به نام خدا

تحقیق معماری کامپیوتر

((جمع کننده ها))

استاد : دکتر زرندی

سید محمد مهدی احمدپناه 9031806 / حمیدرضا رمضانی 9031062

اردی بهشت 1392

مباحث پایه جمع کننده ها :

در ابتدا با پیاده سازی half adder آشنا شدیم که به صورت زیر است:

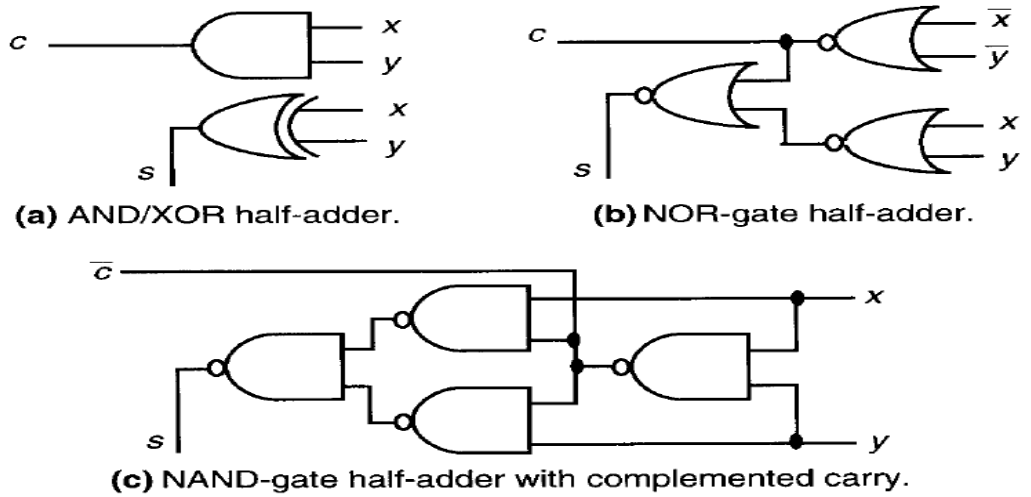


Fig. 5.1 Three implementations of a half-adder.

سپس با نحوه پیاده سازی یک full adder :

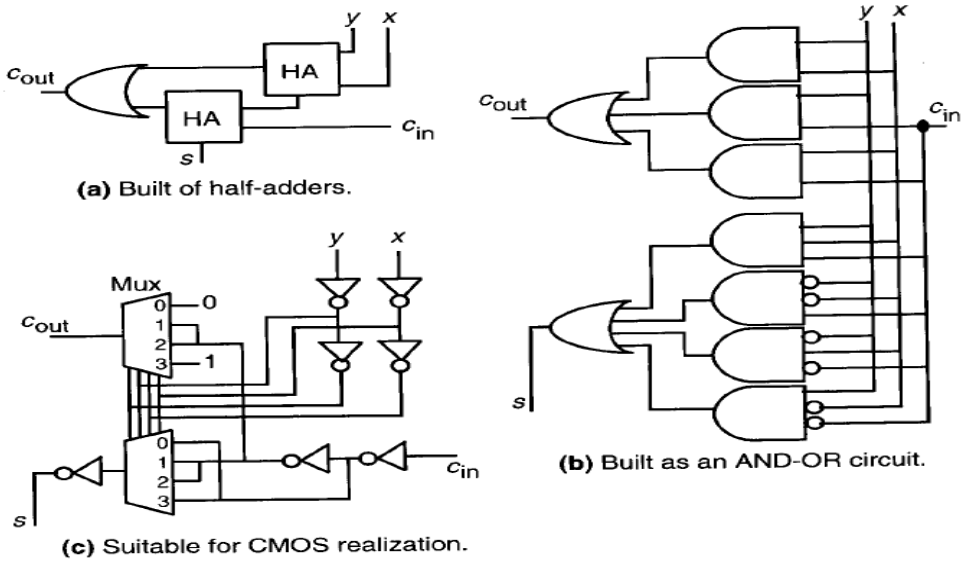


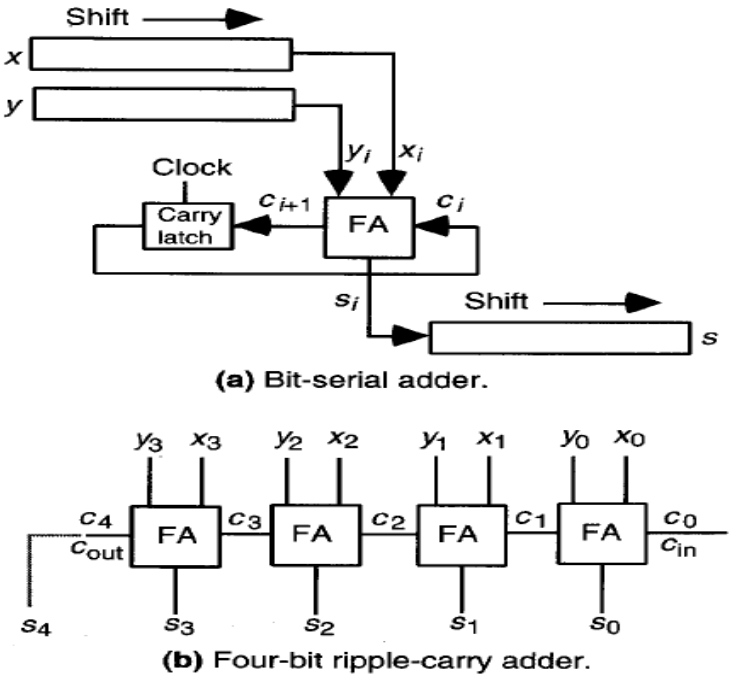
Fig. 5.2 Possible designs for a full adder in terms of half-adders, logic gates, and CMOS transmission gates.

