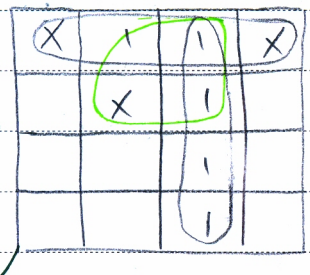
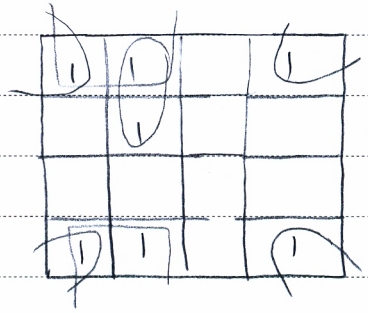
Year. Month. Date. ( )

Example : تابع $F = \Sigma(1,3,7,11,15)$ به همراه حالت‌های بی اهمیت $D = \Sigma(0,2,5)$ در نظر بگیرید. ساده‌ترین تابعی که بدست می‌آید را به کمک جدول کارنو بنویسید.

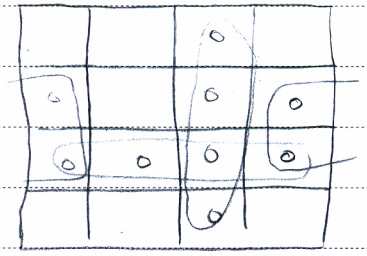


Example : تابع منطقی زیر را : الف : به صورت جمع حاصلضرب ها ب. به صورت ضرب حاصلجمع ها ساده کنید.

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



$F = \bar{B}\bar{C} + \bar{B}\bar{D} + \bar{A}CD$



$\bar{F} = B\bar{D} + CD + AB$

$\Downarrow$

$F = (\bar{B}+D)(\bar{C}+\bar{D})(\bar{A}+\bar{B})$

برای پیاده‌کردن ساده‌ترین حالت یک تابع به کمک مالتی‌پلکسر، صفر ها را دسته‌بندی می‌کنیم و در انتها دسته‌های بوجود آمده اولاً به صورت حاصلجمعی است ثانیاً هرکدام از متغیرهای ورودی که در دسته‌ها باشند آنرا می‌نویسیم، مثلاً اگر $\bar{A}$ در دسته باشد A می‌نویسیم

پیاده سازی مدار منطقی به کمک NAND :

با توجه به این که در صفت پیاده سازی عملی مدارات منطقی به کمک NAND و NOR انجام می‌شود. به دنبال راهی هستیم تا تمام مدارات خود را به کمک یکی از این دو گیت پیاده سازی نماییم.

ابتدا باید ثابت کنیم یک گیت NAND با یک گیت Invert-OR معادل است، آن گاه می‌توان تمام مدارات خود را با سه مرحله زیر تبدیل به گیت‌های NAND کنید.

تمرین: با کمک جدول صحت 2x2 این اثبات را انجام دهید.

