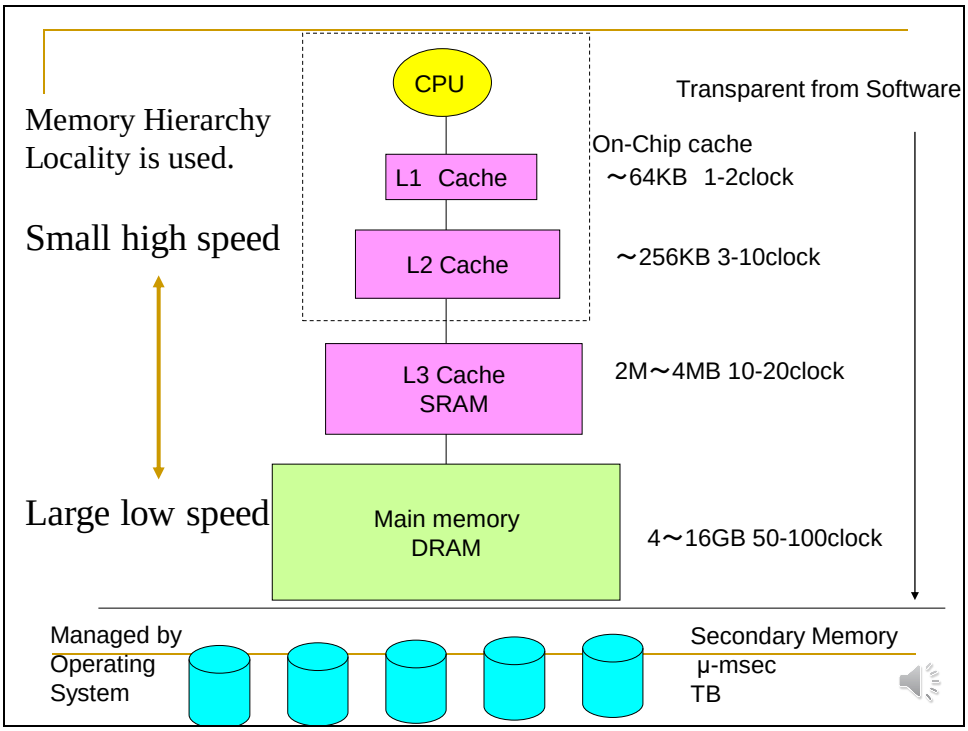
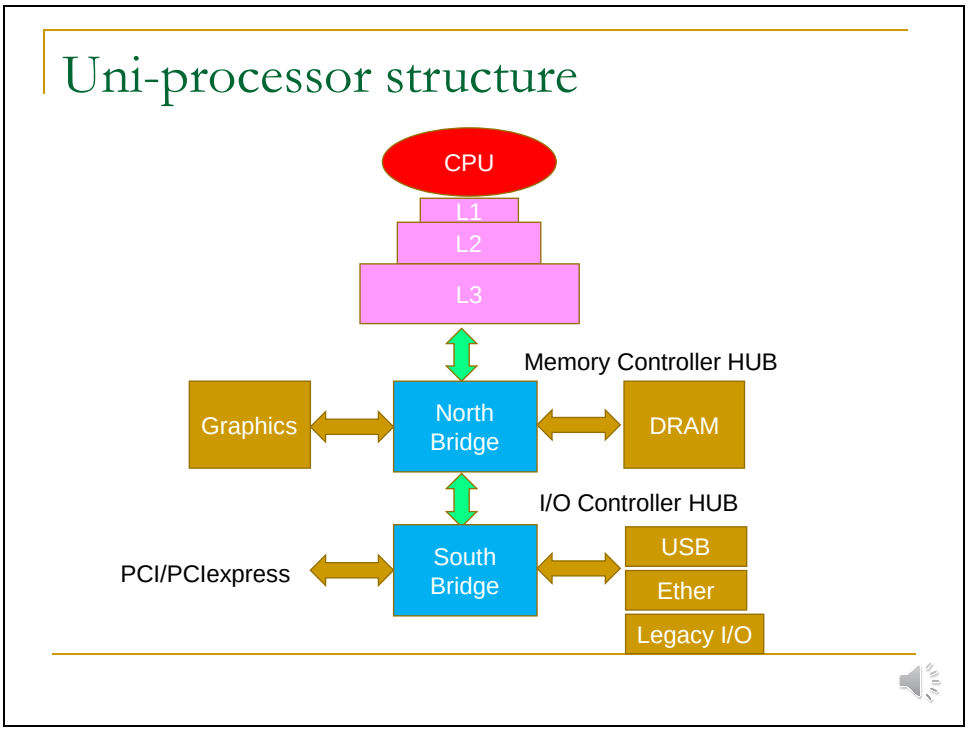
Bus / Crossbar Switch

AMANO, Hideharu
hunga@am. ics. keio. ac. jp

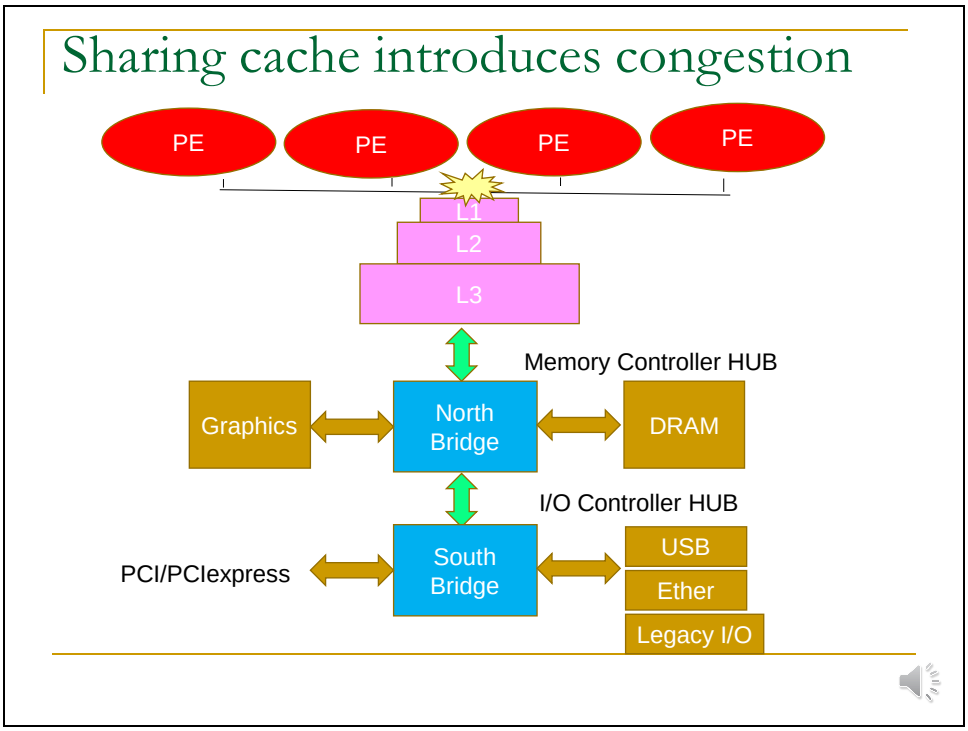
Today, I am talking about the basic knowledge on buses and crossbar switches used in UMA or multi-core.



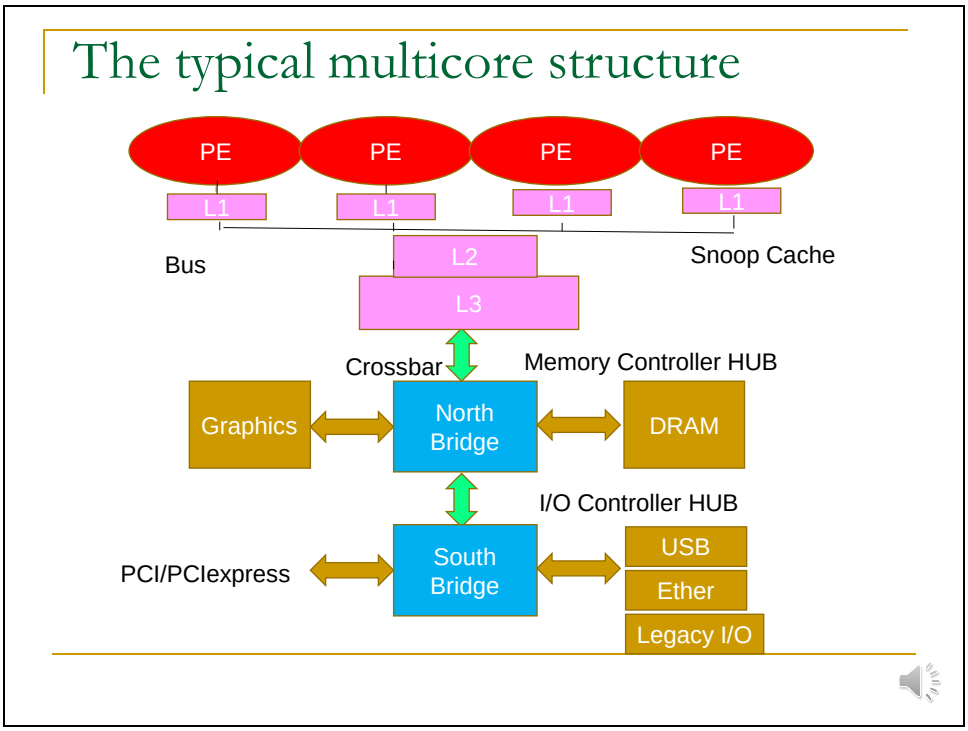
First of all, let's review on the memory hierarch. It makes the use of special and temporal locality of memory access. That is, a high speed but small memory is located near the CPU. L1 or level 1 cache is directly connected to the CPU, and L2 cache is located on the same chip. L3 cache is an on-board cache but recently it is also embedded in the same chip. These cache memory modules are all implemented with SRAM or static ram. The main memory is implemented with the dynamic RAM. From the main memory, all memory modules are transparent, that is, they have to be controlled by the hardware. The secondary memory is managed by the operating system. It has been built with magnetic disk, but recently, flush memory has taken place rapidly.



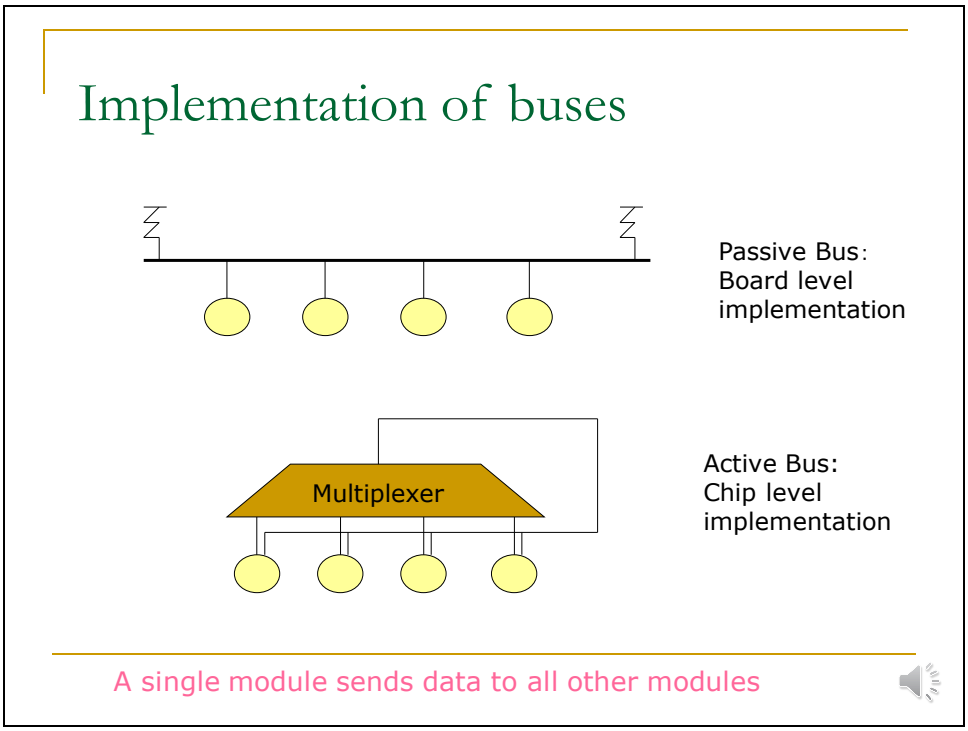
The memory hierarchy is implemented with two types of bus or switch: North bridge and south bridge. They are Intel's terminorogy. North bridge is a switch for high speed devices such as DRAMs and graphics processing unit. On the other hand south bridge is a bridge to I/O devices, USB, Ethernet and other legacy I/Os. PCIe bus is also connected.



