Sisteme logice secventiale complexe

Unitate de executie
Unitate de comandă
Microprogramare

S.l. Dr. Ing. Vlad-Cristian Miclea

Universitatea Tehnica din Cluj-Napoca
Departamentul Calculatoare

CUPRINS

PLAN CURS

- Partea 1 – VHDL
  1. FPGA
  2. Limbajul VHDL – 1
  3. Limbajul VHDL – 2
  4. Limbajul VHDL – 3
- Partea 2 – Implementarea sistemelor numerice
  5. **Realizarea unui sistem numeric complex; Unitate de executie**
  6. **Unitate de comanda; Microprogramare**
- Partea 3 – Automate
  7. Automate finite
  8. Stari
  9. Automate sincrone
  10. Automate asincrone
  11. Identificarea automatelor
  12. Automate fara pierderi
  13. Automate liniare
- Partea 4 – Probleme si discutii

CONTEXT

Sisteme numerice sincrone complexe

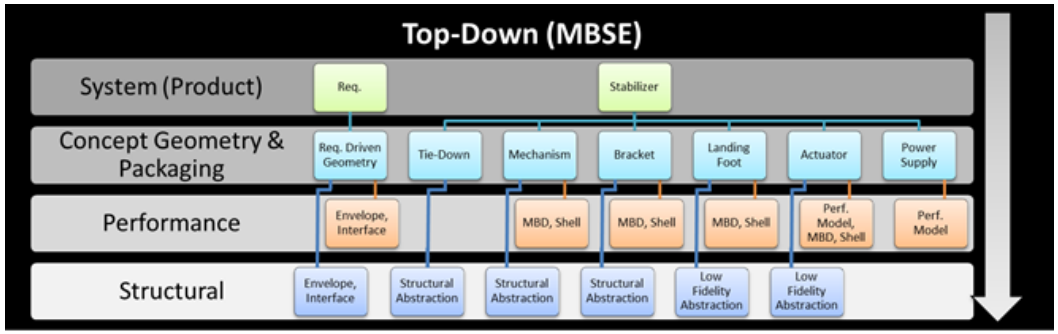
- Realizarea unor sisteme digitale complexe
- Se va discuta abordarea, definirea, analiza, implementarea si testarea onor sisteme hardware complexe
- Exemplu: un cuptor de gatit
- Se va discuta realizarea componentelor principale: UC si UE
- Unitate de executie
  - Numaratoare, registre, MUX-uri, ALU
  - Trebuie accesate la momentul potrivit
- Unitate de comanda/control
  - Generare semnale de control
  - Generarea urmatoarei stari
  - Unitate cablata (hardwired)
  - Microprogramare

Definirea problemei

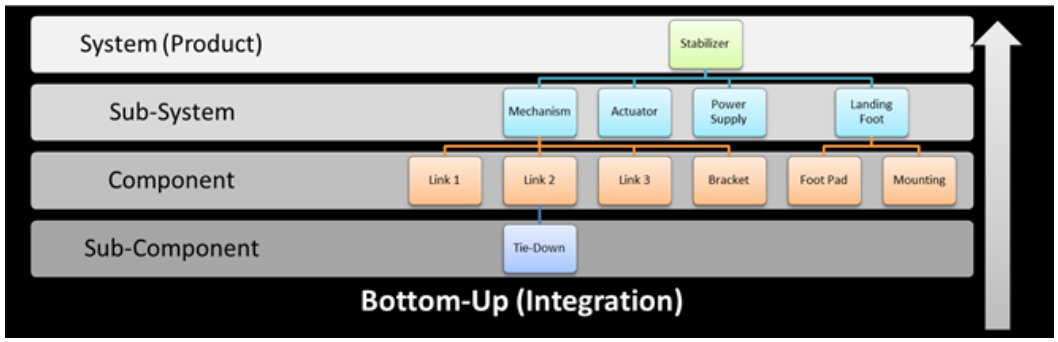
- Cuptor de gatit – exemplul de la laborator (putin mai complicat)
  - Gateste la 4 temperaturi predefinite: 180, 200, 220, 240 grade
    - preincalzire
    - In fiecare secunda, temperatura creste cu 10 grade;
  - Gateste 25 de minute
  - La sfarsit, se poate extinde perioada cu o valoare de baza de 50 de minute, la care se poate aduna sau reduce 5, 8 sau 12 min fata de perioada de gatire extra
- Detalii:
  - Se asteapta pana cand se apasa buton “Start”
  - Daca Start, se asteapta alege temp – se asteapta introducerea temp (V0,V1,V2,V3);
  - Se preincalzeste cuptorul; Apare un led “Preincalzire”
  - Apoi se stinge “Preincalizre”, se aprinde “InsertMancare” (se sta maxim 5 min)
  - Daca se introduce mancarea, se apasa “Buton_gatire” si se asteapta 25 min (se afiseaza “Gatire”)
  - Dupa finalizare gatire, se asteapta extindere gatire
  - Daca se doreste extra-gatire, se alege optiunea dorita
  - Se calculeaza timpul de extra-gatire
  - Se asteapta timpul de gatire suplimentare; Apare un led “Extra_gatire”
  - La final, se stinge ledul “Gatire” si Extra-gatire si se asteapta un nou proces

Abordarea problemei

- Top-down
  - Se incepe de la “cutia neagra” a sistemului
  - Se cauta componentele conceptuale
  - Pe baza lor, se gandesc componente Implementabile
  - Daca exista componente complexe, ele vor fi la randul lor separate in sub-componente



