LUCRARE LABORATOR NR. 6
INTRODUCERE IN UTILIZAREA LIMBAJULUI VHDL
COMPILAREA SI SIMULAREA UNUI DESIGN

1. Scopul lucrării

Lucrarea de laborator are ca scop familiarizarea cu utilizarea sistemului VHDL şi cu programul *ModelSim* (noţiuni generale, biblioteci, editarea, compilarea şi simularea unui proiect, vizualizarea şi examinarea formelor de undă, noţiuni de limbaj).

2. Particularităţi ale VHDL (*VHSIC Hardware Description Language*)

Limbajele care descriu sistemele electronice digitale poartă denumirea de limbaje de descriere hardware (*hardware description language*). În prezent cele mai cunoscute şi utilizate limbaje de descriere hardware sunt VHDL şi Verilog, acestea fiind şi standarde IEEE (*Institute of Electrical and Electronics Engineers*).

VHDL este un limbaj de programare de nivel înalt orientat obiect care permite descrierea şi modelarea circuitelor electronice digitale şi nu numai.

Una din deosebirile fundamentale între VHDL ca limbaj de descriere a *hardware-ului* şi limbajele de programare clasice (Pascal, C, C++) constă în semnificaţia codului scris.

Astfel, în timp ce codul într-un limbaj de programare gen C defineşte un program şi este alcătuit în întregime din instrucţiuni care se execută secvenţial (în ordinea în care sunt scrise), codul VHDL are caracter concurent şi reprezintă descrierea (comportamentală, structurală, mixtă) a unui sistem sau dispozitiv fizic. Din acest motiv, în VHDL se recomandă să existe o apreciere cantitativă şi calitativă a codului scris pentru a obţine o modelare corectă a sistemului propus.

În VHDL, pentru a modela sau simula un circuit digital utilizatorul creează **unităţi de proiect** (*design units*). Limbajul VHDL pune la dispoziţie cinci tipuri de unităţi de proiect:

- Declaraţia de entitate (*entity*)
- Arhitectura unei entităţi (*architecture*)
- Declaraţia de configuraţie (*configuration*)
- Declaraţia de pachet (*package*)
- Corpul pachetului (*package body*)

Codul VHDL al unităţilor de proiect definite de utilizator se scrie în cadrul unor fişiere sursă având extensia *.vhd. În cadrul unui fişier sursă .vhd pot exista mai multe unităţi de proiect. Totuşi este recomandat ca fiecare fişier sursă să conţină doar câte o pereche entitate-arhitectură, o configuraţie sau un package împreună cu corpul package-ului, dacă e cazul.

Structura de date utilizată de un sistem VHDL este diferită de cea folosită de un sistem de operare şi este orientată spre conceptul de bază de date. Astfel, pe lîngă noţiunea de director, apare şi noţiunea de bibliotecă.

Bibliotecile conţin rezultatul compilării fişierelor sursă. Aceste rezultate din biblioteci sunt folosite de simulatorul VHDL pentru simularea unei entităţi sau a unei configuraţii.

La laborator, pentru editarea şi compilarea fişierelor sursă, precum şi pentru simularea proiectelor şi vizualizarea rezultatelor va fi folosit programul *ModelSim* al firmei Mentor Graphics (www.model.com). Pentru sistemul de operare Windows programul *ModelSim* este furnizat de producător sub 3 variante: *ModelSim SE*, *ModelSim PE* şi *ModelSim PE Student*

Edition. Primele două variante sunt destinate utilizatorilor din industrie. Varianta *ModelSim PE Student Edition* are licenţa gratuită şi este destinată utilizării în mediul academic.

Simulatorul *ModelSim* suportă atât standardul VHDL-IEEE 1076 actualizat în 1993, precum şi standardul Verilog-IEEE 1364-1995, putând fi astfel folosit pentru lucrul cu ambele limbaje de descriere hardware. Ultimele versiuni ale ModelSim (ver. 6.2) suportă, de asemenea, limbajele *SystemVerilog* şi *SystemC*. *SystemVerilog* este o extensie a limbajului *Verilog* standard în timp ce *SystemC* (devenit standard IEEE în 2005) este un limbaj de descriere care, pe baza unui set de biblioteci de rutine şi macrouri în limbajul C++, permite dezvoltarea de modele la nivel de sistem descrise în C++ .

Pentru o bună desfăşurare a aplicaţiilor în VHDL în cadrul laboratorului se vor respecta cu stricteţe regulile expuse mai jos!

