به نام خداوند بخشنده مهربان

معماری کامپیوتر

دانشکده مهندسی کامپیوتر

دکتر اسدی
بهار ۱۴۰۳

مهدی علی نژاد، ۴۰۱۱۰۶۲۶۶

تمرین سوم

سوال ۱

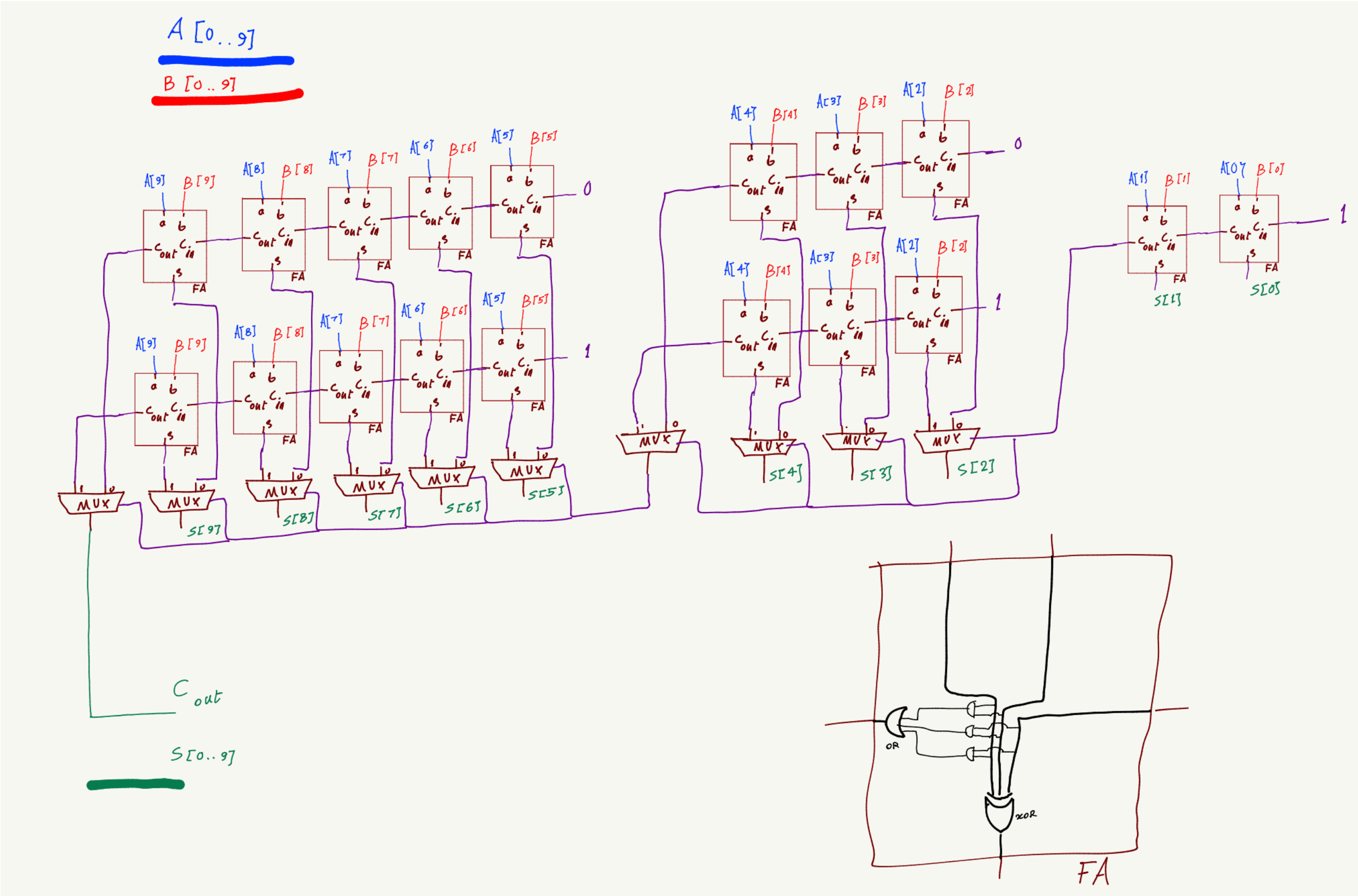
۱. برای جمع کردن دو عدد ۱۰ بیتی از یک جمع‌کننده Carry Select چند مرحله‌ای استفاده می‌کنیم که در مرحله اول ۲ بیت، در مرحله دوم ۳ بیت و در مرحله سوم ۵ بیت را به کمک Ripple Adder جمع می‌کند. هزینه و تأخیر هر گیت را در این طراحی به ترتیب C_g و D_g در نظر یگیرید و هزینه و تأخیر هر MUX را در این طراحی به ترتیب $C_m = 3C_g$ و $D_m = 2D_g$ در نظر بگیرید. فرض شده همه تمام‌جمع‌کننده‌ها[1] با استفاده از توابع زیر پیاده‌سازی شده‌اند:

$$S = XOR(A, B, C_{in})$$

$$Cout = A.B + A.C_{in} + B.C_{in}$$

دقت شود که پهنای ورودی تمامی گیت‌ها و MUX ها یک بیتی هستند و همچنین امکان دسترسی به OR و XOR با سه ورودی وجود دارد که تأخیر و هزینه‌اشان با سایر گیت‌ها یکسان است. (توجه: استفاده از سایر گیت‌های دارای چند ورودی یا MUX به جز $2:1$ ممکن نیست).

Adder بالا را ترسیم کنید. هزینه و تأخیر طراحی بالا را بیان کنید و با یک Ripple Carry Adder معادل مقایسه کنید.



ابتدا هزینه را محاسبه می کنیم. هر FA به اندازه ی $۵C_g$ هزینه دارد و هر ماکس $۳C_g$. در کل نیز ۱۸ FA و ۱۰ MUX استفاده شده است. پس مجموع هزینه برابر است با:

$$۱۸ * ۵C_g + ۱۰ * ۳C_g = ۱۲۰C_g$$

حال برای تاخیر، در هر FA برای رسیدن به S، D_g و برای رسیدن به C_{out}، $۲D_g$ تاخیر داریم.

حال تاخیر مورد نیاز برای رسیدن به هرکدام از بیت های حاصل را حساب می کنیم و بیشترین آنها را به عنوان تاخیر کل اعلام می کنیم.

$$S[۰], S[۱] \to D_g$$

$$S[۲ \sim ۴] \to max(C_{out_{S[۰\sim۱]}}, D_g) + ۳D_g = ۴D_g + ۲D_g = ۶D_g$$

(time needed for calculating carry and inputs)

$$S[۵ \sim ۹] \to max(C_{out_{S[۲\sim۴]}}, D_g) + ۳D_g = ۶D_g + ۲D_g = ۸D_g$$

(time needed for calculating carry and inputs)

$$C_{out} = max(C_{out_{S[۲\sim۴]}}, ۱۰D_g) + ۲D_g = ۱۲D_g$$

پس تاخیر کل برابر با تاخیر حساب شدن C_{out} است که برابر با ۱۲ D_g است. اگر می خواستیم همین کار را با Ripple carry adder انجام بدیم، می بایست از ۱۰ FA استفاده می کردیم، در این حالت هزینه ی صرف شده برابر با $۵۰C_g$ و تاخیری که در جواب مشاهده می کنیم برابر با $۲۰D_g$ خواهد بود.

سوال ۲

۲. فرض کنید، قصد طراحی یک Carry Select Adder با اندازه ۱۲۸ بیتی را داریم. برای این که کمترین میزان تأخیر را داشته باشیم، سایز بلاک‌ها باید در چه اندازه‌ای باشد. فرض کنید که تأخیر تمام‌جمع‌کننده و MUX یکسان باشد. **توجه شود که لزومی ندارد بلوک‌ها ابعاد یکسانی داشته باشند و می‌توانند در ابعاد متفاوت باشند.**

فرض کنید که i بیت این عدد را جدا کرده و C_{out} آن در $i*D+D_{old}$ آماده می شود. پس ما همین مقدار زمان را می توانیم برای محاسبه کردن ورودی های ماکس های بعدی استفاده کنیم که میشه $i+\frac{D_{old}}{D}$ بیت بعد. و کری آنها نیز در $(i+۱)*D+D_{old}$ آماده می شود. پس طبق همین اثبات به صورت استقرایی نشان می دهیم که برای بهینه بودن جمع کننده مان سایز بلاک ها باید دنباله ای از اعداد طبیعی باشد.

