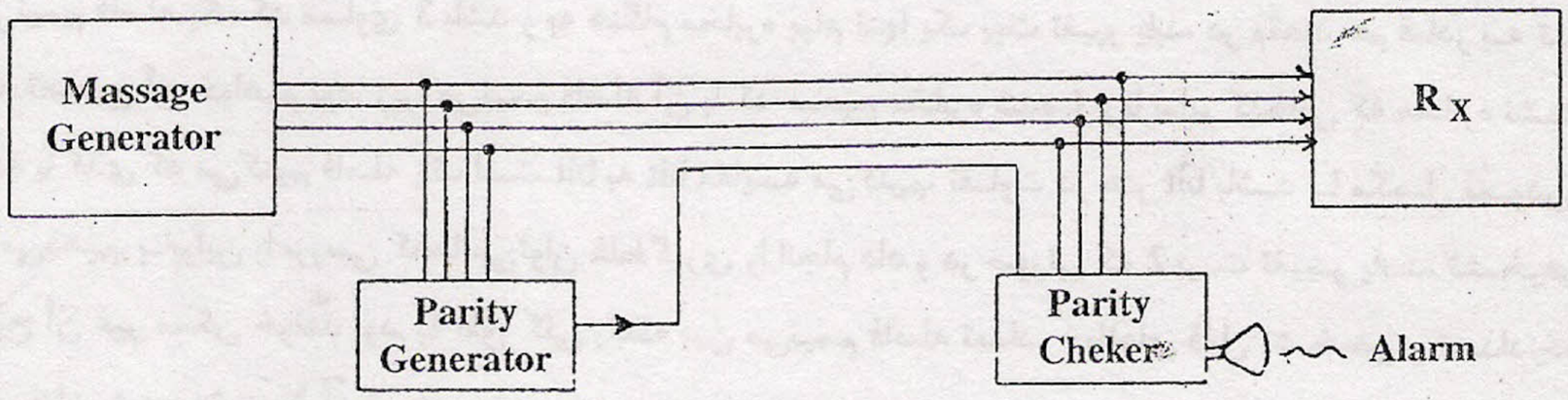
کدهایی با بیش از ۴ بیت:

کدهای 4 بیتی مانند NBCD , x−3 ,... کدخطایاب نیستند، یعنی در صورت مخابره پیام از یک مبدا به یک مقصد در صورتی‌که به دلیل وجود پارازیت بین راه یک bit تغییر یابد در مقصد قادر به تشخیص خطا نخواهیم بود. بنابراین به منظور تشخیص خطا در یک bit باید از 5 – bit Code استفاده کنیم. به عنوان مثال اگر کد رقم 2 دهدهی NBCD، یعنی 0010 را از مبدا مخابره و بین راه بیت دوم آن تغییر یابد و در مقصد پیام 0000 را دریافت کنیم قادر به تشخیص خطا نخواهیم بود، زیرا 0000 یک کد مجاز برای رقم دهدهی صفر در NBCD می‌باشد.

چند نمونه از 5 – bit Code را در زیر مطالعه می‌کنیم.

۱) استفاده از بیت توازن : (Parity Bit)

یکی از روش‌های افزایش کدهای 4 بیتی به 5 بیتی است به طوری که با اضافه نمودن Parity – Bit به عنوان بیت پنجم می‌توان تعداد یک‌های موجود در Code را فرد یا زوج نمود که آن‌ها را به ترتیب Odd Parity و Even Parity گویند. مثلا در صورتی که در کدگذاری از Even Parity استفاده شود و به هنگام مخابره پیام یک بیت تغییر کند پیام دریافت شده در مقصد Odd Parity خواهد بود و بنابراین قادر خواهیم بود وجود خطای تک بیتی را تشخیص دهیم، ولی نمی‌توان آن را تصحیح نمود مدار استفاده از "Parity Bit" به شکل زیر است:



تمرین: مطلوب است طراحی Even Parity Checker , Even Parity Generator (راهنمایی: با استفاده از X-OR).

۲) 2- Out Of -5 Code :

یک کد خطایاب است که در آن برای نشان دادن هر رقم دهدهی از 5 بیت استفاده می‌شود که در هر یک از این کدها 2 بیت یک و بقیه صفر می‌باشند.

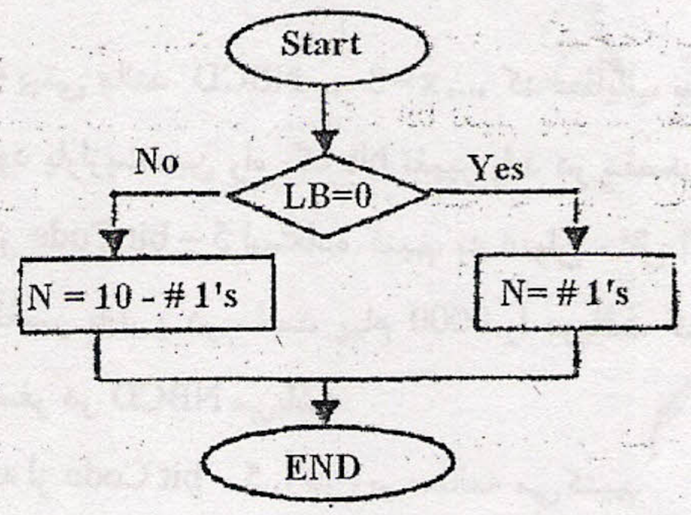
این کد در واقع یک کد نیمه وزن‌دار می‌باشد (Semi Wighted) که به جز کد اختصاص یافته به رقم صفر دهدهی کدهای اختصاص یافته به سایر ارقام دهدهی به ترتیب دارای وزن 7,4,2,1,0 هستند. یعنی این وزن‌ها تنها در مورد Code رقم صفر صادق نیست، ولی درمورد Code بقیه ارقام صادق می‌باشد.

۳) Johnson Code:

کد پنج بیتی است که در آن ابتدا قدم روی هر bit قرار گیرد.

از صفر به یک تبدیل می‌گردد و سپس قدم روی هر bit قرار گیرد از یک به صفر تبدیل می‌گردد.

مزیت Johnson Code در این است که رمزگشایی آن به سادگی انجام می‌شود و با استفاده از فلوچارت زیر می‌توان رقم را تشخیص داد.



کدهایی با بیش از 5 بیت:

تا به حال کدهایی را مورد مطالعه قرار دادیم که تنها قادر به تشخیص خطاهای تک بیتی بودند. برای این‌که بتوان وجود خطای تک بیتی را هم تشخیص داد و هم تصحیح نمود به کدهایی با بیش از 5 بیت نیاز داریم.

تعریف Minimum Distance : حداقل فاصله عبارت است از حداقل تعداد بیت‌هایی که باید در یک کد تغییر یابد تا کد مجازی دیگری از همان سیستم کد گذاری به دست آید. مثلا برای کدهای NBCD، X-3، Gray – Code و Aiken می‌نیمم فاصله برابر 1 ولی برای 2 Out Of 5 می‌نیمم فاصله مساوی 2 می‌باشد.

