Subject
Date

مدار منطقی

ارزیابی:
میان ترم ۲ تا
پایان ترم
تمرین ۸ سری
پروژه نرم افزاری (اجباری) شبیه سازی
پروژه عملی (اختیاری)

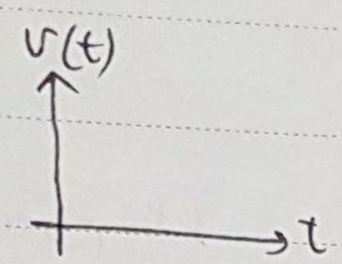
ایمیل: Zandi@kntu.ac.ir

منابع: (نام نویسنده ها) Nielson / wakerly / mano

Subject
Date

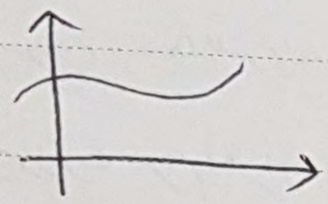
سیگنال هر چیزی که حاوی اطلاعات باشد یک سیگنال است مثل یک صدا، یک تصویر، یک حس درد

سیگنال های الکتریکی سیگنالی که به شکل الکتریکی اطلاعات را قابل پردازش می نماید

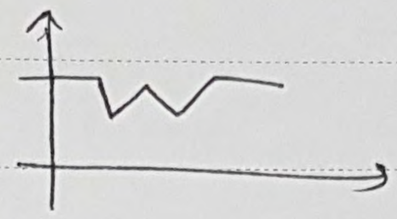


در این درس سیگنال های الکتریکی کاربرد دارند که در طول زمان کار می کنند

این سیگنال ولتاژ الکتریکی ست که تابعی از زمان است



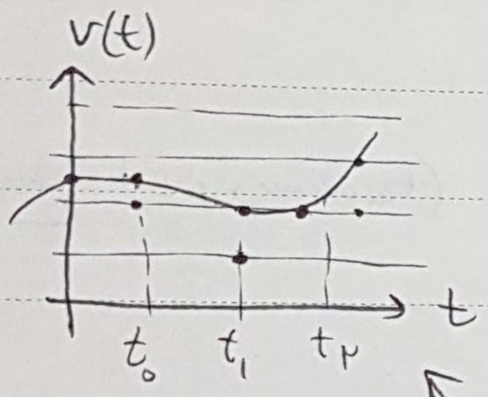
آنالوگ (پیوسته)



دیجیتال (گسسته)

دیجیتال: گسسته بودن < در زمان / در مقدار

Resolution تفکیک پذیری - ریز نمایی



ADC: مدارهایی که سیگنال ها را به صورت آنالوگ دریافت کرده و به دیجیتال تبدیل می کنند

sample rate نمونه برداری در طول زمان

پیوستگی چه در زمان چه در مقدار باشد، مقدارهای دیجیتال در بازه های زمانی گزارش می شود

یعنی ما مقدار گسسته را گزارش می دهیم نه پیوسته را (یعنی مقداری کمتر از مقدار پیوسته (براکتی)) ← قرارداد: بزرگترین عدد صحیح کوچکتر از آن عدد

در گسسته سازی سیگنال ها هم در بعد زمان و هم در بعد مکان مقادیر گسسته ای انتخاب می شوند و سیگنال آنالوگ با توجه به آنها گسسته خواهد شد

به صورت عام مدارها و سیگنال های پیوسته سریع ترند ولی دیجیتال ها مزیت هایی برای ما دارند (دیجیتال مثل پله / آنالوگ مثل سطح شیب دار)

چرا به گسسته سازی علاقه مندیم؟

۱) حافظه: ذخیره سازی راحتتر اطلاعات (چون اطلاعات دونه دونه است)

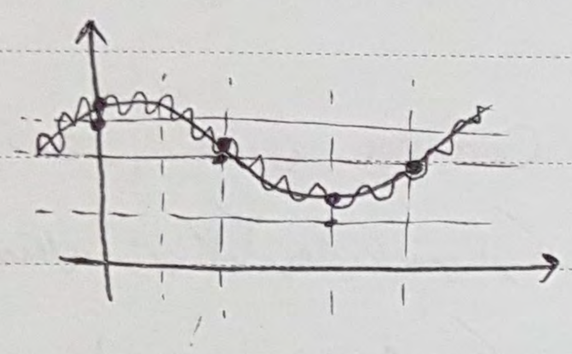
Noise (فارسی شده: نوفه) به هر چیزی غیر از سیگنال مورد نظر (سیگنال اصلی) که در پردازش باعث مزاحمت می شود، گویند که حتی ممکن است اطلاعاتی نداشته باشد

Subject
Date

بدن انسان مثل آنتن عمل می‌کند چون حدود ۱۵۰ ولت از برق شهری، ولتاژ دریافت می‌کند درصورتی که مدارهای ساده حدود ۵ ولت نیاز دارند پس بدن انسان یک جور نویزه که باید حذف شود تا سیگنال بهتر دریافت می‌شود

وقتی سرعت بالا می‌رود حذف نویز خیلی راحتتر است در مدارهای دیجیتال ولی در مدارهای آنالوگ سخت‌تر است

۲) قابلیت حذف (کاهش) نویز



نویز کاملاً قابل حذف نیست ولی می‌شود تاثیر آنرا کاهش داد

به طور مثال در سیگنال مقابل سه زمان اول قابل حذف هستند (حذف نویز)

ولی چهارمی در صورت اندکی تغییر مقدار متفاوتی بدست می‌آید و نمی‌شود کاملاً نویز را حذف کرد

۳) استفاده از منطق شناخته شده در انجام پردازش‌های مورد نیاز ← راحتی ساخت و اجرای الگوریتم‌ها

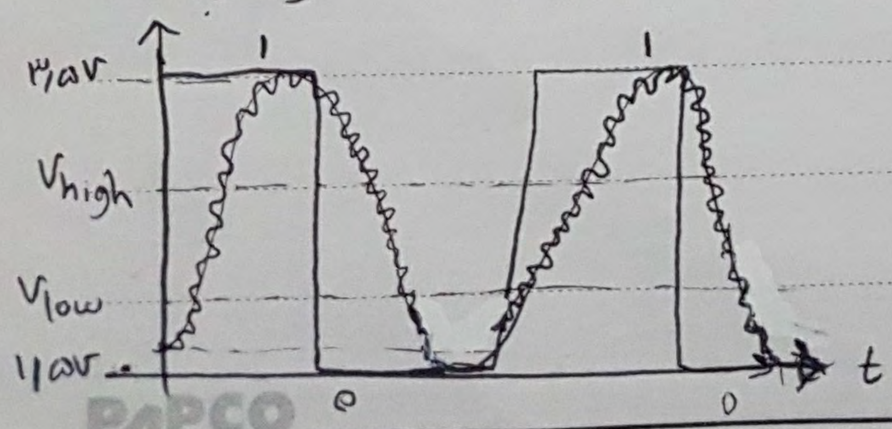
- هردو مقدار آنالوگ و دیجیتال نویز دارند ولی برای آنالوگ خیلی بیشتر است
- ۰ و ۱ مقدار ولتاژ نیست بلکه درست و غلط یا هست و نیست را نشان میدهد و با مقدار صفر و یک ولتاژ نباید اشتباه گرفت
- در اصل در مدارهای دیجیتال منطق های ۲ حالته ۰ و ۱ کاربرد دارد
- برای بهتر شدن کار دو گسستگی زمان و مقدار ادغام می‌شوند
- سیگنال آنالوگ را تا حد امکان نویزش رو می‌گیریم بعد تبدیل به سیگنال پیوسته می‌کنیم و بعد آن را تبدیل به سیستم دودویی می‌کنیم یعنی

* هر کدام از مقادیر استخراج شده توسط اعداد چند بیتی با منطق دودویی (دو حالته) بیان می‌شود (ما با دودویی کار می‌کنیم و قبل آن کاربچه‌های برقه)

* منطق اساسی مورد نیاز که هم الگوریتم‌های مهمی در آن معرفی شده و هم در مدارهای پردازنده اجرای آن بسیار ساده‌تر است، دودویی می‌باشد

- هر سیگنالی دریافت کردیم اگر زیر ۱٫۵V بود منطق آن صفر است و اگر بالای ۳٫۵V بود منطق آن یک است
- حذف نویز قوی دارد

* در عمل و در مدارهای دیجیتال موجود می‌توان با طراحی سطوح ولتاژ معینی مشابه روبه‌رو، تا حد زیادی حذف نویز و پردازش قوی‌تری (قابلیت اطمینان بالاتر) ایجاد نمود



- ram ecc رم‌های سرور نویز خطاها رو کمتر می‌کنند و اطمینان بیشتری دارند

PAPCO

Subject
Date

سه دسته از سیگنال های دیجیتال در عمل تعریف می شوند

۱- عددی

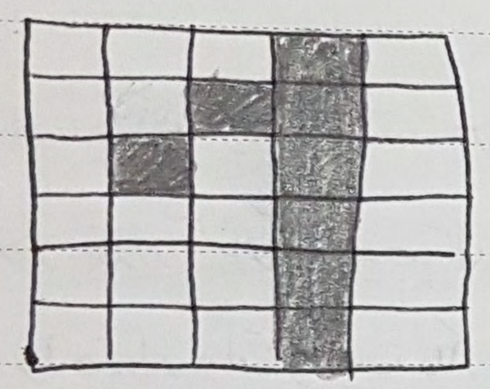
۲- نمایشی ← کارکتر ذخیره می شوند و می توند کارکتر عدد باشه

۳- دستورات

چطوری دسته های بالا را به کامپیوتر بفهمانیم؟ از طریق کدگذاری

مفهوم کدگذاری Coding قرارداد زبان ماشین

مثال: برای کارکترهای نمایشی از کد ASCII استفاده می شود / برای نمایش شکل کارکتر از فونت استفاده می شود که باز هم برای استفاده از فونت از همان صفر و یک استفاده می کنیم (روشن یا خاموش بودن)



عدد 1

عملیات های مفروض عددی:

پیاده سازی با کد دودویی X X

کد دهدهی (Decimal) ||||||||

عملیات جمع

$(11010)_2 = 2+8+16 = 26$

$$\begin{array}{r|l} 52 & 2 \\ \hline 26 & 2 \\ \hline 13 & 2 \\ \hline 6 & 2 \\ \hline 3 & 2 \\ \hline 1 & \end{array}$$

0, 0, 1, 0, 1

$\Rightarrow 52 = (110100)_2$

افراز کردن $52 = 32+16+4 = (110100)_2$

ضرب در عدد 2 $2 \times (10111)_2 = (101110)_2$

(شیفت به چپ و اضافه کردن صفر)

رقم نقلی carry digit

$$\begin{array}{r} 11010 \\ +\ 110100 \\ \hline (1001110)_2 \end{array}$$

$(1001110)_2 = 64+8+4+2 = 78$

$26+52 = 78$ ✓

PAPCO

Subject
Date

تفریق کردن

borrow digit رقم نقلی

52 - 26 = 26

$$
\begin{array}{cccccc}
0 & 1+1 & 0 & 1+1 & 0 & 1+1 \\
\cancel{1} & \cancel{1} & \cancel{0} & \cancel{1} & \cancel{0} & 0 \\
- & 1 & 1 & 0 & 1 & 0 \\
\hline
0 & 1 & 1 & 0 & 1 & 0
\end{array}
$$

$$
\begin{array}{r}
11010 \\
-\ 110100 \\
\hline
\end{array}
$$

با روش قبل بی نتیجه (دلیل: منفی بودن پاسخ)

نمایش و عملیات بر روی اعداد منفی

* در نظر گرفتن یک بیت اضافی برای نشان دادن علامت

استاندارد: بایت: بسته‌ای از ۸ بیت / بیت آخر (هشتم - بیت a_7) می‌تواند به عنوان بیت علامت انتخاب شود

رقم صفر در بیت a_7 به معنای علامت مثبت

و به رقم یک در بیت a_7 به معنای علامت منفی می‌باشد

با توجه به تعریف فوق بازه‌ی اعداد به صورت زیر می‌باشد

-127 ..... +127

در این شرایط 255 حالت مختلف وجود دارد چون $+0 = -0$

0 ... 256 بدون بیت علامت (قبلی)

راه حل اول: مبنای متمم 2

$(2^8 - 1) - 26$

$$
\begin{array}{rl}
2^8 - 1 = & (1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)_2 \\
26 = & (0\ 0\ 0\ 1\ 1\ 0\ 1\ 0)_2 \\
\hline
 & 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1
\end{array}
$$

در مثال فوق انجام عملیات تفریق از عدد $(2^8 - 1)$ متناظر با وارونه کردن ارقام عدد مفروق می‌باشد (یعنی صفر ← یک، یک ← صفر)

Subject
Date

راه حل دوم: مبنای متمم ۲ منهای یک

در این روش و در بیان اعداد در این مبنا "منهای یک" در خود تعریف وجود دارد و برای تبدیل تنها کافی ست عدد اصلی را وارونه کنیم

در این مبنا برای بیان اعداد منفی مشابه قبل از بیت علامت استفاده می‌کنیم ولی اگر عدد اصلی مثبت باشد تنها کافی ست بیت علامت صفر به آن اضافه شود و اگر عدد اصلی منفی است با بستن غیر از اضافه نمودن بیت علامت یک ارقام آن نیز وارونه گردند

[ ] : متمم ۲ منهای یک

$$2^n + A - B = A + [B]_{2^-} + 1 = A + (2^n - 1 - B) + 1 = 2^n - 1 + (A - B) + 1 = 2^n + (A - B)$$

$$\underbrace{2^n}_{\text{در نظر نمی‌گیریم (حذف می‌شود)}} + A - B = A + [B]_{2^-} + 1$$

$$2^n = (\qquad\qquad)_2$$ n+1 رقمی است در مبنای 2

مثال: تفریق دو عدد A - B با شرایط A > B

52 - 26 =

در این عملیات در واقع می‌خواهیم دو عدد A و B- را با یکدیگر جمع کنیم پس عدد مثبت یک بدون تغییر باقی می‌ماند و عدد B به صورت گفته شده تبدیل می‌نماییم

بیت علامت

A 0 0 1 1 0 1 0 0

-B 1 1 1 0 0 1 0 1 ← وارونه 0011010

1 0 0 0 1 1 0 0 1

→ 1

0 0 0 1 1 0 1 0

تو رو بیت ۹ ام هر چه بود (1 یا 0) با کل عدد جمع می‌کنیم

بیت های علامت در عملیات تفریق وارد می‌شوند

مثال: 26-52 :

* عدد بدست آمده متمم 2 منهای یک است چون منفی ست

* در پیاده سازی روش پس از جمع A و B- بیت نهم احتمالی را با حاصل جمع قبل جداد جمع می‌نماییم

+26 0 0 0 1 1 0 1 0

-52 + 1 1 0 0 1 0 1 1

0 1 1 1 0 0 1 0 1

→ 0

-26 1 1 1 0 0 1 0 1

بیت علامت — متمم 26

PAPCO

Subject
Date

MSB Most Significant Bit

LSB least Significant Bit

ضرب دو عدد

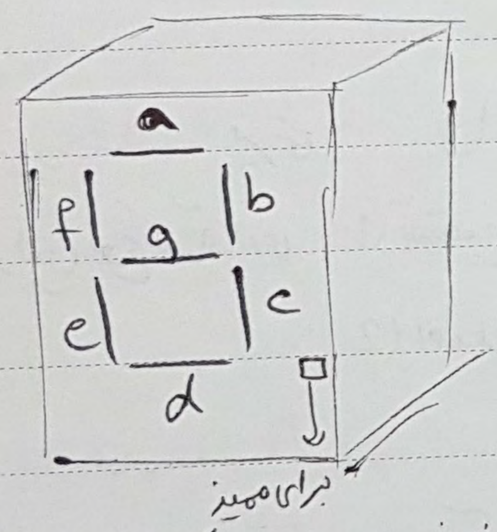
$$
\begin{array}{r}
26 \\
\times\ 5 \\
\hline
\end{array}
\qquad
\begin{array}{r}
11010 \\
101 \\
\hline
11010 \\
000000 \\
1101000 \\
\hline
10000010
\end{array}
$$

تمرین: عملیات تقسیم به صورت تفریق و جابه‌جایی را نشان دهید

تمرین: اگر عدد m بیتی در عدد n بیتی ضرب شود حاصل نهایتاً چند بیتی است؟ اثبات می‌شود m+n

$$(2^m-1)(2^n-1) = 2^{m+n} - 2^m - 2^n + 1 < 2^{m+n} - 1$$

کد BCD Binary Coded Decimal



هدف: نمایش اعداد دهدهی که ذات آنها دودویی می‌باشد 7segment display

برای نمایش ارقام 0 ... 9 دهدهی بایستی به مدار کنترلی این نمایشگرها 4 بیت نیاز داریم

برای مثال اگر بخواهیم عدد 908 دهدهی را توسط مدار دیجیتال بر روی سر نمایشگر هفت قسمتی نمایش دهیم بایستی که BCD زیر توسط مدار دیجیتال به سیستم‌های کنترلی داده شود

$$(1001\ \ 0000\ \ 1000)_{BCD}$$

مبنای هگز (16)

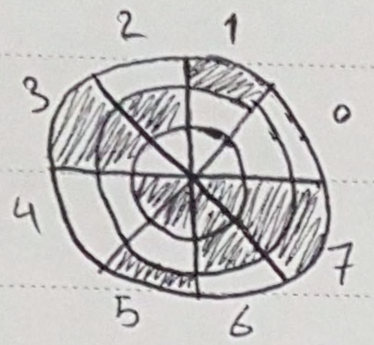
$$A = (1100\ \ 1010)_2 = (\overset{12}{C}\ \overset{10}{A})_{Hex}$$

هدف از این مبنای نمایش خلاصه‌تر اطلاعات دیجیتال می‌باشد

می‌توان با 7segment اعداد A و b و C و را هم نمایش داد که به آن دیگر BCD گفته نمی‌شود

Subject
Date

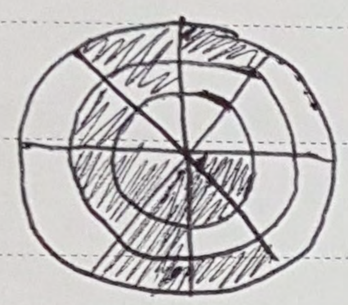
کدگری (Grey)



کد دودویی ساده

مشکل این کد اینه که ممکنه خطا ایجاد کنه و همزمان 2 بایت تغییر کند

* در این کدگذاری هدف ایجاد اختلاف تک بیتی بین هر دو کد مجاور می باشد
* در مدار تشخیص زاویه چرخش حضور کدگری لحاظ شده است که وابسته به ترتیب خاص آن (غیر دودویی) زاویه تشخیص داده می شود



کدگری

000
001
011
010
110
111
101
100

اگر تعداد بیت ها زیاد شود ممکن است ترتیب عوض شود ولی هدف و ویژگی ثابت است

کد تشخیص خطا

پرتی parity

هدف از این نوع کدگذاری 1) تشخیص خطای احتمالی

2) اصلاح آن در صورت امکان می باشد

در ابتدا می خواهیم تشخیص دهیم آیا در بیتی از اطلاعات خطایی رخ داده است یا نه؟

فرض: احتمال خطا در دو بیت از بسته اطلاعات صفر در نظر گرفته می شود

مثال: در کنار هر بایت ارسالی بیت نهم به عنوان بیت پرتی ارسال می شود (بیت پرتی حاوی اطلاعات اصلی نمی باشد بلکه تنها برای تشخیص خطا استفاده شده است)

فرض: در ارسال خطا و در هر بخش تنها احتمال خطای تک بیتی وجود دارد.

