SIMULAREA FUNCȚIONALĂ A CIRCUITELOR CU SIMULATORUL MODELSIM

În această lucrare se prezintă principiul simulatoarelor şi al simulării funcționale asistate de calculator pentru circuite, iar apoi se prezintă un exemplu de simulare utilizând utilitarele puse la dispoziție de sistemul de proiectare Xilinx ISE pentru simularea funcțională a circuitelor. Aceste utilitare sunt programul *HDL Bencher*, care permite crearea unor bancuri de test necesare pentru simulare, şi simulatorul Model*Sim*, care permite simularea circuitelor specificate prin scheme logice sau limbaje de descriere hardware.

1. Simularea asistată de calculator

1.1. Principiul simulării asistate de calculator

Simularea unui circuit permite verificarea funcționării circuitului pe baza specificării acestuia prin oricare din metodele uzuale (limbaje de descriere hardware, scheme, diagrame de stare), înainte de implementarea efectivă a circuitului. Simularea asistată de calculator este metoda cea mai utilizată pentru simularea circuitelor, utilizarea acestei metode devenind posibilă şi pentru sistemele digitale complexe odată cu creşterea semnificativă a performanței sistemelor de calcul moderne. Simularea se realizează cu ajutorul unui program simulator, care construieşte un model al circuitului pe baza descrierii acestuia de către proiectant, model care este echivalent cu o copie virtuală a circuitului proiectat. Simulatorul execută apoi modelul circuitului şi analizează răspunsul acestuia la o serie de combinații ale intrărilor aplicate circuitului într-un anumit interval de timp; o asemenea combinație este numită *vector de test* sau *stimul*.

Pe lângă vectorii de test, proiectantul poate specifica şi ieşirile aşteptate ale circuitului pentru fiecare vector. În acest caz, simulatorul va compara ieşirile generate prin execuția modelului cu cele specificate de proiectant, care sunt cunoscute ca fiind corecte, afişând mesaje de eroare în cazul existenței unor diferențe între acestea. Dacă se detectează erori la simulare, descrierea circuitului poate fi corectată mult mai simplu decât în cazul în care circuitul a fost implementat.

Pentru simularea sistematică a funcționării unui circuit, de multe ori se creează un *banc de test* ("*testbench*"), constând din circuitul care trebuie testat şi module adiționale pentru generarea vectorilor de test şi aplicarea acestora la intrările circuitului. Vectorii de test pot fi reprezentați fie de toate combinațiile posibile ale intrărilor circuitului (dacă este posibil), fie de combinațiile reprezentative pentru toate situațiile de funcționare. Prin utilizarea bancurilor de test este posibilă simularea circuitului în condiții reale, când acestuia i se aplică intrări din mediul exterior, de la un operator uman, sau de la un alt circuit sau sistem digital.

Simularea asistată de calculator are următoarele avantaje principale:

- Simularea permite experimentarea cu diferite variante de proiectare şi condiții de funcționare ale circuitului, fără a fi necesară implementarea acestor variante. Proiectantul va putea implementa apoi varianta cea mai eficientă din punctul de vedere al performanțelor şi al costului.
- Prin utilizarea simulării asistate de calculator este posibilă testarea sistematică a circuitelor prin aplicarea unui număr mare de vectori de test la intrările circuitului, vectori care pot fi generați prin program sau pot fi specificați sub formă tabelară.

- Pentru circuite sau sisteme digitale complexe, simularea asigură reducerea costurilor şi reducerea numărului erorilor de proiectare comparativ cu cazul în care se utilizează un prototip hardware pentru testare. Realizarea unor circuite complexe cum sunt microprocesoarele moderne nu ar fi fost posibilă fără utilizarea intensivă a simulării asistate de calculator.

1.2. Tipuri de simulare

Simularea poate fi executată în diferite etape ale procesului de proiectare şi poate utiliza diferite nivele de abstractizare ale circuitului specificat printr-o anumită metodă.

Simularea funcțională constă în simularea unei descrieri la nivel înalt a circuitului. Această descriere specifică funcționarea circuitului, şi nu structura acestuia. Pentru descrierea circuitului se pot utiliza diferite limbaje de modelare, cum sunt limbajele de programare. În acest caz, simularea constă în compilarea şi execuția modelului respectiv. În cele mai multe cazuri, simularea funcțională se bazează pe un model al circuitului sub forma unei descrieri într-un limbaj de descriere hardware (HDL).

