

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Specializarea: Calculatoare și Tehnologia Informației

Disciplina: Proiectarea Sistemelor Numerice

Titlul Proiectului: **CALCULATOR DE BUZUNAR**

Grupa: 30212

Profesor îndrumător:                                                                                            Student

As. ing. Pop Diana                            Gavrilă Tiberiu-Valentin

:

CUPRINS

1. Specificația proiectului.................................................................................

1. Descrierea schemei bloc cu componentele sale.......................................
2. Proiectare și implementare...........................................................................

* Descrierea schemelor de detaliu
* Proiectarea componentelor
* Proiectarea de ansamblu
* Simularea în ActiveHDL
* Fișier de constrângeri

1. Componentele utilizate si semnalele lor..................................................
2. Justificarea soluției alese.............................................................................
3. Utilizare și rezultate...................................................................................

* Resursele necesare
* Manual de instrucțiuni
* Rezultatele obținute pe plăcuța FPGA

1. Posibilități de dezvoltare ulterioară ........................................................

**1.Specificatia Proiectului**

Calculatorul de buzunar realizează operațiile aritmetice de bază (adunare, scădere, înmulțire, împărțire (cu rest)) cu numere întregi reprezentate prin mărime și semn.

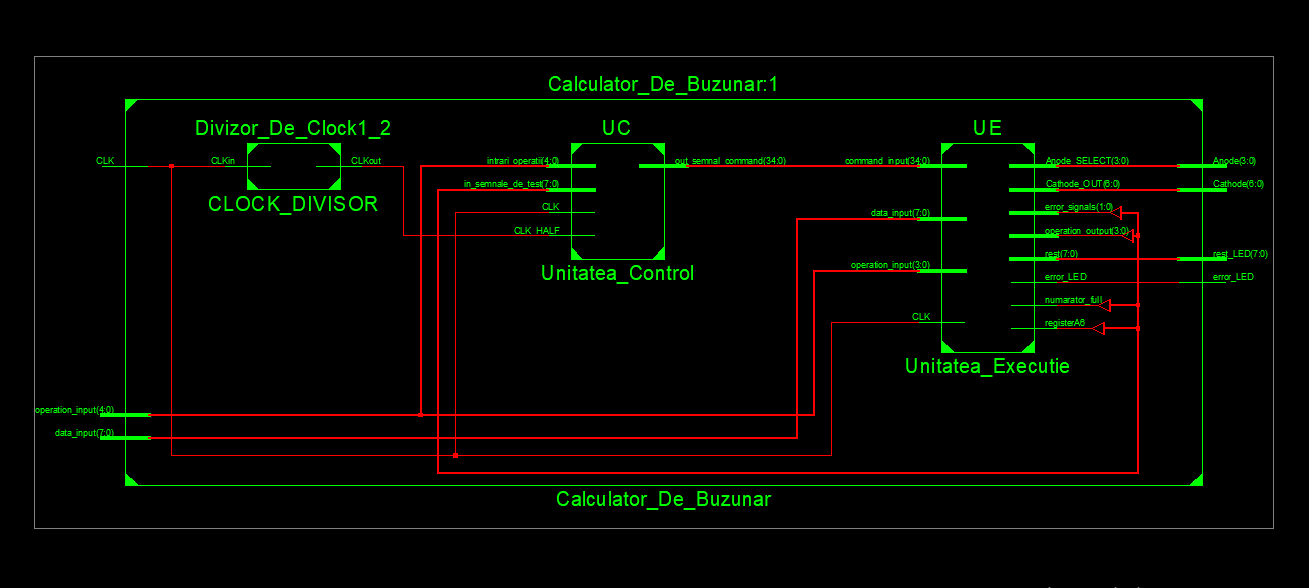
Rezultatul unei operații poate fi folosit în operația următoare, și există un semnal dedicat care aprinde un LED dacă are loc o eroare în efectuarea operației (overflow, sau impărțirea cu 0).

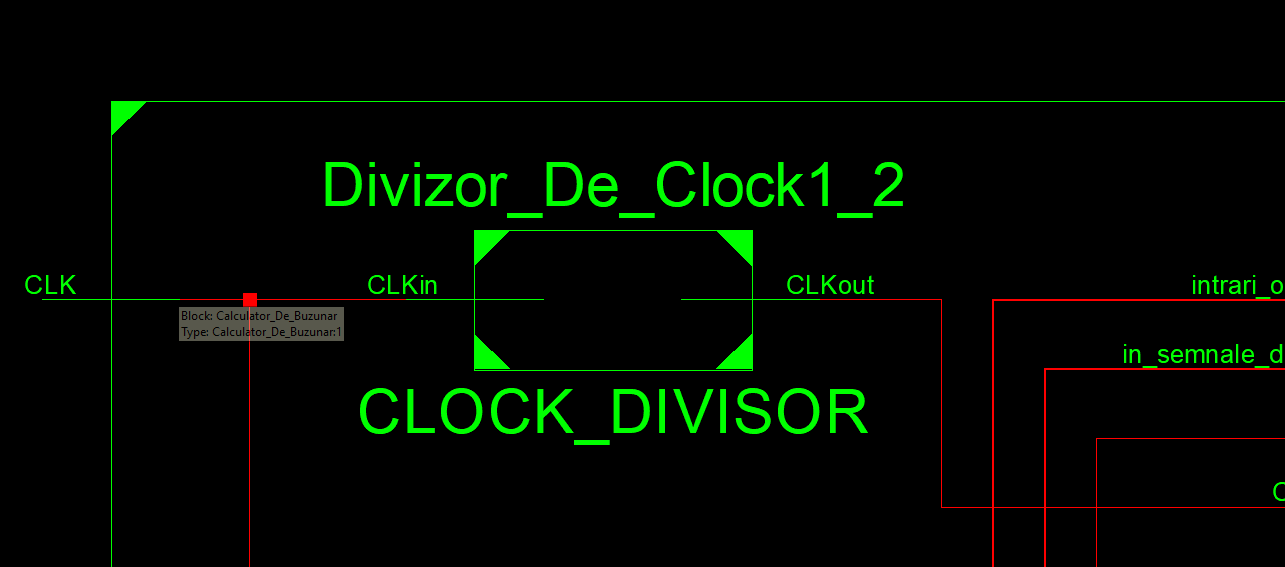
Calculatorul e conceput pentru rularea pe plăcile Nexys4DDR, însă s-a ținut cont de modele mai vechi, astfel că restul împărțirii este afișat sub forma lui de mărime și semn folosind LED-urile plăcii.

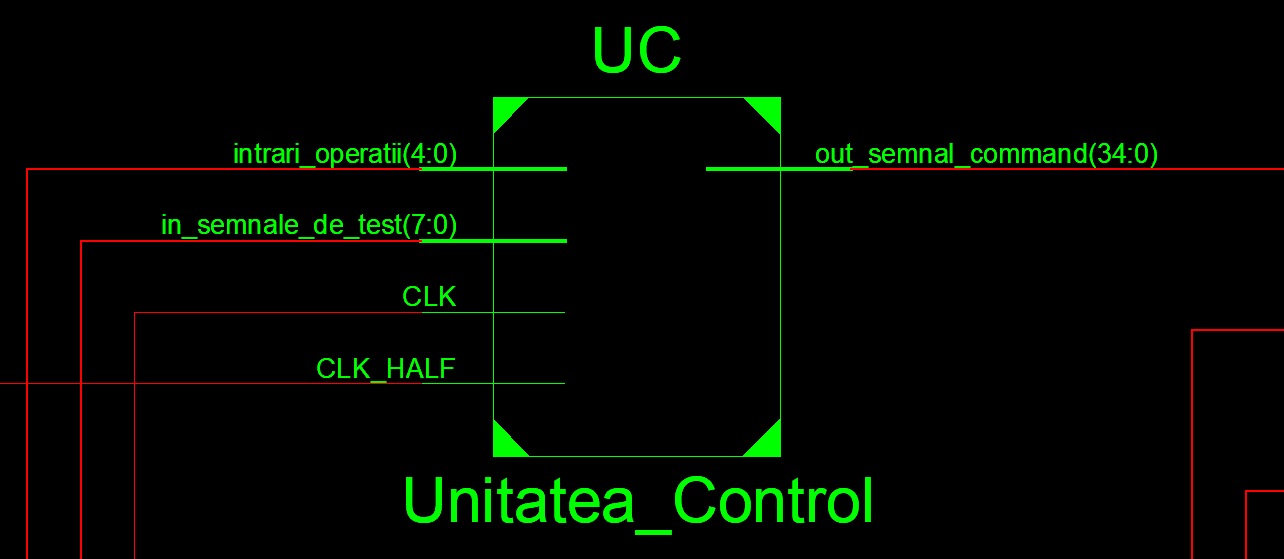
Am ales o Unitate de Control microprogramata pentru asemănarea (din perspectiva mea) cu programarea în limbaj de asamblare. Deși ROM-ul UC poate fi meticulos de realizat, consider că e modalitate de proiectare simplă de înțeles.

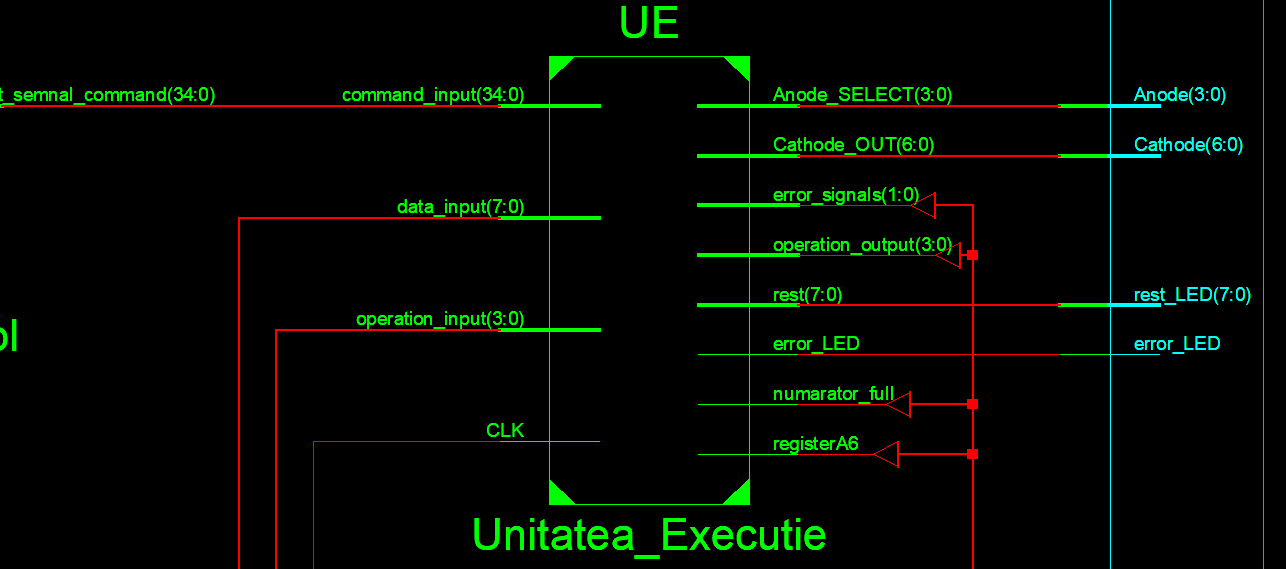
**2.Descrierea schemei block**

**cu componentele sale**







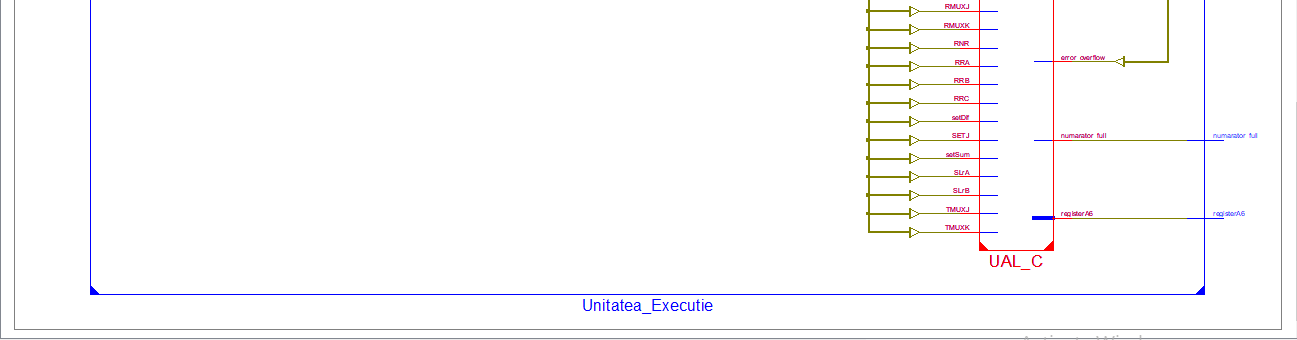
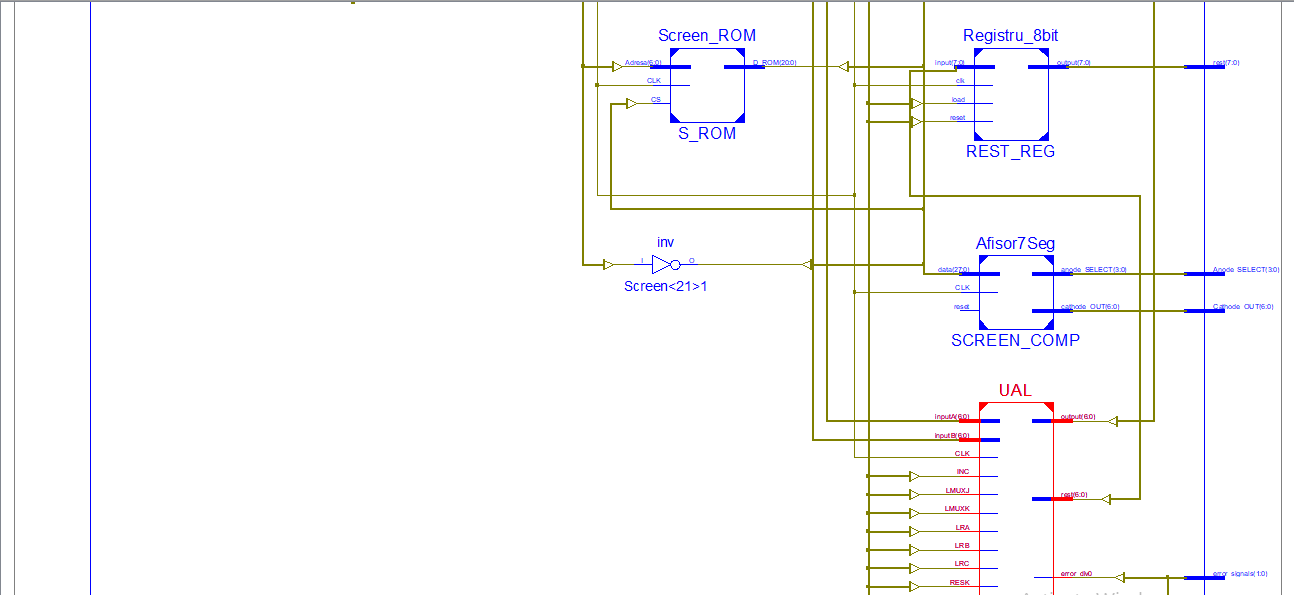
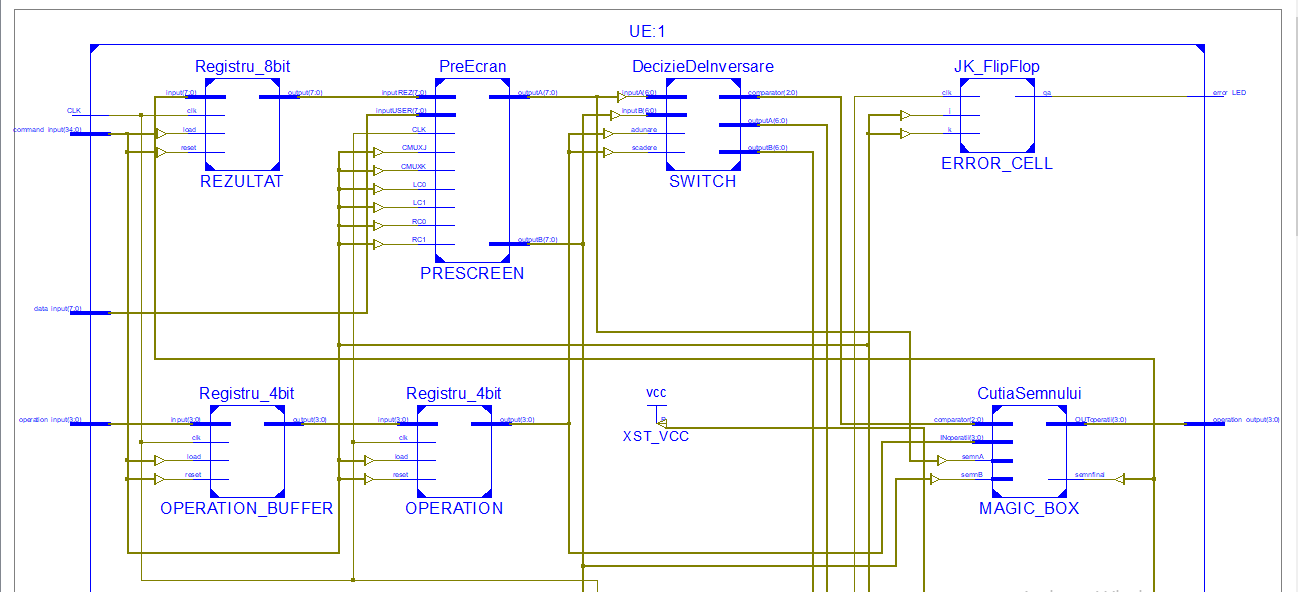


Intrările Calculatorului de buzunar sunt doar un bus de 8 Biți pentru introducerea numerelor prin mărime și semn, și un bus de 5 Biți pentru introducerea operațiilor.

