به نام خدا

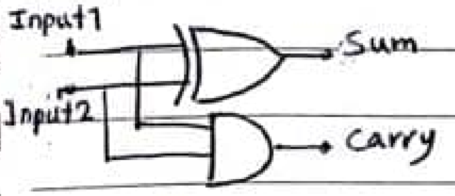
گزارش کار آزمایش پنجم معماری کامپیوتر:

یاسمن نوذری - ریحانه خامحمدی

9931067 9931100

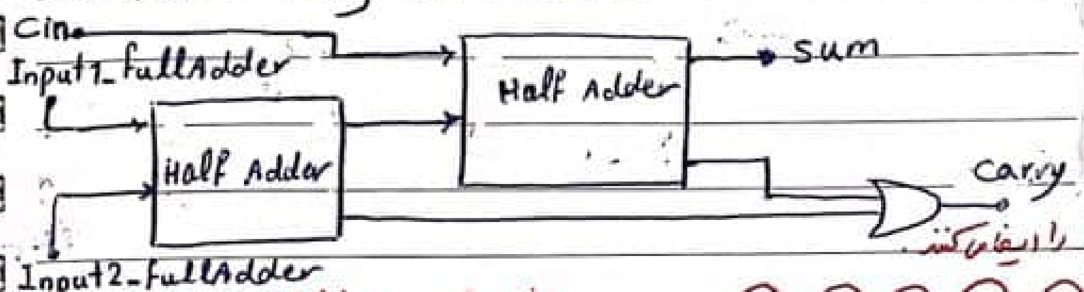
اگر بخواهیم درباره جمع‌کننده‌ها صحبت کنیم در ابتدا باید به ساختار درونی Half Adder و Full Adder اشاره کنیم:

Half Adder:



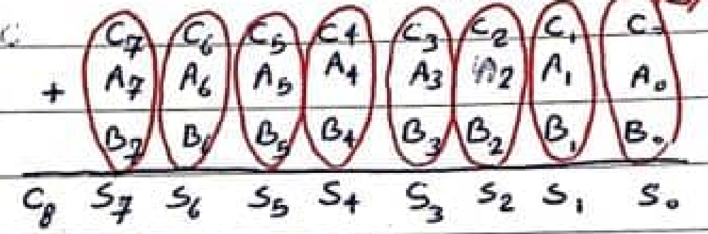
همانطور که مشاهده می‌شود در جمع دو تک bit ؛ XOR آن‌ها sum را و AND شان ؛ carry را حاصل می‌کند. اما این کافی نیست!

Full Adder using Half Adder:

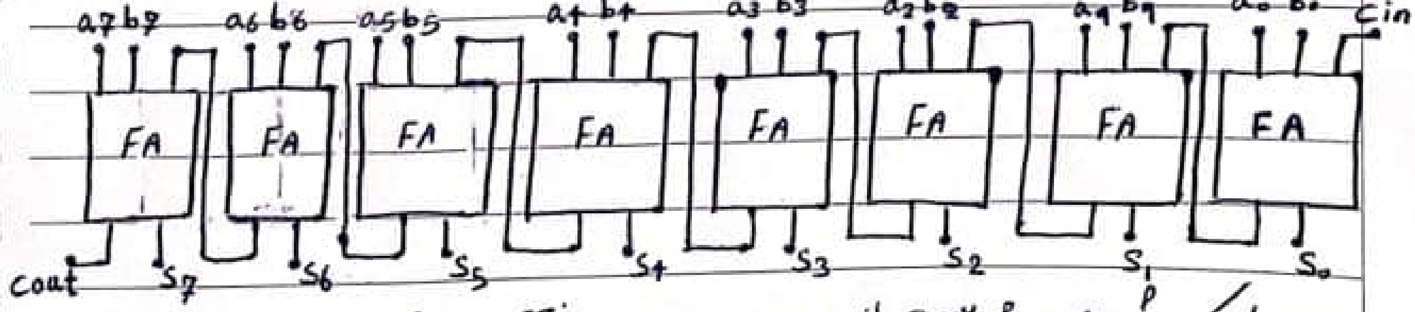


هر یک عملکرد یک full adder را ایفا می‌کند.

