Lecture 03

Lecture contents:

1-Burst transaction

2-Split transaction

3-APB

"Slides 30:37"

المحاضرة اللي فاتت كنا اتكلمنا عن ال AMBA ودة بنشوفه كتير اوي في ال 8-bit microcontrollers

واتكلمنا فيها عن ال AHB وال APB وقلنا ان ال APB دة بيبقى ابطأ بكتير من ال AHB، وممكن يبقى ابطأ منه ب 12 مرة او حتى 16 مرة

دلوقتي هنتكلم عن ال burst & split transactions .. ودول مش موجودين غير في ال AHB

المحاضرة دي مهمة جداً والدكتور قال ان هيكون عليها load كبير في الميدترم

Burst Transaction:

Burst عامةً معناها مجموعة .. فمن اسمها كدة احنا بنستخدمها عشان نبعت حاجات كتير مع بعضها

لو فاكر المحاضرة اللي فاتت كنا بنقول ان ال transaction بيحصل على 3 phases اللي هم arbitration, addressing and data transmission

فاحنا في العادي قبل النهاردة كنا لو عايزين نبعت data من sender ل receiver كنا بنقول ان اول clk cycle هيحصل arbitration .. تاني clk هيحصل addressing وفي التالتة هنبعت **1 block of data**

فاحنا لو كنا عايزين نبعت على نفس ال receiver دة 100 blocks of data مثلاً .. كان هيحصل :

Arbitration – addressing – data – arbitration – addressing – data – arbitration - …

وهنفضل كدة لحد لما نبعت ال 100 block .. أو بمعنى أصح 100 word

في حين اننا ممكن نعمل burst transaction فهنبعت العدد اللي احنا عايزينه من ال words مع بعض وبكدة هنبقى زودنا ال efficiency وقللنا ال over head

Efficiency: نسبة الحاجات المهمة اللي بتحصل / الحاجات ال total

Overhead: نسبة الحاجات المش مهمة اللي بتحصل / الحاجات ال total

فلو انا ببعت 1 data at a time :

ال arbitration هياخد 1 clk وال addressing هياخد 1 clk وال data transmission هتاخد 1 clk ... احنا هنا المهم عندنا مين؟؟

ال data transmission هو اللي يهمنا، فلما نيجي نجيب ال efficiency هنقول :

Efficiency = (data transmission clock cycles) / total cycles = (1 / 3)

في حين ان ال overhead :

Overhead = (not important clk cycles) / total cycles = (2 / 3)

ومثلاً لو كنا عايزين نبعت 100 word يبقى في الحالة دي هنعمل كمان 100 arbitration + 100 addressing، فال efficiency هتبقى 1/3 دة غير ان دة كمان هياخد وقت و power كبير

بس تعالى نشوف لما نعمل كدة في ال burst transaction .. عايزين نبعت 100 words فهنبعت Just 1 arbitration, 1 address وبعد كدة هنبعت ال 100 words كلهم ورا بعض

