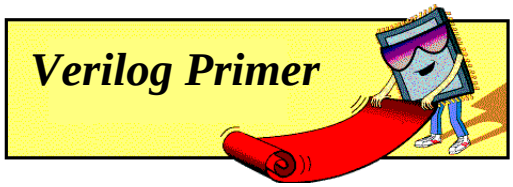
Verilog Primer



© Sital Technology Ltd. All Rights Reserved

Verilog Primer

This book provides fast-track training in the Verilog Hardware Description Language. It is targeted mainly for experienced VHDL designers who want to learn the SystemVerilog course and build a Verification Environment using SystemVerilog for their VHDL design. This book is not a replacement to the Basic Verilog course for those who want to design in Verilog. If you have not used either VHDL or Verilog, you should attend the Basic Verilog course.

Copyright © 2005, Sital Technology Ltd. All Rights Reserved.
No part of this publication may be reproduced, translated, stored in a retrieval system, or transmitted, in any form or by any means, electronic, photocopying, recording or otherwise, without a prior written permission of Sital Technology Ltd.

Sital Technology Ltd, www.sital.co.il, Tel: 09-7633300, Fax: 09-7663394

module

- **כל תכנון**, החל מרכיב בודד, רכיב המכיל מספר מרכיבים וכלה בסביבת בדיקה - **מוגדר כ - module**.
- module מתאר את ממשק התכנון לעולם החיצון.
- module מכיל מימושי החומרה או תהליכי הבדיקה בתוכנית.
- module יכול להיבנות מ- module מהיררכיה נמוכה ממנו.

module

```
module module_name (Port1, Port2, Port3, Port4);

    // PORT LIST
        input Port1, Port2;   // bit inputs
        output [3:0] Port3;   // vector outputs
        inout Port4;


    // DATA TYPES

    // MODULE INSTANCES

    // CONTINUOS ASSIGNMENT

    // PROCEDURAL BLOCKS

    // TASKS AND FUNCTIONS

endmodule
```

- בגוף ה-module ניתן לכתוב:
  - תהליכים התנהגותיים.
  - module Instantiation.
  - Data Flow statements.

module – port list

```
module module_name (port_name_1,port_name_2...);
     // PORT LIST
     mode      port_name_1;
     mode[x:y] port_name_2;

endmodule
```

module_name - שם ה module.

port_name - שם של רגל מוצא/כניסה.

mode - כיוון ה- PORT : input , output, inout.

```
module comparator (a,b,a_bigger_b);
     // port list
     input[7:0]     a,b;
     output         a_bigger_b;
     // data types

     reg a_bigger_b;

     always @(a or b)
     begin
          if (a > b)
               a_bigger_b = 1;
          else a_bigger_b = 0;
     end
endmodule
```

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

module - parameters

- פורמט הצהרת פרמטר:

```
parameter <parameter name> = <default value>
```

- לפרמטר יש ערך default ולכן ניתן לסנתז את ה-module הפרמטרי.

```
module comparator_param (a,b,a_bigger_b);
     //parameter declaration
     parameter      val=7;

     // port list
     input[val:0]        a,b;
     output              a_bigger_b;

     // data types
     reg a_bigger_b;

     always @(a or b)
     begin
          if (a > b)
               a_bigger_b = 1;
          else a_bigger_b = 0;
     end

endmodule
```

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

module - instance

- Instance לפי סדר הצהרת ה - Port ב - module:

```
<module name> <instance_number> (<higher level port or line>
,<higher level port or line>,<higher level port or line> );
```

- Instance לפי שם Port:

```
<module name> <instance_number>
               .<module_port>(<higher level port or line>),
               .<module_port>(<higher level port or line>),

               .<module_port>(<higher level port or line>));
```

- בהתחברות היררכית ל - module ניתן לאלץ פרמטר בעל ערך שונה ע"י פקודת defparam הניתנת **לאחר** ההצהרה על ה-Instance.
- מבנה הפקודה:

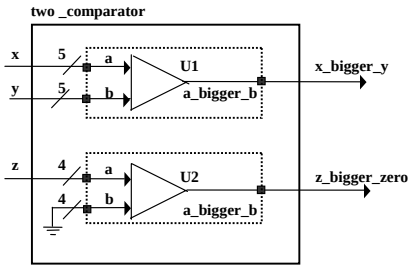
```
defparam <instance name>.<parameter name>=<new value>
```

