A-PDF Watermark Demo. Purchase from www.A-PDF.com to remove the watermark

دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

مدارهای منطقی پیشرفته

دکتر محمدکاظم اکبری

دکتر بهمن جوادی

تابستان ۱۳۸۷

نسخهٔ 5/۱

نسخه پیش‌نویس

Draft

نسخه پیشنویس
Draft

دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

مدارهای منطقی پیشرفته

دکتر محمدکاظم اکبری

دکتر بهمن جوادی

تابستان ۱۳۸۷

نسخهٔ 5/۱

نسخه پیش‌نویس
Draft

مقدمه

از هنگامیکه اولین مدارات منطقی ساخته شدند، زمان زیادی نمی‌گذرد اما در همین مدت کوتاه، تأثیر زیادی بر زندگی بشر گذاشته‌اند. امروزه، کمتر وسیله‌ای وجود دارد که از فن‌آوری مدارات منطقی بهره نبرده باشد. به همین منظور، دانش مربوط به این مدارات با رشد فراوانی روبرو بوده است.

یکی از شاخه‌های این علم، مدارات نا همگام[1] است. از جنبهٔ تئوری، به دلیل عدم وجود سیگنال ساعت در این مدارات، سرعت آنها افزایش یافته و توان کمتری را نسبت به مدارات منطقی همگام[2] مصرف می‌کنند. همچنین، سرعت پاسخگویی این مدارات به سیگنال‌های ورودی، بیشتر است و به صورت بلادرنگ عمل می‌کنند. از این رو، این دسته از مدارات جاذبهٔ زیادی برای انجام تحقیقات بیشتر ایجاد کرده‌اند. بالاخص در محیط‌های نظامی که به سیستم‌های بلادرنگ و کم مصرف نیاز است، مبحث مدارات نا همگام با جدیت دنبال شده است.

اما این مدارات با مشکلاتی نیز همراه هستند. به عنوان مثال، طراحی آنها ساده نیست، نرم‌افزارهای طراحی و ساخت این مدارات، اندک است و در نهایت، آنچه تولید می‌شود با آنچه که از این فن‌آوری انتظار می‌رود، فاصلهٔ زیادی دارد. به همین جهت، رشد تحقیقات و توسعهٔ این مدارات، به تدریج کاهش یافت تا اینکه مجدداً در سال‌های اخیر، ظهور روش‌های جدید و ترکیبی که از مزایای مدارات همگام و نا همگام به صورت همزمان بهره می‌برد، سبب بازگشت مجدد این شاخهٔ علوم دیجیتال به صحنهٔ تحقیقات عملی شده است.

نظامی بودن این شاخه و پیچیدگی‌های آن سبب شده است که منابع موجود در زمینهٔ مدارات نا همگام اندک باشد. به همین جهت، دانشجویان درس مدارهای منطقی، عمدتاً با کمبود منابع و مراجع مورد نیاز در بحث مدارات منطقی پیشرفته، روبرو هستند. هدف از تألیف این کتاب، جبران این کمبود و پر کردن این خلاء بوده است. کتاب حاضر تلاش می‌کند تا در ابتدا مقدمات مدارات دیجیتال را به صورت فشرده بررسی کند و سپس با تفصیل بیشتری، به مدارات نا همگام بپردازد.

---

[1] آسنکرون
[2] سنکرون

نسخه پیش‌نویس

در فصل ۱، اصول پایهٔ مدارات منطقی بیان می‌شوند. سپس، عناصر ترتیبی و حافظه‌ها، در فصل ۲ مورد بررسی قرار می‌گیرند. فصل ۳، روش‌های طراحی مدارات همگام را تشریح کرده و در فصل ۴، چگونگی ساده‌سازی این مدارات مطرح می‌شود.

در ادامه، مباحث مربوط به مدارات نا همگام مورد بررسی قرار می‌گیرد. فصل ۵ به بیان مفاهیم اولیهٔ این مدارات می‌پردازد. فصل ۶ به بررسی دو مشکل اساسی در طراحی و ساخت مدارات نا همگام، یعنی تأخیر و مخاطره اختصاص یافته است. در فصل ۷، ماشین‌های بدون از دست دادن اطلاعات[3] مورد بررسی قرار می‌گیرند. فصل ۸، روش جدید طراحی و ساخت مدارات نا همگام (روش سنتز مارتین) را ارائه می‌دهد .در فصل ۹ افزاره‌های منطقی برنامه‌پذیر معرفی شده و به صورت اجمالی تشریح می‌شوند. منطق مختلط در فصل ۱۰ مطرح می‌شود و در نهایت فصل ۱۱، به بررسی کامل‌تر مبحث آزمون در مدارات نا همگام می‌پردازد.

دانشجویان زیادی در تألیف این کتاب و تکمیل بخش‌های گوناگون آن نقش داشته‌اند که در اینجا از زحمات آقایان **عماد بهرامی سامانی** و **سامان امیرپور امرائی** و **محمد صالحی خانقاه‌بر**، **محسن راجی اسد آبادی** و **میلاد اسدی** قدردانی می‌گردد.

این کتاب نیز از نواقص خالی نیست و رهنمودهای اساتید این علم، می‌تواند در جهت هر چه پربارتر شدن آن، مفید واقع شود.

[3] Information Lossless Machines

www.jozve.org

فهرست مطالب

۱ مقدمه‌ای بر مدارهای منطقی ۲

۱-۱ مقدمه ۳

۱-۲ مدارهای ترکیبی ۷

۱-۲-۱ روش جبری ۸

۱-۲-۲ روش جدول درستی ۹

۱-۳ روشهای ساده‌سازی ۹

۱-۳-۱ جدول کارنو ۱۰

۱-۳-۲ رابطه نمودار ون و جدول درستی ۱۰

۱-۳-۳ خطوط کلی روش ساده سازی با استفاده از جدول کارنو ۱۳

۱-۳-۴ روش جدول بندی کویین – مک کلاسکی برای ساده کردن توابع ۱۹

۱-۴ تمرین ۲۶

۲ معرفی عناصر ترتیبی ۲۸

۲-۱ مقدمه ۲۹

۲-۲ جدول حالت و نمودار حالت ۳۰

۲-۳ عناصر حافظه ۳۳

۲-۳-۱ نگهدار SR ۳۴

۲-۳-۱-۱ جدول تحریک نگهدار SR ۳۷

۲-۳-۱-۲ نگهدار SR گیت‌دار ۳۷

۲-۴ فیلیپ-فلاپ ۳۹

۲-۴-۱ فیلیپ-فلاپ SR پایه و پیرو ۳۹

۲-۴-۱-۱ جدول تحریک و معادلهٔ مشخصه ۴۰

۲-۴-۲ فیلیپ-فلاپ D پایه و پیرو ۴۱

۲-۴-۳ فیلیپ-فلاپ JK پایه و پیرو ۴۲

۲-۴-۴ فیلیپ-فلاپ T ۴۲

۲-۴-۵ زمانبندی در فیلیپ-فلاپ‌ها ۴۴

۲-۵ تمرین ۴۵

۳ طراحی مدارهای ترتیبی همگام ۴۸

۳-۱ طراحی مدارهای ترتیبی همگام ۴۹

۳-۲ مدل میلی ۴۹

۳-۳ مدل مور ۵۱

۳-۴ روال طراحی مدار ترتیبی همگام ۵۳

۳-۵ قدم‌های طراحی مدار ترتیبی همگام ۵۶

نسخه پیش‌نویس Draft

۳-۶ جداول ورودی فیلیپ-فلاپ ۵۷
۳-۷ کنترل کنندهٔ محدود-حالت ۷۰
۳-۸ طراحی های ویژه (ساختارهای پر استفاده) ۷۳
3-8-1 رجیستر ۷۳
۳-۸-۲ شیفت رجیستر ۷۴
۳-۸-۳ *شمارنده‌ها* ۷۵
3-۹ تمرین ۷۷

۴ ساده سازی مدارهای ترتیبی ۸۰

۴-۱ مقدمه ۸۱
۴-۱-۱ *تعاریف ساده سازی* ۸۱
۴-۲ روش ساده سازی با استفاده از جدول چارت دوتایی ۸۲
۴-۳ ساده سازی برای مدارهای دارای حالت بی اهمیت ۸۴
۴-۳-۱ *تعاریف ساده‌سازی* ۸۵
۴-۳-۲ *الگوریتم کامپیوتری برای یافتن مجموعهٔ حداکثر سازگاری* ۸۶
۴-۳-۳ *روش دیاگرام ادغام برای یافتن مجموعهٔ حداکثر سازگاری* ۸۹
۴-۳-۴ *یافتن مجموعهٔ حداقل حداکثر سازگاری* ۹۱
۴-۳-۵ *پیدا کردن محدودهٔ حالات نهایی* ۹۲
۴-۳-۶ *الگوریتم ساده‌سازی برای مدارات ترتیبی نامعلوم* ۹۳
۴-۴ تمرین ۱۰۲

۵ مدارهای ترتیبی ناهمگام ۱۰۵

۵-۱ مقدمه ۱۰۶
۵-۲ مزایای مدارهای ترتیبی ناهمگام ۱۰۶
۵-۳ معایب مدارهای ترتیبی ناهمگام ۱۰۸
۵-۴ مشخصات مدارهای ترتیبی ناهمگام ۱۰۸
۵-۵ توصیف مدارهای ترتیبی ناهمگام ۱۰۹
۵-۶ روش طراحی هافمن (روش طراحی مد اساسی) ۱۰۹
۵-۷ طراحی مدارهای ترتیبی ناهمگام به روش هافمن ۱۱۱
۵-۸ تقسیم‌بندی مدارهای ناهمگام از نظر ورودی ۱۱۴
۵-۹ تخصیص حالت در مدارهای ترتیبی ناهمگام ۱۱۹
۵-۹-۱ *مسابقه* ۱۲۱
۵-۹-۲ *از بین بردن مسابقه* ۱۲۳
۵-۹-۲-۱ *روش مجموعه سطر متصل شده* ۱۲۵
۵-۹-۲-۲ *روش مجموعه سطر اشتراکی* ۱۳۰

نسخه پیش‌نویس Draft

۵-۱۰ تمرین ۱۳۴

۶ تأخیر و مخاطره ۱۳۸

۶-۱ مقدمه ۱۳۹
۶-۲ تقسیم بندی تأخیر ۱۳۹
۶-۳ تعریف مخاطره ۱۴۰
۶-۴ انواع مخاطره در مدارهای ترکیبی ۱۴۱
۶-۴-۱ *تعاریف مخاطره در مدارهای ترکیبی* ۱۴۲
۶-۴-۲ *راه حل‌های رفع مخاطره* ۱۴۵
۶-۴-۲-۱ *منطق سه تایی برای تشخیص مخاطره‌های ایستا* ۱۴۶
۶-۴-۲-۲ *کشف مخاطره با استفاده از منطق هشت مقداره* ۱۴۸
۶-۵ مخاطره در مدارهای ترتیبی ۱۵۱
۶-۵-۱ *مخاطرهٔ اساسی در مدارهای ترتیبی ناهمگام* ۱۵۴
۶-۶ مثال کامل از طراحی و سنتز مدار ترتیبی ناهمگام ۱۶۰
۶-۷ تمرین ۱۶۶

۷ ماشین های بدون از دست دادن اطلاعات ۱۶۹

۷-۱ مقدمه ۱۷۰
۷-۲ مدارهای ترکیبی بدون از دست دادن اطلاعات ۱۷۰
۷-۳ چه وقت می گوییم اطلاعات از دست رفته است؟ ۱۷۱
7-۴ تعریف مدارهای بدون از دست دادن اطلاعات با تعداد حالات متناهی ۱۷۳
۷-۴-۱ *کلاس اول مدارهای بدون از دست دادن اطلاعات* ۱۷۳
7-۴-۲ *کلاس دوم مدارهای بدون از دست دادن اطلاعات* ۱۷۶
۷-۴-۳ *کلاس عمومی مدارهای بدون از دست دادن اطلاعات* ۱۸۱

۸ طراحی آسنکرون و روش سنتز مارتین ۱۸۶

۸-۱ مقدمه ۱۸۷
8-2 طراحی مدارهای آسنکرون Quasi Delay-Insensitive ۱۸۸
۸-۳ روش طراحی عمومی ۱۸۹
8-۴ طراحی مدارهای Traditional QDI ۱۹۰
۸-۵ CHP ۱۹۳

۹ افزاره‌های منطقی برنامه پذیر ۱۹۶

۹-۱ مقدمه ۱۹۷
۹-۲ حافظهٔ فقط خواندنی، اولین PLD ۱۹۸
۹-۲-۱ *انواع مختلف* ROM ۱۹۹

نسخه پیش‌نویس Draft

۹-۲-۲ *مقایسهٔ انواع مختلف ROM* ۲۰۰

۹-۲-۳ *پیاده‌سازی مدارهای منطقی به کمک ROM* ۲۰۰

۹-۳ آرایهٔ منطقی برنامه‌پذیر ۲۰۱

۹-۳-۱ *اتصالات قابل برنامه ریزی* ۲۰۲

۹-۳-۲ *انتخاب قطبیت خروجی* ۲۰۳

۹-۳-۳ *پایه های ورودی/خروجی* ۲۰۴

۹-۴ منطق آرایه‌ای برنامه‌پذیر ۲۰۵

۹-۵ افزاره‌های برنامه‌پذیر برای مدارهای ترتیبی ۲۰۶

۹-۶ دریای گیت ۲۰۷

۹-۷ FPGA ۲۰۸

۹-۷-۱ *ویژگی های* FPGA ۲۱۰

۹-۷-۲ *روال طراحی با* FPGA ۲۱۱

۹-۸ تمرین ۲۱۲

۱۰ منطق مختلط ۲۱۵

۱۰-۱ مقدمه ۲۱۶

10-۲ نمایش منطقها ۲۱۷

۱۰-۳ قراردادهای معادل گذاری ۲۱۸

۱۰-۴ نمایش قطعات سخت افزاری حاصل شده با منطق مختلط ۲۱۸

۱۰-۵ نام متغیرها در منطق مختلط ۲۲۰

10-۶ خاصیت دوگانی AND, OR ۲۲۲

۱۰-۷ نظریه منطق مختلط ۲۲۶

۱۰-۸ آنالیز مدارات با منطق مختلط ۲۳۴

۱۰-۸-۱ *قراردادهای طرح مدار* ۲۳۷

۱۰-۹ ساختار کلی منطق مختلط ۲۳۹

۱۰-۱۰ دیگر صورت‌های نمادسازی منطق مختلط ۲۴۳

۱۰-۱۱ منطق مثبت چیست؟ ۲۴۳

۱۰-۱۲ بکارگیری منطق مختلط برای توابع XOR ۲۵۱

نسخه پیش‌نویس Draft

۱۱ آزمون مدارات منطقی ۲۵۸

۱۱-۱ مقدمه ۲۵۹

۱۱-۲ اعمال اصلی در آزمون مدار ۲۶۰

۱۱-۳ نحوهٔ آزمون مدارهای منطقی ۲۶۱

۱۱-۴ مدل‌های خرابی ۲۶۲

۱۱-۵ آزمون خرابی‌های چسبنده ۲۶۲

۱۱-۶ تولید الگوهای آزمون ۲۶۳

۱۱-۶-۱ *روش اول: آزمون کامل* ۲۶۳

۱۱-۶-۲ *روش دوم: روش OR انحصاری* ۲۶۴

۱۱-۶-۳ *روش سوم: روش حساس‌سازی مسیر* ۲۶۵

۱۱-۶-۳-۱ *الگوریتم حساس سازی مسیر* ۲۶۶

۱۱-۷ نکات تکمیلی تولید آزمون در مدارهای ترکیبی ۲۶۸

۱۱-۷-۱ *مدارهای Fanout Free* ۲۶۸

۱۱-۷-۲ *مدارهای Non-Fanout Free* ۲۶۸

۱۱-۷-۳ *مدارهای دارای افزونگی* ۲۷۰

۱۱-۷-۴ *مدارهای با چند خروجی* ۲۷۰

۱۱-۷-۵ *تفاضل بولی* ۲۷۲

۱۱-۷-۶ *بسط شانون* ۲۷۶

۱۱-۷-۷ *خط نرمال* ۲۸۳

۱۱-۸ آزمون مدارهای ترتیبی همگام ۲۸۴

۱۱-۸-۱ *دنبالهٔ اولیه* ۲۸۴

۱۱-۸-۲ *دنبالهٔ انتقال* ۲۸۵

۱۱-۸-۳ *دنبالهٔ خانگی* ۲۸۶

۱۱-۸-۴ *دنبالهٔ تشخیص* ۲۸۷

۱۱-۹ طراحی برای قابلیت آزمون پذیری ۲۸۹

۱۱-۱۰ تمرین ۲۹۶

۱۲ مراجع ۳۰۰

نسخه پیش‌نویس Draft

نسخه پیشنویس
Draft

فصل ۱

مقدمه‌ای بر مدارهای منطقی

نسخه پیش‌نویس
Draft

۱-۱ مقدمه

در سال ۱۸۴۹، جورج بول فرآیند استدلال و تفکر منطقی را به صورت جبری فرمول‌بندی کرد. این فرمول‌بندی به جبر بول مشهور شد که در ادامه به اختصار مورد بررسی قرار خواهد گرفت.

پایه‌ای‌ترین توصیف فرمول‌بندی جبر بول بر اساس مفاهیم نظریهٔ مجموعه‌ها بیان می‌شود. در نظریهٔ مجموعه‌ها، جبر بول به صورت یک شبکهٔ متمم گسترده تعریف می‌گردد. خلاصه‌ای از این تعریف، به صورت یک دسته اصل موضوع، که عناصر و خواص اصلی جبر بول را توصیف می‌کنند، ارائه می‌شود.

- **اصل موضوع ۱**

تعریف- جبر بول یک دستگاه جبری بسته، مشتمل بر مجموعه حداقل دو عضوی K و دو عملگر "." و "+" است؛ یعنی برای هر a و b متعلق به مجموعهٔ K ، a.b عضوی از K و a+b نیز عضوی از K است. (عملگر "+" را OR و عملگر "." را AND می‌نامند).

- **اصل موضوع ۲**

وجود عضو خنثی : عناصر ۱ و ۰ به ترتیب برای عملگرهای "+" و "." عضو خنثی هستند.

- **اصل موضوع ۳**

جابه‌جاپذیری[1] "+" و "." - برای هر a وb وc متعلق به K داریم:

الف) $a+b=b+a$

ب) $a.b=b.a$

- **اصل موضوع ۴**

شرکت پذیری[2] عملگرهای "+" و "." - برای هر a وb وc متعلق به K داریم:

الف) $a+(b+c)=(a+b)+c$

ب) $a.(b.c)=(a.b).c$

[1] Commutativity
[2] Associativity

نسخه پیش‌نویس Draft

- **اصل موضوع ۵**

توزیع پذیری[1] "+" روی "." و "." روی "+" - برای هر a وb وc متعلق به K داریم:

الف) $a.(b+c)=(a.b)+(a.c)$

ب) $a+(b.c)=(a+b).(a+c)$

- **اصل موضوع ۶**

وجود متمم[2] : برای هر a متعلق به K داریم:

الف) $a.\bar{a}=0$

ب) $a+\bar{a}=1$

بر اساس اصول این موضوعات، قضایای نوشته شده در جدول ۱-۱ به دست می‌آیند.

جدول ۱-۱- اصول موضوعات و قضایای جبر بول

قضیه	همزاد
$a+a=a$	$a.a=a$
$a+1=1$	$a.0=0$
$a+ab=a$	$a.(a+b)=a$
$a+\bar{a}b=a+b$	$a.(\bar{a}+b)=ab$
$\bar{\bar{a}}=a$	
$ab+a\bar{b}=a$	$(a+b).(a+\bar{b})=a$
$ab+a\bar{b}c=ab+ac$	$(a+b).(a+\bar{b}+c)=(a+b).(a+c)$
$\overline{a+b}=\bar{a}.\bar{b}$	$\overline{a.b}=\bar{a}+\bar{b}$
$ab+\bar{a}c+bc=ab+\bar{a}c$	$(a+b).(\bar{a}+c).(b+c)=(a+b).(\bar{a}+c)$
$f(x_1,x_2,...,x_n)=f(1,x_2,...,x_n)+\overline{x_1}\,f(0,x_2,...,x_n)$	
$f(x_1,x_2,...,x_n)=[x_1+f(0,x_2,...,x_n)][\overline{x_1}+f(1,x_2,...,x_n)]$	

[1] Distrubitivity
[2] Complement

نسخه پیش نویس Draft

توابع کلیدی[1]

اصول موضوعات و قضایای جبر بول بیان شدند. توابع کلیدی، مفهوم متناظر تابع در جبر کلیدی هستند و به این صورت تعریف می‌گردند: $x_1, x_2, \ldots, x_n$ را به صورت نمادهایی موسوم به متغیر فرض می‌شوند، که هر کدام می‌تواند برابر با ۰ یا ۱ باشد. $F(x_1, x_2, \ldots, x_n)$ به صورت تابعی کلیدی از متغیرهای $x_1, x_2, \ldots, x_n$ بیان می‌شود.

جدول درستی

هر تابع کلیدی را می‌توان با عبارتهای کلیدی متفاوت و هم ارز نشان داد. اگر تابع کلیدی، به ازای تمام ترکیبهای ممکن ورودی محاسبه شود و به صورت جدولی ارائه گردد، نمایش یکتایی از تابع موسوم به جدول درستی حاصل می‌شود.

در جدول ۱-۲، جداول درستی ارائه شده به ترتیب عملیات پایه‌ای AND ، OR و متمم‌گیری را در الف، ب و ج نشان می‌دهند.

جدول ۱-۲ - جدولهای درستی عملیات پایه‌ای به ترتیب : AND ، OR و متمم‌گیری

(الف)

a	b	$f(a) = a.b$
0	0	0
0	1	0
1	0	0
1	1	1

(ب)

a	b	$f(a) = a + b$
0	0	0
0	1	1
1	0	1
1	1	1

(ج)

a	$f(a) = \bar{a}$
0	1
1	0

شکلهای متعارف[2]

شکلهای متعارف توابع کلیدی، شکلهای SOP(Sum of Products) و POS(Product of Sums) با مشخصات ویژه هستند. شکلهای متعارف SOP و POS برای هر تابع یکتاست.

[1] Switching Functions
[2] Algebraic Forms

نسخه پیش‌نویس Draft

جملات مینیمم

در یک تابع n متغیره، اگر در جمله حاصلضربی، تمام n متغیر به شکل اصلی یا متمم وجود داشته باشند، آن را جمله مینیمم می‌خوانند. اگر تابع تنها به صورت مجموع جملات مینیمم بیان شود، تابع دارای شکل متعارف مجموع حاصلضربها (SOP) است.

برای مثال: $f(A,B,C)=\overline{A}B\overline{C}+AB\overline{C}+\overline{A}BC+ABC$ که شکل متعارف SOP تابع $f(A,B,C)$ ، متشکل از چهار جمله مینیمم است.

برای ساده سازی نوشتن شکل متعارف SOP معمولاً نماد خاصی به منظور نمایش جملات مینیمم به صورت کدهای دودویی n بیتی به کار می‌رود. هر بیت، یکی از متغیرهای جمله مینیمم را به شرح زیر نشان می‌دهد:

- متغیرهای وارون نشده : ۱
- متغیرهای وارون شده : ۰

به عنوان مثال در جملهٔ $\overline{A}B\overline{C}$ ، متغیرهای اول و سوم وارون شده‌اند، لذا کد دودویی معادل این جمله برابر با ۰۱۰ است؛ به عبارت دیگر این جمله را می‌توان با عدد معادل دهدهی ۲ نشان داد.

جملات ماکزیمم

اگر یک جمله حاصل‌جمعی n متغیره، تمام n متغیر را به شکل وارون شده یا نشده در بر داشته باشد، آن را جمله ماکزیمم می‌نامیم. اگر تابع به صورت حاصلضرب جملات حاصلجمعی بیان شود، می‌گوییم تابع دارای شکل متعارف حاصلجمعی (POS) است. برای مثال تابع زیر به صورت شکل متعارف POS است و از چهار جمله ماکزیمم تشکیل شده است.

$$f(A,B,C)=(A+B+C)(A+B+\overline{C})(\overline{A}+B+C)(\overline{A}+B+C)$$

برای جملات ماکزیمم نیز مانند جملات مینیمم نماد خاصی به کار برده می‌شود، ولی با یک تفاوت مهم، کدگذاری به صورت زیر تغییر می‌کند:

- متغیرهای وارون نشده : ۰
- متغیرهای وارون شده : ۱

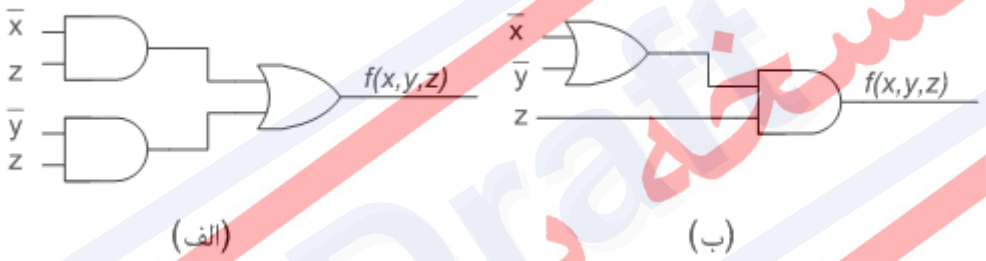
نسخه پیش‌نویس Draft

مثال۱) شکلهای مینیمم SOP و POS تابع $f(X,Y,Z)$ در ذیل را بیابید.

این عبارت را می‌توان به صورت زیر ساده کرد.

$$
\begin{aligned}
f(X,Y,Z) &= \overline{X}Y(Z+\overline{Y}X)+\overline{Y}Z \\
&= \overline{X}YZ+\overline{Y}Z+\overline{X}Y\overline{Y}X \\
&= \overline{X}YZ+\overline{Y}Z \\
&= \overline{X}Z+\overline{Y}Z \\
&= Z(\overline{X}+\overline{Y})
\end{aligned}
$$

دو عبارت آخر به ترتیب شکلهای مینیمم SOP و POS هستند. برای ساخت مدار تابع f با استفاده از شکل مینیمم SOP ، به دو گیت AND دو ورودی و یک گیت OR دو ورودی نیاز است در حالیکه شکل POS تنها به یک گیت AND دو ورودی و یک گیت OR دو ورودی نیاز دارد. (شکل ۱-۱الف و ب)



شکل ۱-۱- (الف) نمایش SOP تابع $f(X,Y,Z)$ (ب) نمایش POS تابع $f(X,Y,Z)$

۱-۲ مدارهای ترکیبی

به منظور ساخت مدارهای ترکیبی یک تابع، ابتدا آن را به صورت مجموعه‌ای از معادلات کلیدی بیان می‌کنند. سپس این معادلات را به وسیلهٔ گیت‌ها، مدارهای منطقی برنامه‌پذیر (PLD) و دیگر عناصر مدارهای منطقی می‌سازند. در تحلیل یک مدار منطقی، فرآیند عکس طی می‌شود. سخت افزار یک مدار دیجیتال داده می‌شود. سپس توصیف عملکرد آن به صورت عبارتهای منطقی، جدول درستی، نمودار زمانبندی یا هر روش دیگری خواسته می‌شود. هدف از تحلیل مدار، تعیین رفتار یک مدار منطقی، تأیید تطابق رفتار مدار با آنچه که از آن انتظار می‌رود، ایجاد زمینه‌های لازم برای تبدیل مدار به شکلهای دیگر، کاهش تعداد گیت‌های به کار رفته در آن، یا ساخت آن با عناصر دیگر است.

نسخه پیش‌نویس Draft

۱-۲-۱ روش جبری

مدارها را می‌توان برای انجام عملیات مشخصی در یک سیستم محاسباتی دیجیتال به کار برد. هر شبکهٔ کلیدی را می‌توان به طور کامل با یک عبارت منطقی بیان نمود، سپس تمام توان جبر کلیدی را صرف کار بر روی عبارت حاصل و تبدیل آن به شکل مطلوب کرد.

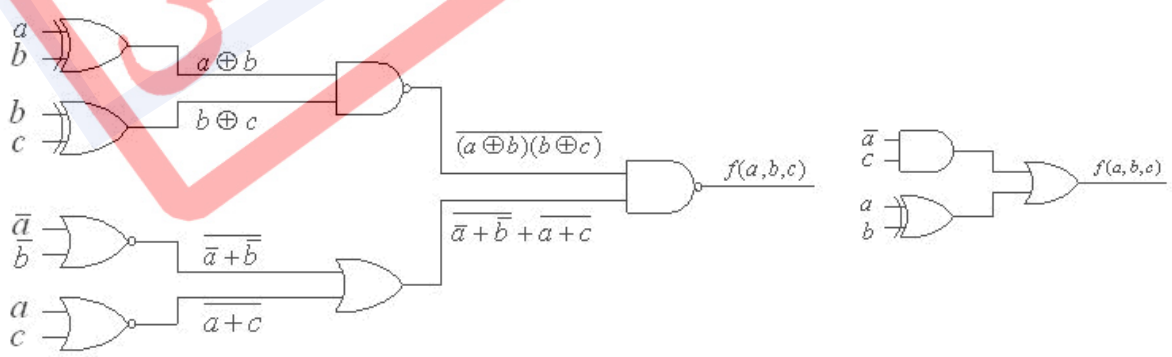
مثال ۲) برای تابع f یک عبارت منطقی و یک مدار ساده شده به دست آورید.

$$f(a,b,c)=\overline{\overline{(a\oplus b)(b\oplus c)}.(\overline{\bar{a}+\bar{b}}+\overline{a+c})}$$

به کمک محاسبات زیر، تابع داده شده را ساده می‌کنیم.

$$\begin{aligned}
f(a,b,c)&=\overline{\overline{(a\oplus b)(b\oplus c)}.(\overline{\bar{a}+\bar{b}}+\overline{a+c})}\\
&=\overline{\overline{(a\oplus b)(b\oplus c)}}+\overline{\overline{\bar{a}+\bar{b}}+\overline{a+c}}\\
&=(a\oplus b)(b\oplus c)+(\bar{a}+\bar{b})(a+c)\\
&=(a\bar{b}+\bar{a}b)(b\bar{c}+\bar{b}c)+(\bar{a}+\bar{b})(a+c)\\
&=a\bar{b}b\bar{c}+a\bar{b}\bar{b}c+\bar{a}b\bar{c}+\bar{a}b\bar{b}c+a\bar{a}+\bar{a}c+a\bar{b}+\bar{b}c\\
&=a\bar{b}c+\bar{a}b\bar{c}+\bar{a}c+a\bar{b}+\bar{b}c\\
&=\bar{a}b\bar{c}+\bar{a}c+a\bar{b}+\bar{b}c\\
&=\bar{a}b\bar{c}+\bar{a}c+a\bar{b}\\
&=\bar{a}b+\bar{a}c+a\bar{b}\\
&=\bar{a}c+a\oplus b
\end{aligned}$$

عبارت حاصل شده را می‌توان به صورت شبکه شکل ۱-۲ در ذیل پیاده‌سازی کرد.



شکل ۱-۲ - شبکه‌های معادل حاصل از تابع $f(a,b,c)$

نسخه پیش‌نویس Draft

۱-۲-۲ روش جدول درستی

در مثال قبل، می‌توان جدول درستی تابع را با استفاده از جملات نتیجه شده از ساده‌سازی تابع اصلی، بدست آورد:

جدول ۱-۳- جدول درستی مربوط به f(a,b,c) مثال ۱-۲

$b\ c\ a$	$\bar{a}c$	$a \oplus b$	$f(a,b,c)$
0 0 0	0	0	0
0 0 1	1	0	1
0 1 0	0	1	1
0 1 1	1	1	1
1 0 0	0	1	1
1 0 1	0	1	1
1 1 0	0	0	0
1 1 1	0	0	0

۱-۳ روشهای ساده‌سازی

به منظور مینیمم کردن و ساده‌سازی جملات و عبارات توابع منطقی، از اصول موضوعات و قضایای جبر بول استفاده می‌شود. به منظور بهره‌گیری سازمان یافته از این اصول و قضایا، الگوریتمهای متفاوتی وجود دارد. روشهایی که در ادامه بررسی می‌گردند، روش‌هایی شهودی هستند. بدین معنا که با استفاده از اطلاعات برگرفته شده از مسئله، راه حلی یافته می‌شود. در این روش‌ها، هنگامیکه گزینش بهینه‌ای هویدا نیست، گزینشها به صورت کم و بیش دلخواه انجام می‌شود. به همین خاطر، روشهای شهودی دستیابی به جواب مینیمم را تضمین نکرده و عموماً پاسخ‌های آنها غیر بهینه می‌باشند؛ گرچه در تعداد زیادی از موارد جوابهای بهینه را به دست می‌دهند.

در کنار روش‌های شهودی، الگوریتم‌هایی دیگری نیز موجوداند که دستیابی به جواب بهینه را تضمین می‌کنند. ولی استفاده از این روشها اغلب بسیار پیچیده‌تر و مشکل‌تر از روشهای شهودی است. به همین دلیل، بسیاری از طراحان به بهره‌گیری از روشهای شهودی بسنده کرده و سادگی را به بهینگی ترجیه می‌دهند.

نسخه پیش‌نویس Draft

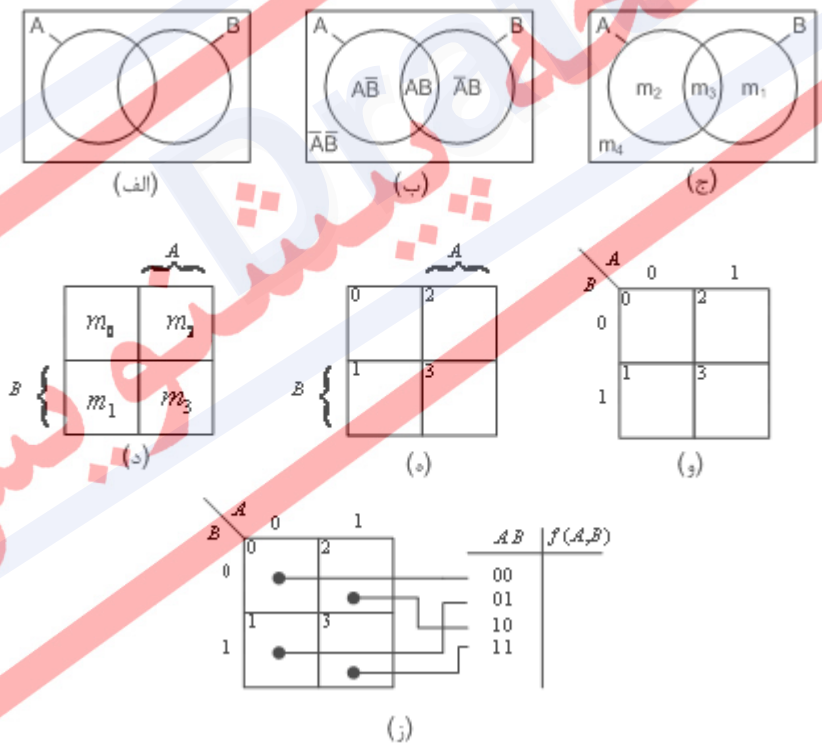
۱-۳-۱ جدول کارنو[1]

ساده سازی توابع منطقی با استفاده از جبر بول در بهترین حالت کار دشواری می‌باشد. در این روش، طریق مشخصی برای اجرا وجود ندارد و عمدتاً می‌بایست بر اساس شهود و تجربه‌های قبلی عمل نمود. جدول کارنو ابزاری است که در حل توابع تا شش متغیر کارآیی دارد.

۱-۳-۲ رابطه نمودار ون و جدول درستی

جدول کارنو چیزی نیست جز تعمیم مفاهیم جدول درستی، نمودار ون[2] و جمله مینیمم. برای نمایاندن این تعمیم به صورت صریح، نحوهٔ به دست آوردن جدول کارنو از نمودار ون تشریح می‌شود.

در نمودار ون شکل ۱-۳-الف، دو مجموعهٔ A وB به همراه افراز مربوطه بر روی مجموعهٔ مرجع، نشان داده شده‌اند. شکل ۱-۳- ب نشان می‌دهد که هر بخش مجزای نمودار ون با یکی از اشتراکات $AB, \overline{A}B, A\overline{B}, \overline{A}\,\overline{B}$ متناظر است. در شکل ۳-ج، بخشهای مجزای نمودار ون با جملات مینیمم m_3, m_2 ,m_1 ,m_0 مشخص شده‌اند.



شکل ۱-۳ - نمودار ون و جدول کارنوی متناظر (برای دو متغیر)

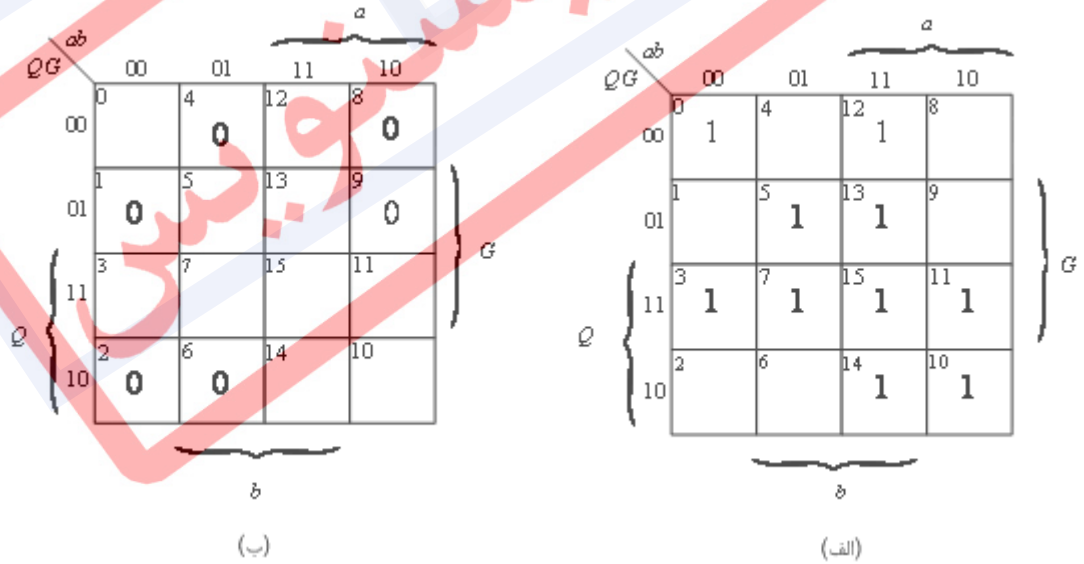
[1] Karnaugh Map
[2] Venn Diagram

این جملات در شکل ۱-۳-د، در یک جدول قرار گرفته‌اند به گونه‌ای که سطوحی که در شکل ۱-۳- ج مجاور هستند، در شکل ۱-۳-د نیز مجاور می‌باشند. اکنون نیمی از نمودار به متغیر A و نیم دیگر به متغیر B اختصاص دارد. در شکل ۱-۳-ه ، تنها زیر نویس هر کدام از جملات متناظر، نوشته شده است. شکل ۱-۳-ه و شکل ۱-۳-و، دو گونهٔ پیاده‌سازی جدول کارنو هستند. در این شکل تناظر خانه های جدول با متغیرها، مثلاً ۰ برای $\overline{A}$ و ۱ برای A ، مشخص شده است. جدول کارنو نمایش تصویری جدول درستی است، بنابراین تناظر یک به یک بین این دو نیز برقرار است. در جدول درستی هر جمله مینیمم یک ردیف متناظر دارد و در جدول کارنو یک خانهٔ متناظر. این رابطه در شکل ۱-۳-ز نشان داده شده است. به نحو مشابه، در صورت استفاده از جملات ماکزیمم نیز تناظر یک به یک بین خانه‌های جدول کارنو و ردیفهای جدول درستی برقرار است.

مثال ۳) تابع زیر را بر روی جدول کارنو رسم کنید.

$$f(a,b,Q,G)=\sum m(0,3,5,7,10,11,12,13,14,15)=\prod M(1,2,4,6,8,9)$$

شکل ۱-۴-الف، نمایش مینیمم این تابع را بر روی جدول کارنو نشان می‌دهد. شکل ۱-۴-ب، تابع را به صورت ضرب جملات ماکزیمم نشان می‌دهد. اگر ترتیب متغیرها عوض شود، اعداد جملات مینیمم و ماکزیمم نیز تغیر می‌کند. بنابراین ترتیب متغیرها در تابع، تعیین کنندهٔ ترتیب متغیرها در جدول کارنو است.



شکل ۱-۴ - جدول‌های کارنو برای $f(a,b,Q,G)$. (الف) جملات مینیمم. (ب) جملات ماکزیمم

مثال ۴) جملات مینیمم تابع زیر را تعیین کنید.

$$f(A,B,C,D)=(\overline{A}+\overline{B})(\overline{A}+C+\overline{D})(\overline{B}+\overline{C}+\overline{D})$$

ابتدا تابع را متمم کرده و سپس قضیهٔ دمورگان را اعمال می‌کنیم.

$$\overline{f}(A,B,C,D)=\overline{(\overline{A}+\overline{B})(\overline{A}+C+\overline{D})(\overline{B}+\overline{C}+\overline{D})}$$
$$=(\overline{\overline{A}+\overline{B}})+(\overline{\overline{A}+C+\overline{D}})+(\overline{\overline{B}+\overline{C}+\overline{D}})$$
$$=AB+A\overline{C}D+BCD$$

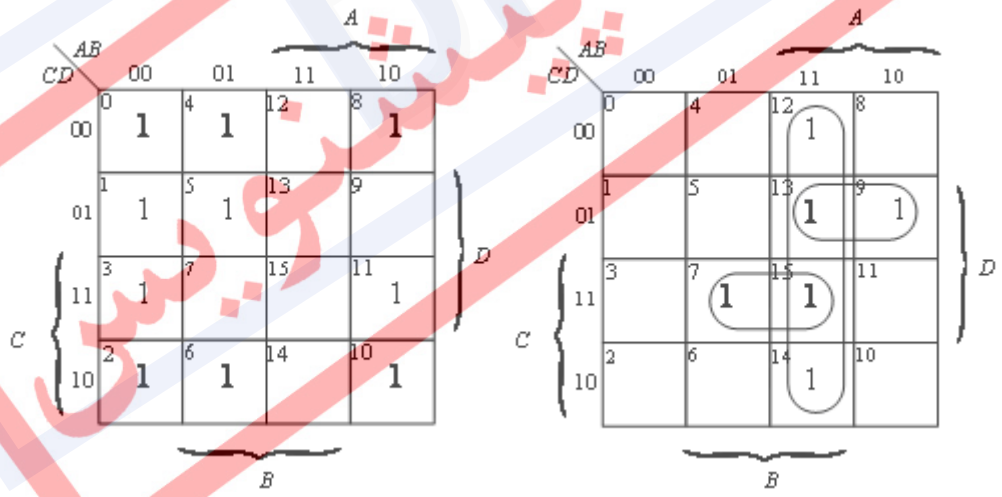
شکل ۱-۵-الف $\overline{f}(A,B,C,D)=AB+A\overline{C}D+BCD$ را بر روی جدول کارنو نشان می‌دهد. با توجه به این جدول کارنو می‌توان نوشت:

$$\overline{f}(A,B,C,D)=\sum m(7,9,12,13,14,15)$$

چون صفرهای جدول کارنو شکل ۱-۵-الف نشان دهنده تابع $f(A,B,C,D)$ است، با تأمل در این جدول می‌نویسیم:

$$f(A,B,C,D)=\sum m(0,1,2,3,4,5,6,8,10,11)$$

که در جدول کارنوی شکل ۱-۵-ب نشان داده شده است.



(ب) (الف)

شکل ۱-۵- (الف) جدول کارنوی متمم تابع f . (ب) جدول کارنوی f

نسخه پیش‌نویس Draft

۱-۳-۳ خطوط کلی روش ساده سازی با استفاده از جدول کارنو

در ساده کردن یک تابع به کمک جدول کارنو، باید به پنج نکتهٔ مهم ذیل توجه کامل شود:

۱. در جدول کارنوی دو متغیره، هر خانهٔ جدول (جملهٔ مینیمم) دو خانهٔ مجاور دارد، هر خانهٔ جدول کارنوی سه متغیره سه خانهٔ مجاور دارد، و به همین ترتیب. در حالت کلی، هر خانهٔ یک جدول کارنوی n متغیره، n خانهٔ مجاور دارد و هر زوج خانه مجاور تنها در یک متغیر متفاوتند.

۲. در ترکیب جملات (خانه‌ها)، همیشه تعداد خانه‌های قرار گرفته در یک دسته، توانی از ۲ است، یعنی ۲ خانه، ۴ خانه، ۸ خانه، و غیره. در یک دستهٔ دوتایی یک متغیر حذف می‌شود، در یک دسته چهارتایی دو متغیر و به همین ترتیب. به طور کلی در یک دسته ۲^n تایی، n متغیر حذف می‌شود.

۳. تا آنجا که امکان پذیر است باید دسته‌های بزرگتری را برگزید؛ هرچه دسته بزرگتر باشد، جملهٔ حاصلضربی حاصل متغیر کمتری دارد.

۴. برای پوشاندن تمام جملات مینیمم، باید از کمترین تعداد دسته ممکن استفاده کرد. وقتی یک جمله مینیمم پوشیده می‌شود که حداقل در یک دسته قرار بگیرد. هرچه تعداد گروه‌ها کمتر باشد، تعداد جملات حاصلضربی حاصل نیز کمتر می‌شود. هر جملهٔ مینیمم را می‌توان به تعداد دلخواه در دسته‌ها به کاربرد. از سوی دیگر باید توجه داشت که هر جمله باید حداقل یک بار به کار رود. هنگامی که تمام جملات مینیمم حداقل یک بار به کار رفتند، عملیات ساده‌سازی به اتمام می‌رسد.

۵. در ترکیب کردن خانه‌ها می‌بایست همواره از خانه‌هایی شروع کرد که کمترین همســـایه‌ها را دارند. برای جمله‌هایی که دارای چند جملهٔ مجاور هستند، تعداد دسته بندیهای بیشتری امکان پذیر است، لذا می‌توان دسته بندی آنها را به تعویق انداخت.

الگوریتم بدست آوردن SOP مینیمم به کمک جدول کارنو

۶. تعداد همسایه های هر جمله مینیمم را در جدول کارنو تعیین کنید.

۷. جملهٔ مینیمم با کمترین همسایه را برگزینید. در صورت وجود چند جملهٔ مینیمم مشابه، یکی را به دلخواه انتخاب کنید.

۸. برای هر جملهٔ مینیمم یک دسته‌بندی انتخاب کنید به صورتیکه آن را بپوشاند. اگر چند دسته‌بندی وجود دارد، گروهی را برگزینید که بیشترین جمله‌های مینیمم پوشیده نشده را بپوشاند.

۹. گامهای ۲و۳ را تکرار کنید تا تمام جملات مینیمم پوشانده شوند.

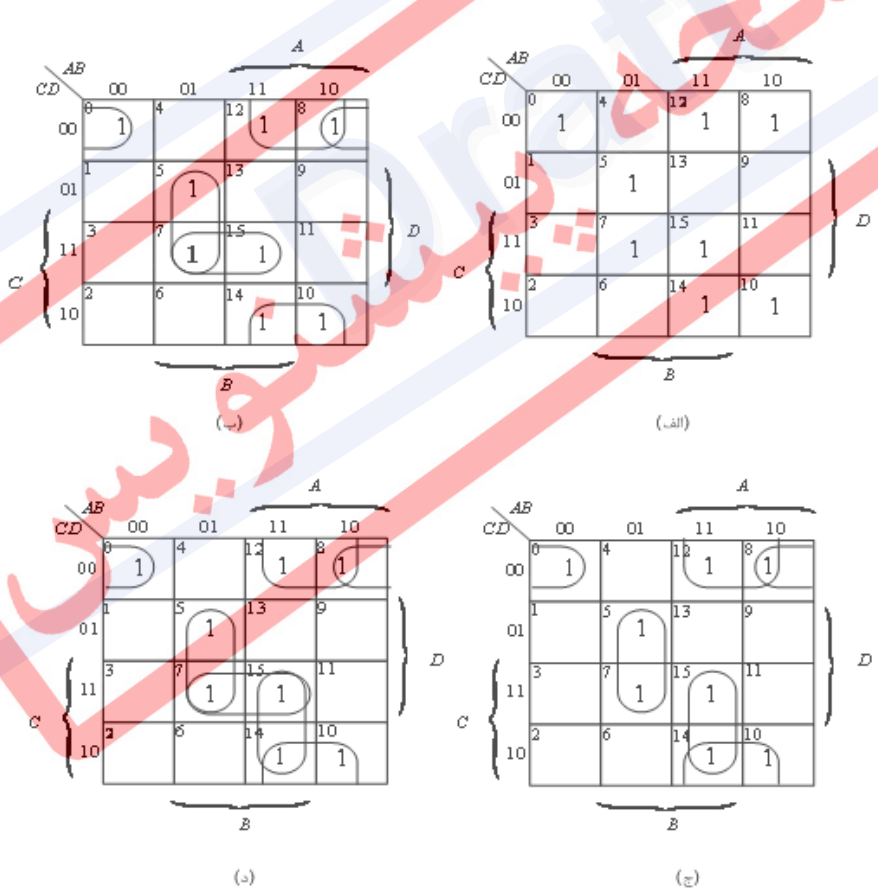
نسخه پیش‌نویس Draft

مثال ۵) تابع زیر را با استفاده از جدول کارنو ساده کنید.

$$f(A,B,C,D)=\sum m(0,5,7,8,10,12,14,15)$$

این تابع در جدول کارنوی شکل ۱-۶-الف نشان داده شده است.

با استفاده از الگوریتم ذکرشده، ابتدا جملهٔ مینیمم m_0 را که تنها دارای یک همسایه است انتخاب می‌کنیم. و دسته‌بندی {۰,۸} را می‌سازیم. m_5 نیز یک همسایه دارد و با دسته‌بندی {۷,۵} پوشانده می‌شود. حال جملات مینیمم دارای دو همسایه را در نظر می‌گیریم. m_{10} را برمی‌گزینیم و دسته‌بندی {۱۴,۱۲,۱۰,۸} را می‌سازیم. تنها جملهٔ مینیمم پوشانده نشده m_{15} است که می‌توان آن را نیز با دسته‌بندی‌های {۱۵,۷} یا {۱۵,۱۴} پوشاند. چون هر دوی این انتخاب‌ها تعداد یکسانی جملهٔ مینیمم را می‌پوشانند و در نتیجه از لحاظ پیچیدگی یکسان هستند، لذا می‌توانیم به دلخواه یکی از آنها را برگزینیم. شکل ۱-۶-ب پوشش مینیمم حاصل از گزینش {۱۵,۷} ، و شکل ۱-۶-ج پوشش مینیمم حاصل {۱۵,۱۴} را نشان می‌دهد. در شکل ۱-۶-د تمام دسته‌بندیها مشخص شده‌اند.



شکل ۱-۶- جدول کارنوی $f(A,B,C,D)$ (الف) تابع. (ب) پوشش مینیمم ۱. (ج) پوشش مینیمم ۲. (د) تمام پوشش‌ها

اکنون با توجه به پوشش مینیمم شکل ۱-۵-ب، می‌توان شکل SOP تابع $f(A,B,C,D)$ را به صورت زیر نوشت:

$$f(A,B,C,D)=\overline{B}\overline{C}\overline{D}+\overline{A}BD+A\overline{D}+BCD$$

و با توجه به پوشش مینیمم شکل ۱-۵-ج خواهیم داشت:

$$f(A,B,C,D)=\overline{B}\overline{C}\overline{D}+\overline{A}BD+A\overline{D}+ABC$$

چون هر دو عبارت SOP از لحاظ تعداد جملات و متغیرها برابرند، لذا هر دو عبارت SOP عبارتهای مینیمم تابع هستند.

عبارتهای POS با استفاده از جدول کارنو

برای ساده کردن توابع POS نیز می‌توان روش مشابه توابع SOP را انجام داد. به علاوه تمام روشهایی که برای ترکیب جملات مینیمم بیان شده‌اند، برای ترکیب جملات ماکزیمم و دستیابی به عبارت POS مینیمم نیز قابل اعمال هستند..

الگوریتم بدست آوردن POS مینیمم به کمک جدول کارنو

۱. تعداد همسایه های هر جملهٔ ماکزیمم را در جدول کارنو تعیین کنید.

۲. جملهٔ ماکزیمم با کمترین همسایه را برگزینید. در صورت وجود چند جملهٔ ماکزیمم مشابه، یکی را به دلخواه انتخاب کنید.

۳. برای هر جملهٔ ماکزیمم یک دسته‌بندی انتخاب کنید، و به این ترتیب جملهٔ ماکزیمم را بپوشانید. اگر چند دسته‌بندی وجود دارد، گروهی را برگزینید که بیشترین جمله‌های ماکزیمم پوشیده نشده را می‌پوشاند.

۴. گامهای ۲و۳ را تکرار کنید تا تمام جملات ماکزیمم پوشانده شود.

مثال ۶) عبارتهای POS و SOP مینیمم تابع زیر را بیابید:

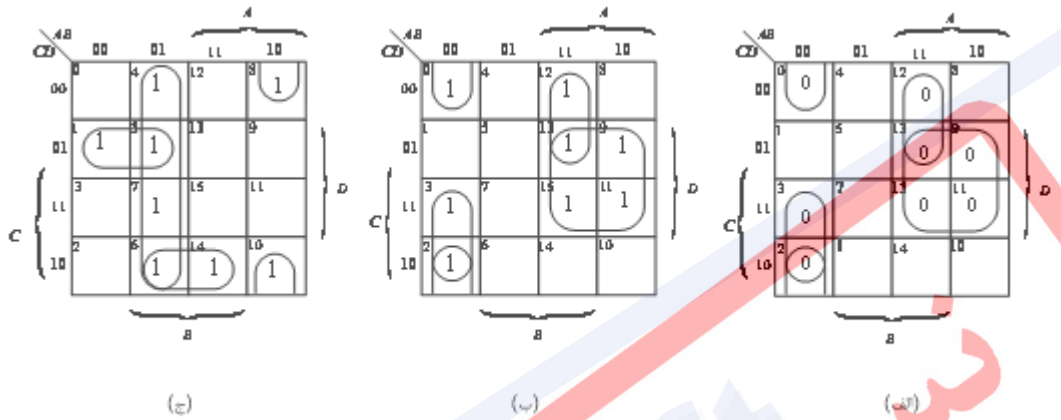
$$f(A,B,C,D)=\prod M(0,2,3,9,11,12,13,15)$$

برای یافتن عبارت POS مینیمم باید جملات ماکزیمم تابع را به صورت شکل ۱-۷-الف در جدول کارنو درج کنیم. با توجه به روش گفته شده، دسته‌بندی {۲،۰} را برای پوشش M_2، {۱۳،۱۲} را برای

نسخه پیش‌نویس Draft

پوشش M_{12} ، و {۱۵,۱۳,۱۱,۹} را برای پوشش M_9 برمی‌گزینیم. به این ترتیب تنها M_3 می‌ماند که می‌توان آن را با {۱۱,۳} یا {۳,۲} پوشش داد. چون هر دو دارای تعداد مساوی جمله هستند، {۳,۲} را برمی‌گزینیم. عبارت POS حاصل به صورت زیر نتیجه می‌شود..

$$f(A,B,C,D) = (A+B+C)(\overline{A}+\overline{B}+C)(\overline{A}+\overline{D})(A+B+\overline{C})$$



شکل ۱-۷- یافتن عبارت‌های POS و SOP یک تابع. (الف) جملات ماکزیمم f. (ب) جملات ماکزیمم متمم f. (ج) جملات مینیمم f .

همچنین می‌توان برای یافتن عبارت POS ، ابتدا تابع $\overline{f}$ را به صورت شکل ۱-۷-ب رسم کرد، و عبارت SOP تابع $\overline{f}$ را یافت.

$$\overline{f}(A,B,C,D) = \overline{A}\overline{B}\overline{C} + AB\overline{C} + AD + \overline{A}\overline{B}C$$

سپس با متمم کردن این عبارت، عبارت زیر حاصل می‌شود:

$$f(A,B,C,D) = \overline{\overline{A}\overline{B}\overline{C} + AB\overline{C} + AD + \overline{A}\overline{B}C}$$
$$= (\overline{\overline{A}\overline{B}\overline{C}})(\overline{AB\overline{C}})(\overline{AD})(\overline{\overline{A}\overline{B}C})$$
$$= (A+B+C)(\overline{A}+\overline{B}+C)(\overline{A}+\overline{D})(A+B+\overline{C})$$

برای یافتن عبارت SOP تابع $f(A,B,C,D)$ می‌بایست جملات مینیمم آن را به صورت شکل ۱-۷-ج رسم نمود. با استفاده از روش پوشش مینیمم، این شکل به دست می‌آید. با توجه به این پوشش خواهیم داشت:

$$f(A,B,C,D) = \overline{A}\overline{C}D + A\overline{B}\overline{D} + \overline{A}B + BC\overline{D}$$

نسخه پیش‌نویس

با توجه به مثال قبل، می‌توان نتیجه گرفت که با داشتن هر توصیفی از یک تابع، می‌توان عبارت SOP مینیمم و نیز عبارت POS مینیمم آن را یافت. برای یافتن عبارت SOP مینیمم، باید با جملات مینیمم کار کرد، لازم به ذکر است که برای یافتن عبارت POS مینیمم، می‌بایست با جملات ماکزیمم عمل نمود. با رسم جملات ماکزیمم یک تابع در جدول کارنو، مانند مثال قبل، به طور خودکار نقشهٔ جملات مینیمم نیز تولید می‌شود، و بر عکس. بنابراین انتخاب عبارتهای SOP یا POS برای تابع نباید بر اساس قالب ابتدایی تعریف تابع صورت گیرد.

در ادامه، تعدادی از اصطلاحات موجود در جداول کارنو بیان می‌شود. (با توجه به جدول ۱-۴)

جدول ۱-۴- یک نمونه از جدول کارنو

C \ AB	00	01	11	10
0	0	1	1	0
1	1	1	1	0

– **Implicant:** عباراتی که باعث یک شدن تابع می‌شوند. این عبارات عضو {0, 1, x} هستند. (x نشانه حالت بی تفاوت است).

– **Minterm:** یک Im است که عضو مجموعهٔ $\{0, 1\}^m$ می‌باشد. مینترم‌ها معمولاً بصورت SOP (Sum-of-Product) نشان داده می‌شوند.

به عنوان مثال در جدول بالا، ۱۱ مینترم وجود دارد:

- *5 single minterms* $\overline{A}BC, \overline{A}B\overline{C}, \overline{A}\overline{B}C, AB\overline{C}, ABC$
- *5 Group of 2 minterms* $\overline{A}B, AB, \overline{A}C, B\overline{C}, BC$
- *1 Group of 4 minterms* $= B$

– **Prime Implicant:** یک Im که بوسیله Im دیگر پوشیده نشود.

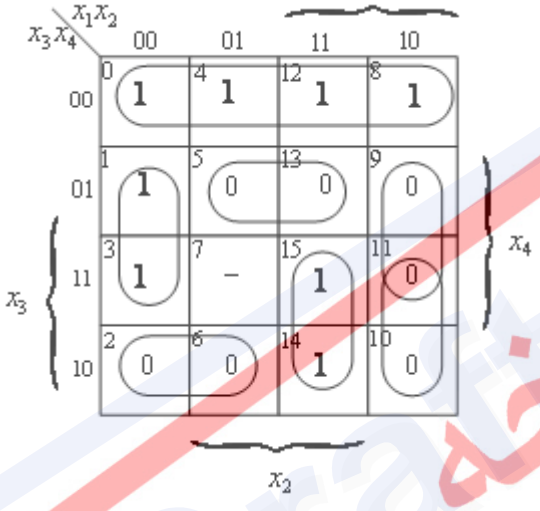
بطور نمونه، در مثال قبل $\overline{A}C, B$ یک P.Im هستند.

– **Essential Prime Implicant:** یک P.Im که حداقل یک مینترم را بپوشاند به شرط آنکه این مینترم توسط P.Im دیگر پوشیده نشود. بطور نمونه، در مثال قبل $\overline{A}C, B$ یک E.P.Im هستند.

نسخه پیش‌نویس Draft

– **Maxterm**: یک Im که عضو مجموعهٔ $\{0, 1\}^m$ باشد. ماکسترم بصورت POS (Product of Sum) نمایش داده می‌شود.

مثال ۷) تابع f را به صورت POS و SOP نمایش دهید. این تابع به صورت جدول کارنوی زیر داده شده است.



شکل ۱-۸ - جدول کارنوی مربوط به مثال ۱-۷

داریم:

$$f(SOP) = \bar{x}_3\bar{x}_4 + \bar{x}_1\bar{x}_2x_4 + x_1x_2x_3$$
$$f(POS) = (\bar{x}_2 + x_3 + \bar{x}_4)(\bar{x}_1 + x_2 + \bar{x}_4)(\bar{x}_1 + x_2 + \bar{x}_3)(x_1 + \bar{x}_3 + x_4)$$

نکته) برای تبدیل این دو تابع به سطح گیت (ساخت مدار با استفاده از گیت‌ها)، می‌بایست یک مرحلهٔ دیگر تحت عنوان نگاشت روی کتابخانه[1] انجام شود. برای سادگی، فعلاً فرض می‌شود که تمام گیت‌ها در کتابخانه موجود می‌باشند.

[1] Library Mapping

نسخه پیش‌نویس Draft

ابتدا فرض می‌کنیم در کتابخانه، همهٔ انواع گیت‌های منطقی وجود دارند (انواع گیت با تعداد ورودی مختلف). هزینهٔ پیاده‌سازی تابع را در هر دو صورت با یکدیگر مقایسه می‌کنیم.

جدول ۱-۵- مقایسه هزینه پیاده‌سازی تابع در حالت SOP و POS

	AND	OR	NOT	Σgates	$\Sigma(gate)_i*w_i$
SOP	3	1	4	4	$3\times1 + 1\times2 = 5$
POS	1	4	4	5	$1\times1 + 2\times4 = 9$

* هزینه گیت AND یک و گیت OR دو فرض شده است.

w_i برابر با وزن گیت است که در این مثال برابر تعداد ورودی‌های گیت می‌باشد.

در محاسبهٔ هزینه، از گیت NOT صرفنظر شده است. زیرا به تعداد یکسان در هر دو روش بکار رفته است.

۱-۳-۴ روش جدول بندی کویین – مک کلاسکی برای ساده کردن توابع

روش کویین– مک کلاسکی (Q-M) یک رهیافت جدول بندی برای مینیمم کردن توابع بولی است. روش Q-M دو مزیت اصلی نسبت به جدول کارنو دارد. اول این که روشی سر راست بوده و به توانایی طراح در تشخیص الگوها بر روی جدول کارنو بستگی ندارد. دوم اینکه توابع با تعداد متغیرهای بیشتری را می‌توان با آن ساده کرد، در صورتیکه جدول کارنو عملاً به ۵ یا ۶ متغیر محدود می‌شود. در روش Q-M یک جستجوی خطی سامان یافته بر روی جملات مینیمم تابع صورت می‌گیرد. هدف از انجام این کار یافتن تمام ترکیبهای جملات مینیمم است که به صورت منطقی، مجاور خواهند بود. در ادامه نشان داده خواهدشد که این روش را می‌توان به توابع چند خروجی نیز تعمیم داد. روش Q-M از فهرست جملات مینیمم n متغیری شروع می‌شود و به ترتیب تمام دسته‌بندی‌های دارای n-1 متغیر، دارای n-2 متغیر، و ... را به دست می‌آورد، تا سرانجام تمام دسته‌بندی‌ها مشخص شوند. چهار گام این فرایند در زیر بیان شده و معنای دقیق هر گام طی مثال‌های بعد از آن روشن می‌شود.

گام۱ : در یک ستون، تمام جملات مینیمم تابعی را که باید ساده شود به صورت دودویی درج می‌کنیم. این جملات را بر حسب تعداد بیتهای ۱ نمایش دودویی‌شان دسته بندی می‌کنیم. سپس تمام جملاتی که دارای تعداد یکسانی عدد ۱ در نمایش دودویی‌شان هستند، در یک گروه قرار می‌گیرند. این

نسخه پیش‌نویس Draft

دسته بندی، تشخیص جملات مینیمم مجاور منطقی را ساده می‌کند، چون دو جملهٔ مینیمم، مجاور منطقی هستند به شرط آنکه تنها در یک حرف (بیت) تفاوت داشته باشند.

گام۲ : با جستجوی کامل، گروه‌های مجاور جملات مینیمم مجاور را تشخیص داده و آنها را در ستونی مشتمل بر دسته‌بندی‌های n-1 متغیره قرار می‌دهیم و جملات مینیمم ترکیب شده را علامت می‌زنیم. در نمایش دودویی هر دسته‌بندی جدید، به جای متغیر حذف شده خط تیره می‌گذاریم. این کار را برای ستون جدید تکرار کرده، با ترکیب دسته‌بندی‌های n-1 متغیره، ستونی از دسته‌بندی‌های n-2 متغیره می‌سازیم. برای ستونهای جدید نیز همین عمل را انجام داده تا اینکه دیگر هیچ دسته‌ای را نتوان ترکیب کرد. تمام جملاتی که در نهایت بدون علامت مانده باشند از دسته‌بندی اول هستند، زیرا در دستهٔ بزرگتری ادغام نشده‌اند. نتیجهٔ نهایی، فهرستی از دسته‌بندی‌های اول تابع است.

گام۳ : جدولی از دسته‌بندی‌های اول ترتیب می‌دهیم، به گونه‌ای که جملات مینیمم در جهت افقی و دسته‌بندی‌های اول در جهت عمودی آن درج شده باشد. هر جملهٔ مینیممی که توسط یک دسته‌بندی پوشانده شود با علامت * در محل برخورد سطر (شامل) و ستون (جمله مینیمم) مشخص می‌گردد.

گام۴: حداقل دسته‌بندی‌های اول برای پوشش جملات مینیمم تابع داده شده را انتخاب می‌کنیم.

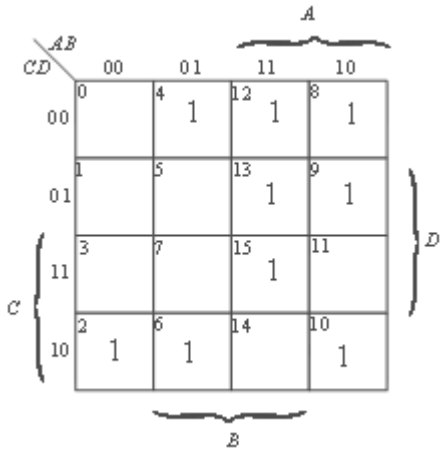
اکنون با یک مثال، این چهار گام را بررسی می‌کنیم.

مثال ۸) تابع زیر را به روش کویین- مک کلاسکی ساده کنید.

$$f(A,B,C,D)=\sum m(2,4,6,8,9,10,12,13,15)$$

شکل ۹ جدول کارنوی این تابع را نشان می‌دهد.

نسخه پیش‌نویس Draft



شکل ۱-۹ - جدول کارنوی تابع $f(A,B,C,D)$ مثال ۱-۸

گام۱. برای شروع به استفاده از روش Q-M ، جملات مینیمم را براساس تعداد ۱های نمایش دودویی شان دسته بندی می‌کنیم. این دسته بندی در جدول زیر نشان داده شده است:

جدول ۱-۶ – دسته‌بندی جملات مینیمم براساس تعداد ۱های نمایش دودویی در روش Q-M

	A D C B	شماره جملات مینیمم
گروه۱ (یک عدد ۱)	0 0 1 0	۲
	0 1 0 0	۴
	1 0 0 0	۸
گروه۲ (دو تا ۱)	0 1 1 0	۶
	1 0 0 1	۹
	1 0 1 0	۱۰
	1 1 0 0	۱۲
گروه۳ (سه تا ۱)	1 1 0 1	۱۳
گروه۴ (چهار تا ۱)	1 1 1 1	۱۵

نسخه پیش‌نویس Draft

گام۲. پس از تکمیل این جدول، جستجوی کاملی را برای یافتن تمام جملات مجاور منطقی به عمل می‌آوریم. بر اساس این روش جدول بعدی را که مشتمل بر سه فهرست از جملات مینیمم است، بدست می‌آوریم. در این روش دو جمله را می‌توان با هم ترکیب کرد در صورتیکه تنها در یک حرف تفاوت داشته باشند. پس در فهرست اول جملات دستهٔ اول را تنها با جملات دسته دوم می‌توان ترکیب کرد. پس از انجام تمام ترکیب‌های بین این دو دسته و وارد کردن نتایج در فهرست دوم، خطی می‌کشیم و جملات دستهٔ دوم و دستهٔ سوم را ترکیب می‌کنیم. این روند ساده را برای فهرست بعد نیز انجام می‌دهیم تا جدول ساده سازی کامل شود.

جدول ۱-۷ – ترکیب جملات مجاور در روش Q-M

فهرست ۱			فهرست ۲			فهرست ۳		
جملات مینیمم	$D\ C\ B\ A$		جملات مینیمم	$D\ C\ B\ A$		جملات مینیمم	$D\ C\ B\ A$	
۲	0 0 1 0	√	۲,۶	0 - 1 0	PI_2	۸,۹,۱۲,۱۳	1 - 0 -	PI_1
۴	0 1 0 0	√	۲,۱۰	- 0 1 0	PI_3			
۸	1 0 0 0	√	۴,۶	0 1 - 0	PI_3			
۶	0 1 1 0	√	۴,۱۲	- 1 0 0	PI_5			
۹	1 0 0 1	√	۸,۹	1 0 0 -	√			
۱۰	1 0 1 0	√	۸,۱۰	1 0 - 0	PI_6			
۱۲	1 1 0 0	√	۸,۱۲	1 - 0 0	√			
۱۳	1 1 0 1	√	۹,۱۳	1 - 0 1	√			
۱۵	1 1 1 1	√	۱۲,۱۳	1 1 0 -	√			
			۱۳,۱۵	1 1 - 1	PI_7			

اولین ردیف فهرست ۱ نشان می‌دهد که جملات مینیمم ۲ و ۶ ترکیب شده‌اند، زیرا تنها در یک حرف تفاوت داشتند. این تفاوت در متغیر B است، بنابراین در محل این حرف در ترکیب‌بندی جدید، یک علامت - گذاشته شده است که نشان دهندهٔ ترکیب جملات مینیمم ۲ و ۶ و حذف حرف B می‌باشد. درستی این ترکیب را می‌توان به سادگی نشان داد:

جملهٔ مینیمم۲ = $\overline{A}\overline{B}C\overline{D}$

جملهٔ مینیمم۶ = $\overline{A}BC\overline{D}$

$$\overline{A}\overline{B}C\overline{D}+\overline{A}BC\overline{D}=\overline{A}C\overline{D}\Rightarrow 0-10$$

نسخه پیش‌نویس Draft

هر جملهٔ مینیمم فهرست که با جملهٔ مینیمم دیگری ترکیب شده باشد، با علامت √ مشخص می‌شود؛ بدین مفهوم که آن جمله در دسته‌بندی بزرگتری قرار گرفته است. یک جمله می‌تواند بیش از یکبار ترکیب شود، ولی تنها یک علامت √ خواهد داشت.

پس از ساختن فهرست ۲ از فهرست ۱، جستجوی دیگری برای ترکیب جملات فهرست ۲ و ایجاد فهرست ۳ انجام می‌دهیم. در اینجاست که اهمیت مشخص کردن حروف حذف شده با خط تیره روشن می‌شود. همانند قبل تنها دو جمله‌ای از فهرست ۲ را می‌توان با هم ترکیب کرد که در یک حرف تفاوت داشته باشند، پس باید بین دو جمله حرف مشترکی حذف شده باشد، یعنی- در محل یکسانی قرار داشته باشد. در فهرست ۲، می‌توان جملات مینیمم ۸،۱۲ و ۹،۱۳ و همچنین ۸،۹ و ۱۲،۱۳ را ترکیب کرده و ترکیب جدید ۱۳،۱۲،۹،۸ فهرست ۳ را ساخت. بررسی فهرست ۲ نشان می‌دهد که ترکیبهای ۸،۱۲ و۹،۱۳ حرف حذف شدهٔ یکسانی دارند و در آنچه باقی مانده فقط در یک حرف متفاوت‌اند. برای ترکیبهای ۸،۹ و۱۲،۱۳ نیز چنین است. پس این چهار جمله در فهرست ۲ علامت خورده و در فهرست ۳ وارد می‌شوند. در این لحظه هیچ دو ترکیب دیگری از فهرست ۲ را نمی‌توان ترکیب کرد. تمام ترکیبهای باقی مانده، دسته‌بندی اول هستند و از PI_1 تا PI_7 نامگذاری شده‌اند. اکنون می‌توان تابع را به صورت مجموعه‌ای از دسته‌بندی‌های اول نوشت. ولی از آنجا که به دنبال مینیمم کردن تابع هستیم، لذا باید حداقل دسته‌بندی‌های اول را به کار برد.

یک روش مناسب برای امتحان درستی عملیات انجام شده این است که عدد جملات مینیمم ترکیب شده را از هم کم کنیم تا ببینیم متغیر درستی حذف شده است یا نه. مثلاً ترکیب ۴،۶ از فهرست ۲ نشان می‌دهد متغیر با وزن ۲=۴-۶ حذف شده است. در این مثال وزنهای ممکن ۸ ، ۴ ، ۲ و ۱ می‌باشند.

برای ترکیب ۱۳، ۱۲، ۹ ، ۸ که عبارت -۰-۱ را نتیجه می‌دهند، داریم:

۱) ۴=۸-۱۲ ۳) ۴=۸-۱۲

۲) ۴=۹-۱۳ ۴) ۱=۱۲-۱۳

پس باید متغیرهای با وزن ۱ و ۴ حذف شده باشند.

گام۳. برای تعیین کوچکترین تعداد دسته‌بندی‌های اول مورد نیاز برای ساختن تابع f، جدول زیر را ترتیب می‌دهیم:

نسخهٔ پیش‌نویس Draft

جدول ۱-۸- تعیین کوچکترین تعداد دسته‌بندی‌های اول مورد نیاز برای ساختن تابع f

شماره جملات / PI ها	2	4	6	√ 8	√ 9	10	√ 12	√ 13	√ 15
** PI_1				×	⊗		×	×	
PI_2	×		×						
PI_3	×					×			
PI_4		×	×						
PI_5		×					×		
PI_6				×		×			
** PI_7								×	⊗

دسته‌بندی‌های اول که دارای متغیر شاخص هستند علامت‌گذاری می‌شوند، مانند PI1 و PI7 .

گام۴. بررسی ستون های این جدول نشان می‌دهد که جملات مینیمم ۹ و ۱۵، تنها با یک دسته‌بندی پوشانده شده‌اند، و دور آنها دایره‌ای رسم شده است. پس PIهای ۱و ۷ را باید برگزید، بنابراین دسته‌بندی اول اصلی هستند. با انتخاب این دو دسته‌بندی اول، جملات مینیمم ۸ ،۱۲ ،۱۳ نیز پوشانده می‌شوند. پنج جملهٔ مینیمم پوشانده شده در بالای شماره‌هایشان علامتگذاری شده‌اند.

اکنون، مسئله برگزیدن حداقل دسته‌بندی‌های اول لازم برای پوشاندن سایر جملات مینیمم یعنی ۲، ۴، ۶ و ۱۰ در دستور کار است. برای این کار یک جدول مختص این دسته‌بندی‌های اول ترتیب می‌دهیم. این جدول در شکل زیر نشان داده شده است. در این جدول تنها جملات مینیمم پوشیده نشده و دسته‌بندی‌های اول نامزد برای پوشاندن آنها منظور شده‌اند.

نسخه پیش‌نویس

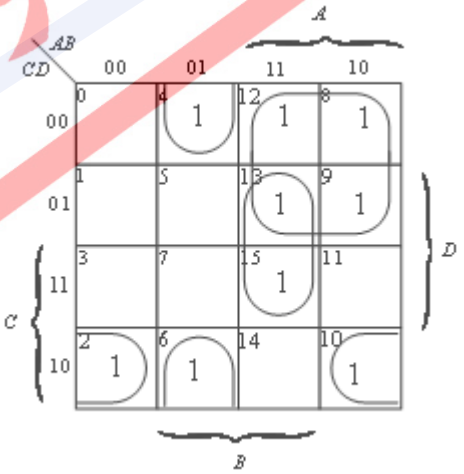
جدول ۱-۹ - برگزیدن حداقل دسته‌بندی‌های اول لازم برای پوشاندن سایر جملات مینیمم

	√ 2	√ 4	√ 6	√ 10
PI_2	×		×	
* PI_3	×			×
* PI_4		×	×	
PI_5		×		
PI_6				×

کدام دسته‌بندی‌های اول را باید برگزید؟ PI_5 و PI_6 مسلماً انتخاب خوبی نیستند. زیرا هر کدام تنها یک جملهٔ مینیمم را می‌پوشانند، و آن جملهٔ مینیمم توسط دسته‌بندی اول دیگری نیز پوشانده می‌شود. توجه کنید که با انتخاب PI_3 و PI_4 می‌توان هر چهار جملهٔ مینیمم ۲، ۴، ۶، و ۱۰ را پوشاند. ستاره‌ها، دسته‌بندی‌های برگزیده شده و علامت √ بالای ستون‌ها، جملات مینیمم پوشانده شده را نشان می‌دهند. پس حداقل دسته‌بندی‌های اول برای ساختن تابع عبارت است از:

$$f(A,B,C,D) = PI_1 + PI_3 + PI_4 + PI_7$$
$$= 1-0-+-010+01-0+11-1$$
$$= A\overline{C} + \overline{B}C\overline{D} + \overline{A}B\overline{D} + ABD$$

دسته بندی متناظر توسط جدول کارنو در شکل ۱-۱۰ نشان داده شده است.


A
AB
CD
00
01
11
10
0
4
12
8
1
1
1

۱-۴ تمرین

۱- با فرض اینکه در مثال۷، کتابخانه فقط دارای گیت NAND با دو ورودی باشد، هزینهٔ پیاده‌سازی مدار را در هر دو روش با یکدیگر مقایسه کنید.

۲- توابع زیر را که همراه با شرایط قابل عدم توجه هستند را با استفاده از نقشه کارنو کمینه کنید:

الف) $f(A,B,C,D)=\sum m(2,9,10,12,13)+d(1,5,14)$

ب) $f(A,B,C,D)=\sum m(1,3,6,7)+d(4,9,11)$

ج) $f(A,B,C,D)=\sum m(3,11,12,19,23,29)+d(5,7,13,27)$

۳- یک شبکه ترکیبی با خروجی چند گانه طراحی کنید ، در صورتی که x_0 و x_1 سیگنالهای ورودی و c_0، c_1 سیگنالهای کنترلی و توابع خروجی f_0 و f_1 باشند.

c_0	c_1	f_0	f_1
0	0	0	0
0	1	x_0	0
1	0	0	x_1
1	1	x_0	x_1

۴- نقشه کارنو توابع زیر را رسم کرده و در هر مورد مینترم ها را مشخص سازید:

الف) $f(A,B,C,D)=\bar{B}CD+\bar{A}B\bar{D}+B\bar{C}D+A\bar{B}D$

ب) $f(A,B,C,D)=\bar{B}\bar{C}\bar{D}+\bar{A}\bar{B}\bar{C}+\bar{A}\bar{C}D+BCD+ABC$

۵- برای توابع زیر POS را تعیین کنید:

الف) $f(A,B,C,D,E)=\prod m(0,1,2,5,7,8,10,15,17,21,22,24,26,29)$

ب) $f(A,B,C,D,E)=\prod m(0,2,4,6,9,11,13,15,16,19,20,25,27,29,31)$

نسخه پیش‌نویس Draft

۶-با استفاده از روش مک کلاسکی ، توابع زیر را کمینه کنید:

الف) $f(A,B,C,D)=\sum m(1,3,6,7,8,9,12,14)$

ب) $f(A,B,C,D)=\sum m(0,2,4,5,10,11,13,15)$

۷-توابع با خروجی چندگانه زیر را با استفاده از روش مک کلاسکی ، کمینه کنید:

الف)
$$f_\alpha(A,B,C,D)=\sum m(4,5,6,15)+d(8,11)$$
$$f_\beta(A,B,C,D)=\sum m(0,2,3,4,5)+d(8,11)$$

ب)
$$f_\alpha(A,B,C,D)=\sum m(3,4,6,11,12)+d(14,15)$$
$$f_\beta(A,B,C,D)=\sum m(4,5,6,11,14)+d(8,12)$$

نسخه پیش‌نویس Draft

فصل ۲

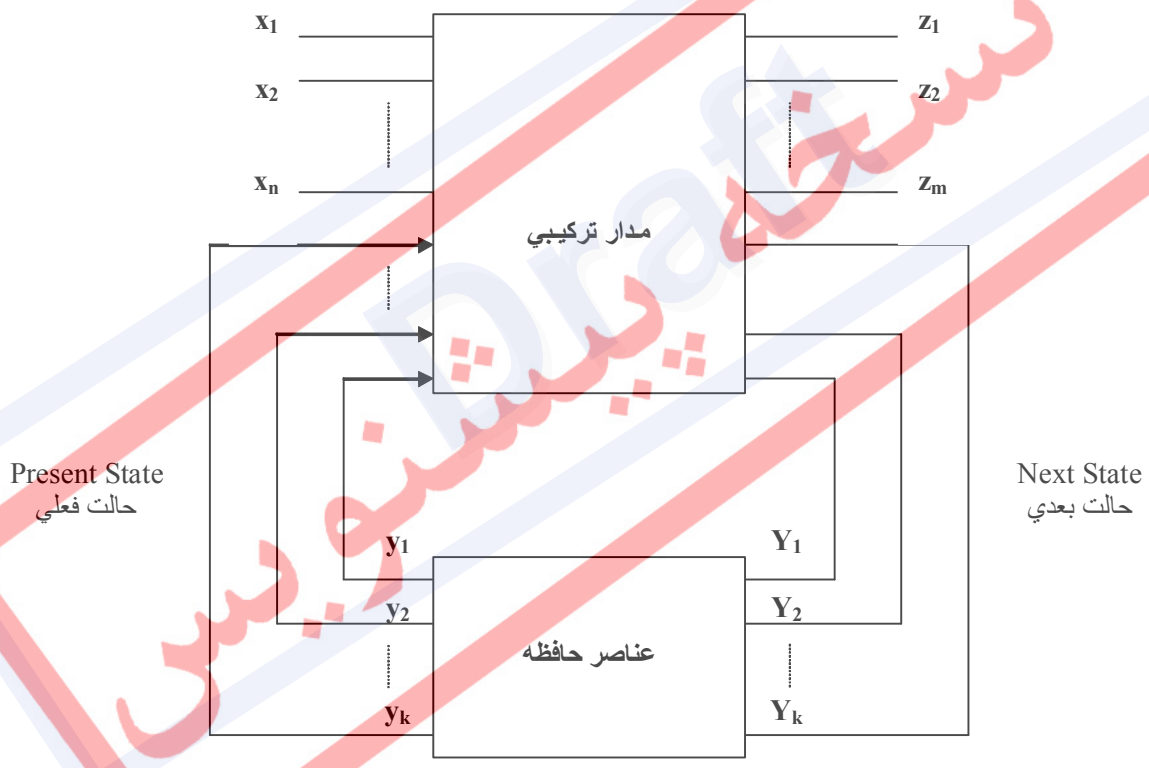
معرفی عناصر ترتیبی

نسخه پیش‌نویس
Draft

۲-۱ مقدمه

مفهوم اساسی مدارات ترتیبی، مفهومی بسیار مهم و پایه‌ای در طراحی سیستم‌های دیجیتال است. در مدارهای ترکیبی، خروجی مدار، تابعی از ورودی‌های زمان حال است. ولی خروجی مدارهای ترتیبی نه تنها تابعی از ورودیهای زمان حال، که تابعی از ورودیهای قبلی مدار نیز هست. تاریخچهٔ ورودیهای مدارات ترتیبی، در عنصر ذخیره کننده‌ای موسوم به حافظه نگهداشته می‌شود. با استفاده از حافظه، امکان حل مسائل متعددی فراهم می‌شود، که با استفاده از مدارهای ترکیبی قابل حل نیست.

شکل ۲-۱، یک مدل از مدارات ترتیبی را نشان می‌دهد که در آن ورودی n تایی $(x_1,\ldots,x_n)$، خروجی m تایی $(z_1,\ldots,z_m)$ و r تایی‌های $(y_1,\ldots,y_r)$ و $(Y_1,\ldots,Y_r)$ به ترتیب حالت فعلی و حالت بعدی نامیده می‌شوند.



شکل۲-۱- مدل نمونهٔ مدارهای ترتیبی

روابط بین این متغیرها را می‌توان به صورت زیر بیان کرد:

$$z_i = g_i(x_1,\ldots,x_n,y_1,\ldots,y_r) \qquad where \qquad i=1,\ldots,m$$
$$Y_j = h_j(x_1,\ldots,x_n,y_1,\ldots,y_r) \qquad where \qquad j=1,\ldots,r$$

g_i و h_j توابع بولی هستند و این معادلات را می‌توان به صورت برداری زیر نوشت:

نسخه پیش‌نویس

$$z=g(x,y)$$

$$Y=h(x,y)$$

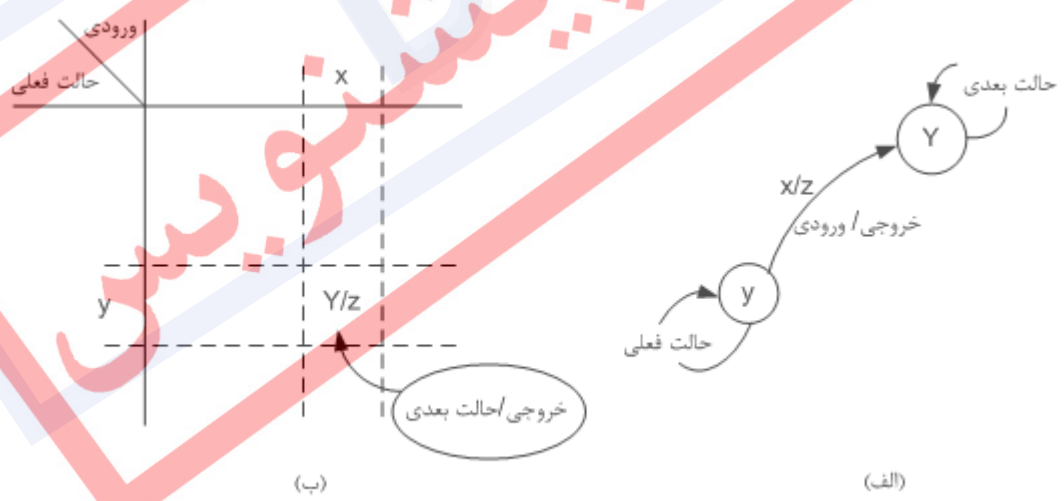
که در آن

$$Y=\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Yr \end{bmatrix} \qquad y=\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_r \end{bmatrix} \qquad x=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad z=\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix}$$

توجه کنید که Y_i, y_i , x_i ,z_i همگی متغیرهای دودویی‌اند.

۲-۲ جدول حالت و نمودار حالت

به کمک معادلات منطقی و معادلات برداری بیان شده در قبل، می‌توان رفتار مدار ترتیبی شکل۲-۱ را به طور کامل مشخص نمود. این توصیف اگر چه کامل است، اما تصویر شفافی از روابط بین متغیرها به دست نمی‌دهد. برای بیان روابط بین ورودی، خروجی، حالت فعلی، و حالت بعدی، از جدول حالت یا نمودار حالت استفاده می‌شود. نمودار حالت نمایش ترسیمی یک مدار ترتیبی است که در آن حالتهای مدار با دایره‌ها و گذرهای حالت با پیکان نشان داده می‌شوند. ورودی x و خروجی متناظر z به صورت شکل ۲-۲- الف، بر روی پیکان نشان داده می‌شود.



