Universitatea
Politehnica
Timişoara

P-ţa Victoriei nr. 2
RO 300006 - Timişoara
Tel: +4 0256 403000
Fax: +4 0256 403021
rector@rectorat.upt.ro
www.upt.ro

Logică digitală

-Curs 13-

Memorii

Outline

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

Clasificare memorii

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)

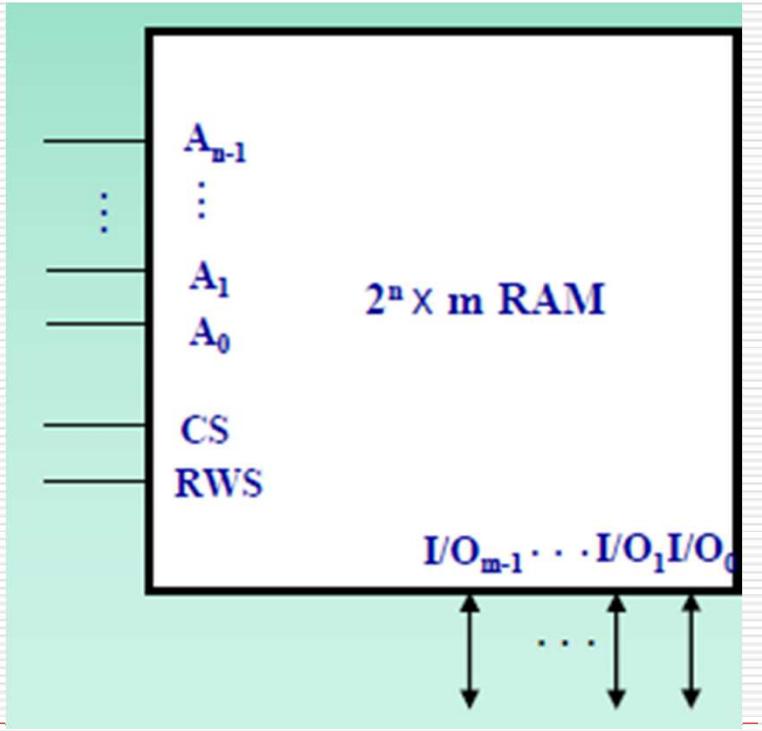
Clasificare memorii

- 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

MEMORY ADDRESS		MEMORY CONTENT
Binary	Decimal	
0…000	0	011…0100
0…001	1	011…0100
0…010	2	101…1100
0…011	3	101…0001
0…100	4	011…0101
0…101	5	010…0101
0…110	6	110…0011
0…111	7	101…0001
⋮	⋮	⋮
1…110	2^n-2	000…0010
1…111	2^n-1	111…0110

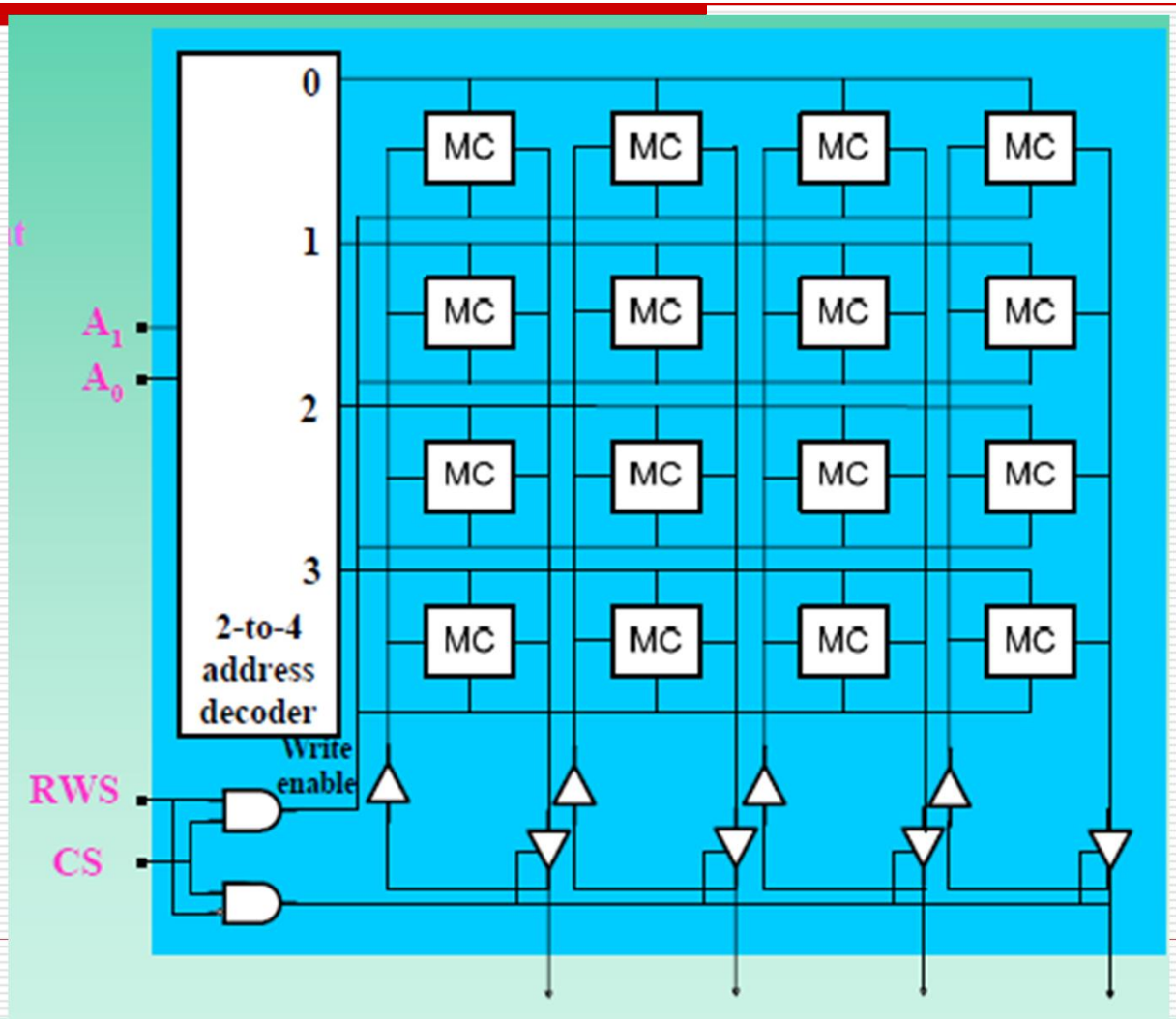
m bits



Random Access Memory (RAM)
Memorie cu acces aleator

- 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?

