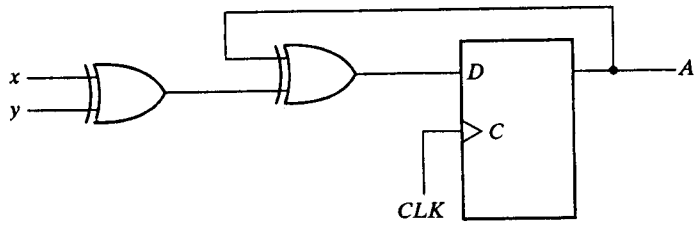
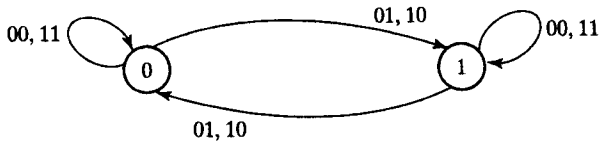
| حالت فعلی | ورودی‌ها | | حالت بعدی |
|---|---|---|---|
| $A$ | $x$ | $y$ | $A$ |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

(ب) جدول حالت




(الف) نمودار مدار




(پ) نمودار حالت

**شکل ۱۷-۵**. مدار ترتیبی با فلیپ فلاپ D

مطابق شکل ۱۷-۵ (ب) لیست شده‌اند. مقادیر حالت بعد، از معادله حالت زیر حاصل می‌شوند:

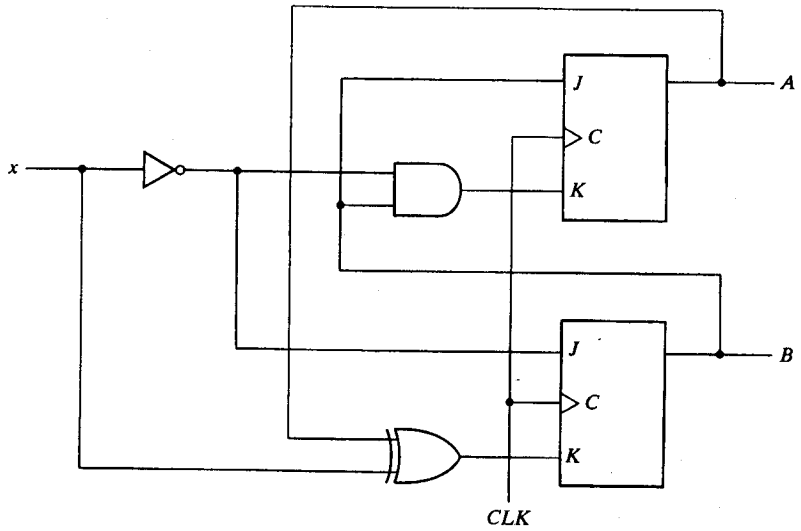
$$A(t + 1) = A \oplus x \oplus y$$

این عبارت یک تابع فرد را بیان می‌دارد و هنگامی برابر 1 است که فقط یک یا سه متغیر برابر 1 باشد. این نکته در ستون حالت بعدی A قابل ملاحظه است.

مدار دارای یک فلیپ فلاپ و دو حالت است. نمودار حالت از دو دایره که هر یک مطابق شکل ۱۷-۵ (پ) متعلق به یک حالت می‌باشد تشکیل گردیده است. حالت فعلی و خروجی، همانطور که با اعداد داخل دوایر نشان داده شده، می‌تواند 0 یا 1 باشد. روی خطوط جهت‌دار به خطوط مورب نیازی نیست زیرا برای مدار ترکیبی هیچ خروجی در نظر گرفته نشده است. دو ورودی، چهار ترکیب ممکن را برای هر حالت ممکن می‌سازند. دو ترکیب ورودی برای هر گذر حالت با یک ویرگول از هم جدا شده‌اند تا شکل مفهوم‌تر باشد.

## تحلیل فلیپ فلاپ‌های JK

یک جدول حالت متشکل از چهار بخش، حالت فعلی، ورودی‌ها، حالت بعدی و خروجی‌هاست. دو مورد اول با لیست حاصل از همه ترکیبات بدست می‌آیند. بخش خروجی از معادلات خروجی حاصل می‌شوند. مقادیر حالت بعدی از معادلات حالت ارزیابی می‌گردند. در فلیپ فلاپ نوع D معادله




**شکل ۱۸-۵**. مدار ترتیبی با فلیپ فلاپ JK

حالت با معادله ورودی یکی است. هنگامی که فلیپ فلاپ‌هایی به جز D مثل JK یا T به کار روند، لازم است به جدول مشخصه یا معادله مشخصه آنها مراجعه شود تا مقادیر حالت بعدی بدست آیند. ما رویه را ابتدا با به‌کارگیری جدول مشخصه و سپس با معادله مشخصه تشریح خواهیم کرد.

مقادیر حالت بعدی یک مدار ترتیبی که از فلیپ فلاپ‌هایی چون نوع JK و T استفاده می‌کنند از رویه زیر بدست می‌آید.

۱- تعیین معادلات ورودی برحسب حالت فعلی و متغیرهای ورودی

۲- لیست مقادیر دودویی هر معادله ورودی

۳- استفاده از جدول مشخصه فلیپ فلاپ برای تعیین مقادیر حالت در جدول حالت.

به عنوان یک مثال، مدار ترتیبی متشکل از دو فلیپ فلاپ A و B از نوع JK و یک ورودی x را طبق شکل ۱۸-۵ ملاحظه نمایید. مدار دارای خروجی خاص نیست و بنابراین نیازی به ستون خروجی در جدول حالت وجود ندارد (خروجی‌های فلیپ فلاپ را می‌توان در این حالت به عنوان خروجی در نظر گرفت). می‌توان مدار را با معادلات ورودی زیر بیان کرد.

$$J_A = B \qquad K_A = Bx'$$
$$J_B = x' \qquad K_B = A'x + Ax' = A \oplus x$$

جدول حالت مدار ترتیبی در جدول ۴-۵ نشان داده شده است. ستون‌های حالت فعلی و ورودی، هشت حالت ممکن را لیست کرده‌اند. مقادیر دودویی زیرستون‌های "ورودی فلیپ فلاپ‌ها" بخشی از جدول حالت نیستند، ولی برای ارزیابی حالت بعدی که در مرحله ۲ از رویه ذکر شد لازم‌اند.

**جدول ۴-۵**. جدول حالت برای مدار ترتیبی با فلیپ فلاپ JK

| حالت فعلی | | ورودی | حالت بعدی | | ورودی های فیلیپ فلاپ | | | |
|---|---|---|---|---|---|---|---|---|
| A | B | x | A | B | $J_A$ | $K_A$ | $J_B$ | $K_B$ |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

این مقادیر دودویی مستقیماً از چهار معادله ورودی مشابه با آنچه برای جدول درستی یک عبارت بول حاصل می‌شوند، بدست آمده‌اند. حالت بعدی هر فلیپ فلاپ از ورودی‌های J و K و جدول مشخصه فلیپ فلاپ JK در جدول ۱-۵ حاصل می‌گردند. چهار حالت برای بررسی وجود دارد. وقتی J = 1 و K = 0 باشد، حالت بعدی 1 است. وقتی J = 0 و K = 1 است، حالت بعدی 0 می‌باشد. با J = K=0، تغییری در حالت وجود ندارد و حالت بعدی با حالت فعلی یکی است. وقتی J = K = 1 باشد. بیت حالت بعدی متمم بیت حالت فعلی است. مثال‌های دو حالت فوق‌الذکر در جدول هنگام AB=10 و x=0 رخ می‌دهند. بنابراین $J_A = K_A = 0$ بوده و حالت فعلی A = 1 است. به این ترتیب حالت بعدی A با حالت فعلی تفاوتی نداشته و مقدار آن 1 است. در همان سطر از جدول $J_B = K_B = 1$ است. چون حالت فعلی B = 0 می‌باشد، حالت بعدی B متمم شده و به 1 تغییر می‌یابد.

می‌توان مقادیر حالت بعدی را از ارزیابی معادلات حالت در معادله مشخصه هم بدست آورد. این کار با دنبال کردن روال زیر میسر است.

۱- معادلات ورودی فلیپ فلاپ را برحسب حالت فعلی و متغیرهای ورودی بدست آورید.

۲- معادلات ورودی را در معادلات مشخصه فلیپ فلاپ جایگزین نمایید تا معادلات حالت حاصل شود.

۳- از معادلات حالت برای تعیین مقادیر حالت بعدی در جدول حالت استفاده نمایید.

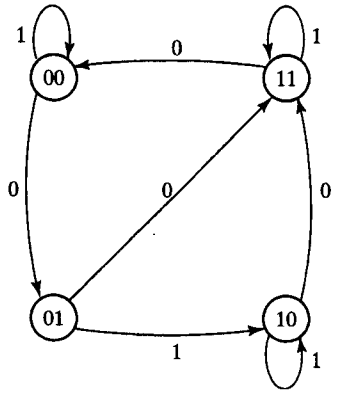
معادلات ورودی فلیپ فلاپ JK شکل ۱۸-۵ در فوق ملاحظه گردید. معادلات مشخصه برای فلیپ فلاپ‌ها از جایگزینی A و B به جای اسم Q بدست می‌آیند:

$$A(t + 1) = JA' + K'A$$
$$B(t + 1) = JB' + K'B$$

از جایگزینی $J_A$ و $K_A$ از معادلات ورودی، معادله حالت برای A بدست می‌آید:

$$A(t + 1) = BA' + (Bx')'A = A'B + AB' + Ax$$

معادله حالت مقادیر بیتی ستون زیر "حالت بعدی" A را در جدول حالت فراهم می‌سازد. به طور مشابه،




**شکل ۱۹-۵**. نمودار حالت شکل ۱۸-۵

معادله حالت برای فلیپ فلاپ B با جایگزینی مقادیر $J_B$ و $K_B$ بدست می‌آید.

$$B(t+1) = x'B' + (A \oplus x)'B = B'x' + ABx + A'Bx'$$

معادله حالت مقادیر بیتی را برای ستون زیر "حالت بعدی" B در جدول حالت فراهم می‌نماید. توجه کنید که وقتی معادله حالت به کار رود ستون‌های زیر ورودی‌های "فلیپ فلاپ" در جدول ۴-۵ لازم نیستند.

نمودار حالت مدار ترتیبی در شکل ۱۹-۵ نشان داده شده است. چون مدار دارای خروجی نیست اعداد روی خطوط جهت‌دار خارج شده از دوایر، تنها مقادیر ورودی x را بیانگر هستند.

### تحلیل با فلیپ فلاپ‌های T

تحلیل یک مدار ترتیبی با فلیپ فلاپ‌های T روال یکسانی با نوع JK دارد. مقادیر حالت بعدی در جدول حالت با جدول مشخصه ۱-۵ یا با معادله مشخصه زیر بدست می‌آیند.

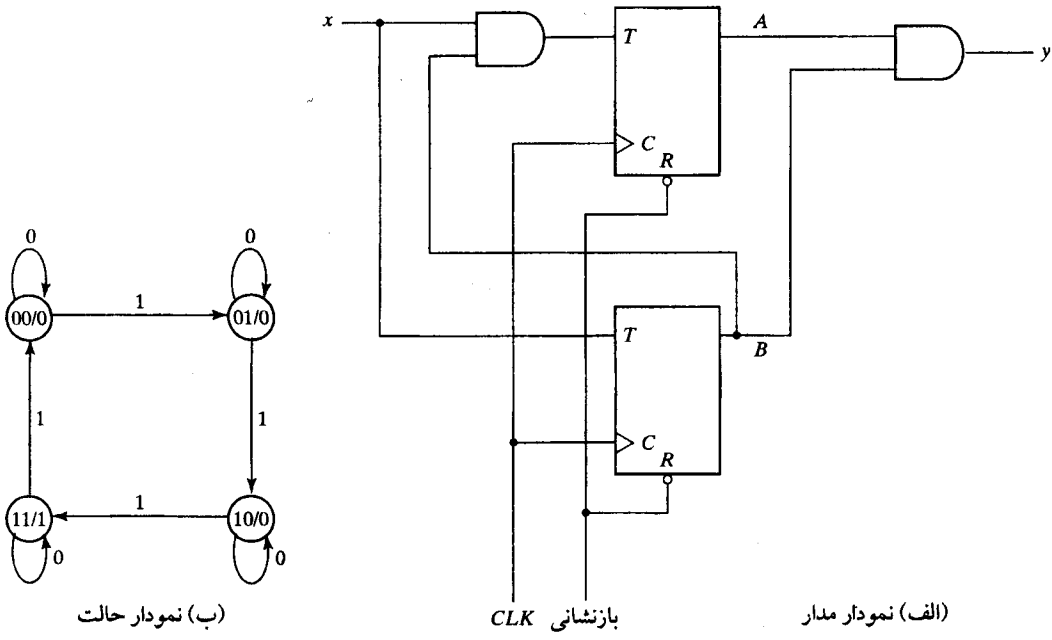
$$Q(t+1) = T \oplus Q = T'Q + TQ'$$

مدار ترتیبی شکل ۲۰-۵ را ملاحظه نمایید. این مدار دارای دو فلیپ فلاپ A و B، یک ورودی x و یک خروجی y است. این مدار با دو معادله ورودی و یک معادله خروجی قابل توصیف می‌باشد.

$$T_A = Bx$$
$$T_B = x$$
$$y = AB$$

جدول حالت برای مدار در جدول ۵-۵ لیست شده است. مقادیر y از معادله خروجی بدست می‌آیند. مقادیر حالت بعدی از معادلات حالت و با جایگزینی $T_A$ و $T_B$ در معادلات مشخصه حاصل می‌شوند، یعنی:

$$A(t+1) = (Bx)'A + (Bx)A' = AB' + Ax' + A'Bx$$
$$B(t+1) = x \oplus B$$




**شکل ۲۰-۵** . مدار ترتیبی با فلیپ فلاپ‌های T

مقادیر حالت بعدی در جدول حالت از عبارات مربوط به دو معادله حالت بدست می‌آید.

نمودار حالت مدار در شکل ۲۰-۵ (ب) ملاحظه می‌شود. مادامی که ورودی x برابر 1 است، مدار به عنوان یک شمارنده دودویی با رشته 00، 01، 10 و 11 عمل می‌کند و در نهایت به 00 باز می‌گردد. وقتی $x = 0$ است، مدار در همان حال باقی می‌ماند. در حالت 11، خروجی $y = 1$ است. در اینجا خروجی فقط به حالت فعلی وابسته بوده و مستقل از ورودی است. دو مقدار داخل هر دایره با یک خط مورب از هم جدا شده‌اند تا حالت فعلی و خروجی از هم تفکیک شوند.

**جدول ۵-۵** . جدول حالت برای مدار ترتیبی با فلیپ فلاپ‌های T

| حالت فعلی | | ورودی | حالت بعدی | | خروجی |
|---|---|---|---|---|---|
| A | B | x | A | B | y |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |

### مدل‌های میلی و مور

عمومی‌ترین مدل یک مدار ترتیبی دارای ورودی‌ها، خروجی‌ها و حالات داخلی است. معمولاً مدارهای ترتیبی به دو صورت مدل مور و میلی ارائه می‌شوند. تفاوت آنها در نحوه تولید خروجی است. در مدل میلی، خروجی تابعی از ورودی و حالت فعلی است. در مدل مور، خروجی فقط تابعی از حالت فعلی می‌باشد. وقتی این دو مدل در کتب و دیگر منابع تکنیکی مورد بحث قرار می‌گیرند، مدارهای ترتیبی را ماشین حالت متناهی یا FSM گویند. مدل میلی یک مدار ترتیبی را FSM میلی یا ماشین میلی نامند. مدل مور را نیز FSM مور یا ماشین مور خوانند.

مثالی از مدل میلی در شکل ۱۵-۵ نشان داده شد. خروجی y تابعی از هر دو ورودی x و حالت فعلی A و B است. نمودار حالت مربوطه در شکل ۱۶-۵ نشان می‌دهد که مقادیر ورودی‌ها و خروجی‌ها با خط موربی در امتداد خطوط جهت‌دار بین حالت‌ها از یکدیگر جدا شده‌اند.

مثالی از مدل مور در شکل ۱۸-۵ نشان داده شد. در اینجا خروجی فقط تابعی از حالت فعلی است. نمودار حالت مربوطه در شکل ۱۹-۵ تنها دارای ورودی‌هایی است که در امتداد خطوط جهت‌دار ذکر شده‌اند. خروجی‌ها همان حالات فلیپ فلاپ‌ها هستند که در داخل دوایر نشان داده شده‌اند. مثال دیگری از یک مدل مور در شکل ۲۰-۵ ملاحظه می‌شود. خروجی فقط به مقادیر موجود در فلیپ فلاپ‌ها وابسته است و بنابراین تنها تابعی از حالت فعلی است. مقدار ورودی در نمودار حالت در امتداد خط جهت‌دار ذکر شده، ولی خروجی‌های مدار در داخل دایره همراه با حالت فعلی نشان داده شده‌اند.

در مدل مور، خروجی‌های مدار ترتیبی با ساعت همزمان است زیرا آنها تنها به خروجی‌های فلیپ فلاپ بستگی دارند که به پالس ساعت وابسته‌اند. در مدل میلی، خروجی‌ها فقط هنگامی که ورودی‌ها در طول پالس ساعت تغییر کنند، عوض می‌شوند. به علاوه، خروجی‌ها ممکن است مقادیر غلط لحظه‌ای نیز داشته باشند زیرا بین لحظه ورود داده و تغییر خروجی‌ها تأخیر وجود دارد. برای همزمان کردن مدار نوع میلی، ورودی‌های مدار ترتیبی باید با پالس ساعت همزمان گردند و خروجی‌ها فقط باید در لبه پالس نمونه‌برداری شوند.

## ۵-۵ HDL برای مدارهای ترتیبی

زبان توصیف سخت‌افزاری Verilog HDL در بخش ۹-۳ معرفی شد. توصیف مدارهای ترکیبی و معرفی مدل‌سازی رفتاری در بخش ۱۱-۴ ارائه گردید. در این بخش بحث مدل‌سازی رفتاری را ادامه می‌دهیم و مثال‌هایی از فلیپ فلاپ‌ها و مدارهای ترتیبی را ارائه خواهیم داد.

### مدل‌سازی رفتاری

در Verilog HDL دو نوع عبارت رفتاری موجود است: initial و always. رفتار initial تنها یک بار و در $t = 0$ اجرا می‌شود. رفتار always مرتباً اجرا شده و تا پایان شبیه‌سازی تکرار می‌گردد. روند رفتاری با استفاده از کلمه کلیدی initial یا always و به دنبال آن عبارت یا بلوکی از عبارت در بین

کلمات کلیدی begin و end آغاز می‌شود. هر مدول می‌تواند حاوی تعداد دلخواهی عبارت initial و یا always باشد. این عبارات به صورت همروند نسبت به یکدیگر از زمان 0 اجرا می‌گردند.

عبارت initial فقط یک بار اجرا می‌شود. این عبارت در شروع شبیه‌سازی آغاز و پس از تکمیل اجرای همه عبارات پایان می‌پذیرد. همانطور که در انتهای بخش ۱۱-۴ مطرح شد، عبارت initial برای تولید سیگنال‌های ورودی در شبیه‌سازی طرح مفید است. در هنگام شبیه‌سازی یک مدار ترتیبی، لازم است یک منبع ساعت برای تریگر کردن فلیپ فلاپ‌ها تولید شود. در زیر دو راه تولید ساعت آزاد گرد (آزادکار) نشان داده شده است:

```
initial
   begin
      clock = 1'b0 ;
      repeat (30)
      #10 clock = ~ clock;
   end
```

```
initial
    begin
      clock = 1'b0;
      #300  $finish;
    end
always
      #10  clock = ~clock;
```

در نوع اول، بلوک initial بین کلمات کلیدی begin و end قرار دارد. ساعت در زمان 0 در 0 تنظیم می‌شود. ساعت هر 10 واحد زمانی یک بار متمم شده و 30 مرتبه تکرار می‌گردد. این کار 15 سیکل ساعت را با سیکل 20 واحد زمانی تولید می‌نماید. در نوع دوم بلوک initial ساعت را در زمان 0 در 0 قرار می‌دهد. پس از 10 واحد زمانی، عبارت always مرتباً ساعت را هر 10 واحد زمانی یک بار متمم می‌کند و در نتیجه زمان سیکل 20 واحد زمانی خواهد بود. شبیه‌سازی در پاسخ به $finish در زمان 300 پایان می‌پذیرد.

عبارت always را می‌توان با تأخیرهایی که برای مدت معین رخ می‌دهند و یا صحت شرایط معین و رخداد وقایع کنترل کرد. در اینجا شرط براساس رخداد واقعه مورد بحث قرار گرفته است. نوع عبارت به شکل زیر است:

```
always  @ (event control expression)
Procedural assignment statements.
```

عبارت کنترل واقعه، شرطی را که باید رخ دهد تا اجرای عبارات تخصیص اجرایی فعال شود، مشخص می‌کند. متغیرها در سمت چپ عبارات اجرایی باید از نوع داده reg و به همان ترتیب نیز باید اعلان شود. سمت راست می‌تواند هر عبارتی باشد که با عملگرهای تعریف شده در Verilog تولید مقدار بنماید.

عبارت کنترل رخداد واقعه که به آن لیست حساسیت هم می‌گویند، وقایعی را مشخص می‌کند که باید رخ دهد تا اجرای عبارات اجرایی واقع در بلوک always آغاز گردد. عبارات داخل بلوک به ترتیب اجرا می‌شوند و پس از آخرین اجرا، عملیات متوقف می‌گردد. آنگاه عبارت always منتظر رخداد واقعه یا پدیده دیگری می‌گردد. در اینجا دو نوع رخداد واقعه وجود دارد: وقایع حساس به سطح و وقایع حساس به لبه. وقایع حساس به سطح در مدارهای ترکیبی و لچ‌ها رخ می‌دهند. مثلاً عبارت

```
always @ (A or B or Reset)
```

موجب اجرای عبارات اجرایی در بلوک always می‌گردد به شرطی که تغییری در A یا B یا ورودی

بازنشان (Reset) رخ دهد. در مدارهای ترتیبی همزمان، تغییرات باید فقط در پاسخ به یک گذر پالس ساعت رخ دهد. گذر می‌تواند در یک لبه مثبت یا لبه منفی اتفاق بیفتد. Verilog HDL با دو کلمه کلیدی negedge و posedege مراقب این شرایط می‌باشد. مثلاً عبارت

```
always @(posedge clock or negedge reset)
```

فقط عبارات اجرایی را در قبال یک گذر مثبت پالس ساعت و یا در لبه منفی بازنشانی اجرا می‌کند.

تخصیص اجرایی تخصیصی است که در یک عبارت initial یا always ذکر می‌گردد. این برخلاف تخصیص مداوم یا پیوسته مورد بحث در بخش ۱۱-۴ همراه با مدل‌سازی روند داده است که در آن عبارت مرتباً ارزیابی می‌شود. دو نوع تخصیص اجرایی وجود دارد: بلوکی و غیربلوکی . این دو با سمبل‌هایی که به کار می‌روند از یکدیگر تفکیک می‌گردند. تخصیص بلوکی از سمبل (=) به عنوان عملگر تخصیص استفاده می‌نماید و تخصیص‌های غیربلوکی سمبل (= <) را به کار می‌برد.

عبارات تخصیص بلوکی به طور متوالی و برحسب لیست شدن در بلوک ترتیبی، اجرا می‌شوند. تخصیص‌های غیربلوکی عبارات را از سمت راست ارزیابی می‌کند و تخصیص را تا اتمام ارزیابی عبارات در سمت چپ پدید نمی‌آورند. دو نوع تخصیص را می‌توان با تشریح آن بهتر درک نمود. دو تخصیص بلوک اجرایی زیر را در نظر بگیرید:

```
B = A
C = B + 1
```

عبارت اول A را به B منتقل می‌کند. دومین عبارت مقدار جدیدی را با افزودن B به میزان یک واحد و انتقال آن به C انجام می‌دهد. در پایان C برابر با A + 1 است. اکنون دو عبارت تخصیص غیربلوکی را در نظر بگیرید:

```
B <= A
C <= B + 1
```

وقتی عبارات اجرا شوند، عبارات سمت راست ارزیابی شده و در مکانی موقت نگهداری می‌شوند. مقدار A در یک مکان از حافظه نگهداری می‌شود در حالی که B + 1 در مکانی دیگر حفظ می‌گردد. در پایان عبارات در بلوک متوالی ارزیابی شده و ذخیره می‌شود، و تخصیص به مقصد سمت چپ صورت می‌گیرد. در این حال C حاوی مقدار اولیه B به علاوه 1 است. اغلب مثال‌ها در این فصل و فصل بعد می‌توانند از تخصیص‌های بلوکی استفاده کنند. هنگامی که با طراحی سطح انتقال ثباتی، مثل فصل ۸، سروکار داریم تخصیص غیربلوکی بهتر و تحکم آمیز است.

### فلیپ فلاپ‌ها و لچ‌ها

مثال‌های ۱-۵ تا ۴-۵ HDL توصیفی از انواع فلیپ فلاپ‌ها و لچ D را نشان داد. لچ D شفاف است و در حین فعال بودن کنترل ورودی، پاسخ لازم به یک تغییر در ورودی داده، را با یک تغییر در خروجی فراهم می‌سازد. توصیف مدولی یک لچ D در مثال ۱-۵ HDL نشان داده شد. این مدول دارای دو ورودی، D و کنترل C ، و یک خروجی Q است. چون Q در یک عبارت اجرایی ارزیابی می‌شود، باید از نوع reg تصور شود. لچ‌ها به سطوح سیگنال ورودی پاسخ می‌دهند بنابراین ورودی‌ها در عبارت کنترل

مثال ۵-۱، HDL

```
//Description of D latch (See Fig. 5-6)
module D_latch (Q,D,control);
   output Q;
   input D,control;
   reg Q;
   always @ (control or D)
   if (control) Q = D;      //Same as: if (control == 1)
endmodule
```

واقعه بدون نشانه لبه و به دنبال @ در عبارت always ذکر می‌شوند. در اینجا تنها یک عبارت تخصیص اجرایی بلوکی وجود دارد و انتقال ورودی D را به خروجی Q اگر کنترل صحیح (در منطق 1) باشد مشخص می‌نماید. توجه کنید که این عبارت هر بار که تغییر در D رخ دهد و کنترل در 1 باشد، اجرا خواهد شد.

مثال ۵-۲ HDL دو فلیپ فلاپ D را با دو مدول توصیف می‌کند. اولی فقط به پالس ساعت واکنش نشان می‌دهد و دومی دارای یک ورودی بازنشانی غیرهمزمان نیز هست. خروجی Q باید علاوه بر خروجی بودن به عنوان داده نوع reg هم اعلان شود. دلیل این است که در یک عبارت تخصیص اجرایی این خروجی، هدف می‌باشد. کلمه کلیدی posedge انتقال ورودی D را به Q فقط در لبه مثبت گذر CLK تضمین می‌نماید. هر تغییر در D در هر زمان دیگر Q را تغییر نخواهد داد.

مدول دوم علاوه بر ساعت همزمان کننده، دارای ورودی بازنشان غیرهمزمان نیز هست. برای تولید چنین فلیپ فلاپی فرم خاصی از عبارت if به‌کار می‌رود. عبارت واقعه یا رخداد پس از سمبل @ در عبارت always ممکن است هر تعداد از وقایع لبه از نوع posedge یا negedge باشد. یکی از وقایع باید رخداد یک ساعت در نظر گرفته شود. بقیه وقایع شرایطی را مشخص می‌نمایند که تحت آن منطق غیرهمزمان اجرا

مثال ۵-۲، HDL

```
//D flip-flop
module D_FF (Q,D,CLK);
   output Q;
   input D,CLK;
   reg Q;
   always @ (posedge CLK)
     Q = D;
endmodule

//D flip-flop with asynchronous reset.
module DFF (Q,D,CLK,RST);
   output Q;
   input D,CLK,RST;
   reg Q;
   always @(posedge CLK or negedge RST)
     if (~RST) Q = 1'b0;     // Same as: if (RST == 0)
     else Q = D;
endmodule
```

می‌شود، هر if یا else if در عبارات تخصیص اجرایی مربوط به یک واقعه غیرهمزمان است. آخرین عبارت else مربوط به رخداد ساعت است. در مدول دوم مثال ۲-۵ دو رخداد لبه وجود دارد.

واقعه negedge RST (بازنشانی) از نوع غیرهمزمان است زیرا با عبارت if(~RST) انطباق دارد. مادامی که RST = 0 است، Q به 0 پاک می‌گردد. اگر CLK گذر مثبت داشته باشد، اثرش بلوکه می‌شود. تنها وقتی که RST = 1 است واقعه ساعت posedge به طور همزمان D را به Q منتقل می‌نماید.

معمولاً لازم است فلیپ فلاپ‌ها دارای سیگنال ورودی بازنشانی یا پیش تنظیم باشند، در غیر این صورت، حالت اولیه مدار ترتیبی را نمی‌توان معین کرد. یک مدار ترتیبی را نمی‌توان با HDL شبیه‌سازی کرد مگر این که حالت اولیه‌ای همراه با یک سیگنال ورودی به آن تخصیص یابد.

مثال ۳-۵ HDL ساخت فلیپ فلاپ T یا JK را با فلیپ فلاپ D و تعدادی گیت نشان می‌دهد. مدار با استفاده از معادلات مشخصه فلیپ فلاپ‌ها توصیف شده است.

$$Q(t+1) = Q \oplus T \qquad \text{برای یک فلیپ فلاپ } T$$
$$Q(t+1) = JQ' + K'Q \qquad \text{برای یک فلیپ فلاپ } JK$$

**مثال ۳-۵ HDL**

```
//T flip-flop from D flip-flop and gates
module TFF (Q,T,CLK,RST);
   output Q;
   input T,CLK,RST;
   wire DT;
   assign DT = Q ^ T ;
//Instantiate the D flip-flop
   DFF TF1 (Q,DT,CLK,RST);
endmodule

//JK flip-flop from D flip-flop and gates
module JKFF (Q,J,K,CLK,RST);
   output Q;
   input J,K,CLK,RST;
   wire JK;
   assign JK = (J & ~Q) | (~K & Q);
//Instantiate D flipflop
   DFF JK1 (Q,JK,CLK,RST);
endmodule

//D flip-flop
module DFF (Q,D,CLK,RST);
   output Q;
   input D,CLK,RST;
   reg Q;
   always @ (posedge CLK or negedge RST)
     if (~RST) Q = 1'b0;
     else Q = D;
endmodule
```

مثال ۴-۵، HDL

```
// Functional description of JK flip-flop
module  JK_FF (J,K,CLK,Q,Qnot);
   output Q,Qnot;
   input  J,K,CLK;
   reg  Q;
   assign Qnot = ~ Q ;
   always @ (posedge CLK)
          case ({J,K})
            2'b00: Q = Q;
            2'b01: Q = 1'b0;
            2'b10: Q = 1'b1;
            2'b11: Q = ~ Q;
          endcase
endmodule
```

مدول اول TFF یک فلیپ فلاپ T ساخته شده با DFF را ذکر می‌کند. سیم DT طبق نیاز برای تبدیل فلیپ فلاپ D به T به XOR مربوط به Q و T تخصیص یافته است. ذکر مقدار DT به جای D در مدول DFF، تولید فلیپ فلاپ T موردنظر را می‌نماید. فلیپ فلاپ JK هم به روشی مشابه با استفاده از معادله مشخصه آن انجام می‌شود تا به این ترتیب جایگزینی برای D در DFF باشد.

مثال ۴-۵ HDL روش دیگری برای توصیف یک فلیپ فلاپ JK را نشان می‌دهد. در اینجا ما به جای معادله مشخصه از جدول مشخصه استفاده می‌کنیم. شرط انشعاب چندگانه case، عدد دو بیتی حاصل از بیت‌های J و K را چک می‌کند. مقدار case ({K و J}) ارزیابی شده و با مقادیر در لیست عباراتی که به دنبال می‌آیند مقایسه می‌شوند. اولین مقداری که با شرط صحیح تطابق داشته باشد اجرا می‌گردد. چون ترکیب J و K یک عدد دو بیتی را تولید می‌کند، عدد می‌تواند 00، 01، 10 یا 11 باشد. اولین عدد مقدار J و دومی مقدار K را بدست می‌دهد. چهار شرط ممکن مقدار حالت بعدی Q پس از اعمال لبه مثبت پالس ساعت را مشخص می‌نماید.

## نمودار حالت

عملکرد مدارهای ترتیبی در HDL به وسیله قالب مشابهی با نمودار حالت توصیف می‌گردد. نمودار حالت مدل میلی در مثال ۵-۵ HDL ارائه شده است. ورودی، خروجی، ساعت و بازنشانی به روش معمولی اعلان می‌شوند. حالت فلیپ فلاپ‌ها با شناسه Prstate و Nxtstate اعلان می‌گردد. این متغیرها مقدار حالت یک مدار ترتیبی را نگه می‌دارند. تخصیص دودویی حالت با استفاده از یک عبارت parameter صورت می‌گیرد. Verilog تعریف ثابت‌ها را در مدول با کلمه کلیدی parameter اجازه می‌دهد. به چهار حالت S0 تا S3 اعداد دودویی 00 تا 11 تخصیص داده شده است. عبارت S2=2'b10 بر S2 = 2 ترجیح داده می‌شود. اولی از دو بیت برای ذخیره یک ثابت استفاده می‌کند. دومی اعداد دودویی 32 (یا 64) بیت را نتیجه می‌دهد.

```
//Mealy state diagram (Fig. 5-16)
module Mealy_mdl (x,y,CLK,RST);
  input x,CLK,RST;
  output y;
  reg y;
  reg [1:0] Prstate, Nxtstate;
  parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;
   always @ (posedge CLK or negedge RST)
      if (~RST) Prstate = S0;  //Initialize to state S0
      else Prstate = Nxtstate; //Clock operations
   always @ (Prstate or x)     //Determine next state
         case (Prstate)
            S0: if (x) Nxtstate = S1;
                  else Nxtstate = S0;
            S1: if (x) Nxtstate = S3;
                  else Nxtstate = S0;
            S2: if (~x)Nxtstate = S0;
                  else Nxtstate = S2;
            S3: if (x) Nxtstate = S2;
                  else Nxtstate = S0;
         endcase
   always @ (Prstate or x)     //Evaluate output
         case (Prstate)
            S0: y = 0;
            S1: if (x) y = 1'b0; else y = 1'b1;
            S2: if (x) y = 1'b0; else y = 1'b1;
            S3: if (x) y = 1'b0; else y = 1'b1;
         endcase
  endmodule
```

توصیف HDL از سه بلوک always استفاده می‌نماید که به طور همروند اجرا می‌شوند و از طریق متغیرهای مشترک با هم واکنش نشان می‌دهند. اولین عبارت always مدار را به حالت اولیه S0 = 00 باز می‌نشاند و عملکرد ساعت‌دار همزمان را بیان می‌دارد. عبارت Prstate = Nxtstate تنها در پاسخ به لبه مثبت پالس ساعت اجرا می‌گردد. این بدان معنی است که هر تغییر در مقدار Nxtsate در بلوک always دوم به عنوان یک واقعه posedge به prstate منتقل می‌شود. دومین بلوک always حالت بعدی را به عنوان تابعی از حالت فعلی ورودی معین می‌نماید. شرط انشعاب چند مسیری، رشته حالات را در نمودار حالت شکل ۱۶-۵ دنبال می‌کند. سومین بلوک always خروجی را به عنوان یک تابع از حالت فعلی و ورودی ارزیابی می‌نماید. هر چند این بلوک به طور جداگانه به خاطر وضوح لیست شده است، ولی می‌تواند همراه با بلوک دوم ذکر شود. توجه کنید که اگر ورودی x در هر حالتی تغییر کند، مقدار خروجی y ممکن است تغییر یابد.

مثالی از نمودار حالت مدل مور در مثال ۶-۵ HDL آورده شده است. این مثال نشان می‌دهد که می‌توان گذرهای حالت را تنها با یک بلوک always نشان داد. حالت فعلی مدار با متغیر state بیان

```
//Moore state diagram (Fig. 5-19)
module Moore_mdl (x,AB,CLK,RST);
   input x,CLK,RST;
   output [1:0]AB;
   reg [1:0] state;
   parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;
      always @ (posedge CLK or negedge RST)
         if (~RST) state = S0;  //Initialize to state S0
         else
         case (state)
           S0: if (~x) state = S1; else state = S0;
           S1: if (x)  state = S2; else state = S3;
           S2: if (~x) state = S3; else state = S2;
           S3: if (~x) state = S0; else state = S3;
         endcase
   assign AB = state;          //Output of flip-flops
endmodule
```

می‌شود. گذرهای حالت با لبه مثبت ساعت یعنی posedge CLK طبق شرایط عبارت case رخ می‌دهند. خروجی مدار مستقل از ورودی است و مستقیماً از خروجی‌های فلیپ فلاپ اخذ می‌شوند. خروجی 2 بیت AB با عبارت assign معین می‌شود و برابر با مقدار حالت فعلی است.

## توصیف ساختاری

مدارهای ترکیبی را می‌توان در HDL با استفاده از عبارات سطح گیت یا روند داده توصیف کرد. مدارهای ترتیبی از عبارات رفتاری استفاده می‌کنند تا عملکرد فلیپ فلاپ‌ها را توصیف نمایند. چون یک مدار ترتیبی از فلیپ فلاپ‌ها و گیت‌ها استفاده می‌کند، ساختارش با ترکیبی از عبارات روند داده و رفتاری توصیف می‌گردد. فلیپ فلاپ‌ها با عبارت always توصیف می‌شوند. بخش ترکیبی را می‌توان با عبارات assign و معادلات بول توصیف کرد. آنگاه می‌توان بلوک‌های جداگانه را با ذکر با هم ترکیب کرد.

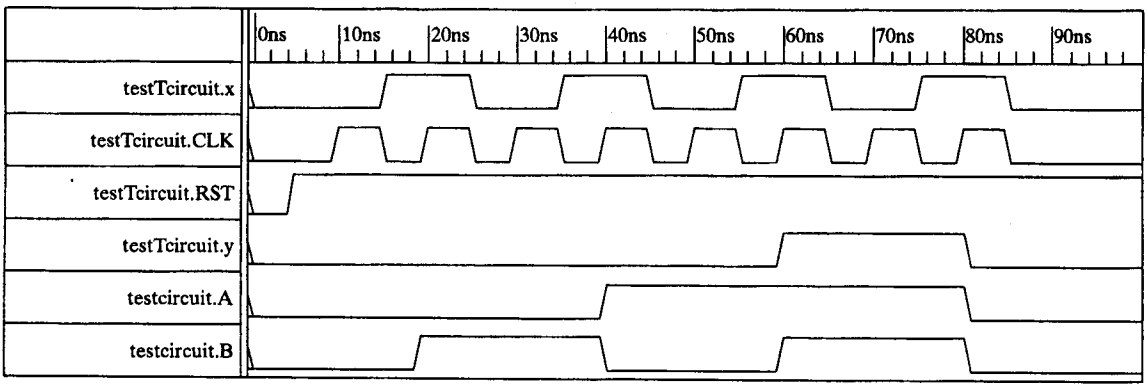
توصیف ساختاری یک مدار ترتیبی در مثال ۵-۷ HDL نشان داده شده است. در مثال دو مدول وجود دارد. اولی مدار شکل ۵-۲۰ (الف) را توصیف می‌کند. دومی فلیپ فلاپ T را توصیف می‌نماید. مدول دیگری نیز وجود دارد که نیروی محرکه لازم را برای تست عملکرد مدار فراهم می‌سازد. مدار ترتیبی یک شمارنده دودویی است که با ورودی x کنترل می‌شود. خروجی y وقتی که شمارش دودویی به 11 برسد، فعال می‌شود. فلیپ فلاپ‌های A و B نیز به عنوان خروجی‌ها به حساب آمده‌اند تا بتوان عملکرد آنها را چک کرد. معادلات ورودی فلیپ فلاپ‌ها و معادله خروجی با عبارات assign با توجه به عبارات بولی ارزیابی می‌شوند. آنگاه فلیپ فلاپ T با TA و TB و معادلات ورودی ذکر می‌گردد.

مدول دوم فلیپ فلاپ T را توصیف می‌کند. ورودی RST فلیپ فلاپ را در لبه منفی به 0 باز می‌نشاند. عملکرد مدار با معادله مشخصه‌اش، یعنی $Q(t+1) = Q \oplus T$ مشخص می‌شود.

```
//Structural description of sequential circuit
//See Fig. 5-20(a)
module Tcircuit (x,y,A,B,CLK,RST);
   input x,CLK,RST;
   output y,A,B;
   wire TA,TB;
//Flip-flip input equations
   assign TB = x,
          TA = x & B;
//Output equation
   assign y = A & B;
//Instantiate T flip-flops
   T_FF BF (B,TB,CLK,RST);
   T_FF AF (A,TA,CLK,RST);
endmodule

//T flip-flop
module T_FF (Q,T,CLK,RST);
   output Q;
   input T,CLK,RST;
   reg Q;
     always @ (posedge CLK or negedge RST)
       if (~RST) Q = 1'b0;
       else Q = Q ^ T;
endmodule

//Stimulus for testing sequential circuit
module testTcircuit;
  reg x,CLK,RST;  //inputs for circuit
  wire y,A,B;     //output from circuit
  Tcircuit TC (x,y,A,B,CLK,RST);  // instantiate circuit
  initial
     begin
         RST = 0;
         CLK = 0;
      #5 RST = 1;
         repeat (16)
      #5 CLK = ~CLK;
     end
  initial
     begin
          x = 0;
      #15 x = 1;
          repeat (8)
      #10 x = ~ x;
     end
endmodule
```




**شکل ۲۱-۵**. شبیه سازی خروجی با مثال ۷-۵ HDL

مدول محرک، ورودی‌ها به مدار را برای چک کردن پاسخ فراهم می‌سازد. اولین بلوک initial هشت سیکل ساعت را با پریود 10ns تهیه می‌کند. دومین بلوک inital هر تغییر در ورودی x راکه در لبه منفی پالس ساعت رخ دهد معین می‌سازد. نتیجه شبیه‌سازی در شکل ۲۱-۵ ملاحظه می‌گردد. خروجی A و B وارد رشته‌ای از ترکیبات دودویی 00 ، 01 ، 10 و 11 و سپس به 00 می‌شود. تغییر در شمارش در هنگام لبه مثبت ساعت و با شرط $x = 1$ رخ می‌دهد. اگر $x = 0$ باشد شمارش تغییری نمی‌کند. وقتی هر دو A و B برابر 1 باشد خروجی y برابر 1 خواهد بود. این خود صحت عمل مدار را تأیید می‌نماید.