Simularea logică reprezintă analiza funcționării unui circuit pe baza valorii unor variabile logice. Această simulare este numită şi simulare *la nivelul transferurilor între registre* (RTL – *Register Transfer Level*), deoarece variabilele simulate sunt cele păstrate în registre. Simularea evaluează în fiecare ciclu de ceas funcțiile logice ale căror valori sunt înscrise în registre.

Simularea la nivel de circuite se referă la analiza funcționării unor modele ale circuitelor reprezentate prin interconectarea unor dispozitive electronice cum sunt tranzistoare, rezistențe şi condensatoare. Această simulare constă în calcularea nivelelor de tensiune în funcție de timp din toate nodurile circuitului sau o parte a nodurilor. Aceasta presupune formularea şi rezolvarea unui mare număr de ecuații diferențiale, necesitățile de memorie şi de timp de calcul fiind ridicate.

Simularea temporală constă în simularea funcționării circuitului după determinarea întârzierii reale a semnalelor pe diferitele căi de date. Această simulare se utilizează în special pentru circuitele programabile, la care întârzierile semnalelor nu vor fi cunoscute decât în urma implementării circuitului, atunci când se poate determina numărul de conexiuni programabile utilizate pentru rutarea semnalelor. Informațiile despre întârzierile semnalelor sunt furnizate simulatorului printr-o operație numită *adnotare inversă*. Simularea temporală permite analiza funcționării în condiții reale a circuitului şi determinarea frecvenței maxime de funcționare a acestuia.

1.3. Funcționarea unui simulator

Un program de simulare necesită două tipuri de intrări: un fişier cu descrierea circuitului care trebuie simulat şi un set de stimuli care definesc toate semnalele de intrare pe durata timpului de simulare. Circuitul poate fi descris printr-o listă de conexiuni sau printr-un limbaj de descriere hardware. În cazul în care circuitul este specificat prin scheme sau diagrame de stare, trebuie creată mai întâi o descriere echivalentă a circuitului sub forma unei liste de conexiuni. Această descriere poate fi într-un format specific listelor de conexiuni (de exemplu, EDIF – *Electronic Design Interchange Format*), sau poate fi o descriere structurală a circuitului într-un limbaj de descriere hardware).

Proiectantul trebuie să specifice setul de stimuli care va fi aplicat la intrările circuitului pe durata simulării acestuia. Opțional, proiectantul poate specifica şi semnalele de ieşire care trebuie generate de circuit pentru fiecare stimul. Pentru definirea stimulilor, sistemele CAD sau simulatoarele pun la dispoziție diferite interfețe, cele mai utilizate fiind interfețele grafice, interfețele bazate pe fişiere text şi cele la nivelul liniei de comandă. Interfețele grafice permit specificarea valorii semnalelor aplicate la intrările circuitului sub forma unor diagrame de timp. Aceste interfețe sunt utile atunci când trebuie definit un număr redus de intrări. Atunci când există un număr mai mare de semnale de intrare, sau trebuie definit un număr mare de tranziții ale acestor semnale, este mai utilă specificarea într-un fişier text a unor comenzi care definesc valorile semnalelor de intrare şi

momentele de timp în care semnalele îşi modifică valoarea. Pot exista şi comenzi care specifică execuția simulării pentru un anumit timp. Introducerea comenzilor de simulare prin linia de comandă este utilă atunci când trebuie modificată valorile unor semnale care au fost definite anterior prin alte metode.

Simulatoarele utilizate în proiectarea asistată de calculator sunt simulatoare bazate pe evenimente. Aceste simulatoare împart intervalul de timp pentru care se realizează simularea în intervale de timp cu durate foarte reduse, un asemenea interval fiind numit *pasul simulării*. Acest pas este exprimat printr-un multiplu întreg al unei unități de timp care se numeşte *limita de rezoluție*. Simulatorul nu poate măsura intervale de timp mai reduse decât această limită. Pentru simularea la nivel de porți sau la nivel RTL, limita de rezoluție poate fi, de exemplu, de 1 picosecundă (ps). După trecerea unui interval de timp egal cu pasul simulării, simulatorul determină toate semnalele ale căror valori au fost modificate pe durata ultimului pas al simulării, o asemenea modificare fiind numită *eveniment*. Pentru fiecare semnal de intrare care s-a modificat, simulatorul reevaluează modelul circuitului şi determină noile valori ale semnalelor care sunt afectate de această modificare. Această reevaluare determină modificarea altor semnale şi generarea altor evenimente.

