تمرین دوم تئوری ساختار کامپیوتر

سیدمحدرضا جوادی 402105868

پاییز 1403،دکتر اسدی

Table of Contents

سوال 1)

الف)

هرخانه نمایانگر بازه هایی 1 ثانیه ای است. چپ ترین خانه 0-1 و بعدی 0-2 و... است. کار کردن با w و دستور ورودی خروجی با IO و سرکشی با p نمایش می دهیم. iیعنی کاری رخت نداده(idle). استفاده از / به معنی همزمان است مثل w/p که هم نظرسنجی می کند وهم مشغول کار

IO	w	w	w	w	w	w/p	i	i	i	p	IO	w	w	w	w	w	w	i	i	p	IO	w	w

ب)

نماد گذاری ها مانند قبل، وقفه را با iنشان می دهیم، اینجا idleنداریم کلا

IO	w	w	w	w	w	w	i	i	i	IO	w	w	w	w	w	w	i	i	i	IO	w	w	w

سوال 2)

توجه: من گاهی به صورت لفظی بجای " سیگنال را فعال می کند" از واژه "سیگنال را می فرستد" استفاده کردم، در این متن نوشته من این دو معادلند و صرفا یک لفظ متفاوت است و ماهیت امر که سیگنال را 0 یا 1 می کند تغییر نمی دهد، صرفا هنگام بیان اینگونه توصیف شده. هنگام مطالعه مطلب من این مورد را به خاطر داشته باشید وگرنه می تواند موجب سردرگمی یا سوء تفاهم میان من و شما مصحح گرامی شود.

الف)

در کل اهمیت آنان وجود راهی برای هماهنگی در خلل وجود کلاک است، حال دقیق تر توضیح می دهیم:

اهمیت و روش کار strobe:

با ارسال سیگنالstrobe فرستنده با گیرنده همگام و هماهنگ می شوند تا اطلاعات منتقل شود. اهمیت این روش در همگام سازی فرستنده و گیرنده و اطمینان از ارسال و دریافت صحیح است.

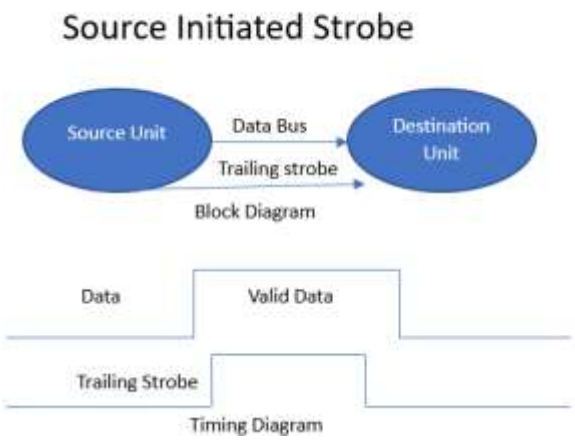
اهمیت و روش کار hand-shaking:

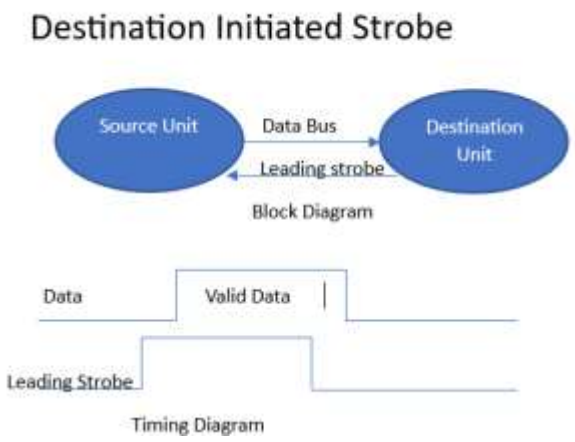
در این حال 2 سیگنال data valid به معنی ارسال شدن اطلاعات صحیح و data accepted به معنی دریافت صحیح اطلاعات را داریم. اهمیت این روش در کنار ایجاد هماهنگی و همگام کردن ارسال آسنکرون، جلوگیری از خطاها و اطمینان از انتقال صحیح داده هاست.