x	y	f_1	f_2	f_3
0	0	1	0	1
0	1	1	1	1
1	0	1	1	1
0	1	0	1	0

$f_1 = NAND$

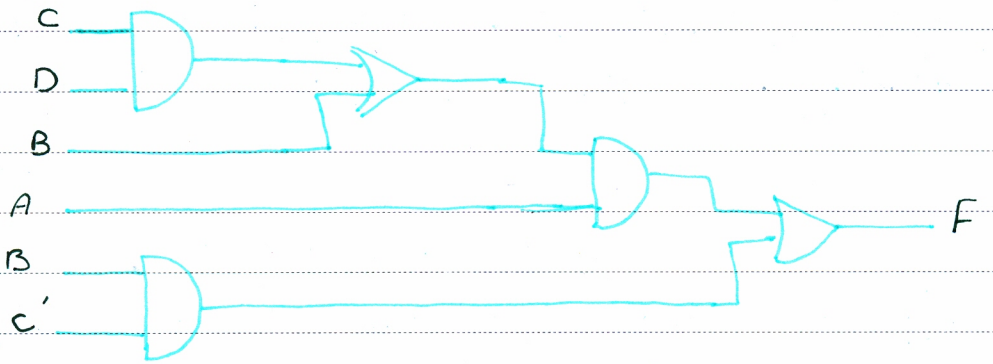
$f_2 = OR$

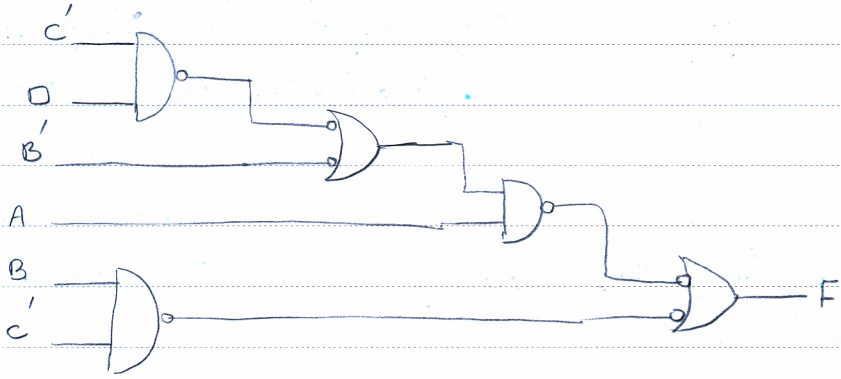
$f_3 = Invert\text{-}OR$

$\Rightarrow f_1 = f_3 \Rightarrow NAND \equiv Invert.OR$

1. هرجا گیت AND دیدیم به جای آن NAND می‌گذاریم.
2. تمام OR‌ها به Invert-OR تبدیل می‌شوند و با توجه به تساوی Invert-OR و NAND آنها را با هم جایگزین می‌کنیم.
3. چک می‌کنیم که NOT‌ها کل مدار را دچار اشکال نکنند. اگر NOT اضافه‌ای وجود داشت با یک NOT دیگر که همان NAND تک ورودی می‌شود اصلاح می‌کنیم.

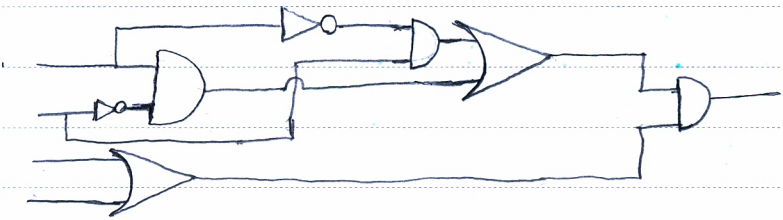
Example: می‌خواهیم مدار زیر را با NAND پیاده سازی کنیم.

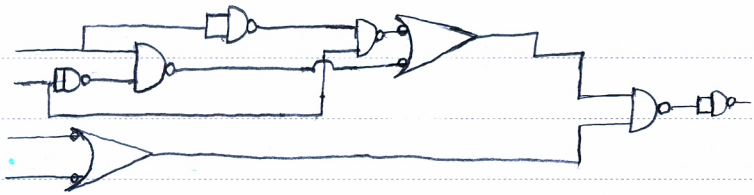


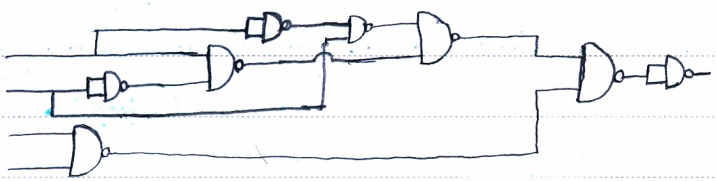


مدار زیر را به گیت NAND پیاده سازی کنید. Example:

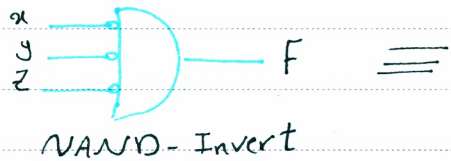
$F = (AB' + A'B)(C + D')$

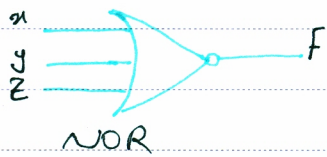






پیاده سازی توابع منطقی به گیت NOR :





این جا هم هرجا گیت OR بود یا NOR و AND بود با Invert-And جایگزین می‌کنیم. مجدداً تمام NOT های ایجاد شده را بررسی می‌کنیم.

