ΗΥ430 – Εργαστήριο Ψηφιακών Κυκλωμάτων

Διδάσκων: Χ. Σωτηρίου, Βοηθός: (θα ανακοινωθεί)

http://inf-server.inf.uth.gr/courses/CE430/

1η διάλεξη – Παρουσίαση του μαθήματος

Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

Δομή της Γλώσσας

- Μοιάζει αρκετά με την C
  - Προ-επεξεργαστή (Preprocessor)
  - Λέξεις Κλειδιά (Keywords)
  - Τελεστές

```
=
==, !=
<, >, <=, >=
&& ||
? :
```

```
& and
| or
~ not
^ xor
```

```
`timescale 1ns / 1ns

`define dh 2
// e.g.: q <= #`dh d; //

`undef dh

`ifdef dh / `ifndef dh
   ...
`else
   ...
`endif

`include "def.h"
```

- Είναι βασισμένη στην έννοια του «γεγονότος» (event)

Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

Λίστες Ευαισθησίας (Sensitivity Lists)

- @ ( ) (διαβάζεται at)
  - Λογική έκφραση
  - Επιτρέπονται μόνο οι εκφράσεις:
    - or
    - posedge (+ακμή), negedge (-ακμή)

```
always @(posedge clk or negedge rst_)
 ...

always @(opcode or b or c)
  if (opcode == 32'h52A0234E)
    a = b ^ (~c);

always @(posedge a or posedge b)
 ...
always @(posedge a, posedge b)
always #5 clk=~clk
```

- Στην περιγραφή συνδυαστικής λογικής, όλα τα σήματα πρέπει να περιλαμβάνονται
- Οι (+, -) ακμές χρησιμοποιούνται μόνο για
  - Ρολόγια
  - Σήματα αρχικοποίησης (reset)

Σε περίπτωση που κάποια από τα σήματα σε συνδυαστικό always παραλειφθούν, στην προσομοίωση δεν θα προαχθεί το γεγονός για να ενεργοποιηθεί το τμήμα always.

Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

Τμήματα `always` και `initial`

Τμήμα **initial**	Τμήμα **always**

initial
begin
  // run once
  a=0; b=0;
  #5; a=1; b=1;
end
	always @(b **or** c)
	begin
	  // run always
	  a <= b & c;
	end

▸ Εκτελείται μια φορά, στην εκκίνηση της προσομοίωσης ▸ Δεν επαναλαμβάνεται ▸ Χρησιμοποιείται για να παρέχει διανύσματα εισόδου στο κύκλωμα ▸ Δεν είναι συνθέσιμο	▸ Εκτελείται στην εκκίνηση της προσομοίωσης ▸ Είναι άπειρος βρόχος ▸ Χρησιμοποιείται για να περιγράφει διαρκή και μόνιμη συμπεριφορά (συνδυαστική ή ακολουθιακή) ▸ Είναι συνθέσιμο

Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

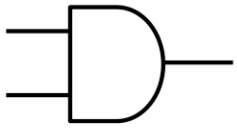
Τύποι Δεδομένων και Πράξεις

- Τιμές Σήματος σε Λογική 4-ρων τιμών (4-value logic)

Τιμή	Ερμηνεία	Χρήση
0	Λογικό 0, άρνηση	Λογικό 0
1	Λογικό 1, κατάφαση	Λογικό 1
X	Άγνωστο ή Μη αρχικοποιημένο	i. Τιμή εκκίνησης ακολουθιακών στοιχείων και σημάτων, ii. Έξοδος πύλης με εισόδους στο Z, iii. Τιμή σε περίπτωση ταυτόχρονης ανάθεσης (0 και 1)
Z	Υψηλής εμπέδησης –ασύνδετο ή τρικατάστατο	i. Τιμή μη οδηγούμενης εισόδου, ii. Έξοδος τρικατάστατου οδηγητή

Τύποι Δεδομένων και Πράξεις

- AND 2 εισόδων:



AND 4-ρων τιμών	0	1	X	Z
0	0	1	0	0
1	0	1	X	X
X	0	X	X	X
Z	0	X	X	X

```
initial ...
```

```
always @(posedge clk)
 if (reset) ...
 else ...
```

Τύποι Δεδομένων και Πράξεις

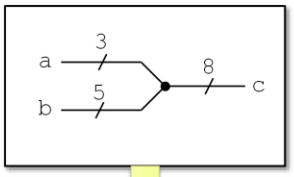
```
if (a==1'bx)
// report error
```

- Μπορεί να έχει νόημα για την προσομοίωση
- Δεν έχει πραγματικό νόημα στην κυκλωματική υλοποίηση

Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

Συνένωση Σημάτων




```
// sign-extend 8-bit number //

wire [7:0] unsigned;
wire [15:0] sign_extend = {

 (unsigned[7] ? 8'hFF : 8'h0),
  unsigned

};
```

```
wire [2:0] a;
wire [4:0] b;

wire [7:0] c = {a , b};
```

```
wire [2:0] a;
wire [4:0] b;

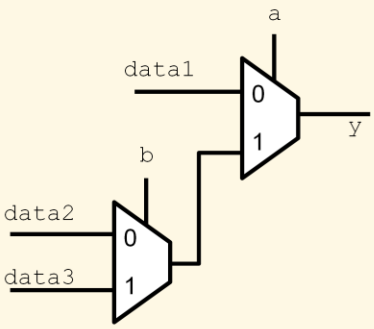
wire [7:0] c = {a , b};
wire [11:0] d = {2{b} , b};
wire [11:0] d = {b, b , b};
wire [25:0] e = {2{3{b}} , a};
```

Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

Συνθήκη `if-else`

Περιγραφή Verilog	Κυκλωματική Μορφή

`always @(a or b or data1 or data2 or data3)`
`begin`
 	
`  if (a == 0)`
`  y = data1;`
`  else begin`
`  if (b == 0)`
`  y = data2;`
`  else`
`  y = data3;`
`  end`
 	
`end`
	

Συνθήκη if-else

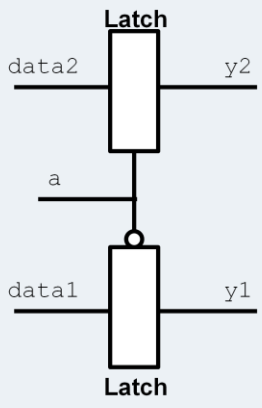
Σε περίπτωση ένθετου if, το else αντιστοιχεί στο κοντινότερο προηγούμενο if

```
always @(a or b or data1…)
  if (a==0)
    if (b==0)
      y = data1;
    else
      y = data2;
end
```

```
always @(a or b or data1…)
  if (a==0)
  begin
    if (b==0)
      y = data1;
  end
  else
      y = data2;
end
```

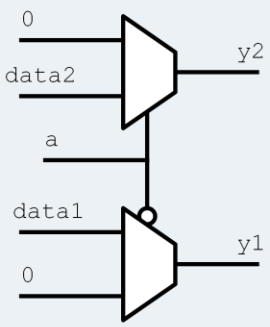
Συνθήκη `if-else`

Περιγραφή Verilog	Κυκλωματική Μορφή

`always @(a or b or data1 or data2)`
 	
`  if (a==0)`
`  y1 = data1;`
`  else`
`  y2 = data2;`
 	
`end`
	

Συνθήκη `if-else`

Περιγραφή Verilog	Κυκλωματική Μορφή

`always @(a or b or data1 or data2)`
`y1 = 0;`
`y2 = 0;`
`if (a==0)`
`y1 = data1;`
`else`
`y2 = data2;`
 	
`end`
	

Συνθήκη case

```
always @(a or b or c or d or sel )
begin
  case (sel)
    2'b00: out <= a;
    2'b01: out <= b;
    2'b10: out <= c;
    2'b11: out <= d;
    default: out <= 5'bx;
  endcase
end
```

```
always @(a or b or c or d or sel )
begin
  case (sel)
    2'b00: out <= 2'b00;
    2'b01: out <= 2'b01;
    2'b10: out <= 2'b11;
    2'b11: out <= 2'b10;
    default: out <= 2'bx;
  endcase
end
```

Μετατροπέας Δυαδικού Αριθμού σε Κώδικα grey

00→00
01→01
10→11
11→10

Συνθήκη case

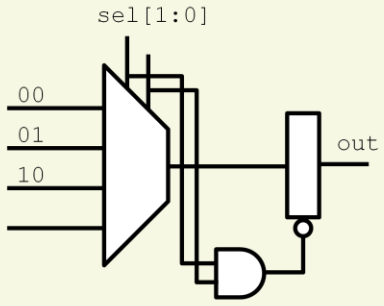
Ελλιπής Συνθήκη case

```
always @(a or b or c or d
or sel )
begin
  case (sel)
    2'b00: out <= 2'b00;
    2'b01: out <= 2'b01;
    2'b10: out <= 2'b11;

  endcase
end
```

Συνεπάγεται Latches!

Οι περιπτώσεις που δεν καλύπτονται συνεπάγονται μνήμη, δηλ. `out <= out`



Συνθήκη `case` με Αδιάφορες (DC) τιμές

casex	casez
casex (A) // X, ? Or Z = DC input // 4'b1XXX : Y = 3; 4'b01XX : Y = 2; 4'b001X : Y = 1; **default** : Y = 0; **endcase**	**casez** (A) // ? Or Z = DC input // 4'b1??? : Y = 3; 4'b01?? : Y = 2; 4'b001? : Y = 1; **default** : Y = 0; **endcase**

- Οι αδιάφορες τιμές (don't cares – DC) αντιστοιχούν **με 0 ή 1**
  - Χρησιμοποιούνται για απλοποίηση στο B

Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

Βρόχοι `for` και `while`

- Λειτουργούν όπως σε παραδοσιακές γλώσσες προγραμματισμού
- **Δεν υποστηρίζονται:**
  - `break, continue`
  - `i++, i--`
- Χρήση:
  i. testbench
  ii. επαναληπτική εμφάνιση

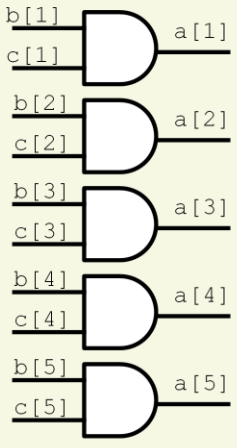
```
integer i;
initial begin
  for (i = 0; i < 10; i = i + 1)
   begin
    $display ("i= %d",i);
   end
end
```

```
integer j;

initial begin
  j=0;
  while (j < 10)
  begin
    $display ("j= %b",j);
    j=j + 1;
  end
end
```

Βρόχοι `for` και `while`

Περιγραφή Verilog	Κυκλωματική Μορφή

`integer i;`
 	
`always @(b or c) begin`
`  for (i = i; i < 5; i = i + 1)`
`  begin`
`  a[i] <= b[i] & c[i];`
`  end`
`end`
 	
Συνθέσιμο
	

Βρόχοι `for` και `while`

Περιγραφή Verilog	Κυκλωματική Μορφή

`integer i;`
 	
`always @(b or c) begin`
`  for (i = i; i < 5; i = i + 1)`
`  begin`
`  a <= i;`
`  #10`
`  end`
`end`
 	
Μη Συνθέσιμο
	

Στην προσομοίωση το a θα πάρει την τιμή 4…

Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

Παράμετροι και Χρήση τους

Καταχωρητής Μεταβλητού Πλάτους

```
module RegLd(D, Q,
load, clk);

parameter N = 8;
parameter dh = 2;

input  [N-1:0] D;
output [N-1:0] Q;
input  load, clk;
reg [N-1:0] Q;

always @(posedge clk)
 if (load)
  Q = #dh D;

endmodule
```

Πιθανές Εμφάνισεις του

```
RegLd reg0(d0, q0, ld, clk);

RegLd #(16,2) reg1(d1, q1,
ld, clk);

RegLd reg2(d2, q2, ld, clk);

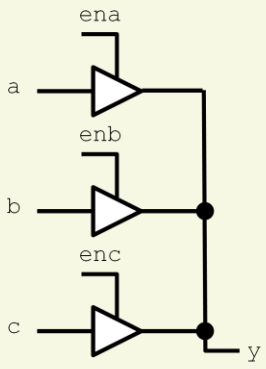
defparam  reg2.N = 4;
defparam  reg2.dh = 4;
```

Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

Οδηγοί Τριών Καταστάσεων

Κύκλωμα	Περιγραφή Verilog

	`module tristates(a, b, c, ena, enb, enc, y);`
	
	`input a, b, c, ena, enb, enc;`
	`output y;`
	
	`assign y = ena ? a : 1'bz;`
	`assign y = enb ? b : 1'bz;`
	`assign y = enc ? c : 1'bz;`
	
	`endmodule`

Οδηγοί Τριών Καταστάσεων

Περιγραφή Μονάδας με `inout`	Εμφάνιση

`module tristate(en, clk, data);`
`input en, clk;`
`inout [7:0] data;`
 	
`wire [7:0] data = (en) ? data_out : 8'bz;`
 	
`always @(posedge clk)`
`begin`
`  if (!en)`
`   case (data)`
`    ...`
`endmodule`
	`wire [7:0] bus;`
	
	`tristate tr0(en0, clk, bus);`
	`tristate tr1(en1, clk, bus);`
	`tristate tr2(en2, clk, bus);`
	
	Δεν χρησιμοποιούνται σε EDA

Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

Μνήμες από Καταχωρητές

- Μνήμες υποστηρίζονται ως πίνακες ψηφίων
  - `reg [7:0] mem [3:0];`
    - Μνήμη με τέσσερις λέξεις των 8-bit
- Αρχικοποίηση – ειδικές διαδικασίες συστήματος
  - `$readhmemh, $readmemb`
  - `$writememh, $writememb`
- Για μεγάλες μνήμες SRAM ή Αρχείο Καταχωρητών (Register File) επεξεργαστή, είναι προτιμότερο να χρησιμοποιούνται έτοιμες μονάδες
  - η υλοποίηση από την σύνθεση δεν είναι ανάλογη σε ταχύτητα και εμβαδό

Μνήμες από Καταχωρητές

Παραδείγματα Περιγραφής Verilog

```
wire [15:0] word_in;
wire [15:0] word_out;
wire [ 9:0] addr;
reg  [15:0] memory [1023:0];

always @(posedge clk)
begin
 if (we)
  memory[addr] = word_in;
 else
  word_out = memory[addr];
end
```

