فصل پنجم

منطق ترتیبی همزمان

در یک مدار ترتیبی، مقدار خروجی در هر لحظه از زمان نه تنها به مقادیر ورودی‌ها در همان لحظه بلکه به مقدار قبلی خروجی/خروجی‌ها (به نام حالت مدار) نیز بستگی دارد. بنابراین در مدارات ترتیبی، به نحوی از عناصر «حافظه» برای «به خاطرسپاریِ» گذشته‌ی خروجی‌ها استفاده می‌شود. این عناصر حافظه، یک مسیر فیدبکی (یا پسخورد) از خروجی به ورودی فراهم می‌کنند. مقادیر دودویی ذخیره شده در این عناصر حافظه، «حالت مدار» نامیده می‌شود. حالت مدار نیز با گذشت زمان تغییر کرده و نحوه‌ی تغییر آن به ورودی‌های مدار و حالت قبلیِ مدار بستگی دارد.

دیاگرام بلوکی یک مدار ترتیبی (Sequential)

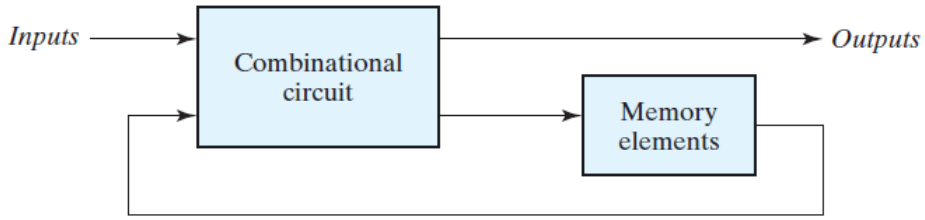
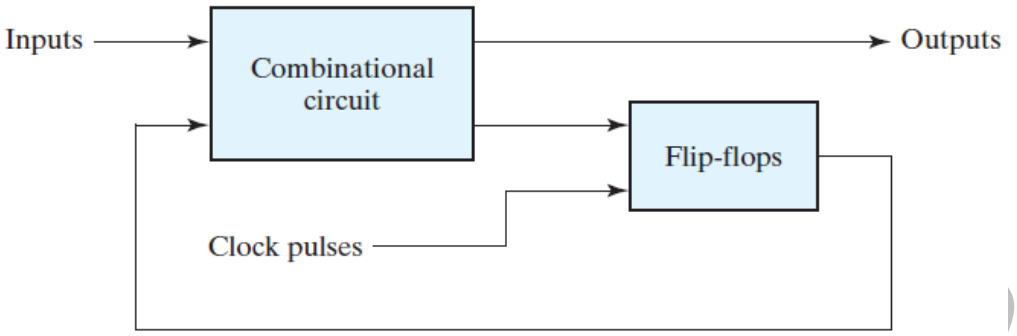


FIGURE 5.1
Block diagram of sequential circuit

دو نوع مدار ترتیبی: سنکرون یا همزمان، آسنکرون یا غیرهمزمان

در اولی یک سیگنال زمان‌بندی به نام ساعت یا «کلاک» وجود دارد که تغییرات خروجی بر حسب ورودی و حالت مدار با اسن سیگنال هماهنگ و تنظیم می‌شود. در دومی، هیچ سیگنال زمان‌بندی و هماهنگ‌سازی وجود ندارد بلکه ترتیب تغییر ورودی‌ها (در هر لحظه‌ی دلخواه از زمان) تعیین کننده‌ی رفتار مدار خواهد بود. در عمل، عمدتاً از نوع سنکرون استفاده می‌شود زیرا مداراتی همیشه پایدار هستند اما مدارات آسنکرون، با مشکل جدی ناپایداری مواجه هستند. در این فصل، تنها به مدارات ترتیبی سنکرون پرداخته می‌شود.

عناصر پایه‌ایِ ذخیره‌ساز اطلاعات (بیتها) در مدارات ترتیبی سنکرون، فلیپ فلاپ (Flip Flop) نامیده می‌شوند. هر فلیپ فلاپ قادر به ذخیره‌ی یک بیت (0 یا 1) است. دیاگرام بلوکی یک مدار ترتیبی سنکرون کلاک‌دار در شکل زیر نشان داده شده است.



(a) Block diagram

(b) Timing diagram of clock pulses

FIGURE 5.2
Synchronous clocked sequential circuit

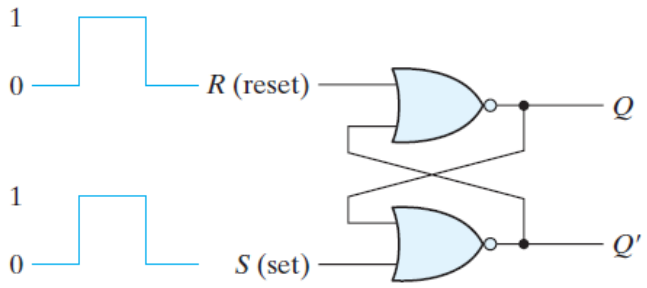
منظور از «خروجی‌های مدار» در شکل اخیر، خروجی‌های یک مدار ترکیبی (متعلق به مدار کلی)، خروجی‌های فلیپ‌فلاپ‌ها، و یا هر دو نوع خروجی است.

فلیپ‌فلاپ‌ها حساس به لبه (Edge sensitive) هستند؛ در مقابل، عناصر حافظه‌ای که حساس به سطحِ سیگنال (Level sensitive) هستند، لچ (Latch) نامیده می‌شوند. فلیپ‌فلاپ‌ها در مدارات ترتیبی سنکرون و لچ‌ها در مدارات ترتیبی آسنکرون استفاده می‌شوند. فلیپ‌فلاپ‌ها بر پایه‌ی لچ‌ها ساخته می‌شوند؛ لذا در این جا ابتدا کار را با معرفی چند نمونه لچ شروع می‌کنیم.

لچ SR

به دو طریق می‌توان لچ SR ساخت: با استفاده از گیت NAND و با استفاده از گیت NOR.

با استفاده از گیت NOR:



(a) Logic diagram

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

(b) Function table

FIGURE 5.3
SR latch with NOR gates

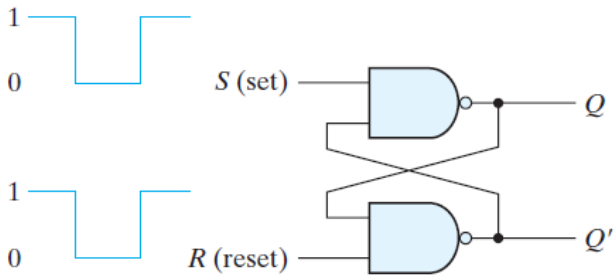
این لچ دو ورودی S (نشاندن[1]) و R (بازنشانی)[2] دارد.

حالت SR=00 موجب حفظ مقادیر قبلی می‌شود (و لچ در حالت معمولی در این وضعیت نگهداری می‌شود؛) لذا مقدار خروجی‌های Q و Q' بستگی به مقادیر قبلی خود دارند (یعنی همان ویژگی حافظه که در لچ وجود دارد). هرگاه بخواهیم مقدار خروجی لچ را تغییر دهیم، یکی از ورودی‌های S یا R را برای لحظه‌ای تغییر داده و سپس به مقدار 0 برمی‌گردانیم تا لچ دوباره در حالت «حفظ مقدار قبلی» قرار بگیرد. (توجه شود که (۱) هر دو ورودی را نباید همزمان از مقدار 00 به 11 تغییر داد زیرا یکی از دو ورودی اندکی زودتر 1 شده و برای مدت کوتاهی، خروجی‌های Q و Q' مقدار متفاوتی نسبت به حالت دیگر به خود می‌گیرند؛ گرچه در انتها و در هر دو حالت، خروجی‌های Q و Q' سرانجام به مقدار 00 خواهند رسید اما این نامشخص بودن لحظه‌ایِ خروجی‌ها ممکن است نامطلوب و دردسرساز باشد. (۲) هر دو ورودی را نباید همزمان از مقدار 11 به 00 تغییر داد به دلیلی که به زودی گفته خواهد شد).

خروجی‌های Q و Q' همیشه متمم یکدیگرند مگر در حالت SR=11 لذا توصیه می‌شود از وقوع این حالت جلوگیری و اجتناب شود. علت دیگر این توصیه این است که اگر لچ در حالت SR=11 باشد (و بنابراین، خروجی‌های Q و Q' هر دو مقدار 0 به خود بگیرند) و بخواهیم همزمان دو ورودی S و R را 0 کنیم، عملاً یکی از این دو ورودی اندکی زودتر از دیگری 0 شده و همین امر باعث می‌شود مقدار خروجی‌های لچ دقیقاً قابل تعیین نباشد (و بستگی به این داشته باشد که کدام ورودی زودتر به 0 رسیده است)

با استفاده از گیت NAND:

[1] Set
[2] Reset



(a) Logic diagram

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

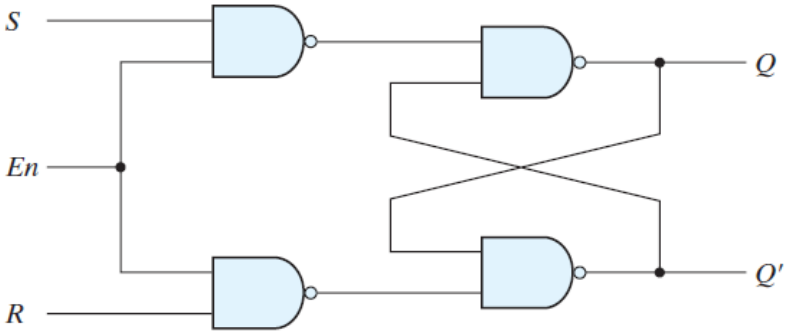
(b) Function table

FIGURE 5.4
***SR* latch with NAND gates**

مشاهده می‌شود که عملکرد این لچ متمم عملکرد لچ قبلی است؛ لذا گاهاً به آن لچ S'R' نیز گفته می‌شود.

اصلاح عملکرد لچ SR

با افزودن یک ورودی کنترلی به منظور تعیین زمان تغییر حالت لچ می‌توان عملکرد آن را تغییر داد.



(a) Logic diagram

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

(b) Function table

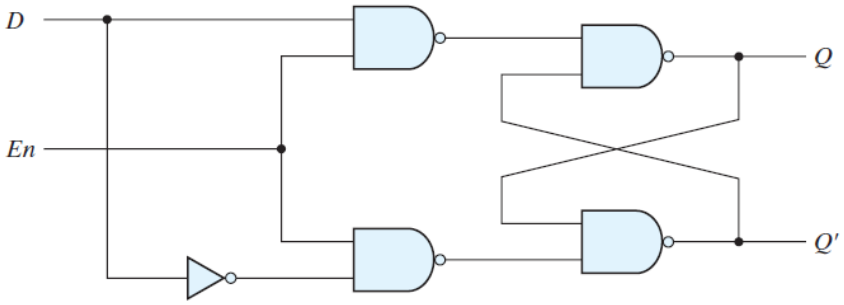
FIGURE 5.5
***SR* latch with control input**

در این جا حالت نامعین[1] زمانی رخ می‌دهد که هر سه ورودی برابر 1 باشند. با این کار، مقدار 0 روی هر دو ورودی لچ SR پایه قرار می‌گیرد که در نتیجه، این ورودی‌ها حالت نامعین را تولید می‌کنند. هنگامی که ورودی کنترل به 0 بازمی‌گردد، نمی‌توان حالت بعدی را معین کرد زیرا به این بستگی دارد که کدام یک از دو ورودی S و R زودتر به 0 (به 1 ؟؟ـ م) بروند. این حالت نامعین موجب می‌شود مدیریت مدار مشکل شود و بنابراین، به ندرت از این مدار استفاده می‌شود. با این وجود، این مدار از اهمیت لازم برخوردار است زیرا سایر لچ‌ها و فلیپ‌فلاپ‌های مهم از روی این لچ ساخته می‌شوند.