توابع XOR :

قبلاً دیدیم که XOR و XNOR برای دو ورودی به شکل زیر است :

$$(x \oplus y) = xy' + x'y$$
$$(x \oplus y)' = xy + x'y'$$

حال اگر $x \oplus y \oplus z$ را بخواهیم، خواهیم داشت :

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

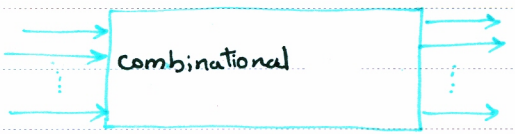
XOR ≡

x \ yz	00	01	11	10
0		1		1
1	1		1	

$$\Rightarrow (x \oplus y \oplus z) = x'yz' + xy'z' + \big((x'+y)z \times (x+y')z\big)$$

فصل چهارم : مدارات ترکیبی — Combinational circuits

اکنون که با جدول صحت، Gates و توابع بولین آشنا شدیم، می‌توانیم مدارهای ترکیبی را معرفی کنیم. منظور از مدارات ترکیبی، مدارهایی هستند که خروجی در هر لحظه به ورودی در همان لحظه بستگی دارد. (به عبارت دیگر حافظه ندارد)



طراحی یک مدار ترکیبی

قبلاً در آنالیز یک مدار داشتیم برای آن که خروجی را بدست آوریم، به تک تک خروجی هایی که از ورودی ها به دست آمده است، Label می دادیم. سپس هرکدام از خروجی ها را به کمک جدول صحت تعیین می کردیم و در مرحله آخر خروجی نهایی به کمک خروجی ها و باز هم جدول صحت به دست می آمد.

منظور از آنالیز آن است که خروجی به ازای هر ورودی چقدر می شود.

رجوع شود به صفحه Figure 4.2 — 128

Example: خروجی های F_1 و F_2 را با Label گذاری تعیین کنید.

رجوع شود به صفحه table 4.1 — 129

A	B	C	F_1	F_2	T_1	T_2	T_3

پروسه طراحی: باید مراحل زیر را به ترتیب انجام دهیم:

1. تعیین تعداد ورودی: از روی صورت مسئله تعداد ورودی ها و خروجی ها را حدس می زنیم.
2. رابطه بین ورودی و خروجی را به کمک جدول صحت به دست آوریم.
3. تابع بولین ساده شده را به کمک جدول کارنو بنویسیم.
4. مدار را رسم کنید.

Example: می خواهیم یک کد BCD را به کد excess-3 به کمک یک مدار تبدیل کنید.

منظور از BCD ایجاد اعداد 0 تا 9 به کمک اعداد باینری است.

رجوع شود به صفحه table 4.2 — 131

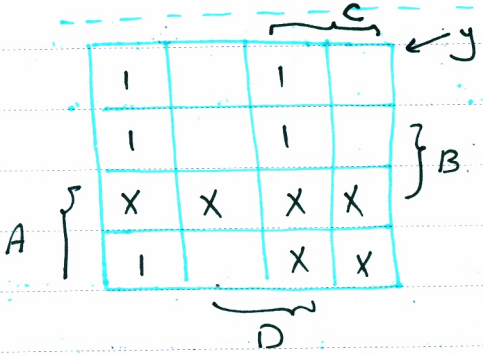
چون چهار خروجی داریم به چهار جدول کارنو نیاز احتیاج داریم. یک تک خروجی ها به صورت تابعی از ورودی ها به دست می آیند و بعد به راحتی مدار را رسم می کنیم.

- برنامه‌ای بنویسید که BCD را به کد excess 3 تبدیل کند.

