**Procesor MIPS Ciclu Unic 16bit** (Raport)

Stef Paula – grupa 30221\_2- an 2020-2021

Cele 4 instructiuni pe care le-am ales suplimentar sunt

* Instructiunile de tip R : 1. XOR

2. SLT

* Instructiunile de tip I: 1. BNE

2. SLTI

Microprocesorul urmareste trasarea algoritmului lui Dijkstra folosindu-ne de o matrice de 4x4 ( reprezentat in memorie sub forma unui vector ), care retine pentru fiecare doua noduri adiacente, ponderea muchiei dintre acestea si urmareste calcularea celui mai scurt drum de la nodul 1 la celelalte noduri.

Implementarea programului in cod conform MIPS ISA

001\_000\_001\_0000010 -- 0.addi $1 $0 4 -> nr de noduri (4)

000\_000\_000\_010\_0\_000 -- 1.add $2 $0 $0 -> nodul de plecare

000\_000\_000\_011\_0\_000 -- 2.add $3 $0 $0 -> nodul curent

000\_000\_000\_100\_0\_000 -- 3.add $4 $0 $0 -> adresa la care gasim un element in matrice

000\_000\_000\_101\_0\_000 -- 4.add $5 $0 $0 -> u.d (ponderea minima a nodului de plecare momentan)

000\_000\_000\_110\_0\_000 -- 5.add $6 $0 $0 -> v.d (ponderea curenta minima a drumului)

000\_000\_000\_111\_0\_000 -- 6.add $7 $0 $0 -> w.d (ponderea nodului ce vrem sa il adaugam)

100\_001\_010\_0100011 -- 7.beq $2 $1 35 -> if(nodul de plecare != nr. de noduri) //while

100\_001\_011\_0011000 -- 8.beq $3 $1 24(linia33) -> if(nodul current != nr de noduri) //while

000\_010\_100\_100\_0\_000 -- 9.add $4 $2 $4

000\_010\_100\_100\_0\_000 -- 10.add $4 $2 $4

000\_010\_100\_100\_0\_000 -- 11.add $4 $2 $4 => adresa pentru matrice = nodul de plecare\*nr de noduri

000\_010\_100\_100\_0\_000 -- 12.add $4 $2 $4

000\_011\_100\_100\_0\_000 -- 13.add $4 $3 $4 -> adresa +=nodul curent

010\_100\_111\_0000000 -- 14.lw $7 0($4) -> pun ponderea in w.d

000\_000\_000\_100\_0\_000 -- 15.add $4 $0 $0 -> reinitializez adresa

001\_000\_100\_0010000 -- 16.addi $4 $0 16 -> dupa cele 16 casute alocate pentru matrice am 4 casute in care voi pune

ponderile minime ale drumurilor intre noduri (pentru 1 va fi implicit 0)

000\_010\_100\_100\_0\_000 -- 17.add $4 $2 $4 -> adaug valoarea nodului curent

010\_100\_101\_0000000 -- 18.lw $5 0($4) -> iau valoarea curenta din memorie pentru drum (initial va fi MAXIMA )

000\_100\_010\_100\_0\_001 --19.sub &4 $4 $2 -> reinitializez pentru a gasi si valoarea minima a nodului curent

000\_011\_100\_100\_0\_000 -- 20.add $4 $3 $4

010\_100\_110\_0000000 -- 21.lw $6 0($4) -> load v.d

000\_101\_100\_111\_0\_000 -- 22.add $7 $5 $4 adun la ponderea drumului valoarea nodului u.d +w.d

000\_100\_101\_111\_0\_111 -- 23.slt $7 $4 $6 if(v.d > u.d + w.d)

100\_001\_111\_0011111 -- 24.beq $7 $1 7(linia31)

000\_101\_000\_100\_0\_000 -- 25.add $4 $6 $0 -> v.d = u.d + w.d

000\_000\_000\_100\_0\_000 -- 26.add $4 $0 $0 ->initializam 4

001\_000\_100\_0010000 -- 27.addi $4 $0 16

000\_011\_100\_100\_0\_000 -- 28.add $4 $3 $4 -> calculam adresa la care vrem sa urcam noua valoare

