دانشکده مهندسی کامپیوتر

طراحی سیستم‌های دیجیتال

مستندات آزمون پایانترم

استاد : دکتر اجلالی

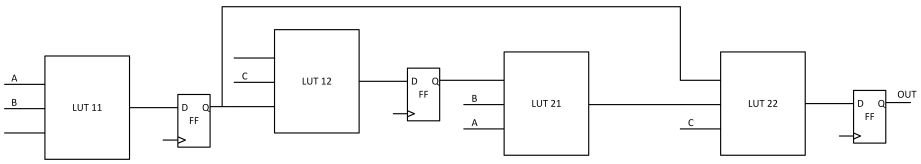
پارسا محمدیان — ۹۸۱۰۲۲۸۴

۲۰ تیر ۱۴۰۰

فهرست مطالب

۱ سوال سه

ابتدا مدار را ساده‌تر و بدون CB و SB رسم می‌کنیم. (شکل ۱)



شکل ۱: ساده شده مدار FPGA

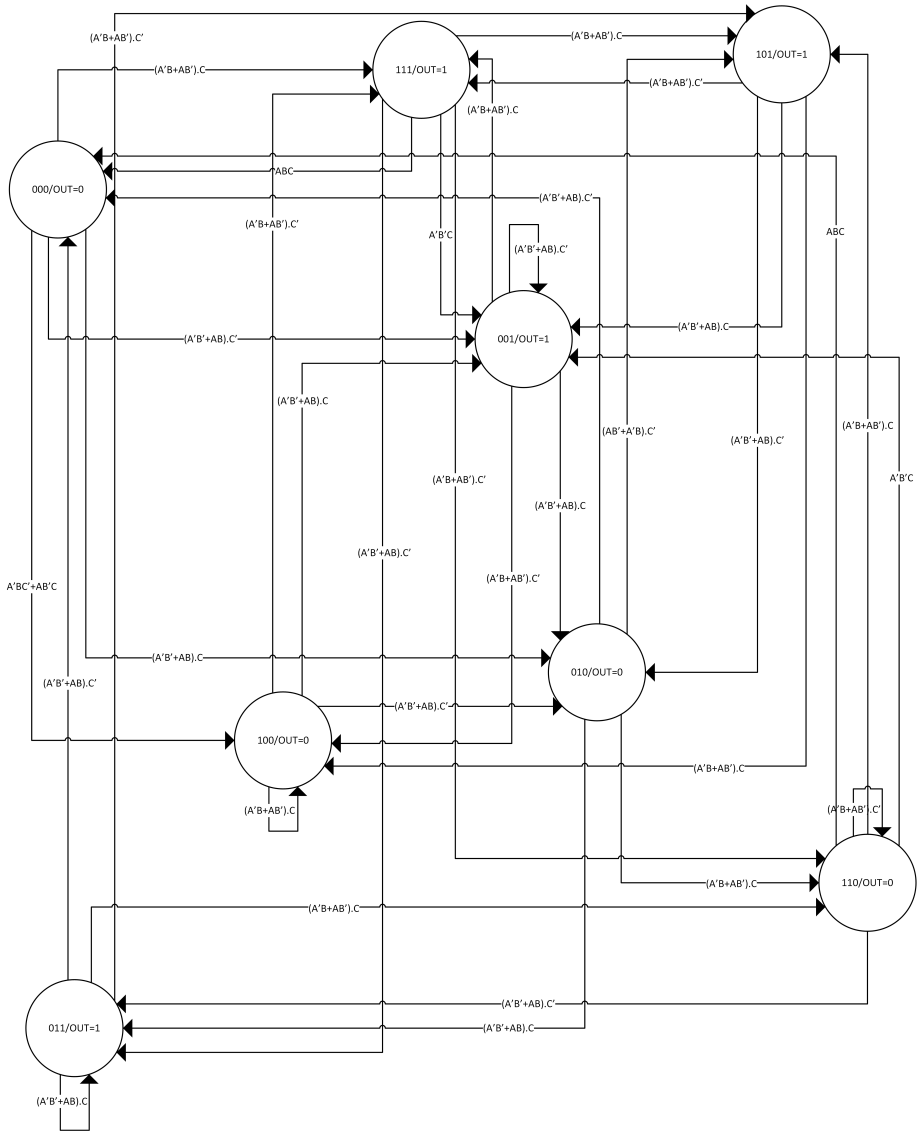
همانطور که از شکل ۱ مشخص است مدار ترتیبی است، چون خروجی به غیر از ورودی به حالت‌ها (مقدار فلیپ فلاپ‌ها) نیز وابسته است.
حال به کمک جدول LUT جدول حالت را بدست می‌آوریم.
در این جدول Q_{11} مقدار فعلی فلیپ فلاپ کنار LUT 11 ، Q_{12} مقدار فعلی فلیپ فلاپ کنار LUT 12 و Q_{22} مقدار فعلی فلیپ فلاپ کنار LUT 22 هستند و Q_{11}^{+} مقدار بعدی فلیپ فلاپ کنار LUT 11 ، Q_{12}^{+} مقدار بعدی فلیپ فلاپ کنار LUT 12 و Q_{22}^{+} مقدار بعدی فلیپ فلاپ کنار LUT 22 هستند.

