



دولت جمهوری اسلامی افغانستان  
اداره تعلیمات تخنیکي و مسلکي  
معاونیت امور اکادمیک  
ریاست نصاب و تربیه معلم

# اساسات دیجیتل

رشته: کمپیوتر ساینس - دیپارتمنت: عمومی  
صنف ۱۳ - سمستر اول

سال: ۱۳۹۹ هجری شمسی



## شناسنامه کتاب

نام کتاب: اساسات دیجیتل  
رشته: کمپیوتر ساینس  
تدوین کننده: محمد رضا رضایی  
همکار تدوین کننده: مرتضی نیازی، لیدا کریمی

- کمیته نظارت: ندیمه سحر رئیس اداره تعلیمات تخنیکي و مسلکی
- عبدالحمید اکبر معاون امور اکادمیک اداره تعلیمات تخنیکي و مسلکی
- حبیب الله فلاح رئیس نصاب و تربیه معلم
- عبدالمتین شریفی آمر انکشاف نصاب تعلیمی، ریاست نصاب و تربیه معلم
- روح الله هوتک آمر طبع و نشر کتب درسی، ریاست نصاب و تربیه معلم
- احمد بشیر هیله من مسؤل انکشاف نصاب، پروژه انکشاف مهارت های افغانستان
- محمد زمان پویا کارشناس انکشاف نصاب، پروژه انکشاف مهارت های افغانستان
- علی خیبر یعقوبی سرپرست مدیریت عمومی تألیف کتب درسی، ریاست نصاب و تربیه معلم

- کمیته تصحیح: دوکتور فضل احمد امینی
- سحر احمدی
- محمد امان هوشمند مدیر عمومی بورد تصحیح کتب درسی و آثار علمی

دیزاین: صمد صبا و سید کاظم کاظمی  
چاپ کال: ۱۳۹۹ هجری شمسی  
تیراژ: ۱۰۰۰  
چاپ: اول  
وب سایت: [www.tveta.gov.af](http://www.tveta.gov.af)  
ایمیل: [info@tveta.gov.af](mailto:info@tveta.gov.af)

حق چاپ برای اداره تعلیمات تخنیکي و مسلکی محفوظ است.



## سرود ملی

دا وطن افغانستان دی	دا عزت د هر افغان دی
کور د سولې کور د تورې	هر بچی یې قهرمان دی
دا وطن د ټولو کور دی	د بلوڅو، د ازبکو
د پښتون او هزاره وو	د ترکمنو، د تاجکو
ورسره عرب، کوچر دي	پامیریان، نورستانیان
براهوي دي، قزلباش دي	هم ایماق، هم پشه یان
دا هیواد به تل ځلېږي	لکه لمر پر شنه آسمان
په سینه کې د آسیا به	لکه زړه وی جاویدان
نوم د حق مو دی رهبر	وایو الله اکبر وایو الله اکبر



## پیام اداره تعلیمات تخنیکي و مسلکي

استادان نهایت گرامی و محصلان ارجمند!

تربیت نیروی بشری ماهر، متخصص و کارآمد از عوامل کلیدی و انکارناپذیر در توسعه اقتصادی و اجتماعی هر کشور محسوب می‌گردد و هر نوع سرمایه‌گذاری بزرگ در بخش‌های مختلف اقتصادی نیازمند به پلان‌گذاری و سرمایه‌گذاری در بخش نیروی بشری و توسعه منابع این نیرو می‌باشد. بر مبنای این اصل و بر اساس فرمان شماره ۱۱ مقام عالی ریاست جمهوری اسلامی افغانستان به تاریخ ۱۳۹۷/۲/۱ اداره تعلیمات تخنیکي و مسلکي از بدنه وزارت معارف مجزا و فصل جدیدی در بخش عرضه خدمات آموزشی در کشور گشوده شد. اداره تعلیمات تخنیکي و مسلکي به‌عنوان متولی و مجری آموزش‌های تخنیکي و مسلکي در کشور محسوب می‌شود که در چارچوب استراتژی ۵ ساله خویش دارای چهار اولویت مهم که عبارت‌اند از افزایش دسترسی عادلانه و مساویانه فراگیران آموزش‌های تخنیکي و مسلکي در سطح کشور، بهبود کیفیت در ارائه خدمات آموزشی، یادگیری مادام‌العمر و پیوسته و ارائه آموزش نظری و عملی مهارت‌ها به‌طور شفاف، کم‌هزینه و مؤثر که بتواند نیاز بازار کار و محصلان را در سطح محلی، ملی و بین‌المللی برآورده کند، می‌باشد. این اداره که فراگیرترین نظام تعلیمی کشور در بخش تعلیمات تخنیکي و مسلکي است، تلاش می‌کند تا در حیطه وظایف و صلاحیت خود زمینه دستیابی به هدف‌های تعیین‌شده را ممکن سازد و جهت رفع نیاز بازار کار، فعالیت‌های خویش را توسعه دهد.

نظام اجتماعی و طرز زندگی در افغانستان مطابق به احکام دین مقدس اسلام و رعایت تمامی قوانین مشروع و معقول انسانی عیار است. اداره تعلیمات تخنیکي و مسلکي جمهوری اسلامی افغانستان نیز با ایجاد زمینه‌های لازم برای تعلیم و تربیت جوانان و نوجوانان مستعد و علاقه‌مند به حرفه‌آموزی، ارتقای مهارت‌های شغلی در سطوح مختلف مهارتی، تربیت کادرهای مسلکي و حرفوی و ظرفیت‌سازی تخصصی از طریق انکشاف و ایجاد مکاتب و انستیتوت‌های تخنیکي و مسلکي در سطح کشور با رویکرد ارزش‌های اسلامی و اخلاقی فعالیت می‌نماید.

فلذا جهت نیل به اهداف عالی این اداره که همانا تربیه افراد ماهر و توسعه نیروی بشری در کشور می‌باشد؛ داشتن نصاب تعلیمی بر وفق نیاز بازار کار امر حتمی و ضروری بوده و کتاب درسی یکی از ارکان مهم فرایند آموزش‌های تخنیکي و مسلکي محسوب می‌شود، پس باید همگام با تحولات و پیشرفت‌های علمی نوین و مطابق نیازمندی‌های جامعه و بازار کار تألیف و تدوین گردد و دارای چنان ظرافتی باشد که بتواند آموزه‌های دینی و اخلاقی را توأم با دست‌آورد‌های علوم جدید با روش‌های نوین به محصلان انتقال دهد. کتابی را که اکنون در اختیاردارید، بر اساس همین ویژگی‌ها تهیه و تدوین گردیده است.

بدین‌وسیله، صمیمانه آرزومندیم که آموزگاران خوب، متعهد و دلسوز کشور با خلوص نیت، رسالت اسلامی و ملی خویش را ادا نموده و نوجوانان و جوانان کشور را به‌سوی قله‌های رفیع دانش و مهارت‌های مسلکي رهنمایی نمایند و از محصلان گرامی نیز می‌خواهیم که از این کتاب به‌درستی استفاده نموده، در حفظ و نگهداشت آن سعی بلیغ به خرج دهند. همچنان از مؤلفان، استادان، محصلان و اولیای محترم محصلان تقاضا می‌شود نظریات و پیشنهادات خود را در مورد این کتاب از نظر محتوا، ویرایش، چاپ، اشتباهات املائی، انشایی و تایپی عنوانی اداره تعلیمات تخنیکي و مسلکي کتباً ارسال نموده، امتنان بخشند.

در پایان لازم می‌دانیم در جنب امتنان از مؤلفان، تدوین‌کنندگان، مترجمان، مصححان و تدقیق‌کنندگان نصاب تعلیمات تخنیکي و مسلکي از تمامی نهادهای ملی و بین‌المللی که در تهیه، تدوین، طبع و توزیع کتب درسی زحمت‌کشیده و همکاری نموده‌اند، قدردانی و تشکر نمایم.

ندیمه سحر

رئیس اداره تعلیمات تخنیکي و مسلکي جمهوری اسلامی افغانستان

ط	مقدمه	.....
۱	فصل اول: سیستم اعداد	.....
۲	سیستم‌های دیجیتال	..... ۱.۱
۴	سیستم اعداد	..... ۱.۲
۷	روش‌های تبدیل قاعده‌های اعداد	..... ۱.۳
۷	تبدیل اعداد از قاعده ۲ به قاعده ۱۰	..... ۱.۳.۱
۸	تبدیل اعداد از قاعده ۲ به قاعده ۸	..... ۱.۳.۲
۸	تبدیل اعداد از قاعده ۲ به قاعده ۱۶	..... ۱.۳.۳
۸	تبدیل اعداد از قاعده ۱۰ به قاعده ۲	..... ۱.۳.۴
۹	تبدیل اعداد از قاعده ۱۰ به قاعده ۸	..... ۱.۳.۵
۹	تبدیل اعداد از قاعده ۱۰ به قاعده ۱۶	..... ۱.۳.۶
۹	تبدیل اعداد اوکتال به باینری	..... ۱.۳.۷
۹	تبدیل اعداد از قاعده ۸ به قاعده ۱۰	..... ۱.۳.۸
۱۰	تبدیل اعداد از قاعده ۸ به قاعده ۱۶	..... ۱.۳.۹
۱۰	تبدیل اعداد از قاعده ۱۶ به قاعده ۲	..... ۱.۳.۱۰
۱۰	تبدیل اعداد از قاعده ۱۶ به قاعده ۸	..... ۱.۳.۱۱
۱۰	تبدیل اعداد از قاعده ۱۶ به قاعده ۱۰	..... ۱.۳.۱۲
۱۲	کاربرد قاعده‌ها	..... ۱.۴
۱۲	متمم‌ها (Complements)	..... ۱.۵
۱۲	متمم $r-1$ (One's Complement)	..... ۱.۵.۱
۱۳	متمم $r$ (Two's Complement)	..... ۱.۵.۲
۱۴	تبدیل تفریق به جمع با استفاده از متمم ۱	..... ۱.۵.۳
۱۵	تبدیل تفریق به جمع با استفاده از متمم ۲	..... ۱.۵.۴
۱۹	فصل دوم: جبر بولی (Boolean Algebra)	.....
۲۰	تعریف جبر بولی	..... ۲.۱
۲۲	تعریف اصول اساسی جبر بولی	..... ۲.۱.۱
۲۳	عملیات بولی	..... ۲.۲
۲۳	متحول (variable):	..... ۲.۲.۱
۲۴	متمم complement:	..... ۲.۲.۲
۲۴	لیترال (Litrал):	..... ۲.۲.۳
۲۴	جمع بولی چند لیترال:	..... ۲.۲.۴
۲۴	ضرب بولی	..... ۲.۲.۵

۲۵	قوانین و مقررات جبر بولی	۲.۳
۲۵	قوانین تبدیلی (Commutative Laws)	۲.۳.۱
۲۵	قانون تبدیلی ضرب برای دو متحول	۲.۳.۱.۱
۲۶	قوانین انجمنی (Associative Laws)	۲.۳.۲
۲۶	قانون انجمنی ضرب برای سه متحول	۲.۳.۲.۱
۲۷	قوانین توزیعی (Distributive Laws)	۲.۳.۳
۲۷	قانون توزیعی برای ۳ متحول	۲.۳.۳.۱
۲۸	قضیه دمورگان	۲.۳.۴
۲۹	اولولیت عملگرها	۲.۳.۵
۲۹	تئوری‌های ساده سازی	۲.۴
۲۹	تئوری اتحادی	۲.۴.۱
۲۹	تئوری جذب	۲.۴.۲
۲۹	تئوری جذب سطحی	۲.۴.۳

## ۳۲ فصل سوم: گیت‌های منطقی

۳۳	گیت‌های منطقی دیجیتال	۳.۱
۳۵	گیت OR	۳.۱.۱
۳۵	گیت AND	۳.۱.۲
۳۶	گیت NOT	۳.۱.۳
۳۷	گیت NOR	۳.۱.۴
۳۸	گیت NAND	۳.۱.۵
۳۸	گیت XOR	۳.۱.۶
۳۹	گیت XNOR	۳.۱.۷
۴۱	گسترش ورودی گیت‌ها	۳.۲
۴۲	گیت‌های با چند ورودی	۳.۳
۴۳	گیت‌هایی با چند بیت ورودی	۳.۴
۴۴	تبدیل مدار به معادله	۳.۵
۴۵	تبدیل معادله به مدار	۳.۶

## ۵۰ فصل چهارم: توابع بولی

۵۱	توابع بولی	۴.۱
۵۴	مینترم‌ها و ماکسترم‌ها	۴.۲
۵۴	مینترم:	۴.۲.۱
۵۴	ماکسترم:	۴.۲.۲
۵۵	مینترم‌ها و ماکسترم‌ها برای تابع سه متحوله باینری	۴.۲.۲.۱
۵۷	جمع مینترم‌ها	۴.۳
۵۸	ضرب ماکسترم‌ها	۴.۴
۵۹	ساده سازی توابع بولی	۴.۵

۶۰	جدول کارنو.....	۴.۶
۶۰	جدول کارنو توابع ۲ متحوله.....	۴.۶.۱
۶۱	جدول کارنو توابع ۳ متحوله.....	۴.۶.۲
۶۴	جدول کارنو توابع ۴ متحوله.....	۴.۶.۳

## فصل پنجم: مدارهای ترکیبی ..... ۷۰

۷۱	مدارهای ترکیبی.....	۵.۱
۷۲	نیم جمع کننده.....	۵.۲
۷۳	تمام جمع کننده.....	۵.۳
۷۵	پیاده سازی تمام جمع کننده توسط دو نیم جمع کننده و یک گیت OR.....	۵.۳.۱
۷۷	جمع کننده چند بیتی.....	۵.۴
۷۷	جمع کننده و تفریق کننده.....	۵.۵
۷۸	دیکودر (Decoder).....	۵.۶
۷۸	دیکودر ۲ به ۴.....	۵.۶.۱
۷۹	دیکودر ۳ به ۸.....	۵.۶.۲
۸۱	دیکودر ۴ به ۱۶.....	۵.۶.۳
۸۲	اینکدر.....	۵.۷
۸۲	اینکدر ۴ به ۲.....	۵.۷.۱
۸۳	اینکدر ۸ به ۳.....	۵.۷.۲
۸۴	مولتی پلکسر.....	۵.۸
۸۴	مولتی پلکسر ۲ به ۱.....	۵.۸.۱
۸۵	مولتی پلکسر ۴ به ۱.....	۵.۸.۲
۸۶	پیاده سازی مولتی پلکسر ۴ به ۱ توسط دیکودر ۲ به ۴.....	۵.۸.۳
۸۷	دی مولتی پلکسر.....	۵.۹
۸۷	دی مولتی پلکسر ۱ به ۴.....	۵.۹.۱
۸۸	پیاده سازی مدار منطقی ترکیبی.....	۵.۱۰
۸۸	پیاده سازی تمام جمع کننده توسط دیکدر ۳ به ۸.....	۵.۱۰.۱

## فصل ششم: مدارهای ترتیبی ..... ۹۲

۹۳	مدارات ترتیبی.....	۶.۱
۹۵	لچ.....	۶.۲
۹۶	لچ SR.....	۶.۲.۱
۹۶	لچ SR توسط گیت NOR.....	۶.۲.۱.۱
۹۶	لچ SR توسط گیت NAND.....	۶.۲.۱.۲
۹۷	لچ SR با ورودی کنترل.....	۶.۲.۲
۹۸	لچ D.....	۶.۲.۳
۹۹	سمبول های لچ ها.....	۶.۲.۴
۱۰۰	فلیپ فلاپ.....	۶.۳

۱۰۱.....	فلیپ فلاپ D.....	۶.۳.۱
۱۰۱.....	فلیپ فلاپ D حساس به لبه منفی.....	۶.۳.۱.۱
۱۰۲.....	فلیپ فلاپ D حساس به لبه مثبت.....	۶.۳.۱.۲
۱۰۳.....	نمادهای ترسیمی برای فلیپ فلاپ‌های D.....	۶.۳.۲
۱۰۴.....	فلیپ فلاپ JK.....	۶.۳.۳
۱۰۶.....	فلیپ فلاپ T.....	۶.۳.۴
۱۰۶.....	فلیپ فلاپ T توسط فلیپ فلاپ JK.....	۶.۳.۴.۱
۱۰۷.....	فلیپ فلاپ T توسط فلیپ فلاپ D.....	۶.۳.۴.۲
۱۰۸.....	معادلات مشخصه فلیپ فلاپ‌ها.....	۶.۳.۵
۱۰۸.....	رجیستر.....	۶.۴
۱۱۰.....	رجیستر با بار شدن موازی.....	۶.۴.۱
۱۱۱.....	شیفت رجیستر.....	۶.۴.۲
۱۱۱.....	انتقال سریال.....	۶.۴.۳
۱۱۳.....	جمع سریال.....	۶.۴.۴
۱۱۵.....	مقایسه جمع کننده سریال و موازی.....	۶.۴.۵
۱۱۵.....	شیفت رجیسترهای یونیورسال.....	۶.۴.۶
۱۱۷.....	شمارنده.....	۶.۵
۱۱۷.....	شمارنده‌های موج گونه.....	۶.۵.۱
۱۱۹.....	شمارنده‌های همزمان.....	۶.۵.۲
۱۲۰.....	شمارنده باینری.....	۶.۵.۳
۱۲۴.....	منابع و مأخذ.....	



کمپیوترهای دیجیتال سیستمی دیجیتلی است که انواع کارهای محاسباتی را انجام می‌دهد. کلمهٔ دیجیتال بدان معناست که معلومات در کمپیوتر توسط متحول‌هایی که تعداد محدودی از مقادیر گسسته را بخود اختصاص می‌دهند، نمایش دیتا می‌شوند. این مقادیر در داخل کمپیوتر به وسیلهٔ اجزایی که می‌توانند، تعداد محدودی از حالت‌های گسسته را در خود حفظ کنند، پروسس می‌شوند. مثلاً، ارقام دسیمال ۰، ۱، ۲، ۳، ... و ۹ ده مقدار گسسته را نمایش می‌دهند. اولین کمپیوترهای الکترونیک دیجیتال که در اواخر دههٔ ۱۹۴۰ میلادی ساخته شدند، برای محاسبات عددی به کار می‌رفتند. در واقع اصطلاح کمپیوتر دیجیتال از کاربرد اعداد گرفته شده است. در عمل، کمپیوترهای دیجیتال با قابلیت بیشتری کار می‌کنند، به شرطی که تنها از دو حالت استفاده شود. به دلیل محدودیت‌های فیزیکی اجزا و نیز به علت گرایش انسان به منطق باینری (یعنی به کارگیری عبارات صحیح و غلط) قطعات دیجیتال که نگهدارندهٔ مقادیر گسسته هستند، فقط دو مقدار که باینری نامیده می‌شوند، استفاده می‌کنند.

کمپیوترهای دیجیتال از سیستم اعداد باینری استفاده می‌کنند که دو رقم بیشتر ندارند: ۰ و ۱. یک رقم باینری بیت خوانده می‌شود. معلومات در کمپیوترهای دیجیتال به وسیلهٔ گروه‌هایی از بیت‌ها نشان داده می‌شوند.

پروسسِ کمپیوتر، معمولاً به سه بخش عمده تقسیم می‌شود: واحد پروسس مرکزی <sup>۱</sup>CPU حاوی یک واحد حساب و منطق <sup>۲</sup>ALU برای کار روی دیتا، تعداد واحد رجیستر برای ذخیره کردن دیتا، واحدهای کنترل <sup>۳</sup>(CU) برای کنترل و اجرای دستورالعمل‌ها.

بر همین اساس این کتاب، دانش مقدماتی لازم برای درک عملیات سخت‌افزاری یک سیستم کمپیوتر را فراهم می‌سازد. در این کتاب با تئوری اعداد دیجیتال و گیت‌های منطقی و مدارهای ترکیبی و ترتیبی که به منظور پروسس و ذخیره سازی دیتا در سخت افزار کمپیوتر مورد استفاده قرار می‌گیرند، آشنا می‌شوید.

---

<sup>۱</sup> Central Processing Unit

<sup>۲</sup> Arithmetic and Logic Unit

<sup>۳</sup> Control Unit



### هدف کلی کتاب

آشنایی با سیستم اعداد، جبر بولی، گیت ها و انواع آن، توابع بولی، مدار های ترکیبی و مدار های ترتیبی.

# فصل اول

## سیستم اعداد



**هدف کلی:** آشنایی محصلان با سیستم اعداد.

**اهداف آموزشی:** در پایان این فصل محصلان قادر خواهند شد تا:

۱. سیستم اعداد را تعریف نماید.
۲. سیستم اعداد به قاعده دو را توضیح دهند.
۳. سیستم اعداد به قاعده هشت را توضیح دهند.
۴. سیستم اعداد به قاعده ده را توضیح دهند.
۵. سیستم اعداد به قاعده شانزده را توضیح دهند.
۶. تبدیل اعداد از یک سیستم به سیستم‌های دیگر.

سیستم‌های دیجیتال کمیت‌های گسسته معلومات که به شکل باینری نمایش داده شده‌اند، دستکاری می‌نمایند. عملوندهای<sup>۱</sup> به کار رفته در محاسبات را می‌توان در سیستم اعداد باینری بیان کرد. دیگر عناصر گسسته از جمله ارقام دسیمیل به صورت کدهای باینری نشان دیتا می‌شوند. پردازش دیتا به وسیلهٔ عناصر منطقی باینری و با استفاده از سیگنال‌های باینری انجام می‌گیرد. کمیت‌ها نیز در عناصر حافظهٔ باینری ذخیره می‌شوند. هدف این فصل معرفی مفاهیم باینری متعدد به صورت یک مرجع برای مطالعات بعدی در فصل‌های آینده است.

## ۱.۱ سیستم‌های دیجیتال

سیستم‌های دیجیتال در زندگی روزانه بشر نقش برجسته‌یی دارند و به این دلیل، ما دورهٔ تکنالوژی فعلی را عصر دیجیتال می‌خوانیم. سیستم‌های دیجیتال در مخابرات، تجارت، کنترل ترافیک، هدایت سفینه‌های فضایی، عمل جراحی، هواشناسی، اینترنت و بسیاری از دیگر زمینه‌های تجاری، صنعتی و علمی به کار می‌روند. ما از تلفن‌های دیجیتال، تلویزیون‌های دیجیتال، دیسک‌های چند منظوره دیجیتال، دوربین‌های دیجیتال و کمپیوترهای دیجیتال استفاده می‌کنیم. مهم‌ترین خاصیت یک کمپیوتر دیجیتال، همگانی بودن آن است. کمپیوتر می‌تواند، رشته‌یی از دستورات، به نام برنامه را که روی دیتاهای مفروض عمل می‌کنند، دنبال نماید. یوزر<sup>۲</sup> می‌تواند، برنامه یا دیتای خود را طبق نیاز انتخاب و اجراء کند. به علت این انعطاف، کمپیوترهای همه منظوره دیجیتال می‌توانند، عملیات پردازش معلومات را در محدودهٔ وسیعی از کاربردها انجام دهند. یکی از ویژگی‌های سیستم دیجیتال توان‌مندی آن‌ها در دستکاری عناصر گسسته معلوماتی است. هر مجموعه‌یی که به تعداد متناهی از عناصر محدود باشد، معلومات گسسته را دارا است. مثال‌هایی از عناصر گسسته، عبارتند از: ۱۰ رقم دسیمیل، ۲۶ حرف الفباء، ۵۲ ورق بازی، ۶۴ مربع بازی شطرنج. کمپیوترهای دیجیتال اولیه، برای محاسبات عددی به کار می‌رفتند. در این حال، عناصر گسسته‌یی به کار رفته، ارقام بودند. نام دیجیتال یا رقمی از این مفهوم حاصل شده است. عناصر گسسته معلوماتی در یک سیستم دیجیتال با کمیت‌های فیزیکی به نام سیگنال نشان داده می‌شوند. رایج‌ترین سیگنال‌های الکتریکی، عبارتند از: ولتاژ و جریان. وسایل الکترونیکی به نام ترانزیستور در مدارهایی که این سیگنال‌ها را پیاده سازی می‌کنند به طور چشم‌گیری به کار می‌روند. سیگنال‌ها، در بسیاری از سیستم‌های دیجیتال الکترونیک امروزی تنها دو مقدار را دارا هستند، پس باینری‌اند. یک رقم باینری که مقدار ۰ و ۱ را دارند. عناصر گسسته معلوماتی با گروهی از بیت‌ها، به نام کدهای باینری نمایش داده می‌شوند. مثلاً، ارقام دسیمیل ۰ تا ۹ در سیستم اعداد دیجیتال با کد چهار بیتی نشان داده می‌شوند. با به کارگیری تکنیک‌های مختلف، گروه‌هایی از بیت‌ها برای نمایش سمبول‌های گسسته تعریف می‌شوند و سپس در توسعهٔ یک سیستم در قالب دیجیتال مورد استفاده قرار می‌گیرد. در نتیجه، یک سیستم دیجیتال سیستمی است که عناصر گسسته معلوماتی به شکل باینری تنظیم می‌کند.

---

<sup>۱</sup>Operand

<sup>۲</sup>User

کمیت‌های معلوماتی یا ذاتاً گسسته‌اند و یا از نمونه برداری کوانتیزه کردن فرآیندهای پیوسته حاصل می‌شوند. به عنوان مثال، یک لیست حقوق، یک فرآیند یا رویداد گسسته بوده و حاوی نام کارمند، شماره تأمین اجتماعی، حقوق هفتگی، مالیات بر درآمد و غیره است. پرداخت به یک کارمند با استفاده از مقادیر دیتای گسسته، مانند حروف الفبایی (نام‌ها)، ارقام (حقوق) و نمادها یا سمبول‌های خاص (مانند \$) پردازش می‌شود. از طرف دیگر یک محقق ممکن است یک پدیده را به صورت پیوسته مشاهده کند، ولی فقط مقادیر خاصی را به صورت جدول ثبت نماید. بنابراین، فرد محقق دیتای پیوسته را نمونه برداری می‌نماید، ولی هر کمیت در جدول را از عناصر گسسته می‌سازد. در بسیاری از حالات، نمونه برداری از یک فرآیند به طور خودکار به وسیله دستگاهی به نام مبدل آنالوگ به دیجیتال انجام می‌شود.

بهترین مثال از یک سیستم دیجیتال، کمپیوتر دیجیتال همه منظوره است. بخش‌های اصلی یک کمپیوتر، عبارتند از: واحد حافظه، واحد پردازش مرکزی و واحدهای ورودی و خروجی. واحد حافظه، برنامه‌ها و دیتاهای وارده، خارج شونده و میانی را ذخیره می‌کند. واحد پردازش مرکزی، اعمال محاسباتی و دیگر عملیات روی دیتاها را بر حسب آن‌چه در برنامه مشخص شده، انجام می‌دهد. دیتاها و برنامه‌هایی که به وسیله استفاده کننده آماده شده‌اند، توسط وسایل ورودی مانند، صفحه کلید به حافظه انتقال می‌یابند. یک وسیله خروجی مثل چاپ‌گر، نتایج حاصل از محاسبات را دریافت کرده و به استفاده کننده ارائه می‌دهد. یک کمپیوتر دیجیتال می‌تواند به چندین وسیله ورودی و خروجی وصل شود. یکی از وسایل مفید در این مورد واحد مخابره است که تبادل دیتا را از طریق اینترنت با دیگر استفاده کنندگان برقرار می‌سازد. یک کمپیوتر دیجیتال دستگاهی توان‌مندی است که نه تنها می‌تواند، محاسبات ریاضی را انجام دهد، بلکه قادر است اعمال منطقی را هم اجراء نماید. به علاوه می‌تواند، جهت تصمیم‌گیری براساس شرایط داخلی یا خارجی برنامه‌ریزی شود.

برای استفاده از مدارهای دیجیتال در تولیدات تجاری دلایل اساسی وجود دارد، کمپیوترهای دیجیتال، دستگاه‌های دیجیتال نیز قابل برنامه‌ریزی‌اند. با تعویض برنامه در وسیله برنامه‌پذیر، سخت افزار یگانه‌یی قابل استفاده در کاربردهای متفاوت است. کاهش قیمت در وسایل دیجیتال به دلیل پیشرفت در تکنالوژی مدارهای مجتمع دیجیتال مرتباً روی می‌دهد. با افزایش تعداد ترانزیستورها در یک قطعه سیلیکان، توابع پیچیده‌تری قابل پیاده‌سازی شده، قیمت هر واحد کاهش یافته و قیمت دستگاه‌های دیجیتال روز بروز کاهش می‌یابد.

دستگاه‌های ساخته شده با مدارهای مجتمع می‌توانند با سرعتی تا صد میلیون عمل در ثانیه را انجام دهند. می‌توان با استفاده از کدهای اصلاح خطا، عملکرد سیستم‌های دیجیتال را به شدت اطمینان بخش نمود. مثالی از این نوع، دیسک چند کاره دیجیتال (DVD)<sup>۱</sup> است که در آن معلومات ویدیویی، صوتی بدون از دست رفتن یک قلم دیتا، ضبط می‌شود. معلومات دیجیتال در DVD چنان ضبط می‌شود که هر کد در هر نمونه دیجیتال قبل از نمایش به‌طور خودکار خطایابی شده و اصلاح می‌شود.

یک سیستم دیجیتال از بهم پیوستن مدل‌های دیجیتال بدست می‌آید. برای درک عمل هر مدل، دانش و آگاهی مدارهای دیجیتال و عمل منطقی آن‌ها لازم است. این کتاب ابزار اصلی طراحی دیجیتال، مانند:

<sup>۱</sup> Digital Versatile Disk

ساختارهای منطقی گیتی، مدارهای ترکیبی و ترتیبی و وسایل منطقی برنامه پذیر را ارائه می‌کند. طراحی دیجیتال را در سطح انتقال بین رجیستر معرفی می‌نماید. در مورد مدارهای ترتیبی غیرهم‌زمان (آسنکرون یا غیر همگام) و دیگر خانواده‌های منطقی دیجیتال مجتمع بحث می‌کند. مدارهای مجتمع تجاری را معرفی کرده، نشان می‌دهد که چگونه در یک آزمایشگاه برای انجام آزمایشات به هم وصل می‌شوند.

یک گرایش مهم در طراحی دیجیتال، استفاده از زبان توصیف سخت افزاری (HDL)<sup>۱</sup> است. (HDL) نوعی زبان برنامه‌ریزی است که برای توصیف مدارهای دیجیتال به صورت متن به کار می‌رود. این زبان برای شبیه سازی یک سیستم دیجیتال و اطمینان از صحت عمل آن قبل از ساخت مورد استفاده قرار می‌گیرد. HDL در کنار ابزارهای طراحی منطقی دیگر، برای خودکارسازی طراحی استفاده می‌شود.

## ۱.۲ سیستم اعداد

اعدادی که در عصر حاضر به‌طور وسیعی از آنها استفاده می‌کنیم، شاید در حدود ۱۰ تا ۱۲ هزار سال پیش به‌وجود آمده‌اند و بعدها برای شمارش این اعداد، اسم‌ها و قوانینی وضع شد. گسترش شمارش اعداد در قاعده‌های مختلف، سیستم‌های مختلفی را ایجاد کرد که در حال حاضر هر یک از این سیستم‌ها در موارد خاصی مورد استفاده قرار می‌گیرند.

یکی از قاعده‌هایی که از زمان قدیم تا کنون مورد استفاده قرار گرفته است، قاعده ۱۰ (ده دهی) است که بر قاعده شماره انگشتان دست‌ها بوده و چنین ترتیب ذهنی را برای آنها به‌وجود آورده‌اند.

### سیستم اعداد به قاعده ده (Decimal)

کلمه Decimal (اعشاری) از کلمه لاتین Decem به معنای ده و هم‌چنان کلمه مصری Deka به معنای ده گرفته شده است. سیستم اعداد به قاعده ده از ده علامت ۰ و ۱ و ۲ و ۹... تشکیل شده‌اند. برای شمارش از ۰ تا ۹ از این علامت‌ها استفاده می‌کنیم و برای نشان دادن اعداد بزرگ‌تر از ۹، این علامت‌ها را طبق قواعد خاصی با یک دیگر ترکیب می‌کنیم (پشت سر هم قرار می‌دهیم). چنان‌که می‌دانید، موقعیت مکانی هر عدد (هر علامت) یا رقم معنای خاصی دارد؛ مثلاً با دو رقم ۶ و ۴ دو عدد ۴۶ و ۶۴ را می‌توان ساخت که از نظر معنا با هم متفاوت‌اند. در سیستم اعشاری، هر عدد را می‌توان به صورت توان‌هایی از ۱۰ نشان داد؛ به این دلیل به آنها سیستم اعشاری می‌گویند؛ مثلاً:


$$3296 = 3000 + 200 + 90 + 6 = 3 \times 10^3 + 2 \times 10^2 + 9 \times 10^1 + 6 \times 10^0$$

به طور کلی، در سیستم اعشاری هر عدد صحیح را می‌توان به صورت زیر نوشت.

$$N = a_n \times 10^n + a_{n-1} \times 10^{n-1} + \dots + a_2 \times 10^2 + a_1 \times 10^1 + a_0 \times 10^0$$

<sup>۱</sup> Hardware Description Language

ضرایب  $a_1, a_2, \dots, a_{n-1}, a_n$  و  $a$  می‌توانند بین صفر تا ۹ باشند. توان‌های ۱۰ ارزش مکانی هر یک از رقم‌ها را مشخص می‌کند، مثلاً:

$$45531 = 4 \times 10^4 + 5 \times 10^3 + 5 \times 10^2 + 3 \times 10^1 + 1 \times 10^0$$


یک‌ها   ده‌ها   صدها   هزارها   ده‌هزارها

در عدد ۴۵۵۳۱، رقم ۴ مربوط به  $a_n$ ، رقم ۵ مربوط به  $a_{n-1}$ ، رقم ۵ صدها مربوط به  $a_{n-2}$ ، رقم ۳ مربوط به  $a_{n-3}$  و رقم ۱ مربوط به  $a_{n-4}$  است. در این مثال  $n=4$  است در نتیجه  $a_n = a_4$ ،  $a_{n-1} = a_3$  و  $a_{n-4} = a_1$  می‌شود.

### سیستم اعداد به قاعدهٔ دو (Binary)

در سیستم اعداد به قاعدهٔ دو، ارقام به کار رفته ۰ و ۱ (دوتا) هستند. برای شمارش صفر و یک از این علامت‌ها استفاده می‌کنیم و برای نمایش دادن اعداد بزرگ‌تر از یک، این دو علامت را طبق قواعد خاصی پشت سر هم قرار می‌دهیم. در این سیستم هر علامت متناسب با مکانی که در آن قرار می‌گیرد (یا موقعیت رقم)، ارزش خاصی پیدا می‌کند. به طور کلی در سیستم اعداد به قاعدهٔ دو هر عدد را می‌توان به صورت زیر نوشت:

$$N = a_n \times 2^n + a_{n-1} \times 2^{n-1} + \dots + a_2 \times 2^2 + a_1 \times 2^1 + a_0 \times 2^0$$

در این جا ضرایب  $a_1, a_2, \dots, a_n$  می‌توانند صفر یا یک باشند. در سیستم اعداد به قاعدهٔ دو به هر رقم صفر یا یک، یک بیت (Binary Digit=Bit) می‌گویند، مثلاً: عدد ۱۱۰۱ یک عدد چهار بیتی است.

در گذشته به هر چهار بیت یک نیبل (Nibble) می‌گفتند و در حال حاضر به هر هشت بیت یک بایت (Byte) گفته می‌شود. واحد بزرگ‌تر از بایت، کیلوبایت معادل  $2^{10}$  بایت یا ۱۰۲۴ بایت و میگابایت معادل  $2^{20}$  بایت یا ۱۰۲۴ کیلوبایت است.

می‌دانیم که در یک سیستم باینری ارزش اولین بیت برابر یک، ارزش دومین بیت برابر ۲ (دو برابر رقم قبل)، ارزش بیت سومین بیت برابر ۴ (دو برابر رقم قبل) و ارزش چهارمین بیت برابر ۸ (دو برابر رقم قبل) و .... است.

$$(b_0 \ b_1 \ b_2 \ b_3 \ b_4 \ \dots)_2 \text{ عدد باینری در حالت کلی}$$

$$(1 \ 2 \ 4 \ 8 \ 16 \ 32 \ \dots)_2 \text{ ارزش مکانی بیت‌ها}$$

$$(2^0 \ 2^1 \ 2^2 \ 2^3 \ 2^4 \ 2^5 \ \dots)_2 \text{ ارزش مکانی بیت با قاعدهٔ دو}$$

مثال: عدد باینری 10011، دارای ارزش مکانی به صورت زیر است.

$$10011 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

در یک عدد باینری بیت اول از سمت راست کم ارزش‌ترین بیت است که به آن (Least Significant Bit) و به آخرین بیت در سمت چپ که با ارزش‌ترین بیت است (MSB (Most Significant Bit گفته می‌شود.

### سیستم اعداد به قاعدهٔ هشت (Octal)

در سیستم اکتال (هشت) قاعدهٔ عدد ۸ و تعداد هشت رقم به صورت (۰ و ۱ و ۲ و ... و ۷) است. برای نمایش دادن اعداد از صفر تا هفت، از این علامت‌ها استفاده می‌شود. برای اعداد بزرگ‌تر از هفت، این علامت‌ها را طبق قواعد خاصی پشت سر هم قرار می‌دهیم. در این سیستم مانند سیستم دسیمال، هر عدد موقعیت خاص خود را دارد. معادل اعشاری اعداد اکتال مشابه اعداد باینری از رابطهٔ زیر به دست می‌آید. با این تفاوت که به جای عدد ۲، عدد ۸ قرار می‌گیرد.

$$N = a_n \times 8^n + a_{n-1} \times 8^{n-1} + \dots + a_2 \times 8^2 + a_1 \times 8^1 + a_0 \times 8^0$$

ضرایب  $a_n$  تا  $a_0$  می‌توانند مقادیری بین صفر تا هفت باشند، مثلاً: عدد اکتال  $(5236)_8$  در رابطه فوق طوری ذیل تطبیق می‌شود.

$$5236 = 5 \times 8^3 + 2 \times 8^2 + 3 \times 8^1 + 6 \times 8^0$$

### سیستم اعداد به قاعدهٔ شانزده (Hexa Decimal)

در سیستم شانزده، ۱۶ رقم شامل (۰، ۱، ۲، ...، ۹، A، B، C، D، E و F) به کار می‌رود. در این سیستم برای نمایش عددهای بیشتر از ۹ و کمتر از ۱۶ باید از یک حرف استفاده کرد و نمی‌توان مثلاً عدد ۱۰ را به همین صورت نشان داد؛ چون یک عدد دو رقمی است که هم صفر و هم یک دارد و با صفر و یک اصلی اشتباه می‌شود؛ به همین دلیل از حروف استفاده می‌شود که:

$$A=10, B=11, C=12, D=13, E=14, F=15$$

برای اعداد بزرگ‌تر از ۱۶ این ارقام را طبق قواعد خاصی پشت سر هم قرار می‌دهیم. مشابه همان قواعدی که در سیستم اکتال بیان شد، با این تفاوت که قاعده در این جا عدد ۱۶ است. در این سیستم اعداد نیز، هر عدد موقعیت خاص خود را دارد. معادل اعشاری اعداد هیگزادسیمال از رابطهٔ زیر بدست می‌آید.

$$N = a_n \times 16^n + a_{n-1} \times 16^{n-1} + \dots + a_2 \times 16^2 + a_1 \times 16^1 + a_0 \times 16^0$$

ضرایب  $a_n$  تا  $a_0$  می‌توانند مقادیری بین صفر تا ۱۵ باشند، مثلاً عدد  $(A14E)_{16}$  در قاعده ۱۶ نوشته نمایید.

$$A14E = A \times 16^3 + 1 \times 16^2 + 4 \times 16^1 + E \times 16^0 = 10 \times 16^3 + 1 \times 16^2 + 4 \times 16^1 + 14 \times 16^0$$



### ۱.۳ روش‌های تبدیل قاعده‌های اعداد

وقتی که ما بیش‌تر از یک سیستم عددی داریم، تبدیل اعداد از یک سیستم به سیستم دیگر بسیار مهم است. برای ما آسان‌تر است که با اعداد دسیمال سروکار داشته باشیم؛ ولی در سیستم‌های دیجیتالی اعداد باینری بیش‌تر به کار می‌رود.