[1] Undefined

لچ D (یا لچ شفاف[1])

یک راه برای حذف حالت نامعین در لچ SR این است که مطمئن شویم دیگر حالت SR=11 رخ نمی‌دهد.



(a) Logic diagram

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

(b) Function table

FIGURE 5.6
D latch

علت نامگذاری «لچ D» قابلیت نگهداری و ذخیره‌ی داده در این نوع لچ است. علت نامگذاری «لچ شفاف» این است که مادام که کنترل C (یا همان En در شکل اخیر) برابر 1 است، هر گونه تغییرات و مقادیر ورودی D به خروجی منتقل می‌شود. لچ D اخیر از نوع فعال‌ـ بالا است. اگر در ورودی En یک گیت NOT قرار دهیم، از نوع فعال‌ـ پایین می‌شود. نمودار گرافیکی سه نوع لچی که تاکنون دیدیم (لچ SR، لچ S'R'، و لچ D) در شکل زیر نشان داده شده است.

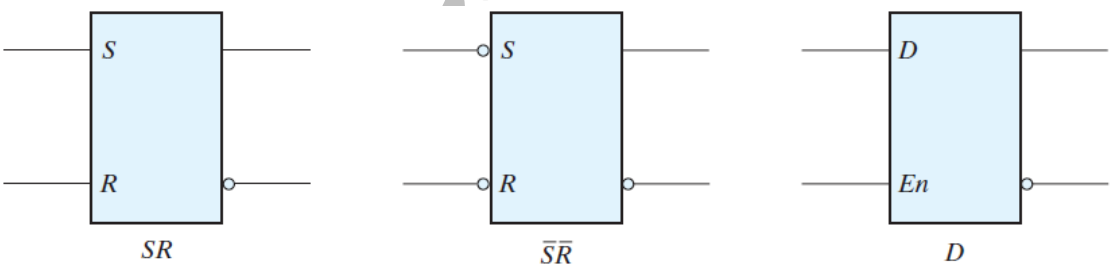


FIGURE 5.7
Graphic symbols for latches

فلیپ‌فلاپ‌ها

مشکل یک لچ این است که در تمام بازه‌ی زمانی که ورودی کنترل مقدار فعال (مثلاً 1) دارد، هر لحظه که ورودی تغییر کند، خروجی نیز تغییر می‌کند (شکل زیر رفتار یک فلیپ‌فلاپ و یک لچ را نشان می‌دهد). لذا اگر یک لچ را در ساختار یک مدار ترتیبی که در شکل ۵ـ۲ نشان داده شده است، به کار بگیریم، اگر ورودی کنترل 1 شود، لچ شروع به تغییر

[1] Transparent latch

خروجی خود بر حسب مقدار ورودی اعمال شده به آن می کند. خروجی این لچ از طریق یک مدار ترکیبی دوباره به ورودی خود آن لچ متصل شده است؛ لذا (با فرض این که ورودی کنترل همچنان 1 است،) ممکن است ورودی جدیدی به لچ اعمال شده و خروجی نیز به تبع آن، تغییر کند و این وضعیت همچنان ادامه پیدا کرده و بنابراین، ناپایداری داشته باشیم.

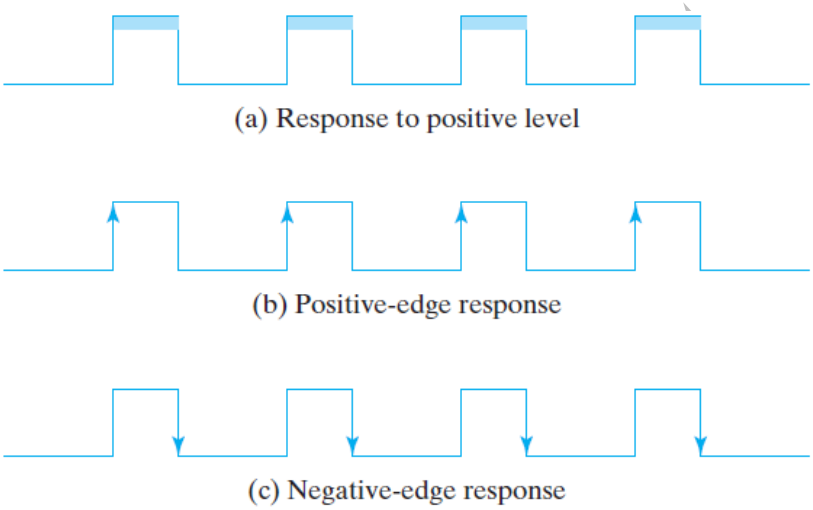


FIGURE 5.8
Clock response in latch and flip-flop

در فلیپ‌فلاپ‌ها این مشکل با حساس کردن آن به لبه‌ی (و نه سطح) سیگنال حل شده است. برای اصلاح رفتار لچ و تهیه‌ی یک فلیپ فلاپ دو راه وجود دارد که در ادامه به بررسی آنها پرداخته می‌شود.

فلیپ فلاپ حساس به لبه

این فلیپ فلاپ از دو لچ SR که در آرایش حاکم‌ـ تابع[1] به هم متصل شده‌اند، تشکیل شده است.

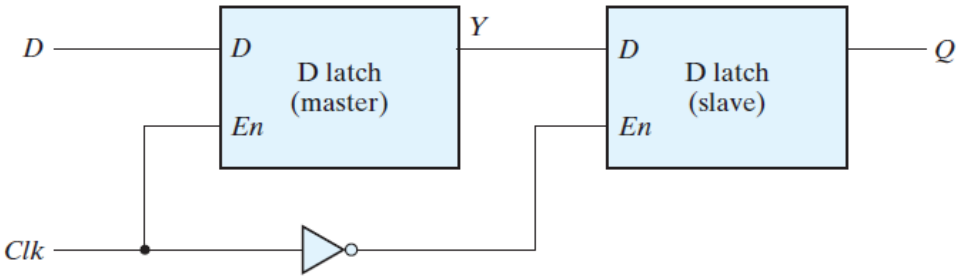


FIGURE 5.9
Master–slave *D* flip-flop

[1] Master–Slave

فلیپ فلاپ فوق حساس به لبه‌ی پایین رونده (یا لبه‌ی منفی) است. برای تحریک[1] آن با لبه‌ی بالا رونده (یا لبه‌ی مثبت) می‌توان از یک گیت NOT در ورودیِ Clk استفاده کرد.

فلیپ فلاپ D حساس به لبه

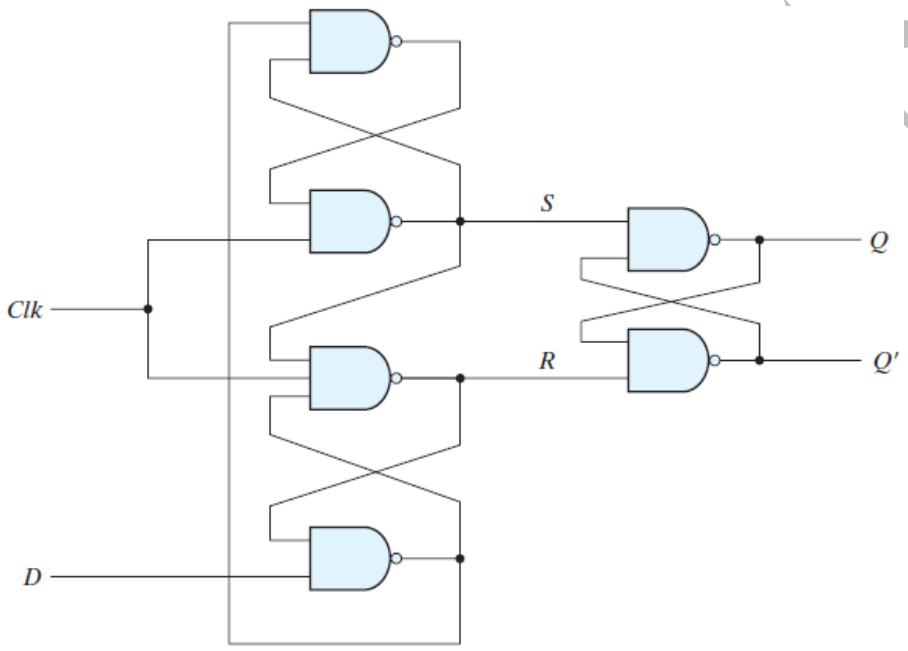


FIGURE 5.10
D-type positive-edge-triggered flip-flop

این فلیپ فلاپ از سه لچ SR تشکیل شده است. دو لچ ورودی و یک لچ خروجی. در حالت Clk=0 دو ورودی S و R لچ خروجی مقدار 1 داشته و بنابراین، خروجی فلیپ فلاپ (Q و Q') در حالت فعلی خود حفظ می‌شوند. حال، اگر کلاک را به 1 تغییر دهیم بر حسب این که قبل از این لحظه، ورودی D چه مقداری داشته است یکی از ورودی‌های S و R لچ خروجی تغییر می‌کنند. اگر هنگام وقوع لبه‌ی بالارونده‌ی کلاک، D=0 باشد، R به 0 تغییر می‌کند (S بدون تغییر باقی می‌ماند) و بنابراین، فلیپ فلاپ به حالت ریست (بازنشانی) رفته و Q=0 می‌شود. در طول تمام بازه‌ی زمانی که Clk=1 است، اگر تغییری در ورودی D رخ دهد، پایه‌ی R در همان وضعیت 0 خود باقی می‌ماند و بنابراین، تغییرات D دیگر به خروجی Q نمی‌رسد و فلیپ فلاپ اصطلاحاً قفل کرده است. حال اگر کلاک به 0 بازگردد، R به 1 برگشته و بنابراین، مجدداً فلیپ فلاپ به حالت حفظ حالت فعلی برمی‌گردد (بنابراین، لبه‌ی پایین رونده‌ی کلاک تاثیری روی فلیپ فلاپ ندارد).

به طور مشابه، اگر در حین وقوع لبه‌ی بالارونده‌ی کلاک، ورودیِ D مقدار 1 داشته باشد، پایه‌ی S به مقدار 0 تغییر کرده (R همچنان 1 باقی می‌ماند) و بنابراین، فلیپ فلاپ به حالت ست رفته و خروجیِ Q مقدار 1 می‌گیرد. در طول تمام بازه‌ی

[1] Trigger

زمانی که Clk=1 است، اگر تغییری در ورودی D رخ دهد، پایه‌ی S در همان وضعیت 0 خود باقی می‌ماند و بنابراین، تغییرات D دیگر به خروجی Q نمی‌رسد و فلیپ فلاپ اصطلاحاً قفل کرده است. حال اگر کلاک به 0 بازگردد، S به 1 برگشته و بنابراین، مجدداً فلیپ فلاپ به حالت حفظ حالت فعلی برمی‌گردد.

