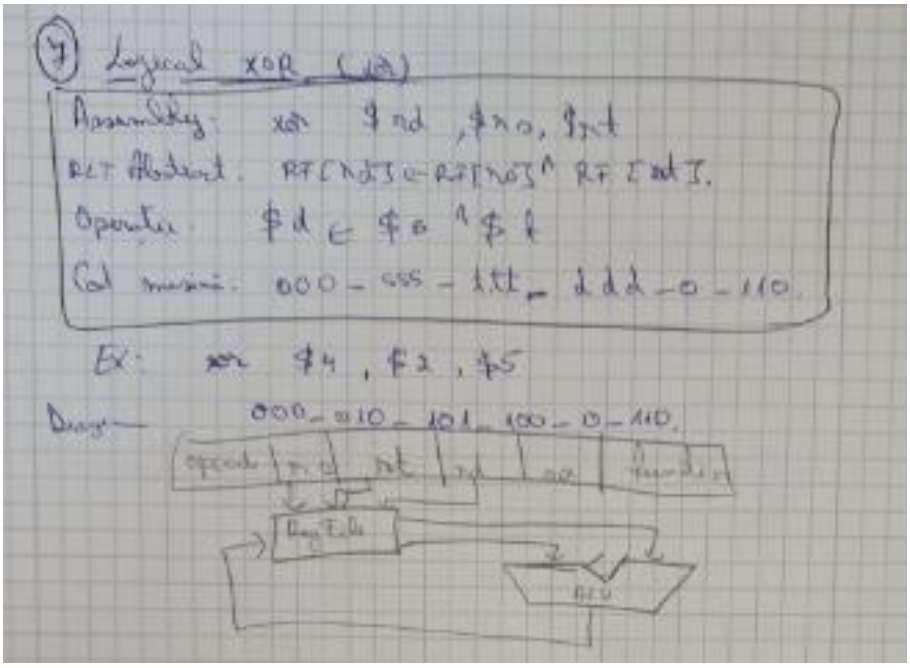
Raport MIPS 16

-Strujan Florentina-

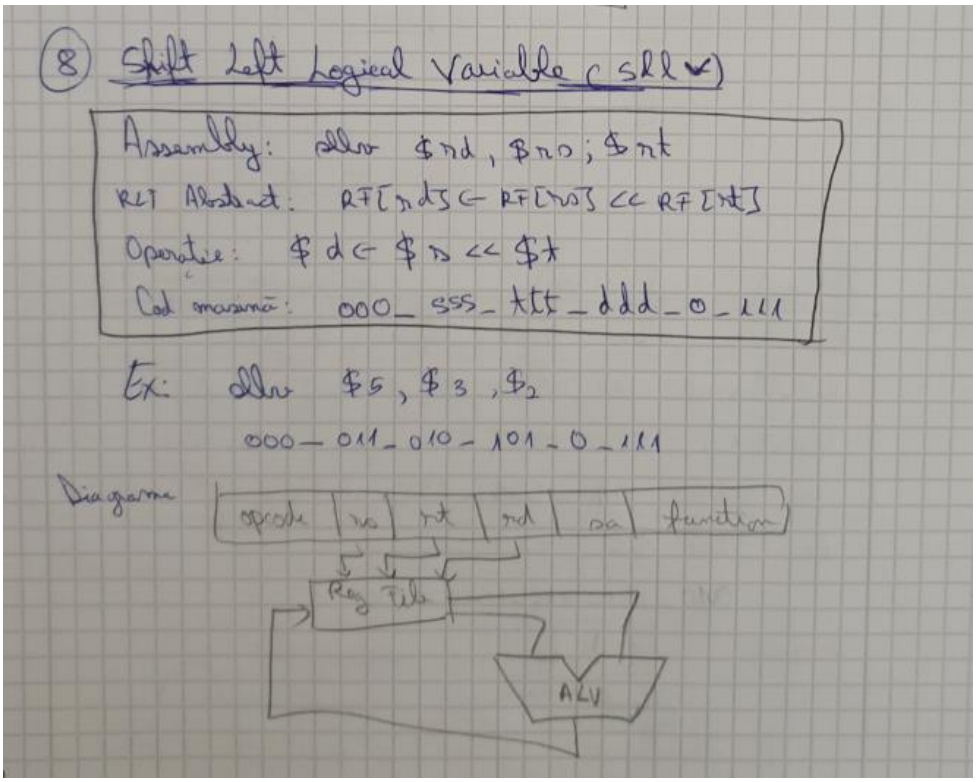
-Gr 302210-

a. <u>cele 4 instructiuni alese suplimentar</u>

