دانشکده مهندسی برق

مدارهای منطقی و سیستم‌های دیجیتال

آزمایش 7 - طراحی، پیاده‌سازی و تست مدارهای ترتیبی بر روی FPGA (قسمت اول)

آزمایش ۷ - طراحی، پیاده سازی و تست مدار های ترتیبی بر روی FPGA (قسمت اول)

در این آزمایش به آشنایی با سیگنال clock در بخش PL بورد ZYNQ و مفهوم تاخیر زمانی خواهیم پرداخت.

قطعات و تجهیزات مورد نیاز

- بورد Zynq (AXPZ7010)
- یک عدد کابل USB و سیم پاور بورد
- نرم‌افزار Vivado 2019.1

۱ـ پیش گزارش

۱. با مراجعه به دیتاشیت تراشه Zynq (بخش ۳-۲)، مشاهده می کنیم که کریستال سازنده کلاک برای بخش PL (FPGA)، به پین U18 متصل شده است. مقدار فرکانس آن را پیدا کنید. برای ایجاد تاخیر ۲ ثانیه ای، چند کلاک باید بگذرد؟ عدد حاصل در چند بیت جا می شود؟

۲. کد وریلاگ ماژولی را بنویسید که در هر کلاک یک LED را روشن یا خاموش کند. ورودی این ماژول سیگنال clk و خروجی آن سیگنال LED است که صفر یا یک شدن آن باعث خاموش یا روشن شدن LED می شود.

۳. کد وریلاگ ماژول LEDBlinkPL را بنویسید که یک LED را هر ۲ ثانیه یک بار خاموش و روشن کند. ورودی ماژول، سیگنال clk و خروجی آن سیگنال LED است. ماژول طراحی شده را شبیه سازی کنید. (راهنمایی: برای ایجاد تاخیر ۲ ثانیه ای از یک counter استفاده کنید که در هر کلاک، مقدار آن یکی افزایش یافته و بعد از تعداد مشخصی کلاک که در سوال یک محاسبه کردید، مقدار آن دوباره صفر شده و LED روشن یا خاموش شود.)

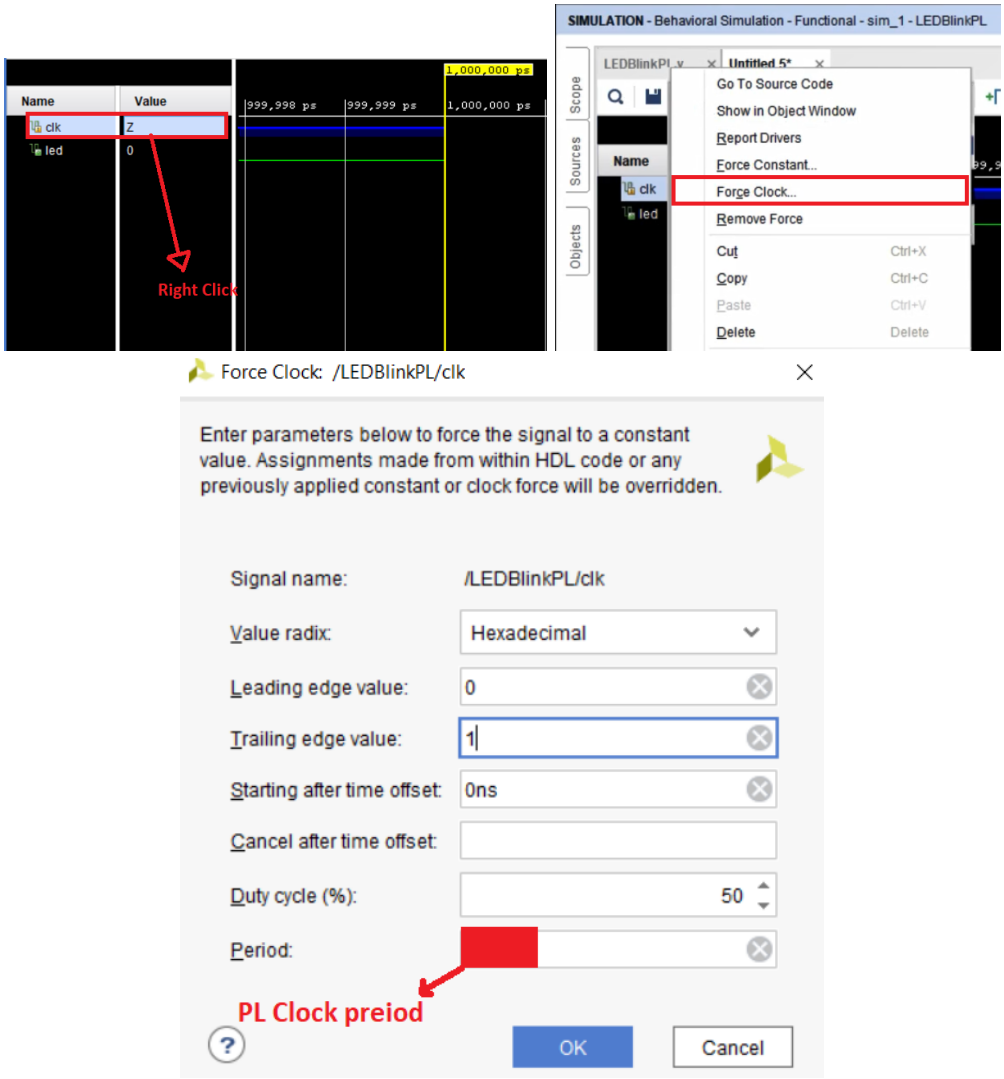
```
module LEDBlinkPL(
          input clk,
          output reg [0:0] LED
  );
      initial LED = 0; //Only for simulation (unsynthesizable)
      //Code Here
  endmodule
```