P	D_7	D_6	.				D_0

PAPCO

Subject
Date

دو دسته اصلی از پیرتی به شکل زیر تعریف می‌شود

۱) پیرتی زوج: تعداد یک های مجموعه نهایی (بیت + پیرتی) زوج باشد

۲) پیرتی فرد: " " " " " " فرد "

$\underset{P}{\underset{\uparrow}{1}}\ \underbrace{1\,10\,10110}_{\text{بایت}}$

در عمل یکی از این دو دسته انتخاب (قرارداد) می‌شود و در ارسال و دریافت استفاده خواهد شد

در ادامه مثال قبل

فرستنده با اضافه نمودن بیت پیرتی اطلاعات صحیح را ارسال می‌کند

ممکن است در طول عملیات رسیدن اطلاعات به گیرنده خطای تک بیتی رخ دهد

گیرنده: نه بیت دریافت خواهد شد که طبق قرارداد اولیه از نوع اطلاعات خبر دارد

برای اطمینان از صحت اطلاعات گیرنده عملیات زیر را برای بررسی صحت انجام خواهد داد

حالت ۱: خطایی رخ نداده است ✓ 111010110

حالت ۲: خطای تک بیتی رخ داده است ✗ 111011110

گیرنده مجددا پیرتی مجموعه را چک می‌کند و اگر با قرارداد اولیه صادق نباشد اعلام خطا می‌نماید

کد همینگ

فرض: همچنان امکان خطای تک بیتی وجود دارد و نه بیشتر

در ساده‌ترین حالت 4 بیت اطلاعات به همراه 3 بیت پیرتی ارسال خواهد شد $D_0\ D_1\ D_2\ D_3\ P_1\ P_2\ P_3$

خواهیم دید که در صورت رخ دادن خطای تک بیتی گیرنده می‌تواند آن را تشخیص و اصلاح نماید

قرارداد برای ایجاد بیت‌های پیرتی همراه اطلاعات اصلی:

$P_1 : \{P_1\ D_0\ D_1\ D_3\}$ پیرتی زوج مجموعه

$P_2 : \{P_2\ D_0\ D_2\ D_3\}$ پیرتی زوج مجموعه

$P_3 : \{P_3\ D_1\ D_2\ D_3\}$ پیرتی زوج مجموعه

در فرستنده با توجه به قرارداد فوق بیت های پیرتی تهیه شده و همراه اطلاعات اصلی ارسال می‌شوند

PAPCO

Subject
Date

ممکن است در طی ارسال از فرستنده به گیرنده خطای تک بیتی رخ داده باشد

گیرنده ارقام روبرو دریافت شده اند $D_0' \; P_1' \; D_2' \; D_3' \; P_1' \; P_2' \; P_3'$

می خواهیم بررسی کنیم که آیا اطلاعات دریافت شده صحیح می باشند یا خیر

گیرنده مطابق قرارداد اصلی بیتی مجموعه ها را در موردبیت های دریافتی (دارای ') چک می کند

حالت اول تمامی سه مجموعه صحیح باشند ⇐ هیچ خطایی رخ نداده است

حالت دوم یکی از مجموعه ها ناصحیح باشند

بیت پریتی موجود در آن مجموعه خطا دریافت شده است ⇐ اطلاعات اصلی صحیح بوده اند

حالت سوم دوتا از مجموعه ها ناصحیح باشند

بیت مشترک بین این دو مجموعه که در مجموعه سوم نمی باشد، خطا دریافت شده است

حالت چهارم هر سه مجموعه ناصحیح باشند ⇐ خطا در D_3

Subject
Date

جبر بولین (Boolean Algebra)

پایه های این جبر همان متغیر های منطقی صفر و یک می باشند

دو عملگر اصلی در این جبر تعریف می شوند، این دو عملگر به اصول تصمیم گیری در منطق مربوط می شوند

And و

Or یا

در نهایت در منطق می توان به صورت زیر بیان نمود:

برای نتیجه گیری خاصی هم اتفاق اول و هم اتفاق دوم بایستی رخ دهد (مفهوم عملگر And)

برای نتیجه گیری خاصی یا اتفاق اول یا اتفاق دوم بایستی رخ دهد (مفهوم عملگر Or)

And $a \wedge b = a \text{ And } b = a.b = ab$

Or $a \vee b = a \text{ Or } b = a+b \longrightarrow 1+1=1$

اصول موضوع:

ا.م 1- این جبر در یک دستگاه بسته تعریف شده است. (حاصل عملیات های انجام شده توی خود مجموعه هست)

ا.م 2- وجود عضو خنثی برای عملگرها

$a+0=a \qquad a.1=a$

ا.م 3- جابه جایی پذیری

$a+b=b+a \qquad a.b=b.a$

ا.م 4- شرکت پذیری

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

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

ا.م 5- پخش پذیری

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

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

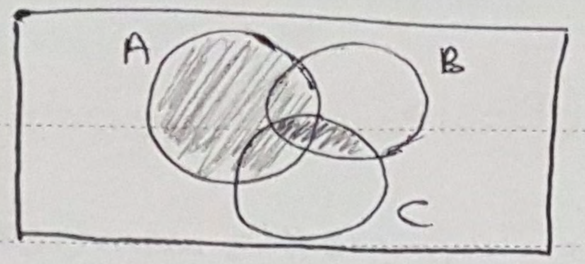
جدول صحت (Truth table)

a	b	c	bc	a+bc	a+b	a+c	(a+b).(a+c)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

سمت چپ = سمت راست

PAPCO

Subject
Date



نمودار ون

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

ا.م 6 وجود متمم

$a+\bar{a}=1 \qquad a.\bar{a}=0$

هر متغیری مانند $\bar{a}$ که در روابط فوق صدق کند، متمم a نامیده می‌شود

نکته: اگر $a+b=1$ باشد آیا b متمم a می‌باشد؟ خیر هر دو یک باشند ⇐ باید هر دو رابطه برقرار باشند

قضایای مهم جبر بولین

ق. 1 همتوانی

$a+a=a \qquad a.a=a$

ا.م 5 $a+a\bar{a}=(a+a)(a+\bar{a})$ $\qquad a.a=a$ بنا بر اصل همزادی برقرار است

ا.م 6 $a+0=(a+a)(1)$

ا.م 2 $a=a+a$

قضیه همزادی (دوگانی) در این درس به عنوان اصل پذیرفته می‌شود

اگر یک اتحاد برقرار باشد می‌توان نتیجه گرفت که اتحاد دیگری که به آن دوگان عبارت اول گفته می‌شود نیز برقرار است

دوگان یک عبارت به صورت زیر تعریف می‌شود

$+ \leftarrow \cdot$

$\cdot \leftarrow +$

$0 \leftarrow 1$

$1 \leftarrow 0$

ق. 2 عناصر بی‌اثر ساز

$a+1=1 \qquad a.0=0$

ق. 3 متمم متمم

$\bar{\bar{a}}=a$

ق. 4 جذب

$a+ab=a \qquad a(a+b)=a$

ق. 5

$a+\bar{a}b=a+b \qquad a(\bar{a}+b)=ab$

ق. 6

$ab+a\bar{b}=a \qquad (a+b)(a+\bar{b})=a$

PAPCO

Subject
Date

ق. 7

$ab+a\bar{b}c=ab+ac \qquad (a+b)(a+\bar{b}+c)=(a+b)(a+c)$

ق. 8

$ab+\bar{a}c+bc=ab+\bar{a}c \qquad (a+b)(\bar{a}+c)(b+c)=(a+b)(\bar{a}+c)$

*ق. 9 دمورگان

$\overline{a+b}=\bar{a}\cdot\bar{b} \qquad \overline{a\cdot b}=\bar{a}+\bar{b}$

به کمک ا.م.ح اثبات می‌کنیم که $\bar{a}\cdot\bar{b}$ و $a+b$ متمم یکدیگرند

$$\begin{cases}(a+b)+(\bar{a}\cdot\bar{b})=1\\(a+b)\cdot(\bar{a}\cdot\bar{b})=0\end{cases}$$

$$a+b+\bar{a}\bar{b}=1 \xrightarrow[5\text{ ق}]{a+\bar{a}\bar{b}=a+b} b+a+\bar{b}=1 \xRightarrow{\text{ا.م.ح}} (b+\bar{b})+a \xRightarrow{2\text{ ق}} 1+a=1$$

$$(a+b)\cdot(\bar{a}\bar{b})=a\bar{a}\bar{b}+b\bar{a}\bar{b}=0$$

**ق. شانون

تابع منطقی: تابعی است از یک یا چند متغیر منطقی به فضای منطقی $\{0,1\}$

$$f(x_1,x_2)=x_1 f(1,x_2)+\bar{x}_1 f(0,x_2)$$

$x_1=0 \quad f(0,x_2)=0\cdot f(1,x_2)+1\cdot f(0,x_2)=0+f(0,x_2)=f(0,x_2)$

$x_1=1$ به همین ترتیب

$$f(1,x_2)=x_2 f(1,1)+\bar{x}_2 f(1,0)$$

نتیجه نهایی $\Rightarrow f(x_1,x_2)=x_1x_2 f(1,1)+x_1\bar{x}_2 f(1,0)+\bar{x}_1x_2 f(0,1)+\bar{x}_1\bar{x}_2 f(0,0)$

با کمک ق. شانون، می‌توان برای هر تابع منطقی دلخواه یک عبارت جبری مشخص بدست آورد

PAPCO

Subject
Date

عبارت جبری And تنها درصورتی خروجی آن یک است که هردو عبارت ورودی یک باشند

$f(1,1)=1$ $\quad f(0,1)=0$ $\quad f(1,0)=0$ $\quad f(0,0)=0$

باکمک قضیه شانون $\quad f(x_1,x_2)=x_1x_2$

کافی ست شکل تصویر دامنه به برد را بدانیم

$$f(x_1,x_2,x_3)=x_1x_2x_3\,f(1,1,1)+x_1x_2\bar{x}_3\,f(1,1,0)+x_1\bar{x}_2x_3\,f(1,0,1)$$
$$+x_1\bar{x}_2\bar{x}_3\,f(1,0,0)+\bar{x}_1x_2x_3\,f(0,1,1)+\bar{x}_1x_2\bar{x}_3\,f(0,1,0)$$
$$+\bar{x}_1\bar{x}_2x_3\,f(0,0,1)+\bar{x}_1\bar{x}_2\bar{x}_3\,f(0,0,0)$$

مثال: مداری طراحی کنید که در یک شرکت اگر دقیقا 2 معاون در پشت میز خود کلید خودشان را بفشارد، درب ورودی باز شود

(۳ معاون حضور دارند)

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

چون جاهایی که $f=0$ نابودکننده عبارت است پس در نظر نمی‌گیریم

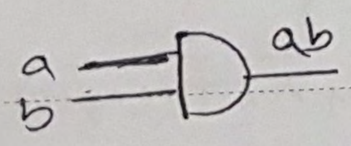
و فقط $f=1$ ها را در نظر می‌گیریم

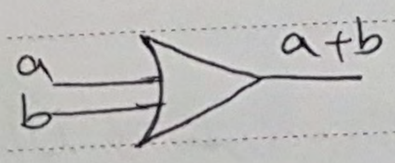
$$f(x_1,x_2,x_3)=\bar{x}_1x_2x_3+x_1\bar{x}_2x_3+x_1x_2\bar{x}_3$$

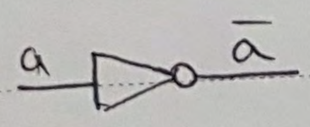
مدار منطقی

متناسب با عملیات های اصلی منطقی 2 عملگر اصلی تعریف نمودیم

گیت (دروازه) های منطقی







Not Gate

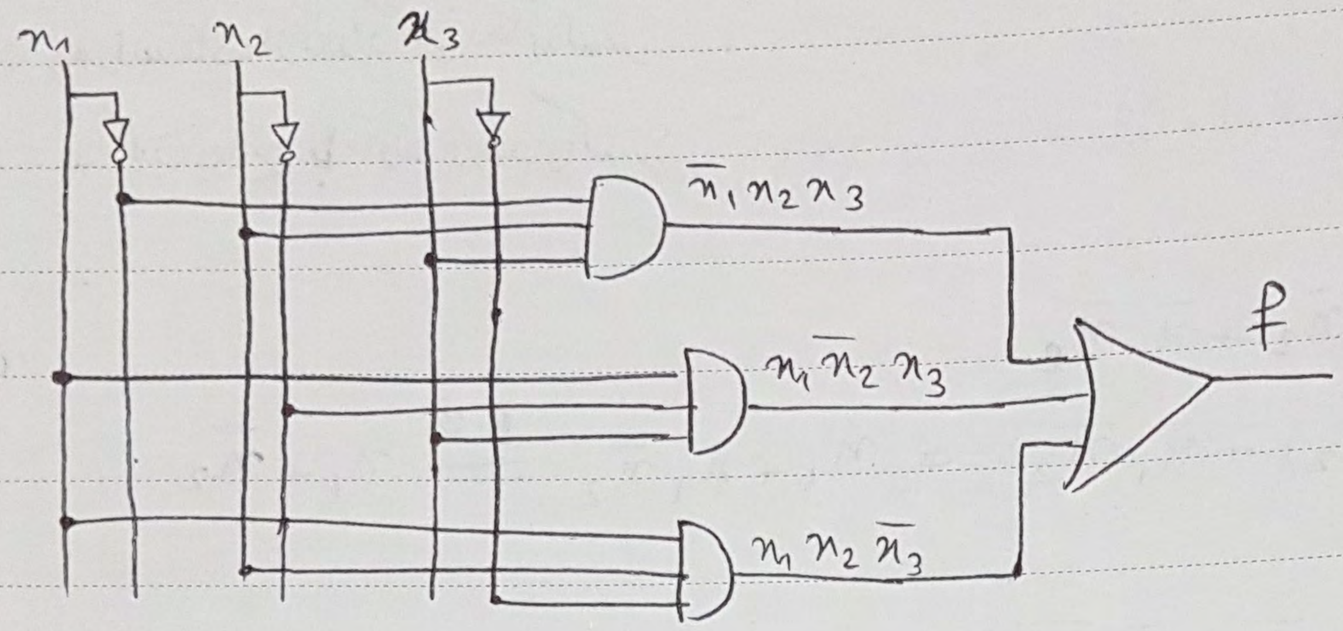
PAPCO

Subject
Date

• برای اتصال

○ برای نات ها

$$f(x_1, x_2, x_3) = \bar{x}_1 x_2 x_3 + x_1 \bar{x}_2 x_3 + x_1 x_2 \bar{x}_3$$



$$f(x_1, x_2, x_3) = \left(x_1 + x_2 + x_3 + f(0,0,0)\right)\left(x_1 + x_2 + \bar{x}_3 + f(0,0,1)\right)\left(x_1 + \bar{x}_2 + x_3 + f(0,1,0)\right)$$

$$\left(x_1 + \bar{x}_2 + \bar{x}_3 + f(0,1,1)\right)\left(\bar{x}_1 + x_2 + x_3 + f(1,0,0)\right)\left(\bar{x}_1 + x_2 + \bar{x}_3 + f(1,0,1)\right)$$

دوگان شانون

$$\left(\bar{x}_1 + \bar{x}_2 + x_3 + f(1,1,0)\right)\left(\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + f(1,1,1)\right)$$

قضیه شانون و دوگان آن با هم برابرند چون سمت چپ هیچ کدوم از عبارات دوگان سازی را ندارد و تغییری نمی کند و این هیچ ربطی به دوگان سازی ندارد

در دوگان شانون، $f = 1$ نابود کننده است و برای استفاده از دوگان شانون باید $f=0$ ها را در نظر بگیریم

سوال معاون ها با استفاده از دوگان شانون

$$f(x_1, x_2, x_3) = (x_1 + x_2 + x_3)(x_1 + x_2 + \bar{x}_3)(x_1 + \bar{x}_2 + x_3)$$
$$(\bar{x}_1 + x_2 + x_3)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3)$$

تعداد 0 ها و 1 ها تعیین کننده ی این است که از شانون استفاده کنیم یا دوگان شانون

اگر 0 ها کمتر بود از دوگان شانون

اگر 1 ها کمتر بود از شانون

PAPCO

Subject
Date

هدف: ساده سازی توابع منطقی در راستای تهیه ساده ترمدار متناظر

* استفاده از تعداد گیت های کمتر
* گیت هایی با تعداد ورودی کمتر

مثال:

$$f(x_1, x_2) = x_1 x_2 + x_1 \bar{x}_2 + \bar{x}_1 \bar{x}_2$$

$$= x_1 (x_2 + \bar{x}_2) + \bar{x}_1 \bar{x}_2 = x_1 + \bar{x}_1 \bar{x}_2 \overset{\text{ق.ت}}{=} x_1 + \bar{x}_2$$

$$f(x_1, x_2) = x_1 x_2 + x_1 \bar{x}_2 + \bar{x}_1 \bar{x}_2$$

$$= x_1 x_2 + x_1 \bar{x}_2 + x_1 \bar{x}_2 + \bar{x}_1 \bar{x}_2 \qquad (a + a = a)$$

تکرار جملات هیچ مانعی ندارد

$$= x_1 (x_2 + \bar{x}_2) + \bar{x}_2 (x_1 + \bar{x}_1)$$

$$= x_1 + \bar{x}_2$$

SOP

در اتحاد اصلی نخستین عبارت جبری به صورت جمع جملاتی است که در یکدیگر ضرب شده اند sum of products (SOP)
به هر کدام از این جملات استاندارد که حاوی تمام متغیرها و یا متمم آنها می باشند، مینی ترم (minterm) گفته می شود

مینی ترم m	x_1	x_2	f
0	0	0	1
	0	1	0
2	1	0	1
3	1	1	1

$$f(x_1, x_2) = \sum m(0, 2, 3)$$

POS

در دوگان اتحاد نخستین عبارت جبری به صورت ضرب جملاتی ست که در آنها جمع متغیرها حضور دارند product of sums (POS)
به هر کدام از این بخش های استاندارد که حاوی تمامی متغیرها و یا متمم آنها می باشند، ماکس ترم (Manterm) می گویند

ماکس ترم M	x_1	x_2	f
	0	0	1
1	0	1	0
	1	0	1
	1	1	1

$$f(x_1, x_2) = \prod M(1) = x_1 + \bar{x}_2$$

PAPCO

Subject
Date

$$f(x_1, x_2, x_3) = \bar{x}_1 x_2 x_3 + x_1 \bar{x}_2 x_3 + x_1 x_2 \bar{x}_3$$

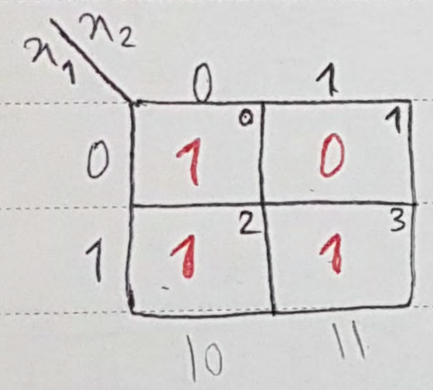
$$= \sum m(3,5,6) = \prod M(0,1,2,4,7)$$

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

اعداد موجود در سیاهه جملات SOP و POS مکمل یکدیگر می باشند

برای ساده سازی توابع روشهای مختلفی ابداع شده است که یکی از ساده ترین روش ها استفاده از جدول کارنو (Karnaugh) که به صورت دیداری ساده سازی انجام خواهد شد، می باشد

این جدول شامل سلولهائی است که هرکدام متناظر با یکی از سطور جدول صحت (حالتهای ممکن ورودی) می باشند



اختلاف تک بیتی { 00, 10 } { 00, 10 }

اختلاف دوبیتی { 00, 11 }

هر دو سلول مجاور در جدول کارنو (همسایه) اختلاف تک بیتی دارند
همسایه بودن یعنی فیلع مشترک داشتن (ضرب بردری همسایه نفیت)

$$\bar{x}_1 \bar{x}_2 + x_1 \bar{x}_2 = (\bar{x}_1 + x_1)\, \bar{x}_2 = 1 \cdot \bar{x}_2 = \bar{x}_2$$

پس قطعاً سلولهای همسایه قابل ادغام هستند و نتیجه ادغام آنها ساده تر شدن تابع است

PAPCO

Subject
Date

برای انجام ساده‌سازی بایستی از یکی از بیان های SOP و POS آغاز کنیم و تابع جبری متناظر را در هر کدام مستقلاً ساده می‌کنیم

SOP:

* در بسیاری موارد تنها سلول‌های مین‌ترم‌ها در جدول با رقم 1 نشانه‌گذاری می‌شوند و 0 ها (ماکس‌ترم ها) خالی می‌مانند
* سلول های {0,2} و سلول‌های {2,3} همسایه می‌باشند
* با هدف پوشاندن تمامی 1 ها و انتخاب بزرگترین مجموعه های ممکن (حتی با تکرار مین‌ترم ها) به ساده‌ترین عبارت ممکن در بیان SOP خواهیم رسید
* مجددا تابع جبری متناظر ساده شده را بدست می‌آوریم

$$f(x_1, x_2) = \overset{\text{افقی}}{x_1} + \overset{\text{عمودی}}{\overline{x_2}}$$

$x_1 \backslash x_2$	0	1
0	1	
1	1	1

* کافی‌ست به مجموعه‌ها و مقادیر متناظر ورودی‌ها دقت کنیم ← متغیری که مقدار آن تغییر می‌کند حذف می‌شود
مابقی متغیرها که ثابت مانده‌اند متناظر با تعریف SOP ظاهر خواهند شد

