

Γλώσσες Περιγραφής Υλικού I

Μη προγραμματιζόμενοι επεξεργαστές

Νικόλαος Καββαδίας
nkavn@uop.gr

04 Μαΐου 2011

Σκιαγράφηση της διάλεξης

- Μη προγραμματιζόμενοι επεξεργαστές
 - Η οργάνωση των μη-προγραμματιζόμενων επεξεργαστών
 - Χειριστής/μονάδα ελέγχου (controller)
 - Χειριστής δεδομένων (datapath)
 - Στοιχεία αλγοριθμικής σχεδίασης
 - Υλοποίηση με μηχανή πεπερασμένων καταστάσεων με χειριστή δεδομένων (FSMD: Finite-State Machine with Datapath)
 - Προσομοίωση της συμπεριφοράς ενός μη προγραμματιζόμενου επεξεργαστή
 - Σύγκριση με διανύσματα αναφοράς
- Πλήρες παράδειγμα: επεξεργαστής για τον υπολογισμό του μέγιστου κοινού διαιρέτη (GCD) δύο θετικών ακεραίων αριθμών

Μη προγραμματιζόμενοι επεξεργαστές

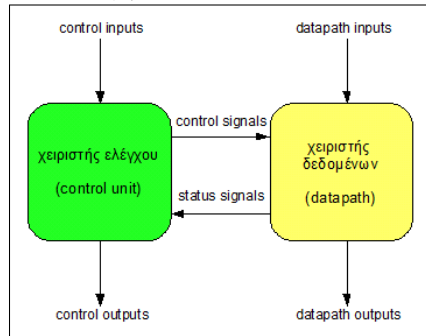
- Μη προγραμματιζόμενοι επεξεργαστές είναι εκείνα τα κυκλώματα τα οποία έχουν σχεδιαστεί έτσι ώστε να μπορούν να επιλύσουν ένα μόνο πρόβλημα
- Η μη προγραμματισιμότητα οφείλεται στον τρόπο σχεδιασμού των μηχανισμών ελέγχου των λειτουργιών επεξεργασίας δεδομένων που συμβαίνουν στον επεξεργαστή
- Η λογική ελέγχου στον επεξεργαστή είναι καλωδιωμένη (hardwired) και γενικά δεν μπορεί να τροποποιηθεί μετά την υλοποίηση του επεξεργαστή
- Ένας μη προγραμματιζόμενος επεξεργαστής αποτελείται από το χειριστή ελέγχου (controller ή control unit) και το χειριστή δεδομένων (datapath)

Η οργάνωση ενός μη-προγραμματιζόμενου επεξεργαστή (1)

- Ο χειριστής ελέγχου παράγει σήματα ελέγχου (control signals) για τη δρομολόγηση των μηχανισμών που λαμβάνουν χώρα στο χειριστή δεδομένων
- Στο χειριστή δεδομένων πραγματοποιείται η επεξεργασία δεδομένων με τα αρχικά δεδομένα να λαμβάνονται από κάποια εξωτερική πηγή ή από στοιχεία αποθήκευσης (π.χ. μνήμες ROM, RAM) και τα τελικά να αποθηκεύονται σε καταχωρητές ή μνήμες
- Ο χειριστής δεδομένων επιστρέφει στο χειριστή ελέγχου σήματα κατάστασης (status signals) τα οποία κατευθύνουν τη μετάβαση ανάμεσα στις εσωτερικές καταστάσεις του χειριστή ελέγχου
- Ο χειριστής ελέγχου υλοποιείται συχνά ως FSM

Η οργάνωση ενός μη-προγραμματιζόμενου επεξεργαστή (2)

- Η αρχιτεκτονική FSMD (Finite-State Machine with Datapath) αποτελεί ένα είδος περιγραφής μη προγραμματιζόμενων επεξεργαστών σε επίπεδο RTL
- Καταστάσεις του FSM ενσωματώνουν μηχανισμούς του χειριστή δεδομένων
- Γενικό σχηματικό διάγραμμα