برای شبیه سازی این ماژول، حداکثر مقدار متغیر فرعی counter را $۱۰^{-۶}$ برابر مقداری که برای ایجاد تاخیر ۲ ثانیه ای به دست آوردید، قرار دهید. یعنی اگر در ماژول طراحی شده، این طور تنظیم کرده اید که با رسیدن counter به مقدار $a \times ۱۰^{۶}$ ، مقدارش صفر شده و خروجی تغییر وضعیت دهد، برای شبیه سازی به شکل زیر عمل کنید:
حداکثر مقدار counter قبل از دوباره صفر شدن را برابر a قرار دهید و سپس با پایان شبیه سازی، برای سنتز و پیاده سازی روی بورد آن را به مقدار اولیه تنظیم شده برگردانید.

همچنین، همان طور که در قطعه کد بالا مشاهده می کنید، خط پنجم (initial) تنها برای شبیه

سازی است تا در هنگام آغاز شبیه سازی، مقدار اولیه ای برای خروجی داشته باشیم. در سنتز و پیاده سازی روی بورد، این خط سنتز نخواهد شد.

برای شبیه سازی سیگنال clk که یک سیگنال متناوب است، طبق مراحل زیر که در تصاویر آورده شده اند، به جای Force Constant که در آزمایش های قبلی استفاده می کردید، در پنجره شبیه سازی روی سیگنال clk راست کلیک کرده و گزینه Force Clock را انتخاب کنید. سپس پنجره باز شده را طبق شکل ۷-۱ تنظیم کنید و دوره تناوب را برابر دوره تناوب کلاک PL قرار دهید و شبیه سازی را راه اندازی کنید.



شکل ۷-۱: نحوه ست کردن سیگنال clk در شبیه سازی

۴. بخش دوم دستور کار را مطالعه کرده و با تکمیل کد زیر، ماژولی طراحی کنید که در آن LED ها طبق شکل ۷-۲، با فرکانس 2 Hz تغییر وضعیت دهند و فرکانس پس از فشردن button1 نصف و پس از فشردن button2 دو برابر شود.

```
module LED_flow(
    input wire clk,
    input wire button1,
    input wire button2,
    output reg [4:0] LED
    );

    // Code Here

endmodule
```