در صورتی که می‌نیمم فاصله یک کد مساوی 3 باشد و به هنگام مخابره پیام تنها یک بیت تغییر یابد، در مقصد هم قادر به تشخیص خطا و هم قادر به تصحیح آن خواهیم بود، زیرا می‌نیمم فاصله آن با کد صحیح مخابره شده 1 و با سایر کدهایی که مخابره نشده‌اند 2 می‌باشد، در نتیجه با کدی که می‌کنیم فاصله یک است bit به bit مقایسه می‌کنیم، تفاوت در هر bit باشد با مکمل نمودن آن bit تصحیح را انجام می‌دهیم. بنابراین با بررسی کدها می‌توان غلط‌گیری را انجام داد و در صورتی که 2 بیت تغییر یابد، تشخیص خطا ممکن ولی تصحیح آن غیر ممکن خواهد بود. به طور کلی رابطه بین می‌نیمم فاصله تعداد خطاهای قابل تشخیص و تعداد خطاهای قابل تصحیح را می‌توان به صورت زیر بیان نمود:

$$m = c + d + 1 \quad ; \quad d \geq c$$

که در آن m: می‌نیمم فاصله d: تعداد خطاهای قابل تشخیص و C: تعداد خطاهای قابل تصحیح می‌باشد.

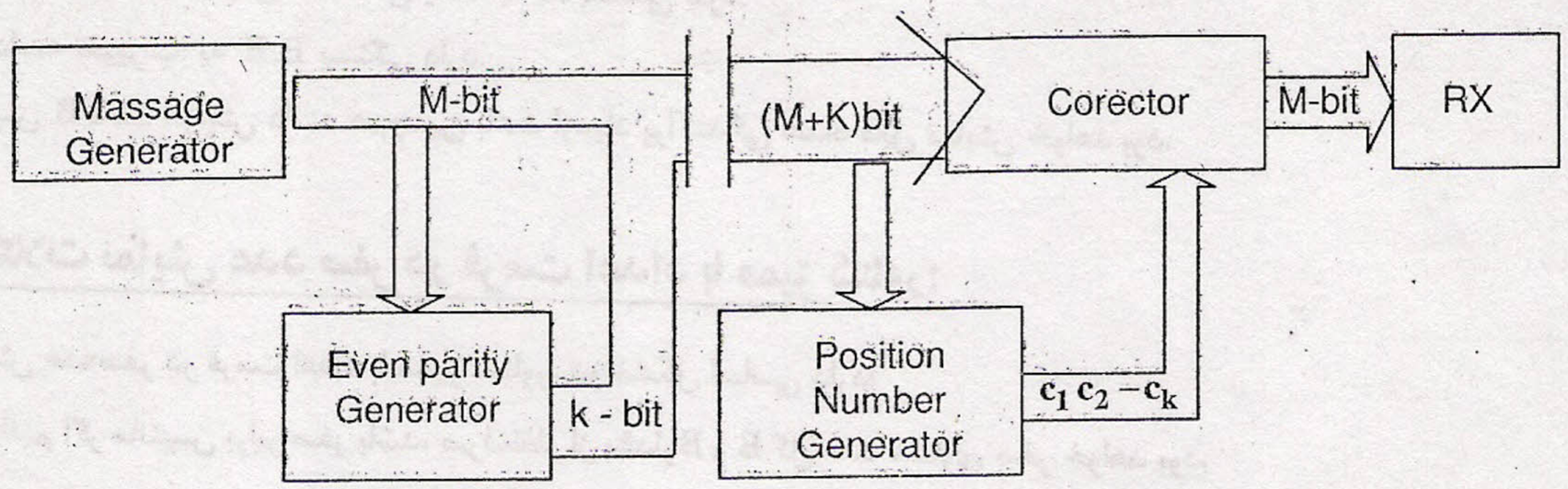
جدول زیر با توجه به m تعداد خطاهای قابل تشخیص و قابل تصحیح را نشان می‌دهد.

m	d	c
1	0	0
2	1	0
3	2	0
	1	1
4	3	0
	2	1
5	4	0
	3	1
	2	2

در این روش تا جایی پیش می‌رویم که بتوانیم تنها وجود خطا در 1 – bit را هم تشخیص دهیم و هم تصحیح کنیم. برای این منظور کد همینگ را مطالعه می‌کنیم.

کد همینگ: *Hamming - Code*

در این کد می‌نیمم فاصله برابر 3 است و می‌توان وجود خطای تک بیتی را هم تشخیص داد و هم‌چنین آن را تصحیح نمود که مدار آن به صورت زیر می‌باشد.



اگر به کد m بیتی، k بیت پاریتی (Parity) اضافه کنیم، کدهای m+k بیتی تولید می‌شوند که در آن موضع بیت‌ها را از چپ به راست با 3,2,1....(m+k) شماره‌گذاری می‌کنیم. بیت پاریتی روی بیت‌های پیام کنترل انجام می‌دهند و عدد دودویی $(c_1c_2 ,.....,c_n)_2$ تولید می‌کند که معادل دهدهی آن محل خطا را مشخص می‌سازد و با متمم‌گیری از آن بیت می‌توان خطا را تصحیح نمود. اگر عدد دهدهی حاصل صفر باشد به مفهوم این خواهد بود که در پیام مخابره شده خطایی رخ نداده است.

چون تعداد کدهای نامطلوب (m+k) و تعداد کدهای مطلوب 1 می‌باشد، بنابراین این تعداد بیت‌های توازن را می‌توان از رابطه زیر به دست آورد.

$$2^k \geq (m+k)+1$$

مثال: فرض کنید پیام مخابره شده یک رقم دهدهی در NBCD باشد، آن‌گاه داریم:

$$m=4 \Rightarrow 2^k \geq 4+k+1 \Rightarrow k=3$$

برای این‌که بیت‌های توازن بتوانند روی بیت‌های پیام کنترل انجام دهند، باید آن‌ها را در مواضعی متناظر با وزن‌های 2^i قرار داد. (یعنی در مکان‌های 1 و 2 و 4) بنابراین کلیه بیت‌هایی که موضع آن‌ها متناظر با توان صحیحی از 2 باشد. مربوط به Parity – Bit و بقیه مربوط به بیت‌های پیام خواهد بود.

Bit Position	1	2	3	4	5	6	7
Bit Name	p_1	p_2	b_3	p_4	b_5	b_6	b_7

به عنوان مثال طراحان کامپیوتر IBM پایه را B=16 انتخاب می‌کنند.

مشخصات اعداد با ممیز شناور

۱) نوع عدد : Mixed

۲) دقت: به تعداد بیتهای اختصاص یافته به M بستگی دارد.

۳) دامنه تغییرات: به B,E بستگی دارد.

افزایش B باعث افزایش دامنه هم‌چنین باعث ازدیاد پراکندگی اعداد قابل نمایش خواهد بود.