شکل۲-۲ - جدول ونمودار حالت .(الف) نمودارحالت. (ب) جدول حالت

شکل۲-۲- ب نمایش مدار ترتیبی را توسط جدول حالت نشان می‌دهد. تمام بردارهای ورودی x در بالا و تمام بردارهای حالت y در سمت چپ قید شده‌اند. در خانه‌های جدول، حالت بعدی Y و خروجی z قید می‌شوند. این جدول مشخص می‌کند که به ازای ورودی x و قرار داشتن در حالت y ، مدار به حالت

نسخه پیش‌نویس Draft

Y می‌رود و خروجی z را تولید می‌کند. در عمل، در نمودارها و جدولهای حالت معمولاً به جای بردار از نماد استفاده می‌شود. مثلاً مداری ترتیبی با متغیرهای حالت y_1 و y_2 را در نظر بگیرید، $y=\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ ،

پس بردار y می‌تواند یکی از چهار مقدار زیر را داشته باشد:

$$y=\begin{bmatrix} 0 \\ 0 \end{bmatrix}=A \quad ، \quad y=\begin{bmatrix} 0 \\ 1 \end{bmatrix}=B \quad ، \quad y=\begin{bmatrix} 1 \\ 0 \end{bmatrix}=C \quad ، \quad y=\begin{bmatrix} 1 \\ 1 \end{bmatrix}=D$$

بنابراین این مدار تنها چهار حالت ممکن دارد که می‌توان آنها را با A، B، C و D نشان داد. به طور کلی اگر تعداد عناصر حافظه مدار r و تعداد حالتهای آن N_s باشد، بین r و N_s رابطهٔ زیر برقرار است:

$$2^{r-1} < N_s \le 2^r$$

مثال ۱) یک مدار ترتیبی را با یک متغیر ورودی x، دو متغیر حالت y_1 و y_2 ، و یک متغیر خروجی z در نظر بگیرید:

<u>ورودی</u>

x=0 x=1

<u>حالتها</u>

$[y_1, y_2] = [0\ 0] = A$

$[y_1, y_2] = [0\ 1] = B$

$[y_1, y_2] = [1\ 0] = C$

$[y_1, y_2] = [1\ 1] = D$

<u>خروجی</u>

Z=1 Z=0

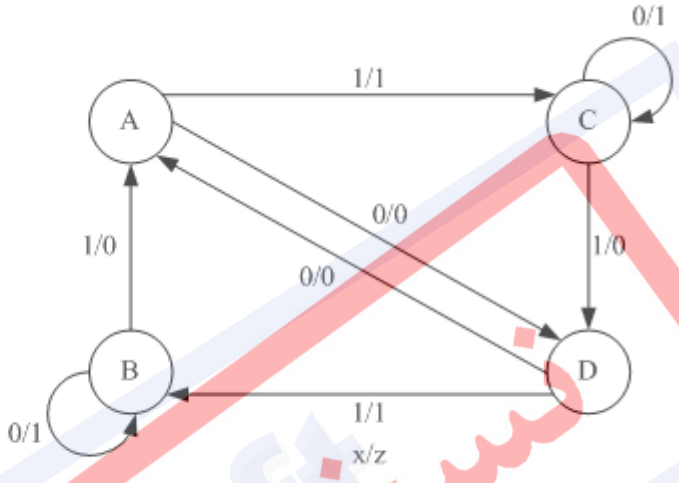
نسخه پیش‌نویس Draft

نمودار حالت این مدار ترتیبی در شکل ۳ نشان داده شده است. فرض کنید مدار ابتدا در حالت A قرار دارد. اگر ورودی x=0 باشد حالت بعدی D و خروجی z=0 است. این اطلاعات را می‌توان علاوه بر نمودار حالت، از جدول حالت نیز کسب کرد. حال رشتهٔ ورودی زیر را به مدار اعمال می‌کنیم:

X=0110101100

حالت اولیه	ورودی x: 0	ورودی x: 1
A	D/0	C/1
B	B/1	A/0
C	C/1	D/0
D	A/0	B/1

(الف)



(ب)

شکل ۲-۳ - (الف) جدول حالت. (ب) نمودارحالت

اگر حالت اولیهٔ A باشد، به ازای رشتهٔ ورودی داده شده، مدار به صورت زیر رفتار می‌کند:

جدول ۲-۱ – رفتار مدار به ازاء حالت اولیه A و ورودی داده‌شده

زمان	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
حالت فعلی	A	D	B	A	D	B	B	A	C	C	C
ورودی	۰	۱	۱	۰	۱	۰	۱	۱	۰	۰	
حالت بعدی	D	B	A	D	B	B	A	C	C	C	
خروجی	۰	۱	۰	۰	۱	۱	۰	۱	۱	۱	

پس با دادن رشتهٔ ورودی I=0110101100 به مدار، رشتهٔ خروجی z=0100110111 ایجاد شده و حالت نهایی مدار C خواهد شد.

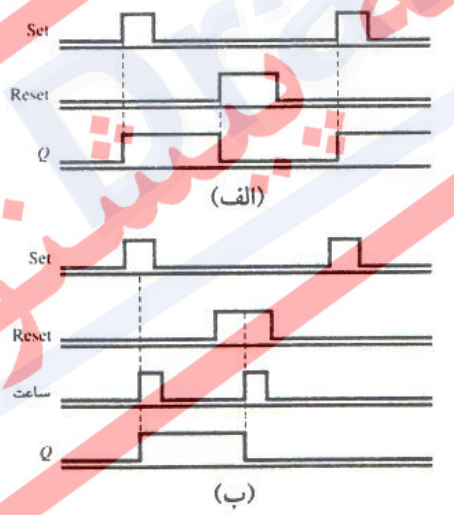
نسخه پیش‌نویس Draft

۲-۳ عناصر حافظه

همانطور که پیش از این اشاره شد، واحد حافظه، یک بخش اصلی از مدار ترتیبی است. در کاربردهای سوئیچینگ، اغلب عناصر حافظه، مدارهای الکترونیکی دو پایا هستند. یعنی مدارهایی که می‌توانند در هر یک از دو حالت ممکن ۰ و ۱ به مدت نامحدودی باقی بمانند. برای ذخیرهٔ مقدار ۱، آن را در حالت ۱ و برای ذخیرهٔ مقدار ۰، آن را در حالت ۰ قرار می‌دهند. خروجی Q مدار، وضعیت فعلی حافظه را نشان می‌دهد. هر مدار حافظه یک یا چند ورودی تحریک دارد، که برای تحریک مدار و بردن آن به حالت دلخواه به کار می‌روند. عناصر حافظه معمولاً بر حسب ورودیهای تحریکشان نامگذاری می‌شوند.

دو نوع عنصر حافظهٔ اصلی مدارهای سوئیچینگ، به ترتیب نگهدار[1] و فیلیپ-فلاپ نام دارند.

- **نگهدار** – مداری است که سیگنال ورودی تحریک آن حالت خروجی آن را تعیین می‌کند. ورودی تحریکی که نگهدار را به حالت ۱ می‌برد ورودی set ، و ورودی تحریکی که نگهدار را به حالت ۰ می‌برد ورودی reset نامیده می‌شود. عملکرد نگهدار در شکل ۲-۴- الف نشان داده شده است.



شکل ۲-۴ - (الف) زمانبندی نگهدار. (ب) زمان بندی فیلیپ-فلاپ

- **فیلیپ-فلاپ** – تفاوت فیلیپ-فلاپ با نگهدار در وجود سیگنال کنترلی موسوم به ساعت است. سیگنال ساعت به فلیپ-فلاپ فرمان می‌دهد که برحسب ورودیهایش تغییر کند. هم در نگهدار و هم در فیلیپ-فلاپ، حالت بعدی توسط ورودیهای تحریک تعیین می‌شود. ولی همانطور که شکل ۲-۴

[1] Latch

نسخه پیش‌نویس Draft

نشان داده شده، نگهدار با تغییر ورودیهای تحریکش تغییر حالت می‌دهد، ولی فیلیپ-فلاپ قبل از تغییر حالت منتظر سیگنال ساعت می‌ماند. یعنی حالت نهایی فیلیپ-فلاپ توسط مقدار ورودیهای تحریک در زمان رخ دادن سیگنال ساعت تعیین می‌شود. به این ترتیب می‌توان تغییر فیلیپ-فلاپهای یک مدار ترتیبی را با یک سیگنال ساعت مشترک همگام کرد.

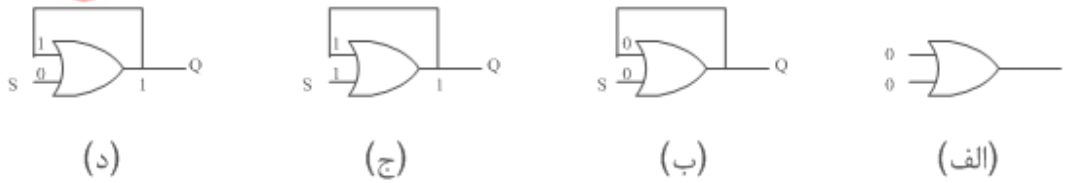
انواع عناصر حافظه در جدول ۲-۲ نمایش داده شده اند.

جدول ۲-۲- انواع عناصر حافظه

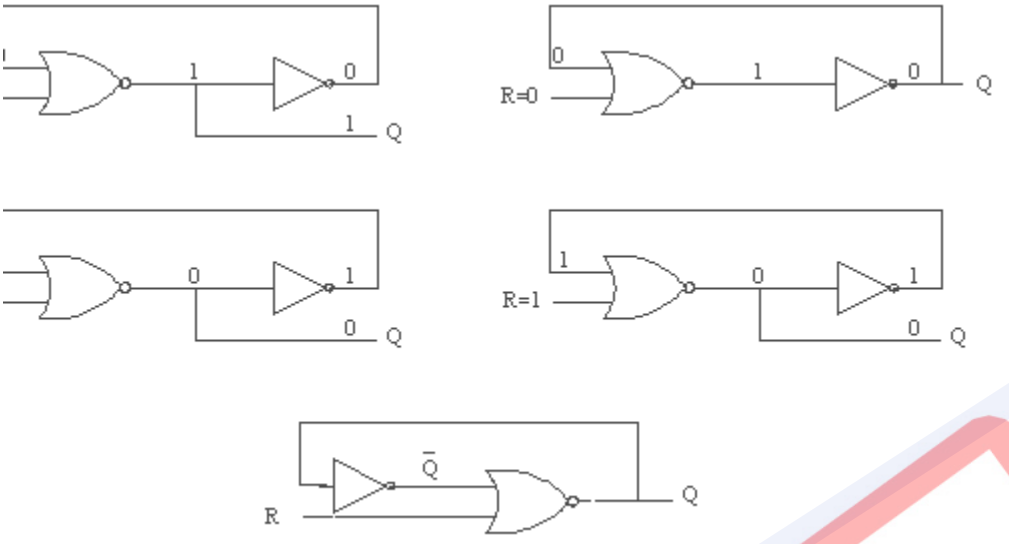
Device	تعداد اجزاء	توصیف اجزاء
74LS73A	۲	فیلیپ-فلاپ jk با تریگر لبه ای و clear
7474	۲	فیلیپ-فلاپ D با تریگر لبه مثبت و preset و clear
74LS75	۴	نگهدار با enable
7476	۲	فیلیپ-فلاپ jk با تریگر پالسی و preset و clear
74111	۲	فیلیپ-فلاپ jk پایه-پیرو با preset و clear و قفل داده
74116	۲	نگهدار D ۴ بیتی با clear و enable دوگانه
74175	۴	فیلیپ-فلاپ D با تریگر لبه مثبت و clear
74273	۸	فیلیپ-فلاپ D با تریگر لبه مثبت و clear
74276	۴	فیلیپ-فلاپ jk با تریگر لبه مثبت و preset و clear
74279	۴	نگهدار SR با ورودی های صفر فعال

۲-۳-۱ نگهدار SR

در شکل ۵- الف، یک گیت OR نشان داده شده است. دو ورودی ۰ به آن داده شده است. اگر خروجی به صورت شکل ۵- ب به ورودی برگردانده شود، گیت با خروجی ۰ در حالت پایدار قرار می‌گیرد. اگر مطابق شکل ۵- ج به ورودی مستقل s مقدار ۱ اعمال شود، خروجی ۱ می‌شود. بنابراین خروجی Q به مقدار ۱ set می‌شود. اگر ورودی مجدداً ۰ شود، خروجی ۱ باقی می‌ماند. این عنصر به طور دائم در حالت ۱ می‌ماند و نگهدار set نامیده می‌شود.



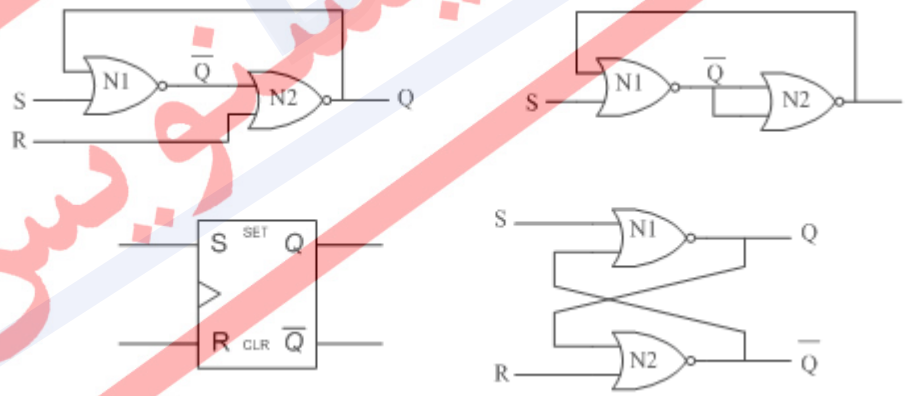
شکل ۲-۵- نگهدار set



شکل ۲-۶ - نگهدار reset

شکل ۲-۶ نمایانگر یک مدار reset است. این مدار نیز کارکردی مشابه مدار set دارد با این تفاوت که در حالت صفر به صورت پایدار باقی می‌ماند.

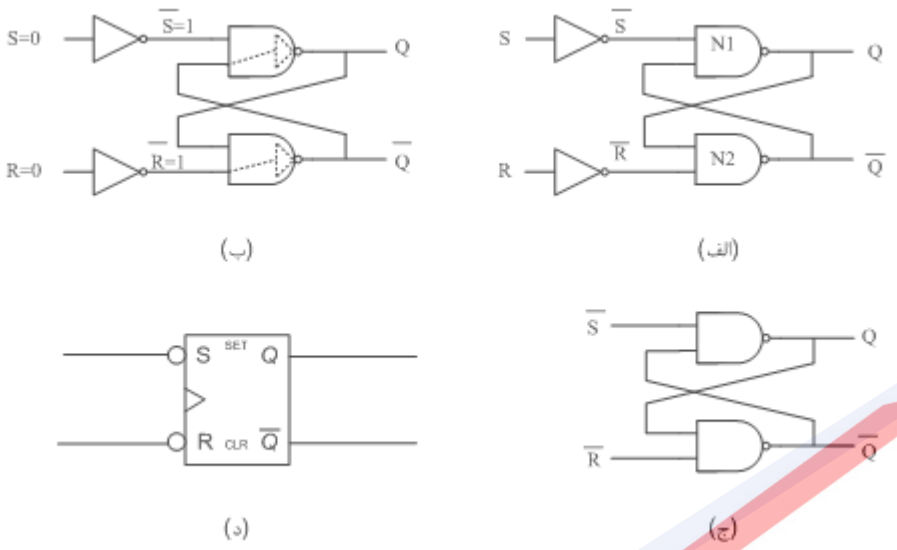
عناصری که دائماً در یک حالت منطقی باقی می‌مانند، بجز در مواردی بسیار نادر، عناصر کارآمدی نیستند. اگر دو ویژگی نگهدارها به صورت همزمان به کار برده شود، می‌توان در مواقع لزوم نگهدار را set یا reset کرد. شکل ۲-۷، نگهدار SR را نشان می‌دهد.



شکل ۲-۷ - نگهدار SR

این مدار نیز مانند نگهدار set عمل می‌کند. اگر ورودی پایین گیت N2 قطع شود، این پایه را می‌توان به صورت ورودی تحریک reset به کار برد. اگر به جای گیت‌های NOR از گیت‌های NAND استفاده شود، نگهدار SR با ساختار NAND ساخته می‌شود. (شکل ۲-۸)

نسخه پیش‌نویس Draft



شکل ۲-۸ - نگهدار SR با ساختار NAND

اگر هر دو ورودی NAND برابر با ۰ باشد، مدار به صورت زیر عمل می‌کند:

گیت N1 (S=0): $\overline{\overline{S}.\overline{Q}} = \overline{1.\overline{Q}} = \overline{\overline{Q}} = Q$

گیت N2 (R=0): $\overline{\overline{R}.Q} = \overline{1.Q} = \overline{Q}$

اگر هر دو ورودی ۱ باشند داریم:

گیت N1 (S=1): $Q = \overline{\overline{S}.\overline{Q}} = \overline{0.\overline{Q}} = \overline{0} = 1$

گیت N2 (R=1): $\overline{Q} = \overline{\overline{R}.Q} = \overline{0.Q} = \overline{0} = 1$

و یا اگر یکی ۰ و دیگری ۱ باشد خواهیم داشت:

گیت N1 (S=1,R=0): $Q = \overline{\overline{S}.\overline{Q}} = \overline{0.\overline{Q}} = \overline{0} = 1$

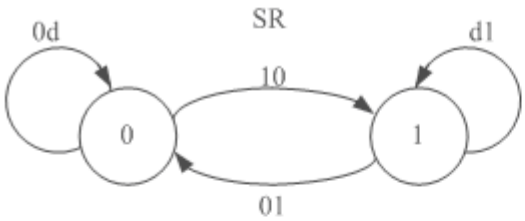
گیت N2 (S=0,R=1): $\overline{Q} = \overline{\overline{R}.Q} = \overline{\overline{1}.Q} = \overline{0.Q} = \overline{0} = 1$

نسخه پیش‌نویس Draft

۲-۳-۱-۱ جدول تحریک نگهدار SR

عملکرد منطقی نگهدار SR در جدول تحریک شکل ۲-۹- الف خلاصه شده است. از جدول کارنو در شکل ۲-۹- ج می‌توان عبارت Q^* را که به معادلهٔ مشخصهٔ نگهدار SR موسوم است، به دست آورد:

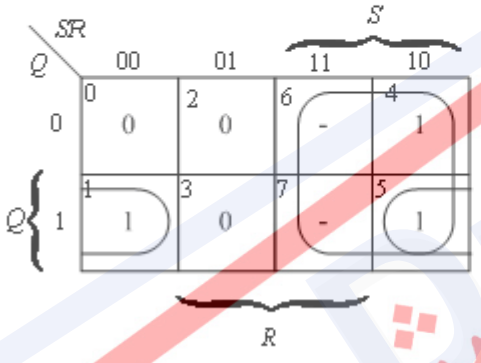
$$Q^* = S + \overline{R}Q$$



(ب)

ورودیهای تحریک S	R	حالت فعلی Q	حالت بعدی Q^*	
0	0	0	0	بدون تغییر
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	×	غیرمجاز
1	1	1	×	

(الف)



(ج)

شکل ۲-۹ - مشخصات نگهدار SR

این عملکرد را می‌توان در سه مورد دسته بندی کرد.

مورد۱: S=R=0 که معادله به $Q^*=Q$ تبدیل می‌شود، یعنی تغییر حالتی صورت نمی‌گیرد.

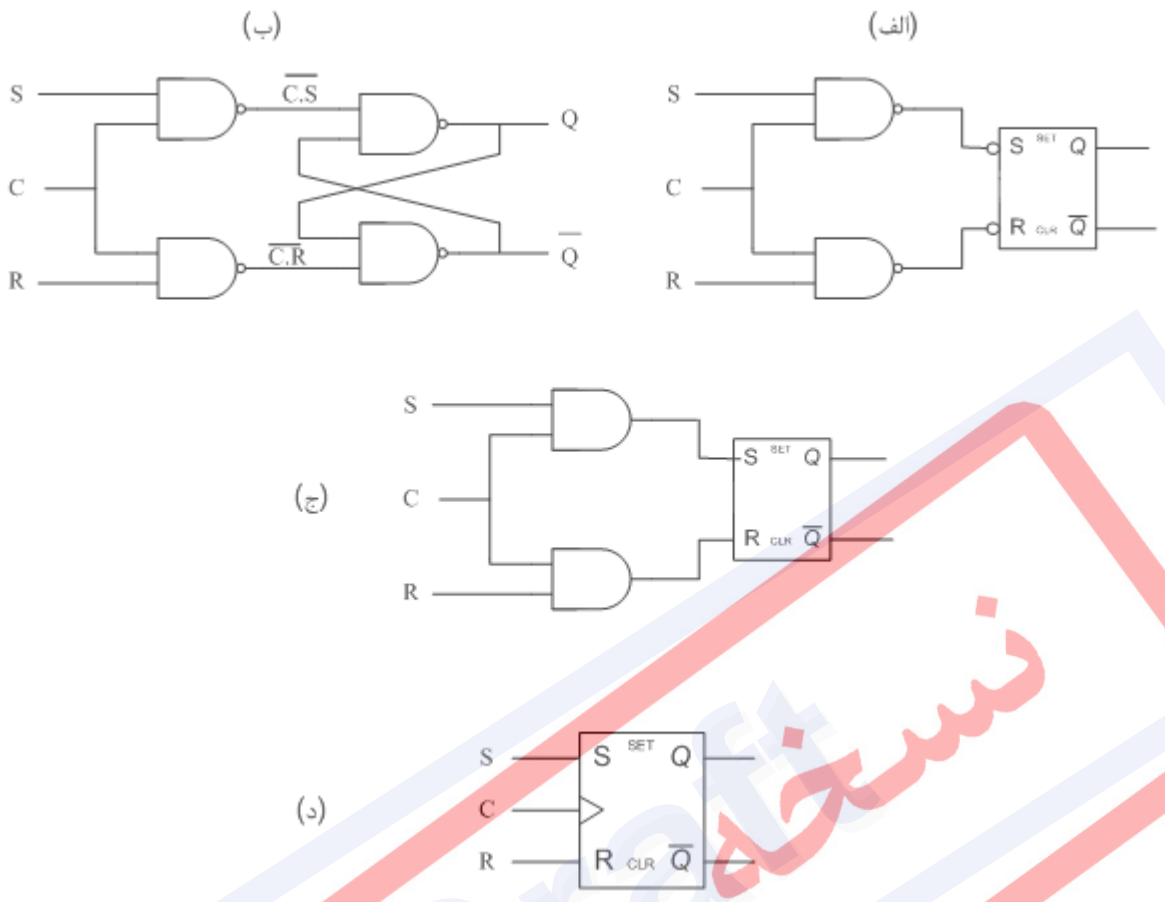
مورد۲: S=1,R=0 که معادله به $Q^*=1$ تبدیل می‌شود، و عمل set را نشان می‌دهد.

مورد۳: S=0,R=1 معادله به $Q^*=0$ تبدیل می‌شود، و عمل reset را نشان می‌دهد.

۲-۳-۱-۲ نگهدار SR گیت‌دار

در شکل۲-۱۰- الف، سیگنال کنترلی c به نگهدار SR اضافه شده است تا امکان اعمال کنترل بر روی ورودیهای S و R نگهدار، فراهم شود.

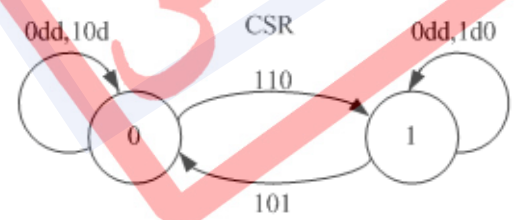
نسخه پیش‌نویس



شکل۲-۱۰- نگهدار SR گیت‌دار

اگر c=0 باشد، و ورودیهای نگهدار در S=R=0 قرارداشته باشند، نگهدار پایدار باقی می‌ماند.

به ازای c=1 ، نگهدارSR به صورت جدول تحریک شکل ۲-۱۱- الف خواهد بود.



ورودی توانا‌کننده C	ورودیهای تحریک S	R	حالت فعلی Q	حالت بعدی Q^*	
0	×	×	0	0	توقف
0	×	×	1	1	
1	0	0	0	0	بدون تغییر
1	0	0	1	1	
1	0	1	0	0	Reset
1	0	1	1	0	
1	1	0	0	1	Set
1	1	0	1	1	
1	1	1	0	×	غیرمجاز
1	1	1	1	×	

(ب) (الف)

شکل ۲-۱۱ - نگهدار SR (الف) جدول تحریک. (ب) نمودار حالت.

نسخه پیش‌نویس Draft

و معادلهٔ مشخصهٔ آن به صورت زیر است:

$$Q^* = SC + \overline{R}Q + \overline{C}Q$$

۲ – ۴ فیلیپ–فلاپ

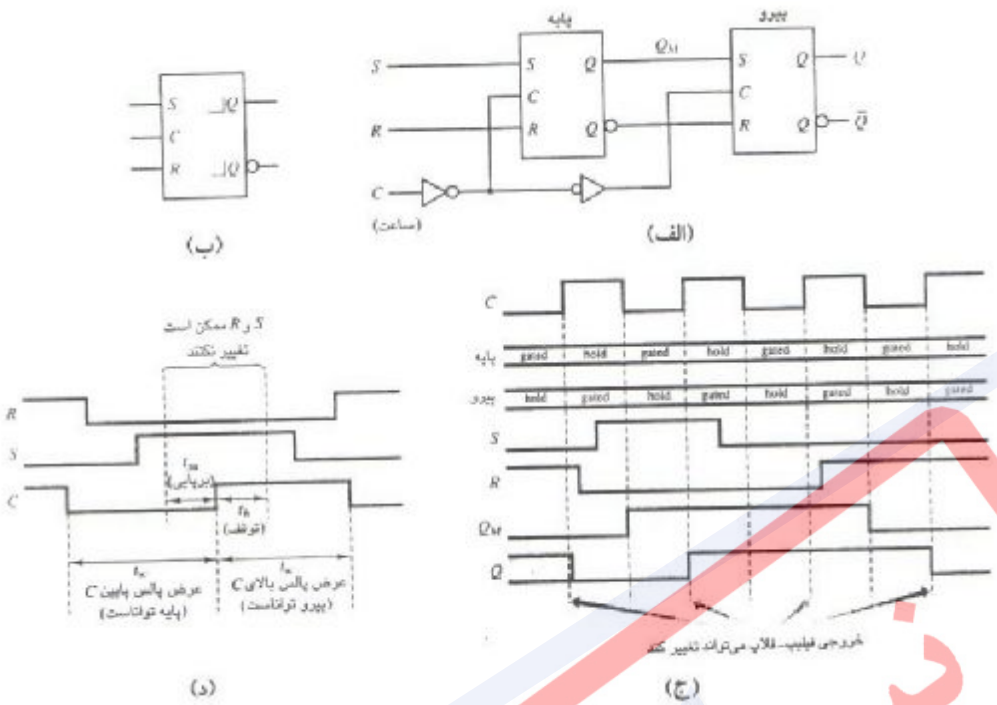
نگهدارها برای استفاده در مدارهای منطقی ترتیبی همگام مناسب نیستند. به عنوان مثال در شکل ۱، سیگنال‌های خروجی عناصر حافظه، سیگنال‌های ورودی مدار ترکیبی نیز هستند و برعکس. اگر ورودی فعال کنندهٔ نگهدار (c) برابر با ۱ باشد، نگهدار نیز به صورت یک مدار ترکیبی عمل می‌کند! پس ممکن است دو مدار ترکیبی با اتصال فیدبکی ایجاد شود، که امکان نوسان و ایجاد رفتار ناپایدار را به وجود می‌آورد. برای حل این مشکل، یک سیگنال کنترل زمان بندی خاص موسوم به ساعت به کار برده می‌شود که زمانهای تغییر حالت عناصر حافظه را معین می‌کند.

۲-۴-۱ فیلیپ–فلاپ SR پایه و پیرو

یک روش برای پیشگیری از رفتار ناپایدار توصیف شده، به کارگیری دو آرایش پایه و پیرو[1] در شکل ۲-۱۲- الف است. فعال کننده‌های فیلیپ-فلاپ‌های پایه و پیرو، به صورت سیگنال‌های وارون نسبت به یکدیگر، تحریک می‌شوند. وقتی سیگنال C صفر است، نگهدار پایه به عنوان گیت و نگهدار پیرو به عنوان ذخیره‌ساز عمل می‌کنند. در این حالت، تغییرات سیگنال‌های تحریک S و R فقط بر روی نگهدار پایه تأثیر می‌گذارند. هنگامی که سیگنال C برابر با ۱ شود، نگهدار پایه به عنوان ذخیره‌ساز و نگهدار پیرو به عنوان گیت عمل خواهند کرد. در نتیجه، خروجی نگهدار پایه به خروجی نگهدار پیرو منتقل می‌شود. در این هنگام، تغییرات ورودی تأثیری بر نگهدار پایه نخواهند داشت.

[1] Master-Slave

نسخه پیش‌نویس



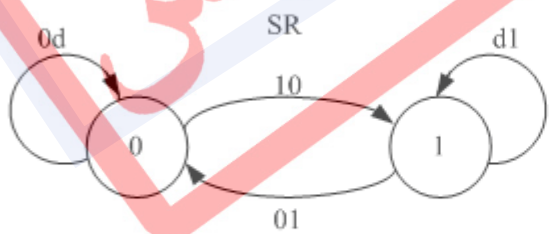
شکل۲-۱۲ - فیلیپ-فلاپ SR پایه و پیرو

۲-۴-۱-۱ جدول تحریک و معادلهٔ مشخصه

شکل‌های **2-۱۳**- الف و ب به ترتیب جدول تحریک و نمودار حالت فیلیپ-فلاپ SR پایه و پیرو را نشان می‌دهند.

ورودیهای تحریک S	R	حالت فعلی Q	حالت بعدی Q^*	
0	0	0	0	بدون تغییر
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	×	غیرمجاز
1	1	1	×	

(الف)



(ب)

شکل ۲-۱۳ - مشخصات فیلیپ-فلاپ SR پایه و پیرو

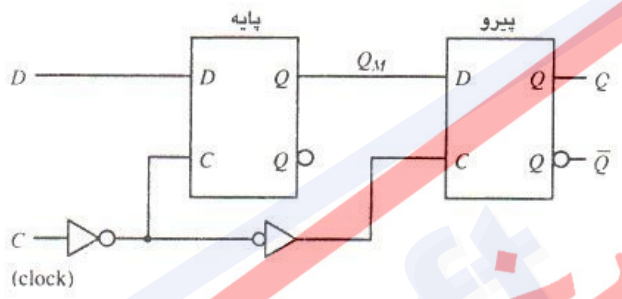
نسخه پیش‌نویس Draft

معادلهٔ مشخصهٔ آن نیز به صورت زیر است:

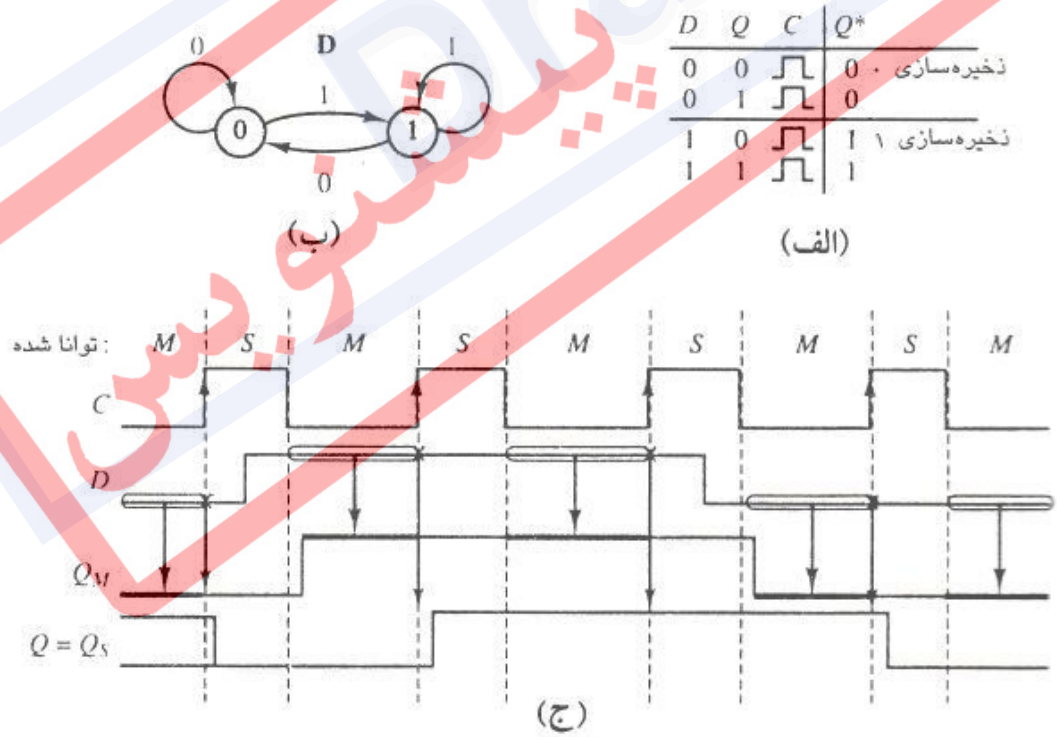
$$Q^{+} = S + \overline{R}Q$$

۲-۴-۲ فیلیپ-فلاپ D پایه و پیرو

با اتصال دو نگهدار D به صورت شکل ۲-۱۴ می‌توان یک فیلیپ-فلاپ D پایه و پیرو ساخت. نگهدار پایه در زمان ۰ بودن سیگنال ساعت و نگهدار پیرو در زمان ۱ بودن سیگنال ساعت، به صورت گیت عمل می‌کنند. مشخصات این فیلیپ-فلاپ در شکل ۲-۱۵ نشان داده شده است.



شکل ۲-۱۴ - فیلیپ-فلاپ D پایه و پیرو

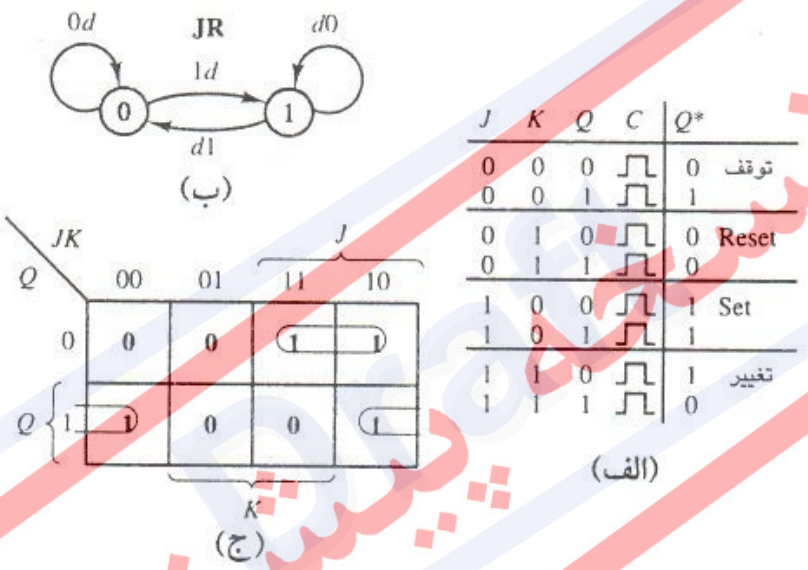


شکل ۲-۱۵ - مشخصات فیلیپ-فلاپ D پایه و پیرو

نسخه پیش‌نویس Draft

۲-۴-۳ فیلیپ-فلاپ JK پایه و پیرو

فیلیپ-فلاپ JK را می‌توان تعمیمی از فیلیپ-فلاپ SR به حساب آورد. فیلیپ-فلاپ JK به صورت یک فیلیپ-فلاپ SR با ورودیهای J=S و K=R عمل می‌کند. در فیلیپ-فلاپ SR ترکیب ورودی S=R=1 مجاز نیست، ولی در فیلیپ-فلاپ JK از این ترکیب به عنوان یک وجه کاری مفید استفاده می‌شود؛ که به آن وجه تغییر می‌گویند: فیلیپ-فلاپ JK به ازای ورودی J=K=1 تغییر حالت می‌دهد، یعنی $1 \rightarrow 0$ یا $0 \rightarrow 1$. این چهار وجه کاری در جدول تحریک شکل ۲-۱۶- الف و نمودار حالت شکل ۲-۱۶- ب نشان داده شده است.



شکل ۲-۱۶ - مشخصات فیلیپ-فلاپ JK

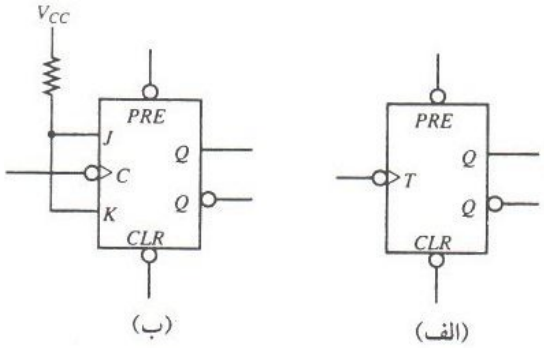
معادلهٔ مشخصهٔ آن به صورت زیر است:

$$Q^{+} = \overline{K}Q + J\overline{Q}$$

۲-۴-۴ فیلیپ-فلاپ T

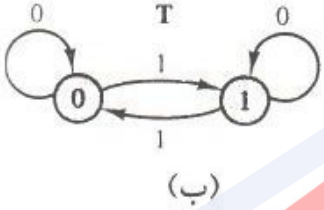
یکی از واحدهای سازندهٔ مدارهای منطقی ترتیبی، که پالس‌های روی یک خط ورودی را می‌شمارد، فیلیپ-فلاپ T است. فیلیپ-فلاپ T در آی‌سی‌های شمارنده به کار می‌رود. این فیلیپ-فلاپ تنها یک ورودی تحریک دارد و شکل ۲-۱۷- الف نماد منطقی فیلیپ-فلاپ را نشان می‌دهد. عملکرد این عنصر بدین صورت است که به ازای هر گذر سیگنال تحریک T، مقدار خروجی آن تغییر می‌کند.

نسخه پیش‌نویس



شکل ۲-۱۷ - فیلیپ-فلاپ T

در شکل ۱۸ جدول تحریک، نمودار حالت و معادلهٔ مشخصهٔ آن، نشان داده شده است.



T	Q	Q*	
↓	0	1	تغییر
↓	1	0	تغییر

(الف)

$$Q^* = \overline{Q}$$

معادله مشخصه

شکل ۲-۱۸ - مشخصات فیلیپ-فلاپ T

به طور خلاصه می‌توان تمام معادلات مشخصهٔ فیلیپ-فلاپ‌ها را به صورت جدول زیر نمایش داد.

جدول ۲-۳ - خلاصهٔ مشخصات نگهدارنده‌ها و فیلیپ-فلاپ‌ها

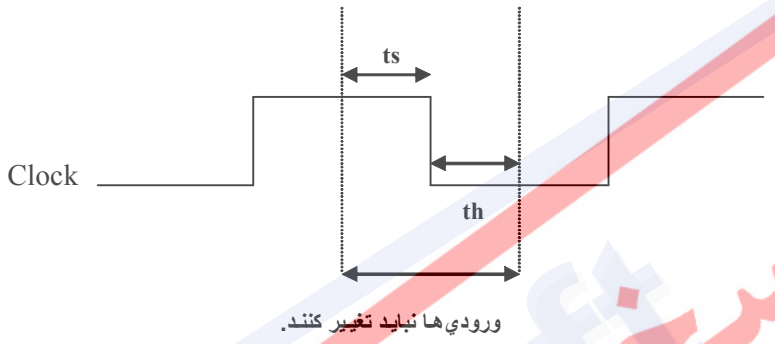
وسیله	معادلهٔ مشخصه
نگهدار SR	$Q^* = S + \overline{R}Q$
نگهدار SR دروازه‌دار	$Q^* = SC + \overline{C}R + \overline{C}Q$
نگهدار D	$Q^* = DC + \overline{C}Q$
فیلیپ - فلاپ SR	$Q^* = S + \overline{R}Q$
فیلیپ - فلاپ D	$Q^* = D$
فیلیپ - فلاپ JK	$Q^* = \overline{K}Q + J\overline{Q}$
فیلیپ - فلاپ T (بازیگر لبه‌ای)	$Q^* = \overline{Q}$
فیلیپ - فلاپ T (ساعتدار)	$Q^* = T\overline{Q} + \overline{T}Q$

نسخه پیش‌نویس Draft

۲-۴-۵ زمانبندی در فیلیپ-فلاپ‌ها

- زمان آماده‌سازی[1] (ts): زمان پیش از آمدن سیگنال ساعت. در این مدت، ورودی‌ها نباید تغییر کنند تا هنگامیکه سیگنال ساعت تولید می‌شود، ورودی‌های جدید در گیت مورد نظر اعمال شوند.
- زمان نگهداری[2] (th): زمان پس از آمدن سیگنال ساعت. در این مدت نیز، ورودی‌ها نباید تغییر کنند تا اثر گذاری لازم را بر روی گیت مورد نظر داشته باشند.

در شکل ۲-۱۹ این دو محدودیت زمانی نسبت به سیگنال ساعت، نشان داده شده است.



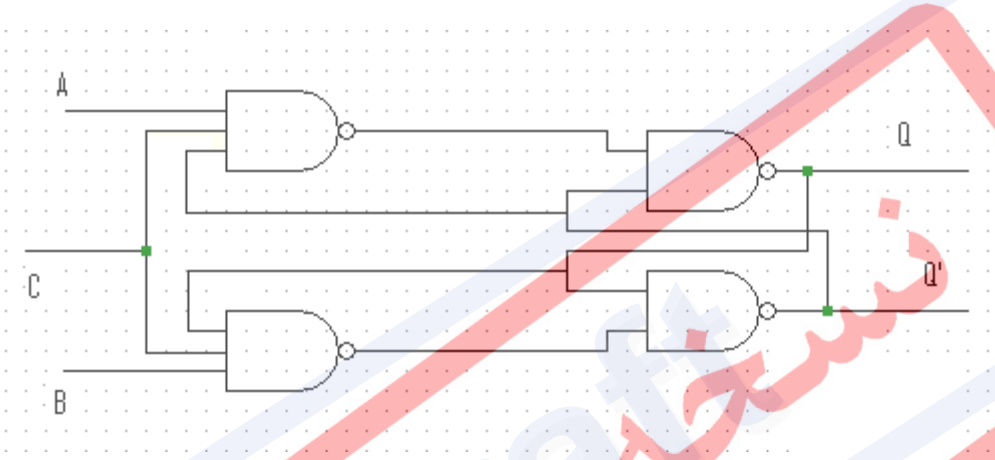
شکل ۲-۱۹ - زمانبندی فیلیپ-فلاپ

نسخه پیش‌نویس
Draft

[1] Setup Time
[2] Hold Time

۲-۵ تمرین

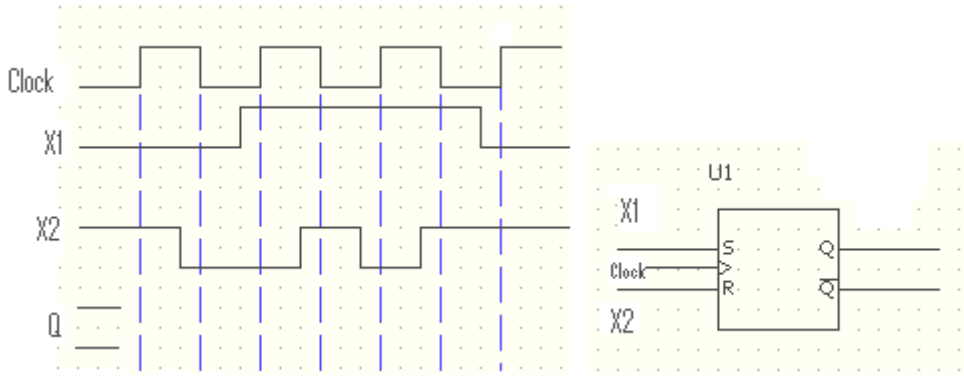
۱-آیا مدار شکل زیر یک طراحی معتبر برای یک لچ را نشان می دهد؟توضیح دهید.اگر پاسخ مثبت است ، جدول تحریک زیر را کامل کرده و توضیح دهید که آیا می توان از آن به عنوان یک لچ SR استفاده کرد و چگونه؟



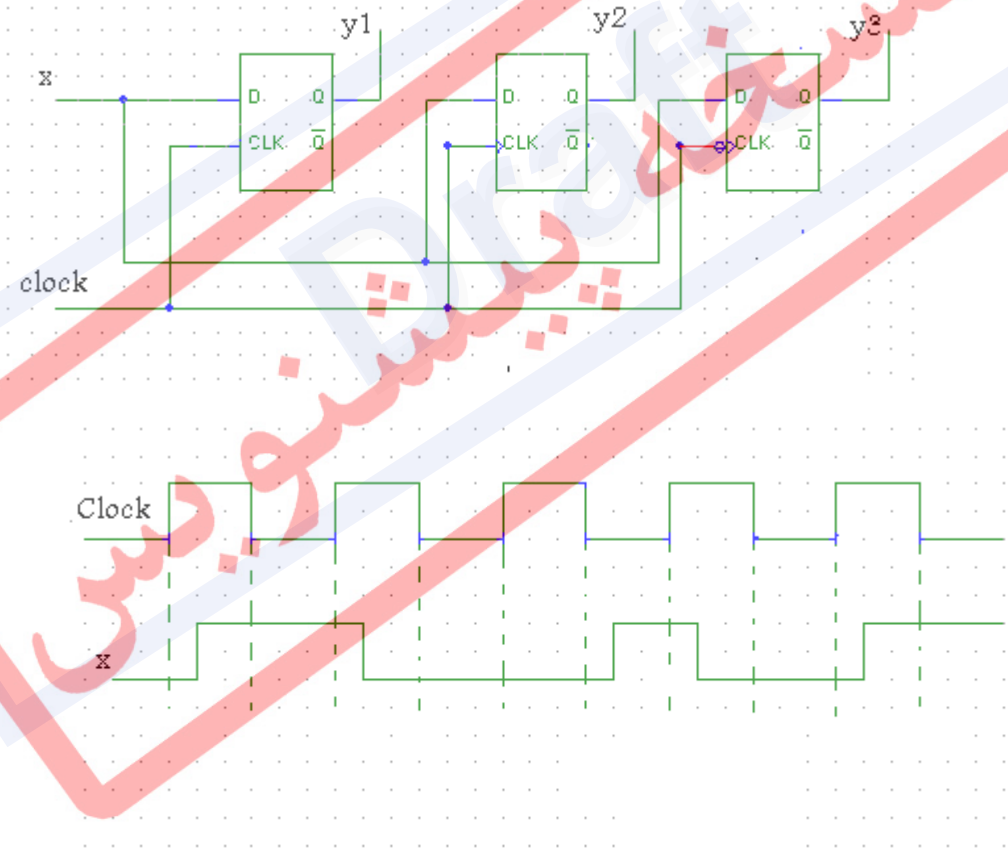
A	B	C	Q	Q^*	Mode
0	0	0	0	0	No change
0	0	0	1	1	No change

۲- فلیپ فلاپ داده شده در شکل زیر را در نظر بگیرید.ضمن کامل کردن دیاگرام زمانی و رسم خروجی Q آن توضیح دهید که آیا این سیستم به حالت ناپایدار سوق می یابد.توجه کنید که فلیپ فلاپ با لبه مثبت ساعت تریگر می شود و شرط S=R=1 دوبار با ورودی ها ایجاد می شود.

نسخه پیش‌نویس Draft

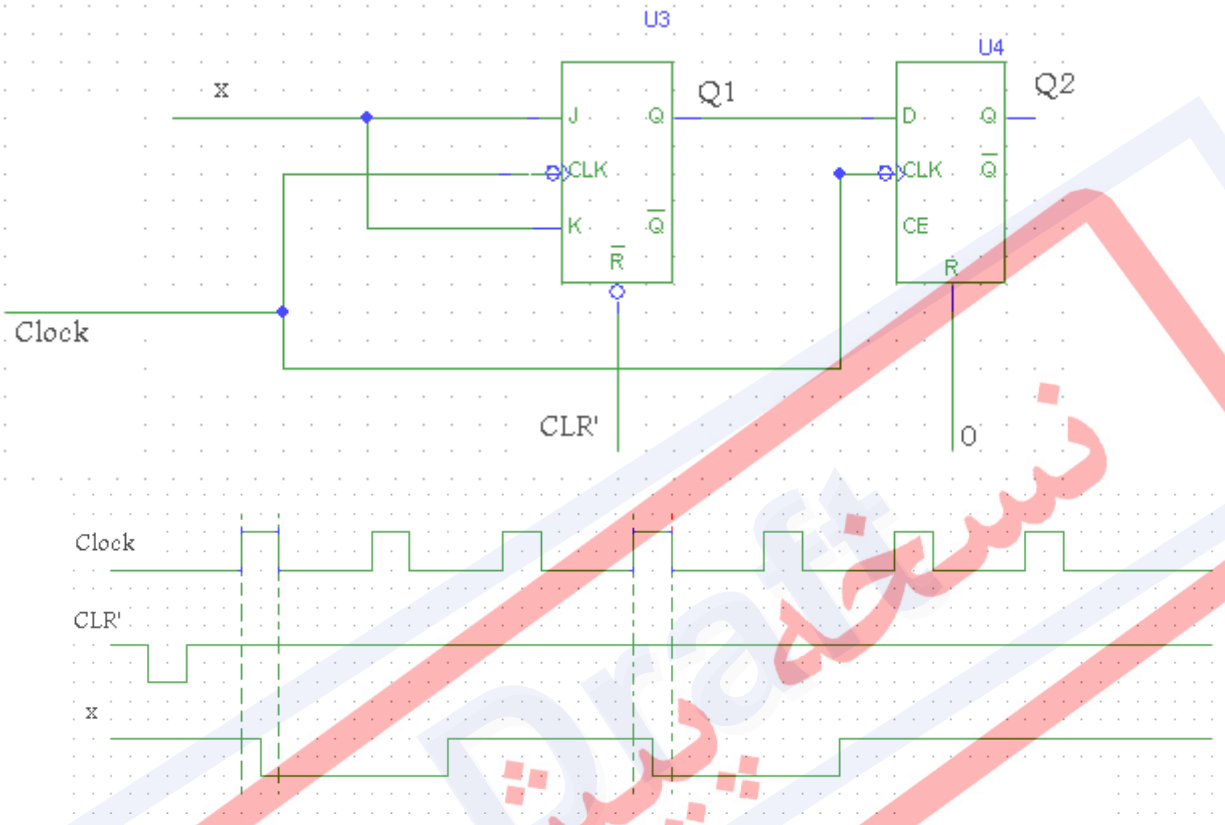


۳-مدار شکل زیر شامل یک لچ **D**، یک فلیپ فلاپ Dحساس به لبه بالا رونده و یک فلیپ فلاپD حساس به لبه پایین رونده است. دیاگرام زمانی شکل زیر را با کشیدن شکل موج سیگنال های y1، y2 وy3 کامل کنید.



نسخه پیش نویس Draft

۴- مدار شکل زیر شامل یک فلیپ فلاپ و یک فلیپ فلاپ است. دیاگرام زمانی شکل زیر را با کشیدن شکل موج های سیگنال های Q1 و Q2 و با فرض اینکه فلیپ فلاپ جک حساس به لبه پایین رونده باشد کامل کنید.



۵-توضیح دهید که چرا حالت موجب می شود که برای یک لچ حالت ناپایدار پیش بیاید.

۶-توضیح دهید چگونه از حالت ناپایدار با به کار گرفتن هر یک از موارد زیر جلوگیری به عمل می آید:

۱) لچ D

۲) لچ JK

۳) فلیپ فلاپ T

۷-یک فلیپ فلاپ JK از نوع master-slave با ورودی های نا همگام present و clear و با استفاده از گیت های NOR طراحی کنید.

نسخه پیش‌نویس Draft

فصل ۳

طراحی مدارهای ترتیبی همگام

Draft
نسخه پیشنویس

۳-۱ طراحي مدارهاي ترتيبي همگام

در این بخش به بحث طراحی مدارهای ترتیبی همگام خواهیم پرداخت. در طراحی این مدارات باید توجه داشت که مدار هرگز نباید به حالت بی تفاوت[1] برود. به عبارت دیگر، مدار باید به صورت رفتار کند، در غیر این صورت، حالت بعدی مدار مشخص نخواهد بود.

نکته) منظور از حالت بی تفاوت در مدارات ترتیبی به شرح ذیل می‌باشد:

۱. مدار هرگز نباید به آن حالت برود.

۲. نباید به ازای یک ورودی خاص، مدار به یک حالت بی تفاوت برود.

۳. فرق نمی‌کند که مدار به این حالت برود یا نرود.

در ادامه، انواع ماشین‌های حالت که برای طراحی این نوع مدارها مورد استفاده قرار می‌گیرند، بررسی خواهند شد.

۳-۲ مدل میلی

در مدل میلی[3]، خروجی‌ها توابعی از ورودی‌ها و حالت فعلی هستند. شکل‌های ۳-۱ الف و ب به ترتیب نمودار حالت و جدول حالت مدل میلی را نشان می‌دهند. مدل میلی را مدار انتساب‌گذر نیز می‌نامند زیرا خروجی مدار همراه گذرهای حالت (کمان‌های نمودار حالت) مشخص می‌شود. به بیان دیگر خروجی‌های مدار توابعی از حالت‌های فعلی و ورودی‌ها به صورت بیان شده با معادلات

$$Z_i = g_i(X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_r) \qquad i=1,\ldots,r$$

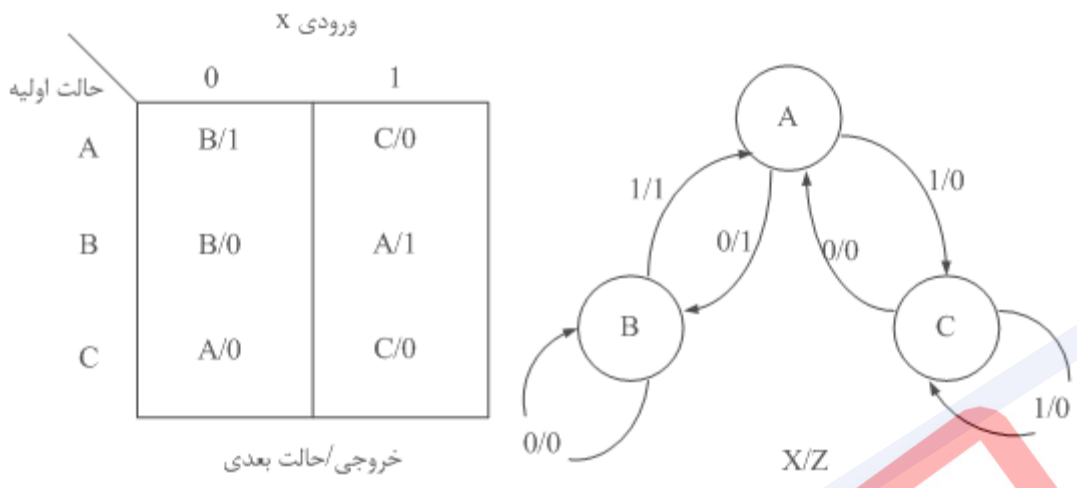
$$Y_j = h_j(X_i, \ldots, X_n, Y_1, \ldots, Y_r) \qquad j=1,\ldots,r$$

هستند. مثال زیر، این رابطه را نشان می‌دهد.

---

[1] Don't Care
[2] Deterministic
[3] Mealy Model

نسخه پیش‌نویس Draft



شکل ۳-۱- مدل میلی. (الف) نمودار حالت. (ب) جدول حالت

مثال ۱) می‌خواهیم خروجی مدار ترتیبی تعریف شده در شکل۳-۱ را به ازای رشتهٔ ورودی زیر بیابیم.

$$X=011010$$

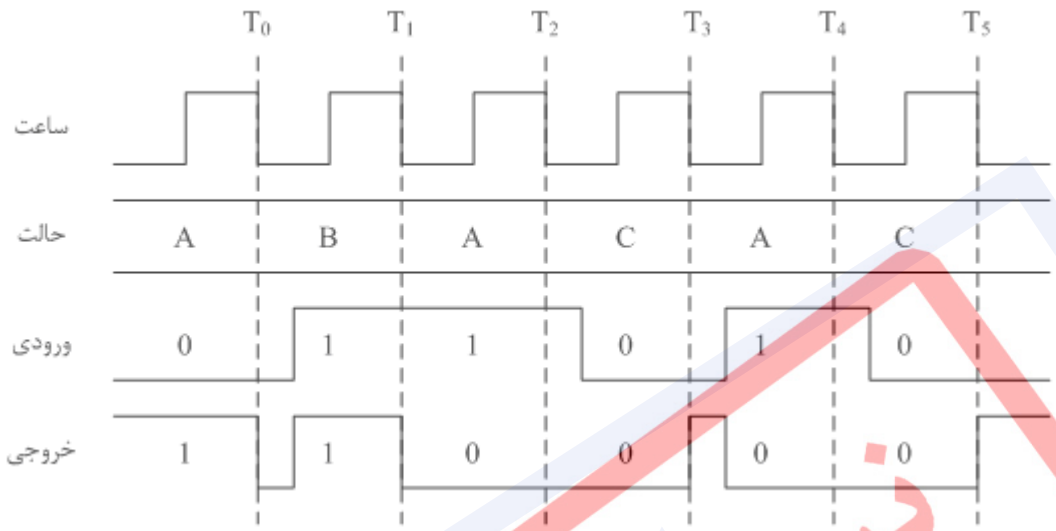
فرض می‌کنیم در زمان ۰ مدار در حالت A قرار دارد و ورودی $X=0$ اعمال می‌شود. با توجه به جدول یا نمودار حالت دیده می‌شود که خروجی $Z=1$ و حالت بعدی برابر B است. به همین ترتیب، رفتار مدار به صورت زیر مشخص می‌شود:

جدول۳-۱ – رفتار مدار به ازاء حالت اولیه A و ورودی 011010

زمان	۰	۱	۲	۳	۴	۵	۶
حالت فعلی	A	B	A	C	A	C	A
ورودی	۰	۱	۱	۰	۱	۰	
حالت بعدی	B	A	C	A	C	A	
خروجی	۱	۱	۰	۰	۰	۰	

پس اگر مدار در ابتدا در حالت A قرار داشته باشد و رشتهٔ ورودی X=011010 به آن داده شود، خروجی $Z=110000$ را تولید کرده و در نهایت در حالت A قرار خواهد گرفت.

نسخه پیش‌نویس Draft



شکل ۳-۲ - زمان بندی مدار مثال ۱

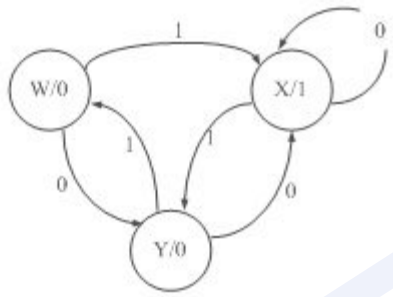
شکل ۳-۲، زمان بندی واقعی سیستم را به ازای رشتهٔ ورودی داده شده نشان می‌دهد. در این نمودار فرض شده است که حالت مدار در گذر سیگنال ساعت از مقدار H به مقدار L ، تغییر می‌کند. باید توجه داشت که خروجی Z می‌تواند چه در هنگام تغییر ورودی و چه در هنگام تغییر حالت مدار، تغییر کند زیرا Z تابعی از هر دوی این مقادیر است. در نتیجهٔ همین عامل، دو تغییر غیر منتظره در خروجی دیده می‌شود. در T_0 با رفتن مدار به حالت B ، خروجی برابر صفر می‌شود ولی با تغییر ورودی به ۱، به مقدار ۱ بر می‌گردد. در حالت T_3 نیز اتفاق مشابهی رخ می‌دهد. پس در مدل میلی باید دقت داشت که خروجی‌ها تنها در حالت پایدار پس از تغییر ورودی، مورد بررسی قرار گیرند.

۳-۳ مدل مور

شکل ۳-۳، آرایش دیگری از نمودار حالت را نشان می‌دهد. این قالب را مدل مور[1] می‌نامند و در آن خروجی تنها به ازای حالت فعلی مدار بیان می‌شود. حرف نشان داده شده در دایره‌ها، نشان دهندهٔ حالت مدار و عدد موجود در دایره‌ها، نشان دهندهٔ خروجی مدار در حالت مربوطه است.

[1] Moore Model

نسخه پیش‌نویس



حالت فعلی \ ورودی x	0	1	خروجی
W	Y	X	0
X	X	Y	1
Y	X	W	0

شکل۳-۳ - مدل مور. (الف) نمودار حالت. (ب) جدول حالت

جدول حالت نیز در مدل مور، دارای قالب جدید است. خروجی را می توان از کنار حالت‌های بعدی به ستون جدیدی منتقل کرد، زیرا برای تمام حالت‌های بعدی، خروجی یکسانی وجود دارد. توجه به این نکته دارای اهمیت است که خروجی‌ها به حالت فعلی مربوط می‌شوند نه به حالت بعدی. برای مدل مور، رابطهٔ تابعی به صورت معادلات زیر بیان می‌شود.

$$Z_i = g_i(Y_1, Y_2, \ldots, Y_r) \qquad i=1,\ldots,m$$

$$\Rightarrow Z = g(y)$$

زیرا خروجی تنها تابعی از حالت فعلی است.

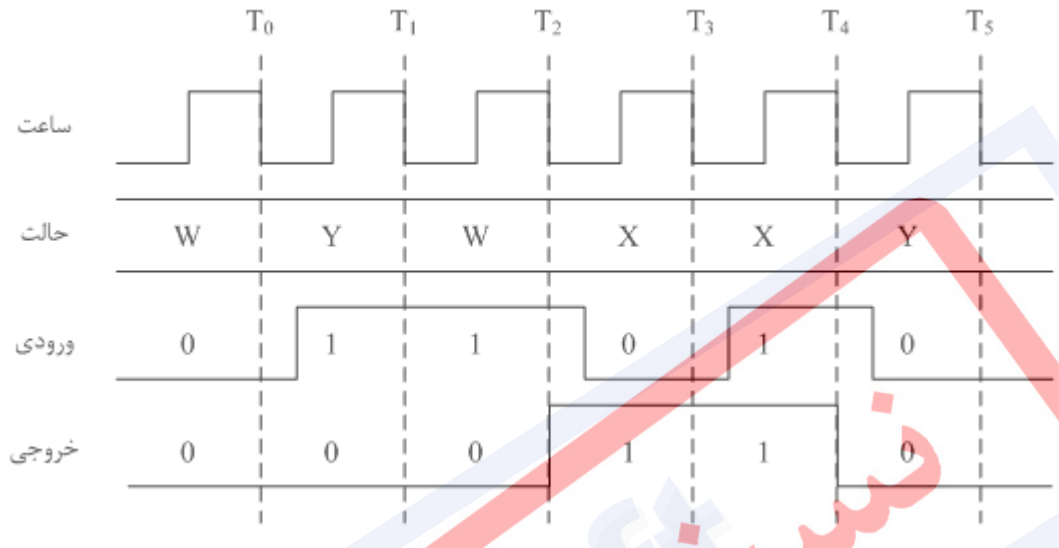
مثال ۲) رشتهٔ ورودی زیر را به مدار بیان شده در شکل ۳-۳، اعمال می‌کنیم. مدار در ابتدا، در حالت W قرار دارد.

جدول ۳-۲ – رفتار مدار به ازاء ورودی X=011010 و شروع از حالت W

زمان	۰	۱	۲	۳	۴	۵	۶
حالت فعلی	W	Y	W	X	X	Y	X
ورودی	۰	۱	۱	۰	۱	۰	
حالت بعدی	Y	W	X	X	Y	X	
خروجی	۰	۰	۰	۱	۱	۰	

نسخه پیش‌نویس Draft

خروجی تنها با توجه به حالت فعلی از روی نمودار حالت یا جدول حالت بدست می‌آید. شکل ۳-۴، نمودار زمان بندی را به ازای رشتهٔ ورودی داده شده نشان می‌دهد. تمام تغییر حالت‌ها در گذرهای سیگنال ساعت از مقدار H به مقدار L رخ می‌دهند.



شکل ۳-۴ - زمان بندی مدار مثال ۲

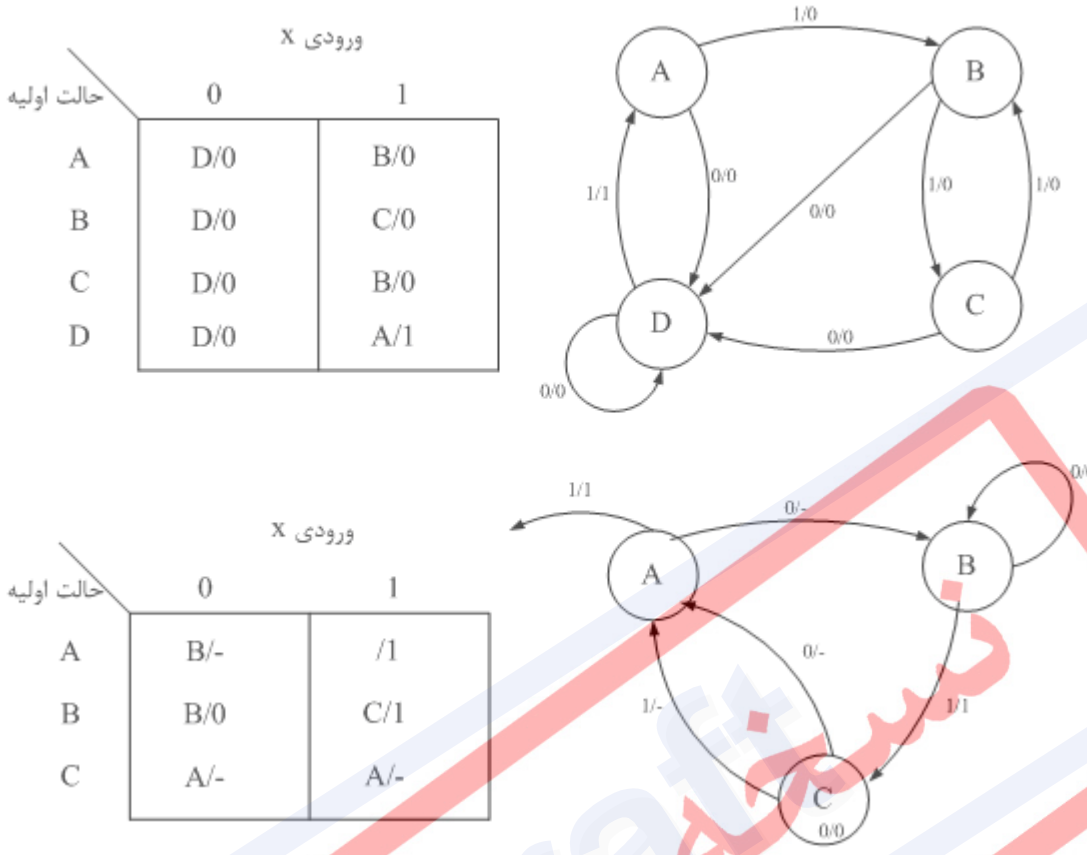
با توجه به شکل بالا مشاهده می‌شود که در مدل مور، تمام تغییرات خروجی با ساعت همگام است زیرا خروجی تنها تابعی از حالت فعلی است و بنابراین تنها هنگامی تغییر می‌کند که حالت مدار تغییر کند. پس به رغم تغییرات ورودی، خروجی پایدار می‌ماند؛ برخلاف آنچه که در مدل میلی رخ می‌دهد. لذا خروجی‌های مدل مور، نوعاً رفتار بهتری نسبت به مدار میلی دارند، یعنی تغییرات ورودی باعث ایجاد تغییرات ناخواسته در خروجی نمی‌شود.

مزیت اصلی استفاده از مدل میلی در طراحی مدارهای ترتیبی این است که چون خروجی این مدل هم به حالت مدار و هم به ورودی‌های آن بستگی دارد، طراح می‌تواند توابع خروجی و گذر حالت را با انعطاف بیشتری طراحی کند و در نتیجه مدار حاصل نسبت به مدار طرح شده بر اساس مدل مور، که در آن خروجی تنها تابعی از حالت مدار است، حالت‌های کمتری خواهد داشت.

۳-۴ روال طراحی مدار ترتیبی همگام

طراحی مدارهای ترتیبی همگام با مشخص کردن جدول یا نمودار حالت مطلوب آغاز می‌شود. مدارهایی که در آنها تمام زوج حالت‌های "خروجی / بعد" ، کاملاً مشخص شده‌اند، مدارهای کاملاً معین نامیده می‌شوند. مدارهایی که چندین حالت بعدی یا خروجی دلخواه دارند، مدارهای دارای تعیین ناکامل نامیده می‌شوند. شکل ۵، نمونه هایی از این دو نوع را نشان می‌دهد.

نسخه پیش‌نویس



شکل ۳-۵ - (الف) مدارهای دارای تعیین کامل. (ب) مدارهای دارای تعیین ناکامل

در ادامه، روش طراحی مدارهای ترتیبی همگام طی چند مثال ساده تشریح می‌شود.

مثال ۳) می‌خواهیم با فیلیپ-فلاپ D ساعت‌دار، مداری با جدول حالت تعریف شده در شکل ۳-۶- الف، بسازیم. ابتدا باید مقادیری برای حالت‌های نشان داده شده با حروف تعیین کنیم. این عمل را انتساب حالت می‌نامند. به این منظور مقادیر نشان داده شده در جدول شکل ۳-۶- ب را برمی‌گزینیم. با گذاشتن مقادیر برگزیده به جای حالت‌های حرفی، جدول حالت دودویی یا جدول گذر شکل ۳-۶- ج بدست می‌آید. جدول گذر، تمام اطلاعات لازم برای یافتن توابع بخش ترکیبی مدار را در بر دارد. سپس، مطابق شکل‌های ۳-۶- د و ۳-۶- ه، جدول گذر را به جدول خروجی و جدول ورودی فیلیپ-فلاپ تقسیم می‌کنیم.

نسخه پیش‌نویس Draft

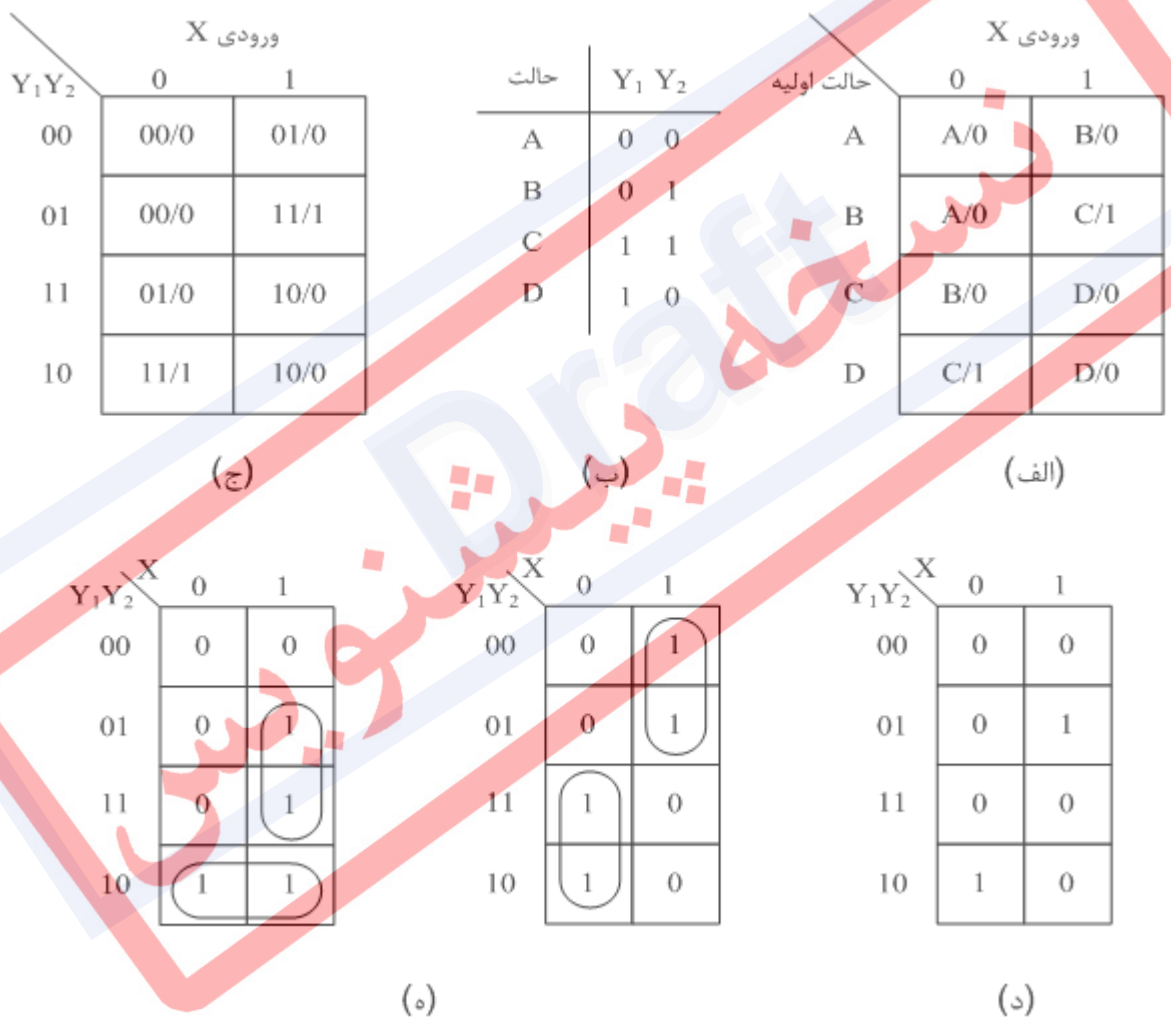
جدول ورودی فیلیپ-فلاپ را جدول تحریک می‌نامند. با توجه به جداول تحریک و خروجی، خواهیم داشت:

$$D_1 = y_1\overline{y_2} + xy_2$$

$$D_2 = x\overline{y_1} + \overline{x}y_1 = x \oplus y_1$$

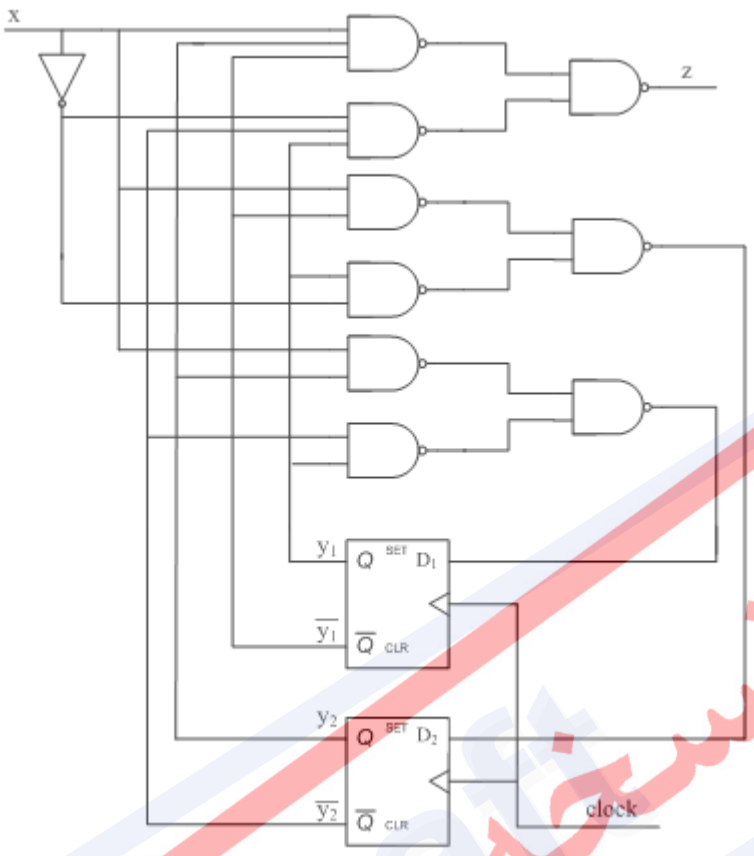
$$Z = x\overline{y_1}y_2 + \overline{x}y_1\overline{y_2}$$

شکل ۳-۷ ، مدار منطقی کامل را نشان می‌دهد. مدار ترکیبی با دو مقطع گیتی NAND ساخته شده است.



شکل ۶ – (الف) جدول حالت (ب) مقادیر منتسب به متغیر ها (ج) جدول حالت دودویی یا جدول گذر (د) جدول خروجی (ه) جداول تحریک

نسخه پیش‌نویس Draft



شکل ۳-۷ - مدار منطقی کامل مثال ۳

در مثال قبل، سؤالاتی مطرح می‌شود از جمله: انتساب حالت به چه صورتی انجام می‌شود؟ اگر بخواهیم مدار را با فیلیپ-فلاپ دیگری بسازیم، چه باید کرد؟ از کجا می‌توان فهمید که جدول حالت داده شده، بهترین صورت بیان مدار است؟ (زیرا بسیاری از جداول حالت، دارای حالت‌های اضافی هستند که می‌توان آنها را حذف کرد). در ادامه تلاش می‌کنیم به این پرسش‌ها پاسخ دهیم.

۳-۵ قدم‌های طراحی مدار ترتیبی همگام

گام ۱ . یافتن جدول حالت با استفاده از توصیف مسئله.

گام ۲ . یافتن مدار معادل مینیمم، با استفاده از روشهای تقلیل حالت.

گام ۳ . به دست آوردن جداول گذر حالت و خروجی، با استفاده از انتساب حالت.

گام ۴ . تعیین نوع حافظه‌ها یا فیلیپ-فلاپ‌های مدار وسپس محاسبهٔ جدول تحریک آنها.

گام ۵ . تعیین معادلات منطقی ورودی فیلیپ-فلاپ‌ها و خروجی مدار، با توجه به جداول تحریک.

نسخه پیش‌نویس Draft

گام ۶ . رسم مدار منطقی ترتیبی با استفاده از معادلات منطقی به دست آمده و عناصر حافظهٔ انتخاب شده.

گام اول، خلاقیت طراح را می‌طلبد و کسب این توانایی نیازمند تجربه است.

گام دوم، با هدف مینیمم کردن عناصر حافظه با حذف حالت‌های اضافی انجام می‌شود. در گام سوم، می‌توان انتساب را بصورت دلخواه انجام داده یا یکی از روش‌های انتساب، که مینیمم مدار ترکیبی لازم را بدست می‌دهد، بکار گرفت.

گام چهارم، تحلیل مشخصات فیلیپ-فلاپ انتخاب شده را طلب می‌کند تا بتوان جداول تحریک فیلیپ-فلاپ مذکور را به دست آورد. گامهای پنجم و ششم واضح بوده و تنها به منظور تکمیل روش بیان شده، ذکر گردیده‌اند.

۳ – ۶ جداول ورودی فیلیپ-فلاپ

جدول گذر، تغییر حالت لازم برای هر فیلیپ-فلاپ حافظه را در بر دارد. برای تعیین ورودی‌های هر یک از فیلیپ-فلاپ‌های حافظه، می‌توان از جدول ورودی فیلیپ-فلاپ استفاده کرد. جداول ورودی شکل ۸، مشخصات هر یک از فیلیپ-فلاپ‌ها را در بر دارند. در این جدول‌ها، نمادهای زیر به کار برده شده است:

t : زمان فعال شدن سیگنال ساعت.

Q(t) : حالت فیلیپ-فلاپ در لحظهٔ فعال شدن سیگنال ساعت.

Q(t +E) : حالت بعدی فیلیپ-فلاپ پس از فعال شدن سیگنال ساعت.

عمدتاً از فیلیپ-فلاپ D استفاده می‌شود چون حالت بعدی آن با ورودی فعلی آن یکسان است و در نتیجه جدول تحریک آن مستقیماً از جدول گذر بدست می‌آید. برای فیلیپ فلاپ های دیگر، باید جدول تحریک با توجه به جداول ورودی شکل ۳-۸ محاسبه گردد.

نسخه پیش‌نویس Draft

تغییرحالت		ورودی های موردنیاز
$Q(t)$	$Q(t+\varepsilon)$	$D(t)$
0	0	0
0	1	1
1	0	0
1	1	1

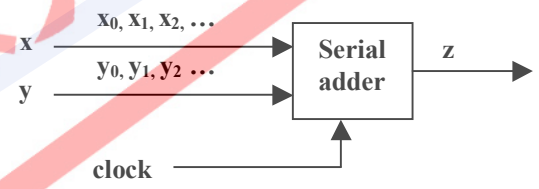
تغییرحالت		ورودی های موردنیاز	
$Q(t)$	$Q(t+\varepsilon)$	$S(t)$	$R(t)$
0	0	0	—
0	1	1	0
1	0	0	1
1	1	—	0

تغییرحالت		ورودی های موردنیاز
$Q(t)$	$Q(t+\varepsilon)$	$T(t)$
0	0	0
0	1	1
1	0	1
1	1	0

تغییرحالت		ورودی های موردنیاز	
$Q(t)$	$Q(t+\varepsilon)$	$J(t)$	$K(t)$
0	0	0	—
0	1	1	—
1	0	—	1
1	1	—	0

شکل ۳-۸ - جداول تحریک انواع فیلیپ-فلاپ‌ها

مثال ۴) می‌خواهیم یک جمع کنندهٔ سریال[1] طراحی کنیم که دو ورودی و یک خروجی دارد. این مدار در هر لحظه، دو بیت ورودی را جمع کرده و در خروجی قرار می‌دهد.



دو حالت q_0 و q_1 را به صورت زیر در نظر می گیریم:

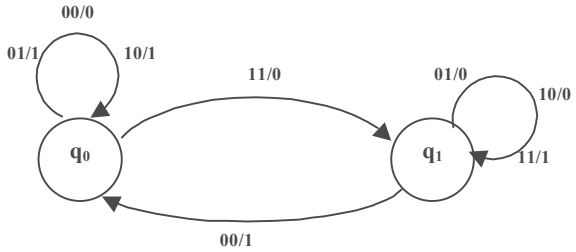
q_0: هنگامیکه رقم نقلی نداریم. (c=0)

q_1: هنگامیکه رقم نقلی داریم. (c=1)

نمودار حالت و جدول حالت این مدار در شکل ۳-۹ نشان داده شده است.

[1] Serial Adder

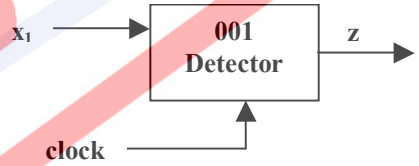
نسخه پیش‌نویس Draft



xy	00	01	11	10
q_0	0 $q_0,0$	2 $q_0,1$	6 $q_1,0$	4 $q_0,1$
q_1	1 $q_0,1$	3 $q_1,0$	7 $q_1,1$	5 $q_1,0$

شکل ۳-۹ - نمودار حالت و جدول حالت مدار جمع کنندهٔ سریال

مثـال ۵) می‌خواهیم مداری بسازیم که دنبالهٔ ورودی ۰۰۱ را تشخیص بدهد، یعنی خروجی فقط هنگامی برابر با یک باشد که دنبالهٔ ۰۰۱ به ورودی داده شود. این مدار یک ورودی و یک خروجی دارد. خروجی در حالت عادی صفر است و در صورت تشخیص دنبالهٔ ۰۰۱ به یک تغییر می‌کند و به محض آمدن ورودی بعدی، (با سیگنال ساعت) دوباره صفر می‌شود.



با توجه به توضیحات بالا، مشخص است که مدار سنکرون و دارای سیگنال ساعت بوده و ورودی‌ها بصورت سریال وارد می‌شوند. گامهای لازم برای طراحی یک مدار ترتیبی را که پیش از این ذکر شد، برای این مدار طی می‌کنیم. این مراحل عبارتند از:

- تحلیل مدار
- سنتز مدار

نسخه پیش‌نویس Draft

۱) تحلیل مدار

در ابتدا به تعریف حالات می‌پردازیم.

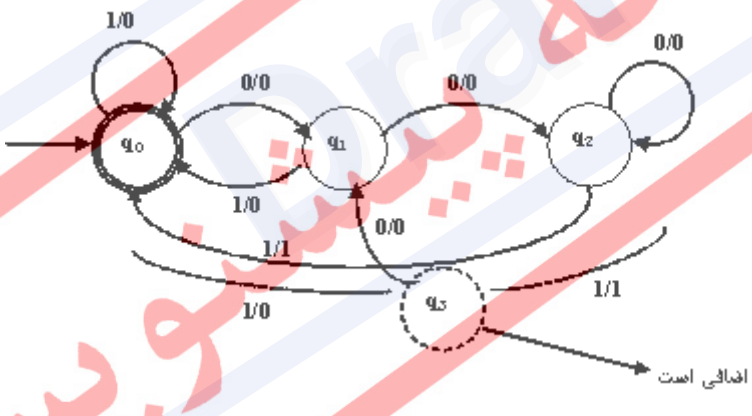
q_0 : حالت شروع.

q_1 : اولین ورودی صفر دیده شد.

q_2 : دومین ورودی صفر دیده شد.

q_3 : اولین بیت یک بعد از دو ورودی متوالی صفر دیده شد.

در شکل ۳-۱۰، دیاگرام مربوط به حالت های مدار مشاهده می‌شود. همان طور که در شکل دیده می‌شود، حالت q_3 را می‌توان حذف کرد چرا که این حالت بیانگر زمانی است که دنباله تشخیص داده شده باشد. می‌دانیم که مدار می‌بایست پس از هر بار تشخیص دنباله Reset شود. پس می‌توان به جای تعریف حالت جدید، به همان حالت اولیه مدار بازگشت و خروجی را نیز یک کرد. در کنار دیاگرام حالت، جدول حالت نیز نشان داده شده است. این جدول در واقع بیان و توصیف دیگری از مدار است.



X	0	1
q_0	q_1, 0	q_0, 0
q_1	q_2, 0	q_0, 0
q_2	q_2, 0	q_0, 1

شکل ۳-۱۰ - دیاگرام و جدول حالت مدار تشخیص دنبالهٔ ۰۰۱

نسخه پیش‌نویس Draft

۲) سنتز مدار

طبق تعریف، تبدیل توصیف یک مدار به یک مدار سخت‌افزاری را سنتز گویند که خود شامل مراحل زیر است:

- تخصیص حالت (State Assignment)

تعداد فیلیپ-فلاپ‌ها = (تعداد بیت‌ها) $n = [\log_2$ تعداد حالت $]$

نکتهٔ ۱) نحوهٔ تخصیص حالت از نظر عملکرد مدار تفاوتی نمی‌کند.

نکتهٔ ۲) نحوهٔ تخصیص حالت در هزینهٔ مدار مؤثر است. (هزینهٔ پیاده‌سازی مدار)

جدول ۳-۳ دو تخصیص حالت مختلف

حالت	تخصیص حالت ۱	تخصیص حالت ۲
q_0	01	00
q_1	10	01
q_2	00	10

- تشکیل جدول حالت به کمک حالت‌های تخصیص داده شده (Y – Z Matrix)

در این قسمت، همان جدول حالت شکل 3-10 با تخصیص حالتی که در قسمت قبل انجام گرفته است، بازنویسی می‌شود.

جدول ۳ –۴ بازنویسی جدول حالت شکل ۳-۱۰ با تخصیص حالت ۱ از جدول ۳-۳

		0	1
01	q_0	10, 0	01, 0
10	q_1	00, 0	10, 0
00	q_2	00, 0	01, 1
11	q_3	-, -	-, -

- تعیین نوع فیلیپ-فلاپ مورد نظر

نکتهٔ۱) عملکرد مدار مستقل از نوع فیلیپ-فلاپ است.

نکتهٔ۲) خروجی مدار مستقل از نوع فیلیپ-فلاپ است.

نسخه پیش‌نویس Draft

- پیدا کردن جدول تحریک[1] فیلیپ-فلاپ از روی Y-Z Matrix ()

این جدول بیان می‌کند که ورودی‌های فیلیپ-فلاپ‌ها می‌بایست چگونه باشند تا تغییرات مورد نظر اعمال گردد. جدول تحریک، کاملاً وابسته به نوع فیلیپ-فلاپ است و برای تک تک ورودی‌های هرکدام از فیلیپ-فلاپ‌ها نیز باید یک جدول تحریک جداگانه محاسبه شود.

- رسم جداول کارنو/ جدول بندی

تمام ورودی‌های فیلیپ-فلاپ‌ها که در بخش قبل به دست آمد، به کمک جداول کارنو ساده می‌شوند. (برای هر ورودی فیلیپ-فلاپ، به یک جدول کارنو نیاز هست.) از جداول کارنو برای ساده‌سازی خروجی‌ها نیز استفاده می‌شود.

- پیاده‌سازی مدار حاصل (شبیه‌سازی مدار حاصل)

در این مرحله، مدار به کمک گیت‌های منطقی پیاده‌سازی می‌شود. با توجه به هزینه و زمان زیاد لازم برای انجام این کار، راه حل جایگزین شبیه‌سازی به کمک ابزارهای شبیه‌ساز است که توسط آنها می‌توان صحت عملکرد مدار طراحی شده را بررسی نمود[2].

- نگاشت به کتابخانه[3]

در این مرحله تمام گیت‌های منطقی مورد استفاده در مرحلهٔ سنتز، به وسیلهٔ عناصر کتابخانه‌ای موجود، پیاده‌سازی می‌شوند. برای مثال، به منظور پیاده‌سازی گیت NOT که در کتابخانه وجود ندارد، از گیت NOR یا NAND که در کتابخانه موجود است، استفاده شود.

[1] Excitation Table
[2] Verification
[3] Library Mapping

نسخه پیش‌نویس Draft

مثـال ۶) در این مثال تمام مراحل طی شده در مثال قبل، مجدداً انجام گرفته و مورد بررسی قرار می‌گیرد.

مرحلهٔ ۱ – تخصیص حالت در این مثال به صورت زیر انجام گرفته است. کد حالت‌ها را در ستون سمت چپ جدول ملاحظه می‌کنید.

جدول ۳-۵ – تخصیص حالت انجام گرفته برای مثال ۶

حالت اولیه		ورودی X: 0	ورودی X: 1
000	q_1	q_2, 0	q_3, 0
111	q_2	q_5, 1	q_4, 1
011	q_3	q_4, 0	q_2, 1
101	q_4	q_1, 1	q_5, 0
100	q_5	q_5, 1	q_3, 0

مرحلهٔ ۲ – در این مرحله جدول را با کدهای تخصیص داده شده در قسمت قبل بازنویسی می‌کنیم.

