بسمه تعالی

درس مدارهای منطقی / سیستم‌های دیجیتال ۱

تکلیف کامپیوتری ۳

پیاده‌سازی واحد تولیدکننده عدد شبه‌تصادفی

دانشکدگان فنی دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر
دکتر بیژن علیزاده
نیم‌سال اول سال تحصیلی ۰۴-۱۴۰۳
دستیار آموزشی: محمد حسین نیکخواه قمی، سید صدرا قوامی
mhr.nikkhah@gmail.com , sadra.ghavami@gmail.com

مقدمه

هدف از این تمرین، آشنایی با حافظه‌های D-Latch، D-Flip-Flop و رجیسترها و آشنایی با کاربردهای این نوع حافظه‌ها می‌باشد. در ابتدا، با پیاده‌سازی D-Latch، ساختار پایه D-Flip-Flop ساخته خواهدشد. در ادامه، به کمک DFF، ماژول Shift-Register را پیاده‌سازی می‌شود. در نهایت، با استفاده از این ماژول، ساختاری برای تولید اعداد به صورت تصادفی ارائه می‌شود.

تولیدکننده اعداد شبه‌تصادفی (PRNG)[1] به نوعی از الگوریتم‌ها گفته می‌شود که دنباله‌ای از اعداد را تولید می‌کنند که به نظر تصادفی هستند، اما در واقع قطعی بوده و به کمک معادلات ریاضی قابل بیان هستند. این دنباله عددی وابسته به مقدار اولیه‌ای به نام Seed است. به ازای هر Seed، یک دنباله یکتا تولید شده و با تغییر آن، دنباله تغییر می‌کند. این الگوریتم‌ها به خصوص در رمزنگاری حائز اهمیت هستند و در بسیاری از سیستم‌ها به صورت سخت‌افزاری پیاده‌سازی می‌شوند.

یک روش ساختن چنین الگوریتمی به کمک واحدهای سخت‌افزاری، LFSR[2] است. این سخت‌افزار شامل یک شیفت رجیستر است که در ابتدا حاوی مقدار Seed می‌باشد. ورودی سریال این شیفت رجیستر، تابع خطی از بیت‌های شیفت رجیستر در هر لحظه است. معروف‌ترین عملیات برای تابع خطی، عملیات XOR است. این روش از جهت پیاده‌سازی بسیار ساده بوده و از سرعت بالایی برخوردار است. اما این سادگی باعث می‌شود از نظر امنیتی قدرتمند نباشد، به‌طوری‌که با داشتن بخش نسبتا کمی از این دنباله، می‌توان به دنباله دست یافت. به همین

[1] Pseudo Random Number Generator (PRNG)
[2] Linear Feedback Shift Register

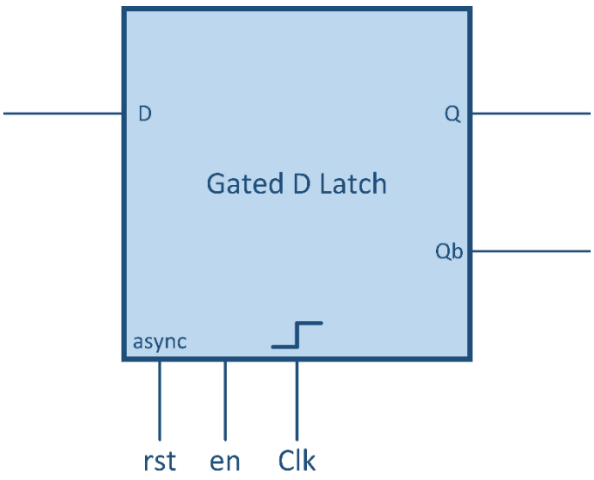
جهت، نوع پیچیده‌تری از این واحدها به نام NFSR[3] ارائه شده‌است. واحد NFSR مشابه LFSR است با این تفاوت که ورودی سریالی شیفت رجیستر، تابع غیرخطی از بیت‌های آن است. این ماژول‌ها، مقاومت بیشتری در مقابل حملات رمزگاری نسبت به LFSR از خود نشان داده‌اند. این در حالی است که طراحی NFSRهای بزرگ با رعایت کردن محدودیت‌های زمانی و تاخیر کم، مشکل‌ساز است. به همین جهت، طراحی یک PRNG به صورتی که از هر دو این روش‌ها بهره بگیرد، بسیار مفید خواهد بود. در این تمرین، نوعی از PRNG به نام Grain[4] پیاده‌سازی می‌شود، که ترکیبی از دو تولیدکننده اعداد تصادفی LFSR و NFSR است.