حال برای اینکه پیدا کنیم چه نقطه ی شروعی می تواند ما را به کمترین میزان تاخیر برساند مسئله ی بهینه سازی زیر را حل کنیم.

$$\frac{n(n+۱)}{۲}-\frac{i(i+۱)}{۲}=۱۲۸, min_i(n+i+۱)$$

$n+i+۱$ در واقع مقدار تاخیر ما است که در تلاش برای مینیمم کردن آن هستیم. طی بهینه سازی های طولانی، به این نتیجه می رسیم که بهترین حالت زمانی اتفاق می افتد که $i=۱, n=۱۵$ باشد، البته با کف گرفتن به این اعداد رسیدیم و تعدادی بیت اضافه می آید.

پس از بلوک هایی با سایز های

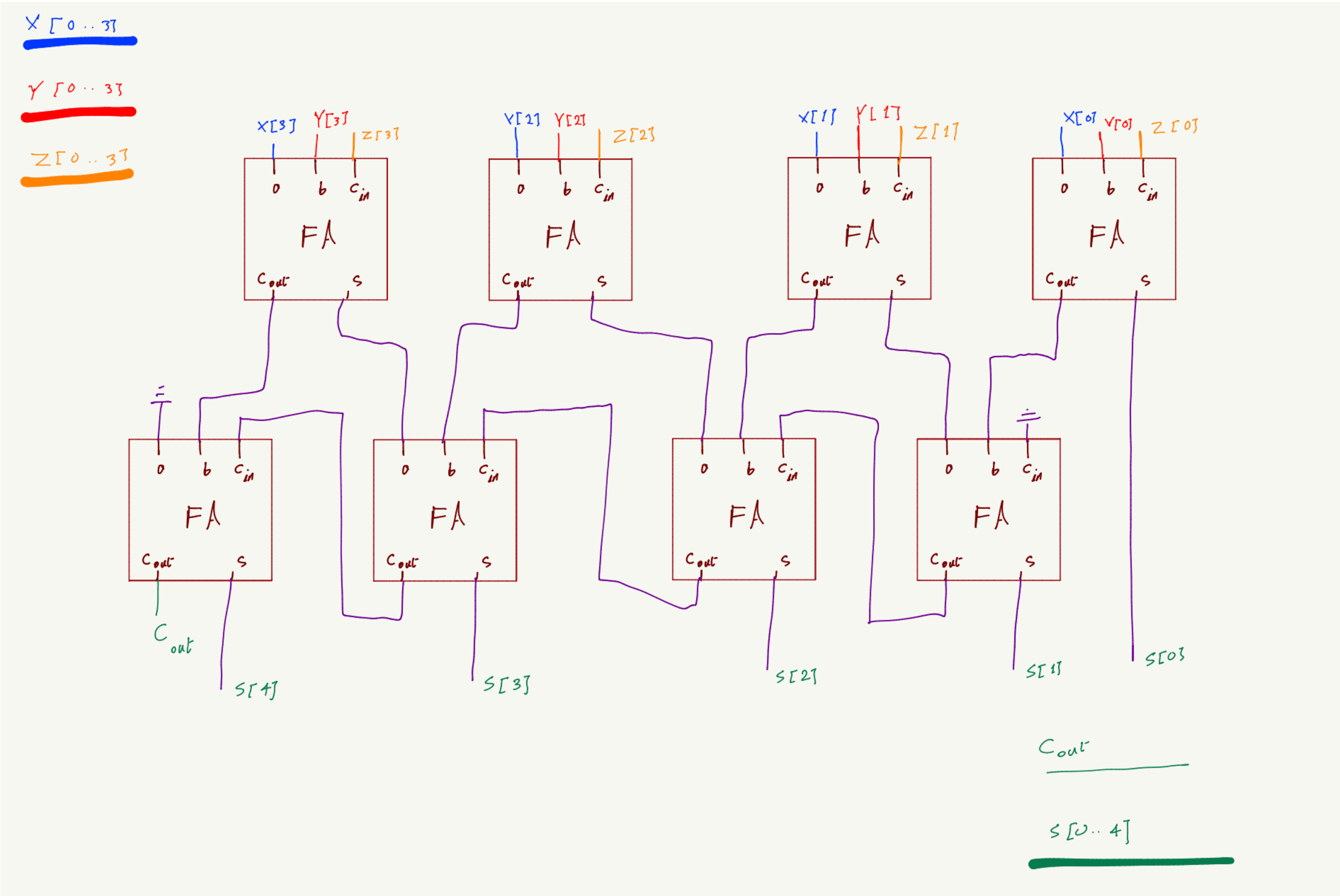
$$۲(normal), ۲, ۳, ..., ۱۵, ۷$$

استفاده می کنیم و تاخیر کل مدار $۱۷D$ می شود.

سوال ۳

۳. در مورد Carry Save Adder تحقیق کنید و نحوه کارکرد آن و تفاوت آن با سایر جمع‌کننده‌هایی که در درس با آن آشنا شدید را توضیح دهید. سپس یک Carry Save Adder با ۸ عدد تمام‌جمع‌کننده طراحی کنید که بتواند اعداد ۴ بیتی را جمع کند. برای جمع زدن Carry و Sum در مرحله اول، می توانید از روش Ripple Carry Adder استفاده کنید.

carry save adder یک طراحی از جمع کننده است که برای جمع اعداد ۳ یا بیشتر استفاده می شود. طرز کار آن نیز به این صورت است که ابتدا خود حاصل را بدون در نظر گرفتن کری حساب می کند و به صورت جداگانه کری ها را محاسبه می کند. سپس عدد تشکیل شده از کری ها و حاصل را جمع می کند تا به حاصل جمع نهایی برسد. در این روش چون کری ها به صورت جدا جمع زده می شوند، آن اثر دنباله وار افزایش تاخیر مدار را ندارند.