جدول ۳-۶ – بازنویسی جدول با استفاده از کدهای تخصیص یافته

	y_1	y_2	y_3	0	1
q_1	0	0	0	111, 0	011, 0
	0	0	1	-, -	-, -
	0	1	0	-, -	-, -
q_3	0	1	1	100, 0	111, 1
q_4	1	0	0	000, 1	101, 0
q_5	1	0	1	101, 1	011, 0
	1	1	0	-, -	-, -
q_2	1	1	1	101, 1	100, 1

مرحلهٔ ۳ – در حل این مسئله از فیلیپ-فلاپ نوع T استفاده شده است.

نسخه پیش‌نویس

مرحلهٔ ۴ – در این قسمت، جداول تحریک فیلیپ-فلاپ‌ها را بدست می‌آوریم.

جدول ۳-۷ – جداول تحریک فلیپ فلاپ‌ها

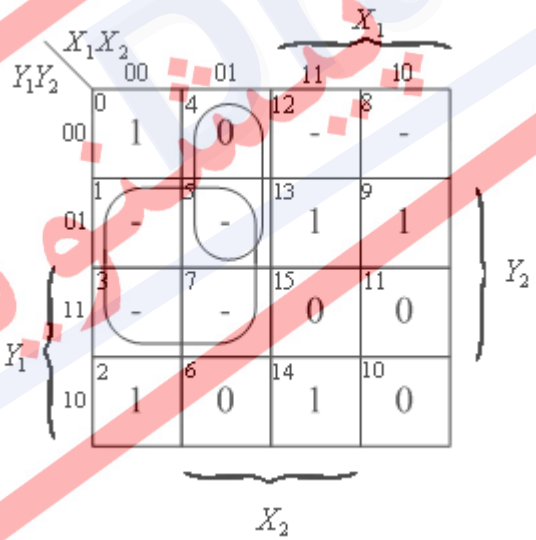
	0	1
000	111, 0	011, 0
001	-, -	-, -
010	-, -	-, -
011	111, 0	100, 1
100	100, 1	001, 0
101	000, 1	110, 0
110	-, -	-, -
111	010, 1	011, 1

$\lfloor \log_2^8 \rfloor = 3$

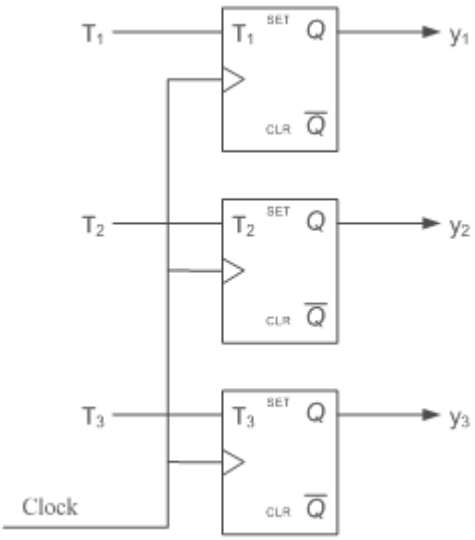
تعداد فیلیپ-فلاپ‌ها

(تعداد بیت‌ها)

مرحلهٔ ۵ – جداول تحریک حاصل از قسمت قبل را به وسیلهٔ جدول کارنو به عبارات منطقی ساده شده تبدیل می‌کنیم.



نسخه پیش نویس
Draft



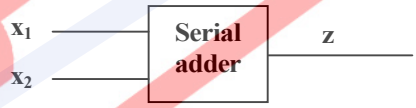
$$T_1 = \bar{x}\bar{y}_3 = \bar{y}_1 y_2 + x y_3 \bar{y}_2$$

شکل ۱۱ – مدار منطقی نهایی

مدار نهایی مشابه شکل ۱۱ خواهد بود

مثـال ۷) سنتز مدار جمع کنندهٔ سریال که پیش از این در مثال‌های قبل، طراحی شده است.

همان طور که در زیر ملاحظه می‌کنید، مدار دارای دو حالت است. این حالت‌ها به ازای مقادیر بیت نقلی تعریف شده‌اند.



q_0: بیت نقلی صفر C=0

q_1: بیت نقلی یک C=1

نسخه پیش‌نویس
Draft

y \ X_1X_2	00	01	11	10
0	0 $q_0,0$	2 $q_0,1$	6 $q_1,0$	4 $q_0,1$
1	1 $q_0,1$	3 $q_1,0$	7 $q_1,1$	5 $q_1,0$

مرحلهٔ ۱ – تخصیص حالت.

0	
1	

مرحلهٔ ۲ – بازنویسی جدول حالت.

y \ X_1X_2	00	01	11	10
0	0 0,0	2 0,1	6 1,0	4 0,1
1	1 0,1	3 1,0	7 1,1	5 1,0

مرحلهٔ ۳ – برای این مثال، از یک فیلیپ-فلاپ نوع D استفاده می‌کنیم. توجه کنید که استفاده از این نوع فیلیپ-فلاپ، به خوانا شدن مدار کمک می‌کند، چرا که در ساخت این مدار به ذخیره سازی بیت نقلی نیاز داریم و این فیلیپ-فلاپ می‌تواند این کار را به وضوح انجام دهد.

مرحلهٔ ۴ – به دست آوردن جدول تحریک فیلیپ-فلاپ

نسخه پیش‌نویس Draft

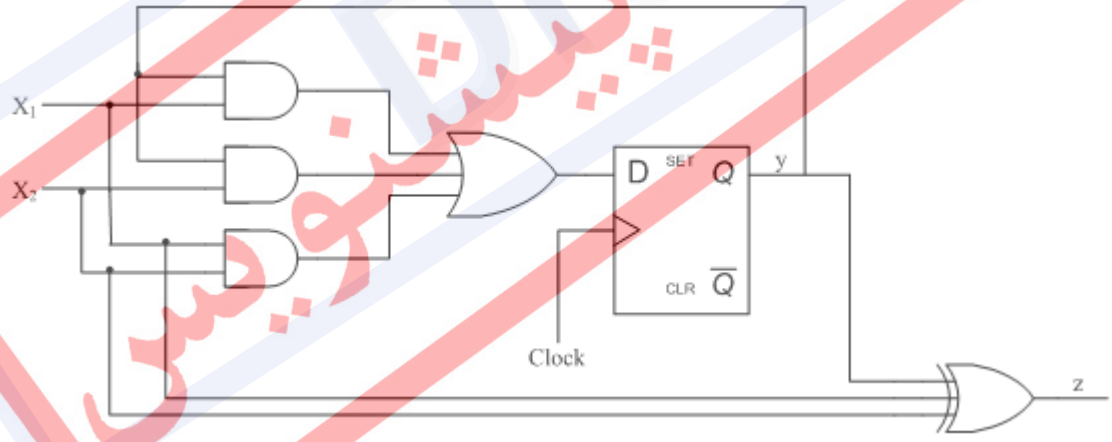
y \ x_1x_2	00	01	11	10
0	0	0	1	0
1	0	1	1	1

مرحلهٔ ۵ – رسم جدول کارنو.

y \ x_1x_2	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$D = x_1 y + x_2 y + x_1 x_2$$
$$Z = x_1 \oplus x_2 \oplus y$$

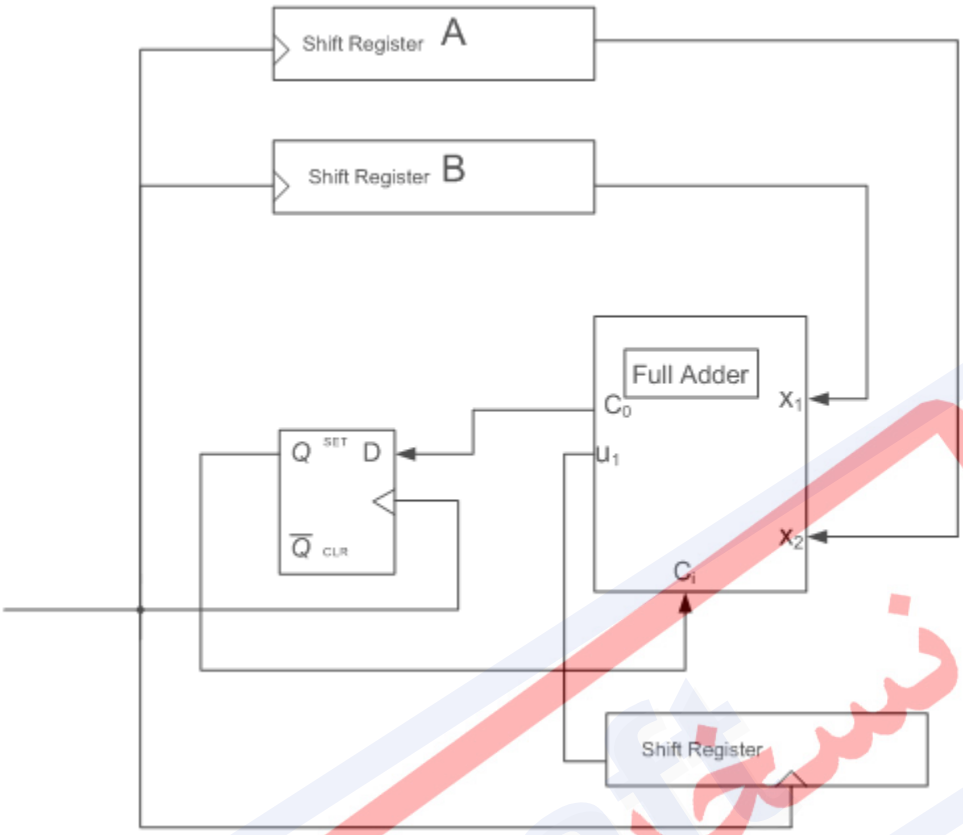
در نهایت، مدار زیر حاصل می‌شود:



حال می‌خواهیم مرحلهٔ آخر، یعنی نگاشت به کتابخانه را، به مدار بالا اعمال کنیم. فرض می‌کنیم فقط عناصر تمام جمع کننده و فیلیپ-فلاپ نوع D را در اختیار داریم. مدار فوق بصورت شکل ۱۲ در می‌آید. البته برای دادن ورودی به مدار و همچنین به دست آوردن نتیجهٔ جمع، به شیفت رجیستر[1] نیاز داریم. بنابراین اگر پریود کلاک برابر t باشد، جمع دو عدد n بیتی، n کلاک طول می‌کشد ($T = n \times t$).

[1] Shift Register

نسخه پیش‌نویس Draft

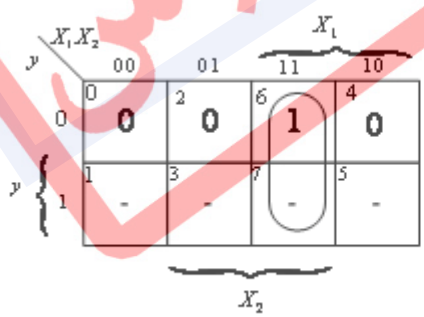


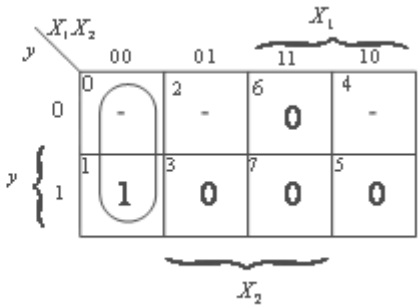
شکل 3-12 – مدار منطقی حاصل بعد از نگاشت به کتابخانه

مثـال ۸) سنتز مدار قبلی (جمع کنندهٔ سریال) با استفاده از JK فیلیپ-فلاپ.

مراحل ۱ و ۲ فرقی نمی‌کند.

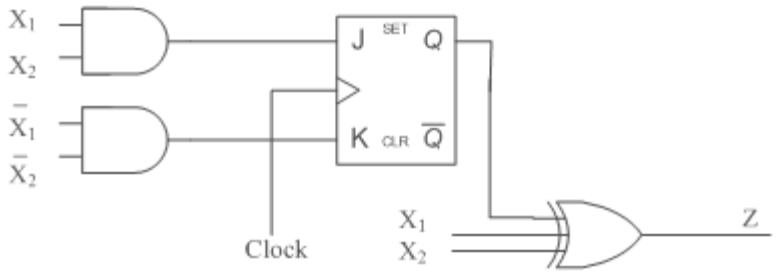
مرحلهٔ ۳ – می‌خواهیم ببینیم با تغییر نوع فیلیپ-فلاپ، چه تغییراتی در روند طراحی ایجاد می‌شود. به همین منظور از فیلیپ-فلاپ JK استفاده می‌کنیم.





$$Z = x_1 \oplus x_2 \oplus y$$

نسخه پیش‌نویس Draft



با مقایسه مدار حاصل از طراحی به وسیله JK-FF و D-FF به وضوح دیده می‌شود که استفاده از D-FF باعث پیچیدگی بیشتر مدار می‌شود. ولی این نکته را باید در نظر گرفت که تحلیل مداری که با D-FF طراحی شده است، بسیار ساده‌تر است.

مثـال ۹) می‌خواهیم جدول حالت زیر را با هر دو تخصیص حالت A و B سنتز نماییم. برای این منظور یک بار از T-FF و یک بار هم از D-FF استفاده می‌کنیم.

جدول 3-۸ - جدول حالت و تخصیص حالت‌های مختلف مربوط به مثال ۹

حالت اولیه \ x	0	1
q_1	q_3	q_2
q_2	q_4	q_1
q_3	q_1	q_3
q_4	q_2	q_3

	State Ass. A		State Ass. B	
q_1	0	0	0	0
q_2	0	1	1	1
q_3	1	0	1	0
q_4	1	1	0	1

طراحی با فیلیپ-فلاپ T:

تخصیص حالت A

$$T_1 = \bar{x}$$

$$T_2 = x(\bar{y}_1 + y_2)$$

تخصیص حالت B

$$T_1 = \bar{x} + \bar{y}_1 + y_2$$

$$T_2 = x(\bar{y}_1 + y_2)$$

همانطور که در جدول بالا مشاهده می‌شود، واضح است که در صورت استفاده از T-FF تخصیص حالت A بهتر از B است. چراکه دارای پیچیدگی و هزینه کمتری است. (تعداد گیت‌های کمتر و ساده‌تری در آن بکار رفته است)

نسخه پیش‌نویس Draft

طراحی با فیلیپ-فلاپ D:

تخصیص حالت A	تخصیص حالت B
$D_1 = \bar{x}\bar{y}_1 + xy_1$	$D_1 = \bar{y}_1 + x\bar{y}_2$
$D_2 = \bar{x}y_2 + x\bar{y}_1\bar{y}_2$	$D_2 = \bar{x}y_2 + x\bar{y}_2\bar{y}_2$

در اینجا ملاحظه می کنید که برعکس حالت قبل که تخصیص حالت A بهتر بود، تخصیص حالت B بهتر است. چراکه هزینهٔ کمتری دارد. این تغییر به خاطر نوع فیلیپ-فلاپ‌های مورد استفاده در طراحی است. می‌توان نتیجه گرفت که برای یک طراحی کم هزینه، انتخاب نوع فیلیپ-فلاپ‌ها دارای اهمیت زیادی است.

۳-۷ کنترل کنندهٔ محدود-حالت

کاربرد عمدهٔ مدارهای کنترل، پاسخگویی به سیگنال‌های خروجی یک مدار یا شرایط به وجود آمده در داخل آن مدار است. مشخصهٔ این مدارها، محدود بودن حالت‌های آنها است. به همین خاطر، کنترل کنندهٔ محدود-حالت نامیده می‌شوند. یکی از متداول‌ترین کاربردهای این کنترل کننده‌ها، در واحد کنترل کامپیوترها و دیگر سیستم‌های دیجیتالی است. این سیستم‌ها دارای دو بخش اصلی می‌باشند: مسیر داده و واحد کنترل. در مسیر داده، عملیاتی چون عملیات ریاضی و تبدیل‌های دیگر بر روی داده‌ها انجام می‌شود. مسیر داده‌ها معمولاً از واحدهای ترکیبی مانند واحد ریاضی-منطقی و مالتی پلکسر تشکیل می‌شود و در آنها برای ذخیره کردن داده‌ها، از ثبات استفاده می‌گردد. واحد کنترل، به مسیر داده فرمان می‌دهد که چه عملی باید انجام شود. فرمان‌ها باید به گونه‌ای باشند که عملیات لازم در پاسخ به ورودی‌ها و شرایط مختلف مدار، انجام گیرند. در طراحی واحد کنترل، می‌بایست ورودی‌ها و خروجی‌های آن معلوم شده و الگوریتم کنترل به صورت نمودار حالت نوشته شود.

مثال ۱۰) می‌خواهیم برای یک روبات مسیریاب، کنترل کنندهٔ محدود-حالت طراحی کنیم. این روبات در اتاقی قرار دارد که موانعی در آن قرار داده شده است. هدف این روبات، پیدا کردن مسیری به خارج از اتاق است. برای رسیدن به این هدف، روبات دارای حس‌گری است که هنگام رسیدن به مانع، خروجی $x=1$ را تولید می‌کند. روبات دارای ۲ خط کنترلی است: $Z_1=1$ که روبات را به سمت چپ

می‌چرخاند و $Z_2=1$ که روبات را به سمت راست می‌چرخاند. هر بار روبات به مانعی برخورد می‌کند، به سمت راست پیچیده و آنقدر به حرکت خود ادامه می‌دهد تا مجدداً به مانعی برخورد کند. این بار به سمت چپ چرخیده و حرکت خود را تا رسیدن به مانع بعدی، ادامه می‌دهد و باز به همین ترتیب (شکل۳- ۱۳).

روبات دارای ۴ حالت به شرح زیر است:

حالت A : مانعی نیست و گردش قبلی به سمت چپ بوده است.

حالت B : مانع وجود دارد و باید به سمت راست گردش کند.

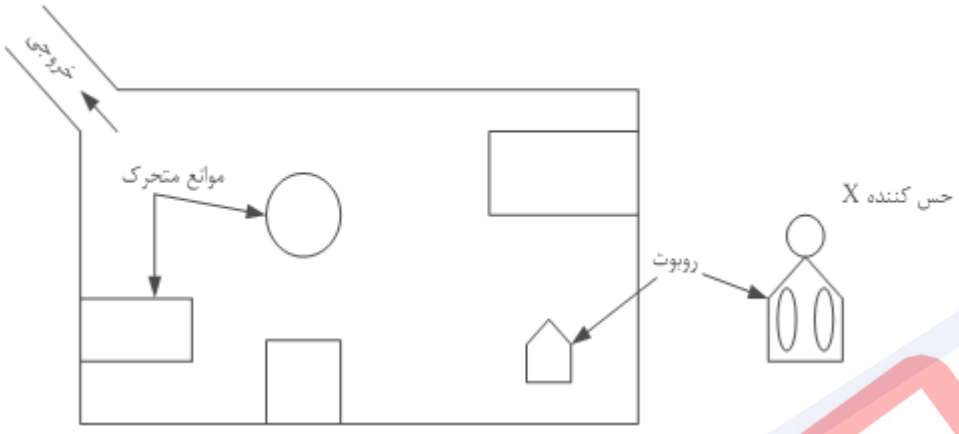
حالت C : مانعی نیست و گردش قبلی به سمت راست بوده است.

حالت D : مانع وجود دارد و باید به سمت چپ گردش کند.

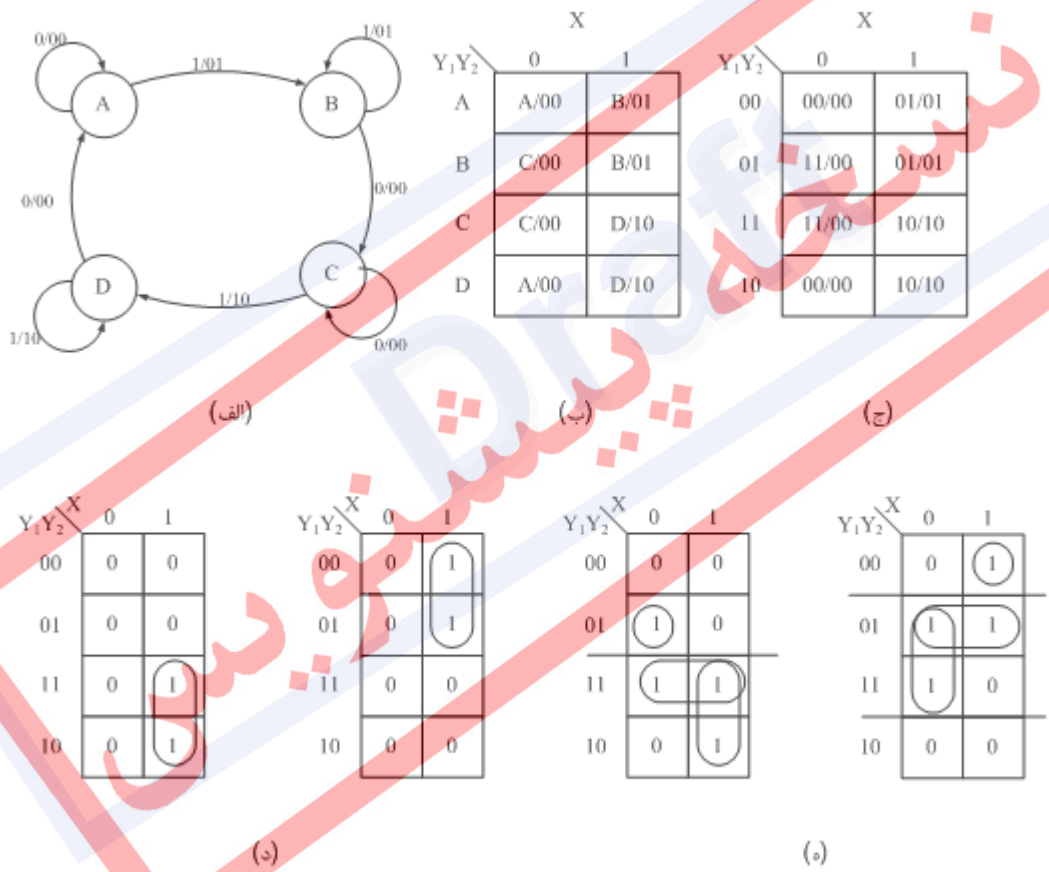
شکل 3-۱۴، نمودار حالت واحد کنترل را نشان می‌دهد. توجه کنید که کنترل کننده در حالت A باقی می‌ماند و روبات دور نمی‌زند تا زمانیکه وجود مانعی تشخیص داده شود. سپس به حالت B رفته و آنقدر به راست می‌پیچد تا دیگر مانعی وجود نداشته باشد. آنگاه به حالت C می‌رود و دیگر نچرخیده، به صورت مستقیم به راه خود ادامه می‌دهد.کنترل کننده در حالت C باقی می‌ماند تا این که دوباره به مانعی برخورد کند. در این هنگام، وارد حالت D شده و آنقدر به سمت چپ می‌چرخد تا دیگر مانعی وجود نداشته باشد. آنگاه وارد حالت A می‌شود.

جدول حالت در شکل 3-۱۴ نشان داده شده است. انتساب حالت را می‌توان به صورت A=00, B=01, C=11, D=10, برگزید. شکل، جدول گذر دودویی را نشان می‌دهد. از این جدول، جداول خروجی Z_1 و Z_2 به دست می‌آید که به کمک آنها می‌توان معادلات خروجی زیر را یافت.

نسخه پیش‌نویس



شکل3-13-روبات مسیریاب



شکل **3-۱۴**- طراحی کنترل کنندهٔ روبات. (الف) نمودار حالت. (ب) جدول حالت. (ج) جدول گذر. (د) جدول‌های خروجی. (ه) جدول‌های تحریک.

برای طراحی، از فیلیپ-فلاپ‌های D استفاده می‌کنیم.

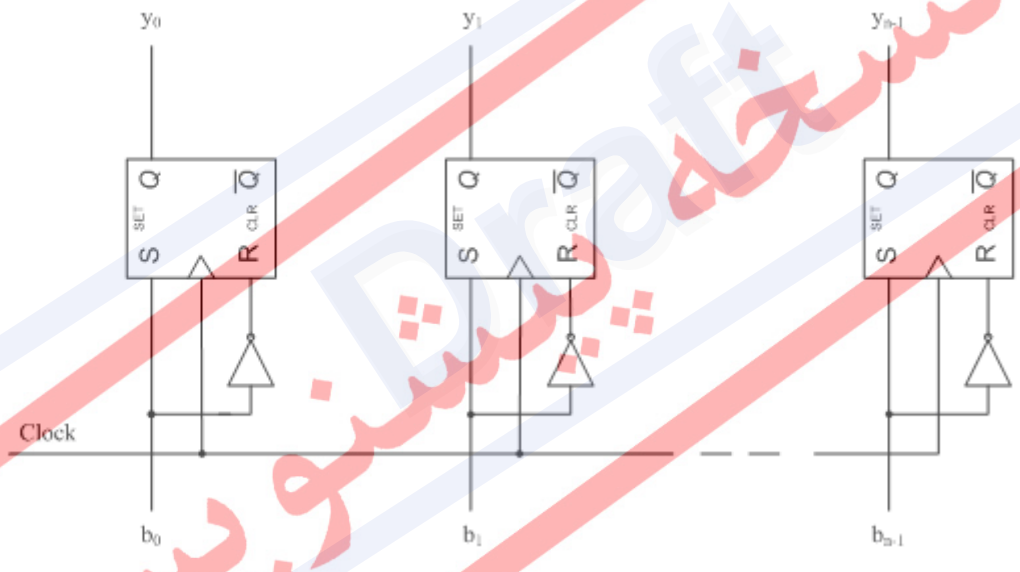
نسخه پیشنویس Draft

$$Y_1 = (xy_2)y_1 + (x + y_2)y_1$$
$$D_1 = (xy_2)\bar{y}_1 + (x + y_2)y_1$$
$$Y_2 = (xy_1)y_2 + (x + y_1)y_2$$
$$D_2 = (x\bar{y}_1)\bar{y}_2 + (\bar{x}y_1 + \bar{y}_1)y_2$$

۳-۸ طراحی های ویژه (ساختارهای پر استفاده)

همانطور که در مثال‌های قبل مشاهده شد، تعدادی از مدارهای ترتیبی همگام وجود دارند که بسیار پراستفاده هستند و در اغلب طراحی‌ها بصورت قسمت‌های آماده مورد استفاده قرار می‌گیرند. در این قسمت به معرفی تعدادی از آنها می‌پردازیم.

۳-۸-۱ رجیستر



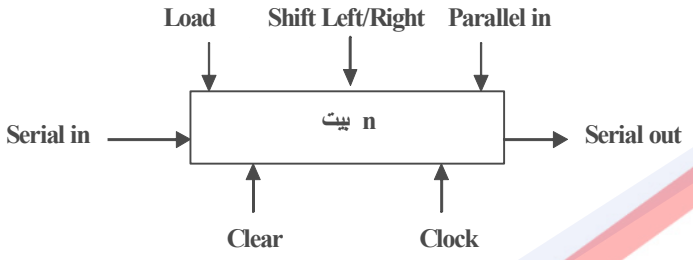
شکل 3-15 رجیستر n بیتی

آنچه رجیسترها را از هم مجزا می‌کند، ظرفیت ذخیره سازی بر حسب بیت است. برای ساختن یک رجیستر n بیتی، به n فیلیپ-فلاپ نیاز داریم. برای مثال، در شکل بالا برای ساخت رجیستر، از RS-FF استفاده شده است. این رجیستر، ساده‌ترین نوع رجیستر می‌باشد. ورودی رجیستر برابر $b_0b_1\ldots b_{n-1}$ و خروجی آن برابر $y_0y_1\ldots y_{n-1}$ می‌باشد.

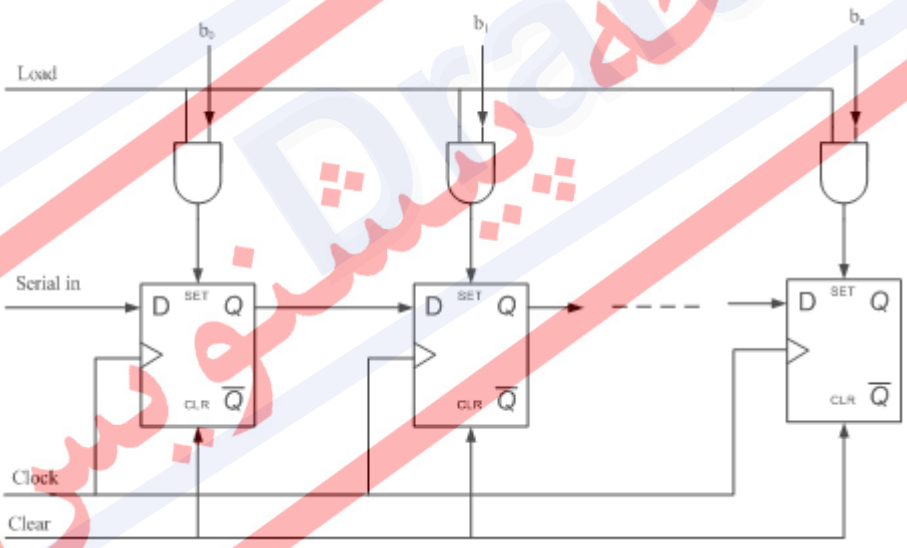
نسخه پیش‌نویس Draft

۳-۸-۲ شیفت رجیستر

از نظر ساختار بسیار شبیه رجیستر است، با این تفاوت که امکان خروجی به صورت سریال را نیز دارد. یعنی می‌تواند دادهٔ ذخیره شده را از راست یا چپ به خروجی هدایت کند. دیاگرام کلی یک شیفت رجیستر در شکل زیر مشخص شده است.



بیشترین کاربرد این مدار در تبدیل سریال به موازی یا موازی به سریال است. یک نمونه از کاربردهای آن در جمع کنندهٔ سریال نشان داده شده است. برای فهم بیشتر از نحوهٔ عملکرد این مدار، یک نمونهٔ مدار داخلی آن را که در شکل ۱۶ آمده است، بررسی می‌نماییم.



شکل 3-۱۶- شیفت رجیستر

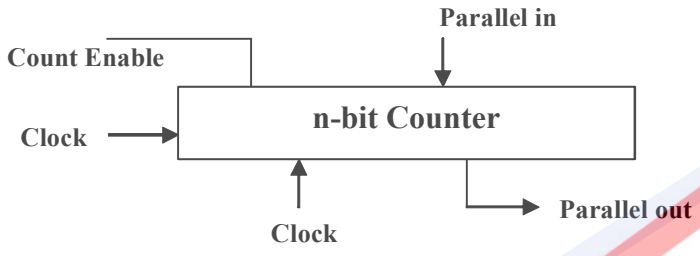
نکتهٔ مهم در این مدار، نحوهٔ بارکردن[1] ورودی است. برای این کار ابتدا باید پایه Clear را فعال کرد تا مقدار همهٔ فیلیپ-فلاپ‌ها صفر شود. سپس Clear را غیر فعال کرده و بعد از اعمال ورودی، پایهٔ Laod را فعال می‌کنیم. به این ترتیب مقادیر ورودی به درستی وارد رجیستر می‌شوند. بعد از غیرفعال کردن پایه Load امکان عمل شیفت وجود خواهد داشت.

[1] Load

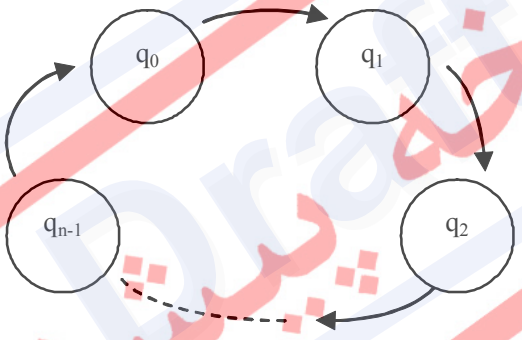
نسخه پیش‌نویس Draft

۳-۸-۳ شمارنده‌ها

مداری که برای تولید یک دنبالهٔ عددی استفاده می‌شود، شمارنده[1] نام دارد. شمارنده‌های ساده، فقط شمارش ترتیبی انجام می‌دهند که البته بیشترین استفاده را نیز دارند. اما می‌توان آنها را به ترتیب دلخواه تغییر داد. دیاگرام کلی یک شمارنده در شکل زیر آمده است.



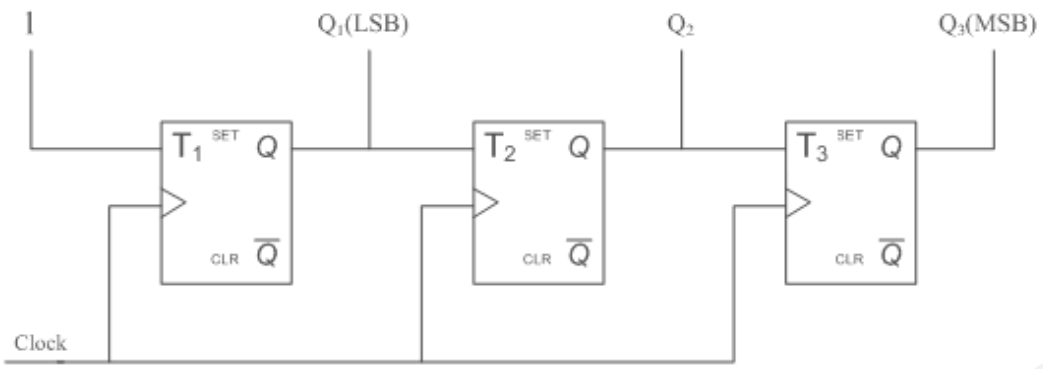
دیاگرام حالت یک شمارنده ترتیبی بسیار ساده است و فقط گذرهای رو به جلو دارد. معمولاً در حالت نهایی، شمارنده به حالت شروع بازمی‌گردد. حالت کلی آن در شکل زیر مشخص شده است.



معمولاً برای ساختن شمارنده‌ها از T-FF استفاده می‌شود. به این دلیل که به کمک این فیلیپ-فلاپ می‌توان براحتی عمل تبدیل ۰ به ۱ و برعکس را که سازندهٔ دنبالهٔ ترتیبی است، ایجاد نمود. یک مثال ساده از یک جمع کنندهٔ سه بیتی در شکل ۱۷ نشان داده شده است. همانطور که ملاحظه می‌شود، بیت کم ارزش بطور مستقیم به ۱ وصل شده است. دنبالهٔ شمارش این مدار بین صفر تا هفت خواهد بود.

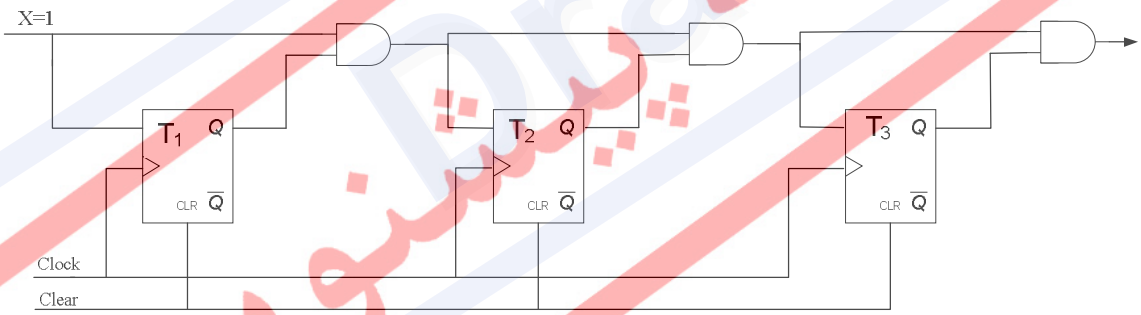
[1] Counter

نسخه پیش‌نویس Draft



شکل 3-۱۷- شمارنده

یک سیگنال پراستفاده در شمارنده‌ها، فعال ساز شمارش[1] است. در شکل ۱۸، X معادل سیگنال فعال ساز شمارش است. همانطور که ملاحظه می‌کنید اگر مقدار این سیگنال برابر ۱ باشد، شمارنده مانند قبل عمل می‌کند. اگر مقدار آن برابر ۰ شود، ورودی همهٔ فیلیپ-فلاپ‌ها صفر شده و بنابراین شمارنده از شمارش می‌ایستد. روش این چنینی برای متوقف کردن شمارش بسیار بهتر از گذاشتن یک گیت کنترلی بر سر راه کلاک مدار است. اصولاً در مدارات همگام همواره از گذاشتن گیت در سر راه سیگنال ساعت پرهیز می‌شود چرا که عواقب زیادی را به همراه خواهد داشت.



شکل 3-۱۸ - شمارنده با سیگنال فعال‌کننده

معادلات این شمارنده برای حالت n بیتی بصورت زیر خواهد بود:

$$T_1 = X$$
$$T_2 = XQ_1$$
$$T_3 = XQ_1.Q_2$$
$$\vdots$$
$$T_n = XQ_1Q_2\cdots Q_{n-1}$$

[1] Count Enable

نسخه پیش‌نویس Draft

۳ – ۹ تمرین

۱– T_2 و T_3 و Z را از جدول کارنو مثال ۶ بدست آورید و مدار نهائی را رسم کنید.

۲–با فلیپ–فلاپ نوع D مدار مثال ۶ را دوباره سنتز کنید و دو مدار حاصل را با یکدیگر مقایسه نمایید.

۳ -در کتاب‌های Data Sheet (TTL) به دنبال رجیسترهای n بیتی گشته و مشخصات دو نمونه از آنها را پیدا کنید

۴ -تمرین قبل را برای شیفت رجیستر نیز انجام دهید.

۵ -تمرین قبل را برای دو نمونه شمارنده نیز انجام دهید.

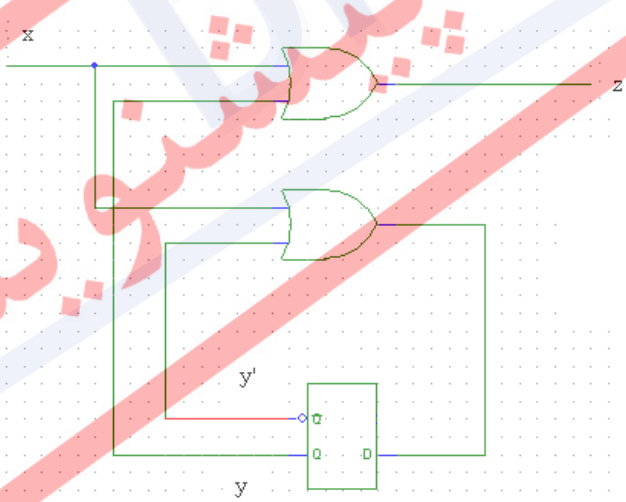
۶–برای مدار ترتیبی شکل زیر موارد زیر را بیابید:

۱) جدول حالت (A=0,B=1)

۲) دیاگرام حالت

۳) یک دیاگرام زمانی در صورتی که حالت شروع برابر $y^0 = 0$ و x = 001011000 باشد.

این مدار حساس به لبه است.



۷–مدار منطقی مطابق با معادلات منطقی زیر را با استفاده از فلیپ فلاپ های D حساس به لبه بکشید:

$$Y_1 = \overline{x} \oplus y_1 = x \otimes y_1$$
$$Y_2 = x + y_1 + y_2$$
$$z = xy_1\overline{y_2}$$

نسخه پیش‌نویس Draft

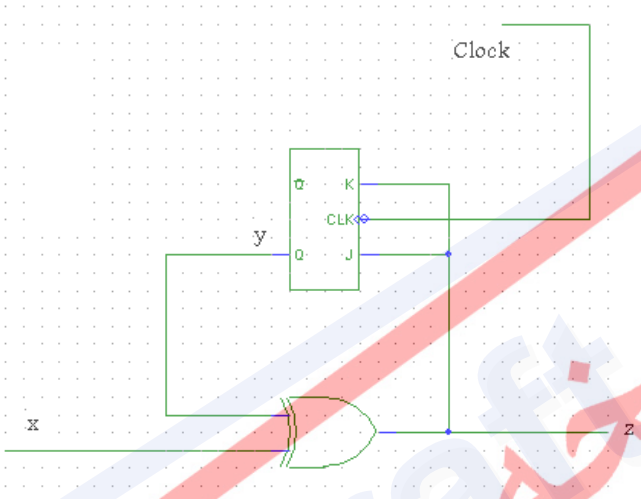
۸-اگر مدار ترتیبی شکل زیر زنجیره خروجی زیر را بدهد:

z = 1101111

در حالیکه ورودی به صورت زیر است:

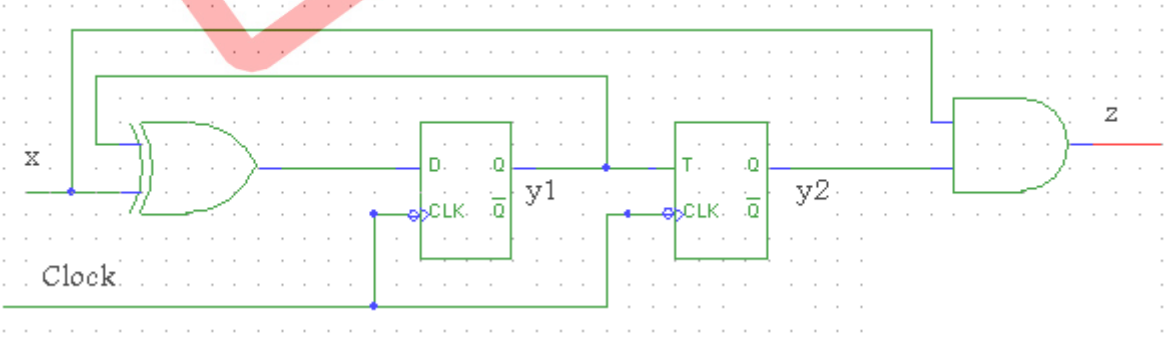
x = 01101010

حالت شروع چیست؟



۹-دیاگرام حالت مربوط به مدار ترتیبی شکل زیر ۸.۱۰ را با توجه به جدول تخصیص حالت زیر بیابید.

y_1	y_2
0	0
0	1
1	1
1	0



نسخه پیش‌نویس
Draft

۱۰-یک مدار تولید کد پریتی سریال طراحی کنید.این مدار ترتیبی از بیتها را می گیرد و مشخص می سازد که این رشته از بیتها دارای تعدادی زوج یا فرد از ۱ است.خروجی این مدار که با P نشان می دهیم، در حالتی که رشته دارای تعداد زوج از ۱ باشد ۰ و در صورتی که دارای تعدادی فرد از ۱ باشد ۱ خواهد بود.

نسخه پیشنویس
Draft

فصل ۴

ساده سازی مدارهای ترتیبی

نسخه پیش‌نویس
Draft

۴-۱ مقدمه

همان طور که می‌دانید، تعداد حالت‌ها در مدارهای ترتیبی تأثیر مستقیم در هزینه و سرعت عملکرد مدار دارد. در نتیجه، یکی از راه های طراحی مدارات سریعتر و ارزان تر آن است که از حداقل حالات ممکن جهت پیاده سازی مدار استفاده شود. حذف حالت های زائد به چند دلیل حائز اهمیت است:

۱. **هزینه:** تعداد عناصر حافظه با تعداد حالت‌ها نسبت مستقیم دارد.

۲. **پیچیدگی:** هر چه تعداد حالت‌های مدار بیشتر باشد، طراحی و ساخت آن پیچیده‌تر است.

۳. **اشتباه تحلیل کامپیوتری:** برنامه‌های تشخیص عیب غالباً با این فرض عمل می‌کنند که حالت زائد وجود ندارد.

به عنوان مثال در شکل زیر، یک جدول حالت و ساده سازی شدهٔ آن، نشان داده شده است. برای رسیدن به این هدف، روش‌هایی برای حداقل کردن تعداد حالات یک مدار ترتیبی وجود دارد که در ادامه شرح داده می‌شود.

جدول ۴-۱ یک جدول حالت و ساده‌سازی شده آن

حالت اولیه	ورودی X: 0	ورودی X: 1
q_1	q_2, 0	q_4, 0
q_2	q_3, 1	q_5, 1
q_3	q_1, 0	q_2, 0
q_4	q_1, 0	q_4, 0
q_5	q_1, 0	q_4, 0

تعداد فیلیپ-فلاپ

N = 3

بعد از ساده سازی

حالت اولیه	ورودی X: 0	ورودی X: 1
q_1	q_2, 0	Q_4, 0
q_2	q_3, 1	Q_5, 1
q_3	q_1, 0	Q_2, 0
q_4	q_1, 0	Q_4, 0

تعداد فیلیپ-فلاپ

N = 2

همان طور که ملاحظه می‌کنید، q5 و q4 حالت‌های معادل هستند.

۴-۱-۱ تعاریف ساده سازی

تعریف ۱) حالت q' از جدول کارنو M' (جدول مینیمم شده)، حالت q از جدول M (جدول حالت اولیه) را می‌پوشاند، اگر به ازای هر ورودی، خروجی‌های یکسانی داشته باشند.

نسخه پیش‌نویس Draft

تعریف ۲) جدول M' جدول M را می‌پوشاند، اگر به ازای هر حالتی از M، حداقل یک حالت در M' وجود داشته باشد که آن را بپوشاند.

تعریف ۳) اگر جدول M بوسیلهٔ جدول M' پوشانده شود و جدول M' تعداد حالات کمتری نسبت به M داشته باشد، ساده‌سازی انجام شده است. در این مرحله، هر حالت از M بوسیلهٔ یک حالت از M' پوشانده می‌شود. این حالت‌ها را حالت معادل[1] می‌گویند.

۴-۲ روش ساده سازی با استفاده از جدول چارت دوتایی[2]

می‌خواهیم در جدول زیر حالات معادل را بیابیم. در روش چارت دوتایی تمام حالات را دو به دو با یکدیگر مقایسه می‌کنیم.

جدول ۴-۲ جدول نمونه برای ساده‌سازی با استفاده از چارت دوتایی

حالت اولیه \ ورودی X	0	1
q_1	q_3, 1	q_4, 1
q_2	q_3, 0	q_2, 0
q_3	q_4, 0	q_1, 0
q_4	q_1, 0	q_3, 0

چارت دوتایی جدول بالا را در زیر ملاحظه می کنید.

q_2	××		
q_3	××	q_3,q_4 / q_2,q_1	
q_4	××	q_3,q_1 / q_2,q_3	q_4,q_1 / q_1,q_3
	q_1	q_2	q_3

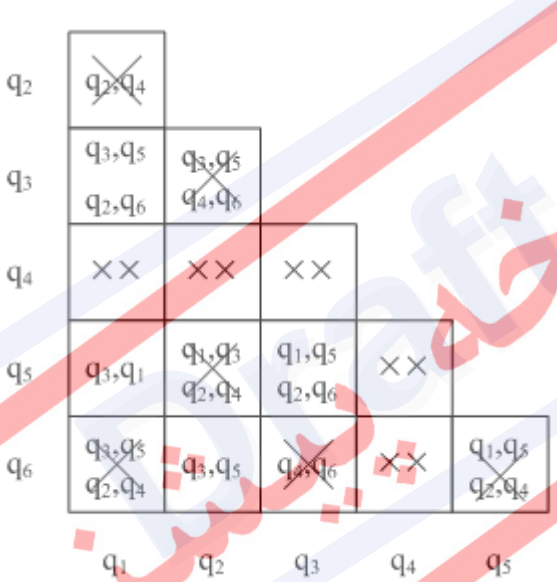
[1] Equivalent State
[2] Pair Chart

نسخه پیش‌نویس

از آنجایی که هیچ دو حالتی با هم معادل نیستند، این جدول ساده نمی‌شود. اما در مثال زیر داریم:

جدول ۳-۴ – جدول حالت نمونه برای ساده‌سازی، دارای حالات معادل

	0	1
q1	q3, 0	q2, 0
q2	q3, 0	q4, 0
q3	q5, 0	q6, 0
q4	q1, 1	q5, 1
q5	q1, 0	q2, 0
q6	q5, 0	q4, 0



لم ۱: قانون تعدی (شرکت پذیری برای معادل بودن)

If $(q_i \sim q_j)$ and $(q_j \sim q_k) \rightarrow q_i \sim q_k$

لم ۲: اگر $q_1 \sim q_k$، $q_2 \sim q_k$، ... و $q_j \sim q_k$ باشد، آنگاه همگی حالت‌ها با یکدیگر معادل هستند.

نکته) این لم‌ها فقط برای جداول حالت کاملاً معلوم (بدون حالت بی تفاوت) صادق است.

ادامهٔ مثـال: حالت‌های معادل بصورت زیر خواهند بود،

$$\left\{\begin{array}{ll} q1 \sim q3 & \checkmark \\ q2 \sim q6 & \\ q1 \sim q5 & \checkmark \\ q3 \sim q5 & \checkmark \end{array}\right. \quad \text{در نتیجه} \Leftarrow \quad \left\{\begin{array}{l} q1 \sim q3 \sim q5 \\ q2 \sim q6 \end{array}\right.$$

نسخه پیش‌نویس

- حالت‌های مینیمم شده: (حالات جدول M')

B1 = q1, q3, q5

B2 = q2, q6

B3 = q4

- جدول M':

جدول۴-۴ – جدول حالت ساده‌سازی شده مربوط به جدول حالت ۳

حالت اولیه	ورودی X: 0	ورودی X: 1
B_1	B_1, 0	B_2, 0
B_2	B_1, 0	B_3, 0
B_3	B_1, 1	B_1, 1

۴-۳ ساده سازی برای مدارهای دارای حالت بی اهمیت

استفاده از چارت دوتایی و جدول دلالت[1] برای مدارات کاملاً معلوم[2] قابل استفاده است. یعنی مداراتی که حالت بی‌اهمیت در آنها وجود ندارد. می‌خواهیم نشان بدهیم که وجود حالت های بی تفاوت چگونه مشکل ساز می‌شود. به مثال زیر توجه نمایید:

	0	1
1	3, 0	2, 0
2	2, -	1, 0
3	1, 1	2, 0

جدول ۴-۵ – جدول حالت نمونه دارای حالات بی‌اهمیت

[1] Implication Table

[2] Completely Specified Circuit

نسخه پیش‌نویس Draft

2	2,3 1,2	
3	×	1,2 1,2
	1	2

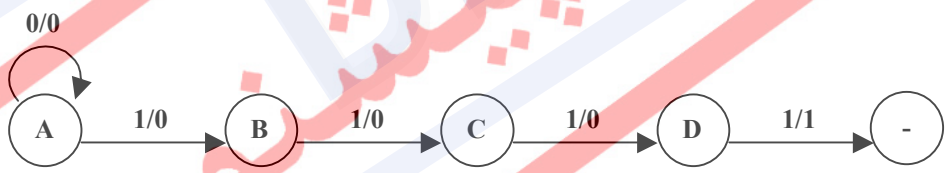
بنابراین خواهیم داشت:

$$A = 1 \sim 2$$
$$B = 2 \sim 3$$

همانطور که ملاحظه می‌نمایید، در جدول حالت نهایی برای ستون ورودی یک، ابهام وجود دارد. یعنی از یک سو با توجه به خانهٔ ۱-۳ در جدول چارت دوتایی، حالات ۱ و ۳ نمی‌توانند با یکدیگر معادل باشند و از سوی دیگر تساوی حالات ۱ و ۳ از جدول نتیجه شده است. دلیل این تناقض، وجود حالت بی‌اهمیت در جدول اولیه است. در ادامه، به بررسی حل این مشکل می‌پردازیم.

۴-۳-۱ تعاریف ساده‌سازی

۱- یک دنبالهٔ ورودی را قابل اعمال[1] به یک حالت گویند، اگر وقتی آن ورودی اعمال می‌گردد، تمام حالت‌های بعدی مشخص شوند. (تنها استثنا در مورد حالت نهائی می‌باشد.)



دنباله قابل اعمال برای حالت A: 0111 و 1111

دنباله غیرقابل اعمال برای حالت A: 11111

۲- دو حالت Si و Sj را سازگار[2] گویند اگر به ازای هر دنبالهٔ ورودی قابل اعمال به آنها، به یک دنبالهٔ خروجی یکسان برسند.

۳- مجموعهٔ حالت‌های سازگار را کلاس سازگاری[3] گویند.

[1] Applicable
[2] Compatible
[3] Compatibility Class

نسخه پیش‌نویس Draft

۴- حداکثر سازگاری[1] (MCs)، یک کلاس سازگاری است که همهٔ کلاس‌های سازگاری را پوشش می‌دهد.

مثـال: داریم: $MCs=\{ACE\} \leftarrow \{AC, AE, CE, ACE\}$

برای ساده سازی مدارهای دارای حالت بی‌اهمیت، در ابتدا باید حداکثر سازگاری را پیدا نمود. در ادامه، روش‌های موجود برای پیدا کردن مجموعهٔ MCs شرح داده می‌شود.

۴-۳-۲ الگوریتم کامپیوتری برای یافتن مجموعهٔ حداکثر سازگاری

۱. رسم جدول چارت دوتایی

۲. در این جدول، ابتدا با سمت راست‌ترین ستون آغاز می‌کنیم. اگر این ستون شامل یک زوج سازگار است، مجموعهٔ MCs را به صورت $L=\{c_n c_{n-1}\}$ انتخاب می‌کنیم که c_i نشان دهندهٔ نام حالات سازگار است. اگر این حالات سازگار نیستند، مجموعهٔ MCs را به صورت $L=\{c_n,c_{n-1}\}$ انتخاب می‌کنیم.

۳. به ستون بعدی می‌رویم که حالت q_k می‌باشد. S_k را مجموعهٔ تمام حالت‌های سازگار با q_k را در نظر می‌گیریم. اگر در این ستون، حالت سازگاری با q_k وجود ندارد، مجموعهٔ S_k را برابر با تهی در نظر می‌گیریم.

$$L' = \{c'_i \mid c'_i = (S_k \cap c_i) \cup \{q_k\} \text{ for all } c_i \in L\}$$

۴. مجموعهٔ L دوباره محاسبه می‌شود: $L=L\cup L'$. در این مرحله، حالاتی که زیر مجموعهٔ حالتی دیگر باشند، حذف می‌شوند.

۵. قدم ۳ و ۴ این روال را برای همه ستون‌ها تکرار می‌کنیم. در پایان، L مجموعهٔ حداکثر سازگاری است.

برای روشن شدن عملکرد این الگوریتم، به مثال زیر توجه کنید.

[1] Maximum Compatibilities

نسخه پیش‌نویس Draft

مثال ۱) مرحلهٔ یک:

2	2,3 1,2	
3	×	1,2
	1	2

مرحلهٔ دو: L={23}

مرحلهٔ سه:

$$q_k = 1$$
$$s_k = \{2\}$$
$$L' = \{(2 \cap 2) \cup 1), (2 \cap 3) \cup 1)\} = \{12\}$$
$$L = L \cup L' = \{12,2,3\} = \{12,3\}$$

مرحلهٔ چهار:

$$A = \{12\} \qquad B = \{3\}$$

مثـال ۲)

جدول ۴-۶ – جدول حالت ساده‌سازی نشده مربوط به مثال ۲

حالت اولیه	I_1	I_2	I_3
1	3, 0	-	2, -
2	-	4, 0	6, -
3	5, 1	-	-, 0
4	-	1, 1	-
5	1, -	-	6, -
6	4, -	5, -	6, -

مرحلهٔ یک:

نسخه پیش‌نویس Draft

2	2,6✓				
3	××	✓			
4	✓	××	✓		
5	1,3×	✓	×	✓	
6	3,4 2,6✓	4,5✓	4,5✓	1,5×	1,4✓
	1	2	3	4	5

مرحلهٔ دو:

$L = \{56\}$

$q_k = 3$

$s_3 = \{46\}$

$L' = \{ (46 \cap 56) \cup 3,$

$(46 \cap 54) \cup 3\} = \{36, 43\}$

مرحلهٔ سه:

$q_k = 4$

$S_k = \{s\}$ $\quad L' = \{(5 \cap 56) \cup 4\} = 54$ $\quad L = L \cup L' = \{56, 54, 36, 34\}$

مرحلهٔ چهار:

$L = L \cup L' = \{56, 54\}$

$q_k = 2$

$S_2 = \{356\}$

$L' = \{(356 \cap 56) \cup 2, (356 \cap 45) \cup 2, (356 \cap 36) \cup 2, (356 \cap 34) \cup 2\}$

$L' = \{256, 52, 362, 32\}$

$L = L \cup L' = \{256, 63, 45, 34, 56, 36\}$

$q_k = 1$

$S_1 = \{246\}$

$L' = \{(246 \cap 256) \cup 1, (246 \cap 236) \cup 1, (246 \cap 45) \cup 1, (246 \cap 84) \cup 1\}$

$L' = \{126, 14\}$

$\Rightarrow L = L' \cup L = \{126, 14, 256, 236, 45, 34\} = MCs$

نسخهٔ پیش‌نویس Draft

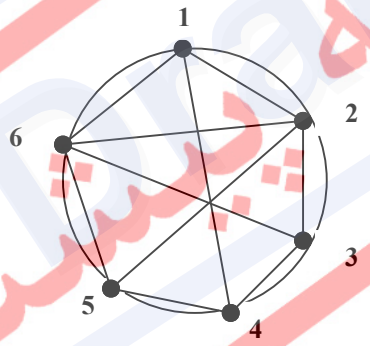
۴-۳-۳ روش دیاگرام ادغام برای یافتن مجموعهٔ حداکثر سازگاری

در این روش از یک گراف به نام دیاگرام ادغام استفاده می‌کنیم. این روش برای مدارهای کوچک مناسب می‌باشد. در این گراف:

- هر حالت مدار، یک گره از گراف روی محیط یک دایره است.
- یال‌ها، نمایانگر سازگاری بین دو حالت است.

برای پیدا کردن مجموعهٔ حداکثر سازگاری، محیط‌های بسته و کامل با اندازهٔ حداکثر را در این گراف پیدا می‌کنیم. در واقع باید چند ضلعی‌هایی که کاملاً متصل[1] هستند را از این گراف استخراج نماییم. توجه کنید که این چند ضلعی‌ها حداقل در یکی از اضلاع نباید با یکدیگر همپوشانی داشته باشند. همچنین همه یال‌های حالت‌ها باید پوشش داده شود.

برای نشان دادن نحوهٔ عملکرد این روش، مثال آخر بخش قبل را با این روش انجام می‌دهیم. در زیر دیاگرام ادغام برای این مثال را مشاهده می‌کنید.



$$MC'S = \{263, 265, 126, 34, 45, 14\}$$

به کمک این گراف می‌توان حداکثر ناسازگاری[2] را نیز بدست آورد. برای این منظور، می‌بایست گراف دوگان را از دیاگرام ادغام به دست آورد و روال قبل را روی گراف حاصل، تکرار کرد.

[1] Strongly connected
[2] Maximum Incompatibility

نسخه پیش‌نویس Draft

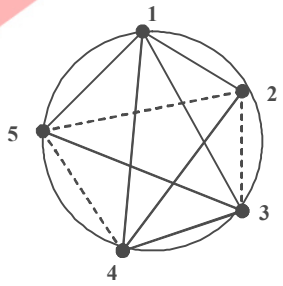
مثـال ۳)

حالت اولیه	ورودی X: 0	ورودی X: 1
1	1, -	-
2	3, 1	2, 0
3	4, 0	-, 1
4	-	2, -
5	1, 0	3, 1

جدول ۷ – جدول حالت ساده‌سازی نشده مربوط به مثال ۳

	1	2	3	4
2	13✓			
3	14✓	××		
4	✓	✓	✓	
5	✓	××	14✓	23×

گراف سازگاری با خطهای پررنگ و گراف ناسازگاری با خطوط خط چین مشخص شده است.



$$MC'S = \{124, 135, 134\}$$

$$M\ \ Incompatibility = \{52,\ 54,\ 23,1\}$$

نسخه پیش‌نویس
Draft

۴-۳-۴ یافتن مجموعهٔ حداقل حداکثر سازگاری

در ادامهٔ کار ساده‌سازی، باید مجموعهٔ حداکثر سازگاری را حداقل کنیم. مجموعهٔ حداقل باید سه شرط داشته باشد:

- **کامل بودن**[1]: اجتماع تمام مجموعه‌های انتخاب شده باید تمام حالت‌های ماشین اولیه را شامل شود. (شامل همه حالت‌ها باشد.)
- **یکپارچگی**[2]: مجموعه انتخاب شده باید بسته باشد. به این معنی که حالت‌های بعدی از هر کلاس سازگاری از مجموعهٔ انتخاب شده، در یک کلاس سازگاری یکسان وجود داشته باشد. یعنی اگر $i \to p$ و $j \to q$ و i و j در یک کلاس سازگاری باشند، p و q هم باید در یک کلاس سازگاری باشند.
- **حداقل بودن**[3]: باید حداقل تعداد کلاس‌های سازگاری که شرایط بالا را داشته باشند، انتخاب کرد.

برای روشن شدن تعریف یکپارچگی به مثال زیر توجه نمایید.

جدول 4-۸ – یک جدول حالت نمونه و حالت ساده‌سازی شده آن که خاصیت یک پارچگی را دارا می‌باشد

حالت اولیه	ورودی X: 0	ورودی X: 1
1	1, -	-
2	3, 1	2, 0
3	4, 0	-, 1
4	-	2, -
5	1, 0	3, 1

حالت اولیه	X: 0	X: 1
1	1	-
3	4	-
5	1	3

$MC'S = \{124, 135, 134\}$

همانطور که ملاحظه می‌کنید برای کلاس سازگاری 135، تمام حالت‌های بعدی در کلاس 134 است و در این شرایط یکپارچگی وجود دارد. اگر حالت‌های بعدی در یک کلاس سازگاری نمی‌بود، آن‌گاه

[1] Completeness
[2] Consistency - [3] Close
[3] Minimality

نسخه پیش‌نویس Draft

مجموعهٔ انتخاب شده، بسته نیست. شرط یکپارچگی باید برای همه کلاس‌های سازگاری برقرار باشد. برای این کار از جدول بستار[1] استفاده می‌شود که برای مثال بالا بصورت زیر خواهد بود:

	0	1
124	13	2
135	14	3
134	14	2

جدول 4-۹ – جدول بستار برای جدول حالت مربوط به جدول 4-۸

- قضیه: مجموعهٔ حداکثر سازگاری، همواره کامل و یکپارچه می‌باشد.
- قضیه: مجموعهٔ حداکثر سازگاری برای مدارات کاملاً معلوم، حداقل می‌باشد. اما برای مدارات نامعلوم اینگونه نیست.

بنابراین برای پیدا کردن مجوعهٔ حداقل، باید روشی استفاده شود. در اکثر مراجع و کتابهای این زمینه از روش‌های سعی و خطا برای پیدا کردن حداقل حداکثر سازگاری‌ها استفاده می‌شود. در این روش‌ها یکی از قدمهای مهم، پیدا کردن محدودهٔ جواب خواهد بود.

۴-۳-۵ پیدا کردن محدودهٔ حالات نهایی

برای پیدا کردن محدوهٔ حالات نهایی، فرض می‌کنیم که تعداد آنها K باشد و بین دو عدد محصور باشد: $L \le K \le U$ که L محدوه پایین و U محدوده بالا است. پیدا کردن U بسادگی امکان پذیر خواهد بود و برابر تعداد اعضای مجموعهٔ حداکثر سازگاری است و مشخص است که حالات نهایی از این مقدار بزرگتر نخواهد بود. برای L باید توجه داشت که ناسازگاری‌های موجود در حالت‌ها تعیین کننده است. در واقع، اندازهٔ بزرگترین ناسازگاری، مقدار L را مشخص می‌کند.

برای نمونه، در مثال قبل خواهیم داشت:

$$M = Min\{3,5\} = 3 \Rightarrow M = 3$$
$$MC'S = \{124, 135, 134\}$$
$$M\ Incompatibility = \{23, 25, 54\}$$
$$L = Max\{2, 2, 2\}$$
$$\Rightarrow \quad 2 \le k \le 3$$

یعنی برای حالت MC's ، تعداد اعضا نشان دهندهٔ U و برای MInC's ، تعداد اجزای بزرگترین عضو، نشان دهندهٔ L است.

[1] Clouser Table

نسخهٔ پیش‌نویس Draft

۴-۳-۶ الگوریتم ساده‌سازی برای مدارات ترتیبی نامعلوم

برای ازبین نرفتن انسجام مطلب، در این قسمت قدم‌های مورد نیاز برای ساده سازی را اشاره می‌نماییم:

۱. رسم جدول چارت دوتایی

۲. پیدا کردن MC's و MInC's

۳. پیدا کردن حد پایین و بالا برای حالت‌های نهائی $L \le K \le U$

۴. پیدا کردن مجموعه‌ای از کلاس‌های سازگاری که سه شرط کامل بودن، یکپارچگی و حداقل بودن را داشته باشد.

۵. تولید جدول حالت ساده شده.

تا کنون، سه قدم اول شرح داده شده است. برای انجام قدم چهارم اقدامات زیر را انجام می‌دهیم:

۱. مجموعهٔ کلاس‌های[1] p_i را بصورت زیر پیدا می‌کنیم:

اگر c_i یک کلاس سازگاری باشد و c_{ij} مجموعهٔ حالت‌های بعدی که از c_i با ورودی I_j بدست می‌آید، p_i زیر مجموعهٔ c_{ij} است با این شرایط:

- c_{ij} بیشتر از یک حالت داشته باشد یا تهی باشد.
- $c_{ij} \not\subset c_i$ زیرمجموعه خودش نباشد. (یک کلاس سازگاری به خودش برنگردد و خودش را پوشش ندهد.)
- $c_{ij} \not\subset c_i$ if $c_{ik} \in p_i$ زیرمجموعه کلاس دیگری در p_i نباشد.

برای قدم بعدی باید چند تعریف انجام دهیم:

تعریف ۱) یک کلاس سازگاری مثل c_i می‌تواند به کلاس سازگاری c_j غلبه کند، اگر دو شرط زیر برقرار باشد: $c_j \subset c_i$ و $p_i \subseteq p_j$ ، مانند:

C_i	P_i
126	34
26	34

126 بر 26 غلبه می‌کند.

[1] Class Set

نسخه پیش‌نویس Draft

تعریف ۲) یک کلاس سازگاری را که هیچ کلاسی بر آن غلبه نکند، سازگاری اصلی[1] (PC) گویند.

۲. پیدا کردن سازگاری‌های اصلی با استفاده از تعاریف ۱ و ۲.

۳. پیدا کردن گراف التزام[2] به صورتی که گره‌ها سازگاری‌های اصلی و یال‌ها نشانگر ارتباط با c_i ها باشد.

۴. مجموعهٔ عناصرکوچکترین (با کمترین تعداد گره) دور در این گراف که شامل تمام حالت‌ها باشد، جواب نهایی است.

نسخه پیش‌نویس Draft

مثال ۴)

	I_1	I_2	I_3
1	3, 0	-	2, -
2	-	4, 0	6, -
3	5, 1	-	-, 0
4	-	1, 1	1, -
5	1, -	-	6, -
6	4, 1	5, -	6, -

جدول 4-۱۰ – جدول حالت ساده‌سازی نشده مربوط به مثال ۴

حل:

$$MC's = \{126, 14, 236, 34, 45, 256\}$$
$$M\ Incompatibility = \{135, 24, 64\}$$
$$L = Max\{3, 2, 2\} = 3 \quad \Rightarrow \quad 3 \le k \le 6$$

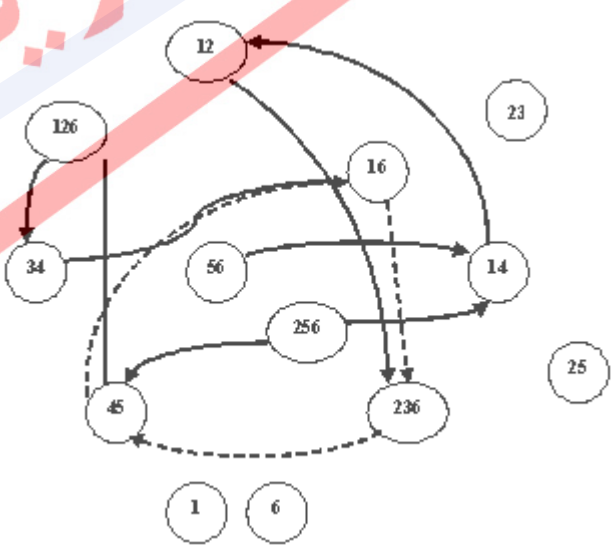
در اینجا باید سازگاری‌های اصلی را پیدا کنیم. گراف سازگاری در شکل ۱ نشان داده شده است.

---

[1] Prime Compatibility
[2] Implication Graph

جدول 4-۱۱ – جدول سازگاری مربوط به مثال ۴

	C_i	P_i
	126	34,45,26
	12	26
غلبه با 236	26	45
	16	34,26
	14	Φ
	236	45
تکراری	26	
	36	45
	23	Φ
	34	Φ
	45	16
	256	14,45
	25	Φ
تکراری	26	
	56	14
	1	Φ
	2	Φ
	3	Φ
	4	Φ
	5	Φ
	6	Φ



شکل 4-۱- گراف سازگاری مثال ۴

نسخه پیش‌نویس
Draft

چند نکته در رسم این گراف قابل به ذکر است:

* گره با بیشترین ورودی همیشه انتخاب اول است.

* چون 26 با 236 غلبه شده، گره 12 به جای 26 به 236 وصل می‌شود.

* در پیدا کردن دور در گراف در مرحلهٔ ۵ الگوریتم فوق، باید دقت شود که گره 34 هم انتخاب شده است چون 16 به ازای تعدادی از ورودی‌ها به 34 رفته است.

بنابراین مجموعهٔ حداقل که حالت‌های نهایی مدار می‌باشد، شامل دور مشخص شده در گراف می‌باشد. در نتیجه $K=\{16,236,45,34\}$ و در نهایت جدول نهایی بصورت زیر ساده می‌شود:

جدول 4-۱۲ – جدول حالت ساده‌سازی شده مربوط به مثال ۴

	I_1	I_2	I_3
(16) A	D, 0	C, -	B, -
(236) B	C, 1	C, 0	A, 0
(45) C	A, -	A, 1	A, -
(34) D	C, 1	A, 1	A, 0

مثال ۵)

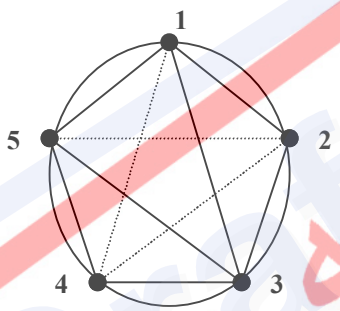
جدول 4-۱۱ – جدول حالت ساده‌سازی نشده مربوط به مثال ۵

	0	1
1	5, 0	1, 0
2	4, 0	2, 0
3	5, -	3, -
4	1, 1	1, 1
5	1, -	2, -

نسخه پیش‌نویس Draft

2	54 12			
3	13	23 45		
4	××	××	15 13	
5	15 12	14×	15 23	15
	1	2	3	4

زوج های سازگار {(1,2), (1, 3), (1, 5), (2, 3), (3, 4), (3, 5), (4, 5)}



MCS = {345, 123, 135}

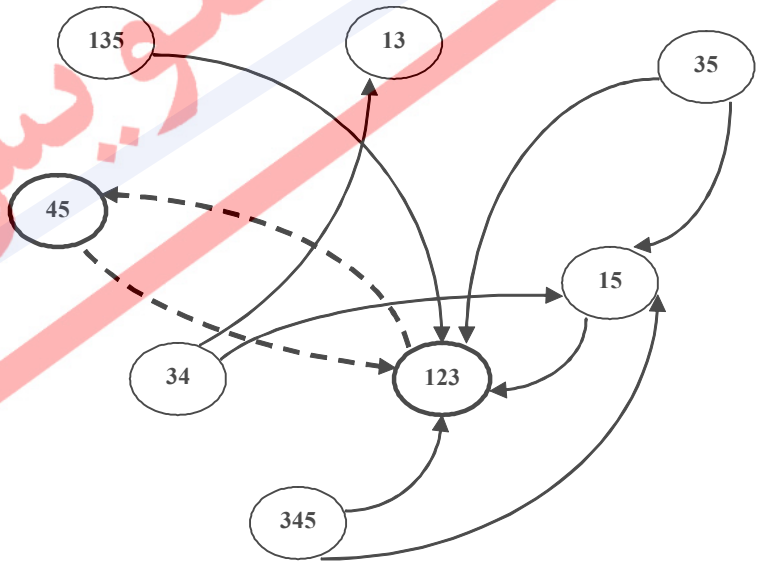
MinCS = {14, 24, 25, 3}

$L = 2 \leq K \leq U = 3$

نسخه پیش‌نویس
Draft

جدول 4-۱۲ – جدول سازگاری مربوط به مثال ۵

	C_i	P_i
	135	~~123,15~~
	13	Φ
	15	12
	35	15,23
	123	45
غلبه	~~12~~	~~45~~
تکراری	~~13~~	
غلبه	~~23~~	
	345	15,23
	34	15,13
	45	12
تکراری	~~35~~	
غلبه	~~1~~	~~Φ~~
	2	Φ
غلبه	~~3~~	~~Φ~~
	4	Φ
	5	Φ



شکل4-۲- گراف سازگاری مثال ۵

نسخه پیش‌نویس
Draft

بر طبق گراف بدست آمده، مجموعهٔ جواب برابر K={123,45} می‌باشد.

جدول4-۱۳ – جدول حالت ساده‌سازی شده مربوط به مثال ۵

		0	1
(123)	B1	B2, 0	B1, 0
(45)	B2	B1, 1	B1, 1

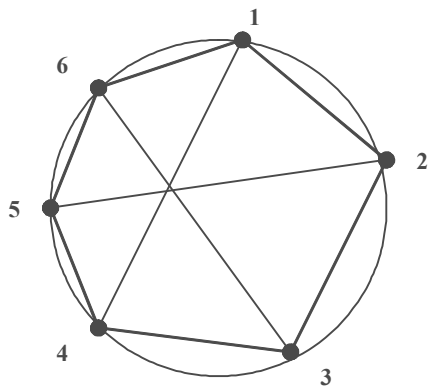
مثـال ۶)

جدول 4-۱۴ – جدول حالت ساده‌سازی نشده مربوط به مثال ۶

	I1	I2	I3
1	3, 0	-	6, -
2	-	4, 0	6, -
3	5, 1	-	-, 0
4	-	1, 1	1, -
5	1, -	-	6, -
6	2, -	5, -	6, -

2	✓				
3	××	✓			
4	16	××	✓		
5	~~13~~	✓	~~15~~	16	
6	23	45	25	~~15~~ 16	12
	1	2	3	4	5

نسخه پیش‌نویس Draft

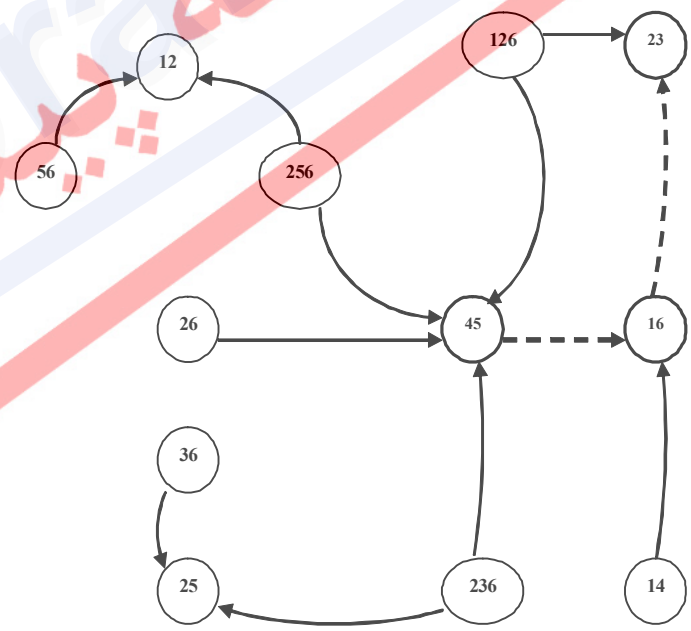


MC’s= {126, 236, 256, 14, 45, 34}

MinC’s = {135, 4, 64}

L=3 ≤ K ≤ U=6

	C_i	P_i
	126	23,45
	12	Φ
	26	45
	16	23
	14	16
	236	25,45
	23	Φ
تکراری	~~26~~	
	36	25
تکراری	~~26~~	
	34	Φ
	45	16
	256	12,45
	25	Φ
	56	12
	~~1~~	~~Φ~~
	~~2~~	~~Φ~~
	~~3~~	~~Φ~~
	~~4~~	~~Φ~~
	~~5~~	~~Φ~~
	~~6~~	~~Φ~~



نسخه پیش‌نویس Draft

شکل 4-۳- گراف سازگاری مثال ۶

جدول 4-۱۵ – جدول حالت ساده‌سازی شده مربوط به مثال ۶

		I1	I2	I3
16	A	B, 0	C, -	A, -
23	B	C, 1	C, 0	A, 0
45	C	A, -	A, 1	A, -

نسخه پیش‌نویس
Draft

۴-۴ تمرین

۱- جدول حالت کاهش یافته برای مدار ترتیبی همزمان نشان داده شده در جدول حالت زیر را محاسبه کنید.

	I	J
A	B/0	C/0
B	D/0	E/0
C	F/0	G/0
D	A/1	B/1
E	C/0	D/0
F	F/0	G/0
G	B/0	F/0

۲- با استفاده از یک جدول دلالت مدار ترتیبی نشان داده شده در جدول زیر را به یک ماشین با کمترین تعداد حالت تبدیل کنید.

	I	J
A	A/0	C/0
B	D/1	A/0
C	F/0	F/0
D	E/1	B/0
E	G/1	G/0
F	C/0	C/0
G	B/1	H/0
H	H/0	C/0

۳- معادله منطقی جهت طراحی و ساخت یک مدار ترتیبی که با استفاده از جدول حالات داده شده، معرفی شده است را بدست آورید ، با استفاده از:

الف-فلیپ فلاپ نوع D

ب- فلیپ فلاپ ساعت دار نوع JK

ج- فلیپ فلاپ ساعت دار نوع SR

د- فلیپ فلاپ ساعت دار نوع T

نسخه پیش نویس Draft

Y1	Y2		X=0	X=1
0	0	A	B/0	C/0
0	1	B	D/0	A/1
1	0	C	A/1	D/0
1	1	D	D/1	B/1

۴- با استفاده از روش تخصص حالتی که در این فصل بررسی کردیم ، اختصاص حالت برای مدار ترتیبی همگام نشان داده شده در جدول را بدست آورید.

	X=0	X=1
A	B/0	E/0
B	D/0	A/1
C	D/1	A/0
D	B/1	C/1
E	A/0	A/0

۵- تخصیص حالت را برای مدار ترتیبی نشان داده شده در جدول زیر ، با استفاده از روش بررسی شده در این فصل ، بدست آورید.

	X=0	X=1
A	B/0	E/0
B	D/0	A/1
C	D/1	A/0
D	B/1	C/1
E	A/0	A/0

نسخه پیش‌نویس Draft

۶- تخصیص حالت را برای مدار نشان داده شده در جدول حالت زیر بررسی کنید.

	X=0	X=1
A	B/0	D/1
B	A/1	C/0
C	D/0	A/0
D	C/1	B/1

۷- برای جدول حالت زیر ، طراحی مدار را با استفاده از هر یک از سه تخصیص حالت واحد و برای ۴ حالت و با استفاده از عناصر حافظه زیر بدست آورید:

الف-با استفاده از فلیپ فلاپ ساعت دار نوع T

ب- با استفاده از فلیپ فلاپ ساعت دار نوع JK

ج- با استفاده از فلیپ فلاپ ساعت دار نوع SR

	X=0	X=1
A	C/0	D/0
B	C/0	A/0
C	B/0	D/0
D	A/1	B/1

نسخه پیش نویس Draft

فصل ۵

مدارهای ترتیبی ناهمگام

نسخه پیشنویس
Draft

۵-۱ مقدمه

همانطور که می‌دانید، در مدارهای همگام عامل همگام کننده، پالس ساعت[1] می‌باشد. در مورد ساعت، به یک ایجادکنندهٔ ساعت احتیاج داریم. همچنین، باید زمان‌بندی ساعت را نیز مشخص کنیم. این زمان‌بندی با توجه به بیشترین زمان مورد نیاز در مدار محاسبه می‌شود که به آن گلوگاه می‌گویند. در مدارهای دارای ریزپردازنده، تقریباً ۳۰ درصد از توان مصرفی مدار، صرف ایجاد ساعت می‌شود. لازم به ذکر است که به علت تأخیر در رسیدن پالس ساعت به قطعات (به خاطر مقاومت سیم‌ها)، ممکن است مدار دچار اختلال شود که به این پدیده کجی ساعت[2] می‌گویند. همچنین، ساعت مدار دارای دورهٔ تناوبی برابر با بزرگترین دورهٔ تناوب قطعات مصرفی مدار است. حال اگر فاصلهٔ دورهٔ تناوب‌ها بسیار زیاد باشد، به آن کارائی در بدترین حالت می‌گویند.

حال اگر عامل هماهنگ‌کننده (پالس ساعت) در مدارهای ترتیبی را حذف کنیم، به مدار ناهمگام می‌رسیم. این مدار، مداری ترتیبی است که خروجی آن فقط وابسته به ورودی و شرایط فعلی مدار است. به بیان دیگر، مدارهای ترتیبی عامل همگام سازی ندارند. اما این به این معنی نیست که این مدارها معادل مدارهای ترکیبی هستند.

تعریف مدارهای ترتیبی ناهمگام[3]: مدارهای ترتیبی ناهمگام مدارهایی هستند که خروجی آنها فقط به ورودی و حالت فعلی وابسته است و عامل همگام سازی برای عناصر حافظه در آنها وجود ندارد.

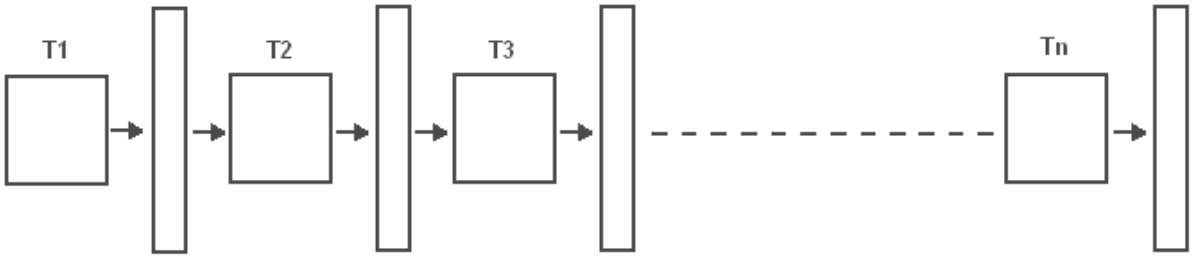
۵-۲ مزایای مدارهای ترتیبی ناهمگام

- **حذف کجی ساعت:** که یک مشکل بزرگ در مدارهای ترتیبی همگام است.
- **کارایی:** به حالت میانگین می‌رسد[4].

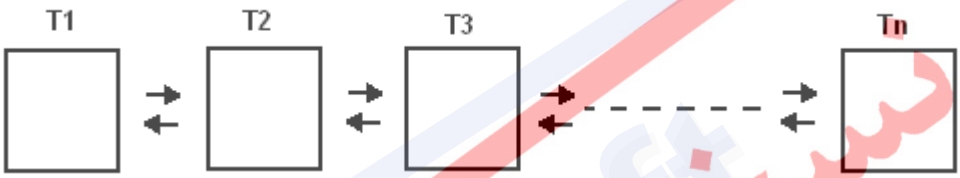
در Pipelining در مدارهای ترتیبی همگام داشتیم:

[1] Clock
[2] Clock Skew
[3] Asynchronous circuits
[4] Average Case Performance

نسخه پیش‌نویس Draft



اگر داده‌ای بخواهد از این n ماجول عبور کند و از آن خارج شود باید به اندازهٔ $T_D = n \times T_{Clock}$ منتظر آماده شدن خروجی بمانیم یعنی به اندازه $n \times \max(Ti)$. در نتیجه مدارهای ترتیبی همگام در بدترین حالت کارایی[1] کار می‌کنند . حال اگر مدارهای ترتیبی ناهمگام را در نظر بگیریم:



زمان آماده شدن خروجی در این مدارات عبارت است از: $T_D = \Sigma Ti$ در نتیجه مدارهای ترتیبی ناهمگام در حالت میانگین کار می‌کنند.

- **زمان‌بندی ساده و راحت‌تر:** برای مدارات همگام، طراحی به صورت جامع (Global) انجام می‌شود، در نتیجه زمان بندی بسیار سخت است. به همین دلیل روی طراحی تک تک ماجول ها زیاد نمی‌توان مانور داد. اما در مدارهای ناهمگام، می‌توان هر ماجول را به تنهایی تغییر داد به نحوی که بتواند با سایر اجزای مدار درست کار کند.
- **بهینه‌سازی:** بهینه‌سازی محلی می‌تواند به صورت بهتری انجام گیرد زیرا وابستگی کمتری بین یک ماجول از مدار با ماجول‌های دیگر وجود دارد. در بهینه‌سازی مدارات همگام، باید بدترین عامل را بهینه کرد تا حاصل $TD = n\times \max(Ti)$ بهبود یابد، اما در طراحی مدارات ناهمگام، هر ماجول که بهینه شود در عملکرد کل مدار تأثیر مثبت دارد.
- **توان مصرفی:** به دلیل حذف ساعت، ارتباط مدار ساده‌تر و توان و کارایی آن بیشتر می‌شود. این در حالی است که در مدارهای همگام حدود ۳۰٪ از توان مصرفی مدار توسط ساعت مصرف می‌شود. در

[1] Worst Case Performance

نسخهٔ پیش‌نویس Draft

مدارهای ناهمگام، برقراری ارتباط بین دو ماجول از طریق سیگنال‌هایی مثل Request و Ack برقرار می‌شود. این گذرها هرچه کمتر باشد، مصرف توان نیز کمتر خواهد شد.

- **پایداری:** مدارهای ناهمگام، در شرایط فیزیکی مختلف، پایداری بیشتری دارند، چون زمانبندی آنها بسیار انعطاف پذیر می‌باشد.

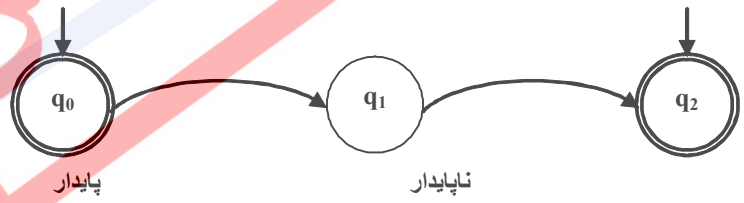
۵-۳ معایب مدارهای ترتیبی ناهمگام

- طراحی پیچیدهٔ مدارهای ناهمگام (به علت حذف سیگنال ساعت، نامنظم شده‌اند)
- ابزارهای طراحی بسیار محدودی دارند. این در حالی است که برای مدارهای همگام، ابزارهای طراحی بسیار قدرتمندی وجود دارد که امکان طراحی‌های پیچیده را فراهم می‌نماید.

۵-۴ مشخصات مدارهای ترتیبی ناهمگام

- ورودی: مانند مدارهای همگام
- حالت :
  - حالت پایدار[1]: حالتهایی هستند که مدار در آن حالت‌ها پایدار می‌ماند.
  - حالت ناپایدار[2]: حالتهایی هستند که مدار در آن حالت‌ها پایدار نیست و از آن حالت فقط گذر می‌کند.

نکته) در مدارهای ترتیبی همگام، تمام حالت‌ها پایدار هستند.



- خروجی

[1] Stable
[2] Unstable

نسخه پیش‌نویس Draft

مدارهای ناهمگام، در یک حالت پایدار (q_0) قرار می‌گیرند، ورودی (I) به آنها اعمال می‌شود، از یک سری حالتهای ناپایدار $\{q_i\}$ $i=1,...,n$ عبور می‌کنند و به یک حالت پایدار q_2 می‌رسند و در این حالت پایدار باقی می‌مانند. همچنین تا زمانی که مدار به حالت پایدار نرسیده، ورودی‌ها باید ثابت بمانند. بنابراین در این مدارات، مدار همیشه از یک حالت پایدار شروع می‌کند و به ازای ورودی مشخص با گذر از یک یا چند حالت ناپایدار، به حالت پایدار بعدی می‌رود. هرچه تعداد حالت‌های ناپایدار بین دو حالت پایدار بیشتر شود، تأخیر مدار بیشتر است. به این ترتیب در این مدارات امکان رخ دادن ناپایداری، بسیار زیاد است.

۵-۵ توصیف مدارهای ترتیبی ناهمگام

برای توصیف مدارهای ناهمگام، از جدول روند[1] (جریان) استفاده می‌کنند که از نظر ساختاری شبیه جدول حالت می‌باشد. تفاوت عمدهٔ این دو جدول در این است که در جدول روند حالت‌های پایدار و ناپایدار از یکدیگر متمایز می‌شوند. (دور حالت‌های پایدار، یک دایره کشیده می‌شود)

جدول ۵-۱ – یک جدول روند نمونه

حالت اولیه \ ورودی XY	00	01	11	10
1	①, 0	3, 1	①, 0	①, 0
2	②, 0	4, 1	-	4, 0
3	③, 1	③, 1	1, 0	4, 0
4	3, 1	④, 0	1, 0	④, 0

۵-۶ روش طراحی هافمن[2] (روش طراحی مد اساسی)

روش هافمن، ساده ترین روش برای طراحی مدارهای ترتیبی ناهمگام است. در این روش طراحی، جدول روند باید سه شرط زیر را داشته باشد و همچنین شرط مد اساسی را نیز رعایت کند.