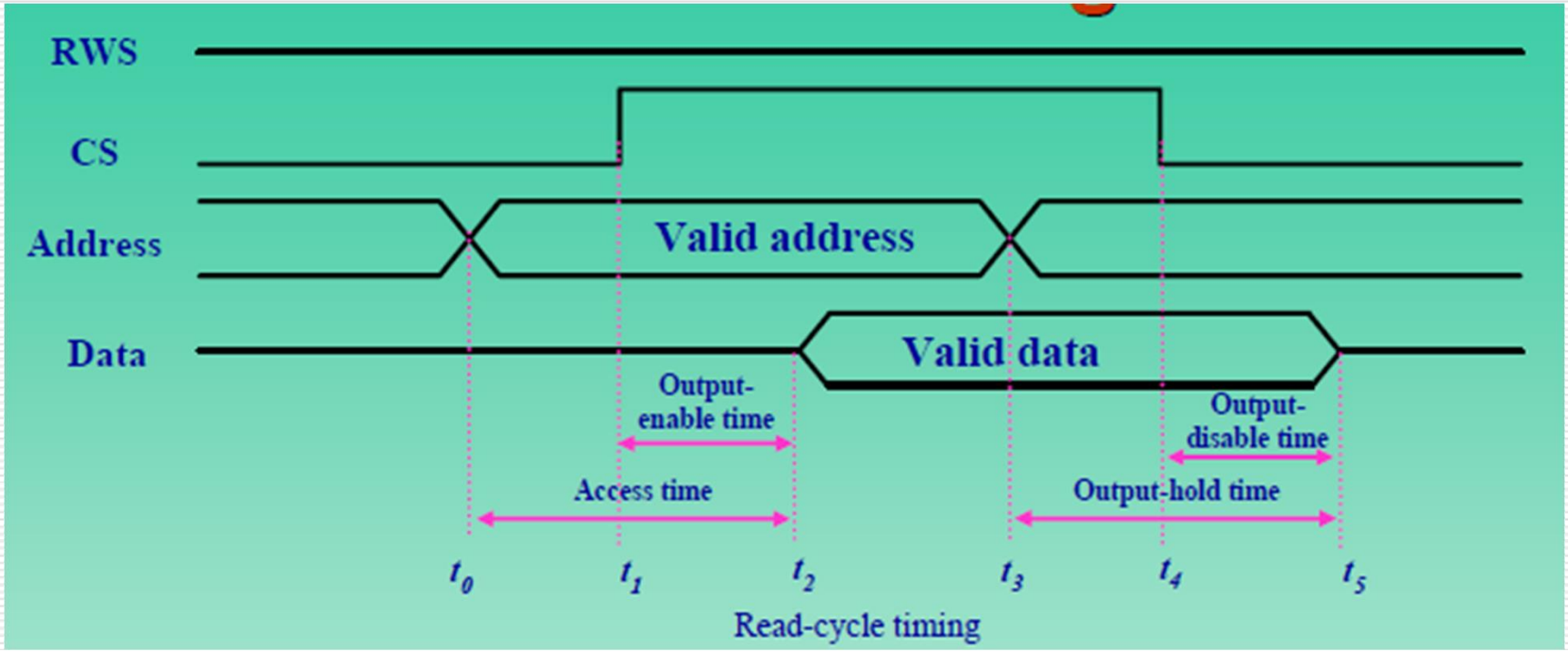
RAM: organizare



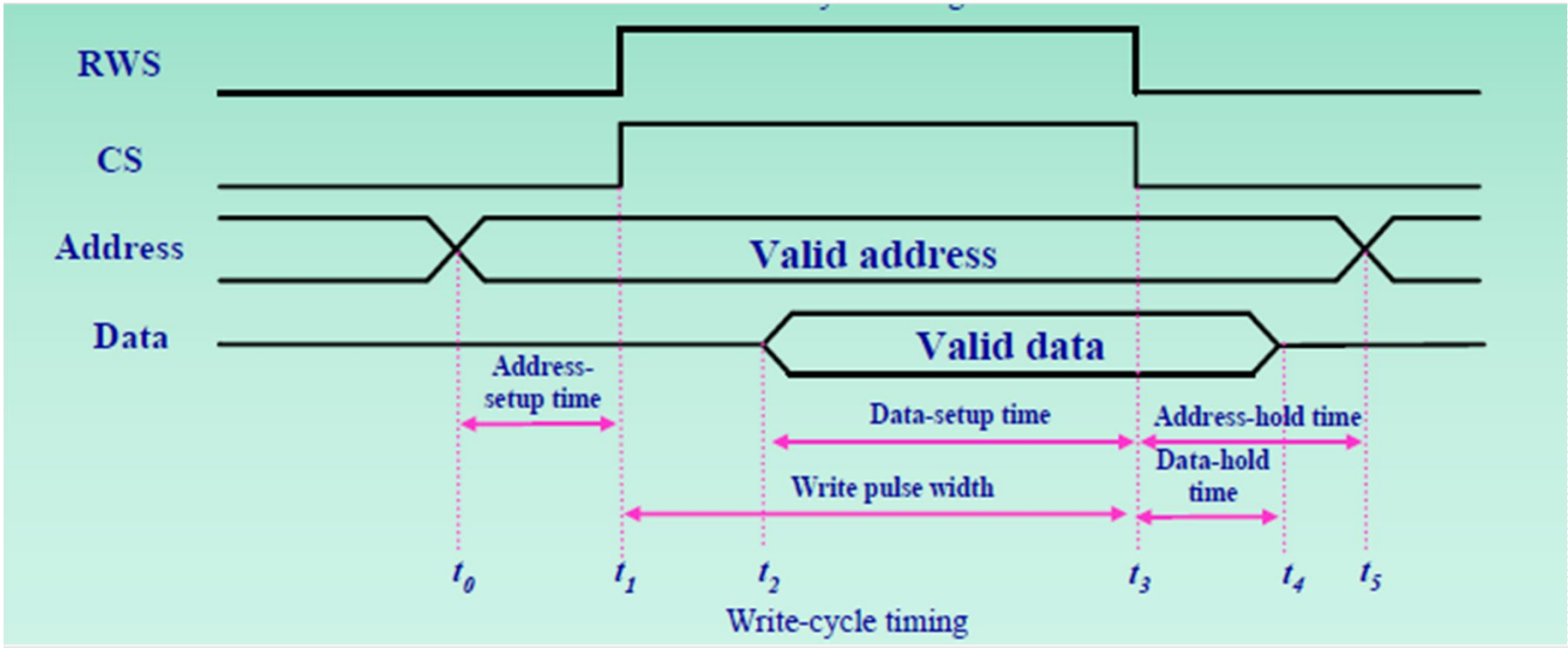
SRAM vs DRAM

- 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

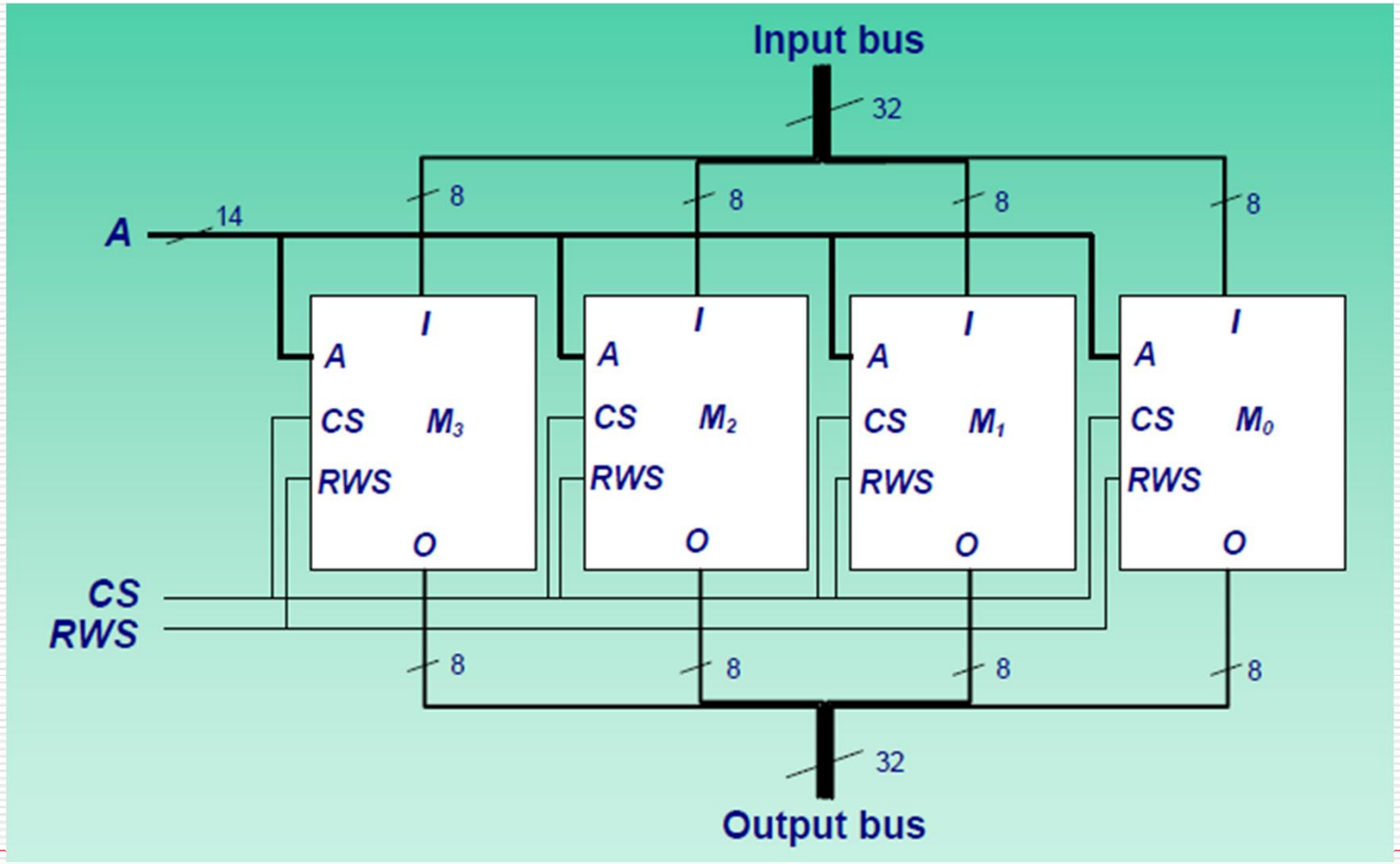
Ciclu RAM de citire



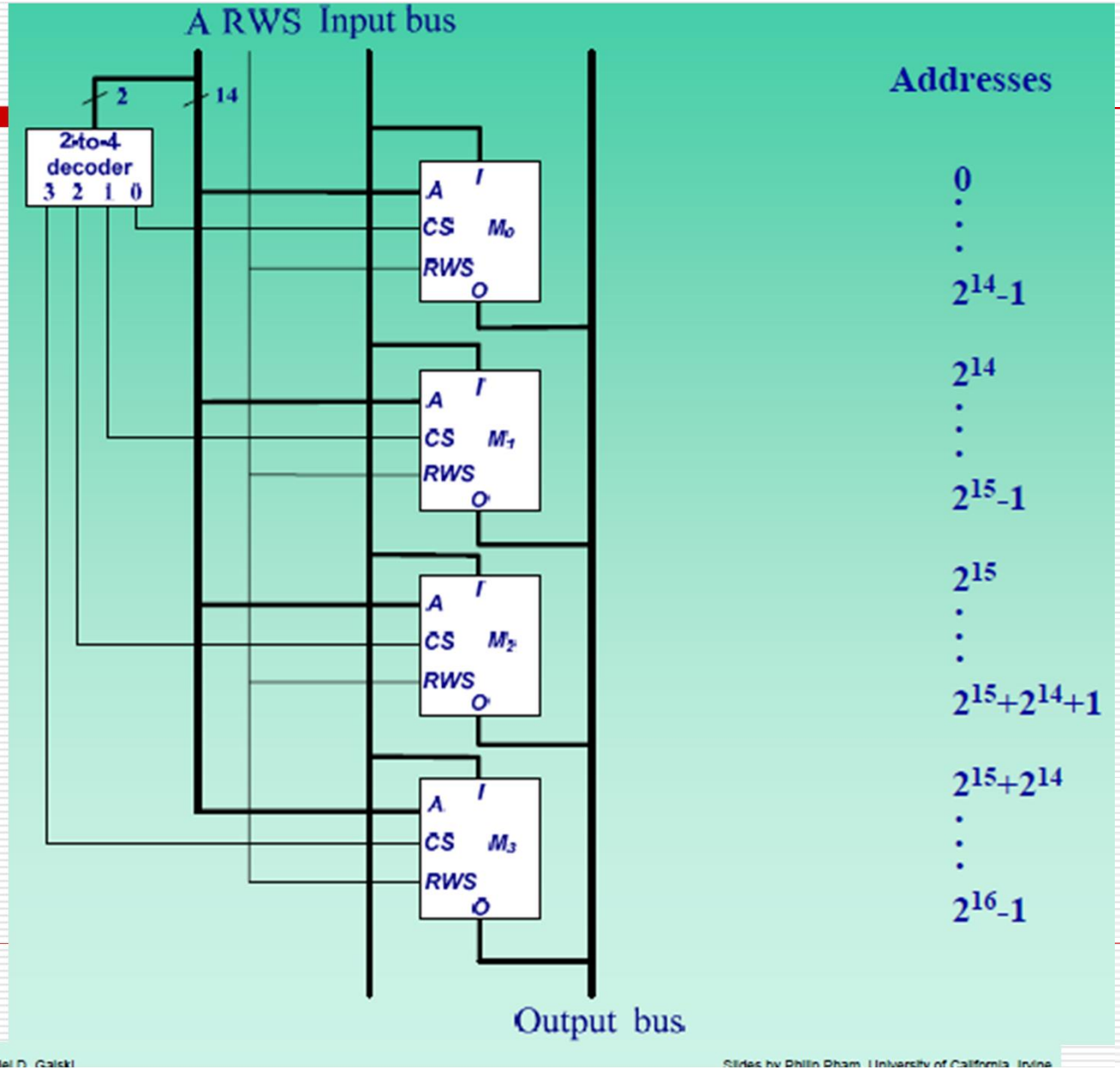
Ciclu RAM scriere



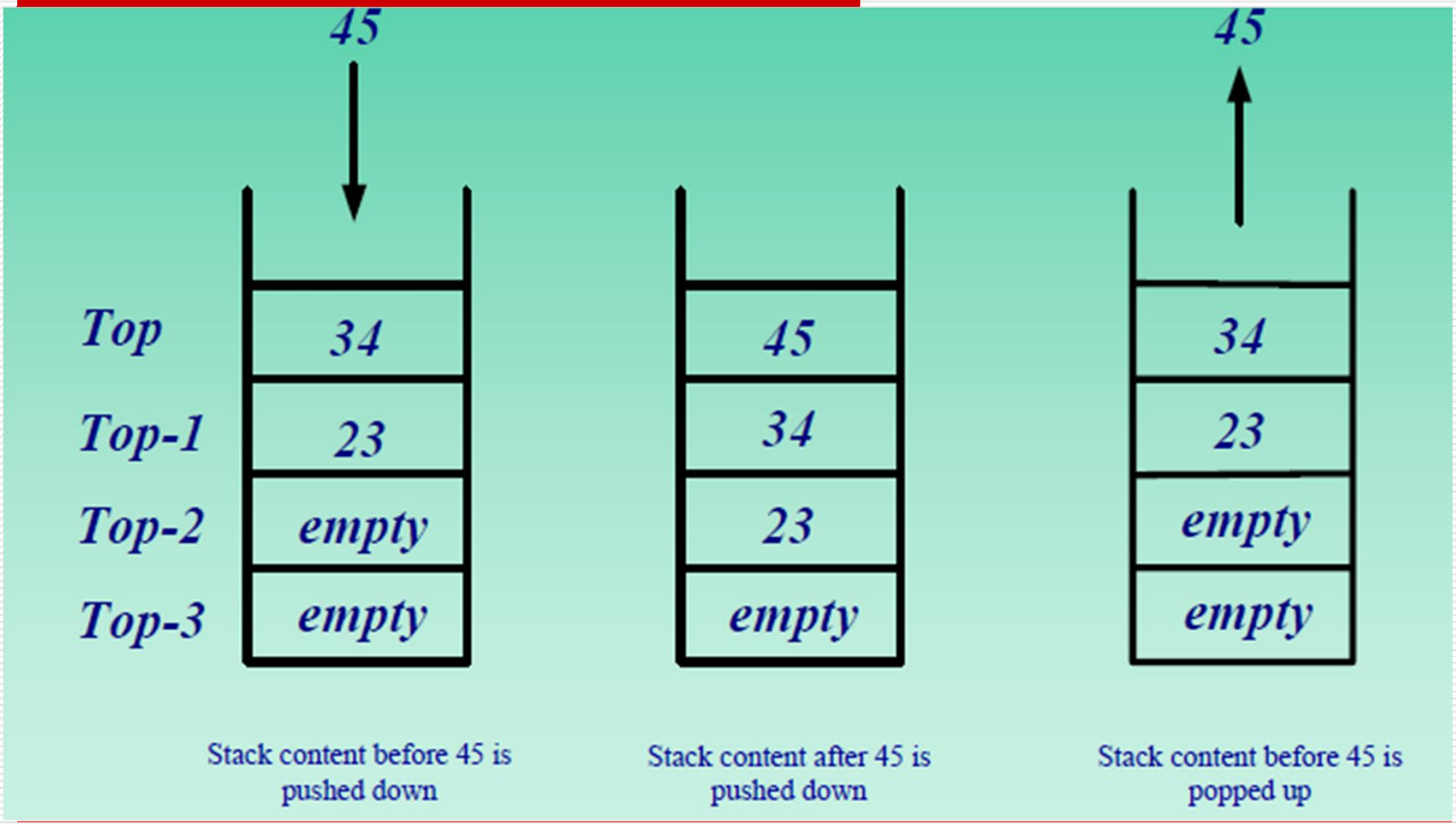
16K x 32 RAM design with 16K x 8 RAMs



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



Stiva


45
Top
34
Top-1
23
Top-2
empty
Top-3
empty
Stack content before 45 is pushed down
45
34
23
empty
Stack content after 45 is pushed down
45
34
23
empty
empty
Stack content before 45 is popped up

Stiva

- 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

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

Control table

Counter outputs			Empty	Full
Q_2	Q_1	Q_0		
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		Operation	Next state
S_1	S_0		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