ب)

به دو شکل انجام می شود:data transfer initialized by source و data transfer initialized by destination، منبع سیگنال strobe را بفرستد، یا مقصد بفرستد. در صورتی که منبع ابتدا سیگنال بدهد به آنtrailing strobe یا سیگنال دنبال کننده می گویند (چون بعد از ارسال داده ارسال می شود) و درصورتی که مقصد شروع کننده و فرستاننده سیگنال باشد به آن leading strobe یا سیگنال رهبر یا جلودار می گویند (چون قبل از ارسال داده فرستاده می شود)

ج)





کلیات عملیات ها از نمودار های بالا مشخص است. تفاوت عمده در این است که کدام یک سیگنال strobe را می فرستد و ارتباط را شروع می کند.

data transfer initialized by source مرحله به مرحله (در منبع صحبت هایم 3 مرحله است و من کمی باز کردم):

1) فرستنده دیتا را در اتوبوس دیتا بارگذاری می کند.
2) فرستنده سیگنال strobe را فعال می کند
3) مقصد با مشاهده فعال بودن سیگنال شروع به خواندن می کند
4) بعداز خوانش سیگنال strobe غیر فعال می شود و پایان فرایند.

data transfer initialized by destination مرحله به مرحله:

1) مقصد (destination) سیگنال strobe خود را فعال می کند تا به منبع (source) اطلاع دهد آماده دریافت داده است.
2) منبع با مشاهده فعال بودن سیگنال اقدام به بارگذاری داده می کند تا داده های جدید بر اتوبوس داده قرار گیرند
3) مقصد با دریافت و خوانش داده ها سیگنال strobeرا غیر فعال می کند و فرایند به اتمام می رسد.

د)

همانطور که احتمالا تا الان برایتان سوال شده باشد، اینکه چه زمانی فرایند خوانش دیتا یا ارسال دیتا متوقف می شود به نظر کار بدیهی نمی آید و می تواند چالش زا باشد. بله، در هنگام ورودی ها و خروجی های ناهمگام این فرایند مشکلات مدیدی را در دریافت اطلاعات صحیح و سالم از منبع به وجود می آورد. دو چالش عمده عبارتند از:

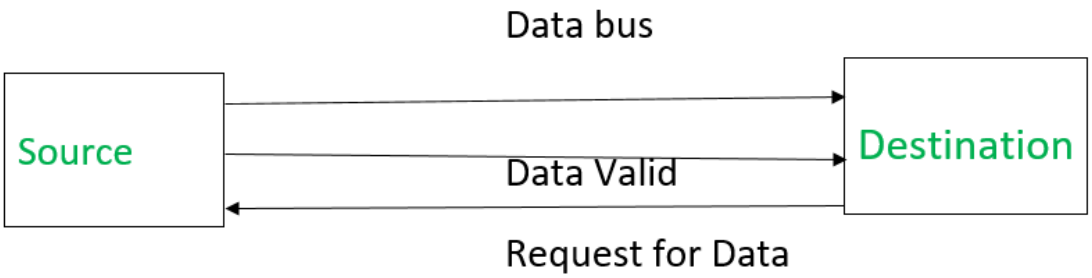
1. در هنگامی که منبع آغاز گر است، فرض می شود که مقصد دیتارا دریافت کرده و خوانده است. با این وجود هیچگاه چنین تضمینی نداریم.
2. در حالتی که مقصد آغاز گر است، فرض می شود منبع اطلاعات را درbus قرار می دهد ولی باز هم تضمینی دراین باره نداریم.

در فرایند handshaking با ایجاد دو سیگنال جدید بجای strobe منبع و مقصد از ارسال و دریافت خود به طرف مقابل خبر می دهند تا چنین ناهماهنگی هایی در ارسال و دریافت پیش نیاید.

ه)

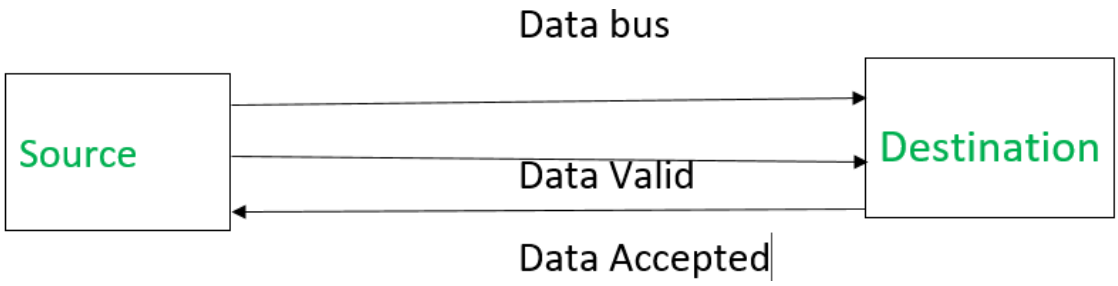
تقسیم بندی hand shakingها هم مشابه بخش قبل است، دو حالت، آغازگر منبع باشد یا مقصد.