**3.Proiectare si Implementare**

***Unitatea De Execuție:***

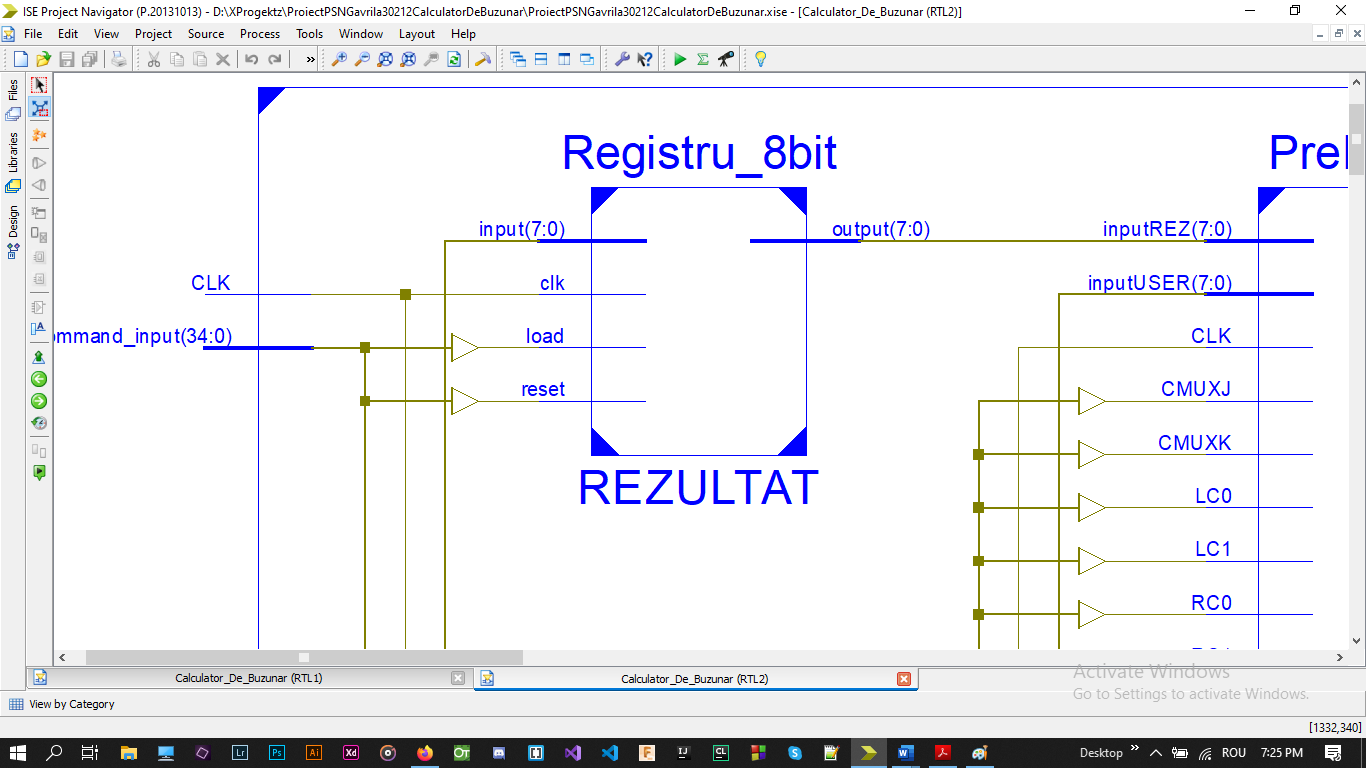
Deși unitatea de execuție este controlată de cealaltă unitate. UE realizează trei lucruri automat, prin intermediul CLC: calculează semnul rezultatului, și în caz de adunare sau scădere, ordonează operanzii astfel încât primul să fie acela mai mare și aduce operanzii la același semn, lucru care face ca operațiile din UAL să fie cu modulul numerelor, de 7 biti, care simplifică proiectarea UC.



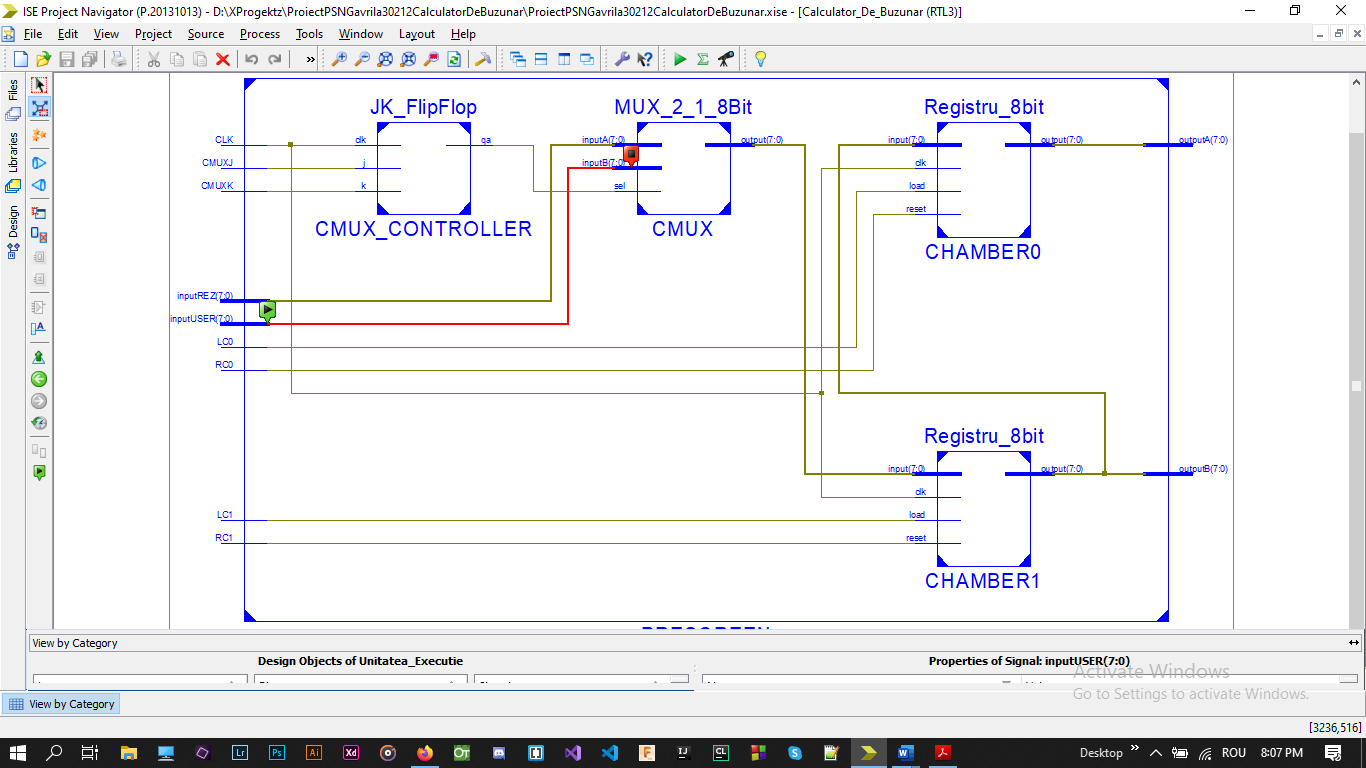
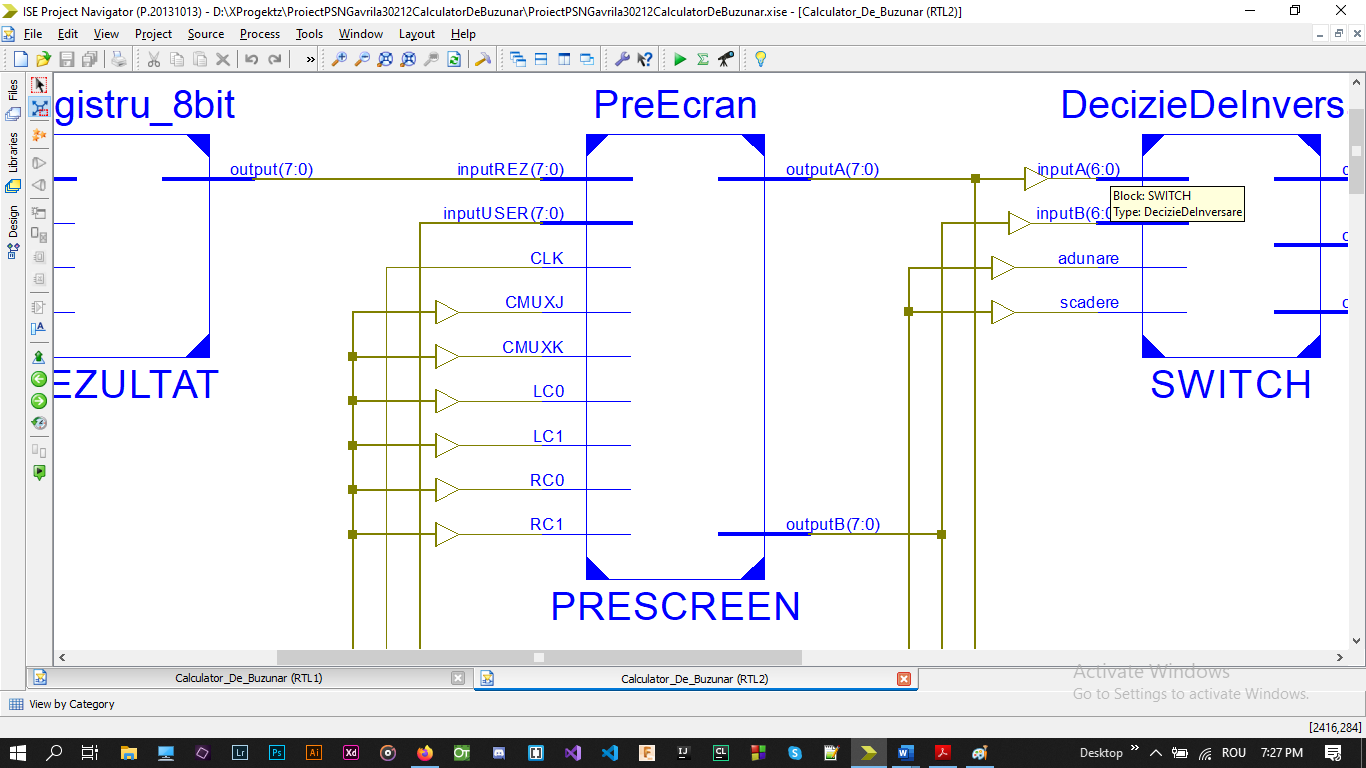
Prin „aduce operanzii la același semn” mă refer la cazurile în care semnul operanzilor schimbă operația care are loc. De exemplu 5-(-4) e de fapt 5+4 sau -3-6 este (-3)+(-6).

Și astfel chiar dacă utilizator introduce o scădere, de exemplu, în funcție de semnul operanzilor, inter poate să se realizeze o adunare de fapt.

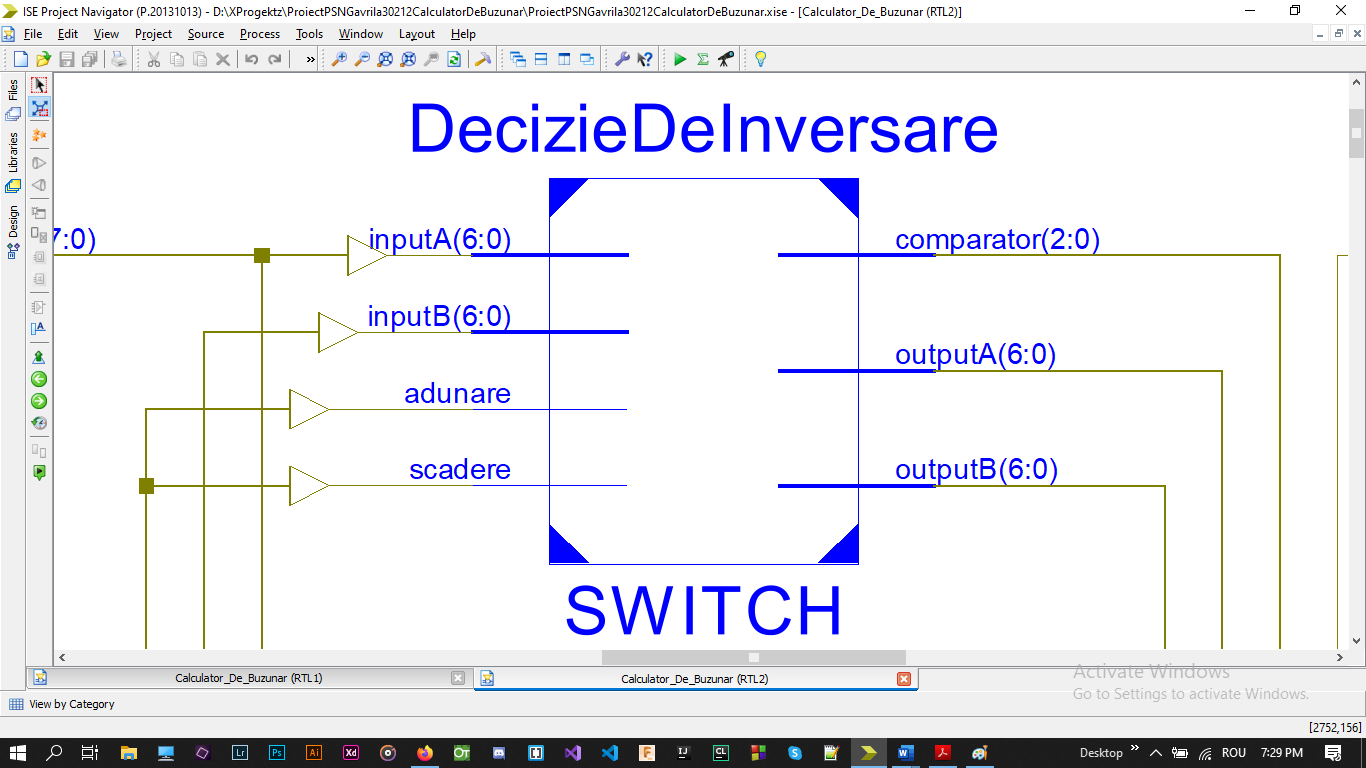
* *Registru\_8Bit „Rezultat” – Reține rezultatul operației din UAL*

