به نام خدا

طراحی کامپیوتری سیستم‌های دیجیتال

پاییز ۱۴۰۴

پروژه 3: **سنتز تولیدکننده هش روی ماژول‌های پایه**

طراحان: هاتف رضائی - سپهر جمالی

هدف پروژه :

در این تمرین می خواهیم با نحوه ی سنتز شدن کد وریلاگ روی FPGA آشنا شویم به این منظور؛ باید مسئله ی ساده تر شده ی تمرین یک را به صورت دستی توسط ماژول های ارائه شده پیاده سازی کنید.

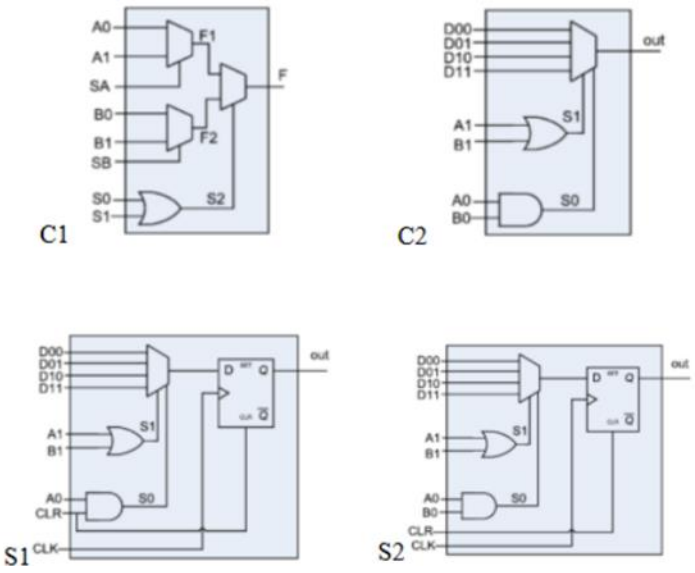
صورت مسئله:

مسئله ی این تمرین ساده شده ی تمرین اول (تولید کننده‌ی هش) است. برای ساده سازی عرض داده ها به 4 کلمه ی 8 بیتی کاهش یافته و همچنین به جای استفاده از ماژول rotate از یک ضرب کننده استفاده می شود که نیمه اول داده را در نیمه دوم آن ضرب می کند.

آشنایی با ماژول های پایه:

ماژول های منطقی پایه که در شکل 1 نشان داده شده‌اند ماژول های منطقی قابل برنامه‌ریزی هستند. با استفاده از آن‌ها می توان مدار های combinational و sequential های مختلفی را پیاده سازی کرد.

هدف این تمرین، پیاده سازی و سنتز طراحی بر روی بخشهای قابل برنامه ریزی یک FPGA هستند. بنابراین ضروری است تا مسیرهای داده و کنترل را ابتدا به صورت ساختاری یا همان Structural طراحی کنید و سپس طرح را بر روی سلول های منطقی قابل برنامه ریزی معرفی شده، سنتز کنید. سلولهای منطقی در دسترس در شکل۱- نمایش داده شده‌اند. دقت کنید که برای طراحی مسیر داده و کنترلر (FSM) شما تنها مجاز به استفاده از سلول های c1، c2، s1، s2 هستید.



شکل1. سلول های قابل برنامه ریزی مربوط به ماژول پایه

برای این تمرین ابتدا کد ماژول ها را که در کنار این فایل آپلود شده است دریافت و بررسی کنید؛ سپس بقیه ماژولهای لازم را که در پروژه ی یکم طراحی کردید با اتصال این ماژول ها پیاده سازی کنید.

نکته1: لازم به ذکر است که مجاز به استفاده از عمل‌گر های منطقی و گیت های پایهٔ زبان وریلاگ، در هیچ یک از ماژول هایی که خودتان می‌نویسید نیستید و تنها می‌توانید از ماژول هایی که به شما داده شده‌است استفاده‌کنید. درصورت هرگونه استفاده از عملگرهای منطقی، نمره ماژول و ماژول هایی که از آن استفاده می کنند را از دست خواهید داد. نیازی به سنتز کردن ROM های ورودی و یا خروجی نیست.

تغییرات نسبت به تمرین اول:

- عرض داده ها از 32 بیت به 8 بیت کاهش یافته است.
- مقادیر اولیه مطابق زیر تغییر کرده اند. (هشت بیت کم ارزش اعداد استفاده شده است.)