روند پیاده‌سازی

D-Flip-Flop

۱. ابتدا ماژول D-Latch را مشابه شکل ۱ در Verilog پیاده‌سازی کنید. پیاده‌سازی شما باید درسطح گیت باشد و شکل مدار را نیز در گزارش خود بیاورید.

۲. طراحی انجام‌شده را با نوشتن یک تست‌بنچ درستی‌سنجی کنید. درحالتی که Clock در سطح ۱ و صفر قرار دارد، ورودی‌ها را تغییر داده و نتیجه شکل موج را در گزارش بیاورید.



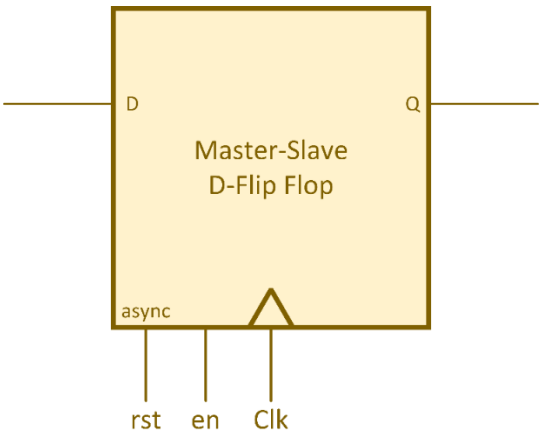
شکل ۱- ماژول D-Latch

[3] Non-Linear Feedback Shift Register

[4] برای اطلاعات بیشتر به مقاله Grain مراجعه کنید: https://www.researchgate.net/publication/272679064

۳. با اتصال دو D-Latch، ساختار D-Flip-Flop را پیاده‌سازی کنید.

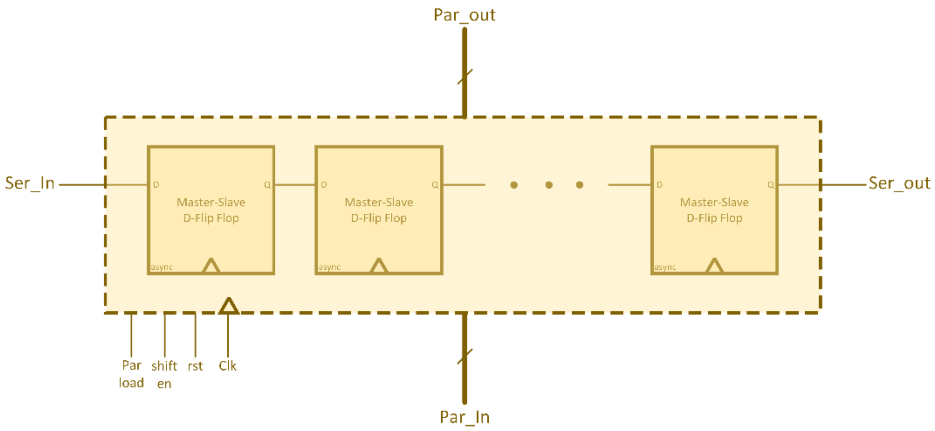
۴. مشابه بخش ۲، تست‌بنچی بنویسید تا تمامی حالات D-Flip-Flop را بررسی کند. نتیجه شکل موج را گزارش کنید.



شکل ۲- ماژول D-Flip-Flop

Shift-Register

۱. از کنار هم قراردادن تعداد کافی از DFF، شیفت‌رجیستر تولید کنید. مطابق شکل ۳، این ماژول علاوه بر clk و rst، دارای ورودی Par_load، برای ذخیره کردن داده n بیتی ورودی Par_in و یک سیگنال کنترلی Shift_en برای فعال‌سازی عملیات شیفت می‌باشد. هم چنین، خروجی n بیتی Par_out، داده‌های موجود در شیفت رجیستر را خروجی می‌دهد.