Νικόλαος Καββαδίας nkavn@uop.gr

Γλώσσες Περιγραφής Υλικού I

Το πρόβλημα του μέγιστου κοινού διαιρέτη δύο αριθμών

- Δεχόμαστε ότι: $gcd(n, 0) = gcd(0, n) = gcd(0, 0) = 0$
- Το ζητούμενο είναι η εύρεση αριθμού m ο οποίος να είναι ο μεγαλύτερος θετικός ακέραιος ο οποίος διαιρεί και τους δύο αριθμούς
- Στα αρχαία Ελληνικά μαθηματικά αντιπροσωπεύει το πρόβλημα εύρεσης κοινής αναφοράς για τη μέτρηση δύο ευθύγραμμων τμημάτων
- Το πρόβλημα του GCD επιλύεται με τον αλγόριθμο του Ευκλείδη

```
unsigned int gcd(unsigned int a, unsigned int b) {  
    assert(a > 0 && b > 0);  
    if (a == b) return a;  
    if (a > b) return gcd(a-b, b);  
    if (b > a) return gcd(a, b-a);  
}
```

- Στον αλγόριθμο του Ευκλείδη, η αναδρομή μπορεί να αποφευχθεί

Νικόλαος Καββαδίας nkavn@uop.gr

Γλώσσες Περιγραφής Υλικού I

Ο αλγόριθμος του μέγιστου κοινού διαιρέτη

- Ο αλγόριθμος υπολογισμού του Μέγιστου Κοινού Διαιρέτη δύο θετικών ακεραίων

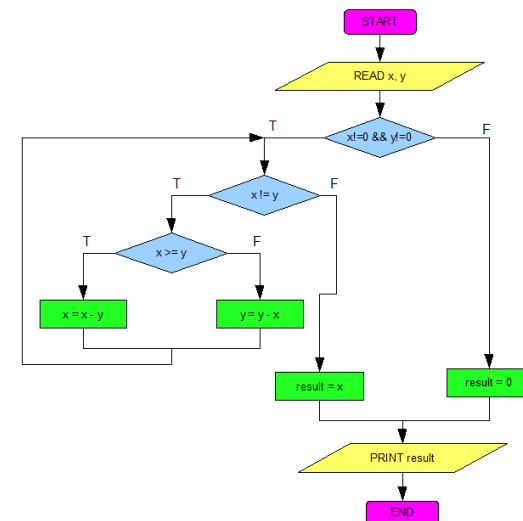
```
int gcd(int a, int b)  
{  
    int result;  
    int x, y;  
  
    x = a;  
    y = b;  
  
    if (x != 0 && y != 0)  
    {  
        while (x != y)  
        {  
            if (x >= y)  
                x = x - y;  
            else  
                y = y - x;  
        }  
    }  
}
```

```
    result = x;  
}  
else  
{  
    result = 0;  
}  
return (result);  
}  
  
int main()  
{  
    int result = gcd(196, 42);  
    return (result);  
}
```

Νικόλαος Καββαδίας nkavn@uop.gr

Γλώσσες Περιγραφής Υλικού I

Αλγοριθμικό διάγραμμα ροής για τον αλγόριθμο GCD



Νικόλαος Καββαδίας nkavn@uop.gr

Γλώσσες Περιγραφής Υλικού I

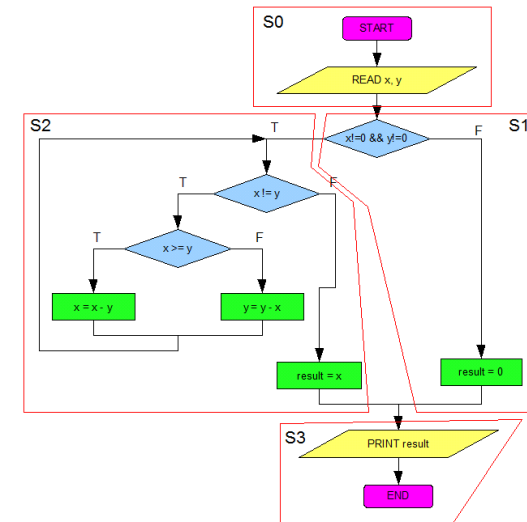
Αριθμητικό παράδειγμα υπολογισμού του GCD

- Ο μικρότερος αριθμός από δύο αριθμούς a, b αφαιρείται από το μεγαλύτερο σε διαδοχικά βήματα
- Όταν οι δύο αριθμοί γίνουν ίσοι, τότε ισούνται με το Μέγιστο Κοινό Διαιρέτη τους
- Σε περίπτωση που η διαδικασία φτάσει μέχρι το σημείο που $a = 1$ ή $b = 1$ τότε οι δύο αριθμοί δεν έχουν μη τετριμμένο GCD, δηλαδή μεγαλύτερο του 1
- Παράδειγμα ($a = 196, b = 42$)

