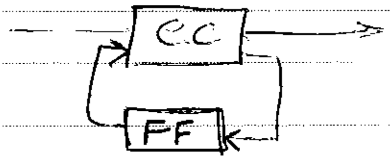
[illegible] ثانیه Hz

Subject: .... Year: .... Month: .... Date: ....

در بعضی وقت دو مدار ترکیبی F و T را باهم می‌کشند و ساعت کلی مدار ترکیبی را به شکل زیر می‌کشند.



سال ۸۹ تست ۷۰

$\cancel{d_{ff}} + d_{max} + t_s < T$

$\emptyset + 10ns + 15ns < T$

$25ns < T \Rightarrow \frac{1}{25} = \boxed{\frac{1}{T}} f$

$\frac{1}{25ns} = \frac{1}{25\times10^{-9}s} = \frac{1}{25\times10^{-9}} = \frac{10^{9}}{25} Hz$

$= \frac{100\times10^{7}}{25} = 4\times10^{7} Hz = 40\times10^{6} Hz = 40 MHz$ گزینه ۳

تأخیر در انتشار یعنی تأخیر مدار ترکیبی یعنی d_{max} یا t_{pd}.
(تأخیر انتشار Propagate delay)

t_{pd} برای مدار ترکیبی است.

Subject: .... Year: .... Month: .... Date: ....

تنظیم در حالت hold time

d_{ff} معمولاً بیشتر از t_h است و رابطه $d_{ff}+d_{min}>t_h$ است اما اگر d_{ff} کمتر از t_h بود باید کاری کنیم که تأخیر d_{min} بیشتر باشد باید کاری کنیم که عمداً این تأخیر زیادشه با گذاشتن گیت الکی یا مقاومت و خازن الکی.

$d_{ff} + d_{max} < \overset{\text{پریود کلاک}}{T} - t_s$

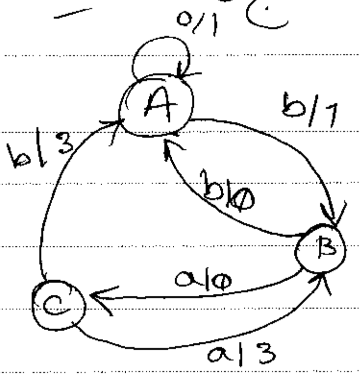
$d_{ff} + d_{max} + t_s < T$ ✓

این را همیشه رعایت می‌کنیم چون مدارمان دیگر ۲ طبقه است چی کار کنیم!! پس فقط مجبوریم پریود کلاک را زیاد کنیم.

پریود کلاک بیشتر از تأخیر مدار ترکیبی (dmax) و مجموع $d_{ff}+t_s$ هم بیشتر باشد.

Subject: .... Year: .... Month: .... Date: ....

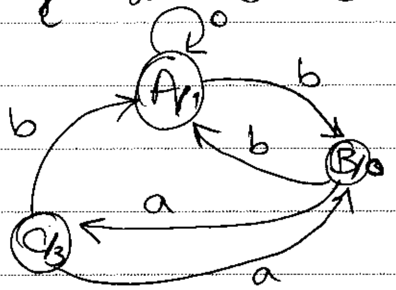
در Moore یالهایی که از یک State خارج می‌شوند باید خروجی یکسان داشته باشند.



پس این یک Moore است.

خروجی فعلی ربطی به ورودی فعلی ندارد.

که می‌آییم خروجی را داخل همان State ها می‌نویسیم



خروجی / حالت

Moore

در مدار Moore هیچ مسیری بین O و i ندارد
خروجی ورودی

در مدار Mealy حتماً مسیری بین O و i وجود دارد

تفاوت Moore با Mealy چیست؟

Subject: .... Year: .... Month: .... Date: ....

FSM Mealy و FSM Moore

Mealy همان حالتی قبلی که از FSM داشتیم

$S[n+1]=T(i[n],s[n])$

$O[n]=F_o(i[n],S[n])$

Moore حالت خاصی از Mealy است. خروجی فعلی فقط تابع حالت فعلی است برخلاف Mealy که خروجی فعلی تابع ورودی فعلی و حالت فعلی است.

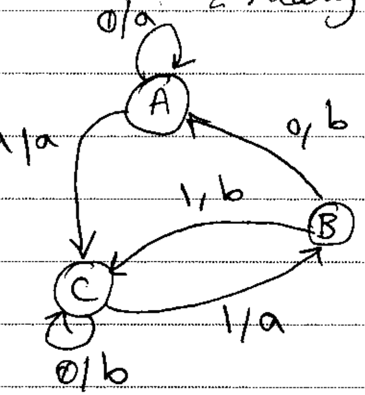
Moore:

$S[n+1]=T(i[n],S[n])$

یا $S[n]=T(i[n-1],S[n-1])$

$O[n]=F_o(S[n])$ خروجی به حالت ربط داره

تعیین کنید Moore است یا Mealy؟



این یک FSM Mealy است

Moore یعنی اگر حالت را بدم خروجی را می‌تونید بگید. / اما اینجا نمی‌تونیم بگیم مثلاً C که هستیم می‌تونیم

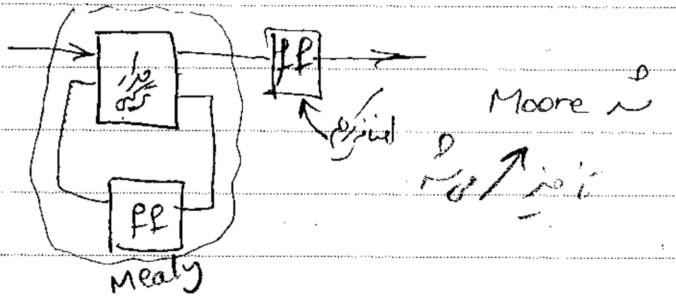
Mealy FSM چگونه به More تبدیل می‌شود؟ آخر کدام بهتر است Mealy یا More

Subject: Year: Month: Date:

نتیجه: هر کسی هر سیستمی بخواهد بسازد با Moore می‌تواند بسازد

اگر FSM، Mealy باشد اگر سر ره یک f.f بگذاریم

تأخیر بیشتر شده اما تبدیل شده به Moore. این اثبات قضیه است



مدار ترکیبی
ff
ff
Mealy
اضافه کردیم

Moore شد

تأخیر ↗ می‌شه

در سیستم‌های دیجیتال هم Moore هستند، که به مراتب نسبت

به Mealy برتری دارد؟

طراحی FSM همانند برنامه‌نویسی است.

کسی می‌تونه FSM را خوب طراحی کنه که به State

معنا ببخشه هر O را مشخص کند چیه؟! بهترین جای

طراحی FSM کار روی State Diagram است.

Subject: Year: Month: Date:

طراحی FSM برای مقایسه کننده سریال

اول reset می‌کنیم و دو عدد مورد نظر یک بیت یک بیت می‌ره داخل مقایسه

می‌شود و خروجی دریافت می‌کنیم.

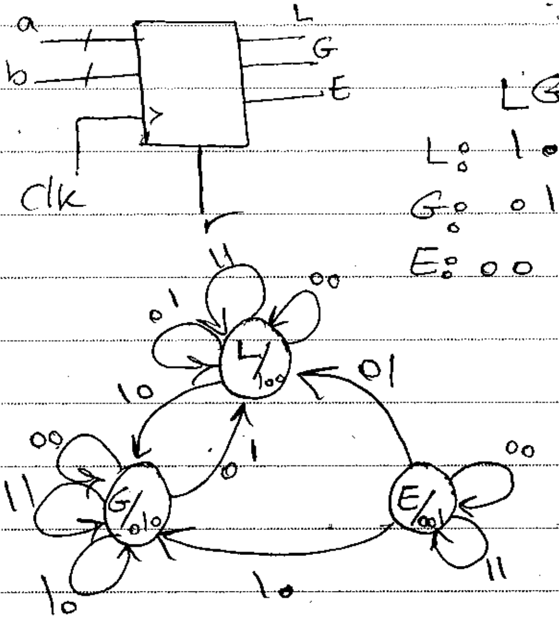
خروجی $L = A < B$

$G = A > B$

$E = A = B$

اول تعیین کنیم از کم ارزش وارد می‌شوند یا پر ارزش؟ فرض کردیم از بیت

کم ارزش وارد می‌شوند.



a
b
clk
L
G
E
L/100
G/010
E/001
11
00
01
10
01
1
0
00
11
10
00
11
10

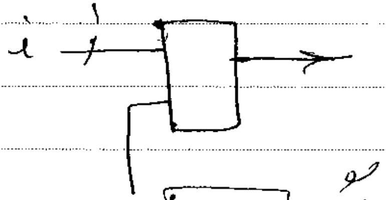
	LGE خروجی‌ها
L:	100
G:	010
E:	001

شروع از E چون

ابتدا اعداد با هم برابرند

مدارهای تشخیص الگو:

یک بیت ورودی دارند. که دو نوع بازشناسی الگو داریم ۱- همپوشانی را قبول دارند ۲- همپوشانی را قبول ندارند.



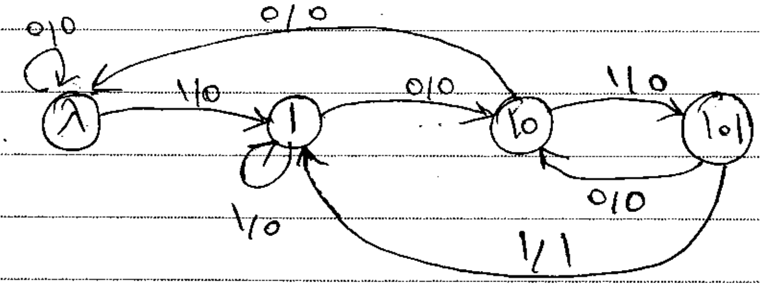
مثال ساخت مدار تشخیص الگو $\boxed{1011}$

0 1 2 3 4 ...

0 1 1 1 0 0 1 0 0 10 11 0 111 0 11 00 11

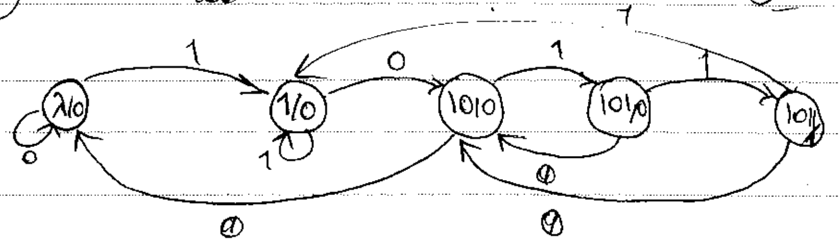
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0

در اینجا همپوشانی را قبول کرده ایم. حال ساخت Mealy


0/0
0/0
1/0
0/0
1/0
λ
1
10
101
1/0
0/0
1/1

خروجی حالت

تشخیص الگوی 1011 برای طراحی Moore


λ/0
1/0
10/0
101/0
1011/1
0
1
1
0
1
1
0
0
0
1

برای طراحی نیاز به ۳ فلیپ فلاپ داریم.

خروجی تابعی از حالت است پس Moore است.

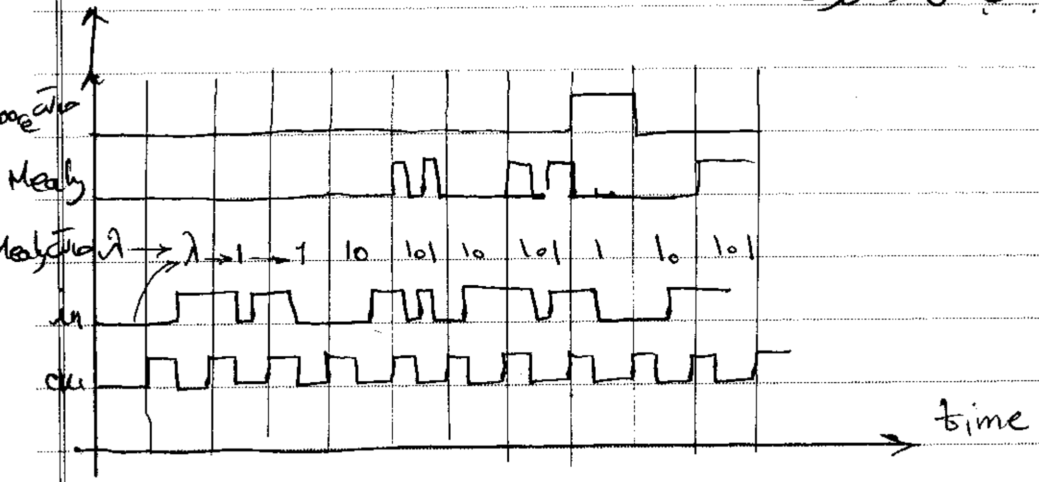
برای شناسایی 1011 دو مدار طراحی کردیم یکی Moore و یکی Mealy است.

Subject: .... Year: .... Month: .... Date: ....

در Mealy چون خروجی تابع ورودی و حالت قبلی است برای همین تغییرات لبه clock حساس نیست و هرجا ورودی تغییر کرده، تغییر می‌کند

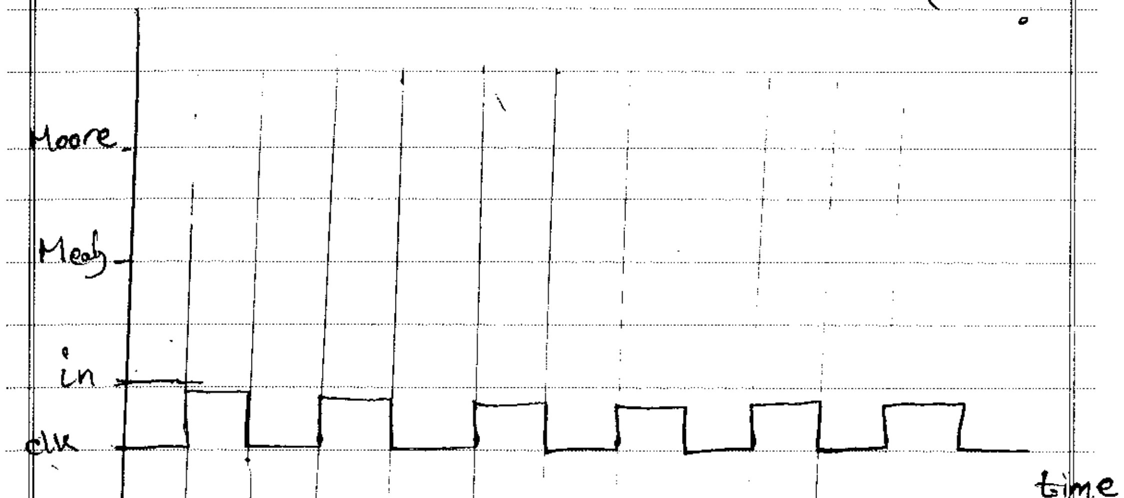
Moore تغییرات فقط در لبه clock است.

* خروجی Moore از معادل Mealy یک clock دیرتر جواب می‌دهد.


