اسلایدهای درس

مدارهای منطقی

دکتر پویا فوده

مهر ۰۱–۰۰

مدارهای منطقی

(طراحی سیستم‌های دیجیتال)

جلسه اول

آشنایی

مباحث امروز

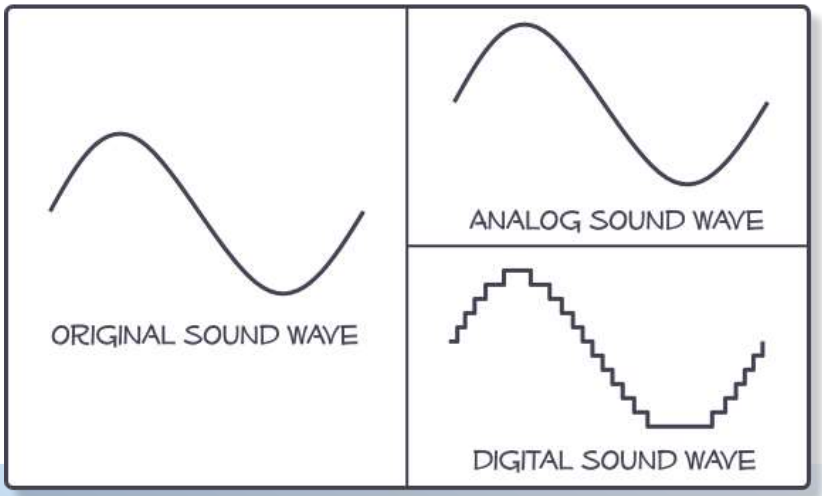
- طراحی دیجیتال یا مدارهای منطقی چیست.
- تاریخچه سیستم‌های دیجیتال (تعریف و تفاوت آنها با سیستم‌های غیر دیجیتال)
- کاربرد در دنیای امروز
- این درس چه فایده‌ای دارد.
- مباحث طول ترم چه خواهد بود.
- کتاب‌ها و منابع این درس کدامند.
- نمره نهایی چگونه محاسبه می‌شود.

دستگاه‌های دیجیتالی

- یک دستگاه دیجیتال، دستگاهی است که در آن داده‌های دیجیتال استفاده و پردازش و احتمالا ذخیره می‌شوند.
- داده‌های دیجیتال مقدارهایی هستند که به صورت گسسته (معمولا عددی صحیح یا اعشاری با تعداد اعشار معلوم) نمایش داده می‌شود. امروزه معمولا یک داده تکی در سیستم‌های دیجیتال الکترونیکی فعلی، دارای مقدار صفر یا یک است.
- پیوسته متضاد گسسته است. بین هر دو مقدار پیوسته مقدارهایی وجود دارد (درجه تنظیم یخچال یا درجه نمایشگر ترازو را برای هم آنالوگ و هم دیجیتال در نظر بگیرید).
- موضوع درس مدارهای منطقی طراحی و ساخت دستگاه‌های دیجیتال الکترونیکی (نه لزوما کامپیوتر) می‌باشد.

دستگاه‌های دیجیتالی و آنالوگ

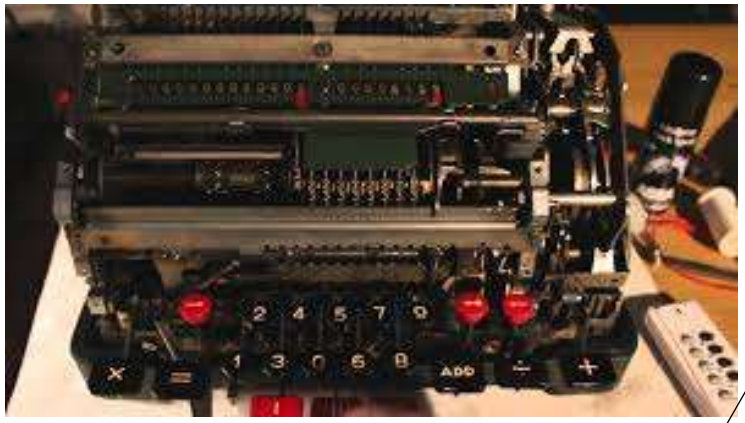
- قدیمی‌ترین سیستم دیجیتال کد ژنتیک موجودات زنده و قدیمی‌ترین سیستم دیجیتال برقی تلگراف است اما با عرضه کامپیوتر، سیستم‌های دیجیتال الکترونیکی کاربرد بسیار گسترده‌ای در زندگی انسان‌ها پیدا کردند.
- یک دستگاه آنالوگ، دستگاهی است که در آن اطلاعات بصورت امواج پیوسته در زمان پردازش و ذخیره می‌شوند. امواج آنالوگ لزوما الکتریکی نیستند و می‌توانند مکانیکی هم باشند.

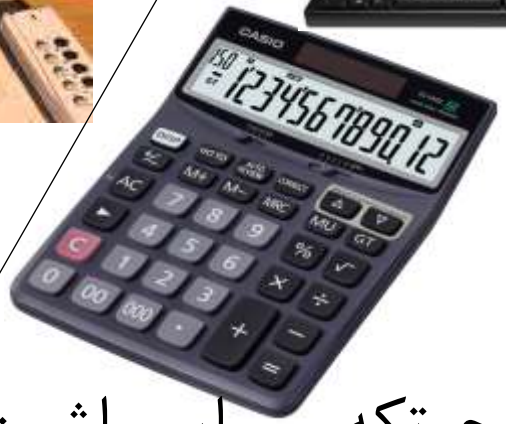


دستگاه‌های دیجیتالی و آنالوگ

- گرامافون کوکی (غیربرقی) یک دستگاه آنالوگ مکانیکی و قدیمی‌ترین دستگاه آنالوگ ساخت دست بشر است.
- در ادبیات کوچه بازاری کلمه آنالوگ را متضاد کلمه دیجیتال قرار داده‌اند و هر نمونه غیر دیجیتال هر دستگاهی که دیجیتالش موجود است را آنالوگ می‌خوانند مانند ساعت مچی و .... چنین استفاده‌ای از کلمه آنالوگ اشتباه است.
دستگاه‌های آنالوگ باید با موج آنالوگ کار کنند.

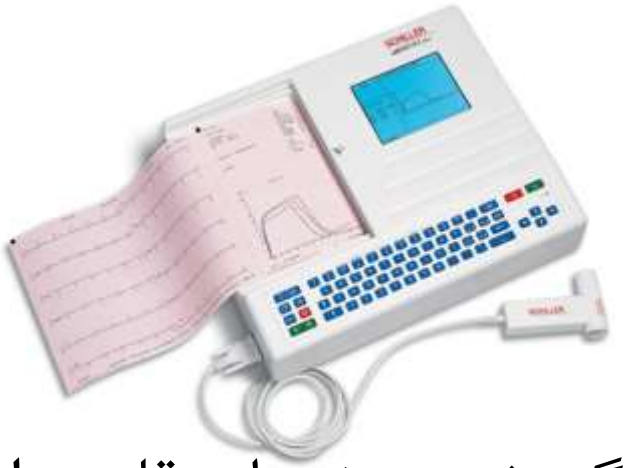
تاریخچه دستگاه های دیجیتال الکترونیکی





- چرتکه و سایر ماشین حساب‌های مکانیکی و ساعت مچی کوکی اطلاعات را به صورت عددی و گسسته پردازش می‌کنند هر چند مکانیکی اند نه الکترونیکی.
- همه سیستم‌های دیجیتال کامپیوتر نیستند. مثلا ساعت مچی و ماشین حساب سمت راست، دستگاهی‌ست که فقط دستگاه دیجیتالی است ولی کامپیوتر (قابل برنامه‌ریزی) نیست.

تاریخچه: دستگاه الکتروکاردیوگراف (نوار قلب)



- دستگاه آنالوگ شدت ضربان قلب را تبدیل به ولتاژ الکتریکی متغیر (موج) می کند. این ولتاژ سوزن چاپگر را حرکت می دهد. اما دستگاه دیجیتال شدت ضربان را اندازه گیری و تبدیل به عدد می‌کند.
- کارکرد هر دو دستگاه یکی است اما نمونه دیجیتال سبکتر و ارزان‌تر است.

تاریخچه: سیستم‌های ضبط و پخش صوت

- ورودی (میکروفون) و خروجی (بلندگو) لزوما آنالوگ اند اما سیستم داخلی دیجیتال (تبدیل موج به عدد) در CDو فلش موجب افزایش حیرت انگیز ظرفیت و کیفیت شده است.

تاریخچه: سنسور دمای اتاق

ترموستات آنالوگ هانیول مدل T6373B1064

۲۴۶,۴۰۰ تومان

ترموستات هانیول سری O3 مدل TF428

۴۳۱,۳۰۰ تومان

- ترموستات مکانیکی (به غلط آنالوگ) و دیجیتال الکترونیکی هر دو دما را تنظیم می‌کنند اما نوع دیجیتال برنامه‌پذیر است: مثلا اگر دما به ۲۴ بود روی فن ۱ و اگر ۲۶ روی فن ۳ روشن می‌شود. همچنین ۳ تا ۶ صبح خاموش.

تاریخچه: دوربین عکاسی

- دوربین شیمیایی (ثبت کننده امواج نوری) تصویر را روی یک فیلم حساس به نور ثبت می کند. دوربین دیجیتال رنگ های نقطه به نقطه تصویر را به صورت عددی ذخیره می‌کند.
- دوربین دیجیتال با کیفیت‌تر است اما مقیاس پذیری‌اش (پیکسل‌هایش) محدود است. تصاویر دیجیتال بدون چاپ قابل استفاده‌اند.

تاریخچه: ارتباطات راه دور

در انتقال صوت و تصویر به صورت دیجیتال الگوریتم‌های فشرده‌سازی و تکنیک‌های انتقال با سیم مسی، فیبر نوری و بی‌سیم ظرفیت انتقال بسیار افزایش داده شده است.

سیستم‌های دیجیتال انتقال داده را بسیار سریع وکیفیت و قیمت سرویس‌های تلفن و تلویزیون را به کلی متحول کرده‌اند.

فایده یادگیری مدارهای منطقی

- این درس در کنار دروس اساسی دیگر مهندسی کامپیوتر (معماری کامپیوتر و ریزپردازنده و ...) شما را از سطح تکنسین (برنامه نویس = تکنسین نرم‌افزار و تعمیرکار = تکنسین سخت‌افزار) بالا برده و به سطح مهندسینی می‌رساند که طرز کار کامپیوتر را عمیقا درک می‌کنند.

- شغل‌هایی که با طراحی سیستم‌های متوسط دیجیتالی را در فهرست موقعیت‌های شغلی شما قرار می‌دهد (کلمات استخدام FPGA را در گوگل جستجو کنید).

- در کنکور ورود به مقاطع بالاتر نقش کلیدی دارد.

مباحث درس

- مقدمات: آشنایی با سیستم‌های دیجیتال، معرفی روش‌های ذخیره داده و عدد. جبر بول، گیت‌های پایه و جدول درستی. معرفی اجمالی ICهای دیجیتال.
- مدارهای منطقی ترکیبی: فورمول نویسی، ساده‌سازی و طراحی مدارهای ترکیبی. تحلیل مدارها. جمع و تفریق‌کننده. دیکدر. مالتی‌پلکسر. مدارهای برنامه‌پذیر.
- مدارهای منطقی ترتیبی: لچ. پالس ساعت. فلیپ فلاپ. معادلات، نمودار و دیاگرام (ماشین) حالت. فرآیند طراحی مدارات سنکرون و تحلیل آنها. مدارهای مور و میلی. رجیستر و شیفت رجیستر و شمارنده.

کتاب و منابع درسی

- کتاب مدارهای منطقی موریس مانو منبع اصلی این درس است.
- ترجمه دکتر سپیدنام از این کتاب (بر خلاف بقیه ترجمه‌ها) صحیح، شیوا و روان است. آخرین چاپ کتاب ویراست پنجم است اما ویراست‌های قبلی همچنان قابل استفاده هستند.
- کتاب استفن براون ترجمه زارع‌پور نیز کتاب خوبی است. خصوصا شکل‌های این کتاب بسیار به فهم مطلب کمک می‌کند.

پیش ارائه جلسات درس

- مقدمه هر جلسه از درس اهمیت زیادی دارد. در این مقدمه گفته می‌شود که قرار است امروز چه مطالبی ارائه شود و این مطالب به چه کاری می‌آید.
- پیش ارائه به صورت فیلمی کوتاه (چند دقیقه‌ای) هر هفته در روز قبل از درس به ادمودو ارسال می‌گردد. توصیه می‌شود این فیلم را در زمانی که فرصت و تمرکز ذهنی دارید مشاهده کنید تا موضوع در ذهن‌تان جایگیر شود و برای درک درسی که فردا در کلاس ارائه می‌شود آمادگی خوبی پیدا نمایید.

نمره پایان ترم

- **ارزیابی مستمر در طول کلاس (۱۳.۵ نمره):**

✓ تمرین‌های هفتگی ۷ نمره ۵ نمره
توجه: نمره تمریناتی که اصالت نداشته باشند در پایان ترم حذف خواهد شد. فرقی بین دهنده و گیرنده تمرین نیست. تقلب در تمرین و امتحان در نهایت مرتکبین و کل کلاس را متضرر خواهد کرد. توضیح بیشتر در وب سایت موجود است.

✓ امتحان میان ترم ۶.۵ نمره

- **ارزیابی پایان ترم:**

✓ امتحان پایان ترم ۶.۵ نمره ۱۵ نمره

✓ نمره ارفاقی به همه بطور مساوی (مقدار آن آخر ترم معلوم می‌شود)

✓ *در صورتی که امتحان پایان ترم بصورت حضوری برگزار شود نمره قرمز رنگ.

- **نمرات اضافه بر سازمان**
- تمرین برجسته (هر تمرین ۰.۵ هر دانشجو حداکثر ۲ نمره در ترم)
- مشارکت برجسته در کلاس (هر بار ۰.۵ هر دانشجو حداکثر ۲ نمره در ترم)
- کوئیز (در ترم معمولا ۳ یا ۴ کوییز خواهد بود که کلا ۱ نمره دارد)

نمرات اضافه بر سازمان

- تمرین یا مشارکت برجسته معمولا هر جلسه به یک نفر داده می‌شود که بهترین تمرین را ارائه یا بهترین مشارکت را در کلاس آنلاین داشته باشد. البته در شرایطی ممکن است به دو نفر داده شود یا به کسی داده نشود.
- تمرین برجسته تمرینی است که به نکاتی که کسی به آن دست نیافته دست یابد و بخشی که همه نادرست یا ناقص حل کرده باشند را حل نماید.
- مشارکت کننده برجسته (ویژه جلسات آنلاین) در کلاس کسی است که سوالی که دیگران نتوانند را پاسخ گوید یا سوال خوبی ی مطرح کند که به فهم درس کمک نماید. استفاده از میکروفون در این مورد می‌تواند نقش زیادی داشته باشد.
- کوییز تک سوالی است که بدون هماهنگی قبلی از درس همان روز یا درس‌های قبل یا مباحث مربوط به درس مطرح می‌شود و در زمان محدود در ادمودو ارسال می‌گردد.

کوییزهای انفرادی و گروهی

- برخی کوییزها انفرادی هستند و برخی بصورت گروهی برگزار می‌شوند.
- قوانین: دانشجویان بطور تصادفی در گروه‌هایی چند نفره قرار خواهند گرفت. می‌توانید با کمک هم به حل مساله بپردازید و در نهایت یک برگه با ذکر شماره گروه و نام همه اعضا گروه تحویل داده خواهد شد و به همه نمره یکسان تعلق می‌گیرد.
- در موارد زیر نمره صفر به همه اعضای گروه تعلق خواهد گرفت:
  ۱- عدم ارسال پاسخ در ادمودو.
  ۲- ارسال بیش از یک پاسخنامه از یک گروه.
  ۳- اگر نام گروه و اعضای آن با آنچه هنگام تعیین گروه‌ها اعلام شده متفاوت باشد (کم یا زیاد باشد.)
- در کلاس‌های حضوری گروه‌بندی توسط من در کلاس انجام می‌شود و گروه‌ها دور هم نشسته و سوال را با مشورت هم حل می‌کنند. در کلاس آنلاین اعضای گروه در یک اتاق مستقل مجازی قرار گرفته ئ زمان جلسه همه امکان نوشتن در کادر چت و استفاده از میکروفون را دارند. کسی هر کس می‌تواند خودش را ارائه دهنده کند و روی تخته سیاه بنویسد یا آن را برای همه باز کند.

تمرین‌ها

- تمرین‌ها: هر هفته به صورت مرتب. زمان تحویل تا فقط هفته آینده. تمرین تاریخ گذشته ارزشی ندارد.
- برای تحویل تمرین‌ها، دریافت اسلایدها، امتحان میان‌ترم، و سوال در مورد درس از سیستم ادمودو edmodo استفاده خواهد شد.
- برای اطلاع از جزییات شرایط تحویل تمرین‌ها و دستورالعمل استفاده از ادمودو به سایت من مراجعه شود: whiteboard.ir
  (هم از طریق وب سایت می‌شود و هم با نصب اپ اندروید یا IOS.)
- کد ادمودو (کلاس منطقی سه شنبه): fnny7s

نکته ۱: نام خود را فارسی وارد کنید. نه پنگلیش.
نکته ۲: اسلایدها و تمرین‌ها احتمال زیاد دارد که با ترم پیش متفاوت باشد. همیشه اسلایدهای جدید را بصورت هفتگی دریافت کنید.
نکته۳: تمرینات را PV و ایمیل نکنید. Zip و Rar نفرستید. راهنما را بخوانید.

مدارهای منطقی

(طراحی سیستم‌های دیجیتال)

جلسه دوم

نمایش داده‌ها در
سیستم‌های دیجیتال

بحث امروز

- سیستم‌های دیجیتال و منطق دودویی
- اعداد دهدهی، دودویی (باینری) و شانزده‌شانزدهی (هگزادسیمال)
- اعداد منفی
- کدهای BCD و گِری
- کد برای حروف الفبا

داده در سیستم‌های گوناگون

- در سیستم‌های آنالوگ داده به صورت پیوسته و قیاسی ظاهر می‌شود. مثل باری که روی ترازو قرار گرفته و شاهین ترازو را به نسبت سنگینی خود حرکت می‌دهد.
- اما در سیستم‌های دیجیتال داده‌ها بصورت گسسته و عددی ظاهر می‌شوند.

عدد چیست؟ مبنای ده

- متداول‌ترین و ساده‌ترین روش نوشتن اعداد، نوشتن آنها در سیستم مبنایی است که در آن هر رقم بسته به اینکه چندمین رقم از سمت راست است وزنی به صورت توانی پیدا می‌کند.
- همانطور که می‌دانید اعداد طبیعی را می‌توان در هر مبنایی (دو و بالاتر) نشان داد. تنها محدودیت این است که بشر برای مبناهای بالاتر از ۱۰ رقم‌های لازم را طراحی نکرده است.
- بدون شک مبنای ۱۰ (ساخته شده به علت وجود ۱۰ انگشت) برای ارتباط با انسان‌ها بهترین و در واقع تنها مبنای ممکن است. چرا که همه ما از کودکی به آن خو کرده‌ایم و فقط قادر به فهم اعداد این سیستم هستیم.

مبنای ده

- در همه مبناها از جمله مبنای ۱۰ بارزشترین رقم سمت چپ و بی ارزشترین رقم سمت راست قرار دارد.

1367

بی ارزشترین رقم با ارزشترین رقم

مبنای ده

- در همه مبناها از جمله مبنای ۱۰ بارزشترین رقم سمت چپ و بی ارزشترین رقم سمت راست قرار دارد.

- در همه مبناها از جمله مبنای ۱۰، ضرب کردن هر رقم در توان‌هایی از مبنا مقدار آن عدد را می‌دهد:

$$1367 = 1 * 10^3 + 3 * 10^2 + 6 * 10^1 + 7 * 10^0$$

چرا مبنای دو

- در سیستم‌های آنالوگ (قیاسی) داده مستقیما تبدیل به یک موج پیوسته شبه سینوسی می‌شود. در سیستم‌های آنالوگ تغییرات ناگهانی با طبیعت سیستم سازگاری ندارد.
- پیاده‌سازی مبناهای مختلف در سیستم‌های الکترونیکی دیجیتال امکان‌پذیر هست، اما به صرفه و کارا نیست. مثلا می‌توان مبنای ۱۰ را با ۱۰ ولتاژ گوناگون بین ۰ تا ۹ ولت شبیه سازی کرد. با این کار (به دلیل غیر ایده‌آل بودن و خطا پیدا کردن سیگنال همانطور که در شکل می‌بینید):

۱. هزینه ساخت مدار افزایش می‌یابد
۲. خطا افزایش می‌یابد
۳. سرعت کاهش می‌یابد

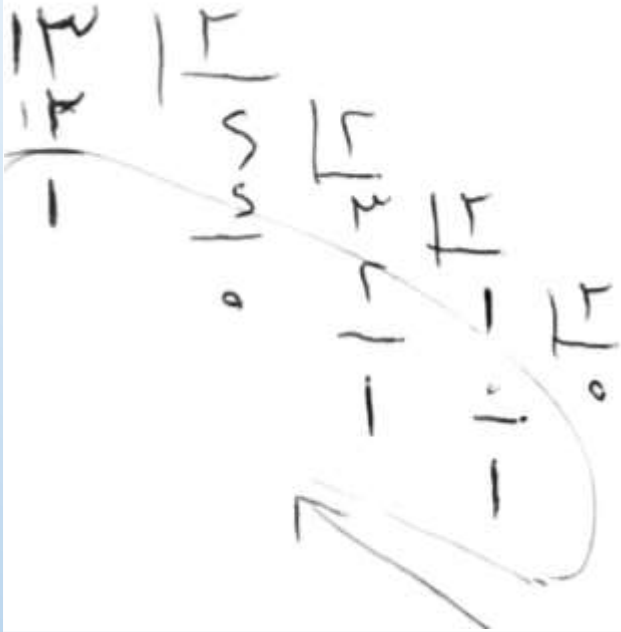
مبنای دو

- تنها مبنای عملی برای سیستم‌های دیجیتال و کامپیوتر مبنای ۲ است (فقط دو پله 0 و 1، سیگنال چند پله‌ای فقط زمان دیجیتایز کردن سیگنال‌های آنالوگ به کار می‌رود)، در حالی که مبنای دو با اعداد بسیار طولانی و برای ما انسان‌ها که به مبنای ۱۰ عادت داریم قابل درک نیست.
- برای تبدیل دستی مبنای ۱۰ به مبنای ۲ از روش تقسیمات متوالی استفاده می‌شود.

مثال: عدد 13

مبنای دو

- تنها مبنای عملی برای سیستم‌های دیجیتال و کامپیوتر مبنای ۲ است (فقط دو پله 0 و 1، سیگنال چند پله‌ای فقط زمان دیجیتایز کردن سیگنال‌های آنالوگ به کار می‌رود)، در حالی که مبنای دو با اعداد بسیار طولانی و برای ما انسان‌ها که به مبنای ۱۰ عادت داریم قابل درک نیست.
- برای تبدیل دستی مبنای ۱۰ به مبنای ۲ از روش تقسیمات متوالی استفاده می‌شود.



مثال: عدد 13 در مبنای ده
برابراست با 1101 در مبنای دو

(آنقدر عدد را بر دو تقسیم می‌کنیم تا تمام شود یعنی خارج قسمت صفر گردد. باقیمانده‌ها، پایین به بالا پر ارزش به کم ارزش خواهند بود)

مبنای دو

- برای تبدیل مبنای ۲ به ۱۰ هم از همان روش ضرب توانی استفاده می‌شود.

مثال: عدد 1101

مبنای دو

- برای تبدیل مبنای ۲ به ۱۰ هم از همان روش ضرب توانی استفاده می‌شود.

مثال: عدد 1101 در مبنای دو برابراست با ۱۳در مبنای ده

$$1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 =$$
$$8 + 4 + 0 + 1 = 13$$

مبنای شانزده

- برای طراحان سیستم (انسان‌ها) مبنای 10 راحت است و برای خود سیستم (کامپیوتر) مبنای 2. تبدیل این دو مبنا اگرچه امکان‌پذیر است اما زمان‌بر و اشتباه‌خیر است.
- راه حل این مشکل مبنای 16 یا هگزادسیمال است. این مبنا از یک طرف بسیار آسان به مبنای 2 (باینری) قابل تبدیل است، از طرف دیگر فهم و ثبت آن برای انسان نسبتا آسانتر است.

تنها مشکلی که وجود دارد این است که برای «ده» تا «پانزده» هیچ رقمی وجود ندارد که برای آن از شش حرف اول الفبای انگلیسی یعنی a تا f استفاده می‌شود.

مبنای شانزده

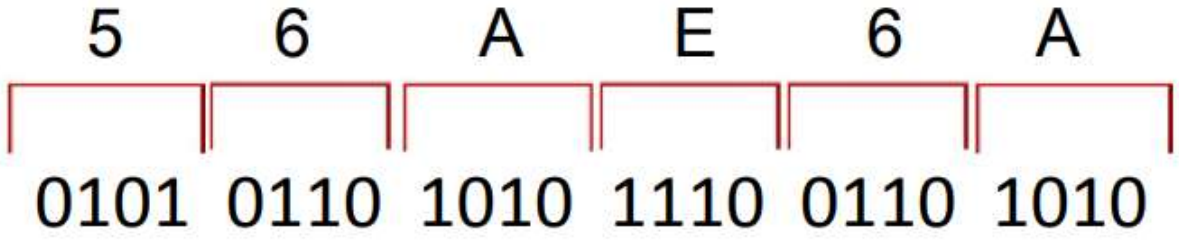
- تبدیل مبنای دو به ۱۶ و بالعکس بسیار آسان است.
- هر ۴ رقم باینری بطور مستقل به ۱ رقم هگزادسیمال تبدیل می‌شوند و بالعکس. (برای تبدیل ۴ رقم باینری به ۱ رقم هگزادسیمال باید این ۱۶ عدد را بصورت یک جدول داشته باشیم یا ذهنی محاسبه کنیم)

$$01010110101011100110101010_2$$

مبنای شانزده

- تبدیل مبنای دو به ۱۶ و بالعکس بسیار آسان است.
- هر ۴ رقم باینری بطور مستقل به ۱ رقم هگزادسیمال تبدیل می‌شوند و بالعکس. (برای تبدیل ۴ رقم باینری به ۱ رقم هگزادسیمال باید این ۱۶ عدد را بصورت یک جدول داشته باشیم یا ذهنی محاسبه کنیم)

$01010110101011100110101010_2$ $56AE6A_{16}$



جدول تبدیل

decimal	hexadecimal	binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

جمع و تفریق در مبنای ۲

- روش محاسبه مانند مبنای ۱۰ (روش دبستانی) جمع و تفریق است با این تفاوت که آنجا ده بر یک داشتیم و اینجا دو بر یک (در جمع نقلی و در تفریق قرضی).

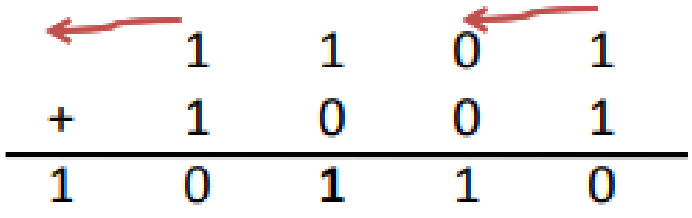
	1	1	0	1
+	1	0	0	1

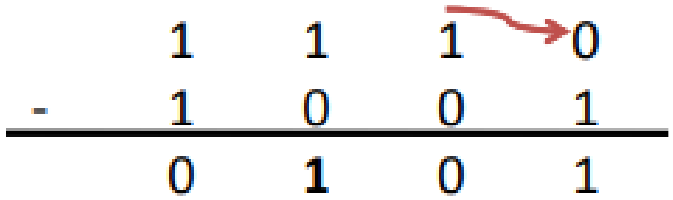
	1	1	1	0
-	1	0	0	1

- چون اندازه عدد روی کاغذ محدود نیست، ما مساله‌ای بنام سرریز نداریم. اما در کامپیوتر، مثلا اگر اندازه اندازه فضای ما چهار بیت باشد در جمع آخرین ارقام سمت چپ ما امکان سرریز داریم.

جمع و تفریق در مبنای ۲

- روش محاسبه مانند مبنای ۱۰ (روش دبستانی) جمع و تفریق است با این تفاوت که آنجا ده بر یک داشتیم و اینجا دو بر یک (در جمع نقلی و در تفریق قرضی).





- چون اندازه عدد روی کاغذ محدود نیست، ما مساله‌ای بنام سرریز نداریم. اما در کامپیوتر، مثلا اگر اندازه اندازه فضای ما چهار بیت باشد در جمع آخرین ارقام سمت چپ ما امکان سرریز داریم.

اعداد منفی باینری

- تا اینجا فرض بر آن بود که اعداد مثبت هستند. برای نمایش اعداد منفی (عدد را یک بیت بزرگتر در نظر گرفته) پرارزش‌ترین رقم را خالی می‌گذاریم (برای علامت رزرو می‌کنیم).
- برای نمایش اعداد منفی سه روش متفاوت وجود دارد که هر سه متداول‌اند.

1. روش علامتدار
2. روش مکمل ۱
3. روش مکمل ۲

روش علامتدار

- در این روش پرارزش‌ترین بیت، بعنوان بیت علامت منها در نظر گرفته می‌شود.

00001100 = +12 فضا را ۸ بیتی فرض می کنیم

10001100 = - 12

- سیستم با مشاهده علامت، متوجه منفی بودن عدد می‌شود و به جای جمع منها و به جای منها جمع می‌کند.

روش مکمل-یک و مکمل-دو

- در مکمل-یک تمامی بیت‌های عدد منفی برعکس می‌شوند (صفرها یک و یک ها صفر):

  +12 =

  -12 =

- در مکمل-دو ما 1 را به مکمل-یک اضافه می‌کنیم:

  -12 =

- توجه: برای کار با این سیستم باید حداقل **یک بیت اضافه‌تر** در نظر بگیریم (حتی اگر عدد مثبت و بیت اضافه صفر باشد) **طول دو عملوند هم باید مساوی** شود. این کار با افزودن 0 به سمت چپ عملوندها ممکن است. بنابراین مثلا با ۸ بیت حداکثر ۱۲۷ را می توان نشان داد، و نه ۲۵۵.

روش مکمل-یک و مکمل-دو

- در مکمل-یک تمامی بیت‌های عدد منفی برعکس می‌شوند (صفرها یک و یک ها صفر):

+12 = 00001100

-12 = 11110011

- در مکمل-دو ما 1 را به مکمل-یک اضافه می‌کنیم:

-12 = 11110100

- توجه: برای کار با این سیستم باید حداقل **یک بیت اضافه‌تر** در نظر بگیریم (حتی اگر عدد مثبت و بیت اضافه صفر باشد) **طول دو عملوند هم باید مساوی** شود. این کار با افزودن 0 به سمت چپ عملوندها ممکن است. بنابراین مثلا با ۸ بیت حداکثر ۱۲۷ را می توان نشان داد، و نه ۲۵۵.

جمع و تفریق مکمل ۲

- کافی است اعداد را (چه مثبت چه منفی) با هم جمع کنیم. برای تفریق عملوند دوم منفی می‌شود. اگر حال عددی منفی باشد به شکل مکمل ۲ خواهد بود.

مثال 31 – 6 = 25

31: 011111

6: 000110

-6: مکمل ۱

-6: مکمل ۲

- اگر رقم نقلی به پرارزشترین بیت (علامت) وارد شده باشد باید از آن خارج شود و اگر نشده باشد نباید خارج شود وگرنه سرریز شده است.

+:

25:

کد BCD

- در کدینگ‌ها، دیگر تبدیل مبنا و نمایش واقعی عدد مطرح نیست بلکه به طور دلخواه و با هدفی معین تعدادی 0 و 1 به یک رقم، عدد، یا حرف کاراکتری نسبت داده می‌شود.
- اگر تعامل با انسان (که نیازمند مبنای 10 است)، بخش عمده یک سیستم دیجیتال باشد می‌توان به جای اعداد باینری از کد BCD استفاده کرد.

کد BCD

- در این کد هر چهار بیت برای نمایش یک رقم ده دهی استفاده می‌شود. در واقع یک سوم حافظه در این شرایط هدر می‌رود اما در عوض تبدیل آن به اعداد ده دهی بسیار ساده است (شبیه هگزادسیمال).
- مثال: 0001 0100 1001 = 149
- در جمع BCD هرگاه حاصل بزرگتر از ۹ باشد آن را با (0110) جمع نموده و رقم نقلی هم ارسال می‌کنیم (چون چهار بیت بیش از ۹ نمی‌شود).
- اعداد BCD چندین رقمی به راحتی روی نمایشگر عددی (7-Segment) نمایش داده می‌شوند (مثل نمایشگر پمپ بنزین‌ها).

کد BCD

1 + 7 =

جمع بالا دقیقا عین جمع باینری است. چون حاصل کمتر از ۱۰ است نقلی به رقم بعد نمی‌فرستیم.

4 + 7 =

1011 + **0110** = 0001 =(1)

حاصل 1 می‌شود. به نقلی جمع دومی آخر کاری نداریم و در هر حال یک نقلی به رقم بعد می‌فرستیم.

9 + 9 =

0010 + **0110** = 1000 =(8)

حاصل این بار 8 می‌شود. این بار جمع دومی نقلی ندارد اما اولی نقلی داشته که باز کاری با آن نداریم و به رقم بعد نقلی می‌فرستیم.

کد BCD

$1 + 7 = 0001 + 0111 = 1000 = (8)$

جمع بالا دقیقا عین جمع باینری است. چون حاصل کمتر از ۱۰ است نقلی به رقم بعد نمی‌فرستیم.

$4 + 7 = 0100 + 0111 = 1011 = (11)$

$1011 + \mathbf{0110} = 0001 = (1)$

حاصل 1 می‌شود. به نقلی جمع دومی آخر کاری نداریم و در هر حال یک نقلی به رقم بعد می‌فرستیم.

$9 + 9 = 1001 + 1001 = 0010$

$0010 + \mathbf{0110} = 1000 = (8)$

حاصل این بار 8 می‌شود. این بار جمع دومی نقلی ندارد اما اولی نقلی داشته که باز کاری با آن نداریم و به رقم بعد نقلی می‌فرستیم.

کد گری

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Gray code

- این کد ۴ بیتی شبیه کد هگزادسیمال است اما در این کد هر عدد با عدد قبل و بعد فقط در یک بیت تفاوت دارد.
- یکی از کاربردهای این کد اینکه این احتمال خطا را در سنسورهایی که داده‌های پیوسته را دیجیتایز و ارسال می‌کنند را کم می‌کند.

کد برای حروف الفبا

- کد اسکی یک کد هفت بیتی است برای کاراکترها که در آن هر کد به یکی از حروف بزرگ و کوچک، ارقام و علائم پرکاربرد در الفبای انگلیسی اختصاص یافته است.
مثال Q: 1010001 (81)
- از آنجا که ۷ بیت از ۸ بیت استفاده شده در فضای باقیمانده می‌توان علائم کم کاربرد و یا الفبای (فقط) یک زبان دیگر را تعریف کرد.
- امروزه برای نمایش حروف الفبای بقیه زبان‌ها از یونیکد استفاده می‌شود که دو بایتی است.

مثال: هر پیامک شامل حداکثر ۱۱۲۰ بیت است که می‌تواند ۱۶۰ کاراکتر اسکی یا ۷۰ کاراکتر یونیکد باشد.

کد برای حروف الفبا

Unicode	Contextual forms			
	Final	Medial	Initial	Isolated
U-0621	N/A	N/A	N/A	ء
U-0623	ـأ		أ	
U-0626	ـئ	ـئـ	ئـ	ئ
U-0624	ـؤ		ؤ	
U-0627	ـا		ا	
U-0628	ـب	ـبـ	بـ	ب
U-067E	ـپ	ـپـ	پـ	پ
U-062A	ـت	ـتـ	تـ	ت
U-062B	ـث	ـثـ	ثـ	ث
U-062C	ـج	ـجـ	جـ	ج
U-0686	ـچ	ـچـ	چـ	چ
U-062D	ـح	ـحـ	حـ	ح

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
?	63	0077	0x3f	_	95	0137	0x5f				

مدارهای منطقی

(طراحی سیستم‌های دیجیتال)

جلسه سوم

گیت‌های منطقی و جبر بول

بحث امروز

- جبر بولی
- گیت‌های اصلی
- تابع‌های بولی
- سایر گیت‌های بولی
- IC های دیجیتال

جبر

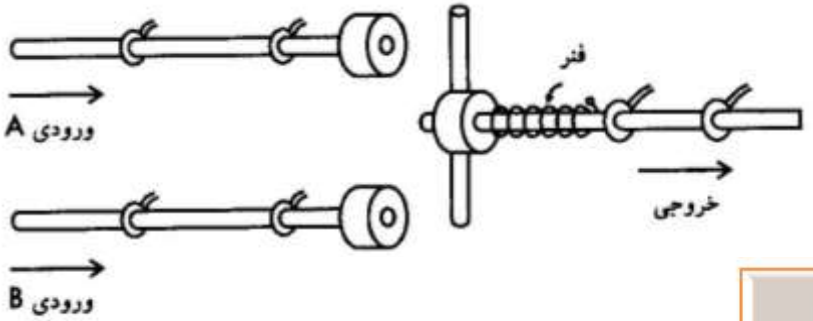
- جبر سیستمی است ریاضی. بر خلاف حساب که در آن فورمول‌نویسی فقط برای محاسبه اعداد (2+4/2) استفاده می‌شود، در جبر نمادهایی (حروف الفبا) بجای مقادیر نامشخص می‌نشینند (x=3+y/3).
- زبان جبر برای فورمول‌نویسی‌های محاسباتی و جبری بسیار خلاصه‌تر و دقیق‌تر از زبان طبیعی (مثل فارسی) است ولی قدرت بیان آن (برای همه مطالب از جمله مطالب غیر ریاضی) از زبان طبیعی بسیار کمتر است.

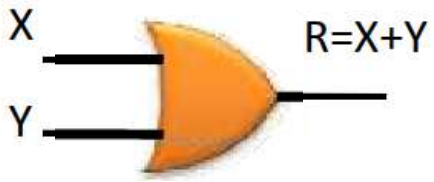
جبر بولی

- در هر سیستم جبری دامنه (مجموعه مقادیر مجاز) و نیز عملگرهای تعریف شده وجود دارد. مثلا در یک سیستم جبری دامنه می‌تواند اعداد صحیح و عملگرها جمع و ضرب معمولی باشند. این سیستم جبر و حساب برای محاسبات روزمره ابداع شده است.
- جورج بول ریاضی‌دان انگلیسی بود. او ابداع‌کننده یک سیستم ریاضی به نام جبر بول است.
- جبر بول که بر اساس نظام دو مقداری (درست/غلط) با هدف فورموله کردن منطق ابداع شد. بعدها با اختراع سیستم‌های دیجیتال که بر اساس (0/1) بودند، این سیستم ریاضی به عنوان پایه ریاضی طراحی دیجیتال به کار گرفته شد.
- در سیستم‌های دیجیتال دامنه دو سطح hi و low ولتاژ است (معمولا صفر و پنج ولت) و نیز عملگرها گیت هستند.

گیت‌های منطقی: "یا"

- علامت آن "+" است که خوانده می‌شود یا.





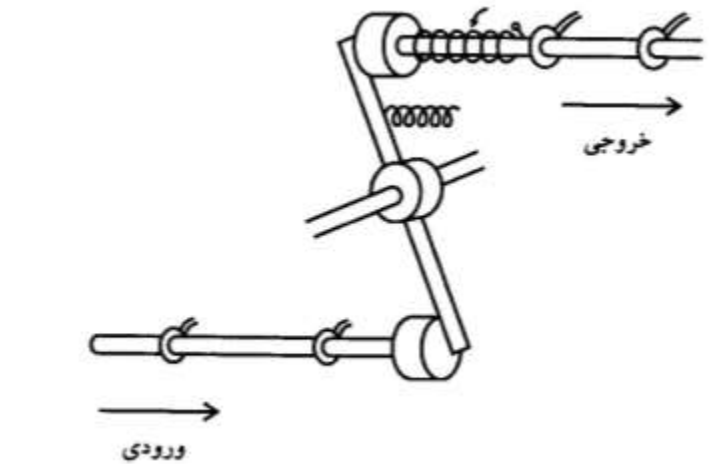
OR	0	1
0	0	1
1	1	1

X	Y	R=X OR Y R= X + Y
0	0	0
0	1	1
1	0	1
1	1	1

1 + Y = 1

گیت‌های منطقی: "نه"

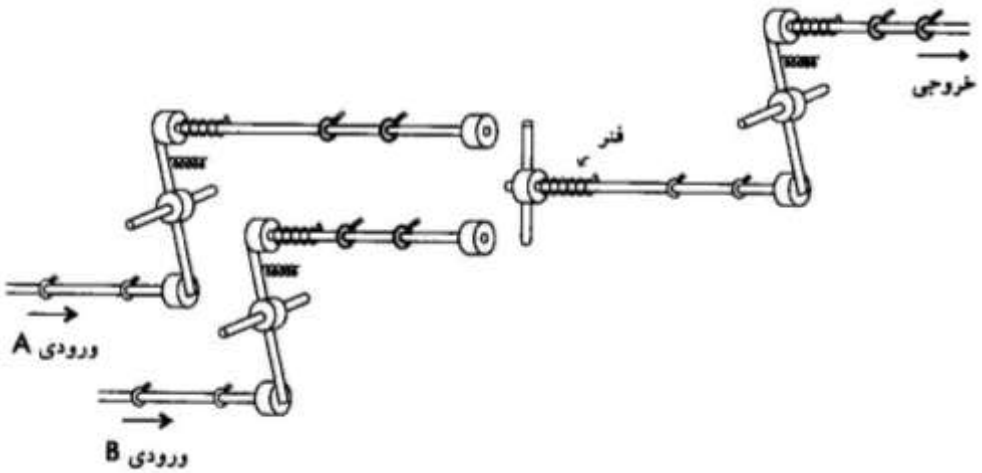
- علامت آن ’ (پریم) یا "¬" است که خوانده می‌شود نه.

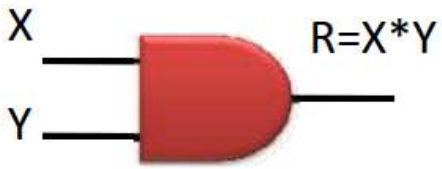


X	R=X’ R= NOT X
0	1
1	0

گیت‌های منطقی: "و"

- علامت آن "." یا "X" است که خوانده می‌شود وَ.





AND	0	1
0	0	0
1	0	1

	Y	R=X AND Y R= X * Y
0	0	0
0	1	0
1	0	0
1	1	1

0 * Y = 0

ویژگی‌های جبر بول

- دو عملگر دوتایی دارد + و . و یک مکمل کننده ’.
- جبر بول بسته است. یعنی اگر a و b بولی (باینری) باشند آنگاه (a + b) و همچنین (a . b) نیز 0 و 1 خواهد بود.
- دارای عضو خنثی است که 0 برای + و 1 برای x می‌باشد (عین جبر معمولی). یعنی

$$(a+0) = a$$

$$(a.1) = a$$

- جابجایی پذیر است. یعنی

$$(a+b) = (b+a)$$

$$(a.b) = (b.a)$$

ویژگی‌های جبر بول

- توزیع پذیر است. یعنی

a.(b+c) = (a.b)+(a.c)

a+(b.c) = (a+b).(a+c) (این دومی در جبر معمولی نیست)

- معکوس‌پذیری .....:

$$a + X = 0$$

$$a \cdot X = 1$$

- هر مقدار یک مکمل دارد، این مفهوم در جبر معمولی نیست و از نظریه مجموعه‌ها آمده است.

$$a + a' =$$

$$a \cdot a' =$$

قضایای نام دار جبر بول

تکرر	a+a = a	a.a = a
کرانداری	a+1 = 1	a.0 = 0
جذب	a+(a.b) = a	a.(a+b) = a
شرکت‌پذیری	a+(b+c)=(a+b)+c	a.(b.c)=(a.b).c
دولا شدگی	(a')' = a	
دمورگان	**(a+b)'= a'.b'**	**(a.b)'= a'+b'**

- **دوگانی:** اگر عبارتی صحیح باشد دوگان (لنگه دیگر) آنهم صحیح است. لنگه هر عبارت آن است که + را تبدیل به . و 0 را تبدیل به 1 و بالعکس.
تمام ویژگی‌ها و قضایای قبل دولنگه‌ای هستند.

توسعه قضایا

- جابجایی پذیری، شرکت‌پذیری و دمورگان قابل توسعه به هر تعداد متغیر هستند (محدود به دو متغیر نیستند). مثلا:

$$a + ((b+c)+d) = a + b + c + d$$

$$((d + a) + b) + c = a + b + c + d$$

$$(a + b + c + d)' = a'b'c'd'$$

اثبات قضایای بولی با جدول درستی

- بر خلاف جبر معمولی که دامنه آن بی‌نهایت عدد است، دامنه جبر بول فقط 0 و 1 است بنابراین می توان به راحتی همه حالات ممکن را با جدول درستی نشان داده و درستی (یا نادرستی) یک عبارت را بررسی و اثبات کرد.

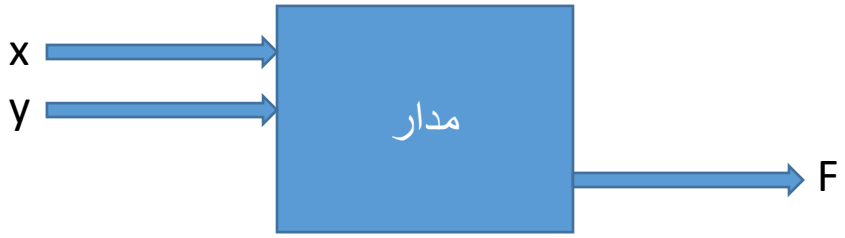
a	b	c	b*c	a+(b*c)	a+b	a+c	(a+b)*(a+c)
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

مثال: اثبات توزیع‌پذیری جمع

a+(b.c)=
(a+b).(a+c)

تابع بولی

- یک عبارت است که تعدادی (مثلا ۳) متغیر دارد. مقدار هر یک از متغیر ها می‌تواند 0 یا 1 باشد و مقدار تابع که بر اساس آنها تعیین می‌شود هم یا 0 یا 1 خواهد بود.
- مقدار تابع بولی را می‌توان با جدول درستی بررسی کرد.
- تابع بولی را می‌توان با یک مدار که از گیت‌های منطقی درست شده است، پیاده سازی کرد. در واقع متغیرها سیم‌های برقی هستند که بعنوان ورودی وارد مدار می‌شوند و تابع سیمی است که بعنوان خروجی از آن خارج می‌شود.



تابع بولی؛ یک مثال ساده

$$F1 = x + y'z$$

x	y	z	F_1

تابع بولی؛ یک مثال ساده

$$F1 = x + y'z$$

x	y	z	F_1	F_2
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0

تابع بولی؛ یک مثال دیگر

$$F2 = x'y'z + x'yz + xy'$$

x	y	z	F_1	F_2
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0

فاکتورگیری از x’z با استفاده از ویژگی توزیع‌پذیری

$$x'y'z + x'yz + xy' =$$

تابع بولی؛ یک مثال دیگر

$$F2 = x'y'z + x'yz + xy'$$

x	y	z	F_1	F_2
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0

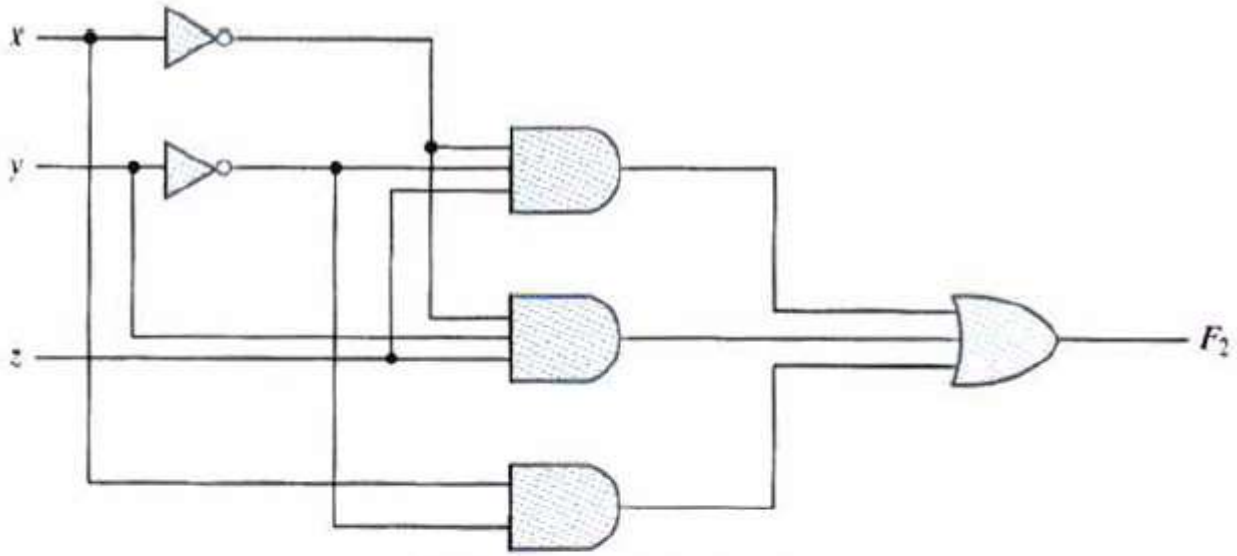
فاکتورگیری با استفاده
از ویژگی توزیع‌پذیری

$x'y'z + x'yz + xy' =$

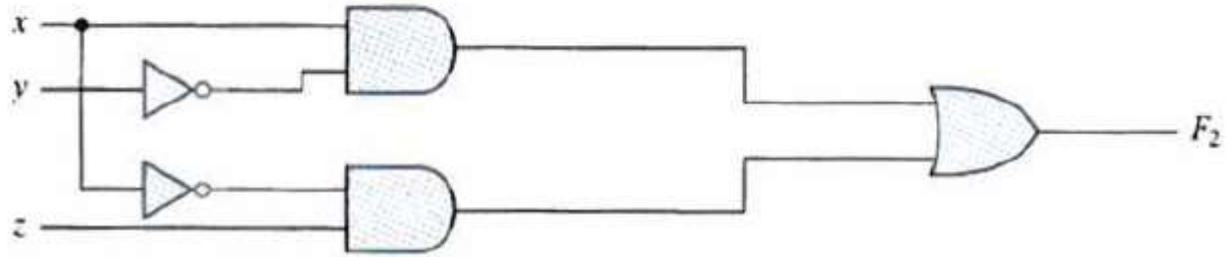
$x'z(y' + y) + xy' =$

$x'z + xy'$

تفاوت طراحی دو فرمول یکسان



(a) $F_2 = x'y'z + x'yz + xy'$



(b) $F_2 = xy' + x'z$

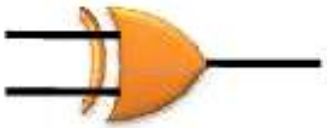
- هر دو یک کار را می‌کنند اما پایینی ارزانتر، سبکتر و کم‌مصرف‌تر است (چون فورمولش ساده‌تر است).

فرمول نوشتن برای مسائل ساده

- برای مسائل ساده می‌توان مستقیما و بصورت ذهنی فرمول تابع خروجی را نوشت.
- مثلا: چهار نفر دور میزی نشسته‌اند و هر کدام یک دگمه فشاری جلویشان قرار دارد (چهار متغیر ورودی مدار از این شستی‌ها می‌آیند). مدار یک خروجی (تابع) دارد که لامپ وسط میز را روشن می‌کند. اگر بخواهیم:
  الف) وقتی همه دگمه ها را فشردند لامپ روشن شود.
  ب) اگر حداقل یک نفر دگمه‌اش را فشرد روشن شود.

