1.

(آ) مشابه نظر اقای بهرامیان، 9 7 6 4 3 3 جواب مساله است و خداشاهده نمیدونم دقیقا با چه محاسباتی این جواب حاصل شده... نهایتا بتونم یه کد سی بزنم که این جواب رو با چک کردن حالت بندی بده... بهرحال تاخیر حاصل از این تقسیم بندی را محاسبه میکنیم.

در ابتدا، سه بیت نخست و سه بیت دوم با یکدیگر جمع می‌شوند. اولین بیت کنترلی توسط مالتی‌پلکسر اول، پس از ۶ نانوثانیه، از پایان جمع سه بیت اول، تولید می‌شود. در همین بازه زمانی، جمع سه بیت دوم نیز کامل شده است. پس در ادامه، پس از ۹ نانوثانیه، بیت کنترلی بعدی تعیین می‌گردد. در این بازه، طی ۸ نانوثانیه اول، جمع چهار بیت بعدی نیز انجام شده و در نتیجه، بیت کنترلی بعدی در نانوثانیه دوازدهم در دسترس قرار می‌گیرد. در طول این ۱۲ نانوثانیه، جمع شش بیت بعدی نیز تکمیل شده و بیت کنترلی بعدی در نانوثانیه پانزدهم تولید می‌شود. در این مرحله، هفت بیت دیگر نیز جمع شده‌اند و بیت کنترلی بعدی در نانوثانیه هجدهم به دست می‌آید. در نهایت، طی این روند، جمع ۹ بیت باقی‌مانده نیز انجام شده و خروجی نهایی در 21 نانوثانیه آماده می‌گردد.

(ب) در این عملیات میدانیم که حتما به تعداد بیت ها شیفت داریم. در حالتی که تمامی بیت‌ها صفر یا یک باشند، تنها به عملیات شیفت نیاز داریم که حداقل تاخیر امکان پذیر و همان 64 نانو ثانیه هست. مانند ورودی 32 بیتی 0 دادن.

اما در حالتی که هیچ دو رقمی مشابه نباشند (تغییر مداوم بین صفر و یک)، اجرای عملیات جمع و تفریق در هر مرحله ضروری است. برای مکمل هم 1 تاخیر xor داریم که 22= 21+1. (حدس میزنم محاسباتش با32*16+64هم هندل بشه) پس سر جمع به عنوان کران بالا 768 را داریم.مانند ورودی 0xAAAAAAAA

2.

(آ) در Carry-Skip Adder (CSKA)، بیت‌ها رو به بلوک‌هایی تقسیم می‌کنیم.در هر بلوک، جمع به‌صورت Ripple-Carry انجام می‌شود، اما یه مکانیزم خاص برای skip کردن carry وجود دارد. ایده اینه که اگر تمام بیت‌های یک بلوک اجازه‌ی عبور carry رو بدن (یعنی propagate باشن ،که با 1 بودن pi=ai⊕bi ها سنجیده میشود) ، دیگر لازم نیست carry از طریق تک‌تک بیت‌ها منتقل شود. در این صورت، carry می‌تواند از کل بلوک بپرد (skip کنه) و مستقیم به بلوک بعدی برود. یعنی بیت نقلی از ورودی مستقیماً به خروجی منتقل میشود.

(ب) در Ripple-Carry Adder، carryباید از بیت 0 تا بیت n-1 به ترتیب برود، و این باعث زمان زیاد می‌شود. در Carry-Skip اگر بیت‌های یک بلوک propagate باشن‌د، carry فوری به بلوک بعدی منتقل می‌شوند. که این باعث کاهش تاخیر خوبی میشود اما اگر هیچ بلوکی propagate نباشد، هیچ فرقی با یک Ripple-Carry معمولی ندارد.

(ج) فرض کنید یک بلوک ۳ بیتی داریم با بیت‌های A_2,A_1,A_0 و B_2,B_1,B_0

برای هر بیت i ، تعریف کنیم $p_i=A_i \oplus B_i$. برای اینکه carry از ابتدا تا انتهای بلوک skip بشود، باید: $P=p_2 \cdot p_1 \cdot p_0=1$ شود. پس P را به ورودی یک MUX میدهیم. اگر P = 1 باشه، یعنی تمام بیت‌ها propagateهستن، پس carry-in می‌تونه مستقیماً به carry-out بره. پس ورودی 1 مالتی پلکسر carry-in و ورودی 0 آن ، carry تولیدشده از Ripple-Carry داخل بلوک است. خروجی مالتیپلکسر نیز carry-out است.