از طرفی، ما هم به اعداد دسیمال احتیاج داریم و هم به اعداد باینری، زیرا کمپیوتر اعداد باینری را می‌شناسد در صورتی که روی صفحه کمپیوتر باید اعداد دسیمال ظاهر شود. در نتیجه همواره در سیستم‌های دیجیتالی تبدیل اعداد دسیمال به اعداد باینری در مورد اطلاعات ورودی و اما برعکس، تبدیل اعداد باینری به اعداد دسیمال در مورد اطلاعات خروجی مورد نیاز است.

اکثر سیستم‌های دیجیتالی با اعداد در سیستم باینری کار می‌کنند. هم‌چنین استفاده از سیستم اعداد در قاعده اکتال (۲³) و هیگزادسیمال (۲⁴) که به صورت توان‌هایی از ۲ نوشته می‌شوند، در ساده کردن این تبدیلات بسیار موثر هستند.

#### ۱.۳.۱ تبدیل اعداد از قاعده ۲ به قاعده ۱۰

از سمت راست عدد باینری شروع می‌کنیم. اولین رقم باینری در ۲ به توان صفر، رقم دوم در ۲ به توان ۱، رقم سوم در ۲ به توان ۲، رقم چهارم در ۲ به توان ۳ و به همین ترتیب توان‌های ۲ یکی یکی بالا رفته و حاصل این ضرب‌ها را با هم جمع کرده که برابر با معادل دسیمال عدد باینری می‌شود.

$$(111000)_2 = 32 + 16 + 8 + 0 + 0 + 0 = (56)_{10}$$

جدول (۱-۰): تبدیل از باینری به دسیمال

حاصل ضرب رقم در توان‌های ۲	
$0 * 2^0 =$	0
$0 * 2^1 =$	0
$0 * 2^2 =$	0
$1 * 2^3 =$	8
$1 * 2^4 =$	16
$1 * 2^5 =$	32
$32+16+8+0+0+0$	56 نتیجه جمع

### ۱.۳.۲ تبدیل اعداد از قاعده ۲ به قاعده ۸

تبدیل از باینری به اوکتال به سادگی با تفکیک عدد باینری به گروه‌های سه رقمی در دو طرف نقطه باینری بدست می‌آید. سپس به هر گروه یک رقم قاعده هشت تعلق می‌گیرد. مثال زیر روال مربوطه را نشان می‌دهد:

$$\underbrace{(10)}_2 \underbrace{110}_6 \underbrace{001}_1 \underbrace{101}_5 \underbrace{011}_3 . \underbrace{111}_7 \underbrace{100}_4 \underbrace{000}_0 \underbrace{110}_6)_2 = (26153,7406)_8$$

### ۱.۳.۳ تبدیل اعداد از قاعده ۲ به قاعده ۱۶

تبدیل از قاعده دو به قاعده شانزده نیز مشابه با روند فوق است، با این تفاوت که عدد باینری به گروه‌های چهار رقمی تفکیک می‌شوند:

$$\underbrace{(10)}_2 \underbrace{1100}_C \underbrace{0110}_6 \underbrace{1011}_B . \underbrace{1111}_F \underbrace{0010}_2)_2 = (2C6B.F2)_8$$

### ۱.۳.۴ تبدیل اعداد از قاعده ۱۰ به قاعده ۲

عدد ۵۶ را به باینری تبدیل کنید. ابتدا ۵۶ را بر ۲ تقسیم می‌کنیم تا خارج قسمت ۲۸ و باقی‌مانده ۰ بدست آید. خارج قسمت، مجدداً بر ۲ تقسیم می‌شود تا خارج قسمت و باقی‌مانده جدیدی بدست آید. باقی‌مانده‌ها به ترتیب به دست آمده و از سمت راست عدد باینری نوشته می‌شود. این روال تا رسیدن به خارج قسمت ۰ ادامه می‌یابد. ضرایب عدد باینری مورد نظر به طریق زیر از باقی‌مانده‌ها بدست می‌آید:

جدول (۲-۱): تبدیل از دسیمال به باینری

ضریب عدد باینری	باقی‌مانده		خارج قسمت	
$a_0 = 0$	0	+	28	$56/2=$
$a_1 = 0$	0	+	14	$28/2=$
$a_2 = 0$	0	+	7	$14/2=$
$a_3 = 1$	1	+	3	$7/2=$
$a_4 = 1$	1	+	1	$3/2=$
$a_5 = 1$	1	+	0	$1/2=$

$$(56)_{10} = (a_5 a_4 a_3 a_2 a_1 a_0)_2 = (111000)_2 \text{ جواب:}$$

### ۱.۳.۵ تبدیل اعداد از قاعده ۱۰ به قاعده ۸

عدد داده شده را به ترتیب تا زمانی تقسیم ۸ می‌نمایم که خارج قسمت صفر شود و باقی مانده‌های هر مرحله تقسیم را به ترتیب در یک ستون می‌نویسیم و در اخیر باقی مانده‌ها را به ترتیب برعکس (از آخر به طرف اول) در یک سطر نوشته می‌کنیم. در نتیجه معادل اوکتال عدد داده شده به دست خواهد آمد.

مثال: عدد  $(177)_{10}$  را به قاعده هشت تبدیل نمایید.

$$177/8=22 \text{ باقی مانده } \rightarrow 1$$

$$22/8=2 \text{ باقی مانده } \rightarrow 6$$

$$2/8=0 \text{ باقی مانده } \rightarrow 2$$

$$(177)_{10}=(261)_8$$

### ۱.۳.۶ تبدیل اعداد از قاعده ۱۰ به قاعده ۱۶

عدد داده شده را به طور متواتر تقسیم ۱۶ می‌کنیم و باقی مانده‌ها را به ترتیب در یک ستون می‌نویسیم و در اخیر باقی مانده‌ها را به ترتیب برعکس از آخرین باقی مانده به طرف اولین باقی مانده نوشته می‌کنیم. به قاعده ده تبدیل می‌شود.

مثال: عدد  $(4768)_{10}$  را به سیستم شانزده تبدیل نمایید.

$$4768/16=298 \text{ باقی مانده } \rightarrow 0$$

$$298/16=18 \text{ باقی مانده } \rightarrow 10=(A)$$

$$18/16=1 \text{ باقی مانده } \rightarrow 2$$

$$1/16=0 \text{ باقی مانده } \rightarrow 1$$

$$(4768)_{10}=(12A0)_{16}$$

### ۱.۳.۷ تبدیل اعداد اوکتال به باینری

تبدیل از قاعده هشت یا شانزده به باینری با روشی عکس روش بالا انجام می‌شود. هر رقم قاعده هشت با سه رقم قاعده دو معادل خود جای‌گزین می‌شود.

$$(673.124)_8 = (\underbrace{110}_6 \underbrace{111}_7 \underbrace{011}_3 \underbrace{.001}_1 \underbrace{010}_2 \underbrace{100}_4)_2$$

### ۱.۳.۸ تبدیل اعداد از قاعده ۸ به قاعده ۱۰

عدد داده شده را قرار موقعیت آن ضرب یک از توان‌های هشت مربوط نموده و حاصل ضرب‌ها را جمع می‌نماییم؛ در نتیجه عدد معادل آن در سیستم ده به دست خواهد آمد. هر عدد نظر به موقعیت خود ضرب یکی از طاقت‌های هشت خواهد شد.

مثال: عدد  $(632)_8$  را به قاعده ده تبدیل نمایید.

$$632=6 \times 8^2 + 3 \times 8^1 + 2 \times 8^0 = 6 \times 64 + 3 \times 8 + 2 \times 1 = 384 + 24 + 2 = (410)_{10}.$$

### ۱.۳.۹ تبدیل اعداد از قاعده ۸ به قاعده ۱۶

عدد داده شده را اولاً از سیستم هشت به سیستم ده تبدیل نموده، ثانیاً عدد بدست آمده (سیستم دو) را به سیستم ۱۶ طور محتوای عنوان تبدیل اعداد از قاعده ۲ به قاعده ۱۶ انجام می‌نماییم.

مثال: عدد  $(۲۵۶)_۸$  را به قاعده ۱۶ تبدیل نمایید.

$$(۲۵۶)_{۸} = (۰۱۰,۱۰۱,۱۱۰)_{۲} = (۰۱۰۱۰۱۱۱۰)_{۲}$$

$$(۰۱۰۱۰۱۱۱۰)_{۲} = ۰۰۰۰۱۰۱۰۱۱۱۰ = ۰۱۰(A)۱۴(E) = (۰AE)_{۱۶}$$

### ۱.۳.۱۰ تبدیل اعداد از قاعده ۱۶ به قاعده ۲

به طور مشابه، هر رقم قاعده شانزده با چهار رقم باینری معادلش جای‌گزین خواهد شد. این مطلب در مثال‌های زیر تشریح شده است:

$$(306.D)_{16} = (\underbrace{0011}_3 \underbrace{0000}_0 \underbrace{0110}_6 . \underbrace{1101}_D)_{2}$$

### ۱.۳.۱۱ تبدیل اعداد از قاعده ۱۶ به قاعده ۸

عدد داده شده را اولاً از سیستم شانزده به سیستم ده تبدیل نموده، ثانیاً عدد بدست آمده (سیستم ده) را به سیستم ۸ طور محتوای عنوان تبدیل اعداد از قاعده ۱۰ به قاعده ۸ انجام می‌نماییم.

مثال: عدد  $(AE)_{۱۶}$  را به سیستم هشت تبدیل نمایید.

$$AE = A(10) E(14) = 1010 \ 1110 = (10101110)_2$$

$$10101110 = 010 \ 101 \ 110 = (256)_8$$

### ۱.۳.۱۲ تبدیل اعداد از قاعده ۱۶ به قاعده ۱۰

برای تبدیل کردن یک عدد از سیستم شانزده به سیستم ده، هر عدد را نظر به موقعیت آن به یکی از توان‌های شانزده ضرب نموده و حاصل ضرب‌ها را باهم جمع می‌نماییم، در نتیجه حاصل جمع عبارت از معادل عدد داده شده به سیستم ده می‌باشد.

توان‌های معمول ۱۶ عبارت‌اند از:

$$16^0=1, 16^1=16, 16^2=256, 16^3=4096, 16^4=65536, \dots$$

مثال: عدد  $(F4C)_{۱۶}$  را به سیستم ده تبدیل کنید.

$$F4C = FX16^2 + 4X16^1 + CX16^0 = 15X256 + 4X16 + 12X1 = 3840 + 64 + 12 = (3916)_{10}$$

پس از مطالعه مقادیر لیست شده در جدول (۴-۱)، به سادگی می‌توان رقم قاعدهٔ شانزده را برای هر گروه از ارقام باینری به خاطر سپرد.

جدول (۳-۱): اعداد در قاعده‌های ۲ و ۸ و ۱۶

شانزده تایی (قاعده 16)	هشتایی (قاعده 8)	باینری (قاعده 2)	دسیمال (قاعده 10)
0	00	0000	00
1	01	0001	01
2	02	0010	02
3	03	0011	03
4	04	0100	04
5	05	0101	05
6	06	0110	06
7	07	0111	07
8	10	1000	08
9	11	1001	09
A	12	1010	10
B	13	1011	11
C	14	1100	12
D	15	1101	13
E	16	1110	14
F	17	1111	15

## ۱.۴ کاربرد قاعده‌ها

اساس کار با اعداد باینری، به دلیل این که تعداد ارقام‌شان سه یا چهار برابر معادل‌شان در قاعده ده می‌باشد، مشکل است. مثلاً، عدد باینری ۱۱۱۱۱۱۱۱۱۱ معادل ۴۰۹۵ است. با این وجود کمپیوترهای دیجیتال اعداد باینری را به کار می‌برند و گاهی نیز لازم است تا استفاده کننده مستقیماً به وسیله اعداد باینری با ماشین ارتباط برقرار کند. یک راه برای حفظ سیستم باینری در کمپیوتر که در ضمن تعداد ارقام را برای انسان کاهش می‌دهد، استفاده از رابطه بین سیستم اعداد باینری و اوکتال یا شانزده است. با این روش، انسان برحسب اعداد قاعده هشت یا شانزده فکر کرده و در مواقعی که ارتباط مستقیم با ماشین لازم است، تبدیل لازمه را با بررسی این اعداد انجام خواهد داد. به این ترتیب عدد باینری (۱۱۱۱۱۱۱۱۱۱) که دارای ۱۲ رقم است در قاعده هشت به صورت چهار رقم (۷۷۷۷) و یا در قاعده شانزده به شکل (FFF) در می‌آید. به هنگام تبادل معلومات با انسان، نمایش قاعده هشت یا شانزده اعداد باینری مطلوب‌تر است، زیرا که در این قاعده‌ها اعداد با (۱/۳ یا ۱/۴) تعداد ارقام‌شان در باینری قابل نمایش‌اند. بنابراین، اغلب کتابچه‌های راهنمای کمپیوتر از اعداد قاعده هشت یا شانزده برای نمایش کمیت‌های باینری استفاده می‌کنند. گرچه نمایش قاعده شانزده مناسب‌تر به نظر می‌رسد، ولی انتخاب یکی از این دو کاملاً اختیاری است.

## ۱.۵ متمم‌ها (Complements)

متمم‌ها در کمپیوترهای دیجیتال، برای ساده کردن عمل تفریق و یا عملیات منطقی به کار می‌روند. در هر قاعده چون، دو نوع متمم وجود دارد. یکی متمم قاعده و دیگری متمم قاعده کاهش یافته است. شکل اول به متمم و دومی به متمم  $r - 1$  موسوم است. وقتی که مقدار قاعده یا پایه را جای‌گزین کنیم، برای اعداد باینری، متمم‌های ۲ و ۱ و برای اعداد دسیمال، متمم‌های ۱۰ و ۹ را خواهیم داشت.

### ۱.۵.۱ متمم $r-1$ (One's Complement)

با فرض داشتن عددی  $n$  رقمی، مانند:  $N$  در قاعده ۲، متمم  $(r - 1)$  عدد به صورت  $(r^n - 1)$  تعریف می‌شود. برای اعداد دسیمال،  $r = 10$  و  $r - 1 = 9$  است و به این ترتیب متمم ۹ عدد  $N$  برابر  $N - (10 - 1)$  خواهد بود. در این جا ۱۰ نمایشگر عددی است که متشکل از یک ۱ و به دنبال آن  $n$  عدد ۰ می‌باشد. ۱۰ - ۱ عددی است که با  $n$  عدد ۹ نشان داده می‌شود. مثلاً، اگر  $n = 4$  باشد،  $104 = 10000$  و  $9999 = 104 - 1$  داریم. به این ترتیب نتیجه می‌شود که متمم و یک عدد دسیمال با تفریق هر رقم از ۹ حاصل خواهد شد. به چند مثال عددی زیر توجه کنید:

متمم ۹ عدد ۵۴۶۷۰۰ برابر است با  $453299 = 546700 - 99999$

متمم ۹ عدد ۰۱۲۳۹۸ برابر است با  $987601 = 012398 - 99999$

برای اعداد باینری،  $r = 2$  و  $r - 1 = 1$  است، بدین ترتیب متمم 1 عدد  $N$ ،  $(2^n - 1) - N$

خواهد بود. مجدداً،  $2^n$  برابر با یک عدد باینری است که از یک 1 و  $n$  عدد 0 تشکیل شده است.  $2^n - 1$  یک عدد باینری متشکل از  $n$  عدد 1 می‌باشد. مثلاً، اگر  $n = 4$  باشد، داریم  $(10000)_2$  و

$(1111)_2 = 2^4 - 1$  بنابراین، متمم 1 یک عدد باینری از تفریق هر رقم از 1 بدست می‌آید. با این وجود، هنگام تفریق ارقام باینری از 1، یکی از دو حالت  $1 - 0 = 1$  و  $1 - 1 = 0$  را خواهیم داشت، که سبب می‌شود، هر بیت از 0 به 1 و از 1 به 0 تبدیل شود. بنابراین، متمم یک عدد باینری با تغییر 1 ها به 0 و 0 ها به 1 حاصل می‌شود. در زیر مثال‌هایی آورده شده است:

متمم 1 عدد 1011000 برابر است با 0100111

متمم 1 عدد 0101101 برابر است با 1010010

متمم اعداد  $(r - 1)$  هشت و شانزده به ترتیب از تفریق ارقام آن‌ها از 7 یا (F 15 دسیمال) حاصل می‌شود.

## ۱.۵.۲ متمم r (Two's Complement)

متمم ۲ یک عدد  $n$  رقمی، مانند  $N$  در قاعده ۲ به صورت  $r - N$  به  $N + 0$  در  $N = 1$  و برابر با 0 در  $N = 1$  تعریف می‌شود. از مقایسه این متمم با متمم  $(r - 1)$  نتیجه می‌شود که متمم  $r$  از جمع 1 با متمم  $(r - 1)$

حاصل می‌شود. زیرا  $1 + [(r^n - 1) - N] = r^n - N$  می‌باشد. به این ترتیب متمم 10 یک عدد دسیمال، مانند 2389 برابر است با  $7611 + 1 = 7610$  که از جمع 1 به مقدار متمم 9 حاصل می‌شود. متمم 2 عدد باینری 101100 برابر است با  $010100 + 1 = 010011$  و از جمع 1 با مقدار متمم 1 بدست می‌آید.

چون  $10^n$  عددی است که با یک 1 و  $n$  عدد 0 به دنبال آن نمایش دیتا می‌شود،  $10^n - N$  که متمم 10 عدد  $N$  است نیز با تغییر ندادن 0 های کم ارزش‌تر و تفریق اولین رقم غیر صفر کم ارزش‌تر از 10 و تفریق همه رقم‌های با ارزش‌تر از 9 حاصل می‌شود.

متمم 10 عدد 012398 برابر 987602 می‌باشد.

متمم 10 عدد 246700 برابر 753300 است.

متمم 10 اولین عدد با تفریق 8 از 10 در کم ارزش ترین مکان و تفریق دیگر ارقام از 9 حاصل شده است. متمم 10 دومین عدد بدین ترتیب حاصل گشته است که دو 0 کم ارزش تر رها می شوند، 7 از 10 و دیگر ارقام از 9 تفریق می گردند.

به طور مشابه، متمم 2 می تواند با رها کردن همه اهای کم ارزش تر و نیز تغییر نکردن اولین او جای گزینی همه اها با 1 ها و 1 ها با 0 ها در دیگر ارقام با ارزش تر حاصل شود.

متمم 2 عدد 1101100 برابر 0010100 است.

متمم 2 عدد 0110111 برابر 1001001 است. متمم 2 اولین عدد با رها کردن دو کم ارزش تر و اولین او سپس جای گزینی همه 1 ها با 0 و 0 ها با 1 در چهار رقم با ارزش تر باقی مانده بدست می آید. متمم 2 دومین عدد با رها کردن اولین و متمم کردن دیگر ارقام حاصل می شود.

در تعاریف قبلی، فرض شد که اعداد دارای نقطه ممیز نیستند. اگر عدد اولیه  $N$  حاوی ممیز باشد، باید آن را موقتاً حذف نمود، تا متمم  $r$  و  $(r-1)$  بدست آید. آن گاه، آن را به مکان مربوطه اش باز می گردانیم. توجه داشته باشید که متمم یک متمم، عدد را به حالت اولیه اش باز می گرداند. متمم ۲ عدد  $N$  برابر  $r^n - N$  است. متمم یک متمم برابر است با  $r^n - (r^n - N) = N$  که همان عدد اولیه است.

### ۱.۵.۳ تبدیل تفریق به جمع با استفاده از متمم ۱

عدد اول را نوشته و با متمم ۱ عدد دوم جمع می کنیم. اگر حاصل جمع بیت اضافه (Carry) داشت، کری را حذف و باقی را با یک جمع می کنیم تا حاصل تفریق بدست آید. اگر حاصل جمع کری نداشت، دوباره از عدد حاصل جمع متمم ۱ می گیریم. جواب حاصل تفریق البته با علامت منفی می باشد.

مثال:

تفریق های زیر را با استفاده از متمم ۱ انجام دهید.

$$\text{الف) } X - Y = 1010100 - 1000011$$

$$\begin{array}{r} X \\ + Y \text{ عدد 1 متمم } \\ \hline 1010100 \quad 0010000 \\ + 0111100 \quad + 1 \\ \hline 10010000 \quad 0010001 \end{array}$$

حاصل جمع کری دارد. کری را حذف و باقی را با یک جمع می کنیم که برابر است با حاصل تفریق:

$$X - Y = 0010001$$

$$\text{ب) } Y - X = 1000011 - 1010100$$



$$\begin{array}{r} Y \\ +X \text{ عدد 1 متمم} \\ \hline 1000011 \\ +0101011 \\ \hline 1101110 \end{array}$$

حاصل جمع بیت اضافه ندارد. پس از حاصل دوباره متمم ۱ می‌گیریم که جواب تفریق است، البته با علامت منفی:

$$Y - X = -0010001$$

#### ۱.۵.۴ تبدیل تفریق به جمع با استفاده از متمم ۲

عدد اول را نوشته با متمم ۲ عدد دوم جمع می‌کنیم. اگر حاصل جمع کری داشت، کری را حذف می‌کنیم تا حاصل تفریق بدست آید. اگر حاصل جمع کری نداشت، دوباره از عدد حاصل جمع متمم ۲ می‌گیریم. جواب حاصل تفریق البته با علامت منفی می‌باشد.

مثال:

تفریق‌های زیر را با استفاده از متمم ۲ انجام دهید:

$$X - Y = 1010100 - 1000011 \text{ (الف)}$$

$$\begin{array}{r} X \\ +Y \text{ عدد 2 متمم} \\ \hline 1010100 \\ +0111101 \\ \hline 10010001 \end{array}$$

حاصل جمع بیت اضافه دارد. کری را حذف می‌کنیم و باقی برابر است با حاصل تفریق:

$$X - Y = 0010001$$

$$Y - X = 1000011 - 1010100 \text{ (ب)}$$

$$\begin{array}{r} Y \\ +X \text{ عدد 2 متمم} \\ \hline 1000011 \\ +0101100 \\ \hline 1101111 \end{array}$$

حاصل جمع بیت اضافه ندارد. پس از حاصل دوباره متمم ۲ می‌گیریم که جواب تفریق است؛ البته با علامت منفی:

$$Y - X = -0010001$$



منطق باینری با متحول‌هایی که دو ارزش گسسته و عملیاتی که مفهوم منطقی دارند، سر و کار دارد. دو مقداری که متحول‌ها اختیار می‌کنند، ممکن است با نام‌های مختلفی نام‌گذاری شوند (صفر و یک، True و False، High و Low). اما برای ما بهتر است آن را بر حسب بیت تصور کنیم و مقادیر ۱ و ۰ را به آن تخصیص دهیم. منطق باینری معرفی شده در این فصل معادل با جبری به نام جبر بول است.

در این فصل قاعده‌های مختلف اعداد (۲، ۸، ۱۰ و ۱۶) و نحوه تبدیل این قاعده‌ها به هم معرفی گردید.

در فصول بعد به استفاده از این قاعده‌ها در منطق دیجیتال سخت افزار کامپیوتر می‌پردازیم.



۱. اعداد قاعده ۸ و ۱۶ بین ۱۶ تا ۳۲ را لیست کنید. با استفاده از کاراکترهای A و B برای دو رقم آخر، اعداد ۱۰ تا ۲۶ را در قاعده ۲ لیست نمایید.

۲. اعداد زیر را از قاعده باینری به قاعده هشت، ده و شانزده تبدیل کنید:

$$(11100111011)_2 - a$$

$$(1111000010)_2 - b$$

$$(101011011001)_2 - c$$

$$(111111001101)_2 - d$$

۳. اعداد زیر را از قاعده ده به قاعده دو، هشت و شانزده تبدیل کنید:

$$(2466)_{10} - a$$

$$(223)_{10} - b$$

$$(715)_{10} - c$$

$$(123)_{10} - d$$

۴. اعداد زیر را از قاعده هشت به قاعده دو، دسیمل و شانزده تبدیل کنید:

$$(2466)_8 - a$$

$$(223)_8 - b$$

$$(715)_8 - c$$

$$(123)_8 - d$$

۵. اعداد زیر را از قاعده شانزده به قاعده دو، هشت و دسیمل تبدیل کنید.

$$(A5)_{16} - a$$

$$(31B)_{16} - b$$

$$(F16)_{16} - c$$

$$(914)_{16} - d$$

۶. متمم‌های ۲ و ۱۰ اعداد دسیمل زیر را بدست آورید.

$$13162346 - a$$

$$74516310 - b$$

$$11001000 - c$$

$$11111111 - d$$

۷. عدد دسیمل ۳۴۵ را به دو روش به باینری تبدیل نمایید:

$a$  - مستقیماً به باینری تبدیل نمایید.

$b$  - ابتدا به قاعده شانزده و سپس از قاعده شانزده به باینری تبدیل سازید و همچنان بگویید کدام روش سریع‌تر است؟

۸. متمم 16 عدد AF3B را بدست آورید.

## فصل دوم

### جبر بولی (Boolean Algebra)



هدف کلی: آشنایی محصلان با جبر بولی.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. جبر بولی را تعریف نمایند.
۲. خواص جبر بولی را شرح دهند.
۳. انواع جبر بولی و قضایای جبر بولی را توضیح دهند.
۴. اصول جبر بولی و اتحادهای اصلی جبر بولی را شناسایی نمایند.

همان‌طور که می‌دانیم در سیستم‌های کامپیوتری و سخت افزار کامپیوترها فقط اعداد باینری ۰ و ۱ مورد استفاده قرار می‌گیرد. بنابراین، چون یک سیستم کامپیوتری فقط اعداد ۰ و ۱ را می‌شناسد، باید تمام عملیات ریاضی، شامل جمع، تفریق، ضرب، تقسیم و... را روی این اعداد انجام دهد. پس ریاضیات مورد استفاده شده در کامپیوتر ریاضیاتی فقط شامل ۰ و ۱ خواهد بود که به ریاضیات بولی یا به اصطلاح جبر بولی<sup>۱</sup> معروف می‌باشد. در این فصل نگاهی جامع و مختصر به جبر بولی و قوانین و قواعد مربوط به جبر بولی و کاربردهای آن در کامپیوتر می‌اندازیم تا بتوانید، نحوه محاسبات کامپیوترها بر اساس این ریاضیات خاص تجزیه و تحلیل نموده و از آن در ساخت مدارهای منطقی دیجیتال استفاده نمایید.

در این فصل با تعریف جبر بولی و مفاهیم اولیه جبر بولی آشنا می‌شوید. جبر بولی ریاضیاتی است که روی اعداد ۰ و ۱ مورد استفاده قرار می‌گیرد. اکثر قوانین ریاضیات بولی با ریاضیات قاعده ۱۰ یکسان است، ولی تفاوت‌هایی نیز دارد. لذا نیازمند مطالعه این قوانین و مقررات در این فصل می‌باشد.

## ۲.۱ تعریف جبر بولی

جبر بول را می‌توان، مانند هر سیستم منته ریاضی، به وسیله مجموعه‌یی از عناصر، یک مجموعه از الگوها و تعدادی از اصول با بدیهیات تعریف نمود. یک مجموعه از عناصر کلکسیونی از اشیاء است که دارای خواص مشترکی باشند. اگر در یک مجموعه  $Y$  و  $X$  عناصر مشخصی از آن باشند، آنگاه  $X \in S$  به این معناست که  $X$  عضوی از مجموعه  $S$  و  $Y \notin S$  یعنی  $Y$  عضوی از  $S$  نیست. یک مجموعه با تعداد قابل شمارشی از عناصر با یک جفت آکولاد مشخص می‌شود:  $A = \{1,2,3,4\}$ ، یعنی عناصر مجموعه  $A$  عبارتند از: 1,2,3,4 یک عملگر باینری روی یک مجموعه از عناصر،  $S$ ، قانونی است که به هر جفت از عناصر  $S$ ، یک عنصر منحصر به فرد از  $S$  را تخصیص دهد.

به عنوان مثال، رابطه  $a + b = c$  را در نظر بگیرید. (\*) را یک عملگر باینری می‌خوانیم به شرطی که بتواند، عنصر  $C$  را به جفت عنصر  $b$  و  $a$  منتسب نماید. ضمن این که رابطه  $a, b, c \in S$  معتبر باشد.

با این وجود اگر  $a, b \in S$  و  $c \notin S$  باشد، (\*) یک عملگر باینری نیست.

اصول یک سیستم ریاضی، فرضیات اولیه را تشکیل می‌دهند که با استفاده از آن‌ها می‌توان قوانین، تئوری‌ها و خواص سیستم را نتیجه گرفت. مهم‌ترین اصول بکار رفته در فرموله کردن ساختارهای جبری عبارتند از:

۱. **بسته بودن (Closure):** یک مجموعه  $S$  نسبت به عملگر باینری بسته است به شرطی که برای هر جفت عنصر از  $S$ ، این عملگر عنصر منحصر به فردی از آن را به جفت عنصر منتسب نماید. به عنوان مثال، مجموعه اعداد طبیعی  $N = \{1,2,3,4, \dots\}$  را نسبت به عملگر جمع (+) بسته گوئیم

<sup>۱</sup> Boolean Algebra

زیرا برای هر دو عنصر  $a \in N$  و  $b$  عنصر دیگری مانند  $c \in N$  می‌توان یافت، به طوری که  $a + b = c$  باشد. مجموعه اعداد طبیعی نسبت به عملگر تفریق بسته نیست، چون داریم

$$2 - 3 = -1 \quad 2, 3 \in N, -1 \notin N$$

۲. اصل اتحادی (Associative Law): یک عملگر باینری (\*) روی مجموعه  $S$  شرکت پذیر است

اگر داشته باشیم به ازای همه مقادیر  $x, y, z \in S$

$$(x * y) * z = x * (y * z)$$

۳. اصل تبدیلی (Commutative Law): یک عملگر (\*) روی یک مجموعه دارای خاصیت تبدیلی است. هرگاه به ازای هر  $x, y \in S$

$$x * y = y * x$$

۴. عنصر عینیت (Identity Element): مجموعه و نسبت به عملگر (\*) روی مجموعه  $S$  دارای

عنصر عینیت است، اگر عنصر  $e \in S$  با خاصیت زیر موجود باشد.

$$e * x = x * e = x \quad x \in S$$

مثال: عنصر 0 یک عنصر شناسه نسبت به عملگر (+) روی مجموعه اعداد صحیح است، چون

$$I = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$$

به ازای هر  $x \in I$   $x + 0 = 0 + x = x$

مجموعه اعداد طبیعی  $N$  دارای عنصر شناسه نیست زیرا  $0$  جزو مجموعه نمی‌باشد.

۵. معکوس (Inverse): مجموعه‌ای چون  $S$  با عنصر شناسه و نسبت به عملگر (\*) دارای معکوس

است به شرطی که برای هر  $x \in S$ ، یک  $y \in S$  وجود داشته باشد به نحوی که  $x * y = e$

مثال: در مجموعه اعداد صحیح،  $I$  با  $e = 0$  معکوس عنصر  $a$  برابر  $(-a)$  است.

$$a + (-a) = 0$$

۶. اصل توزیع پذیری (distributive law): اگر (\*) و (.) دو عملگر روی مجموعه  $S$  باشند، (\*)

$$x * (y \cdot z) = (x * y) \cdot (x * z)$$

مثالی جبری در این مورد میدان یا حوزه است. میدان مجموعه‌یی از عناصر است، همواره با دو عملگر باینری که هر یک دارای خواص 1 تا 5 بوده و هر دو عملگر برای تشکیل خاصیت ۶ با یکدیگر ترکیب می‌شوند. مجموعه اعداد حقیقی، همراه با عملگرهای باینری (+) و (.)، میدان اعداد حقیقی را تشکیل می‌دهند. میدان اعداد حقیقی قاعده جبر معمولی و حساب است. عملگرها و اصول دارای مفاهیم زیر هستند:

عملگر باینری (+) جمع را تعریف می کند.

شناسه جمع 0 است.

معکوس جمع، تفریق می باشد.

عملگر باینری (•) ضرب را تعریف می نماید.

شناسه ضرب ۱ می باشد.

معکوس ضرب  $a = 1/a$  تقسیم را تعریف می کند، یعنی  $(1/a) = 1/a$ . تنها اصل توزیع پذیری قابل اعمال مربوط به عملگر (1) روی (+) است:

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

### ۲.۱.۱ تعریف اصول اساسی جبر بولی

در سال 1854 جورج بول یک برخورد سیستماتیک با منطق را معرفی نمود و برای اهداف خود یک سیستم جبری را که امروزه آن را جبر بول می نامیم، پایه ریزی کرد. در سال ۱۹۳۸ نیز سی.ای. شانون<sup>۱</sup> یک جبر بول دو مقداری به نام جبر سوئیچینگ را معرفی کرد که در آن خواص مدارهای سوئیچینگ با این جبر قابل ارائه است. برای تعریف مستدل جبر بول، ما اصول فرموله شده به وسیله ای.ی. هانتینگتون در سال 1904 را بکار می بریم.

جبر بول یک ساختار جبری است که با عناصر مجموعه، یعنی B، همراه با دو عملگر باینری (+) و (•) تعریف می شود، به شرطی که اصول زیر (هانتینگتون) در آن معتبر باشد.

۱. مجموعه نسبت به عملگر (+) بسته باشد.

مجموعه نسبت به عملگر (۱) بسته باشد.

۲. یک عنصر شناسه ۰ برای (+) وجود داشته باشد

یک عنصر شناسه ۱ برای (•) وجود داشته باشد.

۳. مجموعه نسبت به (+) دارای خاصیت جابجایی باشد:  $x + y = y + x$  و مجموعه نسبت

به (•) دارای خاصیت جابجایی باشد:  $x.y = y.x$

۴.

a. (•) نسبت به (+) توزیع پذیر است:

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

---

<sup>۱</sup> Glaud Shanon



b. (+) نسبت به (.) توزیع پذیر است:

$$x + (y \cdot z) = (x + y) \cdot (x + z)$$

۵. برای هر عنصر  $x \in B$ ، عنصری مثل  $x' \in B$  وجود دارد (به آن متمم  $X$  می‌گوییم) به نحوی که:

$$x + x' = 1 \text{ و } x \cdot x' = 0 \quad (a)$$

۶. حداقل دو عنصر  $x, y \in B$  وجود دارد به نحوی که  $x \neq y$  باشد.

با مقایسه جبر بول با حساب و جبر معمولی (حوزه یا میدان اعداد حقیقی) تفاوت‌های زیر قابل ملاحظه‌اند:

۱. اصول هانتینگتون<sup>۱</sup> فاقد اصل شرکت پذیری است. با این وجود، این اصول برای جبر بول معتبر

و

برای هر دو عملگر از دیگر اصول قابل استنتاج است.

۲. اصل توزیع پذیری (+) روی (.)، یعنی:

$$x + (y \cdot z) = (x + y) \cdot (x + z)$$

برای جبر بول معتبر است، ولی در جبر معمولی قابل قبول نیست.

۳. جبر بول دارای معکوس‌های جمع و ضرب نیست؛ بنابراین عملگرهای تفریق و تقسیم وجود

ندارند.

۴. اصل 5 عملگری به نام متمم را معرفی می‌نماید که در جبر معمولی وجود ندارد.

۵. جبر معمولی در مورد اعداد حقیقی بحث می‌کند که یک مجموعه با بی‌نهایت عنصر را شامل

می‌شود. جبر بول در مورد مجموعه‌یی از عناصر،  $B$ ، بحث می‌نماید که هنوز آن را معرفی

نکرده‌ایم، ولی بعداً در جبر بول دو مقداری با دو ارزش معرفی خواهد شد (کاربرد بعدی ما از

این جبر مورد توجه است) و در آن  $B$  به صورت مجموعه‌یی از دو عنصر 0 و 1 تعریف می‌شود.

## ۲.۲ عملیات بولی

### ۲.۲.۱ متحول (variable)

سمبولی است که مقدار منطقی را نشان می‌دهد.

c, b, a

<sup>۱</sup> Huntington

### ۲.۲.۲ متمم (complement)

معکوس (variable) متحول را متمم می‌نامند.

$$a', b', c'$$

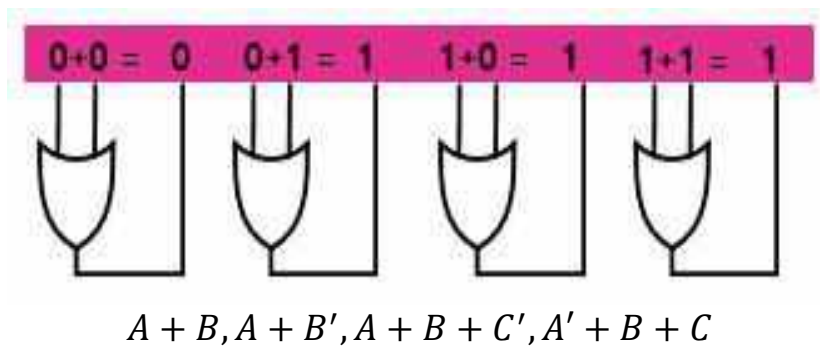
### ۲.۲.۳ لیترال (Litral)

هر متحول و یا متمم را لیترال نیز می‌گویند.

$$a, a', b, b', c, c'$$

### ۲.۲.۴ جمع بولی چند لیترال

جمع بولی معادل عملیات (OR) است.



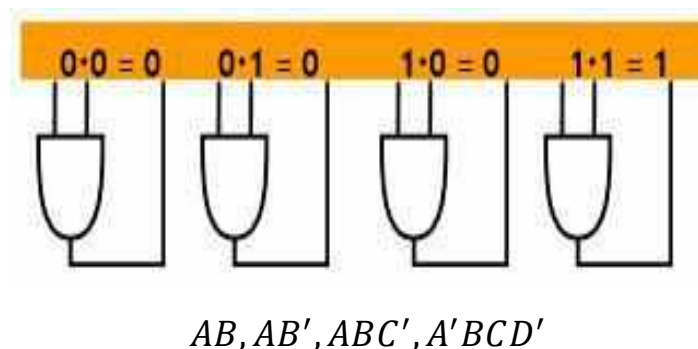
شکل (۲-۱): عملیات جمع بولی

زمانی 1 می‌شود که تنها یک لیترال و یا چند لیترال 1 شند.