- در هر ستون از جدول روند باید حداقل یک حالت پایدار موجود باشد، در غیر این صورت مدار نوسان کننده می‌شود. مثلاً اگر در جدول اساسی بالا، حالت پایدار موجود در سطر اول و ستون سوم را برداریم، آنگاه ستون سوم، حالت پایدار نخواهد داشت و در صورت ورود مدار به چنین ستونی، مدار نوسان خواهد کرد و به حالت ثابتی نخواهد رسید.

[1] Flow Table
[2] Hoffman Method (Fundemental Mode)

نسخه پیش‌نویس Draft

- هر انتقال می‌بایست از یک حالت پایدار شروع شده و به یک حالت پایدار نیز خاتمه یابد.
- در حین عملکرد مدار، نباید ورودی‌ای داده شود که مدار را به حالت بی‌اهمیت ببرد.

شرط مد اساسی: در مدارهای طراحی شده که ترتیبی و ناهمگام هستند، ورودی فقط می‌تواند در یک بیت تغییر کند. برای مثال، ورودی نمی‌تواند از ۰۰ مستقیماً به ۱۱ تبدیل شود. بنابراین مدارهایی که یک بیت ورودی دارند، همیشه این شرط را دارند.

$$۰۰ \to ۱۱ \Rightarrow \begin{cases} ۰۰ \to ۰۱ \to ۱۱ \\ \text{یا} \\ ۰۰ \to ۱۰ \to ۱۱ \end{cases}$$

در این روش ما با دو نوع جدول روند برخورد خواهیم داشت:

- **مد اساسی:** با تغییر ورودی، همیشه به یک حالت پایدار می‌رسیم.
- **غیر مد اساسی:** با تغییر ورودی، همیشه به یک حالت پایدار نمی‌رسیم.

همچنین این جداول می‌توانند به یکی از دو صورت زیر باشند:

- **خروجی یک تغییر[1] (SOC):** در هر انتقال، خروجی فقط یک بار تغییر حالت می‌دهد.
- **خروجی چند تغییر[2] (MOC):** در هر انتقال، خروجی بیش از یک بار تغییر حالت می‌دهد.

مثـال ۱)

جدول ۵ -۲- جدول روند مربوط به مثال ۱

حالت اولیه	ورودی X: 0	ورودی X: 1
1	①, 00	2, 00
2	②, 11	3, 01
3	4, 11	③, 01
4	④, 11	3, 01

X : 0 → 1

State: 1 → 2 → 3 → ③

Output: 00 → 00 → 01 → 01

[1] Single Output Change
[2] Multipe Output Change

نسخه پیش‌نویس Draft

همانطور که ملاحظه می‌نمایید، این انتقال از نوع SOC می‌باشد. اگر تمامی انتقال‌ها در یک جدول از نوع SOC باشد، آنگاه جدول SOC است.

جدول 5-3 – جدول روند مربوط به مثال ۱

حالت اولیه	ورودی X: 0	ورودی X: 1
1	①, 00	2, 10
2	②, 11	3, 01
3	4, 10	③, 01
4	④, 10	3, 01

X : 0 → 1

State: 1 → 2 → 3 → ③

Output: 00 → 10 → 01 → 01

به عنوان مثال در جدول بالا، خروجی دو بار تغییر کرده است. بنابراین، این انتقال از نوع MOC است. در شکل۵-۱، آنچه هنگام یک انتقال MOC در سیگنال‌های خروجی رخ می‌دهد، مشاهده می‌شود.

شکل 5-۱ – انتقال MOC

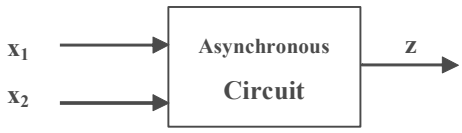
۵-۷ طراحی مدارهای ترتیبی ناهمگام به روش هافمن

مشابه آنچه در مورد مدارهای ترتیبی همگام اشاره شد، ابتدا باید مدار را توصیف کنیم. بنابراین اولین قدم پیدا کردن جدول روند با شرط مد اساسی می‌باشد.

مثـال ۲)

می‌خواهیم یک مدار ترتیبی ناهمگام که دارای دو ورودی و یک خروجی است را طراحی کنیم. به صورتی که اگر $x_1=0$ شود، خروجی نیز برابر صفر شود و هرگاه $x_1=1$ باشد و x_2 تغییر کند، $z=1$ شود و خروجی تا زمانیکه $x_1=0$ شود، در همین حالت باقی بماند(x_1 شبیه Reset عمل می کند).

نسخه پیش‌نویس Draft



مراحل انجام کار:

۱. از یک حالت پایدار اولیه شروع می‌کنیم.

۲. حالات گذر غیر مجاز را حالت بی تفاوت قرار می‌دهیم.

۳. حالت های گذر جدید را پیدا کرده و شماره گذاری می‌کنیم.

۴. برای هر سطر فقط یک حالت پایدار در نظر می‌گیریم (در مراحل بعدی, تعداد حالت‌های ناپایدار در یک سطر، می‌تواند بیشتر شود).

۵. خروجی را فقط برای حالت پایدار در نظر می‌گیریم.

۶. قدم های بالا را تا زمانی ادامه می‌دهیم که حالت جدیدی اضافه نشود.

در زیر، جدول حاصل از عملیات طراحی را ملاحظه می‌نمایید.

جدول 5-4 – جدول روند مربوط به مثال ۲

حالت اولیه	ورودیهای x_1x_2 00	01	11	10
1	①, 0	2	-	3
2	1	②, 0	4	-
3	1	-	5	③, 0
4	-	2	④, 0	6
5	-	2	⑤, 1	6
6	1	-	5 (⑥, 1)	⑥, 1

مثـال ۳) می‌خواهیم مدار ترتیبی ناهمگام که یک شمارنده به پیمانه ۴ است (mod 4) و دارای دو ورودی و دو خروجی است را طراحی کنیم. دیاگرام و نحوهٔ کار مدار بصورت زیر است:

نسخه پیش‌نویس Draft

جدول 5-5 – دیاگرام و نحوه کار مدار مربوط به مثال ۳

X_1	x_2	عملیات مدار
0	0	شمارنده پاک می‌شود. (00)
0	1	به شمارنده یکی اضافه می‌شود. (+1)
1	0	به شمارنده دو تا اضافه می‌شود. (+2)
1	1	مقدار شمارنده تغییر نکند. (x)

مانند مثال قبل، همان قدم‌ها را تکرار می‌کنیم. برای ساده‌تر شدن مطلب، در کنار هر حالت در هر سطر از جدول روند، عمل آن نیز نوشته شده است.

جدول 6-5 – جدول روند مربوط به مثال ۳

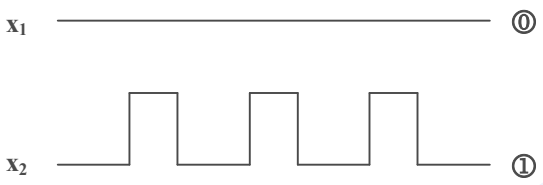
مقدار	عمل	حالت اولیه	ورودیهای x_1x_2 00	01	11	10
00	(clear)	1	①, 00	2	-	3
00	(+1)	2	1	②, 01	4	-
00	(+2)	3	1	-	5	③, 10
01	(x)	4	-	6	④, 01	7
10	(x)	5	-	8	⑤, 10	9
01	(+1)	6	1	⑥, 10	5	-
01	(+2)	7	1	-	10	⑦, 11
10	(+1)	8	1	⑧, 11	10	-
10	(+2)	9	1	-	11	⑨, 00
11	(x)	10	-	12	10, 11	13
00	(x)	11	-	2	11, 00	3
11	(+1)	12	1	12, 00	11	-
11	(+2)	13	1	-	4	13, 01

نسخه پیش‌نویس Draft

۵-۸ تقسیم‌بندی مدارهای ناهمگام از نظر ورودی

دو نوع ورودی برای مدارهای ترتیبی ناهمگام وجود دارد:

- **ورودی پالس**[1]: مقدار صفر، با عدم وجود پالس در ورودی و مقدار یک، با وجود پالس در ورودی مشخص می‌شود.



شکل 5-2 – ورودی پالس

- **ورودی سطح**[2]: دقیقاً مانند ورودی‌های معمولی که در مدارهای همگام نیز استفاده می‌شود.

همانطور که به نظر می‌رسد، بصورت طبیعی از ورودی سطح استفاده می‌شود. اما مدارهای ترتیبی ناهمگام با ورودی پالس، تحت شرایطی می‌توانند مانند مدارهای همگام سنتز شوند که نکتهٔ بسیار قابل توجهی است. شرایط مدارات ناهمگام با ورودی پالس که می‌توانند با روش همگام سنتز شوند، عبارت است از:

۱) پالس‌ها بطور همزمان به دو یا چند ورودی اعمال نشود. (شرط مد اساسی)

۲) انتقال‌های اجزای حافظه، فقط با پالس‌های ورودی شروع می‌شود.

۳) ورودی‌ها یا بصورت معمولی یا بصورت NOT شده استفاده می‌شوند و به هر دو صورت نمی‌توانند وجود داشته باشند.

توجه کنید که نبود پالس ساعت نشان می‌دهد که تغییر حالت دادن نگه‌دار یا فیلیپ-فلاپ باید با پالس های ورودی انجام شود، یعنی تمام اطلاعات زمان بندی سیستم باید از پالس‌های ورودی کسب شود. پس پالس‌های ورودی هم اطلاعات ورودی را به دست می‌دهند و هم وظیفهٔ پالس ساعت در همگام کردن مدار را انجام می‌دهند. روش گام به گام بیان شده در زیر برای سنتز این مدارها، همان روشی است که برای مدارهای همگام بیان شد. ولی جزئیات کار چنانچه در مثال‌های بعدی نشان داده می‌شود، متفاوت است.

---

[1] Pulse Type Input
[2] Level Type Input

نسخه پیش‌نویس Draft

گام ۱: نمودار حالت/جدول حالت را بیابید.

گام ۲: جدول حالت را ساده کنید.

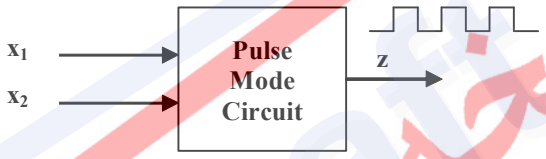
گام ۳: انتساب حالت مناسبی برگزینید و جدول انتقال/خروجی را بسازید.

گام ۴: نوع فیلیپ-فلاپ را تعیین کنید و معادلات تحریک آنها را بیابید.

گام ۵: معادلات خروجی را بیابید.

گام ۶: عناصر منطقی مناسب را برگزینید و نمودار مدار را رسم کنید.

مثـال ۴) می‌خواهیم یک مدار ناهمگام بصورت پالسی طراحی کنیم که دارای دو ورودی x_1 و x_2 و خروجی z است. هرگاه پالس x_1-x_2-x_2 آمد خروجی z=1 است و بعد از آمدن دو ورودی بعدی z=0 خواهد شد.

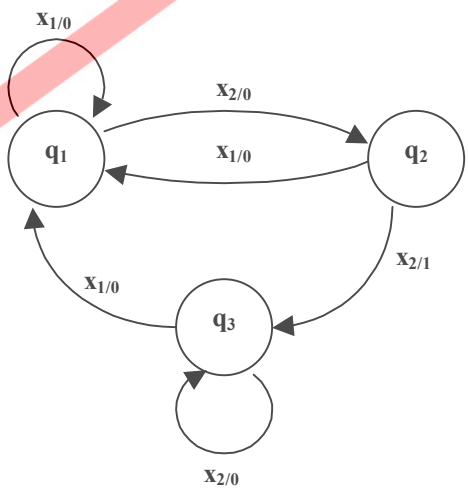


چون خروجی بصورت پالس است (یک لحظه صفر و یک لحظه یک)، پس ماشین میلی است. حالتهای زیر را برای مدار در نظر می‌گیریم و دیاگرام حالت مدار را رسم می‌نماییم.

q_1: آخرین ورودی x_1 بوده است.

q_2: وقتی بعد از x_1 ورودی x_2 بیاید.

q_3: وقتی بعد از x_1 ورودی x_2 بیاید و بعد از آن دوباره x_2 بیاید.



نسخه پیش‌نویس

با توجه به دیاگرام رسم شده، جدول حالت مدار را رسم می‌کنیم. همانطور که دقت می‌کنید به ازاء دو ورودی در مدار، دو ستون در جدول وجود دارد.

جدول 5-۷ - جدول روند مربوط به مثال ۴

حالت اولیه	ورودیهای x_1x_2	
	x_1	x_2
q_1	$q_1, 0$	$q_2, 0$
q_2	$q_1, 0$	$q_3, 1$
q_3	$q_1, 0$	$q_3, 0$

حال می‌توان سنتز این مدار را با روشی که قبلا برای مدارهای همگام گفته شد، انجام داد. مشخص است که برای این کار به دو فیلیپ-فلاپ نیاز داریم. از فیلیپ-فلاپ نوع T استفاده می‌کنیم. جدول بعد از تخصیص حالت، در زیر مشخص شده است.

جدول 5-۸ – جدول روند مربوط به مثال ۴ بعد از تخصیص حالت

y1y2		ورودیهای x_1x_2	
		x1	x2
00	q1	00, 0	01, 0
01	q2	00, 0	11, 1
11	q3	00, 0	11, 0
10		-	-

جداول تحریک فیلیپ-فلاپ‌ها و خروجی در زیر بدست آمده است:

y_1y_2	x_1	x_2
q_1 00	0	0
q_2 01	0	1
q_3 11	1	0
10	-	-

y_1y_2	x_1	x_2
q_1 00	0	1
q_2 01	1	0
q_3 11	1	0
10	-	-

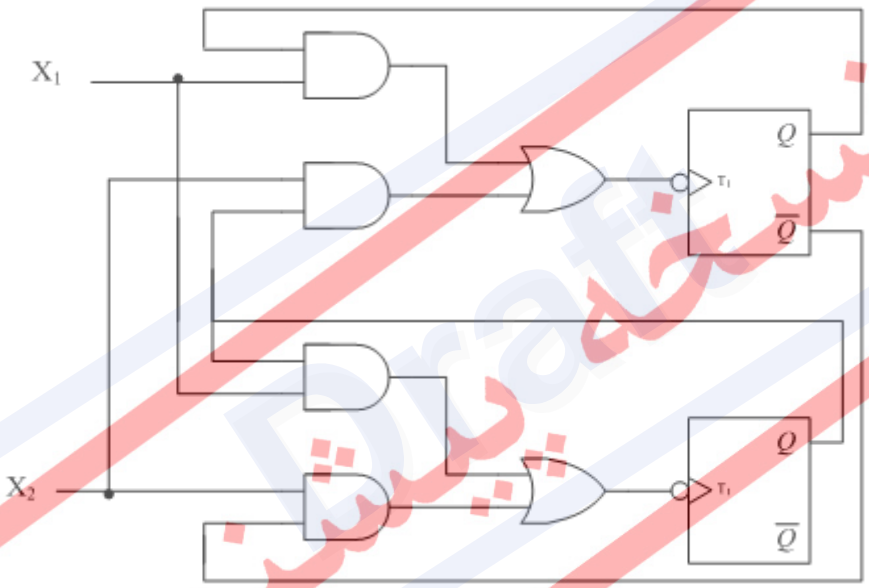
$$T_1 = y_1x_1 + \bar{y}_1y_2x_2 \qquad T_2 = y_2x_1 + \bar{y}_2x_2$$

نسخه پیش‌نویس

y_1y_2	x_1	x_2
q_1 00	0	0
q_2 01	0	1
q_3 11	0	0
10	-	-

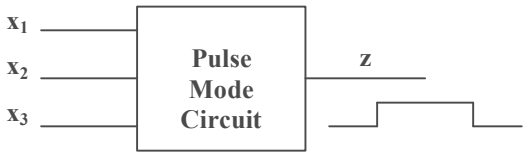
$Z = y_2x_1y_1'$

مشاهده می شود که در معادلات T_1, T_2 و Z در هیچ جا ترکیب x_1 و x_2 با هم وجود ندارد. چون در مدارهای پالس، نباید دو ورودی همزمان فعال باشند. شکل مدار نهایی در شکل ۳ آمده است.



شکل 5-۳ - مدار نهایی مثال ۴

مثال ۵) می‌خواهیم یک مدار آسنکرون طراحی کنیم که سه ورودی دارد و یک خروجی که ورودی‌ها بصورت پالس هستند. اگر ورودی x_1-x_2-x_3 باشد، خروجی از صفر به یک تبدیل می‌شود. می‌خواهیم این خروجی یک باقی بماند تا ورودی x_2 بیاید.



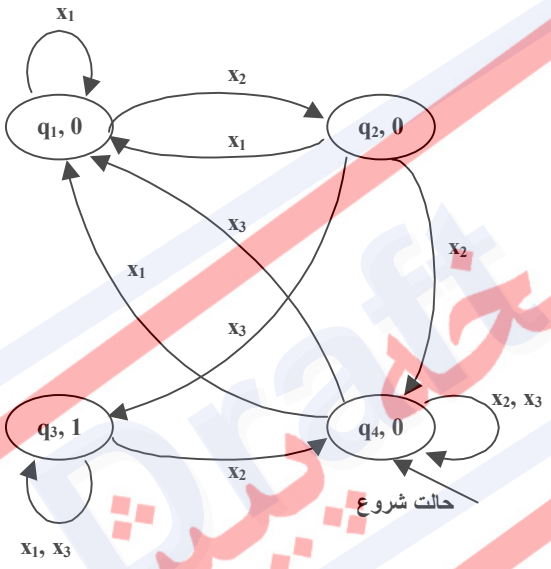
چون خروجی بصورت سطح است، پس ماشین حالت مدار مور است. حالتهای زیر را برای مدار در نظر می‌گیریم و دیاگرام حالت مدار را رسم می‌نماییم.

q_1: آخرین ورودی x_1 بوده است.

q_2: وقتی بعد از x_1 ورودی x_2 بیاید.

q_3: وقتی بعد از x_1 ورودی x_2 بیاید و بعد از آن x_3 بیاید.

q_4: وقتی بعد از x_1 ورودی x_2 بیاید و بعد از آن x_3 بیاید.



جدول 5-۹ – جدول روند مربوط به مثال ۵

حالت اولیه	ورودیها x_1	X_2	x_3
q_1	q_1, 0	q_2, 0	q_4, 0
q_2	q_1, 0	q_4, 0	q_3, 1
q_3	q_3, 1	q_4, 0	q_3, 1
q_4	q_1, 0	q_4, 0	q_4, 0

$q_1 \rightarrow 00$
$q_2 \rightarrow 01$
$q_3 \rightarrow 10$
$q_4 \rightarrow 11$

نسخه پیش‌نویس Draft

جدول‌های حالت بعد و جدول‌های متناظر تحریک فیلیپ-فلاپ SR ، در شکل نشان داده شده است. توجه کنید که برای امکان استفاده از فیلیپ-فلاپ SR ، پالس ورودی باید آنقدر طولانی باشد که تغیر حالت فیلیپ-فلاپ تضمین شود. همچنین، به خاطر داشته باشید که در ساده کردن به کمک جدول کارنو، دسته بندی تنها باید در ستون‌ها انجام شود.

در نهایت، جداول زیر به دست می‌آید.

جدول 5-۱۰ - جدول‌های حالت بعد و جدول‌های متناظر تحریک فیلیپ-فلاپ SR مربوط به مثال 5

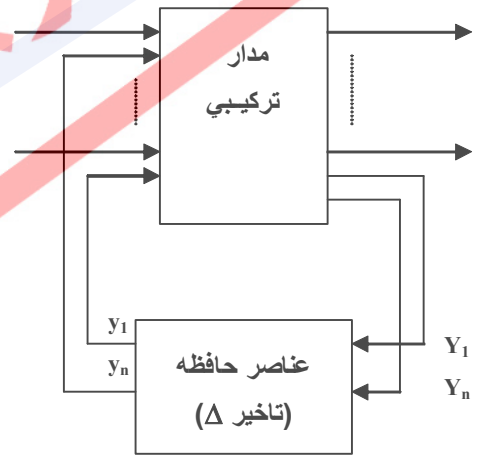
$y_1 y_2$	x_1	x_2	x_3
00	00	01	11
01	00	11	10
11	10	11	10
10	00	11	11

z
000
001
000
101

S_1	R_1	S_2	R_2
001	--0	011	-00
011	000	0-0	101
---	000	--0	011
0--	100	011	-00

۵-۹ تخصیص حالت در مدارهای ترتیبی ناهمگام

همانطور که قبلا اشاره شد، یکی از قدم‌های مهم در سنتز مدارهای ترتیبی، تخصیص حالت می‌باشد. در مدارهای همگام، نشان داده شد که تخصیص حالت فقط در هزینهٔ مدار تأثیر دارد، اما نمی‌تواند باعث مشکلی در کار مدار شود. اما در مورد مدارهای ترتیبی نا همگام اینگونه نیست. در این بخش سعی خواهیم کرد این مشکلات را شناسایی و روش‌های حل آن را بررسی نماییم. برای شروع، ساختار کلی مدارهای ترتیبی نا همگام را در شکل ۴ یادآوری می نماییم.



شکل 5-۴ – ساختار مدارهای ترکیبی

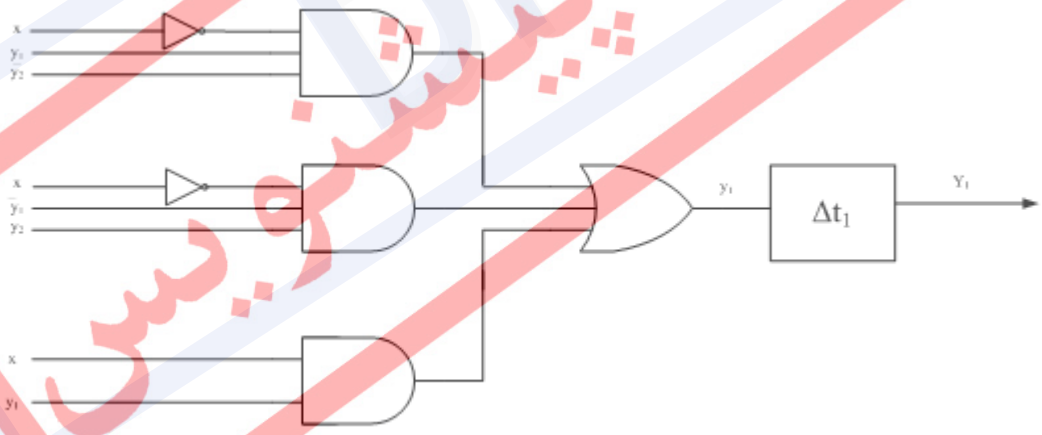
نسخه پیش‌نویس

نکتهٔ مهم در این قسمت، تأخیر موجود در عناصر حافظه است که باعث ایجاد مشکل می‌شود. برای بحث بیشتر در ادامه به یک مثال اشاره می‌شود. فرض کنید مدار متناظر با جدول روند زیر سنتز و آمادهٔ کار شده است.

جدول 5-۱۱ – جدول روند نمونه برای بخش ۵-۸

y_1y_2		ورودی 0	ورودی 1
00	1	①	2
01	2	3	②
10	3	③	4
11	4	1	④

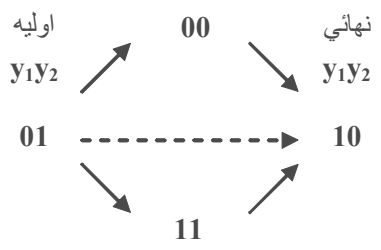
شکل نهایی مدار در شکل ۵ آمده است. ملاحظه می‌کنید که دو عنصر حافظه با تأخیرهای متفاوت در مدار وجود دارد. تخصیص حالت انجام شده در جدول روند مشخص شده است. حال فرض کنید که مدار در حالت 01 و با ورودی 1 قرار دارد (حالت 2 پایدار). اگر ورودی به صفر تغییر کند، طبق جدول مدار باید به حالت 3 ناپایدار رفته و سپس در 3 پایدار شود.



شکل ۵-۵ - مدار نهایی

اما چون حالت 3 دارای کد 10 می‌باشد، تغییر حالت بصورت زیر خواهد بود:

نسخه پیش‌نویس Draft



بنابراین، تغییر حالت از یکی از این دو مسیر صورت می‌گیرد. با فرض اینکه مدار از مسیر اول می‌رود ($\Delta t_2<\Delta t_1$) مدار به حالت 00 می‌رود و چون ورودی صفر است در حالت 1 پایدار می‌شود! اگر از مسیر دو حرکت کند ($\Delta t_2>\Delta t_1$)، به حالت 11 می‌رود و چون ورودی صفر است به حالت 1 ناپایدار می‌رود و سپس در حالت 1 پایدار می‌شود! مشاهده می‌کنید که در این حالت مدار درست عمل نمی‌کند و این مشکل از اینجا ناشی می‌شود که تغییر حالت موجب تغییر در دو بیت کد می‌شود و یک حالت مسابقه[1] بین دو مسیر ایجاد می‌شود که در ادامه در مورد آن بحث می‌نمایم.

۵-۹-۱ مسابقه

هنگامیکه بیشتر از یک بیت ورودی تغییر کند، بسته به سرعت تغییر اجزای مدار و ترتیب تغییر ورودی‌ها، مدار از حالت‌های ناخواسته‌ای عبور می‌کند. به این حالت، مسابقه می‌گویند. به طور کلی دو نوع مسابقه اتفاق می‌افتد:

- **مسابقه غیربحرانی**[2]: از هر دو مسیر به حالت نهایی دلخواه می‌رسیم.
- **مسابقه بحرانی**[3]: حالت نهایی از مسیرهای متفاوت با هم مساوی نیست.

[1] Race
[2] Non-Critical Race
[3] Critical Race

نسخه پیش‌نویس Draft

مثال ۶)

جدول 5-۱۲ – جدول روند پس از تخصیص حالت بخش ۵-۸-۱

	y_1y_2	ورودیهای x_1x_2 00	01	11	10
(a)	00	00	01	00	01
(b)	01	00	01	10	01
(c)	10	00	10	10	11
(d)	11	00	10	00	11

فرض کنید حالت کامل را بصورت $x_1x_2y_1y_2$ تعریف کنیم. در دو شرایط مختلف وضعیت را بررسی می‌کنیم. دقت کنید که تخصیص حالت انجام شده و در جدول روند مشخص شده است.

۱ - مسابقه غیربحرانی (0000 → 1011 :حالت کلی)

جدول ۵-۱۳ – مسابقه غیر بحرانی

زمان	حالت کلی	
t0	1011	
t1	0011	→ y1y2 = 00 Y1Y2 = 11
t2		دوحالت اتفاق می‌افتد →
t3	0000	y1y2 = 00 Y1Y2 = 00

ورودی تغییر می‌کند

$Dt_1 > Dt_2 : t_1 = t_2 + Dt$ ق®0010 0000

$Dt_1 < Dt_2 : t_2 = t_1 + Dt$ ق®0001 0000

همانطور که در این مثال مشاهده می‌کنید از هر دو مسیر مدار به حالت صحیح 00 می‌رود، بنابراین این مسابقه غیربحرانی و بدون مشکل برای مدار است.

نسخه پیش‌نویس Draft

۲- مسابقه بحرانی (1110 → 1001 :حالت کلی)

ورودی از 10 به 11 می‌رود. مشخص است که کد حالت دو بیت تغییر می‌کند.

جدول ۵-۱۴ – مسابقه بحرانی

زمان	حالت کلی	$(\Delta t_2<\Delta t_1)$	زمان	حالت کلی	$(\Delta t_2>\Delta t_1)$
t0	1001		t0	1001	
t1	1101	→ y1y2 = 10 , Y1Y2 = 01	t1	1101	
t2	1100	→ y1y2 = 10 , Y1Y2 = 00	t2	1111	→ y1y2 = 00
t3	1100	مدار در 00 پایدار می‌شود	t3	1110	→ y1y2 = 10
			t4	1110	

در این مثال چون حالت‌های نهایی مساوی نشد، بنابراین مدار عملکرد درستی ندارد و مسابقه بحرانی است.

۵-۹-۲ از بین بردن مسابقه

برای از بین بردن مسابقه باید تخصیص حالت طوری باشدکه حالتهای مجاور[1] دارای کدهای مجاور (تفاوت فقط در یک بیت) باشند. در این صورت هیچگاه مسابقه رخ نمی‌دهد؛ چه بحرانی چه غیربحرانی. در این صورت ما یک تخصیص حالت بدون مسابقه انجام داده‌ایم.

قبل از اینکه به توضیح این روش‌ها بپردازیم، به بحث در مورد حالت‌های مجاور می‌پردازیم. همانطور که از اسم آنها مشخص است، حالت‌هایی هستند که امکان گذر کردن به یکدیگر را با تحریک یک ورودی دارند. برای درک بهتر به مثال زیر توجه کنید.

[1] Adjacent State

نسخه پیش‌نویس

مثـال ۷) این جدول روند را در نظر گرفته و حالت‌های مجاور را پیدا کنید.

جدول۵-۱۵ – جدول روند مربوط به مثال ۱ بخش ۵-۸-۲

حالت اولیه	ورودیهای x_1x_2 00	01	11	10
1	①	2	3	①
2	4	②	②	-
3	1	③	③	-
4	④	5	④	-
5	6	⑤	2	-
6	⑥	3	4	⑥

برای هر حالت در سطر متناظر بررسی می‌کنیم که به کدام حالت‌ها گذر می‌کند. مثلاً برای حالت 1 با ورودی‌های 01 و 11 به حالت 2 و 3 گذر می‌کند، پس آنها حالت‌های مجاور هستند. همچنین در بقیهٔ جدول، اگر حالت مورد نظر از یک حالت دیگر نیز گذر شود، باید آن حالت را نیز به مجموعهٔ حالات مجاور اضافه کنیم. مثلاً در مورد حالت 2 ، از آنجا که حالات 1 و 5 در انتقالات خود از این حالت عبور می‌کنند، لذا 2 و 5 را نیز باید به حالت‌های مجاور حالت 2 اضافه نمود. کل حالت‌های مجاور در جدول زیر مشخص شده است.

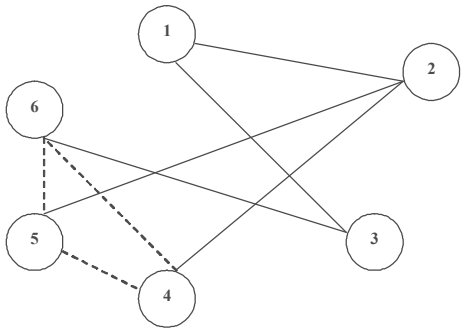
جدول ۵-۱۶ – مشخص کردن حالات مجاور

حالت	حالت‌های مجاور
1	2, 3
2	4, 5, 1
3	1, 6
4	2, 5, 6
5	2, 4, 6
6	3, 4, 5

برای تخصیص حالت با توجه به مجاورت حالت، از یک گراف به نام گراف مجاورت[1] کمک می‌گیریم. گره‌های این گراف حالت‌ها و یال‌های آن مجاورت‌های موجود می‌باشد. برای مثال گراف مجاورت مثال قبل در شکل ۶ مشخص شده است.

[1] Adjacent Graph

نسخه پیش‌نویس Draft



شکل ۵-۶ -گراف مجاورت

حال فرض کنید که فقط می‌خواهید به سه حالت 4، 5 و 6 کد حالت با تفاوت یک بیت انتساب دهید. مشخص است که این کار امکان پذیر نیست. در واقع هر سیکل با طول ۳ در گراف مجاورت، با چنین مشکلی مواجه است. بنابراین باید روشی برای حل این مشکل پیدا کنیم.

دو روش کلی برای تخصیص حالت به این صورت مطرح است:

- **اختصاص چند کد**[1]: به هر حالت بیش از یک کد داده می‌شود.
- **اختصاص یک کد**[2]: به هر حالت فقط یک کد داده می‌شود.

۵-۹-۲-۱ روش مجموعه سطر متصل شده[3]

این روش جزء روشهای چند کد می‌باشد. در این روش با توجه به گراف مجاورت، یک جدول کارنو تخصیص حالت رسم می‌شود. این جدول کاملاً شبیه جدول کارنو است، با این تفاوت که فقط در آن حالت‌ها قرار می‌گیرند. در این جدول می‌توان یک حالت را در چند خانه تکرار کرد. نکتهٔ مهم در این تکرار این است که حتماً باید خانه‌ها با هم مجاور باشند.

برای مثال، جدول کارنوی تخصیص حالت برای مثال قبل می‌تواند بصورت زیر باشد:

y_3 \ y_1y_2	00	01	11	10
0	1	4	6	3
1	2	4	5	5

[1] Multiple State Assignmet
[2] Unicode State Assignmet
[3] Connected Row Set

نسخه پیش‌نویس Draft

همانطور که ملاحظه می‌کنید، حالت‌های 4 و 5 در خانه‌های مجاور تکرار شده‌اند و با این کار تمام مجاورت‌های موجود در گراف مجاورت پوشش داده شده است. بنابراین تخصیص حالت بدون مسابقه بصورت زیر خواهد بود.

جدول ۵-۱۷ – تخصیص حالت بدون مسابقه

$y_1y_2y_3$	حالت
000	1
001	2
100	3
010	4
011	4
111	5
101	5
110	6

دقت کنید که ممکن است چندین جواب مختلف برای این جدول پیدا کنید. برای مثال، جدول زیر نیز یک پاسخ صحیح دیگر است.

y_3 \ y_1y_2	00	01	11	10
0	2	4	6	1
1	2	5	6	3

مثـال۸)

جدول 5-۱۸ - جدول روند مربوط به مثال ۲ بخش ۵-۸-۲

حالت اولیه	ورودیهای x_1x_2 00	01	11	10
1	2	①	①	①
2	②	6	4	②
3	2	③	③	6
4	5	④	④	2
5	⑤	3	1	⑤
6	2	⑥	3	⑥

نسخه پیش‌نویس Draft

مجاورتهای موجود:

حالت	حالتهای مجاور
1	2, 5
2	4, 6, 1, 3
3	2, 5, 6
4	2, 5
5	4, 3, 1
6	2, 3

جدول ۱۹ – حالات مجاور مربوط به مثال ۲

جدول کارنو تخصیص حالت:

y_3 \ y_1y_2	00	01	11	10
0	2	1	5	6
1	2	4	5	3

با توجه به حالت‌های تخصیص داده شده، باید جدول روند را دوباره نویسی کنیم. همانطور که مشاهده می‌کنید، حالت‌های تکراری مانند یک حالت پایدار جدید به جدول اضافه می‌شوند و این جریمه‌ای است که برای رفع مشکل مسابقه باید پرداخت کرد.

جدول 5-۲۰ – جدول روند پس از تخصیص حالت

حالت اولیه	$Y_1y_2y_3$	00	01	11	10
(2)	000	000	100	001	000
(2)	001	001	000	011	001
(1)	010	000	010	010	010
(4)	011	111	011	011	001
(6)	100	000	100	101	100
(3)	101	001	101	101	100
(5)	110	110	111	010	110
(5)	111	111	101	110	111

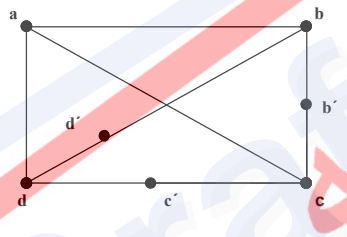
نسخه پیش‌نویس Draft

مثال ۹)

جدول 5-21 – جدول روند مربوط به مثال ۳ بخش ۵-۸-۲

حالت اولیه	ورودیهای x_1x_2 00	01	11	10
a	a, 0	c, 0	a, 0	b, 0
b	a, 1	b, 1	c, 1	b, 0
c	d, 0	c, 0	c, 1	d, 1
D	d, 0	b, -	a, -	d, 1

باتوجه به مجاورت‌های موجود، گراف مجاورت به صورت زیر خواهد بود. در این مثال می‌خواهیم نشان دهیم که برای تکرار کردن حالت می‌توان از گراف مجاورت کمک گرفت. در این مثال برای حل مشکل، سه حالت اضافه می‌شود که در واقع تکرار حالت‌های متناظر در گراف هستند.



بنابراین جدول کارنو تخصیص حالت بصورت زیر بدست می‌آید.

y_3 \ y_1y_2	00	01	11	10
0	a	b	b'	c
1	d	d'	-	c'

جدول 5-۲۲ – جدول روند مربوط به مثال ۳ پس از تخصیص حالت

$y_1y_2y_3$	حالت اولیه	ورودیهای x_1x_2 00	01	11	10
000	A	000	010	000	001
001	B	000	001	011	001
011	b´	001	011	010	011
010	C	110	010	010	110
110	c´	100	110	110	100
100	D	100	101	000	100
101	d´	101	001	100	101
	-	-, -		-, -	-, -

نسخه پیش‌نویس Draft

مثـال ۱۰)

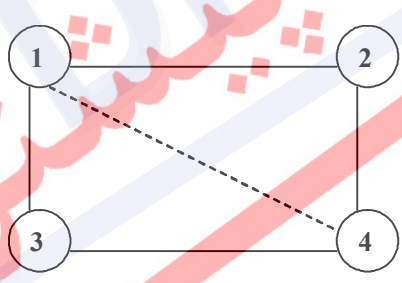
جدول 5-۲۳ – جدول روند مربوط به مثال ۴ بخش ۵-۸-۲

حالت اولیه	ورودیهای x_1x_2 00	01	11	10
1	①	2	3	①
2	②	②	4	1
3	1	4	③	③
4	1	④	④	3

جدول 5-۲۴ – مشخص کردن حالات مجاور

حالت	حالت مجاور
1	2, 3, 4
2	1, 4
3	1, 4
4	1, 2, 3

گراف مجاورت برای این جدول در زیر مشخص شده است. اگر به این گراف دقت کنید، در صورتی که مجاورت حالت 1 و 4 وجود نداشت، احتیاج به اضافه کردن حالت نداشتیم.



بنابراین اگر در سطر چهارم جدول حالت 1 با 3 جابجا شود این اتفاق رخ می‌دهد و مشکلی نیز برای مدار پیش نمی‌آید. چون مدار از طریق حالت 3 به حالت 1 گذر می‌کند و فقط یک گذر اضافی انجام می دهد که زیاد مهم نیست. اما برای تخصیص حالت نیازی به اضافه کردن حالت ندارد و جدول کارنو تخصیص حالت آن بصورت زیر خواهد شد.

جدول 5-25 – جدول تخصیص حالت مربوط به مثال ۴

	0	1
0	1	3
1	2	4

نسخه پیش‌نویس Draft

۵-۹-۲-۲ روش مجموعه سطر اشتراکی[1]

این روش جزء روشهای یک کد است و برای تخصیص حالت، به هر حالت یک کد داده می‌شود. همانطور که از نام این روش مشخص است، در جدول کارنوی تخصیص حالت، تکرار نداریم و بجای آن خانه‌های جدول را به اشتراک می‌گذاریم. در ادامه نحوه عملیات را با چند مثال نشان می‌دهیم.

مثال (۱۱)

جدول 5-26 – جدول روند مربوط به مثال ۵ بخش ۵-۸-۲

حالت اولیه	ورودیهای x_1x_2 00	01	11	10
1	①	2	3	①
2	4	②	②	-
3	1	③	③	-
4	④	5	④	-
5	6	⑤	2	-
6	⑥	3	4	⑥

جدول ۵-۲۷ – مشخص کردن حالات مجاور

حالت	حالت مجاور
1	2, 3
2	1, 4, 5
3	1, 6
4	2, 5, 6
5	2, 4, 6
6	3, 4, 5

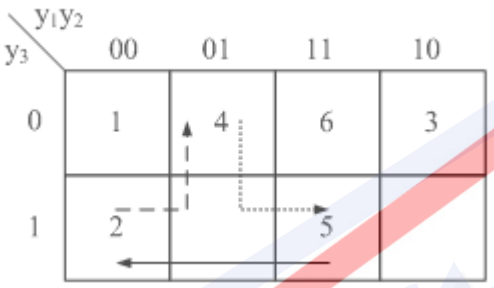
اکنون برای هر ستون جدول روند، یک جدول رسم می‌کنیم:

[1] Shared Row Set

نسخه پیش‌نویس Draft

جدول 5-۲۸ – رسم جداول مربوط به ستون‌های جدول روند

P. S	N. S	P. S	N. S	P. S	N. S	P. S	N. S
1	1	1	2	1	3	1	1
2	4	2	2	2	2	2	-
3	1	3	3	3	3	3	-
4	4	4	5	4	4	4	-
5	6	5	5	5	2	5	-
6	6	6	3	6	4	6	6



همانطور که مشخص است، در هر ستون جدول خانهٔ روند، یک خانه نباید دوبار مورد استفاده قرار گیرد و مشترک شود وگرنه کدگذاری غلط می‌شود. جدول روند نهایی در زیر مشخص شده است.

جدول 5-29 – جدول روند مربوط به مثال ۵ پس از تخصیص حالت

حالت اولیه	$y_1y_2y_3$	00	01	11	10
۱	000	000	001	100	000
۲	001	011	001	001	-
۴	010	010	011	010	-
مشترک	011	010	111	001	-
۳	100	000	100	100	-
تهی	101	-	-	-	-

نسخه پیش‌نویس Draft

۶	110	110	100	010	110
۵	111	110	111	011	-

نسخه پیشنویس
Draft

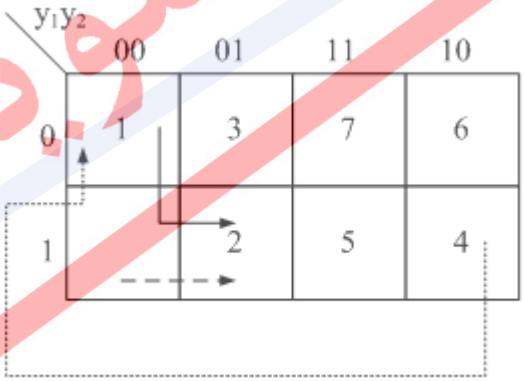
مثـال ۱۲)

جدول 5-30 – جدول روند مربوط به مثال ۶ بخش ۵-۸-۲

حالت اولیه	ورودیهای x_1x_2 00	01	11	10
1	①	2	3	①
2	3	②	②	4
3	③	7	③	③
4	1	④	④	④
5	⑤	⑤	2	7
6	-	4	⑥	1
7	5	⑦	6	⑦

جدول 5-31– مشخص کردن حالات مجاور

حالت	حالت مجاور
1	2, 3, 4, 6
2	1, 3, 4,5
3	1, 2, 7
4	1, 2, 6
5	2, 7
6	1, 4, 7
7	3, 5, 6



نسخه پیش‌نویس
Draft

۵-۱۰ تمرین

۱- تحقیق کنید که آیا جدول روند مثال اول، SOC است یا خیر؟

۲- تحقیق کنید که آیا جدول روند مثال دوم، MOC است یا خیر؟

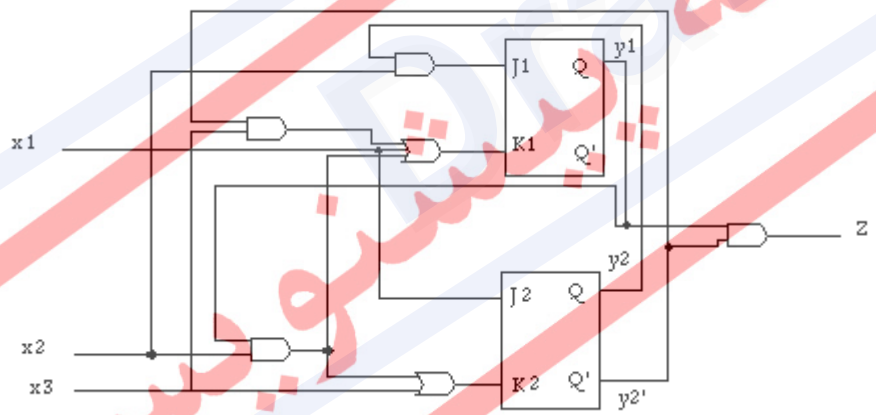
۳- آیا این دو جدول مداساسی هستند یا خیر؟

۴- مثال ۳ را تکرار کنید با فرض اینکه عمل (+2) بصورت (-1) انجام می‌شود.

۵- مدار با ورودی پالس نشان داده شده در شکل را تحلیل کنید:

الف-جدول حالت را تعیین کنید.

ب-طراحی یک دیاگرام زمانی برای مدار به منظور پاسخ گفتن به ترتیب ورودی زیر،به این فرض که x1,x2,x3,y1,y2,J1,K1,J2,K2,Y1,Y2 و Z بر روی دیاگرام باشند.



۶- یک مدار مد پایه با ویژگیهای زیر مطلوبست:دو ورودی (x1,x2) و یک خروجی z نیاز هستند.خروجی z=0 زمانی که x1=x2 است ایجاد می شود.زمانی که x1=0 و x2 از ۱ به ۰ تغییر می یابد یک خروجی z=1 باید حادث گردد.زمانی که x1=1 و x2 از ۱ به ۰ تغییر می یابد یک خروجی z=1 باید حادث شود.از طرف دیگر هیچ تغییر در ورودی ، تغییری در خروجی را سبب نخواهد شد.با توجه به شرح یک جدول جریان طراحی کنید.

۷- یک جدول جریان اولیه برای یک مدار در مد پایه و با توجه به ویژگیهای زیر طراحی کنید.مدار بایستی دارای دو ورودی (x1,x2) و دو خروجی (z1,z2) می باشد.زمانی که ورودی

ها با هم برابر و مساوی ۰ هستند خروجی ها هم بایستی با هم برابر و مساوی ۰ باشند.اگر x1=1 و x2 از ۰ به ۱ تغییر کند ، یک خروجی z1=1 و z2=0 ایجاد خواهند شد.اگر x2=1 و x1 از ۰ به ۱ تغییر کند خروجی z1=1 و z2=0 ایجاد خواهند شد.خروجی های مساوی و برابر ۰ تنها زمانی ایجاد خواهد شد که هر دو ورودی ۰ باشند.با سایر تغییرات در ورودی ها خروجی تغییری نمی کند.

۸- جدول جریان اولیه زیر را به یک جدول با سطرهای تقلیل یافته تبدیل کنید.

	x1x2			
	00	01	11	10
1	①/0	2/-	-/-	3/-
2	4/-	②/1	5/-	-/-
3	1/-	-/-	5/-	③/0
4	④/-	2/-	-/-	6/-
5	-/-	2/-	⑤/-	6/-
6	1/-	-/-	5/-	⑥/1

۹- یک جدول جریان سازگار با سطور تقلیل یافته برای جدول جریان اولیه زیر طراحی کنید.

	x1x2			
	00	01	11	10
1	①/0	2/-	-/-	4/-
2	1/-	②/0	3/-	-/-
3	-/-	2/-	③/0	8/-
4	5/-	-/-	7/-	④/1
5	⑤/1	6/-	-/-	4/-
6	5/-	⑥/1	7/-	-/-
7	-/-	6/-	⑦/1	8/-
8	1/-	-/-	3/-	⑧/0

نسخه پیش‌نویس Draft

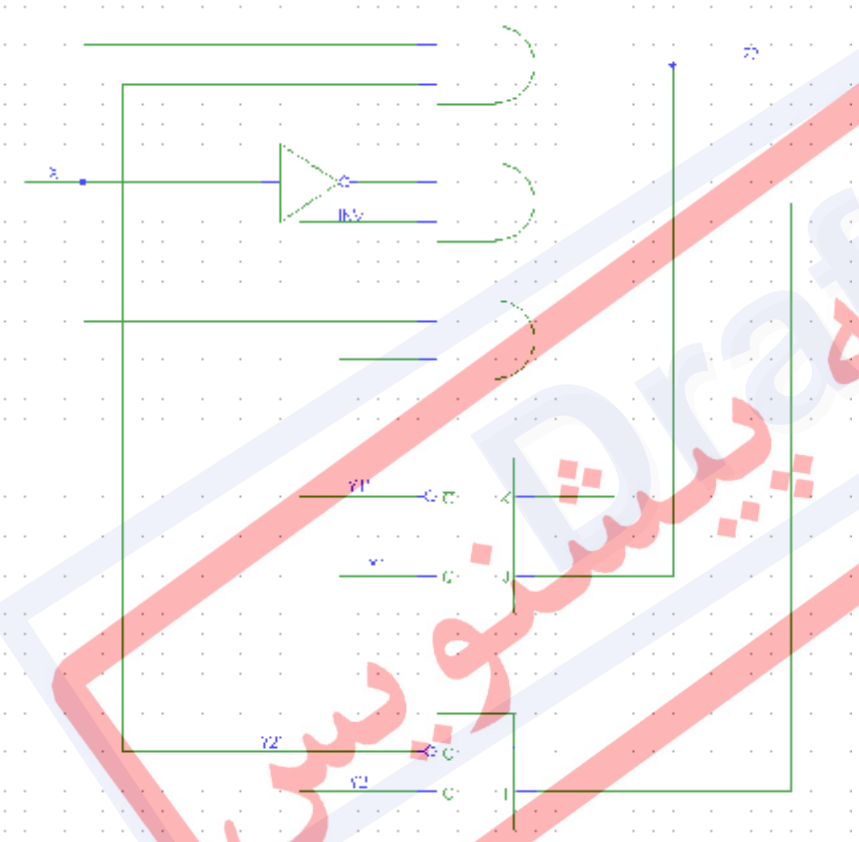
۱۰- یک NAND دو ورودی از یک مدار که دارای دو ورودی (x1,x2) و یک خروجی z و شرایط زیر را دارد بیابید.اولاً z=0 موقعی که x1=0 . خروجی z به سطح منطقی ۱ می رود ، بر اثر اولین تغییر از ۱ به ۰ در x2 زمانی که x1=1 . خروجی در منطق ۱ باقی خواخد ماند تا زمانی که x1 به ۰ بازگردد.

۱۱- مدار ترتیبی ناهمگام شکل را بررسی کنید در حالتی که ورودی مدار که x می باشد در حالت پالسهای سنکرون هستند.موارد زیر را تعیین کنید:

الف-دیاگرام زمانی اگر x=01010010100 و y01y20=11 .

ب-جدول حالت

ج-دیاگرام حالت



نسخه پیش‌نویس Draft

۱۲- یک مدار برای جدول حالت تقلیل یافته جدول جریان زیر بیابید.از روش تخصیص حالت شاخص دار استفاده کنید.فرض کنید گیتهای AND و OR و NOT برای این طراحی در دسترس هستند.

x1x2

Y1Y2		۰۰	۰۱	۱۱	۱۰
۰۰	a	$\bar{a}/0$	$\bar{a}/1$	b/-	c/-
۰۱	b	$a/-$	$\bar{b}/0$	$\bar{b}/0$	d/-
۱۱	c	$a/-$	$a/-$	$\bar{c}/1$	$\bar{c}/1$
۱۰	d	$a/-$	b/-	c/-	d/0

نسخه پیشنویس
Draft

فصل ۶

تأخیر و مخاطره

نسخه پیشنویس
Draft

۶-۱ مقدمه

به هنگام طراحی مدارهای ترتیبی ناهمگام، برای عملکرد درست مدار باید محدودیت‌ها و احتیاط‌های معینی را در نظر گرفت. مدار مورد نظر باید در مد اساسی کار کند، یعنی در هر لحظه از زمان فقط یک ورودی تغییر کند و باید فاقد مسابقه‌های بحرانی باشد. علاوه بر آن پدیدهٔ دیگری نیز وجود دارد که مخاطره[1] نامیده می‌شود و ممکن است باعث نقص در کار مدار شود.

این پدیده در اثر سوییچ کردن‌های زودگذر و ناخواسته در خروجی یک مدار است، زیرا مسیرهای مختلف در مدار، دارای تأخیرهای انتشار متفاوت می‌باشند. مخاطره‌ها در مدارهای ترکیبی رخ می‌دهند و سبب تولید یک مقدار خروجی نادرست آنی می‌گردند. وقتی که در مدارهای ترتیبی ناهمگام چنین وضعیتی رخ می دهد، ممکن است نتیجه آن یک انتقال به یک حالت پایدار نادرست باشد. بنابراین لازم است که عامل ایجاد این مشکل که تأخیر می‌باشد، دقیقتر بررسی شود.

۶-۲ تقسیم بندی تأخیر

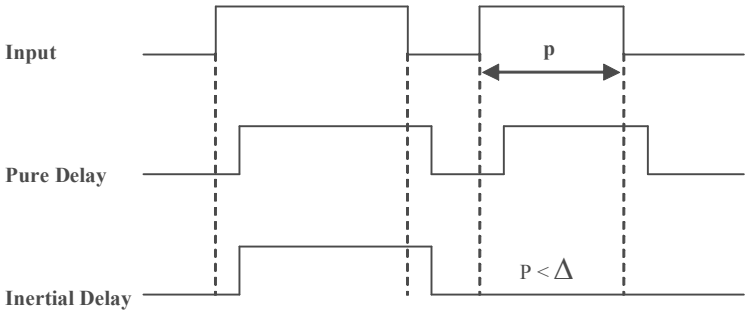
- **تأخیرذاتی**[2]: تأخیر مربوط به گیت‌ها و خطوط ارتباطی است و وابسته به تکنولوژی گیت است. تأخیر ذاتی، عامل اصلی ایجاد مخاطرهٔ اساسی است زیرا اگر تأخیر اضافه شده باعث مخاطره شود، می‌توان آن را حذف کرد.
- **تأخیر اضافه شده**[3]: بر حسب ضرورت و نیاز به مدار اضافه می‌شود و به دو دسته تقسیم می‌شود.
  - **تأخیر خالص**[4]: فقط یک تأخیر مشخص اعمال می‌کند و شکل موج ورودی و خروجی یکسان است. در واقع شبیه مدار شیفت عمل می‌کند. مثلاً دو گیت NOT پشت سر هم یک چنین تأخیری را ایجاد می‌کند.
  - **تأخیرInertial**: یک تأخیر مشخص به سیگنال ورودی اعمال می‌کند، به شرط اینکه پهنای پالس ورودی از مقدار مشخصی مثل Δ بزرگتر باشد، در غیر اینصورت خروجی صفر خواهد بود. مثلاً می‌توان با استفاده از یک فیلیپ-فلاپ نوع D، تأخیر Inertial ایجاد کرد، به این صورت که Δبرابر زمان نگهداری فیلیپ-فلاپ فرض شود.

---

[1] Hazard
[2] Stray Delay
[3] Inserted
[4] Pure Delay

نسخه پیش‌نویس Draft

برای مثال، در شکل ۱ نمونه‌ای از هر دو نوع تأخیر اضافه شده آورده شده است.

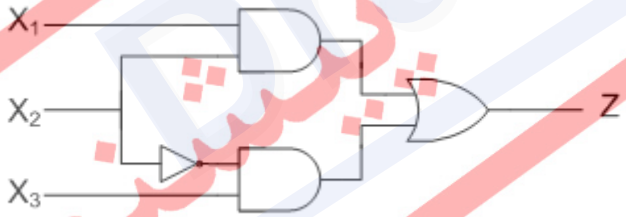


شکل 6-1- تأخیر خالص و تأخیرInertial

۶-۳ تعریف مخاطره

یک مدار دارای مخاطره است، اگر بخاطر بعضی ترکیبات ورودی ممکن، تأخیر ذاتی باعث ایجاد پالس‌های ناخواسته در خروجی و یا رفتن مدار به یک حالت پایدار ناصحیح شود. برای درک بهتر مطلب، به مثال زیر توجه کنید.

مدار شکل ۲ را در نظر بگیرید:

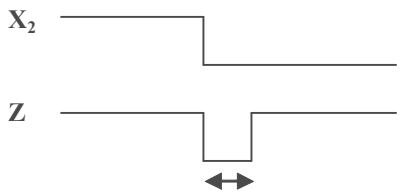


$$Z = x_1 x_2 + x_3 \overline{x_2}$$

شکل 6-۲ - مدار دارای مخاطره

اگر مقادیر $x_1 = x_2 = x_3 = 1$ باشند، مقدار $z=1$ است، اما با تغییر x_2 از ۱ به ۰ به علت تأخیر ذاتی گیت NOT ، در چند لحظه مقدار خروجی اشتباه و غیر معتبر خواهد بود. که در شکل ۳ مشاهده می‌نمایید:

نسخه پیش‌نویس Draft



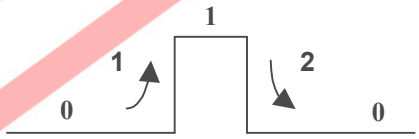
شکل 6-3 - اشتباه در مقدار خروجی

برای بررسی دقیقتر و پیدا کردن راه حل‌های مناسب برای مشکل مخاطره، در ادامه این بحث را ابتدا در مورد مدارهای ترکیبی شروع می‌کنیم و سپس به سراغ مدارهای ترتیبی می‌رویم.

۶-۴ انواع مخاطره در مدارهای ترکیبی

در مدار ترکیبی، یک مدار دارای مخاطره است اگر به خاطر بعضی از ترکیبات ورودی، خروجی مدار از حالت(های) ناخواسته گذر کند. این مخاطره‌ها بصورت زیر تقسیم بندی می‌شوند.

- **مخاطرهٔ ایستا**[1]: خروجی مدار باید در یک تحریک ورودی ثابت بماند، ولی با تغییر ورودی، خروجی نیز یکبار تغییر می‌کند. (و یا به تعداد زوج تغییر می‌کند) و این به دلیل تأخیر خود مدار است. که خودشامل دو نوع است:
  - **مخاطرهٔ صفر**[2]: درتمام حالات، خروجی برابر صفر است. اما در اثر تغییر ورودی، مدت کوتاهی یک می‌شود و سپس به حالت اولیه برمی‌گردد. البته می‌تواند با تغییر ورودی، خروجی به تعداد زوج بیش از دو بار نیز تغییر کند (در واقع می‌تواند با تعداد زوج تغییر به حالت اولیهٔ خود برگردد).

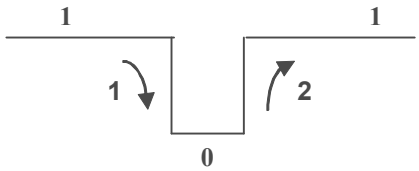


  - **مخاطره یک**[3]: درتمام حالات، خروجی برابر یک است. اما در اثر تغییر ورودی، مدت کوتاهی صفر می‌شود و سپس به حالت اولیه برمی‌گردد. البته می‌تواند با تغییر ورودی، خروجی به تعداد

[1] Static Hazard
[2] Zero Hazard (0-Hazard)
[3] One Hazard (1-Hazard)

نسخه پیش‌نویس Draft

زوج بیش از دو بار نیز تغییر کند (در واقع می‌تواند با تعداد زوج تغییر به حالت اولیهٔ خود برگردد).



- **مخاطرهٔ پویا**[1]**:** خروجی در اثر تغییر ورودی باید تغییر کند، اما در مدت کوتاهی قبل از تغییر نهایی خروجی، خروجی به تعداد فرد تغییر می‌کند و در نهایت به حالت درست برمی‌گردد. خروجی مدار باید عکس شود، ولی بر اثر تغییر ورودی، با سه تغییر (به تعداد فرد) عکس می‌شود. این موضوع در شکل ۴ نشان داده شده است.

شکل 6-4 - مخاطرهٔ پویا

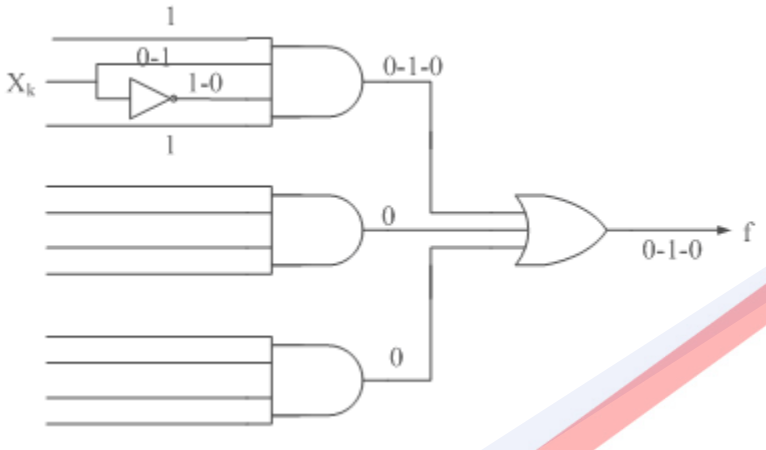
۶-۴-۱ تعاریف مخاطره در مدارهای ترکیبی

- **مدار ایجاد کنندهٔ مخاطرهٔ صفر:** اگر تابع F یک تابع ترکیبی باشد و داشته باشیم $F(I_1) = F(I_2) = 0$ و دو ورودی نیز فقط در یک بیت مانند x_k با هم متفاوت باشند، آنگاه یک پیاده‌سازی دو سطحی بصورت SOP دارای مخاطرهٔ صفر خواهد بود، اگر یک گیت AND داشته باشیم که هر دو ورودی x_k و x'_k را داشته باشد و بقیه ورودی‌های آن یک باشد. همچنین به ازاء I_1 و I_2 خروجی بقیهٔ AND ها صفر باشد.

در اینجا، مقدار ورودی X_k برابر با صفر است. ولی وقتی مقدار آن را یک می‌کنیم، تا گیت NOT مقدار X_k را معکوس کند و به $\overline{X}_k$ تبدیل کند، در یک لحظه مقدار خروجی AND یک می‌شود و سپس به صفر بر می‌گردد و این بدلیل تأخیر در گیت NOT است.

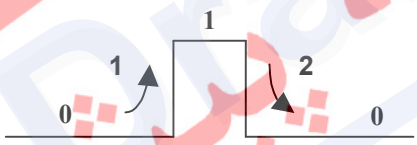
[1] Dynamic Hazard

نسخه پیش‌نویس Draft



شکل 6-۵ - مدار ایجاد کنندهٔ مخاطرهٔ صفر

آنچه در نهایت در خروجی ایجاد می شود مخاطره ی استاتیک صفر است که به صورت زیر است:



- **مدار ایجاد کنندهٔ مخاطرهٔ یک:** اگر تابع F یک تابع ترکیبی باشد و $F(I_1)= F(I_2)=1$ باشد، و این ورودی‌ها فقط در بیت x_k با هم متفاوت باشند، یک پیاده‌سازی دو سطحی SOP، دارای مخاطرهٔ یک خواهد بود اگر به ازاء این ورودی‌ها فقط خروجی یکی از AND ها یک شود.

نسخه پیش‌نویس Draft

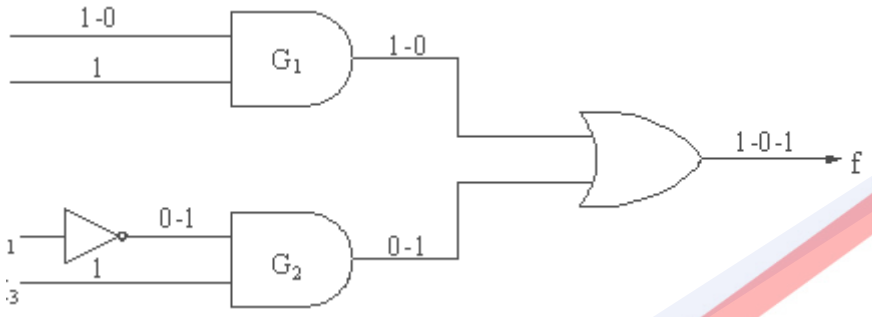
$$f = x_1 x_2 + \bar{x}_1 x_3$$

$$x_1 x_2 x_3$$

$$I_1 = 111$$

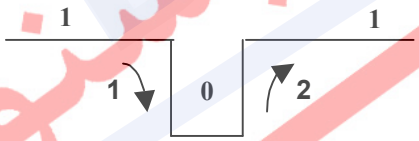
$$x_1 x_2 x_3$$

$$I_2 = 011$$



شکل 6-6 - مدار ایجاد کنندهٔ مخاطرهٔ یک

در اینجا مقدار x_1 در گیت G1 برابر یک است و سپس آنرا تبدیل به صفر می‌کنیم تا گیت NOT مقدار x1 را معکوس کند و به $\bar{x}_1$ تبدیل کند که در G2 مورد استفاده قرار بگیرد. خروجی G1 نیز صفر شده، درنتیجه مقدار خروجی اصلی (f) در یک لحظه صفر شده و سپس یک می‌شود که این از تأخیر در گیت NOT ناشی می‌شود. آنچه در نهایت در خروجی ایجاد می شود مخاطرهٔ استاتیک یک است که به صورت زیر است:



- **مدار ایجاد کنندهٔ مخاطرهٔ پویا:**

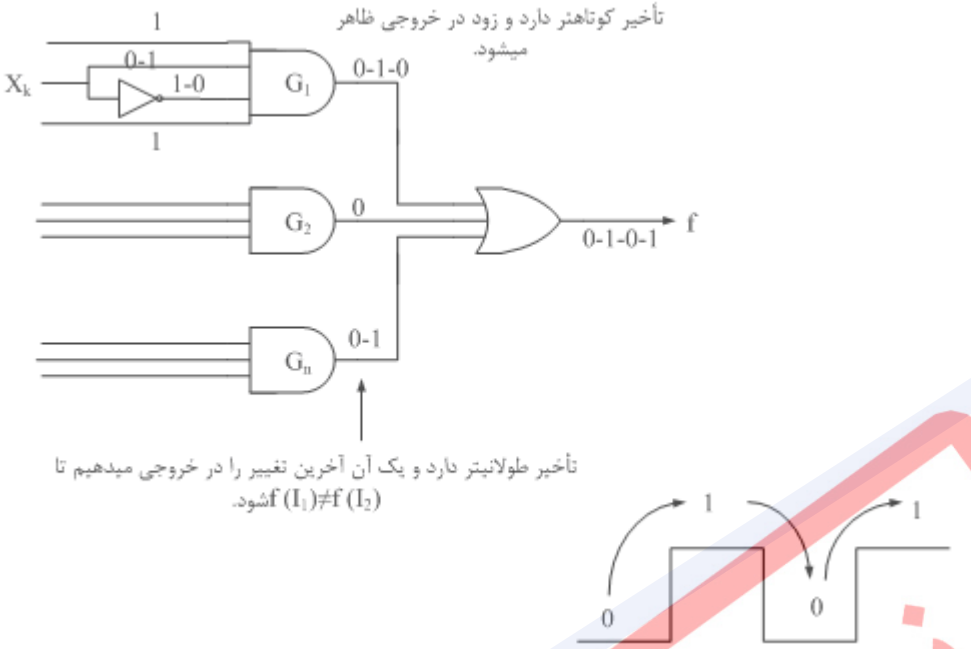
۱- باید یک مدار دو سطحی SOP وجود داشته باشد.

۲- فقط یک ورودی در مدار باید تغییر کند (x_k).

۳- باید یک گیت AND داشته باشیم که ورودی‌اش دارای بیت‌های x_k و x'_k باشد و مابقی ورودی‌هایش برابر با یک باشد.

۴- چون $F(I_1) \neq F(I_2)$ ، باید خروجی یکی از ANDها باشد.

نسخه پیش‌نویس Draft



شکل 6-7- مدار ایجاد کنندهٔ مخاطرهٔ پویا

G1 ابتدا خروجی را بصورت 0-1-0 تغییر می‌دهد تا آنکه Gn آخرین تغییر را در خروجی بوجود آورده و نهایتاً 0-1-0 شود (منظور خروجی است) زیرا $F(I_1) \neq F(I_2)$.

۶-۴-۲ راه حل‌های رفع مخاطره

- **تشخیص مخاطرهٔ صفر:** ساده‌سازی مدار بطوریکه یک ترم بصورت $x_k \bar{x}_k$ باقی بماند.

مثال ۱) فرض کنید در مورد تابع زیر می‌خواهیم این آزمایش را انجام دهیم.

$$E(x) = \bar{x}_1\bar{x}_2 + \bar{x}_1\bar{x}_3 + x_1x_2 + x_1x_3x_4 + \bar{x}_2x_2 + x_2x_3x_4$$

حال اگر در تابع زیر $x_1=1$ و $x_3=0$ باشد، $E(x) = x_2 + \bar{x}_2x_2$ خواهد شد. پس به ازای مقادیر بالا، از نتیجهٔ $E(x)$ متوجه می‌شویم که مخاطرهٔ صفر نداریم.

- **تشخیص مخاطرهٔ یک:** ساده ترین راه، رسم جدول کارنو و پیدا کردن یک‌های مجاور است که در مرحلهٔ ساده‌سازی لحاظ نشده‌اند.

نسخه پیش‌نویس Draft

برای مثال، در مدار زیر ترم x_2x_3 در عبارت اولیه نیست که از نظر منطقی صحیح است. اما باعث ایجاد مخاطرهٔ یک می‌شود. بنابراین برای حل مشکل باید این ترم نیز به مدار اضافه شود.

x_3 \ x_1x_2	00	01	11	10
0	0	0	1	0
1	1	1	1	0

$$f = x_1x_2 + \bar{x}_1x_3$$

$$f = x_1x_2 + \bar{x}_1x_3 + x_2x_3$$

۶-۴-۲-۱ منطق سه تایی برای تشخیص مخاطره‌های ایستا

استفاده از منطق سه تایی یک روش آسان برای یافتن مخاطره‌های ایستا می‌باشد. نحوهٔ عمل آن هم به این صورت است که سه ارزش منطقی شامل 0، ½ و 1 به جای دو ارزش معمول 0 و 1 در اختیار داریم و هر تغییر صفر به یک یا یک به صفر در ورودی‌ها از مقدار ½ نیز عبور می‌کند. بنابراین داریم :

حالت‌های درست $\begin{cases} 0 - \frac{1}{2} - 1 \\ 1 - \frac{1}{2} - 0 \end{cases}$

مخاطرهٔ یک $1 - \frac{1}{2} - 1$

مخاطرهٔ صفر $0 - \frac{1}{2} - 0$

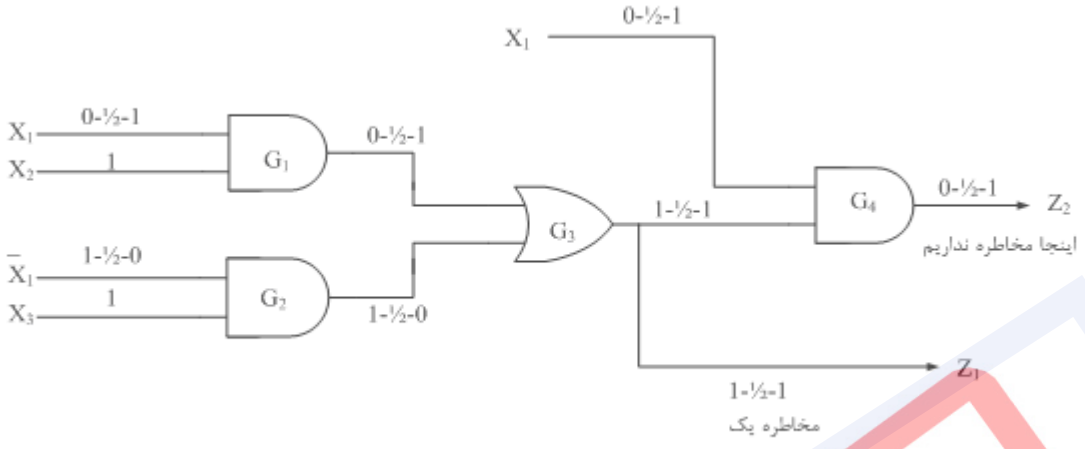
در این منطق، گیت AND مینیمم ورودی را وگیت OR ماکزیمم ورودی را می‌دهد. NOT مقدار ½ نیز برابر با خودش است. لازم به ذکر است که در محاسبهٔ منطق سه مقداره برای مدارات ترتیبی، باید چندین بار مقادیر به دست آمده را در ورودی مدار قرار داد تا به جواب نهایی دست یافت.

مثـال۲) می‌خواهیم ببینیم که برای مدار زیر و برای ورودیهای $I_1 \leftrightarrow I_2$ مخاطرهٔ ایستا داریم یا خیر.

$I_1 = 0\ 1\ 1$ و $I_2 = 1\ 1\ 1$ می‌باشد.

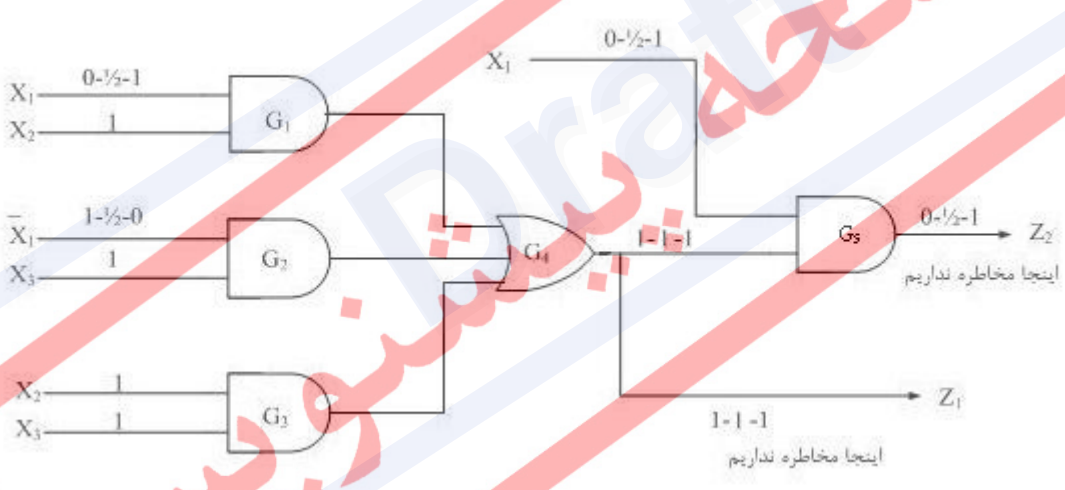
همانطور که ملاحظه می‌کنید در خروجی Z_1 یک مخاطرهٔ یک داریم.

نسخهٔ پیش‌نویس Draft



مثـال ۳) می‌خواهیم ببینیم که برای مدار زیر و برای ورودیهای $I_1 \leftrightarrow I_2$ مخاطره ایستا داریم یا خیر.

I_1=011 و I_2=111 . همانطور که ملاحظه می‌کنید در این مدار مخاطرهٔ ایستا وجود ندارد.



نسخه پیش‌نویس Draft

- **تشخیص مخاطرهٔ پویا:** برای پیدا کردن مخاطرهٔ پویا، باید حتماً تابع را به یکی از دو شکل زیر $x_i + \bar{x}_i x_i$ و یا $x_i(x_i + \bar{x}_i)$ درآوریم. در غیر این صورت، مخاطرهٔ پویا نداریم.

باید توجه داشت که برای پیدا کردن مخاطرهٔ پویا، نمی‌توان هیچ عملی روی تابع انجام داد (از قبیل ضرب و غیره). برای مثال در تابع زیر اگر $x_1=0$ و $x_3=1$ و $x_4=0$ باشد، داریم:

$$E(x) = \bar{x}_1(\bar{x}_2 + \bar{x}_3) + (x_1 + \bar{x}_2)(x_2 + x_3x_4)$$
$$= 1(\bar{x}_2 + 0) + (0 + \bar{x}_2)(x_2 + 0)$$
$$E(x) = \bar{x}_2 + \bar{x}_2x_2$$

بنابراین در این تابع مخاطرهٔ پویا وجود دارد.

هیچ روش اصولی برای حذف این نوع مخاطره وجود ندارد ولی می‌توان با کمک فیلیپ-فلاپ و یا استفاده از تأخیر Inertial این نوع مخاطره را حذف کرد.

۶-۴-۲-۲ کشف مخاطره با استفاده از منطق هشت مقداره

منطق ۸-مقداری را می‌توان برای کشف مخاطرهٔ ایستا و پویا در مدارهای ترکیبی بکار برد. ۸ مقدار ممکن یک علامت الکتریکی عبارتند از: 0، 1، + که مشخص کنندهٔ انتقال 0→1 می‌باشد، - که مشخص کنندهٔ انتقال 1→0 می‌باشد، S_0 که مشخص کنندهٔ مخاطرهٔ ایستای 0 می‌باشد، S_1 که مشخص کنندهٔ مخاطرهٔ ایستای 1 می‌باشد، +D که مشخص کنندهٔ مخاطرهٔ پویا در انتقال 0→1 می‌باشد و -D که مشخص کنندهٔ مخاطرهٔ پویا در انتقال 1→0 می‌باشد. بر خلاف منطق سه مقداره، در اینجا فقط یکبار محاسبه کافی می‌باشد. جدول‌های زیر، خروجی‌های عناصر منطقی AND, OR و معکوس کننده را برای منطق ۸- مقداری نشان می‌دهد.

جدول 6-1 - خروجی‌های عنصر منطقی AND برای منطق ۸- مقداری

AND	1	0	+	-	S1	S0	D+	D-
1	1	0	+	-	S1	S0	D+	D-
0	0	0	0	0	0	0	0	0
+	+	0	+	S0	D+	S0	D+	S0
-	-	0	S0	-	D-	S0	S0	D-
S1	S1	0	D+	D-	S1	S0	D+	D-
S0	S0	0	S0	S0	S0	S0	S0	S0
D+	D+	0	D+	S0	D+	S0	D+	S0
D-	D-	0	S0	D-	D-	S0	S0	D-

نسخه پیش‌نویس Draft

جداول 6-۲ و 6-۳ - خروجی‌های عناصر منطقی OR و NOT برای منطق ۸- مقداری

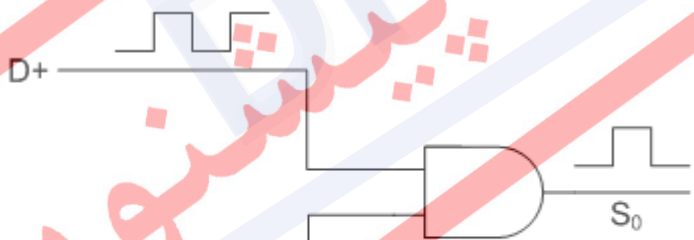
OR	1	0	+	-	S1	S0	D+	D-
1	1	1	1	1	1	1	1	1
0	1	0	+	-	S1	S0	D+	D-
+	1	+	+	S1	S1	D+	D+	S1
-	1	-	S1	-	S1	D-	S1	D-
S1	1	S1	S1	S1	S1	S1	S1	S1
S0	1	S0	D+	D-	S1	S0	D+	D-
D+	1	D+	D+	S1	S1	D+	D+	S1
D-	1	D-	S1	D-	S1	D-	S1	D-

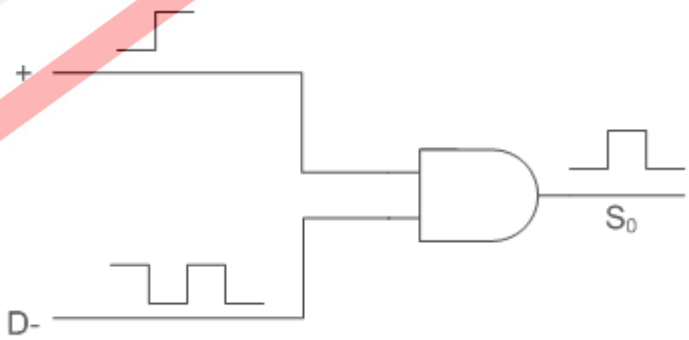
NOT

x	$\overline{X}$
0	1
1	0
+	-
-	+
S0	S1
S1	S0
D+	D-
D-	D+

برای مثال، چند نمونه از موارد جدول در زیر آمده است:

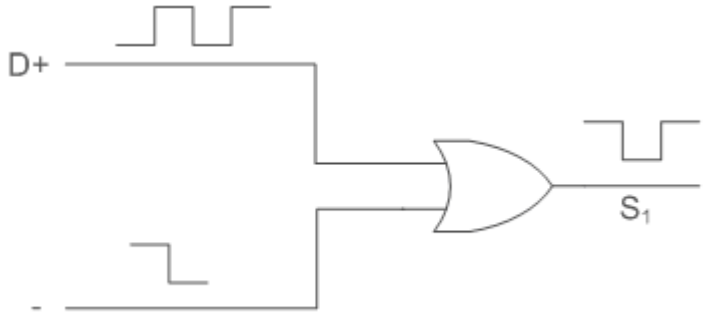
گیت AND حداقل ورودی را عبور می‌دهد.





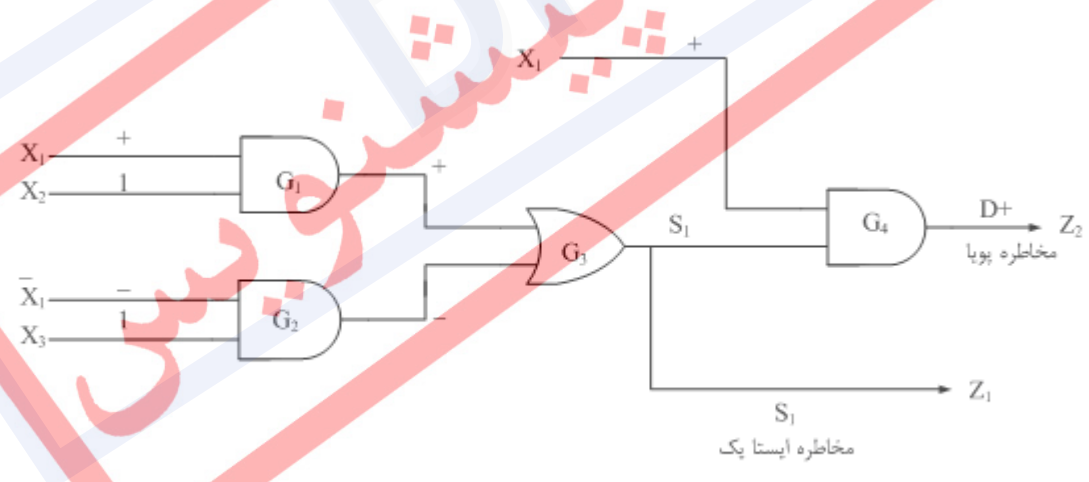
نسخه پیش‌نویس

گیت OR حداکثر ورودی را عبور می‌دهد.



این خروجی برای بدترین حالت است که در لحظه ای پهنای پالسهای +D و -D برابر نباشند.

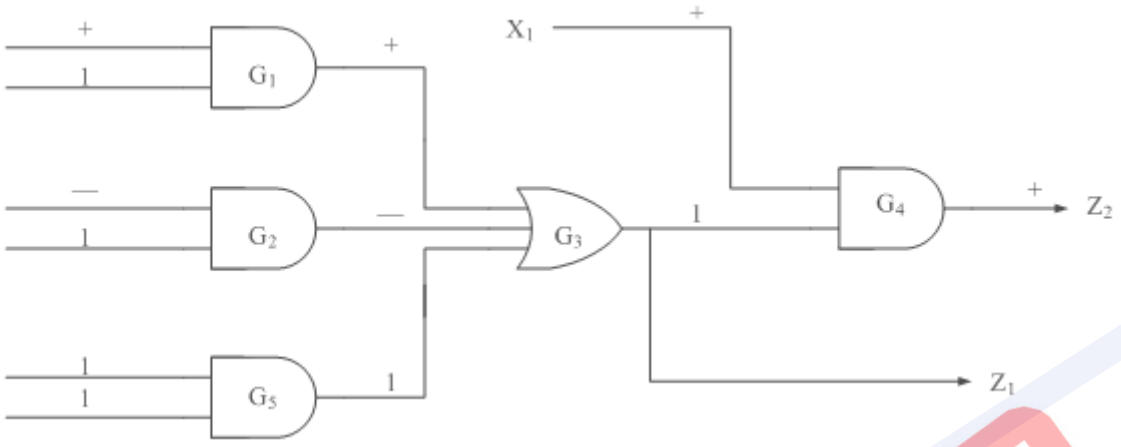
مثال ۴) برای مدار شکل ۸، انتقال $I_1 \rightarrow I_2$ که $I_1=011$ و $I_2=111$ می‌باشد، ورودی‌های G_1 برابر +, 1 و خروجی آن + می‌باشد. ورودیهای G_2 برابر -, 1 و خروجی آن − میباشد. ورودیهای G_3برابر + و − و خروجی آن S_1 می‌باشد که مشخص کنندهٔ مخاطرهٔ ایستای 1 در خروجی Z_1 است. ورودی‌های G_4 برابر + و S_1 و خروجی آن +D می‌باشد که مشخص کنندهٔ مخاطرهٔ پویا در Z_2 می‌باشد.



شکل 6-8– انتقال مخاطره به خروجی

همچنین برای مدار شکل ۹، خروجی گیت G_1 برابر +، خروجی G_2 برابر − و خروجی G_5 برابر 1 است. با استفاده از اصول جابجایی و انجمنی که می‌توان نشان داد برای این منطق صحیح می‌باشد، خروجی G_3 برابر 1 و خروجی G_4 برابر + است. در نتیجه این انتقال بدون مخاطره خواهد بود.

نسخه پیش‌نویس Draft



شکل 6-9 – انتقال مخاطره به خروجی

۶-۵ مخاطره در مدارهای ترتیبی

در این بخش به بررسی انواع مخاطره در مدارهای ترتیبی می‌پردازیم. توجه داشته باشید که در مدارهای ترتیبی همگام بدلیل وجود عامل همگام کننده، هیچ نوع مخاطره‌ای ایجاد نمی‌شود و این یکی از محاسن بسیار بزرگ این مدارات است. اما در مقابل، در مدارهای ترتیبی ناهمگام انواع مخاطره‌های زیر اتفاق می‌افتد.

- **مخاطرهٔ حالت**[1]: به دلیل وجود مخاطره در مدار ترکیبی که حالت‌های بعدی را تولید می‌کند، مدار به یک حالت ناپایدار ناخواسته می‌رود.

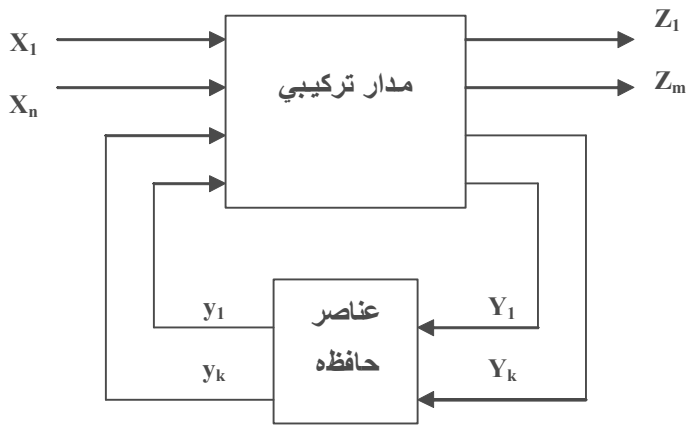
نکته) در صورتیکه برای حالت‌های ناپایدار خروجی تعریف نکنیم، صرفاً عملکرد مدار کند می‌شود. اما اگر برای حالت‌های ناپایدار خروجی تعریف کنیم، ممکن است سبب ایجاد مشکل شده و در خروجی، مخاطره ایجاد شود.

- **مخاطرهٔ خروجی**[2]: به دلیل وجود مخاطره در مدار ترکیبی، در خروجی مخاطره ایجاد می‌شود.

نکته) عامل ایجاد دو مخاطرهٔ بالا، وجود مدار ترکیبی است (شکل پایین).

[1] Steady State Hazard
[2] Output Hazard

نسخه پیش‌نویس Draft

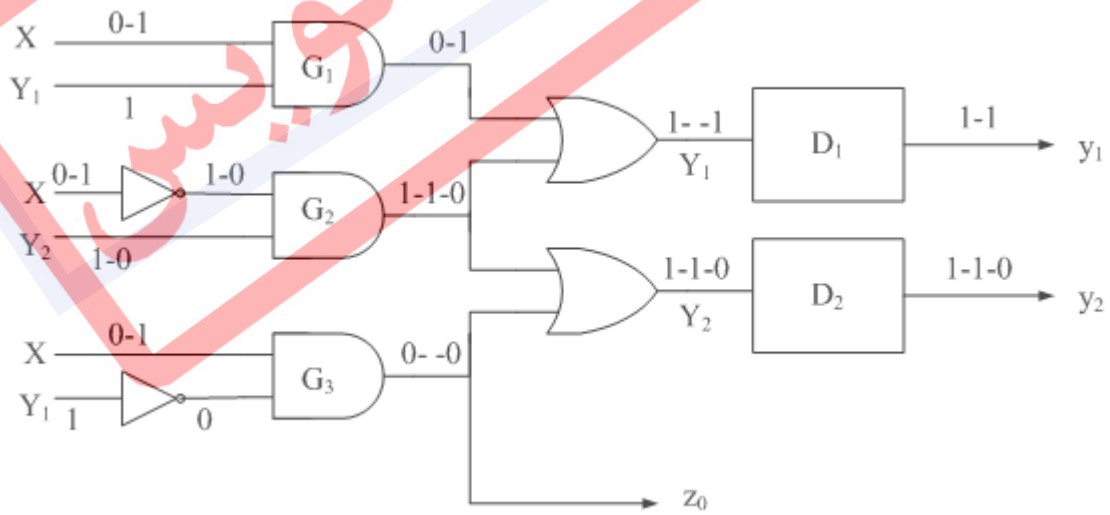


- **مخاطرهٔ اساسی**[1]: به دلیل وجود تأخیرهای نامساوی در مسیر بازخورد، ممکن است به ازای بعضی از انتقال‌های ورودی، مدار به یک حالت پایدار نادرست برود.

