

הטכניון – מכון טכנולוגי לישראל
הפקולטה להנדסת חשמל



מעבדה 1

SYS VERILOG – Cook Book

גרסה 0.17

מרץ 2019

נכתב ע"י אלכס גרינשפון ודודי בר-און

תוכן עניינים

3.....	Miscellaneous	.1
4.....	משתנים	.2
5.....	השמות	.3
6.....	פעולות אריתמטיות ולוגיות OPERANDS	4.
6.....	הצבה מותנית	.5
7.....	יצירת חוג (Loop) בתהליך – שכפול קוד	.6
8.....	תכן הירארכי	.7
9.....	תהליך סינכרוני פשוט (ללא מכונת מצבים)	.8
9.....	תהליך צירופי (אסינכרוני)	.9
10.....	מכונת מצבים תהליך סינכרוני ותהליך אסינכרוני	.10
11.....	העתקת קוד לדו"ח תוך שמוש ב- NOTEPAD++	.11
12.....	הבהרות והמלצות שמקלות על העבודה	.12

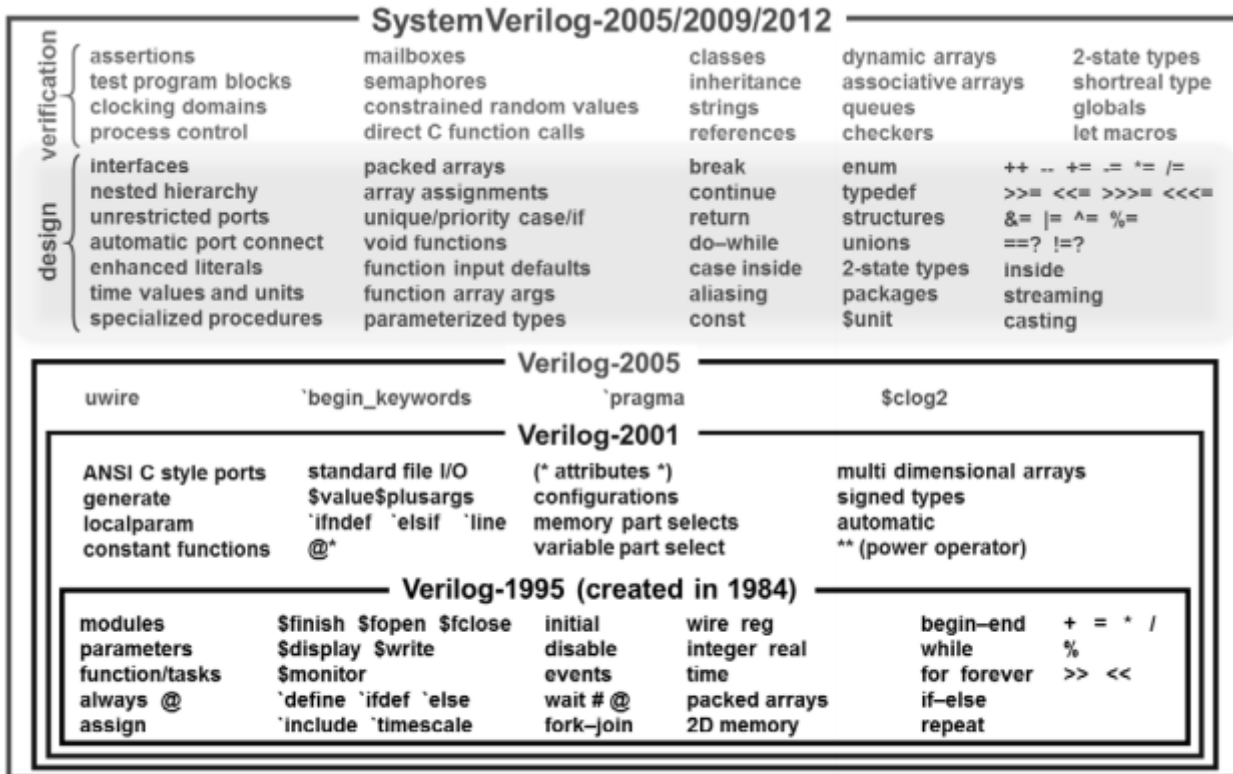


Figure 1. Verilog to SystemVerilog growth chart

Source : http://sutherland-hdl.com/papers/2013-SNUG-SV_Synthesizable-SystemVerilog_paper.pdf