نماد گرافیکی فلیپ فلاپ حساس به لبه

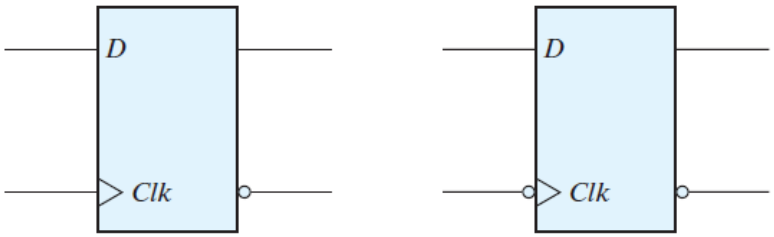


FIGURE 5.11
Graphic symbol for edge-triggered *D* flip-flop

علامت مثلث نشان‌دهنده‌ی حساس بودن فلیپ فلاپ به لبه است. وجود علامت حباب[1] نشان‌دهنده‌ی حساس بودن فلیپ فلاپ به لبه‌ی منفی و در غیر این صورت، لبه‌ی مثبت است.

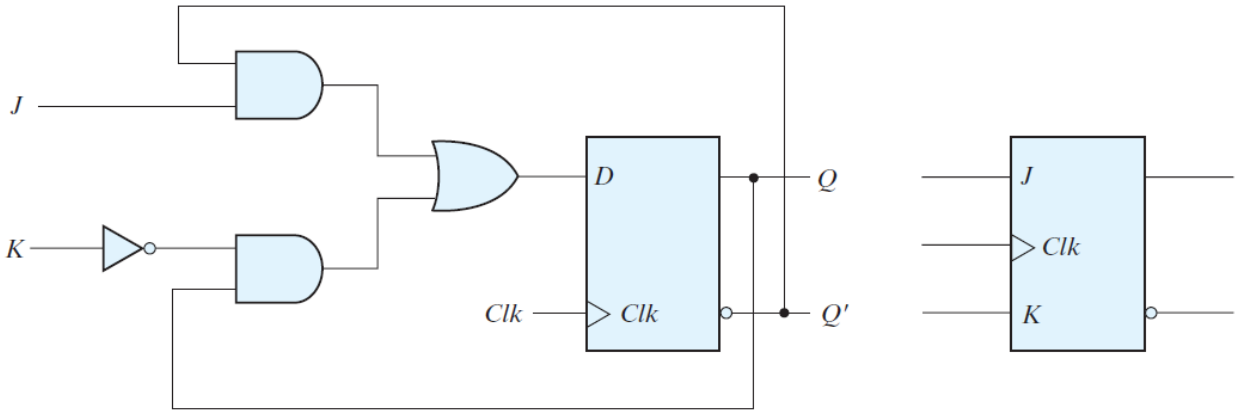
فلیپ‌فلاپ‌های دیگر

گرچه فلیپ فلاپ D (با نام اختصاری DFF) ساده‌ترین و متداولترین فلیپ فلاپ است (زیرا از کمترین تعداد گیت‌ها تشکیل شده است)، اما فلیپ‌فلاپ‌های دیگری هم هستند که از آنها در کاربردهای مختلف استفاده می‌شود. از جمله‌ی این فلیپ‌فلاپ‌ها، فلیپ فلاپ JK (با نام اختصاری JKFF) و فلیپ فلاپ T (TFF) است.

فلیپ فلاپ JK (JKFF)

در حالت کلی، یک فلیپ فلاپ قادر به انجام سه کار است: ۱ کردن، صفر کردن، و متمم کردن. فلیپ فلاپ D تنها قادر به انجام دو کار اول است؛ در مقابل، فلیپ فلاپ JK قادر به انجام هر سه کار است. این فلیپ فلاپ به صورت زیر از روی یک DFF قابل ساخت است. در این شکل، سمبل یا نماد گرافیکی JKFF نیز نشان داده شده است.

[1] Bubble



(a) Circuit diagram (b) Graphic symbol

FIGURE 5.12
JK flip-flop

در JKFF، کار اصلی را یک DFF برعهده دارد که معادله‌ی ورودیِ D این فلیپ فلاپ به صورت زیر است:

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

با آمدن هر لبه‌ی بالارونده‌ی کلاک، مقداری که در ورودی D ایجاد شده است، به خروجی Q (و متمم این مقدار به خروجیِ Q') می‌رود. بنابراین، میتوان از جدول زیر که به نام «جدول مشخصه[1]» شناخته می‌شود، برای تعریف رابطه‌ی مقدار بعدیِ خروجیِ Q (که با علامت Q(t+1) مشخص شده است) بر حسبِ ورودی‌های JKFF و مقدار قبلیِ خروجی Q (که با علامت Q(t) مشخص شده است،) استفاده کرد:

JK Flip-Flop

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

جدول مشخصه‌ی DFF نیز به صورت زیر است:

D Flip-Flop

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

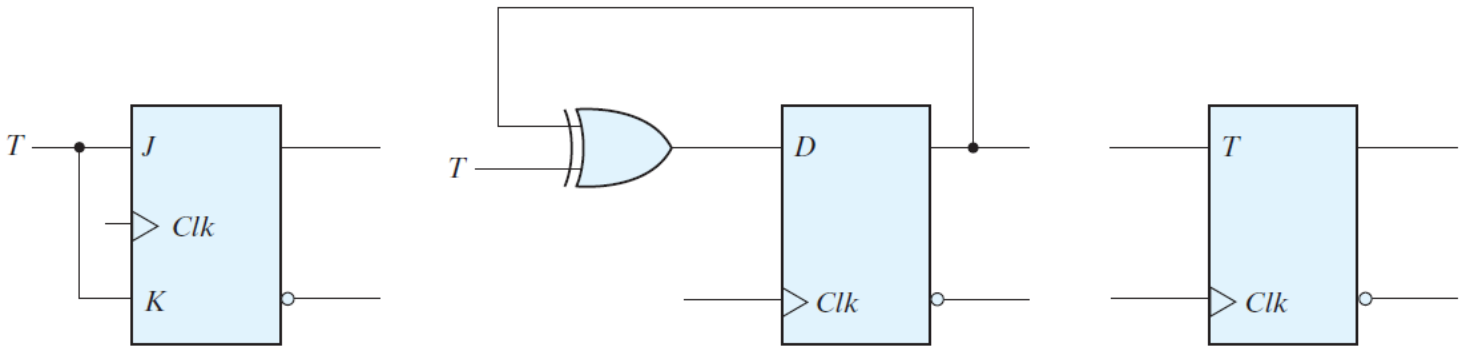
[1] Characteristics table

ملاحظه می‌کنید که DFF به خودی خود، فاقد قابلیت حفظ مقدار قبلی است.

فلیپ فلاپ T (TFF)

فلیپ فلاپ T[1] (فلیپ فلاپ دگر وضع) یک فلیپ فلاپ متمم‌ساز است. یعنی، علاوه بر عمل حفظ مقدار قبلی، می‌تواند مقدار خروجی (یا حالت) خود را متمم کرده و به عنوان حالت جدید خارج کند.

برای ساخت یک JKFF می‌توان از فلیپ فلاپ D و یا از فلیپ فلاپ JK استفاده کرد. شکل زیر این دو روش را به همراه سمبل گرافیکی این فلیپ فلاپ نشان می‌دهد.



(a) From *JK* flip-flop (b) From *D* flip-flop (c) Graphic symbol

FIGURE 5.13
T flip-flop

با توجه به مدار شکل (ب)، با آمدن لبه‌ی بالارونده‌ی کلاک، مقداری که در ورودی D وجود دارد، به خروجی منتقل شده و حالت بعدی، Q(t+1) را تعیین می‌کند. در این مدار داریم:

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

بنابراین، جدول مشخصه‌ی TFF به صورت زیر خواهد بود:

T Flip-Flop

T	Q(t + 1)	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

مدار شکلی (الف) هم همین رفتار را نشان می‌دهد.

[1] Toggle Flip Flop

معادله‌ی مشخصه

معادله‌ی مشخصه[1]، معادله‌ای است که حالت بعدی فلیپ فلاپ را بر حسب ورودی(های) فلیپ فلاپ و حالت فعلی آن مشخص می‌کند. منظور از «حالت فعلی فلیپ فلاپ» مقدار خروجی آن تا قبل از آمدن لبه‌ی بالارونده‌ی کلاک و منظور از «حالت بعدی فلیپ فلاپ» خروجی آن بعد از رخ دادن لبه‌ی بالارونده‌ی کلاک است. البته یک کلاک می‌تواند حساس به لبه‌ی پایین‌رونده باشد؛ در این صورت، لبه‌ی پایین‌رونده ملاک تشخیص خواهد بود.

معادلات مشخصه برای سه نوع فلیپ فلاپی که دیدیم:

برای DFF:

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

برای JKFF:

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

برای TFF:

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

برخی ورودی‌های مستقیم با عملکرد خاص برای فلیپ‌فلاپ‌ها

برخی تعاریف:

ورودی سنکرون: به ورودی‌ای گفته می‌شود که هرگاه فعال شود، باید در حالت فعال باقی بماند تا این که لبه‌ی فعال کلاک رخ دهد. پس از رخ دادن لبه‌ی فعال کلاک، عمل مربوط به آن ورودی انجام خواهد شد.

ورودی آسنکرون: به ورودی‌ای گفته می‌شود که به محض فعال شدن و بدون بستگی به حالت کلاک، عمل مربوط به آن ورودی انجام می‌شود.

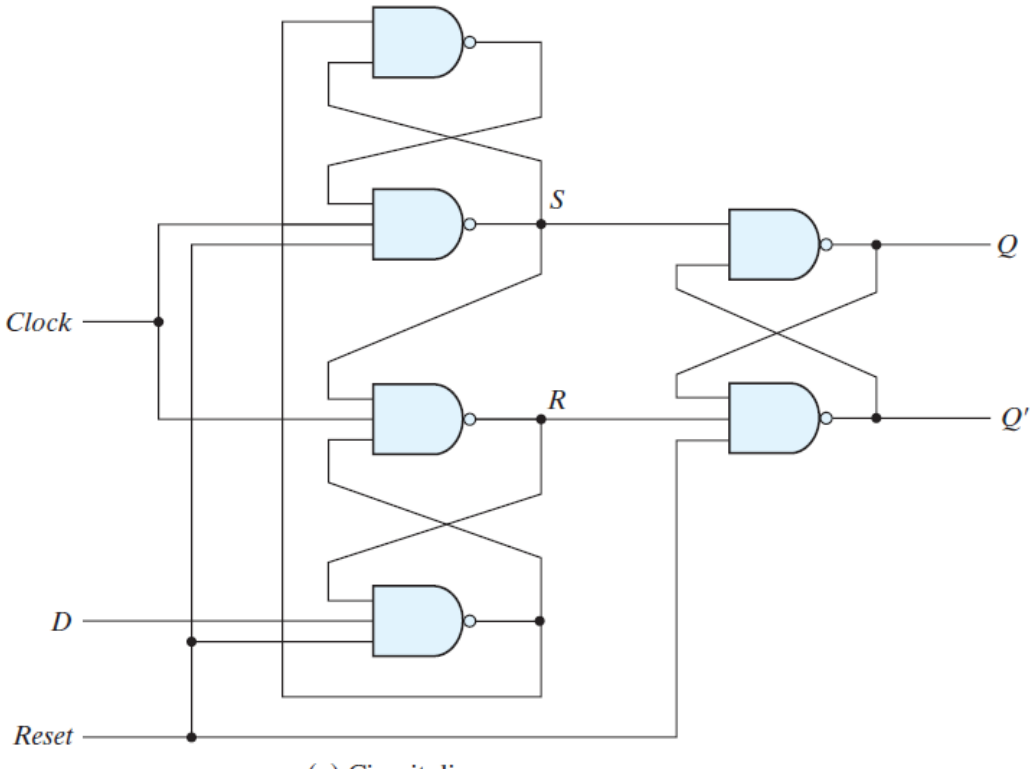
ورودی بازنشانی/ریست (Reset) یا پاک کردن (Clear): به ورودی‌ای گفته می‌شود که هرگاه فعال شود، خروجی فلیپ فلاپ صفر می‌شود.