۵. با نحوه ایجاد تاخیر و تغییر فرکانس ورودی به فرکانس مد نظر آشنا شدید. حال با تکمیل کد صفحه بعد، ماژولی طراحی کنید که یک ورودی کلاک (50MHz) و یک ورودی ۲ بیتی به نام mode داشته باشد. طبق جدول ۷-۲، با تنظیم مقادیر mode می خواهیم ماژول طراحی شده، سیگنال کلاکی با فرکانس مطلوب ما تولید نماید:

mode	out_clk freq.
0	4 Hz
1	6 Hz
2	8 Hz
3	10 Hz

جدول ۷-۲: فرکانس clock خروجی ماژول به ازای مقادیر مختلف mode

```
module freq_converter(
    input wire in_clk,
    input wire [1:0] mode,
    output reg out_clk
    );
    // Code Here
endmodule
```

۶. در این بخش می خواهیم یک بازی طراحی کنیم. توضیحات اولیه بخش ۳ دستور کار را مطالعه کنید تا با روند بازی آشنا شوید، سپس ماژول زیر را تکمیل کنید. دقت داشته باشید که تنظیم کلاک توسط ماژول freq_converter که در بخش قبل نوشتید انجام می شود و نیازی نیست در این ماژول از مفهوم counter و ایجاد تاخیر استفاده کنید.

```
module miniGame(
    input wire clk,
    input wire gameReset,
    input wire button,
    output reg [4:0] LED,
    output reg [1:0] mode // Level number
    );

    // Code Here

endmodule
```

(دقت داشته باشید که اسامی ورودی و خروجی ها دلخواه است و لزوما نباید مثل تکه کد بالا باشد اما دکمه ریست بازی را "rst" یا "reset" نام گذاری نکنید چرا که در نرم افزار به دلایلی که در آزمایش بیان می شود به مشکل می خورید.)

۲- دستور کار

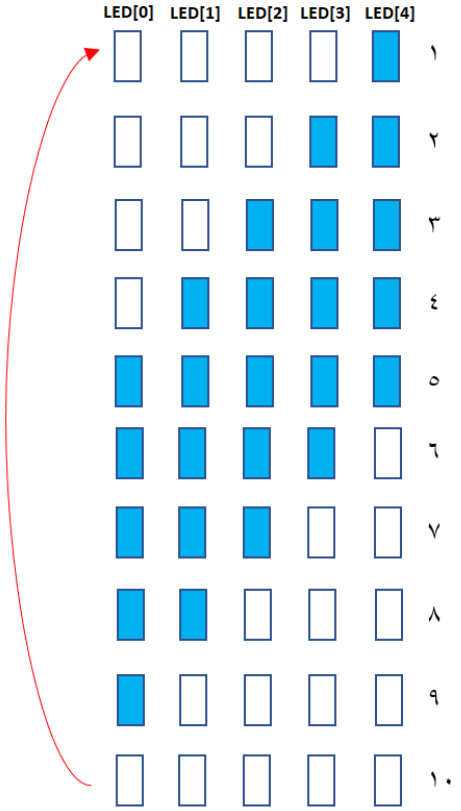
۲-۱- بخش اول: (زمان تقریبی: ۴۵ دقیقه)

۲-۱-۱- مدار توصیف شده در سوال ۲ پیش گزارش را روی بورد پیاده سازی کرده و نتایج را نمایش دهید. همان طور که می دانید، این ماژول یک مدار ترتیبی است؛ پس نیاز به کلاک دارد و باید ورودی کلاک آن به یک منبع نوسان ساز تولید کننده کلاک متصل شود. برای این کار، در بخش I/O ports یا فایل xdc.، با مراجعه به دیتاشیت بورد Zynq، سیگنال clk ماژول خود را به پینی از تراشه (U18) که به کریستال کلاک بورد برای بخش PL متصل شده، وصل کنید و LED را به پین مربوط به یکی از LED های روی بورد (V20, W19, W18, V18, V17) متصل کنید.

۲-۱-۲- کد ماژولی که در سوال ۳ پیش گزارش طراحی کردید را سنتز و روی بورد پیاده سازی کنید و نتیجه را نمایش دهید. برای بخش I/O ports یا فایل xdc. مانند بخش اول عمل کنید.