## ۶-۵ کاهش و تخصیص حالت

تحلیل مدارهای ترتیبی از نمودار مدار آغاز و به جدول یا نمودار حالت پایان می‌یابد. طراحی یک مدار ترتیبی با مجموعه‌ای از مشخصات و ویژگی‌ها شروع و به یک نمودار منطقی ختم می‌شود. روال‌های طراحی از بخش ۷-۵ به بعد ارائه خواهند شد. این بخش خواص معینی از مدارهای ترتیبی که ممکن است تعداد گیت‌ها و فلیپ فلاپ‌ها را در طراحی کاهش دهد مورد بحث قرار خواهد داد.

### کاهش حالت

کاهش تعداد فلیپ فلاپ‌ها در یک مدار ترتیبی را کاهش حالت می‌نامند. الگوریتم‌های کاهش حالت رویه‌هایی را شامل می‌شود که در آن تعداد حالات در یک جدول حالت کاهش می‌یابد، ضمن این که نیازهای ورودی و خروجی حفظ می‌گردند. چون m فلیپ فلاپ، $2^m$ حالت را تولید می‌کنند، کاهش در تعداد حالات ممکن است گاهی به کاهش در تعداد فلیپ فلاپ‌ها منجر گردد. یک تأثیر پیش‌بینی نشده در کاهش تعداد فلیپ فلاپ‌ها این است که گاهی مدار معادل با چند فلیپ فلاپ کمتر ممکن است گیت‌های بیشتری در بخش ترکیبی لازم داشته باشد.

ما روال کاهش حالت را با مثالی تشریح می‌کنیم. برای این کار ویژگی‌های نمودار حالت شکل ۲۲-۵

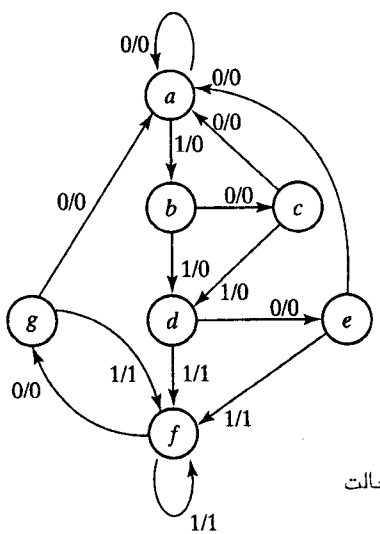
را برمی‌گزینیم. در این مثال تنها رشته‌های ورودی- خروجی اهمیت دارند؛ و از حالات داخلی فقط برای تهیه رشته موردنظر استفاده می‌شود. به این دلیل حالات درون دایره‌ها با سمبل‌های حرفی، به جای مقادیر دودویی ذکر شده‌اند. این برخلاف یک شمارنده دودویی است که در آن مقدار دودویی رشته حالات به عنوان خروجی به بیرون برده می‌شوند.

تعداد رشته‌های ورودی قابل اعمال به مدار بی‌نهایت است؛ و هر کدام، رشته خروجی منحصر به فرد خود را دارد. مثلاً رشته ورودی 01010110100 را با حالت اولیه $a$ در نظر بگیرید. هر ورودی 0 یا 1، یک خروجی 0 یا 1 تولید کرده و موجب می‌شود مدار به حالت بعدی برود. با توجه به نمودار حالت ملاحظه می‌شود که رشته خروجی و حالت برای رشته ورودی مفروض به طریق زیر بدست می‌آید. با فرض قرار داشتن مدار در حالت اولیه $a$، یک ورودی 0 یک خروجی 0 تولید کرده و مدار در همان حالت $a$ باقی می‌ماند. با حالت فعلی $a$ اگر ورودی 1 باشد، خروجی 0 و حالت بعدی b است. با حالت فعلی b و ورودی 0، خروجی 0 و حالت بعدی c است. با ادامه این روند رشته کامل را مطابق زیر خواهیم یافت.

| حالت | $a$ | $g$ | $f$ | $g$ | $f$ | $f$ | $e$ | $d$ | $c$ | $b$ | $a$ | $a$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ورودی | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| خروجی | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

در هر ستون، حالت فعلی، مقدار ورودی و مقدار خروجی را داریم. حالت بعدی هر حالت در ستون بعدی نوشته شده است. باید متذکر شد که در این مدار، حالات از درجه دوم اهمیت برخوردارند زیرا ما فقط به رشته خروجی حاصل از رشته ورودی علاقمندیم.

اکنون اجازه بدهید فرض کنیم که یک مدار ترتیبی را که نمودار حالت آن کمتر از 7 حالت دارد در اختیار داریم و مایلیم آن را با مداری که نمودار حالتش در شکل ۲۲-۵ داده شده است مقایسه نماییم. اگر ورودی یکسانی به هر دو مدار اعمال شود و خروجی‌های مشابهی برای تمام رشته ورودی تولید گردد،




**شکل ۲۲-۵**. نمودار حالت

**جدول ۶-۵** . جدول حالت

| حالت فعلی | حالت بعدی | | خروجی | |
|---|---|---|---|---|
| | $x = 0$ | $x = 1$ | $x = 0$ | $x = 1$ |
| $a$ | $a$ | $b$ | 0 | 0 |
| $b$ | $c$ | $d$ | 0 | 0 |
| $c$ | $a$ | $d$ | 0 | 0 |
| $d$ | $e$ | $f$ | 0 | 1 |
| $e$ | $a$ | $f$ | 0 | 1 |
| $f$ | $g$ | $f$ | 0 | 1 |
| $g$ | $a$ | $f$ | 0 | 1 |

می‌توان یکی را به جای دیگری جایگزین کرد. موضوع کاهش حالت یافتن راهی برای کاهش حالات در مدار ترتیبی بدون تغییر رابطه ورودی- خروجی است.

اکنون به منظور کاهش حالات این مثال به پیش می‌رویم. ابتدا به جدول حالت نیاز داریم. در اعمال رویه‌ها برای کاهش حالت استفاده از جدول، مناسب‌تر از استفاده از نمودار است. جدول حالت مدار در جدول ۶-۵ لیست شده و مستقیماً از نمودار حالت بدست آمده است. الگوریتم کاهش یک جدول حالت کامل در اینجا بدون اثبات ارائه شده است: یعنی "دو حالت را معادل گوییم اگر برای هر عنصر از مجموعه ورودی‌ها، دقیقاً خروجی یکسانی را تولید کنند و مدار را به همان حالت یا به یک حالت معادل ببرند". وقتی دو حالت معادل هستند، بدون تغییر رابطه ورودی- خروجی، می‌توان یکی از آن دو را حذف کرد.

حال این الگوریتم را به جدول ۶-۵ اعمال می‌نماییم. با مراجعه و پیشروی در جدول حالت به دنبال دو حالتی می‌گردیم که به حالت بعدی یکسانی بروند و به ازاء هر دو ترکیب، ورودی- خروجی یکسانی داشته باشند. حالت g و e این چنین حالت‌هایی هستند. هر دو به حالات $a$ و f رفته و خروجی‌های 0 و 1 را برای $x = 0$ و $x = 1$ تولید می‌کنند. بنابراین حالات g و e معادلند و یکی از این حالات قابل حذف است. روال حذف یک حالت و جایگزینی با معادلش در جدول ۷-۵ ملاحظه می‌شود. سطری که حالت فعلی g را دارد حذف و حالت g در ستون حالت بعدی با e جایگزین می‌گردد. اکنون حالت فعلی f دارای حالت بعدی e و f و خروجی‌های 0 و 1 به ترتیب برای $x = 0$ و $x = 1$

**جدول ۷-۵** . کاهش جدول حالت

| حالت فعلی | حالت بعدی | | خروجی | |
|---|---|---|---|---|
| | $x = 0$ | $x = 1$ | $x = 0$ | $x = 1$ |
| $a$ | $a$ | $b$ | 0 | 0 |
| $b$ | $c$ | $d$ | 0 | 0 |
| $c$ | $a$ | $d$ | 0 | 0 |
| $d$ | $e$ | $f$ | 0 | 1 |
| $e$ | $a$ | $f$ | 0 | 1 |
| $f$ | $e$ | $f$ | 0 | 1 |

**جدول ۸-۵** . جدول حالت کاهش یافته

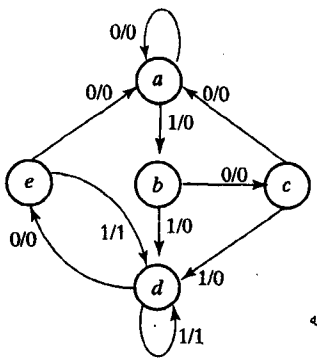
| حالت فعلی | حالت بعدی | | خروجی | |
|---|---|---|---|---|
| | $x = 0$ | $x = 1$ | $x = 0$ | $x = 1$ |
| $a$ | $a$ | $b$ | 0 | 0 |
| $b$ | $c$ | $d$ | 0 | 0 |
| $c$ | $a$ | $d$ | 0 | 0 |
| $d$ | $e$ | $d$ | 0 | 1 |
| $e$ | $a$ | $d$ | 0 | 1 |

می‌باشد. حالات بعدی و خروجی‌های یکسانی در سطر مربوطه به حالت فعلی d ملاحظه می‌شود. بنابراین حالات f و d معادلند و می‌توان حالت f را حذف و آن را با d جایگزین کرد. جدول کاهش یافته نهایی در جدول ۸-۵ ملاحظه می‌گردد. نمودار حالت برای جدول کاهش یافته فقط دارای پنج حالت بوده و در شکل ۲۳-۵ نشان داده شده است. این نمودار حالت مشخصات اولیه ورودی- خروجی را داراست و رشته خروجی موردنظر را برای هر رشته ورودی فراهم می‌سازد. لیست حاصل از نمودار حالت شکل ۲۳-۵ مربوط به رشته ورودی به کار رفته قبلی است. توجه کنید که گرچه رشته حالت متفاوت است، ولی رشته خروجی یکسانی تولید شده است.

| حالت | $a$ | $a$ | $b$ | $c$ | $d$ | $e$ | $d$ | $d$ | $e$ | $d$ | $e$ | $a$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ورودی | | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| خروجی | | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |

در حقیقت اگر ما g را با e و f را با d جایگزین کنیم، این رشته همان رشته شکل ۲۱-۵ است.

یافتن حالات معادل با چک کردن هر جفت حالت با روالی که از جدول ایجاب استفاده می‌کند صورت می‌گیرد. جدول ایجاب متشکل از مربعاتی است که هر یک از این مربعات به یک جفت حالت معادل تعلق دارد. با به‌کارگیری صحیح جدول، می‌توان همه حالات معادل را در جدول حالت بدست آورد. کاربرد جدول ایجاب در کاهش تعداد حالات در جدول حالت در بخش ۵-۹ نشان داده شده است. مدار ترتیبی این مثال تعداد حالات را از هفت به پنج کاهش داد. به طورکلی، کاهش در تعداد حالات در جدول حالت ممکن است به مداری با عناصر کمتر منجر شود. با این وجود کاهش حالات در جدول حالت، تضمینی در کاهش تعداد فلیپ فلاپ یا تعداد گیت‌ها ندارد.




**شکل ۲۳-۵** . نمودار حالت کاهش یافته

جدول ۹-۵. سه تخصیص ممکن دودویی حالت

| حالت | تخصیص 1 دودویی | تخصیص 2 کد گری | تخصیص 3 یک بارز |
|---|---|---|---|
| *a* | 000 | 000 | 00001 |
| *b* | 001 | 001 | 00010 |
| *c* | 010 | 011 | 00100 |
| *d* | 011 | 010 | 01000 |
| *e* | 100 | 110 | 10000 |

## تخصیص حالت

هنگام طراحی یک مدار ترتیبی با عناصر فیزیکی، لازم است به حالات، مقادیر دودویی کد شده‌ای تخصیص یابد. برای مداری با m حالت، باید کدی n بیتی اختیار شود که در آن $2^n \geq m$ است. مثلاً، با سه بیت می‌توان کدی را برای هشت حالت از 000 تا 111 تعریف کرد. اگر جدول حالت شکل ۶-۵ به کار برود، باید مقادیر دودویی را به هفت حالت اختصاص دهیم؛ تنها حالت باقیمانده هم مورد استفاده نیست. اگر جدول حالت ۸-۵ به کار برود، تنها پنج حالت به تخصیص دودویی نیاز دارند و در این مورد سه حالت به کار نرفته وجود خواهد داشت. در طراحی با حالات به کار نرفته به عنوان حالات بی‌اهمیت برخورد می‌شود. چون حالات بی‌اهمیت معمولاً در یافتن مدار ساده‌تر مفیدند، به نظر می‌رسد مدار پنج حالته گیت‌های ترکیبی کمتری از مدار هفت حالته داشته باشد.

ساده‌ترین راه کد دادن به پنج حالت استفاده از پنج عدد صحیح در شمارش دودویی در تخصیص 1 مطابق جدول ۹-۵ است. تخصیص مناسب دیگر کدگری در تخصیص 2 است. در این ستون به هنگام تغییر از یک عدد به یک عدد دیگر تنها یک بیت تغییر می‌یابد. این کد استقرار توابع بول را در نقشه کارنو برای ساده‌سازی آسان‌تر می‌کند. تخصیص بعدی که در طراحی سیستم‌های کنترل به کار می‌رود تخصیص 1 بارز است. در این آرایش تعداد بیت‌ها با تعداد حالات مدار برابر است. در هر زمان مشخص، تنها یک بیت برابر 1 و بقیه بیت‌ها در 0 نگهداری می‌شوند. در این تخصیص برای هر حالت یک فلیپ فلاپ به کار می‌رود.

جدول ۱۰-۵، جدول حالت کاهش یافته با به‌کارگیری تخصیص 1 به سمبل‌های حرفی حالات است. یک تخصیص متفاوت دیگر، جدول حالت دیگری با مقادیر دودویی متفاوت را نتیجه می‌دهد. فرم دودویی جدول حالت برای بخش ترکیبی مدار ترتیبی مورد استفاده است. پیچیدگی مدار ترکیبی به انتخاب تخصیص حالت دودویی بستگی دارد.

جدول ۱۰-۵. جدول حالت کاهش یافته با تخصیص 1، دودویی

| حالت فعلی | حالت بعدی | | خروجی | |
|---|---|---|---|---|
| | $x = 0$ | $x = 1$ | $x = 0$ | $x = 1$ |
| 000 | 000 | 001 | 0 | 0 |
| 001 | 010 | 011 | 0 | 0 |
| 010 | 000 | 011 | 0 | 0 |
| 011 | 100 | 011 | 0 | 1 |
| 100 | 000 | 011 | 0 | 1 |

گاهی نام جدول گذر در عوض جدول حالت به کار می‌رود. معمولاً نام جدول حالت در مواردی مورد استفاده قرار می‌گیرد که از حروف برای معرفی حالت استفاده بشود. در این کتاب، برای هر دو نوع جدول نام مشترکی را به کار خواهیم برد.

## ۷-۵ روش طراحی

طراحی یک مدار ترتیبی ساعت‌دار با بیان مجموعه‌ای از مشخصات و ویژگی‌های شروع شده و با نمودار منطقی یا لیستی از توابع بول که از آنها نمودار منطقی حاصل می‌شود پایان می‌یابد. برخلاف یک مدار ترکیبی که کلاً با جدول درستی مشخص می‌گردد، مدار ترتیبی برای مشخصات خود نیاز به جدول حالت دارد. اولین قدم در طراحی مدارهای ترتیبی تهیه جدول حالت یا نمایش معادلی از آن، مانند نمودار حالت است.

یک مدار ترتیبی همزمان، از فلیپ فلاپ‌ها و گیت‌های ترکیبی ساخته می‌شود. طراحی مدار شامل انتخاب فلیپ فلاپ‌ها و سپس یافتن ساختار گیتی بخش ترکیبی آن است تا مداری که مشخصات موردنظر را داراست حاصل گردد. تعداد فلیپ فلاپ‌ها با توجه به تعداد حالات لازم در مدار بدست می‌آید. مدار ترکیبی هم با استفاده از جدول حالات و ارزیابی معادلات ورودی و خروجی آن معین می‌گردد. در واقع، به محض این که نوع و تعداد فلیپ فلاپ‌ها معین شد، روند طراحی از مدار ترتیبی به مدار ترکیبی تبدیل می‌شود. بدین ترتیب تکنیک‌های مدار ترکیبی را می‌توان به کار گرفت.

روال طراحی مدارهای ترتیبی همزمان را می‌توان به صورت زیر خلاصه کرد.

۱- با توجه به عبارات و مشخصات مربوط به عملکرد موردنظر، یک نمودار حالت برای مدار بدست آورید.

۲- در صورت لزوم تعداد حالات را کاهش دهید.

۳- به حالات مقادیر دودویی را تخصیص دهید.

۴- جدول حالت کد شده دودویی را بدست آورید.

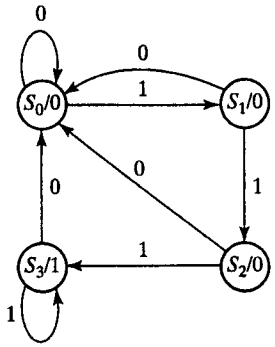
۵- نوع فلیپ فلاپ‌های به کار رفته را مشخص کنید.

۶- معادلات ساده شده ورودی فلیپ فلاپ‌ها و نیز معادلات خروجی را بدست آورید.

۷- نمودار منطقی را ترسیم کنید.

در مشخصاتی که رفتار مدار بیان می‌شود معمولاً فرض بر این است که خواننده با مفهوم منطق دیجیتال آشنایی دارد. در رسیدن به یک تفسیر صحیح از مشخصات مدار، تجربه و هوشمندی طراح نقش عمده‌ای دارد، زیرا توصیف لفظی ممکن است غیرکامل و غیرواقعی باشد. به محض دستیابی به چنین مشخصات و نمودار حالت، می‌توان از روش‌های شناخته شده طراحی برای تکمیل آن استفاده کرد. گرچه روال‌های ویژه‌ای در کاهش حالت و نیز تخصیص آن موجود است، ولی کمتر به وسیله طراحان با تجربه مورد استفاده قرار می‌گیرند.

مراحل ۴ تا ۷ در طرحی را می‌توان با الگوریتم‌های موجود پیاده‌سازی کرد و بنابراین قابل




**شکل ۲۴-۵**. نمودار حالت برای آشکارساز رشته

خودکارسازی است. بخشی از طراحی که روالی منظم را دنبال کند، سنتز یا ترکیب و یا ادغام می‌گویند.

اولین بخش، پرچالش‌ترین بخش طراحی است. ما در اینجا مثال ساده‌ای را ارائه خواهیم کرد تا نشان دهیم چگونه یک نمودار حالت از بیان لفظی بدست می‌آید.

می‌خواهیم مداری طراحی کنیم که به کمک آن سه 1 (و یا بیشتر) متوالی در یک رشته بیتی وارده از طریق یک خط ورودی شناسایی شوند. نمودار حالت مدار در شکل ۲۴-۵ نشان داده شده است. این نمودار با شروع از $S_0$ بدست آمده است. اگر ورودی 0 باشد، مدار در همان حالت قبل باقی می‌ماند، ولی اگر ورودی 1 شود، مدار به حالت $S_1$ می‌رود تا دریافت یک 1 را مشخص کند. اگر ورودی بعدی هم 1 باشد، تغییر به حالت $S_2$ به معنی دریافت دو 1 متوالی است، ولی اگر ورودی 0 باشد، به حالت قبلی $S_0$ باز می‌گردد. سومین 1 متوالی، مدار را به $S_3$ می‌برد. اگر 1های بیشتری تشخیص داده شود، مدار همچنان در $S_3$ باقی می‌ماند. هر ورودی 0 مدار را در این حالت به $S_0$ باز می‌گرداند. و به این ترتیب در ازاء 1های متوالی بیشتر مدار در همان $S_3$ باقی خواهد ماند. این مدار ترتیبی از نوع مدل مور است زیرا وقتی مدار در حالت $S_3$ است خروجی 1 و در غیر این صورت 0 است.

### سنتز با فلیپ فلاپ‌های D

به محض این که نمودار حالت بدست آمد، بقیه طراحی از رویه سنتز سرراستی استفاده می‌کند. در واقع ما می‌توانیم مدار را به کمک توصیف HDL نمودار حالت و ابزارهای مناسب HDL برای یافتن یک netlist، طراحی کنیم. لازم به یادآوری است که توصیف HDL نمودار حالت مشابه با مثال ۶-۵ HDL در بخش ۵-۵ است. برای طراحی دستی مدار، باید کدهای دودویی را به حالات تخصیص دهیم و جدول حالت را لیست نماییم. این کار در جدول ۱۱-۵ انجام شده است. جدول از نمودار حالت شکل ۲۴-۵ با تخصیص مستقیم دودویی بدست می‌آید. برای نمایش چهار حالت از دو فلیپ فلاپ D استفاده می‌کنیم و آنها را با A و B نامگذاری می‌نماییم. در مدار یک ورودی x و یک خروجی y وجود دارد. معادله مشخصه فلیپ فلاپ D برابر $Q(t+1) = D_Q$ است و به این معنی است که مقادیر حالت بعدی در جدول حالت همان حالت ورودی D در فلیپ فلاپ است. معادلات ورودی فلیپ فلاپ

**جدول ۱۱-۵** . جدول حالت برای آشکارساز رشته

| حالت فعلی | | ورودی | حالت بعدی | | خروجی |
|---|---|---|---|---|---|
| A | B | x | A | B | y |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

مستقیماً از ستون‌های حالت بعدی A و B بدست آمده و برحسب مینترم به صورت زیر بیان می‌شوند.

$$A(t+1) = D_A(A, B, x) = \Sigma(3, 5, 7)$$
$$B(t+1) = D_B(A, B, x) = \Sigma(1, 5, 7)$$
$$y(A, B, x) = \Sigma(6, 7)$$

که در آن A و B حالت فعلی فلیپ فلاپ‌های A و B ، x ورودی، و $D_A$ و $D_B$ معادلات ورودی‌اند. مینترم‌های خروجی y از ستون خروجی در جدول حالت بدست می‌آیند.

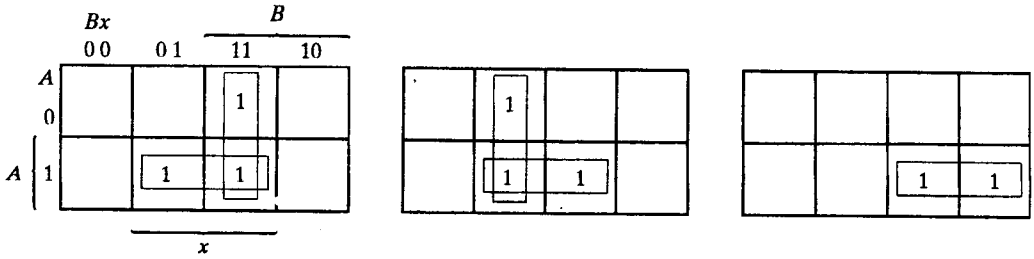
معادلات بولی که به کمک نقشه کارنو ساده شده‌اند در شکل ۲۵-۵ دیده می‌شوند. این معادلات عبارتند از:

$$D_A = Ax + Bx$$
$$D_B = Ax + B'x$$
$$y = AB$$

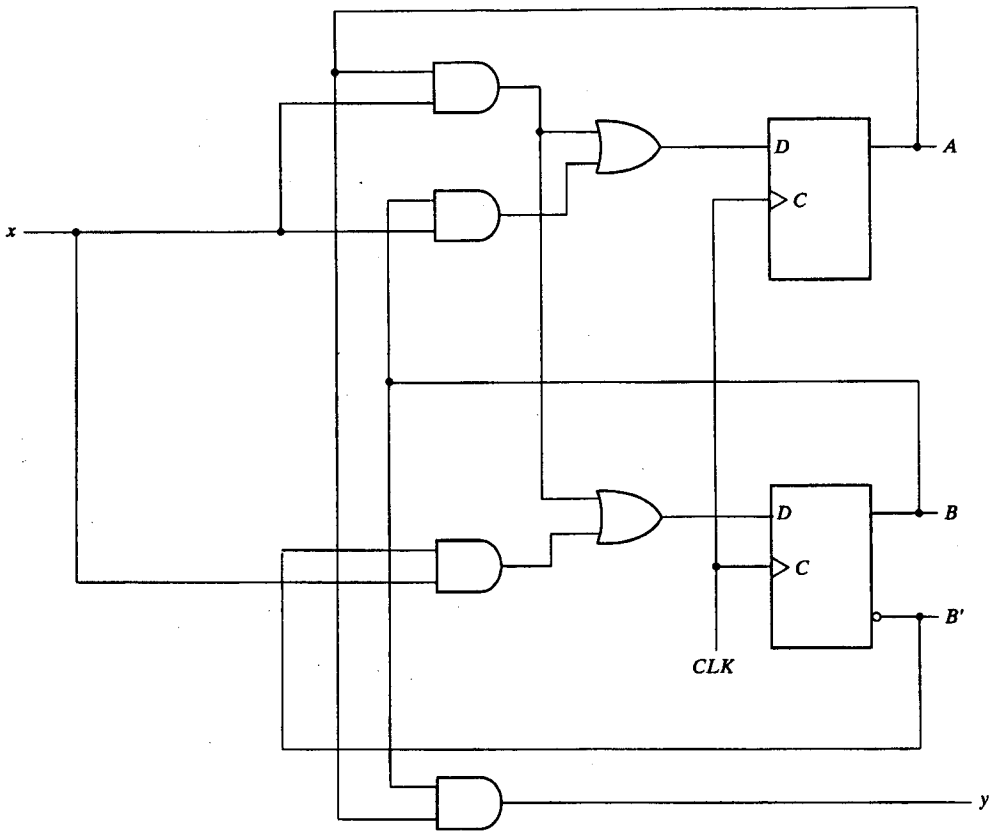
نمودار منطقی مدار ترتیبی در شکل ۲۶-۵ ترسیم شده است.

## جداول تحریک

طراحی یک مدار ترتیبی با فلیپ فلاپ‌هایی به جز نوع D پیچیده است زیرا معادلات ورودی برای مدار باید به طور غیرمستقیم از جدول حالت بدست آید. وقتی که فلیپ فلاپ‌های D به کار می‌روند،




**شکل ۲۵-۵** . نقشه‌ها برای آشکارساز رشته




**شکل ۲۶-۵**. نمودار منطقی آشکارساز رشته

معادلات ورودی مستقیماً از حالت بعدی بدست می‌آیند. این مطلب برای فلیپ فلاپ‌های JK و T چنین نیست. برای تعیین معادلات حالت این فلیپ فلاپ‌ها لازم است رابطه عملیاتی بین جدول حالت و معادلات ورودی را بدست آوریم.

جداول مشخصه فلیپ فلاپ‌ها که در جدول ۱-۵ مشاهده شد وقتی ورودی‌ها و حالات بعدی معلوم‌مند، مقدار حالت بعدی را فراهم می‌کنند. این جداول در تحلیل مدارهای ترتیبی و تعریف کار فلیپ فلاپ‌ها مفیدند. در حین فرآیند طراحی، ما معمولاً گذر از حالت فعلی به حالت بعدی را می‌دانیم و مایلیم وضعیت ورودی فلیپ فلاپ که موجب این گذر می‌گردد را پیدا کنیم. به این دلیل جدولی را بدست می‌آوریم که در آن برای هر تغییر مورد نظری در حالت، ورودی‌های لازم لیست شده باشند. چنین جدولی را جدول تحریک می‌گویند.

جدول ۱۲-۵ جداول تحریک را برای دو فلیپ فلاپ نشان می‌دهد. هر جدول ستونی برای حالت فعلی Q(t) ، حالت بعدی Q(t+1) و نیز ستونی برای هر ورودی دارد تا چگونگی وقوع گذر

**جدول ۱۲-۵**. جداول تحریک فلیپ فلاپ

| $Q(t)$ | $Q(t+1)$ | $T$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

(ب) $T$

| $Q(t)$ | $Q(t+1)$ | $J$ | $K$ |
|---|---|---|---|
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 1 |
| 1 | 1 | X | 0 |

(الف) $JK$

را نشان دهد. از حالت فعلی به حالت بعدی چهارگذر می‌تواند وجود داشته باشد. شرایط ورودی لازم برای هر چهارگذر از اطلاعات موجود در جدول مشخصه بدست می‌آید. سمبل X در جدول حالت بی‌اهمیت را نشان می‌دهد و به این معنی است که 1 یا 0 بودن ورودی اهمیتی ندارد.

جدول تحریک فلیپ فلاپ JK در بخش (الف) مشاهده می‌شود. وقتی که هر دو حالت فعلی و بعدی 0 باشند، ورودی J باید در 0 بماند ولی K می‌تواند 0 یا 1 باشد. به طور مشابه هرگاه هر دو حالت فعلی و بعدی 1 باشند، K در 0 ولی J می‌تواند 0 یا 1 باشد. اگر فلیپ فلاپ قرار باشد از حالت 0 به حالت 1گذر کند، J باید 1 باشد زیرا ورودی J فلیپ فلاپ را در 1 می‌نشاند. با این وجود K ممکن است 0 و یا 1 باشد. اگر $K = 0$ باشد، $J = 1$ فلیپ فلاپ را مطابق میل در 1 می‌نشاند؛ اگر $K = 1$ و $J = 1$ باشد خروجی فلیپ فلاپ متمم شده و از حالت 0 به 1 می‌رود. بنابراین K با علامت بی‌اهمیت X برای گذر از 0 به 1، علامت خورده است. برای گذر از حالت 1 به حالت 0، باید $K = 1$ باشد، زیرا ورودی K فلیپ فلاپ را پاک می‌کند. با این وجود ورودی J می‌تواند 0 یا 1 باشد، زیرا $J = 0$ تأثیری ندارد و $J = 1$ همراه با $K = 1$ فلیپ فلاپ را متمم می‌کند که حاصل آن گذر از 1 به 0 است.

جدول تحریک برای فلیپ فلاپ T در بخش (ب) ملاحظه می‌شود. با توجه به جدول مشخصه، می‌بینیم که وقتی $T = 1$ است، فلیپ فلاپ متمم می‌گردد ولی اگر $T = 0$ باشد، حالت فلیپ فلاپ بی‌تغییر باقی خواهد ماند. بنابراین اگر قرار است فلیپ فلاپ بی‌تغییر بماند، لازم است $T = 0$ باشد و اگر قرار باشد حالت فلیپ فلاپ متمم شود $T = 1$ خواهد بود.

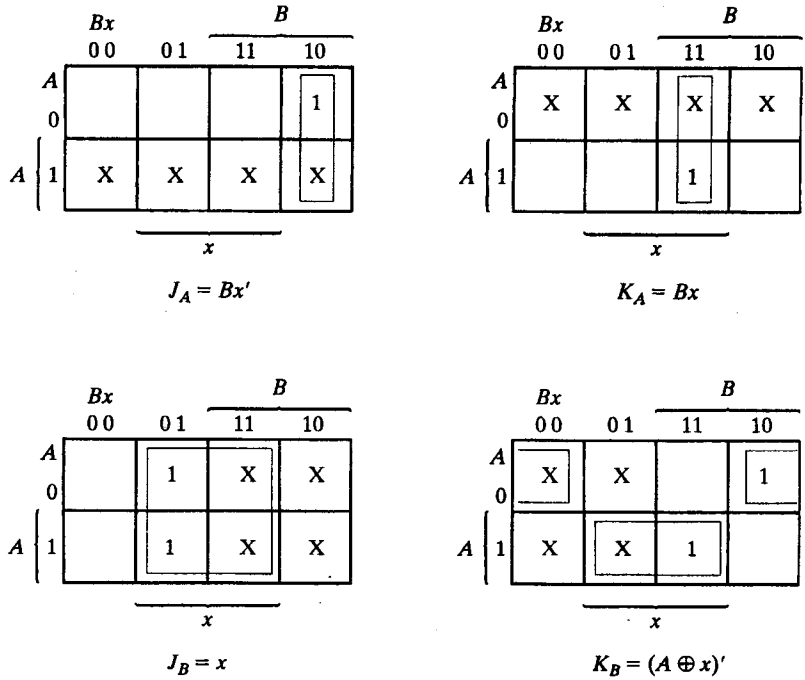
## سنتز با استفاده از فلیپ فلاپ‌های JK

روال سنتز برای مدار ترتیبی حاوی فلیپ فلاپ JK مانند فلیپ فلاپ‌های D است، با این تفاوت که معادلات ورودی باید از حالت فعلی به بعدی ارزیابی شوند، که این کار با توجه به جدول تحریک صورت می‌گیرد. برای تشریح روال، مدار ترتیبی معرفی شده با جدول ۱۳-۵ را طراحی می‌کنیم. علاوه بر ستون‌های حالت فعلی، ورودی و حالت خروجی در جدول حالت معمولی، این جدول وضعیت ورودی فلیپ فلاپ، که معادلات ورودی از آن بدست می‌آیند را نیز نشان می‌دهد. این ورودی‌ها از جدول حالت و به همراهی جدول تحریک فلیپ فلاپ J معین می‌گردند. مثلاً در اولین سطر جدول ۱۳-۵ برای فلیپ فلاپ A گذری از حالت فعلی 0 به حالت بعدی 0 داریم. در جدول ۱۲-۵ برای فلیپ فلاپ J، لازمه این گذر یا انتقال این است که $J = 0$ و K به صورت بی‌اهمیت باشد. بنابراین 0 و X در

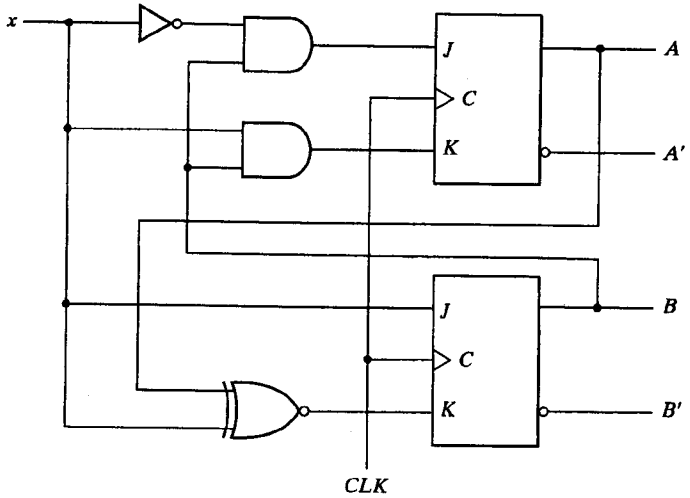
**جدول ۱۳-۵**. جدول حالت ورودی‌های فلیپ فلاپ JK

| حالت فعلی | | ورودی | حالت بعدی | | ورودی‌های فلیپ فلاپ | | | |
|---|---|---|---|---|---|---|---|---|
| A | B | x | A | B | $J_A$ | $K_A$ | $J_B$ | $K_B$ |
| 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | X |
| 0 | 0 | 1 | 0 | 1 | 0 | X | 1 | X |
| 0 | 1 | 0 | 1 | 0 | 1 | X | X | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | X | X | 0 |
| 1 | 0 | 0 | 1 | 0 | X | 0 | 0 | X |
| 1 | 0 | 1 | 1 | 1 | X | 0 | 1 | X |
| 1 | 1 | 0 | 1 | 1 | X | 0 | X | 0 |
| 1 | 1 | 1 | 0 | 0 | X | 1 | X | 1 |

اولین سطر زیر $J_A$ و $K_A$ وارد می‌شوند. چون در سطر اول فلیپ فلاپ B هم از حالت فعلی 0 به حالت بعدی 0 می‌رود، 0 و X به ترتیب زیر $J_B$ و $K_B$ وارد می‌شوند. سطر دوم جدول یک گذر از حالت فعلی 0 به حالت بعدی 1 در فلیپ فلاپ B را نشان می‌دهد. با توجه به جدول حالت، می‌بینیم که گذر از 0 به 1 لازم می‌دارد که $J = 1$ و $K = X$ باشد، بنابراین در سطر دوم و زیر ستون $J_B$ و $K_B$، مقادیر 1 و X وارد می‌شوند. این فرآیند در جدول برای هر سطر و برای هر فلیپ فلاپ ادامه می‌یابد و هر وضعیت ورودی از جدول تحریک در سطر مناسبی برای هر فلیپ فلاپ کپی می‌شود.




**شکل ۲۷-۵**. نقشه‌ها برای معادلات ورودی J و K



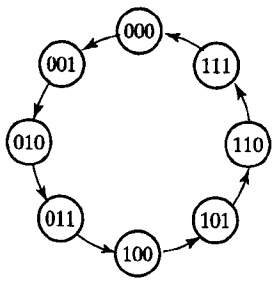

**شکل ۲۸-۵** . نمودار منطقی برای مدار ترتیبی با فلیپ فلاپ‌های JK

ورودی‌های فلیپ فلاپ‌ها در جدول ۱۳-۵، در واقع جدول درستی را برای معادلات ورودی به عنوان تابعی از حالت فعلی A و B و ورودی x به نمایش می‌گذارند. معادلات ورودی در نقشه‌های شکل ۲۷-۵ ساده شده‌اند. مقادیر حالت بعدی در ضمن ساده‌سازی به کار نرفته‌اند زیرا معادلات ورودی فقط تابعی از حالات فعلی و ورودی بیرونی می‌باشند. به مزیت استفاده از فلیپ فلاپ‌های JK در طراحی توجه نمایید. چون تعداد قابل توجهی حالت بی‌اهمیت در جدول وجود دارد، مدار ترکیبی برای معادلات ورودی ساده‌تر خواهد بود، زیرا مینترم‌های بی‌اهمیت معمولاً به ساده‌سازی کمک می‌کنند. اگر حالات بی‌استفاده‌ای در جدول حالت وجود داشته باشد، حالات بی‌اهمیت دیگری نیز در نقشه وجود خواهد داشت.

چهار معادله ورودی برای دو فلیپ فلاپ JK در زیر نقشه‌های شکل ۲۷-۵ لیست شده‌اند. نمودار منطقی مدار ترتیبی در شکل ۲۸-۵ مشاهده می‌شود.

## سنتز با استفاده از فلیپ فلاپ‌های T

سنتز مدارهای ترتیبی با فلیپ فلاپ‌های T به وسیله طراحی یک شمارنده دودویی نشان داده خواهد شد. یک شمارنده دودویی n بیت متشکل از n فلیپ فلاپ است که می‌تواند از 0 تا $2^n-1$ را بشمارد. نمودار حالت یک شمارنده 3 بیت در شکل ۲۹-۵ دیده می‌شود. با توجه به حالات دودویی در




**شکل ۲۹-۵** . نمودار حالت شمارنده دودویی 3 بیت

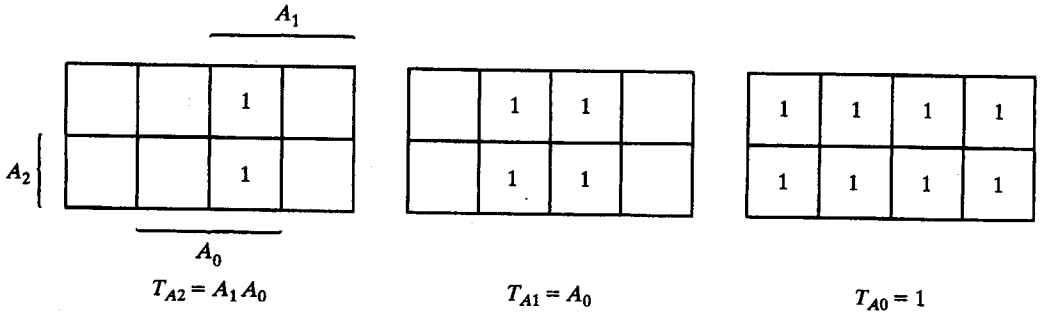
**جدول ۱۴-۵**. جدول حالت برای شمارنده 3 بیت

| حالت فعلی | | | حالت بعدی | | | ورودی‌های فلیپ فلاپ | | |
|---|---|---|---|---|---|---|---|---|
| $A_2$ | $A_1$ | $A_0$ | $A_2$ | $A$ | $A_0$ | $T_{A2}$ | $T_{A1}$ | $T_{A0}$ |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |

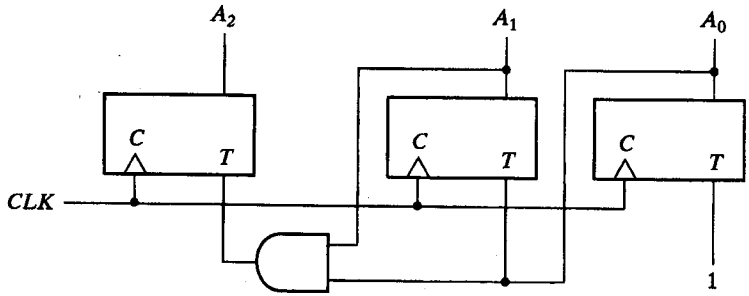
داخل دوایر، خروجی فلیپ فلاپ‌ها پس از 111 با بازگشت به 000 تکرار می‌شوند. خطوط جهت‌دار بین دوایر با مقادیر ورودی و خروجی علامت‌زنی نشده‌اند. به خاطر بسپارید که گذار حالات در مدارهای ترتیبی ساعت‌دار در لبه پالس ساعت به عنوان یک متغیر ورودی جداگانه در نمودار حالت یا جدول حالت ظاهر نمی‌گردد. از این نظر، نمودار حالت شمارنده نیازی به نشان دادن مقادیر ورودی و خروجی در امتداد خطوط جهت‌دار ندارد. در واقع تنها ورودی به مدار همان ساعت و خروجی‌ها هم حالات فعلی فلیپ فلاپ‌ها هستند. حالت بعدی شمارنده کلاً به حالت فعلی‌اش بستگی دارد و گذار حالت هر بار که پالس ساعت تغییر وضعیت دهد، رخ خواهد داد.