برای معرفی نام سیگنال ها در هر حالت توضیحات مختصری می دهیم تا نشان دهیم هر سیگنالی که فلان کار می کند فلان نام را داراست.



حالت اول:Destination initiated Handshaking

در این حالت که مقصد آغاز گر است، سیگنالی با نام request for dataیا درخواست برای داده ارسال می شود، منبع داده را در اتوبوس داده بارگذاری کرده و سیگنال data valid یا داده معتبر است ارسال می کند (یا به لفظ دیگر فعال می کند)



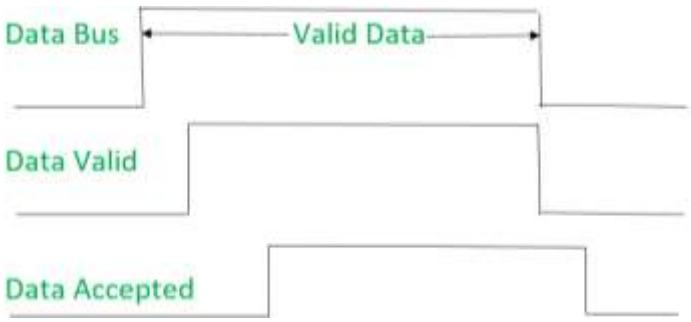
حالت دوم:Source initiated Handshaking

در این حالت منبع دیتارا در اتوبوس داده بارگذاری کرده، سیگنال داده معتبر است (data valid) را فعال می کند. مقصد پس از دریافت و خوانش سیگنالی با نام data accepted یا دیتا مورد قبول واقع گشت منبع را از دریافت صحیح آگاه می کند.

و)

هرچند در بخش قبل تقریبا همه مراحل را بیان کردیم، ولی این بار با رویکرد بررسی جزئی تر به آن ها می پردازیم.

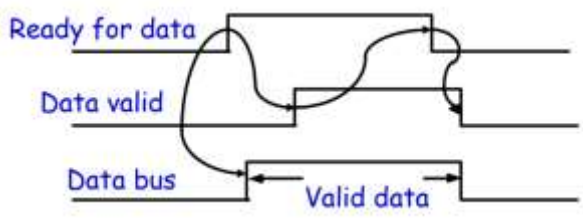
Source initiated Handshaking:



1. منبع دیتارا بارگذاری می کند در اتوبوس داده
2. سیگنال داده معتبر است را فعال می کند
3. مقصد متوجه سیگنال می شود و پس از دریافت و خوانش سیگنال data acceptedرا فعال می کند

4. منبع با دیدن فعال شدن سیگنال data accepted سیگنال داده معتبر است را غیر فعال می کند
5. مقصد هم با غیر فعال شدن سیگنال داده معتبر است، سیگنال داده مورد قبول واقع گشت را غیر فعال می کند و فرایند به اتمام می رسد

:Destination initiated Handshaking



این مورد عکس را از اسلاید ها برداشتم

1. هنگام نیاز مقصد، سیگنال درخواست برای داده فعال می شود
2. منبع با دیدن فعال بودن این سیگنال، داده را در اتوبوس قرار داده و سیگنال data validرا می فرستد
3. مقصد بعد دیدن فعال بودن سیگنال معتبر بودن داده، آن را دریافت کرده و سیگنال درخواست داده را غیرفعال می کند.
4. منبع با غیر فعال شدن سیگنال درخواست، سیگنال داده معتبر را غیرفعال کرده و عملیات به اتمام می رسد.

ز)

ددلاین پایین آمدن آن، زمان پایین آمدن dataاست چرا که ممکن است بعد پایین آمدن دیتا اطلاعات اشتباه در آن قرار بگیرد و مقصد به اشتباه بخاطر فعال بودن سیگنال strobeهمچنان درحال خوانش و دریافت دیتا است. پس بله برای اطمینان بهتر است زودتر پایین بیاید(نهایتا همزمان با دیتا، ولی حتی اپسیلون ثانیه ای بعد هم نباید باشد)

ح)

این هم بله چرا که منبع اطلاع ندارد که دریافت شده یا نه و ممکن است data bus سریعتر پایین بیاید و مشکلات ذکر شده در قسمت قبل رخ دهد. پس مقصد بهتر است زودتر از پایین آمدن دیتا خوانش را قطع کند.

ط)

بله مشابه بالا درصورتی که دیرتر پایین بیاید مقصد دیتای اشتباهی را به جای دیتای درست می گیرد.

سوال3.