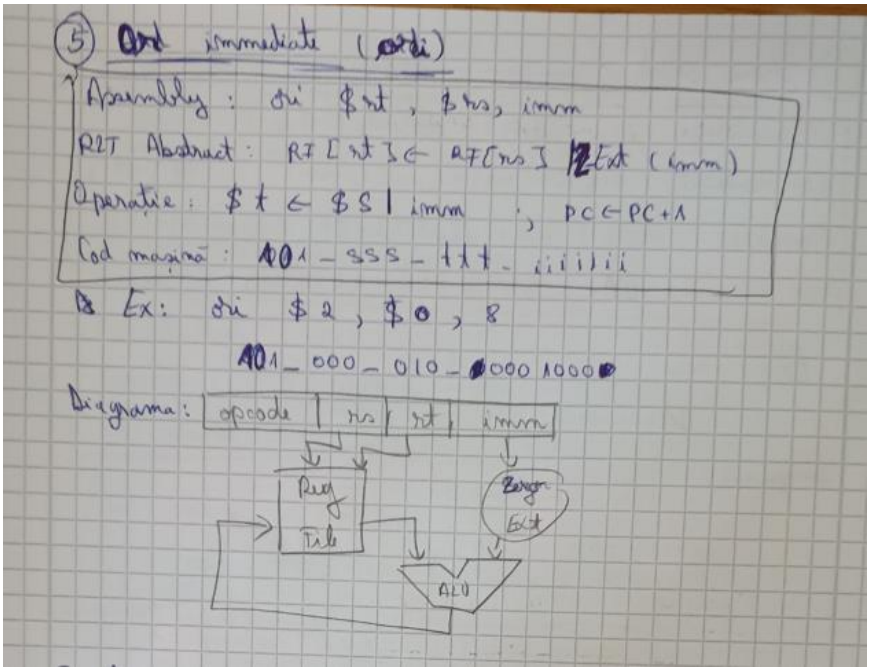
Logical XOR (tip R) – Operatia de sau exclusiv intre doua registre (rs, rt) cu punerea rezultatului intr-un al treilea registru (rd).



