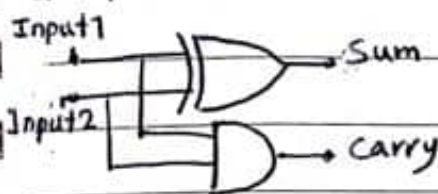


Subject: Date:

یاسین نوروزی - عنانه خاقدی
9931067 9931100

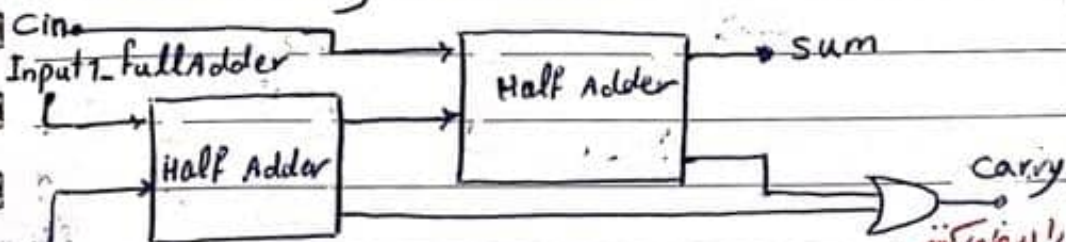
نما خدا
خدا را در آفرینش بنعمت معارف و بصیرت

اگر بخواهیم دوباره جمع کنند ما معیبت کم در ابتدا باید؛ ساختار Half Adder و Full Adder
Half Adder:



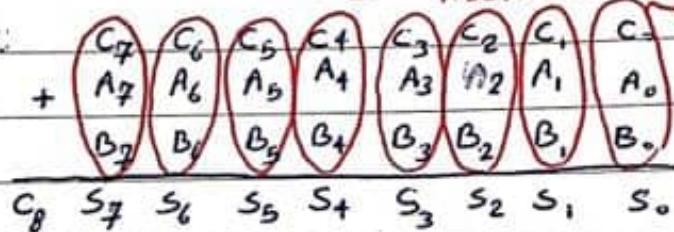
ما می‌توانیم مشاهده می‌شود در جمع ۱ بیت؛ این ما Sum و AND ما؛ Carry را حاصل می‌کند. اما این دهنده نیست!

Full Adder using Half Adder:

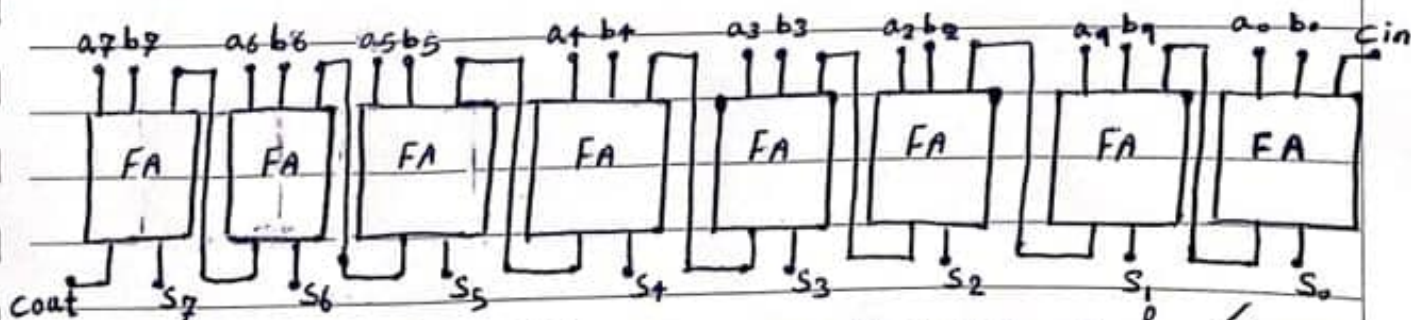


حرکت عملکردی Full Adder را اینجا می‌کنند

Ripple Carry Adder (8 bit)



در جمع دو عدد 8 bit در ابتدا نیز مشاهده می‌شود؛ با شروع جمع؛ در هر جمع یک Carry از جمع ظاهر می‌شود
انجام شده به جمع بعدی افزوده می‌شود