Programul *ModelSim* este instalat în directorul C:\Modeltech\. Acest director, care conţine atât programul *ModelSim* cât şi sistemul VHDL împreună cu setul său de biblioteci nu trebuie modificat sub nici un motiv. Setul de biblioteci VHDL conţine, printre altele, declaraţii ale tipurilor de date predefinite, ale operatorilor si subrutinelor precum şi declaraţii ale tipurilor de date standardizate IEEE. Fişierele sursă .vhd corespunzătoare declaraţiilor menţionate mai sus **pot fi doar citite** pentru a facilita înţelegerea şi utilizarea corectă a tipurilor de date precum şi a funcţiilor şi procedurilor existente. Toate aceste fişiere sursă .vhd sunt dispuse în diverse subdirectoare în directorul `vhdl_src`

```
C:\Modeltech\vhdl_src\
```

De exemplu, fişierul `standard.vhd` care conţine package-ul cu numele `standard` în care sunt declarate tipurile de date predefinite ale limbajului VHDL (bit, bit_vector, boolean, integer, time, etc) se află în directorul:

```
C:\Modeltech\vhdl_src\std\
```

De asemenea, fişierul sursă `stdlogic.vhd` care conţine package-ul `std_logic_1164` standardizat IEEE se află în directorul:

```
C:\Modeltech\vhdl_src\ieee\
```

Fiecare student va crea, edita sau compila fişiere sursă .vhd **numai** în directorul de lucru. **Directorul de lucru** al fiecărui student are calea şi numele de forma:

```
D:\PAC2\G54ggs\VHDLgs\
```

unde `g` – cifra care defineşte grupa, `s` – semigrupa (A sau B)

De exemplu, studenţii din grupa 5403A vor utiliza următorul director de lucru:

```
D:\PAC2\G5403A\VHDL3A\
```

3. Despre biblioteci (*libraries*) in VHDL

În VHDL există două tipuri de biblioteci: biblioteci de lucru (*working libraries*) şi biblioteci de resurse (*resource libraries*). Biblioteca de lucru este biblioteca în care sînt plasate unităţile de proiect după compilare. O bibliotecă de resurse conţine unităţi de proiect care pot fi instanţiate in designul care se compilează. La un moment dat poate exista doar o singură bibliotecă de lucru (implicit se consideră biblioteca **work**), în schimb, pot exista mai multe biblioteci de resurse (chiar şi biblioteca de lucru **work**) în timpul compilării unui design.

Biblioteca **work** în VHDL prezintă proprietăţi speciale: este predefinită în interiorul compilatorului şi nu trebuie declarată explicit. De asemenea, este biblioteca utilizată de simulator în mod implicit pentru destinaţia unităţilor de proiect compilate. Implementarea bibliotecilor nu este definită în limbajul VHDL. In interiorul *ModelSim* bibliotecile sînt implementate ca directoare şi pot avea orice nume permis de sistemul de operare.

4. Etapele simulării cu *ModelSim*

Simularea cu ModelSim a unui proiect descris în VHDL presupune parcurgerea următoarelor etape:

- pornirea programului *Modelsim* şi selectarea directorului de lucru;
- crearea bibliotecii *work* în directorul de lucru selectat şi maparea ei cu biblioteca fizică ;
- crearea şi editarea fişierelor sursă în cod VHDL;
- compilarea fişierelor sursă;
- simularea propriu-zisă;
- vizualizarea rezultatelor.

Pentru a atinge scopul declarat al lucrării, în continuare se va descrie şi se va simula cu *ModelSim* un sumator complet de 1 bit prevăzut cu intrare şi ieşire de împrumut şi transport.

4.1. Pornirea programului *ModelSim* şi selectarea directorului de lucru

În sistemul de operare Windows 2000/XP porniți programul ModelSim acționând dublu-click pe icon-ul de pe desktop  sau

Start > Programs > ModeSim … > ModeSim

Apare fereastra principală a programului care arată similar cu cea din Figura 1:

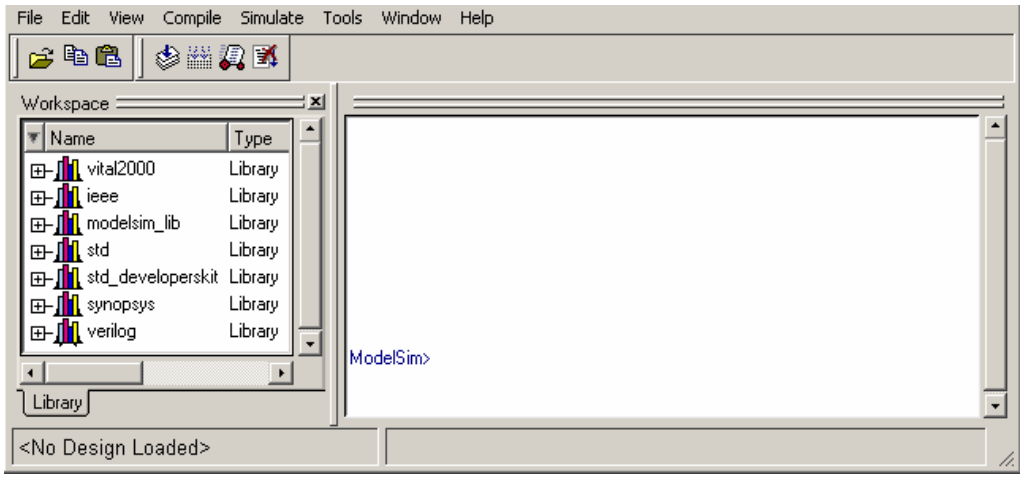


Figura 1 – Fereastra principală a programului *ModelSim*

În fereastra principală a programului *ModelSim* se disting două zone:

– zona din stânga numită *Workspace* care conține structura de biblioteci a mediului VHDL. Se pot observa printre altele bibliotecile `std` şi `ieee`. Pentru a vedea conținutul unei biblioteci se selectează semnul + din dreptul acesteia.