Βήμα	A	B
1	196	42
2	154	42
3	112	42
4	70	42
5	28	42
6	28	14
7	14	14

- Το αποτέλεσμα είναι: $\text{gcd}(196, 42) = 14$

Διαχωρισμός του διαγράμματος ροής σε καταστάσεις



Τεχνικές περιγραφής ενός μη προγραμματιζόμενου επεξεργαστή GCD

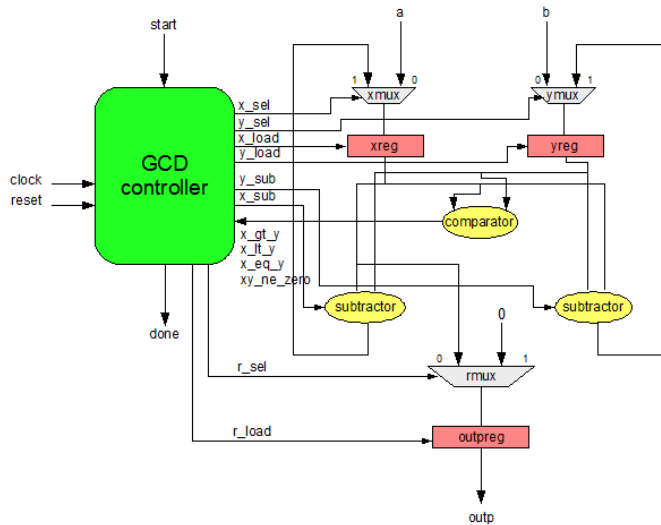
- Μη συνθέσιμη περιγραφή
 - Περιγραφή της συμπεριφοράς του επεξεργαστή σε αλγοριθμικό επίπεδο
 - Χρήση δομών υψηλού επιπέδου όπως δομές επανάληψης `for` και `while`
 - Συχνά χρησιμοποιείται ως τμήμα ενός testbench για την προσομοίωση της συμπεριφοράς του κυκλώματος και την εξαγωγή εξόδων αναφοράς (reference vectors)
- Συνθέσιμη περιγραφή
 - Υλοποίηση με χειριστή ελέγχου (τύπου FSM) και χειριστή δεδομένων (datapath)
 - Υλοποίηση με αρχιτεκτονική FSM/D
 - Για κάθε κατάσταση του FSM/D, σε κάθε καταχωρητή γίνεται ανάθεση νέας τιμής μόνο μία φορά

Μη συνθέσιμη περιγραφή του GCD

- Έστω A, B οι είσοδοι δεδομένων και Y η έξοδος με την τιμή του GCD
- Υλοποίηση με ένα μπλοκ λογικής `always`

```
always @(A or B)
begin: GCD
  A_reg = A;
  B_reg = B;
  if (A_reg != 0 && B_reg != 0)
  begin
    while (A_reg != B_reg)
    begin
      if (A_reg >= B_reg)
        A_reg = A_reg - B_reg;
      else
        B_reg = B_reg - A_reg;
    end
  end
  else
  begin
    A_reg = 0;
  end
  Y = A_reg;
end
```

Το συνολικό κύκλωμα του επεξεργαστή GCD



Περιγραφή της υλοποίησης FSMD του επεξεργαστή GCD (1)

```

module gcd (
    clock, reset, start,
    a, b,
    outp, done);
    parameter WIDTH = 8;
    parameter s0 = 2'b00, s1 = 2'b01,
              s2 = 2'b10, s3 = 2'b11;
    input clock, reset, start;
    input [WIDTH-1:0] a, b;
    output [WIDTH-1:0] outp;
    output done;
    reg [WIDTH-1:0] outp;
    reg done;
    reg [1:0] state;
    reg [WIDTH-1:0] x, y;

    always @(posedge clock or posedge reset)
    begin
        done = 1'b0;
        if (reset) begin
            state = s0;
            x = 0;
            y = 0;
            done = 1'b0;
            outp = 0;
        end
    end
end
    
```

```

else begin
    case (state)
    s0: begin
        if (start) begin
            x = a;
            y = b;
            state = s1;
        end
    end
    else
        state = s0;
    end
end
s1: begin
    if (x != 0 && y != 0)
        state = s2;
    else begin
        outp = 0;
        state = s3;
    end
end
end
    
```

