به نام خدا

درس معماری کامپیوتر
نیم‌سال دوم ۰۴-۰۳
استاد: دکتر اسدی

دانشکده مهندسی کامپیوتر

پاسخنامه تمرین سری سوم

---

۱. (آ) برای پشتیبانی عملیات تفریق در CSA، باید بیت add/sub به ورودی C_{in} تمام‌جمع‌کننده[1] اول متصل شود و برای ورودی دوم همه تمام‌جمع‌کننده‌ها، یک گیت XOR قرار داده شود. این گیت، هر بیت ورودی دوم را با بیت add/sub عملیات XOR انجام می‌دهد تا در صورت لزوم مکمل دو آن را محاسبه کند یا در غیر این صورت، ورودی‌های مدار را بدون تغییر نگه دارد. این قضیه باعث می‌شود که به هر بلوک، تاخیر یک نانوثانیه اضافه شود.
فرض کنیم تابع f(t) نشان‌دهنده حداکثر تعداد بیت‌هایی باشد که می‌توانیم در مدت زمان t نانوثانیه عملیات جمع یا تفریق را روی آن‌ها انجام دهیم. برای به‌دست آوردن تابع $f(t)$ باید دو حالت زیر را در نظر بگیریم:

۱. حالتی که تنها یک بلوک داشته باشیم: در این صورت، حداکثر تعداد بیت‌هایی که می‌توانیم جمع کنیم برابر است با $\lfloor \frac{t-1}{2} \rfloor$. بنابراین

$$f(t) = \lfloor \frac{t-1}{2} \rfloor$$

۲. حالتی که از چندین بلوک استفاده کنیم: در این حالت، با توجه به اینکه تأخیر MUX برابر ۳ نانوثانیه است، بلوک آخر حداکثر $t-3$ نانوثانیه فرصت دارد تا نتیجه را محاسبه کند. در نتیجه، این بلوک می‌تواند حداکثر $\lfloor \frac{t-4}{2} \rfloor$ بیت را محاسبه کند. (اگر فرض کنیم این بلوک n بیتی باشد، زمان محاسبه جمع یا تفریق در آن $2n+1$ نانوثانیه طول می‌کشد که باید $2n+1 \leq t-3$ باشد؛ بنابراین، حداکثر $n = \lfloor \frac{t-4}{2} \rfloor$ به دست می‌آید.)

در حالت دوم، برای محاسبه اندازه بلوک‌های قبلی، یک رابطه بازگشتی به صورت

$$f(t) = f(t-3) + \lfloor \frac{t-4}{2} \rfloor$$

تعریف می‌کنیم. در هر مرحله، حداکثر تعداد بیت‌هایی که بلوک جاری می‌تواند محاسبه کند با $\lfloor \frac{t-4}{2} \rfloor$ مشخص می‌شود و برای محاسبه بلوک‌های قبلی، ۳ نانوثانیه (معادل تأخیر MUX) از زمان کسر می‌شود و تابع دوباره فراخوانی می‌شود.
به طور کلی، با بررسی هر دو حالت بالا، برای محاسبه تابع $f(t)$ به رابطه زیر می‌رسیم:

$$f(t) = \max\left(\lfloor \frac{t-1}{2} \rfloor, \lfloor \frac{t-4}{2} \rfloor + f(t-3) \right)$$

حالات پایه برای این تابع به صورت زیر تعریف می‌شود:

$$f(0) = f(1) = f(2) = 0$$

---

[1] Full-Adder

چون حداقل زمانی که نیاز داریم تا یک بیت را جمع کنیم برابر با ۳ نانوثانیه است.
اکنون، پس از به‌دست آوردن تابع $f(t)$، می‌توانیم مقادیر مختلف تأخیر را در این تابع جایگذاری کنیم تا حداکثر تعداد بیت‌هایی که می‌توان در آن مدت زمان محاسبه کرد را تعیین کنیم.

$$f(21) = \lfloor\frac{21-4}{2}\rfloor + \lfloor\frac{18-4}{2}\rfloor + \lfloor\frac{15-4}{2}\rfloor + \lfloor\frac{12-4}{2}\rfloor + \lfloor\frac{9-4}{2}\rfloor + \lfloor\frac{6-1}{2}\rfloor = 28$$

$$f(22) = \lfloor\frac{22-4}{2}\rfloor + \lfloor\frac{19-4}{2}\rfloor + \lfloor\frac{16-4}{2}\rfloor + \lfloor\frac{13-4}{2}\rfloor + \lfloor\frac{10-4}{2}\rfloor + \lfloor\frac{7-1}{2}\rfloor = 32$$

پس دیدیم که کمترین تاخیر ممکن برای محاسبه جمع یا تفریق ۳۲ بیت برابر با ۲۲ نانوثانیه است و اندازه بلوک‌های آن به ترتیب ۳، ۳، ۴، ۶، ۷ و ۹ است.