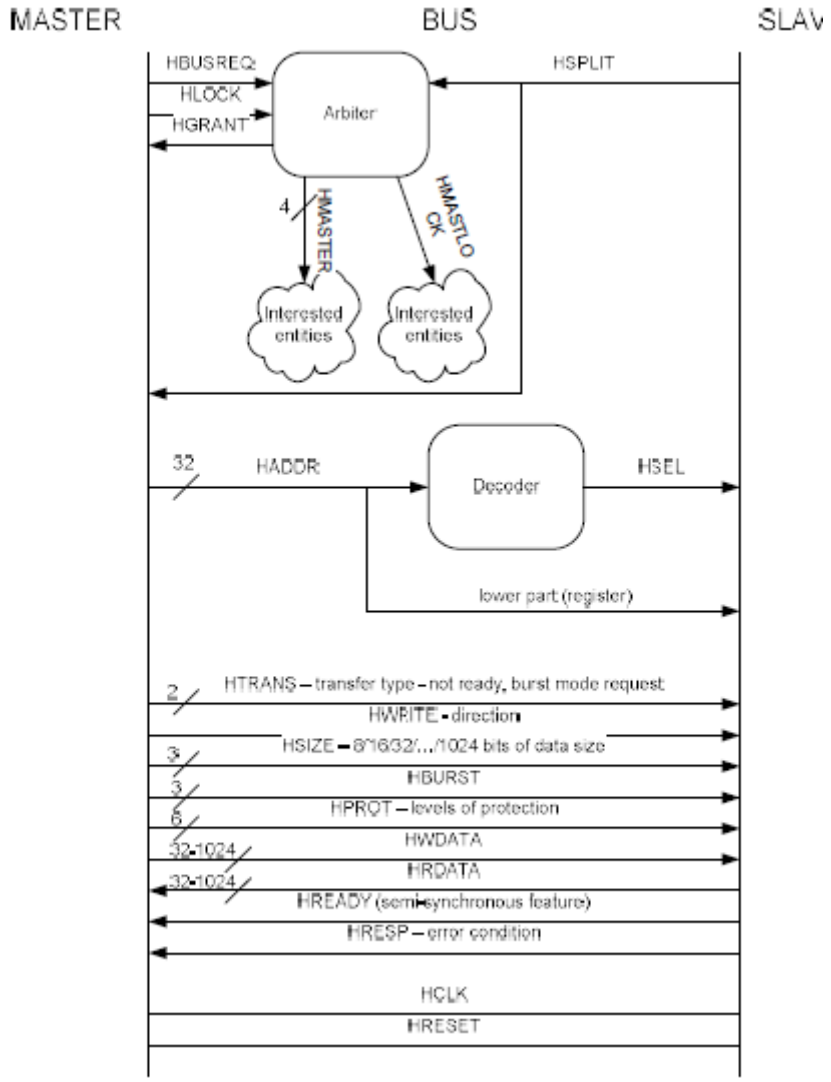
فكدة ال efficiency = 100 / 102 وطبعاً دي أحسن كتير

بس خلي بالك، بما اننا بنبعت arbitration و addressing مرة واحدة بس فمعنى كدة ان ال burst transaction مش بيتعمل غير لو نفس ال master بيكلم slave واحد بس في ال transaction .. وكمان ال addresses اللي الداتا بتتحط فيها جنب بعض (لأن زي ما هو واضح احنا مش بنحط 100 address .. لأ هو address واحد بس بيبقى هو اللي بيتحط فيه اول داتا وبعد كدة ال address بي increment او ي decrement على حسب احنا عايزين ايه)

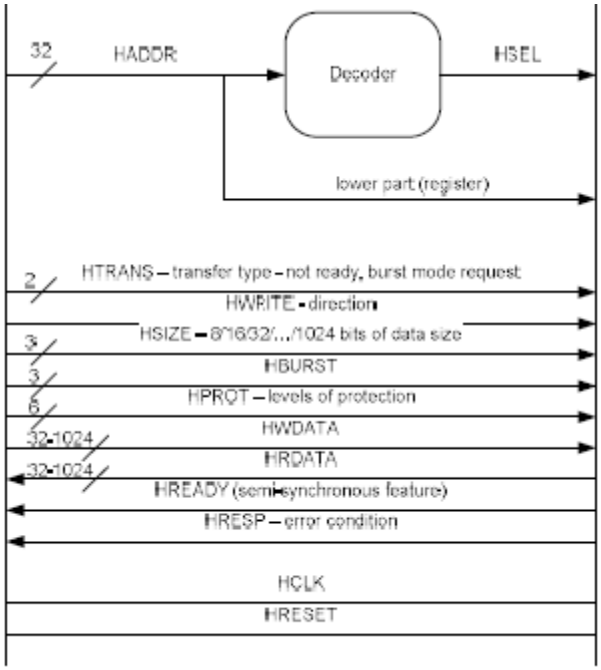
بس ثواني .. تفتكر ينفع نعمل burst transaction ب 100000 words مثلاً؟؟