011\_100\_101\_0000000 -- 29.sw $5 0($4) ->modificam valoarea in memorie

111\_0000000011000 -- 30.j 24 ->ne intoarcem la al doilea if

001\_000\_011\_0000001 -- 31.addi $3 $0 1 --

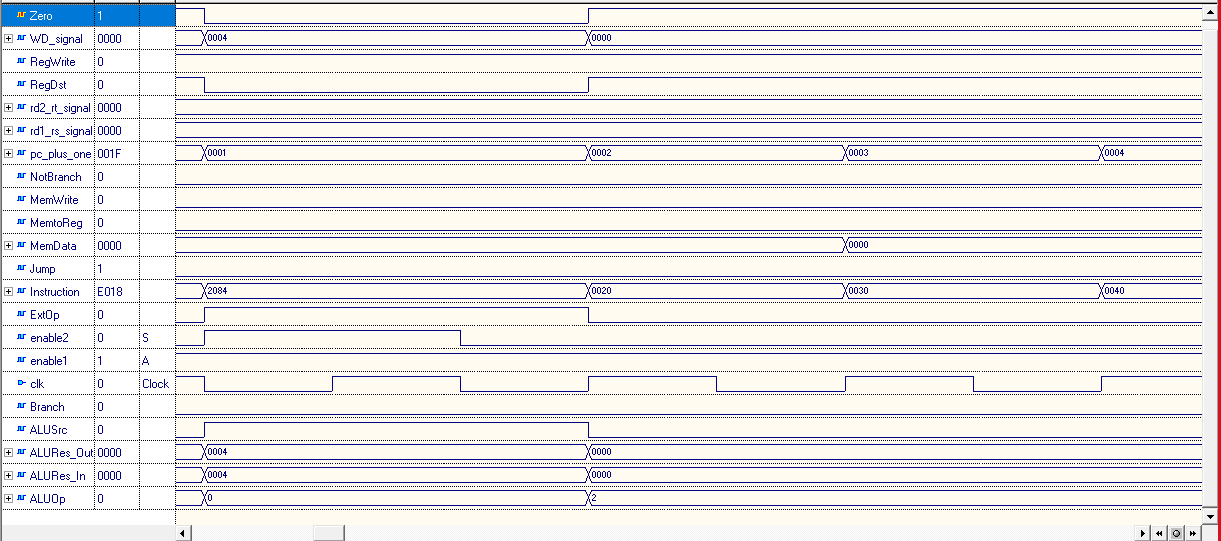
111\_0000000001000 -- 32.j 8

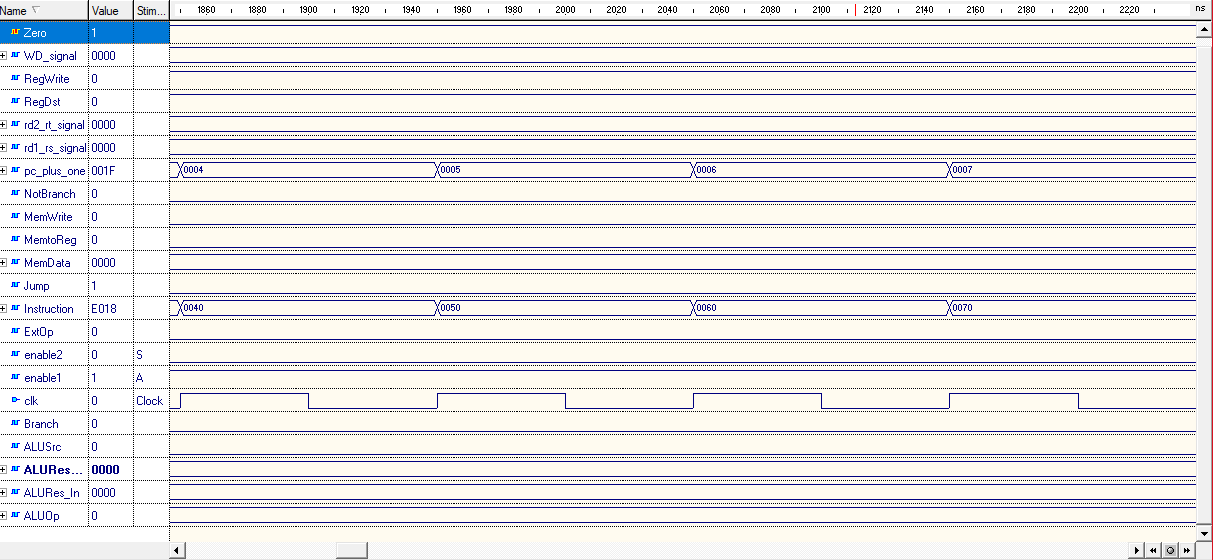
001\_000\_010\_0000001 -- 33.addi $2 $0 1