(د) فرض کنید هر بلوک n بیتی باشد. در هر بیت، احتمال $propagate$ بودن (یعنی $p_i=1$) برابر با 0.5 هست (یکسان بودن یا نبود A_i و B_i) برای اینکه $carry$ از یک بلوک n بیتی $skip$ شود ، باید: $P=p_0 \cdot p_1 \cdot \dots \cdot p_{n-1}=1$. پس احتمال اینکه $carry\ skip$ شود برابر است با: $P_{skip}=(0.5)^n$

3. (آ) ابتدا 011001×101000 را انجام می دهیم. پس:

$M = 011001, Q = 101000, n = 6$

LSB	Q	A	عملیات	Delay
0	101000	000000	load	0ns
0	010100	000000	ASR	8ns
0	001010	000000	ASR	ns8
0	000101	000000	ASR	ns8
0	000101	100111	Not + ADD = SUB	(4+12)ns
1	100010	110011	ASR	8ns
1	100010	001100	ADD	12ns
0	010001	000110	ASR	8ns
0	010001	101101	Not + ADD = SUB	(4+12)ns
1	101000	110110	ASR	8ns

حال جای دو عدد را برعکس میکنیم. پس:

$M = 101000, Q = 011001, n = 6$

LSB	Q	A	عملیات	Delay
0	011001	000000	load	0ns
0	011001	011000	Not + ADD = SUB	(4+12)ns
1	001100	001100	ASR	8ns
1	001100	110100	ADD	12ns
0	000110	111010	ASR	8ns
0	000011	111101	ASR	8ns
0	000011	010101	Not + ADD = SUB	(4+12)ns
1	100001	001010	ASR	8ns
1	010000	000101	ASR	8ns

1	010000	101101	ADD	12ns
0	1010000	110110	ASR	8ns

(ب) با جمع ستون تاخیر دو جدول بالا داریم:

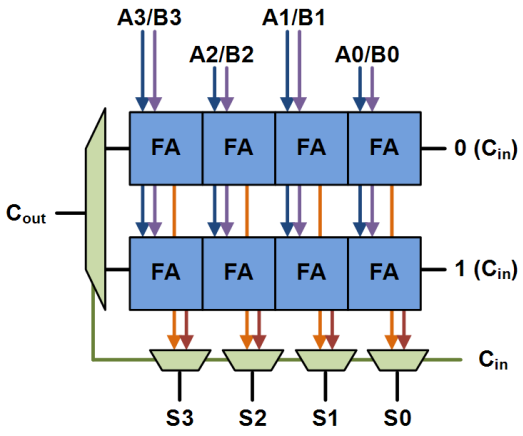
$Delay1 = 92\ ns$

$Delay\ 2 = 104\ ns$

در حالت اول، عملیات مکمل‌گیری تنها یک‌بار انجام می‌گیرد و نتیجه آن در یک ثبات ذخیره می‌شود تا در تکرارهای بعدی مجدداً مورد استفاده قرار گیرد. در مقابل، در حالت دوم، مکمل‌گیری در هر بار اجرای ضرب تکرار می‌شود که دلیل افزایش تاخیر حالت دوم نسبت به اول است. این افزایش تأخیر قابل توجیه است، چراکه در ضرب عدد 011001 نسبت به 101000، تغییرات بیشتری بین سطوح منطقی صفر و یک رخ می‌دهد که منجر به افزایش تعداد عملیات جمع می‌گردد.