مشکلات نمایش عدد صفر در فرمت اعداد با ممیز شناور:

نمایش عددصفر در فرمت اعداد با ممیز شناور، دو مشکل اساسی دارد:

می‌دانیم اگر مانتیس برابر صفر باشد، صرف‌نظر از مقدارB و E کل عدد مساوی صفر خواهد بود.

$$M\times B^{E}$$

$$\text{if } M=0 \text{ then } 0\times B^{E}=0$$

۱) به دلیل Round of Error گاهی مانتیس نتیجه منجر به عدد غیر صفر ولی فوق‌العاده کوچک می‌شود، مثلاً محاسبه زیر را انسان انجام دهد، نتیجه صفر ولی Fixed – point ALU انجام دهد، نتیجه غیر صفر تولید خواهد شد.

$$1-3\times\frac{1}{3}=1-1=0 \qquad \text{(انسان)}$$

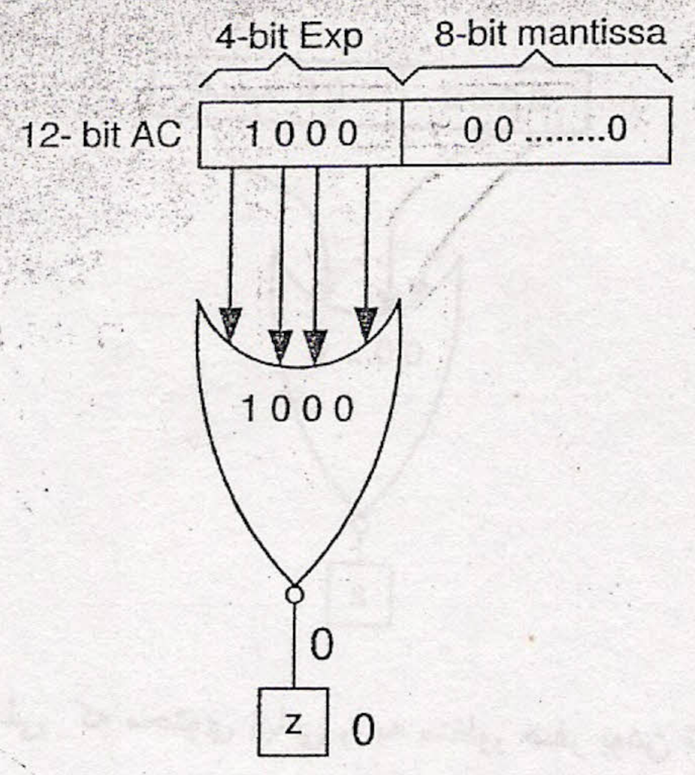
$$1-3\frac{1}{3}=1-3\times(0.33...3)=1-0.99...9=0.000...01 \qquad \text{(کامپیوتر)}$$

$$M\times B^{E\rightarrow \text{اختصاص کوچک‌ترین عدد منفی}}=\frac{M\rightarrow \text{عددی فوق‌العاده کوچک}}{B^{E}\rightarrow \text{عددی فوق‌العاده بزرگ}}\xrightarrow{\text{limit}}O$$

برای رفع این مشکل باید به نمای عدد صفر کوچک‌ترین عدد منفی را اختصاص دهیم. در صورت اختصاص k – bit به ناحیه نما چون کوچک‌ترین عدد منفی قابل نمایش در 2'S Complement مساوی $\left(2^{k-1}\right)$ می‌باشد. بنابراین این عدد را برای همیشه به نمای عدد صفر در فرمت اعداد با ممیز شناور اختصاص می‌دهیم.

۲) اختصاص دادن کوچک‌ترین عدد منفی به نمای عدد صفر، مشکل اول را برطرف می‌کند ولی مشکل دوم را ایجاد می‌کند، زیرا دستورالعمل‌هایی انشعاب شرطی که محتوای ثباتی را به منظور صفر بودن تست می‌کنند، مواجه با مشکل می‌شوند.

مثلاً با فرض 4 – bit Exponent کوچک‌ترین عدد منفی در 2'SComplemat عدد 8- و معادل Binary آن 1000 و در نتیجه Z – Flag صفر خواهد شد و عمل انشعاب انجام نخواهد گرفت.



که برای رفع این مشکل باید از نمای اریب‌دار (Biased – Exponent) استفاده شود. مثلا اگر $k-bit$ به ناحیه نما اختصاص یافته باشد، چون کوچک‌ترین عدد منفی مساوی با -2^{k-1} خواهد بود و نمایش آن به صورت $1\underbrace{000...0}_{k-1}$ خواهد بود و در این صورت دستورالعمل انشعابی مانند IF(AC=0)Then Go To X صحیح انجام نخواهد گرفت. روی این اصل باید از code$(\text{Exess} - 2^{k-1})$ برای نمایش E استفاده نمود که آن را «نمای اریب دار» گویند و مقدار 2^{k-1} را اریب (Biase) گویند.

مثلاً در صورتی‌که به ناحیه E تنها bit – 4 اختصاص دهیم. برای از بین بردن خطای گرد کردن (.R.O.E) باید کوچک‌ترین عدد منفی قابل نمایش در 2's Compl که مقدار دهدهی آن 8- و نمایش Binary آن 1000 می‌باشد، برای همیشه به نمای عدد صفر اختصاص دهیم.

4–bit exp:

$$\underbrace{-2,...,0,...,+(2-1)}_{4-1} \quad \underbrace{}_{4-1}$$

$$-8,...,\ 0,...,\ +7$$

2'scompl:1000 000 0111

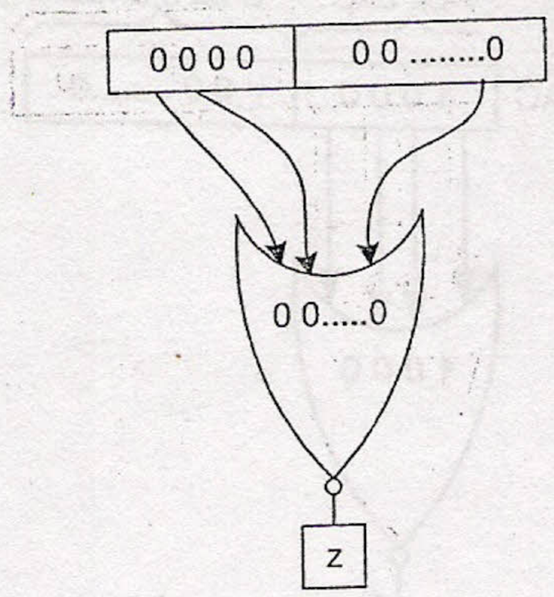
اختصاص 1000 به نمای عدد صفر با توجه به مقدار فوق، Zero – Flag را Reset و در نتیجه دستورالعمل‌های انشعاب شرطی که محتوی ثباتی را به منظور صفر بودن تست می‌کنند، مواجه با مشکل می‌شوند، به منظور رفع مشکل به‌منظور نشان دادن 8- باید از یک سیستم کدگذاری استفاده نماییم که در آن عدد 8- به صورت "0000" نشان داده شود که این سیستم کدگذاری Code(8–Excess) می‌باشد.

