**Parallel Computer Architecture**
**Prof. Hemangee K. Kapoor**
**Department of Computer Science and Engineering**
**Indian Institute of Technology Guwahati**
**Week - 06**
**Lecture - 36**

Lec 36: Phases in Split Transaction Bus

Hello everyone. We are doing module 4 on snoop based multiprocessor design. This is lecture number 7 and we are continuing the discussion on split transaction bus. In this lecture we are going to look at different phases of the split transaction bus. Okay. So when I split the transaction into two parts, I have a request phase and a response phase. So two things to do and they happen independently in time.

So to do these two things independently, I have to manage the resources that is how do I do the bus design. First is for the sending the request, I need a bus for which I will ask for permission. So that is the request bus and on this request bus we are going to give the address and the command, what has to be done. Then when this address and command goes on to the bus, there will be other processors and cache controllers that is the snoopers sitting there as well as the memory who will see this request passing through and they have to take appropriate action.

Subsequently in future a response will be generated. Responses are mainly acknowledgments and data transfer. So you need a separate bus for sending the response because your request bus could be at this moment being used by somebody else. Okay. So you need another set of wires to send your response. Okay. And then the responses are out of order. So if the response is out of order, you need to have a matching ID or a tag associated with the request because when you send a request the response comes much later.

So the response should carry information about which request it corresponds to. So how do we do this? It depends on the number of pending requests. If I can handle 8 pending requests, then I need 3 bit information to identify the response and the request matching. So all these information the bus has to handle. So the bus is now divided into an address bus which is the blue color, the command bus which is the green line, then the data response. Okay.

So we need this and in addition, you also need a 3 bit tag because you have to use this to match the request with the response and then the snoop results. Okay. So this is the

complete structure of my bus. Initially we had clubbed the address and the data together and we were worried only about the snoop values, there was no tag field but now we have this much infrastructure to handle. Okay. Then the data response phase, it depends on what is the size of the block we are transferring and therefore it is architecture specific. Because we are discussing SGI challenge and if I say my blocks are 128 bytes which is 1024 bits. So I need to transfer 1024 bits on the bus and my bus is 256 bit wide.

So you are going to take 4 trips to do this work. Okay. So over 4 clock cycles you are going to send 1024 bits. Now when we send these 4 cycles, the 5th cycle cannot be immediately used for the next data but it has to be left to vacant or blank as a turnaround time. So the technology requires a single cycle turnaround time. Hence I am going to need 4 for data transfer, 1 for turnaround making it 5 cycles of a data response. Okay.

So this is my requirement for the response and because response is taking 5 cycles, I have to make sure that can I fit my request also in 5. Okay. So that if everything is taking same time, we can have a pipeline behavior established. So we will see this as an example in detail. Okay. So what are we going to do? We are going to see how the address request is generated. Then we discussed the data response in the previous slide.

But before I send the data response, if you go back here, the 5 cycles of the data response are used only for transferring data but you cannot transfer data right away. You need to take permission to send the data, that is bus arbitration has to be required, then the receiver should be ready to accept the data. So all this homework has to be done before I can do this and that is done as part of the data request. So the response is actually combined with a request on the data bus followed by the data transfer. Okay. So these are the 3 things we have to look at. Okay.

So now coming back to the address request which is a 5 cycle phase. OKay. So what happens in this address request, that is what happens in those 5 cycles. The first cycle we are going to arbitrate for the bus and in the second cycle we may get the bus that is the bus grant. So when the bus grant comes, it comes with a tag assigned to this request because the bus controller knows what are the IDs of the pending request and it will give you a ID for your new request. Once you get the tag ID, you have to put the address onto the bus and the command.

So when you put the address and the command onto the bus in the third clock cycle, this travels onto the bus. Later, fourth cycle, this request is traveling, other caches read this request and they will now compare this address with their caches. Right. You have to do a bus loop. So you will compare your cache, you will take an appropriate decision that is should this cache reply with data, should this cache invalidate the block or should it

ignore the request. Okay. So all this processing is done in the decode that is the fourth cycle.

So end of fourth cycle, you would know that is every processor connected to the bus will know whether they should stay in the same state or they have to change the state. At the same time, they will also know whether they are supposed to supply the data or memory will give the data. Okay. So all this happens in the fourth cycle. In case, the some processor is busy doing processing of the cache and it is not able to finish its loop because it is very ambitious to say that you finished your answer in one clock cycle. If it is not able to do this, there is the fifth cycle.

So in the fifth cycle, it probably finishes or it can prolong the fifth cycle that is it will not send an acknowledgement onto the bus saying that I am taking more time, please wait, do not finish the request phase. Okay. So the fifth cycle is elongated beyond five cycles until all processors finish the snoop. Okay. So these are the five cycles: arbitrate, resolve, put the address, decode the address. During decode, all the snoop information is ready, state changes are recorded. They are not may or may not be done depends on the protocol, but you would know that should you shift from E to S or M to S and the fifth cycle is the ACK cycle. Okay.

