دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

طراحی کامپیوتری سیستم‌های دیجیتال

بهار ۱۴۰۱

دستیاران آموزشی: سارا رضائی منش، شایان شبیهی

مقدمه

در این تمرین از شما خواسته می‌شود در فاز اول یک کنترلر و مسیر داده برای مدار خواسته شده، **روی کاغذ** طراحی کنید. در فاز بعدی این تمرین را با استفاده از زبان توصیف سخت‌افزاری Verilog پیاده‌سازی خواهید کرد. مهلت تحویل این تمرین در مجموع دو هفته درنظر گرفته شده‌است. در انتهای هفته‌ی اول بایستی فاز اول و در انتهای هفته‌ی دوم، فاز دوم را آپلود کنید.

توجه: انجام این تمرین به صورت گروه‌های دونفره خواهد بود.

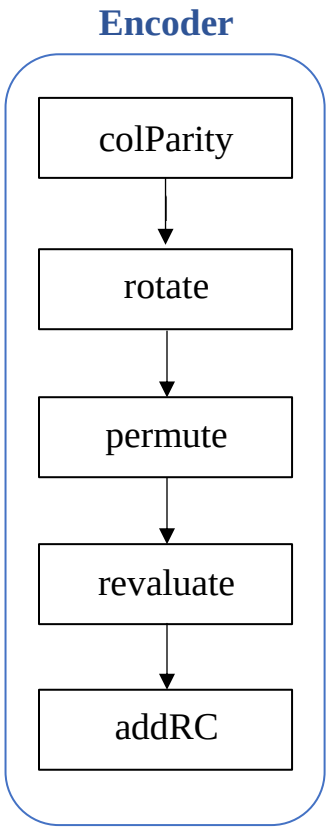
توضیحات پروژه

هدف از این تمرین طراحی تابعی به نام encoder است. برای طراحی encoder لازم است چند تابع جدید به تابع permutor که در تمرین اول طراحی کردید اضافه کنیم. تابع encoder یک ورودی ۱۶۰۰ بیتی دریافت می کند و در نهایت یک خروجی رمزگذاری شده بازمی‌گرداند. عملیات encode در این تابع خود متشکل از چند تابع است به طوریکه خروجی هر تابع، ورودی تابع بعدی خواهد بود. جزئیات این توابع در ادامه توضیح داده شده است.

تابع encoder

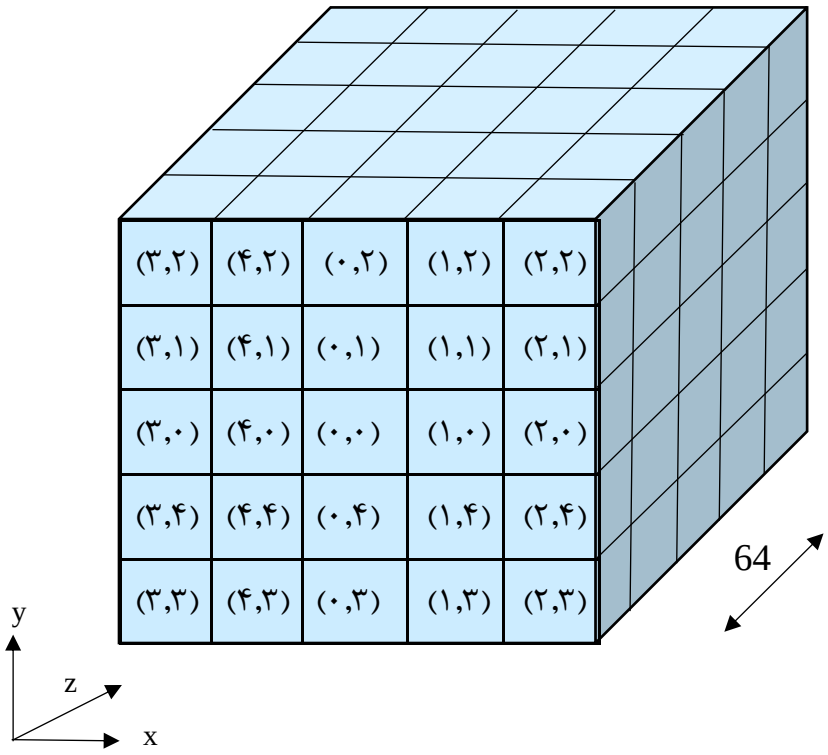
تابع encoder عملیات رمزگذاری را با استفاده از پنج تابع انجام می دهد که تابع سوم را در پروژه قبل پیاده سازی کرده‌اید.