4–bit Exp:

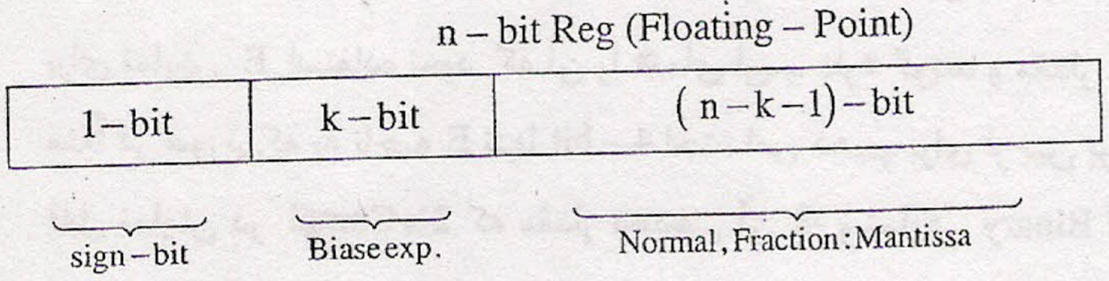
2'5: $\underbrace{1000}_{-8},...,0000,....,0111$

(x –8)Code: $\underbrace{0000}_{-8},1000,...,1111$

در نتیجه:



بنابراین دیگر دستورالعمل‌های انشعاب شرطی که محتوی ثباتی را به منظور صفر بودن تست می‌کنند، مواجه با مشکل نمی‌شوند. با توجه به نمایش نما در Biased exp (نمای اریب‌دار) و مقدار B متوجه می‌شویم که علامت کل عدد $M\times B^{E}$ فقط به علامت M بستگی دارد. به منظور یکنواختی در نمایش اعداد با ممیز ثابت و اعداد با ممیز شناور، bit علامت مانیتس را در سمت چپ‌ترین موضع در نظر می‌گیریم. با فرض n-bit Register فرمت اعداد با ممیز شناور به صورت زیر خواهد بود.



چون B مقدار ثابت است، به صورت توکار در مدار جای داده می‌شود.

دامنه اعداد قابل نمایش

در نمایش اعداد با ممیز ثابت به‌وسیله ثبات n بیتی $\left(2^{32-1}-1\right)+\ldots,0,\ldots,-2^{32-1}$ می‌باشد. برای نمایش اعداد خارج از این دامنه باید از فرمت اعداد با ممیز شناور باید استفاده نمود.

به عنوان مثال: با فرض ثبات 32 بیتی دامنه اعداد قابل نمایش $\left(2^{32-1}-1\right)+\ldots,0,\ldots,-2^{32-1}$ می‌باشد.

$$N=\left(1\underbrace{00\ldots0}_{18\text{ تا صفر}}\right)_{10}=100\times10^{16}=10\times10^{17}=1\times10^{18}=\cdot1\times10^{19}=\cdot01\times10^{20}=\ldots$$

عدد N خیلی بزرگ‌تر از $\left(2^{32-1}-1\right)+$ می‌باشد. بنابراین برای نمایش آن در ثبات 32 بیتی باید از فرمت اعداد با ممیز شناور به صورت مرتب<M و E> استفاده نمود. اما بهترین روش برای نمایش N استفاده از 1×10^{19}. می‌باشد که آن را صورت نرمال گویند.

بنابراین باید شرط نرمال بودن مانتیس کسری و Binary را مطالعه کنیم.

هنجار سازی (Normalization) :

استفاده از صورت نرمال دارای ۲ مزیت است:

۱) نمایش هر عدد با ممیز شناور منحصر به فرد خواهد بود.

۲) تعداد ارقام با معنی قابل نمایش افزایش خواهد یافت.

با فرض ثبات 9 بیتی که در آن bit – 3 به نما و bit – 6 به مانتیس اختصاص یافته است. تنها می‌توان bit – 6 از مانتیس را نشان داد.

با فرض:

$$M\times B^{E}=.00001011011\times 2^{7}=.1011011\times 2^{3}$$

3-bitE	6- bit M	3-bit E	6-bit M
1 1 1	. 0 0 0 0 0	0 1 1	. 1 0 1 1 0 1
(روش a)		(روش b)	

ملاحظه می‌شود، روش b که به صورت نرمال است تعداد bitها با معنی بیشتری را نشان می‌دهد.

شرط نرمال بودن مانتیس کسری:

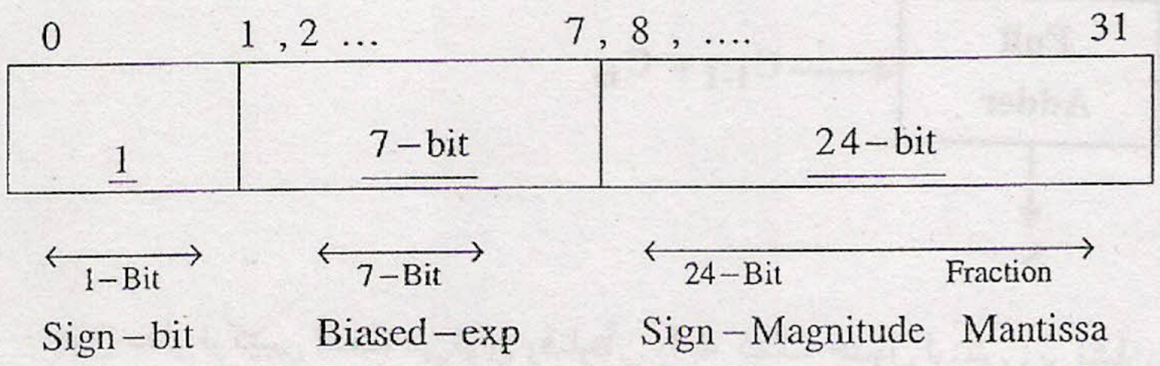
۱) در MSB :Sign Mag. مخالف صفر باشد. $\frac{1}{2}\le M<1=1xx...=1\times\frac{1}{2}+...$

۲) در 2's complement : باید Sign Bit و اولین بیت پشت ممیز مخالف هم باشند:

Sign Bit = 1 .0xx...x

Sign Bit = 0 .1xx...x

تمرین: با استفاده از فرمت اعداد با ممیز شناور در کامپیوتر IBM، نمایش کامپیوتری 0.125×16^{5} را به‌دست آورید:



عملیات ریاضی *(Arithmetic Operation)*:

حداقل انتظاری که از هر Fixed – Point ALU داریم این است که قادر به انجام چهارعمل اصلی جمع، تفریق، ضریب و تقسیم روی اعداد با ممیز ثابت باشد. ولی هر چهار عمل اصلی را می‌توان به کمک مدار Adder انجام داد، زیرا:

$$A-B=A+(-B)=A+2'sB=A+\left(\underbrace{1'sB}_{(X-OR)Gate}+1\right)$$

بنابراین با استفاده از مدار $(X-OR)Gate, Adder$ عمل تفریق را می‌توان به کمک جمع انجام داد. هم‌چنین ضرب را به کمک جمع مکرر و تقسیم را به کمک تفریق مکرر و نهایتاً به‌وسیله جمع مکرر انجام داد.