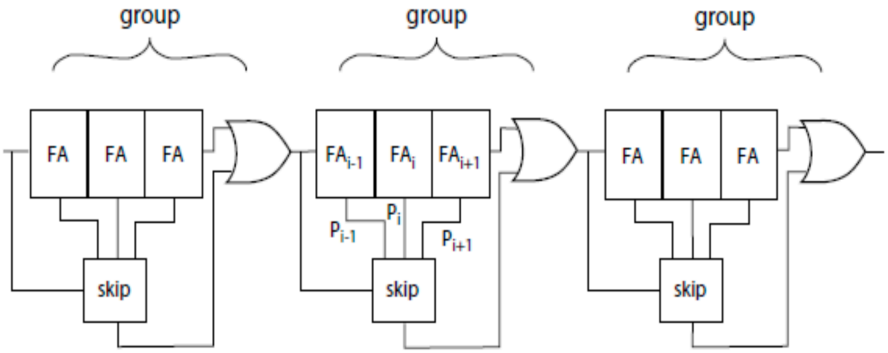
(ب) با توجه به بخش قبل، تاخیر CSA ما برابر با ۲۲ نانوثانیه شده است. برای آن که کمترین تاخیر را در عملیات ضرب داشته باشیم، باید ضرب‌کننده[2] کمترین میزان تغییر را داشته باشد تا کمترین عملیات جمع یا تفریق انجام شود پس برای حالت کمترین تاخیر، ضرب‌کننده ما باید برابر با ۰ باشد. تاخیر آن در این حالت برابر با ۶۴ نانوثانیه می‌شود و ما صرفا باید ۳۲ بار عملیات shift را انجام دهیم.
برای حالت بیشترین تاخیر، باید بیشترین تعداد جمع یا تفریق صورت گیرد که این حالت با ضرب‌کننده زیر انجام می‌شود:

01010101010101010101010101010101

در این حالت ما ۳۲ عملیات shift داریم که ۶۴ نانوثانیه طول می‌کشد و همچنین ۱۶ عملیات جمع و ۱۶ عملیات تفریق داریم که هرکدام ۲۲ نانوثانیه طول می‌کشد که در کل تاخیر جمع و تفریق ما برابر با ۷۰۴ نانوثانیه می‌شود و تاخیر حداکثری ضرب‌کننده برابر با ۷۶۸ نانوثانیه می‌شود.

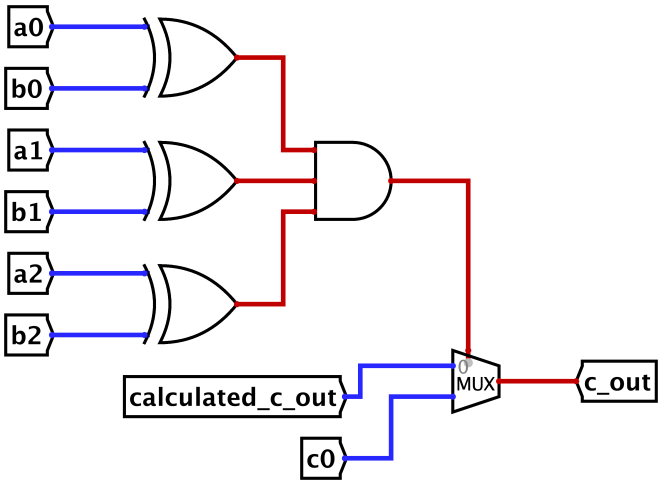
[2] Multiplier

۲. (آ) جمع‌کننده carry-skip از بلوک‌های متعددی تشکیل شده است که هر بلوک شامل تعداد تمام‌ـجمع‌کننده است که به صورت جمع‌کننده ripple-carry هستند. همچنین هر بلوک شامل یک واحد منطق skip است. در صورتی که شرط skip در بلوک جمع‌کننده برقرار شود، بدون محاسبه رقم نقلی در هر تمام‌ـجمع‌کننده، رقم نقلی ورودی به رقم نقلی خروجی منتقل می‌شود که باعث افزایش سرعت بلوک جمع‌کننده می‌شود.