حالت Moore
Mealy
حالت Mealy
λ → λ → 1 → 1 10 101 10 101 1 10 101
in
clk
time

Subject: .... Year: .... Month: .... Date: ....

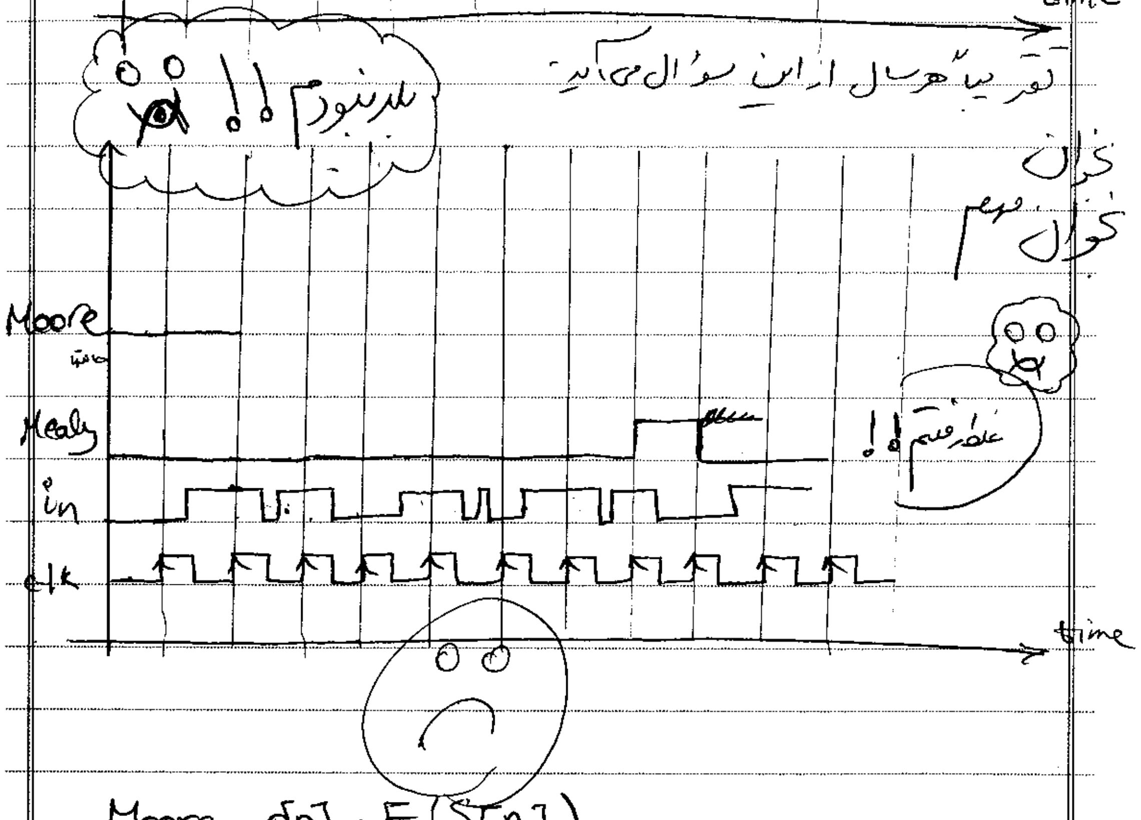
جلسه ششم


Moore
Mealy
in
clk
time

تقریباً هر سال از این سؤال می‌آید

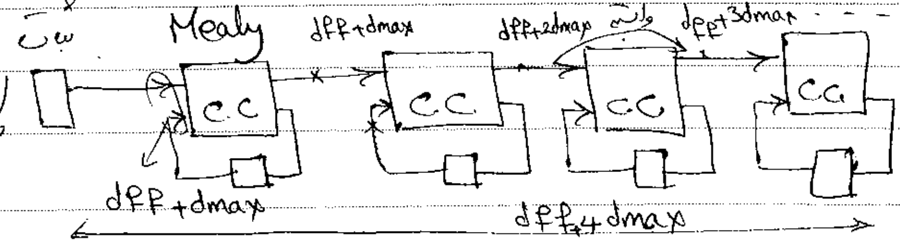
کوئیز

کوئیز مهم


بلد نبودم !!
Moore
حالت
Mealy
in
clk
time
غلط فهمیدم !!

Moore $O[n] = F(S[n])$

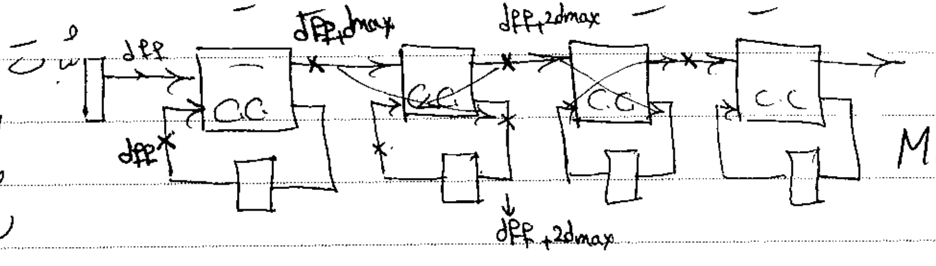
کاهش


ثبات
Mealy
dff+dmax
dff+2dmax
وابسته
dff+3dmax
C.C
C.C
C.C
C.C
dff+dmax
dff+4dmax
Mealy

کلاک بزرگ
سرعت کم

در Mealy چون وابسته به قبلی است همین طور تأخیرها شده پس

در کل خیلی تأخیر می‌ره بالا و کلاک خیلی زیاد می‌شود.


ثبات
dff
dff+dmax
dff+2dmax
C.C
C.C
C.C
C.C
dff
dff+2dmax
Moore

کلاک کوچک
سرعت بیشتر

در کل برای Moore می‌شه $dff+2d_{max}$ است.
با هر تعداد که سری شده بودن.

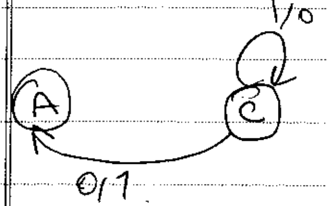
فهمیدم

اگر چند Moore را بهم وصل کنیم سرعت چه می‌شود؟
اگر چند Mealy ؟

سال ۸۶ شماره ۸۷

یادگرفتم

۱) حالت‌ها را بدرستی بنویس



۲) در حالت C با توجه به دیاگرام خروجی را رسم کن

FSM ، Moore را بهم وصل می‌کنیم سرعت کاهش نمی‌یابد

FSM های Mealy را بهم وصل کنیم سرعت کاهش می‌آید

clock بالا می‌ره.

مثلاً ۳ تا Moore با کلاک ۲۰ بهم وصل کنیم نتیجه کل باز کلاک ۲۰

اما ۳ تا Mealy با کلاک ۲۰ بهم وصل کنیم نتیجه کل با کلاک [illegible] ۶۰
می‌شه

P ⟹ q

Subject: شرط کافی Year: شرط لازم Month: Date:

باران می‌آید ⟹ هوا ابری است

A=B ⟹ D=E

D=E شرط لازم برای برآورده شدن A=B است.

و A=B شرط کافی است برای برآورده شدن D=E است.

شرط را بدون پیوند لازم و کافی استفاده کنید.

تست ۷۰ سال ۸۸ ← گزینه ۲

خودتان حل کنید.

Subject: Year: Month: Date:

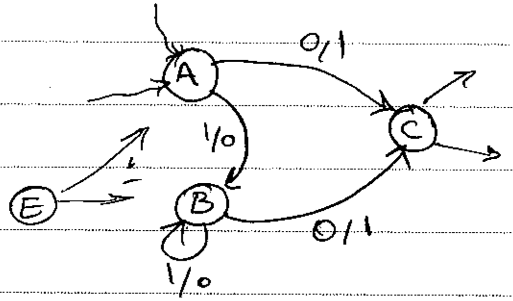
ساده‌سازی FSM ←

اگر در FSM دو حالت باشند که این ویژگی را داشته باشند.

۱) به ازای ورودی‌های یکسان خروجی یکسان داشته باشند

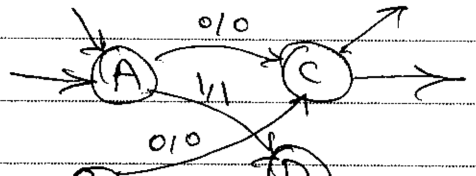
۲) و حالت بعدی یکسانی داشته باشند.

۳) آنگاه آن دو حالت معادلند.



A,B هر دو یکسانند (معادلند) پس هر چه یال به B رفته می‌توانیم وصل کنیم به A و B را حذف می‌کنیم ⟸ ساده‌سازی.

A,B در دیاگرام زیر به صورت مشروط معادلند به شرطی معادلند که D و E معادل باشند



Pair Table

هرکی معادل بود ✓ ـ هرکی معادل نبود X

مشروط

B	X			
C	X	~~B=C~~		
D	X	X	D=C	
E	✓	X	X	X
	A	B	C	D

اگر B را با C معادل بگیریم معادل می‌شوند.

② یک کاری کنیم گزاره‌ها درست باشد

B=C ⟹ B=C ⟶ $\overset{ن}{>}$ ⟹ $\overset{ن}{>}$ ⟶

D=C ⟹ D=C ⟶ $\underset{>}{ن}$ ⟹ $\underset{>}{ن}$

T باشد

A=E A,E معادل

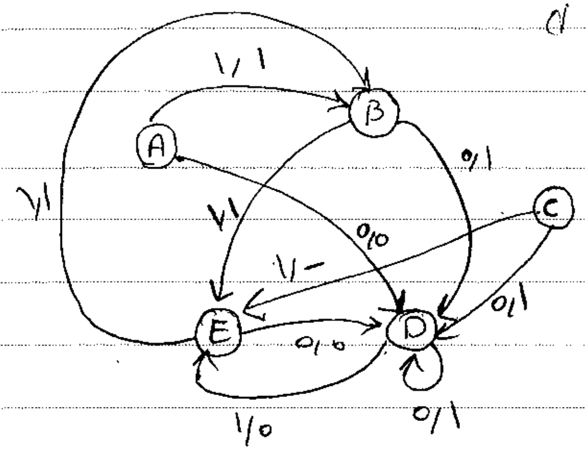
⟹ B=C B,C معادل

در کل ۳ حالت ماند

A _ B _ D ✓

مثال برای ساده‌سازی FSM :

مثال:



حل: باید از الگوریتم استفاده کنیم تا بتوانیم به صورت صحیح و امن ساده کنیم

زیرا اگر همینطوری رو هوا ساده کنیم غلط و خطرناکه.

الگوریتم ساده‌سازی:

	A	B	C	D	E
A					
B					
C					
D					
E					

Pair table

Bit Slice

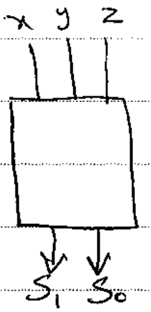
می‌توان مداری ساخت که داده‌های یک بیتی را گرفته و بطوری که برای n بیت بتوانیم بهم وصل کرد پس مدار Bitslice دو قابلیت زیر را دارد:

۱- قابلیت پردازش داده یک بیتی را داشته باشد.

۲- قابلیت اتصال به خودش را داشته باشد.

جمع کننده ۳ بیتی

x	y	z	S_1	S_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



این جمع کننده ۳ بیت را باهم جمع کند

⇓

حاصل روی ۲ بیت

مدار داخلی این جمع کننده

z \ xy	00	01	11	10
0			1	
1		1	1	1

$S_1 = xy + xz + yz$

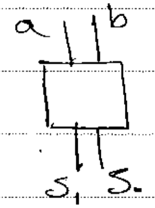
z \ xy	00	01	11	10
0		1		1

$S_0 = x \oplus y \oplus z$

پیاده‌سازی با [illegible]

۹- مطالعات حسابی و ترتیبی:

۱- جمع کننده: ساده‌ترین جمع کننده، جمع کننده‌ای است که دو عدد ۱ بیتی را باهم جمع کند.



a	b	S_1	S_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

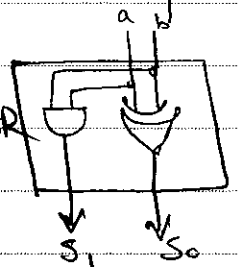
Carry

$C_1 = ab$

$S_0 = a \oplus b$

Sum

Half_ADDER



معمولاً این مدارات برای ورودی با تعداد کم هستند

برای ورودی با تعداد زیاد از روش‌های طراحی خاص باید استفاده کنیم

فرمول H.A چگونه است؟

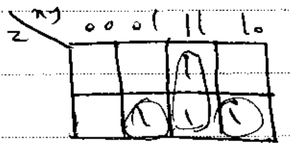
فرمول FA چگونه است؟

دو تا HA با یک گیت ——— می‌شود F.A.

Subject: Year: Month: Date:

برای آنکه از اشتراک سخت افزاری استفاده کنند می‌آیند مدارها پیچیده می‌کنند

اما کاری می‌کنند که از اشتراک سخت افزاری بیشتر استفاده کرد.

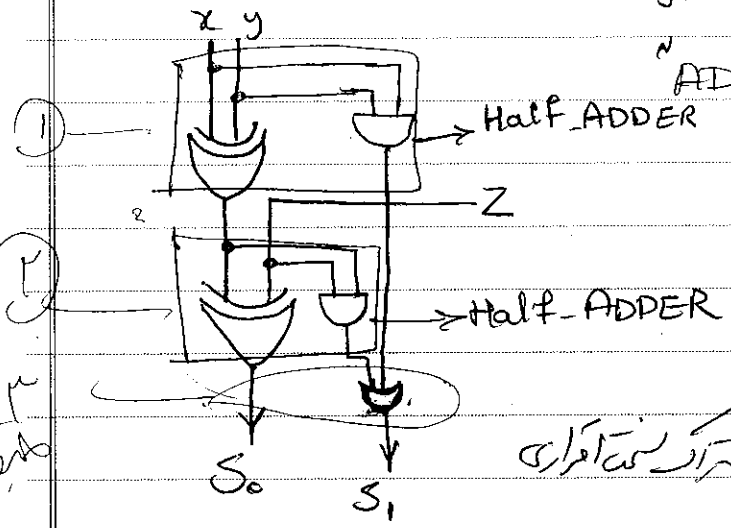

z
xy
00
01
11
10
1
1
1
1

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

$S_1 = xy + x\bar{y}z + \bar{x}yz$ Carry

$S_1 = xy + z(\underline{x \oplus y})$ → اشتراک

$S_0 = (x \oplus y) \oplus z$


x y
①
Half_ADDER
Z
②
Half_ADDER
٣
S₀
S₁

یک جمع کننده ADDER

به این جمع کننده که از اشتراک سخت افزاری استفاده کرده می‌گویند

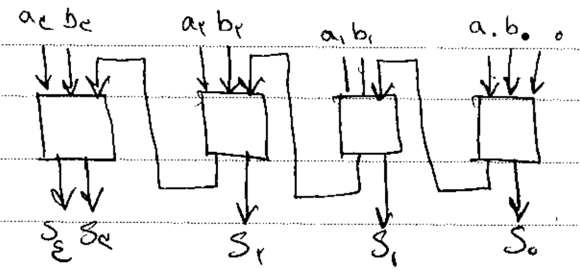
Full ADDER

Full ADDER ۳ تا تک بیت را با هم جمع می‌کند.