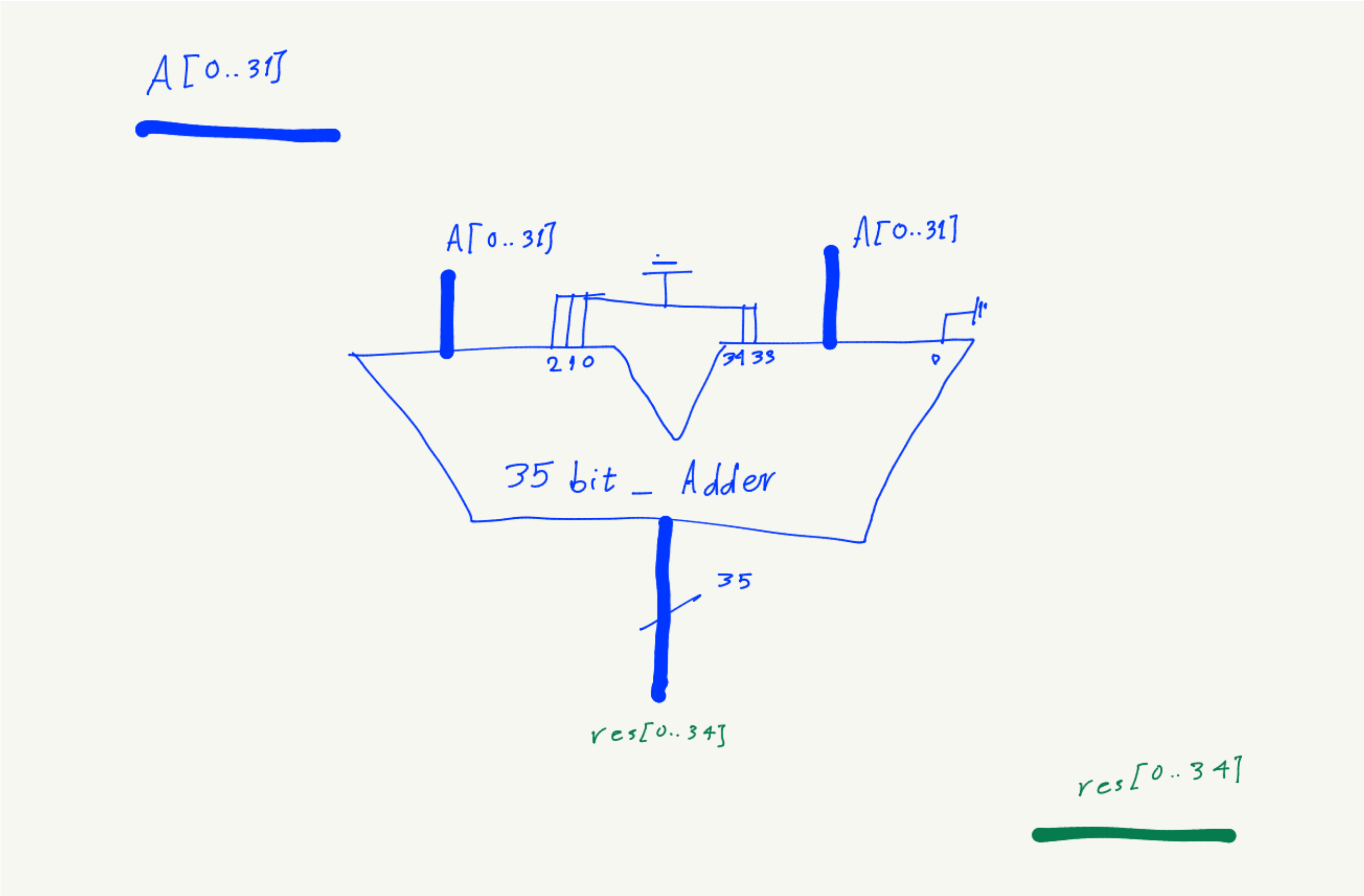


سوال ۴

۴. در کامپیوترهای امروزی مدارهایی به منظور ضرب یک ثابت[۲] در یک عدد ثابت وجود دارد. به عنوان مثال در کامپیوترهای امروزی برای ضرب کردن یک ثابت در توان‌های عدد دو از Barrel Shifter استفاده می‌شود. همان طور که می‌دانید صرفاً انتقال عدد به سمت چپ به اندازه‌ی n عملاً همان ضرب عدد در 2^n هست. اما یک عملیات دیگر که بسیار مورد استفاده است ضرب یک عدد دودویی در ۱۰ است. از این رو فرض کنید که قرار است یک مدار طراحی کنید که عدد بدون علامت ۳۲ بیتی را به صورت دودویی بگیرد و آن را در عدد ۱۰ ضرب کند. برای این منظور سریع‌ترین مدار ممکن را طراحی نمایید.

$$۱۰a = ۲^۳a + ۲a$$

طبق معادله ی بالا، برای ۱۰ برابر کردن عددی کافیست ۸ برابر آن را با ۲ برابرش جمع کنیم، که در واقع تنها تعدادی شیفت و یک جمع نیاز است.



