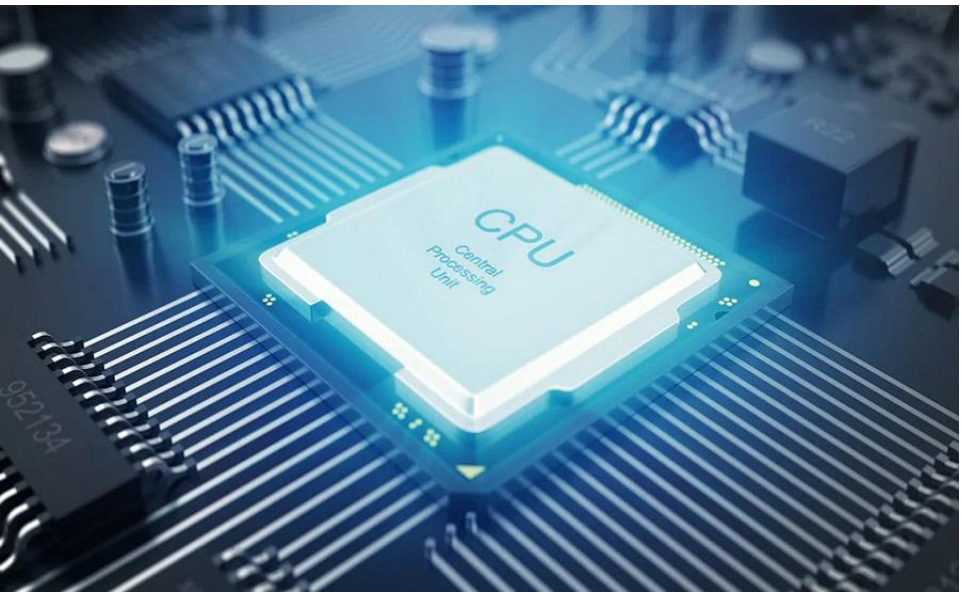
Ψηφιακά Συστήματα HW σε Χαμηλά Επίπεδα Λογικής ΙΙ



ΚΑΡΑΜΗΤΟΠΟΥΛΟΣ ΠΑΝΑΓΙΩΤΗΣ

ΑΕΜ 9743

email: karamitopp@ece.auth.gr

Περιεχόμενα

1 Up and Down Counter

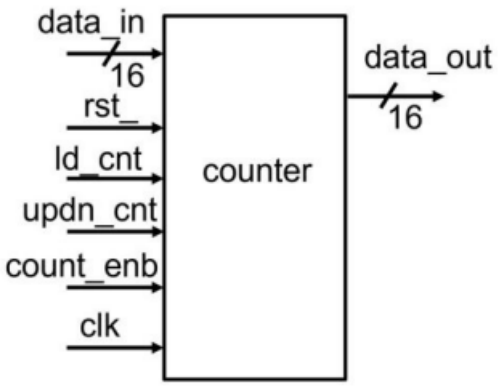


Figure 1 Top level diagram of the 16 bit Up/Down Counter

1.1 Περιγραφή εισόδων/εξόδων

Ο 16 bit Up/Down Counter έχει τις εξής εισόδους:

- Δεδομένα εισόδου 16bit (data_in)
- Ένα σήμα ρολογιού (clk)
- Ένα σήμα enable (active High count_enb)
- Ένα σήμα load (active Low ld_cnt)
- Ένα σήμα που ρυθμίζει αν ο counter μετράει προς πάνω ή προς τα κάτω (updn_cnt)
- Ένα ασύγχρονο σήμα αρχικοποίησης (active Low rst_)

Ως έξοδο έχει τα δεδομένων εξόδου 16bit (data_out)

1.2 Περιγραφή της λειτουργίας του κυκλώματος

- Ο counter λειτουργεί στην θετική ακμή του ρολογιού, και έχει ενα ασύγχρονο reset (active low). Όταν το ld_cnt είναι στο λογικό 0, τότε φορτώνονται τα δεδομένα εισόδου, στην έξοδο του counter. Η λειτουργία φόρτωσης έχει μεγαλύτερη προτεραιότητα από την λειτουργία αύξησης/μείωσης τις τιμής του counter (count_enb). Καθώς το count_enb βρίσκεται στο λογικό 1, αν το updn_cnt βρίσκεται στο λογικό 1, τότε ο counter μετράει προς τα πάνω, αλλίως αν το updn_cnt βρίσκεται στο λογικό 0, τότε ο counter μετράει προς τα κάτω, αν το count_enb βρίσκεται στο λογικό 0, ο μετρητής διατηρεί τα δεδομένα εξόδου σταθερά.

1.3 System Verilog Code

Χρησιμοποιώντας την γλώσσα περιγραφής υλικού System Verilog, ο κώδικας που αντιστοιχεί στην παραπάνω περιγραφή του Up/Down counter φαίνεται στην εικ. 2. Οι είσοδοι/έξοδοι του κυκλώματος είναι logic.

```
module UpDownCounter(input logic clk, rst_, ld_cnt, updn_cnt, count_enb, input logic [15:0] data_in, output logic [15:0] data_out);

always_ff @(posedge clk, negedge rst_)
begin
        if(!rst_)
                data_out <= 16'b0;
        else if(!ld_cnt)
                data_out <= data_in;
        else if(count_enb)
                begin
                if(updn_cnt)
                        data_out <= data_out + 1'b1;
                else
                        data_out <= data_out - 1'b1;
                end
        else
                data_out <= data_out;
end
endmodule
```

Figure 2: System Verilog Code Up/Down Counter

1.4 Testbench

Για την επαλήθευση της σωστής λειτουργίας του κυκλώματος

δημιουργήθηκε το αρχείο δοκιμής, το οποίο φαίνεται στην εικ. 3.

```
module test_UpDownCounter;

bit rst_ = 1;
bit ld_cnt = 1;
bit clk, updn_cnt, count_enb;
bit [15:0] data_in;
wire [15:0] data_out;

UpDownCounter dut(.rst_(rst_), .clk(clk), .ld_cnt(ld_cnt), .updn_cnt(updn_cnt), .count_enb(count_enb), .data_in(data_in), .data_out(data_out));
bind dut SVA_UpDownCounter pdut (.clk(clk), .rst_(rst_), .ld_cnt(ld_cnt), .count_enb(count_enb), .updn_cnt(updn_cnt), .data_out(data_out));

always #1 clk = ~clk;

always @(posedge clk)
 $display($stime,,,"clk=%b rst_=%b ld_cnt=%b updn_cnt=%b count_enb=%b data_out=%d",clk, rst_, ld_cnt, updn_cnt, count_enb, data_out);

initial
begin
updn_cnt = 0;
count_enb = 0;
data_in = 16'b1111111;
rst_ = 0;
#2 rst_ = 1;
count_enb = 1;
updn_cnt = 1;
#20 count_enb = 0;
#4 ld_cnt = 0;
#2 ld_cnt = 1;
#12 rst_ = 0;
#2 rst_ = 1;
#8 count_enb = 1;
#12 updn_cnt = 0;
#22 ld_cnt = 0;
#2 ld_cnt = 1;
#4 count_enb = 0;
#4 rst_ = 0;
#2 $finish;
end
endmodule
```

Figure 3: Testbench

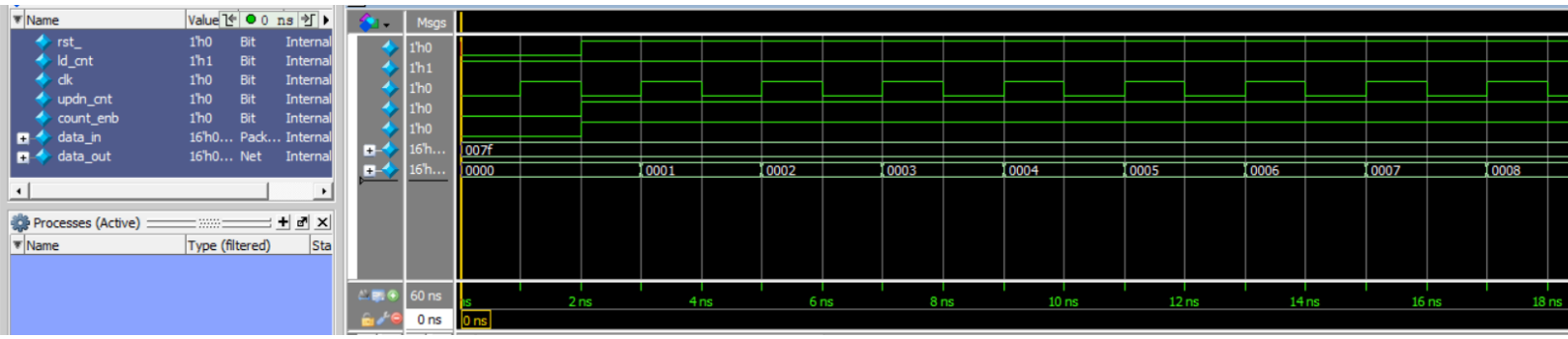


Figure 4

Εικ. 4: Αρχικά το reset πέφτει στο λογικό 0, ώστε να αρχικοποιηθεί η έξοδος του counter στο 0. Έπειτα στα 2ns το rst_, updn_cnt, count_enb οδηγούνται στο λογικό 1 (αφού το updn_cnt είναι στο λογικό 1 ο counter θα μετράει προς τα πάνω). Στην επόμενη θετική ακμή του ρολογίου (3ns) η έξοδος του counter πηγαίνει στο 1 (0001 δεκαεξαδικό). Στην μεθεπόμενη θετική ακμή του ρολογίου (5ns) η έξοδος του counter γίνεται 2 (0002 δεκαεξαδικό) κ.ο.κ.

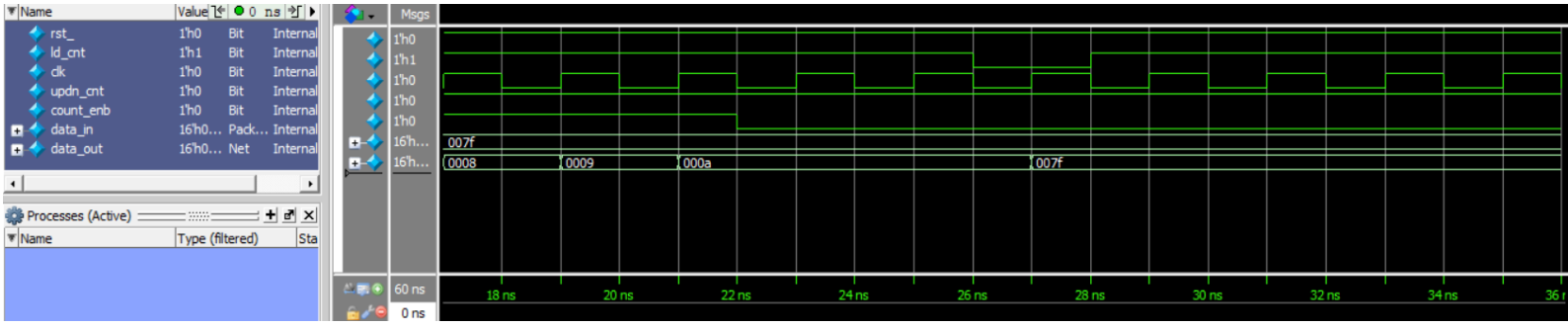


Figure 5

Εικ. 5: Καθώς το rst_, updn_cnt, count_enb βρίσκονται στο λογικό 1, ο counter συνεχίζει να μετρά προς τα πάνω. Την στιγμή 22ns το count_enb πηγαίνει στο λογικό 0, οπότε ο μετρήτης σταματά να μετράει προς τα άνω/κάτω και τα δεδομένα εξόδου παραμένουν σταθερά (000a) όσο το count_enb είναι 0. Την στιγμή 26ns το ld_cnt γίνεται 0, και στην επόμενη θετική ακμή του ρολογίου (27ns) φορτώνονται στην έξοδο τα δεδομένα εισόδου (data_in -> 007f δεκαεξαδικό). Μέχρι και την χρονική στιγμή 40ns (και Εικ. 6) δεν θα αυξήθει η τιμή της εξόδου του μετρητή (παραμένει 007f) γιατί το count_enb είναι 0.

