Lecture 2

Lecture contents:

-Transactions

-Synchronization protocols

-parallel vs serial communications

المحاضرة اللي فاتت اتكلمنا عن ال buses وقلنا ان ال bus دة هو الحاجة اللي بتوصل بين ال transmitter وال receiver

A bus is a shared connection between transmitter & receiver

وقلنا ان ال information اللي ال bus بيكون شايلها بتبقى دايماً حاجة من 3 :

-data

-address

-control signals

النهاردة بقى هنشوف ازاي ال information دي بتتنقل من على ال bus .. تعالى ناخد مثال الأول في حياتنا

لو احنا رايحين بنك وعايزين نسحب منه فلوس .. اول حاجة بنعملها اننا بندخل البنك، **نستنى دورنا** ، اول لما دورنا ييجي بنبدأ نقول للبنك **احنا عايزين نسحب من أنهي حساب بالظبط** فبنديله رقم الحساب .. بعد كدة بنبدأ **نسحب الفلوس**

اللي بيحصل في البنك دة بالظبط هو اللي بيحصل جوة البروسيسور .. ساعات بيكون فيه كذا ماستر عايز يتحكم في ال bus، فبدل ما يتخانقوا مع بعض على مين اللي هياخد ال bus الأول بيحصل حاجة اسمها arbitration بتقول مين اللي هيمسك ال bus دة، واللي بيحدد مين اللي هيمسك ال bus هو ال arbiter .. والتحديد بيكون على أساس معين وغالباً بيبقى على أساس ال priority

يعني لو ال cpu عايز ي access ال bus في نفس الوقت اللي ال LCD عايزة فيه ال bus ساعتها ال arbiter بيدي ال permission لل cpu الأول

** وزي ما قلنا ال arbitration دة مش بيحصل غير لو فيه كذا master على نفس ال bus .. انما لو مافيش غير ماستر واحد بس يبقى ال arbitration مالوش لازمة لأن الماستر مش هيلاقي حد يتخانق معاه

بعد ما ال arbitration بيحصل الماستر بيبدأ يبعت ال address بتاع ال slave اللي عايز يكلمه على ال bus .. وممكن الماستر يكلم slave واحد بس في المرة الواحدة ودة اسمه Uni-casting وممكن يكلم كذا slave في نفس الوقت ودة اسمه multi-casting وممكن يكلم كل ال slaves اللي على ال bus وساعتها يبقى اسمه broadcasting

بعد كدة بقى بيحصل ال actual data transfer .. اللي هو سحب او إيداع الفلوس في البنك

فعشان نلخص الكلام اللي فات دة:

Transaction is done in 3 phases in the case of multi masters on bus:

1- Arbitration
2- Addressing
3- Actual data transfer

And is done in just 2 phases in the case of single master:

1- Addressing
2- Actual data transfer

نيجي بقى نتكلم في حاجة تانية .. دلوقتي ال information بتتبعت على ال bus، بس ساعات مش بتبقى stable وغالبا بتبقى garbage فمالهاش لازمة .. فاحنا عايزين protocol يخلي ال information دي stable في الوقت اللي انا عايز اقراها فيه

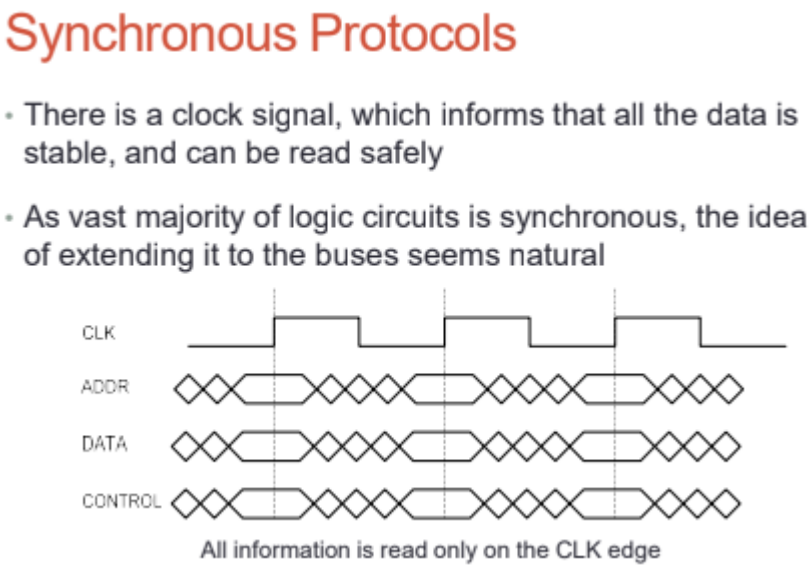
هو دة اللي اسمه synchronization .. بن synchronize ال transmitter وال receiver مع بعض عشان لما ال receiver ييجي يقرا من ال transmitter يعرف يقرا ال stable information اللي هو محتاجها

