

گزارش کار آزمایشگاه DSD

**آزمایش شماره 3**



10 فروردین 1400

عرشا اخوان

محمدحسین عبدی

علیرضا ایلامی

|  |  |  |
| --- | --- | --- |
| تاریخ آزمایش: 10 فروردین 1400 | موضوع: توصیف جریان داده | **شماره آزمایش: 3** |
| **علیرضا ایلامی**  **97101286** | **محمدحسین عبدی**  **97110285** | **عرشیا اخوان**  **97110422** |

مقدمه:

در این آزمایش قصد داریم دو مقایسه‌گر، یکی cascadable 4 bit comparator و دیگری یک serial comparator بسازیم.

* این آزمایش صرفا کد وریلاگ است.

شرح آزمایش:

در هر دو حالت سریال و حالت 4 بیت ثابت، یک بیت LGN داریم. (Less / Greater Not) که در مقایسه دو عدد، اگر اولی از دومی کوچکتر بود خروجی یک میشود. در غیر اینصورت خروجی صفر است.

البته حالت تساوی اصلا اینجا چک نمیشود.

ابتدا شرط مساوی بودن دو عدد را با یکدیگر چک میکنیم اگر مساوی بود، اصلا LGN بررسی نمیشود و صفر خروجی میدهد. در غیر اینصورت محاسبه LGN آغاز میشود.

برای حالت سریال، باید با کلاک کار کنیم. و برای این آزمایش نیاز بود که DFF را خودمان طراحی کنیم.

ابتدا ماژول فلیپ‌فلاپ را در فایل وریلاگ dff.v طراحی کردیم و پس از تست کارکرد آن، آن را داخل بقیه کد وریلاگ comp\_seq.v (خطوط 17 تا 20) آوردیم. (در صورت آزمایش گفته شده از هیچ توصیف اضافه‌ای تعریف نکنید.)

برای طراحی DFF، دقیقا مانند معماری خود فلیپ فلاپ از 4 گیت NAND استفاده کرده‌ایم.

گزارش تست بنچ:

تست بنچ در فایلی به نام testbench.v نوشته شده است.

در ابتدا دو ماژول ترتیبی و ترکیبی Comparator (مقایسه‌کننده) را تعریف کرده و کلاک را نیز ست کرده‌ایم.

در فایل seed.dat یک عدد رندوم قرار داده ایم.

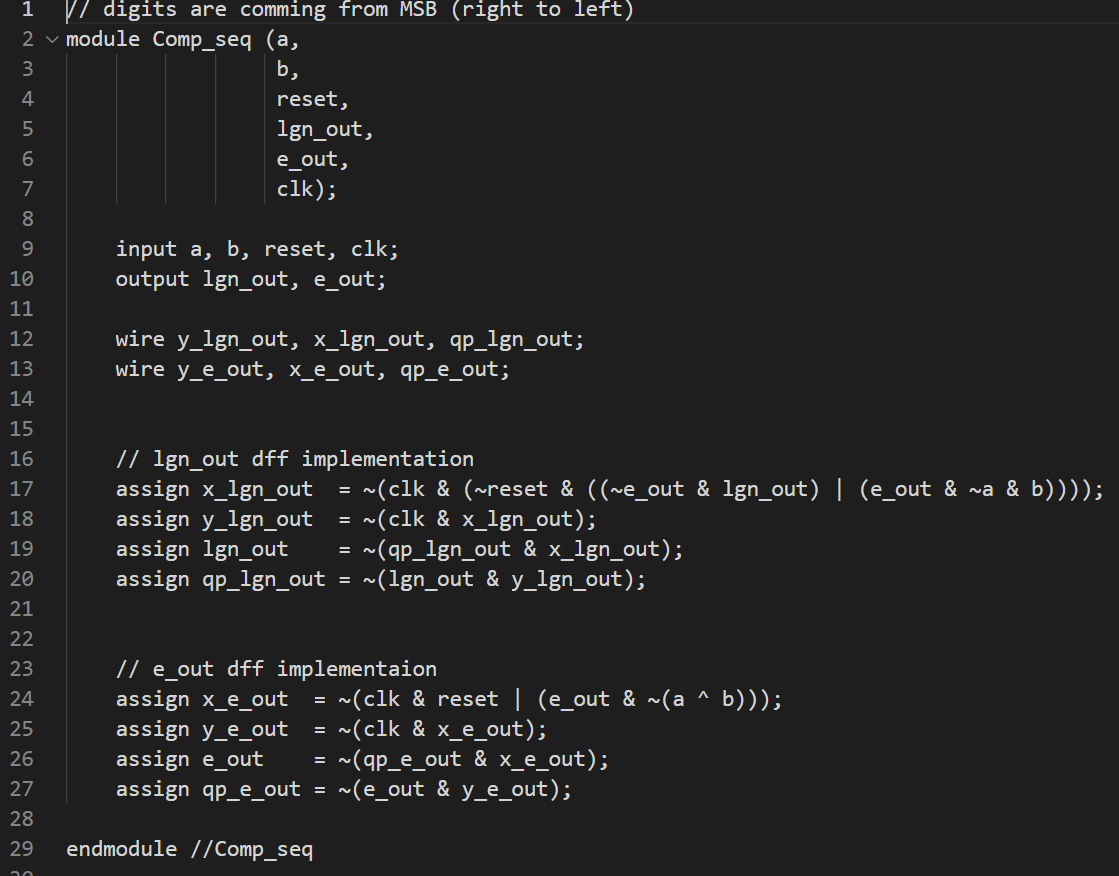
این عدد رندوم، همان random$ ماست.

