**1η  Εργαστηριακή Άσκηση Σχεδιασμός Συστημάτων VLSI**

**Κόλλιας Ιωάννης 1084578**

**Ποδηματά Θεοδώρα Παναγιώτα 1090031**

**4ο Έτος 2023-2024**

**Ερώτημα 1**

Επιθυμούμε να ελέγξουμε τη λειτουργία ενός 9-bit rca για όλες τις πιθανές εισόδους. Στο κώδικα του adder\_td.v και rca.v αλλάζουμε την παράμετρο του width αφού χρειαζόμαστε 9 bit(parameter width = 9). Για να δουλεύει σωστά το simulation χρειαζόμαστε ένα breakpoint στη σειρά 25 με το display και στο κώδικα του rca.v **αλλάζουμε assign temp\_c[0] = 1'b0; σε assign temp\_c[0] = 1'b1;**

Εικόνα που περιέχει κείμενο, λογισμικό, λογισμικό πολυμέσων, εικονίδιο υπολογιστή

Περιγραφή που δημιουργήθηκε αυτόματα

Αν δεν κάνουμε την αλλαγή:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, εικονίδιο υπολογιστή

Περιγραφή που δημιουργήθηκε αυτόματα

**Ερώτημα 2**

Στο δεύτερο ερώτημα μας ζητείται να φτιάξομε ένα testbench που θα ελέγχει τη λειτουργία του mac. Αρχικά ορίζουμε τις παραμέτρους για την είσοδο και την έξοδο. Έπειτα, δηλώνουμε όλα τα στοιχεία που χρησιμοποιούμε στο κύκλωμά μας και στο mac#(…) συμπληρώνουμε πάλι τις παραμέτρους και τα απαραίτητα στοιχεία. Για να γίνει ο έλεγχος ορίζουμε το ρολόι να εναλλάσσετε από 0 σε 1 με περίοδο 10ns και ενώ το reset=0, μετά πρέπει να γίνει 1 για να ξεκινήσει η λειτουργία του κυκλώματος. Τέλος, για κάθε θετική ακμή ου ρολογιού παράγονται 2 τυχαίοι αριθμοί και εμφανίζεται το άθροισμά τους. Αν ταυτίζονται τα 2 αθροίσματα result=out τότε το check (ο έλεγχος) είναι 1, άρα σωστό. Αντιθέτως θα είχαμε x, απροσδιόριστο.

Εικόνα που περιέχει διάγραμμα, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει στιγμιότυπο οθόνης, κείμενο, γραμμή, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, αριθμός, οθόνη

Περιγραφή που δημιουργήθηκε αυτόματα

module mac\_tb;

parameter inwidth=4;

parameter outwidth=9;

parameter delay = 5;

reg [inwidth-1:0] a,b;

reg [outwidth-1:0] result;

reg check,clock=0,reset;

wire [outwidth-1:0] out;

mac #(.iwidth(inwidth),.swidth(outwidth)) DUT (

.clk(clock),

.rstn(reset),

.a\_i(a),

.b\_i(b),

.sum\_o(out)

);

initial begin

forever

#(delay) clock<=~clock;

end

initial begin

result<=0;

reset<=0;

#(delay) reset=1;

forever begin

@(posedge clock) begin

a = $random % (2<<inwidth);

b = $random % (2<<inwidth);

result=a\*b+out;

#(1)

if (out == result) check=1'b1;

else check=1'bX;

end

end

end

endmodule

**Ερώτημα 3**

Σκοπός του ερωτήματος αυτού είναι η δημιουργία δύο μετρητών. Ο πρώτος μετρητής c0, μετράει κανονικά από το 0 έως το 15 και στη συνέχεια παίρνει κάθε χρονική στιγμή που έχει την τιμή 15, το όρισμα του δεύτερου μετρητή c1. Έπειτα, προσθέτει 1 στην τιμή του c1 και συνεχίζει να μετράει από αυτό. Ο δεύτερος μετρητής μετράει σταθερά 0 για 15 κύκλους περιόδου(10ns), 1 για τους επόμενους 15 κύκλους κλπ.