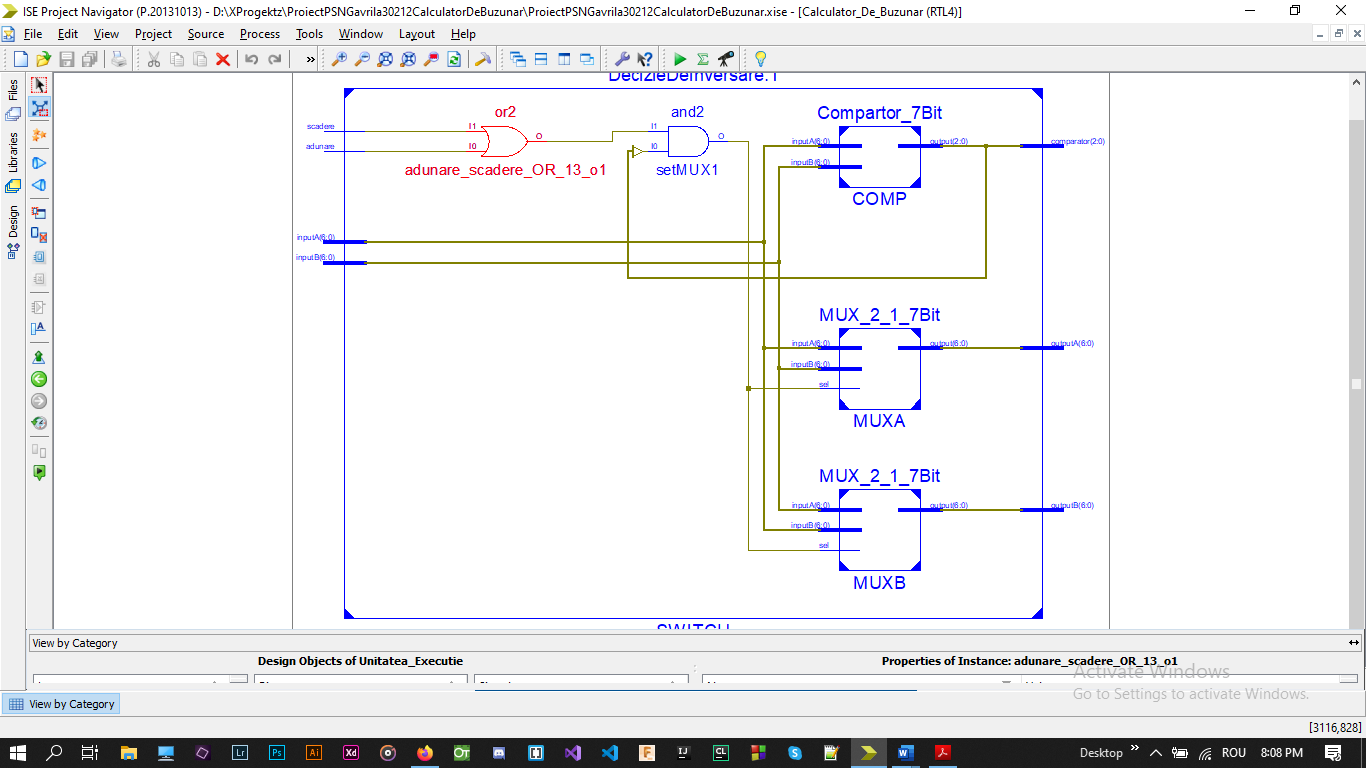


* *PreEcran: Reține operanzi sub forma lor de mărime și semn, citindu-i fie de la utilizator fie de la UAL (fiind rezultatul unei operații care poate fi folosit în altă operații).*



* *DecizieDeInversare: dacă operația care se efectuează e o adunare sau o scădere, se asigură că primul operand e acela mai mare*

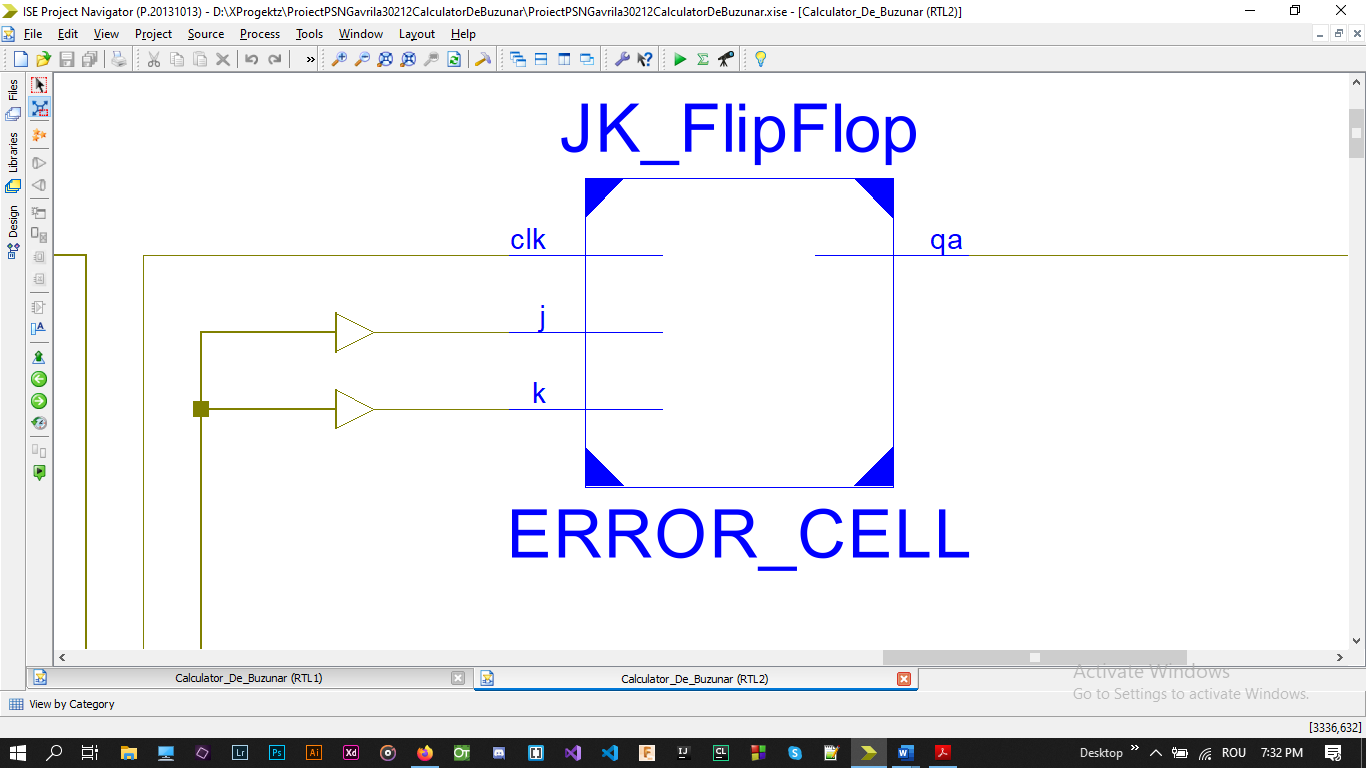




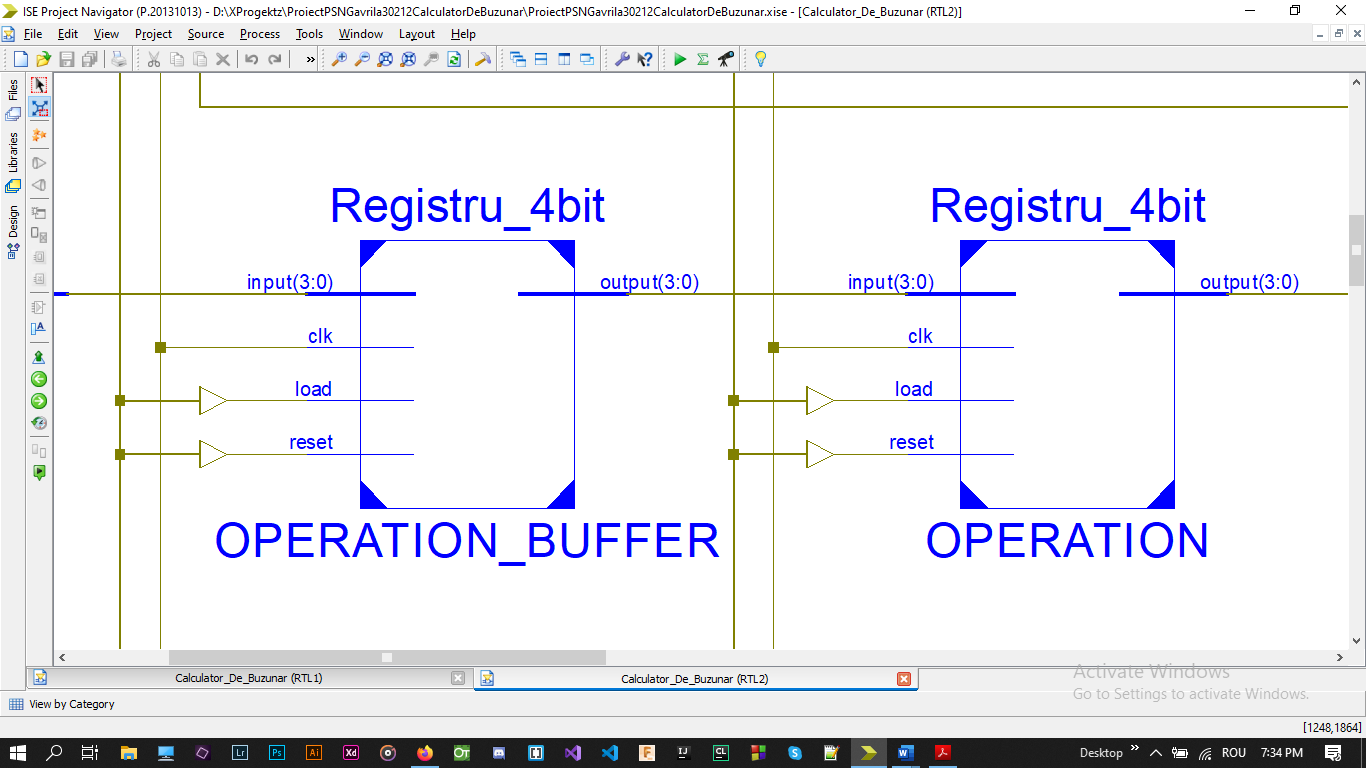
*(intrările din stânga, de sus în jos: scădere, adunare, inputA, inputB)*

*(ieșirile din dreapta, de sus în jos: comparator, outputA, outputB)*

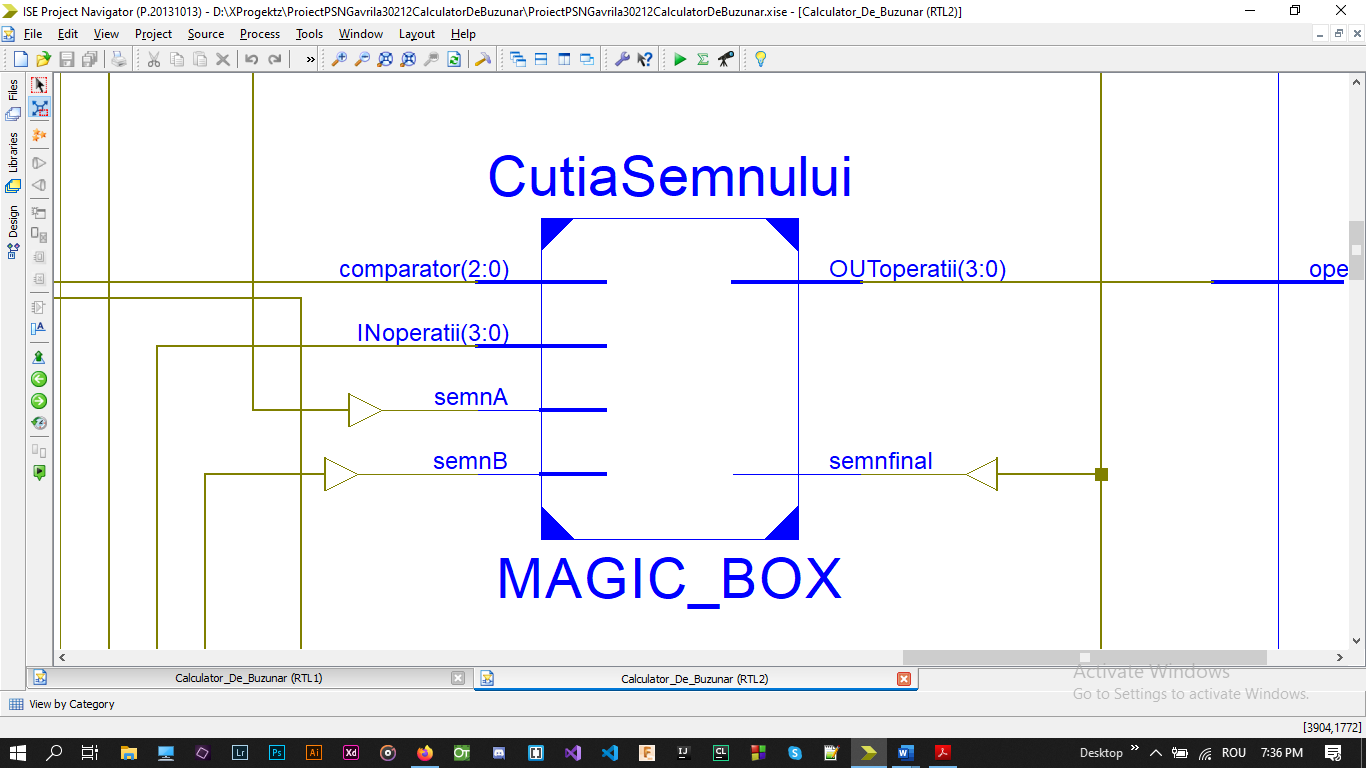
* *JK\_FlipFlop („ERROR\_CELL”)*

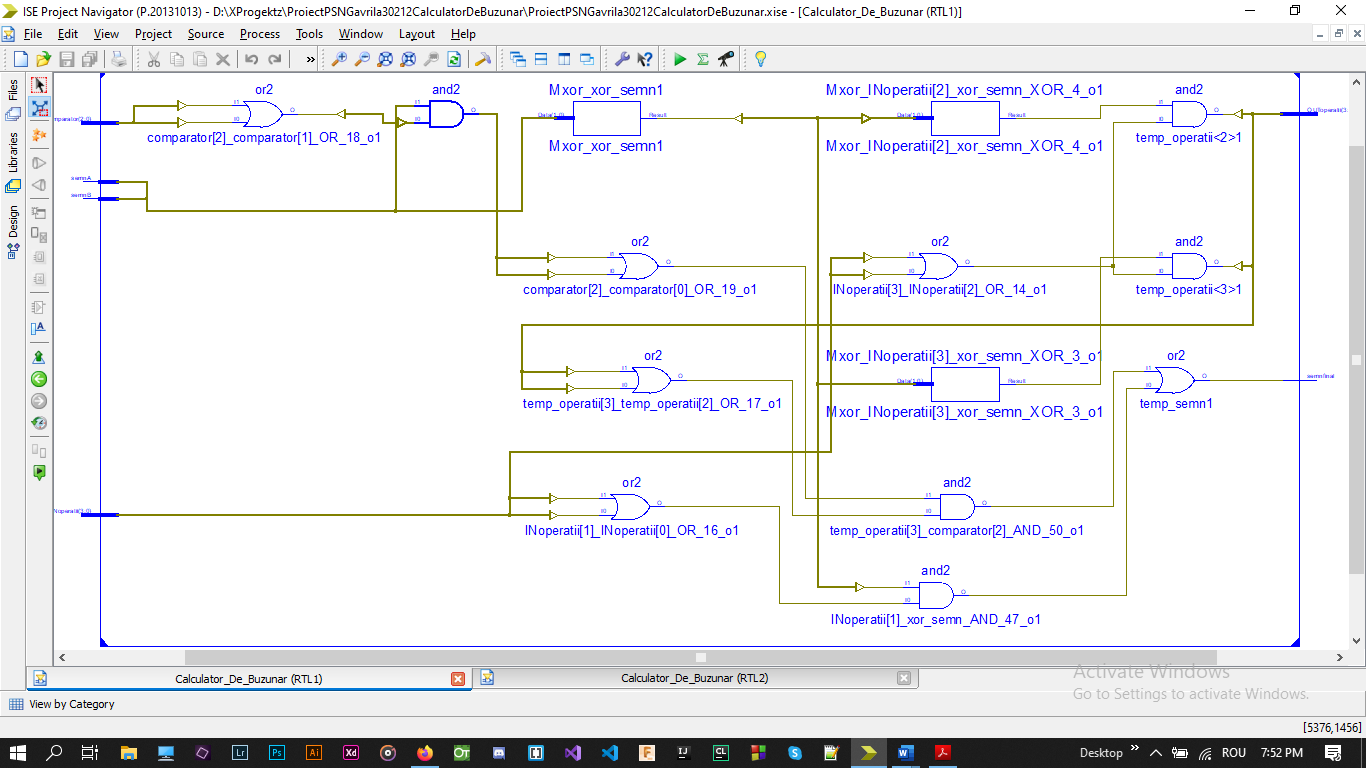


* *Registre 4-Bit „Operation\_buffer” și „Operation” – Registrii care retin operațiile (de la utilizator)*