Subject: Year: Month: Date:

$a_3a_2a_1a_0$

$b_3b_2b_1b_0$

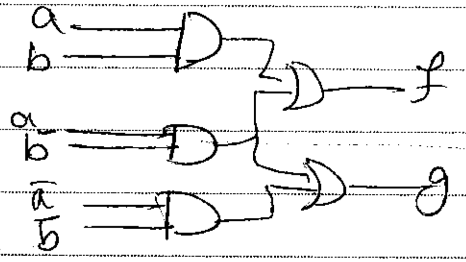

a₃ b₃
a₂ b₂
a₁ b₁
a₀ b₀ 0
S₄ S₃
S₂
S₁
S₀

تکنیک اشتراک سخت افزار

اگر تعداد توابع بیشتر از یکی باشد می‌توان از اشتراک سخت افزار استفاده نمود مثلاً:

$f = ab + a\bar{b}$

$g = ab + \bar{a}\bar{b}$



اشتراک بسته گیتی

جلسه دهم ۹

طرح مقایسه‌کننده به شکل BitSlice

$a_{n-1} a_{n-2} \cdots\cdots a_2 a_1 a_0$

$b_{n-1} b_{n-2} \cdots\cdots b_2 b_1 b_0$

E_i = یعنی تا این موقعیت دو عدد با هم مساوی‌اند

$G_i = a>b$ عدد اول بزرگتر " "

$L_i = a<b$ کوچکتر " "

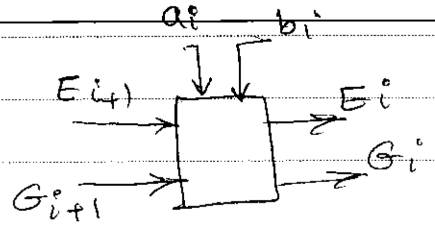
جهت مقایسه از پرارزش‌ترین بیت شروع می‌کنیم.

$E_i = E_{i+1}\overline{(a_i \oplus b_i)}$ یعنی هم دو با هم برابر

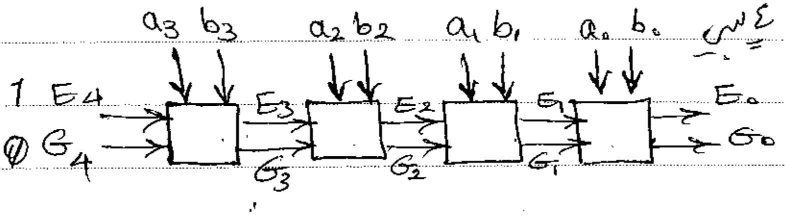
$G_i = G_{i+1} + E_{i+1}\cdot a_i\overline{b_i}$

یا $G_{i+1}=1$ باشد

G_i وقتی ۱ است یا G_{i+1} یک باشد یا $E_{i+1}\cdot a_i\overline{b_i}$ یک باشد



مقایسه‌کننده ۴ بیتی



در ابتدا E4=1 چون در ابتدا بیت‌های قبل همه صفرند پس هر دو عدد با هم برابرند.

G4 نیز صفر چون در ابتدا هر دو با هم برابرند

حالا برای ساخت مدار n بیتی باید n مدار این طراحی کنید

می‌شه BitSlice که خیلی خوبه چون نیازه کار برای n بیت طراحی شه

مقایسه‌کننده صد بیتی

۱۷ بیت and

" ۱۳ (Xnor) not + ~~x~~-or

" ۱۳ or

می‌خواهد.

Subject: ……… Year: ……… Month: ……… Date: ………

جلسه دهم (۱۰)

طراحی مقایسه کننده با تکنیک BitSlice

$a_{n-1} a_{n-2} \ldots a_2 a_1 a_0$

$b_{n-1} b_{n-2} \ldots b_2 b_1 b_0$

E_i = یعنی تا این موقعیت دو عدد با هم مساوی‌اند

$G_i = a>b$ عدد اول بزرگتر " "

$L_i = a<b$ کوچکتر " "

جهت مقایسه از پرارزش‌ترین بیت شروع می‌کنیم.

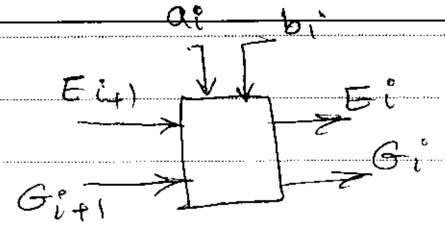
$E_i = E_{i+1} \overline{(a_i \oplus b_i)}$ یعنی هم دو با هم برابر

$G_i = G_{i+1} + E_{i+1} \cdot a_i \overline{b_i}$

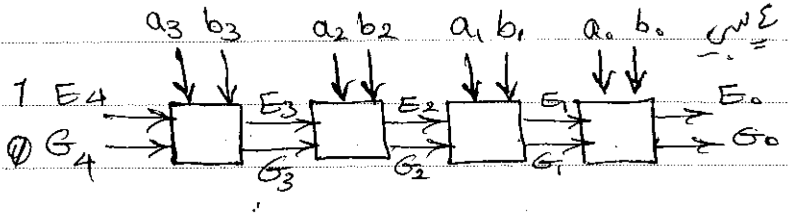
یا $G_{i+1} = 1$ باشد

G_i وقتی ۱ است یا G_{i+1} یک باشد یا $E_{i+1} \cdot a_i \overline{b_i}$ یک باشد.

Subject: ……… Year: ……… Month: ……… Date: ………



یک مقایسه کننده ۴ بیتی



درابتدا چون E4=1 در ابتدا بیت‌های قبلی همه صفر پس هر دو عدد با هم برابرند.

G4 نیز صفر چون در ابتدا هر دو با هم برابرند

برای ساخت مدار n بیتی باید n مدار ۱ بیتی طراحی شود

می‌شود BitSlice که یعنی خوبه چون نیازه به کار برای n بیت طراحی

مقایسه کننده ۴ بیتی

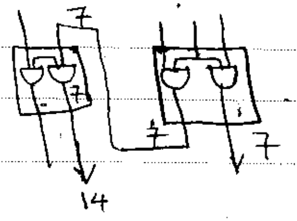
۱۳ بیت and می‌خواهد.

" ۱ (Xnor) not + ~~X~~-or

" ۱۳ or

Subject: Year: Month: Date:

مدار با فرض داخلی زیر شده



⇐ هر مدار 14ns تأخیر دارد.

$C_i = a_i b_i$

$E_i = E_{i+1}\overline{(a_i \oplus b_i)}$ رابطه بازگشتی Recursive

$G_i = G_{i+1} + E_{i+1} a_i \bar{b}_i$ است پس تأخیر کل

مدار با اضافه کردن هر یک مداری، اضافه می‌شود.

سؤال:

با فرض اینکه تأخیر هر گیت 1 باشد. یک F.A برای 4 بیت چین

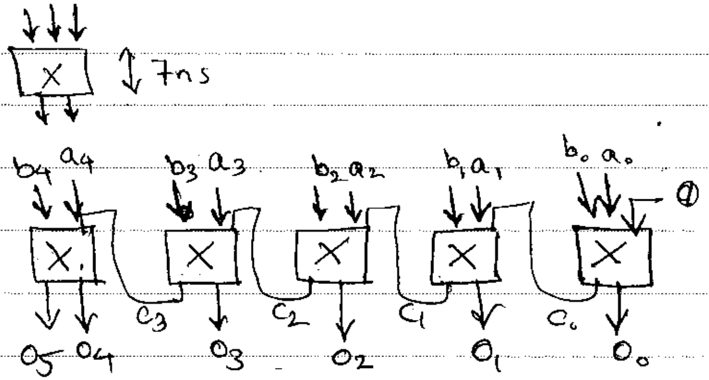
گیت BitSlice چگونه است؟

Subject: Year: Month: Date:

* فرض کنید مداری را به صورت BitSlice طراحی کرده‌اند.

جدول BitSlice به صورت زیر بدست آمده است و تأخیر آن را داده است 7ns

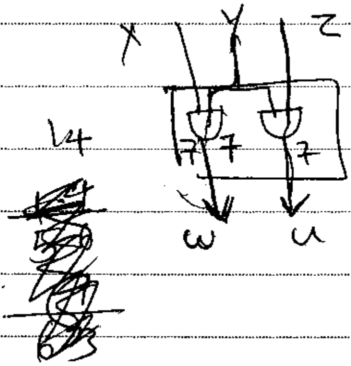
است.


X
7ns
b_4 a_4
b_3 a_3
b_2 a_2
b_1 a_1
b_0 a_0
①
X
X
X
X
X
C_3
C_2
C_1
C_0
O_5 O_4
O_3
O_2
O_1
O_0

تأخیر = نمی‌شه مشخص کرد. چون داخل □ را نمی‌دانیم چیست؟

اما اگر 7ns ثابت مسیر برای بین ورودی و C_i ها باشد آنگاه می‌شه گفت

35ns است اما اگر داخلش را بدانیم می‌تونیم بگیم.


X
Y
Z
14
7
7
7
w
u

اگر داخلش مسیر اینرو باشد

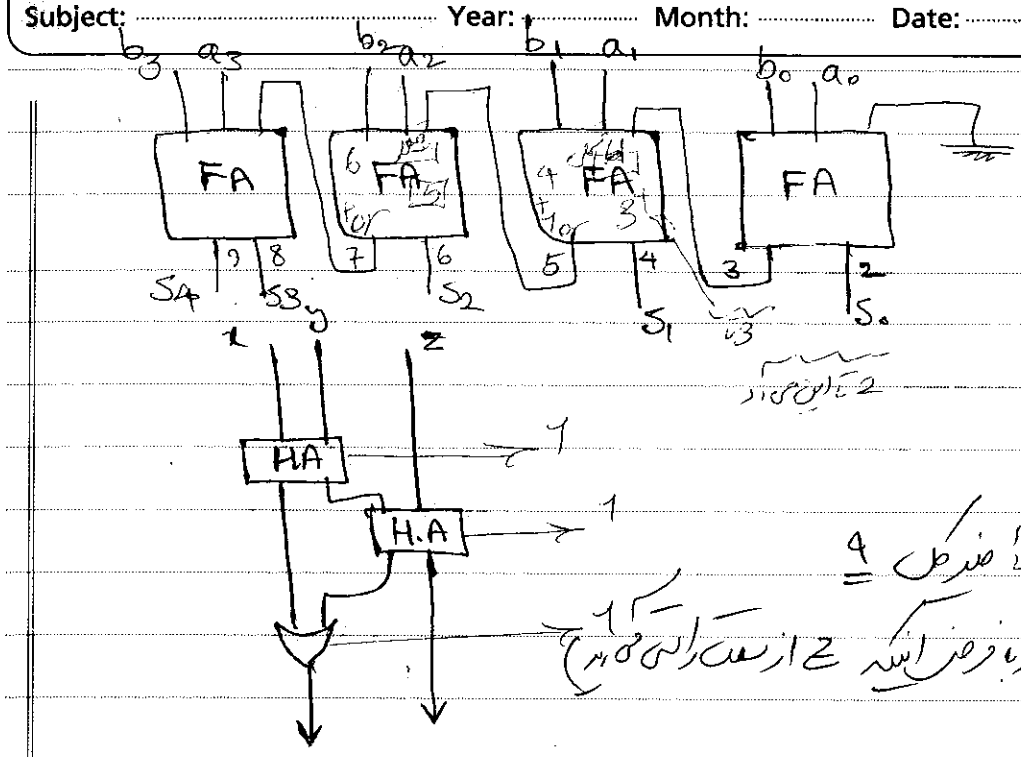
تأخیر 14 می‌شد.

نکته: در هر FA هر سه ورودی یکسان‌اند اما از نظر فیزیکی تأخیر یکسانی ندارند.

از کجا بفهمیم که BitSkew باعث این میشه که تأخیر بیشتری بشود؟ اگر روابط Recursive شد.

(تست ۶۵ سال ۸۱ ~~منطقی~~ معماری)

چون خروجی به قبلی ربطی ندارد پس تأخیر تأثیری ندارد. رابطه به صورت Recursive نیست.

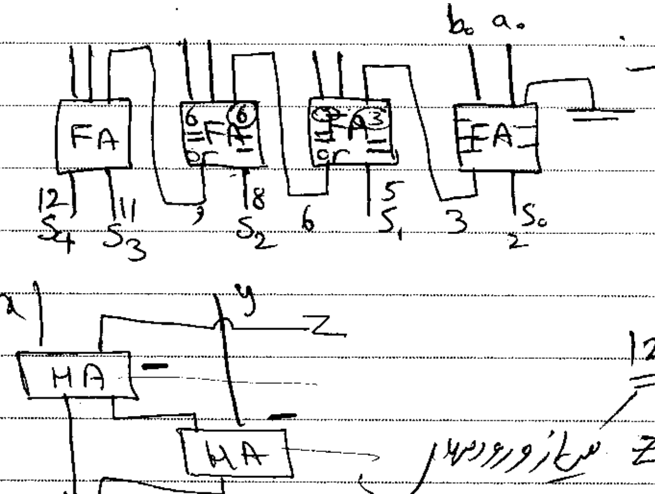


b3 a3
b2 a2
b1 a1
b0 a0
FA
FA
FA
FA
9 8 7 6 5 4 3 2
S4 S3 y S2 S1 S0
x z
HA
H.A
1
1

تاخیر کل 9

(با فرض اینکه Z از سمت راستی میاد)

حالت دوم Z ورودی H.A بالایی باشد و y از مدار نقلی بدهد



b0 a0
FA
FA
FA
FA
12 11 9 8 6 5 3 2
S4 S3 S2 S1 S0
x y z
HA
HA

تاخیر کل 12

(با فرض اینکه Z میاد از ورودی بالا)

اگر یک رابطه Recursive را کاری کنیم که از این حالت خارج شود و بازگشتی نباشد می‌گویند

مشکل کُند بودن مدار BitSlice

مدار کند می‌شود چون Recursive است برای اینکه تند شود باید از حالت Recursive

در بیاوریم. پس باید توابع بازگشتی را بسط دهیم.

در BitSlice چون n محدود است براحتی می‌توان از حالت Reco

خارج کرد و آنرا بسط داد.