Αρχικοποίηση

```
always @(negedge rst_n)
  $readmemh("memory.dat",
memory);


initial
begin
 $readmemh("memory.dat",
memory);
End
```

memory.dat:

```
0F00
00F1
0F02
```

Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

Συναρτήσεις

Συναρτήσης

- Δήλωση
  - `function [range_or_type] function_name;`
    `input declarations;`
    `local variable declarations;`
    `statements;`
    `endfunction`
- Αποτέλεσμα με ανάθεση
  - `function_name = expression;`
- Χρήση
  - `function_name(expression, …)`
  - με την σειρά που ορίζονται στην δήλωση

Παράδειγμα

```
function [15:0] average;

input [15:0] a, b, c, d;

begin
  average = (a + b + c +
  d) >> 2;
end

endfunction;
```

```
wire x =
average(e, f, g, h);
```

Συναρτήσεις

- Η καθιερωμένη επιστροφή είναι το 1-bit
- Επιτρέπονται πολλαπλά ορίσματα εισόδου
- Μια συνάρτηση μπορεί να κάνει χρήση άλλης
- Μια συνάρτηση δεν επιτρέπεται να χρησιμοποιεί διαδικασίες (tasks)
- Δεν υποστηρίζεται η αναδρομή
- Δεν υποστηρίζονται εκφράσεις με την έννοια του χρόνου
  - Καθυστερήσεις
  - Γεγονότα
- Χρησιμοποιούνται για την σύνθεση συνδυαστικής λογικής

Συναρτήσεις

Παραδείγματα

```
function calc_parity;
input [31:0] val;
begin
   calc_parity = ^val;
end
endfunction
```

```
function [7:0] GetByte;
input [63:0] Word;
input [3:0] ByteNum;
integer Bit;
reg [7:0] temp;

begin

for (Bit=0; Bit<=7; Bit=Bit+1)
   temp[Bit] =
   Word[((ByteNum-1)*8)+Bit];

GetByte = temp ;

Endfunction
```

Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

Διαδικασίες (Tasks)

- Οι διαδικασίες είναι ουσιαστικά υπο-ρουτίνες
- Μπορούν να περιέχουν πολλαπλές εισόδους και πολλαπλές εξόδους
- Επιτρέπονται οι χρονικές λειτουργίες
  - Καθυστερήσεις (`#`)
  - Γεγονότα (`@, wait`)
- Δεν υποστηρίζουν αναδρομή
- Μπορούν να κάνουν χρήση από άλλες διαδικασίες και συναρτήσεις
- Μπορεί να μην είναι συνθέσιμες, εξαρτάται από την περιγραφή…

Διαδικασίες (Tasks)

Παραδείγματα

```
task ReverseByte;
input [7:0] a;
output [7:0] ra;
integer j;

begin
  for (j = 0; j <= 7; j++)
  begin
   ra[j] = a[7-j];
  end
end
endtask
```

```
task read_mem;
input [15:0] address;
output [31:0] data;

begin
  read_request = 1;
  wait (read_grant)
     addr_bus = address;
  data = data_bus;
  #5 addr_bus = 16'bz;
  read_request = 0;
end
endtask
```

Συναρτήσεις και Διαδικασίες

- Ορίζονται εντός μονάδων και είναι **τοπικές**
- Χρησιμοποιούνται μόνο για περιγραφή συνδυαστικών τμημάτων
- Δεν μπορούν να εμπεριέχουν τμήματα always ή initial
- Μπορούν όμως να χρησιμοποιούνται μέσα σε always ή initial
- Γενικά, οποιοδήποτε διαδικασία μπορεί να περιγραφεί εναλλακτικά από μια (ή περισσότερες) συνάρτησεις

Συναρτήσεις και Διαδικασίες

Συναρτήσεις	**Διαδικασίες**
Μπορούν να χρησιμοποιούν άλλες συναρτήσεις αλλά όχι διαδικασίες	Μπορούν να χρησιμοποιούν άλλες συναρτήσεις και διαδικασίες
Εκτελούνται σε μηδενικό χρόνο	Μπορούν να διαρκούν συγκεκριμένο χρόνο
Δεν επιτρέπουν χρονικό έλεγχο (καθυστερήσεις, γεγονότα)	Επιτρέπονται οι χρονικοί έλεγχοι
Έχουν τουλάχιστον μια είσοδο	Έχουν μηδενικές ή περισσότερες εισόδους και εξόδους
Επιστρέφουν μια έξοδο	Επιστρέφουν έμμεσα μια ή περισσότερες εξόδους

Διαδικασίες Συστήματος

- Καθιερωμένες διαδικασίες που ορίζει ένα εργαλείο EDA
  - Ξεκινούν με `$`, λ.χ. `$monitor`