سه متغیره

$$f(x_1, x_2, x_3) = \sum m(3,5,6) = \prod M(0,1,2,4,7)$$

$x_1 \backslash x_2x_3$	00	01	11	10
0	0	1	3	2
1	4	5	7	6

* کدگذاری گری استفاده شده است تا اختلاف تک بیتی بین همسایه‌ها ایجاد کند
* اگر تابعی n متغیره باشد به ازای هر سلول بایستی n همسایه وجود داشته باشد
* در این جداول کدگذاری استفاده شده است همسایگی به صورت چرخشی نیز وجود دارد

SOP

$x_1 \backslash x_2x_3$				
0			1	
1		1		1

POS:

$x_1 \backslash x_2x_3$	00	01	11	10
0	0	0	1	3 / 2
1	4	5	7	6

سه مجموعه 2 تایی و یک مجموعه تکی

$$f = (\overline{x_1} + \overline{x_2} + \overline{x_3})(x_2 + x_3)(x_1 + x_2)(x_1 + x_3)$$

{7} مجموعه {0,4} مجموعه {0,1} مجموعه {0,2} مجموعه

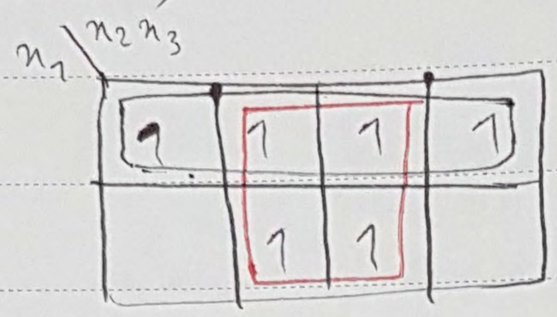
PAPCO

Subject
Date

نکته: دقت می‌کنیم که در بیان‌های SOP و POS ترتیب اشاره متغیرهای ورودی بسیار اهمیت دارد بنابراین در اشاره شماره جملات مین ترم و یا ماکس ترم حتماً بایستی به ترتیب ورودی‌ها اشاره کرد

$f(x_1, x_2, x_3) = \sum m(0,1,2)$ 010
$f(x_3, x_2, x_1) = \sum m(0,2,4)$ 001
000

مثال: مجموعه‌های صحیح را انتخاب کنید



* مجموعه‌های 4 تایی مربعی و یا مستطیلی (L قابل قبول نیست) قابل ادغام اند

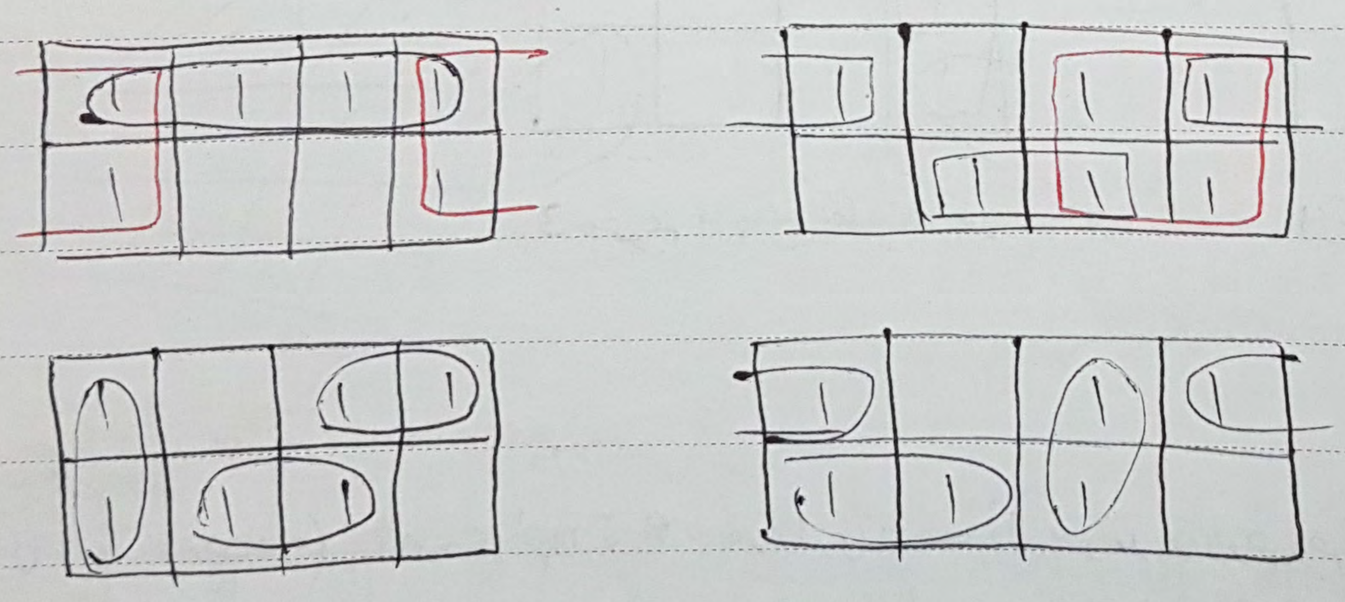
مجموعه قرمز: $\underbrace{\bar{x}_1 \bar{x}_2 x_3 + \bar{x}_1 x_2 x_3} + \underbrace{x_1 \bar{x}_2 x_3 + x_1 x_2 x_3}$

$= \bar{x}_1 x_3 (\bar{x}_2 + x_2) + x_1 x_3 (\bar{x}_2 + x_2)$

$= x_3 (\bar{x}_1 + x_1) = x_3$

$f = \overset{\text{قرمز}}{x_3} + \overset{\text{سیاه}}{\bar{x}_1}$

مثال:

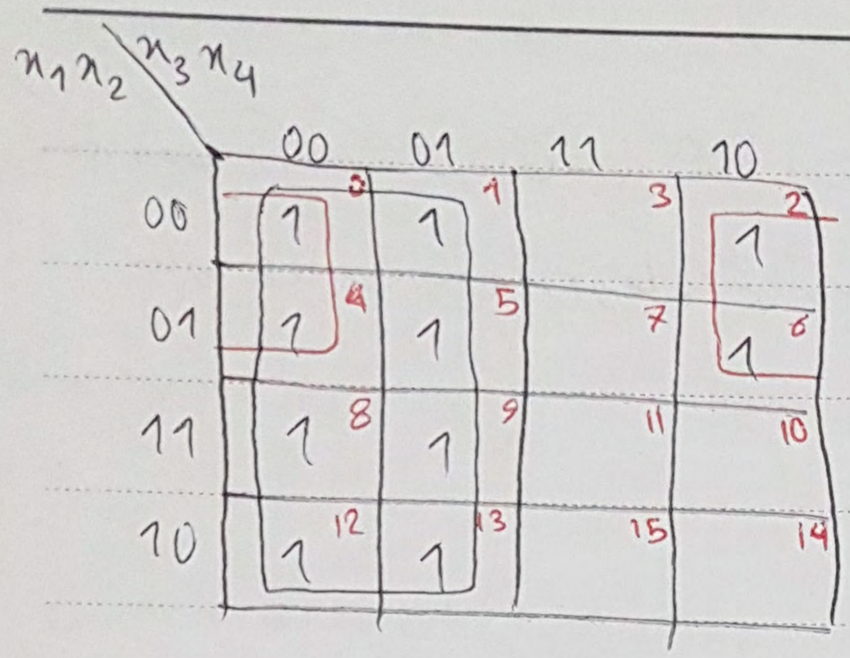


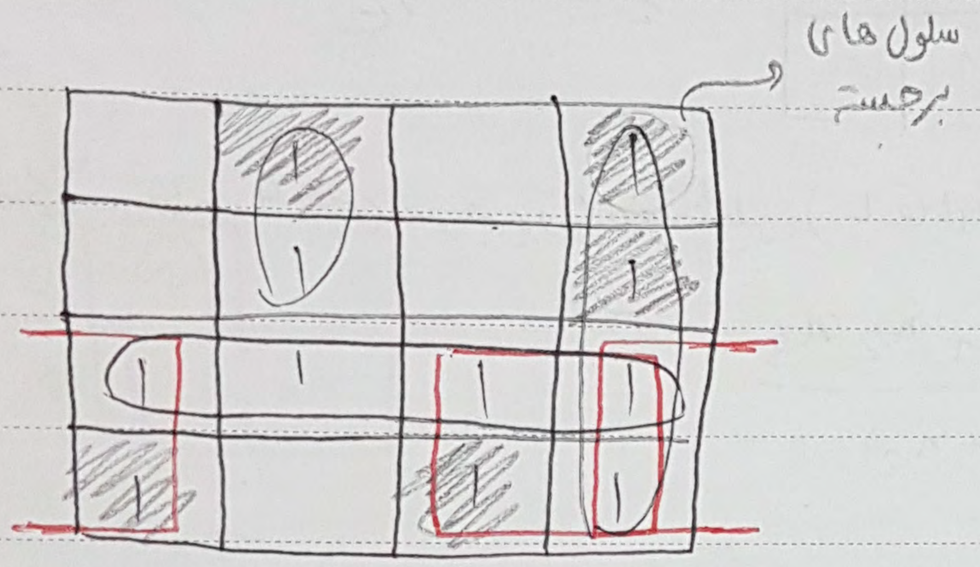
* مجموعه اضافی ضرورتی ندارد
* ممکن است پاسخ یکتا نباشد

Subject
Date

مثال 4 متغیره

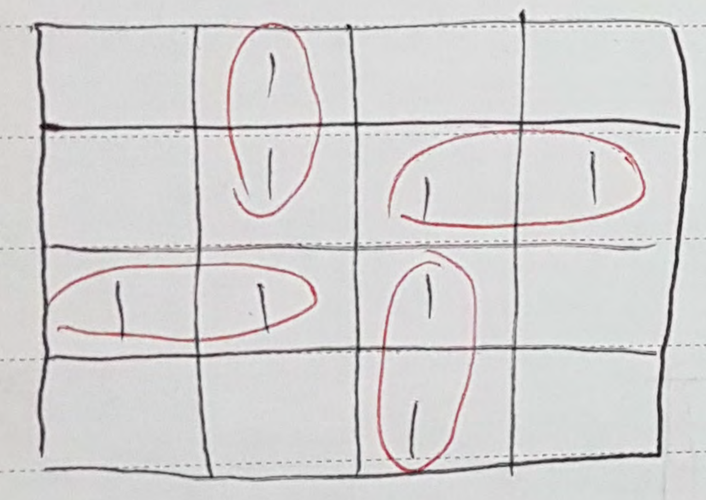
مجموعه 8 تایی مستطیلی 2×4 قابل ادغام است



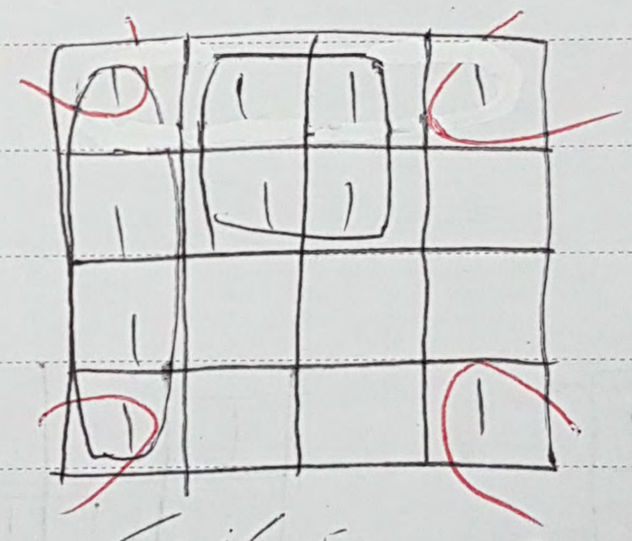


4 مجموعه 4 تایی و یک مجموعه 2 تایی

** مثال **



4 تا مجموعه 2 تایی کافی نیست و نیاز به 4 تایی وسط نیست



3 مجموعه 4 تایی کفایت می کند

الگوریتم یافتن بهترین انتخاب مجموعه

شامل (نتیجه دهنده) Implicant : به هر عبارتی که مستقلاً بتواند مقدار خروجی را تعیین نماید می تواند خود جملات مین ترم و یا ادغامی از آنها باشد (I)

شامل اول : prime Implicant : به شاملی گفته می شود که شامل دیگری آن را نپوشاند (PI)

سلول برجسته Distinguished cell : سلولی که تنها در یک شامل اول قرار داشته باشد (DC)

PAPCO

Subject
Date

شامل اول اساسی Essential PI : شامل اولی که حاوی حداقل یک سلول برجسته باشد (EPI)

گام اول: شناسایی تمامی شامل‌ها

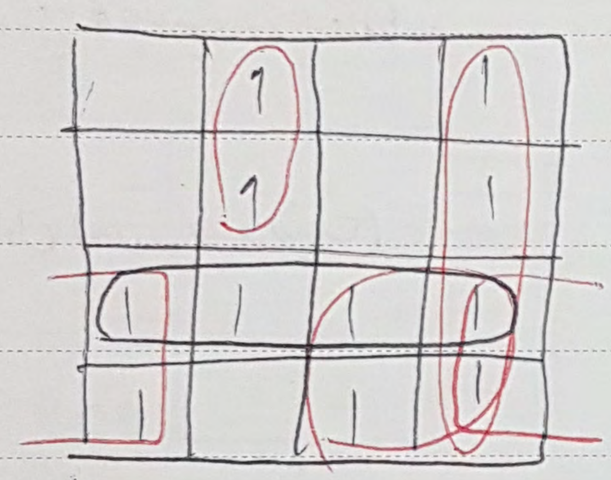
گام دوم: شناسایی شامل‌های اول

گام سوم: شناسایی سلول‌های برجسته

گام چهارم: در نظر گرفتن شامل‌های اول اساسی در خروجی تابع

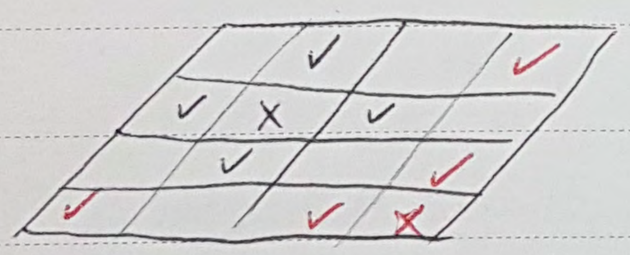
گام پنجم: پوشاندن ما بقی سلول‌های مورد نظر با کمترین تعداد شامل‌های اول و بزرگترین آن‌ها

در مورد مثال **



گام سوم انجام شده در صفحه قبل

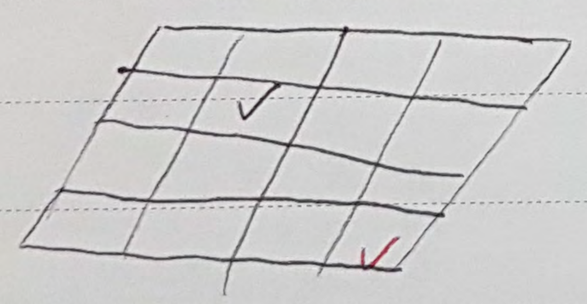
مجموعه سیاه رنگ در گام پنجم انتخاب شده است



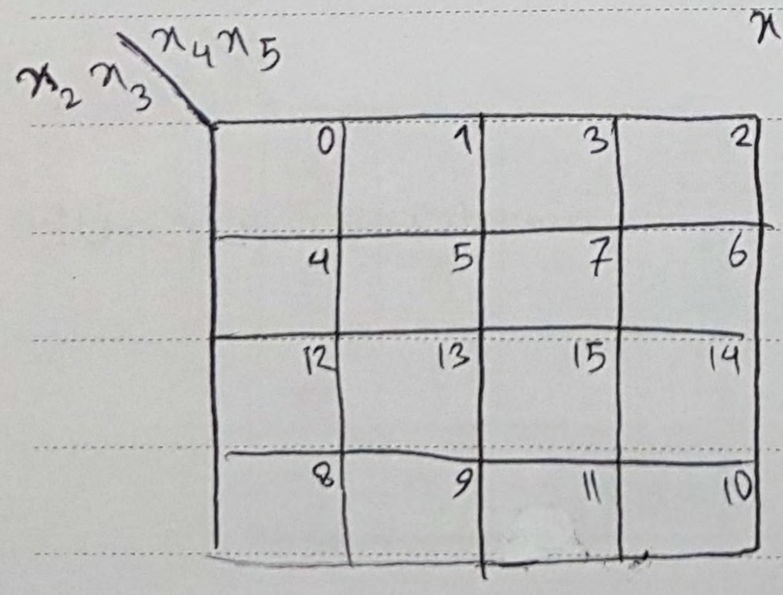
✓ خانه مورد نظر

✓ همسایه سلول مورد نظر

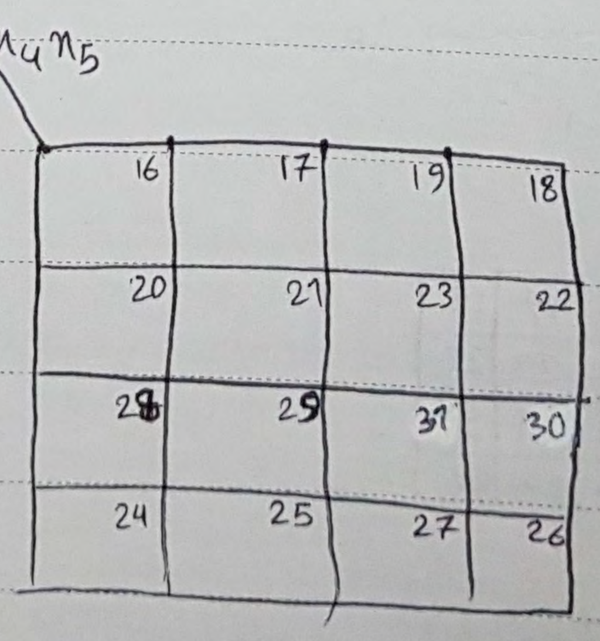
5 متغیره



همسایگی پنجم در صفحه دیگر نشان داده شده است که برای انتخاب مجموعه‌های مشترک بین دو صفحه بایستی سلول‌های متقارن در هر دو وجود داشته باشد



$x_1 = 0$



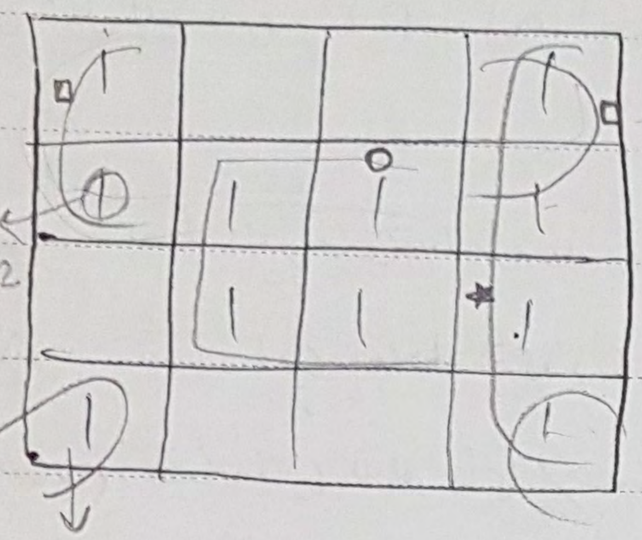
$x_1 = 1$

PAPCO

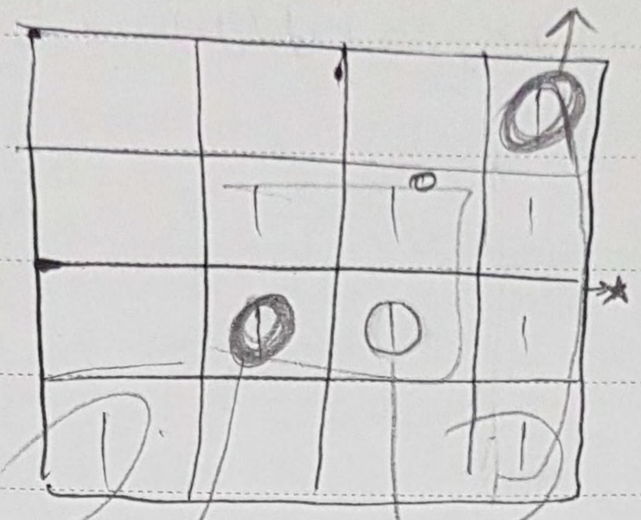
Subject

Date

صورت ساده مجموعه‌های زیر را بیابید



برجسته نیست
چون توی 2 تا 4 تایی هستا
که هر دو شامل اول

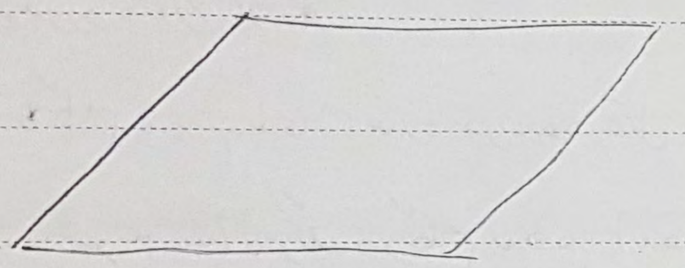


8 تایی گذاشتیم => برجسته هست

برجسته هست => 8 تایی وسط باشه

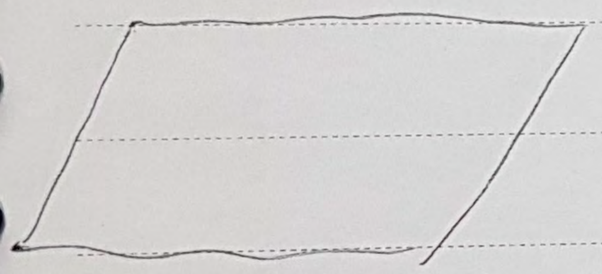
برجسته نیست
چون توی 2 تا 8 تایی هست
که هر دو شامل اول هستند