– zona din dreapta este cea în care se introduc la linia de comandă (prompter) `ModelSim>` comenzile ce se doresc a fi executate.

Observație: Majoritatea comenzilor pot fi selectate din meniul ferestrelor programului *ModelSim*. În acest caz la prompter apare în mod automat comanda selectată împreună cu atributele sau parametrii corespunzători.

Selectarea directorului de lucru

În prealabil trebuie să ne asigurăm că directorul de lucru există. Dacă nu există atunci acesta trebuie creat folosind, de exemplu, o fereastră Windows Explorer. Aşa cum s-a

menționat anterior, directorul de lucru este unic pentru fiecare semigrupă, de exemplu,

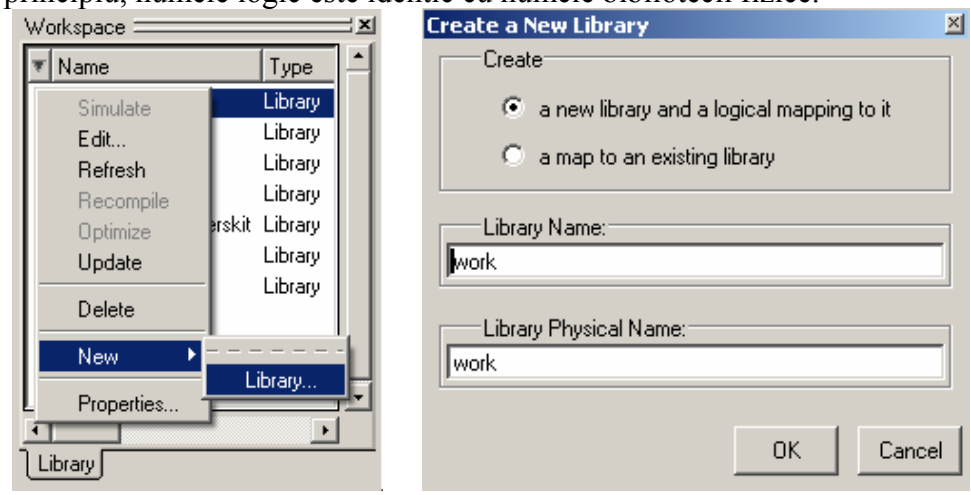
```
D:\PAC2\G5403A\VHDL3A\
```

Din fereastra principală a *ModelSim* selectați directorul de lucru astfel:

Meniu fereastra principală: **File > Change Directory …**

4.2. Crearea bibliotecii *work*

Înainte de a putea compila un fişier sursă VHDL, în directorul de lucru selectat trebuie creată biblioteca de lucru (denumită **work**) unde compilatorul va depune rezultatul compilării. Totodată, pe lângă crearea bibliotecii fizice **work** (care va apare ca subdirector în directorul de lucru) trebuie realizată şi operația de mapare a acesteia prin care i se asociază un nume logic. În principiu, numele logic este identic cu numele bibliotecii fizice.



(a) (b)

Figura 2 – Crearea bibliotecii work

Crearea bibliotecii **work** şi maparea (asocierea) acesteia cu un nume logic se face astfel:

La linia de comandă:
```
vlib work
vmap work work
```
sau, cu mouse-ul poziționat în zona *Workspace*, se execută

click-dreapta > **New** > **Library**

ca în figura 2a. În caseta de dialog *Create a New Library* care apare (figura 2b) se selectează "*a new library and a logical mapping to it*". Asigurați-vă că în câmpurile *Library Name* şi *Library Physical Name* este scris **work**, apoi selectați OK. Această acțiune va determina crearea unui subdirector numit **work** în directorul de lucru. Acest subdirector reprezintă biblioteca **work** şi conține inițial un fişier particular denumit **_info**. Totodată, ca urmare a comenzii *vmap* în directorul de lucru se va copia şi se va modifica fişierul *modelsim.ini* de inițializare a sistemului VHDL. Remarcați în zona *Workspace* că în lista de biblioteci a apărut şi biblioteca **work**.

Observație: Nu creați subdirectorul **work** şi fişierul **_info** cu comenzi Windows. Utilizați întotdeauna comanda *vlib* sau varianta cu mouse descrisă mai sus. Nu ştergeți şi nu modificați cu comenzi Windows fişierele cu rezultatul compilării din cadrul bibliotecii **work**. De asemenea, nu ştergeți fişierul *modelsim.ini*.