* *CLC „CutiaSemnului” – Decide semnul rezultatului, și află operația efectivă*

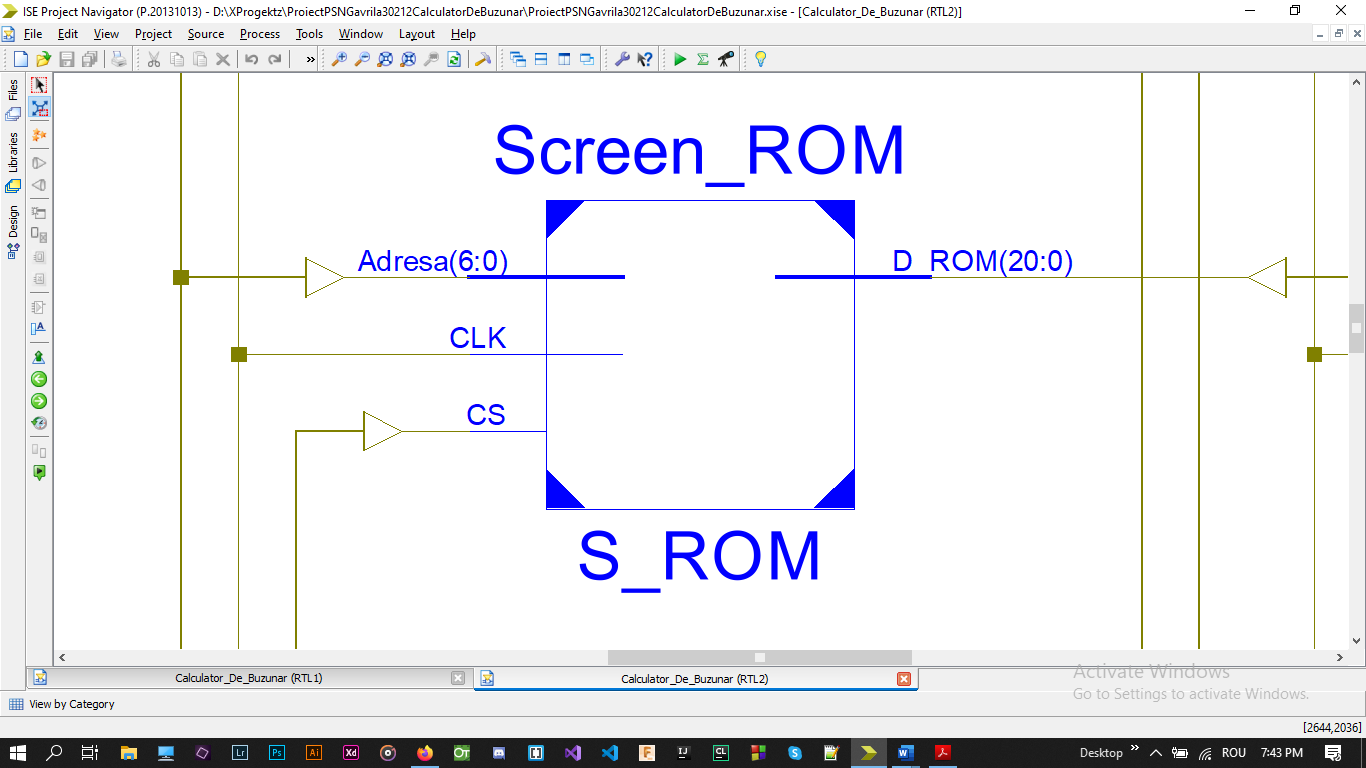




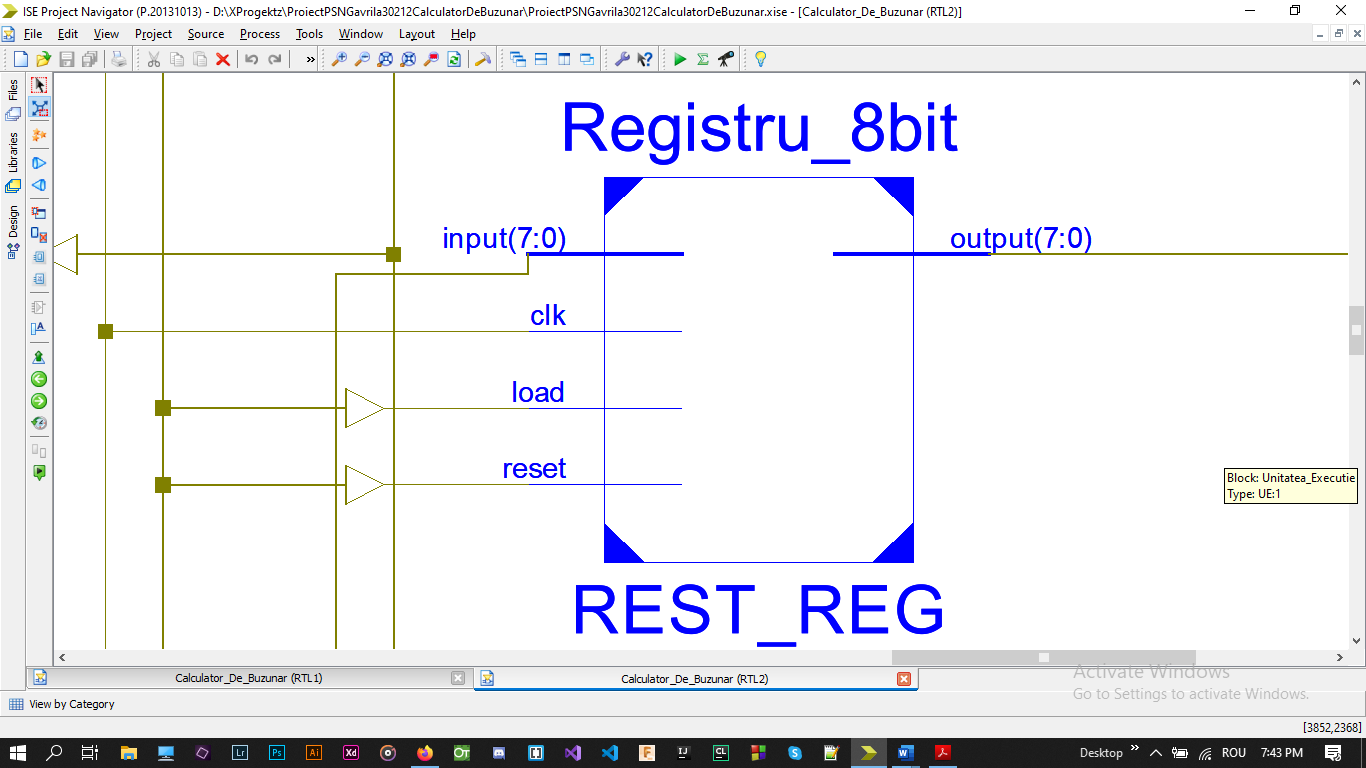
*(intrările din stânga, de sus în jos: comparator(2:0), semnA, semnB, INoperații)*

*(ieșirile din dreapta, de sus în jos: OUToperatii, semnfinal)*

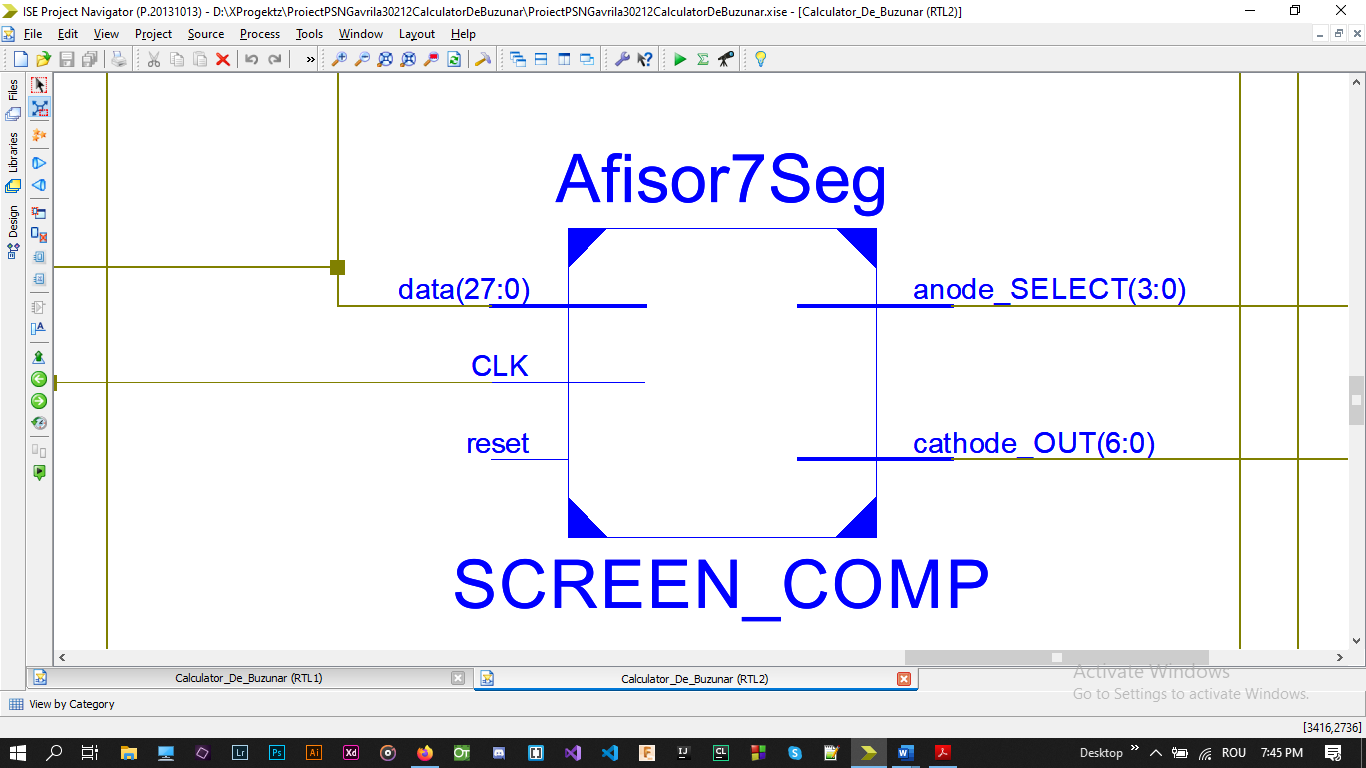
* *Sreen ROM – pentru conversia din binar în 7 Segmente (doar modulul este convertit), bitul de semn controlează direct un led din afișor*