مثـال ۵) می‌خواهیم در مدار شکل ۱۰ وضعیت مخاطره‌ها را بررسی کنیم.

جدول 6 - ۴– جدول روند برای مثال ۱

حالت اولیه	X = 0	X = 1	y_1	y_2
1	①,0	2, 1	0	0
2	3, 0	②, 1	0	1
3	③, 0	4, 0	1	1
4	1, 0	④, 0	1	0



شکل 6-۱۰- مدار دارای مخاطرهٔ اساسی

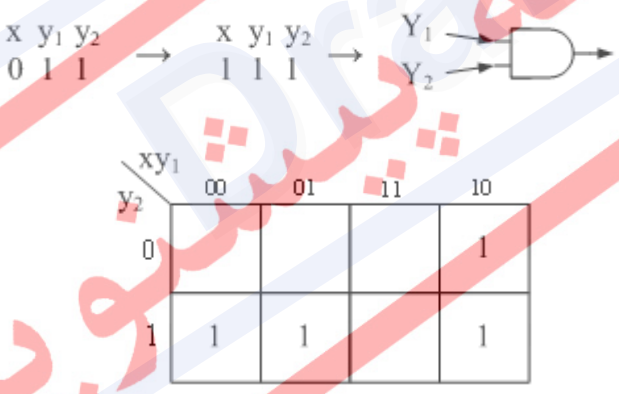
[1] Essential Hazard

نسخه پیش‌نویس Draft

در y_1 یک مخاطره ایجاد می‌شود که از نوع مخاطره یک (در قسمت مدار ترکیبی) می‌باشد و علتش تغییر خروجی G_1، (1-0) و خروجی G_2، (0-1) می‌باشد که ممکن است در y_1 باعث پالس زودگذر $1 \to 0 \to 1$ شود.

حال اگر تأخیر G_1 به حد کافی بیشتر از G_2 شود، مدار بجای آنکه از حالت پایدار 3 $\begin{pmatrix} y_1 & y_2 \\ 1 & 1 \end{pmatrix}$ به حالت پایدار4 $\begin{pmatrix} y_1 & y_2 \\ 1 & 0 \end{pmatrix}$ برود، ممکن است به حالت پایدار 2 $\begin{pmatrix} y_1 & y_2 \\ 0 & 1 \end{pmatrix}$ برود و در آنجا ثابت بماند که وضعیتی غلط است و یک نوع مخاطرهٔ حالت می‌باشد.

یک مخاطرهٔ حالت نیز در انتقال از حالت پایدار 2 $\begin{pmatrix} y_1 & y_2 \\ 0 & 1 \end{pmatrix}$ به حالت پایدار 3 $\begin{pmatrix} y_1 & y_2 \\ 1 & 1 \end{pmatrix}$ وجود دارد که ممکن است به جای حالت پایدار 3 $\begin{pmatrix} y_1 & y_2 \\ 1 & 1 \end{pmatrix}$ به حالت پایدار 1 $\begin{pmatrix} y_1 & y_2 \\ 0 & 0 \end{pmatrix}$ برود. در نتیجه، برای حذف این مخاطره، باید دو گیت اضافی بکار برد؛ یکی برای اضافه کردن y_1y_2 در روبرو و دیگری برای $\bar{y}_1y_2$ برای جدول کارنوی زیر:



$$Y_2 = \bar{x}y_2 + x\bar{y}_1$$

در ضمن، در مدار صفحهٔ قبل برای انتقال از حالت 3 به 4 (در صورتی که در گیت G_3 تأخیر خود گیت از NOT ورودیش کمتر باشد)، Z (خروجی) نیز دارای مخاطره است، زیرا G_3 هم مانند G_1 مخاطره دارد. در نتیجه، Z هم مخاطره خواهد داشت و مخاطرهٔ y_1 و y_2 از نوع مخاطرهٔ حالت می‌باشد. برای رفع این دو نوع مخاطره، باید مخاطره‌های مدارهای ترکیبی موجود در مدار ترتیبی را همانطور که در قسمت‌های قبل ذکر شد، برطرف کرد.

نسخه پیش‌نویس

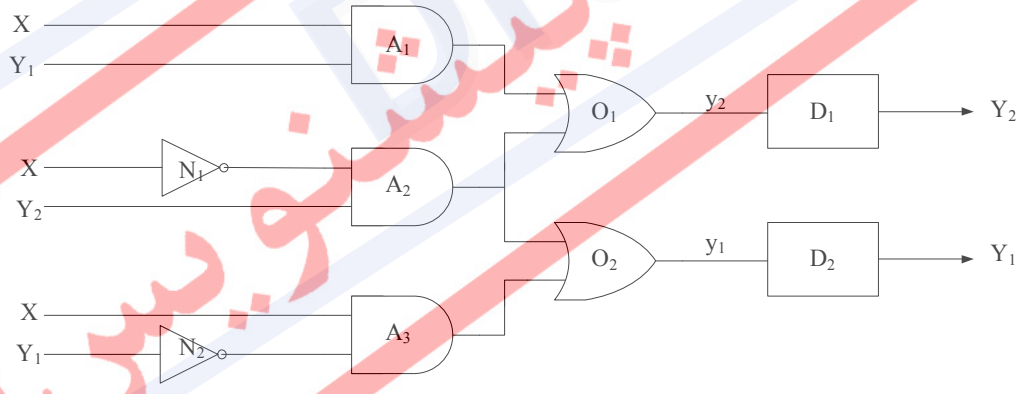
۶-۵-۱ مخاطرهٔ اساسی در مدارهای ترتیبی ناهمگام

مهمترین مخاطره در مدارهای ترتیبی ناهمگام، مخاطرهٔ اساسی است. همانطور که ملاحظه نمودید، دو مخاطرهٔ دیگر وابسته به قسمت ترکیبی مدار است که آسان‌تر می‌توان آنها را تشخیص داده و رفع نمود. اما در مورد مخاطرهٔ اساسی، کار مشکل‌تر است. برای مثال، مدار زیر را در نظر می‌گیریم.

با فرض اینکه تأخیر N_1 بسیار بیشتر از همهٔ اجزاء و مسیر فیدبک باشد، به بررسی یک نمونه از مخاطرهٔ اساسی در جدول روند زیر می‌پردازیم. شکل معادل آن نیز در ادامه آورده شده است.

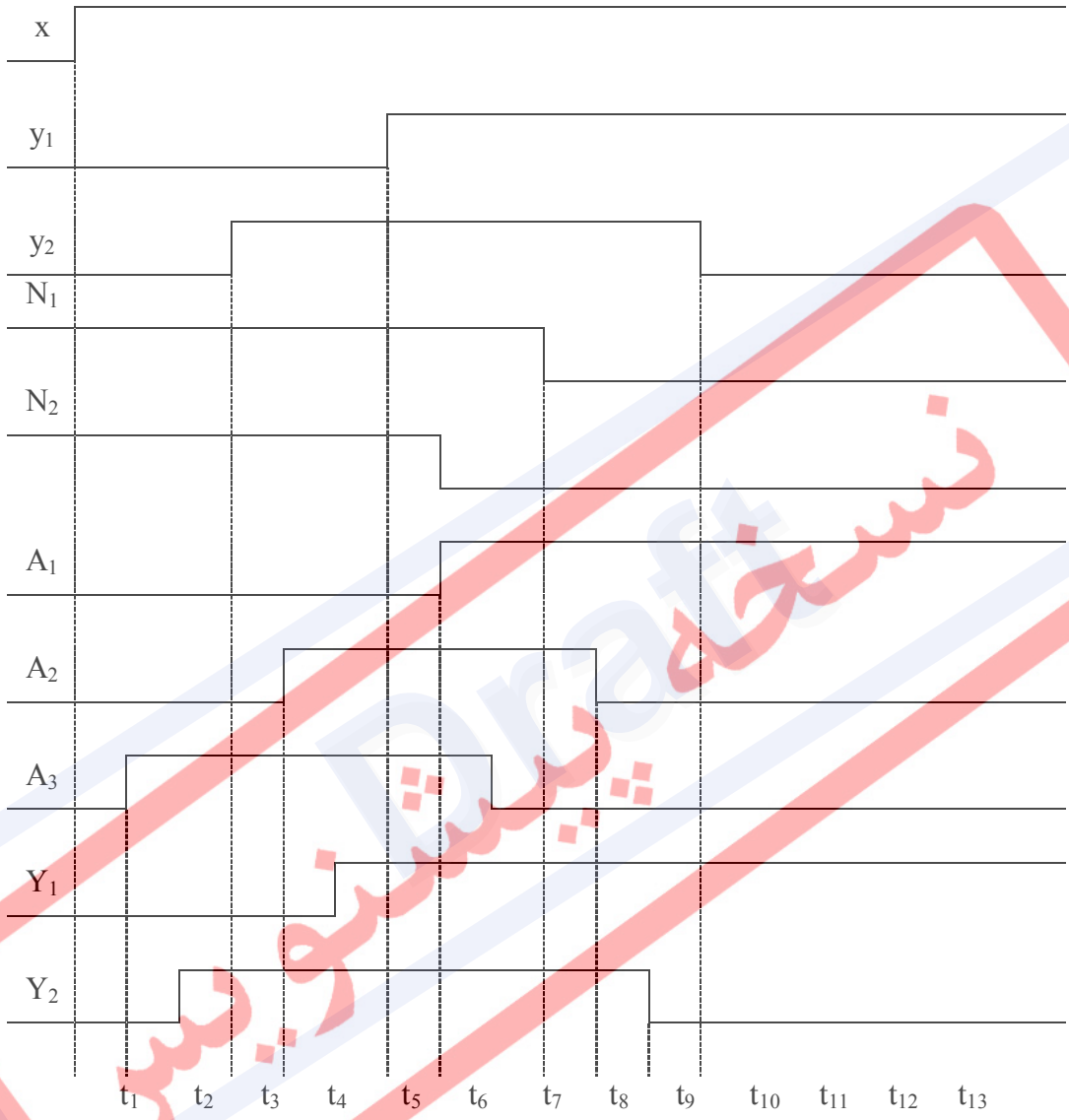
جدول 6-۵ – جدول روند مربوط به مثال بخش ۶-۴-۱

y_1y_2	حالت اولیه	ورودی X: 0	1
00	(1)	00	01
01	(2)	11	01
11	(3)	11	10
10	(4)	00	10



نسخه پیش‌نویس Draft

بافرض اینکه مدار در حالت 00 پایدار است ($y_1y_2 = 00$) و مقدار ورودی از صفر به یک تبدیل می‌شود ($x = 0 \rightarrow 1$) نمودار زمانی شکل ۱۱ را رسم می‌کنیم:



شکل 6-۱۱ – نمودار زمانی

وقایع بحرانی در زمانهای t_5, t_6, t_{10}, t_{13} اتفاق می‌افتد. در زمان t_5 مدار در حالت 01 قرار دارد که حالت درستی است. اما چون هنوز N_1 به تغییر ورودی پاسخ نداده است، مقدار A_2 ، 1 می‌گردد که باعث می‌شود در t_6، Y_1 برابر 1 شود و سپس A_3 صفر می‌شود. زمانیکه N_1 صفر می‌شود باعث می‌گردد که A_2 در زمان t_{11} صفر شود، و در نتیجه $Y_2=0$ خواهد شد. در t_{13}، چون $Y_2=0$ بوده است، Y_2 نیز صفر خواهد شد که مدار به حالت نادرست $y_1y_2=10$ می‌رود.

راه حل: در مسیر Y_2 باید تأخیر کافی قرار گیرد.

نسخه پیش‌نویس Draft

تشخیص مخاطرهٔ اساسی از جدول روند: فرض کنید I_1 و I_2 دو ورودی مجاور حالت q_i از جدول جریان باشد. بصورتی که $N(q_i, I_1)=q_i$ و $N(q_i, I_2)=q_i$, در آن صورت این جدول (برای گذر $I_1 \rightarrow I_2$ از حالت q_i) دارای مخاطرهٔ اساسی است، اگر با اعمال دنبالهٔ ورودی $I_2I_1I_2$ نتیجهٔ نهایی q_k باشد و داشته باشیم. $q_k \neq q_i$

طبق جدول جریان مثال قبل، از یک حالت پایدار (مانند 1) شروع می‌کنیم و با ورودی I_2 که ورودی اولیهٔ آن ($I_2=0$) بوده با ورودی اولی ($I_2=X=1$) به یک حالت پایدار می‌رویم (حالت 2). حال اگر در اینجا ورودی را عوض کنیم یعنی دوباره ($I_1=X=0$) شود، آنوقت به حالت پایدار 3 می‌رویم. اگر در اینجا دوباره ورودی اول را به آن بدهیم یعنی ($I_2=X=1$) ، آنوقت به یک حالت پایدار 4 می‌رسیم که با حالت پایدار اولیه در لحظهٔ صفر که 2 بود، فرق دارد. یعنی با دو بار تغییر ورودی، دوباره به آن حالتی که باید می‌رفتیم نرفتیم، بنابراین مخاطرهٔ اساسی وجود دارد.

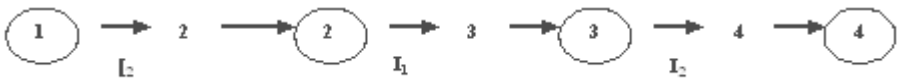
مثـال ۶) مخاطرهٔ اساسی را در انتقال‌های مختلف از جدول روند زیر بررسی کنید.

Y_1Y_2	0	1
00	(00) 1	01
01	11	(01) 2
11	(11) 3	10
10	00	(10) 4

$q_i = 1$

$I_1 = 0 \Rightarrow N(1, 0) = 1$

$I_2 = 1 \Rightarrow N(1, 1) = 2$



در گذر از ۱ به ۲ مخاطرهٔ اساسی وجود دارد. $q_i = 1,\ q_j = 2,\ q_k = 4 \Rightarrow q_j \neq q_k \Rightarrow$

نسخه پیش‌نویس Draft

$q_i = 2$

$I_1 = 1 \Rightarrow N(2, 1) = 2$

$I_2 = 0 \Rightarrow N(2, 0) = 3$

در گذر از ۲ به ۳ مخاطرهٔ اساسی وجود دارد. $q_i = 2, q_j = 3, q_k = 1 \Rightarrow q_j \neq q_k \Rightarrow$

$q_i = 3$

$1 = 0 \Rightarrow N(3, 0) = 3$

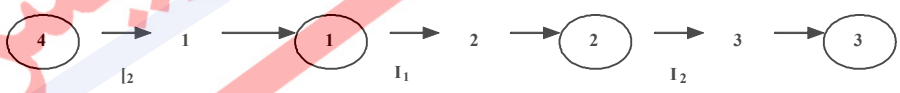
$I_2 = 1 \Rightarrow N(3, 1) = 4$

در گذر از ۳ به ۴ مخاطرهٔ اساسی وجود دارد. $q_i = 3, q_j = 4, q_k = 2 \Rightarrow q_j \neq q_k \Rightarrow$

$q_i = 4$

$I_1 = 1 \Rightarrow N(4, 1) = 4$

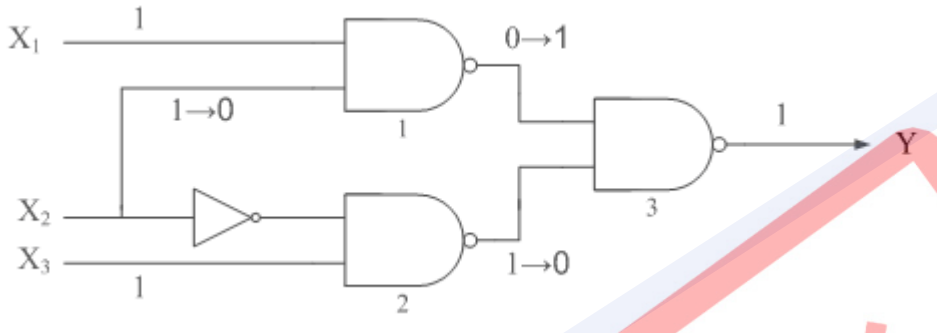
$I_2 = 0 \Rightarrow N(4, 0) = 1$



روش دیگر برای اجتناب از خروجی‌های تصادفی ایستا در مدارهای ترتیبی ناهمگام، پیاده‌سازی مدار با نگهدارهای SR است. اعمال یک سیگنال صفر آنی به ورودی‌های S یا R در یک نگهدار NOR، روی حالت‌های مدار هیچ تأثیری نخواهد گذاشت. بطور مشابه، اعمال یک سیگنال یک آنی به ورودی‌های S و R در یک نگهدار NAND، تأثیری روی حالت نگهدار ندارد. در شکل زیر دیده می‌شود که اگر هر دو ورودی گیت ۳، ۱ شوند، جملهٔ جمع حاصل ضرب‌های دو طبقه که با گیت های NAND پیاده سازی شده ممکن است یک خروجی تصادفی ۱ ایستا داشته باشد و برای یک لحظهٔ آنی، خروجی مدار را از ۱ به

نسخه پیش‌نویس Draft

۰ تغییر دهد. اما اگر گیت ۳ قسمتی از یک نگهدار باشد، سیگنال آنی ۱ ، اثری روی خروجی نخواهد داشت، زیرا ورودی سومی از طرف مکمل شدهٔ نگهدار به گیت مذکور می‌آید که برابر ۰ است و بنابراین خروجی را در مقدار ۱ نگه می‌دارد. برای روشن شدن این مطلب، یک لچ SR ساخته شده از NAND را با توابع بولی شکل ۱۲ برای S وR در نظر بگیرید:



$$S = AB + CD$$
$$R = \overline{A}C$$

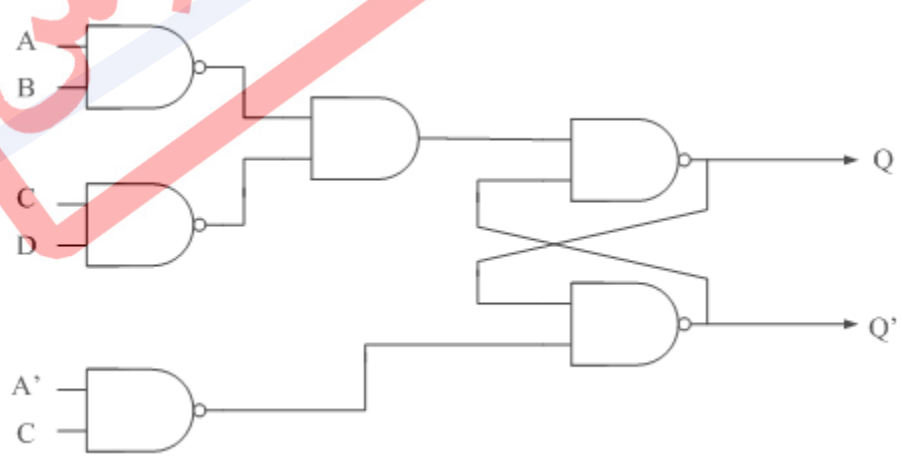
شکل 6-۱۲- توابع بولی لچ SR

چون این یک نگهدار NAND است، باید مقادیر مکمل شده را به ورودی‌ها اعمال نماییم.

$$S = \overline{(AB + CD)} = \overline{(AB)}\,\overline{(CD)}$$
$$R = \overline{(\overline{A}C)}$$

این پیاده‌سازی، در شکل ۱۳ نشان داده شده است. S با دو گیت NAND و یک گیت AND درست شده است. تابع بولی برای خروجی Q عبارت است از:

$$Q = \overline{(\overline{Q}S)} = \overline{[\overline{Q}\,\overline{(AB)}\,\overline{(CD)}]}$$



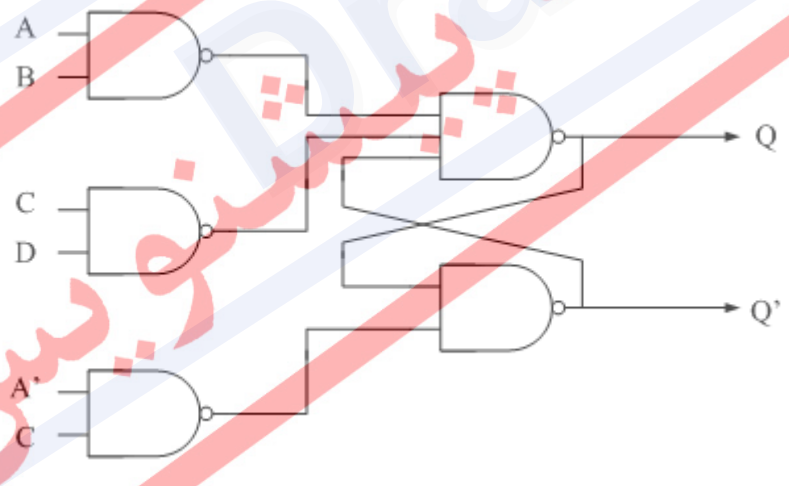
شکل 6-۱۳ - پیاده سازی $Q = \overline{(\overline{Q}S)} = \overline{[\overline{Q}\,\overline{(AB)}\,\overline{(CD)}]}$

نسخه پیش‌نویس Draft

این عبارت در شکل زیر با دو طبقه از گیت‌های NAND ساخته شده است. اگر خروجی Q برابر ۱ باشد، $\overline{Q}$ برابر ۰ است. اگر دو تا از سه ورودی بطور آنی ۱ شوند، گیت NAND مرتبط با خروجی Q ، در ۱ باقی خواهد ماند زیرا $\overline{Q}$ در ۰ باقی می‌ماند.

شکل زیر مداری را نشان می‌دهد که می‌تواند برای ساخت مدارهای ترتیبی نا همگام مورد استفاده قرار گیرد. دو گیت NAND که نگهدار را تشکیل می‌دهند معمولاً دو ورودی دارند. هر چند که اگر توابع S و R ، هنگامی که به صورت جمع حاصلضرب‌ها بیان می‌شوند، شامل دو جملهٔ حاصلضرب یا بیشتر باشند، گیت NAND مربوطه در نگهدار SR دارای سه ورودی یا بیشتر خواهد بود.

لذا این دو جمله در عبارت جمع حاصل ضرب‌های اصلی برای S ، برابر AB وCD هستند و هر یک با یک گیت NAND که خروجی آن به ورودی نگهدار NAND اعمال می‌گردد، پیاده‌سازی می‌شوند. به این ترتیب، هر متغیر حالت به یک مدار دو طبقه از گیت های NAND نیاز دارد. طبقهٔ اول از تعدادی گیت NAND تشکیل شده است که جملات حاصلضرب عبارت بولی اصلی S وR را پیاده سازی می‌کند و طبقهٔ دوم دو ارتباط متقاطع نگهدار SR را با ورودی‌هایی که از خروجی هر یک از گیت‌های NAND طبقهٔ اول می‌آیند، شکل می‌دهد.



شکل 6-14- پیاده سازی $Q=\overline{(\overline{Q}S)}=\overline{[\overline{Q}\overline{(AB)}\overline{(CD)}]}$ با دو طبقه از گیت‌های NAND

نسخه پیش‌نویس Draft

۶-۶ مثال کامل از طراحی و سنتز مدار ترتیبی ناهمگام

در این قسمت برای کامل شدن مطلب، یک مثال کامل از طراحی یک مدار ترتیبی ناهمگام آورده می‌شود. در ابتدا، قدم‌های طراحی و سنتز را دوباره مرور می‌کنیم.

۱. تبدیل توصیف نوشتاری به جدول روند
۲. ساده‌سازی
۳. تخصیص حالت بدون مسابقه
۴. انتساب خروجی به حالت‌های ناپایدار
۵. تعیین نوع فیلیپ-فلاپ و تعداد فیلیپ-فلاپ‌های مورد نیاز
۶. جدول تحریک فیلیپ-فلاپ‌ها
۷. جدول کارنو برای فیلیپ-فلاپ‌ها و جدول کارنوی خروجی
۸. آزمون مخاطره‌ها و از بین بردن آنها
۹. بدست آوردن عبارات نهایی
۱۰. شبیه‌سازی مدار

شرح مسئله:

طراحی مدار داخلی یک فیلیپ-فلاپ T. (فیلیپ-فلاپ T خود یک قطعه همگام است ولی داخل آن به صورت ناهمگام ساخته شده است). نحوهٔ عملکرد این فیلیپ-فلاپ به این صورت است که مدار شامل دو ورودی C و T و یک خروجی Q است. اگر T=1 باشد و پالس ساعت از یک به صفر تغییر کند، مقدار خروجی مکمل می‌شود (تریگر در لبه منفی)، در غیر این صورت تحت هر شرایط دیگری، خروجی ثابت می‌ماند.

نسخه پیش‌نویس Draft

جدول توصیف :

جدول 6-۶ – جدول توصیف

حالت	ورودی T	ورودی C	خروجی Q	توضیحات
a	1	1	0	خروجی اولیه صفر
b	1	0	1	بعد از حالت a
c	1	1	1	خروجی اولیه یک
d	1	0	0	بعد از حالت c
e	0	0	0	بعد از d و f
f	0	1	0	بعد از حالت e و a
g	0	0	1	بعد از حالت b و h
h	0	1	1	بعد از حالت q و c

مرحلهٔ ۱: تبدیل توصیف نوشتاری به جدول روند

جدول 6-۷ – جدول روند بدست آمده از جدول توصیف ۶

	TC 00	01	11	10
a	-	f, -	a, 0	b, -
b	g, -	-	c, -	b, 1
c	-	h, -	c, 1	d, -
d	e, -	-	a, -	d, 0
e	e, 0	f, -	-	d, -
f	e, 1	f, 0	a, -	-
g	g, 1	h, -	-	b, -
h	g, -	h, 1	c, -	-

نسخه پیش‌نویس
Draft

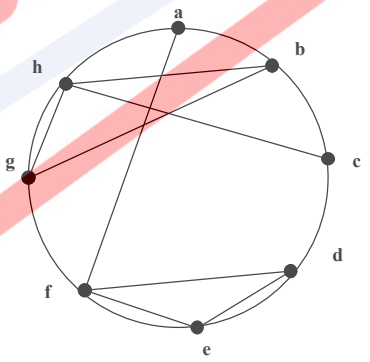
مرحلهٔ ۲: ساده‌سازی

	a	b	c	d	e	f	g
b	a,c						
c		b,d					
d	b,d		a,c				
e	b,d	e,g b,d	f,h	✓			
f	✓	e,g a,c	f,h a,c	✓	✓		
g	f,h	✓	b,d	e,g h,d	✓	e,g f,h	
h	f,h a,c	✓	✓	e,g a,c	e,g f,h		✓

مجموعه زوج‌های سازگاری:

{ (a, f) و (b, g) و (b, h) و (c, h) و (d, e) و (d, f) و (e, f) و (g, h) }

گراف ادغام:



MC´S= { (f, e, d), (b, h, g), (c, h), (a, f) }

Min.C´S = { (a, b, c, d), (c, e, g), (f, g), (h, f) }

U= min {8, 4} = 4

L= max {4, 3, 2, 2} = 4

$4 \le K \le 4$

نسخه پیش‌نویس Pratt

جدول 6-۸ – جدول روند ساده‌سازی شده

	00	01	11	10
a, f	e, -	f, 0	a, 0	b, -
b, g, h	g, 1	h, 1	c, -	b, 1
c, h	g, -	h, 1	c, 1	d, 1
d, e, f	c, 0	f, 0	a, -	d, 0

	00	01	11	10
a	d, -	a, 0	a, 0	b, -
b	b, 1	b, 1	c, -	b, 1
c	b, -	c, 1	c, 1	d, -
d	d, 0	d, 0	a, -	d, 0

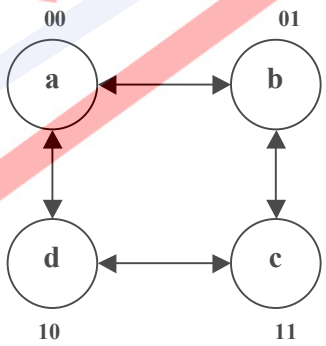
مرحلهٔ ۳ : تخصیص حالت بدون مسابقه

جدول 6-۹ – جدول روند ساده‌سازی شده

	00	01	11	10
a = 00	10	00	00	01
b = 01	01	01	11	01
c = 11	01	11	11	10
d = 10	10	10	00	10

تعیین حالت‌های مجاور و رسم گراف مجاورت:

حالت	حالتهای مجاور
a	d, b
B	c, a
C	b, d
D	a, c



مرحلهٔ ۴ : انتساب خروجی به حالت‌های ناپایدار

الگوریتم انتساب خروجی به حالت‌های ناپایدار :

i : خروجی حالت مبدأ

j : خروجی حالت مقصد

نسخه پیش‌نویس Draft

k : خروجی حالت ناپایدار میانی

$$a, i \longrightarrow b, k \longrightarrow b, j$$

if $(i = j)$

$k = i = j$

if $(i \neq j)$

$(k = i)$ or $(k = j)$

مرحلهٔ ۵ : تعیین نوع فیلیپ-فلاپ و تعداد فیلیپ-فلاپ‌های مورد نیاز

در این مثال، از نگهدار SR استفاده می‌کنیم.

تعداد مورد نیاز :

بنابراین به دو فیلیپ-فلاپ نیاز

$$\left[\mathrm{Log}_2^4\right] = 2$$

داریم.

مراحل ۶و۷: جدول تحریک و کارنوی فیلیپ-فلاپ‌ها

Y_1Y_2 \ TC	00	01	11	10
00	1	0	0	0
01	0	0	1	0
11	0	×	×	×
10	×	×	0	×

$S_1 = Y_2TC + Y_2'T'C'$

Y_1Y_2 \ TC	00	01	11	10
00	0	×	×	×
01	×	×	0	×
11	1	0	0	0
10	0	0	1	0

$R_1 = Y_2T'C' + Y_2'TC$

نسخه پیش‌نویس Draft

Y_1Y_2 \ TC	00	01	11	10
00	0	0	0	1
01	×	×	×	×
11	×	×	×	0
10	0	0	0	0

$S_2 = Y'TC'$

Y_1Y_2 \ TC	00	01	11	10
00	×	×	×	0
01	0	0	0	0
11	0	0	0	1
10	×	×	×	×

$R_2 = Y_1T'C$

جدول کارنو برای خروجی :

Y_1Y_2 \ TC	00	01	11	10
00	0	0	0	×
01	1	1	1	1
11	1	1	1	×
10	0	0	0	0

نسخه پیش‌نویس
Draft

۶-۷ تمرین

۱- با توجه به جدول تحریک زیر:

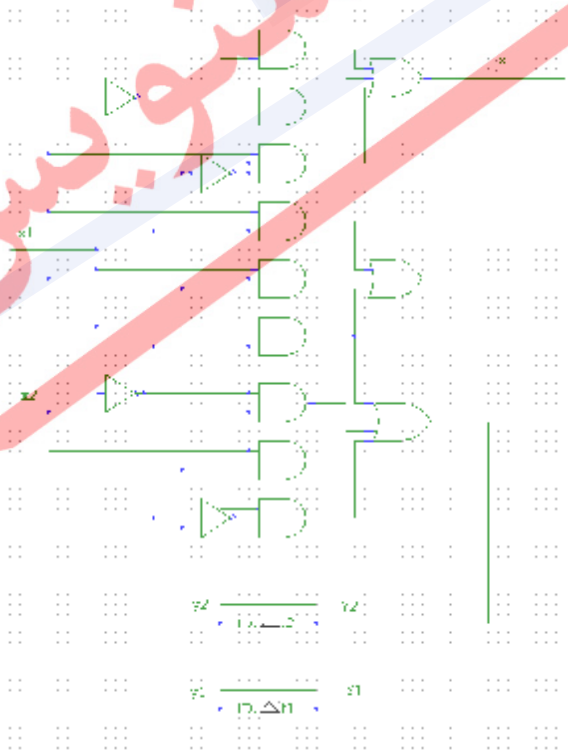
الف) تمامی حالات مسابقه را در این جدول بیابید.

ب) آیا این مسابقه ها بحرانی هستند یا نه؟

پ)آیا در این جدول دور وجود دارد؟

y1y2 \ x1x2	00	01	11	10
00	00	00	11	01
01	11	01	10	01
11	00	00	10	10
10	11	10	11	10

۲-مدار شکل زیر را تحلیل کنید و تعیین کنیدکه آیا این مدار داراي مسابقه بحراني است. در این صورت، یك دیاگرام زماني بکشید تا تاثیري که مسابقه مي تواند بر روي پاسخ مدار داشته باشد را نشان دهد.



نسخه پیش‌نویس

۳- برای جدول جریان کاهش یافته زیر تخصیص حالتی را تعیین کنید که فاقد مسابقه بحرانی باشد.

جدول تحریک مربوطه را بسازید.

	x = 0	x = 1
a	a/0	d/-
b	b/1	c/-
c	a/0	c/0
d	b/0	d/1

۴- با توجه به جدول روند کاهش یافته زیر، تخصیص حالت دیگری برای این مدار ناهمگام ترتیبی که فاقد مسابقه بحرانی باشد. پیاده سازی دو سطحی با گیت NOR با استفاده از تاخیر های inertial انجام دهید.

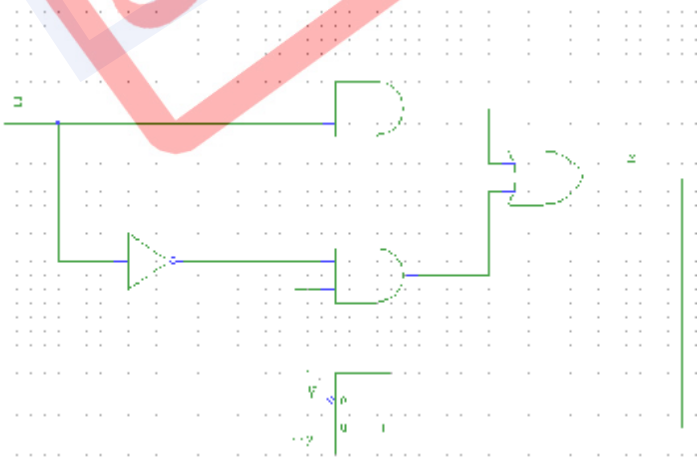
	x_1x_2 = ۰۰	۰۱	۱۱	۱۰
a	a/0	b/-	a/1	b/-
b	a/-	b/0	c/-	b/0
c	a/-	c/1	c/0	b/-

۵- مدار ناهمگام ترتیبی شکل زیر، را تحلیل کنید. این مدار ورودی پالسی x دارد. موارد زیر را بیابید.

الف) یک دیاگرام زمانی در صورتی که حالت شروع برابر $y^0 = 0$ و $x = 001011000$ بـا شد .

ب) جدول حالت.

پ) دیاگرام حالت.



نسخه پیش‌نویس Draft

در راه حل شما ممکن است عرض پالس ورودی x را برابر زمان تاخیر فلیپ فلاپ T در نظر بگیرید. توضیح دهید اگر عرض پالس ورودی کمی بزرگتر از تاخیر فلیپ فلاپ باشد چه تاثیری بر عملکرد این مدار ترتیبی دارد. نتیجه گیری خود را بر روی دیاگرام زمانی قسمت الف نشان دهید.

نسخه پیشنویس
Draft

فصل ۷

ماشین های بدون از دست دادن اطلاعات

نسخه پیشنویس
Draft

۷-۱ مقدمه

یک مبدل بدون از دست دادن اطلاعات[1]، مبدلی است که تنها با دانستن دنباله خروجی، دنباله ورودی متناظر را تشخیص می دهد. کاربرد چنین مبدل هایی در آماده سازی دیتا جهت انتقال روی کانال هایی است که درجه امنیت بالا داشته، و یا در معرض نویز قرار دارند.

در واقع ماشین های بدون از دست دادن اطلاعات دسته مهمی از ماشین های با تعداد حالات متناهی هستند که دنباله ورودی ها را به دنباله خروجی ها تبدیل می کنند، آن طور که پس از آزمایشی با طول متناهی روی ماشین، دنباله ورودی با دانستن دنباله خروجی متناظر، حالت های اولیه و نهایی، و همچنین یک سری از ویژگی های ماشین در تبدیل ورودی به خروجی، قابل تشخیص است. یافتن معکوس چنین ماشین هایی نیز مورد توجه قرار دارد.

۷-۲ مدارهای ترکیبی بدون از دست دادن اطلاعات

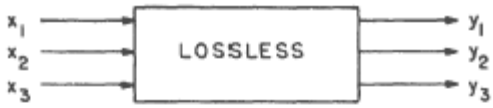
مدارهای ترکیبی، ماشین هایی با تعداد حالت های متناهی هستند که فقط یک حالت داشته و به این ترتیب فاقد حافظه می باشند. چنین مدارهایی در واقع اجزای اصلی مدارهای دارای حافظه به شمار می روند. بنابراین در اینجا اشاره کوتاهی به آن ها می کنیم:

هر مدار ترکیبی به ازای یک ورودی (یا یک مجموعه از ورودی ها)، یک خروجی(یا یک مجموعه خروجی) یکتا می دهد. حال برای یک مدار ترکیبی بدون از دست دادن اطلاعات هر خروجی نیز، یک ورودی یکتا را تعیین می کند. به عبارت دیگر در چنین مداری هیچ دو ورودی، خروجی یکسان تولید نمی کنند. تناظر یک به یک در جدول درستی چنین مداراتی به راحتی دیده می شود. در یک مدار با n ورودی و n خروجی، شرط lossless بودن ضمناً بر قابل حل بودن معادلاتی که چگونگی وابستگی ورودی ها (x) به خروجی ها (y) را نشان می دهند، دلالت دارد.

به عنوان مثال مداری را که در شکل ۱ توصیف شده است، در نظر بگیرید. همانطور که ملاحظه می گردد، معادلات شکل ۱-b به علت داشتن عبارتی نظیر x_1x_3 غیر خطی می باشند. شرط lossless بودن برقرار است، چرا که همان گونه که در جدول درستی مدار در شکل ۱-c مشاهده می شود، ردیف های ستون سمت راست (خروجی) جایگشتی از ردیف های ستون سمت چپ (ورودی) است. در شکل های ۱-d و ۱-e جدول درستی مدار معکوس و همچنین معادلات xها بر حسب yها نشان داده شده است.

[1] Information-lossless

نسخه پیش نویس Draft



(a)

$$\begin{cases} y_1 = 1 \oplus x_1 \oplus x_3 \oplus x_1 x_2 \oplus x_1 x_3 \\ y_2 = 1 \oplus x_1 \oplus x_2 \oplus x_3 \\ y_3 = 1 \oplus x_1 \oplus x_2 \oplus x_1 x_2 \oplus x_2 x_3 \end{cases}$$

(b)

x_1	x_2	x_3	y_1	y_2	y_3
0	0	0	1	1	1
0	0	1	0	0	1
0	1	0	1	0	0
0	1	1	0	1	1
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	1	1	0
1	1	1	1	0	1

(c)

y_1	y_2	y_3	x_1	x_2	x_3
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	1	0	1
0	1	1	0	1	1
1	0	0	0	1	0
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	0	0	0

(d)

$$\begin{cases} x_1 = 1 \oplus y_1 \oplus y_3 \oplus y_1 y_2 \\ x_2 = y_1 \oplus y_2 y_3 \\ x_3 = y_2 \oplus y_3 \oplus y_1 y_2 \oplus y_2 y_3 \end{cases}$$

(e)

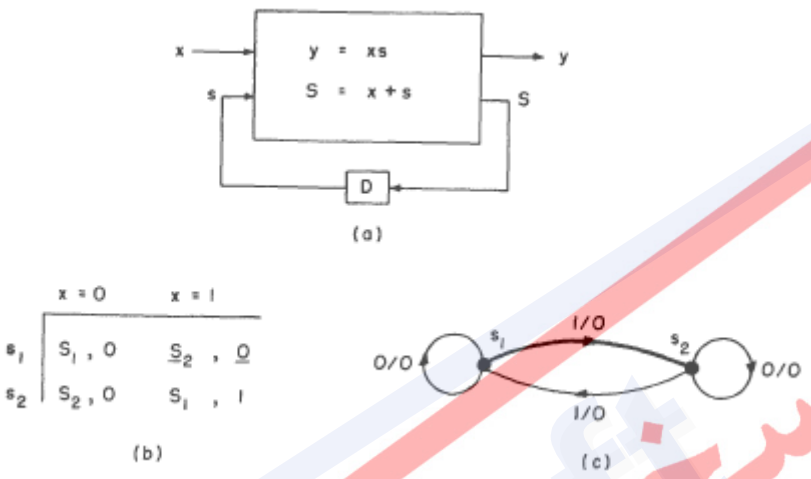
شکل 7-۱ – مثال مدار ترکیبی بدون از دست دادن اطلاعات

نسخه پیش نویس Draft

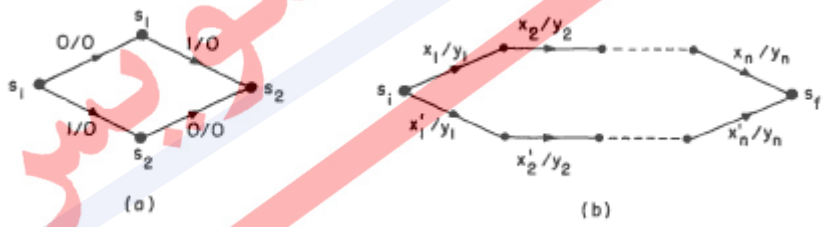
۷-۳ چه وقت می گوییم اطلاعات از دست رفته است؟

از دست رفتن یا گم شدن اطلاعات به این معنی است که تغییر حالت مدار طوری اتفاق بیفتد که اطلاعات راجع به ورودی های گذشته از دست برود. برای مثال درباره مدار شکل ۲ اگر ناظر خروجی بداند که حالت اولیه مدار S_1 بوده و سپس دو صفر را در خروجی ببیند، برای ناظر اطلاعات از دست رفته است، حتی اگر بداند مدار در نهایت به حالت S_2 رفته است، زیرا که دنباله ورودی می تواند هر دو دنباله ۰ ۱ و یا ۱ ۰ باشد.

در چنین شرایطی آنالیز بیشتر دنباله خروجی ها پس از حالت اولیه و قبل از رسیدن به حالت نهایی هیچ کمکی به یافتن دنباله ورودی نخواهد کرد. در شکل ۳ چگونگی از دست رفتن اطلاعات و غیر قابل تشخیص بودن مسیر و در نتیجه دنباله ورودی در مدار شکل ۲ نشان داده شده است.



شکل 7-۲ – توصیف یک مدار ترتیبی



شکل 7-۳ – وضعیت از دست رفتن اطلاعات

نسخه پیش‌نویس Draft

۷-۴ تعریف مدارهای بدون از دست دادن اطلاعات با تعداد حالات متناهی[1]

با توجه به تعاریف بالا، واضح است که از دست دادن اطلاعات[2] در مدارهایی اتفاق می افتد، که دو یا چند دنباله ورودی متفاوت را، به یک دنباله خروجی نگاشت کند.

دقیقتر بگوییم: یک مدار ترتیبی، lossless تعریف می شود، اگر و فقط اگر هیچ دو حالت S_i و S_f (نه الزاماً متفاوت)، و هیچ دو دنباله ورودی متفاوت با طول یکسان $\{x\}$ و $\{x'\}$ و دنباله خروجی $\{y\}$ وجود نداشته باشند، به طوری که مدار با هر دو $\{x\}$ و $\{x'\}$ از S_i به S_f منتهی شده و $\{y\}$ را خروجی بدهد. به عبارت دیگر می توان گفت یک مدار lossless است، اگر و فقط اگر برای یک تجربه طولانی غیرمعین، که در آن حالات اولیه و نهایی و دنباله های خروجی شناخته شده باشند، با داشتن جدول یا دیاگرام حالت مدار بتوان دنباله ورودی را تشخیص داد.

۷-۴-۱ کلاس اول مدارهای بدون از دست دادن اطلاعات[3]

روندهایی را که برای تشخیص lossless بودن یا نبودن، مورد استفاده قرار می گیرند، می توان به چند دسته تقسیم کرد. جدول روند شکل۴-a و جدول به دست آمده در شکل ۴-b را در نظر بگیرید. سطر اول از جدول به دست آمده نشان می دهد که اگر حالت اولیه مدار s_1 باشد، حالت بعدی مدار بسته به اینکه خروجی ۰ یا ۱ باشد، به ترتیب s_4 و یا s_3 است. باقی سطرهای جدول نیز به همین شکل تفسیر می شوند. در مثالی که در اینجا مطرح شده است، به وضوح می توان دید که هر دو گذر از هر حالت اولیه، به خروجی های متفاوتی منجر می شوند. بنابراین آنچه تحت عنوان دنباله های موازی در شکل ۵ دیده شد، وجود ندارد. برای چنین مدارهایی که IL-1 (کلاس اول مدارهای بدون از دست دادن اطلاعات) خوانده می شوند، مدارهای معکوسی می توان یافت، که با دریافت دنباله ی خروجی، دنباله ورودی را تشخیص دهند. مدار معکوس IL-1، تعداد حالات مساوی با تعداد حالات مدار اصلی خواهد داشت.

---

[1] Information-Lossless Finite-State Circuits

[2] information loss

[3] Class I Information-Lossless Circuits

نسخه پیش نویس

	x = 0	x = 1
s_1	$S_3, 1$	$S_4, 0$
s_2	$S_4, 0$	$S_1, 1$
s_3	$S_4, 1$	$S_2, 0$
s_4	$S_3, 0$	$S_2, 1$

(a)

	y = 0	y = 1
s_1	S_4	S_3
s_2	S_4	S_1
s_3	S_2	S_4
s_4	S_3	S_2

(b)

شکل 7-۴ – تست یک مدار کلاس ۱: (a)جدول روند (b)جدول تست

- لم ۱ : یک مدار ترتیبی IL-1 است، اگر و فقط اگر $z(q_i,I_j) \neq z(q_i,I_k)$ برای همه $I_j \neq I_k$.

لازم به ذکر است: $Z(q,I)$ به معنی خروجی است که در حالت q و با ورودی I حاصل می شود.

بنابراین در جدول حالت چنین مداری، خروجی های یکسان در یک سطر جدول وجود ندارند.

مثال ۱) جدول حالت مداری به شکل زیر است. دنباله خروجی، ۱ ۰ ۱ دیده شده و حالت اولیه مدار هم ۱ است.

جدول 7-۱ – جدول حالت مربوط به مثال ۱

حالت اولیه	ورودی X: ۰	۱
۱	۱و۰	۲و۱
۲	۱و۱	۴و۰
۳	۴و۰	۳و۱
۴	۲و۰	۴و۱

به وسیله لم ۱ تشخیص داده می‌شود، که مدار بالا IL-1 است. جدول تست را به شکلی که توضیح داده شد تشکیل می دهیم:

نسخه پیش‌نویس Draft

جدول 7-۲ – جدول تست مربوط به مثال ۱

حالت اولیه	ورودی Y ۰	۱
۱	۱	۲
۲	۴	۱
۳	۴	۳
۴	۲	۴

دنباله ورودی از روی دنباله خروجی و حالت اولیه به دست می آید، که عبارت است از: ۱ ۱ ۱ .

خروجی

حالت

ورودی

بنابراین در ماشین های IL-1 با داشتن حالت اولیه مدار و دنباله خروجی، دنباله ورودی قابل تشخیص خواهد بود.

نسخه پیش‌نویس Draft

۷-۴-۲ کلاس دوم مدارهای بدون از دست دادن اطلاعات[1]

مثال دسته ی دیگری از مدارهای بدون از دست دادن اطلاعات در شکل ۵ دیده می شود. چهار سطر بالای جدول شکل ۵-b به شکلی مشابه مثال قبل به دست می آید، با این تفاوت که هم اکنون الزاماً نمی توان تنها با دانستن خروجی، ورودی متناظر با آن خروجی را تشخیص داد. برای مثال سطر دوم جدول را می توان به این صورت تفسیر کرد: اگر حالت اولیه مدار s_2 باشد، آن گاه خروجی $y=0$ اجتناب ناپذیر است، و در نتیجه نمی توان مطمئن بود که حالت بعدی s_1 بوده است یا s_3 و به همین ترتیب نمی توان مشخص کرد که $x=0$ یا $x=1$.

چهار سطر ابتدایی جدول جدید، نشان می دهد که میان حالات s_1 و s_3، و همچنین بین s_1 و s_4 تردید وجود دارد. دو سمبل s_{13} و s_{14} به عنوان انتخابگر برای سطرهایی که به چهار سطر اول اضافه می شوند، وارد جدول شده اند. برای مثال سمبلی که در ستون $y=1$ و سطر s_{13} وجود دارد، s_{134} است چرا که آنچه که در سطرهای s_1 و s_3 دیده می شود، s_3 و s_{14} است. این مدخل جدید جدول به ما می گوید که اگر شک داریم که مدار در حالت s_1 یا s_3 قرار دارد و خروجی مدار را $y=1$ مشاهده کرده ایم، آن گاه شک جدید ما باید بین s_1، s_3، و s_4 باشد. روند ایجاد سطرهای جدید تا جایی که مورد نیاز است ادامه می یابد، تا وقتی که دیگر نیازی به ایجاد سطر جدیدی نبوده، و جدول کامل می شود. اگر در روند اضافه کردن اندیس از سطرهای عنصری[2] برای یافتن اندیس سطرهای ترکیبی[3]، شرایطی پیش نیاید که یک اندیس در دو سطر عنصری تکرار شود، آن گاه مداری که تست شده مدار بدون از دست دادن اطلاعات است. مثال ما مداری از این نوع است.

در شکل ۵-a مستقیماً دیده می شود که جدول روند یک مدار loss-less را توصیف می کند، چرا که تنها دو گذر به هر حالت ختم شده و در هر گذر نیز خروجی متفاوت است. چنین ماشین هایی را IL-2 (کلاس دوم مدارهای بدون از دست دادن اطلاعات) می نامیم. بنابراین امکان دنباله های موازی یعنی آن چیزی که در شکل ۳-b دیده شد، وجود ندارد. به علاوه دانستن حالت نهایی مدار و آخرین سمبل خروجی، برای یافتن حالت های بعدی تا حالت نهایی کافی است.

---

[1] Class II Information-Lossless Circuits
[2] component
[3] composite

بنابراین در یک آزمایش متناهی روی یک مدار IL-2 ، با دانستن حالت نهایی مدار و خروجی، می توان دنباله ورودی را تشخیص داد. یادآوری می کنیم که ماشین های IL-1 با داشتن حالت اولیه و دنباله ورودی، دنباله خروجی را به دست می دادند.

	x = 0	x = 1
s_1	S_2 , 0	S_3 , 1
s_2	S_1 , 0	S_3 , 0
s_3	S_4 , 1	S_1 , 1
s_4	S_2 , 1	S_4 , 0

(a)

	y = 0	y = 1
s_1	S_2	S_3
s_2	S_{13}	—
s_3	—	S_{14}
s_4	S_4	S_2
s_{13}	S_2	S_{134}
s_{14}	S_{24}	S_{23}
s_{23}	S_{13}	S_{14}
s_{24}	S_{134}	S_2
s_{134}	S_{24}	S_{1234}
s_{1234}	S_{1234}	S_{1234}

(b)

شکل 7-۵ – تست یک مدار کلاس ۲: (a)جدول روند (b)جدول تست

- لم ۲ : یک مدار ترتیبی IL-2 است، اگر هیچ زوج (حالت بعدی ، خروجی) بیش از یک بار در جدول حالت مدار نیامده باشد.

مثال ۲) جدول حالت مداری به شکل زیر است. دنباله خروجی، ۱ ۱ ۰ دیده شده و حالت نهایی مدار هم ۴ است.

جدول 7-۳ – جدول حالت مربوط به مثال ۲

حالت اولیه	ورودی X: ۰	۱
۱	۱و۰	۲و۰
۲	۲و۱	۳و۱
۳	۴و۱	۱و۱

نسخه پیش‌نویس Draft

۴	۴و۰	۳و۰

به وسیله لم ۲ تشخیص داده می شود، که مدار بالا IL-2 است. جدول تست را به شکلی که توضیح داده شد تشکیل می دهیم:

جدول 7-۴ – جدول تست مربوط به مثال ۲

	ورودی Y	
حالت اولیه	۰	۱
۱	۱۲	–
۲	–	۲۳
۳	–	۱۴
۴	۳۴	–

در این مثال نیازی به تکمیل جدول و تشکیل سطرهای چند حالته نیست. همان طور که ملاحظه می گردد، هر حالت فقط یک بار در هر ستون (یعنی به ازای خروجی ۰ یا ۱) آمده، و به همین علت به سادگی و با داشتن حالت نهایی و دنباله خروجی، دنباله ورودی که در اینجا ۰ ۱ ۱ است، به دست می آید:

خروجی : ۰ ۱ ۱

حالت : ۱ ۲ ۳ ۴

اولیه

ورودی:

- لم ۳ : اگر زوج (حالت بعدی ، خروجی) بیش از یک بار در جدول حالت مدار آمده باشد، آن گاه ماشین IL-2 است، فقط اگر این زوج تنها در یک ستون از جدول حالت تکرار شده باشد.

نسخه پیش نویس Draft

مثال ۳) جدول حالت مداری به شکل زیر است. دنباله خروجی، ۰ ۰ ۰ ۰ دیده شده و حالت نهایی مدار نیز ۳ است

جدول 7-۵ – جدول حالت مربوط به مثال ۳.

حالت اولیه	ورودی X: ۰	ورودی X: ۱
۱	**۳و۰**	۱و۰
۲	**۳و۰**	۲و۰
۳	۴و۱	۱و۱
۴	۴و۱	۲و۱

به وسیله لم ۳ تشخیص داده می شود، که مدار بالا IL-2 است. جدول تست را به شکلی که توضیح داده شد تشکیل می دهیم:

نسخه پیش‌نویس Draft

جدول 7-۶ – جدول تست مربوط به مثال ۳

حالت اولیه	ورودی Y ۰	۱
۱	۱۳	–
۲	۲۳	–
۳	–	۱۴
۴	–	۲۴
۱۳	۱۳	۱۴
۲۳	۲۳	۱۴
۱۴	۱۳	۲۴
۲۴	۲۳	۲۴

به این ترتیب از روی جدول به شکل زیر ورودی پیدا می شود:

:

خروجی

:حا

لت‌اولیه ۲ ۲ ۲ ۲

ورودی:

نسخه پیش‌نویس Draft

۷-۴-۳ کلاس عمومی مدارهای بدون از دست دادن اطلاعات[1]

مدارهای بسیاری وجود دارند که نه به طور خالص از کلاس اول هستند و نه می توانند در کلاس دوم قرار بگیرند. برای همه چنین مدارهایی، تستی که قبلاً شرح داده شد، قابل اعمال است.

- لم ۴ : یک مدار ترتیبی GIL است، اگر و فقط اگر هیچ حالت اولیه ای دو مدخل یکسان (خروجی، حالت بعدی) نداشته باشد، یعنی در یک سطر زوج تکراری نداشته باشیم، و همچنین هیچ دو حالتی که زوج (خروجی، حالت بعدی) یکسان دارند، در جدول تست با هم ترکیب نشوند

مثال ۴)

جدول 7-۷ – جدول حالت مربوط به مثال ۴

حالت اولیه	ورودی X: ۰	ورودی X: ۱
۱	3,0	4,1
۲	4,0	3,1
۳	1,0	2,0
۴	3,1	4,1

ملاحظه می شود که زوج (۱و۴) به ازای حالت های ۱ و ۴، و همچنین زوج (۱و۳) به ازای حالت های ۲ و ۴ در جدول حالت تکرار شده اند.

برای تشخیص GIL بودن مدار، ابتدا می بینیم که هیچ زوج تکراری در یک سطر وجود ندارد. از طرفی جدول تست را به شیوه گذشته تشکیل داده و بررسی می کنیم:

[1] General Information-Lossless Circuits

نسخه پیش نویس

جدول 7-۸ – جدول تست مربوط به مثال ۴

حالت اولیه	ورودی Y ۰	۱
۱	۳	۴
۲	۴	۳
۳	۱۲	–
۴	–	۳۴
۱۲	۳۴	۳۴
۳۴	۱۲	۳۴

همان طور که در بالا مشاهده می شود، در جدول تست حالت های ۱ و ۴، و همچنین ۲ و ۴ با هم ترکیب نشده اند، بنابراین مدار GIL است چون هر دوشرط را دارد.

حال فرض می کنیم حالت اولیه مدار ۱ و حالت نهایی آن ۲ باشد و خروجی ۰ ۱ ۰ ۰ هم دیده شده باشد، می خواهیم ورودی را به دست آوریم:

روش کار به این شرح است که عمل اسکن را یک بار رو به جلو و یک بار رو به عقب انجام می دهیم. در اینجا در اسکن اول بعضی از ورودی ها مجهول می مانند، و در اسکن دوم معین می شوند:

خ

روجی

حالت ۱ ۳ ۱۲ ۳۴ ۳۴ ۱۲

ورودی اسکن رو

به جلو

ح

نسخه پیش نویس Draft

الت

ورودی اسکن رو

به عقب

نسخه پیشنویس
Draft

مثال ۵) جدول حالت مداری به صورت زیر است، بررسی می کنیم که آیا GIL هست یا خیر.

جدول 7-۹ – جدول حالت مربوط به مثال ۵

حالت‌اولیه	ورودی X: ۰	ورودی X: ۱
۱	2,0	3,0
۲	4,0	2,1
۳	3,1	2,0
۴	1,1	3,0

ملاحظه می شود که زوج (۰و۳) به ازای حالت های ۱ و ۴، و همچنین زوج (۰و۲) به ازای حالت های ۳ و ۱ در جدول حالت تکرار شده اند.

برای تشخیص GIL بودن مدار، ابتدا می بینیم که هیچ زوج تکراری در یک سطر وجود ندارد. از طرفی جدول تست را به شیوه گذشته تشکیل داده و بررسی می کنیم آیا ترکیب ۱ و ۴، یا ۱ و ۳ در جدول تست دیده می شود، یا نه.

جدول 7-۱۰ – جدول تست مربوط به مثال ۵

حالت‌اولیه	ورودی X: ۰	ورودی X: ۱
۱	۲۳	-
۲	۴	۲
۳	۲	۳
۴	۳	۱
۲۳	۲۴	۲۳
۲۴	۳۴	۱۲
۳۴	۲۳	۱۳
...	...	...

با دیدن ۱۳ دیگر ادامه نداده و نتیجه می گیریم که ماشین GIL نیست.

نسخه پیش نویس Draft

برای مثال اگر حالت اولیه را ۱ و حالت نهایی را ۲ بگیریم و خروجی ۰ ۰ ۰ ۱ ۰ مشاهده گردد، ورودی قابل تشخیص نیست.

نسخه پیشنویس
Draft

فصل ۸

طراحی آسنکرون و روش سنتز مارتین

نسخه پیش‌نویس
Draft

۸-۱ مقدمه

وقتی طراحی و تست مدارات VLSI به دلیل درگیری با مسأله توزیع کلاک، تغییرات پارامترهای فیزیکی، توان مصرفی و پیچیدگی طراحی به بن‌بست رسیده بود، گروهی از محققان در سراسر جهان در حال توسعه دادن تکنیکی بودند که به طور قابل توجهی این مشکلات را با حذف کلاک کاهش می‌داد.

اختراع مدارات آسنکرون به دهه ۵۰ میلادی بازمی‌گردد. با این وجود به علت وجود مشکلات مربوط به مخاطره[1]، این روش طراحی, کاربرد و معروفیت لازم را کسب نکرده بود. در سالهای اخیر روشهای مختلفی برای تولید مدارات آسنکرون عملی با فرضیه‌های تأخیرهای زمانی متفاوت مطرح شده است. روش سنتز مارتین معروفترین و قویترین این روشهاست. در اواسط دهه هشتاد در دانشگاه Caltech گروهی به سرپرستی دکتر مارتین، یک مدار پیچیده را به صورت آسنکرون طراحی نمودند. در سال ۱۹۸۸، اولین میکروپروسسور آسنکرون به صورت کامل توسط این گروه طراحی شد.

این بخش به توصیف طراحی و تولید مدارهای آسنکرون به روش Quasi Delay-Insensitive (QDI) می‌پردازد. در این روش بطور سیستماتیک یک توصیف سطح بالا را، از طریق یک سری مراحل تبدیل با حفظ معنا، به شبکه‌ای از عناصر مدار تبدیل می‌کند. با توجه به قوانین به‌کار برده شده در این تبدیل، مدارهای ساخته شده به این روش بدون Hazard می‌باشند، و صرفنظر از تأخیر موجود در عناصر مختلف و سیم‌ها، درست عمل می‌کند و تنها پیش فرضی که در مورد تأخیرها در نظر گرفته می‌شود این است که اختلاف تأخیر مربوط به شاخه‌های مختلف یک انشعاب (مسأله معروف به Isochronic Fork) ناچیز در نظر گرفته می‌شود. با توجه به این فرضیه در مورد تأخیرها، مدارهای QDI بسیار توانمند می‌باشند و در عین حال به راحتی قابل تست می‌باشند و تحمل آنها در مقابل تغییرات پارامترهای فیزیکی خیلی بهتر می‌باشد.

---

[1] Hazard

www.jozve.org

۸-۲ طراحی مدارهای آسنکرون Quasi Delay-Insensitive

به مدارهای آسنکرونی که ما در اینجا معرفی می‌کنیم، Quasi delay-Insensitive و یا QDI گفته می‌شود. مدارهای QDI از هیچ فرض و یا اطلاعاتی راجع به تأخیر در سیمها استفاده نمی‌کند به جز فقط کمی استثناء . این استثناء مربوط به شاخه‌های مختلف یک انشعاب می‌باشد که isochronic forks نامیده می‌شود و به این معنی است که تأخیرها در شاخه‌های مختلف یک انشعاب همسان فرض می‌شود.

ما به منظور برطرف نمودن تمام فرضیات در مورد زمانبندی، شامل کلاک، تمام اطلاعات راجع به اعتبار و وجود یک سیگنال را در خود سیگنال کد می‌کنیم. یک بیت دیتا می‌تواند توسط یک کانال دوتایی که شامل دو سیم اطلاعات (یکی برای نشان دادن '0' و دیگری نشان دهنده '1') و یک سیم برای درخواست/تأیید می‌باشد پیاده‌سازی شود. با این کد نمودن ما از چهار فاز برای سنکرون نمودن ارتباطات استفاده می‌کنیم. برای ارتباطات مقدار بیشتری اطلاعات، معمولاً کانال دو تایی را به کانال N تایی گسترش می‌دهیم که شامل یک سیم برای درخواست/تأیید و N خط دیتا که به روش one-hot عمل می‌کند.

به عنوان مثال به عملیات دریافت L?x توجه نمایید (برای شرح مختصری بر علامت‌گذاری CHP به بخش CHP مراجعه نمایید)، که L نشان‌دهنده کانال ورودی بولی با دو خط دیتای ورودی L:0 و L:1 و یک خط خروجی تأیید L:a می‌باشد. یکی از چندین حالت بسط دستدهی[1] چهار فازه ممکن برای عملیات دریافت به صورت زیر می‌باشد:

$$receive \equiv [L.0 \longrightarrow x\downarrow \;[]\; L.1 \longrightarrow x\uparrow];\; L.a\uparrow;\; [\neg L.0 \wedge \neg L.1];\; L.a\downarrow$$

[1] handshaking expansions (HSE)

www.jozve.org

به طور مشابه، به عملیات ارسال R!x توجه نمایید، که R نشان‌دهنده کانال ورودی بولی با دو خط دیتای خروجی R:0 و R:1 و یک خط ورودی تأیید R:a می‌باشد. یک حالت ممکن HSE از ارسال که با HSE بالا در مورد دریافت سازگار است در زیر مشاهده می‌کنید:

$$send \equiv [\neg x \longrightarrow R.0\uparrow [] \; x \longrightarrow R.1\uparrow]; \; [R.a]; \; (R.0\downarrow, \; R.1\downarrow); \; [\neg R.a]$$

۸-۳ روش طراحی عمومی

اگرچه ابزار و سبک مدارها در طول سالها تغییر نموده‌اند، روش کلی سنتز دست نخورده و کامل باقی مانده است. این روش براساس تبدیل معناشناسی[1] بنا شده است که می‌تواند به صورت خودکار و یا دستی تولید شود. به خاطر اینکه این تبدیل، معنای برنامه را حفظ می‌کند، نتایج توسط ساختن اصلاح می‌شود.

اولین مرحله این روش، این است که رفتار مدار آسنکرون را به صورت برنامه CHP ترتیبی مشخص نمایید. اولین تبدیل (Process decomposition)، جایگزین نمودن برنامه ترتیبی با فرآیند CHP همروند می‌باشد؛ تبدیل می‌تواند روی فرآیند جدید تکرار شود.

تبدیل بعدی، (Handshaking expansion)، دیتا را فقط با استفاده از متغیرهای بولی کد می‌کند (Node هایی که در مدار احتمالی وجود دارند)، و تمام عملیات ارتباطی را به دستدهی[2] چهار فازه تبدیل می‌کند.

سرانجام، HSE را کامپایل می‌کنیم تا به قوانین ساخت دست یابیم. قوانین ساخت به فرم $B \rightarrow x\uparrow$ or $B \rightarrow x\downarrow$ می‌باشد، که ترم اول گره x را به Vdd (True) می‌نشاند وقتی عبارت بولی B (که Guard نامیده می‌شود) درست باشد، و در ترم دوم، گره x به GND (False) نشانده می‌شود وقتی عبارت بولی B درست باشد. برای مثال، ما می‌توانیم یک گیت NAND دو ورودی را با قوانین ساخت زیر بیان کنیم:

---

[1] Semantics-preserving
[2] Handshake

$$a \wedge b \longrightarrow x\downarrow \quad , \qquad \neg a \vee \neg b \longrightarrow x\uparrow$$

تمام قوانین ساخت در یک مجموعه به صورت همروند اجرا می‌شود. نظر باینکه ترتیبی مشخص نشده است، ترتیب به صورت ضمنی می‌باشد و توسط فرآیند کامپایل اداره می‌شود، که نبودن هزارد و وجود پایداری در مدار (اگر یک guard، true شود، همانطور true باقی می‌ماند تا آن قانون ساخت اجرا شود) و عدم تداخل (guardهایی که برای $x\uparrow$ و $x\downarrow$ می‌باشند هرگز باهم و در یک زمان true نمی‌شوند.) را تضمین می‌کند.

۸-۴ طراحی مدارهای Traditional QDI

پایه مدارهای منطقی بافر می‌باشد که که ورودی‌ها دارای یک مقدار x روی یک کانال (به نام L) و خروجی‌ها نتیجه یک محاسبه، f(x)، روی کانال دیگر (به نام R) می‌باشد. در CHP، این بافر به صورت $*[L?x; R!f(x)]$ درمی‌آید. یک بافر ممکن است دارای چندین ورودی و خروجی باشد که شامل شرط نیز باشند.

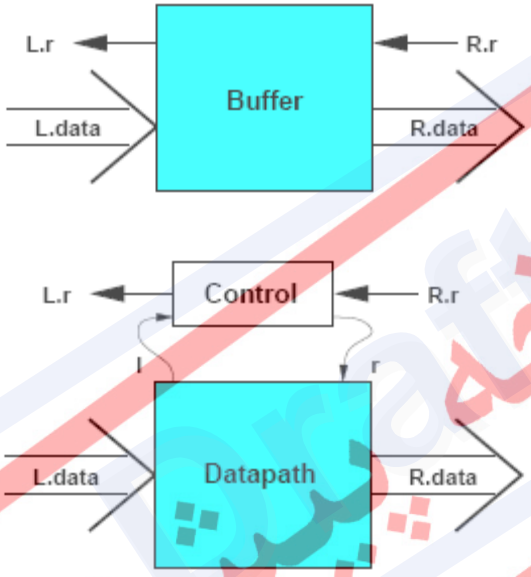
به منظور سادگی، فرض کنید f یک عمل همانی است. بنابراین، HSE برای بافر با جایگزینی L?x و R!x با HSE مربوط به آن به دست می‌آید. به عنوان نمونه در زیر:

$$*[L.r\uparrow;\ [L.0 \longrightarrow x\downarrow\ []\ L.1 \longrightarrow x\uparrow];\ L.r\downarrow;\ [\neg L.0 \wedge \neg L.1];$$
$$[R.r];\ [\neg x \longrightarrow R.0\uparrow [] x \longrightarrow R.1\uparrow];\ [\neg R.r];\ (R.0\downarrow,\ R.1\downarrow)\ ]\ .$$

برای آسان نمودن تجزیه، از یک HSE متفاوت با آنچه در قسمت قبل انجام دادیم استفاده می‌کنیم. به جای یک سیگنال تأیید، از یک سیگنال درخواست استفاده می‌کنیم که: L.r برای HSE دریافتی، و R.r برای HSE ارسالی می‌باشد.

نسخه پیش‌نویس Draft

برای یک مسیر داده[1] یک بیتی و یک تابع ساده، این HSE را به صورت مستقیم پیاده‌سازی می-کنیم. اما در صورت بزرگ شدن مسیر داده و پیچیده شدن تابع، غیر قابل کنترل می‌شود. در طراحی traditional QDI ، پیاده‌سازی استاندارد شامل قسمت کنترل[2] و مسیر داده جدا می‌باشد. در مثال بالا، ما احتیاج به معرفی کردن یک زوج سیگنال داخلی l و r برای اتصال قسمت‌های Control و مسیر داده داریم که در شکل ۱ نشان داده شده است.



شکل 8-۱ - معرفی زوج سیگنال داخلی l و r

HSE برای Control به صورت زیر می‌باشد:

$$control \equiv *[L.r\uparrow;\ [l];\ L.r\downarrow;\ [\neg l\ \wedge\ R.r];\ r\uparrow;\ [\neg R.r];\ r\downarrow\ ]\ .$$

HSE برای مسیر داده شامل دو قسمت است، یک قسمت برای نوشتن دیتا داخل رجیستر:

$$write\ \ x \equiv *[[L.0\ \longrightarrow\ x\downarrow\ []\ L.1\ \longrightarrow\ x\uparrow];\ l\uparrow;\ [\neg L.0\ \wedge\ \neg L.1];\ l\downarrow]\ ,$$

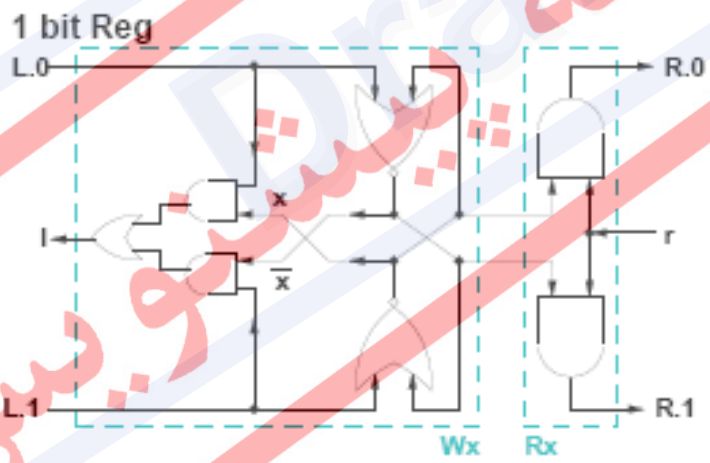
و قسمت دیگر برای خواندن مقدار x :

[1] Datapath
[2] Control

نسخه پیش‌نویس Draft

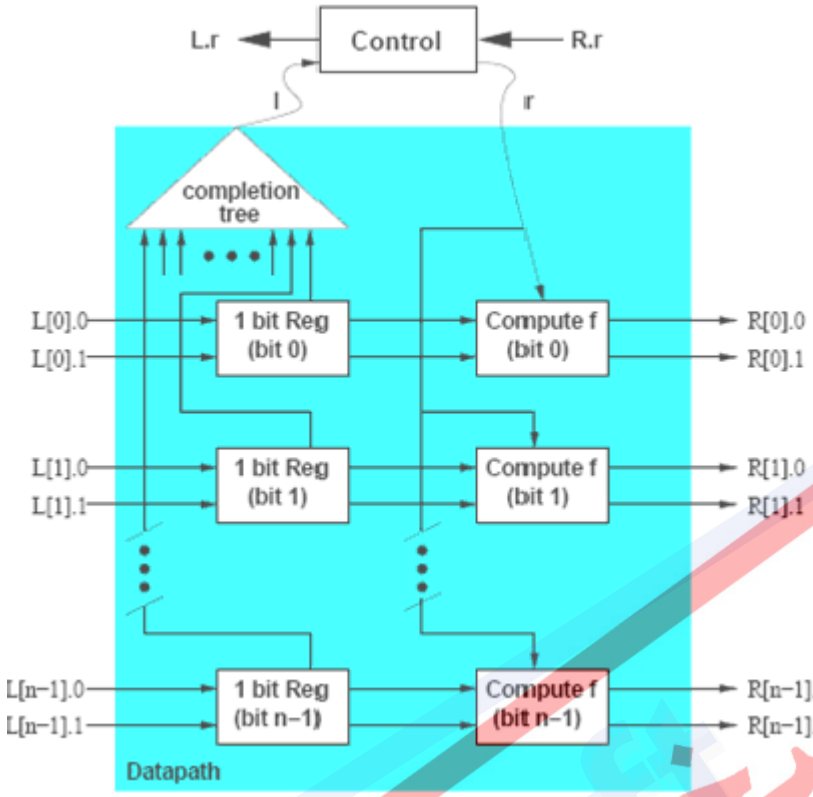
$$read\ x \equiv *[\ [r];\ [\neg x \longrightarrow R.0\uparrow [] x \longrightarrow R.1\uparrow];\ [\neg r];\ (R.0\downarrow,\ R.1\downarrow)]\ .$$

سیگنال 1 توسط قسمت نوشتن مسیر داده تولید می‌شود تا نوشتن مقدار بولی حاضر روی سیمهای L.0 و L.1 داخل x را تأیید کند. شکل ۲ پیاده‌سازی استاندارد-گیت مسیر داده یک بیتی را که با HSE بالا مشخص شده است نشان می‌دهد. شکل ۳ پیاده‌سازی مسیر داده را برای یک مقدار دلخواه تعداد بیت ورودی و تابع دلخواه f نشان می‌دهد. بخش Completion tree یک سیگنال یک بیتی به بخش کنترلی می‌فرستد: '1' وقتی تمام رجیسترهای یک بیتی با یک مقدار معتبر که روی سیمهای ورودی رجیستر قرار دارند مقدار دهی شوند، '0' وقتی تمام رجیسترهای یک بیتی به یک مقدار اولیه بازنشانی شوند.



شکل 8-۲ - مسیر داده یک بیتی

نسخه پیش‌نویس
Draft



شکل 8-۳ - مسیر داده برای یک مقدار دلخواه تعداد بیت ورودی و تابع دلخواه f

۵-۸ CHP

- تکرار نامحدود عبارت S : [S]*
- برای متغیر بولی x، $x\uparrow$ و $x\downarrow$ به ترتیب مقدار '1' و '0' را به x نسبت می‌دهند.
- آرایش ترتیبی توسط ';' مشخص می‌شود. آرایش همروند دو عملیات ابتدایی توسط ',' نشان داده می‌شود.
- عبارت انتخابی قطعی $[G_1 \rightarrow S_1 [] \ldots [] G_n \rightarrow S_n]$ ، که G_i ها عبارات منطقی (Guard) هستند و S_iها قطعه برنامه می‌باشند، بدین صورت اجرا می‌شوند: تا زمانیکه یک نگهبان (guard) درست شود صبر می‌کند، سپس S_i مربوط به G_i درست، اجرا می‌شود.
- عبارت [G] خلاصه شده عبارت یک نگهبانی [G → skip] می‌باشد و نشان دهنده انتظار برای درست ارزیابی شدن G می‌باشد.

نسخه پیش‌نویس Draft

- توالی اعمالی که دستدهی[1] چهار فازه را پیاده‌سازی می‌کند، تناوب انتظار و انتساب بولی می‌باشد: $[B]; x\uparrow; [\neg B]; x\downarrow$. این توالی بسط دستدهی[2] یا HSE نامیده می‌شود.
- ارتباط: R!e به معنی "ارسال مقدار e داخل کانال R" و L?x به معنی "دریافت مقداری از کانال L و ذخیره آن در متغیر x" می‌باشد.

نسخه پیش‌نویس
Draft

[1] Handshake
[2] handshaking expansion

نسخه پیشنویس
Draft

فصل ۹

افزاره‌های منطقی برنامه پذیر

نسخه پیش‌نویس
Draft

۹-۱ مقدمه

در این قسمت به بحث و بررسی در مورد افزاره‌های منطقی برنامه‌پذیر[1] می‌پردازیم که امروزه به طور وسیعی از آنها برای پیاده‌سازی مدارهای منطقی استفاده می‌شود. همانطور که در بخشهای قبل صحبت شد، یکی از راههای آزمون صحت عملکرد مدار طراحی شده، شبیه‌سازی است. اما تجربه نشان داده است که شبیه‌سازی به تنهایی نمی‌تواند تمام مشکلات طراحی، خصوصاً مشکلاتی که مربوط به ملاحظات پیاده‌سازی است را نشان دهد. بنابراین، بهترین راه حل برای اطمینان از عملکرد مدار طراحی شده، ساخت نمونهٔ آزمایشگاهی است. انواع پیاده‌سازی مدارهای منطقی به شرح زیر است.

- **پیاده سازی آزمایشگاهی:** این نوع پیاده سازی هزینهٔ کمتر و قابلیت اطمینان پائین‌تری دارد.
- **پیاده سازی سفارشی**[2]**:** این نوع پیاده سازی هزینهٔ بیشتر و قابلیت اطمینان بالاتری دارد.

در مقدمهٔ این مطلب باید اشاره کرد که با افزایش سطح تجمع سازی (تعداد گیت‌ها در آی‌سی)، تعداد آی‌سی‌ها کاهش یافته است. طی ۲۰ سال گذشته، تعداد گیت در یک تراشه از چند گیت در آی‌سی‌های SSI سری به بیش از ۱ میلیون گیت در آی‌سی‌های VLSI امروزی رسیده است. با توجه به جدول زیر و پیشرفت فن‌آوری VLSI ، پیاده‌سازی آزمایشگاهی پیشرفت‌های زیادی داشته است و امکان تولید نمونه‌های آزمایشگاهی با هزینهٔ بسیار پایین وجود دارد.

جدول 9-۱ - جدول مقیاس مدارهای مجتمع دیجیتال

- Small Scale Integration (SSI)	حداکثر 10 گیت
- Medium Scale Integration (MSI)	10 تا 100 گیت
- Large Scale Integration (LSI)	100 تا 1000 گیت
- Very Large Scale Integration (VLSI)	بیشتر از 1000 گیت تا 10×10^6 گیت
- Ultra Large Scale Integration (ULSI)	بیشتر از 10×10^6 گیت تا 10^9 گیت

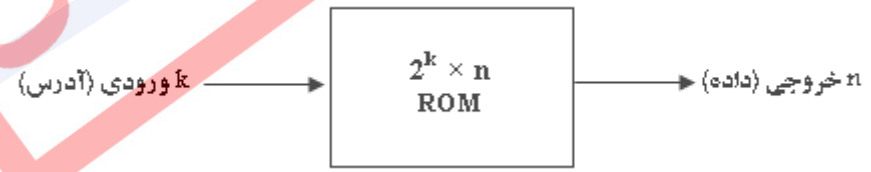
[1] Programmable Logic Device (PLD)
[2] Coustom Design

نسخه پیش‌نویس

۹-۲ حافظهٔ فقط خواندنی، اولین PLD

اولین افزاره‌ای که بعنوان PLD مورد استفاده قرار گرفت، حافظهٔ فقط خواندنی[1] بود. توسط این حافظه امکان پیاده‌سازی مدارهای ترکیبی بصورت SOP وجود دارد. در این قسمت، یک مرور کلی بر انواع حافظه می‌نماییم. به طور کلی، در سیستم های دیجیتال دو نوع حافظه مورد استفاده قرار می گیرد: حافظهٔ با دستیابی تصادفی[2] و حافظهٔ فقط خواندنی. حافظهٔ RAM اطلاعات جدید را برای ذخیره می‌پذیرد تا بعداً مورد استفاده قرار گیرد. عمل ذخیره کردن اطلاعات جدید را عمل نوشتن در حافظه می‌گویند. فرآیند انتقال اطلاعات ذخیره شده از حافظه به بیرون را عمل خواندن می‌گویند. حافظهٔ RAM هر دو عمل خواندن و نوشتن را انجام می‌دهد اما حافظهٔ ROM فقط عمل خواندن را انجام می‌دهد. به این معنی که اطلاعات دودویی ذخیره شده در حافظه در هر زمان قابل بازیابی است. بنابراین، حافظهٔ ROM یک وسیلهٔ منطقی برنامه‌پذیر است. پس به طور کلی، حافظهٔ فقط خواندنی (ROM) اساساً یک وسیلهٔ ذخیره‌سازی است که در آن اطلاعات دودویی به طور دائم نگهداری می‌شود. این اطلاعات باید توسط طراح مشخص شود و سپس برای ایجاد الگوی اتصالات داخلی، به واحد حافظه وارد گردد. به محض ایجاد الگو، حتی با قطع و وصل منبع تغذیه، اطلاعات در آن باقی خواهد ماند. با استفاده از این خاصیت ROM از آن بعنوان یک Lookup Table برای پیاده‌سازی مدارها استفاده می‌شود.

نمودار بلوکی ROM در شکل زیر نشان داده شده است. این حافظه دارای k بیت ورودی و n بیت خروجی است. ورودی‌ها، آدرس حافظه را مهیا می‌کنند و خروجی‌ها، بیت‌های دادهٔ کلمهٔ ذخیره شدهٔ انتخابی به وسیلهٔ آدرس را فراهم می‌نمایند. تعداد کلمات در حافظه با توجه به k بیت خط آدرس، برابر 2^k کلمه است.



[1] Read Only Memory (ROM)
[2] Random Access Memory (RAM)

۹-۲-۱ انواع مختلف ROM

یک ROM به چهار روش مختلف برنامه‌ریزی می‌شود. اولین روش که برنامه‌ریزی ماسک نامیده می‌شود، به وسیله سازنده در آخرین مرحلهٔ ساخت قطعه صورت می‌گیرد. در این روش، ساخت ROM مستلزم پر شدن جدول درستی مربوطه به وسیلهٔ سفارش دهنده است. این جدول ممکن است روی فرم مخصوصی که توسط سازنده ارائه می‌شود، و یا قالب خاصی، روی خروجی‌های کامپیوتر ایجاد گردد. بر طبق جدول مشتری، سازنده، ماسک مربوطه را برای تولید 1 ها و 0 ها برای مسیر می‌سازد. این روش بسیار پرهزینه است زیرا سازنده مبلغ خاصی را برای ماسک کردن ROM از مشتری می‌گیرد. به همین دلیل، برنامه‌ریزی ماسک هنگامی اقتصادی است که تعداد زیادی از یک نوع ROM سفارش داده شود.

برای تعداد کم، نوع دوم آن به نام حافظهٔ فقط خواندنی برنامه‌پذیر[1] اقتصادی‌تر است. هنگام سفارش، همهٔ فیوزهای PROM در حالت سالم قرار دارند، بدین معنی که جریان را از خود عبور می‌دهند. می‌توان با استفاده از یک پالس ولتاژ قوی و از طریق پایهٔ خاصی، این فیوزها را سوزاند. فیوز سوخته منطق 0 و فیوز دست نخورده منطق 1 را ایجاد می‌نماید. این امکان به کاربر اجازه می‌دهد تا PROM را در آزمایشگاه برنامه‌ریزی کند. لذا دستگاه خاصی به نام برنامه نویس PROM برای انجام این کار به صورت تجاری در اختیار قرار دارد. روال برنامه‌نویسی یا برنامه‌ریزی سخت‌افزاری PROM ها برگشت ناپذیر است و به محض انجام آن، الگویی ثابت و دائمی در آن ایجاد می‌شود که تغییر ناپذیر می‌باشد (به این معنی که فقط یکبار قابل برنامه ریزی می‌باشند). بنابراین، در صورت لزوم باید آنها را دور انداخت و آی‌سی جدیدی را برنامه‌ریزی کرد. لذا در این موارد از سومین نوع، به نام ROM برنامه‌پذیر قابل پاک شدن[2] استفاده می‌شود. اگر تراشه از طریق پنجرهٔ کوارتز تعبیه شده در روی بستهٔ آی‌سی، در معرض نور فرا بنفش قرار گیرد، بار ذخیره شده به سرعت تخلیه می‌شود و به این ترتیب پس از پاک شدن، EPROM به حالت اولیهٔ خود بازگشته و لذا می‌توان اطلاعات جدیدی را در EPROM ذخیره کرد. می‌توان فرایند پاک شدن و برنامه ریزی را تا دستیابی به طرح نهایی ادامه داد.

نوع چهارم ROM ، برنامه پذیر قابل پاک شدن الکتریکی[3] است. این حافظه شبیه EPROM است با این تفاوت که می‌توان اتصالات برنامه‌ریزی شدهٔ قبلی را به صورت الکتریکی پاک کرد. مزیت این کار در این است که لازم نیست برای پاک کردن و برنامه ریزی، آن را از مدار خارج کرد. پس E2PROM برای کاربردهایی که تغییر اطلاعات باید بدون جابجایی فیزیکی آی‌سی صورت گیرد، مفید است.

---

[1] Programmable Read Only Memory (PROM)
[2] Erasable Programmable Read Only Memory (EPROM)
[3] Electrical Erasable Programmable Read Only Memory (E2PROM)

نسخه پیش‌نویس

۹-۲-۲ مقایسهٔ انواع مختلف ROM

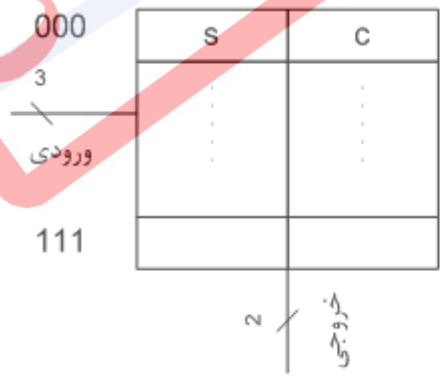
EPROM و E2PROM ها بسیار پیچیده‌تر از ROM هستند و به همین خاطر گران‌ترند. همچنین، تأخیر EPROM و E2PROM ها از تأخیر PROM و ROM ها بیشتر است، زیرا با تکنولوژی NMOS و CMOS ساخته می‌شوند، حال آنکه PROM و ROM ها معمولاً با ترانزیستور دو قطبی ساخته می‌شوند. اما قیمت بیشتر و تأخیر بیشتر در EPROM وE2PROM ها به وسیلهٔ امکان پاک کردن و برنامه‌ریزی مجدد جبران می‌شود. EPROM و E2PROM ها به کار رفته در فرآیند طراحی، برای تولید نهایی نگهداشته می‌شوند. ولی در تولید انبوه، استفاده از PROM و ROM ها، هزینهٔ تولید را کاهش می‌دهد.