Όνομα Διαδικασίας	Λειτουργία
$time	Επιστρέφει τον χρόνο της προσομοίωσης
$display	Τυπώνει τιμές σημάτων – ανάλογη της printf `$display("format-string", expr1, …, exprn);` `%d` (decimal), `%h` (hex), `%b` (binary), `%t` (time)
$monitor	Παρακολουθεί σήματα ως γεγονότα, και τα τυπώνει όταν αποκτήσουν νέα τιμή – έχει ανάλογα ορίσματα όπως η `$display`
$stop	Διακόπτει την προσομοίωση
$finish	Ολοκληρώνει την προσομοίωση
$random	Επιστρέφει ένα 32-bit ψευδοτυχαίο αριθμό
$readmemh, $readmemb	Ανάγνωση περιεχομένων μνήμης

Διαδικασίες Συστήματος

Παραδείγματα

```
$display("Error at time %t: value is %h, expected %h", $time,
actual_value, expected_value);
```

```
$monitor("cs=%b, ns=%b", cs, ns)
```

```
$random %64
```

Όνομα Διαδικασίας	Λειτουργία
$dumpon, $dumpoff, $dumpvars	Ορίσματα αρχείου Verilog Change Dump (VCD)
$setup, $hold, $period	Έλεγχοι χρονικών περιορισμών
$fopen, $fclose, $fmonitor, $fdisplay	Συναρτήσεις αρχείων
$sdf_annotate	Επισύναψη αρχείου SDF (Standard Delay Format) στις εμφανίσεις

Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

Χρόνος και Γεγονότα

- Η σημασιολογία και προσομοίωση βασίζονται σε αξιολόγηση γεγονότων

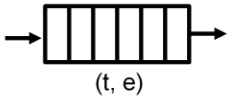
```
a = #5 b; #10 c = a;
```

```
wait (a) b = 1;
```

```
always @(posedge clk) a = b + 1;
always clk = #(`period/2) ~clk;
@(negedge clk);
```

```
a <= b;
b <= a;
c <= a & b;
```

- Κάθε πρόταση της Verilog συνεπάγεται



  - Αξιολόγηση εκφράσεων (Δεξιά)
  - Εισαγωγή της ανάθεσης στην ουρά γεγονότων (Αριστερά)
- Για γεγονότα που συμβαίνουν παράλληλα δεν υπάρχει εγγύηση ως προς την σειρά τους

Χρόνος και Γεγονότα

Ανάθεση	Σημασιολογία
`Register_data_type = expression;`	**Κλειδωμένη (blocking) ανάθεση** η έκφραση αξιολογείται και γίνεται η ανάθεση πριν την εκτέλεση επόμενης γραμμής i. Αξιολόγηση άμεσα Δεξιά και Ανάθεση Αριστερά, ii. Εκτέλεση εν σειρά της επόμενης πρότασης.
`Register_data_type <= expression;`	**Μη κλειδωμένη (non-blocking) ανάθεση** η έκφραση αξιολογείται, αλλά η ανάθεση θα γίνει στο τέλος του τρέχοντος βήματος χρόνου και η εκτέλεση συνεχίζει i. Αξιολόγηση άμεσα Δεξιά, ii. Ανάθεση αριστερά στο τέλος του χρόνου Δ, iii. Άμεση εκτέλεση της επόμενης πρότασης.

Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

Γεγονότα και Καθυστερήσεις

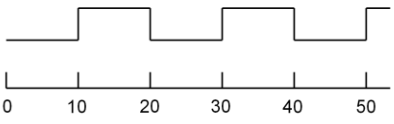
Εξωτερική Καθυστέρηση	`#5 a = b + c;`
	`#4 d = a;`
		@t=5: a = b + c ; //b, c @t=5 //
		@t=9: d = a ;// a @t=9 //

Εσωτερική Καθυστέρηση	`a = #5 b + c;`
	`d = #4 a;`
		@t=5: a = b + c ; //b, c @t=0 //
		@t=9:d = a; // a @t =5 //

Με μη-κλειδωμένες αναθέσεις	`a <= #5 b + c;`
	`d <= #4 a;`
		@t=5: a = b + c; //b, c @t=0 //
		@t=4:d = a; // a @t =0 //

Γεγονότα και Καθυστερήσεις

- Κάθε πρόταση συνδέεται με τον αρχικό της χρόνο
- Τμήματα `initial`, `always` λειτουργούν σειριακά, εκτός αν περιέχουν αναθέσεις <=




```
initial begin                          0
 a = 0; b = 0; c = 0;
 clk = 0;
end

always begin                           10, 20, 30, 40, 50
 clk = #10 1;
 clk = #10 0;                          4, 14, 34
end
wire #4 [3:0] comb = a + b;

always @(posedge clk)                  10, 30
 a <= b + 1;
always @(posedge clk)                  10, 30
 b <= c + 1;
always @(posedge clk)                  15, 35
 c <= #5 a + 1;
```