OK. What happens when a multicore is used instead of the uniprocessor? The simplest idea is to share the L1 cache. But, it is impossible because of the severe access contention at the L1 cache, since the L1 cache is very tightly connected to the processing element.

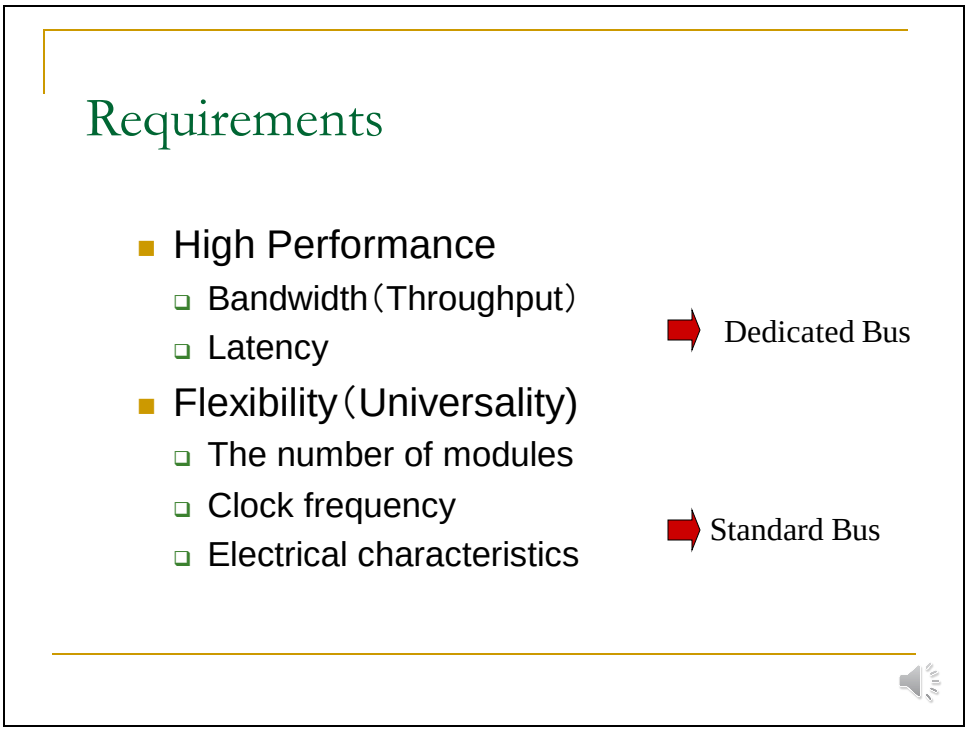


So, how about providing the private cache as the L1 cache and sharing the L2 cache. It is possible just if the cache coherence problem is solved. I will talk about this problem in the next lesson, and skip it today. Anyway, note that the bus is used to connect cores and the L2 cache. Of course, this bus is not a traditional passive bus.



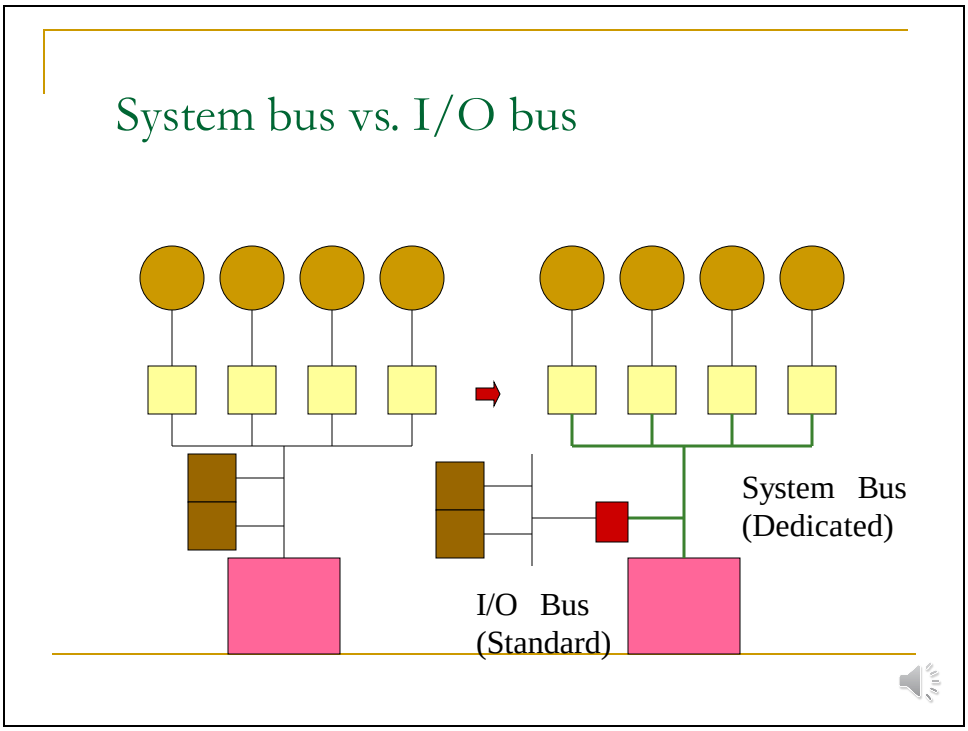
In old days, the bus is implemented with just wires pulled up registers for termination. This style of bus is still used for the back-plane bus of desktop computers. Various types of components can be connected but the operational speed tends to be slow.

Instead, buses in the chip is implemented with active gates. For example, a multiplexer is used in this case, it selects one of inputs data and distributes to all components. The definition of the bus is that anyone of the connected modules can send its data to all other members. This is sometimes called a multi-drop bus. In computers, only multi-drop buses are used.



There are inconsistent requirements to bus. First is high performance. The bandwidth or throughput indicates the total amount of data to be transferred in a cycle. The latency means the time from the sender starting data to the receiver receiving them. The high speed bus is needed to have a large bandwidth and short latency.

On the other hand, flexibility or universality is also required. That is, a bus is needed to connect a certain number of modules which use various clock frequency and electrical characteristics. Since it is difficult to satisfy both, we provide two types of buses: dedicated bus and standard bus in a system.



Most of multicore system uses a dedicated system bus to connect cache system and high speed accelerators. Also it provides standard I/O bus through the bridge chip. North-and-South bridge structure which was introduced before is one of this example.

Synchronous vs. Asynchronous

- Synchronous bus
  - Data is sent synchronized with a clock
    - Easy to handshake, block (continuous) data transfer
    - Module numbers/types are limited
  - PCI、Mbus、PCIx、PCI express, On chip buses
  - Performance centric
- Asynchronous bus
  - Data is sent without a system clock
    - Variable modules can be connected
  - VME、Futurebus+

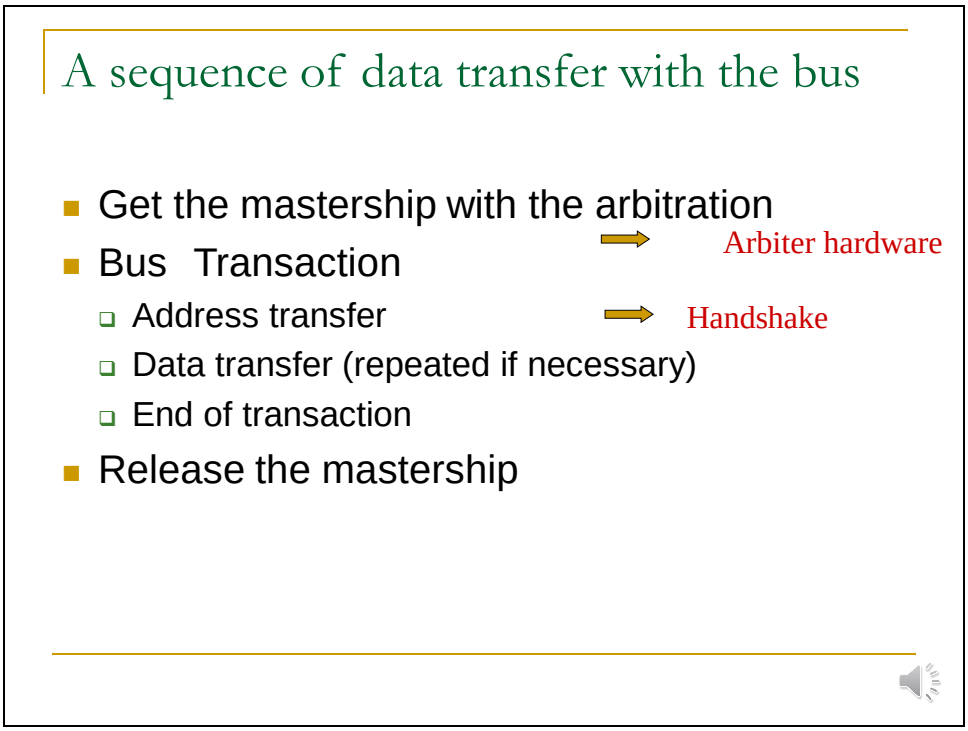
Recently, asynchronous buses are not commonly used

Buses are sometimes classified into two categories: synchronous or asynchronous. In synchronous buses, data are sent synchronized with a clock. This bus can support a high speed block data transfer but modules numbers and/or types are limited. They must follow the fixed bus clock. On the contrary, in asynchronous buses, there is no system clock, and so every data transfer requires a handshake. However, this classification is rather old-fashioned. Recent buses are all synchronous bus, and asynchronous buses seem to be extencted.

