q_2	Q_1	Q_0	Empty	Full
0	0	0	1	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	1

Output table

Registru deplasare

Present state S_1	S_0	Operation	Next state Q_3 Q_2 Q_1 Q_0
0	0	No change	Q_3 Q_2 Q_1 Q_0
0	1	Load input	I_3 I_2 I_1 I_0
1	0	Shift left	Q_2 Q_1 Q_0 I_R
1	1	Shift right	I_L Q_3 Q_2 Q_1

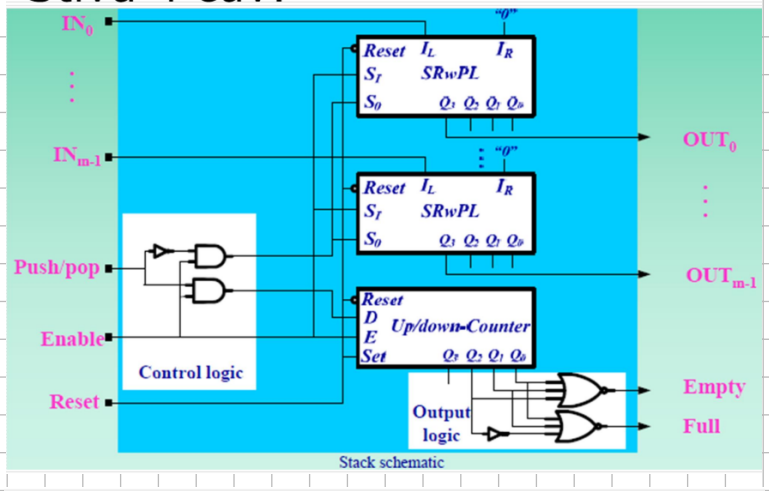
Operation table

Numărător

Load	E	D	Operations
0	0	X	No change
0	1	0	Count up
0	1	1	Count down
1	X	X	Load the input

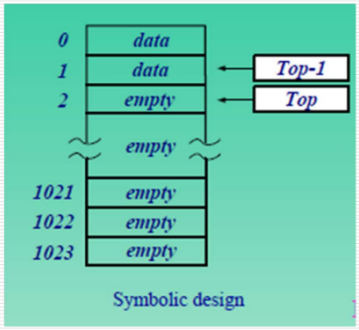
Operation table

Stiva 4 cuv.



Stack schematic

Stiva – implementare fol. 1kB SRAM



Symbolic design

Push/Pop	Enable	Operations
X	0	No change
0	1	Push
1	1	Pop

Operation table

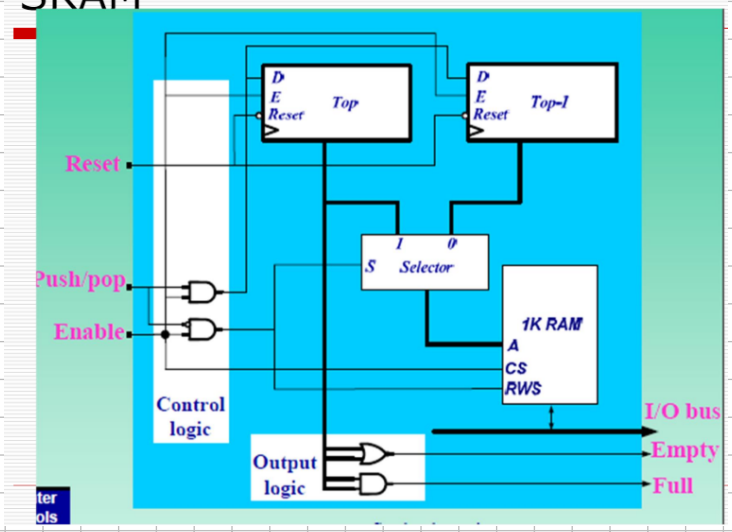
Push/Pop Enable		Selector control S	Memory controls CS	RWS	Counter controls D	E
X	0	X	0	0	X	0
0	1	1	1	1	0	1
1	1	0	1	0	1	1

