

**NAPIER UNIVERSITY**  
**SCHOOL OF COMPUTING**

**FIRST DIET (SEMESTER TWO) EXAMINATION**

**SESSION 2000-2001**

**MODULE: EE42005**

**COMPUTER ENGINEERING**

**DATE:**

**DURATION: 3 HOURS**

**START TIME:**

**EXAMINER(S)**

**DR. A. ARMITAGE  
DR W.BUCHANAN**

**QUESTION PAPER DATA**

**Number of pages - XX  
Number of questions - 6  
Number of sections - ONE**

**INSTRUCTION TO CANDIDATES**

**Attempt any FOUR questions.**

**PLEASE READ THE FULL INSTRUCTIONS BEFORE COMMENCING WRITING**

1. (a) Explain the main types of cache architecture, and contrast their operation.
- (b) Outline the states of the MESI protocol which is used in cache control, and highlight the requirement for it, and for each of the states.

**Marks awarded:**

- Define each of the cache types, with an outline of each [6].
- Expanded discussion of a look-through cache, outlining its operation [2].
- Expanded discussion of a look-aside cache, outlining the concept of a cache hit or miss [2].
- Expanded discussion of an associate cache [1].

**SAMPLE ANSWER**

(a) The main cache architectures are:

- **Look-through cache.** In a look-through cache, the system memory is isolated from the processor address and control buses. In this case the processor sends a memory request directly to the cache controller, which then determines whether it should forward it to its own memory or the system memory. The cache controls whether the processor address contents are latched through to the DRAM memory and whether the contents of the DRAM's memory, is loaded onto the processor data bus (through the data transceiver). [3]
- **Look-aside cache.** A look-aside cache is where the cache and system memory connect to a common bus. System memory and the cache controller see the beginning of the processor bus cycle at the same time. If the cache controller detects a cache hit, then it must inform the system memory before it tries to find the data. If a cache miss is found, then the memory access is allowed to continue. [2]
- **Write-through cache.** With a write-through cache, all memory address accesses are seen by the system memory when the processor performs a bus cycle. [1]
- **Write-back cache.** With a write-back cache, the cache controller controls all system writes. It thus does not write the system memory unless it has to. [1]
- **Direct mapped cache.** When the contents of a new memory location have to be stored in the cache SRAM, there is only one possible place for it to be stored. [1]
- **Set associative cache.** When the contents of a new memory location is to be stored, there is more than one place for it to be stored in the set associative cache environment. The number of places is referred to as ways. For example, in a two-way set associative cache there are two possible places; in a four-way set associative cache there are four possible places. [2]

(b)

**Marks awarded:**

- Define the problem of more than one cache [1].
- Outline that it occurs with a level-1 and level-2 cache [1], or with an on-chip cache [1].
- Define that MESI synchronizes cache operations [1].
- Basic definition of each of the four states [4].
- For Modified: define that the cache line is marked as available only in a single cache [1].
- For Exclusive: define that new data is written to a cache location then the system memory is not updated automatically (write-back) [1], and on a multicache system, an exclusive cache line is stored on only one of the caches in a system [1].
- For Shared: define that a shared cache line always contains the most up-to-date value [1]; Write accesses to a shared cache line are always switched through to the external data bus independently of the cache strategy (write-through, write-back), so that the shared cache lines in the other caches are invalidated [1].
- For Invalid: A cache line marked as invalid is logically not available in the cache [1]; Every access to an invalid cache line leads to a cache miss [1].

**SAMPLE ANSWER**

A major problem occurs when there is more than one cache [1]. This can happen when there is an L1-cache and an L2-cache [1], or when there is more than one processor, each of which has an on-chip cache [1]. The MESI (modified, exclusive, shared, invalid) protocol implemented by the Pentium is used to synchronize cache operations [1]. The four states defined in the protocol are:

- **Modified (M)** – in this state the data in the cache is more recent than the corresponding location in memory. The cache line is then marked that it is available only in a single cache of the complete system. [2]
- **Exclusive (E)** – in this state the data in the cache is the same as the corresponding memory location. If new data is written to a cache location then the system memory is not updated automatically (write-back). On a multicache system, an exclusive cache line is stored on only one of the caches in a system. Thus it can be read and overwritten without the need for an external bus cycle. [3]
- **Shared (S)** – in this state the data in the cache is the same as the corresponding location in memory. If new data is written to this cache location, system memory is updated at the same time (write-through). The shared cache line can be stored within other caches in the system; it is – as the name suggests – shared with a number of other caches. A shared cache line always contains the most up-to-date value; in this way, the cache always services read accesses. Write accesses to a shared cache line are always switched through to the external data bus independently of the cache strategy (write-through, write-back), so that the shared cache lines in the other caches are invalidated. The address given out during the bus cycle is used as an external inquiry cycle for invalidating the cache lines in the other caches. At the same time, the contents of the main memory are also updated. The write operation in the local cache itself only updates the contents of the cache; it is not invalidated. [3]
- **Invalid (I)** – in this state data in the cache location is not most recent data, or a flush has taken place. A cache line marked as invalid is logically not available in the cache, which could be because the cache line itself is empty or contains an invalid entry, that is, not

updated. Invalid or empty tag entries also cause cache lines to be marked as invalid. Every access to an invalid cache line leads to a cache miss. [3]

2. (a) **Contrast, by outlining the basic operation of accessing DRAM memory, how FPM DRAM, EDO DRAM, Burst EDO DRAM, and SDRAM improve the speed of DRAM accesses.**

**Marks awarded:**

- Basic outline of how DRAM is accessed with the enhanced methods [3].
- FPM. Define that it takes 5 cycles to access memory [1]; Define 5-3-3-3 operation [1].
- EDO. Define that the column address is not deactivated [1]; Define 5-2-2-2 operation [1].
- BEDO. Define uses of a pipeline state and counter [1]; Define 5-1-1-1 operation [1].
- SDRAM. Define burst mode [1]; Define synchronized by the system clock [1]; Contrast with asynchronous transfers [1].

### **SAMPLE ANSWER**

DRAM uses an array of rows and columns. When accessing the memory, the memory management unit:

- Reads data by first activating the appropriate row of the array.
- Activates the appropriate column, then validates the data and transfers the data to the system.
- The column is then deactivated, which introduces an unwanted wait state during which the processor has to wait for the memory to finish the transfer.
- The output data buffer is then turned off, ready for the next memory access.

[3]

- FPM. FPM takes five clock cycles to access the memory. In burst mode the timing is 5-3-3-3, which means it takes five clock cycles to read the first element of data (with four wait-states), with the next three elements each taking three clock cycles. [2]
- EDO. Allows for faster accesses, as it does not require the column to be deactivated and the output buffer to be turned off before the next data transfer starts. It can thus give a burst timing of 5-2-2-2. [2]
- Burst extended data out (BEDO) DRAM improves the EDO DRAM principle by including a pipeline stage and a 2-bit burst counter. BEDO eliminates the wait states by accessing further data and can achieve system timings of 5-1-1-1. [2]
- Synchronous DRAM. Most memory accesses in the PC are sequential, thus SDRAM is designed to fetch all the bits in a burst as fast as possible. For this, an on-chip burst counter allows the column part of the address to be incremented very rapidly, which helps speed up retrieval of information in sequential reads. The memory control thus provides the initial location of the block and its size, and the SDRAM provides the data as fast as the CPU can read them. This output process is synchronized by the system clock (typically at 100MHz). This is an important feature, as previous memory systems (such as FPM

DRAM and EDO DRAM) have used asynchronous transfers (stop–start). After the data transfer has been set up, the SDRAM data burst can transfer at a rate of up to 100MHz, and thus uses a 10-ns clock transfer period. This gives a burst rate of 5-1-1-1. [3]

- (b) Explain the architecture and operation of Direct RDRAM, and highlight the main design decisions that must be taken.

(13)

Total Marks [25]

### Marks awarded:

- Define the controller with a 16-bit channel [1]; Define RSL logic for loading and fan-out [1].
- Define Direct DRRAM devices connecting to a 16-bit channel [1]; Define ground plane on one side and signals on the other [1].
- Show diagram with channels of different rates [1].
- Define termination of signals for matching [1]; define 1.8V termination with pull -up resistor [1];
- Define differential mode signals and reason for using these [2].
- Show method of using clocks to and from the master (top diagram in Figure Q2.1) [2].
- Show basic diagram of clock signals and how they are routed (bottom diagram in Figure Q2.1) [2].