و زمانی 0 می‌شود که همه‌ی لیترال‌ها 0 باشند.

### ۲.۲.۵ ضرب بولی

ضرب بولی معادل عملیات AND است.



شکل (۲-۲): عملیات ضرب بولی

زمانی ۱ می‌شود که همه‌ی لیترال‌ها ۱ باشد.

زمانی ۰ می‌شود که تنها یک لیترال و یا چند لیترال ۰ باشد.

## ۲.۳ قوانین و مقررات جبر بولی

### ۲.۳.۱ قوانین تبدیلی (Commutative Laws)

قانون تبدیلی جمع برای دو متحول: در عملیات جمع یا OR با دو و یا چند متحول می‌توان جای متحول‌ها را با هم تبدیل کرد، چرا که تبدیلی در جمع اثری روی نتیجه‌ی جمع نخواهد داشت.

$$A + B = B + A$$



شکل (۲-۳): قانون تبدیلی در جمع بولی

### ۲.۳.۱.۱ قانون تبدیلی ضرب برای دو متحول

در عملیات ضرب یا AND با دو و یا چند متحول می‌توان جای متحول‌ها را با هم تبدیل کرد، چرا که قانون تبدیلی در ضرب اثری روی نتیجه‌ی ضرب نخواهد داشت.

$$AB = BA$$

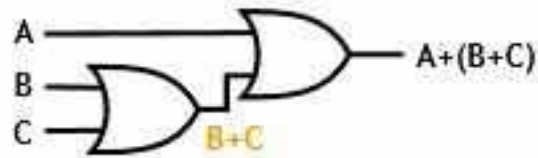


شکل (۲-۴): قانون تبدیلی

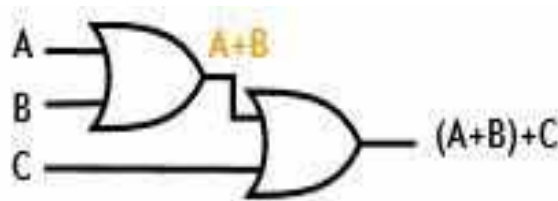
### ۲.۳.۲ قوانین انجمنی (Associative Laws)

قانون انجمنی جمع برای سه متحول: در جمع یا OR با سه متحول و یا چند متحول جای قوس را می‌توان تبدیل کرد، چرا که جای قوس روی حاصل جمع تأثیر نمی‌گذارد.

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



≡

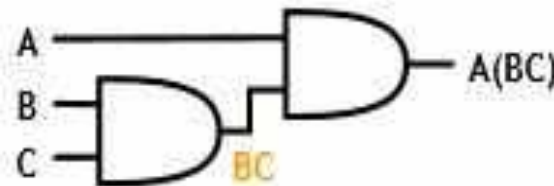


شکل (۲-۱): قانون انجمنی جمع بولی

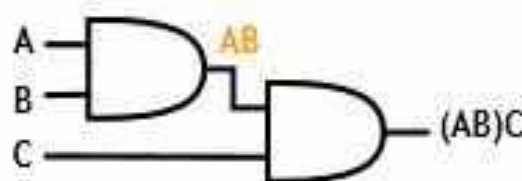
### ۲.۳.۲.۱ قانون انجمنی ضرب برای سه متحول:

در ضرب یا AND با سه متحول یا چند متحول جای قوس را می‌توان تبدیل کرد، چرا که جای قوس روی حاصل ضرب تأثیر نمی‌گذارد.

$$A(BC) = (AB)C$$



≡



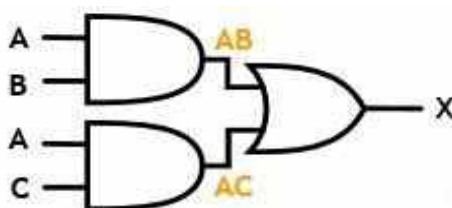
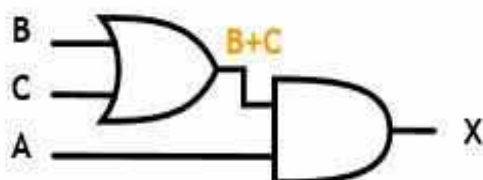
شکل (۲-۲): قانون انجمنی ضرب بولی

## ۲.۳.۳ قوانین توزیعی (Distributive Laws)

### ۲.۳.۳.۱ قانون توزیعی برای ۳ متحول

به قانون توزیعی می‌گویند که متحول خارج قوس می‌تواند در تک تک متحولان داخل قوس ضرب گردد. نتیجه هر دو حالت با هم برابر است.

$$A(B + C) = AB + AC$$



$$X = AB + AC$$

شکل (۳-۲): قانون توزیعی

جدول (۱-۲): قوانین جبر بولی

قانون	
1	$A+0=A$
2	$A+1=1$
3	$A.0=0$
4	$A.1=A$
5	$A+A'=1$
6	$A.A'=0$
7	$A+A=A$
8	$A.A=A$
9	$A''=A$
10	$A+AB=A$
11	$A+A'B=A+B$
12	$(A+B)(A+C)=A+BC$

### ۲.۳.۴ قضیه دمورگان

NOR دو متحول برابر است با AND معکوس‌های دو متحول.

$$\overline{X + Y} = \bar{X} \bullet \bar{Y}$$

NOR      Negative-AND

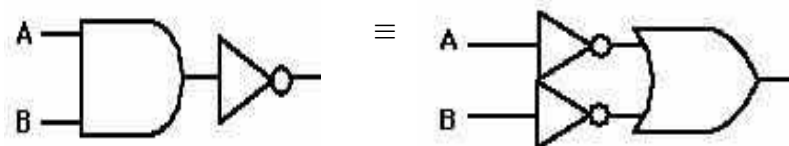


شکل (۴-۲): قانون دمورگان

NAND دو متحول برابر است با OR معکوس‌های دو متحول:

$$\overline{X \bullet Y} = \bar{X} + \bar{Y}$$

NAND      Negative-OR



شکل (۵-۲): قانون دمورگان

### ۲.۳.۵ اولویت عملگرها

اولویت عملگرها در معادلات جبر بولی به صورت زیر می باشد:

بالاترین اولویت را در معادلات جبر بولی قوس دارد، یعنی در معادلات اولین عملیه‌یی که باید انجام شود، علیه داخل قوس می باشد. بعد از قوس، اولویت بعدی را معکوس یا NOT دارد. هم چنین بعد از معکوس، اولویت بعدی را ضرب یا AND دارد. و در آخر بعد از ضرب، اولویت بعدی را جمع یا OR دارد.

$$() > ' > * > +$$

### ۲.۴ تئوری‌های ساده سازی

تئوری‌های زیر به منظور ساده سازی معادلات جبر بولی و مدارهای منطقی استفاده می گردد.

#### ۲.۴.۱ تئوری اتحادی<sup>۱</sup>

$$XY + XY' = X$$

$$X(Y + Y') = X.1 = X \rightarrow \text{ثبوت}$$

$$(X + Y)(X + Y') = X$$

$$XX + XY' + YX + YY' = X + X(Y + Y') + 0 = X \rightarrow \text{ثبوت}$$

#### ۲.۴.۲ تئوری جذب<sup>۲</sup>

$$X + XY = X$$

$$X(1 + Y) = X.1 = X \rightarrow \text{ثبوت}$$

$$X(X + Y) = X$$

$$XX + XY = X + XY = X \rightarrow \text{ثبوت}$$

#### ۲.۴.۳ تئوری جذب سطحی<sup>۳</sup>

$$(X + Y')Y = XY$$

$$XY + YY' = XY + 0 = XY \rightarrow \text{ثبوت}$$

---

<sup>۱</sup> Uniting

<sup>۲</sup> Absorption

<sup>۳</sup> Adsorption



جبر بولی ریاضیاتی است که روی اعداد ۰ و ۱ مورد استفاده قرار می‌گیرد. برعلاوه آن، اکثر قوانین ریاضیات بولی با ریاضیات قاعده ۱۰ یک‌سان است، ولی تفاوت‌هایی نیز دارد. قوانین جبر بولی شامل عملیات مختلفی چون ضرب و جمع و تفریق و... می‌باشد. قوانینی برای ساده سازی معادلات جبر بولی و همچنین تبدیل معادلات جبر بولی از یک شکل به شکل‌های دیگر مورد مطالعه قرار گرفت.





۱. عبارات بولی زیر را با تعداد حداقل لیترال ساده کنید؟

- (1)  $xy + xy'$
- (2)  $(x + y)(x + y')$
- (3)  $xyz + x'y + xyz'$
- (4)  $xyz + x'yz + xyz + x'yz'$
- (5)  $(A + B)'(A' + B')'$
- (6)  $(x + y + z')(x' + y' + z)$

۲. عبارات بولی زیر را با توجه به قوانین خوانده شده، ساده کنید؟

- (1)  $A'C' + ABC + AC'$
- (2)  $(x'y' + z)' + z + xy + wz$
- (3)  $A'B(D' + C'D) + B(A + A'CD)$
- (4)  $(A' + C)(A' + C')(A + B + C'D)$
- (5)  $ABCD + A'BD + ABC'D$

۳. کدام یکی از موارد زیر صحیح است؟

- (1)  $* > () > ' > +$
- (2)  $+ > ' > * > ()$
- (3)  $() > ' > * > +$
- (4)  $+ > * > ' > ()$

۴. کدام یکی از موارد صحیح نمی باشد؟

- (1)  $A + A'B = A + B$
- (2)  $A + A = A$
- (3)  $A + 1 = A$
- (4)  $A.A'' = 0$

۵. کدام یکی از موارد درست می باشد؟

- (1)  $A' + A' = 1$
- (2)  $A'' + AB = A$
- (3)  $A + A = 0$
- (4)  $A'' =$

## فصل سوم

### گیت‌های منطقی

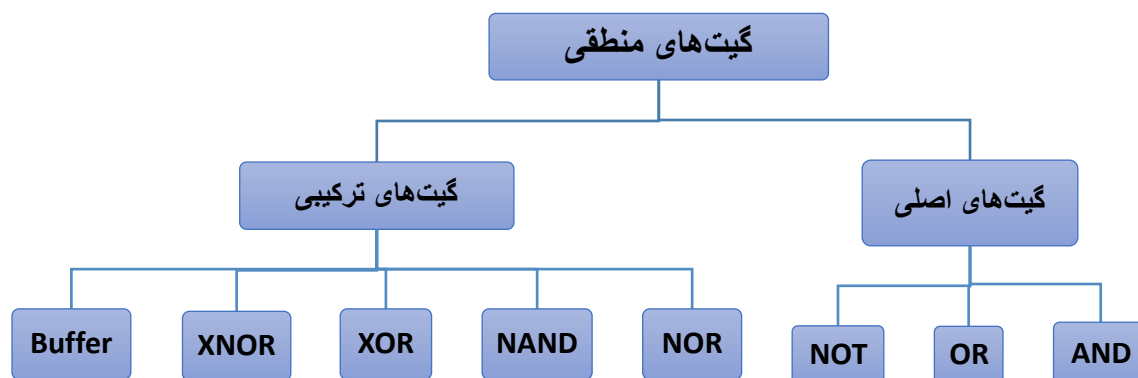


هدف کلی: آشنایی محصلان با گیت‌های منطقی.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. گیت‌های منطقی را تعریف کرده بتواند.
۲. گیت‌های منطقی را تجزیه و تحلیل بتواند.
۳. گیت‌های ترکیبی را شرح دهند.

گیت‌های 1 منطقی عناصر اساسی ساخت مدارهای دیجیتال هستند. در واقع یک گیت کوچک‌ترین واحد سازنده مدارهای منطقی می‌باشد. گیت‌های منطقی انواع مختلفی دارند. هر گیت خصوصیات و ویژگی‌ها و کاربردهای خاص خود را دارا می‌باشد. ویژگی‌های منحصربه‌فرد هرگیت آن را از دیگر گیت‌های منطقی متمایز می‌سازد. هر گیت بر اساس ورودی‌هایی که می‌گیرد، خروجی‌هایی بر اساس آن تولید می‌نماید. گیت‌های دیجیتال می‌توانند، یک یا چندین ورودی داشته باشند، اما تمامی گیت‌های منطقی دیجیتال تنها یک خروجی دارند. گیت‌های اصلی یا پایه، شامل 3 گیت اصلی، گیت NOT، گیت OR و گیت AND می‌باشند. گیت‌های دیگری هم است که از ترکیب این گیت‌های اصلی می‌توانند ساخته شوند که به آن‌ها گیت‌های فرعی گفته می‌شود. گیت‌های فرعی شامل، گیت NAND، گیت NOR، گیت XOR و گیت XNOR می‌باشند. در مجموع این هفت گیت، گیت‌هایی هستند که در ساخت مدارهای منطقی مورد استفاده قرار می‌گیرند. یک مدار منطقی دیجیتال شامل ترکیبی از این گیت‌های منطقی است که یک کار منطقی را انجام می‌دهد و بر اساس یک تعداد ورودی، خروجی مورد نظر را تولید می‌کنند. در این فصل با گیت‌های منطقی و نحوه ساخت مدارهای منطقی و تبدیل مدارهای منطقی به معادلات منطقی و بر عکس آشنا خواهید شد.



### ۳.۱ گیت‌های منطقی دیجیتال

چون توابع بول برحسب عملگرهای AND، OR و NOT بیان شده‌اند، پیاده کردن آن‌ها با استفاده از این گونه گیت‌ها ساده‌تر خواهد بود. امکان ساخت گیت‌ها برای دیگر اعمال منطقی در عمل مورد توجه است. فکتورهایی که باید به هنگام ساخت آن‌ها در نظر گرفته شوند عبارتند از:

۱. امکان سنجی و اقتصادی بودن روش ساخت به هنگام استفاده از قطعات فیزیکی؛

<sup>۱</sup> Gate

۲. امکان گسترش ورودی گیت‌ها به بیش از دو؛

۳. در نظر گرفتن خواص اصلی عملگرهای باینری مثل جابجایی و شرکت‌پذیری؛

۴. توانایی گیت در پیاده‌سازی توابع به تنهایی یا همراه با سایر گیت‌ها از شانزده تابع معرفی شده.

در جدول دو تابع برابر با مقدار ثابت و چهارتای دیگر دوبار تکرار شده‌اند. بنابراین، تنها ده تابع برای تهیه گیت‌های منطقی کاندید هستند. دو تابع نهی و استلزام دارای خاصیت جابجایی یا شرکت‌پذیری نیستند - این دو تابع (نهی و استلزام) به وسیله طراحان مدارات منطقی به کار می‌روند، ولی به ندرت در منطق کمپیوتر از آنها استفاده می‌شود. لذا به عنوان گیت‌های منطقی ستندرد مورد استفاده نمی‌باشند. هشت تابع دیگر یعنی: XOR، NOR، NAND، OR، AND، NOT، Buffer و XNOR به عنوان گیت‌های ستندرد در طراحی سیستم‌های دیجیتال به کار می‌روند.

مدار NOT یا معکوس‌گر وضعیت منطقی یک متحول باینری را معکوس می‌نماید.

سمبول گرافیکی یک معکوس‌گر (که به آن حباب می‌گویند) بیانگر متمم شدن است. سمبول مثبت به تنهایی علامت بافر<sup>۱</sup> می‌باشد. یک بافر عمل انتقال را انجام می‌دهد. این مدار صرفاً در تقویت توان سیگنال‌ها استفاده شده و معادل با دو مدار متوالی معکوس‌گر است.

تابع NAND متمم AND است و همان‌طور که از سمبول گرافیکی آن مشخص است، از یک سمبول AND و یک حباب تشکیل شده است. تابع NOR هم متمم OR است و با یک سمبول OR و به دنبال آن یک حباب نمایش داده می‌شود. گیت‌های NAND و NOR به طور گسترده به عنوان گیت‌های ستندرد مورد استفاده قرار گرفته و بیشتر از OR و AND مورد توجه‌اند. این بدان علت است که گیت‌های NAND و NOR به‌سادگی به‌وسیله مدارهای ترانزیستوری قابل تولید بوده و می‌توان به راحتی توابع بول را با آنها پیاده‌سازی کرد. گیت XOR دارای سمبول مشابهی با OR است، بجز این‌که یک خط منحنی در سمت ورودی‌اش کشیده شده است. گیت XNOR متمم XOR است و لذا حباب کوچکی در خروجی آن وجود دارد.

هر گیت منطقی دیجیتال دارای سه مشخصه سمبول گرافیکی<sup>۲</sup>، جدول درستی<sup>۳</sup> و معادله<sup>۴</sup> می‌باشد که آن گیت را از سایر گیت‌های منطقی متمایز می‌سازد.

---

<sup>۱</sup> Buffer

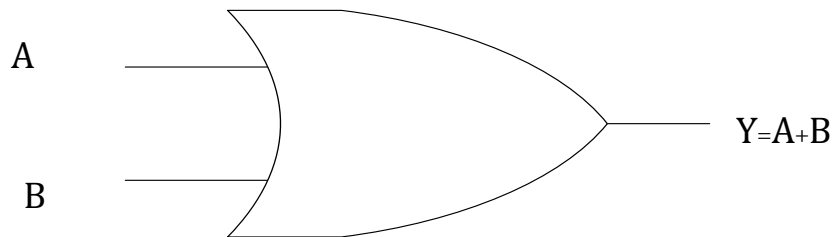
<sup>۲</sup> symbol

<sup>۳</sup> Truth Table

<sup>۴</sup> Boolean Expression

### ۳.۱.۱ گیت OR

حداقل دو ورودی و تنها یک خروجی دارد و ورودی‌ها را با هم جمع منطقی می‌کند. در این گیت فقط زمانی نتیجه صفر است که تمام ورودی‌ها صفر باشد.



شکل (۳-۱): سمبول گرافیکی گیت OR

جدول (۳-۱): جدول درستی گیت OR

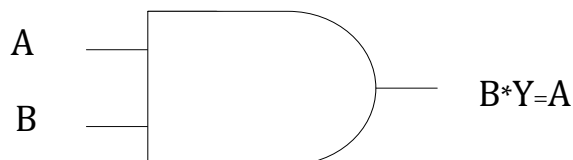
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

---- معادله ----

$$Y=A+B$$

### ۳.۱.۲ گیت AND

حداقل دو ورودی و تنها یک خروجی دارد. این گیت معمولاً عملیه ضرب بولی را بین متحولین انجام می‌دهد. در این گیت تنها زمانی نتیجه (۱) است که تمام ورودی‌ها دارای قیمت یک باشد در غیر آن نتیجه صفر است.



شکل (۳-۲) سمبول گرافیکی گیت AND

جدول (۳-۲): جدول درستی گیت AND

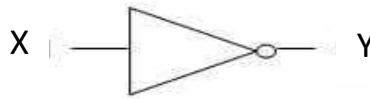
A	B	A*B
0	0	0
0	1	0
1	0	0
1	1	1

---- معادله ----

$$Y=A*B$$

### ۳.۱.۳ گیت NOT

گیت NOT یا معکوس گر، وضعیت منطقی یک متحول باینری را معکوس می نماید. گیت NOT به نام گیت معکوس گر نیز یاد می شود که تنها یک ورودی و یک خروجی دارد. سمبول گرافیکی یک معکوس گر (که به آن حباب می گویند)، بیانگر متمم شدن است.



شکل (۳-۳): سمبول گرافیکی گیت NOT

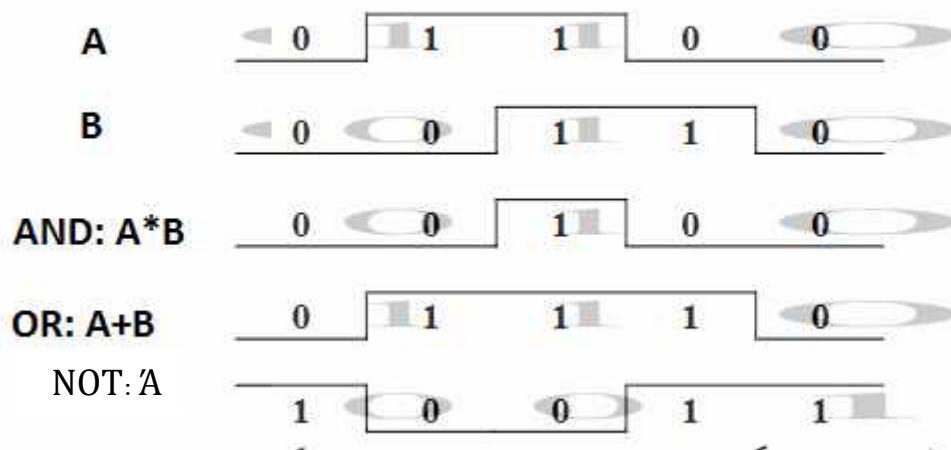
جدول (۳-۳): جدول درستی گیت NOT

A	Y
0	1
1	0

---- معادله ----

$$Y=A$$

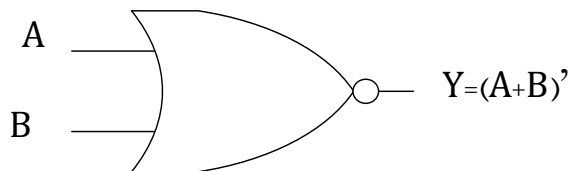
مدار زمانی گیت های اصلی و ستندرد (Timing diagram): مدار زمان یا Timing Diagram دیاگرامی است که سیگنال های ورودی را دریافت کرده و با توجه به عملیات مورد نظر ما مقدار سیگنال خروجی را تعیین می کند.



سیگنال‌های ورودی و خروجی برای گیت‌های AND, OR, NOT

### ۳.۱.۴ گیت NOR

تابع NOR متمم OR است و با یک سمبول OR و به دنبال آن یک حباب نمایش دیتا می‌شود.



شکل (۴-۶): سمبول گرافیکی گیت NOR

جدول (۴-۳): جدول درستی گیت NOR

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

---- معادله ----

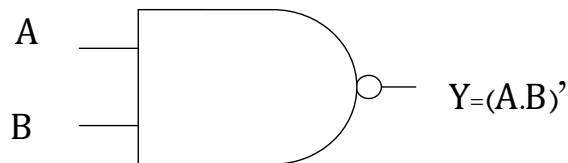
$$Y=(A+B)'$$

---- نماد گیت ----



### ۳.۱.۵ گیت NAND

تابع NAND متمم AND است و همان‌طور که از سمبول گرافیکی آن مشخص است از یک سمبول AND و یک حباب تشکیل شده است.



شکل (۳-۵): سمبول گرافیکی گیت NAND

جدول (۳-۵): جدول درستی گیت NAND

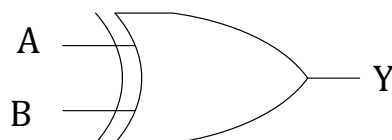
A	B	Y
۰	۰	۱
۰	۱	۱
۱	۰	۱
۱	۱	۰

----- معادله -----

$$Y = (A.B)'$$

### ۳.۱.۶ گیت XOR

گیت XOR دارای سمبول مشابهی با OR است، بجز این که یک خط منحنی در سمت ورودی‌اش کشیده شده است که حداقل دو ورودی و تنها یک خروجی دارد. زمانی که ورودی‌ها برابر باشند و یا یک‌سان باشند صفر می‌شود.



شکل (۳-۶): سمبول گرافیکی گیت XOR



جدول (۳-۶): جدول درستی گیت XOR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

---- معادله ----

$$Y = A'B + AB'$$

---- نماد گیت ----

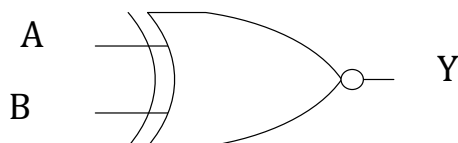


### ۳.۱.۷ گیت XNOR

گیت XNOR متمم XOR است و لذا حباب کوچکی در خروجی آن وجود دارد.

حداقل دو ورودی و تنها یک خروجی دارد.

زمان که ورودی‌ها برابر باشد و یا یکسان باشد، خروجی یک می‌شود.



شکل (۳-۷): سمبول گرافیکی گیت XNOR

جدول (۷-۳): جدول درستی گیت XNOR

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

---- معادله ----

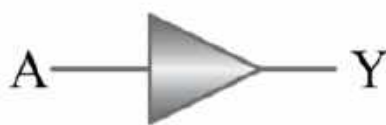
$$Y = A'B' + AB$$

---- نماد گیت ----



**گیت بافر (Buffer):** این گیت مانند گیت NOT یک ورودی و یک خروجی دارد. ولی بر خلاف گیت NOT مقدار سیگنال ورودی را معکوس نمی کند یعنی اگر ورودی یک باشد دوباره آن را به خروجی یک تبدیل می کند.

شکل زیر سمبول منطقی گیت بافر را نشان می دهد.



**گیت بافر**

شکل (۸-۳): سمبول گرافیکی گیت Buffer

جدول (۸-۳): جدول درستی گیت Buffer

Y	A
0	0
1	1

معادله منطقی  $Y = A$

## کاربرد

۱. تقویت سیگنال و گرفتن انشعاب

۲. ایجاد تأخیر در سیگنال ورودی

گیت بافر معمولاً به عنوان جداکننده بین دو طبقه استفاده می‌شود و از بارگذاری روی خروجی جلوگیری می‌نماید. در داخل گیت بافر مدار تقویت کننده جریان وجود دارد که میزان جریان دهی خروجی را افزایش می‌دهد.

## ۳.۲ گسترش ورودی گیت‌ها

گیت‌هایی که در قبل نشان داده شدند، بجز برای گیت NOT، قابل گسترش به بیش از دو ورودی می‌باشند. اگر عمل باینری یک گیت جابجا و شرکت پذیر باشد. می‌توان ورودی‌های آن را گسترش داد. اعمال AND و OR که در جبر بول تعریف شده‌اند، این خاصیت را از خود به نمایش گذاشته‌اند. برای تابع OR داریم:

$$x + y = y + x \text{ (جابجایی)}$$

$$(x + y) + z = x + (y + z) = x + y + z \text{ (شرکت پذیری)}$$

این روابط بیانگر تعویض پذیری ورودی‌های گیت و قابل گسترش بودن متحول‌های ورودی به بیش از دو در تابع OR است.

توابع NAND و NOR جابجا پذیرند و ورودی آن‌ها می‌تواند به بیش از دو افزایش یابد، مشروط بر این که در تعریف تابع مختصر تغییری صورت گیرد. مشکل این است که NAND و NOR شرکت پذیر نیستند یعنی این  $[(x \downarrow y) \downarrow z \neq x(y \downarrow z)]$  نکته در شکل و معادلات زیر به نمایش گذاشته شده‌اند.

$$(x \downarrow y) \downarrow z = [(x + y)'z]' = (x + y)z' = xz' + yz'$$

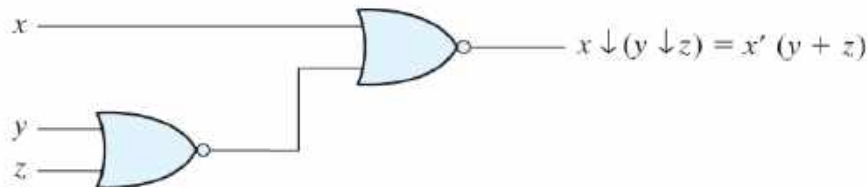
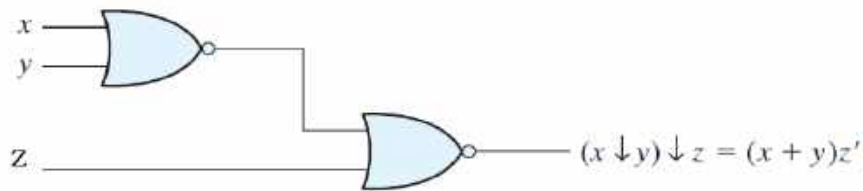
$$x \downarrow (y \downarrow z) = [x + (y + z)']' = x'(y + z) = x'y + x'z$$

NAND و یا NOR برای غلبه بر این مشکل آن AND و یا OR چند ورودی را به عنوان متمم

تعریف می‌کنیم؛ بنابراین:

$$x \downarrow y \downarrow z = (x + y + z)'$$

$$x \uparrow y \uparrow z = (xyz)'$$



شکل (۳-۰): جابجایی ورودی‌های گیت‌ها

باید قوس‌ها به شکل صحیح ( $NOR$ ) و ( $NAND$ ) انتخاب شوند تا بیانگر ترتیب صحیح گیت‌ها باشند. برای درک این مطلب، مدار شکل زیر را ملاحظه نمایید. برای این مدار تابع بول باید به شکل زیر نوشته شود.

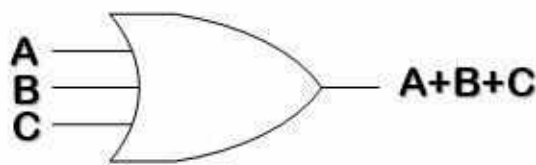


شکل (۳-۰): گسترش ورودی گیت‌ها

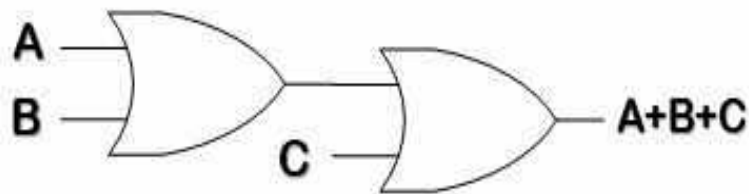
### ۳.۳ گیت‌های با چند ورودی

گیت‌های منطقی به غیر از گیت NOT که تنها یک ورودی دارد، می‌توانند حداقل دو ورودی داشته باشند. یعنی هرگیت می‌تواند، بیشتر از دو ورودی مثلاً ۳، ۴، ۱۰، ۱۰۰ و... هر تعداد بیشتری از ورودی‌ها را داشته باشد. در این صورت تغییر خاصی در محاسبات گیت صورت نمی‌گیرد. گیت ورودی‌ها را دو به دو با هم روی‌شان عملیات منطقی را انجام می‌دهد تا نتیجه و خروجی نهایی گیت به دست آید.

به عنوان مثال، در شکل زیر گیت OR دارای ۳ ورودی A, B, C می‌باشد. در این صورت حاصل خروجی به این صورت محاسبه می‌شود که ابتداء A و B با هم OR شده و سپس نتیجه این OR با ورودی C وارد OR بعدی شده تا خروجی نهایی بدست آید. شکل (۲-۱۰) و شکل (۲-۱۱) مشابه هم هستند، زیرا ورودی‌های یکسان و خروجی برابر با هم دارند.

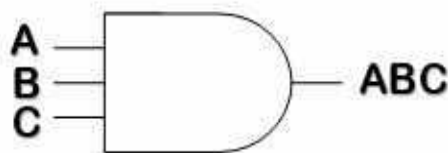


شکل (۱۱-۳): گیت OR با چند ورودی

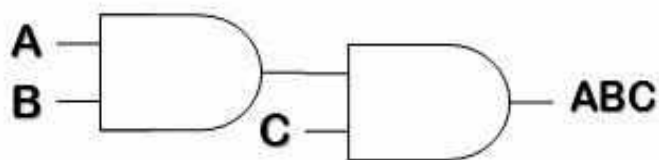


شکل (۱۲-۳): گیت OR با چند ورودی

به عنوان مثال، در شکل زیر گیت AND دارای ۳ ورودی  $A, B, C$  می‌باشد. در این صورت حاصل خروجی به این صورت محاسبه می‌شود که ابتدا  $A$  و  $B$  با هم AND شده و سپس نتیجه این AND با ورودی  $C$  وارد AND بعدی شده تا خروجی نهایی بدست آید. شکل (۱۲-۲) و شکل (۱۳-۲) مشابه هم هستند، زیرا ورودی‌های یک‌سان و خروجی برابر با هم دارند.



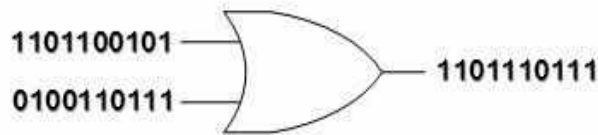
شکل (۱۳-۳): گیت AND با چند ورودی



شکل (۱۴-۳): گیت AND با چند ورودی

#### ۳.۴ گیت‌هایی با چند بیت ورودی

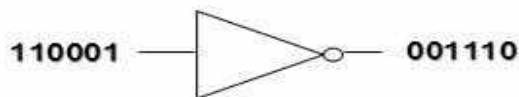
گیت‌های منطقی می‌توانند، چندین بیت ورودی داشته باشند. اگر گیتی چندین بیت ورودی داشته باشد، از هر ورودی یک بیت وارد شده و عملیات منطقی مستقل از سایر بیت‌های ورودی انجام می‌شود. به همین ترتیب بیت‌های بعدی یکی یکی وارد شده و خروجی تنها بر اساس همان ورودی‌ها در همان لحظه محاسبه می‌شود.



شکل (۱۵-۳): گیت OR با چند بیت ورودی



شکل (۱۶-۳): گیت AND با چند بیت ورودی

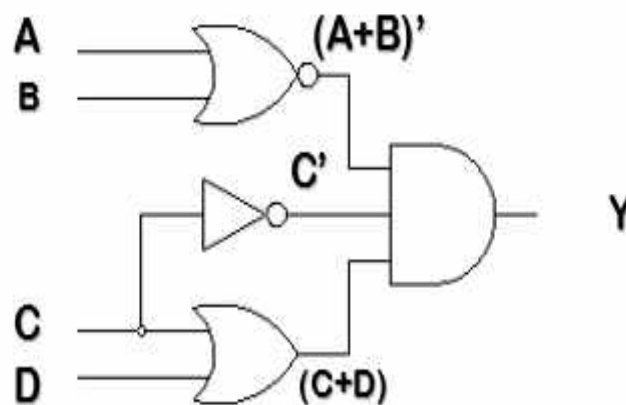


شکل (۱۷-۳): گیت NOT با چند بیت ورودی

### ۳.۵ تبدیل مدار به معادله

برای نوشتن معادله از روی مدار، از سمت چپ مدار؛ یعنی سمت ورودی‌ها شروع می‌کنیم، خروجی هر گیت را به ترتیب از سمت چپ نوشته، سطح به سطح پیش رفته تا به خروجی نهایی، یعنی  $Y$  برسیم که مقدار  $Y$  برابر با مقدار معادله مربوط به مدار می‌باشد.

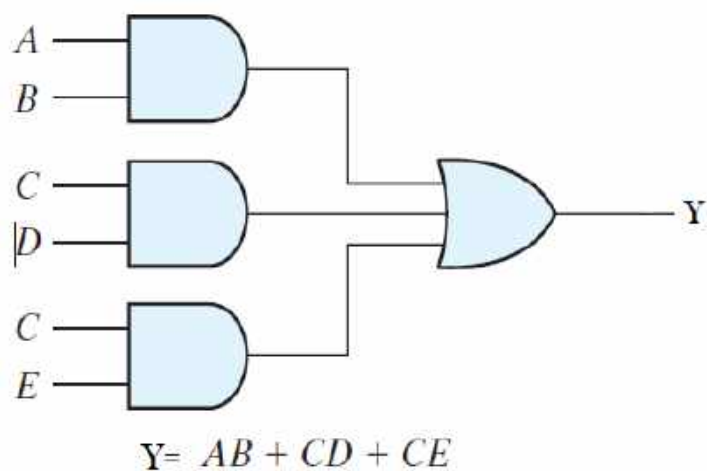
مثال اول:



شکل (۱۸-۳): تبدیل مدار به معادله

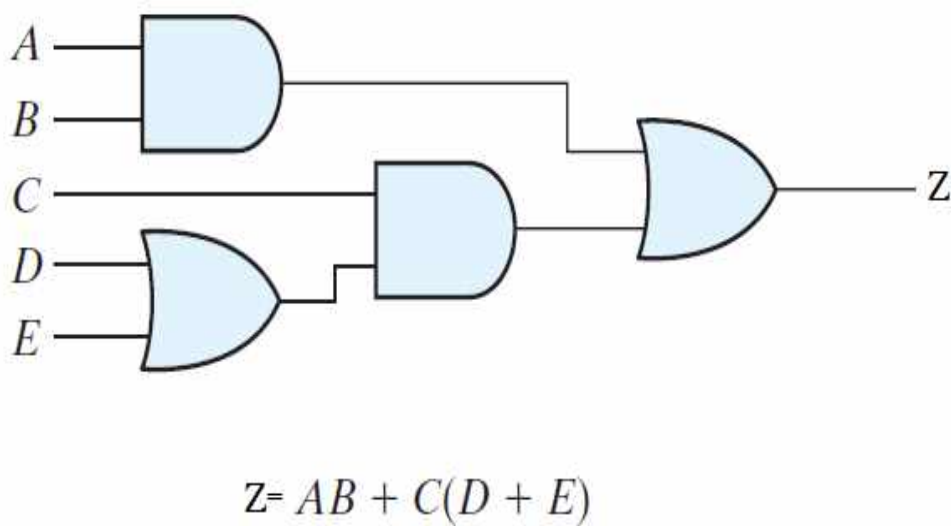
$$Y = (A+B)' * (C') * (C+D)$$

مثال دوم:



شکل (۳-۱۹): تبدیل مدار به معادله

مثال سوم:



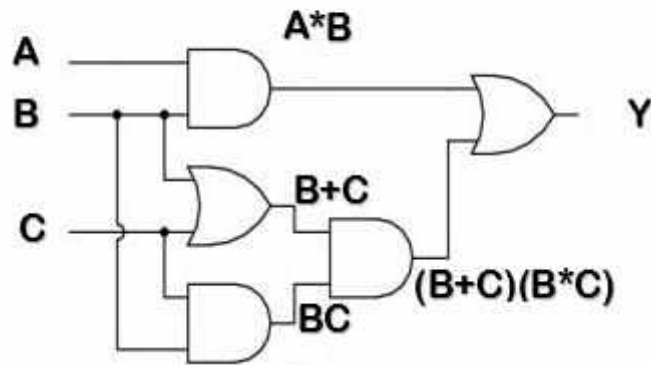
شکل (۳-۲۰): تبدیل مدار به معادله

### ۳.۶ تبدیل معادله به مدار

معادله مورد نظر را تحلیل و بررسی می‌کنیم که مدار شامل چه گیت‌هایی می‌باشد و ترتیب قرار گرفتن گیت‌ها در مدار به چه شکل می‌باشد؟ سپس شروع به رسم مدار می‌کنیم:

مثال اول:

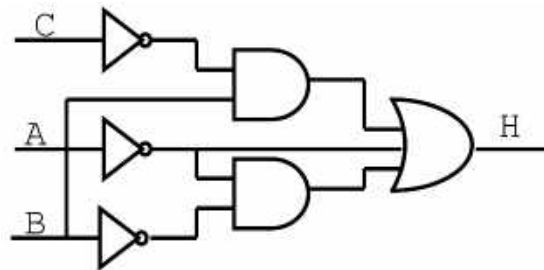
$$Y = (A * B) + ((B + C)(B * C))$$



شکل (۲۱-۳): تبدیل معادله به مدار

مثال دوم:

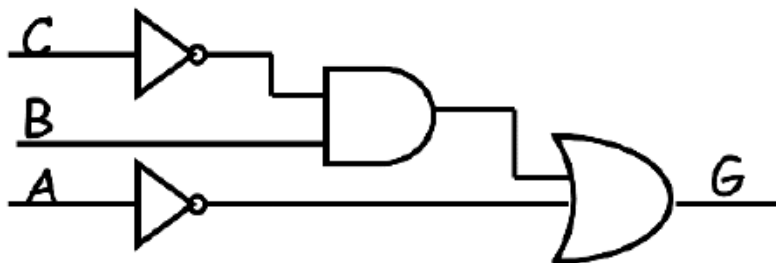
$$H = A' + B.C' + A'.B'$$



شکل (۲۲-۳): تبدیل معادله به مدار

مثال سوم:

$$G = A' + B.C$$



شکل (۲۳-۳): تبدیل معادله به مدار





در این فصل انواع گیت‌های منطقی معرفی گردیدند. گیت‌ها کوچک‌ترین واحد منطقی سازنده مدارهای دیجیتال می‌باشند. در مجموع تعداد ۷ گیت مورد مطالعه قرار گرفت. گیت‌های اصلی شامل AND، OR و NOT و گیت‌های فرعی شامل NOR، NAND، XOR و XNOR می‌باشد. هر گیت دارای سه مشخصه مهم جدول درستی، سمبول و معادله می‌باشد. تمام گیت‌ها به غیر از گیت NOT می‌توانند، تعداد دو یا تعداد بیشتری ورودی داشته باشند. تمامی گیت‌های منطقی تنها یک خروجی دارند. گیت‌ها اگر چندین بیت ورودی داشته باشند، ورودی‌هایشان به صورت مجزا از سایر بیت‌ها دو بیت دو بیت با هم روی‌شان عملیات منطقی صورت می‌گیرد. با ترکیب گیت‌های منطقی با هم مدارهای منطقی ساخته می‌شود. هر مدار، دارای یک معادله نظیر می‌باشد که از روی مدار منطقی قابل نوشتن است. از روی هر معادله منطقی می‌توان مدار نظیر آن معادله را رسم کرد.