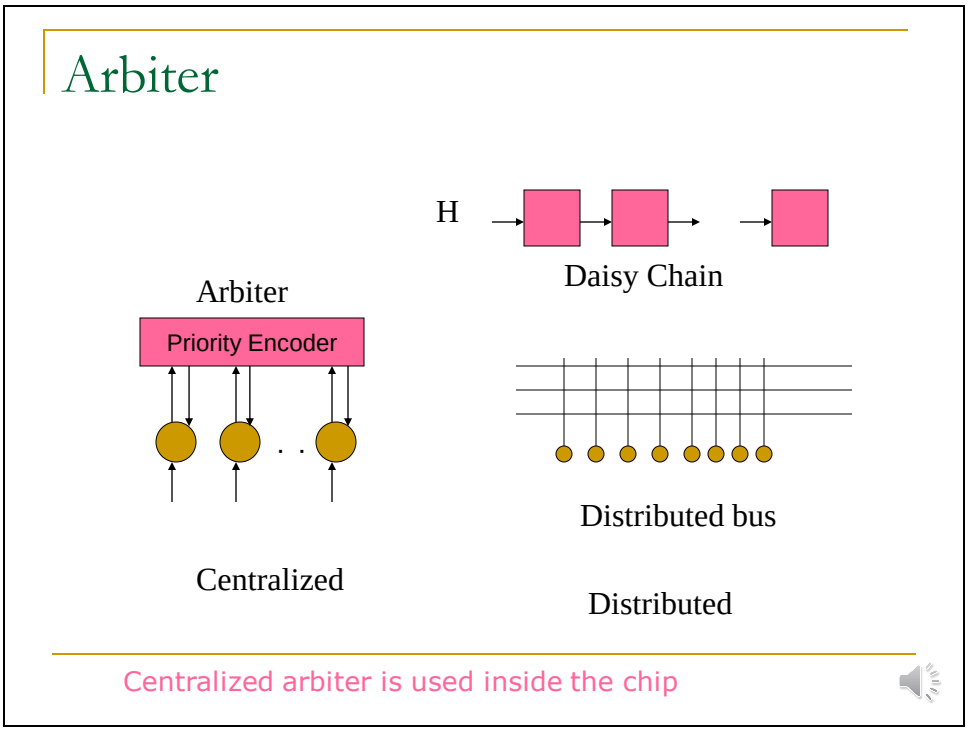
Terms around bus

- Transaction: A continuous data transfer of address and data
- Arbitration : An operation for taking a right to control the bus
- Bus Master: a module which had a right of controlling the bus through the arbitration
- Bus Slave : modules except the bus master

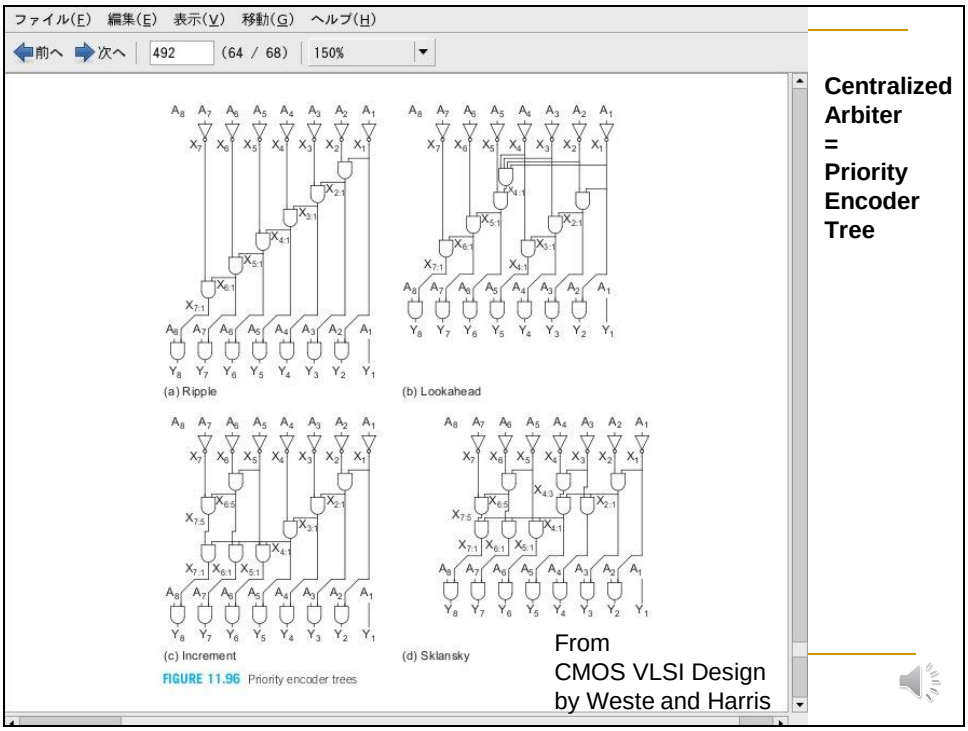
Now, let me explain some terms around bus.



When you want to use a bus, first, you have to get the mastership with the arbitration. Then, you start the bus transaction. Usually, address and data are multiplexed, so first, address is transferred. At that time, handshake is needed between the master and slaves. Then, data transfer starts. Multiple data are often transferred iteratively synchronized with a clock in the synchronous bus. At the end of transaction, sometimes, data for error detecting or correction code are transferred. Then, the master releases the mastership, and the next transaction starts by another master.



Arbiter is classified into centralized and distributed. The simplest centralized arbiter is a priority encoder. It selects a winner from multiple competitors. For multicores, since every module is embedded into a chip, the centralized arbiter is used.

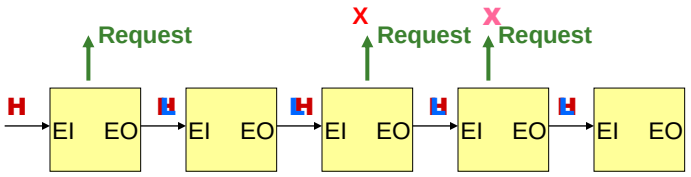


Centralized Arbiter = Priority Encoder Tree

From
CMOS VLSI Design
by Weste and Harris

There are various types of priority encoders. They somehow, resemble to adders. The simplest one uses the ripple carry structure. Like the adder, it has the problem of long delay time. So, look ahead, incremented, and Skransky methods have been proposed to reduce the delay. Of course, they need extra hardware.

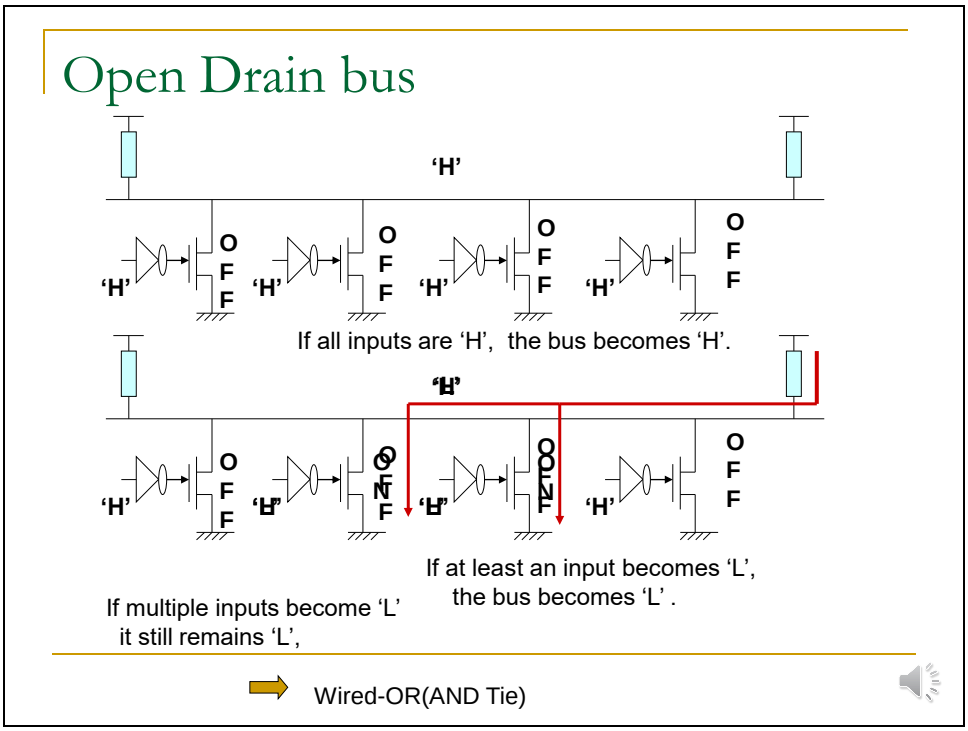
Daisy Chain



If no request EI→EO
The request can be issued only if EI is H level
When the request is issued, EO becomes L level

Right side module has a low priority
Left side module has a high priority

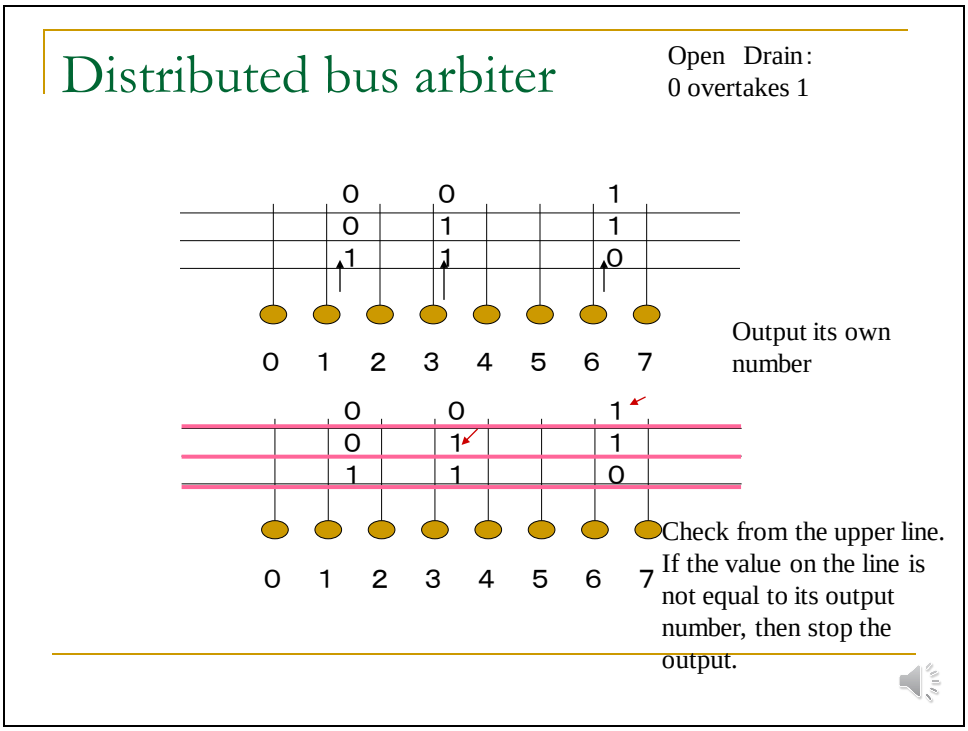
Daisy chain is the simplest arbiter. The modules are connected tandemly with the signals EI and EO. The rule is simple, if there is no requests, the module transfers the level of EI to EO. The request can be issued only if EI is High level, and negates EO. All the following modules' EI turns Low, thus they cannot issue the request. Obviously, the leftmost module as the highest priority. Since this method is easy to implement, it is often used for an arbiter Direct Memory Access or DMA requests.



The traditional backplane bus uses the open drain gate. In such a gate, the drain of output transistor is opened and connected together.

The high level is supplied through the termination resisters. In this case, if all transistors are off, the bus wire becomes high because it is pulled up by resisters.

