به نام خدا

طراحی کامپیوتری سیستم‌های دیجیتال - پاییز ۱۴۰۴

پروژه ۱: طراحی Hash Generator RTL

طراحان: امیرحسین کریلی – کیمیا فیروزبخش – هاتف رضایی

هدف پروژه:

این پروژه با پیاده‌سازی سخت‌افزاری یک Hash Generator در سطح RTL، به مرور و تقویت مفاهیم طراحی Datapath و Controller به شیوه Huffman، به‌کارگیری مکانیزم‌های Handshaking، و اجرای فرایند شبیه‌سازی و تأیید عملکرد سیستم می‌پردازد.

صورت مسئله:

هش، مقداری با طول ثابت است که از داده‌های ورودی با طول متغیر تولید می‌شود و به‌گونه‌ای طراحی شده که کوچک‌ترین تغییر در ورودی منجر به تغییر قابل توجه در مقدار هش گردد. این ویژگی باعث می‌شود که هش به عنوان امضای دیجیتال داده‌ها عمل کرده و در تشخیص تغییرات ناخواسته یا دستکاری داده‌ها بسیار موثر باشد. علاوه بر این، هش به گونه‌ای طراحی شده که بازیابی داده اصلی از روی مقدار هش امکان‌پذیر نباشد و احتمال وجود دو داده متفاوت با هش یکسان به حداقل برسد.

در این پروژه، پیام ۱۲۸-بیتی ورودی به چهار کلمه ۳۲-بیتی تقسیم شده و پس از اجرای الگوریتمی ۶۴ مرحله‌ای، خروجی هش ۱۲۸-بیتی تولید می‌شود.

طراحی و پیاده‌سازی:

طراحی شامل چهار بخش اصلی است:

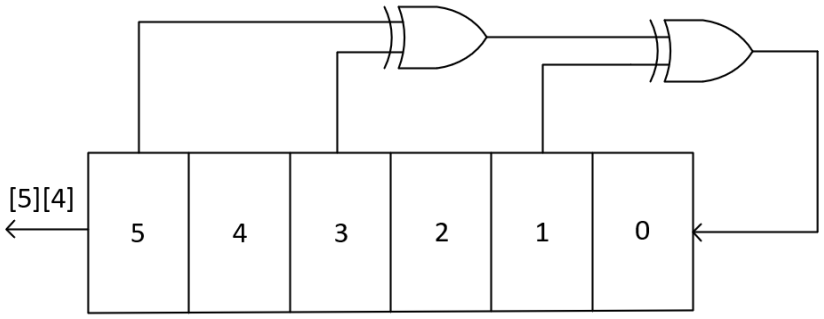
- بخش تولید عدد شبه‌تصادفی
- بخش تولید هش
- ارتباط و هماهنگی بین این دو بخش
- درستی‌سنجی

۱. بخش تولید عدد شبه‌تصادفی

این بخش مسئول تولید اعداد شبه‌تصادفی است که به عنوان شاخص برای انتخاب کلمات خاصی از پیام به کار می‌روند. الگوریتم تولید عدد تصادفی مبتنی بر یک شیفت‌رجیستر با عملیات بیت به بیت مثل XOR روی بیت‌های مشخص است. این ماژول در هر سیکل کلاک با توجه به بیت‌های خاصی که به صورت XOR ترکیب شده‌اند، مقدار جدیدی تولید می‌کند که به ظاهر تصادفی است اما از الگوریتمی کاملاً مشخص پیروی می‌کند.

شما باید ماژولی طراحی کنید که پس از مشاهده یک پالس کامل روی سیگنال start_rnd، عدد ٦-بیتی روی باس ورودی خود را در رجیستر داخلی ذخیره می‌کند و مطابق شکل۱ آن را شیفت می‌دهد. پس از شش بار شیفت دادن، دو بیت پرارزش را با همان ترتیب روی باس خروجی ۲-بیتی خود قرار می‌دهد و سیگنال done_rnd را به مدت یک سیکل برابر با ۱ می‌کند.