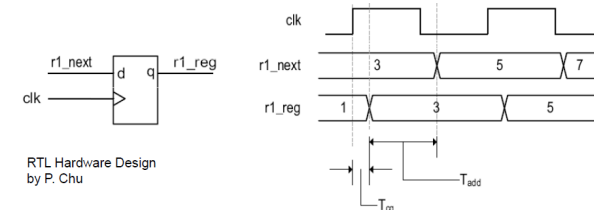
Περιγραφή της υλοποίησης FSMD του επεξεργαστή GCD (2)

```

s2: begin
    if (x > y) begin
        x = x - y;
        state = s2;
    end
    else if (x < y) begin
        y = y - x;
        state = s2;
    end
    else begin
        outp = x;
        state = s3;
    end
end
s3: begin
    done = 1'b1;
    state = s0;
end
endcase
end
end
endmodule
    
```

Τεχνική περιγραφής με ζεύγη σημάτων *_next, *_reg

- Πολλές φορές προτιμάται ο διαχωρισμός των σημάτων εγγραφής (*_next) και ανάγνωσης (*_reg) της τιμής ενός καταχωρητή
- Κάθε φυσικός καταχωρητής του FSMD περιγράφεται από δύο τέτοια σήματα
- Στο αριστερό μέλος των αναθέσεων (στη λογική επόμενης κατάστασης) εμφανίζεται η τιμή προς εγγραφή
- Στο δεξί μέλος των αναθέσεων χρησιμοποιείται η τιμή που ήδη υπάρχει στον καταχωρητή



Εναλλακτική υλοποίηση του FSMD για τον επεξεργαστή GCD (1)

```
module gcd (
  clock, reset, start,
  a, b,
  outp, done);
  parameter WIDTH = 8;
  parameter s0 = 2'b00, s1 = 2'b01,
             s2 = 2'b10, s3 = 2'b11;
  input clock, reset, start;
  input [WIDTH-1:0] a, b;
  output [WIDTH-1:0] outp;
  output done;
  reg done;
  reg [WIDTH-1:0] outp;
  reg [1:0] state;
  reg [WIDTH-1:0] x_reg, x_next;
  reg [WIDTH-1:0] y_reg, y_next;

  always @(posedge clock or posedge reset)
  begin
    done = 1'b0;
    // Assignment to *_reg signals at reset
    if (reset) begin
      state = s0;
      x_reg = 0;
      y_reg = 0;
      done = 1'b0;
      outp = 0;
    end
  end
```

```
else begin
  // Assignment to *_reg signals for
  // storing the old results
  x_reg = x_next;
  y_reg = y_next;
  case (state)
    s0: begin
      if (start) begin
        x_next = a;
        y_next = b;
        state = s1;
      end
    end
    else
      state = s0;
  end
  s1: begin
    if (x_reg != 0 && y_reg != 0)
      state = s2;
    else begin
      outp = 0;
      state = s3;
    end
  end
end
```

Εναλλακτική υλοποίηση του FSMD για τον επεξεργαστή GCD (2)

```
s2: begin
  if (x_reg > y_reg) begin
    x_next = x_reg - y_reg;
    state = s2;
  end
  else if (x_reg < y_reg) begin
    y_next = y_reg - x_reg;
    state = s2;
  end
  else
    state = s3;
end
s3: begin
  done = 1'b1;
  outp = x_reg;
  state = s0;
end
endcase
end
endmodule
```

Περιγραφές ελέγχου/επαλήθευσης λειτουργίας (testbenches)