۲-۲- بخش دوم: (زمان تقریبی: ۴۰ دقیقه)

در این بخش می خواهیم جریانی از LED ها از سمتی وارد و از سمت دیگر خارج شود. مراحل ورود و خروج LED ها را در شکل ۷-۲ مشاهده می کنید:

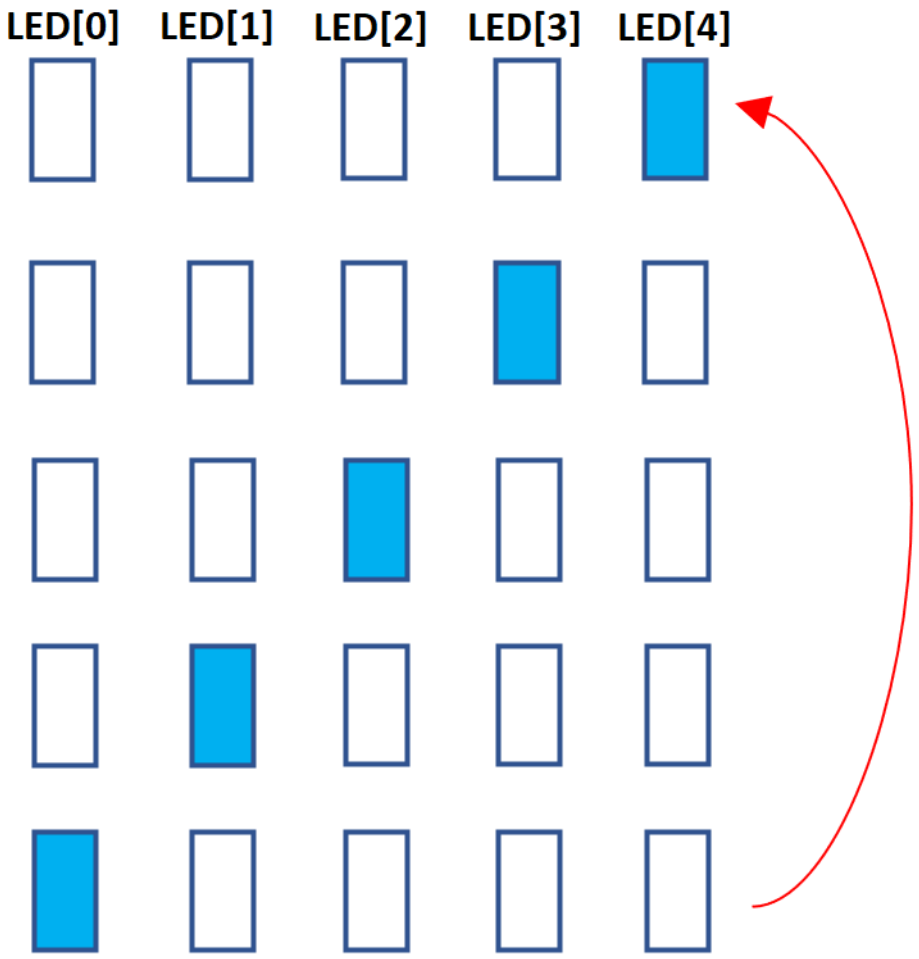


شکل ۷-۲: نحوه تغییر وضعیت LED ها

با توجه به مفهوم counter که در بخش های قبل یاد گرفتید، فرکانس تغییر هر وضعیت به وضعیت بعدی را روی 2 Hz تنظیم کنید. حال می خواهیم با دو کلید متفاوت کنترل سرعت این جریان را به دست بگیریم؛ به گونه ای که با فشردن کلید اول (button1)، سرعت جدید $\frac{1}{2}$ سرعت فعلی شود و با فشردن کلید دوم (button2) سرعت ۲ برابر شود. خروجی شما باید همانند این ویدیو باشد.