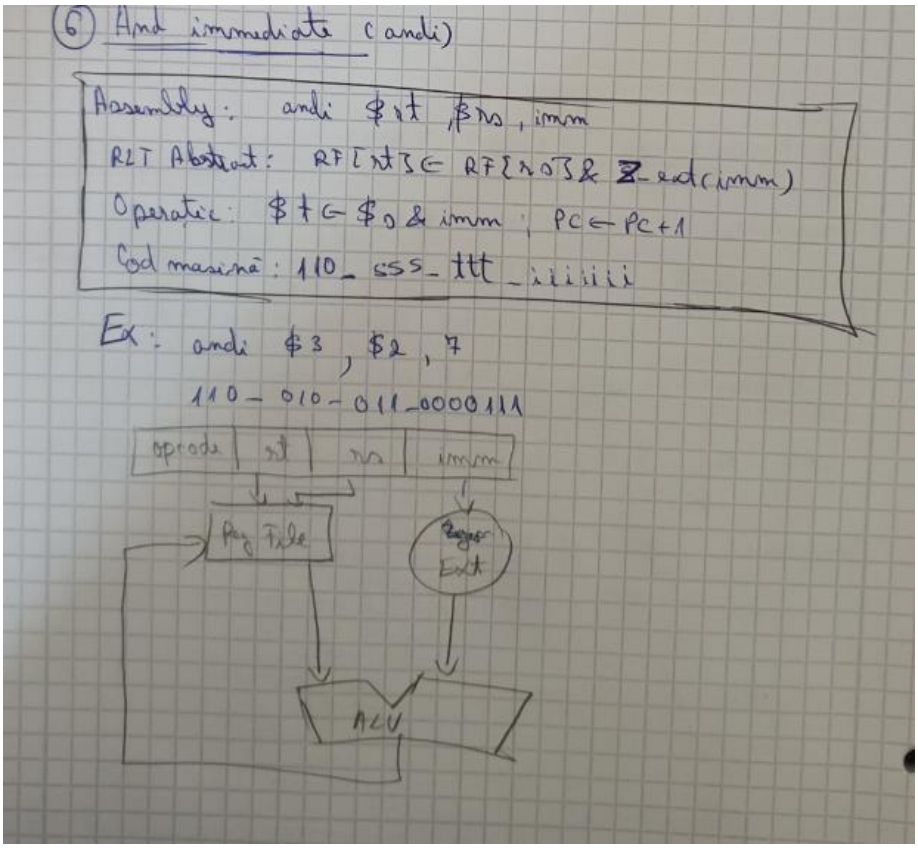
Shift Left Logical Variable(tip R) – Operatia de deplasare la stanga a unui registru(rs) cu un numar de pozitii dat de un alt registru(rt), rezultatul fiind pus intr-un al treilea registru(rd).



Or immediate(tip I) – Operatia de sau logic intre un registru(rs) si un imediat, cu salvarea rezultatului in alt registru (rt).



And immediate(tip I) – Operatia de si logic intre un registru(rs) si un imediat, cu salvarea rezultatului in alt registru (rt).



b. <u>tabelul cu valorile semnalelor de control pentru toate instructiunile, facut in laboratoarele anterioare.</u>

Instrucțiune	Opcode *Instr(15-13)*	RegDst	ExtOp	ALUSrc	Branch	Br<?> (opțional)	Jump	MemWrite	MemtoReg	RegWrite	ALUOp (2:0)	func *Instr(2-0)*	ALUCtrl (2:0)
ADD	000	1	X	0	0		0	0	0	1	010 (R)	000	000(+)
SUB	000	1	X	0	0		0	0	0	1	010(R)	001	001(-)
SLL	000	1	X	0	0		0	0	0	1	010(R)	010	101(<<l)
SRL	000	1	X	0	0		0	0	0	1	010(R)	011	110(>>l)
AND	000	1	X	0	0		0	0	0	1	010(R)	100	010(&)
OR	000	1	X	0	0		0	0	0	1	010(R)	101	011(|)
XOR	000	1	X	0	0		0	0	0	1	010(R)	110	100(^)
SLLV	000	1	X	0	0		0	0	0	1	010(R)	111	111(>>v)
ADDI	001	0	1	1	0		0	0	0	1	001(+)	XXX	000(+)
LW	010	0	1	1	0		0	0	1	1	001(+)	XXX	000(+)
SW	011	X	1	1	0		0	1	X	0	001(+)	XXX	000(+)
BEQ	100	X	1	0	1		0	0	X	0	011(-)	XXX	001(-)
ORI	101	0	0	1	0		0	0	0	1	100(|)	XXX	011(|)
ANDI	110	0	0	1	0		0	0	0	1	101(&)	XXX	010(&)
J	111	X	X	X	X		1	0	X	0	XXX	XXX	XXX

c. descrierea in cuvinte (si cod C daca e disponibil) a programului scris si executat de procesor (cel incarcat in memoria rom)