جدول ۱۴-۵، جدول حالت برای یک شمارنده دودویی 3 بیت است. سه فلیپ فلاپ با $A_2$ و $A_1$ و $A_0$ نام‌گذاری شده‌اند. شمارنده‌های دودویی غالباً با فلیپ فلاپ‌های T ساخته می‌شوند زیرا دارای خاصیت متمم‌سازی هستند. جدول تحریک برای ورودی‌های T از جدول تحریک فلیپ فلاپ T و وارسی گذر از حالت فعلی به حالت بعدی حاصل می‌گردد. به منظور تشریح، ورودی‌های فلیپ فلاپ را در سطر 001 ملاحظه کنید. حالت فعلی 001 و حالت بعدی 010 است که حالت بعدی مربوط به رشته شمارش است. از مقایسه این دو شماره مشاهده می‌شود که $A_2$ از 0 به 0 می‌رود، بنابراین $T_{A2}$ با 0 نشان‌گذاری شده است، زیرا فیلپ فلاپ $A_2$ نباید به هنگام رخداد پالس ساعت عوض شود. $A_1$ از 0 به 1 می‌رود بنابراین $T_{A1}$ با 1 علامت زده شده است زیرا فلیپ فلاپ باید با لبه پالس ساعت متمم شود. به طور مشابه $A_0$ از 1 به 0 می‌رود که بیانگر متمم شدن است و بنابراین $T_{A0}$ با 1 نشان‌گذاری شده است. آخرین ردیف با حالت فعلی 111 با اولین مقدار شمارش یعنی 000 مقایسه می‌شود که در این هنگام حالت بعدی است. رفتن از تمام 1 به تمام 0 لازم می‌دارد تا هر سه فلیپ فلاپ متمم گردد.

معادلات ورودی فلیپ‌فلاپ در نقشه شکل ۳۰-۵ ساده شده است. توجه کنید که در $T_{A0}$ همه مینترم‌ها 1 هستند زیراکم‌ارزش‌ترین بیت در هر شمارش متمم می‌گردد. تابع بولی که در آن همه مینترم‌ها لحاظ شود برابر مقدار ثابت 1 است. معادلات ورودی نوشته شده در زیر هر نقشه بخش ترکیبی شمارنده را مشخص می‌نماید. با استفاده از این توابع و سه فلیپ فلاپ، نمودار منطقی شمارنده مطابق شکل ۳۱-۵ خواهد بود.




**شکل ۳۰-۵**. نقشه برای شمارنده دودویی




**شکل ۳۱-۵**. نمودار منطقی شمارنده دودویی 3 بیت

## مسائل

۵-۱ لچ D شکل ۶-۵ با چهار گیت NAND و یک وارونگر ساخته شده است. سه راه دیگر ساخت آن را در زیر ملاحظه کنید. در هر حالت، نمودار منطقی را رسم و عملکرد مدار را اصلاح نمایید.
(الف) گیت‌های NOR برای بخش لچ SR و گیت‌های AND برای دو گیت دیگر ممکن است به یک وارونگر هم نیاز داشته باشید.
(ب) از گیت‌های NOR برای هر چهار گیت استفاده نمایید. ممکن است وارونگر هم لازم داشته باشد.
(پ) فقط از چهار گیت NAND استفاده کنید (بدون وارونگر) این کار را می‌توانید با اتصال خروجی گیت فوقانی در شکل ۶-۵ که به ورودی گیت پایینی می‌رود، در عوض خروجی وارونگر انجام دهید. (خروجی گیت بالایی به لچ SR می‌رود).

۵-۲ با فلیپ فلاپ D یک فلیپ فلاپ JK بسازید. از یک مولتی پلکسر و یک وارونگر استفاده کنید.

۵-۳ نشان دهید که معادله مشخصه برای خروجی متمم شده فلیپ فلاپ JK مطابق زیر است.

$$Q'(t+1) = J'Q' + KQ$$

۵-۴ یک فلیپ فلاپ PN چهار نوع عمل را داراست: پاک‌کردن به 0، بدون تغییر، متمم و نشاندن به 1. هر یک از موارد فوق به ترتیب با PN برابر با 00، 01، 10 و 11 انجام می‌شود.

(الف) جدول مشخصه را پیاده کنید. (ب) معادله مشخصه را بدست آورید.

(پ) جدول تحریک را پیاده نمایید.

(ت) نشان دهید که می‌توان فلیپ فلاپ PN را به فلیپ فلاپ D تبدیل کرد.

۵-۵ تفاوت جدول درستی، جدول حالت، جدول مشخصه و جدول تحریک را بیان کنید. همچنین تفاوت تابع بول، معادله حالت، معادله مشخصه و معادله ورودی فلیپ فلاپ را توضیح دهید.

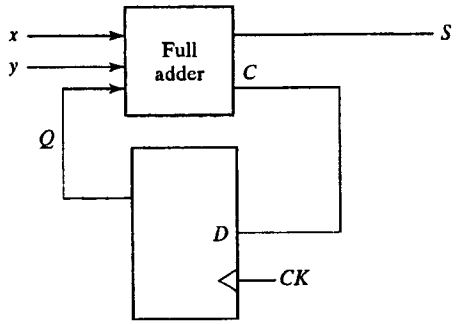
۵-۶ یک مدار ترتیبی با دو فلیپ فلاپ A و B از نوع D؛ دو ورودی x و y، و یک خروجی z با معادلات حالت بعدی و خروجی مشخص شده است.

$$A(t+1) = x'y + xA$$
$$B(t+1) = x'B + xA$$
$$z = B$$

(الف) نمودار منطقی مدار را رسم کنید. (ب) جدول حالت را برای مدار ترتیبی لیست کنید.

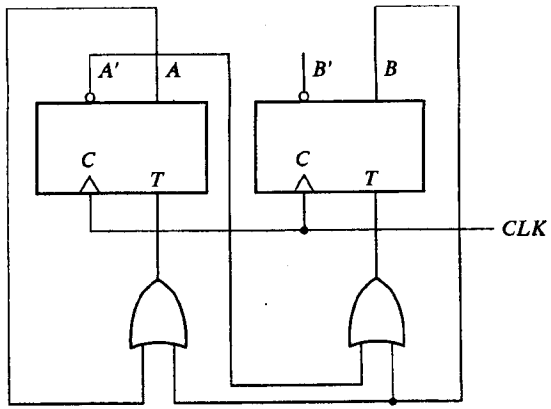
(پ) نمودار حالت مربوطه را بکشید.

۵-۷ یک مدار ترتیبی دارای یک فلیپ فلاپ Q، دو ورودی x و y و یک خروجی S است. این مدار دارای یک جمع‌کننده کامل متصل به یک فلیپ فلاپ D است، شکل (م ۵-۷). جدول حالت و نمودار حالت مدار ترتیبی را بدست آورید.




شکل (م ۵-۷)

۵-۸ جدول و نمودار حالت مدار ترتیبی شکل (م ۵-۸) را بدست آورید. عملکرد مدار را توضیح دهید.




شکل (م ۵-۸)

۵-۹ یک مدار ترتیبی دو فلیپ فلاپ A و B از نوع JK و یک ورودی x دارد. مدار با معادلات ورودی فلیپ فلاپ زیر تعریف شده است.

$$J_A = x \qquad K_A = B'$$
$$J_B = x \qquad K_B = A$$

(الف) معادلات حالت A(t+1) و B(t+1) را با جایگزینی معادلات ورودی در J و K بدست آورید.
(ب) نمودار حالت مدار را رسم کنید.

۵-۱۰ یک مدار ترتیبی دارای دو فلیپ فلاپ A و B از نوع JK، دو ورودی x و y و یک خروجی z است. معادلات ورودی فلیپ فلاپ و معادله خروجی عبارتند از:

$$J_A = Bx + B'y' \qquad K_A = B'xy'$$
$$J_B = A'x \qquad K_B = A + xy'$$
$$z = Ax'y' + Bx'y'$$

(الف) نمودار منطقی مدار را رسم نمایید. (ب) جدول حالت را پیاده کنید.
(پ) معادلات حالت را برای A و B بدست آورید.

۵-۱۱ با شروع از 00 در شکل ۵-۱۶، گذار حالت و رشته خروجی حاصل از یک رشته ورودی 0101101101110111110 را بدست آورید.

۵-۱۲ تعداد حالات را در جدول حالت زیر کاهش داده و جدول کاهش یافته را نشان دهید.

| حالت فعلی | حالت بعدی | | خروجی | |
|---|---|---|---|---|
| | $x = 0$ | $x = 1$ | $x = 0$ | $x = 1$ |
| $a$ | $f$ | $b$ | 0 | 0 |
| $b$ | $d$ | $c$ | 0 | 0 |
| $c$ | $f$ | $e$ | 0 | 0 |
| $d$ | $g$ | $a$ | 1 | 0 |
| $e$ | $d$ | $c$ | 0 | 0 |
| $f$ | $f$ | $b$ | 1 | 1 |
| $g$ | $g$ | $h$ | 0 | 1 |
| $h$ | $g$ | $a$ | 1 | 0 |

۵-۱۳ با شروع از حالت $a$ و رشته ورودی 01110010011، رشته خروجی را برای موارد زیر بدست آورید.
(الف) جدول حالت مسئله قبل.
(ب) جدول حالت کاهش یافته در مسئله قبل. نشان دهید که برای هر دو رشته خروجی یکسانی بدست می‌آید.

۵-۱۴ تخصیص دودویی 2 را از جدول ۵-۹ در حالات جدول ۵-۸ جایگزین کنید و جدول حالت دودویی را بدست آورید.

۵-۱۵ یک جدول حالت برای فلیپ فلاپ JK با Q به عنوان حالت فعلی و حالت بعدی و ورودی‌های J و K لیست نمایید. مدار ترتیبی حاصل از جدول حالت را طراحی کنید و نشان دهید که معادل با شکل ۵-۱۲ (الف) است.

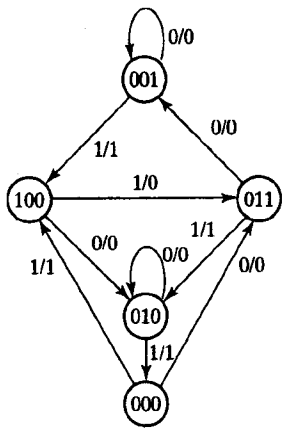
۵-۱۶ یک مدار ترتیبی با دو فلیپ فلاپ A و B از نوع D و یک ورودی x طراحی نمایید. وقتی x = 0 است، حالت مدار بدون تغییر باقی بماند. وقتی x = 1 است مدار وارد حالات 00، 01، 11 و 10 و بازگشت به 00 شده و کار را تکرار کند.

۵-۱۷ یک متمم 2 ساز یک ورودی، یک خروجی بسازید. مدار رشته‌ای از بیت‌ها را دریافت کرده و متمم 2 آن را در خروجی تولید می‌نماید. مدار قابل بازنشانی غیرهمزمان است تا عملیات را شروع و پایان دهد.

۵-۱۸ یک مدار ترتیبی با دو فلیپ فلاپ A و B نوع JK و دو ورودی E و x طراحی نمایید. اگر E = 0 باشد، مدار بدون توجه به x در حالت فعلی خود می‌ماند. وقتی E = 1 و x = 1 است وارد حالت 00، 01، 10 و 11 و بازگشت به 00 گشته و کار را تکرار کند. وقتی E = 1 و x = 0 است مدار وارد حالات 00، 11، 10 و 01 و به 00 رفته و عمل را تکرار نماید.

۵-۱۹ یک مدار ترتیبی دارای سه فلیپ فلاپ A، B و C؛ یک ورودی x و یک خروجی y است. نمودار حالت در شکل (م ۱۹-۵) مشاهده می‌گردد. قرار است مدار با استفاده از حالات بی‌اهمیت طراحی گردد. مدار طراحی شده را تحلیل کنید تا اثر حالات بی‌اهمیت ملاحظه شود.

(الف) از فلیپ فلاپ D استفاده کنید. (ب) از فلیپ فلاپ JK استفاده نمایید.




شکل (م ۱۹-۵)

۵-۲۰ مدار ترتیبی نمودار حالت شکل (۱۹-۵) را با فلیپ فلاپ‌های T طراحی کنید.

۵-۲۱ تفاوت اصلی بین یک عبارت initial و یک عبارت always در Verilog HDL چیست؟

۵-۲۲ شکل موج تولید شده با عبارت initial زیر را رسم کنید.

```
initial
  begin
  w = 0;  #20 w = 1;  # 50 w = 0;  # 30 w = 1;  #10 w = 0;
  end
```

۵-۲۳ عبارات زیر را با فرض این که Reg A در ابتدا مقدار 30 را دارد ملاحظه نمایید.

(الف)
```
RegA = 125
RegB = RegA
```

(ب)
```
RegA <= 125
RegB <= RegA
```

پس از اجرا مقادیر Reg A و Reg B چه هستند.

۵-۲۴ یک توصیف رفتاری HDL برای فلیپ فلاپ D بنویسید. برای آن ورودی پیش تنظیم و پاک کردن پیش‌بینی نمایید (به شکل ۱۳-۱۱ مراجعه شود).

۵-۲۵ یک فلیپ فلاپ حساس به لبه مثبت خاص دو ورودی D1 و D2 و یک ورودی کنترل دارد که یکی از آن دو را انتخاب می‌کند. توصیف رفتاری HDL آن را بنویسید.

۵-۲۶ یک توصیف رفتاری HDL با استفاده از عبارت if-else مبتنی بر مقدار حالت فعلی، برای فلیپ‌فلاپ JK بنویسید.

۵-۲۷ توصیف HDL مثال ۵-۵ را با ترکیب گذر حالات و خروجی در یک بلوک always بازنویسی کنید.

۵-۲۸ مدار ترتیبی شکل ۵-۱۷ را شبیه‌سازی نمایید.

(الف) توصیف HDL نمودار حالت را بنویسید.

(ب) توصیف HDL نمودار مدار را بنویسید.

(پ) یک HDL محرک با رشته‌ای از ورودی‌های 00، 01، 11، 10 بنویسید. نشان دهید که پاسخ برای هر دو توصیف یکی است.

۵-۲۹ توصیف HDL یک شمارنده دودویی 2 بیت در شکل ۵-۲۰ نشان داده شد. از مدول محرک مثال ۵-۷ HDL استفاده کنید و نشان دهید که پاسخ خروجی مشابه شکل موج ۵-۲۱ است.

۵-۳۰ نمودار منطقی را برای مدار ترتیبی توصیف شده با مدول HDL زیر رسم کنید.

```
module Seqcrt (A,B,C,Q,CLK);
   input A,B,C,CLK;
   output Q;
   reg Q,E;
   always @ (posedge CLK)
      begin
     E <= A & B;
     Q <= E | C;
      end
endmodule
```

چه تغییر، اگر لازم است، باید در مدار لحاظ شود اگر دو عبارت آخر از تخصیص بلوکی به جای غیربلوکی استفاده نماید.

## مراجع


1. HAYES, J. P. 1993. *Introduction to Digital Logic Design.* Reading, MA: Addison-Wesley.

2. WAKERLY, J. F. 2000. *Digital Design: Principles and Practices,* 3rd ed. Upper Saddle River, NJ: Prentice Hall.

3. KATZ, R. H. 1994. *Contemporary Logic Design.* Upper Saddle River, NJ: Prentice Hall.

4. MANO, M. M. and C. R. KIME. 2000. *Logic and Computer Design Fundamentals,* 2nd ed. Upper Saddle River, NJ: Prentice Hall.

5. NELSON V. P., H. T. NAGLE, J. D. IRWIN, and B. D. CARROLL. 1995. *Digital Logic Circuit Analysis and Design.* Upper Saddle River, NJ: Prentice Hall.



# ثَبّات‌ها و شمارنده‌ها

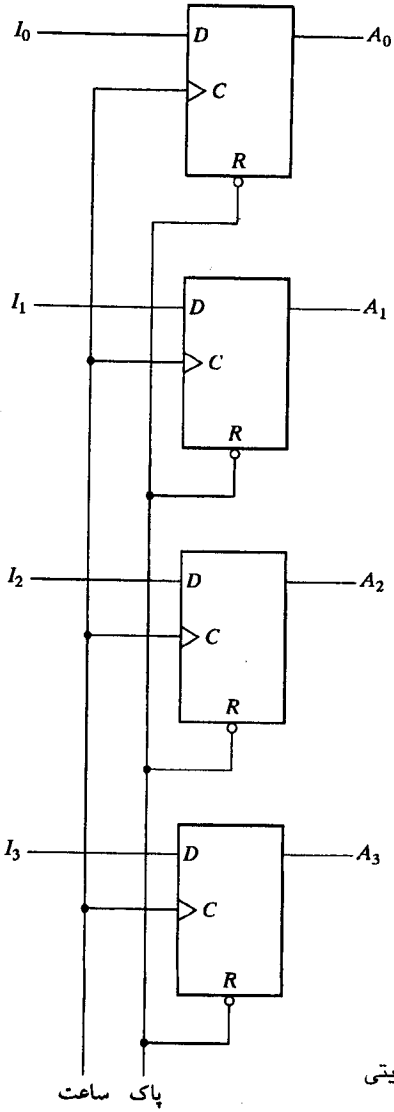
## ۶-۱ ثَبات‌ها

یک مدار ترتیبی ساعت‌دار متشکل از گروهی از فلیپ فلاپ‌ها و گیت‌های ترکیبی است که به منظور تشکیل یک مسیر پسخورد 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 پسخورد وجود نداشته باشد.) اما مداری با فلیپ فلاپ حتی در نبود گیت‌های ترکیبی باز هم یک مدار ترتیبی است. مدارهای حاوی فلیپ فلاپ‌ها معمولاً برحسب کارشان و نه با نام مدار ترتیبی دسته‌بندی می‌شوند. دو نوع از این مدارها ثبات‌ها و شمارنده‌ها هستند.

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

یک شمارنده اساساً یک ثبات 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. هر چند که شمارنده‌ها نوع خاصی از ثبات می‌باشند، معمولاً آنها را با نام‌های متفاوت از ثبات‌ها جدا می‌کنند.

انواع متنوعی از ثبات‌ها در بازار وجود دارند. ساده‌ترین ثبات، فقط از فلیپ فلاپ‌ها و بدون هرگونه گیتی تشکیل شده است. شکل ۶-۱ چنین ثباتی 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




**شکل ۱-۶.** ثبات 4 بیتی

اطلاعات دودویی در چهار ورودی به داخل ثبات 4 بیت منتقل می‌گردند. می‌توان هر لحظه چهار خروجی را نمونه‌برداری کرده و اطلاعات دودویی ذخیره شده در ثبات را بدست آورد. ورودی پاک به ورودی باز نشان (R) همه فلیپ فلاپ‌ها می‌رود. وقتی این ورودی به 0 رود، همه فلیپ فلاپ‌ها به طور غیرهمزمان بازنشانی (0) می‌شوند. ورودی پاک کردن برای 0کردن ثبات قبل از عمل ساعت‌زنی مفید است. در حین عمل معمول ساعت زنی، ورودی‌های R باید در منطق 1 قرارگیرند. توجه کنید که برای 0 کردن همه حالات در یک ثبات، می‌توان از پاک کردن ، یا بازنشانی استفاده کرد.

### ثبات با بار شدن موازی

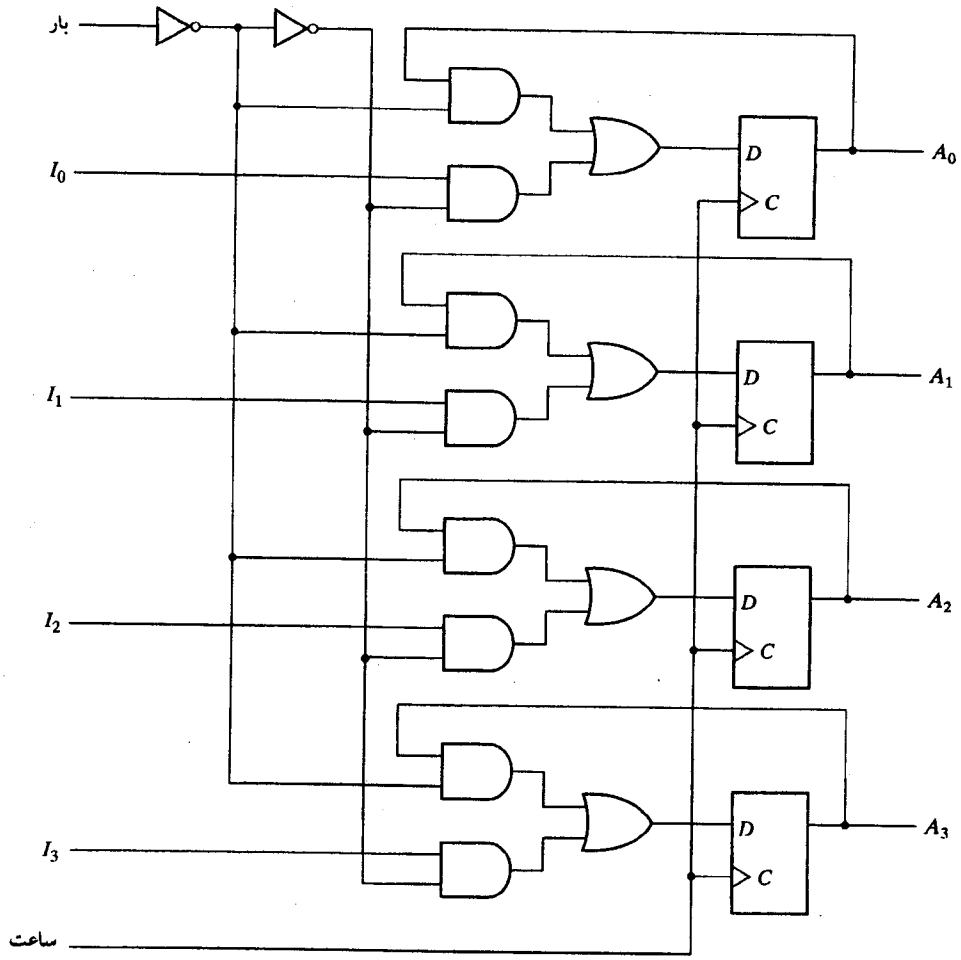
سیستم‌های دیجیتال همزمان دارای یک مولد ساعت اصلی‌اند که رشته‌ای از پالس‌های ساعت را به طور پیوسته فراهم می‌سازند. پالس‌های ساعت به همه فلیپ فلاپ‌ها و ثبات‌ها در سیستم اعمال می‌گردند. ساعت اصلی مانند پمپی است که ضربان ثابتی را برای همه بخش‌های سیستم فراهم می‌نماید. برای تأثیر یک پالس ساعت خاص بر روی یک ثبات خاص، باید یک کنترل جداگانه به کار برده شود. انتقال اطلاعات جدید به یک ثبات را بار شدن ثبات نامند. اگر همه بیت‌های ثبات به طور همزمان با یک پالس بار شوند گوییم بار شدن موازی است. لبه ساعت اعمال شده به ورودی‌های C ثبات شکل ۱-۶ موجب می‌شود تا هر چهار ورودی به طور موازی بار گردند. در این آرایش اگر بخواهیم ثبات بدون تغییر رها شود، باید ساعت از مدار قطع گردد. این کار با کنترل سیگنال ورودی ساعت به وسیله گیت فعال‌ساز انجام می‌شود. با این وجود قرار دادن گیت‌ها در مسیر ساعت به این معنی است که یک کار منطقی صورت گرفته است. استقرار گیت‌ها موجب تولید تأخیرهای نابرابر در ورودی فلیپ فلاپ‌ها می‌گردد. برای همزمانی کامل سیستم، باید مطمئن بود که همه پالس‌های ساعت به طور همزمان به هر نقطه از سیستم می‌رسد و بنابراین همه فلیپ فلاپ‌ها به طور همزمان تریگر می‌شوند. اعمال پالس ساعت از طریق گیت، تأخیرهای متغیری را موجب می‌شود و ممکن است سیستم را از همزمانی خارج کند. به این دلیل پیشنهاد می‌شود که کنترل عمل یک ثبات با ورودی‌های D به جای کنترل ساعت در ورودی‌های C فلیپ فلاپ‌ها انجام گیرد.

یک ثبات 4 بیتی با ورودی کنترل بار شدن که از طریق گیت‌ها به ورودی‌های D فلیپ فلاپ هدایت شده در شکل ۲-۶ ملاحظه می‌گردد. ورودی بار شدن به ثبات عملی را که در هر پالس ساعت اتفاق می‌افتد مشخص می‌کند. وقتی که ورودی بار برابر با 1 است، داده در چهار ورودی در لبه مثبت پالس ساعت بعدی به داخل ثبات منتقل می‌شود. وقتی ورودی بار شدن 0 است، خروجی‌های فلیپ فلاپ‌ها به ورودی‌های خودشان وصلند. اتصال پسخوردی از خروجی به ورودی لازم است زیرا فلیپ فلاپ D دارای حالت "بی‌تغییر" نیست. در هر لبه پالس ساعت، ورودی D حالت بعدی فلیپ فلاپ را مشخص می‌نماید. برای بدون تغییر نگهداشتن فلیپ فلاپ، لازم است D را با حالت فعلی فلیپ فلاپ یکسان نماییم.

پالس‌های ساعت مرتباً به ورودی‌های C اعمال می‌گردند. ورودی بار شدن، پذیرش اطلاعات جدید و یا حفظ اطلاعات فعلی را در ثبات معین می‌کند. انتقال اطلاعات از ورودی‌ها یا خروجی‌های ثبات در پاسخ به یک لبه ساعت به طور همزمان در هر چهار بیت انجام می‌گیرد.

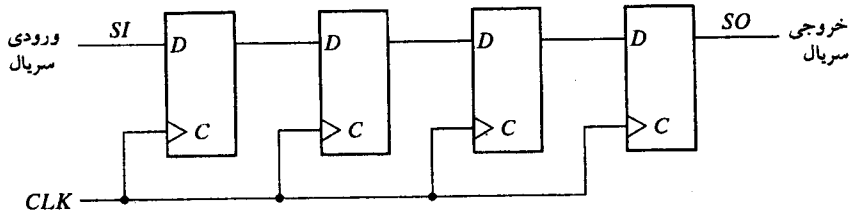
## ۲-۶ شیفت رجیسترها

ثباتی که بتواند اطلاعات دودویی‌اش را به سمت راست یا چپ شیفت یا جابجا کند، شیفت رجیستر یا ثبات جابجایی نامیده می‌شود. ساختار منطقی یک شیفت رجیستر، از زنجیره‌ای از فلیپ فلاپ‌ها تشکیل شده که در آن خروجی یک فلیپ فلاپ به ورودی فلیپ فلاپ دیگر متصل است. همه فلیپ فلاپ‌ها پالس ساعت مشترکی دریافت می‌کنند. پالس‌های ساعت اطلاعات را از یک طبقه به طبقه دیگر جابجا می‌کنند.




**شکل ۲-۶**. ثبات 4 بیتی با بار شدن موازی

ساده‌ترین شیفت رجیستر طبق شکل ۳-۶ فقط از فلیپ فلاپ‌ها استفاده می‌کند. خروجی یک فلیپ فلاپ مفروض به ورودی D فلیپ فلاپ سمت راست خود متصل است. هر پالس ساعت محتوای ثبات را یک بیت به راست جابجا می‌کند. ورودی سریال، تعیین کننده اطلاعاتی است که از منتهی‌الیه سمت چپ در حین جابجایی وارد می‌شود. خروجی سریال از خروجی سمت راست‌ترین فلیپ فلاپ اخذ می‌گردد. گاهی لازم است تا جابجایی را طوری کنترل کنیم که فقط با پالس‌های معینی رخ دهد. این کار با ممانعت از پالس ساعت در رسیدن به ثبات امکان‌پذیر است. بعد نشان خواهیم داد که عمل جابجایی می‌تواند از طریق ورودی‌های D به جای ورودی ساعت ثبات کنترل گردد. در هر صورت اگر شیفت رجیستر شکل ۳-۶ به کار رود، می‌توان عمل جابجایی را به وسیله یک گیت AND و ورودی که جابجایی را کنترل می‌کند تحت کنترل در آورد.

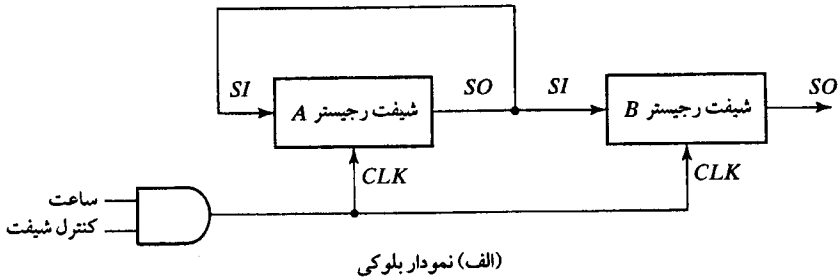



**شکل ۳-۶**. شیفت رجیستر 4 بیتی

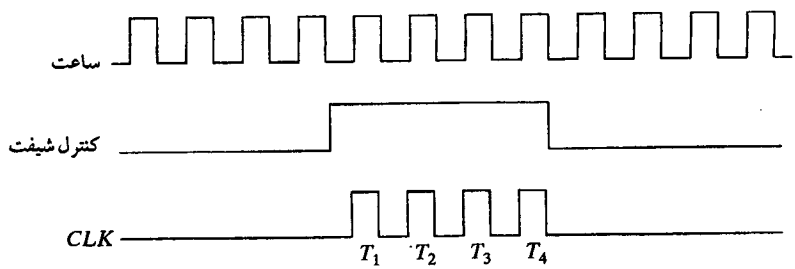
## انتقال سریال

اگر یک سیستم دیجیتال هر بار یک بیت را انتقال دهد و یا دستکاری نماید، گوییم در مد سریال کار می‌کند. با جابجایی یک بیت به خارج ثبات مبدأ و ورود به ثبات مقصد، اطلاعات هر بار یک بیت انتقال می‌یابد. این برخلاف انتقال موازی است که در آن همه بیت‌های ثبات به طور همزمان انتقال می‌یابند.

انتقال سریال اطلاعات از ثبات A به ثبات B طبق نمودار بلوکی شکل ۴-۶ (الف) با شیفت رجیستر انجام می‌شود. خروجی سریال (SO) از ثبات A به ورودی سریال (SI) در ثبات B وصل است. برای پیشگیری در از دست دادن اطلاعات ذخیره شده در ثبات مبدأ، اطلاعات ثبات A از خروجی سریال به ورودی سریالش چرخانده می‌شود. در حین عمل جابجایی مقدار اولیه ثبات B به بیرون منتقل شده و از بین می‌رود، مگر این که به ثبات سومی انتقال یابد. ورودی کنترل جابجایی زمان و تعداد دفعاتی که ثبات‌ها جابجا می‌شوند را معین می‌سازد. این کار با یک گیت AND انجام می‌گردد و طی آن پالس‌های ساعت اجازه عبور به پایانه‌های CLK را به هنگام فعال بودن کنترل جابجایی خواهند داشت.




(الف) نمودار بلوکی




(ب) نمودار زمانی

**شکل ۴-۶**. انتقال سریال از ثبات A به ثبات B

**جدول ۱-۶**. مثال انتقال سریال

| پالس زمانی | شیفت رجیستر **A** | شیفت رجیستر **B** |
|---|---|---|
| مقدار اولیه | 1 0 1 1 | 0 0 1 0 |
| پس از $T_1$ | 1 1 0 1 | 1 0 0 1 |
| پس از $T_2$ | 1 1 1 0 | 1 1 0 0 |
| پس از $T_3$ | 0 1 1 1 | 0 1 1 0 |
| پس از $T_4$ | 1 0 1 1 | 1 0 1 1 |

فرض کنید که شیفت رجیسترها هر کدام دارای چهار بیت باشند. واحد کنترلی که انتقال را مدیریت می‌کند باید طوری طراحی شود که شیفت رجیسترها را در طول سیگنال کنترل جابجایی برای مدت چهار پالس ساعت فعال‌سازد. این مطلب در نمودار زمانبندی شکل ۴-۶ (ب) ملاحظه می‌شود. سیگنال کنترل جابجایی با ساعت همگام است و مقدارش درست پس از لبه منفی پالس ساعت تغییر می‌یابد. در چهار پالس ساعت بعدی سیگنال کنترل جابجایی فعال است و خروجی گیت AND متصل به ورودی‌های CLK چهار پالس $T_1$ ، $T_2$ ، $T_3$ و $T_4$ را تولید می‌نماید. هر لبه بالا رونده پالس یک جابجایی را در هر ثبات انجام می‌دهد. چهارمین پالس کنترل جابجایی را 0 نموده و موجب می‌شود تا شیفت رجیسترها غیرفعال شوند.

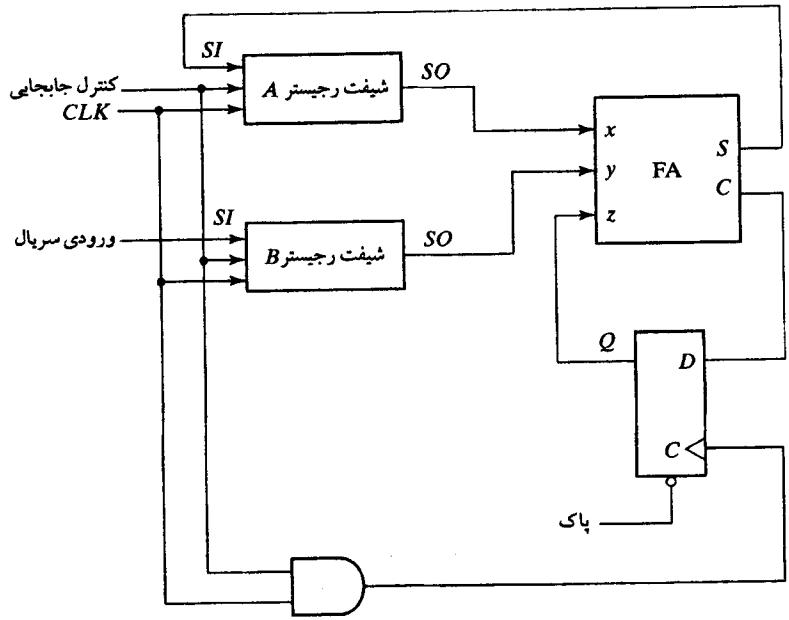
فرض کنید که محتوای دودویی A قبل از جابجایی 1011 و B برابر 0010 باشد. انتقال سریال از A به B در چهار مرحله رخ می‌دهد، جدول ۱-۶. با اولین پالس، $T_1$ ، سمت راست‌ترین بیت A به سمت چپ‌ترین بیت B منتقل می‌گردد و نیز به سمت چپ‌ترین بیت A می‌چرخد. در همان زمان تمام بیت‌های A و B یک مکان به راست جابجا می‌شوند. خروجی سریال قبلی از B در سمت راست‌ترین مکان از بین رفته و مقدار آن از 0 به 1 تبدیل می‌گردد. سه پالس بعدی اعمال مشابهی را انجام می‌دهند و بیت‌های A و B را هر بار یک بیت به راست جابجا می‌کنند. پس از چهارمین جابجایی، کنترل جابجایی به 0 رفته و هر دو ثبات A و B دارای مقدار 1011 خواهند بود. بنابراین محتوای A به B منتقل شده است، ضمن این که A همچنان بدون تغییر باقی می‌ماند.

با توجه به این مثال تفاوت بین مدهای سریال و موازی کاملاً آشکار است. در مد موازی، اطلاعات همه بیت‌های ثبات در دسترس است و همگی می‌توانند با یک پالس ساعت به طور همزمان انتقال یابند. در مد سریال ثبات‌ها دارای یک ورودی سریال و یک خروجی سریال هستند، اطلاعات هر بار یک بیت انتقال می‌یابد و ثبات‌ها در یک جهت جابجا می‌شوند.

## جمع سریال

عملیات در کامپیوترهای دیجیتال معمولاً به صورت موازی صورت می‌گیرد زیرا این روش سریع‌ترین نوع است. عملیات سریال کندتر است، ولی به قطعات کمتری نیاز دارد. برای ارائه مد سریال، در اینجا یک جمع‌کننده سریال را نشان می‌دهیم. نوع موازی آن در بخش ۴-۴ مشاهده شد.

دو عددی که قرار است بطور سریال با هم جمع شوند در دو شیفت رجیستر ذخیره می‌شوند. بیت‌ها،




**شکل ۵-۶**. جمع کننده سریال

هر جفت یک بار به وسیله یک جمع‌کننده کامل سریال (FA)، با هم جمع می‌شوند، شکل ۵-۶. نقلی خروجی جمع‌کننده کامل به فلیپ فلاپ D انتقال می‌یابد. آنگاه خروجی این فلیپ فلاپ به عنوان نقلی ورودی به جفت بیت باارزش‌تر بعدی اضافه می‌شود. با انتقال حاصل جمع به A، که با جابجایی A انجام می‌گردد، می‌توان از یک ثبات برای هر دو مقدار مضاف‌الیه و حاصل جمع استفاده کرد. ورودی سریال ثبات B می‌تواند برای انتقال یک مقدار دودویی جدید به کار رود و در همان زمان بیت‌های مضاف در حین جمع، به خارج جابجا می‌شوند.

طرز کار یک جمع‌کننده سریال به شرح زیر است. در ابتدا ثبات A مقدار مضاف‌الیه و ثبات B مقدار مضاف را نگه می‌دارند، ضمن این که فلیپ فلاپ نقلی به 0 پاک شده است. خروجی Q از فلیپ فلاپ نقلی ورودی در z را تهیه می‌کند. کنترل جابجایی هر دو ثبات و فلیپ فلاپ نقلی را فعال می‌سازد، به نحوی که در پالس ساعت بعدی، هر دو ثبات یک بار به راست جابجا می‌شوند، بیت حاصل جمع از S به سمت چپ فلیپ فلاپ A می‌رود و رقم نقلی به فلیپ فلاپ Q منتقل خواهد شد. کنترل جابجایی ثبات‌ها را به تعداد پالس‌هایی برابر با تعداد بیت‌های ثبات‌ها فعال می‌کند. در قبال هر یک پالس ساعت جدید، یک بیت حاصل جمع جدید به A می‌رود. یک نقلی جدید به Q رفته و هر دو ثبات یک بار به سمت راست جابجا می‌گردند. این روند تا از کار افتادن کنترل جابجایی ادامه می‌یابد. بنابراین، جمع با عبور هر جفت بیت به همراه نقلی قبلی از یک جمع‌کننده کامل و انتقال حاصل جمع به ثبات A، در هر بار یک بیت، انجام می‌گردد.

در آغاز، ثبات A در فلیپ فلاپ نقلی به 0 پاک می‌شود و سپس اولین عدد از B به آن اضافه می‌گردد. ضمن جابجایی B به جمع‌کننده، دومین عدد از طریق ورودی سریال وارد می‌شود. این عدد به محتوای ثبات A اضافه می‌شود، و در همان هنگام سومین عدد وارد ثبات می‌گردد. این کار می‌تواند برای تشکیل جمع دو، سه یا چند عدد و ذخیره حاصل جمع در ثبات A تکرار شود.

از مقایسه جمع‌کننده سریال با موازی بخش ۴-۴، چندین تفاوت ملاحظه می‌گردد. جمع‌کننده موازی از ثبات‌هایی با امکان بار شدن موازی استفاده می‌کند، در حالی که جمع‌کننده سری شیفت رجیسترها را به کار می‌برد. تعداد جمع‌کننده‌های مدار موازی برابر تعداد بیت‌های اعداد دودویی است، در صورتی که جمع‌کننده سریال از یک جمع‌کننده کامل و یک فلیپ فلاپ برای ذخیره نقلی و خروجی استفاده می‌نماید. دلیل ذخیره نقلی این است که در اعمال سریال نتیجه یک جمع بیتی در هر زمان نه تنها به ورودی‌های فعلی بلکه به ورودی‌های قبلی که باید در فلیپ فلاپ‌ها ذخیره شوند نیز بستگی دارد.

برای نمایش روش طراحی اعمال سریال با مدارهای ترتیبی دوباره جمع‌کننده سریال را با استفاده از جدول حالت طراحی می‌کنیم. ابتدا فرض می‌نماییم که دو شیفت رجیستر برای ذخیره اعدادی که قرار است با هم به طور سریال جمع شود موجود باشد. خروجی‌های سریال از ثبات‌ها را x و y می‌نامیم. مدار ترتیبی موردنظر در حال حاضر فاقد شیفت رجیستر است ولی هنگام تکمیل آن به مدار اضافه خواهد شد. مدار ترتیبی دارای دو ورودی x و y است که دو بیت با ارزشتر دو عدد را برای مدار تهیه می‌کنند، یک خروجی S که بیت حاصل جمع را تولید می‌نماید و یک فلیپ فلاپ Q برای ذخیره رقم نقلی است. جدول حالتی که مدار ترتیبی را تعریف می‌کند در جدول ۲-۶ نشان داده شده است. حالت فعلی Q مقدار فعلی نقلی است. نقلی فعلی در Q با ورودی‌های x و y جمع می‌شود تا بیت جمع را در خروجی S تولید نماید. حالت بعدی Q برابر نقلی فعلی است. توجه کنید که وارده‌های جدول حالت، با وارده‌های جدول جمع‌کننده کامل یکی است جز این که نقلی ورودی اکنون حالت فعلی Q ، و نقلی خروجی اکنون حالت بعدی آنست .