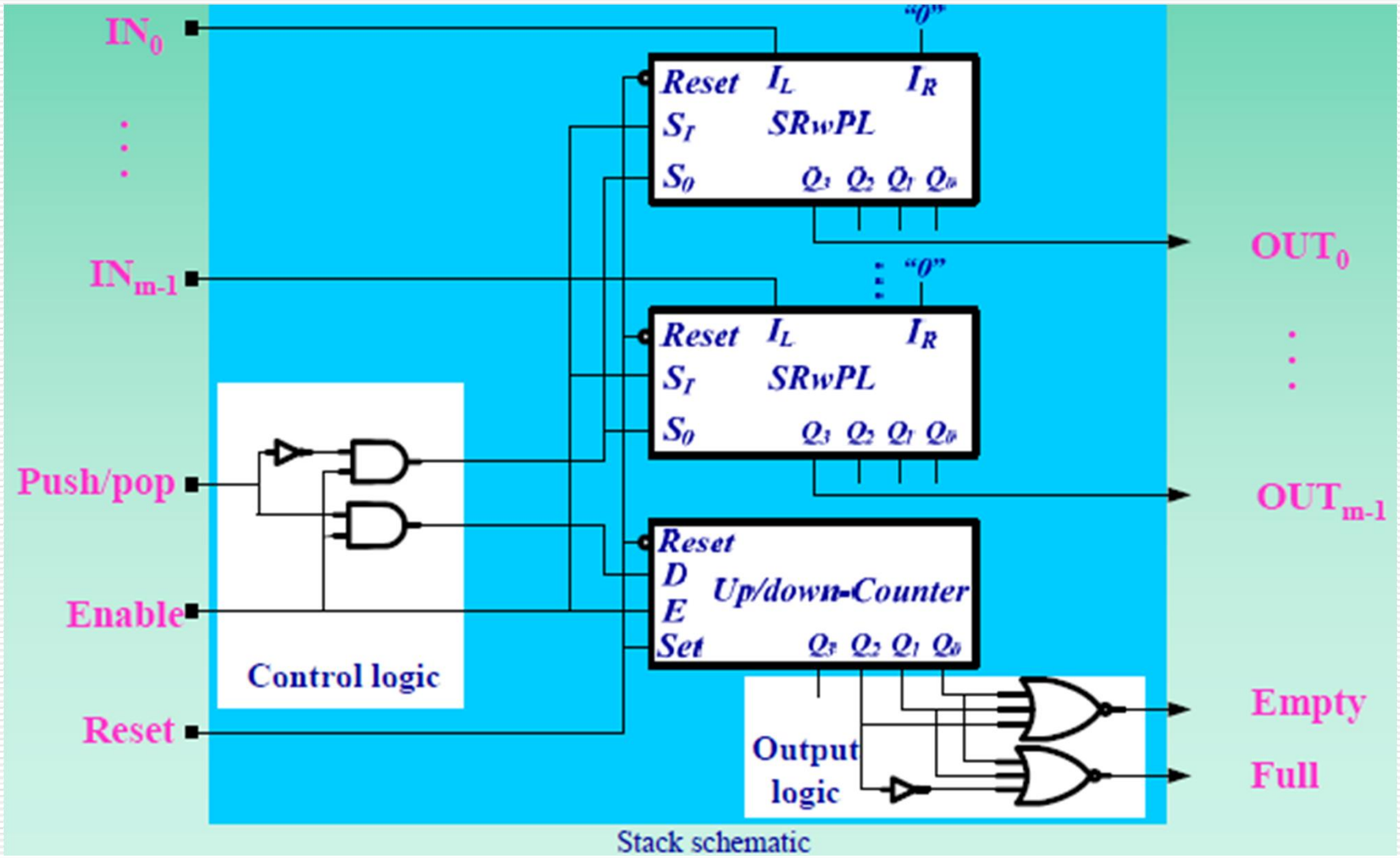
Operation table

Numărător

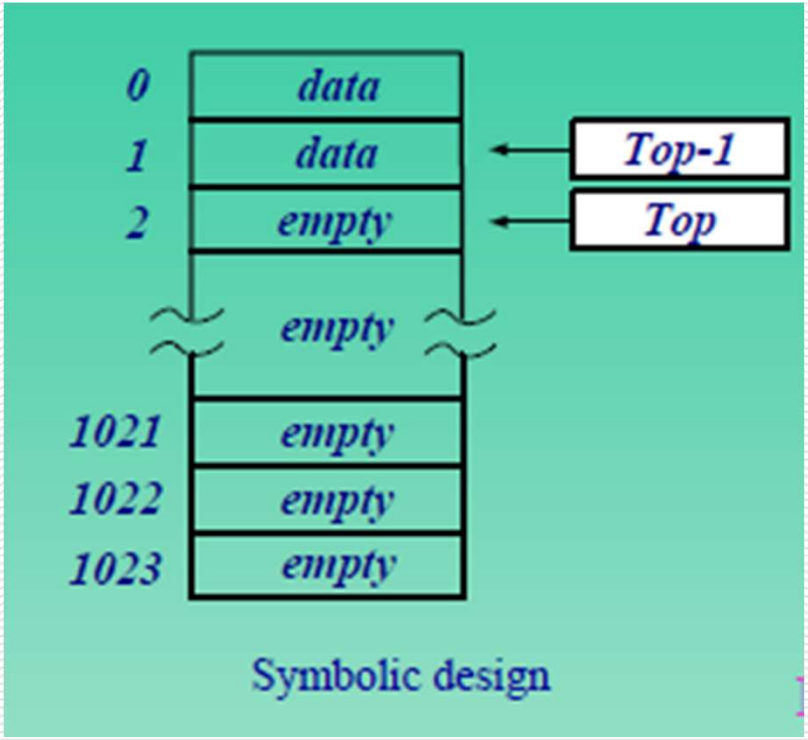
Load	E	D	Operations
0	0	X	No change
0	1	0	Count up
0	1	1	Count down
1	X	X	Load the input

Operation table

Stiva 4 cuv.


IN0
INm-1
Push/pop
Enable
Reset
"0"
Reset
IL
IR
SI
SRwPL
S0
Q3 Q2 Q1 Q0
"0"
Reset
IL
IR
SI
SRwPL
S0
Q3 Q2 Q1 Q0
Reset
D
E
Set
Up/down-Counter
Q3 Q2 Q1 Q0
Control logic
Output logic
OUT0
OUTm-1
Empty
Full
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

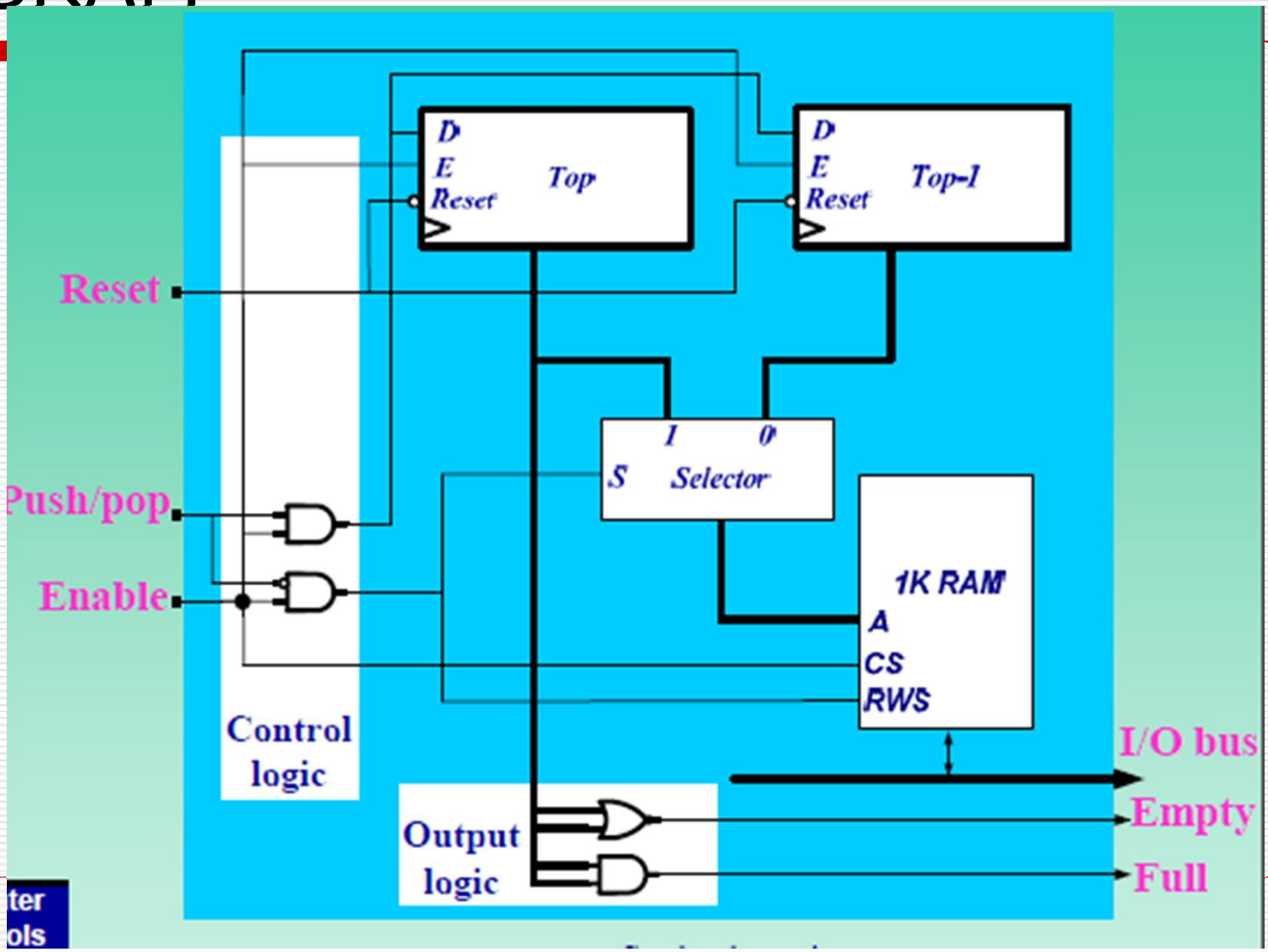
		Selector control	Memory controls		Counter controls	
Push/Pop	Enable	S	CS	RWS	D	E
X	0	X	0	0	X	0
0	1	1	1	1	0	1
1	1	0	1	0	1	1