ففيه 3 أنواع من ال synchronization :

1- Synchronous
2- Semi-synchronous
3- Asynchronous

Synchronous:

الماستر والـ slave بيتفقوا انهم هيشتغلوا بـ clock معينة بحيث انها تقولي ان الـ info عند حتة معينة في الـ clock دي stable :



في الديزاين اللي فوق دة هنلاقي ان عند كل +ve edge للـ clk الـ address, data, control كلهم stable فال receiver يقدر يقراهم

بس خلي بالك ماينفعش الـ rec. يقرا الـ 3 مع بعض .. لازم يقرا الداتا بعد ما يقرا الـ address، وممكن يقرا الـ address والـ control مع بعض في نفس الـ clock cycle

بس ماينفعش يقرا الداتا مع الـ address والكونترول في نفس الـ clk cycle .. اصل ساعتها هو لسة مش هيبقى عارف الداتا دي هتتحط فين .. فهو لو قرا الداتا في نفس الوقت اللي بياخد فيه الـ add. & control هيحط الداتا دي فين؟؟!

فهنا الداتا هتتحط على الـ bus في الـ 2nd clock cycle مش في الـ 1st

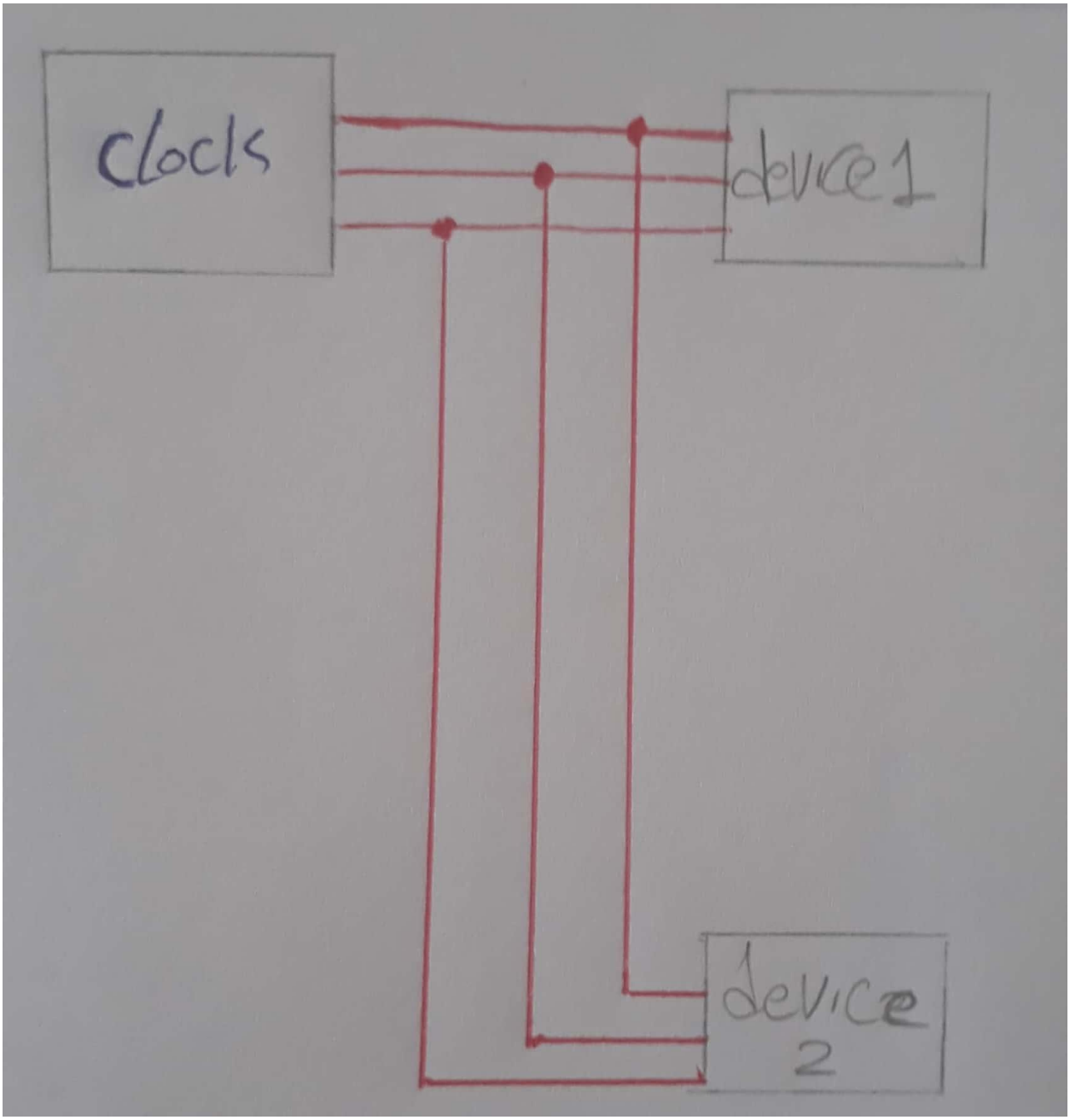
- في الديزاين دة هو اشتغل على ال +ve edge .. بس في ديزاين تاني انا ممكن اشتغل على ال – ve edge .. بس ساعتها لازم اعمل حسابي ان قبل ال –ve edge وعندها وبعدها تبقى ال information is stable