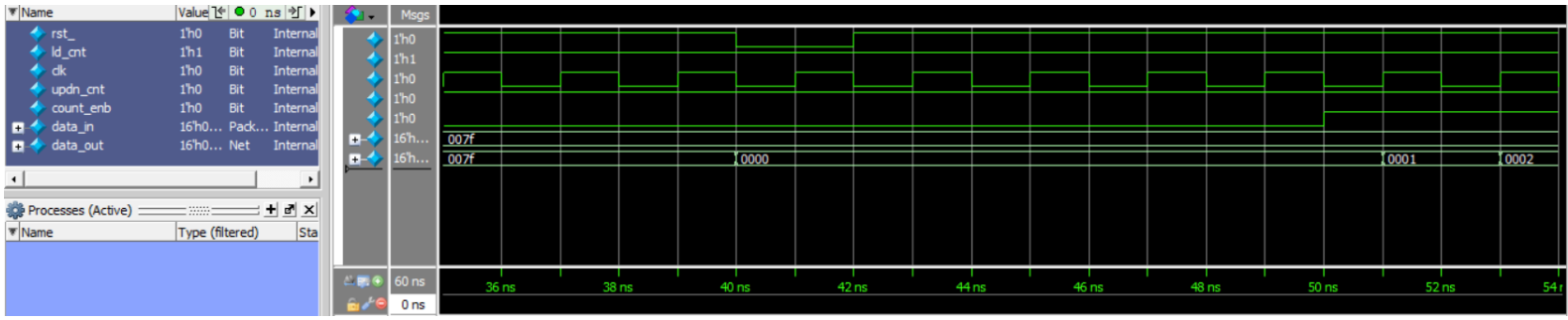


Figure 6

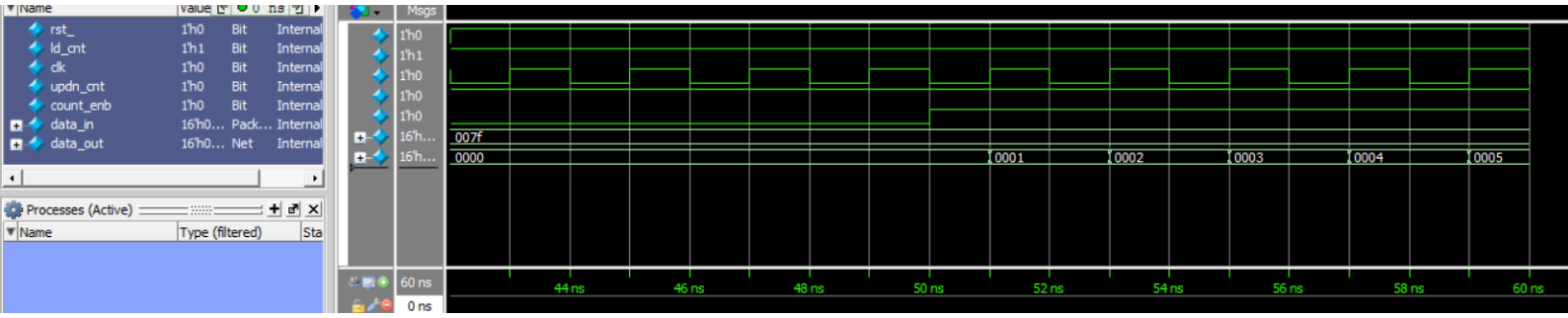


Figure 7

Εικ. 6 και Εικ. 7: Την χρονική στιγμή 40ns το rst_ πηγαίνει στο λογικό 0, οπότε η έξοδος του κυκλώματος ξανα-μηδενίζεται. Το count_enb συνεχίζει να είναι 0 και ld_cnt=1, rst_=1 μέχρι την στιγμή 50ns, οπότε η έξοδος του κυκλώματος μέχρι τότε παραμένει σταθερή (0000). Την 50ns γίνεται το count_enb 1, οπότε στην επόμενη θετική ακμή του ρολογίου (51ns), η τιμή της εξόδου του μετρήτη γίνεται 1 κ.ο.κ.

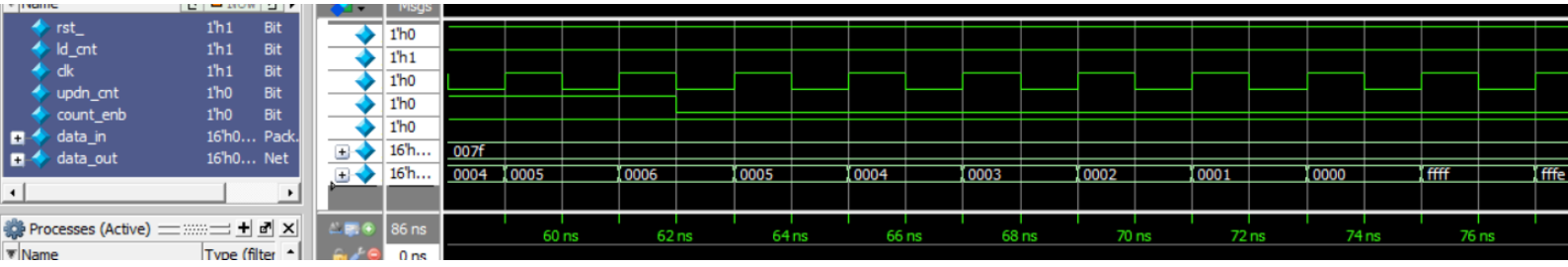


Figure 8

Εικ. 8: Από την χρονική στιγμή 62ns το updn_cnt γίνεται 0, το count_enb είναι ήδη 1, οπότε ο μετρητής θα αρχίσει να μετράει προς τα κάτω. Στην επόμενη θετική του ρολογιού (63ns) η έξοδος του μετρήτη μειώνεται στο

5 από το 6 που ήταν προηγουμένως. Την μεθεπόμενη θετική ακμή του ρολογιού (65ns) η έξοδος του μετρήτη γίνεται 4 κ.ο.κ. Την στιγμή 73ns ο μετρητής έχει φτάσει στην ελάχιστη δυνατή τιμή που μπορεί να πάρει (μηδέν). Στην επόμενη θετική ακμή του ρολογιού(75ns) η έξοδος μεταβαίνει από το μηδέν στην μέγιστη τιμή του counter (ffff δεκαεξαδικό).

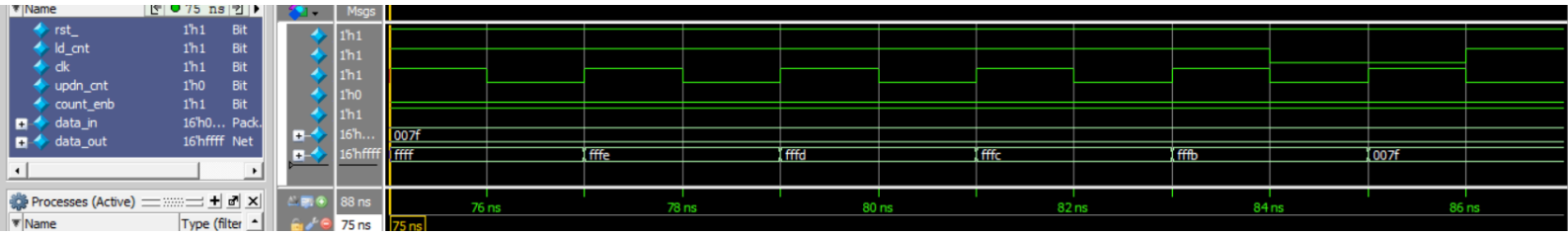


Figure 9

Εικ. 9: Η έξοδος του μετρητή συνεχίζει να μειώνεται σε κάθε θετική ακμή του ρολογίου αφού το updn_cnt είναι 0 και τα ld_cnt, count_enb είναι 1. Επομένως την στιγμή 77ns η έξοδος του μετρητή θα γίνει fffe, την στιγμή 79ns η έξοδος του μετρητή θα γίνει fffd, την στιγμή 81ns η έξοδος του μετρητή θα γίνει fffc, την στιγμή 83ns η έξοδος του μετρητή θα γίνει fffb. Έπειτα την στιγμή 84ns το ld_cnt γίνεται 0, οπότε την επόμενη θετική ακμή του ρολογίου (85ns) η είσοδος (007f) φορτώνεται στην έξοδο.

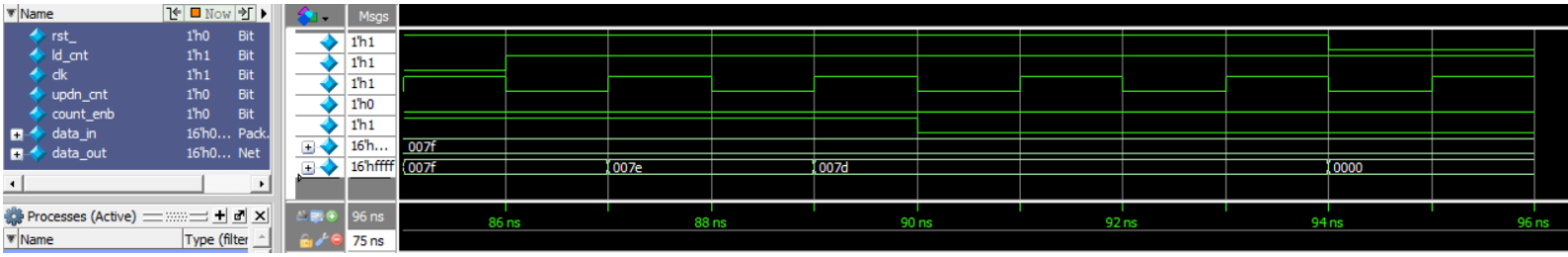


Figure 10

Εικ. 10: Η έξοδος του μετρητή συνεχίζει να μειώνεται σε κάθε θετική ακμή του ρολογίου αφού το updn_cnt είναι 0 και τα ld_cnt, count_enb είναι 1. Επομένως την στιγμή 87ns η έξοδος του μετρητή θα γίνει 007e, την στιγμή 89ns η έξοδος του μετρητή θα γίνει 007d. Την στιγμή 90ns το count_enb γίνεται 0, οπότε η έξοδος (007d) δεν θα αλλάξει όσο count_enb = 0, ld_cnt=1 και rst_=1. Ενώ την στιγμή 94ns το rst_ γίνεται 0, και μηδενίζεται ο counter.

1.5 SVA

Χρησιμοποιώντας SVA, οι properties παράγουν τα κατάλληλα μηνύματα (PASS/FAIL) στο κατάλληλο παράθυρο του προσομοιωτή, προκειμένου να γίνεται γνωστό αν ικανοποιούνται ή όχι οι εξής έλεγχοι:

- Όταν το reset είναι asserted, τότε η έξοδος του counter πρέπει να είναι 0.
- Αν το ld_cnt είναι deasserted (δηλ. είναι 1) και το count_enb είναι 0, τότε η έξοδος δεν πρέπει να έχει αλλάξει. Αυτήν η ιδιότητα πρέπει να απενεργοποιείται όταν το rst_ είναι στο 0.
- Αν το ld_cnt είναι deasserted (δηλ. είναι 1) και το count_enb είναι 1, τότε αν το updn_cnt είναι 1, η έξοδος πρέπει να έχει αυξηθεί, αλλίως αν το updn_cnt είναι 0, η έξοδος του counter πρέπει να έχει μειωθεί. Αυτήν η ιδιότητα πρέπει να απενεργοποιείται όταν το rst_ είναι στο 0.

