Lucrarea 2

Etapele simulării cu *ModelSim*

Această lucrare prezintă modul de realizare a unui model Verilog sau VHDL, precum şi posibilităţile de simulare cu ajutorul simulatorului *ModelSim*.

2.1 Scopul lucrării

- Construcţia unui modul de simulare pentru un latch şi pentru un bistabil D.
- Construcţia unui mediu de simulare.
- Configurarea *ModelSim*.
- Examinarea formelor de undă ale semnalelor.
- Prezentarea ferestrelor simulatorului *ModelSim*.
- Compararea funcţionării a două module.

2.2 Construcţia modelului de simulare pentru un latch D

Modelul VHDL al unui latch D este prezentat în continuare:

```
-- Entitatea
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;

ENTITY D_latch IS
  PORT(Ck, D: IN std_logic;
       Q, QN: OUT std_logic);
END D_latch;

-- Arhitectura
ARCHITECTURE Comportament OF D_latch IS
BEGIN

Latch: PROCESS(Ck, D)
  BEGIN
    IF (Ck='1') THEN
      Q  <= D;
      QN <= Not D;
    END IF;
  END PROCESS;

END Comportament;
```

Modulul Verilog al unui latch D este prezentat în continuare:

```
module D_latch (Ck, D, Q, QN);

input    Ck;
input    D;
output   Q;
output   QN;

  assign Q  = Ck ?  D : Q;
  assign QN = Ck ? ~D : QN;

endmodule
```

Observaţii:

- Unui cuplu entitate-arhitectură din VHDL îi corespunde un modul în Verilog.
- În VHDL, porturile se descriu în cadrul entităţii. În Verilog, porturile se descriu la începutul modulului.
- În VHDL, funcţionalitatea sau structura sistemului este descrisă în cadrul arhitecturii. Lista de senzitivităţi a procesului *Latch* conţine semnalele care determină re-evaluarea ieşirilor.
- Funcţionalitatea modelului Verilog este descrisă prin atribuirile semnalelor de iesire prin specificaţii *assign*.
- Un model Verilog alternativ pentru latch, similar cu modelul VHDL este prezentat în continuare:

```
module D_latch (Ck, D, Q, QN);
input    Ck;
input    D;
output   Q;
output   QN;

always @(Ck or D)
  if (Ck) begin
    Q <= D;
    QN <= ~D;
  end

endmodule
```

2.3 Construcţia mediului de simulare pentru un latch D

Pentru verificarea funcţionalităţii componentei descrise, este necesară crearea unui mediu care să asigure generarea unui set de stimuli pentru modelul de testat.

Un mediu de testare minimal este compus dintr-un modul de testare care, pe lângă instanţierea componentei de testat (D.U.T. - *Device Under Test*), generează şi stimuli pentru aceasta. O schemă de principiu a unui astfel de modul este cea prezentată în figura 2.1.

O variantă îmbunătăţită a acestui modul de test constă în realizarea unei componente separate care să aibă rolul unui generator de stimuli. Acest se numeşte test-bench (prescurtat TB). Această metodologie asigură posibilitatea de încapsulare a sursei stimulilor precum şi reutilizarea modulului TB. Un modul general de test, care conţine o entitate TB, este prezentat în figura 2.2.

Pentru studierea funcţionalităţii modelului de latch D, se va folosi varianta cea mai simplă pentru construirea modulului de test.

Descrierea VHDL a unui modul de test este prezentată în continuare:

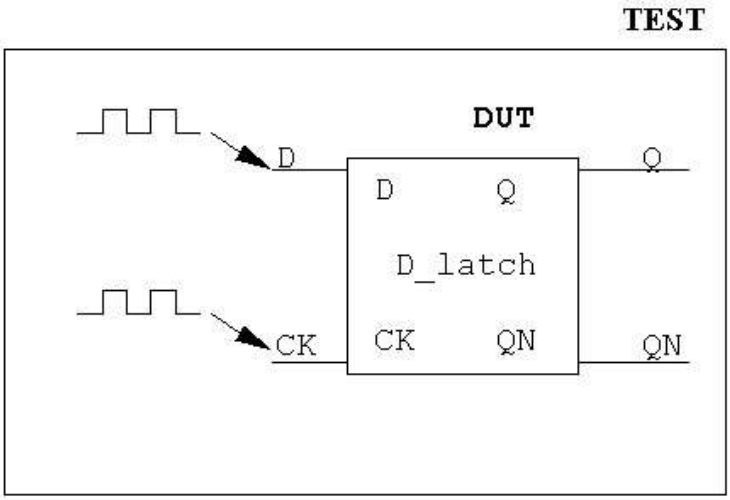

Figura 2.1: Schema unui modul de test minimal.