> Challenge: how to synchronize different devices operating at different frequencies?

عشان نشوف المشكلة دي تعالى نفكر لو عملنا clock division وكل component ياخد ال frequency اللي هو محتاجها .. بس ال division دة اكيد مش ideal فهيحصل shifting لل clock بمقدار معين .. ودة اسمه clock skew

طيب افرض احنا جامدين زحليقة وعرفنا (بطريقةٍ ما) اننا نظبط ال frequencies بتاعت كل ال components اللي معانا بحيث انهم كلهم يشتغلوا بنفس ال clock من غير division .. هل ساعتها هنبقى حلينا المشكلة؟؟

الإجابة لأ .. بص كدة على الديزاين دة:

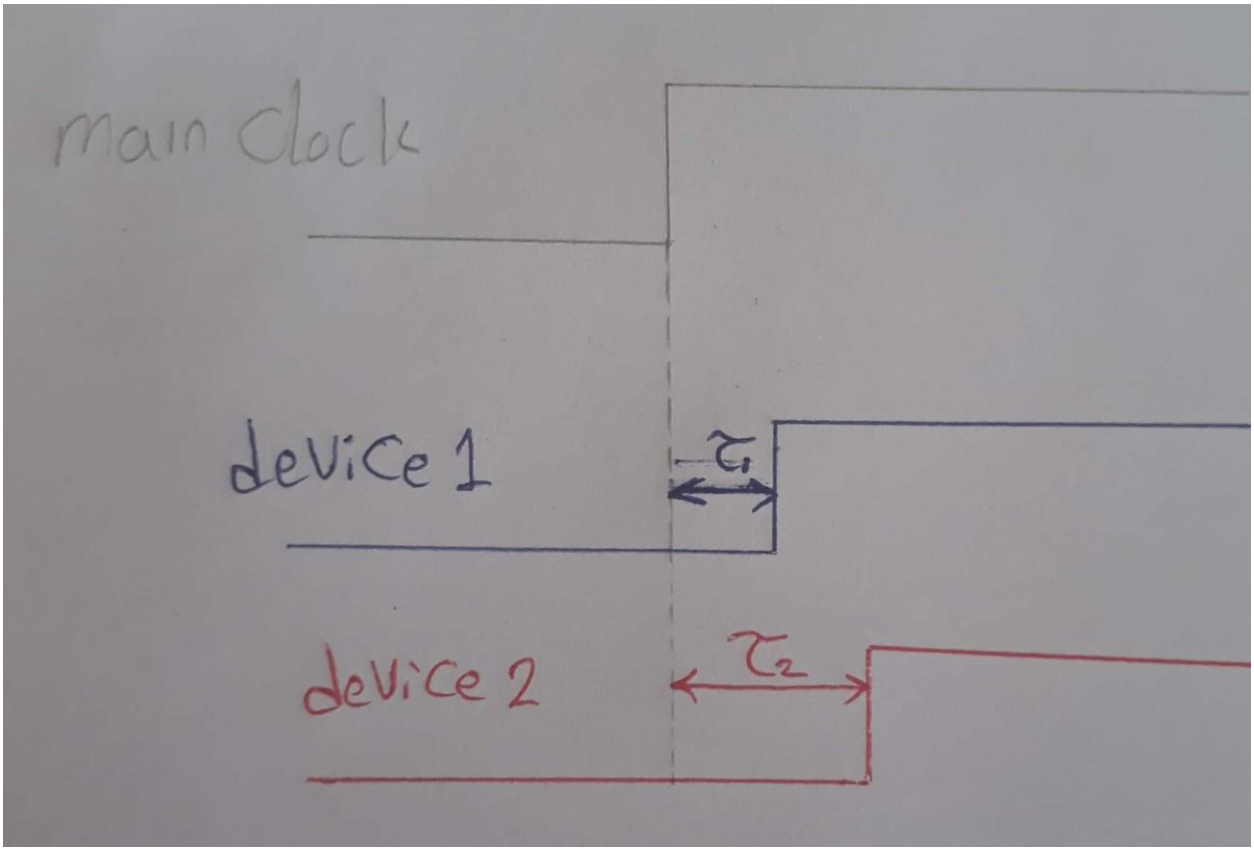


ال clk متوصلة مع device 1 &device 2 .. واحنا عارفين ان فيه حاجة اسمها propagation delay بتحصل و دي بتساوي المسافة اللي ال information ماشية فيها جوة ال medium على سرعتها في ال medium دة

$$T = d/s$$

فاكيد سرعة ال clk عند device 1 اكبر منها عند device 2 لأن device 1 اقرب لل clk من device 2 فبرضه هيحصل skew