**جدول ۲-۶**. جدول حالت یک جمع کننده سریال

| حالت فعلی | ورودی ها | | حالت بعدی | خروجی | ورودی‌های فلیپ فلاپ | |
|---|---|---|---|---|---|---|
| $Q$ | $X$ | $y$ | $Q$ | $S$ | $J_Q$ | $K_Q$ |
| 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 |

اگر از فلیپ فلاپ D برای Q استفاده شود، مدار به شکل ۵-۶ کاهش می‌یابد. اگر برای Q از JK استفاده نماییم، لازم است مقادیر ورودی‌های J و K را با ارجاع به جدول تحریک (جدول ۱۲-۵) معین کنیم. این کار در دو ستون آخر جدول ۲-۶ انجام شده است. دو معادله ورودی فلیپ فلاپ و معادله خروجی با نقشه به صورت زیر ساده می‌شوند.

$$J_Q = xy$$
$$K_Q = x'y' = (x + y)'$$
$$S = x \oplus y \oplus Q$$

نمودار مدار در شکل ۶-۶ نشان داده شده است. مدار متشکل از سه گیت و یک فلیپ فلاپ JK می‌باشد. برای تکمیل جمع‌کننده سریال دو شیفت رجیستر هم به آن اضافه شده است. توجه کنید که خروجی S نه فقط تابعی از x و y است، بلکه تابعی از Q هم می‌باشد. حالت بعدی Q تابعی از حالت فعلی Q و مقادیر x و y رسیده از خروجی‌های شیفت رجیسترهاست.

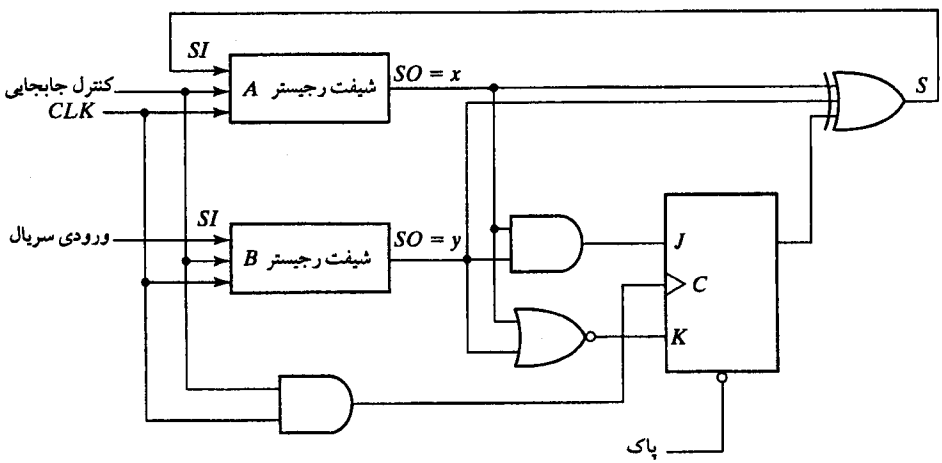
## شیفت رجیسترهای یونیورسال

اگر خروجی فلیپ فلاپ‌های یک شیفت رجیستر قابل دسترسی باشد، آنگاه می‌توان اطلاعات 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، آنگاه داده وارده موازی به ثبات 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 ثبات به 0.

۲- ورودی ساعت برای همزمانی اعمال.




**شکل ۶-۶**. فرم دوم یک جمع کننده سریال

۳- کنترل جابجایی به راست برای فعال کردن عمل جابجایی به راست و خطوط ورودی و خروجی سریال مربوط به جابجایی به راست.

۴- کنترل جابجایی به چپ برای فعال کردن عمل جابجایی به چپ و خطوط ورودی و خروجی سریال مربوط به جابجایی به چپ.

۵- یک کنترل بارکردن موازی برای فعال کردن انتقال موازی و n خط ورودی مربوط به انتقال موازی.

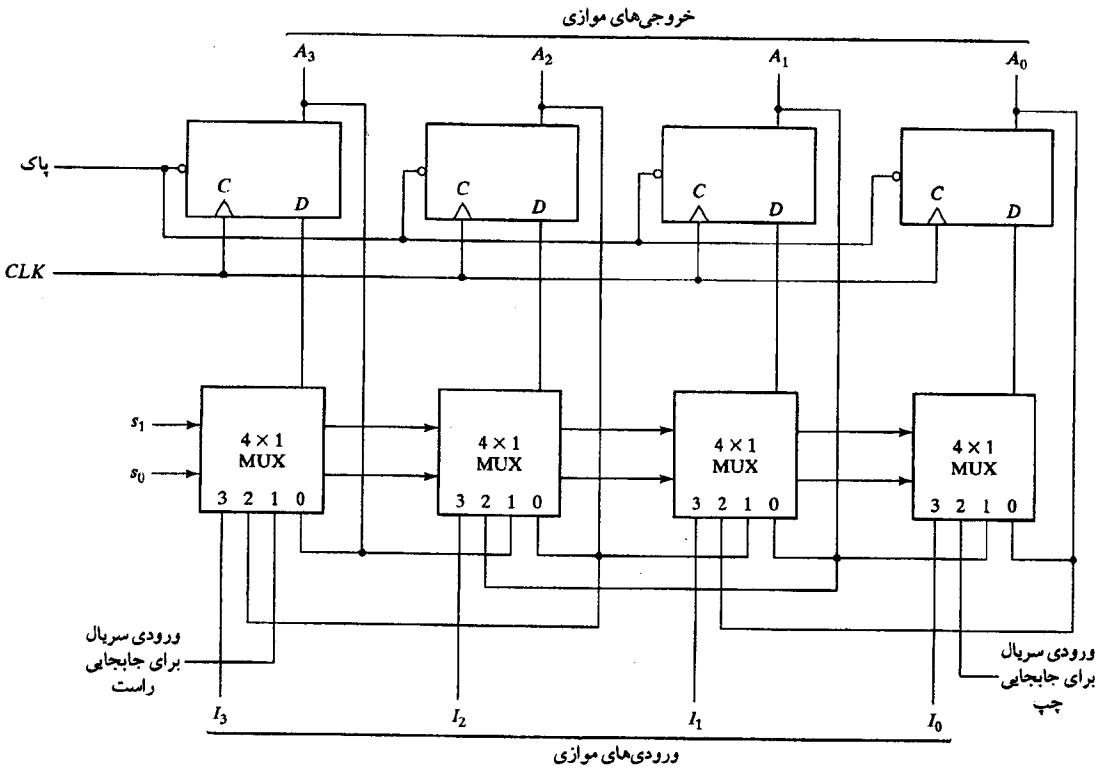
۶- n خط خروجی موازی.

۷- حالت کنترلی که علیرغم وجود پالس ساعت اطلاعات را در ثبات بدون تغییر نگه می‌دارد.

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

ثباتی که فقط قادر به جابجایی داده در یک جهت باشد را شیفت رجیستر یک جهته گویند. اگر در دو جهت جابجا نماید آن را شیفت رجیستر دو جهته می‌نامند. اگر ثبات قادر به جابجایی دو جهته و بار شدن موازی باشد به آن شیفت رجیستر یونیورسال گویند.

نمودار یک شیفت رجیستر 4 بیت یونیورسال با همه فلیپ فلاپ‌های فوق‌الذکر در شکل ۷-۶ نشان داده شده است. این مدار از چهار فلیپ فلاپ D و چهار مولتی پلکسر ساخته شده است. چهار مولتی پلکسر




**شکل ۷-۶**. شیفت رجیستر یونیورسال 4 بیتی

**جدول ۳-۶**. جدول عملکرد ثبات شکل (۷-۶)

| عملکرد ثبات | وضعیت کنترل | |
|---|---|---|
| | $S_0$ | $S_1$ |
| بلاتغییر | 0 | 0 |
| شیفت - راست | 1 | 0 |
| شیفت - چپ | 0 | 1 |
| بارکردن موازی | 1 | 1 |

دو ورودی انتخاب مشترک $S_1$ و $S_0$ دارند. ورودی 0 در هر مولتی پلکسر وقتی انتخاب می‌شود که $S_1S_0=00$ باشد، ورودی 1 با $S_1S_0 = 01$ انتخاب می‌گردد، و به طور مشابه دو ورودی باقیمانده‌انتخاب می‌گردند. ورودی‌های انتخاب مد عملیات ثبات را طبق وارده‌های جدول (۳-۶) کنترل می‌کنند. وقتی $S_1S_0 = 00$ است، مقدار فعلی ثبات به ورودی‌های D از فلیپ فلاپ‌ها اعمال می‌گردد. این وضعیت مسیری را از خروجی هر فلیپ فلاپ به ورودی‌اش ایجاد می‌نماید. لبه ساعت بعدی، مقداری را که از قبل در آن ذخیره شده وارد فلیپ فلاپ می‌کند و بنابراین هیچ تغییری در حالت رخ نمی‌دهد. وقتی $S_1S_0 = 01$ است، پایه ورودی 1 از مولتی پلکسر دارای مسیری به ورودی‌های D فلیپ فلاپ‌هاست. این موجب عمل جابجایی به راست می‌گردد، که در آن ورودی سریال به فلیپ فلاپ $A_3$ وارد می‌شود. وقتی $S_1S_0 = 10$ است، یک عمل جابجایی به چپ صورت می‌گیرد و طی آن دیگر ورودی به فلیپ فلاپ $A_0$ خواهد رفت. بالاخره وقتی $S_1S_0 = 11$ است، اطلاعات دودویی روی خطوط ورودی موازی به طور همزمان در لبه پالس بعدی وارد ثبات می‌گردند.

شیفت رجیسترها اغلب برای اتصال و ارتباط سیستم‌های دیجیتالی که با فواصل دوری از یکدیگر قرار دارند به کار می‌روند. مثلاً فرض کنید که بخواهیم کمیتی n بیتی را بین دو نقطه جابجا کنیم. اگر فاصله زیاد باشد، استفاده از n خط موازی گران تمام می‌شود. استفاده از یک خط و انتقال سریال اطلاعات به صورت یک بیت در هر بار اقتصادی‌تر است. فرستنده داده n بیتی را به صورت موازی وارد شیفت رجیستر کرده و داده را به صورت سریال در طول خط ارسال می‌دارد. گیرنده داده را به طور سریال وارد یک شیفت رجیستر می‌نماید. وقتی هر n بیت دریافت شد، می‌توان از خروجی‌های ثبات آنها را به صورت موازی دریافت کرد. بنابراین فرستنده یک عمل تبدیل موازی به سریال داده و گیرنده یک عمل تبدیل سریال به موازی را انجام می‌دهد.

## ۳-۶ شمارنده‌های موج‌گونه

ثباتی که براساس اعمال پالس‌های ورودی وارد رشته حالات از پیش تعیین شده‌ای می‌گردد، شمارنده نام دارد. پالس‌های ورودی ممکن است پالس‌های ساعت و یا از یک منبع بیرونی با توالی ثابت و یا متغیر باشند. رشته حالات ممکن است رشته اعداد دودویی و یا رشته حالات دیگری باشد. شمارنده‌ای که رشته اعداد دودویی را دنبال می‌کند، شمارنده دودویی نامیده می‌شود. یک شمارنده n بیتی متشکل

از n فلیپ فلاپ بوده و می‌تواند از 0 تا $2^n-1$ را بشمارد.

شمارنده‌ها به دو صورت وجود دارند: شمارنده‌های موج‌گونه و شمارنده‌های همزمان. در یک شمارنده موج‌گونه، تغییر وضعیت خروجی فلیپ فلاپ به عنوان منبع تریگر کردن دیگر فلیپ فلاپ‌ها عمل می‌کند. به بیان دیگر، ورودی C بعضی از و یا همه فلیپ فلاپ‌ها با پالس‌های ساعت مشترکی تریگر یا راه‌اندازی نمی‌شوند. برعکس در شمارنده همزمان، ورودی‌های C همه فلیپ فلاپ‌ها ساعت مشترکی، را دریافت می‌نمایند. شمارنده‌های همزمان در دو بخش بعد ارائه شده‌اند. در اینجا شمارنده‌های موج‌گونه BCD و دودویی را ارائه کرده و نحوه کار آنها را توضیح می‌دهیم.

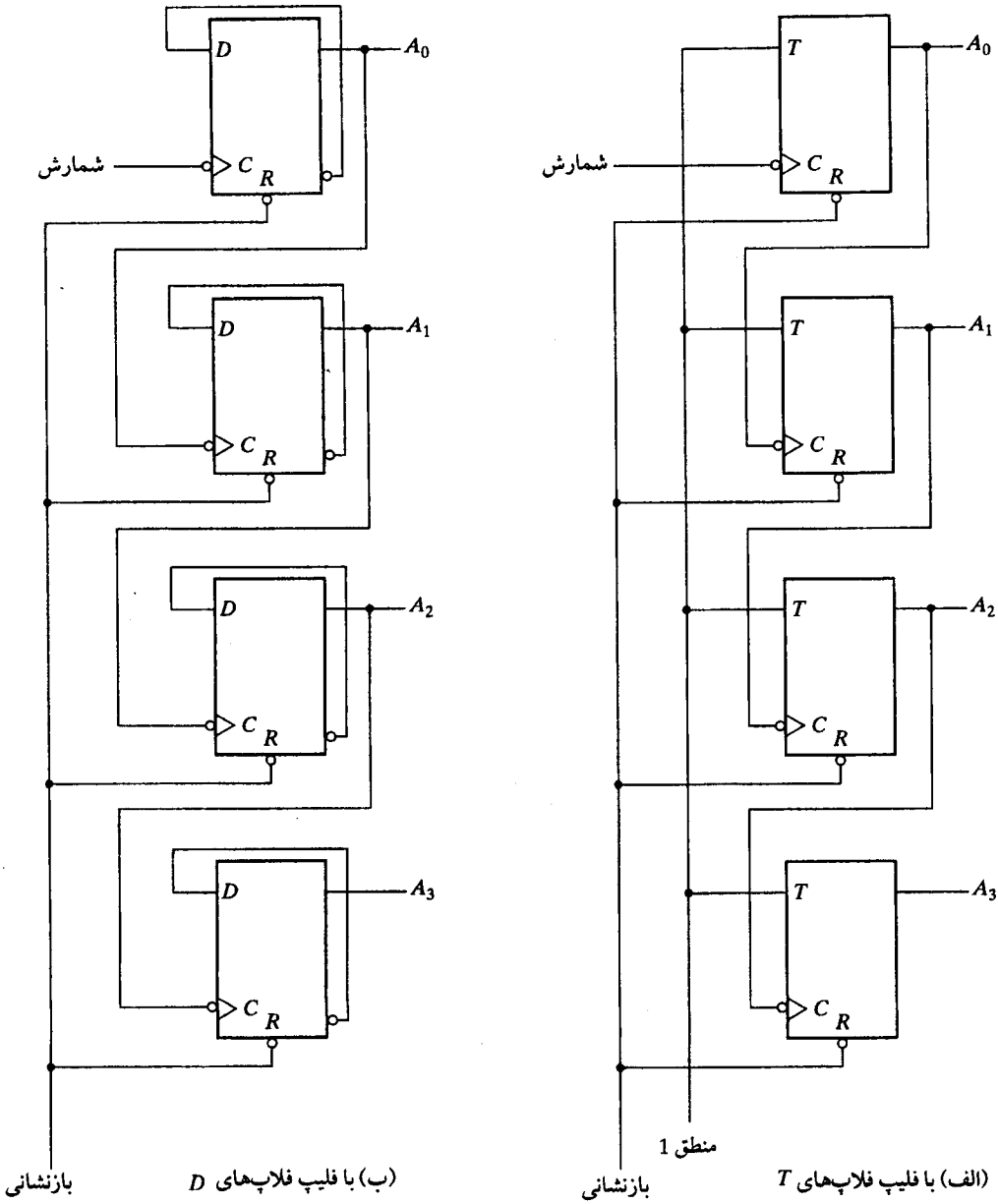
## شمارنده موج‌گونه دودویی

یک شمارنده موج‌گونه دودویی از یک سری اتصال بین فلیپ فلاپ‌های متمم‌ساز تشکیل شده است. که خروجی هر فلیپ فلاپ به ورودی C فلیپ فلاپ مرتبه بالاتر وصل است. فلیپ فلاپی که کم‌ارزش‌ترین بیت را نگه می‌دارد، پالس‌های مورد شمارش را دریافت می‌کند. فلیپ فلاپ متمم‌ساز را می‌توان با یک فلیپ فلاپ JK که در آن J و K به هم وصل‌اند و 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 دودویی 4 بیت در شکل ۸-۶ نشان داده شده است. شمارنده با فلیپ فلاپ‌های متمم‌ساز 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 کم‌ارزش‌ترین بیت را نگه می‌دارد پالس‌های شمارش را دریافت می‌کند. ورودی‌های T همه فلیپ فلاپ‌ها در (الف) به طور دایم به منطق 1 متصل‌اند. این شرایط موجب می‌شود تا با گذر منفی در ورودی C فلیپ فلاپ متمم شود. حباب جلو نشانه‌گر دینامیک ( ◁ ) 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 1 به 0 برود.

برای درک عملکرد شمارنده دودویی 4 بیت، به 9 عدد دودویی اول در جدول (۴-۶) مراجعه کنید.

**جدول ۴-۶**. رشته شمارش دودویی

| $A_3$ | $A_2$ | $A_1$ | $A_0$ |
|---|---|---|---|
| 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 |




**شکل ۸-۶**. شمارنده موج‌گونه دودویی 4 بیتی

شمارش از 0 دودویی شروع و با هر پالس در ورودی افزایش می‌یابد. پس از شماره 15 شمارنده برای تکرار به 0 باز می‌گردد. بیت کم‌ارزش‌تر $A_0$ با هر پالس شمارش ورودی متمم می‌شود. هر بار که $A_0$ از 1

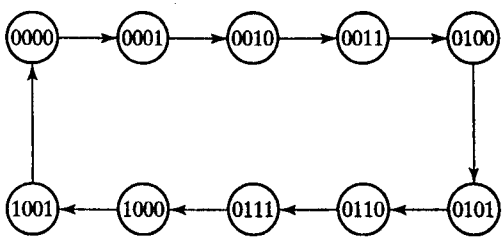
به 0 برود، $A_1$ را متمم می‌سازد. هر بار که $A_1$ از 1 به 0 برود، $A_2$ را متمم می‌نماید. هر بار که $A_2$ از 1 به 0 برود، $A_3$ را متمم می‌کند. و به همین ترتیب بیت‌های بالاتر در شمارنده موج‌گونه تغییر می‌کنند. به عنوان مثال، گذر از 0011 به 0100 را در نظر بگیرید. $A_0$ با پالس ساعت متمم می‌شود. چون $A_0$ از 1 به 0 می‌رود، $A_1$ تریگر شده و متمم می‌گردد. در نتیجه $A_1$ از 1 به 0 می‌رود که به نوبه خود موجب متمم شدن $A_2$ گشته و آن را از 0 به 1 خواهد برد. $A_2$ نمی‌تواند $A_3$ را تریگر کند زیرا $A_2$ یک گذر مثبت را تولید می‌کند و فلیپ فلاپ هم تنها به گذر منفی واکنش نشان می‌دهد.

بنابراین شمارش از 0011 به 0100 با تغییر نوبتی بیت‌ها رخ می‌دهد، به طوری که شمارنده از 0011 به 0010، سپس به 0000 و بالاخره به 0100 خواهد رفت. هر بار یکی از فلیپ فلاپ‌ها تغییر کرده و تغییر به پیش می‌رود و انتشار سیگنال در شمارنده از یک طبقه به طبقه دیگر مثل حرکت موج می‌ماند.

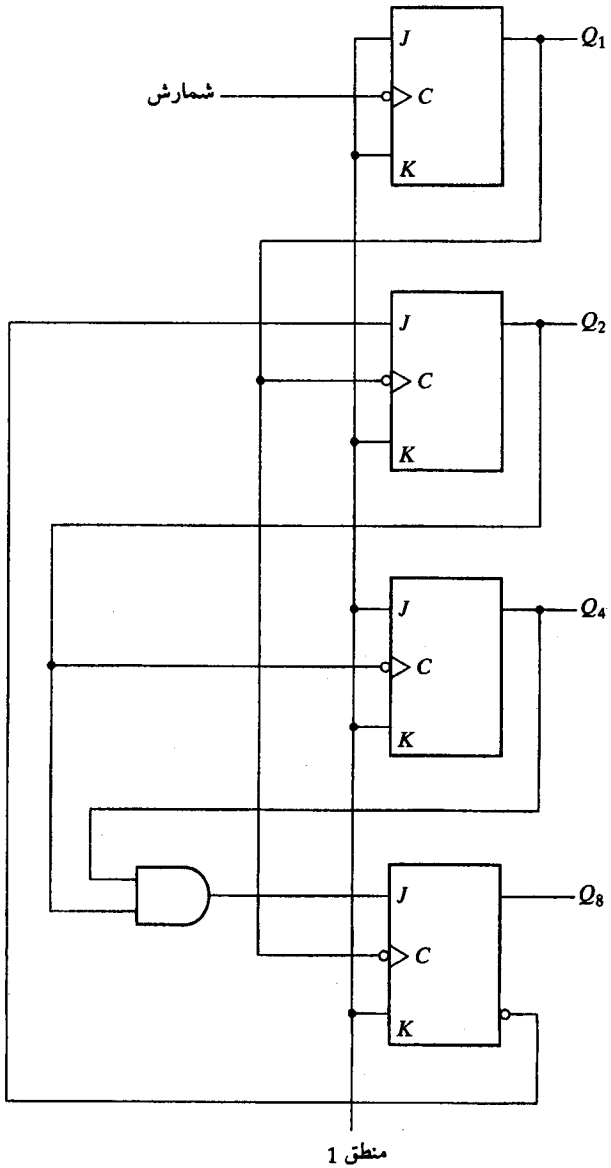
یک شمارنده دودویی با شمارش معکوس را پایین شمارگویند. در پایین شمار، شمارش با هر ورودی پالس شمارش، یک واحد کم می‌شد. شمارش یک پایین شمار 4 بیت از 15 شروع و به صورت 14، 13، 12، ... ,0 پایان یافته و سپس به 15 باز می‌گردد. لیستی از رشته شمارش یک شمارنده پایین شمار نشان می‌دهد که کم‌ارزش‌ترین بیت با هر پالس شمارش متمم شده است. هر بیت دیگر در رشته، اگر بیت کم‌ارزش‌تر قبل از آن از 0 به 1 برود، متمم می‌گردد. بنابراین نمودار یک پایین‌شمار مشابه شکل ۸-۶ خواهند بود، به شرطی که همه فلیپ فلاپ‌ها با لبه مثبت ساعت تریگر شوند. (حباب در ورودی C باید حذف شود.) اگر از فلیپ فلاپ‌های حساس به لبه منفی استفاده شود، آنگاه ورودی C هر فلیپ فلاپ باید به خروجی متمم فلیپ فلاپ قبلی وصل گردد. آنگاه وقتی که خروجی غیر متمم از 0 به 1 برود، متمم از 1 به 0 رفته و فلیپ فلاپ بعدی را آن طور که باید متمم خواهد کرد.

## شمارنده BCD موج‌گونه

یک شمارنده دهدهی رشته‌ای از ده حالت را دنبال کرده و پس از 9 به 0 باز می‌گردد. چنین شمارنده‌ای باید حداقل چهار فلیپ فلاپ برای نمایش هر رقم دهدهی داشته باشد، زیرا یک رقم دهدهی با کد چهار بیتی نشان داده می‌شود. رشته حالات در شمارنده دهدهی به وسیله کد دودویی مربوطه برای نمایش هر رقم دیجیتال معین می‌گردد. اگر BCD به کار رود، رشته حالات مطابق نمودار حالت شکل ۹-۶ خواهد بود. این جدول مشابه با جدول دودویی است. به جز این که پس از 1001 برای عدد دهدهی 9، 0000 را برای رقم دهدهی 0 خواهیم داشت.

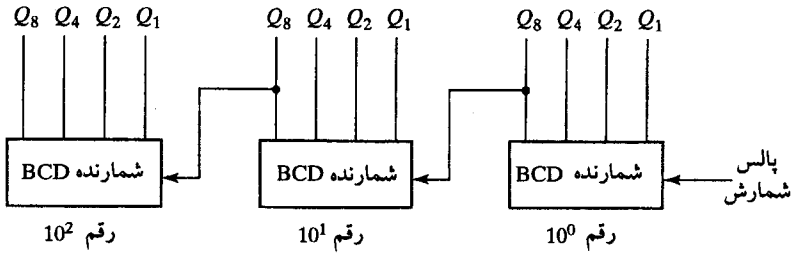



**شکل ۹-۶**. نمودار حالت یک شمارندهٔ دهدهی BCD




**شکل ۱۰-۶.** شمارندهٔ موج گونهٔ BCD

نمودار منطقی یک شمارنده BCD موج‌گونه با استفاده از فلیپ فلاپ JK در شکل ۱۰-۶ دیده می‌شود. چهار خروجی با حروف Q و اندیسی در زیر آن برای مشخص کردن وزن آن در BCD علامت‌گذاری شده است. توجه کنید که خروجی $Q_1$ به ورودی‌های C در هر دو ورودی $Q_2$ و $Q_8$ اعمال شده است و خروجی $Q_2$ هم به ورودی C از $Q_4$ وصل است. ورودی‌های J و K یا دائماً به 1 وصلند و یا به خروجی




**شکل ۱۱-۶**. نمودار بلوکی یک شمارنده دهدهی BCD با سه دهه

فلیپ فلاپ‌های دیگر وصل شده‌اند. شمارنده موج‌گونه یک مدار غیرهمزمان است. سیگنال‌ها بسته به ترتیبی که در آن از 1به 0 می‌روند، روی فلیپ فلاپ‌ها اثر می‌گذارند. عمل یک شمارنده با لیستی از حالات گذر هر فلیپ فلاپ قابل تفسیر است. این حالات از نمودار منطقی و دانستن چگونگی عملکرد یک فلیپ فلاپ JK حاصل می‌شود. به خاطر بسپارید وقتی که ورودی C از 1 به 0 می‌رود، اگر $J = 1$ باشد، فلیپ فلاپ 1 می‌شود و اگر $K = 1$ باشد به 0 پاک می‌گردد، و نیز اگر $J = K = 1$ باشد متمم شده و بالاخره با $J = K = 0$ حالت فلیپ فلاپ بی‌تغییر خواهد بود.

برای تحقیق و اطمینان از این که این حالات به ترتیب در شمارنده BCD رخ می‌دهند باید مطمئن شویم که گذر حالات فلیپ فلاپ‌ها رشته‌ای راکه به وسیله نمودار حالت شکل ۹-۶ مشخص شده دنبال می‌کند. حالت $Q_1$ پس از هر پالس ساعت عوض می‌شود. هر بار $Q_1$ از 1 به 0 برود و $Q_8 = 0$ باشد $Q_2$ متمم می‌شود. وقتی $Q_8 = 1$ شود، $Q_2$ در 0 می‌ماند. هر بار $Q_2$ از 1 به 0 برود $Q_4$ متمم می‌گردد. مادامی که $Q_2$ و $Q_4$ در 0 باشند، $Q_8$ در 0 خواهد ماند. وقتی هر دو $Q_2$ و $Q_4$ برابر 1 شوند، با تغییر 1 به 0 خروجی $Q_1$، خروجی $Q_2$ متمم می‌شود. با گذر بعدی $Q_1$، $Q_8$ پاک می‌شود.

شمارنده BCD شکل ۱۰-۶ یک شمارنده دهدهی است، زیرا از 0 تا 9 می‌شمارد. برای شمارش از 0 تا 99 دو شمارنده دهدهی لازم داریم. شمارش 0 تا 999 سه شمارنده دهدهی لازم دارد. شمارنده‌های دهدهی چند رقمی با اتصال سری شمارنده‌های BCD ساخته می‌شوند که هر کدام برای یک دهه است. یک شمارنده دهدهی سه رقمی در شکل ۱۱-۶ دیده می‌شود. ورودی‌ها به دومین و سومین دهه از $Q_8$ دهه قبل وارد می‌شوند. وقتی $Q_8$ در یک دهه از 1 به 0 می‌رود، شمارنده دهه بالاتر را تریگر می‌کند، ضمن این که خودش از 9 به 0 باز می‌گردد.

## ۶-۴ شمارنده‌های همزمان

شمارنده‌های همزمان در اعمال پالس ساعت به ورودی فلیپ فلاپ‌ها با شمارنده‌های موج‌گونه تفاوت دارند. یک ساعت مشترک همه فلیپ فلاپ‌ها را به طور همزمان تریگر می‌کند در صورتی که در نوع شمارنده‌های موج‌گونه هر بار فقط یک فلیپ فلاپ تریگر می‌شود. تصمیم بر متمم شدن یک فلیپ فلاپ از مقادیر داده‌های ورودی مانند T و 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 متمم می‌گردد.

روال طراحی شمارنده‌های همزمان در بخش ۷-۵ ارائه شد و طراحی شمارنده دودویی 3 بیت با توجه به شکل ۳۱-۵ انجام شد. در این بخش، تعدادی از مدارهای همزمان و عملکرد آنها را توضیح می‌دهیم.

## شمارنده دودویی

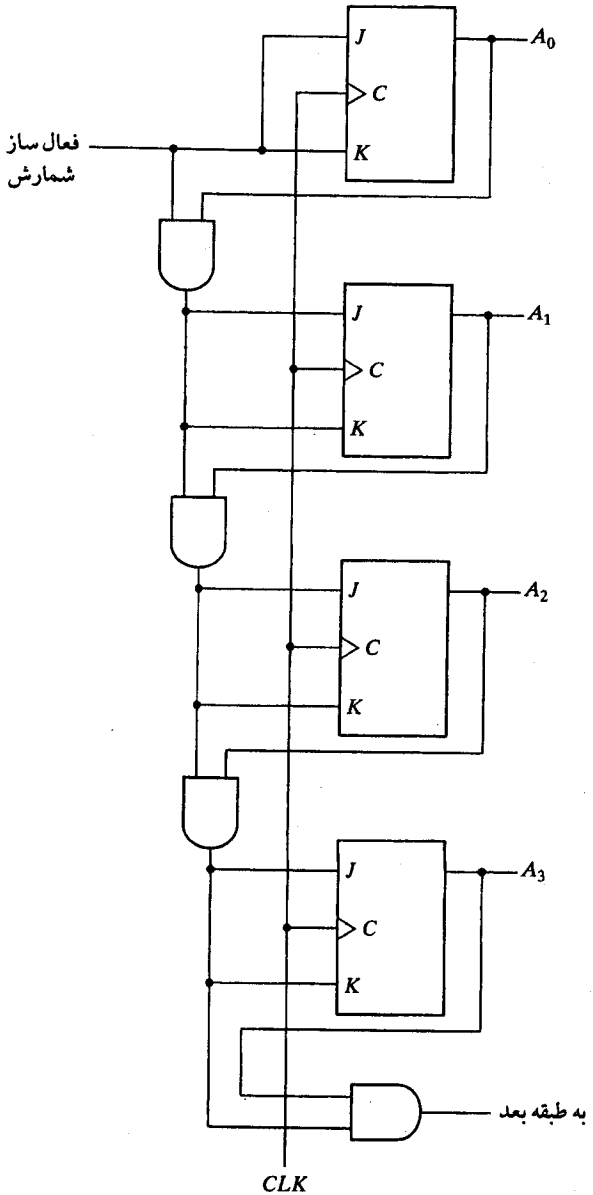
طراحی یک شمارنده دودویی آن قدر ساده است که نیازی به پیگیری مراحل طراحی را ندارد. در شمارنده دودویی همزمان، فلیپ فلاپ واقع در کم‌ارزش‌ترین مکان با هر پالس یکبار متمم می‌شود. فلیپ فلاپ‌های واقع در هر مکان هنگامی متمم می‌شود که همه فلیپ فلاپ‌های پایین‌تر 1 باشند. مثلاً اگر حالت فعلی یک شمارنده 4 بیت $A_3A_2A_1A_0 = 0011$ باشد، شماره بعدی 0100 خواهد بود. لازم به یادآوری است که در مثال فوق $A_0$ مرتباً متمم می‌شود. $A_1$ هنگامی متمم می‌گردد که $A_0$ برابر 1 باشد. $A_2$ هنگامی 1 می‌شود که $A_1A_0 = 11$ باشد. با این وجود $A_3$ متمم نمی‌شود زیرا حالت فعلی $A_2A_1A_0 = 011$ است، چون حالت تمام 1 وجود ندارد.

شمارنده‌های دودویی همزمان الگوی منظمی دارند و می‌توان آنها را با متمم کردن فلیپ فلاپ‌ها و گیت‌ها ساخت. نظم الگو را می‌توان با توجه به شکل ۱۲-۶ ملاحظه کرد. ورودی‌های C همه فلیپ فلاپ‌ها به ساعت مشترکی وصل‌اند. شمارنده با ورودی فعال‌ساز شمارش، فعال می‌گردد. اگر ورودی فعال‌ساز 0 باشد، ورودی همه Jها و Kها برابر 0 خواهند بود و بنابراین ساعت قادر نخواهد بود حالت شمارنده را عوض کند. در اولین طبقه، $A_0$، اگر شمارنده فعال شود $J = K = 1$ خواهد بود. در دیگر طبقات، Jها و Kها به شرطی 1 هستند که همه طبقات کم‌ارزش‌تر آنها برابر 1 و ورودی شمارش هم فعال شده باشد. در هر طبقه، زنجیره گیت‌های AND منطق لازم را برای ورودی‌های J و K فراهم می‌کنند. شمارنده را می‌توان به هر تعداد از طبقات گسترش داد که در آن هر طبقه یک گیت AND و یک فلیپ فلاپ اضافی خواهد داشت و هرگاه همه فلیپ فلاپ‌های طبقات قبل 1 شوند خروجی AND برابر با 1 خواهد بود.

توجه داشته باشید که فلیپ فلاپ‌ها در لبه مثبت ساعت تریگر می‌شوند. قطبیت ساعت، آنطور که در شمارنده‌های موج‌گونه مهم بود، در اینجا اهمیت ندارد. شمارنده همزمان با هر یک از دو لبه مثبت یا منفی پالس ساعت تریگر می‌گردد. فلیپ فلاپ‌های متمم‌ساز در یک شمارنده دودویی می‌توانند از نوع JK، T یا D با گیت‌های XOR باشند. هم‌ارزی سه نوع فلیپ فلاپ در شکل ۱۳-۵ مشخص شده است.

## بالا- پایین شمار دودویی

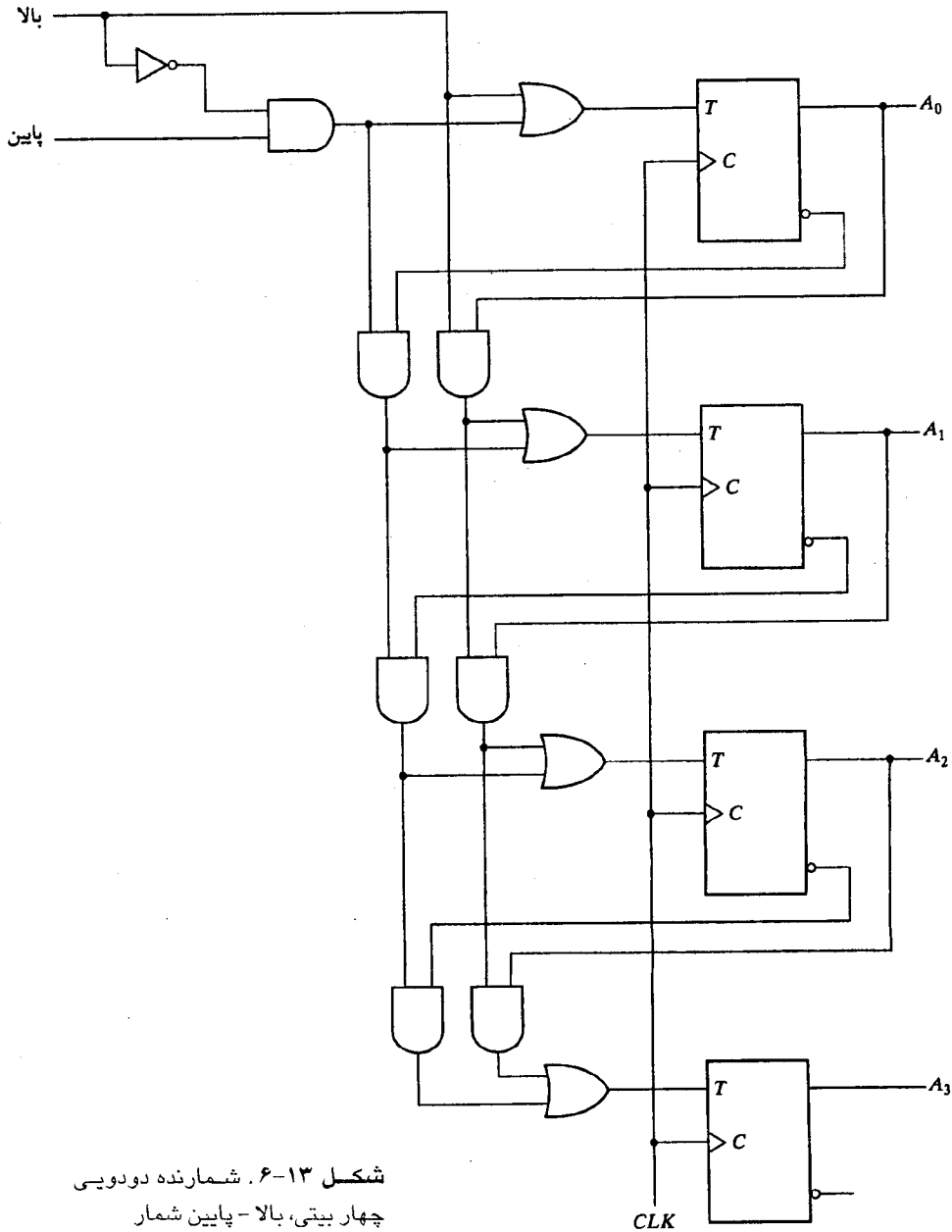
یک شمارنده پایین شمار دودویی همزمان وارد حالات معکوسی از 1111 به سمت 0000 و سپس به 1111 می‌شود تا شمارش را تکرار کند. می‌توان شمارنده پایین شماری به روش معمول ساخت، ولی نتایج از وارسی شمارش دودویی پایین شمار قابل پیش‌بینی هست. به این ترتیب که بیت مکان کم‌ارزش‌تر با هر پالس متمم می‌شود. هر بیت در هر مکان دیگر اگر همه بیت‌های کم‌ارزش‌تر 0 باشند،




**شکل ۱۲-۶**. شمارنده دودویی همزمان 4 بیتی

متمم می‌گردد. مثلاً پس از حالت فعلی 0100، حالت بعدی 0011 قرار دارد. کم‌ارزش‌ترین بیت همواره متمم می‌گردد. بیت باارزش‌تر دوم چون بیت اول 0 است، متمم می‌شود. سومین بیت متمم می‌شود زیرا دو بیت اول برابر 0 اند. ولی چهارمین بیت تغییر نمی‌کند چون همه بیت‌های پایین رتبه 0 نیستند.

یک شمارنده پایین شمار می‌تواند مشابه شکل ۱۲-۶ ساخته شود، با این تفاوت که ورودی گیت‌ها از خروجی‌های متمم فلیپ فلاپ‌های قبلی می‌آیند. می‌توان دو عمل بالا و پایین شمار را با هم ترکیب کرد و به این ترتیب بالا- پایین شمار ساخت. مدار چنین شمارنده‌ای که از فلیپ فلاپ‌های T استفاده می‌کند در شکل ۱۳-۶ آمده است. این مدار دارای یک ورودی کنترل بالا و یک ورودی کنترل پایین




**شکل ۱۳-۶** . شمارنده دودویی چهار بیتی، بالا - پایین شمار

است. وقتی ورودی بالا برابر 1 است، مدار رو به بالا می‌شمارد، زیرا ورودی‌های T سیگنال‌های خود را از مقادیر خروجی نرمال فلیپ فلاپ‌ها دریافت می‌کنند. وقتی ورودی پایین برابر 1 است ورودی بالا برابر 0 است، مدار رو به پایین می‌شمارد زیرا خروجی‌های متمم شده فلیپ فلاپ‌های قبلی به ورودی‌های T اعمال شده‌اند. وقتی هر دو مقدار بالا و پایین 0 باشند، مدار تغییر نکرده و شماره ثابت می‌ماند. وقتی هر دو ورودی بالا و پایین 1 باشند، مدار رو به بالا می‌شمارد. این موجب می‌شود تا همیشه تنها یک عمل اجرا گردد.

### شمارنده BCD

یک شمارنده BCD دهدهی کد شده به دودویی از 0000 تا 1001 و بعد 0000 می‌شمارد. به دلیل بازگشت از 9 به 0، یک شمارنده BCD دارای الگوی منظمی همچون شمارنده دودویی نیست. برای راه‌اندازی مدار یک شمارنده همزمان BCD، لازم است از روال طراحی یک مدار ترتیبی استفاده شود.

جدول حالت یک شمارنده BCD در جدول ۵-۶ لیست شده است. وضعیت ورودی فلیپ فلاپ‌های T از حالات فعلی و بعدی بدست می‌آیند. یک خروجی y هم در جدول دیده می‌شود. وقتی حالت فعلی 1001 باشد این خروجی برابر 1 است. به این ترتیب y می‌تواند شمارش دهه باارزش‌تر بعدی را فعال کرده و به طور همزمان از 1001 به 0000 برود.

معادلات ورودی فلیپ فلاپ‌ها را می‌توان به کمک نقشه ساده کرد. حالات بی‌استفاده برای مینترم‌های 10 الی 15 جملات بی‌اهمیت تلقی می‌شوند. توابع ساده شده عبارتند از:

$$T_{Q1} = 1$$
$$T_{Q2} = Q_8' Q_1$$
$$T_{Q4} = Q_2 Q_1$$
$$T_{Q8} = Q_8 Q_1 + Q_4 Q_2 Q_1$$
$$y = Q_8 Q_1$$

**جدول ۵-۶**. جدول حالت یک شمارنده BCD

| حالت فعلی | | | | حالت بعدی | | | | خروجی | ورودی های فلیپ فلاپ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $Q_8$ | $Q_4$ | $Q_2$ | $Q_1$ | $Q_8$ | $Q_4$ | $Q_2$ | $Q_1$ | $y$ | $TQ_8$ | $TQ_4$ | $TQ_2$ | $TQ_1$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |

می‌توان مدار را به سادگی با چهار فلیپ فلاپ T و پنج گیت AND و یک OR طراحی کرد. شمارنده‌های BCD همزمان را می‌توان برای شمارش اعداد دهدهی با هر طول به صورت متوالی به‌یکدیگر متصل کرد. این نوع سری‌سازی در شکل ۱۱-۶ دیده شد، با این تفاوت که خروجی y باید به ورودی شمارش دهه باارزش‌تر بعدی وصل گردد.