(ب) جمع‌کننده carry-skip جمع‌کننده‌ای است که بر خلاف سایر جمع‌کننده‌های سریع دیگر، سرعت را تنها به ازای برخی از حالت‌های ورودی بهبود می‌دهد. بنابراین به ازای برخی از ورودی‌ها، بهبود زمانی نسبت به جمع‌کننده ripple-carry رخ نمی‌دهد.

(ج) منطق skip رقم نقلی به شکل زیر است:



(د) تعداد کل حالت‌های ورودی عبارتند از: 2^{2n}

تعداد کل حالت‌های ورودی که باعث برقرار شدن شرط رد شدن می‌شوند عبارتند از: $2\times1\times2\times1\times2\times\ldots=2^n$

بنابراین احتمال رد شدن بیت نقلی در این بلوک برابر می‌شود با: $\frac{2^n}{2^{2n}}=\frac{1}{2^n}$

۳. می‌توانیم محاسبه نماییم که برای هر عمل تفریق نیاز به ۱۶ns داریم، در نتیجه حال می‌توانیم جداول انجام ضرب در دو حالت مختلف را رسم نماییم و تاخیر هر کدام را محاسبه نماییم.

(آ) ۱. multiplicand = 101000

A	Q	Q-۱	M	Log	Delay
۰۰۰۰۰۰	۰۱۱۰۰۱	۰	۱۰۱۰۰۰	Initialization	۰
۰۱۱۰۰۰	۰۱۱۰۰۱	۰	۱۰۱۰۰۰	A = A – M	۱۶
۰۰۱۱۰۰	۰۰۱۱۰۰	۱	۱۰۱۰۰۰	Shift	۲۴
۱۱۰۱۰۰	۰۰۱۱۰۰	۱	۱۰۱۰۰۰	A = A + M	۳۶
۱۱۱۰۱۰	۰۰۰۱۱۰	۰	۱۰۱۰۰۰	Shift	۴۴
۱۱۱۱۰۱	۰۰۰۰۱۱	۰	۱۰۱۰۰۰	Shift	۵۲
۰۱۰۱۰۱	۰۰۰۰۱۱	۰	۱۰۱۰۰۰	A = A – M	۶۸
۰۰۱۰۱۰	۱۰۰۰۰۱	۱	۱۰۱۰۰۰	Shift	۷۶
۰۰۰۱۰۱	۰۱۰۰۰۰	۱	۱۰۱۰۰۰	Shift	۸۴
۱۰۱۱۰۱	۰۱۰۰۰۰	۱	۱۰۱۰۰۰	A = A + M	۹۶
۱۱۰۱۱۰	۱۰۱۰۰۰	۰	۱۰۱۰۰۰	Shift	۱۰۴

۲. multiplicand = 011001

A	Q	Q-۱	M	Log	Delay
۰۰۰۰۰۰	۱۰۱۰۰۰	۰	۰۱۱۰۰۱	Initialization	۰
۰۰۰۰۰۰	۰۱۰۱۰۰	۰	۰۱۱۰۰۱	Shift	۸
۰۰۰۰۰۰	۰۰۱۰۱۰	۰	۰۱۱۰۰۱	Shift	۱۶
۰۰۰۰۰۰	۰۰۰۱۰۱	۰	۰۱۱۰۰۱	Shift	۲۴
۱۰۰۱۱۱	۰۰۰۱۰۱	۰	۰۱۱۰۰۱	A = A – M	۴۰
۱۱۰۰۱۱	۱۰۰۰۱۰	۱	۰۱۱۰۰۱	Shift	۴۸
۰۰۱۱۰۰	۱۰۰۰۱۰	۱	۰۱۱۰۰۱	A = A + M	۶۰
۰۰۰۱۱۰	۰۱۰۰۰۱	۰	۰۱۱۰۰۱	Shift	۶۸
۱۰۱۱۰۱	۰۱۰۰۰۱	۰	۰۱۱۰۰۱	A = A – M	۸۴
۱۱۰۱۱۰	۱۰۱۰۰۰	۱	۰۱۱۰۰۱	Shift	۹۲

همانطور که می‌توانید مشاهده نمایید در هر دو حالت به جواب 110110101000 رسیدیم که همان 600- و نتیجه‌ی درست است.