سایر گیت‌های منطقی

- XOR



A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

- XNOR

A	B	A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

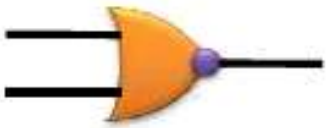
XOR خوانده می شود یای اختصاصی و علامتش ⊕

سایر گیت‌های منطقی

- NAND

A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

- NOR

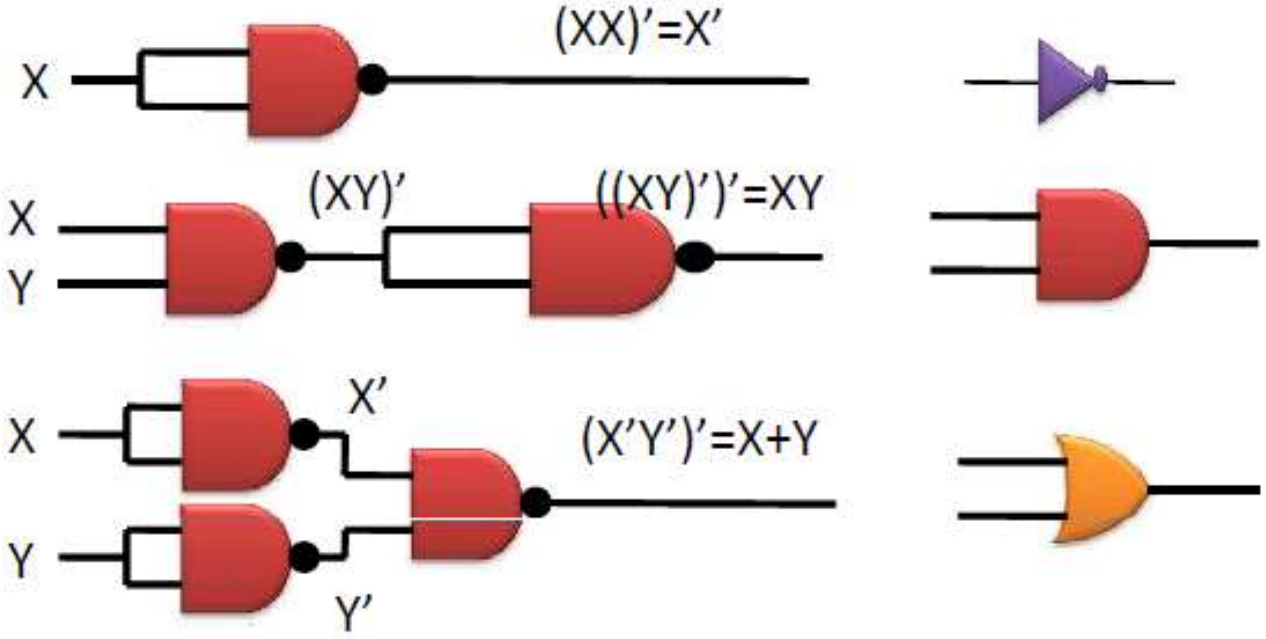


A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

مزیت این گیت‌های معکوس ارزان قیمت بودن آنهاست.

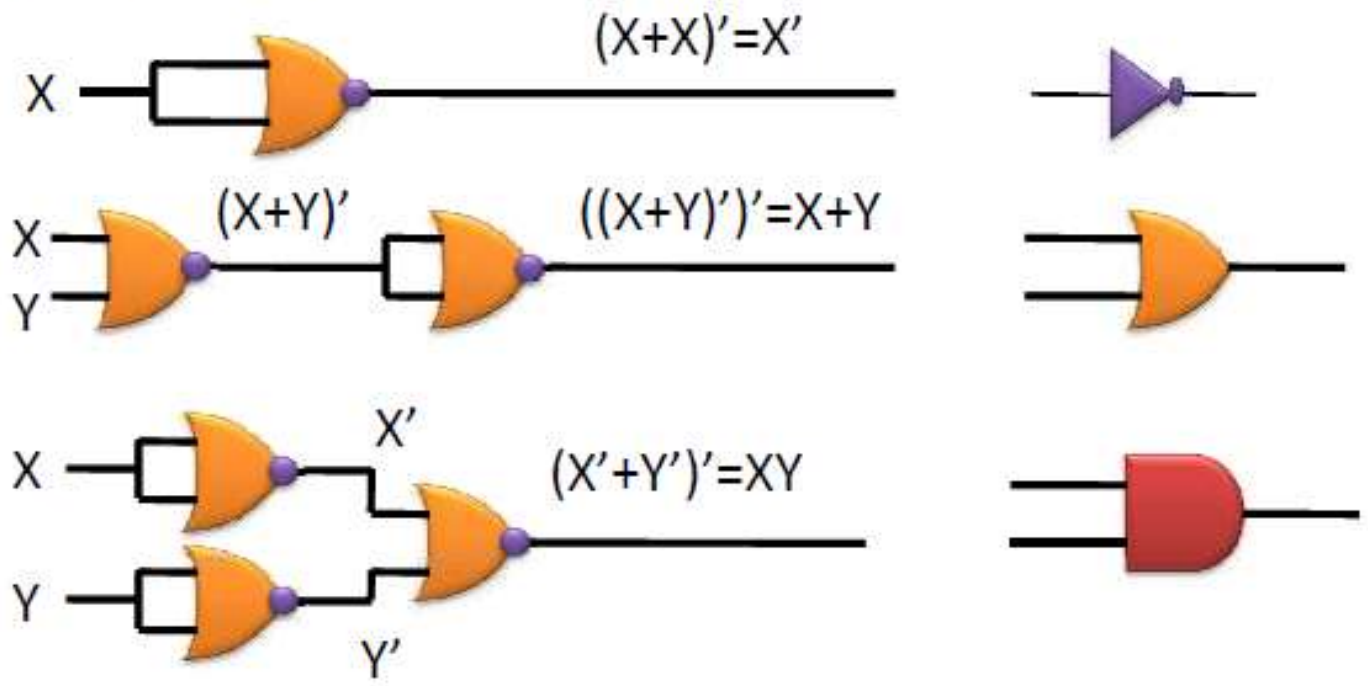
NAND همه‌کاره (یونیورسال) است

- (xx)’ = x’
- ((xy)’)’= xy
- (x’y’)’= x+y

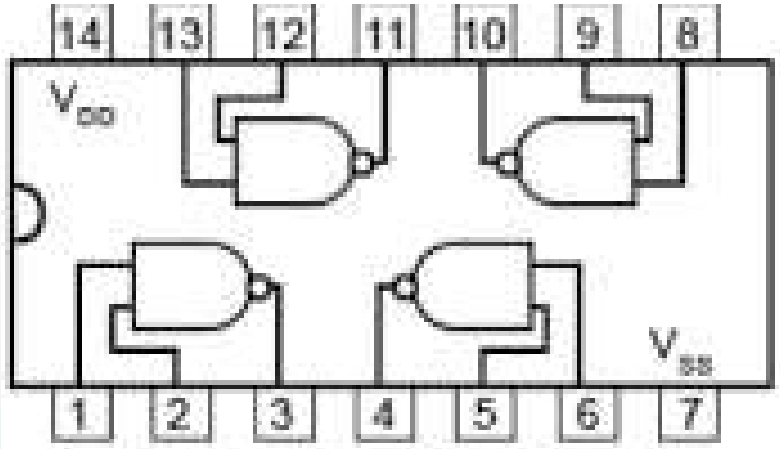


NOR هم همه‌کاره است

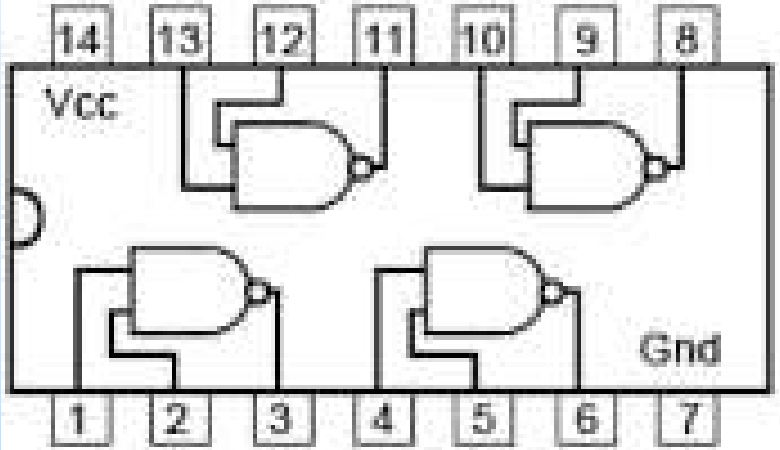
- $(x+x)' = x'$
- $((x+y)')' = x+y$
- $(x'+y')' = x.y$



IC های دیجیتال (منطقی)



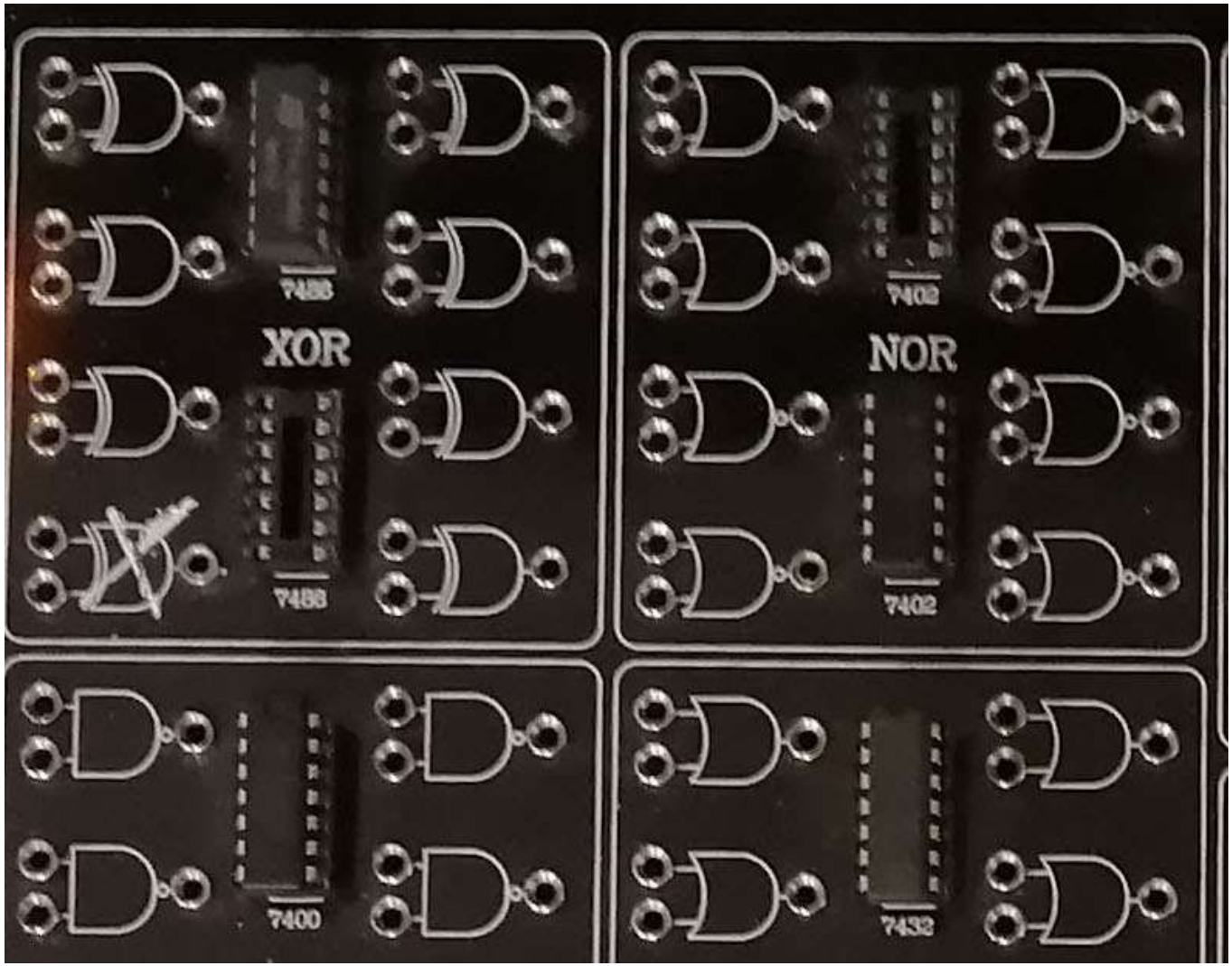
4011 CMOS NAND



7400 TTL NAND

- IC های منطقی از نظر ساختار الکترونیکی چند نوعند. همانطور که در شکل می‌بینید از نظر عملکرد منطقی با هم تفاوتی ندارند اما در کارایی تفاوت‌هایی دارند.
- برای نمونه دو نوع پراستفاده TTL و CMOS را با هم مقایسه می‌کنیم.

IC های دیجیتال (منطقی)



تفاوت IC های TTL و CMOS

مصرف	به شرایط گوناگون بستگی دارد اما در کل CMOS کم مصرف‌تر است و مناسب مدارات باطری‌دار می‌باشند.
گنجایش خروجی	هر خروجی TTL می‌تواند به حداکثر ۱۰ ورودی وصل شود و این تعداد در مورد CMOS ۵۰ است.
مقابله با نویز	هر دو عملکرد بسیار خوبی دارند.
تاخیر	با تغییر ورودی ۱۰ میلی ثانیه طول می‌کشد تا تغییر در خروجی TTL ظاهر شود واین برای CMOS ۷۰ است.
سازگاری	خروجی TTL می‌تواند ورودی CMOS را راه‌اندازی کند اما CMOS نمی‌تواند TTL را راه بیاندازد.
مقاومت فیزیکی	قطعات TTL مقاوم‌تر هستند اما قطعات CMOS آسیب‌پذیرند و حتی با تماس دست ممکن است بسوزند.

تمرین

- دو تابع زیر را در نظر بگیرید. اگر توانستید آنها را ساده کنید (در طول عملیات بنویسید از کدام ویژگی‌ها و قضایا استفاده کرده اید) و سپس مدار آن را رسم کنید.

$Fa = x(x+y)(x+z)$

$Fb = xyz' + x'y'$

مدارهای منطقی

(طراحی سیستم‌های دیجیتال)

جلسه چهارم

فورمول نویسی و ساده سازی

مباحث امروز

- تبدیل جدول درستی به فورمول
- فرم‌های کانونی، استاندارد و غیر استاندارد
- ساده‌کردن فورمول با جدول کارنو

از جدول درستی به تا مدار

- طراح مدار معمولا پس از درک مساله، در قدم اول جدول درستی مورد نظر برای مدارش را می‌کشد.
- در مراحل بعد این جدول باید تبدیل به یک فورمول بولی شود و بعد فورمول باید تا جایی که ممکن است ساده شود و سپس مدار بوسیله گیت‌ها طراحی گردد و نهایتا ساخت برد الکترونیکی انجام و IC روی آن مستقر می‌گردد.

جدول به فورمول

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- دو برخورد ممکن است. یکی اینکه 1 ها را در نظر بگیریم:

F1 = x’y’z

- و روش دیگر اینکه برای 0 ها فورمول بنویسیم:

F1 =

F1 =

- نوشتن به روش اول «جمله کوچک» یا مینترم و روش دوم «جمله بزرگ» یا ماکسترم نامیده می‌شود.

جدول به فورمول

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- دو برخورد ممکن است. یکی اینکه 1 ها را در نظر بگیریم:

$$F1 = x'y'z + xy'z' + xyz$$

- و روش دیگر اینکه برای 0 ها فورمول بنویسیم:

$$F1 = (x'y'z' + x'yz' + x'yz + xy'z + xyz')'$$

$$F1 = (x+y+z)(x+y'+z)(x+y'+z')(x'+y+z')(x'+y'+z)$$

- نوشتن به روش اول «جمله کوچک» یا مینترم و روش دوم «جمله بزرگ» یا ماکسترم نامیده می‌شود.

جدول به فورمول

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- حالا برای F2 هم می‌توانیم تابع به صورت مینترم و ماکسترم بنویسیم.
- برای ماکسترم نیازی به مرحله محاسبه نیست. فقط کافی است اگر متغیر 0 بود ساده و اگر 1 بود با پریم نوشته شود (برعکس مینترم).

$$F2 = x'yz + xy'z + xyz' + xyz$$

$$F2 = (x+y+z)(x+y+z')(x+y'+z)(x'+y+z)$$

روش دیگر نوشتن ماکسترم و مینترم

			Minterms		Maxterms	
x	y	z	Term	Designation	Term	Designation
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

F2 = x’yz + xy’z + xyz’ + xyz جمله کوچک با ام کوچک

F2 = m3 + m5 + m6 + m7 = ∑(3,5,6,7)

F2 = (x+y+z)(x+y+z’)(x+y’+z)(x’+y+z)

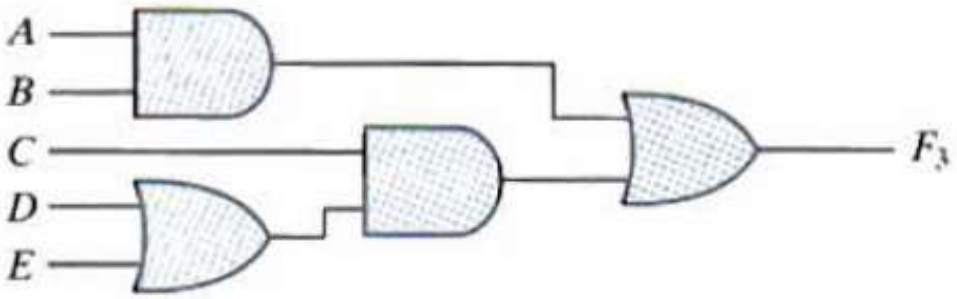
F2 = M0.M1.M2.M4 = ∏(0,1,2,4)

جمله بزرگ با ام بزرگ

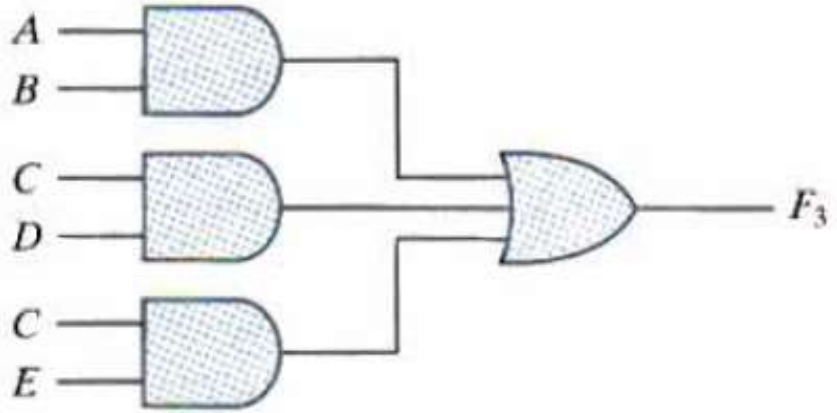
چند تعریف

- فرم‌های کانونی: عبارتی که به شکل مجموع مینترم‌ها یا ضرب ماکسترم‌ها باشد، در فرم کانونی است.
- فرم استاندارد: عبارتی که به طور کلی به شکل مجموع مضروبان SOP و یا مضروب مجموعان POS باشد، استاندارد است.
- در واقع فرم کانونی (که در هر جمله آن همه متغیرها حضوردارند) حالت خاصی از فرم استاندارد می‌باشد (مجموع مینترم‌ها نوعی SOP و ضرب ماکسترم‌ها نوعی POS است).
- فرم غیر استاندارد: عبارتی است که بصورت SOP یا POS نباشد.

مقایسه استاندارد و غیر استاندارد



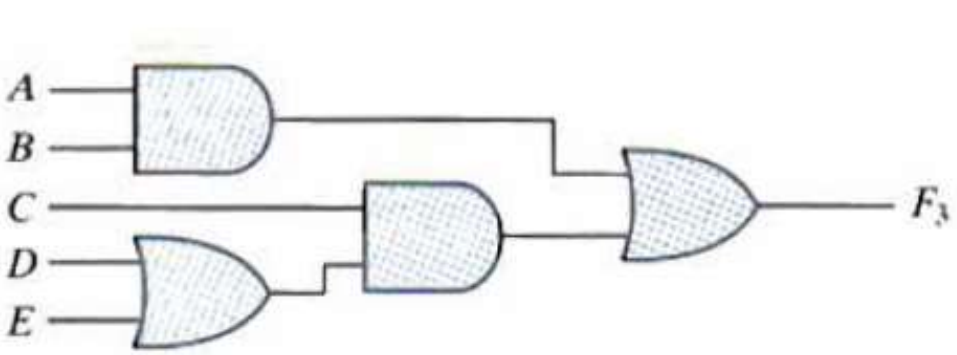
(a) $AB + C(D + E)$



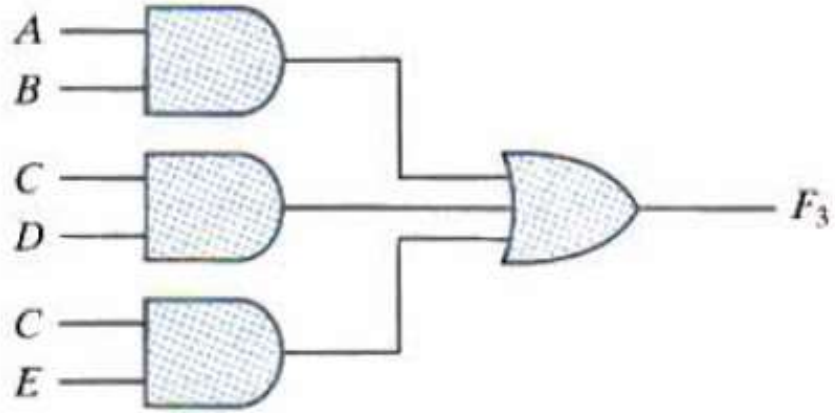
(b) $AB + CD + CE$

مقایسه استاندارد و غیر استاندارد

- در شکل زیر یک تابع به دو فرم غیراستاندارد (a) و استاندارد (b) نوشته شده است.
- در فرم استاندارد، مدار حداکثر در دو طبقه به نتیجه می‌رسد. تعداد طبقات از نظر تاخیر انتشار اهمیت دارد (فرض می‌شود که not ورودی‌ها ابتدا در دسترس است و بنابراین در طبقات و تعداد گیت‌ها شمرده نمی‌شود).



(a) $AB + C(D + E)$



(b) $AB + CD + CE$

ساده سازی

- مزایای فرم کانونی:

1. به راحتی از جدول درستی بدست می‌آید.
2. مقایسه تابع‌ها را آسان می‌کند (خصوصا اگر به شکل m یا M نوشته شود).
3. با حداقل تاخیر (دو گیت) پیاده‌سازی می‌شود.

- عیب فرم کانونی:

احتمالا ساده‌ترین فرم فورمول نیست و پیاده‌سازی با گیت‌های کمتر ممکن است امکان‌پذیر باشد.

ساده سازی

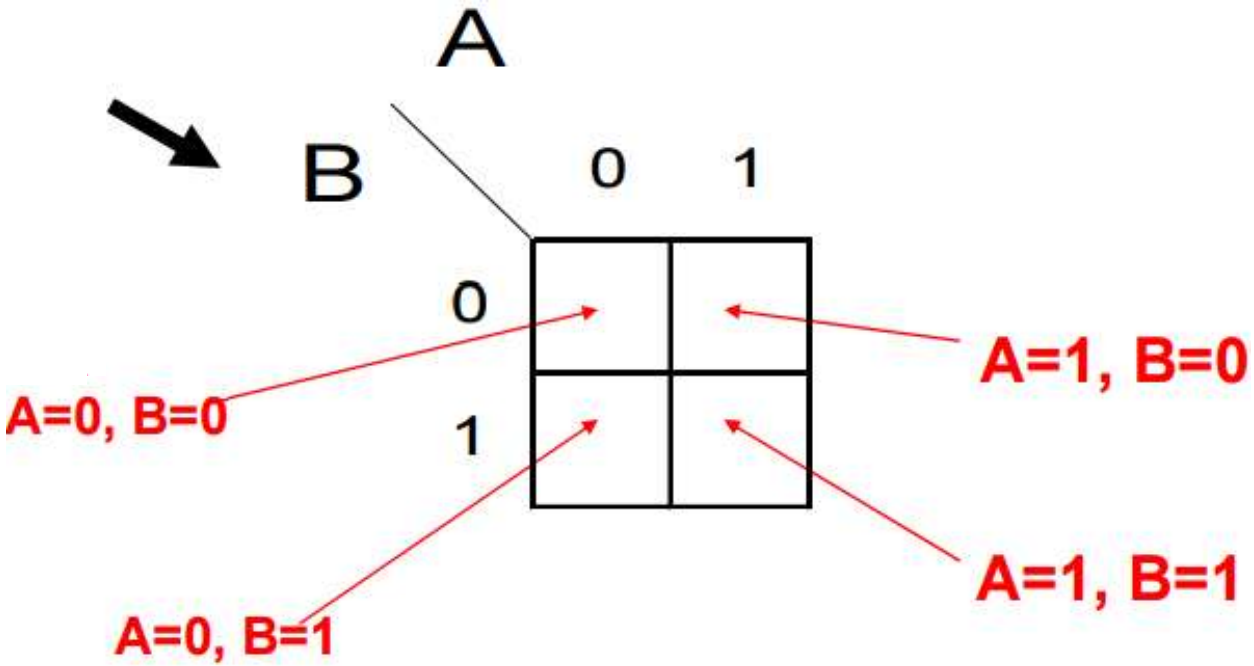
- برای ساده‌سازی فورمول‌ها می‌توان از دستکاری جبری (مثل فاکتورگیری) استفاده کرد، اما چون این روش سیستماتیک نیست معلوم نیست که بتوانیم چگونه عمل کنیم.
- برای تبدیل فرم‌های کانونی به فرم استاندارد ساده‌تر از روش کشیدن جدول کارنو استفاده می‌شود.

جدول کارنو

- یک روش دیداری برای ساده کردن فرم‌های استاندارد است.
- نسبت به محاسبات جبری هم قابل فهم و انجام‌تر است و هم نتیجه‌اش تضمین‌شده است. همینطور روش جبری ممکن است به فورمول غیراستاندارد برسد.
- از نظر تئوری با هر تعداد متغیر قابل رسم است، اما در عمل برای توابعی با ۲ تا ۴ متغیر به راحتی رسم می‌شود برای ۵ یا ۶ متغیر رسم آن مشکل ولی ممکن است و برای متغیرهای بیشتر آنقدر مشکل می‌شود که عملا قابل انجام نیست.

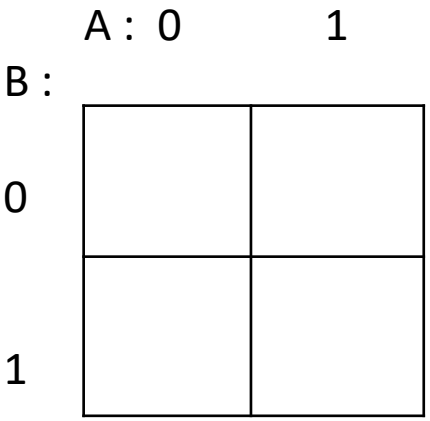
جدول کارنو برای فورمول دو متغیره

- برای دو متغیر یک جدول ساده ۲ X ۲ روی کاغذ می‌کشیم.



- در جدول کارنو با هر ابعادی، هر خانه با خانه‌های بالا، پایین، چپ و راست همسایه است. جدول از لبه محدودیت ندارد و اگر از یک طرف خارج شویم از طرف دیگر وارد می‌شویم. هر دو خانه همسایه فقط در یک بیت اختلاف دارند.

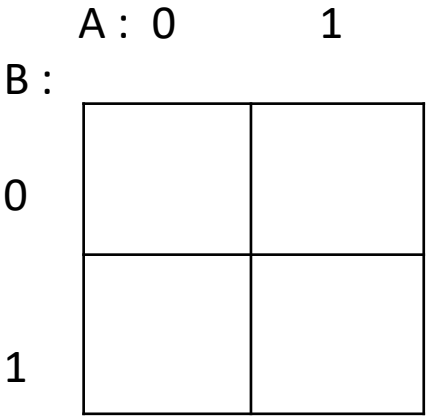
مثالی از جدول کارنو دو متغیره



$$F = A'B' + A'B$$

- پس از کشیدن جدول خالی، بر اساس فورمول تابع داخل خانه‌های جدول صفر و یک می‌گذاریم.

مثالی دیگر از جدول کارنو دو متغیره



$$F = A'B' + A'B + AB' + AB$$

- اگر بتوان دور ۴ همسایه خط کشید ۴ عبارت یکی شده و ۲ متغیر حذف می‌شوند. F =
- در جدول‌های بزرگتر ممکن است ۸ یا ۱۶ یا .... همسایه دورشان خط کشیده شده و ۳ یا ۴ یا .... متغیر حذف شوند.

جدول کارنو سه متغیره

A :	BC: 00	01	11	10
0	m0	m1	m3	m2
1	m4	m5	m7	m6

- بصورت یک جدول ۲ X ۴ رسم می‌شود که هر خانه سه همسایه دارد (در واقع جدولی است سه بعدی که برای نمایش روی کاغذ دو بعدی شده).
- از آنجا که هر خانه با همسایگانش باید فقط در یک بیت تفاوت داشته باشد، جای دو ستون آخر عوض می‌شود (کد گری).

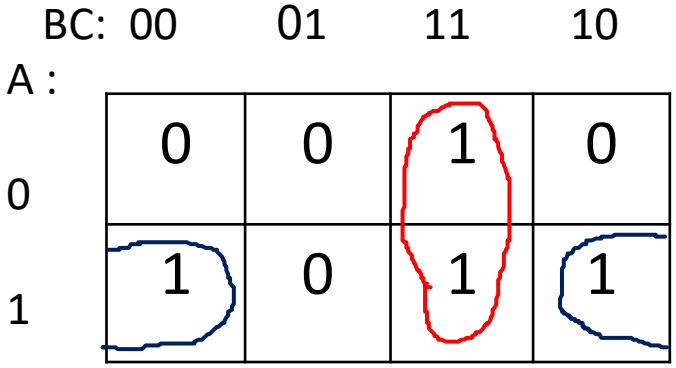
مثالی از جدول کارنو سه متغیره

A : BC:	00	01	11	10
0	m0	m1	m3	m2
1	m4	m5	m7	m6

تابع زیر را ساده کنید:

$$F = \sum(3,4,6,7)$$

مثالی از جدول کارنو سه متغیره



تابع زیر را ساده کنید:

$F = \sum(3,4,6,7)$

پاسخ:

$F = BC + AC'$

جدول کارنو چهار متغیره

AB : CD:	00	01	11	10
00	m0	m1	m3	m2
01	m4	m5	m7	m6
11	m12	m13	m15	m14
10	m8	m9	m11	m10

- بصورت یک جدول ۴ X ۴ رسم می‌شود که هر خانه چهار همسایه دارد (بالا پایین چپ راست).
- اینجا هم جای دو ستون آخر عوض می‌شود هم دو سطر آخر (باز هم مشابه روالی که در کد گری داشتیم).
- برای امتحان کردن کارنویی که کشیده‌اید:

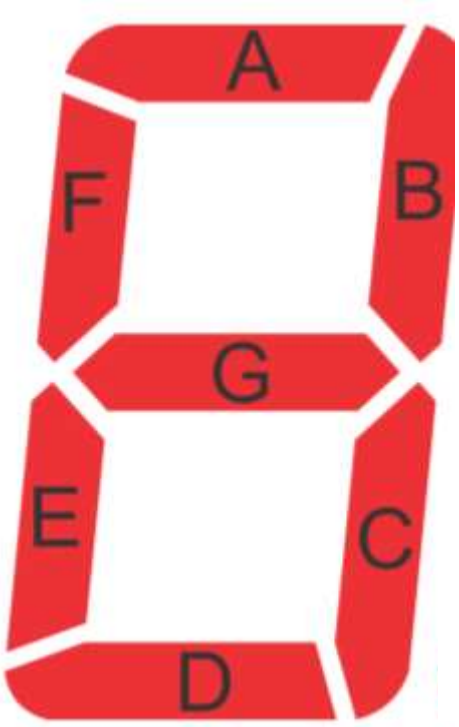
http://electronics-course.com/karnaugh-map

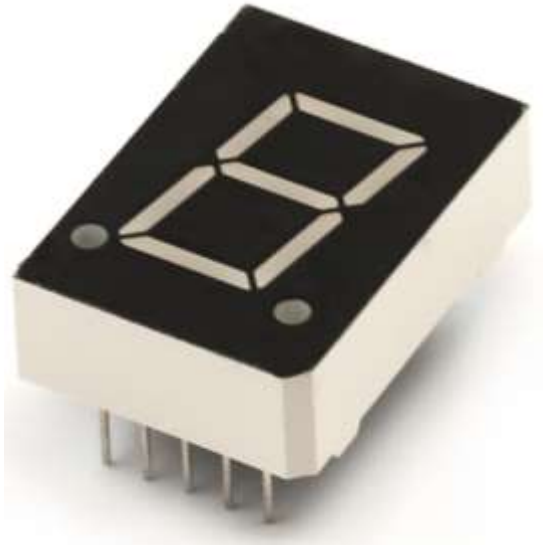
حالات بی اهمیت

- اگر در جدول درستی ما، حالاتی داشته باشد که 0 یا 1 بودن خروجی درآنها برای طراح هیچ فرقی نداشته باشد، در جدول درستی بجای آنها X قرار می‌دهیم.
- X ها را می‌توان در صورت نیاز با 1 ها یا با 0 ها دسته‌بندی کرد و دورشان خط کشید.

یک مثال کامل

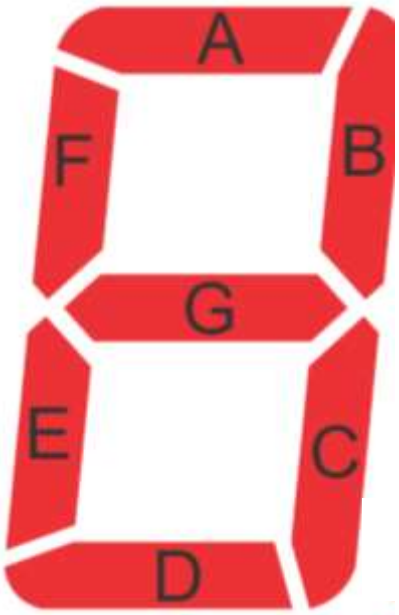
- سون سگمنت یک قطعه الکترونیکی است با شش لامپ که با روشن شدن برخی آنها یک رقم به شخص نشان داده می‌شود.

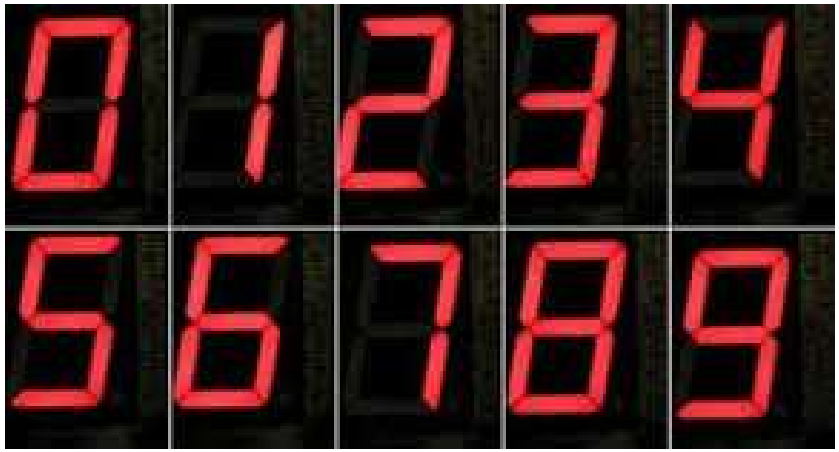




یک مثال کامل

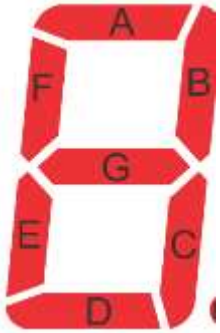
- مداری ساده شده رسم کنید که ۴ بیت عدد BCD را دریافت کند و بر اساس آن سگمنت A سون سگمنت را روشن کند.





- همانطور که می‌بینید، این سگمنت برای اعداد 0,2,3,5,6,7,8,9 روشن و برای بالای 9 (a تا f) بی‌تفاوت است (چون که اصلا قرار نیست اینها در ورودی ظاهر شوند).

جدول درستی و تابع

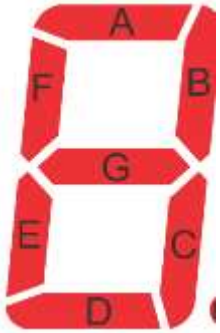


F(A,B,C,D) =

d(A,B,C,D) =

w	x	y	z	Hex	A
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	2	1
0	0	1	1	3	1
0	1	0	0	4	0
0	1	0	1	5	1
0	1	1	0	6	1
0	1	1	1	7	1
1	0	0	0	8	1
1	0	0	1	9	1
1	0	1	0	A	X
1	0	1	1	B	X
1	1	0	0	C	X
1	1	0	1	D	X
1	1	1	0	E	X
1	1	1	1	F	X

جدول درستی و تابع



$F(A,B,C,D) = \sum(0,2,3,5,6,7,8,9)$

$d(A,B,C,D) = \sum(10,11,12,13,14,15)$

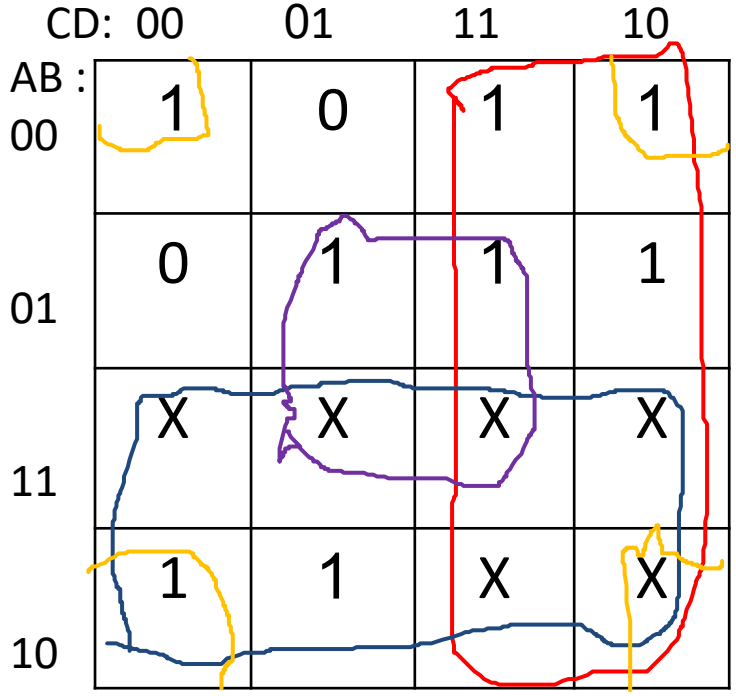
رسم جدول

AB : \ CD:	00	01	11	10
00	m0	m1	m3	m2
01	m4	m5	m7	m6
11	m12	m13	m15	m14
10	m8	m9	m11	m10

$$F(A,B,C,D) = \sum(0,2,3,5,6,7,8,9)$$

$$d(A,B,C,D) = \sum(10,11,12,13,14,15)$$

رسم جدول



$F(A,B,C,D) = \sum(0,2,3,5,6,7,8,9)$

$d(A,B,C,D) = \sum(10,11,12,13,14,15)$

$F = C + A + BD + B'D'$

مثالی دیگر از جدول کارنو چهار متغیره

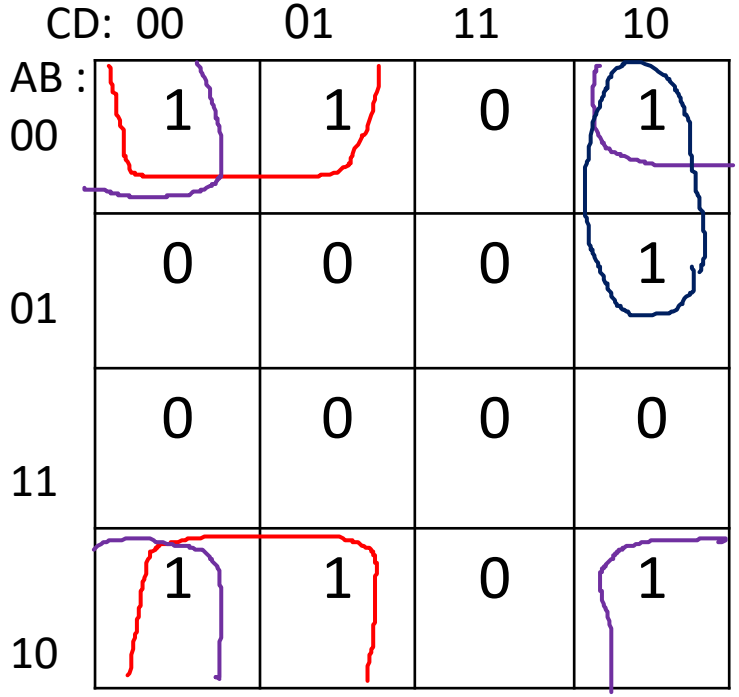
AB : \ CD:	00	01	11	10
00	m0	m1	m3	m2
01	m4	m5	m7	m6
11	m12	m13	m15	m14
10	m8	m9	m11	m10

تابع زیر را ساده کنید:

$$F = A'B'C' + B'CD' + A'BCD' + AB'C'$$

(توجه داشته باشید که این فورمول استاندارد هست اما کانونی نیست و ما می‌خواهیم یک فورمول استاندارد ساده‌تر بیابیم).

مثالی دیگر از جدول کارنو چهار متغیره



تابع زیر را ساده کنید:

$$F = A'B'C' + B'CD' + A'BCD' + AB'C'$$

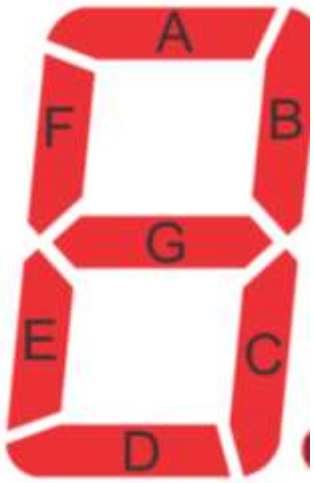
پاسخ:

$$F = B'D' + B'C' + A'CD'$$

تمرین

- برای تبدیل یک رقم BCD به یکی از سگمنت‌های 7Segment ۱- جدول درستی بکشید (اختیاری).
۲- تابع به شکل جمع مینترم‌ها $\sum m_x$ بنویسید. ۳- جدول کارنو رسم کنید و تابع را ساده کنید.
۴- تابع ساده شده را به صورت مدار رسم کنید.
سگمنتی که شما در نظر می‌گیرید:

آخرین رقم شماره دانشجویی	سگمنت مورد نظر
1 یا 2	B
3	C
4 یا 5	D
6 یا 7	E
8 یا 9	F
0	G



مدارهای منطقی

(طراحی سیستم‌های دیجیتال)

جلسه پنجم

مدارهای منطقی ترکیبی

مباحث امروز

- مدار منطقی ترکیبی چیست
- انواع مدارهای دوطبقه
- امکان اتصال خروجی‌ها
- مفهوم تاخیر و هَزارد
- تحلیل و طراحی مدارهای منطقی

مدار منطقی ترکیبی

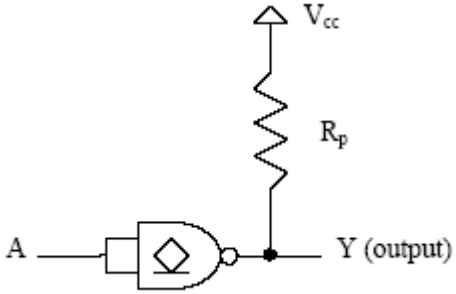
- یک مدار منطقی ترکیبی دارای n ورودی و m خروجی است که m و n می‌تواند هر عددی بزرگتر از صفر باشد.
- خروجی مدار ترکیبی در هر زمان فقط به ورودی‌ها بستگی دارد (مثل f که یک تابع است که در فورمول آن ورودی‌ها حضور دارند)، در مقابل مدار ترتیبی به ورودی‌ها و حالت قبلی مدار هم بستگی دارد.

انواع مدار منطقی ترکیبی

- مدارهای منطقی ترکیبی را معمولا به صورت دو طبقه طراحی می‌کنند (فورمول استاندارد).
- مدارهای قبلی به شکل (and-or) با در نظر گرفتن 1 های جدول درستی یا جدول کارنو و یا شکل (or-and) با در نظر گرفتن 0 های جدول درستی یا جدول کارنو طراحی می‌شدند.
- طراحی‌های دیگری نیز ممکن است که با توجه به ارزانی nand و nor برای تولید انبوه اقتصادی‌ترند:
  (nand-nand) (nor-nor) (nor-or)
  (nand-and) (nand-or) (nor-and)

اتصال خروجی‌ها

- در IC هایی مانند TTL معمولی و یا CMOS اتصال بین خروجی‌ها ممکن نیست. در صورت 1 شدن یکی و 0 شدن دیگری، جریان زیادی از دومی گذشته و آن را می‌سوزاند.
- در برخی انواع مانند TTL کلکتور باز و یا ECL اتصال ممکن است.
- در TTL کلکتور باز گره زده خروجی‌ها تولید and و در ECL تولید or می‌کند.
- در TTL کلکتور باز حالت 1 بدون ولتاژ (قطع) است و باید هر خروجی با یک مقاومت به +vcc وصل (pull up) شود.

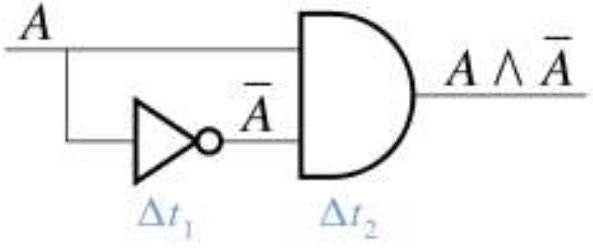


تاخیر

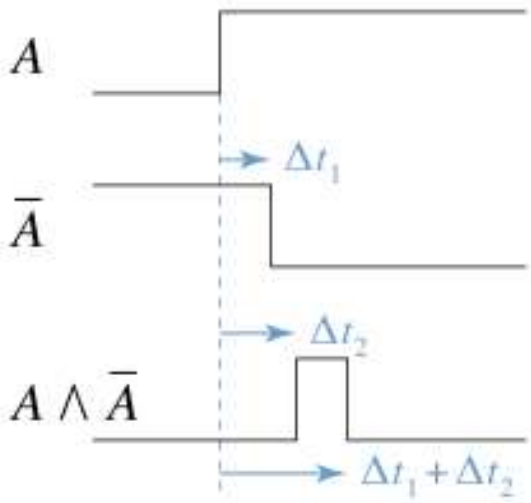
- هر چند از نظر تئوری خروجی یک مدار ترکیبی در هر لحظه از زمان وابسته به ورودی‌هاست، اما در عمل پس از تغییر ورودی مدتی طول می‌کشد تا نتیجه بر خروجی ظاهر شود.
- مدت تاخیر بستگی به نوع گیت‌ها و تعداد طبقات مدار ترکیبی دارد.
- قبل از تثبیت شدن خروجی اگر مقدار آن خوانده شود احتمالا دارای خطا خواهد بود.

هَزارد

- جمله AA’ همیشه نادرست است و پیاده سازی مدار آن هم باید همیشه خروجی 0 داشته باشد.



- اما در واقع پس از هر بار تغییر A برای مدت کوتاهی مقدار 1 در خروجی ظاهر می‌شود که علت آن تاخیر گیت not است به این اتفاق race hazard یا قلیچ (تقه) گفته می‌شود.



- اگر تاخیر و عواقب آن در طراحی مدار مد نظر قرار نگیرد مدار ساخته شده قابل اطمینان نخواهد بود.

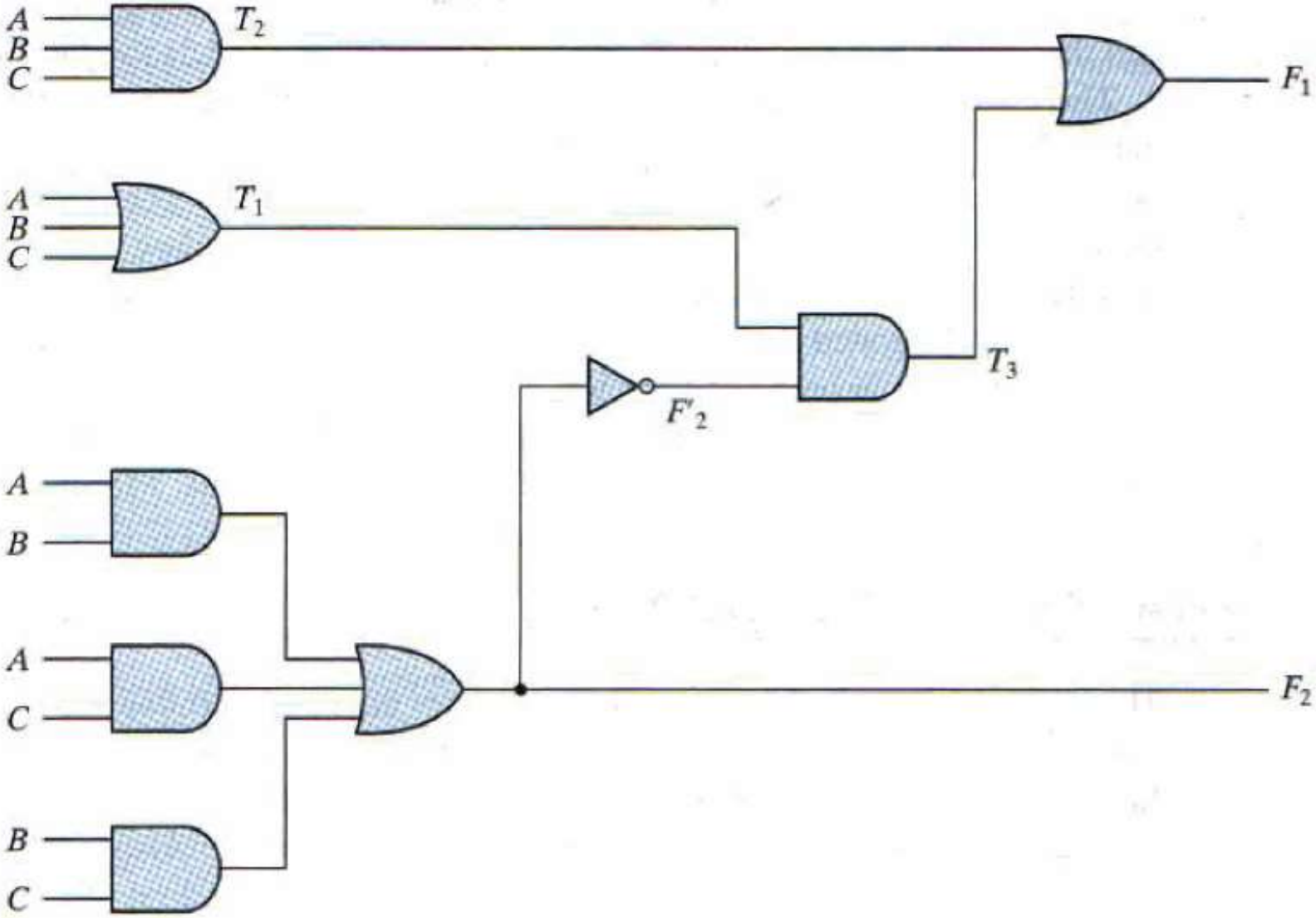
تحلیل مدار

- تحلیل مدار به کاری گفته می‌شود که در آن با داشتن یک مدار منطقی، تابع آن مدار را پیدا می‌کنیم و یا در صورت داشتن یک جدول درستی منطبق بودن کار مدار با آن جدول را بررسی می‌کنیم.