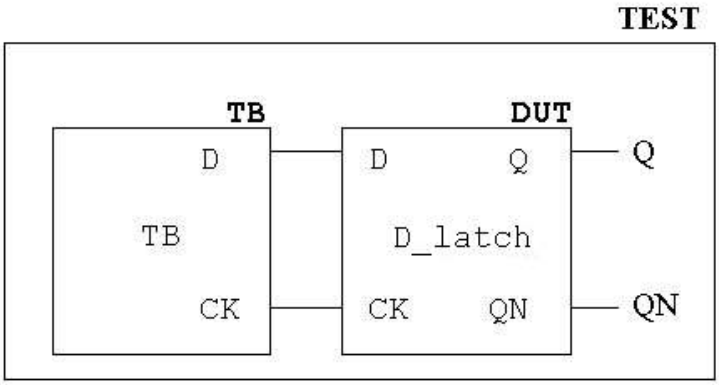

Figura 2.2: Schema pentru un modul de testare reutilizatbil.

```
-- Entitatea de test
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;

ENTITY Test IS
END Test;

-- Arhitectura asociată entităţii de test
ARCHITECTURE Test OF Test IS
  SIGNAL D, Q, Qn: std_logic;
  SIGNAL Ck: std_logic := '0';

  COMPONENT Dcomp
    PORT(Ck, D: IN std_logic;
         Q, QN: OUT std_logic);
  END COMPONENT;
  FOR ALL: Dcomp USE ENTITY WORK.D_latch(Comportament);
```

```
BEGIN

DUT: Dcomp PORT MAP(
        Ck => Ck,
        D => D,
        Q => Q,
        QN => QN
);

Ck <= Not Ck AFTER 10 ns;
D <= '0', '1' AFTER 5 ns, '0' AFTER 35 ns, '1' AFTER 37 ns, '0' AFTER 45 ns;

END Test;
```

În Verilog, acelaşi modul de testare este descris astfel:

```
//Modul de test
‘timescale 1ns/100ps

module Test ();
reg   D;
wire  Q, QN;
reg   Ck;

D_latch DUT (
        .Ck(Ck),
        .D(D),
        .Q(Q),
        .QN(QN)
);

initial
begin
     Ck = 0;
     D  = 0;
 #5  D  = 1;
 #30 D  = 0;
 #2  D  = 1;
 #8  D  = 0;
 end

 always  #10 Ck <= ~Ck;

endmodule
```

Observaţii:

- În VHDL, instanţierea componentei de testat se face explicit, prin definirea unei componente şi asocierea acesteia cu enitatea-arhitectura de latch definită anterior. VHDL-93 acceptă asocierea implicită, prin nume, a componentelor.

- În Verilog, instanţierea componentei de testat se face implicit, asocierea făcându-se prin nume.
- Descrierea semnalului de ceas se face prin stabilirea unei valori iniţiale şi prin comutarea valorii acestuia după o jumătate de perioadă.
- Valorile aflate după simbolul # reprezintă unităţi de timp. Valoarea absolută a unităţii de timp în Verilog se introduce cu directiva *'timescale*.

2.4 Simularea modelelor de latch D

Pentru a simula modelele de latch D, se parcurg următoarele etape:

1. Copiaţi fişierele Verilog (cu extensia ".v") şi VHDL (cu extensia ".vhd") din directorul *\asd_hdl\exemple\l2* în directorul *C:\Home*. Acest director va fi director de proiect în *ModelSim*.
2. Lansaţi în execuţie simulatorul *ModelSim*.
3. Setaţi directorul cu fişierele Verilog ca director curent.

   Fereastra principală: File > Change Directory...
4. Creaţi biblioteca de proiect, (denumită **work**) în directorul curent. În această bibliotecă se vor plasa modelele compilate:

   Fereastra principală: Design > Create a New Library...