*می توانیم از جمع کننده ی ۳۶ بیتی استفاده کنیم و اون موقع دیگر هیچ از دست رفت اطلاعات و اورفلویی نخواهیم داشت.(دسترسی به فایل شماتیک نداشتم ادیتش کنم)

سوال ۵

۵. اگر بخواهیم ضرب علامت‌دار دو عدد ۰۰۰۰۱۱ و ۰۱۱۱۰۱ را با الگوریتم booth انجام دهیم، با فرض اینکه هر عمل جمع ۱۰ns و هر عمل انتقال ۲ns و هر مکمل‌گیری ۵ns طول بکشد، زمان ضرب با این الگوریتم و حاصل ضرب چه مقدار خواهد بود (مراحل ضرب به روش Booth نوشته شود)؟

(p)multiplician = 011101, (-p)-multiplician = 100011, (Q)multiplier = 000011, (a)acc = 000000, b = 0

1. $\overline{Q_{LSB}b} = 10 \rightarrow$ subtract,
a = 100011
then asr(arithmetic shift right);
a = 110001, Q = 100001, b = 1

2. $\overline{Q_{LSB}b} = 11 \rightarrow$ asr;
a = 111000, Q = 110000, b = 1

3. $\overline{Q_{LSB}b} = 01 \rightarrow$ add,
a = 010101
then asr;
a = 001010, Q = 111000, b = 0

4.

