**حل تمرین دو** **مهلت تحویل ساعت ۲۴ روز جمعه ۲۶ اسفند**

**به موارد زیر توجه کنید:**

۱- حتما نام و شماره دانشجویی خود را روی پاسخ‌نامه بنویسید.

۲- کل پاسخ تمرینات را در قالب یک فایل pdf با شماره دانشجویی خود نام‌گذاری کرده در سامانه CW بارگذاری کنید.

۳- این تمرین ۶۰ نمره دارد که معادل ۰٫۶ نمره از نمره کلی درس است.

۴- در صورت مشاهده هر گونه مشابهت نامتعارف هر دو (یا چند) نفر کل نمره این تمرین را از دست خواهند داد.

---

۱- (۱۰ نمره) برای جمع دو عدد ۱۶ بیتی از یک جمع‌کنندهٔ Carry Select استفاده می‌کنیم که در مرحله اول و دوم سه بیت و در مرحله سوم و چهارم به ترتیب ۴ و ۶ بیت را با هم جمع می‌کند. هزینه و تاخیر هر گیت را به ترتیب $D_G$ و $Cost_G$ و هزینه و تاخیر هر MUX را به ترتیب $D_{MUX}$ و $Cost_{MUX}$ درنظر بگیرید و فرض کنید هر تمام‌افزا (full adder) را با استفاده از روابط زیر ساخته‌ایم:

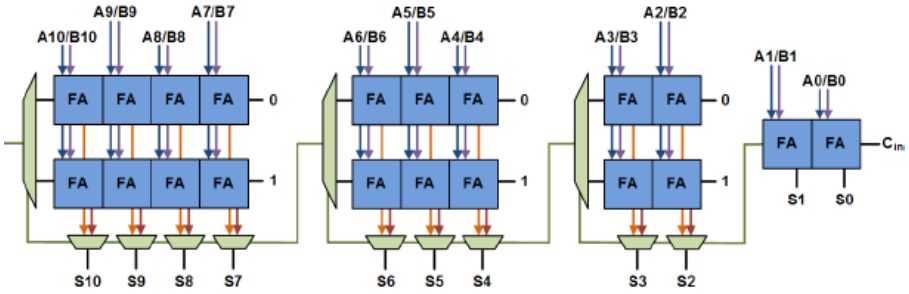
$S = XOR(a, b, c_{in})$

$C_{out} = a.b + a.c_{in} + b.c_{in}$

هزینه و تاخیر تولید حاصل‌جمع نهایی را محاسبه کنید.

پاسخ:

شکل جمع‌کننده مشابه با شکل زیر خواهد بود، با این تفاوت که تعداد بیت‌ها در هر مرحله متفاوت است.




در مرحله اول سه تمام‌افزا داریم و نیازی به mux نداریم. در مراحل بعد به ترتیب ۶، ۸ و ۱۲ تمام‌افزا و ۴، ۵ و ۷ مولتی‌پلکسر داریم. برای ساخت هر تمام‌افزا ۵ گیت نیاز داریم (یک XOR، سه گیت AND و یک گیت OR). بنابراین هزینه این جمع‌کننده عبارت است از:

$$Cost_{Adder} = (4 + 5 + 7)Cost_{Mux} + 5(3 + 6 + 8 + 12)Cost_G = 16Cost_{Mux} + 145Cost_G$$