فمن الآخر لازم يحصل clock skew وساعات بيبقى ال skew كبير فبيبقى ال delay ممكن ياخد 3 clock cycles مثلاً . وطبعا دي مشكلة



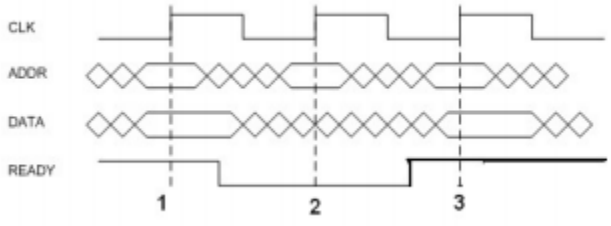
فيعني من الآخر صعب اوي اظبط ال synchronization بتاع كل ال components اللي عندي ب clock واحدة .. فحل المشكلة دي بييجي في ال semi-synchronous

Semi-synchronous:

هو نفس concept ال synchronous بالظبط بس الزيادة عنه ان ممكن كذا slave بسرعات مختلفة يتكلموا مع الماستر بنفس ال clock .. ولو ال slave بطئ بيقدر يقول للماستر يهدى عليه شوية عن طريق signal اسمها ready او wait .. ف طول ما ال slave رافع ال ready يبقى هو شغال مع الماستر .. اول ما ينزل السيجنال دي الماستر هيعرف ان ال slave مش عايز أي infoدلوقتي فهيستنى لحد لما ال slave يرفع ال ready تاني عشان يكمل

Semi-synchronous Protocols

- There is a clocking signal synchronously to which a request from master occur.
- The response from slave is indicated on dedicated line (i.e. READY or WAIT), whose value is sampled by the clock signal.