Στην Εικ. 11 φαίνεται ο κώδικας ο οποίος αναπτύχθηκε για αυτόν τον σκοπό.

```
module SVA_UpDownCounter(input clk, rst_, ld_cnt, count_enb, updn_cnt, input [15:0] data_out);
property rst1;
        @(negedge rst_) 1'b1 |=> @(posedge clk)  (data_out == 0);

endproperty

property p1;
        @(posedge clk) disable iff(!rst_) (ld_cnt && !count_enb) |=> $stable(data_out);
endproperty
property p2;
        @(posedge clk) disable iff(!rst_) (ld_cnt && count_enb) |-> if(updn_cnt) ##1 (data_out ==  $past(data_out, 1) + 1'b1)  else ##1 (data_out ==  $past(data_out, 1) - 1'b1);
endproperty

reset1: assert property (rst1) else $display($time,,, "\t\t %m FAIL");
reset: cover property(rst1) $display($time,,, "\t\t %m PASS");

propert1: assert property (p1) else $display($time,,, "\t\t %m FAIL");
property1: cover property (p1) $display($time,,, "\t\t %m PASS");

propert2: assert property (p2) else $display($time,,, "\t\t %m FAIL");
property2: cover property (p2) $display($time,,, "\t\t %m PASS");
endmodule
```

Figure 11

$reset \rightarrow$ έλεγχος Νο 1 εκφωνήσης

$property1 \rightarrow$ έλεγχος Νο 2 εκφωνήσης

$property2 \rightarrow$ έλεγχος Νο 3 εκφωνήσης

Στις Εικ. 12, 13 και 14 φαίνονται τα αποτελέσματα που τυπώνονται στην κονσόλα του προσομοιωτή. Όλοι οι πρόσθετοι έλεγχοι είναι PASS, επαληθεύεται έτσι η σωστή συμπεριφορά του counter. Πιο συγκεκριμένα:

```
 1  clk=1 rst_=0 ld_cnt=1 updn_cnt=0 count_enb=0 data_out=    0
          1     test_UpDownCounter.dut.pdut.reset PASS
 3  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=1 data_out=    0
 5  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=1 data_out=    1
          5     test_UpDownCounter.dut.pdut.property2 PASS
 7  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=1 data_out=    2
          7     test_UpDownCounter.dut.pdut.property2 PASS
 9  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=1 data_out=    3
          9     test_UpDownCounter.dut.pdut.property2 PASS
11  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=1 data_out=    4
         11     test_UpDownCounter.dut.pdut.property2 PASS
13  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=1 data_out=    5
         13     test_UpDownCounter.dut.pdut.property2 PASS
15  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=1 data_out=    6
         15     test_UpDownCounter.dut.pdut.property2 PASS
17  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=1 data_out=    7
         17     test_UpDownCounter.dut.pdut.property2 PASS
19  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=1 data_out=    8
         19     test_UpDownCounter.dut.pdut.property2 PASS
21  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=1 data_out=    9
         21     test_UpDownCounter.dut.pdut.property2 PASS
23  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=0 data_out=   10
         23     test_UpDownCounter.dut.pdut.property2 PASS
25  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=0 data_out=   10
         25     test_UpDownCounter.dut.pdut.property1 PASS
27  clk=1 rst_=1 ld_cnt=0 updn_cnt=1 count_enb=0 data_out=   10
         27     test_UpDownCounter.dut.pdut.property1 PASS
29  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=0 data_out=  127
31  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=0 data_out=  127
         31     test_UpDownCounter.dut.pdut.property1 PASS
33  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=0 data_out=  127
         33     test_UpDownCounter.dut.pdut.property1 PASS
```

Figure 12

Την στιγμή 1ns, το rst_ έχει πέσει στο 0, data_out = 0 => PASS ο έλεγχος No1 της εκφώνησης (το property που σχετίζεται με το reset -> No1).

Έπειτα την στιγμή 3ns τα rst_, ld_cnt , updn_cnt, count_enb είναι 1, και η έξοδος γίνεται από 0 που ήταν, 1 => την στιγμή 5ns PASS ο έλεγχος No3 (της εκφώνησης).

Ομοίως συνεχίζεται η ίδια διαδικασία μέχρι και την στιγμή 23ns

Τις χρονικές στιγμές 23ns και 25ns το count_enb είναι 0, η έξοδος παραμένει σταθερή => τις χρονικές στιγμές 25ns και 27ns αντίστοιχα PASS ο έλεγχος No2 (της εκφώνησης).

Την χρονική στιγμή 27ns το ld_cnt είναι 0, οπότε φορτώνεται στην έξοδο το data_in που είναι το 127 (θα τυπωθεί όμως στην κονσόλα στον επόμενο κύκλο).

Τις χρονικές στιγμές 29ns, 31ns, 33ns, 35ns, και 37ns το ld_cnt έχει γίνει 1 αλλά το count_enb είναι 0, η έξοδος παραμένει σταθερή => τις χρονικές στιγμές 31ns, 33ns, 35ns, 37ns και 39ns αντίστοιχα PASS ο έλεγχος No2.

```
            33     test_UpDownCounter.dut.pdut.property1 PASS
 35  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=0 data_out=  127
            35     test_UpDownCounter.dut.pdut.property1 PASS
 37  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=0 data_out=  127
            37     test_UpDownCounter.dut.pdut.property1 PASS
 39  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=0 data_out=  127
            39     test_UpDownCounter.dut.pdut.property1 PASS
 41  clk=1 rst_=0 ld_cnt=1 updn_cnt=1 count_enb=0 data_out=    0
            41     test_UpDownCounter.dut.pdut.reset PASS
 43  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=0 data_out=    0
 45  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=0 data_out=    0
            45     test_UpDownCounter.dut.pdut.property1 PASS
 47  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=0 data_out=    0
            47     test_UpDownCounter.dut.pdut.property1 PASS
 49  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=0 data_out=    0
            49     test_UpDownCounter.dut.pdut.property1 PASS
 51  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=1 data_out=    0
            51     test_UpDownCounter.dut.pdut.property1 PASS
 53  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=1 data_out=    1
            53     test_UpDownCounter.dut.pdut.property2 PASS
 55  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=1 data_out=    2
            55     test_UpDownCounter.dut.pdut.property2 PASS
 57  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=1 data_out=    3
            57     test_UpDownCounter.dut.pdut.property2 PASS
 59  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=1 data_out=    4
            59     test_UpDownCounter.dut.pdut.property2 PASS
 61  clk=1 rst_=1 ld_cnt=1 updn_cnt=1 count_enb=1 data_out=    5
            61     test_UpDownCounter.dut.pdut.property2 PASS
 63  clk=1 rst_=1 ld_cnt=1 updn_cnt=0 count_enb=1 data_out=    6
            63     test_UpDownCounter.dut.pdut.property2 PASS
 65  clk=1 rst_=1 ld_cnt=1 updn_cnt=0 count_enb=1 data_out=    5
            65     test_UpDownCounter.dut.pdut.property2 PASS
 67  clk=1 rst_=1 ld_cnt=1 updn_cnt=0 count_enb=1 data_out=    4
```

Figure 13

Την στιγμή 41ns, το rst_ έχει πέσει στο 0, data_out = 0 => PASS ο έλεγχος No1 στην εκφώνηση (το property που σχετίζεται με το reset).

Έπειτα τις χρονικές στιγμές 43ns, 45ns, 47ns, και 49ns τα rst_, ld_cnt και updn_cnt είναι 1 αλλά το count_enb είναι 0, η έξοδος παραμένει σταθερή => τις χρονικές στιγμές 45ns, 47ns, 49ns και 51ns αντίστοιχα PASS ο έλεγχος No2.

Την χρονική στιγμή 51ns τα rst_, ld_cnt , updn_cnt και count_enb είναι 1, η έξοδος αυξάνει => την χρονική στιγμή 53ns PASS ο έλεγχος No3.

Ομοίως συνεχίζεται η ίδια διαδικασία μέχρι και την στιγμή 63ns.

```
65  clk=1 rst_=1 ld_cnt=1 updn_cnt=0 count_enb=1 data_out=    5
          65     test_UpDownCounter.dut.pdut.property2 PASS
67  clk=1 rst_=1 ld_cnt=1 updn_cnt=0 count_enb=1 data_out=    4
          67     test_UpDownCounter.dut.pdut.property2 PASS
69  clk=1 rst_=1 ld_cnt=1 updn_cnt=0 count_enb=1 data_out=    3
          69     test_UpDownCounter.dut.pdut.property2 PASS
71  clk=1 rst_=1 ld_cnt=1 updn_cnt=0 count_enb=1 data_out=    2
          71     test_UpDownCounter.dut.pdut.property2 PASS
73  clk=1 rst_=1 ld_cnt=1 updn_cnt=0 count_enb=1 data_out=    1
          73     test_UpDownCounter.dut.pdut.property2 PASS
75  clk=1 rst_=1 ld_cnt=1 updn_cnt=0 count_enb=1 data_out=    0
          75     test_UpDownCounter.dut.pdut.property2 PASS
77  clk=1 rst_=1 ld_cnt=1 updn_cnt=0 count_enb=1 data_out=65535
          77     test_UpDownCounter.dut.pdut.property2 PASS
79  clk=1 rst_=1 ld_cnt=1 updn_cnt=0 count_enb=1 data_out=65534
          79     test_UpDownCounter.dut.pdut.property2 PASS
81  clk=1 rst_=1 ld_cnt=1 updn_cnt=0 count_enb=1 data_out=65533
          81     test_UpDownCounter.dut.pdut.property2 PASS
83  clk=1 rst_=1 ld_cnt=1 updn_cnt=0 count_enb=1 data_out=65532
          83     test_UpDownCounter.dut.pdut.property2 PASS
85  clk=1 rst_=1 ld_cnt=0 updn_cnt=0 count_enb=1 data_out=65531
          85     test_UpDownCounter.dut.pdut.property2 PASS
87  clk=1 rst_=1 ld_cnt=1 updn_cnt=0 count_enb=1 data_out=  127
89  clk=1 rst_=1 ld_cnt=1 updn_cnt=0 count_enb=1 data_out=  126
          89     test_UpDownCounter.dut.pdut.property2 PASS
91  clk=1 rst_=1 ld_cnt=1 updn_cnt=0 count_enb=0 data_out=  125
          91     test_UpDownCounter.dut.pdut.property2 PASS
93  clk=1 rst_=1 ld_cnt=1 updn_cnt=0 count_enb=0 data_out=  125
          93     test_UpDownCounter.dut.pdut.property1 PASS
95  clk=1 rst_=0 ld_cnt=1 updn_cnt=0 count_enb=0 data_out=    0
          95     test_UpDownCounter.dut.pdut.reset PASS
```

Figure 14

Συνεχίζεται η ίδια διαδικασία μέχρι την στιγμή 85ns. (Την χρονική στιγμή 75ns η έξοδος έχει φτάσει στην ελάχιστη δυνατή τιμή της, την μηδενική, επομένως την στιγμή 77ns η τιμή της εξόδου που «μειώθηκε» είναι η μέγιστη δυνατή δηλ. 65535).

Την χρονική στιγμή 85ns το ld_cnt είναι 0, οπότε φορτώνεται στην έξοδο τα data_in που είναι το 127 (θα τυπωθεί όμως στην κονσόλα στον επόμενο κύκλο).

Τις χρονικές στιγμές 87ns και 89ns, και τα rst_, ld_cnt και count_enb είναι 1 αλλά το updn_cnt είναι 0, η έξοδος μειώνεται => την χρονική στιγμή 89ns και 91ns αντίστοιχα PASS ο έλεγχος No3 (η έξοδος έχει μειωθεί κατά 1).

Την χρονική στιγμή 91ns τα rst_ και ld_cnt είναι 1 αλλά τα updn_cnt και count_enb είναι 0, η έξοδος παραμένει σταθερή (125) => την χρονική στιγμή 93ns PASS ο έλεγχος No2.

Την στιγμή 95ns, το rst_ έχει πέσει στο 0, data_out = 0 => PASS ο έλεγχος No1 στην εκφώνηση (το property που σχετίζεται με το reset No1).

2 Απλή Σύγχρονη FIFO (First-in First-Out) Μνήμη

Οι FIFO είναι ίσως ο πιο κοινός μηχανισμός αξιόπιστης μεταφοράς δεδομένων μεταξύ ασύγχρονων περιοχών ρολογιού.