Q_{11}	Q_{12}	Q_{22}	A	B	C	Q_{11}^{+}	Q_{12}^{+}	Q_{22}^{+}	out
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	1	0	1	0	0
0	0	0	0	1	0	1	0	0	0
0	0	0	0	1	1	1	1	1	0
0	0	0	1	0	0	1	0	0	0
0	0	0	1	0	1	1	1	1	0
0	0	0	1	1	0	0	0	1	0
0	0	0	1	1	1	0	1	0	0
0	0	1	0	0	0	0	0	1	1
0	0	1	0	0	1	0	1	0	1
0	0	1	0	1	0	1	0	0	1
0	0	1	0	1	1	1	1	1	1
0	0	1	1	0	0	1	0	0	1
0	0	1	1	0	1	1	1	1	1
0	0	1	1	1	0	0	0	1	1
0	0	1	1	1	1	0	1	0	1
0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	1	1	0
0	1	0	0	1	0	1	0	1	0
0	1	0	0	1	1	1	1	0	0
0	1	0	1	0	0	1	0	1	0
0	1	0	1	0	1	1	1	0	0
0	1	0	1	1	0	0	0	0	0

0	1	0	1	1	1	0	1	1	0
0	1	1	0	0	0	0	0	0	1
0	1	1	0	0	1	0	1	1	1
0	1	1	0	1	0	1	0	1	1
0	1	1	0	1	1	1	1	0	1
0	1	1	1	0	0	1	0	1	1
0	1	1	1	0	1	1	1	0	1
0	1	1	1	1	0	0	0	0	1
0	1	1	1	1	1	0	1	1	1
1	0	0	0	0	0	0	1	0	0
1	0	0	0	0	1	0	0	1	0
1	0	0	0	1	0	1	1	1	0
1	0	0	0	1	1	1	0	0	0
1	0	0	1	0	0	1	1	1	0
1	0	0	1	0	1	1	0	0	0
1	0	0	1	1	0	0	1	0	0
1	0	0	1	1	1	0	0	1	0
1	0	1	0	0	0	0	1	0	1
1	0	1	0	0	1	0	0	1	1
1	0	1	0	1	0	1	1	1	1
1	0	1	0	1	1	1	0	0	1
1	0	1	1	0	0	1	1	1	1
1	0	1	1	0	1	1	0	0	1
1	0	1	1	1	0	0	1	0	1
1	0	1	1	1	1	0	0	1	1
1	1	0	0	0	0	0	1	1	0
1	1	0	0	0	1	0	0	1	0
1	1	0	0	1	0	1	1	0	0
1	1	0	0	1	1	1	0	1	0
1	1	0	1	0	0	1	1	0	0
1	1	0	1	0	1	1	0	1	0
1	1	0	1	1	0	0	1	1	0
1	1	0	1	1	1	0	0	0	0
1	1	1	0	0	0	0	1	1	1
1	1	1	0	0	1	0	0	1	1
1	1	1	0	1	0	1	1	0	1
1	1	1	0	1	1	1	0	1	1
1	1	1	1	0	0	1	1	0	1
1	1	1	1	0	1	1	0	1	1
1	1	1	1	1	0	0	1	1	1
1	1	1	1	1	1	0	0	0	1

حال با توجه به جدول حالت نمودار حالت را رسم می‌کنیم. (شکل ۲) در شکل ۲ حالت‌ها با کدهای باینری که از کنار هم گذاشتن خروجی فلیپ فلاپ بدست می‌آید شماره‌گذاری شده‌اند.

$$\text{شماره حالت} = \overline{Q_{11}Q_{12}Q_{22}}$$



شکل ۲: نمودار حالت کشیده شده از روی جدول حالت

از آنجایی که ۳ ورودی داریم به صورت کلی از هر حالت سه یال به حالات دیگر خارج می‌شود. حال با ساده‌سازی (ادغام یال‌هایی که مبدا و مقصد مشترک دارند و شروطشان) این یال‌ها کمتر می‌شوند.