بنابراین طراح باید سعی کند مدار جمع کننده‌ای از نوع فوق العاده سریع را طرح کند که در این صورت می‌توان هر چهار عمل اصلی را به کمک جمع و با سرعت زیاد انجام داد و چون در جمع دو عدد در 2's Complement بیت علامت نیز مانند سایر بیت‌ها در محاسبات شرکت می‌کند. بنابراین ما فقط مدار جمع کننده برای اعداد بدون علامت را مطالعه خواهیم کرد.

توجه به این نکته ضروری است که پیچیدگی مدار جمع کننده به کدی بستگی دارد که برای نمایش اپراندها به کار می‌رود که ما فقط Binary Adder را بررسی می‌کنیم.

Binary Adder

بنا به قضیه شانون سریع‌ترین مدار جمع کننده را می‌توان به صورت مدار دو سطحی AND – OR متناظر با عبارت بولی مجموع حاصل‌ضرب‌ها طرح نمود. ولی در عمل دو عامل محدود کننده وجود دارد:

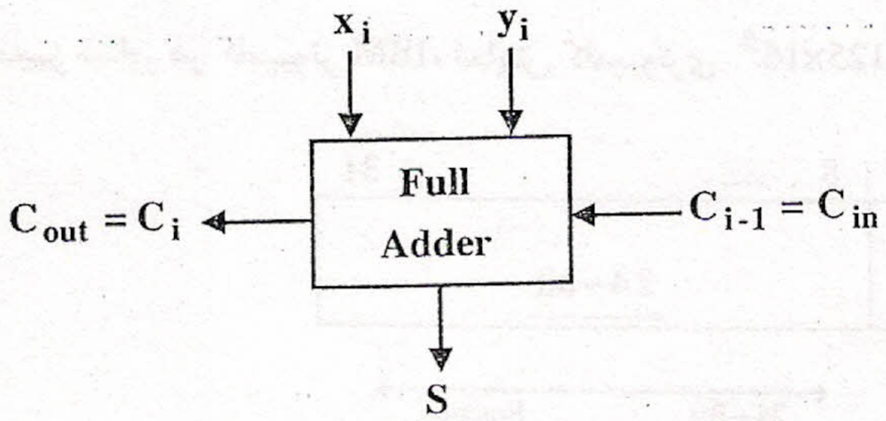
۱ـ پیدایش Fan – in

۲ـ افزایش تعداد Gateها

بنابراین باید تا اندازه‌ای از سرعت جمع کننده صرف‌نظر نمود. ابتدا مدارهای جمع کننده m بیتی $(m < n)$ را طرح نمود که در آن Fan–in به وجود نیاید و سپس اگر $n = mk$ باشد، k تا از آن‌ها را به صورت پشت سر هم به هم وصل کرد و ارقام نقلی را به صورت موجی بین آن‌ها پخش نمود تا مدار جمع کننده n بیتی ایجاد گردد. در صورتی‌که m=1 باشد، مدار جمع کننده را Basic Binary – Adder یا 1–bit Binary–Adder گویند. که به دو صورت ترکیبی و ترتیبی می‌توان آن را پیاده‌سازی نمود.

تمام افزایشگر : *(Full – Adder)*

مدار ترکیبی است که می‌تواند حاصل جمع دو عدد یک بیتی را تولید کند. نمودار بلوکی آن به صورت زیر می‌باشد.



چون مدار ترکیبی است، می‌توان رفتار آن را به کمک جدول ارزش زیر نشان داد.

X_i	Y_i	C_{i-1}	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

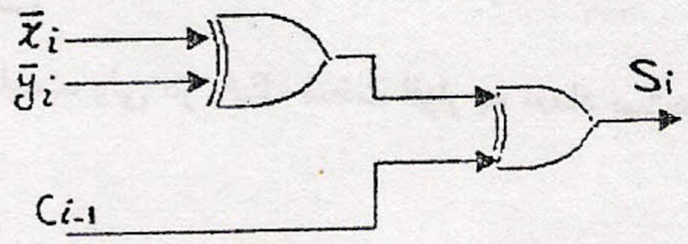
c_{i-1} \ $x_i y_i$	00	01	11	10
0		(1)		(1)
1	(1)		(1)	

$$S_1 = \sum_1 (1,2,4,7) = \bar{x}_1 y_1 \bar{c}_{-1} + x_1 \bar{y}_1 c_1 + \bar{x}_1, \bar{y}_1, c_{2-1} + x_2 y_2 c_{2-1}$$