## شمارنده دودویی با بار شدن موازی

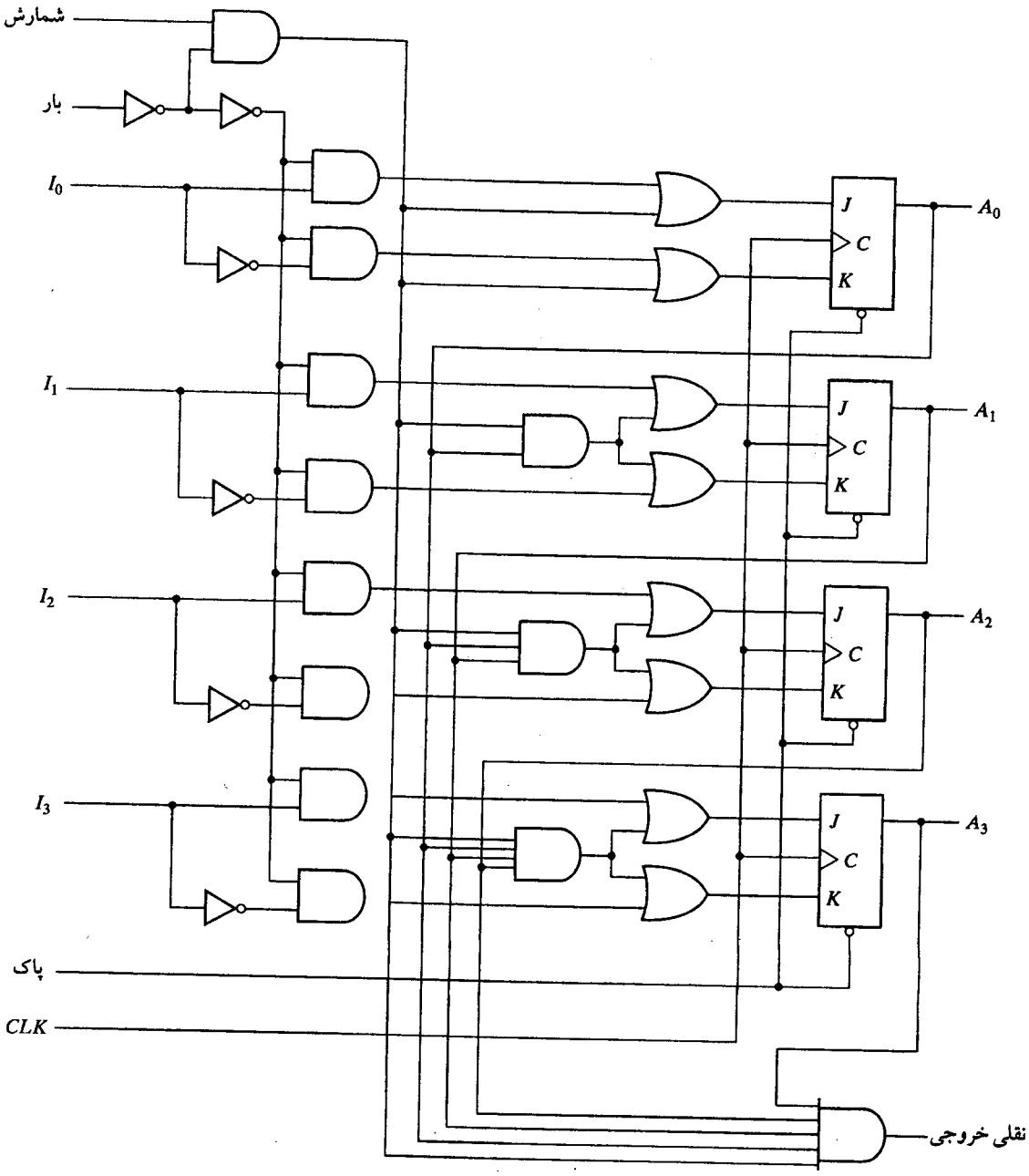
اغلب شمارنده‌هایی که در سیستم‌های دیجیتال به کار می‌روند نیاز به قابلیت انتقال موازی یک عدد دودویی اولیه به داخل شمارنده، قبل از شروع دارند. شکل ۱۴-۶ نمودار منطقی یک ثبات 4 بیت را نشان می‌دهد، که قابلیت بار شدن موازی دارد و می‌تواند به عنوان یک شمارنده به کار رود. اگر ورودی کنترل بار شدن در وضعیت 1 باشد، عمل شمارش را غیرفعال می‌کند و موجب انتقال داده از چهار ورودی داده به چهار فلیپ فلاپ می‌گردد. اگر هر دو ورودی کنترل 0 باشد، پالس‌های ساعت حالت ثبات را عوض نمی‌کند.

ضمن فعال بودن ورودی شمارش، اگر همه فلیپ فلاپ‌ها در 1 باشند، خروجی نقلی 1 می‌شود. این حالتی است که طی آن فلیپ فلاپ بیت باارزش‌تر بعدی متمم می‌گردد. خروجی نقلی برای گسترش شمارنده به بیش از چهار بیت نیز مفید است. هنگام تولید مستقیم نقلی خروجی به دلیل کاهش تأخیر آن سرعت شمارنده افزایش می‌یابد. در رفتن از 1111 به 0000، تنها یک گیت تأخیر وجود دارد، در صورتی که در زنجیره گیت AND شکل ۱۲-۶، چهار گیت تأخیر موجود است. به طور مشابه هر فلیپ فلاپ به یک گیت AND مرتبط است که مستقیماً خروجی همه فلیپ فلاپ‌ها را دریافت می‌کند.

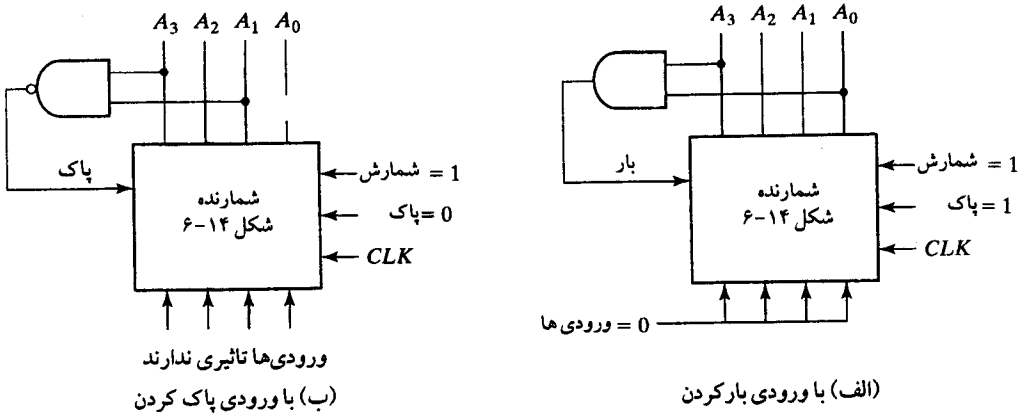
عملکرد شمارنده در جدول ۶-۶ خلاصه شده است. چهار ورودی کنترل، یعنی پاک، CLK، بار و شمارش حالت بعدی را معین می‌کنند. ورودی پاک غیرهمزمان است و هر وقت 0 شود بدون توجه به وجود پالس ساعت یا دیگر ورودی‌ها، شمارنده را پاک می‌کند. این مطلب با وارده X در جدول ذکر شده که به معنی حالات بی‌اهمیت برای ورودی‌هاست. در دیگر حالات ورودی پاک در 1 قرار دارد. با قرار داشتن ورودی‌های بار و شمارش در 0، خروجی‌ها عوض نمی‌شوند. با ورودی بار در 1، انتقال از $I_0 - I_3$ به ثبات در لبه مثبت پالس ساعت انجام می‌شود. مستقل از مقدار ورودی شمارش، داده ورودی در ثبات بار می‌شود، زیرا ورودی شمارش با فعال شدن ورودی بار، غیر فعال می‌شود. اگر ورودی شمارش کنترل شده باشد، ورودی بار در 0 خواهد بود.

**جدول ۶-۶**. جدول عملکرد شمارنده شکل (۱۴-۶)

| تابع | شمارش | بار | CLK | پاک کننده |
|---|---|---|---|---|
| به 0 پاک می‌شود | X | X | X | 0 |
| ورودی بارشونده | X | 1 | ↑ | 1 |
| شمارش حالت دودویی بعدی | 1 | 0 | ↑ | 1 |
| بلاتغییر | 0 | 0 | ↑ | 1 |




**شکل ۱۴-۶** . شمارنده دودویی 4 بیتی با امکان بار شدن موازی




**شکل ۱۵-۶**. دو روش دسترسی به شمارنده BCD با امکان بار شدن موازی

یک شمارنده با بار شدن موازی را می‌توان برای تولید هر رشته شمارش موردنظر به کار برد. شکل۱۵-۶ دو راه تولید شمارش BCD با بار شدن موازی را نشان می‌دهد. در هر حال، برای فعال کردن شمردن از طریق ورودی CLK ، کنترل شمارش در 1 قرار داده می‌شود. همچنین به خاطر بسپارید که کنترل بار از شمردن جلوگیری می‌کند و عمل پاک مستقل از دیگر ورودی‌های کنترل است.

گیت AND در شکل ۱۵-۶ (الف) وقوع حالت 1001 را شناسایی می‌کند. شمارنده در آغاز به 0 پاک می‌شود و سپس ورودی‌های پاک و بار به 1 برده می‌شوند بنابراین شمارنده همواره فعال خواهد بود. مادامی که خروجی گیت AND برابر 0 است، هر لبه ساعت مثبت، شمارنده را یک بار افزایش می‌دهد. وقتی که خروجی به 1001 برسد، $A_0$ و $A_3$ برابر 1 خواهند شد و بنابراین خروجی گیت AND برابر 1 می‌گردد. این حالت ورودی بار را فعال می‌کند و بنابراین در لبه ساعت بعدی ثبات نمی‌شمارد، ولی از طریق چهار ورودی بار خواهد شد. چون همه ورودی‌ها به منطق 0 مرتبط هستند و همه مقادیر 0 به دنبال شماره 1001 ، وارد ثبات می‌گردند، بنابراین طبق آنچه در BCD لازم است از 0000 تا 1001 شمرده و سپس به 0000 باز می‌گردد.

در شکل ۱۵-۶ (ب)، گیت AND شماره 1010 را تشخیص می‌دهد، و به محض وقوع آن، ثبات پاک می‌شود. شماره 1010 برای ماندن فرصت چندانی نمی‌یابد زیرا ثبات بلافاصله به 0 خواهد رفت. هنگام رفتن از 1010 به 1011 در خروجی $A_0$ یک جرقه کوچک رخ داده و بلافاصله به 0000 خواهد رفت. این جرقه لحظه‌ای ممکن است مطلوب نباشد و به این دلیل، این آرایش پیشنهاد نمی‌گردد. اگر شمارنده دارای وروی پاک کردن باشد، می‌توان پس از 1001 آن را پاک کرد.

## ۶-۵ دیگر شمارنده‌ها

می‌توان شمارنده‌ها را برای تولید هر رشته از حالات طراحی کرد. یک شمارنده تقسیم بر N که به آن N شمار هم می‌گویند، شمارنده‌ای است که یک رشته N حالتی را تکرار می‌کند. رشته ممکن است

شمارش دودویی یا دیگر رشته‌های اختیاری را دنبال کند. از شمارنده‌ها برای تولید سیگنال‌های زمانبندی جهت کنترل یک رشته از اعمال در یک سیستم دیجیتال استفاده می‌شود. شمارنده را می‌توان با شیفت رجیسترها هم ساخت. به این دلیل، به معرفی چند شمارنده غیردودویی می‌پردازیم.

## شمارنده با حالات بی‌استفاده

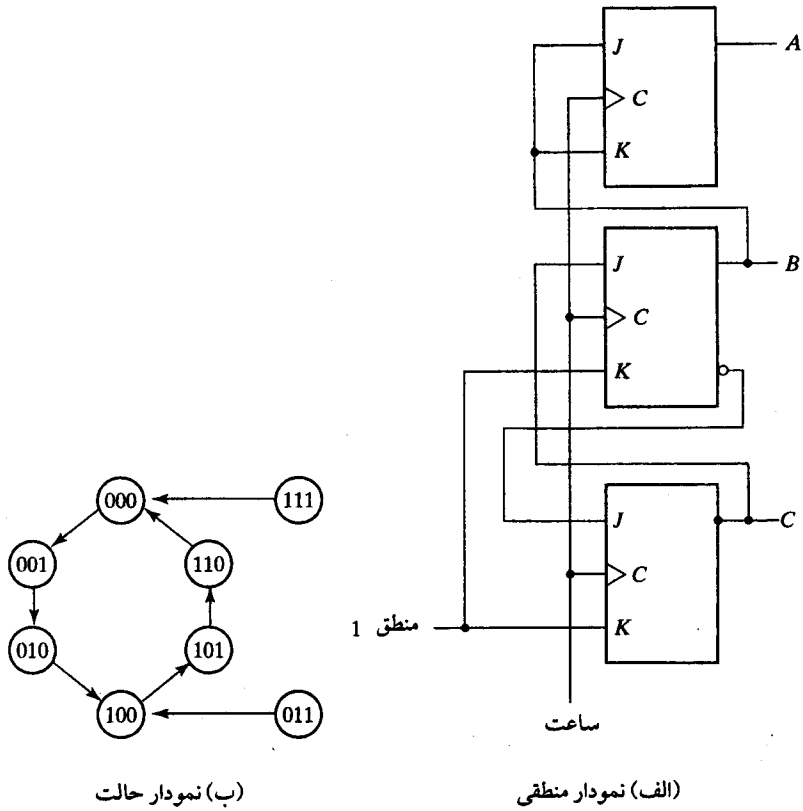
مداری با n فلیپ فلاپ دارای $2^n$ حالت دودویی است. مواردی وجود دارد که در آن یک مدار ترتیبی حالات کمتری از حداکثر فوق را به کار می‌برد. حالاتی که به کار نروند در جدول حالت لیست نمی‌شوند. هنگام ساده کردن معادلات ورودی، حالات به کار نرفته را می‌توان به عنوان حالت بی‌اهمیت یا حالت خاص بعدی تلقی کرد. به محض طراحی و ساخت مدار عوامل خارجی ممکن است آن را وارد یکی از حالات به کار نرفته کنند. در این حال لازم است مطمئن شویم که مدار بالاخره وارد یکی از حالات معتبر خواهد شد و بنابراین عملکرد معمول خود را دنبال خواهد کرد. در غیر این صورت، اگر مدار ترتیبی در میان حالات به کار نرفته به عنوان حالات بی‌اهمیت تصور شوند، آنگاه پس از طراحی مدار، باید در مورد اثر آنها تحقیق به عمل آید. پس از طراحی می‌توان حالت بعدی حاصل از حالت به کار نرفته را با تحلیل مدار بدست آورد.

به منظور تشریح، شمارنده مربوط به جدول ۷-۶ را ملاحظه کنید. شمارش، رشته‌ای تکراری از شش حالت است، که فلیپ فلاپ‌های B و C شماره‌های دودویی 00، 01 و 10 را تکرار می‌کنند و A نیز هر سه شماره یک بار از 0 به 1 و بالعکس تغییر می‌نماید. رشته شمارش شمارنده دودویی سر راست نیست و دو حالت 011 و 111 در شمارش لحاظ نشده‌اند. انتخاب فلیپ فلاپ‌های JK شرایط ورودی جدول را به دنبال خواهد داشت. ورودی‌های $K_B$ و $K_C$ تنها 1 و X را در ستون‌های خود دارند، بنابراین این حالات همیشه برابر 1 هستند. معادلات ورودی دیگر را می‌توان با مینترم‌های 3 و 7 به عنوان بی‌اهمیت مشخص کرد. معادلات ساده شده به قرار زیراند:

$$J_A = B \qquad K_A = B$$
$$J_B = C \qquad K_B = 1$$
$$J_C = B' \qquad K_C = 1$$

**جدول ۷-۶**. جدول حالت شمارنده

| حالت فعلی | | | حالت بعدی | | | ورودی های فلیپ فلاپ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | A | B | C | $J_A$ | $K_A$ | $J_B$ | $K_B$ | $J_C$ | $K_C$ |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | X | 0 | X | 1 | X |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | X | 1 | X | X | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | X | X | 1 | 0 | X |
| 1 | 0 | 0 | 1 | 0 | 1 | X | 0 | 0 | X | 1 | X |
| 1 | 0 | 1 | 1 | 1 | 0 | X | 0 | 1 | X | X | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | X | 1 | X | 1 | 0 | X |




**شکل ۱۶-۶**. شمارنده‌ای با حالات به کار رفته

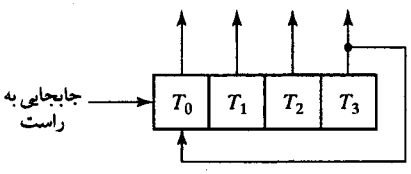
نمودار منطقی شمارنده در شکل ۱۶-۶ (الف) دیده می‌شود. چون دو حالت به کار نرفته وجود دارد، مدار را برای تعیین اثر آن‌ها تحلیل می‌کنیم. اگر مدار به علت وقوع یک سیگنال خطا به حالت 011 برود، پس از اعمال یک پالس ساعت به 100 خواهد رفت. با بررسی نمودار منطقی و توجه به این که با $B = 1$ لبه ساعت بعدی A متمم و C به 0 می‌رود، و وقتی $C = 1$ باشد لبه پالس بعدی B را متمم می‌کند، این نتیجه‌گیری به عمل آمده است. به طریقی مشابه، می‌توان حالت بعدی را از حالت فعلی 111 به 000 ارزیابی کرد.

نمودار حالت همراه با حالات به کار نرفته در شکل ۱۶-۶ (ب) ملاحظه می‌شود. اگر مدار به علت عوامل خارجی وارد یکی از حالات بی‌استفاده شود، پالس شمارش بعدی آن را به یکی از حالات معتبر خواهد برد و آنگاه مدار به طور صحیحی به شمارش خود ادامه خواهد داد. بنابراین، مدار خود تصحیح است. یک شمارنده خود تصحیح اگر در یکی از حالات به کار نرفته برود، نهایتاً پس از یک یا چند پالس ساعت به رشته شمارش طبیعی باز خواهد گشت.

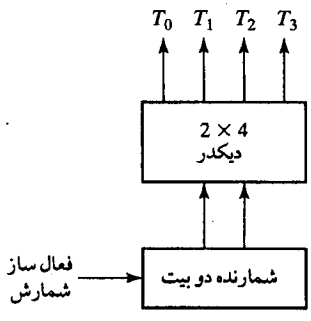
## شمارنده حلقوی

سیگنال‌های زمانبندی که رشته عملیاتی را در یک سیستم دیجیتال کنترل می‌کنند با یک شیفت رجیستر یا یک شمارنده و یک دیکدر قابل تولیدند. یک شمارنده حلقوی یک شیفت رجیستر چرخشی است که در آن هر بار تنها یک فلیپ فلاپ در حالت 1 است و همه دیگر فلیپ فلاپ‌ها صفراند. تنها بیت نشانده (1) به فلیپ فلاپ بعدی جابجا می‌شود تا رشته‌ای از سیگنال‌های زمانبندی تولید گردد.

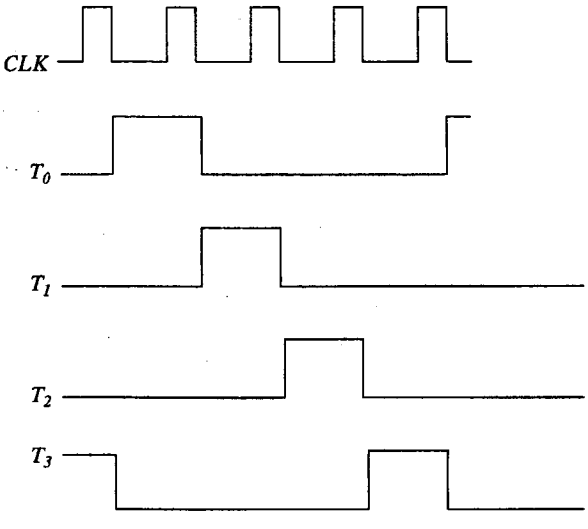
شکل ۱۷-۶ (الف) یک شیفت رجیستر 4 بیت متصل به یک شمارنده حلقوی را نشان می‌دهد. مقدار اولیه ثبات 1000 است. تنها بیت فوق‌الذکر با هر پالس ساعت به سمت راست جابجا می‌شود و چرخش هم از $T_3$ به $T_0$ اتفاق می‌افتد. هر چهار پالس ساعت یک بار یکی از فلیپ فلاپ‌ها در حالت 1 قرار می‌گیرد. هر خروجی پس از یک گذر لبه منفی پالس ساعت، 1 می‌گردد و در طول سیکل ساعت بعدی در 1 باقی می‌ماند.




(الف) شمارنده حلقوی (مقدار اولیه = 1000)




(ب) شمارنده و دیکدر




(پ) رشته‌ای از چهار سیگنال زمانی

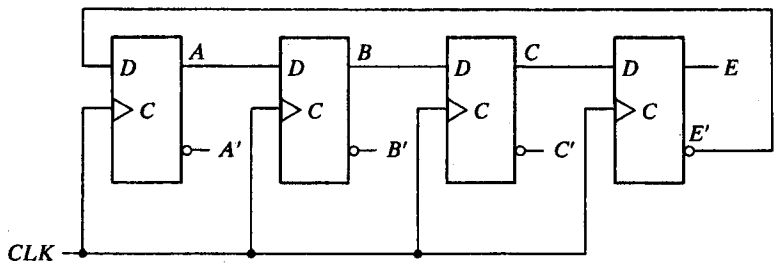
**شکل ۱۷-۶**. تولید سیگنالهای زمانی

سیگنال‌های زمانبندی با یک شمارنده 2 بیت که به چهار حالت جدا از هم می‌رود نیز تولید می‌گردد. دیکدر شکل ۱۷-۶ (ب) چهار حالت شمارنده را دیکد می‌کند و رشته سیگنال‌های زمانبندی لازم را ایجاد می‌نماید.

برای تولید $2^n$ سیگنال، به یک شیفت رجیستر با $2^n$ فلیپ فلاپ و یا به یک شمارنده دودویی n بیتی همراه با یک دیکدر n به $2^n$ نیاز است. مثلاً 16 سیگنال زمانبندی می‌تواند با یک شیفت رجیستر 16 بیتی به عنوان یک شمارنده حلقوی یا یک شمارنده 4 بیت و یک دیکدر 4 به 16 تولید گردد. در حالت اول، به 16 فلیپ فلاپ نیاز است. در دومی، 4 فلیپ فلاپ و 16 گیت AND چهار ورودی برای دیکدر لازم می‌باشد. در این حالت تعداد فلیپ فلاپ‌ها کمتر از شمارنده حلقوی است و دیکدر تنها گیت دو ورودی نیاز دارد. این ترکیب را شمارنده جانسون نامند.

## شمارنده جانسون

یک شمارنده حلقوی $k$ بیتی، یک بیت را بین $k$ فلیپ فلاپ گردش می‌دهد تا بدین وسیله $k$ حالت قابل تفکیک تولید شود. اگر شیفت رجیستر به صورت یک شمارنده حلقوی دنباله چرخان در آید، تعداد این حالات دو برابر می‌شود. شمارنده حلقوی دنباله چرخان یک شیفت رجیستر دوار است که خروجی متمم آخرین فلیپ فلاپ به ورودی اولین فلیپ فلاپ متصل شده است. شکل ۱۸-۶ (الف) چنین شیفت رجیستری را نشان می‌دهد. اتصال پسخوردی از خروجی متمم سمت راست‌ترین فلیپ فلاپ به




(الف) شمارنده حلقوی چهار طبقه دنباله چرخان

| رشته اعداد | خروجی‌های فلیپ فلاپ A | B | C | E | گیت AND لازم برای خروجی |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | $A'E'$ |
| 2 | 1 | 0 | 0 | 0 | $AB'$ |
| 3 | 1 | 1 | 0 | 0 | $BC'$ |
| 4 | 1 | 1 | 1 | 0 | $CE'$ |
| 5 | 1 | 1 | 1 | 1 | $AE$ |
| 6 | 0 | 1 | 1 | 1 | $A'B$ |
| 7 | 0 | 0 | 1 | 1 | $B'C$ |
| 8 | 0 | 0 | 0 | 1 | $C'E$ |

(ب) رشته شمارش و دیکد کردن مورد نیاز

**شکل ۱۸-۶**. ساختار یک شمارنده جانسون

ورودی سمت چپ‌ترین فلیپ فلاپ ایجاد می‌گردد. شیفت رجیستر مزبور محتوای خود را باهر پالس ساعت یک بار به سمت راست جابجا می‌کند، و در همان زمان مقدار متمم فلیپ فلاپ E به فلیپ فلاپ A منتقل می‌گردد. با شروع از حالت پاک شده، شمارنده حلقوی وارد رشته حالات هشتگانه‌ای می‌شود، شکل ۱۸-۶ (ب). به طور کلی، یک شمارنده حلقوی دنباله چرخان $k$ بیتی یک رشته $2k$ حالته را دنبال می‌نماید. این شمارنده از حالتی که همه بیت‌های آن 0 است شروع می‌نماید و هر عمل شیفت یک 1 را از سمت چپ وارد می‌کند تا همه ثبات‌ها با 1 پر شود. در دنباله عملیات 0ها از سمت چپ وارد می‌شوند تا وقتی که همه بیت‌های ثبات مجدداً با 0 پر شوند.

شمارنده جانسون، شمارنده حلقوی دنباله چرخان $k$ بیتی به همراه $2k$ گیت برای دیکدر کردن و تهیه $2k$ سیگنال زمانبندی خروجی است. گیت‌های دیکدر در شکل ۱۸-۶ نشان داده نشده‌اند، اما در آخرین ستون جدول مشخص گردیده‌اند. پس از وصل هشت گیت لیست شده در جدول، ساختار شمارنده جانسون کامل خواهد بود. نظر به این که هر گیت در حین یک حالت ویژه توانا می‌شود، خروجی گیت‌ها، به ترتیب هشت سیگنال زمانبندی را تولید خواهند کرد.

دیکد کردن یا رمزگشایی یک شمارنده k بیتی حلقوی دنباله چرخان در بدست آورن $2k$ دنباله زمانی، از یک الگوی منظم پیروی می‌کند. حالتی که همه بیت‌ها 0 هستند توسط خروجی‌های متمم دو فلیپ فلاپ واقع در منتهی‌الیه دیکد می‌شود. کلیه حالات دیگر با الگوی 0 و 1 یا 1 و 0 مجاور دیکد می‌شوند. به عنوان مثال، دنباله 7 دارای یک الگوی 0 و 1 در فلیپ فلاپ‌های B و C است. بنابراین خروجی دیکد شده با گرفتن متمم B و خروجی طبیعی C یعنی $B'C$ بدست می‌آید.

یکی از معایب مدار شکل ۱۸-۶ (الف) این است که اگر مدار به حالت بی‌استفاده وارد شود، شروع به دنبال کردن حالات نامعتبر کرده و هرگز راهش را به یک حالت معتبر نخواهد یافت. این شکل را می‌توان با تصحیح مدار به صورتی که از حالت نامعتبر دوری جوید، رفع کرد. یکی از روش‌های اصلاح، قطع خروجی فلیپ فلاپ B است که به ورودی D فلیپ فلاپ C می‌رود

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

که $D_C$ معادله ورودی فلیپ فلاپ برای ورودی D در فلیپ فلاپ C است. شمارنده جانسون را می‌توان با هر طول زمانی ساخت. تعداد فلیپ فلاپ‌های لازم نصف تعداد سیگنال‌های زمانبندی است. تعداد گیت‌ها برای ساخت آن نصف تعداد سیگنال‌های زمانبندی بوده و فقط گیت دو ورودی نیاز دارد.

## ۶-۶ HDL برای ثبات‌ها و شمارنده‌ها

ثبات‌ها و شمارنده‌ها را می‌توان در هر یک از سطوح رفتاری و ساختاری توصیف کرد. در سطح رفتاری، ثبات با انواع عملیاتی که مشابه جدول تابع انجام می‌دهد مشخص می‌گردد. یک توصیف ساختاری مدار را برحسب مجموعه‌ای از اجزاء مانند گیت‌ها، فلیپ فلاپ‌ها و مولتی پلکسرها نشان می‌دهد. انواع عناصر برای تشکیل توصیف سلسله مراتبی طرح، مشابه یک نمودار منطقی ذکر می‌گردد. در این فصل سه مدار را برای تشریح دو نوع توصیف به کار خواهیم برد.

## شیفت رجیستر

شیفت رجیستر یونیورسال که در بخش ۶-۶ ارائه شد، یک شیفت رجیستر دو جهته با بار شدن موازی است. چهار عملی که به وسیله ثبات ساعت‌دار اجرا شده‌اند در جدول (۶-۶) دیده می‌شوند. می‌توان ثبات را به طور غیرهمزمان هم پاک کرد. توصیف رفتاری یک شیفت رجیستر یونیورسال در مثال ۱-۶ HDL نشان داده شده است. در مجموع، دو ورودی انتخاب، دو ورودی سریال، یک ورودی موازی 4 بیت و یک خروجی موازی 4 بیت وجود دارند. بلوک always پنج عملی که با ثبات قابل اجرا است را توصیف می‌نماید. ورودی Clr ثبات را به طور غیرهمزمان پاک می‌کند. برای این که ثبات به لبه مثبت ساعت واکنش نشان دهد، Clr باید در سطح منطقی بالا باشد. چهار عمل ساعت‌دار ثبات از مقادیر دو ورودی انتخاب در عبارت case مشخص می‌گردند (S1 و S0 بعد از کلمه کلیدی case به ترتیب در داخل کروشه قرار گرفته‌اند.) عمل جابجایی با عملیات زنجیره‌ای ورودی سریال و سه فلیپ فلاپ انجام می‌شود. مثلاً عبارت

```
A = {rtin, A[3:1]}
```

به معنی جابجایی زنجیره ورودی سریال به سمت راست (rtin) با فلیپ فلاپ‌های $A_3$، $A_2$ و $A_1$ است تا یک عدد 4 بیتی تشکیل شده و به A[3 : 0] انتقال یابد. این موجب تولید یک عمل جابجایی به راست می‌گردد. توجه کنید که عمل مدار مستقل از هر سخت‌افزاری بیان شد.

ساختار ثبات با مراجعه به نمودار منطقی شکل ۷-۶ قابل توصیف است. نمودار نشان می‌دهد که ثبات با چهار مولتی پلکسر و چهار فلیپ فلاپ D ساخته شده است. توصیف ساختاری ثبات در

مثال ۱-۶، HDL

```
//Behavioral description of
//Universal shift register
// Fig. 6-7 and Table 6-3
module shftreg (s1,s0,Pin,lfin,rtin,A,CLK,Clr);
   input s1,s0;                    //Select inputs
   input lfin, rtin;               //Serial inputs
   input CLK,Clr;                  //Clock and Clear
   input [3:0] Pin;                //Parallel input
   output [3:0] A;                 //Register output
   reg [3:0] A;
   always @ (posedge CLK or negedge Clr)
     if (~Clr) A = 4'b0000;
      else
        case ({s1,s0})
         2'b00: A = A;             //No change
         2'b01: A = {rtin,A[3:1]}; //Shift right
         2'b10: A = {A[2:0],lfin}; //Shift left
         2'b11: A = Pin;           //Parallel load input
        endcase
endmodule
```

```
//Structural description of
//Universal shift register(see Fig. 6-7)
module SHFTREG (I,select,lfin,rtin,A,CLK,Clr);
   input [3:0] I;            //Parallel input
   input [1:0] select;       //Mode select
   input lfin,rtin,CLK,Clr;  //Serial inputs,clock,clearn
   output [3:0] A;           //Parallel output
 //Instantiate the four stages
   stage ST0 (A[0],A[1],lfin,I[0],A[0],select,CLK,Clr);
   stage ST1 (A[1],A[2],A[0],I[1],A[1],select,CLK,Clr);
   stage ST2 (A[2],A[3],A[1],I[2],A[2],select,CLK,Clr);
   stage ST3 (A[3],rtin,A[2],I[3],A[3],select,CLK,Clr);
endmodule

//One stage of shift register
module stage(i0,i1,i2,i3,Q,select,CLK,Clr);
   input i0,i1,i2,i3,CLK,Clr;
   input [1:0] select;
   output Q;
   reg Q;
   reg D;
//4x1 multiplexer
   always @ (i0 or i1 or i2 or i3 or select)
         case (select)
            2'b00: D = i0;
            2'b01: D = i1;
            2'b10: D = i2;
            2'b11: D = i3;
         endcase
//D flip-flop
   always @ (posedge CLK or negedge Clr)
         if (~Clr) Q = 1'b0;
         else Q = D;
endmodule
```

مثال۲-۶ HDL نشان داده شده است. در این مثال دو مدول وجود دارد. اولین مدول ورودی‌ها، خروجی‌ها و ساعت سپس طبقات ثبات را مشخص می‌کند. چهار بخش فوق ارتباطات درونی بین چهار طبقه را معین می‌سازد و جزئیات ساخت ثبات را طبق نمودار منطقی مهیا می‌سازد. دومین مدول دو بلوک always دارد. اولین بلوک always ، مولتی پلکسر و دومی فلیپ‌فلاپ را توصیف می‌نماید. روی هم رفته آنها یک مرحله از ثبات را تعریف می‌نمایند.

### شمارنده همزمان

مثال ۳-۶ HDL شمارنده همزمان را با بار شدن موازی، شکل ۱۴-۶، توصیف می‌کند. شمارش

```
//Binary counter with parallel load
//See Figure 6-14 and Table 6-6
module counter (Count,Load,IN,CLK,Clr,A,CO);
  input Count,Load,CLK,Clr;
  input [3:0] IN;                    //Data input
  output CO;                         //Output carry
  output [3:0] A;                    //Data output
  reg [3:0] A;
  assign CO = Count & ~Load & (A == 4'b1111);
  always @ (posedge CLK or negedge Clr)
    if (~Clr) A = 4'b0000;
    else if (Load)  A = IN;
    else if (Count) A = A + 1'b1;
    else A = A;
endmodule
```

(count) بار شدن (load)، CLK و Clr ورودی‌هایی هستند که کار ثبات را طبق عملیات جدول (۶-۶) معین می‌نمایند. شمارنده چهار ورودی داده، چهار خروجی داده و یک خروجی نقلی دارد. نقلی خروجی CO با یک مدار ترکیبی تولید می‌شود و با یک عبارت assign مشخص می‌گردد. اگر شمارش به 15 برسد CO = 1 می‌شود و در این حال شمارنده در وضعیت شمارش است. بنابراین CO = 1 خواهد بود به شرطی که Count = 1 و Load = 0 و A = 1111 باشد؛ در غیر این صورت CO = 0 است. بلوک always عملی را که بسته به مقادیر Clr، Load و Count مشخص می‌شود انجام می‌دهد. یک سیگنال منفی در Clr، A را به 0 می‌نشاند. در غیر این صورت اگر Clr = 1 باشد، یکی از سه عمل به هنگام لبه مثبت ساعت انجام می‌گیرد. عبارت if، else if و else به صورت زیر عمل تصمیم‌گیری را اجرا می‌کنند.

```
      if Clr = 0                                    Clear A to 0
else if (Clr = 1 and) Load = 1                      Load inputs to A
else if (Clr = 1 and Load = 0 and) Count = 1        Increment A
   else (Clr = 1 and Load = 0 and Count = 0)        No change in A
```

سلسله مراتب مربوط به عبارات if–else با آنچه در جدول ۶-۶ ملاحظه شد مطابقت دارد.

## شمارنده موج‌گونه

توصیف ساختاری یک شمارنده موج‌گونه در مثال ۴-۶ HDL نشان داده شده است. اولین مدول چهار فلیپ فلاپ متمم‌ساز را که در مدول دوم به صورت CF(Q, CLK, Reset) تعریف شده ذکر می‌کند. ساعت (ورودی C) اولین فلیپ فلاپ به ورودی Count بیرونی وصل است (count جایگزین CLK در F0 شده است). ورودی ساعت فلیپ فلاپ دوم به خروجی اول متصل است ($A_0$ جایگزین

```
//Ripple counter (See Fig. 6-8(b))
module ripplecounter (A0,A1,A2,A3,Count,Reset);
   output A0,A1,A2,A3;
   input Count,Reset;
//Instantiate complementing flip-flop
   CF F0 (A0,Count,Reset);
   CF F1 (A1,A0,Reset);
   CF F2 (A2,A1,Reset);
   CF F3 (A3,A2,Reset);
endmodule

//Complementing flip-flop with delay
//Input to D flip-flop = Q'
module CF (Q,CLK,Reset);
   output Q;
   input CLK,Reset;
   reg Q;
   always @ (negedge CLK or posedge Reset)
      if (Reset) Q = 1'b0;
      else  Q = #2 (~Q);      // Delay of 2 time units
endmodule

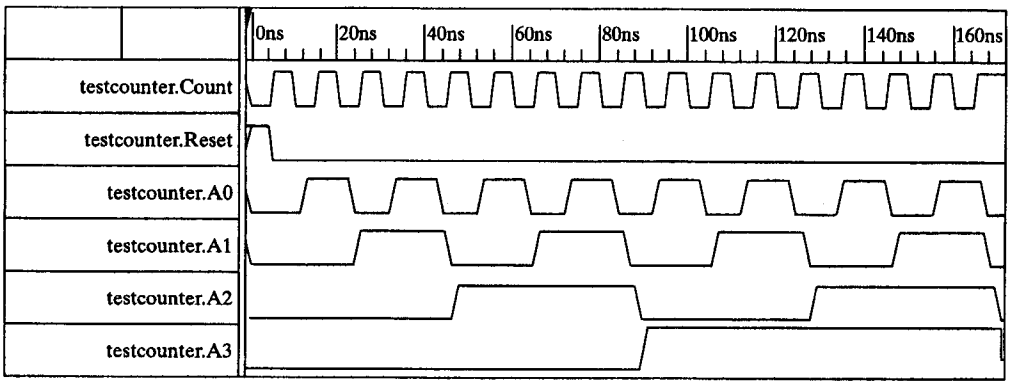
//Stimulus for testing ripple counter
module testcounter;
   reg Count;
   reg Reset;
   wire A0,A1,A2,A3;
//Instantiate ripple counter
   ripplecounter RC (A0,A1,A2,A3,Count,Reset);
always
    #5 Count = ~Count;
initial
 begin
      Count = 1'b0;
      Reset = 1'b1;
   #4 Reset = 1'b0;
   #165 $finish;
 end
endmodule
```

CLK در F1 شده است). به طور مشابه، ساعت هر فلیپ فلاپ دیگر به خروجی فلیپ فلاپ قبلی وصل می‌باشد. به این ترتیب فلیپ فلاپ‌ها زنجیروار به هم متصل‌اند تا یک شمارنده موج‌گونه طبق شکل ۸-۶ (ب) بدست آید.

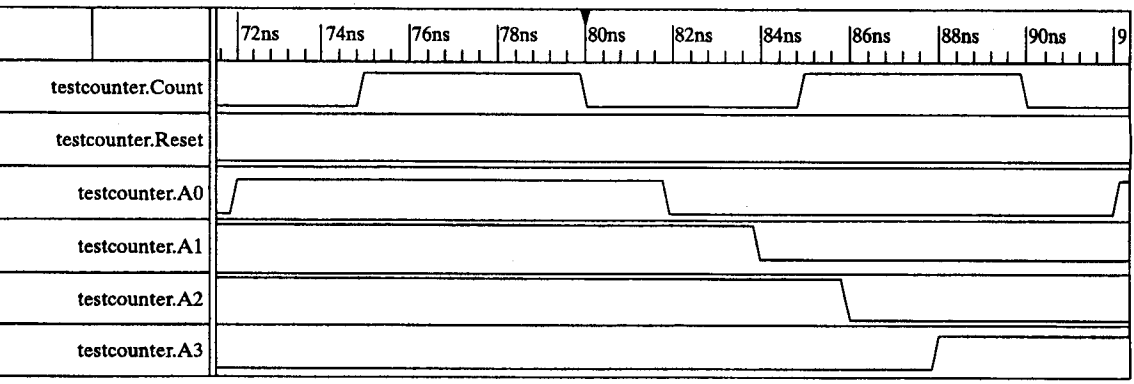
دومین مدول فلیپ فلاپ متمم‌سازی را با تأخیر نشان می‌دهد. مدار فلیپ فلاپ متمم‌ساز با اتصال خروجی متمم به ورودی D ایجاد می‌شود. یک ورودی reset (بازنشانی) در فلیپ فلاپ لحاظ شده است

تا شمارنده مقداردهی اولیه شود. شبیه‌سازهای HDL بدون مقداردهی اولیه قادر به تهیه مقادیر خروجی نیستند. به فلیپ فلاپ تأخیری برابر 2واحد زمانی تخصیص یافته است و این تأخیر از لحظه اعمال ساعت تا لحظه متمم شدن ساعت محاسبه می‌شود. این تأخیر بصورت عبارت Q=#2(~Q) نشان داده می‌شود.