این تابع یک ورودی ۱۶۰۰ بیتی دریافت کرده و ۵ تابع نمایش داده شده در شکل زیر را به ورودی اعمال می‌کند این عملیات ۲۴ بار تکرار خواهد شد و در نهایت خروجی تکرار ۲۴ام به عنوان متن رمز نگاری شده در نظر گرفته می‌شود. نمای کلی توابع به صورت زیر است:



توضیحات توابع

تمام توابع توضیح داده شده در ادامه، مانند پروژه قبل، بر روی یک ماتریس ۵ در ۵ در ۶۵ به صورت زیر اعمال می شوند:

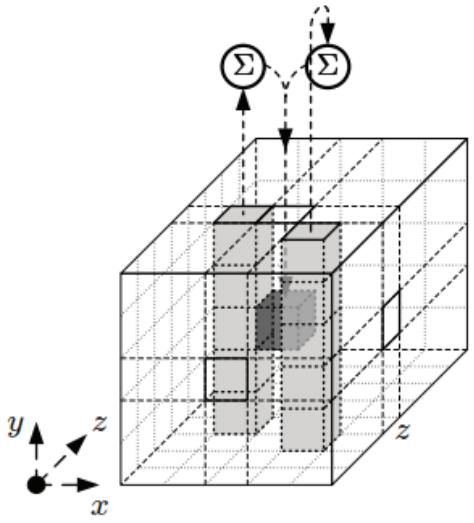


۱. تابع colParity

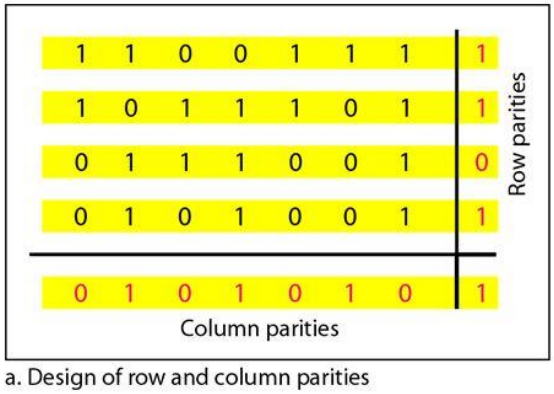
این تابع برای هر بیت در ماتریس A به صورت زیر تعریف می‌شود:

$$A[i][j][k] = A[i][j][k] \oplus parity(A[0\ldots4][j+1][k-1]) \oplus \text{parity(A[0…4][j + 1][k − 1]}$$

این تابع به ازای هر بیت در ماتریس، parity دو ستون مشخص شده در فرمول بالا را محاسبه کرده و عملیات xor را بین نتایج بدست آمده و خانه انتخاب شده انجام می‌دهد. شکل زیر عملیات colParity را برای یکی از خانه های ماتریس نشان می‌دهد:



parity یک مجموعه بیت مشخص می کند که تعداد یک ها در این مجموعه زوج است یا فرد. در صورت فرد بودن، نتیجه این عملیات یک بیت ۱ و در غیر اینصورت نتیجه یک بیت ۰ است. چند مثال از اعمال parity بر اعداد دودویی در شکل زیر آمده است:



a. Design of row and column parities

راهنمایی: برای انجام عملیات parity می توانید از xor کردن تمام بیت های مجموعه استفاده کنید.