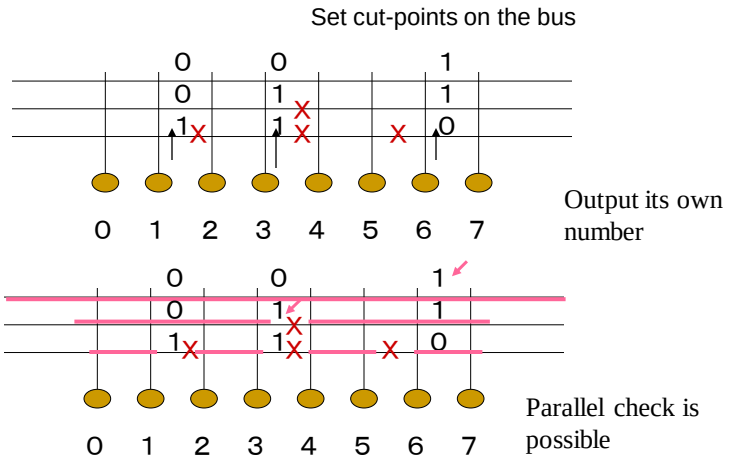
If at least an input becomes low, the transistor turns on, and the level becomes low. The case of two or more transistors are ON, the same thing happens. Since at least an input becomes low, the wire is low. Thus, this mechanism is sometimes called Wired-OR. Some people call it AND Tie.



By using open drain buses, a log2N distributed bus arbiter can be built. We need log2N buses for N modules. Here, there are 8 modules, so three wires are needed. Each module sends its own binary number to the bus. Here, assume that module 1, 3, and 6 issue the request.

Then, the bus is checked from the highest digit. If the level is not the same as its own level, it removes its request from all wires. In this case, the module six withdraws since it receives 0 instead of its issued 1. Then, the second wire is checked. This time 3 withdraws. Finally the module 1 wins.

Modified method (Keio's patent)



The problem of the distributed arbiter is it takes Log2N clocks to fix the result. Keio university invented the improved the method. The key of magic is to cut the wire and divides it into 4 segments and 2 segments like this.

Then the same algorithm is applied, but this time the winner can be selected just with a clock cycle, since 0 is not propagated beyond the cut. I think it is a good idea, and Keio University got a patent. Unfortunately, no one was interested in this idea and this patent was expired without getting any money.

Starvation Problem

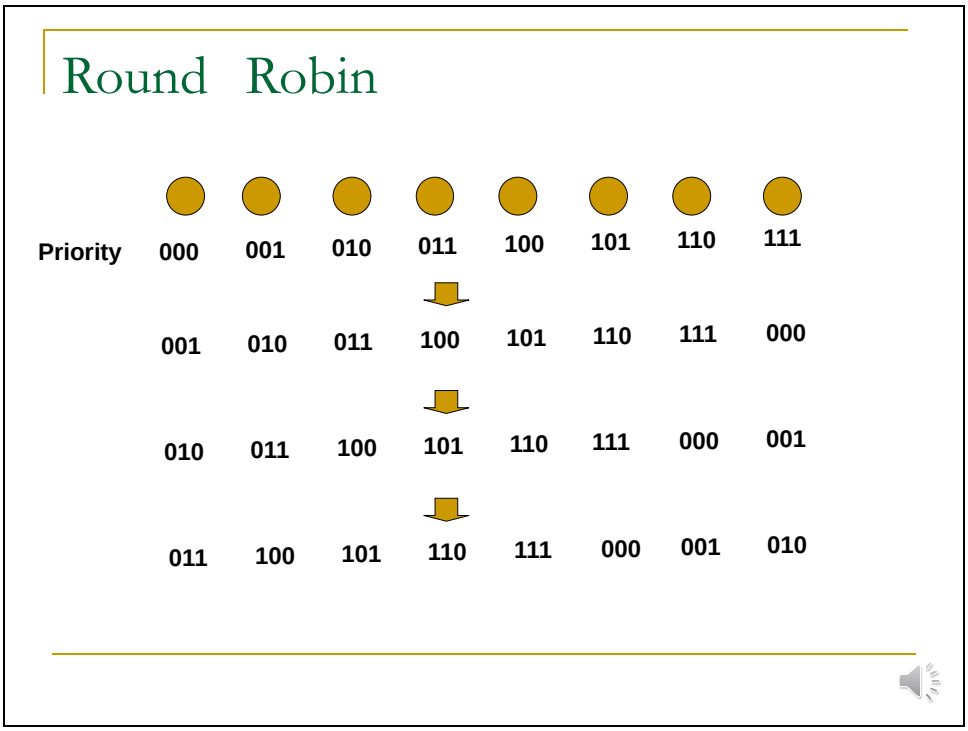
- If the priority of the arbiter is fixed, a weak module cannot use the bus continuously.
- Central arbiter

→ Round robin priority scheduling

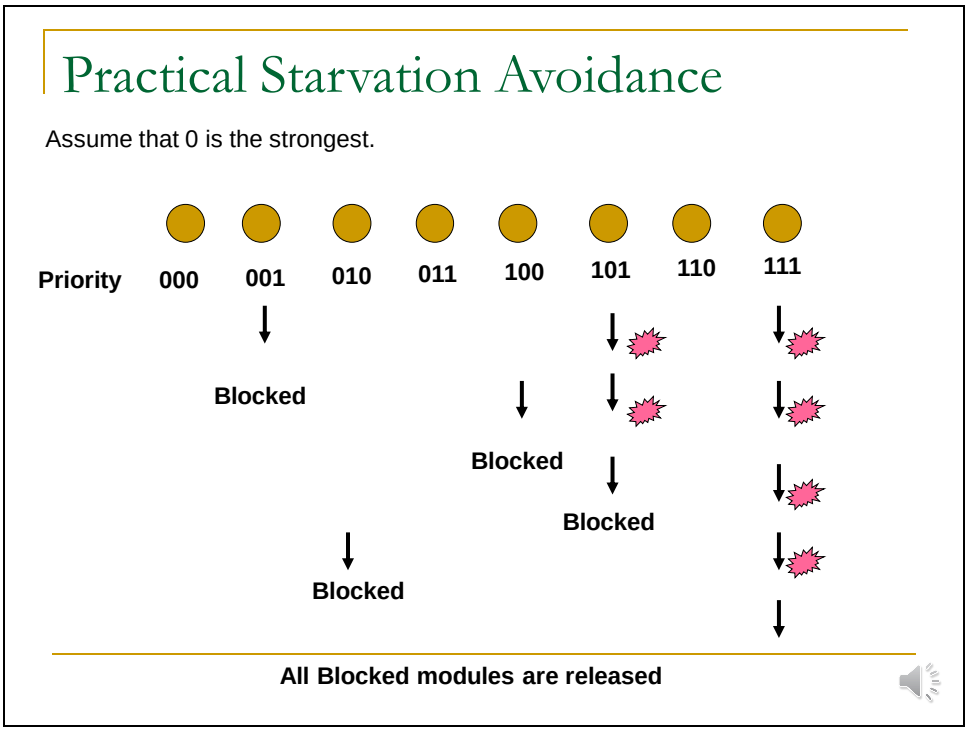
- Distributed arbiter

→ The next request cannot be issued until all requesting modules satisfy their requests.

If the priority level of the arbiter is fixed, the starvation problem may happen. That is, a weak module cannot use the bus continuously. For the central arbiter, the round robin priority scheduling can be implemented. That is the priority level is shifted between modules. For the distributed arbiter, it is hard to be implemented. In this case pheudo round robin method is used.

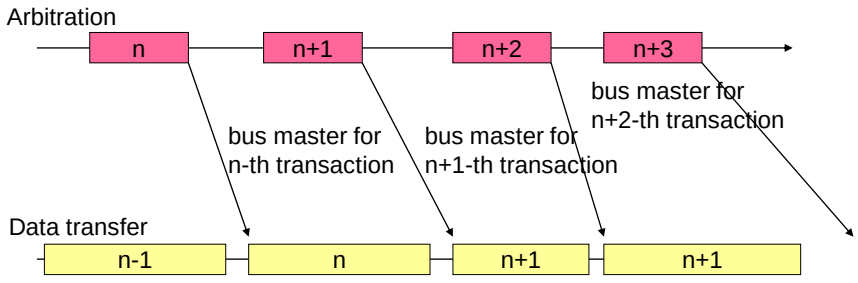


This diagram shows the round robin priority scheduling. After an arbitration is finished, the priority shifts so that the fairness of the bus is kept.



Since the round robin scheduling takes cost, a simpler policy is often used. The rule is simple, a module who won the arbitration cannot issue the next request until there is no requesting module. This diagram shows an example.

Overlap between the arbitration and data transfer

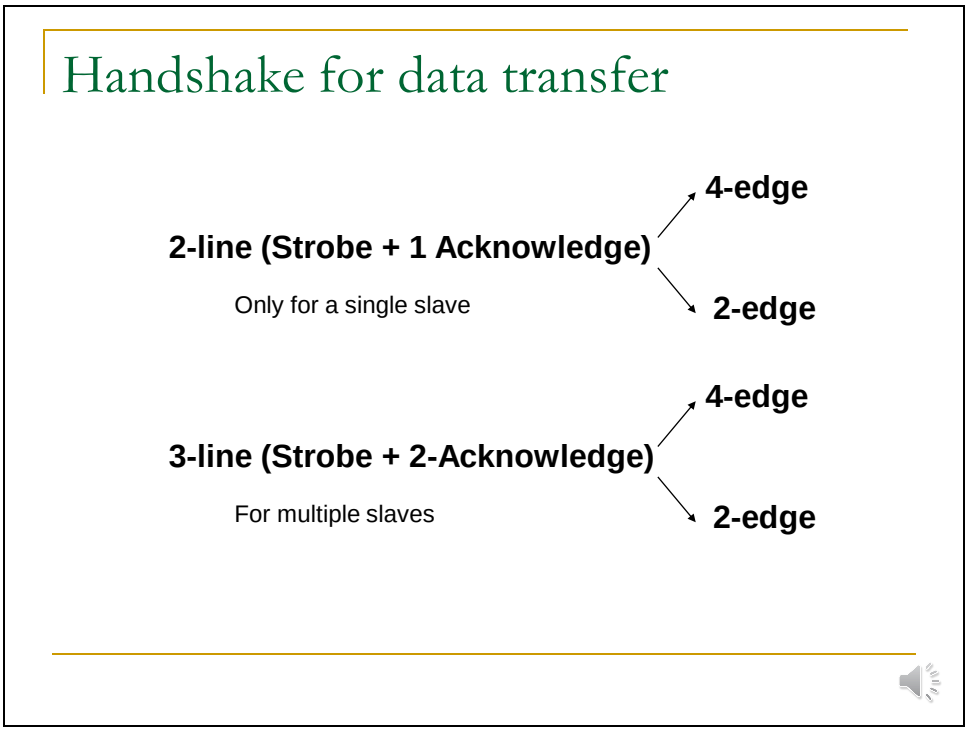


So, the arbitration time is not critical in most cases.