۱. مدارهای منطقی معادلات زیر را رسم کنید:

$$(1) Y = A(B \oplus D) + C'$$

$$(2) Y = A + B + B'(A + C)$$

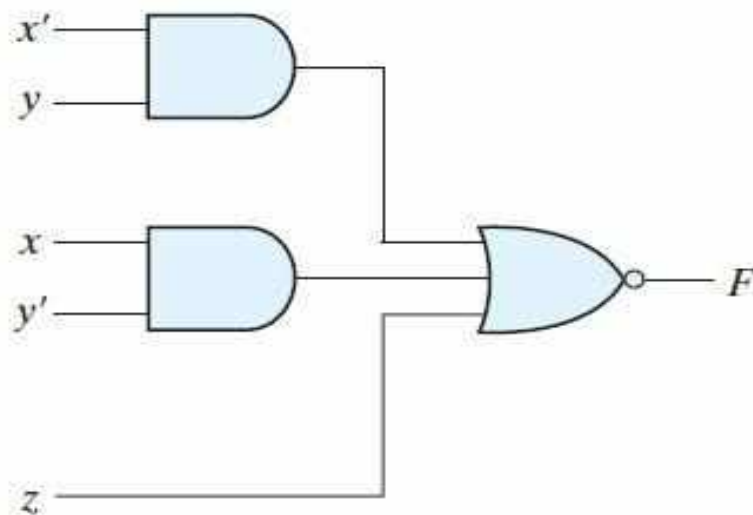
$$(3) Y = A + CD + ABC$$

$$(4) Y = (A \oplus C)' + B$$

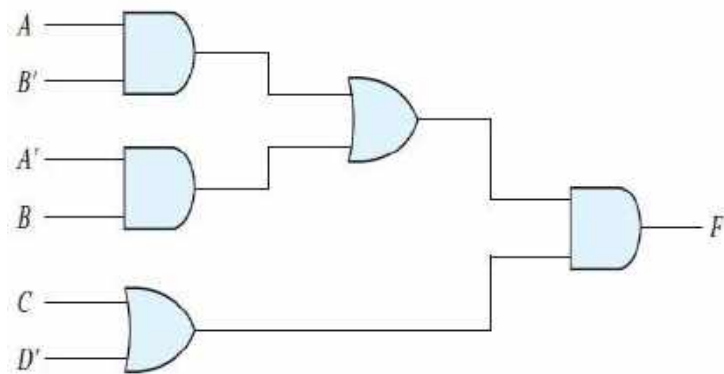
$$(5) Y = (A' + B')(C + D')$$

$$(6) Y = ((A + B')(C' + D))EF$$

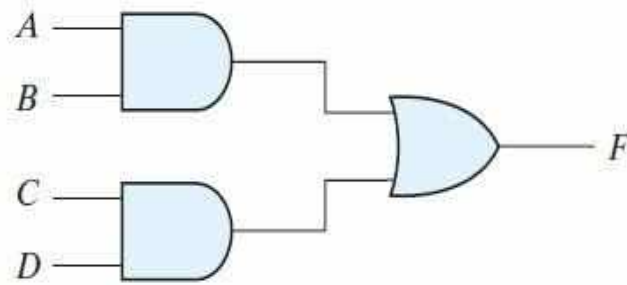
۲. معادله مربوط به مدار منطقی زیر را بنویسید:



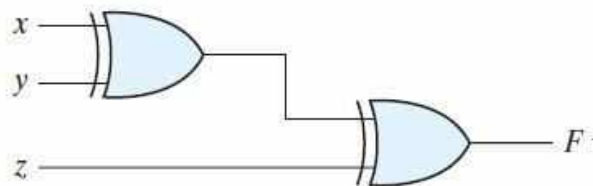
۳. معادله مربوط به مدار منطقی زیر را بنویسید:



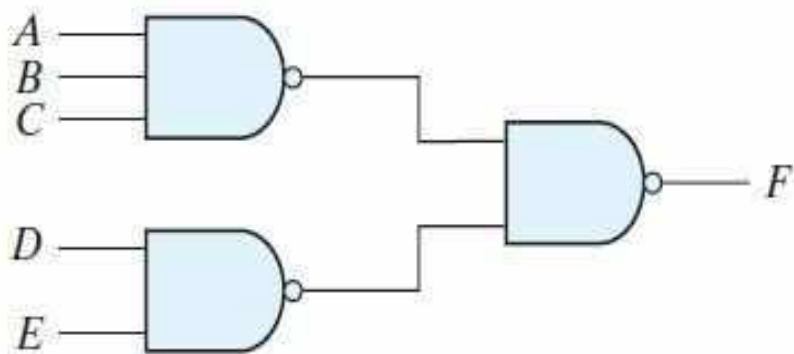
۴. معادله مربوط به مدار منطقی زیر را بنویسید:



۵. معادله مربوط به مدار منطقی زیر را بنویسید:



۶. معادله مربوط به مدار منطقی زیر را بنویسید:



## فصل چهارم

### توابع بولی



هدف کلی: آشنایی محصلان با توابع بولی.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. توابع بولی را تعریف کرده بتوانند.
۲. انواع توابع بولی را شرح دیتا بتوانند.
۳. با استفاده از Karnaugh map توابع بولی را ساده بتوانند.

در فصل‌های قبل با تئوری اعداد دیجیتال، گیت‌های منطقی و جبر بولی آشنا شدید. در این فصل، به توابع بولی می‌پردازیم. هر تابع بولی به یک مدار منطقی دیجیتال اشاره می‌کند. هر مدار منطقی برای خود یک تابع بولی یا معادله بولی معادل خود را دارد. هر تابع بولی می‌تواند مشخص کند که یک مدار با توجه به هر مقدار ورودی چه عکس‌العملی در خروجی‌اش از خود نشان می‌دهد و مقدار خروجی برای یک مدار را مشخص می‌کند. هر مدار منطقی می‌تواند، ورودی‌های مختلفی را دریافت کند که این ورودی‌ها می‌توانند از حالات مختلف و مشخصی پیروی کنند. یک تابع بولی بر اساس حالت‌های مختلف ورودی‌ها می‌تواند، تصمیم بگیرد که مدار در خروجی به چه حالتی برود و خروجی‌ها بر اساس معیارهای از پیش تعریف شده در تابع بولی تغییر خواهند کرد. هر تابع بولی می‌تواند، مشخص کند که مدار منطقی از چه گیت‌هایی تشکیل شده است و گیت‌ها با چه ترتیب و سلسله‌مراتبی در ساختار مدار کنار هم قرار گرفته‌اند. با ساده سازی توابع بولی می‌توان مدارهای منطقی متناظر با هر مدار را نیز ساده نمود. چرا که با ساده سازی توابع بولی مربوط به یک مدار می‌توان مداری معادل با همان مدار قبل و با تمام خصوصیاتش ساخت. مدار جدید به دلیل ساده سازی تعداد کمتری از گیت‌های منطقی را مصرف نموده است. پس هزینه ساخت مدارها کاهش پیدا خواهد کرد. در ضمن، چون تعداد گیت‌ها کاهش پیدا کرده است، بدین ترتیب تعداد سطوح مدار منطقی کاهش یافته است. یک مقدار ورودی از لحظه ورود تا خروج از تعداد گیت‌ها و سطوح کمتری عبور خواهد کرد. پس با سرعت بیشتر و در زمان کمتری مقدار خروجی یک مدار منطقی به دست خواهد آمد. بنابر این، ساده سازی توابع بولی و مدارهای دیجیتال باعث افزایش سرعت مدارهای منطقی و صرفه‌جویی در هزینه و زمان خواهد شد. یکی از بهترین و رایج‌ترین روش‌های ساده سازی توابع بولی استفاده از جدول کارنو می‌باشد. در این فصل، توابع بولی، قوانین و مقررات و روش‌های ساده سازی آن‌ها را مرور خواهیم کرد.

## ۴.۱ توابع بولی

جبر بول، جبری است که با متحول‌های باینری و عملیات منطقی سروکار دارد. یک تابع به وسیله یک عبارت جبری متشکل از متحول‌های باینری، ثابت‌های ۰ و ۱ و سمبول‌های عملیاتی منطقی تشکیل شده است. برای مقدار مفروضی از متحول‌های باینری، تابع می‌تواند ۱ یا ۰ باشد. به عنوان مثال، تابع بولی زیر را در نظر بگیرید:

$$F1 = x + y'z$$

اگر  $x = 1$  یا اگر  $y = 0$  و  $z = 1$  باشد، آن‌گاه  $F = 1$  خواهد بود. یک تابع بول رابطه منطقی را بین متحول‌ها بیان می‌کند. این تابع با تعیین مقدار باینری برحسب همه مقادیر ممکن متحول‌ها ارزیابی می‌شود. یک جدول بولی به صورت یک جدول درستی هم می‌تواند، نشان داده شود. جدول درستی لیستی از ۱‌ها و ۰‌ها است که به متحول‌های باینری تخصیص می‌یابد و ستونی که مقدار نتایج را برای هر ترکیب نشان می‌دهد. تعداد سطرها در جدول درستی  $2^n$  است که  $n$  تعداد متحول‌ها در تابع است. ترکیبات باینری برای جدول درستی از شمارش اعداد باینری و از ۰ تا  $2^n - 1$  بدست می‌آید. جدول ۴-۱ درستی تابع  $F_1$  را نشان

می‌دهد. در این جدول هشت ترکیب باینری ممکن برای تخصیص بیتی به سه متغیر  $Z$  و  $Y$  و  $X$  وجود دارد. ستونی که برچسب  $F_n$  دارد و در ازاء هر ترکیب ۰ یا ۱ است. جدول نشان می‌دهد که وقتی  $x = 1$  یا  $yz = 01$  باشد، تابع  $F_1$  برابر ۱ است.

یک تابع بول را می‌توان از یک عبارت جبری به یک نمودار مداری متشکل از گیت‌های منطقی تبدیل کرد. نمودار مدار منطقی  $F_1$  در شکل (۱-۴) نشان داده شده است. برای تولید متمم ورودی  $Y$  معکوس‌گر وجود دارد. برای جمله  $y'z$  یک گیت AND و برای ترکیب آن دو یک گیت OR بکار رفته است. در نمودارهای مدار منطقی، متحول‌های تابع به عنوان ورودی مدار و متحول باینری  $F_1$  به عنوان خروجی مدار در نظر گرفته می‌شوند. برای نمایش  $F_1$  در یک جدول درستی تنها یک راه وجود دارد. با این وجود، وقتی تابع به شکل یک عبارت جبری است، می‌تواند به شکل‌های متفاوتی نشان داده شود.

$$F_1 = x + y'z$$

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

جدول (۱-۴): جدول درستی تابع بولی  $F$

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

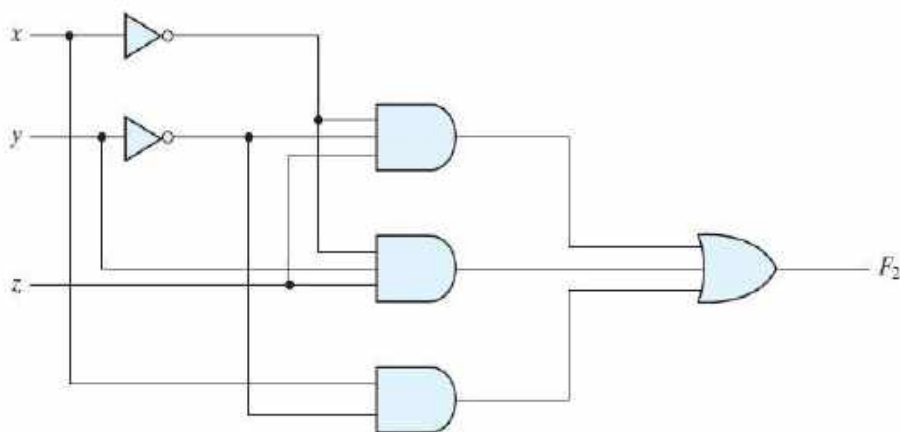
عبارت خاصی که برای مشخص کردن تابع مورد استفاده قرار می‌گیرد، اتصالات میان گیت‌ها در نمودار مدار منطقی را دیکته می‌نماید. گاهی اوقات ممکن است با دستکاری یک عبارت بولی توسط قوانین جبر بول،

عبارت ساده‌تری برای یک تابع بدست آوریم و بنابراین، تعداد گیت‌ها در مدار و تعداد ورودی‌ها به هر گیت را کاهش دهیم. مثلاً، تابع بولی زیر را در نظر بگیرید:

$$F_2 = x'y'z' + x'yz + xy'$$

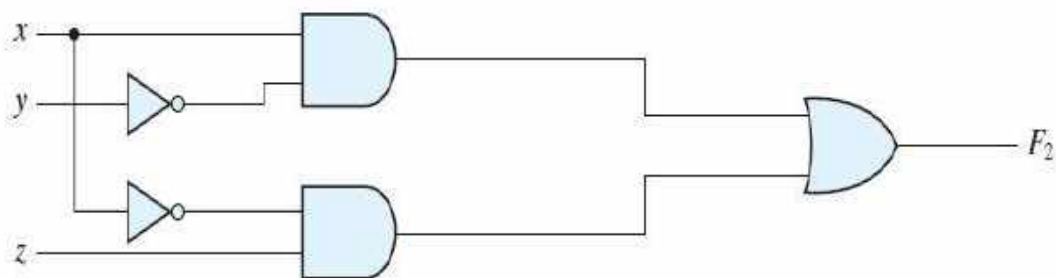
این تابع در شکل (۴-۲) پیاده سازی شده است. متحول‌های  $X$  و  $Y$  به کمک معکوس‌گر متمم شده‌اند، تا  $x'$  و  $y'$  بدست آیند. سه جمله در عبارت با سه گیت AND پیاده سازی شده‌اند. گیت OR نیز، گیت OR منطقی سه جمله را فراهم می‌سازد. جدول درستی  $F_2$  در جدول (۴-۱) آمده است. وقتی که  $xyz = 001$  یا  $011$  و یا  $10$  باشد (بدون توجه به  $z$ )، تابع برابر ۱ است، در غیر این صورت ۰ است. این شرایط چهار ۱ و چهار ۰ برای  $F_2$  تولید می‌کنند.

$$F_1 = x + y'z$$



شکل (۴-۱): مدار تابع بولی  $F_2$

$$F_2 = x'y'z' + x'yz + xy'$$



$$F_2 = xy' + x'z$$

شکل (۴-۲): مدار ساده شده  $F_2$

اکنون ساده سازی ممکن برای تابع را با اعمال بعضی از ویژگی‌های جبر بول ملاحظه کنید:

تابع تنها به دو جمله کاهش یافته و قابل پیاده سازی با گیت مطابق شکل (۳-۴) است. بدیهی است که مدار شکل (۴-۳) ساده‌تر از (۴-۲) می‌باشد، ولی هر دو یک تابع را پیاده سازی می‌کنند. تساوی دو عبارت را می‌توان به کمک جدول درستی هم تحقیق کرد. عبارت ساده شده، وقتی  $xy = 0$  یا  $xy = 1$  باشد، برابر ۱ است. این تابع هم همان چهار ۱ را در جدول تولید می‌کند. چون هر دو عبارت جدول درستی یکسانی را تولید می‌کنند که به آن‌ها معادل گوییم. بنابراین، دو مدار به ازاء همه ترکیبات ممکن متحول‌های ورودی، خروجی‌های یکسانی دارند. هر دو عبارت تابع یکسانی را تولید می‌کنند، ولی یکی از آن‌ها گیت‌ها و ورودی‌های کمتری نسبت به دیگری دارد و بنابراین، چون سیم بندی و قطعات کمتری نیاز است بر دیگری ترجیح دیتا می‌شود.

## ۴.۲ مینترم‌ها<sup>۱</sup> و ماکسترم‌ها<sup>۲</sup>

### ۴.۲.۱ مینترم:

یک متحول باینری ممکن است به شکل معمولی  $X$  و یا متمم  $X'$  ظاهر شود. اکنون تصور کنید که دومتحول باینری  $X$  و  $Y$  با عملگر AND با هم ترکیب شوند. چون هر متحول ممکن است به هر یک از دو شکل فوق ظاهر شود، چهار ترکیب برای آن‌ها متصور است:

$xy, xy', x'y, x'y'$ . هر یک از این چهار جمله AND را یک مینترم یا یک جمله ضرب ستندرد گویند. به‌طور مشابه  $n$  متحول را می‌توان ترکیب کرده و  $2^n$  مینترم به وجود آورد.  $2^n$  مینترم مختلف را می‌توان با روشی مشابه با آنچه در جدول (۲-۴) آمده، نشان داد. اعداد باینری از صفر تا  $2^n - 1$  زیر ستون  $n$  متحول لیست شده‌اند. هر مینترم از AND تمام  $n$  متحول بدست می‌آید که در آن هر متحول معکوس متعلق به بیت ۰ و متحول غیر معکوس با ۱ نشان داده می‌شود. سمبول هر مینترم نیز در جدول با  $m_j$  نشان داده شده است. که در آن  $j$  معادل دسیمال عدد باینری مربوط به مینترم است.

### ۴.۲.۲ ماکسترم

به‌طریق مشابهی،  $n$  متحول یک جمله OR تشکیل می‌دهند که هر متحول ممکن است معکوس و یا غیر معکوس باشد.  $2^n$  ترکیب ممکن را ماکسترم یا جمع ستندرد گویند.

$x + y, x + y', x' + y, x' + y'$  هر یک از این چهار جمله OR را یک ماکسترم می‌گویند. به‌طور مشابه  $n$  متحول را می‌توان ترکیب کرده و  $2^n$  ماکسترم به وجود آورد.  $2^n$  ماکسترم مختلف را می‌توان با روشی مشابه با آنچه در جدول (۲-۴) آمده، نشان داد. اعداد باینری از صفر تا  $2^n - 1$  زیر ستون  $n$  متحول لیست

<sup>1</sup>Minterm

<sup>2</sup>Maxterm



شده‌اند. هر ماکسترم از OR تمام  $n$  متحول بدست می‌آید که در آن هر متحول معکوس متعلق به بیت ۱ و متحول غیر معکوس با ۰ نشان دیتا می‌شود. سمبول هر ماکسترم نیز در جدول با  $M_i$  نشان دیتا شده است. که در آن  $i$  معادل دسیمیل عدد باینری مربوط به ماکسترم است.

#### ۴.۲.۲.۱ مینترم ها و ماکسترم ها برای تابع سه متحوله باینری

جدول (۴-۲): مینترم ها و ماکسترم های تابع ۳ متحوله

علامت	ماکسترم	علامت	مینترم	Z	Y	X
$M_0$	$x + y + z$	$m_0$	$x'y'z'$	0	0	0
$M_1$	$x + y + z'$	$m_1$	$x'y'z$	1	0	0
$M_2$	$x + y' + z$	$m_2$	$x'yz'$	0	1	0
$M_3$	$x + y' + z'$	$m_3$	$x'yz$	1	1	0
$M_4$	$x' + y + z$	$m_4$	$xy'z'$	0	0	1
$M_5$	$x' + y + z'$	$m_5$	$xy'z$	1	0	1
$M_6$	$x' + y' + z$	$m_6$	$xyz'$	0	1	1
$M_7$	$x' + y' + z'$	$m_7$	$xyz$	1	1	1

مثال:

یک تابع بول می‌تواند به صورت جبری با استفاده از جدول درستی و با تشکیل مینترم‌های هر ترکیب از متحول‌هایی که برای تابع، ۱ را تولید می‌کنند و اجرای عملگر روی OR، همه این جملات ایجاد شود. مثلاً، در جدول (۴-۳) با ترکیبات 111 و 100، 001 به صورت  $x'y'z'$  و  $xy'z'$  و  $xyz$  بیان می‌شود. چون هر

یک از این مینترم‌ها  $F=1$  را ایجاد می‌نماید. پس:

$$F_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$$

به سادگی می‌توان نشان داد که:

$$F_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$$

جدول (۳-۴) جدول درستی مثال تابع F

$x$	$y$	$z$	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

این مثال‌ها، خصوصیت مهمی از جبر بول را به نمایش می‌گذارند. یعنی هر تابع بولی را می‌توان به صورت جمع مینترم‌ها نشان داد "جمع به معنی OR جملات است.

اکنون متمم تابع بول را ملاحظه نمایید، می‌توان آن را با تشکیل مینترم‌هایی در جدول درستی که  $\cdot$  تابع را تولید می‌کنند، ایجاد کرد و سپس آن‌ها را OR نمود. متمم را چنین است:

$$F_1 = x'y'z' + x'yz' + x'yz + xy'z + xyz'$$

اگر متمم  $f'_1$  را بدست آوریم تابع  $f_1$  را بدست خواهد آمد.

$$\begin{aligned} F'_1 &= (x + y + z) (x + y' + z) (x' + y + z') (x' + y' + z) \\ &= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6 \end{aligned}$$

بطور مشابه، می‌توان عبارت  $f_2$  را از جدول بدست آورد:

$$\begin{aligned} F_2 &= (x + y + z) (x + y + z') (x + y' + z) (x' + y + z) \\ &= M_0 \cdot M_1 \cdot M_2 \cdot M_4 \end{aligned}$$

این مثال‌ها، نیز دومین خاصیت جبر بول را به نمایش می‌گذارند: هر تابع بول را می‌توان به صورت ضرب ماکسترم‌ها (ضرب به معنی AND است) در آورد. روال تهیه ضرب ماکسترم‌ها مستقیماً از جدول درستی به این شکل است. برای هر ترکیبی از متحول‌ها، ماکسترم‌هایی که در تابع  $\cdot$  تولید می‌کنند را تشکیل دهید، سپس AND همه ماکسترم‌ها را بدست آورید. توابع بول که به صورت جمع مینترم‌ها یا ضرب ماکسترم بیان شوند را شکل متعارف نامند.

### ۴.۳ جمع مینترم‌ها

قبلاً بیان شد که برای هر  $n$  متحول باینری  $2^n$  مینترم مجزا وجود دارد و هر تابع بولی می‌تواند به صورت مجموعی از مینترم‌ها در آید. مینترم‌هایی که جمع آن‌ها توابع بول را تعریف می‌کنند، آن‌هایی هستند که ۱ های تابع را در جدول درستی تشکیل می‌دهند. چون، تابع در قبال هر مینترم می‌تواند ۰ و یا ۱ باشد، چون  $2^n$  مینترم وجود دارد، می‌توان، تعداد توابع ممکن که با  $n$  متحول ایجاد می‌شود را  $2^{2^n}$  دانست. گاهی بهتر است تابع بول را برحسب جمع مینترم‌ها بیان کنیم. اگر در شکل نبود، می‌توان ابتداء آن را به صورت جمع جملات AND در آورد. آنگاه هر ترم برای یافتن همه متحول‌ها در آن واری می‌شود. اگر یک یا چند متحول وجود نداشته باشند، می‌توان جمله را در عبارتی مثل  $AND, x + x'$  نمود، که  $x$  یکی از متحول‌های مفقود شده است. مثال زیر مطلب را روشن می‌کند.

مثال:

تابع بولی  $F = A + B'C$  را به صورت جمع مینترم‌ها درآورید؟

تابع سه متحول  $A, B, C$  دارد و در اولین جمله  $A$ ، دو متحول مفقود است، بنابراین:

$$A = A(B + B') = AB + AB'$$

این تابع هنوز هم یک متحول کسر دارد:

$$\begin{aligned} A &= AB(C + C') + AB'(C + C') \\ &= ABC + ABC' + AB'C + AB'C' \end{aligned}$$

جمله دوم  $B'C$  یک متحول کم دارد.

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

با ترکیب همه جملات داریم:

$$\begin{aligned} F &= A + B'C \\ &= ABC + ABC' + AB'C + AB'C' + A'B'C \end{aligned}$$

دیده می‌شود که  $AB'C$  دوبار تکرار شده است و برحسب تئوری  $(x + x = x)$  می‌توان یکی از آن‌ها را حذف کرد. با مرتب نمودن مینترم‌ها به ترتیب صعودی داریم:

$$\begin{aligned} F &= A'B'C + AB'C' + AB'C + ABC' + ABC \\ &= m_1 + m_4 + m_5 + m_6 + m_7 \end{aligned}$$

جدول (۴-۴): جدول درستی مثال تابع F

A	B	C	F
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	1

گاهی بهتر است تابع بول را وقتی به صورت جمع مینترم‌ها است، به شکل خلاصه زیر نشان دهیم:

$$F(A, B, C) = \sum(1, 4, 5, 6, 7)$$

سمبول جمع  $\sum$  به معنی OR جملات است؛ اعدادی که به دنبال آن می‌آیند، نیز مینترم‌های تابع هستند. حروف داخل قوس در جلو F، لیستی از متحول‌های تشکیل دهنده جملات مینترم را نشان می‌دهند. روش دیگری برای تشکیل مینترم‌های تابع بول، تهیه مستقیم جدول درستی تابع از عبارت جبری و سپس خواندن مینترم‌ها از جدول درستی است. تابع بول، مثال قبل را در نظر بگیرید:

$$F = A + B'C$$

جدول درستی در جدول (۴-۴) مستقیماً از عبارت جبری با لیست هشت ترکیب زیر متحول‌های A، B و C و اعمال ۱ زیر ستون F برای ترکیباتی که در آن  $A = 1$  و  $BC = 0$  است فراهم شده است. سپس از جدول درستی می‌توان مشاهده کرد که مینترم‌های تابع، جملات ۱، ۴، ۵، ۶ و ۷ می‌باشند.

#### ۴.۴ ضرب ماکسترم‌ها

هر یک از  $2^n$  تابع متشکل از n متحول را می‌توان به صورت ضرب ماکسترم‌ها نیز بیان کرد. برای بیان توابع جبر بولی به عنوان ضرب ماکسترم‌ها، ابتداء باید جملات OR را تشکیل دهیم. این کار را می‌توان با استفاده از قانون توزیع پذیری  $x + yz = (x + y)(x + z)$  انجام داد. سپس هر متحول مفقود در هر جمله OR با  $xx'$  می‌شود. این روش با مثال زیر روشن‌تر خواهد شد.

مثال:

تابع بول  $F = xy + x'z$  را به صورت ضرب ماکسترم‌ها نشان دهید. ابتداء تابع را با استفاده از اصل توزیع پذیری به شکل جملات OR در می‌آوریم:

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

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

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

تابع سه متحول دارد:  $x$ ،  $y$  و  $z$ . هر جمله OR فاقد یک متحول است بنابراین:

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

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

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

با ترکیب همه جملات و حذف تکراری‌ها، خواهیم داشت:

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

$$F = M_0 \cdot M_2 \cdot M_4 \cdot M_5$$

نمایش ساده‌تر به شکل زیر است:

$$F(x,y,z) = \prod(0,2,4,5)$$

سمبول ضرب،  $\prod$ ، بیانگر AND ماکسترم‌ها یا ضرب ماکسترم‌ها است. اعداد داخل قوس شماره ماکسترم‌های تابع‌اند.

## ۴.۵ ساده سازی توابع بولی

ساده سازی یا حداقل سازی سطح گیت، اشاره بر یافتن معادله ساده و بهینه دارد که یک مدار دیجیتال را توصیف می‌کند. عمل کرد که کاملاً قابل درک است ولی هنگامی که بیش از چند ورودی وجود دارد، اجرای آن به روش‌های دستی مشکل است. خوشبختانه ابزارهای مبتنی بر کامپیوتر ساخت (سنتز) می‌توانند، مجموعه بزرگی از معادلات بول را به خوبی و سریع‌تر حداقل کنند. با این وجود درک طراح از توصیف ریاضی و حل مسئله اهمیت دارد. این بخش نقش پایه برای درک این عناوین مهم است و شما را قادر می‌سازد تا یک طراحی دستی از یک مدار ساده را انجام دیتا و خود را برای استفاده از ابزار طراحی مدرن آماده کنید.

## ۴.۶ جدول کارنو

پیچیدگی گیت‌های منطقی دیجیتال که یک تابع بول را پیاده سازی می‌کنند، مستقیماً به پیچیدگی عبارات جبری که توسط آن تابع پیاده سازی می‌شوند، بستگی دارد. گرچه جدول درستی یک تابع نمایش منحصر به فردی دارد، اما وقتی به صورت جبری بیان شود، می‌تواند، فرم‌های متفاوتی داشته باشد. عبارات بول را می‌توان به صورت جبری ساده کرد. با این وجود، این روش حداقل سازی به دلیل کمبود قوانین خاص در پیشگویی مرحله بعدی فرآیند دستکاری، مشکل است. روش نقشه، روالی ساده را برای ساده سازی توابع بول پیش پا می‌گذارد. این روش را می‌توان شکل مصور جدول درستی تصور کرد. روش نقشه را نقشه کارنو یا نقشه K هم می‌نامند.

نقشه، نموداری است متشکل از مربعات که هر مربع یک مینترم از تابع را نشان می‌دهد. چون هر تابع بول را می‌توان به مجموعی از مینترم‌ها نشان داد، بنابراین، نتیجه می‌شود که یک تابع بولی در نقشه را می‌توان با مربعاتی که مینترم‌های متعلق به آن‌ها در تابع وجود دارد به صورت گرافیکی شناسایی کرد. در واقع نقشه، نمایشی عینی از همه راه‌هایی است که یک تابع ممکن است در شکل ستندرد داشته باشد. با تشخیص همه الگوهای مختلف، استفاده کننده می‌تواند، عبارات جبری مختلفی برای یک تابع بدست آورده و از میان آن‌ها ساده‌ترین را انتخاب کند. عبارات ساده شده، حاصل از نقشه همیشه به یکی از دو شکل ستندرد، جمع حاصل ضرب‌ها و ضرب حاصل جمع‌ها می‌باشد. فرض بر این است که ساده‌ترین عبارت جبری، دارای حداقل جملات با کم‌ترین لیترال در هر جمله باشد. این فرض نموداری با حداقل گیت را فراهم نموده، تعداد ورودی‌ها به گیت نیز حداقل خواهد بود. بعد خواهیم دید که ساده‌ترین عبارت منحصر به فرد نیست. گاهی ممکن است دو یا چند عبارت بیابیم که معیار حداقل سازی را برآورد. در این حالت هریک از دو حل رضایت بخش خواهد بود.

### ۴.۶.۱ جدول کارنو توابع ۲ متحوله

جدول کارنو، برای توابع ۲ متحوله در شکل زیر نشان داده شده است. در این جدول چهار مینترم برای دو متحول وجود دارد. از این رو جدول متشکل از چهار مربع است که هر یک متعلق به یک مینترم می‌باشد. • و ۱ موجود در هر سطر و ستون مقدار متحول را نشان می‌دهند. متحول  $x$  در سطر 0 معکوس و در سطر 1 غیر معکوس است. به طور مشابه  $y$  در ستون 0 معکوس و در ستون 1 غیرمعکوس است.

هر یک از مینترم‌هایی که در تابع وجود دارند، مقدارشان را در جدول کارنو برابر با ۱ قرار می‌دهیم. مینترم‌هایی که در تابع مربوطه وجود ندارند را ۰ می‌گذاریم. حال داخل جدول کارنو دنبال سطرها و ستون‌هایی می‌گردیم که مقدار همه خانه‌های‌شان برابر با ۱ می‌باشد. ویژگی مشترک آن سطر و ستون را در تابع ساده شده جدید می‌نویسیم. تعداد خانه‌هایی که مشترک با هم می‌گیریم در یک سطر یا ستون باید توانی از ۲ باشد. یعنی ۲ یا ۴ یا ۸ یا ۱۶ و ...

جدول (۴-۵): جدول کارنو ۲ متحوله

		$y$	
		0	1
$x$	0	$m_0$ $x'y'$	$m_1$ $x'y$
	1	$m_2$ $xy'$	$m_3$ $xy$

(a)

(b)

مثال:

معادله سه متحوله زیر را با استفاده از جدول کارنو ساده کنید.

$$F(a, b) = \sum m(1, 2, 3)$$

جدول (۴-۶): مثالی از جدول کارنو ۲ متحوله

		$y$	
		0	1
$x$	0	$m_0$	$m_1$ 1
	1	$m_2$ 1	$m_3$ 1

$x + y$

## ۴.۶.۲ جدول کارنو توابع ۳ متحوله

یک جدول کارنو برای توابع ۳ متحوله در شکل زیر مشاهده می‌شود. برای ۳ متحول ۸ مینترم وجود دارد. بنابراین، جدول کارنو از ۸ خانه تشکیل شده است. توجه کنید که مینترم‌ها بر اساس ترتیب باینری مرتب نشده‌اند، بلکه ترتیب‌شان بر اساس کد مشخصی فهرست شده‌اند. ویژگی این ترتیب این است که هنگام

عبور از یک ستون به ستون مجاورش تنها یک بیت از نظر مقدار تغییر می‌کند. برای نشان دادن رابطه بین مربع‌ها و ۳ متحول جدول، جدول با اعدادی در هر سطر و هر ستون علامت گذاری شده است. مثلاً، خانه متعلق به  $m_5$  مربوط به سطر ۱ و ستون ۰۱ است. وقتی دو عدد در کنار هم قرار گیرند، عدد باینری ۱۰۱ حاصل می‌شود که معادل دسیمال آن عدد ۵ می‌باشد. به طریق دیگری هم می‌توان به خانه  $m_5 = xy'z$  نگاه کرد، به این ترتیب که بگوییم  $m_5$  در سطر مربوط به  $x$  و در ستون متعلق به  $y'z$  (ستون ۰۱) قرار دارد. حال هریک از مینترم‌هایی که در تابع وجود دارند، مقدارشان را در جدول کارنو برابر با ۱ قرار می‌دهیم. مینترم‌هایی که در تابع مربوطه وجود ندارند را ۰ می‌گذاریم. سپس داخل جدول کارنو دنبال سطرها و ستون‌هایی می‌گردیم که مقدار همه خانه‌هایشان برابر با ۱ می‌باشد. ویژگی مشترک آن سطر و ستون را در تابع ساده شده جدید می‌نویسیم.

تعداد خانه‌های هم‌جواری که مشترک باهم می‌گیریم، در یک سطر یا ستون باید همواره توانی از ۲ باشد. یعنی: ۱ یا ۲ یا ۴ یا ۸ و....

هر قدر تعداد بیشتری از خانه‌ها یا مربعات هم‌جوار ترکیب شوند، جمله حاصل ضرب نتیجه، تعداد کم‌تری لیترال خواهد داشت.

یک خانه یک مینترم را نمایش می‌دهد و دارای ۳ لیترال است.

دو خانه مجاور یک جمله ۲ لیترال را نشان می‌دهند.

چهار خانه هم‌جوار یک جمله با ۱ لیترال را نشان می‌دهند.

هشت خانه هم‌جوار که تمام جدول را می‌پوشانند، همواره تابع ۱ را تولید می‌کنند.

جدول (۷-۴): جدول کارنو ۳ متحوله

				$y$			
				$yz$			
				00	01	11	10
$x$	0	$m_0$ $x'y'z'$	$m_1$ $x'y'z$	$m_3$ $x'yz$	$m_2$ $x'yz'$		
	1	$m_4$ $xy'z'$	$m_5$ $xy'z$	$m_7$ $xyz$	$m_6$ $xyz'$		
				$z$			

$m_0$	$m_1$	$m_3$	$m_2$
$m_4$	$m_5$	$m_7$	$m_6$

(a)

(b)



### مثال

معادله سه متحولۀ زیر را با استفاده از جدول کارنو ساده کنید.

$$F_1(a, b, c) = \sum m(2, 3, 4, 5)$$

جدول (۴-۸): مثالی از جدول کارنو ۳ متحولۀ

		$y$		
		11	10	
$x$	$yz$	00	01	
0		$m_0$	$m_1$	
				$x'y$
1		$m_4$	$m_5$	
				$xy'$
				$z$

$$F_1(a, b, c) = \sum m(2, 3, 4, 5) = x'y + xy'$$

### مثال

معادله سه متحولۀ زیر را با استفاده از جدول کارنو ساده کنید.

$$F_2(x, y, z) = \sum m(0, 2, 4, 5, 6)$$

جدول (۴-۹): مثالی از جدول کارنو ۳ متحولۀ

		$y$		
		11	10	
$x$	$yz$	00	01	
0		$m_0$	$m_1$	
				$y'z'$
1		$m_4$	$m_5$	
				$xy'$
				$z$

$$F = z' + xy'$$

## مثال

معادله سه متحولۀ زیر را با استفاده از جدول کارنو ساده کنید.

$$F(A, B, C) = \sum m(1, 2, 3, 5, 7)$$

جدول (۴-۱۰): مثالی از جدول کارنو ۳ متحولۀ

		$B$			
		$BC$		$A'B$	
		00	01	11	10
$A$	0	$m_0$	$m_1$	$m_3$	$m_2$
			1	1	1
$A$	1	$m_4$	$m_5$	$m_7$	$m_6$
			1	1	
		$C$			

$$F = C + A'B$$

### ۴.۶.۳ جدول کارنو توابع ۴ متحولۀ

جدول کارنو، برای توابع ۴ متحولۀ در شکل زیر نشان داده شده است. در جدول ۱۶ جمله مینترم فهرست شده و به هر یک خانه‌ای تخصیص داده شده است، سطرها و ستون‌ها براساس کد خاص شماره گذاری شده است. بین هر دو سطر یا ستون مجاور تنها یک رقم تغییر می‌کند. مینترم متعلق به هر خانه از ترکیب شماره سطر و شماره ستون به دست می‌آید. مثلاً، وقتی اعداد سطر سوم ۱۱ و ستون دوم ۰۱ ترکیب شوند، عدد باینری ۱۱۰۱ حاصل می‌گردد که معادل عدد دسیمال ۱۳ است. بنابراین، جدول در سطر سوم و ستون دوم مینترم  $m_{13}$  را نمایش می‌دهد. ساده کردن توابع بولی ۴ متحولۀ مشابه با روش به کار رفته برای توابع ۳ متحولۀ است. مربعات مجاور مربعاتی هستند که در کنار یکدیگر هستند. به علاوه جدول در سطحی واقع است و لبه‌های بالا و پایین و چپ و راست آن نیز مجاور است تا به این ترتیب مربعات هم‌جوار را بسازند. مثلاً:  $m_0$  و  $m_2$  و نیز  $m_3$  و  $m_{11}$  هر کدام خانه‌های مجاور را می‌سازند.