Un semnal nu poate fi actualizat în acelaşi timp cu un alt semnal din care este generat, deoarece un semnal nu îşi pot modifica valoarea instantaneu. De aceea, modificarea valorii unui semnal este planificată de simulator pentru un moment de timp ulterior față de timpul curent de simulare (T_c), după o întârziere numită *întârziere delta* (δ). Deci, modificarea valorii semnalului va fi executată la momentul de timp $T_{c+\delta}$.

La modificarea valorii unui semnal, se execută un *ciclu delta* în care se modifică valorile tuturor semnalelor care depind de primul semnal. Dacă semnalele modificate afectează alte semnale, vor fi planificate alte evenimente pentru timpul curent de simulare, la momentele de timp $T_{c+2\delta}$, $T_{c+3\delta}$, ... La aceste momente vor fi executate alte cicluri delta pentru actualizarea valorii tuturor semnalelor. Toate aceste cicluri delta se execută pentru acelaşi moment de simulare (T_c), astfel încât timpul de simulare nu avansează decât atunci când nu mai sunt alte evenimente planificate pentru timpul curent de simulare şi toate semnalele au fost actualizate. Deci, întârzierea delta este de fapt o întârziere zero, doar în mod conceptual fiind considerată ca o întârziere infinitezimală, fiind introdusă pentru a indica succesiunea operațiilor executate de simulator pentru actualizarea valorii semnalelor.

2. Exemplu de simulare

În această secțiune se prezintă etapele care trebuie executate pentru simularea funcționării unui numărător simplu de 4 biți, care numără în sens direct (de la 0000 la 1111).

2.1. Crearea fişierului sursă pentru numărător

Descrierea în limbajul VHDL a numărătorului este prezentată mai jos. Pentru actualizarea valorii numărătorului se utilizează o funcție de incrementare a unui vector de biți.

```
library ieee;
use ieee.std_logic_1164.all;

entity num4 is
   port (clk : in std_logic;
         rst : in std_logic;
         num : buffer std_logic_vector(3 downto 0));
end num4;

architecture simul of num4 is

   function inc_bv (a: std_logic_vector) return std_logic_vector is
      variable r: std_logic_vector (a'range);
      variable c: std_logic;
   begin
      c := '1';
      for i in a'range loop
         r(i) := a(i) xor c;
```

```
         c    := a(i) and c;
      end loop;
      return r;
   end inc_bv;

begin
   process (clk, rst)
   begin
      if (rst = '1') then
         num <= (others => '0');
      elsif (clk'event and clk = '1') then
         num <= inc_bv (num);
      end if;
   end process;
end simul;
```

Creați un fișier text cu extensia `.vhd` (de exemplu, cu programul *Notepad*). Copiați descrierea VHDL a numărătorului în acest fișier, iar apoi salvați fișierul.

2.2. Crearea proiectului

Pentru crearea unui nou proiect, executați următoarele operații:

1. Lansați în execuție programul *Project Navigator* din cadrul sistemului de proiectare Xilinx ISE (*Start* → *All Programs* → *Xilinx ISE 9.2i* → *Project Navigator*). În ecranul acestui program, selectați comanda *File* → *New Project*. Va fi afișată fereastra de dialog *Create New Project*. În câmpul *Project Location* din această fereastră, selectați directorul în care se va crea proiectul (un subdirector din directorul `D:\Student\`). În câmpul *Project Name* introduceți numele proiectului, de exemplu, **labssc1_1**. Selectați apoi butonul *Next*.
2. În fereastra de dialog *Device Properties*, selectați tipul circuitului FPGA pentru care se creează proiectul. În câmpul *Product Category* selectați **All**. În câmpul *Family* selectați **Spartan3**, în câmpul *Device* selectați **XC3S200**, în câmpul *Package* selectați **FT256**, iar în câmpul *Speed* selectați **–4**. În câmpul *Simulator* selectați **Modelsim-XE VHDL**. Selectați apoi butonul *Next*.
3. În fereastra de dialog *Create New Source*, selectați butonul *Next*.
4. În fereastra de dialog *Add Existing Sources*, selectați butonul *Next*.
5. În fereastra de dialog *Project Summary*, selectați butonul *Finish* pentru a crea proiectul.
6. În ecranul *Project Navigator*, selectați comanda *Project* → *Add Copy of Source* pentru adăugarea la proiect a fișierului cu descrierea numărătorului. În fereastra *Add Copies of Existing Sources* selectați fișierul sursă creat anterior, după care selectați butonul *Open*.
7. În fereastra de dialog *Adding Source Files*, păstrați asocierea implicită pentru unitățile de proiectare, și anume, *Synthesis/Imp + Simulation*. Selectați apoi butonul *OK*, prin care se va afișa un sumar al proiectului.

2.3. Compilarea fișierului sursă

Pentru compilarea fișierului sursă al numărătorului, executați următoarele operații:

1. În fereastra *Sources for* din partea stânga sus a ecranului *Project Navigator*, selectați opțiunea *Synthesis/Implementation* pentru afișarea proceselor corespunzătoare etapelor de sinteză și implementare. Procesele afișate vor permite și compilarea fișierului sursă.
2. În fereastra *Sources for*, executați un clic dublu pe numele fișierului sursă pentru afișarea conținutului acestui fișier în fereastra de editare.

3. În fereastra *Processes*, expandați lista proceselor de sinteză (*Synthesize – XST*) și executați un clic dublu pe procesul *Check Syntax* pentru compilarea fișierului sursă. În cazul în care sunt erori de compilare, corectați aceste erori și recompilați fișierul.

2.4. Crearea unui fișier pentru bancul de test

Pentru crearea unui fișier care va conține vectorii de test, fișier care va fi utilizat apoi pentru crearea unui banc de test, procedați astfel:

1. În ecranul *Project Navigator*, selectați comanda *Project* → *New Source* pentru crearea unui nou fișier. În fereastra de dialog *Select Source Type*, selectați *Test Bench WaveForm* ca tip al noului fișier. În câmpul *File Name* introduceți numele fișierului care va fi creat. Acest nume trebuie să fie diferit de numele fișierului sursă VHDL. De exemplu, dacă numele fișierului sursă este **num**, numele fișierului pentru bancul de test poate fi **num_tb**. Selectați apoi butonul *Next*.
2. În fereastra de dialog *Associate Source*, păstrați asocierea implicită cu entitatea numărătorului (**num4**), iar apoi selectați butonul *Next*.
3. În fereastra de dialog *Summary*, selectați butonul *Finish*. Prin aceste operații, se va lansa în execuție programul *HDL Bencher* și se va afișa fereastra *Initialize Timing*.
4. În fereastra *Initialize Timing*, se pot seta parametrii de timp ai semnalului de ceas, ai semnalelor de intrare și ai celor de ieșire. Pentru acești parametri, păstrați valorile implicite. Aceste valori sunt ilustrate în Figura 1. Deselectați opțiunea *GSR (FPGA)* din zona *Global Signals*. Selectați apoi butonul *Finish*, prin care se va afișa fereastra programului *HDL Bencher*, conținând semnalele de intrare și de ieșire ale numărătorului. Semnalul de ceas *clk* este inițializat în mod automat conform parametrilor care au fost setați în fereastra *Initialize Timing*.