- Η αρχή λειτουργίας είναι ότι η λέξη που γράφτηκε πρώτη (χρονικά πιο νωρίς), θα διαβαστεί επίσης πρώτη (χρονικά πιο σύντομα)
- Δείκτες (pointers) ελέγχουν τι γράφεται και τι διαβάζεται από τη FIFO
- Το πλάτος της FIFO εξαρτάται από το πλάτος των δεδομένων που μεταφέρονται μεταξύ των περιοχών
- Το βάθος της FIFO εξαρτάται από τη σχέση μεταξύ των ρολογιών των δύο περιοχών και το κόστος (σε επιφάνεια)
- Υπάρχουν **σύγχρονες** αλλά και ασύγχρονες FIFO
- Τύποι FIFO, δύο θυρών (dual port) και δύο ρολογιών (dual clock)

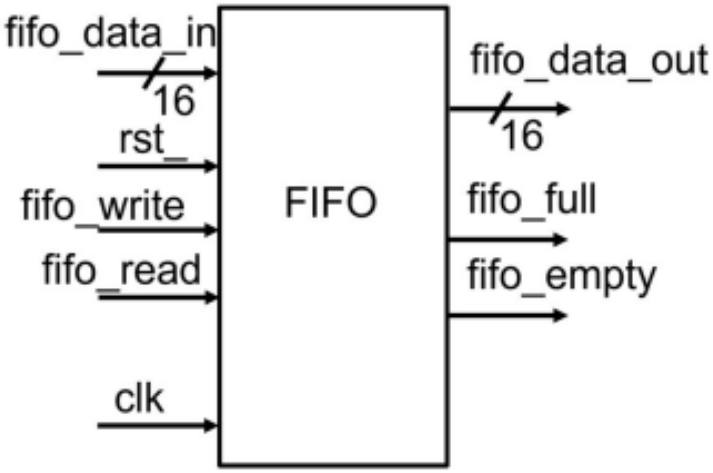


Figure 15 Top level diagram of the synchronous FIFO

2.1 Περιγραφή εισόδων/εξόδων

Η Σύγχρονη FIFO έχει τις εξής είσοδους:

- Τα δεδομένα είσοδου *16 bit (fifo_data_in)
- Ένα σήμα ρολογίου (clk)
- Ένα σήμα «αιτήματος» εγγραφής (fifo_write)
- Ένα σήμα «αιτήματος» ανάγνωσης (fifo_read)
- Ένα ασύγχρονο σήμα αρχικοποίησης (rst_)

Και τις εξής εξόδους:

- ❖ Τα δεδομένα εξόδου *16 bit (fifo_data_out)
- ❖ Ένα σήμα πλήρους fifo (fifo_full)
- ❖ Ένα σήμα κένη fifo (fifo_empty)

*Το πλάτος και το βάθος της fifo παραμετροποιείται, στην εφαρμογή μας το πλάτος και το βάθος είναι 16.

2.2 Περιγραφή της λειτουργίας του κυκλώματος

- ➢ Η σύγχρονη FIFO λειτουργεί στην θετική ακμή του ρολογίου, και έχει ένα ασύγχρονο reset(active low). Η μνήμη πρακτικά ελέγχεται από έναν δείκτη εγγραφής και έναν δείκτη ανάγνωσης (wr_ptr και rd_ptr αντίστοιχα πλάτους *“size_bits” στην προκειμένη περίπτωση 4 bit).
- ➢ Χρησιμοποιήθηκε ένας unpacked πίνακας fifo_memory (μνήμη FIFO), με παραμετροποιημένο το πλάτος και το βάθος.
- ➢ Ο δείκτης εγγραφής αυξάνεται κατά 1 κάθε φορά που υπάρχει αίτημα εγγραφής και η FIFO δεν είναι γεμάτη, μηδενίζεται κατά την αρχικοποίηση, σε κάθε άλλη περίπτωση παραμένει σταθερός.
- ➢ Ο δείκτης ανάγνωσης αυξάνεται κατά 1 κάθε φορά που υπάρχει αίτημα ανάγνωσης και η FIFO δεν είναι κενή, μηδενίζεται κατά την αρχικοποίηση, σε κάθε άλλη περίπτωση παραμένει σταθερός.
- ➢ Η μνήμη περιέχει έναν μετρητή (cnt πλάτους *“size_bits+1” στην προκειμένη περίπτωση 5 bit) που αυξάνεται κατά 1 σε μια εγγραφή, μειώνεται κατά 1 σε μια ανάγνωση, μηδενίζεται κατά την αρχικοποίηση, παραμένει σταθερός σε κάθε άλλη περίπτωση (ταυτόχρονη εγγραφή και ανάγνωση κ.λ.π.). Αυτός ο μετρητής χρησιμοποιείται για να γίνεται το fifo_full 1 όταν η fifo είναι γεμάτη και για να γίνεται το fifo_empty 1 όταν η fifo είναι άδεια, σε κάθε άλλη περίπτωση τα fifo_empty και fifo_full είναι στο λογικό 0.
- ➢ Κατά την αρχικοποίηση τα δεδομένα εξόδου “fifo_data_out” μηδενίζονται. Κάθε φορά που υπάρχει αίτημα ανάγνωσης και η FIFO δεν είναι κενή, ανατίθενται τα δεδομένα που βρίσκονται στην θέση μνήμης που ορίζει ο rd_ptr στα δεδομένα εξόδου, αλλιώς τα δεδομένα εξόδου “fifo_data_out” παραμένουν σταθερά. Κάθε φορά που υπάρχει αίτημα εγγραφής και η FIFO δεν είναι γεμάτη, γίνεται

εγγραφή των δεδομένων εισόδου “fifo_data_in” στην θέση μνήμης που ορίζει ο wr_ptr.

*ο αριθμός των bits του μετρητή, καθώς και ο αριθμός των bits wr_ptr και rd_ptr παραμετροποιείται, γιατί εξαρτώνται από το βάθος της FIFO το οποίο είναι παραμετροποιημένο.

2.3 System Verilog Code

Χρησιμοποιώντας την γλώσσα περιγραφής υλικού System Verilog, ο κώδικας που αντιστοιχεί στην παραπάνω περιγραφή τις σύγχρονης FIFO φαίνεται στην εικ. 16. Οι είσοδοι/έξοδοι του κυκλώματος είναι logic.

```
module synchronous_FIFO
#(parameter width = 16, size_bits = 4, depth = 16)
(input logic clk, rst_, fifo_write, fifo_read, input logic [width-1:0] fifo_data_in, output logic [width-1:0] fifo_data_out, output logic fifo_full, fifo_empty);

logic[size_bits-1:0] wr_ptr;
logic[size_bits-1:0] rd_ptr;
logic [size_bits:0] cnt;

logic [width-1:0]fifo_memory[depth-1:0];

always_comb
begin
        if(cnt == 0)
                fifo_empty = 1;
        else
                fifo_empty = 0;

        if(cnt == depth)
                fifo_full = 1;
        else
                fifo_full = 0;
end
always_ff @(posedge clk, negedge rst_)
begin
        if(!rst_)
                cnt <= 0;
        else if((fifo_write && !fifo_full) && (fifo_read && !fifo_empty))
                cnt <= cnt;
        else if(fifo_write && !fifo_full)
                cnt <= cnt + 1;
        else if(fifo_read && !fifo_empty)
                cnt <= cnt - 1;
        else
                cnt <= cnt;
end
```

```
always_ff @(posedge clk, negedge rst_)
begin
        if(!rst_)
                fifo_data_out <= 0;
        else if(fifo_read && !fifo_empty)
                fifo_data_out <= fifo_memory[rd_ptr];
        else
                fifo_data_out <= fifo_data_out;
end

always_ff @(posedge clk, negedge rst_)
begin
        if(fifo_write && !fifo_full)
                fifo_memory[wr_ptr] <=  fifo_data_in;
        else
                fifo_memory[wr_ptr] <= fifo_memory[wr_ptr];
end

always_ff @(posedge clk, negedge rst_)
begin
        if(!rst_)
        begin
                wr_ptr <= 0;
                rd_ptr <= 0;
        end
        else
        begin
                if(fifo_write && !fifo_full)
                        wr_ptr <= wr_ptr + 1;
                else
                        wr_ptr <= wr_ptr;

                if(fifo_read && !fifo_empty)
                        rd_ptr <= rd_ptr + 1;
                else
                        rd_ptr <= rd_ptr;
        end
end
endmodule
```

Figure 16: System Verilog Code synchronous FIFO

2.4 Testbench

Για την επαλήθευση της σωστής λειτουργίας του κυκλώματος δημιουργήθηκε το αρχείο δοκιμής, το οποίο φαίνεται στην εικ. 17.

```
module test_synchronous_FIFO
#(parameter width = 16, size_bits = 4, depth = 16)();
bit rst_ = 1;
bit clk, fifo_write, fifo_read;

bit [width-1:0] fifo_data_in;
wire [width-1:0] fifo_data_out;
wire fifo_full, fifo_empty;


synchronous_FIFO #(.width(width), .depth(depth), .size_bits(size_bits)) dut(.rst_(rst_), .clk(clk), .fifo_write(fifo_write), .fifo_read(fifo_read),
.fifo_data_in(fifo_data_in), .fifo_data_out(fifo_data_out), .fifo_full(fifo_full), .fifo_empty(fifo_empty));
bind dut SVA_synchronous_FIFO #(.width(width), .depth(depth), .size_bits(size_bits)) pdut(.clk(clk), .rst_(rst_), .fifo_write(fifo_write),
.fifo_read(fifo_read), .fifo_full(fifo_full), .fifo_empty(fifo_empty), .fifo_data_out(fifo_data_out), .cnt(dut.cnt), .wr_ptr(dut.wr_ptr), .rd_ptr(dut.rd_ptr));

always #1 clk = ~clk;

always @(posedge clk)
$display($stime,,,"clk=%b rst_=%b fifo_write=%b fifo_read=%b fifo_data_in=%d fifo_data_out=%d fifo_full=%b fifo_empty=%b cnt=%d wr_ptr=%d rd_ptr=%d",clk, rst_,
fifo_write, fifo_read, fifo_data_in, fifo_data_out, fifo_full, fifo_empty, dut.cnt, dut.wr_ptr, dut.rd_ptr);
```

```
initial
begin
fifo_write = 0;
fifo_read = 0;
fifo_data_in = 16'b0000011;
rst_ = 0;
#2 rst_ = 1;
fifo_write = 1;
#2 fifo_data_in = 16'b0000010;
#2 fifo_data_in = 16'b0000011;
#2 fifo_data_in = 16'b0000100;
#2 fifo_data_in = 16'b0000101;
#2 fifo_data_in = 16'b0000110;
#2 fifo_write = 0;
fifo_read = 1;
#4 fifo_write = 1;
fifo_read = 0;
#2 fifo_data_in = 16'b0000100;
#2 fifo_data_in = 16'b0000101;
#2 fifo_data_in = 16'b0000110;
#2 fifo_data_in = 16'b0000111;
#2 fifo_data_in = 16'b0001000;
#2 fifo_data_in = 16'b0001001;
#2 fifo_data_in = 16'b0001010;
#2 fifo_data_in = 16'b0001011;
#2 fifo_write = 0;
fifo_read = 1;
#28 fifo_write = 1;
fifo_read = 0;
#2 fifo_data_in = 16'b0100010;
#2 fifo_data_in = 16'b0100011;
#2 fifo_data_in = 16'b0100100;
#2 fifo_data_in = 16'b0100101;
#2 fifo_data_in = 16'b0100110;
#2 fifo_data_in = 16'b0100111;
#2 fifo_data_in = 16'b0101000;
#2 fifo_data_in = 16'b0101001;
#2 fifo_data_in = 16'b0101010;
#2 fifo_data_in = 16'b0111011;
#2 fifo_data_in = 16'b0110010;
#2 fifo_data_in = 16'b0110011;
#2 fifo_data_in = 16'b0110100;
#2 fifo_data_in = 16'b0110101;
#2 fifo_data_in = 16'b0110110;
```

```
#2 fifo_data_in = 16'b0110111;
#2 fifo_data_in = 16'b0111000;
#2 fifo_data_in = 16'b0111001;
#2 fifo_data_in = 16'b0111010;
#2 fifo_data_in = 16'b0111011;
#2 fifo_write = 0;
fifo_read = 1;
#36 fifo_write = 1;
fifo_read = 0;
#2 fifo_data_in = 16'b0111010;
#2 fifo_data_in = 16'b0111011;
#2 fifo_data_in = 16'b0110111;
#2 fifo_data_in = 16'b0111000;
#2 fifo_write = 0;
fifo_read = 0;
#2 fifo_data_in = 16'b0111001;
#2 fifo_data_in = 16'b0111010;
fifo_read = 1;
fifo_write = 1;
#2 fifo_data_in = 16'b0111011;
#8 fifo_write = 0;
#14 fifo_read = 0;
fifo_write = 1;
fifo_data_in = 16'b1111011;
#2 fifo_write = 0;
fifo_read = 1;
#2 rst_ = 0;
#2 $finish;
end

endmodule
```