مقایسه کننده:

$$E_i = E_{i+1}\cdot\overline{(a_i\oplus b_i)}$$

$E_4=1$ (توضیح ۳ صفحه قبل) اول کار

$$E_3 = E_4\cdot\overline{(a_3\oplus b_3)} = \overline{a_3\oplus b_3}$$

$$E_2 = E_3\cdot\overline{(a_2\oplus b_2)} = \overline{(a_3\oplus b_3)}\cdot\overline{(a_2\oplus b_2)}$$

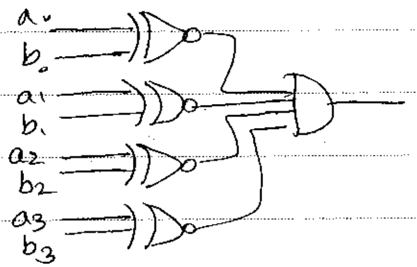
$$E_1 = E_2\cdot\overline{(a_1\oplus b_1)} = \overline{(a_3\oplus b_3)}\cdot\overline{(a_2\oplus b_2)}\,\overline{(a_1\oplus b_1)}$$

$$E_0 = \overline{(a_0\oplus b_0)}\,\overline{(a_1\oplus b_1)}\,\overline{(a_2\oplus b_2)}\,\overline{(a_3\oplus b_3)}$$

پس E0 از حالت Recursive خارج شد پس می‌توان مدار E0 را با 4 Xnor

و یک and ۴ ورودی ایجاد کرد. که تاخیر دیگر زیاد نمی‌شود. پس حالا دیگر

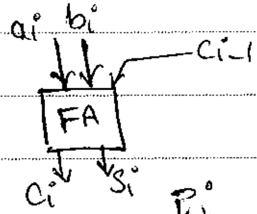
مدار کند نیست.



جمع کننده CLA

همان جمع کننده Bit Slice است که روابط بازگشتی آنرا بسط داده‌اند

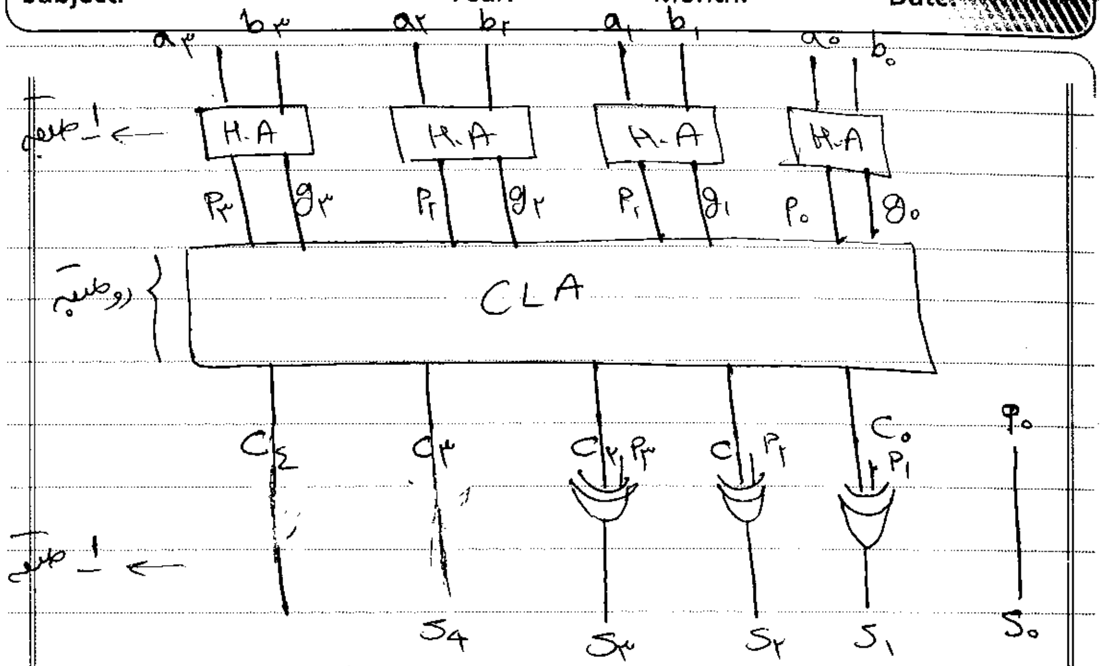
تا از آن حالت خارج شود. و سریع‌تر است.



$$\begin{cases} S_i = (\overbrace{(a_i\oplus b_i)}^{P_i})\oplus C_{i-1} = P_i\oplus C_{i-1} \\ C_i = \underbrace{a_ib_i}_{g_i} + C_{i-1}\overbrace{(a_i\oplus b_i)}^{P_i} = G_i + C_{i-1}P_i \end{cases}$$

فرض ورودی‌ها g_i و P_i هستند.

اگر $g_i=1 \leftarrow C_i=1$ (C_i را تولید می‌کند)


a۳ b۳ a۲ b۲ a۱ b۱ a۰ b۰
H.A
H.A
H.A
H.A
۱ طبقه
P۳ g۳ P۲ g۲ P۱ g۱ P۰ g۰
دو طبقه
CLA
C۴ C۳ C۲ P۳ C۱ P۲ C۰ P۱ P۰
۱ طبقه
S۴ S۳ S۲ S۱ S۰

پس در کل 4 طبقه در این جمع کننده داریم که مستقل از تعداد بیت‌ها است. هر چه هم تعداد زیاد شود تاخیر تغییری نمی‌کند ولی [illegible]

این الگو را با روابط بازگشتی Rec ساختیم .

Carry

CLA ۴ بیتی

می‌دانید $C_{-1}=0$ است .

$C_{-1}=0$ $\qquad$ $C_0=G_0$

$C_1=G_1+G_0P_1$ $\qquad$ $C_2=G_2+C_1P_2=G_2+G_1P_2+G_0P_1P_2$

$C_3=G_3+C_2P_3=G_3+G_2P_3+G_1P_2P_3+G_1P_1P_2P_3$

$C_4=G_4+C_3P_4=G_4+G_3P_4+G_2P_3.P_4+G_1P_2P_3P_4+G_0P_1P_2P_3P_4$

$C_4=G_4+G_3P_4+G_2P_3P_4+G_1P_2P_3P_4+G_0P_1P_2P_3P_4$

SOP

همه اینها SOP هستند و همه ۲ طبقه هستند یعنی همه تعداد اینها را بسط کنیم کنار هم بگذاریم تاخیر ۲ می‌شود [illegible] می‌شود و حالت Recursive ندارد.

قطعات ترتیبی

تنوع محدودی دارند که ۲ دسته‌ی مهم هستند

شیفت‌ها: سریال / موازی / چندکاره

شمارنده‌ها

Register (ثبات)

یک دسته فلیپ فلاپ را ثبات گوییم. یعنی از f.f تشکیل شده‌اند که نشان می‌دهیم همان f.f، اما ثبات چند تا f.f کنار هم هستند. که معمولاً ثبات‌ها ~~[illegible]~~ وقتی معنی دارند که بیت‌های‌شان کنار هم باشند یعنی این بیت‌ها با هم معنی می‌دهند و تک تک بیت‌ها هیچ معنی ندارد. مثلاً یک ثبات ۴ بیتی که عدد را ذخیره کرده هر ۴ تا معنی دارند نه تک تک آنها.

{ چند f.f باهم = ثبات }

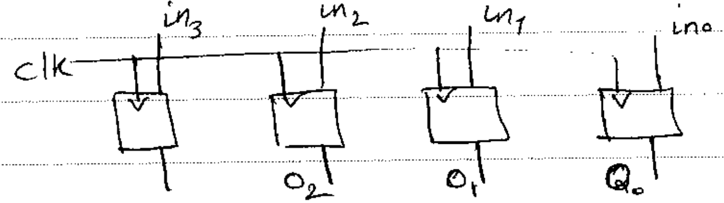
ثبات چیست؟

۲۳ بهمن جلسه ۱۰

شیفت: سریال / موازی

ثبات‌های موازی:

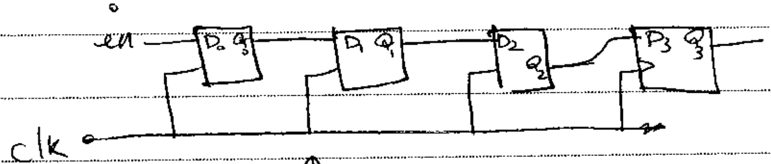
مداری شبیه شکل زیر دارند بطوریکه با خوردن clk همان موقع به بعد وارد می‌شوند in0، in1، in2، ... همزمان با هم وارد می‌شوند ⇐ ثبات موازی



با خوردن clk همه ورودی‌ها همزمان وارد می‌شوند.

ثبات‌های سریال:

هر دفعه که clk می‌خوره هر f.f یک ورودی توش می‌ره. که به این ثبات سریال گویند. که به این شیفت رجیستر نیز گویند.



این ۴ کلاک لازم است.

مهم (Voice قسمت دوم جلسه ۱۰ را دوباره گوش کن)

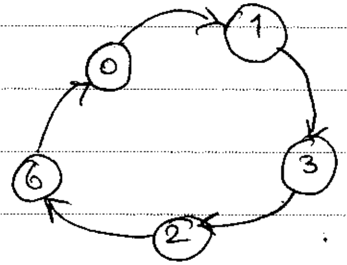
مثلاً برای نوشتن ۴ بیت ثبات ۴ کلاک لازم است

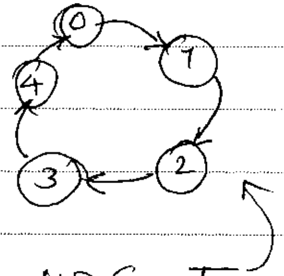
Shift Register شیفت چپ، راست چپ و راست فرقی ندارد

شمارنده Counter

هر FSM که شکل نمودار حالتش شکل حلقوی باشد شمارنده گوییم

این شمارنده است چون شکل نمودار حالتش حلقوی است .





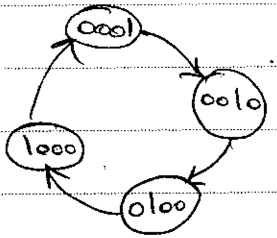
Up Counter چون یک شمارنده‌ای است که یکی یکی می‌ره بالا و وقتی به بزرگترین مقدار رسید دوباره می‌رسه به کمترین مقدار

Down Counter شمارنده‌ای که از بالاترین مقدار شروع می‌کنه یکی یکی کم می‌شه تا برسه به کمترین مقدار و سپس دوباره از بالاترین مقدار شروع می‌کنه ...

** بهش می‌گن Down Counter 3 بیتی چون از 7 تا 0 را می‌شمرد

Counter n بیتی باید از 0 تا 2^n-1 را بشمارد که می‌تواند دو حالت Up Counter و Down Counter باشد .

TSG زمان‌بندی : همیشه 1 دانه 1 دارند .



و این 1 توی هر مرحله میره چپ یا راست .

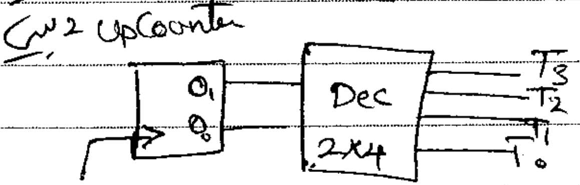
و (TSG 4 بیتی 4 State داره)

مراحل ساخت TSG
روش اول

1ـ قرار دادن یک UpCounter 2 بیتی .

2ـ قرار دادن $Dec_{2\times4}$.

3ـ نهایتاً $T_3T_2T_1T_0$ که می‌شه TSG



Subject: .... Year: .... Month: .... Date: ....

روش سوم جانسون (با شمارنده جانسون)

فرایندی هر دو را دارد و شیفت کند معکوس آخری را انداخته باشد.

۱- یک Shift-Regi ، n بیتی گذاشته

۲- خروجی آخری را می‌گیریم و نات می‌کنیم و می‌دهیم به ورودی Shift Register.

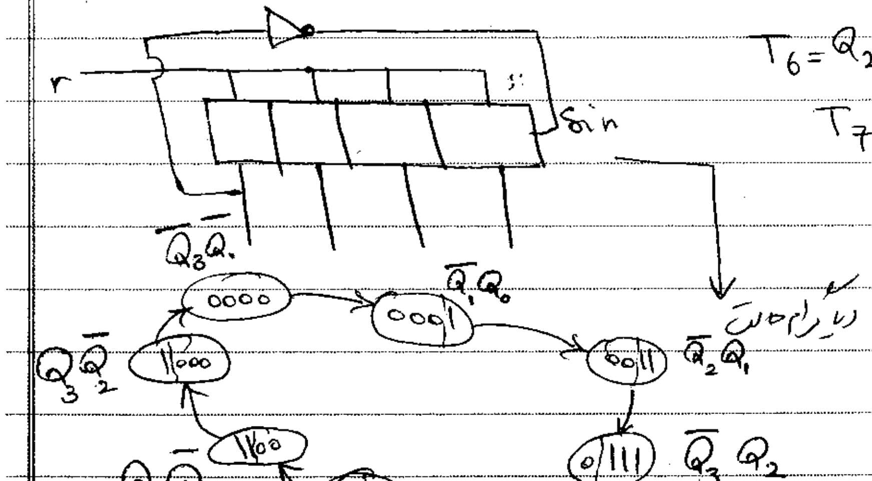
می‌بینیم حالت مثل زیر است که دارای 8 حالت است و بعد به صورت TSG هست. حال با Q ها روابط را درست می‌کنیم.

$T_0 = \bar{Q}_3 \bar{Q}_0$ $T_1 = \bar{Q}_1 Q_0$ $T_2 = \bar{Q}_2 Q_1$

$T_3 = \bar{Q}_3 Q_2$ $T_4 = Q_3 Q_0$ $T_5 = Q_1 \bar{Q}_0$

$T_6 = Q_2 \bar{Q}_1$

$T_7 = Q_3 \bar{Q}_2$

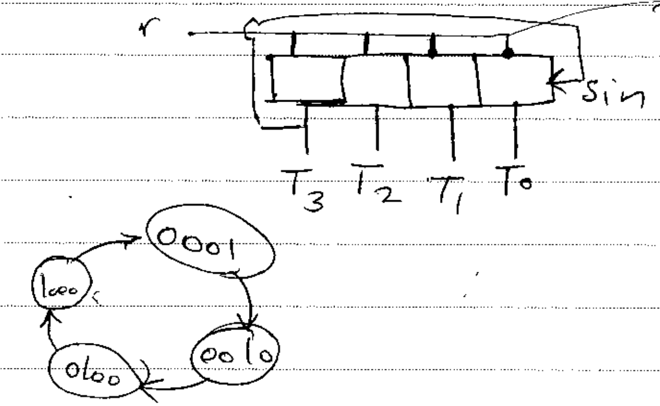

r
Sin
$\bar{Q}_3 \bar{Q}_0$
0000
$\bar{Q}_1 Q_0$
0001
دیاگرام حالت
0011
$\bar{Q}_2 Q_1$
0111
$\bar{Q}_3 Q_2$
1000
$Q_3 \bar{Q}_2$
1100

Subject: .... Year: .... Month: .... Date: ....