حل سه متغیر

A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	1	1	0	1
1	0	1	1	1	1	1	0
1	1	0	0	1	1	1	1
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

$z = \bar{D}$



$y = \bar{C}\bar{D} + CD = CD + (C+D)'$

Figure 4.3 جدول شود

$x = B'C + B'D + BC'D' = B(C+D) + B(C+D)'$

$w = A + BC + BD = A + B(C+D)$

وجود $(C+D)$ با توجه به این که در سه خروجی دیده می‌شود، به شکل موارد کوتاه‌تر خواهد بود

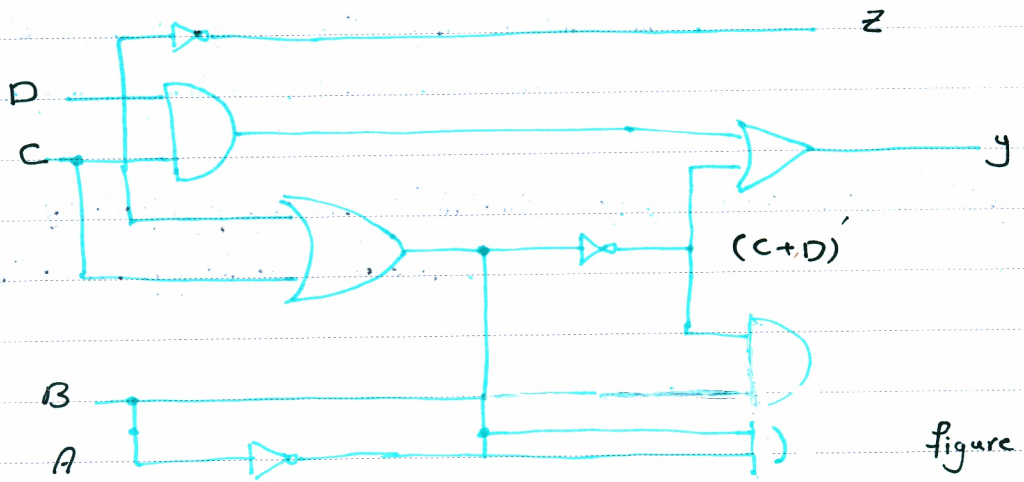
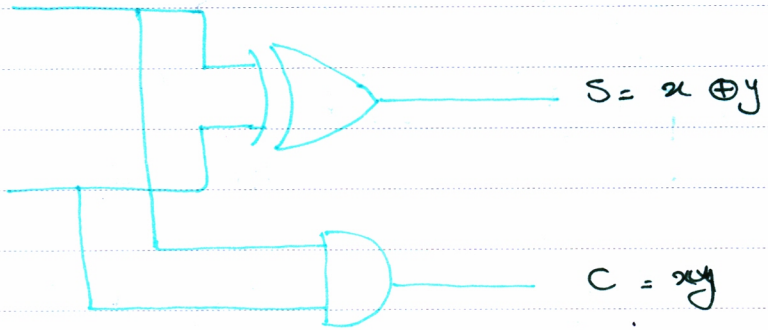


Figure 4.4

مدارهای نیم جمع‌کننده و تمام جمع‌کننده

- مدارهای نیم جمع (Half Adder): این مدارات دو ورودی x و y را گرفته و با هم جمع می‌کنند. یک حاصل جمع (Sum) و یک بیت نقلی (carry) به وجود می‌آورند.

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



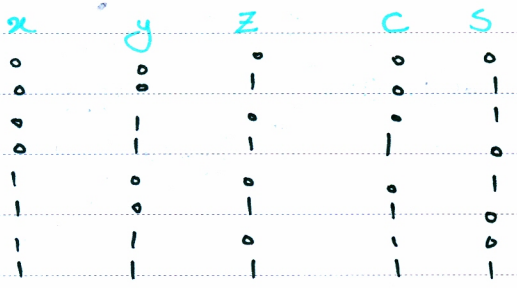
- مدارهای تمام جمع‌کننده (Full Adder): زمانی که می‌خواهیم دو عدد n بیتی را با هم جمع کنیم همواره یک حاصل جمع داریم و یک بیت نقلی مثلاً:

C_1 C_0 a_1 a_0

b_1 b_0

c_1 S_1 S_0

یک Full Adder یک مدار ترکیبی است که برای جمع سه بیت استفاده می‌شود. این مدار سه ورودی (دو تای آنها بیت‌های دو عدد جمع شونده و دیگری بیت نقلی از طبقه قبل است) و دو خروجی (اولی حاصل جمع و دومی بیت نقلی) تشکیل شده است. حاصل جمع این سه عدد می‌تواند 0 ـ 1 ـ 2 ـ 3 باشد که برای ساخت 2 یا سه به 2 بیت نیاز است: ⟸ وجود دو خروجی الزامی است.



رجوع شود به Figure 4.7 :

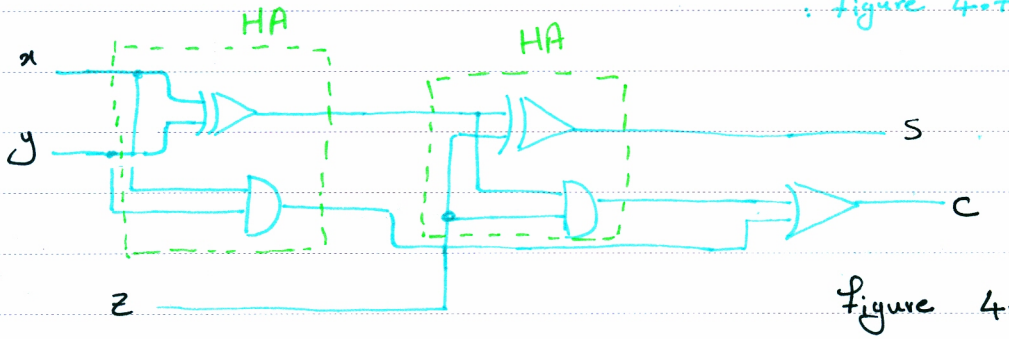


Figure 4.8

Example : یک مدار جمع کننده 4 بیتی توسط F.A بسازید

با توجه به این که مرجع بیت های متناظر عدد باهم جمع می شوند ورودی هریک از F.A ها باید متناظر با دو عدد باشد و ورودی سوم آنها بیت نقلی است که حاصل جمع طبقه قبل برای آن ساخته است.

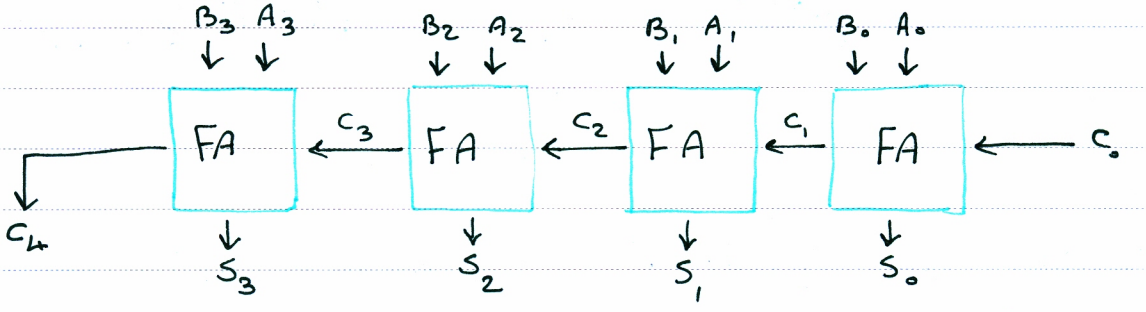


Figure 4.9

Example :

$$
\begin{array}{cccccc}
 & 1 & 0 & 0 & 1 & 0 \\
 & C_4 & C_3 & C_2 & C_1 & C_0 \\
A & & 1 & 1 & 0 & 1 \quad 13 \\
B & + & 1 & 0 & 0 & 1 \quad 9 \\
\hline
 & 1 & 0 & 1 & 1 & 0 \quad 22 \\
 & & S_2 & S_1 & S_0 &
\end{array}
$$

می خواهیم به کمک خروجی های نشان داده شده‌ی C S G P به ازای بیست آوریم و مداری در سطح با carry ها طراحی کنیم.