Figure 17: Testbench

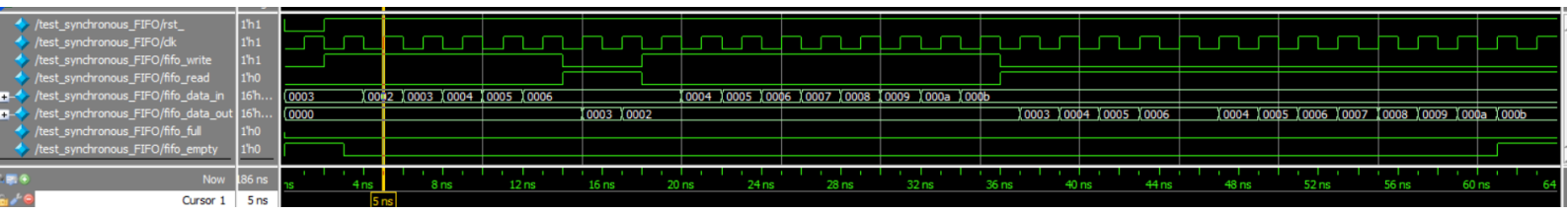


Figure 18

- Εικ. 18: Αρχικά την στιγμή 0ns το rst_ κατέρχεται στο λογικό 0, τα δεδομένα εξόδου αρχικοποιούνται στο 0, το fifo_empty είναι στο

λογικό 1 (αφού η FIFO είναι κενή), τα fifo_read και fifo_write είναι στο λογικό 0, και το fifo_full είναι στο λογικό 0.

- Την χρονική στιγμή 2ns το rst_ και το fifo_write γίνονται 1, επόμενως κατά την ανερχόμενη ακμή του ρολογίου (3ns) εγγράφονται τα δεδομένα εισόδου (0003 16δικό) στην θέση 0 της FIFO και αφού πλέον υπάρχουν δεδομένα στην FIFO το fifo_empty έγινε 0. Ομοίως τις επόμενες ανερχόμενες ακμές του ρολογιού 5ns, 7ns, 9ns, 11ns και 13ns εγγράφονται τα δεδομένα εισόδου 0002, 0003, 0004, 0005, 0006 αντίστοιχα στις αντίστοιχες θέσεις (1, 2, 3, 4, 5) της FIFO, η FIFO δεν είναι κενή ούτε πλήρης για αυτό το fifo_empty, και fifo_full είναι στο 0.

- Στο χρονικό διάστημα 14ns-18ns το rst_ , fifo_read είναι 1, το fifo_write είναι 0, στην ανερχόμενη ακμή του ρολογίου (15ns) διαβάζονται στην έξοδο τα δεδομένα της θέσης 0 της FIFO δηλ το 0003, στην επόμενη θετική ακμή του ρολογιού(17ns), διαβάζονται τα δεδομένα της θέσης 1 της FIFO δηλ. το 0002, μέσα σε αυτό το χρονικό διάστημα η FIFO δεν είναι κενή ούτε πλήρης για αυτό το fifo_empty, και fifo_full είναι στο 0.

- Στο χρονικό διάστημα 18ns-36ns το rst_ , fifo_write είναι 1, το fifo_read είναι 0, στις ανερχόμενες ακμές του ρολογιού 19ns, 21ns, 23ns, 25ns, 27ns, 29ns, 31ns, 33ns και 35ns εγγράφονται τα δεδομένα εισόδου 0006, 0004, 0005, 0006, 0007, 0008, 0009, 000a και 000b αντίστοιχα στις αντίστοιχες θέσεις (6, 7, 8, 9, 10, 11, 12, 13, 14) της FIFO, μέσα σε αυτό το χρονικό διάστημα η FIFO δεν είναι κενή ούτε πλήρης για αυτό το fifo_empty, και fifo_full είναι στο 0.

- Στο χρονικό διάστημα 36ns-64ns το rst_ , fifo_read είναι 1, το fifo_write είναι 0, στις ανερχόμενες ακμές του ρολογίου 37ns, 39ns, 41ns, 43ns, 45ns, 47ns, 49ns, 51ns, 53ns, 55ns, 57ns, 59ns και 61ns διαβάζονται στην έξοδο τα δεδομένα των θέσεων 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 αντίστοιχα της FIFO (δηλ το 0003, 0004, 0005, 0006, 0006, 0004, 0005, 0006, 0007, 0008, 0009, 000a και 000b αντίστοιχα) , μέσα σε αυτό το χρονικό διάστημα η FIFO δεν είναι πλήρης για αυτό το fifo_full είναι στο 0, αλλά την χρονική στιγμή 61ns η FIFO έχει αδειάσει για αυτό το fifo_empty γίνεται 1. Την στιγμή 63ns η FIFO είναι άδεια οπότε δεν μπορεί να γίνει ανάγνωση δεδομένων.

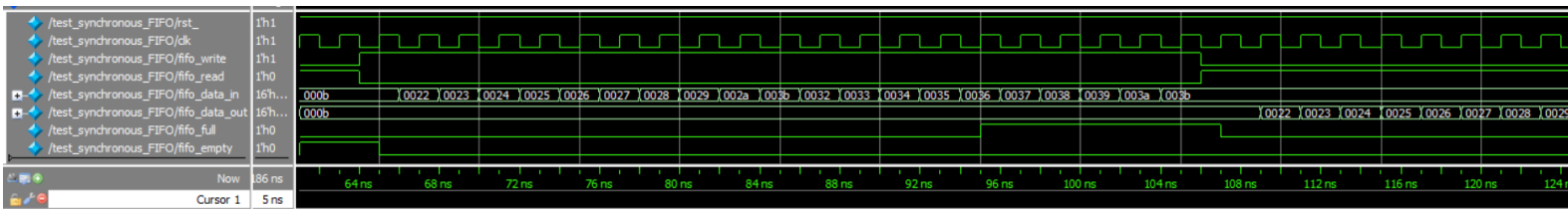


Figure 19

- Εικ. 19: Στο χρονικό διάστημα 64ns-106ns το rst_ , fifo_write είναι 1, το fifo_read είναι 0, στις ανερχόμενες ακμές του ρολογιού 65ns, 67ns, 69ns, 71ns, 73ns, 75ns, 77ns, 79ns, 81ns, 83ns, 85ns, 87ns, 89ns, 91ns, 93ns και 95ns εγγράφονται τα δεδομένα εισόδου 000b, 0022, 0023, 0024, 0025, 0026, 0027, 0028, 0029, 002a, 003b, 0032, 0033, 0034, 0035 και 0036 αντίστοιχα στις αντίστοιχες θέσεις (15,0,1,2, … 13, 14) της FIFO. Την χρονική στιγμή 95ns στην FIFO έχουν εγγραφεί 16 δεδομένα οπότε είναι πλήρης, το fifo_full γίνεται 1, και δεν μπορούν να εγγραφούν στις επόμενες ανερχόμενες ακμές του ρολογιού(97ns, 99ns, 101ns, 103ns και 105ns) νέα δεδομένα. Την στιγμή 64ns η FIFO είναι κενή, fifo_empty -> 1, ενώ την στιγμή 65ns το fifo_empty γίνεται 0 γιατί γράφηκε το 000b.

- Στο χρονικό διάστημα 106ns-124ns το rst_ , fifo_read είναι 1, το fifo_write είναι 0, στις ανερχόμενες ακμές του ρολογίου 107ns, 109ns, 111ns, 113ns, 115ns, 117ns, 119ns, 121ns και 123ns διαβάζονται στην έξοδο τα δεδομένα των θέσεων (15,0, 1, 2, 3, 4, 5, 6, 7) αντίστοιχα της FIFO (δηλ το 000b, 0022, 0023, 0024, 0025, 0026, 0027, 0028 και 0029 αντίστοιχα), μέσα σε αυτό το χρονικό διάστημα η FIFO είναι αρχικά πλήρης για αυτό το fifo_full είναι στο 1, αλλά την χρονική στιγμή 107ns έχει αδειάσει μια θέση της FIFO για αυτό το fifo_full γίνεται 0, ενώ το fifo_empty είναι 0 μιας που η fifo δεν είναι κενή.

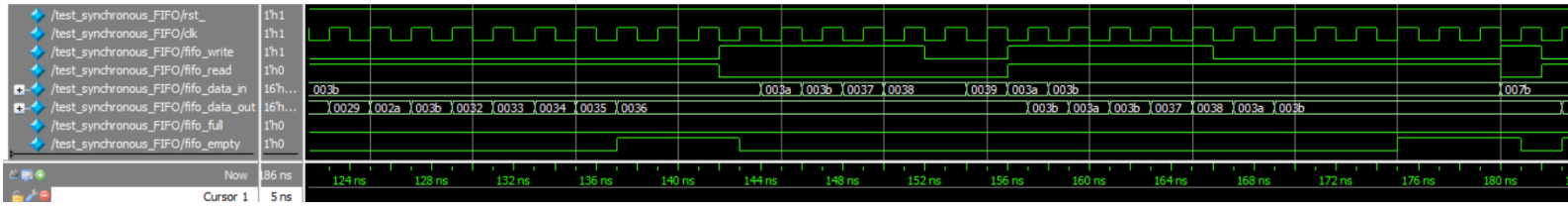


Figure 20

- Εικ. 20: Στο χρονικό διάστημα 124ns-142ns το rst_ , fifo_read είναι 1, το fifo_write είναι 0, στις ανερχόμενες ακμές του ρολογίου 125ns, 127ns, 129ns, 131ns, 133ns, 135ns και 137ns διαβάζονται στην έξοδο τα δεδομένα των θέσεων 8,9, 10, 11, 12, 13 και 14 αντίστοιχα της FIFO (δηλ το 002a, 003b, 0032, 0033, 0034, 0035 και 0036 αντίστοιχα), μέσα σε αυτό το χρονικό διάστημα η FIFO δεν είναι ποτέ πλήρης για αυτό το fifo_full είναι στο 0. Το fifo_empty είναι αρχικά 0 μιας που η fifo δεν είναι κενή, αλλά την χρονική στιγμή 137ns η FIFO έχει αδειάσει για αυτό το

fifo_empty γίνεται 1. Τις χρονικές στιγμές 139ns και 141ns (θετικές ακμές του ρολογιού) δεν γίνεται ανάγνωση των δεδομέων γιατί η FIFO είναι κενή.