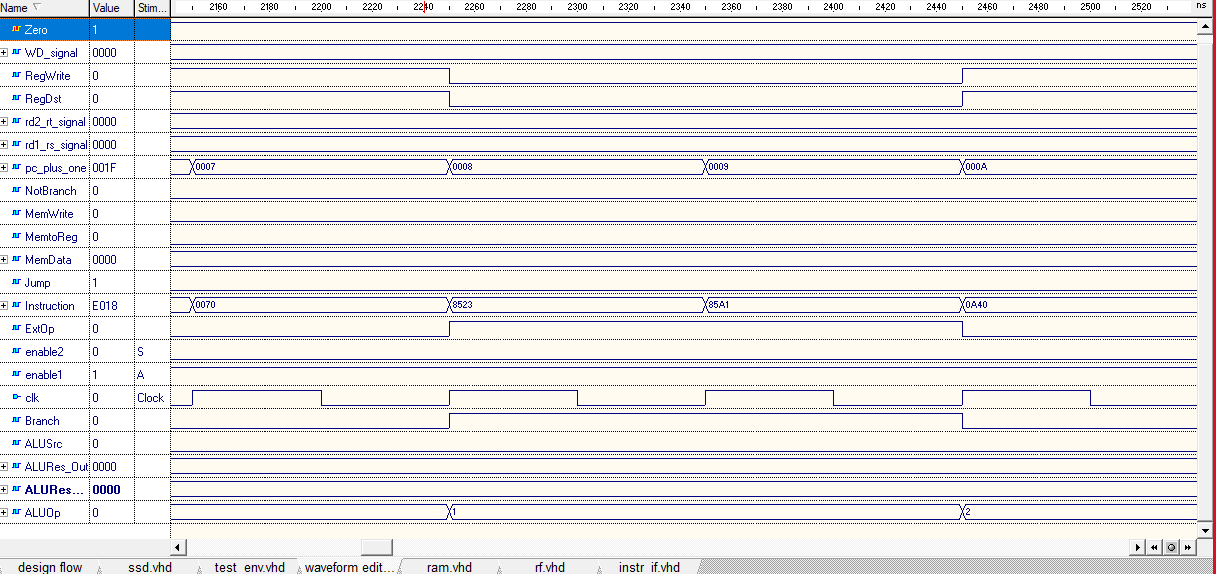
111\_0000000000111 -- 34.j 7

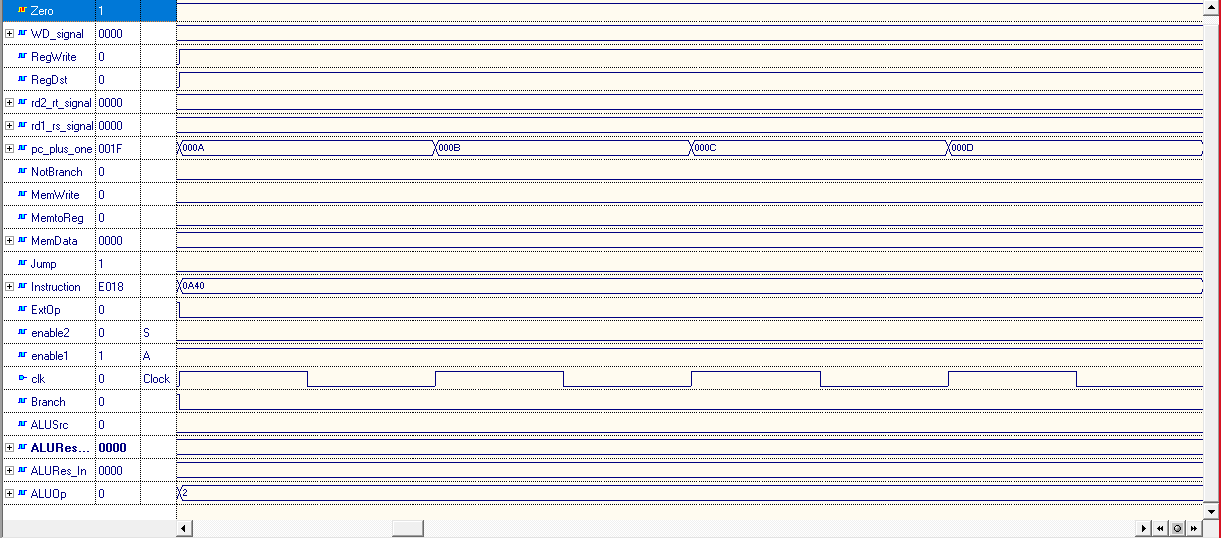
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instrucțiune** | **Opcode** *Instr(15-13)* | **RegDst** | **ExtOp** | **ALUSrc** | **Branch** | **Nbranch** | **Jump** | **JmpR** (opțional) | **MemWrite** | **MemtoReg** | **Reg Write** | **ALUOp (1:0)** | **func**  *Instr(2-0)* | **ALUCtrl (2:0)** |
| add | 000 | 1 | x | 0 | 0 | 0 | 0 |  | 0 | 0 | 1 | 10 | 000 | 000 |
| sub | 000 | 1 | x | 0 | 0 | 0 | 0 |  | 0 | 0 | 1 | 10 | 001 | 001 |
| sll | 000 | 1 | x | 0 | 0 | 0 | 0 |  | 0 | 0 | 1 | 10 | 010 | 010 |
| srl | 000 | 1 | x | 0 | 0 | 0 | 0 |  | 0 | 0 | 1 | 10 | 011 | 011 |
| and | 000 | 1 | x | 0 | 0 | 0 | 0 |  | 0 | 0 | 1 | 10 | 100 | 100 |