- Σε μία σχεδίαση, το testbench αντιστοιχεί στο υψηλότερο επίπεδο ιεραρχίας
- Το module ενός testbench δεν περιλαμβάνει καμία δήλωση θύρας, μπορεί όμως να περιλαμβάνει δηλώσεις για διάφορες παραμέτρους και βοηθητικές μεταβλητές
- Στο testbench, ελέγχεται η λειτουργία του top-level module του συνολικού κυκλώματος
- Σε ένα testbench μπορεί να συμβαίνουν τα εξής:
 - Παραγωγή σημάτων διέγερσης από μη-συνθέσιμο μοντέλο του κυκλώματος και άμεση σύγκριση των αναμενόμενων με τις πραγματικές τιμές εξόδου κατά την προσομοίωση
 - Ανάγνωση εξωτερικού αρχείου με διανύσματα αναφοράς (reference vectors) για τη λήψη εισόδων και τη σύγκριση εξόδων
 - Εγγραφή αποτελεσμάτων και άλλων πληροφοριών σε εξωτερικό αρχείο

Οι διεργασίες \$readmemb και \$readmemh

- Διεργασίες για τη φόρτωση πληροφορίας που βρίσκεται αποθηκευμένη σε εξωτερικό αρχείο, σε μία μνήμη Verilog

```
$readmemx("filename", <memname>, <<saddr> <, <faddr>>?>?);
```

- όπου: x είναι b (δυναμικό σύστημα), ή h (δεκαεξαδικό)
- <memname>: όνομα της μνήμης
- <saddr>: αρχική διεύθυνση της μνήμης για καταχώρηση. Αν δεν καθοριστεί, τα δεδομένα αποθηκεύονται από τη μικρότερη διεύθυνση
- <faddr>: τελική διεύθυνση της μνήμης για καταχώρηση
- Το αρχείο μπορεί να περιέχει μόνο αριθμητικές τιμές και (προαιρετικά) σχόλια της Verilog
- Παραδείγματα

```
parameter WIDTH = 8, TESTS = 20;
reg [WIDTH-1:0] a_b_yref_arr [1:TESTS*3];
...
$readmemh("test_data_hex.txt", a_b_yref_arr);
```

Διεργασίες διαχείρισης αρχείων δεδομένων

- Οι διεργασίες `$fopen` και `$fclose` χρησιμοποιούνται για το άνοιγμα και το κλείσιμο ενός αρχείου του χρήστη
- Οι διεργασίες `$fdisplay`, `$fwrite`, `$fmonitor` δέχονται τα ίδια ορίσματα με τις αντίστοιχες διεργασίες που απευθύνονται στην έξοδο κονσόλας (`$display`, `$write`, `$monitor`), με τη διαφορά ενός επιπλέον ορίσματος που χρησιμοποιείται ως δείκτης αρχείου
- Ο δείκτης αποθηκεύεται ως `integer` των 32-bit. Κάθε bit του αντιστοιχεί σε διαφορετικό ρεύμα αρχείου
- Χρήση της `$fopen`: `descriptor = $fopen("name of file");`, επιστρέφοντας 0 στην περίπτωση που δεν υπάρχει το αρχείο για να προσπελαστεί
- Χρήση της `$fclose`: `$fclose(descriptor);`
- Το ψηφίο LSB του δείκτη αρχείου αντιστοιχεί στην τυποποιημένη έξοδο (standard output)

Η διεργασία \$random

- Η διεργασία `$random` παρέχει ένα μηχανισμό για τη γέννηση ψευδοτυχαίων αριθμών
- Ένας νέος αριθμός παράγεται με κάθε κλήση της `$random`
- Παράγει αριθμούς τύπου `integer` (32-bit)
- Παράδειγμα

```
parameter SEED = 17;
reg [31:0] vector;

always @(posedge clock)
    vector = $random(SEED);
```

Η διαμόρφωση αρχείου VCD: Value Change Dump

- Ένα αρχείο VCD περιέχει πληροφορία σχετικά με τις αλλαγές τιμών επιλεγμένων μεταβλητών (εισόδων, εξόδων και εσωτερικών μεταβλητών) σε μία περιγραφή Verilog
- Αποτελεί αρχείο ASCII το οποίο οπτικοποιείται με τη μορφή κυματομορφών των μεταβλητών του κυκλώματος, με τη βοήθεια ειδικών εργαλείων
- Γέννηση αρχείου VCD

```
// Define VCD file name.
$dumpfile("filename.vcd");

// Set active only one of the following calls to $dumpvars

// Dump all the variables in the model to the VCD file.
$dumpvars;

// Dump all variables in top and no variables in modules instantiated by top.
$dumpvars(1, top);

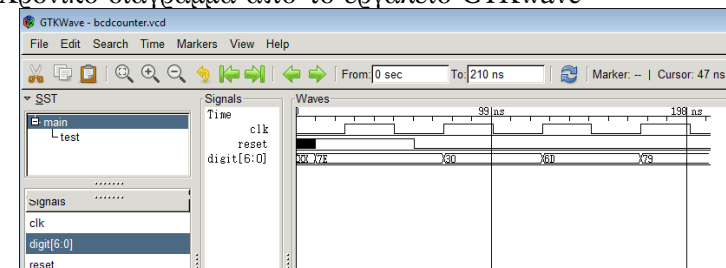
// Dump all variables in top and all modules instantiated by top.
$dumpvars(0, top);
```

