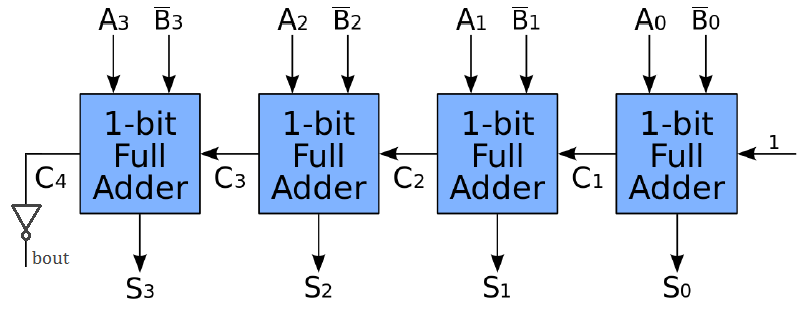
**گزارش دستورکار ششم آزمایشگاه معماری کامپیوتر**

نگار موقتیان، 9831062

**ماژول complement\_subtractor\_4bit**

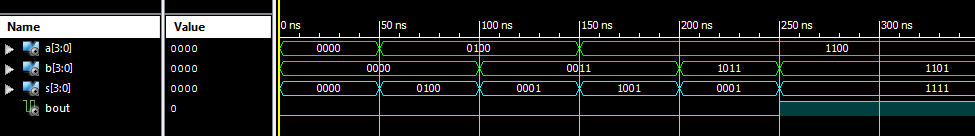
در این قسمت از آزمایش می­خواهیم یک تفریق کننده با استفاده از جمع کنندۀ آبشاری 4 بیتی ساخته شده در آزمایش قبل طراحی کنیم. برای این کار ابتدا سعی می­کنیم رابطۀ تفریق میان دو عدد را به یک رابطۀ جمع تبدیل کنیم.

بنابراین اگر با استفاده از یک جمع کنندۀ 4 بیتی عدد A، مکمل عدد B و عدد 1 را با یکدیگر جمع کنیم می­توانیم حاصل A – B را بدست آوریم. مدار این تفریق کنندۀ 4 بیتی به شکل زیر خواهد بود.

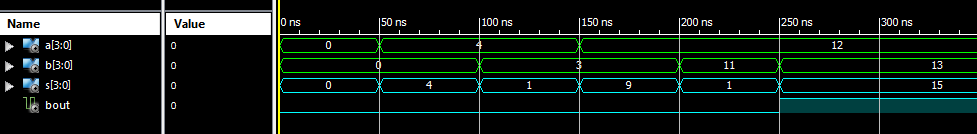


نکته­ای که باید در این مدار به آن توجه داشت این است که cout مدار جمع کنندۀ آبشاری حتما باید یک باشد. زیرا طبق روابط بالا یک 2n به پاسخ مساله اضافه کرده­ایم که باید خود را در پاسخ حاصل شده نشان دهد. یک بودن cout در مدار جمع کنندۀ آبشاری معادل این است که در طبقۀ آخر رقم قرضی (bout) نداشته باشیم. وجود رقم قرضی به این معناست که عدد A از B کوچک­تر بوده، لذا نتیجۀ حاصل شده در دنیای اعداد بی­علامت معتبر نیست.

پس از آن شبیه سازی این مدار توسط test bench نوشته شده و به ازای مقادیر مختلف A و B انجام شد، تا از درستی رفتار مدار ساخته شده اطمینان حاصل شود.



برای سهولت بیش­تر در بررسی سیگنال­های ورودی و خروجی تمام اعداد چند بیتی را با فرض بی­علامت بودن به مبنای 10 می­بریم. نتیجۀ تنظیمات گفته شده به صورت زیر است.



همانطور که مشاهده می­شود در 250 نانوثانیۀ اول شبیه­سازی خروجی تولید شده به ازای مقادیر A و B معتبر و صحیح است. اما پس از آن عدد A از B کوچک­تر بوده و سیگنال bout برابر با یک می­شود. لذا طبق توضیحات داده شده در بالا نتیجۀ حاصل شده معتبر نیست.

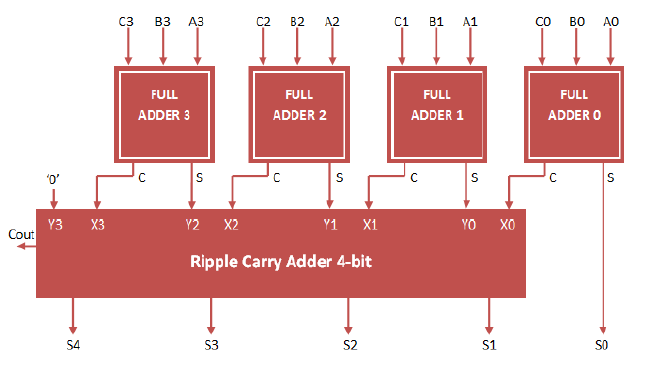
**ماژول carry\_save\_adder\_4bit**

حال در این قسمت از آزمایش می­خواهیم یک جمع کنندۀ ذخیره­گر نقلی با استفاده از جمع کنندۀ آبشاری 4 بیتی ساخته شده در آزمایش قبل طراحی کنیم.