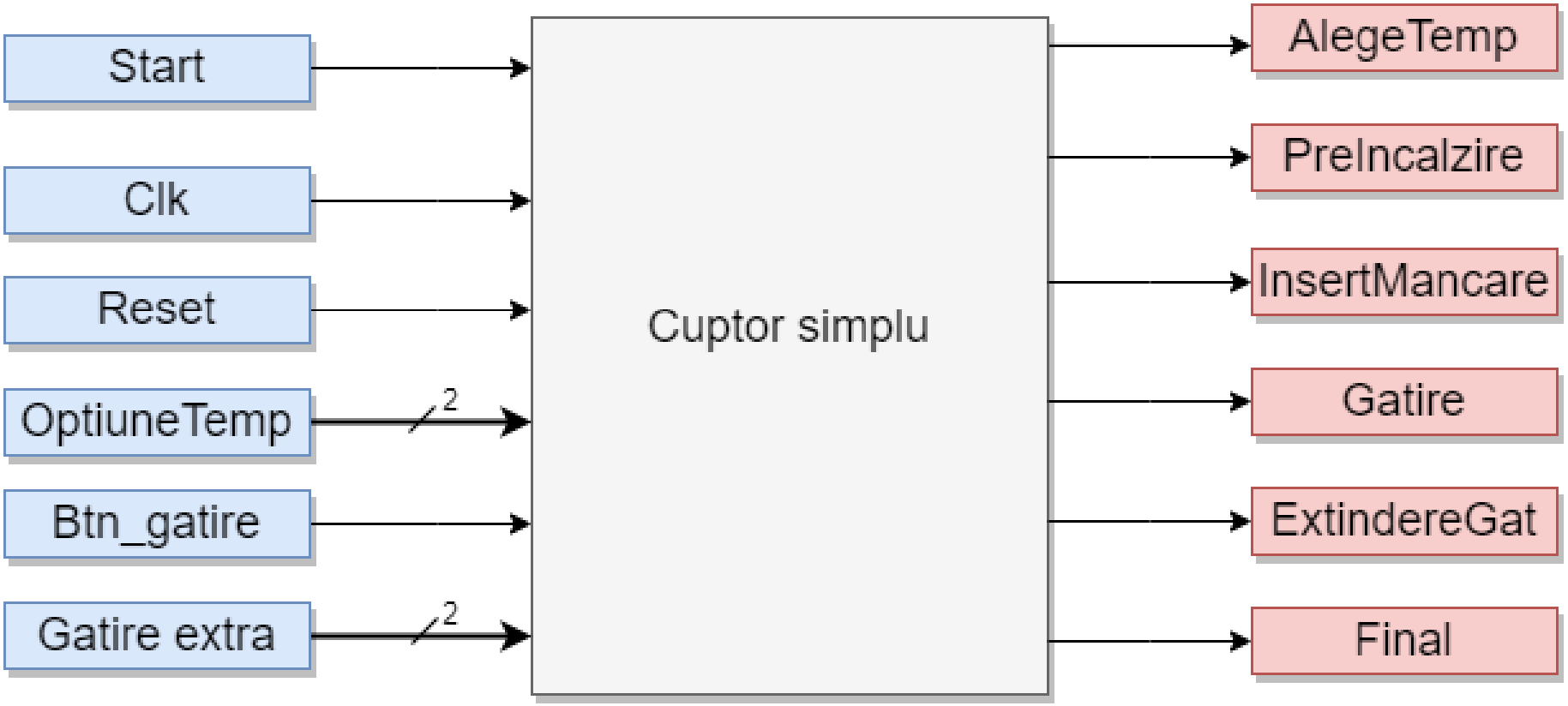
- Bottom-up
  - Se vor realiza sub-componentele simple (ex. porti logice)
  - Pe baza lor, se vor construi componentele necesare



  - Componentele vor fi integrate conform componentelor conceptuale
  - Se vor realiza legaturile intre componente
  - Vom realiza sistemul final, care trebuie sa corespunda formal cutiei negre realizata initial

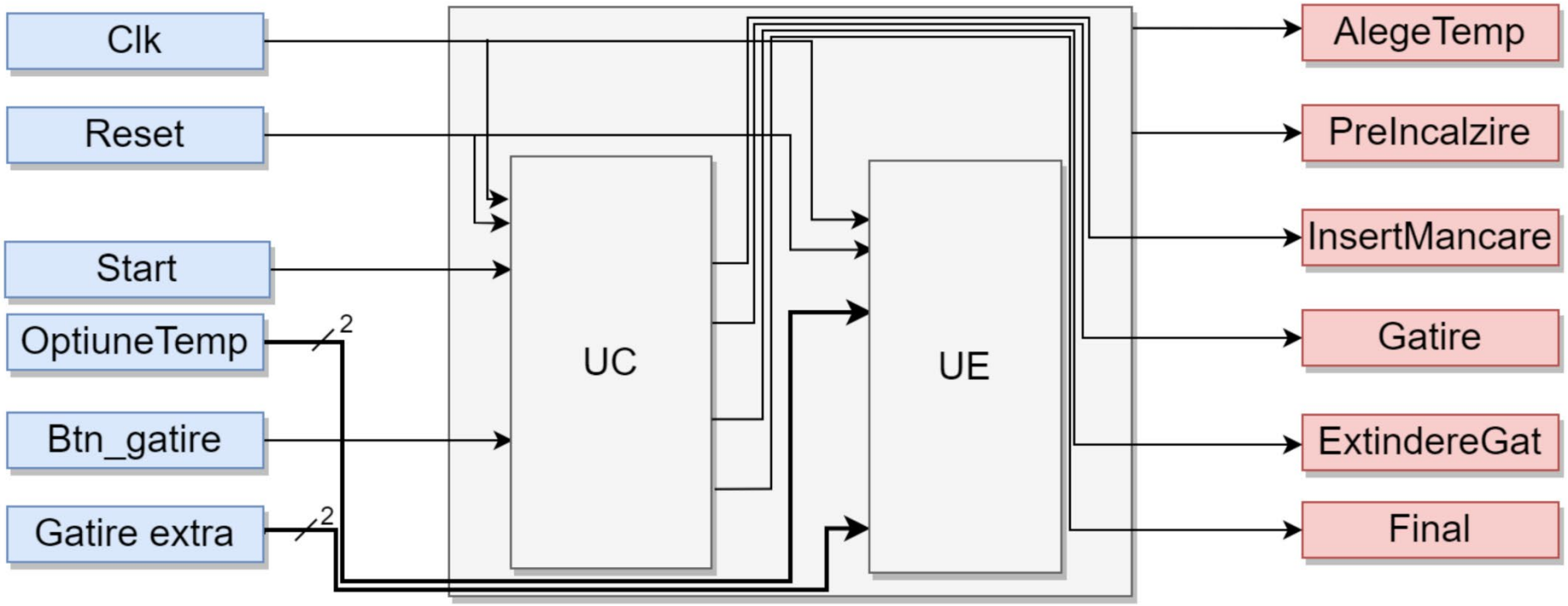
Schema bloc

- Evidentiaza intrarile si iesirile sistemului
- Pot exista intrari/iesiri care sa nu fie evidente (ex buton pt validare date)
- Se identifica use-case-urile sistemului
  - Se identifica pas-cu-pas fiecare etapa prin care trece sistemul
  - Se vor descoperi eventualele actiuni ascunse
  - Pot fi adaugate eventuale semnale de intrare/iesire