În geneal, comenzile pentru lucrul cu biblioteci sînt următoarele :
-vlib -creează o bibliotecă;
-vdel -şterge o unitate de proiect din biblioteca specificată;
-vdir -listează selectiv conţinutul unei biblioteci;
-vmap -defineşte maparea (asocierea) dintre numele logic al unei biblioteci şi numele fizic (director) al acesteia, modificând astfel fişierul *modelsim.ini*.

Odată finalizat procesul de creare a bibliotecii **work** se poate trece la editarea fişierului sursă VHDL şi compilarea acestuia.

4.3. Editarea fişierului sursă în cod VHDL

Editarea fişierului în cod VHDL se poate face cu orice editor simplu de text (Notepad sau Wordpad). Totuşi, este recomandat să se utilizeze editorul de text propriu al *ModelSim* pentru a beneficia de facilităţile oferite de acesta, de exemplu recunoaşterea cuvintelor rezervate din VHDL sau numerotarea liniilor de cod.

Pentru a deschide editorul de text al *ModelSim* (fereastra *source*) în scopul editării unui fişier sursă nou, din fereastra principală se execută:

Meniu fereastra principală: **File** > **New** > **Source** > **VHDL**

Observaţii:

1. Deschiderea editorului de text al ModelSim în scopul vizualizării unui fişier sursă existent se poate face cu comanda

Linia de comandă: `view source`

sau

Meniu fereastra principală: **View** > **Source**

2. Deschiderea pentru editare a unui fişier sursă nou poate fi realizată şi din meniul ferestrei *source* astfel:

Meniu fereastra source: **File** > **New** > **VHDL**

În editorul de text al *ModelSim* se va edita codul VHDL, prezentat mai jos, corespunzător sumatorului complet pe un bit, a cărui structură internă este prezentată în figura 3. Fişierul sursă care conţine declaraţia pentru entitatea *FullAdder* şi arhitectura *Ecuatii* va fi salvat sub numele **FullAdder.vhd**. În VHDL codul este “case insensitive” astfel că nu există nici o deosebire între literele mari şi mici.

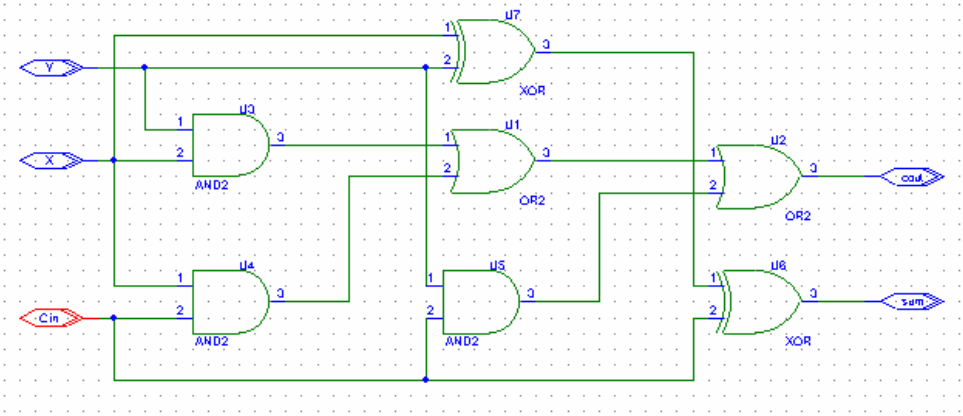


Figura 3 – Structura internă a sumatorului complet pe un bit

Codul VHDL pentru modelarea sumatorului complet pe un bit este următorul:

(fişier FullAdder.vhd)

```
1     library IEEE;
2     use IEEE.std_logic_1164.all;
3     entity FullAdder is
4     port ( X,Y : in Bit;
5          Cin:   in Bit;
6          Cout: out Bit;
7          Sum: out Bit);
8     end FullAdder ;

12    architecture Ecuatii of FullAdder is
13    begin
14          Sum<= X xor Y xor Cin;
15          Cout<= (X and Y) or (X and Cin) or (Y and Cin);
16    end;
```

O reprezentare grafică a modelului descris de entitatea "FullAdder" şi de arhitectura "Ecuatii" este prezentată in figura 4.