ابتدا باید توجه داشت که می­توان به دو صورت به یک تمام جمع کننده نگاه کرد. یک دید مانند دیدی است که تا به اینجا داشته­ایم. یک تمام جمع کننده دو عدد یک رقمی را با یک رقم نقلی جمع می­زند. اما می­توان به آن به صورت یک جمع کننده برای 3 عدد یک رقمی نیز نگاه کرد، رویکردی که در این قسمت از آزمایش از آن استفاده می­کنیم.

نکتۀ بعدی این است که تا به این­جا همواره برای جمع چند عدد رقم نقلی را به گونه­ای تولید می­کردیم و به ورودی تمام جمع کنندۀ طبقۀ بعد می­دادیم. در حالی که در این نوع جمع کننده ما این رقم­های نقلی را ذخیره کرده، در ادامه آن را مانند یک عدد جدید در نظر گرفته و در نهایت آن را با حاصل جمع رقم­ها به صورت مستقل (در این­جا با استفاده از یک جمع کنندۀ آبشاری) جمع می­زنیم.

مدار این جمع کننده مانند زیر می­باشد.



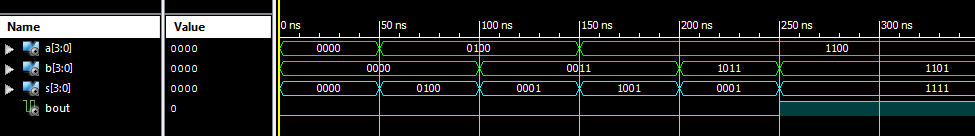
خروجی نهایی از کنار هم گذاشتن نتیجۀ Cout و S ها بدست می­آید. زیرا مجموع 3 عدد 4 بیتی نهایتاً برابر است با:

(1111)2 + (1111)2 + (1111)2 = (101101)2

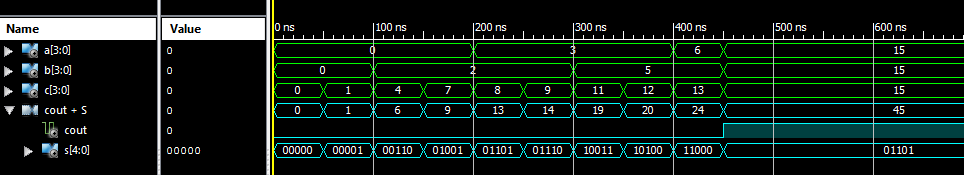
بنابراین برای نشان دادن حاصل جمع این 3 عدد حداقل نیاز به 6 رقم داریم.

حال با توجه به این توضیحات این جمع کننده را مطابق مدار داده شده و با استفاده از تمام جمع کننده و جمع کنندۀ 4 بیتی آبشاری ساخته شده در آزمایش قبل، به صورت ساختاری پیاده­سازی می­کنیم.

در ادامه شبیه سازی این مدار توسط test bench نوشته شده و به ازای مقادیر مختلف A، B و C انجام می­شود تا از درستی رفتار مدار ساخته شده اطمینان حاصل کنیم.



برای سهولت بیش­تر در بررسی سیگنال­های ورودی و خروجی از کنار هم گذاشتن سیگنال­های خروجی S و cout یک Virtual Bus ساخته و تمام اعداد را با فرض بی­علامت بودن به مبنای 10 می­بریم. نتیجۀ تنظیمات گفته شده به صورت زیر است.



همانطور که در این شکل دیده می­شود، به ازای مقادیر مختلف مجموع این سه عدد به درستی محاسبه شده است.

**ماژول­های carry\_generator\_logic و carry\_look\_ahead\_adder\_4bit**

در این قسمت از آزمایش می­خواهیم یک جمع کنندۀ پیش بینی کنندۀ رقم نقلی با استفاده از تمام جمع کننده­های ساخته شده در آزمایش قبل طراحی کنیم. انگیزۀ اصلی برای طراحی این نوع جمع کننده بهبود تاخیر جمع کنندۀ آبشاری است. همانطور که می­دانیم در جمع کنندۀ آبشاری هر طبقه باید منتظر طبقۀ قبلی بماند و علت اصلی تاخیر، رقم نقلی می­باشد. حال اگر بتوانیم به طریقی این رقم­های نقلی را با استفاده از A، B و cin با سرعت بیش­تری محاسبه کنیم (زیرا این سه ورودی­اند و در لحظۀ شروع و بدون هیچ تاخیری آن­ها را داریم) می­توانیم جمع کننده­ای بهینه بسازیم.