So we can extend this if you have not finished the snoop. Okay. Then next is the data request and then the data response. Data response is sending the data, but before sending the data you have to get permission. So five cycles, first cycle arbitrate. Now here I am arbitrating for the data bus. Right.

So you are arbitrating for the data bus, you will get the bus. Second cycle, once you get the bus, what should you send? Here, third cycle we put the address of the block we are interested in and the command. In the data request, which is the address cycle, I am not going to put the address, but I am going to put the tag. Okay. So if the request went for address 5000, but the response will not come for address 5000, but the response will come for the message ID or message number 3. Okay. So pending request ID will be used here instead of the address.

And why do we do this? Why not the address? Because at, if you want to send the address, then the address lines here, the address bus is common, the address lines will now be in use by another request. Okay. So address lines are dedicated to request, hence you cannot reuse them for your data responses. So therefore, we are going to use the tag lines. In the third cycle, put the tag and once you put the tag, this tag reaches everybody and then the receiver, if receiver is ready to take the data, it can indicate so and therefore, we have one empty cycle here and once the receiver indicates that it is ready to receive,

it will start sending the data in the fifth clock cycle. Okay. So fifth cycle, we actually start sending data.

So the first 256 bits of your block/ Right. We had a block 1024, this was the block size had to be sent in four pieces. So the first piece is D0, D1, D2, D3. Right. So that is the amount of data I have to send and I have to keep one empty cycle here as a turnaround time. So among the data, the first chunk of 256 bits is sent in the fifth cycle of the data request. What about the others? These, now these three will be sent in the data response because response is only for sending the data and the bus is already acquired. Okay.

So because we have already sent D0, what we start from here is 1, 2 and 3, the first cycle D1, second cycle D2, third cycle D3. So you have finished sending all the four chunks and then you have a empty. Okay. See D1, D2, D3 and empty. So four cycles of the next phase are covered up by this.

So four cycles done. What happens in the fifth cycle? The fifth cycle, the D0 of the subsequent block can start moving because on the data bus. So the data request phase does not use the data bus, so it only uses the tag, data response uses the data bus. Okay. Along with this, what about the snoop? The snoop results which you have computed will now be conveyed as part of the response on the separate snoop lines. You have data bus lines and then snoop lines on which the snoop results will go. We now see an example of the split transaction bus as to exactly how the various phases get used.

So here I am assuming that processor P1 has the block and processor P2 wants to get that. So overall we want P1 processor to send the data to P2. So this is the operation which will happen eventually. All right. So because P2 wants the block, it is going to send a bus read request. So this read request will go during the first address request phase.

So it will start from here. So P2 sends the address request on the bus. So it first puts the request, it eventually gets a grant. So we are assuming it gets the bus grant. Once it gets the bus grant, it puts the address of the block B in the third clock cycle, okay. So you keep matching the clock cycles.

Address request, cycle 1, address request grant, cycle 2 and it puts the address in cycle number 3. So correspondingly these three actions take place. After that the fourth cycle is the decode cycle. So during this decode cycle, the address has already gone onto the bus and here P1 will actually see the block. That is it sees the address passing onto the bus and it realizes that yes it has the block and it should be providing the block.

So P1 realizes it has to take an action and then sends an acknowledgement on the bus. So during clock cycle 4, we decode it and the decode happens that is the address comparison happens with the address passing onto the bus with the address stored inside P1. So P1 does a cache tag comparison for that particular address. Once there is a cache hit and it realizes it is supposed to supply the data block, it sends an acknowledgement in cycle 5. In case P1 takes more time to do the tag comparison, it can extend the ACK cycle, that is it waits for one more clock cycle and then sends an acknowledgement.

So this way P1 can say that it is now prepared to give the data block to P2. So this ends the address request phase. So address request phase finishes here. What happens next? So next is actually P1 has to start giving the data. So data gets sent in the data request as well as data response phases.

So data request phase will start from cycle number 6. Okay. So P1 starts sending the data from cycle 6. So here we are in cycle 6. This is the position where P1 sends a request for the data bus.

It will get a grant. So we will assume that it gets the grant for the bus and once it gets the grant, it has to send the tag. So in the address phase that is the third clock cycle, it has to put the tag onto the bus. Why the tag? Because P2 sent the address and when P2 sent the address, a tag was associated with this particular request. So we can say that when the bus was granted, a tag was associated with this request and I am assuming , suppose the tag is a 3 bit number 100. So I am assuming tag 100 gets assigned to this particular transaction.

So P1 puts this tag here that is it puts the tag 100 in cycle 8. So when this tag goes onto the bus, P2 who is supposed to receive the data will check the tag. Because P2 wanted the data with the transaction ID 100. When 100 response is coming, P2 indicates that it is now ready to receive the data. Right. So here I would say that P2 shows willingness to receive the data. Right. If P2 is busy, then this transaction cannot happen.