| or | 000 | 1 | x | 0 | 0 | 0 | 0 |  | 0 | 0 | 1 | 10 | 101 | 101 |
| xor | 000 | 1 | x | 0 | 0 | 0 | 0 |  | 0 | 0 | 1 | 10 | 110 | 110 |
| slt | 000 | 1 | x | 0 | 0 | 0 | 0 |  | 0 | 0 | 1 | 10 | 111 | 111 |
| addi | 001 | 0 | 1 | 1 | 0 | 0 | 0 |  | 0 | 0 | 1 | 00 |  | 000 |
| lw | 010 | 0 | 1 | 1 | 0 | 0 | 0 |  | 0 | 1 | 1 | 00 |  | 000 |
| sw | 011 | x | 1 | 1 | 0 | 0 | 0 |  | 1 | x | 0 | 00 |  | 000 |
| beq | 100 | x | 1 | 0 | 1 | 0 | 0 |  | 0 | x | 0 | 01 |  | 001 |
| bne | 101 | x | 1 | 0 | 0 | 1 | 0 |  | 0 | x | 0 | 01 |  | 001 |
| slti | 110 | 0 | 1 | 1 | 0 | 0 | 0 |  | 0 | 0 | 1 | 11 |  | 111 |
| j | 111 | x | x | x | x | x | 1 |  | 0 | x | 0 | xx |  | xxx |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

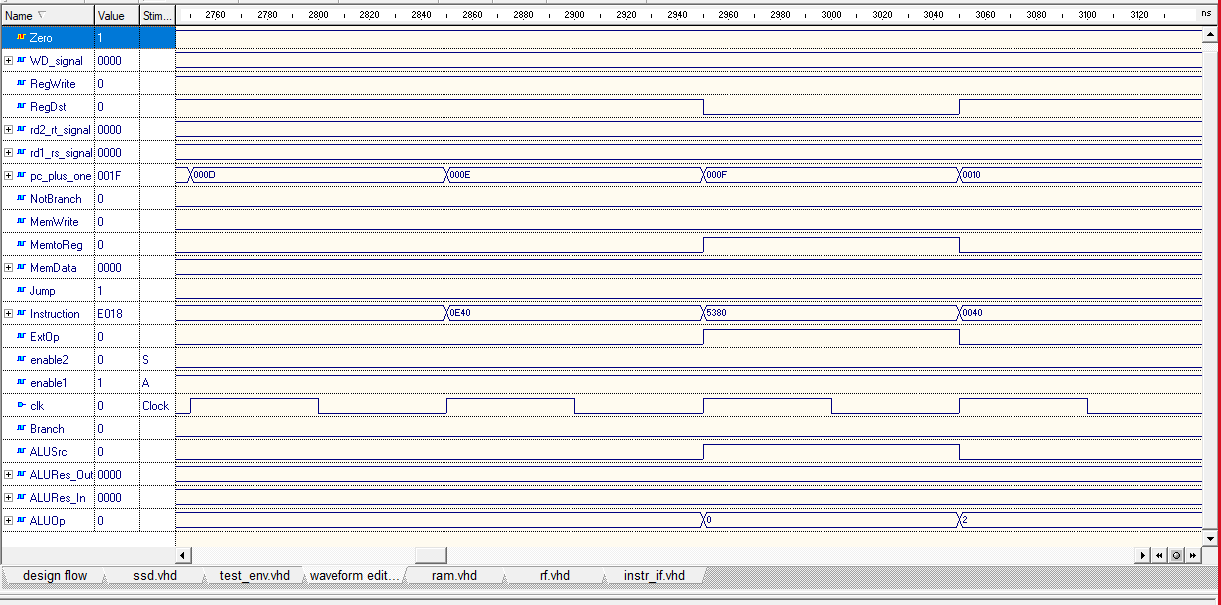
Trasarea :