ادامه

۱- استفاده از Shift Register و آنی که آخری را بیرون بدیم سمت ورودی

این reset می‌شه بقیه صفر می‌شه


r
Sin
T_3
T_2
T_1
T_0
0001
0010
0100
1000

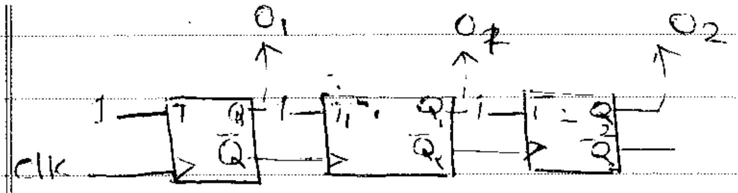
(n بیتی)
TSG در روش اول نیاز به $\lceil \log_2^n \rceil$ تا f.f است

(n بیتی)
TSG در دوم نیاز به n تا f.f است.

روش اول وقتی به درد می‌خوره که تعداد سیگنال‌ها توانی از 2 باشد

(الف)
که هرزه نشود.

حاشیه !!

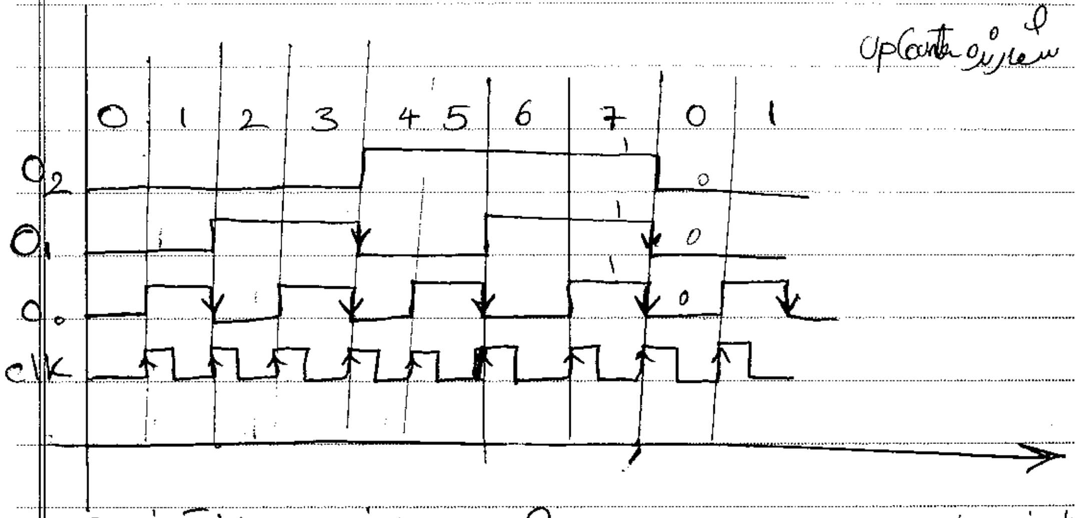

O_1
O_7
O_2
1
clk
T
Q
$\bar{Q}$
1
Q_1
$\bar{Q}_1$
1
Q_2
$\bar{Q}_2$

مدار اول

مدار ترتیبی سنکرون باید clock هایشان به یک جا وصل شود پس شکل بالا

یک مدار ترتیبی سنکرون نیست . چون clk ها به یک جا وصل نیست .

شمارنده UpCounter

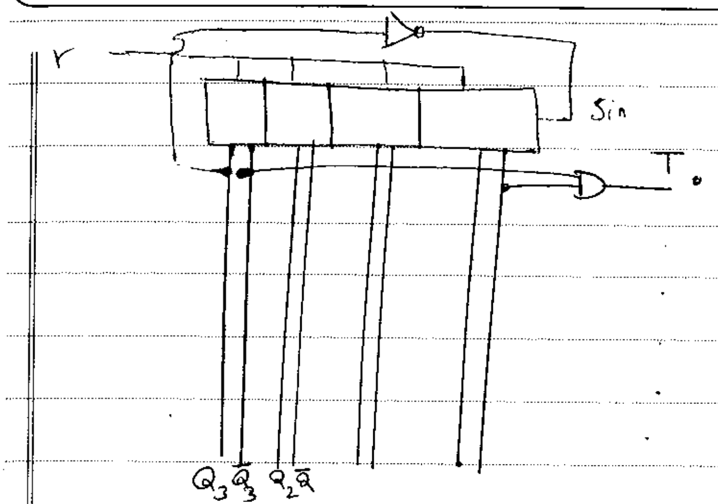


میدانیم طبق فلیپ T وقتی T=1 باشد f.f وضعیت خود را نقیض می‌کند

O_1 جاهایی که O_0 لبه منفی باشد نقیض می‌شود.

$\bar{Q}_1$ وقتی لبه مثبت باشد یعنی Q_1 لبه منفی داشته باشد

P Voice


۲
Sin
T_0
Q_3 $\bar{Q}_3$ Q_2 $\bar{Q}_2$

بهمین ترتیب بقیه

مزایای جانسون :

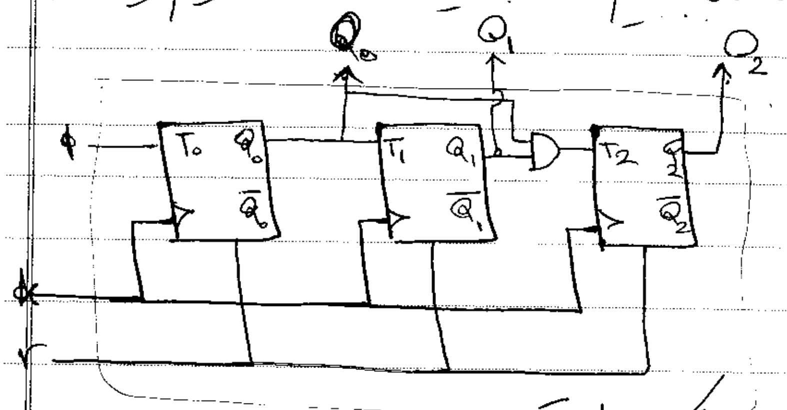
۱. تعداد سیگنال‌های تایمینگ که می‌دهد زوج است.

۲. گیت‌ها and و ۲ ورودی است. مثلاً اگر ۳۲ تا سیگنال بخواهید

۳۲ تا گیت and ۲ ورودی می‌خواهد.

Subject: Year: Month: Date:

حال شکل شمارنده را می کشیم که به این روش ما توانستیم یک شمارنده سنکرون ایجاد کنیم.



مدار دوم

⟸ تولید یک شمارنده سنکرون است.

* توان مصرفی این شکل بیشتر است. *

هر ۳ با هم تغییر می کنند

توی سنکرون وقتی clock می خوره همه با هم تغییر می کنند. اما توی اولی

که آسنکرون است خروجی ها دارای تأخیر هستند که این باعث می شود که تغییرات

کمی کمی می باشد که البته تأخیر آنها خیلی کم است.

مدار دوم سخت افزار بیشتری لازم دارد و هر چه بیتها بیشتر شود بیشتر and

ها (بیشتر)

بیشتر می شود اما مدار اول سخت افزار کمتر و ساده تری دارد.

اگر در مدار دوم دو تا بگیریم و آخری اولی را به اول دومی وصل کنیم یک ۶ بیتی

Subject: Year: Month: Date:

می بینیم که توانستیم یک شمارنده upcounter ۳ بیتی ایجاد کنیم.

که به این شمارنده می گویند شمارنده آبشاری گویند.

شمارنده آبشاری:

شمارنده هایی هستند که خودشان آسنکرون هستند اما وقتی از بیرون

بهشون نگاه می کنیم سنکرون هستند.

حال ساخت شمارنده سنکرون

* حال ساخت شمارنده به روش خودمان یعنی جدول حالت ...

P.S			N.S			T_2	T_1	T_0
Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	فلیپ فلاپ T		
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	0	1
1	1	1	0	0	0	1	1	1

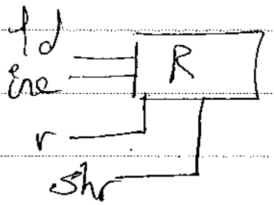
با ۳ فلیپ فلاپ T می سازیم. می بینیم که:

T_0 همیشه ۱ / T_1 همان Q_0 است

Subject: .... Year: .... Month: .... Date: ....

ثبات‌های چندکاره ○ Universal Register

می‌تواند هم ثبات باشد هم شمارنده.



دارای پایه‌های زیر است.

۱ـ ld . برای بارگذاری مقدار در رجیستر

۲ـ inc برای افزایش مقدار

۳ـ reset

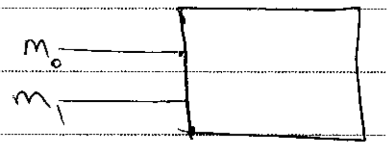
۴ـ shr شیفت رجیستر

چنین کارها می‌تواند انجام دهند که براساس ~~هر~~ پایه فعال بوده، کار مربوطه را انجام می‌دهد.

یعنی مواقع

که برای کارهای مختلف پایه‌ای mode داریم که بکمک آن بیت‌ها می‌توانیم بگیم

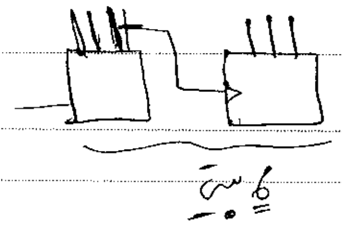
چه کار کند.



m_0	m_1	
0	0	LD
0	1	inc

Subject: .... Year: .... Month: .... Date: ....

اینا رو کرده‌ایم. که اصطلاحاً می‌گویند Casketable هستند.



مدار اول خیلی ساده است و رایج است.

مهم

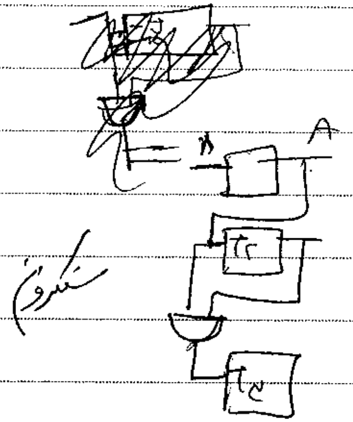
اگر $\bar{Q}$ را به لبه مثبت وصل کنند Up Counter می‌شود.

اگر $\bar{Q}$ را به لبه منفی وصل کنند Down Counter است

Q با لبه مثبت وصل کنند Down Counter است.

Q با لبه منفی وصل کنند Up Counter است.

$\bar{Q}$	+	Up.
$\bar{Q}$	−	Down.
Q	+	Down
Q	−	Up.



مثال یک ثبات چند کاره بسازید که این کارها را انجام دهد، که این ثبات چهار بیتی باشد.

m1	m0	
0	0	حفظ مقدار
0	1	inc
1	0	shr
1	1	reset

و ورودی‌های Mode مانند جدول سمت راست باشد و برای

ساخت این ثبات از D_ff استفاده می‌کنیم

★ وقتی 00 است یعنی حفظ مقدار یعنی خروجی همان باشد که در ورودی

بوده است پس خروجی را می‌دهیم به پایه 0 مالتی پلکسر.

★ وقتی 01 است یعنی خروجی = ورودی + 1 پس برای +1 حالت

از H.A کمک بگیریم A+1 شده می‌شه B: که دقت کنید ۴ بیت

جمع کننده باید به پایه 1 مالتی پلکسر وصل شود.

★ وقتی 10 است حالت shr که باید باشد.

★ وقتی 11 است حالت reset پس ۳ مالتی پلکسر باید بزنیم = 0

(تا سر اینجا خواندیم)

فصل جدید

Programable Logic Device = PLD

وسایلی هستند که در کارخانه ساخته می‌شوند اما سخت افزار داخلی آنها در کارخانه شکل نهایی نمی‌گیرد. که کاربر می‌تواند سخت افزار داخل آنرا تغییر دهد. در واقع یک IC خام است که می‌توان داخل آن را هر چیزی ساخت. که می‌خواهیم.

که چند دسته دارند ۱_ SPLD تا 100 گیت می‌تواند بسازد (چند صد)

۲_ CPLD تا چند هزار گیت داخلشان است.

۳_ FPGA Field Programmable g تا چند صد هزار گیت داخلشان است. (تا میلیون)

(Field به محیط بیرون از کارخانه گویند.)

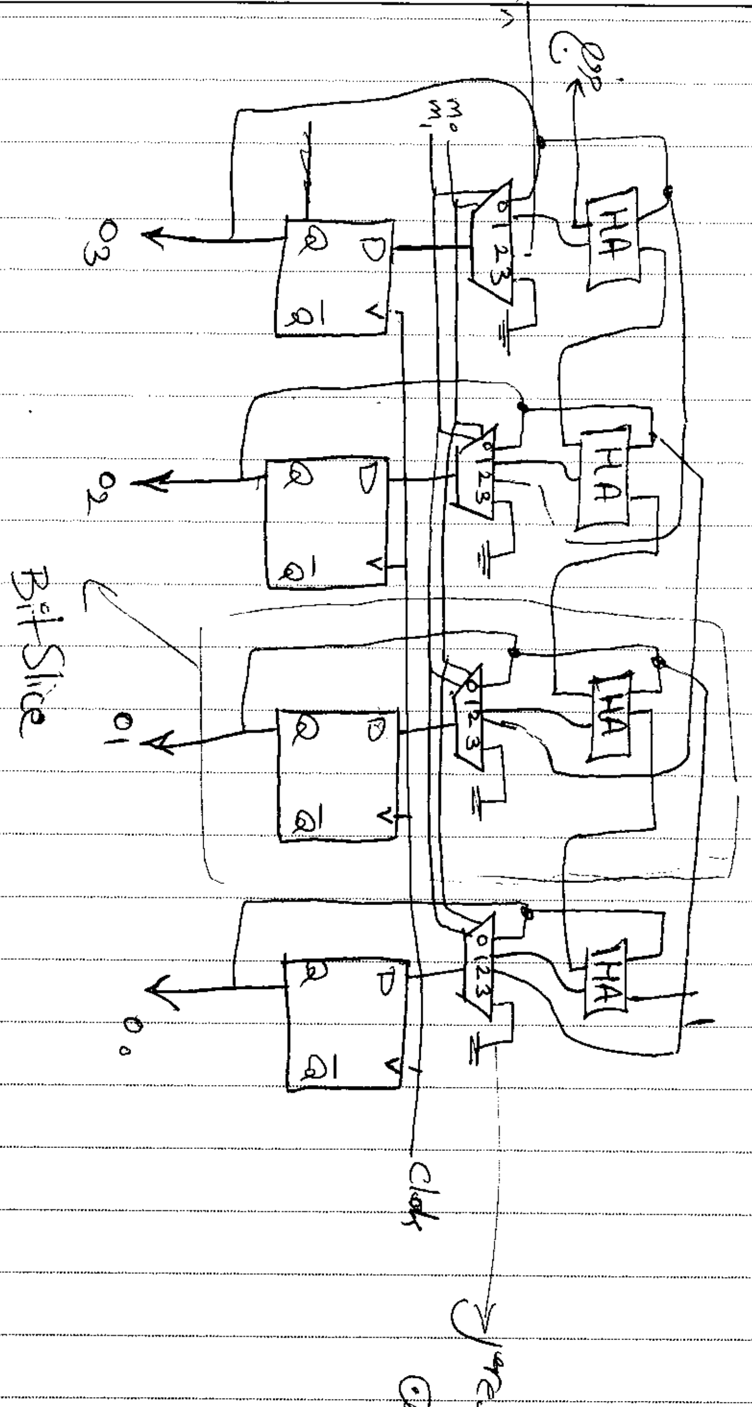
Spld ها خودشان ۳ دسته می‌شوند که شامل:

ROM _۱

PLA _۲

PAL _۳

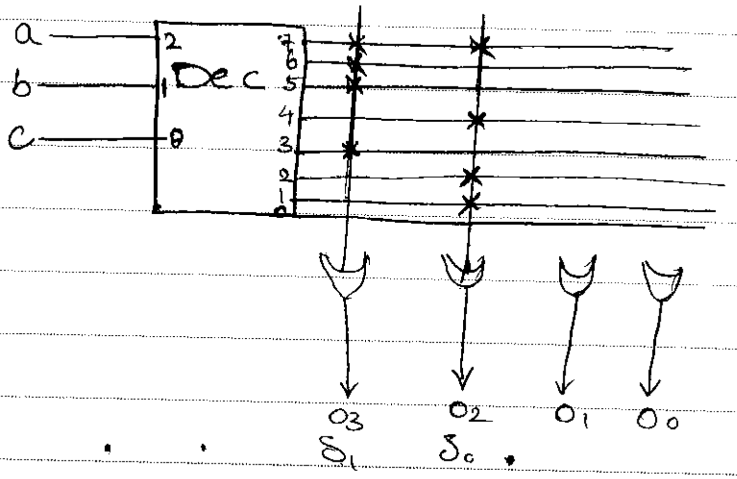
هستند.