Παράδειγμα Καθυστερήσεων σε Αναθέσεις

Περιγραφή Verilog

```
module va(a, b, c1, c2, c3, c4);
input a, b;
output c1, c2, c3, c4;
reg c1, c2, c3, c4;

   always @(a or b)
     begin
         #3 c1 = (a & b);
     end
   always @(a or b)
     begin
         c2 = #3 (a & b);
     end
   always @(a or b)
     begin
         #3 c3 <= (a & b);
     end
   always @(a or b)
     begin
         c4 <= #3 (a & b);
     end
endmodule
```

Testbench

```
`timescale 1ns/10ps
module va_testbench;
   reg a, b;
   wire c1, c2, c3, c4;
   va va_inst (a, b, c1, c2, c3, c4);
   initial
   begin
      a = 0;
      b = 0;
      $monitor("time %d ns: a=%b, b=%b,
c1=%b, c2=%b, c3=%b, c4=%b", $time, a,
b, c1, c2, c3, c4);
      #50
      #1 a = 1; b = 1;
      #3 a = 0;
      #1 a = 1;
      #100 a = 1;
      b = 1;
   end
endmodule
```

Παράδειγμα Καθυστερήσεων σε Αναθέσεις


```
always @(a or b)
  begin
      #3 c1 = (a & b);
  end
always @(a or b)
  begin
      c2 = #3 (a & b);
  end
```

```
always @(a or b)
     begin
         #3 c3 <= (a & b);
     end
   always @(a or b)
     begin
         c4 <= #3 (a & b);
     end
```


<u>Αναλυτική Ανάλυση Σειράς Γεγονότων:</u>

@t=51, a = 1, b = 1,

@t=54, c1 = (a & b)@t=54
@t=52, c2 = (a & b)@t= 51
@t=54, c3 <= (a & b)@t=54
@t=54, c4 <= (a & b)@t= 51

@t=54, a = 0,
@t=55, a = 1,

@t=57, c1 = (a & b)@t=57
@t=57, c2 = (a & b)@t=54
@t=57, c3 <= (a & b)@t=57
@t=57, c4 <= (a & b)@t=54

ΑΚΥΡΩΝΕΤΑΙ από την ανάθεση στο t=57 – (@t=58, c1 = (a & b)@t=58)
ΑΚΥΡΩΝΕΤΑΙ από την ανάθεση στο t=57 – (@t=58, c2 = (a & b)@t=55)
@t=58, c3 <= (a & b)@t=58
@t=58, c4 <= (a & b)@t=58

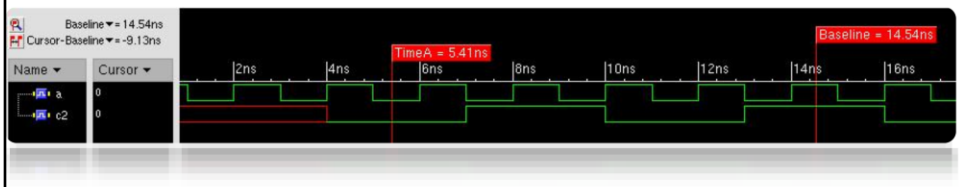
Παράδειγμα Καθυστερήσεων σε Αναθέσεις

Περιγραφή Verilog

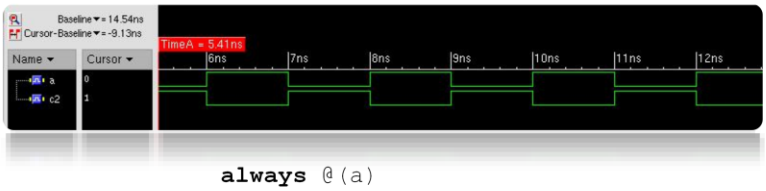
```
`timescale 1ns/10ps
module alt_va;
   reg c2;
   reg a;
   initial begin
      a = 1;
   end
   always @(a)
     begin
       c2 = #3 a;
     end
   always
     begin
       #1 a = ~a;
     end
endmodule
```

- Ποια η διαφορά μεταξύ
  - `c2 = #3 a;`
    - και
  - `c2 <= #3 a;`

Παράδειγμα Καθυστερήσεων σε Αναθέσεις




```
always @(a)
  c2 = #3 a;
```




```
always @(a)
  c2 <= #3 a;
```


Παρόλο που το σήμα a αλλάζει κάθε 1ns, η κλειδωμένη ανάθεση θα δειγματοληπτεί κάθε 3ns. Οι ενδιάμεσες αναθέσεις ακυρώνονται.