- האופרטור # מאלץ פרמטרים **בהצהרה** על ה-Instance.

```
<Module_name> #(<new value>) <instance name> <ports>
```

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

Instance example




```
module two_comparator (x,y,z,x_bigger_y,z_bigger_zero);
     // potr list
     input[4:0]     x,y;
     input[3:0]     z;
     output         x_bigger_y,z_bigger_zero;
     // data types

     // comparator instance
     comparator_param u1(.a(x),
                         .b(y),
                         .a_bigger_b(x_bigger_y));
     defparam u1.val=4;

     comparator_param #(3)u2 (z,4'b0,z_bigger_zero);

endmodule
```

- Instantiation ניתן לבצע **לפי סדר** - U2 או **לפי שם** - U1.
- Parameter override ניתן לבצע בעזרת defparam או בעזרת #.

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

VALUE SET

- קו ב – Verilog יכול לקבל אחד מ- 4 ערכים :

0 = logic zero, false condition

1 = logic one, true condition

Z = high impedance

X = unknown value (ערך default לכל קו)

- ב – Verilog יש 2 סוגים של קוים : REG, NET.

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

NET

- NET הוא קו שאינו מחזיק ערך.
- NET נדחף ע"י port חיצוני או ע"י module פנימי.
- פורמט הצהרה על NET :

```
<net_type> <[range L:R]> <(strength)> <net name>;
```

- NET מכיל קבוצה של types שונים :
  - wire, tri
  - supply0, supply1
  - wor, trior
  - wand, triand
  - trireg
  - tri0, tri1
- NET יכול להיות vector של סיביות או ביט בודד.

  ב - default סיבית אחת וערך X.

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

STRENGTH

- לכל NET ניתן להגדיר strength.
- בעזרת strength ניתן לשלוט על תוצאת ה-resolve בין 2 קוים.
- רמות strength :

Strength level	degree
Supply	strongest
Strong	↑
Pull	
Large *	
Weak	
Medium *	
Small *	
highZ	weakest

* לקווים המחזיקים מטען.

- רמות strength שימושיות בהגדרת NET המחובר לפרימיטיבים ב - SWITCH, STRUCTURAL LEVEL.

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

סוגי NET

- סוגי NET שימושיים : - wire לשימוש NET כללי.

  - tri ל - Bus דו כיווני המקבל ערכי Z .
- שימושי wor wand:

```
module wor_try (a,b,out);
    //port list
    input a,b;
    output out;

    //data type
    wire a,b;
    wor out;

    assign out=a|b;
endmodule
```

בדוגמא הנ"ל יסונתז שער OR ב- Open drain Wired logic.

השימוש ב wor, wand מאפשר חיבור 2 מוצאים חוקי ללא אילוצי strength.

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

REG

- REG הוא קו המחזיק מידע.
- REG הוא ה- Data type השימושי ב-Procedural blocks.
- פורמט הצהרה על REG :

```
<reg_type> <[range L:R]>  <reg name>;
```

- פורמט הצהרה על REG מסוג ARRAY :

```
reg [msb:lsb] <memory_name> [first address: last address]
integer <array_name> [first address: last address]
```

לא ניתן לבצע ישירות פעולה על סיבית בודדת ב – ARRAY של ווקטורים.

- REG יכול להיות vector של סיביות.
  ב - default : סיבית אחת וערך x.

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

REG

- REG מכיל קבוצה של types שונים :
  - reg - וקטור unsigned.
  - integer - מספר signed ברוחב 32bit.
  - real - מספר floating point – signed.
  - time - מספר unsigned ברוחב 64bit. מאחסן את זמן הסימולציה.
  - realtime - מספר real מאחסן את זמן הסימולציה.

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

integer real & time

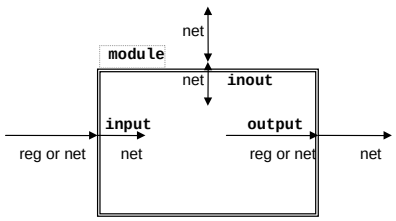
- ב-real ניתן לייצג מספר עשרוני (2.13) או מעריכי (4e3).
- בהצבת real ל-integer המספר מעוגל לערך הקרוב ביותר למספר השלם. בדוגמא ל – 2.
- הפונקציה $time מחזירה את זמן הסימולציה.

```
module reg_numbers;
     //port list
     //data types
     real delta1,delta2;
     integer    i;
     time simulation_time;

     initial
     begin
          delta1=4e3;
          delta2=2.13;
          i=delta2;
     end
     initial simulation_time=$time;
endmodule
```

* real ו - time לא ניתנים לסינתזה.