   Prompter: vlib work
5. Compilaţi fişierele sursă Verilog. Proiectul considerat constă din două fişiere sursă, fiecare conţinînd cîte un modul. Fişierul **latch.v** conţine un modul **latch**, care implementează latch-ul de tip D. Celălalt fişier, **test.v**, conţine modulul **test** utilizat pentru verificarea modulului **latch**. Compilarea fişierelor se face prin selectarea butonului *Compile*.

   Prompter: vlog latch.v

   Această comandă va deschide caseta de dialog *Compile HDL Source Files* (figura 2.3).

   Selectaţi ambele fişiere sursă (control-clic stânga pe **latch.v**, apoi pe **test.v**). Selectaţi *Compile*. Urmăriţi în fereastra principală eventualele mesaje de eroare sau atenţionare. Selectaţi apoi *Done* pentru închiderea casetei de dialog.
6. Lansaţi simulatorul selectând butonul *Load Design*.

   Prompter: vsim test

   Va apare caseta de dialog *Load Design*, similar cu cea prezentată în figura 2.4.

   Caseta de dialog *Load Design* permite selectarea din biblioteca specificată a unităţii de proiectare ce se va simula. Se poate selecta şi rezoluţia de simulare. Biblioteca implicită este **work** iar rezoluţia de simulare implicită este 1 ns. Selectaţi *Design Unit* **test** şi selectaţi butonul *Load* pentru acceptare.

   În simulare se va vedea că aceste două fişiere sunt configurate ierarhic având instanţiat un modul **latch** (numele instanţei este DUT). Structura proiectului se va putea vizualiza ulterior în fereastra *Structure*.

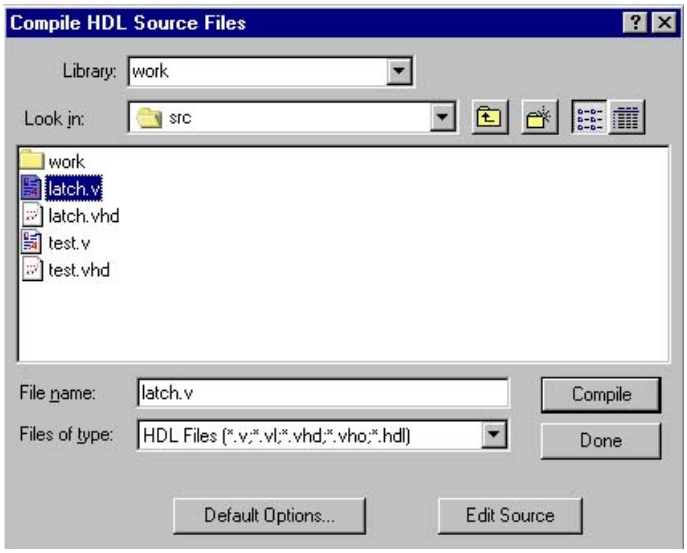

Figura 2.3: Caseta de dialog *Compile HDL Source Files*.

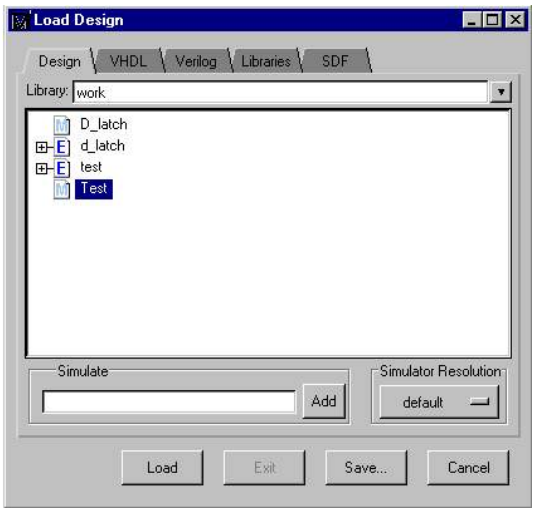

Figura 2.4: Caseta de dialog *Load Design*.