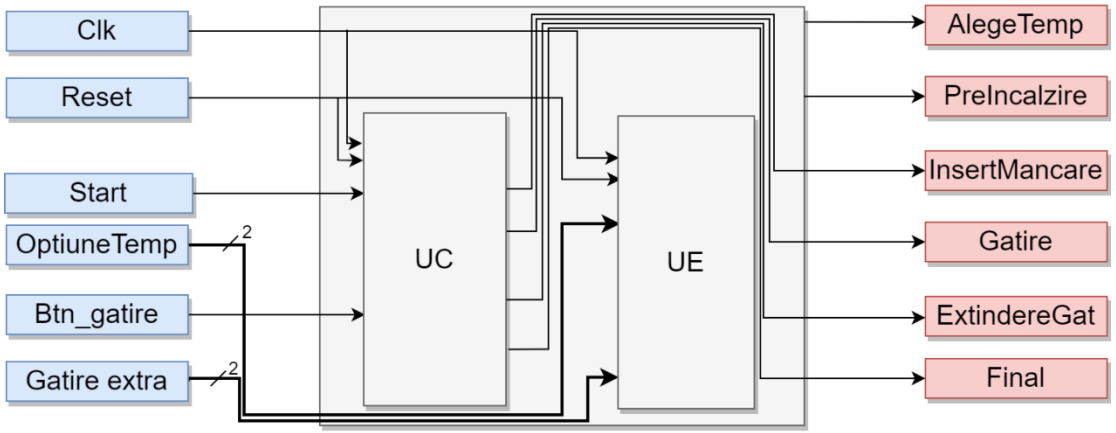
Schema bloc - componente

- Prima divizare conceptuala a sistemului: UC vs UE
- Unitate de comanda/control
  - Logica de control din sistem
- Unitate de executie
  - Resursele necesare pentru system
  - Componentele (alcatuite eventual din sub-componente)



Schema bloc - componente

- Separarea semnalelor - necesara
- Semnale de date
  - Intrari:
    - Valori necesare pt anumite lucruri
    - Adrese, timpi, cost-uri etc.
  - Iesiri:
    - Valori de afisat pt utilizator
    - Timp ramas, temperatura curenta etc.
  - Pentru sistemul nostru...
- Semnale de control
  - Intrari:
    - Butoane de confirmare, butoane de anulare
  - Iesiri:
    - Avertizarea utilizatorului – indrumarea lui pentru pasul current
    - Led-uri, semnale sonore etc.
  - Pentru sistemul nostru...



Resurse necesare

- Trebuie definite legaturile dintre UC si UE
- Resursele necesare pentru a lua decizii
- Pattern f intalnit:
  - UC transmite un enable sau reset pentru a activa/dezactiva o resursa
  - Resursa (parte din UE) genereaza semnale spre UC pentru a lua decizii
  - Orice informatie pe baza careia se ia o decizie trebuie sa vina de la o Resursa
- Resursele
  - Circuite simple, care pot fi implementate direct
    - Numaratoare, MUX-uri, Registre, Memorii
  - Resurse complexe (algoritmi)
    - Apar in prima descompunere ca niste cutii negre mai mici
    - Trebuie descompuse mai departe in sub-componente (pot avea inclusive sub-UC)
- O resursa poate genera iesiri spre utilizator
  - Ex: Putem vizualiza timpul de gatire
  - Afisarea valorii de la un numarator

UNITATEA DE EXECUTIE

Sinteza unităţii de execuţie UE

- se bazează pe folosirea unui limbaj de descriere
- conduce la realizarea cablată a unităţii de execuţie
- realizarea rezultă din interconectarea componentelor combinaţionale şi secvenţiale disponibile sub formă de circuite integrate sau realizate in VHDL
- se folosesc în principal:
  - Memorii ROM, multiplexoare şi unităţi aritmetice şi logice pentru componentele combinaţionale
  - numărătoare şi registre pentru componentele secvenţiale

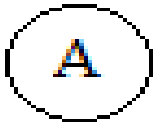
METODA GENERALĂ DE SINTEZĂ

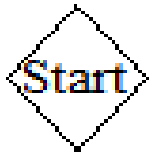
Etapele sintezei UE

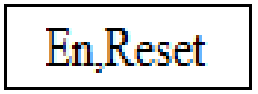
- 1. declararea a resurselor UE, alaturi de descrierea funcţională a sistemului numeric, cu ajutorul unei organigrame în care limbajul de descriere se aplică registrelor şi resurselor UE
- 2. construirea schemei UE şi declararea eventualelor registre şi a resurselor adiţionale
- 3. realizarea UE cu ajutorul componentelor combinaţionale şi secvenţiale MSI disponibile sau implementarea lor folosind limbajul VHDL

Organigrama (digrama de stari)

- Notatii:
  - Stari – cercuri
    - stare reprezinta un moment de timp (o perioada)
  - Decizii – romb-uri
    - Intrari in sistem, pot veni din exterior sau de la componente
  - Afisari – dreptunghi-uri
    - Iesiri din sistem, pot merge spre exterior sau spre alte componente
  - Iesirile pot fi inainte, sau dupa decizii
  - De obicei, fiecare iesire/decizie apare dupa o stare
    - Pot fi iesiri care sa depinda doar de stare
    - Pot fi iesiri care sa depinda de decizii
- Tabla/drawio – realizata impreuna