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

$$2t \times 8 = 16t$$

این روند شامل Carry و Sum می‌باشد؛ بنابراین تأخیر هر مرحله:

$$\text{delay (Carry)} = 2 \times 8t = 16t$$

File Edit View Project Source Process Tools Window Layout Help

View: Implementation Simulation

Hierarchy

- RippleCarryAdder
 - xc6slx9-2cpg196
 - HalfAdder - Structural (HalfAdder.vhd)
 - OR_gate - gatelevel (OR_gate.vhd)
 - RippleCarryAdder8bit - Structural (RippleCarryAdder8bit.vhd)

No Processes Running

Processes: RippleCarryAdder8bit - Structural

- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
- Translate
- Map

Start Design Files Libraries

Design Summary (Implemented) OR_gate.vhd RippleCarryAdder8bit.vhd HalfAdder.vhd

```

70      Sum => s(3),
71      Carry => c(4));
72
73      FullAdder5 : FullAdder port map (input1_fullAdder => a(4),
74      input2_fullAdder => b(4),
75      Cin => c(4),
76      Sum => s(4),
77      Carry => c(5));
78
79      FullAdder6 : FullAdder port map (input1_fullAdder => a(5),
80      input2_fullAdder => b(5),
81      Cin => c(5),
82      Sum => s(5),
83      Carry => c(6));
84
85      FullAdder7 : FullAdder port map (input1_fullAdder => a(6),
86      input2_fullAdder => b(6),
87      Cin => c(6),
88      Sum => s(6),
89      Carry => c(7));

```

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%

ISE Project Navigator (P.20131013) - C:\Users\Asus\Downloads\Music\session5\RippleCarryAdder\RippleCarryAdder.xise - [Design Summary (Implemented)]

File Edit View Project Source Process Tools Window Layout Help



Design View: Implementation Simulation

Hierarchy

RippleCarryAdder
xc6sxc9-2cpg196
HalfAdder - Structural (HalfAdder.vhd)
OR_gate - gatelevel (OR_gate.vhd)
RippleCarryAdder8bit - Structural (RippleCarryAdder8bit.vhd)

No Processes Running

Processes: RippleCarryAdder8bit - Structural

Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Translate
Map

Start Design Files Libraries

Design

Options

Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Translate
Map

No Processes Running

Processes: RippleCarryAdder8bit - Structural

Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Translate
Map

Start Design Files Libraries

HalfAdder Project Status (04/21/2022 - 11:06:26)			
Project File:	RippleCarryAdder.xise	Parser Errors:	No Errors
Module Name:	RippleCarryAdder8bit	Implementation State:	Placed and Routed
Target Device:	xc6sxc9-2cpg196	Errors:	No Errors
Product Version:	ISE 14.7	Warnings:	3 Warnings (3 new)
Design Goal:	Balanced	Routing Results:	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	Timing Constraints:	
Environment:	System Settings	Final Timing Score:	0 (Timing Report)

Device Utilization Summary				
Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	0	11,440	0%	
Number of Slice LUTs	8	5,720	1%	
Number used as logic	8	5,720	1%	
Number using O6 output only	0			
Number using O5 output only	0			

Console

++

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 :

Ripple carry adder سرعت بسیار پایین بود و تا Carry ورودی رسید باید منتظر بمانیم

به یک روش دیگر Carry look ahead هست در این روش

بر حسب شرط Carry بت میزنیم و منتظر توان Carry حسب رابطه

آنها در واقع FA ها را این adder بصورت موازی میزنیم. از این کار
درانفعیم زیرا توقف نمیکنیم:

$$\text{Carry generate} \rightarrow g_i = A_i B_i$$

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

بنا بر این برای Sum و Carry داریم:

$$\text{Sum}_i = A_i \text{ xor } B_i \text{ xor } C_i = p_i \text{ xor } C_i$$

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

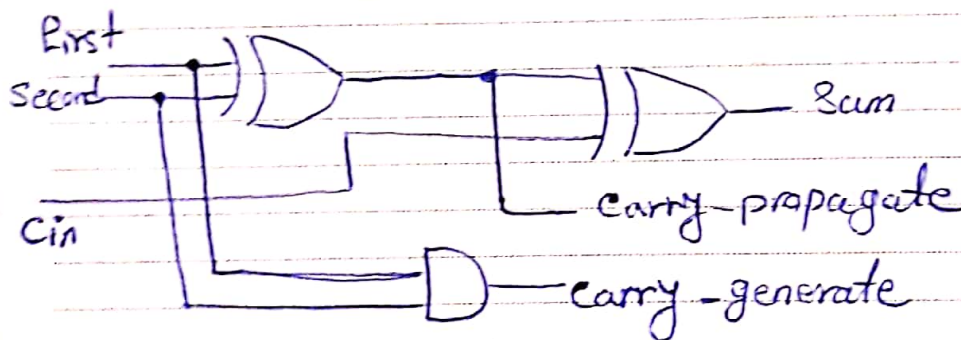
چون رابطه بالا می توان در حقیقت C_i را در برابر حسب رابطه

افزودیم به این است که در عمل این عملیات عدد بسیار بالا باشد است

دفعه دقت اول شده توسط مدارهای بالا هر دو برابر اند. Carry ها نیز
به بیت های آخر از تعداد بیت های Carry منقسم به دارا مدار خنجر یا در بیت های And
 OR عتد که Carry ها به مدارهای دو بیت می توانست.

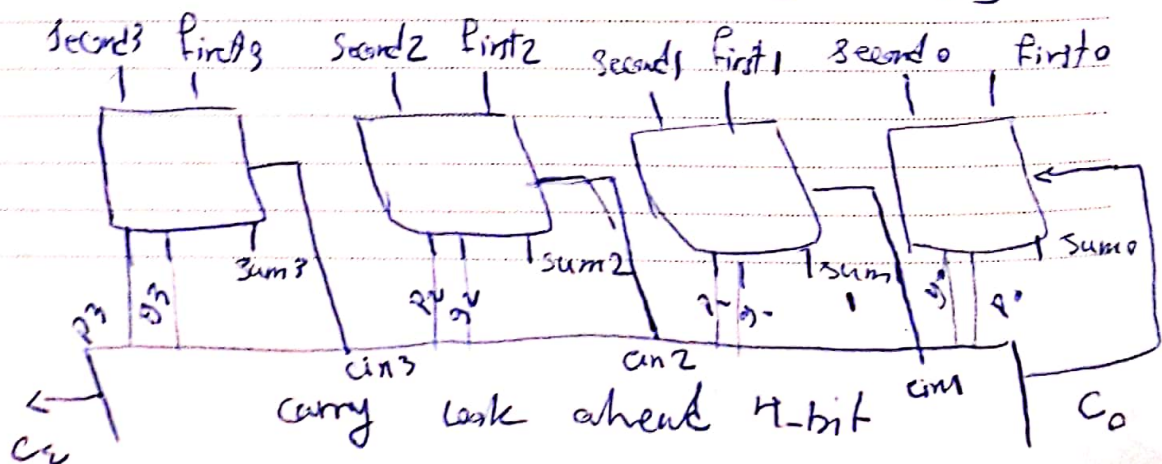
برای ساده سازی CLA استر از برای نه قضای و مدارهای Carry ها را حذف می کنند

یک راه محله در استکان می بینم برابر اینها است. Full adder می تواند Carry را
نیز تولید کند و توقف می بینم. Carry ها به صورت زیر است:



حالا برای طراحی CLA از Full adder استفاده می کنیم. Sum و Carry تولید

شود از خروجی Carry یک Carry به سیم دهیم. Carry ها به یک Carry در دست
کند به صورت زیر است:



صوت آخرها، صوت / راء:

تمريض

[illegible]