Pentru simularea modelelor VHDL, succesiunea etapelor este aceeaşi, cu deosebirea că la compilarea fişierelor VHDL din linia de comandă se foloseşte comanda:

Prompter: vcom latch.vhd

Prompter: vcom test.vhd

2.5 Vizualizarea grafică a semnalelor

Pentru vizualizarea semnalelor generate de către modulele descrise sunt necesare următoarele etape:

1. Deschideţi ferestrele *Signals* şi *Wave* prin introducerea următoarelor comenzi la prompterul VSIM din fereastra principală:

   Prompter: view signals wave

Fereastra principală: View > Signals, View > Wave

2. Stabiliţi semnalele care se vor afişa în fereastra *Wave*, selectând fereastra *Signals* şi apelând meniul *View > Wave > Signals in Region* (figura 2.5). Exersaţi diferitele funcţii

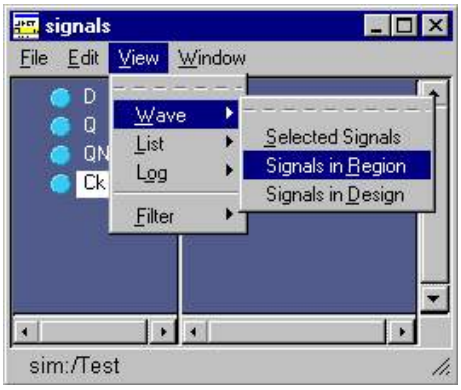

Figura 2.5: Fereastra *Signals*.

de rulare. Selectaţi butonul *Run*. Această acţiune va determina rularea simulării pentru un timp selectat (implicit 100 ns).

Prompter: run

Fereastra principală: Run > Run 100 ns

3. Schimbaţi durata rulării la 200 ns folosind selectorul de lîngă butonul *Run* şi apoi apăsaţi din nou butonul. Acum simularea va ajunge la 300 ns (100 ns iniţiale plus încă 200 ns). Timpul afişat în partea de jos a ferestrei principale prezintă acest lucru.

4. Activaţi fereastra *Wave* pentru a vedea formele de undă ale semnalelor. Pentru a vizualiza formele de undă, testaţi diferitele variante de zoom. Semnalele afişate trebuie să fie asemănătoare celor prezentate în figura 2.6.

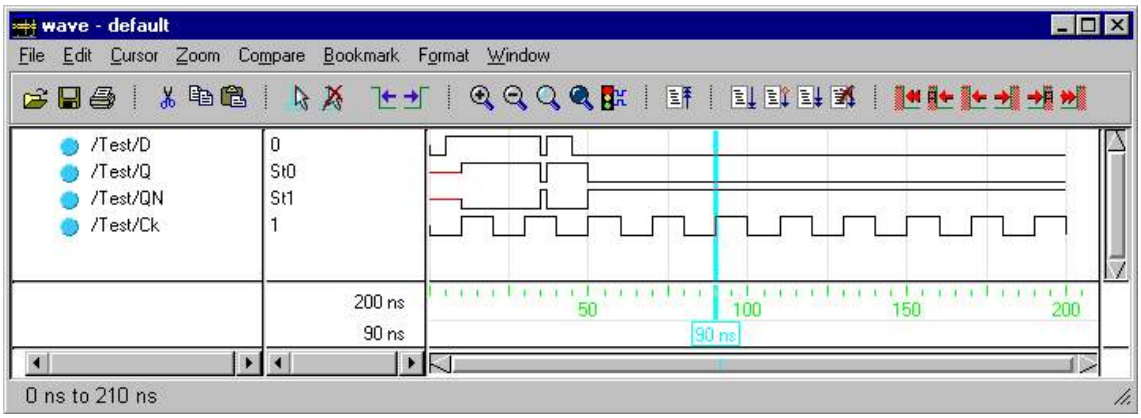

Figura 2.6: Fereastra *Wave*.

5. Studiaţi funcţionalitatea modelului descris, pe baza formelor de undă a semnalelor de ieşire, în funcţie de configuraţia semnalelor de intrare. Demonstraţi că formele de undă aparţin unui latch de tip D.

2.6 Teme

1. Creaţi un modul latch activ pe palierul de 0 al ceasului.
2. Creaţi un modul TB pentru generarea de stimuli pentru latch-ul de tip D. Modificaţi modulul de test corespunzător.
3. Construiţi un modul de latch cu reset asincron. Testaţi modulul construit.
4. Justificaţi prezenţa semnalului D în lista de senzitivităţi a procesului L (în cadrul arhitecturii latch a entităţii D).
5. Explicaţi de ce nu funcţionează următorul sistem:

```
-- Entitatea
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;

ENTITY D IS
  PORT(Ck, D: IN std_logic;
       Q, QN: OUT std_logic);
END D;

-- Arhitectura
ARCHITECTURE Latch OF D IS
BEGIN
L: PROCESS(Ck, D)
  BEGIN
    IF (Ck='1') THEN
      Q  <= D;
      QN <= Not Q;
    END IF;
  END PROCESS;
END Latch;
```

   Cum se poate corecta codul respectiv, pentru ca sistemul să funcţioneze ca un latch de tip D?

