

Testbench-uri în Verilog

Oprițoiu Flavius
flavius.opritoiu@cs.upt.ro

16 octombrie 2024

Introducere

Obiective:

- ▶ Construirea unităților testbench pentru verificarea modulelor Verilog

De citit:

- ❗ Lattice Semiconductor: "A Verilog HDL Test Bench Primer", Note de aplicație, [Latt99]

Modulele Verilog care vor fi testate sunt referite ca **Circuit Under Test (CUT)**.

Testbench-ul:

- Generează vectorii de intrare pentru CUT
- Analizează ieșirile CUT-ului
- Oferă detalii textuale privind rezultatul testului

Abordarea testbench-ului

Metoda de construire a testbench-urilor:

- ▶ pentru fiecare intrare a CUT, se prevede în testbench un semnal *reg* având același nume și aceeași lățime
- ▶ pentru fiecare ieșire a CUT, se prevede în testbench un semnal *wire* având același nume și aceeași lățime
- ▶ se instanțiază modulul CUT, conectând fiecare port al lui la semnalele corespunzătoare definite mai sus
- ▶ sunt generate intrările pentru CUT

Pentru generarea intrărilor, se pot folosi oricare din tiparele descrise în continuare.

Generarea intrării de ceas pentru un CUT

Generarea unui semnal de ceas cu factor de umplere de 50%, de perioadă dată:

```
localparam CLK_PERIOD = 100;
reg clk;
initial begin
    clk = 1'd0;
    forever #(CLK_PERIOD/2) clk = ~clk;
end
```

Important: Semnalul *clk* construit mai sus este generat indefinit, determinând simularea să ruleze la nesfârșit!

Generarea intrării de ceas pentru un CUT (contin.)

Generarea unui semnal de ceas cu factor de umplere de 50%, de perioadă dată, rulând pentru un număr specificat de cicluri de tact:

```
localparam CLK_PERIOD = 100;
localparam RUNNING_CYCLES = 50;
reg clk;
initial begin
    clk = 1'd0;
    repeat (2*RUNNING_CYCLES) #(CLK_PERIOD/2) clk = ~clk;
end
```

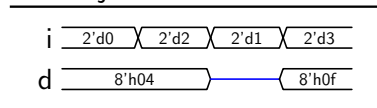
Generarea intrării de reset pentru un CUT

Generarea unui semnal de reset, activ la 0, activat la momentul inițial, pentru o durată de timp specificată:

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

Generarea unor forme de undă la intrările CUT

Se consideră un CUT cu 2 intrări, i , pe 2 biți și d , pe 8 biți și un proces de test care generează cele 2 intrări ca în diagrama de timp de mai jos:




Nefiind precizată o unitate de timp, pentru concizie, se consideră o durată de 10 unități de timp alocată fiecărei configurații de la intrarea i . Pentru că intrarea d se modifică sincron cu i , modificarea ei se face la momente de timp multipli de 10 unități de timp.

Fiecare intrare va fi generată în propriul bloc `initial`.

Notă: Linia albastră, de semi-înălțime din diagrama intrării d semnifică faptul că semnalul nu are niciun driver: se află în impedanță ridicată.

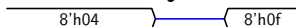
Generarea unor forme de undă la intrările CUT (contin.)

Codul de mai jos furnizează stimulii la intrarea i , la distanță de 10 unități de timp: 

```
initial begin
    i = 2'd0;      //valoarea lui i la 0
    #10 i = 2'd2;  //valoarea lui i la 10
    #10 i = 2'd1;  //valoarea lui i la 20
    #10 i = 2'd3;  //valoarea lui i la 30
end
```


Generarea unor forme de undă la intrările CUT (contin.)

Codul de mai jos furnizează stimulii la intrarea *d*:



```
initial begin
    d = 8'h04;      //valoarea lui d la 0
    #20 d = 8'dz;   //valoarea lui d la 20
    #10 d = 8'd0f;  //valoarea lui d la 30
end
```

Generarea exhaustivă a intrărilor CUT

Se consideră un CUT cu 3 intrări: o intrare x pe 2 biți, o intrare d pe 4 biți și o intrare en pe un singur bit.

Fragmentul de mai jos generează toate cele 128 de configurațiile de intrare posibile (2^7), la distanță de 20 unități de timp, fiecare:

```
integer i;  
initial begin  
    {x, d, en} = 0;  
    for (i = 1; i < 128; i = i + 1)  
        #20 {x, d, en} = i;  
    #20;  
end
```

Studiu de caz

Exercițiu: Construiți testbench-ul pentru verificare exhaustivă a unui decodor 2-la-4, cu intrare de enable și ieșiri active la 0, a cărei implementare este disponibilă [▶ aici](#) (slide 12).

Soluție:

```
1  module dec_2x4_tb ;
2      reg [1:0] s ;
3      reg e ;
4      wire [3:0] y ;

6      dec_2x4 cut (
7          .s(s) ,
8          .e(e) ,
9          .y(y)
10     ) ;

14     integer i ;
15     initial begin
16         {s , e} = 0 ;
17         for ( i=1; i<8; i=i+1)
18             #20 s = i ;
19         #20 ;
20     end
21 endmodule
```

Simularea testbench-ului în Modelsim

Se descarcă scriptul cu opțiuni de configurare *run.txt* de [aici](#) și este adaptat conținutul său proiectului curent:

- ▶ sunt adăugate fișierele sursă Verilog, separate prin spațiu, la lista *sourcefiles* din linia 5
- ▶ este modificat numele modulului top indicat de variabila *topmodule* din linia 10; în mod tipic, acest nume este numele modulului testbench (iar **nu** numele unui fișier Verilog)
- ▶ este executat scriptul cu comanda `do run.txt`
- ▶ sunt folosite oricare din comenzile specifice Modelsim pentru simulare

Comenzi Modelsim pentru simulare

`add wave *`

adaugă semnalele modulului top în fereastra formelor de unde pentru inspectarea vizuală a semnalelor

`run -all`

rulează simularea la nesfârșit, sau până când niciun semnal nu își mai modifică valoarea

`run 600`

rulează simularea timp de 600 unități de timp

`restart`

repornește simularea de la momentul 0

`quit -sim`

descarcă modulul simulat curent fără a părăsi mediul Modelsim

`do run.txt`

recompilează și repornește simularea

Referințe bibliografice

[Latt99] L. Semiconductor. A verilog hdl test bench primer.
[Online]. Available: <https://people.ece.cornell.edu/land/courses/ece5760/Verilog/LatticeTestbenchPrimer.pdf> (Last accessed 17/04/2016).