به نام خدا

گزارش فعالیت عملی در آزمایشگاه
آزمایش 3 درس آزمایشگاه DSD

امیرمحمد شربتی (402106112) – پوریا رحمانی (402111418)

هفته سوم – 1404/8/26

این توضیحات در مورد کاری ست که در آزمایشگاه انجام دادیم و مربوط به پیاده سازی عملی مدار روی برد FPGA است. این بخش در مورد آزمایش 3 (مقایسه کننده ترتیبی و ترکیبی) است.

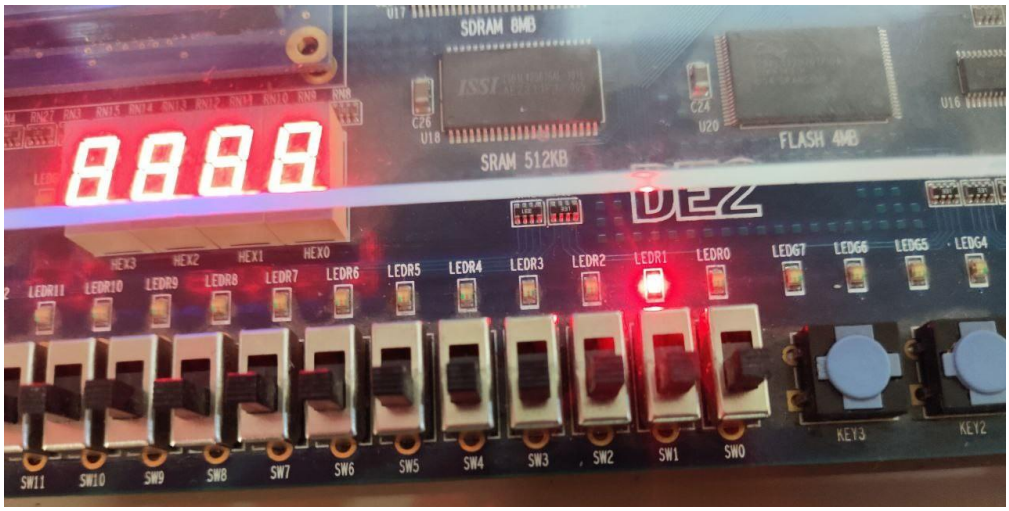
توجه شود که بخش عمده توضیحات در مورد کد و چالش ها و ایرادات احتمالی و... در پیش گزارش است. فایلی به عنوان report که در کنار سایر فایل ها، در قسمت پیش گزارش آپلود شده است دارای توضیحات مفصلی برای همین است.

بنابراین در اینجا صرفا کارهایی که در آزمایشگاه انجام دادیم و نتایجی که دریافت کردیم و چالش های پیاده سازی روی FPGA ذکر می‌شود و برای سایر جزئیات مهم پیاده سازی و شبیه سازی و... شما را به فایل report (که پیش گزارش است) ارجاع می‌دهم.

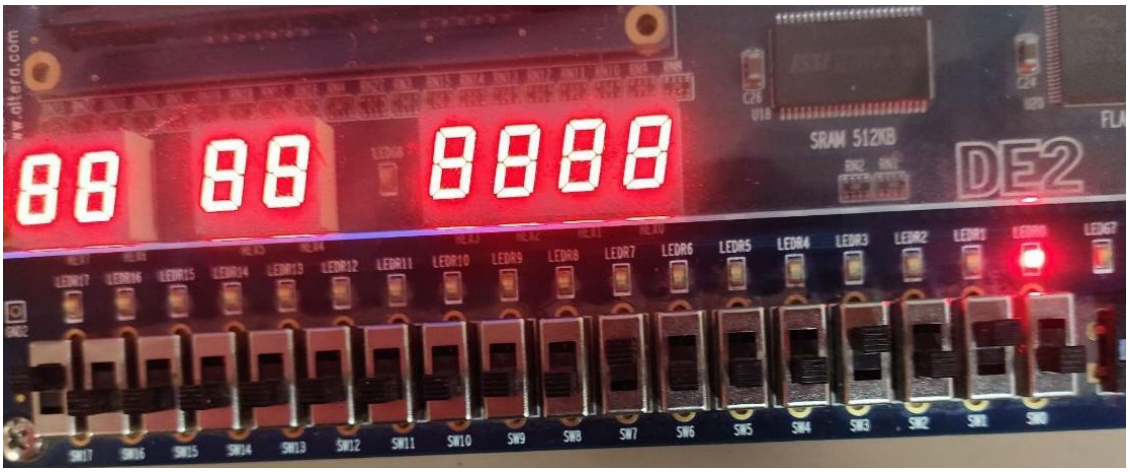
این آزمایش شامل دو بخش است که در بخش اول باید یک مدار ترکیبی پیاده سازی میشد. خدا رو شکر این کد بدون مشکل روی FPGA اجرا شد و ما نتایج مطلوب مدار را مشاهده کردیم.