Programul scris si executat de processor reprezinta verificarea valorii 64 ca a 6-a putere a lui 2. Initial se initializeaza 3 registri cu valorile corespunzatoare: 2 (numarul a carui putere se va verifica), 64 (valoarea ce trebuie sa rezulte) si 6 (puterea corespunzatoare). Se va stoca in memorie val 6 la adresa 3, dupa care se va pune in alt registru, supa care initializez un registru cu 0. Apoi incrementez acest registru cu 1 (urmatoarea putere dupa 0) , iar pe cel initializat cu 2 il inmultesc cu 2 pentru a afla urmatoarea putere. Se va testa daca s-a ajuns la valoarea 64. Daca nu, se vor relua instructiunile incepand cu adunarea cu 1, iar in caz de succes se vor scade registrii corespunzatori puterii corecte si a celei rezultate si se va compara rezultatul cu 0. In caz de egalitate se va stoca in memorie la adresa 6 valoarea puterii, in caz contrat intrandu-se intr-o bucla infinita incepand cu stocarea la adresa 3 a lavorii 6.

d.trasarea executiei programului

Pas	SW(7:5)	"000"	"001"	"010"	"011"	"100"	"101"	"110"	"111"	Doar la salturi	
	Instr (*în asamblare*)	Instr (*hexa*)	PC+1	RD1	RD2	Ext_Imm	ALURes	MemData	WD	BranchAddr	JumpAddr
Init	ADDI	X"2082"	x"0001"	x"0000"	-	x"0002"	x"0002"	-	X"0002"	-	-
1	ADDI	X"2140"	x"0002"	X"0000"	-	X"0040"	X"0040"	-	X"0040"	-	-
2	ORI	X"A186"	x"0003"	X"0000"	-	X"0006"	X"0006"	-	X"0006"	-	-
3	SW	X"6183"	x"0004"	X"0000"	-	X"0003"	X"0003"	-	-	-	-
4	LW	X"4203"	x"0005"	X"0000"	-	X"0003"	X"0003"	X"0006"	X"0006"	-	-
5	SUB	X"0E31"	x"0006"	X"0006"	X"0006"	-	X"0000"	-	X"0000"	-	-
6	ADDI	X"2D81"	x"0007"	X"0000"	-	X"0001"	X"0001"	-	X"0001"	-	-
7	SLL	X"009A"	x"0008"	X"0002"	-	X"0001"	X"0004"	-	X"0004"	-	-
8	BEQ	X"8881"	x"0009"	X"0004"	-	X"0001"	X"003C"	-	-	X"000A"	-
9	J	X"E006"	x"0006"	-	-	-	-	-	-	-	X"0006"
10	ADDI...	X"11D9"	x"0007"	X"0001"	-	X"0001"	X"0002"	-	X"0002"	-	-

e. daca exista activitati (parti din procesor) incomplete din laboratoarele 4-7 se va mentiona explicit acest lucru in raport

Nu exista.

f. corectitudinea descrierii vhdl (este totul descris in vhdl, exista erori?, print screen cu RTL schematic la nivelul de sus unde se vad unitatile implementate IF, ID, EX, etc)

Intregul cod a fost descris in vhdl. Pe parcursul implementarii am avut 3 erori:

Eroare 1. Din cauza unui proces din IF

[Place 30-574] Poor placement for routing between an IO pin and BUFG. If this sub optimal condition is acceptable for this design, you may use the CLOCK_DEDICATED_ROUTE constraint in the .xdc file to demote this message to a WARNING. However, the use of this override is highly discouraged. These examples can be used directly in the .xdc file to override this clock rule.