Miscellaneous .1

<pre>import ComplexPkg::*; import ComplexPkg::Complex; import ComplexPkg::add; `include "parts/count.v"</pre>	הגדרות include
<pre>N_n// active low qpc_vld_d <= #1 qpc_vld; // delay logic [1:0] array_ps; // present state logic [1:0] array_ns; // next state</pre>	naming Conventions
<pre>module test (output logic q, output logic [7:0] test_bus, output instruct_bus_t instr, input logic s); endmodule</pre>	הגדרת MODULE וממשק IN OUT
<pre>// We recommend you to always write // comments /* We also recommend you always write Long Comments */</pre>	הערות
<pre>Logic a; Logic b; Logic [1:0] din; always_comb begin din = {a,b}; end</pre>	תהליך אסינכרוני ומשתנים לוקליים

2. משתנים

<code>logic din;</code>	משתנה 1-BIT
<code>logic a, b;</code>	משתנה
<code>logic [3:0] a, b;</code>	ווקטור (4-BIT)
<code>logic [31:0] a, b;</code>	ווקטור (32-BIT)
<code>logic unsigned ui;</code> <code>logic signed si;</code>	משתנה UNSIGNED
	קבועים
<code>localparam logic [25:0] FreqDiv = 26'd20;</code>	משתנה לוקלי
<code>parameter logic [25:0] FreqDiv = 26'd20;</code>	משתנה חיצוני מאפשר קוד גנרי
<code>typedef enum {s0, s1, s2, s3} state_type;</code> <code>state_type state;</code>	Enumeration) (ENUM
<code>enum {a=0, b=7, c, d=8} alphabet;</code>	
<code>logic [7:0] xByte;</code> <code>logic vgaLine [639:0];</code>	מערכים חד ממדי
<code>logic [9:0] [639:0] vgaLinePixel; //packed</code> <code>logic [9:0] vgaLinePixel [639:0]; //unpacked</code> <code>logic t_2d [5:0] [3:0];</code> <code>logic [5:0] [3:0] c_2d;</code>	מערך דו ממדי
<code>assign t_2d[0] [0] = 1'b1;</code> <code>assign c_2d =</code> <code>32'b1111_1111_0000_1111_1010_0111_0000_1000;</code> <code>assign c_2d = 32'hF_F_0_F_A_7_0_8;</code> <code>assign c_2d [6] = 4'h5;</code>	
<code>const logic [3:0] [7:0] T_DATA = {</code> <code>8'h08,</code> <code>8'h11,</code> <code>8'h22,</code> <code>8'h43</code> <code>};</code>	מערך קבועים
<code>logic [7:0] v</code> <code>assign v = 8'h1A;</code>	משתנה HEX
<code>typedef logic [255:0] [7:0] table_type;</code> <code>const table_type char2morse_table =</code> <code>{default:8'h0};</code>	מערך חלקי של קבועים
<code>typedef struct packed signed</code> <code>{bit[3:0] a, b;} uint8;</code>	STRUCT
<code>0 1 Z X</code> <code>X</code>	רמות לוגיות משמשת לסימולציות לתיאור מצב לא חוקי

3. השמות

<pre>always_comb begin // only blocking case (state) idle: begin count_start = 0; load = 0; parity_ok = 0 ; data_valid = 0 ; end endcase end</pre>	BLOCKING =
<pre>logic [3:0] a,b,c, varX; always_ff @(posedge clk) begin varX = 4'b1111; // warning- blocking a <= 4'b0001; b <= varX; c <= a ; end</pre>	Non-blocking <=
שימו לב - הצבת BLOCKING ל VAR לא מחכים שעון	התוצאה תהיה:
	
<pre>assign vector_signal_8 = vector_signal_16[15:8];</pre>	הצבה חלקית
<pre>assign logic_signal = vector_signal_8[5];</pre>	לקיחת BIT בודד
<pre>dout_1={4'b1111,3'b011,1'b0};</pre>	שרשור ביטים
<pre>assign dout_2[7] = 1'b1; assign dout_2[5] = 1'b0;</pre>	השמות לתא בודד
<pre>logic [255:0] dout_3; assign dout_3 = '{ default: 256'b0}</pre>	השמות באמצעות default
<pre>assign data_out = mem[address];</pre>	גישה למערך בכתובת address
<pre>assign sl_vector_signal_8 ='{default: '0};</pre>	

4. פעולות אריתמטיות ולוגיות OPERANDS

Table 11-2—Operator precedence and associativity

Operator	Associativity	Precedence
() [] :: .	Left	Highest ↓ Lowest
+ - ! ~ & ~& ~ ^ ~^ ^~ ++ -- (unary)		
**	Left	
* / %	Left	
+ - (binary)	Left	
<< >> <<< >>>	Left	
< <= > >= inside dist	Left	
== != === !== ==? !=?	Left	
& (binary)	Left	
^ ~^ ^~ (binary)	Left	
(binary)	Left	
&&	Left	
	Left	
?: (conditional operator)	Right	
-> <->	Right	
= += -= *= /= %= &= ^= = <<= >>= <<<= >>>= := :/ <=	None	
{ } { }	Concatenation	