در پیاده سازی a اگر از half adder با پیاده سازی nand استفاده کنیم باید به جای or از nand استفاده کنیم. که مدار nand only میشود. همچنین یک full adderرا می توانیم در دو سطح nand-nand و and-or پیاده سازی کنیم.

در مرحله بعد با پیاده سازی یک Bit Serial Adderآشنا شدیم که به صورت زیر کار می کند:

با هر کلاک بیت های اول X , Y توسط شیفت رجیستر وارد FA می شوند و با هم جمع می شوند. و حاصل در شیفت رجیستر دیگری ریخته می شود. مانند شکل بعدی:

همینطور با پیاده سازی یک Ripple Adder که ساده ترین و کندترین طراحی است آشنا شدیم که به صورت زیر است:



(a) Bit-serial adder.

(b) Four-bit ripple-carry adder.

Fig. 5.3 Using full adders in building bit-serial and ripple-carry adders.

اگر جمع کننده آبشاری را با تکنولوژی CMOS یعنی با FA پیاده سازی شده در شکل c 5.2 پیاده سازی کنیم مساحت مصرفی آن به λ^2 (نصف ابعاد پیاده سازی با حالت های دیگر) کاهش می یابد.

مسیر بحرانی در یک جمع کننده آبشاری عبارت است از Cinتا Cout چون بیشترین تاخیر را دارد. رابطه تاخیر برای یک جمع کننده آبشاری به صورت زیر است:

$$T_{\text{ripple-add}} = T_{\text{FA}}(x, y \rightarrow c_{\text{out}}) + (k-2) \times T_{\text{FA}}(c_{\text{in}} \rightarrow c_{\text{out}}) + T_{\text{FA}}(c_{\text{in}} \rightarrow s)$$

که در بدترین حالت برابر با 2kd می شود که برای kهای بزرگ خوب نیست. تاخیر Bit serial Adder هم O(K) است چون علاوه بر FA باید منتظر Clockهم باشیم.

همچنین FA , HA به قدری قوی هستند که می توانیم از آنها برای محاسبه توابع غیر حسابی استفاده کنیم.

همانند شکل درج شده در زیر علاوه بر حاصل جمع باید پرچم های Overflow , Negative , Cout و zero را هم بدست بیاوریم. که هر کدام از این پرچمها کاربردهای خود را دارند.

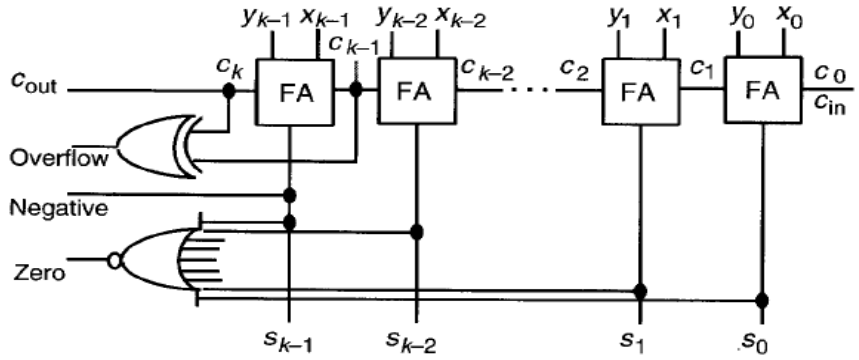


Fig. 5.7 Two's-complement adder with provisions for detecting conditions and exceptions.

با مفهوم زنجیره Carry آشنا شدیم که به این معناست که از زمانی که Carryتولید میشود و تا زمانی که حاصل جمع صفر باشد و Carry out داشته باشیم را زنجیره carryگوییم. زنجیره Carryتعیین کننده تاخیر جمع کننده است. همچنین احتمال اینکه Carryدر مکان i ام تولید شود و تا خانه j ام برود برابر با 2^{-j+i} است. همچنین اثبات شده که در حالت متوسط طول زنجیره برابر با Log k است. جمع کننده Carry Completion detection با استفاده از متوسط طول زنجیره تاخیر را به O(Log k) است.در ابتدا ، همه Carry ها ناشناخته (Unknown) هستند و با استفاده از رابطه زیر Carry out را تعیین می کنیم و به زنجیره انتشار Carryها تزریق می شوند.

$$(b_{i+1}, c_{i+1}) = (\overline{x_i + y_i}, x_i y_i)$$

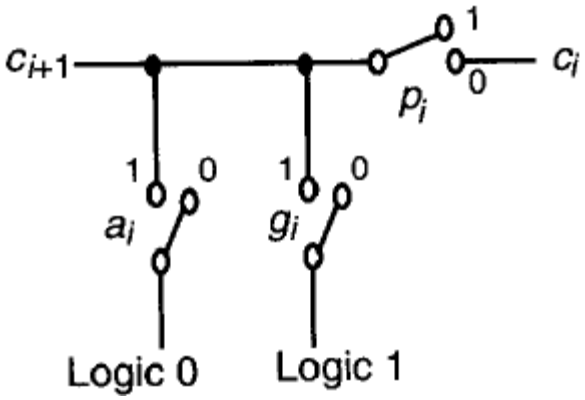
با یک AND کلی از همه di = bi + ci ها می توان فهمید کار تشخیص Carry پایان یافته یا نه (all done signal).