۲ سوال شش

۱.۲ شرط ورودی In نسبت به CLK

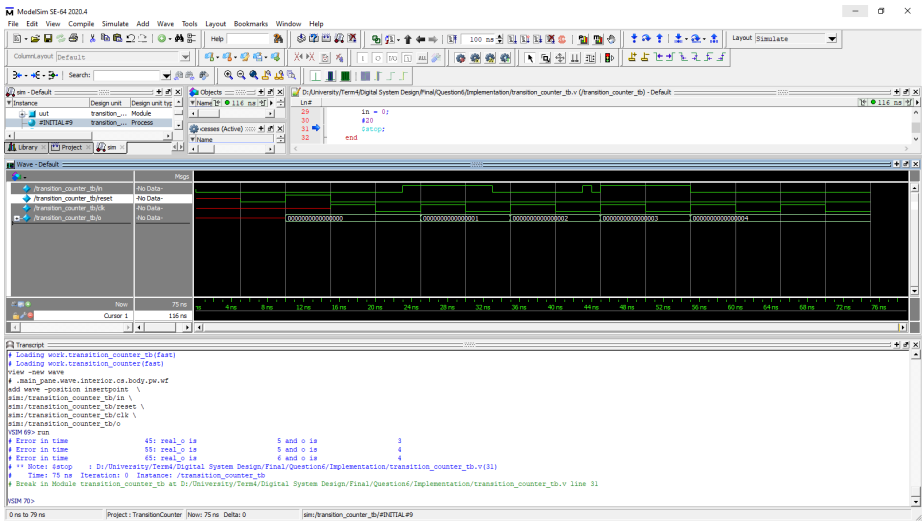
در حالت کلی، هر تغییر ورودی In باید تا triger شدن کلاک باقی بماند تا اثرش دیده شود.
اگر فرض کنیم ورودی In تغییر می‌کند، و قبل از triger شدن کلاک، زوج بار تغییر می‌کند، آنگاه وقتی با triger شدن کلاک با مقدار قبلی خود (به منظور کشف تغییر) مقایسه می‌شود، چون زوج بار تغییر کرده است نتیجه مقایسه برابری است. پس در این صورت اصلا تغییرات شمرده نمی‌شوند.
از طرفی دیگر اگر فرض کنیم ورودی In تغییر می‌کند، و قبل از triger شدن کلاک، فرد بار تغییر می‌کند، آنگاه وقتی با triger شدن کلاک با مقدار قبلی خود (به منظور کشف تغییر) مقایسه می‌شود، نتیجه مقایسه نشان می‌دهد نسبت به مقدار قبلی خود تغییر کرده است و یک تغییر شمرده می‌شود. در حالیکه می‌دانیم ممکن است بیش از یک بار (مثلا ۵ بار) تغییر کرده باشد.

۲.۲ توصیف رفتاری

جزئیات پیاده‌سازی در فایل transmition_counter.v موجود است. در این ماژول قسمتی که از سیستم تسک برای چاپ خطا بر روی صفحه استفاده شده قابل سنتز نبوده و تنها در شبیه‌سازی عمل می‌کند. توجه شود که چک کردن خطا در لبه بالارونده کلاک انجام می‌شود.

۳.۲ شبیه‌سازی و تست کد

برای اطمینان از صحت عملکرد مدار، تست بنچ در فایل transmition_counter_tb.v نوشته شده است. در قسمت اول آن تغییرات از شرط بخش الف پیروی می‌کنند پس تغییرات به درستی شمرده می‌شوند و خطایی رخ نمی‌دهد. در قسمت دوم که تغییرات مطابق شرط بخش اول نیستند خطای مناسب چاپ می‌شود.
جزئیات اجرای شبیه‌سازی در شکل ۳ قابل مشاهده است.