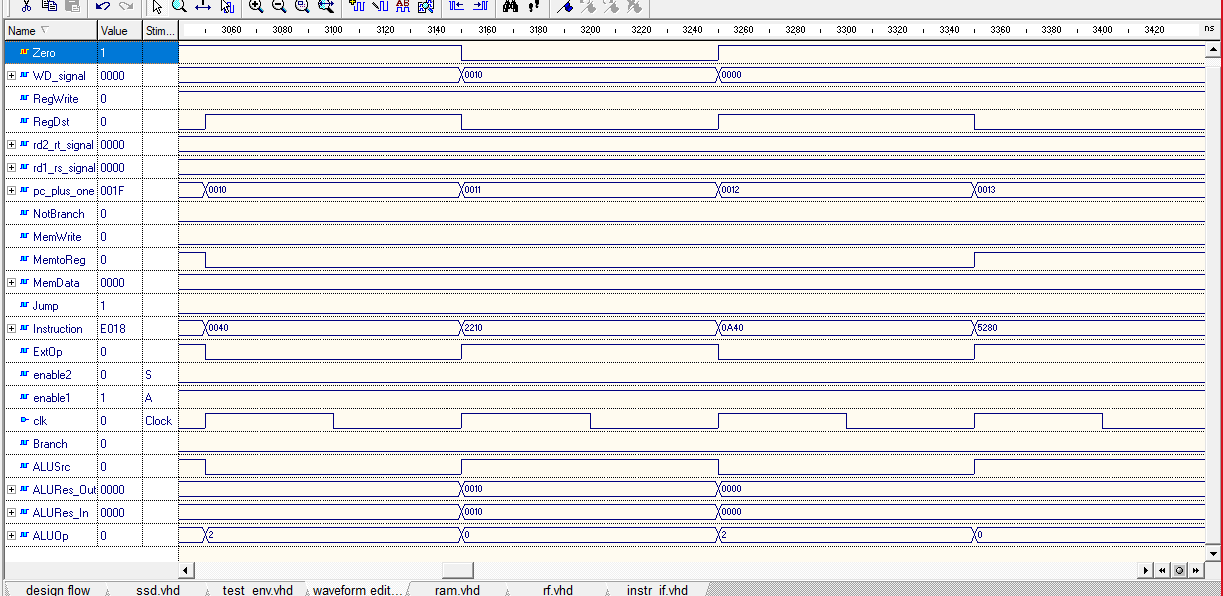


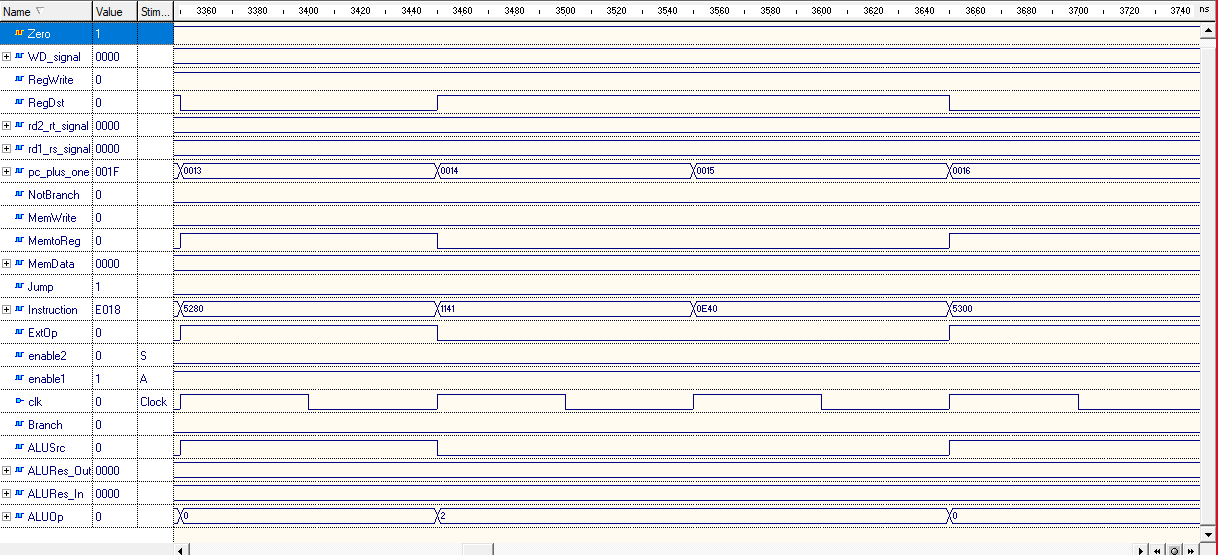


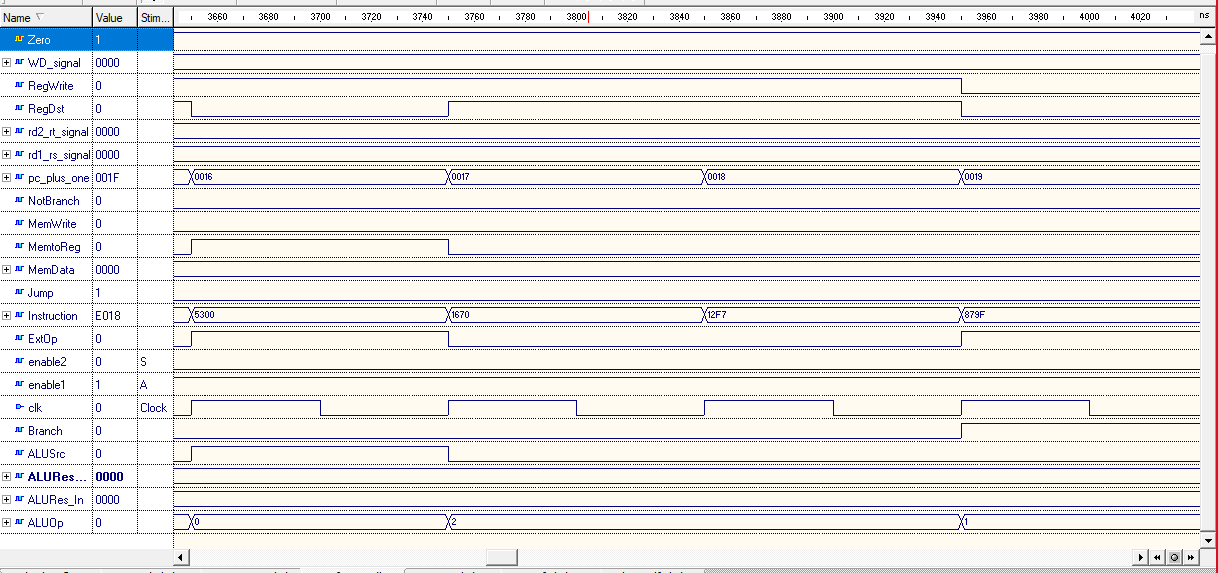