الإجابة لأ .. لأن دة هيهنج ال CPU .. لأن مافيش غير الماستر دة بس اللي بي access الميموري لوحده ومش سايب حد تاني يشتغل .. متخيل المشكلة؟؟

تعالى بقى نشوف ال signals بتاعت ال burst transaction عاملة ازاي وبتعمل ايه :



الرسمة دي في slide 32 ممكن تشوفها لو مش باينة هنا .. احنا دلوقتي هنركز على الجزء اللي تحت منها (الجزء اللي فوق دة تبع ال split transaction اللي احنا لسة ماتكلمناش عنه) .. واحنا هن assume ان ال communication دة single master, single slave



تعالى بقى نشوف كل signal من دول بتعمل ايه ..

HCLK:

ال H دي يعني High performance .. معنى كدة ان دي ال clock بتاعت ال AHB ... وال clock بتاعت ال APB هيبقى اسمها PCLK

وال HCLK دي داخلة على كل ال devices المتوصلة مع ال AHB .. عشان كدة مش هتلاقي فيه سهم ناحية ال master او ال slave زي الباقي .. وال clock دي active high (يعني بت trigger عند ال +ve edge)

وعادةً ال HCLK دي بتكون من 20 MHZ وانت طالع

افرض بقى ان عندنا 2 clock domains (يعني فيه devices شغالة بسرعة معينة على clk معينة و devices تانية شغالة على سرعة تانية) .. تفتكر هينفع ن handle حاجة زي كدة؟؟

.. اه هينفع، ودي ميزة في ال AMBA .. ان ال standard بتاع ال AMBA سامح لنا اننا نعمل bus splitting لكل clock من اللي عندنا .. يعني ال AHB هيتقسم على 2 buses .. ودة هيحصل عن طريق bridge معين اسمه **AHB 2 AHB HUB**

والكلام دة طبعاً لو احتاجناه (يعني هو مش موجود طول الوقت وفي كل ديزاين)

HRESET:

نفس الكلام وبرضه هتلاقيها متوصلة مع كل ال devices بس هي active low .. فعشان نعمل reset للسيستم كله بنخلي ال signal دي low لمدة معينة (ال standard ان المدة دي 16 clk cycles .. ممكن الوقت دة يتغير بتغيير الديزاين فالفيصل في الموضوع دة هيبقى موجود في الداتاشيت)

HADDR:

دة بيبقى 32 signals وبيعبر عن ال address اللي الماستر عايز يبعت فيه الداتا .. هنلاحظ ان فيه جزء من ال signal بيدخل على decoder وجزء تاني تحت اسمه low part .. ليه دة حصل؟؟

اللي حصل دة عشان نقسم ال address، احنا عارفين طبعاً ان ال address دة عبارة عن segment, offset ... فباخد منه عدد معين من ال bits (بيختلف من ديزاين للتاني) وندخله على ال decoder عشان ي active device معين من اللي عايزين نكلمهم .. وهو دة ال segment

طيب ال segment اللي ال address بيشاور عليه دة بيعرف منين انه بقى activated ؟؟

عن طريق ال HSEL اللي خارجة من ال decoder

.. تمام احنا كدة خلاص عرفنا مين ال device اللي هيكلم الماستر .. عايزين بقى نعرف ال address او ال register اللي احنا عايزين نتكلم معاه جوة ال device دة .. هو دة بقى ال low part او ال offset

.. خلي بالك برضه ميزة تانية في ال AMBA ان ال arbiter يقدر هو كمان يشوف ال address .. طب ودة فايدته ايه؟؟!