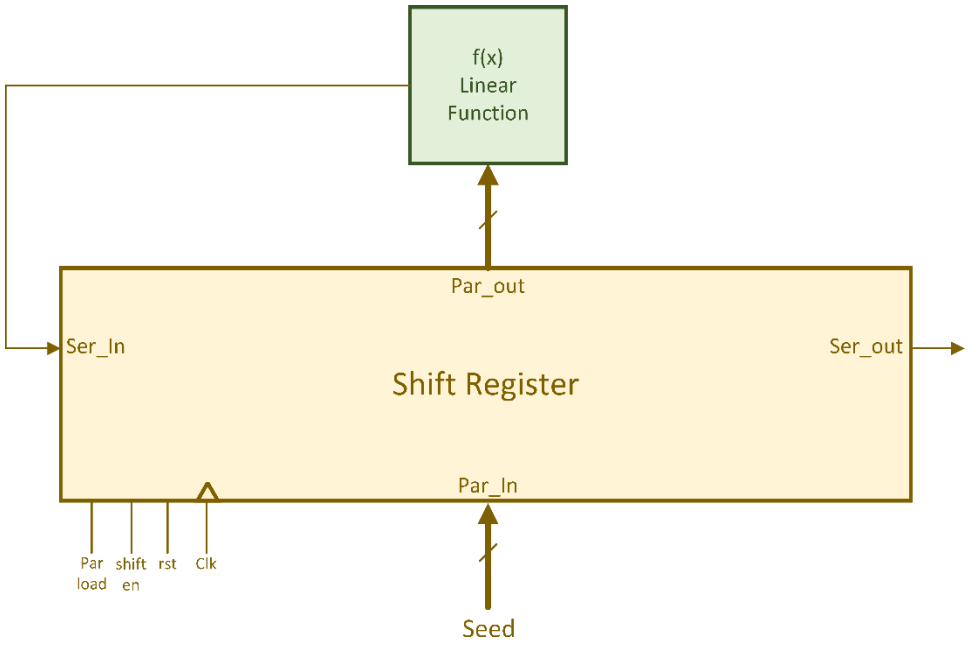
شکل ۳- ماژول Shift Register

برای اتصال و نمونه‌گیری از ماژول‌های خود، می‌بایست از بلوک generate در Verilog استفاده کنید. به کمک generate، می‌توانید با نوشتن حلقه for، تعدادی اتصال و یا نمونه‌گیری‌های مشابه را بدون احتیاج به تکرار در نوشتن، پیاده‌سازی کنید. ابتدا با پیاده‌سازی با تعداد بیت کمتر شروع کنید و از صحت اتصالات خود مطمئن شوید. در نهایت، دو ماژول رجیستر ۸۰ بیتی و ۲۴ بیتی ارائه دهید.

۲. صحت عملکرد این دو ماژول را بررسی کنید. صحت شیفت خوردن داده در هنگام فعال بودن shift_en، و ذخیره‌سازی داده Par_in در هنگام فعال بودن Par_load را بررسی کرده و شکل‌موج‌ها را گزارش کنید.

LFSR

ماژول LFSR را مطابق شکل ۴ طراحی کنید. این ماژول از یک شیفت‌رجیستر ۸۰ بیتی به همراه تابع خطی f تشکیل شده است:



شکل ۴- ماژول LFSR

همانطور که در شکل مشخص است، مقدار Seed به عنوان داده موازی ورودی به Shift register داده می‌شود. داده ذخیره شده در این رجیستر از طریق Par_out قابل دسترسی است. به کمک این خروجی، تابع f را به صورت زیر پیاده کنید:

$$f(X) = X_{62} \oplus X_{51} \oplus X_{38} \oplus X_{23} \oplus X_{13} \oplus X_0$$

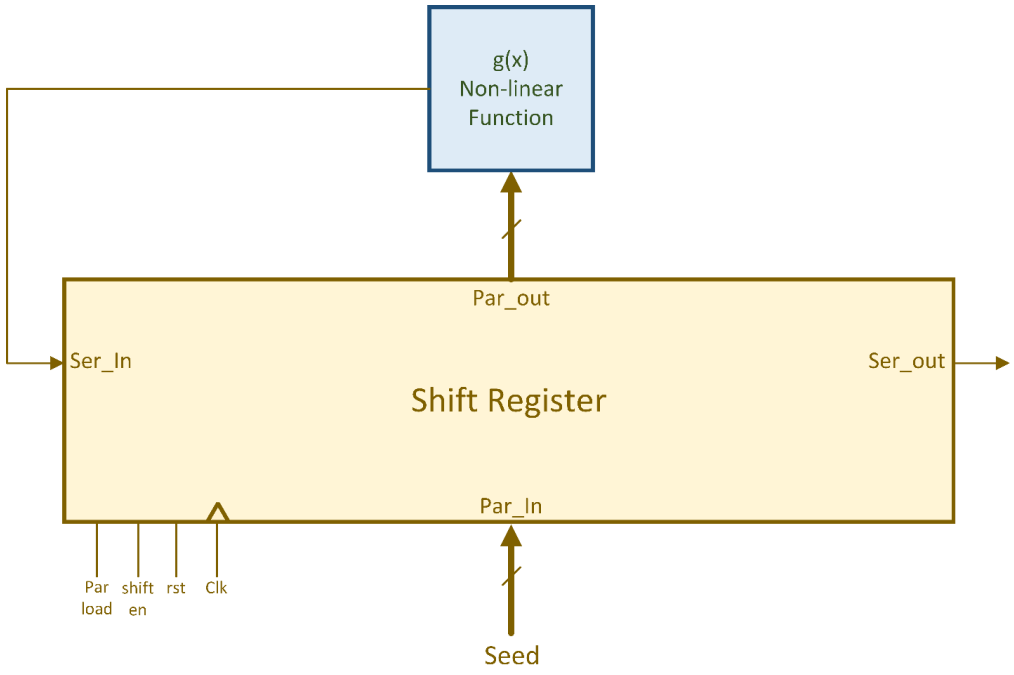
که در این تابع، ورودی X[79:0] بوده و X_i، بیت i ام از X است.