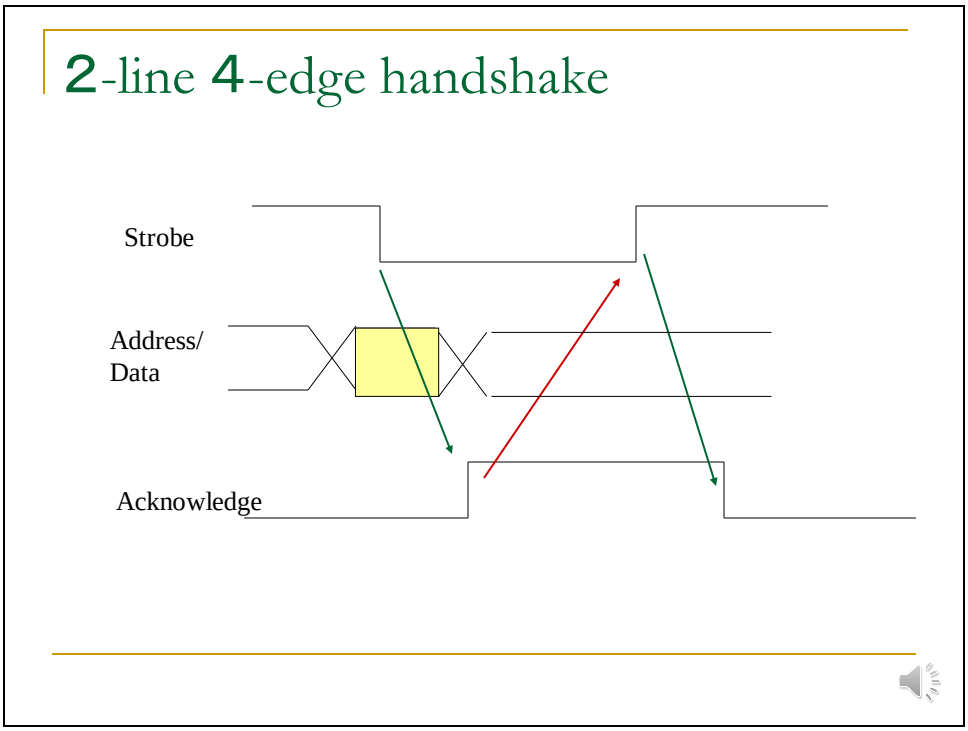
The bus arbitration is usually done overlapped with bus transaction itself. Thus, during a transaction, the next bus master is selected, and the time of bus arbitration is hidden. This is why the speed of arbiter is sometimes not treated as a critical matter.

glossary-1

- Arbiter 調停回路
- Arbitration　調停操作、バスマスタを選ぶ
- Bus master　バスマスタ、バスの利用権を管理するモジュール
- Bus slave　バススレーブ、バスの利用権を持たないモジュール（マスタからスレーブに常にデータを転送するわけではないので注意！）
- Centralized 集中型　⇔　Distributed 分散型
- Daisy Chain　Arbiterの一方法で、ヒナゲシの花輪から来ている
- Transaction　バス上でデータを転送するための一連の操作
- Open drain　オープンドレイン、バスの作り方の一つで、出力トランジスタをオープンにして抵抗につなぐ。全てがOFFのときのみHレベルになり、どれか一つでもONになるとLレベルになる。この操作をワイヤードORと呼ぶ。
- Starvation　飢餓状態、バスの利用権を獲得できない状態が長期間続くこと
- Round-robin　ラウンドロビン、優先順位をArbitration毎に隣りのモジュールに移動していく方法

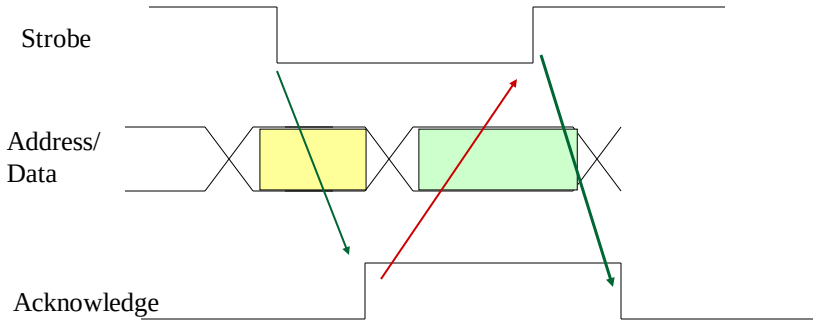


Now, let me explain about the handshake mechanism. For a bus with a single slave, 2-line handshake is enough. If it transfers data with four edges, it is called 4-edge handshake, and with 2 edges, it is called 2-edge handshake. For multiple slaves we need three lines.



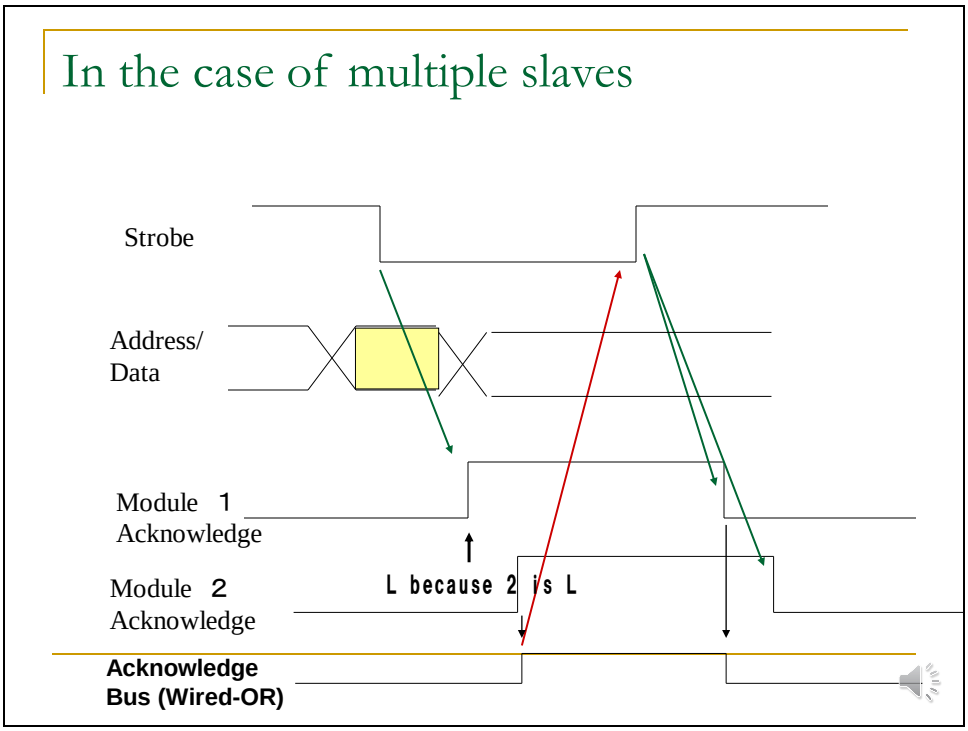
This shows a waveform of the 2-line 4-edge handshake. Here is an example of transferring the data from the master to a slave. The master set the address or data on the bus and assert the strobe. The slave checks the strobe and receives the content on the bus at the negative edge of the strobe signal. After receiving the slave changes the acknowledge low to high. The master removes the content and changes strobe to high. After checking it, the slave returns the acknowledge signal and goes to the next step. This method uses four edges to send one item on the bus.

2-line 2-edge handshake



Data ttem is transferred with both edges of the strobe

4 edge handshake is time consuming. So, we can send the next data at the rising edge of the strobe. This is called 2 edge handshake. Address or data are transferred at the both edge.

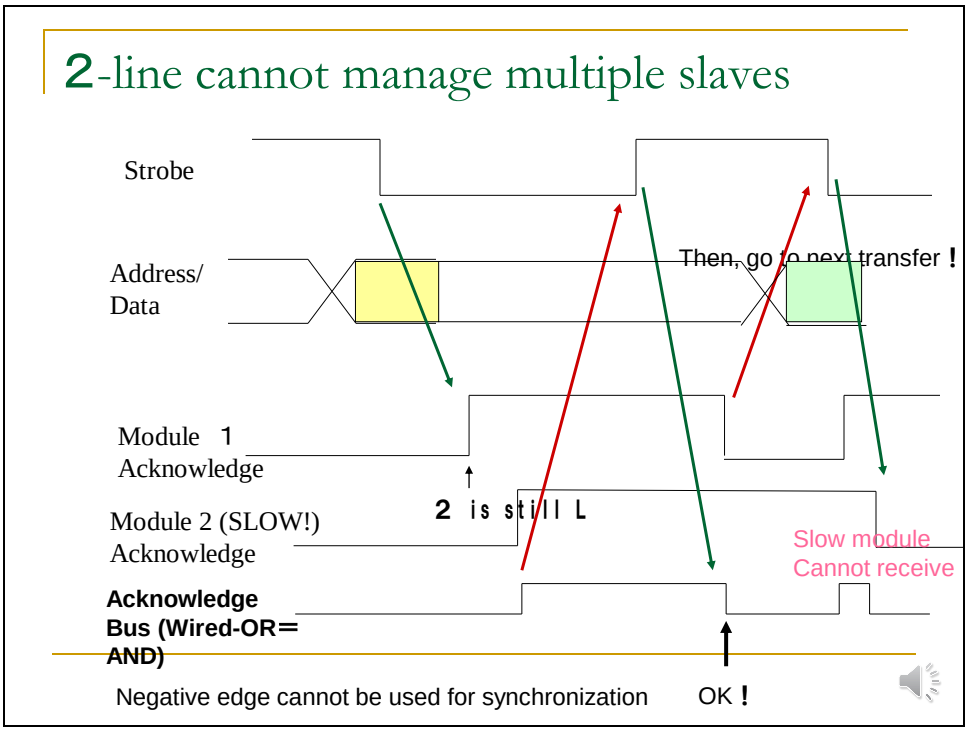


Two line handshake does not work well when the number of slaves become two or more. In this case, when both slaves receive the data, they turn acknowledge line high. Since open drain bus is used for acknowledge line, it turns high when the slower module changes the level. This diagram seems well.

Quiz

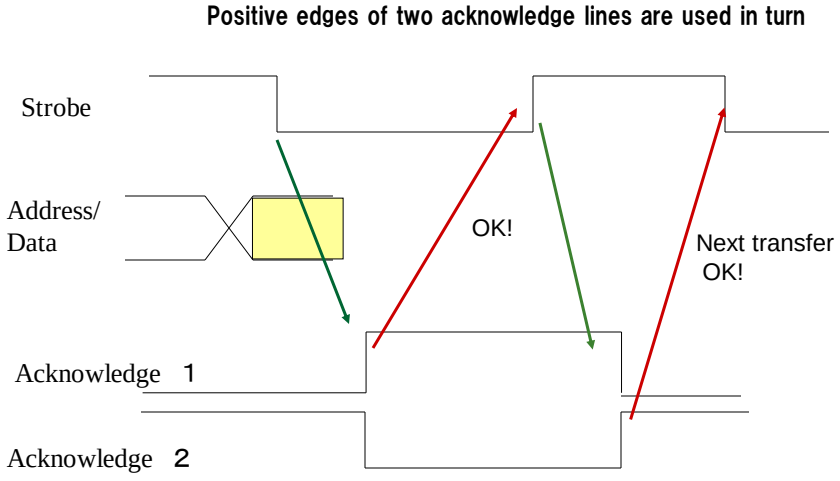
- 3-line handshake (1 for strove and 2 for acknowledge) is used for multiple slaves.
- Why 2-line handshake cannot manage multiple slaves?

Let’s think well. Why 2-line handshake cannot manage multiple slaves and another line is needed?



