Supliment de laborator

Construirea testbench-urilor în limbajul Verilog

1. Obiective

O1. Construirea unităților testbench pentru verificarea modulelor Verilog

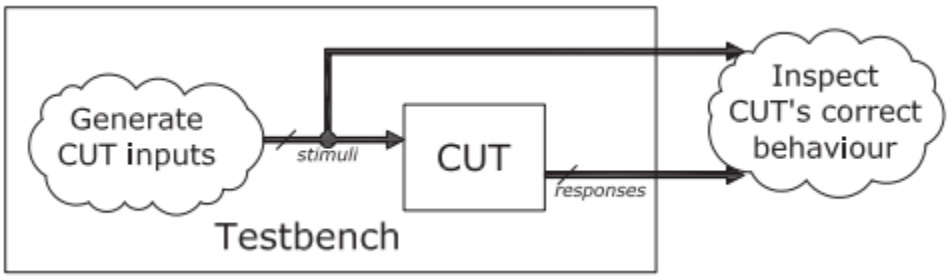
Abordarea testbench-ului

Modulul Verilog implementat, a cărui corectitudine trebuie verificată, este cunoscut și sub numele de **Circuit Under Test (CUT)**, sau **Device Under Test (DUT)**.

Fișierul **testbench**:

- evaluează corectitudinea CUT-ului în scopul simulării acestuia
- generează vectorii de intrare pentru CUT
- analizează ieșirile CUT-ului (autonom sau manual)
- furnizează informație textuală la testele reușite/eronate (opțional)

Diagrama de mai jos expune formele de undă intrare-ieșire corespunzătoare modului de inspectare a corectitudinii unui CUT.



Testbench-ul instanțiază CUT-ul, generează intrările ei și dirijează toate porturile CUT-ului către exterior, pentru inspecție amănunțită.

Notă: Fiecare stimul va fi generat într-un bloc dedicat **`initial.`** Semnalele care transportă stimulii la intrările CUT sunt numite la fel ca intrările la care se conectează. Aceste semnale sunt declarate ***de tip registru ieșire***. Semnalele care transportă ieșirile CUT-urilor la exterior sunt declarate de tip ***output***.

Generarea intrării de ceas pentru un CUT

Următorul cod generează un semnal de ceas, cu o perioadă de 100ns și factor de umplere de 50%, pornind de la nivelul logic 0:

```
reg clk;
initial begin
  clk = 1'd0;
  forever #50 clk = ~clk;
end
```

Semnalul ***clk*** se comută la fiecare jumătate de ciclu, folosind specificatorul de întârziere **# 50**.

Pentru limitarea unui tact care rulează nedefinit, simbolul **$** de finalizare a task-ului poate fi folosit pentru a forța terminarea simulării. Fragmentul de cod de mai jos sfârșește testbench-ul după 2000ns:

```
initial begin
#2000 $finish;
end
```

Codul de mai jos generează 100 de cicluri de ceas pornind de la nivelul logic 1, fiecare ciclu având o perioadă de 150 ns și un factor de umplere de 50%:

```
initial begin
clk = 1'd1;
repeat (200) #75 clk = ~clk;
end
```

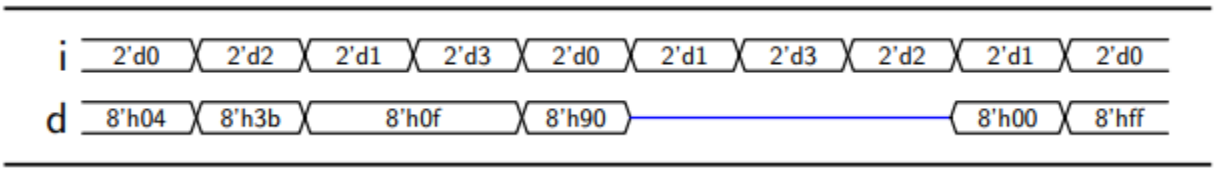
Semnalul ***clk*** este comutat de 200 de ori pentru generarea a 100 de cicluri folosind constructul **`repeat`** (200).