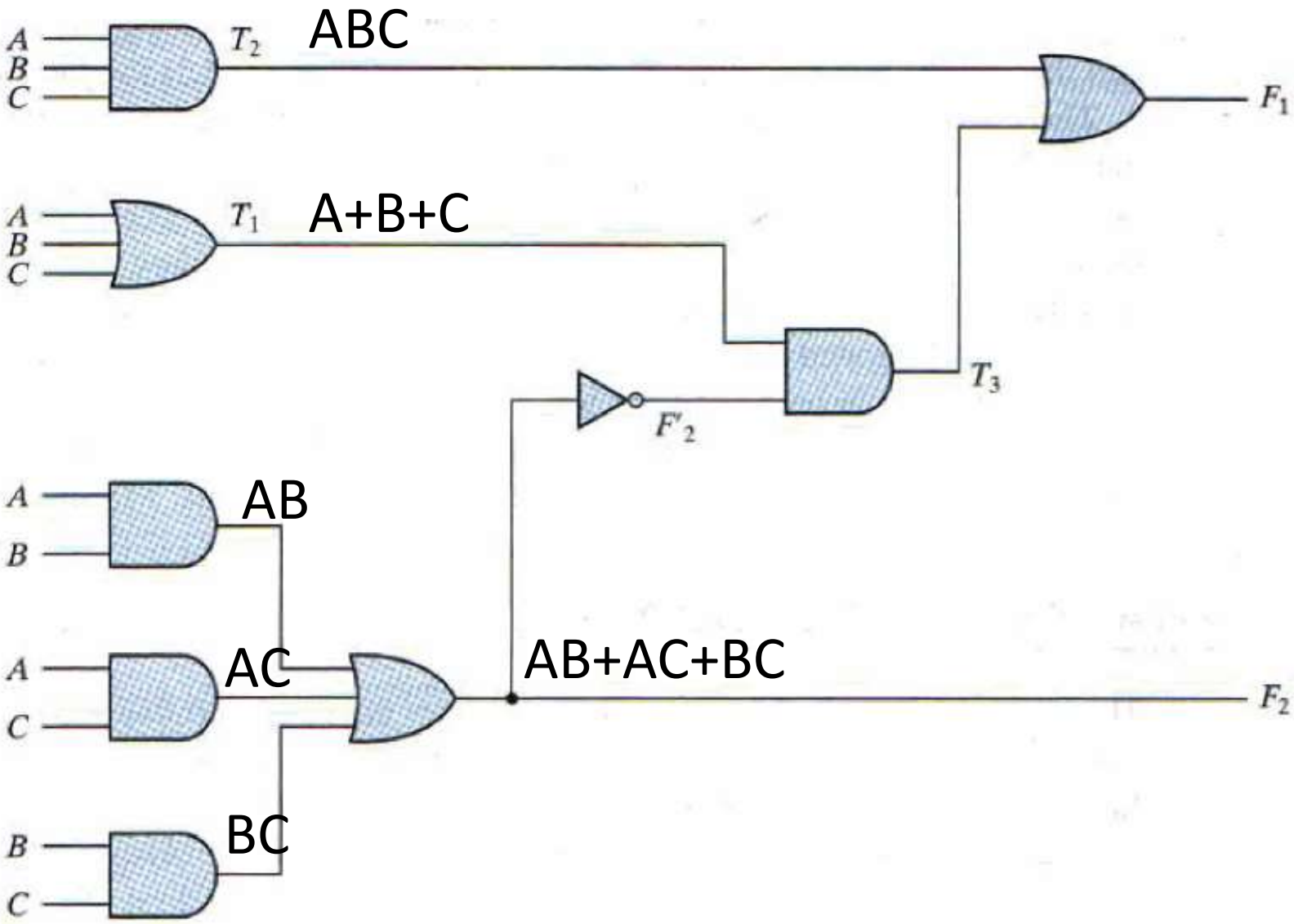
برای تحلیل یک مدار ترکیبی:

1. متغیرهای ورودی را به عنوان مقادیر معلوم در نظر می‌گیریم (اگر نام ندارند، نامی بر هر یک می‌گذاریم).
2. هر گیتی که تمام ورودی‌هایش معلوم است را در نظر گرفته و خروجی‌اش را بدست می‌آوریم و می‌نویسیم.
3. مرحله ۲ را آنقدر تکرار می‌کنیم تا خروجی نهایی پیدا شود.

مثال



مثال



ادامه مثال

$$F1 = T3 + T2 = F2'T1 + T2 =$$

$$(AB + AC + BC)' (A + B + C) + ABC$$

$$= (A' + B')(A' + C')(B' + C')(A + B + C) + ABC$$

$$= (A' + B'C')(AB' + AC' + BC' + B'C) + ABC$$

$$= A'BC' + A'B'C + AB'C' + ABC$$

می‌توانید حدس بزنید F1 چه کار می‌کند؟

ادامه مثال

$$F1 = T3 + T2 = F2'T1 + T2 =$$

$$(AB + AC + BC)' (A + B + C) + ABC$$

$$= (A' + B')(A' + C')(B' + C')(A + B + C) + ABC$$

$$= (A' + B'C')(AB' + AC' + BC' + B'C) + ABC$$

$$= A'BC' + A'B'C + AB'C' + ABC$$

می‌توانید حدس بزنید F1 چه کار می‌کند؟

فرد بودن تعداد 1 ها را نشان می‌دهد: توازن فرد

توجه: چون هدف از تحلیل ساخت مدار نیست (مدار را داریم)، نیازی به کارنو ندارد. قدری کار کردن با فورمول کفایت می‌کند.

فورمول کانونی معمولا کار مدار را نشان می‌دهد.

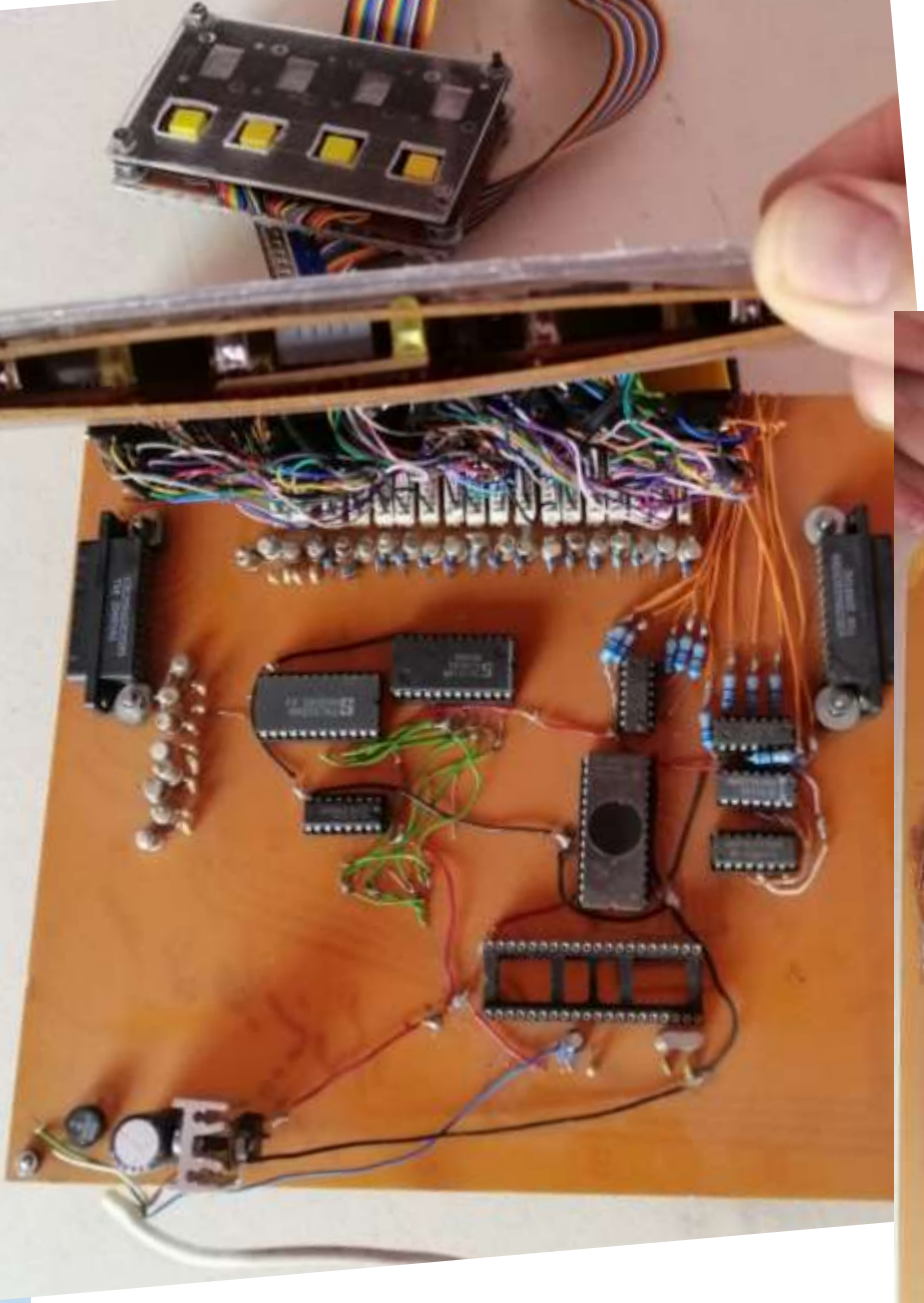
طراحی مدار

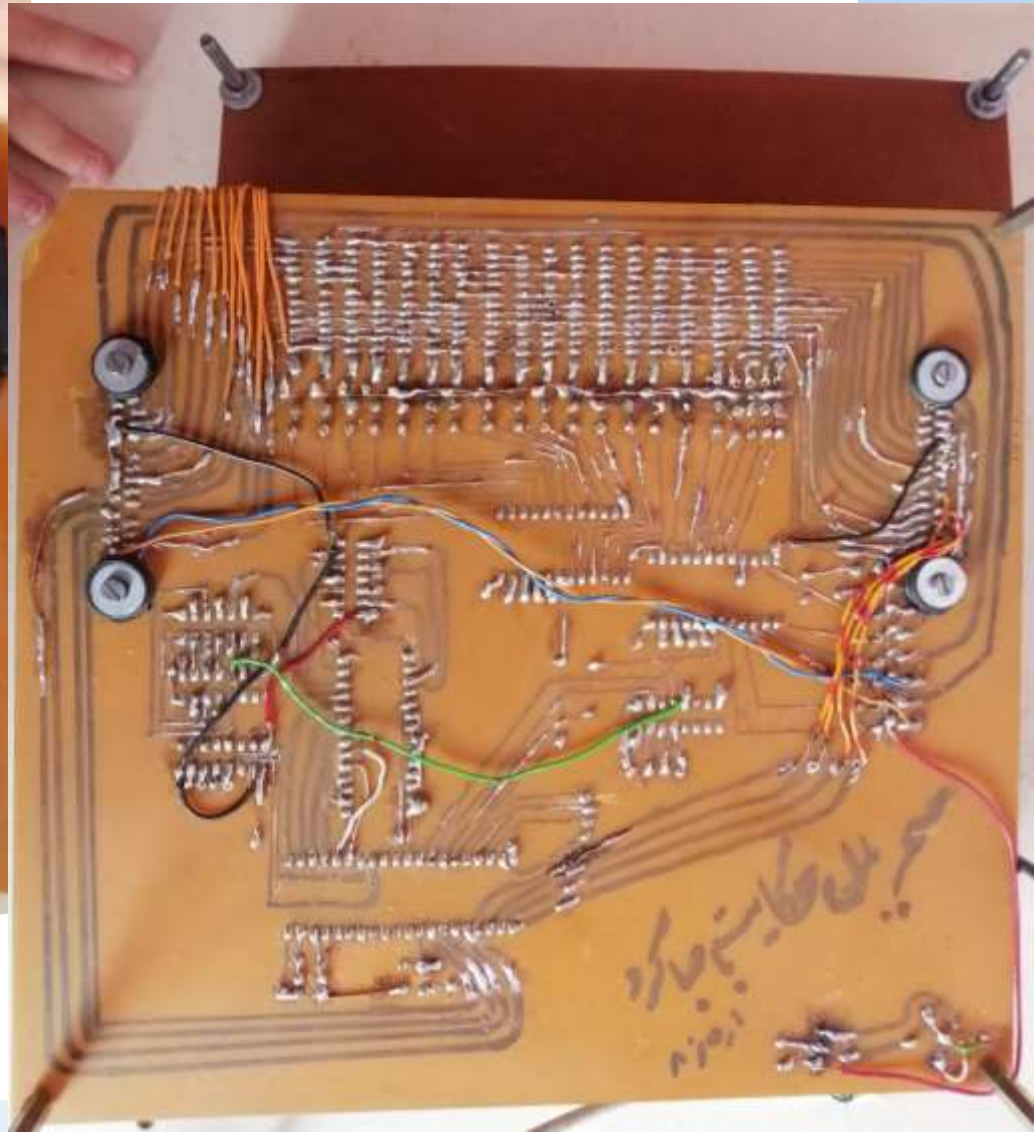
- طراحی مدار به کاری گفته می‌شود که با دانستن توقعی که از مدار داریم، آن را به صورت گیت‌های منطقی رسم می‌کنیم. برای این کار:

1. ورودی و خروجی‌های مدار را شناسایی و نام‌گذاری می‌کنیم (مثل راه‌انداز سون‌سگمنت که جلسه قبل با آن آشنا شدیم چهار ورودی و هفت خروجی دارد).
2. جدول درستی را رسم می‌کنیم.
3. با روش‌هایی مانند جدول کارنو فورمول ساده شده خروجی‌ها را پیدا می‌کنیم.
4. مدار را رسم می‌کنیم.
5. با انجام تحلیل از درستی مدار مطمئن می‌شویم.

طراحی مدارهای منطقی (سیستم‌های دیجیتال) بصورت دستی

- آنچه در این درس می‌آموزیم طراحی مدارهای دیجیتال بصورت دستی و بر روی کاغذ است.
- طراحی کاغذی شامل پروسه طراحی شماتیک مدار، تحلیل دستی برای اطمینان از درست بودن طراحی و طراحی بردی که ICها و قطعات روی آن قرار می‌گیرند می‌شود.
- در دو اسلاید بعد یک سیستم دیجیتال را می‌بینید که تمام مراحل ساخت آن بصورت دستی انجام گرفته است.





طراحی مدارهای منطقی (سیستم‌های دیجیتال) به کمک کامپیوتر

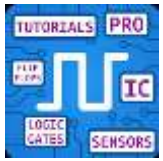
- طراحی به کمک کامپیوتر کار طراح را ساده می‌کند، البته طراح باید قبلا طراحی دستی را درک کرده باشد.
- برای انجام طراحی به صورت کامپیوتری باید طرح مورد نظر را به یکی از زبان‌های HDL (زبان توصیف سخت‌افزار) نوشت.
- این نوشتار یک «کد کامپیوتری» (کد توصیفی) هست؛ اما یک «برنامه کامپیوتری» (مثل برنامه زبان C) نیست. در اینجا فقط مدار توصیف می‌شود اما اجراهای الگوریتمی مد نظر نیست.

طراحی مدارهای منطقی (سیستم‌های دیجیتال) به کمک کامپیوتر

- معروفترین زبان‌های توصیف سخت افزار Verilog و VHDL هستند که در درس «طراحی کامپیوتری سیستم‌های دیجیتال» با یکی از آنها آن آشنا می‌شوید.
- نرم‌افزارهایی که با این کد کار می‌کنند میتوانند خدمات مختلفی ارائه دهند مثل: تحلیل و شبیه‌سازی مدار (تست مدار قبل از پیاده‌سازی واقعی)، ساده‌سازی طراحی و طراحی برد مدار چاپی. این نرم‌افزارها روی کامپیوتر PC کار می‌کنند و بسیار قوی و حرفه‌ای هستند ، از جمله ModelSim و Proteus

طراحی مدارهای ساده با اپ‌های موبایل

- کار با نرم‌افزارهای حرفه‌ای طراحی مدار نیازمند تجربه و تمرین است، اما اپلیکیشن‌هایی برای گوشی و تبلتهای اندروید وجود دارند که کارکردن با آنها ساده و لذت‌بخش است.
- شما می‌توانید بخش رسم مدار تمرین‌ها را بعد از این با این برنامه‌ها انجام دهید:



- Smart Logic Simulator

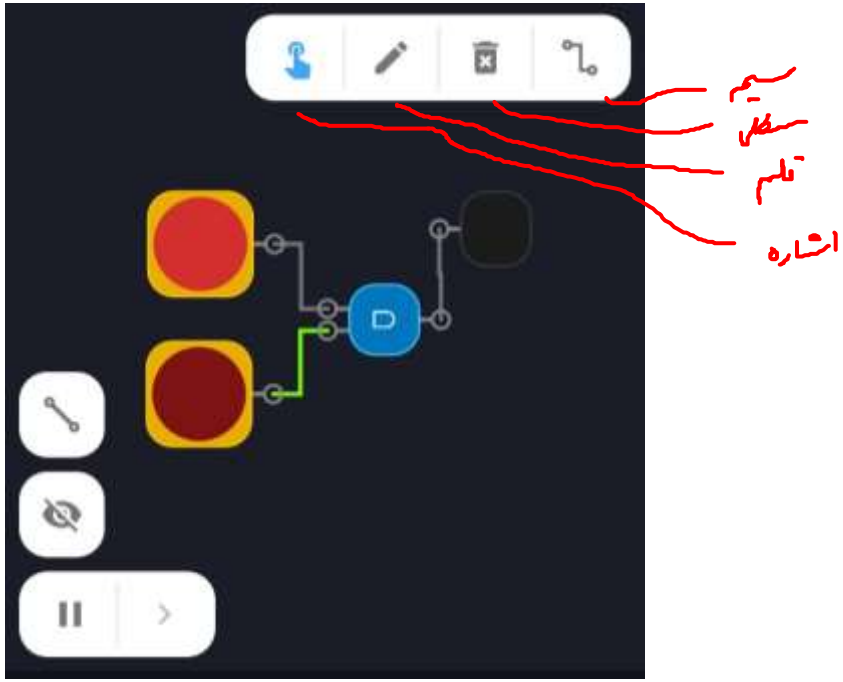
- Physics Lab

Smart Logic Simulator

- یکی از جالبترین و راحت‌ترین و با امکانات‌ترین ابزارهای طراحی مدار است.
- گیت‌های پایه، مالتی‌پلکسر، لچ و انواع فلیپ فلاپ و دیکدر سون سگمنت قابل دسترسی است.
- ورودی و خروجی‌ها تنوع بالایی دارند و از امکانات ویژه گوشی (که در کامپیوتر نیست) حداکثر استفاده شده است.
- ورودی‌های پر کاربرد عبارتند از: 0 و1 ثابت، دگمه فشاری (همزمان می‌شود با چند انگشت چند دگمه را فشار داد) و زدنی، نوسان‌ساز با فرکانس‌های مختلف، سنسور حرکت، قطب‌نما و نورسنج گوشی.
- خروجی‌های پرکاربرد عبارتند از لامپ، سون‌سگمنت، چراغ راهنمایی، ماتریس، ویبره گوشی.

Smart Logic Simulator

- وارد قسمت Your Circuits و CREATE NEW را و بعد Project برنید.
- با زدن قلم وارد محیط ادیت می‌شوید. با + قطعه اضافه کنید و سپس آنها را مرتب بچینید. با لمس دو انگشتی می‌توانید زوم صفحه را تغییر دهید.
- با زدن علامت سیم می‌توانید بین قطعات سیم‌کشی کنید. با زدن سطل آشغال هم می‌توانید قطعات و سیم‌ها را حذف کنید.
- با زدن علامت اشاره انگشت (چپ‌ترین) به حالت اجرا بروید.

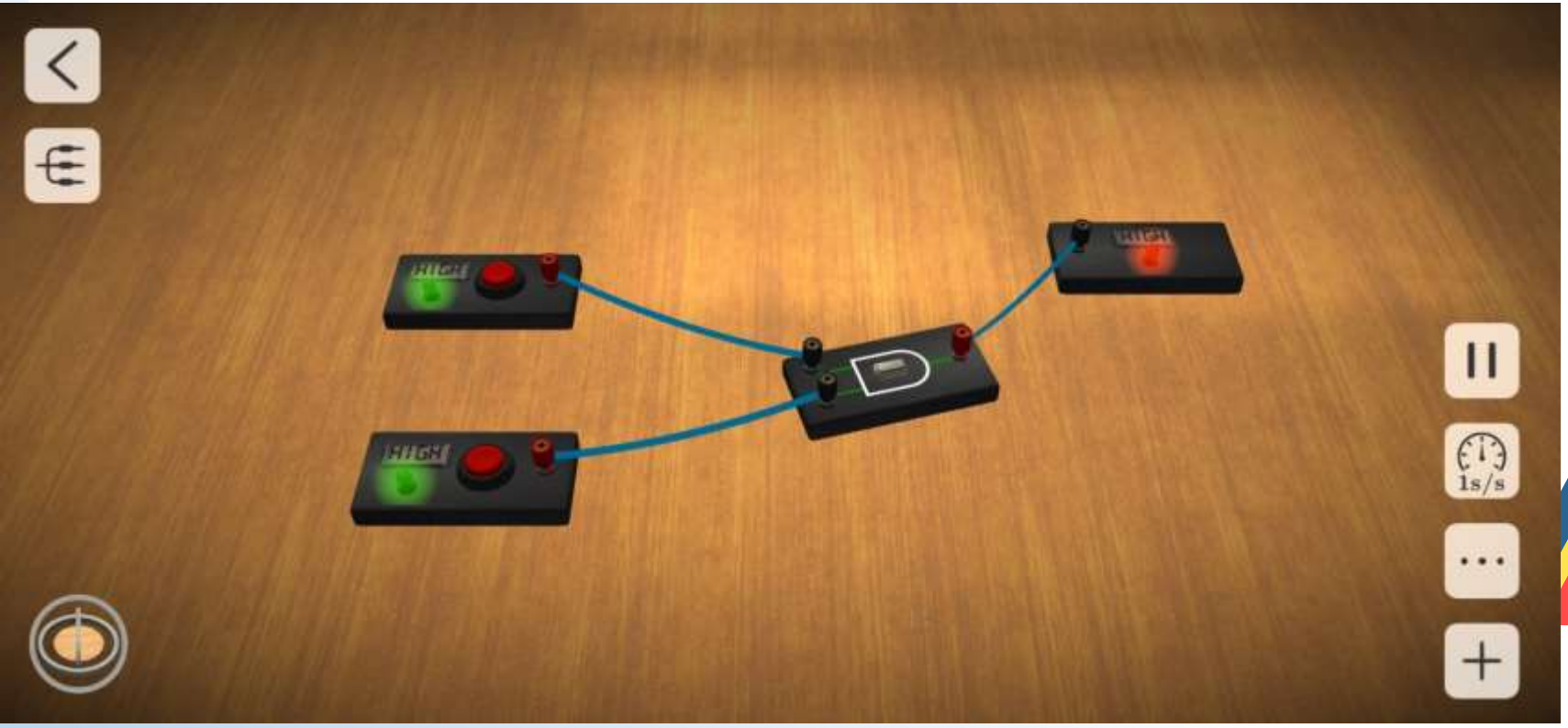


Physics Lab

- به غیر از طراحی مدارهای منطقی، مدارهای ساده برق را می‌توانید در این اپ طراحی کنید. همچنین بخش‌های دیگری برای باز طراحی منظومه شمسی و ذرات ریزاتمی وجود دارد.
- گرافیک آن بسیار جالب است و روی یک میز کار چوبی قطعات را می‌چینید و با سیم به هم وصل می‌کنید.
- گیت‌های ساده منطقی، سویچ ورودی و چراغ خروجی در آن وجود دارد. همچنین نیم و تمام جمع کننده.
- قطعات دیگری مثل مالتی‌پلکسر و انواع فلیپ فلاپ در آن با سیستم امتیازی یا پولی قابل دسترسی است.

Physics Lab

- برای شروع، وارد circuit sandbox شوید، + را بزنید و بعد قسمت قطعات digital را انتخاب و هر چه می‌خواهید را روی میز قرار دهید و بعد سیم‌کشی کنید.



طراحی مدار

- بحث جلسات قبل تماما در مورد انجام پروسه طراحی بود.
- در ادامه با طراحی چند مدار کاربردی در کامپیوتر و سیستم‌های دیجیتال، بحث طراحی را ادامه می‌دهیم.

تمرین

- می‌خواهیم یک سیستم دیجیتال برای رای‌گیری در هیات مدیره یک شرکت طراحی نماییم. در جلسات رییس و دو عضو هیات مدیره حضور دارند و با فشردن یک دگمه فشاری رای می‌دهند و روشن شدن یک چراغ تصویب شدن مورد رای‌گیری را نشان می‌دهد. رای رئیس دو تا حساب می‌شود. مصوبه فقط در صورتی تصویب می‌شود که تعداد آرا زوج باشد (۰،۲،۴ رای). دگمه زیر دست رییس ورودی x و دو دگمه دیگر ورودی‌های y و z را تشکیل می‌دهند و خروجی سیستم که f نام دارد لامپ وصل است
- برای طراحی مدار ابتدا جدول درستی را رسم کنید و بعد از روی آن جدول فورمول fرا به صورت کانونی بدست آورید و با استفاده از آن مدار را طراحی کنید. سپس از روی جدول درستی، جدول کارنو رسم کنید و با استفاده از آن یک فورمول ساده شده استاندارد بدست آورید و بار دیگر مدار را رسم کنید. دو مداری که رسم کرده‌اید را از نظر تعداد طبقه و تعداد گیت مقایسه کنید.

مدارهای منطقی

(طراحی سیستم‌های دیجیتال)

جلسه ششم

مدارات ترکیبی پرکاربرد

مباحث امروز

- جمع و تفریق کننده
- دیکدر و انکدر
- مالتی‌پلکسر

مدار نیم جمع‌کننده

- این مدار قرار است دو عدد یک بیتی را با هم جمع کند؛

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 (با نقلی ۱)

مدار نیم جمع‌کننده

- این مدار قرار است دو عدد یک بیتی را با هم جمع کند؛ بنابراین مدار دارای دو ورودی است و نیز دو خروجی (حاصل جمع و نقلی احتمالی).

ورودی‌ها را x و y و حاصل‌جمع را S و نقلی را C نام‌گذاری می‌کنیم:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 (با نقلی ۱)

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

مدار نیم جمع‌کننده

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

- فورمول خروجی‌ها ساده است و بدون جدول کارنو می‌توان آن را نوشت:

C =

S =

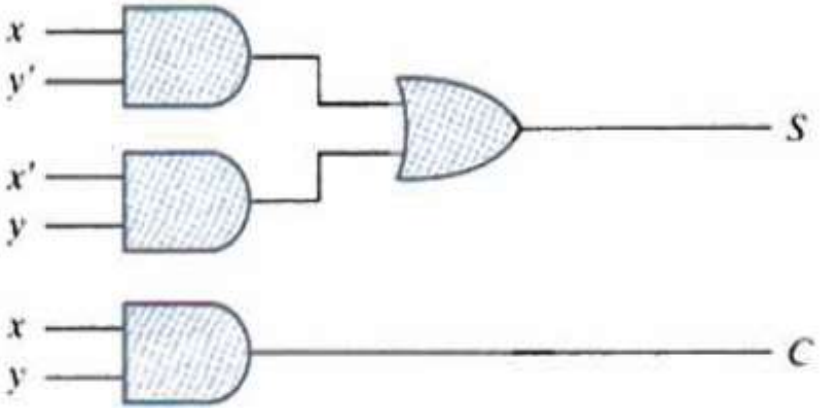
مدار نیم جمع‌کننده

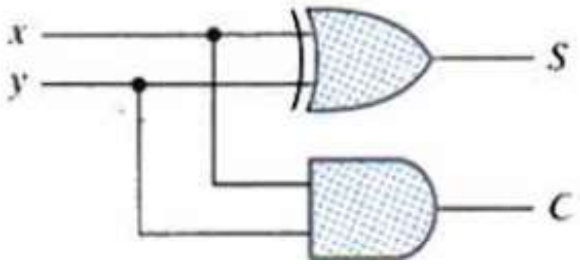
x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

- فورمول خروجی‌ها ساده است و بدون جدول کارنو می‌توان آن را نوشت:

$C = xy$

$S = x'y + xy' = x \oplus y$





مدار جمع‌کننده کامل

مدار جمع‌کننده کامل

- زمانی که دو عدد باینری چند رقمی را با هم جمع می کنیم هر یک از ارقام (غیر از کم‌ارزشترین رقم) غیر از دو رقم، یک رقم نقلی (کری) را هم در ورودی دارند. هدف از طراحی این مدار جمع کردن هم دو بیت با یکدیگر است و مدار دارای سه ورودی است و دو خروجی.

x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

مدار جمع‌کننده کامل

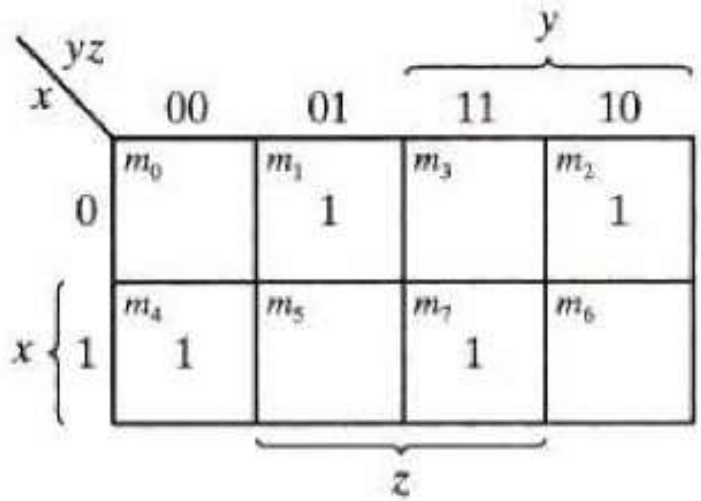
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

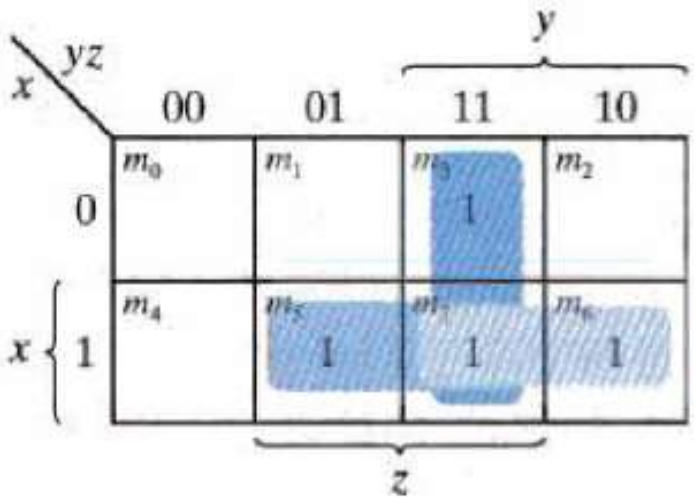
S =

C=

مدار جمع‌کننده کامل

x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1





$S = x'y'z + x'yz' + xy'z' + xyz$

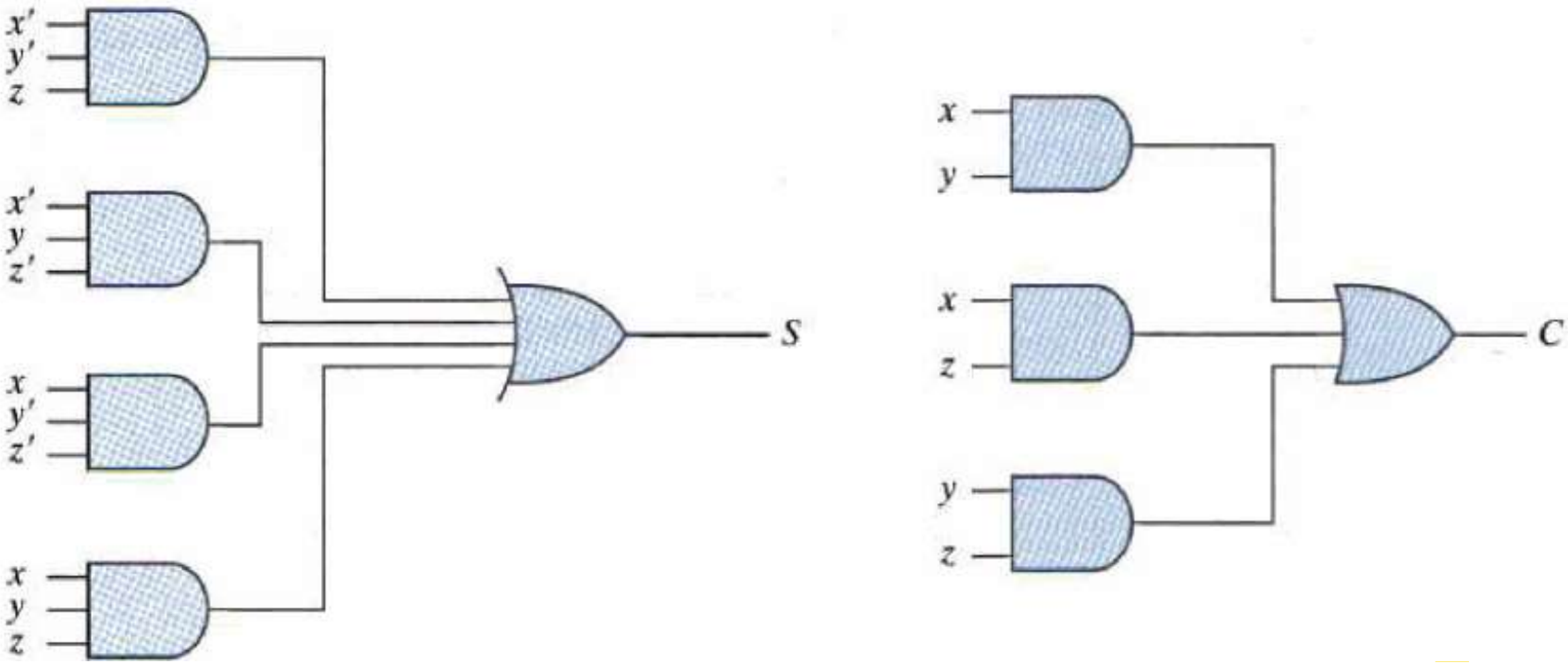
$C = xy + xz + yz$

مدار جمع‌کننده کامل

$S = x'y'z + x'yz' + xy'z' + xyz$

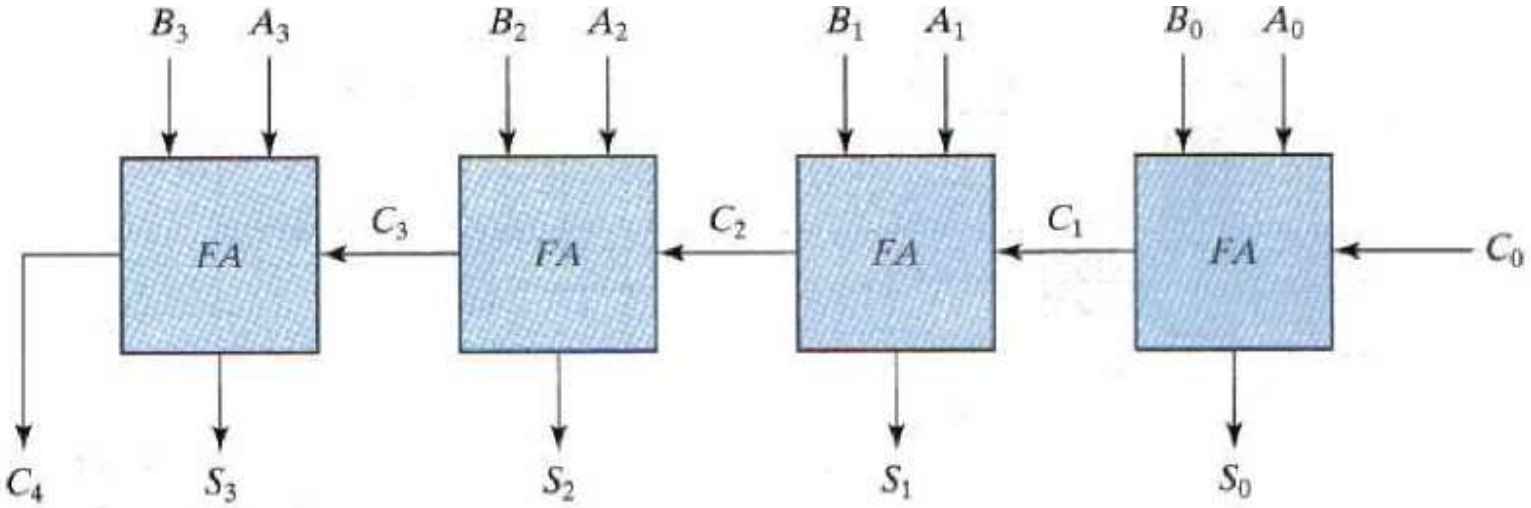
$C = xy + xz + yz$

پیاده‌سازی دوطبقه به فرم استاندارد با ۹ گیت



جمع‌کننده اعداد باینری (چند رقمی)

- با داشتن چند جمع کننده کامل و اتصال آنها به صورت سری (برای انتقال نقلی) می‌توان اعداد چند رقمی باینری را با هم جمع کرد (**گسترش جمع‌کننده**).

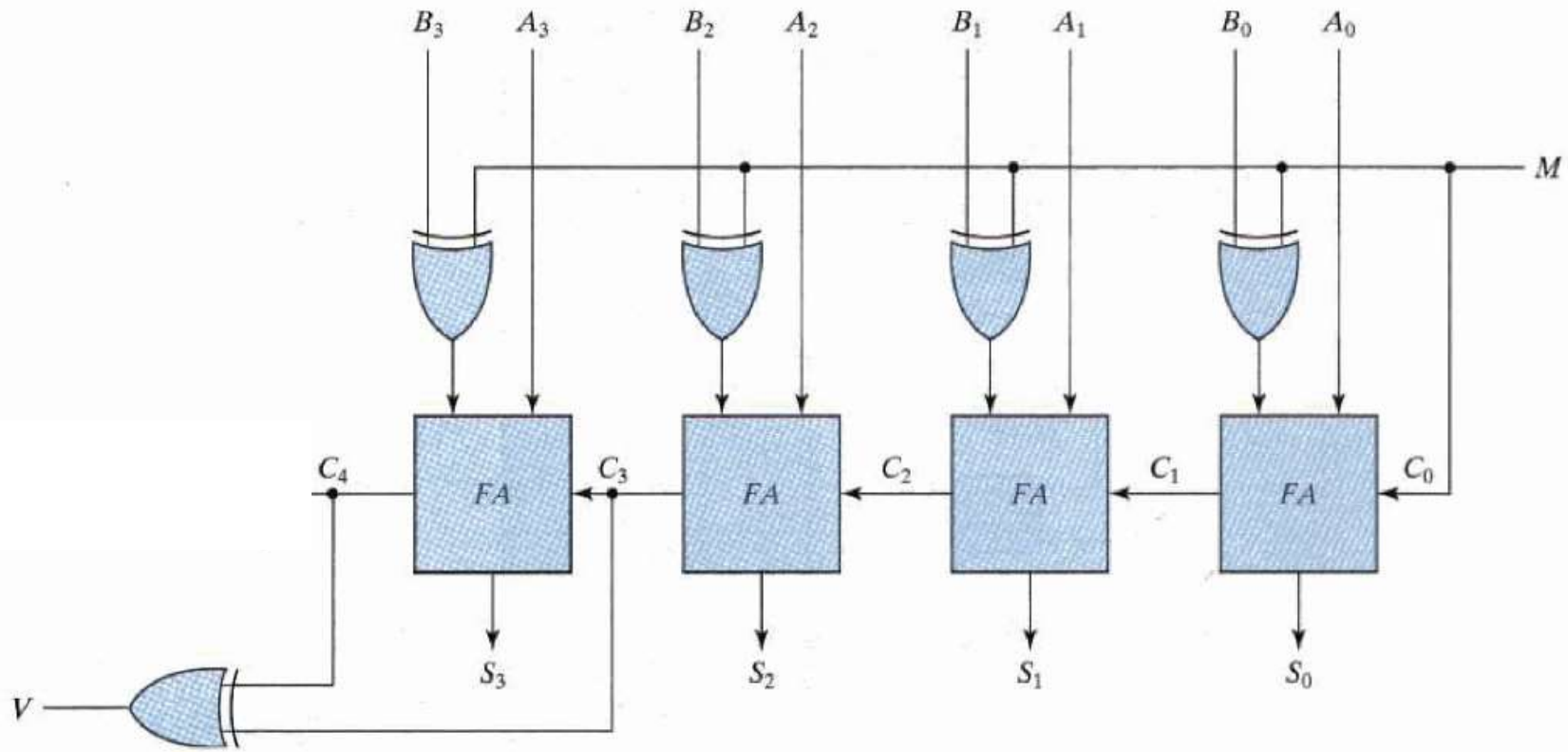


- عامل محدود کننده در اینجا تاخیر است. برای اطمینان از انتقال کامل نقلی و درست بودن جواب باید به اندازه تاخیر جمع کننده ضربدر تعداد طبقات صبر کنیم. البته می‌توان جمع کننده‌های چند بیتی با پیچیدگی ساخت.

تفریق

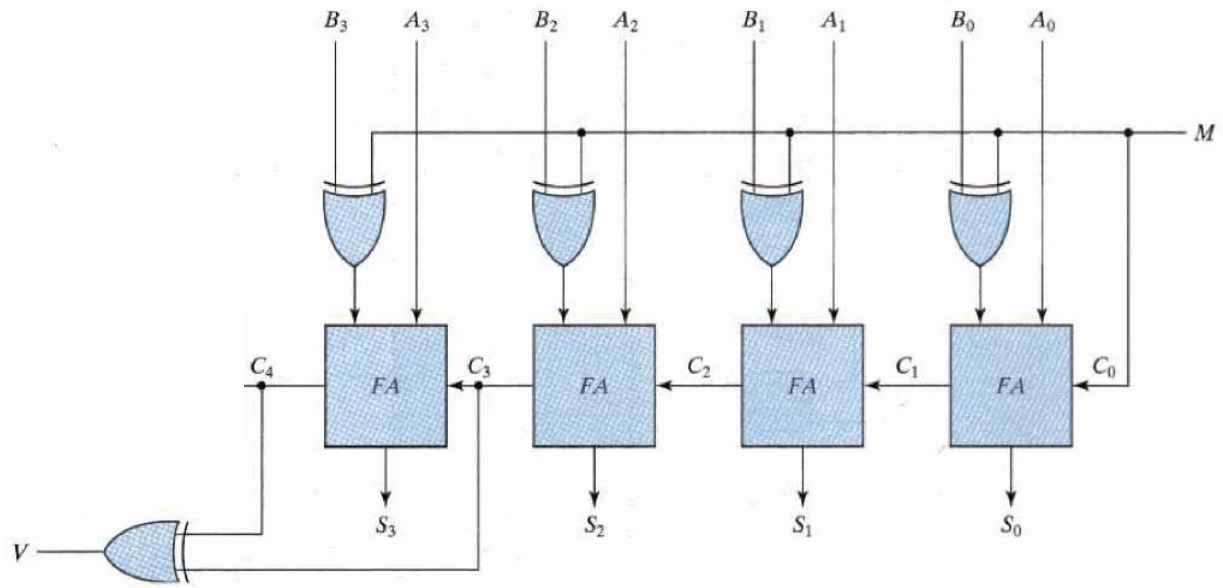
- قبلا دیدیم که برای تفریق می‌توان از روش مکمل ۲ استفاده کرد.
- در این روش تمام بیت‌ها را معکوس نموده و با ۱ جمع می‌کنیم؛ که برای جمع کردن می‌توان نقلی به اولین جمع‌کننده وارد نمود.
- با یک روش ابتکاری مداری ساخته می‌شود که هم جمع کننده است و هم تفریق کننده.
- XOR با صفر و یک؟؟

جمع و تفریق کننده



- M اگر یک باشد مدار تفریق وگرنه جمع می‌کند.
- XOR اگر یک طرفش 0 باشد طرف دیگر را مکمل می‌کند و اگر یک طرفش 1 باشد طرف دیگر عینا منتقل می‌شود.

جمع و تفریق کننده



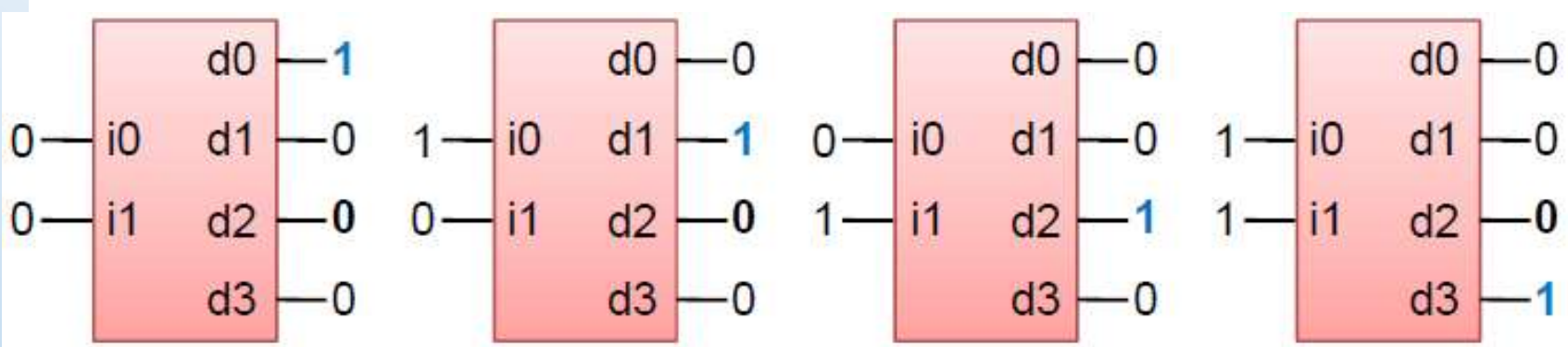
- در جمع اعداد بدون علامت، خروج کری از پرارزشترین بیت نشانه سرریز است.
- در جمع اعداد علامت‌دار از نوع مکمل ۲ اگر کری وارد بیت آخر شود باید از آن خارج شود و اگر وارد نشود نباید خارج شود وگرنه سرریز است. این مورد با یک XOR دیگر برای و ایجاد خروجی V پیاده‌سازی شده است.

دیکدر

- وقتی وارد یک بیمارستان می‌شوید به دربان می‌گویید کجا می‌خواهید بروید (مثلا رادیولوژی).
- دربان با توجه به اطلاعی که از شما گرفته یکی از خطوط روی زمین را نشان می‌دهد و می‌گوید آن را دنبال کنید.

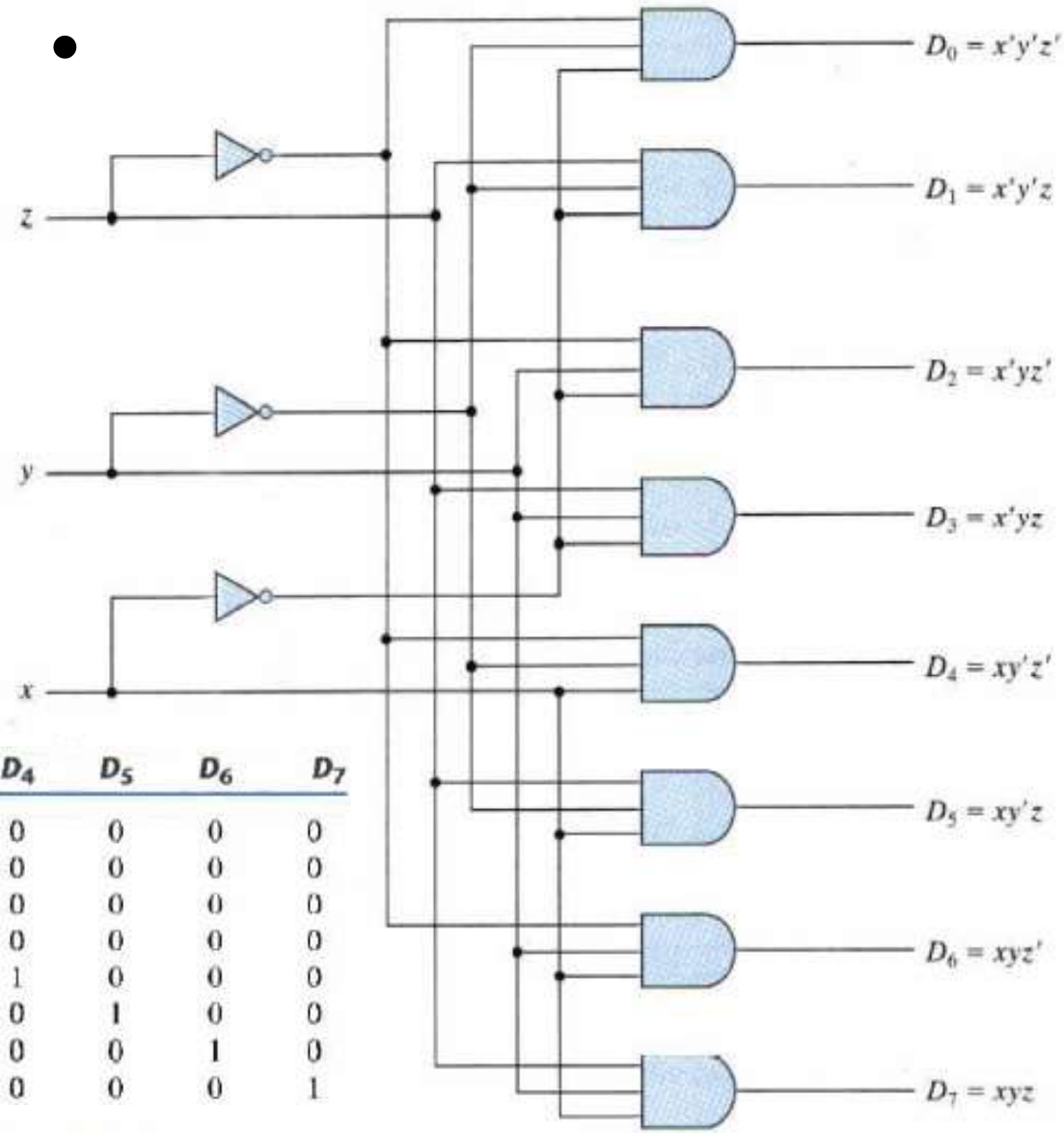
دیکدر

- مداری است دارای n ورودی و (حداکثر) 2^n خروجی
- این مدار بر اساس آنچه در ورودی هست، یکی از خطوط خروجی را فعال می‌کند.
- مثال زیر دیکدری است با ۲ ورودی و ۴ خروجی:



طراحی داخلی دیکدر

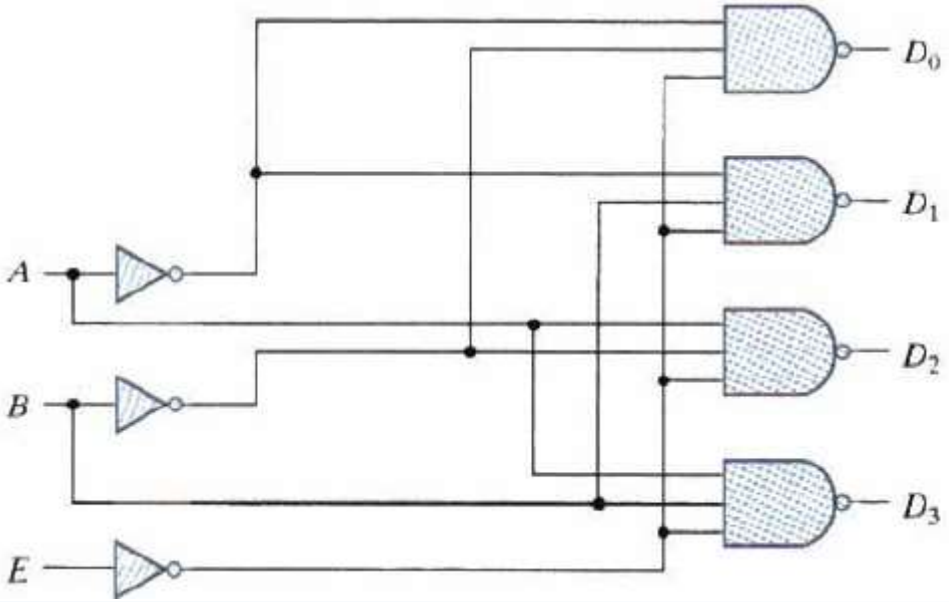
- همانطور که می‌بینید دیکدر مداری است یک طبقه که تمام مینترم‌ها را پوشش می‌دهد (یعنی برای هر سطر جدول یک خروجی دارد).



x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

ورودی فعال‌ساز، خروجی معکوس

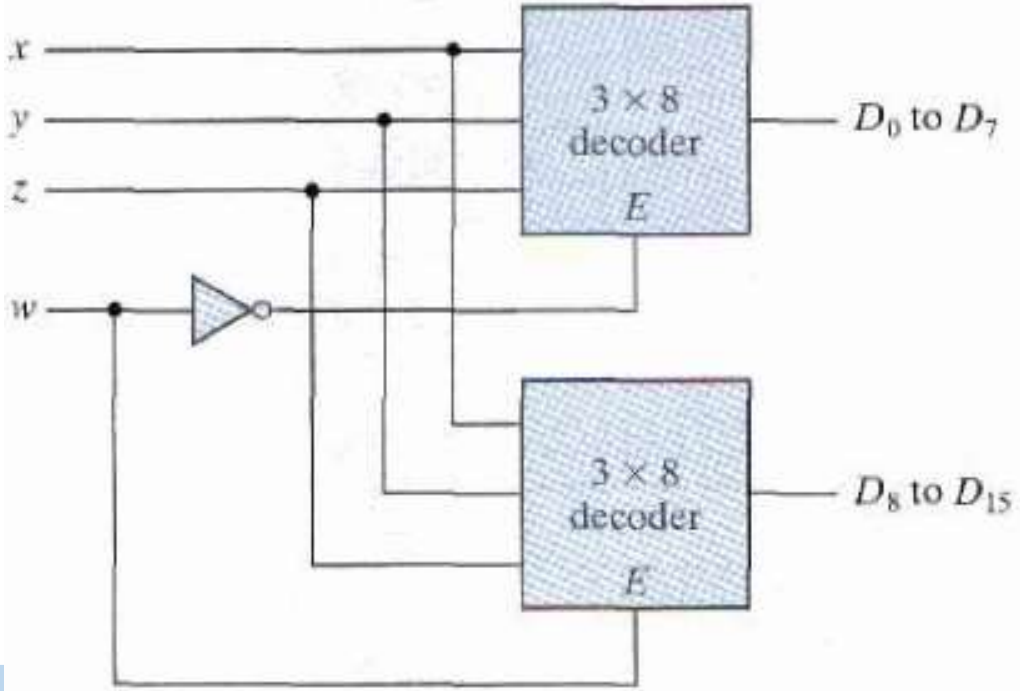
- می‌توان در دیکدر یک ورودی اضافی داشت بنام فعال‌ساز (E=enable) که اگر فعال‌ساز 0 باشد هیچکدام از خطوط خروجی 1 نخواهند شد.
- دیکدر را می‌توان بجای and با nand ساخت که در این صورت خروجی معکوس (خطوط غیر فعال 1 و خط فعال 0) است.