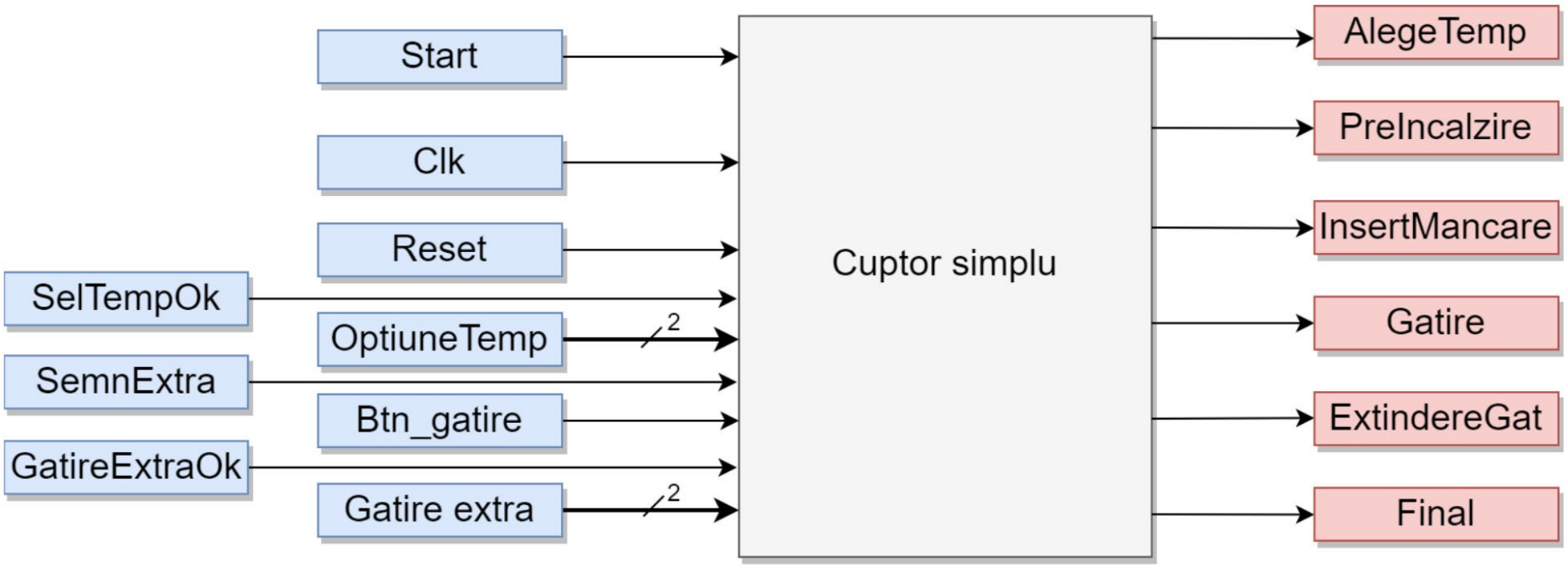






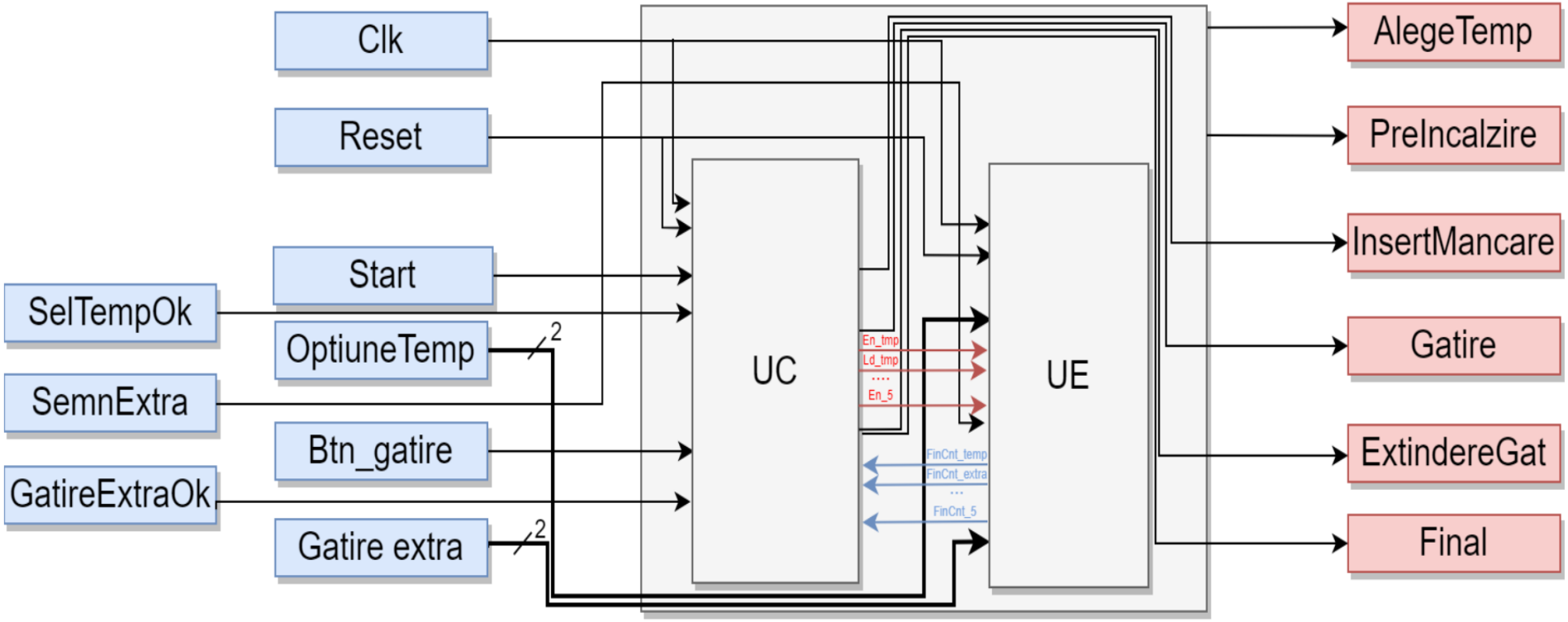
Schema bloc (iteratia 2)

- Pe baza organigramei – ne dam seama de noile intrari/iesiri
- Pot exista intrari/iesiri care sa nu fie evidente (ex buton pt validare date)
  - *SelTempOk* – buton pentru validarea seletiei temperaturii
  - *GatireExtraOk* – buton pentru validarea extinderii temperaturii
  - *SemnExtra* – creste/descreste timpul de gatire – poate fi un sw
- Pot fi si altele, gasite la o noua iteratie



Componente principale (iteratia 2)

- Apar semnalele de control
  - De la UC spre UE
  - Rol de enable, load, reset sau de transmisie date
- Semnalele de raspuns
  - De la UE spre UC
  - Finalizare numarare