۹-۲-۳ پیاده‌سازی مدارهای منطقی به کمک ROM

در ROM ها، سرعت مستقل از ورودی و حالت است. چون بصورت یک Lookup Table عمل می‌کنند. به عبارت دیگر، زمان تأخیر انتشار برابر زمان دسترسی به حافظه است. برای مثال، پیاده‌سازی یک تمام جمع کننده با ROM را در زیر ملاحظه می‌نمایید. ورودی و خروجی‌ها دقیقاً شبیه تمام جمع‌کننده است. بنابراین، ورودی‌های حافظه سه بیت Ai و Bi و Ci-1 می‌باشند و خروجی‌های آن Si و Ci است. در این صورت، هشت خانه (2^3) از حافظه و از هر خانه دو بیت (S,C) استفاده می‌شود. محتویات هر خانه، متناظر با ورودی آن و دقیقاً برابر مقدار در جدول درستی مدار تمام جمع کننده است.

$$Si = Ai \oplus Bi \oplus Ci\text{-}1$$

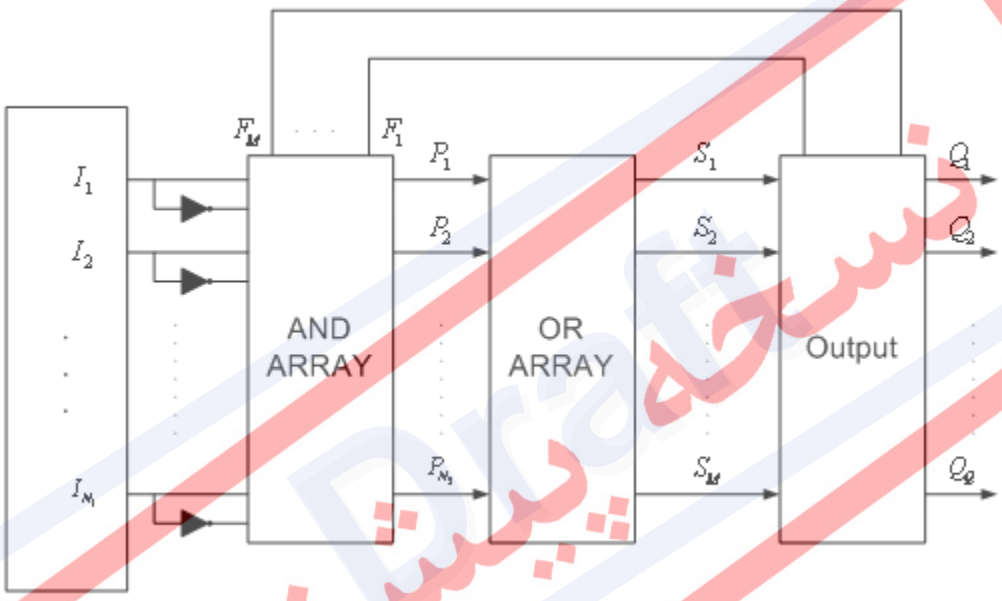
$$Ci = Ai.\ Bi + (Ai + Bi).Ci\text{-}1$$



ورودی	خروجی
3	2

۹-۳ آرایهٔ منطقی برنامه‌پذیر

اولین افزارهٔ برنامه‌پذیر که دقیقاً برای این کار ساخته شده بود، آرایهٔ منطقی برنامه‌پذیر[1] (PLA) می‌باشد. ترکیب یک آرایهٔ AND برنامه‌پذیر و یک آرایهٔ OR برنامه‌پذیر را آرایهٔ منطقی برنامه‌پذیر PLA می‌نامند. ساختار این افزاره را در شکل ۱ ملاحظه می‌نمایید. ورودی‌های مدار Ii هستند، حاصلضرب‌ها توسط آرایهٔ AND تولید شده(Pi) و برای جمع شدن به آرایهٔ OR ارسال می‌شوند. در نهایت، خروجی‌های بدست آمده (Si) با عبور از یک طبقه که در ادامه در مورد آن صحبت خواهد شد، به خروجی افزاره منتقل می‌شوند.

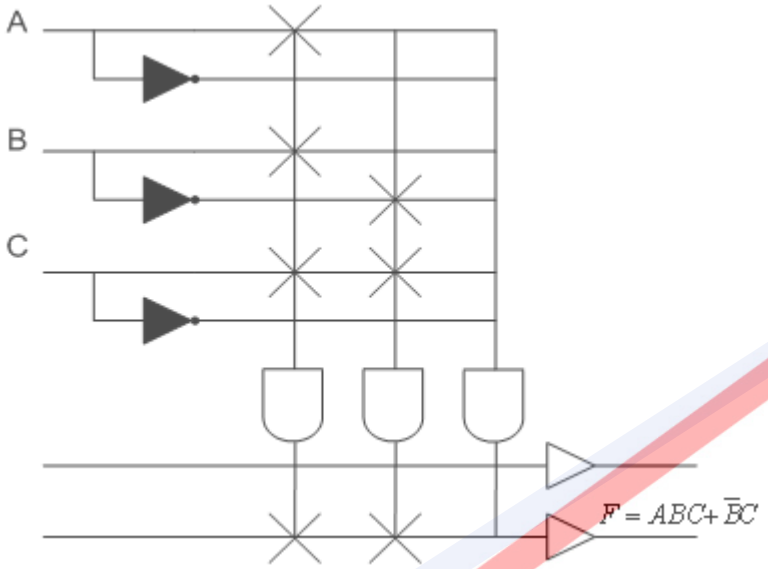


شکل 9-۱ - آرایهٔ منطقی برنامه‌پذیر (PLA)

[1] Programmable Logic Array

نسخه پیش‌نویس
Draft

مثال) پیاده سازی تابع $F = ABC + \overline{B}C$ با PLA :



توجه کنید که نمایش سیمها در اینجا دیاگرامی است، یعنی اگر A و B را به گیت اول وصل کرده‌ایم بدین معنی نیست که دو ورودی را به یک سیم وصل کرده‌ایم.

۹-۳-۱ اتصالات قابل برنامه ریزی

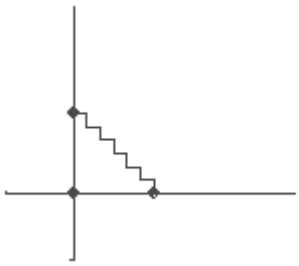
همانطور که در قسمت قبل و در مثال مشاهده نمودید، در افزاره‌های برنامه‌پذیر، قسمت اتصالات قابل برنامه‌ریزی است و تعداد و ظرفیت گیت‌ها ثابت است. در ادامه، بطور مختصر در مورد این اتصالات بحث می‌نماییم. روش برنامه‌ریزی افزاره (اتصالات قابل برنامه‌ریزی) با توجه به نوع افزاره به دو صورت انجام می‌شود.

- **اتصالات براساس فیوز**[1]

در این حالت در محل تمام اتصالات به شکل زیر یک فیوز وجود دارد. واضح است که در این حالت تمام اتصالات وصل هستند و برای برنامه ریزی باید اتصالات زائد را از بین برد. بنابراین، برنامه‌ریزی مانند سوزاندن فیوزهای اضافی خواهد بود. برای مثال، علامت‌هایی که در مثال قبل وجود دارد، نشان دهندهٔ باقی ماندن فیوز در آن اتصالات و سوزاندن بقیه فیوزها می‌باشد.

[1] Fuse Based

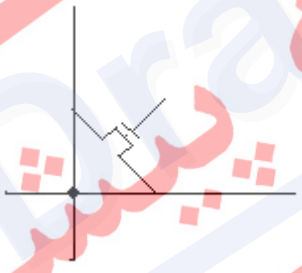
نسخه پیش‌نویس Draft



مشخص است که به دلیل ساختار این اتصالات، افزارهای مورد نظر فقط یکبار برنامه‌ریزی می‌شوند و امکان برگرداندن فیوزهای سوخته وجود ندارد.

- **اتصالات براساس ترانزیستورهای اثرمیدان[1]**

در این نوع اتصالات، از ترانزیستورهای MOSFET در محل اتصالات به شکل زیر استفاده می‌شود. این ترانزیستورها، امکان بارشدن در محل ورودی گیت را دارا می‌باشند و بعد از قرار گرفتن بار الکتریکی مثبت روی گیت، وصل می‌شوند. در این صورت، اتصالات در حالت عادی قطع هستند و بعد از بارکردن وصل می‌شوند.



نسخه پیش‌نویس Draft

۹-۳-۲ انتخاب قطبیت خروجی

یکی از عملیاتی که در لایهٔ خروجی انجام می‌شود، انتخاب قطبیت خروجی[2] است. برای این کار باید امکان انتخاب قطبیت بصورت زیر فراهم باشد.

قطبیت خروجی

Active low

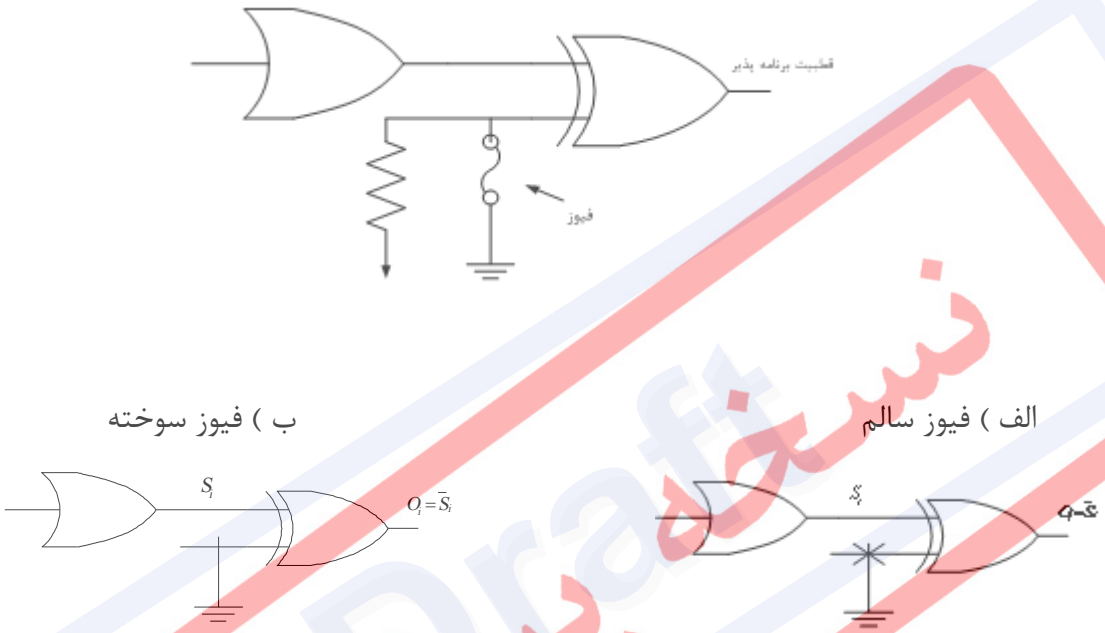
T	0
F	1

Active high

T	1
F	0

[1] Field Programmable
[2] Output Polarity

برای ایجاد خروجی برنامه پذیر، یک گیت XOR با ورودی فیوزدار به کار می‌رود. ورودی فیوزدار ۰ یا ۱ است، بسته به اینکه فیوز سالم بماند یا سوخته شود. با توجه به عمل XOR ، وقتی فیوز سالم است طبق شکل (الف)، خروجی عبارت است از $O_i = S_i \oplus 0 = S_i$ ، پس خروجی یک فعال[1] است. اگر فیوز سوخته شود در آنصورت طبق شکل۲ (ب) خروجی عبارت است از $O_i = S_i \oplus 1 = \overline{S_i}$ پس خروجی صفر فعال[2] است.



شکل 9-۲ - انتخاب قطبیت خروجی

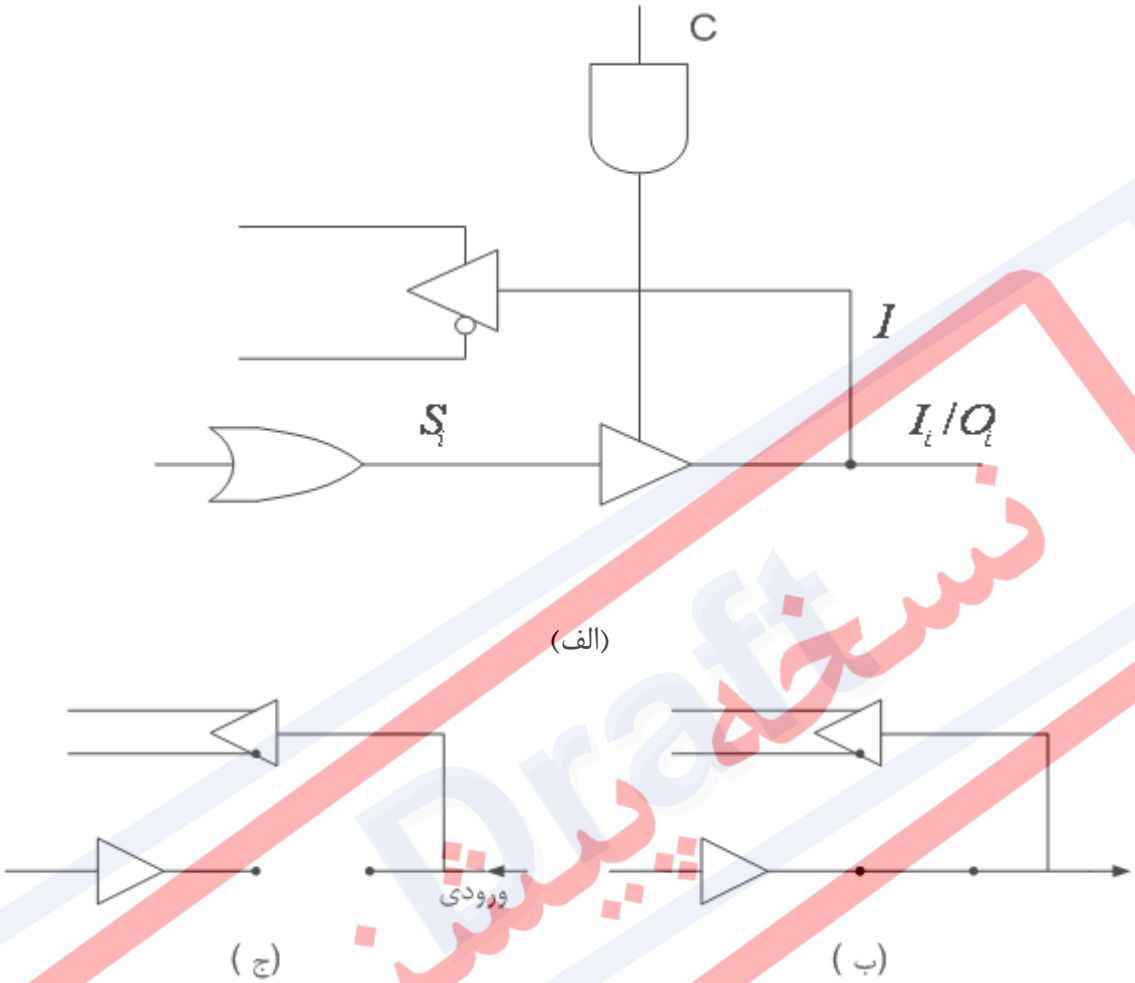
۹-۳-۳ پایه های ورودی/خروجی

ویژگی دیگری که در بسیاری از وسایل منطقی برنامه‌پذیر دیده می‌شود، وجود پایه‌های ورودی/خروجی به صورت شکل۳ (الف) است که باز هم در طبقهٔ خروجی قرار می‌گیرد (IO_m یک پایهٔ دو طرفهٔ I/O است). پایهٔ دو طرفه، بوسیلهٔ یک بافر سه حالته[3] ایجاد می‌شود. خط کنترل این بافر به یکی از جملات حاصل ضربی متصل است. وقتی خط کنترل یک شود، این بافر فعال شده و مطابق شکل۳ (ب) به صورت اتصال کوتاه (یا کلید بسته) عمل می‌کند. در این حالت، پایهٔ خروجی تشکیل می‌شود. این مقدار به آرایهٔ AND بازخورد داده می‌شود و می‌توان آن را برای ایجاد جملات حاصلضربی به کار برد.

[1] Active High
[2] Active Low
[3] Tri-State Buffer

نسخه پیش‌نویس Draft

اگر خط کنترل صفر باشد، بافر غیر فعال است. در آن صورت، مطابق شکل۳ (ج) مدار به صورت اتصال باز (کلید باز) عمل می‌کند. لذا این پایه از طریق خط بازخورد، یک ورودی آرایهٔ AND به حساب می‌آید.



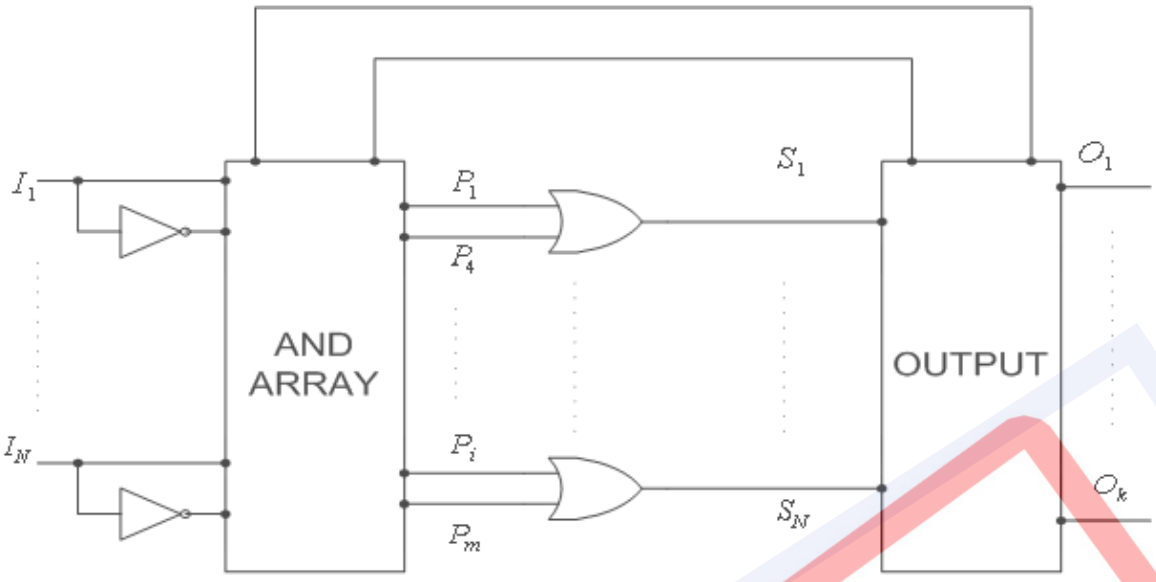
شکل 9-۳ -(الف) پایه های ورودی/خروجی بوسیله بافر (ب) حالت خروجی (ج) حالت ورودی

۹-۴ منطق آرایه‌ای برنامه‌پذیر

با توجه به هزینهٔ بالا و سرعت پایین در افزارهٔ PLA ، طراحان به این فکر افتادند که با کاهش انعطاف پذیری در این افزاره، علاوه برکاهش هزینه، سرعت مدار را نیز افزایش دهند. این کار منجر به ساخت افزارهٔ جدید با نام "منطق آرایه‌ای برنامه‌پذیر[1]" (PAL) شد. در این افزار که شباهت زیادی به PLA دارد، آرایه OR قابل برنامه ریزی نیست و ورودی های آن ثابت فرض شده است. دیاگرام کلی آن در شکل ۴ مشخص شده است.

[1] Programmable Array Logic

نسخه پیش‌نویس Draft



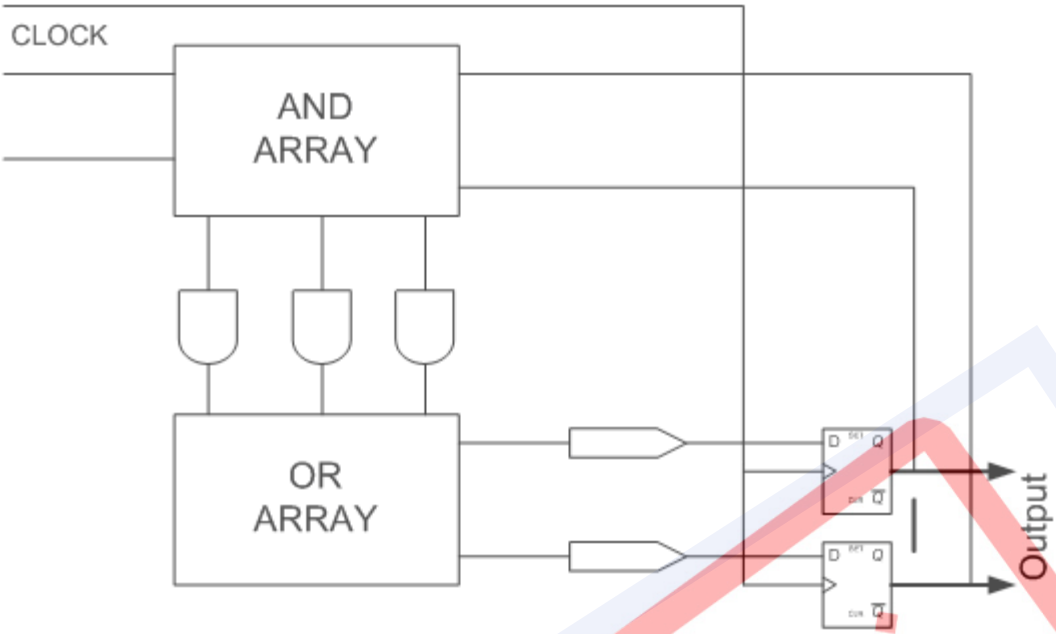
شکل 9-۴ - منطق آرایه‌ای برنامه‌پذیر (PAL)

به این ترتیب برنامه‌ریزی آن بسیار شبیه PLA می‌باشد. فقط قسمت OR را ثابت فرض می‌کنیم.

۹-۵ افزاره‌های برنامه‌پذیر برای مدارهای ترتیبی

همانگونه که قبلا اشاره شد، ساختار مدارهای ترتیبی از مدارهای ترکیبی و عناصر حافظه تشکیل شده است. بنابراین با اضافه کردن عناصر حافظه به افزاره‌های شرح داده شده، امکان پیاده‌سازی مدارهای منطقی ترتیبی فراهم می‌شود. این مورد برای هر دو نوع PAL و PLA صادق است. برای مثال یک نمونه از افزارهٔ برنامه‌پذیر PLA دارای فیلیپ-فلاپ (رجیستر)[1] در شکل۵ آمده است. همانطور که ملاحظه می‌کنید، ساختار آن کاملاً منطبق بر ساختار مدارهای ترتیبی همگام است. به این ترتیب مدارهای سنتز شده در بخش‌های قبلی را می‌توان روی این افزاره برنامه‌ریزی نمود و عملکرد آنها را آزمود.

[1] Registered PLA



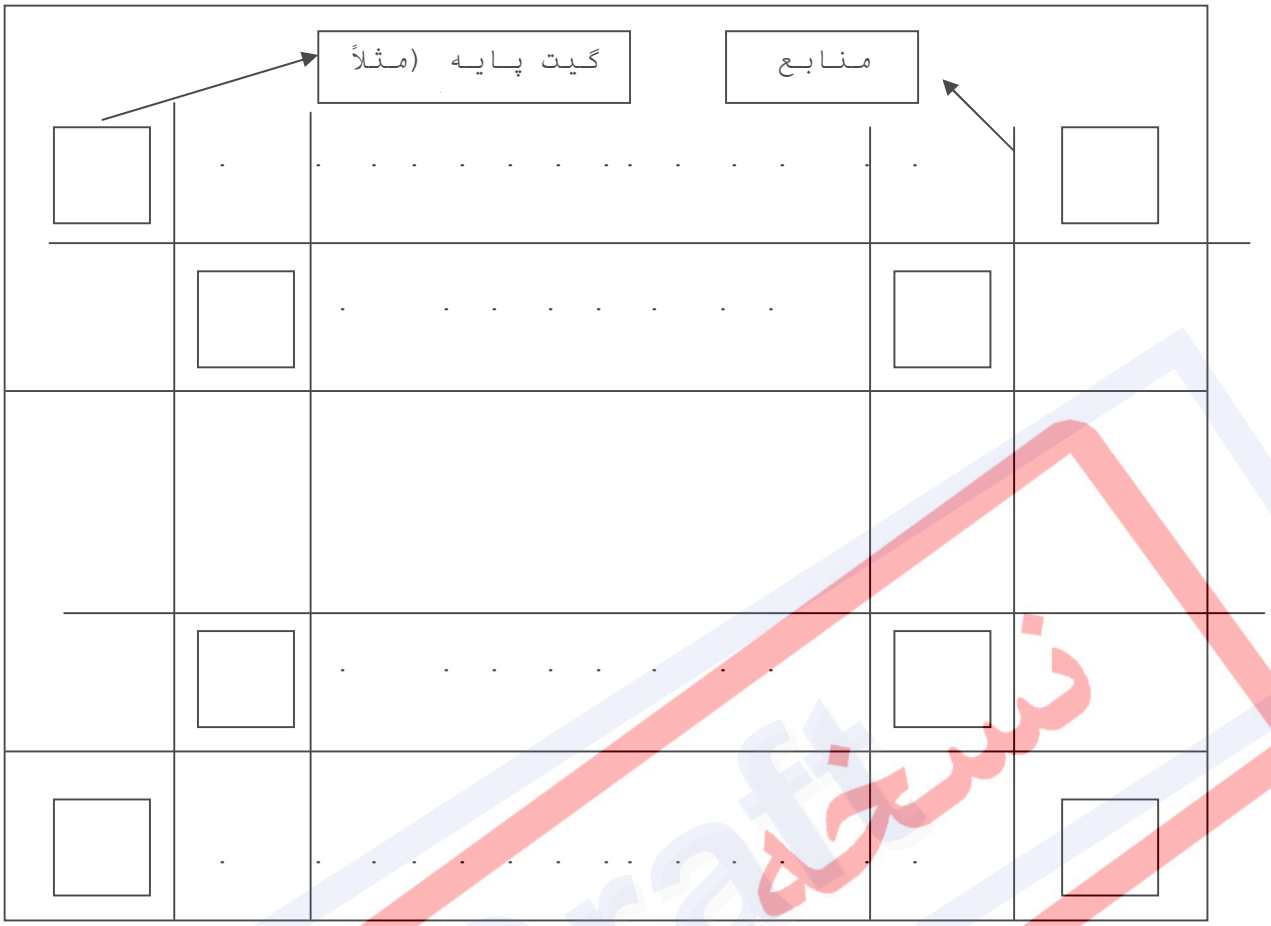
شکل 9-۵ - افزارهٔ برنامه‌پذیر PLA دارای فیلیپ-فلاپ

۹-۶ دریای گیت

با پیشرفت فن‌آوری VLSI و تراکم بیشتر تراشه‌ها، افزاره‌هایی با تعداد زیادی گیت ساخته شد. یکی از مهمترین آنها دریای گیت[1] می‌باشد. در این افزاره‌ها که دیاگرام کلی آن در شکل ۶ دیده می‌شود، مانند قبل منابع مسیریابی قابل برنامه‌ریزی هستند و امکان برقراری اتصالات مختلف را فراهم می‌کنند. علاوه بر مسیریابی، امکان برنامه ریزی غیر از حالت SOP نیز وجود دارد که می‌تواند در بعضی کاربردهای پیچیده مفید باشد. همچنین در این مجموعه نمی‌توان ساختار داخلی سلول‌ها را تغییر داد.

[1] Sea of Gate

نسخه پیش‌نویس Draft



شکل 9-۶ – دریای گیت

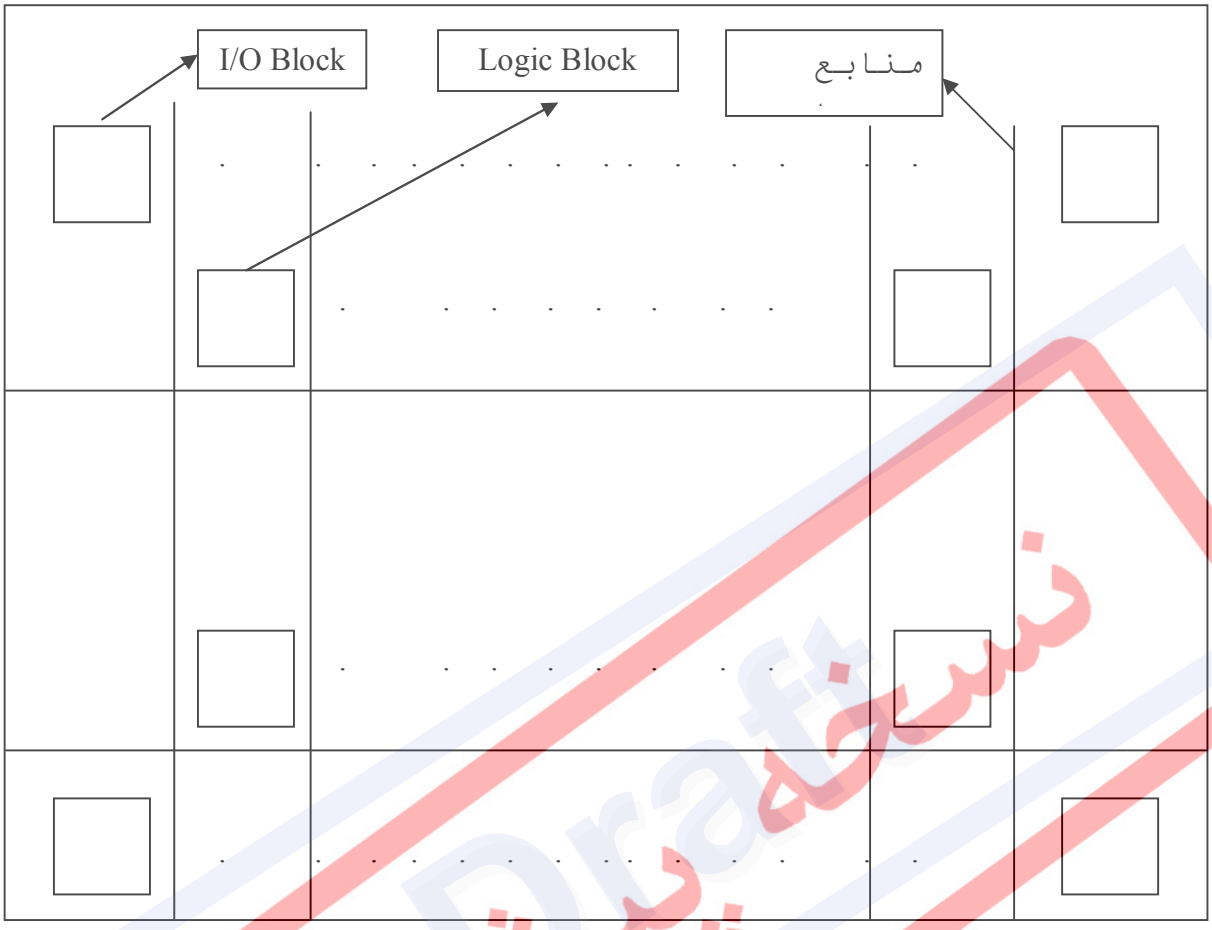
FPGA ۷-۹

تکامل یافتهٔ تمامی عناصر قبلی، FPGA[1] ها هستند. همانطور که در شکل ۷ مشاهده می‌نمایید، ساختار آنها بسیار شبیه دریای گیت است. تفاوت مهم آنها در این است که در FPGA امکان برنامه‌ریزی ساختارهای داخلی نیز وجود دارد. به همین دلیل، آنها را به نام بلوک‌های منطقی تعریف می‌کنند و قابلیت تبدیل به انواع مدارات ترکیبی[2] را دارند. اجزای یک FPGA عبارتند از:

- **بلوکهای منطقی:** برای پیاده‌سازی بخش‌های مختلف مدار.
- **بلوکهای ورودی/خروجی:** برای PADهای ورودی/خروجی استفاده می‌شوند.
- **منابع مسیریابی[3]:** اتصالات بین دو قسمت بالا را انجام می‌دهند.

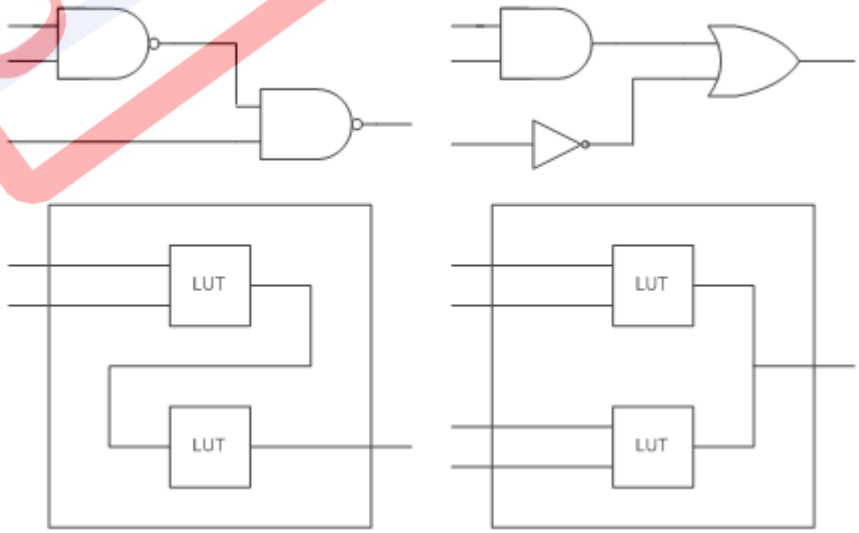
[1] Field Programmable Gate Array
[2] Logic Block
[3] Routing Resource

نسخه پیش‌نویس Draft



شکل ۹-۷- FPGA

داخل هر بلوک منطقی می‌توان مدارات مختلفی قرار داد، که یکی از رایج‌ترین آنها Lookup Table است. همچنین از ساختارهای گیت NAND نیز همانطور که در شکل ۸ ملاحظه می‌کنید، استفاده می‌شود.



نسخه پیش‌نویس Draft

شکل 9-۸ - Lookup Table استفاده شده در یک بلوک منطقی

روش‌های مختلفی برای پیاده‌سازی منابع مسیریابی وجود دارد که در زیر بصورت خلاصه به آنها اشاره شده است:

۱- SRAM-based: کنترل سوئیچ توسط یک خانه از SRAM انجام می‌شود.

۲- EPROM-based: سوئیچ‌ها ساختاری شبیه EPROM دارند و با تزریق بار گیت بر سوئیچ منتقل می‌شود.

۳- Anti-fuse-based: سوئیچ‌ها از نوع Anti-fuse هستند و با اعمال جریان الکتریکی به یک مسیر با مقاومت پایین تبدیل می‌شوند.

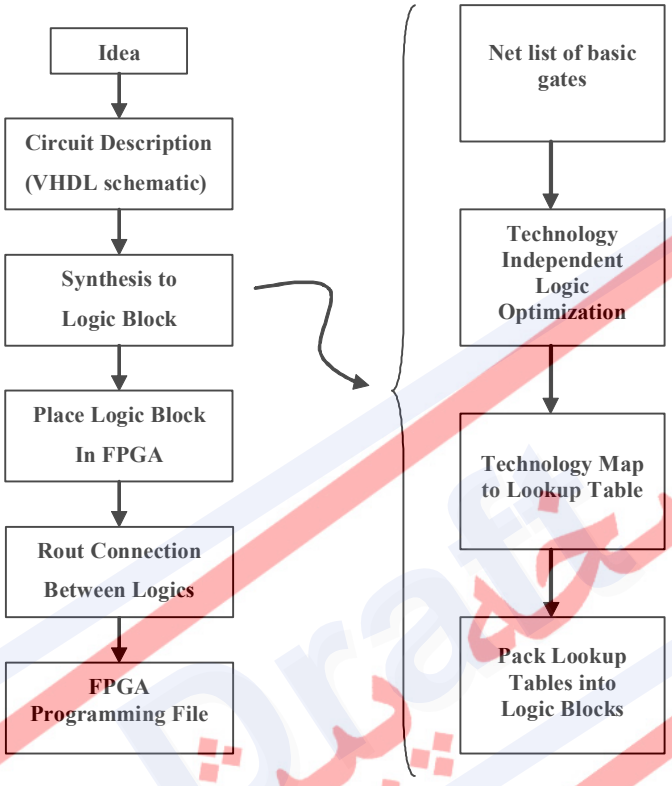
۹-۷-۱ ویژگی های FPGA

- پشتیبانی از سرعت‌های بالا
- تراکم نسبتاً مناسب
- زمان و هزینه نسبتاً کم برای پیاده‌سازی مدار
- زمان شبیه‌سازی کم
- امکان نمونه‌سازی آسان
- تغییر و بهبود آسان و کم هزینه

نسخه پیش‌نویس Draft

۹-۷-۲ روال طراحی با FPGA

شکل ۹، روال طراحی مدار روی FPGA از ابتدا تا مرحلهٔ پیاده‌سازی روی آن را نشان می‌دهد.



شکل 9-۹ - روال طراحی با FPGA

نسخه پیش‌نویس
Draft

۹-۸ تمرین

۱- با توجه به جدول حالت زیر مدار ترتیبی با ۴ حالت و با تخصیص حالت داده شده طراحی کنید. شماره پینی که به هر ورودی، خروجی، و متغیر حالت تخصیص داده شده است و همچنین معادله منطقی را در یک فرمت مناسب بنویسید.

الف) PAL 16R6

ب) PLS 155 به همراه فلیپ فلاپ هایی که برای عملکرد JK پیکربندی شده اند.

	X	
	0	1
A	B/0	C/0
B	D/0	A/1
C	A/1	D/0
D	D/1	B/1

y_1	y_2
0	0
0	1
1	1
1	0

۲- جدول حالت کاهش داده شده و جدول تخصیص حالت زیر داده شده است. معادله منطقی مدار ترتیبی همگام و دیاگرام منطقی مربوطه را با استفاده از ابزار زیر مشخص کنید.

الف) PAL 16R6

ب) PLS155 و با استفاده از فلیپ فلاپ هایی که دارای عملکرد JK هستند.

پ) 22V10 .

	X	
	0	1
A	A/0	B/0
B	C/0	B/0
C	D/0	B/0
D	A/1	B/0

y1	y2
0	0
0	1
1	1
1	0

نسخه پیش‌نویس Draft

۳- یک PAL16R6 برای جدول حالت کاهش یافته زیر طراحی کنید. این کار را با استفاده از تخصیص حالت one-hot که نشان داده شده، انجام دهید.

x

	0	1
A	B/0	D/0
B	A/0	C/1
C	D/1	C/0
D	B/1	E/1
E	C/0	A/0
F	E/0	F/1

y1	y2	y3	y4	y5	y6
1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

۴- یک ثبات ۸ بیتی با قابلیت بار کردن موازی با استفاده از PLS105 طراحی کنید. این ثبات باید دارای ۸ بیت D0-D7 به عنوان ورودی و ۸ بیت Q0-Q7 به عنوان خروجی، ورودی CLK برای پالس کلاک، و ورودی کنترلی PRE که نشان دهنده حاضر بودن داده ها ست، باشد. ورودی ها و خروجی ها و فیوز های ارتباطی را روی یک کپی از دیاگرام PLS105 علامت گذاری کنید.

۵- یک مبدل کد باینری BCD به کد excess-3 با استفاده از موارد زیر طراحی کنید:

الف) شبکه منطقی (مجموعه دو سطحی گیت های NAND)

ب) PLA

پ) ROM

ت) PAL

به خاطر داشته باشید که ابعاد ورودی و خروجی PLA ، ROM و PAL را بنویسید.

۶- توابع زیر را با استفاده از موارد زیر پیاده سازی کنید:

الف) یک دیکدر ۴ به ۱۶ و گیت های منطقی

ب) PLA

نسخه پیش نویس

ب)ROM

پ) PAL

$F1(A,B,C,D) = \sum m(0,1,2,3,6,9,11)$

$F2(A,B,C,D) = \sum m(0,1,6,8,9)$

$F3(A,B,C,D) = \sum m(2,3,8,9,11)$

۷- از یک ROM ۶× ۳۲ برای تبدیل یک عدد ۶ بیتی به معادل ۲ رقمی با نمایش BCD استفاده کنید.

$(a5\ a4\ a3\ a2\ a1\ a0) = [(x3\ x2\ x1\ x0)_{BCD}\ (y3\ y2\ y1\ y0)_{BCD}]_{10}$

محتوای ROM را در قالب یک جدول درستی نشان دهید.

(راهنمایی: $x0 = 0$, $y0 = a0$)

۸- یک بالا\پایین شمارنده ۴ بیتی با پیمانه ۱۲ با استفاده از یک PLS155 به همراه فلیپ فلاپ هایی که برای عملکرد JK پیکربندی شده اند، بسازید. شمارنده باید دارای ورودی های موازی $\{D, C, B, A\}$ و خروجی های $\{Q_D, Q_C, Q_B, Q_A\}$ ورودی پالس ساعت CLK و دو ورودی انتخاب تابع S1S0 باشد. توابع شمارنده در جدول زیر نشان داده شده اند:

S_0	S_1	Mode
0	0	No operation
0	1	Load
1	0	Count up
1	1	Count down

الف) معادله منطقی موجود را به دست آورده و پین هایی که به هر ورودی، خروجی و متغیر حالت اختصاص یافته را نشان دهید.

ب) ورودی ها و خروجی ها و فیوز های ارتباطی را روی یک کپی از دیاگرام PLS155 علامت گذاری کنید.

نسخه پیش نویس Draft

فصل ۱۰

منطق مختلط

نسخه پیشنویس
Draft

۱۰-۱ مقدمه

جبر بولی و جدول درستی، دو راه برای نمایش روابط منطقی هستند. برای استفاده از این موارد در جهان واقعی، باید راه‌های فیزیکی برای نمایش True و False داشته باشیم؛ مثلاً سوییچ باز و بسته یا سطح ولتاژ. هر یک از این مثال‌ها فقط دو حالت دارند و راه طبیعی برای به قانون در آوردن رفتار این وسایل فیزیکی، جبر بولی است. اگر بیشتر از دو حالت وجود می‌داشت، جبر خیلی پیچیده‌تری نیاز بود تا این حالت‌های چند گانه را به صورت قانون در آورد. در مدارهای الکترونیکی دیجیتالی از سطح ولتاژ برای نمایش حالت‌ها استفاده می‌کنیم.

در اینجا تلاش می‌کنیم تا در مسیر گسترش و بهبود بخشیدن روش‌های سیستماتیک جهت ساختن و بیان مدارات سخت‌افزاری گام برداریم.

برای ساختن مدارهای سخت‌افزاری، از یک سری معادلات و ساختارهای جبری بهره می‌بریم که در این طرح‌ها، سخت‌افزارها به صورت یکسری از نمادها و سمبل‌ها مشخص هستند. به این تصویر اولیهٔ مدار که تمامی اجزای مدار با سمبل‌های خاصی در آن مشخص شده‌اند، دیاگرام مدار می‌گویند.

سابقاً بر ساختن طرح‌های مداری به‌صورت زیر تأکید می‌شد:

۱) بتوانیم از روی جبر و عبارت منطقی یک مدار به یک درک سخت‌افزاری آن مدار دست یابیم.

۲) باید بتوان با آنالیز یک مدار به تابع منطقی آن مدار پی برد (جبر بولی معادل آن مدار کدام است)

دو مرحله بالا بر ساخت و تحلیل مدار تأکید دارند که امروزه در تکنیک‌های طراحی خود را چندان ملزم به پیروی از آن‌ها نمی‌دانیم. در واقع، شرایط بالا یک دیاگرام مداری کامل و واضح را می‌طلبد که هم سخت‌افزار و هم عبارت منطقی را به‌طور کامل شرح داده باشد. حال، چندین مورد از این خصوصیات که برای طراحی یک مدار کامل مورد نیاز است را بررسی می‌کنیم:

۱) طرح مداری عبارت بولی را به‌صورت گیت‌های AND و OR و NOT و... نمایش می‌دهیم.

۲) ارتباط بین متغیرهای منطقی (T, F) با سطح ولتاژ آن‌ها در هر نقطه از مدار می‌بایست مشخص باشد (منطق مثبت یا منفی).

۳) طرح می‌بایست هر جز سخت‌افزاری را به‌طور دقیق و واضح مشخص کند.

اما کلید موفقیت جهت دستیابی به موارد بالا، منطق مختلط است. این ایده به‌طور متمرکز در سال ۱۹۷۱ مطرح شد.

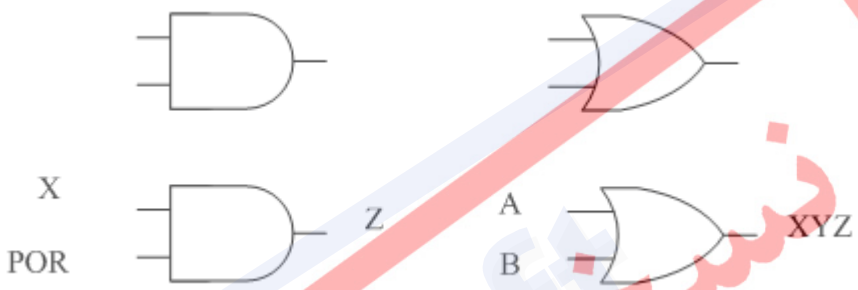
نسخه پیش‌نویس

منطق مختلط در کامپیوترهای TRANSAC استفاده شده است. در این مبحث به تفصیل و تشریح متدهای منطق مختلط می‌پردازیم. قوانین طراحی در این روش مشخص و معین هستند. طراحی آنها به صورت بالا به پایین است.

۱۰ – ۲ نمایش منطق‌ها

ورودی ها در یک سمبل از سمت چپ وارد می‌شود و از سمت راست آن خروجی خارج می‌گردد.

سمبل های زیر بیانگر Z=X.POR XYZ=A+B هستند (و همین طور سمبل های AND,OR)



این سمبل‌های گرافیکی می‌توانند بیش از دو ورودی داشته باشند. به طور مثال، در شکل ۱ گیت AND با ۲و ۵ ورودی نشان داده شده است و شکل ۲ بیانگر گیت OR ۲و۵ ورودی است.

شکل 10-۱ - گیت AND با ۲و ۵ ورودی

شکل 10-۲ - بیانگر گیت OR ۲و۵ ورودی

در یک دیاگرام منطقی، هر سمبل گرافیکی به عنوان نمادی است جهت بیان کردن یک تابع منطقی در کاربردهای مورد نظر.

قطعات سخت افزاری، مدارات مجتمعی هستند که ارزش‌های منطقی آنها توسط ولتاژی که درون سیم‌های مسی رابط وجود دارد، معادل گذاری شده است.

نسخه پیش‌نویس

۱۰-۳ قراردادهای معادل گذاری

چگونه یک وسیله به T,F اشاره می‌کند؟ در وسایل الکترونیکی، دارای دو سطح منطقی و دو سطح ولتاژ معادل با آن هستیم که بنا به نحوهٔ تخصیص دادن، به دو دسته زیر تقسیم می‌شوند:

۱) T توسط H نمایش داده می‌شود (F بواسطه L نشان داده شده است)

۱) T توسط L نمایش داده می‌شود (F بواسطهH نشان داده شده است)

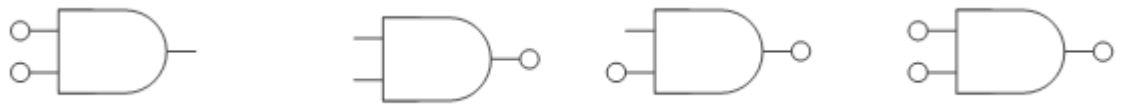
در اولین فرم معادل‌گذاری، گفته می‌شود که از منطق مثبت بهره برده‌ایم (T=H) و در حالت دوم از منطق منفی سود جسته‌ایم (T=L). اگر از یکی از قواعد بالا به طور ثابت استفاده کنیم می‌گوییم از قرارداد منطق مثبت (یا منطق منفی) بهره برده‌ایم. اما منطق مختلط به ما اجازه می‌دهد تا در هر نقطه از مدار، از هر دو معادل‌گذاری بالا استفاده نماییم.

۱۰-۴ نمایش قطعات سخت افزاری حاصل شده با منطق مختلط

عبارت (T=L) توسط یک دایرهٔ کوچک روی ترمینال مربوطه در سمبل منطقی نشان داده می‌شود و عدم حضور این دایرهٔ کوچک بیانگر T=H است. دایره‌ها، مقدار منطقی این عبارت را تغییر نمی‌دهند، بلکه فقط می‌گویند آنها می‌بایست در منطق منفی مورد بررسی قرار گیرند. به عنوان مثال، شکل ۳ نمونه‌ای از بکارگیری تابع AND با استفاده منطق مختلط است. هر کدام از این سمبل‌ها نمایانگر یک قطعهٔ سخت افزاری با جدول درستی زیر است:

جدول 10-۱ – جدول درستی مربوط به یک گیت

LOGIC		
A	B	M
F	F	F
F	T	F
T	F	F
T	T	T



شکل 10-۳ - نمونه‌های مختلف گیت AND در منطق مختلط

و هر کدام از آنها بیانگر نوعی خاص از یک سخت افزار است و از آنجا که هم جدول منطقی و هم جدول معادل ولتاژی را برای آنها داریم، می‌توانیم ماهیت هر یک از آنها را مشخص کنیم.

جدول منطقی مربوط به یک گیت ثابت است و می‌توان آن را به آسانی و سریع نوشت. حال بنا به قرارداد، جدول ولتاژ متناسب با آن را می‌نویسیم و سرانجام با داشتن این دو جدول و رجوع به یک کتابچهٔ راهنما، می‌توان وسیلهٔ مزبور را شناسایی کرد.

جدول 10-۲ – جدول منطقی و جدول ولتاژی برای یک گیت

LOGIC

A	B	M
F	F	F
F	T	F
T	F	F
T	T	T

VOLTAGE

A	B	M
L	L	H
L	H	H
H	L	H
H	H	L

به‌عنوان مثال، در شکل بالا نماد گیت به‌طور کامل رفتار منطقی و ولتاژی آن را بیان می‌کند. آیا حال ما می‌توانیم یک وسیلهٔ فیزیکی پیدا کنیم که با این جدول ولتاژ هم خوانی داشته باشد؟

در واقع شکل به‌صورت یک گیت And است با این تفاوت که تنها یک Logic ندارد و به‌صورت منطق مختلط است. با یک نگاه به کتابچهٔ راهنما مربوط به خانواده TTL، در می‌یابیم که یک چیپ QUAD 74LS00 NAND با دو ورودی رفتار مشابه با جدول‌های به‌دست آمده دارد. حال یک ۰ سمبل برای تشریح عملیات منطقی و قطعهٔ سخت‌افزاری مربوط به آن داریم. کلمهٔ quad در اسم یک چیپ دلالت می‌کند بر اینکه چیپ شامل چهار گیت مشابه است و همینطور triple می‌گوید سه گیت مشابه و همینطور الی آخر. تعداد گیت‌ها، به ازای هر چیپ برآورد می‌شود؛ به واسطهٔ تعداد ورودی‌ها. خروجی‌ها نیز به ازای هر گیت و تعداد پین‌ها روی چیپ.

کلمهٔ NAND در یک تراشهٔ 74LS00 سابقهٔ تاریخی دارد و بیانگر این مطلب است که این تابع، منطق NAND را در منطق مثبت پیاده‌سازی کرده است. منطق مختلط، علاقهٔ چندانی به بکارگیری NAND در بیان عبارت جبری ندارد.

نسخه پیش‌نویس

قطعهٔ مورد استفاده و عمومی دیگر TTL، تحت عنوان تراشهٔ quad 74LS02 NOR با دو ورودی است که آن می‌تواند همانند یک قطعهٔ OR عمل کند. زمانی که در ورودی T=H (منطق مثبت) و در خروجی T=L (منطق منفی) در نظر بگیریم، نام مدار OR است. همچنین به طریقی مشابه، از مدارات ترکیبی با منطق مثبت استخراج شده در جایی که این قطعه به عنوان یک تابع NOT OR منطقی عمل می‌کند.

جدول 10-۳ – جدول منطقی و جدول ولتاژی برای یک گیت

LOGIC

P	Q	R
F	F	F
F	T	T
T	F	T
T	T	T

VOLTAGE

P	Q	R
L	L	H
L	H	L
H	L	L
L	H	L

۱۰ – ۵ نام متغیرها در منطق مختلط

در یک مدار سخت‌افزاری ولتاژها بیانگر ارزش متغیرهای منطقی هستند در واقع اشاره ما به یک سطح ولتاژ خاص بیانگر مقدار ارزش یک متغیر منطقی است همانگونه که ما مقادیر ورودی و خروجی را با نام متغیرهای منطقی نامگذاری می‌کنیم نیاز به یک قانون کلی داریم که سطح ولتاژ انتساب داده شده برای هر ارزش یک متیغر منطقی را شرح دهد ما قرارداد زیر را برای نامگذاری سیگنال‌ها و بیان ارتباط آن‌ها با متغیرهای منطقی بیان می‌داریم.

۱) اگر سیگنال به‌صورت T=L فرض شود L. بیانگر یک متغیر منطقی است.

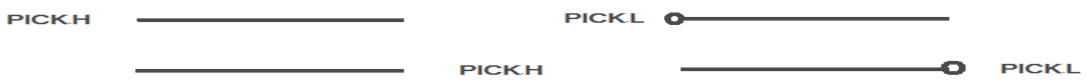
۲) اگر سیگنال به‌صورت T=H فرض شود H. بیانگر یک متغیر منطقی است.

(T=L یعنی مقدار ارزش درستی یک عبارت منطقی با سطح ولتاژ پایین نشان‌داده می‌شود اگر سطح ولتاژ پایین بود ارزش منطقی یک متغیر درست است)

به‌عنوان مثال متغیر منطقی Pick می‌تواند در یک حلقه هم به صورت سیگنال Pick.L و هم به صورت Pick.H ظاهر شود بنا به مقدار ولتاژ سیگنال.

نسخه پیش‌نویس Draft

نماد کلی برای L به‌صورت خطی است که همراه با یک دایره و H نیز تنها به‌صورت یک خط نمایش داده می‌شود.



دو نکته مهم که باید به آن توجه شود عبارتند از :

۱) در هر دو شیوه نمایش متغیر منطقی هیچ تغییری نمی‌کند. L و H بودن تنها بیانگر تفاوت در فرم فیزیکی مربوط به یک متغیر منطقی است.

۲) L به‌معنای ولتاژ پایین نیست بلکه آن معنی می‌دهد که اگر ولتاژ پایین است آنگاه ارزش متغیر منطقی درست است (منطق منفی) و به‌طرز مشابه برای H

عموماً یک دیاگرام مداری شامل هر دو سیگنال برای یک متغیر منطقی است و ما ولتاژ قراردادی و اسم سیگنال‌ها را روی سیم‌ها در یک دیاگرام مداری می‌بایست مشخص کنیم. ممکن است شما فکر کنید که اینکار زائد است اما باید بدانید که برای اینکه مدار واضح و مشخص باشد اینکار ضروری است اگر شما خطوط و نماد سیگنال‌ها را به‌طور کامل نمایش دهید آنگاه دیاگرام مداری شما نمایانگر تمامی اطلاعات ولتاژی و منطقی در مدار خواهد بود به‌صورت واضح (و البته قراردادی)

هرگاه ضرورت ایجاب کرد که نام یک متغیر منطقی یا سیگنال بیش از یکبار در یک طراحی ذکر شود می‌توان به استفاده از نمادها برای سیگنال‌ها رو آورد و این مورد زمانی روی می‌دهد که سیگنال در بیشتر از یکی از صفحات دیاگرام مداری خود را نشان می‌دهد هنگامی که طراح یک لیست کلی از نام سیگنال‌های بکار رفته در مدار را مشخص می‌کند (و همین‌طور شرح مختصری در مورد آن سیگنال‌ها)به‌عنوان مثال در شکل زیر سیگنالی که در یک صفحه از دیاگرام مداری وجود دارد به‌عنوان ورودی برای یک مکان دیگر تلقی می‌شود با یک نمادگذاری مناسب هرگز هیچ‌گونه تردیدی در مورد تشخیص درست سیگنال‌ها از هم پیش نخواهد آمد.

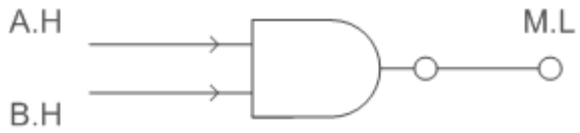
نسخه پیش‌نویس

نامگذاری سیگنال‌ها بسیار مهم است یک نمادگذاری ویژه برای ایجاد تمایز بین دو سیگنال برای یک متغیر همواره چندان قطعیتی در مقیاس عمومی ندارد چرا که هرکس متناسب با شیوه خود به نمایش این سیگنال‌ها می‌پردازد به‌عنوان مثال، برخی از مردم ترجیح می‌دهند که علاماتی چون + و − و ↑ و ↓ را همراه با نام متغیر بکار برند و برخی دیگر از یک انتساب و بیان آشکار چون T=L استفاده می‌کنند اما در این کتاب ما دو سیگنال مربوط به یک متغیر منطقی را با H. و L. نمایش خواهیم داد.

۱۰ – ۶ خاصیت دوگانی AND, OR

نکته منفی این دو منطق این است که به دلیل وجود قانون دمورگان، تبدیل یک عبارت منطقی به مدار متناظر یا نوشتن عبارت منطقی یک مدار کار راحتی نیست. مخصوصا در مورد گیتهای NAND و NOR که باعث پیچیده شدن کار می‌شود. برای حل این مشکل منطق سومی پیشنهاد شده است که از ترکیب دو منطق قبلی است و به منطق مختلط[1] مشهور است. در این منطق هر سیگنال می‌تواند در منطق منفی یا منطق مثبت باشد و قوانینی که در ادامه اشاره می‌شود، به ما اجازه می‌دهد که براحتی مدارات و عبارتهای منطقی را بیکدیگر تبدیل نماییم. این منطق برخلاف دو منطق دیگری فقط صوری است و مدار واقعی همیشه در منطق مثبت یا منفی کار می‌کند. اما می‌تواند در موارد زیر به ما کمک منطقی راحت به مدار برسیم.از مدار راحت به عبارت منطقی برسیم و بالعکس.

[1] Mixed Logic



LOGIC

A	B	M
F	F	F
F	T	F
T	F	F
T	T	T

≡

VOLTAGE

A	B	M
L	L	H
L	H	H
H	L	H
H	H	L

LOGIC

P	Q	R
F	F	F
F	T	T
T	F	T
T	T	T

≡

VOLTAGE

P	Q	R
H	H	L
H	L	H
L	H	H
L	L	H

نسخه پیش‌نویس
Draft

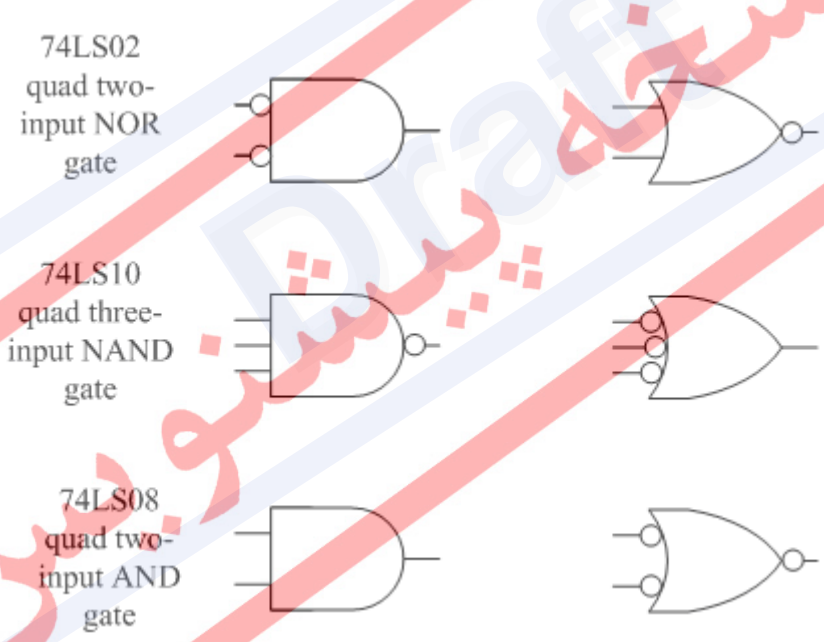
شکل 10-۴ - پیاده سازی قطعه 74LS00 با گیت OR و جداول ولتاژ و منطقی آن

در اینجا مثالی را بررسی می‌کنیم (با تابع منطقی OR)

با بررسی جدول ولتاژ درمی‌یابیم آن معادل یکی از سمبل‌های مثال قبلی است و از اینرو دوباره بیانگر قطعه سخت‌افزاری 74LS00 است ولی اینبار با گیت OR پیاده سازی شده است در این استفاده از تابع منطقی OR مقدار T = L در هر دو ورودی و T = H در خروجی فرض شده است. لذا می‌توان گفت

ما قطعه 74LS00 را با دو تابع منطقی OR و AND پیاده سازی کرده‌ایم. این دوگانی بین AND و OR همواره وجود دارد. این بواسطه خاصیت دوگانی آنها در علم جبر است. در واقع یک AND و OR با خاصیت دوگانی دارای دایره‌های معکوس و در نقطه مقابل هم هستند. به طرز مشابه می‌توان از قواعد منطق مختلط در بدست آوردن مابقی قطعات سخت‌افزاری نیز بهره ببرید. در شکل ۵ چند قطعه مرسوم و کاربردی را رسم کرده‌ایم و همینطور سمبل‌های متناظر با منطق مختلط آنها نیز نشان داده شده است.

در عمل پیدا کردن تابع منطقی مورد استفاده برای یک تراشه که از منطق مختلط استفاده می‌کند کار ساده‌ایست از اینرو که در دفترچه‌های راهنما سمبل یک گیت به عنوان نمونه داده می‌شود. و برای پی بردن به توابع منطقی بقیه تراشه‌ها می‌توان از جابجایی و خواص دوگانی بهره برد. هر چند برای پی بردن به عمل منطقی پیاده سازی شده توسط یک گیت نام آن کفایت می‌کند. بسیاری از تراشه‌های مداری مجتمع که با AND و OR پیاده سازی شده‌اند به ابزار طراحی انعطاف پذیر و قدرتمندی جهت پیاده سازی توابع منطقی خود را معرفی کردند.

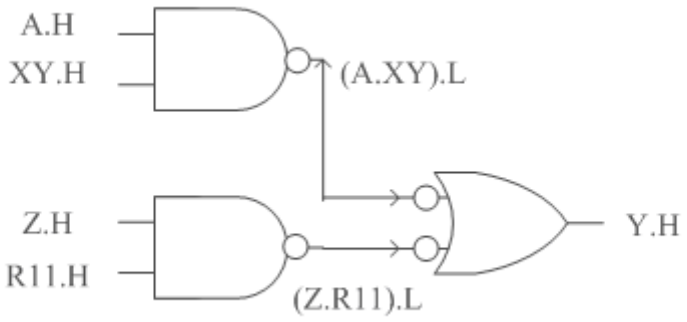


شکل 10-۵ - سمبل‌های متناظر با منطق مختلط برای چند قطعه مرسوم

چند مثال از مدارات ساده:

برای روشن شدن موضوع اجازه بدهید مدار شکل ۶ را مورد بررسی قرار دهیم.

نسخه پیش‌نویس Draft



شکل 10-۶ - مدار ترکیبی با منطق مختلط

a) دو قطعه سخت‌افزاری 74LS00 وجود دارد که تابع منطقی آنها مشابه یک AND است که در ورودی T = H و در خروجی T = L است.

b) یک قطعه سخت‌افزاری دیگر وجود دارد (74LS00) که تابع منطقی آن به صورت یک تابع OR است که در ورودی‌ها T = L و در خروجی T = H است.

c) هفت قطعه سیم مسی وجود دارد که برای هدایت ولتاژ برای متغیرهای منطقی A , XY , Z , R11 , (A.XY) , Y , (Z.R11)

d) ارزش درستی توسط سطح بالای ولتاژ ی در عبارات A.H، XY.H و Z.H و RN.H و ۲.H نشان داده می‌شود.

e) ارزش درستی در عبارات (A*XY).L و (Z*RH) .L سطح پایین ولتاژ نمایش داده شده است.

f) این مدار تابع منطقی $Y = A * XY + Z * RH$ را پیاده‌سازی می‌کند.

حال شکل 2.7 را درنظر می‌گیریم.

a) یک قطعه سخت‌افزاری (74 L500) وجود دارد که تابع منطقی پیاده‌سازی شده توسط آن مشابه یک تابع OR است. هنگامی T=L در ورودی و T=H در خروجی داشته باشیم.

b) یک قطعه سخت‌افزاری دیگر (دوباره 74LS00) نیز وجود دارد که مشابه یک تابع AND عمل می‌کند هنگامی که T=H در ورودی‌ها و T=L در خروجی باشد.

c) پنج قطعه سیم مسی حامل ولتاژ برای متغیرهای منطقی P و R_2 و $(P+R_2)$ و AB و A_3 داریم.

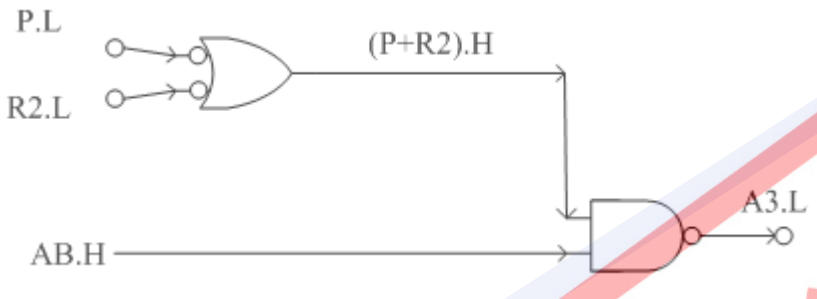
d) مقدار ولتاژی معادل گذاری شده برای متغیرهای منطقی درست H. $(P+R_2)$ و H. (AB) برابر سطح بالای ولتاژ است.

نسخه پیش‌نویس Draft

e) مقدار ارزش درستی در متغیرهای P.L و R_2.L و A_3.L با سطح پایین ولتاژ نمایش داده می‌شود.

f) مدار تابع $A_3=AB*(P+R_2)$ را پیاده‌سازی می‌کند.

شکل‌های ۶ و ۷ هر دو بیانگر یک طرح کلی از مدارات ترکیبی که از قواعد منطق مختلط پیروی می‌کند هستند.

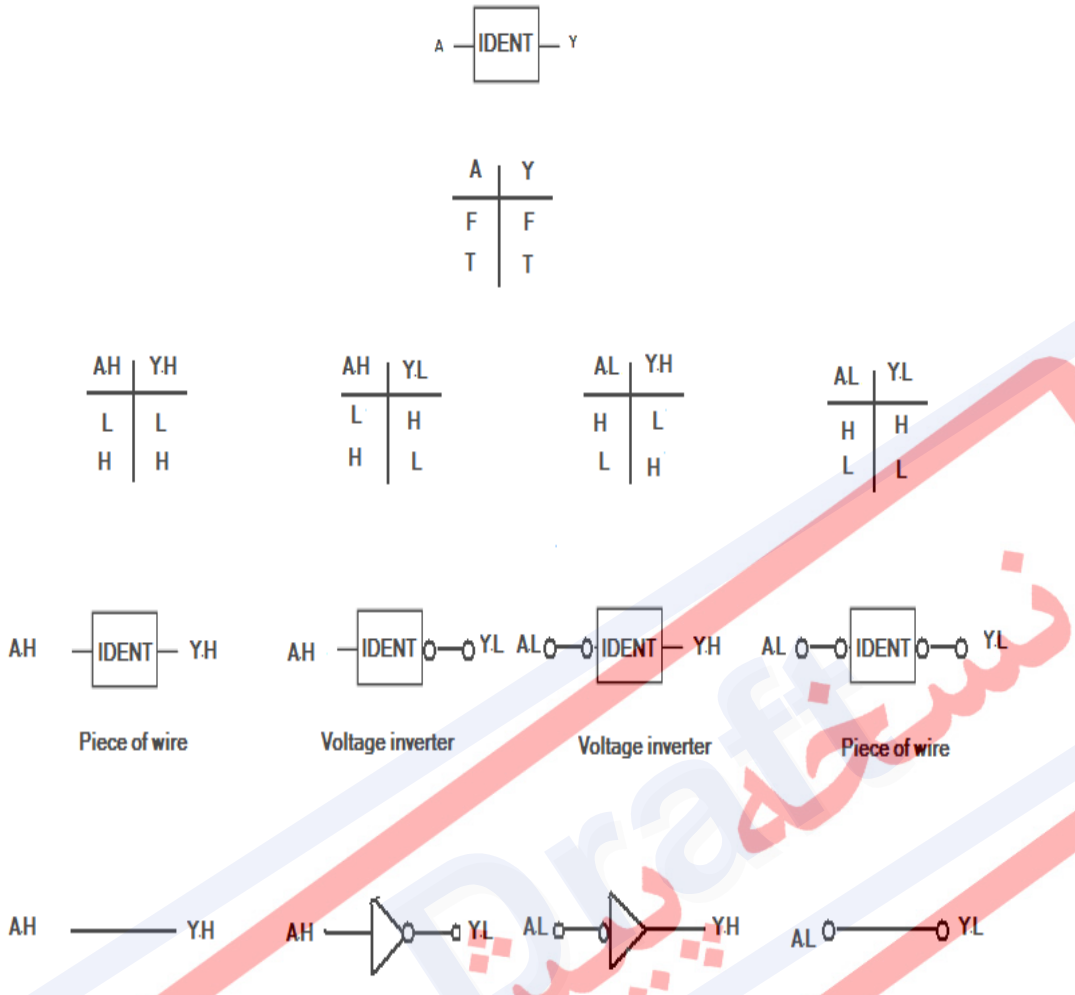


شکل 10-۷ - مدار ترکیبی با منطق مختلط

۱۰-۷ نظریه منطق مختلط

با این مقدمه و نشانه‌گذاری‌های مقدماتی اجازه دهید مقدمات تئوری منطق مختلط را بررسی کنیم (شکل ۸) به شکل ۸ نگاه کنید یک متغیر منطقی تحت عنوان A وارد یک جعبه می‌شود که یک سری اعمال منطقی روی آن صورت می‌گیرد و نهایتاً خروجی Y حاصل می‌شود (به گونه‌ای که Y=A) جدول منطقی برای این عمل تنها به یک صورت وجود دارد که در شکل مشخص است اما حال ما می‌توانیم چهار جدول ولتاژ برای آن در نظر بگیریم به صورت (A.H,Y.H) و (A.H,Y.L) و (A.L,Y.H) و (A.L,Y.L) که هر کدام از این موارد یک دیاگرام مداری خاص را ارائه می‌دهد.

نسخه پیش‌نویس Draft



شکل 10-۸ - یک جعبه با عمل Y=A

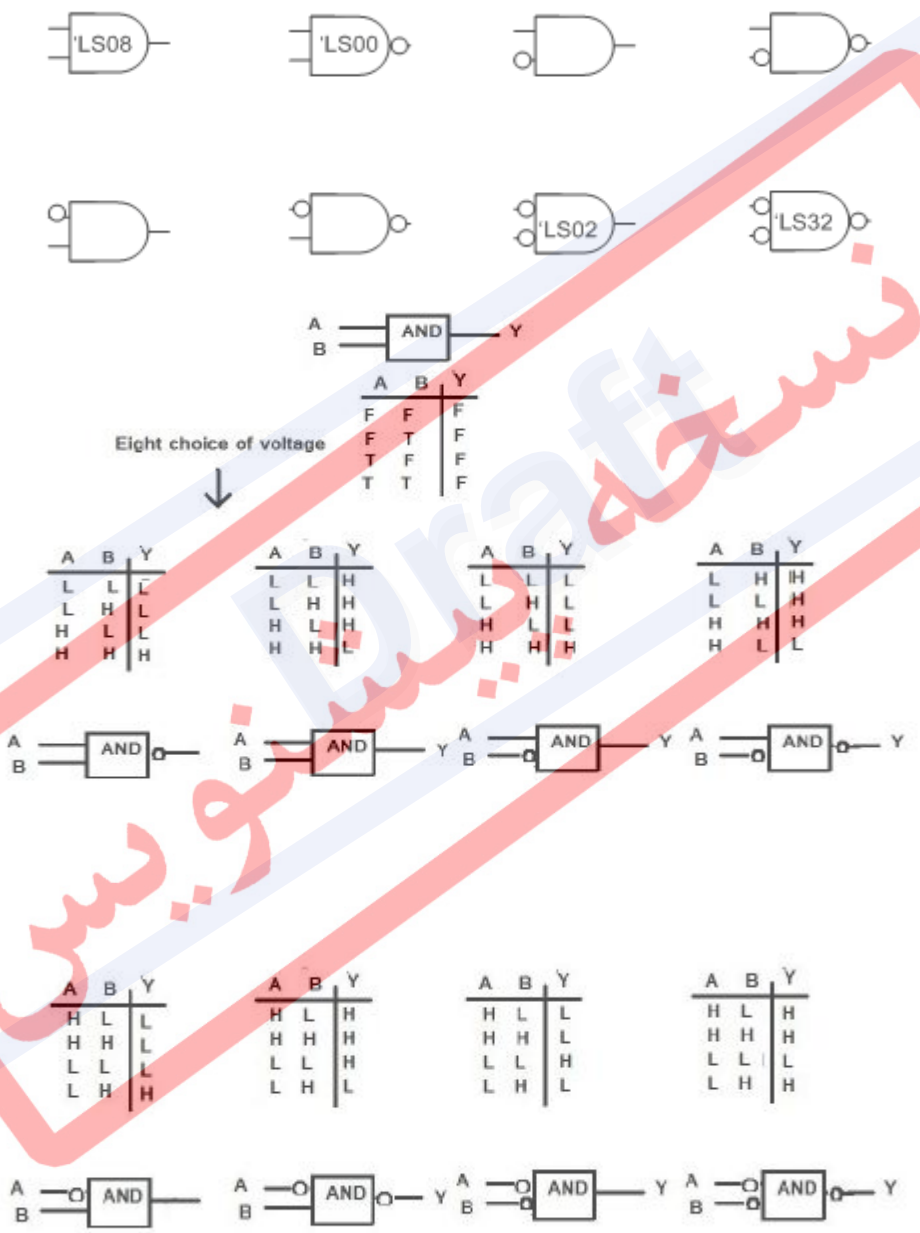
طراحی تراشه قبلی نیازمند درک چهار جدول ولتاژ است که وابسته به چهار دیاگرام ما است. هنگامی که A.H نتیجه می‌دهد Y.H را ما تنها به یک سیم نیازمندیم از این رو ولتاژ هیچگونه تغییری در گذر مسیر خود ندارد و به طرز مشابه هنگامی که A.L منجر به Y.L می‌شود ما تنها به یک سیم نیاز داریم ولی از سوی دیگر جدول ولتاژ برای A.H و Y.L بیانگر یک معکوس کننده ولتاژ است به عنوان نمونه گیت معکوس کننده 74LS04.

برای A.L و Y.H یک مثلث با یک دایره تنها روی ورودی یا خروجی یک نشانه مرسوم برای بیان معکوس کننده است. حال ما چهار مسیر اجرایی برای عملیات خود داریم دو تا از آنها نیاز به یک تکه سیم دارند و دوتای آنها یک گیت معکوس کننده نیاز دارند. در منطق مختلط ما از هر چهارتای آنها

نسخه پیش‌نویس Draft

استفاده می‌کنیم. یک ایده منطقی که بیانگر کمینه بودن قطعات سخت‌افزاری به کار رفته در یک مدار است استفاده از مسیر سیمی را پیشنهاد می‌کند. حال اگر طرحی یک تغییر در سطح ولتاژ را بطلبد بدون اینکه تغییری در منطق ما حاصل آید راه کارهایی توسط منطق مختلط ارائه خواهد شد.

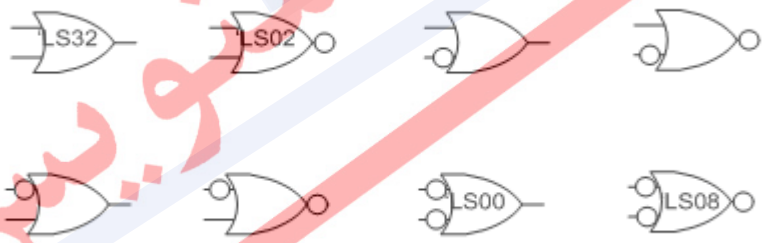
حال یک مدار AND منطقی را در نظر می‌گیریم که در شکل ۹ بیان شده است. دو متغییر منطقی A و B به عنوان ورودی و خروجی حاصل از آن به فرم Y = A.B.



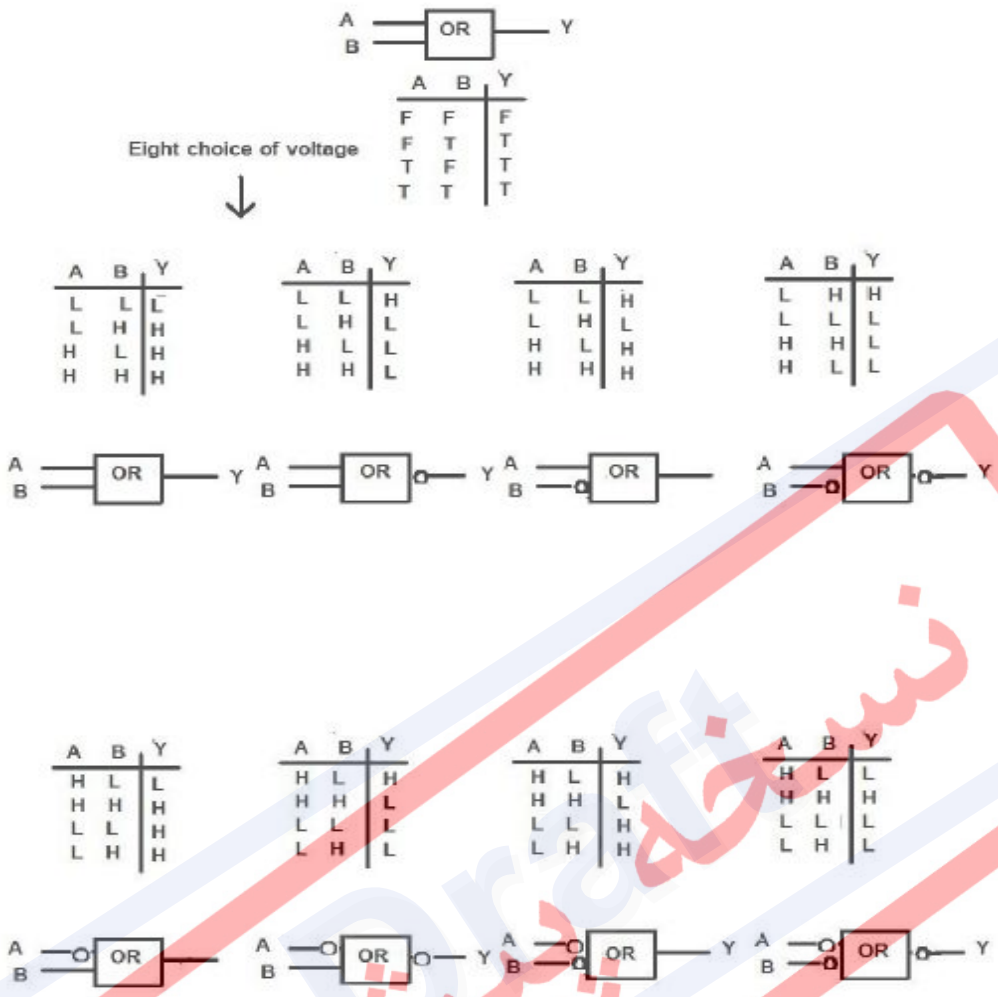
شکل 10-۹ - گیت AND

نسخه پیش‌نویس Draft

با وجود دو ورودی و یک خروجی ما می‌توانیم هشت حالت مختلف را برای معادل گذاری سطح ولتاژ آنها مطرح کنیم. هشت جدول ولتاژ بدست می‌آید و هشت تابع انتساب که هر انتساب بیانگر یک دیاگرام متفاوت نسبت به قبلی است. دایره روی سنبل بیانگر استفاده از منطق منفی است. حال با رجوع به یک کتابچه راهنما درمی‌یابیم که از این هشت حالت چهار مورد آن به عنوان قطعات سخت‌افزاری در بازار موجود است که عبارتند از 74LS08 AND هنگامی که T = H باشد، 74LS00 NAND هنگامی که T = L در خروجی باشد، 74LS02 NOR هنگامی که T = L در ورودی باشد و نهایتاً 74LS32 OR هنگامی که هم ورودی و هم خروجی در منطق منفی (T = L) باشند. در مثال بالا دیدیم که منطق مختلط چهار ساختار بلاکی مفید را برای گیت AND معرفی کرده است. وجود بقیه مدارات از لحاظ تئوری درست و مجاز است ولی از لحاظ عملی ساخت آنها مشکل است و عملاً در ساخت و طراحی مدارات به کار نمی‌رود. حال می‌توان به قدرت منطق مختلط پی برد. در جایی که منطق مثبت تنها به یک شکل اجازه استفاده می‌دهد (74LS08) و منطق منفی نیز وجود تنها یک گیت را مجاز می‌داند (74LS32) منطق مختلط انعطاف بیشتری از خود نشان داده و چهار ساختار گیتی را معرفی می‌کند. مدار گیت OR نیز در شکل ۱۰ به طور مشابه رفتار یکسان با گیت AND مطرح شده در بالا دارد. دوباره هشت جدول برای ولتاژها و هشت دیاگرام مداری متفاوت ایجاد خواهد شد و حال رجوع به یک کتابچه راهنمای TTL به ما می‌گوید که 74LS32 OR مانند یک گیت AND رفتار می‌کند زمانی که T = L در ورودی و خروجی فرض شود و مانند یک گیت OR است هنگامی که در هم ورودی و هم خروجی T = Hاست.



شکل 10-۱۰ - گیت OR



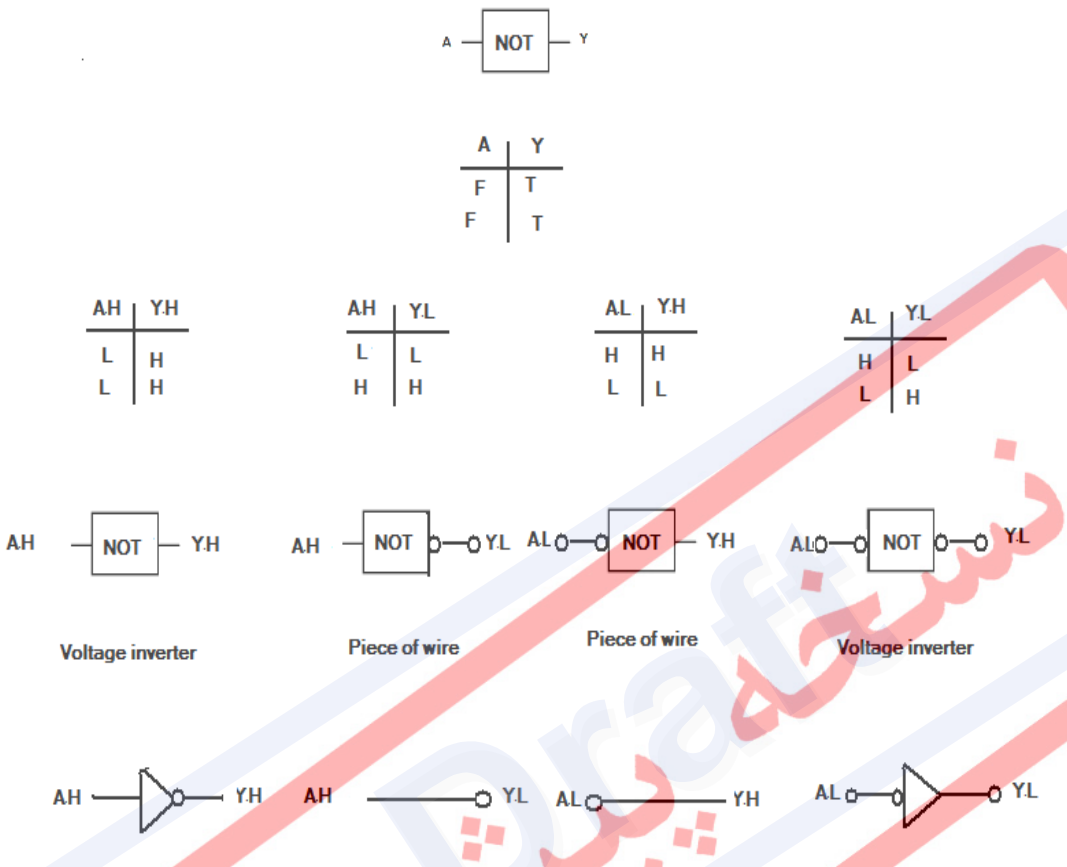
شکل 10-۱۰ – (ادامه)

به‌طور مشابه در 74LS02 NOR به‌صورت OR منطقی عمل می‌کند وقتی که T=L روی خروجی باشد و در 74L500 NAND هنگامی که T=L در هر دو ورودی باشد و همینطور 74LS08 AND هنگامی که T=L در ورودی و خروجی باشد.

منطق مختلط به ما چهار مورد آسان جهت پیاده‌سازی گیت OR منطقی می‌دهد. مشابه گیت AND چهار مورد دیگر نیز از لحاظ تئوری درست هستند و مجاز به شمار می‌آیند اما به‌آسانی به‌وسیله مدارات مجتمع قابل دسترسی نیست و این در حالی است که در منطق مثبت و منطق منفی تنها تراشه‌های 74LS32 و 74LS08 (به‌ترتیب) مجاز هستند و سرانجام اجازه دهید به گیت NOT نیز نگاهی بیاندازیم در این مورد مقدار خروجی برابر مقداری نقیض ورودی است جدول منطقی همانگونه که نشان می‌دهد مقدار ورودی و خروجی نقیض یکدیگر به شمار می‌آیند حال می‌توان با استفاده از جدول منطقی چهار

نسخه پیش‌نویس Draft

جدول ولتاژ برای این جدول ساخت به‌صورت (A.H, Y.H) و (A.H , Y.L) و (A.L, Y.H) و (A.L , Y.L)



شکل 10-۱۱ - گیت NOT

که متناسب با آن به چهار دیاگرام خواهیم رسید که این چهار جدول ولتاژ به‌عنوان یک (به ترتیب) معکوس‌کننده ولتاژبه‌صورت یک تکه سیم، یک تکه سیم و یک معکوس‌کننده ولتاژ هستند. این نکته جالب به‌نظر می‌رسد که در منطق مختلط می‌توان معکوس‌کننده ولتاژ را با یک سیم نمایش داد. مطلبی که برای دسترسی به این نکته ضروری است همانگونه که از شکل مشهود است سطح ولتاژ دو طرف سیم می‌بایست با هم فرق داشته باشد در جائیکه منطق مثبت و منفی تنها یک وسیله در جلوی پای ما می‌گذارند Mixed logic این توانایی را دارد که چهار مورد پیشنهاد کند دو مورد با استفاده از قطعات سخت‌افزاری دو مورد تنها با استفاده از یک سیم رابط بین ورودی و خروجی.

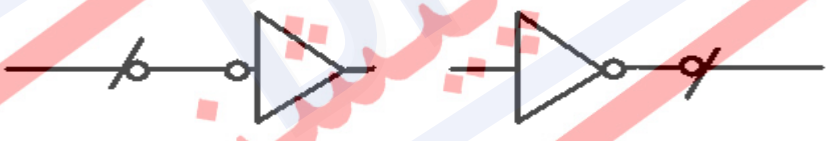
در موردی که ما با استفاده از یک سیم که ابتدا را به انتها وصل کرده است یک معکوس‌کننده می‌سازیم زمانی است که می‌خواهیم سطح ولتاژ در طی یک سیم عوض شود اما منطق مربوطه تغییری

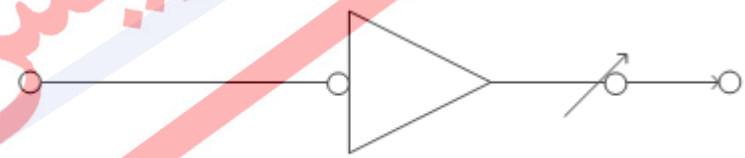
نسخه پیش‌نویس Draft

پیدا نکند یعنی $A.L \rightarrow A.H$ در هویت منطقی for free در طی یک سیم مقدار ولتاژ در دو سر یک سیم یکسان است حال ما به یک معکوس‌کننده for free دست یافته‌ایم که مقدار معادل ولتاژی یک متغیر منطقی در دو سر یک سیم با هم برابر نیستند.

اما شکل نمایش برای این گیت به‌صورت یک اسلش و یک دایره است که در آن اسلش بیانگر این است که مقدار معادل ولتاژی یک متغیر برای کدام نقطه تغییر کرده است و در ضمن سمت نقطه‌ای که این اتفاق روی داده است یک دایره قرار داده می‌شود.

هنگامی که ما به یک مقدار معادل ولتاژی برای یک متغیر نیاز داریم که در ورودی و خروجی یکسان باشد، که مقدار منطقی خروجی عکس حالت ورودی باشد نه شکلی مانند شکل زیر بهره می‌بریم.



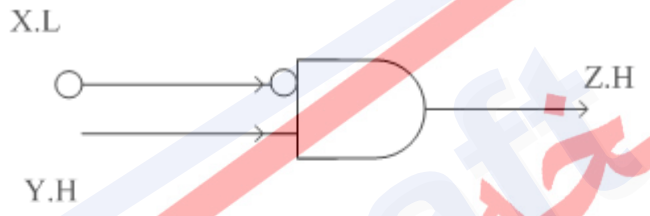


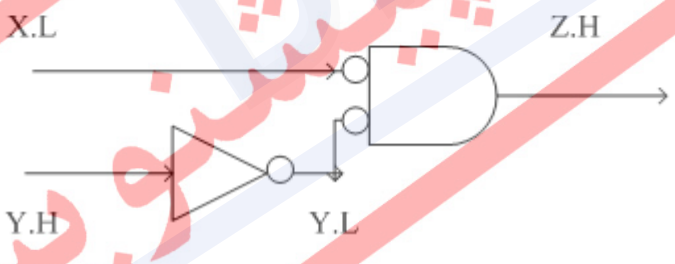
OR

نسخه پیش‌نویس Draft

در عمل عموماً از وجود دایره بیانگر T=L بر روی اسلش یک معکوس‌کننده منطقی خودداری می‌کنیم. چرا که عملاً به ما اطلاعات چندانی در مورد سیم مرتبط با آن تا سر پایانی آن ارائه نمی‌دهد و در این کتاب ما از نماد اسلش بدون دایره استفاده می‌کنیم.