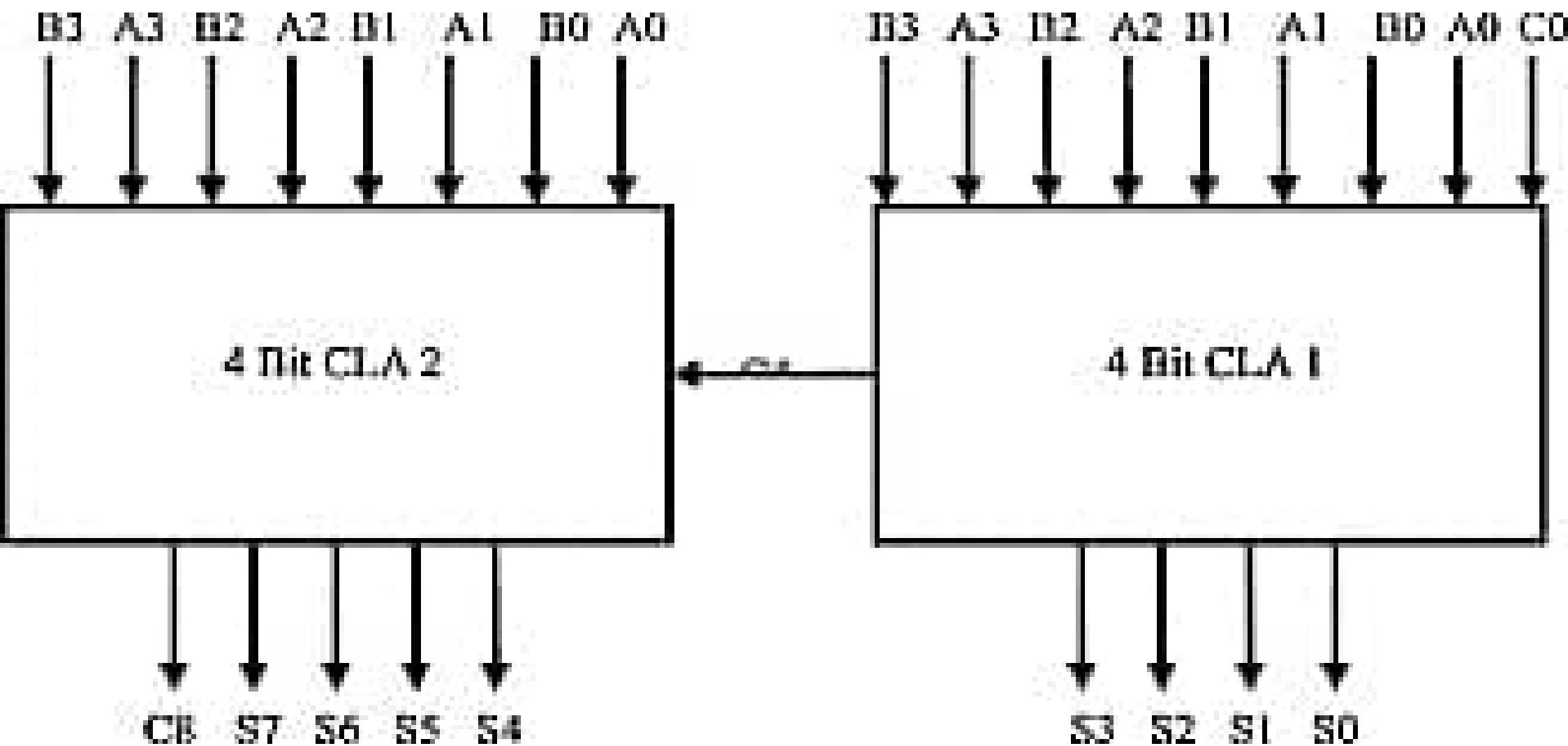
$$\text{Cost} \rightarrow 3n + \frac{n(n+3)}{2}$$