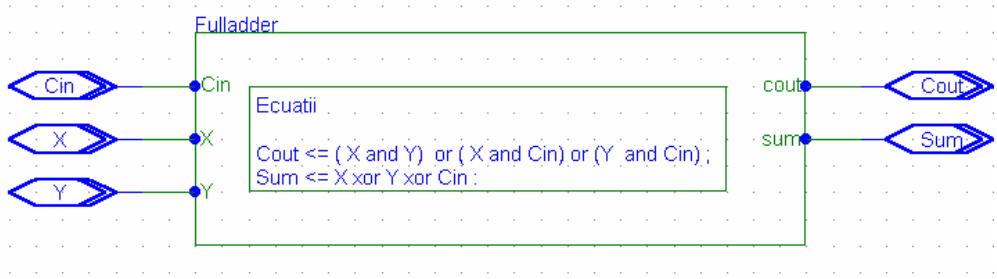


Figura 4 – Reprezentarea grafică a modelului sumatorului complet pe un bit

Explicaţii privind codul VHDL al sumatorului complet pe un bit:

În VHDL orice componentă *hardware* (poate fi o simplă poartă logică sau microprocesor) este alcătuită din două părţi: declaraţia de entitate (*entity interface)*, care defineşte interfaţa modelului cu exteriorul şi corpul arhitecturii (*architecture body*) care defineşte implementarea, descrierea comportamentului sau a structurii modelului.

Declaraţia de entitate, cuprinsă între liniile 3 şi 8, începe cu cuvîntul cheie *entity* urmat de numele entităţii "FullAdder". Entitatea poate fi vazută ca o cutie neagră deoarece defineşte doar porturile prin care aceasta se interconectează cu celelalte componente din sistem.

Pentru un port se specifică *modul*, adică sensul de curgere a informatiei şi tipul de date ce poate fi manipulat de portul respectiv.

Declaraţia de entitate se încheie cu cuvântul rezervat *end* urmat de numele entităţii. Cuvintele cheie în VHDL sunt elemente structurale ale acestuia şi nu pot fi folosite în alte scopuri.

Între cuvintele rezervate *is* şi *end* se află lista de porturi a entităţii, care conţine semnalele de interfaţă cu care modelul poate comunica cu mediul exterior. Lista este alcătuită din declaraţii de semnale de interfaţă separate de punct şi virgulă.

Fiecare declaraţie de semnal de interfaţă declară unul sau mai multe porturi de interfaţă.

Sînt precizate **numele** semnalelor ("X", "Y", "Cout"), apoi *modul*, aici de intrare ("*in*") sau ieşire ("*out*") şi în final un *tip*. Tipul "bit" este predifinit în VHDL şi conţine valorile "0" sau "1".

Modul indică direcţia schimbului de date cu exteriorul. Astfel, semnalele "X", "Y", "Cin" pot fi citite (se vor regasi in partea dreapt[a instrucţiunii de atribuire pentru semnale),

dar nu pot fi modificate în interiorul arhitecturilor entității. La citire ele trec informația în interiorul modelului.

Semnalele "Sum" şi "Cout" pot fi modificate în interiorul arhitecturii entității. La scriere ele trec informația în exteriorul modelului.

Arhitectura asociată entității FullAdder este descrisă de liniile 12–16 şi conține două specificații de atribuire de semnale (liniile 14 şi 15) care definesc valoarea semnalelor "Sum" şi "Cout" în funcție de semnalele de intrare.Valoarea semnalului "Sum" este dată de operatorul SAU-EXCLUSIV aplicat celor trei semnale de intrare, obținînd astfel funcția impară a semnalelor de intrare.

Această modalitate de obținere a semnalelor este denumită descriere sub formă de ecuații a semnalelor deoarece ieşirile modelului sînt definite ca ecuații ale semnalelor de intrare.

O caracteristică fundamentală a arhitecturii este dată de executarea specificațiilor din corpul său în paralel (corpul arhitecturii e mediu concurențial). Astfel, ordinea specificațiilor nu influențează comportamentul modelului.

4.4 Compilarea fişierului sursă

Pentru a compila un fişier sursă, în fereastra principală a *ModelSim* se selectează

Meniu fereastra principală: **Compile** > **Compile…**

sau se apasă pe icon-ul  situat sub bara de meniuri şi apoi, din caseta de dialog care apare (figura 5), se selectează fişierul dorit. În cazul de față selectați fişierul **FullAdder.vhd**. După selectarea fişierului, pentru startarea procesului de compilare se apasă pe **Compile** apoi **Done**.

De asemenea, compilarea fişierului sursă se poate face cu comanda **vcom** astfel:

Linia de comandă: `vcom FullAdder.vhd`

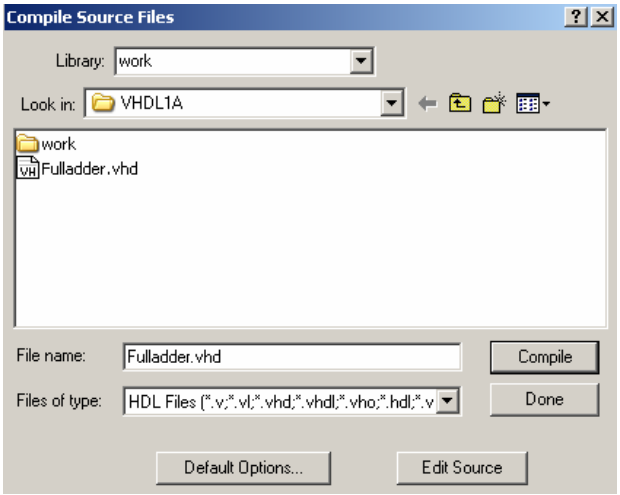


Figura 5 – Caseta de dialog pentru selectarea unui fişier sursă pentru a fi compilat

După compilare în fereastra principală a *ModelSim* trebuie să urmărim dacă procesul de compilare a avut succes sau nu. Se va acorda o atenție deosebită mesajelor de eroare care pot apare precum si mesajelor de avertizare (*warning-uri*).