باید مراقب hazard هم باشیم.در بهترین حالت تاخیر جمع کننده k بیتی برابر با یک گیت است در زمانی که هیچ انتشار نقلی نداشته باشیم و در بدترین حالت 2k+1 گیت که یعنی همه نقلی ها از Cin به Cout برود. و در حالت متوسط تاخیر بصورت 2log k + 1 گیت است.چون تاخیر به اطلاعات بستگی دارد پس برای سیستم ها آسنکرون (ناهمگام) بهتر است.

وقتی ورودی یک عمل جمع ثابت باشد طراحی مدار آن راحت تر و بهینه تر است . پس شمارنده ها را می توان سریعتر از جمع کننده های عمومی ساخت.

چنین می توان تعریف کرد:

$$g_i = x_i y_i$$
$$p_i = x_i \oplus y_i \qquad s_i = x_i \oplus y_i \oplus c_i$$
$$a_i = \overline{x}_i \overline{y}_i = \overline{x_i + y_i}$$



Manchester Adder از gi و pi و ai و ... استفاده می کند و یک مدار جمع کننده طراحی می کند.در هر لحظه فقط یکی از کلیدهای ai و gi و pi بسته خواهد بود.

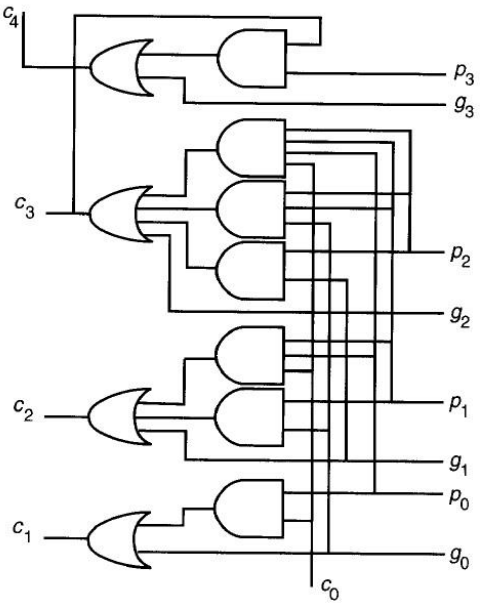
Manchester Carry Chain نوعی از Carry lookahead است که با استفاده از logic اشتراکی ، از تعداد ترانزیستورهای مصرفی می کاهد. تاخیر مدار هم به سه قسمت تقسیم می شود : زمانی که کنترل سیگنال کلیدها را آماده کند ، زمان برپاکردن کلیدها ، تاخیر انتشار سیگنال که از k کلید می گذرد.

در حالتی که O(k) باشد از Ripple Adder سریعتر است چون فقط یک تاخیر کلید داریم بجای اینکه دو تاخیر گیت داشته باشیم.بخاطر همین بیشتر در زنجیره های کوتاه کاربرد دارد.

جمع کننده های پیش بینی کننده نقلی:

برای بزرگتر از 4 بیت غیرعملی است.برای پیاده سازی word های بزرگ می توان دو کار کرد:

اول اینکه مبنای جمع را زیاد کنیم (تاخیر زیاد و مدار ساده) ، و دوم اینکه پیش بینی هایمان را چند سطحه کنیم (عملی است)



از روابط زیر استفاده می کنیم:

تحلیل زمانی مدار :

تولید سیگنالهای g و p برای بلوک 4 بیتی (یک گیت)

پیشبینی سیگنالهای Carry in،C4،C8،C12 برای بلوکها (دوگیت)

پیش بینی نقلی های داخلی در هر بلوک چهار بیتی (دو گیت)

محاسبه حاصل جمع بیت ها (دو گیت)

$$g_{[i,i+3]} = g_{i+3} + g_{i+2}p_{i+3} + g_{i+1}p_{i+2}p_{i+3} + g_i p_{i+1} p_{i+2} p_{i+3}$$

$$p_{[i,i+3]} = p_i p_{i+1} p_{i+2} p_{i+3}$$

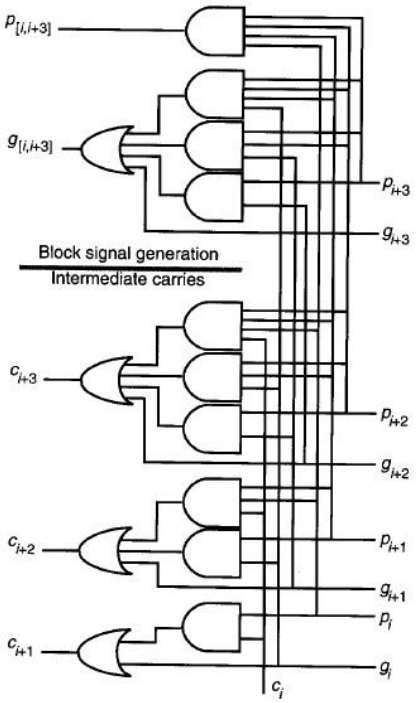
در مدار صفحه بعد ، نمای کلی یک CLA را داریم و :