فرض کنید می خواهیم یک [illegible] ایجاد کنیم

$S_1 = ab + ac + bc = \sum m(3,6,5,7)$

$S_0 = a \oplus b \oplus c = \sum m(1,2,4,7)$


a
b
c
2
1
0
Dec
7
6
5
4
3
2
1
0
O_3
O_2
O_1
O_0
S_1
S_0

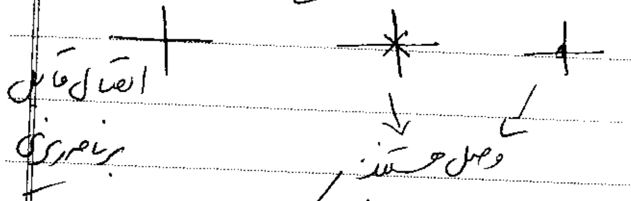
OTP یک بار مصرف است. PLD ی فقط یکبار بشه از آن استفاده کر

MTP چند بار مصرف است. PLD ی چند بار بشه از آن استفاده کرد

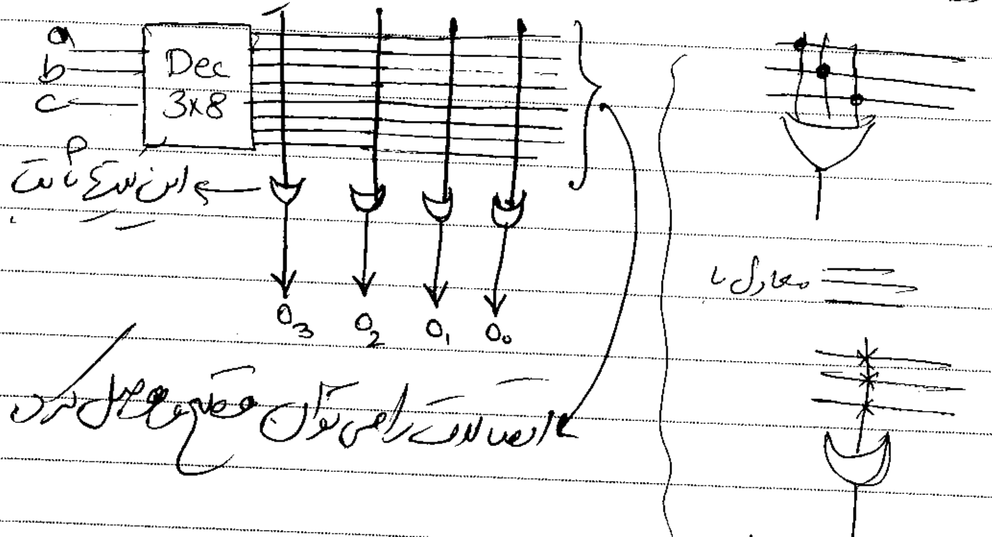
SPLD ها گیت‌ها ثابت هستند تنها جایی که می توان تغییر داد سیم‌ها و اتصالات هستند

اساس SPLD ها برنامه ریزی کردن آنها می باشد داخل SPLD های

دارد که اتصال قابل برنامه ریزی را می توان وصل کرد یا قطع کرد که


اتصال قابل برنامه ریزی
وصل هستند

ROM ساختارش به این صورت است که که (می دانیم که می توان هر مدار را با Σmin پیاده سازی کرد) در این جا از Dec3x8 کمک می گیریم


a
b
c
Dec 3x8
این سطح ثابت
O_3
O_2
O_1
O_0
اتصالات را می توان قطع و وصل کرد
معادل با

Subject: Year: Month: Date:

با داشتن جدول برنامه‌ریزی به کامپیوتر می‌توان PLA را برنامه‌ریزی کرد.

که جدول برنامه‌ریزی از آرایه اتصالات بدست می‌آید حالتی که X است عدد 1 و حالتی که X نیست عدد 0 را جایگزین می‌کنیم. که اینها برای PLA هستند که تعداد گیت‌های کمتری دارد

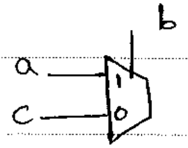
« نکته در مورد ROM »

به ROM بعنوان حافظه نگاه کنیم که ورودی به ROM همان آدرس خانه جدول برنامه‌ریزی است و آنچه در خروجی دیده می‌شود محتوای آن خانه جدول برنامه‌ریزی است. و چون غیر قابل تغییر است به آن می‌گویند Read only Memory.

بعضی از افراد ROM را استفاده می‌کنند برای ذخیره داده‌ها که غیر قابل تغییر باشند و قرار است آن داده‌ها به صورت ثابت بمانند.

Subject: Year: Month: Date:

مثال



$f(a,b,c)=ab+\bar{b}c$

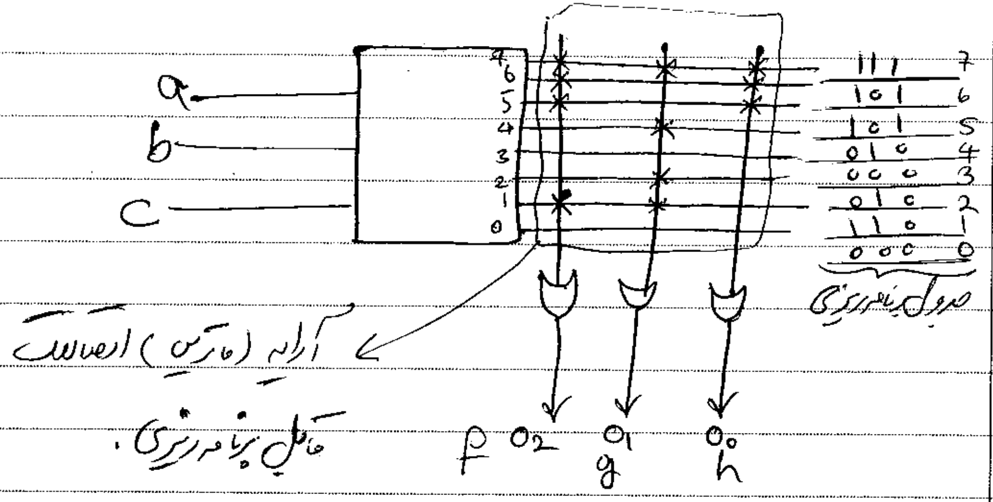
$g(a,b,c)=a\oplus b\oplus c$

$h(a,b,c)=ab+ac$

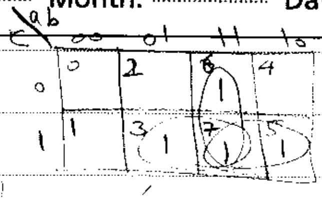
$f(a,b,c)=ab+\bar{b}c=\sum m(1,5,6,7)$

$g(a,b,c)=a\oplus b\oplus c=\sum m(1,2,4,7)$

$h(a,b,c)=ab+ac=\sum m(5,6,7)$


a
b
c
7
6
5
4
3
2
1
0
111 7
101 6
101 5
010 4
000 3
010 2
110 1
000 0
جدول برنامه‌ریزی
آرایه (ماتریس) اتصالات
قابل برنامه‌ریزی.
f O_2
O_1 g
O_0 h

ماتریس and را برنامه‌ریزی نمی‌توان کرد



$f(a,b,c) = \sum m(3,5,6,7)$

۳pt $ab + ac + bc$

$g(a,b,c) = ab + \bar{b}c$

۲pt

$h(a,b,c) = \bar{a}\bar{c} + \bar{b}$

پس در PLA باید ساده‌سازی انجام شود.

f و g را توانستیم با 4pt بسازیم. اما حالا برای h جا نداریم حالا چکار کنیم. باید h را نقیض کنیم ببینیم شاید بشه بسازیم. یا ترکیبی از f و g شاید h بدهد و همه حالتهای ممکن را بررسی کنیم.

$\bar{h}(a,b,c) = \overline{\bar{a}\bar{c} + \bar{b}}$
$= (a+c).b = \underbrace{ab}+\underbrace{bc}$

داریم!

با چه فرقی دارد؟ می‌بینیم.

در اتصالات قابل برنامه‌ریزی است ریسمان ضمن پالسی صورت می‌گیرد.

به گونه‌ای که داخل SPL ها افراد فکر این هستند که چند طبقه تا حد برنامه‌ریزی

ROM همه چی را بصورت Σmin نشان می‌دهد یعنی ساده نشده.

و این نشان می‌دهد که در داخل ROM، ۲ تا Dec است و Dec یک سخت‌افزار پیچیده دارد یعنی حجمش زیاد است. پس ساده‌سازی نداریم.

و این مشکل ROM است که برای حل مشکل یک PLA معرفی کردند.

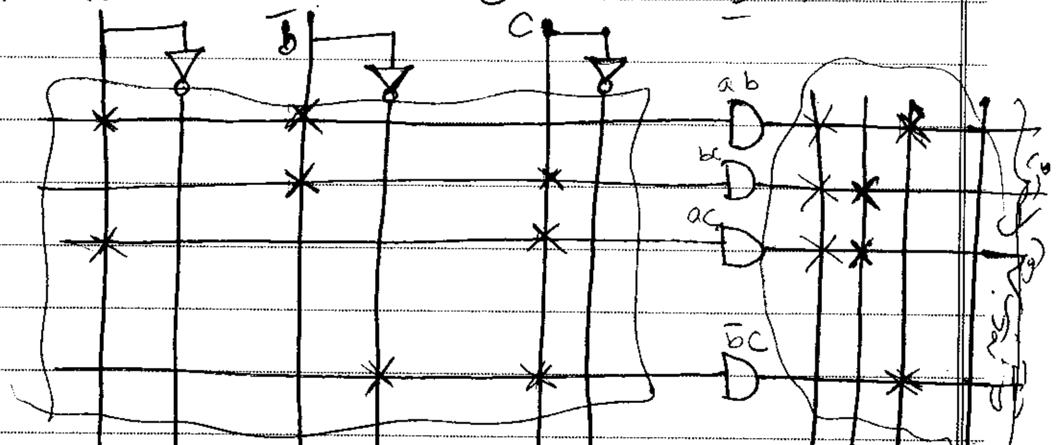
که فرقی که دارد این است که داخل Dec نیست و توابع داخل آن به صورت ساده‌سازی شده است.

<< PLA >>

PLA به این صورت است که داخلش Dec نیست.

هم ماتریس and

هم ماتریس OR دارد



Subject: .... Year: .... Month: .... Date: ....

در PAL باز نیاز به ساده سازی است.

مثلاً

$f(a,b,c)=ab+bc+ac$

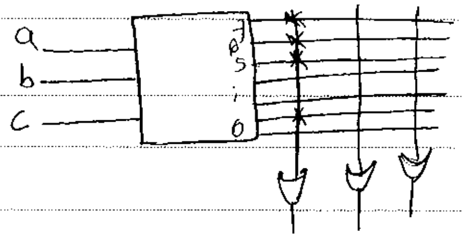
$g(a,b,c)=ab+\bar{b}c$

اینها را نمی‌توانیم توی PAl مثل قرار داد.

$h(a,b,c)=\bar{a}\bar{c}+\bar{b}$

Subject: .... Year: .... Month: .... Date: ....

در مدار ROM یک طبقه قابل برنامه‌ریزی داریم



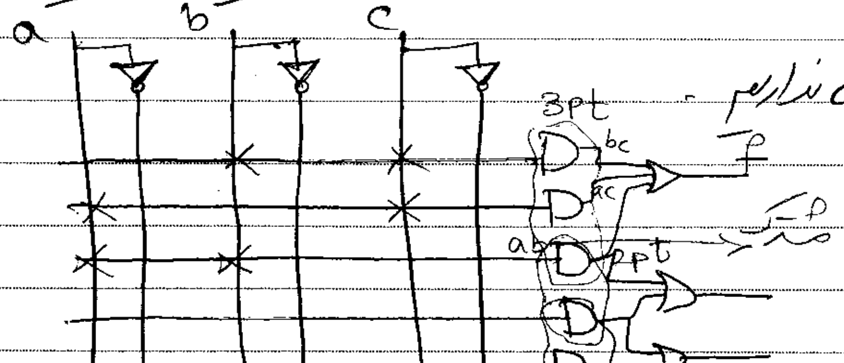
PLA نیز ۲ طبقه قابل برنامه‌ریزی داریم هر طبقه هم شامل ۱۵ تا است.

و اینها خیلی بد است پس می‌آییم از PAL استفاده می‌کنیم.

PAL که مشکل سرعت PLA را حل می‌کنیم

دسته قسمت and به or را نمی‌توانند برنامه‌ریزی کرد و فقط مانند ROM یک ماتریس قابل برنامه‌ریزی دارد. PAL ماتریس and داریم

ماتریس or نداریم.



۱۴

جلسه آخر

نکته: مقایسه بین ROM، PAL و PLA.

	سادگی طراحی	سادگی مدار	تاخیر	ماتریس‌های اتصالی [illegible]
ROM	ساده	پیچیده	۱	or
PAL	پیچیده	ساده	۱	and
PLA	متوسط	ساده	۲	or + and

Pal، Pla نیاز به ساده‌سازی دارند و در بعضی مواقع می باید نات توابع را بسازیم تا بتونیم مدار را طراحی کنیم (قصد پیدا کردن حداکثر اشتراک)

اما در ROM مدار را همان شکل بدون نیاز به ساده‌سازی می‌توانیم پیاده‌سازی کنیم. اما از نظر سادگی مدار ROM پیچیده است.