Mesajele de avertizare au rolul de a atenționa pe proiectant în privința neîndeplinirii câtorva condiții substanțiale de către codul VHDL compilat. De exemplu, unul dintre acestea este cel cu privire la neincluderea în lista de sensitivități a unui proces a unui semnal care este citit în interiorul său. Remedierea avertismentului nu se face în mod automat prin includerea semnalului respectiv în listă deoarece este posibil ca acel proces să nu necesite să fie senzitiv la toate semnalele care sunt citite în interiorul său .

Mesajele de confirmare a compilării fişierului FullAdder.vhd în biblioteca **work** sint:

```
vcom  -work work E:/PAC2/G5401A/VHDL1A/Fulladder.vhd
... …
# -- Loading package standard
# -- Loading package std_logic_1164
# -- Compiling entity fulladder
# -- Compiling architecture ecuatii of fulladder
```

Dacă la compilare nu apar erori şi au fost rezolvate problemele legate de mesajele avertisment se poate trece la simulare.

4.5 Simularea unui design VHDL

Pentru a efectua simularea sumatorului pe un bit, la această etapă se parcurg 4 paşi:

- încărcarea în simulator a perechii entitate-arhitectură a sumatorului;
- generarea formelor de undă pentru porturile de intrare ale sumatorului;
- selectarea semnalelor a căror forme de undă se doresc a fi vizualizate în urma simulării;
- lansarea în execuţie a simulării.

La primul pas în vederea simulării, în fereastra principală a *ModelSim* se încarcă în simulator perechea entitate-arhitectură din biblioteca **work** fie cu comanda **vsim** de la linia de comandă, fie din meniul ferestrei principale, astfel:

La linia de comandă: `vsim work.fulladder(ecuatii)`

sau

Meniu fereastra principală: **Simulate** > **Simulate …**

după care, din caseta de dialog (figura 6) se apasă pe semnul + din dreptul bibliotecii **work**, apoi pe semnul + din dreptul entităţii *fulladder*, în continuare se selectează arhitectura *ecuatii* şi, în final, se apasă OK.

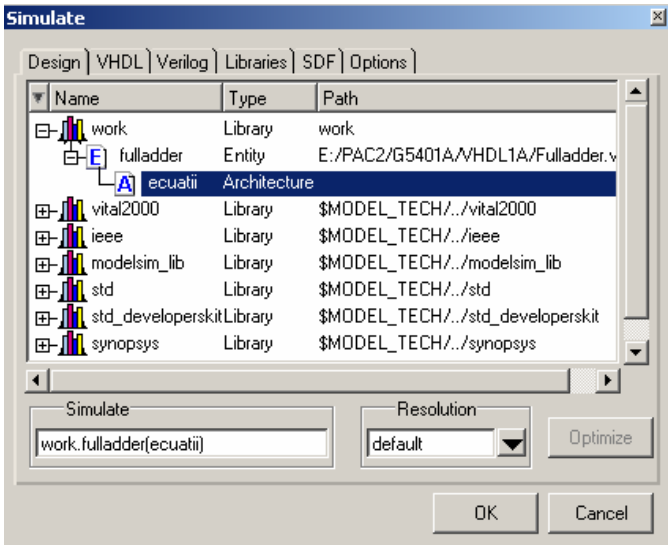


Figura 6 – Caseta de dialog pentru selectarea unui design în vederea simulării

După efectuarea acestui prim pas urmărim din nou fereastra principală unde putem vedea mesajele de confirmare a încărcării în simulator a designului din biblioteca **work**:

```
ModelSim> vsim work.fulladder(ecuatii)
# vsim work.fulladder(ecuatii)
# Loading C:\Modeltech\win32/../std.standard
# Loading C:\Modeltech\win32/../ieee.std_logic_1164(body)
# Loading work.fulladder(ecuatii)
```

Dacă apar doar mesajele de încărcare atunci înseamnă că designul poate fi simulat.

Dar, atenţie, acest lucru nu înseamnă automat că am descris corect dispozitivul propus, ci doar că nu avem erori de limbaj VHDL! Pot exista la acest nivel doar greşeli de concepţie care ne aparţin in totalitate! De exemplu, expresiile care se atribuie porturilor SUM şi COUT pot fi greşite.