ورودی پیش‌نشاندن/پریست (Preset) یا نشاندن مستقیم (Direct set): به ورودی‌ای گفته می‌شود که هرگاه فعال شود، خروجی فلیپ فلاپ ۱ می‌شود.

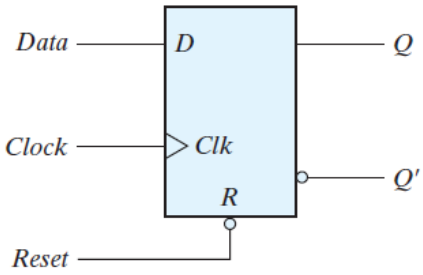
[1] Characteristics equation

علت نیاز به ورودی‌های خاص فوق‌الذکر: خروجیِ فلیپ فلاپ‌ها پس از وصل شدن تغذیه‌شان، دارای مقدار از قبل مشخصی نبوده و رفتاری تصادفی دارند؛ لذا برای اجبار این خروجی به شروع شد از یک وضعیت از قبل مشخص می‌توانیم از ورودی‌های خاص و مستقیم که اشاره شد، استفاده کنیم.

یک نمونه ورودی ریست آسنکرون فعال‌ـ پایین که در یک DFF حساس به لبه‌ی مثبت استفاده شده است، در شکل زیر نشان داده شده است.



(a) Circuit diagram



(b) Graphic symbol

R	Clk	D	Q	Q′
0	X	X	0	1
0	↑	0	0	1
0	↑	1	1	0

(b) Function table

FIGURE 5.14
D flip-flop with asynchronous reset

(توجه: مقدار R در دو سطر انتهایی جدول عملکرد اخیر، باید 1 باشد)

تحلیل مدارهای ترتیبی کلاک‌دار

(۱) منظور از «مدار ترتیبی کلاک‌دار» یک مدار شامل تعدادی فلیپ‌فلاپ کلاک‌دار است؛ چنین مداری می‌تواند شامل مدارات ترکیبی نیز باشد اما باز هم یک مدار «ترتیبی» نامیده می‌شود.

(۲) منظور از تحلیل مدار، توصیف «رفتار مدار» در شرایط کاری مختلف است. عواملی که «رفتار مدار» را تشکیل می‌دهند شامل «ورودی‌های مدار»، «خروجی‌های مدار»، و «حالات مدار» است. منظور از یک «حالت مدار» همان حالت/خروجیِ یک فلیپ‌فلاپ است. بنابراین، یک مدار به تعداد فلیپ‌فلاپ‌هایش، «حالت» دارد.

(۳) «خروجی مدار» با «حالت مدار» لزوماً یکی نیست؛ هر «حالت مدار» می‌تواند «یک خروجی مدار» نیز محسوب شود بسته به این که طراح آن مدار چه تصمیم گرفته باشد. اما هر خروجی مدار، لزوماً نشان دهنده‌ی یک حالت مدار نیست.

(۴) ابزارهای تحلیل مدار عبارتند از: «دیاگرام منطقی[1]»، «جدول حالت[2]»، «دیاگرام/نمودار حالت[3]»، «معادله‌ی حالت[4] یا معادله‌ی گذر[5]»، «معادله‌ی ورودی[6] یا معادلات تحریک[7]» و «معادله‌ی خروجی[8]».

(۵) در «تحلیل مدار»، دیاگرام منطقیِ مدار مشخص است (یا به سادگی قابل شناسایی و ترسیم است) و ما باید ابزارهای اشاره شده را در مورد مدار محاسبه یا نمایش دهیم. در مقابل، در «طراحی مدار»، دیاگرام منطقیِ مدار از قبل مشخص نیست اما رفتار مطلوب مدار را به ما داده‌اند و ما باید دنبال یافتن مداری باشیم که رفتار خواسته شده از ما را بر آورده کند.

دیاگرام منطقی: به ترسیم گرافیکی یک مدار با استفاده از نمادهای گرافیکی گیت‌های منطقی و اتصال مناسب آنها با خطوط گفته می‌شود.

معادله‌ی ورودی: عبارتی جبری است که رابطه‌ی ورودی/ورودی‌های هر فلیپ فلاپ را بر حسب ورودی‌های مدار و حالات فعلی مدار توصیف می‌کند.

معادله‌ی خروجی: عبارتی جبری است که خروجی یا خروجی‌های مدار را بر حسب ورودی‌های مدار و حالات فعلی مدار توصیف می‌کند.

جدول حالت: جدولی است که حالات بعدی مدار را بر حسب حالات فعلی و ورودی‌های آن مدار توصیف می‌کند. «**معمولاً برای راحتی مقادیر خروجی را نیز در داخل جدول حالت و به ازاء ترکیبات مختلف ورودی‌ها و حالات فعلی مدار وارد می‌کنند**ـ در این زمینه به چند صفحه بعد، «مثالی از جدول حالت» مراجعه کنید».

---

[1] Logic diagram
[2] State table
[3] State diagram
[4] State equation
[5] Transition equation
[6] Input equation
[7] Excitation equation
[8] Output equation

معادله‌ی حالت: معادله یا معادله‌هایی (یا همان عبارت‌های جبری) است که حالات بعدی مدار را بر حسب حالات فعلی و ورودی‌های آن مدار توصیف می‌کنند. «**با ترکیب معادلات ورودی و معادله‌ی مشخصه‌ی هر فلیپ فلاپ، می‌توان به معادلات حالت رسید**».

دیاگرام حالت: ترسیمی گرافیکی از حالت‌های مختلف موجود در مدار و نحوه‌ی حرکت/تغییر از یک حالت به حالت دیگر بسته به مقادیر مختلف ورودی‌های مدار است. **دیاگرام حالت، ترجمه‌ای گرافیکی از جدول حالت است.**

تعداد حالتهای مدار برابر با تعداد فلیپ‌فلاپ‌ها (N) و تعداد دایره‌ها در دیاگرام حالت، برابر با 2^N است.

مثالی از معادله‌ی حالت:

اگر فرض کنیم دیاگرام منطقی یک مدار به صورت زیر باشد،

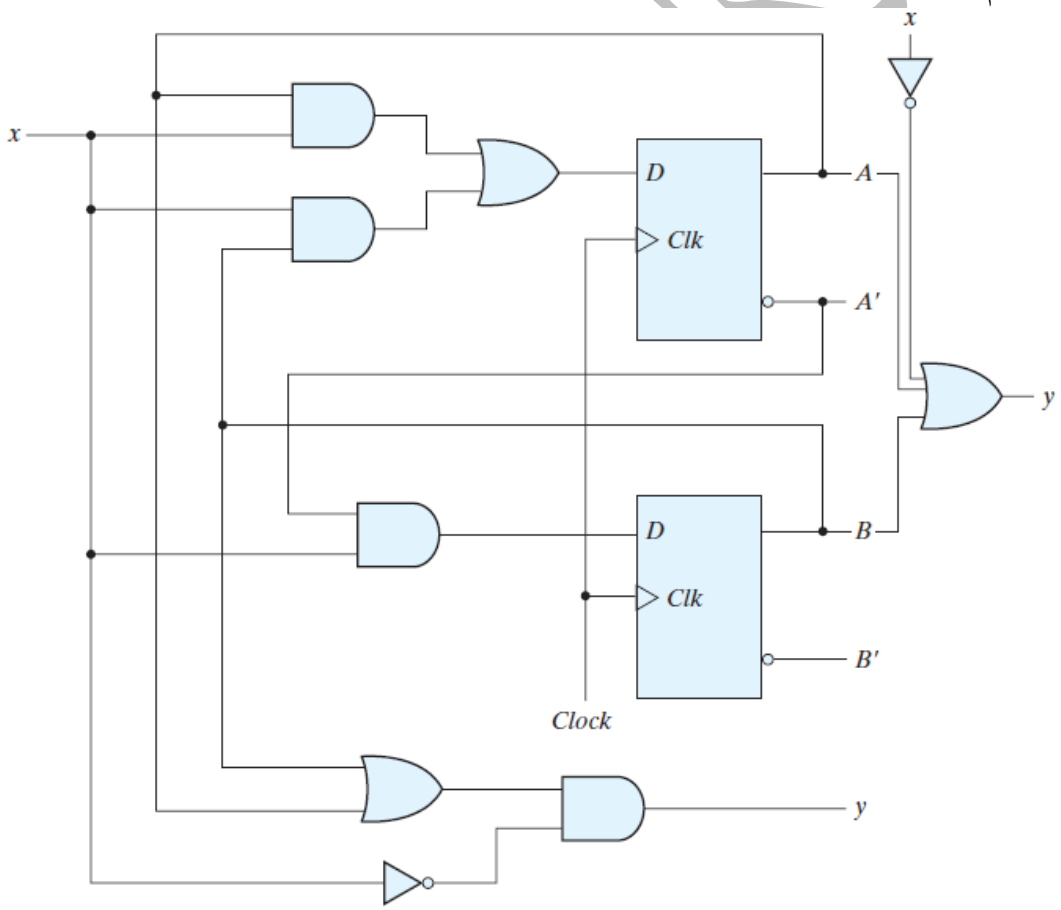


FIGURE 5.15
Example of sequential circuit

معادلات حالت را به صورت زیر می‌توان به دست آورد:

$$A(t+1) = A(t)x(t) + B(t)x(t)$$

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

برای سادگی و اختصار در نوشتن، می‌توان معادلات اخیر را به صورت زیر نیز نوشت:

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

مثالی از معادله‌ی خروجی:

در مدار مثال قبل داریم:

$$y(t) = [A(t) + B(t)]x'(t)$$

این معادله‌ی خروجی را به صورت فشرده‌تر زیر نیز می‌توانیم بنویسیم:

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

مثالی از جدول حالت:

در مدار مثال قبل، جدول حالت به صورت زیر قابل نوشتن است:

Table 5.2
State Table for the Circuit of Fig. 5.15

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

این جدول را به صورت فشرده‌تر زیر نیز می‌توان نوشت:

Table 5.3
Second Form of the State Table

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

مثالی از دیاگرام/نمودار حالت:

همان طور که اشاره شد، «دیاگرام حالت، ترجمه‌ای گرافیکی از جدول حالت است». متناظر با هر سطر از جدول حالت، یک خط جهت‌دار در دیاگرام حالت وجود دارد و بالعکس؛ بنابراین، «**از جدول حالت می‌توان به دیاگرام حالت و از دیاگرام حالت، می‌توان به جدول حالت رسید**».