< set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets sw_IBUF[1]] >

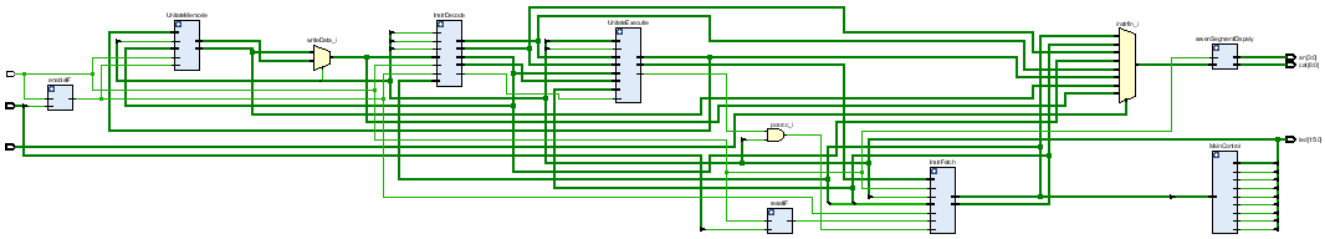
sw_IBUF[1]_inst (IBUF.O) is locked to IOB_X0Y12

and sw_IBUF_BUFG[1]_inst (BUFG.I) is provisionally placed by clockplacer on BUFGCTRL_X0Y0

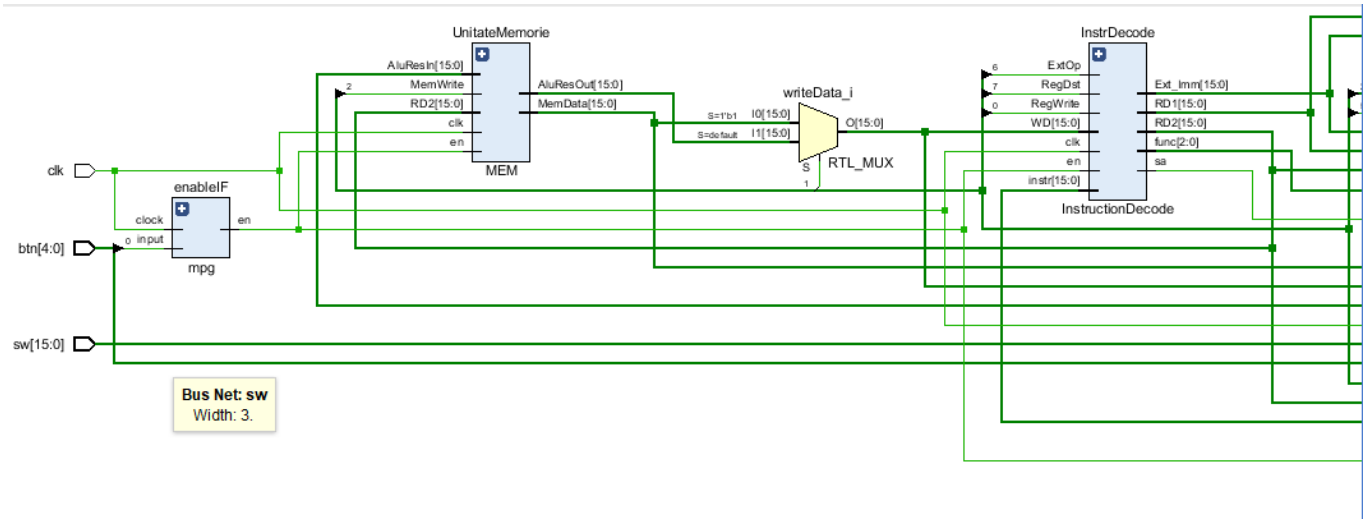
Eroare 2. Intrarile de la RF erau lasate pe 4 biti, uitand sa le schimb la 3 biti ca adaptare pentru MIPS 16.