در این آزمایش top level entity پروژه، فایل combinational_comparator.v است که روی برد تست شد. ابتدا باید 8 بیت برای جفت اعداد 4 بیتی ورودی روی مدار تنظیم کنیم. برای 8 بیت ورودی از پین های sw استفاده کردیم. پین های sw0 تا sw3 برای عدد اول (در کد عدد a) و پین های sw4 تا sw7 برای عدد دوم (یعنی همان b). سه خروجی داریم که این ها را با LED های برد مشخص کردیم. LEDR0 یعنی عدد اول بزرگتر از عدد دوم است، یعنی a > b. LEDR1 به معنای برابری و LEDR2 یعنی عدد اول کوچکتر از عدد دوم است، یعنی a < b.

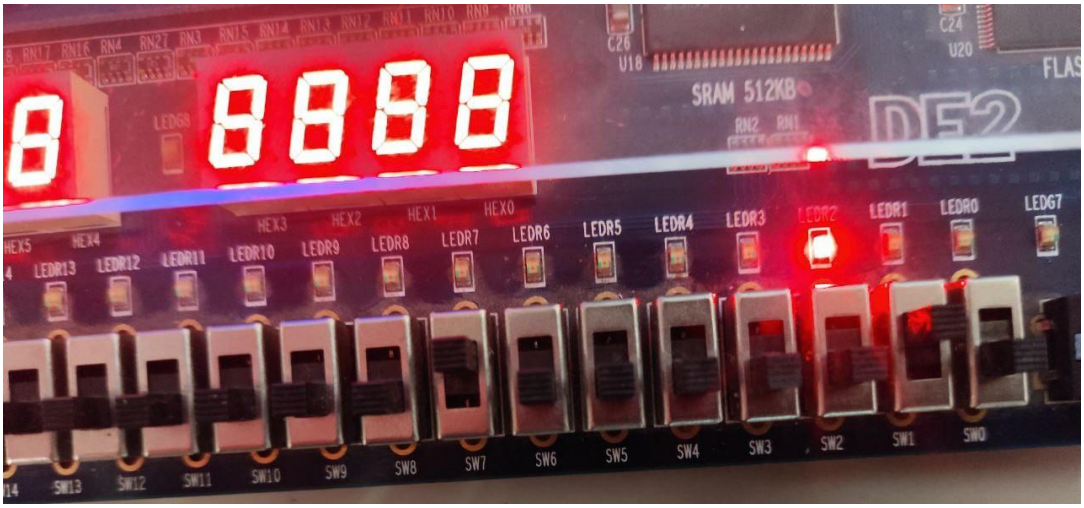
همه چیز طبق انتظار پیش رفت. در اینجا عکس برخی تست هایی که در آزمایشگاه انجام دادیم، قرار می‌دهم:



در این دو عکس دو عدد ورودی برابر هستند (در عکس اول هر دو صفر و در عکس دوم هر دو 10 یا همان 1010 در مبنای دو هستند.) و به درستی خروجی a_equal_b یک می‌شود و بقیه صفر هستند.



در این دو تست هم به درستی a > b نشان داده شده است.



این عکس هم مثالی از حالتی است که b > a و خروجی مدار a_less_b = 1 است و مابقی صفر هستند.