در مدار مثال قبلی، تعداد حالتهای مدار برابر ۲ (برابر با تعداد فلیپ‌فلاپ‌ها) و بنابراین، تعداد دایره‌ها برابر با $2^2=4$ است. دیاگرام حالت به صورت زیر است:

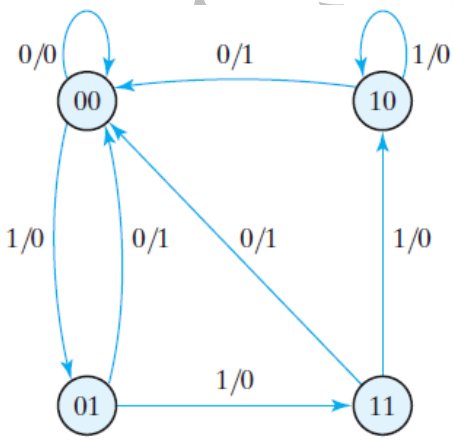


FIGURE 5.16
State diagram of the circuit of Fig. 5.15

مثالی از معادله‌ی ورودی:

در مدار مثال قبل، معادلات ورودی به صورت زیر هستند:

$$D_A = Ax + Bx$$
$$D_B = A'x$$

نام‌گذاری D_A به معنای ورودی یک DFF است که خروجی آن فلیپ فلاپ، A نام دارد.

مثالی از تحلیل مدار مبتنی بر DFF:

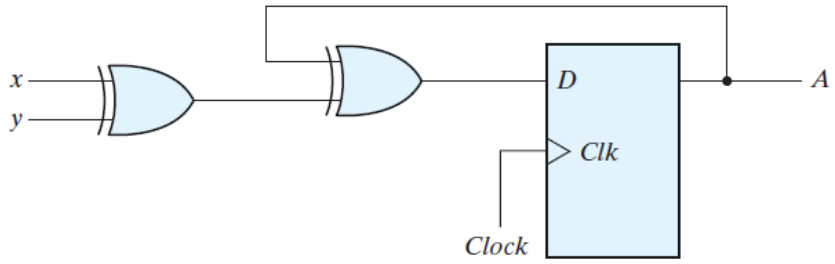
فرض کنیم تنها معادله‌ی ورودی یک مدار به صورت زیر به ما داده شده باشد:

$$D_A = A \oplus x \oplus y$$

می‌خواهیم مدار را تحلیل کنیم. از داده‌های مساله، اطلاعات زیر معلوم می‌شود:

- مدار تنها دارای یک DFF است که خروجی آن، A نام دارد.
- مدار دارای دو ورودی خارجی/بیرونی به نام‌های x و y است.
- برای مدار هیچ خروجی تعریف یا در نظر گرفته نشده است زیرا معادله‌ی خروجی به ما داده نشده است.

با توجه به معادله‌ی ورودی، دیاگرام منطقی مدار به صورت زیر قابل ترسیم است:



(a) Circuit diagram

با ترکیب معادله‌ی مشخصه‌ی DFF (یعنی معادله‌ی Q(t+1)=D) با معادله‌ی ورودی داده شده در صورت مساله، می‌توان به معادله‌ی حالتِ زیر رسید:

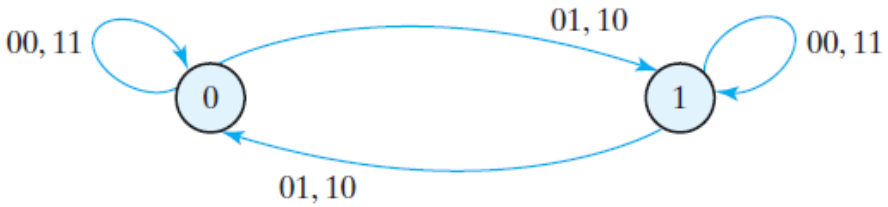
$$A(t+1) = A \oplus x \oplus y$$

حالا، با در دست داشتن معادله‌ی حالت، می‌توان به راحتی به جدول حالت رسید. این جدول به صورت زیر می‌شود:

Present state	Inputs		Next state
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

(b) State table

دیاگرام حالت ترجمه‌ی سطر به سطرِ جدول حالت است؛ بنابراین، در این جا دیاگرام حالت به صورت زیر خواهد بود:



(c) State diagram

مثالی از تحلیل مدار مبتنی بر JKFF:

مدار زیر را در نظر بگیرید:

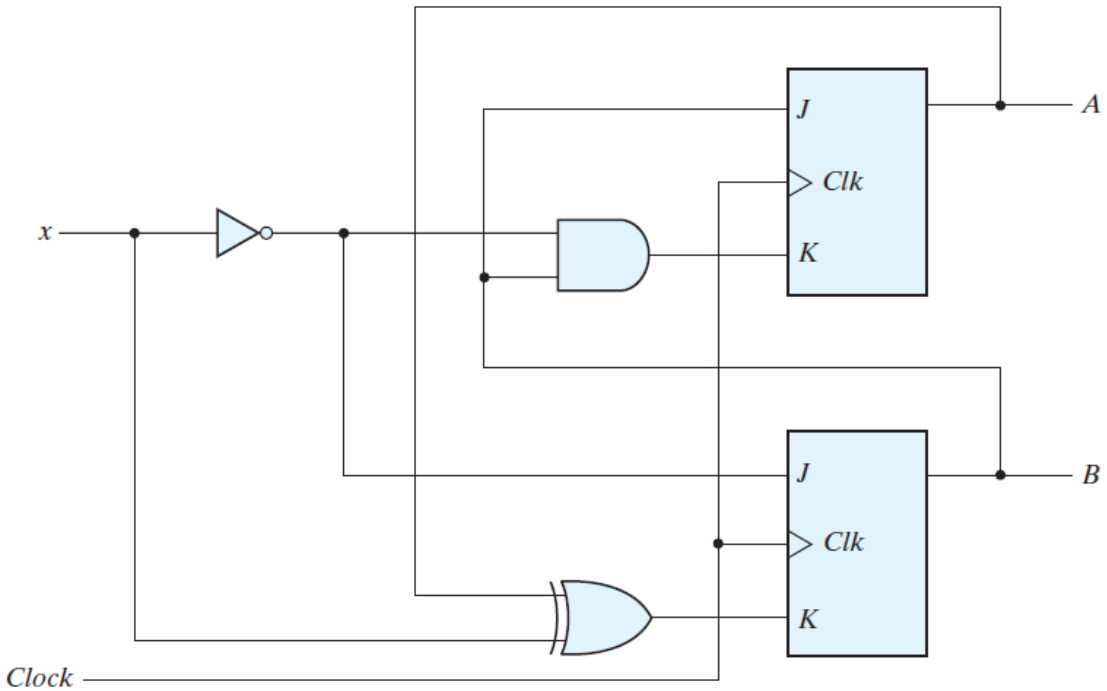


FIGURE 5.18
Sequential circuit with *JK* flip-flop

مدار فاقد خروجی است پس نیازی به نوشتن معادلات خروجی نبوده و بنابراین در جدول حالت نیازی به تخصیص ستونهایی به خروجی‌ها نیست.
معادلات ورودی (فلیپ فلاپ‌ها) به صورت زیر است:

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

راه اول برای محاسبه‌ی جدول حالت:
حالا با توجه به این معادلات ورودی، معادله‌ی مشخصه‌ی فلیپ‌فلاپ‌ها، و با توجه به ترکیبات ممکن حالات فعلی و ورودی‌های مدار (در این جا x) می‌توانیم حالت بعدی را محاسبه کرده و در نتیجه، جدول حالت را کامل کنیم.

راه دوم برای محاسبه‌ی جدول حالت:
ابتدا معادلات حالت را به دست آورده و سپس از روی آن، جدول حالت را پر می‌کنیم. برای به دست آوردن معادلات حالت، معادلات ورودی را در معادلات مشخصه‌ی فلیپ فلاپ‌ها جایگذاری می‌کنیم.

استفاده از راه اول: جدول حالت به صورت زیر خواهد بود:

Table 5.4
State Table for Sequential Circuit with JK Flip-Flops

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

استفاده از راه دوم: معادلات حالت به صورت زیر به دست می‌آیند:

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

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

حالا به راحتی جدول حالت پر شده و در این حالت، نیازی به وارد کردن ستونهای مربوط به مقادیر ورودی‌های فلیپ‌فلاپ‌ها در جدول حالت (آن طور که در جدول حالت اخیر انجام شده است،) نیست.
حال، دیاگرام حالت به سادگی از روی جدول حالت مانند شکل زیر، قابل ترسیم است. در این شکل تنها مقادیر ورودی x روی خطوط جهت‌دار مشخص شده است زیرا مدار فاقد خروجی است.

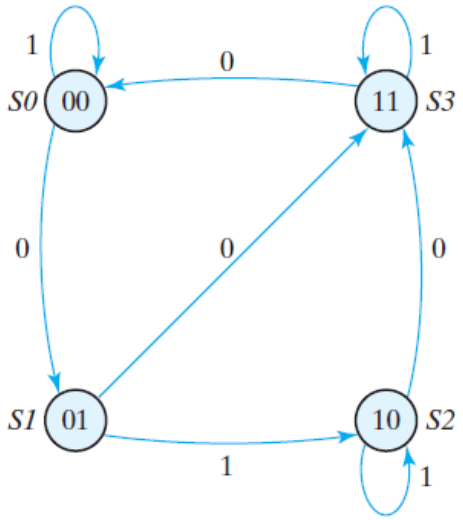
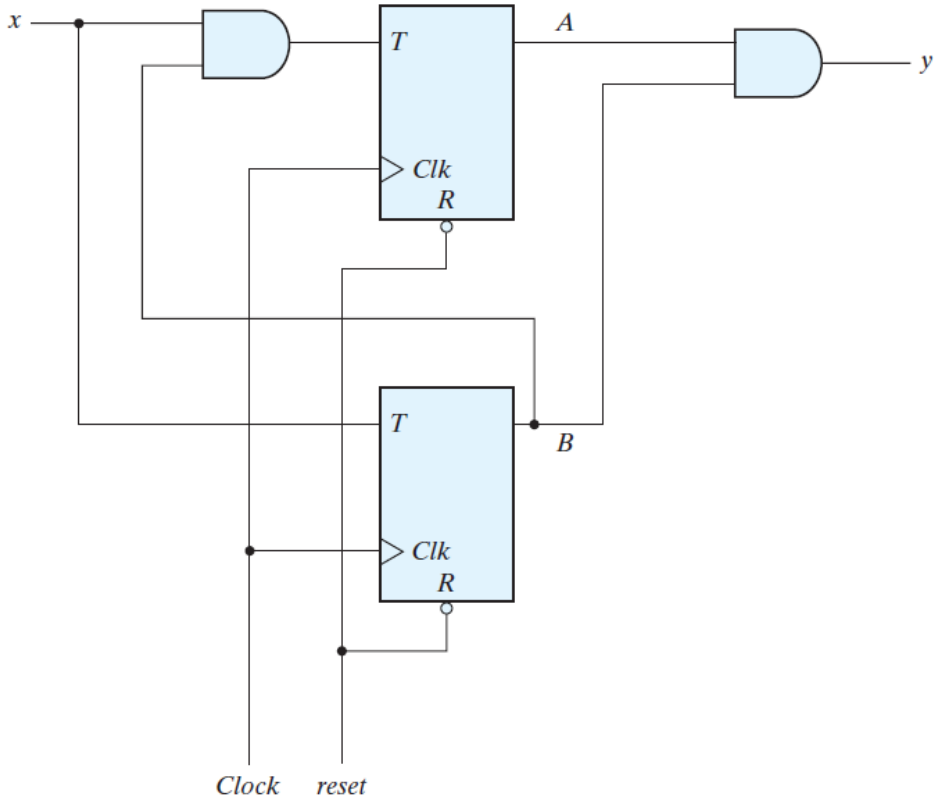


FIGURE 5.19
State diagram of the circuit of Fig. 5.18

مثالی از تحلیل مدار مبتنی بر TFF:

مدار زیر را در نظر بگیرید.



