Organizarea Calculatoarelor
LABORATOR 9 – Implementare MIPS pe placa S3

SCOPUL LUCRĂRII

Procesorul MIPS se va implementa pe placa S3 şi va fi testat cu un program care calculează

$$r = 3a - 2b$$

unde *a* şi *b* sunt doi operanţi de 4 biţi ce se vor citi prin intermediul comutatoarelor glisante. Rezultatul *r* se va afişa prin intermediul celor 4 afişoare cu 7 segmente.

Aspecte practice.

Proiectul ISE pentru simulare foloseşte pentru operaţiile de intrare/ieşire 3 locaţii speciale din memoria de date, detaliate anterior.

Reamintiţi-vă:

- locaţia de la adresa **x"0000_0040"** este de tipul **„Read only"** şi întotdeauna data citită din această locaţie are valoarea marcherului I/O (portului) de intrare **INW0**.
- locaţia de la adresa **x"0000_0044"** este de tipul **„Read only"** şi întotdeauna data citită din această locaţie are valoarea marcherului I/O (portului) de intrare **INW1**.
- Locaţia de la adresa **x"0000_0048"** este de tipul **„Write only"**. Data scrisă în această locaţie va apare în exteriorul sistemului prin intermediul marcherului I/O (portului) de ieşire **OUTW0**.

Toate cele trei locaţii, **INW0**, **INW1** şi **OUTW0,** au lăţimea de căi de date MIPS, adică 32 de biţi.

Pentru implementarea proiectului ar fi nevoie de o placă de dezvoltare cu cel puţin 62 de comutatoare şi 32 de LED-uri. Deoarece placa de care dispunem are numai 8 comutatoare, 8 LED-uri şi 4 afişoare 7-segmente, vom construi un modul de interfață cu următoarele care face următoarele prelucrări:

- Va genera semnalul **INW0** astfel:
  - biţii **INW0(31:4)** sunt conectați la masă. Valoarea lor este întotdeauna ‚0' logic.
  - biţii **INW0(3:0)** sunt conectați la comutatoarele **SW3** ÷ **SW0** de pe placa S3 (vezi documentaţia plăcii S3 disponibilă la http://www.cs.ucv.ro/~lemeni/Downloads/S3BOARD-rm.pdf#) sau placa S3.

  În concluzie interfaţă va genera **INW0(31:0)** după cum urmează:

```
INW0(31 downto 4) <= x"0000_000";

INW0(3)           <= SW3;
INW0(2)           <= SW2;
INW0(1)           <= SW1;
INW0(0)           <= SW0;
```

- Va genera semnalul **INW1(31:0)** similar cu modul de generare al lui **INW0**:
  - biţii **INW1(31:4)** sunt conectați la masă. Valoarea lor este întotdeauna ‚0' logic.
  - biţii **INW1(3:0)** sunt conectați la comutatoarele **SW7** ÷ **SW4** de pe placa S3 .

  Interfaţă va genera **INW0(31:0)** după cum urmează:

```
INW1(31 downto 4) <= x"0000_000";

INW1(3)           <= SW7;
INW1(2)           <= SW6;
INW1(1)           <= SW5;
INW1(0)           <= SW4;
```

- Ieşirea OUTW0 va apare în exteriorul sistemului prin intermediul celor 4 afişoarelor 7 segmente. În funcţie de starea pushbutonului **BTN0** se va afişa fie jumătatea inferioară, fie jumătatea superioară a acestei locaţii. Pentru afişarea prin intermediul celor 4 afişoarelor 7 segmente este nevoie de porturile de ieşire **AN3**, **AN2**, **AN1** şi **AN0** pentru anozi şi de porturile de ieşire **a**, **b**, **c**, **d**, **e**, **f** şi **g** pentru segmente. Modul de conectare al acestor porturi se va detalia în momentul definirii fişierului de constrângeri.