Asynchronous:

هنا مافيش clock .. فالماستر وال slave بيتفقوا مع بعض على طريقة تانية غير ال clk يفهموا بيها بعض، والطريقة دي بتختلف على حسب ال protocol

في ال example اللي جي دة المفروض الماستر هو اللي هيقرا من ال slave

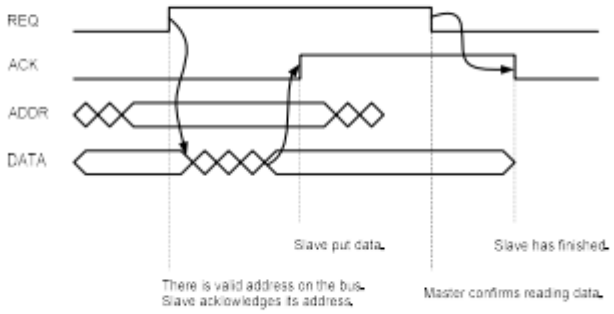
فالماستر هيبعت request انه يتكلم وهيبعت معاه ال address اللي عايز يكلمه وهيستنى لحد لما ال slave اللي الماستر عايز يكلمه يشوف ال address بتاعه .. لما هيشوفه هيبعت للماستر ويقوله تمام .. تمام دي اللي هي ال Acknowledge فهيرفع ال ack signal .. وهيبعت الداتا

اول لما الماستر يشوف ال ack هيبدأ يقرا الداتا لحد لما يقراها كلها .. دلوقتي عايز يقول لل slave انه خلاص قراها .. فهينزل ال request

ال slave لما يشوف كدة هيقوم هو كمان منزل ال ack وشكراً

Asynchronous Protocols

- There is no implicit time constraints – the asynchronous events are issued both by master and slave.



Is gaining popularity in on-chip communications as it does not suffer the constraints/delays imposed by the slowest signal.

دلوقتي بقى لو كان الماستر هو اللي بيكتب كان ايه الفرق اللي هيحصل؟؟

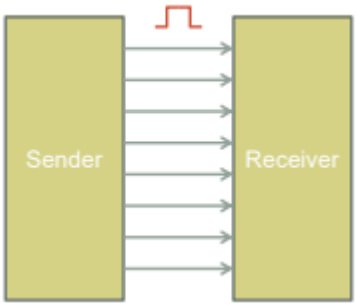
لو كان بعت الداتا بعد ال ack كدة يبقى هو كدة بيتأخر .. فهيبعت الداتا بعد ال request .. بس ال still مش هيقرا الداتا دي غير بعد لما يقرا ال address slave

فكدة بالنسبة لل components اللي كان بيحصل لها clock skew في ال synchronous ممكن دلوقتي تشتغل asynchronous .. بس لسة مشكلة ال asynch. في ال wiring

تعالى بقى نتكلم شوية عن ال serial & parallel buses

Serial and parallel buses:

البداية كانت بال parallel bus .. اللي هو السلك التخين اللي فيه اسلاك كتير اوي دة زي السلك بتاع الشاشة اللي في الكمبيوتر .. فالداتا بتتبعت على كل الاسلاك اللي جوة دي in parallel وفي نفس الوقت



وفيه برضه ال serial bus اللي زي ال usb اللي لما نقطعه هنلاقي فيه سلكة واحدة بس للداتا، لأن الداتا بتمشي ورا بعضها على نفس السلكة

في ال serial bus ال processor بيحتاج shift register عشان يقدر ياخد bit ورا bit .. بس ال parallel مش محتاج لأن كل الداتا اللي بتتحط في ال registers بتجيله مرة واحدة

وبرضه ميزة ال ال parallel عن ال serial هي ان ال parallel اسرع من ال serial .. ودة منطقي لأن احنا قلنا ان في ال parallel الداتا كلها بتتبعت مع بعضها في نفس الوقت (في نفس ال clock cycle)على عكس ال serial اللي الداتا بتتبعت فيه ورا بعضها فعشان ننقل 1 bit محتاجين 1 clock cycle

