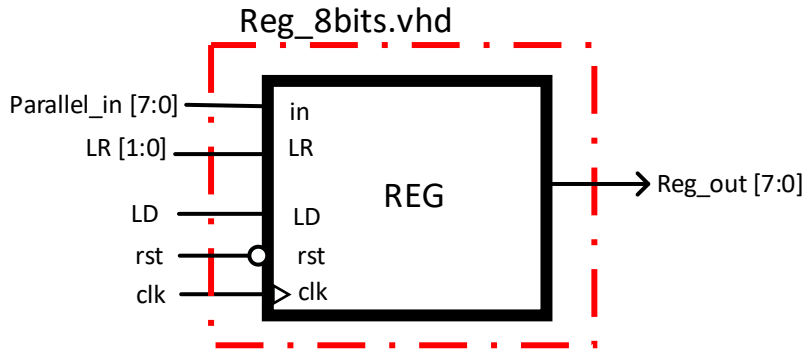
توضیحات آزمایش ششم

هدف از این این آزمایش طراحی رجیستر با قابلیت انتقال ریاضی و منطقی به چپ و راست است. با استفاده از D-Flip flop یک رجستر ۸ بیتی با داده‌های ورودی موازی و پورت‌های ورودی و خروجی همانند شکل 1 باشد طراحی کنید. این طرح باید با توجه به سیگنال های ورودی عملکرد مناسب را انجام دهد. پورت‌ها به صورت زیر تعریف می‌شوند:

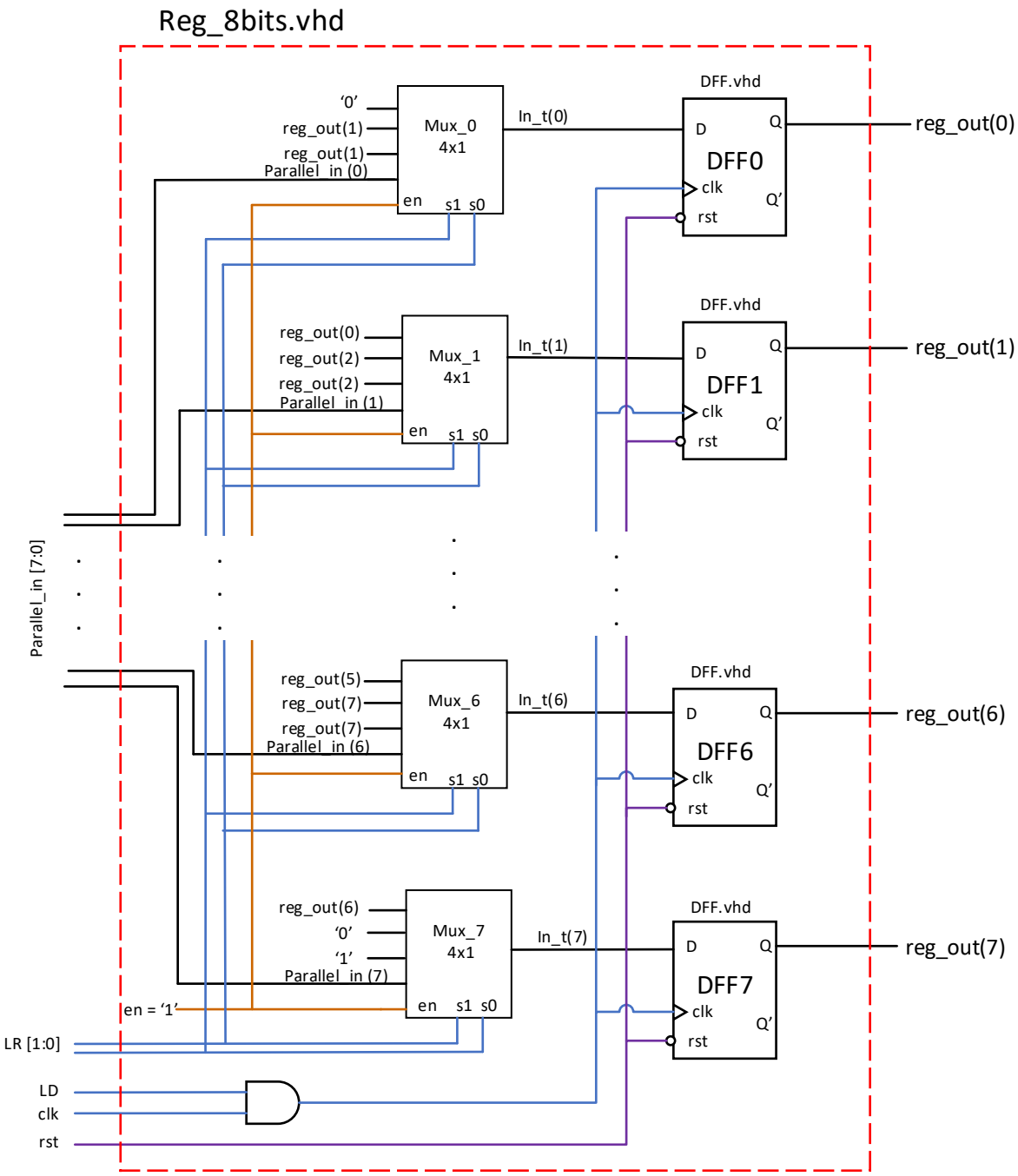
- Parallel_in ورودی رجیستر خاصیت load موازی است و مقادیر آن همزمان در رجیستر ذخیره می‌شوند.
- clk ورودی پالس ساعت است رجیستر باید حساس به لبه بالارونده کلاک باشد.
- سیگنال کنترلی rst غیرهمگام با پالس ساعت و Low-active باید باشد. rst تک بیتی و از نوع std_logic است و زمانی که مقدار سیگنال rst= ‘0’ باشد، مقدار رجیستر برابر با ”00000000“ گردد. در غیر این صورت رجیستر با توجه به ورودی‌های کنترلی رفتار کند.
- ورودی load تک بیتی و از نوع std_logic تعریف گردد. در صورت یک بودن، ورودی ۸ بیتی Parallel_IN همزمان با لبه بالارونده کلاک در ثبات بارگذاری می‌شود و در صورت صفر بودن ثبات مقدار قبلی خود را نگه می‌دارد.
- LR ورودی دوبیتی برای مشخص کردن نوع انتقال چپ و یا راست ریاضی و منطقی است. ”LR=”11 ثبات مقدار ورودی را در خود ذخیره کند. ورودی انتقال به چپ ریاضی و منطقی یک حالت فرض شوند. در جدول ۱ حالت‌های مختلف انتقال برای ورودی LR آورده شده است.