پس از این به آزمایش 2 می‌پردازیم. همانطور که در فایل report که مربوط به پیش گزارش است ذکر شده است، این آزمایش پیچیدگی های بیشتری نسبت به آزمایش قبل دارد. بنده از تکرار مواردی که در پیش گزارش ذکر شده خودداری کرده و صرفا مواردی که در آزمایشگاه به آن بر خوردیم را توضیح می‌دهم.

یکی از مشکلاتی که در آزمایشگاه داشتیم (و با جنابعالی هم صحبت کردم) این بود که اگر clk = 1، یعنی وقتی کلاک high است، ورودی مدار تغییر کند، آیا در این کلاک خروجی باید تغییر کند یا نه. مثلا وقتی clk = 1 بیت اول 5 بار تغییر کند باید چه شود. کدی که قبلا پیاده سازی شده بود، در کلاک بالا به تغییر ورودی حساس بود. یعنی نتیجه در یک کلاک به خاطر تغییر ورودی تغییر می‌کرد. اما بهتر بود مداری پیاده سازی شود که وقتی در clk = 1 مقدار ورودی تغییر می‌کند، خروجی تغییر نکند (در واقع ورودی های جدید را ignore کند) و در کلاک بعدی خروجی آخرین ورودی را نشان دهد.

برای اجرای این منطق (که قبلا به آن فکر شده بود و سریع در آزمایشگاه اصلاح شد)، کافی بود سه سیگنال دیگر هم اضافه شود که مشابه prev_a_less/equal/greater_b فقط وقتی کلاک پایین است تغییر کند. در این صورت وقتی کلاک بالا است، خروجی فعلی مدار a_less/equal/greater_b به ورودی ها حساس نیست و خروجی در طول بالا بودن کلاک ثابت است.

```
assign equal_one_bit = reset ? (clk ? equal_one_bit : (~(a ^ b))) : 1'b1;
assign a_greater_one_bit = reset ? (clk ? a_greater_one_bit : (a & ~b)) : 1'b0;
assign a_less_one_bit = reset ? (clk ? a_less_one_bit : (~a & b)) : 1'b0;

assign a_equal_b = reset ? (clk ? (prev_a_equal_b & equal_one_bit) : a_equal_b) : 1'b1;
assign a_greater_b = reset ? (clk ? (a_greater_one_bit | (prev_a_greater_b & equal_one_bit)) : a_greater_b) : 1'b0;
assign a_less_b = reset ? (clk ? (a_less_one_bit | (prev_a_less_b & equal_one_bit)) : a_less_b) : 1'b0;
```

بنابراین این پیاده سازی کمی تغییر کرد و به همین دلیل ما این کد را به پیوست گزارش پس از آزمایش مجددا ارسال کردیم.

بنده ماژول تست این مدار را هم برای ارزیابی صحت آن کمی تغییر دادم و وقتی clk = 1 بود، ورودی ها را تغییر دادم اما بدون مشکل خروجی درست را دیدم.

```
for (i = 0; i < 8; i = i + 1) begin
    current_bit_a = A[i];
    current_bit_b = B[i];
    #10 clk = ~clk;  // simulate level change

    // When the clock is high, the input bits change, but we don't see the effect on the outputs.
    current_bit_a = 1'b0;
    current_bit_b = 1'b1;
    #10;
    $display("time=%0t  i=%0d || A=%b  B=%b || a_greater_b=%b  a_equal_b=%b  a_less_b=%b",
            $time, (i), get_partial(A, i), get_partial(B, i), a_greater_b, a_equal_b, a_less_b);
    clk = ~clk;  // return clock
end
```