۲. تابع rotate

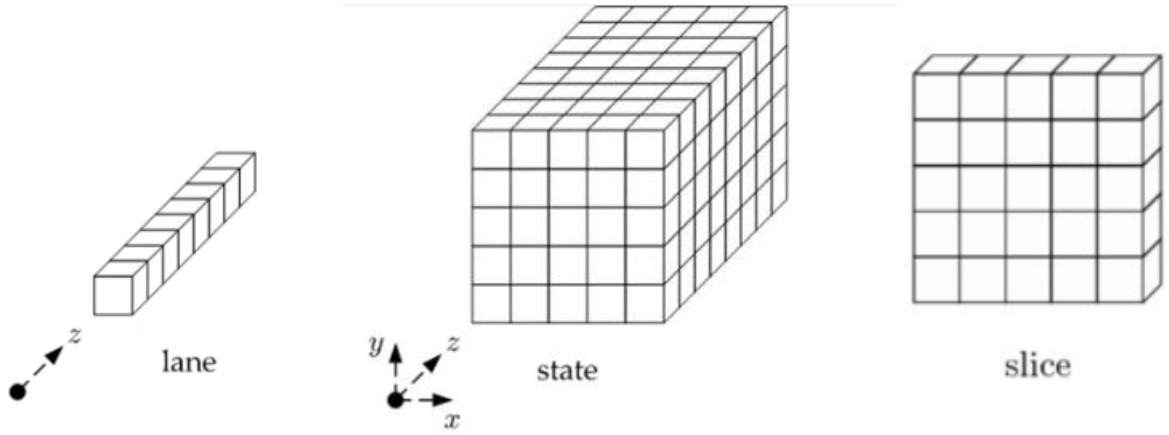
این تابع برای هر بیت در ماتریس A به صورت زیر تعریف می‌شود:

$$A[x,y,z] = A[x,y,z] \ if \ x = y = 0$$

$$A[x,y,z] = A[x,y,(z-(t+1)(t+2)/2) \ mod \ 64] \ otherwise$$

$$0 < t < 24 \text{ and } \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^t \begin{pmatrix} 1 \\ 0 \end{pmatrix} mod \ 5 = \begin{pmatrix} x \\ y \end{pmatrix}$$

قبل از توضیح دقیق تر عملکرد این تابع، لازم است به مفهوم lane و slice اشاره شود. در یک ماتریس، lane و slice همانند شکل زیر تعریف می‌شوند:



عملکرد تابع به شرح زیر است:

۱) Lane ای که در موقعیت (x, y) = 0 قرار دارد بی تغییر باقی می‌ماند و در lane های دیگر بیت ها با اعمال فرمول داده شده به موقعیت های دیگر در راستای محور z منتقل می‌شوند. مشخص است که بیت های متعلق به یک lane به یک میزان جا به جا می‌شوند.

۲) متغیر t برای تعیین میزان شیفت هر بیت و مقدار شیفتی که به هر lane اختصاص داده می‌شود مورد استفاده قرار می‌گیرد.

۳) ۲۴ شیفت انجام شده در این تابع توسط فرمول زیر محاسبه می‌شوند:

$$\frac{(t+1)(t+2)}{2} mod \ 64$$

۴) مقدار t بین ۰ تا ۲۴ قرار دارد و برای هر مقدار t، موقعیت متناظر آن در ماتریس به صورت زیر تعریف می‌شود:

$$\begin{pmatrix}0 & 1\\ 2 & 3\end{pmatrix}^{t}\begin{pmatrix}1\\ 0\end{pmatrix} mod\ 5 = \begin{pmatrix}x\\ y\end{pmatrix}$$

به عنوان مثال برای t = 3 داریم:

$$\begin{pmatrix}x\\ y\end{pmatrix} = \begin{pmatrix}0 & 1\\ 2 & 3\end{pmatrix}^{3}\begin{pmatrix}1\\ 0\end{pmatrix} mod\ 5$$

$$= \begin{pmatrix}0 & 1\\ 2 & 3\end{pmatrix}\begin{pmatrix}0 & 1\\ 2 & 3\end{pmatrix}\begin{pmatrix}0 & 1\\ 2 & 3\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} mod\ 5$$

$$= \begin{pmatrix}0 & 1\\ 2 & 3\end{pmatrix}\begin{pmatrix}0 & 1\\ 2 & 3\end{pmatrix}\begin{pmatrix}0\\ 2\end{pmatrix} mod\ 5$$

$$= \begin{pmatrix}0 & 1\\ 2 & 3\end{pmatrix}\begin{pmatrix}2\\ 6\end{pmatrix} mod\ 5 = \begin{pmatrix}0 & 1\\ 2 & 3\end{pmatrix}\begin{pmatrix}2\\ 1\end{pmatrix} mod\ 5$$

$$= \begin{pmatrix}1\\ 7\end{pmatrix} mod\ 5 = \begin{pmatrix}1\\ 2\end{pmatrix}$$