برای محاسبه تاخیر اول به این نکته توجه می‌کنیم که در هر تمام‌افزا آماده شدن بیت نقلی (C) به اندازه تاخیر دو گیت و آماده شدن بیت جمع (S) به اندازه تاخیر یک گیت XOR زمان می‌برد، بنابراین همیشه بیت‌های نقلی دیرتر آماده می‌شوند. اگر بیت‌های نقلی هر مرحله را به ترتیب C3، C6، C10 و C16 بنامیم، تاخیر آماده شدن هر کدام را می‌توانیم به این صورت محاسبه کنیم:

$$Delay(C3) = 6D_G$$

$$Delay(C6) = 6D_G + D_{Mux}$$

$$Delay(C10) = \max(8D_G, Delay(C6)) + D_{Mux} = \max(8D_G, 6D_G + D_{Mux}) + D_{Mux} = 6D_G + D_{Mux} + \max(2D_G, D_{mux})$$

$$Delay(C16) = \max(12D_G, Delay(C10)) + D_{Mux} = \max(12D_G, 6D_G + D_{Mux} + \max(2D_G, D_{Mux})) + D_{Mux}$$
$$= 6D_G + D_{Mux} + \max(6D_G, D_{mux} + \max(2D_G, D_{Mux}))$$

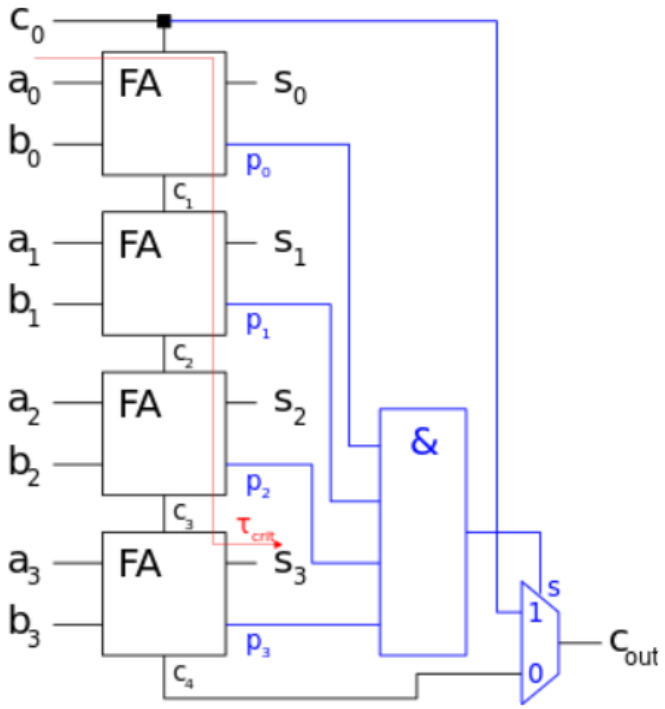
اگر فرض کنیم $DMux \approx 3DG$، تاخیر کلی $6DG + 3Dmux$ خواهد بود.

۲- (۲۰ نمره) شکل زیر یک جمع‌کنندهٔ چهار بیتی از نوع Carry Skip Adder را نشان می‌دهد.

الف- توضیح دهید این جمع‌کننده چطور کار می‌کند و چرا نتیجهٔ به دست آمده درست است.

ب- زمان لازم برای تولید بیت نقلی خروجی ($D_{Cout}$) بسته به مقدار ورودی‌های A و B متفاوت است. کمترین و بیشترین مقدار $D_{Cout}$ را بر حسب تاخیر یک تمام‌افزا ($D_{FA}$)، یک گیت AND چهار ورودی ($D_{A4}$) و یک MUX دو به یک ($D_{MUX}$) حساب کنید.

ج- می‌خواهیم با کنار هم گذاشتن چهار جمع‌کننده از همین نوع دو عدد ۱۶ بیتی را با هم جمع کنیم. کمترین و بیشترین مقدار زمان لازم برای آماده شدن حاصل‌جمع را محاسبه کنید. فرض کنید ورودی $C_0$ هر جمع‌کنندهٔ چهاربیتی به خروجی $C_{out}$ جمع‌کنندهٔ قبلی متصل است.




