Organizarea Calculatoarelor
LABORATOR 3 – Implementarea şi simularea unui sumator pe 4 biţi

SCOPUL LUCRĂRII

Pe baza sumatorului elementar proiectat în laboratorul precedent se cere construirea unui sumator cu operanzi reprezentaţi pe 4 biţi. Sumatorul se va simula şi apoi se va implementa folosind placa de dezvoltare S3.

Pentru a putea aduna numere reprezentate pe *n* biţi, sumatoarele pe un bit se conectează în cascadă, ca în figura următoare:

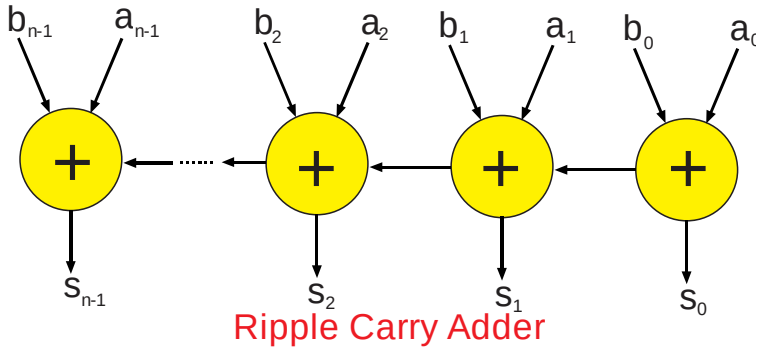

figura 1

În esenţă un astfel de sumator adună biţii asemănător cu modul uman de a face adunarea, cu observaţia că oamenii execută adunarea bit cu bit în **timp**, pe când sumatorul de mai sus face această operaţie în **spaţiu**.

Această soluţie trebuie implementată folosind ISE. Cea mai la îndemână soluţie, bazată pe nivelul dumneavoastră actual de cunoştinţe referitor la implementarea de tip schemă, ar fi desenarea sumatorului elementar de *n* ori urmată de conectarea lui Cout de la rangul *i* la Cin al rangului *i*+1. În figura următoare este prezentată această soluţie pentru *n*=8.

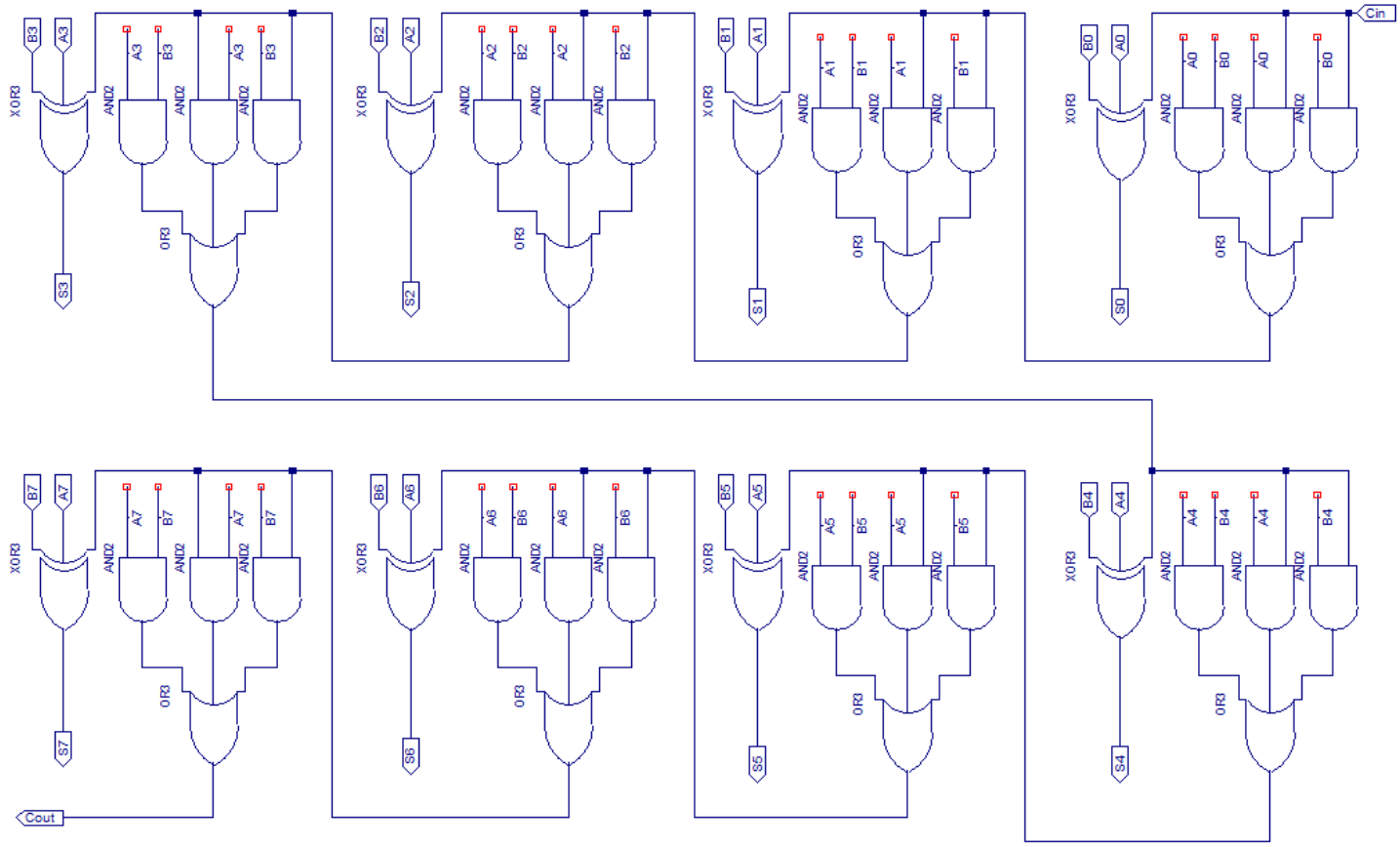

figura 2

Se observă că implementarea acestei soluţii duce la o schemă de mari dimensiuni. Imaginaţi-vă cum ar arăta schema unui sumator pe 32 de biţi! Dacă schema sumatorul pe 32 de biţi este foarte mare cum ar arăta schema unui procesor?

Mărimea enormă a unei astfel de scheme impune o organizare ierarhică a proiectului asemănătoare cu organizarea proiectelor software. De exemplu, în C, întotdeauna va exista o funcţie *main* din care vom apela funcţii, care la rândul lor vor apela alte funcţii ş.a.m.d. Modelul bazat pe apelul de funcţii nu este cea mai bună analogie deoarece pentru o funcţie exista un singur cod maşină. Acest cod este executat indiferent de apel. Diferenţă între două apeluri constă din locul de unde se face apelul şi eventual parametrii apelului. În C însă, în afară de apelul de funcţie, mai există şi apeluri macro. De fiecare dată când se face un apel macro se generează codul macro-ului. Astfel, dacă un macro este apelat de 10 ori, codul din respectivul macro se generează de 10 ori. În cazul schemelor hardware situaţia este asemănătore cu apelul macro.

În acest laborator se va prezenta **organizarea ierarhică** a proiectelor de tip schematic bazată pe module şi se va introduce noţiunea de **magistrală**.