LR (2 bits)	function
“00”	**Left shift (arithmetic and logic)**
“01”	**Right shift (logical)**
“10”	**Right shift (Arithmetic)**
“11”	**Load Parallel Input data**

نام entity و پورت‌های ورودی و خروجی ماژول‌های مورد نظر به صورت زیر باشد. تهیه testbench الزامی است.



شکل ۱) ماژول رجیستر 8 بیتی و تعریف پورت‌های ورودی و خروجی



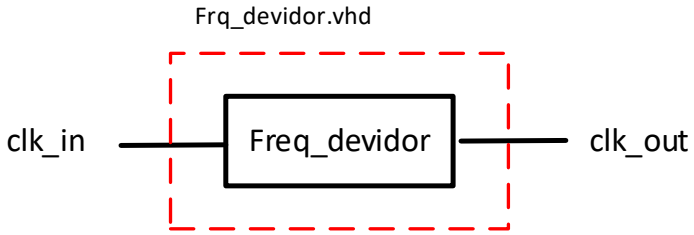
شکل ۲) بلوک دیاگرام رجیستر ۴ بیتی

برای مقداردهی خروجی در ورودی مالتی پلکسر می‌توان پورت‌های Q را از نوع Inout تعریف کرد و یا اینکه ابتدا خروجی‌های Dff را در سیگنال میانی ریخته و مقدار سیگنالهای میانی را در وردی مالتی پلکسر و پورت‌های خروجی که از نوع out هستند بریزید.

۲) چگونگی ایجاد مقدار تأخیر مشخص برای نمایش خروجی روی برد: (در حد شبیه سازی کافی است.)

برای ایجاد تأخیر و قابل رویت شدن تغییرات خروجی‌ها بر روی برد، از شمارنده برای تنظیم و کاهش فرکانس کلاک روی برد استفاده می‌شود. فرکانس کاری برد موجود در آزمایشگاه ۴۰ مگاهرتز می‌باشد، بنابراین برای دستیابی به تأخیر ۱ ثانیه، نیاز به شمارنده‌ای است که به میزان 4×10^{7}-1 بار بشمارد تا تأخیری به میزان ۱ ثانیه ایجاد گردد.

در این بخش از آزمایش شمارنده‌های طراحی و به کد بخش ۱ اضافه کنید تا بتوان خروجی‌های شیفت رجیستر را با تأخیرهای مناسب بر روی برد مشاهده نمود. در testbench فرکانس کلاک ورودی را در برابر با 40 مگاهرتز قرار دهید.