$P_i = A_i \oplus B_i$ $S_i = P_i \oplus C_i$

$G_i = A_i B_i$ $*C_{i+1} = G_i + P_i C_i$ مدار صفحه‌ی 139

برای اینکه carry ها را از C_0 تا C_3 یکی یکی بسازیم به صورت زیر عمل می کنیم:

C_0 = input carry

$* \longrightarrow i=0 \longrightarrow C_1 = G_0 + P_0 C_0$ figure 4.11

$i=1 \longrightarrow C_2 = G_1 + P_1 C_1 = G_1 + P_1(G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0$

$i=2 \longrightarrow C_3 = G_2 + P_2 C_2 = G_2 + P_2(G_1 + P_1 G_0 + P_1 P_0 C_3) = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_3$

قبلاً دیدیم که برای تفریق دو عدد A, B با بیتی A را با متمم 2 عدد B جمع کنیم.

یادآوری: متمم از متمم 2 یک عدد متمم 1 آن به علاوه یک است و متمم از متمم یک عدد toggle بودن همه بیت ها است.

طراحی یک مدار جمع کننده ـ تفریق کننده چهار بیتی:

در مدار شکل زیر figure 4.13 اگر M=0 باشد مدار جمع کننده و اگر M=1 مدار با توجه به توضیحات بالا تفریق کننده است.

تمرین: خروجی که از $C_3 \oplus C_4$ بدست آمده است برای مشخص کردن وضعیت over flow در این مدار می باشد. مدار را تحلیل کنید. page 143

جمع کننده Decimal:

برای آن که دو عدد در مبنای ده را با هم جمع کنیم بیشترین حاصل جمعی که برای 2 بیت ایجاد می شود 9+9+1 است.

قصد داریم که به کمک دو جمع کننده باینری یک جمع کننده Decimal چهار بیتی بسازیم

جدول صحت آنرا به شکل زیر خواهیم داشت :

Table 4.05

در ورودی به کمک اعداد باینری از صفر تا دوازده ایجاد می کنیم بدین ترتیب به 5 بیت احتیاج داریم . در خروجی از صفر تا 9 ما شش ورودی ایجاد می شود ولی برای 10 تا 19 - صفر ده تا نه دوازده جداگانه و دهگان 10 تا 19 (یک) در یک بیت مجزا در نظر گرفته می شود .

ابتدا خروجی را بدست می آوریم : $C = k + z_8 z_4 + z_8 z_2$

بدین ترتیب مدار را به شکل زیر طراحی می کنیم page 146 - figure 4.14

با دقت در جدول صحت می بینیم که c برای ایجاد دهگان در خروجی استفاده می شود . یعنی زمانی که $C = 0$ است . یا صفر تا نه در خروجی طرف هستیم و زمانی که $C = 1$ است با 10 تا 19 مواجه هستیم .

مدار را به شکل figure 4.14 در نظر بگیرید .

Example : فرض کنید a_0 به عنوان کم ارزش ترین بیت عدد A برابر 4 و b_0 برابر کم ارزش ترین بیت عدد B برابر 7 باشد . می خواهیم به کمک این مدار جزخروجی S_1, S_2, S_4, S_8, C را بدست آوریم .

$4 + 7 = 11$ ← 1 001

$11 + 6 = 17$ → 10001

می دانیم که عدد 11 از یک در یکان و یک در دهگان تشکیل شده است . یک در یکان باید به کمک $Z_1 Z_2 Z_4 Z_8$ ساخته شود ولی می دانیم که 11 در باینری برابر 1011 است . برای اینکه 1011 به 0001 تبدیل شود (چون که یکان یک دوده در (1) باید که عدد 11 با 6 جمع شود .

$$\begin{array}{r} 1011 \\ +\ 0110 \\ \hline 10001 \end{array}$$