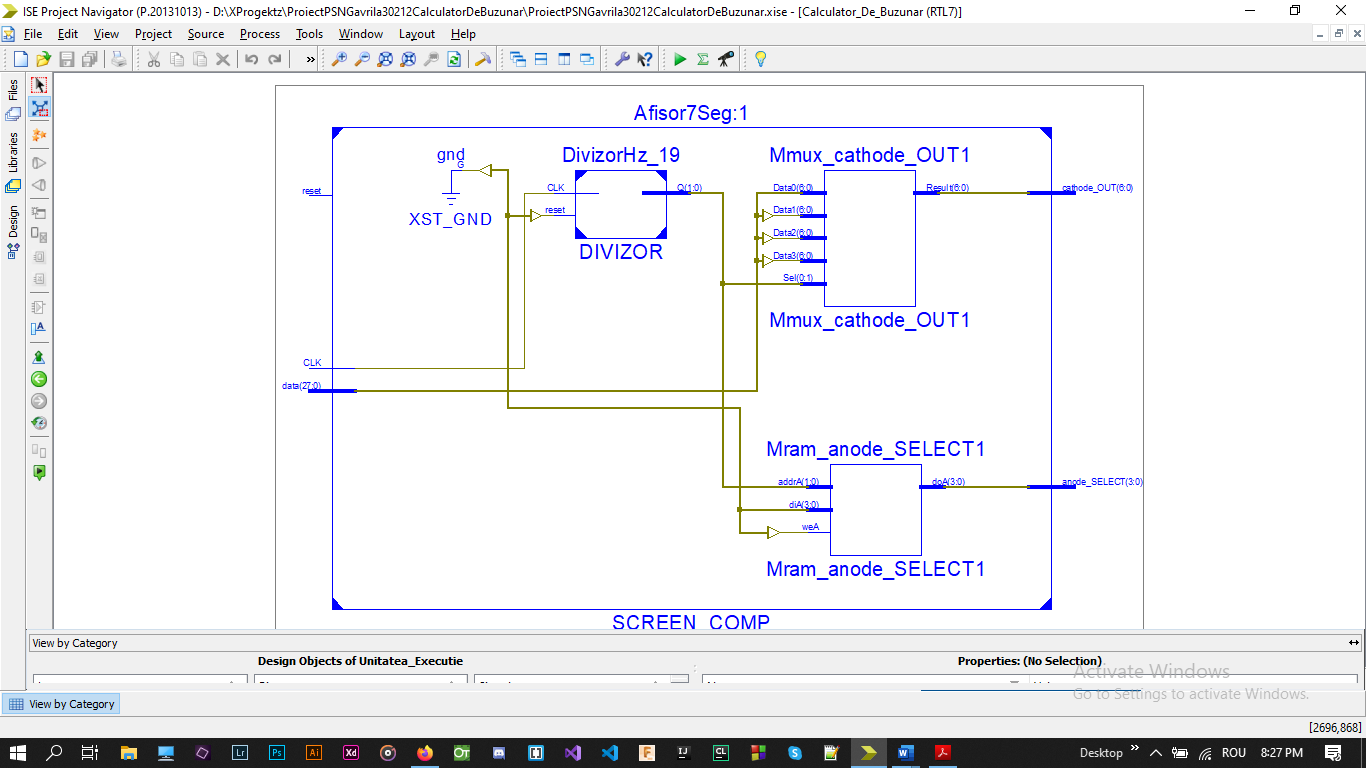


* *Registru 8-Biți „REST\_REG” – pentru reținerea restului*

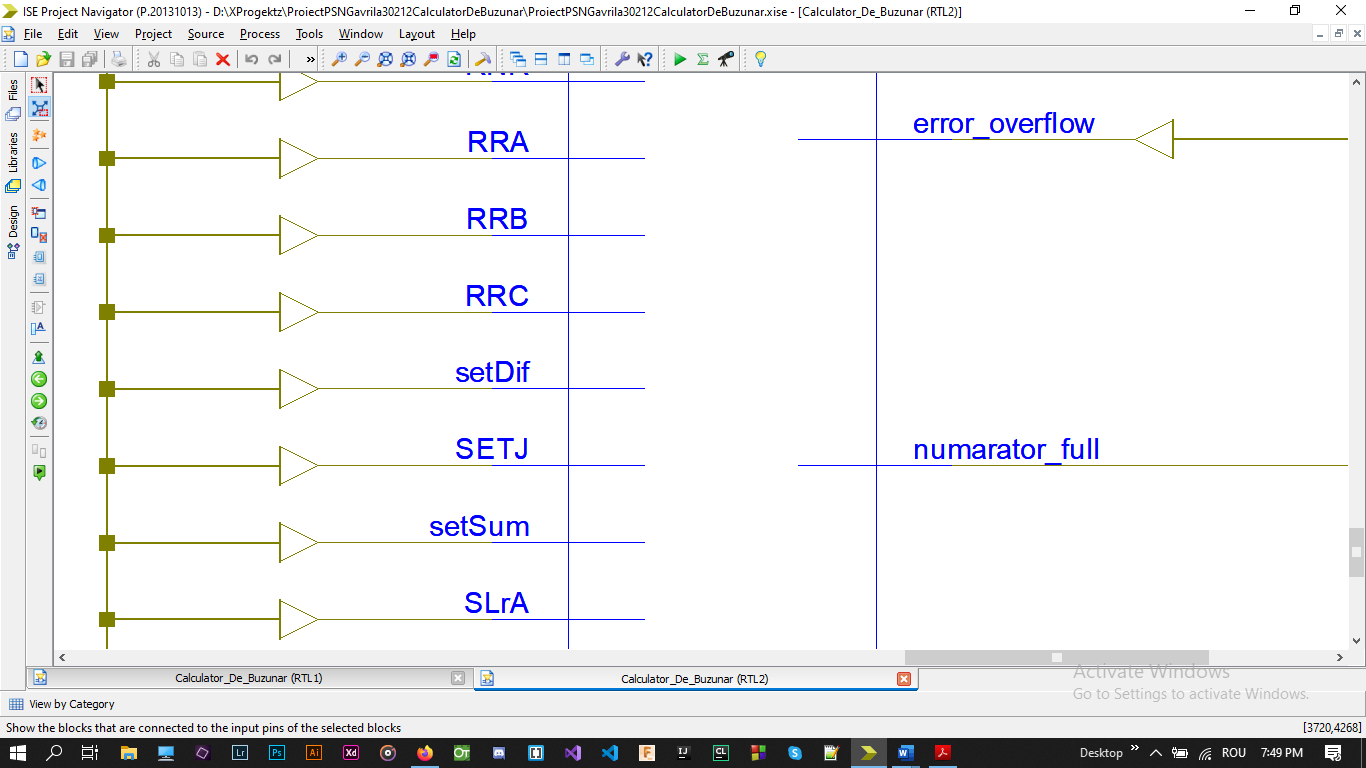
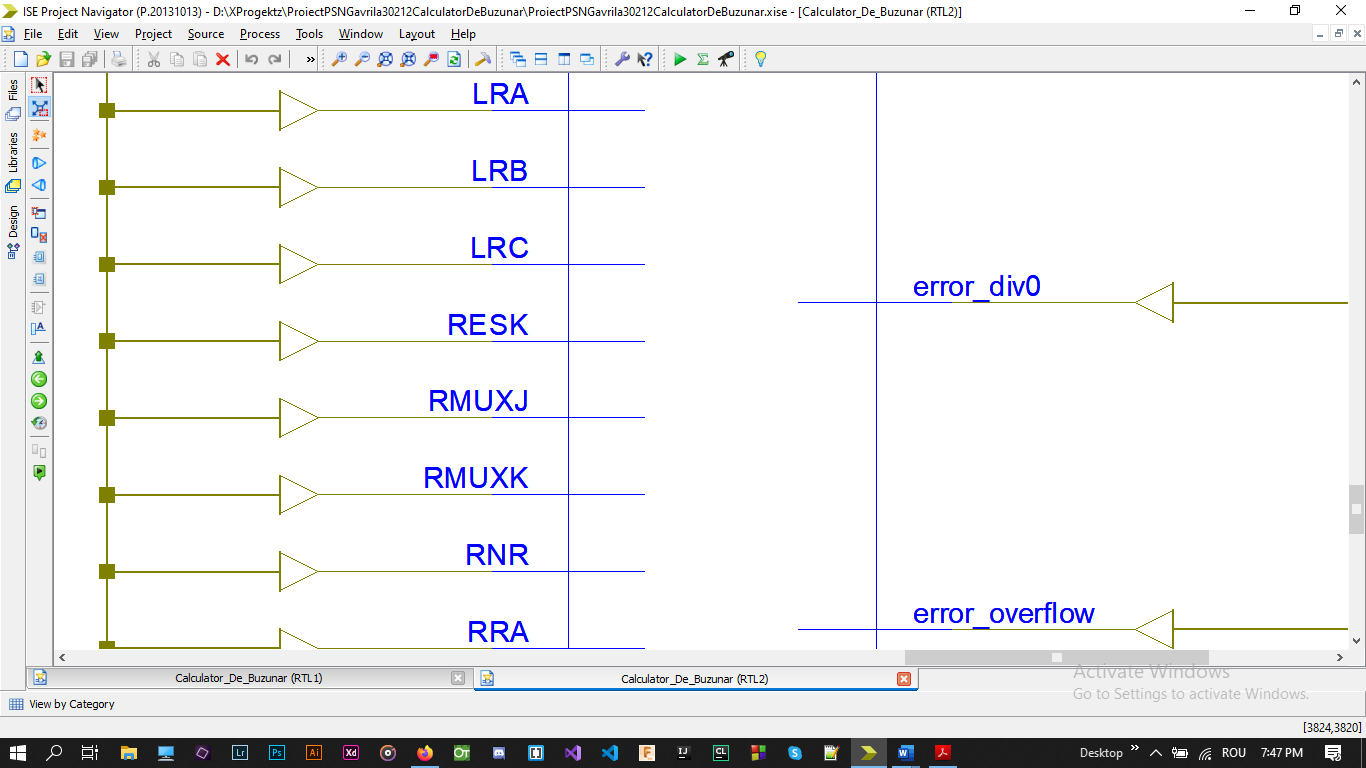
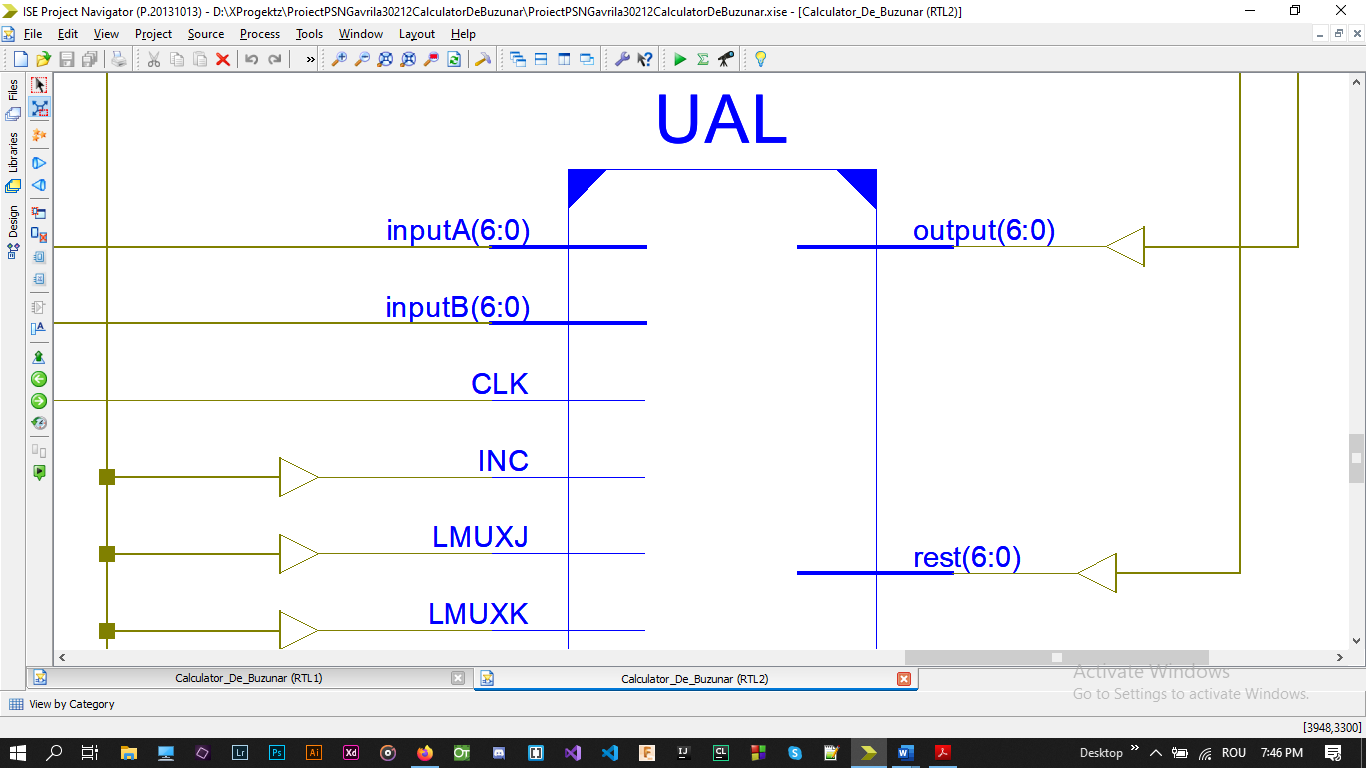