برای اینکه با هر سیمولیت این رندوم بودن ما تفاوت کند، ما به کمک یک بش در فایل seed.dat عدد رندوم را نوشته ایم. (به علت محدودیت لینوکس اینکار را کرده ایم. در نسخه ویندوز $random flag وجود دارد.)

سپس، ابتدا ماژول ترکیبی را سیمولیت کرده (خطوط 55-62) و سپس ماژول ترتیبی را شبیه سازی کرده ایم. (63-79)

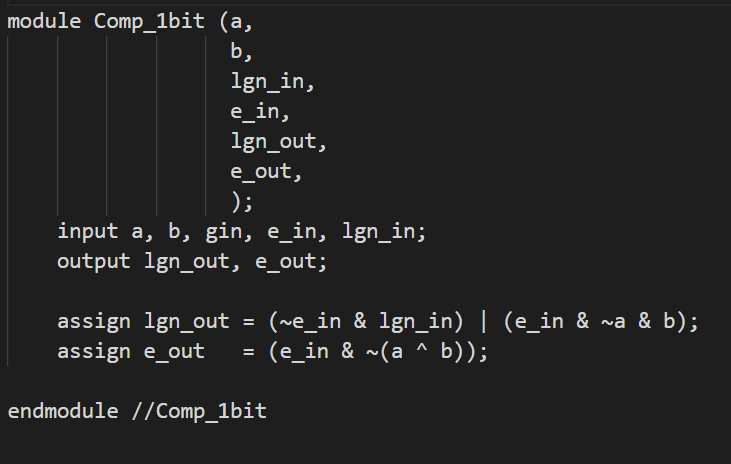
یک فایل تست بنچ نیز برای خود فلیپ فلاپ نوع D که در مقایسه‌کننده ترتیبی از آن استفاده کرده‌ایم، نوشته شده است. (برای زمانی که جدا نوشته شده بود و تستش کردیم.)

کد وریلاگ مقایسه‌گر ترتیبی در شکل زیر موجود است:

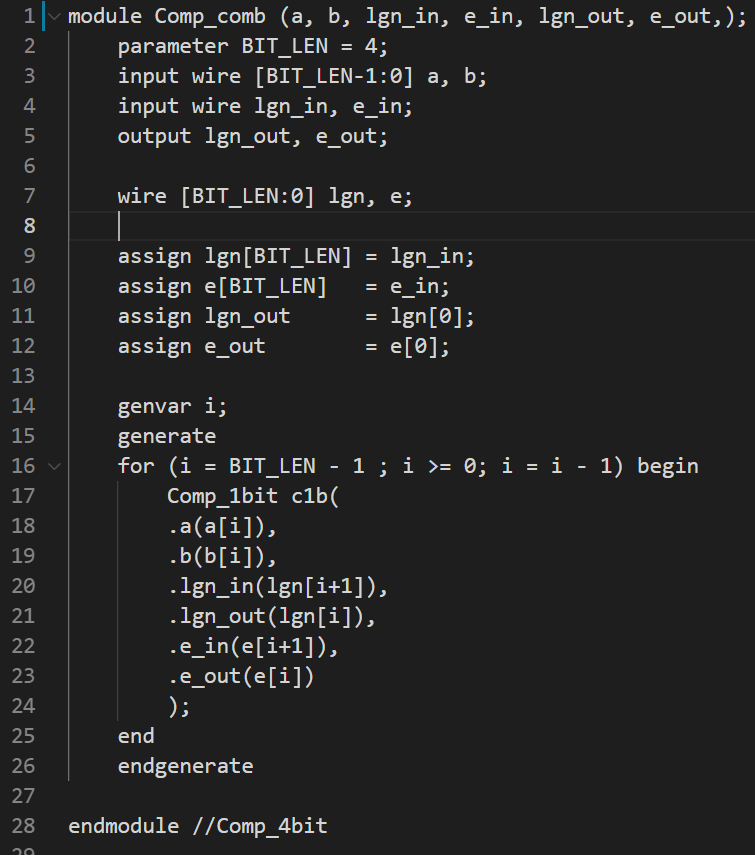


کد وریلاگ مقایسه‌گر ترکیبی در شکل زیر موجود است:

مقایسه گر یک بیتی:

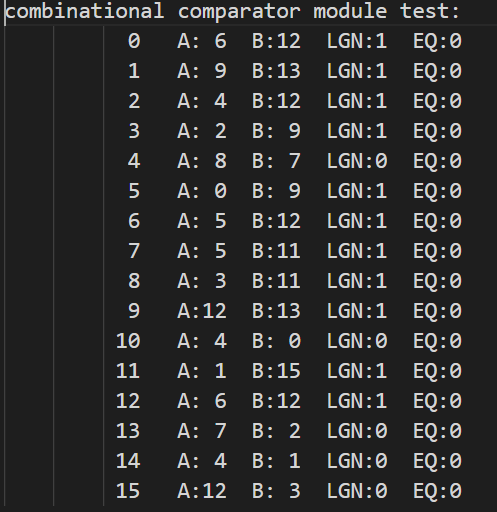


مقایسه گر 4 بیتی:

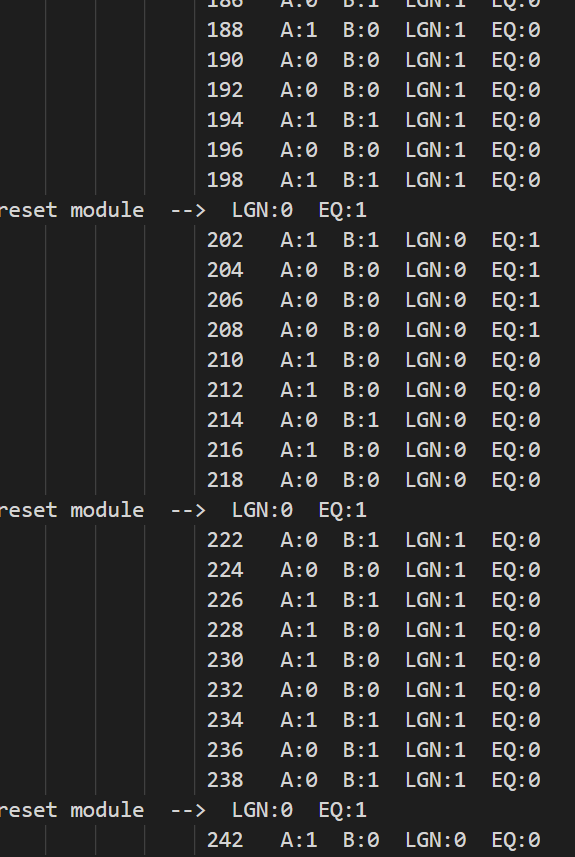


تمامی نتایج تست در پوشه report در فایل result.txt قرار داده شده است.

برای مثال، تست مقایسه‌گر ترکیبی به شرح زیر است:



این هم مثالی از تست مقایسه‌گر ترتیبی:



فایل تست بنچ به شرح زیر است:

`timescale 1ns/1ps

`define NULL 0

module testbench();

reg [3:0] a, b;

reg A, B;

reg reset, clk;

wire lgn, e, LGN, E;

Comp\_comb #(.BIT\_LEN(4)) comp0(

.a(a),

.b(b),

.lgn\_in(1'b0),

.e\_in(1'b1),

.lgn\_out(lgn),

.e\_out(e)

);

Comp\_seq comp1(

.a(A),

.b(B),

.reset(reset),

.lgn\_out(LGN),

.e\_out(E),

.clk(clk)

);

always

begin

clk = 0;

forever #1 clk = ~clk;

end

integer data\_file;

integer scan\_file;

integer seed;

initial begin

data\_file = $fopen("seed.dat", "r");

if (data\_file == `NULL) begin

$display("data\_file handle was NULL");

$finish;

end

scan\_file = $fscanf(data\_file, "%d", seed);

if (scan\_file == `NULL) begin

$display("integer read error");

$finish;

end

end

integer i, j;

initial begin

#2 // waiting for reading random seed from file "seed.dat"

$display("combinational comparator module test:");

for (i = 0; i < 16; i = i + 1) begin

a = {$random(seed)}%16;

b = {$random(seed)}%16;

#1

$display(i, " A:%d B:%d LGN:%b EQ:%b", a, b, lgn, e);

end

$display("sequantial comparator module test:");

for (i = 0; i < 16; i = i + 1) begin

A = 0;

B = 0;

reset = 1;

#2;

$display("reset module --> LGN:%b EQ:%b", LGN, E);

j = {$random(seed)}%20+1;

while (j > 0) begin

A = {$random(seed)}%2;

B = {$random(seed)}%2;

reset = 0;

#2

$display($time, " A:%d B:%d LGN:%b EQ:%b", A, B, LGN, E);

j = j - 1;

end

end

$finish;

end

endmodule

پایان