دستور کار آزمایشگاه مدارهای منطقی و معماری کامپیوتر مبتنی بر شبیه سازی

ویرایش ۹۹۱

محمود نادران طحان، استادیار گروه مهندسی کامپیوتر

دانشگاه شهید چمران اهواز

مقدمه:

با توجه به برگزاری کلاس‌های درس به صورت مجازی به خاطر شیوع بیماری کرونا، این دستور کار حاوی ۱۳ شبیه سازی نرم افزاری است. شبیه سازی‌های ۱ تا ۸ در وب سایت https://circuitverse.org انجام خواهند شد. این وب سایت حاوی یک تعداد قطعه مدار منطقی است که با اتصال آن‌ها می‌توانید عملکرد مدار را ارزیابی کنید. تغییر رنگ سیم‌ها به خاطر مقادیر ۰ و ۱ به شما در درک بهتر مدار منطقی و عیب یابی آن‌ها کمک می‌کند. شبیه سازی‌های ۹ تا ۱۱ با استفاده از نرم افزار Atanua انجام خواهند شد. در این نرم افزار تعدادی از تراشه‌های سری 74xx استفاده خواهند شد. با مراجعه به برگه داده (data sheet) این تراشه‌ها عملکرد پایه‌ها را فرا خواهید گرفت. شبیه سازی‌های ۱۲ و ۱۳ با استفاده از زبان توصیف سخت افزار Verilog انجام خواهند شد. در این قسمت می‌توانید از نرم افزارهای MoelSim یا Active-HDL استفاده کنید. توصیه می‌شود از نرم افزار Active-HDL که محیط ساده تری دارد، استفاده کنید.

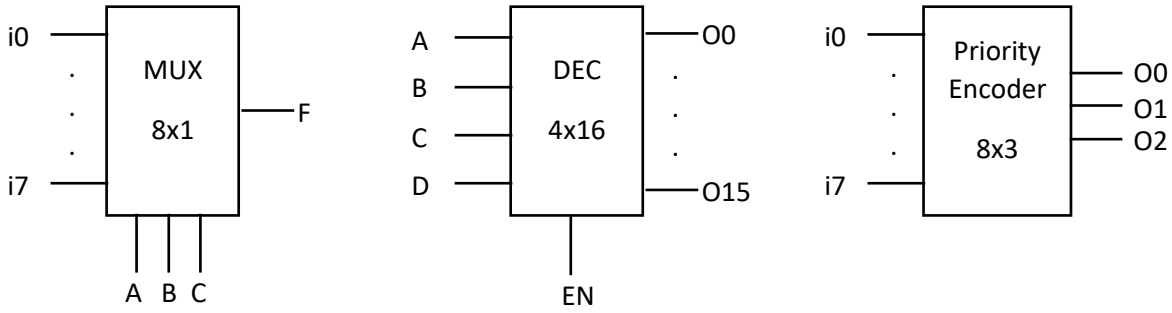
تمرین‌های این دستور کار به گونه‌ای هستند که بیشتر آن‌ها در یک طول یک هفته قابل انجام هستند. با این حال برخی از آن‌ها (سری وریلاگ) ممکن است بیشتر از یک هفته نیاز داشته باشند. در ابتدای ترم فعالیت‌های مربوط به تمرین‌ها در سایت به همراه مهلت آپلود آن‌ها تعیین می‌شوند. تحویل تمرین‌ها بعد از مهلت قابل قبول نیستند. با این حال با توجه به بارم تمرین‌ها در صورتی که یک تمرین را از دست دادید، می‌توانید با تحویل به موقع تمرین‌های بعدی جبران کنید. بارم تمرین‌ها به شرح زیر است.

شبیه سازی	بارم (از ۲۶)	مدت زمان (از ۱۲ هفته)
۱	۲	۱
۲	۲	۱
۳	۲	۱
۴	۱	۱
۵	۱	
۶	۲	۱
۷	۲	۱
۸	۳	۱
۹	۱	۱
۱۰	۱	
۱۱	۳	۱
۱۲	۳	۱
۱۳	۳	۲

شبیه سازی ۱: مدارهای ترکیبی پایه

مدارهای زیر را با گیت‌های منطقی پیاده سازی کنید و عملکرد آن‌ها را با توجه به مقادیر مختلف ورودی بررسی کنید. برای خروجی از LED استفاده کنید.