Control table

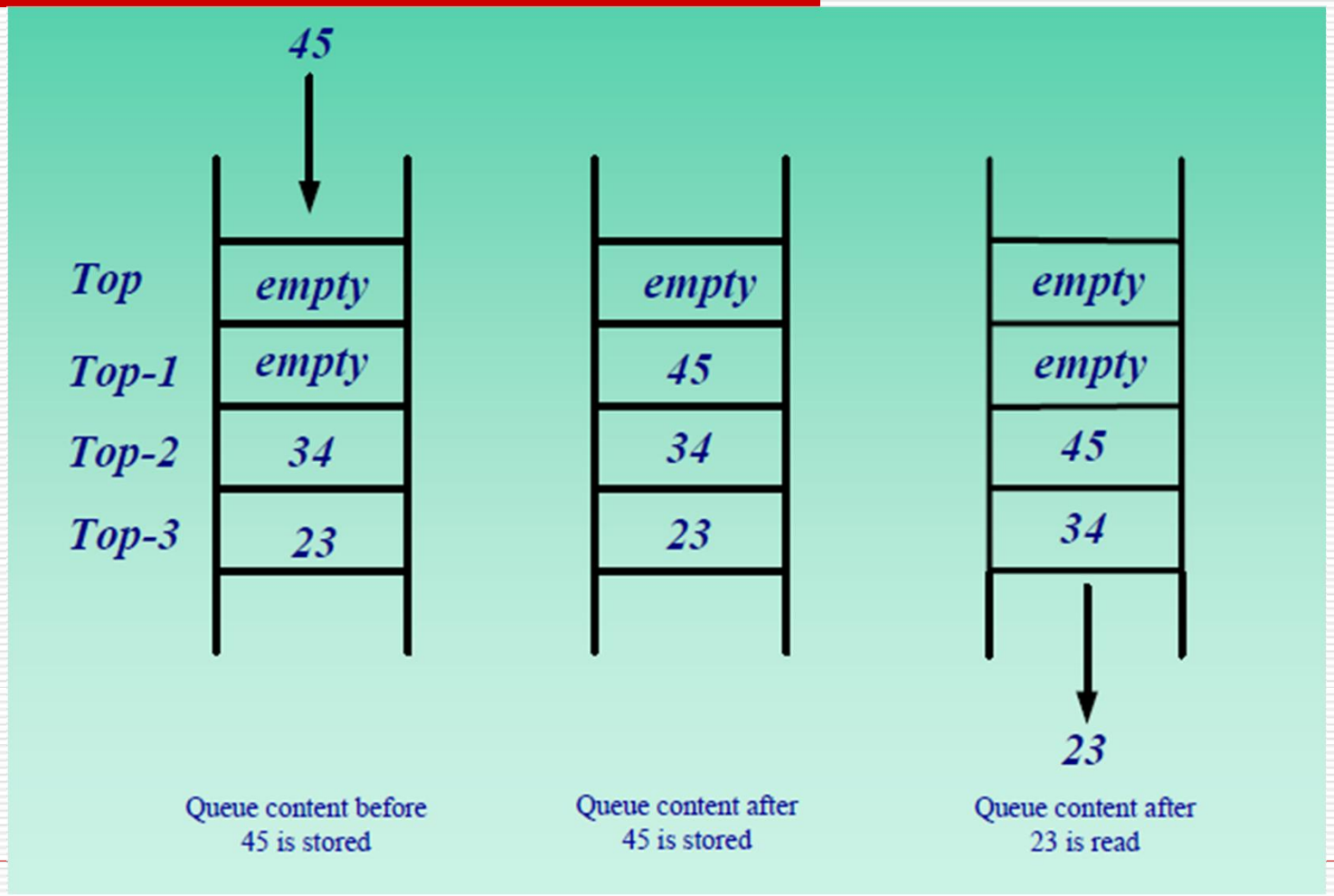
Stiva – implementare fol. 1kB SRAM

- 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


45
Top
Top-1
Top-2
Top-3
empty
empty
34
23
empty
45
34
23
empty
empty
45
34
23
Queue content before 45 is stored
Queue content after 45 is stored
Queue content after 23 is read

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

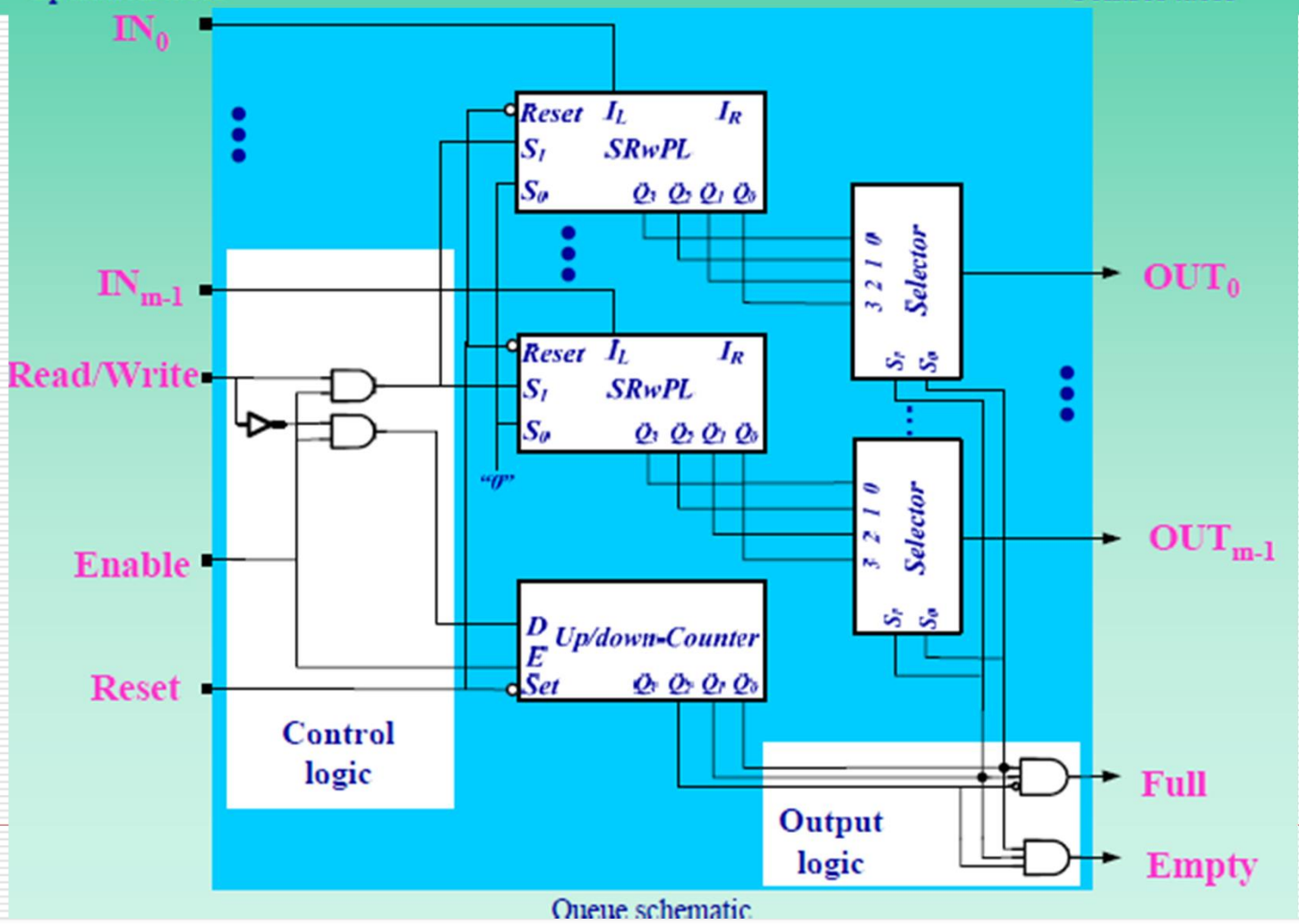
FIFO – 4 cuvinte

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

Operation table

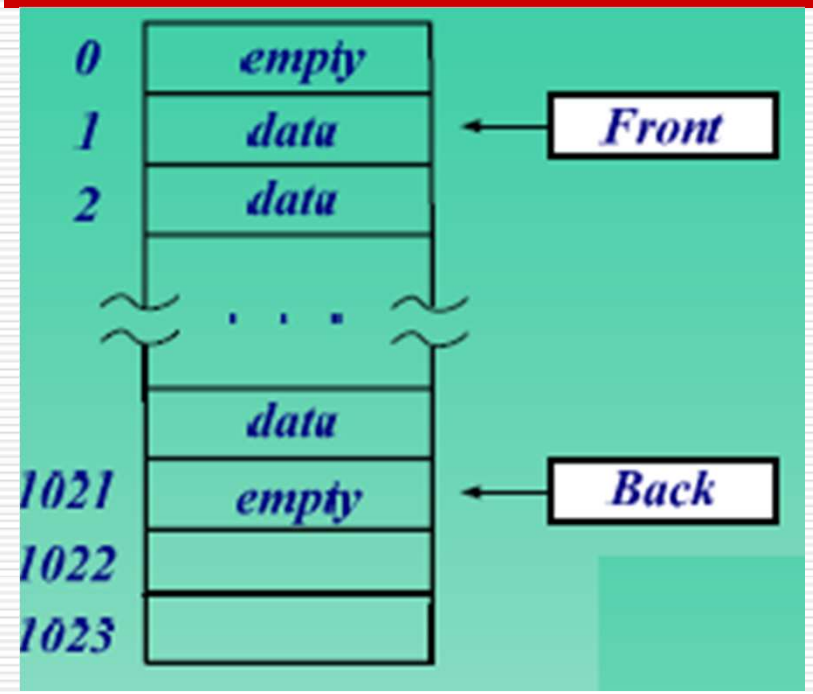
READ/WRITE	ENABLE	S_1	S_0	D	E
X	0	0	0	X	0
0	1	0	0	1	1
1	1	1	0	0	1