Pentru implementare pe placa S3 modulul de interfaţă definit mai sus ar trebui adăugat la proiectul MIPS. Interfaţă se conectează cu MIPS prin semnalele **INW0**, **INW1** şi **OUTW0** şi cu placa S3 prin porturile (markerii IO) comutatoarele **SW0,…**, **SW7**, pushbutonul **BTN0**, anozii **AN0,….AN3** şi segmentele **a, b, …g**. Conectarea ar trebui făcută (NU o faceţi!) ca în figura următoare:

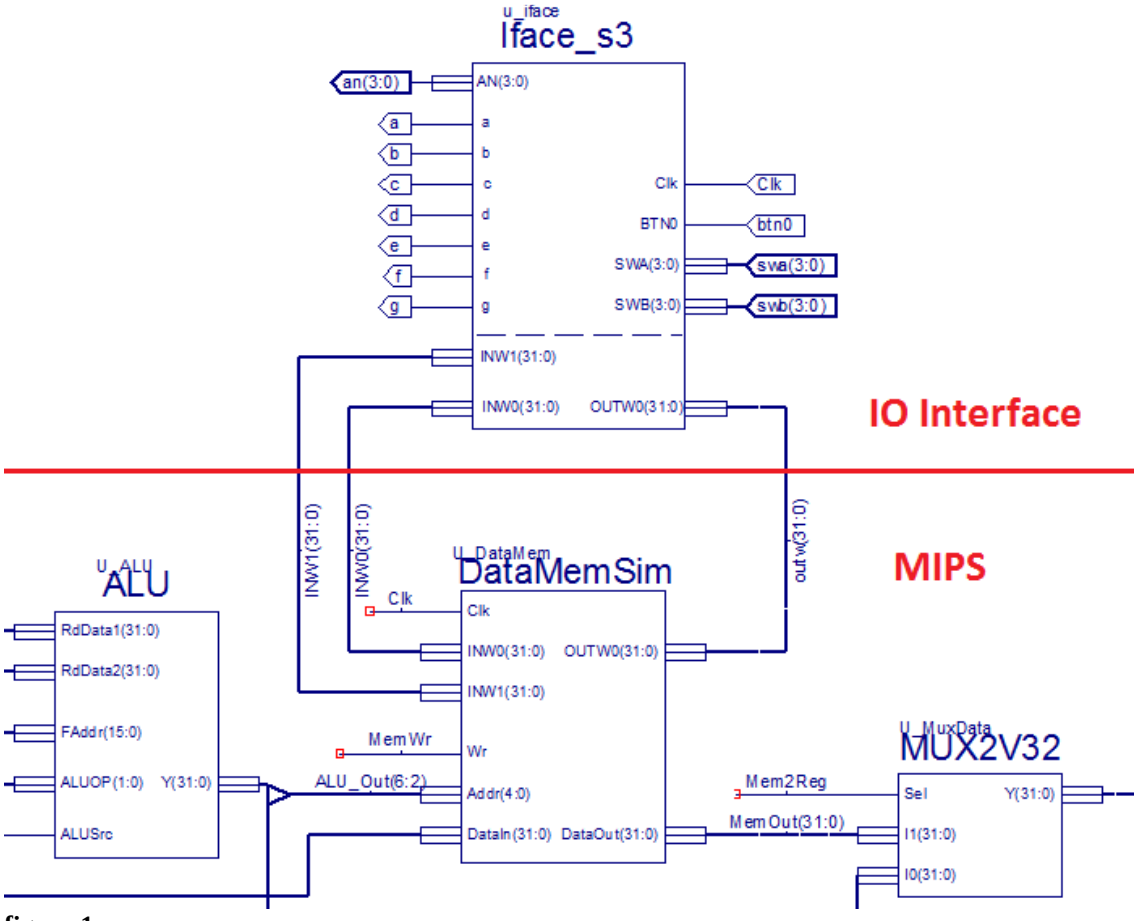

figura 1

Implementarea din figura 1 prezintă un dezavantaj fundamental. Deoarece semnalele INW0 şi INW1 au primii 28 de biţi zero, anumite aspecte funcţionale se vor testa foarte greu sau nu se vor mai putea testa de loc. Un astfel de aspect este depăşirea în cazul adunării. Cum mai putem testa semnalizarea depăşirii dacă cea mai mare valoare atât pentru INW0 cât şi pentru INW1 este 0x0000_000f ? Adunarea INW0+INW1=0x0000_000f+0x0000_000f =0x0000_001e nu produce depăşire!