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

כללי חיבורי reg ו-wire



- port כניסה ל - module : חייב להתחבר לקו מסוג net.
  לכניסה יכולים להתחבר קווי reg או net.
- port יציאה וקוים פנימיים : או REG או NET.
  - reg כאשר ה - port נדחף ע"י בלוק.
  - net כאשר ה - port נדחף ע"י פקודת assign או מ - port של module פנימי.
  - ליציאת module יכול להתחבר רק קו net.
- inout : port דו-כיווני יכול להתחבר לקו מסוג net בכל צד.

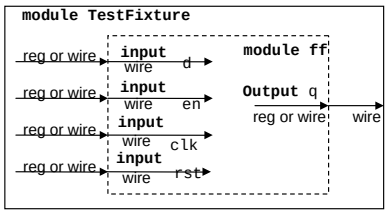
Verilog Primer - © Sital Technology Ltd. All Rights Reserved

כללי חיבורי reg ו-wire

- נתון module Flip Flop -

```
module ff (d,q,clk,rst,en);
```

- סביבת הבדיקה צריכה להתחבר באופן הבא:



- חיבורי ה - reg wire הנדרשים:

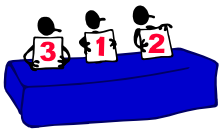
```
module TestFixture;

    reg d_Signal,clk_Signal,rst_Signal,en_Signal;
    wire q_Signal;

    ff U1 (  .d(d_Signal),
             .q(q_Signal),
             .clk(clk_Signal),
             .rst(rst_Signal),
             .en(en_Segnal));

   // Enter fixture code here
endmodule // TestFixture
```

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

ייצוג מספרים

- ב - Verilog ניתן לייצג מספרים בצורת integer או real.
- פורמט הצהרה על מספר: `<size>'<base><number>`
  - size - מספר עשרוני המציין את מספר הסיביות במספר
  - base - אפשרויות לייצוג בסיסים: b – בינארי, h – הקסהדצימלי, o – אוקטלי, d – עשרוני
  - number – מספר לפי הפורמט המוגדר בגודל ובבסיס.

אם לא מצוינים size ו-base, נקבל מספר עשרוני בגודל 32bit.



Verilog Primer - © Sital Technology Ltd. All Rights Reserved

ייצוג מספרים

דוגמאות:

Sized numbers :
4'b1010 // 4 bit binary number;
16'hf01d // 16 bit hexadecimal number;
12'O7071 // 12 bit octal number;
16'd123 // 16 bit decimal number;
8'b1100_0000 // equivalent of 8'b11000000

Unsized numbers :
450 // 32 bit decimal number;
'hed // 32 bit hexadecimal number;
'o45 // 32 bit octal number;

X,Z numbers :
12'hefx // 12 bit hexadecimal number, lower nibble is unknown;
16'bz // 16 bit high impedance;
7'hx // 7 bit hexadecimal unknown;
4'b10?? // equivalent of a 4'b10zz;

Negative numbers :
-8 // negative integer number
-8'd3 // 8 bit 2's complement of 3;

Real numbers :
4e3 // equals 4000;
4.56

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

Continuous Assignments

- פורמט הצהרה על Continuous assignment:

```
assign <net name[range L:R]>=<assignment, assignment>;
```

- דוגמאות:

```
// example 1 - OR gate
    wire line_out;
    assign line_out=a|b;

// example 2 – assignment with delay
    wire[7:0]    input_a;
    assign #30 input_a=10;

// example 3
    wire[7:0]    sig1,sig2;
    wire[7:0]    sig3;
    assign   sig3[3:0]=sig1[7:4] & sig2[3:0],
             sig3[7:4]=sig1[3:0] | sig2[7:4];
```

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

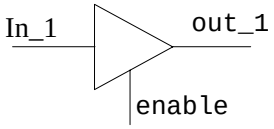
Conditional Continuous Assignments

- פורמט השמה רציפה בתלות בתנאי.

```
assign <net name>=(<condition>)?
    <true_expression>:<false_expression>;
```