E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

گسترش دیکدر

- همانطور که با چند جمع‌کننده کوچک یک جمع‌کننده بزرگ ساختیم، می‌توانیم با چند دیکدر کوچک دیکدری بزرگ بسازیم (به این کار گسترش دیکدر می‌گویند).
- در مثال زیر با داشتن دو دیکدر ۳ به ۸ دارای فعال‌ساز، یک دیکدر ۱۶ خطی ساخته شده است.



ساخت مدارهای ترکیبی با دیکدر

- دیکدر تمامی مینترم‌های ممکن را تولید می‌کند، بنابراین با or کردن برخی خروجی‌های دیکدر می‌توان انواع توابع منطقی را ساخت.
- از آنجا که خود دیکدر مداری یک طبقه است، مدار ساخته شده به این روش دارای دو طبقه تاخیر بوده که معادل مدارهای استاندارد است.
- این روش برای مدارهای ترکیبی با خروجی‌های متعدد به صرفه است.

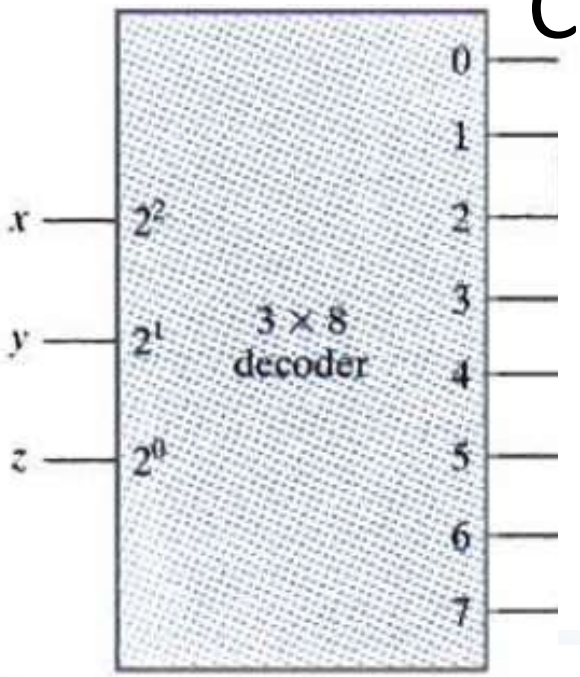
ساخت جمع‌کننده با دیکدر

x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

- مثال: تمام جمع‌کننده (با سه ورودی و دو خروجی) را با دیکدر طراحی کنید.

(یک یا سه بیت ۱) $S(x,y,z) = \sum(\ ,\ ,\ ,\ )$

(دو یا سه بیت ۱) $C(x,y,z) = \sum(\ ,\ ,\ ,\ )$



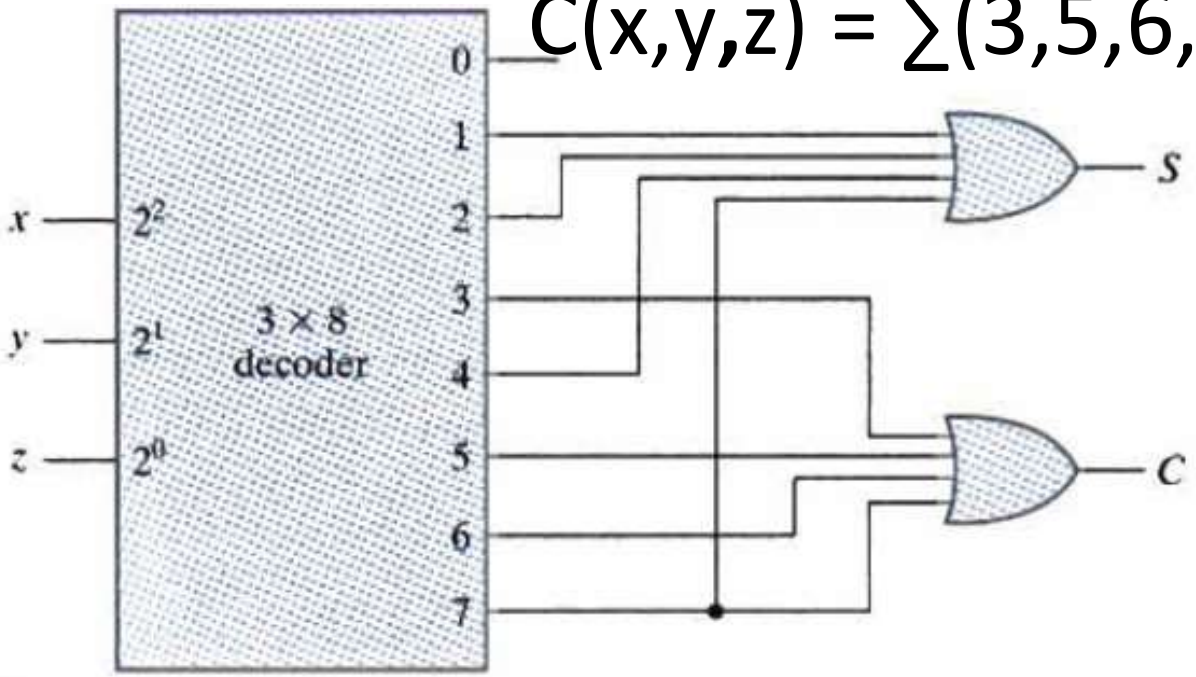
ساخت جمع‌کننده با دیکدر

x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

- مثال: تمام جمع‌کننده (با سه ورودی و دو خروجی) را با دیکدر طراحی کنید.

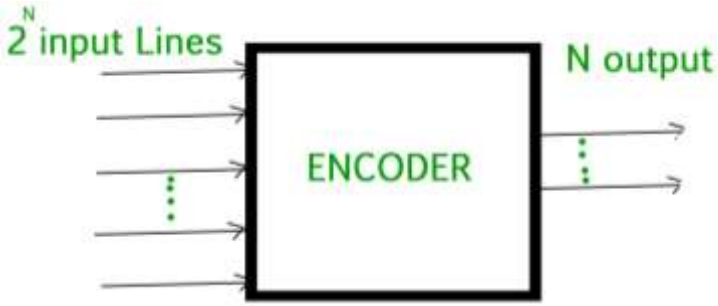
(یک یا سه بیت ۱) $S(x,y,z) = \sum(1,2,4,7)$

(دو یا سه بیت ۱) $C(x,y,z) = \sum(3,5,6,7)$



انکدر

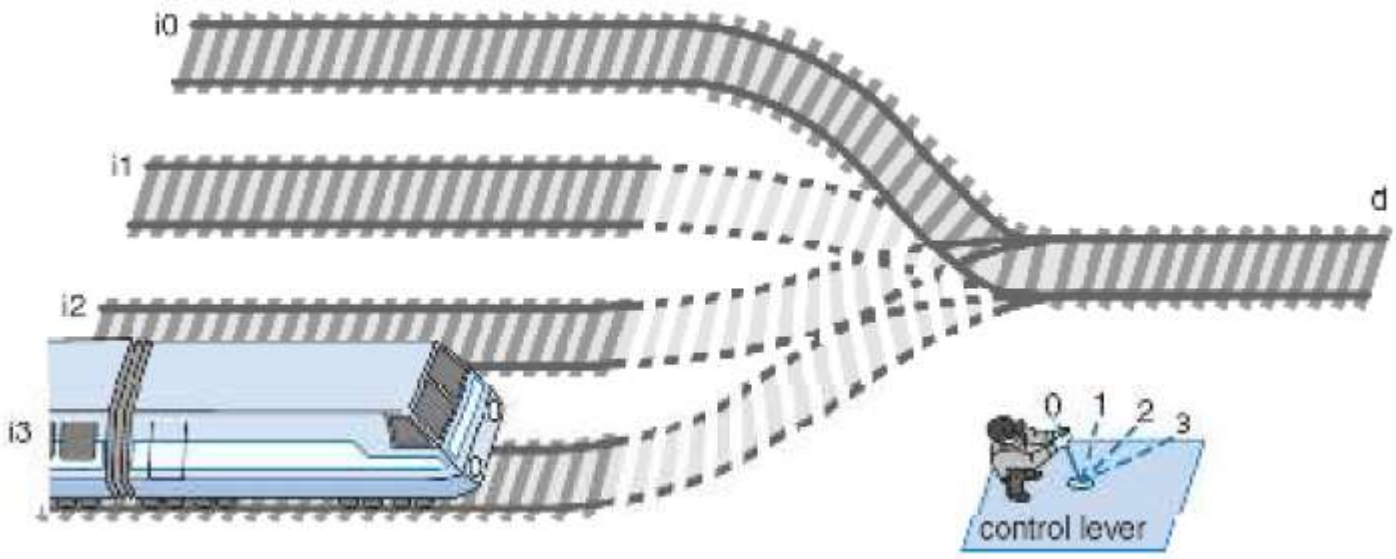
- انکدر مداری است که برعکس دیکدر عمل می‌کند؛ یعنی 2^n ورودی دارد و n خروجی. هر زمان فقط یک ورودی مقدار 1 دارد و بر آن اساس مقدار خروجی شکل می گیرد. مثلا اگر خط ورودی D2 فعال باشد، 010 روی خطوط خروجی خواهد بود.
- انکدرها معمولا اولویت لاین بالا دارند، یعنی اگر (به اشتباه) هم خط D2 و هم D4 فعال باشند، فقط D4 (شماره بزرگتر) مورد توجه قرار می‌گیرد.



Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

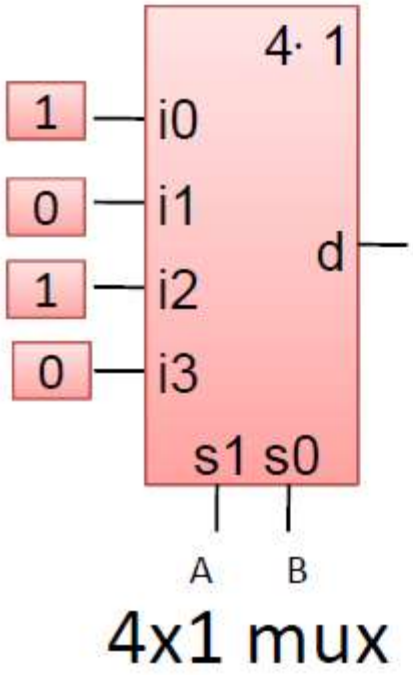
مالتی‌پلکسر

- در مسیر راه‌آهن در شکل زیر تعدادی خط قطار وارد می‌شوند اما سوزن‌بان با توجه به کدی که دارد فقط به یک قطار اجازه عبور می‌دهد.



مالتی‌پلکسر

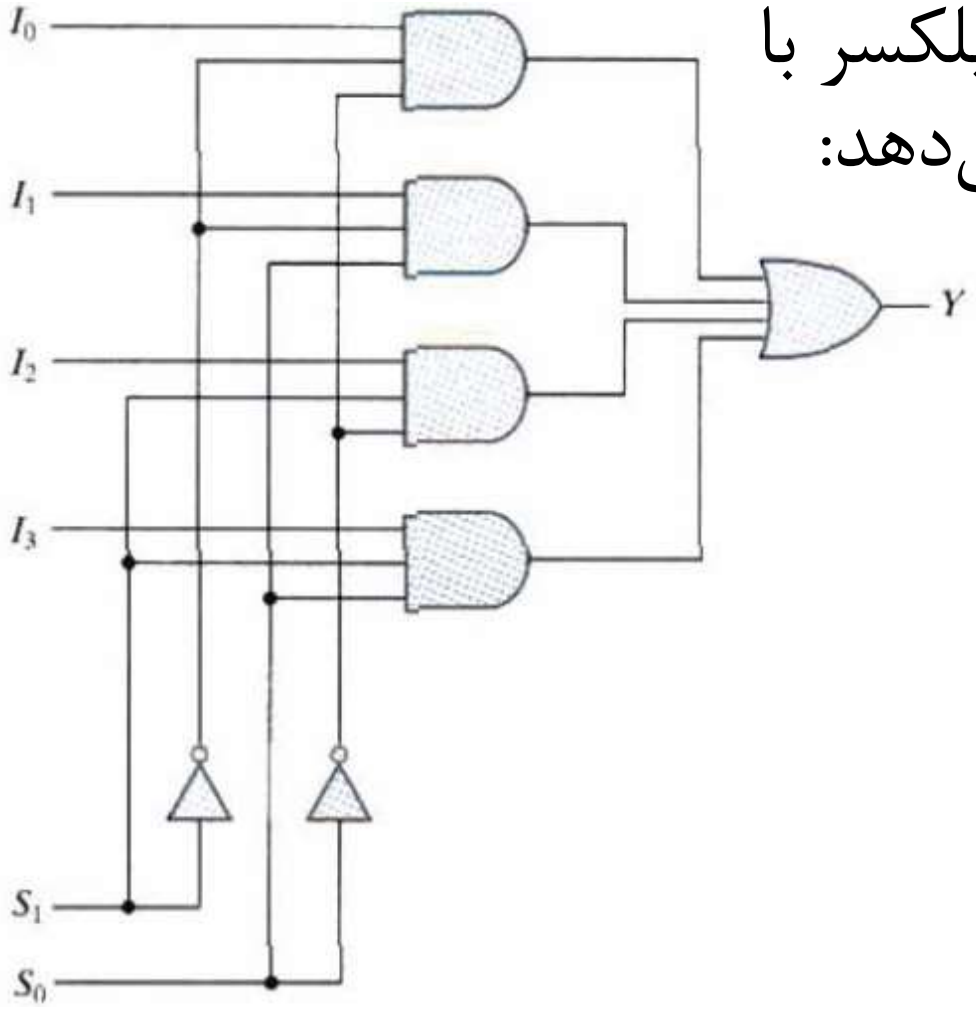
- مداری است دارای n ورودی کنترلی، 2^n ورودی دیتا و تنها یک خروجی.
- این مدار بر اساس آنچه در ورودی هست، یکی از خطوط ورودی را در خروجی شبیه‌سازی می‌کند.
- مثال زیر مالتی‌پلکسری است با n=2



F (A, B) = ∑m(0,2)

طراحی داخلی مالتی‌پلکسر

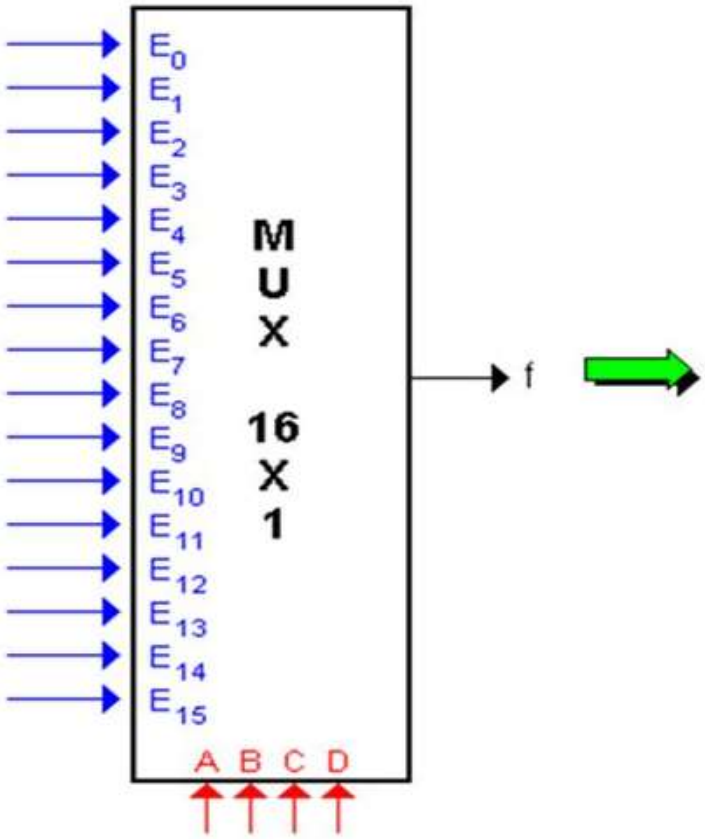
- مالتی‌پلکسر از نظر طراحی داخلی دو طبقه است.
- شکل یک مالتی‌پلکسر با n=2 را نشان می‌دهد:



S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

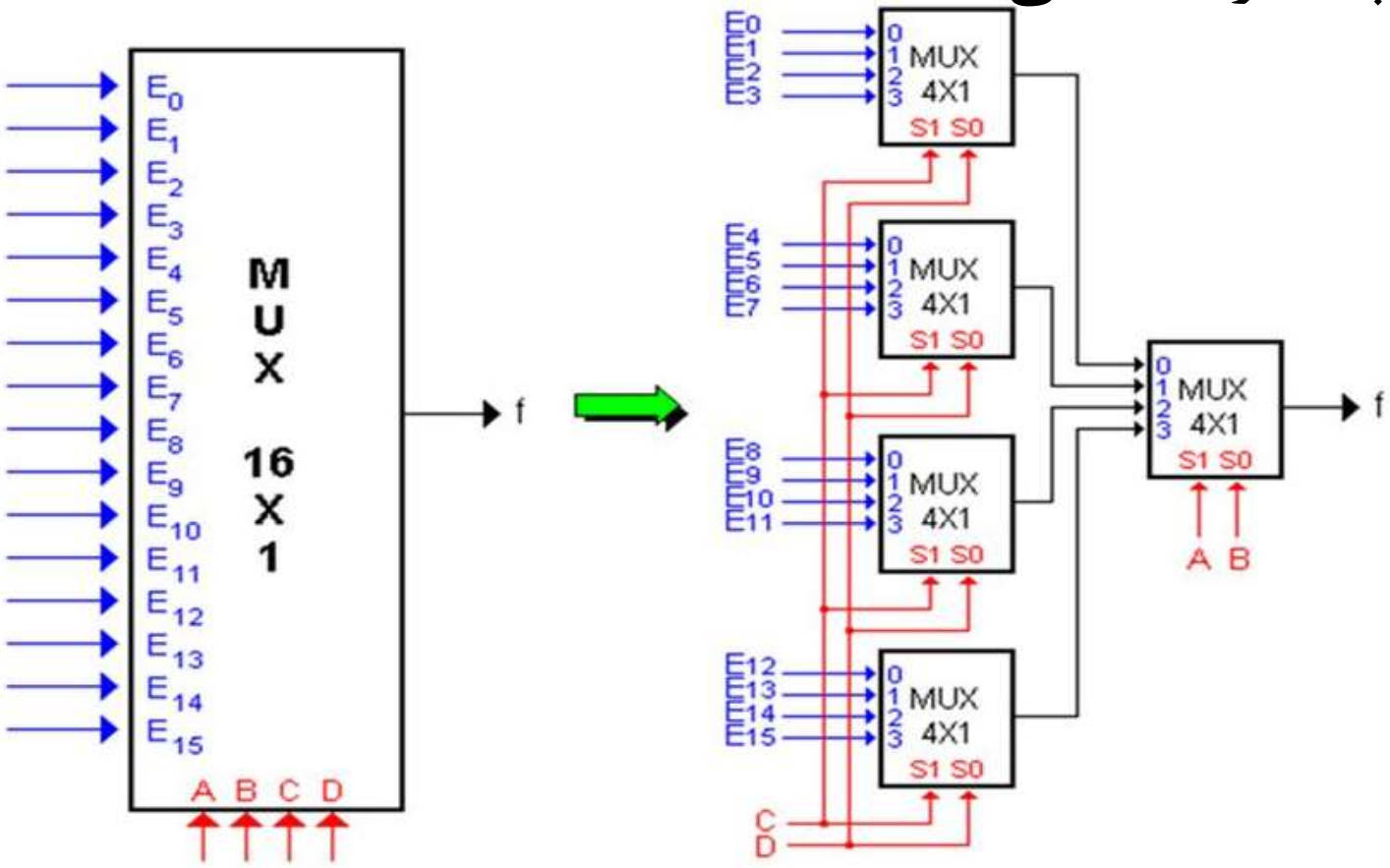
گسترش مالتی‌پلکسر

با اضافه کردن یک مالتی‌پلکسر به چند مالتی‌پلکسر کوچک کی‌توان مالتی‌پلکسر بزرگتری ساخت. همانطور که در مثال زیر با داشتن چهار مالتی‌پلکسر ۲ خط آدرسی دارای فعال‌ساز، یک مالتی‌پلکسر ۴ خطی ساخته شده است.



گسترش مالتی‌پلکسر

با اضافه کردن یک مالتی‌پلکسر به چند مالتی‌پلکسر کوچک می‌توان مالتی‌پلکسر بزرگتری ساخت. همانطور که در مثال زیر با داشتن چهار مالتی‌پلکسر ۲ خط آدرسی دارای فعال‌ساز، یک مالتی‌پلکسر ۴ خطی ساخته شده است.

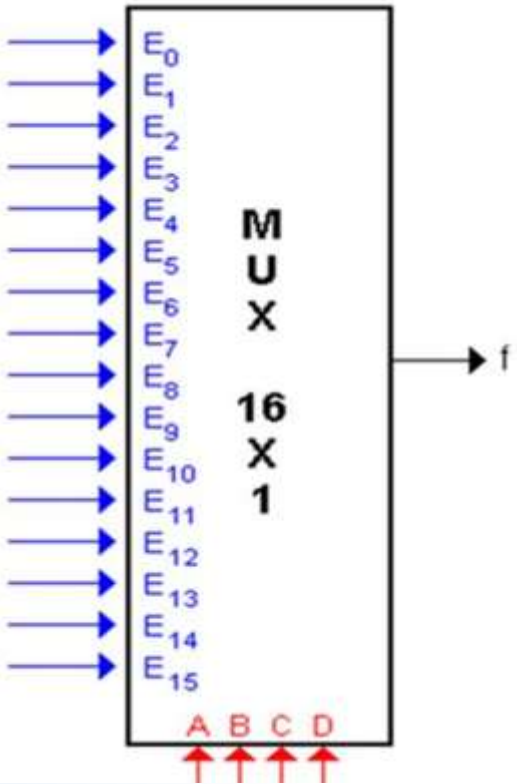


ساخت مدارهای ترکیبی با مالتی‌پلکسر

- میتوان متغیرهای مدار را به خطوط کنترلی مالتی‌پلکسر وصل کرد و بعد با 0 و 1 قرار دادن ورودی‌ها جدول درستی را عینا پیاده‌سازی کرد. یعنی برای مثال زیر یک ماکس چهارخطه استفاده کنیم.

مثال:

$F(A,B,C,D) = \sum(1,3,4,11,12,13,14,15)$



ساخت مدارهای ترکیبی با مالتی‌پلکسر

- میتوان متغیرهای مدار را به خطوط کنترلی مالتی‌پلکسر وصل کرد و بعد با 0 و 1 قرار دادن ورودی‌ها جدول درستی را عینا پیاده‌سازی کرد. یعنی برای مثال زیر یک ماکس چهارخطه استفاده کنیم.

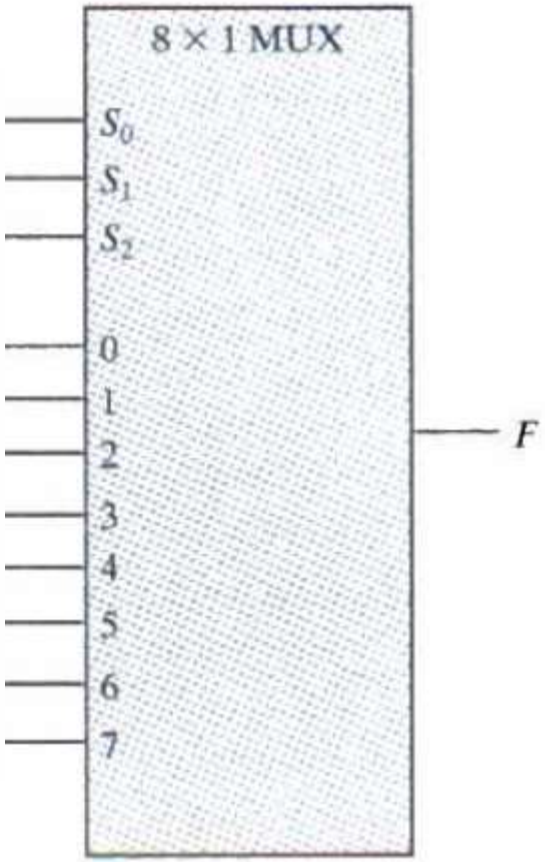
مثال: $F(A,B,C,D) = \sum(1,3,4,11,12,13,14,15)$

- **روش فوق** گرچه ممکن است اما **بهینه نیست**. بهتر است برای n متغیر از یک مالتی‌پلکسر با n-1 خط کنترلی استفاده کنیم و ورودی‌ها را 0 و 1 و D و D' قرار دهیم (D متغیری است که به ورودی‌ها نداده‌ایم).

مثالی از ساخت مدارهای ترکیبی با مالتی‌پلکسر

مثال: $F(A,B,C,D) = \sum(1,3,4,11,12,13,14,15)$

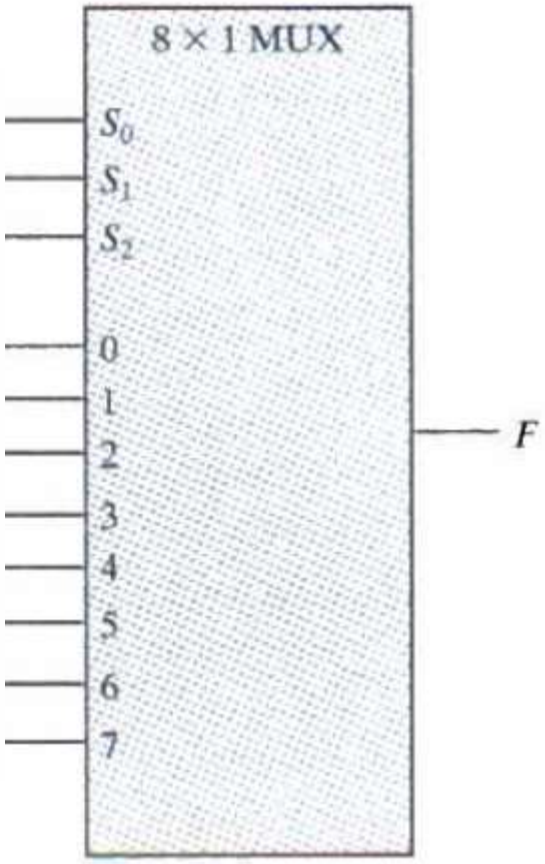
A	B	C	D	F	
0	0	0	0	0	F =
0	0	0	1	1	
0	0	1	0	0	F =
0	0	1	1	1	
0	1	0	0	1	F =
0	1	0	1	0	
0	1	1	0	0	F =
0	1	1	1	0	
1	0	0	0	0	F =
1	0	0	1	0	
1	0	1	0	0	F =
1	0	1	1	1	
1	1	0	0	1	F =
1	1	0	1	1	
1	1	1	0	1	F =
1	1	1	1	1	



مثالی از ساخت مدارهای ترکیبی با مالتی‌پلکسر

مثال: $F(A,B,C,D) = \sum(1,3,4,11,12,13,14,15)$

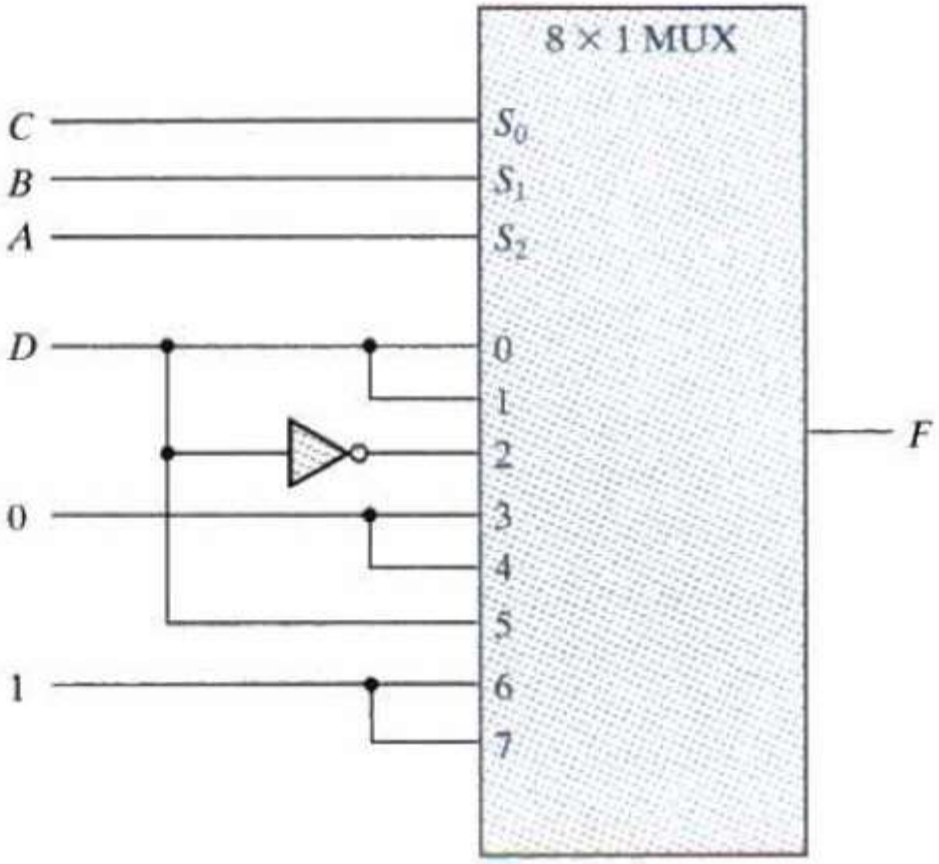
A	B	C	D	F	
0	0	0	0	0	$F = D$
0	0	0	1	1	
0	0	1	0	0	$F = D$
0	0	1	1	1	
0	1	0	0	1	$F = D'$
0	1	0	1	0	
0	1	1	0	0	$F = 0$
0	1	1	1	0	
1	0	0	0	0	$F = 0$
1	0	0	1	0	
1	0	1	0	0	$F = D$
1	0	1	1	1	
1	1	0	0	1	$F = 1$
1	1	0	1	1	
1	1	1	0	1	$F = 1$
1	1	1	1	1	



مثالی از ساخت مدارهای ترکیبی با مالتی‌پلکسر

مثال: $F(A,B,C,D) = \sum(1,3,4,11,12,13,14,15)$

A	B	C	D	F	
0	0	0	0	0	$F = D$
0	0	0	1	1	
0	0	1	0	0	$F = D$
0	0	1	1	1	
0	1	0	0	1	$F = D'$
0	1	0	1	0	
0	1	1	0	0	$F = 0$
0	1	1	1	0	
1	0	0	0	0	$F = 0$
1	0	0	1	0	
1	0	1	0	0	$F = D$
1	0	1	1	1	
1	1	0	0	1	$F = 1$
1	1	0	1	1	
1	1	1	0	1	$F = 1$
1	1	1	1	1	



تمرین

- دو تابع زیر را در نظر بگیرید:

$F1(A, B, C) = \sum(0,1,2,4)$

$F2(A, B, C) = \sum(0,5,6,7)$

اگر این توابع با جدول کارنو ساده شوند می‌شود:

$F1 = (AB + AC + BC)'$

$F2 = AB + AC + A'B'C'$

- این دو تابع را با ۱. دیکدر ۲. مالتی‌پلکسر طراحی و رسم کنید.

مدارهای منطقی

(طراحی سیستم‌های دیجیتال)

جلسه هفتم

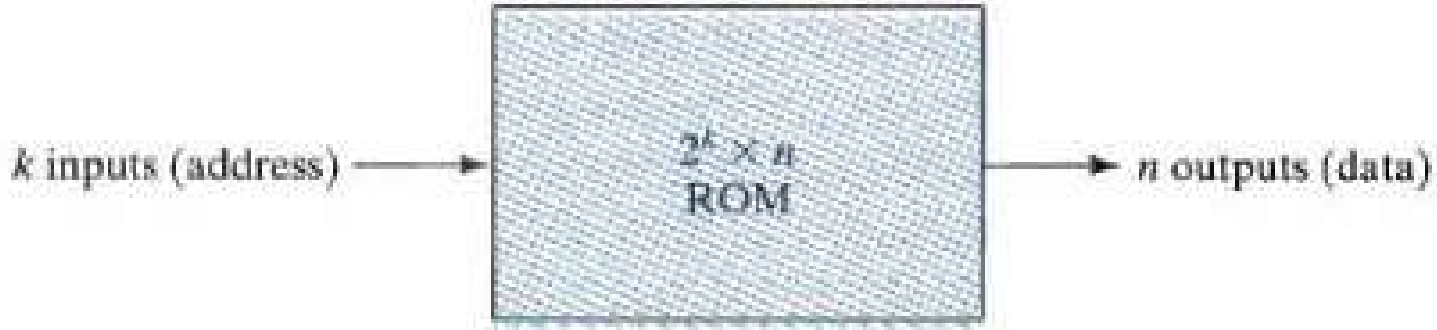
المان‌های قابل برنامه ریزی
برای ساخت مدار -
طراحی یک مدار ترکیبی

مباحث امروز

- المان‌های قابل برنامه ریزی: ROM, PAL, PLA
- طراحی یک ماشین‌حساب جمع‌کننده ترکیبی:
- شناخت اجزای آن شامل:
ورودی کیبورد،
جمع‌کننده BCD و
راه‌انداز سون سگمنت.

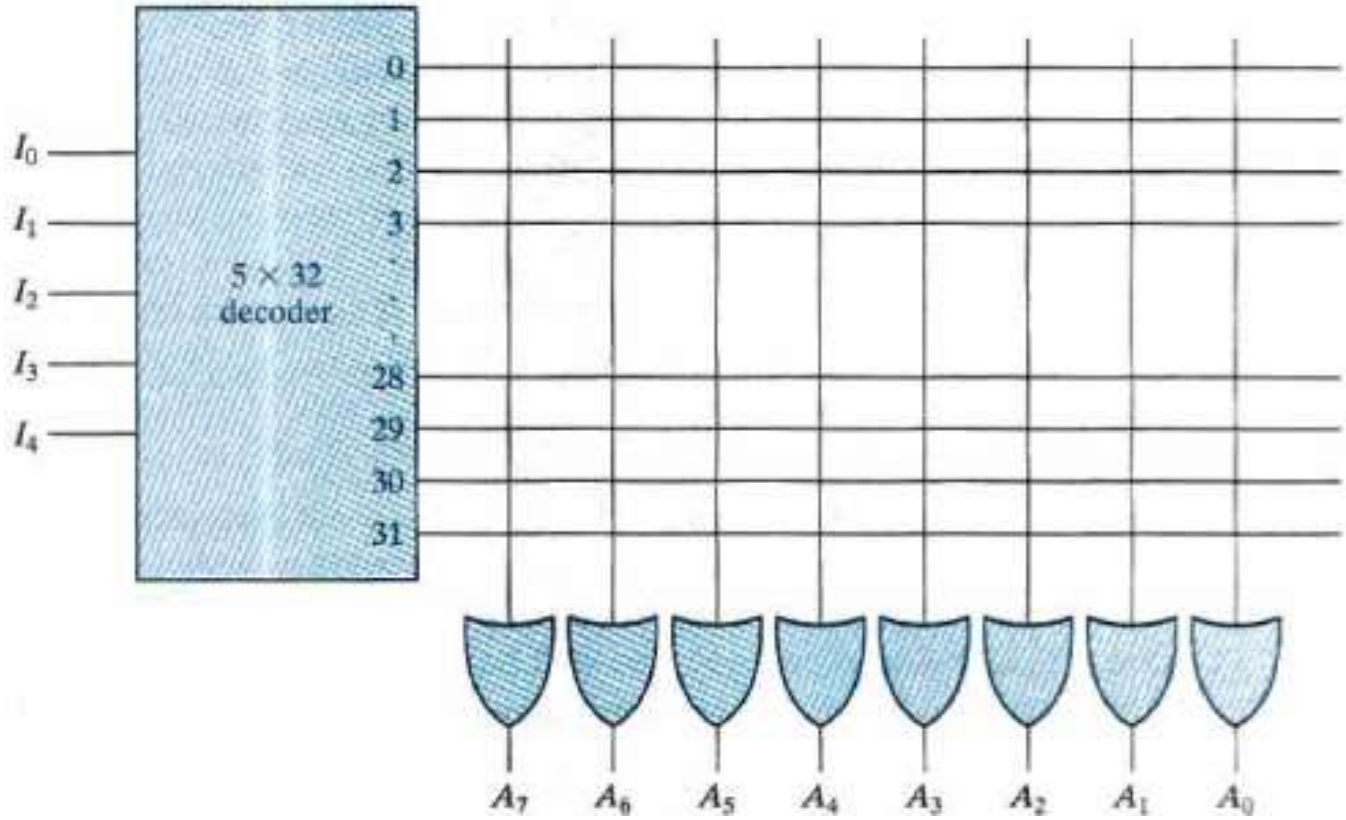
حافظه فقط خواندنی ROM

- [این مبحث در فصل هفتم کتاب مانو (ویرایش چهارم) از بخش ۷.۵ به بعد دنبال می‌شود.]
- رام یک قطعه دیجیتالی است که k خط ورودی و n خط خروجی دارد. بین n و k هیچ رابطه‌ای نیست.
- با یک ترکیب خاص در خطوط ورودی، یک ترکیب خاص و از پیش تعیین شده در خروجی ظاهر می‌شود.
- ما اینجا از رام بعنوان یک المان ساخت مدار استفاده می‌کنیم، نه به عنوان یک عنصر حافظه‌ای.



انتقال اطلاعات به رام

- اطلاعات رام که توسط طراح تعیین می‌شود، یک بار و برای همیشه به رام منتقل می‌شود و این الگو در آن غیرقابل تغییر باقی می‌ماند (در شکل زیر رام بصورت یک دیکدر و تعدادی or ساخته شده و تمام ۲۵۶ تقاطع خطوط قابل نقطه گذاری به منظور اتصال هستند).

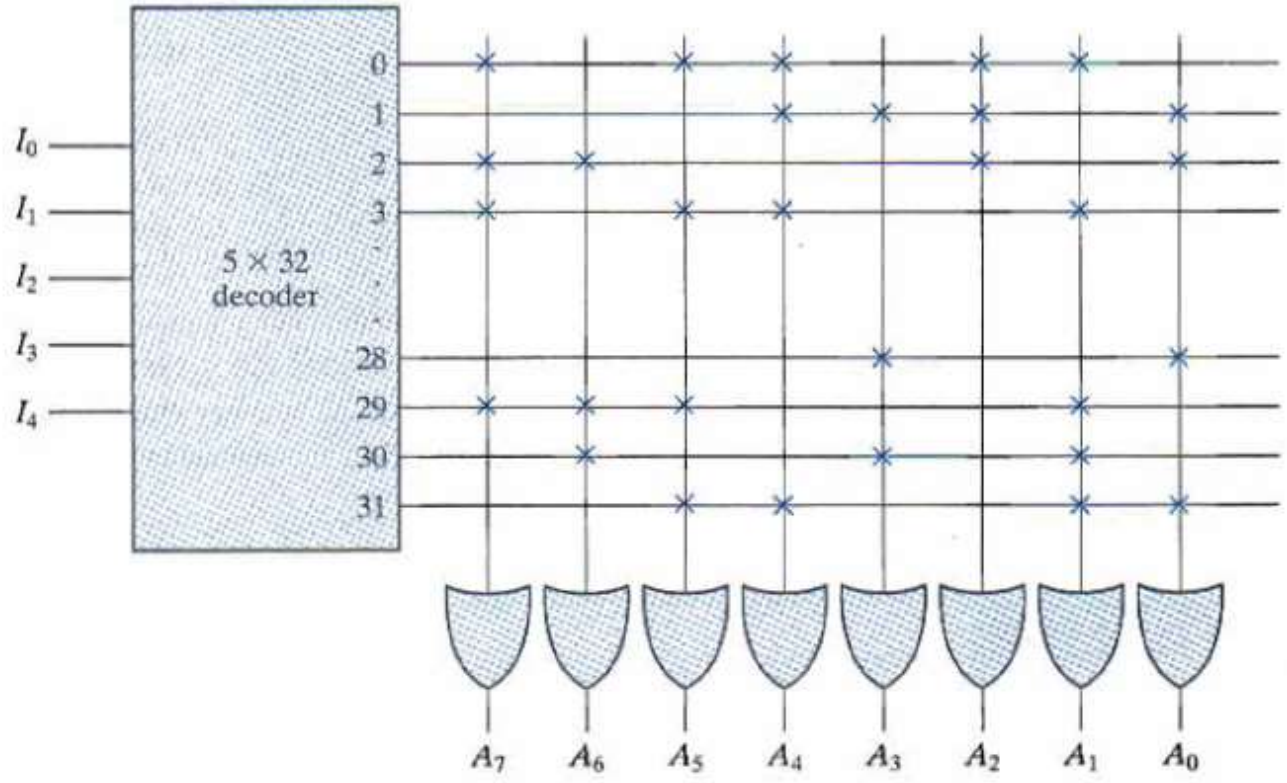


ساخت مدارهای ترکیبی با رام

- رام چندین خروجی دارد و هر کدام می‌تواند یک جدول درستی را پیاده نماید (شبیه دیکدر+or اما جای یک خروجی، چندین خروجی دارد).
- هر کدام از خروجی‌های رام بدون نیاز به هیچ گیت دیگری مطابق جدول درستی است (بر خلاف دیکدر که نیاز به or کردن خروجی‌ها دارد).
- چون مدارات داخلی رام با سوزاندن فیوزهای داخلی شکل می‌گیرد و بسیار سریع است اما چون نیاز به دیکد آدرس به همراه یک or دارد، همچنان دو طبقه تاخیر خواهد داشت.

مثالی ازساخت مدارهای ترکیبی با رام

- برای ساخت یک مدار ترکیبی از یک را با ۵ خط ورودی و ۷ خط خروجی استفاده شده.
- هر یک از این ۷ خط یک جدول درستی ۳۲ سطری دارند از جمله: $A7(I4,I3,I2,I1,I0) = \Sigma\,(0, 2, 3, \ldots, 29)$



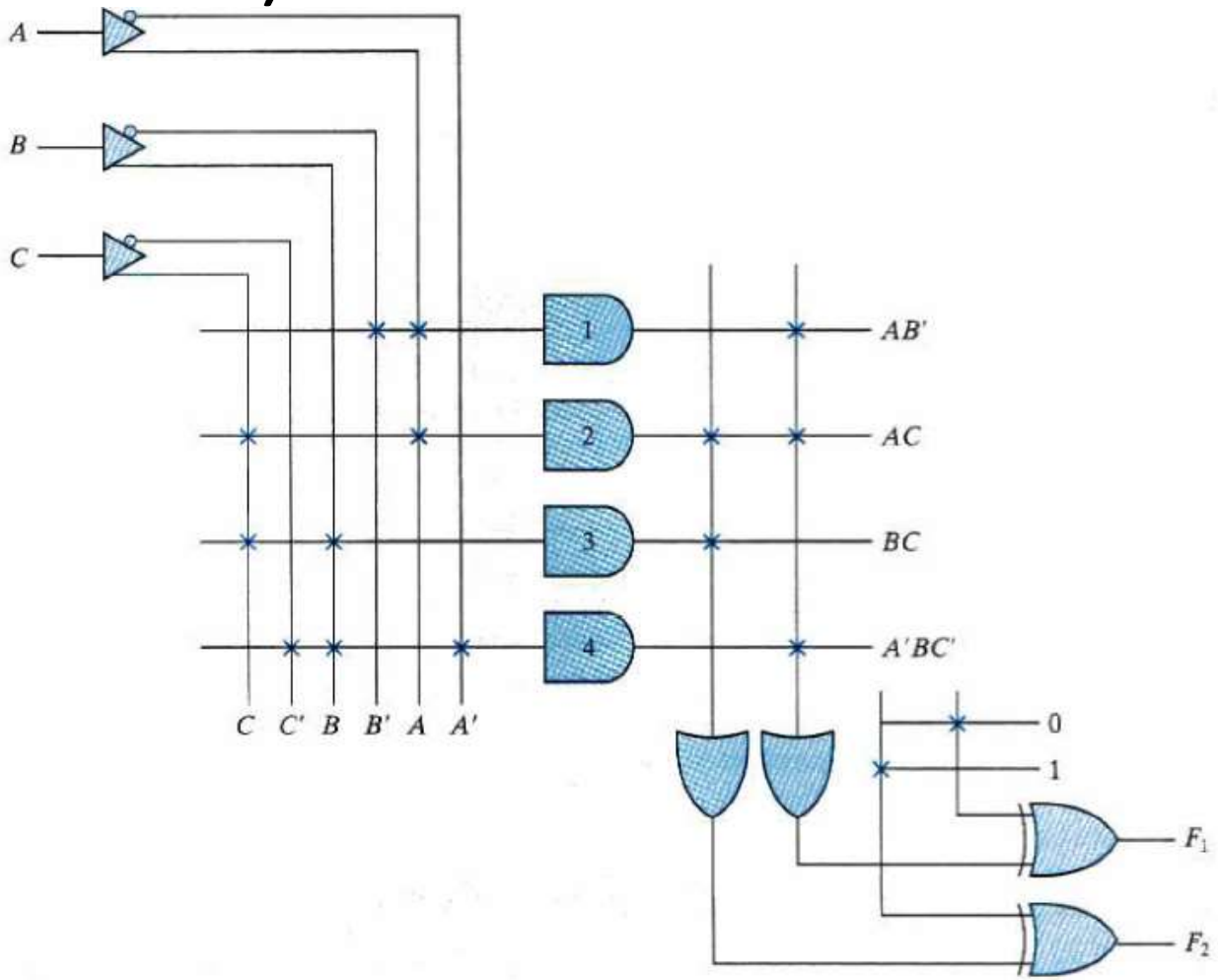
آرایه منطقی قابل برنامه‌ریزی PLA

- هر خروجی رام یک عبارت کانونی را پیاده‌سازی می‌کند (در هر مینترم تمام متغیرها حاضرند). ساخت مدار با دیکدر و مالتی‌پلکسر هم با فورمول کانونی بود.
- این در حالی است که با روش‌های ساده‌سازی مانند جدول کارنو ما توانستیم توابع را ساده کنیم و درنتیجه جمله‌هایی با تعداد کمتر متغیر داشته باشیم.
- PLA قطعه‌ای است که می‌تواند عبارات استاندارد SOP را پیاده‌سازی کند.
- PLA های موجود در بازار از نظر تعداد ورودی و خروجی و مضروب داخلی متنوع هستند.

مثال: طراحی مداری با PLA

F1 = AB' + AC + A'BC'

F2 = (AC + BC)'



پیاده‌سازی مدار با PLA

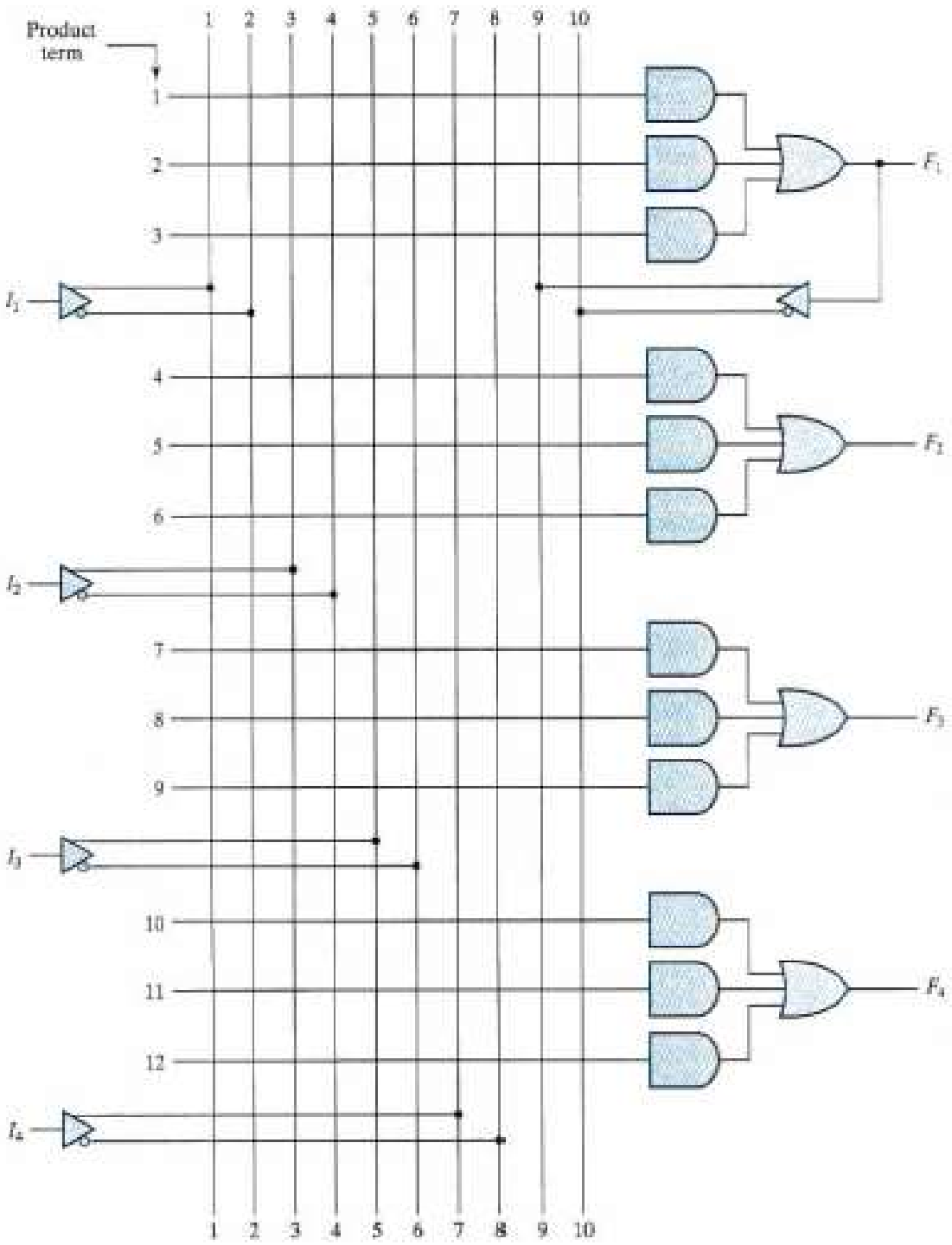
- PLA قطعه‌ای است که اختصاصا برای پیاده‌سازی توابع منطقی ساخته شده است (بر خلاف ROM که عنصری حافظه‌ای است).
- همانطور که در مثال دیدید، طراحی همانند رام در با سوزاندن فیوزها انجام می‌شود، اما در PLA در دو قسمت (آرایه) این کار صورت می‌گیرد. سمت چپ برای ساختن یک حاصل‌ضرب (مثل AB') و سمت راست برای انتخاب تعدادی از این حاصل‌ضرب‌ها و مجموع ساختن آنها.
- مکمل کننده تابع خروجی (قسمت سوم) شاید موجود باشد.