مقادیر جابه‌جایی بیت ها (مقداری که از z کم می شود) برای هر lane در جدول زیر آمده است و نیازی به محاسبه مجدد آن نیست. می توانید این مقادیر را به صورت مستقیم در طراحی خود استفاده کنید:

y/x	x = 0	x = 1	x = 2	x = 3	x = 4
y = 4	18	2	61	56	14
y = 3	41	45	15	21	8
y = 2	3	10	43	25	39
y = 1	36	44	6	55	20
y = 0	0	1	62	28	27

توجه: دقت کنید که داده ها به صورت اسلایس از حافظه خوانده می‌شوند.

۳. تابع permute

توضیحات این تابع به صورت کامل در تمرین کامپیوتری اول آورده شده است.

۴. تابع revaluate

این تابع تنها تابع غیر خطی encoder است. تابع revaluate برای هر بیت در ماتریس A به صورت زیر تعریف می‌شود:

$$A[x,y,z] = A[x,y,z] \oplus (NOT(A[x+1,y,z])\ AND\ A[x+2,y,z])$$

این تابع مقدار هر بیت در ماتریس را بر اساس مقدار همان بیت و مقدار دوبیت بعدی آن که در همان ردیف قرار دارند تغییر می‌دهد.

۵. تابع addRC

این تابع بر روی ماتریس A به صورت زیر تعریف می‌شود:

$$A[0,0] = A[0,0] \oplus RC[i_r]$$

مقدار $RC[i_r]$ به ازای هر بار اجرای پنج مرحله بالا متفاوت و در هر مرحله ثابت است. وظیفه تابع addRC این است که هر lane در موقعیت [0, 0] را با یک ثابت ترکیب کند.(bitwise xor)

همانطور که گفته شد، در این تمرین کامپیوتری، توابع ذکر شده ۲۴ بار روی ماتریس وروی اعمال می شوند پس ۲۴ مقدار متفاوت RC خواهیم داشت. این مقادیر به صورت hexadecimal در جدول زیر آورده شده‌اند:

```
RC[ 0] = 0x0000000000000001        RC[12] = 0x000000008000808B
RC[ 1] = 0x0000000000008082        RC[13] = 0x800000000000008B
RC[ 2] = 0x800000000000808A        RC[14] = 0x8000000000008089
RC[ 3] = 0x8000000080008000        RC[15] = 0x8000000000008003
RC[ 4] = 0x000000000000808B        RC[16] = 0x8000000000008002
RC[ 5] = 0x0000000080000001        RC[17] = 0x8000000000000080
RC[ 6] = 0x8000000080008081        RC[18] = 0x000000000000800A
RC[ 7] = 0x8000000000008009        RC[19] = 0x800000008000000A
RC[ 8] = 0x000000000000008A        RC[20] = 0x8000000080008081
RC[ 9] = 0x0000000000000088        RC[21] = 0x8000000000008080
RC[10] = 0x0000000080008009        RC[22] = 0x0000000080000001
RC[11] = 0x000000008000000A        RC[23] = 0x8000000080008008
```

همانطور که مشخص است، هر ثابت ۶۴ بیت دارد و هر lane هم طبق صورت پروژه شامل ۶۴ بیت است.

توجه: نکات مربوط به فایل های ورودی و خروجی همانند پروژه قبل هستند و مجدد در زیر ذکر شده اند.

فایل های ورودی:

نام این فایل‌ها به صورت "input_i.txt" بوده که در آن i شماره فایل است. تمامی فایل‌های ورودی واقع در فولدر "tests" هستند که در کنار صورت پروژه آپلود شده‌اند. هر فایل شامل ۶۴ خط می‌باشد که در هر خط ۲۵ درایه یک صفحه در محور x-z آورده شده است. ترتیب این صفحات از راستای $y = 0$ تا $y = 63$ است. یعنی در خط اول ۲۵ درایه جلوترین صفحه و در خط آخر، ۲۵ درایه عقب‌ترین صفحه آورده شده است.