Ripple carry Adder: (8 bit)



در جمع دو عدد 8 bit در بالا نیز مشاهده می‌شود که با شروع جمع ؛ در هر جمع یک carry از جمع‌های پیشین انجام شده به جمع بعدی افزوده می‌شود.



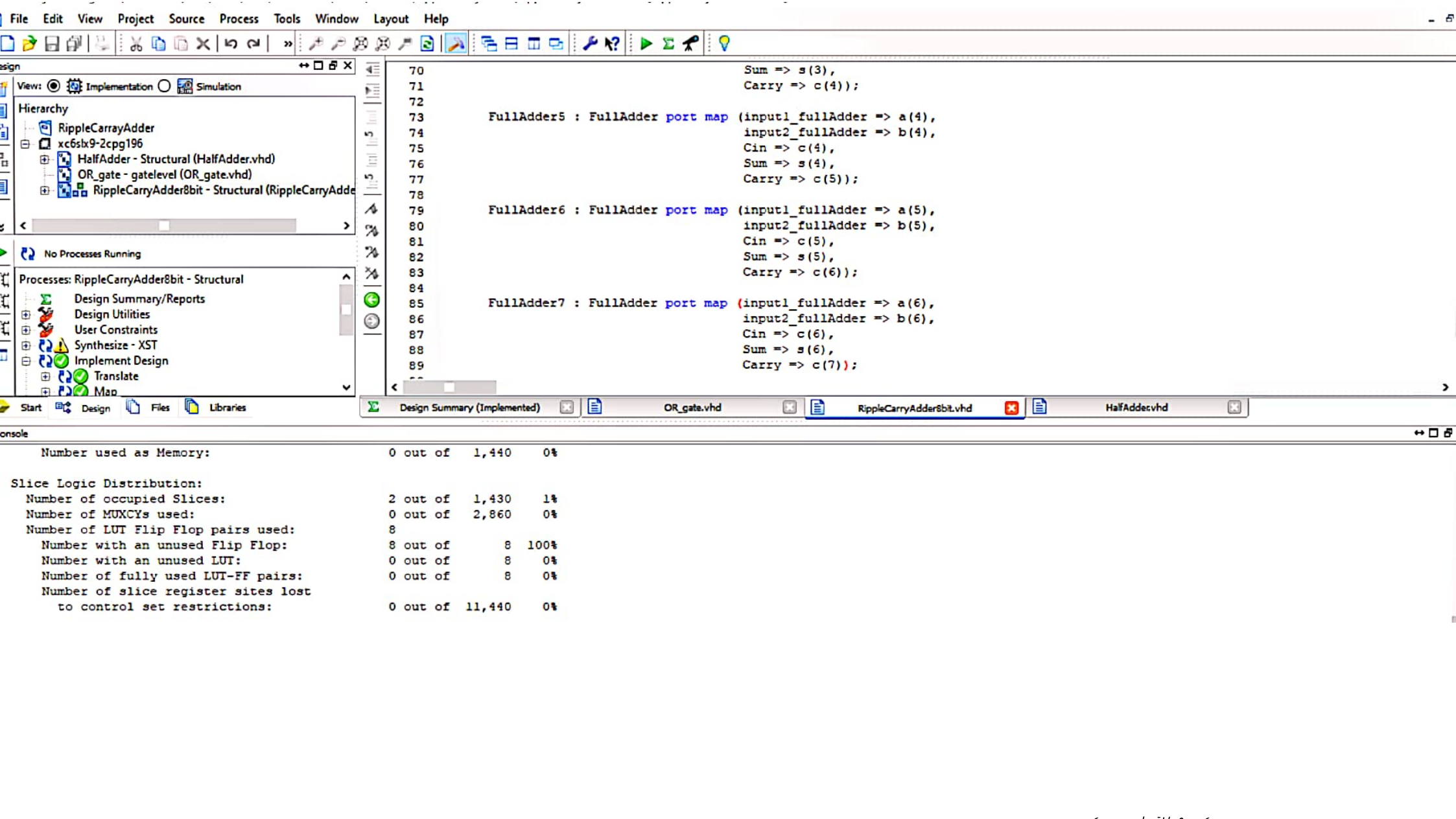
همانطور که مشاهده می‌شود carry از هر مرحله به مرحله بعد منتقل می‌شود و این موجب تاخیری در محاسبات می‌شود تا به Full Adder بعدی منتقل شود بنابراین اگر تاخیر در هر مرحله t در نظر گرفته شود. ← تاخیر هر FA برابر با 2t می‌شود. بنابراین برای جمع دو عدد 8 بیتی:

$2t \times 8 = 16t$

این روند شامل محاسبه carry و یک sum می‌باشد بنابراین تاخیر به طور کلی:

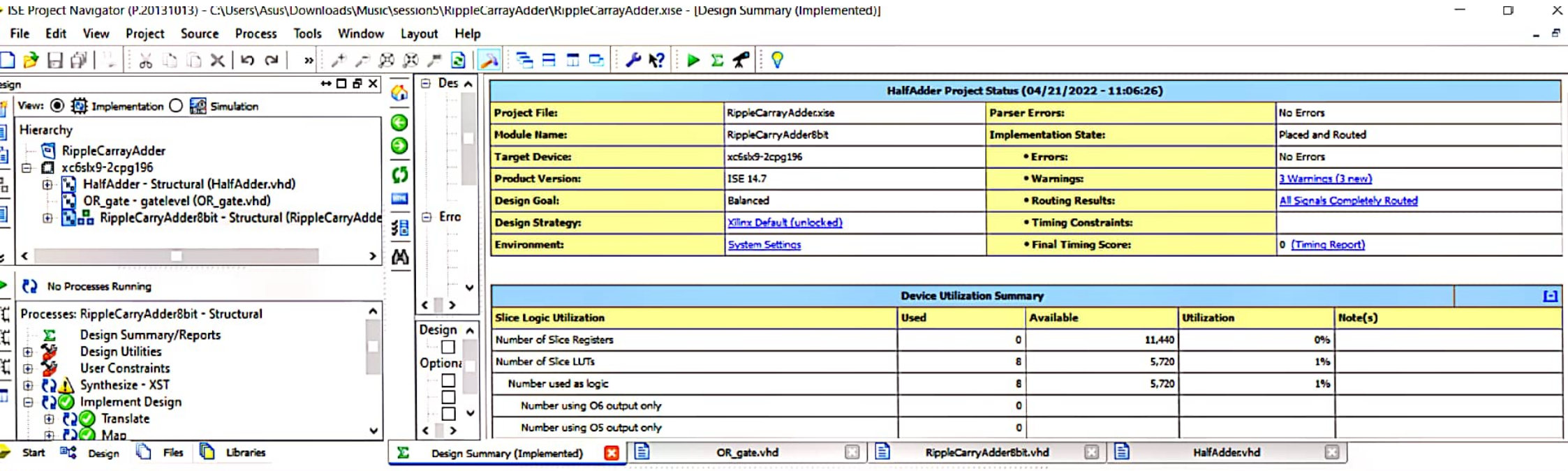
delay (carry) = $2 \times 8d = 16d$

Scanned with CamScanner




```
    Number used as Memory:                0 out of   1,440    0%

Slice Logic Distribution:
  Number of occupied Slices:              2 out of   1,430    1%
  Number of MUXCYs used:                  0 out of   2,860    0%
  Number of LUT Flip Flop pairs used:     8
    Number with an unused Flip Flop:      8 out of       8  100%
    Number with an unused LUT:            0 out of       8    0%
    Number of fully used LUT-FF pairs:    0 out of       8    0%
    Number of slice register sites lost
      to control set restrictions:        0 out of  11,440    0%
```




```
Asynchronous Control Signals Information:
----------------------------------------
No asynchronous control signals found in this design

Timing Summary:
---------------
Speed Grade: -2

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 52.393ns
```