4. جمع کردن در یک واحد زمانی (حالت پایه) حداکثر برای 1 بیت با یک عدد جمع کننده بدون مالتی پلکسر است. پس $f(1)=1$

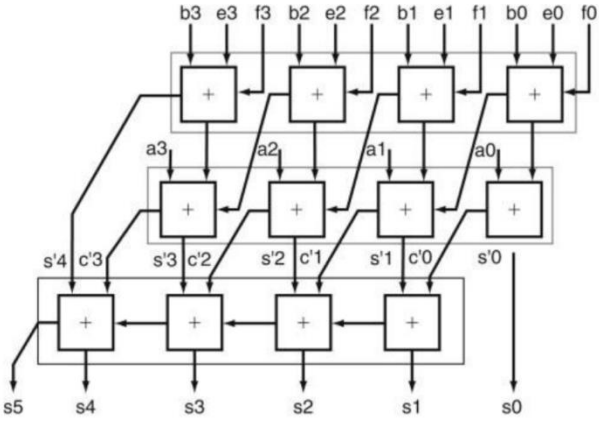
فرض کنید در n واحد زمانی حداکثر $f(n)$ بیت جمع زده باشیم، درین صورت در $n+1$ واحد زمانی میتوانیم $2f(n)$ بیت را جمع بزنیم. زیرا میتوان یک بلوک برای پیاده سازی $f(n)$ بیت اول و دو بلوک برای پیاده سازی $f(n)$ بیت دوم (یک بار با cin صفر و باری دیگر با 1) را در n واحد زمانی طی کرد و با $f(n)$ تا مالتی پلکسر که هر کدام یک بیت خروجی متناظر از دو بلوک $f(n)$ بیت دوم را به عنوان ورودی و $carry$ بلوک $f(n)$ بیت اول را به عنوان $select$ میگیرند.(چون گفتین به صورت مختصر توضیح بدیم، ثابت نمیکنم چرا $f(n+1)$ از $2f(n)$ بیشتر نمیتونه بشه) پس داریم $f(n)=2f(n)=2n$



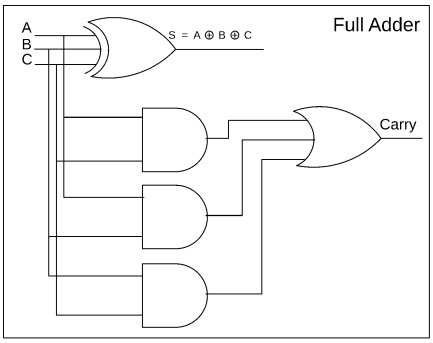
5. (آ) ایده‌ی $CarrySaveAdder$ این است که به جای اینکه $carry$ رو بلافاصله منتقل کنیم، ما $carry$ و sumرو جداگانه ذخیره می‌کنیم، بدون اینکه $carry$ رو بلافاصله با بقیه‌ی بیت‌ها جمع بزنیم.

عملکرد این روش برای جمع 4 عدد همانطور که در تصویر مشخص شده است، به 3 عدد قطعه نیاز دارد که هر کدام بیت های متناظر 3 دسته عدد را جمع میکنند و $carry$ و sum هر جمع 3 بیت را خروجی میدهند.

در سطح اول بیت های متناظر 3 عدد (b_i,e_i,f_i) باهم جمع میشوند و در سطح دوم بیت های متناظر a با sum های خروجی از بین های متناظرش در 3 عدد



دیگر، و با carry مرحله قبلش (a_i, sum_i, c_{i-1}) جمع میشود. (توجه کنید این رفتن c_i به جمع کننده $i+1$ به دلیل این است که ارزش بیتی آن بیشتر از $a_i,b_i,..$ است چون از جمع حداقل 2 بیت 1 از b_i,e_i,f_i ایجاد شده است). به طور مشابه در سطح سوم چیزی مشابه Ripple carry (c''_{i-1}, c'_{i-1}, s'_1) که c''_{i-1} در جمع کننده قبل همین سطح تولید شده، در هر جمع کننده با هم جمع میشوند و (البته چون c'_{-1} و c''_{-1} نداریم، s'_0 به طور مستقیم خارج میشود. در نهایت $s_5s_4s_3s_2s_1s_0$ پاسخ ماست.



(ب) در هر کدام از سطح اول و دوم، تاخیر کل برابر تاخیر یک full adder است. که مشابه شکل روبرو برابر تاخیر یک گیت XOR است که برابر همان d است. پس تا اکنون $2d$، حال سراغ سطح سوم می رویم که میدانیم تاخیر آن همان تاخیر یک carry_look_ahead برای دو عدد 4 بیتی است. محاسبه Giو Piها یک گیت or،یک and و یک xorدارد. محاسبه هر Piیک گیت xor و محاسبه هر Giیک and دارد. لذا سطح سوم کل تاخیر $3d$ را دارد و روی هم رفته تاخیر برابر است با: $d + d + 3d = 5$