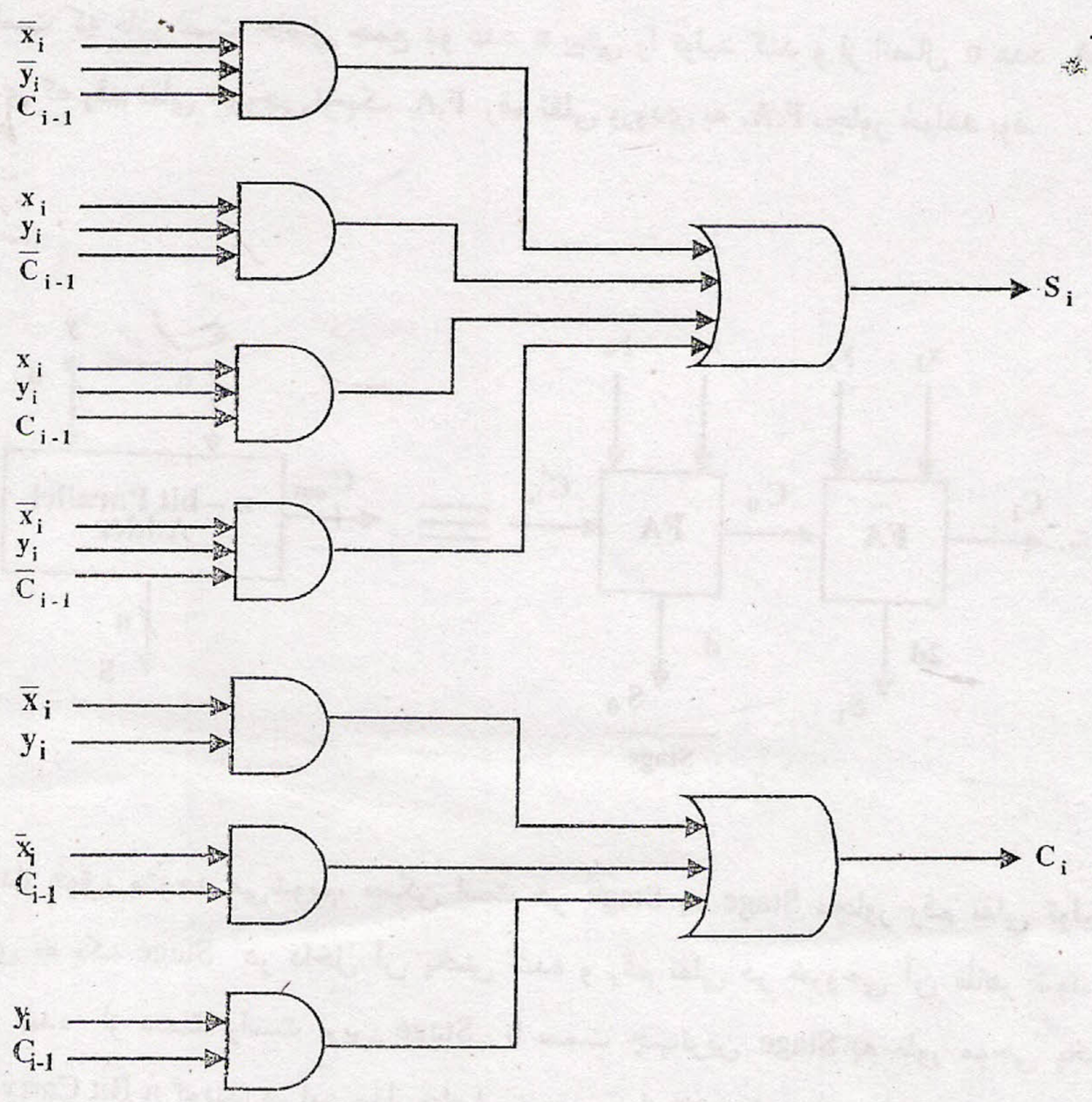
$$S_i = x_i \oplus y_i \oplus c_{i-1}$$

c_{i-1} \ $x_i y_i$	00	01	11	10
0	0	2	6 (1)	4
1	1	3 (1)	7 (1)	5 (1)

$$C_i = \sum_1 (3,5,6,7) = x_i y_i + x_i c_{i-1} + y_i c_{i-1}$$

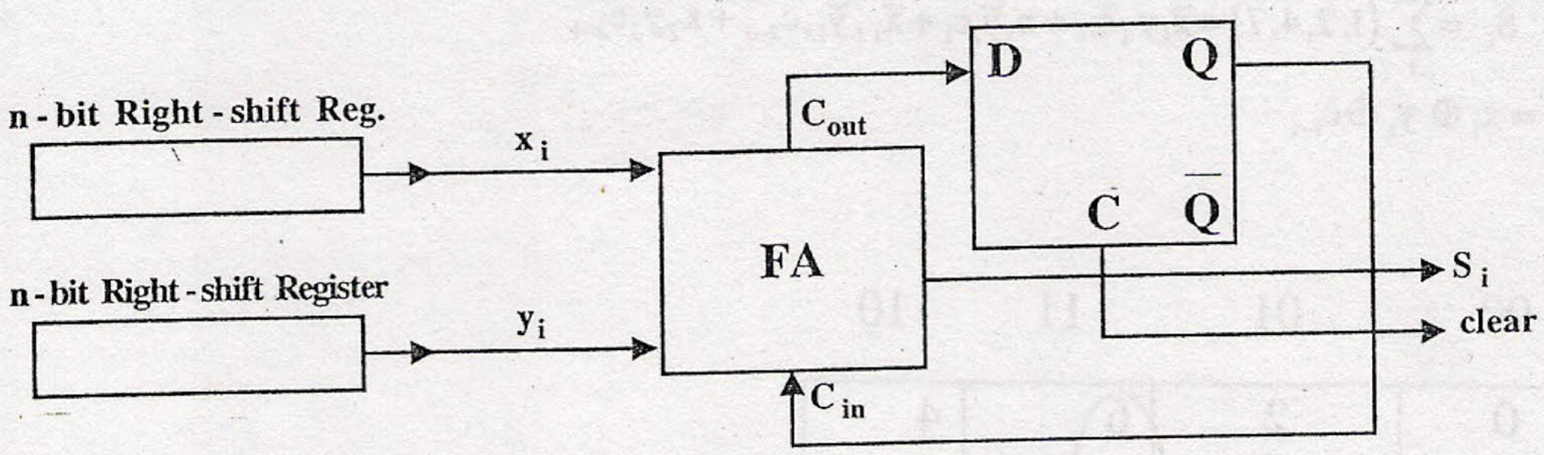


در صورتی‌که فقط از Gate های AND و OR استفاده کنیم، می‌توان داخل Full – Adder را به صورت زیر طرح نمود که به صورت مدار دوسطحی AND – OR می‌باشد.



جمع کننده سری : *1-bit Serial Adder:*

مدار ترتیبی است که با اتصال یک F.A و یک D−Flip Flop به وجود می‌آید و در صورتی که تاخیر مدار دو سطحی F.A برابر d و تاخیر FF مساوی D باشد، آن‌گاه حاصل جمع دو عدد تک بیتی بعد از تاخیر (d+D) تولید خواهد شد.



مقایسه *Serial Adder* و *Full Adder* :

۱) حاصل جمع دو عدد تک بیتی در صورت استفاده از Serial Adder پس از تاخیر (d+D) واحد زمان تولید می‌شود. ولی در صورت استفاده از F.A پس از تاخیر d واحد زمان بنابراین F.A سریعتر است.

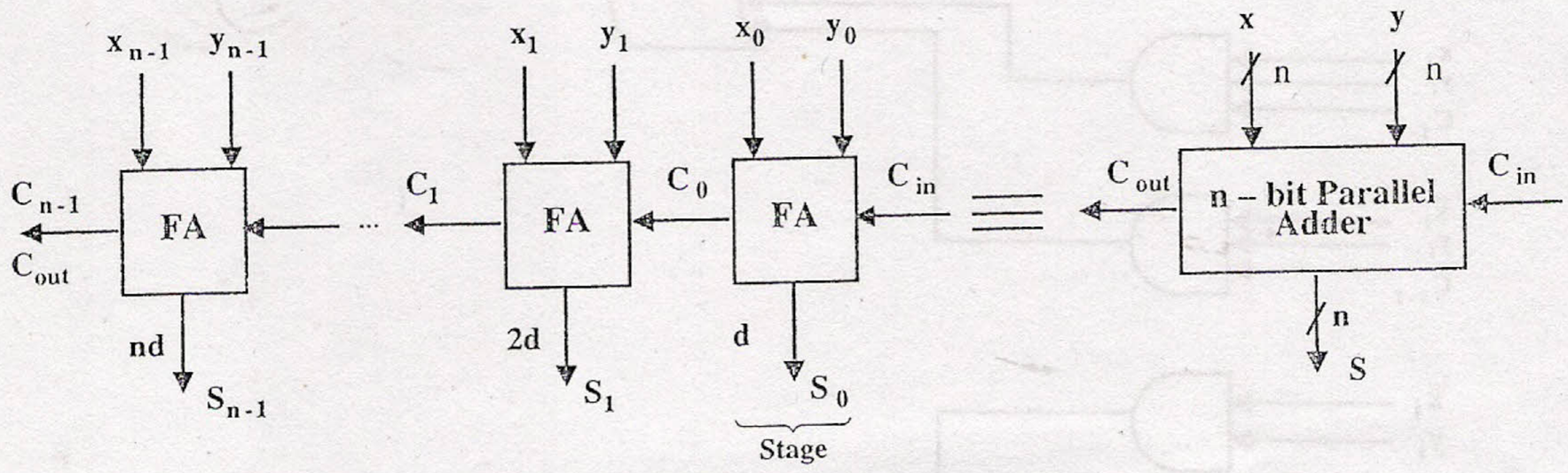
۲) در Serial Adder سخت افزار مستقل از تعداد بیت‌های اپراند ورودی است، ولی در F.A.، سخت افزار به تعداد بیت‌های اپراند ورودی بستگی خطی دارد.