Carry look ahead :

در Ripple adder سرعت بسیار پایین بود و ما تا Cout مرحله قبل نمی‌رسید باید صبر می‌کردیم

پس یک روش دیگر که هست Carry look ahead ارائه شد. در این روش ما

برای هر بیت منتظر Cout بیت قبلی نیستیم و مستقلا می‌توان Cout هر بیت را به دست

آورد. در واقع FA های این adder به صورت موازی عمل می‌کنند. برای این کار

دو تابع مهم زیر را تعریف می‌کنیم:

Carry generate $\rightarrow g_i = A_i B_i$

Carry propagate $\rightarrow P_i = A_i \text{ xor } B_i$

بنابراین برای Sum و Carry out داریم:

$$Sum_i = A_i \text{ xor } B_i \text{ xor } C_i = P_i \text{ xor } C_i$$

$$C_{i+1} = A_i B_i + A_i C_i + B_i C_i = g_i + C_i P_i$$

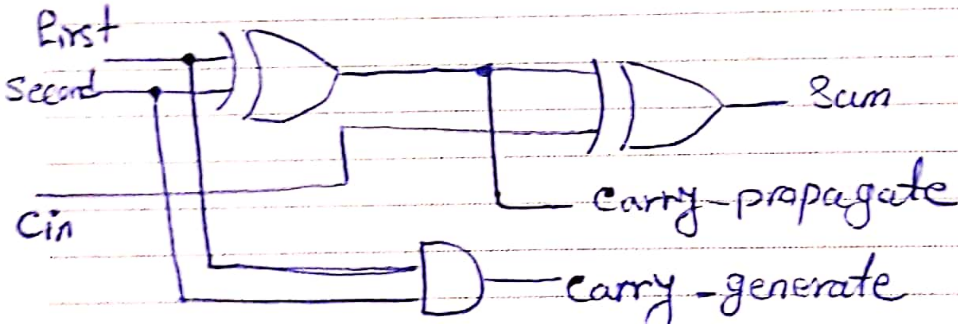
طبق رابطه‌های بالا می‌توان در لحظه اول C_i را برای هر بیت حساب کرد

اما مشکل مهم این است که در عمل اگر تعداد بیت عدد بسیار بالا باشد مثلا ۸ بیت

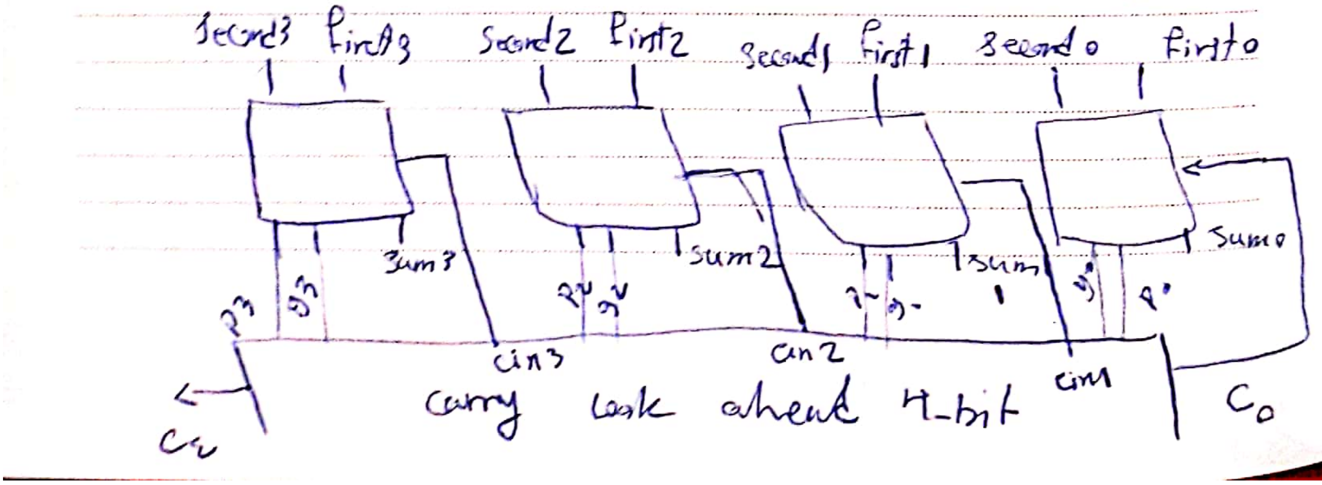
وسعت و فضای اشغال شده توسط مدار ها بسیار بالا می رود برای اینکه Cout های مربوط به بیت های آخر از تعداد خیلی بالای عملیات منطقی که دارای تعداد خیلی زیادی گیت های And و OR هستند که آنها بسیار بالا می رود تشکیل شده است.