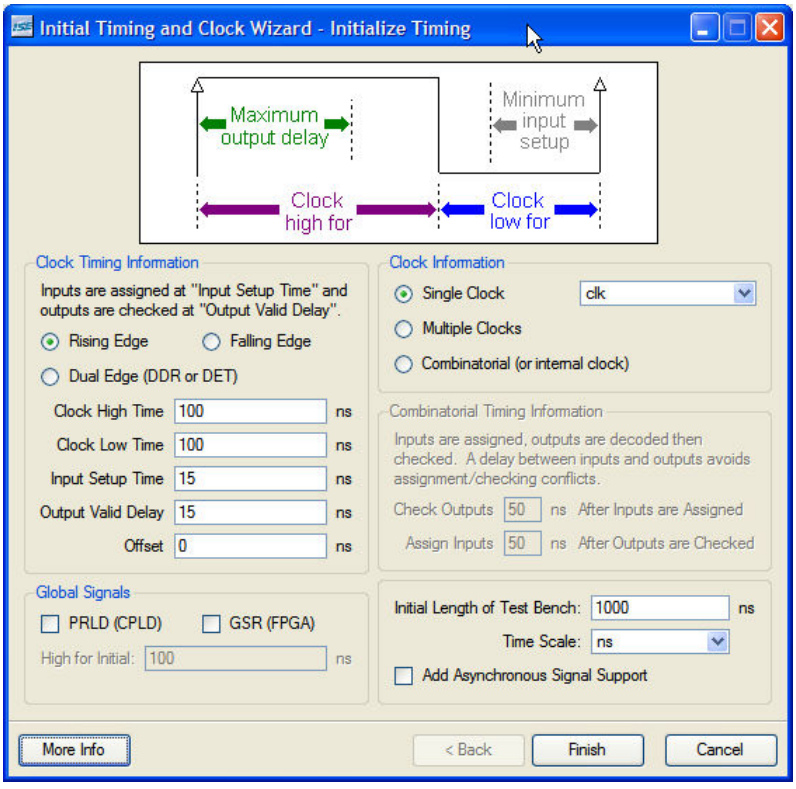

Figura 1. Fereastra *Initialize Timing* a programului *HDL Bencher*.

5. Pentru inițializarea semnalului *rst*, setați acest semnal la valoarea 1 la timpul de simulare 0 ns, iar apoi setați semnalul la valoarea 0 la timpul de simulare 100 ns. Pentru aceasta, executați un clic în zona albă din stânga primei celule albastre din linia semnalului *rst*, iar apoi executați un clic în interiorul primei celule albastre (cea corespunzătoare timpului de simulare de 100 ns).
6. Setați timpul de sfârşit al bancului de test la 3200 ns. Pentru aceasta, selectați comanda *Test Bench* → *Set End of Test Bench*, prin care se va afişa caseta de dialog *Set End of Test Bench*. În câmpul *Test Bench Ends* introduceți valoarea 3200, după care selectați butonul *OK*. În acest moment, fereastra *HDL Bencher* arată ca în Figura 2.

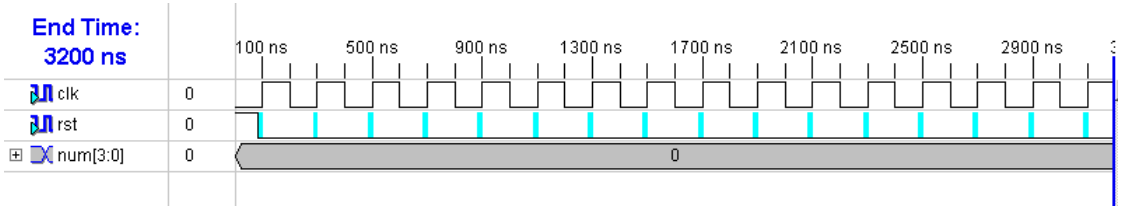

Figura 2. Fereastra *HDL Bencher* cu formele de undă ale semnalelor de intrare.

7. Salvați fişierul cu bancul de test prin comanda *File* → *Save* sau cu butonul *Save* .
8. Închideți fereastra programului *HDL Bencher*.