دوباره تأکید می‌کنیم یک معکوس‌کننده ولتاژ هیچ‌گونه تغییری در منطق یک عبارت نمی‌دهد A.H→A.L و یک متغیر منطقی یکسان در دو سر سیم بکار برده می‌شود، جهت جلوگیری از بروز تشابه اسمی بین معکوس‌کننده ولتاژ و معکوس‌کننده منطقی برای معکوس‌کننده ولتاژاز عبارت "OOPS" استفاده می‌کنیم برای یک "oops" ما نیاز داریم سطح ولتاژ را برای یک متغیر منطقی عوض کنیم. اجازه دهید تابع $Z = X.Y$ را با ورودی‌های X و Y به‌صورت X.L و X.H پیاده‌سازی کنیم ما می‌بایست یک قطعه سخت‌افزاری فراهم کنیم مطابق شکل زیر عمل کند.



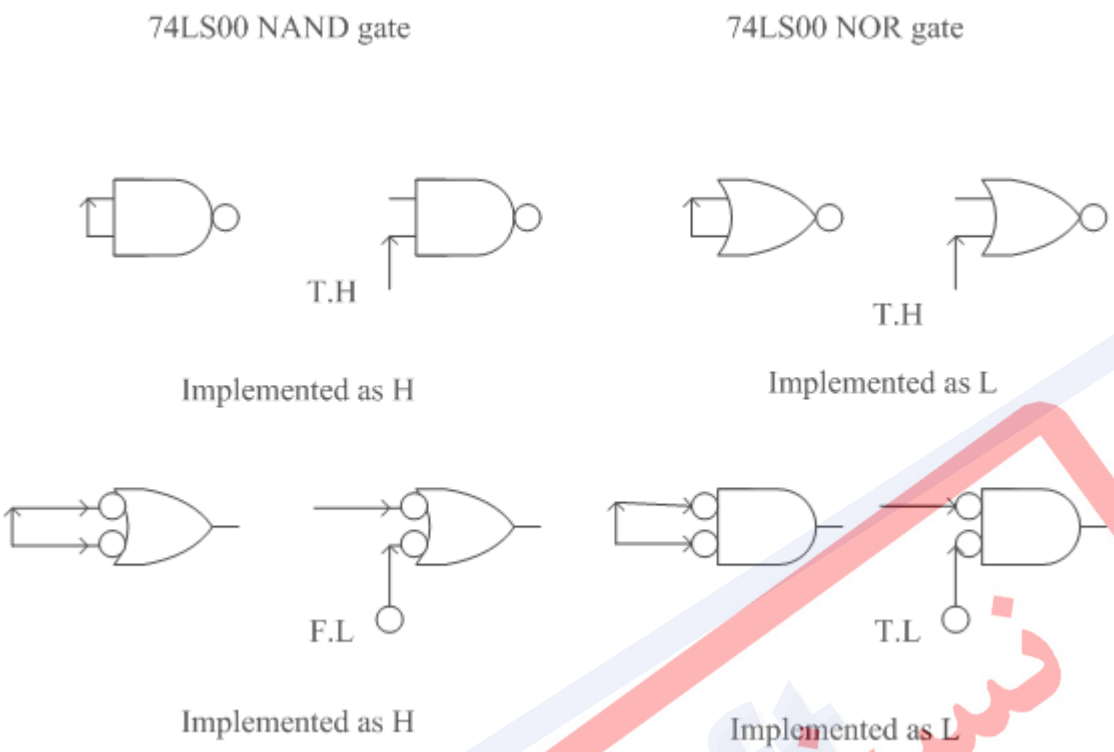


هیچ مدار موجود در بازار به‌صورت بالا عمل نمی‌کند اما با استفاده از یک 74LS02 و بکارگیری آن در مدار بالا صورت زیر عملکرد مشابه مدار بالا را خواهیم داشت.

ما یک معکوس‌کننده ولتاژ را برای تغییر سیگنال Y.H به Y.L در ورودی AND بکار برده‌ایم.

یک معکوس‌کننده 74LS04 تنها راه جهت معکوس کردن ولتاژ نیست استفاده از توابع NAND و NOR نیز راه‌های مختلفی پیش‌روی ما می‌گذرد که همان عملکرد را نتیجه می‌دهد همانگونه که در شکل ۱۲ نشان داده شده است

نسخه پیش‌نویس Draft



شکل 10-۱۲ - معکوس‌کننده

شما می‌توانید به این عملکرد با استفاده از خصوصیات توابع OR, AND و ترکیب آنها برسید. این مطلب بیانگر کامل بودن منطق مختلط است حال به کمک آن ما نشانه‌هایی داریم که به واسطه آن‌ها می‌توانیم مقدار معادل ولتاژی را برای متغیرها نمایش دهیم و با آن می‌توانیم هر وسیله سخت‌افزاری را با یک نماد و سمبل نشان دهیم ما توانسته‌ایم مرز بین عبارات منطقی و ولتاژ معادل آن‌ها را مشخص کنیم (آن‌ها را نسبت به هم مستقل کنیم) یعنی هر گونه ارزش درستی با هر معادل ولتاژی می‌تواند در آن باشد این نشانی‌گذاری‌ها و تئوری می‌تواند برای ساختن بلاک‌های دیجیتالی پیچیده‌تر مورد استفاده قرار گیرد. ما حال با دانش خود می‌توانیم به تحلیل و سنتز مدارات با منطق مختلط بپردازیم.

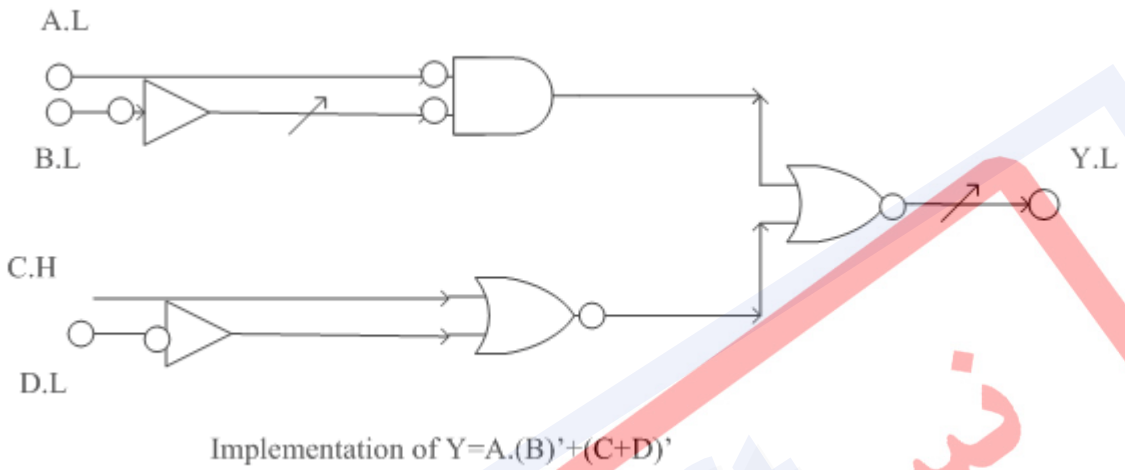
۱۰-۸ آنالیز مدارات با منطق مختلط

مسئله عمده در طراحی مدارات و طراحی دیاگوم آن‌ها باید به‌گونه‌ای باشد که فردی که در حال بررسی یک طرح است بتواند معادله جبری اصلی را مدنظر طراح آن مدار بوده است بازیابی کند. مدارات با منطق مختلط شرط زیر را برآورده می‌کنند آنالیز یک مدار با منطق مختلط بسیار ساده است اما نکته‌ای که وجود دارد نادیده گرفتن دایره‌ها و معکوس‌کننده‌ها است هر اسلش به‌عنوان یک Not منطقی

نسخه پیش‌نویس

شمرده می‌شود و سمبل‌های AND و OR نیز به صورت تابع AND و OR مورد تحلیل واقع می‌شود و سرانجام با درنظر گرفتن موارد بالا رسیدن به معادله نهایی.

برای نمونه خواندن مدار شکل ۱۳ منجر به یافتن معادله منطقی $Y = A^* \overline{B} + \overline{(C+D)}$ می‌شود.



$$Y = A^* \overline{B} + \overline{(C+D)}$$

شکل 10-۱۳ – مدار برای معادله منطقی

به‌عبارت دیگر به این معناست که Y معادل است با A و نقیض B یا نقیض (C یا D) اگر شما احساس می‌کنید نیاز دارید به یکسری اطلاعات اضافی در دیاگرام مداری می‌توانید اسامی سیگنال‌های میانی و دایره‌های گمشده روی اسلش را نیز قرار دهید شما قادر خواهید بود سریعاً مهارت خود را در خواندن مدارات منطق مختلط بدون این موارد فهرست علائم نیز بیابید.

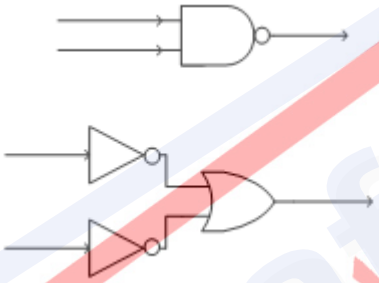
سنتر مدارات منطق مختلط :

برای پیاده‌سازی یک معادله منطقی داده شده شروع کنید به طرح‌ریزی یک تصویر اولیه از آن معادله با استفاده از AND و OR و اسلش برای Not منطقی.

ورودی‌ها و خروجی‌ها را با اسامی مناسب برای متغیرهای منطقی نام‌گذاری نمایید در این نقطه تأکید بر منطق‌ها است که در این موارد یک ساختار مقدماتی و پایه‌ای به ما می‌دهد که به واسطه آن می‌توانیم مدار کامل شده را تحلیل کنیم و اگر مقادیر ورودی یا خروجی یک سطح ولتاژی ثابت برای متغیر منطقی داشتند مقادیر متناسب با آن‌ها (H. یا L.) و دایره مربوط را اضافه می‌کنیم این مرحله برخی از متغیرهای منطقی را به سیگنال‌ها تبدیل می‌کند و معادل ولتاژی را برای برخی از خطوط ثابت نگه می‌دارد.

شما ممکن است فرض کنید که متغیرهای منطقی بدون تمثال‌های شرح داده شده توسط هر دو صورت سیگنال (L, H) قابل دسترسی است و شما می‌توانید از این انعطاف‌پذیری در سنتز مدارات خود استفاده کنید.

عموماً ابزار در دسترس برای پیاده‌سازی عملیات AND و OR محدود خواهند شد که با محدودیت سخت‌تر، آسان‌تر شدن سنتز مدار حاصل می‌آید به‌عنوان مثال برای نمونه اگر شما تنها به یک تراشه 74LS00 NAND دسترسی داشته باشید سنتز مدار آشکار و واضح خواهد بود اگر چه معادله جبری مربوط به آن بسیار پیچیده باشد ولی در این مورد ما هیچ‌گونه انعطاف‌پذیری نخواهیم داشت و تنها نشانه‌های قابل دسترسی عبارتند از:



در این قطعه به تنهایی کافیست. با این وسیله ما هم یک گیت OR داریم و هم یک گیت NOR و اگر نیاز به یک معکوس‌کننده ولتاژ داشتیم می‌توانیم آن را نیز روی مدارات بالا به‌دست آوریم و به NOT منطقی نیز نیازی نداریم.

اما با داشتن بلاک‌های سازنده مختلف و اینکه خود را محدود به تنها یک گیت خاص نکنیم ما قدرت انعطاف بیشتری خواهیم داشت. ما می‌خواهیم مدار خود را بهینه سازیم.

یک عامل مهم و واضح در بهینه‌سازی کمینه کردن تعداد معکوس‌کننده‌های ولتاژ است اما فاکتورهای دیگر اغلب نسبی هستند مانند استفاده زیادی از گیت‌ها در یک تراشه هنگامی که مقادیر گوناگونی از یک تراشه موجود است طراح ممکن است ذوق و سلیقه خود را در برآورده کردن و پیاده‌سازی یک معادله بکار برد.

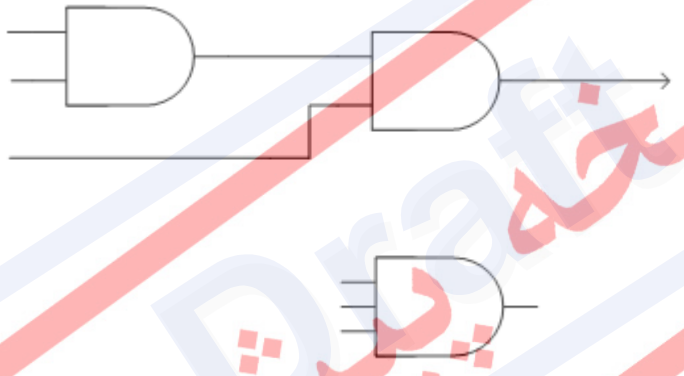
طراحی یک مدار که بتواند یک معادله را پیاده‌سازی کند بسیار آسان است و ایجاد یک طراحی زیبا و منحصر به فرد هنر است با این حال فرآیند کلی صریح و بی‌پرده است انتخاب یک تراشه مشابه برای یک گیت قرار دادن نشانه‌های منطق مختلط مربوطه به‌وسیله اضافه کردن دایره‌های مورد نیاز برای نشانه‌های منطقی و اضافه کردن معکوس‌کننده‌های ولتاژ در جایی که نیاز است به ساختن دایره‌های جهت اینکه

نسخه پیش‌نویس Draft

مدار با مقادیر منطقی مربوطه مطابقت کند و سرانجام حرکت به سمت یک عنصر بعدی و ادامه روال بالا برای آن عنصر.

در طی حرکت به جلو در سنتز یک مدار ممکن است شما پی ببرید که با بکار بردن یک تراشه دیگر در مرحله قبلی مدار بهینه‌تر خواهد بود به‌همین دلیل ممکن است شما به عقب بازگردید و از نو آغاز کنید در طی یک زمان کوتاه سرانجام ما به یک نقطه مشترک می‌رسیم که همان طرح کلی مدار خواهد بود.

توابع AND و OR مربوط به بیش از دو متغیر می‌تواند با دو یا چند گیت دو ورودی یا یک گیت با ورودی‌های بیشتری مشخص شود به‌عنوان مثال هر دو فرم در شکل ۱۴ مشخص شده است که بیانگر یک گیت ۳ ورودی است شرایط طراحی فرصتی جهت انتخاب بین آن دو را فراهم می‌آورد.



شکل 10-۱۴ – گیت AND ۳ ورودی

۱۰-۸-۱ قراردادهای طرح مدار

دیاگرام‌های یک مدار نشان‌دهنده منطق سیستم‌های دیجیتالی است. آن‌ها همچنین سخت‌افزارها را نیز مشخص می‌کنند.

اطلاعات موجود در دیاگرام به‌واسطه موارد استفاده آن‌ها تغییر می‌کند. در یک دیاگرام مداری که براساس ساختارهای موجود پایه‌ریزی شده است هر گیت می‌بایست ۴ عامل سخت‌افزاری را بیان کند.

۱) یک نامگذاری دلخواه برای یک تراشه ویژه شامل گیت مانند A_{13}

۲) نوع و ماهیت مدار مجتمعی که از آن استفاده می‌کنیم مانند 74LS04

۳) تعداد پین‌ها برای هر گیت ورودی و خروجی

۴) کاربرد و وظیفه هر پین چیست؟ (برای چه کاری قرار داده شده است)

نسخه پیش‌نویس Draft

برچسب تراشه‌ها و نوع آن‌ها هر دو مطابق سمبل بکار رفته برای آن‌ها مشخص خواهند شد.

تعداد پین‌ها خارج سمبل یک تراشه مشخص می‌شود نزدیک به خطوط ورودی خروجی و برچسب وظایف پین‌ها مطابق با نماد مجاور ورودی یا خروجی مربوطه درج می‌شود.

نشانه‌گذاری منطق مختلط برای گیت‌های AND و OR عموماً جهت پی بردن به ماهیت و نوع وسیله مورد طراحی کفایت می‌کند عموماً در مواردی که آشفتگی و بی‌نظمی زیاد نیست از ذکر برچسب برای تراشه‌های مشخص جلوگیری می‌کنیم به‌منظور جلوگیری از ایجاد شلوغی طرح مدار. اما ذکر کامل مشخصات تراشه‌ها همواره مفید است.

برچسب‌گذاری برای کارهای مرتبط با پین‌ها عموماً در بیشتر گیت‌ها ضروری نیستند از این‌رو که خود نمادها بیانگر وظایف ورودی و خروجی‌ها هستند.

ولی در مدارات پیچیده‌تر ذکر این موارد اغلب مفید است و سبب ایجاد وضوح در طرح ما می‌شوند لذا ما از ذکر کارهای پین‌ها روی گیت‌ها خودداری می‌کنیم.

در دیاگرام‌های مداری ما عموماً تمایل داریم به یک ساختار حقیقی نزدیک شویم از این‌رو ما در این گونه موارد ما به ذکر برچسب تراشه‌ها و تعداد پین‌ها می‌پردازیم اگرچه مجبور شویم علائم مربوط به نوع تراشه را حذف کنیم بیشتر دیاگرام‌ها در این کتاب بیشتر با خصوصیات منطقی مدار سر و کار دارند تا خصوصیات سخت‌افزاری آن. اما عموماً ما از ذکر تعداد پین‌ها برچسب تراشه‌ها نیز برای بهبود بخشیدن به قابلیت خواندن مدار صرف‌نظر می‌کنیم.

یک تراشه عموماً شامل چندین نوع مختلف گیت است برای مثال 74LS02 دارای چهار گیت NOR است. یک طرح عمومی برای مدارات ترکیبی عبارت است از نامگذاری هر یک از گیت‌ها با مقداری معادل با کسری از مقدار تمامی گیت‌ها به‌عنوان نمونه هر گیت NOR مربوط به یک تراشه 74LS02 تحت عنوان 74LS02 $\frac{1}{4}$ یا $\frac{1}{4}$ '$LS02$ نامگذاری می‌شود که با این روش یک دیاگرام مداری خیلی راحت با وجود این سمبل‌ها غیر واضح خواهد شد. لذا ما در این کتاب از اینگونه نشانه‌گذاری کسری صرف‌نظر می‌کنیم مگر اینکه عدم وجود آن سبب ایجاد آشفتگی می‌شود.

شما ممکن است قراردادهای ما را بپذیرید در این بخش ما دیاگرام‌ها را به‌طور کامل با نشان دادن برچسب هر گیت و نوع وسیله سخت‌افزاری و تعداد پین‌ها نشان می‌دهیم و سپس در بخش بعدی ما اطلاعات رسم و یا گرام خود را تنها به موارد مفید محدود می‌کنیم.

نسخه پیش‌نویس

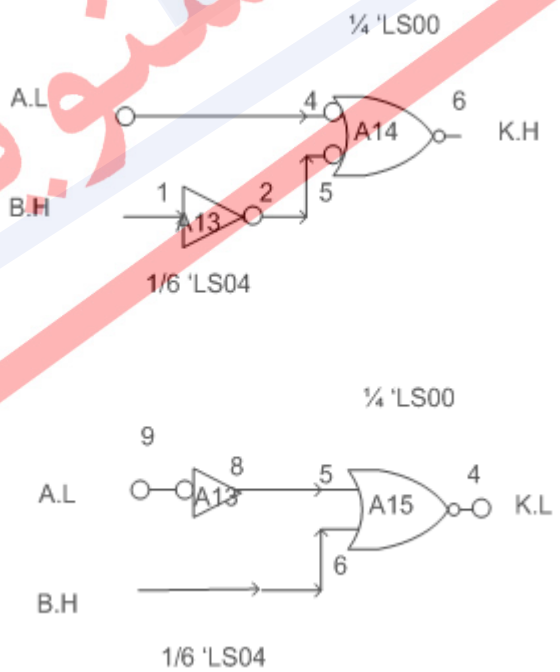
۱۰ – ۹ ساختار کلی منطق مختلط

کار آزمودگی در سنتز مدارات با منطق مختلط در گرو چند عامل مهم است که تعدادی از آن‌ها را ما بیان می‌کنیم و به بقیه با رشد مهارت و تجربه خود در اینکار پی خواهید برد عادت ما به قراردادن یک دایره کوچک در عبارت اسلش در واقع یک قرارداد است شما در رد یا پذیرش آن مختار هستید در واقع اسلش خود یک قرارداد است از اینرو می‌توان به وجود یک NOT منطقی از دیاگرام ولتاژی یک مدار بدون وجود اسلش پی برد.

با این حال ما قویاً توصیه به استفاده از اسلش جهت نمایش Not منطقی می‌کنیم چرا که وضوح مدار طراحی شده را افزایش می‌دهد در تقابل بین معکوس‌کننده‌های ولتاژ و پلاریته ولتاژها در منطق مختلط فرصت‌های مناسب تصمیم‌گیری رخ می‌دهد که تصمیم ما بر بهینه بودن مدار تأثیر می‌گذارد.

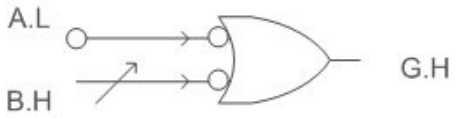
به‌طور مثال یک طراحی برای k=A+B درنظر بگیرید با سیگنال‌های ورودی A.L , B.H هر پیاده‌سازی به آسانی گیت‌هایی که دارای یک معکوس‌کننده روی یکی از پایه‌های ورودی خود هستند قابل دسترسی است.

شکل ۱۵ دو طراحی را برای تابع منطقی بالا نمایش می‌دهد حال کدامیک از آن‌ها بهتر است؟ برای یک طراح که بعدها دوباره به سیگنال k.L نیاز دارد شکل ۱۵(ب) بهتر است و ۱۵(الف) نیز در موقعیت متناسب خودش بکار خواهد رفت.

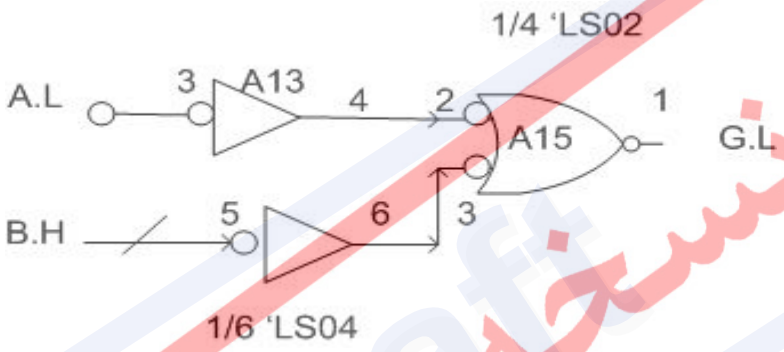


شکل 10-۱۵ – مدار با معادله منطقی k=A+B

مثال بعدی واضح‌تر است برای طراحی مداری با تابع منطقی ورودی $G = A + \overline{B}$ و دوباره با ورودی‌های A.L و B.H یک مدار خوب عبارت است از:



این مدار هیچگونه معکوس‌کننده‌ای ندارد.حال مورد دیگری را بررسی می‌کنیم فرض کنید می‌خواهیم عبارت G.L را در خروجی داشته باشیم برای اینکار هم می‌توان از مدار زیر بهره برد



و هم از مدار شکل قبلی که یک NOT در خروجی آن واقع شده است. حال کدام مورد بهتر است؟ در طراحی مدارات به نحوه چیدن گیت‌ها می‌بایست توجه لازم را داشت تا به یک مدار با مقدار بیشینه بهینگی دست پیدا کنیم.

در مورد تغییرات اصلاحی مدار خود که با هدف بهینگی آن انجام می‌شود. احتیاط کنید. شکل 2.16 سه مدار را نشان می‌دهد

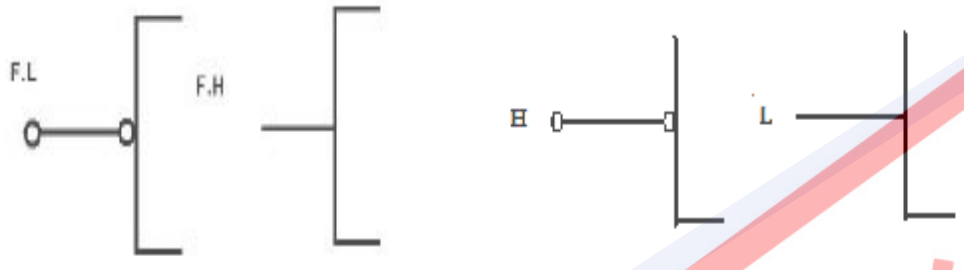
هر کدام یک پیاده‌سازی از معادلات $M = J.\overline{X}$ ، $N = G + X$ این مدارات گیت‌ها و سیم‌های یکسان دارد و از این‌رو می‌بایست به صورت معادل از لحاظ منطقی عمل کند اما خطوط کلی راهنما را برای طراحی مدارات به یاد آورید هر طراحی می‌بایست منطق واقعی و اصلی آنچه مدنظر بود را نشان دهد و فقط شکل ۱۶ یک پیاده‌سازی مناسب را برای منطق اولیه ارائه می‌دهد و این غیر محتمل به‌نظر می‌رسد که دو معکوس کننده متغیر X در شکل ۱۶ در مراحل ساده‌سازی همچنان حفظ شود. ما قانون مفید زیر را در بکارگیری Not منطقی در مدارات پیچیده یافته‌ایم

.اسلش را در سمت راست‌ترین نقطه سیم سیگنال مربوطه آن که در دسترسی است قرار دهید.

به‌طور متناوب یک ورودی به کمک یک وسیله سخت‌افزاری روی T یا F قرار می‌گیرد که این کار اغلب به‌وسیله تراشه‌های پیچیده‌ای انجام می‌شود که بعدها مورد بحث قرار می‌گیرد در جائیکه شما

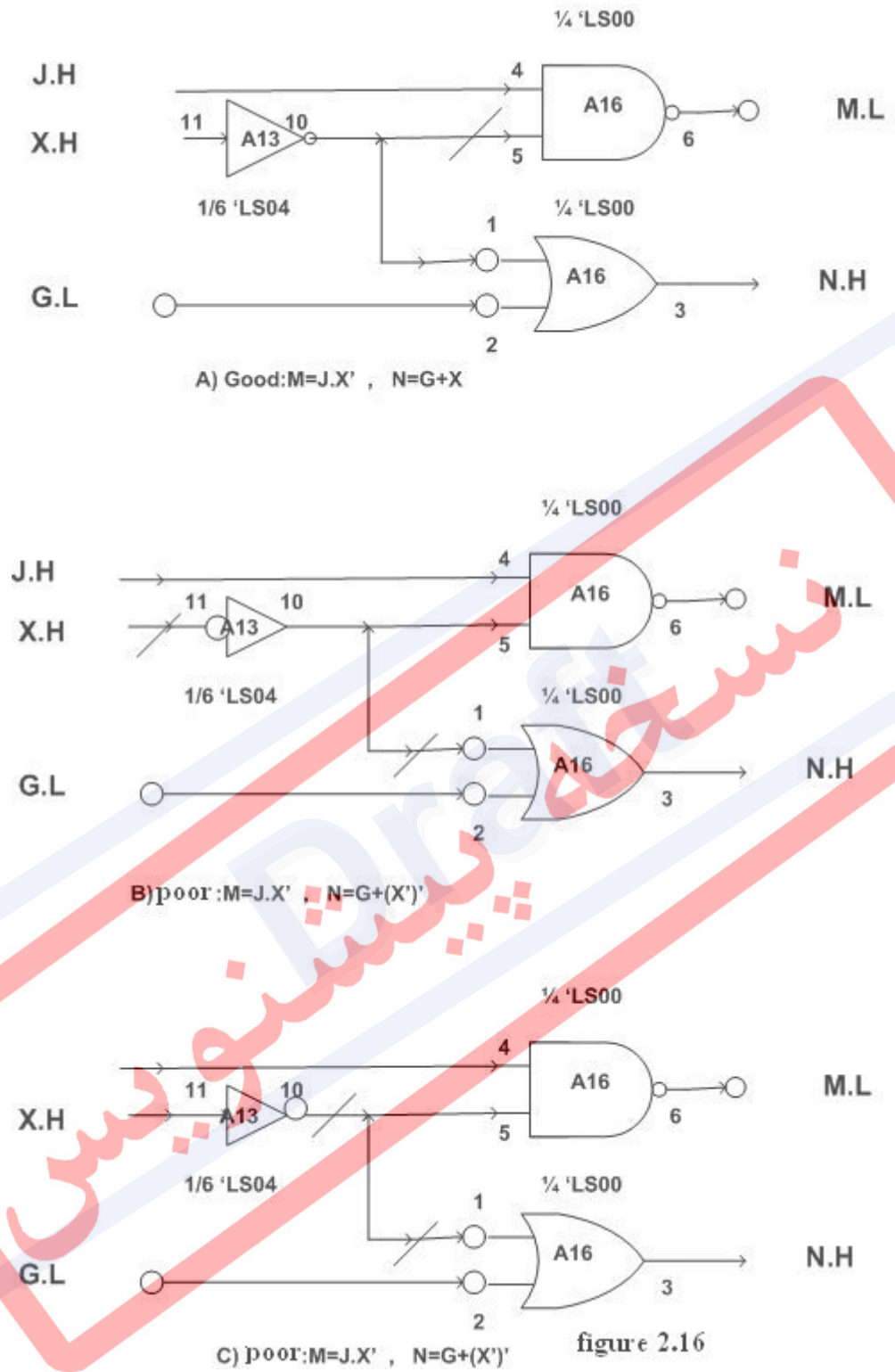
نسخه پیش‌نویس Draft

تمایل دارید یک خصوصیت ویژه یک تراشه را دائماً در حالت فعال قرار دهید یا از حالت فعال خارج سازید فرض کنید ما می‌خواهیم بگوییم هرگز آن کار را انجام نده برای اینکار ما یک سیگنال دائمی با ارزش درستی منفی (False) نیاز داریم که در اینجا ما دو نحوهٔ نمایش در منطق مختلط برای بیان ارزش درستی منفی داریم.



در این عمل ولتاژ روی سیم هیچگاه تغییری نمی‌کند برای راحتی ما گاهی اوقات سطح ولتاژ(H,L) را در یک دیاگرام نشان می‌دهیم.

نسخه پیش‌نویس
Draft

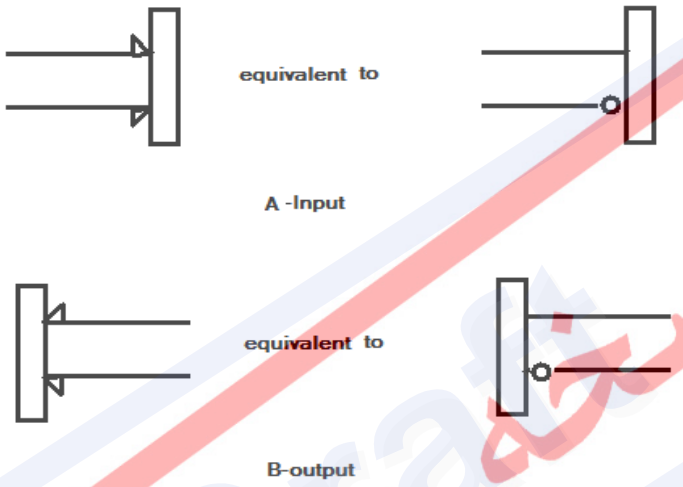


شکل 10-۱۶ - پیاده‌سازی های مختلف از معادلات منطقی $M = J.\overline{X}$ ، $N = G + X$

نسخه پیش‌نویس Draft

۱۰-۱۰ دیگر صورت‌های نمادسازی منطق مختلط

وجود یا عدم وجود یک دایره کوچک در یک منطق مختلط نشان‌دهنده سطح ولتاژ در هر گره در یک دیاگرام است یک نماد محبوب دیگر در منطق مختلط در دیاگرام مداری یک مثلث کوچک است که پلاریته ولتاژ را نشان می‌دهد یک مثلث کوچک در بالای خط بیانگر T=H است و یک مثلث در پایین آن بیانگر T=L.



شکل 10-۱۷ - نماد مثلث کوچک برای پلاریته ولتاژ

۱۱-۱۰ منطق مثبت چیست؟

منطق مثبت به طرز گسترده در کاربردهای مهندسی استفاده می‌شوند. ترکیب به سهولت قابل یادگیری است، اما این سهولت یادگیری فریبنده است، برای این که طراحی مدارهای حقیقی با منطق مثبت به محدود کردن (خلاصه کردن) قوانین و انتقال‌ها (تبدیلات) ناهنجار منجر می‌شود.

هنگامی که یک مدار ترکیبی که از منطق مثبت استفاده می‌کند، True (صحیح) منطقی همیشه با ولتاژ بالا نشان داده می‌شود و False (غلط) با ولتاژ کم نشان داده می‌شود. تحت این شرایط، جدول ولتاژ برای گیت (NOR) 74LS02 می‌تواند تبدیل به جدول درستی برای تابع NOR منطقی شود.

به طریق مشابه، جدول‌های ولتاژ برای گیت NAND 74LS00، گیت AND 74LS08 و گیت OR 74LS32 می‌توانند به‌عنوان جدول صحیح برای NAND منطقی، AND و OR ترجمه شوند. یک معکوس‌کننده (inverter) ولتاژ برای مثال به‌صورت تابع معکوس‌کننده منطقی عمل می‌کند در منطق

نسخه پیش‌نویس Draft

مثبت ما همچنین به یک معکوس‌کننده منطقی با استفاده از گیت‌های AND و OR دست پیدا کنیم با دادن سیگنال‌های یکسان به ورودی‌های هر دو گیت.

Voltage

A	B	Y
F	F	T
F	T	F
T	F	F
T	T	F

Positive Logic

A	B	Y
L	L	H
L	H	L
H	L	L
H	H	L

Nor Logic Function

از اینرو در مسائل بعدی ما استفاده از NAND و NOR یا معکوس‌کننده را برای ساختن معکوس‌کننده آزاد می‌گذاریم یک کاربرد قانون دمورگان نشان می‌دهد که تابع NAND ممکن است به یک تابع OR با ورودی معکوس شده تبدیل شود.

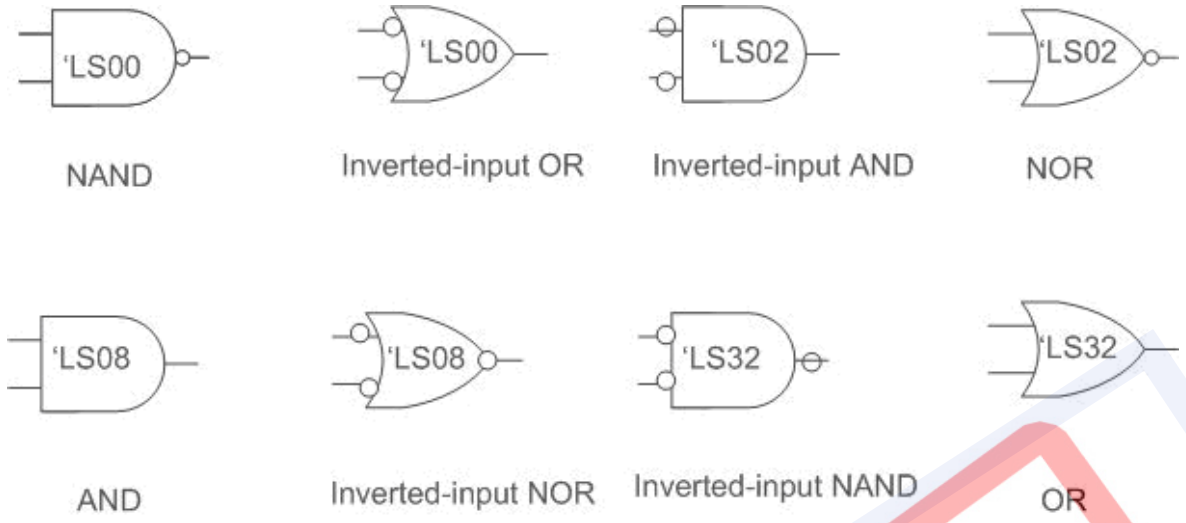
$$\text{A NAND B} = \overline{AB} = \overline{A} + \overline{B}$$

به‌طور مشابه تبدیل‌های NOR و OR تولید یک AND با ورودی معکوس شده یک NOR با ورودی معکوس‌شده و یک NAND با ورودی معکوس شده می‌کند در منطق مثبت رابطه بین سطح ولتاژها و ارزش درستی به‌طور کامل و دقیق مشخص است برداشتن یک گیت سخت‌افزاری معادل برداشتن یک تابع منطقی از معادله کلی است.

توابع AND و OR ماهیتاً براساس ترانزیستورها چیده شده‌اند. ساختن AND و OR نیازمند قرار دادن معکوس‌کننده ولتاژ در یک تراشه هستند چرا که ترانزیستور ذاتاً به‌صورت معکوس‌کننده عمل می‌کند.

کاربران منطق مثبت برای بکار بردن AND و OR از پیاده‌سازی NAND و NOR استفاده می‌کنند. با استفاده از قوانین دمورگان آن را بدل به توابع AND و OR خواهند کرد یک نشانه مرسوم برای مدارات مبتنی بر منطق مثبت یک دایره کوچک بیانگر عمل معکوس‌کننده منطقی است شکل ۱۸ چندین گیت مرسوم در منطق مثبت را نشان می‌دهد.

نسخه پیش‌نویس Draft

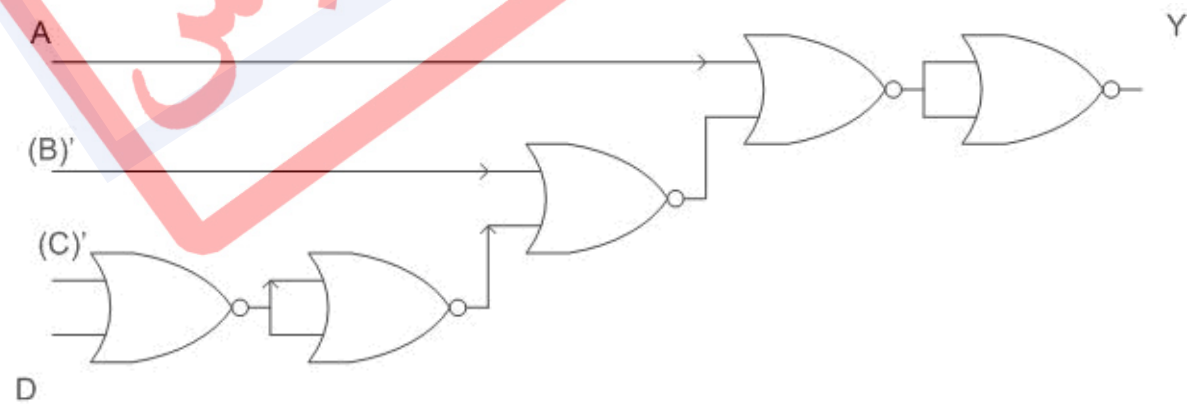


شکل 10-۱۸ - چندین گیت در منطق مثبت

برای پیدا کردن معادله منطقی، یک طراح در منطق مثبت می‌بایست منطق را به‌طور کلی در هر دو فرم جبری و گرافیکی تبدیل کند به فرمی شبیه و نزدیک به گیت موردنظر. در این فرآیند اصالت معادله اصلی از بین خواهد رفت یک دیدگاه جبری تلاش می‌کند با قوانین دمورگان دوباره معادله اصلی را قالب‌ریزی کند توسط تراشه‌هایی که منطق مثبت را پشتیبانی می‌کنند. به‌طور مثال معادله زیر را درنظر بگیرید که به‌طور یکسان با تابع NOR دو ورودی پیاده‌سازی شده است ما ممکن است این معادله را با معادله‌ای صرفاً وابسته به NOR و عملگرهای معکوس‌کننده بیان کنیم. طی چندین مرحله خسته‌کننده اضافی‌تر. مانند:

$$Y = A + \overline{B.\overline{C} + \overline{\overline{b}}} \qquad \overline{\overline{A + \overline{\overline{B} + \overline{\overline{C} + b}}}} \qquad = A + \overline{\overline{B} + \overline{\overline{C} + b}}$$

که شکل زیر بیانگر تابع منطقی معرفی شده و پیاده‌سازی شده با گیت NOR است.



شکل 10-۱۹ - مدار برای معادله منطقی $Y = A + \overline{B.\overline{C} + D}$

نسخه پیش‌نویس

کتاب‌های طراحی دیجیتالی در جایی که تکنیک‌های منطبق مثت مورد استفاده هستند شامل رهنمودهایی از طرح‌های مداری با استفاده از گیت‌های NAND و NOR است بدون اینکه از انجام تبدیلات جبری استفاده کنیم آن‌ها مجموع قوانین موردنیاز را برای NAND و NOR به‌صورت مجزا بیان می‌کنند و یکسری ترکیب‌های ویژه OR, AND, NOR, NAND نیز مورد بحث واقع می‌شود تمامی این اصول نیاز دارند به‌صورت SOP یا POS شروع شوند در نتیجه یک طراح به طراحی به‌صورت یک بیان منطقی منحصر به فرد برای به‌دست آوردن مدار وادار می‌شود.

حال می‌خواهیم مدارات مبتنی بر منطق مثبت را نشان دهیم با استفاده از ساده‌ترین مجموعه قوانین سنتز گیت‌های مدارات دو مرحله‌ای.

عبارت مرحله در سطر بالا بیانگر بیشترین تعداد گیت واقع شده بین ورودی و خروجی یک گیت است.

با فرض اینکه NANDهای چندین ورودی در دسترس است قوانین عبارتند از:

۱- ساده سازی تابع به‌صورت حاصل جمع حاصلضرب‌ها (SOP)

۲- رسم یک گیت NAND برای هر حاصل ضرب که حداقل دو متغیر دارد متغیرها به شکل ورودی به تابع NAND است این گیت‌های NAND اولین مرحله به‌شمار می‌آیند.

۳- یک گیت NAND را به تنهایی رسم کنید با استفاده از هر دو حالت معادل { NAND یا گیت و با ورودی معکوس شده } که این گیت به‌عنوان مرحله دوم شمرده می‌شود و ورودی‌های آن از مرحله اول بدست می‌آید.

۴- برای یک AND که به‌صورت تک متغیره است یک معکوس‌کننده قرار دهید در مرحله اول و به‌طور متناوب از متمم متغیرها به‌عنوان ورودی برای مرحله دوم استفاده کنید.

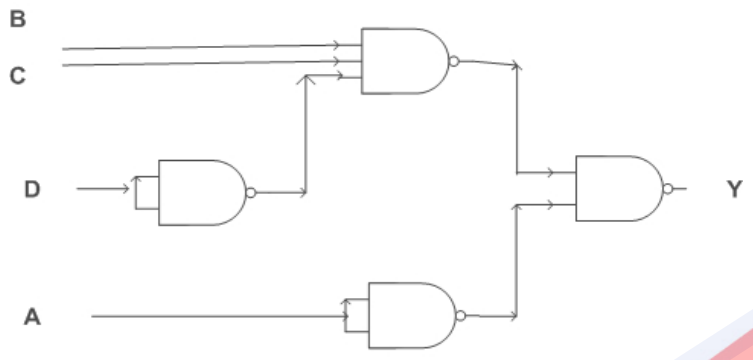
بکار بردن این قوانین برای مثال قبلی تولید یک مدار دو مرحله‌ای می‌کند که نتیجه در شکل ۲۰ مشخص شده است.

قوانین برای دیگر سنتزهای دو مرحله‌ای در عمل پیچیده‌تر هستند. کتاب‌های منطق مثبت شامل قوانین برای ساختن مدارات به‌طور کلی چندین مرحله‌ای نمی‌شوند چرا که قوانین ممکن است بسیار سنگین باشد.

برای رسیدن به منطق ترکیبی این تکنیک‌ها غیرضروری و حتی مضر هستند. با منطق ترکیبی، ما به آسانی نمی‌توانیم هر مداری را برای معادلات منطقی، با هر پیچیدگی، با استفاده از چیپ‌های موردنظر،

نسخه پیش‌نویس Draft

در خلال تنظیم معادله منطقی ساختار اصلی را گسترش می‌دهیم. در منطق ترکیبی موارد خاصی برای گیت‌ها و انواع مختلف وجود ندارد.



Positive logic implementation of Y=A+B.C.(D)'
using Nand gate rules

شکل 10-۲۰ - مدار برای معادله منطقی $Y = A + \overline{B.\overline{C} + D}$

در فصل‌های بعدی، که با مدارهای پیچیده نرم‌افزاری شامل ورودی‌ها و خروجی‌های صحیح در منطق مثبت و منطق منفی برخورد می‌کنیم مزیت عملیات منطق ترکیبی بیشتر، مشخص می‌شود.

خواندن نمودار مدارهای منطق مثبت:

روش یک منطق ترکیبی برای اجرای یک عملیات منطقی (NOT) بدون استفاده از ابزار فیزیکی و روش‌های مرتبط با ابزارهای فیزیکی (معکوس‌کننده) که هیچ منطقی را اجرا نمی‌کند، مستقیماً از تأکید ما بر این نکته که منطق اصلی باید نمودار مدار آشکار باشد. مدارهای ترکیبی منطق مثبت و منطق منفی که در آن‌ها منطق و ولتاژ دقیقاً با هم منطبق هستند، معکوس‌کننده، معکوس منطبق را اجرا می‌کند. گاهی اوقات ما سعی می‌کنیم یک معادله منطقی را از یک مدار ترکیبی قطعی (فیکس) استخراج کنیم همان‌طور که ما نشان داده‌ایم، بازیابی منطق اصلی ممکن نیست. مدار تنها یک نسخه تبدیل یافته از اصل را نشان می‌دهد. آیا یک منطق‌دهی مختلط می‌توانند یک مدار منطق مثبت را بخواند؟ در حقیقت چند دیدگاه در این زمینه وجود دارد یک روش آن است که نمودار منطق مثت مستقیماً خوانده شود، یعنی تفسیر گیت‌های NAND به عنوان NOT AND منطقی، گیت‌های NOR به عنوان NOT OR منطقی و معکوس کننده‌ها مانند NOT منطقی و به همین ترتیب.

سپس نوشتن یک معادله منطقی از مدار با استفاده از این دیدگاه ما بجای A NAND B ، A . B و بجای A NOR B نیز A + B نوشته می‌شود. اگر شما از تعدد معکوس کننده‌های منطقی در نتیجه ناراضی هستید می‌توانید قانون دمورگان را برای حذف برخی از آنها به کار برید.

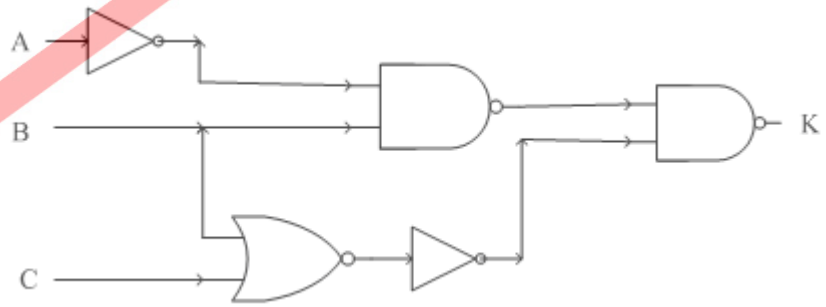
نسخه پیش‌نویس Draft

دیدگاه های مشابه به ما امکان خواندن نمودارهای منطق منفی به روشی مشابه روش بالا می‌دهد. به عنوان مثال به شکل ۲۱ توجه کنید. یک مدار با منطق مثبت با دنبال کردن دستورات داریم:

$$K = \overline{(\overline{\overline{A}.B}).(\overline{\overline{\overline{B}+\overline{C}}})}$$
$$= \overline{(A+\overline{B})(B+\overline{C})}$$
$$= \overline{(A+\overline{B})+(B+\overline{C})}$$
$$= \overline{A}.B + B.\overline{C}$$

شکل 10-۲۱ - روش جبری

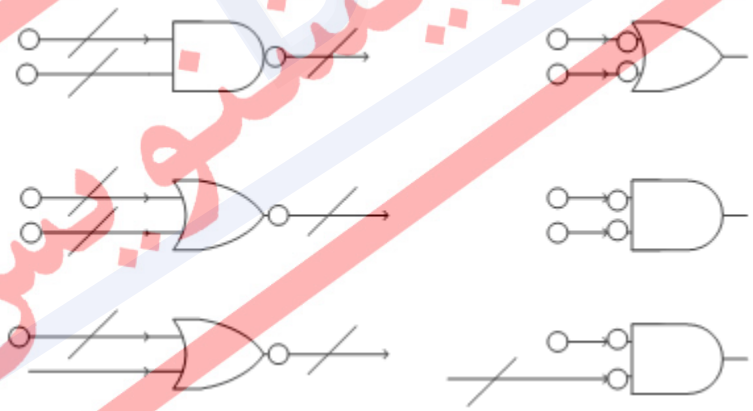
چه زمانی ما دستکاری عبارت K را متوقف می‌کنیم؟به کمک مدار شکل ۲۱ ما به جواب نمی‌رسیم و تمام شکل‌های جبری منطقی برای K درست است اما ما هیچ راهی برای مشخص کردن منظور اصلی طراح نداریم ما فقط می‌توانیم مطمئن باشیم که منظور طراح معادله اول در شکل ۲۱ نبوده‌است. این روش خواندن دیاگرام منطق مثبت از یک معادله منطقی از نمودار نامشخص مدار تولید می‌کند و همچنین معادله را به فرم ساده‌تری در می‌آورد. روش دیگر تبدیل دیاگرام مدار از منطق مثبت به منطق مختلط است به طوری که ما فقط یک معادله را می‌خوانیم این روش یک روش گرافیکی است در حالی که روش اول یک روش جبری است. برای روش ترسیمی مراحل زیر را طی کنید. ابتدا H. را به ورودی‌ها و خروجی‌های منطق مثبت اضافه کنید و اگر مایل هستید یک ورودی یا خروجی منفی را با شکل‌های منطق مختلط غیر منفی جایگزین کنید. برای مثال یک ورودی $\overline{G}$ به یک $\overline{G}.H$ منطق مختلط تبدیل می‌شود. و شما ممکن است آن را به شکل G.L با یک دایره بر روی خط ورودی تعریف کنید. سپس هر کجا که دایره در انتهای یک خط جا نمی‌گیرد یک اسلش برای تأکید NOT منطقی ضمنی وارد کنید.



شکل 10-۲۲ - روش گرافیکی

در جایی که گیت‌ها توسط اسلش پر شده‌اند شما ممکن است راه‌حل خود را با جایگزینی AND و ORها با همتایان آن‌ها در منطق مختلط ساده‌سازی و مختصر کنید که این مورد یکی از خواص قانون دمورگان است در دیاگرام حاصل یک دایره برای بیان معکوس‌کننده و یک تغییر در سمبل‌های منطقی به دوگان معادل آن‌ها پدیدار خواهد شد. دایره معکوس‌کننده نیاز به اصلاح اسلش‌های روی گیت ورودی و خروجی دارد که منجر به ایجاد یک مدار ساده‌تر می‌شود این فرآیند در شکل ۲۲ نشان داده شده است بعد از این تبدیل به یک مدار در منطق مختلط خواندن منطق مدار نیز ساده‌تر خواهد شد در شکل ۲۳ ما یک تبدیل از نوع تبدیل به منطق مختلط را نشان داده‌ایم.نتیجه مشابه با معادله دوم در شکل ۲۱ است دوباره ما نمی‌توانیم مطمئن باشیم که آن معادله اولیه طراح است چرا که قالب معادله اولیه را حفظ نکرده است.

معادله دوم در شکل ۲۱ به یکی از حالات اولیه شروع اشاره می‌کند در سنتز با منطق مختلط برای این معادله یکی از انتخاب‌ها ورودی A.H و B.H و C.L مطابق با شکل ۲۱ است ما می‌توانیم فرض کنیم که شکل خروجی K نیز مشخص نیست شکل ۲۴ یک مدار با منطق مختلط را برای معادله دوم در شکل ۲۱ نشان می‌دهد از آنجا که ما به K اجازه داده‌ایم به هر دو صورت یک سیگنال (L, H) نشان داده شود شکل ۲۴ شامل یک معکوس‌کننده کمتر از شکل۲۱ است. آیا این نوع طراحی بهتر است؟ چراکه در استفاده گیت‌ها صرفه جویی کرده است.



شکل 10-۲۳ - تبدیل به منطق مختلط

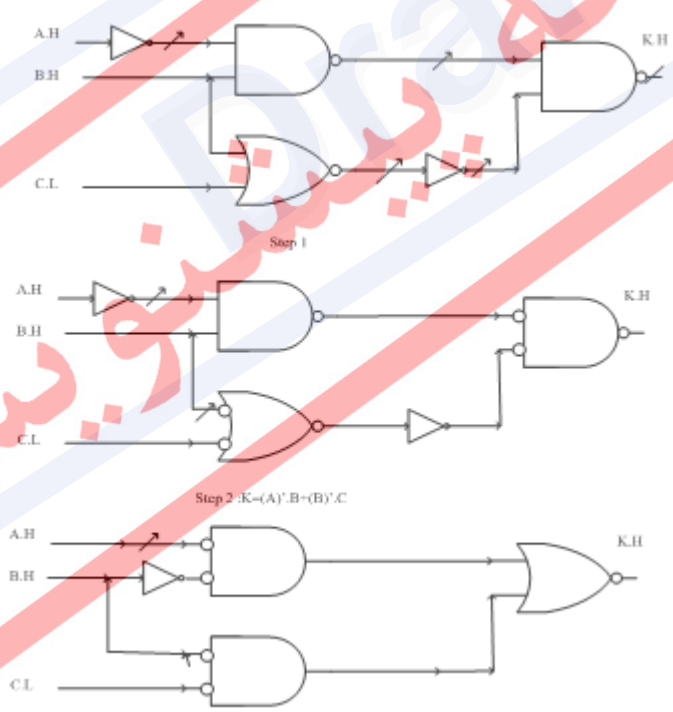
حال حالتی را فرض کنید که ما فقط نیاز به سیگنال K.H در مراحل بعدی داریم پس نیاز به 74LS04 داریم تا بتوانیم K.L را به K.H تبدیل کنیم از اینرو منطق مختلط از قبل توسط ولتاژها اشغال نشده است بلکه طرح مدار و دیاگرام ولتاژها را بنا به ضرورت تعیین می‌کند متدهای منطق مختلط فرصت‌های مناسبی را برای صرفه‌جویی در بکارگیری گیت‌ها فراهم می‌آورد منطق مختلط همیشه یک

نسخه پیش‌نویس Draft

نتیجه ایجاد می‌کند که حداقل از لحاظ سخت‌افزاری از دیگر سیستم‌ها اقتصادی‌تر است منطق مختلط همراه آن‌قدر در مصرف گیت‌ها صرفه‌جویی می‌کند که مقدار معادله اولیه حفظ می‌شود.

ما بحث خود را در مورد دیگر قراردادها در طراحی با یک هشدار به اتمام می‌رسانیم در برخی از بخش‌های انجمن مهندسی الکترونیک منطق یک و منطق صفر اشاره به ولتاژ بالا و ولتاژ پایین دارد. این تعریف از متغیر منطقی بسیار قدیمی است و به دورانی باز می‌گردد که مدارات دیجیتالی بین سطوح مختلف ولتاژی به‌درستی تمایز قائل شدند.

اصطلاحات علمی نشان می‌دهد که آشفتگی زمانی رخ می‌دهدکه متغیرهای منطقی و سطح ولتاژ آن‌ها به‌عنوان دو مقوله مستقل تلقی نشوند.این عقیده دست و پا شکسته امروزه راهی هوشمندانه‌تر را برای مطرح کردن فکرهای کاربردی‌تر جهت جدا کردن ولتاژ و منطق از هم ارائه کرده است. اما شما گاهی اوقات با این موارد استفاده قدیمی روبرو می‌شوید بنا براین هنگام مطالعه این مستندات و هنگامی که با بقیه طراحان در این زمینه صحبت می‌کنید باید هوشیار باشید.



شکل 10-۲۴ - مدار با منطق مختلط برای معادله $K=\overline{(A+\overline{B})(B+\overline{C})}$

نسخه پیش‌نویس Draft

۱۲-۱۰ بکارگیری منطق مختلط برای توابع XOR

ما روی AND, OR و NOT به‌عنوان عناصر اصلی طراحی مدار تأکید کردیم آیا شما توابع منطقی دیگری را می‌شناسید که با دو متغیر در طراحی مورد استفاده واقع شود.

تابع XOR فقط زمانی درست است که دو ورودی دارای ارزش منطقی متفاوتی باشند یکی درست در حالیکه دیگری غلط است و تابع COINCIDENCE فقط زمانی درست است که هر دو متغیر ورودی یکسان باشند. هر دو درست یا هر دو غلط نماد مربوط به این تابع منطقی به‌صورت زیر است:

EXCLUSIVE OR

COINCIDENCE

این عملگرها مطابق جدول درستی زیر می‌باشند. مشاهده کنید که مقدار COINCIDENCE عکس حالت EOR است تابع منطقی پیاده‌سازی شده توسط توابع COINCIDENCE, EOR توسط AND و OR و NOT به صورت زیر است (به ترتیب) $A\ EOR\ B = \overline{A}.B + A.\overline{B}$ و $A\ COI\ B = \overline{A}.\overline{B} + A.B$

LOGIC			
A	B	A ⊕ B	A ⊙ B
F	F	F	T
F	T	T	F
T	F	T	F
T	T	F	T

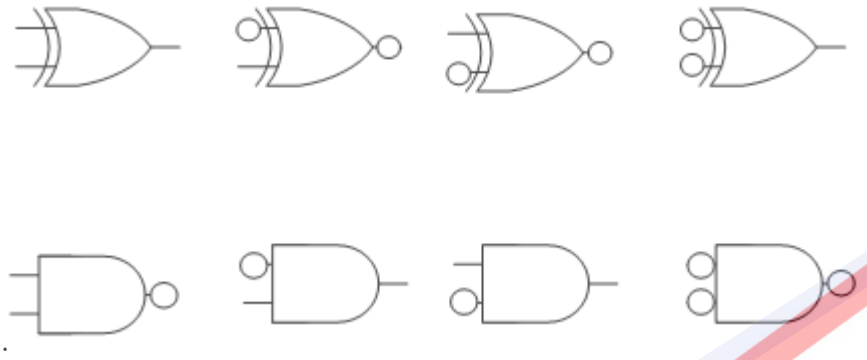
A ⊕ B =(A)'.B+A.(B)'

A ⊙ B =A.B+(A)'.(B)'

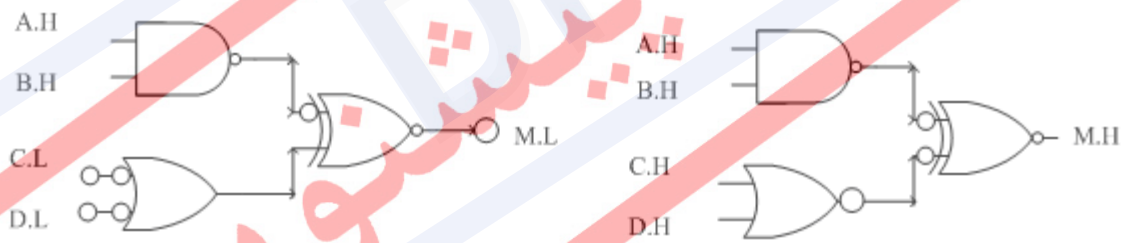
در ارزیابی سلسله مراتبی برای عملگرهای منطقی EOR و COINCIDENCE هر دو پایین‌تر از NOT و بالاتر از AND قرار دارند یک نگاه اجمالی به یک کتابچه راهنمای TTL بیان می‌کند که یک جدول ولتاژ به‌صورت زیر داریم حال با توجه به روش سابق بررسی خود که مقادیر ورودی می‌تواند هم در

نسخه پیش‌نویس Draft

منطق مثبت و هم منفی باشد به چهار سمبل و شکل زیر می‌رسیم که از چهار جدول و دیاگرام مختلف حاصل شده‌اند (74LS86) همینطور چهار سمبل زیر را نیز برای COINCIDENCE بیان می‌کند



حال به شکل‌ها توجه کنید ببینید که سمبل‌های EOR تعداد زوجی دایره دارند در حالی که برای COINCIDENCE تعداد آن‌ها مقداری فرد است پس این تراشه شگفت انگیز (74LS86) هشت سمبل را برای جدول منطقی ما ارائه کرده است و ما بنابه نیاز خود در طراحی یک مدار خاص ممکن است از هر چهار سمبل EOR استفاده کنیم. برای نمونه در شکل 10-۲۵ دو طرح مختلف از یک تابع منطقی که فقط با EOR پیاده‌سازی شده‌اند را نشان می‌دهد. با وجود استفاده از منطق مختلط وجود چنین حالت‌هایی کاملاً بدیهی است.

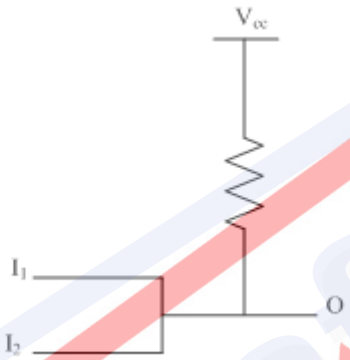


شکل 10-۲۵ - دو طرح مختلف از پیاده‌سازی یک تابع منطقی با EOR

گیت‌های Open -Collector:به گیت‌هایی گفته می‌شود که خروجی حاصل از آن‌ها را می‌توان به هم متصل کرد که در طراحی مدار قسمت شبکه بالابر ولتاژ جهت رساندن مقدار ولتاژ خروجی به یک در آن تعبیه نشده است و تنها یک مقاومت مقدار کل خروجی را به VCC متصل می‌کند.به‌طور کامل‌تر می‌توان گفت در مدارات مجتمع TTL ما نمی‌توانیم دو خروجی را به‌طور مستقیم به هم وصل کنیم چرا که خروجی‌ها می‌توانند مقادیر مختلفی را در خروجی برای خود بگیرند و این به مدار صدمه می‌زند به‌طور مثال اگر یکی از خروجی‌ها در وضعیت کمینه ولتاژ باشد و دیگری در وضعیت بیشینه، یک رقابت در

نسخه پیش‌نویس Draft

خروجی اتفاق می‌افتد که می‌خواهد خروجی نهایی را صفر یا یک کند. مقدار جریان زیادی از VCC به سمت زمین می‌رود و توان ایستای مدار بسیار زیاد می‌شود ما در این حالت برای ترکیب کردن سیگنال‌ها از گیت‌های AND و OR استفاده می‌کنیم بنا به اینکه تابعی که قصد پیاده‌سازی آن را داریم چه تابعی را پیاده‌سازی خواهد کرد حال اگر تعداد خروجی‌ها بسیار زیاد باشد و بخواهیم همه آن‌ها را با هم AND یا OR کنیم گیت زیادی هزینه کرده‌ایم حال چه باید کرد در اینجاست که کلکتور باز[1] به کمک ما می‌آید. این تکنولوژی یکی از شاخه‌های TTL است اینجاست که اجازه می‌دهد خروجی بدون استفاده از گیت-های AND و OR تابع منطقی AND و OR را پیاده سازی کند.



در گیت‌های کلکتور باز مدار خروجی ناکامل است و مانند گیت‌های معمولی TTL یک مسیر داخلی جهت دست یافتن به VCC ندارد یک گیت معمولی هر دو سطح ولتاژ را می‌تواند فراهم آورد ولی برای یک گیت کلکتور باز قسمت بالا برنده ولتاژتنها زمانی عمل می‌کند که تمامی عوامل پایین‌آورنده ولتاژ قطع باشند هرگاه یکی از خروجی ایجاد کننده یک گیت کلکتور باز در سطح کمینه قرار گیرد مقدار خروجی نهایی نیز در سطح کمینه خواهد بود.

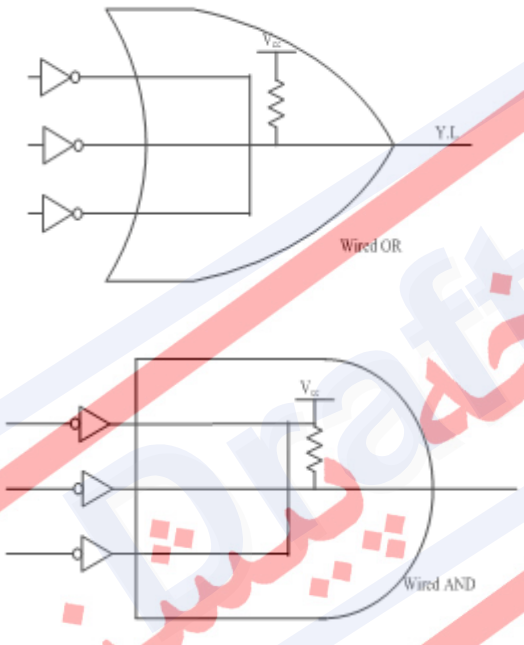
حال سوالی که مطرح می‌شود تکلیف خروجی منطقی یک گیت کلکتور باز چیست؟ هنگامی که دو خروجی کلکتور باز به‌عنوان ورودی به یک جعبه مداری واقع می‌شوند جدول ولتاژ به صورت زیر است. اگر ما مقدار T=H را انتخاب کنیم برای تمامی خروجی‌ها و ورودی‌ها این جدول ولتاژ تبدیل می‌شود به جدول منطقی یک AND و با انتخاب T=L برای تمام ورودی‌ها و خروجی‌ها ما به جدول درستی مدار OR می‌رسیم. نتیجه برای حالتی که مقدار ورودی‌ها را بیش از دو تا درنظر بگیریم نیز همانند همین حالت است و دراینجا به یک قانون کلی در مورد مدارات کلکتور باز می‌رسیم یک گیت کلکتور باز وقتی که مقدار درستی منطقی با مقدار سطح بالای ولتاژ معادل گذاری شود مشابه یک گیت AND عمل

[1] Open Collector

نسخه پیش‌نویس

می‌کند و وقتی مقدار درستی منطقی با مقدار سطح پایین ولتاژ نمایش داده شود مشابه یک گیت OR عمل می‌کند.

اجازه دهید یک تراشه به‌خصوص کلکتور باز را درنظر بگیریم معکوس‌کننده 7406 گیتی از این نوع دست برای دست یافتن به رفتاری معادل گیت OR مقدار T=L در خروجی هر 7406 قرار داده می-شود.در شکل ۲۶ ما مدار کلکتور باز را برای یک گیت OR و AND مجازی حاصل شده از همین طریق نشان داده‌ایم.



شکل 10-۲۶ - مدار کلکتور باز برای یک گیت OR و AND

در ضمن شکل بیانگر سمبل مرسوم برای نمایش گیت‌های OR و AND به‌صورت کلکتور باز است در شکل 2.26 منطق با یک نقطه مورد تأکید قرار گرفته است (L. یا H.) در دیاگرام‌های واقعی شما می‌توانید از این نماد استفاده کنید یا از آن صرف‌نظر کنید.

در حال حاضر گیت‌های کلکتور باز برای برخی تکنولوژی‌های خاص استفاده می‌شود این گیت‌ها در حال حاضر در طراحی مدارات ترکیبی کاربرد زیادی ندارد اما با این تکنولوژی قدرت منطق مختلط خودنمایی می‌کند. که موضوع را با یک مثال روشن می‌کنیم. فرض کنید می‌خواهیم عبارت زیر را پیاده‌سازی کنیم $out = AB.C + D + E$ با استفاده از 7406. در منطق مختلط مطابق روش زیر عمل می‌کنیم. ما نیاز به یک گیت OR سه ورودی داریم که یکی از ورودی‌ها خود حاصل AND سه ورودی

نسخه پیش‌نویس Draft

دیگر است که هر یک از این دو گیت خود نیاز به مقاومت بالابر دارند در خروجی گیت AND مقدار T=H است که برای اینکه وارد گیت OR شود مقدار آن T=L می‌شود برای اینکار می‌توان از یک 7406 بهره برد حال با استفاده از مطالب بالا به مداری مشابه شکل ۲۷ می‌رسیم.

این طراحی تنها به یک تراشه و دو مقاومت بالا کش[1] نیاز دارد.

توابعی که کمتر مورد استفاده قرار گرفته‌اند:

برای به پایان رساندن بهره خود به استفاده از متدهای منطق مختلط برای توابع منطقی ساده امتحان می‌کنیم. چگونه می‌توانیم به تابع منطقی یک تراشه با بیش از دو ورودی پی ببریم. طراحان سخت‌افزار از گیت‌های AND و OR و NOT و EOR و COINCIDENCE به طور گسترده استفاده می‌کنند دیگر توابع با دو متغیر ورودی، کمتر مورد استقبال طراحان قرار می‌گیرد یک جدول درستی برای یک تابع با دو ورودی چهار ردیف دارد و ۱۶ تابع خروجی متفاوت به‌واسطه این دو ورودی می‌تواند ایجاد شود. جدول ۱ شامل ارزش‌های درستی برای این ۱۶ تابع است. (از Z0 تا Z15) که شما تاکنون با چند تا از این توابع آشنا شده‌اید Z1 کار تابع AND را انجام می‌هد. Z6 تابع EOR است. Z7 تابع OR و Z9 نیز نشانگر COINCIDENCE.

جدول 10-۴ – ارزش‌های درستی برای ۱۶ تابع ممکن دو متغیره

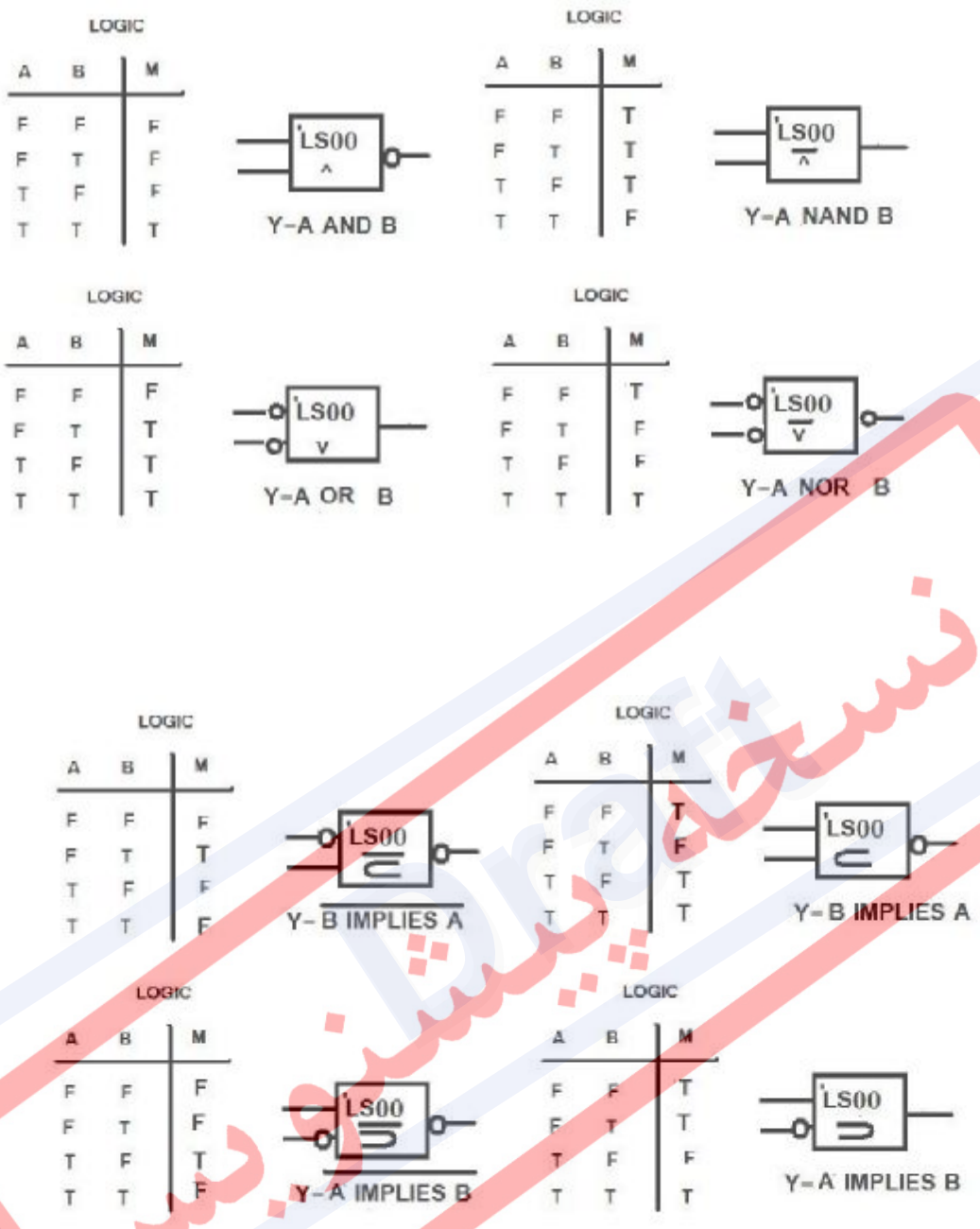
A	B	Z0	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8	Z9	Z10	Z11	Z12	Z13	Z14	Z15
F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T
F	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T
T	F	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T
T	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T

می‌دانیم که منطق مختلط راه‌حل‌های گوناگونی جهت مدیریت این توابع دارد توابع مختلف موجود در جدول را می‌توان با منطق مختلط بیان کرد مثلاً Z0 و Z15 به ترتیب بیانگر توابع همیشه غلط و همیشه درست هستند که به‌آسانی با وصل کردن یک سیم به یک منبع ولتاژ مناسب حاصل می‌آیند Z3

[1] Pull up

و Z5 و Z10 و Z12 توابعی حقیقی از یک عبارت تک متغیره هستند که عملیاتی چون نقیض عبارت و ... در این دسته قرار دارند اما شش تابع باقیمانده شباهت کمتری به توابع معمول و در دسترس و موجود در بازار دارند. ما گیت 74LS86 را یافتیم که دو تابع منطقی EOR و COINCIDENCE که هر کدام را با چهار راه مختلف پیاده‌سازی می‌کرد و هر یک از این هشت شیوه قرارداد خاصی برای ورودی و خروجی و به تبع آن حالات خاصی از دو گیت را فراهم می‌آورد در حالیکه ما فقط از دو تا از این موارد استفاده می‌کنیم. تابع منطقی پیاده‌سازی شده توسط ۶ حالت دیگر چه می‌شود؟ تمام این هشت حالت در شکل 2.28 نشان داده شده است با بررسی جدول مربوطه می‌توان دریافت که گیت 74LS00 خود به تنهایی می‌تواند هشت تابع منطقی مختلف را ایجاد کند در واقع می‌توان گفت منطق مختلط با دادن آزادی به طراح دست او را در انتساب مقادیر ولتاژ به مقادیر منطقی باز گذاشته است در شکل 2.28 توجه کنید که چگونه منطق مختلط تابع NAND را ایجاد می‌کند بدون دایره‌ای تغییردهنده پلاریته در خروجی و ورودی. حال آنکه در منطق مثبت تابع NAND با 74LS00 و با یک دایره تغییر پلاریته دهنده ایجاد شده است. بحث ما در مورد مقدمات و متدهای منطق مختلط در اینجا به اتمام می‌رسد حال شما با درک این منطق می‌توانید گیت مورد نیاز خود را بنا به قراردادهای مربوطه بسازید. در فصل بعدی ما ساختارهای ترکیبی پیچیده‌تری را شرح می‌دهیم هر کدام از آن‌ها دارای یکی یا بیشتر سمبل از منطق مختلط برای اضافه شدن به دیاگرام طراحی شماست.

پیش‌نویس Draft



نسخه پیش نویس

شکل 10-۲۷ - هشت تابع دو متغیره

فصل ۱۱

آزمون مدارات منطقی

نسخه پیش‌نویس
Draft

۱۱-۱ مقدمه

عمل آزمون[1] بخش حساس و بحران زای فرآیند تولید برای مدارهای منطقی دیجیتالی است. هر قسمت به هنگام ترک کارخانه باید به دقت چک شود تا از عملکرد صحیح آن اطمینان حاصل گردد. تلاش‌های لازم برای ایجاد رویهٔ آزمون و آزمون واقعی مدار، قیمت کل محصول و زمان حمل آن را برای بازار مصرف به طور مؤثری افزایش خواهد داد. در نتیجه، تهیهٔ استراتژی عیب یابی و اعمال روند آزمون باید حتی الامکان کارا باشد تا بتواند هر نوع عیبی را در مدار تشخیص دهد.

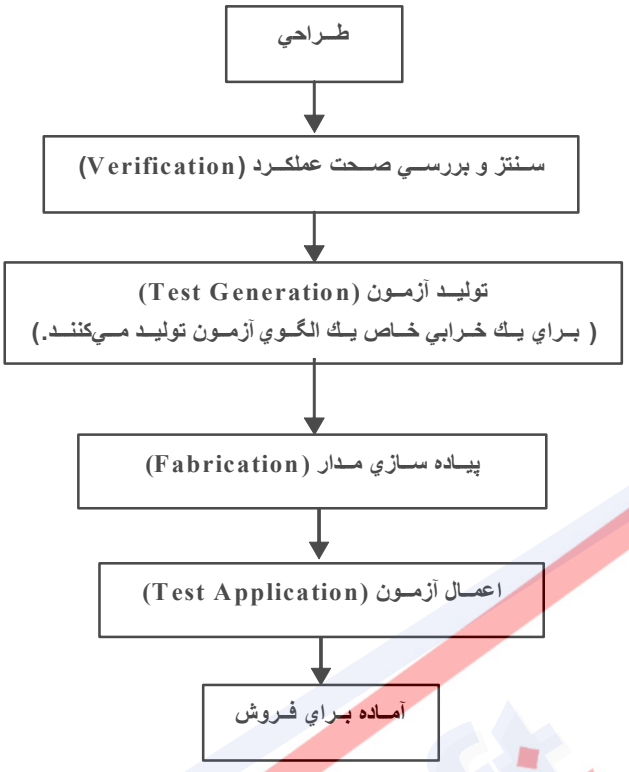
- انواع مشکلات در عملکرد مدار:

  - مشکلات مرحله ساخت- هر قسمت به هنگام ترک کارخانه باید به دقت چک شود تا از عملکرد صحیح آن، طبق طرح، اطمینان حاصل گردد. (در کارخانهٔ سازنده، قبل از تحویل به مشتری)
  - در مرحلهٔ کار مدار (توسط کاربر یا توسط مدار)

در ادامه به توضیح در مورد جایگاه آزمون مدار در روال طراحی یک مدار می‌پردازیم. در شکل زیر این روال مشخص شده است. همانطور که قبلا اشاره شد، بعد از طراحی مدار برای تبدیل آن به عناصر و گیت‌های واقعی، آن را سنتز می‌کنند و صحت عملکرد[2] مدار را به کمک شبیه سازی با ابزارهای خاص انجام می‌دهند. بعد از اتمام این قسمت و قبل از شروع روال ساخت، بر اساس خرابی‌های مشخص، الگوهای آزمون براساس مدار سالم تهیه می‌شود. بعد از ساخت مدار و برای اطمینان از اینکه در روال ساخت مشکلی برای مدار ایجاد نشده است، الگوهای تولیدی را به مدار اعمال می‌کنند. در صورت موفقیت این آزمون، مدار برای فروش و استفادهٔ کاربر ارسال می‌شود. در غیراینصورت، مدار خراب است و قابل استفاده نیست و معمولاً برای بازیافت به قسمت‌های دیگر ارسال می‌شود.

[1] Test
[2] Verification

نسخه پیش‌نویس Draft



۱۱-۲ اعمال اصلی در آزمون مدار

- **تشخیص خرابی**[1]: روند تعیین وجود و یا عدم وجود خرابی. به مجموعه‌ای از ورودی‌های مدار منطقی که برای تشخیص خرابی در مدار به کار می‌روند، مجموعهٔ آزمون تشخیص خرابی[2] (FDTS) می‌گویند.
- **مکان‌یابی خرابی**[3]: فرآیندی است که خرابی موجود در دستگاه را مشخص می‌نماید. به مجموعه‌ای از ورودی‌های مدار منطقی که برای یافتن محل خرابی استفاده می‌شود، مجموعهٔ آزمون مکان خرابی[4] (FLTS) می‌گویند.

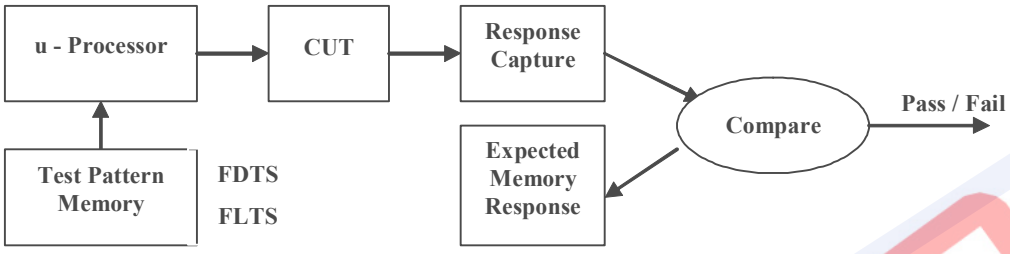
در اکثر مدارات، با تشخیص خرابی سروکار داریم و مکان‌یابی فایده‌ای برای ما ندارد. چون مدارات مورد نظر ما بسیار فشرده بوده و در این مدارات قسمت خراب را نمی‌توان تعویض کرد. در ادامه، قسمت آزمون مدارات را بصورت مفصل‌تری بررسی می‌نماییم.

[1] Fault Detection
[2] Fault Detection Test Set
[3] Fault Location
[4] Fault Location Test Set

نسخه پیش‌نویس Draft

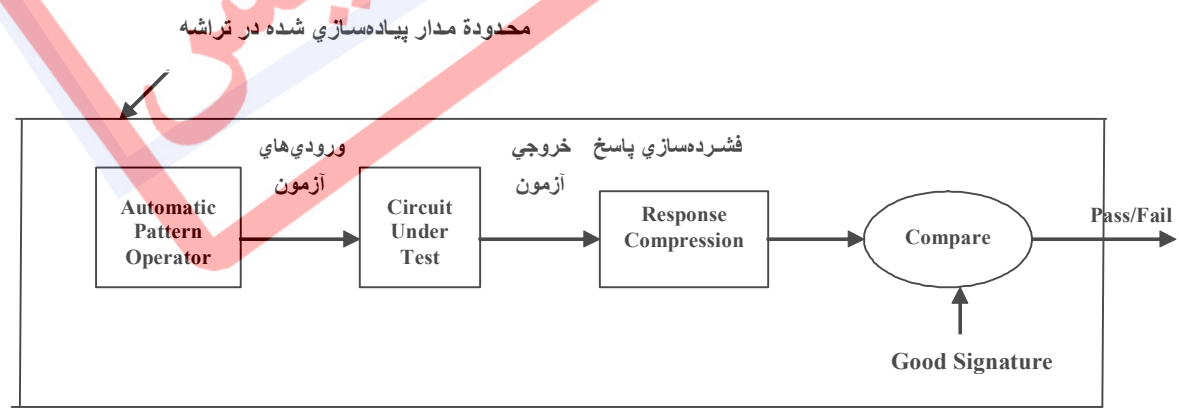
۱۱-۳ نحوهٔ آزمون مدارهای منطقی

آزمون مدارهای منطقی به دو شکل انجام می‌شود، یا کاربر مدار را شخصاً آزمون می‌کند یا خود مدار این امکان را دارد که خود را آزمون کند.



نحوهٔ کار در اینجا بدین صورت است که لیستی از ورودی‌ها را در اختیار داریم و جواب‌های مورد انتظار را نیز در اختیار داریم. با اعمال ورودی‌ها، تعدادی جواب از مدار حاصل می‌شود که با مقایسهٔ آنها با جواب‌های درست، به خرابی یا درستی مدار پی می‌بریم. غالباً، ورودی‌های آزمون در حافظه ذخیره شده و به هنگام اعمال به مدار تحت آزمون[1] (CUT)، به وسیلهٔ ریزپردازنده فراخوانی می‌گردند. روش بالا بیشتر در کارخانه‌های سازندهٔ مدار متداول است، چون به تجهیزات خاصی نیاز دارد.

اما روشی که امروزه متداول است بدین صورت است که بخشی از مدارات مورد نیاز برای آزمون را در درون خود مدار قرار می‌دهیم. الگوهای آزمون در این مدارها، به کمک مدارهای خاصی و بصورت تصادفی تولید می‌شوند. در اینجا، خروجی مدار تحت آزمون به کمک یک مدار دیگر فشرده شده و بصورت یک امضا در می‌آید. سپس، این خروجی با امضای صحیح که قبلاً تهیه شده مقایسه می‌شود و مشخص می‌شود که مدار دچار خرابی شده یا خیر. این مدارها Built In Self Test (BIST) نام دارند.



[1] Circuit Under Test

۱۱-۴ مدل‌های خرابی

برای بررسی خرابی‌ها، باید بتوانیم آنها را مدل کنیم. در ادامه، دو مدل معروف را بررسی می‌نماییم:

- **مدل اول**

  - **خرابی دائمی**[1]: خرابی‌ای است که با زمان تغییر نمی‌کند. به عبارت دیگر بعد از ایجاد خرابی، خرابی باقی می‌ماند. مثل سوختن یک مقاومت.
  - **خرابی گذرا**[2]: خرابی‌ای است که رخ داده اما با گذشت زمان رفع می‌گردد. مثل ورود یک نویز. زمان اتفاق افتادن آن معلوم نیست. در نتیجه آزمون مدار را دشوار می‌کند.

از بین این دو نوع خطا، فقط می‌توانیم خطاهای دائمی را تشخیص دهیم.

- **مدل دوم**

  - **خرابی منطقی**[3]: اتصال مستقیم به صفر یا یک که سبب می‌شود وسیلهٔ منطقی مفروض، کلاً به نحوی دیگر عمل کند.
  - **خرابی غیرمنطقی**[4]: مثل قطع شدن مدار در V_{DD}.

مدل خرابی دائمی، بیشتر رخ می‌دهد و تشخیص آن نیز ساده‌تر است.

۱۱-۵ آزمون خرابی‌های چسبنده

برای مطالعهٔ اثر خرابی‌ها روی مدار منطقی، باید یک مدل برای خرابی ایجاد کرد. یک مدل مرسوم و مفید برای نمایش خرابی‌ها، مدل خرابی چسبنده[5] می‌باشد. در این مدل، خرابی در مدار به صورت سیمی نمایش داده می‌شود که به منطق صفر[6] (s-a-0) و یا به منطق یک[7] (s-a-1) متصل است. برای بررسی این خطاها، به بیان چند تعریف می‌پردازیم.

$f(x_n)$: مدار بدون خرابی.

---

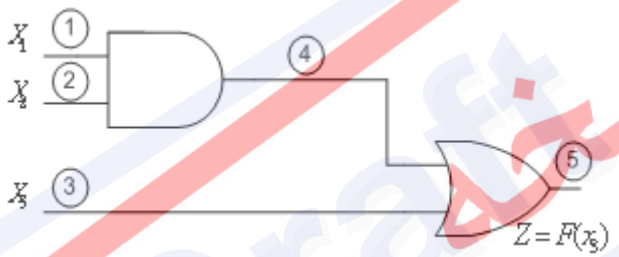
[1] Permanent
[2] Intermittent
[3] Logical
[4] non-Logical
[5] Stuck at Fault
[6] Stuck at Zero
[7] Stuck at One

نسخه پیش‌نویس Draft

$f^{p/d}(x_n)$: همان مدار با خرابی P/d را نمایش می‌دهد که P شمارهٔ سیم و d=0 برای s-a-0 و d=1 برای s-a-1 و n تعداد متغیرهای ورودی است. در واقع، تنها با یک خطا در کل مدار برخورد داریم.

$$f(x_n)=(x_1,x_2,\ldots,x_n)$$
$$f(x_n)=\ fault\ \ free$$
$$f^{p/d}(x_n)=\begin{cases} P: wire\ label \\ d:\begin{cases} 0: s-a-0 \\ 1: s-a-1 \end{cases} \end{cases}$$

مثـال 1)



برای هر سیم یک برچسب در نظر گرفته می‌شود. در ادامه، تابع سالم به همراه دو تابع با خرابی‌های متفاوت را می‌بینید.

$$f(x_3)=x_1x_2+x_3$$
$$f^{3/0}(x_3)=x_1x_2$$
$$f^{2/1}(x_3)=x_1+x_3$$

۱۱-۶ تولید الگوهای آزمون

فرایندی است که برای یک خرابی خاص یک الگوی آزمون تولید می‌کند. تولید آزمون را به صورت روند تعیین آزمون برای عیبی مفروض در یک شبکه تعریف می‌کنیم. هرگاه بیش از یک آزمون وجود داشته باشد، همهٔ آزمون‌هایی که می‌توانند برای تشخیص عیب به کار روند معلوم می‌گردند.

۱۱-۶-۱روش اول: آزمون کامل

فرض کنید $f(x_n)$ خروجی یک شبکهٔ منطقی باشد که برای تشخیص عیب احتمالی آزمون می‌شود. باید ورودی‌هایی را به مدار اعمال کنیم که خروجی آنها با خروجی حالت معمول فرق کند. واضح

نسخه پیش‌نویس Draft

است که تمام مجموعهٔ 2^n ورودی ممکن یک شبکه می‌توانند به عنوان یک FDTS به کار روند. به کار بردن همهٔ ورودی‌های ممکن برای آزمون یک مدار را آزمون کامل[1] می‌نامند. در صورتیکه تعداد سیم‌های ورودی مدار زیاد باشد، انجام این آزمون غیر ممکن است. با این وجود، این روش همانطور که در مثال زیر دیده می‌شود، بسیار سر راست است. اغلب شبکه‌ها، بدون استفاده از روش آزمون کامل نیز به طور مناسبی قابل آزمون هستند.