- אם התנאי מתקיים, ה - NET מקבל את ערך ה - TRUE.
- דוגמאות:

```
// example1 – 3 state buffer
    assign out_1=(enable==1 ? in_1 : 1'bz);

// or:
    assign out_1=(!enable ? 1'bz : in_1);
```



always

- always חייב להיות מושהה ע"י **רשימת רגישויות** ו**או** ע"י **פקודות** #,@ או wait (סימולציה) בתוך התהליך.
- רשימת הרגישויות מתנה את הכניסה לתהליך בשינוי באחד מהאותות.
- לאחר כניסה, התהליך יתבצע עד סופו ויותנע מחדש רק עם שינוי נוסף באחד מהאותות שברשימת הרגישויות.
- always מותנע החל מרגע 0 של הסימולציה.
- מבנה always לסינתזה:

```
always @(<sensitivity list>)
    begin :<block name> //block name is optional

    <sequential statements>
    end
```

* begin ו-end נדרשים כאשר יש יותר מפעולה אחת בבלוק.

always

- דוגמא ל-always המכילים רשימת רגישויות:

```
//data type
reg[15:0]      out;

// procedural block
always @(posedge clk or negedge rst)
          if (!rst)
               out<=0;
          else
               if (en)
                    out<=in;
```

Synchronous Structure

```
////////////////////////////////////////////////////////////
```

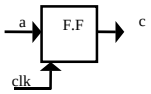
```
// data types
reg value_out;

// procedural block
always @(a or b or c or d)

/* all the inputs must be in the sensitivity list !! */
     begin
          if (a > b)
               value_out = c;
          else value_out = d;
     end
```

Combinatorial Structure

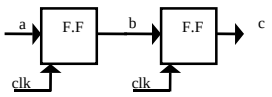
Blocking Assignments

- Blocking Assignment הנה השמה ל - REG.
  מבוצעת רק ב - Procedural block (always, initial).
- Blocking Assignment עוצרת את רצף הבלוק (always) ומבוצעת מיידית .
- השמת Blocking מבוצעת ב-DELTA בתוך ה-TICK ולא בסופו.
- דוגמא:

```
reg b,c;

always @(posedge clk)
     begin
          b=a;
          c=b;
     end
endmodule
```



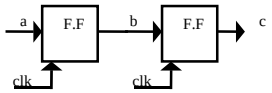

```
reg b,c;

always @(posedge clk)
     begin
          c=b;
          b=a;
     end
endmodule
```



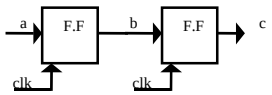
NON Blocking Assignments

- NON Blocking Assignment הנה השמה ל - REG.
  מבוצעת רק ב - Procedural block (always, initial).
- NON Blocking Assignment אינה עוצרת את רצף הבלוק (always) ומבוצעת בסופו.
- השמות NON Blocking מבוצעות בסוף ה - TICK אחת אחרי השניה.
- דוגמא :

```
reg b,c;

always @(posedge clk)
     begin
          b<=a;
          c<=b;
     end
endmodule
```




```
reg b,c;

always @(posedge clk)
     begin
          c<=b;
          b<=a;
     end
endmodule
```



* אין משמעות לסדר כתיבת ההשמות.

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

כתיבה לסימולציה

- בלוק לסימולציה : initial (always משמש גם בסימולציה)
- מבנה בלוק לסימולציה : - סדרתי = begin end
  - מקבילי = fork join
- **בלוק ללא רשימת רגישויות** מותנע באופן מידי ונעצר באחת מפקודות # , @, event או wait שבתוכו.
- דוגמא לתהליך ללא רשימת רגישויות :

```
// simulation only
initial
    clk1=0;

always
    #5 clk1=~clk1;

initial #1000 $stop; //The $stop function stops the
                     simulation.
                  //$finish finishes the simulation
```

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

כתיבה לסימולציה

- מבנה always לסימולציה :

• בלוק ללא רשימת רגישויות.

```
always // no sensitivity list
     begin
     <timing control>
     @ <statement>  // wait for event
     # <statement>  // wait for time (incremental)
     wait (<condition>)//level condition
     named event
     end
```

• בלוק סדרתי עם רשימת רגישויות.

```
always @(<sensitivity list>)
     begin:<block name> //block name is optional
     <sequential statements>
     <timing control also available>
     end
```

• בלוק מקבילי עם רשימת רגישויות.

```
always @(<sensitivity list>)
     fork:<block name> //block name is optional
     <parallel statements>
     <timing control also available>
     join
```

• begin ו-end נדרשים כאשר יש יותר מפעולה אחת בבלוק.