Control table

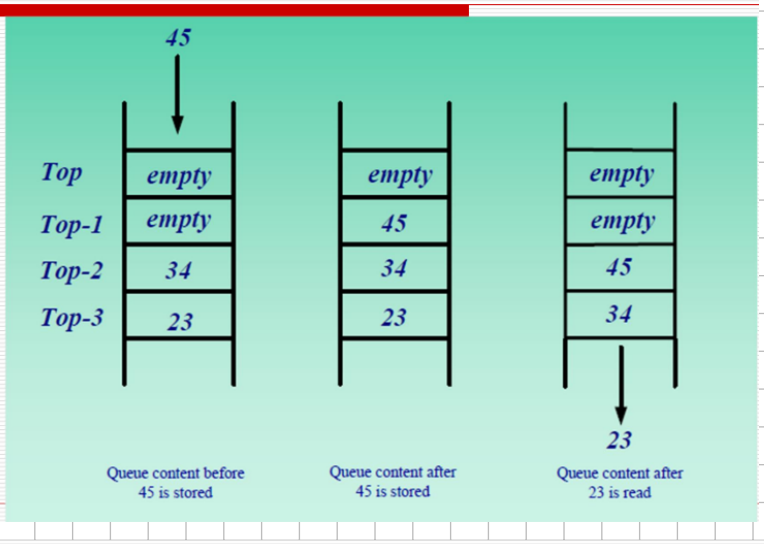
Stiva – implementare fol. 1kB SRAM

- Push: Data loc . RAM (TOP); Increment Top, Top-1
- Pop: Data loc. RAM (Top-1) Date; Decrement Top, Top-1
- Stivă plină: Top=1023;
- Stivă goală: Top=0;
- Locația cu adresa 1023 nu e încărcată niciodată (11 1111 1111)

Stiva – implementare fol. 1kB SRAM



FIFO



FIFO – 4 cuvinte

READ/WRITE	ENABLE	OPERATIONS
X	0	No change
0	1	Read
1	1	Write

Operation table

READ/WRITE	ENABLE	S_1 S_0	D E
X	0	0 0	X 0
0	1	0 0	1 1
1	1	1 0	0 1

Control table

Registru deplasare

Present state			Next state
S_1	S_0	Operation	Q_3 Q_2 Q_1 Q_0
0	0	No change	Q_3 Q_2 Q_1 Q_0
0	1	Load input	I_3 I_2 I_1 I_0
1	0	Shift left	Q_2 Q_1 Q_0 I_R
1	1	Shift right	I_L Q_3 Q_2 Q_1

Operation table

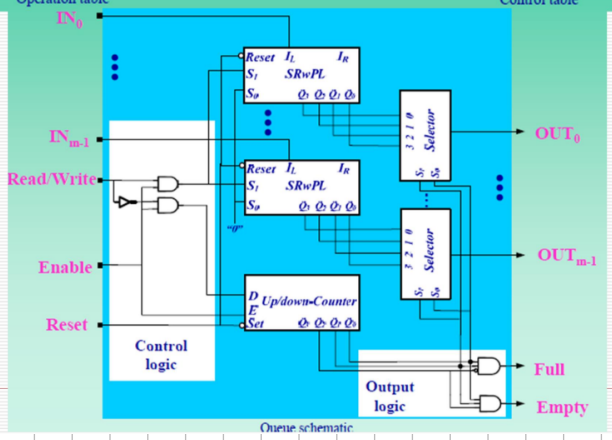
FIFO – 4 cuvinte

READ/WRITE	ENABLE	OPERATIONS
X	0	No change
0	1	Read
1	1	Write

Operation table

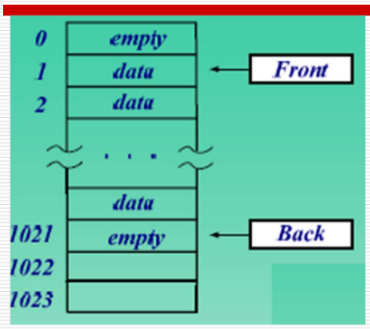
READ/WRITE	ENABLE	S_1	S_0	D	E
X	0	0	0	X	0
0	1	0	0	1	1
1	1	1	0	0	1