Desfăşurarea lucrării

Pasul 1: Crearea proiectului.

Se face conform metodologiei prezentate în primul laborator. Proiectul se va numi as4. Alegeţi acelaşi FPGA ca în laboratorul 1! Numele as4 vine de la adunare-scădere pe 4 biţi. În acest laborator se va implementa sumatorul urmând ca în laboratorul următor se va adăuga hardware-ul pentru scădere.

Pasul 2: Adăugarea sumatorului pe 1 bit.

Sumatorul elementar se va comporta ca un macro, adică va fi „apelat” de 4 ori în schema sumatorului pe 4 biţi. „**Apelat**” este un termen folosit la software. În proiectele hardware termenul folosit este „**instanţiat**”. În continuare vom adopta strategia de proiectare de jos în sus. Mai întâi vom desena sumatorul elementar, apoi vom conecta în cascadă 4 sumatoare elementare pentru a construi un sumator pe 4 biţi şi în final sumatorul pe 4 biţi va fi folosit pentru a proiecta un sumator/scăzător (evident pe 4 biţi).

Mai întâi ar trebui desenat sumatorul elementar. Deoarece această operaţie a fost făcută în laboratorul precedent vom folosi schema sum1g.sch, care deja există. Se poate copia acest fişier din folderul proiectului sum1g în folderul acestui proiect şi apoi se poate adăuga cu Add source… Operaţia de copiere manuală se poate evita prin folosirea opţiunii Add Copy of Source… Pentru această faceţi clic dreapta în fereastra Sources şi din meniul contextual ce va apare alegeţi Add Copy of Source…, ca în figura 3.

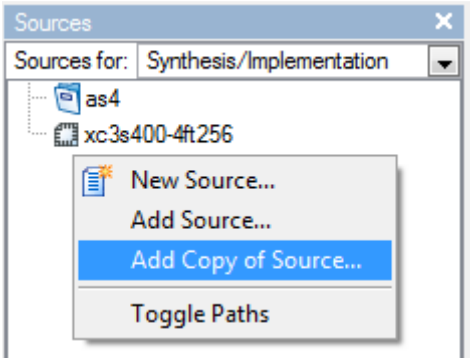

figura 3

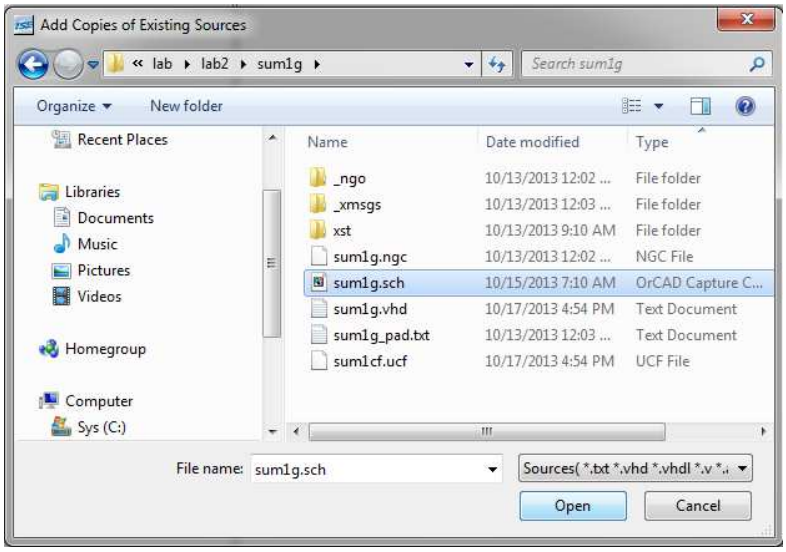

figura 4

Ca urmare va apare fereastra **Add Copies of Existing Sources**. Navigați până la folderul în care se află sum1g.sch, selectaţi-l şi apoi apăsaţi **Open**, ca în figura 4.

Apoi va apare o **fereastră de confirmare**. Apăsați OK şi treceţi mai departe.

Se adaugă la proiect un nou fişier de tip schematic cu numele sum4.sch. Acesta va fi folosit pentru desenarea sumatorului pe 4 biţi, bazat pe 4 sumatoare elementare. (**New source → Schematic**)

Următoarea operaţie constă în **crearea unui simbol grafic** prin care se va reprezenta sumatorul elementar la nivele ierarhice superioare. În acest scop în fereastra **Sources** se selectează **sum1g** iar în fereastra de procese asociate se deschide **Design Utilities** şi apoi se face dublu clic stânga pe **Create Schematic Symbol**, ca în figura 5.

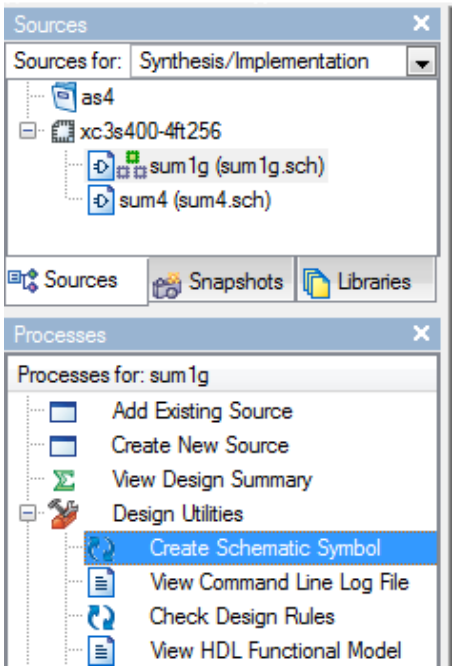

figura 5

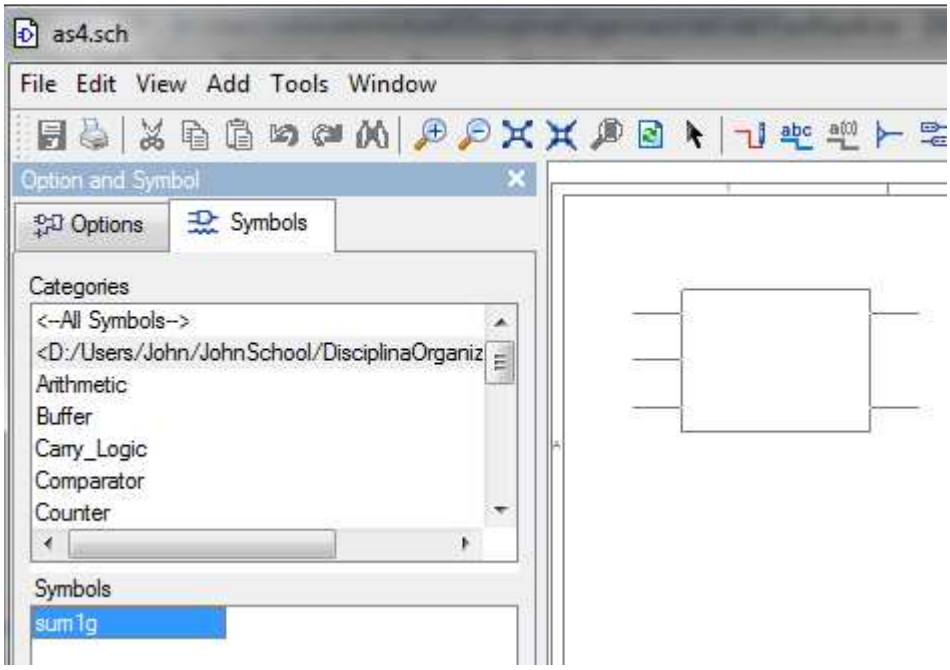

figura 6

Simbolul creat pentru sum1g se va folosi pe planşa sum4.sch. Deschideţi pentru editare schema sum4.sch şi se selectaţi câmpul Tab Symbols. Se remarcă apariţia unei noi opţiuni şi anume *<working_folder/nume proiect>*. Simbolul din această locaţie este **sum1g** (figura 6). Se plasează acest simbol pe suprafaţa de desenare la fel cum au fost plasate porţile î în laboratorul precedent.