ترکیب خانه‌های هم‌جوار به راحتی با بررسی جدول ۴ متحولۀ قابل تشخیص است:

یک خانه یک مینترم را نمایش می‌دهد و جمله آن ۴ لیترالی است.

دو خانه هم‌جوار یک جمله ۳ لیترالی را می‌سازند.

چهار خانه هم‌جوار یک جمله ۲ لیترالی را نشان می‌دهند.

هشت خانه هم‌جوار یک جمله ۱ لیترالی را نمایش می‌دهند.

شانزده خانه هم‌جوار تابعی برابر با ۱ را تولید می‌کنند.

هیچ ترکیب دیگری از خانه‌ها یا مربع‌ها نمی‌تواند، تابع را ساده کند.

جدول (۴-۱۱): جدول کارنو ۴ متحول

$m_0$	$m_1$	$m_3$	$m_2$
$m_4$	$m_5$	$m_7$	$m_6$
$m_{12}$	$m_{13}$	$m_{15}$	$m_{14}$
$m_8$	$m_9$	$m_{11}$	$m_{10}$

(a)

(b)

مثال: معادله ۴ متحوله زیر را با استفاده از جدول کارنو ساده کنید.

$$F(w, x, y, z) = \sum m(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$$

جدول (۴-۱۲): مثالی از جدول کارنو ۴ متحول

$$F = y' + w'z' + xz'$$

$$F = y' + w'z' + xz'$$

## مثال

معادله ۴ متحولۀ زیر را با استفاده از جدول کارنو ساده کنید.

$$F(A,B,C,D)=\Sigma(0,1,2,5,8,9,10)$$

جدول ۱۱-۴

		CD		C		CD	
AB		00	01	11	10		
BC'D'	00	m <sub>0</sub> 1	m <sub>1</sub> 1	m <sub>3</sub> 0	m <sub>2</sub> 1		
	01	m <sub>4</sub> 0	m <sub>5</sub> 1	m <sub>7</sub> 0	m <sub>6</sub> 0	BCD'	
A	11	m <sub>12</sub> 0	m <sub>13</sub> 0	m <sub>15</sub> 0	m <sub>14</sub> 0	B	
	10	m <sub>8</sub> 1	m <sub>9</sub> 1	m <sub>11</sub> 0	m <sub>10</sub> 1	AB	
				D			

Note:  $BC'D' + BCD' = BD'$



با استفاده از قوانین مربوط به توابع بولی می‌توان یک مدار منطقی را از یک شکل به شکل دیگر معادل آن مدار تبدیل کرد. با ساده سازی توابع بولی، می‌توان مدارهای منطقی متناظر با هر مدار را نیز ساده نمود. چرا که با ساده سازی توابع بولی مربوط به یک مدار می‌توان مداری معادل با همان مدار قبل و با تمام خصوصیاتش ساخت. مدار جدید به دلیل ساده سازی تعداد کمتری از گیت‌های منطقی را مصرف نموده است. پس هزینه ساخت مدارها کاهش پیدا خواهد کرد. در ضمن چون تعداد گیت‌ها کاهش پیدا کرده است، بدین ترتیب تعداد سطوح مدار منطقی کاهش یافته است. یک مقدار ورودی از لحظه ورود تا خروج از تعداد گیت‌ها و سطوح کمتری عبور خواهد کرد. پس با سرعت بیشتر و در زمان کمتری مقدار خروجی یک مدار منطقی به دست خواهد آمد. بنابر این، ساده سازی توابع بولی و مدارهای دیجیتال باعث افزایش سرعت مدارهای منطقی و صرفه جویی در هزینه و زمان خواهد شد. یکی از بهترین و رایج‌ترین روش‌های ساده سازی توابع بولی استفاده از جدول کارنو می‌باشد.



۱. ابتدا برای معادله زیر جدول درستی (Truth Table) مربوطه را بسازید. سپس به کمک جدول درستی معادله را به صورت جمع مینترم‌ها بنویسید؟

$$A(B+CD) = 1$$

۲. معادله زیر را به صورت جمع مینترم‌ها و ضرب ماکسترم‌ها بنویسید؟

$$F = Y' + X'Z'$$

۳. معادله مربوط به جدول درستی زیر را بنویسید؟

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

۴. معادله‌های دو متحولۀ زیر را با استفاده از جدول کارنو ساده کنید؟

$$W(A,B) = AB' + A'B'$$

$$X(A,B) = AB + AB'$$

$$Y(A,B) = A'B + AB' + A'B'$$

$$Z(A,B) = AB + AB' + A'B$$

۵. معادله‌های ۳ متحولۀ زیر را با استفاده از جدول کارنو ساده کنید؟

$$F(A, B, C) = \sum m(0, 1, 2, 3, 4, 5)$$

$$G(x, y, z) = \sum m(0, 1, 6, 7)$$

$$H(A, B, C) = \sum m(0, 2, 3, 4, 6)$$

$$I(x, y, z) = \sum m(0, 1, 2, 3, 7)$$

$$J(x, y, z) = \sum m(1, 2, 3, 6, 7)$$

$$K(A, B, C) = \sum m(0, 1, 5, 7)$$

۶ - معادله‌های ۴ متحولۀ زیر را با استفاده از جدول کارنو ساده کنید؟

$$F(A, B, C, D) = \sum m(0, 1, 2, 3, 4, 5, 8, 9, 10, 11)$$

$$F(w, x, y, z) = \sum m(0, 1, 2, 3, 4, 5, 8, 9, 13, 15)$$

$$F(A, B, C, D) = \sum m(4, 6, 7, 15)$$

$$F(w, x, y, z) = \sum m(3, 7, 11, 13, 14, 15)$$

$$F(A, B, C, D) = \sum m(1, 4, 5, 6, 12, 14, 15)$$

$$F(w, x, y, z) = \sum m(0, 1, 2, 4, 5, 7, 11, 15)$$

$$F(A, B, C, D) = \sum m(2, 3, 10, 11, 12, 13, 14, 15)$$

$$F(w, x, y, z) = \sum m(0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$$

## فصل پنجم

### مدارهای ترکیبی



هدف کلی: آشنایی محصلان با مدارهای ترکیبی.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

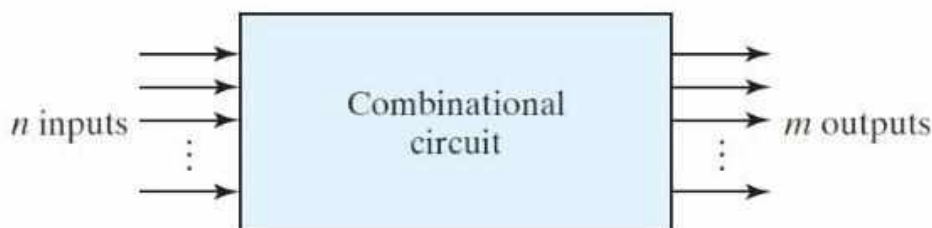
۱. مدارهای ترکیبی را تعریف نمایند.
۲. مدارهای ترکیبی را شرح دهند.
۳. مدارهای ترکیبی طراحی بتوانند.
۴. مدارها را برای عملیات محاسباتی بدانند.



مدارهای منطقی در سیستم‌های دیجیتال می‌توانند، از نوع ترکیبی و یا ترتیبی باشند. یک مدار ترکیبی متشکل از تعدادی گیت منطقی است که خروجی آن‌ها در هر لحظه‌ای از زمان مستقیماً به وسیله ورودی‌های همان لحظه معین می‌شود و به ورودی‌های قبلی بستگی ندارد. این نوع مدار، پردازشی را انجام می‌دهد که با مجموعه‌ای از توابع بولی مشخص می‌شود. مدارهای ترتیبی علاوه بر گیت‌های منطقی از عناصر حافظه نیز استفاده می‌کنند. خروجی‌های آن‌ها تابعی از ورودی‌ها و حالت عناصر حافظه است. در نتیجه خروجی یک مدار ترتیبی نه تنها به مقادیر فعلی ورودی‌ها، بلکه به ورودی‌های قبلی وابسته بوده و عملکرد مدار باید به وسیله حالات داخلی و ترتیب زمانی ورودی‌ها مشخص شود.

## ۵.۱ مدارهای ترکیبی<sup>۱</sup>

یک مدار ترکیبی از متحول‌های ورودی، گیت‌های منطقی و متحول‌های خروجی تشکیل شده است. گیت‌های منطقی سیگنال‌هایی را از ورودی‌ها دریافت کرده و سیگنال‌هایی را برای خروجی‌ها تولید می‌نمایند. این فرآیند معلومات باینری مفروض در ورودی را به معلومات موردنیاز در خروجی تبدیل می‌کند. نمودار کلی یک مدار ترکیبی در شکل دیده می‌شود. متحول باینری ورودی از منبع بیرونی دریافت و  $m$  متحول خروجی به مقصد بیرونی ارسال می‌شوند. هر متحول ورودی و یا خروجی به‌طور فیزیکی به‌صورت یک سیگنال نشان داده می‌شوند و این سیگنال‌ها نیز ۰ و ۱ منطقی را نمایش می‌دهند. در بسیاری از کاربردها، منبع و مقصد، رجیسترهای ذخیره‌سازی هستند. اگر رجیسترها به همراه گیت‌های منطقی به کار روند، کل مدار به نام مدار ترتیبی شناخته خواهد شد.



شکل (۵-۷): مدارات ترکیبی

برای  $n$  متحول ورودی، ۲ ترکیب ممکن باینری از ورودی‌ها وجود دارد. برای هر ترکیب ممکن از ورودی‌ها، فقط یک مقدار برای خروجی موجود است. بنابراین، یک مدار ترکیبی با یک جدول درستی که مقادیر خروجی‌ها را در برابر هر ترکیب از متحول‌های ورودی لیست می‌نماید، نشان داده می‌شود. یک مدار ترکیبی با  $m$  تابع بولی نیز قابل نمایش است که هر یک متعلق به یک خروجی است. هر تابع خروجی برحسب  $n$  متحول ورودی بیان می‌شود.

<sup>۱</sup> Combinational Logic

در فصل ۱، در مورد اعداد و کدهای باینری که کمیت‌های گسسته‌یی از معلومات را نمایش می‌دهند، مطالبی را آموختیم. متحول‌های باینری به طور فیزیکی با ولتاژها و دیگر انواع سیگنال‌ها نشان دیتا می‌شوند. سیگنال‌ها نیز در سیستم‌های منطقی دیجیتال برای اجرای توابع مورد نیاز دستکاری می‌شوند. در ادامه جبر بول را به عنوان روشی جبری در بیان توابع منطقی معرفی نمودیم. آموختیم که چگونه عبارت بول را برای دستیابی به یک پیاده سازی اقتصادی، ساده کنیم. در این فصل با استفاده از معلومات فصل‌های قبل، تحلیل و طراحی مدارهای ترکیبی را فرموله می‌نماییم. با حل مثال‌های نمونه‌یی، فهرستی از توابع اصلی مهم برای درک سیستم‌های دیجیتال فراهم خواهد شد.

مدارهای ترکیبی متعددی وجود دارند که در طراحی سیستم‌های دیجیتال به کرات به کار می‌روند. این مدارها به صورت مجتمع در دسترس بوده و به عنوان قطعات استاندارد دسته بندی شده‌اند. آن‌ها توابع دیجیتال خاصی را که عموماً در طراحی سیستم‌های دیجیتال مورد نیازند، اجرا می‌کنند. در این فصل، ما مهم‌ترین مدارهای ترکیبی استاندارد، مانند جمع کننده‌ها، تفریق کننده‌ها، دیکودرها، اینکدرها و مولتی پلکسرها را معرفی می‌کنیم. این قطعات، به صورت مدارهای مجتمع  $MSI^1$  مجتمع با فشردگی متوسط در دسترس‌اند. به آن‌ها در مدارهای پیچیده  $VLSI^2$ ، مانند مدارهای مجتمع خاص  $ASIC^3$ ، سلول‌های استاندارد هم می‌گویند. توابع سلول‌های استاندارد در داخل مدارهای  $VLSI$  به همان شکل به هم متصل می‌شوند که در طراحی  $MSI$  متشکل از چند  $IC$ ، وصل شدند.

## ۵.۲ نیم جمع کننده<sup>۴</sup>

نیم جمع کننده یا Half Adder یک مدار ترکیبی است که دو بیت ورودی را باهم جمع منطقی می‌کند و حاصل جمع و رقم کری را تولید می‌کند دیاگرام و جدول دیاگرام و جدول درستی نیم جمع کننده به صورت شکل ذیل می‌باشد

جدول (۵-۲): جدول درستی نیم جمع کننده

X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$C = x \cdot y \rightarrow C = x \text{ And } y$$

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

<sup>۱</sup> Medium Scale Integration

<sup>۲</sup> Very Large Scale Integration

<sup>۳</sup> Application Specific Integrated Circuit

<sup>۴</sup> Half Adder

```

graph LR
    x1((x)) --- AND1[AND]
    yp1((y')) --- AND1
    AND1 --- OR1[OR]
    xp1((x')) --- AND2[AND]
    y1((y)) --- AND2
    AND2 --- OR1
    OR1 --- S((S))
    x2((x)) --- AND3[AND]
    y2((y)) --- AND3
    AND3 --- C((C))
  
```

### ۵.۳ تمام جمع کننده<sup>۱</sup>

جدول (۵-۳): جدول درستی تمام جمع کننده

x	Y	Z	C	S
•	•	•	•	•
•	•	\	•	\
•	\	•	•	\
•	\	\	\	•

۷۳

۱	۰	۰	۰		۱	$xy'z'$
۱	۰	۱	۱	$xy'z$	۰	
۱	۱	۰	۱	$xyz'$	۰	
۱	۱	۱	۱	$xyz$	۱	$xyz$

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

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

جدول (۴-۵): ساده سازی تابع S

		$y$			
		$yz$	00	01	11
$x$	0	$m_0$	$m_1$	$m_3$	$m_2$
	1	$m_4$	$m_5$	$m_7$	$m_6$
			1		1
		1		1	

معادله S ساده نمی

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

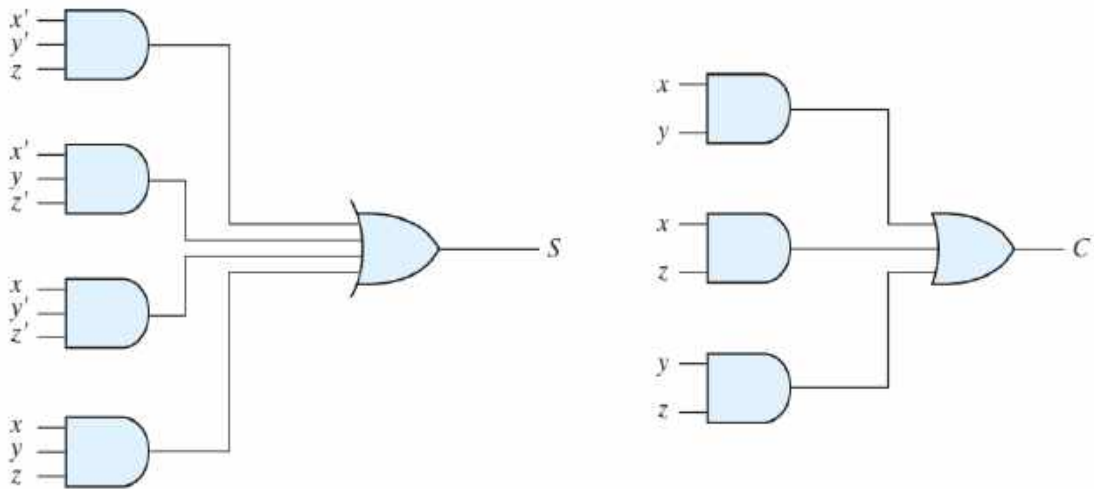
جدول (۵-۵): ساده سازی تابع C

		$y$			
		$yz$	00	01	11
$x$	0	$m_0$	$m_1$	$m_3$	$m_2$
	1	$m_4$	$m_5$	$m_7$	$m_6$
				1	
			1	1	1

ساده شده معادله‌ی C:

$$C = xy + xz + yz$$

مدار تمام جمع کننده:



شکل (۵-۱۰): مدار تمام جمع کننده (به شکل ساده)

### ۵.۳.۱ پیاده‌سازی تمام جمع کننده توسط دو نیم‌جمع کننده و یک گیت OR

برای پیاده‌سازی تمام جمع کننده توسط دو نیم‌جمع کننده و یک گیت OR، معادله S و C را به شکل متفاوتی زیر ساده می‌کنیم:

معادله مربوط به S به صورت زیر ساده می‌شود:

$$\begin{aligned} S &= z \oplus (x \oplus y) \\ &= z'(xy' + x'y) + z(xy' + x'y)' \\ &= z'(xy' + x'y) + z(xy + x'y') \\ &= xy'z' + x'yz' + xyz + x'y'z \end{aligned}$$

معادله مربوط به متحول S به شکلی ساده شد که با یک گیت XOR سه ورودی و خروجی های X، y و z می توان S را در خروجی به دست آورد.

معادله مربوط به متحول C به شکل زیر ساده می شود:

برای ثبوت معادله ذیل از قضیه دمورگان استفاده می نمایم.

$$(XY' + X'Y)' = (XY)'(X'Y)' = (X' + Y'')(X'' + Y')$$

نظر به قانون جبر بول داریم که:

$$Y'' = Y$$

$$X'' = X$$

$$= (X' + Y)(X + Y') = (XX' + X'Y' + XY + YY')$$

نظر به قانون جبر بول داریم که:

$$XX' = 0$$

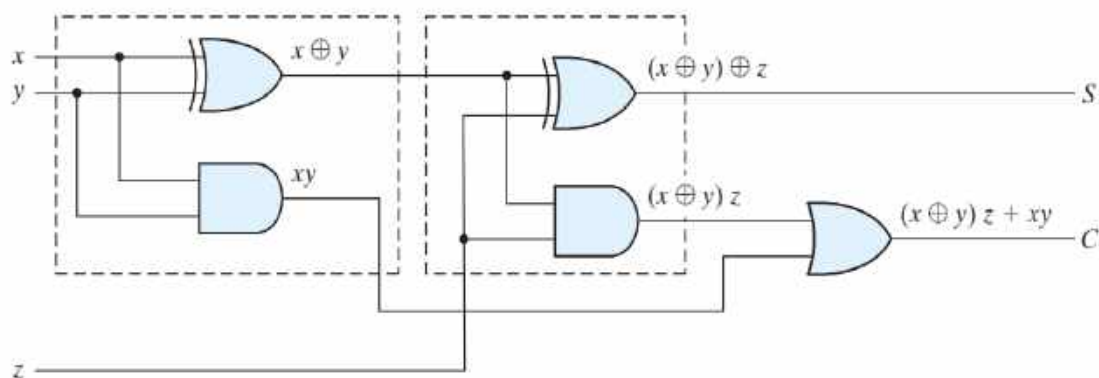
$$YY' = 0$$

پس معادله بدست آمده عبارت اند از:

$$= (0 + X'Y' + XY + 0) = (X'Y' + XY)$$

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

مدار تمام جمع کننده، حاصل از معادلات ساده شده فوق، تشکیل شده از مدار ۲ نیم جمع کننده و یک گیت منطقی OR به شکل زیر می باشد:



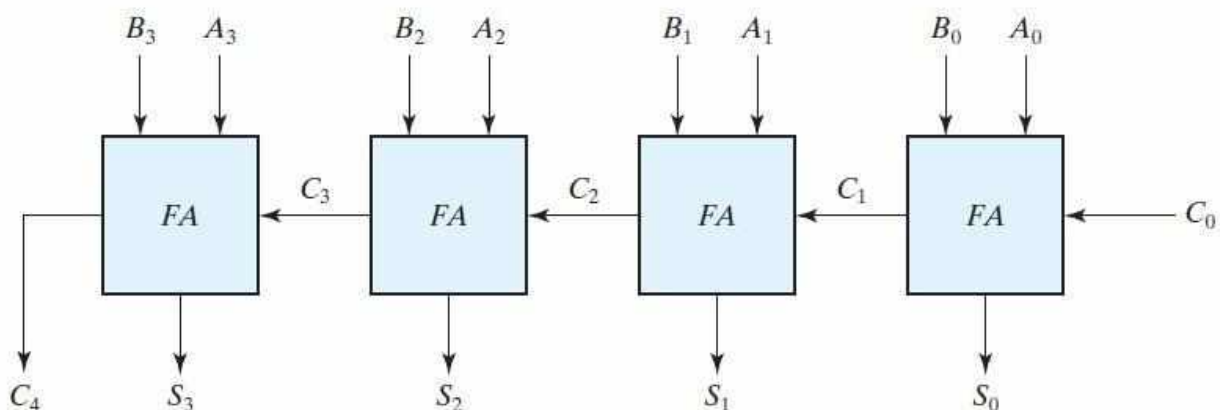
شکل (۵-۵): مدار تمام جمع کننده توسط ۲ نیم جمع کننده و یک گیت OR

## ۵.۴ جمع کننده چند بیتی<sup>۱</sup>

جمع کننده چند بیتی یا Ripple Carry Adder (RCA) مداری است که دو عدد چند بیتی را باهم به صورت باینری جمع می کند. در ورودی اعداد  $A$  و  $B$  به عنوان ورودی جمع کننده وارد می شوند. بعد از محاسبه، نتیجه مجموع دو عدد باینری  $A$  و  $B$  توسط جمع کننده، نتیجه به عنوان  $S$  در خروجی نشان داده می شود. اگر اعداد ورودی  $n$  بیت باشد، در ساخت این مدار از یک نیم جمع کننده و به تعداد  $n-1$  تمام جمع کننده استفاده می شود.

$$\begin{array}{r}
 C_3 \quad C_3 \quad C_3 \\
 a_3 \quad a_2 \quad a_1 \quad a_0 \\
 + \quad b_3 \quad b_2 \quad b_1 \quad b_0 \\
 \hline
 C_4 \quad S_3 \quad S_2 \quad S_1 \quad S_0
 \end{array}$$

## ۵.۵ جمع کننده و تفریق کننده<sup>۲</sup>



شکل (۵-۶): مدار جمع کننده چند بیتی یا RCA

مدار جمع کننده و تفریق کننده یا Adder-Subtractor، مداری می باشد که قابلیت انجام عملیات جمع و هم عملیات تفریق را دارد. این مدار می تواند جمع باینری دو عدد و یا حاصل تفریق دو عدد را حساب کند. در ساخت این مدار به مقدار اعداد  $n$  بیتی به تعداد  $n$  تمام جمع کننده و تعداد  $n$  گیت XOR نیاز است.

<sup>۱</sup> Ripple Carry Adder (RCA)

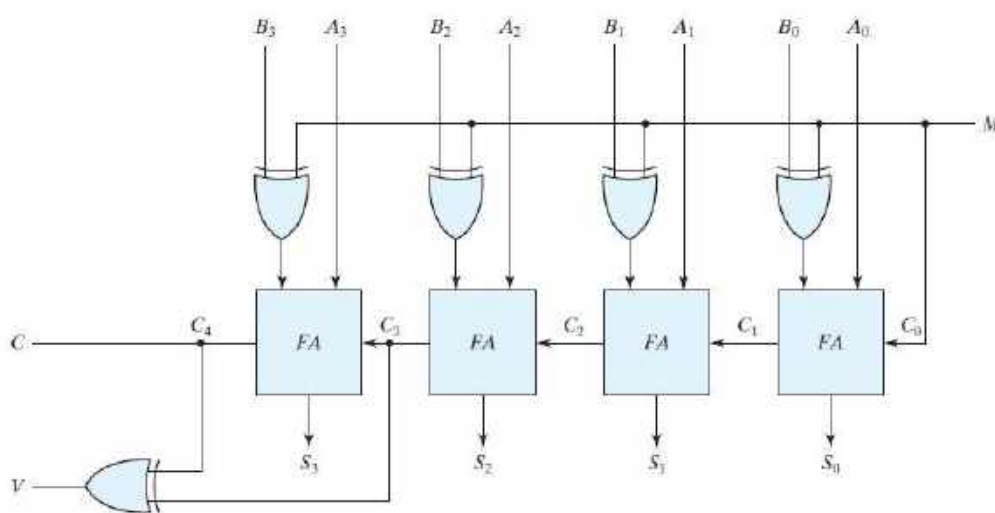
<sup>۲</sup> Adder-Subtractor

اگر مقدار بیت کنترولی  $M$  برابر با عدد ۰ باشد، اعداد  $A$  و  $B$  را با هم جمع می‌کند. در صورتی که مقدار بیت کنترولی  $M$  برابر با عدد ۱ باشد، مدار عدد  $B$  را از عدد  $A$  تفریق خواهد کرد.

$$M = 0 \Rightarrow A + B$$

$$M = 1 \Rightarrow A - B$$

## ۵.۶ دیکودر (Decoder)



شکل (۷-۵): مدار جمع کننده و تفریق کننده چند بیتی

کمیت‌های گسسته معلوماتی در سیستم‌های دیجیتال با کدهای باینری نشان داده می‌شوند. یک کد باینری  $n$  بیتی قادر است تا  $2^n$  عنصر گسسته معلومات کد شده را نشان دهد. یک دیکدر مدار ترکیبی است که معلومات باینری را از  $n$  خط ورودی به حداکثر  $2^n$  خط خروجی به منحصر به فرد تبدیل می‌کند.

هدف از آن‌ها تولید  $2^n$  مینترم از  $n$  متحول ورودی است. دیکدر مدار منطقی است که تعداد  $n$  بیت ورودی و تعداد  $2^n$  بیت خروجی دارد. مدارهای دیکدر ۲ به ۴ و دیکدر ۳ به ۸ و همچنین دیکدر ۴ به ۱۶ در ادامه شرح داده شده‌اند.

### ۵.۶.۱ دیکدر ۲ به ۴

این دیکدر تعداد ۲ ورودی و تعداد ۴ خروجی دارد. این دیکدر می‌تواند، برای تابع دو متحوله مینترمه‌هایش را مشخص کند. در ساخت این مدار ترکیبی از ۴ عدد گیت منطقی AND استفاده شده است.

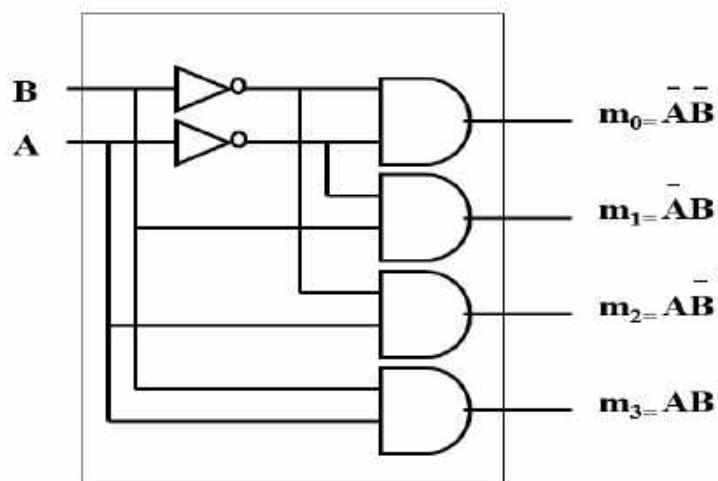
جدول درستی:



جدول (۵-۶): جدول درستی دیکدر ۲ به ۴

A	B	$m_0$	$m_1$	$m_2$	$m_3$
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

مدار منطقی



شکل (۵-۸): مدار ترکیبی دیکدر ۲ به ۴

## ۵.۶.۲ دیکدر ۳ به ۸

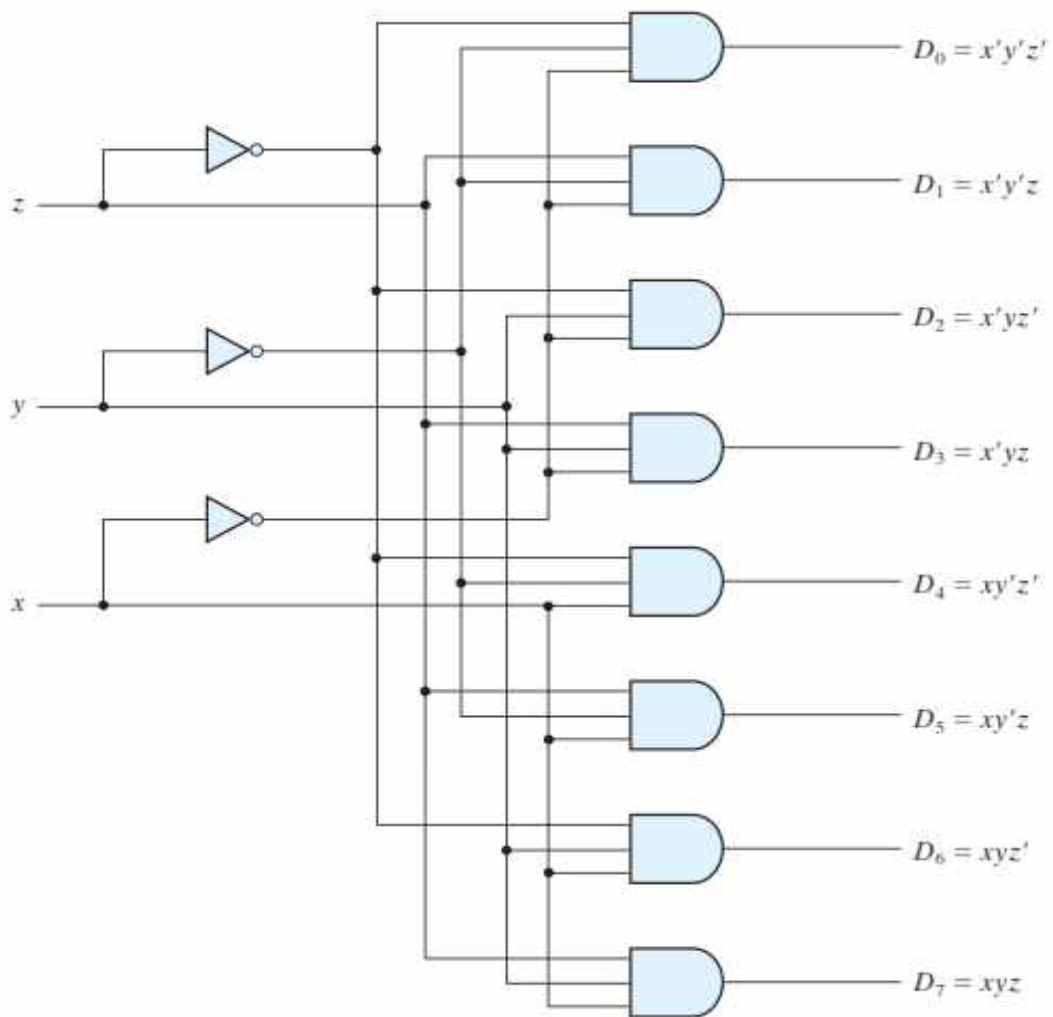
این دیکدر تعداد ۳ ورودی و تعداد ۸ خروجی دارد. این دیکدر می‌تواند، برای تابع ۳ متحوله مینترم‌هایش را مشخص کند. در ساخت این مدار ترکیبی از ۸ عدد گیت منطقی AND استفاده شده است.

جدول (۷-۵): جدول درستی دیکدر ۳ به ۸

Truth Table of a Three-to-Eight-Line Decoder

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

مدار منطقی



شکل (۷-۹) مدار ترکیبی دیکدر ۳ به ۸

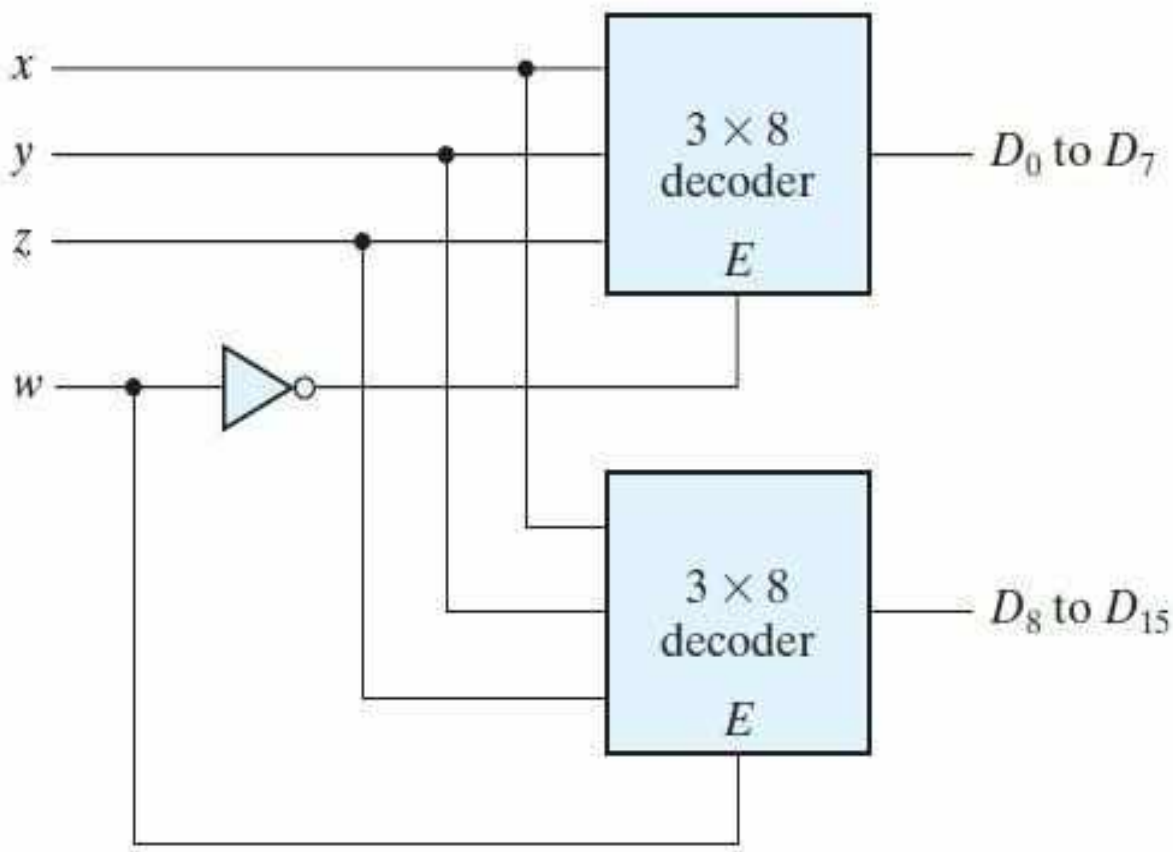
### ۵.۶.۳ دیکدر ۴ به ۱۶

این دیکدر تعداد ۴ ورودی و تعداد ۱۶ خروجی دارد. این دیکدر می‌تواند، برای تابع ۴ متحوله مینترم‌هایش را مشخص کند. در ساخت این مدار ترکیبی از دو دیکدر ۳ به ۸ استفاده شده است.

جدول درستی:

جدول (۵-۸): جدول درستی دیکدر ۴ به ۱۶

W	X	Y	Z	D
0	0	0	0	$D_0$
0	0	0	1	$D_1$
0	0	1	0	$D_2$
0	0	1	1	$D_3$
0	1	0	0	$D_4$
0	1	0	1	$D_5$
0	1	1	0	$D_6$
0	1	1	1	$D_7$
1	0	0	0	$D_8$
1	0	0	1	$D_9$
1	0	1	0	$D_{10}$
1	0	1	1	$D_{11}$
1	1	0	0	$D_{12}$
1	1	0	1	$D_{13}$
1	1	1	0	$D_{14}$
1	1	1	1	$D_{15}$



شکل (۵-۱۰): مدار ترکیبی دیکدر ۴ به ۱۶

## ۵.۷ اینکدر<sup>۱</sup>

یک، اینکدر مداری است که عکس عمل یک دیکدر را انجام می‌دهد. یک اینکدر دارای  $2^n$  خط ورودی و  $n$  خط خروجی است. خطوط خروجی کد باینری مربوط به مقدار باینری ورودی را تولید می‌نماید.

در این جا مدار دیکدر ۴ به ۲ و نیز دیکدر ۸ به ۳ نمایش داده شده است.

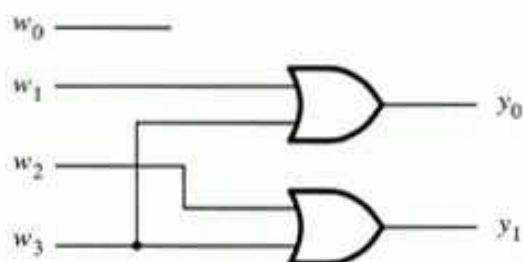
### ۵.۷.۱ اینکدر ۴ به ۲

اینکدر ۴ به ۲ تعداد ۴ ورودی و تعداد ۲ خروجی دارد. این اینکدر می‌تواند، برای تابع ۲ متحوله مقدار مینترمها را گرفته و مقدار متحول‌هایش را در خروجی‌اش مشخص کند. در ساخت این مدار ترکیبی از دو گیت OR استفاده شده است.

<sup>۱</sup> Encoder

جدول (۵-۸): جدول درستی اینکدر ۴ به ۲

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



شکل (۵-۱۱): مدار ترکیبی اینکدر ۴ به ۲

## ۵.۷.۲ اینکدر ۸ به ۳

اینکدر ۸ به ۳ تعداد ۸ ورودی و تعداد ۳ خروجی دارد. این اینکدر می‌تواند، برای تابع ۳ متحوله مقدار مینترم‌ها را گرفته و مقدار متحول‌هایش را در خروجی‌اش مشخص کند. در ساخت این مدار ترکیبی از 3 گیت OR استفاده شده است.