(ب) این بخش هم در جدول بخش آ مشخص شده که می‌توانید مشاهده نمایید که در حالتی که multiplicand = 011001 باشد، سرعت انجام ضرب بیشتر است.

۴. تاخیر نهایی خروجی یک مالتیپلکسر با ورودی‌های A، B و S به فرم زیر است:

$$max(t_A, t_B, t_S) + 1$$

که منظور از t_X تاخیر آماده شدن ورودی X است.
حال با این دانش، جواب به فرم زیر خواهد بود که خروجی تابع f حداکثر تعداد بیت قابل جمع در x واحد زمان است.

$$f(x) = 2 \times f(x-1)$$

$$f(1) = 1$$

دلیل این موضوع نیز این است که ما بیشینه تعداد بیت را در صورتی می‌توانیم جمع بزنیم که تمامی ورودی‌های مالتیپلکسر در دیرترین زمان ممکن برای اینکه خروجی در زمان مطلوب حاضر شود حاضر شوند که با توجه به داده سوال، باید در ۱ واحد زمانی زودتر حاضر شده باشند. از آنجا که ورودی‌های A و B عملا تعدادی بیت یکسان را با ورودی نقلی متفاوت جمع می‌زنند، آن‌ها را به یک چشم نگاه می‌کنیم. حال با فرض اینکه بخواهیم در x واحد زمانی بیشترین تعداد بیت را جمع بزنیم، کافیست خروجی نقلی حاصل از جمع بیشترین بیت قابل جمع در $x-1$ واحد زمانی را به عنوان ورودی انتخاب‌گر و خروجی حاصل جمع همین تعداد بیت را به عنوان داده در حال انتخاب بدهیم.

۵. الف) سه عدد اول در مرحله‌ی اول جمع می‌شوند. در مرحله‌ی دوم عدد چهارم و نتیجه‌ی جمع مرحله‌ی قبل با هم جمع می‌شوند. بیت های carry در این مرحله همان بیت‌های C_{out} مرحله‌ی قبلی هستند. در نهایت نیز از یک جمع‌کننده‌ی عادی در مرحله‌ی آخر استفاده می‌شود.

ب) طبق فرضیات، تاخیر یک Full Adder برابر با d خواهد بود. ابتدا تاخیر یک CLA ۴ بیتی را محاسبه می‌کنیم.

تولید p_i و g_i از روی a_i و b_i = d

تولید c_iها از روی p_i و g_i = d

تولید s_iها پس از آماده شدن c_iها = d

پس تاخیر CLA مرحله‌ی آخر برابر با 3d خواهد بود. ورودی‌های لازم برای CLA هم طبق شکل پس از 2d حاضر خواهد بود. بنابراین تاخیر کلی برابر با 5d است.

۶. (آ) فرض کنید امید ریاضی مربوط به تعداد عملیات‌های جمع n بیتی را با $SUM(n)$، امید ریاضی تعداد عملیات‌های تفریق n بیتی را بار $SUB(n)$ و تعداد عملیات‌های انتقال را با $SHIFT(n)$ نشان می‌دهیم.

از آنجا که عملیات‌های مربوط به ضرب Booth در ضرب دو عدد n بیتی، n بار تکرار خواهند شد، تنها بیت‌هایی که باید برای محاسبه این امید ریاضی‌ها لحاظ شوند، صرفا به n بیت ضرب شونده و یک بیت ۰ ابتدایی که سمت راست آنها قرار می‌گیرد، محدود می‌شود. حالتی که جمع رخ می‌دهد، حالتی است که 01 رخ دهد. برای تفریق نیز 10 و همچنین در همه حالات ممکن دو بیت کنار هم، انتقال رخ می‌دهد. بنابراین داریم:

$\overline{a_{n-1}a_{n-2}a_{n-3}...a_0} \times \overline{b_{n-1}b_{n-2}b_{n-3}...b_0}$

$\overline{b_{n-1}b_{n-2}b_{n-3}...b_0}0$

$SUM(n) = (n-1) \times \frac{1}{2} \times \frac{1}{2} = \frac{n-1}{4}$

$SUB(n) = (n-1) \times \frac{1}{2} \times \frac{1}{2} + \frac{1}{2} = \frac{n+1}{4}$

$SHIFT(n) = n$

(ب) حال با استفاده از جواب بخش قبل، متوسط زمان تاخیر را بدست می‌آوریم.