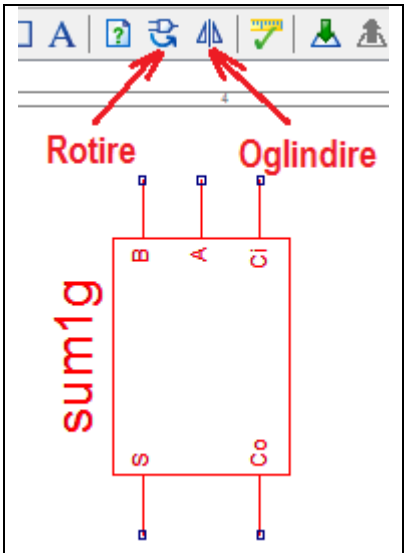

figura 7

Observaţie: dacă opţiunea *<working_folder/nume proiect>* nu apare, închideţi sum4.sch şi apoi redeschideţi-o.

Pentru a poziţiona optim simbolul sum1g se folosesc butoanele de rotire şi oglindire din figura 7. După o rotire şi o oglindire, poziţia simbolului ar trebuie să fie cea din figură**. Este posibil ca să nu obţineţi exact situaţia din figură** deoarece poziţia terminalelor A, B, Ci, S şi Co depinde de ordinea în care a fost desenată sum1g.

In continuare se va explica tehnica de rearanjarea terminalelor în cazul în care se pleacă de la situaţia din figura 7. Această tehnică este generală şi poate fi aplicată în orice situaţie.

Pentru a putea folosi simbolul sum1g la desenarea sumatorului pe 4 biţi, ar trebui ca poziţiile pinilor S şi Co să fie inversate. Desenarea se poate face şi fără această inversare, dar legătura Co – Ci ar intersecta conexiunea S – marcher IO.

Pentru a inversa poziţia lui S cu cea a lui Co trebuie modificat simbolul sum1g. În acest scop se procedează după cum urmează:

1. Se face clic dreapta pe simbolul sum1g. Din meniul contextual apărut se selectează Symbol iar apoi, din noul meniu se face clic pe Edit symbol.
2. Se selectează ansamblul S – linie – pin, prin desenarea unui dreptunghi în jurul acestora (cu butonul stânga al mouse-ului apăsat), ca în figura 8a. **Atenţie: întotdeauna selectaţi numele, linia şi pinul împreuna, niciodată separat!**
3. Se mută ansamblul S – linie – pin, în poziţia din figura 8b.
4. Se selectează ansamblul Co – linie – pin, ca în figura 8c.

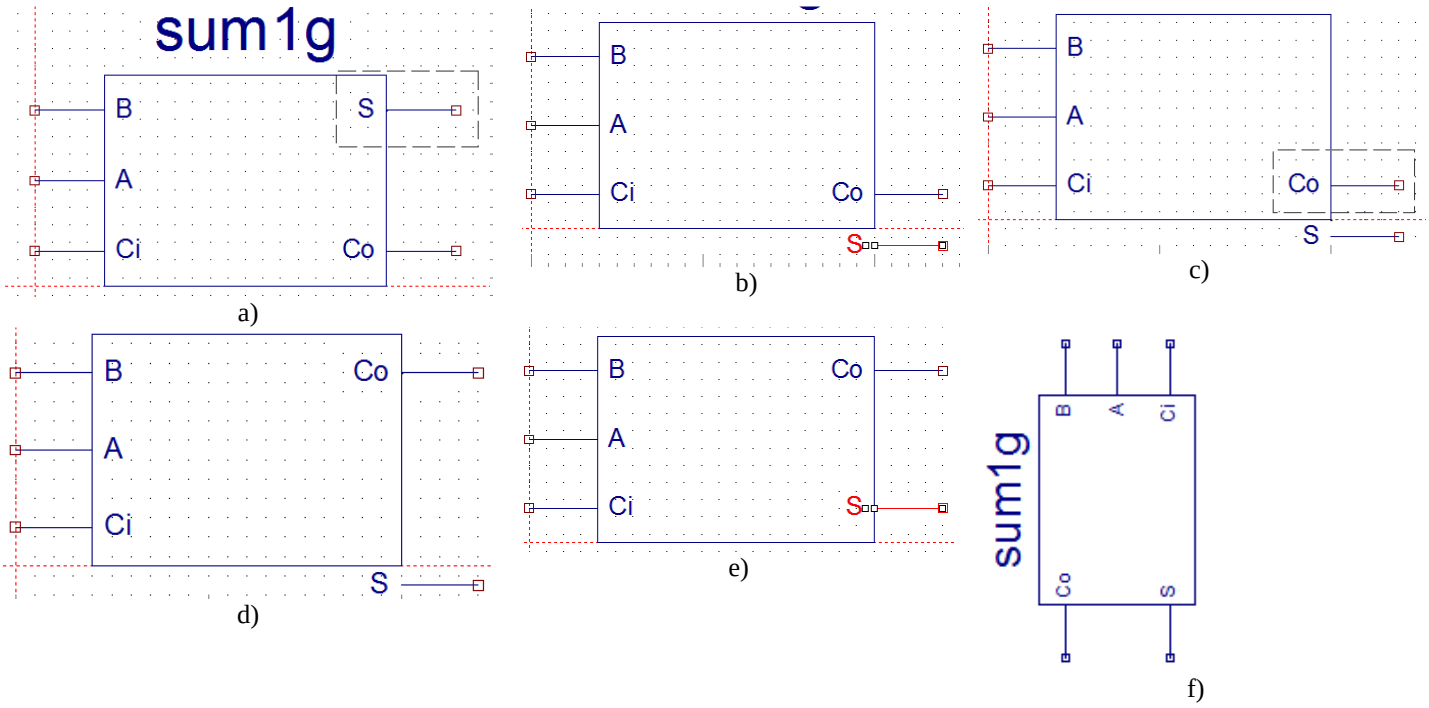

figura 8

5. Se mută ansamblul Co – linie – pin, în poziţia din figura 8d.
6. Se selectează şi apoi se mută ansamblul S – linie – pin în poziţia din figura 8e.
7. Se verifică validitatea simbolului cu Check Schematic( ), se salvează iar apoi se închide fereastra de editare.