بیان توابع 6 متغیره در جدول کارنو

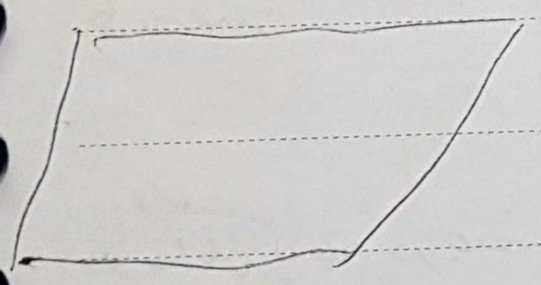


$x_1 x_2 = 00$

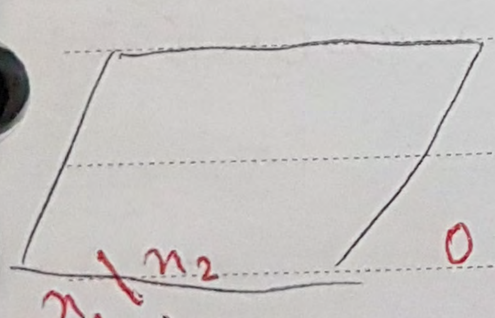
بیان اول (مورد پسند نیست)



01



11



10

1

$x_1 \backslash x_2$

0

$x_3 x_4 \backslash x_5 x_6$

0

0	1	3	2
4	5	7	6
12	13	15	14
8	9	11	10

16	17	19	18
20	21	23	22
28	29	31	30
24	25	27	26

1

32	33	35	34
36	37	39	38
44	45	47	46
40	41	43	42

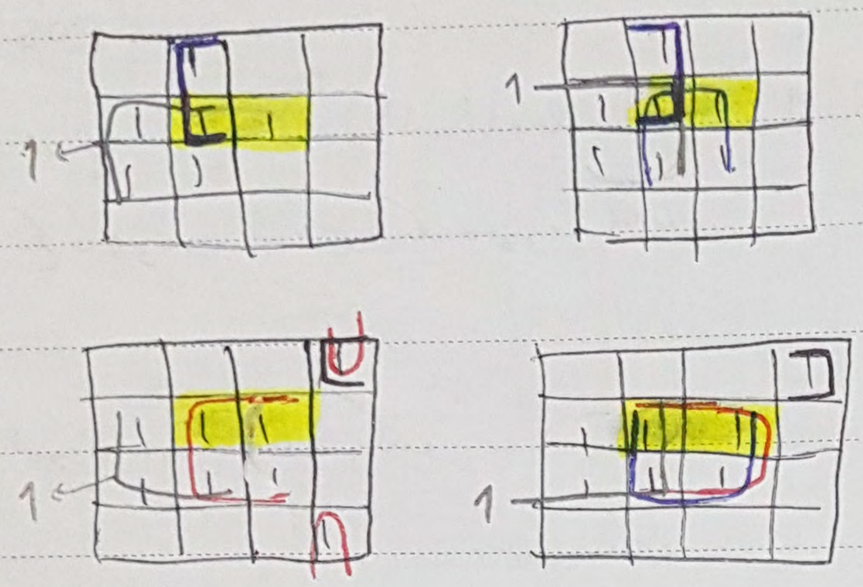
48	49	51	50
52	53	55	54
60	61	63	62
56	57	59	58

بیان دوم (مورد پسند ما هستند)

PAPCO

Subject

Date



1×16

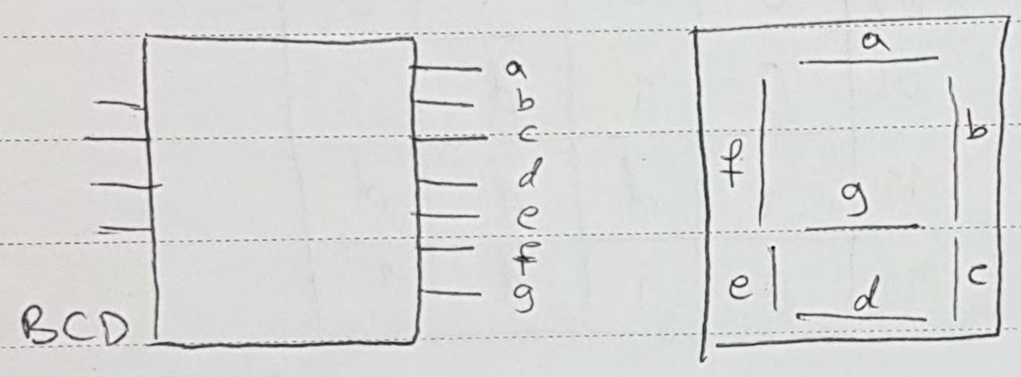
3×8

1×4

2×2

حالات رها (بی اهمیت) Don't Care

می‌خواهیم مداری طراحی کنیم که ارقام دهدهی را بر روی نمایشگر 7S نشان دهد



می‌خواهیم در ابتدا مدار تهیه کننده خروجی e را طراحی کنیم

از آنجایی که می‌دانیم (صورت مسئله) در ورودی BCD متناظر دودویی اعداد 10 ... 15 هیچگاه اعمال نخواهد شد

پس مقدار خروجی متناظر با این مقادیر اهمیتی نخواهند داشت

MSB LSB

$\underbrace{x_3 x_2}_{\text{پراهمیت ها}} \underbrace{x_1 x_0}_{\text{کم اهمیت ها}}$

x_3x_2 \ x_1x_0	00	01	11	10
00	1 (0)	0 (1)	0 (3)	1 (2)
01	0 (4)	0 (5)	0 (7)	1 (6)
11	d (12)	d (13)	d (15)	d (14)
10	1 (8)	0 (9)	d (11)	d (10)

ساده سازی:

Sop:

همچنان با معیارهای کمترین تعداد و بزرگترین مجموعه‌ها طراحی می‌تواند مقدار d ها را یک یا صفر در نظر بگیرد

در مثال فوق
$m(10, 14) = 1$ مابقی d ها صفر
$m(10, 12, 14) = 1$ مابقی d ها صفر X انتخاب اشتباه

که در هر دو انتخاب (P) با 2 مجموعه 4 تایی می‌توان تابع را ساده کرد

PAPCO

Subject
Date

	0	0	
0	0	0	
d	d	d	d
	0	d	d

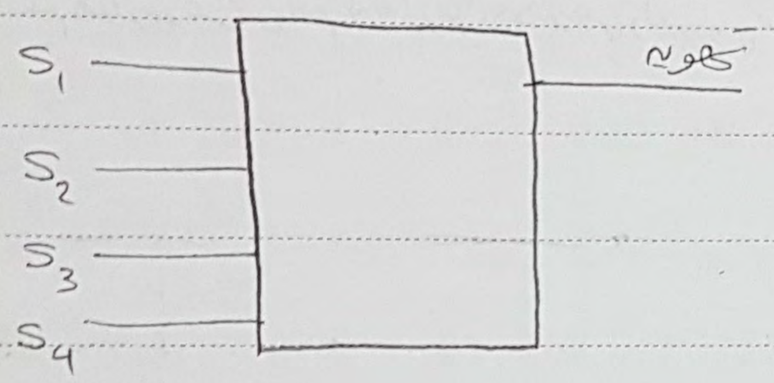
POS :

در این شرایط $M(11,12,13,15)=0$ و $M(10,14)=1$

یک مجموعه‌ی 8 تایی و یک مجموعه 4 تایی

تمرین ← 7s را طراحی کنید

مثال: در شرکتی 4 اتاق وجود دارد و 2 کارمند در آن مشغول به فعالیت هستند در هر اتاق سنسور IR وجود دارد که می‌تواند حضور فردی در اتاق را تشخیص دهد مداری طراحی کنید که اگر در دو اتاق مختلف حضور داشته باشند تهویه روشن شود



S_4S_3 \ S_2S_1	00	01	11	10
00	0	0	1	0
01	0	1	d	1
11	1	d	d	d
10	0	1	d	1

SOP :

S_4S_3 \ S_2S_1				
			1	
	1	d	d	1
1	d	d	d	d
	1	d	d	1

سلول‌های d به عنوان سلول برجسته در نظر گرفته نمی‌شوند

در این شرایط 6 مجموعه 4 تایی وجود دارند

0	0		0
0		d	
	d	d	d
0		d	

POS

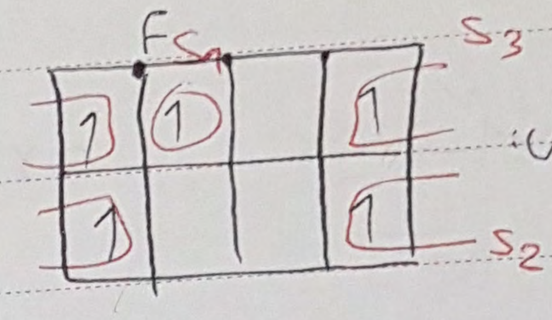
4 مجموعه 2 تایی بدون کمک dها $(d=1)$

Subject
Date

سوال: آیا در مدارهایی که دارای چند خروجی هستند، بایستی هر خروجی را جداگانه ساده نمود یا روش بهتری وجود دارد؟

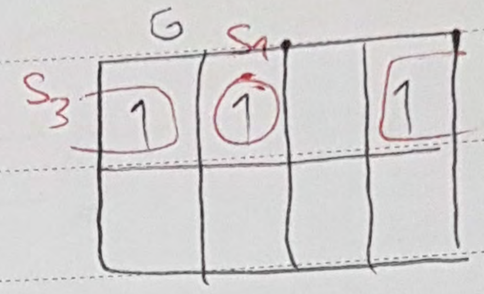
بله ممکن است در صورت ساده سازی یکپارچه ی تمام خروجی ها به مدار کلی ساده تری برسیم

مثال:



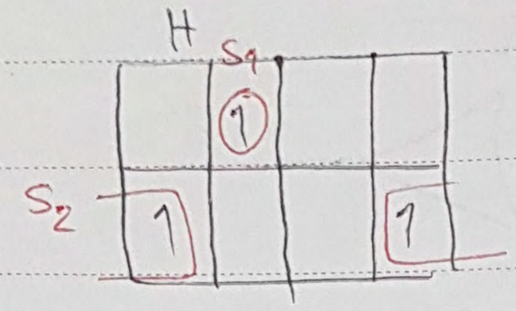
یک چهارتایی و یک دوتایی

$S_1+S_2+S_3$



2 تا 2 تایی

S_1+S_3

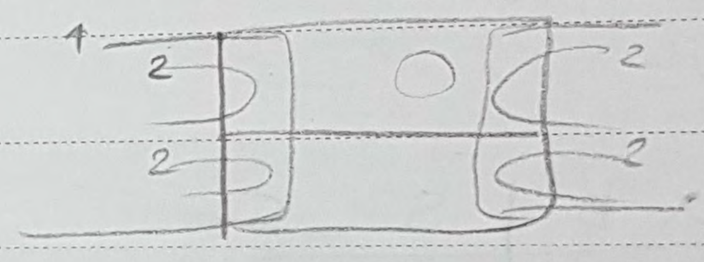


یک مجموعه دوتایی و یک تکی

S_1+S_2

روش ساده سازی Q M کوئین مک کلاسکی

تمرین: توسط نرم افزار دلخواه روش Q M را پیاده سازی کنید



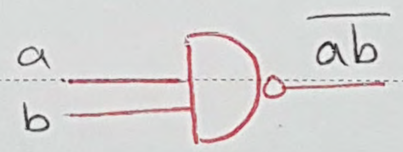
1) هیچ گیتی نمی خواد

2) یک گیت OR دو ورودی را به 3 ورودی تغییر می دهیم

گیت های کمکی

گیت Nand نزد فلش اصطلاح کامپیوتری

Nand : And + Not



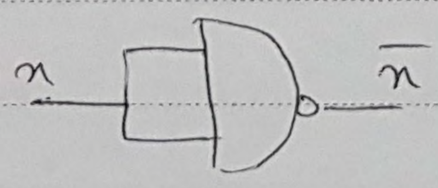
از لحاظ عملی و ساخت، هزینه و مابقی پارامترهای مدار در داخل گیت Nand از بقیه گیت های موجود بهتر می باشد

گیت Nand یک گیت کامل (Universal) می باشد

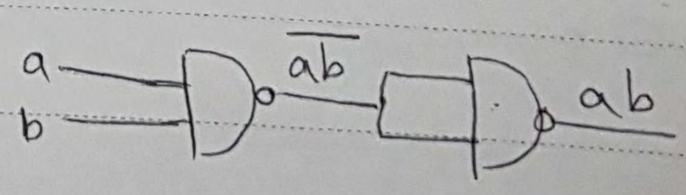
هر مدار دلخواهی را می توان با تعداد کافی و تنها از گیت های Nand تهیه نمود

می بینیم که تمام گیت های پایه را می توان با گیت های Nand تهیه نمود

Not

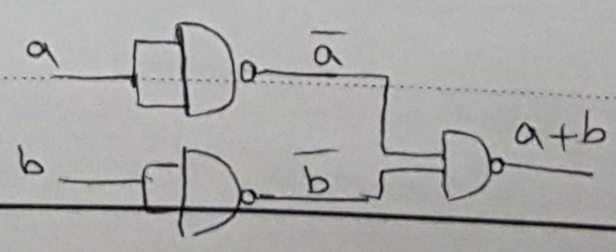


And



or $\overline{a \cdot b} = \bar{a} + \bar{b}$

$\overline{\bar{a} \cdot \bar{b}} = a + b$



PAPCO

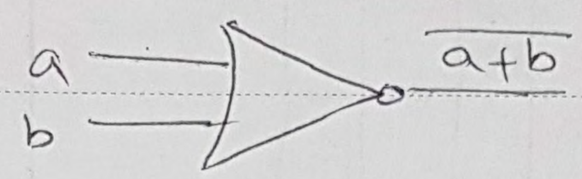
Subject
Date

ترکیب های

در عمل الگوریتم های پیچیده‌تر و جامع‌تری در مورد ساده سازی مدار با گیت‌های Nand وجود دارد که سازنده حتی به نسخه های تخصصی دیگران دسترسی ندارد

گیت NOR

OR + Not



گیت NOR

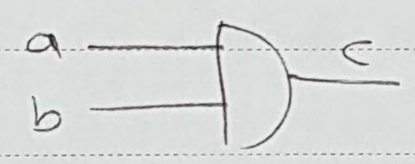
گیت NOR نیز یک گیت کامل است (به سادگی Nand نیست و از این استفاده نمی کنیم خیلی)

گیت بعدی را با یک مثال یاد می گیریم: می خواهیم دو عدد تک بیتی را جمع کنیم، مداری معرفی کنید که بیت های خروجی حاصل جمع (S) و رقم نقلی (C) را محاسبه کند

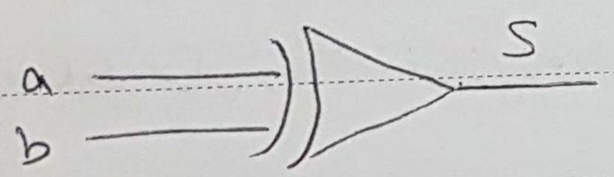
$$\begin{array}{r} 1 \\ +\ 1 \\ \hline 10 \end{array}$$

← حداکثر ۲ بیت نیاز است

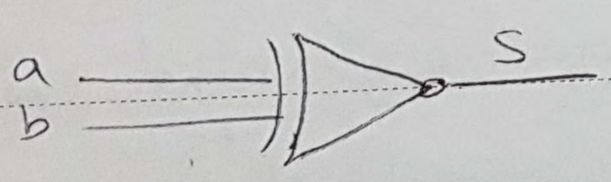
a	b	S	C
0	0	0	0
0	1	1	0
1	1	0	1
1	0	1	0



$\bar{a}b + a\bar{b}$



گیت XOR نابرابری ورودی ها را نشان می دهد $S = a\bar{b} + \bar{a}b = a \oplus b$



گیت XNOR برابری ورودی ها را تشخیص می دهد $S = ab + \bar{a}\bar{b} = a \odot b$

1) $a \oplus b = \overline{a \odot b}$

2) $a \oplus b = a \odot \bar{b}$

* اگر در امتحان آمد جبری باید حل بشود

PAPCO

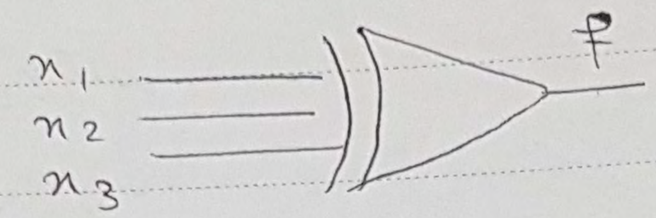
Subject

Date

مثال: مقابل را برای خودمون ثابت کنیم

$x_1 \backslash x_2x_3$			
	1		1
1		1	

$$f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$$

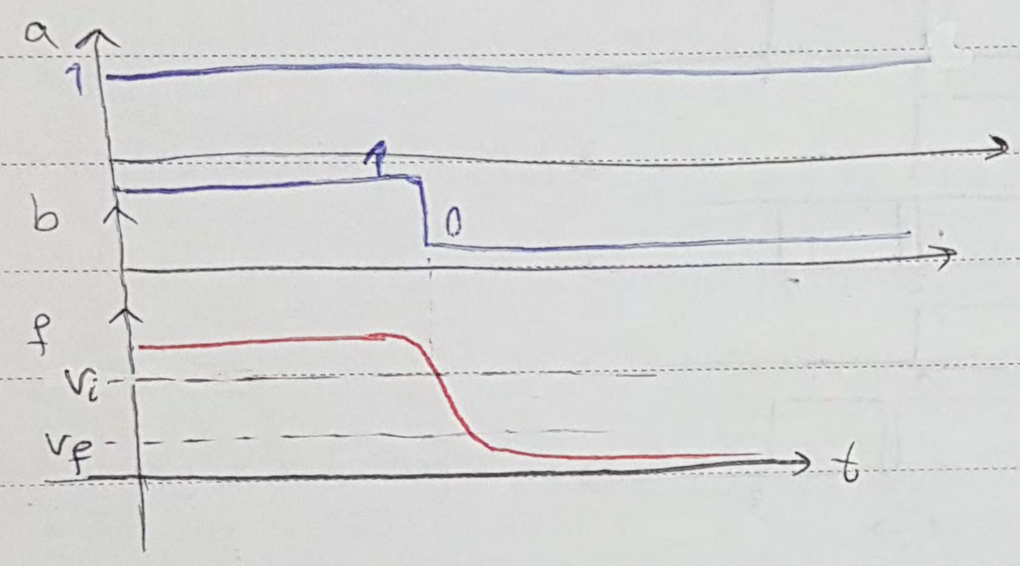


دیاگرام زمانی

در عمل هیچ کدام از گیت ها ایده آل نیستند

برای تهیه خروجی منطقی متناظر به زمان نیاز دارند (تاخیر انتشار)

فرض کنید در مثال زیر تغییرات ورودی در زمان گفته شده است و تاخیر زمانی خروجی 2^{ns} باشد