(a) Circuit diagram

معادلات ورودی و خروجی مدار به صورت زیر است:

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

در این جا هم می‌توانیم به کمک جدول مشخصه (روش اول) و یا معادله‌ی مشخصه‌ی فلیپ‌فلاپ (روش دوم) به جدول حالت و از آن جا به دیاگرام/نمودار حالت برسیم. اگر از روش دوم عمل کنیم، به کمک معادلات ورودی و معادلات مشخصه، می‌توانیم به صورت زیر به معادلات حالت مدار برسیم:

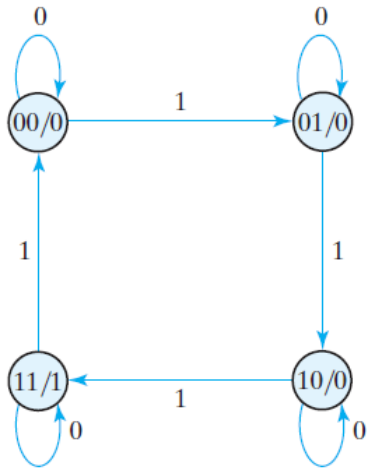
$$A(t+1) = (Bx)'A + (Bx)A' = AB' + Ax' + A'Bx$$
$$B(t+1) = x \oplus B$$

در هر حال، جدول حالت به صورت زیر به دست خواهد آمد:

Table 5.5
State Table for Sequential Circuit with T Flip-Flops

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

و دیاگرام حالت به صورت نمایش داده شده در شکل زیر خواهد بود:



(b) State diagram

مدل‌های میلی و مور برای ماشین‌های حالت متناهی

معمولاً مدارهای ترتیبی در دو قالب یا مدل به نام مدل میلی (Mealy) و مدل مور (Moore) و مطابق با شکل‌های زیر ارائه و پیاده‌سازی می‌شوند. تفاوت این دو مدل تنها مربوط به نحوه‌ی تولید خروجی است (این مطلب در شکل‌های زیر نیز به خوبی مشخص است).

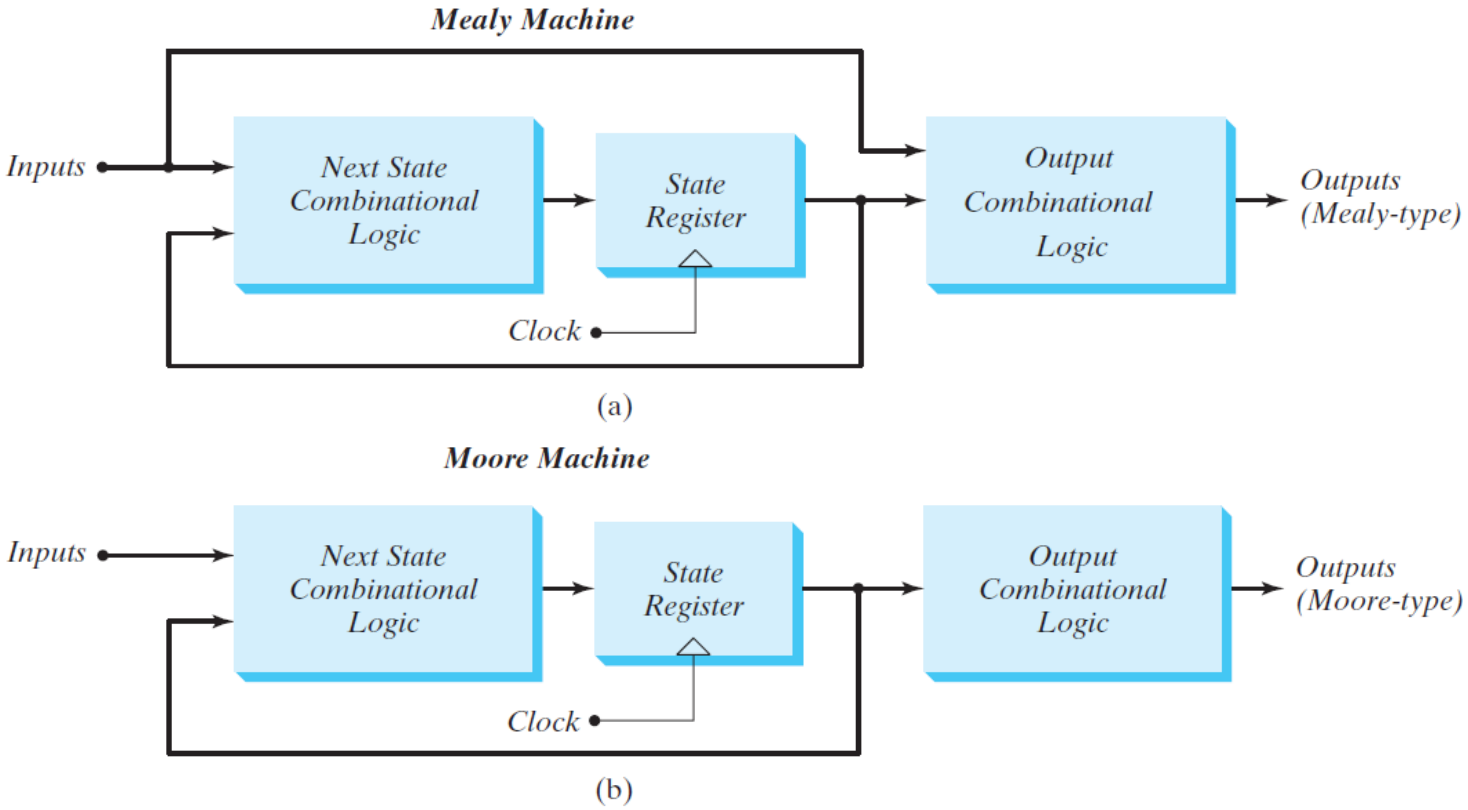


FIGURE 5.21
Block diagrams of Mealy and Moore state machines

در مدل میلی، خروجی‌ها تابعی از ورودی‌های مدار و حالات فعلی مدار هستند؛ اما در مدل مور، خروجی‌ها فقط تابعی از حالات فعلی می‌باشند. به دو مدل فوق‌الذکر، ماشین حالت متناهی یا FSM[1] گفته می‌شود.

نمونه‌ای از مدل میلی، مدار شکل ۵ـ۱۵ و نمونه‌ای از مدل مور، مدار شکل ۵ـ۱۸ و مدار شکل ۵ـ۲۰ است.

در مدل مور، خروجی‌های مدار ترتیبی با کلاک همزمان (یا سنکرون) هستند زیرا این خروجی‌ها تنها به خروجی‌های فلیپ‌فلاپ‌ها وابسته هستند که این خروجی‌ها نیز به نوبه‌ی خود به پالس کلاک وابسته هستند. اما در یک مدل میلی، هر زمان که ورودی‌ها تغییر کنند، حتی در طول زمانیِ پالس کلاک، خروجی نیز تغییر می‌کند. خروجی‌های مدل میلی ممکن است مقادیر لحظه‌ایِ غلط و اشتباه به خود بگیرند زیرا در این نوع از مدارات، از لحظه‌ای که ورودی‌های مدار تغییر می‌کنند تا لحظه‌ای که خروجیِ فلیپ فلاپ‌ها تغییر می‌کنند مقداری تاخیر وجود داشته و در طول همین بازه‌ی زمانی، خروجی‌های

[1] Finite State Machine

مدار مقادیر لحظه‌ای نامعتبری به خود می‌گیرند (گاهاً به این تغییرات نامطلوب، جهش یا glitch گفته می‌شود). برای حل این مشکل و بنابراین، سنکرون کردن مدار میلی، باید ورودی‌های مدار با پالس کلاک سنکرون شوند (یعنی تغییرات ورودی‌ها هماهنگ با لبه‌ی کلاک باشد؛ بدین ترتیب که) خروجی‌های مدار «**قبل از لبه‌ی فعال کلاک**» نمونه‌برداری شده و ورودی‌های مدار در لبه‌های غیرفعال مدار تغییر کنند تا ثبات و پایداری آنها قبل از وقوع لبه‌ی فعال کلاک، تضمین شود؛ این امر باعث می‌شود خروجی‌ها درست قبل از لبه‌ی فعال کلاک معتبر و آماده برای نمونه‌برداری باشند.

کاهش حالت

منظور از «کاهش حالت» الگوریتمهایی به منظور کاهش تعداد حالات موجود در دیاگرام حالت است؛ با کاهش تعداد حالات موجود در دیاگرام حالت می‌توان امید داشت تعداد فلیپ‌فلاپ‌های مورد استفاده در مدار نیز کاهش یابد گرچه لزوماً همیشه این امر رخ نخواهد داد؛ همچنین، نکته‌ی دوم این است که کاهش تعداد فلیپ فلاپ‌ها ممکن است موجب افزایش حجم مدار ترکیبی مورد نیاز شود. در هر حال، این روش را می‌توان به نوعی معادل با روش «جدول کارنو» دانست که به جای ساده‌سازی مدارات ترکیبی، به ساده‌سازی مدارات ترتیبی می‌پردازد.

کاهش حالت در مواقعی که تنها دنباله‌ی ورودی و خروجی مدار مهم هستند، مفید است؛ در این مدارات، حالات داخلی فقط به منظور تولید این رشته از مقادیر ورودی‌ـخروجی استفاده می‌شوند. این مطلب در برخی کاربردها مانند یک شمارنده که مقادیر و تعداد حالات مهم هستند، صادق نیست.

مثال: دیاگرام حالت زیر را ساده کنید.