جمع کننده موازی : (*Ripple Adder* یا *Parallel – Adder*)

مدار ترکیبی است که قادر است حاصل جمع دو عدد n بیتی را تولید کند و از اتصال n عدد F.A. به طور پشت سر هم به وجود می‌آید. به طوری که رقم نقلی خروجی از یک F.A. رقم نقلی ورودی به F.A. مجاور خواهد بود.

$X = x_{n-1}....x_2, x_1, x_0$

$Y = y_{n-1}.... y_2, y_1, y_0$



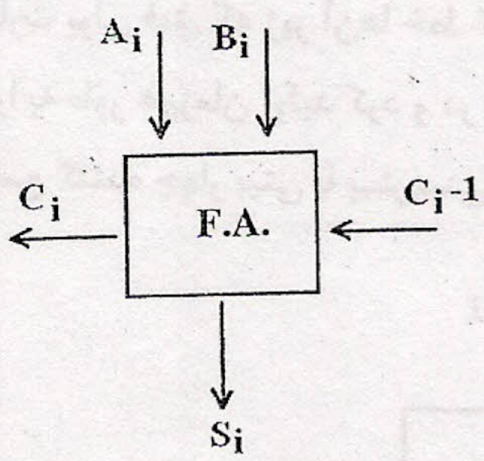
با توجه به نمودار فوق، متوجه می‌شویم، ممکن است هر Stage به Stage مجاور رقم نقلی تولید کند و هم‌چنین ممکن است رقم نقلی ورودی به یک Stage در داخل آن پخش شده و رقم نقلی در خروجی آن ظاهر شود. در بدترین حالت ممکن است رقم نقلی تولید شده از سمت راست ترین Stage تا سمت چپ‌ترین Stage به طور موجی پخش شود در این صورت مدار را n-Bit Carry Ripple Adder گویند. در این مدار حاصل جمع پس از تاخیر nd تولید خواهد شد.

جمع کننده سریع: *High Speed Binary Adder*

مهم‌ترین هدف در طراحی مدار جمع کننده‌های سریع تقلیل Carry Ripple Time (زمان تلف شده برای پخش موجی ارقام نقلی است) و در صورتی‌که ارقام نقلی ورودی به Stage ها فقط به متغیرهای ورودی از دو اپراند بستگی داشته باشد، آن‌گاه می‌توان ارقام نقلی ورودی به کلیه Stage ها را به طورهم‌زمان تولید نمود. و در نتیجه بیت‌های حاصل جمع را به صورت هم‌زمان تولید می‌شود.

جمع کننده با پیش بینی ارقام نقلی *(Carry Look ahead Adder)*:

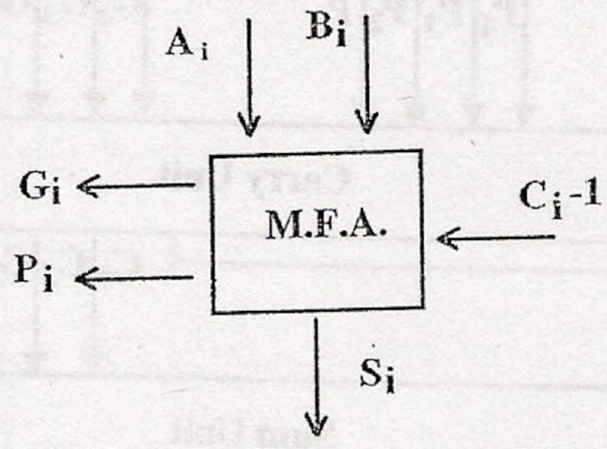
برای طراحی n-Bit Carry Lookahead Adder ابتدا رابطه بین رقم نقلی ورودی و خروجی به یک F.A. را بررسی می‌کنیم.



A_i	B_i	C_i	
O	O	O	
O	1	C_{i-1}	Propagated Carry: $P_i = A_i \oplus B_i$
1	O	C_{i-1}	
1	1	1	Generated Carry: $G_i = A_i . B_i$

پس طرح FA اصلاح شده (Modefied Full Adder) به صورت زیر خواهد بود:

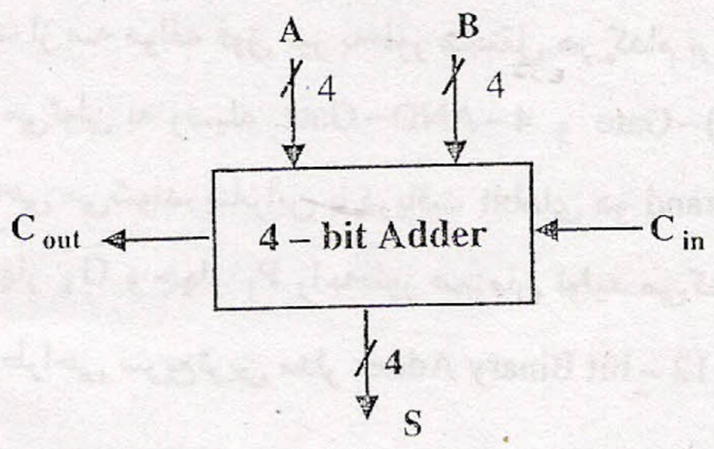
$$G_i = A_i B_i$$
$$P_i = A_i \oplus B_i$$
$$C_i = G_i + P_i C_{i-1}$$
$$S_i = P_i \oplus C_{i-1}$$



مثال: مطلوب است طراحی 4 – Bit Carry Lookahead Adder

$$C_4 \xleftarrow{C_3\ C_2\ C_1\ C_{in}} \begin{array}{l} A = A_4\ A_3\ A_2\ A_1 \\ \underline{B = B_4\ B_3\ B_2\ B_1} \\ S = S_4\ S_3\ S_2\ S_1 \end{array}$$

$$\begin{cases} C_1 = G_1 + C_{in} P_1 \\ C_2 = G_2 + C_1 P_2 \\ C_3 = G_3 + G_2 P_3 \\ C_4 = G_4 + C_3 P_4 \end{cases} \qquad \begin{cases} S = P \oplus C_{in} \\ S_2 = P_2 \oplus C_1 \\ S_3 = P_3 \oplus C_2 \\ S_4 = P_4 \oplus C_3 \end{cases}$$



با توجه به عبارات فوق متوجه می‌شویم که عبارت بولی C_i ها به طور بازگشتی تعریف شده‌اند. در صورتی‌که بتوانیم با بسط دادن عبارت بولی حالت بازگشتی بودن آنها را از بین ببریم، قادر خواهیم بود ۴ بیت رقم نقلی را به طور هم‌زمان تولید کنیم و چون P_i ها و G_iها به طور هم‌زمان در دست هستند، بنابراین می‌توان ۴ بیت حاصل‌جمع را به طور هم‌زمان تولید نمود.