אל תיקחו סיכון - שימו תמיד סוגריים

<pre>assignment_operator ::= // from A.6.2 = += -= *= /= %= &= = ^= <<= >>= <<<= >>>= conditional_expression ::= // from A.8.3 cond_predicate ? { attribute_instance } expression : expression inc_or_dec_operator ::= ++ --</pre>	פעולות חשבוניות על שלמים
<pre>unary_operator ::= // from A.8.6 + - ! ~ & ~& ~ ^ ~^ ^~</pre>	פעולות לוגיות
<pre>stream_operator ::= >> << // from A.8.1</pre>	הזזה SHIFT
<pre>binary_operator ::= + - * / % == != === !== ==? !=? && ** < <= > >= & ^ ^~ ~^ >> << >>> <<< -> <-></pre>	פעולות השוואה (relations): תוצאת הפעולה היא מסוג Boolean

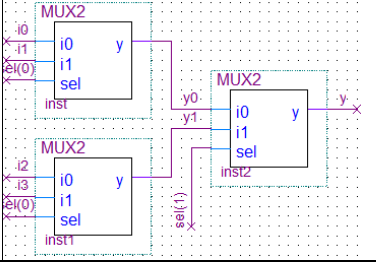
5. הצבה מותנית

<pre>if (a == b) equal = '1; else equal = '0;</pre>	הגדרת IF
<pre>case (din) 2'b11 : y = '1; Default: y = '0; endcase</pre>	הגדרת CASE
<pre>assign busa = drive_bus ? data : 16'bz;</pre>	הצבה מותנית

6. יצירת חוג (Loop) בתהליך – שכפול קוד

<pre>always_comb begin for (int i = 0; i < 7; i ++) begin invert = false; if (invert == true) y_1[i] <= ! x_1[i]; else begin y_1[i] <= x_1[i]; if (x_1[i] == '1') invert = true; end end</pre>	<p>חוג for חוג for מתבצע כל עוד האינדקס בגבולות הנתונים. אסור לשנות את ערכו בתוך החוג.</p> <p>דוגמה מערכת שמוזנת מווקטור כניסה x ומפיקה וקטור יציאה y, שניהם ברוחב 8 סיביות. x ו-y מיצגים מספרים בעלי סימן בשיטת המשלים ל-2. המערכת מבצעת פעולה חשבונית של היפוך סימן: $y = -x$.</p>
<pre>int A_4, i_4; assign A_4 = 3; logic [3:0] z; always_comb begin : count1s int i; z = '0; i = 0; while (i <= 3) begin i_4 = i; if (A_4 == i) begin z[i] = 1'b1; end i++; end</pre>	<p>חוג while חוג while מבצע את הפסוקים הסדרתיים שלו כל עוד התנאי הנבדק הוא true. התנאי נבדק לפני כל איטרציה.</p> <p>דוגמה: אות המוצא Z, וקטור באורך 4, יקבל '1' במקום הנתון על ידי אות הכניסה A, שיכול להיות בעל ערך 1 עד 4.</p>

7. תכן הירארכי

<pre> module mux2(output logic y , // input logic i0 , input logic i1, // input logic sel //); assign y = sel ? i1 : i0; endmodule </pre>	<p>הגדרת מודול תחתון בהירארכיה</p>
<pre> module mux4x1 (input logic [1:0] sel, input logic i0, i1, i2, i3, output logic y_mux) ; logic mux2_y0, mux2_y1 ; mux2 mux2_1 (.i0(i0), .i1(i1), .sel(sel[0]), .y(mux2_y0)); mux2 mux2_2 (.i0(i2), .i1(i3), .sel(sel[1]), .y(mux2_y1)); mux2 mux2_3 (.i0(mux2_y0), .i1(mux2_y1), .sel(sel[1]), .y(y_mux)); endmodule </pre>	<p>שימוש ברכיבים משמשים לכתיבה הירארכית. יש להגדיר את כל אבני הבניה COMPONENTS כ- בארכיטקטורה העליונה</p> <p>דוגמה: בורר $1 \leq 4$ שמורכב משלושה בוררי יסוד $1 \leq 2$</p> 
<pre> module mux2v #(parameter SIZE=8) (output q, input [SIZE-1:0] a, input [SIZE-1:0] b, input sel); endmodule mux2v #(.width(8)) mux2v_1 (.a(a), .b(b), .sel(sel), .q(q)); </pre>	<p>תכן גנרי שימוש ב GENERIC</p>

8. תהליך סינכרוני פשוט (ללא מכונת מצבים)