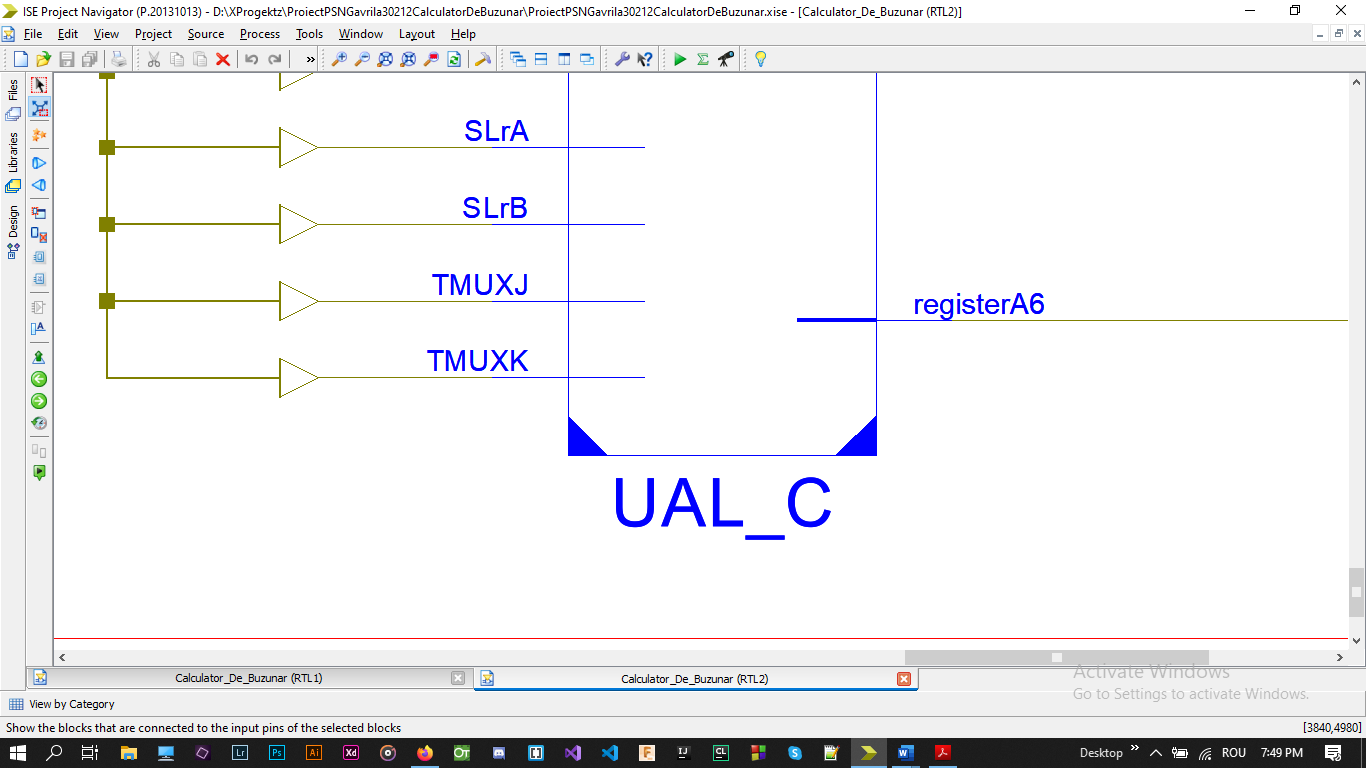
* *Afișor 7 segmente -*



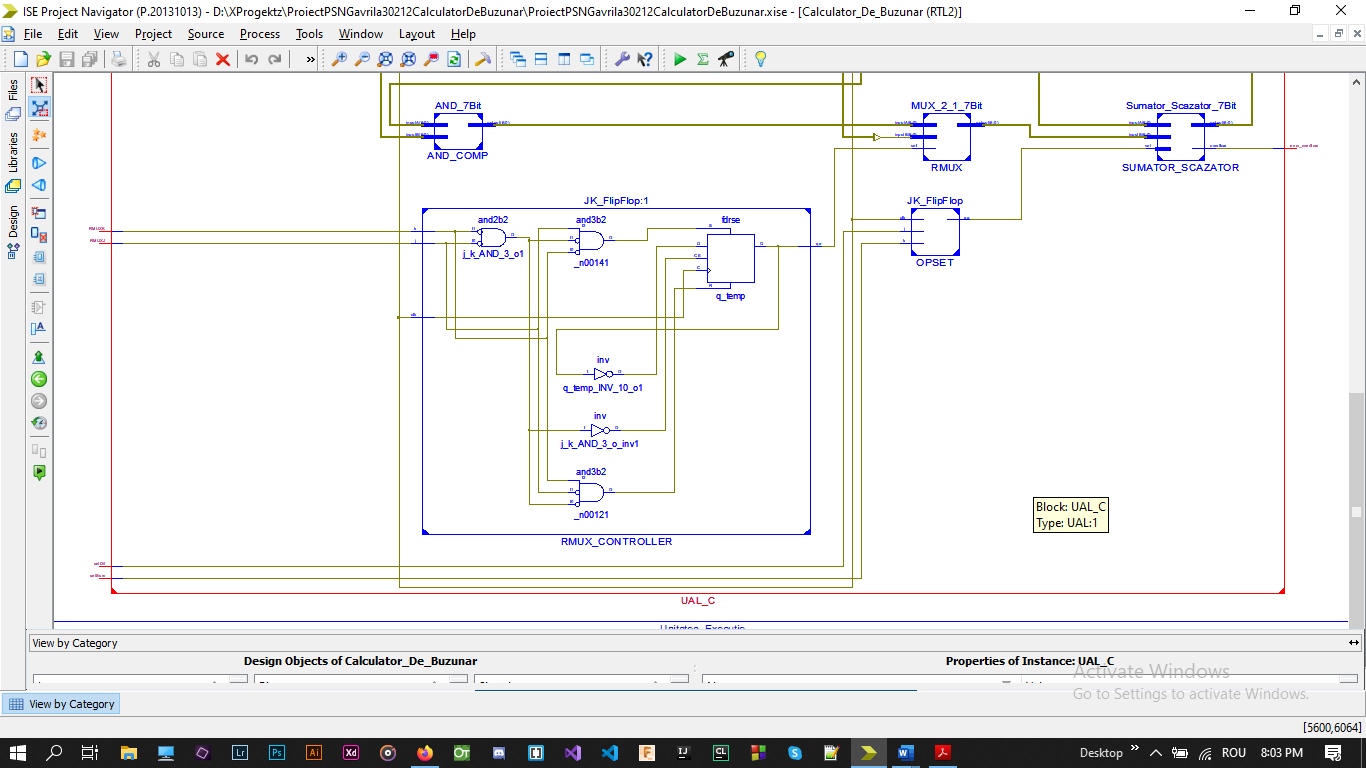
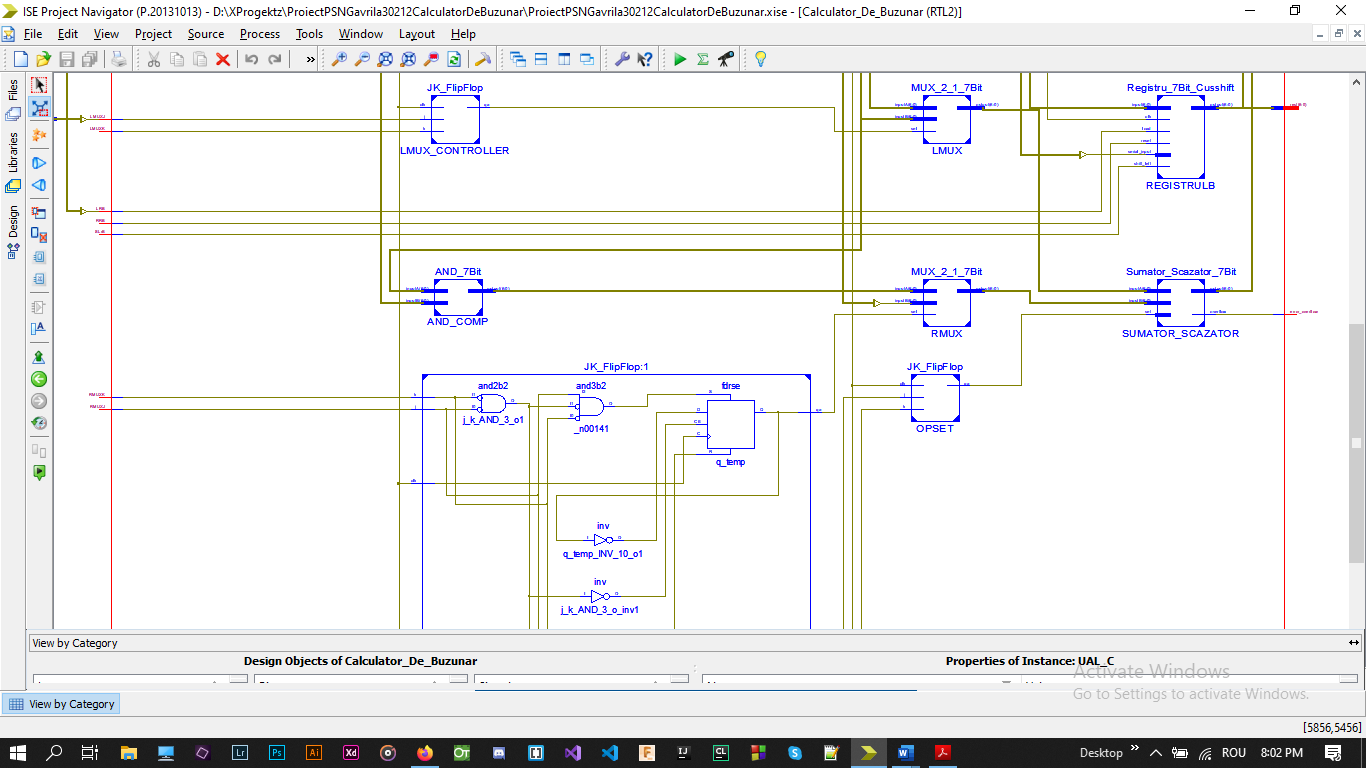
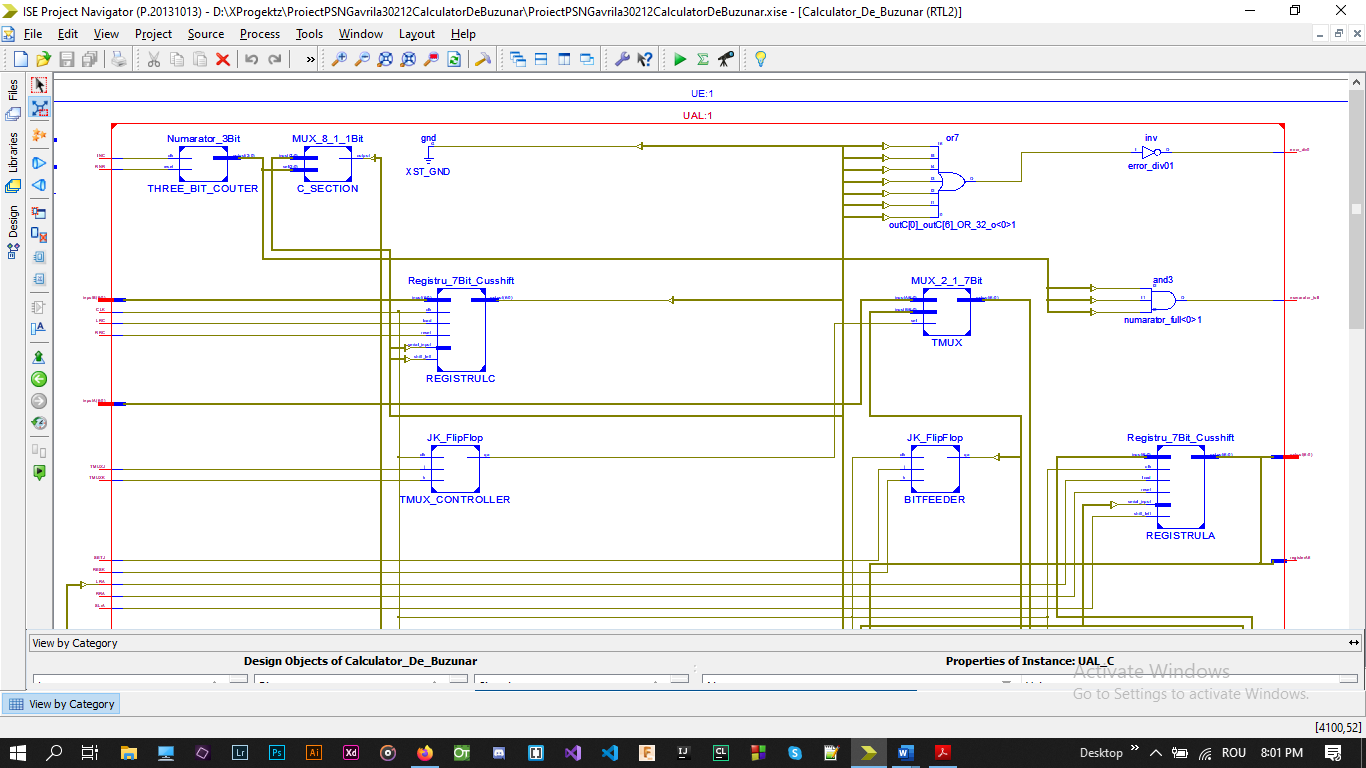


* *Unitatea Aritmetico-Logica:*

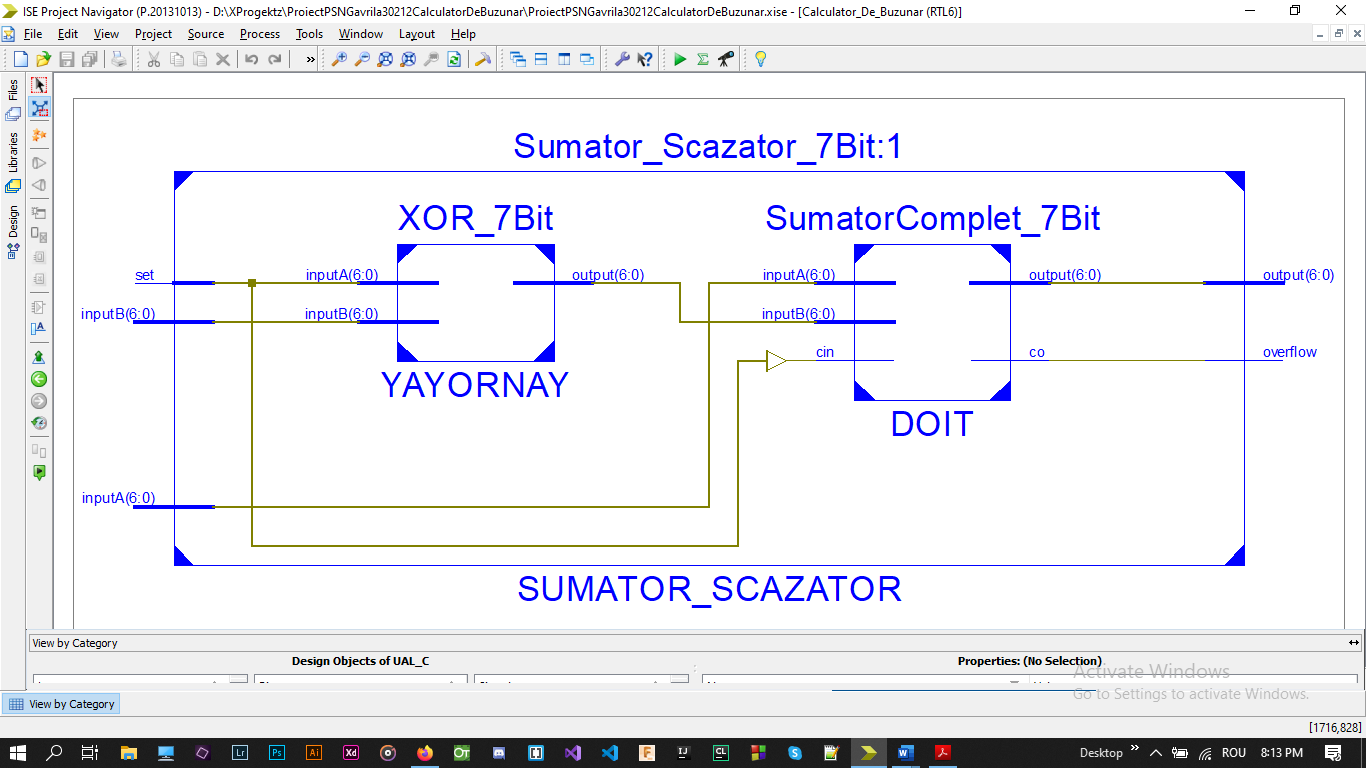


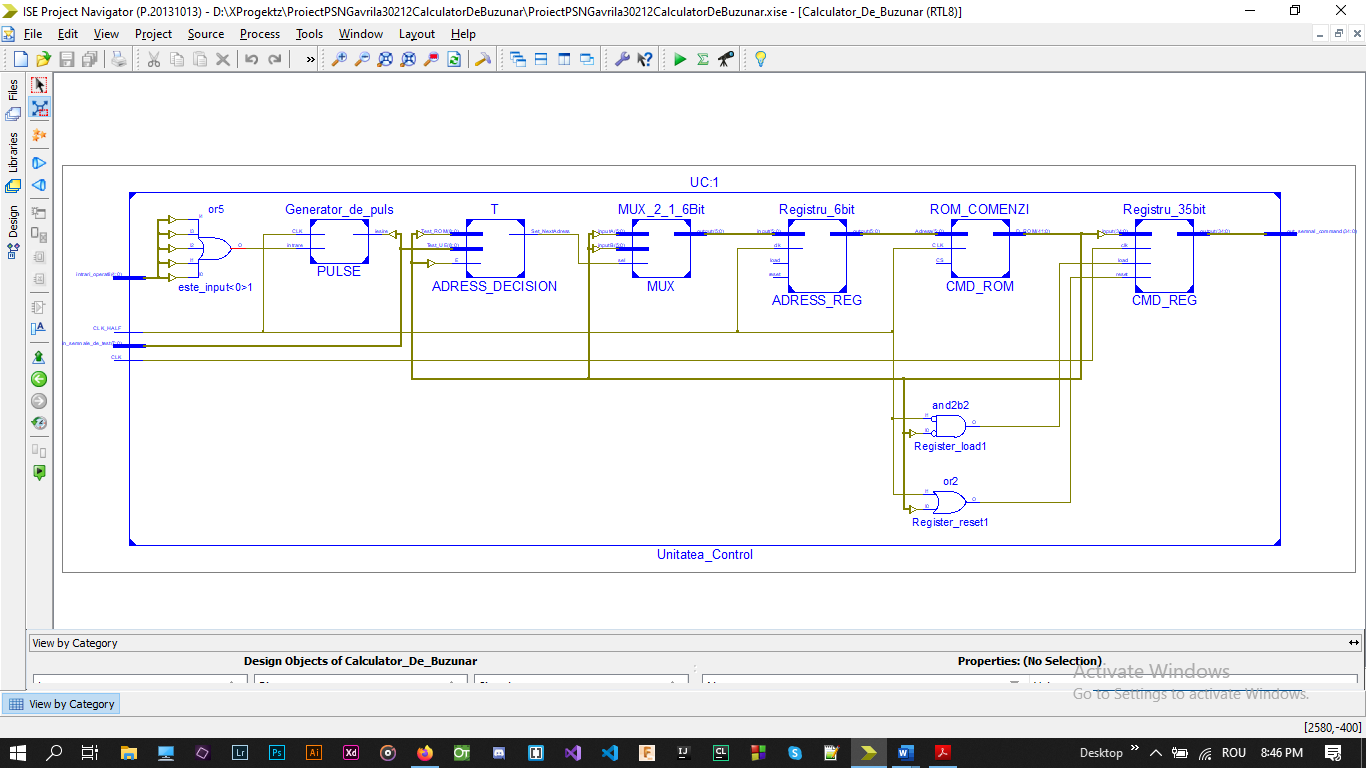


*UAL-Detaliat*

*Conține 3 registre A,B,C folosite pentru toate operațiile,* 

* *Sumator/Scăzător (din UAL)*



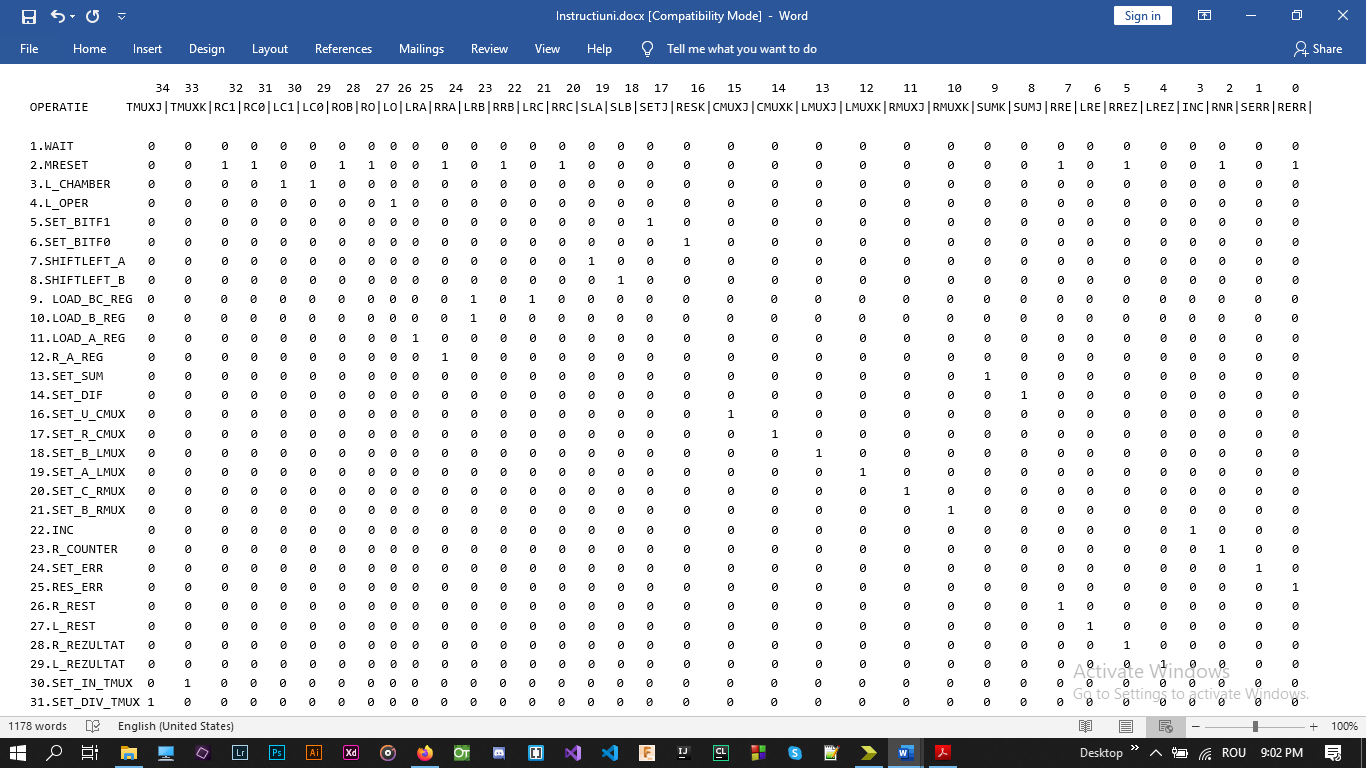
***Unitatea de execuție:***

Există două tipuri de instrucțiuni în ROM-ul UC, instrucțiuni de comandă, și instrucțiuni de test.