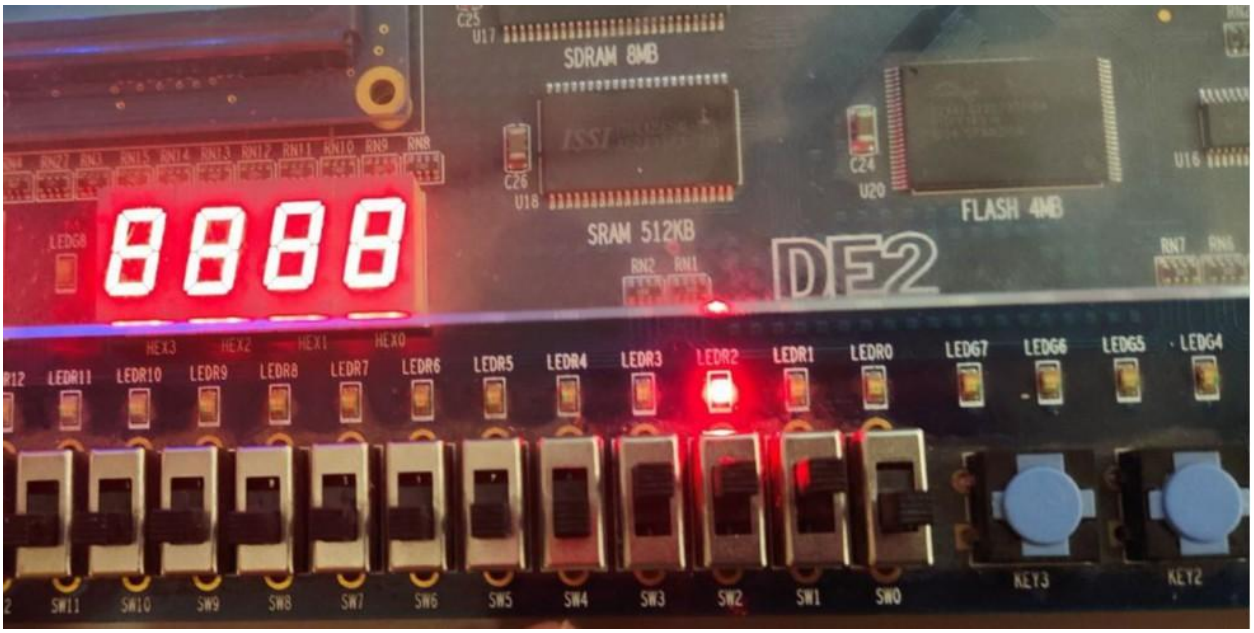
به خاطر همین تغییر اندک در testbench، این فایل هم مجددا ارسال شد. (که البته این تغییر مهمی نیست، صرفا کاری ست که برای بررسی صحت مدار قبل از آپلود آن روی برد، انجام شد.)

پس از این تغییرات همه چیز مطابق انتظار است.