شکل ۳: نتیجه اجرای شبیه‌سازی transition counter

۳ سوال هفت

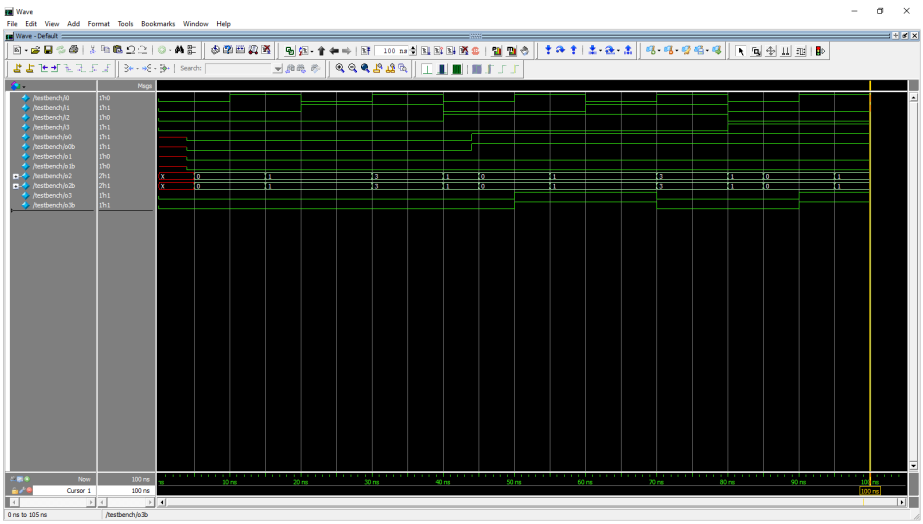
۱.۳ پیاده‌سازی برنامه پایتون

برنامه نوشته شده که در فایل dataflow2behavioral/main.py موجود است به صورت
python main.py -i <inputFile> -o <outputFile>
اجرا شده و inputFile را مورد پردازش قرار می‌دهد و حاصل را در outputFile می‌ریزد. بیشتر پردازش‌ها توسط Regex انجام شده است.

۲.۳ تست کردن برنامه

۱.۲.۳ تست فرضی

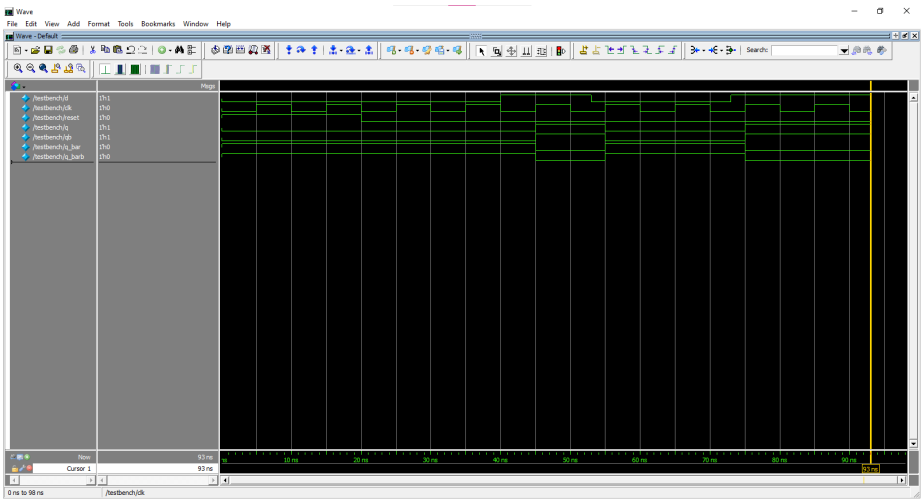
در این تست یک برنامه فرضی نوشته شده که از انواع Continues Assignment ها استفاده می‌کند. این کد را با برنامه پایتون به کد رفتاری تبدیل می‌کنیم. سپس ماژولی برای تست آن به این صورت می‌نویسیم که تمام حالات ورودی را مقداردهی کند سپس خروجی دو ماژول رفتاری و جریان داده را از طریق شکل موج مقایسه می‌کنیم. کدهای جریان داده، رفتاری و تست به ترتیب در فایل‌های dataflow.v و behavioral.v و testbench.v موجود هستند. نتیجه اجرای testbench نیز در شکل ۴ قابل مشاهده است.



شکل ۴: همانطور که مشاهده می‌شود خروجی‌های متناظر کاملا یکسان هستند. (دقت شود که خروجی‌های oi مربوط به توصیف جریان داده و خروجی‌های oib مربوط به توصیف رفتاری هستند)

۲.۲.۳ تست عملیاتی

حال کد تمرین ۵ هیمن درس که یک واحد dff به صورت جریان داده بود را مورد تست قرار می‌دهیم. باز هم مشاهده می‌شود که شکل موج خروجی‌ها کاملا یکسان است. (شکل ۵)



شکل ۵: همانطور که مشاهده می‌شود خروجی‌های متناظر کاملا یکسان هستند. (دقت شود که خروجی‌های x مربوط به توصیف جریان داده و خروجی‌های xb مربوط به توصیف رفتاری هستند)