Deoarece simbolul sum1g a fost modificat, când se revine la editarea schemei sum4 va apare o fereastra de dialog în care se precizează că simbolul sim1g a fost modificat. Se apasă butonul **Update** iar apoi OK. **Dacă nu apare fereastra de update, închideţi sum4.sch şi apoi redeschideţi-o.** Dacă toate operaţiile au fost executate corect, se va obţine o nouă formă a simbolului sum1g, ca în figura 8f.

Simbolul sumatorului pe un bit din figura 8f se va folosi pentru a desena un sumator pe patru biţi. Sumatorul pe 4 biţi va efectua operația:

r=**a**+**b**, **a**=$a_3a_2a_1a_0$, **b**=$b_3b_2b_1b_0$, **r**=$r_3r_2r_1r_0$.

De asemenea sumatorul va calcula şi transportul de ieşire de la rangul 3, numit în continuare C4. O posibilă schema a acestui sumator este prezentată în figura 9, partea stângă. **În figura pentru fiecare semnal de intrare sau de ieşire s-a folosit un marcher IO.**

Deoarece scopul final este să implementăm un procesor, este de aşteptat ca sumatorul pe care-l vom proiecta să fie folosit în schema procesorului tot prin intermediul unui simbol. Dacă pentru fiecare semnal de

intrare sau de ieşire s-ar folosi un marcher IO, simbolul pentru sumatorul pe 4 biţi ar fi cel din figura 9, partea dreaptă.

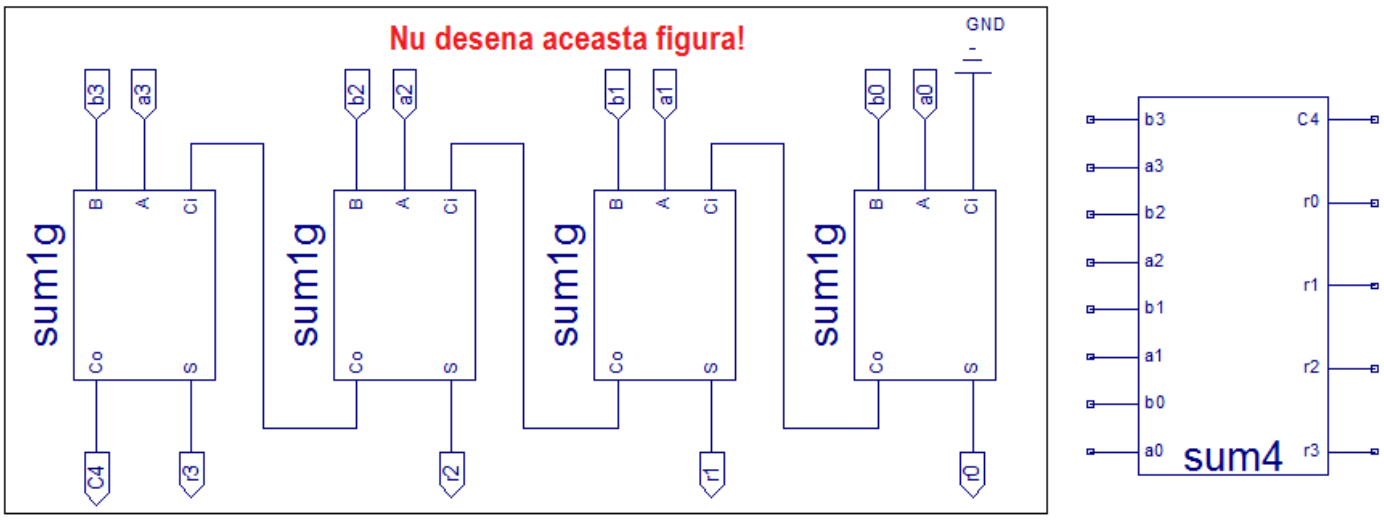

figura 9

Dacă simbolul unui sumator pe 4 biţi are 13 terminale, atunci simbolul asociat unui sumator pe 32 de biţi, cât este lăţimea căii de date la MIPS, ar avea 32 de terminale pentru operandul A, 32 pentru B şi încă 32 pentru S. În total 96 de terminale! Este clar că un asemenea simbol este prea mare putea a putea desena cu el scheme inteligibile.

Soluţia este conectarea prin magistrale. În general conectarea elementelor unei scheme se face prin fire, prin nume şi prin grupe de fire. Grupele de fire se numesc magistrale.

În cazul în care prelucrările se fac pe un număr mare de biţi conectarea prin magistrale micşorează numărul de conexiuni.

Magistralele sunt pentru hardware ce sunt vectorii unidimensionali pentru programare!

Pasul 3: Crearea schemei sumatorului pe 4 biţi folosind magistrale.

Deschideţi **sum4.sch** în ISE. Plasaţi 4 sumatoare de 1 bit şi conectaţi-le ca în figura 10.

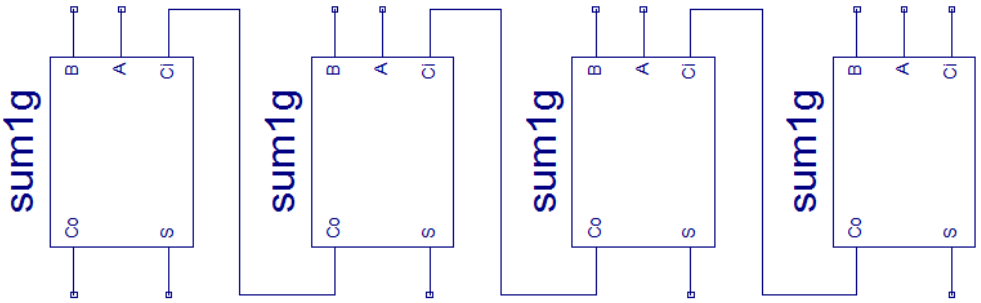

figura 10

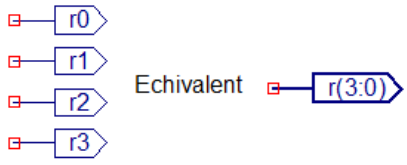

figura 11

NU desenaţi această figură!

În cazul sumatorului pe 4 biţi din figura 9 suma devenea disponibilă în exterior prin intermediul a 4 marcherii IO numiţi r0, r1, r2 şi r3. Cei patru marcheri pot fi înlocuiţi cu un singur marcher IO de tip bus, şi anume **r(3:0)**, ca în figura 11. **NU desenaţi figura 11!**

Pentru desenarea schemei sumatorului cu magistrale procedaţi după cum urmează:

1. Adăugaţi o conexiune care se termină în aer la ambele capete iar apoi la capătul din dreapta adăugaţi un marcher IO de tip output, ca în figura 12.