ترتیب ۲۵ درایه واقع در هر خط، بدین صورت است که از خانه پایین چپ $(x, z) = (3,3)$ شروع کرده و با حرکت به سمت راست و بالا به خانه بالا راست $(x, z) = (2,2)$ می‌رسیم. مسیر خواندن خانه این درایه‌ها در چند مرحله اول به شکل زیر است:

$(3,3) \Rightarrow (4,3) \Rightarrow (0,3) \Rightarrow (1,3) \Rightarrow (2,3) \Rightarrow (3,4) \Rightarrow (4,4)$ ...

نکته مهم: برای خواندن ورودی تنها مجاز به استفاده از یک رجیستر ۲۵ بیتی هستید.

فایل های خروجی:

نام این فایل ها را به صورت "output_i.txt" در نظر بگیرید که در آن i شماره فایل است. تمامی فایل‌های خروجی را در همان فولدر "tests" بسازید و ذخیره کنید. درایه‌ها را به همان ترتیبی که خوانده‌اید (پس از جایگشت دادن)، در فایل خروجی بنویسید.

مواردی که در حین پیاده سازی باید در نظر بگیرید (فاز یک):

- طراحی شما در فاز اول باید کاملا قابلیت پیاده سازی در Verilog را داشته باشد. لازم است در فاز دوم تمرین حتما همان مدار فاز اول را در وریلاگ پیاده‌سازی کرده و تحویل دهید.

مواردی که باید در فاز اول تحویل دهید:

- گزارش کار شامل طراحی کنترلر (FSM) و مسیرداده بر روی کاغذ.

مواردی که در حین پیاده سازی باید در نظر بگیرید (فاز دو):

- عملیات مطرح شده باید تا جایی که امکان دارد به صورت همروند بر روی ماتریس ورودی اعمال گردند.
- در صورتی که متوجه شدید طراحی شما در فاز اول مشکل داشته است، آن را در این مرحله اصلاح کنید و مشکل و راه‌حل را در گزارش به صورت کامل توضیح دهید.
- بخشی زیادی از نمره نهایی شما، مربوط به اجرای درست برنامه می‌شود. بدین منظور با بررسی تست کیس ها مختلف و متنوع از اجرای درست برنامه مطمئن شوید.
- این پروژه تحویل حضوری دارد و برنامه شما با تست کیس های جدید بررسی خواهد شد.

مواردی که باید در فاز دوم تحویل دهید:

- تمامی فایل‌های لازم برای اجرای پروژه (فایل‌های hdl، تست بنچ و...).
- خروجی‌های تست کیس‌ها مطابق روشی که ذکر شد.
- گزارش کار (مسیر داده و طراحی کنترلر، اشکالات فاز اول و نتایج خروجی).

نکات پایانی

- در فاز اول این تمرین باید مدار را با جزئیات کامل روی کاغذ طراحی کنید و در فاز دوم آن را به کمک زبان توصیف سخت افزاری Verilog پیاده سازی کرده و کارکرد صحیح آن را بررسی کنید.
- برای فاز دوم تمرین، لازم است فایل های HDL و testbench خود را مطابق ساختار توضیح داده شده در trunk/doc در subdirectory های trunk آپلود کنید. همچنین، اطمینان حاصل کنید که با اجرای trunk/sim/sim_top.tcl تست بنچ شما اجرا می‌شود. برای اجرای این اسکریپت میتوانید از دستور زیر در Modelsim استفاده کنید:

>> do <sim_file>

- لازم است فرمت خروجی مدار شما دقیقا مطابق ساختار مطرح شده برای ورودی باشد. توجه کنید که صحت کارکرد مدار شما با تست های آماده بررسی خواهد شد.
- فایل ها و گزارش خود را تا قبل از موعد تحویل هر فاز، با نام‌های CAD_HW2_P1_<SID>.zip و CAD_HW2_P2_<SID>.zip به ترتیب در محل های مربوطه برای فاز اول و دوم در صفحه درس آپلود کنید.
- هدف از این تمرین، یادگیری شماست! در صورت کشف تقلب، مطابق با قوانین درس برخورد خواهد شد.
- در صورت داشتن هرگونه سوال یا ابهام از طریق ایمیل های زیر با دستیاران آموزشی در ارتباط باشید.

sara.rezaeimanesh2000@gmail.com
shabihish@gmail.com

موفق باشید