مدار Priority Encoder 8x3 برای فشرده سازی چند خط ورودی به تعداد کمتری خط در خروجی به کار می‌رود. خط‌های خروجی عملاً نمایش دهنده مقدار باینری خط ورودی انتخاب شده است. اگر چند خط ورودی همزمان یک باشند، مقدار خروجی برابر نمایش باینری آن خطی از ورودی است که مقدار بیشتری دارد. بنابراین سایر خط‌های با مرتبه پایین don't care هستند. تنها حالت غیر مجاز و استثناء این است که همه ورودی‌ها صفر باشد. ابتدا جدول صحت آن را بنویسید سپس معادلات خروجی بر حسب ورودی را تعیین و در نهایت مدار را رسم کنید.



شبیه سازی ۲: مدارهای ترتیبی پایه

مدارهای RS latch و فلیپ فلاپ‌های D، JK و T را رسم و ارزیابی کنید. توجه کنید که برای پیاده سازی باید از گیت‌های منطقی استفاده کنید و بلوک‌های فلیپ فلاپ در سایت قابل قبول نیستند. با استفاده از فلیپ فلاپ D، یک شیفت رجیستر ۸ بیتی به صورت Parallel-in/Serial-out مانند تراشه 74166 طراحی کنید.

شبیه سازی ۳: جمع کننده

مدار یک جمع کننده Ripple carry و یک جمع کننده Carry lookahead ۴ بیتی را پیاده سازی کنید.

شبیه سازی ۴: دیکدر BCD به Seven Segment LED

یرای این کار ابتدا جدول صحت را بنویسید. ورودی ۴ بیت A و B و C و D هستند و خروجی‌ها ۷ پایه a و b و c و d و e و f و g هستند. به ازای هر ترکیب ورودی (از ۰ تا ۹) مشخص کنید که کدام خروجی ها باید ۱ باشند. سپس معادله هر خروجی را بنویسید و با گیت‌های منطقی رسم کنید.

شبیه سازی ۵: مدار تقسیم فرکانس

ابتدا با استفاده از یک فلیپ فلاپ D، یک مقدار تقسیم بر ۲ بسازید. برای این کار خروجی not(Q) را به ورودی D وصل کنید. با اتصال یک کلاک ۱ هرتز و یک LED در خروجی Q مشاهده کنید که فرکانس کلاک ورودی نصف می‌شود. سپس تعداد طبقات را افزایش دهید (۳ طبقه) و به صورت آبشاری (cascade) به یکدیگر وصل کنید. در این قسمت، خروجی فلیپ فلاپ مرحله بالاتر به کلاک فلیپ فلاپ مرحله پایین تر وصل می‌شود. عملکرد مدار را با بررسی ترتیب روشن شدن LEDها بررسی کنید. آیا ترتیب خاموش و روشن شدن LEDها چگونه است؟ کدام فلیپ فلاپ MSB و کدام فلیپ فلاپ LSB هستند؟

شبیه سازی ۶: پیاده سازی ALU ۴ بیتی (قسمت اول)

در این شبیه ساز ابتدا یک سلول ALU یک بیتی کامل را که قابلیت پیاده سازی AND، OR، جمع و تفریق دارند، با استفاده از گیت‌های منطقی پیاده سازی کنید. سپس از آن به عنوان یک بلوک در طراحی ALU ۴ بیتی استفاده کنید. ALU نهایی همچنین باید سیگنال zero را پیاده سازی کند. جزئیات بیشتر در کتاب معماری پترسون وجود دارد.

شبیه سازی ۷: پیاده سازی ALU ۴ بیتی (قسمت دوم)

در ادامه شبیه سازی شماره ۶، واحد کنترل ALU را پیاده سازی کنید و آن را به ALU ۴ بیتی وصل کنید. در این قسمت، مدار طراحی شده در شبیه سازی ۴ را به عنوان یک بلوک BCD-to-7-segment-LED به این صورت استفاده کنید که مقدار دسیمال دو عدد ورودی و عدد خروجی را نمایش دهد. سپس با تغییر ورودی‌های واحد کنترل بررسی کنید که مدار ALU اعمال جمع، تفریق، AND و OR را انجام می‌دهد.

شبیه سازی ۸: آشکار ساز دنباله

برای آشکار سازی دنباله ۱۰۰۱، یک ماشین MEALY و یک ماشین MOORE طراحی کنید. برای ارسال ورودی یک بیتی به ماشین، شیفت رجیستر طراحی شده در شبیه سازی ۲ را به عنوان یک بلوک به ماشین وصل کنید. در این صورت یک مقدار ۸ بیتی در رجیستر بارگذاری می‌شود و در هر کلاک یک بیت از سمت راست به ماشین وارد می‌شود. در صورتی که ماشین دنباله ۱۰۰۱ را تشخیص داد، یک LED در خروجی روشن خواهد شد.