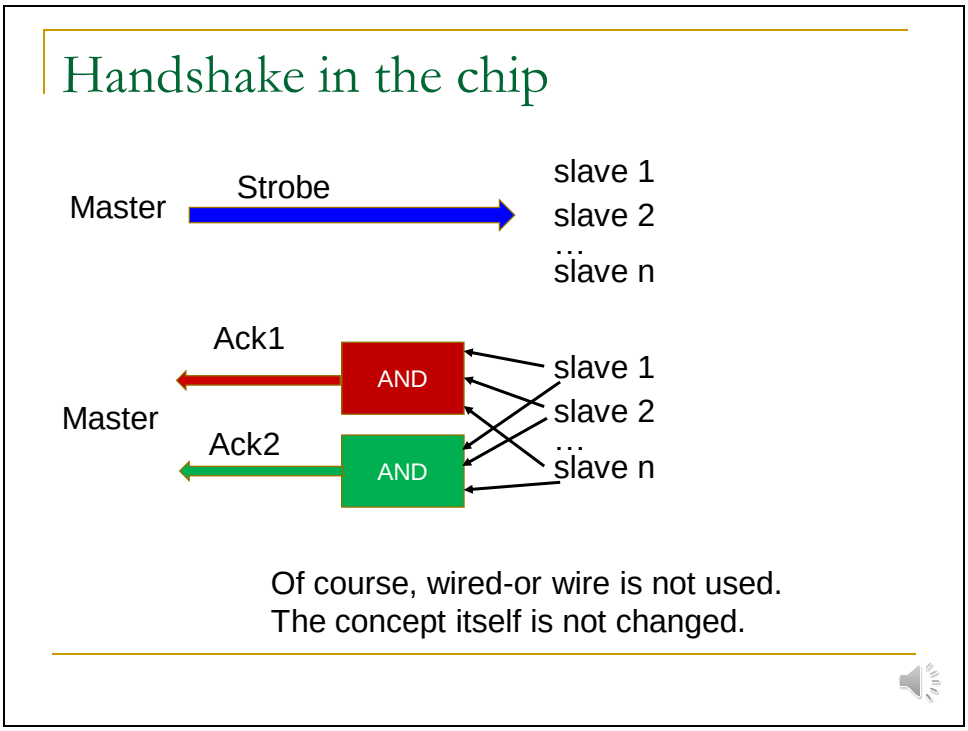
This diagram illustrates the reason. Assume that the module 2 is extremely slow. Even the case, the acknowledge for receiving data will transferred well because of the wired or bus. When the slower module turns the level high, the bus level becomes high. However, when master wants to go to next step and turns the strobe high. In this case, the acknowledge signal becomes low when the faster module turns the level low. So, the master cannot recognize whether the slower module is ready or not. In this case, the master can go the next bus transaction and misunderstand the acknowledge of the slower module.

3-line handshake

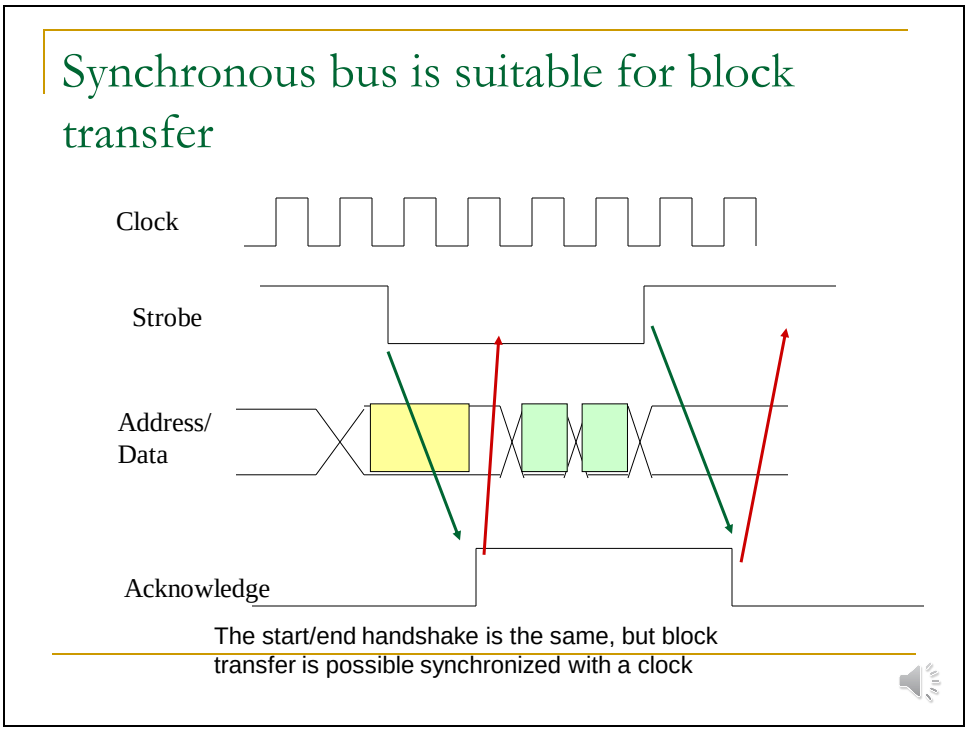


3-line 2-edge handshake is also possible

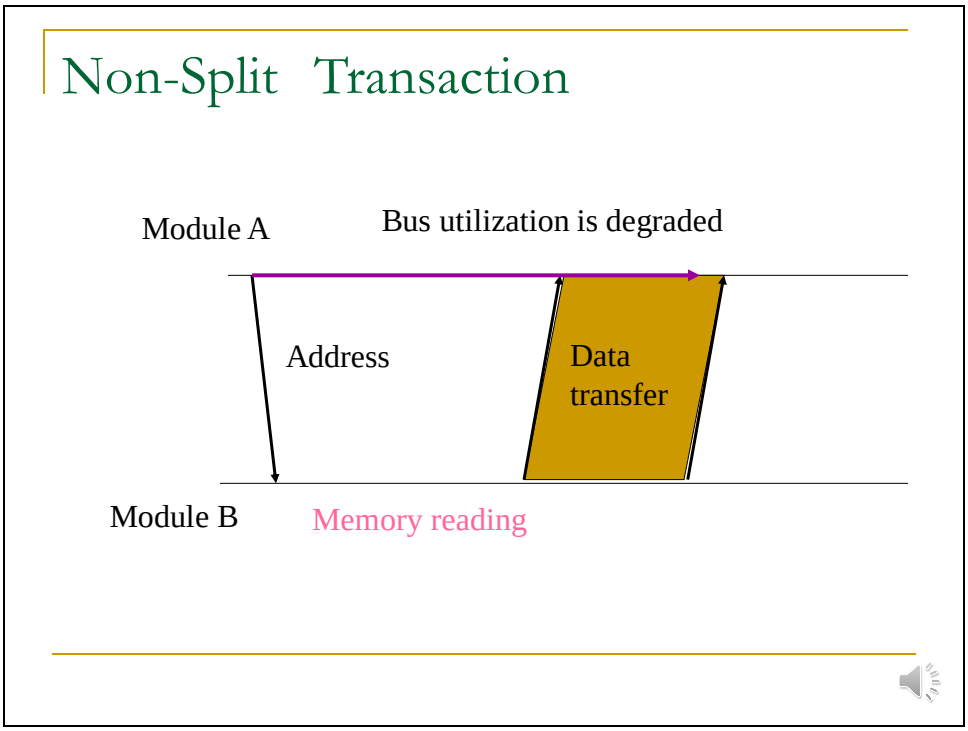
This problem can be solved by providing another acknowledge signals and using them in turn. In this case, only the rising edge can be used for the handshake, and it certainly becomes high when the slower module turns to high. This diagram shows 3-line 4-edge handshake, but 3-line 2-edge handshake is possible.



Inside the chip, of course, the wired-or wire is not used, but the concept itself is not changed. Two acknowledge signals are used to inform master whether slaves are ready or not.

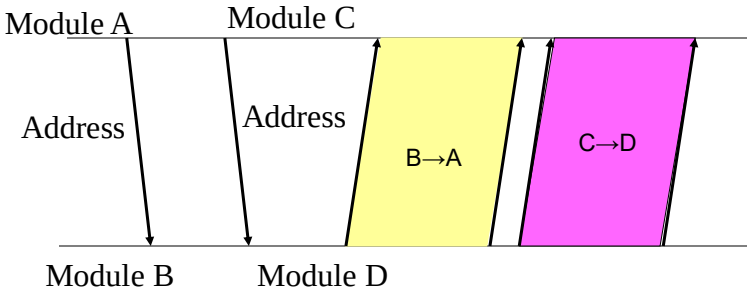


Usually, the handshake is required when bus transaction is started. But, the data transfer can be done synchronized with the bus clock, and continuous data transfer can be done. This is why the synchronous bus is suitable for block or burst data transfer. After that, the handshake is taken.



On the basic bus, the bus is locked during the waiting time. Assume that the module A is a core and module B is a memory. When a core sends an address with reading request, the memory starts reading the data. After the memory is ready, the data block is transferred.

Split Transaction



Split transaction of A→B

Transaction C→D is executed

For efficient use of such a waiting time, the split transaction is used. After sending the address and reading request, the module A releases the bus. So, the module C can use the bus for its purpose. After the waiting time, the split transaction from B to A starts again. After that, the data transfer from C to D is executed.

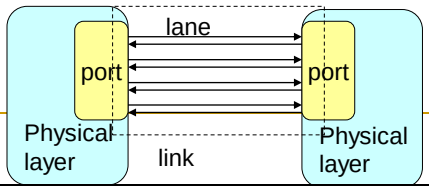
Advanced I/O Buses

- PCI bus was widely used, but it could not cope with recent computer system.
  - 32bit/33MHz, 64bit/66MHz
- New standard I/O bus
  - PCI-X
    - 64bit/133MHz DDR/QDR
  - PCI Express
    - Point-to-point serial data transfer
    - 1 lane:2.5Gbps
    - x2, x4, x8
  - Now, PCI Express is used instead of PCI bus.

As the bus for Personal Computers, PCI bus was widely used until about 1990's. It was 32bit bus with 33MHz clock and then extended to 64bit with 66 MHz. But, it could not cope with the performance improvement of personal computers. Then, the PCI-X, a straight extension was shortly, but they were replaced with a new concept PCI express. It has been widely used.

PCI Express

- Consisting of serial one-to-one bidirectional connection wires called lanes.
- Each lane supports 2.5Gbps/5Gbps (Physical Speed)
- Multiple lanes can be used as a link(x4, x8, x16 and x32).
- The data is transferred in a packet called TLP (Transaction Layer Packet).
- Interconnection network rather than the bus, but the protocol of traditional PCI bus is supported.



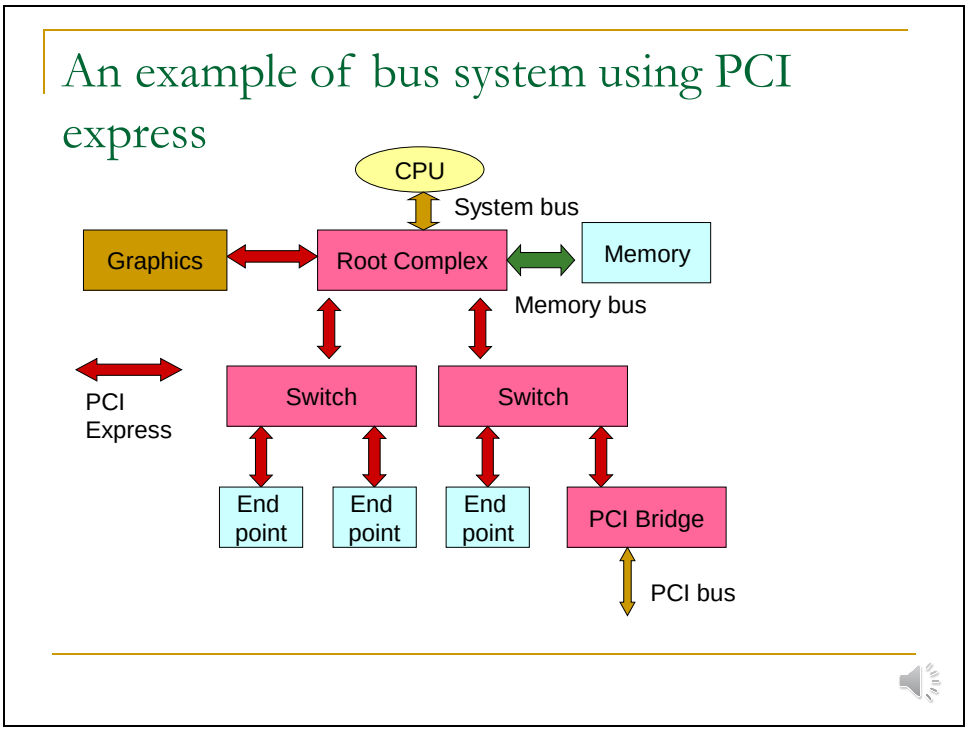
PCI Express is actually not a bus in traditional sense, but a serial high speed one-to-one bidirectional connection. There are several bi-directional lanes and form a link. The data is transferred in a packet called TLP or Transaction Layer Packet. It is a packet switching network, but the protocol of old PCI bus is supported.