۲-۳- بخش سوم (زمان تقریبی: ۹۰ دقیقه)

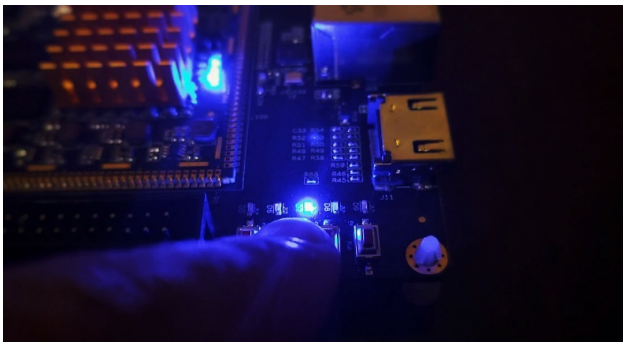
در این بخش قصد داریم یک بازی ساده طراحی کنیم. روند بازی به این شکل است که با شروع بازی یک LED از سمتی روشن شده و به سمت دیگر حرکت می کند و با خارج شدن از طرف دیگر مجددا از سمت اولیه وارد می شود.



شکل ۷-۱۸: روند تغییر وضعیت LED ها

بازیکن می بایست هنگامی که LED وسط روشن می شود، دکمه را فشار دهد. با فشردن دکمه در زمان مقرر، بازیکن به مرحله بعد می رود و LED مجددا از اولین خانه شروع به حرکت می کند. در هر مرحله، سرعت حرکت LED بیشتر و فشار دادن دکمه در زمان مقرر سخت تر می شود. بازی دارای ۳ مرحله با سه سرعت کم، متوسط و سریع است. چنانچه بازیکن به مرحله سریع برسد و آن

را پشت سر بگذارد، تمامی LED ها روشن می شود و بازی به اتمام می رسد و بازیکن می تواند از طریق دکمه gameReset بازی را مجددا تجربه کند. اما در صورتی که در هر بخش دکمه زود یا دیر تر از موعد فشار داده شود، بازی از مرحله ۱ که LED دارای سرعت کم است ، از سر گرفته می شود.
خروجی بازی شما باید مشابه این ویدیو باشد. (در ابتدا نحوه برنده شدن در بازی و در ادامه فرآیند شروع مجدد بازی در صورت باختن را مشاهده می کنید.)



شکل ۷-۱۹: تصویری از ویدیوی خروجی صحیح

برای پیاده سازی این بخش، مراحل زیر را انجام دهید:

- قدم اول (ماژول freq_converter) :

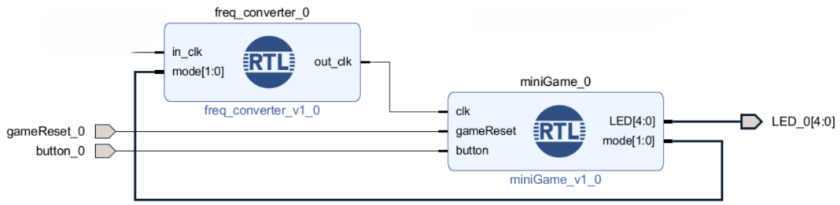
در ۲ بخش قبل مشاهده کردید که برای آنکه سرعت حرکت LED معقول و قابل رؤیت باشد از یک متغیر کمکی مانند counter کمک گرفتیم. برای اینکه از درگیری با این امر در ماژول گیم فارغ شویم، ابتدا یک ماژول کم و زیاد کننده فرکانس می سازیم. این ماژول را که در سوال ۵ پیش گزارش نوشته اید، وارد محیط Vivado کنید.

- قدم دوم (ماژول miniGame) :

همانطور که بالاتر گفته شد، این ماژول دارای یک ورودی کلاک و دو ورودی دکمه که یکی برای ثبت وضعیت LED و دیگری برای ریست کردن بازی است. خروجی ۵ بیتی آن نیز به LED ها اختصاص داده می شود و خروجی ۲ بیتی آن نشانگر mode یا مرحله ای است که بازیکن در آن قرار دارد. این ماژول را که در سوال ۶ پیش گزارش نوشته اید وارد محیط Vivado کنید.

- قدم سوم (برقراری اتصالات در محیط Block Design) :

در نهایت، اتصالات دو ماژول قبل را مطابق شکل ۷-۲۰ برقرار کنید:



شکل ۷-۲۰: شکل نهایی Block design