Control table



Queue schematic

FIFO - 1kB SRAM



Read/Write	Enable	Operations
X	0	No change
0	1	Read
1	1	Write

Operation table

Read/Write	Enable	S	CS	RWS	E (Front)	E (Back)
X	0	X	0	X	0	0
0	1	1	1	0	1	0
1	1	0	1	1	0	1

Control table

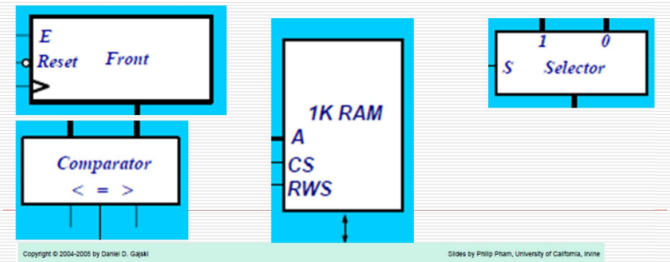
FIFO - 1kB SRAM

Read/Write	Enable	Operations
X	0	No change
0	1	Read
1	1	Write

Operation table

Read/Write	Enable	S	CS	RWS	E (Front)	E (Back)
X	0	X	0	X	0	0
0	1	1	1	0	1	0
1	1	0	1	1	0	1

Control table



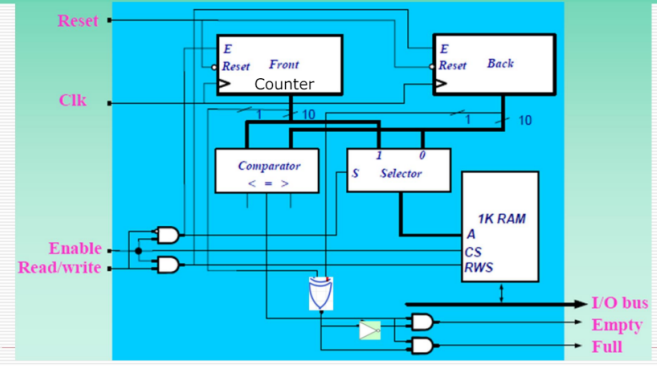
Slides by Philip Pham, University of California, Irvine

Read/Write	Enable	Operations
X	0	No change
0	1	Read
1	1	Write

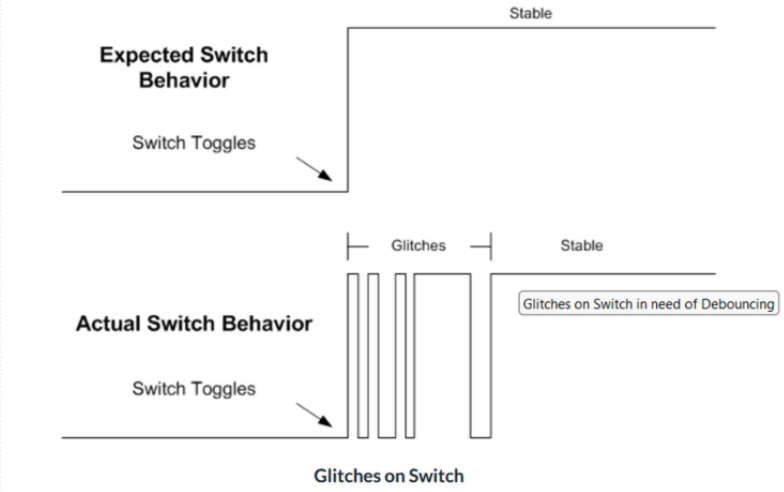
Operation table

Read/Write	Enable	S	CS	RWS	E (Front)	E (Back)
X	0	X	0	X	0	0
0	1	1	1	0	1	0
1	1	0	1	1	0	1

