



به نام خدا



## طراحی کامپیوتروی سیستم‌های دیجیتال

پاییز ۱۴۰۴

### پروژه ۳: سنتز تولیدکننده هش روی مازول‌های پایه

طراحان: هاتف رضائی - سپهر جمالی

#### هدف پروژه :

در این تمرین می‌خواهیم با نحوه‌ی سنتز شدن کد وریلگ ر روی FPGA آشنا شویم به این منظور؛ باید مسئله‌ی ساده‌تر شده‌ی تمرین یک را به صورت دستی توسط مازول‌های ارائه شده پیاده‌سازی کنید.

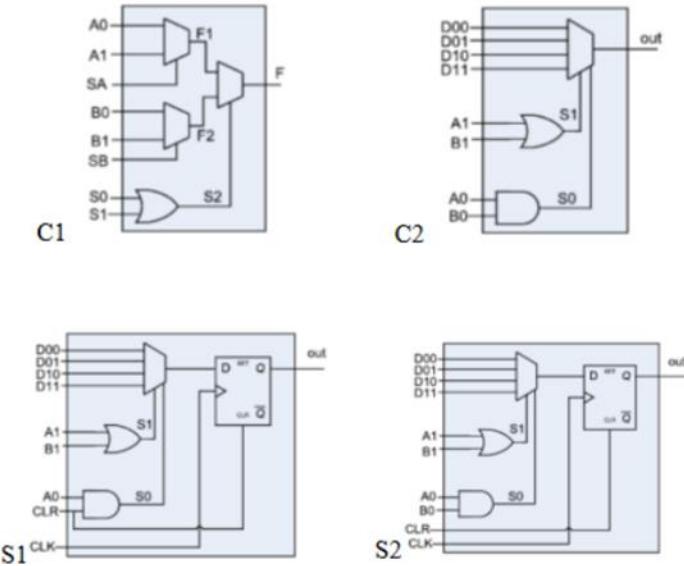
#### صورت مسئله:

مسئله‌ی این تمرین ساده‌شده‌ی تمرین اول (تولید کننده‌ی هش) است. برای ساده‌سازی عرض داده‌ها به 4 کلمه‌ی 8 بیتی کاهش یافته و همچنین به جای استفاده از مازول rotate از یک ضرب کننده استفاده می‌شود که نیمه اول داده را در نیمه دوم آن ضرب می‌کند.

#### آشنایی با مازول‌های پایه:

مازول‌های منطقی پایه که در شکل 1 نشان داده شده‌اند مازول‌های منطقی قابل برنامه‌ریزی هستند. با استفاده از آن‌ها می‌توان مدار‌های sequential و combinational را مختلفی را پیاده‌سازی کرد.

هدف این تمرین، پیاده‌سازی و سنتز طراحی بر روی بخش‌های قابل برنامه‌ریزی 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 سلول‌های منطقی قابل برنامه‌ریزی معرفی شده، سنتز کنید. سلول‌های منطقی در دسترس در شکل ۱-نمایش داده شده‌اند. دقت کنید که برای طراحی مسیر داده و کنترل (FSM) شما تنها مجاز به استفاده از سلول‌های c1، s1، c2، s2 هستید.



شکل 1. سلول های قابل برنامه ریزی مربوط به مازول پایه

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

**نکته 1:** لازم به ذکر است که مجاز به استفاده از عملگرهای منطقی و گیت‌های پایه زبان وریلگ، در هیچ یک از مازول هایی که خودتان می‌نویسید نیستید و تنها می‌توانید از مازول هایی که به شما داده شده است استفاده کنید. در صورت هرگونه استفاده از عملگرهای منطقی، نمره مازول و مازول هایی که از آن استفاده می‌کنند 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.

- به جای مازول 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 پرازش ضرب می‌کند و خروجی ضرب کننده به عنوان خروجی مازول قرار می‌گیرد. (ضرب کننده با دو ورودی 4 بیتی و خروجی 8 بیتی)

```

1 Initialize:
2     a0 := 0x01
3     b0 := 0x89
4     c0 := 0xfe
5     d0 := 0x76
6
7 Break 32-bit message into four 8-bit words M[0..3]
8
9 A := a0
10 B := b0
11 C := c0
12 D := d0
13
14 for i from 0 to 63 do
15     if 0 ≤ i ≤ 15 then
16         F := (B and C) or ((not B) and D)
17     else if 16 ≤ i ≤ 31 then
18         F := (D and B) or ((not D) and C)
19     else if 32 ≤ i ≤ 47 then
20         F := B xor C xor D
21     else if 48 ≤ i ≤ 63 then
22         F := C xor (B or (not D))
23
24     x := i
25     repeat 6 times:
26         fb := (bit5 of x) xor (bit3 of x) xor (bit1 of x)
27         x := (x << 1) or fb
28         rnd := (bit5 of x shifted left by 1) OR (bit4 of x)
29
30     F := F + A + constant[i] + M[rnd]
31
32     A := D
33     D := C
34     C := B
35     B := B + F[width-1:width/2] * F[width/2-1:0]
36 end for
37
38 digest := A || B || C || D

```

شکل 2. خط 35 بخش تغییریافته الگوریتم

**مواردی که در حین پیاده سازی باید در نظر بگیرید:**

نحوه گرفتن ورودیها و خروجی دادن برنامه، باید دقیقاً مطابق با پروژه اول درس باشد.  
اکیدا توصیه می شود طراحی به صورت سلسله مراتبی انجام شود. یعنی از مازول های  $s_1, s_2, c_1, c_2, c_3$  استفاده و با آنها سایر مازول ها را به صورت سلسله مراتبی پیاده سازی و استفاده کنید.

**نکته 2:** به صورت آزاد از مازول ها استفاده نکنید! اگر تعداد گیت های (جمع گیت ها به ازای هر مازول) که برای پیاده سازی استفاده کردہ اید از 4800 تا بیشتر شوند به ازای هر درصد مازول 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 مازولهای تعریف شده به مشکل نخوردید. (راهنمایی: می‌توانید از ایده ی one hot استفاده نمایید)

مواردی که باید تحويل دهید:

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

امتیازی:

- اگر مازول 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 مازول کاملا درست باشد) به ترتیب 10, 15, 20 نمره ی امتیازی تعلق می‌گیرد.

روش شمردن تعداد مازول 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 مازول 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 مازول:

- 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 تحويل، با نام zip.<SID><HW3\_CAD> 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 ورودی های ماژول ها می تواند تا حدودی کاستی های کد را در بخش های دیگر جبران کند.
- هدف این تمرین یادگیری شماست! در صورت کشف تقلب، مطابق با قوانین درس برخورد خواهد شد.

موفق باشید