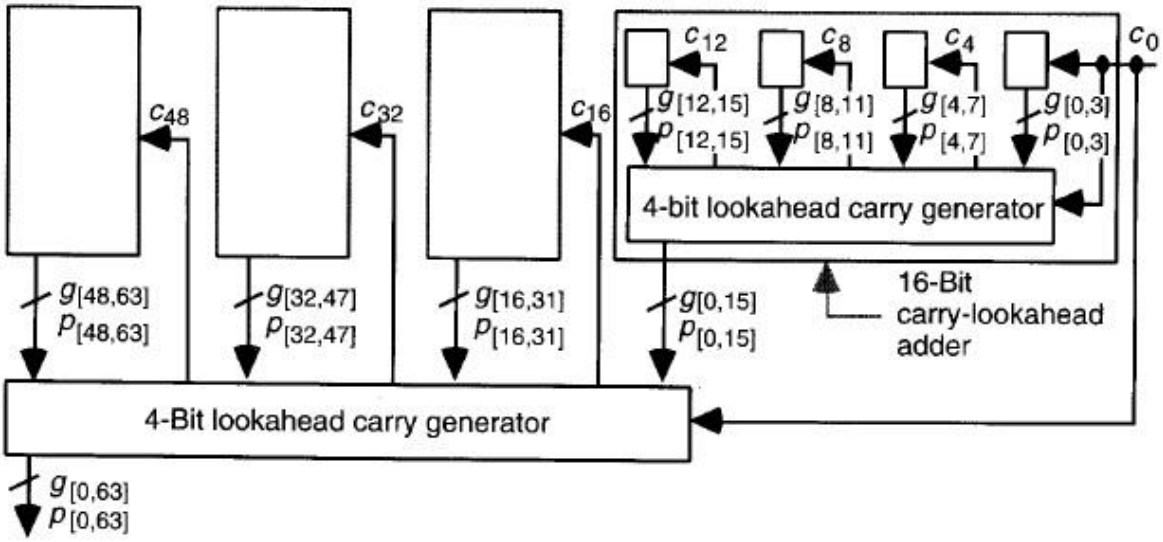
$$T_{\text{lookahead-add}} = 4 \log_4 k + 1 \text{ gate levels}$$

می توان بجای بلوکهای چهار بیتی از بلوکهای شش بیتی یا هشت بیتی استفاده کنیم که این باعث می شود که تاخیر زیاد شود و تعداد بلوکها کم.از طرفی fan-in زیاد می شود و هزینه افزایش می یابد.

جمع کننده Ling یک نوع از CLA است که در سخت افزار کاهش بسیاری یافته است.اصلاح Ling شامل انتشار hi بجای ci است.

$$h_i = c_i + c_{i-1}$$



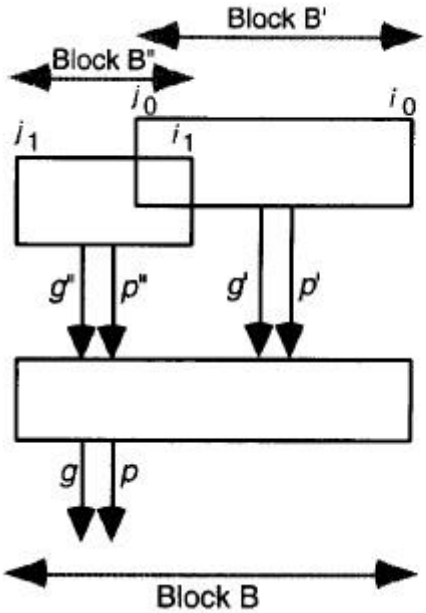


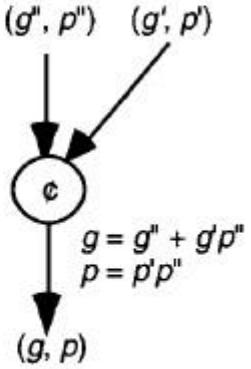
یعنی خواهیم داشت :

$$\begin{aligned} s_i &= p_i \oplus c_i \\ &= p_i \oplus h_i t_{i-1} \\ &= (t_i \oplus h_{i+1}) + h_i\ g_i\ t_{i-1} \end{aligned}$$