جدول (۵-۹): جدول درستی اینکدر ۸ به ۳

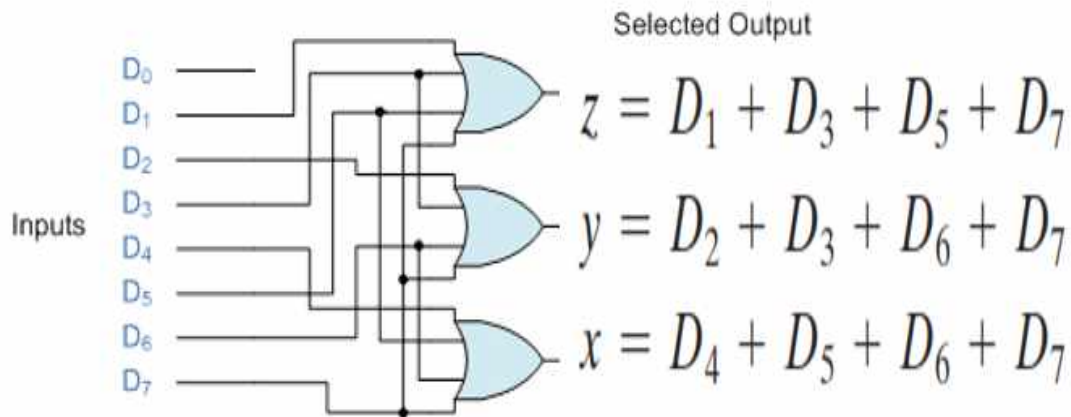
*Truth Table of an Octal-to-Binary Encoder*

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

$$X = D_4 + D_5 + D_6 + D_7$$

$$Y = D_2 + D_3 + D_6 + D_7$$

$$Z = D_1 + D_3 + D_5 + D_7$$



شکل (۵-۱۲): مدار ترکیبی اینکدر ۸ به ۳

## ۵.۸ مولتی پلکسر<sup>۱</sup>

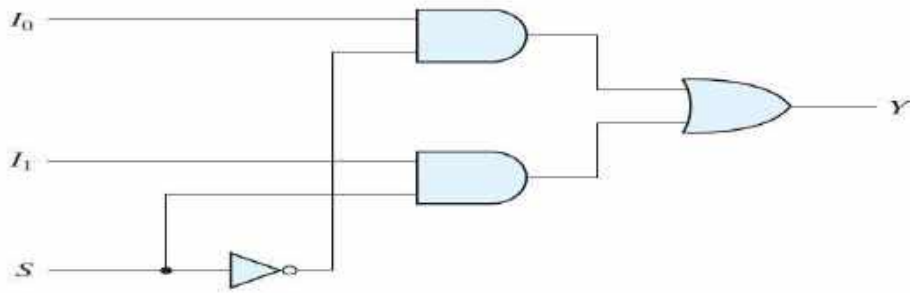
یک مولتی پلکسر مدار ترکیبی است که معلومات باینری را از تعدادی خط ورودی دریافت کرده و آن‌ها را به یک خط خروجی هدایت می‌نماید. انتخاب یک ورودی خاص به وسیله مجموعه‌یی از خطوط انتخاب انجام می‌شود. معمولاً  $2^n$  خط ورودی و  $n$  خط انتخاب وجود دارد و ترکیب بیتی تعیین کننده ورودی انتخاب شده است. مدار مولتی پلکسر، یکی از ورودی‌ها را انتخاب کرده و مقدار آن را در خروجی نشان می‌دهد.

مولتی پلکسر با نام MUX نیز نشان داده می‌شود.

### ۵.۸.۱ مولتی پلکسر ۲ به ۱

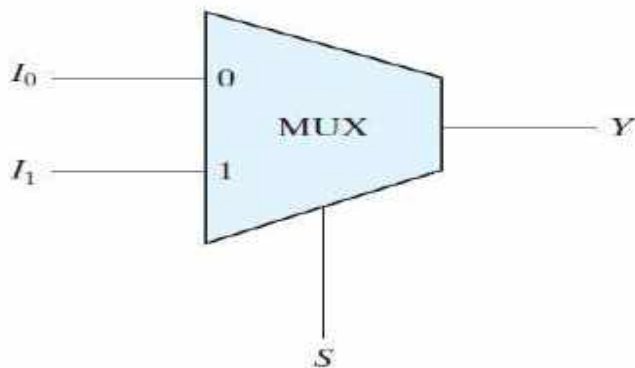
یک مولتی پلکسر ۲ به ۱، یکی از دو منبع ۱ بیت را طبق شکل به یک مقصد مشترک متصل می‌کند. مدار دارای دو خط ورودی دیتا، یک خروجی و یک خط انتخاب  $S$  است. وقتی  $S=0$  باشد، گیت AND بالایی فعال شده و  $I_0$  به خروجی راه می‌یابد. وقتی  $S=1$  باشد، گیت AND پایینی فعال شده و  $I_1$  به خروجی متصل می‌شود. مولتی پلکسر مثل یک کلید الکترونیک عمل کرده و یکی از دو منبع را انتخاب می‌نماید.

<sup>۱</sup>Multiplexer



شکل (۵-۱۳): مدار ترکیبی مولتی پلکسر ۲ به ۱

بلوک دیاگرام مولتی پلکسر ۲ به ۱



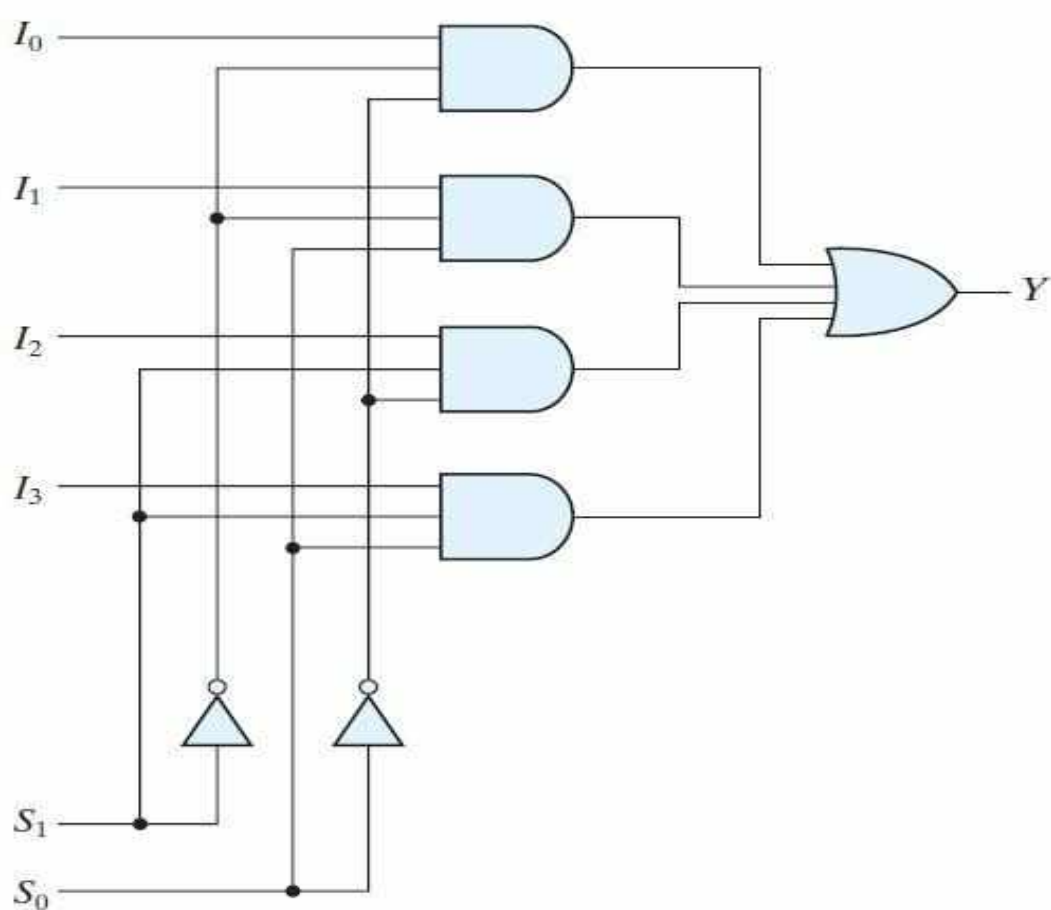
شکل (۵-۱۴): بلوک دیاگرام مولتی پلکسر ۲ به ۱

## ۵.۸.۲ مولتی پلکسر ۴ به ۱

یک مولتی پلکسر ۴ به ۱، در شکل زیر دیده می‌شود. هر یک از ۴ ورودی  $I_0$  تا  $I_3$  به یک ورودی گیت AND اعمال می‌شود. خطوط انتخاب  $S_0$  و  $S_1$  برای انتخاب گیت AND خاص دیکد می‌شوند. خروجی گیت‌های AND به یک گیت OR اعمال می‌شوند تا خروجی یک خط را ایجاد کنند. جدول، ورودی را که از مولتی پلکسر عبور کرده نشان می‌دهد. برای نمایش عمل مدار، حالتی را که  $S_1 S_0 = 10$  است ملاحظه کنید. گیت مربوط به ورودی  $I_2$  دارای دو ورودی ۱ و یک ورودی متصل به  $I_2$  است. سه گیت دیگر هر یک حداقل یک ۰ در ورودی خود دارند و بنابراین، خروجی‌شان ۰ می‌شود. خروجی گیت OR، اکنون برابر با  $I_2$  است و به این ترتیب مسیری از ورودی انتخابی به خروجی ایجاد شده است. یک مولتی پلکسر را انتخاب‌گر دیتا هم می‌خوانند، زیرا یکی از چند ورودی را انتخاب کرده و معلومات باینری را به خط خروجی هدایت می‌کند.

جدول (۵-۱۰): جدول انتخابگر مولتی پلکسر ۴ به ۱

$S_0$	$S_1$	$Y$
۰	۰	$I_0$
۰	۱	$I_1$
۱	۰	$I_2$
۱	۱	$I_3$

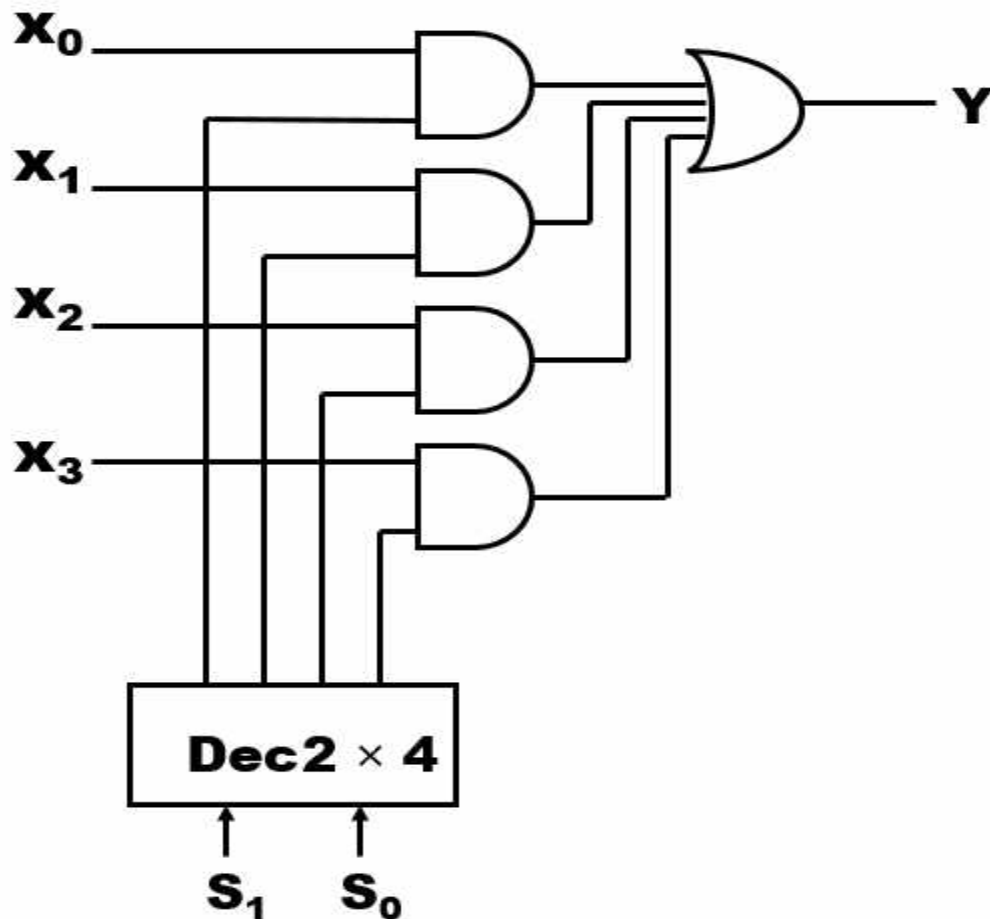


شکل (۵-۱۳) مدار ترکیبی مولتی پلکسر ۴ به ۱

### ۵.۸.۳ پیاده سازی مولتی پلکسر ۴ به ۱ توسط دیکدر ۲ به ۴

در این قسمت مولتی پلکسر ۴ به ۱، توسط دیکدر ۲ به ۴ ساخته شده است. با استفاده از دیکدر به کار رفته می توان، قسمت انتخاب گر یا Selector مربوط به مدار قبل را جایگزین کرد و مدار ساده تری ساخت.





شکل (۵-۱۴): مدار مولتی پلکسر ۴ به ۱ توسط دیکودر

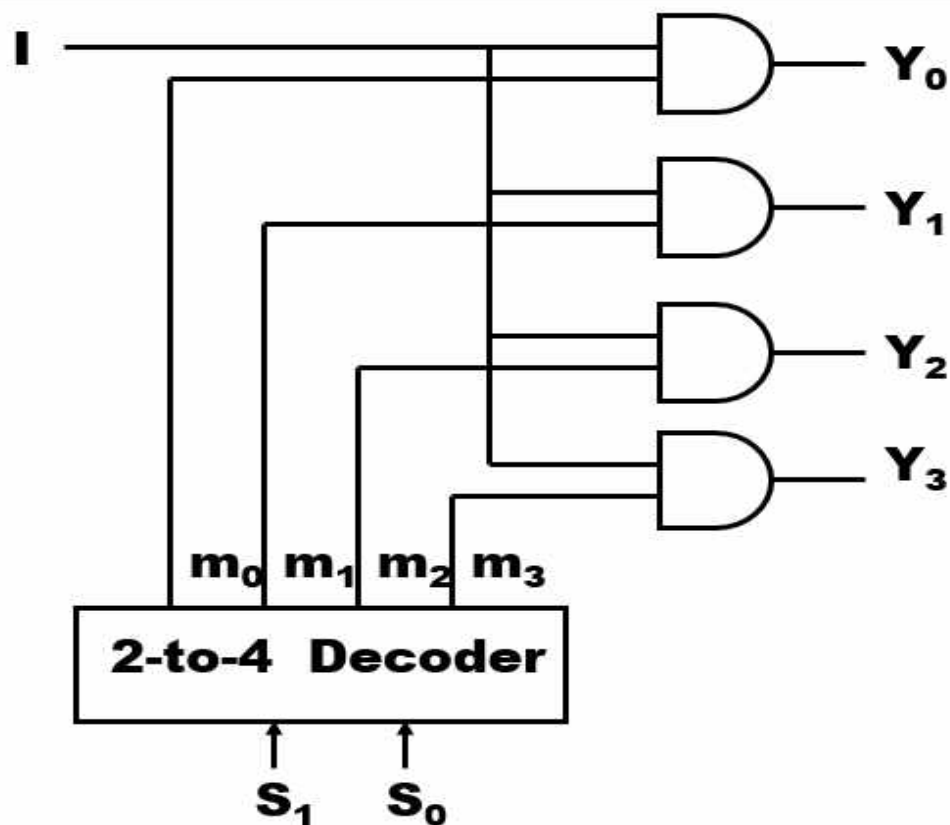
## ۵.۹ دی مولتی پلکسر<sup>۱</sup>

دی مولتی پلکسر عکس عمل مولتی پلکسر را انجام می‌دهد. هر دی مولتی پلکسر تعداد ۱ ورودی، تعداد  $2^n$  خروجی و  $n$  بیت انتخاب‌گر دارد. دی مولتی پلکسر، یکی از خروجی‌ها را توسط انتخاب‌گر انتخاب نموده و مقدار ورودی را در آن خروجی انتخاب شده، نمایش می‌دهد. دی مولتی پلکسر به DEMUX نیز معروف است.

### ۵.۹.۱ دی مولتی پلکسر ۱ به ۴

دی مولتی پلکسر ۱ به ۴ دارای یک خط ورودی، ۴ خط خروجی و ۲ بیت انتخاب‌گر می‌باشد. در شکل زیر مدار ترکیبی مربوط به دی مولتی پلکسر ۱ به ۴ نمایش داده شده است.

<sup>۱</sup> DeMultiplexer



شکل (۵-۱۵): مدار ترکیبی دی مولتی پلکسر ۱ به ۴

## ۵.۱۰ پیاده سازی مدار منطقی ترکیبی

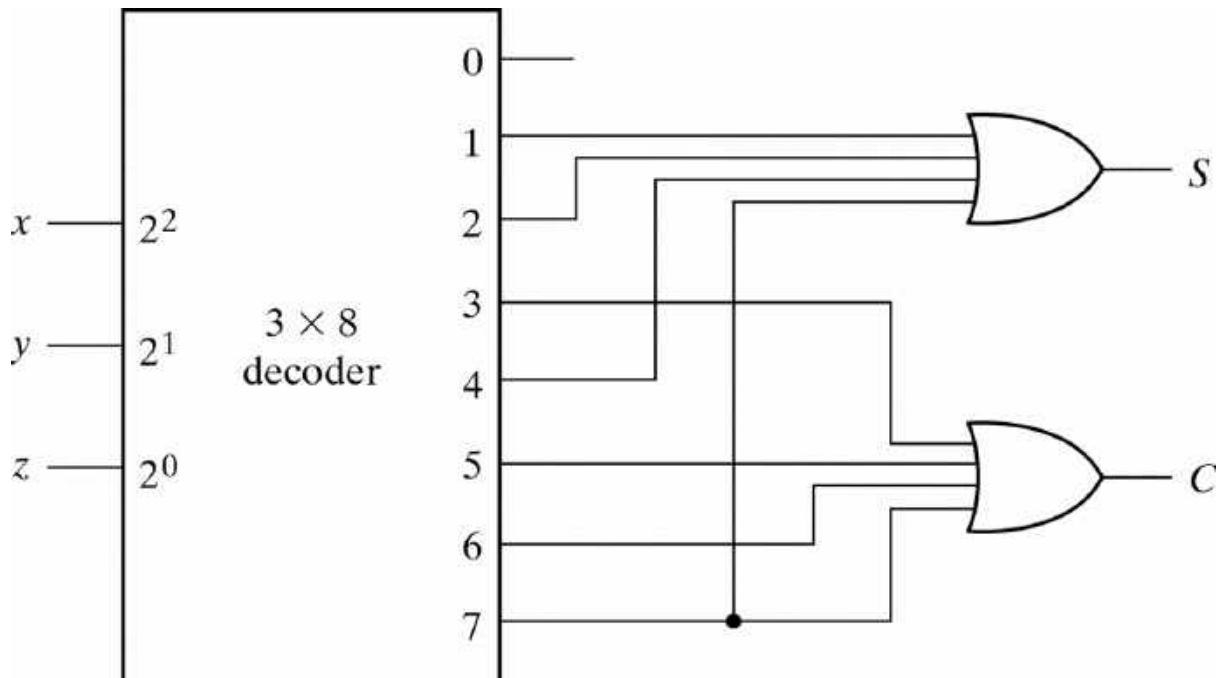
### ۵.۱۰.۱ پیاده سازی تمام جمع کننده توسط دیکدر ۳ به ۸

در این قسمت با استفاده از مدار دیکدر ۳ به ۸ به سادگی یک تمام جمع کننده (Full Adder) ساخته شده است. با استفاده از جدول درستی تمام جمع کننده و مینترمهای خروجی یک دیکدر این مدار ساخته می شود. در ساخت این مدار علاوه بر دیکدر ۳ به ۸ از دو گیت OR نیز استفاده شده است. یک دیکدر برای تولید همه مینترمهای حاصل از متحولهای ورودی انتخاب می شود. ورودی هر گیت OR از خروجیهای دیکدر بر حسب لیست مینترم هر تابع انتخاب می گردند. با توجه به جدول درستی تمام جمع کننده، توابع مدار ترکیبی را به صورت مجموع مینترمها بدست می آوریم:

$$S(x, y, z) = \sum(1, 2, 4, 7)$$

$$C(x, y, z) = \sum(3, 5, 6, 7)$$

چون ۳ ورودی و در مجموع ۸ مینترم وجود دارد، به یک دیکدر ۳ به ۸ خط احتیاج است. دیکدر هشت مینترم را برای  $y, x$  و  $z$  تولید می‌کند. گیت OR برای خروجی  $S$ ، جمع منطقی مینترم‌های ۱، ۲، ۴ و ۷ را تشکیل می‌دهد. گیت OR جمع منطقی مینترم‌های ۳، ۵، ۶ و ۷ را برای تولید خروجی  $C$  به کار می‌برد.



شکل (۵-۱۶): پیاده سازی تمام جمع کننده توسط دیکودر



مدارهای ترکیبی، مدارهای هستند که خروجی‌های آن‌ها در یک زمان به ورودی‌هایشان در همان لحظه بستگی دارد. مدارهای ترکیبی، مدارهای هستند که به‌منظور پروسس و محاسبه به کار می‌روند. مدار یک نیم‌جمع کننده حاصل جمع دو بیت باینری را باهم حساب می‌کند. مدارهای تمام جمع کننده حاصل جمع باینری ۳ بیت را محاسبه می‌کند. مدارهای، مانند جمع کننده چند بیتی، حاصل جمع دو عدد باینری را محاسبه می‌کند. مدار جمع کننده و تفریق کننده، علاوه بر محاسبه حاصل جمع دو عدد حاصل تفریق دو عدد را نیز محاسبه می‌کند. مدارهای دیکدر و اینکدر برای رمزنگاری و رمزگشایی دیتا استفاده می‌شوند. مدارهای ترکیبی مولتی پلکسر، یکی از ورودی‌ها را انتخاب و مقدارش را در خروجی نشان می‌دهد. مدار دی مولتی پلکسر، یکی از خروجی‌ها را انتخاب و مقدار ورودی را در آن خروجی فقط نمایش می‌دهد. مدارهای ترکیبی انواع مختلفی دارند که در این فصل فقط با تعدادی از مهم‌ترین این مدارهای آشنا شدید. در فصل بعد با انواع دیگری از مدارها به نام مدارهای ترتیبی آشنا خواهید شد.



## سوالات و فعالیت های فصل پنجم

۱. مدار یک نیم جمع کننده را رسم کنید؟
۲. مدار تمام جمع کننده را رسم کنید؟
۳. مدار و جدول مربوط به دیکدر ۳ به ۸ را رسم کنید؟
۴. مدار مربوط به اینکدر ۴ به ۲ را رسم کنید؟
۵. مدار مربوط به مولتی پلکسر ۴ به ۱ را رسم کنید؟
۶. مدار مربوط به دی مولتی پلکسر ۴ به ۱ را رسم کنید؟

### فعالیت ها

۱. مدار یک نیم جمع کننده را در نرم افزارهای شبیه ساز، مانند LogiSim رسم کنید.
۲. مدار تمام جمع کننده را در نرم افزارهای شبیه ساز، مانند LogiSim رسم کنید.
۳. مدار و جدول مربوط به دیکدر ۳ به ۸ را در نرم افزارهای شبیه ساز، مانند LogiSim رسم کنید.
۴. مدار مربوط به اینکدر ۴ به ۲ را در نرم افزارهای شبیه ساز، مانند LogiSim رسم کنید.
۵. مدار مربوط به مولتی پلکسر ۴ به ۱ را در نرم افزارهای شبیه ساز، مانند LogiSim رسم کنید.
۶. مدار مربوط به دی مولتی پلکسر ۴ به ۱ را در نرم افزارهای شبیه ساز، مانند LogiSim رسم کنید.

## فصل ششم

### مدارهای ترتیبی



هدف کلی: آشنایی محصلان با مدارهایی ترتیبی.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. مدارهای ترتیبی را تعریف نمایند.
۲. روش طراحی مدارهای ترتیبی را توضیح دهند.
۳. مدارهای ترتیبی را با کاربردهای ویژه آن تعریف کند.

در فصل‌های قبلی با گیت‌های منطقی و مدارهای منطقی ترکیبی آشنا شدید. مدارهای ترکیبی، برای محاسبات منطقی استفاده می‌شوند. در این فصل، به نحوه استفاده از گیت‌های منطقی در ساخت عناصر حافظه می‌پردازیم. یکی از اساسی‌ترین کارهایی که در یک کامپیوتر انجام می‌شود، ذخیره سازی اطلاعات در حافظه کامپیوترها می‌باشد. از طرف دیگر، در هنگام محاسبات پیچیده‌یی که در کامپیوتر توسط سی پی یو انجام می‌گیرد، نیاز به ذخیره اطلاعات در حال پروسس می‌باشد. لذا ذخیره سازی و عناصر حافظه، نقش اساسی در ساخت سیستم‌های دیجیتال پیدا می‌کنند. عناصر حافظه در عنوان‌های مختلف و با کاربردهای فراوانی وجود دارند. در این قسمت، عناصر پایه حافظه مورد بررسی قرار گرفتند. عناصر حافظه جز دسته دیگری از مدارهای دیجیتال منطقی به نام مدارهای ترتیبی می‌باشند. مدارهای ترکیبی، مدارهایی بودند که خروجی‌های‌شان در یک زمان تنها به ورودی‌های‌شان در همان زمان بستگی داشتند. اما مدارهای ترتیبی، مدارهایی هستند که خروجی‌هایشان، علاوه بر ورودی فعلی به ورودی‌های قبلی‌شان نیز بستگی دارند که این امر سبب ایجاد خصوصیت ذخیره سازی در مدارهای ترتیبی می‌شود. در این فصل، به بررسی دقیق مدارهای ترتیبی، انواع و کاربرد آن می‌پردازیم.

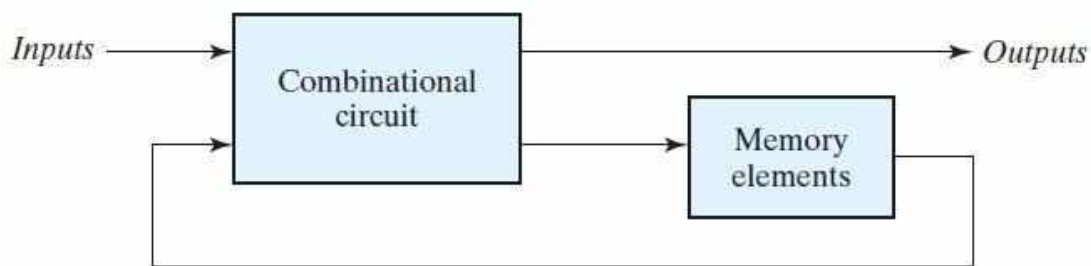
## ۶.۱ مدارات ترتیبی<sup>۱</sup>

مدارهای دیجیتالی که تاکنون بررسی شدند، از نوع ترکیبی بودند. در این مدارها، خروجی‌ها همه به ورودی‌های جاری وابسته‌اند. گرچه به نظر می‌رسد که هر سیستم دیجیتال دارای مدارهای ترکیبی است، بسیاری از سیستم‌هایی که در عمل با آن مواجه هستیم، حاوی عناصر حافظه هم می‌باشند و بنابراین، لازم است تا این سیستم‌ها برحسب منطق ترتیبی مورد بررسی قرار گیرند.

نمودار بلوکی یک مدار ترتیبی در شکل (۶-۱) نشان داده شده است. این مدار متشکل از مداری ترکیبی است که عناصر حافظه برای ایجاد یک مسیر پسخورد به آن وصل شده‌اند. عناصر حافظه قطعاتی هستند که می‌توانند، معلومات باینری را ذخیره کنند. معلومات باینری ذخیره شده در این عناصر در هر لحظه از زمان حالت مدار ترتیبی در آن زمان است. مدار ترتیبی معلومات باینری را از ورودی‌های بیرونی دریافت می‌کند. این ورودی‌ها همراه با حالت فعلی عناصر حافظه، مقدار باینری خروجی‌ها را معین می‌نماید. آن‌ها شرط تغییر حالت در عناصر حافظه را نیز معین می‌سازند. نمودار بلوکی نشان می‌دهد که خروجی‌های یک مدار ترتیبی نه فقط تابعی از ورودی‌ها هستند، بلکه به حالت فعلی عناصر حافظه نیز وابسته می‌باشند. حالت بعدی عناصر حافظه نیز تابعی از ورودی‌های بیرونی و حالت فعلی است. بنابراین، یک مدار ترتیبی با ترتیب زمانی ورودی‌ها، خروجی‌ها و حالات داخلی مشخص می‌شود.

---

<sup>۱</sup> Sequential Logic



شکل (۶-۱): مدارات ترتیبی

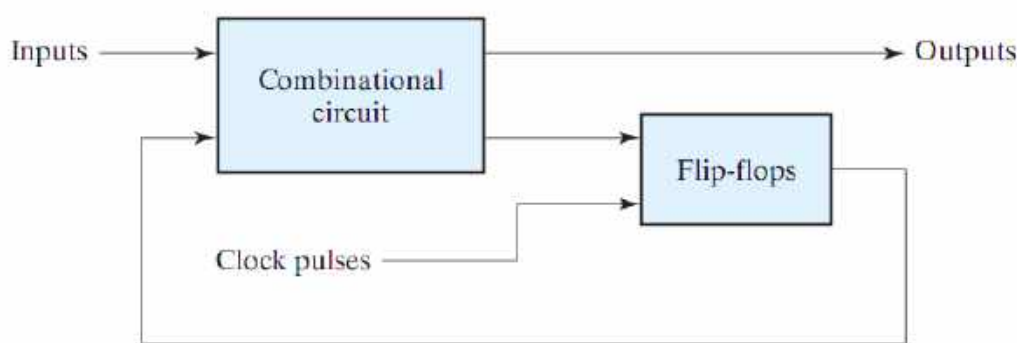
دو نوع مدار ترتیبی وجود دارد که دسته‌بندی آن‌ها به زمان‌بندی سیگنال آن‌ها وابسته است. مدار ترتیبی هم‌زمان، همگام سیستمی است که رفتار آن با توجه به دانش و آگاهی از سیگنال‌هایش در هر لحظه گسسته‌یی از زمان قابل تعریف می‌باشد. رفتار یک مدار ترتیبی غیر هم‌زمان به ترتیب تغییر سیگنال‌های ورودی آن که می‌توانند در هر لحظه‌یی از زمان روی مدار تأثیر کنند وابسته باشد. عناصر حافظه‌یی که به‌طور معمول در مدارهای ترتیبی غیرهم‌زمان به کار می‌روند، نوعی وسایل تأخیر زمانی هستند. قابلیت نگهداری یک وسیله تأخیر زمانی به زمان انتشار سیگنال در وسیله بستگی دارد. در عمل تأخیر، انتشار در گیت‌های منطقی درونی برای ایجاد تأخیر کفایت می‌کند، بنابراین، واحد تأخیر واقعی می‌تواند، مورد نیاز نباشد. در سیستم‌های غیرهم‌زمان نوع گیت، عناصر حافظه متشکل از گیت‌های منطقی است که در واقع تأخیر انتشار آن‌ها عمل ذخیره‌سازی را تداعی می‌نماید. بنابراین، در چنین مواقعی یک مدار ترتیبی غیرهم‌زمان را می‌توان مداری ترکیبی با پسخورد دانست. به دلیل وجود پسخورد در بین گیت‌های منطقی، هر مدار ترتیبی غیرهم‌زمان هر لحظه ممکن است، ناپایدار شود. مسئله بی‌رجیستری حاکم، مشکلات زیادی را بر طراح تحمیل خواهد کرد.

با توجه به تعریف، یک مدار ترتیبی هم‌زمان سیگنال‌هایی را مورد استفاده قرار می‌دهد که فقط در لحظات گسسته‌یی از زمان روی عناصر حافظه‌اش اثر می‌گذارد. در این مدارها هم‌زمانی با وسیله‌یی به نام مولد ساعت تحقق می‌یابد و طی آن رشته متناوبی از پالس ساعت به وسیله این دستگاه تولید می‌شود. پالس‌های ساعت در سرتاسر سیستم توزیع می‌شوند، به نحوی که عناصر حافظه، تنها هنگام رسیدن هر پالس تحت تأثیر ورودی خود قرار می‌گیرند. در عمل پالس‌های ساعت به همراه دیگر پالس‌ها که تغییرات لازم را در حافظه ایجاد می‌کنند، همراه هستند. مدارهای ترتیبی هم‌زمانی که پالس‌های ساعت را در ورودی عناصر ذخیره‌ساز خود به کار می‌برند، مدارهای ترتیبی ساعت دار خوانده می‌شوند. ما غالباً در عمل با مدارهای ترتیبی ساعت دار مواجه هستیم. آن‌ها مشکل ناپایداری را ندارند و موضوع زمان‌بندی در آن‌ها به راحتی به مراحل گسسته و مستقل شکسته می‌شود. هر یک از این مراحل یا برش‌های زمانی مستقلاً قابل بررسی می‌باشند.

عناصر ذخیره‌سازی در مدارهای ترتیبی ساعت دار را فلیپ فلاپ می‌گویند. فلیپ فلاپ یک وسیله ذخیره‌سازی باینری بوده و قادر است، یک بیت از معلومات را در خود ذخیره نماید. یک مدار ترتیبی ممکن



است در صورت لزوم تعداد قابل توجهی از این فلیپ فلاپ‌ها را به کار ببرد. نمودار بلوکی یک مدار ترتیبی ساعت دار هم‌زمان در شکل (۶-۲) دیده می‌شود. خروجی‌ها می‌توانند، از یک مدار ترکیبی، یا از فلیپ فلاپ‌ها و یا هر دو حاصل شوند. فلیپ فلاپ‌ها، ورودی‌های خود را از مدار ترکیبی و نیز از سیگنال ساعت که با فواصل زمانی رخ می‌دهند، طبق نمودار زمانی دریافت می‌کنند. حالت فلیپ فلاپ‌ها تنها هنگام تغییر وضعیت یک پالس ساعت عوض می‌شود. وقتی یک پالس ساعت فعال نیست، حلقهٔ پسخورد قطع می‌شود، زیرا حتی اگر خروجی‌های مدار ترکیبی - که ورودی آن‌ها را تغذیه می‌کند- عوض شود، خروجی‌های فلیپ فلاپ تغییر نمی‌نمایند. بنابراین، تغییر وضعیت از یک حالت به بعدی فقط در فواصل زمانی دیکته شده به وسیلهٔ پالس‌های ساعت امکان‌پذیر است.



(a) Block diagram



(b) Timing diagram of clock pulses

شکل (۶-۲): کارکرد مدارات ترتیبی توسط کلاک پالس

## ۶.۲ لچ<sup>۱</sup>

یک فلیپ فلاپ می‌تواند، یک حالت باینری را هنگامی که تغذیه به مدارش اعمال شود، تا مدتی نامحدود نگاه دارد. تفاوت عمده، بین انواع فلیپ فلاپ‌ها، در تعداد ورودی‌ها و نحوهٔ تأثیر آن‌ها در تغییر حالت باینری است. ساده‌ترین انواع فلیپ فلاپ‌ها که با سطوح سیگنال عمل می‌کنند، لچ نامیده می‌شوند. لچ‌ها مدارهای پایه هستند که همه فلیپ فلاپ‌ها با آن‌ها ساخته می‌شوند. لچ‌ها حساس به سطح هستند.

اگرچه لچ‌ها برای ذخیره سازی معلومات باینری و طراحی مدارهای ترتیبی غیر هم‌زمان مفید اند، ولی عملاً در مدارهای ترتیبی هم‌زمان به کار نمی‌روند، انواع فلیپ فلاپ‌هایی که در مدارهای ترتیبی مورد استفاده قرار می‌گیرند، در ادامه فصل معرفی می‌شوند.

<sup>۱</sup> Latch

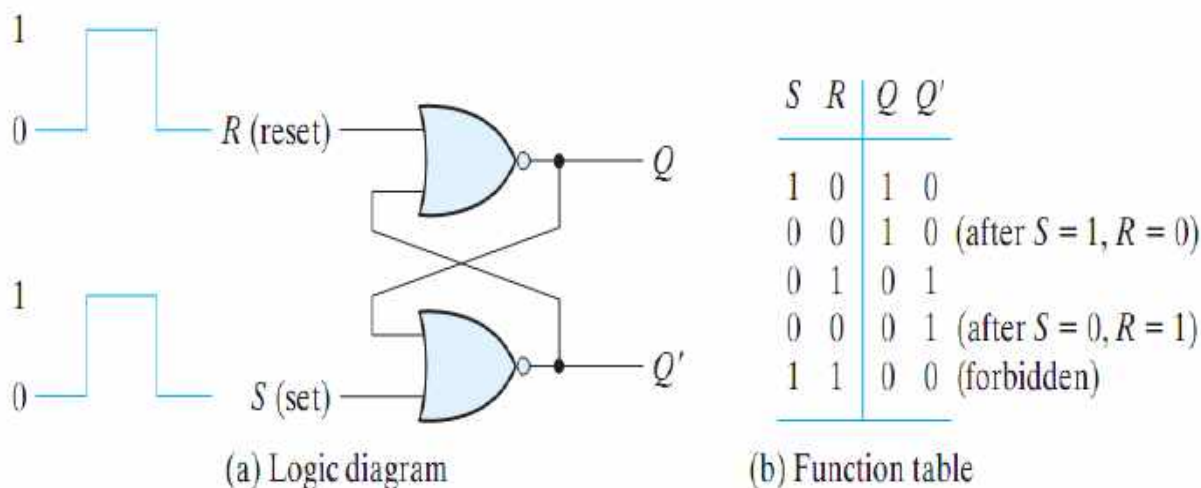
## ۶.۲.۱ لچ SR

### ۶.۲.۱.۱ لچ SR توسط گیت NOR

این نوع از لچ SR از دو گیت منطقی NOR تشکیل شده که به طور متقاطع به هم وصل شده‌اند. این مدار دو ورودی دارد که با S به معنی SET و R به معنی RESET نام‌گذاری شده‌اند. لچ SR ساخته شده از دو گیت NOR در شکل زیر دیده می‌شود. لچ، دارای دو حالت مفید است. وقتی خروجی  $Q = 1$  و

$Q' = 0$  باشند، لچ در حالت Set است. اگر  $Q = 0$  و  $Q' = 1$  باشند، لچ در حالت Reset است. خروجی‌های  $Q$  و  $Q'$  معکوس یکدیگر هستند. با این وجود، وقتی هر دو ورودی به طور هم‌زمان ۱ شوند، حالت تعریف نشده ۰ برای دو خروجی رخ می‌دهد.

تحت شرایط معمولی، هر دو ورودی در ۰ نگه داری می‌شوند، مگر این که بخواهیم حالت لچ را تبدیل کنیم. یعنی، زمانی که هر دو ورودی برابر با ۰ باشند، لچ حالت سابق خود را حفظ می‌کند. اگر در حالت Set باشد با ورودی‌های ۰ و ۰ در حالت Set باقی می‌ماند و اگر در حالت Reset باشد، با ورودی‌های ۰ و ۰ در حالت Reset باقی می‌ماند.