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

משפטי @ , #, wait, event

- מאפשרים למתכנן להשהות את הביצוע של הבלוק.
- Regular event control : `@(<edge_of_signal>)`

```
always
begin
    @(clk1) out=b;
    @(negedge b or posedge c) a=b;
```

הבלוק יושהה עד לשינוי באחד מהאותות שברשימה.

- Level sensitive : `wait(<signal>)`

```
wait(d);
```

הבלוק יושהה עד לקיום התנאי.
משפט ה - wait מתבצע עבור רמת הלוגיקה הראשונה המתאימה. בשינוי רמה נוסף לא מתקיימת השהיה נוספת.

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

משפטי @ , #, wait, event

- Named event :

* מגדירים event עבור תנאי.
* ה - event ישהה את פעול הבלוק עד לקיום התנאי.

```
event   last_value;
always @(last_value) a=1'b1;

always @(posedge clk)
    begin
        if (counter==8'd50)
        ->last_value ;
    end
```

- Delay control : `#<time delay>`

```
#20 clk=!clk;
```

הפעולה מושהית לזמן המוגדר לאחר ה - #.
השהיית 0# מבטיחה את ביצוע ההשמה ב-delta האחרונה.
יחידות הזמן של # מוגדרות ב-`timescale.

- אופרטור # משמש גם ב- parameter override כשמבצעים Instantiation.

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

בלוק initial

- בלוק initial מבוצע פעם אחת.
- בלוק initial מתבצע ברגע 0 של הסימולציה.
- initial משמש לאתחול משתנים, Monitoring וכל תהליך שאמור להתבצע פעם אחת.
- מבנה initial :

```
initial
    begin:<block name> //block name is optional
        <sequential statements>
    end


initial
    fork:<block name> //block name is optional
        <parallel statements>
    join
```

- begin ו-end נדרשים כאשר יש יותר מפעולה אחת בבלוק.

- דוגמא initial :

initial clk=1'b0;

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

פקודת if else

- המשפט ההתנהגותי המקובל ביותר.
- משפט סידרתי. אם מתקיים התנאי מתבצע המשפט הראשון. יש אפשרות לבדוק תנאים נוספים ולבצע את המשפטים שאחריהם או לבצע משפט אם לא התקיים שום תנאי.
- פורמט הצהרה על פקודת if :

```
if (<condition>)
    begin
    <assignments>;
    end
else
    begin
    <assignments>;
    end
```

* בתוך משפט ה - if , begin end נדרשים כשיש יותר מהשמה אחת.

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

פקודת if else

- דוגמאות:

```
reg data_out;

//blocks
always @(enable or data_in)
begin
    if (enable)
        data_out=data_in;
    else
        data_out=1'b0;
end

//data types
wire slct1,slct2,slct3,d1,d2,d3,d4;
reg data_out;

//blocks
always @(slct1 or slct2 or slct3 or d1 or d2 or d3 or
d4)
begin
    if (slct1)
        data_out=d1;
    else    if (slct2)
                data_out=d2;
            else    if (slct3)
                        data_out=d3;
                    else data_out=d4;
end
```

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

פקודת case

- שימושיות כאשר ערך של משתנה בורר בין מספר פעולות אפשריות.
- רק אחת מהאופציות תתקיים.
- דוגמא:

```
case (selector)
    3'b100   : data_out=d1;
    3'd2     : data_out=d2;
    3'b001   : data_out=d3;
    3'b011   : data_out=d4;
    default  : ;
endcase
```

- פעולת case מבוצעת bit אחרי bit ולכן כל האפשרויות חייבות להיות באורך שווה (ניתן להוסיף אפסים).
- begin end אפשרי לכל אלטרנטיבה של case , כשיש יותר מהשמה אחת.
- ניתן לשרשר case בתוך case.