تولید و انتشار سیگنال ها برای بلوک ادغام شده B می تواند از رابطه زیر به دست آید:

$$\begin{aligned} g &= g'' + g' p'' \\ p &= p' p'' \end{aligned}$$

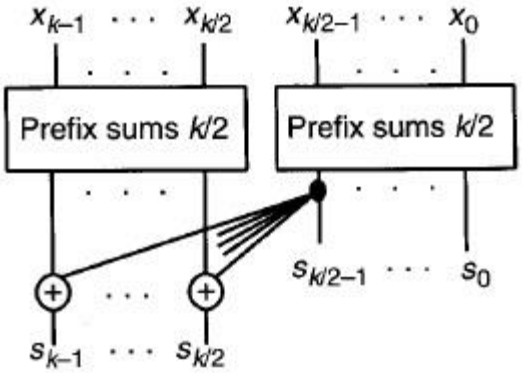




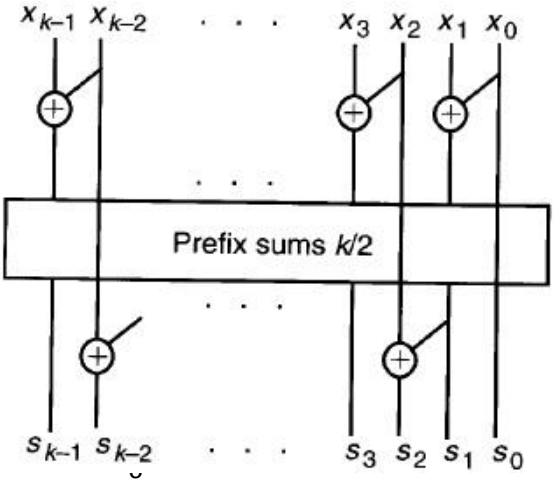
ما می توانیم از استراتژی های گوناگونی برای ترکیب کردن حاصل جمع پیشوندی موازی استفاده کنیم.

تاخیر مدار پایین صفحه: log k

هزینه : (k/2) log k



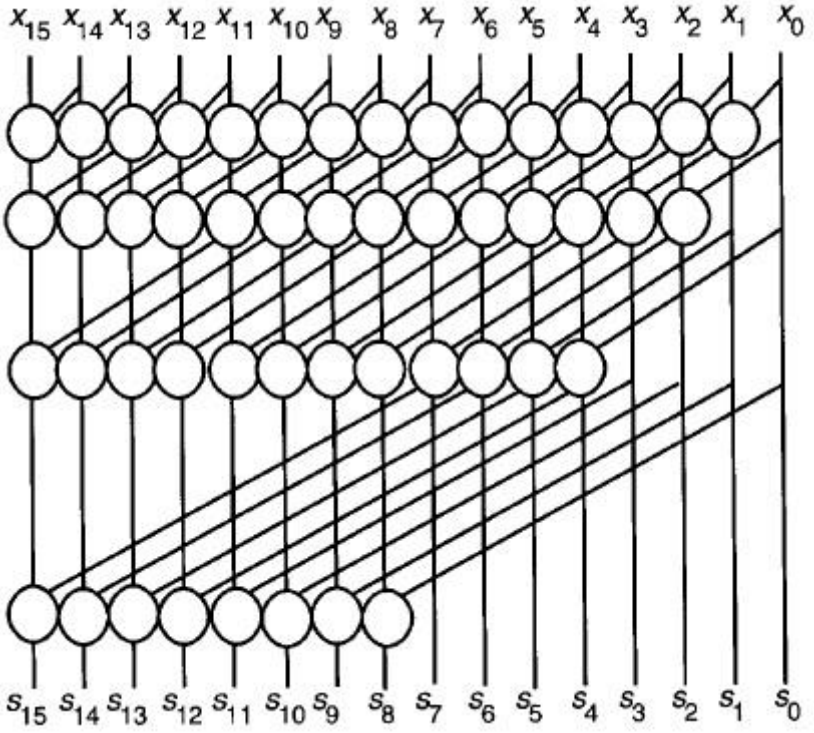
$(g, p) = (g', p')¢(g'', p'')$ means $g = g'' + g' p'', p = p' p''$



یک طراحی دیگر که آن نیز بر پایه تقسیم و غلبه است ، در مدار روبرو است.

تاخیر این مدار 2logk – 1 است و هزینه آن 2k – logk – 2 است.

پس ، طراحی اول سریعتر و گرانتر است .مدار پایین صفحه قبل سمت راست بعنوان گراف پیشوندی موازی Brent – Kung معروف است.



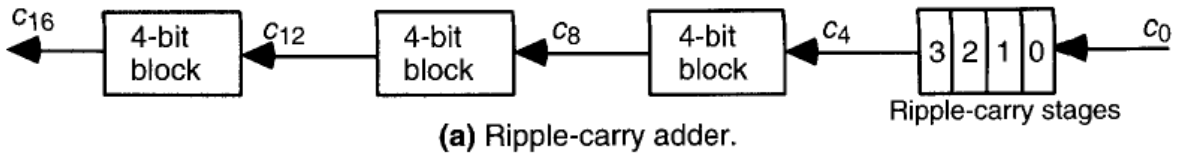
شکل مقابل یک گراف پیشوندی موازی Kogg – Stone است.این طراحی سریعترین پیاده سازی ممکن یک محاسبه پیشوندی موازی است.(اگر فقط بلوکهای دارای دو ورودی مجاز باشد)اما هزینه آن می تواند برای k های بزرگ گران باشد.

می توان ترکیبی از این دو گراف (طراحی هیبریدی)را برای طراحی شبکه پیشوندی موازی پیشنهاد داد.