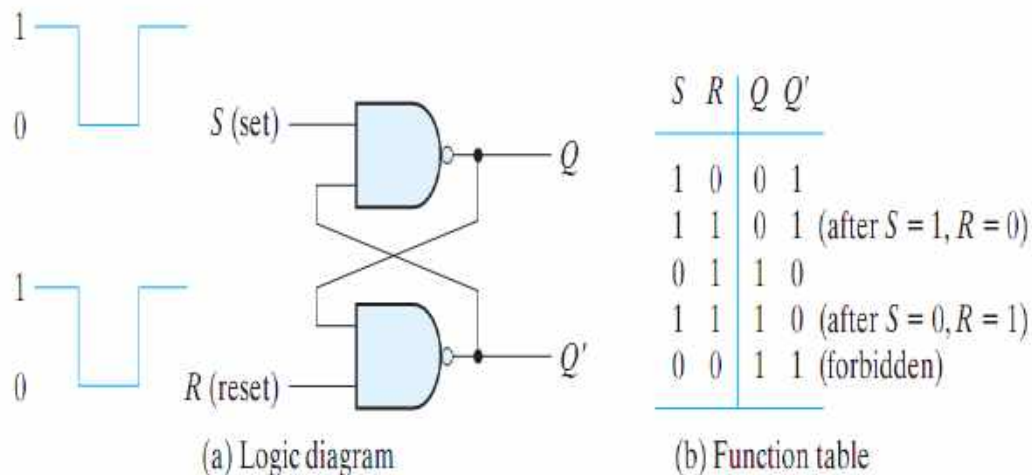
شکل (۶-۳): مدار ترتیبی لچ SR توسط گیت NOR

### ۶.۲.۱.۲ لچ SR توسط گیت NAND

لچ SR با دو گیت NAND در شکل زیر مشاهده می‌شود. این مدار به طور معمول با ۱ در هر دو ورودی‌اش کار می‌کند. مگر این که بخواهیم حالت لچ را تغییر دهیم. اعمال ۰ به S موجب می‌شود Q به ۱ برود و لچ به حالت Set. وقتی که ورودی S به ۱ بازگردد، مدار در همان حالت ۱ باقی می‌ماند. پس از بازگشت هر دو

ورودی به ۱، ما مجاز به تغییر حالت لچ با ورود مقدار ۰ به ورودی R هستیم. این موجب می شود تا مدار به حالت Reset برود و حتی پس از بازگشت هر دو ورودی به ۱، لچ در همان حالت باقی بماند.

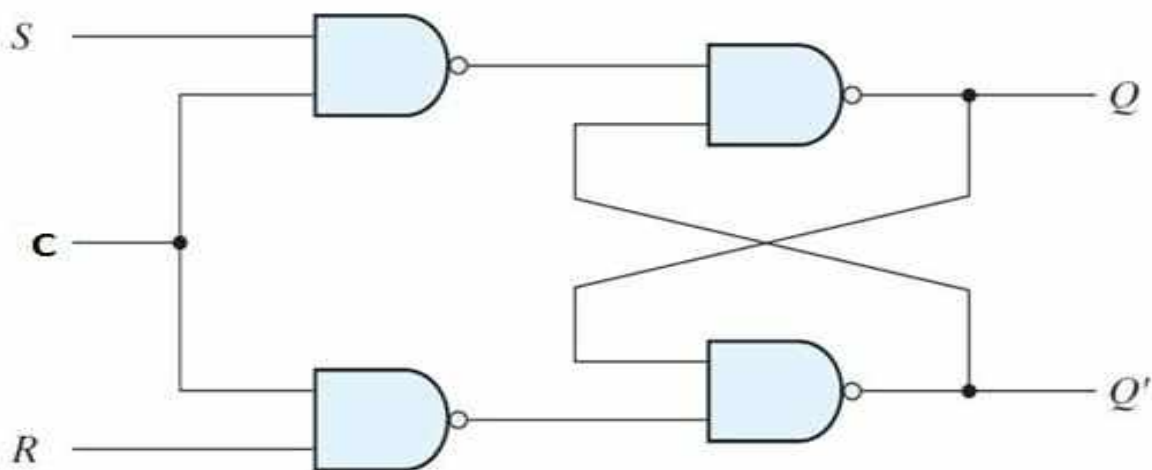
حالتی که برای لچ SR با گیت NAND غیر مجاز است، هنگامی است که هر دو ورودی به طور همزمان ۰ باشند. بنابراین، باید از وقوع این حالت جلوگیری کرد.



شکل (۴-۶): مدار ترتیبی لچ SR توسط گیت NAND

## ۶.۲.۲ لچ SR با ورودی کنترل

عملکرد لچ SR با افزودن یک ورودی کنترل برای تعیین زمان تغییر حالت لچ اصلاح می شود. یک لچ کنترل دار در شکل زیر مشاهده می شود. این مدار شامل یک لچ SR پایه و دو گیت NAND اضافه است. ورودی کنترل C به عنوان یک سیگنال فعال ساز برای دو ورودی عمل می کند. هنگامی که ورودی کنترل در ۰ باقی بماند، خروجی گیت های NAND در سطح منطقی ۱ باقی می ماند. این وضعیت حالت بدون تغییر یا No Change برای لچ SR می باشد. حالت Set با  $S=1$ ،  $R=0$  و  $C=1$  حاصل می شود. برای تغییر وضعیت به حالت Reset باید  $S=0$ ،  $R=1$  و  $C=1$  باشد. در هر حال، وقتی C به ۰ بازگردد، مدار در حالت فعلی خود بدون تغییر باقی می ماند و عدم تغییر حالت مستقل از مقادیر S و R می باشد. حالت غیر مجاز یا تعریف نشده زمانی رخ می دهد که هر ۳ ورودی برابر با ۱ باشند.



شکل (۵-۶): مدار ترتیبی، لچ SR با ورودی کنترل

جدول (۱-۶): جدول عملکرد لچ SR با ورودی کنترل

$En$	$S$	$R$	Next state of $Q$
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$ ; reset state
1	1	0	$Q = 1$ ; set state
1	1	1	Indeterminate

### ۶.۲.۳ لچ D

یکی از راه‌های حذف حالت غیر مجاز یا تعریف نشده در لچ SR این است که مطمئن شویم S و R هرگز به‌طور هم‌زمان به مقدار ۱ نمی‌روند. این کار با لچ D میسر است.

این لچ تنها دو ورودی دارد: ۱-C (کنترل)؛ ۲-D (دیتا).

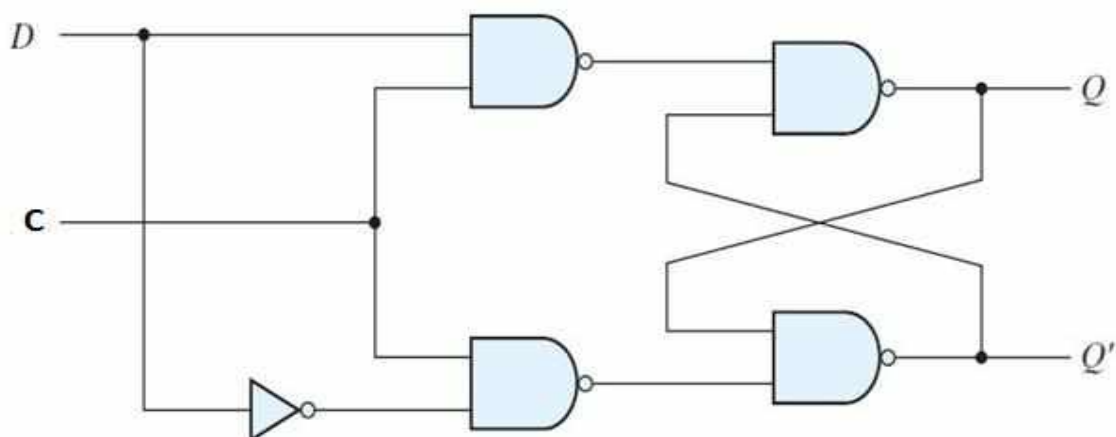
ورودی D مستقیماً به ورودی S و معکوس آن به ورودی R وصل می‌شود. هنگامی که ورودی کنترل در ۰ قرار دارد، لچ SR متقاطع دارای ۱ در هر دو ورودی بوده و مدار نمی‌تواند، تغییر حالت دهد. در واقع مقدار D هم نقشی ندارد. وقتی  $C=1$  باشد، ورودی D نمونه برداری می‌شود. اگر  $D=1$  باشد، خروجی Q به ۱ می‌رود، به این ترتیب مدار در حالت Set است. اگر  $D=0$ ، خروجی Q به ۰ رفته و مدار به حالت Reset می‌رود.

لچ D نامش را از قابلیت نگهداری دیتا دریافت کرده است. این لچ، برای ذخیره موقت معلومات باینری بین یک محیط و یک واحد مناسب است. معلومات باینری حاضر در ورودی دیتای لچ D هنگامی که ورودی فعال

شود، به خروجی  $Q$  منتقل می‌شود. هنگامی که ورودی کنترل فعال است، خروجی تغییرات ورودی را دنبال می‌کند. این وضعیت مسیری از  $D$  به خروجی ایجاد می‌کند، به این دلیل لچ  $D$  را لچ شفاف<sup>۱</sup> هم می‌خوانند. وقتی ورودی کنترل غیرفعال شود، معلومات باینری حاضر قبلی در ورودی، در خروجی  $Q$  باقی می‌ماند تا دوباره ورودی کنترل فعال گردد.

جدول (۶-۲): جدول عملکرد لچ  $D$

$En$	$D$	Next state of $Q$
0	X	No change
1	0	$Q = 0$ ; reset state
1	1	$Q = 1$ ; set state

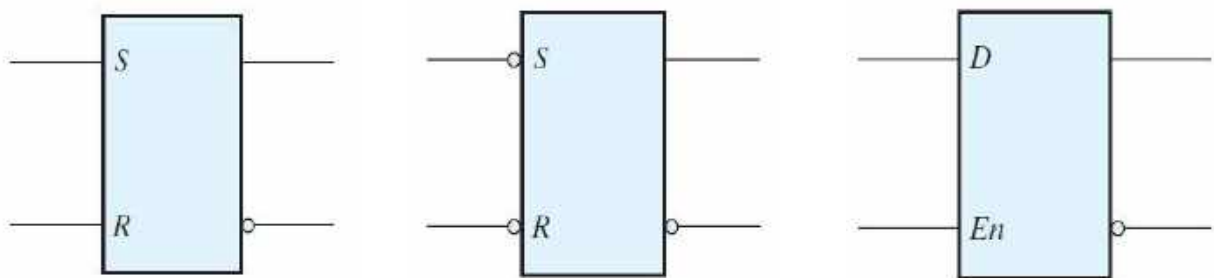


شکل (۶-۶): مدار ترتیبی لچ  $D$

#### ۶.۲.۴ سمبول‌های لچ‌ها

سمبول گرافیکی برای انواع لچ در شکل زیر آمده است. لچ با یک بلوک مستطیلی مشخص می‌شود که در آن ورودی‌ها در سمت چپ و خروجی‌ها در سمت راست نشان داده می‌شوند. یکی از خروجی‌ها  $Q$  و دیگری معکوس خروجی  $Q'$  را نشان می‌دهد.

<sup>۱</sup> Transparent Latch



SR NOR GATE LATCH SR NAND GATE LATCH D LATCH

شکل (۶-۷): سمبول لچ‌ها

### ۶.۳ فلیپ فلاپ<sup>۱</sup>

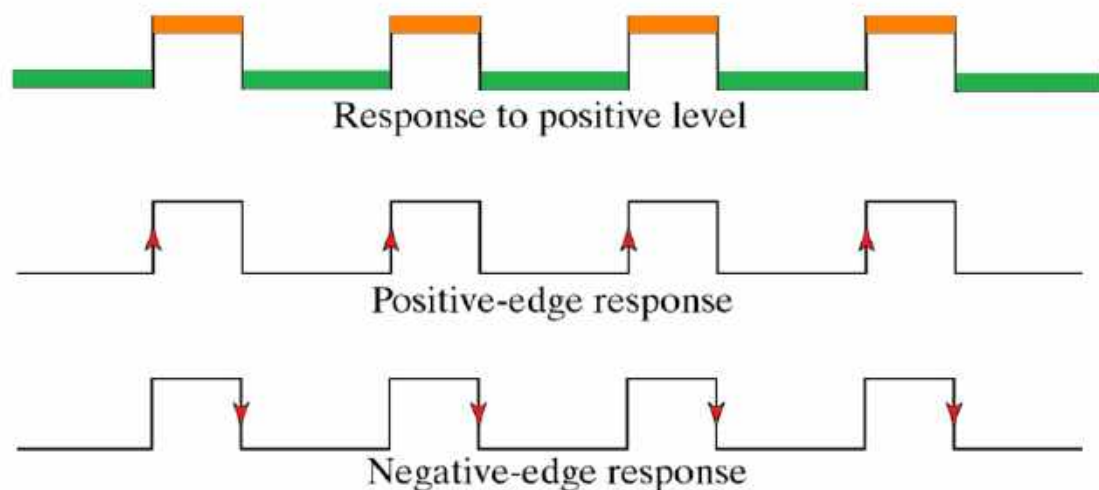
حالت یک لچ یا یک فلیپ فلاپ با تغییر در ورودی کنترل عوض می‌شود. این تغییر لحظه‌یی را تریگر گویند و انتقال مربوط به آن را تریگر کردن فلیپ فلاپ خوانند. لچ D با پالس‌ها در ورودی کنترلش اساساً یک فلیپ فلاپ است که هر زمان پالس به سطح منطقی 1 برود تریگر می‌شود. هنگامی که پالس ورودی کنترل در این سطح بماند هر تغییری در ورودی دیتا، خروجی و حالت لچ را عوض خواهد کرد.

یک مدار ترتیبی، از خروجی فلیپ فلاپ به ورودی‌های مدار ترکیبی دارای مسیر پس‌خورد است. در نتیجه ورودی‌های فلیپ فلاپ ممکن است از خروجی همان و یا دیگر فلیپ فلاپ‌ها راه اندازی شوند. وقتی که از لچ‌ها به عنوان عناصر مورد استفاده قرار گیرند، مشکل اساسی به وجود می‌آید. به محض تغییر پالس ساعت به منطق 1، انتقال حالت لچ‌ها آغاز می‌شود. در حالی که پالس ساعت هنوز فعال است، حالت جدید لچ در خروجی ظاهر می‌گردد. این خروجی به ورودی لچ‌ها از طریق مدار ترکیبی وصل می‌شود. اگر پالس ساعت در منطق 1، باشد و ورودی اعمال شده به لچ‌ها تغییر کند، لچ به مقادیر جدید واکنش نشان داده و خروجی جدیدی رخ خواهد داد. نتیجه این واکنش وضعیت پیش بینی نشده‌یی است؛ زیرا حالت لچ‌ها ممکن است با قرار داشتن پالس ساعت در سطح فعال هم‌چنان به تغییر خود ادامه دهد. به دلیل این عملکرد غیر مطلوب، خروجی یک لچ وقتی همه لچ‌ها به منبع ساعت مشترکی وصل‌اند نمی‌تواند، مستقیماً و یا از طریق یک مدار منطقی به همان لچ یا دیگر لچ‌ها وصل شود.

فلیپ فلاپ‌ها طوری ساخته می‌شوند که وقتی بخشی از نوع مدار ترتیبی‌اند و از ساعت مشترکی استفاده می‌کنند، عملکردشان صحیح باشد. مشکل لچ این است که به سطح پالس ساعت پاسخ می‌دهد. همان‌طور که در شکل زیر مشاهده می‌شود، وقتی که پالس ساعت در منطق 1 قرار دارد، هر تغییر مثبت در ورودی کنترل موجب می‌شود تا به ازای هر تغییر در ورودی D تغییری در خروجی به وجود آید. نکته کلیدی در یک عملکرد صحیح فلیپ فلاپ‌ها، تریگر شدن آن‌ها در زمان گذر سیگنال است. پالس ساعت از دو انتقال 1 به

<sup>۱</sup> Flip flop

۰ و ۰ به ۱ گذر می‌کند. طبق شکل گذر مثبت به عنوان لبه مثبت و گذر منفی به عنوان لبه منفی شناخته می‌شود. برای اصلاح یک لچ به یک فلیپ فلاپ، دو راه وجود دارد:



شکل (۶-۸): عملکرد مدارها متناسب با کلاک پالس

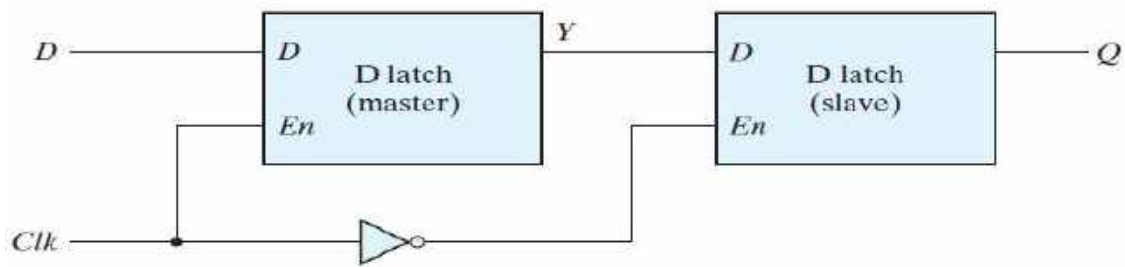
### ۶.۳.۱ فلیپ فلاپ D

#### ۶.۳.۱.۱ فلیپ فلاپ D حساس به لبه منفی

ساختن فلیپ فلاپ D با دو لچ D و یک گیت NOT در شکل زیر ملاحظه می‌شود. اولین لچ را حاکم و دومی را تابع می‌گویند. مدار، ورودی D را نمونه برداری کرده و خروجی Q را فقط در لبه منفی پالس کنترل ساعت (CLK) تغییر می‌دهد. وقتی که پالس ساعت در ۰ است، خروجی وارونگر ۱ می‌باشد. لچ تابع فعال شده و خروجی آن، Q، برابر با خروجی حاکم یعنی Y خواهد شد. لچ حاکم غیر فعال است، زیرا  $CLK=0$  می‌باشد. وقتی که پالس ساعت ورودی به سطح ۱ تغییر وضعیت می‌دهد، دیتا از ورودی بیرونی D به حاکم منتقل می‌گردد. در این حال، مادامی که ساعت در سطح ۱ بماند، تابع غیرفعال خواهد بود زیرا ورودی C آن برابر ۰ است. هر تغییر در ورودی، خروجی Y را عوض می‌کند، ولی نمی‌تواند، خروجی تابع را عوض کند. وقتی که پالس ساعت به ۰ بازگردد، حاکم غیرفعال شده و از ورودی D جدا می‌شود. در همان زمان تابع فعال شده و مقدار Y به خروجی فلیپ فلاپ در Q انتقال می‌یابد. بنابراین، خروجی فلیپ فلاپ فقط در حین گذر پالس ساعت از ۱ به ۰ تغییر می‌کند.

رفتار فلیپ فلاپ حاکم تابع که در بالا توصیف شد، نشان می‌دهد که خروجی فقط در لبه منفی پالس ساعت تغییر می‌نماید.

به این فلیپ فلاپ (Master And slave) نیز می گویند.



شکل (۶-۹): فلیپ فلاپ D حساس به لبه منفی

جدول (۶-۳): جدول عملکرد فلیپ فلاپ D

### D Flip-Flop

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

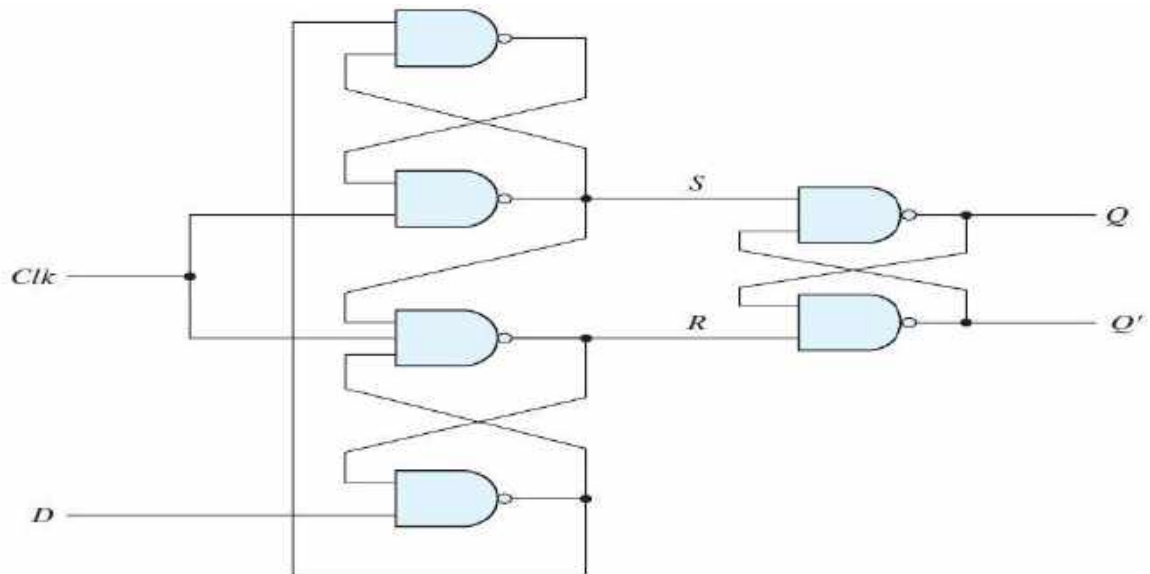
#### ۶.۳.۱.۲ فلیپ فلاپ D حساس به لبه مثبت

نمونه دیگری از فلیپ فلاپ D حساس به لبه، از ۳ لچ SR، مطابق شکل زیر استفاده می کند. دو لچ موجود در این شکل به ورودی های بیرونی D (دیتا) و CLK (ساعت) پاسخ می دهند. لچ سوم، خروجی را برای فلیپ فلاپ تهیه می کند. ورودی های S و R لچ خروجی در  $CLK=0$  در سطح منطق ۱ نگه داری می شوند. این موجب می شود تا خروجی در حالت فعلی خود باقی بماند. ورودی D ممکن است، برابر ۰ یا ۱ باشد. اگر هنگام ۱ شدن CLK،  $D=0$ ، برقرار باشد، R به ۰ تغییر می کند. یعنی فلیپ فلاپ به حالت Reset رفته و در آن  $Q=0$  می گردد. اگر در زمان  $CLK=1$  تغییری در ورودی رخ دهد، پایانه R در ۰ می ماند. بنابراین، فلیپ فلاپ بر علاوه تغییر در ورودی خود، به حالت قفل باقی خواهد ماند. وقتی که ساعت به ۰ بازگردد، R به ۱ می رود و لچ خروجی در وضعیت ساکن و بدون تغییر در خروجی باقی می ماند. به طور مشابه وقتی CLK از ۰ به ۱ می رود، اگر  $D=1$  باشد، S به ۰ تغییر می کند. این موجب می شود تا مدار به حالت ۱ رفته و  $Q=1$  گردد. در هر تغییر در D، هنگامی که  $CLK=1$  است، نمی تواند، خرجی را تحت تأثیر قرار دهد.

به طور خلاصه، وقتی ساعت ورودی در فلیپ فلاپ حساس به لبه مثبت یک انتقال مثبت انجام دهد، مقدار D به Q منتقل می شود. یک لبه منفی از ۱ به ۰ تاثیری بر روی خروجی ندارد. به همین ترتیب سطح منطقی



۱، و نیز سطح منطقی ۰ هم خروجی را تبدیل نمی‌کنند. از این رو این نوع فلیپ فلاپ تنها به لبه ۰ به ۱ پاسخ می‌دهد.



شکل (۶-۱۰): فلیپ فلاپ D حساس به لبه مثبت

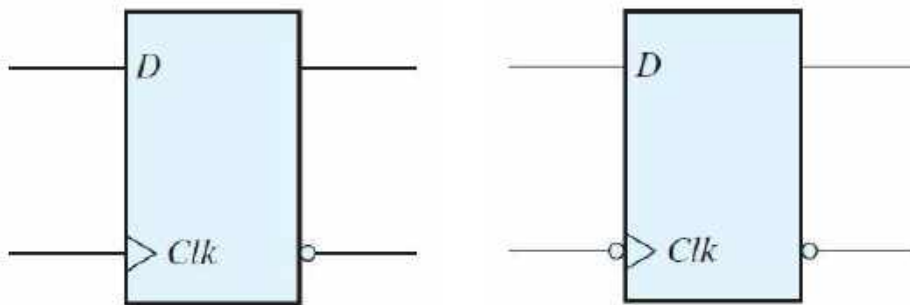
جدول (۴-۶): جدول عملکرد فلیپ فلاپ D حساس به لبه مثبت

<b>D Flip-Flop</b>		
<b>D</b>	<b>Q(t + 1)</b>	
0	0	Reset
1	1	Set

## ۶.۳.۲ نمادهای ترسیمی برای فلیپ فلاپ‌های D

سمبول گرافیکی فلیپ فلاپ D حساس به لبه در شکل زیر مشاهده می‌شود. این سمبول مشابه سمبول لچ D است به جز این که در جلو حرف C علامت فلشی وجود دارد که دینامیکی بودن ورودی را نشان می‌دهد.

نشانگر دینامیک به این معنی است که فلیپ فلاپ به گذر لبه ساعت حساس است. وجود یک دایره در ورودی دینامیکی به معنی نیاز به لبه منفی ساعت است. عدم وجود دایره پاسخ به لبه مثبت را نشان می‌دهد.



*POSTIVE – EDGE NAGATIVE\_EDGE*

شکل (۱۱-۶): سمبول گرافیکی لچ D حساس به لبه مثبت و منفی

### ۶.۳.۳ فلیپ فلاپ JK

با یک فلیپ فلاپ سه عمل را می‌توان انجام داد: Set در ۱، Reset در ۰ و معکوس شدن خروجی.

فلیپ فلاپ JK هر سه کار را انجام می‌دهد. نمودار مدار یک فلیپ فلاپ JK که از یک فلیپ فلاپ D ساخته شده است، در شکل زیر دیده می‌شود. ورودی J، فلیپ فلاپ را در ۱، ورودی K، آن را در ۰ می‌نشاند، وقتی هر دو ورودی در ۱ قرار گیرند، خروجی معکوس می‌شود. صحت این مطلب را می‌توان با بررسی مداری که به ورودی D اعمال شده، تحقیق کرد:

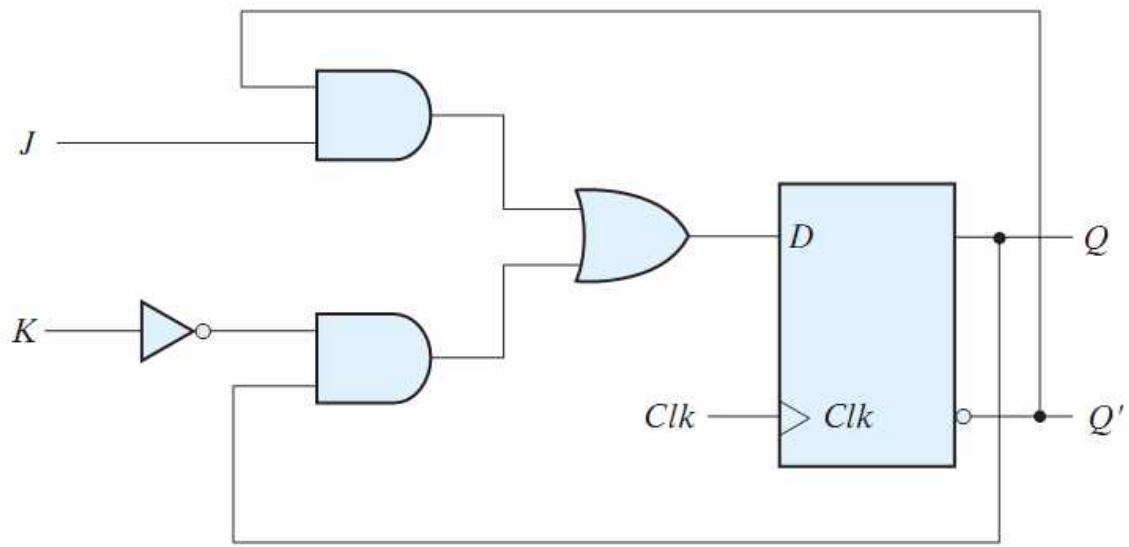
$$D = JQ' + K'Q$$

$$D = 0 * Q' + 1 * Q = 0 + Q = Q$$

$$D = 1 * Q' + 1 * Q = Q' + Q = 1$$

$$D = 0 * Q' + 0 * Q = 0 + 0 = 0$$

$$D = 1 * Q' + 0 * Q = Q' + 0 = Q'$$



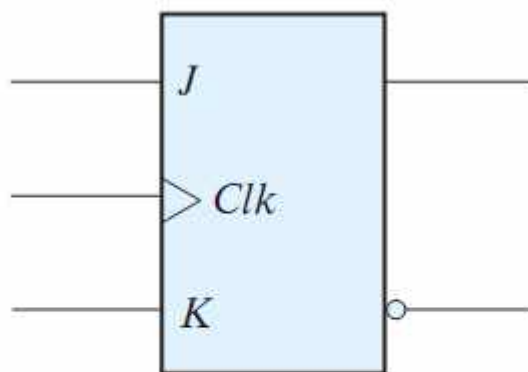
(a) Circuit diagram

شکل (۶-۱۲): مدار ترتیبی فلیپ فلاپ JK

جدول (۶-۴): جدول عملکرد فلیپ فلاپ JK

### 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



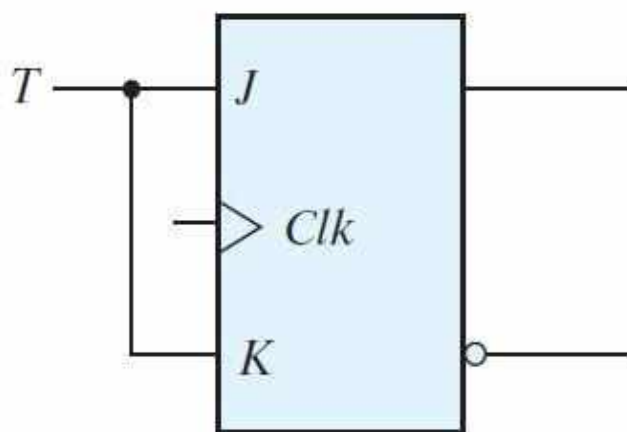
(b) Graphic symbol

شکل (۶-۱۳): سمبول گرافیکی فلیپ فلاپ JK

#### ۶.۳.۴ فلیپ فلاپ T

##### ۶.۳.۴.۱ فلیپ فلاپ T توسط فلیپ فلاپ JK

فلیپ فلاپ T یک فلیپ فلاپ معکوس گر یا متمم ساز است و می توان، آن را با اتصال دو ورودی J و K ایجاد کرد. این عمل در شکل زیر نشان داده شده است. وقتی  $T=0$  باشد،  $(J=K=0)$ ، لبه کلاک، خروجی را عوض نمی کند. وقتی که  $T=1$  است  $(J=K=1)$ ، لبه کلاک، خروجی را معکوس می کند. فلیپ فلاپ متمم ساز در طراحی شمارنده های باینری بسیار مورد توجه است.

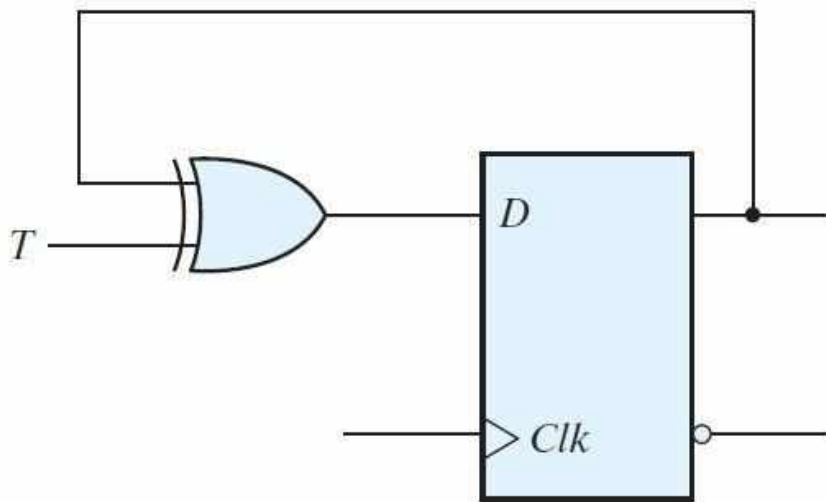


شکل (۶-۱۴): پیاده سازی فلیپ فلاپ T توسط فلیپ فلاپ JK

### ۶.۳.۴.۲ فلیپ فلاپ T توسط فلیپ فلاپ D

یک فلیپ فلاپ T را می‌توان با یک فلیپ فلاپ D و یک گیت XOR مطابق با شکل زیر ساخت. عبارت ورودی D در این حالت برابر است با:

$$D = T \oplus Q = TQ' + T'Q$$

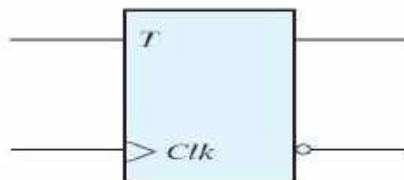


شکل (۶-۱۵): پیاده سازی فلیپ فلاپ T توسط فلیپ فلاپ D

جدول (۶-۵): جدول عملکرد فلیپ فلاپ T

<b>T Flip-Flop</b>		
<b>T</b>	<b>Q(t + 1)</b>	
0	Q(t)	No change
1	Q'(t)	Complement

نماد فلیپ فلاپ T



شکل (۶-۱۶): سمبول گرافیکی فلیپ فلاپ T

### ۶.۳.۵ معادلات مشخصه فلیپ فلاپ‌ها

فلیپ فلاپ D:

$$Q(t + 1) = D$$

فلیپ فلاپ JK:

$$Q(t + 1) = JQ' + K'Q$$

فلیپ فلاپ T:

$$Q(t + 1) = T \oplus Q = TQ' + T'Q$$

### ۶.۴ رجیستر<sup>۱</sup>

یک مدار ترتیبی ساعت دار، متشکل از گروهی فلیپ فلاپ‌ها و گیت‌های ترکیبی است که به منظور تشکیل یک مسیر پشخورد به هم متصل شده‌اند. فلیپ فلاپ‌ها عناصر ضروری مدار هستند، زیرا در غیاب آن‌ها، مدار به یک مدار ترکیبی محض تقلیل می‌یابد (به شرطی که بین گیت‌ها هم مسیر پشخورد وجود نداشته باشد). اما مداری با فلیپ فلاپ حتی در نبود گیت‌های ترکیبی باز هم یک مدار ترتیبی است. مدارهای حاوی فلیپ فلاپ‌ها، معمولاً بر حسب کارشان و نه با نام مدار ترتیبی دسته بندی می‌شوند. دو نوع از این مدارها رجیسترها و شمارنده‌ها هستند.

یک رجیستر گروهی از فلیپ فلاپ‌ها است. هر فلیپ فلاپ قادر است، یک بیت از معلومات را در خود ذخیره نماید. یک رجیستر n بیت، مجموعه‌ای از n فلیپ فلاپ می‌باشد که قادر است n بیت از معلومات باینری را در خود ذخیره نماید. علاوه بر فلیپ فلاپ، یک رجیستر ممکن است، گیت‌های ترکیبی را نیز برای اجرای کارهای پردازشی مختلف داشته باشد. در تعریف جامع‌تر، یک رجیستر متشکل از یک گروه فلیپ فلاپ‌ها و گیت‌هاست که در عمل انتقال با یکدیگر تشریک مساعی دارند. فلیپ فلاپ‌ها معلومات باینری را نگه می‌دارند و گیت‌ها چگونگی انتقال معلومات را به رجیستر معین می‌کنند.

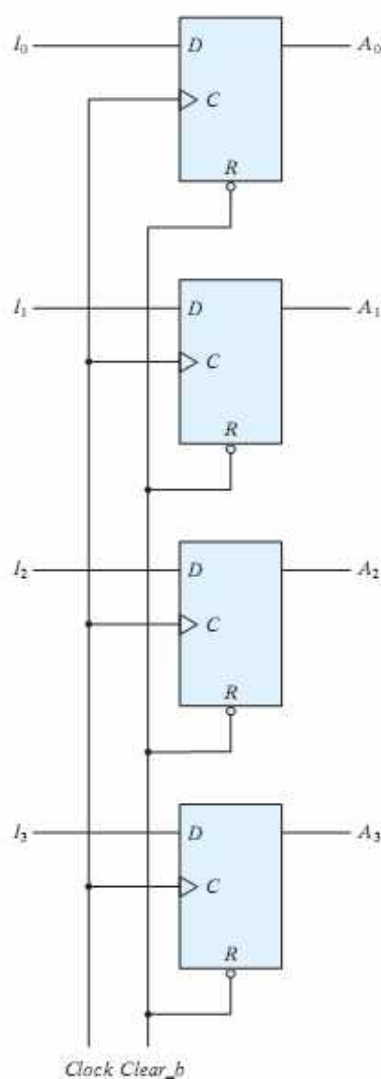
یک شمارنده<sup>۲</sup> اساساً یک رجیستر است که وارد یک رشته از حالات از پیش تعیین شده می‌شود. گیت‌ها در شمارنده‌ها چنان به هم متصل شده‌اند، تا رشته از پیش تعیین شده‌ای از حالات را تولید نمایند. هرچند که شمارنده‌ها نوع خاصی از رجیستر می‌باشند، معمولاً، آن‌ها را بانام‌های متفاوت از رجیسترها جدا می‌کنند.

<sup>۱</sup> Registers

<sup>۲</sup> Counter

انواع متنوعی از رجیسترها در بازار وجود دارند. ساده‌ترین رجیستر، فقط از فلیپ فلاپ‌ها و بدون هرگونه گیتی تشکیل شده است. شکل (۶-۱۷) چنین رجیستری را که از چهار فلیپ فلاپ D ساخته شده نشان می‌دهد.

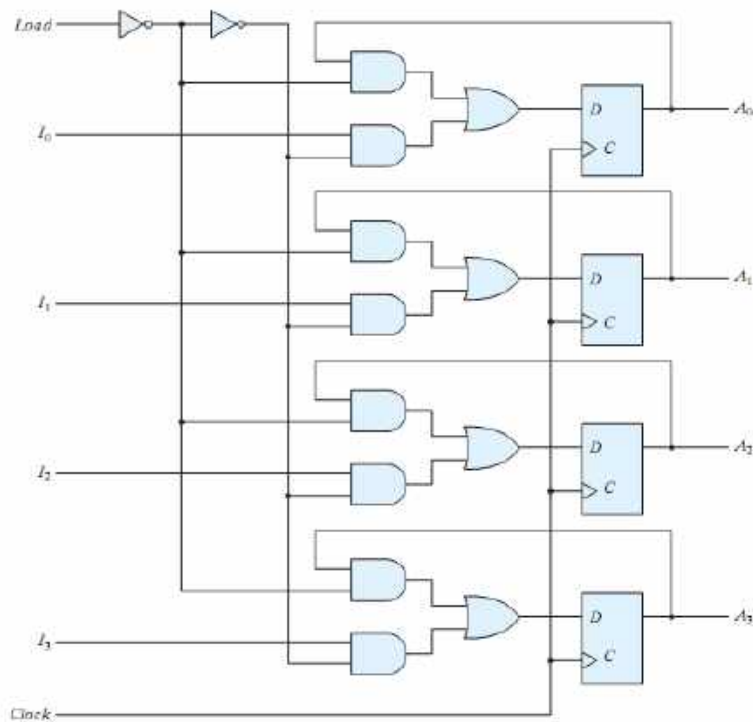
ساعت ورودی مشترک همه فلیپ فلاپ‌ها را با لبه مثبت هر پالس تریگر می‌کند و به این ترتیب معلومات باینری در چهار ورودی به داخل رجیستر ۴ بیت منتقل می‌گردند. می‌توان هر لحظه چهار خروجی را نمونه‌برداری کرده و معلومات باینری ذخیره شده در رجیستر را بدست آورد. ورودی پاک به ورودی باز نشان R همه فلیپ فلاپ‌ها می‌رود. وقتی این ورودی به ۰ رود، همه فلیپ فلاپ‌ها به‌طور غیر هم‌زمانی بازنشانی ۰ می‌شوند. ورودی پاک کردن برای ۰ کردن رجیستر قبل از عمل ساعت زنی مفید است. در حین عمل معمول ساعت زنی، ورودی‌های R باید در منطق ۱ قرار گیرند. توجه کنید که برای ۰ کردن همه حالات در یک رجیستر، می‌توان از پاک کردن، یا باز نشانی استفاده کرد.