فايدته انه بيشوف ال desired device دة مين (عن طريق ال address) وعلى أساسه ممكن يدي priority معينة لل device دة ... فمن الاخر، ساعات ال arbiter بيحدد ال priority على أساس ال address

HSIZE:

دة بيحدد ال Number of bits per word وبيبقى 3 signals .. وفي كل مرة ال master بيكلم فيها ال slave لازم يحدد له ال data size او ال word size .. **عشان كدة هي خارجة من الماستر لل slave**

HBURST:

فاكر ال HSIZE لما قلنا انه بيحدد ال **Number of bits per word** ؟؟

احنا محتاجين كمان نحدد لل slave احنا هنبعت له كام word بالظبط .. صح؟؟

فاللي بيحدد الكلام دة هو ال HBURST .. بيقولي انا هبعت كام data لل slave .. ودة له 5 values :

(1 word – 4 words – 8 words – 16 words – undefined)

كله واضح ماعدا ال undefined .. ودة بيعبر عن ال idle transaction اللي هنتكلم عنه حالاً

HBURST: words per burst

HSIZE: bits per word

HTRANS:

دي بتحدد لل slave نوع ال transaction وبما ان دي 2 bits فاحنا عندنا : 4 transaction modes

1- Idle transaction:

دة باختصار بيبقى transaction without data .. بعمل arbitration وببعت لل address slave وهو بيعتلي HREADY, HRESP .. ودة بنعمله عشان ن check ال device اللي بنكلمه دة available معانا ولا فيه مشكلة

2- Busy transaction:

فاكر ال semi-synchronous mode بتاع المحاضرة اللي فاتت؟؟ لما قلنا ان فيه ready, wait

كنا بنقول ان ال slave بيقول من خلالهم للماستر يهدى عليه شوية

ال busy transacrion دة بقى الماستر فيه هو اللي بيقول لل slave بالراحة عليا ..!

بس ثواني .. ازاي الماستر بيقول كدة برغم ان دايماً الماستر هو الأسرع من ال slave ؟!

-آه الماستر هو الأسرع، بس تخيل مثلاً في ال burst transaction الماستر بيجيله الداتا بسرعة كبيرة اوي .. هو طبعاً قادر ياخدهم، بس ساعات بيبقى عايز يعمل process عليهم (ينقلهم لمكان معين مثلاً او أي حاجة) ودي بتاخد وقت طبعاً، فلما يحس انه مش لاحق يعمل processing لل burst data اللي داخلة عليه بيبعت ال busy transaction

3- Non sequential transaction:

دة بيبقى transaction مش بيبقى part من حاجة قبلها (يعني مش بيبقى معتمد على حاجة قبله) .. زي مثلاً اول word بتتبعت من ال burst .. دي بيبقى مالهاش أي علاقة بالداتا اللي كانت بتتبعت قبل كدة

4- Sequential transaction:

دة بيبقى transaction معتمد على حاجة قبله .. زي باقي ال words في ال burst

HWRITE:

دة من اسمه بيتحكم في ال data direction .. لو ب 0 يبقى الماستر هي read

ولو ب 1 يبقى الماستر هي write

HPROT:

دي بتحدد ال properties بتاعت ال transaction او ال levels of protection ... الماستر بيبعت لل slave من خلالها شوية characteristics مش هندخل في تفاصيلها

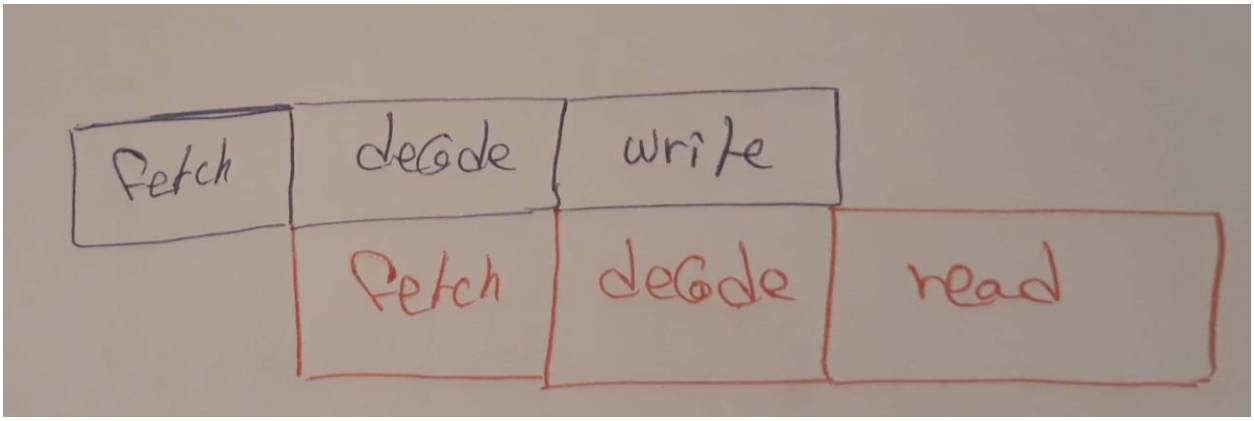
HWDATA, HRDATA:

دول هم ال data bus .. بس فيه سؤال وجيه .. هم ليه 2 separate buses برغم اني كدة كدة في أي حال من الأحوال مش هينفع غير اني يا إما اقرا يا إما اكتب على ال bus في ال 1 cycle؟؟

الإجابة ياسيدي إن دة له علاقة بال pipeline ..!

إفرض مثلاً إن فيه 2 slaves بيكلموا نفس ال master في ال pipeline .. واحد منهم بيكتب للماستر والتاني بيقرا منه

فهيبقى دة ال pipeline (مثلاً)



فالديزاين دة كدة بال 2 buses سمحوا لي إن ال read & write يحصلوا ورا بعض على طول .. على عكس ال bus الواحد اللي كان لازم يعمل delay بين ال read & write

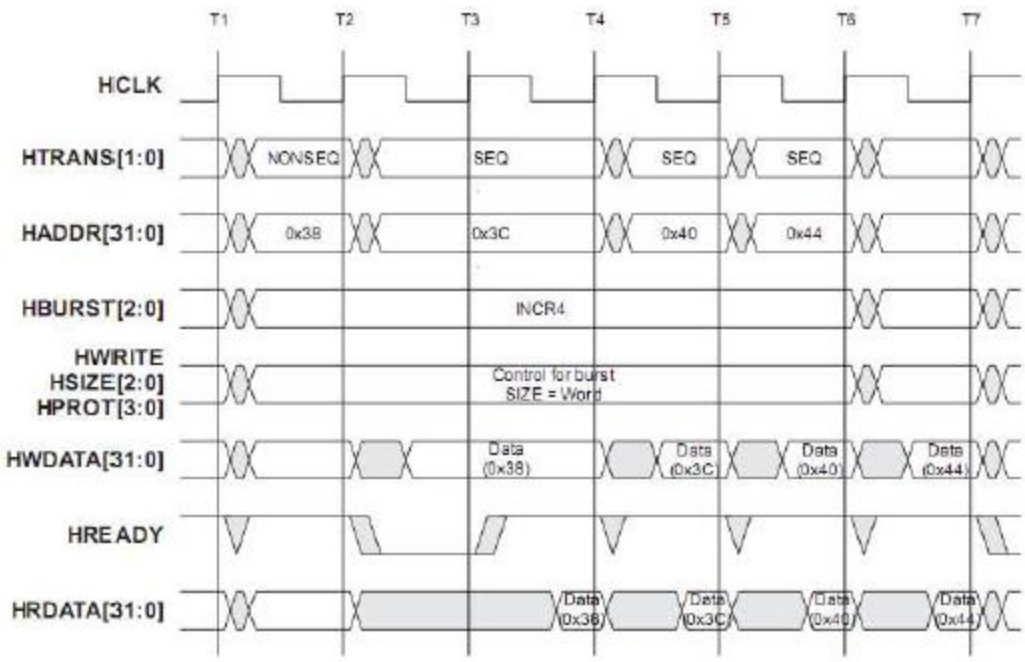
HRESP:

دة احنا شرحنا فايدته فوق في ال idle transaction وقلنا انه بيشوف لو فيه error معين حصل .. ودة طبعاً بيبقى من ال slave للماستر

HREADY:

دة يا كوتش بتاع المحاضرة اللي فاتت (semi synchronous)

كدة احنا خلصنا .. تعالى بقى نرجع ل slide 31 اللي الدكتور قال هجيب سؤال زيها بالظبط في الميدترم



لو بصيت عند T_1 هتلاقي لسة مافيش useful information تخلي ال transaction يبدأ عندها .. فهيبدأ من اول T_2

بص كدة على HTRANS ... في الأول هنلاقي non sequential وبعد كدة شوية sequential signals .. دة مش بيبين لك حاجة؟؟

بيبين لك ان بنسبة كبيرة اوي ال transaction دة هيبقى burst

بص بقى على ال HBURST هتلاقي ان ال signal بتقول لل slave انه 4 bytes (inc4) .. فكدة خلاص يامعلم دة اكيد burst

بص كدة على ال HADDR .. هتلاقي فيه غلطة: مكتوب 4 addresses مع ان في ال burst احنا اتفقنا اننا بنكتب 1 address بس وبعد كدة هو بي increment او ي decrement على حسب احنا عايزينه يعمل ايه

بس الدكتور بيقول اننا في الامتحان هنعمل زي ما هو موجود في ال slides برغم ان دة غلط ومش بيحصل في الحقيقة .. وخلي بالك هنا ال address بي increment by 4 .. تفتكر دة ليه؟؟

لو قلت انه عشان ال HBURST كان 4 bytes يبقى انت اتلخبطت .. لأن ال address incremental دة بيبقى على أساس ال HSIZE اللي بالصدفة كان في الديزاين دة 4 words، وعشان كدة ال address بيزيد ب 4 (مش عشان ال HBURST)

بص بقى على ال HWDATA .. المفروض انها هتروح لل slave في T_3 .. بس ال بص على ال HREADY هتلاقي ان ال slave كان not ready وقتها .. فالماستر هيضطر يأخر كل ال signals دي لحد لما ال slave يبقى ready ودة اللي حصل هنا بعد 1 clk cycle

عشان ال master ي write لل slave في T_4 مش T_3 ..

ولاحظ برضه ان الماستر مجهز ال write data من بدري .. على عكس ال read لأن في ال read ال slave بيتأخر في ال responseعن ال master، عشان كدة اول ما ال slave يبعت على طول الماستر بيقرا

.. خلي بالك كويس ان ال read & write مستحيل يخلصوا مع بعض زي ما هو مرسوم في ال slide (HWDATA & HRDATA) .. بس هو هنا الدكتور بيقول انه معتبر اننا فاهمين كدة فحط الاتنين مع بعض على أساس انهم 2 examples يعني مش 1 example

كدة احنا خلصنا ال Burst transaction

Split Transaction:

تخيل ان فيه peripheral بطئ ماسك ال bus دلوقتي بس لسة بيعمل processing على الداتا اللي معاه قبل ما يطلعها على ال bus .. فهو دلوقتي مش محتاج ال bus في حاجة فبيعمل request issue عشان ال arbiter يشوف لو فيه ماستر تاني كان مستني وعايز يمسك ال bus لحد لما الماستر الأولاني يخلص ويبقى جاهز انه يمسك ال bus .. ساعتها الماستر التاني هيوقف شغله ويرجّع ال bus للماستر الأولاني لحد لما يخلص شغله خالص وبعدين التاني يكمل اللي هو وقف عنده

هو دة ال split transaction باختصار

تعالى بقى نشوف ايه اللي بيحصل بالترتيب:

1-الماستر بي initiate ال communication وبيبدأ يعمل ال transaction

2-لو ال slave اللي الماستر بيكلمه مش جاهز انه يطلع الداتا على ال bus .. ساعتها بيحصل split request issue وال slave بيفضل فاكر مين الماستر اللي كان ماسك ال bus

3- ال arbiter بيشوف باقي الماسترز اللي عايزين يسمكوا ال bus وبيديه لل higher priority