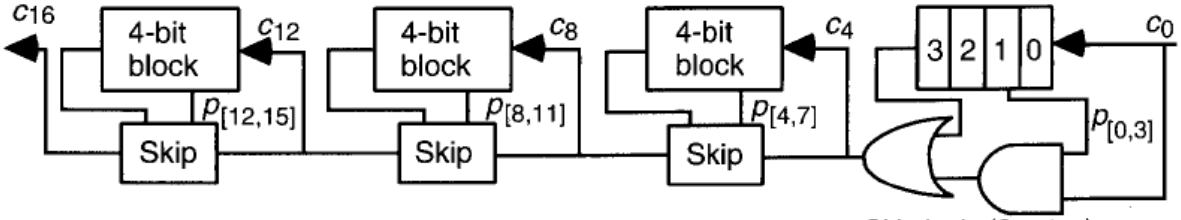
تئوری گرافهای پیشوندی موازی نسبتا غنی و به خوبی توسعه یافته است. اینها برای پیاده سازی VLSI مناسب است ممکن است چنین تلقی شود که برای طراحی های با کارایی بالا خیلی کند است. می توان طراحی های دیگری را نیز پیشنهاد داد.

انواع جمع کننده های سریع:

مدار زیر یک جمع کننده Carry – skip ساده را نشان می دهد.

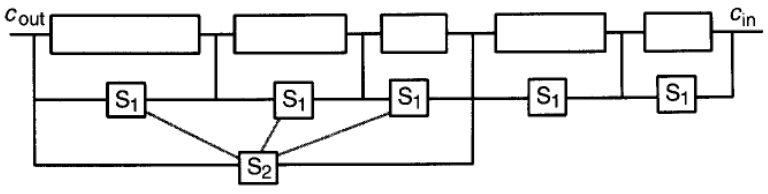


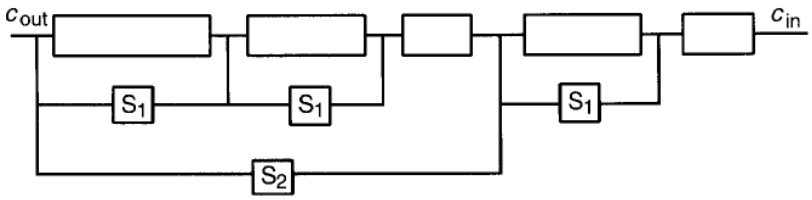
(a) Ripple-carry adder.



(b) Simple carry-skip adder.

می توان از جمع کننده Multi-Level Carry-skip استفاده کنیم.





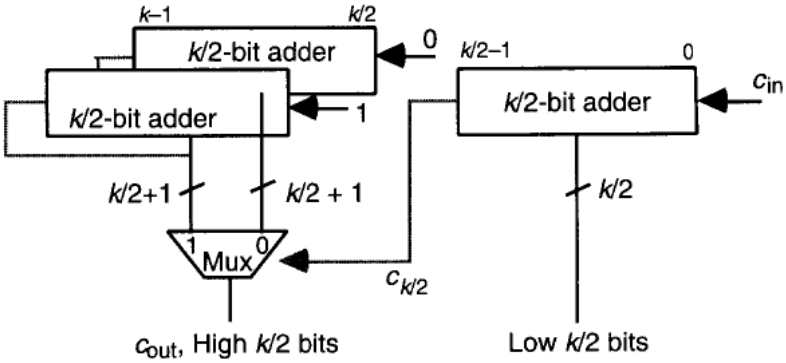
می توان مدار بالا را به شکل مقابل بهبود داد:

جمع کننده Carry – select نوع دیگری از

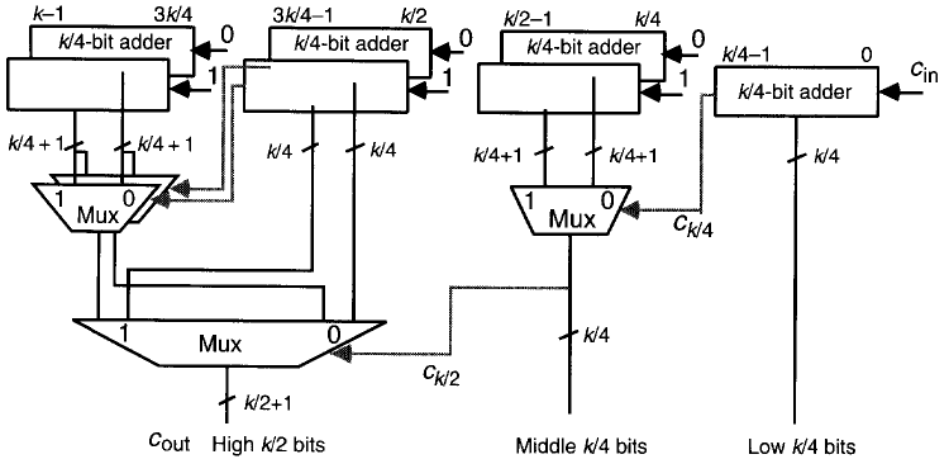
جمع کننده های سریع است که تاخیر و هزینه آن به شکل زیر است:

$$C_{\text{select-add}}(k) = 3C_{\text{add}}(k/2) + k/2 + 1$$

$$T_{\text{select-add}}(k) = T_{\text{add}}(k/2) + 1$$



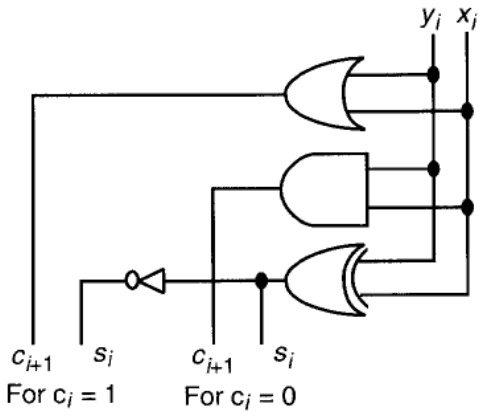
می توان اینها را نیز چند سطحه کرد:



جمع کننده Conditional – sum نیز گونه دیگری از این جمع کننده ها هستند:

$$C(k) \approx 2C(k/2) + k + 2 \approx k(\log_2 k + 2) + kC(1)$$

$$T(k) = T(k/2) + 1 = \log_2 k + T(1)$$



دسته دیگری از جمع کننده ها ، جمع کننده های هیبریدی اند که در شکل های زیر دو مداراز آنها آمده است:

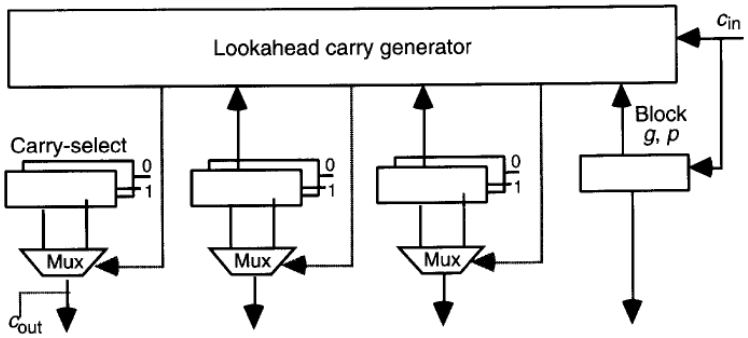


Fig. 7.12 A hybrid carry-lookahead/carry-select adder.

نمودار زیر هم تاخیر برای بیتهای مختلف است :

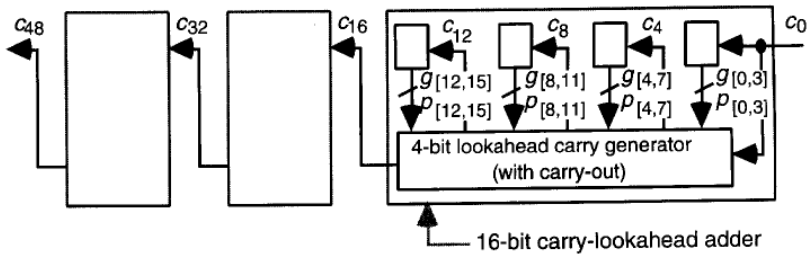
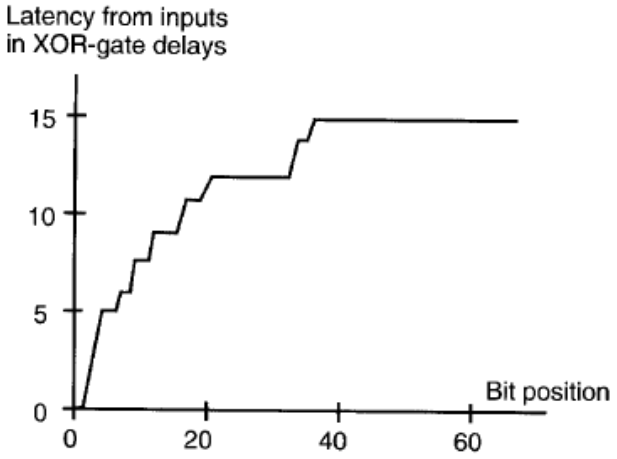


Fig. 7.13 Example 48-bit adder with hybrid ripple-carry/carry-lookahead design.

می توان این جمع کننده های سریع را بهینه کرد.



جمع کننده های چند عملوندی:

درخت carry select adder شکل 8.12 جمع 7 عملوند k بیتی را به جمع 2 عملوند k+2 بیتی کاهش می دهند. این درخت به درخت Wallace معروف است. در حالت کلی یک درخت Wallace با n ورودی k بیتی می تواند دو خروجی k+log n -1 بیتی تولید کند. از آنجا که هر CSA تعداد عملوند ها را با ضریب 1.5 کاهش می دهد، کوتاهترین ارتفاع این درخت با n عملوند از رابطه ی بازگشتی زیر محاسبه می گردد.

$$h(n) = 1 + h(\lceil 2n/3 \rceil)$$

در جمع کننده های لگاریتمی سریع تاخیر تابعی از طول عدد ورودی نیست برای مثال یک جمع کننده ی look ahead برای ورودی های با طول بین 17 تا 32 تاخیر مشابهی دارد.