Pentru simplitate, am conceput ca instrucțiunile de comandă (cu câteva excepții) să reprezinte doar un semnal trimis către UE: (*dați zoom pentru a mări următoarea imagine*)

Tip operație/operații nextAdress

0 35 de biți 6 biți



Iar în instrucțiunile de test, dacă un anumit se semnal de test se potrivește, se sare la o anumită instrucțiune, dacă nu, la alta

Tip padding testflags adresaTrue adresaFals

1 20 de biți 9 biți 6 biți 6 biți

*Unele instrucțiuni realizează mai multe operații pentru scade numărul lor la 64 (utilizând* ***toate*** *combinațiile unui registru de 6 biți), aceste operații sunt separate prin |*

*Acest caracter mai este folosit pentru a indica când* ***nu*** *se trece la următoarea instrucțiune ci se* ***sare*** *la alta.*

**Algoritm UC:**

1. MRESET

STAND\_BY: (000010)

2. WAIT

3. If USER\_INPUT = 1 THEN

CONTINUE (4);

ELSE JUMP TO STAND\_BY (2);

4. SET\_U\_CMUX

5. WAIT

6. L\_OPER | L\_CHAMBER

7. RES\_ERR | R\_REST | SET\_IN\_TMUX | R\_COUNTER

8. LOAD\_BC\_REG

9. IF ADUNARE = 1 THEN

JUMP TO SUM (13);

ELSE CONTINUE (10)

10. IF SCADERE = 1 THEN

JUMP TO DIF (19);

ELSE CONTINUE (11)

11. IF INMULTIRE =1 THEN

JUMP TO MUL (23);

ELSE CONTINUE (12)

12. IF IMPARTIRE = 1 THEN

JUMP TO DIV (34);

ELSE JUMP TO STAND\_BY (2);

SUM: (001100)

13. SET\_SUM | SET\_B\_LMUX | SET\_C\_RMUX

14. WAIT

15. LOAD\_A\_REG

16. L\_REZULTAT

17. IF OVERFLOW = 1 THEN

CONTINUE

ELSE JUMP TO END\_OP ;

18. SET\_ERR | JUMP TO END\_OP

DIF: (010010)

19. SET\_DIF

20. SET\_B\_LMUX | SET\_C\_RMUX

21. LOAD\_A\_REG

22. L\_REZULTAT | JUMP TO END\_OP;

MUL: (010110)

23. R\_A\_REG

24. INC

25. SET\_A\_LMUX | SET\_B\_RMUX | SET\_BITF0

26. SET\_SUM

MUL\_LOOP: 011010

27. LOAD\_A\_REG

28. SHIFTLEFT\_B

29. INC

30. If COUNTER\_FULL = 1 THEN

CONTINUE

ELSE

JUMP TO MUL\_LOOP:

31. L\_REZULTAT

32. IF OVERFLOW = 1 THEN

CONTINUE

ELSE JUMP TO END\_OP;

33. SET\_ERR | JUMP TO END\_OP;

DIV:

34. IF DIVISION\_0 = 1 THEN

CONTINUE

ELSE

JUMP ON 36

35. SET\_ERR; | JUMP ON END\_OP

36. R\_A\_REG | INC | SET\_BITF0

37. SHIFTLEFT\_A

38. SHIFTLEFT\_B

39. SET\_A\_LMUX | SET\_C\_RMUX

40. SET\_DIF

41. LOAD\_A\_REG

42. INC

DIV\_LOOP: (101010)

43. If A6 = 1 THEN

JUMP SUM\_BRANCH;

ELSE JUMP DIF\_BRANCH;

SUM\_BRANCH: (101011)

44. SET\_BITF0

45. SHIFTLEFT\_A

46. SHIFTLEFT\_B

47. SET\_SUM | JUMP ON LOOP\_CHECK

DIF\_BRANCH (101111)

48. SET\_BITF1 | SET\_DIV\_TMUX

49. LOAD\_B\_REG

50. SET\_BITF0

51. SHIFTLEFT\_A

52. SHIFTLEFT\_B

53. SET\_DIF | JUMP ON LOOP\_CHECK

LOOP\_CHECK: (110101)

54. LOAD\_A\_REG

55. INC

56. If COUNTER\_FULL = 1 THEN

CONTINUE

ELSE JUMP TO DIV\_LOOP;

57. If A6 = 1 THEN

CONTINUE;

ELSE JUMP FINAL;

58. SET\_SUM | L\_REST

59. LOAD\_A\_REG | JUMP TO END\_OP

FINAL: 111011

60. SET\_BITF1 | SET\_DIV\_TMUX

61. LOAD\_B\_REG

62. L\_REST | JUMP TO END\_OP

END\_OP:

63. SET\_R\_CMUX | L\_REZ

64. L\_CHAMBER | JUMP TO STAND\_BY;

**ROM-ul UC, realizat pe baza algoritmului:**

ADRESA TIP COMANDA/TEST/ADRESA TRUE ADRESANEXT

000000. 0 00110011001010100000000000010100101 000001

000001. 0 00000000000000000000000000000000000 000010

000010. 1 00000000000000000000000000010000011 000010

000011. 0 00000000000000000001000000000000000 000100

000100. 0 00000100000000000000000000000000000 000101

000101. 0 00001000100000000000000000000000000 000110

000110. 0 01000000000000000000000000010000101 000111

000111. 0 00000000000101000000000000000000000 001000

001000. 1 00000000000000000000100000000001100 001001

001001. 1 00000000000000000000010000000010010 001010

001010. 1 00000000000000000000001000000010110 001011

001011. 1 00000000000000000000000100000100001 000001

001100. 0 00000000000000000000010101000000000 001101

001101. 0 00000000000000000000000000000000000 001110

001110. 0 00000000010000000000000000000000000 001111

001111. 0 00000000000000000000000000000010000 010000

010000. 1 00000000000000000000000010000010001 111110

010001. 0 00000000000000000000000000000000010 111110

010010. 0 00000000000000000000000000100000000 010011

010011. 0 00000000000000000000010100000000000 010100

010100. 0 00000000010000000000000000000000000 010101

010101. 0 00000000000000000000000000000010000 111110

010110. 0 00000000001000000000000000000000000 010111

010111. 0 00000000000000000000000000000001000 011000

011000. 0 00000000000000000010001010000000000 011001

011001. 0 00000000000000000000000001000000000 011010

011010. 0 00000000010000000000000000000000000 011011

011011. 0 00000000000000001000000000000000000 011100

011100. 0 00000000000000000000000000000001000 011101

011101. 1 00000000000000000000000000100011110 011010

011110. 0 00000000000000000000000000000010000 011111

011111. 1 00000000000000000000000010000100000 111110

100000. 0 00000000000000000000000000000000010 111110

100001. 1 00000000000000000000000001000100010 100011

100010. 0 00000000000000000000000000000000010 111110

100011. 0 00000000001000000010000000000001000 100100

100100. 0 00000000000000010000000000000000000 100101

100101. 0 00000000000000001000000000000000000 100110

100110. 0 00000000000000000000001100000000000 100111

100111. 0 00000000000000000000000000100000000 101000

101000. 0 00000000010000000000000000000000000 101001

101001. 0 00000000000000000000000000000001000 101010

101010. 1 00000000000000000000000000001101011 101111

101011. 0 00000000000000000010000000000000000 101100

101100. 0 00000000000000010000000000000000000 101101

101101. 0 00000000000000001000000000000000000 101110

101110. 0 00000000000000000000000001000000000 110101

101111. 0 10000000000000000100000000000000000 110000

110000. 0 00000000000100000000000000000000000 110001

110001. 0 00000000000000000010000000000000000 110010

110010. 0 00000000000000010000000000000000000 110011

110011. 0 00000000000000001000000000000000000 110100

110100. 0 00000000000000000000000000100000000 110101

110101. 0 00000000010000000000000000000000000 110110

110110. 0 00000000000000000000000000000001000 110111

110111. 1 00000000000000000000000000100111000 101010

111000. 1 00000000000000000000000000001111001 111011

111001. 0 00000000000000000000000001001000000 111010

111010. 0 00000000010000000000000000000000000 111110

111011. 0 10000000000000000100000000000000000 111100

111100.` 0 00000000000100000000000000000000000 111101

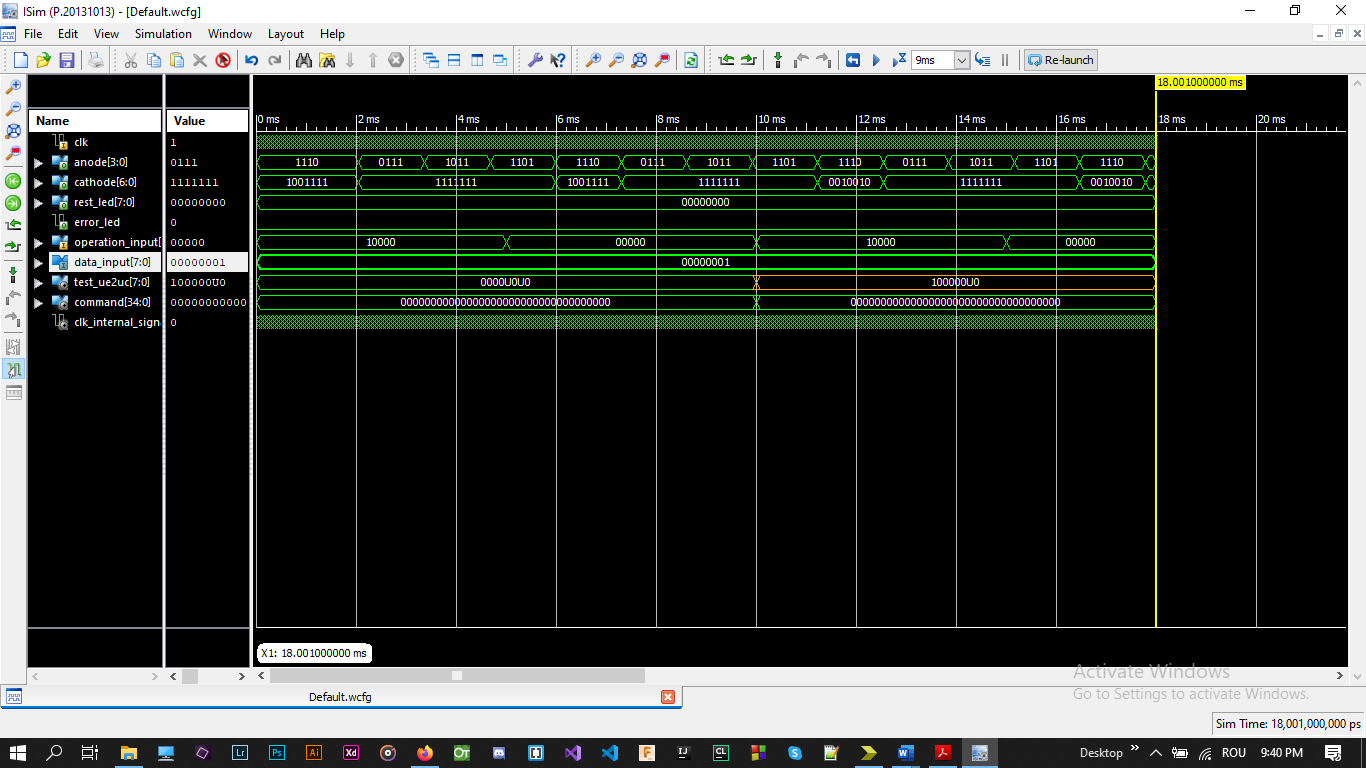
111101. 0 00000000000000000000000000001000000 111110

111110. 0 00000000000000000000100000000000000 111111

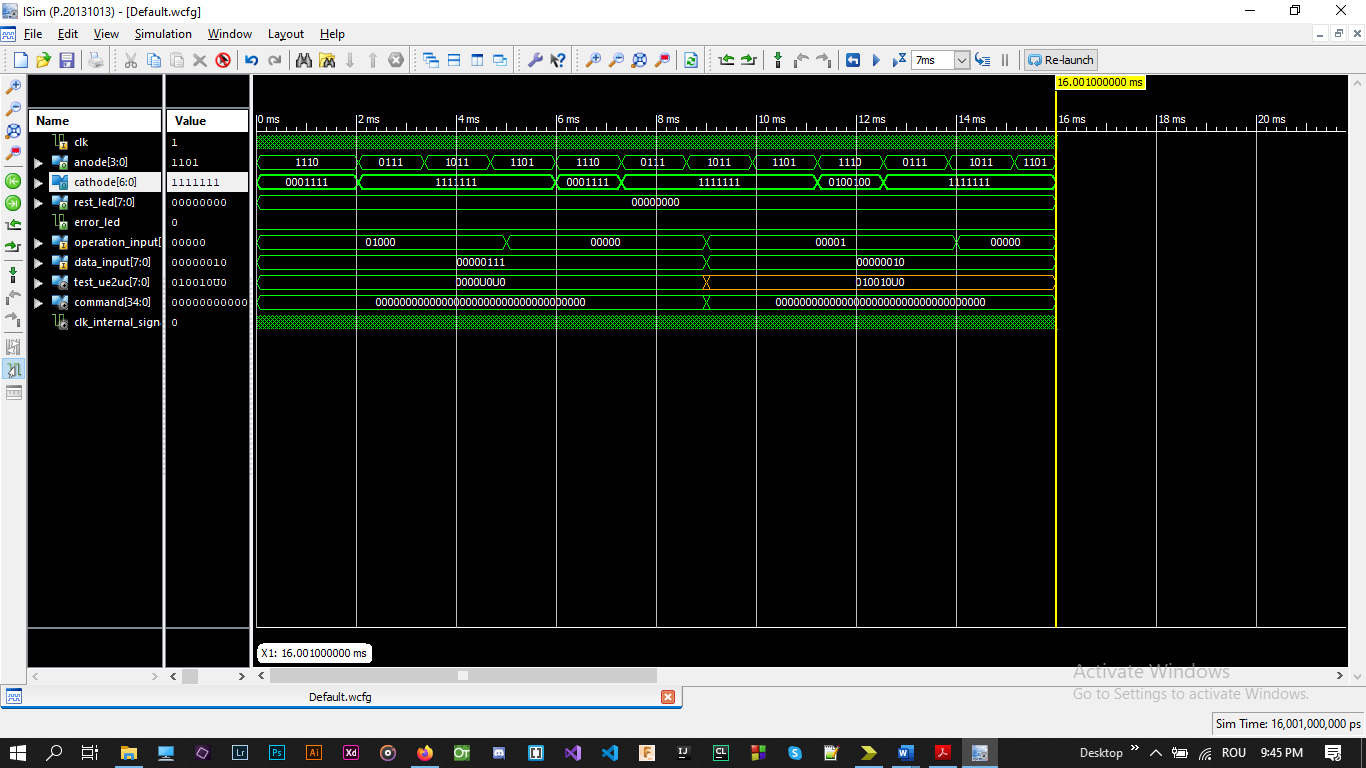
111111. 0 00001100000000000000000000000000000 000001

**Simulări:**

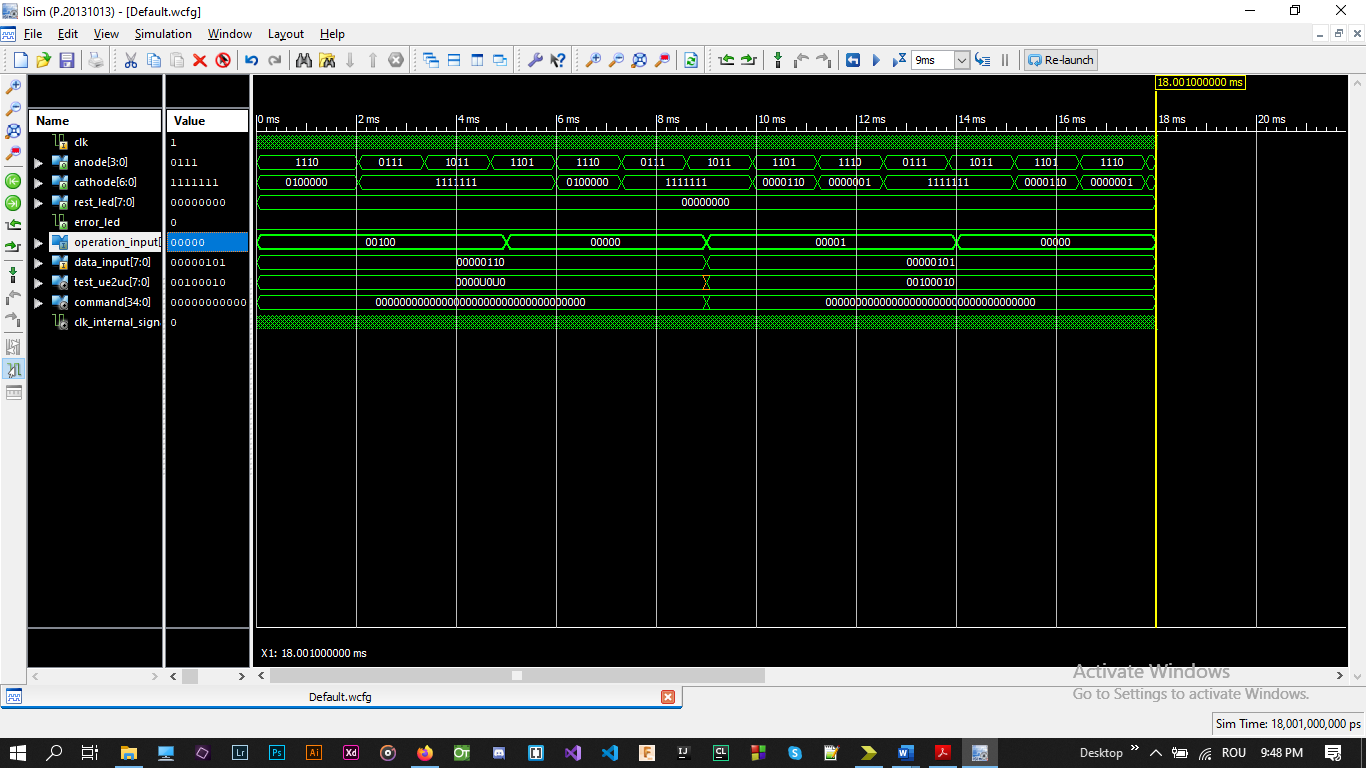
Întreg calculatorul (adunare):