Δημιουργία του αρχείου VCD και οπτικοποίηση κυματομορφών

- Μπλοκ λογικής για τη δημιουργία αρχείου VCD

```
// Write simulation data to a VCD waveform file.
initial begin
    // Open a VCD file for writing
    $dumpfile("bcdcounter.vcd");
    // Dump the values of all nets and wires in module "main", seen from
    // hierarchy level 0
    $dumpvars(0, main);
end
```

- Χρονικό διάγραμμα από το εργαλείο GTKwave



Παραγωγή σήματος ρολογιού για χρήση σε testbench

- Για τον έλεγχο της λειτουργίας σύγχρονων κυκλωμάτων χρειάζεται η δημιουργία μιας εικονικής γεννήτριας ρολογιού
- Περιγράφεται σε ξεχωριστό μπλοκ `always` στο testbench σε σχέση με τη διέγερση των άλλων εισόδων του κυκλώματος
- Παράδειγμα από το σχεδιασμό του BCD counter

```
// file bcdcounter.v
module bcdcounter (clk, reset, digit);
  input   clk, reset;
  output [6:0] digit;
  reg [6:0] digit;
// file bcdcounter_tb.v
parameter ClockPeriod = 50;
parameter ClockHalfPeriod = ClockPeriod/2;
...
// Test clock generation.
initial
  clk = 1'b0;
always
  #ClockHalfPeriod clk = ~clk;
...
```

Έλεγχος ορθής λειτουργίας του επεξεργαστή GCD με testbench (1)

- Τα χαρακτηριστικά του testbench για την επαλήθευση της ορθής συμπεριφοράς του επεξεργαστή GCD
- Λήψη εισόδων από αρχείο με χρήση μεταβλητών
- Σύγκριση αποτελεσμάτων με τα διανύσματα αναφοράς από το αρχείο
- Προσθήκη απαριθμητή επιδόσεων (performance counter) για τη λήψη του αναλυτικού προφίλ εκτέλεσης του επεξεργαστή GCD για διαφορετικές εισόδους

```
...
// Profiling signals
integer ncycles;
...
@always(posedge clock or posedge reset) // PROFILING
begin
  if (reset)
    ncycles = 0;
  else
    ncycles = ncycles + 1;
end
```

Έλεγχος ορθής λειτουργίας του επεξεργαστή GCD με testbench (2)

- Κλήση της διεργασίας `$readmemh` για την ανάγνωση δεδομένων εισόδου από αρχείο
- Κλήση της διεργασίας `$fopen` και εκτύπωση διαγνωστικής εξόδου σε αρχείο

```
...
$readmemh("gcd_test_data_hex.txt", AB_Y_Ref_Arr);
...
SimResults = $fopen("gcd_alg_test_results.txt");
```

- Τα περιεχόμενα του αρχείου κειμένου "gcd_test_data_hex.txt" (A, B, result)

```
15 31 07 // Decimal: 21 49 7
19 1E 05 // Decimal: 25 30 5
13 1B 01 // Decimal: 19 27 1
28 28 28 // Decimal: 40 40 40
FA 6E 0A // Decimal: 250 190 10
05 FA 05 // Decimal: 5 250 5
01 01 01 // Decimal: 1 1 1
00 00 00 // Decimal: 0 0 0
```

Έλεγχος ορθής λειτουργίας του επεξεργαστή GCD με testbench (3)