Resurse necesare - cuptor

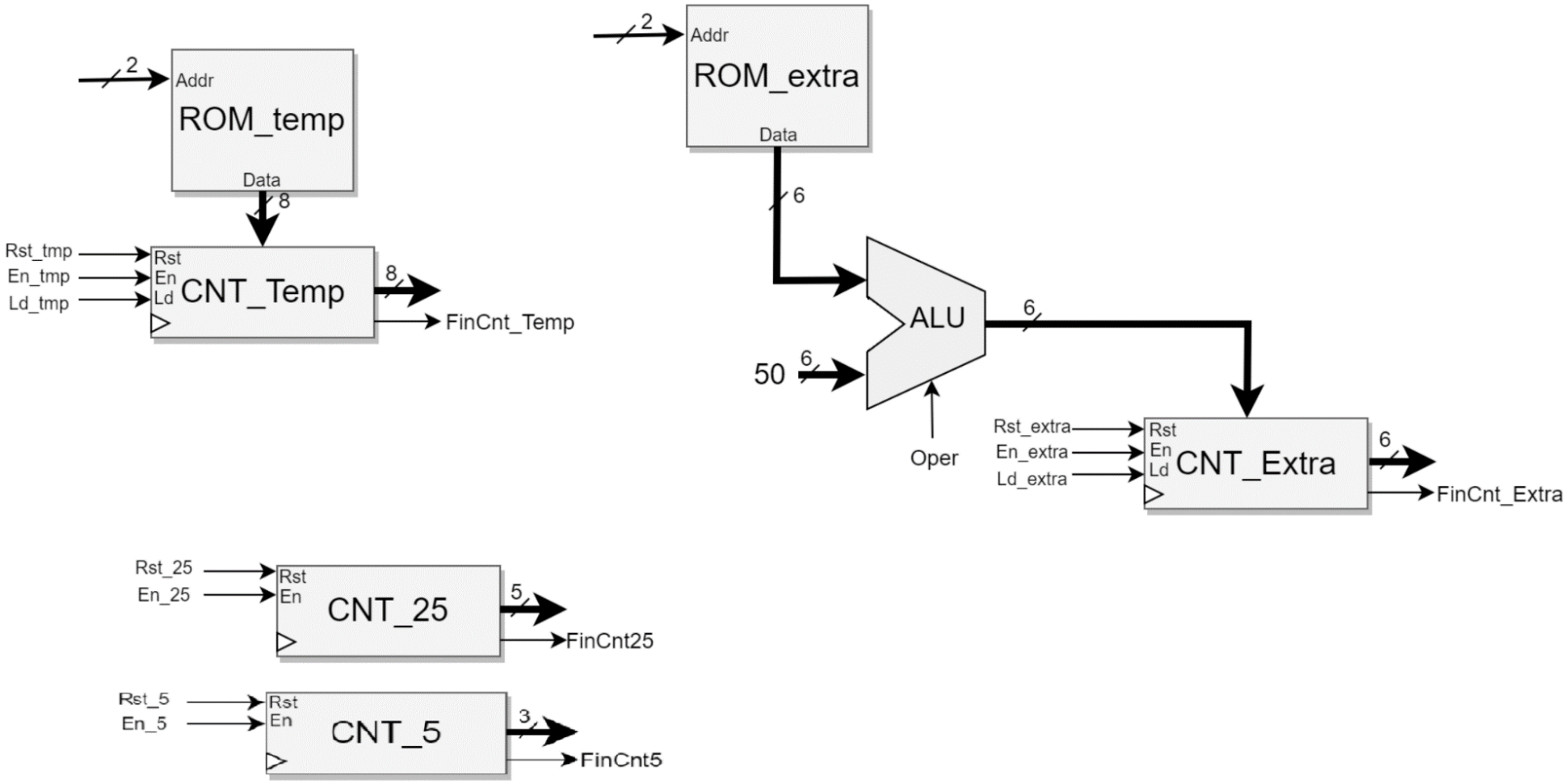
- Numarator pentru temperatura (max 240C) => 8 biti
  - Poate fi pornit (En), resetat (Rst) sau incarcat cu o valoare predefinita (Ld)
- Numarator pentru gatire (25min) => 5 biti
  - Poate fi pornit (En) sau resetat (Rst)
- Numarator pentru asteptare gatire (5min) => 3 biti
  - Poate fi pornit (En) sau resetat (Rst)
- ALU (sumator/scazator) pentru generare timp Extra-gatire
  - max 50+12 = 62 => 6 biti
  - Are nevoie de un semnal pentru stabilire operatie
- Numarator pentru extra-gatire (max 50+12) => 6 biti
  - Poate fi pornit (En), resetat (Rst) sau incarcat cu o valoare predefinita (Ld)

SCHEMA ŞI DECLARAŢIA ADIŢIONALĂ

Resurse

- schema UE interconectează resursele alese conform organigramei
- schema poate eventual să conţină resurse adiţionale
- schema UE pentru cuptor face apel la 2 resurse suplimentare:
  - ROM_tmp_{4x8} = memorie ROM pentru stocarea temperaturilor posibile
  - ROM_extra_{4x6} = memorie ROM pentru stocarea extinderile posibile

SCHEMA UE - CUPTOR



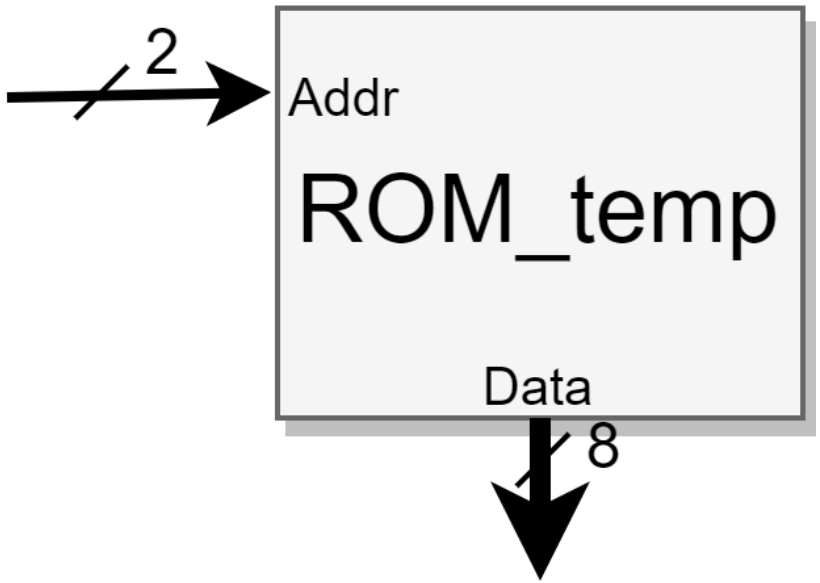
REALIZAREA SCHEMEI UE

Realizare

- alegerea circuitelor fizice pentru resurse şi registre, pentru a se efectua operaţiile din organigramă şi a interconecta variabilele de informaţie conform schemei
- se pun în evidenţă pentru registre şi resurse:
  - operaţiile care trebuie efectuate
  - integratele utilizate, cu funcţiile lor
  - schema logică de interconectare