پاسخ:

الف- در این روش، اگر s=1 خروجی Cout برابر با ورودی Cin است. از طرفی s وقتی یک می‌شود که همهٔ piها یک باشند. می‌دانیم که هر یک از piها حاصل XOR دو بیت ai و bi هستند و تنها در صورتی pi=1 می‌شود که دو بیت ai و bi نامساوی باشند، یعنی یکی ۰ و دیگری ۱ باشد. در چنین شرایطی جمع این دو بیت carry نخواهد داشت. بنابراین، s در صورتی یک می‌شود که جمع دو به دوی هر کدام از بیت‌های ورودی carry نداشته باشد، پس Cout برابر با Cin خواهد بود. در غیر این‌صورت، s=0 شده و Cout از روش معمول به دست می‌آید.

یک راه دیگر برای توجیه کارکرد این مدار، استفاده از رابطه‌ای است که Cout را بر حسب piها بیان می‌کند. این رابطه را در جمع‌کننده‌های CLA دیده‌ایم:

$$P = p_0.p_1.p_2.p_3$$
$$G = g_3 + p_3g_2 + p_3p_2g_1 + p_3p_2p_1g_0$$
$$C_{out} = G + PC_0$$

طبق این رابطه، اگر P=1، Cout=C0 خواهد بود و به همین دلیل مدار بالا درست کار می‌کند.

ب- استفاده از این روش فقط Cout را آن هم در شرایط خاص زودتر حاضر می‌کند. اگر نه، آماده شدنِ حاصل‌جمع در هر حال به اندازه $4D_{FA}$ زمان می‌برد، $3D_{FA}$ برای حاضر شدن $C_3$ و $1D_{FA}$ برای حاضر شدن $S_3$.

در مورد $C_{out}$کمترین تاخیر زمانی رخ می‌دهد که s=1 و بیشترین تاخیر زمانی رخ می‌دهد که s=0، بنابراین:

$$Delay_{min}(C_{out}) = Delay(p_i) + Delay(AND) + Delay(MUX) = D_{FA} + D_{A4} + D_{mux}$$
$$Delay_{max}(C_{out}) = 4D_{FA} + D_{MUX}$$

ج- اگر چهارتا جمع‌کننده از این نوع را به دنبال هم قرار بدهیم، وقتی کمترین تاخیر را داریم که در هر کدام از سه واحد جمع‌کنندهٔ چهاربیتی اول (مربوط به بیت‌های ۰ تا ۱۱) $C_{out}$ در کمترین زمان ممکن حاضر شود:

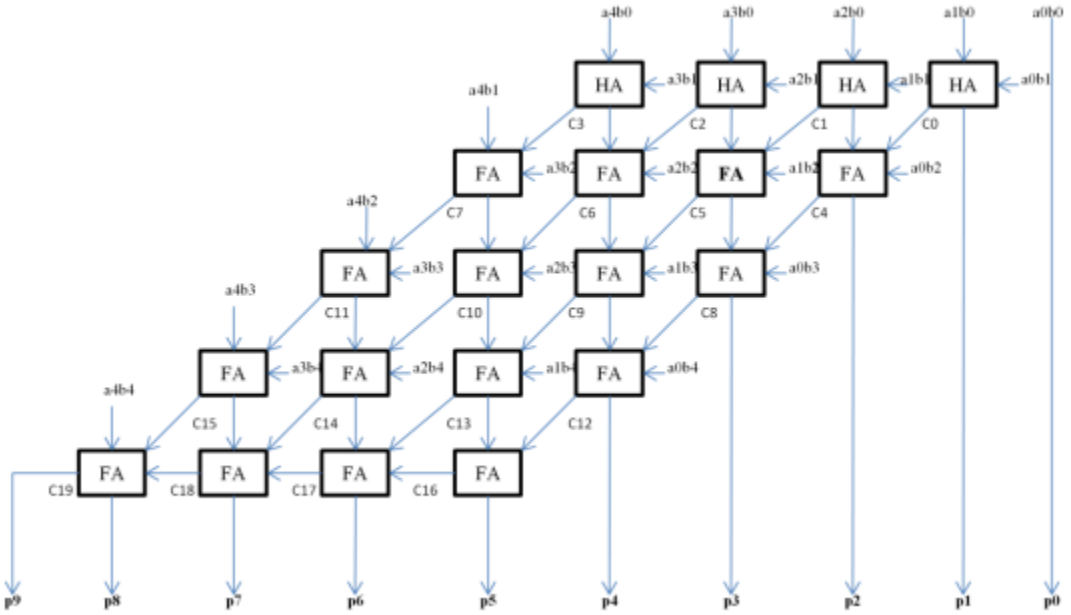
$$Delay_{min}(C_{out1}) = D_{FA} + D_{A4} + D_{mux}$$
$$Delay_{min}(C_{out2}) = Delay_{min}(C_{out1}) + D_{mux}$$
$$Delay_{min}(C_{out3}) = Delay_{min}(C_{out2}) + D_{mux} = D_{FA} + D_{A4} + 3D_{mux}$$
$$Delay_{min}(S_{15}) = Delay_{min}(C_{out3}) + 4D_{FA}$$