- Στο χρονικό διάστημα 142ns-152ns το rst_ , fifo_write είναι 1, το fifo_read είναι 0, στις ανερχόμενες ακμές του ρολογιού 143ns, 145ns, 147ns, 149ns και 151ns εγγράφονται τα δεδομένα εισόδου 003b, 003a, 003b, 0037 και 0038 αντίστοιχα στις αντίστοιχες θέσεις (15,0,1,2,3) της FIFO. Την χρονική στιγμή 142ns η FIFO είναι κενή για αυτό το fifo_empty είναι 1, ενώ την χρονική στιγμή 143ns εγγράφεται στην FIFO το 003b, παύει η FIFO να είναι κενή για αυτό το fifo_empty γίνεται 0. Μέσα σε αυτό το χρονικό διάστημα η FIFO δεν είναι ποτέ πλήρης, για αυτό το fifo_full είναι παραμένει 0.
- Στο χρονικό διάστημα 152ns-156ns το rst_ είναι 1, αλλά το fifo_read και το fifo_write είναι 0, άρα σε αυτό το χρονικό διάστημα δεν γίνεται ούτε κάποια εγγραφή ούτε κάποια ανάγνωση δεδομένων. Η FIFO σε αυτό το χρονικό διάστημα δεν είναι ούτε πλήρης ούτε άδεια για αυτό το fifo_full και το fifo_empty είναι 0.
- Στο χρονικό διάστημα 156ns-166ns το rst_, το fifo_read και το fifo_write, είναι 1, άρα σε αυτό το χρονικό διάστημα γίνονται ταυτόχρονα εγγραφή και ανάγνωση δεδομένων. Στις ανερχομένες ανερχόμενες ακμές του ρολογιού 157ns, 159ns, 161ns, 163ns και 165ns εγγράφονται τα δεδομένα εισόδου 003a, 003b, 003b, 003b και 003b αντίστοιχα στις αντίστοιχες θέσεις 4,5,6,7,8 της FIFO, ενώ παράλληλα τις αντίστοιχες χρονικές στιγμές γίνεται ανάγνωση των δεδομένων των θέσεων 15, 0, 1, 2, 3 αντίστοιχα (003b, 003a, 003b, 0037, 0038). Η FIFO σε αυτό το χρονικό διάστημα δεν είναι ούτε πλήρης ούτε άδεια για αυτό το fifo_full και το fifo_empty είναι 0.
- Στο χρονικό διάστημα 166ns-180ns το rst_ , fifo_read είναι 1, το fifo_write είναι 0, στις ανερχόμενες ακμές του ρολογίου 167ns, 169ns, 171ns, 173ns και 175ns διαβάζονται στην έξοδο τα δεδομένα των θέσεων 4, 5, 6, 7, 8 αντίστοιχα της FIFO (δηλ το 003a, 003b, 003b, 003b και 003b αντίστοιχα), μέσα σε αυτό το χρονικό διάστημα η FIFO δεν είναι ποτέ πλήρης για αυτό το fifo_full είναι στο 0. Το fifo_empty είναι αρχικά 0 μιας που η fifo δεν είναι κενή, αλλά την χρονική στιγμή 175ns η FIFO έχει αδειάσει για αυτό το fifo_empty γίνεται 1. Τις χρονικές στιγμές 177ns και 179ns (θετικές ακμές του ρολογιού) δεν γίνεται ανάγνωση των δεδομέων γιατί η FIFO είναι κενή.

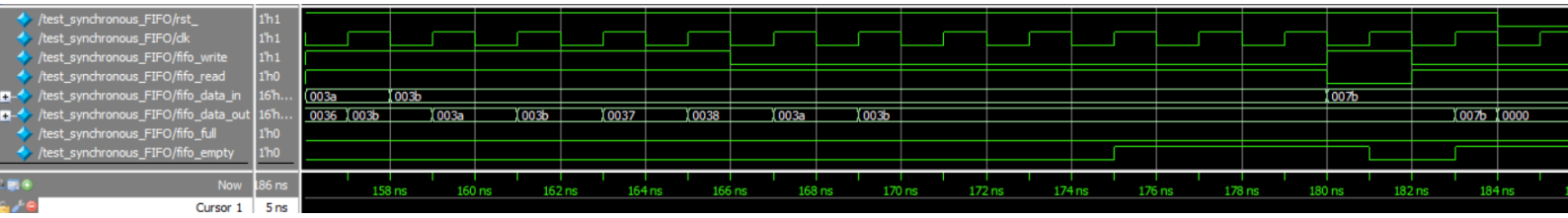


Figure 21

- Εικ.21: Στο χρονικό διάστημα 180ns-182ns το rst_ και το fifo_write είναι 1, το fifo_read είναι 0, στην ανερχόμενη ακμη του ρολογιού (181ns) εγγράφονται τα δεδομένα εισόδου 007b στην θέση 9 της FIFO. Την χρονική στιγμή 180ns η FIFO είναι κενή για αυτό το fifo_empty είναι 1, ενώ την χρονική στιγμή 181ns εγγράφεται στην FIFO το 007b, παύει η FIFO να είναι κενή για αυτό το fifo_empty γίνεται 0. Μέσα σε αυτό το χρονικό διάστημα η FIFO δεν είναι ποτέ πλήρης, για αυτό το fifo_full είναι παραμένει 0.

- Στο χρονικό διάστημα 182ns-184ns το rst_ , fifo_read είναι 1, το fifo_write είναι 0, στην ανερχόμενη ακμη του ρολογιού (183ns) διαβάζεται στην έξοδο το δεδομένο της θέσης 9 της FIFO (δηλ το 007b), μέσα σε αυτό το χρονικό διάστημα η FIFO δεν είναι ποτέ πλήρης για αυτό το fifo_full είναι στο 0. Το fifo_empty είναι αρχικά 0 μιας που η fifo δεν είναι κενή, αλλά την χρονική στιγμή 183ns η FIFO έχει αδειάσει για αυτό το fifo_empty γίνεται 1.

- Την χρονική στιγμή 184ns το rst_ γίνεται 0, η έξοδος της FIFO γίνεται 0, και το το fifo_empty συνεχίζει να είναι 1, αφού η FIFO κατά την αρχικοποίηση αδειάζει.

2.5 SVA

Χρησιμοποιώντας SVA, οι properties παράγουν τα κατάλληλα μηνύματα (PASS/FAIL) στο κατάλληλο παράθυρο του προσομοιωτή, προκειμένου να γίνεται γνωστό αν ικανοποιούνται ή όχι οι εξής έλεγχοι:

- Όταν το reset είναι ενεργό, τότε οι read και write pointers πρέπει να είναι 0, οι indicators empty και full πρέπει να είναι 1 και 0 αντίστοιχα (καθώς η μνήμη είναι άδεια), και ο μετρητής πρέπει να είναι 0.

- Το fifo_empty ενεργοποιείται (γίνεται 1) κάθε φορά που cnt = 0. Αυτήν η ιδιότητα πρέπει να απενεργοποιείται όταν το rst_ είναι στο 0.
- Το fifo_full ενεργοποιείται (γίνεται 1) κάθε φορά που cnt >= 16. Αυτήν η ιδιότητα πρέπει να απενεργοποιείται όταν το rst_ είναι στο 0.
- Εάν η FIFO είναι πλήρης και υπάρχει αίτημα εγγραφής (αλλά όχι ανάγνωσης), τότε ο δείκτης εγγραφής δεν πρέπει να αλλάξει.
- Εάν η FIFO είναι κενή και υπάρχει αίτημα ανάγνωσης (αλλά όχι εγγραφής), τότε ο δείκτης ανάγνωσης δεν πρέπει να αλλάξει.

```
module SVA_synchronous_FIFO
#(parameter width = 16, size_bits = 4, depth = 16)
(input clk, rst_, fifo_write, fifo_read, fifo_full, fifo_empty, input [width-1:0] fifo_data_out, input [size_bits:0] cnt, input [size_bits-1:0] wr_ptr, rd_ptr);
property rst1;
        @(negedge rst_) 1'b1 |=> @(posedge clk)  ((fifo_empty == 1) && (fifo_full == 0) && (cnt == 0) && (rd_ptr==0) && (wr_ptr==0));
        //On reset, the read and write pointers are zero, empty and full indicators are ?1? and ?0?,respectively (since the memory is indeed empty), and the cou
endproperty

property p2;
        @(posedge clk) disable iff(!rst_) cnt == 0 |-> fifo_empty; // The fifo empty signal is asserted whenever cnt = 0. Disable this property if the reset is
endproperty
property p3;
        @(posedge clk) disable iff(!rst_) cnt >= 16  |-> fifo_full; //The fifo full signal is asserted whenever cnt >= 16. Disable this property if the reset is
endproperty
property p4;
        @(posedge clk) /*disable iff(!rst_)*/ (fifo_full && fifo_write && !fifo_read) |=>  $stable(wr_ptr); //If the FIFO is full and there is write request (bu
endproperty
property p5;
        @(posedge clk) /*disable iff(!rst_)*/ (fifo_empty && !fifo_write && fifo_read) |=>  $stable(rd_ptr); //If the FIFO is empty and there is read request (b
endproperty

reset: assert property (rst1) else $display($time,,, "\t\t %m FAIL");
reset1: cover property (rst1) $display($time,,, "\t\t %m PASS");
property2: assert property (p2) else $display($time,,, "\t\t %m FAIL");
propert2: cover property (p2) $display($time,,, "\t\t %m PASS");
property3: assert property (p3) else $display($time,,, "\t\t %m FAIL");
propert3: cover property (p3) $display($time,,, "\t\t %m PASS");
property4: assert property (p4) else $display($time,,, "\t\t %m FAIL");
propert4: cover property (p4) $display($time,,, "\t\t %m PASS");
property5: assert property (p5) else $display($time,,, "\t\t %m FAIL");
propert5: cover property (p5) $display($time,,, "\t\t %m PASS");
endmodule
```

Figure 22

Στην Εικ. 22 φαίνεται ο κώδικας ο οποίος αναπτύχθηκε για αυτόν τον σκοπό.