A0 = 00000001

B0 = 10001001

C0 = 11111110

D0 = 01110110

- فایل ضرایب جدید که با صورت پروژه روی سایت قرار داده شده است جایگزین شود.
- تعداد پیام های ورودی از 128 به 64 کاهش یافته است.
- به جای ماژول rotate که بخش steps و شیفت چپ در آن قرار داشت یک ضرب کننده قرار دهید که ورودی (خروجی جمع کننده) را دریافت کرده و 4 بیت کم ارزش را در 4 بیت پرارزش ضرب میکند و خروجی ضرب کننده به عنوان خروجی ماژول قرار میگیرد. (ضرب کننده با دو ورودی 4 بیتی و خروجی 8 بیتی)

```
Initialize:
    a0 := 0x01
    b0 := 0x89
    c0 := 0xfe
    d0 := 0x76

Break 32-bit message into four 8-bit words M[0..3]

A := a0
B := b0
C := c0
D := d0

for i from 0 to 63 do
    if 0 ≤ i ≤ 15 then
        F := (B and C) or ((not B) and D)
    else if 16 ≤ i ≤ 31 then
        F := (D and B) or ((not D) and C)
    else if 32 ≤ i ≤ 47 then
        F := B xor C xor D
    else if 48 ≤ i ≤ 63 then
        F := C xor (B or (not D))

    x := i
    repeat 6 times:
        fb := (bit5 of x) xor (bit3 of x) xor (bit1 of x)
        x := (x << 1) or fb
    rnd := (bit5 of x shifted left by 1) OR (bit4 of x)

    F := F + A + constant[i] + M[rnd]

    A := D
    D := C
    C := B
    B := B + F[width-1:width/2] * F[width/2-1:0]
end for

digest := A || B || C || D
```

شکل2. خط 35 بخش تغییریافته الگوریتم

مواردی که در حین پیاده سازی باید در نظر بگیرید:

نحوه گرفتن ورودیها و خروجی دادن برنامه، باید دقیقا مطابق با پروژه اول درس باشد.

اکیدا توصیه می شود طراحی به صورت سلسله مراتبی انجام شود. یعنی از ماژول های c1، c2، s1، s2 استفاده و با آنها سایر ماژول ها را به صورت سلسله مراتبی پیاده سازی و استفاده کنید.

نکته2: به صورت آزاد از ماژول ها استفاده نکنید! اگر تعداد گیت هایی (جمع گیت ها به ازای هر ماژول) که برای پیاده سازی استفاده کرده‌اید از 4800 تا بیشتر شوند به ازای هر درصد ماژول اضافه تر یک درصد از نمره ی کل را از دست خواهید داد.

تا جای ممکن کنترلر خود را ساده تعریف کنید تا در ساده کردن جدول کارنوی مربوطه و سپس پیاده سازی آن با ماژولهای تعریف شده به مشکل نخورید. (راهنمایی: می‌توانید از ایده ی one hot استفاده نمایید)

مواردی که باید تحویل دهید:

- طراحی کنترلر و مسیر داده به صورت دستی (برای همه ی ماژول ها)
- کدهای مربوط به زبان وریلاگ (تمام ماژول ها)
- فایل number.txt
- گزارش یا test bench جدا از ماژول های مختلفی که استفاده کرده‌اید.

امتیازی:

- اگر ماژول های استفاده شده از 3800 تا کمتر باشد 5% نمره ی امتیازی می‌گیرید.
- به سه گروهی که کمترین تعداد گیت را دارند (به شرط آنکه از 3800 تا کمتر گیت استفاده کرده باشند و عملکرد ماژول کاملا درست باشد) به ترتیب 20،15،10 نمره ی امتیازی تعلق می گیرد.

روش شمردن تعداد ماژول ها:

کد های **cpp** داده شده را کامپایل کرده و فایل اجرایی (exe.) را در کنار فایل های پروژه بگذارید . لازم به ذکر است اسم فایل های اجرایی باید هم اسم ماژول ها باشد تا دستور $system آن را اجرا کند (اگر اسم فایل خروجی را تغییر دادید باید در دستور سیستم هم تغییر ایجاد کنید). قبل از اجرای simulation فایل number.txt را نیز در کنار پروژه قرار داده و در آن یک 0 بنویسید تا با اجرا شدن برنامه عدد آن به ازای هر instantiate درون فایل به مقدار تعداد گیت های آن زیاد شود و در پایان simulation تعداد گیت های شما درون فایل نوشته شود (زمان سیمولیشن کمی طولانی تر می‌شود).

تعداد گیت های هر ماژول:

- S2 = 13
- S1 = 13
- C2 = 11
- C1 = 10

دستور کامپایل کد های cpp:

```
g++ c1.cpp -o c1.exe
g++ c2.cpp -o c2.exe
g++ s2-s1.cpp -o s2-s1.exe
```

سایر نکات

- - انجام این تمرین به صورت گروه های دونفره خواهد بود:
- فایل ها و گزارش خود را تا قبل از موعد تحویل، با نام zip.<CAD_HW3_<SID در محل مربوطه در صفحه درس آپلود کنید.
- در صورت هرگونه ابهام می توانید از طریق ایمیل با طراحان در ارتباط باشید.
- نام گذاری صحیح متغیرها، تمیزی کد و توضیحات و پارامتری بودن ورودی‌های ماژول‌ها می تواند تا حدودی کاستی‌های کد را در بخش‌های دیگر جبران کند.
- - هدف این تمرین یادگیری شماست! در صورت کشف تقلب، مطابق با قوانین درس برخورد خواهد شد.

موفق باشید