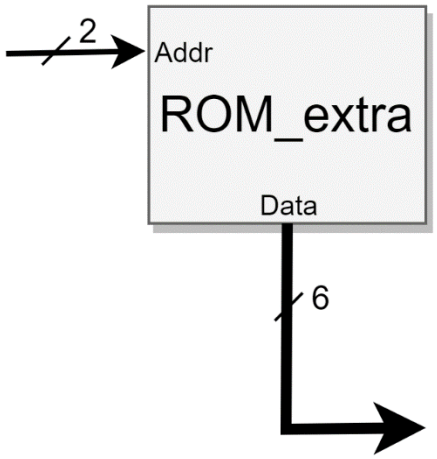
Resursa Memorie ROM_temp

- Pastreaza cele 4 temperaturi pentru gatit
- 180, 200, 220, 240 – pot fi pastrate pe 8 biti
- Nevoie de 4 adrese => 2 biti pentru adresare
- Vom avea o memorie de 2^2 x 8b (capacitate = 32b)
- Structura memoriei (zecimal):
  - ROM(0) – 180
  - ROM(1) – 200
  - ROM(2) – 220
  - ROM(3) – 240



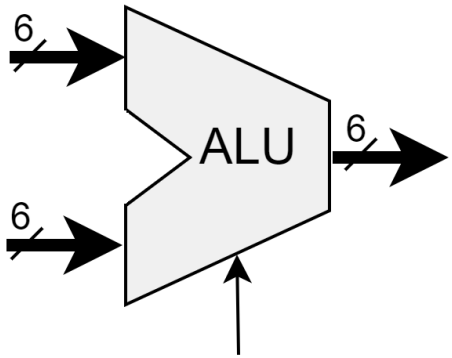
Resursa Memorie ROM_extra

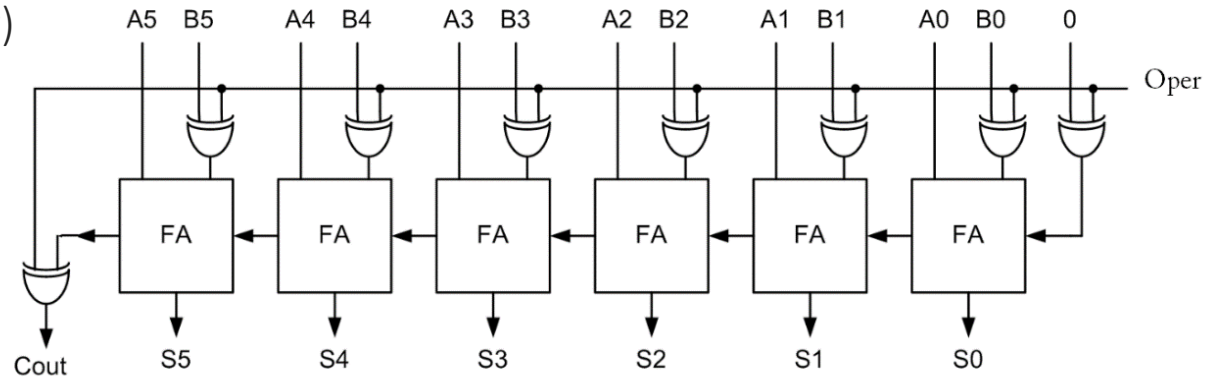
- Pastreaza cele 3 valori de timp pentru extra-gatit
- 5, 8, 12– pot fi pastrate pe 4 biti;
- Vor trebui adunate cu 50 => total pe 6 biti
  - pastram si valorile curente pe acelasi nr de biti, pentru simplitate
- Nevoie de 3 adrese => 2 biti pentru adresare
- Vom avea o memorie de 2^2 x 6b (capacitate = 24b)
- Structura memoriei:
  - ROM(0) – 000101
  - ROM(1) – 001000
  - ROM(2) – 001100
  - ROM(3) – neutilizata (XXXXXX)



Resursa ALU (sumator-scazator)

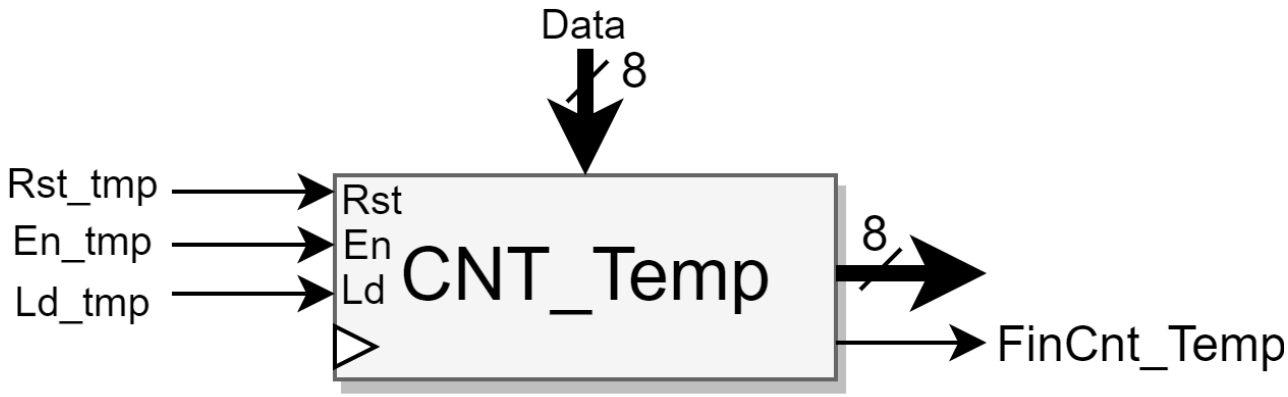
- Operatiile efectuate:
  - A+B
  - A-B
- Calculeaza timpul extra de gatit
- Va insuma/scadea din valoarea predefinita 50 valoarea citita din ROM_extra
- Se poate folosi un sumator-scazator pe 6 biti
  - Inversare B (poarta XOR cu Oper)
  - Oper (rol de carry in):
    - 0 pt adunare
    - 1 pt scadere
  - 6 Sumatoare complete (FA)