شکل ۳) ماژول frequency_devidor

F_{in} = 400 Hz , F_{out} : 1 Hz

scale = f_{in} / f_{out}

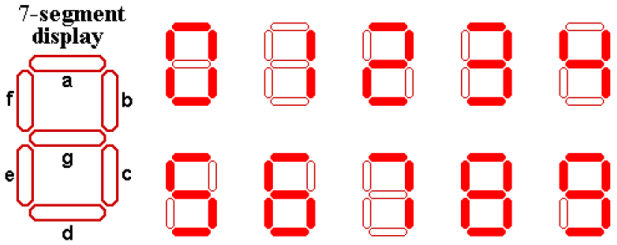
```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3
4   entity frq_devidor is
5   port (
6           clk_in : in std_logic;
7           clk_out : out std_logic
8   );
9   end entity frq_devidor;
10  architecture frq_devidor_arch of frq_devidor is
11  --- scale = fin / fout = Tout / Tin = 400hz/1Hz= 1s/2.50ms = 400
12  signal counter : integer range 1 to 400 := 1;
13  signal temp: STD_LOGIC:='0';
14  begin
15          freq_devidor : process (clk_in)
16          begin
17                  if rising_edge(clk_in) then
18                          if (counter = 200) then
19                                  temp <= not(temp);
20                                  counter <= 1;
21                          else
22                                  counter <= counter + 1;
23                          end if;
24                  end if;
25  end process freq_devidor;
26  clk_out <= temp;
27  end frq_devidor_arch;
```

شکل ۴) کد سخت افزاری تقسیم کننده فرکانس

نمایش محتوای رجیستر بر روی 7-segment

۳) در مرحله بعد باید ماژولی طراحی شود که ۴ بیت کم ارزش خروجی رجیستر را به صورت نمایش segment Seven کدگذاری کند و علاوه بر آن فرکانس ساعت را نیز کم کند. برای اینکار باید یک شمارنده طراحی شود.

نمایشگر هفت قسمتی، از هفت قطعه فتوالکتریک، مانند کریستال مایع یا LED ساخته می‌شود. این قطعات به گونه‌ای که در شکل ۳ مشاهده می‌شود در کنار هم قرار داده شده‌اند. با روشن کردن برخی از این LED ها می‌توان اعداد و همچنین برخی از کاراکترها را نمایش داد.

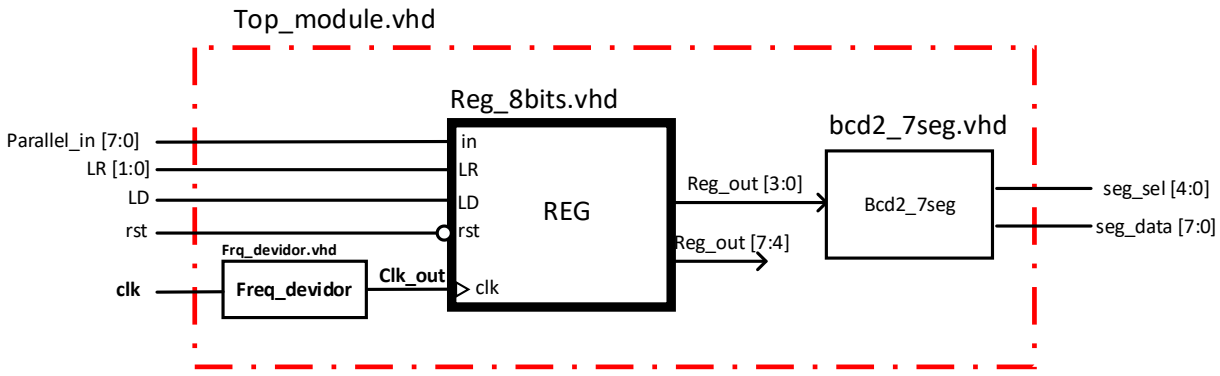


شکل ۴) چگونگی نمایش اعداد

Binary Inputs				Decoder Outputs								7-Segment Display Outputs
D	C	B	A	.	g	f	e	d	c	b	a	
0	0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	**1**	0	0	0	0	0	1	1	0	**1**
0	0	**1**	0	0	1	0	1	1	0	1	1	**2**
0	0	1	1	0	1	0	0	1	1	1	1	3
0	1	0	0	0	1	1	0	0	1	1	0	**4**
0	1	0	1	0	1	1	0	1	1	0	1	5
0	1	1	0	0	1	1	1	1	1	0	1	6
0	1	1	1	0	0	0	0	0	1	1	1	7
1	0	0	0	0	1	1	1	1	1	1	1	**8**
1	0	0	1	0	1	1	0	1	1	1	1	9

شکل ۵) نگاشت مقادیر BCD به Seven-segment

```
begin
with reg_out select
        -- 8'b".gfedcba"
    seg_data <= "00111111" when "0000",  -- 0
            "00000110" when "0001",  -- 1
            "01011011" when "0010",  -- 2
            "01001111" when "0011",  -- 3
            "01100110" when "0100",  -- 4
            "01101101" when "0101",  -- 5
            "01111101" when "0110",  -- 6
            "00000111" when "0111",  -- 7
            "01111111" when "1000",  -- 8
            "01101111" when "1001",  -- 9
            "00111000" when others; -- L letter
seg_sel <= "00001";
```



شکل ۶) بلوک دیاگرام مبدل bcd به کدsegment seven برای نمایش داده ذخیره شده در register