برای این کار ابتدا در ماژول carry\_genarator\_logic این رقم­های نقلی را تولید کرده­ایم، به گونه­ای که اگر در نظر بگیریم:

Pi = Ai + Bi, Gi = Ai . Bi

می­توان رقم نقلی مربوط به هر طبقه را از روابط زیر محاسبه کرد:

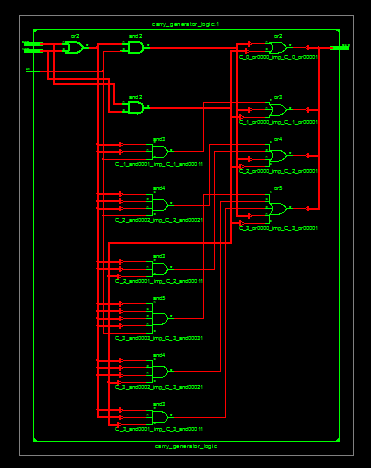
C0 = G0 + (cin . P0)

C1 = G1 + (G0 . P1) + (cin . P0 . P1)

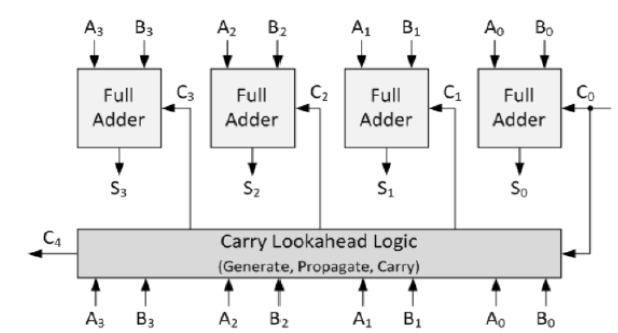
C2 = G2 + (G1 . P2) + (G0 . P1 . P2) + (cin . P0 . P1 . P2)

C3 = G3 + (G2 . P3) + (G1 . P2 . P3) + (G0 . P1 . P2 . P3) + (cin . P0 . P1 . P2 . P3)

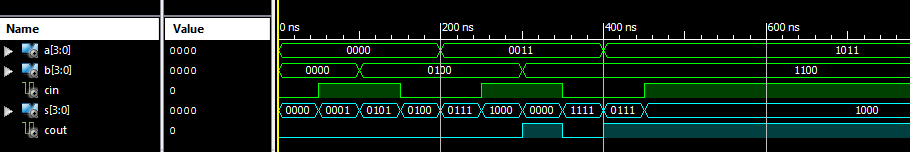
طرح شماتیک این مدار که توسط نرم افزار ISE تولید شده در ادامه آمده­است.



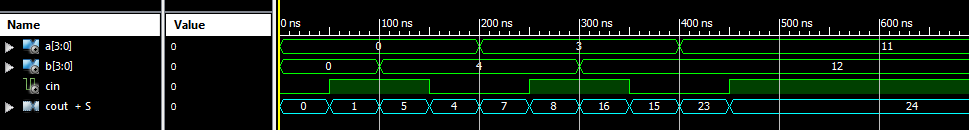
(همانطور که در این طرح شماتیک دیده می­شود گیت­هایی با تعداد ورودی بالا داریم. این قضیه ساخت جمع کنندۀ پیش بینی کنندۀ رقم نقلی با ورودی­های بزرگ را عملاً ناممکن می­سازد)

سپس در ماژول carry\_look\_ahead\_adder\_4bit این رقم­های نقلی را مطابق شکل زیر به تمام جمع­کننده­ها می­دهیم تا حاصل نهایی را بدست آوریم.

پس از آن شبیه سازی این مدار توسط test bench نوشته شده و به ازای مقادیر مختلف A، B و cin انجام شد، تا از درستی رفتار مدار ساخته شده اطمینان حاصل شود.



سپس مانند بخش­های قبل برای سهولت بیش­تر در بررسی سیگنال­های ورودی و خروجی از کنار هم گذاشتن سیگنال­های خروجی S و cout یک Virtual Bus ساخته و تمام اعداد را با فرض بی­علامت بودن به مبنای 10 می­بریم. نتیجۀ تنظیمات گفته شده به صورت زیر است.



همانطور که دیده می­شود جمع­ها به درستی انجام شده­اند.