c0: 0, 1, …, 15, 1, 2, …, 15, 2, 3, …, 15, 3, 4, …, 15, …, …, 14, 15, 15, 0, 1, …, 15, …,

c1: 0, 0, …, 0, 1, 1, …, 1, 2, 2, …, 2, 3, 3, …, 3, …, …, 14, 14, 15, 0, 0, …., 0, …,

Για την υλοποίηση του ερωτήματος χρειαστήκαμε 1 ρολόι, 2 flip-flop που αποτελούν τους μετρητές με 4bit έξοδο, 2 πολυπλέκτες 2-1 και έναν συγκριτή που ελέγχει πότε το σήμα φτάνει στο 15. Το πρώτο flip-flop παίρνει ως είσοδο την έξοδο του πρώτου πολυπλέκτη, ο οποίος έχει ως enable την έξοδο του συγκριτή(15). Αναλυτικότερα, αν ο συγκριτής γίνει 15 τότε δίνει έξοδο 1(του 1ου πολυπλέκτη) και ο πολυκλέκτης παίρνει τιμή από την έξοδο του 2ου μετρητή(flip-flop), προσθέτει 1 και την βάζει σαν είσοδο στον 1ο μετρητή. Ειδάλλως, ο συγκριτής δεν είναι 15 και ο 1ος μετρητής μετράει κανονικά ανά 1.

Όσον αφορά τον δεύτερο μετρητή όταν ο συγκριτής δεν είναι 15 και περνάει το 0 του πολυπλέκτη, η είσοδος είναι ίδια με την έξοδο (μένει σταθερή), αλλιώς παίρνουμε την τιμή εξόδου του πολυπλέκτη, της προσθέτουμε 1 και μένει πάλι σταθερή μέχρι ο 1ος μετρητής να γίνει πάλι 15.

**Κώδικας:**

module counter(clock,reset,c0,c1);

input clock,reset;

output [3:0] c0;

output [3:0] c1;

reg [3:0] c0value;

reg [3:0] c1value;

wire[3:0] mux0,c1plus,c0plus;

wire comparator;

assign c1plus=c1value+1;

assign c0plus=mux0+1;

assign comparator=(c0value==4'd15)?1:0;

assign mux0=(comparator)?c1value:c0value;

always @(posedge clock or negedge reset) begin

if(reset==1'b0)c0value=4'b0;

else c0value<=c0plus;

end

always@(posedge clock or negedge reset) begin

if(reset==1'b0)c1value=4'b0;

else c1value=(comparator)?c1plus:c1value;

end

assign c1=c1value;

assign c0=c0value;

endmodule

Εικόνα που περιέχει διάγραμμα, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, εικονίδιο υπολογιστή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, λογισμικό, λογισμικό πολυμέσων, εικονίδιο υπολογιστή

Περιγραφή που δημιουργήθηκε αυτόματα

Σας παραθέτουμε και τον κώδικα του testbench όπου ελέγχει την σωστή λειτουργία του κυκλώματός μας.

module test;

parameter delay=5;

reg clock,reset;

reg check;

wire [3:0] out0,out1;

reg[3:0] i,j;

counter DUT (

.clock(clock),

.reset(reset),

.c0(out0),

.c1(out1)

);

initial begin

reset=0;

clock=0;

j=0;

i=0;

#(delay) reset=1;

forever

#5 clock<=~clock;

end

initial

begin

forever begin

@(posedge clock) begin

if(i==15&&j!=15)begin

j=j+1;

i=j;

end

else if(j==15)begin

i=0;

j=0;

end

else if(i!=15) i=i+1;

#1

if(i==out0&&j==out1)

check=1'b1;

else

check=1'bx;

end

end

end

endmodule

Εικόνα που περιέχει στιγμιότυπο οθόνης, γραμμή, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, παράλληλα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, παράλληλα

Περιγραφή που δημιουργήθηκε αυτόματα