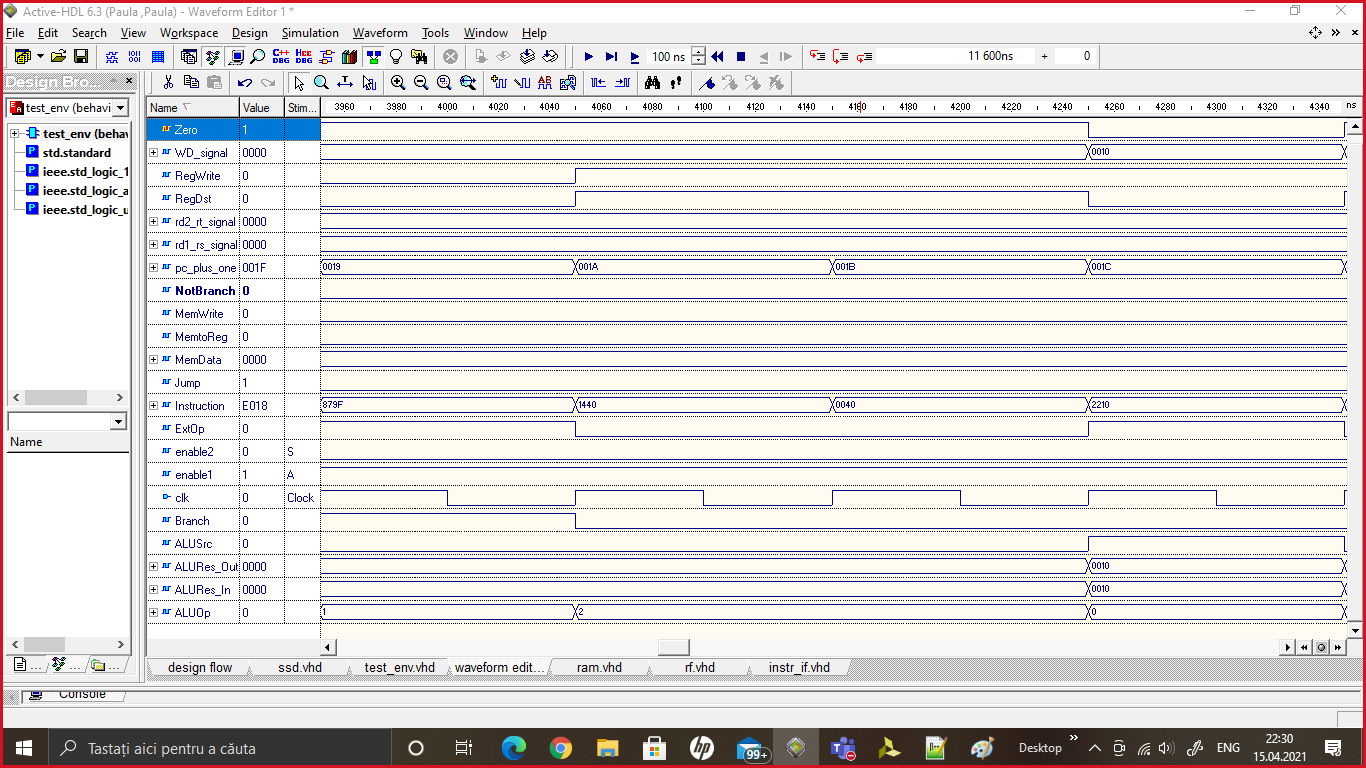


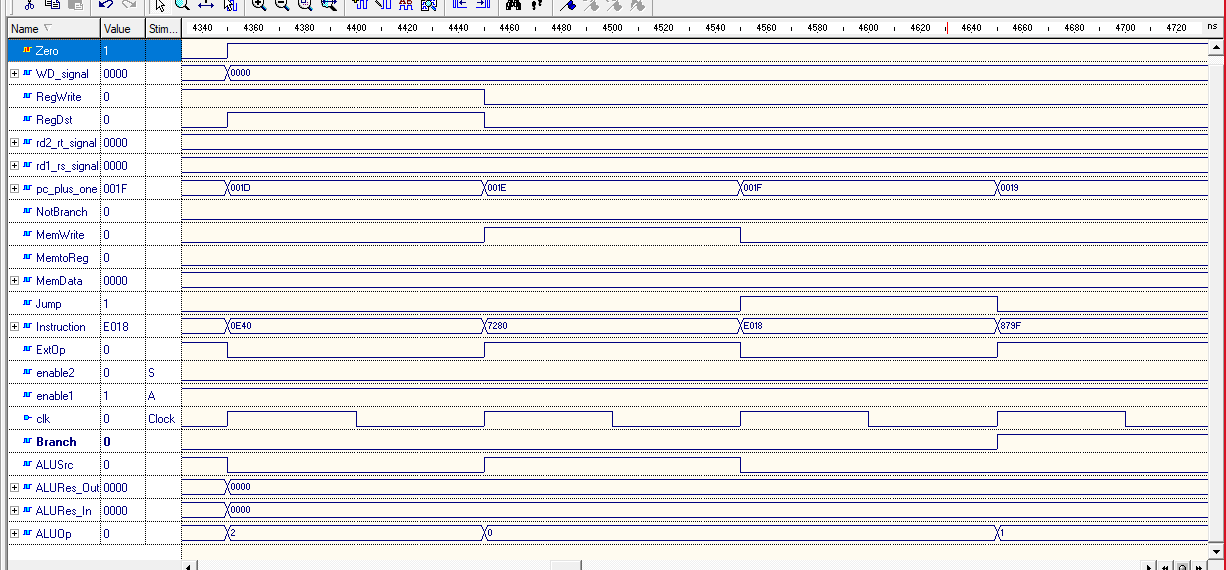