مدارمنطقی آرایه‌ای قابل برنامه‌ریزی PAL

- PAL مشابه PLA است. با این تفاوت که فقط یک قسمت (آرایه) قابل برنامه‌ریزی دارد، نه دو تا.
- در PAL بر خلاف PLA نمی‌توان از جمله مشترک برای چند خروجی استفاده کرد، اما در عوض می‌توان از کل یک خروجی بعنوان ورودی توابع دیگر استفاده نمود.
- طراحی با PAL ساده‌تر ولی محدودتر است. تعداد جمله‌ها (که در یک تابع خروجی با هم جمع می‌شوند) محدود است.

مثال از PAL

- یک PAL با چهار ورودی (متغیر) و چهار خروجی (تابع). حداکثر تعداد جمله‌ها ۳ است. یکی از خروجی‌ها بعنوان ورودی استفاده شده است.



طراحی یک ماشین‌حساب

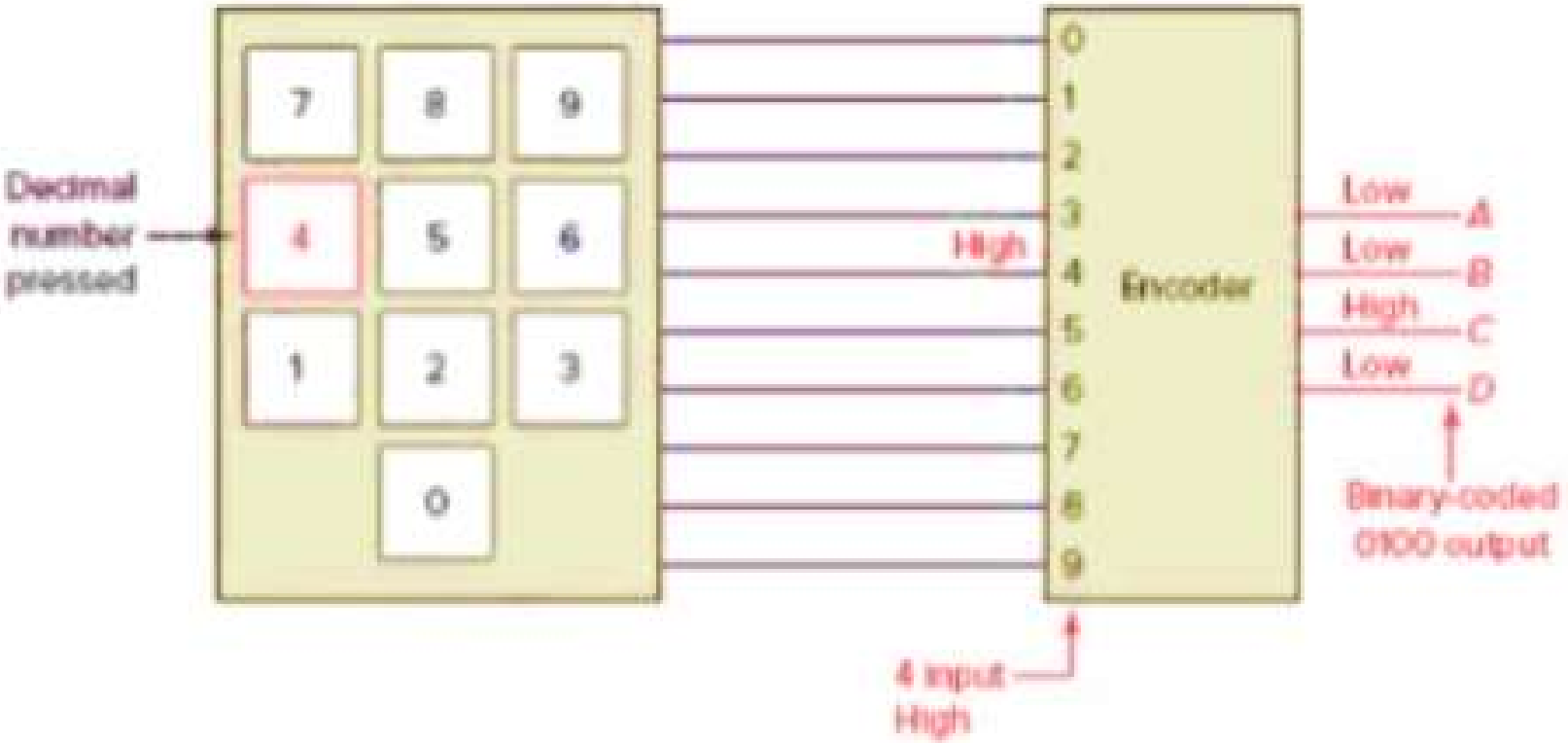
مساله: ماشین‌حساب ترکیبی

- یک ماشین حساب طراحی کنید که دو عدد یک رقمی را با هم جمع کند.
- ورودی این مدار دو صفحه کلید است که اعداد با شماره‌های 0 تا 9 روی دگمه‌ها نوشته شده (کلا ۲۰ خط ورودی). خروجی مدار دو عدد سون‌سگمنت است.
- کاربر همزمان دو دگمه (یکی از صفحه کلید سمت چپ و دیگری از صفحه کلید سمت راست) را فشار می‌دهد و نگه می‌دارد. حاصل جمع این دو عدد باید روی نمایشگر دو رقمی ظاهر شود.

صفحه کلید و انکدر

- برای ساخت یک ورودی تک رقمی، ده عدد دگمه (همانند شستی‌هایی که در مثال رای‌گیری وجود داشت) را به یک انکودر 16x4 وصل می‌کنیم. از آنجا که فقط ۱۰ عدد ورودی داریم خطوط ورودی 10 تا 15 انکودر به جایی وصل نمی‌شوند (در شکل نشان داده نشده).
- چنین مداری بسته به عدد فشار داده شده، یک عدد را روی خروجی (چهارتایی) خود ظاهر می‌کند.
- تا اینجای کار فرقی نمی‌کند که این عدد را باینری بدانیم یا BCD چون برای اعداد کوچکتر از ۱۰ این دو یکی هستند.

صفحه کلید و انکدر



در شکل، high به معنای 1 و low به معنای 0 است.

جمع‌کننده BCD

- از آنجا که خروجی به صورت تک رقمی دهدهی روی نمایشگر نشان داده می‌شود، نیاز به جمع‌کننده BCD داریم.
- همانطور که قبلا گفته شد: اعداد BCD همانند اعداد باینری هستند با این تفاوت که در هر چهار بیت فقط اعداد ۰ تا ۹ نمایش داده می‌شود و نمایش ۱۰ تا ۱۵ (دهدهی) مجاز نیست.
- در چه صورت حاصل دو رقمی می‌شود؟

جمع‌کننده BCD

- اگر حاصل جمع باینری چهار رقمی دارای کری شود (مثل وقتی ۹ و ۸ را جمع می‌کنیم)، یا حاصل کری نداشته باشد اما بزرگتر از ۹ باشد (مثل وقتی ۶ و ۵ را جمع می‌کنیم) جمع BCD در هر دو صورت دارای نقلی (به رقم کناری) خواهد بود.
- تابع زیر کمک می کند بدانیم حاصل دارای کری (بیش از ۱۵) یا بین ۱۰ و ۱۵ است.

ارقام حاصل و کری را چنین می‌نامیم: $K\ Z_8\ Z_4\ Z_2\ Z_1$

جمع‌کننده BCD

- اگر حاصل جمع باینری چهار رقمی دارای کری شود (مثل وقتی ۹ و ۸ را جمع می‌کنیم)، یا حاصل کری نداشته باشد اما بزرگتر از ۹ باشد (مثل وقتی ۶ و ۵ را جمع می‌کنیم) جمع BCD در هر دو صورت دارای نقلی (به رقم کناری) خواهد بود.
- تابع زیر کمک می کند بدانیم حاصل دارای کری (بیش از ۱۵) یا بین ۱۰ و ۱۵ است.

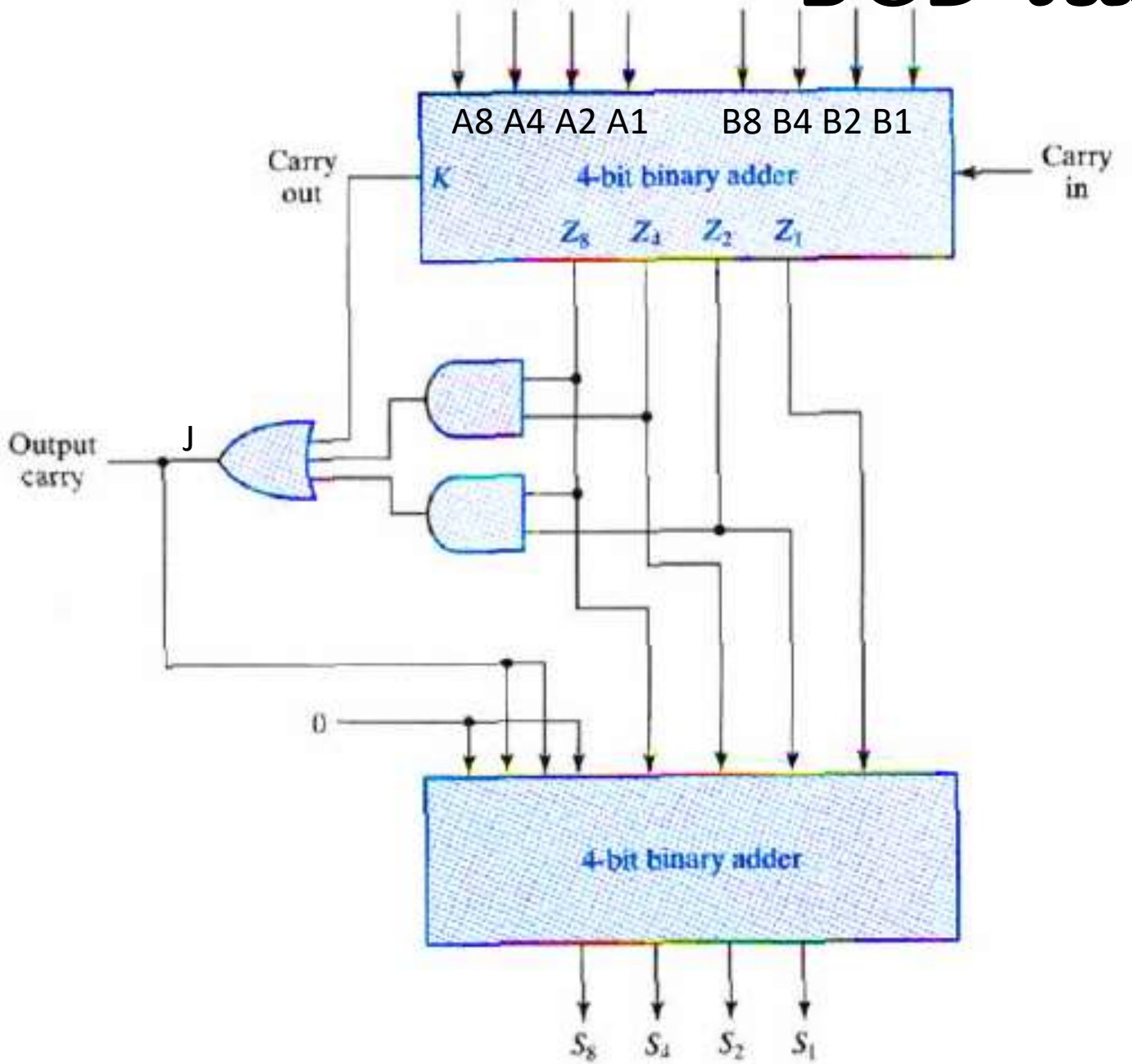
$$C = K + Z_8Z_4 + Z_8Z_2$$

- برای درست کارکردن جمع‌کننده BCD اگر حاصل بزرگتر از ۹ شد؛ ضمن ایجاد نقلی، ۱۰ را از عدد حاصل کم می‌کنیم (طبق محاسبه زیر با 0110 جمع می‌شود).

10 = 0(1010)

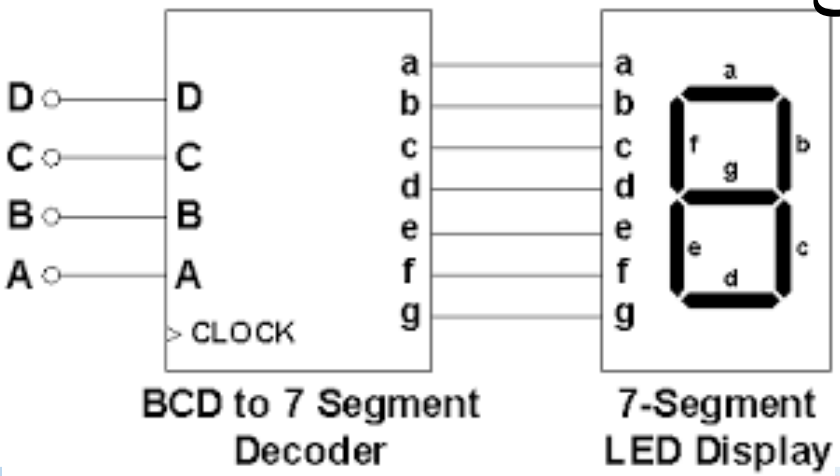
-10 = 1(**0110**)

جمع کننده BCD



سون‌سگمنت و درایور آن

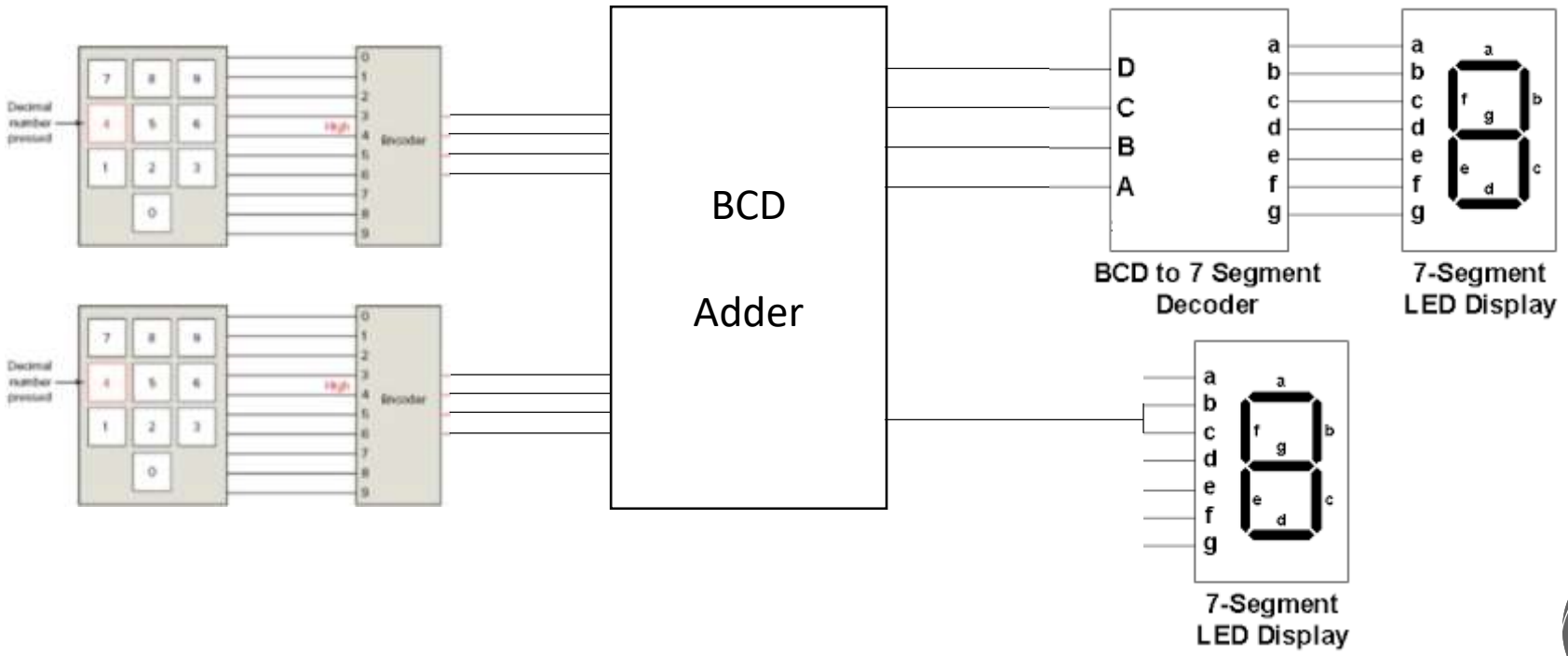
- چند جلسه قبل دیدیم که چطور می‌توان برای راه‌اندازی هر کدام از سگمنت‌ها مداری طراحی کرد.
- همچنین می‌توان راه‌انداز را به صورت یکجا بوسیله ROM یا PLA ساخت، یا اینکه اصلا یک IC که ویژه این کار است (دیکدر سون‌سگمنت) است را خرید.
- در هر حال مداری داریم که ۴ بیت BCD را می‌گیرد و سون‌سگمنت را روشن می‌کند.



طراحی ماشین‌حساب

- با استفاده از کامپوننت‌هایی که امروز شناختیم به راحتی می‌توانیم ماشین حساب را شامل سه بخش (طبقه) ورودی و محاسبه و خروجی تکمیل کنیم.
- خروجی می‌تواند عددی یک رقمی یا دو رقمی باشد. فقط توجه کنید که برای سون‌سگمنت دوم نیازی به درایور نیست. تنها عددی که ممکن است روی آن نمایش داده شود 1 است، پس کافیست سگمنت‌های b و c را به کری خروجی جمع کننده وصل نماییم.

طراحی ماشین‌حساب



نقطه ضعف این ماشین حساب

- هدف ما ساخت یک ماشین حساب ساده یک رقمی مانند ماشین حساب‌های معمولی ولی فقط برای عمل جمع بود، این هدف محقق شد جز یک مساله:
- ماشین‌حساب‌های معمولی عدد را در حافظه خود نگه می‌دارند و لازم نیست دائما دگمه‌های آنها را در حال فشرده شده نگه داشت. اما این ماشین حساب چون یک مدار کاملا ترکیبی است خروجی آن به ورودی لحظه‌ای وابسته است و حافظه‌ای برای نگهداری اعداد ندارد. مدارهای ترتیبی این نقص را رفع می‌کنند.

تمرین

- مدارهای هفته قبل را با PLA طراحی و رسم کنید.
- این ماشین حساب قرار است دو رقم آخر شماره دانشجویی شما را با هم جمع کند. مدار کامل (اسلاید ۱۳و۱۶) را روی کاغذ رسم کنید یا پرینت کنید. کلیه خطوطی که در آنها ولتاژ High وجود دارد را با رنگ قرمز مشخص کنید.

تمرین امتیازی (اختیاری)

- با استفاده از جمع کننده ۴ بیتی کامل (با کری ورودی و کری خروجی) و اضافه کردن چند گیت، مداری طراحی کنید که قدر مطلق ورودی ۴ بیتی با فرمت مکمل ۲ را در خروجی قرار دهد.
- کم ارزشترین رقم شماره دانشجویی‌تان را به BCD تبدیل کرده و به عنوان ورودی روی مدار با قلم قرمز نشان دهید. اگر این شماره ۹ است آن را 1001 و اگر هشت است1010 بگیرید. (چون اعداد علامتدارند، سه بیت حاوی مقدار محسوب شده و طبیعتا ۸ و ۹ منفی حساب خواهند شد.)

مدارهای منطقی

(طراحی سیستم‌های دیجیتال)

جلسه هشتم

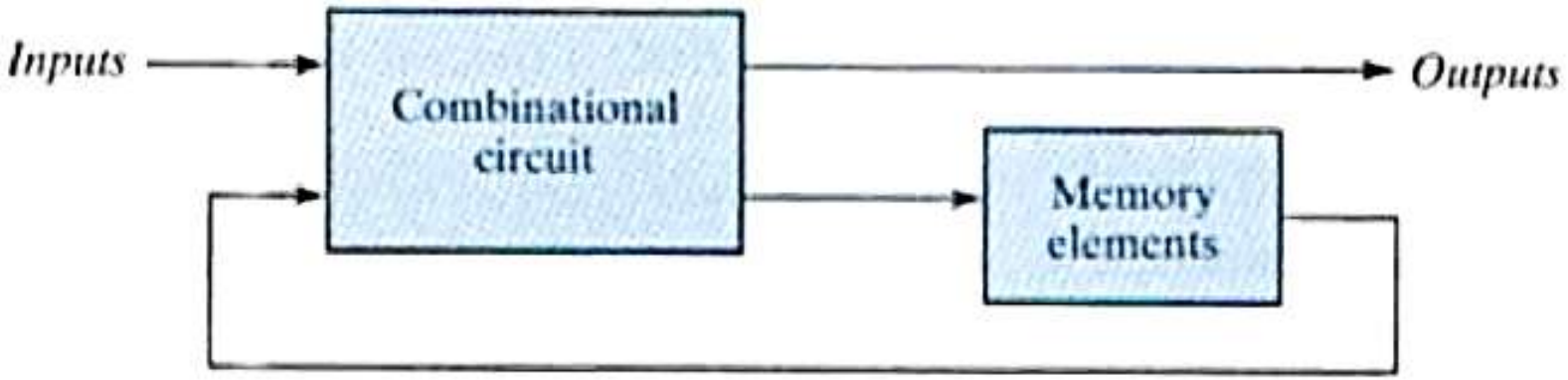
مدارهای منطقی ترتیبی

مباحث امروز

- آشنایی با مدارات ترتیبی
- لچ (SR,D)
- پالس ساعت
- فلیپ فلاپ (D,JK,T)

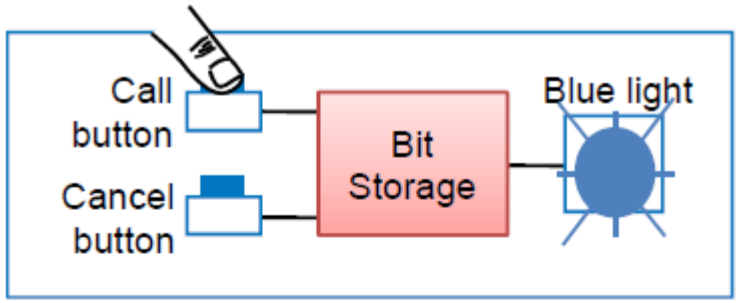
فرق مدارهای ترکیبی و ترتیبی

- یک مدار ترکیبی از عناصری که تابحال شناختیم مثل گیت‌های منطقی، جمع و تفریق کننده، دیکدر، مالتی‌پلکسر و .... تشکیل شده. هر یک ازخروجی‌های این مدار در هر زمان تابعی است ورودی‌ها.
- یک مدار ترتیبی از تعدادی از همان عناصر ترکیبی ساخته شده بعلاوه برخی عناصر حافظه‌دار. خروجی مدار تابعی است از ورودی‌ها و حالت قبلی مدار (در حافظه).

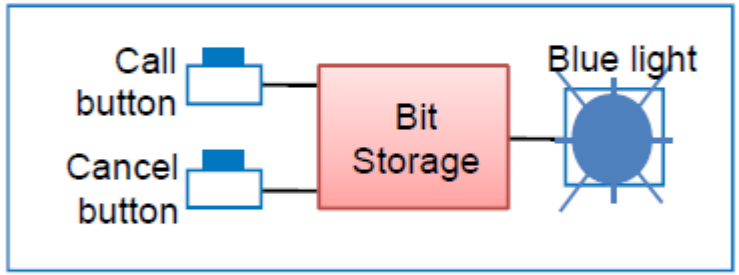


مدار سیستم
احضار مهماندار

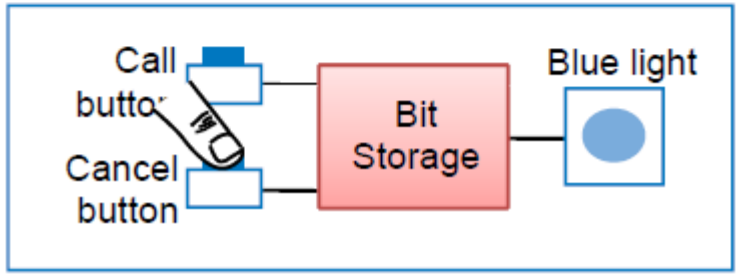
مثالی از عنصر حافظه‌ای



1. Call button pressed – light turns on

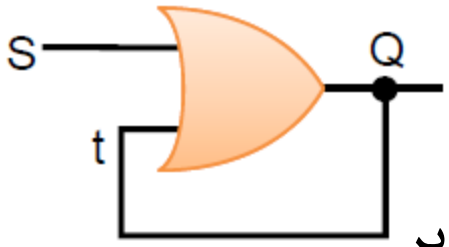


*2. Call button released – light **stays on***



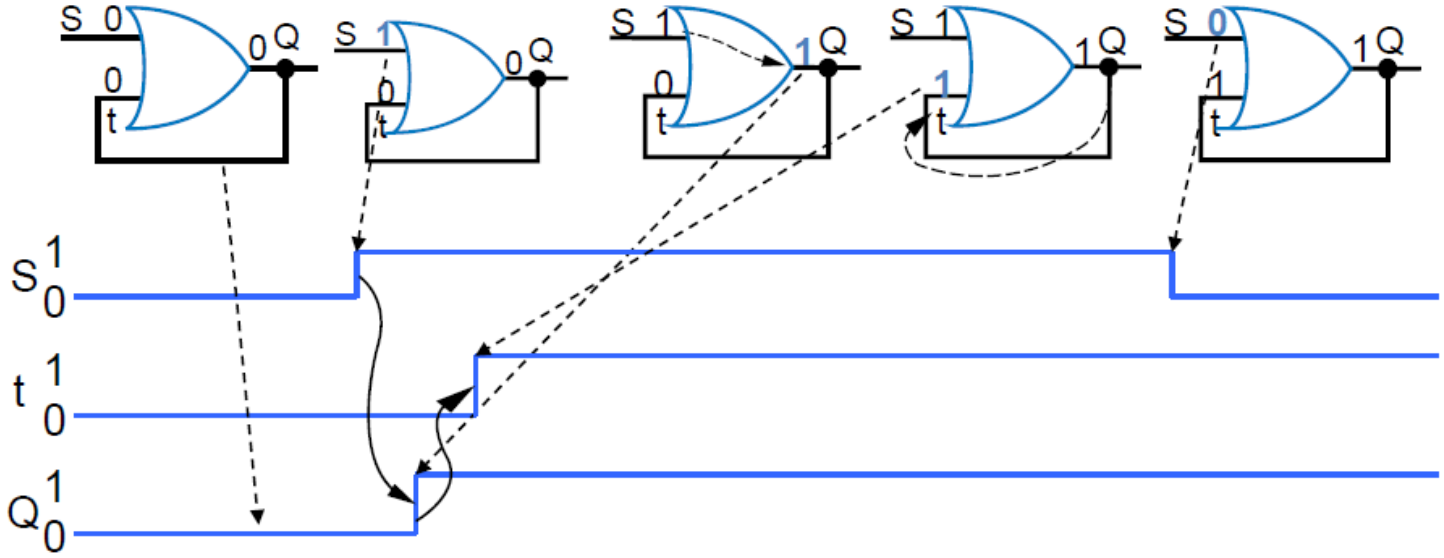
- در هواپیما برای «فراخوانی» مهماندار دگمه‌ای وجود دارد که با فشردن آن چراغی بالای سر مسافر روشن می‌شود. با رها کردن کلید فشاری، چراغ همچنان روشن می‌ماند تا زمانی که مهماندار بیاید و با زدن دگمه دیگری بنام «لغو» چراغ را خاموش کند.
- چگونه چنین مداری بسازیم؟

اولین تلاش برای ساخت مدار مهماندار

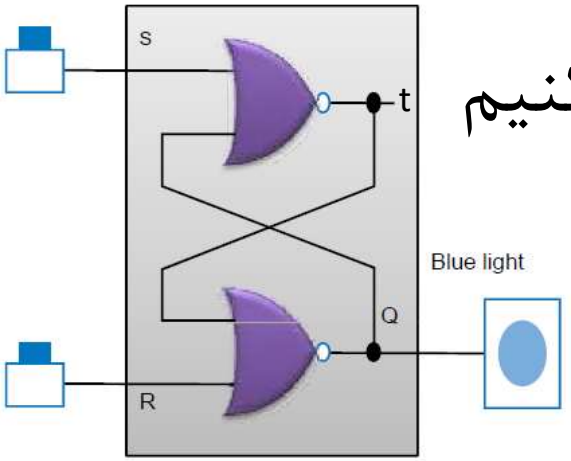


- به نظر می‌رسد که نیاز به نوعی فیدبک (برگرداندن خروجی به ورودی) داریم.
- با زدن دگمه اتفاقات (سیگنال) به ترتیب زیر می‌افتد، هرچند این مدار نیاز ما را برطرف نمی‌کند.

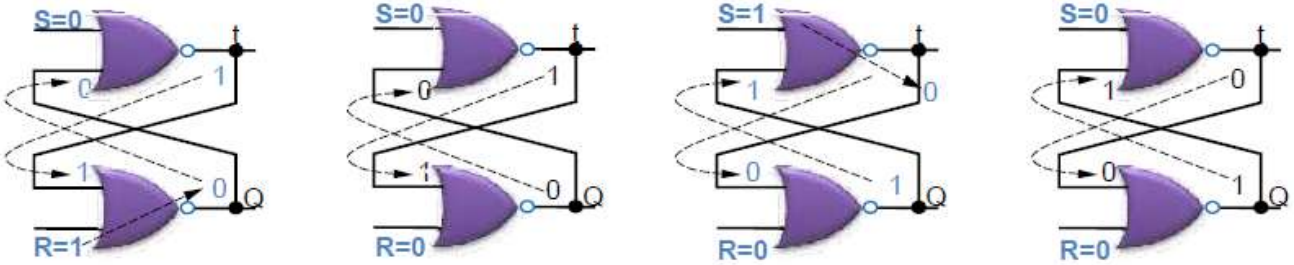
با زدن کلید و ایجاد ورودی S (ست) خروجی برقرار می‌شود و پایدار هم می‌ماند، اما راهی برایَ قطع خروجی ندارد.

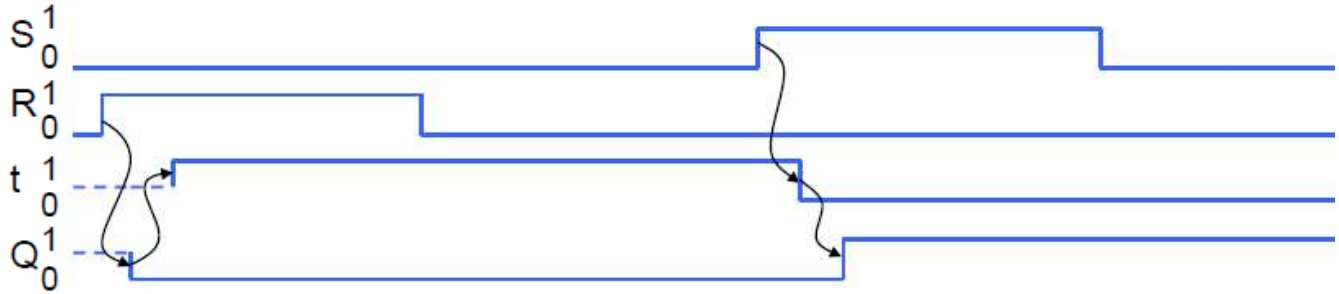


دومین تلاش

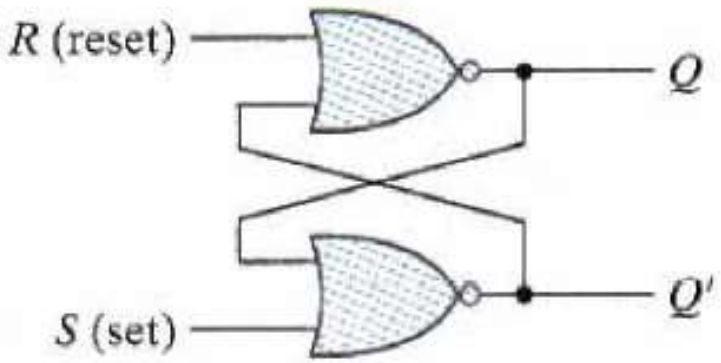


- این بار هم از فیدبک استفاده می‌کنیم اما دو گیت بکار می‌بریم (NOR).
- این مدار همان کاری که از آن انتظار داریم را انجام می‌دهد.
- این مدار (نگهدارنده یک بیت) لچ (چفت) نام دارد.





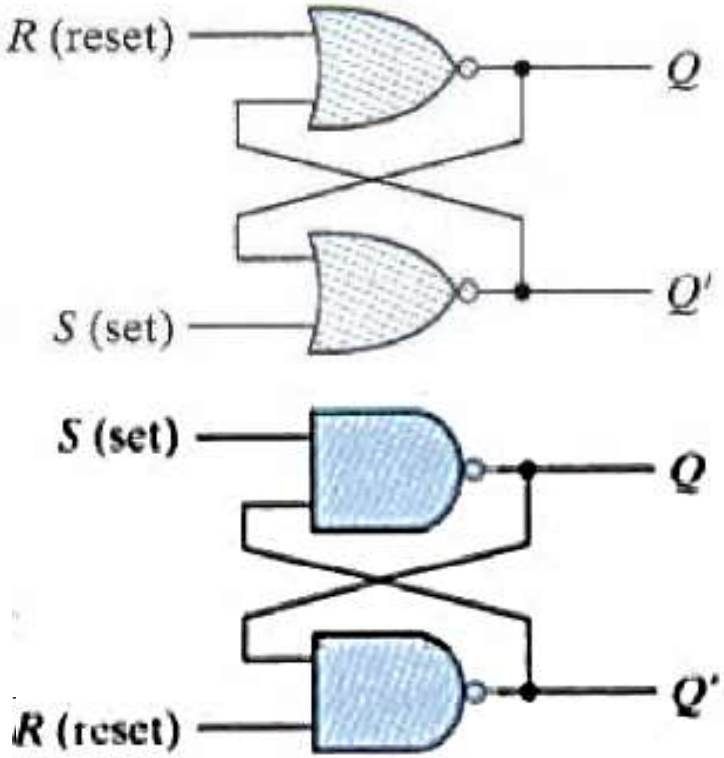
دو نوع لچ



S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	(forbidden)

دو نوع لچ

- لچ را هم می‌توان با NOR ساخت (که دیدیم) هم با NAND

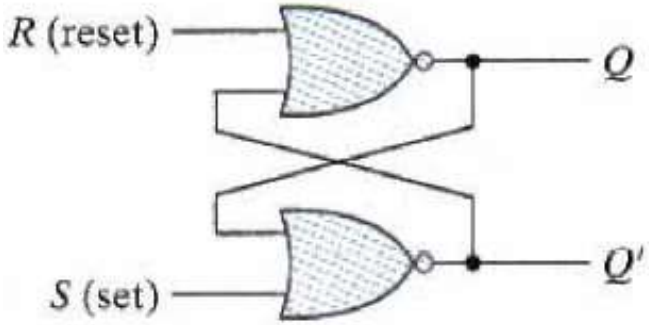


S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	(forbidden)

S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(after $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$)
0	0	1	1	(forbidden)

در مدار دوم ورودی‌ها Active-Low هستند یعنی مثلا ورودی S وقتی 0 باشد مدار را Set می‌کند.

ورودی نگهداری، ورودی ممنوعه

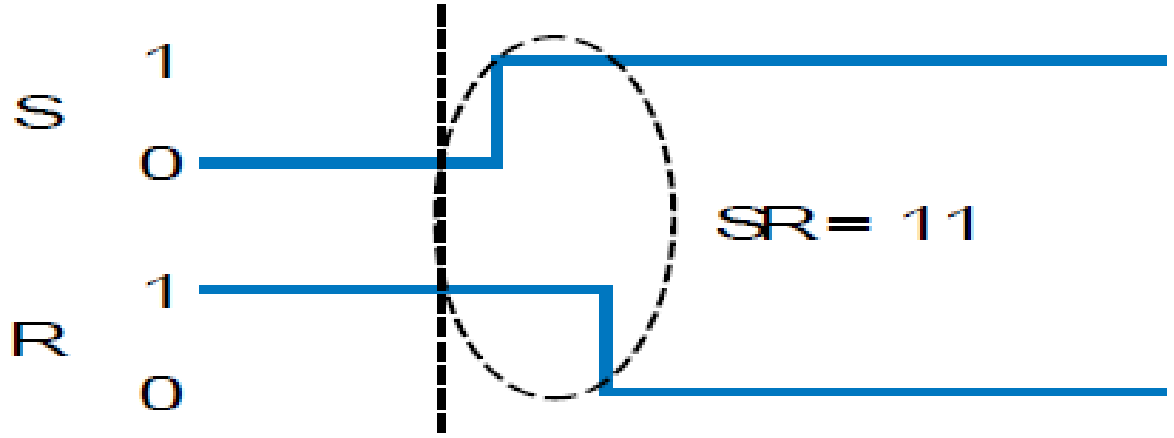


S	R	Q	Q′	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	(forbidden)

- در وضعیتی که هر دو ورودی 0 اند (یعنی نه set می‌کنیم نه reset)، مدار فقط حالت قبلی خودش را نگه می‌دارد. یعنی خروجی (Q) فقط به ورودی‌ها وابسته نیست.
- وضعیتی که هر دو ورودی ست و ریست 1 باشند ممنوع است چون اگر هر دو S و R همزمان فعال شوند Q و Q’ هر دوشان 1 می‌شوند و در ادامه اگر دو دگمه رها شوند مدار ناپایدار می‌شود و خروجی به نوسان می‌افتد و سپس در یک وضعیت نامعلوم می‌ایستد.
- وضعیت نامعلوم چیزی نیست که ما در مدارهای دیجیتال آن را بخواهیم.

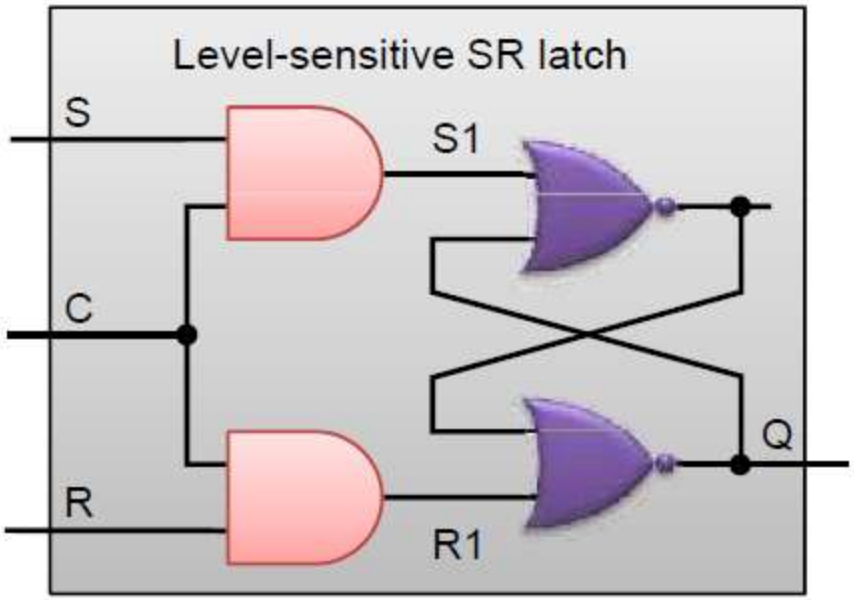
مشکل حالت ممنوعه لچ

- مشکل فقط این نیست که دو دگمه همزمان با هم فشرده شوند.
- ممکن است ورودی‌های S و R از مدارهایی بیایند که قرار نبوده هیچ زمانی R=1 و S=1 شود..... اما این اتفاق افتاده چون این دو از مدارات مختلف با تاخیر متفاوت آمده‌اند (ریس هزارد).



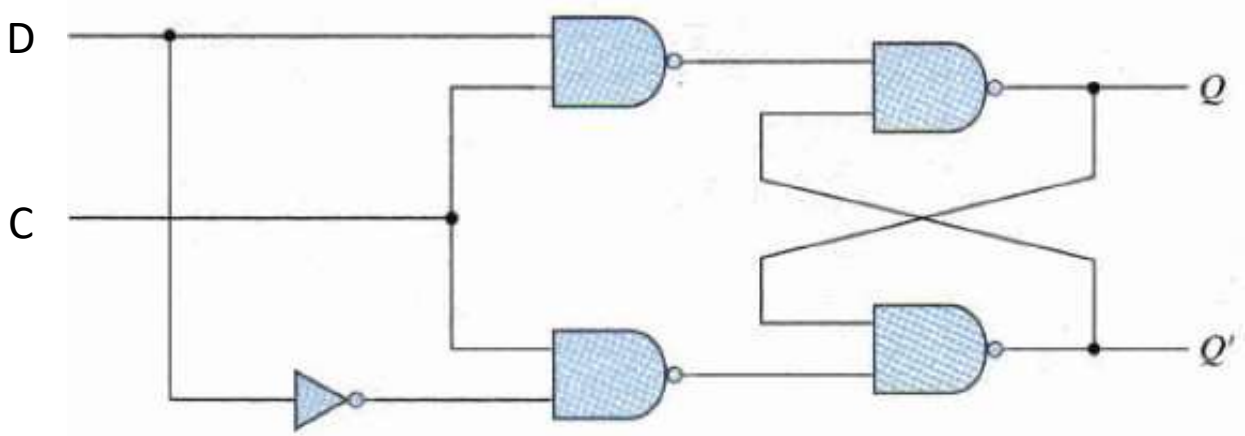
یک راه حل برای مشکل

- یک ورودی بنام C (کنترل) به مدار اضافه می‌کنیم. این ورودی در واقع فعال‌ساز لچ است.
- ورودی‌های R و S فقط وقتی مجاز به خوانده شدن هستند که C=0 باشد. C فقط زمانی 1 می‌شود که زمان کافی برای به پایداری رسیدنR و S طی شده باشد.



لچ D (حافظه یک بیتی)

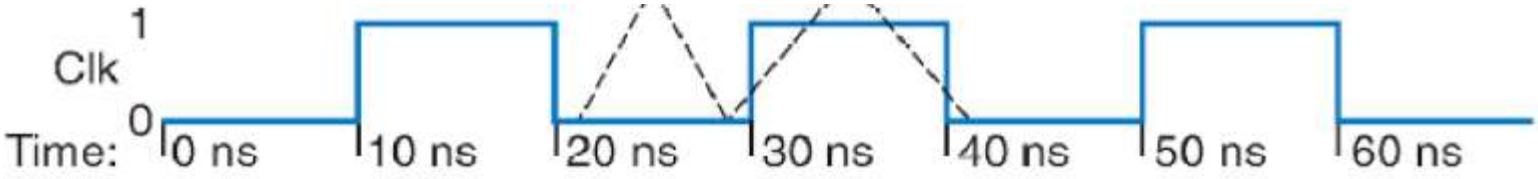
- این لچ فقط دو ورودی دارد. D (data) و C (کنترل فعال‌ساز). با 1 شدن فعال‌ساز آنچه در ورودی D است به خروجی (Q) منتقل می‌شود و پس از غیر فعال‌شدن (C) خروجی به همان حال در آنجا باقی می‌ماند.



- لچ D در واقع یک حافظه یک بیتی است که با C لود می‌شود و با صفر شدن C آنچه لود شده (یک بیت داده) را نگه می‌دارد.

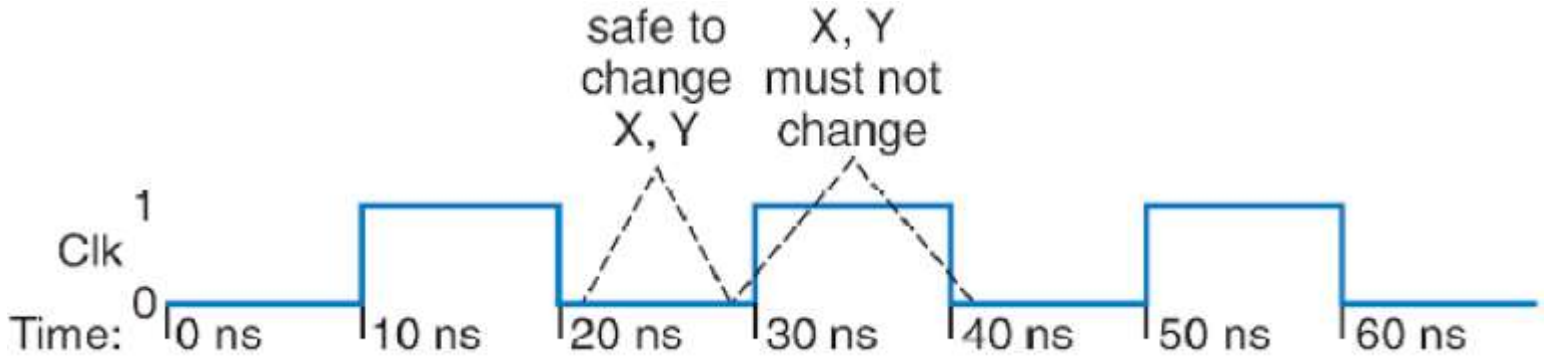
پالس ساعت

- سیگنالی است که بوسیله مولد ساعت تولید می‌گردد؛ دائما یک و صفر می‌شود و این نوسان 0 و 1 با فواصل زمانی ثابت انجام می‌شود.



- فقط یک پالس ساعت در سراسر سیستم دیجیتال طراحی شده در دسترس است و هر قطعه‌ای که به آن نیاز داشته باشد می‌تواند از آن استفاده کند.
- به مداراتی که از ساعت سراسری استفاده می‌کنند مدارات منطقی سنکرون (همزمان) ساعت‌دار می‌گویند.

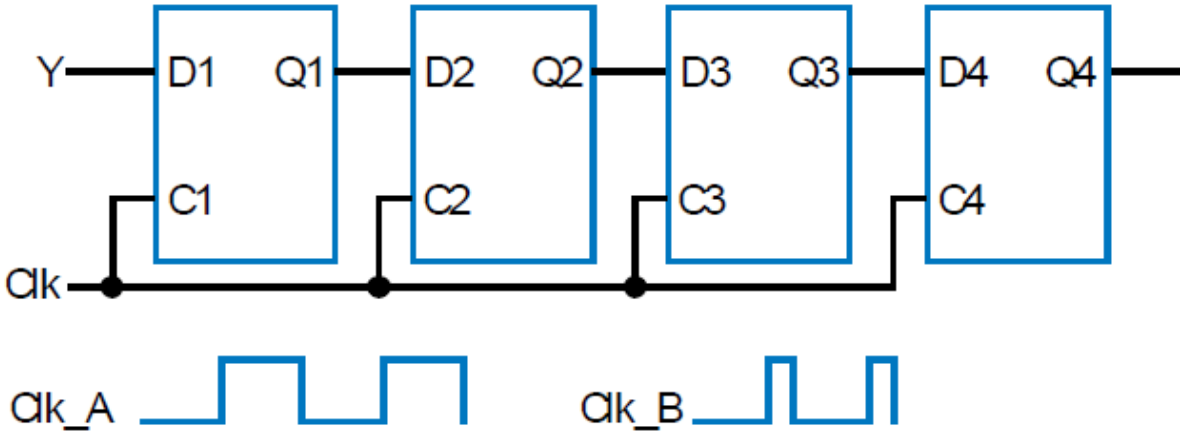
کاربرد پالس ساعت



- از پالس ساعت می‌توان برای ایجاد هماهنگی بین بخش‌های مختلف مدار استفاده کرد.
- مثلا ورودی‌های یک لچ تا زمانی که پالس ساعت که به C (کنترل یا کلاک نامیده شود فرقی ندارد) وصل شده 0 است فرصت برای تغییر دارند اما بعد از 1 شدن آن نباید تغییر نمایند (لچ در این زمان مقدار را از ورودی‌ها برمی‌دارد).

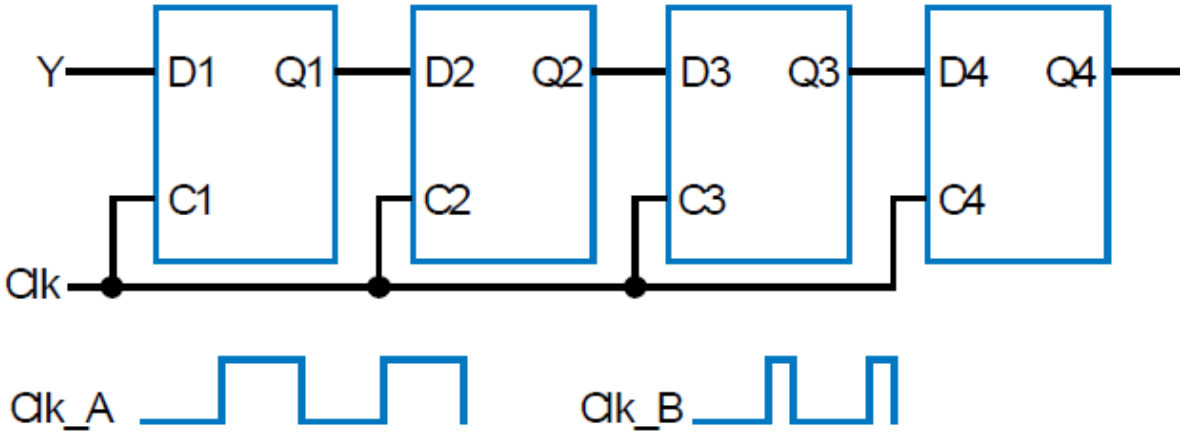
مشکل

- طراحی بدین گونه، حساس به سطح است. یعنی در تمام مدتی که پالس ساعت 1 است ورودی‌ها معتبر شمرده می‌شوند. **سوال:** در مدار زیر پس از یک پالس ساعت، چند لچ D ، ست می‌شوند؟



مشکل

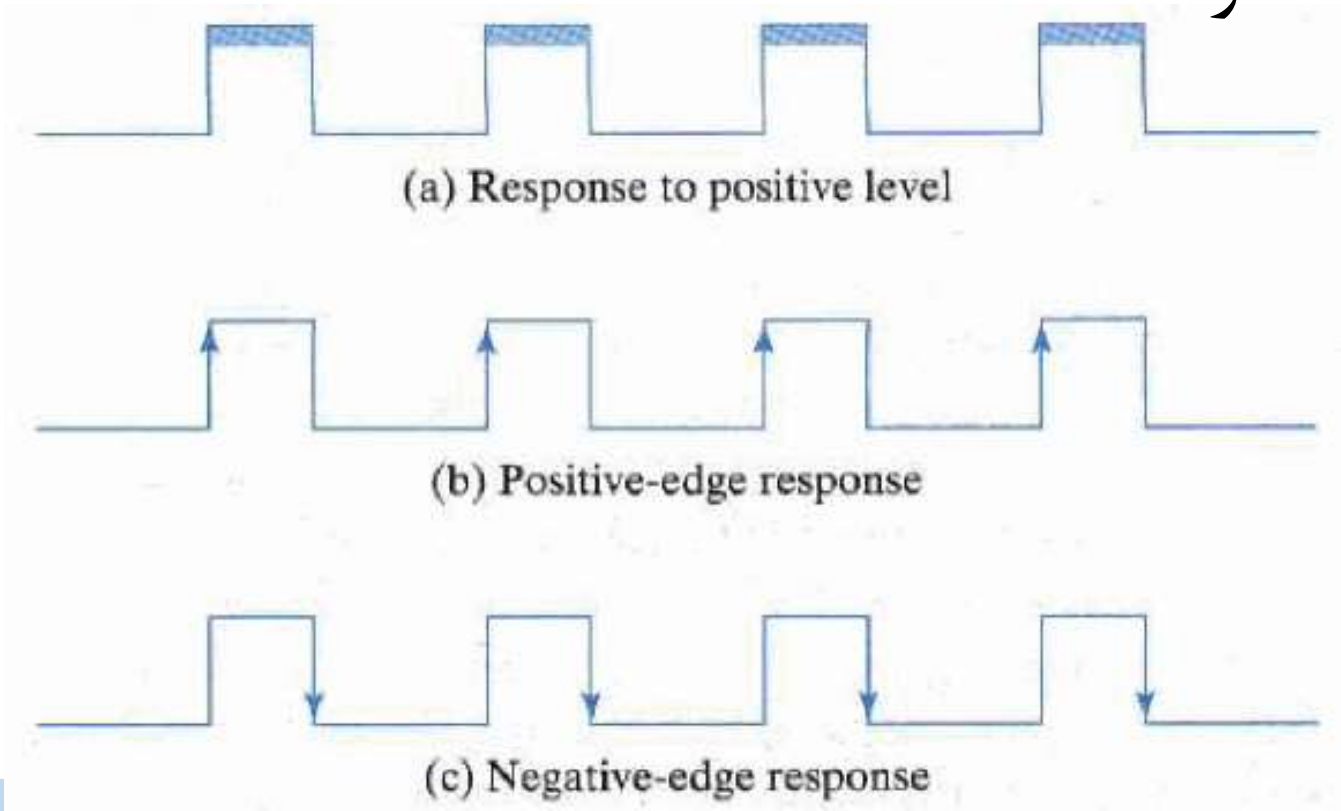
- طراحی بدین گونه، حساس به سطح است. یعنی در تمام مدتی که پالس ساعت 1 است ورودی‌ها معتبر شمرده می‌شوند. **سوال:** در مدار زیر پس از یک پالس ساعت، چند لچ D ، ست می‌شوند؟



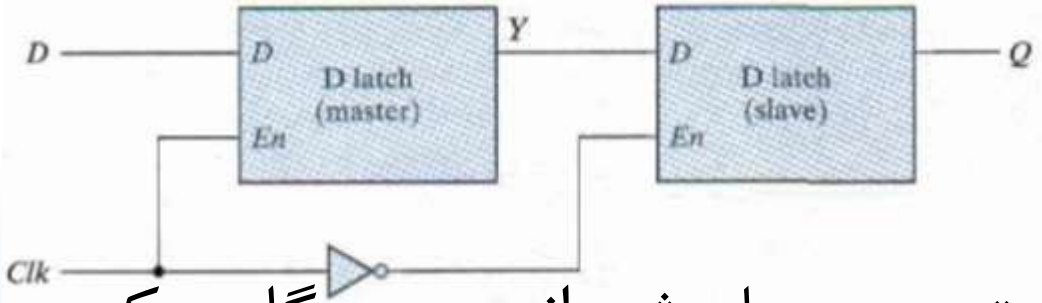
- پاسخ: بستگی به پریود ساعت (طول زمان 1 بودن) دارد. پس در شرایط فعلی کارکرد مدار نامعلوم است.