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

פקודת case

- דוגמא (case מקבילי) :

```
wire slct1,slct2,slct3,d1,d2,d3,d4;
reg data_out;

//blocks
always @(slct1 or slct2 or slct3 or d1 or d2 or d3 or
d4)
begin
//   data_out=d4;
     case ({slct1,slct2,slct3})
     3'b100   : data_out=d1;
     3'd2     : data_out=d2;
     3'b001   : data_out=d3;
     default  :   begin
                      data_out=d4;
                      $display ("default assign");
                      $display ($time);
                  end
     endcase
end
```

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

פקודות casex casez

- פקודות casez, casex מאפשרות לבצע case עם אפשרויות לערכי x,z כערכי Don't care .
- אם קיימות מספר אפשרויות עבור השמה בודדת - רק האפשרות הראשונה תיבחר לסינתזה.
- בפקודת casez ערכי Z או ? מהווים Don't care.
- בפקודת casex ערכי X ,Z או ? מהווים Don't care.
- דוגמא:

```
casex ({in_1,in_2,in_3})
        3'bxx1 : data_out=in_2;
        3'bx1x : data_out=in_3;
        3'b1xx : data_out=in_1;
        default: data_out=in_1;
endcase
```

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

פקודות LOOP

- שימושי כשרוצים לחזור על פעולה מסוימת בצורה איטרטיבית מספר פעמים.
- ניתן לבצע לולאה על תחום מסוים בעזרת פקודת for.
- ניתן לבצע לולאה כתלות בהתניה בעזרת פקודת while.
- ניתן לבצע לולאה למספר קבוע של איטרציות בעזרת פקודת repeat.
- ניתן לבצע לולאה אינסופית בעזרת פקודת forever.
- פקודת disable מאפשרת לנטוש את הלולאה לאחר ביצוע האיטרציה הנוכחית.
- כאשר אין המתנה בתוך הלולאה, כל הפעולות בתוכה מתבצעות ב- TICK שעון אחד !

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

משפט LOOP - דוגמא

- שימוש בלולאה ליצירת עירור בסביבת הבדיקה:

```
`define one 1'b1
`define zero 1'b0
module TF;
    //port list

    //data types
    reg      up,down;
    integer  i;

    //blocks
    always begin
        up=`zero;
        down=`zero;

        for (i=1;i<=17;i=i+1) begin
            #123 up=`one;
            #123 up=`zero;
        end

        #200 ;

        repeat (17) begin
            #123 down=`one;
            #123 down=`zero;
        end
    end

endmodule
```

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

`Compiler Directives

- הוראות לקומפיילר (סינתזה וסימולציה) מאפשרות הרחבה של הפקודות הקיימות.
- Compiler Directives נפוצים:
  - `define
  - `include
  - `timescale
  - `ifdef `else `endif
  - `resetall

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

`Compiler Directives - cont

- `include – מעתיק את תוכן ה - Include file אל הנקודה בה מבוצע ה – INCLUDE.
  - דוגמא:

```
define_file.h
`define one 1'b1

module include;
`include "define_file.h"

    wire down;
    assign down=`one;

endmodule
```

- `timescale - משמש להגדרת רזולוציית הסימולטור ויחידות ה - Delay של #.
  - דוגמא:

```
`include "define.v"
`timescale 10 ns / 1 ns   // # units is 10ns
                          //simulation resolution is 1ns
module include;
reg down;
initial begin
     #5 down=`one;
     #10 $printtimescale; // prints the module timescale
end
endmodule
```

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

system tasks

- הדפסת נתונים למסך.
  - $display
  - $strobe
  - $write
  - $monitor
- כללי.
  - $time
  - $random
- שליטה על סימולציה.
  - $finish
  - $stop
- ניהול FILES.
  - $readmemb, $readmemh
  - $fopen, $fclose
  - $fwrite, $fdisplay, $fmonitor, $fstrobe

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

system tasks

❑ דוגמא ל - $display, $time, $random:

```
reg[7:0] C;
integer  A,B;
real     D;

always @(posedge clk)
    begin
        A=$random(B);
        C=$random;
        D=$random;
        $display ("at rising time %t\t the integer
                        random_val is %h",$time,A);
        $displayb ("the binary random value is ",C);
        $display ("the real random value is %g",D);
        $display ("%M");
    end
```

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

אופרטורים בשפה

❑ פירוט אופרטורים לפי עדיפות:

- Replication : {{}} , concatenation : { }
- unary : + , -
- פעולות אריתמטיות : * , / , + , - , %
- הזזה : << , >>
- יחסיות : < , > , <= ,>=
- שוויון : == , != , === , !==
- bitwise : ~ , & , | , ^ , ~^ or ^~
- Reduction : & , ~& , | , ~| , ^ , ~^ or ^~
- פעולות לוגיות : ! , && , ||
- conditional : ? :