2.7 Prezentarea ferestrelor *Modelsim*

ModelSim conţine o serie de ferestre utile în cadrul simulării şi a depanării. Cele mai utilizate tipuri de ferestre sunt: *List*, *Signals*, *Source*, *Structure*, *Wave*.

În cadrul unui proiect, pentru vizualizarea ferestrelor, se parcurg următoarele etape:

1. Simulaţi proiectul *latch*.
2. Deschideţi fereastra *Structure*:

   Prompter: view structure

   Fereastra principală: View > Structure

   Va apare o fereastră ca cea prezentată în figura 2.7.

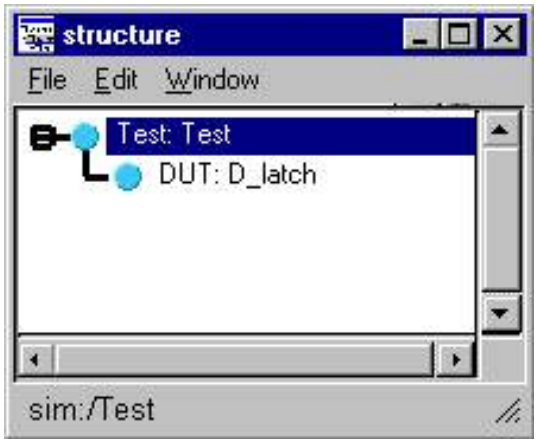

Figura 2.7: Fereastra *Structure*.

Această fereastră prezintă ierarhia elementelor structurale pentru proiectul curent, astfel:

- VHDL (elemente indicate printr-un pătrat albastru închis): pachete, componente instanţiate, blocuri;
- Verilog (elemente indicate printr-un cerc albastru deschis): instanţe, blocuri, task-uri, funcţii.

Puteţi expanda sau contracta vederea asupra ierarhiei apăsând pe casetele care conţin simbolurile '+' sau '-'.

Prima linie a ferestrei *Structure* indică modelul de nivel înalt care este simulat ("top level" - engl.).

Numele unei instanţe din cadrul ferestrei *Structure* constă din următoarele părţi:

- eticheta instanţei;
- numele modulului (Verilog) sau a entităţii (VHDL);
- arhitectura (VHDL).

Prin selectarea unei regiuni din cadrul ferestrei *Structure*, aceasta devine regiunea curentă, iar ferestrele *Signals* şi *Source* îşi actualizează dinamic informaţiile.

3. Deschideţi fereastra *Signals*:

   Prompter: view signals

   Fereastra principală: View > Signals

   Va apare o fereastră ca cea prezentată în figura 2.8.

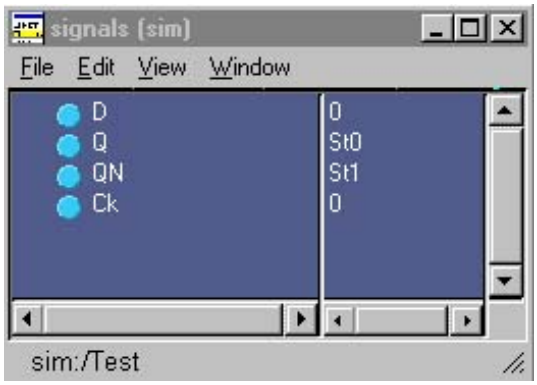

Figura 2.8: Fereastra *Signals*.

Această fereastră prezintă valorile curente ale semnalelor din regiunea selectată în cadrul ferestrei *Structure*. Fereastra este împărţită în două secţiuni:

- secţiunea din partea stîngă conţine numele elementelor HDL din regiunea curentă;
- secţiunea din partea dreaptă conţine valorile instantanee ale elementelor prezentate în partea stângă.

4. Deschideţi fereastra *List*:

   Fereastra Signals: View > List > Signals in Region

   Va apare o fereastră ca cea prezentată în figura 2.9.