راه حل

- اگر بجای سطح، مدار حساس به لبه باشد (یعنی زمان تغییر فقط لحظه بالا رفتن یا پایین آمدن ساعت باشد نه کل زمانی که بالاست) این مشکل پیش نخواهد آمد.



طراحی حساس به لبه



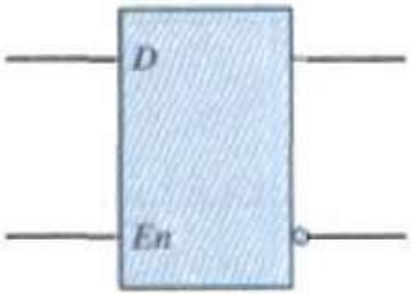
- دو لچ D به این صورت بهم وصل شده‌اند. در هنگامی که خط کنترلی (پالس ساعت) 1 است ورودی از D خوانده می‌شود و روی Y قرار می‌گیرد. در لحظه‌ای که پالس ساعت 0 می‌شود Y تثبیت می‌شود (D دیگرخوانده نمی‌شود). در همین لحظه آخرین مقدار Y به خروجی Q منتقل می‌شود (حساس به لبه منفی یا پایین رونده) و تا پالس پایین رونده بعدی (یک دوره ساعت) آنجا می‌ماند.
- این نوع طراحی دوتایی را طراحی ارباب-نوکر گویند که لچ سمت چپ ارباب و سمت راستی نوکر است (ارباب به بیرون گوش می‌دهد اما نوکر همیشه پیرو چیزی است که ارباب می‌گوید).

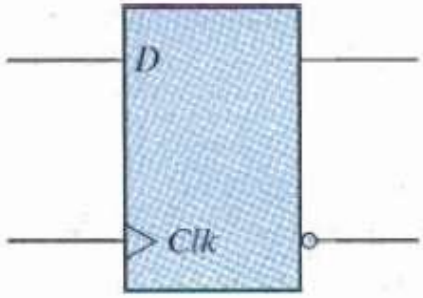
D Flip-Flop

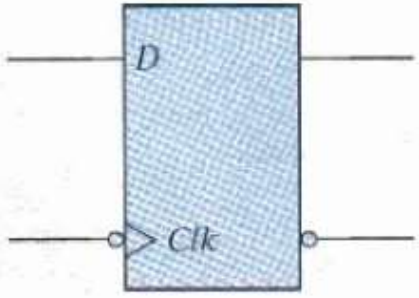
D	Q(t + 1)	
0	0	Reset
1	1	Set

فلیپ فلاپ D

- فلیپ فلاپ همانند لچ یک مدار منطقی دوحالته است. با این تفاوت که فلیپ فلاپ مداری ترتیبی است که ساعتدار است و با لبه ساعت کار می‌کند. در واقع مدار قبلی (که با دو لچ D ساخته شد)، یک فلیپ فلاپ D است.
- زمانی که لچ و فلیپ فلاپ بصورت یک کامپوننت در طراحی رسم شوند به شکل زیر خواهند بود (چپ به راست: ۱. لچ D ۲. فلیپ فلاپ D با تحریک لبه مثبت ۳. با لبه منفی)

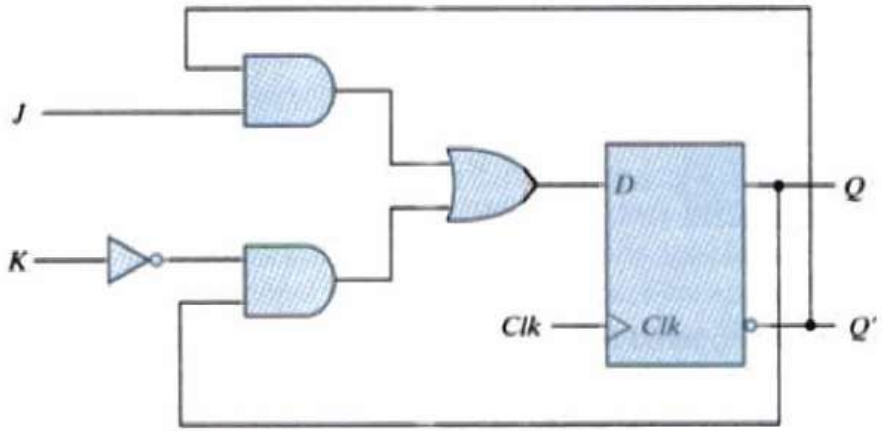






فلیپ فلاپJK

- فلیپ فلاپ JK عملکردی تقریبا شبیه لچ SR دارد.
J خروجی را 1 و K خروجی را 0 می‌کند. 0 بودن هر دو J و K موجب حفظ حالت موجود می‌شود اما اگر هر دو ورودی 1 باشند (متفاوت با SR که در حالت نامعلوم قرار می‌گرفت)، خروجی متمم می‌شود.
- جدول حالت را می‌توان طوری کشید که حالت قبلی جزء ورودی‌ها باشد.

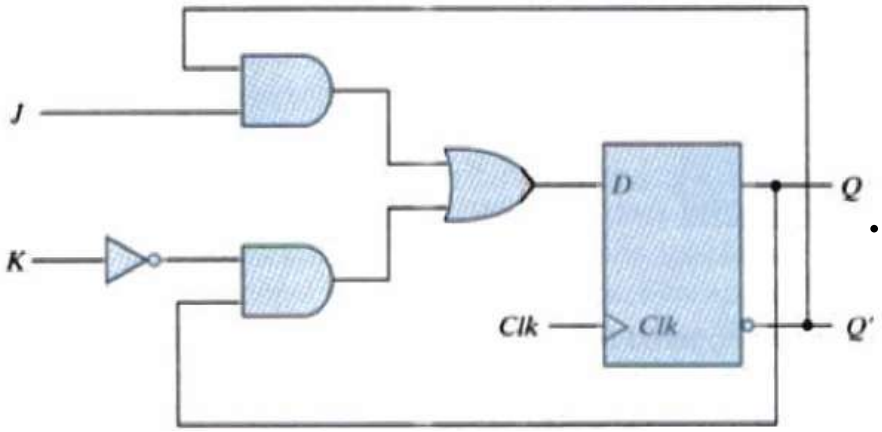


JK Flip-Flop

J	K	Q(t + 1)	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

فلیپ فلاپJK

- فلیپ فلاپ JK عملکردی تقریبا شبیه لچ SR دارد. J خروجی را 1 و K خروجی را 0 می‌کند. 0 بودن هر دو J و K موجب حفظ حالت موجود می‌شود اما اگر هر دو ورودی 1 باشند (متفاوت با SR که در حالت نامعلوم قرار می‌گرفت)، خروجی متمم می‌شود.
- جدول حالت را می‌توان طوری کشید که حالت قبلی جزء ورودی‌ها باشد.



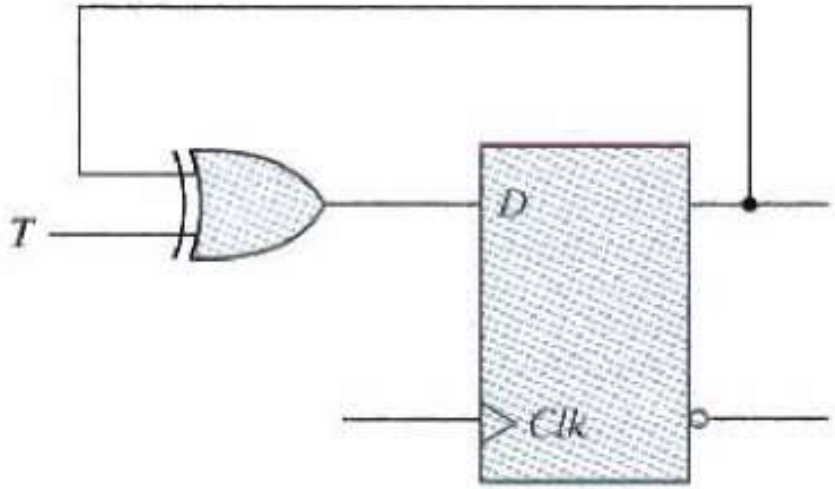
JK Flip-Flop

J	K	Q(t + 1)	
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Complement

J	K	Q(t)	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

فلیپ فلاپ T

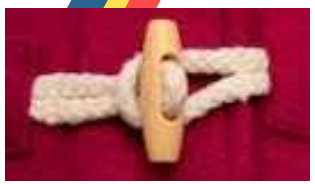
- فلیپ فلاپ T (toggle) شبیه نوع D فقط یک ورودی دارد، این فلیپ‌فلاپ وقتی ورودی 1 است، خروجی را معکوس به می‌کند و اگر صفر باشد حالت را حفظ می‌کند.



T Flip-Flop

T	Q(t + 1)	
0	Q(t)	No change
1	Q′(t)	Complement

- نکته:تمام فلیپ‌فلاپ ها در لحظه روشن شدن (وصل برق) مقدار خروجی تصادفی پیدا می‌کنند.



مدارهای منطقی

(طراحی سیستم‌های دیجیتال)

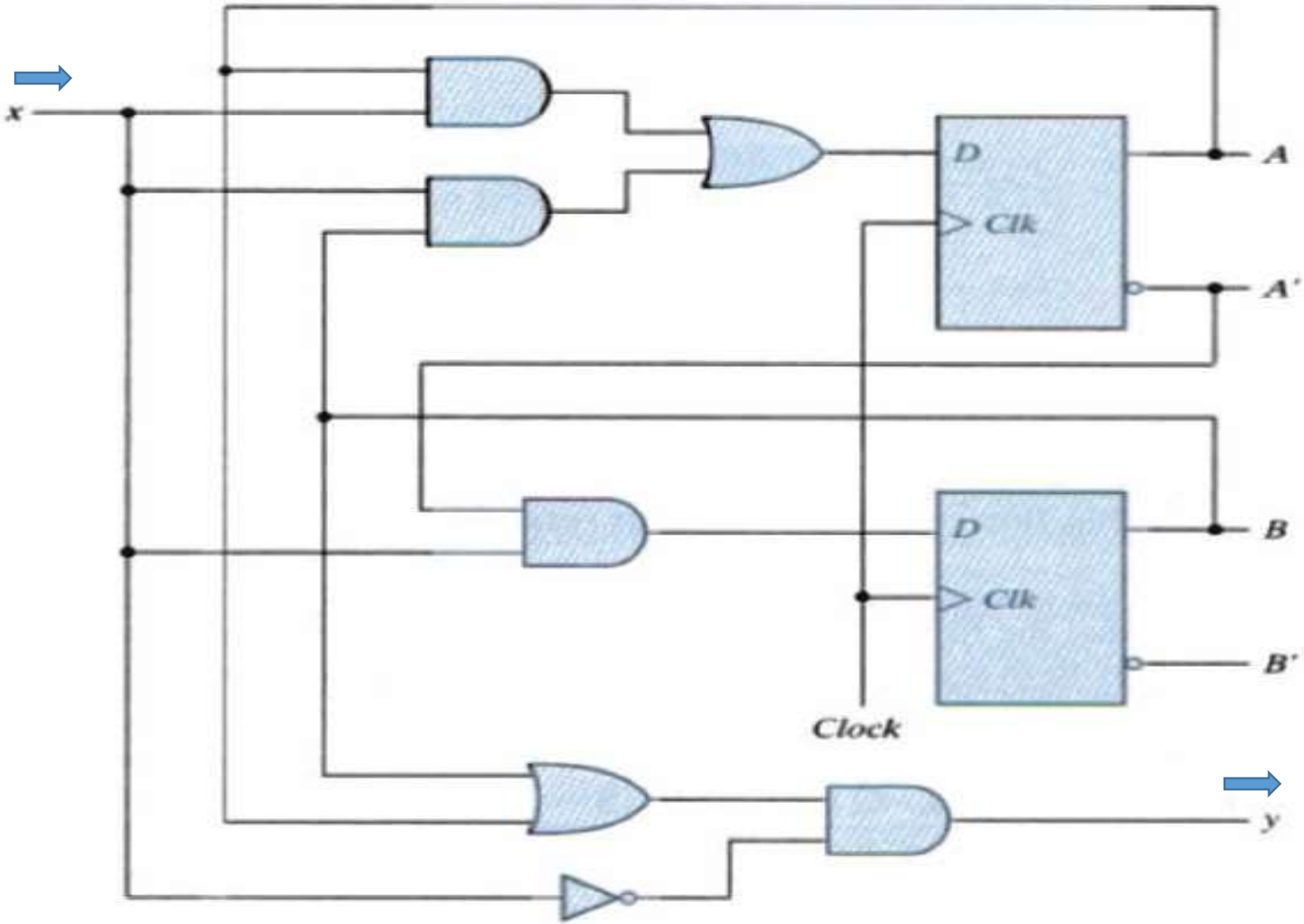
جلسه نهم

تحلیل مدارهای ترتیبی ساعت‌دار

تحلیل مدارهای ترتیبی ساعتدار

- مدار ترتیبی ساعتدار به مداری گفته می‌شود که دارای عناصر ترتیبی که با ساعت کار می‌کنند (فلیپ فلاپ) باشد (البته احتمالا دارای عناصر ترکیبی و گیت‌ها هم هست).
- زمانی که شکل مداری را داریم و با کشیدن جدول و نمودار می‌خواهیم عملکرد مدار را با توجه شرایط بشناسیم، مدار را تحلیل کرده‌ایم.
- شرایط مدار و خروجی‌های آن در مورد مدارات ترکیبی فقط به ورودی بستگی داشت اما اینجا به حالت قبلی مدار هم بستگی دارد.

مدار برای تحلیل



نوشتن معادله‌های حالت

- معادلات برای خروجی فلیپ‌فلاپ‌ها و خروجی‌های رسمی مدار نوشته می‌شود.
- در اسلاید قبل، از آنجا که فلیپ‌فلاپ D شفاف است (با رسیدن ساعت ورودی D را عینا به خروجی می‌برد) پس حالت بعدی (Q_{t+1}) مساوی است با فورمول ورودی D در همین حالا.
- پس بر اساس محاسبه ورودی هر فلیپ‌فلاپ D، خروجی آن در پالس بعدی ساعت معلوم می‌شود.

نوشتن معادله‌های حالت

$$A(t+1) = D_A = A(t)\,x(t) + B(t)\,x(t)$$

$$B(t+1) = D_B = A'(t)\,x(t)$$

- t زمان فعلی و t+1 زمان بعدی است. چون همه ورودی‌ها بر اساس زمان فعلی است معمولا t را نمی‌نویسیم.

$$A(t+1) = Ax + Bx$$

$$B(t+1) = A'x$$

- خروجی لحظه‌ای مدار بر اساس مقادیر حالت فعلی حساب می شود (نه بعدی).

$$y = (A + B)\,x'$$

کشیدن جدول حالت

- این مدار یک ورودی (x) یک خروجی (y) و دو حالت درونی (A و B) دارد (تفاوت جدول درستی و جدول حالت داشتن حالت‌های درونی در کنار ورودی و خروجی‌های عادی است).
- ورودی‌های رسمی در کنار حالت فعلی همگی ورودی حساب می‌شوند یعنی اینجا x در کنار حالت فعلی A و B نقش ورودی دارند (رسما فقط x ورودی است) و y (خروجی رسمی) در کنار حالت بعدی A_{t+1} و B_{t+1} نقش خروجی.
- مثل جدول درستی، در جدول حالت برای ورودی‌ها همه حالات را در نظر می‌گیریم (0 و 1 پرمی‌کنیم) و بعد خروجی‌ها را با فورمول حساب می‌کنیم(مانند مدار ترکیبی).
- پس، سمت چپ جدول ستون‌های ورودی را می‌کشیم و سمت راست را حساب و درج می‌کنیم.

کشیدن جدول حالت

$A(t + 1) = Ax + Bx$

$B(t + 1) = A'x$

$y = (A + B)\, x'$

Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

کشیدن جدول حالت

$A(t + 1) = Ax + Bx$

$B(t + 1) = A'x$

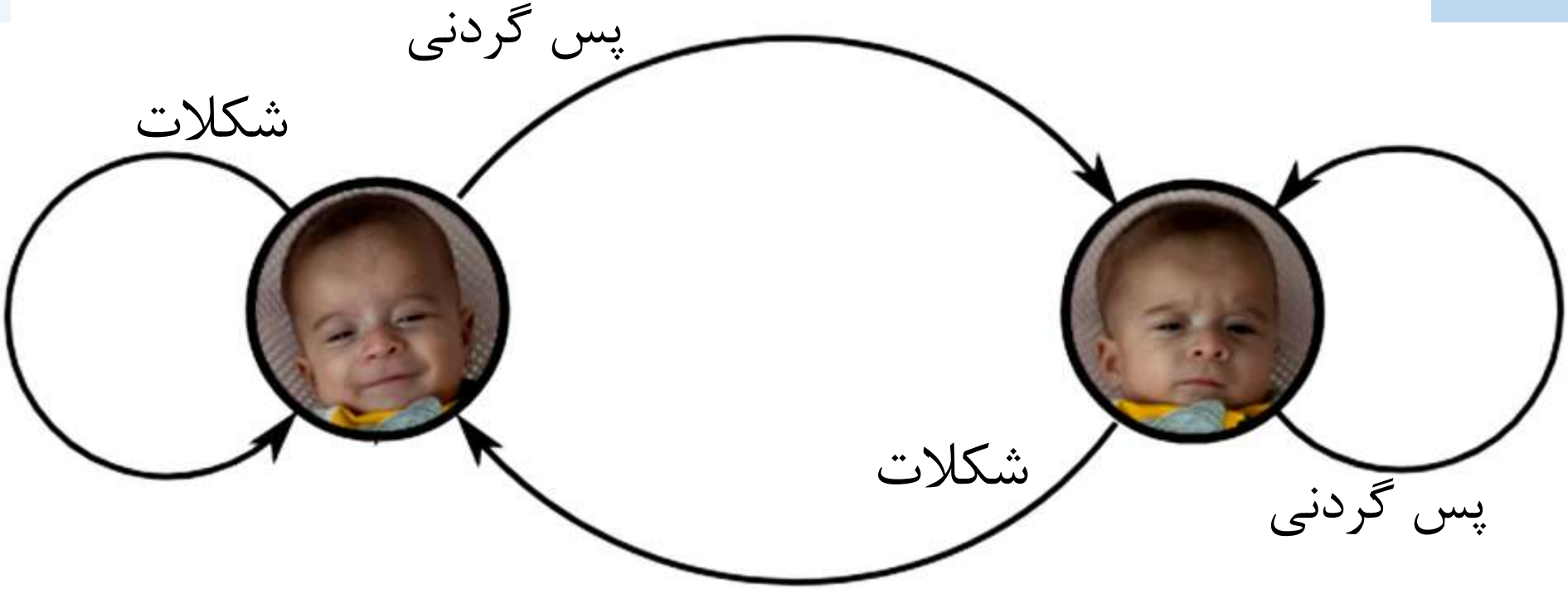
$y = (A + B)\, x'$

Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

آیا درک نحوه کار مدار از روی این جدول آسان است؟

ماشین حالت متناهی

- مدلی است که در آن تعداد معینی حالت وجود دارد و در شرایط مشخص از یک حالت به حالت دیگر منتقل می‌شود.



رسم نمودار حالت

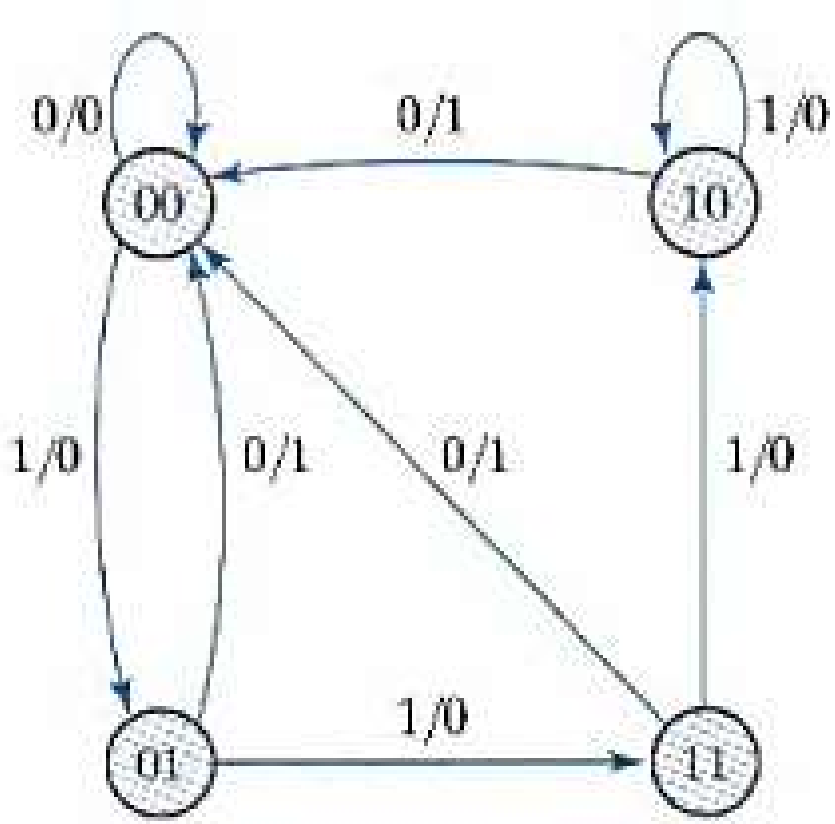
- برای مدار منطقی ترتیبی می‌توان ماشین حالت کشید.
- برای این کار هر گره نمودار بستگی به حالت تمامی فلیپ‌فلاپ‌ها خواهد داشت و هر لبه با توجه به ورودی‌ها منتقل می‌شود.
- پس برای همین مدار، نمودار حالت (از روی جدول حالت) به شکل ماشین حالت متناهی رسم می‌شود. دو رقم عدد داخل دایره‌ها (گره) حالت A,B هستند.
- ممکن است بعضی اوقات حالت بدون توجه به ورودی منتقل شود (یا اصلا مدار ورودی نداشته باشد) که در این صورت روی لبه چیزی نوشته نمی‌شود.

رسم نمودار حالت (ادامه مثال قبل)

Present State		Next State				Output	
		$x = 0$		$x = 1$		$x = 0$	$x = 1$
A	B	A	B	A	B	y	y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

- روی هر لبه، عدد قبل از / مقدار x و (همراه حالت فعلی) تعیین کننده حالت بعدی است. بعد از / مقدار y آمده که خروجی است (اثری هم در تعیین حالت بعد ندارد).

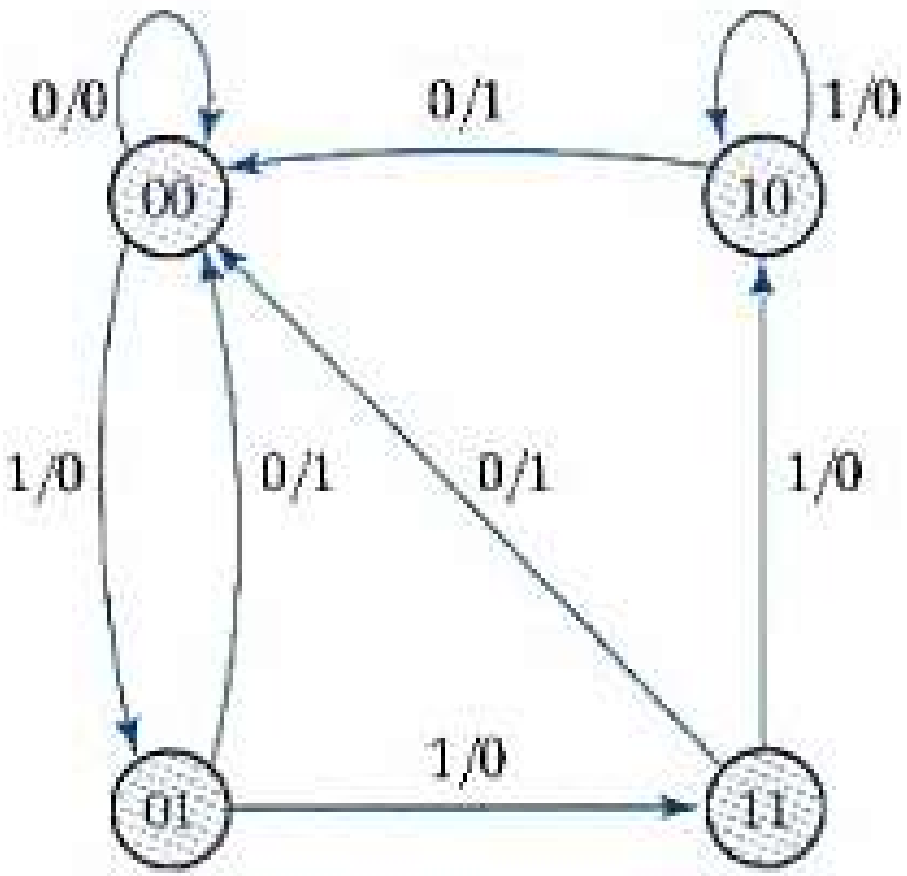
رسم نمودار حالت



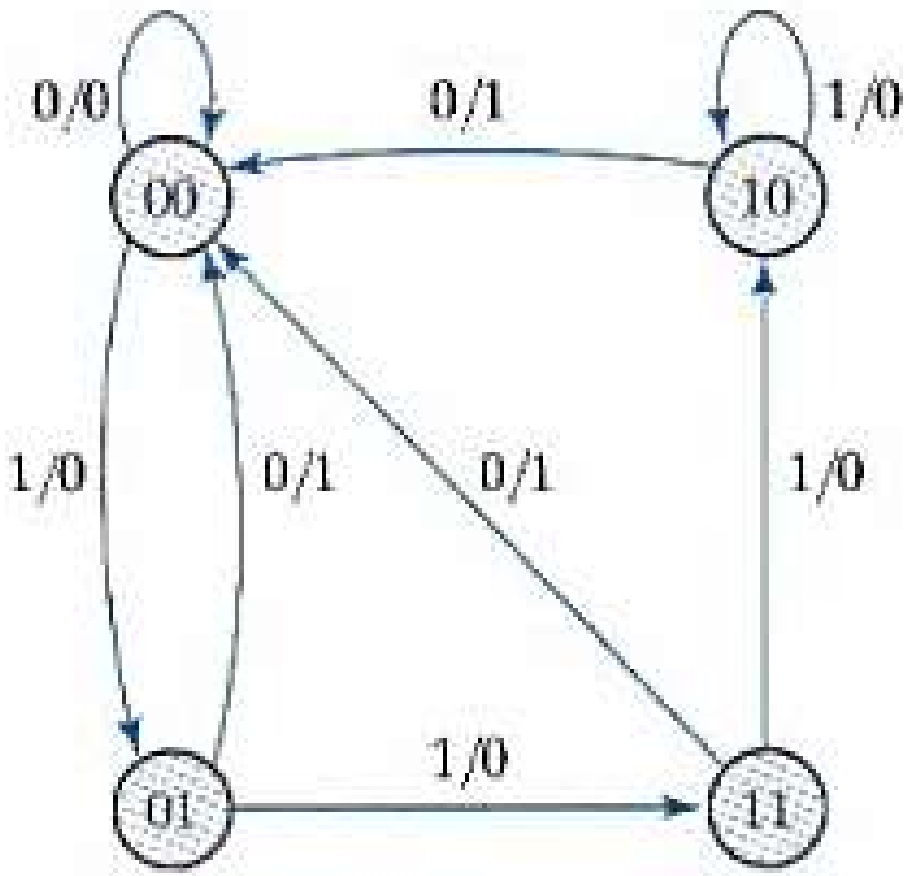
Present State		Next State				Output	
		x = 0		x = 1		x = 0	x = 1
A	B	A	B	A	B	y	y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

- روی هر لبه، عدد قبل از / مقدار x و (همراه حالت فعلی) تعیین کننده حالت بعدی است. بعد از / مقدار y آمده که خروجی است (اثری هم در تعیین حالت بعد ندارد).

تحلیل نهایی مدار



تحلیل نهایی مدار



۰ تا

بیش از ۲

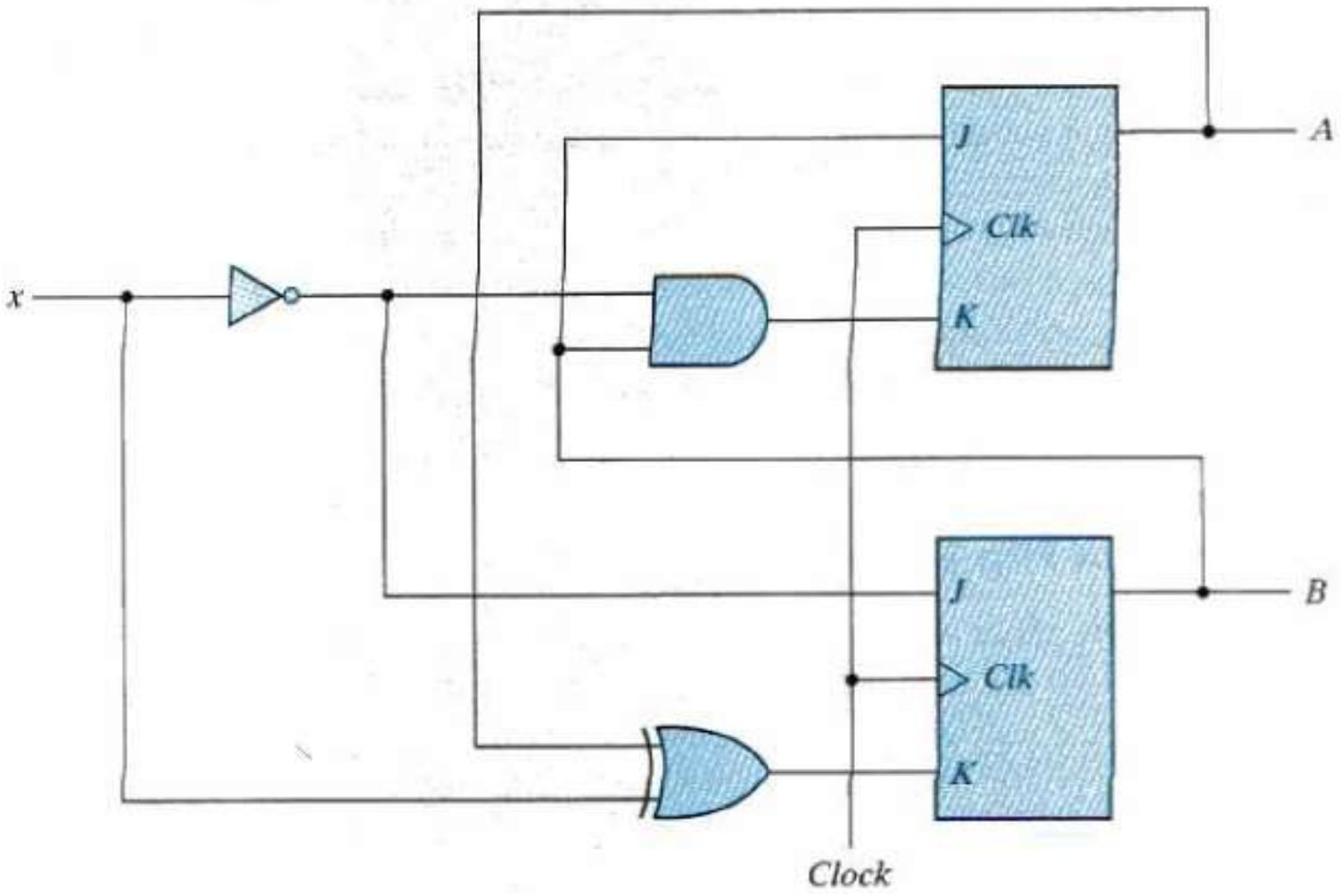
۱ تا

۲ تا

این مدار تعداد 1 های متوالی که بعد از 0 آمده را می‌شمارد که نتیجه: ۰ تا، ۱ تا، ۲ تا و بیش از ۲ تا است.

دومین مثال تحلیل

(اینجا خروجی رسمی نداریم و حالت بعدی
دو فلیپ‌فلاپ را خروجی می‌گیریم)



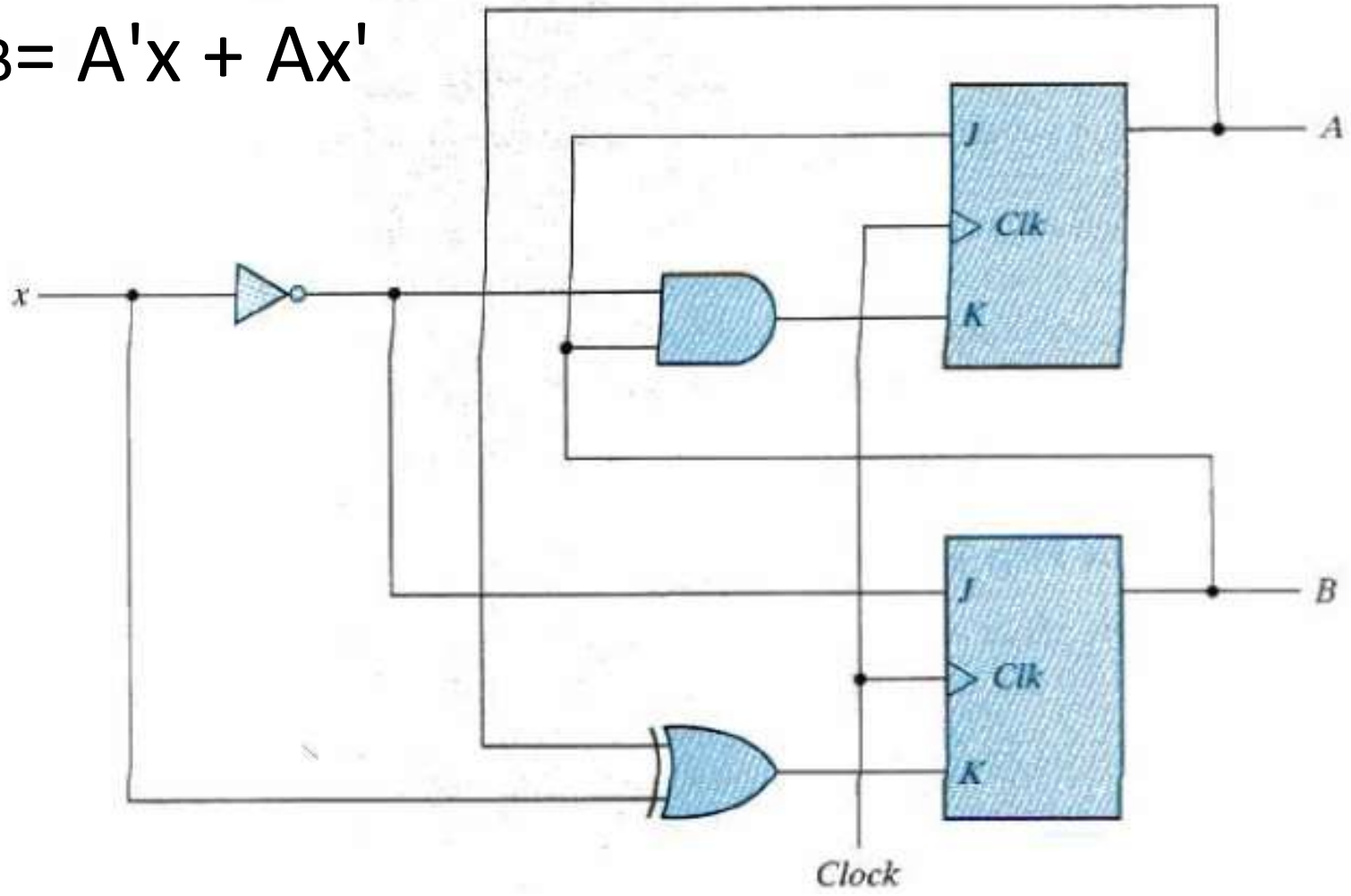
دومین مثال تحلیل

$J_A = B$

$K_A = Bx'$

$J_B = X'$

$K_B = A'x + Ax'$



$J_A = B$

$K_A = Bx'$

$J_B = X'$

$K_B = A'x + Ax'$

جدول حالت

Present State		Input	Next State		Flip-Flop Inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0						
0	0	1						
0	1	0						
0	1	1						
1	0	0						
1	0	1						
1	1	0						
1	1	1						

$J_A = B$

$K_A = Bx'$

$J_B = X'$

$K_B = A'x + Ax'$

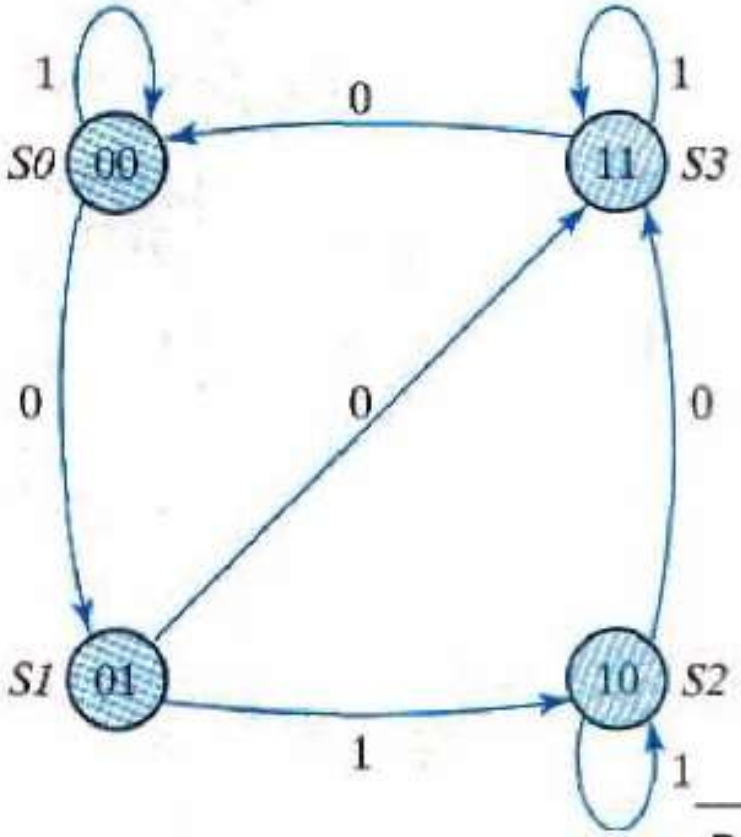
جدول حالت

Present State		Input	Next State		Flip-Flop Inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

تبدیل به دیاگرام

Present State		Input	Next State		Flip-Flop Inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

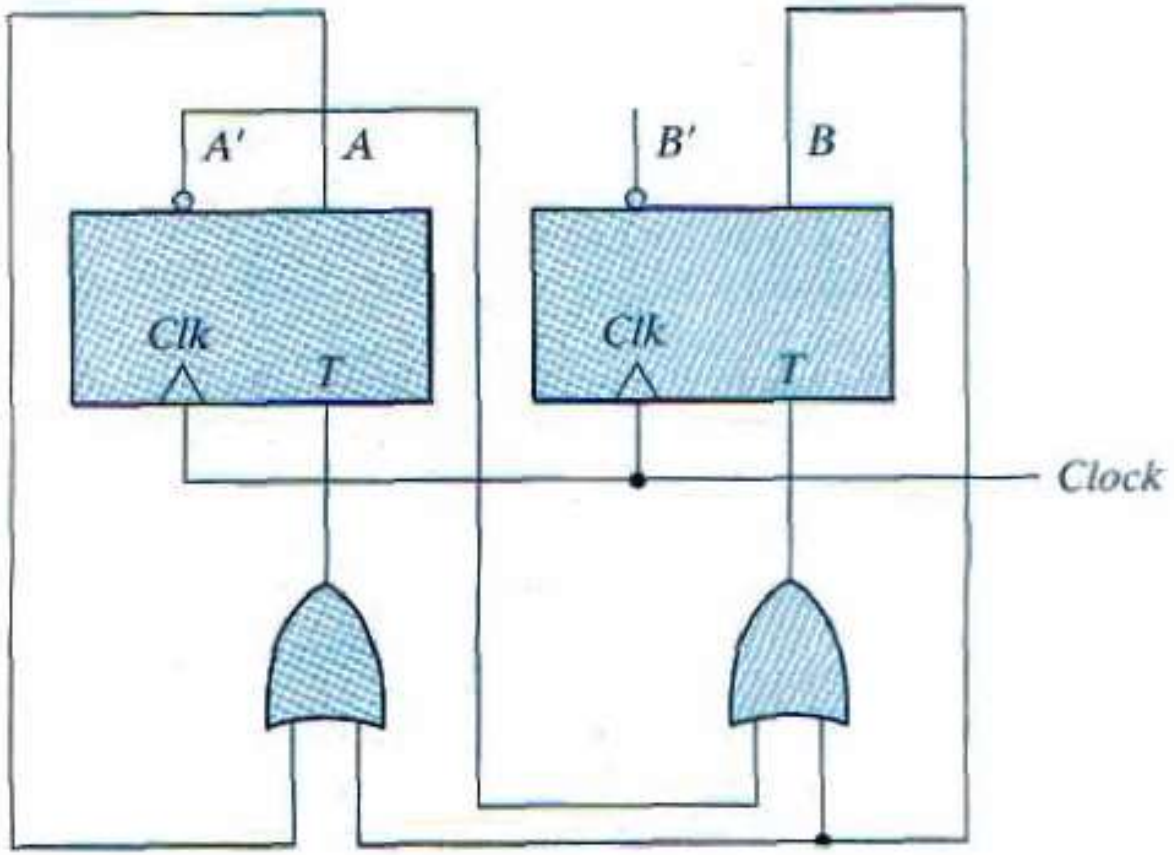
تبدیل به دیاگرام



Present State		Input	Next State		Flip-Flop Inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

تمرین

- برای مدار زیر جدول و دیاگرام حالت بکشید. این مدار ورودی و خروجی رسمی ندارد و فقط در هر پالس ساعت از حالتی به حالتی منتقل می‌شود (حالت بعدی هر فلیپ‌فلاپ A t+1 بر اساس مقدار فعلی آن A و ورودی TA تعیین می‌شود).



مدارهای منطقی

(طراحی سیستم‌های دیجیتال)

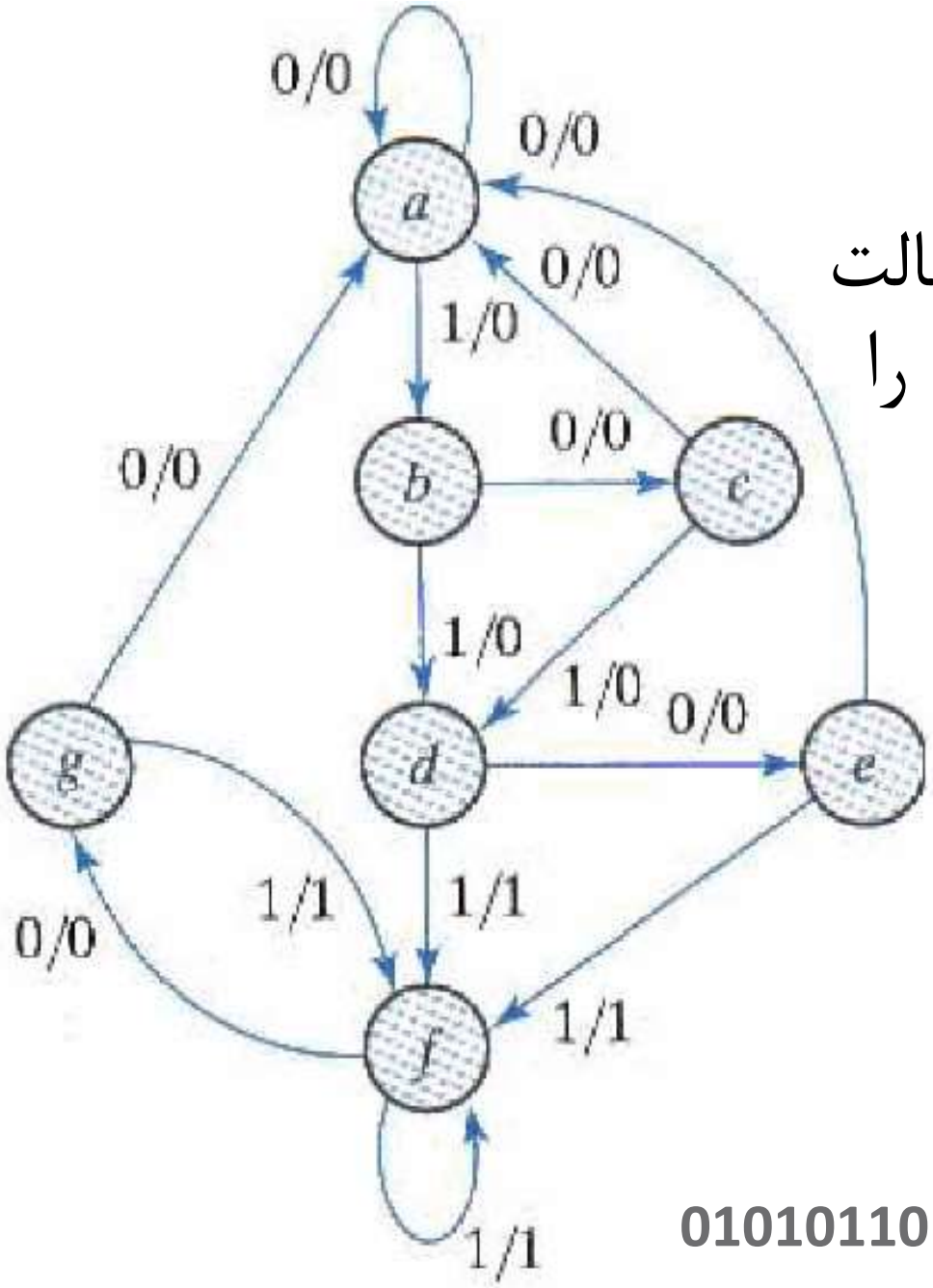
جلسه دهم

طراحی مدارهای ترتیبی ساعتدار

حالات مدار ترتیبی

- تعداد حالت‌های مدار ترتیبی از روی ماشین حالت که کار مدار را وصف می‌کند معلوم می‌شود (تعداد گره‌ها).
- تعداد فلیپ‌فلاپ‌ها به تعداد حالات وابسته است (وقتی 2^n حالت داشته باشیم به n فلیپ‌فلاپ نیاز داریم).
- از همین رو، با کاهش تعداد حالات (در صورت امکان) تعداد فلیپ‌فلاپ‌ها احتمالا کمتر شده و مدار ساده می‌شود.
- کاهش حالت فقط وقتی انجام می‌شود که صورت مساله از ما خواسته باشد و یا مطمئن باشیم که فقط خروجی‌های رسمی مدار مورد استفاده‌اند و خود حالت‌ها (خروجی فلیپ‌فلاپ‌ها) اهمیتی ندارند.

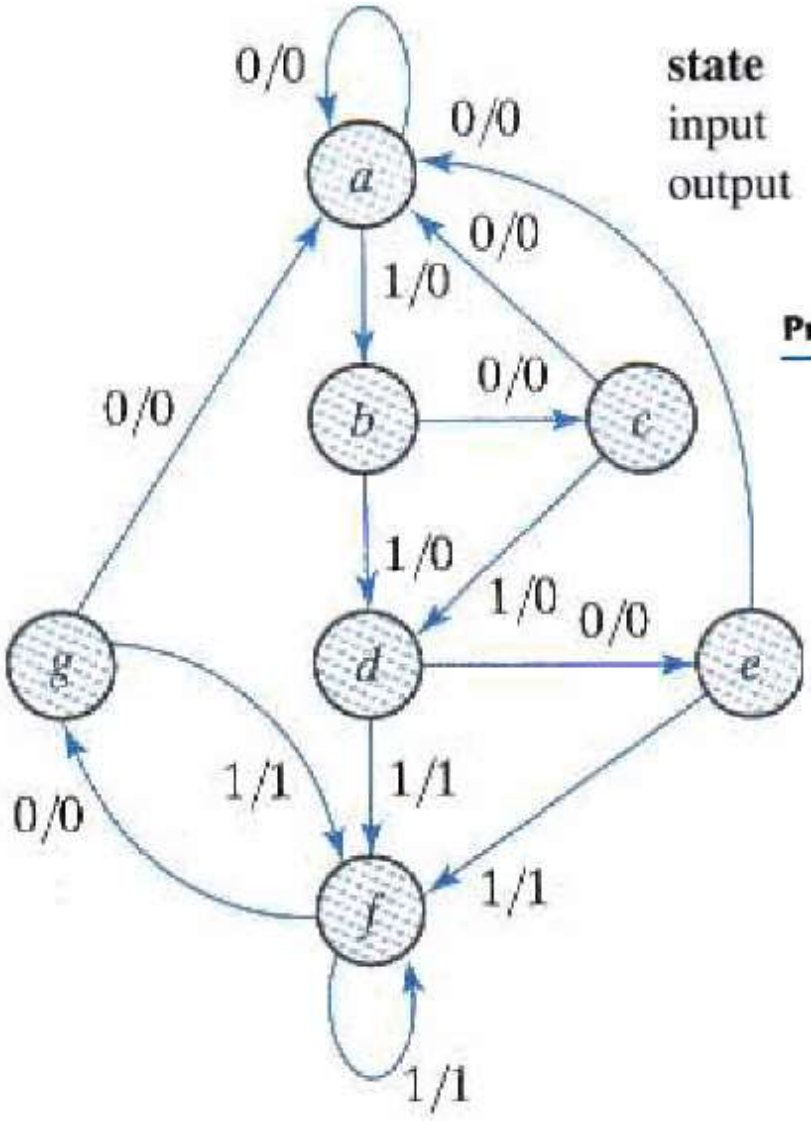
کاهش حالت



- می‌خواهیم دیاگرام حالت (ماشین حالت) مقابل را ساده کنیم.
- برای این کار جدول حالت را می‌کشیم.