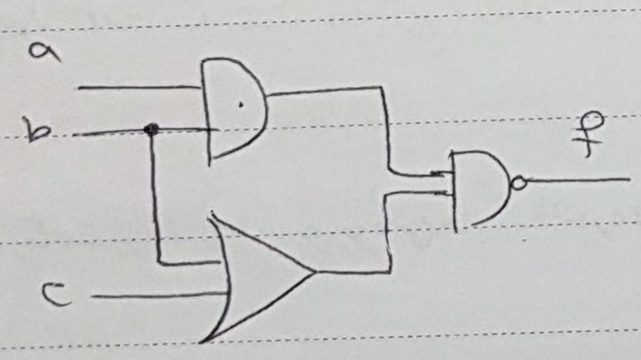


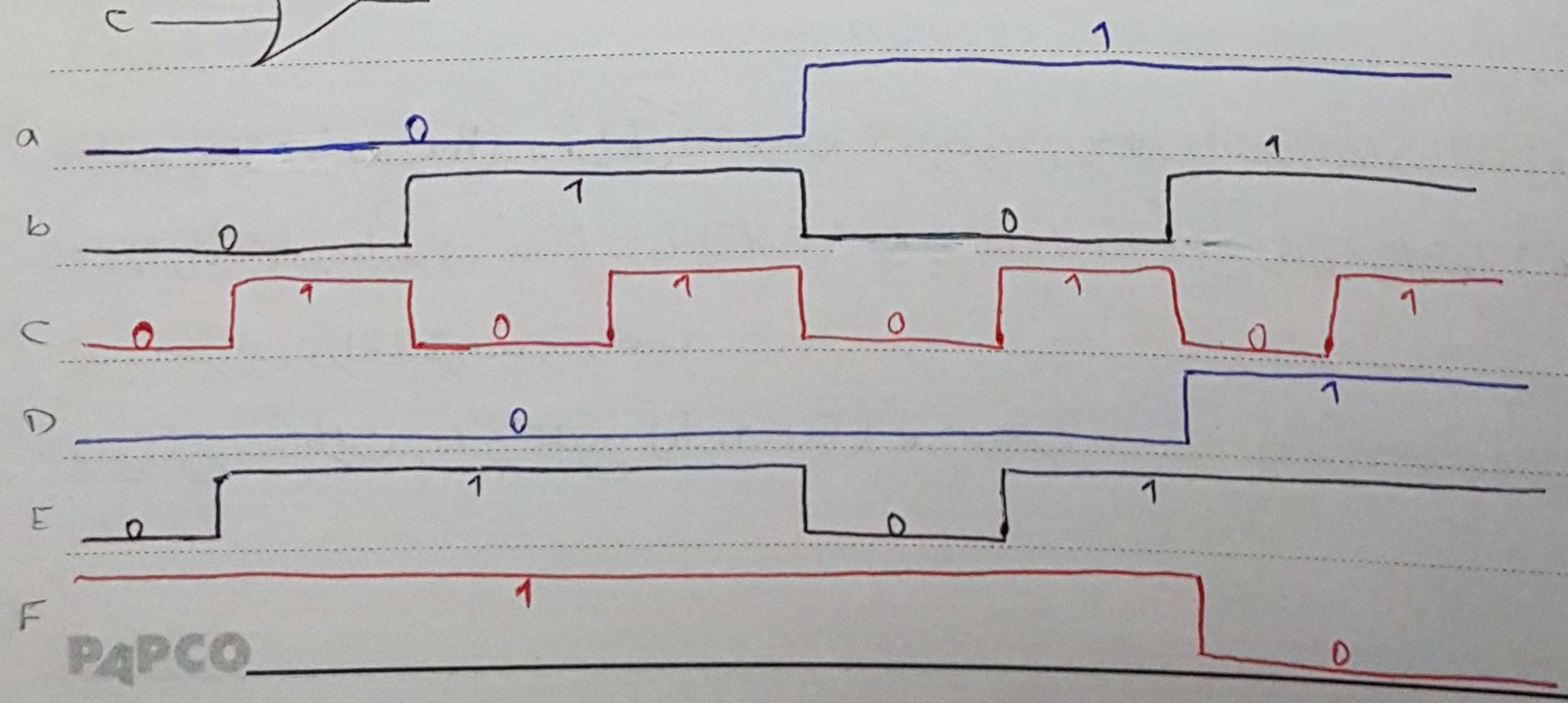
تاخیر انتشار اختلاف زمانی تغییر حالت خروجی بین منطق صفر و یک و یا بالعکس می باشد

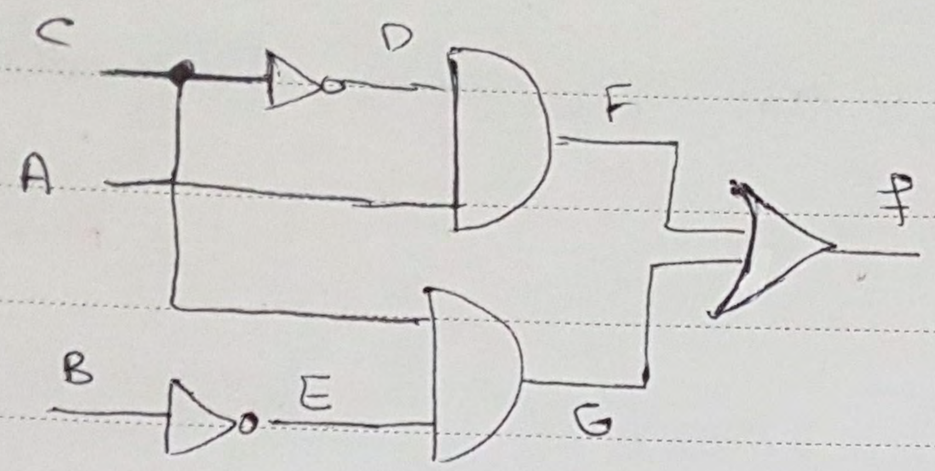
t_f : تاخیر سقوط t_r : تاخیر صعود

t_d : در این درس به معنای تاخیر انتشار چه در صعود و چه در نزول است

مثال: دیاگرام زمانی مقابل را بکشید (بدون تاخیر)

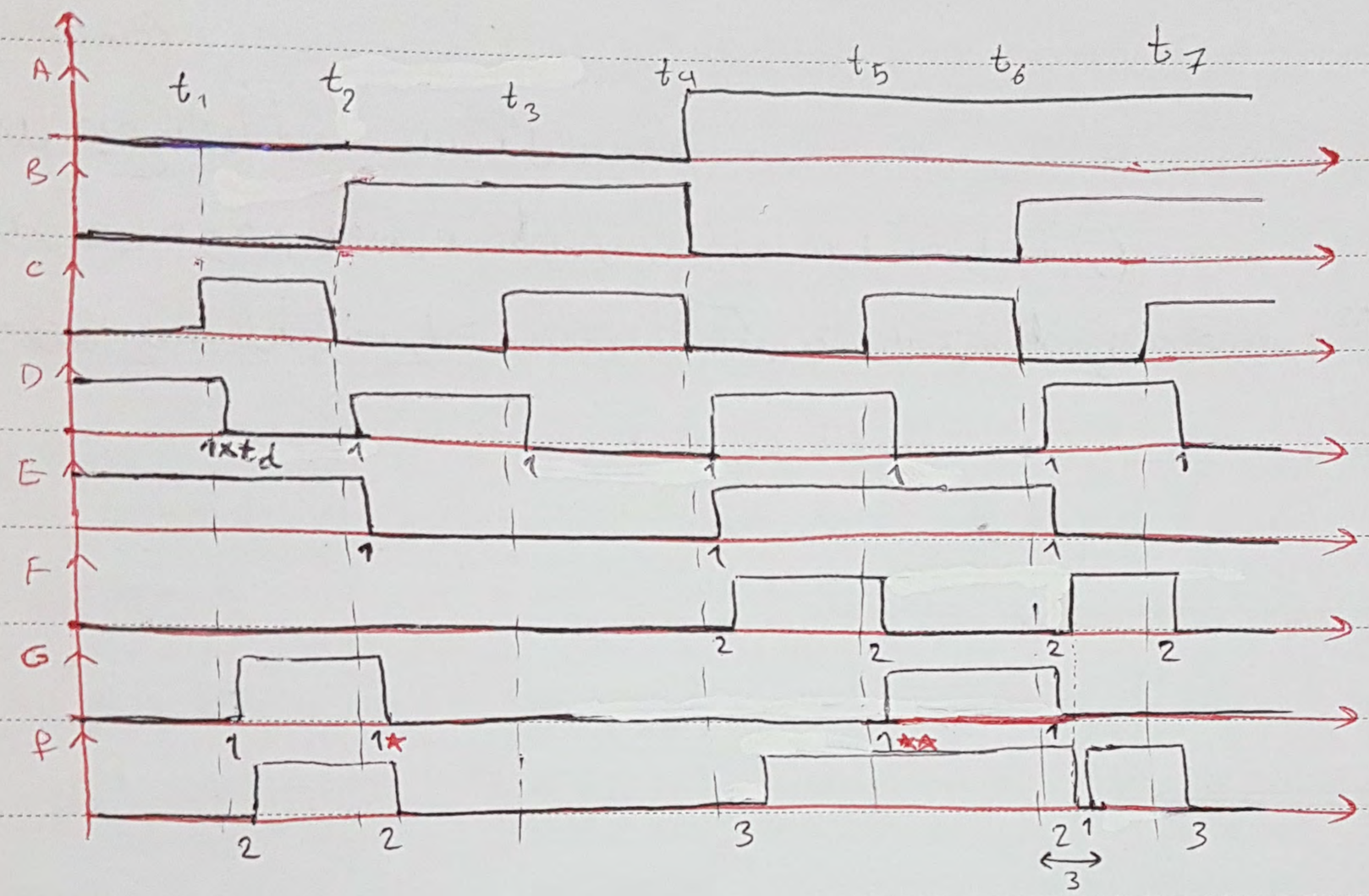






مثال 2-36 ص 133

تمام گیت‌ها دارای تاخیر t_d می‌باشند



★ همین که C از یک به صفر میره کافیه برای تغییر از یک به صفر G

★★ C که تاخیر ندارد

$2\times\binom{8}{2}$ حالت مختلف را باید بررسی کرد تا تمام شرایط تغییر ورودی را ببینیم

همانطور که در مثال فوق می‌بینیم خروجی نهایی به اندازه‌ی تعداد طبقات × تاخیر انتشار هر گیت نسبت به ورودی‌ها تاخیر نشان می‌دهد

تعداد طبقات $\triangleq$ بیشترین تعداد گیت‌ها در مسیرهای بین هر کدام از ورودی‌ها و خروجی

در فاصله زمانی تغییر ورودی‌ها تا تاخیر مدار ممکن است خروجی نتیجه غلط نشان دهد

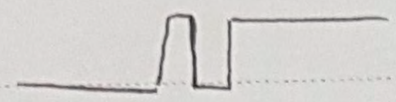
مخاطره زمانی (Hazards) به شرایطی گفته می‌شود که در خروجی برخلاف انتظار مقادیر اشتباه گزارش می‌شود

به عنوان مثال در مدار فوق: مخاطره زمانی ثابت 1 دیده شده است: بایستی خروجی با مقدار 1 ثابت می‌بوده است اما در فاصله زمانی تغییر خروجی در لحظاتی صفر شده است

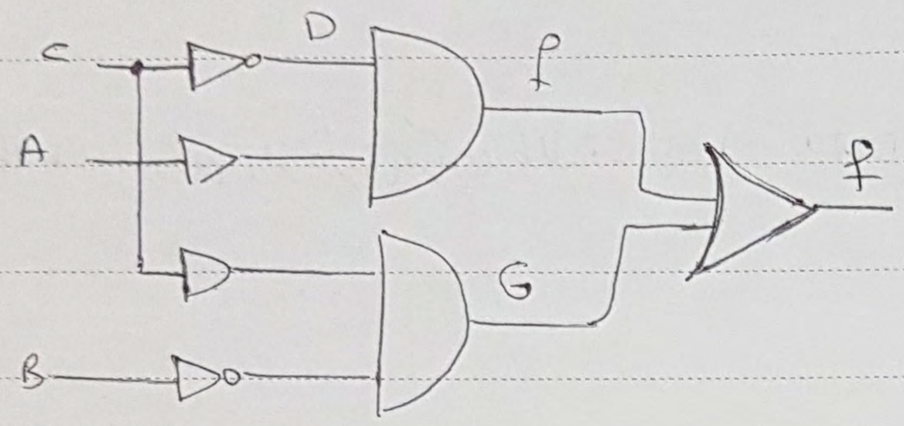
به این نوع مخاطرات زمانی گلیچ (Glitch) گفته می‌شود

Subject
Date

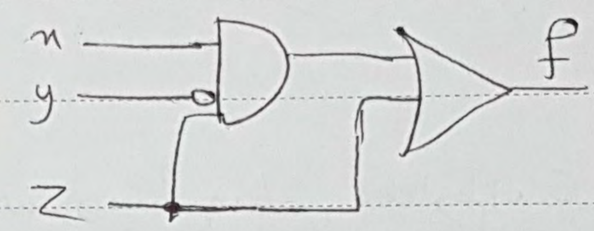
ممکن است در مدارهای دارای طبقات بیشتر خطرات نوسانی رخ دهد مثال:



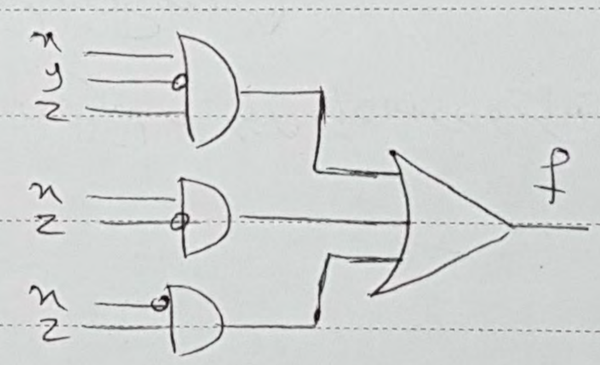
در طراحی مداری دلیل ایجاد این خطرات زمانی اختلاف تاخیر انتشار از ورودی به خروجی مدار در مسیرهای مختلف می باشد
برای حل مشکل فوق می توان به مسیرهای سریع، تاخیر اضافه نمود به عنوان مثال با کمک بافر (Buffer)
همچنین بافر سطوح منطقی بافر را بهبود می دهد



مثال $f(x,y,z) = x\bar{y}z + z$



$f(x,y,z) = x\bar{y}z + xz + \bar{x}z$



می توان از گیت های مشابه در مسیرهای مختلف استفاده نمود که تا حد امکان زمان تاخیر آنها نزدیک به هم باشد
یعنی حالتی که در مثال برای حل مشکل استفاده کردیم بهتر از حالت با فر است

طراحی دروازه ای

Small	scale	Integration	(SSI)	1 ... 10
medium	~	~	(MSI)	10 ... 100
Large	~	~	(LSI)	100 ... 10^4
very Large	~	~	(VLSI)	10^4 ... 10^6
ultra Large	~	~	(ULSI)	10^6 ...

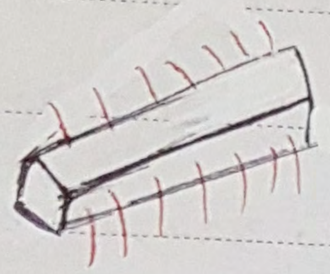
تعداد گیت ها در مقیاس مجتمع سازی

افزایش تعداد درخواست (تقاضا) برای مدارهای خاص

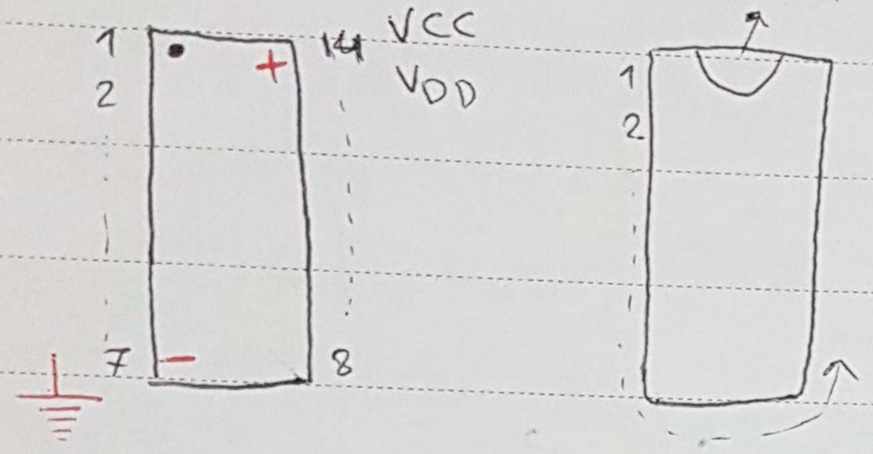
PAPCO

طراحی با کمک ماجول های آماده

با کمک تراشه های موجود در بازار مدارهای منطقی طراحی می شود



برای نابینایان و کم بینایان



اطلاعات مورد نیاز از طریق برگه ی اطلاعات (Data Sheet) بدست می آید

ورودی همواره اعمال می شود ← اگر به ورودی اتصالی داده نشود سطح منطقی نامعلوم است

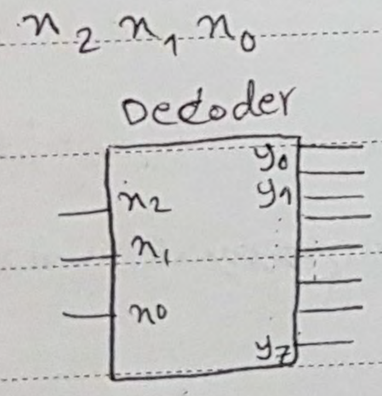
برای اتصال قطبهای منبع ولتاژ (تعیین مستقیم سطح منطقی)

از مقاومتهای pull_up و pull_down استفاده می کنیم تا جریان زیاد نشود و تراشه نسوزد

رمزگشا (De coder)

در این تراشه کد ورودی (پیچیده تر) به کد خروجی (ساده تر) ترجمه (تبدیل) می شود

دکودر دودویی: کد دودویی ورودی را به کد 1 از 2^n خروجی تبدیل می کند

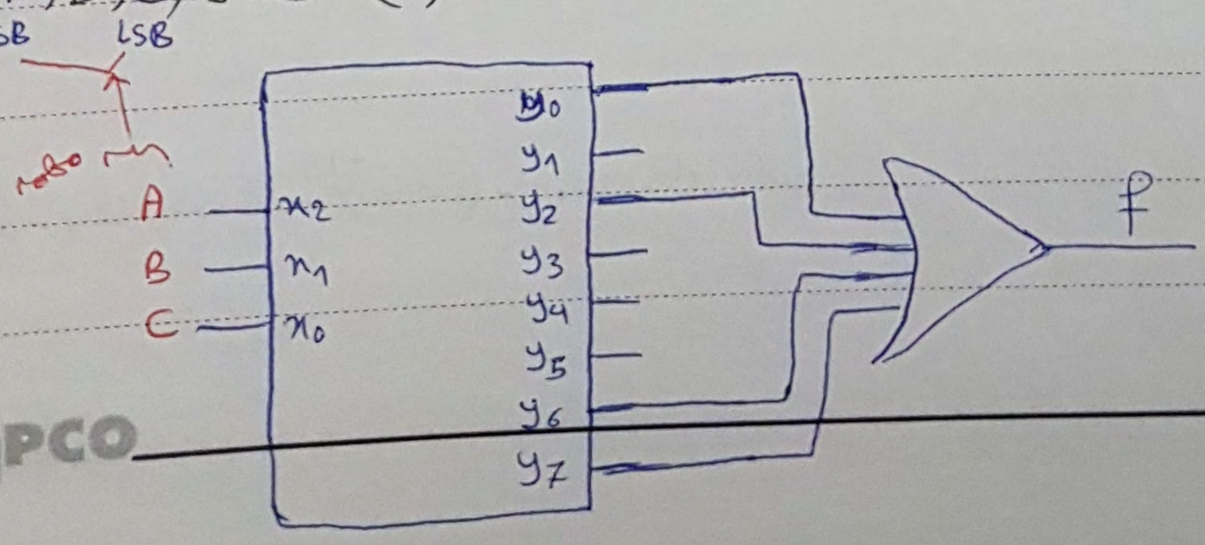


در این تراشه با توجه به کد دودویی ورودی، خروجی دارای شماره متناظر فعال شده و مابقی غیرفعال خواهند بود

n ورودی و 2^n خروجی

مثال) می توان هر تابع منطقی 3 متغیره را به کمک دکودر فوق تهیه نمود

$f(A, B, C) = \sum m(0, 2, 6, 7)$



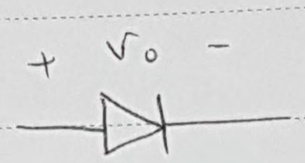
Subject
Date

دکودر BCD to 7S

همان مداری ست که ورودی BCD 4 بیتی را دریافت می‌کند و 7 خروجی کنترلی LED های 7s را ایجاد می‌کند