So P2 shows willingness to accept the data and therefore, P1 can start sending the data from the ACK cycle that is the 10th cycle here. So because the target is ready, so we start sending the data from cycle number 10. Okay. So as you know the data goes in 4 pieces, so D0 goes in cycle 10 and then we use the remaining clock cycles for sending the other pieces of the data. So D1, D2, D3 go and then we have clock cycle 14 which is kept empty for the turnaround time.

So cycle 14 is empty for the turnaround time. What happens in cycle 15? As you can see the color code is different. So here this cycle will be used by another data transfer.

So this is not for this particular transaction. So another transaction's data transfer can take place here. Okay. So this way what happened is P1 has the data block, P2 requested the data block and P1 provided the data block. Okay.

So if I quickly demarcate it in the slide, you can see this portion is when P2 sends the request. Okay. Then from 6 to 10 and also from 11 onwards, so this is the phase when P1 sends the data. Okay. So this is the red color transaction, so one color. So if I quickly change the color for seeing another type of actions happening. So let's take this case. At clock cycle 6, another processor pair wants to communicate.

So here P3 wants the block from P4. So P4 has to send the block to P3. So P4 wants to send the block to P3 that is P3 has to send the request. So the request starts from clock cycle 6. So here P3 sends the request of the address, it gets the grant and then it puts the address.

Let's say it's for block number C. Okay. So let's say its communication is for block number C. So when it gets the grant, it also gets a tag assigned to it. And if we say the tag is 110 or 101, so we had like the previous one was 100, so I will call this 101. So the next transaction's new tag gets assigned. The address of the block C gets put in the third clock cycle.

And then the same thing, decode is when the sender that is P4, P4 checks the address and decides that it has to send the data block. Okay. So P4 during clock cycle 9 and 10 decides that it is supposed to supply the data block and then sends the acknowledgement. Once the acknowledgement is sent, the role of P3's request is completed. So after P3 finishes its request, P4 has to send the data which it will start doing from clock cycle 11 onwards. So first it requests for the data bus during clock cycle 11, eventually it will get a grant, it puts the tag that is 101 in this case, and then proceeds to send the data from this cycle. Okay.

So this is the request when P4 sends the data. So P4 starts sending the data from that particular clock cycle, okay. So with this you can see that we have multiple transactions happening at the same time. So as you can see in clock cycle 6, P3 is sending a request for a data item. At the same time on the data bus, P1 is now trying to get the data bus to put its data which will be then consumed by P2.

So two parallel transactions are happening here. If I go further in clock cycle 11, if you realize there will be three more transactions happening because probably a new request would be passing through here, then you have the green color transaction sending the request on the data bus and the red one which is actually sending the data on the data

bus. Okay. So this way we can establish parallelism, multiple transactions are happening together. Okay. So if I continue the same example, at clock cycle 15 that is here, the actual data transfer starts but if the receiver, which was the receiver? If I go back to the previous slide, let us see what was the receiver, it was P3. Okay. Here we said P3 receives data. But if P3 is not ready. Okay. So if P3 is not ready to receive data, what should it do? Okay. This is not related to snoop.

When snoop is not ready, you have to extend the ACK cycle. This is the extension for snoop. But when you are not ready to receive the data, you have to again not send the acknowledgement. So you have to raise a signal that you are not ready to receive the data. So this signal is seen by the sender, in this case P4. So P4 can see the signal and then stop sending or it will not start sending the data to P3. Okay. So P4 wants to send data to P3, P3 says I am not ready, so P4 waits. Okay.

So this is how transactions can happen in parallel. Only thing is we have to take care that there are no conflicting transactions in one go. All right. So this is another example. You can give it a try or revise that concept by trying to send a BusRd or a BusRdX, what happens in a BusUpgr. Okay. So you can use this as a template to understand the flow of transactions. All right.

So we have done the basic, so we have done the basic read-write things. Now the other two things left is write-back and a bus upgrade. So when you do a write-back, write-back block is, you send the data that is part of the request, is there a response? There is no response. Okay. No response comes, but to send the data as part of the request phase. Request phase only has access to the address bus. Okay.

You do not have the data bus. Data bus access is only acquired during the data response phase. Okay. So in this case, the write-back has to arbitrate both the buses together. The same processor arbitrates both buses together. In the previous examples, there were disjoint processors asking for the bus.

Here the same processor asks. So same processor requests both buses. Once it gets both together only then it starts the write-back, otherwise it does not send. That is how write-back is handled. Then in bus upgrade, as we said, when a bus upgrade is only a request and there is no response, so the same bus controller sends a response to its own cache. Right. Yes, I have succeeded in putting the request onto the bus so you can proceed with writing. So, putting the request onto the bus is like committing the transaction and saying that it is serialized. Okay.

So, these two requests, BusUpgr and the write-back. Write-back acquire both buses

simultaneously and in BusUpgr, the same bus controller sends you the response. And we assume that the write is committed when it gets the bus. Right. So, we have tried to understand how a split transaction bus works. What are the three different phases? We had a address request phase, we had a data request phase and a data response. At a time, three different transactions can be in one of these phases and everything happens in parallel. Okay. So, with this, we finish this lecture. Thank you so much.