Soluţia la problema expusă mai sus se bazează pe un fapt pe care probabil l-aţi remarcat şi dumneavoastră: un testbench se poate genera pentru orice modul din proiect, nu numai pentru modulul din vârful ierarhiei. Pentru ca să putem testa MIPS cu orice valoare a lui INW0 şi INW1, aşa cum am procedat în laboratorul precedent, **nu** trebuie să **modificăm** schema **MIPS**.

Implementarea MIPS rămâne ca în laboratorul precedent. **Principial**, la MIPS **se va conecta** modulul de interfaţa după cum urmează:

1. Pentru MIPS aşa cum este el implementat acum se creează simbolul corespunzător.
2. Se adaugă la proiect o nouă planşe.
3. Prin intermediul simbolului creat la punctul 1 vom cupla MIPS cu modulul de interfață. Cuplarea se face pe noua planşă adăugată la punctul 2. Această nouă planşe devine topul proiectului.

În implementarea schiţată anterior MIPS devine un modul şi poate fi simulat ca şi până acum. În plus se face şi conexiunea cu modulul de interfaţă.

În continuare se va trece la implementare.

Pasul 1: Schema sistemului

Pentru cuplarea cu modulul de interfaţă procedaţi după cum urmează:

1. Deoarece se vor face modificări în ierarhie faceți o copie de siguranţă a folderului proiectului MIPS. Deschideţi proiectul MIPS şi executați Cleanup Project File din meniul Project. Creaţi o arhivă zip a proiectului MIPS – este foarte important!
2. Redenumiţi folderul MIPS (folderul ce conţine proiectul) ca MIPS_Sys. În folderul MIPS_Sys redenumiţi proiectul **MIPS.ise** ca **MIPS_Sys.ise**. Această operație este necesară pentru a nu crea confuzie între proiectul procesorului MIPS şi proiectul sistemului care va conţine procesorul MIPS: arhiva zip va conţine numai procesorul MIPS iar MIPS_sys va conţine procesorul mips plus interfaţa cu placa S3.
3. Deschideţi proiectul MIPS_sys.
4. Creați un simbol pentru schema MIPS care se află în momentul de faţă în vârful ierarhiei.
5. Adăugați la proiect un nou fişier de tip schematic cu numele **mips_sys**. Pe această schemă vom desena sistemul alcătuit din MIPS şi din modulul de interfaţă.
6. Extrageţi din arhiva lucrări de laborator fişierele care descriu modulul interfaţă. Copiaţi fişierele **iface_s3.vhd** şi **iface_s3.sym** în folderul proiectului (MIPS_sys).
7. Adăugaţi fişierul **iface_s3.vhd** la proiect. Pentru aceasta executaţi **Add Source** în fereastra **Sources**.
8. Cuplaţi MIPS cu modulul de interfață conform figurii următoare:

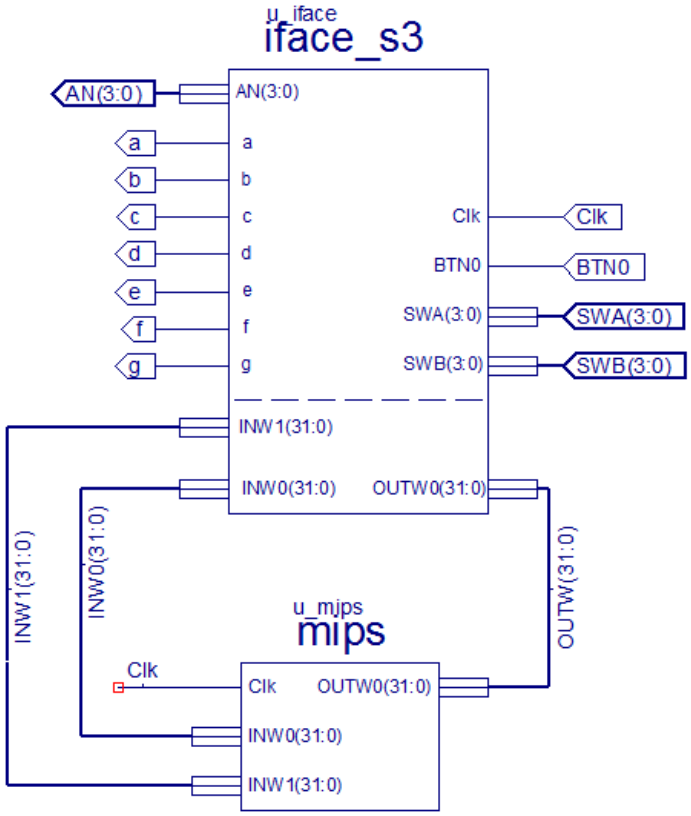

figura 2

Pinii modulului de interfaţa se împart în două categorii : pini pentru conectarea cu MIPS şi pini pentru conectarea cu placa S3.

Pinii pentru conectarea cu MIPS sunt INW0, INW1 şi OUTW0. Aceşti pini se conectează cu pinii cu acelaşi nume ai modulului MIPS. Magistralele prin care se realizează aceste conexiuni primesc numele pinilor pe care îi conectează. De exemplu, magistrala care conectează INW0 MIPS cu INW0 interfaţă se va numi INW0.

Ceilalţi pini ai modulului interfaţă sunt destinaţi conectării cu placa S3. Marcherii IO care se conectează la aceşti pini primesc numele pinului la care se conectează. Astfel:

- pinii SWA(3:0) se vor conecta prin intermediul marcherului SWA(3:0) la comutatoarele glisante SW(3:0). Comutatoarele SW(3:0) se folosesc la generarea lui INW0.
- pinii SWB(3:0) se vor conecta prin intermediul marcherului SWB(3:0) la comutatoarele glisante SW(7:4). Comutatoarele SW(7:4) se folosesc la generarea lui INW1.
- Pinii AN(3:0), a, b, c,.., g se folosesc pentru conectarea cu afişajul multiplexat. Nu trebuie să ştiţi cum funcţionează afişare. Este suficient să faceţi conexiunile ca în figura 2. Veţi învăţă modul de funcţionare al afişorului în semestrul următor.

Verificaţi noua schemă. Sintetizaţi. Dacă aveţi erori, corectaţi-le. Dacă aţi încercat şi nu aţi reuşit, chemaţi profesorul!

Pasul 2: Crearea programului de test pentru implementare.

Scrieţi un program în asamblare MIPS care să execute operaţia 3**a**-2**b**. Acesta se va numi **test3a2b.asm** şi se va memora în directorul **asm**.

Operandul **a** se citeşte prin intermediul comutatoarelor **SWA(3:0)** iar operandul **b** prin intermediul comutatoarelor **SWB(3:0)**. Valoarea calculată se va scrie în locaţia **OUTW0** pentru a putea fi afişată pe cele 4 afişoare cu 7 segmente.

Plecând de la un fişier ROM32x32 deja existent, creaţi fişierul **ROM32x32_3a2b.vhd** pentru a insera codul asamblat cu MARS. Procedaţi asemănător creării fişierului **ROM32x32_test1.vhd** din laboratorul precedent (la pasul 1).