چون Dec داریم اما در Pal و Pla مدار ساده است و دیگه

Dec نداریم تاخیر در ROM یک طبقه قابل برنامه‌ریزی است و تاخیر آن به اندازه یک طبقه است. اما Pla ۲ طبقه قابل برنامه‌ریزی است

اما PLA ۱ طبقه برنامه‌ریزی است.

از نظر ماتریس‌های اتصالات قابل برنامه‌ریزی، ROM ماتریس orاش قابل برنامه‌ریزی است. Pla هم ماتریس or دارد و هم ماتریس and دارد و اما Pal فقط ماتریس and دارد.

★ در Pla و Pal نمی‌دانیم خودش را بسازیم یا $\overline{F}$ چون هدف شناسایی اشتراک‌ها و اگر $\overline{F}$ را بسازیم باید ~~and~~ not را ایجاد کنیم.

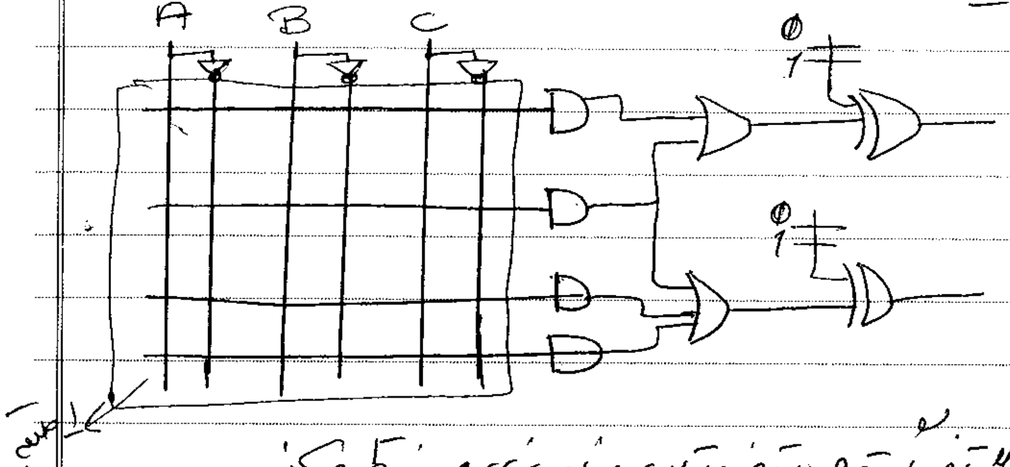
و برای اینکه کاربر از بیرون مجبور به ایجاد not نباشد سازنده‌ها می‌آیند این not را داخل مدار ایجاد می‌کنند.

Subject: .... Year: .... Month: .... Date: ....

می‌آید یک nor می‌گذارند و به دو اتصال برنامه‌ریزی 0 و 1 وصل می‌کنند

اگر 1 باشد $\bar{f}$ می‌آید و اگر 0 باشد خود f می‌آید

در اینجا به دلیل ثابت بودن صفر و یک در سر گیت nor باعث ایجاد طبقه جدید نمی‌شود و تأخیر ایجاد نمی‌کند و طبقه قابل برنامه‌ریزی PAL همان یک طبقه ماتریس and می‌باشد.



نکته کلی: تأخیر گیت‌ها را از تأخیر ماتریس برنامه‌ریزی صرف‌نظر می‌کنیم

Subject: .... Year: .... Month: .... Date: ....

کدینگ Coding کدگذاری

یک مجموعه سمبل‌ها را داریم به نام S که گسسته و محدود است. $S = \underbrace{\{ \dots \}}_{\text{گسسته و محدود}}$

و مجموعه A داریم که هیچ محدودیتی ندارد.

$A \rightarrow S^*$

مجموعه مفاهیم را با رشته‌ای از سمبل‌ها نشان می‌دهیم که به این Coding گویند.

(کدینگ‌ها همیشه یکتا نباشد مثلاً در ادبیات فارسی ما کلمه شیر را برای چندین شیء به کار می‌بریم.)

Coding چندین کاربرد دارد مانند: توان مصرفی، امنیت، کشف و تصحیح خطا.

Coding
- کشف و تصحیح خطا
- محاسباتی

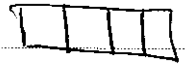
مکمل r، / مکمل r-1.

r= Radix مبنا

r=10، Decimal در زندگی روزمره

مبنای r، ما دارای r رقم هستیم از 0 تا r-1.

همیشه در کامپیوتر، ما محدودیت تعداد خانه داریم، خانه‌های حافظه N.

r= 10 N= 4  $r^N=$ کل اعداد

10^4 تعداد اعداد

r = مبنا ، N تعداد رقم

کوچکترین عدد 0

بزرگترین عدد r^N-1

$$\begin{array}{r} 3763 \\ +\ 8235 \\ \hline \cancel{1}\,1998 \end{array}$$

چون بیشتر از تعداد $\boxed{4}$ N= است این را حذف می‌کنیم

عمل تفریق

راه ۱- استفاده از مدار تفریق کننده چون حجم / هزینه‌بر است از جمع استفاده شود

راه ۲ - مکمل گیری عدد ۲.

* مکمل r ام $C_r(A)=r^N-A$

* مکمل r-1 ام $C_{r-1}(A)=(r^N-1)-A$

* $r^N=\underbrace{100\ldots0}_{N}$

* $(r^N-1)=$ N تا r-1 کنار هم $=(r-1)(r-1)\ldots(r-1)$

مثال $10^4-1=9999$

$2^4-1=1111$

مثال ، N=۴ ، r=10 عدد ۳۲۷۱ مکمل ۱۰ را بگیر

$(۳۲۷۱)\xrightarrow{\text{مکمل ۱۰}}(۶۷۲۹)$

مثال ..

0	9
1	8
2	7
3	6
4	5

$$\begin{array}{r} 3271 \\ 6828+ \\ 1 \\ \hline (6829) \end{array} \leftarrow \text{مکمل ۱۰}$$

مبنای ۲ : مکمل ۲ : $1 + \boxed{\begin{array}{c} 1 \leftarrow 0 \\ 0 \leftarrow 1 \end{array}}$

چون lookup آن فقط می‌شه $\boxed{0|1}$ (نقیض عدد)

اعداد علامت‌دار

$$r^N \text{ عدد} \left\{ \begin{array}{l} 000\cdots0 \rightarrow 0 \\ \vdots \\ (r-1)(r-1)\cdots(r-1) \rightarrow r^N-1 \end{array} \right. \qquad \overbrace{\boxed{\;|\;|\cdots|\;}}^{N}$$

مثال : $N=2$

$r=10$

$$\left. \begin{array}{l} \boxed{0|0} \\ 01 \\ \vdots \\ 99 \end{array} \right\} \text{۱۰۰ عدد}$$

حال قرارداد می‌کنیم که ۹۰٪ اعداد +

۱۰٪ اعداد −

00 | +0
01 | +1
0۲ | +۲
⋮
۸۹ | +۸۹
۹۰ | −۱۰
۹۱ | −۹

$$\begin{array}{r} \text{۱ ۱} \\ \text{۲۷}+ \\ \text{۹۹} \\ \hline \cancel{\text{۱}}\text{۲۶} \end{array} \qquad \text{۹۹} \rightarrow -1$$

مثال .

$$\begin{array}{r} 8291- \\ 3271 \\ \hline 5020 \end{array} \longrightarrow \begin{array}{r} 8291 \\ 6729 \\ \hline \cancel{1}5020 \end{array}$$

مکمل گیری معادل منفی کردن عدد است

اثبات اینکه چرا

$$\cdot\, A + \overset{\text{مکمل } r}{\overline{B}} \equiv A - B$$

$$\cdot\, A + C_r(B) \equiv A - B$$

$$A + B \equiv A + C_r(B) = A + r^N - B = r^N + \underset{\geq 0}{(A-B)}$$

و حتماً $A-B$ همیشه کوچکتر از A است.

$$\underbrace{\cancel{r^N}}_{N+1\text{ بیت}} + \underbrace{(A-B)}_{N\text{ بیت}} = \cancel{1}\underbrace{0000\cdots0}_{N\text{ بیت}} + \underbrace{(A-B)}_{N\text{ بیت}} = N\text{ بیت } (A-B)$$

اضافه

فقط N بیت حساب شود.

$\Leftarrow$ r^N حذف شده است

$$\boxed{\text{مکمل } r-1 = \text{مکمل } r + 1}$$

$$C_{r-1}^{(A)} = (r^N-1) - A$$

(r^N-1) در واقع اینجا هم تفریق نمی‌کند

Subject: Year: Month: Date:

سیستم اعداد متقارن: سیستمی که تعداد اعداد + و − متقارن باشند.

Voice??

اعداد را هم کدام حالت بگیریم. بیت علامت مشخص

$-50 \le x \le 49$ ✓

$-49 \le x \le 50 \longrightarrow$ رقم علامت نداریم چون

00	0
01	1
⋮	
50	+50
51	−49
52	−48
⋮	⋮

بیت علامت نامعلوم ←

Voice??

1:30

Subject: ~~مسئله~~ صفرها Year: Month: Date:

قرارداد مثال قبل اگر رقم سمت چپ ۰ تا ۸ باشد عدد نامنفی و اگر ۹ بود عدد منفی است. به رقم سمت چپ در این مثال رقم علامت گویند.

~~چون ما~~

رقم علامت: رقمی که به کمک آن می‌توان علامت عدد را مشخص کرد.

مثال:

00	0
01	+1
02	+2
⋮	⋮
44	+44
45	+45
46	−54
47	−53
⋮	⋮
99	−1

$\overset{\text{مکمل ۹}}{(54)} = 45 + 1 \rightsquigarrow C_{10}(54) = 46$

در این جا رقم علامت نداریم.

تا ۹ حالت مختلف می‌توانیم تعریف کنیم. برای داشتن رقم علامت

Subject: ………… Year: ………… Month: ………… Date: …………

مدار جمع کننده BCD:

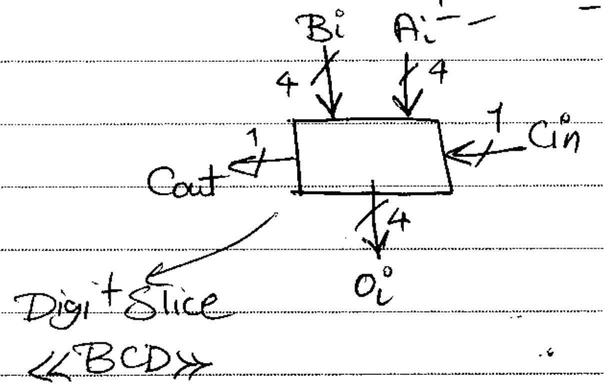
از یک digit Slice استفاده می‌کنیم. که Carry هم در نظر بگیریم که هر کی خواست بتونه Cascate کنه.

هر رقم ~~BCD~~ مبنای ۱۰ در واقع 4 بیت در BCD است پس هر Ai تبدیل به 4 بیت می‌شود.

Carry که یک ستون به ستون دیگه می‌ده یا صفر یا یک چون در بزرگترین حالت 9+9 می‌شد 18 که 1 رقم است (یا صفر یا 1) پس سیم

بعدی‌ها می‌شوند 9 9 ← ⑲

لزوم نمی‌بینیم یک سیم 1 بیتی می‌گیریم.



Digit Slice
<<BCD>>

Subject: ………… Year: ………… Month: ………… Date: …………

BCD ظاهر باینری، مبنا 10.

$329 \xrightarrow{BCD} 0011\ 0010\ 1001$ مبنا 10

$329 \Rightarrow 101001001$ مبنا ۲

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

هر رقم به تنهایی Binary اما رقم‌ها کنار هم قرار داده شده‌اند و تمام می‌شه یک عدد مبنای 10.

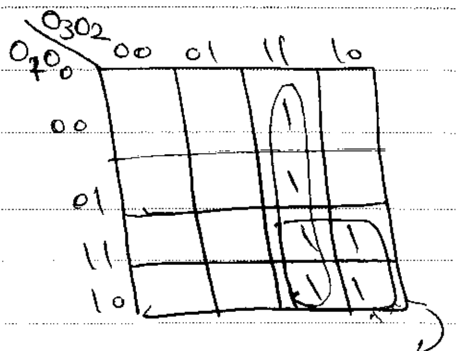
جمع BCD همان جمع مبنای 10 است.

مثال جمع BCP:

```
  ①       ①
  0101 5   0111 7   1000 8
  0110 6   1000 8   0101 5/13
①
0000  0010   0110   0011
```

Cout O1

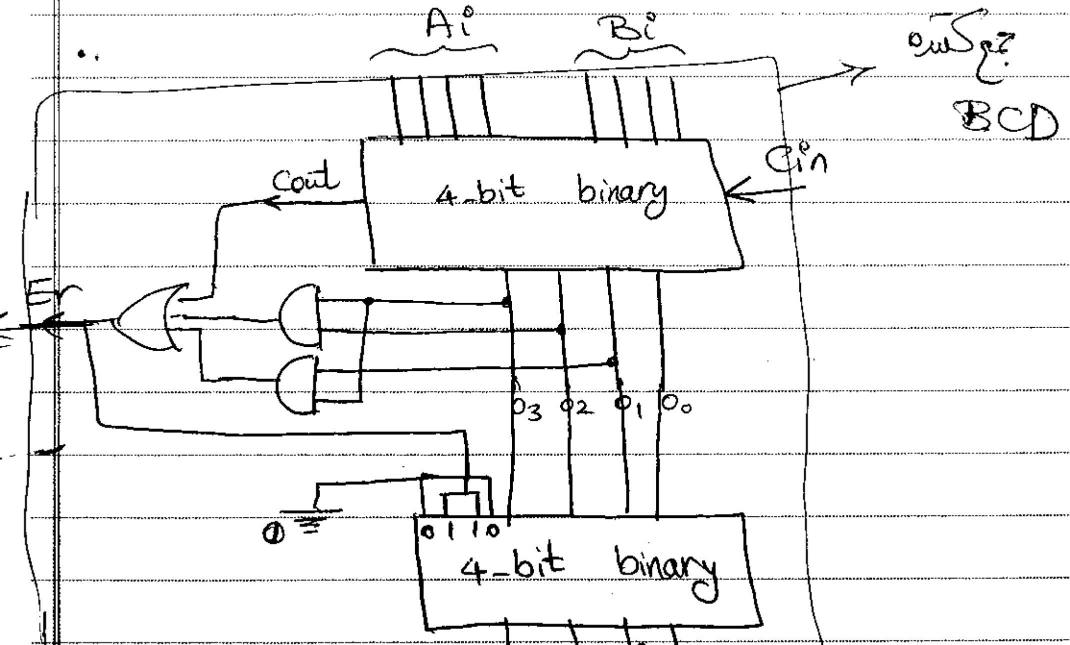
۱۰ : 01010
۱۱ : 01011
۱۲ : 01100
۱۳ : 01101
۱۴ : 01110
۱۵ : 01111
۱۶ : 10000
۱۷ : 10001
۱۸ : 10010
۱۹ : 10011



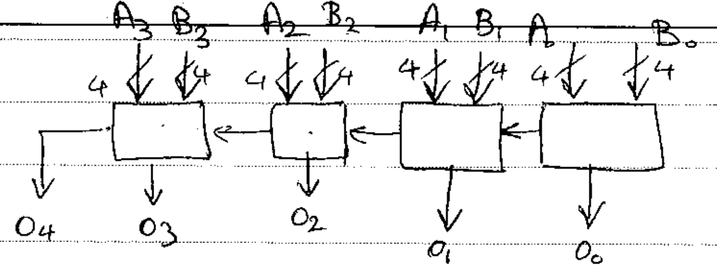
O_3O_2
O_1O_0
00
01
11
10
1
1
1
1
1
1

$E_r = C_{out} + O_3O_2 + O_3O_1$

پس اگر Er=1 بود یعنی جواب برای BCD غلطه.