جدول 11-۱ – آزمون کامل

x_1	x_2	x_3	بدون خطا $f(x_3)$	$f^{1/0}(x_3)$	$f(x_3)^{3/0}$	$f^{1/0}(x_3) \oplus f(x_3)$
0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	0
1	0	1	1	1	0	0
1	1	0	1	0	1	1
1	1	1	1	1	1	0

نقاطی که خروجی مدار با حالت طبیعی فرق دارد را انتخاب کرده و از میان آنها یکی را برمی‌گزینیم. در جدول بالا سطری که علامت گذاری شده، بعنوان الگوی آزمون انتخاب شده است، بنابراین الگوی آزمون $X_1X_2X_3 = 110$ می‌باشد.

۱۱-۶-۲ روش دوم: روش OR انحصاری

روش دیگری که وجود دارد و برای مداراتی که تعداد زیادی ورودی دارند به کار می‌رود، انجام عمل XOR به صورت جبری است. فرض کنید $f(x_n)$ شبکه ای بدون عیب را نمایش دهد و p/d عیبی باشد که آزمون‌ها برای آن بدست خواهند آورد.

$$f(x_n^j) \oplus f^{p/d}(x_n^j) = 1$$

برای مثال داریم:

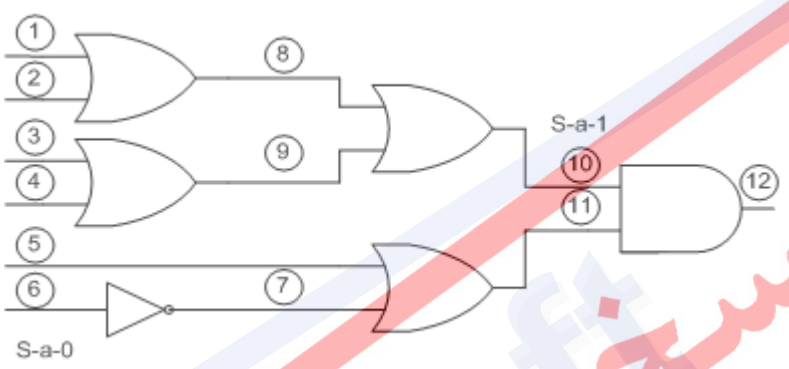
[1] Exhaustive Testing

نسخه پیش‌نویس Draft

$$\begin{cases} f^{3/0} = x_1 x_2 \\ f = x_1 x_2 + x_3 \end{cases} \Rightarrow f \oplus f^{3/0} = (x_1 x_2 + x_3) \oplus x_1 x_2 = (\bar{x}_1 + \bar{x}_2) x_3 = \bar{x}_1 x_3 + \bar{x}_2 x_3$$

با توجه به نتیجهٔ حاصل شده از XOR ، ورودی‌هایی را انتخاب می‌کنیم که این حاصل را یک کنند.

مثـال 2)



$$g = (x_1 + x_2 + x_3 + x_4)(x_5 + \bar{x}_6)$$
$$G^{6/0} = g \oplus g^{6/0} = \left[(x_1 + x_2 + x_3 + x_4)(x_5 + \bar{x}_6)\right] \oplus \left[x_1 + x_2 + x_3 + x_4\right]$$
$$= x_1 \bar{x}_5 x_6 + x_2 \bar{x}_5 x_6 + x_3 \bar{x}_5 x_6 + x_4 \bar{x}_5 x_6 \rightarrow |\times\times\times 0|$$

به عبارت دیگر، باید ورودی X_5 برابر صفر بوده و ورودی X_6 برابر یک باشد. در صورتیکه خروجی برابر یک شود، یعنی در ورودی X_6 نقصی از نوع s-a-0 داریم.

۱۱-۶-۳ روش سوم: روش حساس‌سازی مسیر

روش حساس سازی مسیر[1] از قدیمی‌ترین روشهای آزمون است. در این روش، ابتدا مسیری که شامل خرابی مورد نظر باشد را حساس می‌کنیم. یعنی ورودی‌ها را به گونه‌ای اعمال می‌کنیم که مقدار ورودی مسیر حساس شده مستقیماً در خروجی تأثیر بگذارد. پس از این مرحله، خرابی را تحریک می‌کنیم یعنی ورودی‌ای اعمال می‌کنیم که در صورت خرابی، مدار خروجی‌ای مخالف حالت عادی تولید کند و از $\oplus$ کردن نتیجهٔ حاصل و نتیجهٔ حالت عادی، پی به خرابی مدار می‌بریم.

[1] Path Sensitizations

نسخه پیش‌نویس

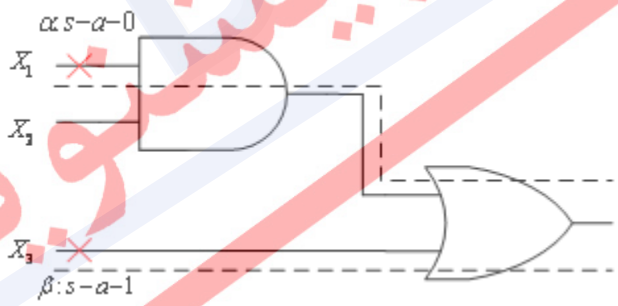
۱۱-۶-۳-۱ الگوریتم حساس سازی مسیر

۱- حساس کردن خرابی: یعنی برای s-a-0 تحریک "1" و برای s-a-1 تحریک "0" را اعمال می‌کنیم.

۲- سپس باید کاری کنیم که خرابی در یک مسیر پیشرو تا خروجی انتقال یابد. مثلاً فرض کنید نام خرابی a باشد. حال اگر این خرابی ورودی یک گیت AND باشد، باید بقیهٔ ورودی‌ها "1" باشند تا خروجی AND وابسته به a شود. بدین معنی که اگر a برابر "1" بود، خروجی "1" و اگر برابر "0" بود، خروجی صفر شود. در مورد گیت NAND نیز همین عملیات را انجام می‌دهیم با این تفاوت که خروجی a' خواهد بود. برای گیت OR باید سایر ورودی‌ها را برابر "0" قرار دهیم تا خروجی برابر a شود. برای گیت NOR خروجی a' خواهد بود.

۳- حال تحریک‌های انجام شده برای حساس سازی مسیر را از خروجی به طرف ورودی‌های اصلی (در یک مسیر رو به عقب) دنبال می‌کنیم تا مقادیر همهٔ ورودی‌های اصلی مشخص شود. در نهایت، ورودی‌های بدست آمده برابر الگوهای آزمون هستند.

مثـال 3)



- برای خرابی α:

قدم اول: ورودی $x_1=1$ باشد تا خرابی تحریک شود.

قدم دوم: از مسیر علامت گذاری شده، خرابی را به خروجی هدایت می‌کنیم. بنابراین باید $x_2=1$ باشد و در طبقهٔ بعد نیز $x_3=0$ تا خرابی به خروجی منتقل شود.

قدم سوم: چون همهٔ ورودی‌های اصلی مشخص شده‌اند، این قدم انجام نمی‌شود و الگوی آزمون برابر $x_1x_2x_3=110$ خواهد بود.

نسخهٔ پیش‌نویس Draft

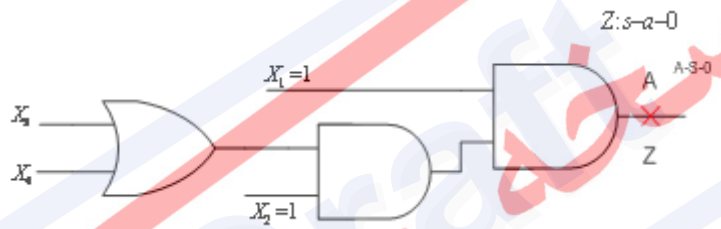
- برای خرابی β:

قدم اول: ورودی $x_3=0$ باشد تا خرابی تحریک شود.

قدم دوم: از مسیر علامت گذاری شده، خرابی را به خروجی هدایت می‌کنیم. بنابراین باید خروجی گیت AND صفر باشد تا خرابی به خروجی منتقل شود.

قدم سوم: چون باید خروجی گیت AND صفر باشد، باید ورودی‌ها را طوری مقداردهی کنییم که خروجی آن صفر شود. بنابراین سه حالت مختلف خواهد شد و الگوی آزمون برابر $x_1x_2x_3=\{000,010,100\}$ خواهد بود.

مثـال4)



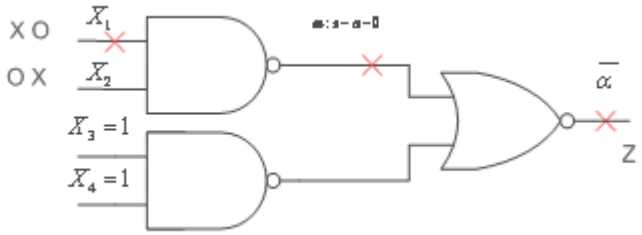
قدم اول: برای تحریک خرابی، باید خروجی گیت AND یک باشد. یعنی خروجی مدار یک باشد.

قدم دوم: چون خرابی در خروجی مدار است، این قدم انجام نمی‌شود.

قدم سوم: عمل برگشت به عقب را انجام می‌دهیم. چون قرار است خروجی یک باشد، هر دو ورودی گیت AND خروجی باید برابر با یک باشد که در نتیجه $x_2=1$. چون خروجی AND طبقهٔ وسط نیز باید یک باشد، بنابراین $x_1=1$ و با توجه به گیت OR ، حداقل یکی از پایه‌های x_3 یا x_4 نیز باید برابر یک باشد. بنابراین الگوهای آزمون برابر $x_1x_2x_3x_4=\{1111,1101,1110\}$ خواهد شد.

نسخه پیش‌نویس Draft

مثال5)



در این مدار برای ساختن مسیر کافی است که $x_3x_4=11$ باشد و برای تحریک مدار ورودی‌های x0 یا 0x به x_2x_1 اعمال می‌کنیم. پس الگوهای آزمون به صورت {1011,0011,0111} خواهد شد. اگر به ساختار مدار دقت کنید، خواهید دید که الگوی آزمون 0111 ، خرابی Z: s-a-1 را نیز تشخیص می‌دهد.

۱۱-۷ نکات تکمیلی تولید آزمون در مدارهای ترکیبی

۱۱-۷-۱ مدارهای Fanout Free

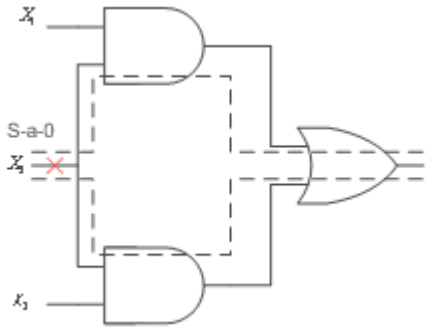
مداراتی که در آنها ورودی‌ها یا خطوط میانی فقط به ورودی یک گیت متصل هستند، مدارهای Fanout Free نامیده می‌شوند. برای آزمون کامل این مدارات، فقط نیاز است که آزمون‌های مورد نیاز برای خرابی‌های ممکن روی ورودی‌های اصلی تولید شود. به دلیل منحصر به فرد بودن مسیر حساس شده، روش حساس سازی مسیر برای این مدارات به خوبی عمل می‌کند.

۱۱-۷-۲ مدارهای Non-Fanout Free

مداراتی هستند که در آنها ممکن است سیگنال‌ها به بیش از یک گیت داده شوند. روش حساس سازی مسیر ممکن است برای این مدارات به خوبی عمل نکند.

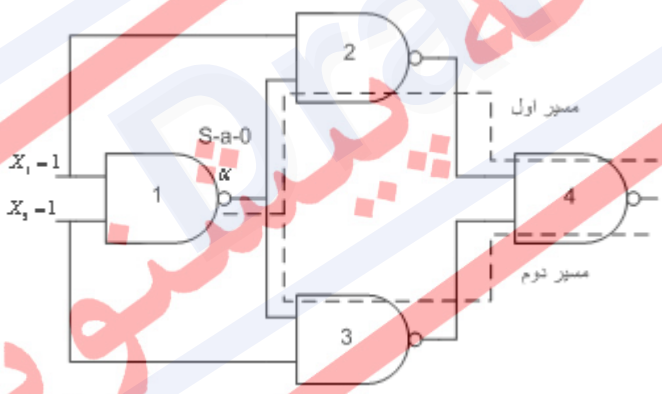
نسخه پیش‌نویس Draft

مثـال6)



این مدار چون دارای شاخه است، پس Fanout Free نیست. بنابراین در به کار بردن روش حساس سازی مسیر باید دقت کرد. با اعمال روش حساس سازی مسیر خواهید دید که از مسیر بالا الگوی 110 و از مسیر پایین الگوی 011 تولید می‌شود. بنابراین در این مثال مشکلی وجود نداشت.

مثال7)



در این مثال نیز دو مسیر متفاوت برای هدایت خرابی به خروجی وجود دارد که در زیر اشاره شده است:

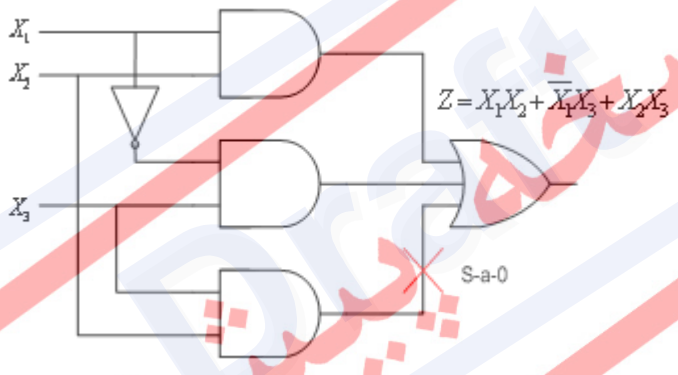
- مسیر اول: برای تحریک خرابی باید خروجی گیت یک، صفر باشد و بنابراین $x_1x_2=11$ خواهد شد. با این حال خرابی به راحتی از گیت دو عبور می‌کند. اما برای عبور از گیت چهار باید ورودی دیگر این گیت که در واقع خروجی گیت سه است، یک باشد. با توجه به اینکه باید $x_2=0$ باشد تا این عمل انجام شود و این مقدار موجب تناقض با مقدار اولیه می‌شود، نمی‌توان از این مسیر الگوی آزمون را پیدا نمود.

نسخه پیش‌نویس

- مسیر دوم: اگر به ساختار مدار دقت نمایید، خواهید دید که مدار کاملا متقارن است و از مسیر دو نیز مانند مسیر اول به بن بست می‌رسیم.
- انتخاب هر دو مسیر: اگر خرابی را از هر دو مسیر به سمت خروجی هدایت نماییم، می‌توانیم این مشکل را حل نماییم. بنابراین الگوی آزمون $x_1x_2=11$ خواهد شد.

۱۱-۷-۳ مدارهای دارای افزونگی

مدارهایی هستند که با انتساب یک سیگنال به 0 یا 1 تابع مدار تغییر نمی‌کند. در نتیجه اگر یک خرابی متناظر با آن بیت روی خط مذکور اتفاق بیفتد، آن خرابی قابل تشخیص نیست. گیت‌های افزونه[1] معمولاً برای جلوگیری از مخاطره به مدار اضافه می‌شوند که خود برای آزمون دردسر ساز هستند. برای مثال، مداری که در بحث مخاطره در مورد آن بحث نمودیم در شکل ۱ مشخص شده است.



شکل 11-۱ – عدم تشخیص خرابی s-a-0 به علت وجود گیت‌های افزونه

با توجه به توضیحاتی که در مثال قبل اشاره شد، نمی‌توان خرابی s-a-0 را که در شکل ۱ نشان داده شده تشخیص داد. به این خرابی‌ها، خرابی غیرقابل تشخیص[2] گفته می‌شود. این خرابی‌ها گرچه در عملکرد مدار خللی ایجاد نمی‌کنند اما هنگامی که در حال شناسایی خطاهای دیگر هستیم مشکلاتی را برای ما بوجود می‌آورند.

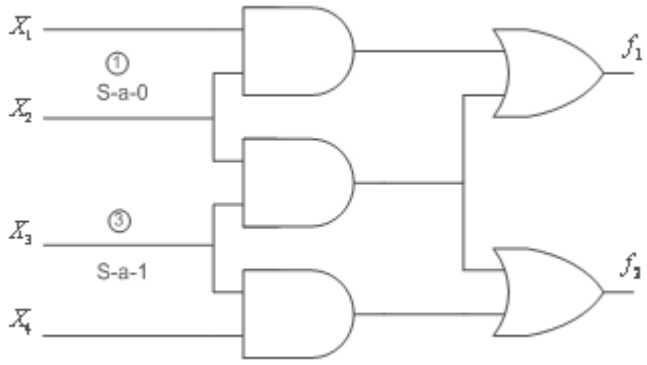
۱۱-۷-۴ مدارهای با چند خروجی

در مدارهای این چنینی، اثر خرابی برای تک تک خروجی‌ها باید به صورت جداگانه بررسی شود که یک نمونه آن در مثال زیر آمده است.

---

[1] Redundant
[2] Undetectable Fault

نسخه پیش‌نویس



$$F^{1/0} = F_1^{1/0} + F_2^{1/0}$$
$$= (F_1 \oplus f^{1/0}) + (F_2 \oplus f^{1/0})$$
$$= \left[(x_1x_2 + x_2x_3) \oplus (x_2x_3)\right] +$$
$$\left[(x_2x_3 + x_3x_4) \oplus (x_2x_3 + x_3x_4)\right]$$
$$= x_1x_2\bar{x}_3 + 0 = x_1x_2\bar{x}_3$$

$$test = 110x = \begin{cases} 1100 \\ 1101 \end{cases}$$

$$F^{3/1} = F_1^{3/1} + F_2^{3/1} = (f_1 \oplus f_1^{3/1}) + (f_2 \oplus f_2^{3/1})$$
$$= \left[(x_1x_2 + x_2x_3) \oplus (x_1x_2 + x_2)\right] + \left[(x_2x_3 + x_3x_4) \oplus (x_2 + x_4)\right]$$
$$= \left[\bar{x}_1x_2\bar{x}_3\right] + \left[x_2\bar{x}_3 + \bar{x}_3x_4\right] = x_2\bar{x}_3 + \bar{x}_3x_4$$

در حالت کلی برای مدارهای با m خروجی داریم.

$$F^{p/d} = F_1^{p/d} + F_2^{p/d} + \cdots + F_m^{p/d}$$

نسخه پیش‌نویس

۱۱-۷-۵ تفاضل بولی[1]

روش دیگر تولید تست برای تشخیص خرابی در مدارهای ترکیبی استفاده از روش تفاضل بولی است . برای اشنایی با این روش ابتدا تعریف و خصوصیات تفاضل بولی را بررسی می کنیم.

تعریف:

تفاضل بولی تابع F(x) نسبت به X_i به صورت زیر محاسبه می شود:

$$\frac{dF(X)}{dX_i}=\frac{dF(X_1,...,X_i,...,X_n)}{dX_i}=F(X_1,...,X_i,...,X_n)\oplus F(X_1,...,\overline{X_i},...,X_n)$$

به وضوح دیده می شود که وقتی $F(X_1,...,X_i,...,X_n)$ نامساوی $F(X_1,...,\overline{X_i},...,X_n)$ باشد $\frac{dF(X)}{dX_i}$ برابر ۱ می شود. درغیر این صورت $\frac{dF(X)}{dX_i}$ برابر ۰ می شود.

تعریف:

F(X) از X_i مستقل است اگر و فقط اگر $\frac{dF(X)}{dX_i}$ برابر ۰ باشد.

قوانین تفاضل بولی:

۱- $$\frac{dF(X)}{dX_i}=\frac{d\overline{F(X)}}{dX_i}$$

۲- $$\frac{d[F(X).G(X)]}{dX_i}=F(X).\frac{dG(X)}{dX_i}\oplus G(X)\frac{dF(X)}{dX_i}\oplus\frac{dF(X)}{dX_i}.\frac{dG(X)}{dX_i}$$

۳- $$\frac{d[F(X)+G(X)]}{dX_i}=\overline{F(X)}.\frac{dG(X)}{dX_i}\oplus\overline{G(X)}\frac{dF(X)}{dX_i}\oplus\frac{dF(X)}{dX_i}.\frac{dG(X)}{dX_i}$$

۴-اگر F(X) از X_i مستقل باشد انگاه خواهیم داشت:

$$\frac{d[F(X).G(X)]}{dX_i}=F(X).\frac{dG(X)}{dX_i}$$

$$\frac{d[F(X)+G(X)]}{dX_i}=\overline{F(X)}.\frac{dG(X)}{dX_i}$$

به منظور استفاده از تفاضل بولی برای تشخیص خرابی به قضایای زیر توجه کنید:

[1] Boolean Difference

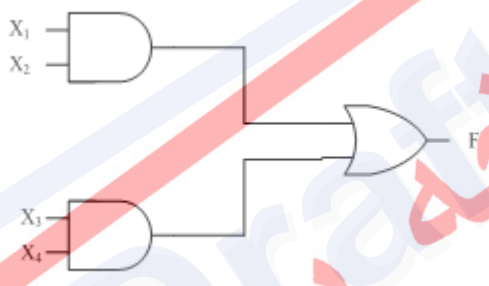
نسخه پیش‌نویس Draft

۱-اگر $\frac{dF(X)}{dX_i}$ برابر ۰ باشد خرابی روی X_i هیچ تاثیری روی خروجی مدار ندارد.

۲-اگر $\frac{dF(X)}{dX_i} \neq 0$ انگاه برای پیدا کردن ورودی تست S-a-0 بایستی $X_i.\frac{dF(X)}{dX_i}$ را برابر ۱ قرار داد.

۳-اگر $\frac{dF(X)}{dX_i} \neq 0$ انگاه برای پیدا کردن ورودی تست S-a-1 بایستی $\overline{X_i}.\frac{dF(X)}{dX_i}$ را برابر ۱ قرار داد.

مثال8)



الف)برای مدارشکل بالا تفاضل بولی را نسبت به X_3 محاسبه کنید.

ب)ورودی تست را در صورت وجود پیدا کنید.

حل قسمت الف:

$$\frac{dF(X)}{dX_3} = \frac{d(X_1X_2 + X_3X_4)}{dX_3}$$

$$= \overline{X_1X_2}\frac{d(X_3.X_4)}{dX_3} \oplus \overline{X_3X_4}\frac{d(X_1.X_2)}{dX_3} \oplus \frac{d(X_1.X_2)}{dX_3}.\frac{d(X_3.X_4)}{dX_3}$$

$$= \overline{X_1X_2}\frac{d(X_3.X_4)}{dX_3}$$

$$= \overline{X_1X_2}[X_3\frac{dX_4}{dX_3} \oplus X_4\frac{dX_3}{dX_3} \oplus \frac{dX_3}{dX_3}.\frac{dX_4}{dX_3}]$$

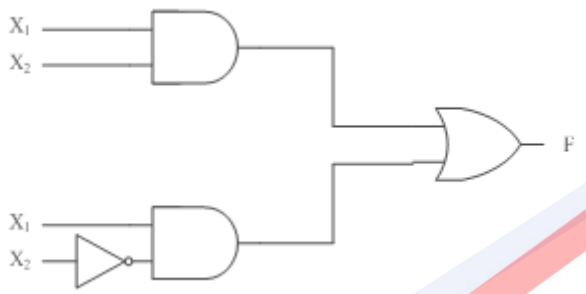
$$= \overline{X_1X_2}X_4$$

حل قسمت ب)

نسخه پیش‌نویس Draft

این بدان معناست که خرابی روی X_3 در صورتی که $\overline{X_1 X_2} X_4$ برابر ۱ باشد روی خروجی مدار اثر خواهد گذاشت.

مثال 9)



الف)برای مدارشکل زیر تفاضل بولی را نسبت به X_2 محاسبه کنید.

ب)ورودی تست را در صورت وجود پیدا کنید.

حل قسمت الف:

$$\frac{dF(X)}{dX_2} = \frac{d(X_1 X_2 + X_1 \overline{X_2})}{dX_2}$$

$$= \overline{X_1 X_2} \frac{d(X_1 \overline{X_2})}{dX_2} \oplus \overline{X_1 \overline{X_2}} \frac{d(X_1 . X_2)}{dX_2} \oplus \frac{d(X_1 . X_2)}{dX_2} . \frac{d(X_1 \overline{X_2})}{dX_2}$$

$$= \overline{X_1 X_2} X_1 \oplus \overline{X_1 \overline{X_2}} X_1 \oplus X_1$$

$$= X_1 \oplus X_1$$

$$= 0$$

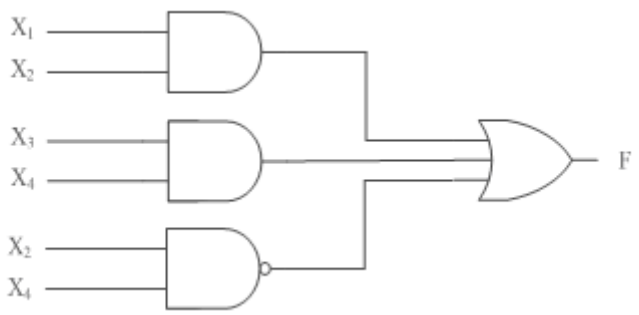
حل قسمت ب)

این بدان معناست که خرابی روی X_2 هیچ اثری روی خروجی مدار ندارد. پس ورودی تست نیز وجود ندارد.

نکته) از تفاضل بولی برای تشخیص خرابی های درونی مدار نیز می توان استفاده کرد. به مثال زیر توجه کنید.

نسخه پیش نویس
Draft

مثال 10)



فرض کنید در مدار زیر در نقطه h خرابی وجود دارد. ابتدا بیان کنید که ایا خروجی به h وابسته است یا نه ؟ سپس در صورت وابستگی ورودی تست برای تشخیص S-a-0 و S-a-1 را پیدا کنید.

حل :

مشخص است که:

$$F = h + (X_3 X_4 + \overline{X_2 X_4})$$

پس :

$$\frac{dF}{dh} = \overline{(X_3 X_4 + \overline{X_2 X_4})}.\frac{dh}{dh} \oplus \bar{h}.\frac{d(X_3 X_4 + \overline{X_2 X_4})}{dh} \oplus \frac{d(X_3 X_4 + \overline{X_2 X_4})}{dh}.\frac{dh}{dh}$$
$$= \overline{(X_3 X_4 + \overline{X_2 X_4})} \oplus 0 \oplus 0$$
$$= X_2 \overline{X_3} X_4$$

برای پیدا کردن ورودی تست S-a-0 :

$$h.\frac{dF}{dh} = X_1 X_2 . X_2 \overline{X_3} X_4 = X_1 X_2 \overline{X_3} X_4 = 1$$

در نتیجه ورودی تست برابر است با 1 0 1 1 .

و برای پیدا کردن ورودی تست S-a-1 :

$$\bar{h}.\frac{dF}{dh} = \overline{X_1 X_2} . X_2 \overline{X_3} X_4 = \overline{X_1} X_2 \overline{X_3} X_4 = 1$$

در نتیجه ورودی تست برابر است با 1 0 1 0 .

نسخه پیش نویس Draft

۱۱-۷-۶ بسط شانون

در بسط شانون هر تابع $F(x_1,\dots,x_i,\dots,x_n)$ را می‌توان به صورت زیر نوشت:

$$F(x_1,\dots,x_i,\dots,x_n)=\overline{x_i}F_i(x_i=0)+x_iF_i(x_i=1)$$

با استفاده از این تعریف، می‌خواهیم بردار آزمون را بدست آوریم. بردار آزمون از طریق رابطه $T=F\oplus F_\alpha$ بدست می‌آید. با جایگذاری بسط شانون در این رابطه به نتیجه زیر خواهیم رسید:

$$T=(x_iF_i(1)+\overline{x_i}F_i(0))\oplus F_\alpha$$

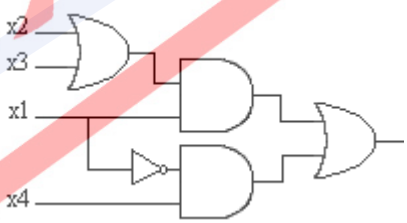
اگر ورودی xi دارای خطای s-a-1 باشد، xi همواره مقدار یک خواهد داشت. بنابراین می‌توانیم به جای xi در تابع $F_\alpha(X)$ مقدار یک قرار دهیم و عبارت بردار آزمون را به صورت زیر ساده کنیم:

$$T=(x_iF_i(1)+\overline{x_i}F_i(0))\oplus F_\alpha(1)$$
$$T=\overline{x_i}[F_i(0)\oplus F_i(1)]$$

این عمل را می‌توان در زمانی که xi دارای خطای s-a-0 است نیز تکرار نمود. در این حالت، عبارت بردار آزمون به صورت زیر درخواهد آمد:

$$T=x_i[F_i(0)\oplus F_i(1)]$$

در شکل زیر، ورودی x_1 دارای خطای s-a-0 است. با استفاده از عبارت بدست آمده می‌توان بردار آزمون را به صورت زیر محاسبه کرد.



شکل ۱۱-۲

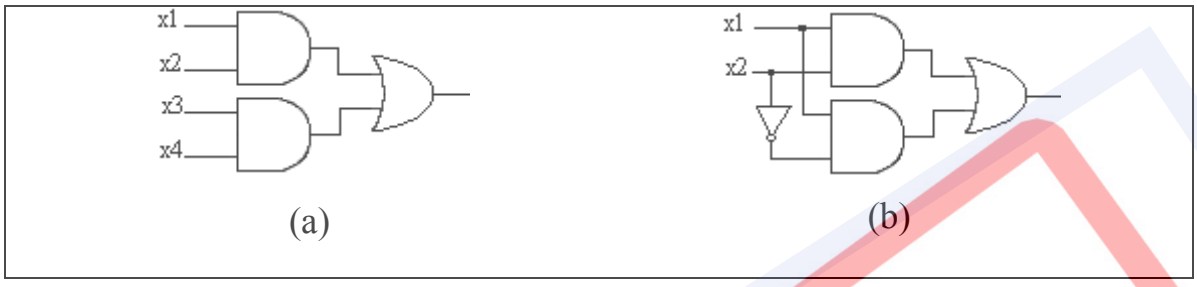
$$F(X)=x_1(x_2+x_3)+\overline{x_1}x_4$$
$$T=x_1\frac{dF(X)}{dx_1}=x_1(F_1(0)\oplus F_1(1))=x_1\overline{x_2}\,\overline{x_3}x_4+x_1x_2\overline{x_4}+x_1x_3\overline{x_4}$$

با این روش پنج بردار آزمون برای آزمون ورودی x_1 بدست آوردیم.

نسخه پیش‌نویس Draft

در مثال‌های زیر، چگونگی استفاده ازخواص اختلاف بولی در تعیین بردار آزمون نشان داده شده است.

مثال۱۱) به مدار شکل ۱۱-۳-a توجه کنید. می‌خواهیم اختلاف بولی تابع نسبت به X_3 را بدست آوریم. تابع پیاده‌سازی شده در این مدار $F(X) = x_1x_2 + x_3x_4$ است.



شکل ۱۱-۳

اگر ورودی X_3 دارای خطای s-a-0 باشد، عبارت $T = x_3 \frac{dF(X)}{dx_3}$ را محاسبه خواهیم کرد و اگر دارای خطای s-a-1 باشد، عبارت $T = \overline{x_3} \frac{dF(X)}{dx_3}$ بردارآزمون ما را مشخص خواهد نمود.

	$\frac{dF(X)}{dx_3} = \frac{d[x_1x_2 + x_3x_4]}{dx_3}$
با استفاده از خاصیت (۵):	$\overline{x_1x_2}\frac{d(x_3x_4)}{dx_3} \oplus \overline{x_3x_4}\frac{d(x_1x_2)}{dx_3} \oplus \frac{d(x_1x_2)}{dx_3}.\frac{d(x_3x_4)}{dx_3}$
با استفاده از خاصیت (۶):	$= \overline{x_1x_2}\frac{d(x_3x_4)}{dx_3}$
با استفاده از خاصیت (۴):	$= x_1x_2\left(x_3\frac{dx_4}{dx_3} \oplus x_4\frac{dx_3}{dx_3} \oplus \frac{dx_3}{dx_3}.\frac{dx_4}{dx_3}\right)$
با استفاده ازخواص (۶) و (۷) :	$= \overline{x_1x_2}x_4$

نتیجه بدست آمده نشان می‌دهد که تنها در صورتی که $\overline{x_1x_2}x_4 = 1$ باشد، وجود خطا بر روی X_3 باعث خراب شدن خروجی خواهد شد. به عبارت دیگر، اگر X_1، X_2 و یا هر دو برابر صفر باشند و X_4 برابر با یک باشد، خطای X_3 به خروجی منتقل خواهد شد.

نسخه پیش‌نویس Draft

مثال12) به مدار منطقی نشان داده شده در شکل ۱۱-۳-a توجه کنید. می‌خواهیم اختلاف بولی آن را نسبت به ورودی x_2 بدست آوریم. داریم:

	$\frac{dF(X)}{dx_2} = \frac{d[x_1 x_2 + x_1 \overline{x_2}]}{dx_2}$
با استفاده از خاصیت (۵):	$= \overline{x_1 x_2} \frac{d(x_1 \overline{x_2})}{dx_2} \oplus \overline{x_1 x_2} \frac{d(x_1 x_2)}{dx_2} \oplus \frac{d(x_1 x_2)}{dx_2} . \frac{d(}{}$
با استفاده ازخاصیت (۱)، (۷) و (۸):	$= \overline{x_1 x_2} x_1 \oplus \overline{x_1 x_2} x_1 \oplus x_1$
	$= x_1 (\overline{x_1 x_2} \oplus \overline{x_1 x_2}) \oplus x_1$
	$= x_1 (\overline{x_1 x_2} x_1 \overline{x_2} + \overline{x_1 x_2} x_1 x_2) \oplus x_1$
	$= x_1 (x_1 \overline{x_2} + x_1 x_2) \oplus x_1$
	$= x_1 \oplus x_1$
	$= 0$

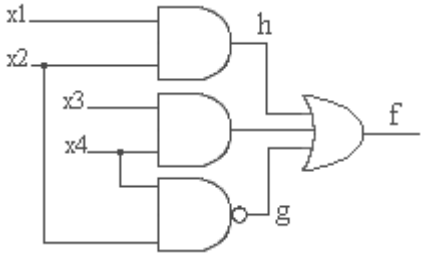
به این ترتیب اگر خطایی در ورودی x_2 رخ دهد تأثیری بر روی خروجی نخواهد گذاشت. در واقع این نتیجه نشان می‌دهد که $F(X)$ تابعی از ورودی x_2 نیست. با ساده سازی تابع خروجی این مطلب اثبات می‌شود($F(X) = x_1 x_2 + x_1 \overline{x_2} = x_1$).

تا کنون روش اختلاف بولی را برای استخراج آزمون در زمانی‌که ورودی دارای خطا باشد به کارگرفته‌ایم. اما از این روش می‌توان برای آزمون خطوط داخلی مدار نیز کمک گرفت.

فرض کنید یک مدار ترکیبی داریم که تابع $F(X)$ را پیاده‌سازی می‌کند، همچنین فرض می‌کنیم که h یک سیم داخلی در این مدار باشد. برای پیدا کردن آزمون مناسب برای h، F را به صورت تابعی از h بیان می‌کنیم. به عبارت دیگر $F(x_1, x_2, ..., x_n, h)$ که خود h نیز تابعی از ورودی‌ها می‌باشد $h(x_1, x_2, ..., x_n)$.

برای تشخیص خطاهای s-a-0 و s-a-1 بر روی سیم داخلی h در مدار شکل ۶-۵، از این روش کمک می‌گیریم.

نسخه پیش‌نویس



شکل ۱۱-۴ - مدار با خطای داخلی

$$F(X) = x_1x_2 + x_3x_4 + \overline{x_2x_4}$$
$$= h + \underbrace{(x_3x_4 + \overline{x_2x_4})}_{G}$$

$$\frac{dF}{dh} = \frac{d(G+h)}{dh} = \bar{h}\frac{dG}{dh} \oplus \bar{G}\frac{dh}{dh} \oplus \frac{dG}{dh}.\frac{dh}{dh}$$

$$= \overline{(x_3x_4 + \overline{x_2x_4})}\frac{dh}{dh} \oplus \bar{h}\frac{d(x_3x_4 + \overline{x_2x_4})}{dh} \oplus \frac{d(x_3x_4 + \overline{x_2x_4})}{dh}.\frac{dh}{dh}$$

$$= \overline{(x_3x_4 + \overline{\overline{x_2x_4}})} \oplus 0 \oplus 0$$

$$= \overline{x_3x_4} \cdot x_2x_4 = (\bar{x}_3 + \bar{x}_4)x_2x_4 = x_2\bar{x}_3x_4$$

بردار آزمون برای h با خطای s-a-0 برابر است با:

$$h.\frac{dF}{dh} = x_1x_2.x_2\bar{x}_3x_4 = x_1x_2\bar{x}_3x_4$$

بردار آزمون برای h با خطای s-a-1 برابر است با:

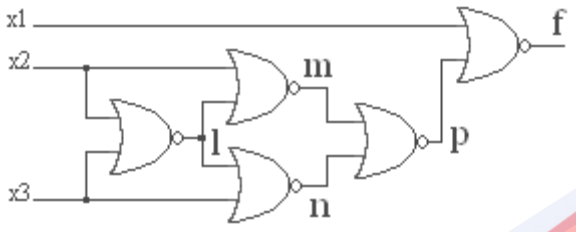
$$\bar{h}.\frac{dF}{dh} = \overline{x_1x_2}.x_2\bar{x}_3x_4 = \bar{x}_1x_2\bar{x}_3x_4$$

در حالت کلی این روش برای بدست آوردن بردار آزمون گره‌های داخلی یک شبکه منطقی مناسب نیست. مثال زیر این مطلب را نشان می‌دهد.

نسخه پیش‌نویس

مثال13) در مدار شکل زیر می‌خواهیم بردار آزمون تمامی ورودی‌ها را بدست آوریم. تابع شبکه عبارت است از:

$$F(X) = \overline{x_1}\,x_2\,\overline{x_3} + \overline{x_1}\,\overline{x_2}\,x_3$$



شکل ۱۱-۴

اختلاف‌های بولی تابع F بر حسب x_1، x_2 و x_3 عبارتند از:

$$\frac{dF}{dx_1} = x_2\,\overline{x_3} + \overline{x_2}\,x_3$$

$$\frac{dF}{dx_2} = \overline{x_1}\,\overline{x_3} + \overline{x_1}\,x_3$$

$$\frac{dF}{dx_3} = \overline{x_1}\,x_2 + \overline{x_1}\,\overline{x_2}$$

نتایج حاصل از اختلاف‌های بولی در جدول زیر آمده است:

جدول ۱۱-۲

	term 1	term 2
$x1_{s-a-0}$	110	101
$x1_{s-a-1}$	010	001
$x2_{s-a-0}$	010	011
$x2_{s-a-1}$	000	001
$x3_{s-a-0}$	011	001
$x3_{s-a-1}$	010	000

نسخه پیش‌نویس Draft

با کمی دقت متوجه می‌شویم که نتایج بالا را می‌توان با یک مشتق‌گیری جبری ساده نیز بدست آورد. این مسأله همواره برقرار نیست. بلکه تنها زمانی قادر به انجام این کار هستیم که در تمامی ترم‌های تابع، تمامی ورودی‌ها نقش داشته باشند.

با انتخاب بردارهای (۱۱۰،۰۱۰،۰۰۱) قادر خواهیم بود تا همه خطاهای ممکن در ورودی‌ها را آزمون کنیم چرا که این سه بردار تمامی خطاها را پوشش می‌دهند. به این ترتیب بجای اعمال ۱۲ بردار آزمون، اعمال تنها ۳ بردار آزمون برای یافتن تمامی خطاها کافی است. با این وجود این مجموعه بردار، برای آزمون خطا در گره‌های داخلی کافی نیست. برای گسترش مجموعه بردار آزمون، به‌طوری‌که تمامی خطوط ورودی و داخلی را بتوانیم آزمون کنیم، لازم است تا همه مسیرهای موجود میان ورودی‌های اصلی و خروجی‌های اصلی مورد بررسی قرار گیرند. این امر به کمک روش اختلاف بولی جزئی[1] انجام‌پذیر می‌باشد.

یک عبارت اختلاف بولی کلی را می‌توان از کنار هم قرار دادن عبارت‌های اختلاف بولی جزئی‌تر بدست آورد. برای مثال اگر $z = f(y)$ و $y = f(x)$ باشند، خواهیم داشت:

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

که در آن dz/dx یک اختلاف بولی جزئی نسبت به X است. با استفاده از اختلاف بولی جزئی در مسیر $x_2 - l - n - p - f$ در مثال قبل داریم:

$$\frac{df}{dx_2} = \frac{df}{dp}.\frac{dp}{dn}.\frac{dn}{dl}.\frac{dl}{dx_2}$$

همچنین:

$$\frac{df}{dp} = \frac{d(\overline{x_1}\,\overline{p})}{dp} = \overline{x_1}$$
$$\frac{dp}{dn} = \frac{d(\overline{m}\,\overline{n})}{dn} = \overline{m} = x_2 + l = x_2 + \overline{x_2} + \overline{x_3} = \overline{x_3}$$
$$\frac{dn}{dl} = \frac{d(\overline{l}\,\overline{x_3})}{dl} = \overline{x_3}$$
$$\frac{dl}{dx_2} = \frac{d(\overline{x_2}\,\overline{x_3})}{dx_2} = \overline{x_3}$$

بنابراین داریم:

$$\frac{df}{dx_2} = \overline{x_1}\,\overline{x_3}$$

[1] Partial Boolean Difference

نسخه پیش‌نویس

بنابراین بردارهای بدست آمده برای آزمون مسیر $x_2 - l - n - p - f$ عبارتند از: (۰۱۰ و ۰۰۰)

با اجرای همین روش برای مسیر $x_3 - n - p - f$ عبارت زیر بدست می‌آید.

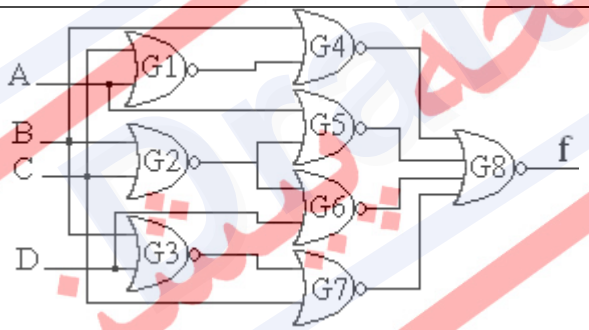
$$\frac{df}{dx_3} = \frac{df}{dp}.\frac{dp}{dn}.\frac{dn}{dx_3} = \overline{x_1}.\overline{m}\overline{l} = \overline{x_1}x_2$$

بردارهای بدست آمده از این مسیر عبارتند از: (۰۱۰ و ۰۱۱)

قضیه ۱) اگر $F = f(g_1, g_2, ..., g_m)$ باشد و برای هر $1 \le i \le m$ داشته باشیم $g_i = g(x_1, x_2, ..., x_n)$، اختلاف بولی جزئی به صورت زیر بسط خواهد یافت:

$$\frac{df}{dx_i} = \frac{df}{dg_1}.\frac{dg_1}{dx_i} \oplus \frac{df}{dg_2}.\frac{dg_2}{dx_i} \oplus ...$$
$$\oplus \frac{d^2 f}{dg_1 dg_2}.\frac{dg_1}{dx_i}.\frac{dg_2}{dx_i} \oplus ...$$
$$\oplus \frac{d^m f}{dg_1 dg_2 ... dg_m}.\frac{dg_1}{dx_i}.\frac{dg_2}{dx_i}...\frac{dg_m}{dx_i}$$

مثال۱۴) با استفاده از قضیه ۱، بردارهای آزمون را برای شکل زیر بدست آورید.



شکل ۱۱-۴

$$f = \overline{G_4 + G_5 + G_6 + G_7}$$

$$G_5 = \overline{A + G_2} = \overline{A}\,\overline{G_2}$$

$$G_6 = \overline{D + G_2} = \overline{D}.\overline{G_2}$$

$$\frac{df}{dG_2} = \frac{df}{dG_5}.\frac{dG_5}{dG_2} \oplus \frac{df}{dG_6}.\frac{dG_6}{dG_2} \oplus \frac{d^2 f}{dG_5 dG_6}.\frac{dG_5}{dG_2}\frac{dG_6}{dG_2}$$

$$\frac{dG_5}{dG_2} = \overline{A} \oplus 0 = \overline{A}$$

$$\frac{dG_6}{dG_2} = \overline{D}$$

$$\frac{df}{dG_5} = \overline{G_4}.\overline{G_6}.\overline{G_7} = BCD + \overline{A}\,\overline{B}.\overline{C}.\overline{D}$$

نسخه پیش‌نویس

$\frac{df}{dG_6} = ABC + \overline{A}\overline{B}.\overline{C}.\overline{D}$
$\frac{d^2 f}{dG_5 dG_6} = \frac{d}{dG_5}\frac{df}{dG_6} = \frac{d(\overline{G_4}.\overline{G_5}.G_7)}{dG_5} = BC + \overline{A}\overline{B}.\overline{C}.\overline{D}$
$G_2 \frac{df}{dG_2} = (\overline{B+C})(\overline{A}\overline{B}.\overline{C}.\overline{D} + \overline{A}BC + BC\overline{D})$

با وجود آن‌که روش اختلاف بولی وجود خطا در یک مدار را بدون اشتباه اعلام می‌کند، نیازمند صرف زمان و حافظه زیادی است.

۱۱-۷-۷ خط نرمال

برای هر خط داخلی در یک مدار، می‌توانیم عبارت $P_a = a_n x + a_1$ را تعریف کنیم. اگر یک خط نرمال باشد و خطایی بر روی آن رخ نداده باشد، $a_1 = 0$ و $a_n = 1$ بوده و $P_a = x$ خواهد شد. در غیر این‌صورت، با رخ دادن خطا $a_1 = 1$ و $a_n = 0$ می‌گردد و $P_a = 1$ خواهد شد. اگر خطای رخ داده s-a-0 باشد، خط نرمال به صورت $\overline{P_a} = a_n \overline{x} + a_1$ درخواهد آمد. در این حالت $P_a = 0$ خواهد بود.

مثال۱) در مدار شکل زیر، برای دو ورودی a و b، دو خط نرمال X_1 و X_2 را به صورت زیر تعریف می‌کنیم:

$P_a = a_n x_1 + a_1$
$P_b = b_n x_2 + b_1$

خط نرمال خروجی برابر با حاصلضرب دو عبارت بالا خواهد بود.

$P_c = P_a.P_b = (a_n x_1 + a_1)(b_n x_2 + b_1)c_n + c_1$

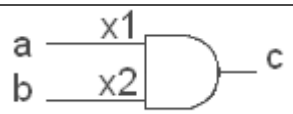
اگر هر دو ورودی سالم باشند،

$P_c = (x_1 + 0)(x_2 + 0)1 + 0 = x_1 x_2$

حال اگر a خراب و b سالم باشد:

$P_c = (0+1)(x_2)1 = x_2$

بردار آزمون از طریق عبارت $T = P_{cNormal} \oplus P_{cFaulty} = x_1 x_2 \oplus x_2 = \overline{x_1} x_2$ بدست می‌آید.



شکل ۱۱-۵ خط نرمال

۱۱-۸ آزمون مدارهای ترتیبی همگام

در مدارهای ترتیبی، الگوی آزمون به دنبالهٔ آزمون[1] تبدیل می‌شود که در واقع به آن دنباله‌ای از الگوهای آزمون نیز می‌گویند. می‌دانیم که معادلات توصیف‌کنندهٔ مدارهای منطقی ترتیبی عبارتند از:

$$Z_i = g_i(x_1, \cdots, x_n, y_1, \cdots, y_n) \quad i = 1, \cdots, m$$

$$Y_i = h_j(x_1, \cdots, x_n, y_1, \cdots, y_n) \quad j = 1, \cdots, r$$

$$y_j^{k+1} = y_j^k$$

در واقع چون مدارهای ترتیبی تابع ورودی و حالت هستند، آزمون آنها باید در یک حالت مشخص و ورودی مشخص انجام گیرد. بنابراین ما نیاز به دنباله‌هایی داریم که بتوانند مدار را به حالت آزمون ببرند. برای اینکار در ادامه به انواع مختلف این دنباله‌ها و نحوهٔ پیدا کردن هرکدام اشاره می‌نماییم.

۱۱-۸-۱ دنبالهٔ اولیه

دنبالهٔ اولیه[2] برای شروع کار مدار است که وقتی اعمال می‌شود، مدار به یک حالت معلوم می‌رود. بجای دنبالهٔ اولیه، می‌توان یک مدار Reset خاص برای مدار در نظر گرفت. ولی این مدار نیز در معرض عیب است و در بسیاری از موارد برای تمام حالت‌های ماشین عملی نیست. پس به این دلیل قبل از معرفی روش‌های ایجاد دنبالهٔ اولیه، باید مطالب زیر را بدانیم. دنبالهٔ آزمون را از نظر تولید دنبالهٔ آغازین بر دو حالت زیر نیز تقسیم بندی می‌کنند:

[1] Test Sequence
[2] Initial Sequence

نسخه پیش‌نویس

۱۱-۸-۲ دنبالهٔ انتقال

دنبالهٔ انتقال[1]، یک دنباله از ورودی است که با کمترین انتقالات مدار را از حالت S_i به حالت S_j می‌برد. برای پیدا کردن این دنباله از درخت انتقال[2] استفاده می‌شود که ریشهٔ آن حالت شروع است. مثال زیر، چگونگی یافتن دنبالهٔ انتقال را نشان می‌دهد.

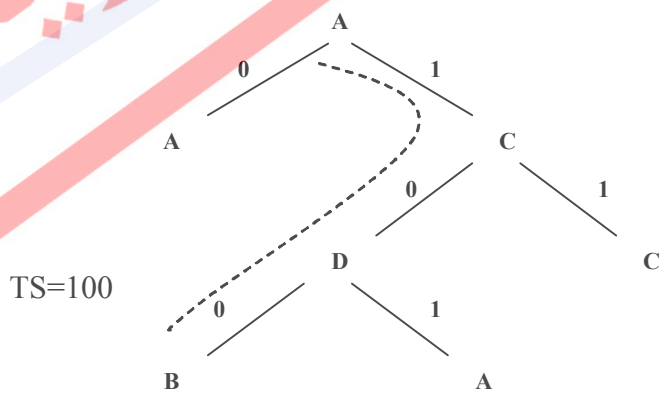
مثـال16)

می‌خواهیم کوتاهترین دنباله‌ای را که مدار ترتیبی با جدول حالت زیر از حالت A به B می‌برد، تعیین کنیم (TS: A → B).

جدول ۱۱-۳- جدول حالت مربوط به مثال بخش ۱۱-۷-۲

حالت اولیه	ورودی X: 0	ورودی X: 1
A	A, 0	C, 1
B	B, 1	C, 0
C	D, 1	C, 0
D	B, 0	A, 1

همانطور که در شکل زیر ملاحظه می‌نمایید، این درخت، یک درخت دودویی است که از حالت A شروع می‌شود و تمام حالت‌های ممکن را در بر می‌گیرد. در اولین نقطه‌ای که به حالت B برسیم کار پایان می‌یابد. همانطور که ملاحظه می‌نمایید: TS = 100



[1] Transfer Sequence
[2] Transfer Tree

نسخه پیش‌نویس Draft

۱۱-۸-۳ دنبالهٔ خانگی

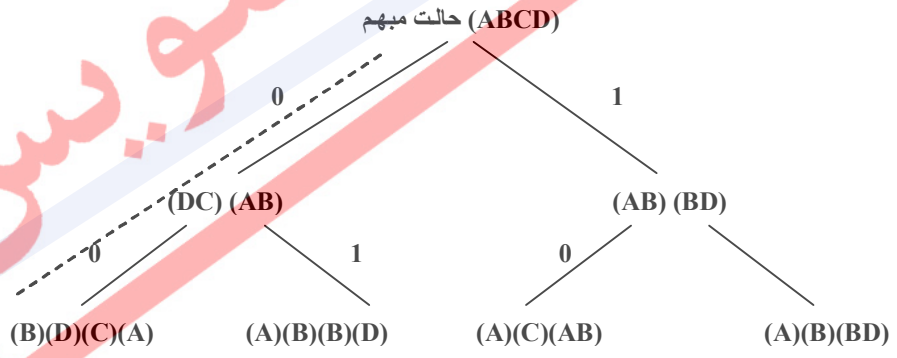
دنبالهٔ خانگی[1]، یک دنباله از ورودی است که به هر حالت اولیه‌ای که داده شود، حالت نهایی، منحصر به فرد خواهد بود. برای پیدا کردن این دنباله نیز نیاز به یک درخت خانگی داریم که ریشهٔ آن تمام حالت‌های مدار می‌باشد و براساس ورودی‌های ممکن برگ‌های آن تشکیل می‌شود. هر برگ نیز برحسب خروجی دسته‌بندی خواهد شد. شرایط پایان کار به صورت زیر خواهد بود:

- در یک برگ همهٔ حالت‌ها از هم جدا باشند و غیر تکراری باشند.
- اگر به یک حالت تکراری برسیم، شاخهٔ مورد نظر را دیگر ادامه نمی‌دهیم.

مثال ۱۷)

جدول ۱۱-۴ – جدول حالت مربوط به مثال بخش ۱۱-۷-۳

حالت اولیه	ورودی X: 0	ورودی X: 1
A	C, 0	A, 1
B	A, 1	B, 0
C	D, 0	B, 1
D	B, 1	D, 0



بنابراین در اینجا HS = 00 می‌باشد، زیرا شامل ورودی‌ای است که اگر به حالت مبهم اولیه داده شود، به برگی می‌رسیم که همهٔ حالت‌های آن از یکدیگر جدا هستند. جدول زیر را برای این مثال بدست

[1] Homing Sequence

نسخه پیش‌نویس Draft

می‌آوریم: (در جدول زیر، جدول تست، فرض بر این است که ورودی 00 به مدار داده شده است و خروجی و حالات آن را مورد بررسی قرار می‌دهیم)

جدول ۱۱-۵ – جدول تست مربوط به مثال بخش ۱۱-۷-۳

حالت اولیه	خروجی	حالت نهایی
A	00	D
B	10	C
C	01	B
D	11	A

۱۱-۸-۴ دنبالهٔ تشخیص

دنبالهٔ تشخیص[1]، یک دنبالهٔ ورودی است که باعث تولید دنبالهٔ خروجی منحصر بفردی می‌شود که از روی خروجی آن می‌توان حالت اولیه را بدست آورد. برای پیدا کردن این دنباله نیز نیاز به یک درخت تشخیص داریم که ریشهٔ آن تمام حالت‌های مدار می‌باشد و براساس ورودی‌های ممکن، برگ‌های آن تشکیل می‌شود. هر برگ نیز برحسب خروجی دسته‌بندی خواهد شد. شرایط پایان کار به صورت زیر خواهد بود:

- رسیدن به وضعیتی که همهٔ حالت‌ها از هم جدا باشند.
- اگر به یک حالت تکراری برسیم، شاخهٔ مورد نظر را دیگر ادامه نمی‌دهیم.
- در یک حالت، تکرار داشته باشیم مثلاً (AA)

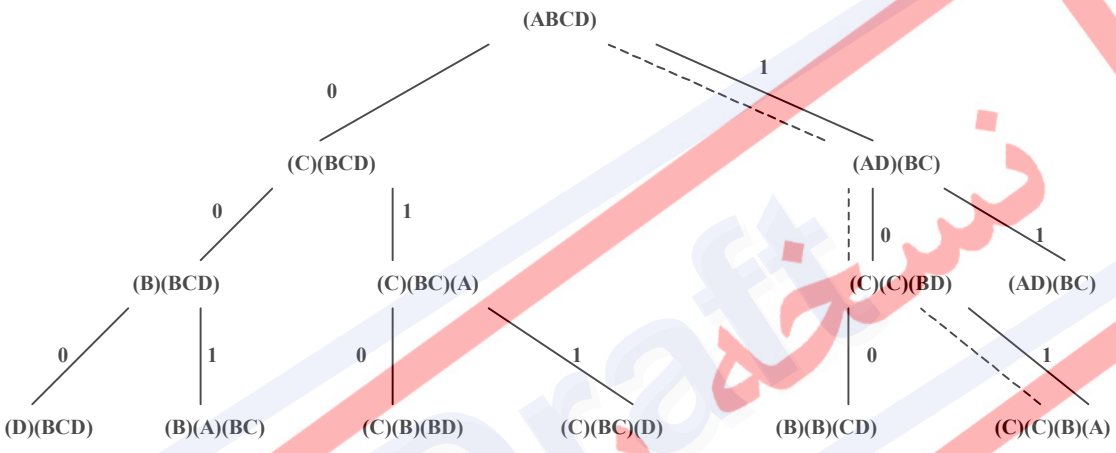
[1] Distinguish Sequence

نسخه پیش‌نویس

مثال 18)

جدول ۱۱-۶ – جدول حالت مربوط به مثال ۱۸ بخش ۱۱-۷-۴

حالت اولیه	ورودی X: 0	ورودی X: 1
A	C, 1	D, 0
B	D, 0	B, 1
C	B, 0	C, 1
D	C, 0	A,0



بنابراین در اینجا DS=101 می‌باشد و در جدول زیر مشاهده می‌نمایید که خروجی‌ها با اعمال این دنباله، منحصر بفرد بوده و می‌توان به کمک آنها حالت اولیه را تشخیص داد.

جدول ۱۱-۷ – جدول تست مربوط به مثال ۱۸ بخش ۱۱-۷-۴

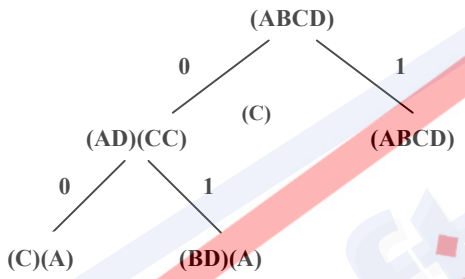
حالت اولیه	خروجی	حالت نهایی
A	001	C
B	100	A
C	101	B
D	011	C

نسخه پیش‌نویس Draft

مثـال۱۹)

جدول ۱۱-۸ – جدول حالت مربوط به مثال ۱۹ بخش ۱۱-۷-۴

حالت اولیه	ورودی X: 0	ورودی X: 1
A	C, 0	B, 0
B	D, 1	C, 0
C	A, 1	A, 0
D	C, 0	D, 0



همانطور که در درخت تشخیص ملاحظه می‌نمایید، این مدار دارای دنبالهٔ تشخیص نیست. در جدول زیر نیز این نکته نشان داده شده است.

جدول ۱۱-۹– جدول تست مربوط به مثال ۱۹ بخش ۱۱-۷-۴

حالت اولیه	خروجی	حالت نهایی
A	01	A
B	10	C
C	10	C
D	01	A

۱۱-۹ طراحی برای قابلیت آزمون پذیری[1]

روش‌های تولید دنبالهٔ آغازین (HS, DS, TS) برای مدارهای بی‌عیب قابل اعمال است. ممکن است در صورت اعمال این دنباله‌های آغازین به ماشین معیوب، نتیجهٔ مطلوب حاصل نشود. بنابراین دو سوال زیر پیش می‌آید:

- آیا در صورت نرسیدن به حالت آغازین مطلوب، این روش می‌تواند عیب را تشخیص دهد؟

[1] Design for Testability

نسخه پیش نویس Draft

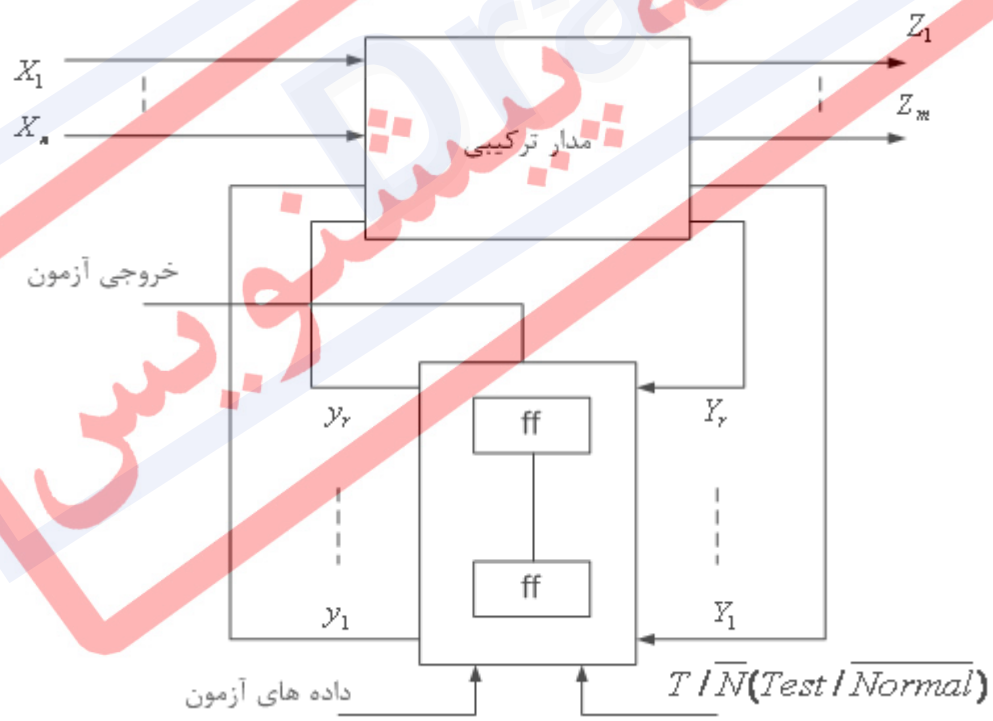
- آیا می‌توان دنبالهٔ آغازین مستقل از عیب طرح کرد؟

طراحی دنباله‌های آغازین و مشاهده برای مدارهایی که تعداد حالت‌های زیادی دارند، غالباً غیر عملی است. در چنین مواردی روش‌های طراحی خاصی برای آزمون‌پذیر کردن مدار بکار می‌رود، به نحوی که امکان بردن مدار به یک حالت مطلوب و مشاهدهٔ خروجی‌های آن ساده‌تر شود. بدین ترتیب، هزینهٔ طرح آزمون بسیار کم می‌شود. در این بخش به بررسی این نوع روش‌ها می‌پردازیم. دو پارامتر مهم در طراحی آزمون‌پذیر عبارتند از:

۱-توانایی مقداردهی هر کدام از سیگنال‌های مدار[1]

۲- توانایی مشاهده روی هر کدام از سیگنال‌های مدار[2]

ایدهٔ این روش از آنجا ناشی می‌شود که می‌خواهیم یک مدار ترتیبی را به صورت ترکیبی آزمون کنیم. در این روش فیلیپ-فلاپ‌ها در دو مد کار می‌کنند. یک مد معمولی و یک مد آزمون. در مد آزمون فیلیپ-فلاپ‌ها به شکل زنجیر و بصورت شیفت رجیستر عمل می‌کنند. حالت‌های ممکن را به وسیلهٔ ورودی‌ها به مدار می‌دهیم.

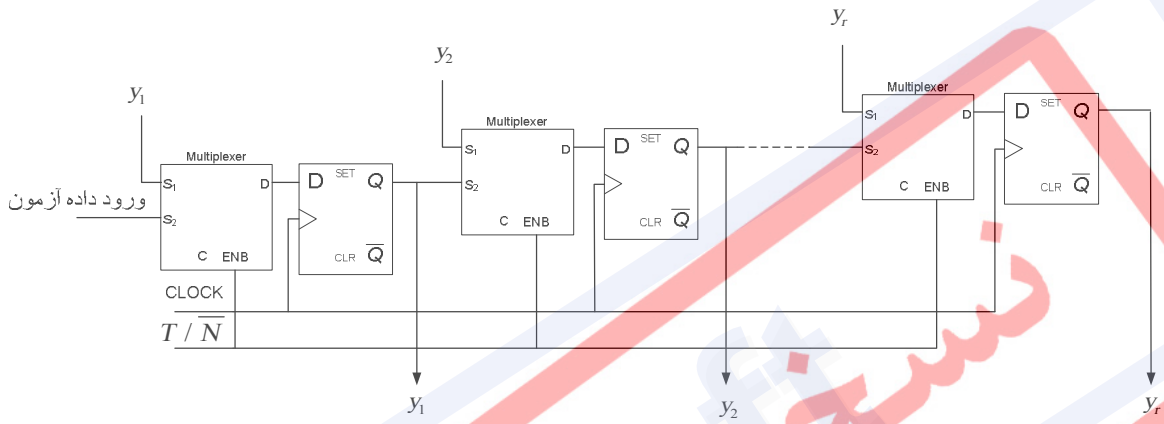


[1] Controlability
[2] Observeability

نسخه پیش‌نویس Draft

در ابتدا $T/\overline{N}=1$ قرار می‌دهیم. حالت‌های مورد نظر را به فیلیپ-فلاپ‌ها می‌دهیم و پس از آن $T/\overline{N}=0$ قرار می‌دهیم و همزمان ورودی‌های مورد نظرمان را نیز اعمال می‌کنیم. در انتها دوباره $T/\overline{N}=1$ قرار می‌دهیم و خروجی حاصل را می‌خوانیم.

در واقع ساختار فیلیپ-فلاپ‌ها به شکل ۲ خواهد بود.



شکل ۱۱-۶- ساختار فیلیپ-فلاپ‌ها در طراحی آزمون‌پذیر

بنابراین الگوریتم آزمون با قدم‌های زیر انجام می‌شود.

۱۱. $T/\overline{N}=1$ (مد آزمون)

۱۲. از ورودی مشخص شده، حالت مورد نظر را وارد فیلیپ-فلاپ‌ها می‌کنیم. برای این عمل به تعداد r پالس ساعت نیاز است.

۱۳. ورودی مورد نظر را اعمال می‌کنیم و $T/\overline{N}=0$ (مد معمولی)

۱۴. یک پالس منتظر می‌شویم.

۱۵. $T/\overline{N}=1$ (مد آزمون)

۱۶. از خروجی حالت نهایی را می‌خوانیم (r پالس ساعت نیاز است)

با افزایش پیچیدگی مدارهای انفرادی VLSI ونیز پیچیدگی کل سیستم، تولید و کاربرد آزمون، ابزاری گران و نه همواره مؤثر برای آزمایش آنها گردیده است. همچنین، در سرعت‌های بالا که بسیاری از سیستم‌های دیجیتالی در آن سرعت‌ها کار می‌کنند، مسایل بسیار دشواری وجود دارد. اینگونه مسایل نیاز به دستگاه‌های بسیار پیچیده‌ای دارند که همیشه هم نمی‌توان از عهدهٔ قیمت آنها برآمد. راه حل

نسخه پیش‌نویس Draft

پیشنهادی برای این مسائل، مجتمع‌سازی مؤثر یک سیستم آزمون خودکار در طراحی تراشه است. به این تکنیک BIST[1] گفته می‌شود. در نتیجه اهداف مدارهای که برای این کار پیشنهاد شده است، عبارتند از:

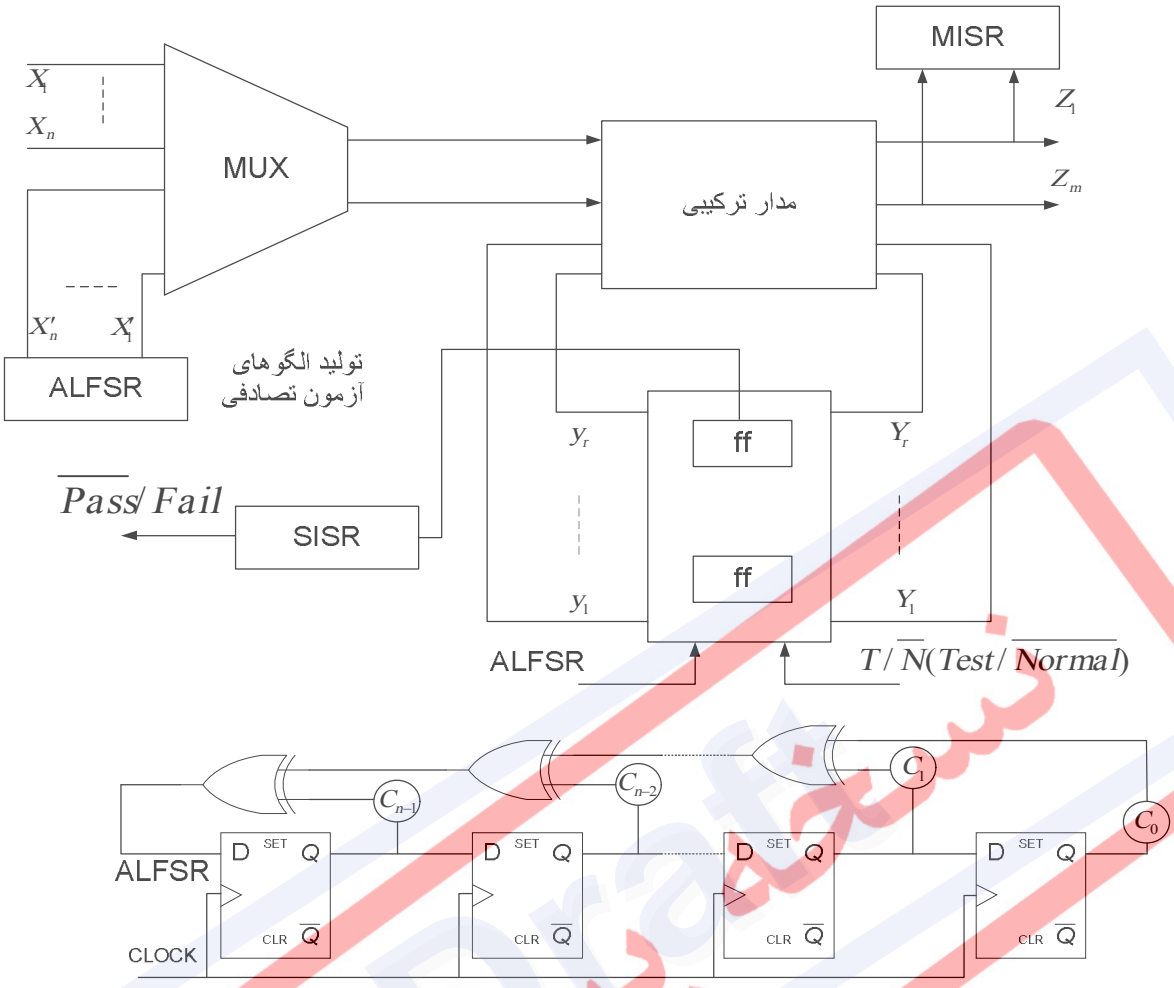
- کاهش هزینه‌های تولید الگوی آزمون
- کاهش حجم دادهٔ آزمون
- کاهش زمان آزمون

در شکل ۳ ساختار کلی این تکنیک را ملاحظه می‌فرمایید. در ورودی از ورودی‌های وابسته به انتخاب یا ورودی‌های عادی و یا از تولید کنندهٔ خودکار الگوهای آزمون استفاده می‌شود که با ساختار ALFSR[2] پیاده‌سازی می‌شود. ساختار این تولید کنندهٔ دنبالهٔ تصادفی در ادامه نشان داده شده است که ساختار بسیار ساده‌ای دارد. این ساختار یک معادلهٔ مشخصهٔ به نام $P_n(x)$ دارد که مشخص کنندهٔ واحدهای موجود در این ساختار است.

همچنین، مشاهده می‌کنید که یک ورودی نیز در قسمت عناصر حافظه در نظر گرفته شده است که با ALFSR تغذیه می‌شود و در زمانی که سیستم در مد آزمون قرار می‌گیرد، فیلیپ-فلاپ‌ها را مقدار دهی می‌کند. برای دیدن مقدار خروجی مدار و مقایسهٔ آن با مقدار درست و تشخیص خراب و یا سالم بودن آن از ساختارهای SISR[3] و MISR[4] استفاده می‌شود که در ادامه ساختارهای آنها بررسی می‌گردد.

[1] Built-In Self-Test
[2] Autonomous Linear Feedback Shift Register
[3] Single Input Shift Register
[4] Multiple Input Shift Register

نسخه پیش‌نویس

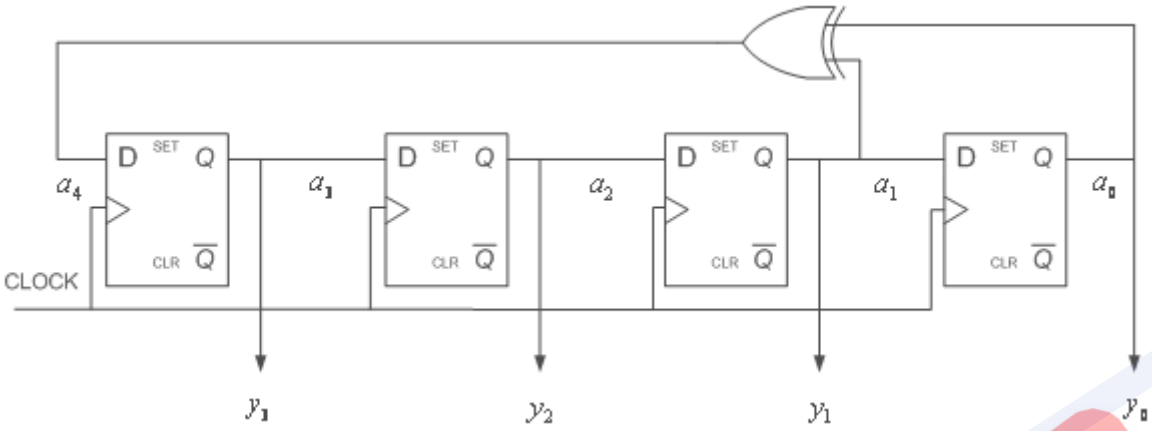


$$P_n(x) = x^n + c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_1 x + c_0 \quad , \quad c_i \in \{0,1\}$$

شکل ۱۱-۷ - ساختار کلی طراحی آزمون‌پذیر

برای فهم بهتر ساختار ALFSR به ذکر یک مثال بسنده می‌کنیم. فرض کنید معادلهٔ مشخصه برابر

$P(x) = x^4+x+1$ باشد. بنابراین ساختار به شکل زیر خواهد شد.



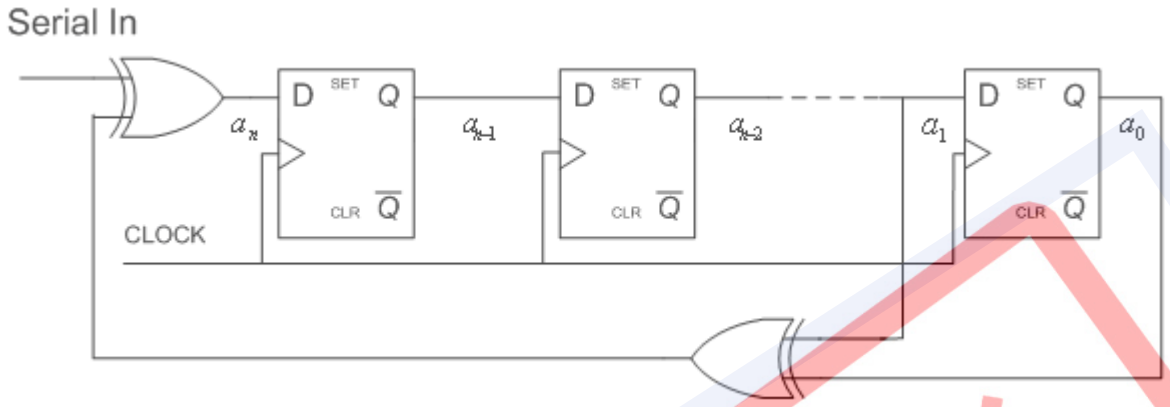
جدول زیر، مقادیر خروجی را در هر پالس ساعت نشان می‌دهد. همانطور که ملاحظه می‌کنید بعد از ۱۵ پالس دوباره دنبالهٔ خروجی تکرار می‌شود. با تغییر معادلهٔ مشخصه، دنباله و پریود تکرار خروجی تغییر خواهد کرد.

جدول ۱۱-۹ – مقادیر خروجی در هر پالس ساعت

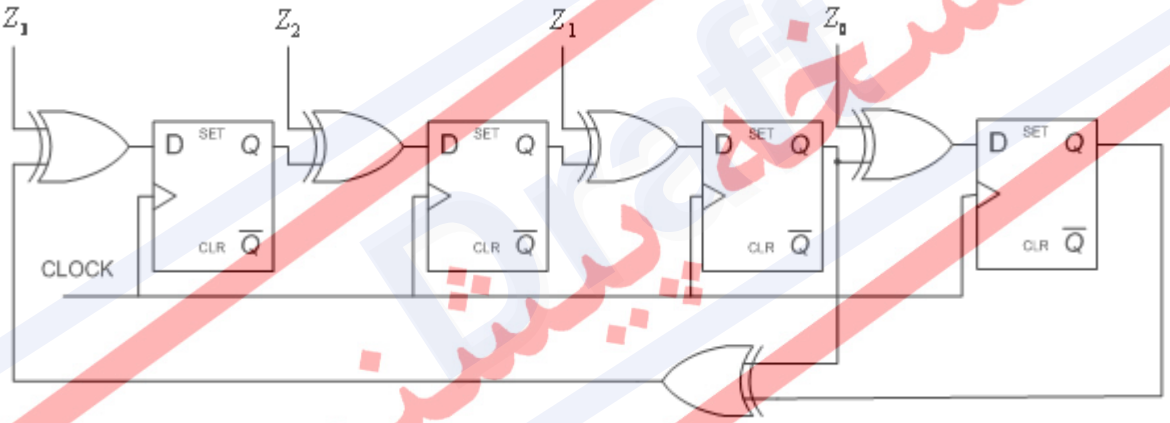
Clock	y_3	y_2	y_1	y_0
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	1	0	0	1
5	1	1	0	0
6	0	1	1	0
7	1	0	1	1
8	0	1	0	1
9	1	0	1	0
10	1	1	0	1
11	1	1	1	0
12	1	1	1	1
13	0	1	1	1
14	0	0	1	1
15	0	0	0	1
16	1	0	0	0

نسخه پیش‌نویس
Draft

ساختار های SISR و MISR نیز بسیار شبیه به ALFSR هستند. تنها تفاوت عمده وجود ورودی در ساختار اول و دوم است. زیرا در این ساختارها ما باید خروجی مدار را تحلیل کنیم. بنابراین ساختار یک ورودی و چند ورودی برای خروجی مدار و حالت مدار مورد نیاز است.



شکل ۱۱-۸- ساختار SISR



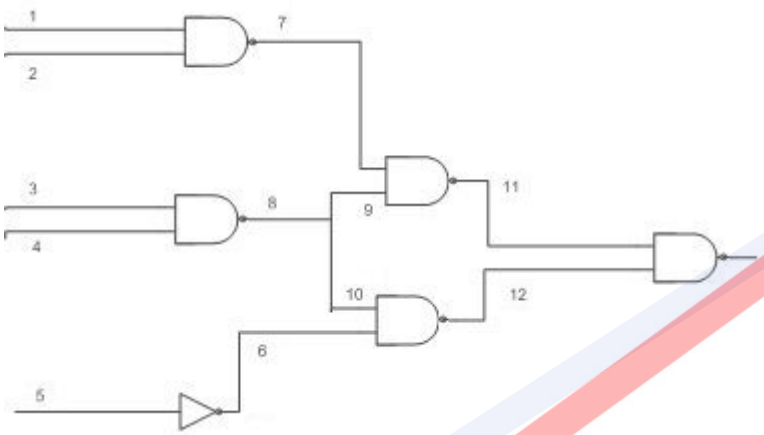
شکل ۱۱-۹ - ساختار MISR

نسخه پیش‌نویس

۱۱-۱۰ تمرین

۱- با استفاده از روش یای انحصاری تمامی تست های زیر را برای مدار داده شده به دست آورید:

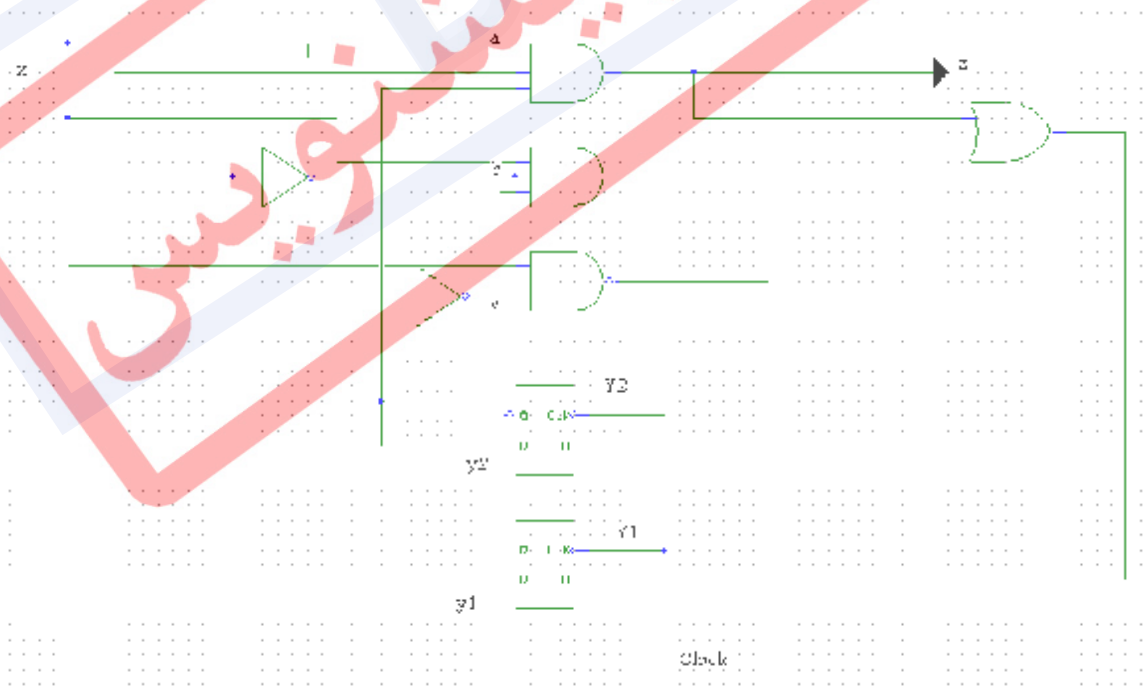
الف) 2/0 ب) 8/1 پ) 9/0 ت) 5/0



۲- تمرین قبل را با استفاده از روش ساختن مسیر تکرار کنید.

۳- با توجه به مدار داده شده در شکل زیر، یک دنباله اولیه و یک دنباله observation برای هر یک از خطا های زیر تعیین کنید:

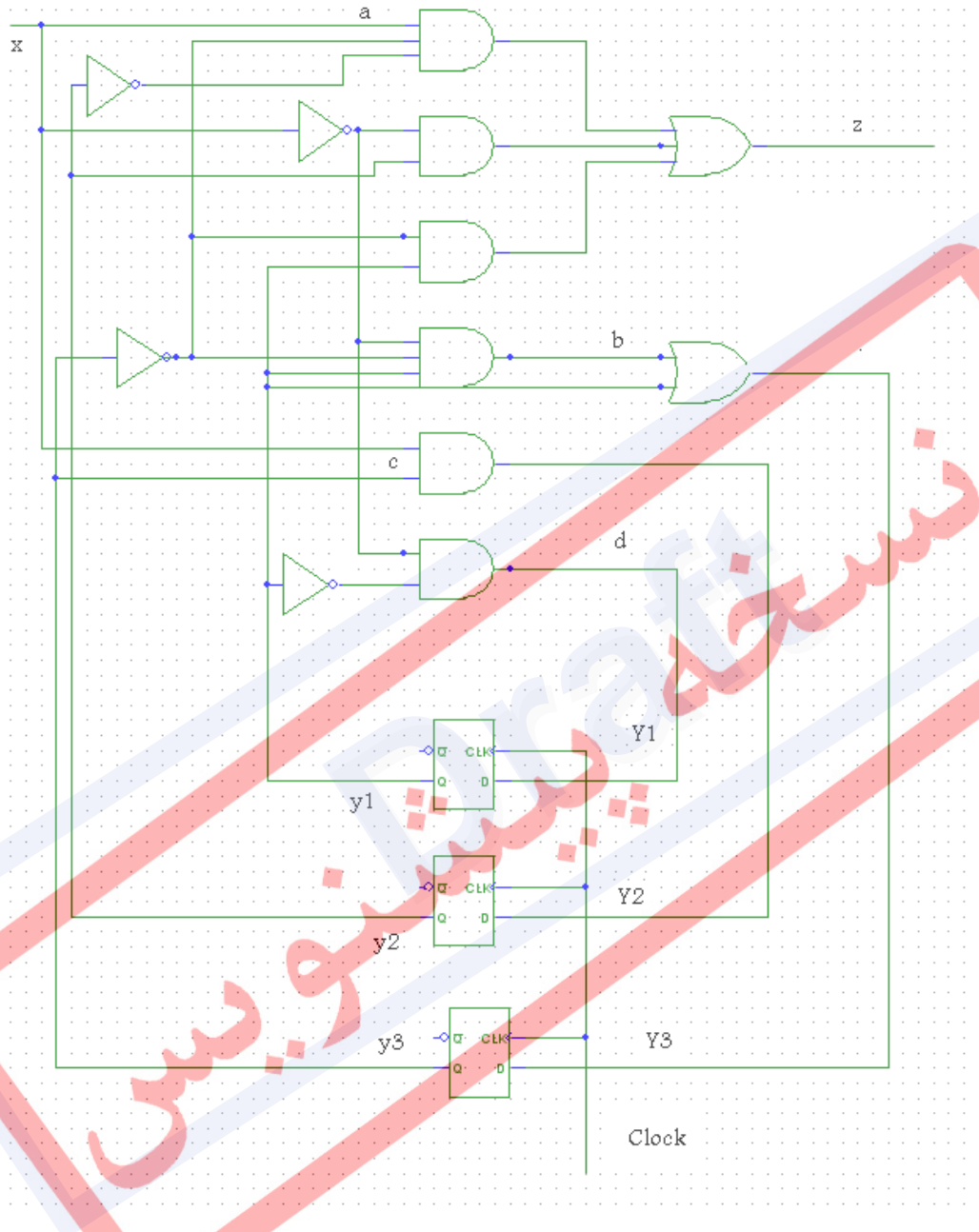
الف) a/1 ب) b/0 پ) c/1



نسخه پیش نویس
Draft

۴- تمرین قبل را برای مدار و خطاهای زیر تکرار کنید:

الف) a/1 ب) b/0 پ) c/0 ت) d/1



نسخه پیش نویس

۵- یک ثبات BILBO ۸ بیتی با جدول تابع زیر طراحی کنید:

B_1	B_2	Function
0	0	AFLSR mode
0	1	Scan mode(shift register)
1	0	Normal mode(parallel load)
1	1	MISR mode

$p(x) = 1 + x + x^5 + x^6 + x^8$

۶- فرم مینیمم FDTS جدول خطای زیر را به دست آورید.

Tests			Faults							
x_1	x_2	x_3	A	B	C	D	E	F	G	H
0	0	0	1							1
1	0	0		1				1		
0	1	0	1		1				1	
0	1	1				1		1		
1	0	0	1							1
1	0	1		1			1		1	
1	1	0						1		1
1	1	1	1			1			1	

نسخه پیش‌نویس Draft

۷-حالت شروع مداري كه جدول تست آن به صورت زير است برابر S1 است. دنباله ورودي ۰۱۱۰۱۰ به مدار داده مي شود و دنباله خروجي ۰۰۱۰۱۰ مشاهده مي شود. آيا مدار داراي خطاست؟ اگر اينچنين است، چه خطايي وجود دارد؟

	initial state	Observation sequence, x			Output sequence, z			Final state	Faults tested
۱	S1	0	1	1	0	D’	D’	S3	a0,d1,e1,h0
۲	S2	1	1	1	D’	1	D	S1	b1,e0
۳	S3	0	1	0	0	1	D’	S4	c0,f0
۴	S4	0	0	1	0	D’	D	S1	a1,c1,f1,g0,h1
۵	S1	1	0		0	D’		S2	b0,d0,g1

نسخه پیش‌نویس
Draft

۱۲ مراجع

1- D. A. HUFFMAN, **CANONICAL FORMS FOR INFORMATION-LOSSLESS FINITE-STATE LOGICAL MACHINES,** TECHNICAL REPORT 349**,** MIT RESEARCH LAB. OF ELECTRONICS, CAMBRIDGE, MASSACHUSETTS, March, 1959.

2- VICTOR. P. NELSON, H. T. NAGEL, B. D. CARROLL, J. D. IRWIN **DIGITAL LOGIC CIRCUIT ANALYSIS & DESIGN,** ENGLEWOOD CLIFFS, NEW JERCY: PRENTICE-HALL, 1995.

3- A. D. FREIDMAN, P. R. MENON, **FAULT DETECTION IN DIGITAL CIRCUITS**, ENGLEWOOD CLIFFS, NJ: PRENTICE-HALL, 1971.

Draft نسخه پیش‌نویس