منطق مثبت و منطق منفی در برخی موارد فعال شدن سیستم به معنای یک بودن سطح منطقی (منطق مثبت) می‌باشد و در برخی موارد فعال شدن آن به معنای صفر بودن سطح منطقی (منطق منفی) است

Ligh Emitting Diode (LED)

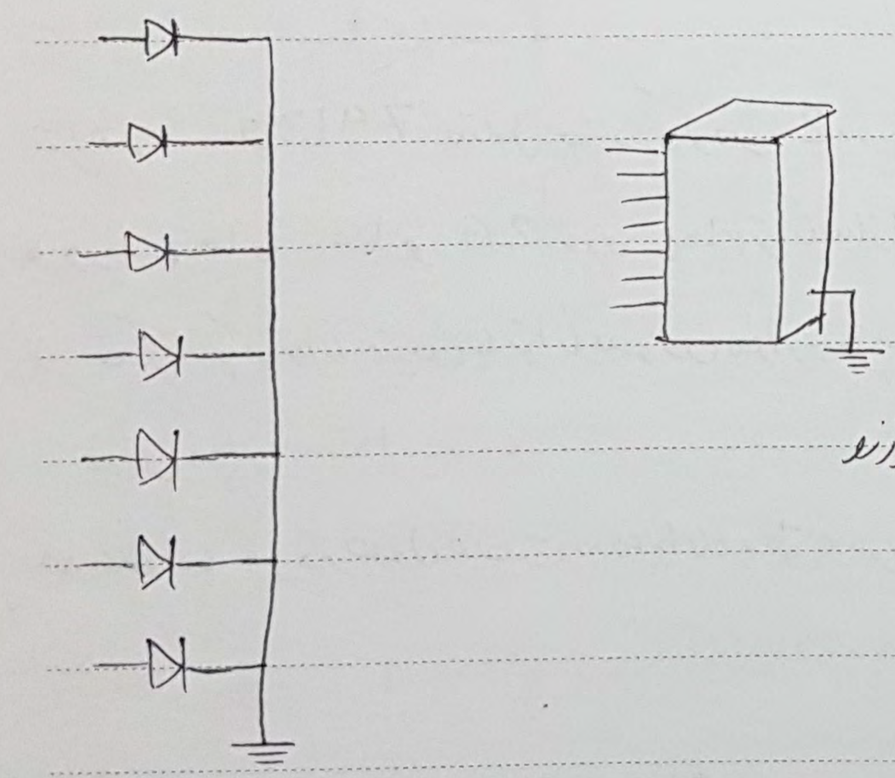


آند مشترک (قطب مثبت مشترک)

کاتد مشترک (قطب منفی مشترک)

} 7s

کاتد مشترک:

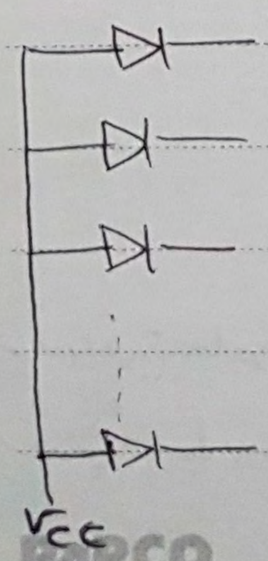


برای روشن نمودن LED های کاتد مشترک بایستی در ورودی از منطق مثبت استفاده کنیم

باید با مقاومت جریان رو کنترل کنیم و مستقیم 5 به آنها ولتاژ ندهیم که می‌سوزند

آند مشترک

برای روشن شدن هر LED بایستی به کاتد آن ولتاژ پایین (سطح صفر) متصل شود (منطق منفی)

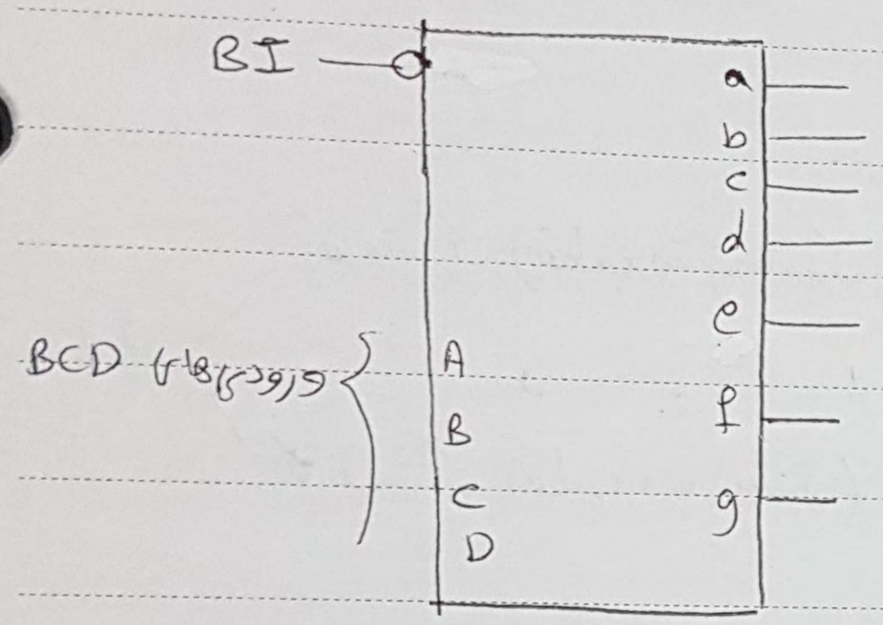


PAPCO

Subject
Date

Positive Logic (Active High)
Negative Logic (Active Low)

تراشه 7447 دکودر BCD 2 7s متناظر با آنمشترک است (منطق منفی)
" 7448 " " " " " " کاتد " (منطق مثبت)



ورودی های منبع تغذیه در ادامه نشان داده نمی شود

BI = Blanking Input

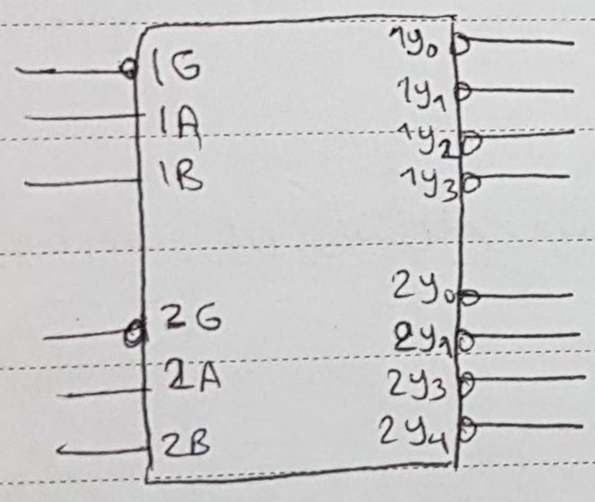
در تراشه 7448 ورودی BI دارای منطق منفی است

← برای خاموش بودن 7s یعنی صفر بودن تمامی خروجی ها (منطق مثبت)

← BI = 0 باید اعمال شود

تراشه 74139 شامل دو عدد دکودر 2 به 4 است

* ورودی های 1G و 2G ورودی های توانا ساز (Enable) می باشند
* دقت کنیم که توانا نمودن یا توانا ننمودن مدار به هیچ وجه به معنای روشن و یا خاموش نمودن مدار نیست!
* علامت بار در جدول صحت به معنای منطق منفی یا به مورد نظر است



با توجه به جدول صحت

MSB LSB
B A

$\overline{G}$ (با اهمیت تر)	B	A	$\overline{y_0}$	$\overline{y_1}$	$\overline{y_2}$	$\overline{y_3}$
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	x	x	1	1	1	1

ورودی های بی اثر در شرایطی که مدار توان است

مطابق اطلاعات سازنده در این شرایط تمامی خروجی ها غیرفعال اند

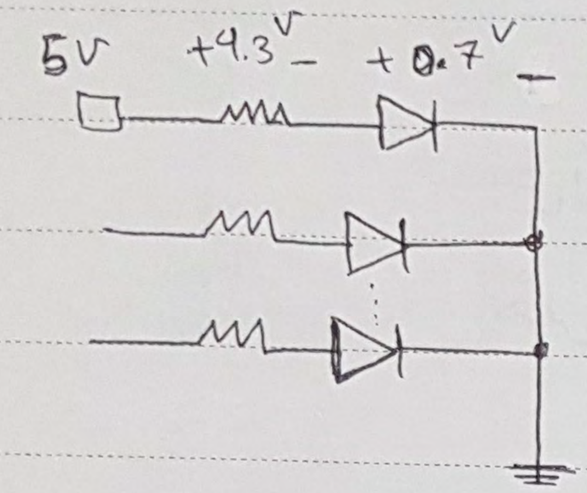
بی اهمیت ها در حالتی است که ورودی ایجاد نمی شود و خروجی برامون اهمیت نداشت

در این شرایط ورودی ها بی اثر هستند نه بی اهمیت

Subject
Date

ملاحظات عملی استفاده از LED

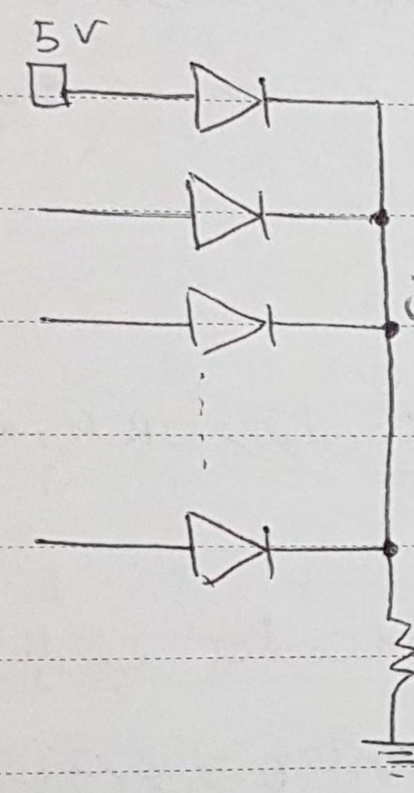
برای جلوگیری از سوختن این چراغها بایستی جریان گذرنده از آنها (ولتاژ اعمال شده به آنها) را کنترل نمود.



$$V_D \simeq 0.7^V$$

$$I_D = I_R = \frac{5-0.7}{R}$$

راه دوم:



فقط یک مقاومت داریم ولی جریان ها متفاوتند ← نورها متفاوتند

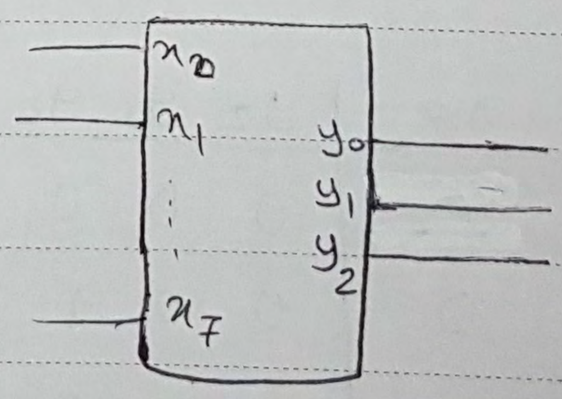
یعنی برای نمایش 1 که LED L2 می‌خواد پرنورتر از 8 است که 7 تا روشنش

$$I_1 = \frac{1}{2} I$$

$$I_8 = \frac{1}{7} I$$

انکودر Encoder رمزگذار

انکودر دودویی:



در CPU بسیار کاربرد دارد

در این انکودر شماره تک ورودی فعال شده در خروجی آشکار می‌شود.

فرض اینکه بیش از یک ورودی فعال شده نخواهیم داشت.

شرایطی که در آن بیش از یک ورودی فعال شده باشد بی اهمیت است

PAPCO

Subject
Date

عملکرد تراشه

x_0	x_1	.....	x_7	y_2	y_1	y_0	
1	0	.....	0	0	0	0	
0	1	.....	0	0	0	1	
0	0	.....	1	1	1	1	
1	1	0	0	d	d	d	بیش از یک ورودی فعال
0	0	.....	0	d	d	d	همگی غیرفعال

$y_0 = x_1 + x_3 + x_5 + x_7$

$y_1 = x_2 + x_3 + x_6 + x_7$

$y_2 = x_4 + x_5 + x_6 + x_7$

$\Leftarrow \begin{cases} y_0 = 1 : \text{عدد ورودی فعال شده فرد بوده است} \\ y_0 = 0 : \text{" " " " زوج " "} \end{cases}$

پس با سه تا OR می‌توان دیکودر دودویی ساده ساخت

دیکودر اولویت priority Encoder

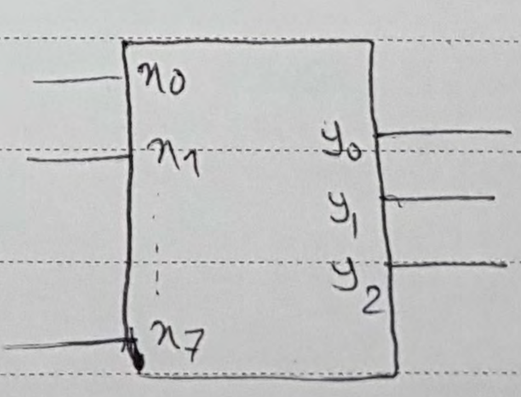
در این تراشه ورودی‌ها مطابق ترتیب شماره خود اولویت بندی شده اند

بنابراین استفاده کننده بایستی ورودی‌های مدنظر خود را بر اساس تراشه فوق به ترتیب متصل نماید

بنابراین در خروجی شماره ورودی فعال شده با بالاترین اولویت آشکار خواهد شد

x_0	x_1	x_2	.....	x_7	y_2	y_1	y_0
1	1	0	.....	0	0	0	0
0	1	1	0	0	0	0	1
0	1	0	.....	1	0	0	1
0	0	0	.....	0	d	d	d

فرض: x_0 بالاترین الویت

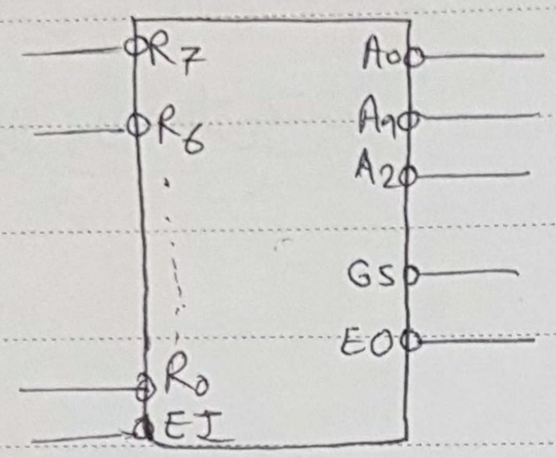


Subject
Date

ترانشه 74148 چون 74 تکرار زیاد می‌شه به صورت 148، هم نشون می‌دن اینو

انکودر الویت 8 ورودی

الویت با شماره‌های بالاتر

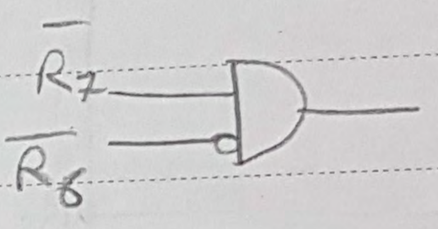


EI : Enable Input

EO : Enable Output

GO : Group Select

$\overline{EI}$	$\overline{R_0}$	$\overline{R_1}$	$\overline{R_2}$	$\overline{R_3}$	$\overline{R_4}$	$\overline{R_5}$	$\overline{R_6}$	$\overline{R_7}$	$\overline{A_2}$	$\overline{A_1}$	$\overline{A_0}$	$\overline{GS}$	$\overline{EO}$
1	x	x	x	x	x	x	x	x	1	1	1	1	1
0	x	x	x	x	x	x	x	0	0	0	0	0	1
0	x	x	x	x	x	x	0	1	0	0	1	0	1
⋮							⋮	⋮		⋮		⋮	
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0



EI : پایه فعال سازی (توانا سازی) عملیات مورد نظر ترانشه می‌باشد (با خاموش بودن ترانشه متفاوت است)
قطع منبع تغذیه
خروجی مدار (ترانشه) در این شرایط توسط شرکت سازنده تعیین می‌شود

GS : ترانشه فعال بوده است و یکی از ورودی‌ها (حداقل) فعال بوده است

EO : ترانشه فعال باشد ولی هیچ کدام از ورودی‌ها فعال نباشند

PAPCO

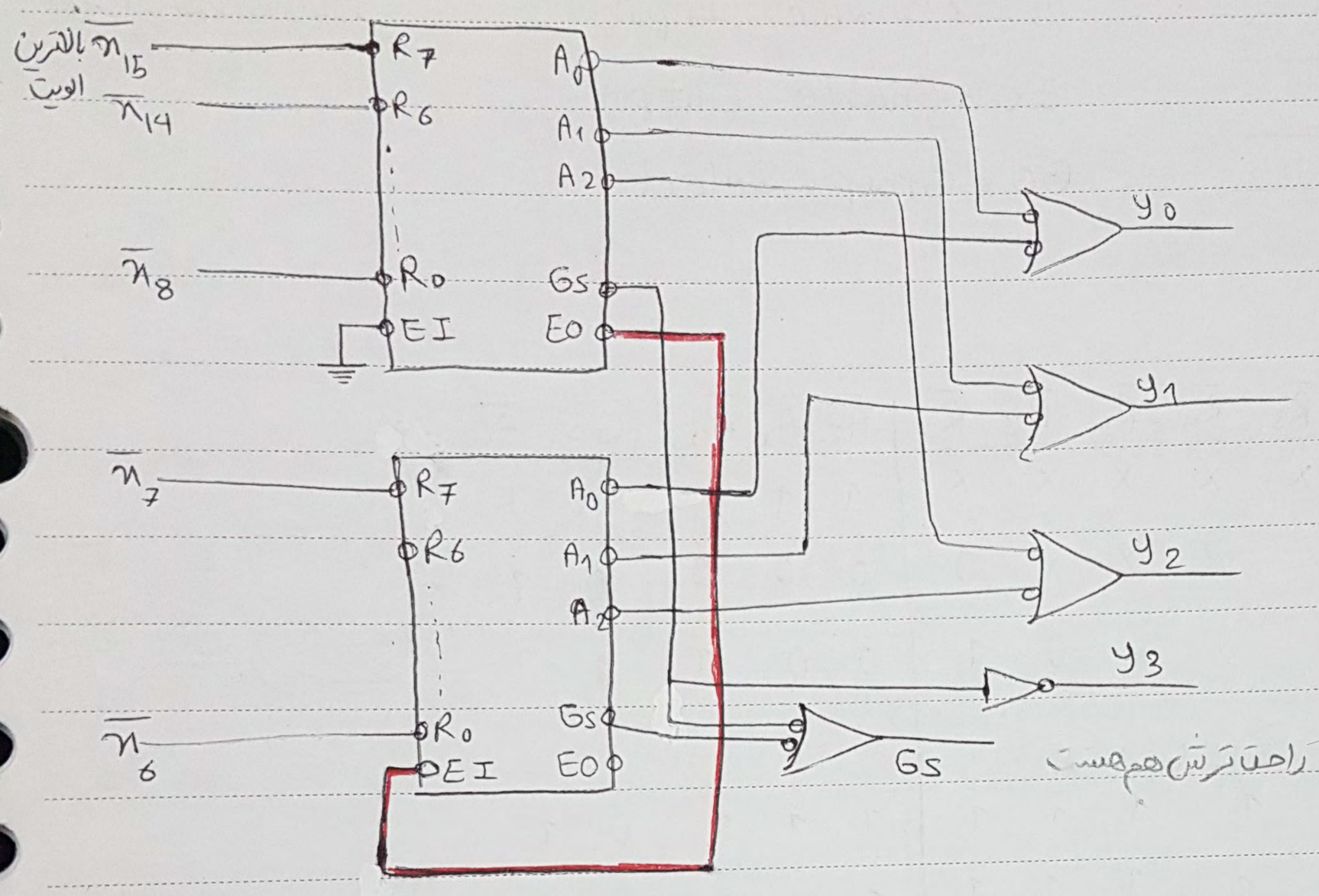
Subject

Date

طراحی انکودر اولویت با 16 ورودی

با کمک تراشه های 148

$y_3\ y_2\ y_1\ y_0$ منطق مثبت

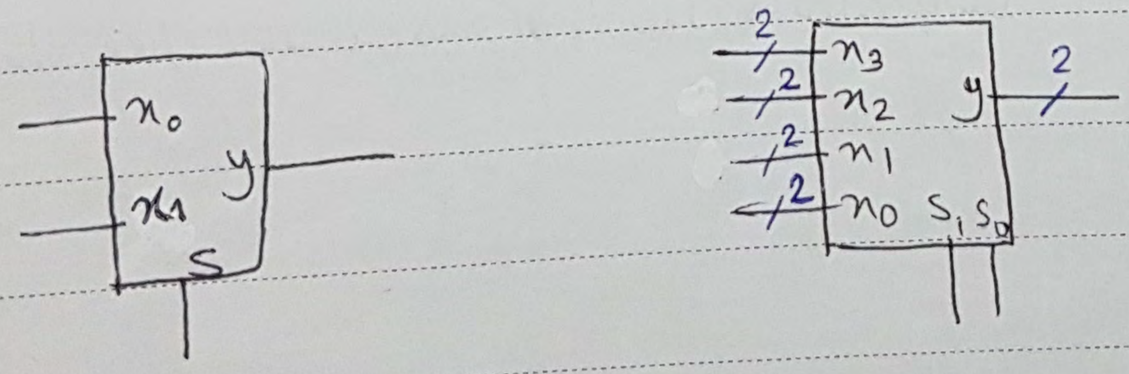


سیستم فوق همواره فعال است چون $\overline{EI}$ در تراشه بالا همواره 0 داده شده است