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

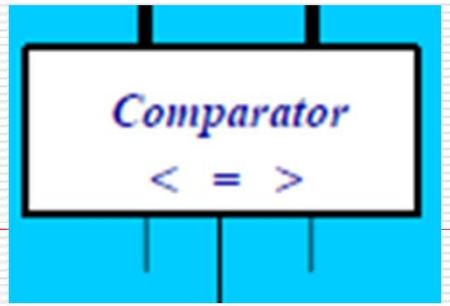
FIFO - 1kB SRAM

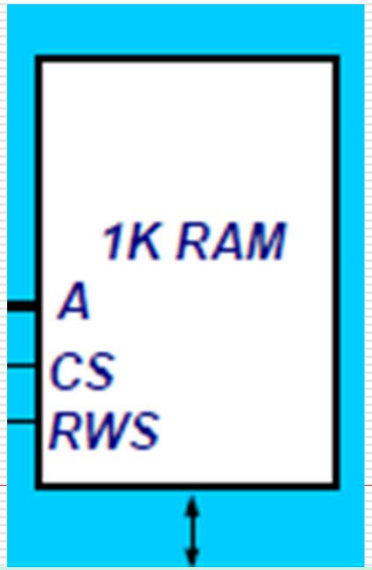
Read/Write	Enable	Operations
X	0	No change
0	1	Read
1	1	Write