Indicaţii pentru a nu avea probleme în simulare:

1. Prima instrucţiune din program trebuie să fie o instrucţiune fără efect, ca de exemplu ADD $0, $0, $0. Această instrucţiune dummy este necesară pentru că de multe ori setarea valorilor INW0 şi INW1 în testbench nu se face suficient de devreme înainte de primul front ridicător al ceasului. Dacă prima instrucţiune este *lw* aceasta poate citi valoarea xxxx_xxxx în loc de valoarea setată.
2. Testbench-ul pentru sistem se va numi **tb_sys** pentru a-l deosebi de testbench-ul pentru MIPS (creat în laboratorul precedent). Creaţi **tb_sys** inspirându-vă din indicaţiile de la pasul 3 din laboratorul precedent. **Asociaţi** testbenchul **tb_sys** cu modulul **mips_sys**.
3. Valoarea calculată **nu se poate vizualiza** prin intermediul porturilor **AN** şi a segmentelor **a**, **b**, …, **g**. Pentru a vizualiza valoarea calculată adăugaţi la fereastra Wave semnalul OUTW0. Acest este disponibil la nivelul uut.

Faceţi simulare Behavioral. În momentul în care funcţionează, chemaţi profesorul pentru validare.

Dacă aţi ajuns aici aveţi nota 5.

Pasul 3: Crearea fişierul de constrângeri

Creaţi fişierul de constrângeri conform tabelului următor:

Top IO Marcher	Resursă S3	Pin FPGA	Top IO Marcher	Resursă S3	Pin FPGA	Top IO Marcher	Resursă S3	Pin FPGA
SWA(0)	SW0	…	AN(0)	AN0	…	a	A	…
SWA(1)	SW1	…	AN(1)	AN1	…	b	B	…
SWA(2)	SW2	…	AN(2)	AN2	…	c	C	…
SWA(3)	SW3	…	AN(3)	AN3	…	d	D	…
SWB(0)	SW4	…			…	e	E	…
SWB(1)	SW5	…			…	f	F	…
SWB(2)	SW6	…			…	g	G	…
SWB(3)	SW7	…			…			…
BTN0	BTN0							
CLK	50 MHz	T9						

După ce aţi specificat alocarea pinilor **salvaţi şi închideţi** PACE.

Constrângerile din tabelul de mai sus sunt de tip **LOCATE** şi specifică modul de conectare al marcherilor I/O ai modulul top la pinii FPGA. Există însă şi un al doilea tip de constrângeri, şi anume **constrângerile de timp**.

Proiectul MIPS foloseşte ca semnal de ceas semnalul CLK care se conectează prin intermediul **pinului T9** la un oscilator extern cu frecvenţa de 50MHz. Software-ul ISE trebui să ştie această valoare şi să facă implementarea astfel încât proiectul să poată funcţiona la frecvenţa de 50 MHz sau mai mult.