Ai
Bi
جمع کننده
BCD
Cout
4_bit binary
Cin
Er
O_3 O_2 O_1 O_0
0
0 1 1 0
4_bit binary



A_3 B_3 A_2 B_2 A_1 B_1 A_0 B_0
4 4 4 4 4 4 4 4
O_4
O_3
O_2
O_1
O_0

که این جمع کننده تک رقمی BCD است.

طراحی مدار جمع کننده:

چون استفاده از جدول حقیقت خیلی طولانی می‌شد ۲ از حالت است صرف نظر کرده

می‌آییم از Component های دیگر استفاده می‌کنیم مثل Dec ...

چون ریخت این جمع کننده شبیه جمع کننده باینری ۴ بیتی است

می‌آییم از آن کمک می‌گیریم.

حال می‌بینیم که جمع تا وقتی که 0 تا 9 است درست است اما وقتی بزرگتر

از 9 می‌شود دیگر جواب صحیح نمی‌ده چون ما Decimal هستیم در BCD.

حال بررسی می‌کنیم که در کجا جواب غلط است (برای باینری درست

و BCD غلط)

پس اگر Er=0 باشد باید عدد 6+ شود تا حاصل صحیح بدست آید

شکل ۴ صفحه قبل

مصطفی الان [illegible] درمان بیماری‌ها [illegible]

توجه گفته دکتر جلالی

فصل ۶ و ۷ و ۸

کتاب بورل حذف می باشد

فقط کتاب زیاد شکل برهای

کتاب است

برای بسته [illegible] کلیه پرانتزها

از ۳ فصل جزو است

جدول سمت مداری که BCD سالم را درست

درست	غلط	
10000	01010	10:
10001	01011	11:
10010	01100	12:
10011	01101	13:
10100	01110	14:
10101	01111	15:
10110	10000	16:
10111	10001	17:
11000	10010	18:
11001	10011	19:

قطعاً اگر هر عدد غلط بالا را 6+ کنیم ⇐ عدد درست را می‌دهد

$$\begin{array}{r} 10011 \\ +\ 0110 \\ \hline 11001 \end{array} \quad \begin{array}{r} 10010 \\ +\ 0110 \\ \hline 11000 \end{array} \quad \begin{array}{r} 10001 \\ +\ 0110 \\ \hline 10111 \end{array}$$

$$\begin{array}{r} 10000 \\ 0110 \\ \hline 10110 \end{array} \quad \begin{array}{r} 01111 \\ 0110 \\ \hline 10101 \end{array} \quad \begin{array}{r} 01110 \\ 0110 \\ \hline 10100 \end{array}$$

$$\begin{array}{r} 01101 \\ 0110 \\ \hline \end{array} \quad \begin{array}{r} 01100 \\ 0110 \\ \hline \end{array} \quad \begin{array}{r} 01011 \\ 0110 \\ \hline \end{array}$$

جلسه ۲

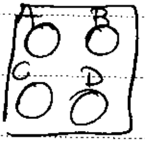
دکتر اجلالی

مدار منطقی

کدهای کشف و تصحیح خطا

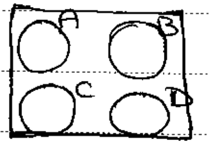
می‌خواهیم بکمک Coding خطاها را کشف کنیم سپس آنها را تصحیح کنیم.

اولین شخصی که بیشتر در این زمینه فعالیت می‌کرده همینگ بوده است. Hamming.



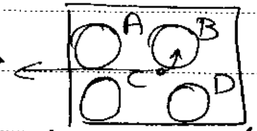
تکه فرش اینطوری که صفحه‌ای دارای 4 جای مشخص برای قرار دادن شیء است.

شانس جابجایی این شیء یک انرژی تصادفی از خارج سیستم به سیستم وارد شده است.



کشف خطا بطوریکه سیستم در وضعیت غیر معتبر است.

اساس تصحیح در کدهای همینگ اینطوری است که به نزدیکترین وضعیت معتبر می‌برد.



خطا - حالت غیر معتبر

تفاوت کار همینگ این است که فاصله‌ها اقلیدسی هستند، اما همینگ یک فاصله خاص دیگری تعریف می‌کند.

«فاصله همینگ»

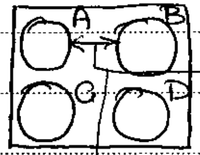
فرض کنید سیستم دیجیتال داریم ۰ به ۱ ، ۱ به ۰ شود خطا باشد بر اثر noise.

اختلاف کد خراب با اصل خودش (سالم) ۱ بیت است.

010110 ← ۱ بیت
010010

اگر کد ۲ بیتش خراب شود فاصله = ۲ بیت است.

اگر قدرت باید طوری باشد که هر روز حداکثر 20cm جابجایی داشته باشد فاصله چقدر باشد که خطا را کشف کنیم. بیشتر از 20cm



اگر بخواهیم تصحیح داشته باشیم باید بیشتر از 2X20cm باشد.

همینگ

کدی که فاصله min آنها ۱ است نه قدرت تصحیح دارد و نه قدرت کشف.

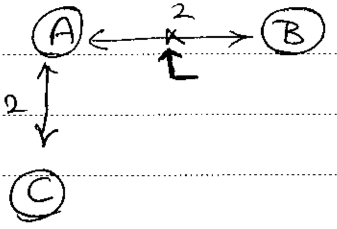
A 00
B 01 → فاصله ۱
C 11 → نه تصحیح
D 10 نه تشخیص

اگر فاصله = ۲ بگیریم آنگاه:

A 000
B 001
C 101
D 110

مثلاً C مورد خطا شود ← 001
می‌فهمیم چون 001 را نداریم پس

می‌فهمیم خطا داریم اما تصحیح نداریم چون:



برای کشف باید فاصله ۳ باشد.

111011
A 000000 فاصله ۵
B 010101 ″ ۴
C 101010 ″ ۲
D 111111 ″ ۱ → تصحیح 11 11 11

مثال: عدد 111011 ← خطا داریم چون در سیستم فوق نیست.

حال برای تصحیح باید به کمترین فاصله تبدیل شود.

۳Parity

Parity زوج وقتی است که تعداد ۱ ها زوج است. } فاصله در هر دو ۲ ا
Parity فرد " " " " فرد " .

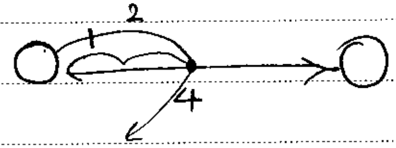
فقط یک بیت تشخیص می‌دهد که البته نمی‌تواند تصحیح کند.
(کشف کند)

000
011
101
111
⊕ → Parity زوج ⊕

001
010
100
111
فرد ⊙ Parity

که همینگ فاصله ۳ ۲ بیت خرابی را کشف می‌کند البته ۱ بیت
را می‌تواند تصحیح کند. SECDED

Sing error Coreti Double error detection



فرض کنید

اگه ۳ باشه فقط ۱ است تصحیح می‌کند اولی یعنی ۲ تا ۳ را به غلط را

★ (Voice قسمت ۲ اولش)

Subject:
Year. Month. Date. ( )

* [illegible]

حداقل چند بیت بگذاریم تا بتوان فاصله همینگ را 3 بگیریم.

A, B, C, D

در اینجا فرض کنید 5 بیت باشد. فاصله 1 نباشد فاصله 2 نباشد

A ‿‿‿‿‿ $1+\binom{5}{1}+\binom{5}{2}=1+5+10=16$

B $=16$

C $=16$

D $=16$

تا کد مختلف $4\times16=64$ می‌خواهیم

اما با 5 بیت نمی‌توان 64 تا ساخت

⟸ $n=6$ درست است پس با 6 بیت می‌توانیم.

حداقل چند بیت بگذاریم تا فاصله همینگ 3 باشد.

$$\left[1+\underbrace{\binom{n}{1}}_{\text{1 نباشد}}+\underbrace{\binom{n}{2}}_{\text{2 نباشد}}\right]\le 2^n$$

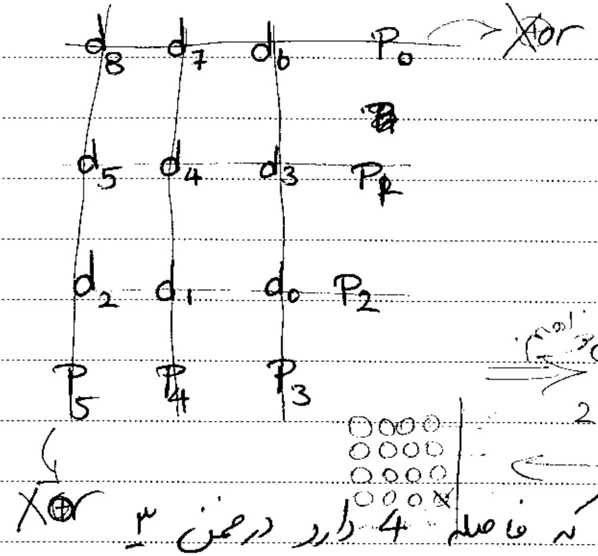
برای 4 عدد

Subject:
Year. Month. Date. ( )

پس برمی‌دارند فاصله همینگ را ۲ می‌گیرند.

$d_8\ d_7\ d_6\ d_5\ d_4\ d_3\ d_2\ d_1\ d_0\ P_5\ P_4\ P_3\ P_2\ P_1\ P_0$

Parity V-H عمودی افقی



وقتی یکی هم خونی ندارد

یعنی P ها دچار مشکل شده‌اند.

این یک

SECDED

برای دارهای 15 تایی چند parity می‌خواهیم

$2\lceil\sqrt{n}\rceil = 2\lceil\sqrt{15}\rceil = 2\times4=8$

این یک طرح Parity H.V است که فاصله 4 دارد در ضمن ۲

بیت را تشخیص (کشف) می‌دهد و ۲ تا ۱ تصحیح می‌کند بدرستی هم تصحیح می‌کند

نکته: یک روش ساخت که با همینگ ۳ یعنی کد را ۳ بار تکرار کن.

00 → 000000
01 → 010101
10 → 101010

فاصله همینگ ← ۳ است

انواع BCD ها

16 × 8 [illegible]

4^4 × ۹×۹×۸×۸

	معمولی BCD	Exc-3	کد $84\bar{2}\bar{1}$	BCD
			$84\bar{2}\bar{1}$	6311
0	0000	0011	0000	0000
1	0001	0100	0111	0010
2	0010	0101	0110	0101
3	0011	0110	0101	0100
4	0100	0111	0100	0110
5	0101	1000	1011	1001
6	0110	1001	1010	1000
7	0111	1010	1001	1010
8	1000	1011	1000	1101
9	1001	1100	1111	1100
		*		

* مکمل 9 این کد می‌شه اینکه 1 به 0 و 0 به 1 بشود

که این جهت تفریق خیلی خوبه. در ضمن جمع کننده‌ها براش خیلی خوب کار می‌کنه. چون وقتی ۲ تا را باهم جمع می‌کنیم در واقع حاصل را باهم جمع کنیم سپس ۳ تا ازش کم کنیم. (خودتان جمع کننده بسازید)

کد $84\bar{2}\bar{1}$ یک کد وزن‌دار است که باز مکمل 9 آن از not شدنش بدست می‌آید.

کد $63\bar{1}1$ هم مکمل 9 آن می‌شه با not بدست آورد.

کد 4221 " " 9 " " " بدست آورد.

کد $84\bar{2}\bar{1}$ 9 بدست آورد.

⇐ کل اینها جمع وزنها = 9 است.

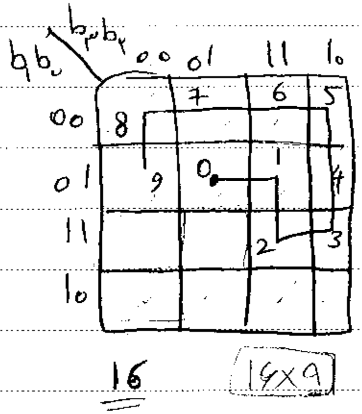
Subject:
Year. Month. Date. ( )

یک روش ساخت کد BCD دیگر.

گری

0	0101
1	1101
2	1111
3	1011
4	1001
5	1000
6	1100
7	0100
8	0000
9	0001

Gray



می‌خواهیم کد گری Self_Com (خودمکمل) بسازیم.

خودمکمل = کدی که مکمل 9 آن از نقیض بیت‌هایش بدست آید.

کدهای انعکاسی (آینه‌ای) عاری سیبی هستند.

چند کد BCD داریم؟

$\frac{10}{0}$
2

$\frac{16}{0000}$

16×15×14..

Subject:
Year. Month. Date. ( )

کد انعکاسی

1000
1001
1011
1010
1110
1111
1101
1100
آینه
0100
0101
0111
0110
آینه
0010
0011
آینه
0001
0000

انعکاسی

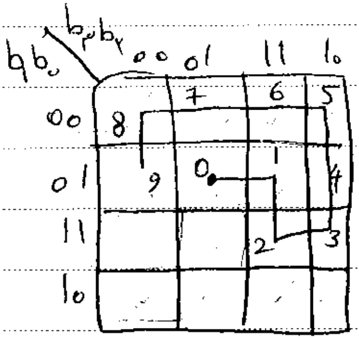
فاصله همینگ $D_H=1$ است.

یک روش ساخت کد BCD دیگر.

گری

0	0101
1	1101
2	1111
3	1011
4	1001
5	1000
6	1100
7	0100
8	0000
9	0001

Gray



16 $\boxed{16\times9}$

می‌خواهیم کد گری Self_Con (خود مکمل) بسازیم.

خود مکمل ـ کدی که مکمل 9 آن از نقیض بیت‌هایش بدست آید.

کدهای انعکاسی (آینه‌ای) مارپیچی هستند

صفر کد BCD نداریم.

$\frac{10}{0}$ $\frac{16}{0000}$

2

16×15×14..

کد انعکاسی

1000
1001
1011
1010
1110
1111
1101
1100

0100
0101
0111
0110
0010
0011
0001
0000

انعکاسی $D_H=1$ (فاصله همینگ) است.