فلو قلنا اننا عايزين ننقل 8-bit data فاحنا في ال parallel محتاجين 1 clock cycle بس في ال serial محتاجين 8 clock cycles

بس الكلام دة لو كنا شغالين ب frequency مش كبيرة .. فلو ال frequency كبرت هيحصل skew .. ودة بييجي بسبب ال wiring اللي اكيد مش هتكون كل الاسلاك perfectly matched فال propagation delay هيختلف من pin للتانية

وهنلاقي ان بسبب ال mutual capacitances & inductances اللي بيزيد تأثيرها عند ال high frequencies بيحصل interference بين الداتا اللي في الأسلاك واللي اكيد هيعمل noise ودة بيبقى اسمه cross talk وبيبقى سبب كبير في ان ال throughput يقل

كمان بما اننا بنشتغل بأسلاك فأكيد الطول هيبقى parameter مهم معانا فهنحتاج نقصر طول السلك على اد مانقدر

فمن الآخر احنا بنييجي عند ال high frequencies ولما نيجي نقارن ال throughput بين ال serial & parallel عند ال frequency دي هنلاقي ان ال serial بقت هي الأسرع من ال parallel بسبب كل الكلام اللي فات دة .. ودي ال equation اللي بت limit ال frequency اللي المفروض نشتغل بيها

$$f_{CLK} = \frac{1}{T_{DMAX} + T_{SKEW}}$$

هنلاقي ان ال parameters بتاعتنا هي الوقت بتاع ابطأ signal بتمشي في السلك وال skew time

.. فمن الآخر بقى، الأحسن اننا نستخدم ال parallel في ال low frequencies ولو المسافة بين ال components صغيرة

ونستخدم ال serial في ال high frequencies ولو المسافة بين ال components كبيرة .. عشان كدة بنلاقي ان اغلب ال on-chip busses بتكون parallel بس ال off-chip buses بتكون serial

	Parallel	Serial
Distance	Short	Long
Frequency	Low	High

- So, the parallel buses are option of choice for on-chip buses, while in off-chip communication one often resorts to serial protocols.

هنتكلم بقى اكتر دلوقتي والمحاضرة الجاية عن ال parallel buses ..

Parallel busses:

فيه نوع مشهور جداً لل parallel busses بتعمله شركة ARM اسمه AMBA

Advanced

Microcontroller

Bus

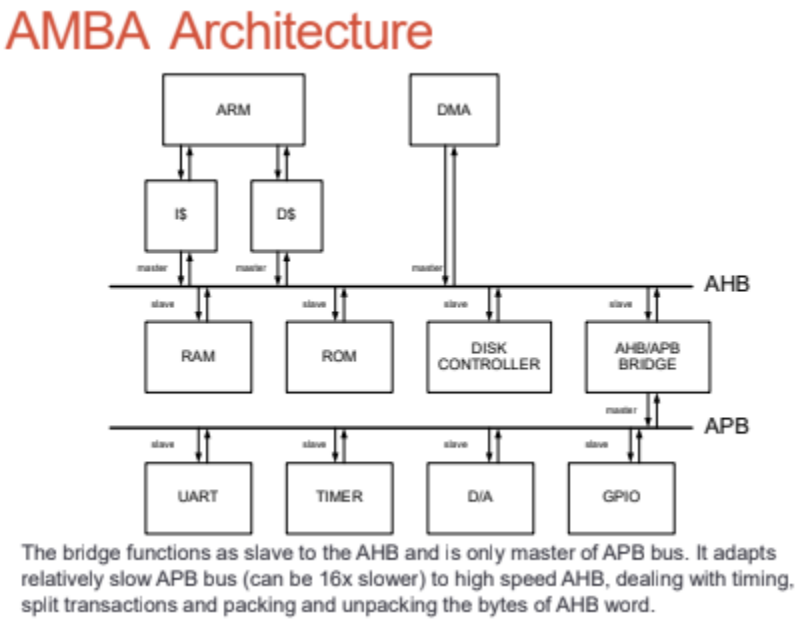
Architecture

ال bus دة هو ال on chip bus اللي بيوصل ال information اللي بتحصل بين ال peripherals المختلفة اللي جوة ال ARM core µC's