بیشترین تاخیر هم وقتی رخ می‌دهد که در هر کدام از سه واحد جمع‌کنندهٔ اول Cout در بیشترین زمان ممکن حاضر شود:

$$Delay_{max}(S_{15}) = 16D_{FA}$$

۳- (۱۰ نمره) در شکل زیر طرحی از یک ضرب‌کننده آرایه‌ای ۵ بیت در ۵ بیت می‌بینید. اگر برای ساخت بیت‌های نقلی و جمع در هر تمام‌افزا (full adder) از مدارهای AND-OR استفاده کنیم و با فرض این که تاخیر هر گیت NOT برابر ۱۰ نانوثانیه و تاخیر گیت‌های AND و OR هر کدام ۲۰ نانوثانیه باشد، زمان لازم برای تولید حاصل‌ضرب (T) چند نانوثانیه خواهد بود؟

اگر بخواهیم با همین روش دو عدد n بیتی را در هم ضرب کنیم و اگر تاخیر گیت‌های NOT و AND و OR را به ترتیب DC، DA و DO بنامیم، یک رابطه برای T به دست‌آورید.

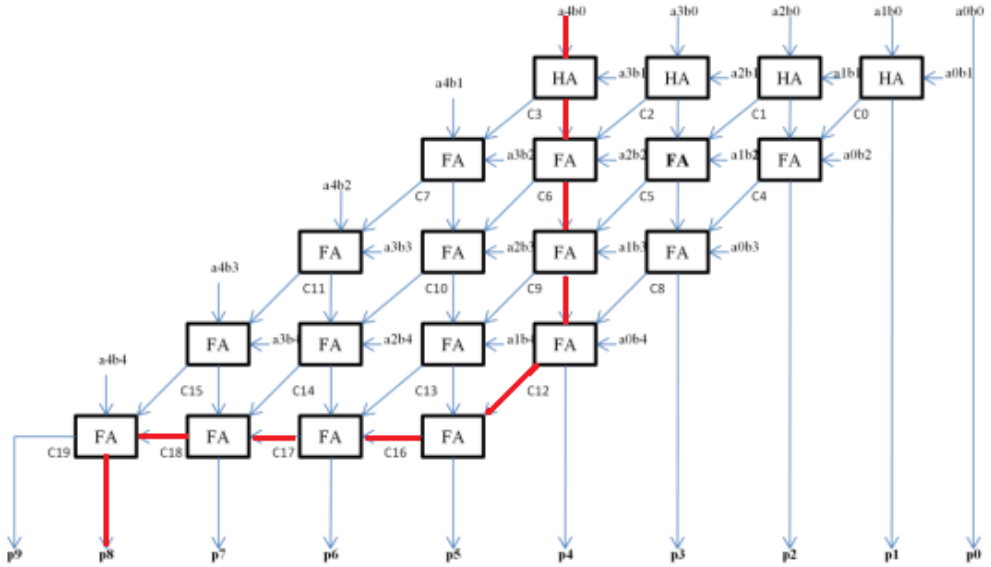



پاسخ:

تاخیر s و c در half adder و full adder این طور محاسبه می‌شود:

$$S_{HA} = DC + DA + DO$$
$$C_{HA} = DA$$
$$S_{FA} = DC + DA + DO$$
$$C_{FA} = DA + DO$$

مسیر بحرانی محاسبه حاصل‌ضرب در شکل زیر نشان داده شده است:




این مسیر شامل یک گیت AND، یک $S_{HA}$، سه $S_{FA}$ و چهار $C_{FA}$ است. بنابراین تاخیر کل عبارت است از:

$$T = DA + 4(DC + DA + DO) + 4(DA + DO) = 9DA + 8DO + 4DC = 9 \times 20 + 8 \times 20 + 4 \times 10 = 380ns$$

برای ضرب دو عدد n بیتی، n-1 ردیف و n-1 ستون خواهیم داشت، بنابراین تاخیر کل از رابطه زیر به دست می‌آید:

$$T = DA + (n-1)(DC + DA + DO) + (n-1)(DA + DO) = (2n-1)DA + (2n-2)DO + (n-1)DC$$

۴- (۲۰ نمره) می‌خواهیم عملیات ممیزثابت را با استفاده از اعداد صحیح مکمل ۲ (که در پردازنده‌های ساده نیز وجود دارند) انجام دهیم. فرض کنید هر عدد ۳۲ بیتی است که ۱۵ بیت آن به بخش کسری اختصاص داده شده است.

الف) نحوه نمایش این اعداد چگونه است؟

ب- بزرگترین و کوچکترین عدد مثبت و کوچکترین عدد منفی قابل نمایش را در این ساختار تعیین کنید.

ج- برای جمع و تفریق این اعداد چگونه باید عمل کرد؟

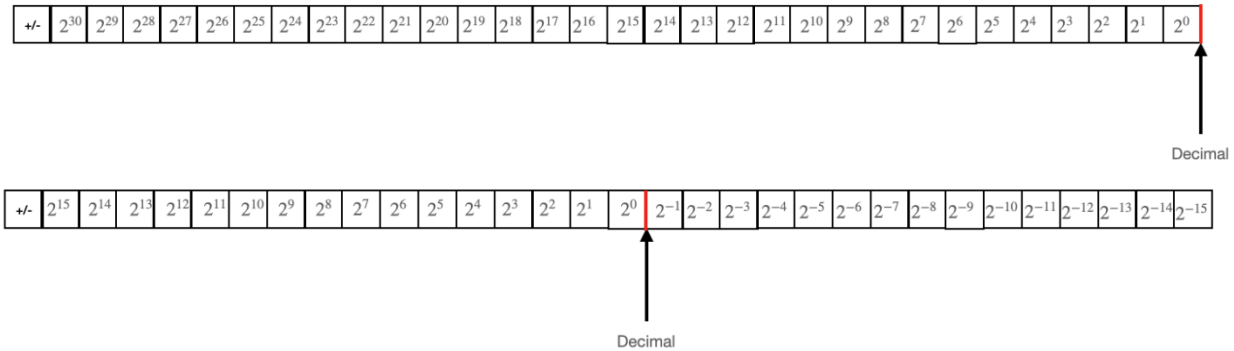
د- نحوه ضرب دو عدد ممیزثابت را با استفاده از عملیات اعداد صحیح توضیح دهید.

ه- یک ماکرو به زبان اسمبلی میپس بنویسید برای ضرب دو عدد با نمایش ممیزثابت. فرض کنید این دو عدد در دو ثبات قرار دارند.