Fragmentul de mai jos generează un semnal de resetare, activ la nivel 0, afirmat din momentul inițial pentru **2ns**:

```
initial begin
rst_b = 1'd0;
#2 rst_b = 1'd1;
end
```

Semnalul de resetare **rst_b** este stabilit la nivel logic 0 și, după 2ns, folosind specificatorul de întârziere **# 2**, semnalul este dezactivat prin setarea acestuia la nivel logic 1.

Să considerăm un CUT cu 2 intrări**, i**, pe 2 biți și **d,** pe 8 biți și procesul de testare pentru a genera intrările ca în plaja de timp din diagrama de mai jos:

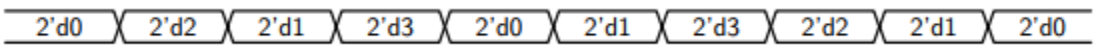


Deoarece nu este dată nici o unitate de timp, pentru scurtcircuit, o durată de **10ns** este luată în considerare pentru fiecare configurație pe intrarea **i**. Deoarece intrarea **d** se modifică în mod sincron cu **i**, se va schimba la momente multiple de 10 ns.

Fiecare din cele două semnale din diagrama de mai sus va fi generat în propriul său bloc inițial.

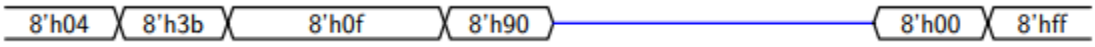
Notă: Intrarea **d** este în impedanță ridicată de la 50ns la 80ns, marcate în diagrama de sincronizare cu o linie de înălțime medie.

Codul de pe pagina următoare furnizează stimuli la intrarea i, la fiecare 10ns (vom considera diagrama suport):



```
initial begin
i = 2'd0; // valoarea lui i la momentul 0 ns
#10 i = 2'd2; // valoarea lui i la momentul 10 ns
#10 i = 2'd1; // valoarea lui i la momentul 20 ns
#10 i = 2'd3; // valoarea lui i la momentul 30 ns
#10 i = 2'd0; // valoarea lui i la momentul 40 ns
#10 i = 2'd1; // valoarea lui i la momentul 50 ns
#10 i = 2'd3; // valoarea lui i la momentul 60 ns
#10 i = 2'd2; // valoarea lui i la momentul 70 ns
#10 i = 2'd1; // valoarea lui i la momentul 80 ns
#10 i = 2'd0; // valoarea lui i la momentul 90 ns
end
```

Astfel, a mai rămas să generăm stimuli pe linia de intrare **d.**



Codul de mai jos va realiza această sarcină:

```
initial begin
d = 8'h04; // valoarea lui d la momentul 0 ns
#10 d = 8'd3b; // valoarea lui d la momentul 10 ns
#10 d = 8'd0f; // valoarea lui d la momentul 20 ns
#20 d = 8'd90; // valoarea lui d la momentul 30 ns
#10 d = 8'dz; // impedanță ridicată de la momentul 50 ns
#30 d = 8'd00; // valoarea lui d la momentul 80 ns
#10 d = 8'dff; // valoarea lui d la momentul 90 ns
end
```

Luați în considerare un CUT cu o singură intrare, x, pe 5 biți. Fragmentul de mai jos generează toate configurațiile posibile de intrare, fiecare vector de intrare fiind stabil pentru 20 ns:

```
integer i;
initial begin
x = 5'd0;
for (i = 0; i < 32; i = i + 1)
#20 x = i[4:0];
end
```

Semnalului x îi este atribuit valoarea întregului conținând cei mai puțin semnificativi 5 biți, după o întârziere corespunzătoare. Acest tip de testbench se mai numește ***verificare exhaustivă***.

2. Referințe bibliografice

[Latt99] L. Semiconductor. A Verilog HDL Test Bench Primer. [Online]. Disponibil: Lattice - A Verilog HDL Test Bench Primer