مدول سوم در مثال ۴-۶ محرک شبیه‌سازی و تست شمارنده موج‌گونه را فراهم می‌نماید. عبارت always ساعتی با سیکل 10 واحد زمانی تولید می‌کند. فلیپ فلاپ‌ها در لبه منفی ساعت تریگر می‌شوند، که در زمان‌های t = 10 ، 20 ، 30 و هر 10 واحد زمانی یک بار رخ می‌دهد. شکل موج‌های حاصل برای این شبیه‌سازی در شکل ۱۹-۶ نشان داده شده است. Count هر 10ns یک بار منفی می‌شود. A0 با هر لبه منفی Count و با 2ns تأخیر متمم می‌گردد. هر فلیپ فلاپ، وقتی فلاپ فلاپ قبلی‌اش از 1 به 0 برود، متمم می‌گردد. پس از t = 80ns همه فلیپ فلاپ‌ها متمم شده‌اند زیرا شمارنده از 0111 به 1000 می‌رود. هر خروجی به اندازه 2n تأخیر دارد و به علت آن $A_3$ در t = 88ns از 0 به 1 و در 168ns از 1 به 0 خواهد رفت.




(الف) از 0 تا 170 ns




(ب) از 70 تا 92 ns

**شکل ۱۹-۶**. شبیه‌سازی خروجی مثال ۴-۶ HDL

## مسائل

۶-۱ در ثبات شکل ۶-۱ یک گیت NAND دو ورودی را لحاظ نمایید و خروجی گیت را به ورودی C همه فلیپ فلاپ‌ها وصل کنید. یکی از ورودی‌های گیت NAND پالس‌های ساعت را از مولد ساعت دریافت می‌کند و ورودی دیگر گیت NAND کنترل بار شدن موازی را مهیا می‌سازد. عملکرد ثبات اصلاح شده را توضیح دهید.

۶-۲ یک ورودی پاک همزمان به ثبات شکل ۶-۲ اضافه کنید. ثبات اصلاح شده قابلیت بار شدن موازی و پاک کردن همزمان را دارد. وقتی که ساعت از لبه مثبت می‌گذرد و ورودی پاک برابر 1 است، ثبات بطور همزمان پاک می‌گردد.

۶-۳ تفاوت بین انتقال موازی و سری چیست؟ چگونگی تبدیل داده سریال به موازی و بالعکس را توضیح دهید. چه نوع ثباتی لازم است؟

۶-۴ محتوای یک ثبات 4 بیتی در آغاز 1101 است. ثبات شش بار با ورودی سریال 101101 به راست جابجا می‌شود. محتوای ثبات پس از هر جابجایی چیست؟

۶-۵ یک شیفت رجیستر 4 بیت یونیورسال شکل ۶-۷ در یک بسته IC قرار دارد.
(الف) نمودار بلوکی IC را بکشید و همه خروجی‌ها و ورودی‌ها را نشان دهید. برای منبع تغذیه دو پایه در نظر بگیرید.
(ب) نمودار بلوکی را با دو آی‌سی برای تولید یک شیفت رجیستر 8 بیت بکشید.

۶-۶ یک شیفت رجیستر 4 بیت با بار شدن موازی با فلیپ فلاپ D طراحی کنید. دو ورودی کنترل shift و Load وجود دارد. وقتی shift = 1 است، محتوای ثبات یک مکان جابجایی می‌شود. وقتی load = 1 و shift = 0 است داده جدید وارد شیفت رجیستر می‌گردد. اگر هر دو ورودی کنترل برابر 0 باشند، محتوای ثبات تغییر نمی‌نماید.

۶-۷ نمودار منطقی یک ثبات 4 بیتی با چهار فلیپ فلاپ D و چهار مولتی پلکسر و ورودی‌های انتخاب مد S1 و S0 را رسم نمایید. ثبات طبق جدول تابع زیر عمل می‌کند:

| عملکرد ثبات | $s_0$ | $s_1$ |
|---|---|---|
| بلاتغییر | 0 | 0 |
| متمم چهارخروجی | 1 | 0 |
| پاک کردن ثبات به 0 | 0 | 1 |
| (همزمان با پالس ساعت) | 1 | 1 |
| بارکردن داده موازی | | |

۶-۸ جمع‌کننده سریال شکل ۶-۶ دو ثبات 4 بیت را به کار می‌برد. ثبات A عدد دودویی 0101 و ثبات B هم 0111 را نگه می‌دارد. فلیپ فلاپ نقلی در ابتدا 0 است. مقادیر دودویی درون ثبات A و فلیپ فلاپ نقلی را پس از هر جابجایی معین کنید.

۶-۹ دو راه پیاده‌سازی جمع‌کننده سریال (A + B) در بخش ۶-۲ نشان داده شد. برای تبدیل به تفریق‌گر (A − B) باید آن را اصلاح کرد.
(الف) با استفاده از شکل ۶-۵ تغییرات لازم برای انجام تفریق $A + \overline{B} + 1$ را نشان دهید.
(ب) با استفاده از شکل ۶-۶، نشان دهید که با اصلاح جدول ۶-۲ جمع‌کننده به تفریق‌گر تبدیل می‌شود.

۶-۱۰ یک متمم 2 ساز با شیفت رجیستر و یک فلیپ فلاپ بسازید. عدد دودویی از یک سمت آن خارج و متمم 2 از سمت دیگر وارد آن می‌گردد.

۶-۱۱ یک شمارنده موج‌گونه دودویی از فلیپ فلاپ‌هایی که در لبه مثبت ساعت تریگر می‌شوند استفاده می‌کند. اگر (الف) خروجی‌های معمولی فلیپ فلاپ‌ها به ساعت و (ب) متمم خروجی‌ها به ساعت وصل شوند، شمارش چگونه است.

۶-۱۲ نمودار منطقی یک پایین شمار موج‌گونه دودویی 4 بیت را (الف) با فلیپ فلاپ‌هایی که در لبه مثبت ساعت و (ب) در لبه منفی ساعت تریگر می‌شوند، رسم نمایید.

۶-۱۳ نشان دهید که یک شمارنده موج‌گونه BCD با یک شمارنده موج‌گونه 4 بیت و پاک‌کننده غیرهمزمان و یک گیت AND که وقوع 1010 را شناسایی کند، قابل ساخت است.

۶-۱۴ چند فلیپ فلاپ در یک شمارنده موج‌گونه دودویی 10 بیت پس از رسیدن به شمارش‌های زیر متمم خواهد شد.
(الف) 1001100111 (ب) 0011111111 (پ) 1111111111

۶-۱۵ یک فلیپ فلاپ از لحظه لبه پالس تا متمم شدن خروجی دارای تأخیر 5ns است. حداکثر تأخیر در یک شمارنده موج‌گونه دودویی 10 بیت که از آن استفاده کند، چقدر است؟ حداکثر فرکانس شمارش مورد استفاده چقدر است؟

۶-۱۶ شمارنده موج‌گونه شکل ۶-۱۰ دارای چهار فلیپ فلاپ و 16 حالت است که تنها 10 حالت آن به‌کار رفته است. مدار را تحلیل کنید و حالت بعدی را برای هر یک از شش حالت به‌کار نرفته معین نمایید. اگر یک سیگنال پارازیت مدار را به حالات به‌کار نرفته بفرستد، چه پیش می‌آید.

۶-۱۷ یک شمارنده همزمان دودویی 4 بیتی را با فلیپ فلاپ‌های D طراحی نمایید.

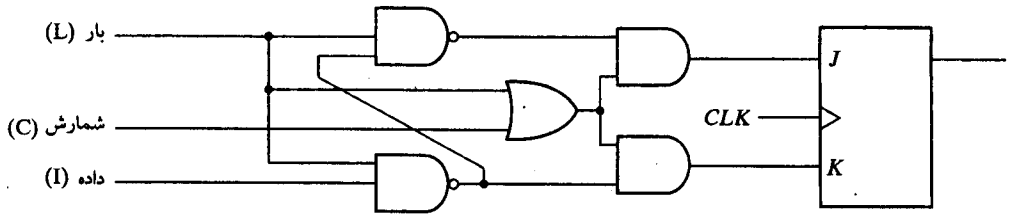
۶-۱۸ در بالا- پایین شمار شکل ۶-۱۳، وقتی هر دو ورودی بالا و پایین فعال شوند، چه رخ می‌دهد؟ مدار را طوری اصلاح کنید که هر دو ورودی 1 باشند ولی شمارنده تغییر حالت ندهد و در همان حال باقی بماند.

۶-۱۹ معادلات ورودی فلیپ فلاپ برای یک شمارنده BCD با فلیپ فلاپ‌های T در بخش ۶-۴ ملاحظه شد. معادلات ورودی را برای این شمارنده (الف) با فلیپ فلاپ‌های JK و (ب) فلیپ فلاپ‌های D بنویسید. سه طرح را مقایسه نمایید و بگویید کدام یک اقتصادی‌تر است.

۶-۲۰ شمارنده دودویی با بار شدن موازی شکل ۶-۱۴ را در یک نمودار بلوکی قرار دهید و همه ورودی‌ها و خروجی‌های آن را نشان دهید.
(الف) اتصالات چهار عدد از این بلوک‌ها را برای تولید یک شمارنده 16 بیت با بار شدن موازی نشان دهید.
(ب) یک شمارنده دودویی بسازید که از 0 تا 64 را بشمارد.

۶-۲۱ شمارنده شکل ۶-۱۴ دارای دو ورودی کنترل بار شدن (L) و شمارش (C) و یک ورودی داده ($I_i$) است.
(الف) معادلات ورودی فلیپ فلاپ را برای J و K از اولین طبقه برحسب L و C و I بنویسید.
(ب) نمودار منطقی اولین طبقه یک مدار مجتمع (74161) در شکل (م ۶-۲۱) دیده می‌شود. نشان دهید که این مدار معادل با مدار بخش (الف) است.

۶-۲۲ با استفاده از مدار شکل ۶-۱۴، سه روش دیگر برای شمارنده مد 12 را بدست آورید:




شکل (م ۲۱-۶)

(الف) با استفاده از گیت AND و ورودی بار شدن

(ب) با استفاده از نقلی خروجی

(پ) با استفاده از گیت NAND و ورودی پاک کردن غیرهمزمان

۲۳-۶ یک مدار زمانبندی که یک سیگنال خروجی را به مدت هشت پالس ساعت ثابت نگهدارد، طراحی نمایید. سیگنال شروع حالت 1 را خارج می‌کند و پس از هشت سیکل ساعت سیگنال به حالت 0 باز می‌گردد.

۲۴-۶ شمارنده‌ای را با فلیپ فلاپ‌های T بسازید تا رشته زیر را تکرار کند: 0، 1، 3، 7، 6، 4. نشان دهید که وقتی حالات 010 و 101 حالات بی‌اهمیتی تصور شوند، شمارنده به درستی کار نمی‌کند. راهی برای اصلاح آن بیابید.

۲۵-۶ لازم است تا شش سیگنال تکرار شونده $T_0$ تا $T_5$ را مشابه شکل ۱۷-۶ (پ) تولید کنیم. مدار را با توجه به موارد زیر طراحی نمایید.

(الف) فقط فلیپ فلاپ (ب) یک شمارنده و یک دیکدر

۲۶-۶ یک سیستم دیجیتال دارای مولد ساعتی است که پالس‌ها را با فرکانس 80MHz تولید می‌کند. مداری طراحی کنید که یک پالس ساعت با زمان سیکل 50ns را تولید نماید.

۲۷-۶ شمارنده‌ای با رشته دودویی 0، 1، 2، 3، 4، 5 و 6 طراحی کنید. از فلیپ فلاپ JK استفاده نمایید.

۲۸-۶ شمارنده‌ای که رشته 0، 1، 2، 4 و 6 را تکرار کند طراحی نمایید از فلیپ فلاپ D استفاده کنید.

۲۹-۶ هشت حالت به کار نرفته در شمارنده حلقوی دنباله چرخان شکل ۱۸-۶ (الف) را لیست کنید. حالت بعدی را برای هر یک از این حالات معین نمایید. نشان دهید که اگر شمارنده خود را در یک حالت نامعتبر بیابد، به حالت معتبر باز نمی‌گردد. مطابق متن کتاب آن را اصلاح کنید و نشان دهید که شمارنده حالات یکسانی را تولید می‌کند ضمن این که با رسیدن به حالت به کار نرفته، به یک حالت معتبر خواهد رفت.

۳۰-۶ نشان دهید که یک شمارنده جانسون با n فلیپ فلاپ یک رشته 2n حالتی تولید می‌کند. 10 حالت مربوط به شمارنده 5 فلیپ فلاپی را لیست نمایید و عبارت بولی هر یک از خروجی‌های 10 عدد گیت AND را معین کنید.

۳۱-۶ توصیف رفتاری و ساختاری HDL ثبات 4 بیت شکل ۱-۶ را بنویسید.

۳۲-۶ (الف) توصیف رفتاری HDL یک ثبات 4 بیت با بار شدن موازی و پاک شدن غیرهمزمان رابنویسید.

(ب) توصیف ساختاری HDL یک ثبات 4 بیت با بار شدن موازی در شکل ۲-۶ را بنویسید. از یک مولتی پلکسر 1 × 2 برای ورودی فلیپ فلاپ‌ها استفاده نمایید. ورودی پاک شدن را نیز لحاظ کنید.

(پ) هر دو توصیف را به کمک برنامه تست چک کنید.

۶-۳۳ برنامه محرک زیر برای شبیه‌سازی یک شمارنده دودویی با بار شدن موازی در مثال ۶-۳ HDL به کار رفته است. با توجه به برنامه، خروجی شمارنده و خروجی نقلی را از t = 0 تا t = 155ns پیش‌بینی کنید.

```
//Stimulus for testing counter
//of Example 6-3
     module testcounter;
       reg Count, Load, CLK, Clr;
       reg [3:0] IN;
       wire CO;
       wire [3:0] A;
       counter cnt (Count, Load, IN, CLK, Clr, A, CO);
       always
           #5 CLK = ~CLK;
         initial
          begin
           Clr = 0;
           CLK = 1;
           Load = 0; Count = 1;
             #5  Clr = 1;
             #50 Load = 1; IN = 4'b1100;
             #10 Load = 0;
             #70 Count = 0;
             #20 $finish;
          end
     endmodule
```

۶-۳۴ توصیف رفتاری HDL را بری شیفت رجیستر 4 بیت شکل ۶-۳ بنویسید.

۶-۳۵ توصیف رفتاری و ساختاری یک بالا- پایین‌شمار 4 بیت که در شکل ۶-۱۳ ملاحظه شد را بنویسید.

۶-۳۶ توصیف رفتاری یک بالا- پایین شمار با بار شدن موازی را برای ورودی‌های کنترل زیر بنویسید.

(الف) شمارنده سه خط کنترل برای سه عمل دارد: بالا، پایین و بار شدن. ترتیب اولویت عبارت است از: بار، بالا و پایین.

(ب) شمارنده برای چهار حالت بالا، پایین، بار و بدون تغییر دو خط انتخاب دارد.

۶-۳۷ توصیف HDL را برای یک شمارنده حلقوی 8 بیتی مشابه با شکل ۶-۱۷ (الف) بنویسید.

۶-۳۸ توصیف HDL را برای یک شمارنده حلقوی دنباله چرخان (شکل ۶-۱۸ (الف)) بنویسید.

۶-۳۹ توصیف رفتاری و ساختاری شمارنده شکل ۶-۱۶ را بنویسید.

## مراجع


1. MANO, M. M. and C. R. KIME. 2000. *Logic and Computer Design Fundamentals,* 2nd ed. Upper Saddle River, NJ: Prentice Hall.

2. NELSON V. P., H. T. NAGLE, J. D. IRWIN, and B. D. CARROLL. 1995. *Digital Logic Circuit Analysis and Design.* Upper Saddle River, NJ: Prentice Hall.

3. HAYES, J. P. 1993. *Introduction to Digital Logic Design.* Reading, MA: Addison-Wesley.



# حافظه و منطق برنامه‌پذیر

## ۷-۱ مقدمه

یک واحد حافظه وسیله‌ای است که اطلاعات دودویی جهت ذخیره شدن به آن منتقل و یا اطلاعاتی که برای پردازش لازم است از آن دریافت می‌شود. وقتی که پردازش داده رخ می‌دهد، اطلاعات از حافظه به ثبات‌های انتخابی در واحد پردازش انتقال می‌یابد. نتایج میانی و پایانی حاصل در واحد پردازش برای ذخیره شدن به واحد حافظه بازگردانده می‌شوند. اطلاعات دودویی واصله از وسیله ورودی هم در حافظه ذخیره می‌گردد و اطلاعات ارسالی به خروجی هم از حافظه برداشته می‌شود. واحد حافظه مجموعه‌ای از سلول‌هاست که قادر است حجم زیادی از اطلاعات دودویی را در خود جمع کند.

در سیستم‌های دیجیتال دو نوع حافظه مورد استفاده قرار می‌گیرد: حافظه با دستیابی تصادفی (RAM) و حافظه فقط خواندنی (ROM). RAM اطلاعات جدید را برای ذخیره می‌پذیرد تا بعد مورد استفاده قرار گیرد. عمل ذخیره کردن اطلاعات جدید در حافظه را عمل نوشتن در حافظه می‌گویند. فرآیند انتقال اطلاعات ذخیره شده در حافظه به بیرون را خواندن نامند. حافظه با دستیابی تصادفی هر دو عمل خواندن و نوشتن را انجام می‌دهد. حافظه فقط خواندنی عمل خواندن را اجرا می‌نماید. این بدان معنی است که اطلاعات دودویی ذخیره شده قبلی در حافظه در هر زمان قابل بازیابی است. با این وجود، اطلاعات موجود را نمی‌توان در حافظه فقط خواندنی نوشت، بلکه فقط می‌توان آن را خواند.

حافظه فقط خواندنی یک وسیله منطقی برنامه‌پذیر است. اطلاعات دودویی که در یک وسیله منطقی برنامه‌پذیر ذخیره می‌شود، به نوعی آرایش یافته و سپس در سخت‌افزار ذخیره می‌شود. این فرآیند را برنامه‌ریزی وسیله گویند. کلمه "برنامه‌ریزی" به یک روال سخت‌افزاری گویند که طی آن بیت‌ها در آرایش سخت‌افزار وسیله وارد می‌شوند.

حافظه فقط خواندنی (ROM) نوعی وسیله منطقی برنامه‌پذیر است. وسیله‌های دیگری از این نوع



(الف) نماد معمولی (ب) نماد منطقی آرایه‌ای

**شکل ۱-۷**. نمودارهای منطقی معمولی و آرایه‌ای برای گیت OR

عبارتند از آرایه منطقی برنامه‌پذیر (PLA)، منطق آرایه‌ای برنامه‌پذیر (PAL)، و آرایه گیتی برنامه‌پذیر موردی (FPGA). یک وسیله منطقی برنامه‌پذیر مداری مجتمع با گیت‌های منطقی داخلی است که از طریق مسیرهای الکترونیک به هم وصل شده و مثل فیوز عمل می‌کند. در ابتدای کار، همه فیوزها دست نخورده هستند. برنامه‌ریزی وسیله به معنی سوزاندن آن دسته از فیوزهاست که باید برای یک آرایش خاصی از تابع منطقی موردنظر، سوزانده شوند. در این فصل، آرایش وسایل منطقی برنامه‌پذیر را معرفی کرده و روال‌های استفاده از آنها را در طراحی سیستم‌های دیجیتال مشخص خواهیم کرد.

در بعضی از وسایل منطقی برنامه‌پذیر ممکن است صدها میلیون گیت مرتبط از طریق صدها هزار مسیر درونی به هم وصل شده باشند. برای این که نمودار منطقی داخلی را به فرم مختصر نشان دهیم، لازم است از یک سمبل گیت خاص قابل اعمال در منطق آرایه استفاده کنیم. شکل ۱-۷ سمبل معمول و آرایه‌ای را برای گیت OR چند ورودی نشان می‌دهد. به جای داشتن چندین خط ورودی به گیت، تنها یک خط به گیت وصل می‌کنیم. خطوط ورودی عمود بر این خط بوده و از طریق فیوزها به آن وصل می‌شوند. به طریقی مشابه می‌توانیم منطق آرایه را برای گیت AND ترسیم کنیم. این نوع نمایش گرافیکی برای ورودی گیت‌ها در ترسیم نمودارهای منطقی آرایه‌ای به کار خواهد رفت.

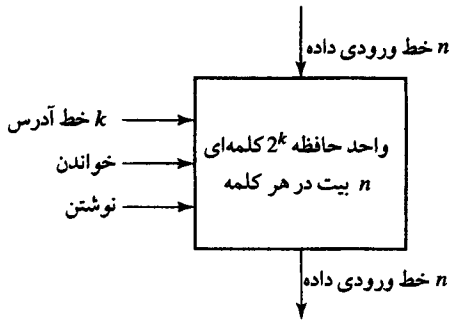
## ۷-۲ حافظه با دستیابی تصادفی RAM

واحد حافظه مجموعه‌ای از سلول‌های ذخیره‌سازی به همراه مدارهای مربوطه لازم برای انتقال اطلاعات به داخل یا به خارج وسیله است. زمان لازم برای هر انتقال اطلاعات و یا از هر مکان موردنظر همواره یکسان است، و به این علت نام حافظه با دستیابی تصادفی به صورت خلاصه RAM به کار رفته است.

یک واحد حافظه اطلاعات دودویی را به صورت گروهی از بیت‌ها به نام کلمه ذخیره می‌نماید. کلمه حافظه گروهی از بیت‌هاست که به صورت واحد حافظه وارد یا خارج می‌شود. کلمه حافظه ترکیبی از 1ها و 0هاست و ممکن است، یک عدد، یک دستور، یک یا چند کاراکتر الفبا عددی، یا هر اطلاعات دودویی کد شده را نمایش دهد. یک گروه هشت بیتی را بایت نامند.

بسیاری از کامپیوترها کلماتی را به کار می‌برند که طولشان مضربی از هشت بیت، یا بایت است. بنابراین کلمه 16 بیتی دو بایت، کلمه 32 بیتی از چهار بایت ساخته می‌شود. ظرفیت یک واحد حافظه براساس تعداد کل بایت‌هایی است که می‌تواند ذخیره کند.

محاوره بین حافظه و محیط از طریق خطوط داده ورودی و خروجی، خطوط انتخاب آدرس و خطوط کنترلی که جهت انتقال را مشخص می‌کند، صورت می‌گیرد. نمودار بلوکی واحد حافظه در




**شکل ۲-۷**. دیاگرام بلوکی یک واحد حافظه

شکل ۲-۷ ملاحظه می‌گردد. n خط داده ورودی اطلاعات ذخیره شونده را مهیا می‌سازند و n خط داده خروجی هم اطلاعات خروجی را تهیه می‌نمایند. k خط آدرس کلمه خاص انتخابی را در میان چند کلمه مشخص می‌نماید. دو ورودی کنترل، جهت انتقال موردنظر را تعیین می‌کنند. ورودی نوشتن موجب می‌شود تا وارده به داخل حافظه برود و ورودی خواندن سبب انتقال داده به خارج از حافظه می‌گردد.

واحد حافظه با تعداد کلمات و تعداد بیت‌ها در هر کلمه مشخص می‌شود. خطوط آدرس یک کلمه خاص را انتخاب می‌کنند. به هر کلمه در حافظه یک عدد شناسایی به نام آدرس تخصیص می‌یابد، که از 0 شروع و تا $2^k-1$ ادامه دارد و در آن k تعداد خطوط آدرس است. انتخاب یک کلمه خاص در حافظه با اعمال آدرس k بیتی به خطوط آدرس صورت می‌گیرد. یک دیکدر این آدرس را پذیرفته و مسیرهای لازم را برای انتخاب کلمه موردنظر باز می‌کند. حافظه‌ها از لحاظ سایز بسیار متغیرند و ممکن است از 1024 کلمه، که به 10 خط آدرس نیاز دارد، تا $2^{32}$ کلمه که 32 خط آدرس را داراست در دسترس باشند. مرسوم است که تعداد کلمات (یا بایت) در حافظه با یکی از حروف K (کیلو)، M (مگا)، یا G (گیگا) ذکر شود. K برابر $2^{10}$، M برابر با $2^{20}$ و G برابر $2^{30}$ است. بنابراین $64K=2^{16}$، $2M=2^{21}$ و $4G=2^{32}$ خواهد بود.

به عنوان مثال واحد حافظه‌ای با ظرفیت 2K کلمه و 16 بیت در هر کلمه را در نظر بگیرید. چون $1K = 1024 = 2^{10}$ و 16 بیت هم دو بایت را تشکیل می‌دهد، می‌توان گفت که حافظه قادر است $2K = 2048$ بایت را در خود جای دهد. شکل ۳-۷ محتوای سه کلمه اول و سه کلمه آخر این حافظه را

| آدرس حافظه | | محتویات حافظه |
|---|---|---|
| دودویی | دهدهی | |
| 0000000000 | 0 | 1011010101011101 |
| 0000000001 | 1 | 1010101110001001 |
| 0000000010 | 2 | 0000110101000110 |
| ⋮ | ⋮ | ⋮ |
| 1111111101 | 1021 | 1001110100010100 |
| 1111111110 | 1022 | 0000110100011110 |
| 1111111111 | 1023 | 1101111000100101 |

**شکل ۳-۷**. محتویات یک حافظهٔ 16×1024

نشان می‌دهد. کلمات با آدرس دهدهی از 0 تا 1023 شناخته می‌شوند. آدرس دودویی معادل شامل 10 بیت است. اولین آدرس با ده عدد 0 مشخص می‌شود و آخرین آدرس هم با ده عدد 1 معین می‌گردد. دلیل این است که 1023 در دودویی برابر 1111111111 می‌باشد. یک کلمه در حافظه با آدرس دودویی‌اش انتخاب می‌گردد. وقتی کلمه‌ای نوشته و یا خوانده شود، حافظه روی هر 16 واحد به صورت یکجا عمل می‌نماید.

حافظه 10 × 64K دارای 10 بیت خط آدرس و 16 بیت در هر کلمه است. به عنوان مثالی دیگر، یک حافظه 10 × 64K دارای 16 بیت آدرس (چون $64K = 2^{16}$ است) و هر کلمه متشکل از 10 بیت می‌باشد. تعداد بیت‌های آدرس در حافظه وابسته به تعداد کل کلماتی است که قابل ذخیره شدن در حافظه می‌باشد و مستقل از تعداد بیت‌های هر کلمه است. تعداد بیت‌ها در آدرس از رابطه $2^k \geq m$ بدست می‌آید، که m تعداد کل کلمات و k تعداد بیت‌های آدرس لازم برای برقراری رابطه فوق است.

## اعمال نوشتن و خواندن

دو عملی که یک حافظه RAM می‌تواند انجام دهد عبارتند از نوشتن و خواندن. سیگنال نوشتن عمل انتقال به داخل و سیگنال خواندن یک عمل انتقال به بیرون را مشخص می‌نماید. برای پذیرش یکی از این سیگنال‌های کنترل، مدارهای درونی داخل حافظه عمل موردنظر را مهیا می‌سازند. مراحل مربوط به وارد کردن یک کلمه به حافظه به قرار زیر است:

۱- آدرس دودویی کلمه موردنظر به خطوط آدرس اعمال می‌گردد.

۲- بیت‌های داده‌ای که باید در حافظه ذخیره شوند به خطوط ورودی داده اعمال می‌گردند.

۳- ورودی نوشتن فعال می‌شود.

آنگاه واحد حافظه بیت‌ها را از خطوط ورودی داده برداشته و آنها را در مکانی که به وسیله خطوط آدرس معین می‌گردد ذخیره می‌نماید.

مراحل مربوط به خارج کردن یک کلمه از حافظه به قرار زیر است.

۱- آدرس دودویی کلمه موردنظر به خطوط آدرس اعمال می‌شود.

۲- ورودی خواندن فعال می‌گردد.

آنگاه واحد حافظه کلمه انتخابی به وسیله آدرس را برداشته و به خطوط داده خروجی اعمال می‌کند. محتوای کلمه انتخابی پس از خوانده شدن تغییر نمی‌نماید.

گاهی قطعات حافظه تجاری موجود در تراشه‌های مدار مجتمع دو خط کنترل خواندن و نوشتن را با آرایش‌های متفاوت دارا هستند. در عوض داشتن دو خط خواندن و نوشتن جداگانه برای کنترل دو عمل، بسیاری از مدارهای مجتمع دو ورودی کنترل مجزا دارند: یکی ورودی قطعه را انتخاب می‌کند و دیگری عمل آن را کنترل می‌نماید. عملیات حافظه‌ای که از این ورودی‌های کنترل حاصل می‌شود در جدول ۱-۷ مشخص شده‌اند.

فعال‌ساز حافظه (گاهی به آن انتخاب‌گر تراشه گویند) برای توانا کردن یک تراشه خاص حافظه از میان چند تراشه موجود به کار می‌رود. وقتی که فعال‌ساز حافظه غیرفعال باشد، تراشه حافظه انتخاب

جدول ۱-۷. ورودی‌های کنترل به تراشه حافظه

| عملکرد حافظه | نوشتن / خواندن | فعال ساز حافظه |
|---|---|---|
| هیچ | X | 0 |
| نوشتن در کلمه منتخب | 0 | 1 |
| خواندن از کلمه منتخب | 1 | 1 |

نشده و هیچ عملی اجرا نمی‌گردد. پس از فعال شدن ورودی نوشتن/ خواندن عمل مورد اجرا را مشخص خواهد کرد.

## توصیف حافظه در HDL

حافظه در Verilog HDL به وسیله آرایه‌ای از ثبات‌ها مدل‌سازی می‌شود. حافظه با کلمه کلید reg و با استفاده از آرایه دوبعدی اعلان می‌گردد. اولین عدد در آرایه تعداد بیت‌ها در یک کلمه و دومی تعداد کلمات در حافظه است. مثلاً یک حافظه 1024 کلمه‌ای با 16 بیت در هر کلمه به صورت زیر اعلان می‌شود.

```
reg[15:0] memword[0:1023];
```

این عبارت یک آرایه 1024 ثباتی را با 16 بیت در هر یک توصیف می‌کند. عدد داخل memword بیانگر تعداد کل کلمات حافظه بوده و برابر با آدرس حافظه است. مثلاً memword[512] به کلمه حافظه‌ای 16 بیتی در آدرس 512 اشاره می‌نماید. عمل یک واحد حافظه در مثال ۱-۷ HDL تشریح شده است. حافظه دارای 64 کلمه چهار بیتی است. دو ورودی کنترل وجود دارد: ReadWrite و Enable. خطوط DataIn و DataOut هر یک چهار بیتی‌اند. ورودی Address باید شش بیتی ($2^6 = 64$)

مثال ۱-۷، HDL

```
//Read and write operations of memory.
//Memory size is 64 words of 4 bits each.
 module memory (Enable,ReadWrite,Address,DataIn,DataOut);
    input  Enable,ReadWrite;
    input [3:0] DataIn;
    input [5:0] Address;
    output [3:0] DataOut;
    reg [3:0] DataOut;
    reg [3:0] Mem [0:63];          //64 x 4 memory
    always @ (Enable or ReadWrite)
      if (Enable)
      if (ReadWrite)
      DataOut = Mem[Address];  //Read
      else
      Mem[Address] = DataIn;   //Write
      else DataOut = 4'bz;          //High impedance state
endmodule
```

باشد، حافظه به صورت یک آرایه ثباتی دوبعدی با Mem اعلان می‌شود که آدرس 64 کلمه را مشخص می‌نماید. عملیات حافظه هنگامی شروع می‌شود که ورودی Enable فعال باشد. ورودی ReadWrite نوع عمل را مشخص می‌سازد. اگر ReadWrite = 1 باشد، حافظه عمل خواندن را با عبارت زیر نمادین می‌کند.

DataOut ← Mem [Address];

این عمل موجب می‌شود تا چهار بیت کلمه حافظه انتخابی مشخص شده با Address به خطوط DataOut منتقل شود. اگر ReadWrite = 0 باشد، حافظه عمل نوشتن را که با عبارت زیر نشان داده شده اجرا خواهد کرد.

Mem [Address] ← DataIn;

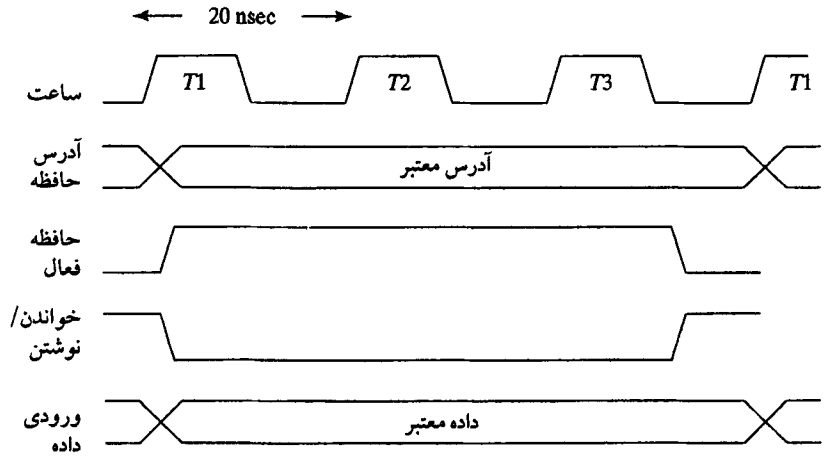
این عبارت سبب انتقال 4 بیت از خطوط DataIn به داخل کلمه حافظه انتخاب شده با Address می‌شود. وقتی که Enable=0 شود، حافظه از کار افتاده و فرض می‌شود که خروجی‌ها به حالت امپدانس بالا می‌روند. این حالت با کلمه کلیدی z مشخص شده و نشان می‌دهد که حافظه دارای خروجی‌های سه حالته است.

## شکل موج‌های زمان‌بندی

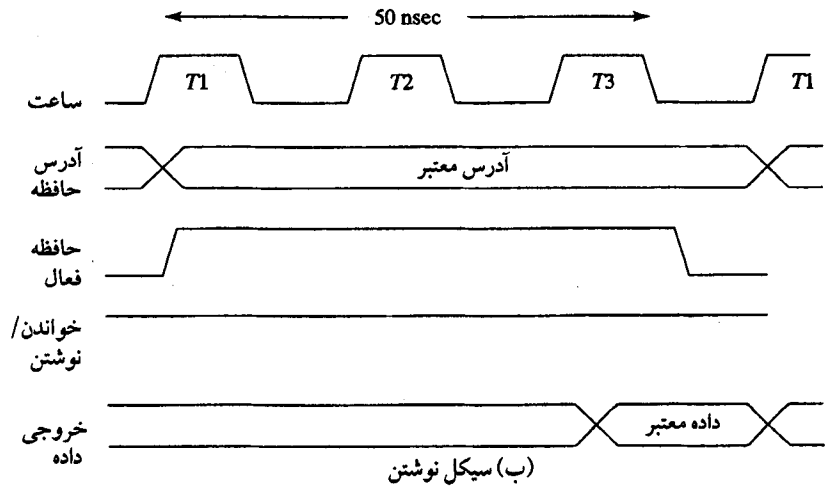
عملکرد یک واحد حافظه با یک وسیله خارجی مانند واحد مرکزی پردازش (CPU) کنترل می‌گردد. CPU معمولاً با ساعتش معرفی می‌شود. با این وجود، حافظه دارای ساعت درونی نیست. عمل خواندن و نوشتن نیز با خطوط کنترل مشخص می‌گردد. زمان دستیابی یک حافظه زمانی است که برای انتخاب یک کلمه و خواندن آن لازم است. زمان سیکل یک حافظه زمان لازم برای تکمیل نوشتن می‌باشد. CPU باید سیگنال‌های کنترل حافظه را تولید کند به نحوی که عملیات ساعت‌دار داخلی با عملیات خواندن و نوشتن همزمان شود. این بدان معنی است که زمان دستیابی و زمان سیکل یک حافظه باید در فاصله‌ای از زمان که مضربی از سیکل‌های ساعت CPU است قرار گیرند.

به عنوان مثال فرض کنید که یک CPU با یک ساعت 50MHz کار کند که در نتیجه پریود آن 20ns خواهد شد. اکنون تصور کنید که CPU با حافظه‌ای تبادل داده نماید که زمان دستیابی‌اش و زمان سیکل آن از 50ns تجاوز نکند. این بدان معنی است که سیکل نوشتن کلمه در حافظه باید در یک دوره 50ns قرار داشته باشد و نیز سیکل خواندن، داده انتخابی را در 50ns یا کمتر خارج کند. (البته دو عدد همیشه یکی نیستند). چون پریود سیکل CPU برابر 20ns است، لازم است دو ونیم و یا سه پالس ساعت برای هر تقاضای حافظه به کار گرفته شود.

زمانبندی حافظه که در شکل ۴-۷ دیده می‌شود برای یک CPU با ساعت 50MHz و حافظه‌ای با زمان سیکل حداکثر 50ns ترسیم شده است. سیکل نوشتن در بخش (الف) سه سیکل 20ns، $T_1$، $T_2$ و $T_3$ را نشان می‌دهد. برای عمل نوشتن، CPU آدرس و داده ورودی را برای حافظه فراهم می‌نماید. این کار در آغاز $T_1$ انجام می‌شود. (دو خط متقاطع در خطوط آدرس و داده، تغییر احتمالی در مقدار خطوط چندگانه را بیان می‌دارد.) پس از تثبیت سیگنال‌های آدرس، سیگنال‌های فعال‌ساز و نوشتن/ خواندن




(الف) سیکل خواندن




(ب) سیکل نوشتن

**شکل ۴-۷**. شکل موج‌های زمانبندی سیکل حافظه

فعال می‌گردند تا به این ترتیب داده‌ها در دیگر حافظه‌ها تخریب نگردند. سیگنال فعال‌ساز حافظه به سطح بالا و سیگنال خواندن/ نوشتن به سطح پایین سوئیچ می‌کنند تا عمل نوشتن انجام شود. دو سیگنال کنترل باید حداقل به مدت 50ns با ثُبات باقی بمانند. سیگنال‌های داده و آدرس باید برای مدت کوتاهی پس از غیرفعال شدن سیگنال‌های کنترل، بدون تغییر باقی بمانند. در پایان سومین سیکل ساعت، عمل نوشتن در حافظه تکمیل شده و CPU می‌تواند در سیکل $T_1$ بعدی دوباره به حافظه دست یابد.

سیکل خواندن در شکل ۴-۷ (ب) دارای آدرسی است که به وسیله CPU برای حافظه مهیا شده

است. سیگنال‌های فعال‌ساز حافظه و خواندن/ نوشتن برای خواندن باید در سطح بالا باشند. حافظه باید، داده کلمه انتخابی به وسیله آدرس را در کمتر از 50ns از لحظه فعال شدن حافظه، روی خطوط داده خروجی قرار دهد. CPU در حین گذر منفی $T_3$ ، داده را به یکی از ثبات‌های داخلی انتقال می‌دهد. سیکل $T_1$ بعدی برای یک تقاضای حافظه دیگر در اختیار است.

## انواع حافظه‌ها

طرز دستیابی به یک سیستم حافظه به نوع قطعات به‌کار رفته بستگی دارد. در یک حافظه RAM ، مکان‌های حافظه را می‌توان در فضا مجزا از یکدیگر تصور کرد، که در آن هر کلمه یک مکان خاص را اشغال کرده است. در یک حافظه با دستیابی ترتیبی، اطلاعات ذخیره شده در محیط بلافاصله قابل دسترسی نیستند و بلکه پس از مدتی بدست می‌آیند. یک دیسک یا نوار مغناطیسی حافظه‌ای از این نوع است. هر مکان از حافظه از زیر هدهای خواندن و نوشتن به ترتیب عبور می‌کنند، ولی اطلاعات فقط وقتی خوانده می‌شود که کلمه مورد تقاضا فرا برسد. در یک حافظه RAM ، زمان دستیابی مستقل از مکان یک کلمه خاص بوده و همیشه کوتاه است. در حافظه با دستیابی ترتیبی، زمان دستیابی به یک کلمه به محل کلمه نسبت به محل هد خواندن وابسته است، بنابراین زمان دستیابی متغیر خواهد بود.

واحدهای مدار مجتمع RAM در دو نوع موجودند: نوع استاتیک و نوع دینامیک. RAM استاتیک (SRAM) اساساً از لچ‌های درونی تشکیل شده و اطلاعات دودویی را ذخیره می‌نمایند. اطلاعات ذخیره شده مادامی که تغذیه به واحد اعمال شده باشد معتبر خواهد ماند. RAM دینامیک (DRAM) اطلاعات دودویی را به فرم بارهای الکتریکی در خازن ذخیره می‌نماید. خازن‌ها به وسیله ترانزیستورهای MOS در داخل تراشه ساخته می‌شوند. بار ذخیره شده در خازن‌ها به تدریج تخلیه می‌گردد و بنابراین با تازه‌سازی حافظه دینامیکی باید خازن‌ها را بازسازی کرد. عمل تازه‌سازی هر چند میلی ثانیه یک بار با چرخش در کلمات برای بازسازی خازن‌ها انجام می‌شود. حافظه DRAM مصرف انرژی کمتر و ظرفیت ذخیره‌سازی بیشتری در تراشه حافظه دارد. SRAM کاربرد ساده‌تر و سیکل خواندن و نوشتن کوتاه‌تری را داراست.