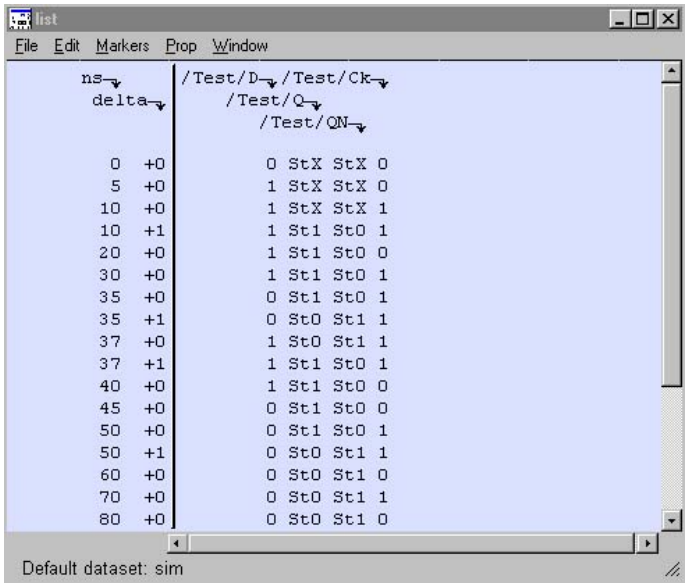

Figura 2.9: Fereastra *List*.

Această fereastră prezintă valorile în cadrul simulării a elementelor selectate în fereastra *Signals*, în format tabelar. Fereastra este împărţită în două secţiuni:

- secţiunea din partea stîngă conţine valorile timpului şi ale ciclului de simulare, vizibile permanent;
- secţiunea din partea dreaptă conţine valorile semnalelor HDL.

5. Deschideţi fereastra *Wave*:

   Fereastra Signals: View > Wave > Signals in Region

   Fereastra prezintă formele de undă a elementelor din regiunea curentă, selectate în cadrul ferestrei *Signals*. Fereastra este împărţită în următoarele secţiuni, în conformitate cu figura 2.10:

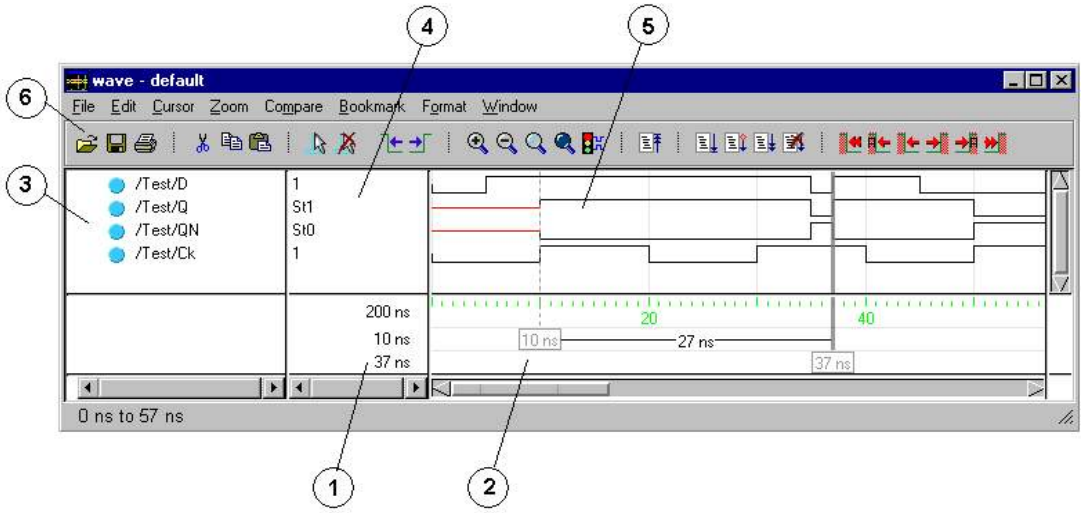

Figura 2.10: Fereastra *Wave*.

- Două secţiuni cursor (1 şi 2), în care sînt afişate valorile timpului pentru fiecare cursor;
- O secţiune cu numele elementelor pentru care se afişează formele de undă (3);
- O secţiune cu valorile curente a elementelor HDL afişate (4);
- O secţiune cu formele de undă (5);
- O secţiune cu bara de comenzi rapide (6).

Semnificaţia icon-urilor este prezentată în figura 2.11.