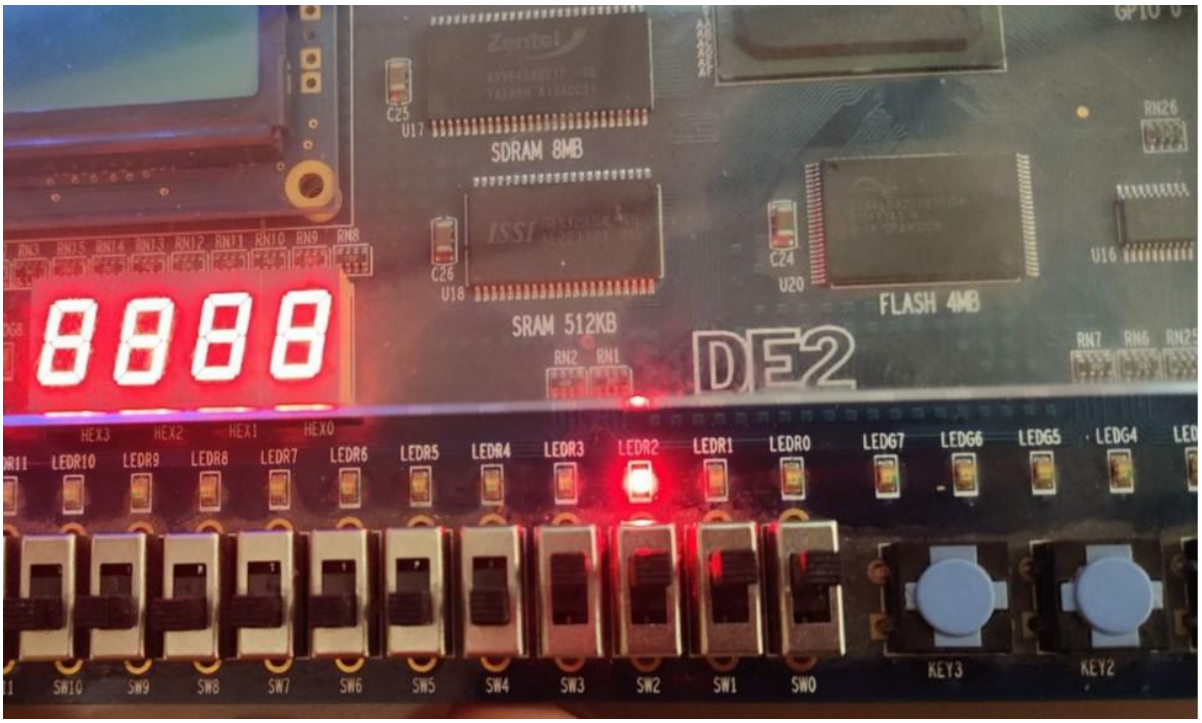
پس از کامپایل، نوبت به تخصیص پین ها می‌رسد. 4 بیت ورودی و سه بیت خروجی داریم. خروجی ها در این حالت عینا مشابه قبل LEDR0 (یعنی a > b)، LEDR1 (a == b) و LEDR2 (a < b) هستند. ورودی ها اما تغییر کرده‌اند. Sw0 برای بیت ورودی عدد اول (a) و sw1 برای بیت ورودی عدد دوم است. sw2 , sw3 هم مربوط به reset و clock هستند. توجه شود که reset = 1 یعنی مدار کار میکند. (active low است).

وقتی کلاک پایین بود بیت های ورودی دو عدد تغییر می‌کرد و وقتی کلاک را بالا می‌دادیم، خروجی درست را میدیدیم. اگر در این حالت ورودی تغییر کند، خروجی تغییر نمی‌کرد. این نتیجه تست دو عدد است که باید به ترتیب آنها را دید. (بیت ها از سمت راست، در واقع بیت کم ارزش می‌آیند). توجه شود توضیح هر عکس پایین آن ذکر شده است:

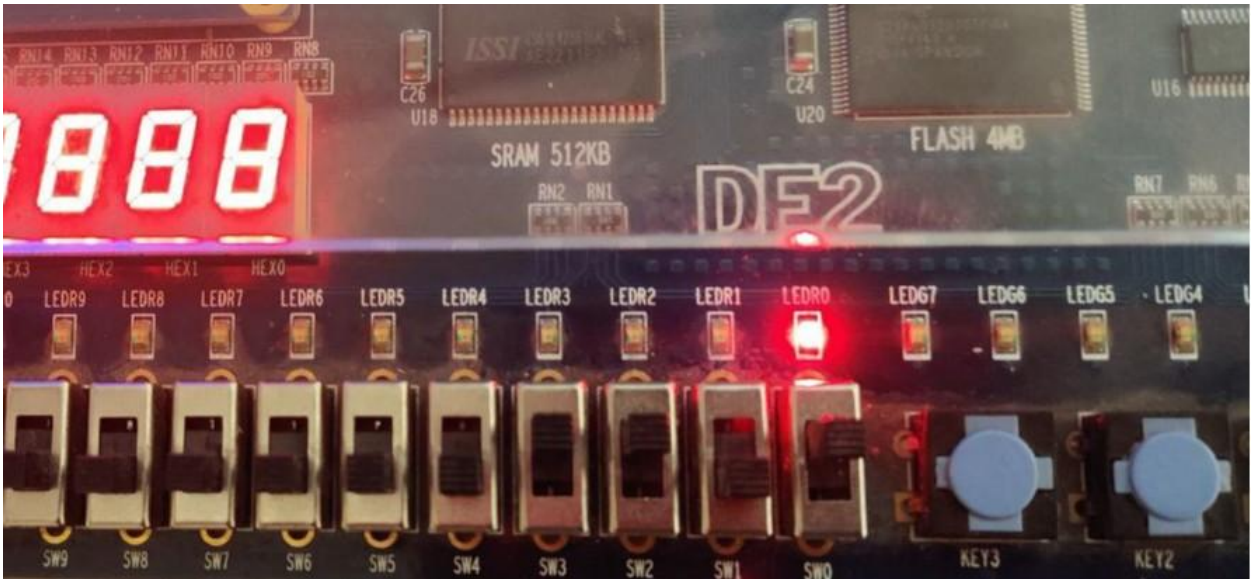
در سمت ورودی در این عکس ها فقط کافی‌ست به sw0 و sw1 که بیت های فعلی a و b هستند، توجه شود. در ابتدا هر دو صفر هستند و خروجی برابری این دو را نشان می‌دهد.