پاسخ مدار به ورودی: 01010110100

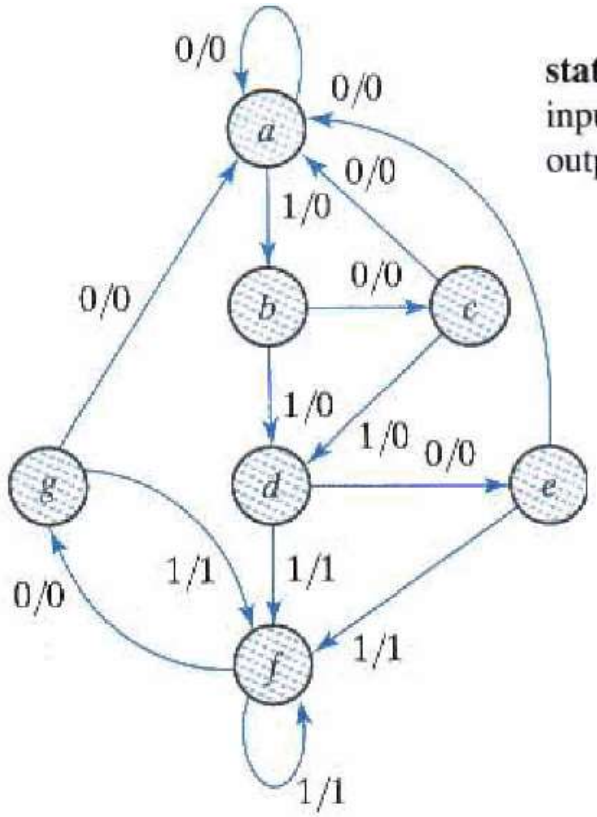
پاسخ مدار به ورودی: 01010110100



Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$

کاهش حالت

پاسخ مدار به ورودی: 01010110100



state	*a*	*a*	*b*	*c*	*d*	*e*	*f*	*f*	*g*	*f*	*g*	*a*
input	0	1	0	1	0	1	1	0	1	0	0	
output	0	0	0	0	0	1	1	0	1	0	0	

	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	*a*	*b*	0	0
b	*c*	*d*	0	0
c	*a*	*d*	0	0
d	*e*	*f*	0	1
e	*a*	*f*	0	1
f	*g*	*f*	0	1
g	*a*	*f*	0	1

- می‌بینید حالات e و g دقیقا دارند مثل هم عمل می‌کنند. بنابراین تغییری پیش نمی‌آید اگر این دو حالت را با هم یکی کنیم.

کاهش حالت

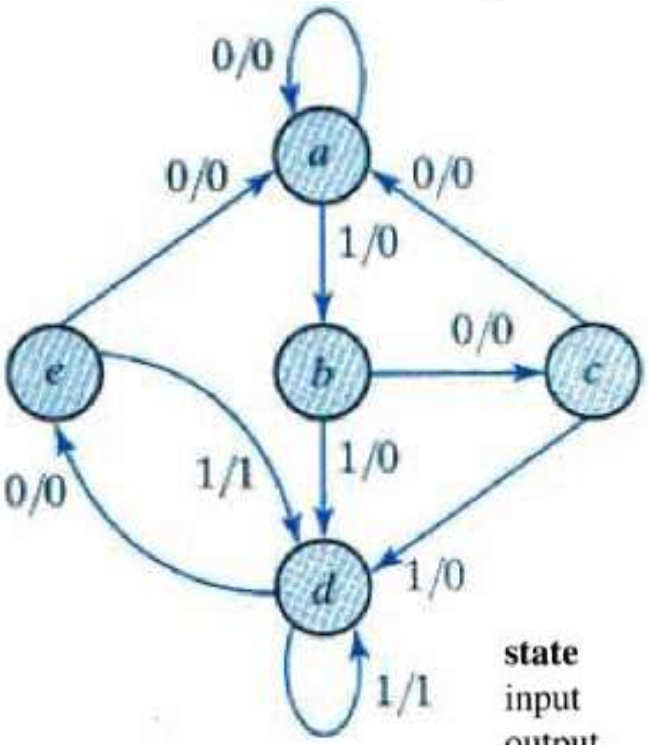
Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

- حالات از ۷ به ۶ کاهش یافت اما باز هم حالات d و f مثل هم شدند. آنها را هم ادغام می‌کنیم.

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

کاهش حالت

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1



پاسخ مدار به ورودی: 01010110100

state	a	a	b	c	d	e	d	d	e	d	e	a
input	0	1	0	1	0	1	1	0	1	0	0	
output	0	0	0	0	0	1	1	0	1	0	0	

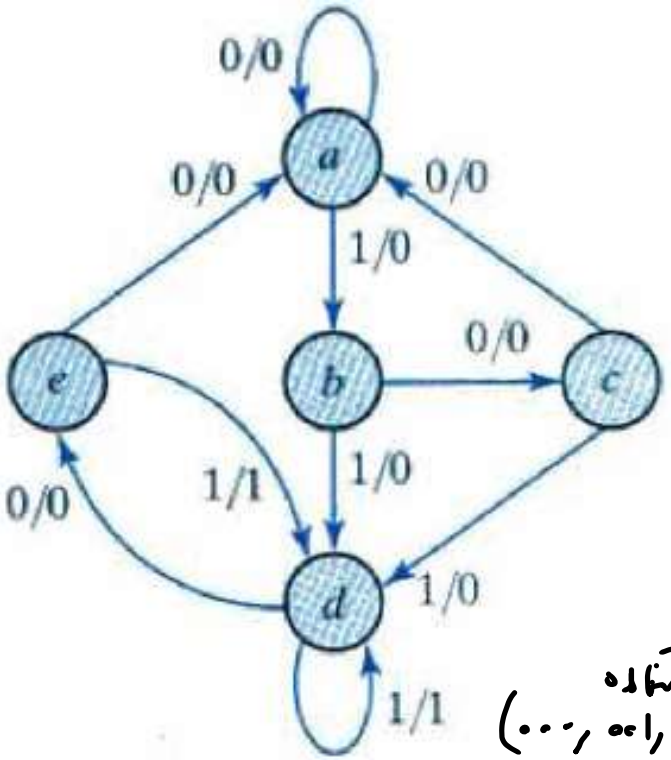
تخصیص حالت

- حالا وقت آن است که به هرکدام از حالات مدار (گره‌هایی که تابحال a,b,c,d,... نامیده می‌شدند)، یک کد (دودویی و دلخواه) اختصاص دهیم.
- تعداد فلیپ‌فلاپ‌ها از تعداد حالت‌ها بدست می‌آید و ممکن است تعدادی هم حالت بلااستفاده داشته باشیم. مثلا در مثال قبل ۵ که حالت داریم که با ۳ فلیپ‌فلاپ قابل پیاده‌سازی است (۳ فلیپ‌فلاپ ۸ حالت می‌دهند).
- ترتیب حالت‌ها را معمولا به همان ترتیبی که اتفاق می‌افتند کدگذاری می‌کنیم (الفبایی هم همینطور بود).
- برای این کار می‌توان از کدهای مختلف استفاده کرد.

روش‌های مختلف تخصیص حالت

- کدباینری: حالت اول (a) را 000 باینری می‌گذاریم و حالت بعد را 001 و الی آخر.
توجه: اگر حالات بلااستفاده داشته باشیم، بهتر است در تخصیص کد باینری بی‌استفاده‌ها را اول قرار دهیم چون موقع ساده کردن با کارنو هر چه جدول پر 1 تر باشد بهتر است.
- کد گری: همانطور که از قبل با آن آشناییم. در هر تغییر حالت (معمولا) فقط یک فلیپ‌فلاپ تغییر می‌کند.
- کد یکی روشن: کد دیگری است که به تعداد حالات فلیپ‌فلاپ می‌خواهد.
- هیچ تضمینی نیست که استفاده از این روش‌های کدینگ مدار نهایی را ساده‌تر کند.

تخصیص حالت



شاید این بهتر باشد
(بی استفاده
۰۰، ۰۱، ۱۰، ۱۱)

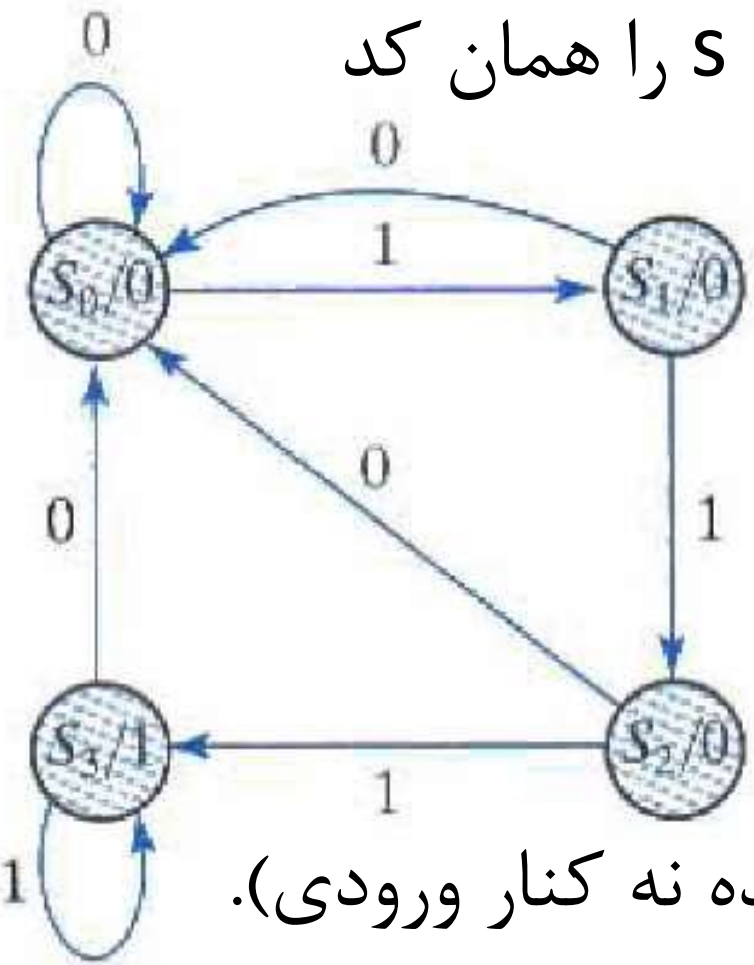
State	Assignment 1, Binary	Assignment 2, Gray Code	Assignment 3, One-Hot
a	000 ۱۱۱	000	00001
b	001 ۱۰۰	001	00010
c	010 ۱۰۱	011	00100
d	011 ۱۱۰	010	01000
e	100 ۱۱۱	110	10000

روال طراحی مدارهای ترتیبی

1. اگر در شروع کار دیاگرام نداشته باشیم، با توجه به توصیفی که از مدار شده و درکی که از کار مدار داریم دیاگرام (ماشین) حالت را رسم می‌کنیم.
2. اگر شد و لازم بود، تعداد حالت‌ها را کاهش می‌دهیم (اگر غیر از خروجی، خود حالت هم برای ما مهم باشد یا اصلا خروجی نداشته باشیم، لازم نیست ساده کنیم).
3. از روی تعداد حالات تعداد فلیپ‌فلاپ‌ها را معلوم کرده به هر حالت یک مقدار دودویی تخصیص می‌دهیم.
4. با توجه به نوع فلیپ‌فلاپ مورد نظرمان (معمولا D مگر آنکه صورت مساله چیز دیگری بگوید) جدول حالت را می‌کشیم.
5. فورمول ورودی فلیپ‌فلاپ‌ها و خروجی‌ها را تعیین کرده، ساده می‌کنیم.
6. مدار را می‌کشیم.

مثال طراحی کامل

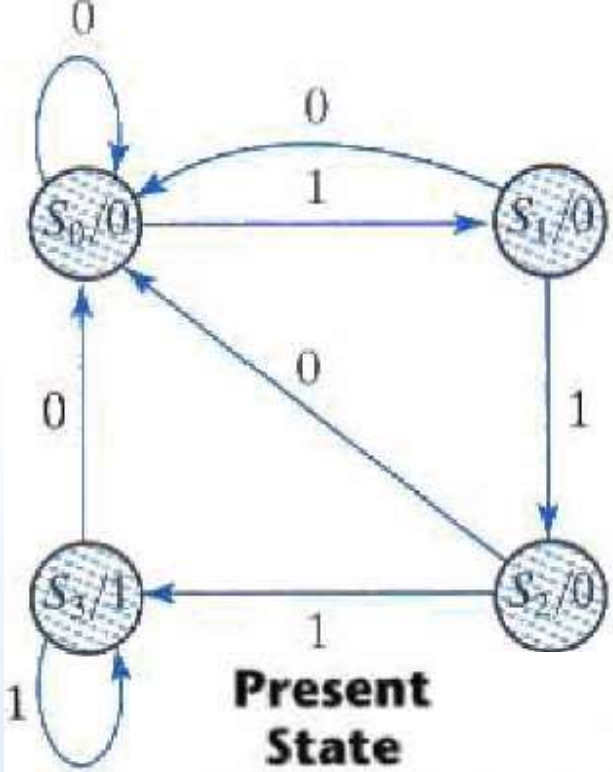
- می‌خواهیم مداری رسم کنیم که کار نمودار حالت زیر را انجام دهد (عدد اندیس s را همان کد باینری حالت می‌گیریم و بعد از ممیز خروجی است)



توجه: در این مدار خروجی **فقط به حالت بستگی** دارد و به ورودی بستگی مستقیم ندارد (به همین خاطر خروجی کنار حالت نوشته شده نه کنار ورودی).

جدول حالت

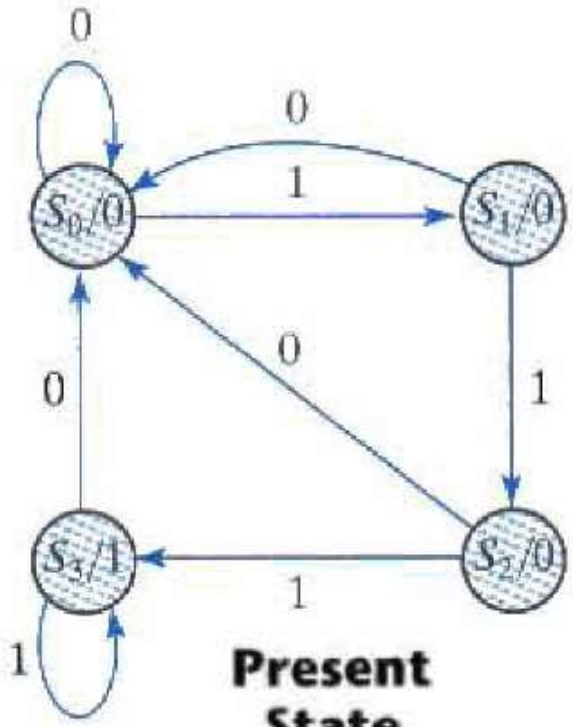
- در جدول ابتدا ورودی‌ها را نوشته و بعد خروجی‌ها را از روی دیاگرام پر می‌کنیم.



Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

جدول حالت

- در جدول ابتدا ورودی‌ها را نوشته و بعد خروجی‌ها را از روی دیاگرام پر می‌کنیم.



Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

ورودی فلیپ‌فلاپ‌ها

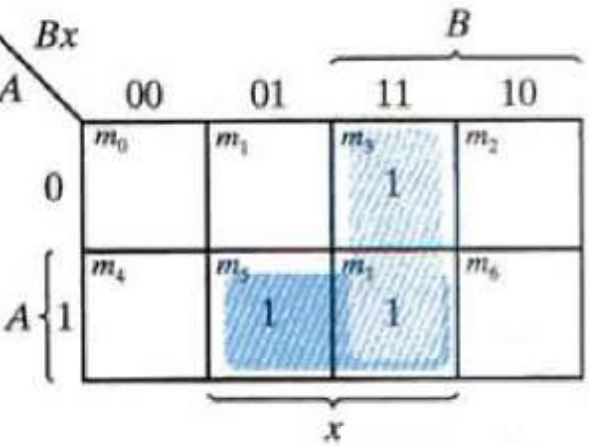
- برای هر خروجی یک کارنو می‌کشیم.

Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

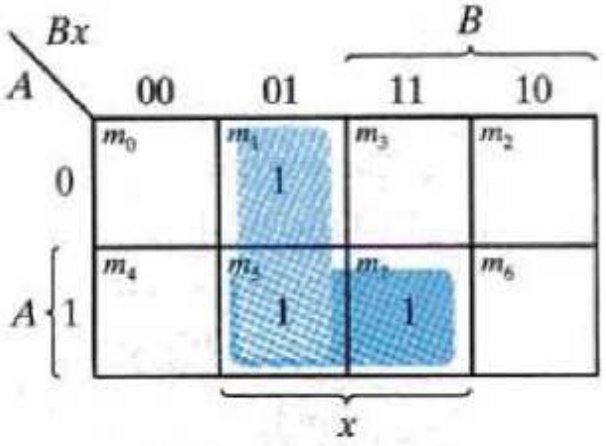
ورودی فلیپ‌فلاپ‌ها

- برای هر خروجی یک کارنو می‌کشیم.

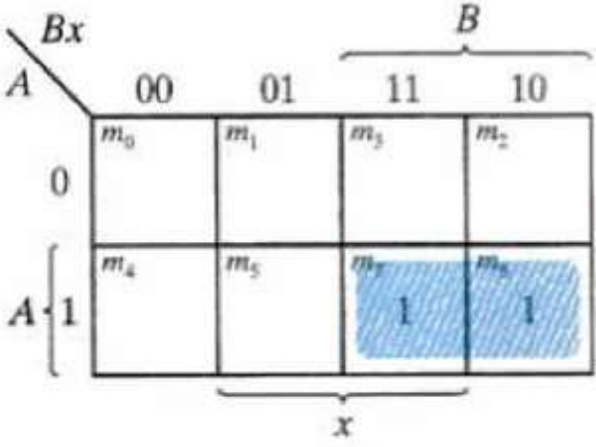
Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1



$D_A = Ax + Bx$



$D_B = Ax + B'x$



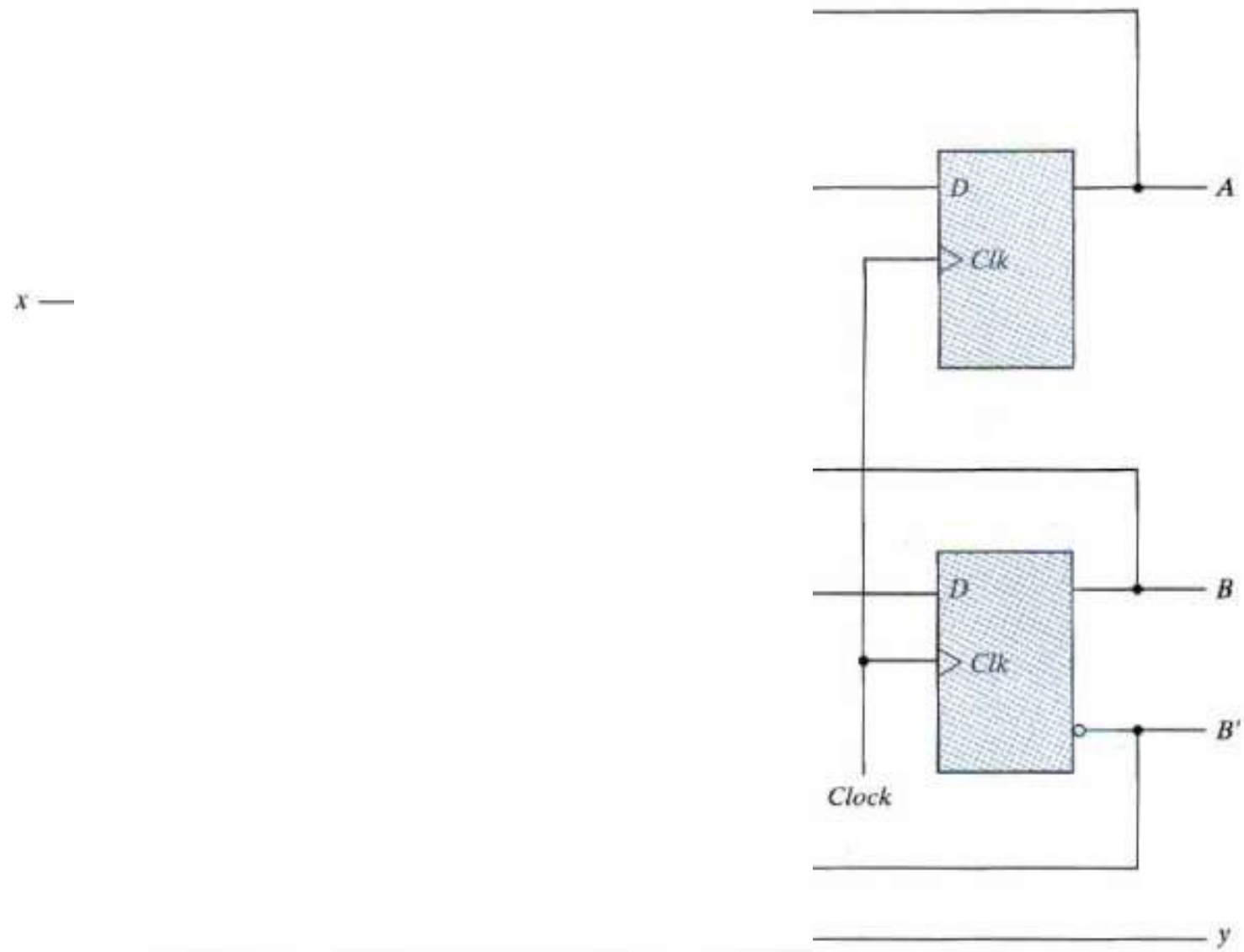
$y = AB$

رسم مدار

$D_A = Ax + Bx$

$D_B = Ax + B'x$

$y = AB$

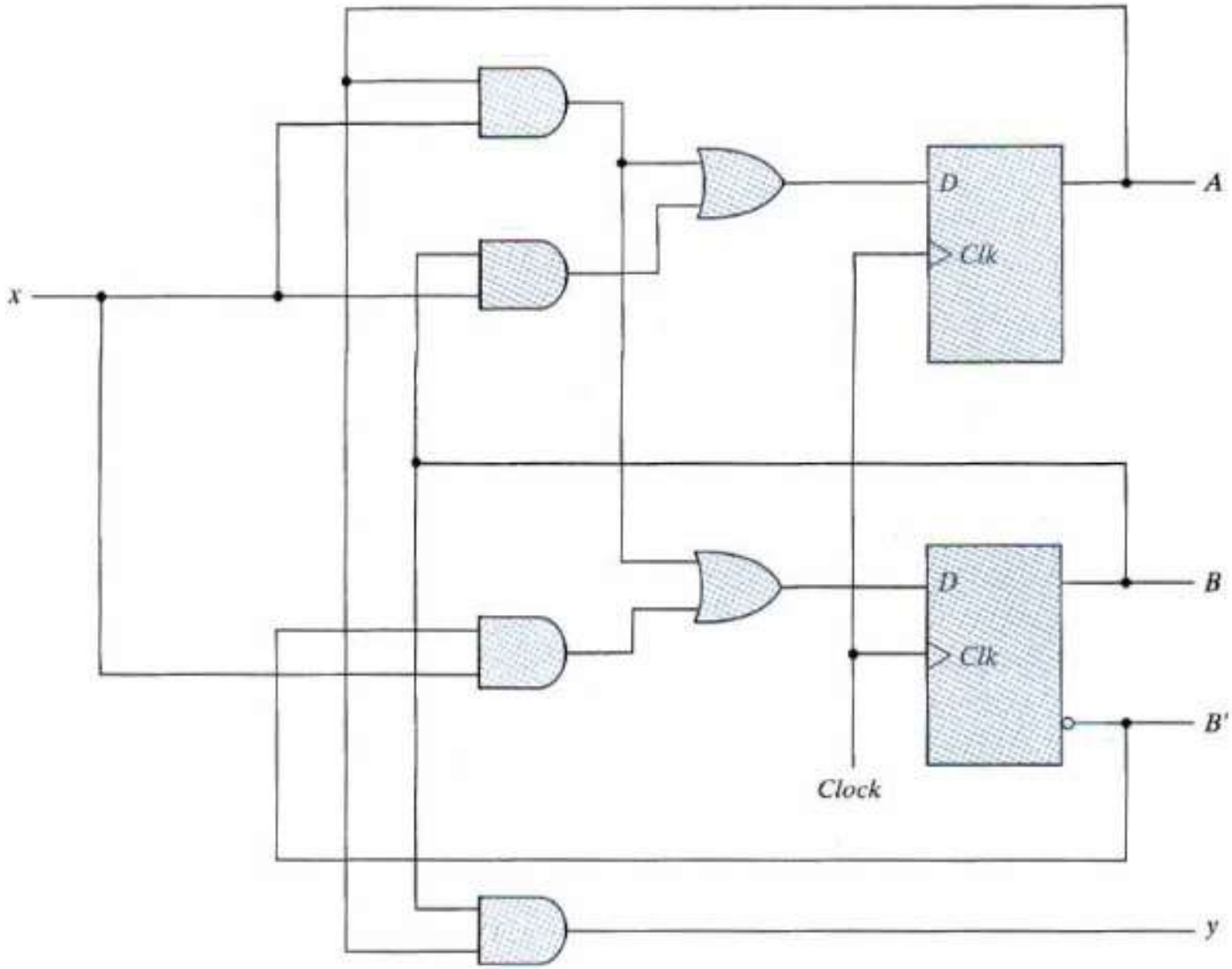


رسم مدار

$D_A = Ax + Bx$

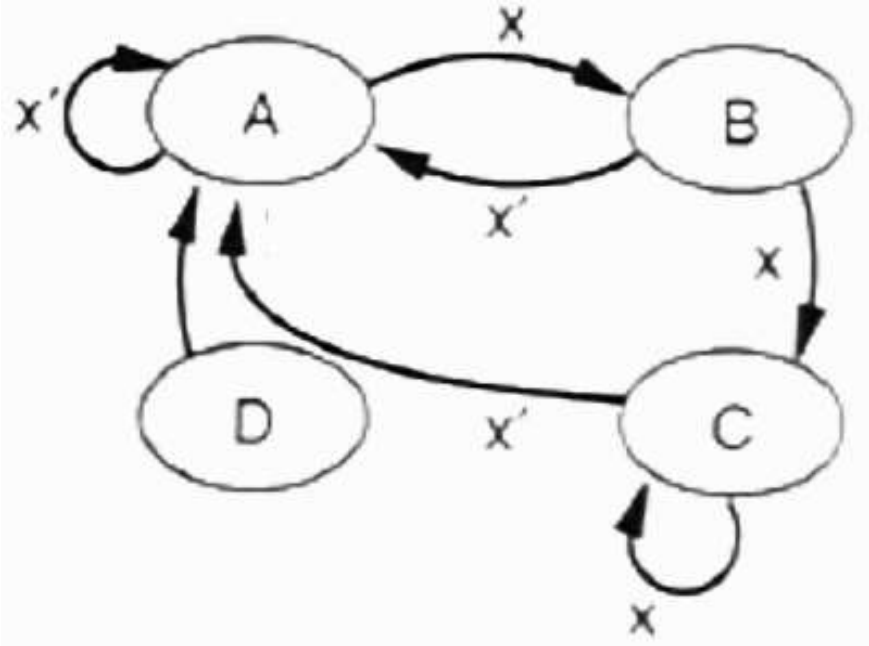
$D_B = Ax + B'x$

$y = AB$



تمرین

- با استفاده از فلیپ‌فلاپ‌های D و گیت‌های منطقی مداری ترتیبی طراحی کنید که با توجه به تنها ورودی x مطابق ماشین حالت شکل زیر تغییر حالت دهد.



مدارهای منطقی

(طراحی سیستم‌های دیجیتال)

جلسه یازدهم

مدارات ترتیبی پرکاربرد

مدارهای ترتیبی سنکرون پرکاربرد

- قبلا دیدیم که با المان‌های پایه ترکیبی (گیت‌ها) چگونه می‌توان مدارات پرکاربردی مثل انکدر و مالتی‌پلکسر و جمع کننده ساخت.
- در اینجا هم با المان‌های پایه ترتیبی (فلیپ‌فلاپ‌ها) می‌توانیم مدارات پرکاربرد و اساسی بسازیم از قبیل: رجیستر، شیفت رجیستر و شمارنده. انتقال موازی و سریال داده را نیز بررسی خواهیم کرد.
- آشنایی با مدارات پرکاربرد (هم ترکیبی و هم ترتیبی) برای شروع درس معماری کامپیوتر ضروری است و بدون آن نمی‌توان چیزی از این درس فهمید.

محل عملوند اول
محل عملوند دوم

مدار ماشین حساب

رجیسترها

- در سیستم‌های دیجیتال، رجیستر محلی است که در آن داده ذخیره می‌شود، اما حافظه رسمی (آدرس‌پذیر) نیست.
- در حافظه (اصلی) تعداد خانه‌های قابل استفاده بسیار زیاد است و دسترسی به آنها نیاز به آدرس‌دهی دارد.
- در مقابل تعداد رجیسترها (در کامپیوتر یا سیستم دیجیتال) کم است ولی دسترسی به آنها بصورت مستقیم است. برخی رجیسترها هم ممکن است کار خاصی داشته باشند. مثلا عملوندهای جمع در آنها ذخیره شود (در مدار ماشین حساب).

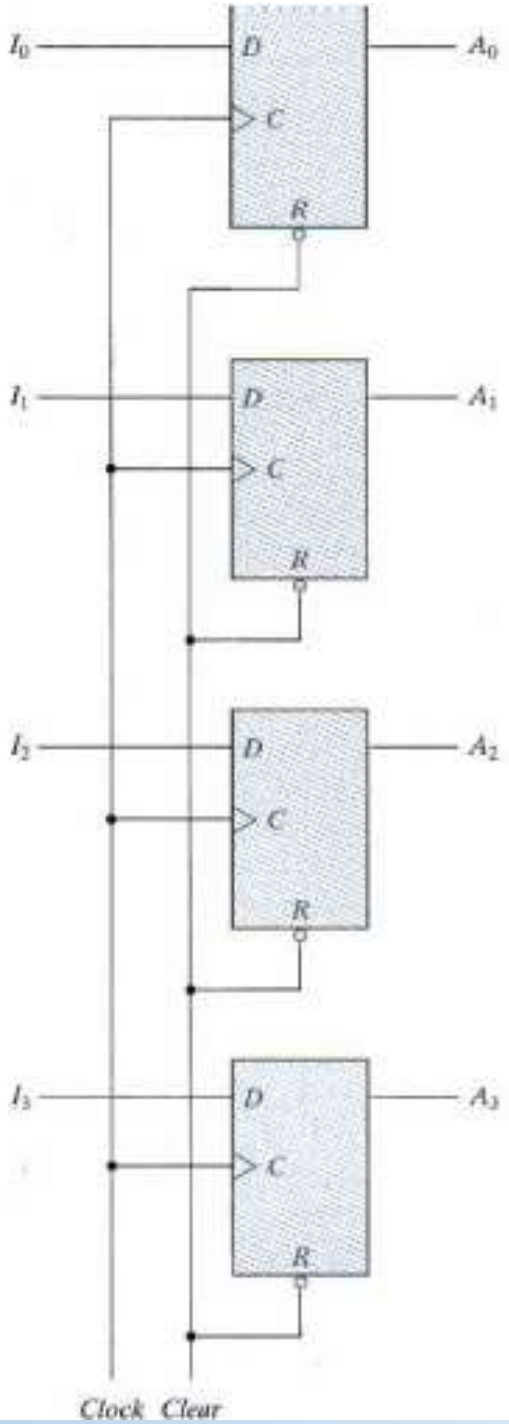
معادل فارسی رجیستر

- معنای لغوی. ثبات یک غلط مصطلح در فارسی! رجیستر در کامپیوتر چیزی را ثبت نمی‌کند. بلکه خودش محل ثبت است.

انگلیسی	نقش	عربی	فارسی
Registration	مصدر	ثبت	ثبت کردن
Registrar	اسم فاعل	ثبات	ثبات/ثبت کننده
Register	اسم مکان	سجل	دفتر ثبت/ثبات؟!

رجیستر

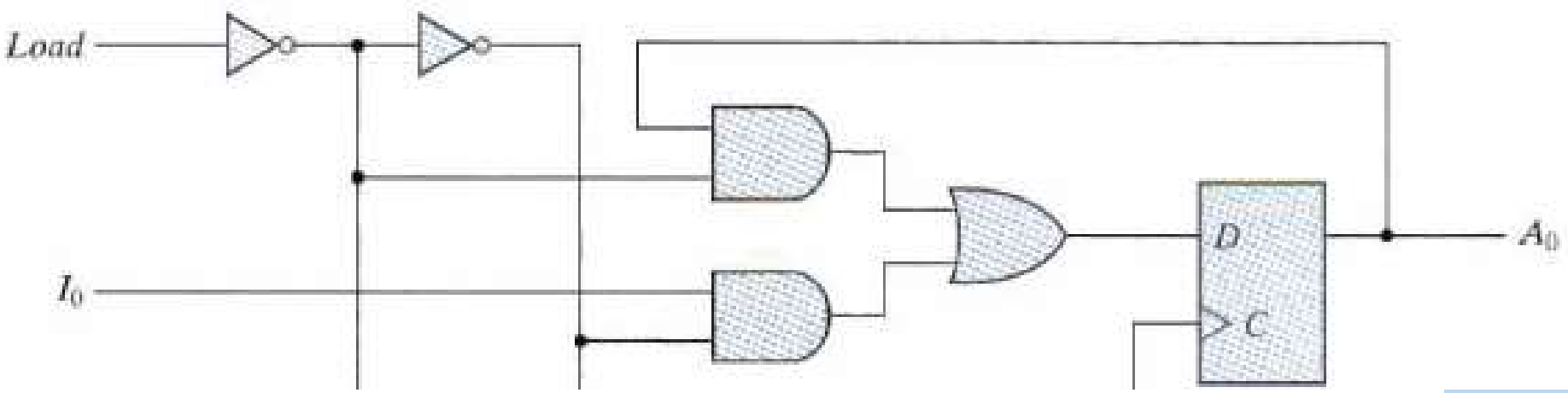
- رجیستر یک مدار ترتیبی ساعت‌دار است که از تعدادی فلیپ‌فلاپ (گاهی همراه با تعدادی گیت ترکیبی) ساخته شده است.
- هر فلیپ‌فلاپ یک بیت از اطلاعات را در خود ذخیره می‌کند و کل رجیستر یک مجموعه چند بیتی (مثلا ۸ یا ۱۶ بیتی) است.

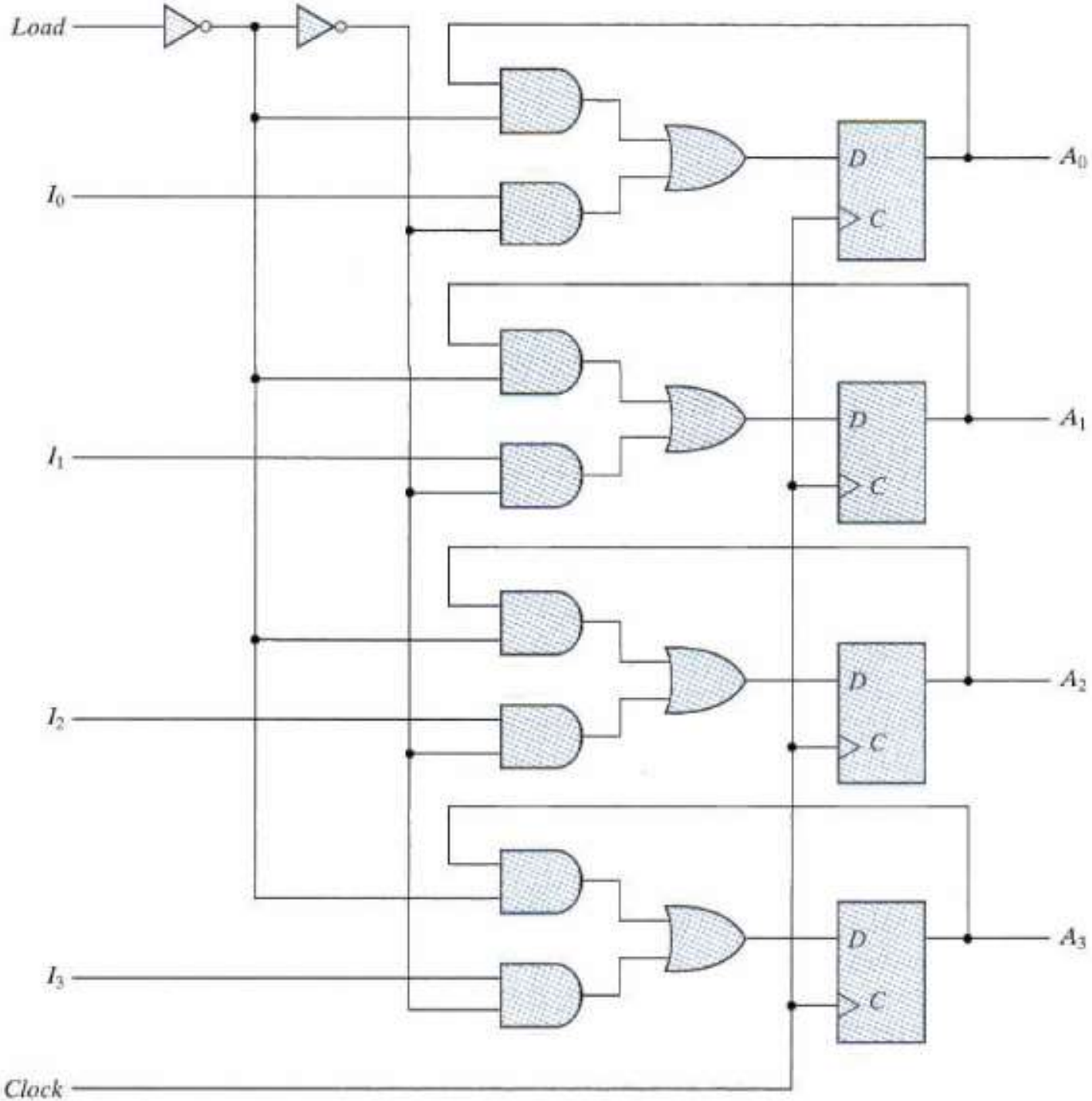


- این نمونه بسیار ساده‌ای از رجیستر ۴ بیتی است که تنها از چهار فلیپ‌فلاپ D دارای ورودی ریست و بدون گیت اضافه تشکیل شده و دیتا را برای یک پالس ساعت نگه می‌دارد.
- با رسیدن هر پالس ساعت مقدار چهار ورودی خوانده می‌شود و به خروجی انتقال می‌یابد. ورودی Clear تمام خروجی‌ها را صفر می‌کند (بدون در نظر گرفتن وضعیت ساعت)
- برای نگهداری دیتا کافیست جلوی رسیدن پالس ساعت گرفته شود.

رجیستر با امکان بارگیری موازی

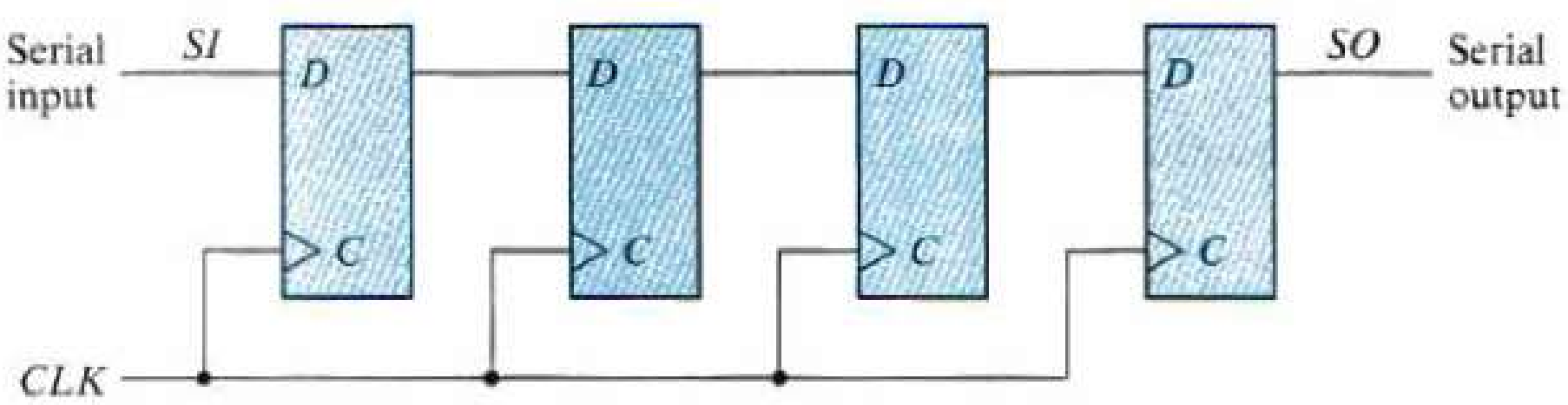
- حال یک نمونه کامل‌تر از رجیستر ۴بیتی را می‌بینیم.
- در چنین رجیستری ساعت همیشه به فلیپ‌فلاپ‌ها متصل است و تمام بیت‌ها در لبه پالس ساعت در رجیستر بارگیری می‌شوند.
- یک ورودی داریم بنام ورودی Load "بارگیری" وقتی این ورودی فعال باشد دیتاهای ورودی با رسیدن پالس ساعت داخل فلیپ‌فلاپ‌ها بارگیری می‌شوند. وقتی ورودی "بارگیری" غیر فعال باشد آنها محتوای خود را حفظ می‌کنند و به ورودی کاری ندارند.


Load
I_0
D
C
A_0


Load
I_0
I_1
I_2
I_3
Clock
D
C
A_0
A_1
A_2
A_3

شیفت رجیستر

- رجیستری که بتواند در هر پالس ساعت دیتای خودش را، یک بیت به چپ (یا راست) منتقل کند، شیفت رجیستر است.
- ورودی یا خروجی را می‌توان به صورت سریال (هر پالس ساعت یک بیت) با شیفت رجیستر، به درون رجیستر منتقل کرد (خواند). در زیر نمونه ساده‌ای را می‌بینید. با قطع کردن پالس بارگیری متوقف می‌شود.



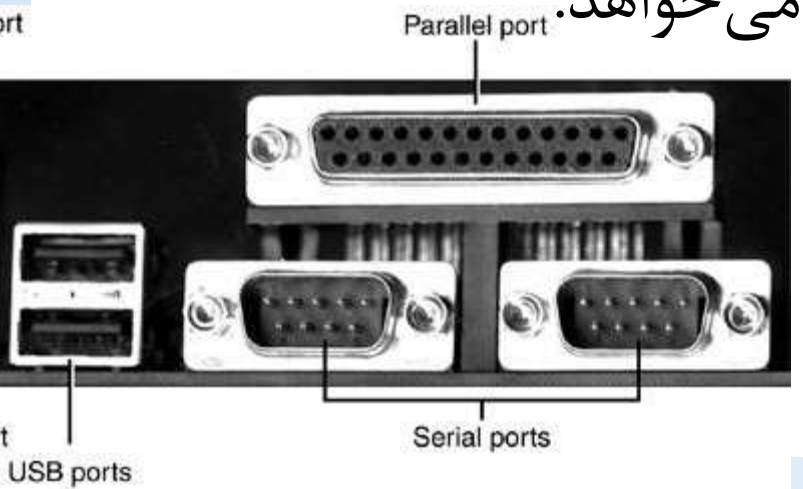
انتقال داده

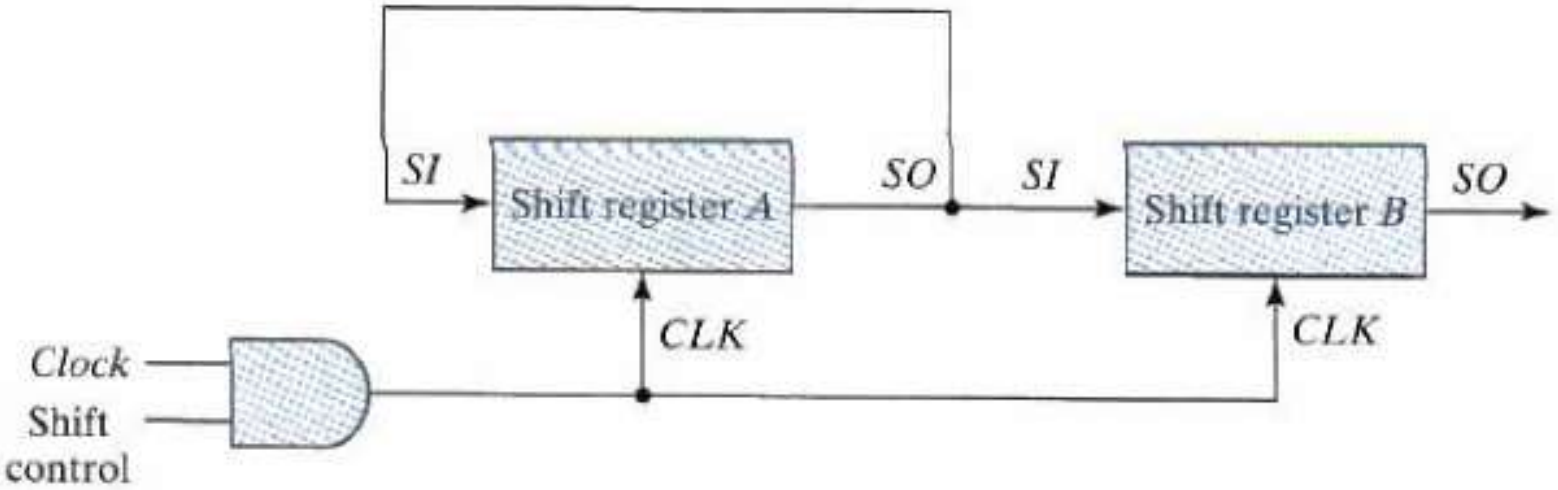
- انتقال داده از جایی به جای دیگر (در واقع بارگذاری داده درون رجیستر) یکی از کارهای مهمی است که هم درون کامپیوتر و سیستم‌های دیجیتال و هم برای ارتباط دو دستگاه مستقل با هم، به آن نیاز داریم.
- همانطور که دیدیم، اگر در این انتقال چند بیت (معمولا ۸، ۱۶، ۳۲ یا ۶۴) با هم (در یک پالس) منتقل شوند انتقال موازی است و اگر این انتقال یک بیت یک بیت صورت گیرد انتقال سریال است.

رجیسترهایی (مثل دو اسلاید قبل) انتقال داده‌شان موازی است.

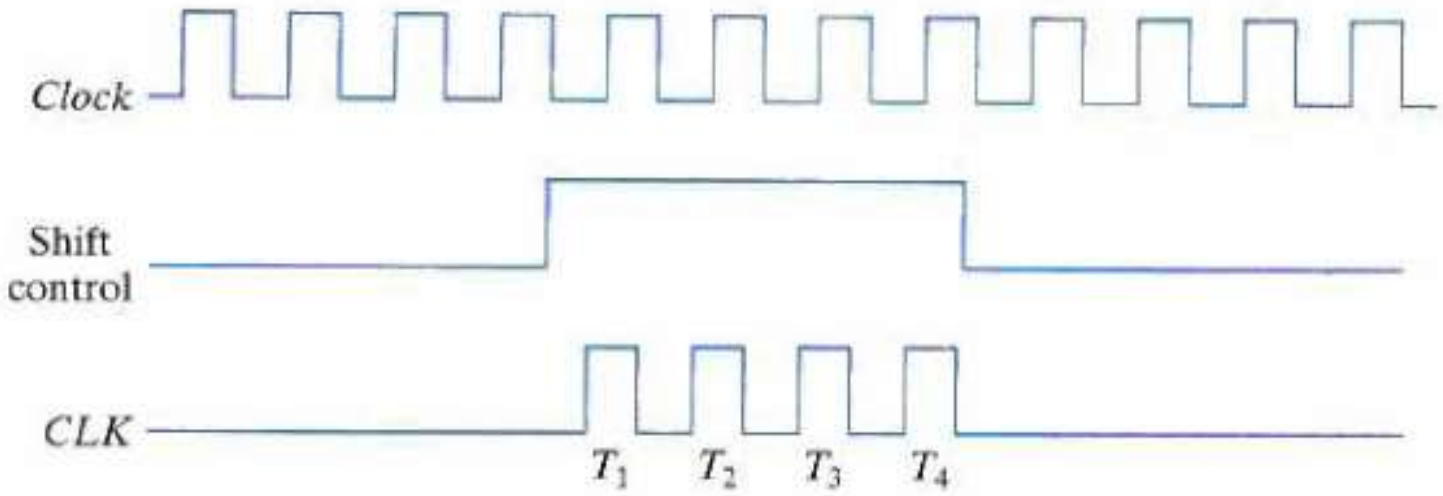
رجیستر با امکان بارگیری سریال

- در شکل بعد، ۴ بیت اطلاعات در رجیستر A وجود دارد که باید به رجیستر B منتقل شود.
- ورودی "کنترل شیفت" کمک می‌کند که تنها چهار پالس ساعت به رجیسترهای ورودی و خروجی برسد.
- فیدبکی که از خروجی A به ورودی آن وجود دارد باعث می‌شود پس از ۴ پالس، رجیستر A همان مقدار اولیه خود را داشته باشد.
- خروجی سریال SO و ورودی سریال SI نامیده می‌شود.
- در انتقال پارالل کل انتقال در یک پالس ساعت انجام می‌شود، در عوض انتقال سریال فقط یک خط می‌خواهد.





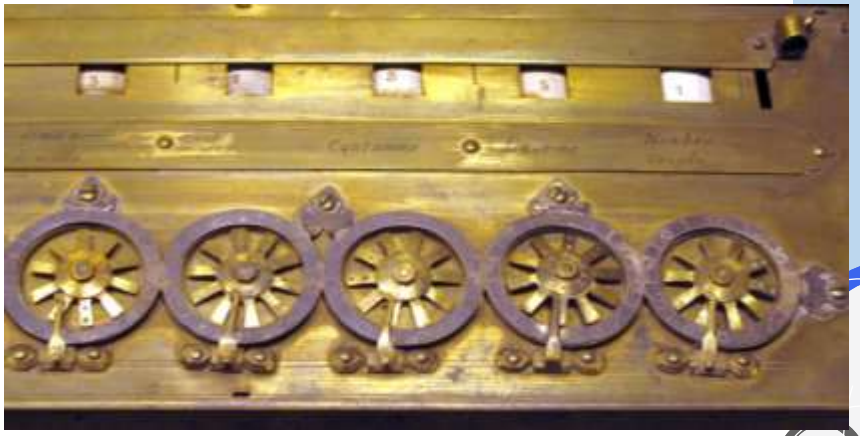
(a) Block diagram



(b) Timing diagram

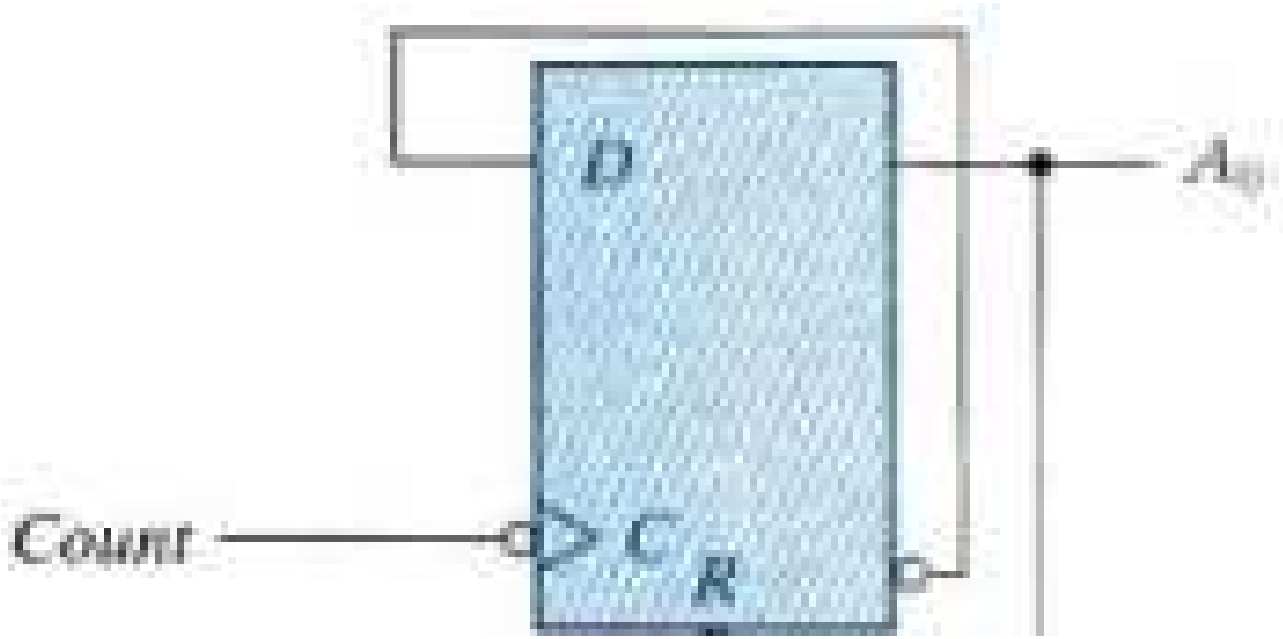
شمارنده

- رجیستری که با رسیدن پالس‌های ساعت (بدون اینکه ورودی دیگری داشته باشد)، به ترتیب خاصی تغییر حالت می‌دهد، شمارنده نام دارد.