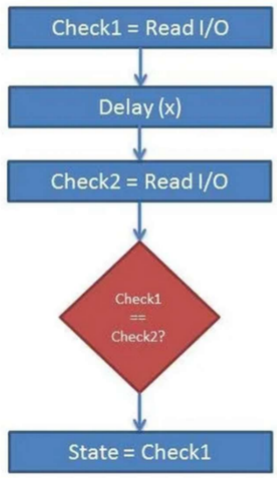
Control table



Debouncing – problem statement



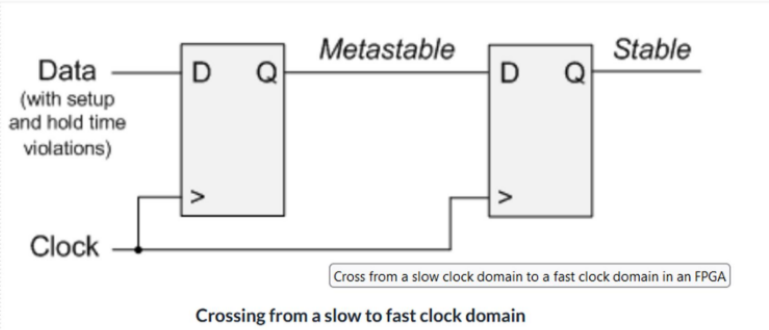
Debouncing – possible method



- 50 MHz clock
- 10 ms delay (worst case)
- ? Counter MAX for generating the delay ?

Clock-domain crossing (I)