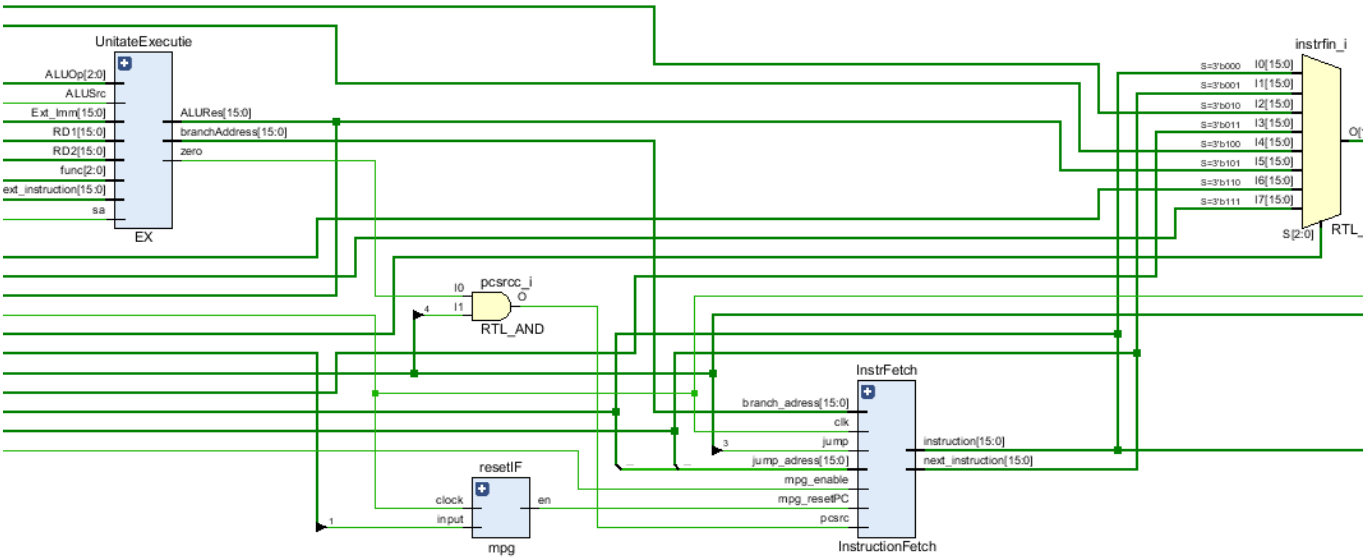
Eroare 3. Nu am pus when others la un case.

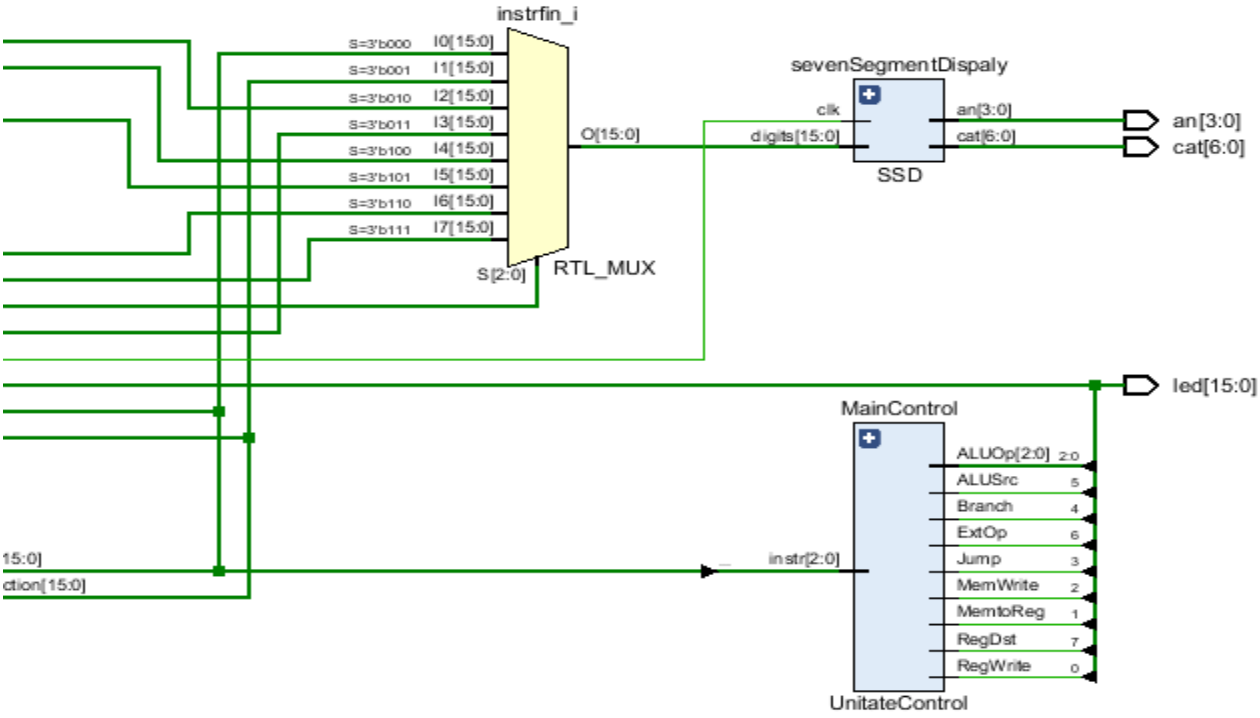
[Synth 8-426] missing choice(s) 3'b000 in case statement