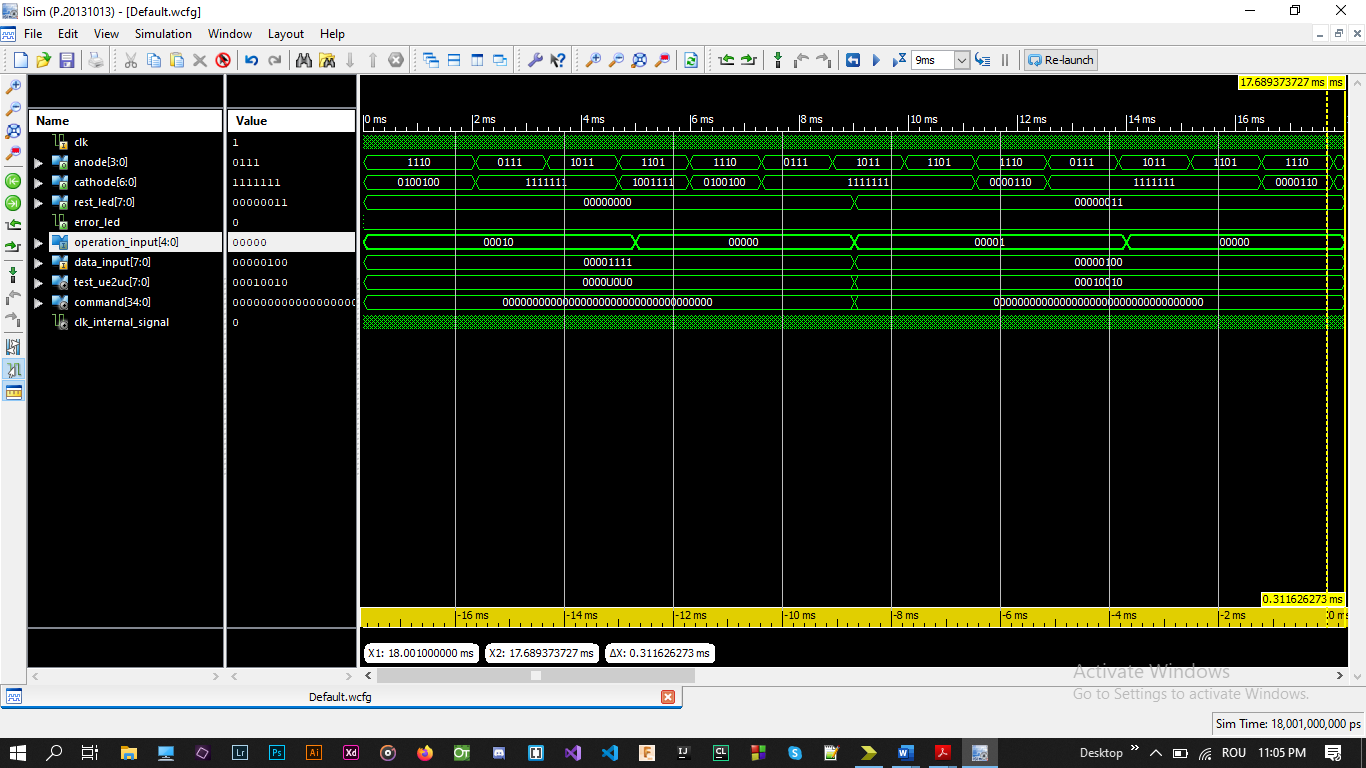
Întreg calculatorul (scădere):



Întreg calculatorul (înmulțire):



Întreg calculatorul (împărțire)



**Fișierul de costrângeri UCF:**

NET "CLK" LOC = E3 | IOSTANDARD = LVCMOS33; #Bank = 35, Pin name = #IO\_L12P\_T1\_MRCC\_35, Sch name = clk100mhz

NET "CLK" TNM\_NET = sys\_clk\_pin;

TIMESPEC TS\_sys\_clk\_pin = PERIOD sys\_clk\_pin 100 MHz HIGH 50%;

NET "data\_input(0)" LOC=J15 | IOSTANDARD=LVCMOS33; #IO\_L24N\_T3\_RS0\_15

NET "data\_input(1)" LOC=L16 | IOSTANDARD=LVCMOS33; #IO\_L3N\_T0\_DQS\_EMCCLK\_14

NET "data\_input(2)" LOC=M13 | IOSTANDARD=LVCMOS33; #IO\_L6N\_T0\_D08\_VREF\_14

NET "data\_input(3)" LOC=R15 | IOSTANDARD=LVCMOS33; #IO\_L13N\_T2\_MRCC\_14

NET "data\_input(4)" LOC=R17 | IOSTANDARD=LVCMOS33; #IO\_L12N\_T1\_MRCC\_14

NET "data\_input(5)" LOC=T18 | IOSTANDARD=LVCMOS33; #IO\_L7N\_T1\_D10\_14

NET "data\_input(6)" LOC=U18 | IOSTANDARD=LVCMOS33; #IO\_L17N\_T2\_A13\_D29\_14

NET "data\_input(7)" LOC=R13 | IOSTANDARD=LVCMOS33; #IO\_L5N\_T0\_D07\_14

NET "operation\_input(0)" LOC=T13 | IOSTANDARD=LVCMOS33; #IO\_L23P\_T3\_A03\_D19\_14

NET "operation\_input(1)" LOC=H6 | IOSTANDARD=LVCMOS33; #IO\_L24P\_T3\_35

NET "operation\_input(2)" LOC=U12 | IOSTANDARD=LVCMOS33; #IO\_L20P\_T3\_A08\_D24\_14

NET "operation\_input(3)" LOC=U11 | IOSTANDARD=LVCMOS33; #IO\_L19N\_T3\_A09\_D25\_VREF\_14

NET "operation\_input(4)" LOC=V10 | IOSTANDARD=LVCMOS33; #IO\_L21P\_T3\_DQS\_14

NET "rest\_LED(0)" LOC=H17 | IOSTANDARD=LVCMOS33; #IO\_L18P\_T2\_A24\_15

NET "rest\_LED(1)" LOC=K15 | IOSTANDARD=LVCMOS33; #IO\_L24P\_T3\_RS1\_15

NET "rest\_LED(2)" LOC=J13 | IOSTANDARD=LVCMOS33; #IO\_L17N\_T2\_A25\_15

NET "rest\_LED(3)" LOC=N14 | IOSTANDARD=LVCMOS33; #IO\_L8P\_T1\_D11\_14

NET "rest\_LED(4)" LOC=R18 | IOSTANDARD=LVCMOS33; #IO\_L7P\_T1\_D09\_14

NET "rest\_LED(5)" LOC=V17 | IOSTANDARD=LVCMOS33; #IO\_L18N\_T2\_A11\_D27\_14

NET "rest\_LED(6)" LOC=U17 | IOSTANDARD=LVCMOS33; #IO\_L17P\_T2\_A14\_D30\_14

NET "rest\_LED(7)" LOC=U16 | IOSTANDARD=LVCMOS33; #IO\_L18P\_T2\_A12\_D28\_14

NET "error\_LED" LOC=V11 | IOSTANDARD=LVCMOS33; #IO\_L21N\_T3\_DQS\_A06\_D22\_14

NET "Cathode(6)" LOC=T10 | IOSTANDARD=LVCMOS33; #IO\_L24N\_T3\_A00\_D16\_14

NET "Cathode(5)" LOC=R10 | IOSTANDARD=LVCMOS33; #IO\_25\_14

NET "Cathode(4)" LOC=K16 | IOSTANDARD=LVCMOS33; #IO\_25\_15

NET "Cathode(3)" LOC=K13 | IOSTANDARD=LVCMOS33; #IO\_L17P\_T2\_A26\_15

NET "Cathode(2)" LOC=P15 | IOSTANDARD=LVCMOS33; #IO\_L13P\_T2\_MRCC\_14

NET "Cathode(1)" LOC=T11 | IOSTANDARD=LVCMOS33; #IO\_L19P\_T3\_A10\_D26\_14

NET "Cathode(0)" LOC=L18 | IOSTANDARD=LVCMOS33; #IO\_L4P\_T0\_D04\_14

NET "Anode(0)" LOC=J17 | IOSTANDARD=LVCMOS33; #IO\_L23P\_T3\_FOE\_B\_15

NET "Anode(1)" LOC=J18 | IOSTANDARD=LVCMOS33; #IO\_L23N\_T3\_FWE\_B\_15

NET "Anode(2)" LOC=T9 | IOSTANDARD=LVCMOS33; #IO\_L24P\_T3\_A01\_D17\_14

NET "Anode(3)" LOC=J14 | IOSTANDARD=LVCMOS33; #IO\_L19P\_T3\_A22\_15

**4.Componentele utilizate si semnalele lor**

* Registru de 35 biți: utilizat în UC pentru trimiterea comenzilor către UE
* Registru de 8 biți: pentru reținerea numerelor înainte de operație, a rezultatului și a restului de la împărțire.
* Registru de 7 biți: pentru reținerea numerelor în UAL
* Registru de 6 biți: utilizat în UC pentru următoarea adresă de memorie
* Registru de 4 biți: pentru reținerea operației care se efectuează
  + *Input – intrarea pentru încărcarea paralelă*
  + *Clk – intrare pentru semnalul de tact*
  + *Reset – intrare pentru resetarea (****sincronă****) a registrului*
  + *Load – semnal pentru încărcarea paralelă*
  + *Output – ieșirile registrului*

*(Denumirile și semnificațiile sunt valabile pentru toate registrele)*

* MUX 2:1 pe 8 și 7 Biți
  + *inputA – intrarea (****magistrală****) care e selectată atunci când semnalul de selecție e 0*
  + *inputB – intrarea (****magistrală)*** *care e selectată atunci când semnalul de selecție e 1*
  + *output – ieșirea care arată semnalele selectate*
  + *sel – intrarea de selecție*
* MUX 8:1 pe 1 Bit
  + *Input – Semnalele de intrare (****magistrală****)*
  + *Output – ieșire semnalul selectat*
  + *Sel – semnalele de selecție(***magistrală*)***
* Numărător pe 3 Biți: pentru numărarea pașilor în algoritmii de înmulțire și împărțire
* Numărător pe 2 Biți: folosit ca componentă în divizorul de frecvență
  + *Reset – intrare pentru resetarea* ***sincronă*** *(ceea ce face reset-ul inutil) a numărătorului*
  + *Clk – intrare semnal de tact*
  + *Output – iesire stare numărător*

*(Denumirile și semnificațiile sunt valabile pentru toate numărătoarele)*

* Numărător pe 2 și 3 Biți (cu Carry): folosit ca componentă în divizorul de frecventă
  + *Reset – intrare pentru resetarea* ***sincronă*** *(ceea ce face reset-ul inutil) a numărătorului*
  + *Clk – intrare semnal de tact*
  + *CE – ieșire de carry pentru următorul numărător*

*(Denumirile și semnificațiile sunt valabile pentru toate numărătoarele)*

* Bistabil JK: pentru controlarea MUX-urilor, LED-ului de eroare și pentru serial input al registrului B din UAL
  + *J – intrare pentru setarea pe valoarea logică 1 a bistabilului*
  + *K- intrare pentru setarea pe valoarea logică 0 a bistabilului*
  + *CLK – intrare semnal de tact*
  + *Qa – ieșire stare bistabil*
* Comparator pe 7 Biți: pentru a decide (după caz) dacă se face schimbarea primului operand cu al doilea, și viceversa
  + *inputA – intrarea A*
  + *inputB - intrarea B*
  + *output – ieșire (****magistrală pe 3 biti****) care arată rezultatul comparării (100 pentru A>B, 010 pentru A=B, 001 pentru A<B)*
* ROM 128x21: pentru convertirea numerelor din binar în 7 segmente
* ROM 64x42: pentru algoritmul de funcționare al calculatorului
  + *CLK – intrare semnal de tact*
  + *Adresa – intrare adresă ROM*
  + *CS – semnal de enable pentru ROM*
  + *D\_ROM – ieșire cu valoarea reținută pe adresa respectivă (dacă e cazul)*

*(Denumirile și semnificațiile sunt valabile pentru ambele ROM-uri)*

* Sumator/Scăzător pe 7 Biți
  + *inputA – intrare pentru primul operand*
  + *inputB – intrare pentru al doilea operand*
  + *set – semnal pentru alegerea din adunare și scădere*
  + *output – ieșirea rezultatului*
  + *overflow – ultimul semnal de carry de la sumator care implică (după cum descrie numele) o depășire a capacității de 7 Biți*
* Bistabile OR, AND, NOT, XOR

**5. Justificarea Soluției Alese**

Unitatea de control microprogramată, cu ajutorul documentației disponibile, este ușor de reprogramat, astfel, lăsând loc pentru multă dezvoltare cu schimbări minime asupra UE. Această metodă de proiectare are asemănări cu programarea în limbaj de asamblare. Calculatorul este simplu de folosit, și cu un pic de dezvoltare ulterioară va deveni și foarte practic. Am încercat să fac un compromis între ușurința de proiectare, ușurința de folosire, și eficiență.

**6. Utilizare si rezultate**

Utilizarea calculatorului, deși simplă, poate fi derutantă din cauza felului in care sunt afișate rezultate/datele.

Principiul de baza constă în introducerea de perechi număr-operație.

De exemplu, presupunem ca doriți sa calculați (17-8)\*2 :

* Mai întâi trebuie calculat rezultatul din paranteză așadar de introduce 17 împreună cu operația „-”.
* (după introducerea operației ar trebui sa apară 17 pe afișor)
* Însă rezultatul operației minus va trebui înmulțit cu 2 așadar se introduce 8 împreună cu operația „\*”.
* (după introducerea operației ar trebui sa apară rezultatul, 9, pe afișor)
* Și deoarece nu dorim sa lucram in continuare cu rezultatul operației introducem 2 si operația „=”. (care nu pune nimic in buffer-ul de operații)
* (după introducerea operației ar trebui sa apară rezultatul final, 18, pe afișor).

Pe afișor va apărea rezultatul ultimei operații, dacă nu există operație de făcut se va afișa ultimul număr introdus.

LED-ul de eroare se aprinde dacă a existat un overflow în calcul sau dacă s-a încercat împărțirea cu 0.

LED-urile de rest arată (dacă s-a efectuat o împărțire) restul sub forma de mărime și semn.

**7.Posibilitati de dezvoltare ulterioara**

* Din cauză ca nu am avut acces personal la o placă de dezvoltare FPGA, am omis în proiectare includerea unui debouncer, și cred că includerea unuia este destul de importantă pentru utilizarea calculatorului pe plăci.
* Regiștri de 8 biți impun o limită destul de severă cu privire la mărimea numerelor cu care putem calcula. Consider că mărirea "range-ului" (mărind registrele de lucru) efectiv de lucru ar face calculatorul mult mai practic.
* Împărțirea cu rest este destul de nefolositoare pentru unele use case-uri ale unui calculator de buzunar, și astfel aș vedea in vedere folosirea de numere zecimale pentru toate operațiile.
* Calculatorului ii lipsește o funcție simplă de memorie, deși nu este extrem de folosită (după experiența proprie) ar mări capacitățile calculatorului.
* Adăugarea de alte operații des folosite precum ridicarea la pătrat sau radicalul în cazul în care se poate lucra cu numere zecimale.
* Introducerea datelor sub formă zecimala ar face calculatorul mult mai "user-friendly".
* Ar fi folositor ca calculatorul să arate pe afișor în timp real numărul pe care îl introduci, acest lucru poate fi luat în considerare.