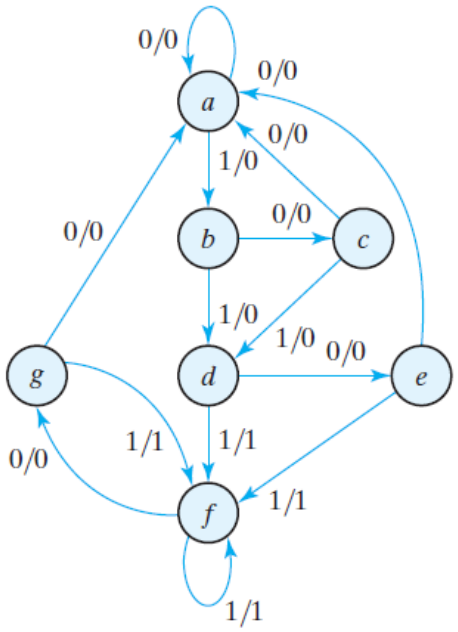


FIGURE 5.25
State diagram

حل:
ابتدا جدول حالت را از روی دیاگرام حالت به دست می‌آوریم:

Table 5.6
State Table

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

حال در جدول حالت، آن حالتهای فعلی را که حالتهای بعدی، مقدار ورودی، و مقدار خروجی کاملاً یکسان دارند پیدا کرده و یکی از این حالتهای فعلی را حذف می‌کنیم زیرا این حالتها با هم معادل و یکسان هستند. برای مثال، حالتهای فعلیِ e و g در جدول فوق چنین خاصیتی دارند؛ پس، آنها را حذف می‌کنیم:

Table 5.7
Reducing the State Table

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

همین کار را ادامه می‌دهیم تا جایی که نتوان حالتهای فعلیِ معادل با هم یافت. در مثال جاری، دو حالت فعلیِ d و f نیز با هم معادل هستند و می‌توان یکی از این دو را حذف کرد:

Table 5.8
Reduced State Table

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

حالا که دیگر نمی‌توان جدول حالت را ساده‌تر کرد، دیاگرام حالت ساده شده‌ی متناظر با جدول حالت ساده شده را می‌توان رسم کرد:

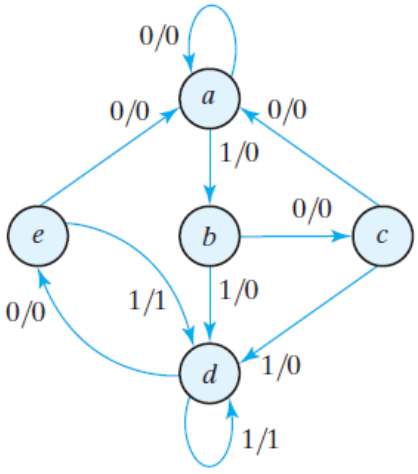


FIGURE 5.26
Reduced state diagram

تخصیص حالت

در طراحی مدارات ترتیبی، پس از رسم دیاگرام حالت باید به هر حالت آن، یک کد دودوییِ منحصربفرد نسبت داده شود. اگر یک دیاگرام حالت دارای m حالت باشد، باید یک کد n بیتی به هر حالت نسبت داد طوری که $2^n \geq m$ باشد. برای مثال با سه بیت می‌توان هشت حالت مختلف را کدگذاری کرد. دیاگرام حالتی که در شکل اخیر نمایش داده شد، دارای هفت حالت بوده و بنابراین حداقل یک کد سه بیتی نیاز داریم ولو این که یکی از کدها مورد استفاده قرار نگرفته باشد.

به شیوه‌های مختلفی می‌توان به حالتها کد دودویی تخصیص داد. سه شیوه‌ی ممکن از تخصیص کد در جدول زیر و برای دیاگرام حالت ساده شده‌ی اخیر نشان داده شده است. این سه شیوه عبارتند از: کد دودویی، کد گری، کد یک بارز.

Table 5.9
Three Possible Binary State Assignments

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

اگر برای مثال اخیر، از کدنویسی دودویی استفاده کنیم، جدول حالت ساده شده به صورت زیر درخواهد آمد:

Table 5.10
Reduced State Table with Binary Assignment 1

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

روال طراحی مدارات ترتیبی کلاک‌دار

مراحل هفت‌گانه‌ی طراحی یک مدار ترتیبی به صورت زیر است:

۱- رسم یک دیاگرام حالت با توجه به توضیحات و مشخصات مساله،

۲- کاهش تعداد حالات در صورت لزوم،

۳- تخصیص مقادیر دودویی به حالات موجود در دیاگرام حالت (ساده شده)،

۴- نوشتن جدول حالت،
۵- تعیین نوع فلیپ فلاپ مورد نظر در طراحی مدار،
۶- به دست آوردن معادلات ساده شده‌ی ورودیِ فلیپ فلاپ‌ها و نیز معادلات خروجیِ مدار،
۷- رسم دیاگرام منطقی مدار.

طراحی با استفاده از فلیپ فلاپ D

مثال: مداری طراحی کنید که سه (یا تعداد بیشتر) 1 متوالی موجود در رشته بیت وارده از طریق یک خط ورودی را شناسایی کند.

قدم اول: رسم دیاگرام حالت

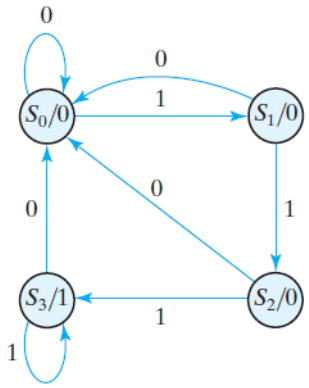


FIGURE 5.27
State diagram for sequence detector

قدم دوم: تخصیص کد و نوشتن جدول حالت

Table 5.11
State Table for Sequence Detector

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

دیاگرام حالت دارای چهار حالت است پس نیاز به دو فلیپ فلاپ داریم. در این مساله از DFF استفاده می‌کنیم و آنها را A و B می‌نامیم. مدار ما شامل یک ورودی و یک خروجی است.

برای به دست آوردن معادلات ورودی فلیپ فلاپ‌ها، باید در جدول حالت ستونهایی ویژه‌ی ورودی‌های فلیپ فلاپ‌ها بگنجانیم که در این ستونها مقادیر ورودی/های فلیپ‌فلاپ‌ها را برای این که از حالت فعلی مشخص به حالت بعدی مشخص برویم، باید تعیین کنیم. خوشبختانه، انجام این کار در مورد DFF ضرورتی ندارد زیرا در یک DFF به این نکته توجه می کنیم که معادله‌ی مشخصه به صورت زیر است:

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

یعنی حالت بعدیِ فلیپ فلاپِ D همان ورودی به فلیپ فلاپ است؛ به بیان دیگر، ورودی‌های D_A و D_B در واقع برابر با مقادیر حالت بعدی، $A(t+1)$ و $B(t+1)$ هستند (پس معادله‌ی توصیف کننده‌ی ورودیِ یک فلیپ فلاپِ D، همان معادله‌ی توصیف کننده‌ی حالت بعدی بوده) و می‌توانیم بنویسیم:

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

فقط فلیپ فلاپ D است که این ویژگی مهم را دارد که حالات بعدی همان مقدار وارد شده به ورودی فلیپ فلاپ هستند. برای بقیه‌ی فلیپ فلاپ‌ها به جدولی به نام «**جدول تحریک**[1]» نیاز داریم. این جدول به ما می‌گوید برای این که از یک حالت فعلی مشخص به حالت بعدی مشخصی منتقل شویم، ورودی‌های آن فلیپ فلاپ چه مقداری باید داشته باشند. در مثالهای بعدی که طراحی را مبتنی بر JKFF و TFF انجام می‌دهند، از جدول تحریک استفاده خواهیم کرد.

به کمک جدول کارنو می‌توانیم معادلات ساده شده‌ی ورودی فلیپ فلاپ‌ها و معادله‌ی خروجی مدار را به دست آوریم:

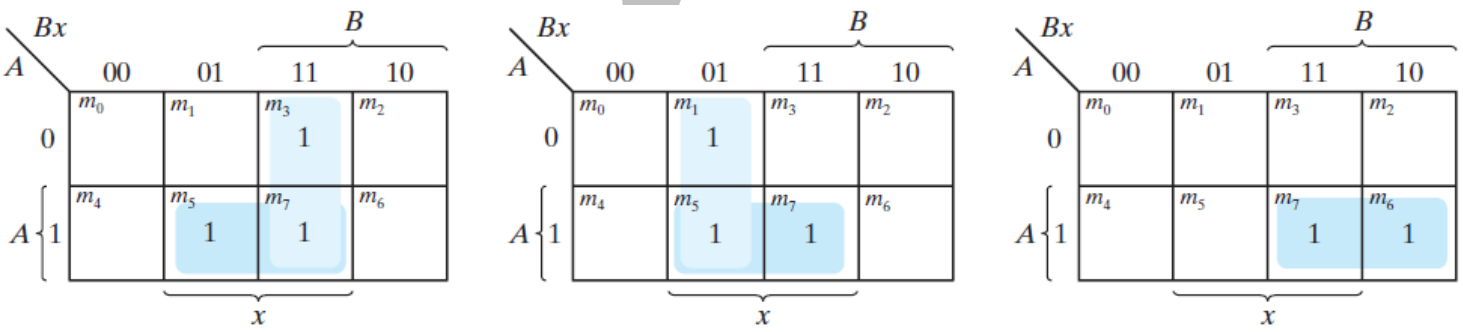


FIGURE 5.28
K-Maps for sequence detector

پس:

[1] Excitation table

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

حالا از روی معادلات ورودی فلیپ فلاپ‌ها و معادلات خروجی مدار، می‌توانیم دیاگرام منطقیِ مدار را رسم کنیم:

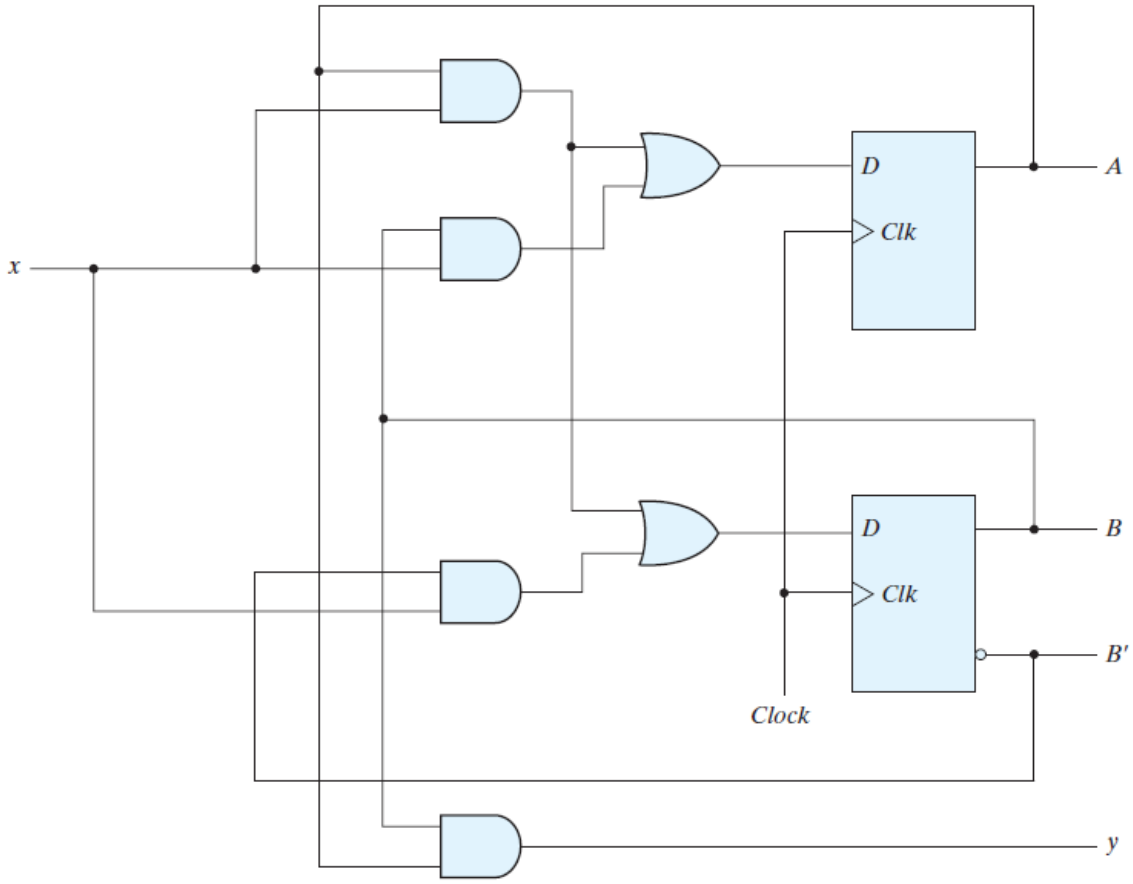


FIGURE 5.29
Logic diagram of a Moore-type sequence detector

جدول تحریک فلیپ فلاپ

جدول تحریک یک فلیپ فلاپ را از روی جدول مشخصه‌ی آن می‌توان به دست آورد. جداول تحریک دو نوع فلیپ فلاپ JK و T در زیر نشان داده شده است.