درست

برای پیاده سازی CLA ۸ بیتی از آنجایی که فضا و تعداد گیت های زیادی را مصرف می کند یک راه جدید را استفاده می کنیم برای این کار ابتدا یک Full adder که بتواند p و g را نیز تولید کند را تعریف می کنیم. شماتیک مدار به صورت زیر است:



حالا برای طراحی cla از ۴ تا Full adder بالا استفاده می کنیم تا sum و p و g تولید شود از طرفی p و g را به یک cla ۴ بیتی می دهیم تا cin های بعدی را درست کند. به صورت زیر است:



هنگامی که یک cla n بیتی را به چند تا cla k بیتی تبدیل کنیم در این صورت تاخیرها به صورت زیر است:

delay Carry → $3(n/k)\,a$ → تاخیر هر گیت

delay Sum → $3(n/k+1)\,a$

پس حالا که ما یک cla ۸ بیتی را به 2 تا cla ۴ بیتی تقسیم می‌کنیم داریم:

Carry → $3a$ Sum → $4a$ ⇒ Cost = $48a$ (۴۸ گیت)

Cost → $3n + \frac{n(n+3)}{2}$

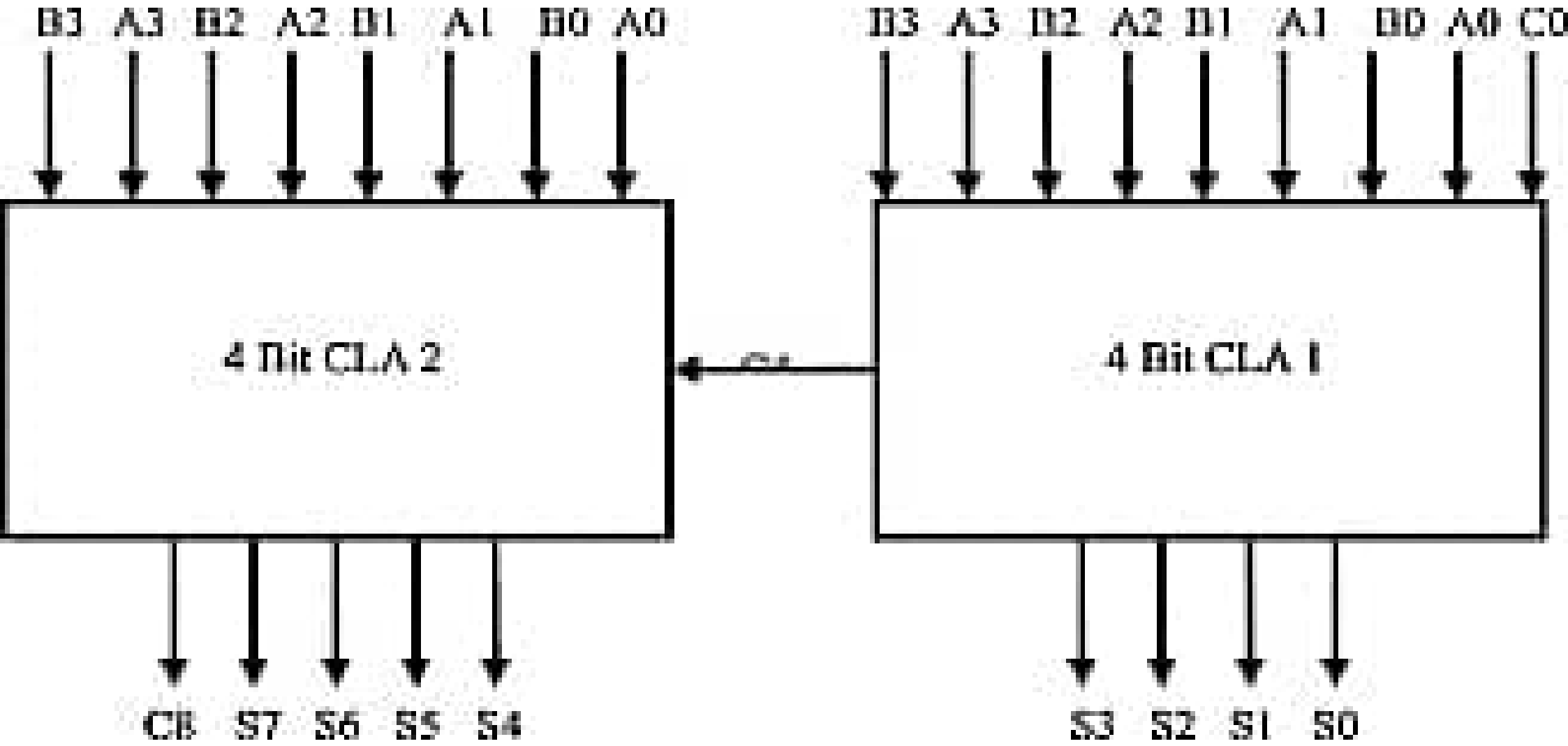
شماتیک مدار به صورت زیر است:

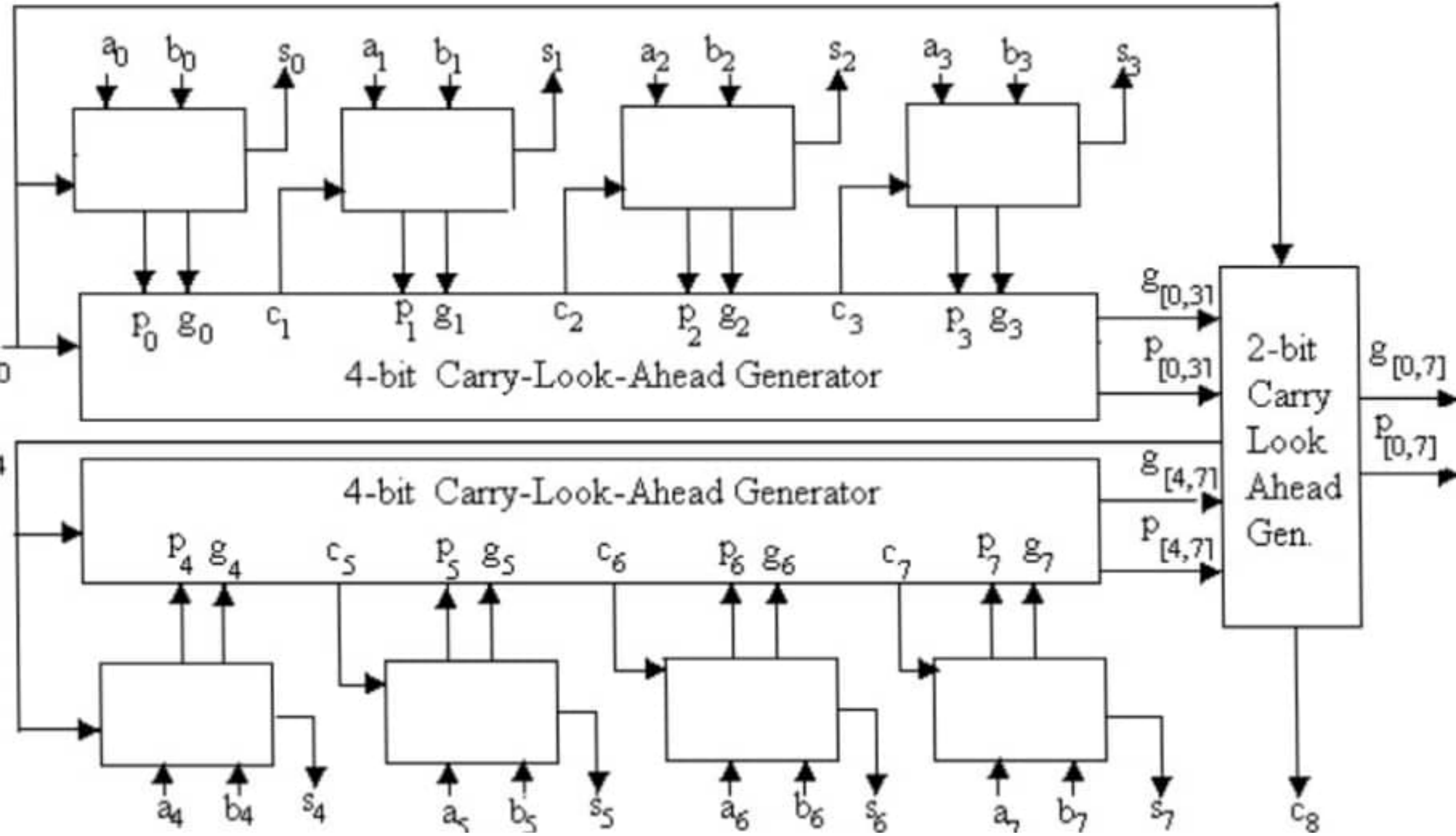
$$C_1 = g_0 + p_0 c_0$$

$$C_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$$

$$C_3 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$$

$$C_4 = g_3 + p_3 g_2 + p_3 g_1 p_2 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0$$





Carry select adder

در مثال قبل دیدیم که هر طبقه cla باید منتظر carry طبقه قبلی باشد. برای حل

این مشکل از CSA استفاده می‌کنیم و منطق آن به این صورت است که هر طبقه

همزمان با بقیه انجام شده و از هر دو مقدار sum، carry را انجام می‌دهیم و بعد با

استفاده از یک Mux با توجه به carry مرحله قبل فقط مقدار درست را انتخاب

و آن را به طبقه بعدی منتقل می‌کنیم.