شکل (۶-۱۷): رجیستر ۴ بیتی

## ۶.۴.۱ رجیستر با بار شدن موازی<sup>۱</sup>

سیستم‌های دیجیتال هم‌زمان دارای یک مولد ساعت اصلی‌اند که رشته‌ای از پالس‌های ساعت را به‌طور پیوسته فراهم می‌سازند. پالس‌های ساعت به همه فلیپ فلاپ‌ها و رجیسترها در سیستم اعمال می‌شوند. ساعت اصلی، مانند پمپی است که ضربان ثابتی را برای همه بخش‌های سیستم فراهم می‌نماید. برای تأثیر یک پالس ساعت خاص بر روی یک رجیستر خاص، باید یک کنترل جداگانه به کار برده شود. انتقال معلومات جدید به یک رجیستر را بار شدن ثابت می‌نامند. اگر همه بیت‌های ثابت به‌طور هم‌زمان با یک پالس بارز شوند، بار شدن موازی است. لبه ساعت اعمال شده به ورودی‌های  $C$  رجیستر شکل (۶-۱۹) موجب می‌شود تا هر چهار ورودی به‌طور موازی بار گردند. در این آرایش اگر بخواهیم رجیستر بدون تغییر رها شود، باید ساعت از مدار قطع گردد. این کار با کنترل سیگنال ورودی ساعت به وسیله گیت فعال ساز انجام می‌شود. با این وجود قرار دادن گیت‌ها در مسیر ساعت به این معنا است که یک کار منطقی صورت گرفته است. استقرار گیت‌ها موجب تولید تاخیرهای نابرابر در ورودی فلیپ فلاپ‌ها می‌شود. برای هم‌زمانی کامل سیستم باید مطمئن بود که همه پالس‌های ساعت به‌طور هم‌زمان به هر نقطه از سیستم می‌رسد و بنابراین، همه فلیپ فلاپ‌ها به‌طور هم‌زمان تریگر می‌شوند. اعمال پالس ساعت از طریق گیت، تاخیرهای متغیری را موجب می‌شود و ممکن است، سیستم را از هم‌زمانی خارج کند. به این دلیل پیشنهاد می‌شود که کنترل عمل یک رجیستر با ورودی‌های  $D$  به جای کنترل ساعت در ورودی‌های  $C$  فلیپ فلاپ‌ها انجام گیرد.



شکل (۶-۱۸): رجیستر ۴ بیتی با بار شدن موازی

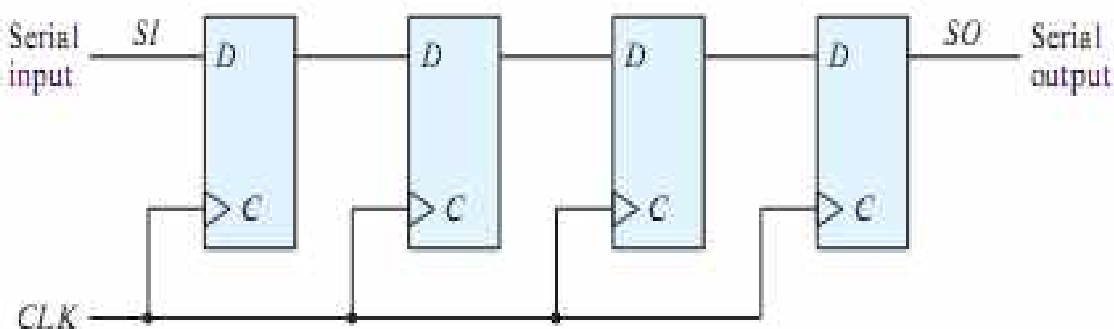
<sup>۱</sup> Register with Parallel Load



## ۶.۴.۲ شیفت رجیستر<sup>۱</sup>

رجیستری که بتواند معلومات باینری اش را به سمت راست یا سمت چپ جا بجا کند، شیفت رجیستر نامیده می شود. ساختار منطقی یک شیفت رجیستر، از چند فلیپ فلاپ تشکیل شده که در آن، خروجی یک فلیپ فلاپ به ورودی فلیپ به فلاپ دیگر متصل است. همه فلیپ فلاپ ها کلاک مشترک دریافت می کنند. کلاک، معلومات را از یک فلیپ فلاپ به فلیپ فلاپ دیگر جا به جا می کند.

ساده ترین شیفت رجیستر طبق شکل (۶-۱۹) فقط از فلیپ فلاپ ها استفاده می کند. خروجی یک فلیپ فلاپ مفروض به ورودی D فلیپ فلاپ سمت راست خود متصل است. هر پالس ساعت، محتوای رجیستر را یک بیت به راست جابه جا می کند. ورودی سریال، تعیین کننده معلوماتی است که از منتهی الیه سمت چپ در حین جا بجایی وارد می شود. خروجی سریال از خروجی سمت راست فلیپ فلاپ اخذ می شود. گاهی لازم است تا جابه جایی را طوری کنترل کنیم که فقط با پالس های معینی رخ دهد. این کار با ممانعت از پالس ساعت در رسیدن به رجیستر امکان پذیر است. بعد نشان خواهیم داد که عمل جابه جایی می تواند از طریق ورودی های D به جای ورودی ساعت رجیستر کنترل شود. در هر صورت اگر شیفت رجیستر شکل (۳-۶) به کار رود، می توان عمل جابجایی را به وسیله یک گیت AND و ورودی که جابجایی را کنترل می کند، تحت کنترل در آورد.



شکل (۶-۱۹): شیفت رجیستر ۴ بیتی

## ۶.۴.۳ انتقال سریال<sup>۲</sup>

اگر یک سیستم دیجیتال هر بار یک بیت را انتقال دهد و یا دستکاری نماید، در مد سریال کار می کند. با جا بجایی یک بیت به خارج رجیستر مبدأ و ورود به رجیستر مقصد، معلومات هر بار یک بیت انتقال می یابد. این بر خلاف انتقال موازی است که در آن همه بیت های رجیستر به طور همزمان انتقال می یابند.

<sup>۱</sup> Shift Registers

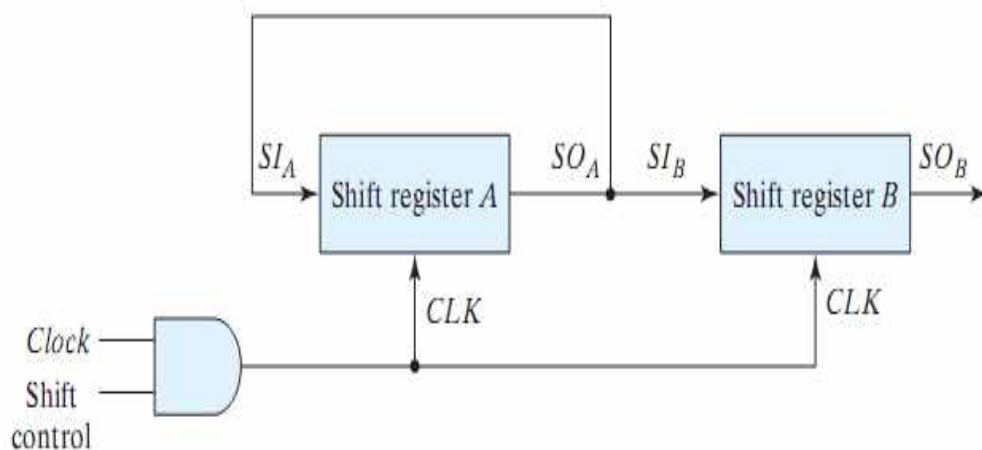
<sup>۲</sup> Serial Transfer

انتقال سریال معلومات از رجیستر A به رجیستر B طبق نمودار بلوکی شکل با شیفت رجیستر انجام می‌شود. خروجی سریال SO از رجیستر A به ورودی سریال SI در رجیستر B وصل است. برای پیشگیری از دست دادن معلومات ذخیره شده در رجیستر مبدأ، معلومات رجیستر A از خروجی سریال به ورودی سریالش چرخانده می‌شود. در حین عمل جابجایی مقدار اولیه رجیستر B به بیرون منتقل شده و از بین می‌رود، مگر این که به رجیستر سومی انتقال یابد. ورودی کنترل جابجایی زمان و تعداد دفعاتی که رجیسترها جابجا می‌شوند را معین می‌سازد. این کار با گیت AND انجام می‌گردد و طی آن پالس‌های ساعت اجازه عبور به پایانه‌های CLK را به هنگام فعال بودن کنترل جابجایی خواهند داشت.

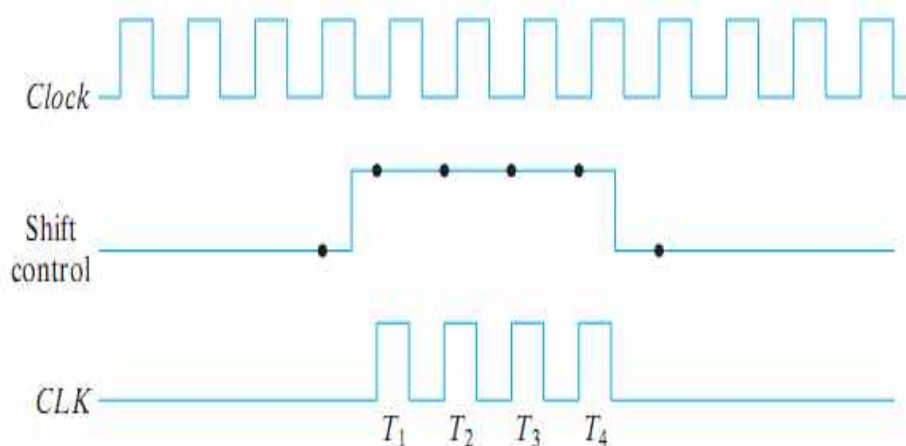
جدول (۶-۶): مثالی از انتقال سریال

### Serial-Transfer Example

Timing Pulse	Shift Register A				Shift Register B			
Initial value	1	0	1	1	0	0	1	0
After $T_1$	1	1	0	1	1	0	0	1
After $T_2$	1	1	1	0	1	1	0	0
After $T_3$	0	1	1	1	0	1	1	0
After $T_4$	1	0	1	1	1	0	1	1



(a) Block diagram



(b) Timing diagram

شکل (۶-۲۰): مدار انتقال سریال

#### ۶.۴.۴ جمع سریال<sup>۱</sup>

عملیات در کامپیوترهای دیجیتال معمولاً به صورت موازی صورت می گیرد، زیرا این روش سریع ترین نوع است. عملیات سریال کندتر است، ولی به قطعات کمتری نیاز دارد. برای ارائه مد سریال، در این جا یک جمع کننده سریال را نشان می دهیم. نوع موازی آن در فصل ۵ مشاهده شد.

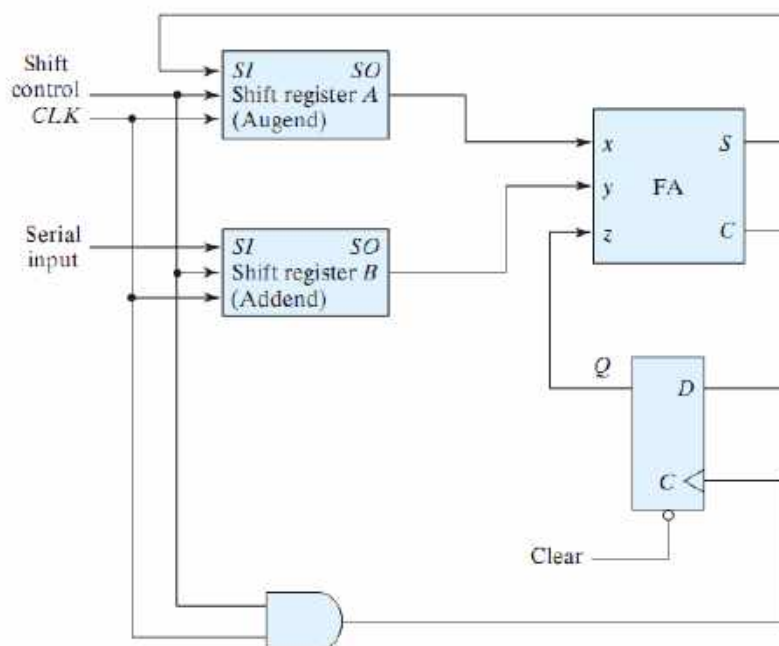
دو عددی که قرار است به طور سریال با هم جمع شوند در دوشیفتر رجیستر ذخیره می شوند. بیت ها، هر جفت یک بار به وسیله یک جمع کننده سریال FA با هم جمع می شوند، مانند شکل (۶-۲۱). نقلی خروجی جمع کننده کامل به فلیپ فلاپ D انتقال می یابد. آن گاه خروجی این فلیپ فلاپ به عنوان نقلی ورودی به

<sup>۱</sup> Serial Addition

جدول (۶-۷): جدول حالت جمع کننده سریال

Present State	Inputs		Next State	Output	Flip-Flop Inputs	
Q	x	y	Q	S	J <sub>Q</sub>	K <sub>Q</sub>
0	0	0	0	0	0	X
0	0	1	0	1	0	X
0	1	0	0	1	0	X
0	1	1	1	0	1	X
1	0	0	0	1	X	1
1	0	1	1	0	X	0
1	1	0	1	0	X	0
1	1	1	1	1	X	0

$$S = X \oplus Y \oplus Q$$



114

## ۶.۴.۵ مقایسه جمع کننده سریال و موازی

جمع کننده سریال:

- ۱ عدد جمع کننده؛
- ۱ عدد فلیپ فلاپ؛
- مدار ترتیبی؛
- کند.

جمع کننده موازی (RCA):

- مدار ترکیبی؛
- N عدد جمع کننده؛
- سریع.

## ۶.۴.۶ شیفت رجیسترهای یونیورسال<sup>۱</sup>

اگر خروجی فلیپ فلاپ‌های یک شیفت رجیستر قابل دسترسی باشد، آن گاه می‌توان معلومات وارده سریال را با جابجایی از خروجی فلیپ فلاپ‌ها به صورت موازی خارج کرد. اگر به شیفت رجیستر یک قابلیت بار شدن موازی اضافه شود، آن گاه داده‌یی وارده موازی به رجیستر را می‌توان با جابجایی به صورت سریال خارج کرد. بعضی از شیفت رجیسترها پایانه‌های لازم را برای انتقال موازی دارا هستند. این مدارها ممکن است قابلیت جابجایی به چپ و راست را هم داشته باشند. عمومی‌ترین شیفت رجیستر دارای امکانات زیر است:

۱. کنترل پاک، برای پاک کردن رجیستر؛
۲. ورودی ساعت بری هم‌زمانی اعمال؛
۳. کنترل جابجایی به راست، برای فعال کردن عمل جابجایی به راست و خطوط ورودی و خروجی سریال مربوط به جابجایی راست؛
۴. کنترل جابجایی به چپ برای فعال کردن عمل جابجایی به چپ و خطوط ورودی و خروجی سریال مربوط به جابجایی چپ؛
۵. یک کنترل بار کردن موازی برای فعال کردن انتقال موازی و n خط ورودی مربوط به انتقال موازی؛
- ۶-n خط خروجی موازی؛

---

<sup>۱</sup> Universal Shift Register

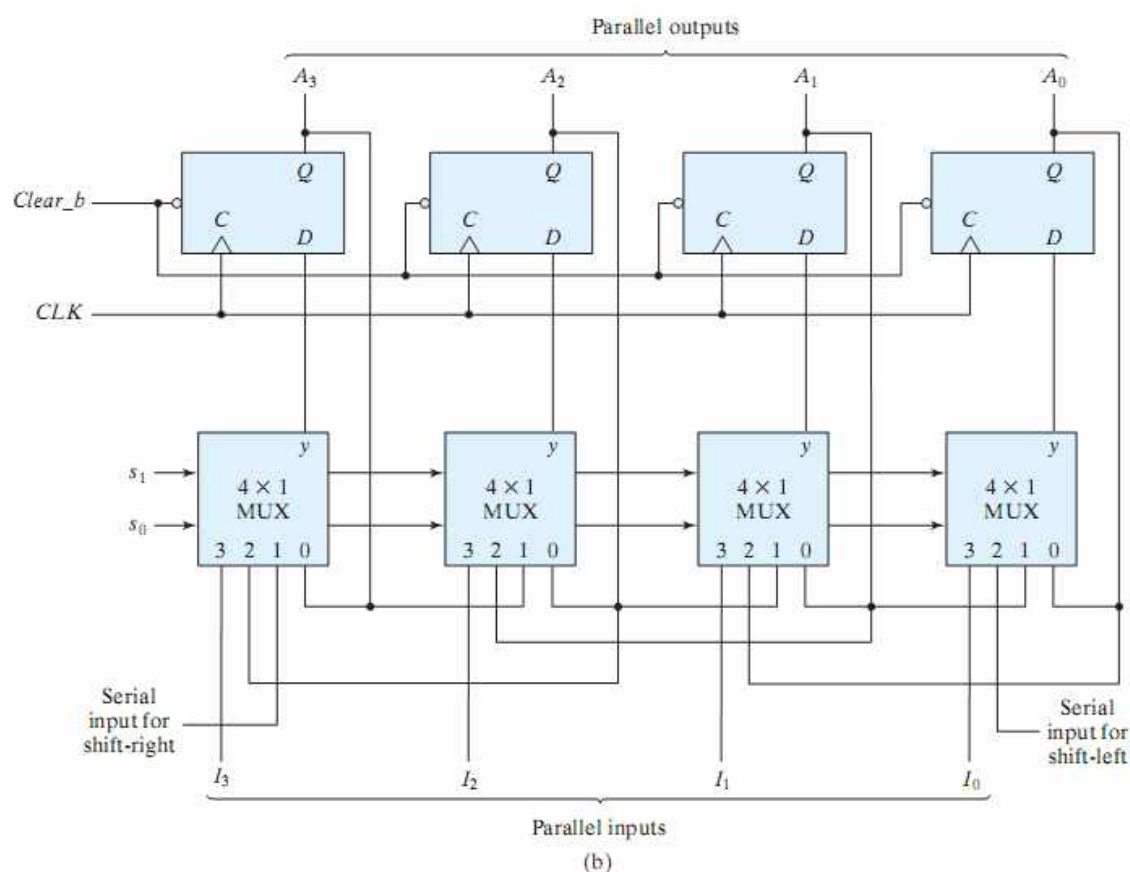
۷. حالت کنترولی که علی‌رغم وجود پالس ساعت معلومات را در رجیستر بدون تغییر نگه می‌دارد.

دیگر شیفت رجیسترها ممکن است، بعضی از امکانات فوق را با حداقل یک عمل جابه جایی یا شیفت داشته باشد.

جدول (۶-۸): جدول عملکرد شیفت رجیستر یونیورسال

**Function Table for the Register**

Mode Control		Register Operation
$s_1$	$s_0$	
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load



شکل (۶-۲۲): مدار شیفت رجیستر یونیورسال

## ۶.۵ شمارنده<sup>۱</sup>

شمارنده، یک رجیستر است که می‌تواند یک دنباله‌ی از پیش تعیین شده از حالت‌ها را دنبال کند. در شمارنده، گیت‌های منطقی طوری به هم وصل شده‌اند تا خروجی مورد نظر تأمین شود.

چرا به شمارنده نیازمندیم:

■ **زمان بندی:** ساختن یک کلاک دقیق در فرکانس‌های پایین (مثل ۱۰ Hz) توسط کریستال‌ها امکان پذیر نیست.

■ **ترتیب:** در پرتاب یک موشک:

پرکردن مخزن‌های سوخت، آتش کردن موتور و ... باید مطابق یک تسلسل دقیق باشد.

■ **شمارش:** چراغ‌های راهنمایی، شمارش ماشین‌ها در ترافیک، ساعت و ...

در این قسمت شمارنده‌های موج گونه و هم‌زمان معرفی شده‌اند:

### ۶.۵.۱ شمارنده‌های موج گونه<sup>۲</sup>

رجیستری که بر اساس اعمال پالس‌های (Pulse) ورودی وارد رشته حالات از پیش تعیین شده می‌شود، شمارنده نام دارد. پالس‌های ورودی ممکن است، پالس‌های ساعت و یا از یک منبع بیرونی با تسلسل ثابت و یا متغیر باشند.

رشته حالات ممکن است که رشته اعداد باینری و یا رشته حالات دیگری باشد. شمارنده‌یی که رشته اعداد باینری را دنبال می‌کند، شمارنده باینری نامیده می‌شود. یک شمارنده  $n$  بیتی متشکل از  $n$  فلیپ فلاپ بوده و می‌تواند، از ۰ تا  $2^n - 1$  را بشمارد.

شمارنده‌ها به دو صورت وجود دارند: شمارنده‌های موج گونه و شمارنده‌های هم‌زمان. در یک شمارنده موج گونه، تغییر وضعیت خروجی فلیپ فلاپ به توان منبع تریگرکردن دیگر فلیپ فلاپ‌ها عمل می‌کند. به بیان دیگر ورودی  $C$  بعضی و یا همه فلیپ فلاپ‌ها با پالس‌های ساعت مشترک تریگر راه اندازی نمی‌شوند. برعکس در شمارنده هم‌زمان، ورودی‌های  $C$  همه فلیپ فلاپ‌ها ساعت مشترک را دریافت می‌نمایند.

### ۶.۵.۲ شمارنده موج گونه باینری<sup>۳</sup>

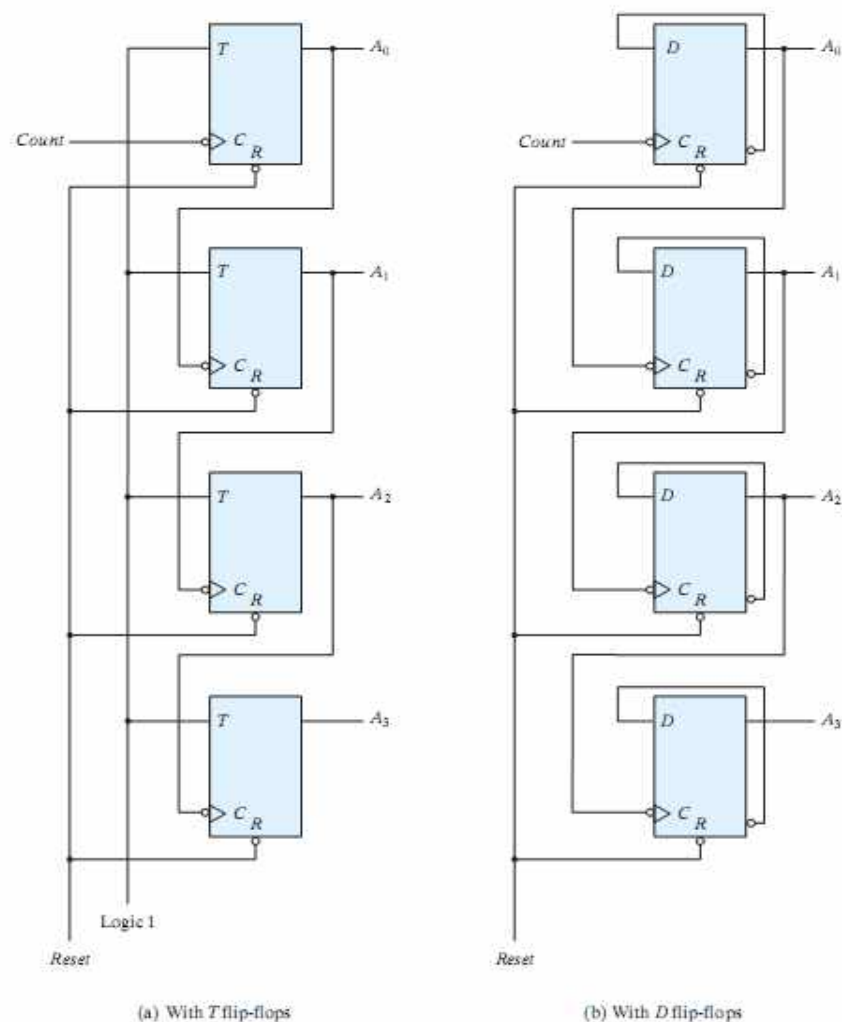
یک شمارنده موج گونه باینری، از یک سری اتصال بین فلیپ فلاپ‌های متمم ساز تشکیل شده است که خروجی هر فلیپ فلاپ به ورودی  $C$  فلیپ فلاپ بالاتر وصل است. فلیپ فلاپی که کم ارزش‌ترین بیت را نگه

<sup>۱</sup> Counter

<sup>۲</sup> Ripple Counters

<sup>۳</sup> Binary Ripple Counter

می‌دارد، پالس‌های مورد مارش ((March را دریافت می‌کند. فلیپ فلاپ متمم ساز را می‌توان با یک فلیپ فلاپ JK که در آن J و K به هم وصل‌اند یا از یک فلیپ فلاپ T ساخت. سومین امکان استفاده از فلیپ فلاپ D است که در آن خروجی متمم به ورودی D وصل است. به این ترتیب ورودی D همواره متمم حالت فعلی بوده و پالس ساعت بعدی موجب متمم شدن خروجی اصلی آن خواهد شد. نمودار منطقی دو شمارنده باینری ۴ بیت در شکل (۶-۲۴) نشان داده شده است. شمارنده با فلیپ فلاپ‌های متمم ساز نوع T در بخش الف و نوع D در بخش (ب) ساخته شده است. خروجی هر فلیپ فلاپ به ورودی فلیپ فلاپ بعدی در رشته متصل است. همان‌طور که گفته شد، فلیپ فلاپی که کم ارزش‌ترین بیت را نگه می‌دارد، پالس‌های شمارش را دریافت می‌کند. ورودی‌های T همه فلیپ فلاپ‌ها در (الف) به‌طور دائم به منطق ۱ متصل‌اند. این شرایط موجب می‌شود تا با گذر منفی در ورودی C فلیپ فلاپ متمم شود. حباب جلو نشانگر دینامیک  $\triangleleft$  در کنار C به این معنی است که فلیپ فلاپ‌ها به لبه منفی ورودی واکنش نشان می‌دهند. گذر منفی هنگامی رخ می‌دهد که خروجی فلیپ فلاپ قبل که به C وصل است، از ۱ به ۰ برود. برای درک عملکرد شمارنده باینری ۴ بیت به ۹ عدد باینری اول در جدول (۶-۱۰) مراجعه کنید.



شکل (۶-۲۳): مدار شمارنده موج گونه



جدول (۶-۹): ترتیب شمارش باینری

*Binary Count Sequence*

$A_3$	$A_2$	$A_1$	$A_0$
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

### مشکل شمارنده‌های موج گونه

- چون خروجی یک فلیپ فلاپ باعث کلاک خوردن فلیپ فلاپ بعدی می‌شود، خروجی فلیپ فلاپ بعدی با کمی تأخیر نسبت به کلاک اصلی معکوس می‌شود.
- این مسئله وقتی حادتر می‌شود که تعداد زیادی از این فلیپ فلاپ‌ها را به هم ببندیم. ممکن است خروجی آخرین فلیپ فلاپ نسبت به کلاک یک پالس ساعت تأخیر داشته باشد که قابل قبول نیست.

راه حل: استفاده از شمارنده‌های هم‌زمان.

- فرق شمارنده‌های هم‌زمان با شمارنده‌های موج گونه در این است که کلاک تمام فلیپ فلاپ‌ها یک‌سان است و با هم تریگر می‌شوند.

### ۶.۵.۳ شمارنده‌های هم‌زمان<sup>۱</sup>

شمارنده‌های هم‌زمان در اعمال پالس ساعت به ورودی فلیپ فلاپ‌ها با شمارنده‌های موج گونه تفاوت دارند. یک ساعت مشترک همه فلیپ فلاپ‌ها را به‌طور هم‌زمان تریگر می‌کند در صورتی که در نوع شمارنده‌های موج گونه هر بار فقط یک فلیپ فلاپ تریگر می‌شود. تصمیم بر ممتد شدن یک فلیپ فلاپ از مقادیر داده‌های ورودی، مانند  $T$ ،  $J$  و  $K$  در لبه ساعت معین می‌شود. اگر  $T=0$  یا  $J=K=0$  باشد حالت فلیپ فلاپ تغییر نمی‌نماید. اگر  $T=1$  یا  $J=K=1$  باشد، فلیپ فلاپ ممتد می‌شود.

<sup>۱</sup> Synchronous Counters

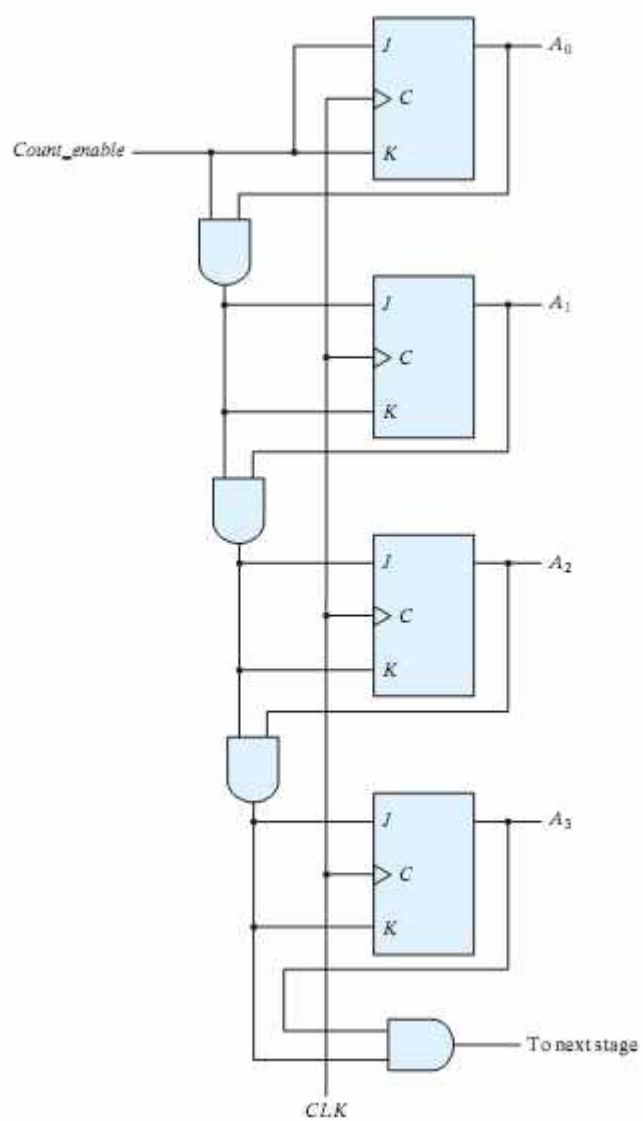
#### ۶.۵.۴ شمارنده باینری<sup>۱</sup>

طراحی یک شمارنده باینری آن قدر ساده است که نیازی به پیگیری مراحل طراحی را ندارد. در شمارنده باینری همزمان فلیپ فلاپ در واقع کم ارزش ترین مکان با هرپالس یکبار متمم می شود. فلیپ فلاپ های واقع در هر مکان هنگامی متمم می شود که همه فلیپ فلاپ ها پایین تر ۱ باشند. مثلاً، اگر حالت فعلی یک شمارنده ۴ بیت  $A_3A_2A_1A_0 = 0011$  باشد، شماره بعدی ۰۱۰۰ خواهد بود. لازم به یادآوری است که در مثال فوق  $A_0$  مرتباً متمم می شود.  $A_1$  هنگامی متمم می شود که  $A_0$  برابر ۱ باشد.  $A_2$  هنگامی ۱ می شود که  $A_1A_0 = 11$  باشد. با این وجود  $A_3$  متمم نمی شود. زیرا، حالات فعلی  $A_2A_1A_0 = 011$  است. چون حالت تمام ۱ وجود ندارد.

شمارنده های باینری همزمان الگوی منظمی دارند و می توان آن ها را با متمم کردن فلیپ فلاپ ها و گیت ها ساخت. نظم الگو را می توان با توجه به شکل (۶-۱۲) ملاحظه کرد. ورودی های C همه فلیپ فلاپ ها به ساعت مشترکی وصل اند. شمارنده با ورودی فعال ساز شمارش، فعال می شود. اگر ورودی فعال ساز ۰ باشد، ورودی همه J ها و K ها برابر ۰ خواهند بود. بنابراین، ساعت قادر نخواهد بود حالت شمارنده را عوض کند. در اولین سطح  $A_0$ ، اگر شمارنده فعال شود  $J=K=1$  خواهد بود. در دیگر سطح ها J ها و K ها به شرطی ۱ هستند که همه طبقات کم ارزش تر آن ها برابر ۱ و ورودی شمارش هم فعال شده باشد. در هر طبقه، زنجیره گیت های AND منطق لازم را برای ورودی های J و K فراهم می کنند. شمارنده را می توان به هر تعداد از طبقات گسترش داد که در آن هر طبقه یک گیت AND و یک فلیپ فلاپ اضافی خواهد داشت و هر گاه همه فلیپ فلاپ های طبقات قبل ۱ شوند، خروجی AND برابر ۱ خواهند بود.

توجه داشته باشید که فلیپ فلاپ ها در لبه مثبت ساعتتریگر می شوند. قطبیت ساعت، آن طور که در شمارنده های موج گونه مهم بود، در این جا اهمیت ندارد. شمارنده همزمان با هر یک از دولبه مثبت یا منفی پالس ساعتتریگر می شود. فلیپ فلاپ های متمم ساز در یک شمارنده باینری می توانند، از نوع JK، T یا D با گیت های XOR باشند. هم عرضی سه نوع فلیپ فلاپ در شکل مشخص شده است.

<sup>۱</sup> Binary Counter



شکل (۶-۲۴۰): شمارنده همزمان باینری



مدارهای ترتیبی مدارهایی اند که علاوه بر ورودی فعلی، مقدار ورودی‌های قبلی نیز بر خروجی‌شان اثر می‌گذارد. از این‌رو مدارهای ترتیبی در ساخت مدارها و عناصر حافظه استفاده می‌شوند. مدارهای ترتیبی دو نوع، هم‌زمان و ناهم‌زمان هستند. عامل هم‌زمانی در مدارهای ترتیبی ساعت یا کلاک است. عنصر پایه و اصلی در ساخت عناصر حافظه لچ‌ها هستند. تمام فلیپ فلاپ‌ها توسط لچ ساخته می‌شوند. هر فلیپ فلاپ توانایی ذخیره‌سازی یک بیت باینری را دارد.

رجیسترها و شمارنده‌ها، دسته دیگری از عناصر حافظه هستند که توسط فلیپ فلاپ‌ها ساخته می‌شوند. رجیسترها برای ذخیره مجموعه‌یی از بیت‌ها استفاده می‌شوند. شمارنده‌ها، ترتیبی از اعداد را پیگیری می‌کنند و برای شمارش به کار می‌روند. شمارنده‌های به‌دو نوع کلی، شمارنده‌های هم‌زمان و شمارنده‌های موج‌گونه تقسیم می‌شوند. مهم‌ترین مشکل شمارنده‌های موج‌گونه تأخیر می‌باشد. به دلیل همین مشکل نیاز به شمارنده‌های هم‌زمان داریم.



## سوالات و فعالیت های فصل ششم

۱. مدار و جدول عملکرد مربوط به لچ SR با ورودی کنترل را رسم کنید.
۲. مدار و جدول عملکرد مربوط به لچ D را رسم کنید.
۳. مدار و جدول عملکرد فلیپ فلاپ JK را رسم کنید.
۴. مدار مربوط به شیفت رجیستر ۸ بیتی را رسم کنید.
۵. مدار مربوط به جمع کننده سریال را رسم کنید.
۶. مدار مربوط به شمارنده همزمان ۴ بیتی را رسم کنید.

### فعالیت ها

۱. مدار و جدول عملکرد مربوط به لچ SR با گیت NOR را در شبیه سازی، مانند logisim رسم کنید.
۲. مدار و جدول عملکرد مربوط به لچ D را در شبیه سازی، مانند logisim رسم کنید.
۳. مدار و جدول عملکرد فلیپ فلاپ D حساس به لبه مثبت را در شبیه سازی، مانند logisim رسم کنید.
۴. مدار مربوط به رجیستر ۴ بیتی را در شبیه سازی، مانند logisim رسم کنید.
۵. مدار مربوط به شیفت رجیستر ۴ بیتی را در شبیه سازی، مانند logisim رسم کنید.
۶. مدار مربوط به شمارنده همزمان ۴ بیتی را در شبیه سازی، مانند logisim رسم کنید.

1. M. Morris Mano, Michael D. Ciletti, 2011. Digital Design, 5<sup>th</sup> ed, Upper Saddle River, NJ: Prentice Hall.
2. Cavanagh, J. J. 1984. Digital Computer Arithmetic. New York: McGraw-Hill.
3. Mano, M. M. 1988. Computer Engineering: Hardware Design. Englewood Cliffs, NJ: Prentice-Hall.
4. Nelson, V. P., H. T. Nagle, J. D. Irwin, and B. D. Carroll. 1997. Digital Logic Circuit Analysis and Design. Upper Saddle River, NJ: Prentice Hall.
5. Schmid, H. 1974. Decimal Computation. New York: John Wiley.
6. Katz, R. H. and Borriello, G. 2004. Contemporary Logic Design, 2nd ed. Upper Saddle River, NJ: Prentice-Hall.
7. Boole, G. 1854. An Investigation of the Laws of Thought. New York: Dover.
8. Dietmeyer, D. L. 1988. Logic Design of Digital Systems, 3rd ed. Boston: Allyn and Bacon.
9. Huntington, E. V. Sets of independent postulates for the algebra of logic. Trans. Am. Math. Soc., 5 (1904): 288–309.
10. IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language, Language Reference Manual (LRM), IEEE Std. 1364-1995, 1996, 2001, 2005, The Institute of Electrical and Electronics Engineers, Piscataway, NJ.
11. IEEE Standard VHDL Language Reference Manual (LRM), IEEE Std. 1076-1987, 1988, The Institute of Electrical and Electronics Engineers, Piscataway, NJ.
12. Mano, M. M. and C. R. Kime. 2000. Logic and Computer Design Fundamentals, 2nd ed. Upper Saddle River, NJ: Prentice Hall.
13. Shannon, C. E. A symbolic analysis of relay and switching circuits. Trans. AIEE, 57 (1938): 713–723.

۱۴. مانو، موریس، ترجمه دکتر قدرت سپید نام (۱۳۸۲)، طراحی دیجیتال، ویرایش سوم، انتشارات خراسان، مشهد ایران.