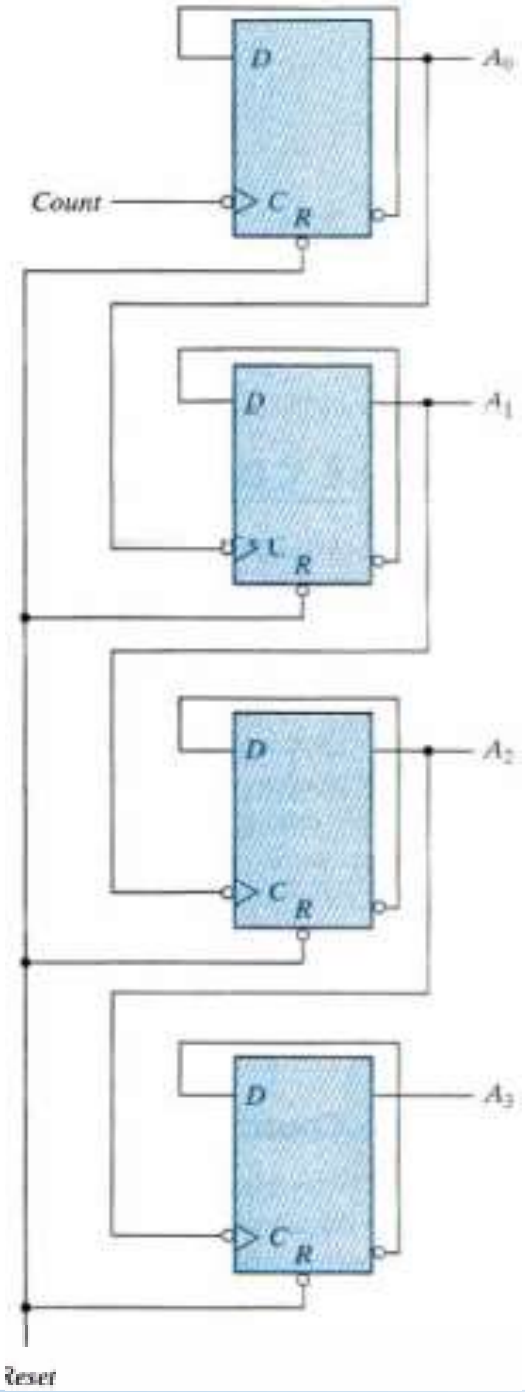


شمارنده‌ها دو نوعند: موجی و سنکرون (در شمارنده موجی پالس ساعت مشترک فقط به یک یا برخی فلیپ‌فلاپ‌ها وارد می‌شود بنابراین تعریف طبق ترتیبی هست اما سنکرون نیست).

شمارنده موجی



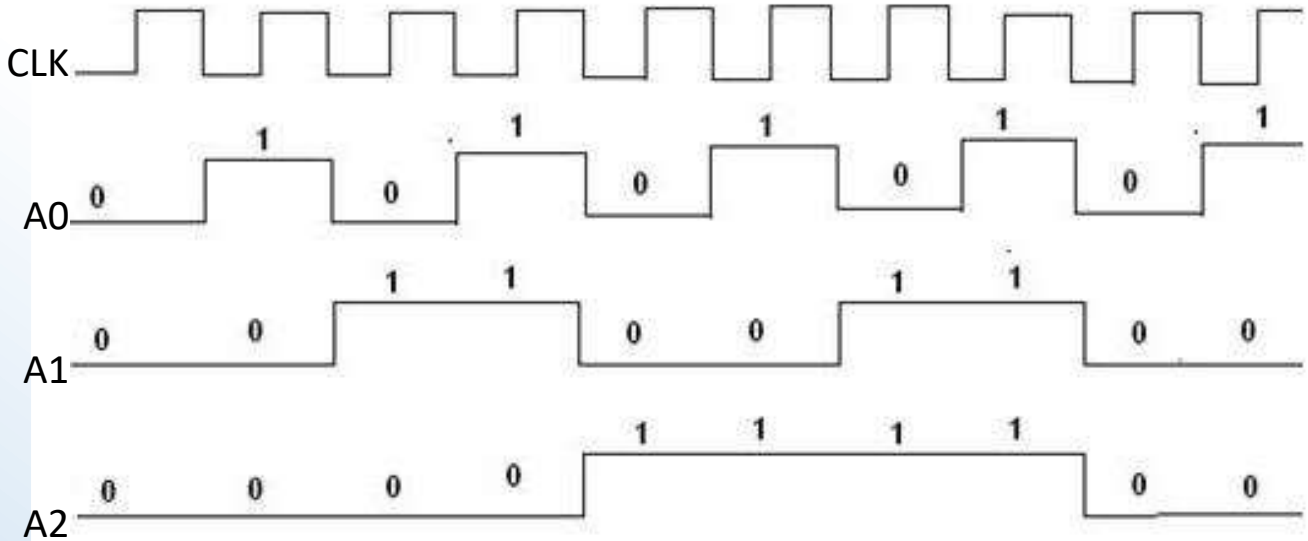
شمارنده موجی

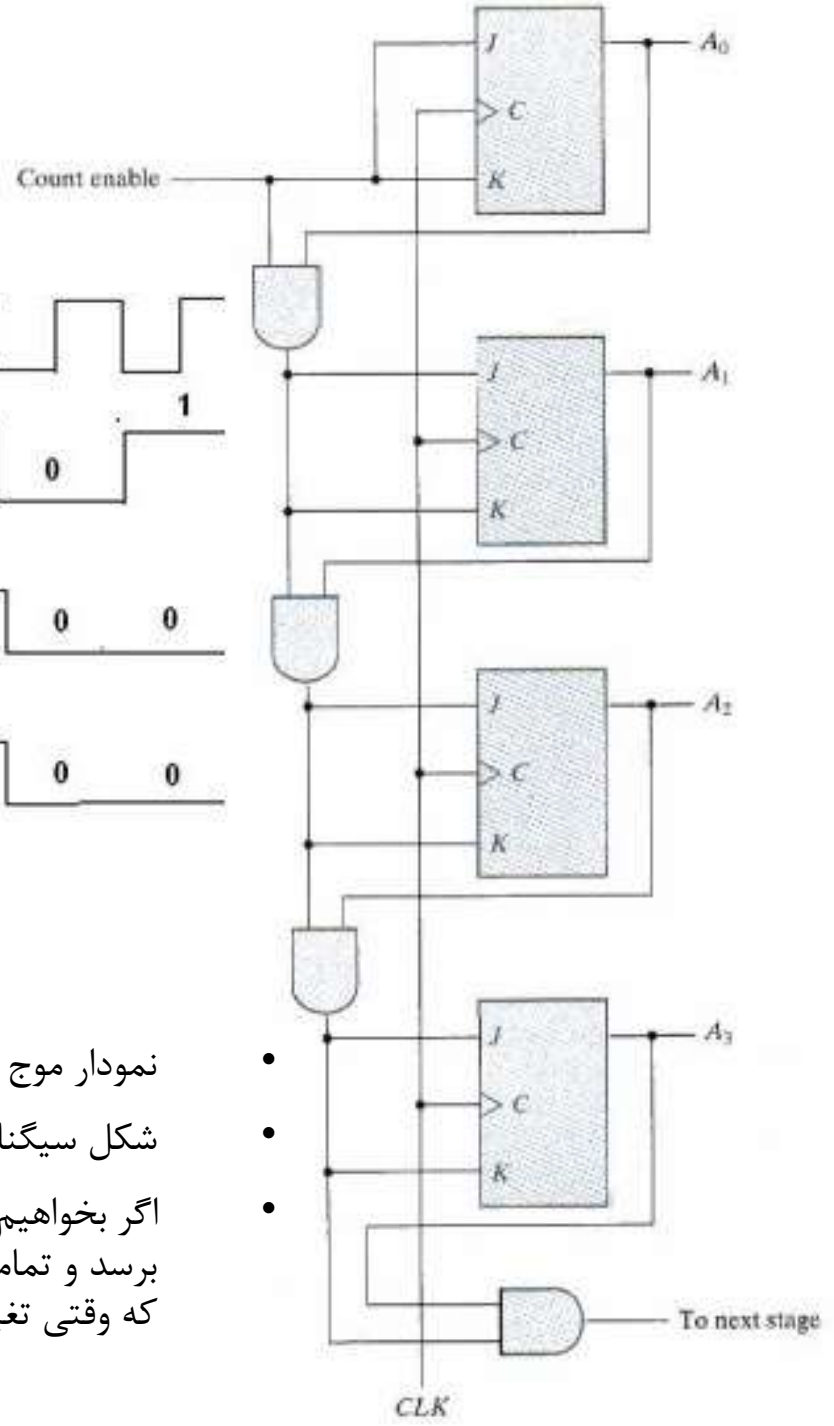


- در شکل یک شمارنده موجی دودویی را می‌بینید.
- ورودی که بناست شمارش شود (احتمالا پالس ساعت) به بالاترین فلیپ‌فلاپ وصل شده و آرایش به گونه‌ای است که هر دور بالا و پایین رفتن ورودی هر فلیپ‌فلاپ موجب یک (بالا یا پایین) خروجی که ورودی پایینی هم هست می‌شود.

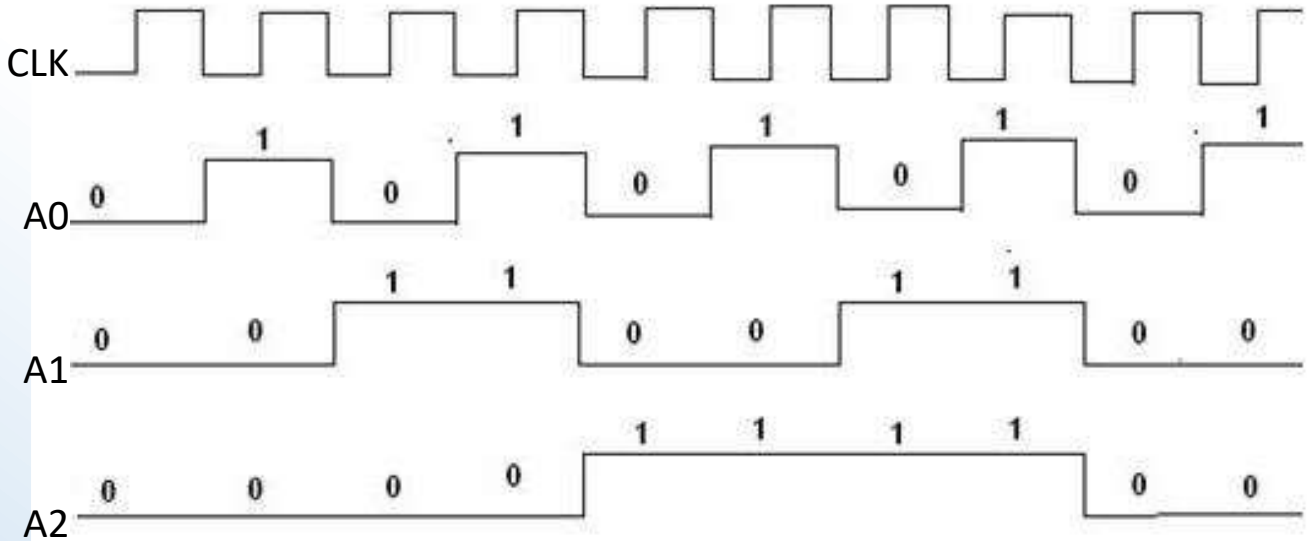
شمارنده سنکرون

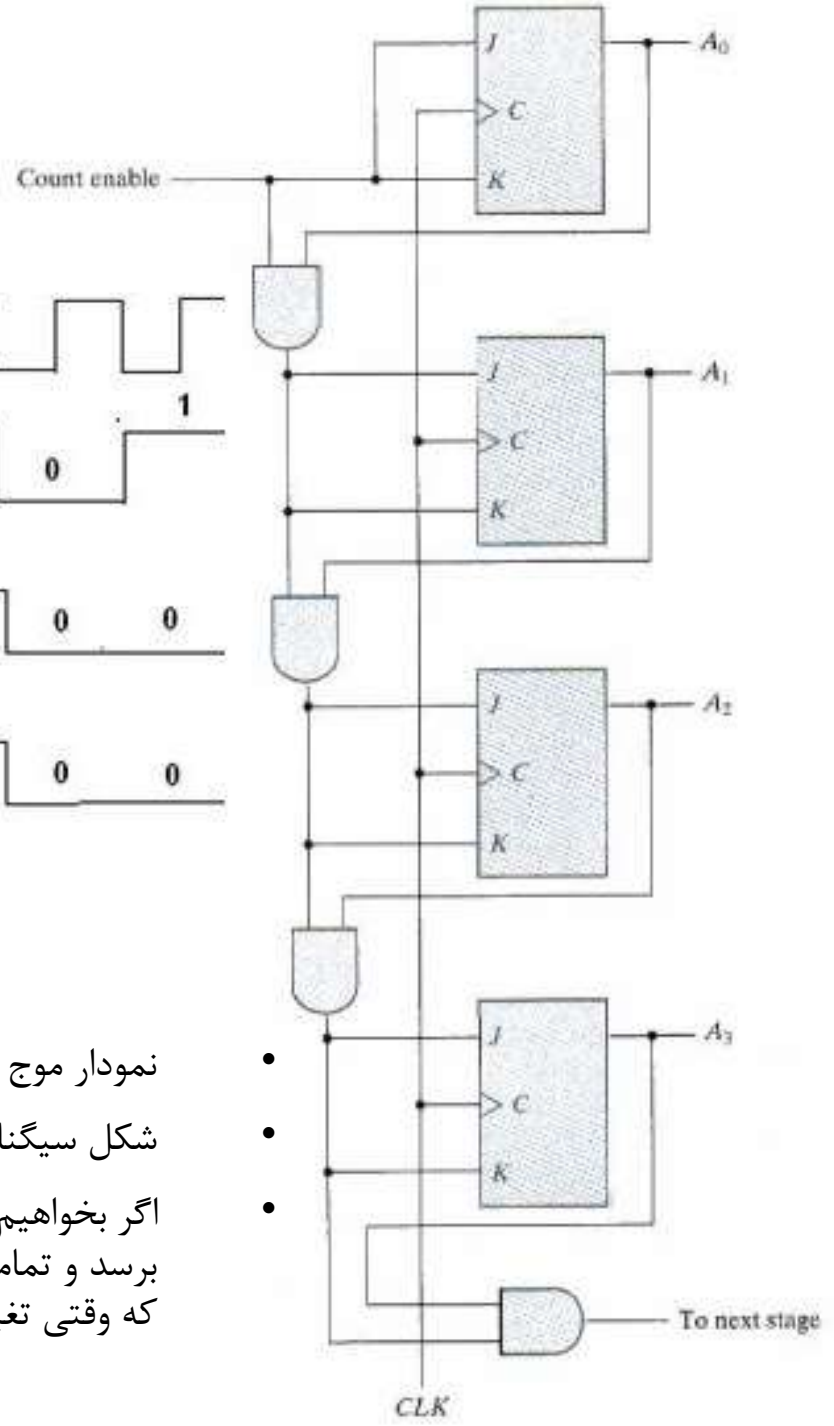
- در شکل بعد یک شمارنده دودویی سنکرون (همه فلیپ‌فلاپ‌ها دارای ساعت مشترک) را می‌بینید.
- در این طراحی از فلیپ‌فلاپ JK استفاده شده اما دو ورودی بهم وصل هستند (می‌شد به سادگی از فلیپ‌فلاپ T استفاده کرد). نتیجه اینکه اگر ورودی 0 باشد خروجی بی‌تغییر است و اگر 1 باشد خروجی معکوس حالت قبل خواهد شد.
- اساس کار مانند شمارنده موجی است. یعنی در هر طبقه طول موج دو برابر می‌شود.
- علاوه بر ساعت یک ورودی داریم که فعال‌ساز شمارش است (بعد از یک شدن آن شمارش آغاز می‌شود).





- نمودار موج حساس به لبه پایین رونده (خروجی آخری یعنی A3 در این شکل نیست).
- شکل سیگنال‌ها هم برای شمارنده موجی و هم شمارنده سنکرون یکی است.
- اگر بخواهیم ساده صحبت کنیم: هر شمارنده وقتی تغییر حالت می‌دهد که پالس ساعت برسد و تمامی فلیپ‌فلاپ‌های بالاتر از آن در حالت 1 باشند (مثل شمارنده پمپ بنزین که وقتی تغییر می‌کند که تمام رقم های کوچکتر 9 باشند).





- نمودار موج حساس به لبه پایین رونده (خروجی آخری یعنی A3 در این شکل نیست).
- شکل سیگنال‌ها هم برای شمارنده موجی و هم شمارنده سنکرون یکی است.
- اگر بخواهیم ساده صحبت کنیم: هر شمارنده وقتی تغییر حالت می‌دهد که پالس ساعت برسد و تمامی فلیپ‌فلاپ‌های بالاتر از آن در حالت 1 باشند (مثل شمارنده پمپ بنزین که وقتی تغییر می‌کند که تمام رقم های کوچکتر 9 باشند).

تمرین

الف) شیفت رجسیتر چه کاربردی دارد؟ نمونه‌ای که ما دیدیم سنکرون است یا آسنکرون؟

ب) انتقال موازی چه مزیتی بر انتقال سریال دارد و انتقال سریال چه مزیتی بر انتقال موازی دارد؟

ج) یک شمارنده سنکرون دوبیتی با فلیپ فلاپ تی رسم کنید. سیگنال‌ها را هم بکشید.

مدارهای منطقی

(طراحی سیستم‌های دیجیتال)

جلسه دوازدهم

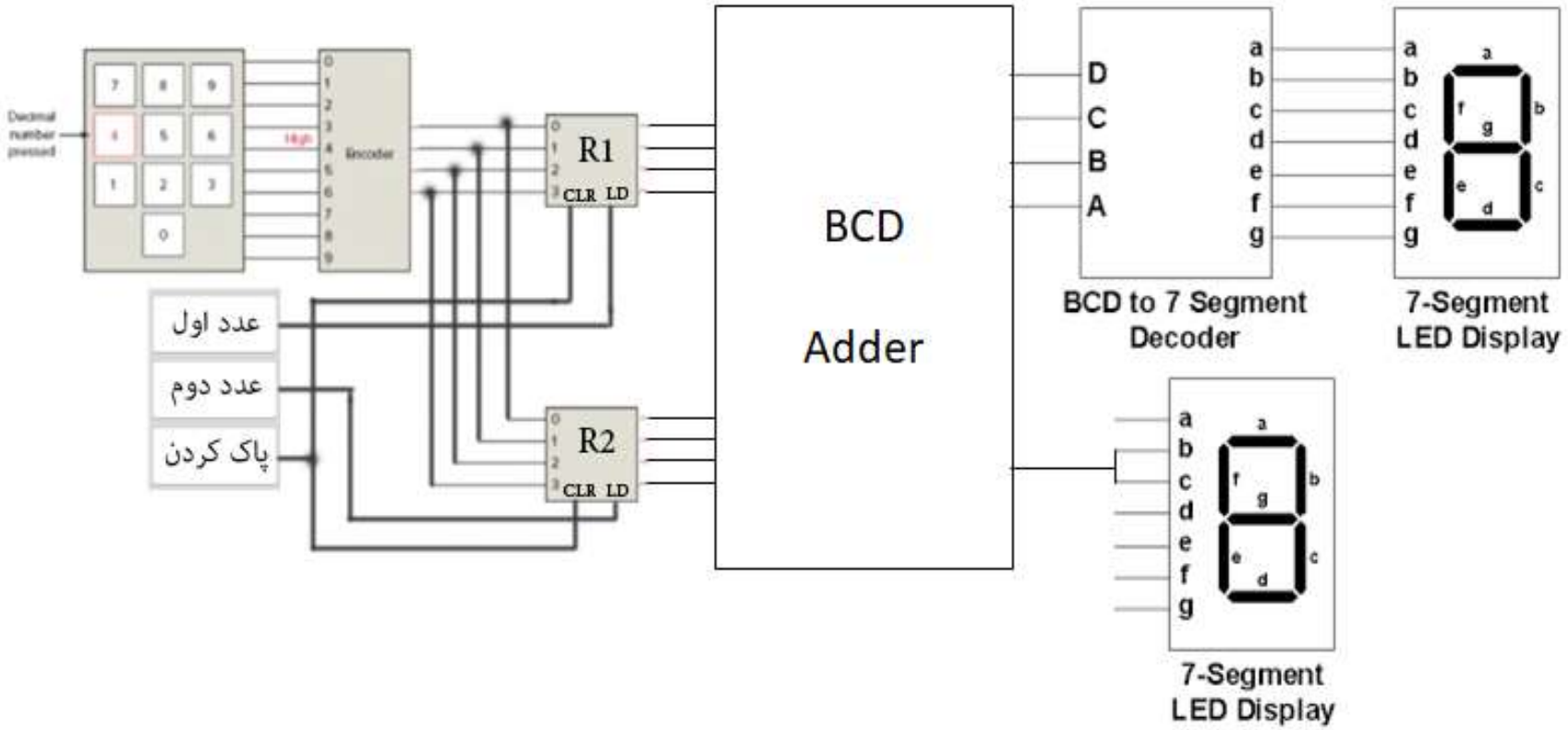
طراحی دو مدار ترتیبی

تکمیل مدار ماشین حساب

- ماشین حسابی که قبلا در این درس طراحی کردیم (به غیر از تک رقمی بودن و انجام تنها عمل جمع) دو ایراد داشت: یکی اینکه نیاز به دو صفحه کلید داشت و دیگری اینکه کاربر باید کلیدها را نگه می‌داشت و به محض رها کردن آنها نتیجه محو می‌شد. اکنون می‌خواهیم این دو ایراد را برطرف کنیم.

تکمیل مدار ماشین حساب

- برای این منظور از دو رجیستر چهاربیتی استفاده می‌کنیم که یکی R1 و دیگری R2 نامیده شده است. هر یک از این رجیسترها یکی از دو عدد را در خود ذخیره میکند.
- کاربر با گرفتن عدد مورد نظر و زدن دگمه بارگیری آن، عدد را به حافظه ماشین حساب منتقل می‌کند. همچنین دگمه ای وجود دارد که هر دو عدد را صفر می‌کند (مثل C در ماشین‌حساب‌های واقعی).
- رجیسترها به یک ساعت متصل هستند. اگر بخواهیم ساعت نداشته باشیم رجیسترها بجای فلیپ‌فلاپ باید لچ داشته باشند.
- بجای این مدار ساده اگر یک مدار چهارحالته جایگزین شود دو کلید بارگیری حذف می‌شوند اما طراحی پیچیده می‌شود.


Decimal number pressed
Encoder
R1
CLR LD
R2
CLR LD
عدد اول
عدد دوم
پاک کردن
BCD
Adder
D
C
B
A
a
b
c
d
e
f
g
BCD to 7 Segment Decoder
7-Segment LED Display
7-Segment LED Display

انواع مدارهای منطقی ترتیبی

- مدارهای منطقی ترتیبی، ماشین‌های حالت متناهی را شبیه‌سازی می‌کنند که با توجه به ورودی در هر سیکل ساعت از حالتی به حالت دیگر می‌روند.
- اگر مقدار کنونی خروجی فقط به حالت کنونی مدار وابسته باشد مدار از نوع ماشین مور Moore است (خروجی روی گره مشخص می‌شود، فقط با لبه ساعت تغییر می‌کند، سنکرون است).
- اگر مقدار کنونی خروجی به حالت کنونی مدار و مقدار کنونی ورودی وابسته باشد مدار از نوع ماشین میلی Mealy است (خروجی روی یال مشخص می‌شود، وسط سیکل ساعت هم عوض می‌شود، سنکرون نیست).

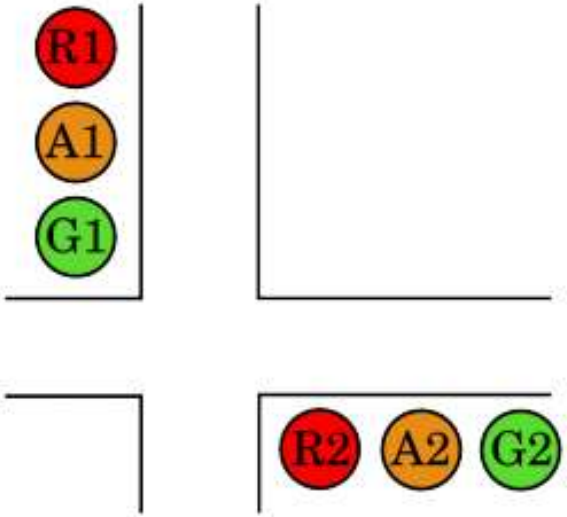
مساله

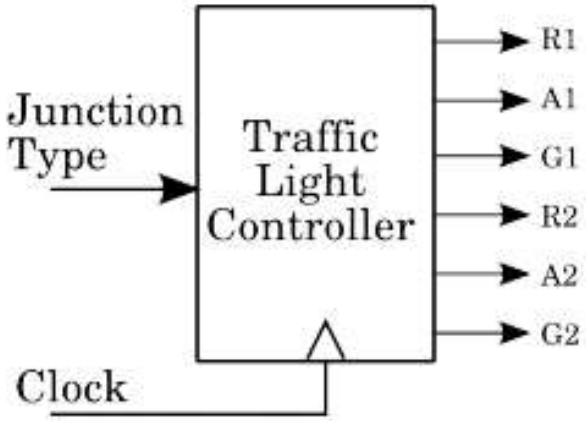
- قصد داریم مداری دیجیتال طراحی کنیم که کنترل چراغ راهنمایی سر چهارراه را بر عهده بگیرد.
- مدار یک ورودی دارد به نام J که اگر چهارراه خلوت باشد 1 و اگر شلوغ باشد 0 است.
- در زمان خلوتی سیستم به حالت معمول چهارراه‌ها عمل می‌کند، اما در زمان شلوغی بلافاصله بعد از قرمز شدن یک چراغ، چراغ دیگر سبز نمی‌شود بلکه برای مدتی هر دو چراغ قرمز می‌ماند و بعد آن یکی سبز می‌شود.
- **حالت مدار** تعیین کننده شش خروجی است که **سبز و قرمز و زرد چراغ شمالی جنوبی و چراغ شرقی غربی** را روشن و خاموش می‌کند.

مساله

خلوت

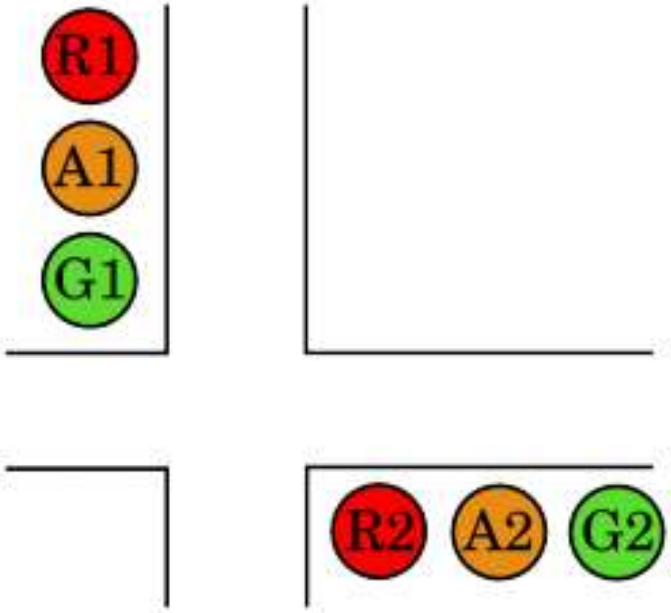
شلوغ





Red,Amber,Green

مساله

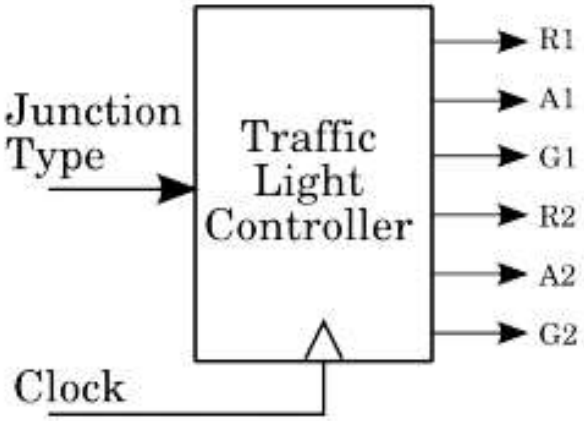


خلوت

سبز	قرمز
زرد	قرمز
قرمز	سبز
قرمز	زرد

شلوغ

سبز	قرمز
زرد	قرمز
قرمز	قرمز
قرمز	سبز
قرمز	زرد
قرمز	قرمز



Red,Amber,Green

مساله

- مدت زمان هر کدام از حالت ها یک سیکل ساعت است. مثلا اگر سیکل ساعت ۵ ثانیه باشد، چراغ سبز و زرد و قرمز هر یک ۵ ثانیه طول می‌کشد. در این مساله قصد متغیر کردن زمان را نداریم گرچه این کار با استفاده از شمارنده به آسانی ممکن است.
- از آنجا که خروجی مدار (رنگ چراغ ها) فقط به حالت مدار بستگی دارد (تغییر J بلافاصله تغییری در خروجی ایجاد نمی‌کند) آن را بصورت مدار مور طراحی خواهیم کرد.

قدم اول: تعداد حالات

خلوت

۱	سبز	قرمز
۲	زرد	قرمز
٤	قرمز	سبز
۵	قرمز	زرد

شلوغ

۱	سبز	قرمز
۲	زرد	قرمز
۳	قرمز	قرمز
۴	قرمز	سبز
۵	قرمز	زرد
۶	قرمز	قرمز

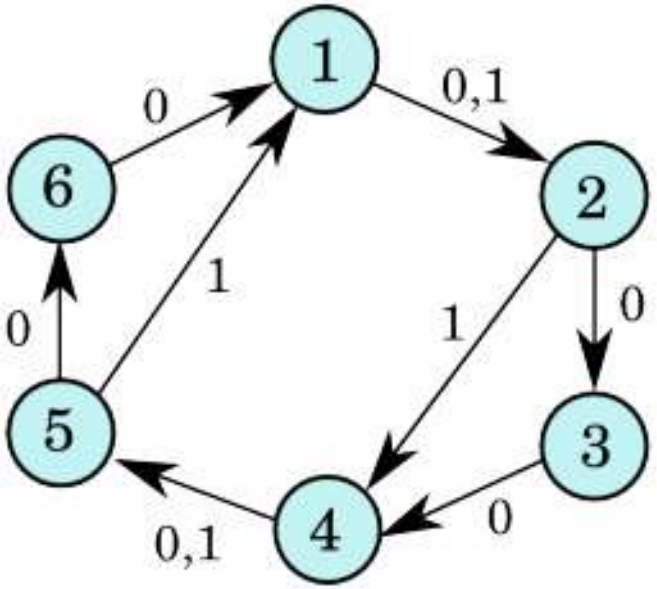
- همانطور که در صورت مساله آمده در زمانی خلوتی به چهار حالت و در زمان شلوغی به شش حالت نیاز دارد. با توجه به مشترک بود حالات در مجموع به شش (نه ده) حالت نیاز داریم. این حالت ها را از ۱ تا ۶ شماره می‌دهیم.
- **قدم دوم:** رسم دیاگرام حالت بدون توجه به خروجی‌ها

State	R1	A1	G1	R2	A2	G2
1	1	0	0	0	0	1
2	1	0	0	0	1	0
3	1	0	0	1	0	0
4	0	0	1	1	0	0
5	0	1	0	1	0	0
6	1	0	0	1	0	0

قدم اول: تعداد حالات

- همانطور که در صورت مساله آمده در زمانی خلوتی به چهار حالت و در زمان شلوغی به شش حالت نیاز دارد. با توجه به مشترک بود حالات در مجموع به شش (نه ده) حالت نیاز داریم. این حالت ها را از ۱ تا ۶ شماره می‌دهیم.
- **قدم دوم:** رسم دیاگرام حالت بدون توجه به خروجی‌ها

State	R1	A1	G1	R2	A2	G2
1	1	0	0	0	0	1
2	1	0	0	0	1	0
3	1	0	0	1	0	0
4	0	0	1	1	0	0
5	0	1	0	1	0	0
6	1	0	0	1	0	0



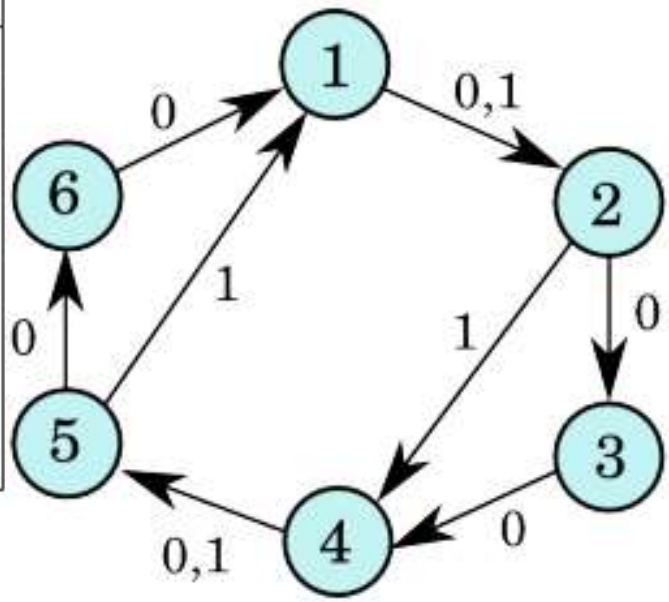
قدم سوم: تصمیم گیری در مورد نوع و تعداد فلیپ فلاپ‌ها

- از آنجا که شش حالت داریم حداقل تعداد فلیپ‌فلاپ‌ها سه عدد خواهد بود (دو حالت بلااستفاده خواهیم داشت).
- ساده‌ترین نوع فلیپ‌فلاپ یعنی نوع D را در اینجا استفاده خواهیم کرد.

قدم چهارم:تخصیص حالت فلیپ‌فلاپ‌ها

- قاعده‌ای وجود ندارد که بهتر بودن یک تخصیص را تضمین‌کند. اما بطور کلی (اگر با مینترم کار کنیم) هر چه تعداد 1 ها بیشتر باشد بعدا در جدول کارنو راحت‌تر ساده خواهند شد.
- بنابراین دو حالت بلااستفاده (7 و 8) را اول(0 و 1) می‌گذاریم و بعد حالات 1 تا 6 را به ترتیب به صورت باینری شماره می‌دهیم.

J	State(t)	Q1	Q2	Q3	State(t+1)	D1	D2	D3
0	7	0	0	0				
0	8	0	0	1				
0	1	0	1	0				
0	2	0	1	1				
0	3	1	0	0				
0	4	1	0	1				
0	5	1	1	0				
0	6	1	1	1				
1	7	0	0	0				
1	8	0	0	1				
1	1	0	1	0				
1	2	0	1	1				
1	3	1	0	0				
1	4	1	0	1				
1	5	1	1	0				
1	6	1	1	1				



J	State(t)	Q1	Q2	Q3	State(t+1)	D1	D2	D3
0	7	0	0	0	X	X	X	X
0	8	0	0	1	X	X	X	X
0	1	0	1	0	2	0	1	1
0	2	0	1	1	3	1	0	0
0	3	1	0	0	4	1	0	1
0	4	1	0	1	5	1	1	0
0	5	1	1	0	6	1	1	1
0	6	1	1	1	1	0	1	0
1	7	0	0	0	X	X	X	X
1	8	0	0	1	X	X	X	X
1	1	0	1	0	2	0	1	1
1	2	0	1	1	4	1	0	1
1	3	1	0	0	X	X	X	X
1	4	1	0	1	5	1	1	0
1	5	1	1	0	1	0	1	0
1	6	1	1	1	X	X	X	X

J	State(t)	Q1	Q2	Q3	State(t+1)	D1	D2	D3
0	7	0	0	0	X	X	X	X
0	8	0	0	1	X	X	X	X
0	1	0	1	0	2	0	1	1
0	2	0	1	1	3	1	0	0
0	3	1	0	0	4	1	0	1
0	4	1	0	1	5	1	1	0
0	5	1	1	0	6	1	1	1
0	6	1	1	1	1	0	1	0
1	7	0	0	0	X	X	X	X
1	8	0	0	1	X	X	X	X
1	1	0	1	0	2	0	1	1
1	2	0	1	1	4	1	0	1
1	3	1	0	0	X	X	X	X
1	4	1	0	1	5	1	1	0
1	5	1	1	0	1	0	1	0
1	6	1	1	1	X	X	X	X

قدم پنجم: رسم جدول کارنو

J Q1 \ Q2 Q3	00	01	11	10
00	X	X	1	0
01	1	1	0	1
11	X	1	X	0
10	X	X	1	0

$D1 = Q2' + Q3Q1' + J'Q1Q3'$

J Q1 \ Q2 Q3	00	01	11	10
00	X	X	0	1
01	0	1	1	1
11	X	1	X	1
10	X	X	0	1

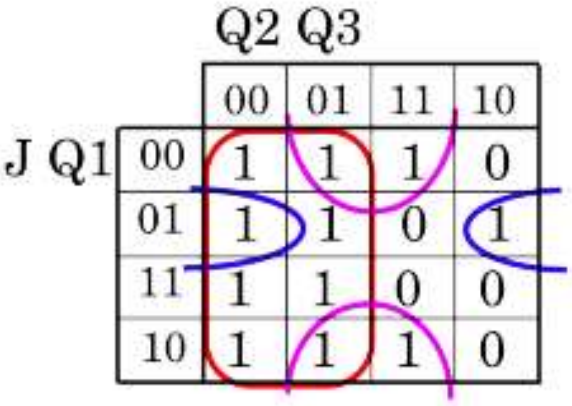
$D2 = Q1Q3 + Q2Q3'$

J Q1 \ Q2 Q3	00	01	11	10
00	X	X	0	1
01	1	0	0	1
11	X	0	X	0
10	X	X	1	1

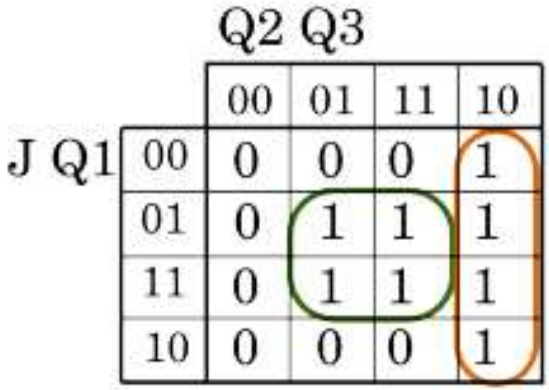
$D3 = J'Q3' + J\,Q1'$

قدم ششم: اصلاح حالات بی اهمیت

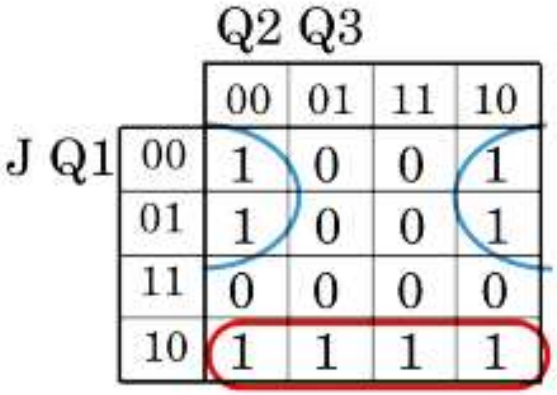
- در جدول کارنو حالات بی اهمیت را برخی صفر و برخی یک گرفتیم. حال این مقادیر را در جدول حالت تاثیر می‌دهیم تا هیچ ابهامی نماند.



J Q1 \ Q2 Q3	00	01	11	10
00	1	1	1	0
01	1	1	0	1
11	1	1	0	0
10	1	1	1	0

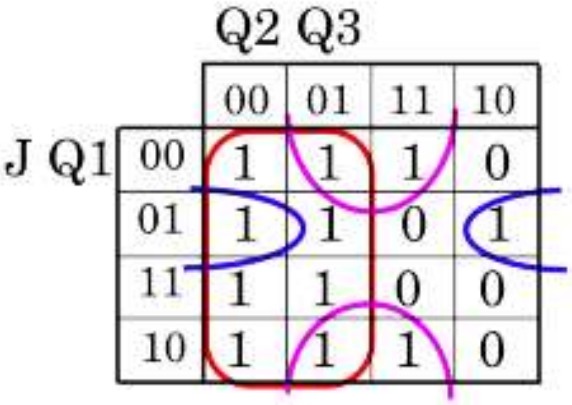


J Q1 \ Q2 Q3	00	01	11	10
00	0	0	0	1
01	0	1	1	1
11	0	1	1	1
10	0	0	0	1

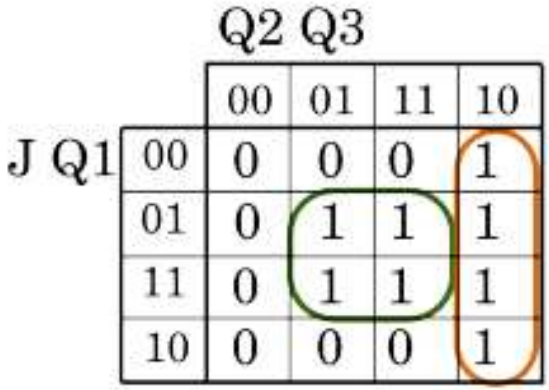


J Q1 \ Q2 Q3	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	0	0	0	0
10	1	1	1	1

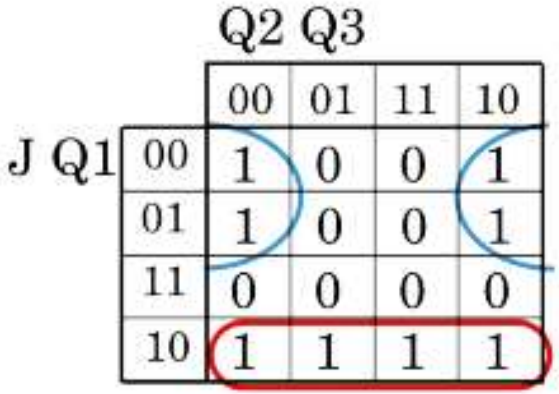
قدم ششم: اصلاح حالات بی اهمیت



J Q1 \ Q2 Q3	00	01	11	10
00	1	1	1	0
01	1	1	0	1
11	1	1	0	0
10	1	1	1	0



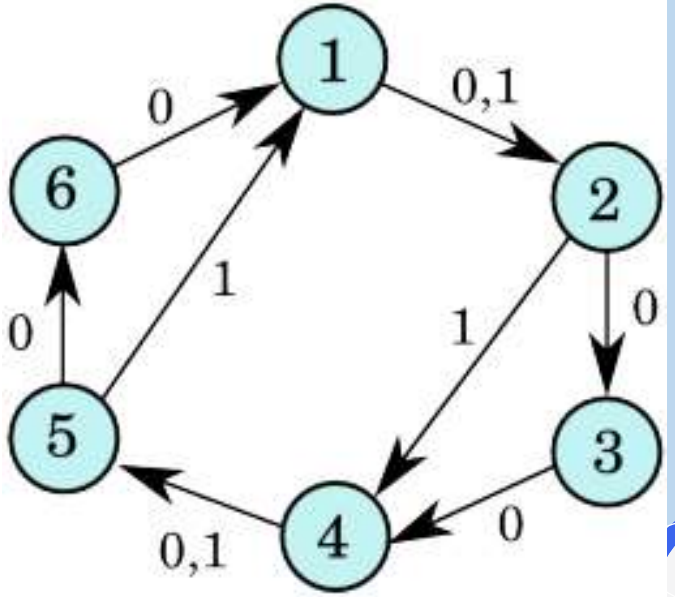
J Q1 \ Q2 Q3	00	01	11	10
00	0	0	0	1
01	0	1	1	1
11	0	1	1	1
10	0	0	0	1



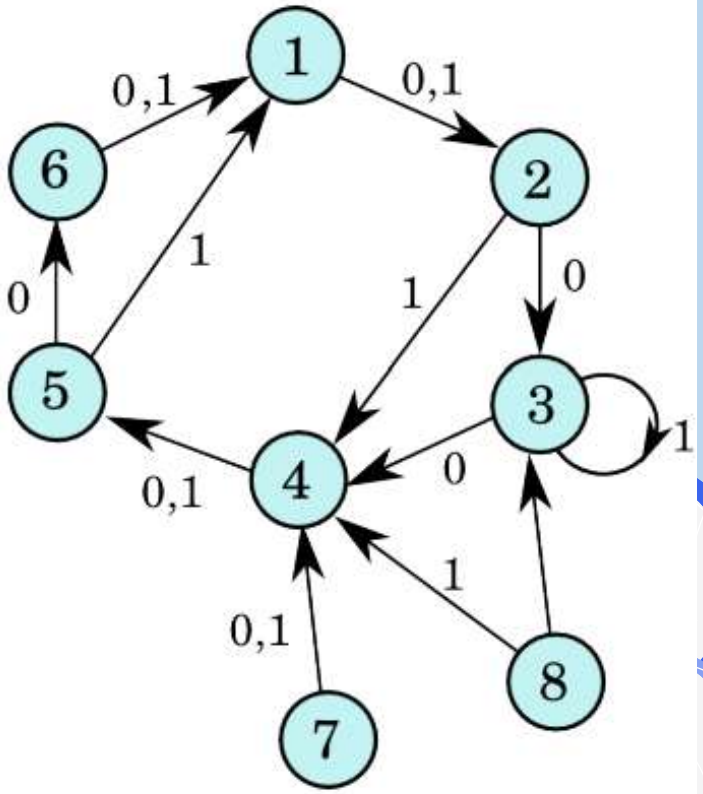
J Q1 \ Q2 Q3	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	0	0	0	0
10	1	1	1	1

J	State(t)	Q1	Q2	Q3	State(t+1)	D1	D2	D3
0	7	0	0	0	X	X	X	X
0	8	0	0	1	X	X	X	X
0	1	0	1	0	2	0	1	1
0	2	0	1	1	3	1	0	0
0	3	1	0	0	4	1	0	1
0	4	1	0	1	5	1	1	0
0	5	1	1	0	6	1	1	1
0	6	1	1	1	1	0	1	0
1	7	0	0	0	X	X	X	X
1	8	0	0	1	X	X	X	X
1	1	0	1	0	2	0	1	1
1	2	0	1	1	4	1	0	1
1	3	1	0	0	X	X	X	X
1	4	1	0	1	5	1	1	0
1	5	1	1	0	1	0	1	0
1	6	1	1	1	X	X	X	X

J	State(t)	Q1	Q2	Q3	State(t+1)	D1	D2	D3
0	7	0	0	0	4	1	0	1
0	8	0	0	1	3	1	0	0
0	1	0	1	0	2	0	1	1
0	2	0	1	1	3	1	0	0
0	3	1	0	0	4	1	0	1
0	4	1	0	1	5	1	1	0
0	5	1	1	0	6	1	1	1
0	6	1	1	1	1	0	1	0
1	7	0	0	0	4	1	0	1
1	8	0	0	1	4	1	0	1
1	1	0	1	0	2	0	1	1
1	2	0	1	1	4	1	0	1
1	3	1	0	0	3	1	0	0
1	4	1	0	1	5	1	1	0
1	5	1	1	0	1	0	1	0
1	6	1	1	1	1	0	1	0



J	State(t)	Q1	Q2	Q3	State(t+1)	D1	D2	D3
0	7	0	0	0	4	1	0	1
0	8	0	0	1	3	1	0	0
0	1	0	1	0	2	0	1	1
0	2	0	1	1	3	1	0	0
0	3	1	0	0	4	1	0	1
0	4	1	0	1	5	1	1	0
0	5	1	1	0	6	1	1	1
0	6	1	1	1	1	0	1	0
1	7	0	0	0	4	1	0	1
1	8	0	0	1	4	1	0	1
1	1	0	1	0	2	0	1	1
1	2	0	1	1	4	1	0	1
1	3	1	0	0	3	1	0	0
1	4	1	0	1	5	1	1	0
1	5	1	1	0	1	0	1	0
1	6	1	1	1	1	0	1	0



قدم هفتم: رسم مدار

- با توجه به دانستن ورودی‌های Q1,Q2,Q3 که از جدول‌های کارنو بدست آمده، می‌توان مدار را رسم کرد.
- اما قبل از آن مشکلی هست. خوب به دیاگرام حالت آخری نگاه کنید.
- اگر زمانی که در وضعیت 3 هستیم چهار راه خلوت شود و اپراتور J را 1 کند، آنگاه برای همیشه در وضعیت 3 می‌مانیم (وقتی در وضعیت 3 هستیم و چنین اتفاقی می‌افتد تصادفا مدار درست عمل می‌کند). برای حل این مشکل باید جدول حالت و بعد جداول کارنو اصلاح شوند تا با هر ورودی J (چه 0 چه 1) از حالت 3 به حالت 4 برویم.

قدم هفتم: رسم مدار

- ساختن خروجی (سبز و قرمز و زرد دو چراغ) از خروجی فلیپ‌فلاپ‌ها (با توجه به اسلاید صفحه ۶) ممکن است (بعد از کارنو با گیت یا PLA و حتی بدون کارنو و طراحی گیتی با ROM براحتی قابل انجامست).
- مثلا در مورد حالت شماره 1 دو چیز را می‌دانیم:
  اولا در این حالت چراغ‌های R1 و G2 روشن هستند.
  دوما در این حالت Q1 و Q3 خاموش و Q2 روشن است.
- به همین صورت می‌توان جدول درستی کشید ورودی‌های آن سه فلیپ‌فلاپ و خروجی‌هایش شش چراغ باشد و مدار خروجی را از روی آن کامل کرد.

تمرین

- اصلاحات لازم که در پاراگراف آخر اسلاید ۲۲ گفته شد را انجام دهید. جدول حالت، دیاگرام حالت جدول کارنو و مدار مربوط به آن را رسم کنید.
- اگر نصف نمره این بخش برایتان کافیست بجای مورد قبل، مدار مربوط به مدار اصلاح نشده (که جدول و دیاگرام حالت و کارنوی آن موجود است) را رسم کنید (فقط مدار)
- برای رسیدن از حالت‌ها به خروجی (خروجی فلیپ‌فلاپ‌ها به چراغ‌ها) از یک ROM استفاده کنید.

تمرین اختیاری

- مدار پاراگراف آخر اسلاید ۳ را رسم کنید تا با وارد کردن دو عدد بترتیب آنها در رجیسترها بار شوند و زدن کلید C موجب پاک کردن رجیسترها و رفتن به حالت اول شود. چنین مداری چهارحالت دارد : ۱. آماده دریافت عدد اول ۲. بارگیری عدد اول ۳. آماده دریافت عدد دوم ۴. بارگیری عدد دوم. مدار دارای دو ورودی است (فعال بودن کلیدها = حاصل or تمام ده کلید و نیز کلید پاک کردن یا C) است. مدار دو خروجی دارد که برای LD رجیسترها هستند.

خسته نباشید

Q&A