احنا هنتعرض لشكلين بس في ال AMBA في الكورس دة :

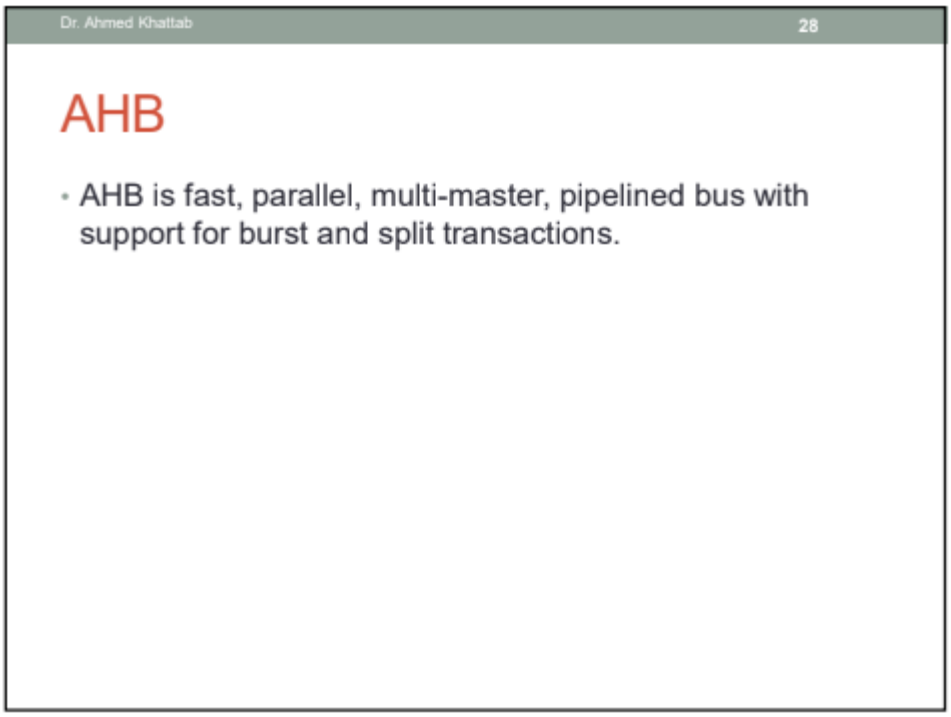
- **AHB** – which stands for ARM High-performance Bus – it links fast peripherals providing high clocking frequency and large throughput. Has very complex hardware and is expensive to implement.
- **APB** – slow bus, very simple in comparison, cheap to implement in hardware

ودة شكل ل architecture فيه ال AHB & APB



هنلاحظ ان ال AHB عليه multi masters بس ال APB مش عليه غير single master بس ودة بيوضح ال complexity بتاعت ال AHB .. وبرضه هنلاقي ان ال bridge بيقدر يكلم AHB & APB وبالتالي بيقدر ي adapt نفسه على سرعة كل واحد فيهم

.. الدكتور قال ان السلايد الجاية دي مهمة جداً برغم ان مافيهاش غير جملة واحدة فيعني خذوا حذركم O.o



Split transaction دي بتحصل في حالة زي مثلاً فيه حاجة واخدة high priority فهي اللي بتكونترول ال bus دلوقتي، بس هي بتعمل حاجة وقتها كبير ومعطلة باقي ال components .. فال high priority component دي بتقول خلاص انا هدي ال bus لحد تاني محتاجه .. فبتقسّم او بت split ال transaction بتاعها على كذا مرحلة.

وال burst transaction دة هنتكلم فيه المحاضرة الجاية ان شاء الله

وال pipelining بقى بتاع دكتور عمر نصر .. في اول clk cycle هيتعمل arbitration و T_1 هتكسب .. ففي ال cycle اللي بعدها هيبدأ ال addressing بتاع T_1 وفي نفس الوقت هيبدأ ال arbitration اللي T_2 هتكسب فيه وهكذا :

Pipelining

- Pipelined bus can perform each of three transaction phases simultaneously