و- اگر بخواهیم پس از ضرب سرریز (overflow) را تشخیص دهیم باید چگونه عمل کنیم؟

پاسخ:

الف- ۱۵ بیت کم‌ارزش بخش کسری و بقیه بخش صحیح عدد را نشان می‌دهند. برای درک بهتر دو شکل زیر را با هم مقایسه کنید.




ب- بزرگترین عدد مثبت وقتی به دست می‌آید که همه بیت‌ها یک باشد که معادل $2^{16} - 2^{-15}$ است. از یک دید دیگر این عدد معادل بزرگترین عدد صحیح مثبت تقسیم بر $2^{15}$ است:

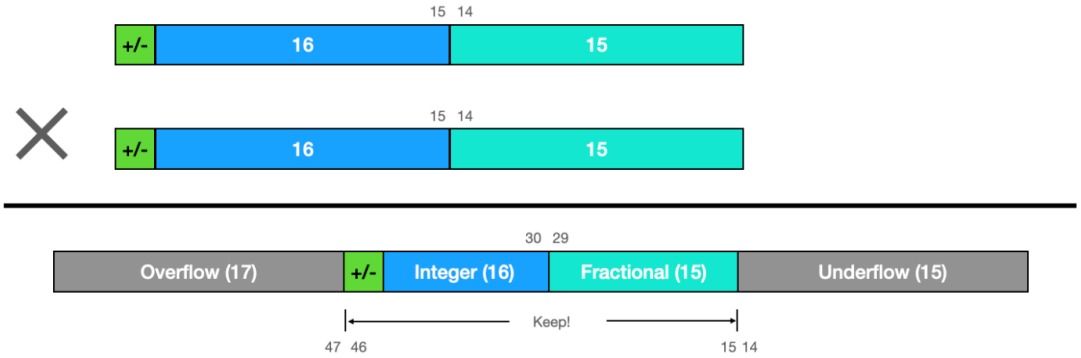
$$\frac{2^{32}-1}{2^{15}} = \sum_{x=-15}^{15} 2^x \simeq 65{,}535.999969$$

کوچکترین عدد مثبت هم برابر خواهد بود با $2^{-15}$

کوچکترین عدد (منفی‌ترین عدد) وقتی به دست می‌آید که همهٔ بیت‌ها صفر باشند، به جز بیت علامت که این عدد برابر است با بزرگترین مقدار منفی تقسیم بر $2^{15}$ که برابر است با $-\frac{2^{32}}{2^{15}} = -65536$

ج- جمع و تفریق این اعداد تفاوتی با جمع و تفرین اعداد صحیح ندارد.

د- حاصل‌ضرب دو عدد ۳۲ بیتی، ۶۴ بیت می‌شود. اگر این دو عدد هر کدام ۱۵ بیت کسری داشته باشند، ممیز بین بیت ۲۹ و ۳۹ قرار خواهد گرفت:




حال کاری که نیاز است انجام دهیم این است که عدد به‌دست آمده را ۱۵ بیت به سمت راست شیفت بدهیم تا ممیز به محل اصلی خود بازگردد. یعنی از ضرب عادی استفاده می‌کنیم و سپس عدد را ۱۵ بیت شیفت می‌دهیم و ۳۲ بیت ابتدایی آن را در نظر می‌گیریم.

ه- ماکروی مورد نظر به شکل زیر خواهد بود:

```
# 16 bits for the integer part
# and 15 bits for the fraction
# r1 & r2 are multiplied
# r0 is the result
# rt is a temporary register
.macro mul32 %r0,%r1,%r2,%rt
   mult %r1,%r2
   mflo %r0
   srl %r0,%r0,15
   mfhi %rt
   sll %rt,%rt,17
   or %r0,%r0,%rt
.end_macro
```

و- باید چک کنیم که بیت علامت عددی که نگه می‌داریم با باقی بیت‌هایی که دور می‌ریزیم (Overflow) یکی باشد.