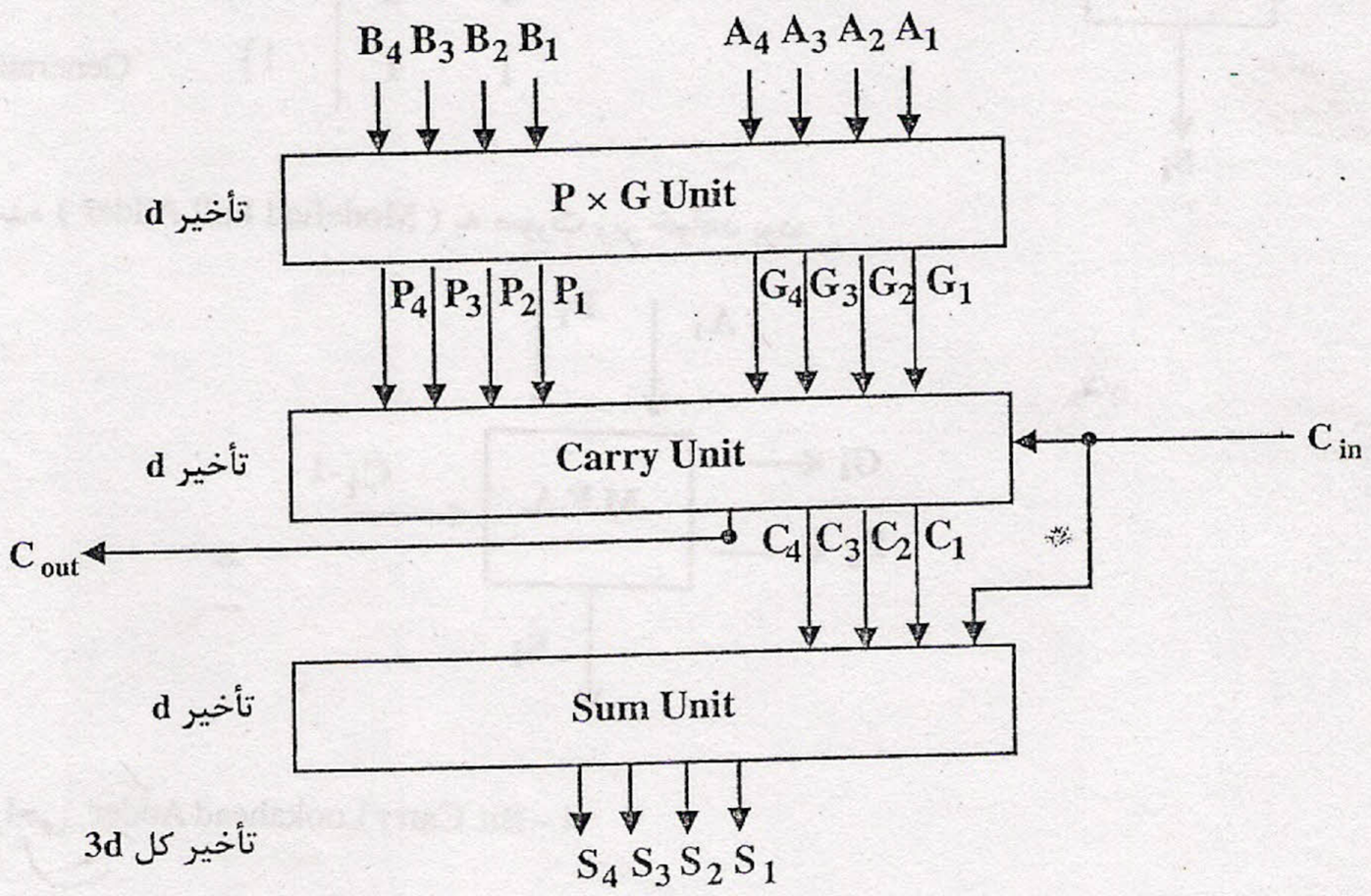
$$C_1 = \underline{G_1 + C_{in}P_1}$$

$$C_2 = G_2 + (G_1 + C_{in}P)P_2 = \underline{G_2 + G_1P_2 + C_{in}P_1P_2}$$

$$C_3 = G_3 + (G_2 + G_1P_1 + C_{in}P_1P_2)P_3 = \underline{G_3 + G_2P_3 + G_1P_2P_3 + C_{in}P_1P_2P_3}$$

$$C_4 = \underline{G_4 + G_3P_4 + G_2P_3P_4 + G_1P_2P_3P_4 + C_{in}P_1P_2P_3P_4}$$

با توجه به ۴ عبارت بولی فوق که زیر آن‌ها خط کشیده شده است، چون همگی تنها به P_i ها و G_i و C_{in} بستگی دارد، بنابراین می‌توان چهار رقم نقلی را به طور هم‌زمان تولید کرد و در نتیجه 4 بیت حاصل جمع طبق مدار زیر به صورت هم‌زمان تولید خواهد شد.
چنین مدار را جمع کننده چهار بیتی با پیش‌بینی ارقام نقلی گویند.



مدار 4 – bit Carry – look ahead Adder

مدار فوق به‌طور کامل بر روی یک IC وجود دارد. هم‌چنین هر یک از سه مولفه فوق نیز به‌طور مستقل هر کدام بر روی یک IC وجود دارد. با توجه به فرمول‌های صفحات قبل واحد P and G unit را می‌توان به وسیله $4-AND-Gate$ و $4-(X-OR)-Gate$ طرح نمود و چون IC ها به صورت یا تمام NAND و یا تمام NOR طراحی می‌شوند، بنابراین با دریافت bitهای دو Operand ورودی، بعد از تاخیر d واحد زمان (d تاخیر مدار دو سطحی فرض شده است) چهار G_i و چهار P_i را به‌طور هم‌زمان تولید می‌کند.

تمرین – با استفاده از مفهوم Carry – Lookahead، مطلوب است طراحی سریع‌ترین مدار 12 – bit Binary Adder.