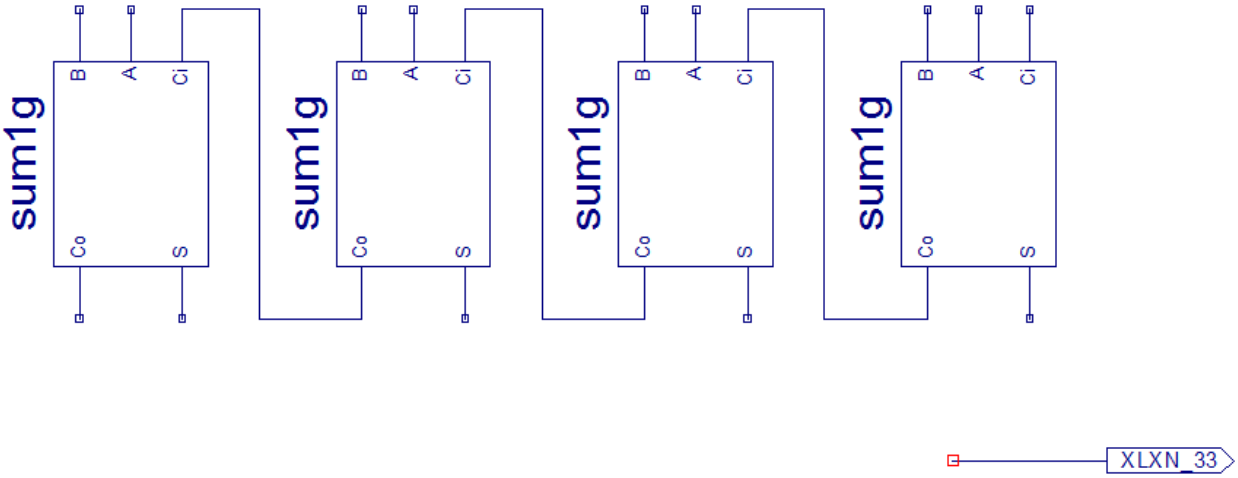

figura 12

2. Schimbaţi numele marcherului IO asignat automat din XLXN_33 în **r(3:0).** Veţi obţine situaţia din figura 13. **Atenţie**: în schema dumneavoastră numele iniţial al marcherului IO poate fi altul.

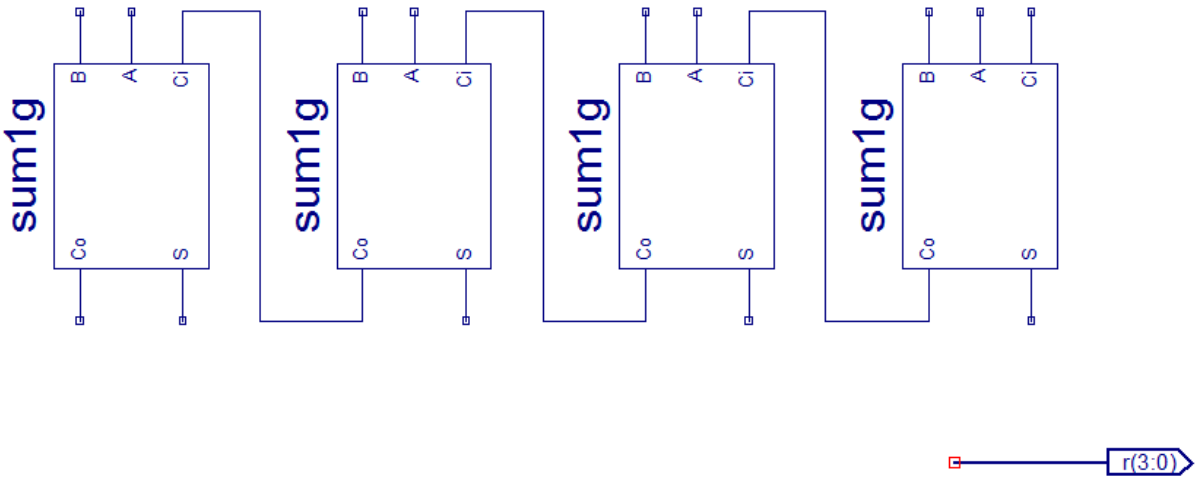

figura 13

Observaţi ca firul ataşat de marcherul IO şi-a schimbat grosimea deoarece este ataşat de un marcher de magistrala. Conexiunea (firul) a devenit magistrală! Dacă faceţi dublu clic magistrala nou apărută, veţi constata că numele acesteia este r(3:0). Toate mediile de desenare, şi ISE nu face excepţie, consideră magistrala o grupare de conexiuni singulare. Relaţia dintre magistrală şi conexiunile singulare definite implicit odată cu definirea magistralei se aseamănă cu definirea unui vector. De exemplu, în C char r[4]; a definit un vector de 4 elemente: r[0], r[1], r[2], r[3]. În ISE magistrala r(3:0) şi defineşte 4 fire numite r(0), r(1), r(2) şi r(3).

3. Deoarece magistrala nu este altceva decât o mulţime de fire, se poate folosi conectarea prin nume explicată în laboratorul 2. Astfel specificarea legăturilor dintre marcherul IO r(3:0) şi ieşirile S ale sumatoarelor de 1 bit s-ar putea face ca în figura 14. **NU desenaţi această figură!**

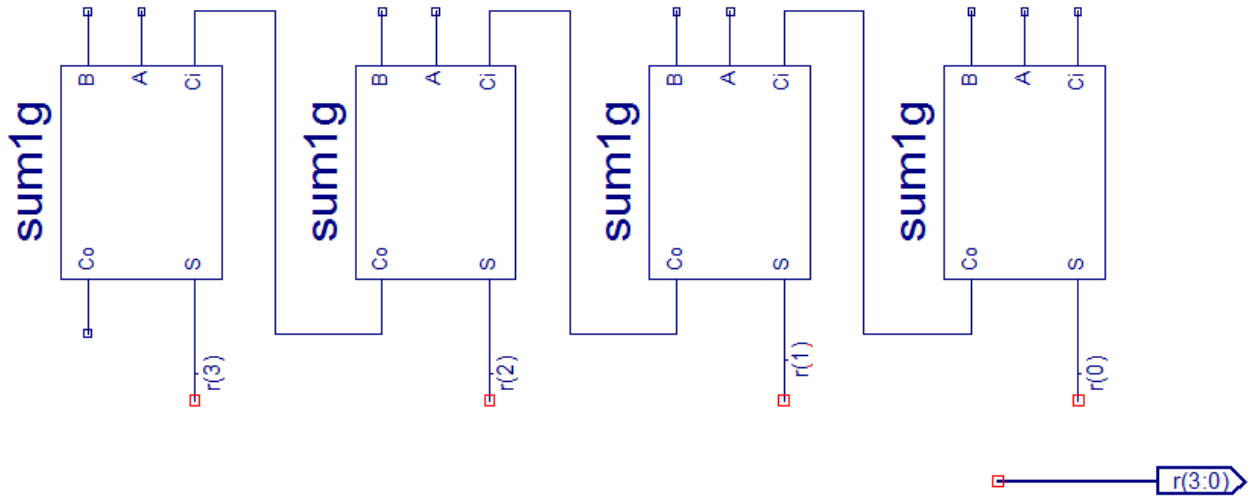

figura 14

Deşi această schema este corectă, are un dezavantaj: modul de conectare marcheri-terminale nu este bine evidenţiat. In extremis, se pot construi scheme în care conexiunile să se facă numai prin nume. Astfel de scheme nu îşi relevă aproape de loc funcţionalitatea.

La cealaltă extrema se pot desena scheme în care conectivitatea să se facă numai cu fire. Şi aceste scheme sunt aproape imposibil de interpretat din cauza numărului mare de fire. De aceea într-o schema trebuie păstrată o balanţă între conectarea prin nume şi conectarea prin fir astfel încât din schemă să rezulte cât mai clar funcţia îndeplinită. În cazul magistralelor, există posibilitatea evidenţierii conectivităţii dacă se procedează după cum urmează.