Din cauza dificultatii intelegerii, am impartit in 3 schema.


UnitateMemorie
AluResIn[15:0]
MemWrite
RD2[15:0]
clk
en
AluResOut[15:0]
MemData[15:0]
MEM
writeData_i
S=1'b1
S=default
I0[15:0]
I1[15:0]
O[15:0]
S
RTL_MUX
InstrDecode
ExtOp
RegDst
RegWrite
WD[15:0]
clk
en
instr[15:0]
Ext_Imm[15:0]
RD1[15:0]
RD2[15:0]
func[2:0]
sa
InstructionDecode
clk
enableIF
clock
input
en
mpg
btn[4:0]
sw[15:0]
Bus Net: sw
Width: 3.


UnitateExecutie
ALUOp[2:0]
ALUSrc
Ext_Imm[15:0]
RD1[15:0]
RD2[15:0]
func[2:0]
ext_instruction[15:0]
sa
ALURes[15:0]
branchAddress[15:0]
zero
EX
instrfin_i
S=3'b000 I0[15:0]
S=3'b001 I1[15:0]
S=3'b010 I2[15:0]
S=3'b011 I3[15:0]
S=3'b100 I4[15:0]
S=3'b101 I5[15:0]
S=3'b110 I6[15:0]
S=3'b111 I7[15:0]
S[2:0]
pcsrcc_i
I0
I1
O
RTL_AND
InstrFetch
branch_adress[15:0]
clk
jump
jump_adress[15:0]
mpg_enable
mpg_resetPC
pcsrc
instruction[15:0]
next_instruction[15:0]
InstructionFetch
resetIF
clock
input
en
mpg


instrfin_i
S=3'b000 I0[15:0]
S=3'b001 I1[15:0]
S=3'b010 I2[15:0]
S=3'b011 I3[15:0]
S=3'b100 I4[15:0]
S=3'b101 I5[15:0]
S=3'b110 I6[15:0]
S=3'b111 I7[15:0]
S[2:0]
O[15:0]
RTL_MUX
sevenSegmentDispaly
clk
digits[15:0]
an[3:0]
cat[6:0]
SSD
an[3:0]
cat[6:0]
led[15:0]
MainControl
instr[2:0]
ALUOp[2:0]
ALUSrc
Branch
ExtOp
Jump
MemWrite
MemtoReg
RegDst
RegWrite
UnitateControl

g. daca a fost testat pe placa si este functional

Nu a fost testat pe placa din cauza ca nu detin una.

Link spre activitatile restante la laborator (imagini) :

https://drive.google.com/open?id=1LcZoLnuYWtHCWpS0CKhBooBv4rqAhcSH