Αρχείο testbench

```
`timescale 1 ns / 10 ps
module gcd_main;
  parameter WIDTH = 8;
  reg clock, reset, start;
  reg [WIDTH-1:0] a, b, y, y_ref;
  reg [WIDTH-1:0] a_reg, b_reg;
  wire [WIDTH-1:0] outp;
  wire done;
  //
  parameter GCD_tests = 8;
  integer N,M;
  reg Passed, FailTime;
  integer SimResults;
  // Declare memory array for test data
  reg [WIDTH-1:0] ab_y_ref_arr[1:GCD_tests*3];
  // Profiling counter
  integer ncycles;
  // Test the GCD implementation
  gcd test(
    .clock(clock), .reset(reset), .start(start),
    .a(a), .b(b),
    .outp(outp), .done(done)
  );
```

Έλεγχος ορθής λειτουργίας του επεξεργαστή GCD με testbench (4)

Αρχείο testbench (συνέχεια)

```
// Test clock generation.
//
// Clock pulse starts from 0.
initial
  clock = 1'b0;
// Free-running clock
always
  #25 clock = ~clock;

// PROFILING
always @(posedge clock or posedge reset)
begin
  if (reset)
    ncycles = 0;
  else
    ncycles = ncycles + 1;
end

// Test GCD algorithm
initial
begin
  // Load contents of "gcd_test_data_hex.txt" into array.
  $readmemh("gcd_test_data_hex.txt", ab_y_ref_arr);
  // Open simulation results file.
  SimResults = $fopen("gcd_alg_test_results.txt");
  Passed = 1; // Set to 0 if fails
  for (N = 0; N < GCD_tests; N = N + 1)
  begin
```

Νικόλαος Καββαδίας nkavn@uop.gr

Γλώσσες Περιγραφής Υλικού I

Έλεγχος ορθής λειτουργίας του επεξεργαστή GCD με testbench (5)

Αρχείο testbench (συνέχεια)

```
a = ab_y_ref_arr[(N*3)+1];
b = ab_y_ref_arr[(N*3)+2];
y_ref = ab_y_ref_arr[(N*3)+3];
////////////////////////////////////
reset = 1;
#50 reset = 0; start = 1;
////////////////////////////////////
wait (done);
#50;
// Diagnostics for GCD algorithm
if (outp != y_ref) begin
  Passed = 0;
  $fdisplay(SimResults, "Error: A=%d, B=%d. Y=%d (%d)", a, b, outp, y_ref);
end
else begin
  $fdisplay(SimResults, "OK: Number of cycles = %d", ncycles);
end
end
if (Passed == 1) begin
  $fdisplay(SimResults, "GCD algorithm test has passed");
  $fclose(SimResults);
  $finish;
end
end
endmodule
```

Νικόλαος Καββαδίας nkavn@uop.gr

Γλώσσες Περιγραφής Υλικού I

Έλεγχος ορθής λειτουργίας του επεξεργαστή GCD με testbench (6)

- Εκτύπωση διαγνωστικής εξόδου στο αρχείο "gcd_alg_test_results.txt"

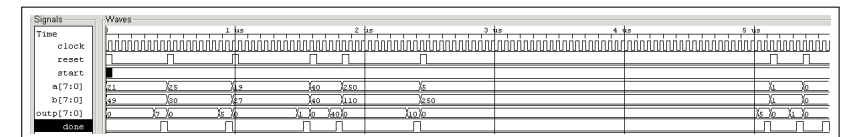
```
OK: Number of cycles = 8
OK: Number of cycles = 9
OK: Number of cycles = 11
OK: Number of cycles = 4
OK: Number of cycles = 11
OK: Number of cycles = 53
OK: Number of cycles = 4
OK: Number of cycles = 3
GCD algorithm test has passed
```

Νικόλαος Καββαδίας nkavn@uop.gr

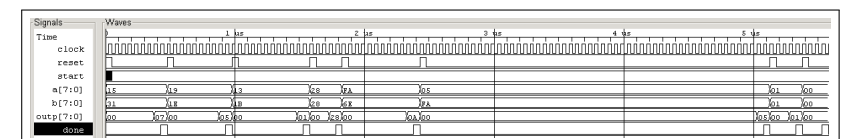
Γλώσσες Περιγραφής Υλικού I

Προσομοίωση του επεξεργαστή GCD

Χρονικό διάγραμμα (τιμές στο δεκαδικό)



Χρονικό διάγραμμα (τιμές στο δεκαεξαδικό)



Νικόλαος Καββαδίας nkavn@uop.gr

Γλώσσες Περιγραφής Υλικού I