ازمایش سوم

موضوع: شمارنده ۴ رقمی توسعه یافته

تاریخ آزمایش: ۱۴۰۲/۷/۲۶

استاد: مهندس جوادی

جواد فرجی (۹۹۵۲۲۰۰۵)

محمد رحمانی (۹۷۵۲۱۲۸۸)

ورودی و خروجی:

نسبت به آزمایش قبلی ورودی **reset** اضافه شده است. چهار خروجی **seg_bin_s** که هر کدام برای ۴ بیت برای نمایش باینری اعداد روی تایمر است. خروجی **alarm** که برای آلارم است.

- **تغییر تایمر به ساعت:**

برای این کار کافیست بخشی از کد که جفت رقم دقیقه و ثانیه را که بعد از ۹۹ تبدیل به ۰۰ میکند را بعد از ۵۹ تبدیل به صفر کنیم و در صورت نیاز یک واحد به دقیقه شمار اضافه شود.

سلکت مربوط به دو نقطه میان دو جفت رقم دقیقه و ثانیه را روشن میکنیم.

```
        if (counterseg1 < "1001") then
            counterseg1 := counterseg1 + 1;
        else
            counterseg1 := "0000";
                if (counterseg2 < "0101") then
                    counterseg2 := counterseg2 + 1;
                else
                    counterseg2 := "0000";
                    if (counterseg3 < "1001") then
                        counterseg3 := counterseg3 + 1;
                    else
                        counterseg3 := "0000";
                        if (counterseg4 < "0101") then
                            counterseg4 := counterseg4 + 1;
                        else
                            counterseg4 := "0000";
                        end if;
                    end if;
                end if;
            end if;

end if;
```

- **نمایش باینری اعداد خروجی:**

برای این کار کافیست مقادیر **CountersegSig** را به خروجی های جدیدی که تعریف کرده ایم انتقال دهیم.

```
191
192     seg_bin_s0 <= counterseg1sig;
193     seg_bin_s1 <= counterseg2sig;
194     seg_bin_m0 <= counterseg3sig;
195     seg_bin_m1 <= counterseg4sig;
```

نکته: این قطعه کد نباید در پروسه باشد و نیاز است در هر بار رندر این اتفاق تکرار شود.

- **تنظیم آلارم:**

برای تنظیم آلارم، نیاز است که در یک زمان مشخص، خروجی **alarm** را که به طور پیشفرض مقدار صفر دارد را یک میکنیم.

نکته: در اینجا آلارم در لحظه ۰۱:۱۰ فعال می شود و بعد از آن خاموش می‌شود.

```
if(counterseg1 = "0000" and counterseg2 = "0001" and counterseg3 = "0001" and counterseg4 = "0000") then
    alarm <= '1';
else
    alarm <= '0';
end if;
```

- **تنظیم کردن reset:**

برای این کار کافیست در پروسه ای که متغیرهای **counterseg** در آن قرار دارند. با یک شدن مقدار **reset**، تمام مقادیر متغیر های **counterseg** برابر "۰۰۰۰" شوند.

```
if(reset /= '1') then
    counterseg1 := "0000";
    counterseg2 := "0000";
    counterseg3 := "0000";
    counterseg4 := "0000";
end if;
```

- تنظیم **start** و **stop**:

برای این کار باید یک پروسه و دو کلید **start** و **stop** در نظر بگیریم. یک سیگنال که برای ذخیره کردن وضعیت تایمر به کار میرود به اسم **holder** که مقدار صفر آن برای **stop** و مقدار **1** آن برای **Start** است.

نکته: باید در فایل **ucf** دو خط اضافی بنویسیم که بیانگر این است که داریم از کلیدها، کاربردی مانند کلاک میگیریم ولی کلاک نیستند.

```
NET "start_btn" CLOCK_DEDICATED_ROUTE = FALSE;
NET "stop_btn" CLOCK_DEDICATED_ROUTE = FALSE;
```

```
process(start_btn, stop_btn)
    begin
        if(start_btn = '0') then
            holder <= '0';
        elsif(stop_btn = '0') then
            holder <= '1';
        end if;
    end process;
```

مپ کردن خروجی ها به روی fpga:

برای مپ کردن روی برد های fpga، با استفاده از داکیومنت موجود، این خطوط را داخل فایل ucf قرار میدهیم:

```
NET "gclk" CLOCK_DEDICATED_ROUTE = FALSE;
NET "start_btn" CLOCK_DEDICATED_ROUTE = FALSE;
NET "stop_btn" CLOCK_DEDICATED_ROUTE = FALSE;
NET "gclk" LOC = P184;
NET "seg_data[0]" LOC = P10;
NET "seg_data[1]" LOC = P7;
NET "seg_data[2]" LOC = P11;
NET "seg_data[3]" LOC = P5;
NET "seg_data[4]" LOC = P4;
NET "seg_data[5]" LOC = P12;
NET "seg_data[6]" LOC = P9;

NET "seg_sel[0]" LOC = P15;
NET "seg_sel[1]" LOC = P20;
NET "seg_sel[2]" LOC = P19;
NET "seg_sel[3]" LOC = P18;
NET "seg_sel[4]" LOC = P16;

NET "seg_bin_s0[0]" LOC = P42;
NET "seg_bin_s0[1]" LOC = P43;
NET "seg_bin_s0[2]" LOC = P44;
NET "seg_bin_s0[3]" LOC = P45;
NET "seg_bin_s1[0]" LOC = P46;
NET "seg_bin_s1[1]" LOC = P48;
```

```
NET "seg_bin_s1[2]" LOC = P50;
NET "seg_bin_s1[3]" LOC = P51;
NET "seg_bin_m0[0]" LOC = P61;
NET "seg_bin_m0[1]" LOC = P62;
NET "seg_bin_m0[2]" LOC = P63;
NET "seg_bin_m0[3]" LOC = P64;
NET "seg_bin_m1[0]" LOC = P65;
NET "seg_bin_m1[1]" LOC = P67;
NET "seg_bin_m1[2]" LOC = P68;
NET "seg_bin_m1[3]" LOC = P71;

NET "reset" LOC = P189;
NET "alarm" LOC = P13;

NET "stop_btn" LOC = P190;
NET "start_btn" LOC = P185;
```

اجرای برنامه بر روی برد fpga:

1. Synthesize
2. Implement design
3. Generate programming

در این سه مرحله گزینه run را میزنیم و در صورتی که مشکل خاصی در برنامه وجود نداشته باشد و به باگ نخوریم به مرحله بعد می‌رویم.

4. Impact

با استفاده از این برنامه، فایل باینری ساخته شده را به programmer انتقال میدهیم و programmer این برنامه را روی بردهای fpga اجرا میکند.