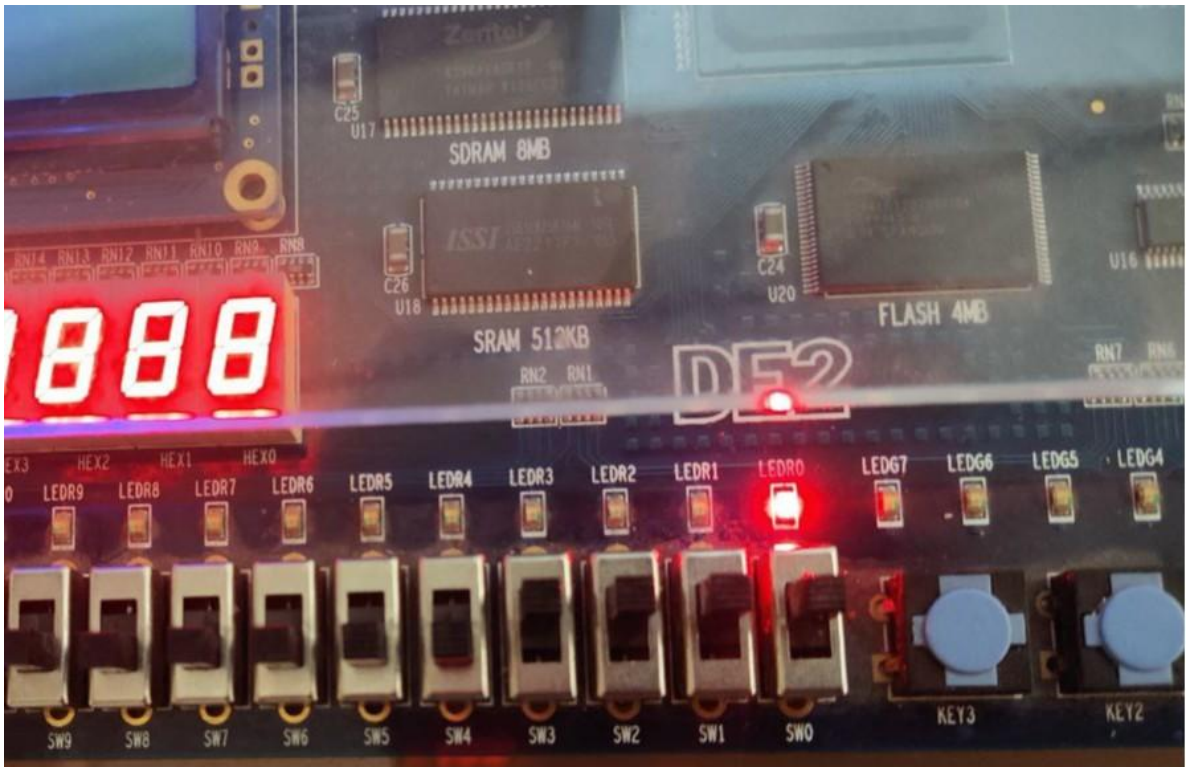
در کلاک بعد، بیت های b = 1 و a = 0 وارد می‌شوند. یعنی دو عدد طبق ورودی های قبل a = 00 و b = 10 هستند. پس b > a که خروجی این عکس است. (a_less_b = 1)