۱. ساختار LSFR را پیاده‌سازی کنید و با نوشتن تست‌بنچ، صحت آن را بررسی کنید. می‌توانید برای پیاده‌سازی شیفت رجیستر، از مدل Behavioral نیز استفاده کنید.
۲. برای بررسی درستی خروجی، از برنامه محک استفاده کنید. حداقل دو نمونه Seed مختلف را تا ۱۰ بیت با خروجی برنامه محک مقایسه کرده و گزارش کنید.

NFSR

ساختار این ماژول در شکل ۵ نمایش داده شده است. همانطور که دیده می‌شود ساختار این مدار با LFSR تفاوت چشمگیری ندارد. تنها نقطه تفاوت این دومدار از یکدیگر این است که تابع اعمال شده روی خروجی شیفت رجیستر، یک تابع غیرخطی است. به این معنا که در این تابع علاوه برعملیات جمع که در مدار شما به کمک گیت XOR پیاده‌سازی می‌شود، چندین رابطه منطقی غیرخطی مانند AND نیز وجود دارد. تابع g(x) که شما وظیفه پیاده‌سازی آن را برعهده دارید، منطق زیر را پیاده‌سازی می‌کند:

$$g(X) = X_0 \oplus X_5 \oplus X_6 \oplus X_9 \oplus X_{17} \oplus X_{22} \oplus (X_4.X_{13}) \oplus (X_8.X_{16}) \oplus (X_5.X_{11}.X_{14}) \oplus (X_2.X_5.X_8.X_{10})$$