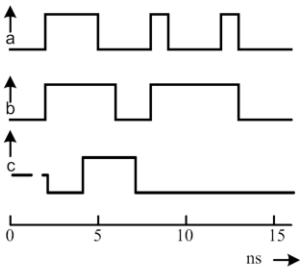
Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

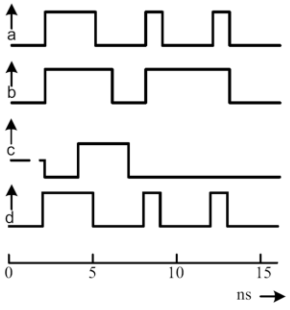
Καθυστερήσεις Συνδέσεων

Παραδείγματα

```
wire a,b,c;
assign #2 c = a & b;
```




```
wire a,b,d;
wire #2 c;
assign c = a & b;
assign d = a & b;
```

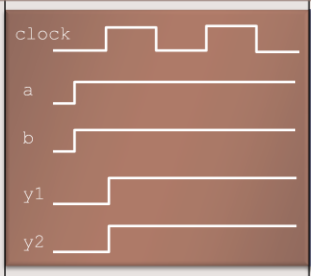
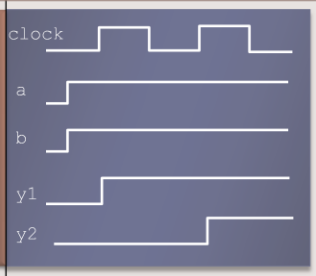


Περιεχόμενα

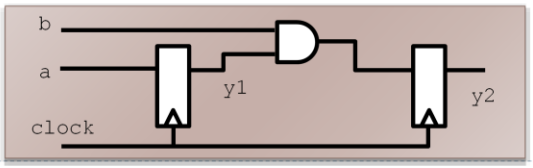
- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

Εξαρτήσεις μεταξύ παράλληλων Τμημάτων

Περιγραφή Verilog	Αξιολόγηση 1ου, 2ου	Αξιολόγηση 2ου, 1ου

`always @(posedge clock)`
`  begin`
`  y1 = a;`
`end`
 	
`always @(posedge clock)`
`  begin`
`  if (y1 == 1)`
`  y2 = b;`
`  else`
`  y2 = 0;`
`end`
		

- **Ασυμφωνία προσομοίωσης συμπεριφοράς και σύνθεσης**



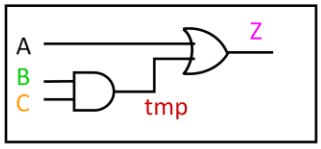
59 HY430 - Διάλεξη 6η - Verilog II - Λεπτομέρειες

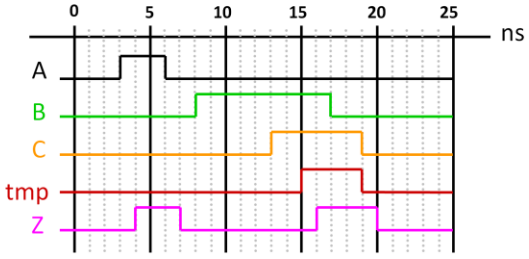
Η ασυμφωνία λύνεται συγχωνεύοντας τα 2 τμήματα always σε ένα

Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

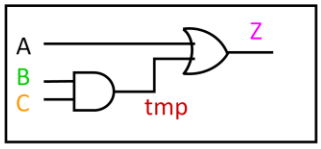
Καθυστερήσεις Πυλών

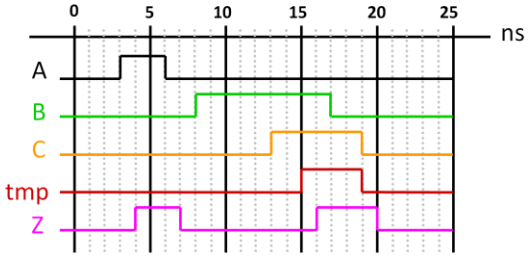




- Έστω Δ_{AND} = 2ns, Δ_{OR} = 1ns, με μηδενική καθυστέρηση των συνδέσεων
- Πόσα και ποια μονοπάτια υπάρχουν;
- Οι κύματομορφές δείχνουν την συμπεριφορά του κυκλώματος για διανύσματα A, B, C.

Καθυστερήσεις Πυλών





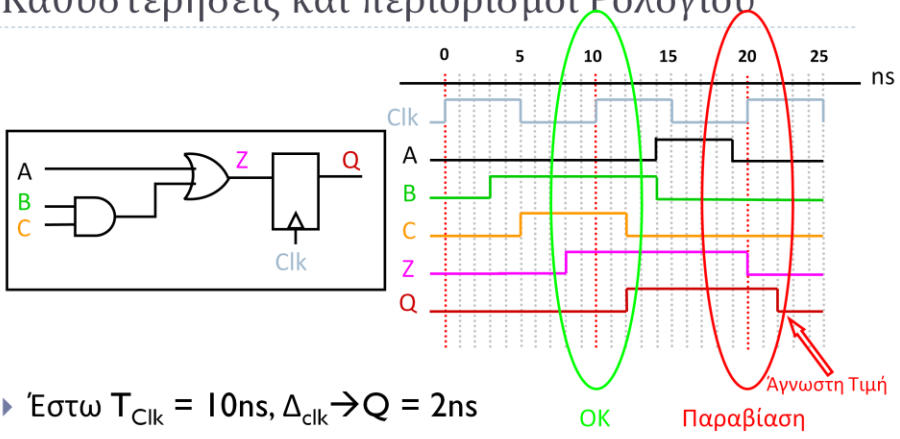
- Έστω Δ_{AND} = 2ns, Δ_{OR} = 1ns, με μηδενική καθυστέρηση των συνδέσεων

```
always@(a or b or c)
begin
  tmp = #2 B & C;
  Z   = #1 A | tmp;
end
```

Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

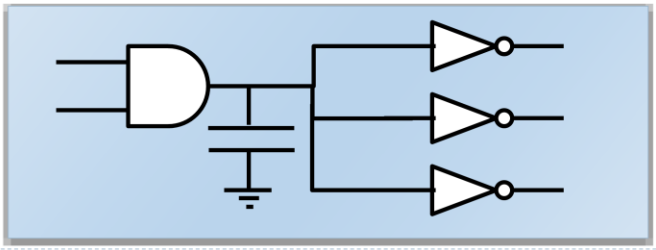
Καθυστερήσεις και περιορισμοί Ρολογιού



- Έστω T_{Clk} = 10ns, $\Delta_{clk} \rightarrow Q$ = 2ns
- Tsetup = 1ns, T_{hold} = 0.5ns
- Παραβιάζονται οι περιορισμοί setup και hold;

Καθυστερήσεις μετά από Σύνθεση

- Οι καθυστερήσεις των πυλών μετά από σύνθεση δεν θα είναι εκ προοιμίου γνωστές
  - Εξαρτώνται από
    - Την συγκεκριμένη πύλη που απαιτείται
    - Το μέγεθος της πύλης
    - Το φορτίο της πύλης από (1) τις πύλες που οδηγεί, (2) τις συνδέσεις που οδηγεί



Παραδείγματα Καθυστερήσεων

Διεργασία	Καθυστέρηση **AND2**
90nm (Pentium III)	2ns
65nm (Pentium 4)	0.5ns
45nm (Core Duo)	0.4ns

Περιεχόμενα

- Δομή της Γλώσσας
- Λίστες Ευαισθησίας (Sensitivity Lists)
- Τμήματα initial και always
- Τύποι Δεδομένων και Πράξεις
- Συνένωση σημάτων
- Εκφράσεις Συνθηκών (if-else, case)
- Βρόχοι (for, while)
- Παράμετροι
- Σήματα τριών καταστάσεων
- Μνήμες
- Συναρτήσεις
- Διαδικασίες (Tasks)
- Γεγονότα
- Καθυστερήσεις
- Καθυστερήσεις Συνδέσεων
- Εξαρτήσεις Παράλληλων Τμημάτων
- Καθυστερήσεις Πυλών
- Περιορισμοί Ρολογιού
- Παράλληλες Αναθέσεις

Παράλληλες Αναθέσεις – `fork/join`

- Το τμήμα `fork` εκτελεί παράλληλα όλες τις προτάσεις μέχρι το `join`
  - Δύσκολα υποστηρίζονται στην σύνθεση
- Μορφή
  - ```
    fork
        statement;
        statement;
        statement;
    ...
    join
    ```
    ```

Παράδειγμα Παράλληλων Αναθέσεων

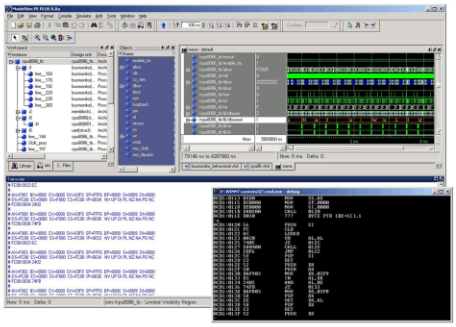
Χωρίς fork/join

```
reg x,y;
reg [1:0] z;
initial begin
  x = 1'b0;      // time 0
  #5 y = 1'b1;  // time 5
  #10 z = {x,y};// time 15
end
```

Με fork/join

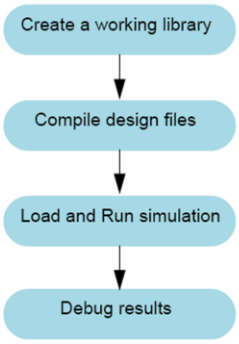
```
reg x,y;
reg [1:0] z;
initial begin
fork
  x = 1'b0;      // time 0
  #5 y = 1'b1;  // time 5
  #10 z = {x,y};// time 10
join
end
```

Modelsim



- Χρησιμοποιείται για την προσομοίωση κυκλωμάτων σε Verilog, VHDL

Modelsim Introduction



1. Δημιουργία βιβλιοθήκης
2. Compile του κώδικα
3. Εκτέλεση και προσομοίωση

Modelsim Window