Table 5.12
Flip-Flop Excitation Tables

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

(a) *JK* Flip-Flop

Q(t)	Q(t + 1)	T
0	0	0
0	1	1
1	0	1
1	1	0

(b) *T* Flip-Flop

طراحی با استفاده از JKFF

می‌خواهیم مداری را مبتنی بر استفاده از JKFF طراحی کنیم که جدول حالت آن به صورت زیر باشد (ستونهای مربوط به ورودی‌های این فلیپ فلاپ را بعداً اضافه کرده‌ایم):

Table 5.13
State Table and JK Flip-Flop Inputs

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

جدول فوق دو حالت A و B دارد؛ بنابراین، مدار شامل دو JKFF است. ستونهای مربوط به ورودی‌های J و K هر کدام از فلیپ فلاپ‌ها را به کمک جدول تحریکِ JKFF نوشته‌ایم تا از روی آنها بتوانیم معادلات ورودی مدار را به دست آوریم. برای این کار به کمک جدول کارنو، معادلاتِ ساده شده‌ی ورودیِ دو فلیپ فلاپ را به صورت زیر به دست می‌آوریم:

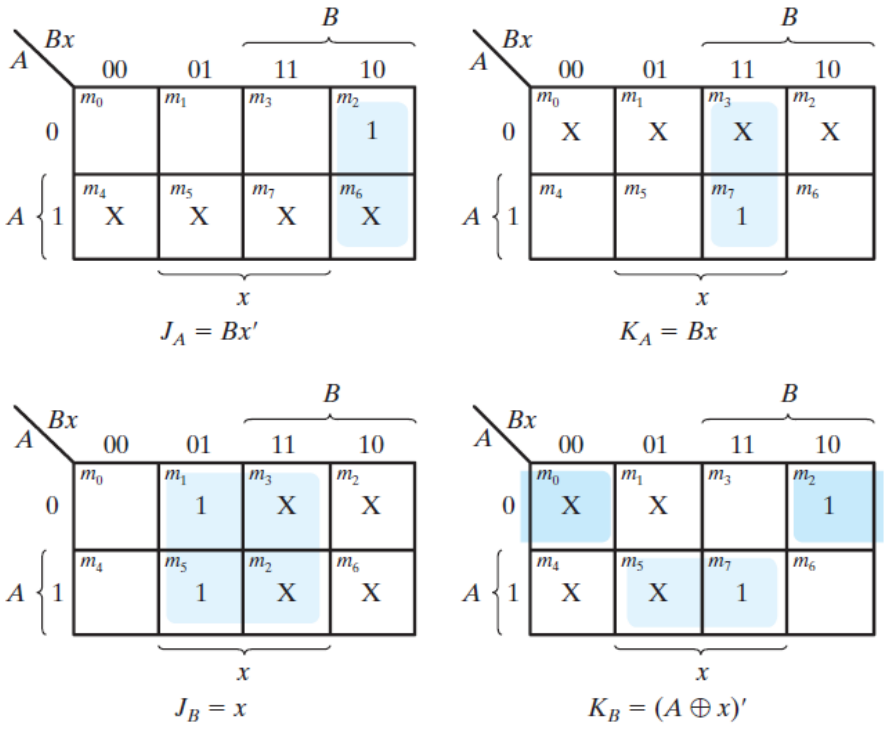


FIGURE 5.30
Maps for *J* and *K* input equations

به مزیت استفاده از JKFF در طراحی مدارات ترتیبی توجه کنید: با توجه به این که تعداد قابل توجهی حالات بی‌اهمیت در جدول وجود دارد، مدار ترکیبیِ ساده‌تری به دست آمده است (مهمترین استفاده‌ی حالات بی‌اهمیت، انجام ساده‌سازی بیشتر است).

حال با توجه به معادلات ورودیِ ساده شده‌ی به دست آمده، می‌توانیم دیاگرام منطقیِ مدار را به صورت شکل زیر ترسیم کنیم:

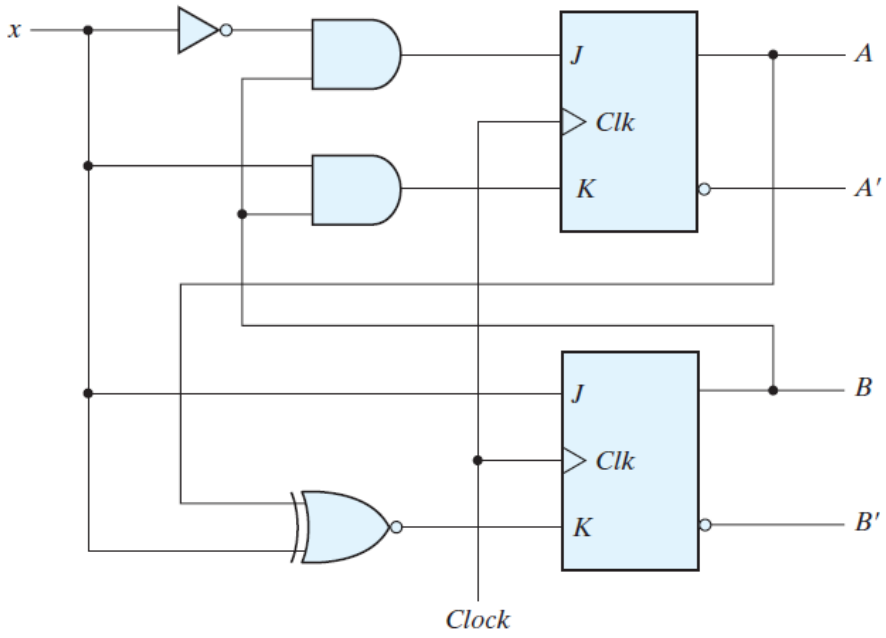


FIGURE 5.31
Logic diagram for sequential circuit with *JK* flip-flops

طراحی با استفاده از TFF

می‌خواهیم یک شمارنده‌ی دودویی سه بیتیِ افزایشی به کمک فلیپ فلاپ T طراحی کنیم.
دیاگرام حالت این شمارنده در شکل زیر نشان داده شده است. هر حالت در این دیاگرام با سه بیت نمایش داده شده است؛ پس، مدار شامل تعداد سه TFF است.

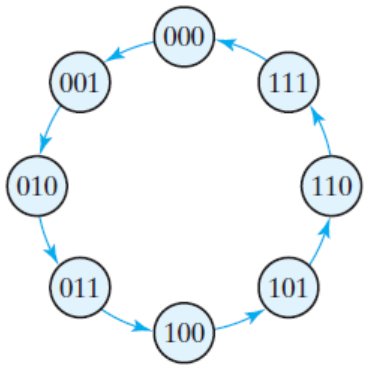


FIGURE 5.32
State diagram of three-bit binary counter

ملاحظه می‌کنید که این مدار فاقد ورودی و خروجی است؛ در واقع، تنها ورودی همان کلاک مدار است که در دیاگرام حالت هیچ گاه نیازی به نمایش آن نیست. از روی این دیاگرام حالت، جدول حالت را به صورت زیر به دست می‌آوریم:

Table 5.14
State Table for Three-Bit Counter

Present State			Next State			Flip-Flop Inputs		
A_2	A_1	A_0	A_2	A_1	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

در جدول فوق، ستونهایی برای ورودی‌های سه فلیپ فلاپ اضافه شده‌اند که مقادیر این ستونها از روی جدول تحریکِ TFF به دست آمده است. حال، به کمک تعدادی جدول کارنو، معادلات ساده‌شده‌ی ورودی‌های فلیپ فلاپ‌ها را به دست می‌آوریم:

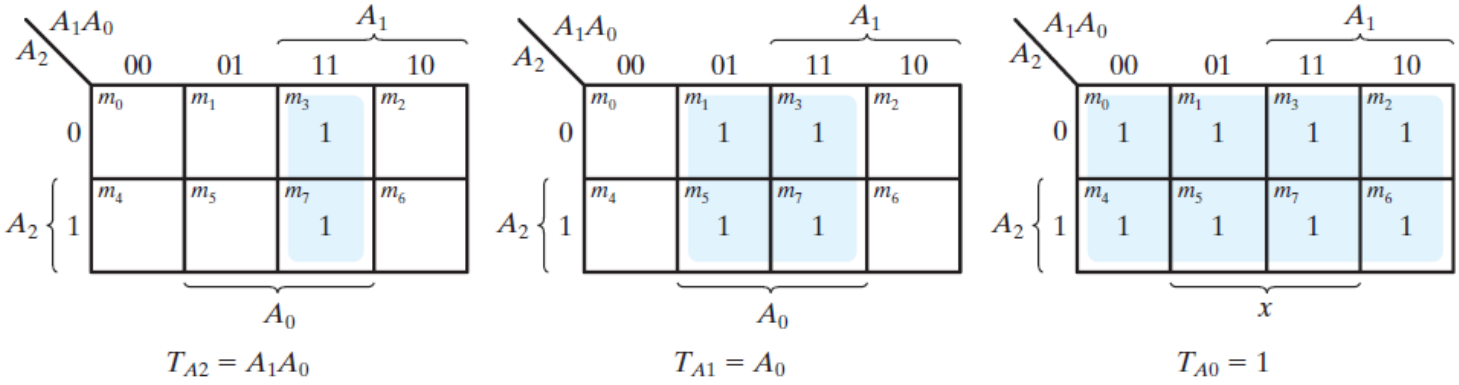


FIGURE 5.33
Maps for three-bit binary counter

در آخرین قدم، دیاگرام منطقیِ مدار را به کمک معادلات ورودی به دست آمده، رسم می‌کنیم:

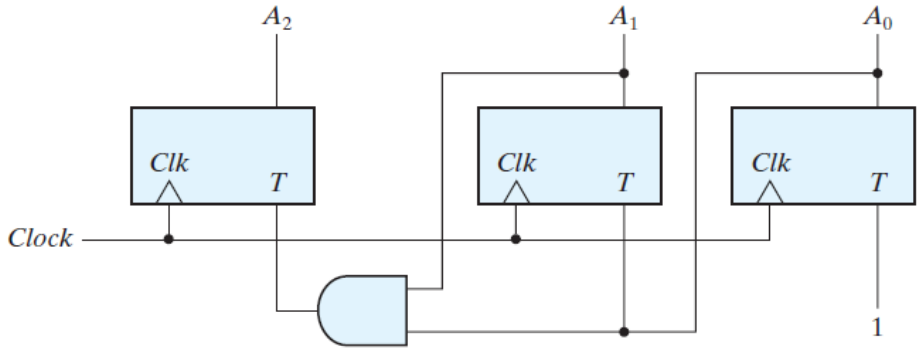


FIGURE 5.34
Logic diagram of three-bit binary counter

براي سلامتي رهبر انقلاب و تعجيل در ظهور حضرت ولي عصر (عج) صلوات