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



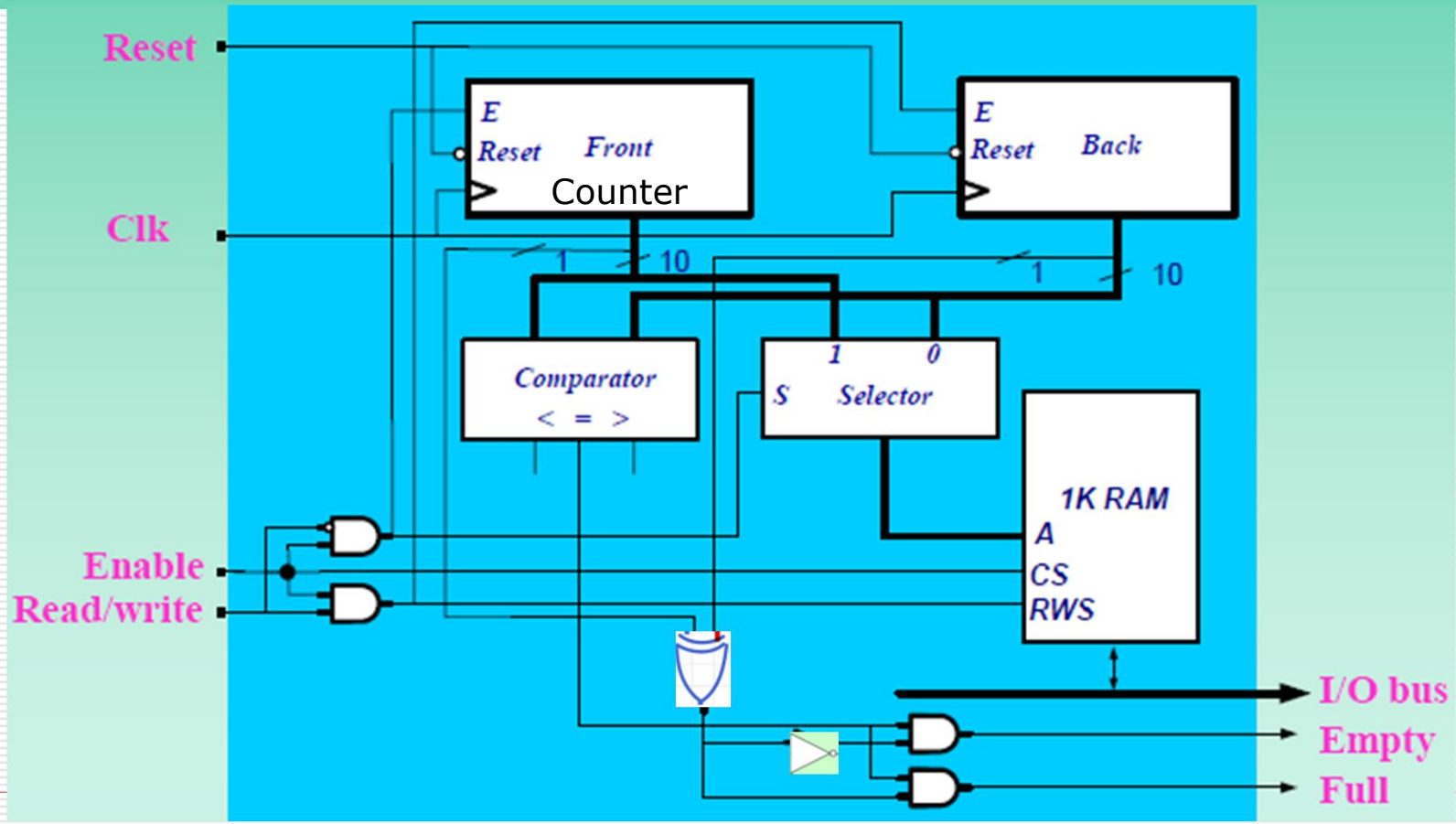


Read/Write	Enable	Operations
X	0	No change
0	1	Read
1	1	Write

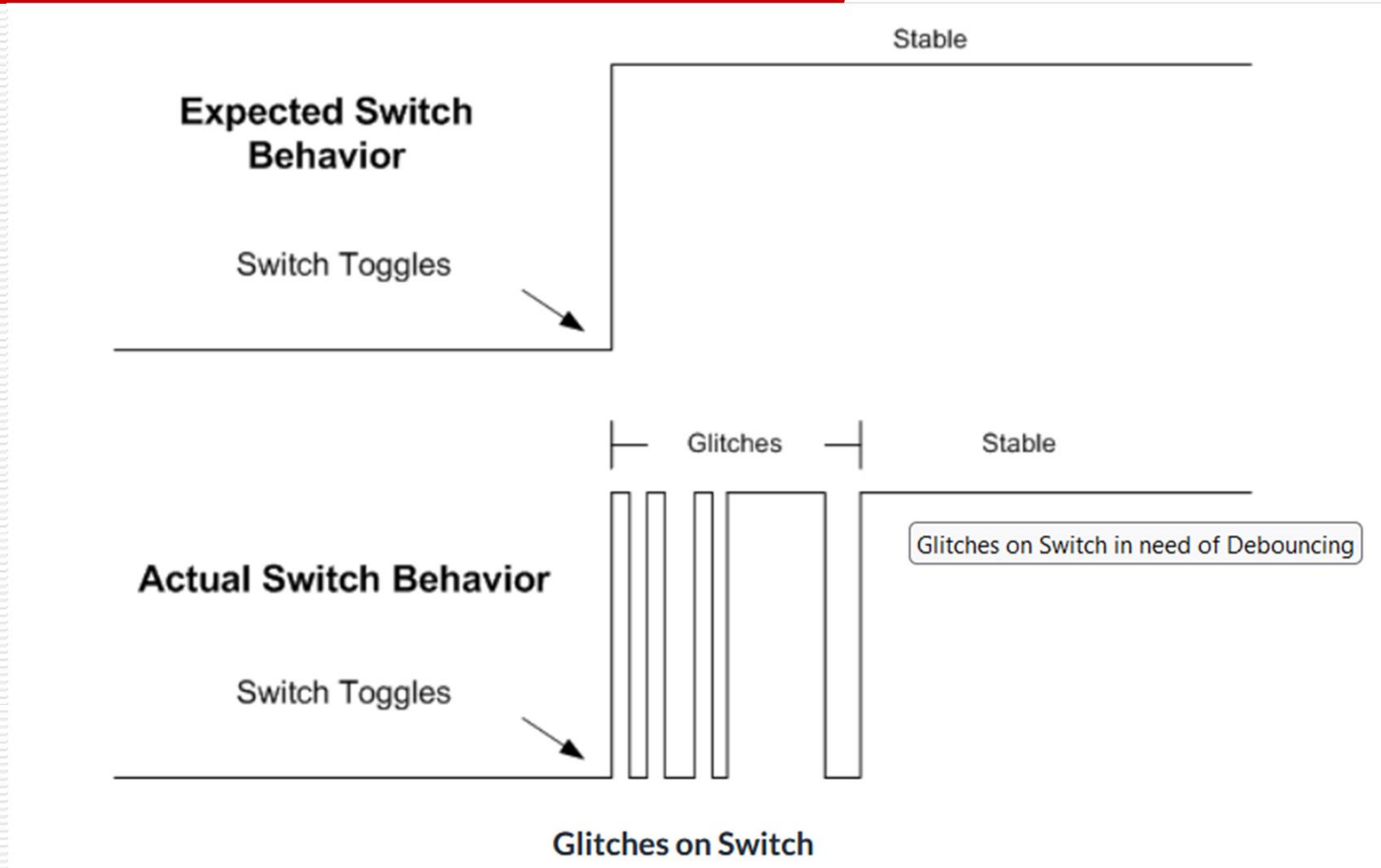
Operation table

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

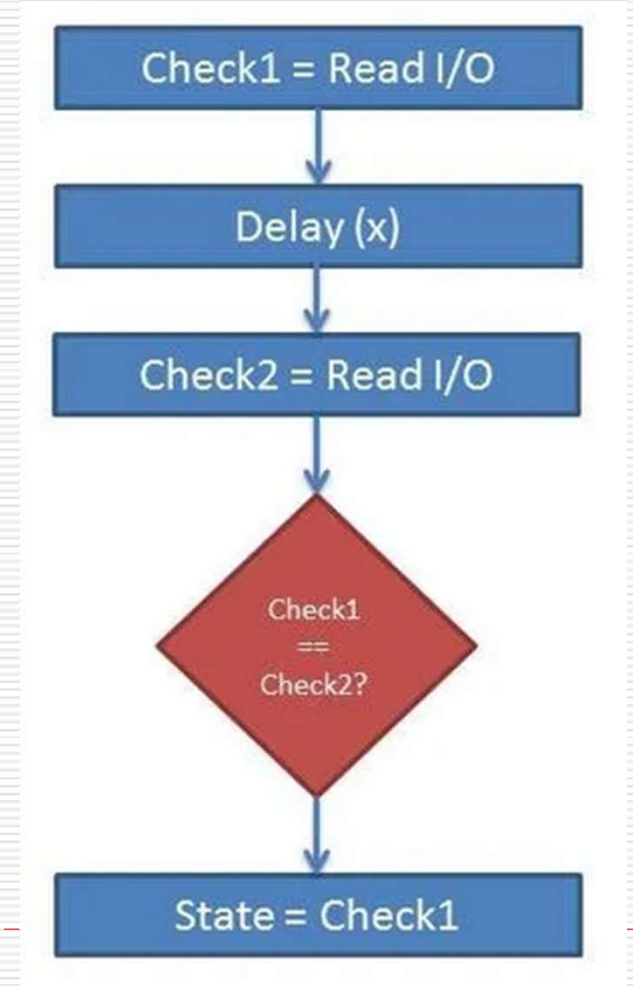