واحدهای حافظه‌ای راکه به هنگام قطع برق اطلاعات خود را از دست می‌دهند حافظه فرّار می‌گویند. مدارهای مجتمع RAM ، چه استاتیک و چه دینامیک، از این رده محسوب می‌شوند زیرا سلول‌های دودویی برای حفظ اطلاعات نیاز به تغذیه بیرونی دارند. برعکس، یک حافظه غیر فرّار، مانند دیسک مغناطیسی، اطلاعات ذخیره شده خود را حتی بعد از حذف تغذیه حفظ می‌نماید. دلیل این است که داده ذخیره شده در عناصر مغناطیسی با جهت مغناطیسی شدن آن تعیین می‌گردد، که پس از حذف منبع تغذیه، باز هم باقی خواهند ماند. نوع دیگر حافظه غیر فرّار، حافظه فقط خواندنی ROM است. خاصیت غیر فرّاری در کامپیوترهای دیجیتال به هنگام خاموشی لازم می‌باشد. داده‌ها و برنامه‌هایی که تغییر نمی‌کنند در ROM ذخیره می‌شوند. برنامه‌های بزرگ معمولاً روی دیسک‌های مغناطیسی نگهداری می‌شود. وقتی که تغذیه روشن شود، کامپیوتر برنامه‌های ROM را می‌تواند استفاده کند. دیگر

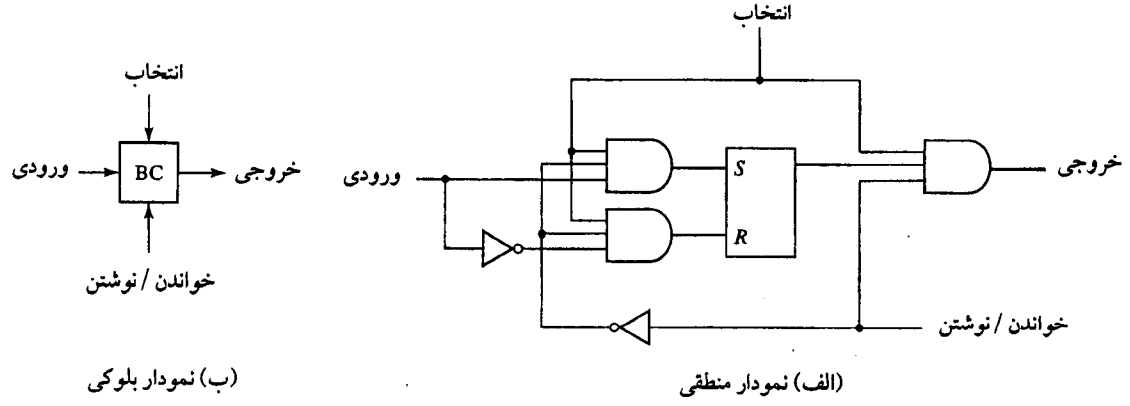
برنامه‌های موجود در یک دیسک مغناطیسی پس از آن به RAM کامپیوتر انتقال می‌یابد. قبل از خاموش کردن کامپیوتر اطلاعات موجود در RAM جهت حفظ و نگهداری به دیسک منتقل می‌گردد.

## ۷-۳ دیکدکردن حافظه

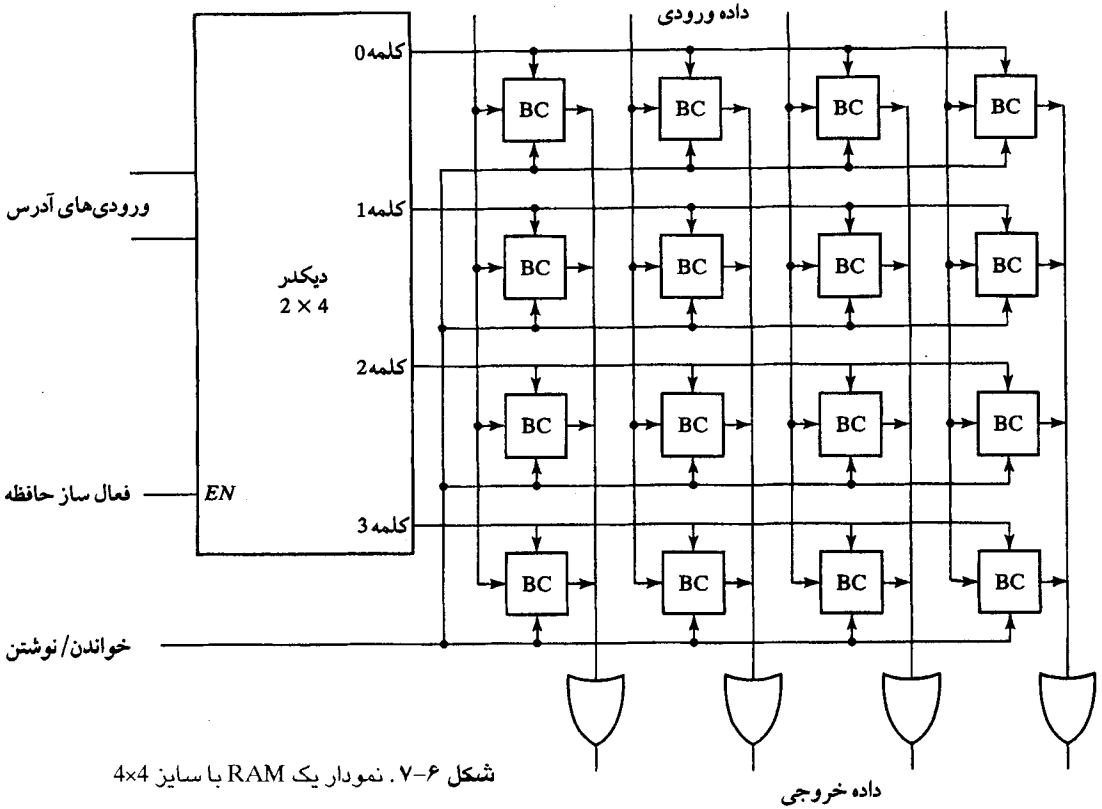
علاوه بر عناصر ذخیره‌سازی در یک واحد حافظه، برای انتخاب کلمه حافظه مشخص شده با آدرس ورودی، به یک دیکدر نیاز است. در این بخش ساختمان داخلی یک حافظه با دستیابی تصادفی را ارائه می‌کنیم و عملکرد دیکدر را نشان خواهیم داد. برای این که بتوان تمام حافظه را در یک نمودار جای داد واحد حافظه دارای 16 بیت حافظه با 4 بیت در هر کلمه فرض شده است. مثالی از آرایش دیکد کردن دو بعدی، برای درک بهتر حافظه‌های بزرگ‌تر آورده شده است. آنگاه مولتی پلکس کردن آدرس را که در مدارهای مجتمع DRAM به کار می‌رود نشان خواهیم داد.

### ساختار درونی

ساختار درونی یک حافظه با دستیابی تصادفی m کلمه‌ای با n بیت در هر کلمه از یک مجموعه ذخیره‌سازی $m \times n$ و مدارهای دیکدکردن برای انتخاب هر کلمه تشکیل شده است. سلول ذخیره‌سازی دودویی بلوک ساختاری پایه واحد حافظه است. مدار معادل یک سلول دودویی که یک بیت از اطلاعات را ذخیره کند در شکل ۵-۷ نشان داده شده است. بخش ذخیره‌سازی آن نیز یک لچ SR به همراه گیت‌های مربوطه است. در واقع سلول یک مدار الکترونیکی ساخته شده از چهار تا شش ترانزیستور می‌باشد. با این وجود بهتر است که آن را با سمبل‌های منطقی مدل‌سازی نماییم. یک سلول ذخیره‌سازی دودویی باید خیلی کوچک باشد تا بتوان سلول‌های بیشتری را در یک فضای کوچک در تراشه جای داد. سلول دودویی یک بیت را در هر لچ داخلی ذخیره می‌نماید. ورودی انتخاب، سلول را برای خواندن یا نوشتن فعال می‌کند و ورودی خواندن/ نوشتن نوع عملکرد سلول را پس از انتخاب تعیین می‌نماید. یک 1 در ورودی خواندن/ نوشتن عمل خواندن را با ایجاد مسیری از لچ به پایانه بیرونی فراهم می‌نماید. یک 0 در ورودی خواندن/ نوشتن عمل نوشتن را با ایجاد مسیری از پایانه ورودی به لچ ممکن می‌سازد.




شکل ۵-۷. سلول حافظه




**شکل ۶-۷**. نمودار یک RAM با سایز 4×4

ساختار منطقی یک RAM کوچک در شکل ۶-۷ نشان داده شده است. این مدار چهار کلمه چهار بیت و جمعاً 16 سلول دودویی دارد. بلوک‌های کوچک برچسب خورده با BC، سلول دودویی را با سه ورودی و یک خروجی طبق شکل ۵-۷ (ب) نمایش می‌دهند. یک حافظه چهار کلمه‌ای به دو خط آدرس نیاز دارد. دو ورودی آدرس از یک دیکدر 4 × 2 برای انتخاب یکی از چهار کلمه عبور می‌نمایند. دیکدر با ورودی فعال‌ساز حافظه، فعال می‌شود. وقتی فعال‌ساز حافظه 0 باشد، همه خروجی‌های دیکدر 0 و هیچ یک از کلمات حافظه انتخاب نمی‌شود. با 1 شدن انتخاب حافظه بنا به مقدار دو خط آدرس، یکی از چهار کلمه حافظه انتخاب می‌گردد. پس از انتخاب کلمه، ورودی خواندن/ نوشتن عمل را معین می‌نماید. در حین خواندن، چهار بیت کلمه انتخابی از گیت‌های OR به پایانه‌های خروجی منتقل می‌شوند (توجه کنید که گیت‌های OR مطابق با قرارداد منطق آرایه‌ای شکل ۱-۷ ترسیم شده‌اند). در هنگام نوشتن، داده موجود در خطوط ورودی به چهار سلول دودویی کلمه انتخابی وارد می‌گردند. سلول‌های دودویی انتخاب نشده غیرفعال شده و مقدار قبلی آنها دست نخورده باقی می‌ماند. وقتی ورودی انتخاب حافظه وارده به دیکدر 0 شود، هیچ یک از کلمات انتخاب نمی‌شوند و محتوای همه

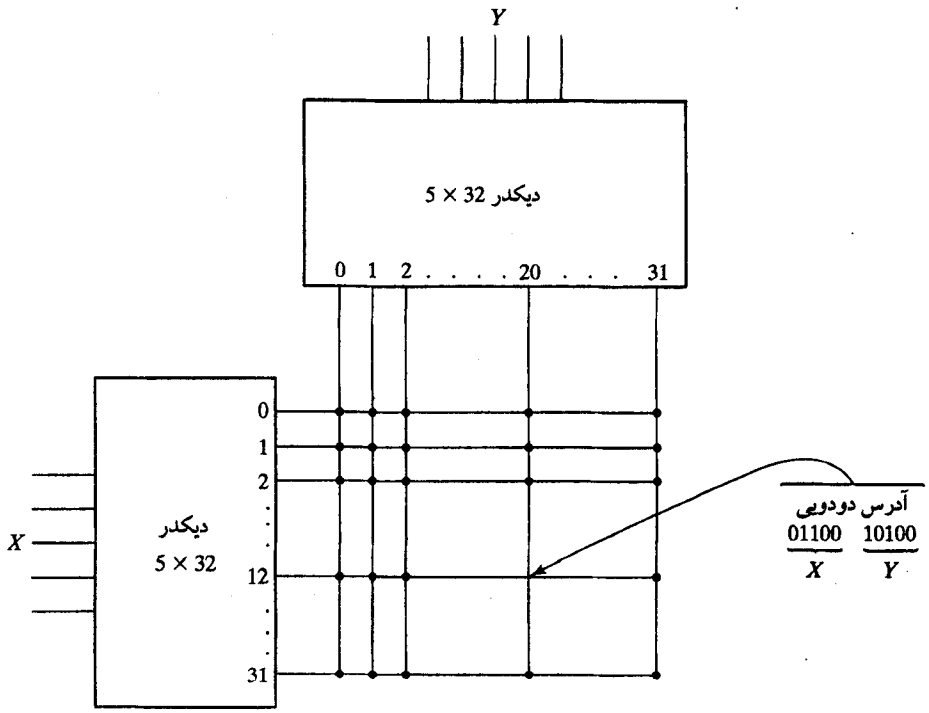
سلول‌ها مستقل از مقدار ورودی خواندن/ نوشتن بدون تغییر باقی می‌مانند.

حافظه‌های RAM تجاری می‌توانند تا هزاران کلمه در هر تراشه و از 1 تا 64 بیت در هر کلمه داشته باشند. ساختار منطقی یک حافظه با ظرفیت بالا توسعه مستقیمی از آرایش شکل ۶-۷ است. حافظه‌ای با $2^k$ کلمه n بیتی در هر کلمه نیاز به k خط آدرس دارد تا وارد دیکدر $k \times 2^k$ گردد. هر یک از خروجی‌های دیکدر یک کلمه n بیتی را برای نوشتن و یا خواندن انتخاب می‌نماید.

## دیکد کردن متقارن

یک دیکدر با k ورودی و $2^k$ خروجی به $2^k$ گیت AND و k ورودی در هر گیت نیاز دارد. با انتخاب طرح دوبعدی و استفاده از دو دیکدر می‌توان تعداد کل گیت‌ها و تعداد ورودی‌های گیت‌ها را کاهش داد. ایده اولیه در دیکد کردن دوبعدی، آرایش سلول‌های حافظه در یک آرایه است که تا حد ممکن به یک مربع نزدیک باشد. در این آرایش، دو دیکدر با k/2 ورودی در عوض دیکدر k ورودی به کار می‌رود. یک دیکدر انتخاب سطر و دیگری انتخاب ستون را در آرایش ماتریسی دوبعدی به عهده دارد.

الگوی انتخاب دوبعدی برای یک حافظه 1k کلمه‌ای در شکل ۷-۷ نشان داده شده است. در عوض استفاده از یک دیکدر 1024 × 10، دو دیکدر 32 × 5 به کار رفته است. هنگام استفاده از دو دیکدر، 64 گیت AND و با 5 ورودی در هر کدام لازم می‌باشد. پنج بیت باارزش‌تر آدرس به ورودی X و پنج بیت




**شکل ۷-۷**. ساختار دیکد کردن دو بعدی برای حاففظه ۱K کلمه‌ای

کم‌ارزش‌تر به ورودی Y می‌رود. هر کلمه در آرایه حافظه با تلاقی یکی از 32 سطر، با یکی از 32 ستون از 1024 کلمه موجود، انتخاب می‌گردد. توجه کنید که هر تقاطع یک کلمه را با هر تعداد بیت نشان می‌دهد.

به عنوان مثال، کلمه‌ای با آدرس 404 را در نظر بگیرید. معادل دودویی 10 بیت عدد 404 برابر با 10100 01100 است. در نتیجه X = 01100 (12 دودویی) و Y = 10100 (دودویی 20) خواهد بود. کلمه n بیتی انتخابی در خروجی شماره 12 دیکدر X و خروجی شماره 20 دیکدر Y قرار دارد. همه بیت‌های کلمه برای خواندن یا نوشتن به کار می‌رود.

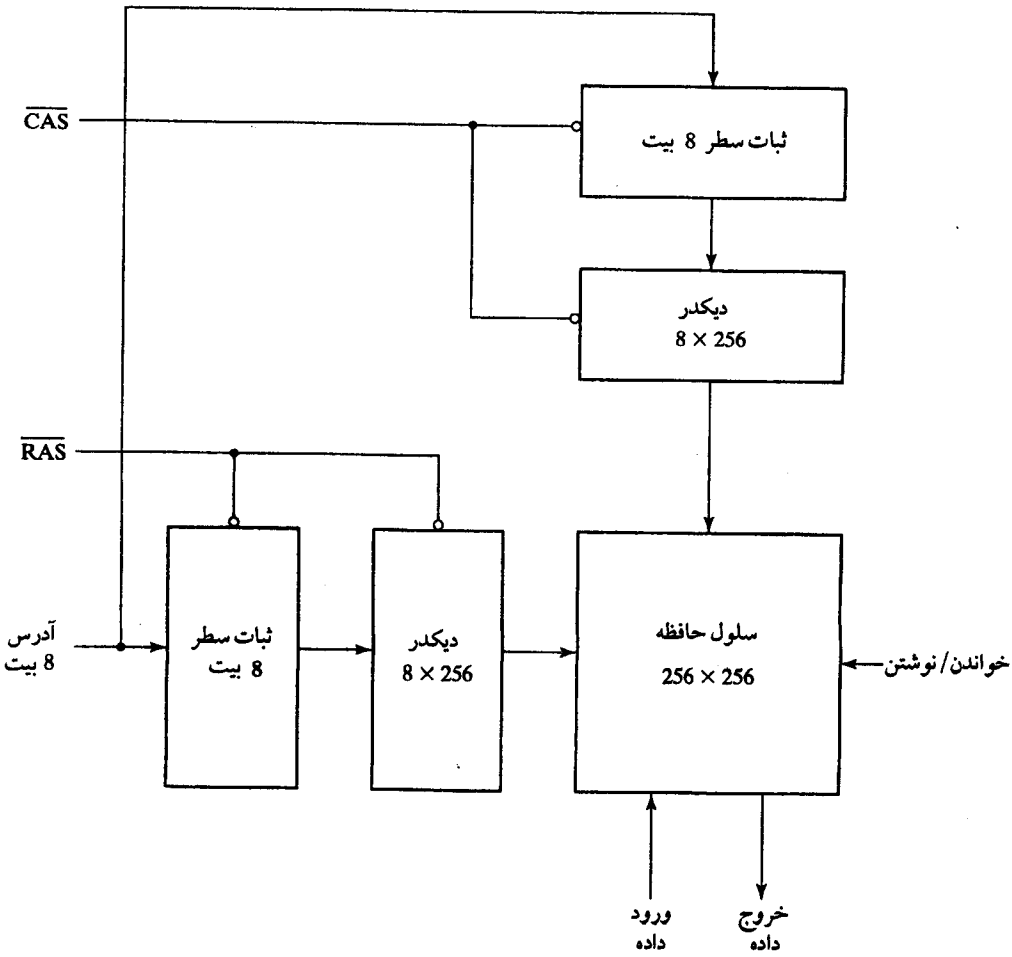
## مولتی پلکس کردن آدرس

سلول حافظه SRAM که در شکل ۵-۷ مدل‌سازی شد معمولاً دارای شش ترانزیستور است. در ساخت حافظه‌های با ظرفیت بالاتر، لازم است تعداد ترانزیستورها را در هر سلول کاهش دهیم. سلول DRAM دارای یک ترانزیستور MOS و یک خازن است. بار ذخیره شده روی خازن با زمان تخلیه می‌شود و بنابراین حافظه به صورت پریودیک باید تازه‌سازی گردد. در DRAM‌ها به دلیل سادگی ساختار، چگالی سلول‌ها چهار برابر چگالی SRAM است. این قابلیت موجب می‌شود تا ظرفیت آن نسبت به حافظه SRAM چهار برابر گردد. قیمت هر بیت DRAM سه یا چهار برابر کمتر از SRAM می‌باشد. صرفه‌جویی دیگر به دلیل توان لازم کمتر در سلول‌های DRAM است. این مزایا موجب شده است تا DRAM در حافظه‌های بزرگ، تکنولوژی ارجح باشد. تراشه‌های DRAM با ظرفیت‌های 64K تا 256M بیت وجود دارند. بسیاری از DRAM‌ها سایز کلمه 1 بیتی دارند، بنابراین برای افزایش سایز کلمه باید چندین تراشه با هم ترکیب گردند.

به دلیل ظرفیت زیاد، دیکد کردن آدرس به صورت آرایه دوبعدی بوده و حافظه‌های بزرگتر حتی چندین آرایه دارند. برای کاهش تعداد پایه‌های بسته تراشه، طراحان از مولتی پلکس کردن آدرس استفاده می‌کنند که به وسیله یک مجموعه از پایه‌های ورودی آدرس عناصر آدرس را در خود جای دهند. در آرایه دوبعدی آدرس به دو بخش تقسیم شده و در زمان‌های مختلف بخش سطر آدرس ابتدا و سپس بخش ستون اعمال می‌گردند. چون برای هر دو بخش آدرس از یک مجموعه پایه استفاده می‌شود، سایز بسته به شدت کاهش می‌یابد.

برای تشریح ایده مولتی پلکس کردن از حافظه 64K کلمه‌ای استفاده شده است. نمودار آرایش دیکد کردن در شکل ۸-۷ دیده می‌شود. حافظه متشکل از آرایه دوبعدی سلول‌ها با 256 سطر در 256 ستون می‌باشد که جمعاً $64K = 2^{16} = 2^8 \times 2^8$ خواهد بود. تنها یک خط ورود داده، و یک کنترل خواندن نوشتن وجود دارد. هشت بیت ورودی آدرس و دو آگاه‌گر آدرس موجود است. آگاه‌گرهای آدرس برای فعال کردن ثبات‌های مربوطه‌شان در سطر و ستون در نظر گرفته شده‌اند. آگاه‌گر آدرس سطر (RAS)، ثبات 8 بیت سطر را فعال می‌کند و آگاه‌گر ستون CAS، ثبات 8 بیتی ستون را فعال می‌نماید. خط بار روی سمبل آگاه‌گر به معنی فعال شدن ثبات‌ها در سطح منفی سیگنال است.

16 بیت آدرس با استفاده از RAS و CAS طی دو مرحله به DRAM اعمال می‌شود. در آغاز هر دو




**شکل ۸-۷**. مولتی پلکس کردن آدرس برای یک 64K DRAM

آگاه‌گر در حالت 1 هستند. آدرس سطر 8 بیتی به ورودی‌های RAS اعمال شده و RAS = 0 می‌گردد. این موجب خواهد شد تا آدرس سطر در ثبات آدرس سطر بار شود. RAS دیکدر سطر را هم فعال می‌کند به نحوی که آدرس سطر دیکد شده و یکی از سطرهای آرایه انتخاب گردد. پس از گذشت زمان نشست انتخاب سطر، RAS به سطح 1 باز می‌گردد. سپس آدرس ستون 8 بیتی به ورودی‌های آدرس اعمال شده و CAS به 0 برده می‌شود. این موجب می‌گردد تا آدرس ستون در ثبات آدرس ستون بار شود و دیکدر ستون را فعال نماید. در این هنگام دو بخش آدرس در ثبات‌های مربوطه هستند، و دیکدرها آنها را رمزگشایی کرده‌اند تا یک سلول انتخاب شود، و آنگاه عمل خواندن یا نوشتن بر روی سلول قابل اجرا خواهد بود. قبل از آغاز اجرای عمل روی حافظه بعدی، CAS به 1 باز گردانده می‌شود.

## ۷-۴ تشخیص و تصحیح خطا

سطح پیچیدگی یک آرایه حافظه ممکن است هر از چندگاه خطاهایی را در ذخیره و بازیابی اطلاعات دودویی موجب شود. قابلیت اطمینان یک واحد حافظه را با استفاده از کدهای تشخیص و تصحیح خطا می‌توان بهبود بخشید. مرسوم‌ترین روش تشخیص خطا، بیت توازن است (بخش ۸-۳ ملاحظه شود). پس از خواندن کلمه از حافظه، توازن آن چک می‌شود. اگر توازن بیت‌های خوانده شده صحیح باشد کلمه داده پذیرفته خواهد شد. اگر حاصل توازن چک شده معکوس باشد، خطایی اعلام می‌شود ولی قابل اصلاح نیست.

یک کد تصحیح خطا، بیت‌های تست توازن چند بیتی تولید می‌کند که همراه با کلمه داده در حافظه ذخیره می‌گردد. هر بیت تست توازن مربوط به گروهی از بیت‌ها در کلمه داده است. وقتی که کلمه از حافظه بازخوانی می‌شود، بیت‌های توازن نیز از حافظه خوانده شده و با مجموعه جدیدی از بیت تست تولید شده که از داده خوانده شده بدست می‌آید مقایسه می‌گردد. اگر بیت‌های تست صحیح باشند، بیانگر عدم وقوع خطاست. اگر بیت‌های تست با توازن ذخیره شده همخوانی نداشته باشد، یک الگوی منحصر به فردی به نام نشانه (syndrome) تولید می‌کنند که می‌توان از آن برای تشخیص بیت خطا استفاده کرد. هر وقت یک خطا در یک بیت رخ دهد، به این معنی است که مقدار آن در حین خواندن و نوشتن از 1 به 0 و یا از 0 به 1 تغییر یافته است. اگر بیت خطای خاصی شناسایی شد، می‌توان با متمم‌کردن آن بیت خطا را اصلاح کرد.

### کدهمینگ

یکی از رایج‌ترین کدهای تصحیح خطا که در حافظه‌های RAM به کار می‌رود به وسیله R.W.Hamming پیشنهاد گردید. در کد همینگ، k بیت توازن به کلمه داده n بیتی اضافه می‌شود تا یک کد جدید n + k بیتی بدست آید. مکان بیت‌ها از 1 تا n + k شماره‌گذاری می‌شود. مکان‌هایی که توانی از 2 هستند برای بیت‌های توازن رزرو شده‌اند. بیت‌های باقیمانده، بیت‌های داده هستند. کد می‌تواند با کلماتی با هر طول به کار برده می‌شود. قبل از ارائه مشخصات عمومی کد، عمل را با یک داده هشت بیتی تشریح می‌نماییم.

به عنوان مثال کلمه داده 8 بیتی 11000100 را در نظر بگیرید. برای کلمه هشت بیتی چهار بیت توازن را لحاظ می‌کنیم و دوازده بیت حاصل را به صورت زیر مرتب می‌نماییم.

| موقعیت بیت | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $P_1$ | $P_2$ | 1 | $P_4$ | 1 | 0 | 0 | $P_8$ | 0 | 1 | 0 | 0 |

چهار بیت توازن $P_1$، $P_2$، $P_4$ و $P_8$ به ترتیب در مکان‌های 1، 2، 4 و 8 قرار دارند. 8 بیت کلمه داده در بقیه مکان‌ها قرار گرفته‌اند. هر بیت توازن به طریق زیر محاسبه می‌شود.

$$P_1 = \text{XOR of bits } (3, 5, 7, 9, 11) = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 0$$
$$P_2 = \text{XOR of bits } (3, 6, 7, 10, 11) = 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 = 0$$
$$P_4 = \text{XOR of bits } (5, 6, 7, 12) = 1 \oplus 0 \oplus 0 \oplus 0 = 1$$
$$P_8 = \text{XOR of bits } (9, 10, 11, 12) = 0 \oplus 1 \oplus 0 \oplus 0 = 1$$

توجه داشته باشید که تابع XOR یک تابع فرد است. این تابع در ازاء تعداد فردی از 1 در متغیرها برابر 1 و در ازاء تعداد زوجی از 1 برابر 0 است. بنابراین، هر بیت توازن طوری تنظیم می‌شود که تعداد کل 1 ها در متغیرهای تست، از جمله بیت توازن همیشه زوج باشد.

کلمه داده هشت بیتی به همراه چهار بیت توازن به صورت یک کلمه مرکب 12 بیتی در حافظه ذخیره می‌گردد. با جایگزینی چهار بیت P در مکان‌های صحیح یک کلمه مرکب 12 بیتی بدست می‌آوریم.

| موقعیت بیت : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

وقتی که 12 بیت فوق‌الذکر از حافظه خوانده می‌شود، برای وجود خطاهای ممکن جستجو می‌گردد. توازن روی ترکیب بیت‌های فوق و بیت توازن چک شده و چهار بیت چک به طریق زیر ارزیابی می‌شود.

$$C_1 = \text{XOR of bits } (1, 3, 5, 7, 9, 11)$$
$$C_2 = \text{XOR of bits } (2, 3, 6, 7, 10, 11)$$
$$C_4 = \text{XOR of bits } (4, 5, 6, 7, 12)$$
$$C_8 = \text{XOR of bits } (8, 9, 10, 11, 12)$$

یک بیت چک 0 به معنی توازن زوج روی بیت‌های چک شده و یک 1 به معنی توازن فرد است. چون بیت‌ها با توازن زوج ذخیره شده بودند، نتیجه $C = C_8C_4C_2C_1 = 0000$ بیانگر عدم وجود خطاست. با این وجود اگر $C \neq 0$ باشد، آنگاه عدد دودویی 4 بیت تشکیل شده با بیت‌های چک یا تست، محل بیت خطا را مشخص خواهد کرد. مثلاً سه حالت زیر را در نظر بگیرید.

| | موقعیت بیت : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| بدون خطا | | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| خطا در بیت 1 | | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| خطا در بیت 5 | | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

در حالت اول خطایی در کلمه 12 بیتی وجود ندارد. در حالت دوم خطایی در مکان شماره 1 وجود دارد زیرا از 0 به 1 تبدیل شده است. حالت سوم خطایی را در محل بیت 5 نشان می‌دهد که از 1 به 0 تبدیل شده است. با ارزیابی XOR بیت‌های مربوطه، چهار بیت چک را به طریق زیر محاسبه می‌کنیم:

| | $C_8$ | $C_4$ | $C_2$ | $C_1$ |
|---|---|---|---|---|
| بدون خطا | 0 | 0 | 0 | 0 |
| خطا در بیت 1 | 0 | 0 | 0 | 1 |
| خطا در بیت 5 | 0 | 1 | 0 | 1 |

بنابراین برای حالت بدون خطا $C = 0000$؛ با یک خطا در بیت 1، $C = 0001$؛ و با یک خطا در محل بیت 5، $C = 0101$ را داریم. عدد دودویی C، وقتی که برابر 0000 نباشد، محل بیت خطا را مشخص می‌کند. خطا را می‌توان با متمم کردن بیت مذکور تصحیح کرد. توجه کنید که خطا می‌تواند در

جدول ۲-۷. محدوده بیت‌های دادهٔ برای n بیت چک

| محدوده های بیت داده ، $n$ | تعداد بیت‌های چک ، $k$ |
|---|---|
| 2-4 | 3 |
| 5-11 | 4 |
| 12-26 | 5 |
| 27-57 | 6 |
| 58-120 | 7 |

یکی از کلمات داده و یا در یکی از بیت‌های توازن باشد.

کد همینگ را می‌توان برای کلمات داده‌ای با هر طول استفاده کرد. به طور کلی، این کد متشکل از k بیت چک، و n بیت داده بوده و مجموعاً $n + k$ بیت را شامل می‌شود. مقدار نشانه ، C ، متشکل از k بیت و در محدوده $2^k$ از 0 تا $2^k-1$ می‌باشد. یکی از این مقادیر که معمولاً غیرصفر است، برای شناسایی عدم وجود خطا مورد استفاده قرار می‌گیرد، و $2^k-1$ مقدار باقیمانده برای تعیین $n + k$ بیت خطا وارد می‌باشد. هر یک از این $2^k-1$ مقدار را می‌توان برای توصیف منحصر وجود یک بیت خطا به کار برد. بنابراین محدوده $2^k-1$ باید بزرگتر از $n + k$ باشد.

$$2^k - 1 \geq n + k$$

با حل n برحسب k داریم:

$$2^k - 1 - k \geq n$$

این رابطه فرمولی برای ایجاد تعداد بیت‌های داده‌ای که همراه با k بیت چک به کار می‌روند، را نشان می‌دهد. مثلاً وقتی $k = 3$ باشد، تعداد بیت‌های داده قابل استفاده برابر با $n \leq (2^3-1 - 3) = 4$ خواهد بود. برای $k = 4$ داریم $2^4-1 - 4 = 11$ و بنابراین $n \leq 11$ می‌باشد. کلمه داده ممکن است از 11 بیت کمتر باشد ولی باید حداقل 5 بیت داشته باشد، در غیر این صورت تنها 3 بیت چک نیاز خواهد بود. این خود وجود 4 بیت چک را در مثال داده 8 بیتی قبلی توجیه می‌نماید. محدوده n برای مقادیر مختلف k در جدول ۲-۷ لیست شده است.

گروه‌بندی بیت‌ها برای تولید توازن و چک کردن از لیستی از اعداد دودویی بین 0 تا $2^k-1$ معین می‌شود. کم‌ارزش‌ترین بیت در اعداد دودویی 1 ، 3 ، 5 ، 7 و غیره برابر 1 است. دومین بیت باارزش‌تر در اعداد دودویی 2 ، 3 ، 6 ، 7 و غیره نیز برابر 1 است. با مقایسه این اعداد با مکان بیت‌های به کار رفته در تولید و چک بیت‌های توازن در کد همینگ رابطه‌ای بین گروه‌بندی بیت‌ها در یک کد و مکان بیت‌های 1 در رشته شمارش ملاحظه می‌شود. توجه کنید که هر گروه از بیت‌ها با عددی که توانی از 2 ، مانند 1 ، 2 ، 4 ، 8 ، 16 و غیره است آغاز می‌شود. این اعداد مکان بیت‌های توازن نیز هستند.

## تصحیح تک خطا، تشخیص جفت خطا

کد همینگ فقط قادر است یک کد را شناسایی و تصحیح کند. خطاهای چندگانه قابل تشخیص نیستند. با افزودن یک بیت توازن دیگر به کلمه کد شده، کد همینگ را برای تصحیح یک خطا و تشخیص

دو خطا می‌توان به کار برد. اگر این بیت اضافی را لحاظ نماییم، آنگاه کد 12 بیتی قبلی تبدیل به $001110010100P_{13}$ خواهد شد که در آن $P_{13}$ از XOR کردن 12 بیت دیگر ارزیابی می‌شود. این موجب می‌شود تا کد به صورت سیزده بیت 0011100101001 در آید (توازن زوج). وقتی که کد 13 بیت از حافظه خوانده شد، بیت‌های تست یا چک و نیز بیت P در تمام 13 بیت ارزیابی می‌گردند. اگر $P = 0$ باشد، توازن صحیح (توازن زوج) است، اما اگر $P = 1$ باشد، آنگاه توازن در 13 بیت غیرصحیح است (توازن فرد). چهار حالت زیر ممکن است اتفاق بیفتد.

اگر $C = 0$ و $P = 0$ باشد، خطا وجود ندارد.
اگر $C \neq 0$ و $P = 1$ باشد، یک خطا رخ داده که قابل اصلاح است.
اگر $C \neq 0$ و $P = 0$ باشد، یک جفت خطا رخ داده که قابل شناسایی است ولی قابل اصلاح نیست.
اگر $C = 0$ و $P = 1$ باشد، یک خطا در بیت $P_{13}$ رخ داده است.

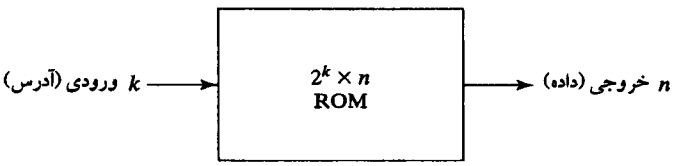
این روال ممکن است بیش از دو خطا را شناسایی کند ولی تصحیح آنها تضمین نمی‌شود.

مدارهای مجتمع از کد همینگ اصلاح شده برای تولید و تست بیت‌های توازن در اصلاح یک خطا و یا شناسایی دو خطا استفاده می‌کنند. کد اصلاح شده از آرایش توازن کاراکتری برخوردار است و در آن تعداد بیت‌های به کار رفته در محاسبه اعمال XOR بالانس شده است. نمونه‌ای از این آی‌سی که از کلمه داده 8 بیت و کلمه تست 5 بیت استفاده می‌نماید نوع 74637 است. مدارهای مجتمع دیگری هم برای کلمات 16 و 32 بیتی وجود دارند. این مدارها می‌توانند در کنار حافظه‌ها برای تصحیح یک خطا و یا تشخیص دو خطا در حین نوشتن و خواندن به کار روند.

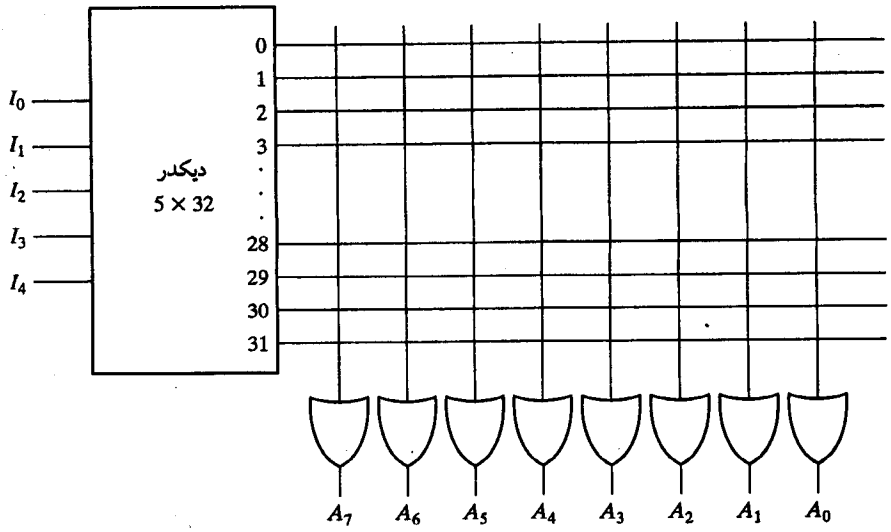
## ۷-۵ حافظه فقط خواندنی ROM

حافظه فقط خواندنی (ROM) اساساً یک وسیله ذخیره‌سازی است که در آن اطلاعات دودویی به طور دائم نگهداری می‌شود. اطلاعات داده باید توسط طراح مشخص شود و سپس برای ایجاد الگوی اتصالات داخلی به واحد حافظه وارد گردد. به محض ایجاد الگو، حتی با قطع و وصل منبع تغذیه، اطلاعات در آن باقی خواهد ماند.

نمودار بلوکی ROM در شکل ۹-۷ دیده می‌شود. این حافظه دارای k ورودی و n خروجی است. ورودی‌ها آدرس حافظه را مهیا می‌کنند و خروجی‌ها بیت‌های داده کلمه ذخیره شده انتخابی به وسیله آدرس را فراهم می‌نمایند. تعداد کلمات در حافظه با توجه به k بیت خط آدرس برابر $2^k$ کلمه است. توجه دارید که ROM فاقد ورودی داده است زیرا عمل نوشتن را انجام نمی‌دهد. تراشه‌های مدار مجتمع




**شکل ۹-۷**. نمودار بلوکی ROM




**شکل ۱۰-۷.** منطق درونی ROM 32×8

ROM دارای یک یا چند ورودی فعال‌ساز هستند و گاهی هم با خروجی سه حالته عرضه می‌گردند تاساخت آرایه‌های بزرگتر ROM را امکان‌پذیر سازند.

به عنوان مثال ROM 8 × 32 را در نظر بگیرید. واحد دارای 32کلمه 8 بیتی است. پنج خط ورودی اعداد دودویی 0 تا 31 را برای آدرس تشکیل می‌دهند. شکل ۱۰-۷ ساختار منطقی درونی ROM را نشان می‌دهد. پنج ورودی به وسیله دیکدر 32 × 5 به 32 خروجی جدا دیکد می‌شوند. هر یک از خروجی‌های دیکدر یک آدرس حافظه را نشان می‌دهند. 32 خروجی دیکدر به هر یک از هشت گیت OR متصل است. در نمودار، قرارداد منطق آرایه‌ای که برای مدارهای پیچیده به کار می‌رود ملاحظه می‌گردد. هر یک OR باید با 32 ورودی در نظر گرفته شود. هر خروجی دیکدر به یکی از ورودی‌های هر گیت OR وصل است. چون هر گیت OR دارای 32 اتصال درونی است. به طور کلی یک ROM $2^k \times n$ دارای دیکدر داخلی $2^k \times k$ و n گیت OR است. هر گیت OR دارای $2^k$ ورودی است که به هر یک از خروجی‌های دیکدر متصل می‌باشد.

تمام 256 تقاطع در شکل ۱۰-۷ قابل برنامه‌ریزی‌اند. یک اتصال قابل برنامه‌ریزی دو خط در عمل به معنی کلیدی است که می‌تواند به هر یک از دو حالت بسته (یعنی خط به هم وصلند) و باز (به معنی دو خط از هم جدا) تغییر وضع دهد. تقاطع قابل برنامه‌ریزی بین دو خط راگاهی نقطه تقاطع هم می‌گویند. برای پیاده‌سازی نقاط تقاطع وسایل فیزیکی متفاوتی وجود دارد. یکی از ساده‌ترین تکنولوژی‌ها از فیوز استفاده می‌کند که معمولاً دو نقطه را به هم وصل می‌نماید، و با اعمال یک پالس ولتاژ شدید به فیوز، اتصال باز یا سوزانده می‌شود.

ذخیره‌سازی درونی یک ROM با جدول درستی که محتوای کلمه را در هر آدرس نشان می‌دهد،

جدول ۳-۷. جدول درستی ROM (ناقص)

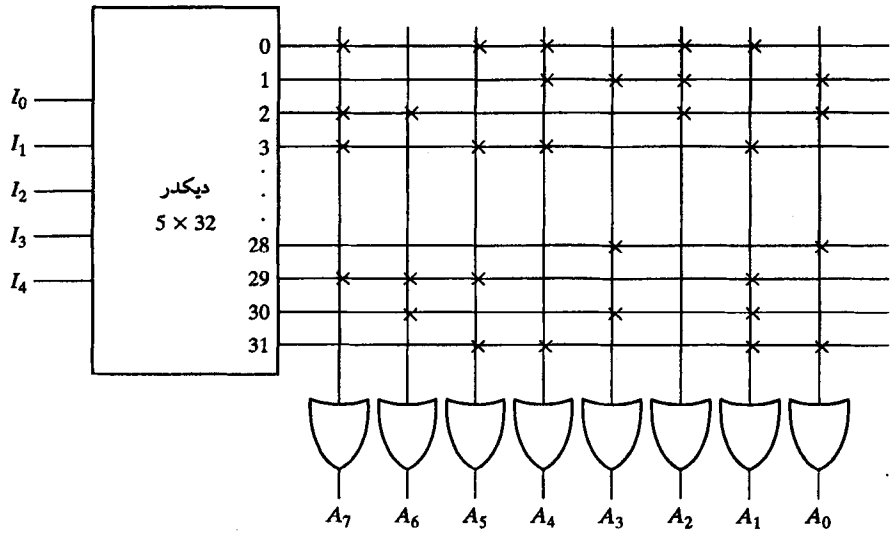
| ورودی‌ها | | | | | خروجی‌ها | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I4 | I3 | I2 | I1 | I0 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| | | ⋮ | | | | | ⋮ | | | | | |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |

صورت می‌گیرد. مثلاً، محتوای یک ROM 8 × 32 ممکن است با یک جدول درستی مشابه جدول (۳-۷) مشخص می‌گردد. در جدول پنج ورودی که در زیر آنها همه 32 آدرس آمده دیده می‌شود. در هر آدرس یک کلمه 8 بیتی ذخیره شده است که در زیر ستون خروجی‌ها لیست شده است. جدول، تنها چهار کلمه اول و چهار کلمه آخر را در ROM نشان می‌دهد. جدول کامل حاوی 32 کلمه خواهد بود.