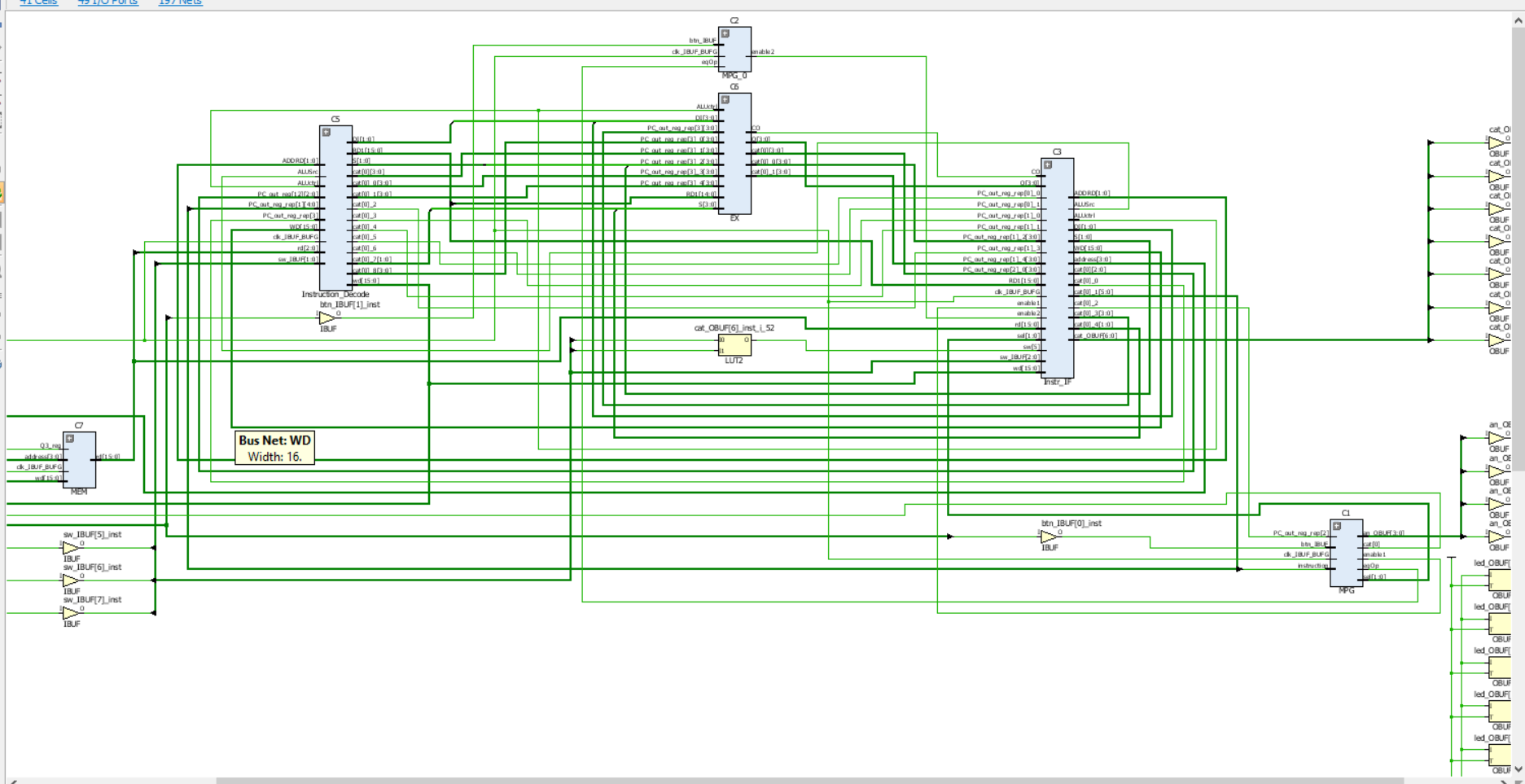








Schema generata in Vivado :



Am testat si procesorul si pe placa Basys3, am avut mici probleme la calcularea valorilor RD1 si RD2 (se observa si din trasarea in simulator).