مالتی پلکسر (Multiplexer) ← MUX مثل اسپیکر که انتخاب می کنیم کدام صدا پخش شود

توسط این تراشه انتخاب می شود که کدام ورودی در خروجی قرار گیرد

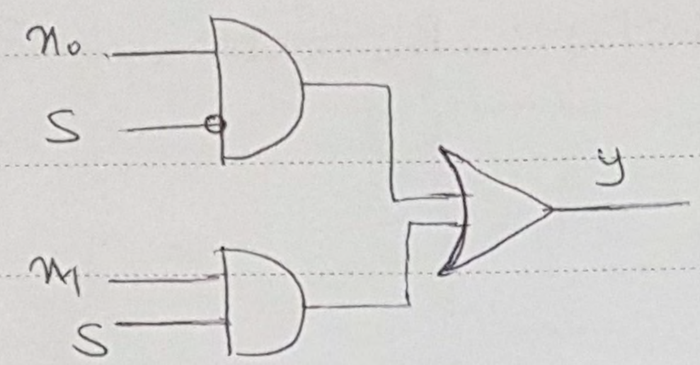
با بیت های select انتخاب می کنیم که کدام ورودی در خروجی قرار گیرد



با کمک بیت های انتخاب S_0 و S_1 مشخص می شود که کدام دسته 2 بیتی ورودی در خروجی 2 بیتی قرار گیرد

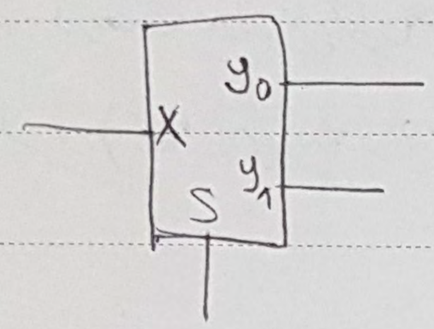
Subject
Date

عملکرد مالتی پلکسر



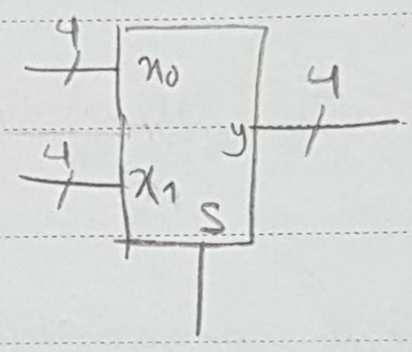
دی مالتی پلکسر Demultiplexer (DMUX)

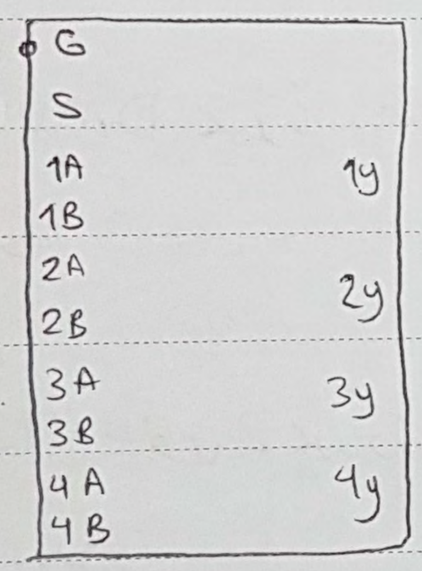
توسط این تراشه انتخاب می‌کنیم که ورودی در کدام خروجی قرار گیرد



تراشه '157

یک مالتی پلکسر 2 به 1 4 بیتی است



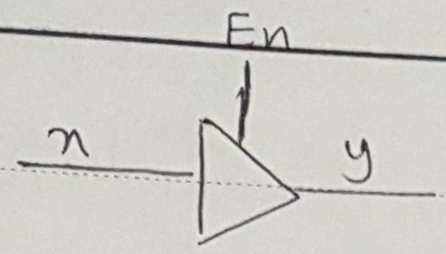


1A, 2A, 3A, 4A
1B, 2B, 3B, 4B
1y, 2y, 3y, 4y

ورودی $\bar{G}$ مجدداً مشابه به توانساز عمل می‌نماید

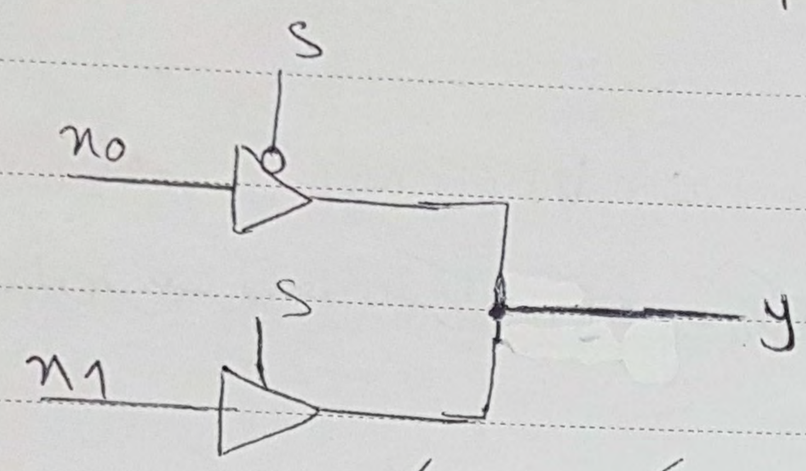
$\bar{G}$	S	y
1	x	0
0	0	A
0	1	B

Subject
Date



En	y
1	x
0	Hi-Z (مدار باز)

بافر 3 حالته 3state_Buffer



مالتی پلکسر 2 به 1 تک بیتی

با طراحی درست مدار می توان خروجی های این گیت ها را به یکدیگر متصل نمود (عملیاتی که در هیچ گیت دیگری قابل انجام نیست)

ترانشه '251

مالتی پلکسری مشابه '151 با خروجی های بافر 3 حالته است

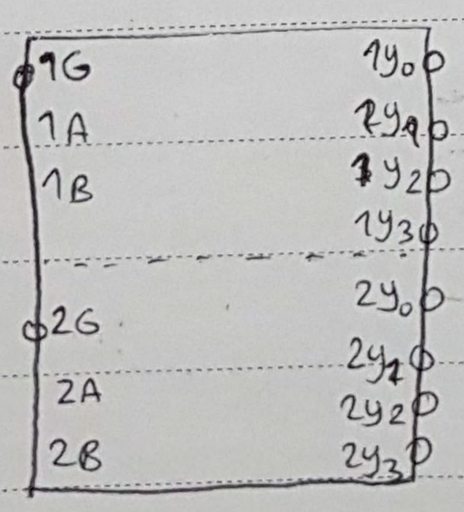
ترانشه DMUX عموماً به صورت مستقل ساخته نمی شود بلکه به صورت ترکیبی دکودر / دی مالتی پلکسر تهیه می شود

ترانشه '139 حاوی 2 عدد دکودر 2 به 4 است که می توان از آن به صورت 2 عدد دی مالتی پلکسر 1 به 4 تک بیتی نیز استفاده نمود

دکودره

$\overline{G}$	B	A	$\overline{y_0}$	$\overline{y_1}$	$\overline{y_2}$	y_3
0	0	1	1	0	1	1
1	x	x	1	1	1	1

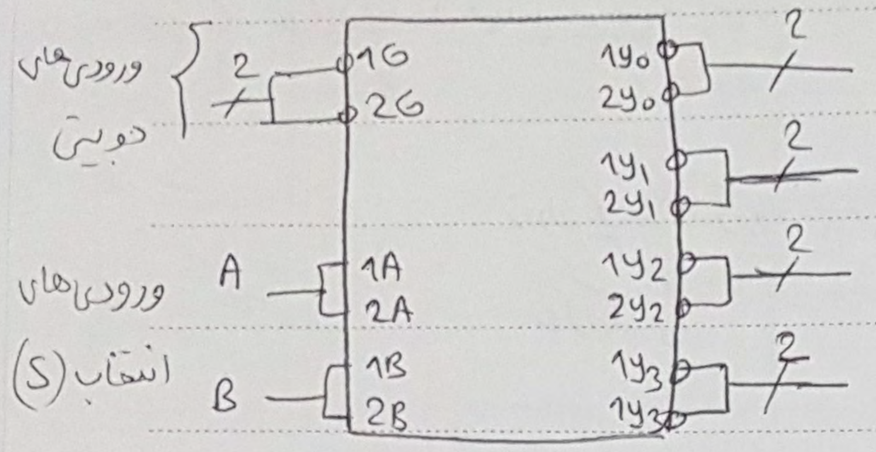
شرکت سازنده می آید:



PAPCO

Subject
Date

دی مالتی پلکسر تراشه صفحه



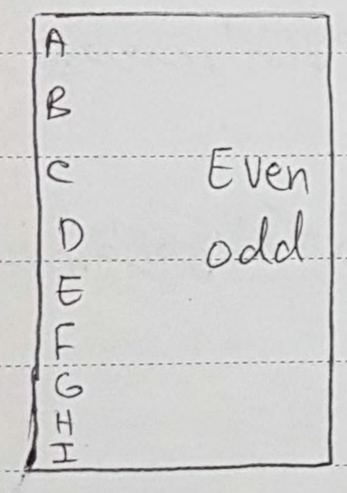
G	B	A	$\overline{y_0}$	$\overline{y_1}$	$\overline{y_2}$	$\overline{y_3}$
0	0	1	1	0	1	1
1	0	1	1	1	1	1

همونا که سازنده می گه:

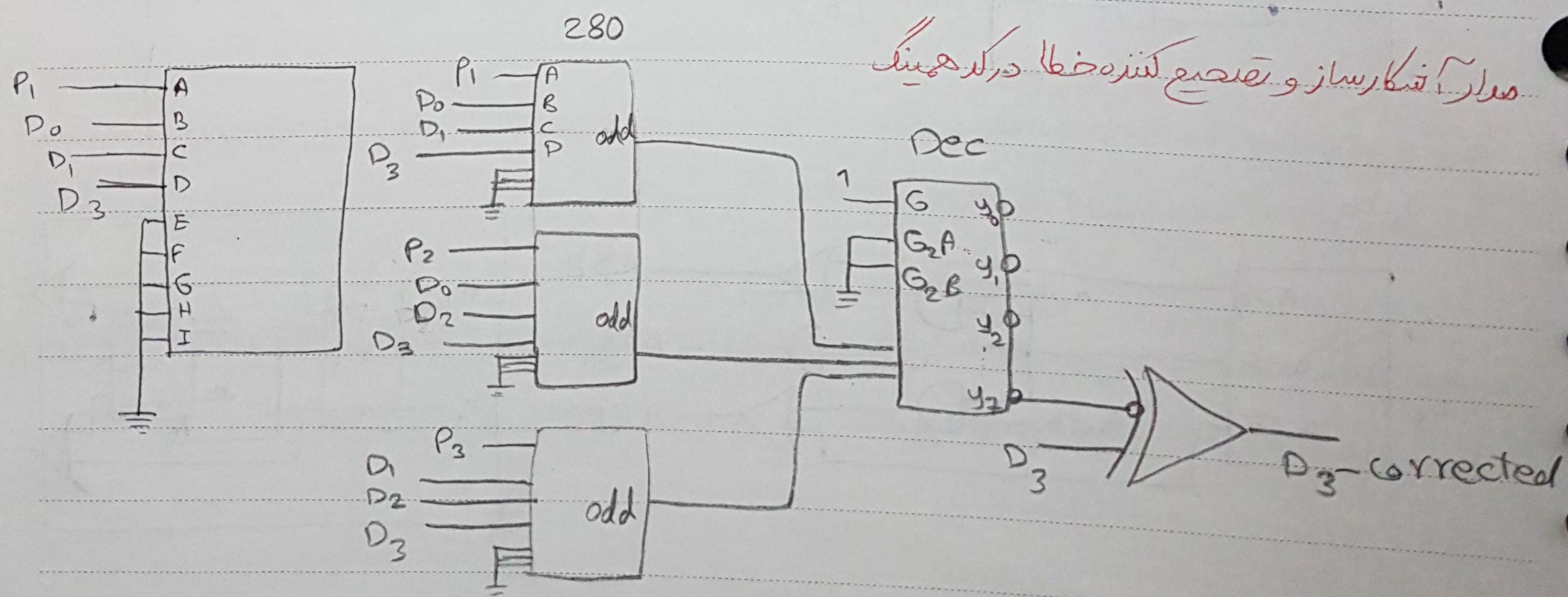
با کمک بیت های انتخاب B A تصمیم می گیریم که ورودی اصلی $\overline{G}$ در کدام خروجی قرار گیرد

تراشه 280

چک کننده و تولید کننده پرتی با 9 ورودی باشد

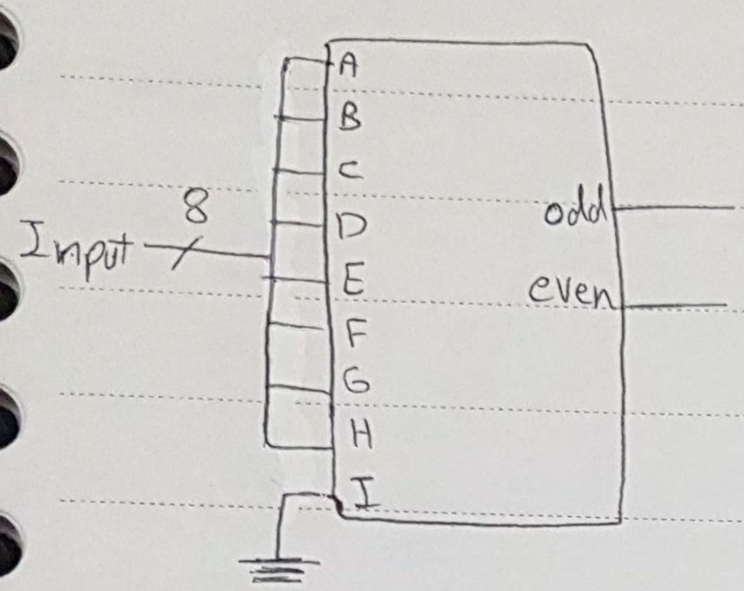


مدار آشکارساز و تصحیح کننده خطا در کد همینگ



Subject
Date

مثال: تولید بیت فهم پرتی برای بایت ورودی



odd = 1 تعداد یک های ورودی فرد

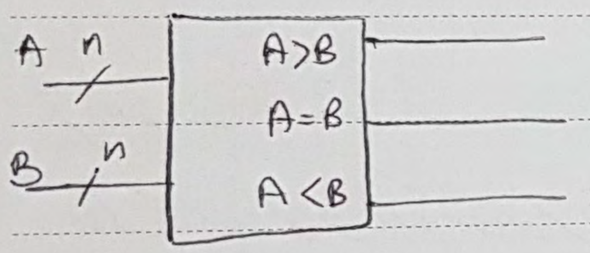
odd = 0 بالعکس

اگر بخواهیم پرتی فرد تولید کنیم، بیت پرتی حاصل را از خروجی even تهیه می کنیم و بالعکس

اما اگر بخواهیم پرتی ورودی ها را چک کنیم، تمامی بیت ها (بایت + p) را به ورودی متصل نموده

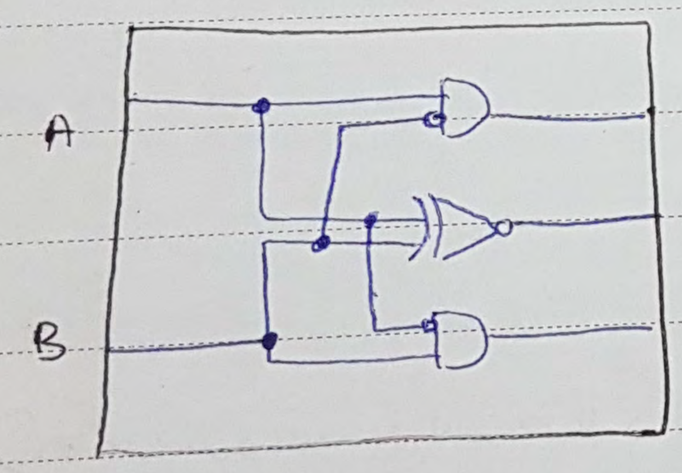
و در خروجی { چک نمودن پرتی فرد: odd = 1 پرتی ورودی فرد بوده است
چک نمودن پرتی زوج even = 1 پرتی ورودی زوج بوده است