منابع:مقاله https://user.eng.umd.edu/~blj/papers/ieeetc65-1.pdf

سایت ها : https://en.wikipedia.org/wiki/Memory_refresh#Types_of_refresh_circuits

https://www.researchgate.net/figure/Timing-parameters-of-distributed-DRAM-Refresh_fig1_261049261

https://www.geeksforgeeks.org/difference-between-sram-and-dram/

https://www.allaboutcircuits.com/technical-articles/introduction-to-dram-dynamic-random-access-memory/

الف)

هر قطعه از DRAM متشکل از یک خازن و یک ترانزیستور است که اطلاعات را در خازن به شکل بار الکتریکی ذخیره می کند و ترانزیستور مسیر دریافت از و ریختن اطلاعات بر خازن را فراهم می کند (مانند یک سوییچ). شارژ بودن یک خازن به عنوان 1 و خالی بودنش 0 محسوب می شود.

ب)

همانطور که در بخش قبل اشاره کردیم این نوع از حافظه از خازن ها تشکیل می شود. با گذشت زمان و نبود انرژی کم کم خالی شده(دشارژ) و اطلاعات از دست می رود و برای اطمینان از باقی ماندن اطلاعات باید اطلاعات را تازه بکنیم. دو پارامتر مهم تازه کردن عبارتند از refresh interval و refresh overhead.

refresh interval:

همانطور که از اسمش پیداست به زمان بین سایکل های رفرش اشاره می کند. یعنی فاصله زمانی بین دو رفرش را می گویند. اهمیت این پارامتر در سرعت رفرش کردن است، که نشان می دهد داده های چقدر سریع باید خوانده و تازه شوند.

refresh overhead:

سربار زمانی، همانطور که از اسمش حدس می زنید به معنای این است که سیستم چقدر زمان از کل زمان فعالیتش را به تازه کردن اختصاص می دهد. هرچه این مقدار کمتر باشد انرژی کمتری برای این کار می گذارد و برای بقیه فعالیت ها زمان و انرژی بیشتری می تواند بگذارد و بهینه تر شود. به بیان دیگر به نوعی با اینکه چند بار در یک واحد زمانی رفرش می کند معادل است(f~1/T)

ج)

مورد اول این است که پردازنده مجبور می شود زمانی را به رفرش کردن اختصاص دهد و اینگونه عملکردش کند تر می شود و بر زمان بقیه دستورات تاثیر می گذارد

مورد بعدی مصرف بالای انرژی است چرا که عملا در رفرش کار مفید و جدیدی انجام نمی دهیم و انرژی را صرف نگهداری کردن داده های سابق می کنیم

مورد آخر اضافه شدن هزینه و پیچیده شدن مدار است چرا که علاوه بر عملکرد های عادی هرپردازنده، حال باید عملیات تازه سازی را با نرخ مشخصی انجام دهد.

د)

زمان نگهداری یا retention time همانطور که از اسمش می توان حدس زد به مدت زمانی که داده در dram باقی می ماند گویند(طبیعتا بدون رفرش در آن زمان، وگرنه مدت نگهداری به مراتب بالا می رود). از آنجا کامپیوتر نیازمند رفرش کردن dram است و این کار انرژی و هزینه می برد، با دانستن زمان ماندگاری می توان برنامه ریزی بهتری برای رفرش ها داشت چرا که داده ای تازه ذخیره می شود نیازی به رفرش ندارد. به بیان دیگر می توان طوری برنامه ریزی کرد که استفاده حداکثری از زمان ماندن اطلاعات در حافظه انجام داد و زودتر از موعد رفرش کردن سودی ندارد. نتیجه استفاده بهینه تر و کمتر از رفرش، و صرف انرژی و هزینه کمتر است.

ه)

اختلاف سطح با رنگ قرمز مشخص شده و در جدول زیر تفاوت ها به شکلی نسبی بیان شده اند:

DRAM	SRAM
ارزان تر	گرانتر
مصرف انرژی بالاتر	مصرف انرژی پایینتر
سرعت پایینتر	سرعت بالاتر
نیاز به تازه کردن دارد	نیاز به تازه کردن ندارد
نیاز به1 ترانزیستور و 1خازن	نیاز به 6 ترانزیستور
ذخیره سازی اطلاعات در چگالی بالاتر	ذخیره سازی اطلاعات در چگالی پایینتر
در سطوح نزدیک بهmain memory	در سطح نزدیک به cache
مناسب برای داده های حجیم	مناسب برای داده های کوچکتر

و)

در سطوح نزدیک به cache ما سه لایه L1 و L2و L3را داریم. در این لایه ها یک تعادلی بین پرفورمنس، قیمت و حافظه و تاخیر برقرار است. در هرلایه: بحث قیمت واضح است، هرچه اندازه حافظه بیشتر شود قیمت بیشتر است. از سوی دیگر افزایش حافظه به کاهش سرعت منجر می شود پس نمی توان تا هر اندازه که خواستیم افزایش حافظه دهیم. این عوامل (قیمت و همینطور سرعت و پرفورمنس) باعث می شود تا در این سطوح حافظه نهانی اندازه حافظه محدود شود.