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

אופרטורים בשפה

❑ Replication : {{}} , Concatenation : { }

- Concatenation : שרשור משתנים בעלי גודל סופי.

```
reg[7:0] a,b;   reg[15:0] e,f,g;   reg c,d;

initial
    begin
        c=1'b1; d=1'b0; a=8'hfa; b=8'h25;
        e={a,b};
        f={a[3:0],b[3:0],a[7:4],c,d,b[7:6]};
        g={a,c};
    end
```

תוצאות השרשור: `e=16'hfa25, f=16'ha5f8, g=16'h01f5`

- Replication : שיכפול ערך.

```
reg[15:0]    f,g,e;
reg          c,d;

initial
    begin
        c=1'b1;
        d=1'b0;
        f={16{c}};
        g={8{2'b10}};
        e={{8{c}},{8{d}}};// concatenate replicated
    end
```

תוצאות השכפול: `f=16'hffff, g=16'haaaa, e=16'hff00`

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

Comments

❑ ב - Verilog ניתן לכתוב הערה בשורה נבחרת ע"י : // .

❑ ניתן לתחום קטע קוד שלם ע"י : /* */ .

```
// this a one line comment
/* this a multiple
line
comment
*/
```

❑ לא ניתן לשרשר קטעי הערה.

Strings

❑ String נכתב בתוך " " – מכיל תווי ASCII.

❑ קטע TEXT לדיווח הודעות בסימולציה \ שמירת ערך ASCII.

דוגמא:

```
//  string for example
$display (" string for example ");
assign s="AB";
```

Verilog Primer - © Sital Technology Ltd. All Rights Reserved

Case sensitivity

- Verilog היא שפה Case sensitive.
- כל הפקודות כתובות ב - lower case.
- דוגמא :

```
reg[3:0] r,R;
```

מוגדרים 2 משתני reg שונים.

Hierarchical names

- ב - Verilog ניתן להצביע על Identifier גם במיקום אחר בהיררכיה של ה - Design ע"י הגדרת ה - Path ההיררכי שלו.
- בעזרת שמות היררכיים ניתן :
  - לאלץ <u>קווים</u> בהיררכיה נמוכה יותר (גם אם הם לא ports).
  - לבצע בקרה (if, case) על <u>קווים</u> מהיררכיה נמוכה יותר.
- פורמט הצבעה על PATH לשם היררכי :

<top module name>.<instance name>.<instance name>. <data type, block, function, task names>

- דוגמא :

```
always @ (tf.uut.dut.counter)
        if (tf.uut.dut.counter==5'd4)
```


מבנה כללי תוכנית Verilog:

```
module MODULE_NAME (Port1, Port2, Port3, Port4);
// PORT LIST
     input Port1, Port2;
     output [3:0] Port3;
     inout Port4;


// DATA TYPES
     reg reg_name,reg_name... ;
     wire wire_name,wire_name... ;
     parameter parameter_name,parameter_name... ;


// CONCURRENT STATEMENTS
     assign output1 = Expression;


// PROCEDURAL BLOCKS
     initial
          begin
               // Statements
          end

     always @(sensitivity list)
          begin
               // Statements
          end


// MODULE INSTANCES...
     COMP U1 (i1, i2);

     COMP U2 (.Module1(i1), .Module2(i2));
```


מבנה כללי תוכנית Verilog המשך:

```
//TASKS AND FUNCTIONS
     task task_name;
          input p1;
          inout p2;
          output p3;
          begin
               // Statements
          end
     endtask


     function [3:0] function_name;
          input p1;
          begin
               // Statements;
          end
     endfunction

endmodule
```


About Sital Technology

Sital technology Ltd. is a leading provider of tools and services for the electronic R&D industry in Israel. We combine representation of leading EDA (Electronic Design Automation) vendors with training capabilities for chip-design and a high-level design center for HDL, PCB and Network simulation projects.

Our customers range from small, one-man-operation businesses, to the largest corporations in Israel, who rely on our products portfolio, vast experience, excellent technical support capabilities and commitment to provide the right solution.

SITAL Technology's prime quality resource is it's creative, talented and professional staff coming from a wide variety of scientific disciplines, ensuring a broad and comprehensive industry-oriented solution.

For more information:

Web: www.sital.co.il
Tel: 09-76633300
Email: info@sital.co.il

© Sital Technology Ltd. All Rights Reserved