مقایسه کننده ها comparators



دو عدد در ورودی اعمال می شود و اندازه این دو عدد مقایسه خواهد شد

مثال: مداری طراحی کنید که دو عدد تک بیتی را با هم مقایسه کنید



A>B { A = 1, B = 0

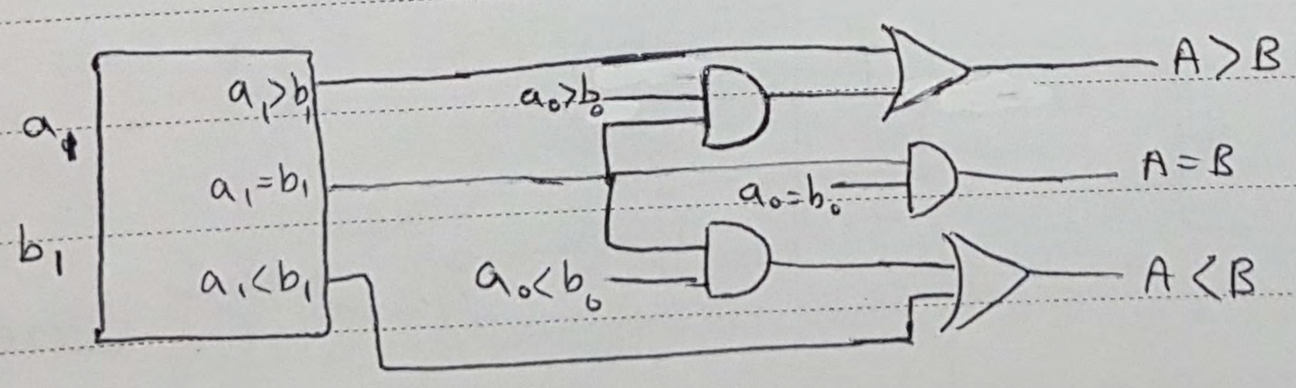
A = B

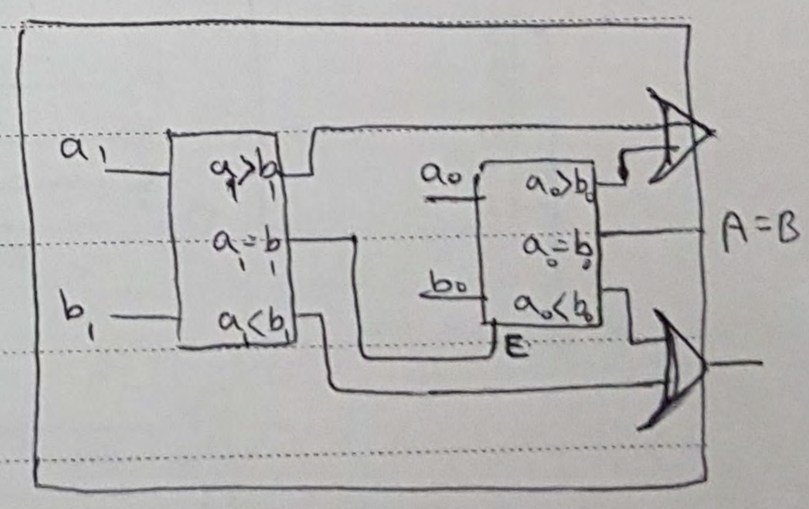
A<B { A = 0, B = 1

روش مقایسه دو عددی های چند بیتی

$A = a_1 a_0$

$B = b_1 b_0$

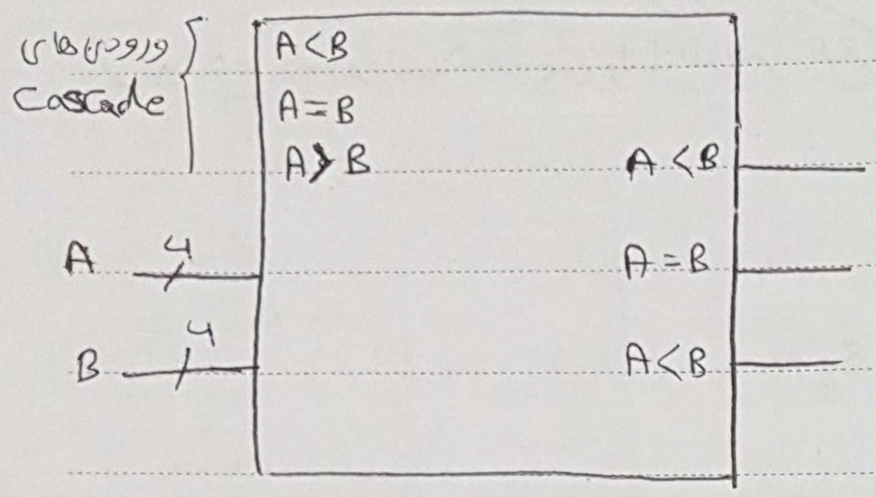




PAPCO

Subject:
Date

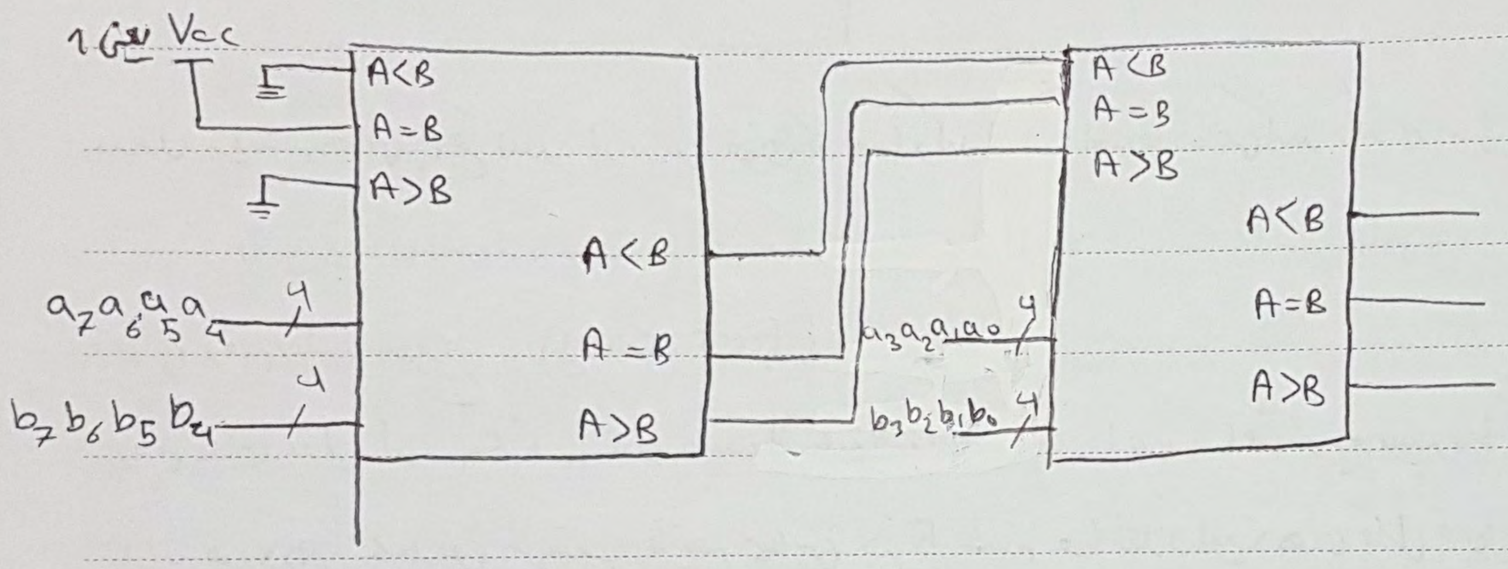
تراشه مقایسه‌گر چهاربیتی '85



مثال: اگر بخواهیم دو بایت A و B را با هم دیگر مقایسه کنیم:

$A = a_7 a_6 \dots a_0$

$B = b_7 b_6 \dots b_0$



در ورودی های Cascade در صورت فعال شدن A>B و A<B باعث می‌شوند که تراشه عملیات مقایسه را انجام ندهد و تنها به خروجی انتقال دهد

امکان ندارد 2 تا خروجی فعال شود

مدارهای جمع / تفریق کننده

مدارهای نیم‌جمع کننده بدون رقم نقلی ورودی HA

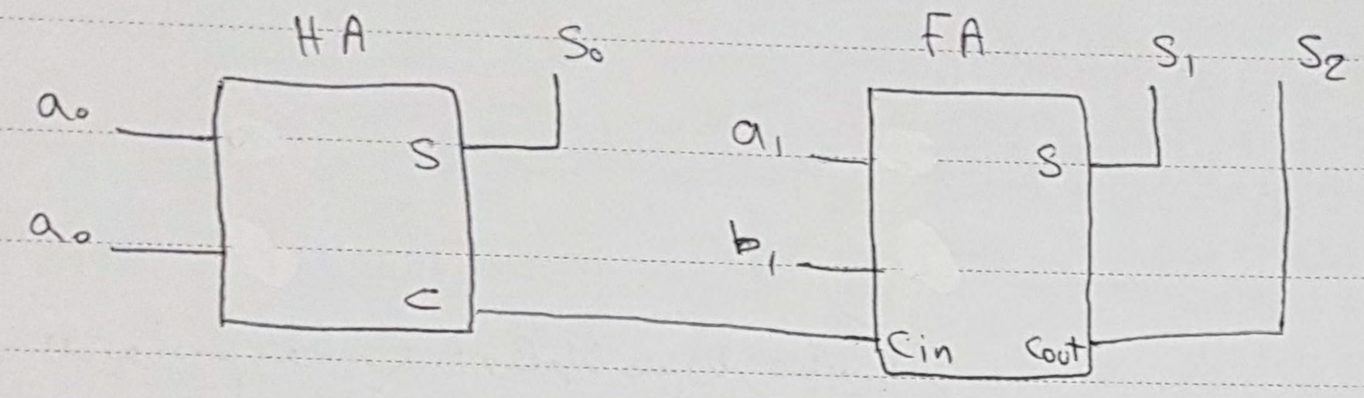
مدارهای تمام جمع کننده همراه با رقم نقلی ورودی FA

می‌توان با کمک مبنای متمم دو منهای یک، عملیات تفریق را به عملیات جمع تبدیل نمود. با در یکی از ورودی‌ها از منطق منفی (گیت های NOT) استفاده کنیم تا مبنا را تغییر دهیم و ...

Subject:
Date

اگر بخواهیم دو ورودی چند بیتی را با یکدیگر جمع کنیم، می‌توان از جمع کننده های تک بیتی کمک گرفت

$a_1 a_0 + b_1 b_0 \longrightarrow S_2 S_1 S_0$



روش فوق در جمع اعداد n بیتی روشی بسیار کند است چون تعداد طبقات از a_0 تا S_{n+1} بسیار زیاد است

در عمل تراشه های جمع کننده از روش فوق که بسیار کند است استفاده نمی‌کنند بلکه به سراغ روشهایی مشابه SOP رفته اند

روش رقم نقلی سریع (Fast carry)

رقم های نقلی از C_1 تا C_n را مستقلاً با توجه با ورودی ها بدست می‌آوریم سپس با کمک FA های تک بیتی حاصل جمع نهایی را ارائه می‌دهیم، بدین ترتیب نتیجه نهایی در 5 طبقه و با تعداد گیت های قابل قبول بدست می‌آید

تراشه '82 جمع کننده کامل دو بیتی

تراشه '83 4 FA بیتی متوالی

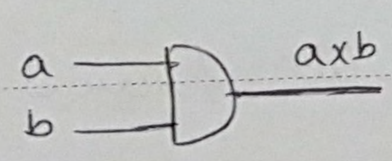
تراشه '182 تولید کننده رقم نقلی سریع

تراشه '283 جمع کننده کامل 4 بیتی بر اساس رقم نقلی سریع

تمرین: با کمک جمع کننده های دودوئی، جمع کننده BCD بسازید $BCD_1 + BCD_2 = (BCD_M , BCD_L)$

ضرب کننده ها

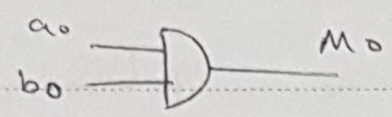
$a \times b \xrightarrow{\text{تک بیتی}}$



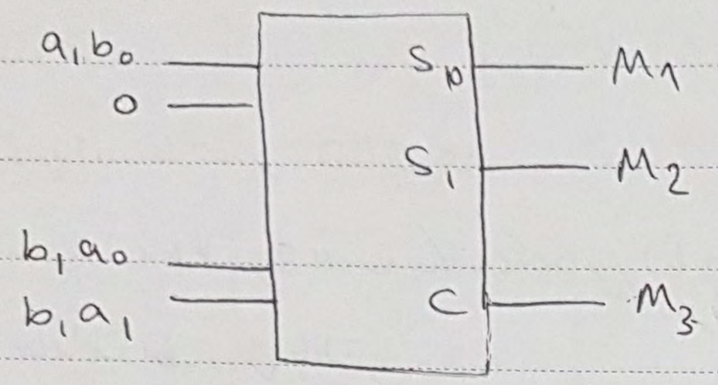
$$
\begin{array}{rrrr}
 & & a_1 & a_0 \\
 & \times & b_1 & b_0 \\
\hline
 & & b_0 a_1 & b_0 a_0 \\
+ & b_1 a_1 & b_1 a_0 & \\
\hline
 & b_1 a_1 & b_0 a_1 + b_1 a_0 & b_0 a_0 \\
M_3 & M_2 & M_1 & M_0
\end{array}
$$

PAPCO

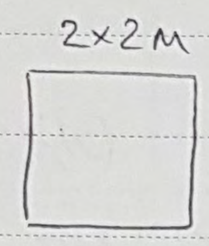
Subject:
Date



جمع کننده 2 بیتی



$$\begin{array}{rrrrr} & & a_1 & a_0 & \\ \times & & b_1 & b_0 & \\ \hline & & b_0a_1 & b_0a_0 & \\ + & b_1a_1 & b_1a_0 & & \\ \hline M_3 & M_2 & M_1 & M_0 & \end{array}$$

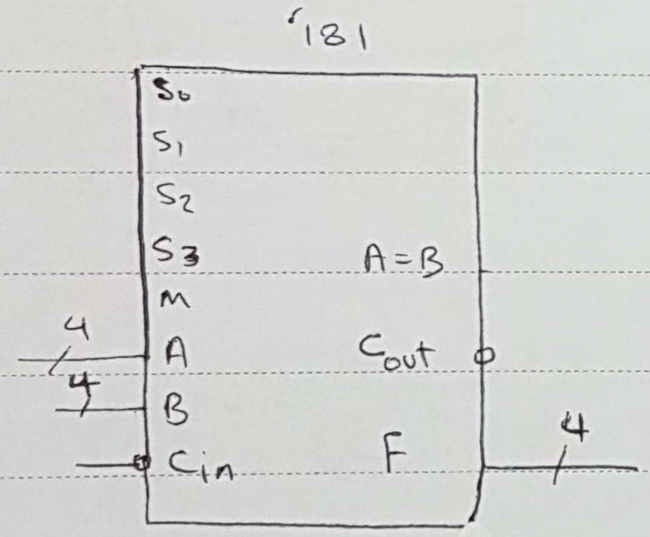


تمرین: با استفاده از ضرب کننده 2 بیتی فوق ضرب کننده 4 بیتی بسازید

واحد محاسباتی و منطقی Arithmatic - Logic - Unit (ALU) ← در امتحان نمی‌آید

در این تراشه‌ها مدارهای مختلف توسط مالتی پلکسر به خروجی واحد انتقال می‌یابد

تراشه '181 ALU 4 بیتی

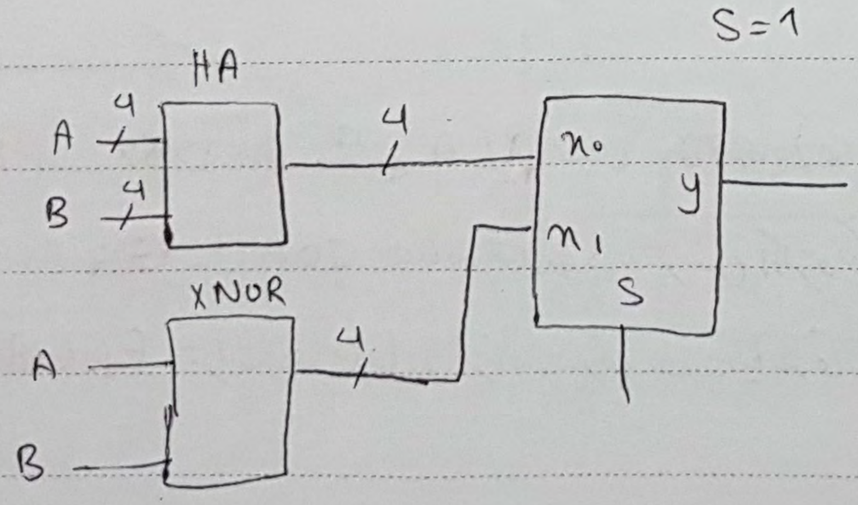


$S_3S_2S_1S_0$	$M=0$	$M=1$
0000	$F=A \text{ plus } \overline{C}_{in}$	$F=\overline{A}$
⋮		
1001	$F=A \text{ plus } B \text{ plus } C_{in}$	$F=A \oplus B$
⋮		

مثال: ALU ای بسازید که A و B چهار بیتی باشند و

$S=0$ A plus B

$S=1$ $A \odot B$



PAPCO

Subject:
Date

تراشه های PLD

Programmable Logic Devic

این تراشه ها قابلیت انجام عملیات دلخواه (قابل اجرا) طراح را دارد

مزایای استفاده از PLD ها

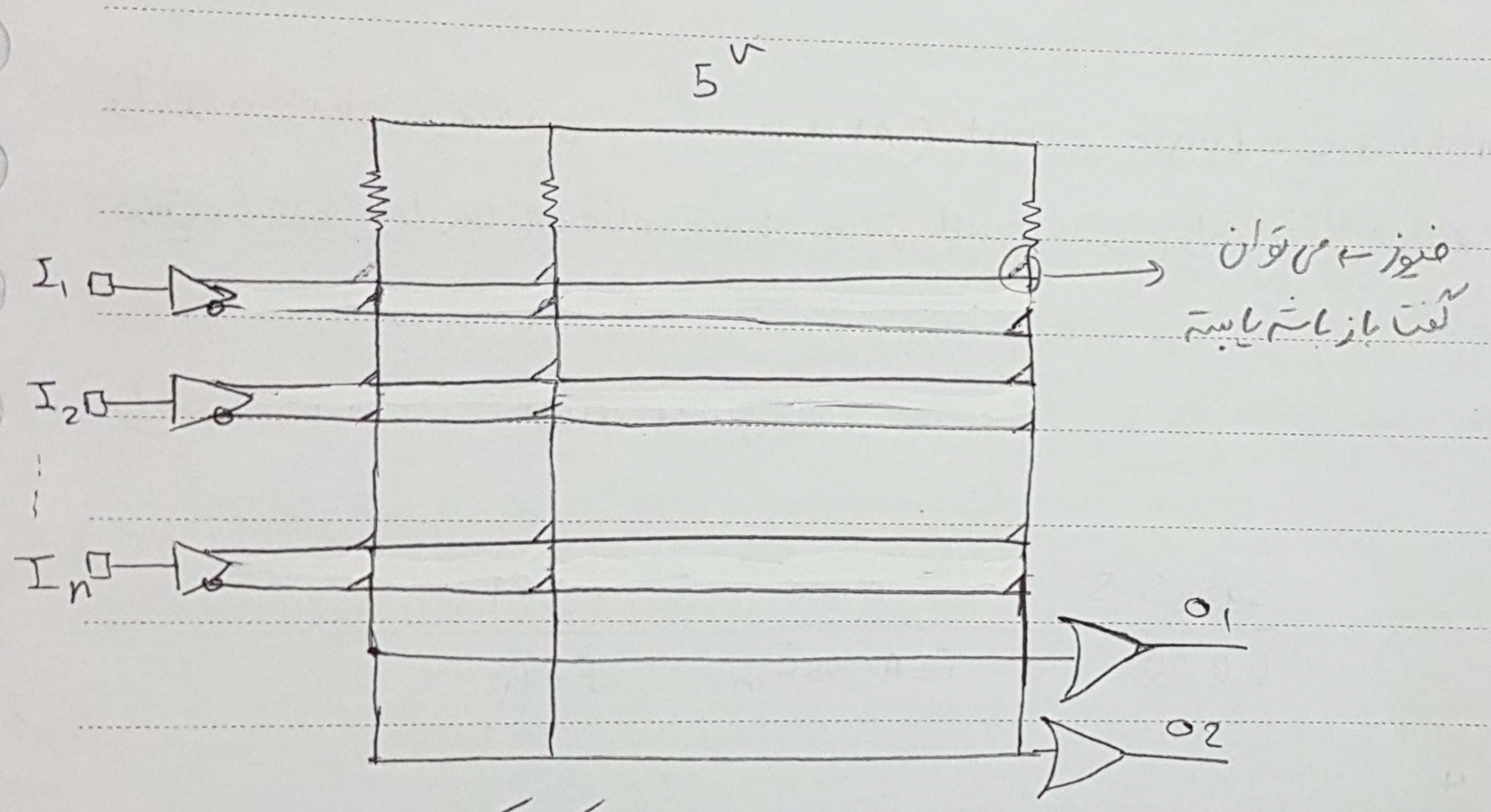
1- سرعت بالاتر: نسبت به کاری که داریم انجام میدهیم برای مدارهای ساده 2 طبقه مورد استفاده الان ما نیست

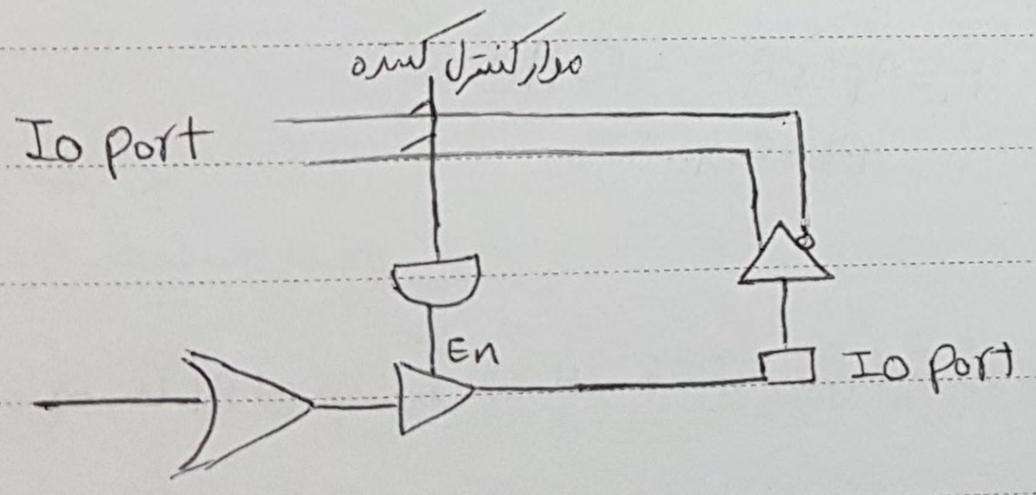
2- قابلیت اطمینان بالاتر

3- هزینه کمتر

4- طراحی سریع (توسط نرم افزار)

5- قابلیت تغییر کاربری (پاک نمود و مجدداً برنامه ریزی کرد)





برخی از پایه ها هم ورودی و هم خروجی می باشند IO port

PROM 1969 سال بعد ← PLA سال 1970 (تمامی مین ترم ها تولید می شوند) ← PAL 1978 (تمامی مین ترم ها را تولید نمی کند اما ارزان تر است)

← GAL IO port اضافه شد + دسترسی الکترونیکی به برنامه ریزی

← FPGA تراشه های قابل برنامه ریزی روز دنیا که قابلیت اجرای توابع مختلف با بالاترین سرعت های موجود در آنها وجود دارد