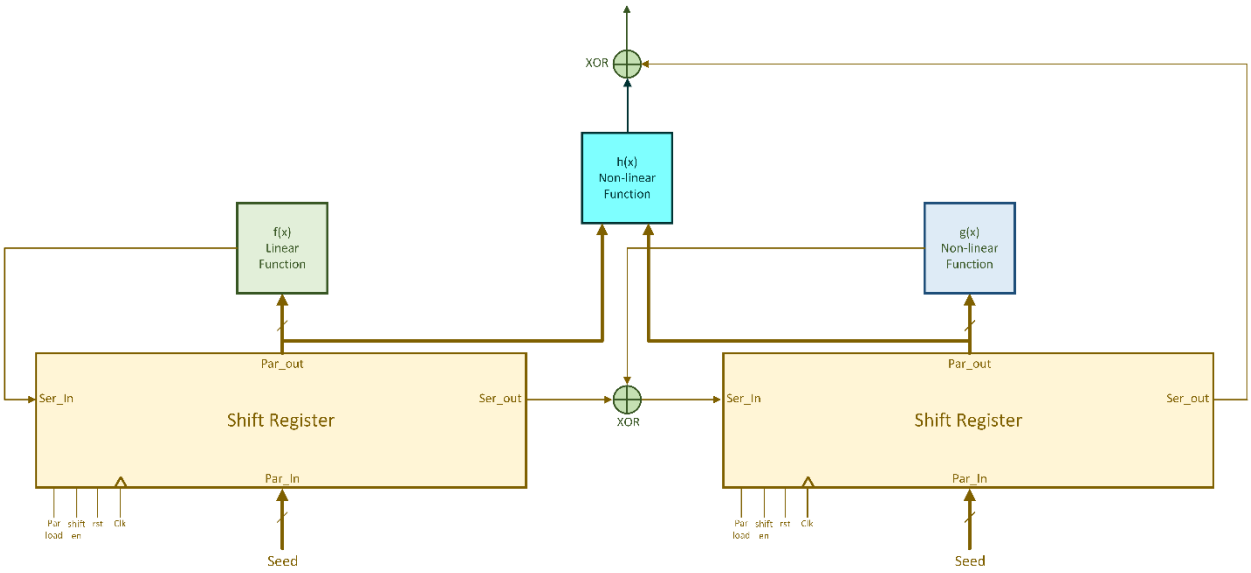
شکل ۵- ماژول NFSR

۱. ساختار NSFR را پیاده‌سازی کنید و با نوشتن تست‌بنچ، صحت آن را بررسی کنید. می‌توانید برای پیاده‌سازی شیفت‌رجیستر، از مدل Behavioral نیز استفاده کنید.
۲. برای بررسی درستی خروجی، از برنامه محک (در فایل مربوط به تمرین ضمیمه شده است) استفاده کنید. حداقل دو نمونه Seed مختلف را تا ۱۰ بیت با خروجی برنامه محک مقایسه کرده و گزارش کنید.

Grain

در این بخش با کنار هم قرار دادن دو ماژول LFSR و NFSR می‌خواهیم ماژول قدرتمندتری برای تولید اعداد تصادفی طراحی کنیم. شکل ۶ نشان‌دهنده مدار مربوط به این طراحی است. همانطور که مشاهده می‌شود، ماژول NFSR تغییر جزئی می‌کند به این صورت که سیگنال تولید شده توسط تابع غیرخطی g(x)، با سیگنال سریال خروجی ماژول LFSR جمع شده (به کمک یک گیت XOR) و نتیجه به عنوان سریال ورودی به ماژول NFSR داده می‌شود. به علاوه تابع غیرخطی دیگری نیز به نام h(x) تعریف می‌شود که خروجی موازی ماژول LFSR و NFSR را دریافت کرده و با انجام عملیات منطقی زیر یک سیگنال خروجی تحویل ما می‌دهد که جمع این سیگنال با سیگنال سریال خروجی NFSR سیگنال خروجی مدار را به ما می‌دهد:

$$h(X_L, X_N) = X_{L0} \oplus X_{L3} \oplus (X_{L1}.X_{L2}) \oplus X_{N0} \oplus (X_{N1}.X_{L5}) \oplus (X_{N3}.X_{L7}) \oplus (X_{L8}.X_{L13}.X_{N5}) \oplus X_{N2}$$



شکل ۶- ماژول Grain

۱. ساختار Grain را به کمک ماژول‌های بخش‌های قبلی پیاده‌سازی کنید و با نوشتن تست‌بنچ، صحت آن را بررسی کنید.

۲. برای بررسی درستی خروجی، از برنامه محک استفاده کنید. حداقل دو نمونه Seed مختلف را تا ۳۲ بیت با خروجی برنامه محک مقایسه کرده و گزارش کنید.

۳. *امتیازی:* در مورد روش‌های نوشتن داده در فایل در Verilog تحقیق کنید. به کمک این روش‌ها، تست بنچ را به گونه‌ای تغییر دهید تا داده‌های خروجی این ماژول به ازای ۱۰۰۰ سیکل Clock، در یک فایل ذخیره کند، و همچنین تعداد رخدادهای 0، 1، 00 (به ازای دو سیکل متوالی)، 01، 10 و 11 را شمارش کند و در نهایت، در فایل تولید شده گزارش کند.

بارم‌بندی سوالات

- پیاده‌سازی D-Latch: ۱۰ نمره
- پیاده‌سازی D-Flip-Flop: ۱۰ نمره
- پیاده‌سازی Shift Register: ۱۰ نمره
- پیاده‌سازی LFSR: ۱۵ نمره

- پیاده‌سازی NFSR: ۱۵ نمره
- پیاده‌سازی NFSR: ۳۰ نمره
- فرمت صحیح گزارش و پوشه‌بندی فایل‌ها : ۱۰ نمره

نکات تحویل تمرین:

۱- پاسخ خود به سوالات را در یک فایل pdf با فرمت DLD_CA#3_StudentNumber.pdf تحویل دهید.

۲- توضیح کدهای مربوط به تمامی بخش‌ها، نتایج شبیه‌سازی‌ها و همچنین توضیحات مربوط به آن‌ها در گزارش بیان شود.

با آرزوی بهترین‌ها برای شما