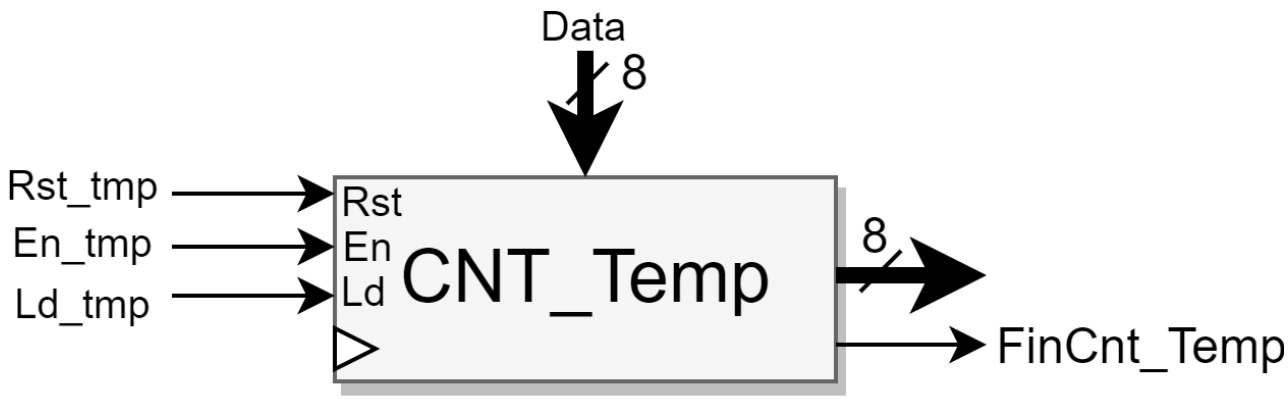
Operatie	Descriere	Oper
ADD	(C6,S5,S4,S3,S2,S1,S0)=(A5,A4,A3,A2,A1,A0) + (B5,B4,B3,B2,B1,B0)+(0,0,0,0,0,0)	0
SUBTRACT	(C6,S5,S4,S3,S2,S1,S0)=(A5,A4,A3,A2,A1,A0) + ($\overline{B5}$, $\overline{B4}$, $\overline{B3}$, $\overline{B2}$, $\overline{B1}$, $\overline{B0}$)+(0,0,0,0,0,1)	1

Resursa Numarator Temp



- Numarator descrescator pe 8 biti modulo *Data* (valoarea maxima data din ROM)
- Va avea o iesire *FinCnt*_Temp care va fi adevarata cand au trecut s-a ajuns la X grade, unde X este incarcat
  - Vom numara descrescator, incepand de la valoarea introdusa (pt simplificarea problemei)
  - *FinCnt_Temp* va fi generat cand numaratorul va ajunge la 0
    - va fi generat prin $\overline{Q7Q6Q5Q4Q3Q2Q1Q0}$ (00000000)
- Trebuie pornit la momentul potrivit – Nevoie de enable (En_tmp)
  - Functioneaza pe ceas, deci are nevoie de CLK
  - Are nevoie de reset, pentru a reincepe o numarare
- Trebuie incarcat (in momentul in care este prima data pornit)
  - Are nevoie de un Ld – care va fi activ DOAR la pornire

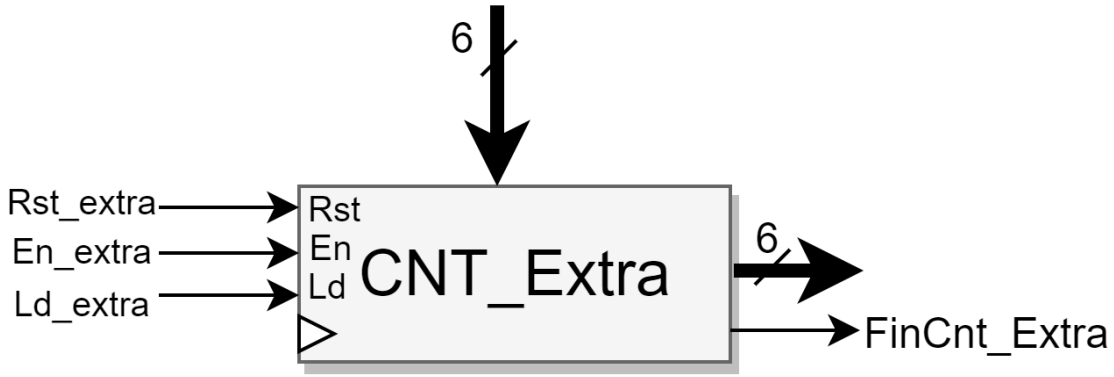
Resursa Numarator Temp



- Operatiile efectuate sunt:
  - NOP – inefectiva
  - CNT_Temp <- 0
  - CNT_Temp <- *Data*
  - CNT_Temp <- CNT_Temp - 1

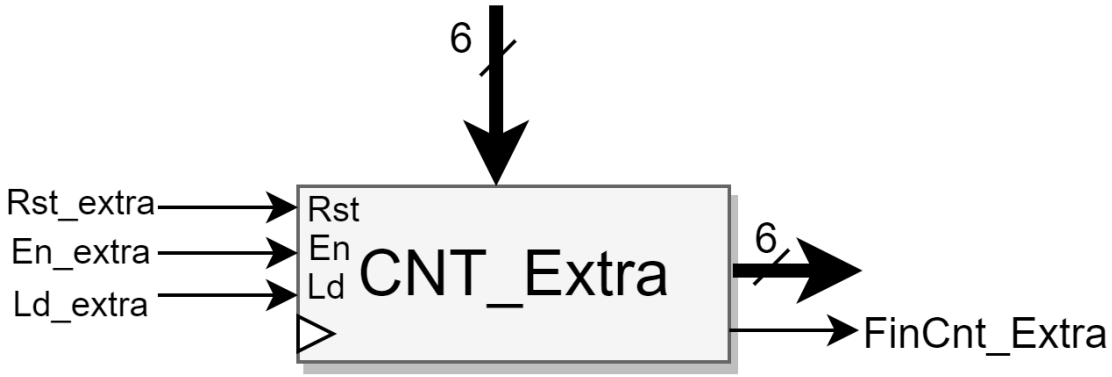
Oper	Descriere	Rst_t	Ld_t	En_t
Reset	(Q7,Q6,Q5,Q4,Q3,Q2,Q1,Q0)=(0,0,0,0,0,0,0,0)	1	X	X
Hold	(Q7,Q6,Q5,Q4,Q3,Q2,Q1,Q0)=(Q7,Q6,Q5,Q4,Q3,Q2,Q1,Q0)	0	0	0
Load	(Q7,Q6,Q5,Q4,Q3,Q2,Q1,Q0)=(D7,D6,D5,D4,D3,D2,D1,D0)	0	1	1
Count	(Q7,Q6,Q5,Q4,Q3,Q2,Q1,Q0)=(Q7,Q6,Q5,Q4,Q3,Q2,Q1,Q0) – (0,0,0,0,0,0,0,1)] mod 256	0	0	1