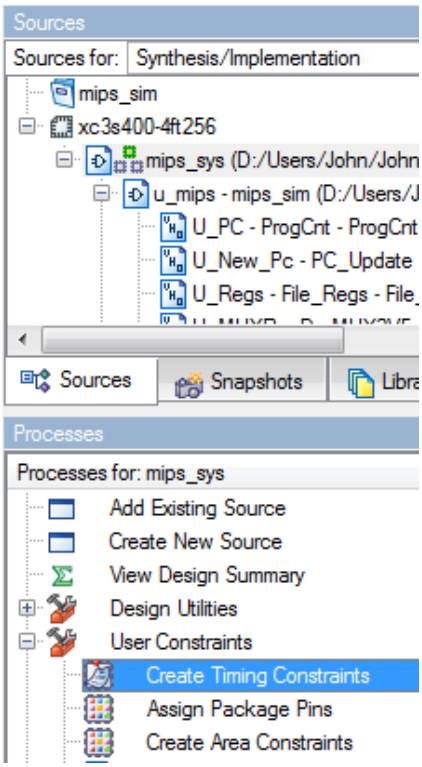

figura 3

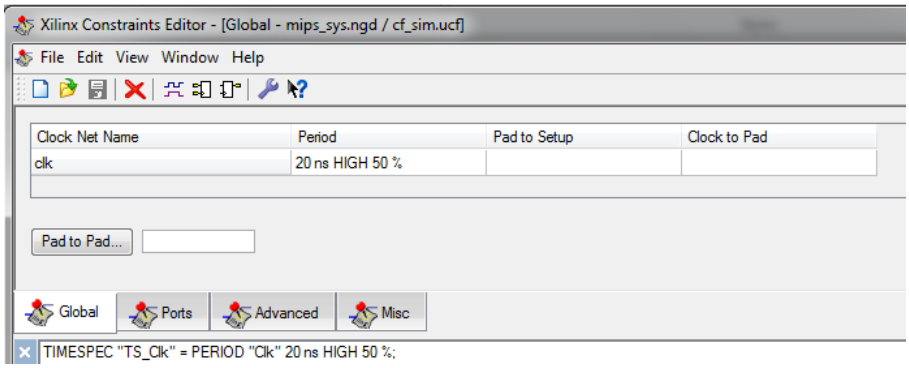

figura 4

Pentru a specifică perioada ceasului, selectaţi „Create Timing Constraints”, ca în figura 3. Va apare fereastra aferentă editorului de **constrângeri de tim**p. Selectaţi câmpul tab **Global**. Deoarece ceasul extern are frecvenţa de 50MHz, vom specifica o perioada de 20 ns. În câmpul **Period** se va introduce 20 iar apoi se va apăsa ENTER, fără să se specifica ns. Trebuie să obţineţi situaţia din figura 4.

Salvaţi şi închideţi editorul de constrângeri!

Pasul 4: Implementarea, generarea fişierului .bit şi testarea pe placa S3.

ATENTIE: Înainte de implementare trebuie schimbată o opţiune implicită a procesului de sinteză. Pentru aceasta, în fereastra **Processes** faceţi clic dreapta pe **Synthesize** şi în meniul contextual ce va apare selectaţi **Properties…**. Va apare fereastra din figura următoare:

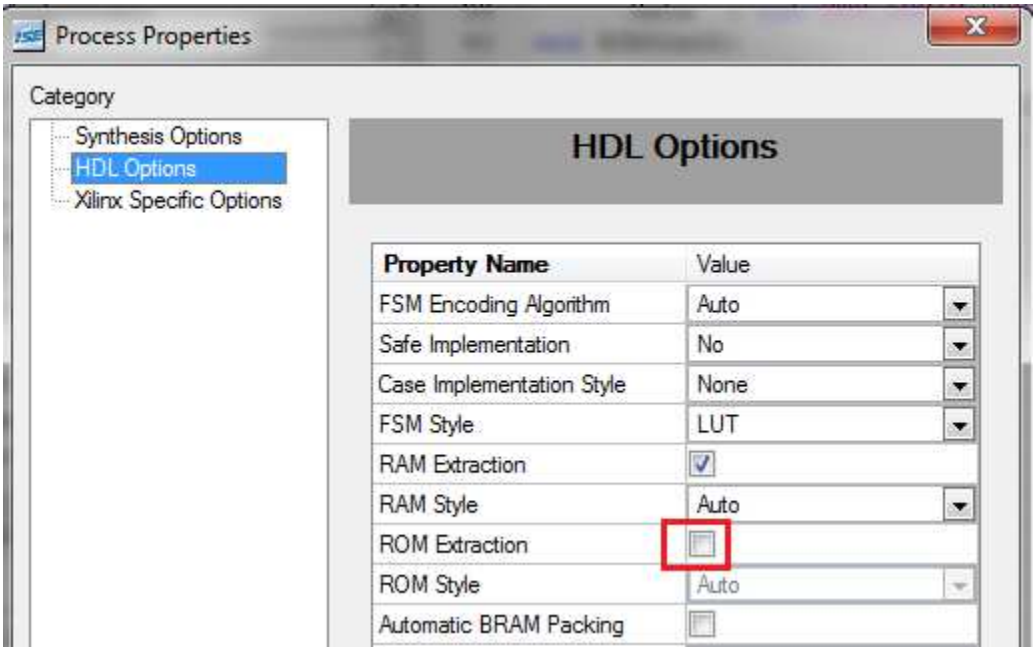

figura 5

În categoria **HDL Option** debifaţi căsuţa de validare **ROM Extraction**, ca în figura 5.