4. Se prelungeşte magistrala r(3:0) ca în figura 15.

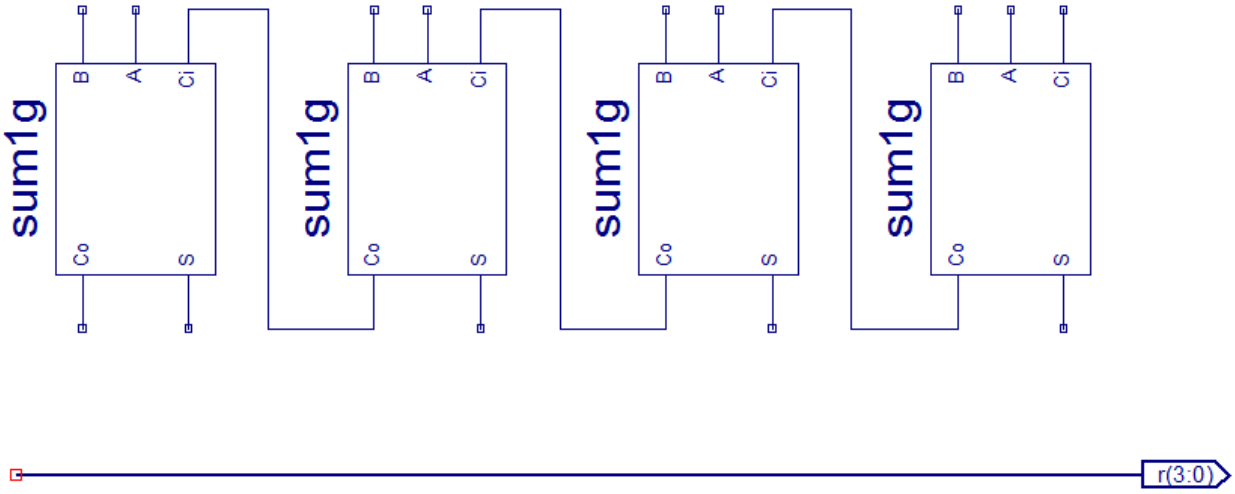

figura 15

5. Se apăsă butonul de desenare fire, se selectează câmpul tab **Options** (daca nu este deja selectat) şi se validează check box-ul „**Automatically**....”, ca în figura 16.

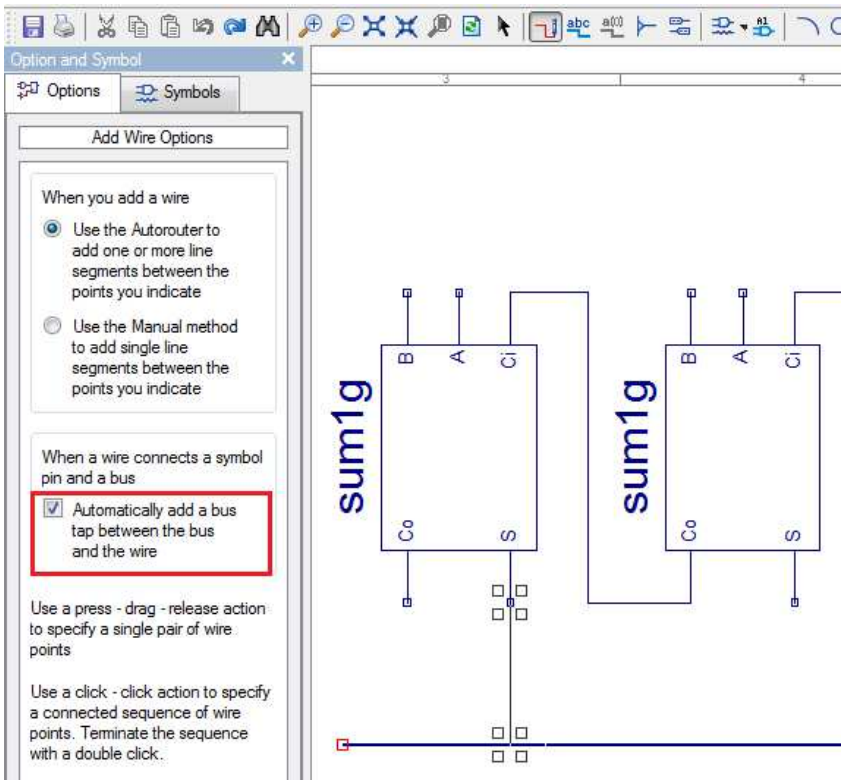

figura 16

6. Pentru a adăuga o conexiune terminal – magistrală, poziţionaţi cursorul în dreptul unui terminal – de exemplu S-ul rangului 3 în figura 16. Poziţionarea este corectă în momentul în care apar cele 4 pătrăţele. Faceţi clic. Apoi, poziţionaţi cursorul pe magistrală. Şi în acest caz, poziţionarea este corectă în momentul în care apar cele 4 pătrăţele. Atenţie: de obicei aceste ultime 4 pătrăţele nu apar, chiar dar poziţionarea este corectă! Faceţi clic. Faptul că un fir este conectat la magistrală este specificat prin apariţia triunghiului isoscel (tap) cu baza aşezată pe magistrală.

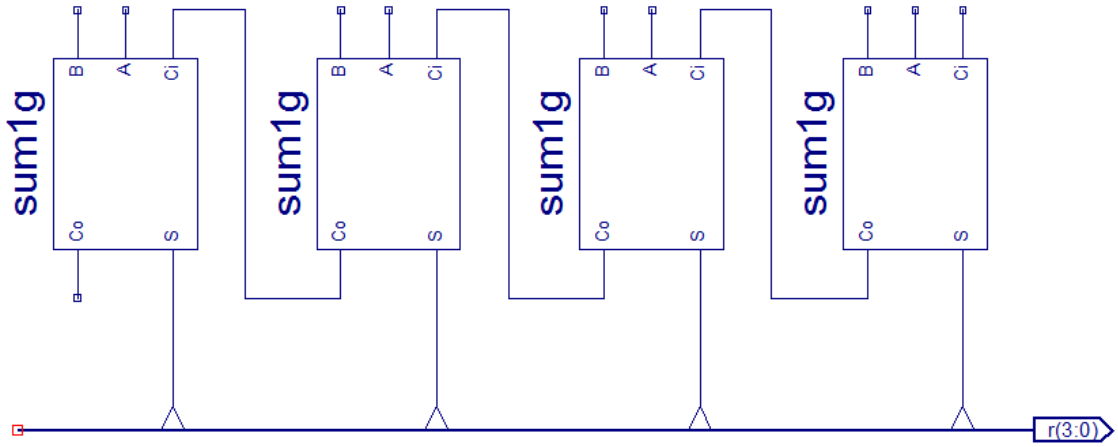

figura 17

7. Conexiunile **magistrală – pin modul** şi **magistrală – marcher IO** se supun unor restricţii.

   Mai întâi restricţiile la conexiunea **magistrală – pin modul**:

   - Magistrala trebuie să aibă **aceeaşi dimensiune** ca terminalul de bus al unui modul la care se conectează.
   - Când un fir se conectează la o magistrală (prin intermediul unui tap sau prin nume) **numele firului trebuie să fie identic cu numele magistralei** iar **indexul firului trebuie să fie între indecşi magistralei**.
   - Când o submagistrală se conectează la o magistrală, numele submagistralei trebuie să fie identic cu numele magistralei iar indecşi submagistralei trebuie să fie un subset al indecşilor magistralei.