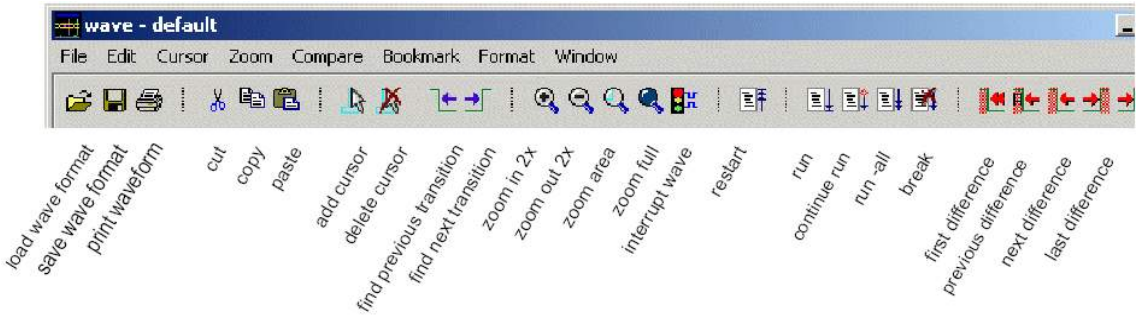

Figura 2.11: Bara de comenzi rapide pentru fereastra *Wave*.

6. Rulaţi modelul simulat pentru 200 ns şi urmăriţi modificările care se produc în cadrul ferestrelor prezentate.

7. Măsuraţi duratele între două fronturi succesive ale semnalelor de ieşire, folosind informaţiile din cadrul ferestrelor *Wave* şi *List*. Poziţionarea cursorului pe frontul următor al semnalului se poate face (în cadrul ferestrei *Wave*), cu tasta TAB sau apăsând pe icon-ul *"Find next transition"*.

2.8 Construcţia modelului HDL al unui bistabil

Pentru modelarea unui bistabil, trebuie creat un proces senzitiv doar la frontul crescător al semnalului de ceas. Acest lucru se modelează astfel:

- VHDL

```
Bist: PROCESS (CK)
   BEGIN
     IF (CK = '1' AND CK'EVENT) THEN
       Q <= D;
       Q <= Not D;
     END IF;
   END PROCESS;
```

- Verilog

```
always @ (posedge ck)
  begin
    q  <= d;
    qn <= ~d;
  end
```

1. Creaţi un model Verilog şi unul VHDL pentru un bistabil de tip D, folosind specificaţiile prezentate.
2. Simulaţi modelele create şi verificaţi funcţionarea acestora conform cerinţelor pentru un bistabil. Formele de undă trebuie să fie similare cu cele prezentate în figura 2.12.

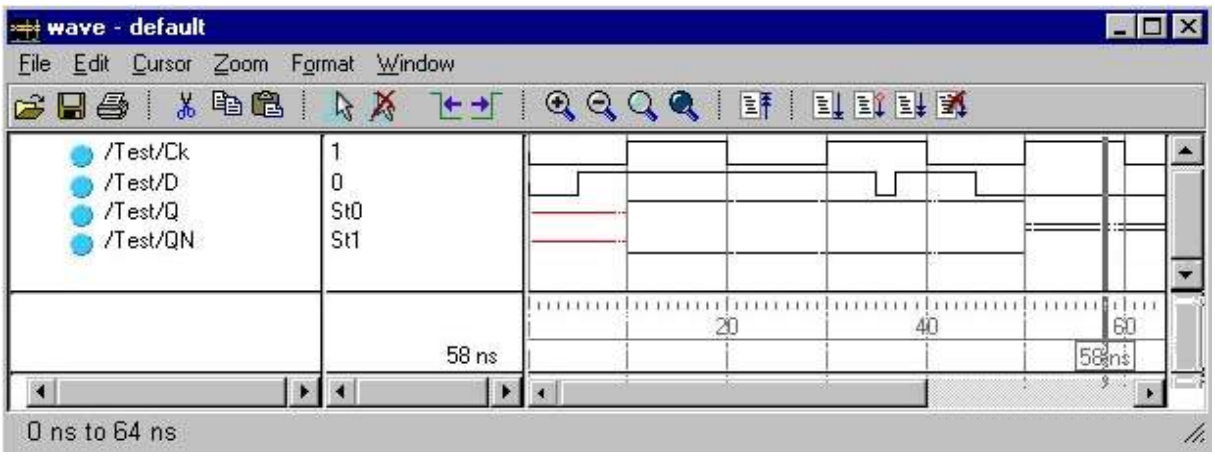

Figura 2.12: Formele de undă ale unui bistabil de tip D.