$reset \rightarrow$ έλεγχος Νο 1 εκφωνήσης

$property2 \rightarrow$ έλεγχος Νο 2 εκφωνήσης

$property3 \rightarrow$ έλεγχος Νο 3 εκφωνήσης

$property4 \rightarrow$ έλεγχος Νο 4 εκφωνήσης

$property5 \rightarrow$ έλεγχος Νο 5 εκφωνήσης

Στις Εικ. 23, 24 και 25 φαίνονται τα αποτελέσματα που τυπώνονται στην κονσόλα του προσομοιωτή. Όλοι οι πρόσθετοι έλεγχοι είναι PASS, επαληθεύεται έτσι η σωστή συμπεριφορά της FIFO. Πιο συγκεκριμένα:

```
  1  clk=1 rst_=0 fifo_write=0 fifo_read=0 fifo_data_in=    3 fifo_data_out=    0 fifo_full=0 fifo_empty=1 cnt= 0 wr_ptr= 0 rd_ptr= 0
           1     test_synchronous_FIFO.dut.pdut.reset1 PASS
  3  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=    3 fifo_data_out=    0 fifo_full=0 fifo_empty=1 cnt= 0 wr_ptr= 0 rd_ptr= 0
           3     test_synchronous_FIFO.dut.pdut.propert2 PASS
  5  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=    2 fifo_data_out=    0 fifo_full=0 fifo_empty=0 cnt= 1 wr_ptr= 1 rd_ptr= 0
  7  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=    3 fifo_data_out=    0 fifo_full=0 fifo_empty=0 cnt= 2 wr_ptr= 2 rd_ptr= 0
  9  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=    4 fifo_data_out=    0 fifo_full=0 fifo_empty=0 cnt= 3 wr_ptr= 3 rd_ptr= 0
 11  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=    5 fifo_data_out=    0 fifo_full=0 fifo_empty=0 cnt= 4 wr_ptr= 4 rd_ptr= 0
 13  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=    6 fifo_data_out=    0 fifo_full=0 fifo_empty=0 cnt= 5 wr_ptr= 5 rd_ptr= 0
 15  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=    6 fifo_data_out=    0 fifo_full=0 fifo_empty=0 cnt= 6 wr_ptr= 6 rd_ptr= 0
 17  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=    6 fifo_data_out=    3 fifo_full=0 fifo_empty=0 cnt= 5 wr_ptr= 6 rd_ptr= 1
 19  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=    6 fifo_data_out=    2 fifo_full=0 fifo_empty=0 cnt= 4 wr_ptr= 6 rd_ptr= 2
 21  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=    4 fifo_data_out=    2 fifo_full=0 fifo_empty=0 cnt= 5 wr_ptr= 7 rd_ptr= 2
 23  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=    5 fifo_data_out=    2 fifo_full=0 fifo_empty=0 cnt= 6 wr_ptr= 8 rd_ptr= 2
 25  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=    6 fifo_data_out=    2 fifo_full=0 fifo_empty=0 cnt= 7 wr_ptr= 9 rd_ptr= 2
 27  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=    7 fifo_data_out=    2 fifo_full=0 fifo_empty=0 cnt= 8 wr_ptr=10 rd_ptr= 2
 29  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=    8 fifo_data_out=    2 fifo_full=0 fifo_empty=0 cnt= 9 wr_ptr=11 rd_ptr= 2
 31  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=    9 fifo_data_out=    2 fifo_full=0 fifo_empty=0 cnt=10 wr_ptr=12 rd_ptr= 2
 33  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   10 fifo_data_out=    2 fifo_full=0 fifo_empty=0 cnt=11 wr_ptr=13 rd_ptr= 2
 35  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   11 fifo_data_out=    2 fifo_full=0 fifo_empty=0 cnt=12 wr_ptr=14 rd_ptr= 2
 37  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   11 fifo_data_out=    2 fifo_full=0 fifo_empty=0 cnt=13 wr_ptr=15 rd_ptr= 2
 39  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   11 fifo_data_out=    3 fifo_full=0 fifo_empty=0 cnt=12 wr_ptr=15 rd_ptr= 3
 41  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   11 fifo_data_out=    4 fifo_full=0 fifo_empty=0 cnt=11 wr_ptr=15 rd_ptr= 4
 43  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   11 fifo_data_out=    5 fifo_full=0 fifo_empty=0 cnt=10 wr_ptr=15 rd_ptr= 5
 45  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   11 fifo_data_out=    6 fifo_full=0 fifo_empty=0 cnt= 9 wr_ptr=15 rd_ptr= 6
 47  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   11 fifo_data_out=    6 fifo_full=0 fifo_empty=0 cnt= 8 wr_ptr=15 rd_ptr= 7
 49  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   11 fifo_data_out=    4 fifo_full=0 fifo_empty=0 cnt= 7 wr_ptr=15 rd_ptr= 8
 51  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   11 fifo_data_out=    5 fifo_full=0 fifo_empty=0 cnt= 6 wr_ptr=15 rd_ptr= 9
 53  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   11 fifo_data_out=    6 fifo_full=0 fifo_empty=0 cnt= 5 wr_ptr=15 rd_ptr=10
 55  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   11 fifo_data_out=    7 fifo_full=0 fifo_empty=0 cnt= 4 wr_ptr=15 rd_ptr=11
 57  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   11 fifo_data_out=    8 fifo_full=0 fifo_empty=0 cnt= 3 wr_ptr=15 rd_ptr=12
 59  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   11 fifo_data_out=    9 fifo_full=0 fifo_empty=0 cnt= 2 wr_ptr=15 rd_ptr=13
 61  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   11 fifo_data_out=   10 fifo_full=0 fifo_empty=0 cnt= 1 wr_ptr=15 rd_ptr=14
 63  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   11 fifo_data_out=   11 fifo_full=0 fifo_empty=1 cnt= 0 wr_ptr=15 rd_ptr=15
           63    test_synchronous_FIFO.dut.pdut.propert2 PASS
 65  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   11 fifo_data_out=   11 fifo_full=0 fifo_empty=1 cnt= 0 wr_ptr=15 rd_ptr=15
           65    test_synchronous_FIFO.dut.pdut.propert2 PASS
           65    test_synchronous_FIFO.dut.pdut.propert5 PASS
 67  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   34 fifo_data_out=   11 fifo_full=0 fifo_empty=0 cnt= 1 wr_ptr= 0 rd_ptr=15
 69  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   35 fifo_data_out=   11 fifo_full=0 fifo_empty=0 cnt= 2 wr_ptr= 1 rd_ptr=15
 71  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   36 fifo_data_out=   11 fifo_full=0 fifo_empty=0 cnt= 3 wr_ptr= 2 rd_ptr=15
 73  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   37 fifo_data_out=   11 fifo_full=0 fifo_empty=0 cnt= 4 wr_ptr= 3 rd_ptr=15
 75  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   38 fifo_data_out=   11 fifo_full=0 fifo_empty=0 cnt= 5 wr_ptr= 4 rd_ptr=15
 77  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   39 fifo_data_out=   11 fifo_full=0 fifo_empty=0 cnt= 6 wr_ptr= 5 rd_ptr=15
 79  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   40 fifo_data_out=   11 fifo_full=0 fifo_empty=0 cnt= 7 wr_ptr= 6 rd_ptr=15
 81  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   41 fifo_data_out=   11 fifo_full=0 fifo_empty=0 cnt= 8 wr_ptr= 7 rd_ptr=15
 83  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   42 fifo_data_out=   11 fifo_full=0 fifo_empty=0 cnt= 9 wr_ptr= 8 rd_ptr=15
```

Figure 23

Την στιγμή 1ns, το rst_ ήδη έχει πέσει στο 0, γίνεται fifo_data_out 0 => PASS ο έλεγχος No1 της εκφώνησης (το property που σχετίζεται με το reset -> No1).

Έπειτα την στιγμή 3ns το rst_ και το fifo_write είναι 1, το fifo_read είναι 0, τα δεδομένα εξόδου, ο cnt, ο rd_ptr και ο wr_ptr είναι 0, το fifo_empty είναι 1 => PASS ο έλεγχος No2.

Στο χρονικό διάστημα 4-61ns το rst_ είναι ανενεργό (δηλ. είναι 1), γίνονται κάποιες εγγραφές και αναγνώσεις, αλλά η FIFO δεν είναι ποτέ κενή ή πλήρης (0<cnt<16), οπότε δεν θα γίνει η αποτίμηση των ελέγχων.

Την στιγμή 63ns το rst_ και το fifo_read είναι 1, το fifo_write είναι 0, τα δεδομένα εξόδου είναι 11, ο cnt είναι 0, ο rd_ptr και ο wr_ptr είναι 15, το fifo_empty είναι 1 => PASS ο έλεγχος No2, και την χρονική στιγμή 65ns PASS ο έλεχγος No5, (η FIFO είναι κενή και υπάχει αίτημα ανάγνωσης(αλλά όχι εγγραφής), τότε ο δείκτης ανάγνωσης παραμένει σταθερός, 15 στην περίπτωση αυτήν).

Την στιγμή 65ns το rst_ και το fifo_write είναι 1, το fifo_read είναι 0, τα δεδομένα εξόδου είναι 11, ο cnt είναι 0, ο rd_ptr και ο wr_ptr είναι 15, το fifo_empty είναι 1 => PASS ο έλεγχος No2.

Στο χρονικό διάστημα 66-83ns το rst_ είναι ανενεργό (δηλ. είναι 1), γίνονται κάποιες εγγραφές (αφού fifo_write = 1), αλλά η FIFO δεν είναι ποτέ κενή ή πλήρης (0<cnt<16), οπότε δεν θα γίνει η αποτίμηση των ελέγχων.

```
 85  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   59 fifo_data_out=   11 fifo_full=0 fifo_empty=0 cnt=10 wr_ptr= 9 rd_ptr=15
 87  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   50 fifo_data_out=   11 fifo_full=0 fifo_empty=0 cnt=11 wr_ptr=10 rd_ptr=15
 89  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   51 fifo_data_out=   11 fifo_full=0 fifo_empty=0 cnt=12 wr_ptr=11 rd_ptr=15
 91  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   52 fifo_data_out=   11 fifo_full=0 fifo_empty=0 cnt=13 wr_ptr=12 rd_ptr=15
 93  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   53 fifo_data_out=   11 fifo_full=0 fifo_empty=0 cnt=14 wr_ptr=13 rd_ptr=15
 95  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   54 fifo_data_out=   11 fifo_full=0 fifo_empty=0 cnt=15 wr_ptr=14 rd_ptr=15
 97  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   55 fifo_data_out=   11 fifo_full=1 fifo_empty=0 cnt=16 wr_ptr=15 rd_ptr=15
          97     test_synchronous_FIFO.dut.pdut.property3 PASS
 99  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   56 fifo_data_out=   11 fifo_full=1 fifo_empty=0 cnt=16 wr_ptr=15 rd_ptr=15
          99     test_synchronous_FIFO.dut.pdut.property3 PASS
          99     test_synchronous_FIFO.dut.pdut.property4 PASS
101  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   57 fifo_data_out=   11 fifo_full=1 fifo_empty=0 cnt=16 wr_ptr=15 rd_ptr=15
         101     test_synchronous_FIFO.dut.pdut.property3 PASS
         101     test_synchronous_FIFO.dut.pdut.property4 PASS
103  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   58 fifo_data_out=   11 fifo_full=1 fifo_empty=0 cnt=16 wr_ptr=15 rd_ptr=15
         103     test_synchronous_FIFO.dut.pdut.property3 PASS
         103     test_synchronous_FIFO.dut.pdut.property4 PASS
105  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   59 fifo_data_out=   11 fifo_full=1 fifo_empty=0 cnt=16 wr_ptr=15 rd_ptr=15
         105     test_synchronous_FIFO.dut.pdut.property3 PASS
         105     test_synchronous_FIFO.dut.pdut.property4 PASS
107  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   11 fifo_full=1 fifo_empty=0 cnt=16 wr_ptr=15 rd_ptr=15
         107     test_synchronous_FIFO.dut.pdut.property3 PASS
         107     test_synchronous_FIFO.dut.pdut.property4 PASS
109  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   11 fifo_full=0 fifo_empty=0 cnt=15 wr_ptr=15 rd_ptr= 0
111  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   34 fifo_full=0 fifo_empty=0 cnt=14 wr_ptr=15 rd_ptr= 1
113  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   35 fifo_full=0 fifo_empty=0 cnt=13 wr_ptr=15 rd_ptr= 2
115  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   36 fifo_full=0 fifo_empty=0 cnt=12 wr_ptr=15 rd_ptr= 3
117  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   37 fifo_full=0 fifo_empty=0 cnt=11 wr_ptr=15 rd_ptr= 4
119  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   38 fifo_full=0 fifo_empty=0 cnt=10 wr_ptr=15 rd_ptr= 5
121  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   39 fifo_full=0 fifo_empty=0 cnt= 9 wr_ptr=15 rd_ptr= 6
123  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   40 fifo_full=0 fifo_empty=0 cnt= 8 wr_ptr=15 rd_ptr= 7
125  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   41 fifo_full=0 fifo_empty=0 cnt= 7 wr_ptr=15 rd_ptr= 8
127  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   42 fifo_full=0 fifo_empty=0 cnt= 6 wr_ptr=15 rd_ptr= 9
129  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   59 fifo_full=0 fifo_empty=0 cnt= 5 wr_ptr=15 rd_ptr=10
131  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   50 fifo_full=0 fifo_empty=0 cnt= 4 wr_ptr=15 rd_ptr=11
133  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   51 fifo_full=0 fifo_empty=0 cnt= 3 wr_ptr=15 rd_ptr=12
135  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   52 fifo_full=0 fifo_empty=0 cnt= 2 wr_ptr=15 rd_ptr=13
137  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   53 fifo_full=0 fifo_empty=0 cnt= 1 wr_ptr=15 rd_ptr=14
139  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   54 fifo_full=0 fifo_empty=1 cnt= 0 wr_ptr=15 rd_ptr=15
         139     test_synchronous_FIFO.dut.pdut.property2 PASS
141  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   54 fifo_full=0 fifo_empty=1 cnt= 0 wr_ptr=15 rd_ptr=15
         141     test_synchronous_FIFO.dut.pdut.property2 PASS
         141     test_synchronous_FIFO.dut.pdut.property5 PASS
143  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   59 fifo_data_out=   54 fifo_full=0 fifo_empty=1 cnt= 0 wr_ptr=15 rd_ptr=15
         143     test_synchronous_FIFO.dut.pdut.property2 PASS
         143     test_synchronous_FIFO.dut.pdut.property5 PASS
145  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   58 fifo_data_out=   54 fifo_full=0 fifo_empty=0 cnt= 1 wr_ptr= 0 rd_ptr=15
```

Figure 24

Στο χρονικό διάστημα 83-95ns το rst_ είναι ανενεργό (δηλ. είναι 1), γίνονται κάποιες εγγραφές (αφού fifo_write = 1), αλλά η FIFO δεν είναι ποτέ κενή ή πλήρης (0<cnt<16), οπότε δεν θα γίνει η αποτίμηση των ελέγχων.

Την στιγμή 97ns το rst_ και το fifo_write είναι 1, το fifo_read είναι 0, τα δεδομένα εξόδου είναι 11, ο cnt είναι 16, ο rd_ptr και ο wr_ptr είναι 15, το fifo_full είναι 1 => PASS ο έλεγχος No3, και την χρονική στιγμή 99ns PASS ο έλεχγος No4, (η FIFO είναι πλήρης και υπάχει αίτημα εγγραφής (αλλά όχι ανάγνωσης), τότε ο δείκτης εγγραφής παραμένει σταθερός, 15 στην περίπτωση αυτήν).