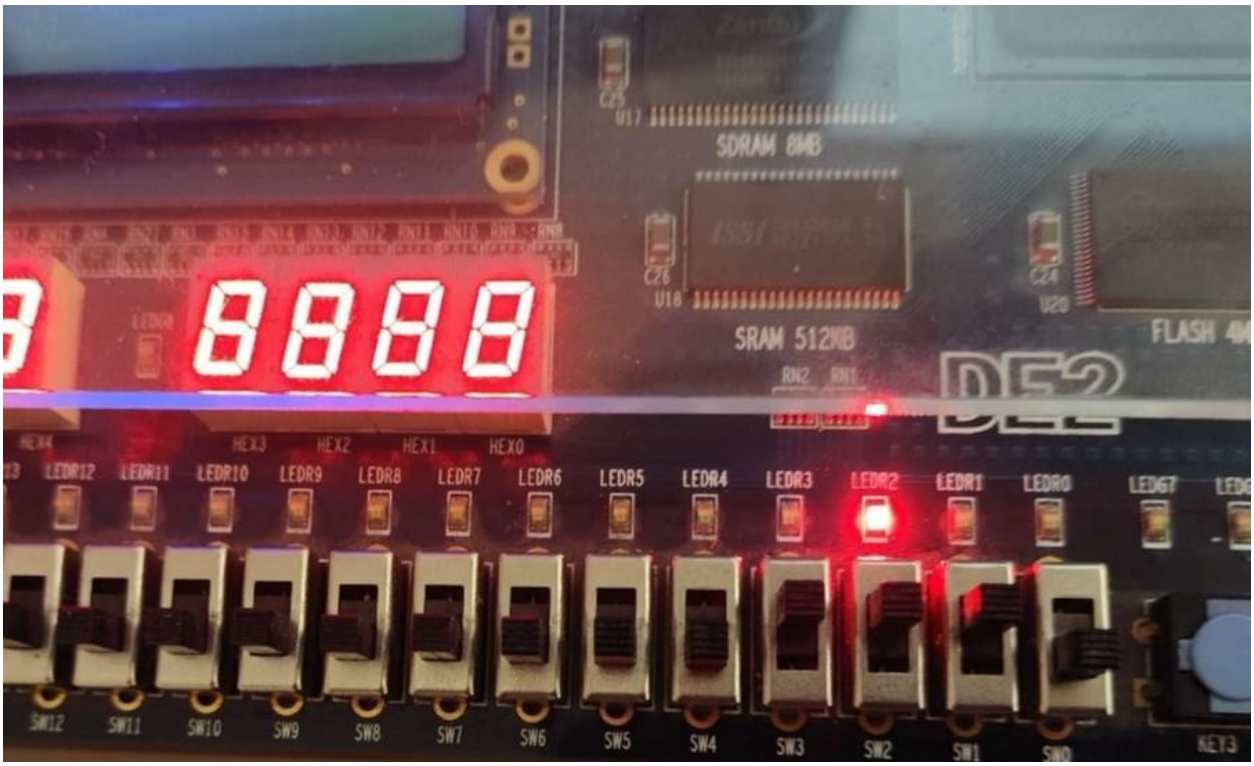
دو بیت بعدی صفر هستند. یعنی a = 000 و b = 010 دو عددی هستند که باید مقایسه شوند. به درستی a<b نشان داده شده است.



بیت بعدی a یک و بیت بعدی b صفر است. یعنی a = 1000 و b = 0010 هستند. پس a > b که خروجی مدار هم همین است.



بعدش هر دو بیت یک هستند. یعنی دو عدد به صورت a = 11000 و b = 10010 هستند. مطابق انتظار خروجی a_greater_b = 1 میشود که یعنی a > b.



برای بیت بعدی هم یک برای عدد دوم و صفر برای عدد اول وارد می‌شوند که یعنی دو عدد a = 011000 و b = 110010 هستند. در این حالت b > a است که مدار هم همین را نشان می‌دهد.

خدا رو شکر همه چیز مطابق انتظار است و هر دو مدار (مقایسه کننده ترکیبی و ترتیبی سریال) به درستی کار کردند.