Control table



Debouncing – problem statement



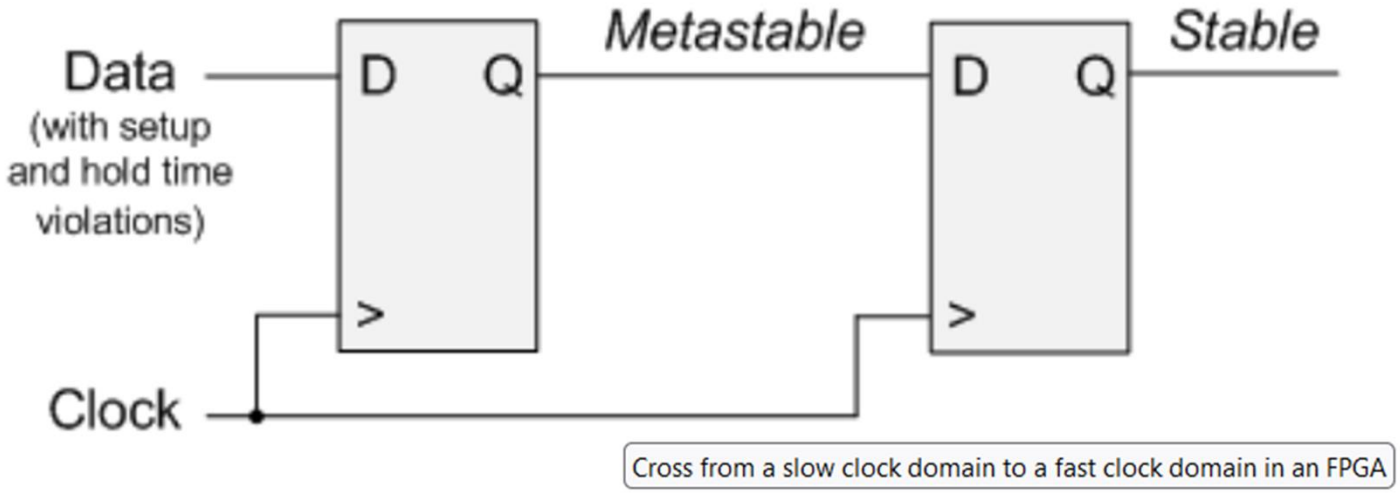
Debouncing – possible method



- 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

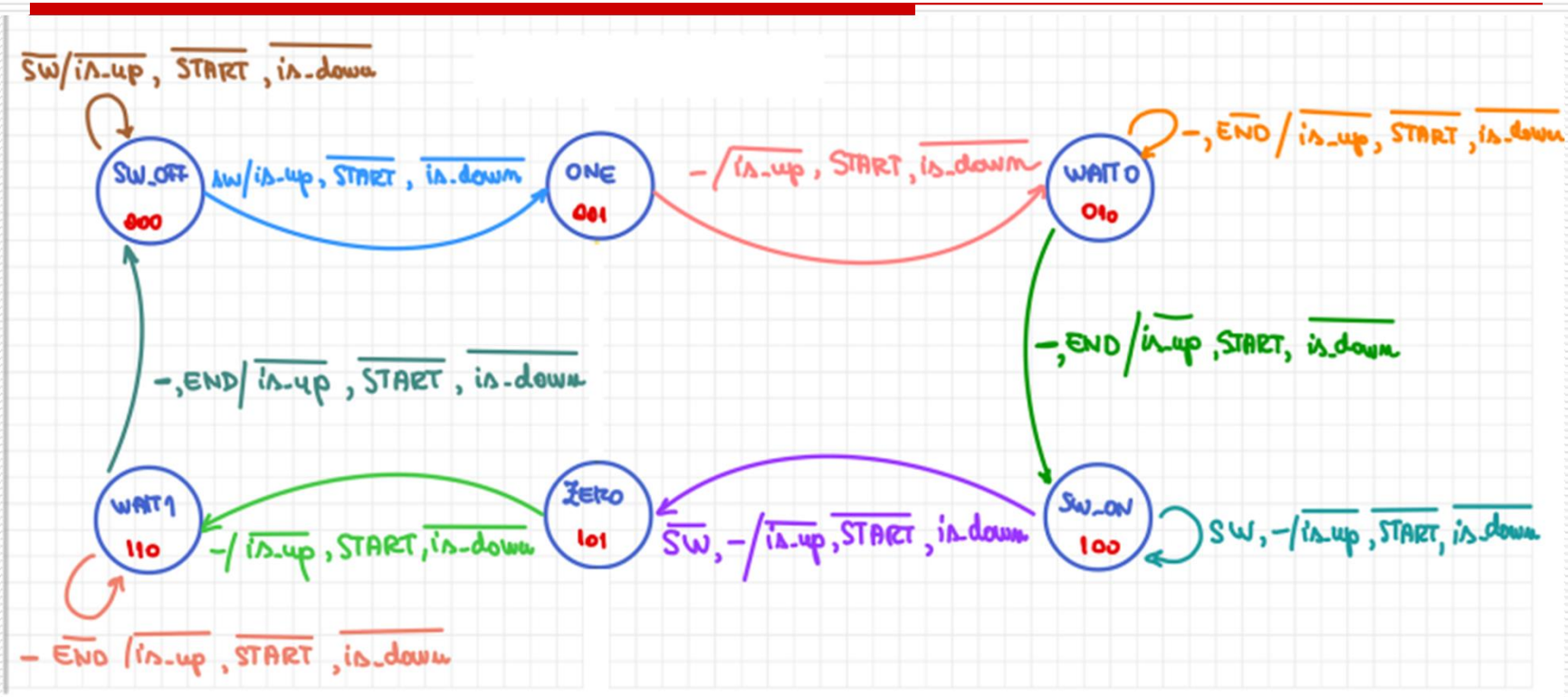
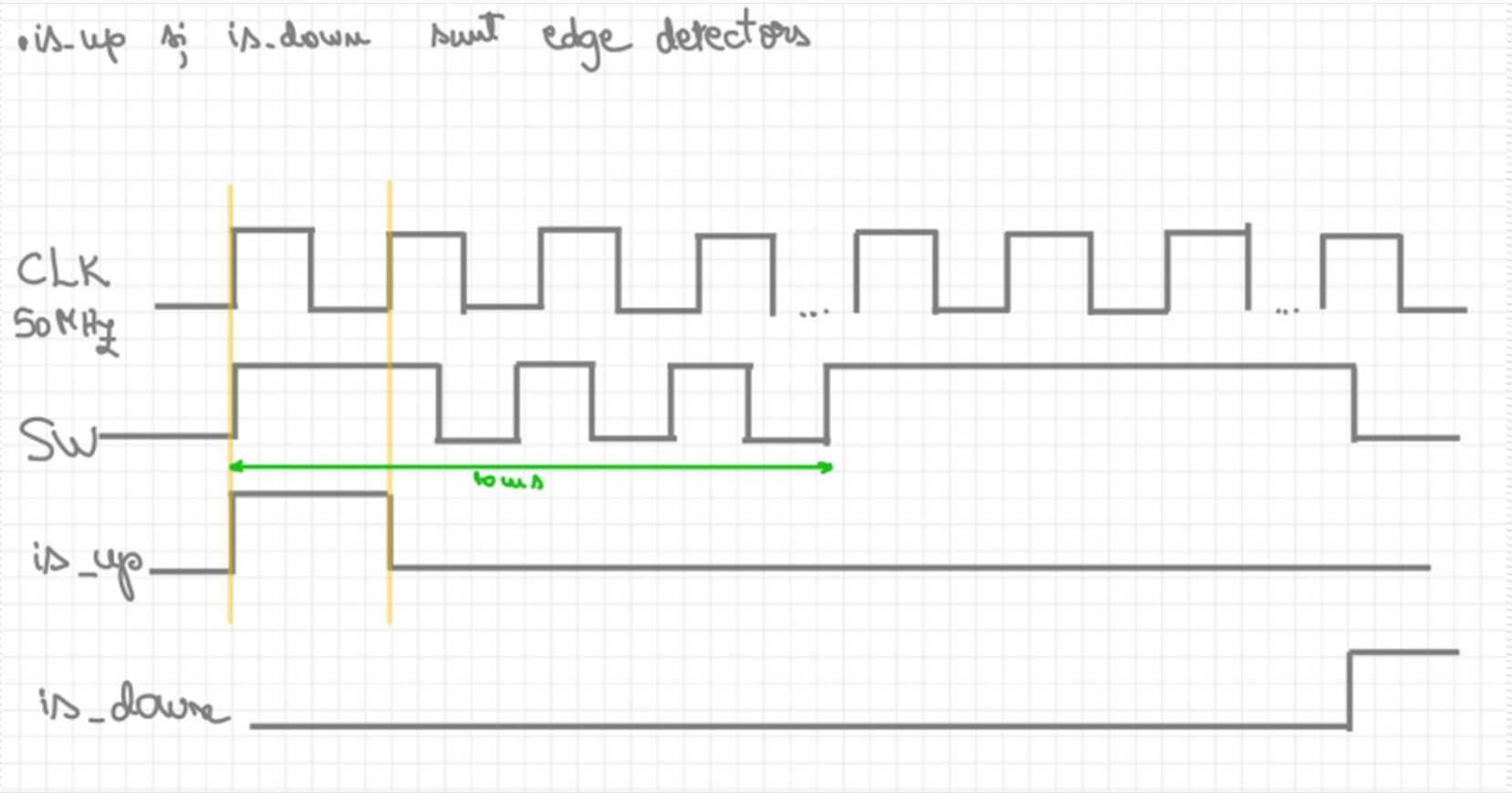
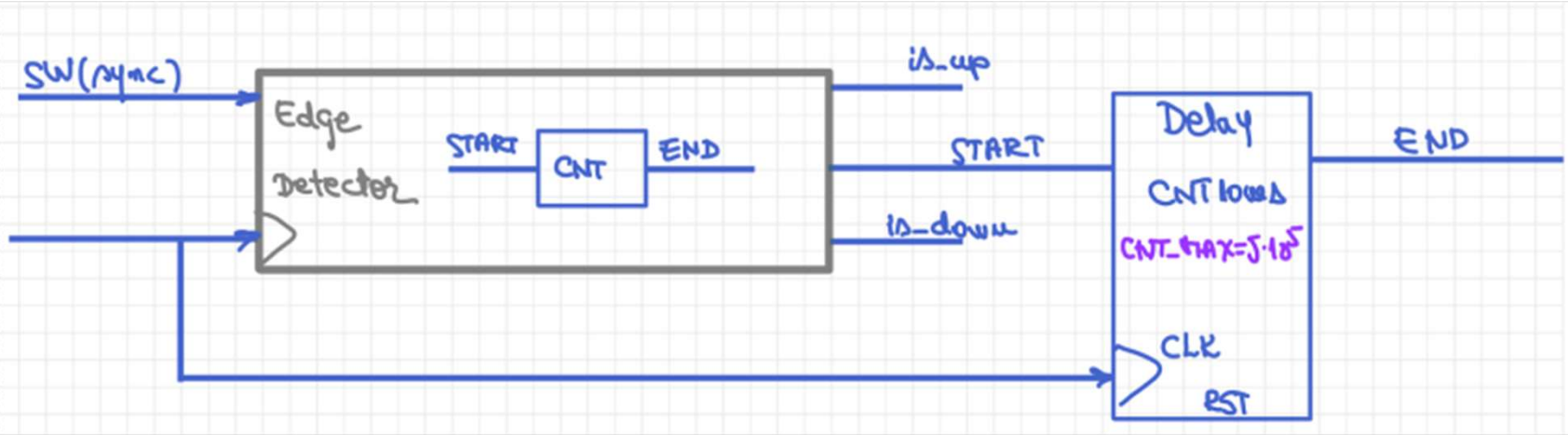