2.9 Compararea comportamentului unui Latch D cu cel al unui Bistabil D

Pentru compararea comportamentului unui **Latch D** cu cel al unui **Bistabil D**, este necesară crearea unui mediu de testare care să conţină instanţe ale celor două modele. De asemenea,

mediul de testare trebuie să conţină un modul de comparare a formelor de undă. Structura unui astfel de mediu este prezentată în figura 2.13.

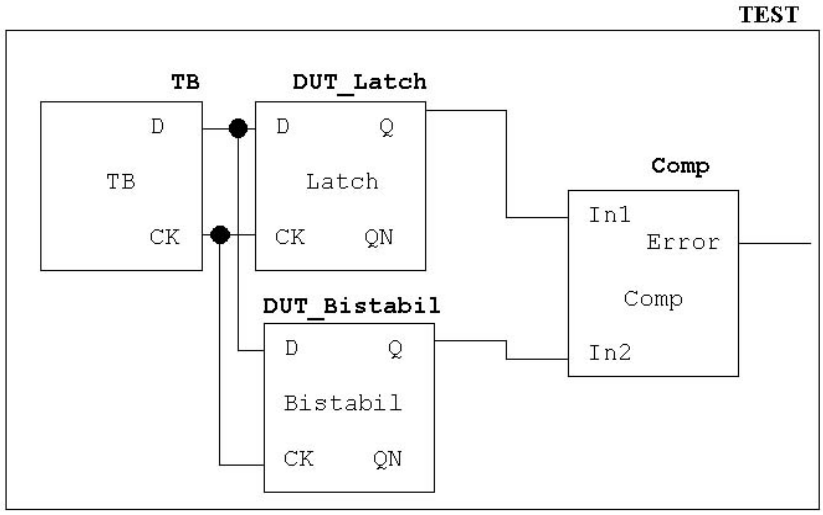

Figura 2.13: Structura unui mediu pentru compararea comportamentului unui Latch D cu cel a unui Bistabil D.

Modelul comparatorului, descris în Verilog este:

```
module comp (in1, in2, error);

input in1, in2;
output error;

assign error = in1 ^ in2;

endmodule
```

Construiţi modulul **TEST** prezentat în figura 2.13 şi explicaţi diferenţele care există între formele de undă ale celor două module (**Latch D** şi **Bistabil D**).

2.10 Teme

1. Creaţi un model de bistabil de tip D cu intrări de Set şi Reset asincrone.
2. Creaţi un registru de tip D pe 8 biţi.
3. Construiţi un bistabil de tip T.
4. Construiţi un model de simulare care conţine două bistabile: unul modelat în VHDL şi unul modelat în Verilog. Studiaţi diferenţele existente în cadrul ferestrelor *ModelSim*.
5. Explicaţi de ce acest bistabil de tip D nu funcţionează corect:

```
module Dff_Res_Bad(D ,Q ,Clock, Reset);

output Q;
input D,Clock,Reset;

reg Q;
wire D;

always @(posedge Clock)
  if (Reset !== 1) Q = D;

always
  if (Reset == 1) Q = 0;

end

endmodule
```

6. Testaţi următorul model de bistabil D:

```
module DFF (D, Q, Clk, Rst);
parameter width = 1,  reset_value = 0;

input [width-1:0] D;
input Clk,Rst;
output [width-1:0] Q;

reg [width-1:0] Q;

initial
  Q = {width{1'bx}};

always @ ( posedge Clk or negedge Rst )
  if ( Rst == 0 )
    Q <= #1 reset_value;
  else
    Q <= #1 D;
```

```
endmodule
```

7. Explicaţi următorul model:

```
module DFFSCAN (D, Q, Clk, Rst, ScEn, ScIn, ScOut);

parameter width = 1,  reset_value = 0;

input [width-1:0] D;
output [width-1:0] Q;
input Clk,Rst,ScEn,ScIn;
output ScOut;

reg [width-1:0] Q;

initial
  Q = {width{1'bx}};

always @ ( posedge Clk or negedge Rst ) begin
  if ( Rst == 0 )
    Q <= #1 reset_value;
  else
    if (ScEn)
      Q <= #1 {Q,ScIn};
    else
      Q <= #1 D;
end

assign ScOut=Q[width-1];

endmodule
```