ادامه می دهیم تا به تعداد بیت های مضروب، شیفت به راست داشته باشیم. حاصل نهایی برابر است با:

$$\overline{aq} = ۰۰۰۰۰۱۰۱۰۱۱۱$$

زمان نهایی:(یک عمل جمع، یک تفریق و ۶ شیفت یا همان انتقال)

$$t = ۱ * (۵ + ۱۰) + ۱ * (۱۰) + ۶ * (۲) = ۳۷ns$$

سوال ۶

۶. با توجه به الگوریتم Booth به سوالات زیر پاسخ دهید.

آ) حداکثر تعداد جمع و تفریق در ضرب booth را برای چهار حالت یعنی ۱) اعداد علامت‌دار n بیتی و تعداد بیت‌های اعداد ورودی زوج ۲) اعداد علامت‌دار n بیتی و تعداد بیت‌های اعداد ورودی فرد ۳) اعداد بدون‌علامت n بیتی و تعداد بیت‌های اعداد ورودی زوج ۲) اعداد بدون‌علامت n بیتی و تعداد بیت‌های اعداد ورودی فرد را به صورت پارمتری محاسبه کنید.

ب) حاصل ضرب 13- * 9- را به روش ضرب booth بدست آورید و تعداد جمع و تفریق‌ها رامحاسبه نمایید. (n=۵)

(آ) i. بدترین حالت وقتی است که بیت های ضرب شونده یکی در میان صفر و یک باشند، چون تعداد بیت ها زوج است، اگر کم ارزش ترین بیت از یک شروع شود، به بدترین حالت رسیده ایم که در آن حالت، n جمع و تفریق داریم.

ii. این حالت نیز مثل حالت بالاست، بدترین حالت زمانی اتفاق می افتد که ضرب شونده با یک شروع شود و یکی در میان صفر و یک باشد. در این حالت نیز n جمع و تفریق نیاز داریم.

iii. تفاوت این حالت با حالت عادی الگوریتم این است که اگر ضرب شونده به ۱ ختم می شود، آنگاه یک جمع نهایی نیز خواهیم داشت، همچنین برای ضرب کننده. پس در حالتی که زوج بیت دارد تفاوتی نمی کند که با یک شروع شود یا صفر ولی همچنان بدترین حالت وقتی است که بیت ها یکی در میان باشند، در این حالت $n+۱$ جمع و تفریق مورد نیاز است.

iv. برای این حالت نیز بدترین وضعیت زمانی است که ضرب شونده با یک شروع شود و به یک ختم شود و در این بین یکی در میان مثبت و منفی باشد که در کل به $n+۲$ جمع و تفریق نیاز داریم.

(ب)

$$-۱۳ = ۱۰۰۱۱, -۹ = ۱۰۱۱۱$$

(p)multiplician = 10011, (-p)-multiplician = 01101, (Q)multiplier = 10111, (a)acc = 000000, b = 0

1. $\overline{Q_{LSB}b} = 10 \rightarrow$ subtract,
a = 01101
then asr(arithmetic shift right);
a = 00110, Q = 11011, b = 1

2. $\overline{Q_{LSB}b} = 11 \rightarrow$ asr;
a = 00011, Q = 01101, b = 1

3. $\overline{Q_{LSB}b} = 11 \rightarrow$ asr;
a = 00001, Q = 10110, b = 1

4. $\overline{Q_{LSB}b} = 01 \rightarrow$ add,
a = 10100
then asr;
a = 11010, Q = 01011, b = 0

5. $\overline{Q_{LSB}b} = 10 \rightarrow$ subtract,
a = 00111
then asr;
a = 00011, Q = 10101, b = 1

end;

حاصل نهایی برابر است با:

$$\overline{aQ} = ۰۰۰۱۱۱۰۱۰۱$$

و در مجموع ۲ تفریق و یک جمع استفاده شد.