Restricţiile la conexiunea **magistrală – marcher IO** :

- Bus-ul conectat la un marcher IO de bus are acelaşi nume ca marcherul.
- Magistrala trebuie să aibă aceeaşi dimensiune marcherul IO la care se conectează.

Realizaţi toate conexiunile aferente lui r(3:0) ca în figura 17.

Atenţie: în momentul în care aţi adăugat conexiuni pin modul – magistrală, aceste conexiuni au primit automat un nume dat de ISE. Va apare o avertizare că respectiva conexiune are un nume temporar, nume care trebuie schimbat deoarece nu respectă regulile enunţate anterior. Dacă faceţi Check Schematic înainte de modificarea numelor veţi obţine o mulţime de erori!

Pentru a schimba numele conexiunilor adăugate apăsaţi butonul Add net name, setaţi Increase the Name şi porniţi cu r(0) conform marcajelor cu dreptunghi roşu din figura 18. Numele se schimbă conform metodei prezentate în laboratorul 2. După ce schimbaţi toate numele trebuie să obţineţi situaţia din figura 18.

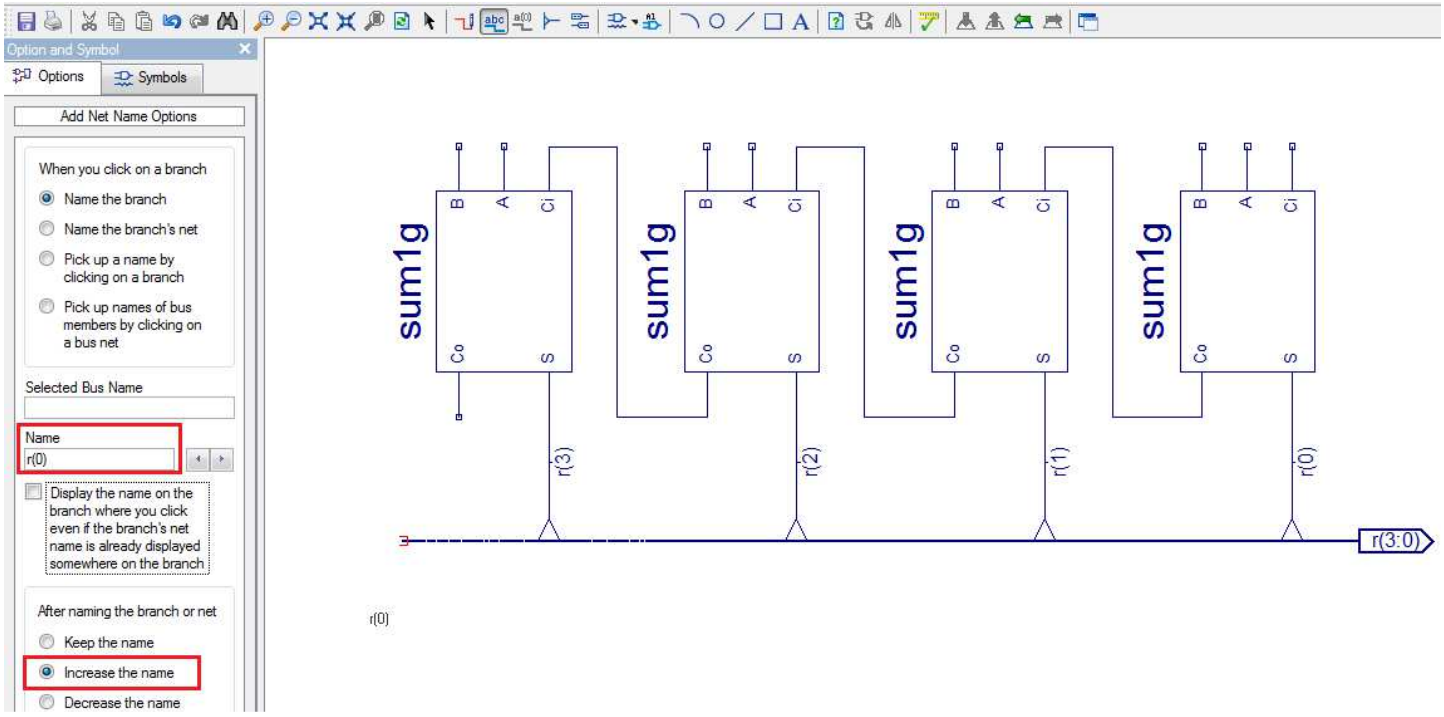

figura 18

Înainte de a trece la desenarea figurii se vor face câteva setări necesare pentru desenarea rapidă şi corectă a schemei. **Atenţie:** Aveţi grijă ca **pinii componentelor** sau **tap-urile unei magistrale** să nu fie poziţionate pe un fir sau pe o magistrală.

Pentru a vedea mai uşor dacă pinii sau tap-urile se poziţionează pe vreo magistrală mai întâi schimbaţi paleta de culori a editorului de scheme. Din meniul **Edit** al ferestrei principale ISE (NU a editorului de scheme) selectaţi **Preference**, **Schematic Editor**, **Colors**, **White Background**, ca în figura 19.

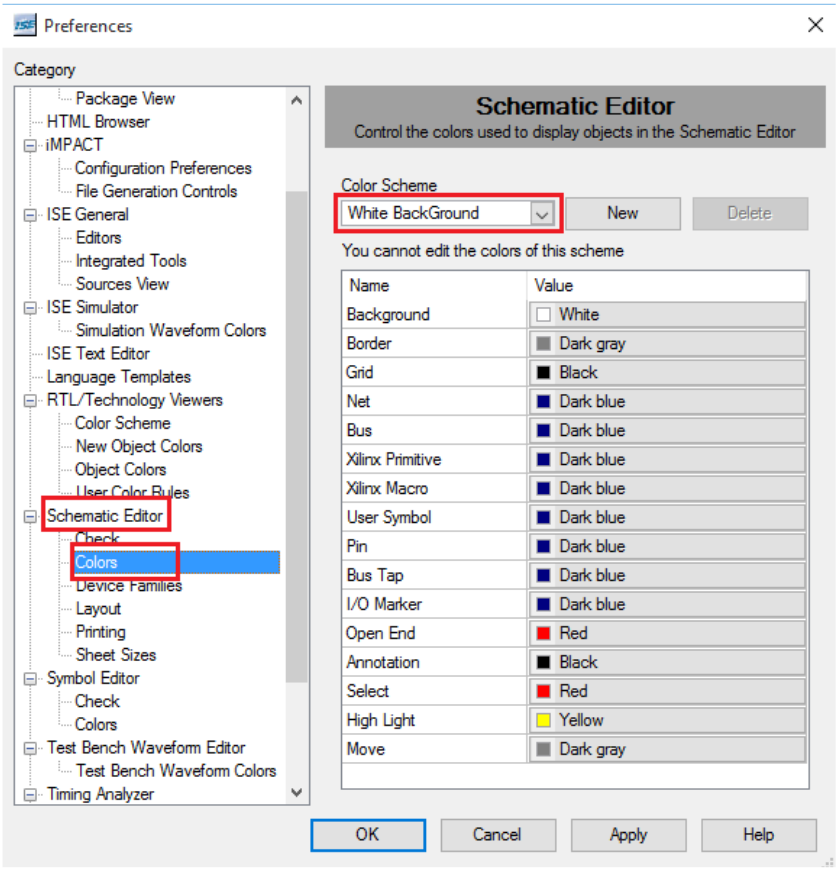

figura 19

Apoi apăsaţi butonul **New** şi vi se va propune numele **CustomScheme** pentru noua schemă de culoare. Acceptaţi iar apoi schimbaţi în **verde** culoarea pentru **Bus** şi **Net**. Cel mai bun verde este cel cu R=0, G=200, B=0. Definiţi acest verde.