2.5. Simularea funcțională

Pentru simularea funcțională a numărătorului, procedați astfel:

1. În fereastra *Sources for* din ecranul *Project Navigator*, selectați opțiunea *Behavioral Simulation* pentru afişarea proceselor corespunzătoare etapei de simulare funcțională.
2. În aceeaşi fereastră, selectați fişierul cu extensia **.tbw** care a fost creat anterior cu programul *HDL Bencher*.
3. În fereastra *Processes*, executați un clic pe semnul + din stânga liniei *ModelSim Simulator* pentru afişarea proceselor disponibile.
4. Executați un clic dublu pe linia *Simulate Behavioral Model*. Prin aceasta, programul *Project Navigator* va crea un fişier de comenzi cu extensia **.fdo**, conținând comenzile care trebuie executate de simulatorul Model*Sim*, după care va lansa în execuție simulatorul. Acesta va executa simularea utilizând vectorii de test specificați în bancul de test, iar apoi va afişa rezultatele simulării în fereastra cu forma de undă a semnalelor (*Wave*). Simulatorul va deschide de asemenea fereastra spațiului de lucru (*Workspace*) şi fereastra semnalelor (*Objects*).
5. Maximizați fereastra *Wave* prin execuția unui clic pe butonul + al acestei ferestre.
6. În fereastra *Wave*, executați un clic pe butonul *Zoom Full* pentru vizualizarea formei de undă complete a semnalelor.
7. În aceeaşi fereastră, observați că secvența de numărare obținută, indicată prin succesiunea valorilor binare ale semnalului *num*, este 0, 8, 4, 12, 2, ..., secvență care nu este corectă (Figura 3). Se mai poate observa că valorile binare ale vectorului *num* corespund secvenței corecte de numărare în cazul în care biții sunt citiți începând cu cel mai puțin semnificativ (în ordine inversă).
8. Încheiați sesiunea de simulare prin închiderea ferestrei principale a simulatorului sau prin execuția comenzii *File* → *Quit*.

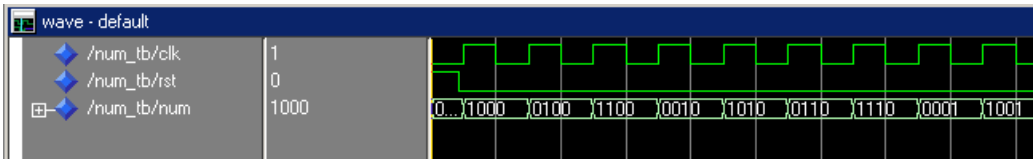

Figura 3. Afişarea rezultatelor simulării în fereastra *Wave* a simulatorului Model*Sim*.

2.6. Corectarea erorii

Pentru corectarea descrierii numărătorului, reveniți în ecranul *Project Navigator* şi deschideți pentru editare fişierul sursă VHDL. Se poate observa că în funcția `inc_bv` vectorul de biți transmis ca parametru este prelucrat în ordinea corespunzătoare domeniului vectorului (`a'range`). Pentru incrementarea corectă a vectorului, bucla `for .. loop` trebuie parcursă începând cu bitul cel mai puțin semnificativ al vectorului. Deci, eroarea se poate corecta prin modificarea următoarei linii sursă

```
for i in a'range loop
```

astfel:

```
for i in a'low to a'high loop
```

Corectați fişierul sursă şi salvați fişierul prin comanda *File* → *Save* sau cu butonul *Save* .

2.7. Reexecutarea simulării funcționale

Pentru reexecutarea simulării funcționale a numărătorului, executați următoarele operații:

1. Compilați din nou fişierul sursă în modul descris în secțiunea 2.3 şi corectați eventualele erori.
2. În fereastra *Sources for* din ecranul *Project Navigator*, selectați opțiunea *Behavioral Simulation*.
3. Lansați în execuție simulatorul Model*Sim* executând un clic dublu pe linia *Simulate Behavioral Model* din fereastra *Processes*.
4. Maximizați fereastra *Wave* a simulatorului, iar apoi executați un clic pe butonul *Zoom Full* .
5. Selectați semnalul *num*. Executați un clic cu butonul din dreapta pe acest semnal şi selectați opțiunea *Radix*, iar apoi opțiunea *Hexadecimal*.
6. Se poate observa că secvența de numărare este acum cea corectă.
7. Închideți fereastra principală a simulatorului Model*Sim*.

3. Aplicații

Răspundeți la următoarele întrebări:

a. Ce este un vector de test şi cum se poate reprezenta acesta?

b. Care sunt avantajele simulării asistate de calculator?

c. Care sunt principalele tipuri de simulare?

d. Ce intrări sunt necesare pentru un simulator şi ce metode pot fi utilizate pentru specificarea acestor intrări?