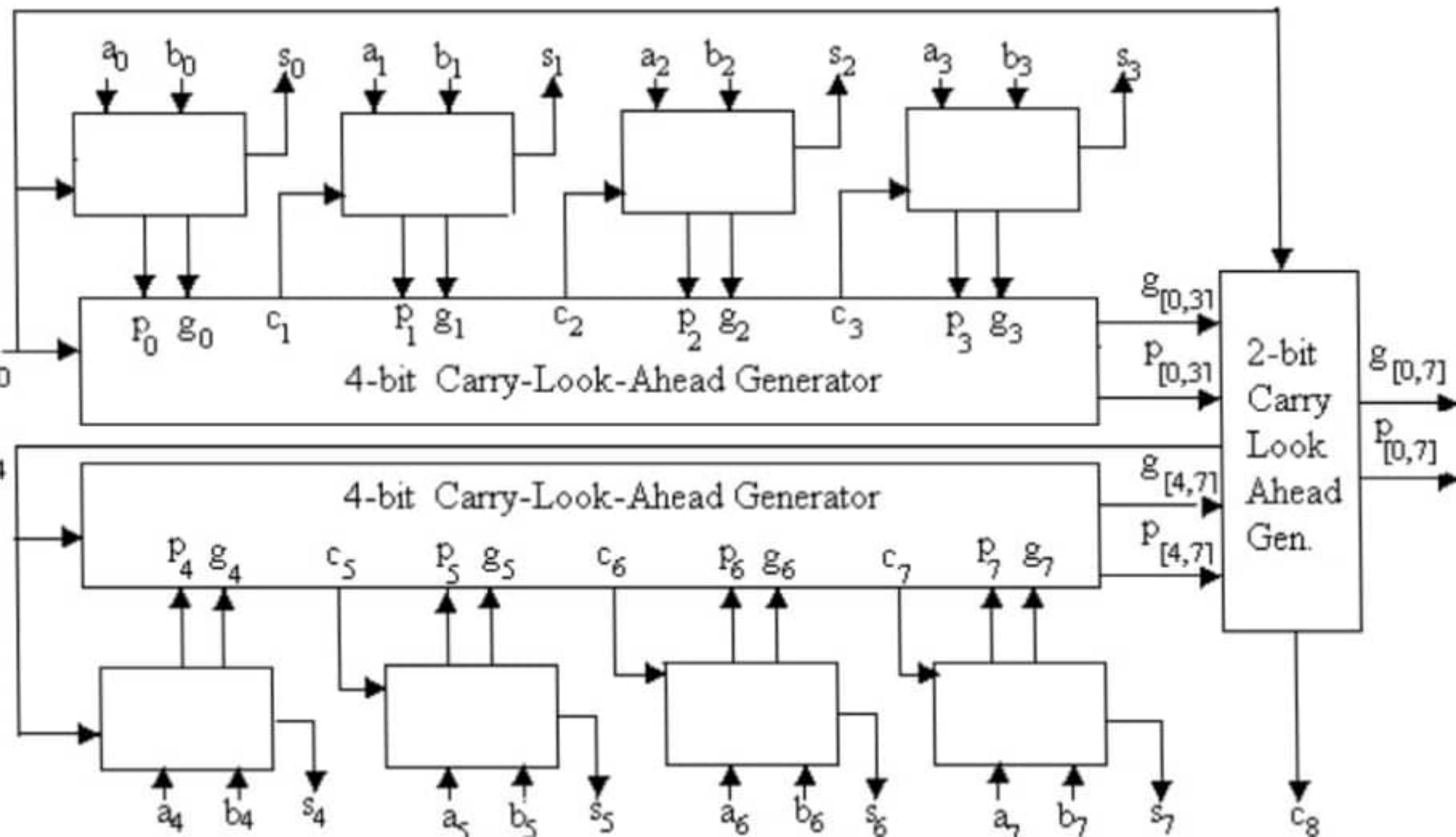
مقامات و ادارات محترمہ،

$$C_2 = g_1 + P_1 g_0 + P_1 P_0 C_0$$

$$C_3 = g_r + P_r g_i + P_r P_i g_o + P_r P_i P_o C_o$$

$$Q_{\Sigma} = g_c + P_c g_r + P_c g_1 P_r + P_r P_2 P_1 g_0 + P_c P_r P_1 P_0 C,$$