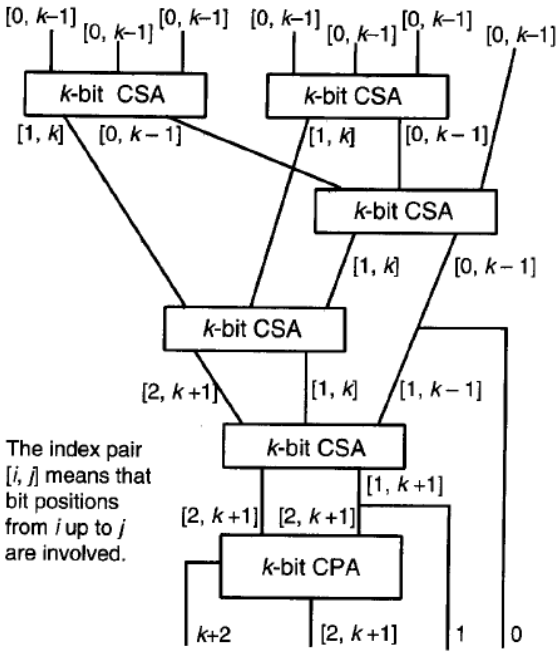
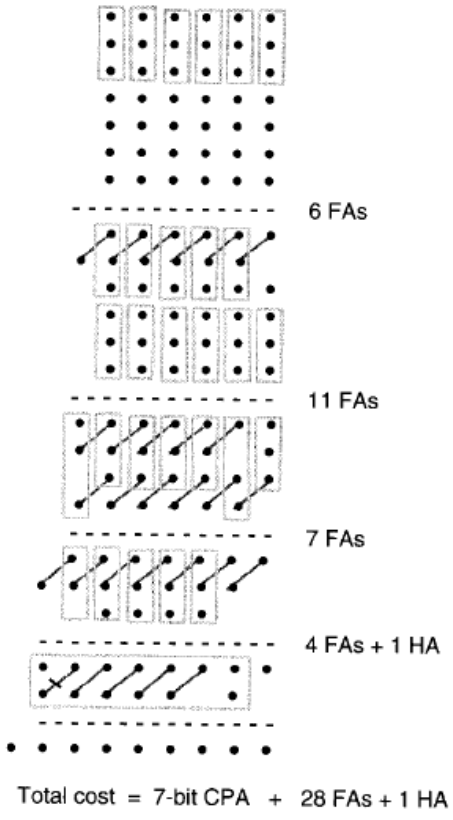


Fig. 8.12 Adding seven k-bit numbers and the CSA/CPA widths required.

TABLE 8.1
The maximum number $n(h)$ of inputs for an h-level carry-save-adder tree

h	$n(h)$	h	$n(h)$	h	$n(h)$
0	2	7	28	14	474
1	3	8	42	15	711
2	4	9	63	16	1066
3	6	10	94	17	1599
4	9	11	141	18	2398
5	13	12	211	19	3597
6	19	13	316	20	5395

در درخت Dadda ما تعداد عملوند ها را به کمترین عدد بعدی در جدول 8.1 که دارای کمترین تعداد FA و HA است کاهش می دهیم. تعداد عملوند های مطلوب 8،7، یا 9 است که با 4 مرحله carry select adder انجام می شوند و دیگر نیازی به کاهش تعداد عملوند ها به کمترین n در جدول نیست زیرا درخت سریع تری تولید نمی کند.



یک full adder تک بیتی می تواند به عنوان یک counter (3;2) استفاده شود. به این معنی که تعداد 1 ها را در بین سه ورودی شمرده و نتیجه را در 2 بیت نمایش دهد. با عمومیت دادن این روش می توان یک (n;[log(n+1)]) ساخت.

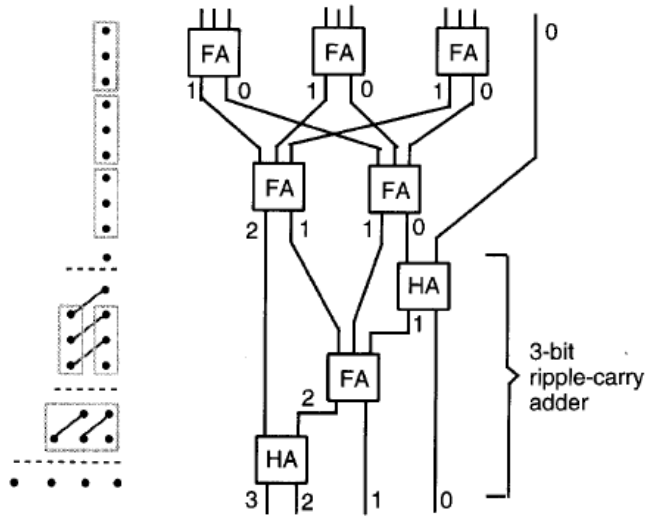


Fig. 8.16 A 10-input parallel counter also known as a (10; 4)-counter.

منابع :

1) B. Parhami, Computer Arithmetic - Algorithms and Hardware Designs, 2000

2) D.M. Harris, Digital Design and Computer Architecture

3) جزوه درس معماری کامپیوتر ، دکتر زرندی

4) ویکی پدیا