$LATENCY(n) = 10 \times SUM(n) + 15 \times SUB(n) + 5 \times SHIFT(n) = 10 \times \frac{n-1}{4} + 15 \times \frac{n+1}{4} + 5 \times n = \frac{45n+5}{4} ns$

۷. (آ) باید بلاک ها را به صورت $i, i, i+1, ..., i+x$ جدا کنیم و معادله زیر را نیز داریم:

$$i + i + (i+1) + ... + (i+x) = 128 \rightarrow (x+2)i + \frac{x(x+1)}{2} = 128 \rightarrow i = \frac{128}{x+2} - \frac{x(x+1)}{2(x+2)}$$

همچنین باید دقت کنیم که بیشترین تاخیر مربوط به بزرگترین بلاک یعنی $i+x$ است درنتیجه تاخیر آن معادل است با $(i+x)D + D$ که برای یافتن کمترین مقدار ممکن مشتق گرفته و معادل ۰ میگذاریم پس داریم:

$$\frac{d}{dD}((i+x)D + D) = 0 \rightarrow i + x + 1 = 0 \rightarrow 128 \rightarrow i = \frac{128}{x+2} - \frac{x(x+1)}{2(x+2)} + x + 1 = 0$$

از آنجا که معادله بالا وقتی $x = 16$ قرار دهیم منفی و وقتی $x = 15$ قرار دهیم مثبت است جواب بین این ۲ عدد است و ما براکت مقدار را استفاده می کنیم یعنی سایز بلاک ها به صورت $2, 2, 3, ..., 14, 15, 7$ می شود. در نتیجه برای هزینه و تاخیر کل داریم:

$$D_{\text{total}} = 15D + D + D = 17D$$

(ب) تاخیر نباید از $65D$ بیشتر باشد و همچنین تا جای ممکن باید از تقسیم بندی گروه ها پرهیز کنیم، بهترین حالت گروه بندی بلاک ها به صورت $64, 64$ می باشد در نتیجه هزینه نهایی به صورت زیر محاسبه می شود:

$$C_{\text{total}} = 64C + 64C \times 2 + 4C + 4C = 200C$$

(ج) پیچیدگی تاخیر این جمع کننده به صورت $O(log_2(n))$ می باشد و پیچیدگی هزینه نیز $O(n)$ که n همان تعداد بیت های ورودی است. این جمع‌کننده به دلیل مصرف کمتر منابع سخت‌افزاری و کارایی مناسب در مدارات کم‌مصرف، پردازنده‌های نهفته و طراحی‌های بهینه از نظر مساحت تراشه استفاده می‌شود.
مقایسه هزینه و تاخیر این جمع کننده با جمع کننده های (آ) و (ب) به صورت زیر است:

$$C_{\text{BK}} < C_{\text{b}} < C_{\text{a}}$$

$$D_{\text{a}} < D_{\text{BK}} < D_{\text{b}}$$

(د) پیچیدگی تاخیر این جمع کننده به صورت $O(log_2(n))$ می باشد و پیچیدگی هزینه نیز $O(nlog_2(n))$ که n همان تعداد بیت های ورودی است. این جمع‌کننده در مدارهای دیجیتال برای انجام عملیات جمع سریع مورد استفاده قرار می‌گیرد. به دلیل ساختار درختی و تأخیر کم، در پردازنده‌ها، واحدهای محاسباتی سریع (ALU)، و طراحی مدارهای VLSI که نیاز به بهره‌وری بالا دارند، کاربرد دارد.
مقایسه هزینه و تاخیر این جمع کننده با جمع کننده های (آ) و (ب) به صورت زیر است:

$$C_{\text{b}} < C_{\text{KS}} < C_{\text{a}}$$

$$D_{\text{KS}} < D_{\text{a}} < D_{\text{b}}$$