روش سخت‌افزاری برنامه‌ریزی ROM سوزاندن فیوزها طبق یک جدول درستی است. مثلاً برنامه‌ریزی ROM بر طبق جدول درستی (۳-۷) آرایش شکل ۱۱-۷ را نتیجه می‌دهد. هر 0 موجود در جدول به معنی عدم وجود اتصال و هر 1 مسیری را که به وسیله اتصال به دست می‌آید، نشان می‌دهد. به عنوان مثال، جدول کلمه 8 بیت 10110010 را در آدرس 3 نشان می‌دهد. چهار عدد 0 واقع در کلمه با سوزاندن فیوزهای بین خروجی 3 دیکدر و ورودی‌های گیت‌های OR مربوطه به خروجی‌های $A_6$ ، $A_3$ ، $A_2$ و $A_0$ ، برنامه‌ریزی می‌شود. چهار 1 در کلمه در نمودار با × علامت‌زنی شده‌اند تا اتصالی را در نمودار منطقی بیان نمایند. وقتی ورودی ROM برابر 00011 می‌باشد، همه خروجی‌های دیکدر 0اند به جز خروجی 3که در منطق 1 است. سیگنال معادل با منطق 1 در خروجی 3 دیکدر از میان اتصالات عبور کرده و به خروجی‌های $A_7$ ، $A_5$ ، $A_4$ و $A_1$ می‌رود. چهار خروجی دیگر در 0 باقی می‌مانند. نتیجه این است که کلمه ذخیره شده 10110010 به هشت خروجی داده ارسال می‌گردد.

## پیاده‌سازی مدارهای ترکیبی

در بخش ۸-۴ دیدیم که یک دیکدر، $2^k$ مینترم را برای k متغیر ورودی تولید می‌کند. با نصب گیت‌های OR برای بدست آوردن جمع مینترم‌های توابع بول توانستیم هر مدار ترکیبی را تولید نماییم. ROM اساساً وسیله‌ای است که هر دو بخش دیکدر و گیت‌های OR را در خود دارد. با انتخاب اتصالات برای مینترم‌های تابع، خروجی‌های ROM برای نمایش متغیرهای خروجی تابع بول در مدار ترکیبی، قابل برنامه‌ریزی خواهد بود.




**شکل ۱۱-۷**. برنامه‌ریزی ROM بر طبق جدول ۳-۷

عملکرد درونی ROM را به دو طریق می‌توان تفسیر کرد. اول این که ROM حافظه‌ای است حاول الگوی ثابتی از کلمات ذخیره شده. تفسیر دوم این است که این واحد یک مدار ترکیبی را پیاده‌سازی می‌نماید. از این دیدگاه هر پایانه خروجی به طور جداگانه یک خروجی برای تابع بول تلقی می‌شود که برحسب جمع مینترم‌ها بیان گشته است. مثلاً ROM شکل ۱۱-۷ را می‌توان یک مدار ترکیبی با هشت خروجی تصور کرد که هر یک تابعی از پنج متغیر ورودی است. خروجی $A_7$ را می‌توان به صورت جمع مینترم‌های زیر نوشت:

$$A_7\ (I_4\ ,\ I_3\ ,\ I_2\ ,\ I_1\ ,\ I_0) = \Sigma(0,\ 2,\ 3,\ \ldots\ ,\ 29)$$

اتصالی که در شکل بالا × علامت زده شده است مینترمی را برای مجموع تولید می‌کند. دیگر نقاط تقاطع اتصال ندارند و بنابراین در جمع لحاظ نشده‌اند.

در عمل، وقتی یک مدار ترکیبی به وسیله ROM طراحی می‌شود، لزومی ندارد اتصال درونی گیت‌ها را نشان دهیم. آنچه که یک طراح باید انجام دهد این است که ROM را با شماره‌اش مشخص نماید و جدول درستی ROM را نیز مهیا سازد. جدول درستی همه اطلاعات لازم برای برنامه‌ریزی ROM را دارد. هیچگونه نمودار منطقی درونی برای تکمیل و همراهی جدول درستی مورد نیاز نیست.

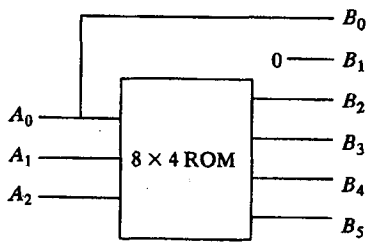
**مثال ۱-۷**

**با استفاده از ROM یک مدار ترکیبی طراحی نمایید. مدار یک عدد 3 بیت را پذیرفته و یک عدد دودویی معادل با مرجع عدد ورودی تولید می‌نماید.**

**جدول ۴-۷**. جدول درستی مدار مثال ۱-۷

| ورودی‌ها | | | خروجی‌ها | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $A_2$ | $A_1$ | $A_0$ | $B_5$ | $B_4$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ | دهدهی |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 9 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 16 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 25 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 36 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 49 |

اولین قدم بدست آوردن جدول درستی مدار ترکیبی است. در بسیاری از موارد بدست آوردن جدول درستی کافی است. در بعضی از حالات هم جدول درستی کوچکتری را برای ROM با توجه به خواص متغیرهای خروجی تعریف می‌کنند. جدول ۴-۷ جدول درستی مدار ترکیبی است. برای لحاظ همه اعداد ممکن دودویی، سه ورودی و شش خروجی لازم است. با توجه به جدول ملاحظه می‌شود که خروجی $B_0$ همواره با ورودی $A_0$ برابر است، بنابراین نیازی به تولید $B_0$ توسط ROM نیست. به علاوه خروجی $B_1$ همواره 0 است، بنابراین خروجی یک ثابت شناخته شده است. در نتیجه لازم است فقط چهار خروجی را با ROM تولید نماییم، و دو خروجی دیگر به راحتی قابل حصول می‌باشند. به این ترتیب حداقل سایزی که ROM به آن نیاز دارد سه ورودی و چهار خروجی است که سه ورودی هشت کلمه را مشخص می‌کنند و بنابراین ROM باید دارای سایز 4 × 8 باشد. پیاده‌سازی ROM این تابع در شکل ۱۲-۷ مشاهده می‌شود که در آن سه ورودی مدار هشت کلمه چهار بیتی را معین می‌نمایند. جدول درستی شکل ۱۲-۷ (ب) حاوی اطلاعات لازم برای برنامه‌ریزی ROM است. نمودار بلوکی شکل ۱۲-۷ (الف) اتصالات مورد نیاز را برای مدار ترکیبی نشان می‌دهد.




(الف) نمودار بلوکی

| $A_2$ | $A_1$ | $A_0$ | $B_5$ | $B_4$ | $B_3$ | $B_2$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 |

(ب) جدول درستی ROM

**شکل ۱۲-۷**. پیاده سازی ROM

## انواع ROM

مسیرهای لازم در یک ROM به چهار روش مختلف برنامه‌ریزی می‌شوند. اولین روش که برنامه‌ریزی ماسک نامیده می‌شود به وسیله سازنده در آخرین مرحله ساخت قطعه صورت می‌گیرد. در

این روش، ساخت ROM مستلزم پر شدن جدول درستی مربوطه به وسیله سفارش دهنده است. این جدول ممکن است روی فرم مخصوصی که توسط سازنده ارائه می‌شود، و یا با قالب خاصی، روی خروجی‌های کامپیوتر ایجاد گردد. بر طبق جدول مشتری، سازنده ماسک مربوطه را برای تولید 1ها و 0ها برای مسیرها می‌سازد. این روش بسیار پرهزینه است زیرا سازنده مبلغ خاصی را برای ماسک کردن ROM از مشتری می‌گیرد. به همین دلیل، برنامه‌ریزی ماسک هنگامی اقتصادی است که تعداد زیادی از یک نوع ROM سفارش داده شود.

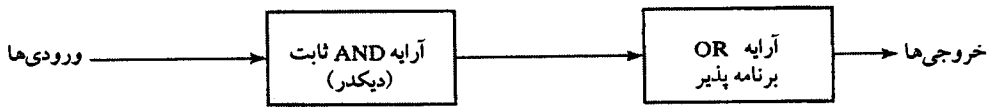
برای تعداد کم، نوع دوم آن به نام حافظه فقط خواندنی برنامه‌پذیر (PROM) اقتصادی‌تر است. هنگام سفارش، همه فیوزهای PROM دست نخورده و همه بیت‌های کلمه در منطق 1 هستند. سوزاندن فیوزها در PROM با استفاده از یک پالس ولتاژ قوی و از طریق پایه خاصی صورت می‌گیرد. فیوز سوخته منطق 0 و دست نخورده منطق 1 را ایجاد می‌نمایند. این امکانات به کاربر اجازه می‌دهد تا PROM را در آزمایشگاه برنامه‌ریزی کرده و به رابطه بین آدرس‌های ورودی و کلمات ذخیره شده دست یابد. دستگاه خاصی به نام برنامه‌نویس PROM برای انجام این کار به صورت تجاری در اختیار قرار دارد. در هر حال، همه رویه‌ها برای برنامه‌ریزی ROMها روال‌های سخت‌افزاری‌اند حتی اگر برنامه‌نویسی کلمات صورت گیرد.

روال برنامه‌نویسی یا برنامه‌ریزی سخت‌افزاری ROMها یا PROMها برگشت‌ناپذیرند و به محض انجام آن، الگویی ثابت و دائمی در آن ایجاد می‌شود که تغییرناپذیر است. به محض ایجاد الگوی بیتی، کل قطعه هنگام تغییر الگو دور ریخته می‌شود. سومین نوع ROM، به نام *PROM* پاک شونده یا EPROM است. می‌توان EPROM را دوباره به حالت اولیه‌اش در آورد، حتی اگر قبلاً برنامه‌نویسی شده باشد. وقتی EPROM را برای مدت معینی تحت تابش اشعه ماوراء بنفش قرار دهیم اشعه موج کوتاه مذکور گیت‌های شناور درونی که نقش اتصالات برنامه‌ریزی شده را دارند، تخلیه می‌کند. پس از پاک شدن، EPROM به حالت اولیه خود بازگشته و با مقادیر جدیدی قابل برنامه‌ریزی است.

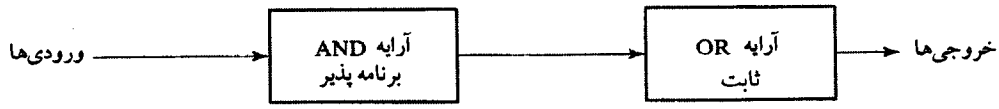
نوع چهارم ROM، *PROM* پاک شونده الکتریکی (EPROM یا EEPROM) است. این حافظه شبیه EPROM است با این تفاوت که اتصالات برنامه‌ریزی شده قبلی را به صورت الکتریکی در عوض اشعه ماوراء بنفش می‌توان پاک کرد. مزیت این کار این است که حافظه بدون نیاز به خارج کردن آن از سوکت پاک می‌شود.

### PLD های ترکیبی

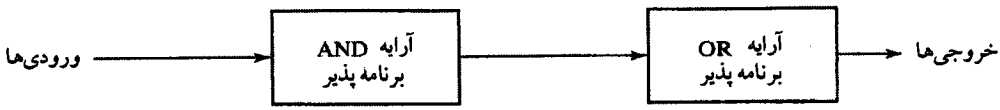
PROM یک وسیله منطقی برنامه‌پذیر ترکیبی در یک مدار مجتمع است که گیت‌های برنامه‌پذیر آن به دو بخش آرایه AND و آرایه OR تفکیک شده است تا جمع حاصلضرب‌ها با AND-OR پیاده‌سازی شود. سه نوع PLD ترکیبی وجود دارند و اختلاف آنها در ایجاد اتصالات برنامه‌پذیر در آرایه AND-OR است. شکل ۱۳-۷ آرایش سه PLD را نشان می‌دهد. حافظه فقط خواندنی برنامه‌پذیر (PROM) دارای آرایه AND ثابت به عنوان دیکدر است ولی آرایه OR آن برنامه‌پذیر است. گیت‌های




(الف) حافظه فقط خواندنی برنامه پذیر (PROM)




(ب) منطق آرایه‌ای برنامه پذیر (PAL)




(پ) آرایه منطقی برنامه پذیر (PLA)

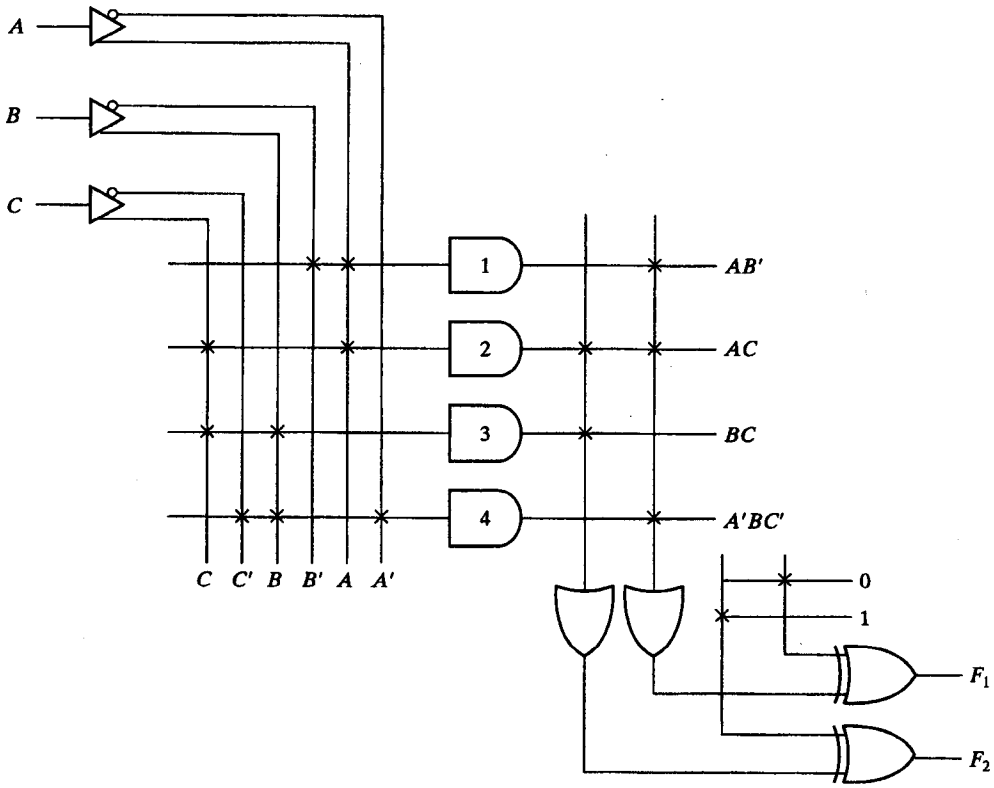
شکل ۱۳-۷. آرایش پایه سه نوع PDL

OR برنامه‌پذیر توابع بول را به صورت جمع حاصلضرب‌ها پیاده‌سازی می‌کنند. منطق آرایه برنامه‌پذیر (PAL) دارای آرایه AND برنامه‌پذیر و آرایه OR ثابت است. گیت‌های AND در تهیه جملات ضرب برای تابع بول برنامه‌ریزی می‌شوند که به صورت منطقی در هر گیت OR جمع شده‌اند. با انعطاف‌ترین PLD، آرایه منطقی برنامه‌پذیر (PLA) است که در آن هر دو آرایه AND و OR برنامه‌پذیرند. جملات ضرب در آرایه AND قابل استفاده مشترک با هر گیت OR برای پیاده‌سازی جمع حاصلضرب موردنظر است. نام‌های PAL و PLA از دو سازنده مختلف در حین ساخت وسایل منطقی برنامه‌پذیر متفاوت اعلان شد. پیاده‌سازی مدارهای ترکیبی با PROM در این بخش نشان داده شد. طراحی این مدارها با PLA و PAL در دو بخش بعد مطرح شده است.

## ۷-۶ آرایه منطقی برنامه‌پذیر PLA

آرایه منطقی برنامه‌پذیر (PLA) از نظر مفهوم شبیه PROM است. به جز این که PLA دیکد کامل متغیرها را فراهم نکرده و بنابراین همه مینترم‌ها را تولید نمی‌کند. در این مدار دیکدر با آرایه‌ای از گیت‌های AND جایگزین شده و می‌تواند برای تولید هر جمله ضرب متغیرهای ورودی برنامه‌ریزی شود. سپس جملات ضرب به گیت‌های OR وصل می‌شود تا جمع حاصلضرب‌ها برای توابع بول موردنظر تهیه شود.

منطق درونی یک PLA با سه ورودی و دو خروجی در شکل ۱۴-۷ دیده می‌شود. از لحاظ تجاری این قطعه به علت کوچکی مقرون به صرفه نیست، ولی در اینجا به منظور نمایش آرایش منطقی PLA




**شکل ۱۴-۷** PAL با سه ورودی، چهار جمله ضرب و دو خروجی

نمونه مطرح شده است. نمودار، سمبل‌های گرافیکی آرایه‌ای را برای مدار مجتمع پیچیده به کار می‌برد. هر ورودی وارد یک بافر و یک وارونگر می‌شود که در نمودار هم با سمبلی مرکب نشان داده شده است. این سمبل دارای هر دو خروجی صحیح و متمم شده است. هر ورودی و متمم آن به ورودی‌های هر گیت AND وصل است. این اتصال با تقاطع خطوط عمودی و افقی انجام گردیده است. خروجی‌های گیت‌های AND به ورودی‌های هر گیت وصلند. خروجی گیت OR به یک گیت XOR رفته است. گیت‌های اخیر جهت دریافت 1 یا 0 برنامه‌ریزی شده‌اند. وقتی ورودی XOR به 1 وصل باشد خروجی معکوس می‌شود ( چون $x \oplus 1 = x'$ ) هنگامی که ورودی XOR به 0 وصل باشد خروجی تغییر نمی‌کند، (چون $x \oplus 0 = x$ ) . توابع بول خاصی که در شکل ۱۴-۷ برای PLA پیاده‌سازی شده است عبارتند از:

$$F_1 = AB' + AC + A'BC'$$
$$F_2 = (AC + BC)'$$

جملات ضرب تولید شده در کنار خروجی هر گیت AND در نمودار نوشته شده است. جمله ضرب از

**جدول ۵-۷**. جدول برنامه ریزی PAL

| | جمله ضرب | ورودی‌ها | | | خروجی‌ها (T) | (C) |
|---|---|---|---|---|---|---|
| | | A | B | C | $F_1$ | $F_2$ |
| AB′ | 1 | 1 | 0 | – | 1 | – |
| AC | 2 | 1 | – | 1 | 1 | 1 |
| BC | 3 | – | 1 | 1 | – | 1 |
| A′BC′ | 4 | 0 | 1 | 0 | 1 | – |

روی نقاط تقاطع متصلی که با × علامت خورده معین می‌شود. خروجی یک گیت OR جمع منطقی جملات ضرب انتخابی را تهیه می‌کند. بسته به اتصال یکی از ورودی‌های گیت XOR ، فرم متمم یا غیرمتمم خروجی حاصل می‌گردد.

نقشه فیوز PLA به شکل جدول می‌تواند نشان داده شود. جدول برنامه‌ریزی مربوط به PLA شکل ۷-۱۴ در جدول ۵-۷ ملاحظه می‌گردد. جدول برنامه‌ریزی از سه بخش تشکیل شده است. اولین بخش جملات ضرب را به صورت عدد لیست کرده است. بخش دوم مسیرهای لازم بین ورودی‌ها و گیت‌های AND را مشخص می‌نماید. سومین بخش مسیرهای بین گیت‌های AND و OR را معین می‌کند. برای هر متغیر خروجی، از T (به معنی صحیح یا متمم نشده) و یا C (به معنی متمم) برای برنامه‌ریزی گیت XOR استفاده می‌نماییم. جملات ضرب لیست شده در سمت چپ بخشی از جدول نیستند، بلکه فقط به منظور داشتن مرجع لحاظ شده‌اند. برای هر جمله ضرب ورودی‌ها با 1 ، 0 یا – (خط تیره) علامت‌زنی شده‌اند. اگر متغیری در جمله ضرب متمم نشده ظاهر شود متغیر ورودی مربوطه با 1 علامت‌گذاری می‌شود. اگر متمم باشد آن را با 0 علامت می‌زنیم. اگر متغیر در جمله ضرب وجود نداشته باشد با علامت خط تیره علامت زده خواهد شد. مسیرهای بین ورودی‌ها و گیت‌های AND در جدول برنامه‌ریزی زیرستون "ورودی‌ها" ذکر شده‌اند. هر 1 از ستون ورودی اتصال را از متغیر ورودی به گیت AND نشان می‌دهد. هر 0 در ستون ورودی، اتصالی را از متمم متغیر به ورودی گیت AND نشان می‌دهد. خط تیره به معنی فیوز سوخته‌ای در هر دو ورودی متغیر و متمم آن است. ضمناً هر پایانه باز، به معنی منطق 1 فرض می‌شود.

مسیرهای بین گیت‌های AND و OR زیر ستون "خروجی‌ها" ذکر شده‌اند. متغیرهای خروجی برای جملات ضرب موجود در تابع با 1 علامت زنی می‌شوند. هر جمله ضربی که در ستون خروجی‌اش 1 دارد به مسیری از خروجی گیت AND به ورودی گیت OR احتیاج دارد. مکان‌هایی که با خط تیره علامت زده شده فیوزهای سوخته شونده را مشخص می‌کنند. فرض بر این است که پایانه‌های باز در ورودی گیت OR ، همچون 0 عمل کنند. بالاخره یک خروجی غیرمتمم، T ، لازم می‌دارد که دیگر ورودی‌های گیت XOR مربوطه به 0 وصل شوند، ولی خروجی متمم، C ، اتصال به 1 را بیانگر است.

سایز یک PLA با تعداد ورودی‌ها، جملات ضرب و تعداد خروجی‌ها معین می‌گردد. نوعی مدار

مجتمع PLA ممکن است 16 ورودی، 4 جمله ضرب و 8 خروجی داشته باشد. برای n ورودی، k جمله ضرب و m خروجی، منطق درونی PLA متشکل از n بافر- وارونگر، k گیت AND ، $k \times m$ اتصال بین آرایه AND و OR و m اتصال مربوط به گیت‌های XOR می‌باشد.

هنگام طراحی سیستم‌های دیجیتال با PLA ، نیازی به نمایش اتصالات درونی آن، همچون شکل ۷-۱۴ وجود ندارد. آنچه لازم است، یک جدول برنامه‌ریزی است که PLA به وسیله آن برنامه‌ریزی می‌گردد. همچون ROM ، PLA هم ممکن است به صورت ماسک برنامه‌پذیر، یا برنامه‌پذیر موردی باشد. با برنامه‌ریزی ماسک، مشتری جدول برنامه PLA خود را به سازنده ارائه می‌کند. این جدول به وسیله سازنده استفاده شده و PLA سفارشی که منطق درونی‌اش طبق سفارش مشتری است، تولید می‌گردد. نوع دوم PLA موجود، آرایه منطقی برنامه‌پذیر موردی (FPLA) نامیده می‌شود. FPLA می‌تواند با یک دستگاه برنامه‌نویس سخت‌افزاری تجاری به وسیله کاربر برنامه‌نویس گردد.

هنگام پیاده‌سازی یک مدار ترکیبی با PLA ، باید بررسی‌های دقیقی به عمل آید تا جملات ضرب جدا از هم کاهش یابد، زیرا PLA دارای تعداد گیت‌های محدودی است. این کار با حداقل کردن تعداد جملات در هر تابع بول امکان‌پذیر است. تعداد لیترال در یک جمله اهمیت ندارد زیرا همه متغیرهای ورودی به هر ترتیب در دسترسند. هر دو نوع تابع خروجی، یعنی متمم یا غیرمتمم باید مشخص گردد تا ببینیم کدام یک با تعداد کمتری جمله ضرب بیان می‌شود و کدام جمله نیز در دیگر توابع مشترک است.

**مثال ۲-۷**

**دو تابع بول زیر را با PLA پیاده‌سازی کنید.**

$$F_1(A, B, C) = \Sigma(0, 1, 2, 4)$$
$$F_2(A, B, C) = \Sigma(0, 5, 6, 7)$$

**دو تابع در نقشه ۱۵-۷ ساده شده‌اند. هر دو فرم متمم و غیرمتمم تابع برحسب جمع حاصلضرب‌ها ساده شده‌اند. ترکیبی که حداقل جملات ضرب را می‌دهد عبارت است:**

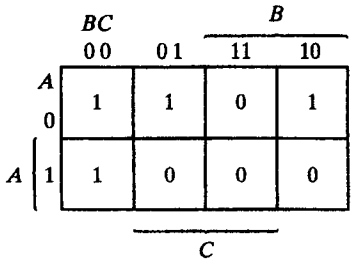
$$F_1 = (AB + AC + BC)'$$

**و**

$$F_2 = AB + AC + A'B'C'$$

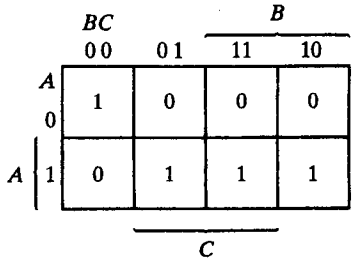
**در این دو تابع چهار جمله ضرب مستقل وجود دارد که عبارتند از AB ، AC ، BC و $A'B'C'$. جدول برنامه‌ریزی PLA برای این ترکیب در شکل نشان داده شده است. توجه کنید که خروجی $F_1$ خروجی غیرمتمم است هر چند که روی حرف متمم (C) نوشته شده است. دلیل این است که $F_1$ با یک مدار AND-OR تولید شده و در خروجی گیت OR در دسترس است. گیت XOR ، تابع را متمم می‌کند تا خروجی صحیح $F_1$ بدست آید.**

❑




$F_1 = A'B' + A'C' + B'C'$

$F_1 = (AB + AC + BC)'$




$F_2 = AB + AC + A'B'C'$

$F_2 = (A'C + A'B + AB'C')'$

**جدول برنامه ریزی PLA**

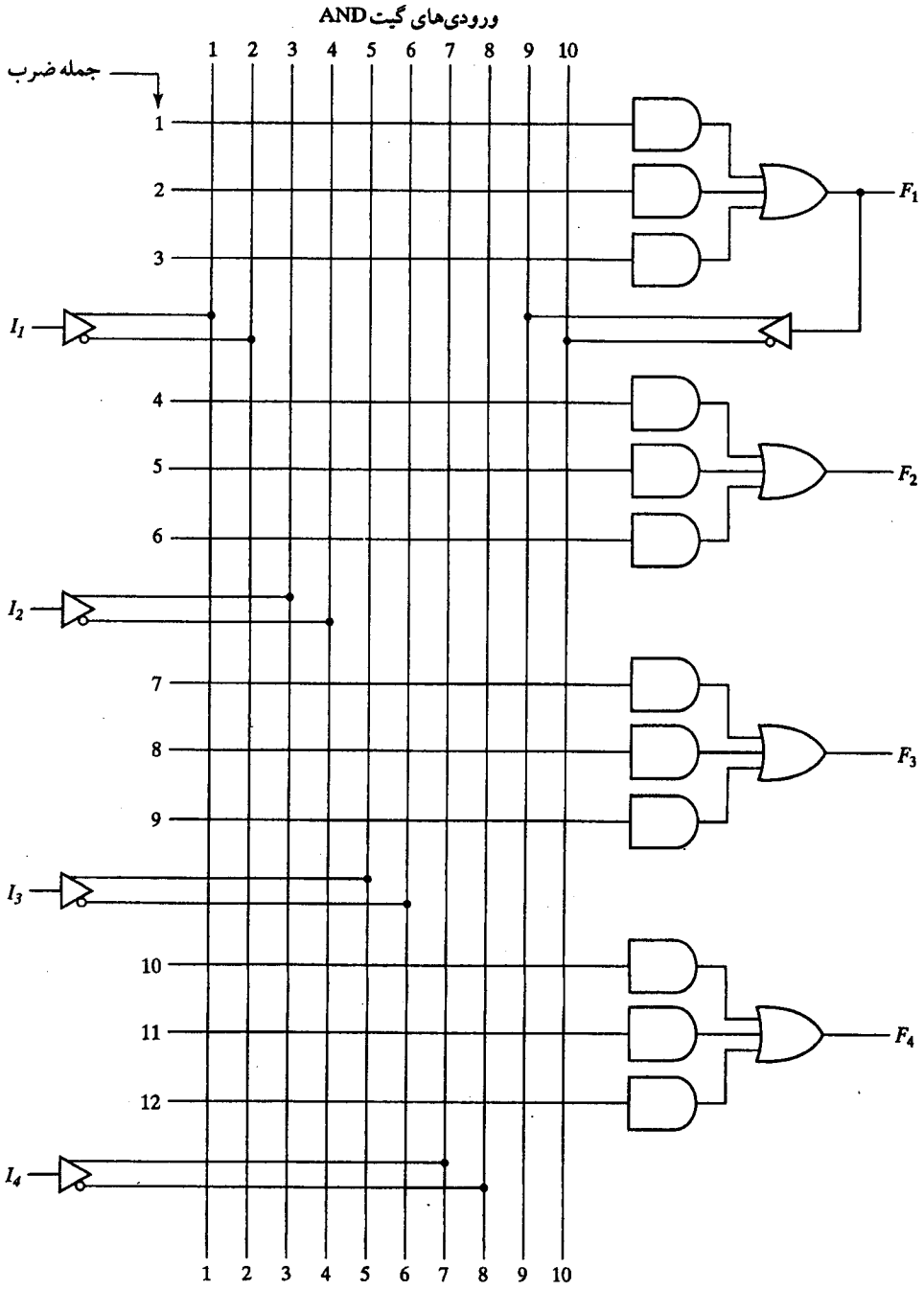
| | جمله ضرب | ورودی‌ها A | B | C | خروجی‌ها (C) $F_1$ | (T) $F_2$ |
|---|---|---|---|---|---|---|
| $AB$ | 1 | 1 | 1 | – | 1 | 1 |
| $AC$ | 2 | 1 | – | 1 | 1 | 1 |
| $BC$ | 3 | – | 1 | 1 | 1 | – |
| $A'B'C'$ | 4 | 0 | 0 | 0 | – | 1 |

شکل ۱۵-۷. حل مثال ۲-۷

مدار ترکیبی به کار رفته در مثال ۲-۷ برای پیاده‌سازی با یک PLA بسیار ساده است. این مثال صرفاً به خاطر تشریح اهداف ارائه شد. PLA های موجود معمولاً تعداد زیادی ورودی و جملات ضرب دارند. ساده‌سازی توابع با تعداد زیادی متغیر باید به کمک روال‌های کامپیوتری انجام شود. برنامه‌های طراحی CAD در این زمینه هر تابع و متمم آن را از نظر تعداد جملات ساده می‌کند. آنگاه برنامه همه توابع را به فرم معمول یا متمم با توجه به حداقل جملات ضرب انتخاب می‌کند. سپس جدول برنامه‌ریزی PLA تولید می‌گردد و نقشه فیوزها فراهم می‌شود. نقشه فیوزها به یک برنامه‌نویس که از یک روال سخت‌افزاری در سوزاندن استفاده می‌کند اعمال می‌گردد تا مدار موردنظر بدست آید.

## ۷-۷ منطق آرایه‌ای برنامه‌پذیر PAL

منطق آرایه‌ای برنامه‌پذیر (PAL) وسیله‌ای است با آرایه OR ثابت و آرایه AND برنامه‌پذیر. چون تنها گیت‌های AND برنامه‌پذیرند، PAL برای برنامه‌ریزی ساده‌تر است ولی به اندازه PLA انعطاف‌پذیر نیست. شکل ۱۶-۷ آرایش منطقی یک PAL نمونه را نشان می‌دهد. این مدار چهار ورودی و چهار خروجی دارد. هر ورودی به یک گیت بافر- وارونگر وصل است و هر خروجی با گیت OR ثابت تولید می‌گردد. در مدار چهار بخش موجود است که هر یک از آرایه AND-OR سه وجهی




**شکل ۱۶-۷** . PAL با چهار ورودی، چهار خروجی AND-OR سه ردیفی

تشکیل شده است. این جمله به این دلیل بکار رفته است تا مشخص شود که سه گیت AND برنامه پذیر در هر بخش و نیز یک گیت OR ثابت وجود دارد. هر گیت AND دارای 10 اتصال ورودی برنامه‌پذیر است. این ویژگی با 10 خط عمودی که هر خط افقی را قطع می‌کند نشان داده شده است. خطوط افقی نمایشگر آرایش چند ورودی گیت AND می‌باشند. یکی از خروجی‌ها به گیت بافر- وارونگر وصل و سپس به دو ورودی گیت AND پسخورد شده است.

PAL های تجاری حامل گیت‌های بیشتری از آنچه در شکل ۱۶-۷ ملاحظه شد، هستند. بعضی از آنها تا هشت ورودی، هشت خروجی، هشت بخش هر یک با سه آرایه AND-OR سه وجهی دارند. پایانه‌های خروجی گاهی با بافرهای سه حالته و وارونگرها راه‌اندازی می‌شوند.

هنگام طراحی با PAL ، توابع بول، تا حدی که در هر بخش قابل پیاده‌سازی شدن باشند، باید ساده شوند. برخلاف PLA ، یک جمله ضرب نمی‌تواند میان دو یا چند گیت OR به اشتراک باشد. بنابراین، هر تابع بدون توجه به وجود اشتراک در جملات ضرب ساده می‌گردد. تعداد جملات ضرب در هر بخش ثابت است، و اگر تعداد جملات در تابع زیاد باشد، ممکن است برای پیاده‌سازی یک تابع بول از دو بخش استفاده شود.

به عنوان مثال از کاربری یک PAL در طراحی یک مدار ترکیبی، توابع بولی زیر را که برحسب مجموع مینترم‌ها داده شده است ملاحظه نمایید.

$$w(A, B, C, D) = \sum(2, 12, 13)$$
$$x(A, B, C, D) = \sum(7, 8, 9, 10, 11, 12, 13, 14, 15)$$
$$y(A, B, C, D) = \sum(0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 15)$$
$$z(A, B, C, D) = \sum(1, 2, 8, 12, 13)$$

با ساده کردن چهار تابع فوق به حداقل تعداد جملات، توابع بولی زیر نتیجه می‌شود

$$w = ABC' + A'B'CD'$$
$$x = A + BCD$$
$$y = A'B + CD + B'D'$$
$$z = ABC' + A'B'CD' + AC'D' + A'B'C'D$$
$$= w + AC'D' + A'B'C'D$$

توجه کنید که تابع z دارای چهار جمله ضرب است. جمع منطقی دوتای آن برابر w است. با به کارگیری w می‌توان تعداد جملات در z را از چهار به سه رساند.

جدول برنامه‌ریزی PAL مشابه PLA است با این تفاوت که فقط ورودی گیت‌های AND باید برنامه‌ریزی شوند. جدول ۶-۷ جدول برنامه‌ریزی PAL را برای چهار تابع بول نشان می‌دهد. جدول به چهار بخش با سه جمله ضرب در هر کدام تقسیم شده است تا با شکل ۱۶-۷ همخوانی داشته باشد. دو بخش اول تنها دو جمله ضرب لازم دارند تا تابع بول پیاده‌سازی شود. بخش آخر برای خروجی z چهار جمله ضرب لازم دارد. با استفاده از خروجی w ، می‌توانیم جملات را به سه برسانیم.

جدول ۶-۷. جدول برنامه ریزی PAL

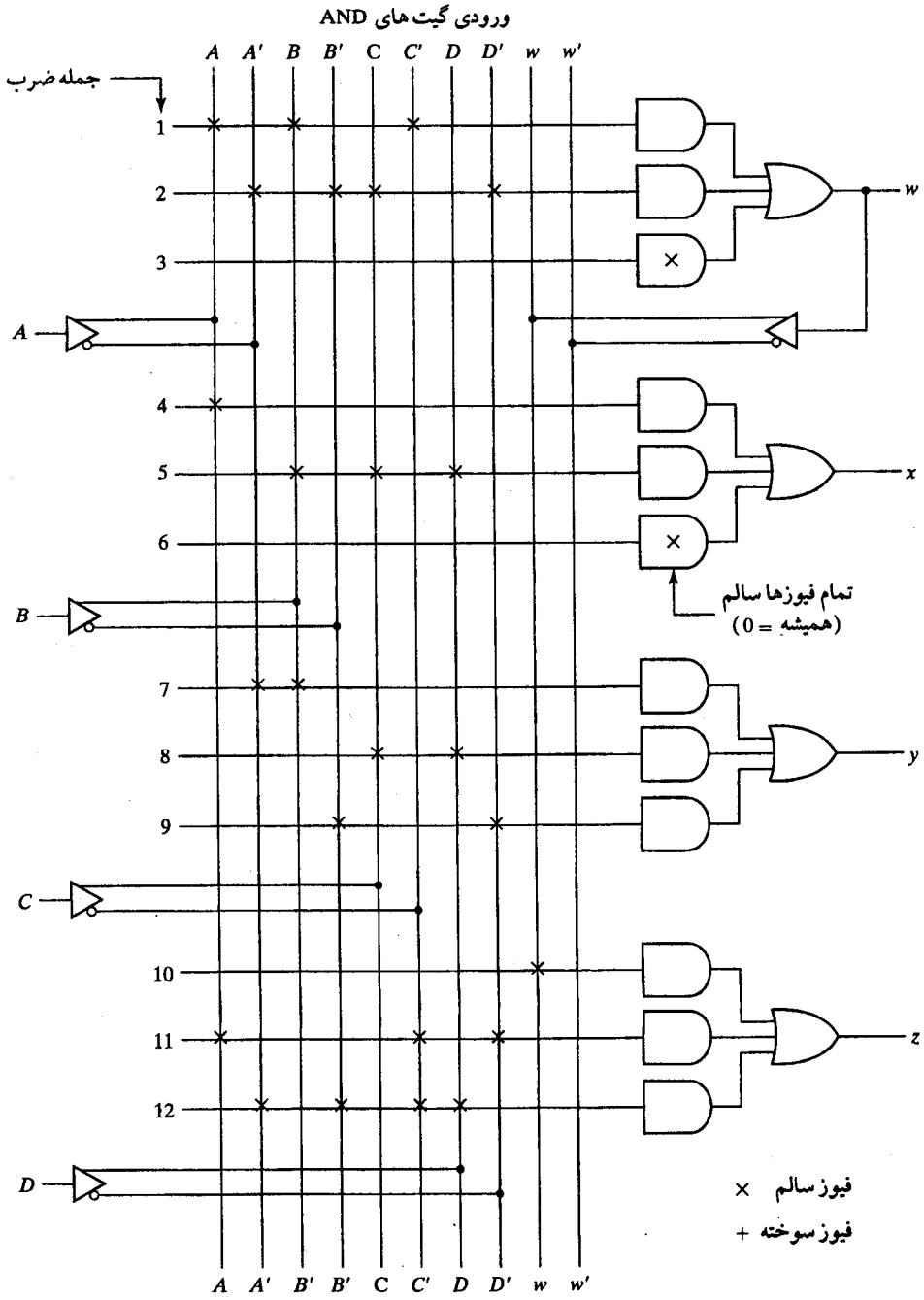
| جمله حاصلضرب | ورودی‌های AND | | | | | خروجی‌ها |
|---|---|---|---|---|---|---|
| | A | B | C | D | W | |
| 1 | 1 | 1 | 0 | – | – | $w = ABC'$ |
| 2 | 0 | 0 | 1 | 0 | – | $+ A'B'CD'$ |
| 3 | – | – | – | – | – | |
| 4 | 1 | – | – | – | – | $x = A$ |
| 5 | – | 1 | 1 | 1 | – | $+ BCD$ |
| 6 | – | – | – | – | – | |
| 7 | 0 | 1 | – | – | – | $y = A'B$ |
| 8 | – | – | 1 | 1 | – | $+ CD$ |
| 9 | – | 0 | – | 0 | – | $+ B'D'$ |
| 10 | – | – | – | – | 1 | $z = w$ |
| 11 | 1 | – | 0 | 0 | – | $+ AC'D'$ |
| 12 | 0 | 0 | 0 | 1 | – | $+ A'B'C'D$ |

نقشه فیوزها برای PAL طبق جدول برنامه‌ریزی در شکل ۷-۱۷ ملاحظه می‌شود. برای هر 1 یا 0 تقاطع را روی نمودار با فیوز سالم علامت می‌زنیم. برای هر خط تیره، نمودار با فیوز سوخته برای هر دو ورودی صحیح (غیرمتمم) و متمم علامت‌زنی می‌شود. اگر گیت AND به کار نرود، همه فیوزهای ورودی سالم رها می‌شوند. چون ورودی مربوطه هر دو فرم متمم و غیرمتمم هر متغیر ورودی را دریافت می‌کند، داریم $AA' = 0$ و خروجی گیت AND همیشه 0 است. طراحی PALها مانند همه PLDها با استفاده از تکنیک‌های کامپیوتری ساده شده است. سوزاندن فیوزهای داخلی یک روال سخت‌افزاری است که به کمک تجهیزات خاص انجام می‌گردد.

## ۷-۸ وسایل برنامه‌پذیر ترتیبی

سیستم‌های دیجیتال با استفاده از فلیپ فلاپ‌ها و گیت‌ها طراحی می‌شوند. چون PLD ترکیبی فقط از گیت‌های ساخته می‌شود، لازم است به هنگام استفاده از آنها در مدارهای ترتیبی، از فلیپ فلاپ‌های بیرونی استفاده شود. وسایل برنامه‌پذیر ترتیبی هر دو نوع مدارهای گیتی و فلیپ فلاپ را دارا هستند. به این ترتیب، مدار را برای اجرای انواع توابع ترتیبی می‌توان برنامه‌ریزی کرد. وسایل برنامه‌پذیر تجاری متعددی در دسترسند و هر وسیله هم تنوع خاص سازنده را دارد. منطق درونی این وسایل برای نمایش بسیار پیچیده است. بنابراین، سه نوع اصلی را بدون تشریح ساختار آن توصیف می‌کنیم.

۱- وسیله منطقی برنامه‌پذیر ترتیبی (SPLD) یا ساده
۲- وسایل منطقی برنامه‌پذیر پیچیده (CPLD)
۳- آرایه گیتی برنامه‌پذیر موردی (FPGA)




**شکل ۱۷-۷**. نقشه فیوز برای PAL طبق جدول (۶-۷)