Resursa Numarator Extra



- Numarator descrescator pe 6 biti modulo *Data* (dat de date din ROM)
- Va avea o iesire *FinCnt*_Extra care va fi adevarata cand au trecut X minute, unde X este incarcat
  - *FinCnt_Extra* va fi generat cand numaratorul va ajunge la 0
  - va fi generat prin $\overline{Q5Q4Q3Q2Q1Q0}$ (000000)
- Trebuie pornit la momentul potrivit – Nevoie de enable (En_extra)
  - Functioneaza pe ceas, deci are nevoie de CLK
  - Are nevoie de reset, pentru a reincepe o numarare
- Trebuie incarcat (in momentul in care este prima data pornit)
  - Are nevoie de un Ld_extra – care va fi activ DOAR la pornire

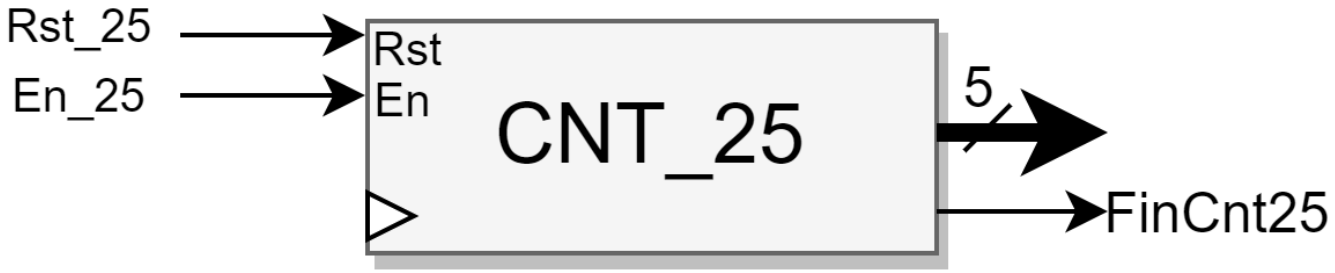
Resursa Numarator Extra



- Operatiile efectuate sunt:
  - NOP – inefectiva
  - CNT_Extra <- 0
  - CNT_Extra <- *Data_Extra*
  - CNT_Extra <- CNT_Extra - 1

Oper	Descriere	Rst_e	Ld_e	En_e
Reset	(Q5,Q4,Q3,Q2,Q1,Q0)=(0,0,0,0,0,0)	1	X	X
Hold	(Q5,Q4,Q3,Q2,Q1,Q0)=(Q5,Q4,Q3,Q2,Q1,Q0)	0	0	0
Load	(Q5,Q4,Q3,Q2,Q1,Q0)=(D5,D4,D3,D2,D1,D0)	0	1	1
Count	(Q5,Q4,Q3,Q2,Q1,Q0)=(Q5,Q4,Q3,Q2,Q1,Q0) – (0,0,0,0,0,1)] mod 64	0	0	1

Resursa Numarator 25 min



- Numarator pe 5 biti modulo 25
- Va avea o iesire *FinCnt*5 care va fi adevarata cand au trecut 5 minute
  - *FinCnt2*5 va fi generat prin $\mathrm{Q4Q3}\overline{Q2Q1}\mathrm{Q0}$ $(25 = 11001)$
- Trebuie pornit la momentul potrivit – Nevoie de enable (En_25)
  - Functioneaza pe ceas, deci are nevoie de CLK
  - Are nevoie de reset, pentru a reincepe o numarare
- Operatiile efectuate sunt:
  - NOP – inefectiva, CNT_25 <- 0 sau CNT_25 <- CNT_25 + 1

Operatie	Descriere	Rst_25	En_25
Reset	(Q4,Q3,Q2,Q1,Q0)=(0,0,0,0,0)	1	X
Hold	(Q4,Q3,Q2,Q1,Q0)=(Q4,Q3,Q2,Q1,Q0)	0	0
Count	(Q4,Q3,Q2,Q1,Q0)=[(Q4,Q3,Q2,Q1,Q0)+(0,0,0,0,1)] mod 32	0	1

Resursa Numarator 5 min

- Numarator pe 3 biti modulo 5
- Va avea o iesire *FinCnt*5 care va fi adevarata cand au trecut 5 minute
  - *FinCnt*5 va fi generat prin $Q2\overline{Q1}Q0$
- Trebuie pornit la momentul potrivit – Nevoie de enable (En_5)
  - Functioneaza pe ceas, deci are nevoie de CLK
  - Are nevoie de reset, pentru a reincepe o numarare
- Operatiile efectuate sunt:
  - NOP – inefectiva, CNT_5 <- 0 sau CNT_5 <- CNT_5 + 1

Operatie	Descriere	Rst_5	En_5
Reset	(Q2,Q1,Q0)=(0,0,0)	1	X
Hold	(Q2,Q1,Q0)=(Q2,Q1,Q0)	0	0
Count	(Q2,Q1,Q0)=[(Q2,Q1,Q0)+(0,0,1)] mod 8	0	1

Resurse - extra

- Resurse necesare ca sistemul sa functioneze pe FPGA
- Divizor de frecventa
  - Toate operatiile sunt in minute
  - Ceasul de pe FPGA – perioada de 10 ns
  - Nevoie de divizare – aplicare ceas divizat peste tot
- Generare de monoimpuls pentru butoane
  - Nu putem conecta un buton la ceas
  - Butoanele – imperfect – nevoie sa luam impulsul o singura data
- Afisoare 7 segmente (SSD)
  - Valorile de pe numaratoare ar trebui afisate
  - Cea mai buna optiune: SSD
  - Nevoie de un circuit special pentru generarea de anod si catod