PCIe standard

	Gen1	Gen2	Gen3
Physical speed（Gbps）	2.5	5	8
Bandwidth (GB/sec)	0.25	0.5	1.0
x8 bandwidth (GB/sec)	2.0	4.0	7.9
Encoding	8b/10b	8b/10b	128b/130b

Physical speed is x1.6, but almost twice practical performance is realized by changing the encoding method.

OK. Let’s review PCIe standard. The physical link speed of Gen3 is 8Gbps, the 1.6 times as that of Gen2. But, by using 128b/130b coding instead of 8b/10b used in Gen2, it achieves almost twice bandwidth that of Gen3.



PCI express is used with a tree like structure. It is connected with CPU through the root complex. It is actually a switch which has multiple links to other switches. I/O modules are connected to end-point.

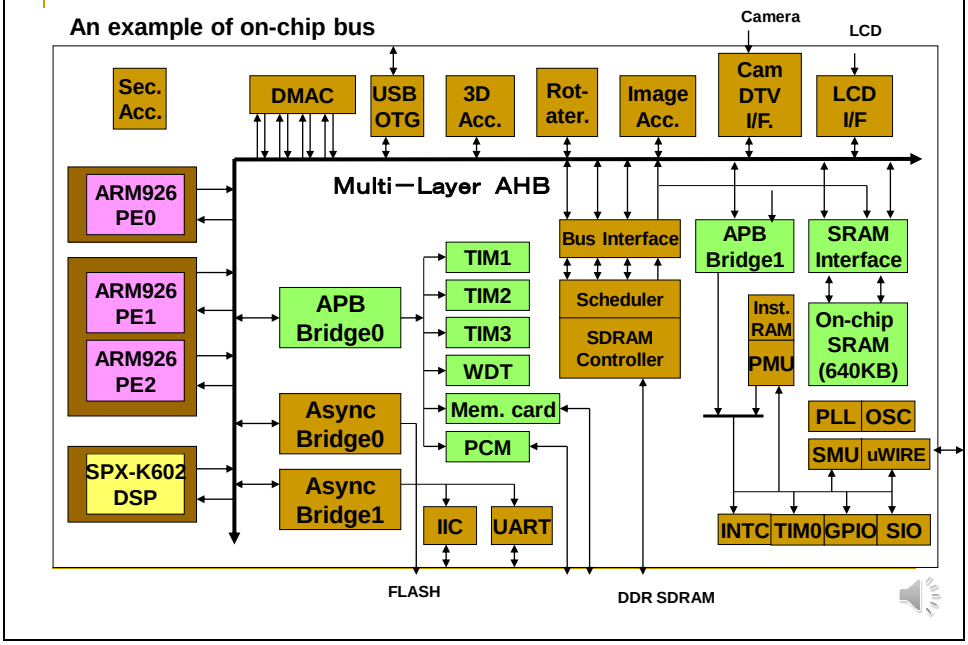
On-chip bus

- For on-chip implementation, various types of IP (Intellectual Property) must be connected.
- Standard bus is required.
  - AMBA (Advanced Microcontroller Bus Architecture): a bus for ARM cores.
  - CoreConnect: a bus for PowerPC cores.
  - Wrapper based buses
    - IPs are wrapped in the standard interface.
- For further performance improvement, NoCs (Network on Chips) are introduced.

→ Introduced in the later part of this lecture

There are various types of on-chip bus, but recently network-on-chips or NoCs are becoming used popularly.

NEC MP211


An example of on-chip bus
Camera
LCD
Sec. Acc.
DMAC
USB OTG
3D Acc.
Rot-ater.
Image Acc.
Cam DTV I/F.
LCD I/F
Multi−Layer AHB
ARM926 PE0
ARM926 PE1
ARM926 PE2
SPX-K602 DSP
APB Bridge0
Async Bridge0
Async Bridge1
TIM1
TIM2
TIM3
WDT
Mem. card
PCM
IIC
UART
Bus Interface
Scheduler
SDRAM Controller
APB Bridge1
Inst. RAM
PMU
SRAM Interface
On-chip SRAM (640KB)
PLL
OSC
SMU
uWIRE
INTC
TIM0
GPIO
SIO
FLASH
DDR SDRAM

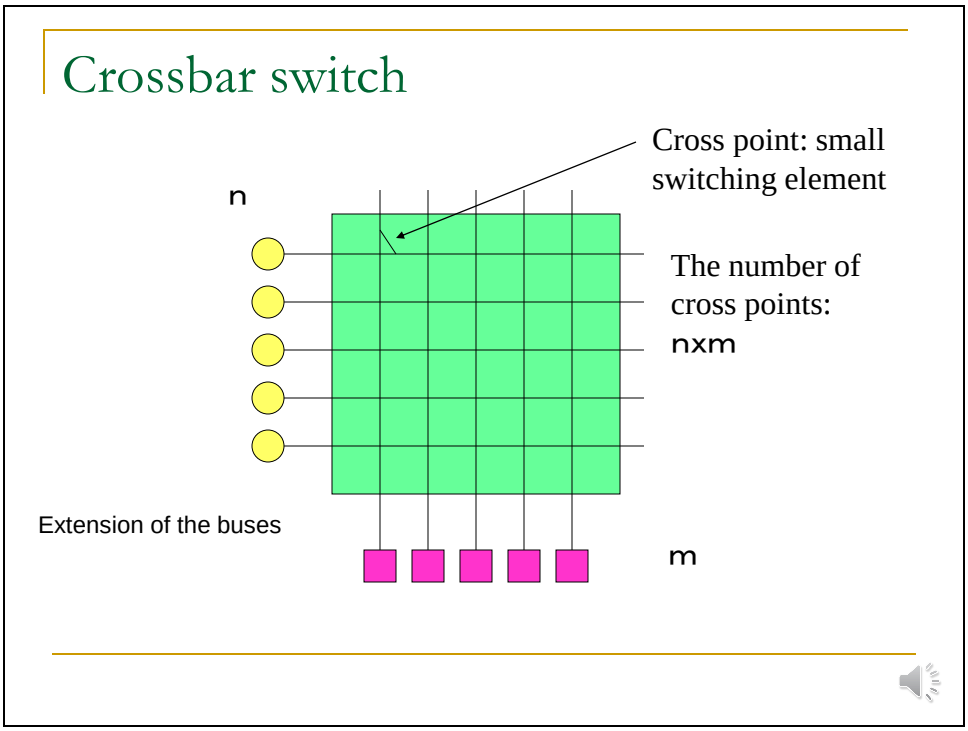
Summary of Bus

- Classic bus with passive wires has been changed to active bus with a kind of switches
- High Speed Bus
  - Synchronous bus with Split Transaction
  - Using active devices
  - It becomes somehow like a packet transfer with switching hub.

OK. This is a summary of buses.

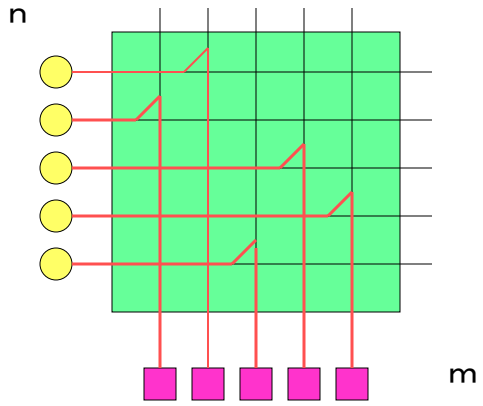
glossary 2

- Handshake 握手のことだがここでは正しく転送するための信号のやりとりを指す
- Synchronous 同期式⇔Asynchronous　非同期式
- Strobe　転送を起動を知らせる信号線
- Acknowledge　Strobeに対する応答用の信号線
- Edge　信号線の変化
- Split transaction　バス転送を中断して途中に他の転送を挟むことを可能にする方法



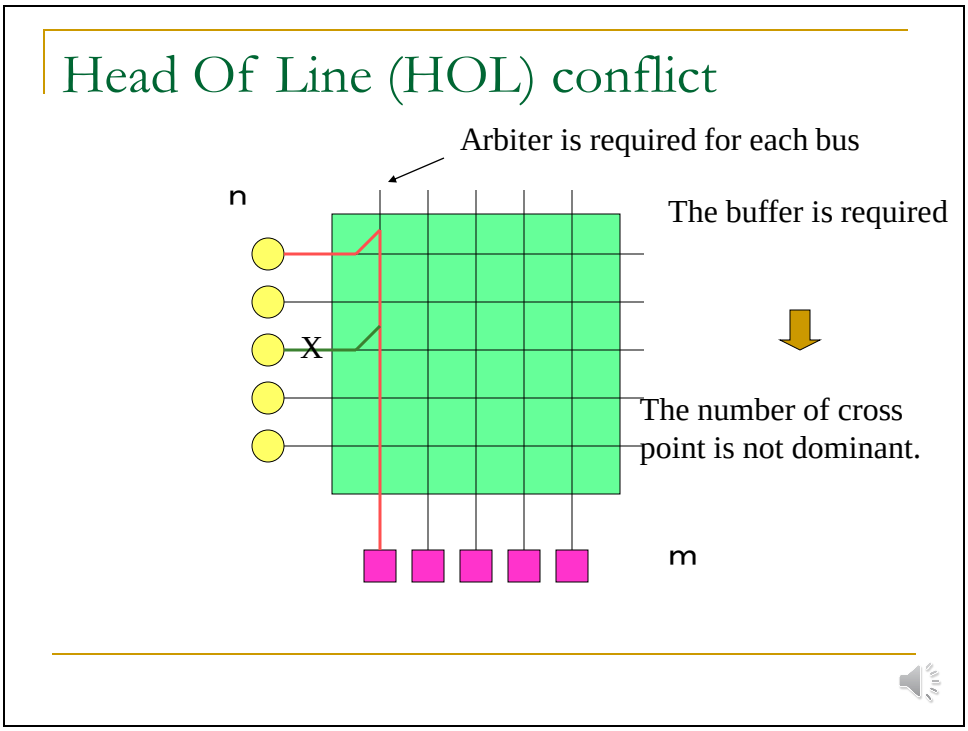
Today's next target is a crossbar switch. Some of you may feel strange, because the crossbar is a typical switch. However, the crossbar switch is actually an extension of multiple buses. Here we assume n cores and m memory. Providing small switching elements at the cross points of their individual buses, the crossbar can be built. That is, it requires nxm cross point switches.