دلیل اینکه خود تعداد لایه ها محدود است تا حدی تاثیر گرفته از دلایل بالاست، به طور مثال هنگام اضافه کردن لایه ای جدید هزینه بسیاری را هم متحمل می شویم و همینطور به دلیل مراتب حافظه و انتقال داده ها تاخیر بین حافظه اصلی و پردازنده بسیار بیشتر می شود. همچنین دلایل جانبی و کم اهمیت تر دیگری مانند محدود بودن فضا در CPU برای اضافه کردن لایه جدید و پیچیدگی طراحی و انرژی مصرفی هم دخیل اند.

پس نتیجه می گیریم حتی با اولویت ماکسیمم کردن سرعت هم بیشتر کردن تعداد لایه ها لزوما مفید نیست و باید تعداد بهینه قرار دهیم.

سوال4.

الف)عملکرد

عمده تفاوت عملکرد در این است که پردازنده به سراغ دستگاه می رود یا دستگاه به دنبال پردازنده

وقفه سیگنالی است که دستگاه به پردازنده می فرستد و درخواست می کند که فورا به یک کار (Task) رسیدگی کند. پردازنده معمولا عملیات خود را (taskیی که درحال اجرا بود) متوقف کرده، وقفه را مدیریت(handle) می کند بعد به عملیاتی که قبل از آن درحال اجرا بود می پردازد. این فرایند بیشتر یک مکانیزم سخت افزاری است تا پروتکل. این وقفه ممکن است در هرلحظه که دستگاه نیاز به پردازش دارد اتفاق بیفتد.

در سیستم polling پردازنده در بازه های مشخص به طور مداوم به دستگاه پیغام می دهد و می پرسد که آیا نیاز به پردازش دارد یا نه. در واقع این مکانیسم بیشتر یک پروتکل است تا فرایند سخت افزاری. فقط در زمان های مشخص (بازه های متوالی) ممکن است رخ دهد.

ب)بهره وری انرژی

در polling به طور مداوم نیاز به انرژی داریم تا پردازنده از تک تک دستگاه ها بپرسید آیا به پردازش نیاز دارند یا نه، این باعث افزایش مصرف انرژی می شود. در آنسو در رویداد وقفه هرگاه نیاز به پردازش باشد خود دستگاه پیام می دهد، اینگونه انرژی کمتری تلف می شود. به طور کلی مصرف نظرسنجی از وقفه بیشتر است.

ج)پیچیدگی

سیستم وقفه یک مکانیزم سخت افزاری است، همینطور پیاده سازی interrupt handlingدر cpu هم پیچیدگی خودش را دارد علاوه بر این ها در یک کامپیوتر چند دستگاه ورودی خروجی داریم و هماهنگی وقفه ها بین آن ها باز هم بر پیچیدگی پیاده سازی می افزاید. در آنسو سیستم نظرسنجی هردفعه یک loopروی دستگاه ها می زند و هرکدام نیاز به پردازش داشت به ترتیب رسیدگی می شود، وگرنه باید تا دوره بعدی نظرسنجی دستگاه صبر کند. پس پیچیدگی سیستم وقفه از سیستم پرسش در کل بیشتر است هرچند هرکدام پیچیدگی خود را دارند.

د) پاسخگویی بی درنگ

در بسیاری از سیستم های بی درنگ از سیستم وقفه استفاده می کنند چرا که هر دستگاه هرگاه نیاز به پردازش داشت، فورا به پردازنده اطلاع می دهد. در آنسو سیستم نظرسنجی وقت زیادی را در حلقه ها و در پرسش تلف می کند و ممکن است تا نوبت به دستگاهی برای پردازش برسد وقت زیادی هدر رفته باشد. بنابراین دستگاه هایی که نیاز به پردازش فوری دارند بهتر است از سیستم وقفه استفاده کنند چون بلافاصله فعالیت پردازنده به آن ها اختصاص می یابد. در کل در بی درنگ بودن وقفه برتری بر نظرسنجی دارد.