De asemenea este util să **nu mai afişaţi gridul de desenare**. Pentru aceasta selectaţi **Preference**, **Schematic Editor, Layout** în şi debifaţi Check Box-ul **Display Grid**.

În continuare adăugaţi marcherii **a** şi **b** plus magistralele aferente şi marcherul **Co** conform schemei din figura 20. Schimbați tipul marcherilor **a** şi **b** din output în input. **Pentru a schimba tipul unui marcher** faceţi dublu clic stânga pe respectivul marcher.

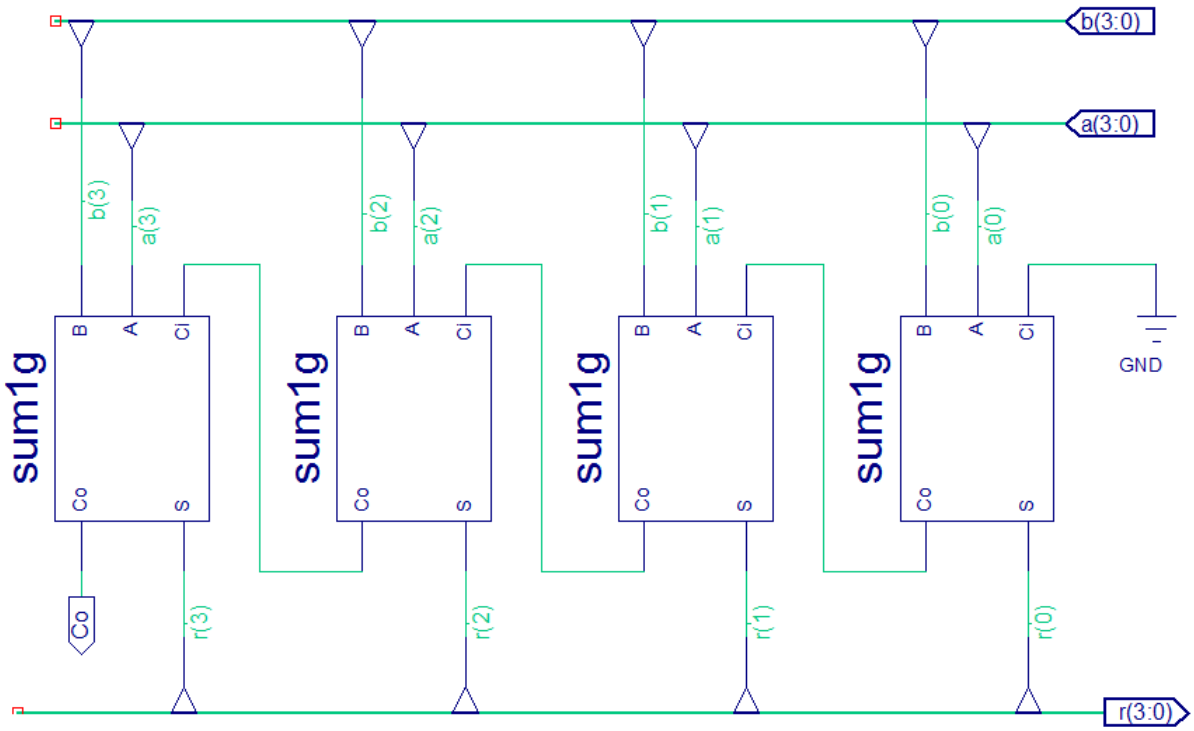

figura 20

Ci de la primul sumator se conectează la ‚0’-logic deoarece este mai uşor de testat pe placa S3. Pentru conectarea la ‚0’-logic selectaţi simbolul **GND** în fereastra **Symbols**.

Încercaţi să desenaţi o schemă cât mai apropiată de cea din figura 20. După ce terminaţi de desenat salvaţi şi verificaţi corectitudinea cu Check Schematic.

Pasul 4: Crearea fişierului de constrângeri şi specificarea acestora.

Cele două numere de 4 biţi, **a**=$a_3a_2a_1a_0$ şi **b**= $b_3b_2b_1b_0$, se introduc cu ajutorul comutatoarelor glisante de pe placa S3 iar rezultatul se afişează prin intermediul LED-urilor.

Alocarea comutatoarelor şi a LED-urilor din placa S3 se face conform tabelului următor.

Top IO Marcher	**Resursă S3**	**Pin FPGA**	**Top IO Marcher**	**Resursă S3**	**Pin FPGA**	**Top IO Marcher**	**Resursă S3**	**Pin FPGA**
a(0)	SW0	…	b(0)	SW4	…	r(0)	LD0	…
a(1)	SW1	…	b(1)	SW5	…	r(1)	LD1	…
a(2)	SW2	…	b(2)	SW6	…	r(2)	LD2	…
a(3)	SW3	…	b(3)	SW7	…	r(3)	LD3	…
						Co	LD4	…

1. Creaţi fişierul de constrângeri conform procedurii din al doilea laborator. Pentru a afla pinii corespunzători acestor resurse, consultă documentaţia S3board, pagina 21 sau placa S3.
2. Se generează fişierul de configurare conform indicaţiilor din primul laborator, se conectează cablul de date şi **înainte de a cupla alimentarea chemaţi profesorul.**
3. Trimiteţi fişierul de configurare în placa de S3 conform indicaţiilor din primul laborator.
4. După configurare, faceţi diferite combinaţii pentru **a** şi **b**. Observaţi rezultatul.

Chemaţi profesorul pentru validare. Dacă ați ajuns aici aveţi nota 5.

Pasul 5: 5 combinaţii.

Pentru nota mai mare ca 5.

Profesorul vă va cere să introduceți operanzii pentru 5 adunări. Operanzii vor fi specificaţi în **hexazecimal**. Studenții vor introduce acești operanzi prin intermediul comutatoarelor glisante, vor nota starea LED-urilor şi vor enunţă rezultatul adunării în **hexazecimal**. Aveţi 20 secunde pentru o adunare.

Antrenaţi-vă! Când sunteţi gata, **chemaţi profesorul pentru verificare**. Veţi obţine între 0 şi 5 puncte în funcție de numărul de adunări pe care le-aţi efectuat corect.