<pre> logic resetn, one_sec_flag; int one_sec; const int sec = 5; always_ff @(posedge clk or negedge resetn) begin int one_sec; const int sec = 5; if (!resetn) begin one_sec <= 0; one_sec_flag <= '0; end else begin one_sec = one_sec + 1; if (one_sec == sec) begin one_sec_flag <= '1; one_sec <= 0; end else begin one_sec_flag <= '0; one_sec <= one_sec + 1; end end end end </pre>	<p>דוגמה</p>
---	---------------------

כל ההצבות NON BLOCKING <=

9. תהליך צירופי (אסינכרוני)

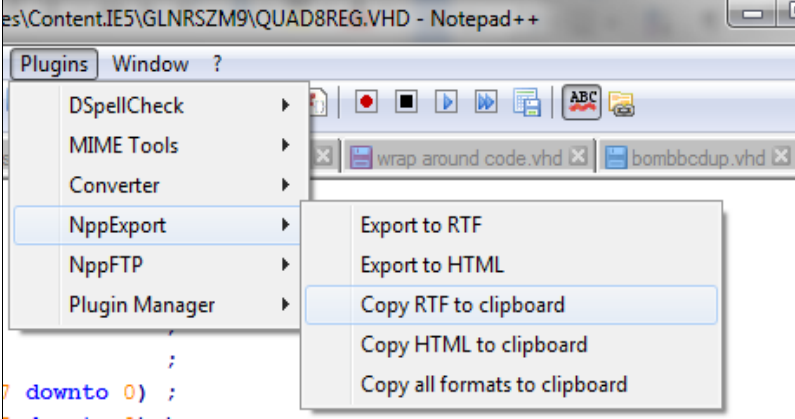
<pre> logic a_gt_b; int a_comb, b_comb; always_comb begin a_gt_b = '0; if (a_comb > b_comb) begin a_gt_b = '1; end end end </pre>	<p>דוגמה</p>
<pre> assign a_gt_b = (a_comb > b_comb) ? '1 : '0 ; </pre>	<p>הצבה פשוטה</p>

כל ההצבות BLOCKING =

10. מכונת מצבים תהליך סינכרוני ותהליך אסינכרוני

<code>enum logic [2:0] {IDLE, WRITE, READ} array_ps, array_ns;</code>	הגדרה
<code>always_ff @(posedge clk or negedge resetN) array_ps <= #1 resetN ? 2'd0 : array_ns;</code>	תהליך סינכרוני
<code>always_comb case (array_ps) IDLE : array_ns = in1 ? READ : (in2 ^ in4) ? WRITE : IDLE; READ : array_ns = (in3 & in2) ? READ : IDLE; WRITE : array_ns = in1 ? IDLE : WRITE; default : array_ns = {2{1'bx}}; endcase</code>	מעברים

11. העתקת קוד לדו"ח תוך שמוש ב- NOTEPAD++

<pre>always_comb begin a_gt_b <= '0; if (a_comb > b_comb) begin a_gt_b <= '1; end end</pre>	<p>ב NOTEPAD++ הקוד מופיע ברור וצבעוני. העתקתו ל- WORD בצורה הרגילה (copy & paste) תבטל את הצבעים:</p>
	<p>על מנת לעשות Copy & Paste מ- NOTEPAD++ תוך שמירת הצבעים, יש לבצע את ההעתקה דרך: Plugins -> NppExport -> Copy RTF to clipboard</p> <p>את NppExport ניתן להוריד מ- http://sourceforge.net/projects/npp-plugins/files/NppExport/</p>
<pre>always_comb begin a_gt_b <= '0; if (a_comb > b_comb) begin a_gt_b <= '1; end end</pre>	<p>אז paste ב- WORD ישמור על העיצוב הצבעוני.</p>
<p>https://github.com/chcg/NPP_ExportPlugin/releases</p> <p>קחו את הקובץ DLL ותשימו אותו בתוך תיקיית: C:\Program Files\Notepad++\plugins ואז תפתחו את ה notepad++-מחדש ואמור לעבוד לכם.</p>	<p>פלג אין לגרסת 64 ביט ותודה לרוה בן סימון</p>

12. הבהרות והמלצות שמקלות על העבודה

גיבוי:

- ☒ בסיום יום עבודה יש לזכור לגבות לרשת DRIVE Z ולענן GOOGLE את עבודתך.
- ☒ DRIVE Z נגיש גם מחוות המחשבים בטכניון אם נכנסים עם **שם המשתמש של המעבדה**.
- ☒ בכל יום לעשות QAR לכל הפרויקט (כולל תוצאות סימולציה, שרטוטים וכו') ולשמור גרסה בענן.
- ☒ לגבות בענן גם את כל מסמכי ה- WORD.