سوال5.

https://www.geeksforgeeks.org/system-bus-design/

https://www.tutorialspoint.com/what-are-the-elements-of-bus-design-in-computer-architecture

الف)

در کامپیوتر،دستگاه ها با سرعت های متفاوتی با داده کار می کنند(به طور مثال سرعت پردازنده بسیار زیادتر از سرعت حافظه است)، گذرگاه وظیفه دارد تا هنگام ارتباط این دستگاه ها، سرعت ها را باهم سازگار و هماهنگ کند. این معنی سازگاری سرعت است.

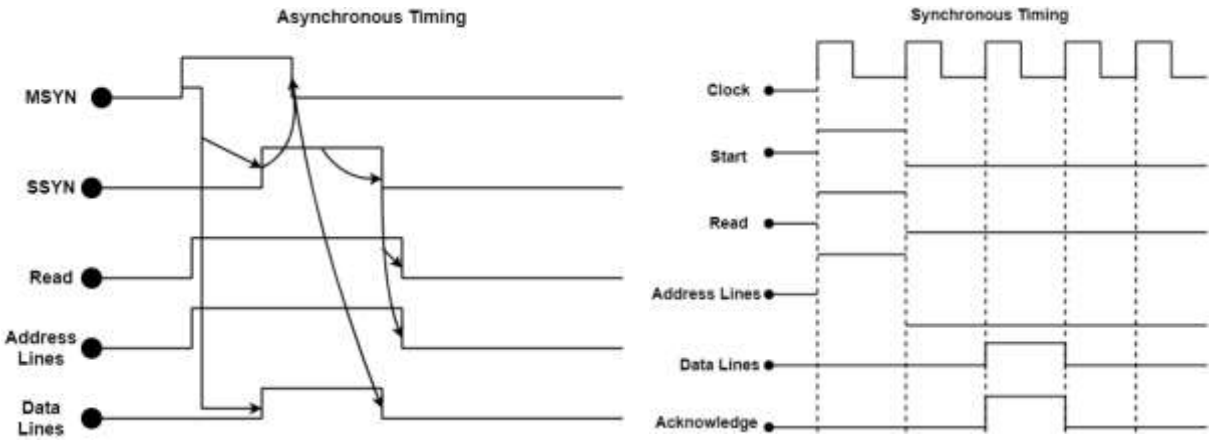
در یک کامپیوتر دستگاه ها ممکن است روش های متفاوتی برای نحوه خوانش یا نوشتن ذرات داده در کنار هم داشته باشند که هر یک پروتکل گوییم. وظیفه دیگر گذرگاه پشتیبانی از این پروتکل هاست تا دستگاه ها درک درستی از داده های ارسالی و دریافتی از یکدیگر داشته باشند. به این سازگاری پروتکل گوییم.

ب)

در گذرگاه های همگام تمامی ارسال ها و دریافت ها با کلاک هماهنگ است و در هرکلاک عملیات خود را انجام می دهند. نتیجه می شود که اتفاقات تمام به کلاک بستگی دارد و می توان سیگنال های دیگر را بر اساس کلاک ترسیم کرد. در صورت کندتر بودن انتقال داده در گذرگاه نسبت به کلاک بین دستگاه ها عدم هماهنگی رخ می دهد و عملیات آسیب می بیند(clock skew)

از آنسو در گذرگاه های ناهمگام هر دستگاه با کلاک درونی خود کار می کند و بنابراین هرگاه نیاز به ارتباط باشد خودش باید با سیگنال هایی به دستگاه مقابل اطلاع دهد. در این روش اتفاقات و بقیه سیگنال ها بیشتر متاثر از سیگنال وارده قبل از خود است. این روش مشکلات هماهنگی پیش می آورد مخصوصا زمانی که دو دستگاه همزمان بخواهند ارتباط برقرار کنند.

برای درک بهتر تفاوت عملکرد می توانید به دو تصویر زیر بنگرید، این دو نمودار برای تفاوت نحوه Timingدر گذرگاه ها هستند:



ج)

طبق تعریفی که در کلاس داشتیم گذرگاه درواقع مجموعه سیم ها و مدار هایی است که وظیفه ارتباط دستگاه هارا متقبل می شود، طول گذرگاه در واقع مربوط به طول فیزیکی سیم هاست که هرچه قدر بیشتر باشد پیام ها زمان بیشتری نیاز دارند.

در آنسو هرچه تعداد وسائل متصله بیشتر شود گذرگاه وظایف بیشتری را برای هماهنگی و انتقال داده بین آنها متحمل می شود.
همینطور با درخواست های متعدد دستگاه(به طور مثال وقفه) گذرگاه باز هم شلوغ تر می شود و ترافیک داده های بیشتر می شود.
نتیجتا با افزایش دستگاه های متصله سرعت آن کاهش پیدا می کند.