În general, în etapa de încărcare a proiectului în simulator se realizează aşa-numitul proces de elaborare a proiectului prin care, de exemplu, componentelor instanţiate în proiect li se asociază perechile entitate-arhitectură specificate. Dacă perechea entitate-arhitectură corespunzătoare nu este găsită în bibliotecă (de exemplu, nu a fost compilată incă), atunci la încărcarea proiectului în simulator se va afişa un mesaj de eroare.

Generarea formelor de undă pentru porturile de intrare ale sumatorului

Generarea formelor de undă pentru intrările X, Y şi CIN ale sumatorului se realizează cu ajutorul comenzii **force**. Vom genera aceste forme de undă astfel încât să parcurgem toate combinaţiile posibile de valori pentru cele 3 intrări pentru a putea urmări rezultatele în concordanţă cu tabelul de adevăr, prezentat mai jos. Primele trei coloane din tabelul de adevăr reprezintă intrările în sumator: cele două intrări de sumat X, respectiv Y şi intrarea de transport Cin. Următoarele două coloane prezintă ieşirile: Sum – rezultatul sumării pe un bit a lui X cu Y şi Cout – transportul rezultat din adunare.

Tabelul 1. Tabelul de adevăr al sumatorului pe 1 bit

X	Y	Cin	Sum	Cout
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

Sintaxa comenzii **force** este următoarea:

force *semnal val_initala, val1 timp1 val2 timp2… valn timpn* [–repeat *interval*]

unde fiecare pereche *valk timpk* semnifică valoarea pe care o va lua semnalul respectiv la momentul de timp *timpk*. Opţiunea [–repeat *interval*] determină ca secvenţa de tranziţii specificate să se repete la intervale de timp egale cu *interval*.

În linia de comandă din fereastra principală introduceţi, pe rând, următoarele comenzi **force**:

```
force X   0, 1 10 ns, 0 20 ns –repeat 20 ns
force Y   0, 1 20 ns, 0 40 ns –repeat 40 ns
force CIN 0, 1 40 ns, 0 80 ns –repeat 80 ns
```

În acest fel, la intrarea X se va aplica un semnal cu factor de umplere 50% şi perioada de 20ns, la intrarea Y semnalul va avea perioada de 40ns iar la CIN va avea perioada de 80ns.

Observaţie: Remarcaţi că în comanda **force** unitatea de măsură a timpului (`ns`) se scrie separat de valoare numerică. Acest aspect este valabil şi în codul VHDL.

Selectarea semnalelor pentru vizualizare

Înainte de a lansa în execuţie simularea circuitului trebuie selectate semnalele care se

doresc a fi vizualizate. Vizualizarea formelor de undă rezultate în urma simulării a semnalelor selectate se va face în fereastra *Wave*. Selectarea semnalelor se realizează în felul următor:

- se deschid ferestrele *Signals* şi *Wave* prin introducerea următoarelor comenzi în fereastra principală:

La linia de comandă: `view signal wave`

sau

Meniu fereastra principală: **View** > **Signals, View** > **Wave**

- Din fereastra *Signals* se selectează semnalele ce se doresc a fi afişate în fereastra Wave după care din meniu (ca în figura 7) se selectează:

Meniu fereastra Signals: **Add** > **Wave** > **Signals in Region** (sau **Selected Signals**)

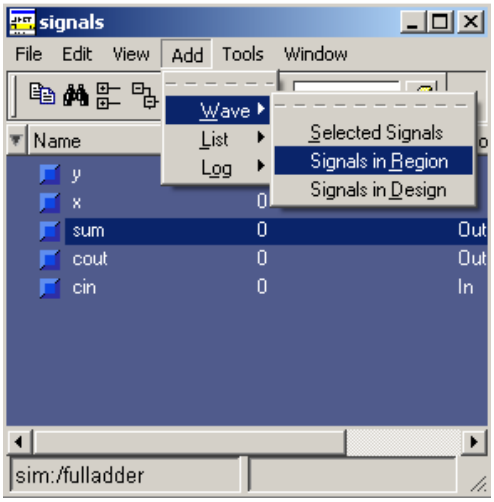


Figura 7 – Selectarea semnalelor din fereastra *Signals* pentru a fi adăugate în fereastra *Wave*.

Observaţie: Semnalele din fereastra *Signals* pot fi aduse în fereastra *Wave* şi folosind facilităţile "drag and drop". Astfel, în fereastra Signals cu control-click se selectează semnalele dorite după care, ţinând apăsat, se trage cursorul în partea stângă a ferestrei **Wave**.

Lansarea în execuţie a simulării

Pentru a lansa în execuţie simularea, de exemplu pe durata a 100ns, din fereastra principală a ModelSim se execută:

La linia de comandă: `run 100 ns`

sau

Meniu fereastra principală: **Simulate** > **Run** > **Run 100 ns**

sau se apasă click pe icon-ul . Acest icon poate fi acţionat şi din alte ferestre (*Source*, *Wave*).