شبیه سازی ۹: جمع کننده یک بیتی با استفاده از گیت‌های منطقی و دیکدر

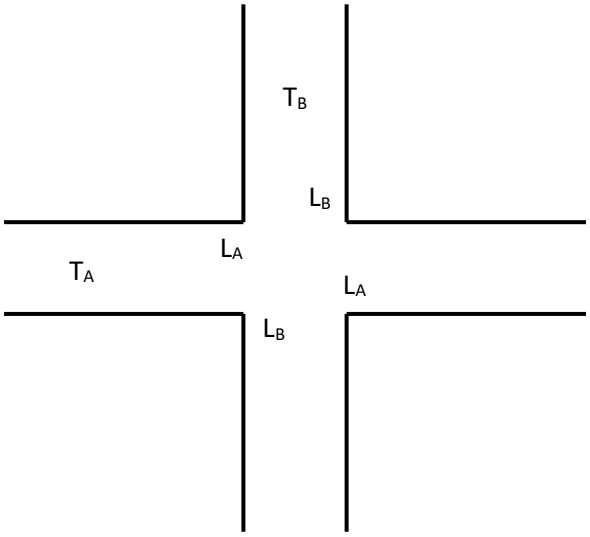
با مراجعه به دستور کار اصلی (آزمایش ۲) و تراشه‌های ۷۴۱۳۸ و ۷۴۰۰ یک جمع کننده یک بیتی با استفاده از نرم افزار Atanua طراحی کنید.

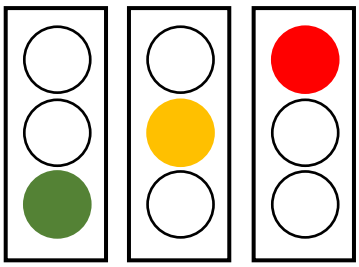
شبیه سازی ۱۰: شمارنده جانسون

یک شمارنده جانسون ۴ بیتی با استفاده از تراشه ۷۴۷۴ و نرم افزار Atanua طراحی کنید. تراشه مربوطه در قسمت Chips قابل دسترسی است. به عملکرد پایه‌ها بر اساس data sheet توجه کنید. برای جزئیات بیشتر به دستور کار اصلی (آزمایش ۴) مراجعه کنید.

شبیه سازی ۱۱: کنترلر ترافیک

چهار راه زیر را در نظر بگیرید. T_A و T_B ترافیک‌های افقی و عمودی در این چهار راه هستند و L_A و L_B دو چراغی هستند که ترافیک را کنترل می‌کنند. یک چراغ راهنمایی در سه وضعیت سبز، زرد و قرمز کار می‌کند. فرض کنید چراغ سبز است. در مرحله بعد چراغ زرد و سپس قرمز می‌شود و این سیکل تکرار می‌شود (بعد از قرمز دوباره سبز می‌شود). همچنین فرض کنید در صورتی که چراغ سبز و ترافیک وجود داشته باشد، آن چراغ سبز خواهد بود و در صورتی از سبز به زرد تغییر می‌کند که در آن مسیر ترافیکی وجود نداشته باشد (یعنی اینکه چند سیکل ساعت سبز باشد، تعیین کننده نیست). توجه کنید زمانی که L_A سبز می‌شود (به عنوان مثال) ترافیک شرقی به غربی و برعکس برقرار خواهد بود. یک ماشین حالت متناهی Moore برای این کنترلر طراحی کنید. سپس با استفاده از تراشه ۷۴۷۴ آن را در نرم افزار Atanua پیاده سازی کنید.





شبیه سازی ۱۲: پیاده سازی ALU ۳۲ بیتی با Verilog

مطابق با توضیحات دستور کار اصلی (آزمایش ۹) و کتاب معماری پترسون، کد وریلاگ یک ALU ۳۲ بیتی را به همراه واحد کنترل پیاده سازی و شبیه سازی کنید. با توجه به این که این مدار ترکیبی است، استفاده از always مجاز نیست.

شبیه سازی ۱۳: پیاده سازی رجیستر فایل

مطابق با توضیحات دستور کار اصلی (آزمایش ۱۰) و کتاب معماری پترسون، کد وریلاگ یک رجیستر فایل را پیاده سازی و شبیه سازی کنید.