د)

همانطور که در ب اشاره کردیم در گذرگاه های همگام ممکن است به دلیل تفاوت های فیزیکی دستگاه ها زمان رسیدن سیگنال کلاک به یک ممکن است کمی تفاوت داشته باشد که به این تفاوت clock skewگوییم

منابع: https://ieeexplore.ieee.org/document/8993375/figures#figures

Johnson, H. & Graham, M. (2003). "High-Speed Signal Propagation: Advanced Black Magic."

Weste, N. H. E., & Harris, D. (2010). "CMOS VLSI Design: A Circuits and Systems Perspective."

یکی از روش ها پیش بینی clock skewاست اینگونه می توان به صورت دستی به دستگاهی که دیرتر سیگنال کلاک به آن می رسد کلاک دیگری با همان فرکانس داد یا کاری کرد که هنگام فرستادن ارسال، سیگنال زودتر به این دستگاه راهی شود. راه خوب دیگر قوی تر کردن منبع سیگنال است تا سرعت ها زیاد شوند و هماهنگی بیشتر شود. راه دیگر کمتر کردن سیم های مسی است تا باز سرعت ها نزدیک تر شود. یک راه جالب دیگر که برعکس راه های قبلیست استفاده از بافر ها برای تاخیر رسیدن سیگنال به دستگاه نزدیک تر به کلاک است. بدیهتا یکسان سازی کلاک ها و قرار دادن یک کلاک کلی هم در نظر گرفتیم اجرا شده، در غیر این صورت این هم جزء راه های خوب است.

ه)

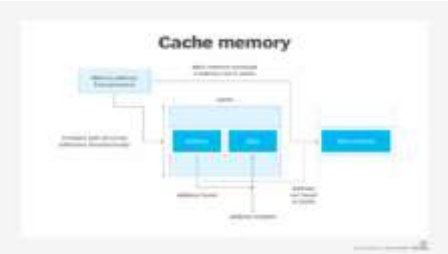
این سیستمی است که با ارائه چند مسیر ارتباطی به ارتباط دستگاه ها کمک می کند. باعث می شود تا زمان انتظار کمتر شود و همینطور سرعت انتقال داده بیشتر شود. عنصر اصلی موفقیت در این سیستم کاهش تعداد چرخه ها برای اجرای عملیات هاست، در گذشته اگر فرضا 3 بار باید 3سیگنال فرستاده می شد الان همه همزمان می توانند ارسال شوند. با افزایش تعداد راه های ارسال داده بین دستگاه ها اتصال آن ها بهبود می یابد و حجم داده ای که می توان ارسال کرد بیشتر می شود. نقش دیگری که می تواند ایفا کند به اصطلاح راه جایگزین است! اگر مثلا ارتباط از یک گذرگاه مختل شود کل عملیات مختل نمی شود بلکه می توان از گذرگاه دیگر استفاده کرد.

6.

الف و ب)

به دلیل مشابه بودن علت دو بخش الف و ب را یکجا توضیح می دهم:

cache عامل این اتفاق است. هرگاه که پردازنده نیاز به خانه ای از حافظه داشته باشد، به سراغ cache می رود. اگرآن خانه در cache بود بر می گرداند و اگر نه cacheبه سراغ حافظه اصلی می رود. مسئله اختلاف سرعت این دستور ها در این است که cache به قصد کاهش زمان میانگین مورد نیاز برای دسترسی طراحی شده، وقتی یکبار خانه ای را از آن می خوانیم اینگونه رفتار نمی کند که فقط همان را در خود قرار دهد، بلکه بلوک حافظه ای که آن آدرس در آن بوده را کلا در خودش می ریزد، چرا که از نظر زمان مورد نیاز فرق زیادی با لود کردن همان یک کلمه ندارد و بسیاری از مواقع هنگامی که خانه ای را برنامه می خواند، خانه های بعد یا قبل آن را نیز در ادامه فرامی خواند و به طور میانگین اینکار باعث بهبود سرعت می شود. در کد داده شده در خط اول آدرسی را می خواهد که در cache نبوده پس بار اول زمان طولانی تری صرف شده تا اطلاعات را لود کند اما در دوخط بعد همان خانه و خانه ای که 1 کلمه فاصله دارند خواسته شده که در بلوک حول همان خانه اول در کش لود شده اند. در آنسو خانه ای که در خط چهارم خواسته شده فاصله زیادی با آدرس اول دارد و در بلوک آن قرار نگرفته، پس حالا که در کش موجود نیست cache مجبور است مانند بار اول به حافظه رجوع کند و آن خانه به همراه بلوک حول آن را لود کند، و اطلاعات خواسته شده را به پردازنده بدهد.