4.6 Vizualizarea rezultatelor

În urma simulării formele de undă rezultate pentru semnalele selectate se pot vizualiza în fereastra *Wave*, aşa cum se observă în figura 8:

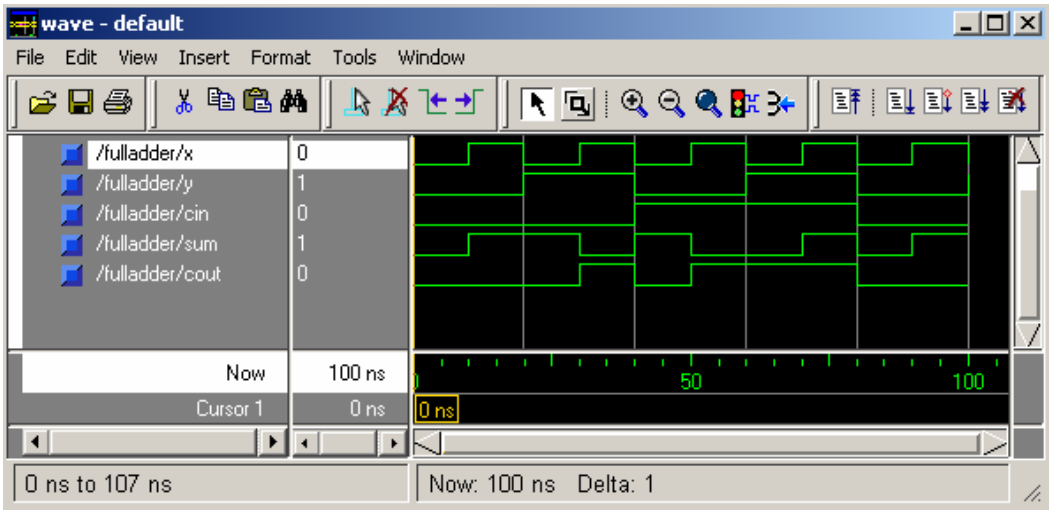


Figura 8. Fereastra Wave cu rezultatele simulării

În fereastra Wave, cu ajutorul icon-urilor următoare  se pot adăuga sau şterge cursoare, respectiv se pot muta cursoarele la momentele tranzițiilor semnalelor.

Încheierea simulării

În final, pentru a ieşi din mediul de simulare al ModelSim, din fereastra principală se execută:

Meniu fereastra principală: **Simulate** > **End simulation**

5 Sarcini de îndeplinit la laborator:

1. Citiți cu atenție referatul şi parcurgeți etapele de la punctul 4 pentru simularea sumatorului pe un bit.
2. Urmăriți formele de undă obținute in fereastra **Wave** şi comentați semnificația lor. Coincid rezultatele cu cele din tabelul de adevăr?

6. Inversați ordinea celor două atribuiri de semnale în arhitectura "Ecuații" a sumatorului. Ce observați? Se modifică sau nu funcționalitatea sumatorului?

7. Scrieți tabelul de adevăr şi codul VHDL pentru declarația de entitate si arhitectura cu ecuații (modelare data-flow) a circuitului din figura 9. Ce funcție logică îndeplineşte el? Compilați şi simulați circuitul cu ModelSim procedând similar ca la sumatorul pe un bit.

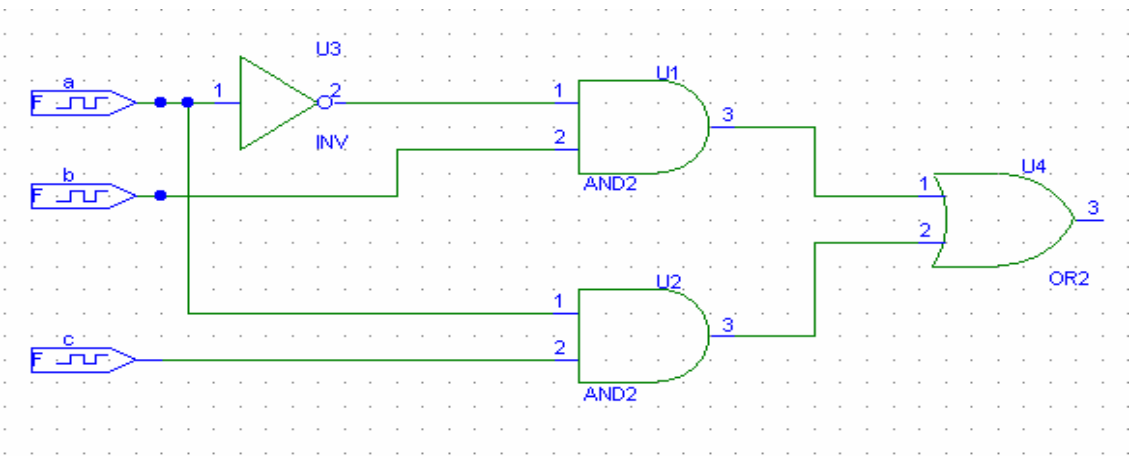


Figura 9.