در این قسمت بایستی برای این ماژول مسیر داده و کنترل‌کننده طراحی کرده و آن را درستی‌سنجی کنید. به عنوان مثال، این ماژول به ازای ورودی ۰۰۰۱۱۱، باید خروجی ۱۰ را تولید کند.



شکل۱. رجیستر داخلی و مکانیزم شیفت دادن

۲. بخش تولید هش

پیام ورودی به صورت یک بلوک ۱۲۸-بیتی تعریف شده است که شامل چهار کلمه ۳۲-بیتی است. در هر مرحله، مقدار شاخص تولید شده توسط بخش شبه‌تصادفی، یک کلمه مشخص از پیام را انتخاب می‌کند. این کلمه انتخاب‌شده با مقدار فعلی هش ترکیب می‌شود.

شما باید ماژولی طراحی کنید که پس از مشاهده یک پالس کامل روی سیگنال start، پیام ۱۲۸-بیتی روی باس ورودی خود را در یک رجیسترفایل داخلی ذخیره کند و مطابق شکل۲ آن را در ٦٤ تکرار پردازش کند. در انتها، مقادیر چهار رجیستر ۳۲-بیتی A, B, C, D را به طوری که A در جایگاه پرارزش قرار دارد، روی باس خروجی ۱۲۸-بیتی خود قرار می‌دهد و سیگنال done را به مدت یک سیکل برابر با ۱ می‌کند.

در این قسمت بایستی برای این ماژول مسیر داده و کنترل‌کننده طراحی کرده و آن را درستی‌سنجی کنید. به عنوان مثال، این ماژول به ازای ورودی [41a801a8e81df62b14a661b85c97bf45]، باید خروجی [c6f6c75d2bbbb9a586cf3291347acdce] را تولید کند.

```
Initialize:
    a0 := 0x67452301
    b0 := 0xefcdab89
    c0 := 0x98badcfe
    d0 := 0x10325476

Break 128-bit message into four 32-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 + leftrotate(F, step[i])
end for

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

شکل۲. الگوریتم کلی تولید هش به همراه قسمت تصادفی

راهنمایی: مقادیر ثابت constant در فایل حافظه constant.mem قرار دارد. باید با استفاده از ROM از این مقادیر استفاده کنید، اما مقادیر ثابت steps را در ماژول rotate تعریف و استفاده کنید:

```
// step specifies the per-round shift amounts
step[ 0..15] := { 7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22 }
step[16..31] := { 5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20 }
step[32..47] := { 4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23 }
step[48..63] := { 6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21 }
```

شکل۳. مقادیر ثابت s به ازای تکرارهای ۰ الی ۶۳

۳. ارتباط و هماهنگی بین بخش‌ها

بخش تصادفی و بخش هش باید به یکدیگر متصل شده و بخش هش باید با استفاده از سیگنال‌های start_rnd و done_rnd به بخش تصادفی فرمان دهد تا با توجه به شمارنده تکرار حلقه اصلی، یک عدد ۲-بیتی تصادفی برای انتخاب کلمه تولید کند.

٤. درستی‌سنجی

روند کار این برنامه برای پردازش یک پیام در فایل example.txt قرار دارد. می‌توانید از این فایل برای عیب‌یابی برنامه خود استفاده نمایید.

مواردی که در حین پیاده سازی باید در نظر بگیرید:

- هر دو بخش (تصادفی و هش) بایستی دارای مسیر داده و کنترل‌کننده به شیوه هافمن و ماژولار باشند.
- ارتباط این دو بخش باید در یک ماژول سطح بالا به نام Top به درستی پیاده‌سازی شود.
- ماشین حالت استفاده شده در کنترل‌کننده‌ها باید ماشین Moore باشد.
- هر یک از عملیات‌های مورد نیاز باید در یک ماژول مستقل گسترش داده شوند.
- داشتن ماژول‌های پیچیده یا تکراری مجاز نیست.
- ماژول Top نهایی باید پس از پردازش یک پیام به حالت اولیه خود بازگشته و آماده دریافت پیام بعدی و پردازش آن باشد.

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 CAD_HW1_<SID>.zip 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