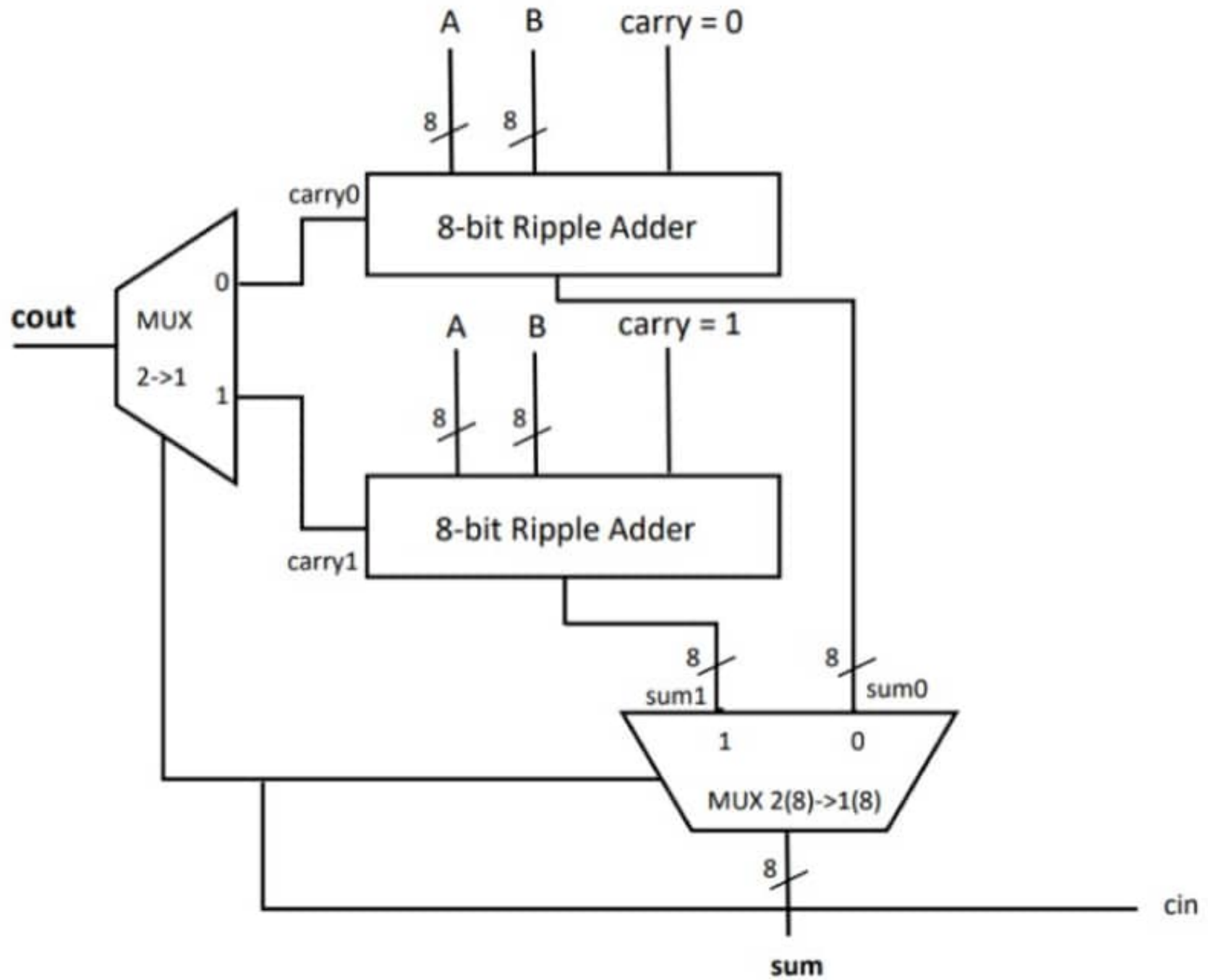


Carry select adder

در مکان قبلی دیدیم که چگونه C_{in} باید منتقل شود. مرحله
این عمل را در CSA استفاده می کنیم و نحوه آن به این صورت است که هر چه
مغایزی باشد از آنجا می آید و از آنجا که در $Carry$ و $Sums$ را انجام می دهیم و بعد
استاده از Max با توجه به $Carry$ مرحله قبل نتایج را درست را انتخاب
و آن را به مرحله بعد منتقل می کنیم.

باید به یاد داشته باشیم که C_{in} از RA است. استفاده می کنیم به این صورت
که در A و B را به این در RA می دهیم و می بینیم که $Carry = 0$ و بار دیگر
! $Carry = 1$ خروجی این در RA را داخل در Max قرار می دهیم. حالا با توجه
به C_{in} خروجی را از این RA ها به C_{in} حساب می کنیم. سوال بعدی:

برای جابجایی در C_{in}
و حساب آن به صورت زیر است:



مقایسه زمان (سرعت) و فضای (حجم) انواع آداپتورها

نوع آداپتور	سرعت	حجم
RA (Ripple adder)	کم	کم
CLA (Carry look ahead)	متوسط	متوسط
CSA (Carry select adder)	متوسط	کم

این مقایسه را به صورت جدولی در بالا به صورت زیر نوشته شد و به صورت جدول درج شد

توجه: در این مقایسه، RA به CLa و CLa به RA نسبت داده شده است