Se va face implementarea şi se va verifica dacă a fost îndeplinită constrângerea de timp aspra perioadei semnalului Clk. Pentru aceasta se vizualizează raportul de plasare şi rutare (**Place and Route Report**). Căutaţi în raport constrângerea asupra lui CLK.

Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
TS_Clk = PERIOD TIMEGRP "Clk" 20 ns HIGH 50%	SETUP	0.001ns	19.999ns	0	0
	HOLD	1.088ns		0	0

figura 6

În coloana **Best Case Achievable** trebuie să obțineți **o valoare mai mică** de 20 ns. În acest caz se spune a fost îndeplinită constrângerea de timp asupra lui Clk.

În cazul în care constrângerea asupra lui CLK nu este îndeplinită, o soluţie posibilă este creşterea efortului de rutare. Pentru aceasta, în fereastra **Processes** faceţi clic dreapta pe **Place & Route** şi în meniul contextual ce va apare selectaţi **Properties…**. Va apare fereastra din figura următoare:

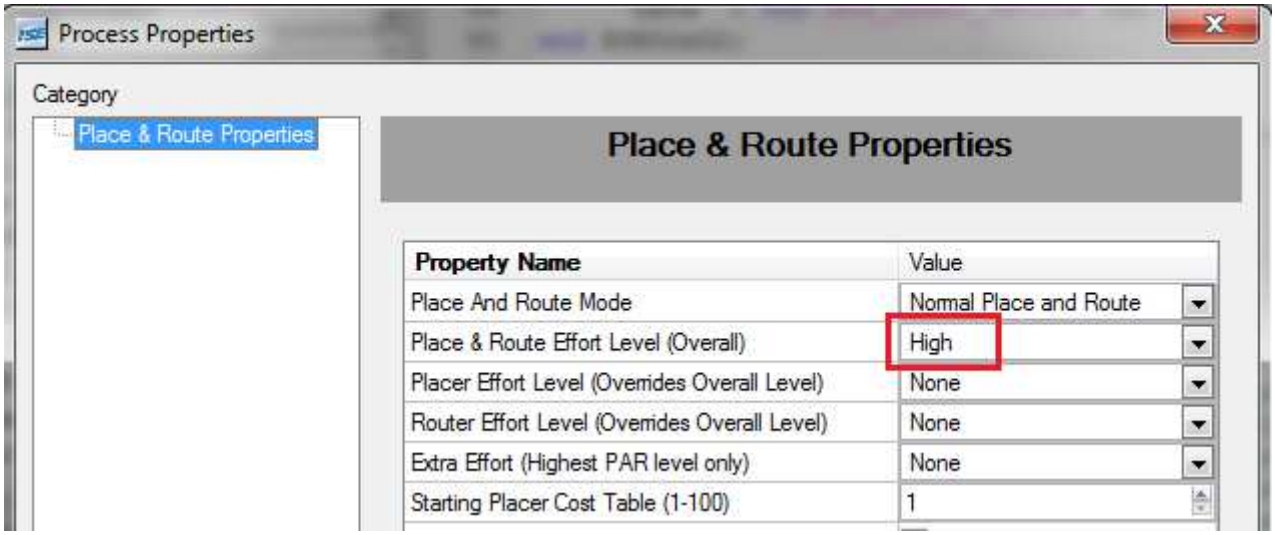

figura 7

Setaţi efortul de plasare şi rutare pe **High** ca în figură şi implementaţi încă o dată.

Dacă nici acum nu este îndeplinită constrângerea asupra lui CLK, chemaţi profesorul!

Generaţi fişierul de programare **.bit**. Nu uitaţi că ceasul de programare este JTAG CLK.

Trimiteţi-l în placa S3 fişierul .bit

Verificaţi dacă funcţionează pentru diferiţi operanzi.

Dacă vi se pare că funcţionează, chemaţi profesorul pentru validare. Profesorul vă va cere să testaţi cu diferite valori pentru a şi b.

În funcţie de procentajul de realizarea a paşilor 3 şi 4 veți fi notați cu note între 5 şi 10.