4-بعد كدة لما ال slave اللي ماكانش جاهز دة يبقى جاهز بيبعت HSPLIT لل arbiter عشان يقول انه جاهز ان الماستر الأولاني يرجع يمسك ال bus

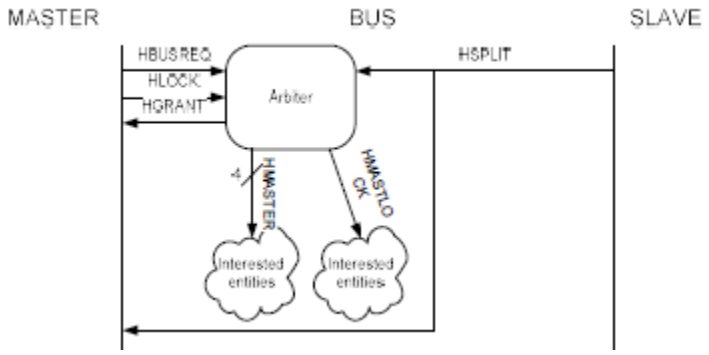
5-ال arbiter بيرجّع ال bus لل master الأولاني خالص

"شرح ال 5 خطوات دول انا مش متأكد منه فلو حد لقى فيه حاجة غلط يقول" .. ودة شرح ال slides :

Stages of Split Transaction

1. Master initiates transaction as usual.
2. If the slave is not ready, it asserts split and remembers the active master (provided by arbiter to anyone interested).
3. The arbiter grants the bus to other masters.
4. Slave asserts HSPLIT line to the arbiter, telling which master can resume.
5. Arbiter restores bus grant to interrupted master.

نرجع بقى ل slide 32 نشوف الجزء اللي فوق اللي احنا سيبناه



احنا شرحنا ال HSPLIT وقلنا ان ال slave بيبعته لل arbiter .. لو ب 0 يبقى ال slave بيقول لل arbiter يعمل split لأنه مش جاهز لسة ، وأول ما تبقى ب 1 يبقى كدة ال slave خلاص جاهز عشان يرجع ال bus للماستر الأولاني

HLOCK:

دي ال master بيبعتها لل arbiter عشان يقوله ان ال transaction اللي شغال دة ماينفعش يتقطع ولازم يكمل للآخر مهما حصل (ال split ممنوع)

HGRANT:

دي signal ال arbiter هو اللي بيبعتها للماستر عشان يقوله ان خلاص ال Bus رجعله تاني

HBUSREQ:

دي signal بتخرج من كل master عايز يمسك ال bus عشان ال arbiter يبقى عارف مين اللي داخل ال arbitration ويحدد مين يكسب على أساس ال priority

HMASTER:

دي signal بتخرج من ال arbiter على كل ال interested components عشان تقولهم ال ID بتاع الماستر اللي الدور جي عليه في ال bus

HMASTLOCK:

لو حد عارفها يقول ياجماعة :D

بس يعني تقريبا كدة والله اعلم هي بتقول لكل ال interested components ان الماستر اللي شغال دلوقتي دة locked ولا لأ .. الكلام دة فتي فلو غلط حد يقول

كدة احنا خلصنا ال AHB

نبدأ بقى في ال APB

APB:

دة single master بس

وهنا ال data size دايماً 8 bits .. مش زي ال AHB اللي كان فيه سماحية كبيرة لل data size

واللي بيقدر يكافئ ال data size دي بين ال AHB & APB هو ال HUB

ال read&write في slides 36 & 37 هنلاقيهم زي بعض تقريبا وهنلاحظ ان هنا عدد الأسلاك أقل من ال AHB لأن دة ابسط وأبطأ

وال PSEL دي بتخرج من الماستر عشان تعرّف ال desired slave انه هو اللي هيتكلم

وال PENABLE دي بتاعت ال master وطول ما هي بصفر يبقى الماستر لا هي read ولا هي write

وهنلاحظ ان ال address وال write data اتكتبوا مع بعض في نفس الوقت تقريبا في حين ان في ال read data كانت متأخرة عن ال address