ج)

Memory mapped IO، زیرا از دستورات اضافه و اختصاصی برای دستگاه ها استفاده نکرده، به این معنی است که در ISAبرای کار با دستگاه های ورودی خروجی دستور اختصاصی نداشتیم و باید شماره آدرس جایی که در حافظه به عنوان محل ذخیره سازی ورودی ها وخروجی ها درنظر گرفتیم را بدهیم (با دستورات lw و sw با دستگاه ورودی خروجی کار می کنیم) همچنین مشاهده می شود که دسترسی به این آدرس هر دفعه مقدار مشابهی زمان مصرف می کند که نشان می دهد ربطی به حافظه نداشته و احتمالا دستگاهی سخت افزاری است.

7.

الف)

منطقا اولین چیزی که به ذهن می رسد افزایش تعداد دیسک هاست.

روش های متنوعی برای تقسیم بندی دیسک ها (disk striping) وجود دارد که برای افزایش توان عملیاتی به نظر raid 0 از بقیه مناسب تر و بدیهی تر است(احتمالا منظور سوال هم همین روش بوده). Raidهای مختف دیگر را به طور خلاصه در تصویر سمت راست می توانید مشاهده کنید.

روش disk striping بخش raid 0 به اینگونه است که اطلاعات را بین چند دیسک تقسیم می کنیم و هنگام فراخوانی یک خانه هریک بخشی از آن را دارند و همزمان می فرستند، این دسترسی موازی به اطلاعات توانایی عملیاتی ما را بیشتر می کند. برای افزایش توان عملیاتی از 200 به 800 از 4 دیسک که هرکدام توان عملیاتی 200مگ بر ثانیه دارند در قالب stripe استفاده می کنیم.

ب)

بله

طبیعتا هنگامی داده ها را بین آنها تقسیم می کنیم این ریسک وجود دارد که حتی اگر یکی از آن ها تاخیر بیشتری داشته باشد تمام فرایند تاخیر می خورد و اطلاعات ارسال شده از دیسک های دیگر تا رسیدن اطلاعات همه دیسک ها بلافایده است. همچنین اگر یکی از دیسک ها به هردلیل از دسترس خارج شود اطلاعات کل مجموعه دیگر قابل استفاده نیست زیرا بدون داشتن اطلاعات دیسک خارج شده از دور بقیه اطلاعات بی معنی می شوند.

ج)

می توان از همان روشی که برای افزایش سرعت دسترسی پردازنده به حافظه اصلی استفاده می شود، استفاده کرد. برای پیاده سازی روش caching ما در بین مسیر دیسک ها تا دستگاه مقصد حافظه های نسبت به دیسک سریعتری مانند SRAM ها یا DRAM قرار می دهیم تا مانند cache برای main memoryعمل کنند، البته با توجه به اختلاف زیاد SRAM و دیسک شاید استفاده از DRAMگزینه بهتری باشد. هرگاه که دستگاه نیاز به اطلاعات داشت به این پردازنده ها رجوع می کند و عملا فرایند مشابه cache ایجاد می شود: اگر در حافظه های سریع بین راه ما نبود آن خانه را از دیسک ها پیدا کرده و کل بلوک حول آن را در خود می ریزد ضمن ارسال خانه مورد نظر به دستگاه. اینگونه به طور میانگین سرعت دسترسی ما افزایش می یابد.

د)

طبیعتا استفاده از روش های raid 1 و raid 5گزینه های از پیش آماده و مناسبی است. نسخه های بعد raid1 آن را بهینه تر و اقتصادی می کنند برای همین به اختصار همین را توضیح می دهیم. در این روش به ازای هر دیسک دیسک دیگری را به عنوان آینه قرار می دهیم تا با mirroringو داشتن بک آپ هرگاه یکی خراب شد به سراغ دیگری برویم. برای فهمیدن اینکه کدام خراب شده هم در سیستم های raid از بیت های parity و رصد هوشمند (smart monitoring) استفاده می شود. طبیعتا نیاز می بینیم که هردفعه بجای نوشتن بر یک دیسک بر خودش و دیسکmirrorبنویسیم که زمان و انرژی بیشتری از ما می گیرد و کارایی(performance) را کم می کند، هرچند اطمینان ما را از ذخیره شدن داده به طور صحیح بیشتر می کند. توجه داشته باشیم که بیاد فضای بیشتری برای همان اطلاعات قبلی و همینطور باید بیت های parityرا برای تشخیص خطا به داده ها اختصاص دهیم که منجر به redundancy می شود.