Non-blocking property



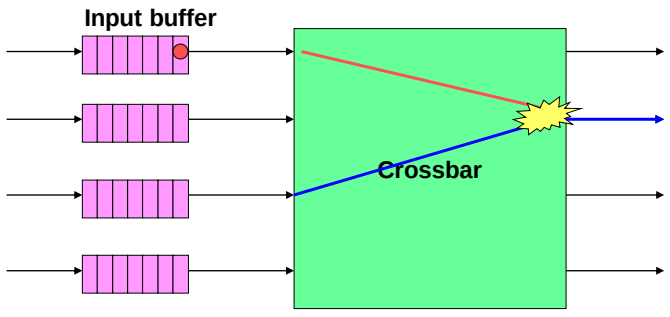
For different destination, conflict free

By make the best use of cross point switches dedicated lines can be formed for the different destination. That is it is conflict free or non-blocking.



However, of course, if the destination module is the same the multiple requests conflict with each other. In this case, like common buses, the arbiters are used. Some people call this conflict, head of line or HOL conflict.

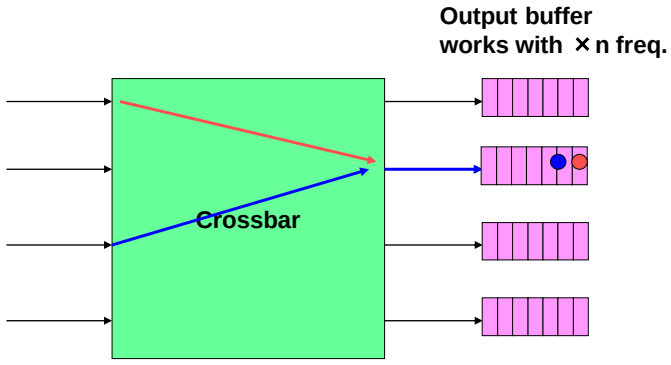
Input buffer switch



One of conflicting packets is selected.
Others are stored Into the input buffer

When conflict occurs, one of the requests is usually stored in the input buffer or FIFO attached to the crossbar. After another request passing the crossbar, the waiting request is transferred. The similar mechanism can be provided for the bus. Parallel machines commonly use this style so called input buffer switch.

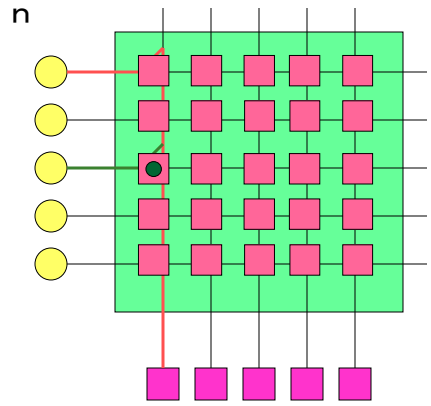
Output buffer switch



Crossbar must work with ×n frequency of input/output rate.
No HOL problem.
Used in switches in WAN, but for parallel machines it is difficult.

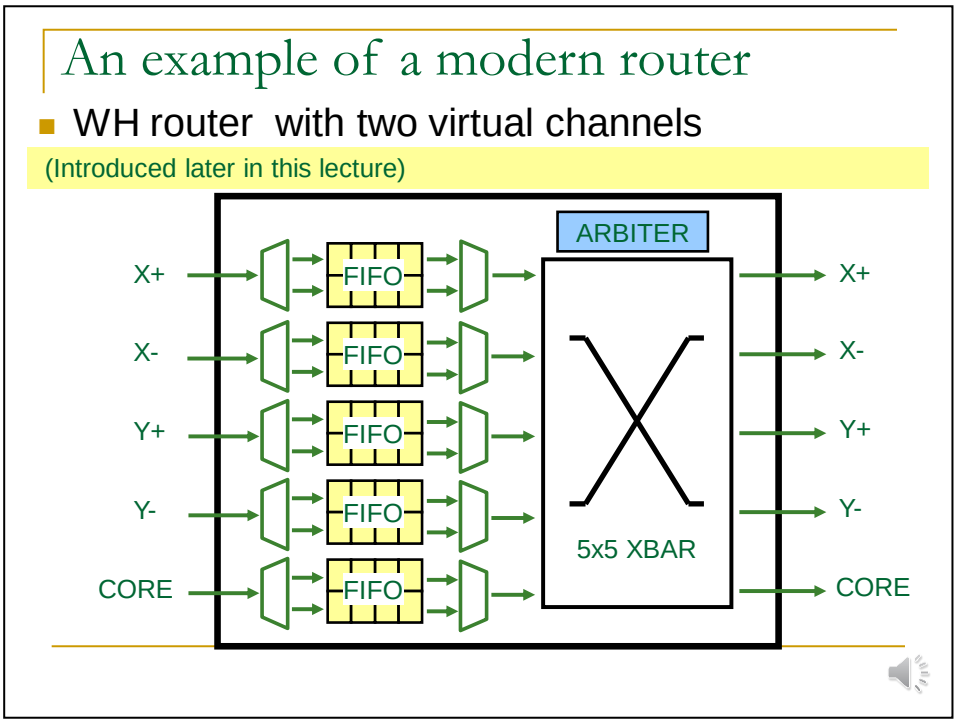
Some people claim that there is output buffer switch. But, for parallel machine, it is rarely used, because for solving the HOL conflict, the multiple data must be transferred to the output buffer. It means that the output buffer must work with n times clock frequency. This situation is almost impossible for multicore machines.

Buffers at cross-point



The buffer is provided at each cross-point. High performance but the total amount of buffer becomes large.

Another idea is providing buffers at cross point. It achieves high performance but the total amount of buffer becomes huge.

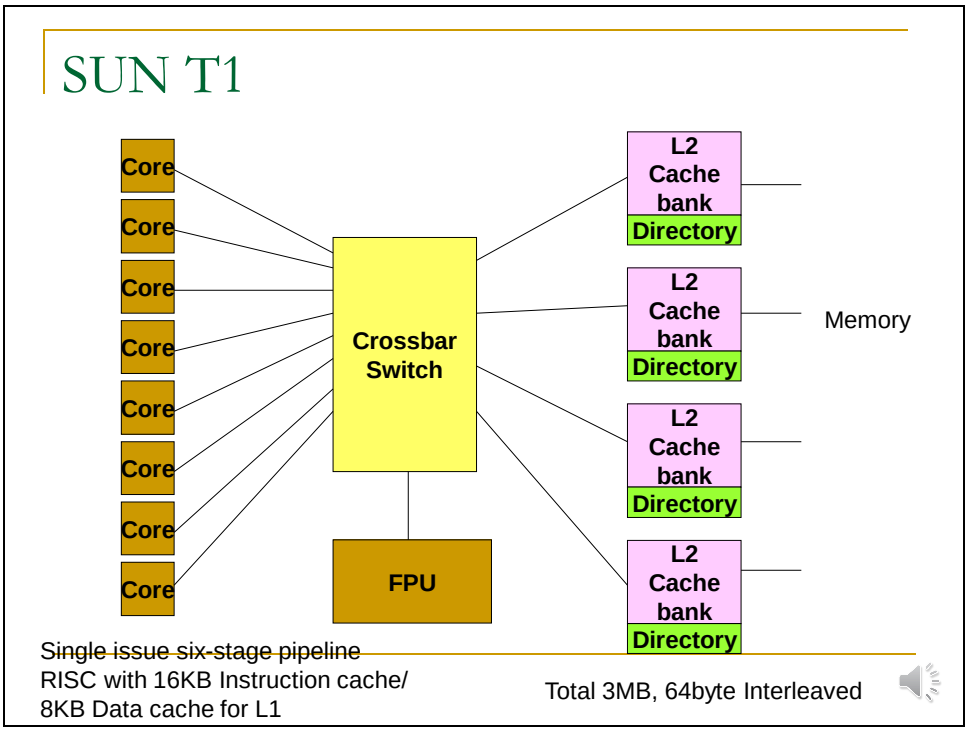


This is a practical design example of a modern router using the crossbar. I will explain this structure later in this lesson.

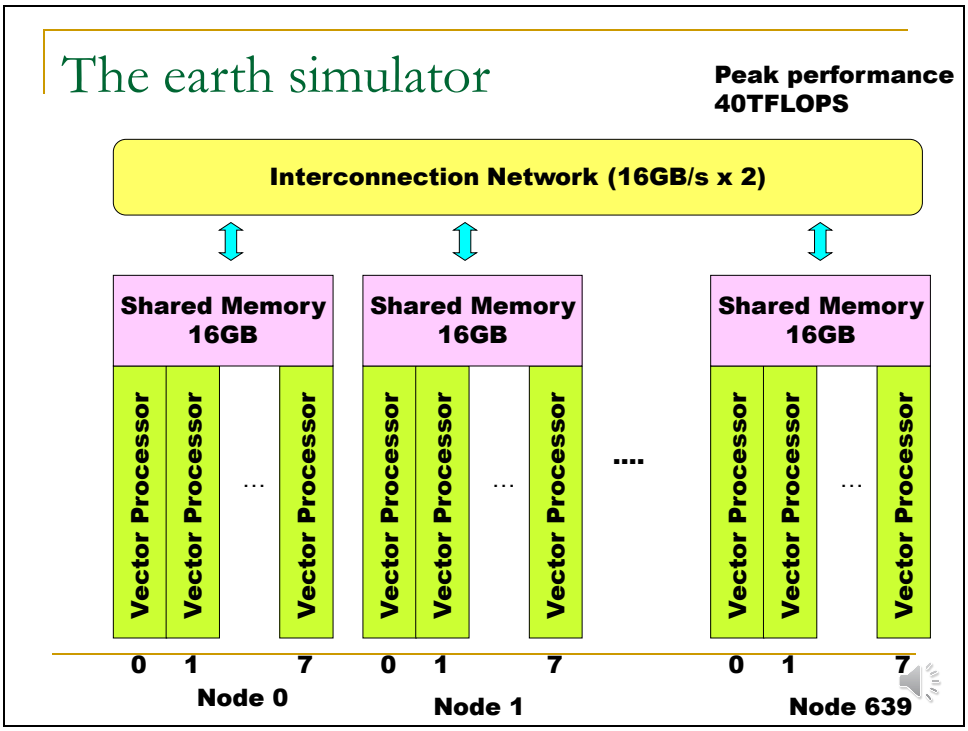
Merit/demerit of Crossbars

- Non-blocking property
- Simple structure/Control
- The hardware for cross-points usually do not limit the system (Fallacy of crossbars)
- Extension is difficult by the pin-limitation of LSIs
  - If pins can be used, a large crossbar can be constructed → Earth simulator

Here, the pros and cons of crossbars are summarized.



This is an example of Sun microsystem's earlier multi core processor.



Japanese supercomputer “the earth simulator” used a huge crossbar. It connected 639 modules.

glossary 3

- Crossbar switch: クロスバスイッチ、ここでは主としてスイッチ本体を指すが、バッファも入れて考える場合もある
- Router:パケットを転送するためのハードウェア全体を指す
- WH, Virtual Channel:この授業のもっとあとで紹介するのでここでは深く追求しないでよい
- Non-blocking, blocking:出力ポートが重ならなければ、衝突が起きないのがノンブロッキング、出力ポートが重ならなくてもスイッチ内部で衝突するのがブロッキング
- HOL conflict:出線競合、出力ポートが重なることで起きる衝突

Homework 2

- Your computer uses PCIe gen2 x 8.
  1. How much maximum bandwidth can be used ?
  2. You want to improve the bandwidth.
  2-1. When you use PCIe gen2 x 16, how much maximum bandwidth can be used?
  2-2. You changed the bus to PCIe gen3 x 8, how much maximum bandwidth can be used?

  Just a simple calculation. You will spend only about 3 minutes.

OK. This homework is too simple. Maybe you will spend 3 minutes.