با برای پیاده سازی از دو تا csa از دو تا RA ۸ بیتی استفاده می‌کنیم به این صورت

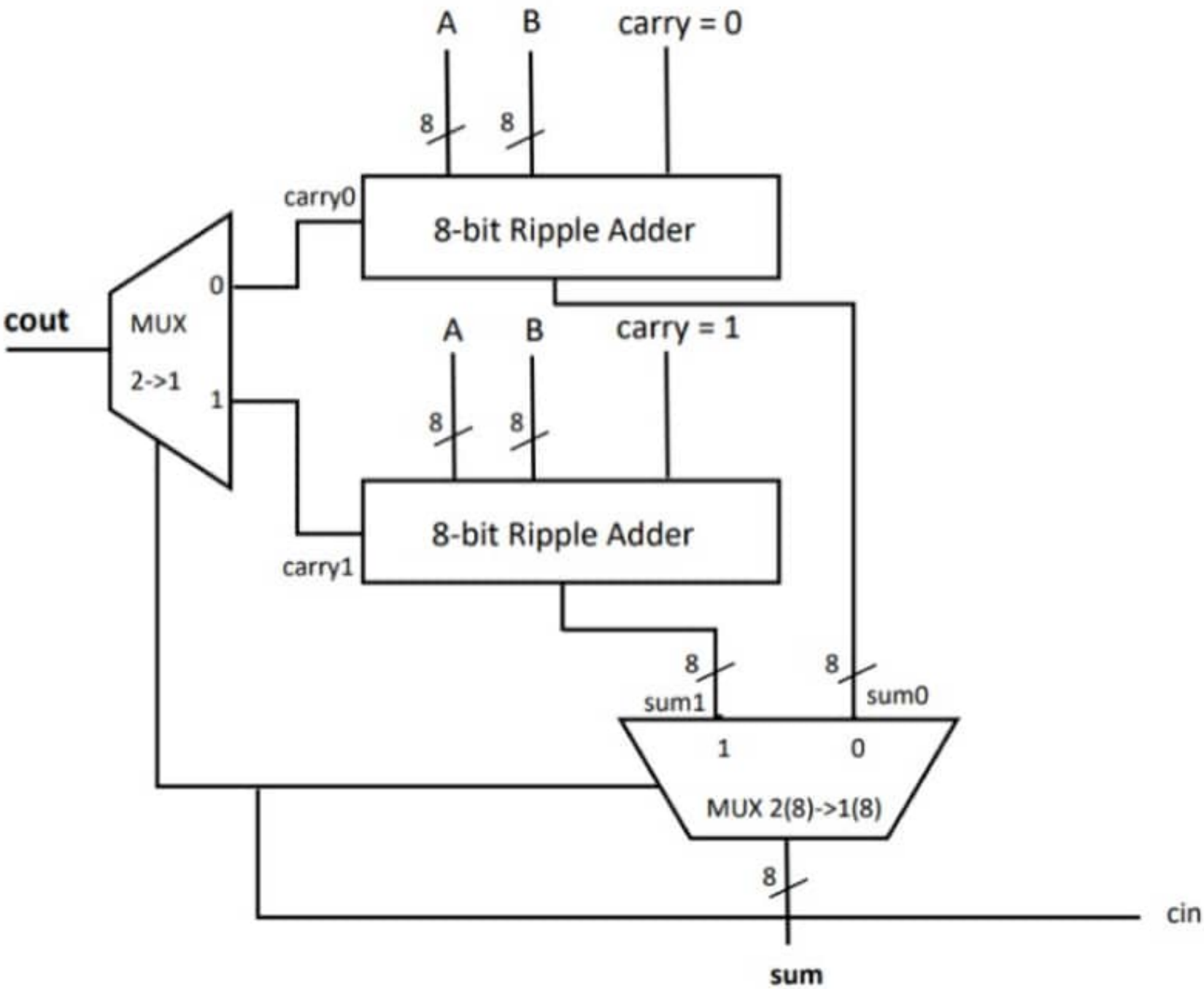
که دو عدد A و B را به این دو تا RA می‌دهیم ولی یک بار با carry = ۰ و بار دیگر

با carry = ۱ خروجی این دو تا RA را داخل دو تا Mux قرار می‌دهیم. حالا با توجه

به مقدار Cin خروجی یکی از این RA ها به عنوان Cin با توجه به همین به عنوان جواب

نهایی ما به دست می‌آید

شماتیک آن به صورت زیر است:



۱۴۰۰

مقایسه زمان (سرعت) و فضا (حجم اشغال شده)

نوع adder	سرعت	حجم
RA (Ripple adder)	کند	کم
Cla (carry look ahead)	متوسط	زیاد
	کندتر از RA	بیشتر از RA
CSa (carry select adder)	زیاد	متوسط
	تقریبا RA	بیشتر از Cla

طبق تعاریف و اطلاعات جدول بالا به صورت نسبی نوشته شده می‌توان

نتیجه گرفت تاخیر RA بیشتر، Cla متوسط و سرعت RA