- **Crossing from slower clock domain to faster clock domain**



Crossing from a slow to fast clock domain

FSM sync sw (OM) cu clock FPGA

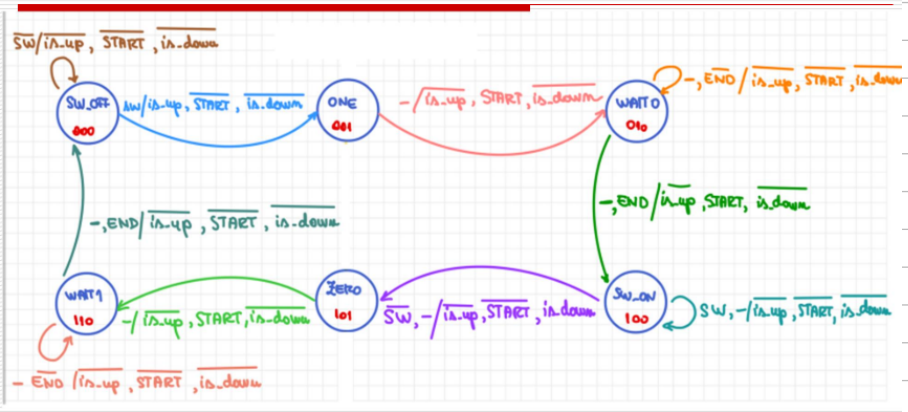
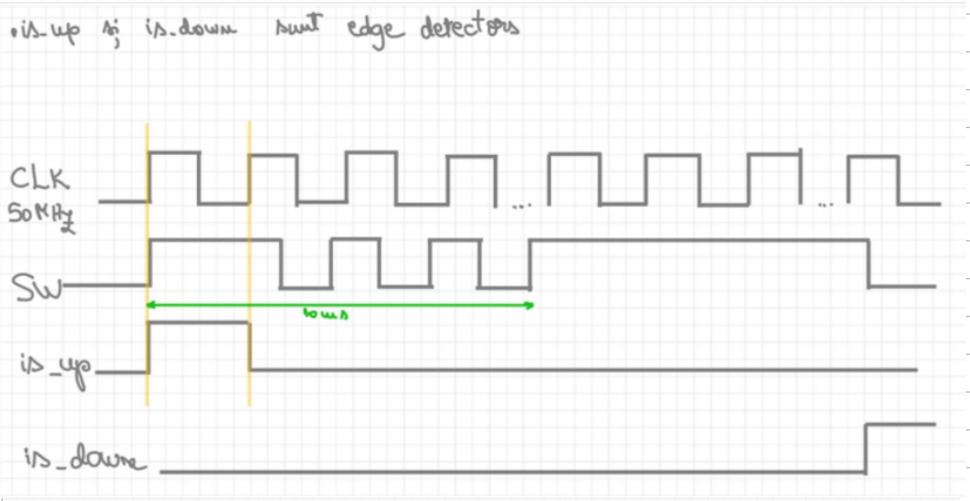
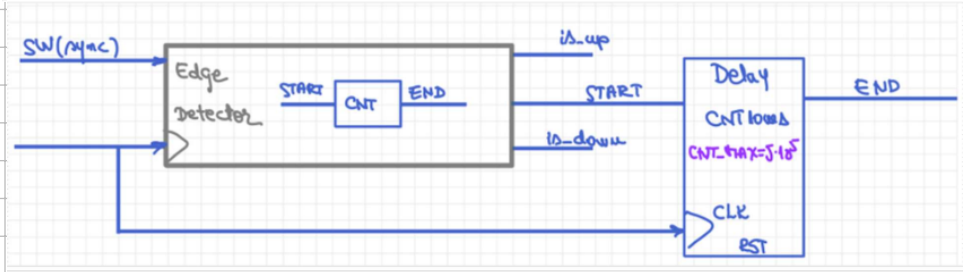
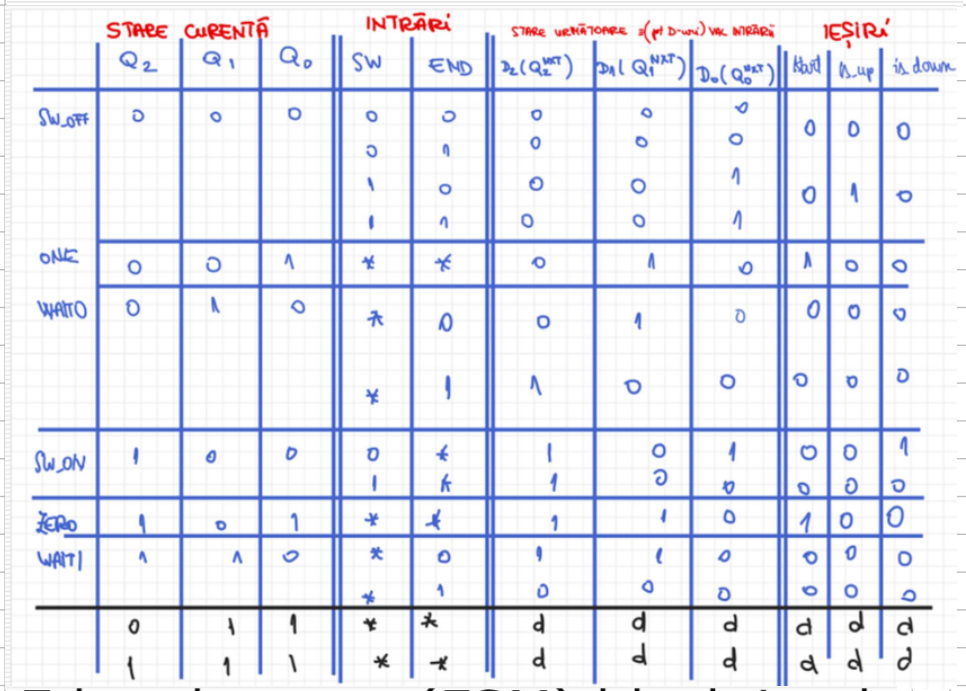


Diagrama de timp







Edge detector (FSM) block impl