Την στιγμή 99ns το rst_ και το fifo_write είναι 1, το fifo_read είναι 0, τα δεδομένα εξόδου είναι 11, ο cnt είναι 16, ο rd_ptr και ο wr_ptr είναι 15, το fifo_full είναι 1 => PASS ο έλεγχος No3, και την χρονική στιγμή 101ns PASS ο έλεχγος No4, (η FIFO είναι πλήρης και υπάχει αίτημα εγγραφής

(αλλά όχι ανάγνωσης), τότε ο δείκτης εγγραφής παραμένει σταθερός, 15 στην περίπτωση αυτήν).

Την στιγμή 101ns το rst_ και το fifo_write είναι 1, το fifo_read είναι 0, τα δεδομένα εξόδου είναι 11, ο cnt είναι 16, ο rd_ptr και ο wr_ptr είναι 15, το fifo_full είναι 1 => PASS ο έλεγχος No3, και την χρονική στιγμή 103ns PASS ο έλεχγος No4, (η FIFO είναι πλήρης και υπάχει αίτημα εγγραφής (αλλά όχι ανάγνωσης), τότε ο δείκτης εγγραφής παραμένει σταθερός, 15 στην περίπτωση αυτήν).

Την στιγμή 103ns το rst_ και το fifo_write είναι 1, το fifo_read είναι 0, τα δεδομένα εξόδου είναι 11, ο cnt είναι 16, ο rd_ptr και ο wr_ptr είναι 15, το fifo_full είναι 1 => PASS ο έλεγχος No3, και την χρονική στιγμή 105ns PASS ο έλεχγος No4, (η FIFO είναι πλήρης και υπάχει αίτημα εγγραφής (αλλά όχι ανάγνωσης), τότε ο δείκτης εγγραφής παραμένει σταθερός, 15 στην περίπτωση αυτήν).

Την στιγμή 105ns το rst_ και το fifo_write είναι 1, το fifo_read είναι 0, τα δεδομένα εξόδου είναι 11, ο cnt είναι 16, ο rd_ptr και ο wr_ptr είναι 15, το fifo_full είναι 1 => PASS ο έλεγχος No3, και την χρονική στιγμή 107ns PASS ο έλεχγος No4, (η FIFO είναι πλήρης και υπάχει αίτημα εγγραφής (αλλά όχι ανάγνωσης), τότε ο δείκτης εγγραφής παραμένει σταθερός, 15 στην περίπτωση αυτήν).

Την στιγμή 107ns το rst_ και το fifo_read είναι 1, fifo_write είναι 0, τα δεδομένα εξόδου είναι 11, ο cnt είναι 16, ο rd_ptr και ο wr_ptr είναι 15, το fifo_full είναι 1 => PASS ο έλεγχος No3.

Στο χρονικό διάστημα 108-137ns το rst_ είναι ανενεργό (δηλ. είναι 1), γίνονται κάποιες αναγνώσεις (αφού fifo_read = 1), αλλά η FIFO δεν είναι ποτέ κενή ή πλήρης (0<cnt<16), οπότε δεν θα γίνει η αποτίμηση των ελέγχων.

Την στιγμή 139ns το rst_ και το fifo_read είναι 1, το fifo_write είναι 0, τα δεδομένα εξόδου είναι 54, ο cnt είναι 0, ο rd_ptr και ο wr_ptr είναι 15, το fifo_empty είναι 1 => PASS ο έλεγχος No2, και την χρονική στιγμή 141ns PASS ο έλεχγος No5, (η FIFO είναι κενή και υπάχει αίτημα ανάγνωσης (αλλά όχι εγγραφής), τότε ο δείκτης ανάγνωσης παραμένει σταθερός, 15 στην περίπτωση αυτήν).

Την στιγμή 141ns το rst_ και το fifo_read είναι 1, το fifo_write είναι 0, τα δεδομένα εξόδου είναι 54, ο cnt είναι 0, ο rd_ptr και ο wr_ptr είναι 15, το fifo_empty είναι 1 => PASS ο έλεγχος No2, και την χρονική στιγμή 143ns PASS ο έλεχγος No5, (η FIFO είναι κενή και υπάχει αίτημα

ανάγνωσης (αλλά όχι εγγραφής), τότε ο δείκτης ανάγνωσης παραμένει σταθερός, 15 στην περίπτωση αυτήν).

Την στιγμή 143ns το rst_ και το fifo_write είναι 1, το fifo_read είναι 0, τα δεδομένα εξόδου είναι 54, ο cnt είναι 0, ο rd_ptr και ο wr_ptr είναι 15, το fifo_empty είναι 1 => PASS ο έλεγχος No2.

Στο χρονικό διάστημα 145-175ns το rst_ είναι ανενεργό (δηλ. είναι 1), γίνονται κάποιες εγγραφές και αναγνώσεις, αλλά η FIFO δεν είναι ποτέ κενή ή πλήρης (0<cnt<16), οπότε δεν θα γίνει η αποτίμηση των ελέγχων.

```
  147  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   59 fifo_data_out=   54 fifo_full=0 fifo_empty=0 cnt= 2 wr_ptr= 1 rd_ptr=15
  149  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   55 fifo_data_out=   54 fifo_full=0 fifo_empty=0 cnt= 3 wr_ptr= 2 rd_ptr=15
  151  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=   56 fifo_data_out=   54 fifo_full=0 fifo_empty=0 cnt= 4 wr_ptr= 3 rd_ptr=15
  153  clk=1 rst_=1 fifo_write=0 fifo_read=0 fifo_data_in=   56 fifo_data_out=   54 fifo_full=0 fifo_empty=0 cnt= 5 wr_ptr= 4 rd_ptr=15
  155  clk=1 rst_=1 fifo_write=0 fifo_read=0 fifo_data_in=   57 fifo_data_out=   54 fifo_full=0 fifo_empty=0 cnt= 5 wr_ptr= 4 rd_ptr=15
  157  clk=1 rst_=1 fifo_write=1 fifo_read=1 fifo_data_in=   58 fifo_data_out=   54 fifo_full=0 fifo_empty=0 cnt= 5 wr_ptr= 4 rd_ptr=15
  159  clk=1 rst_=1 fifo_write=1 fifo_read=1 fifo_data_in=   59 fifo_data_out=   59 fifo_full=0 fifo_empty=0 cnt= 5 wr_ptr= 5 rd_ptr= 0
  161  clk=1 rst_=1 fifo_write=1 fifo_read=1 fifo_data_in=   59 fifo_data_out=   58 fifo_full=0 fifo_empty=0 cnt= 5 wr_ptr= 6 rd_ptr= 1
  163  clk=1 rst_=1 fifo_write=1 fifo_read=1 fifo_data_in=   59 fifo_data_out=   59 fifo_full=0 fifo_empty=0 cnt= 5 wr_ptr= 7 rd_ptr= 2
  165  clk=1 rst_=1 fifo_write=1 fifo_read=1 fifo_data_in=   59 fifo_data_out=   55 fifo_full=0 fifo_empty=0 cnt= 5 wr_ptr= 8 rd_ptr= 3
  167  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   56 fifo_full=0 fifo_empty=0 cnt= 5 wr_ptr= 9 rd_ptr= 4
  169  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   58 fifo_full=0 fifo_empty=0 cnt= 4 wr_ptr= 9 rd_ptr= 5
  171  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   59 fifo_full=0 fifo_empty=0 cnt= 3 wr_ptr= 9 rd_ptr= 6
  173  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   59 fifo_full=0 fifo_empty=0 cnt= 2 wr_ptr= 9 rd_ptr= 7
  175  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   59 fifo_full=0 fifo_empty=0 cnt= 1 wr_ptr= 9 rd_ptr= 8
  177  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   59 fifo_full=0 fifo_empty=1 cnt= 0 wr_ptr= 9 rd_ptr= 9
           177     test_synchronous_FIFO.dut.pdut.property2 PASS
  179  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=   59 fifo_data_out=   59 fifo_full=0 fifo_empty=1 cnt= 0 wr_ptr= 9 rd_ptr= 9
           179     test_synchronous_FIFO.dut.pdut.property2 PASS
           179     test_synchronous_FIFO.dut.pdut.property5 PASS
  181  clk=1 rst_=1 fifo_write=1 fifo_read=0 fifo_data_in=  123 fifo_data_out=   59 fifo_full=0 fifo_empty=1 cnt= 0 wr_ptr= 9 rd_ptr= 9
           181     test_synchronous_FIFO.dut.pdut.property2 PASS
           181     test_synchronous_FIFO.dut.pdut.property5 PASS
  183  clk=1 rst_=1 fifo_write=0 fifo_read=1 fifo_data_in=  123 fifo_data_out=   59 fifo_full=0 fifo_empty=0 cnt= 1 wr_ptr=10 rd_ptr= 9
  185  clk=1 rst_=0 fifo_write=0 fifo_read=1 fifo_data_in=  123 fifo_data_out=    0 fifo_full=0 fifo_empty=1 cnt= 0 wr_ptr= 0 rd_ptr= 0
           185     test_synchronous_FIFO.dut.pdut.reset PASS
ote: $finish    : D:/intelFPGA/21.1/test_synchronous_FIFO.sv(90)
ime: 186 ns  Iteration: 0  Instance: /test_synchronous_FIFO
```

Figure 25

Την στιγμή 177ns το rst_ και το fifo_read είναι 1, το fifo_write είναι 0, τα δεδομένα εξόδου είναι 59, ο cnt είναι 0, ο rd_ptr και ο wr_ptr είναι 9, το fifo_empty είναι 1 => PASS ο έλεγχος No2, και την χρονική στιγμή 179ns PASS ο έλεχγος No5, (η FIFO είναι κενή και υπάχει αίτημα ανάγνωσης (αλλά όχι εγγραφής), τότε ο δείκτης ανάγνωσης παραμένει σταθερός, 9 στην περίπτωση αυτήν).

Την στιγμή 179ns το rst_ και το fifo_read είναι 1, το fifo_write είναι 0, τα δεδομένα εξόδου είναι 59, ο cnt είναι 0, ο rd_ptr και ο wr_ptr είναι 9, το fifo_empty είναι 1 => PASS ο έλεγχος No2, και την χρονική στιγμή 181ns PASS ο έλεχγος No5, (η FIFO είναι κενή και υπάχει αίτημα ανάγνωσης (αλλά όχι εγγραφής), τότε ο δείκτης ανάγνωσης παραμένει σταθερός, 9 στην περίπτωση αυτήν).

Την στιγμή 181ns το rst_ και το fifo_write είναι 1, το fifo_read είναι 0, τα δεδομένα εξόδου είναι 59, ο cnt είναι 0, ο rd_ptr και ο wr_ptr είναι 9, το fifo_empty είναι 1 => PASS ο έλεγχος No2.

Την στιγμή 183ns το rst_ και το fifo_read είναι 1, το fifo_write είναι 0, τα δεδομένα εξόδου είναι 59, ο cnt είναι 1, ο rd_ptr είναι 9 και ο wr_ptr είναι 10, το fifo_empty και το fifo_full είναι 0 => γίνεται ανάγνωση δεδομένων, η FIFO δεν είναι κενή ή πλήρης (0<cnt<16), οπότε δεν θα γίνει η αποτίμηση των ελέγχων.

Την στιγμή 185ns, το rst_ έχει ήδη πέσει στο 0, έχει γίνει fifo_data_out=0 => PASS ο έλεγχος No1 της εκφώνησης (το property που σχετίζεται με το reset -> No1).

Παρατήρηση: Στις εικόνες με κώδικα δεν υπάρχουν τα σχετικά σχόλια, ενώ στα αρχεία .sv που συνοδεύουν την παρούσα αναφορά έχουν προστεθεί σχόλια για την καλύτερη κατανόηση του κώδικα.

3 Βιβλιογραφία-Πηγές

1) Υλικό Μαθήματος Ψηφιακά Συστήματα HW-2, ΤΗΜΜΥ ΑΠΘ, Βασίλειος Παυλίδης.
2) Ψηφιακή Σχεδίαση, 6η Έκδοση, Mano Morris, Ciletti Michael