Diagrama de timp

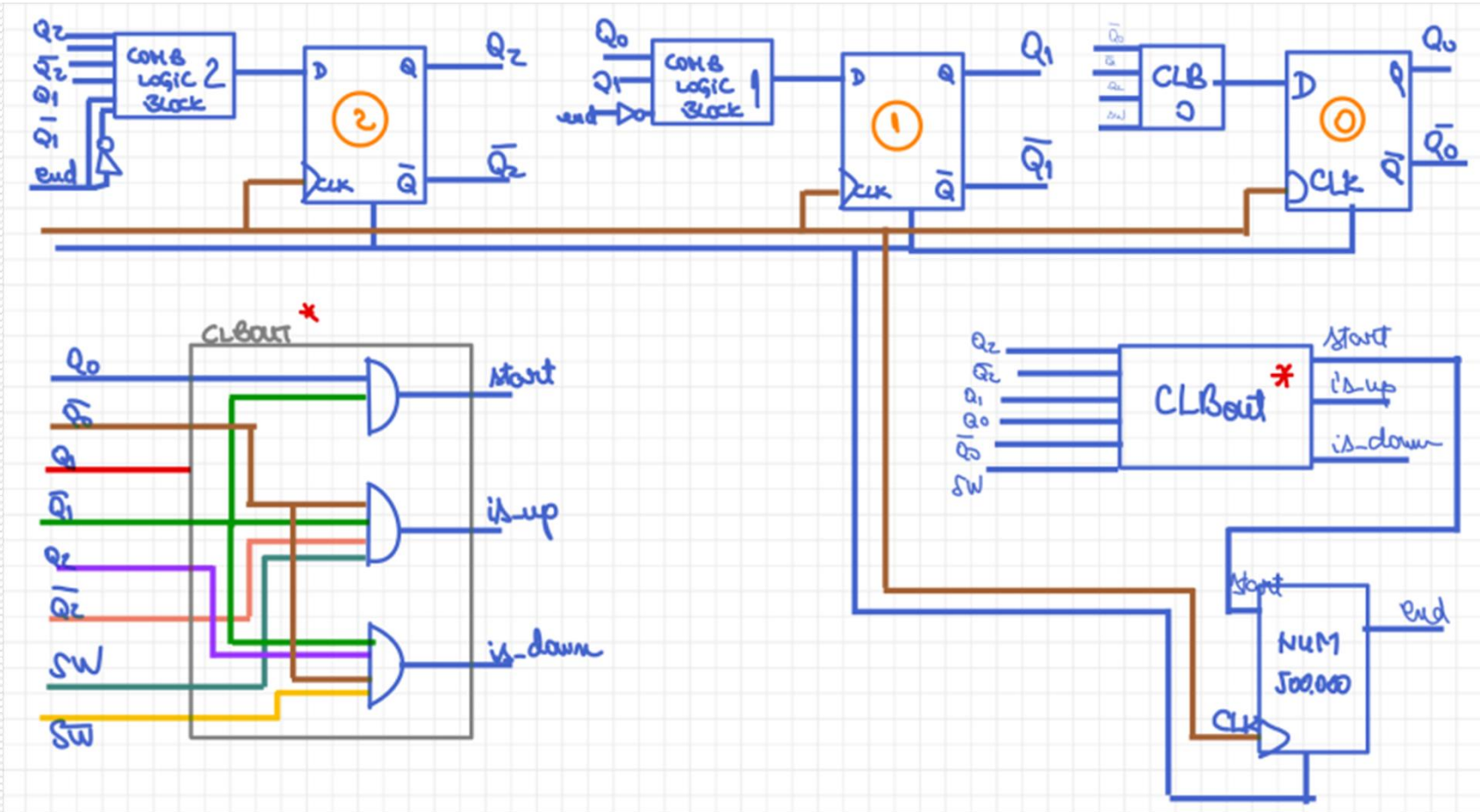


Circuit


SW(sync)
Edge
Detecter
START
CNT
END
is_up
START
is_down
Delay
CNT 10ms
CNT_MAX=5·10^5
END
CLK
RST

	STARE CURENTĂ			INTRĂRI		STARE URMĂTOARE = (pt D-uri) VAL INTRĂRI			IEȘIRI		
	Q_2	Q_1	Q_0	SW	END	$D_2(Q_2^{NXT})$	$D_1(Q_1^{NXT})$	$D_0(Q_0^{NXT})$	start	is_up	is_down
SW_OFF	0	0	0	0	0	0	0	0	0	0	0
				0	1	0	0	0			
				1	0	0	0	1	0	1	0
				1	1	0	0	1			
ONE	0	0	1	*	*	0	1	0	1	0	0
WAIT0	0	1	0	*	0	0	1	0	0	0	0
				*	1	1	0	0	0	0	0
SW_ON	1	0	0	0	*	1	0	1	0	0	1
				1	*	1	0	0	0	0	0
ZERO	1	0	1	*	*	1	1	0	1	0	0
WAIT1	1	1	0	*	0	1	1	0	0	0	0
				*	1	0	0	0	0	0	0
	0	1	1	*	*	d	d	d	d	d	d
	1	1	1	*	*	d	d	d	d	d	d

Edge detector (FSM) block impl



Enough Talking Let's Get To It
!!Brace Yourselves!!