### SAMPLE ANSWER

A direct Rambus channel includes:

- **One or more direct DRAMs.** These connect to a common bus, and can include microprocessors, memory, DSP devices, graphics processors and other fast transfer devices. Data is handshaked using Rambus signaling logic (RSL), where each RSL signal wire has equal loading, and fan-out is routed parallel to each other on the top surface of a PCB, with a ground plane directly underneath it. [2]
- **A controller.** The controller is located at one end with a parallel termination at the other end. In between the RDRAMs are distributed along the bus. The channel is 16 bits wide and uses a small number of very-high-speed signals to carry all address, data and control information at up to 800MHz, which should increase to over 1GHz with improvement in PCB design. It is responsible for generating requests, controlling the flow of data and keeping track of the direct RDRAM states and refresh. [2]

Figure below shows that the 16-bit channel gives a transfer rate of 1.6GB/s. Higher data rates, such as 3.2GB/s and 6.4GB/s, are possible with parallel multiple channels. This new architecture allows the system bus to operate at speeds up to 133MHz. [1]

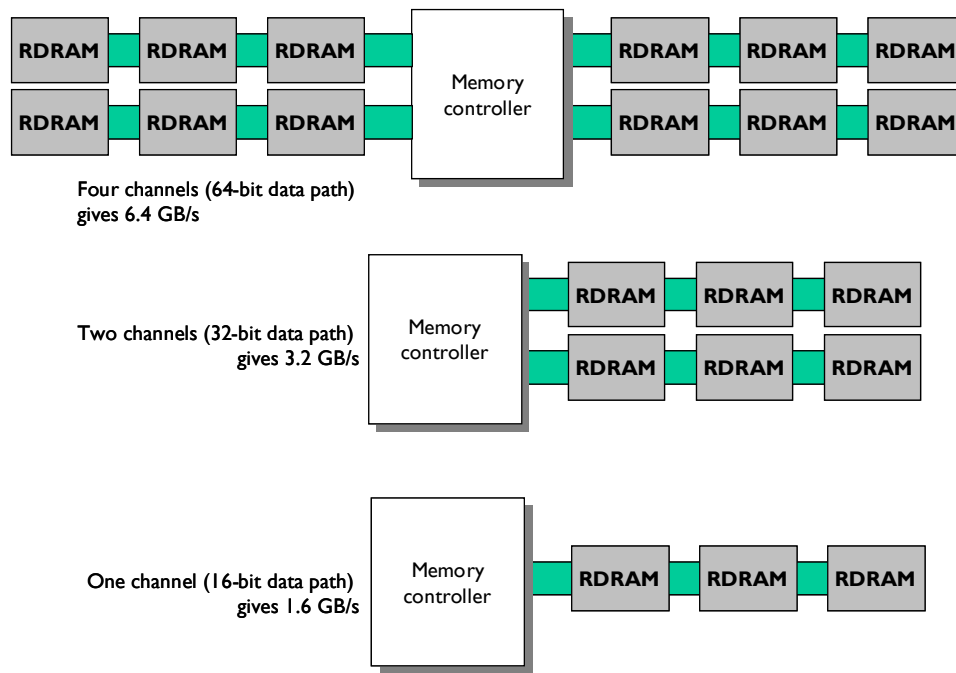


Figure Q2.1 RDRAM configurations

To achieve high clock speeds, all the signal lines must be properly terminated at the other end of the controller. This consists of a pull-up resistor connected to a 1.8V supply rail. These resistor values should match the impedance of the channel so that no reflections occur. [2]

The channel uses differential clocks (which consist of two lines that are not ground). Differential signals improve noise immunity and reduce jitter. The clocks used are clock-to-master (CTM and CTMN) and clock-from-master (CFM and CFMN), as illustrated in Figure 12.8. This clock is 400MHz. It can be seen that the signals for the RIMMs (RDRAM DIMM modules) run along and through each of the modules. Most of the signals run from the controller to the terminator. [2]

At A, the differential 400MHz is generated by a separate clock chip. This clock is then connected sequentially to each of the RIMMs (B). When it reaches the controller, it loops back out (D) and is changed from CTM to CFM, which is then routed back to all the RIMMs, until it is terminated (E). The CTM differential clock operates as the direct RDRAM transmit clock (TCLK) and the controller receive clock. All read data is aligned to the CTM clock. The CFM clock operates as the direct RDRAM receive clock (RCLK) and the controller send clock. All write data, as well as column and row packet data, are aligned to the CFM clock. [2]

As illustrated in the lower part of Figure Q2.2, data and the corresponding clock are aligned and always flow in the same direction through the Rambus channel. This allows the controller and direct RDRAMs to be able to operate at high speed, at 800MB/s (two bytes at 400MHz) in each direction (giving 1.6GB/s). [2]

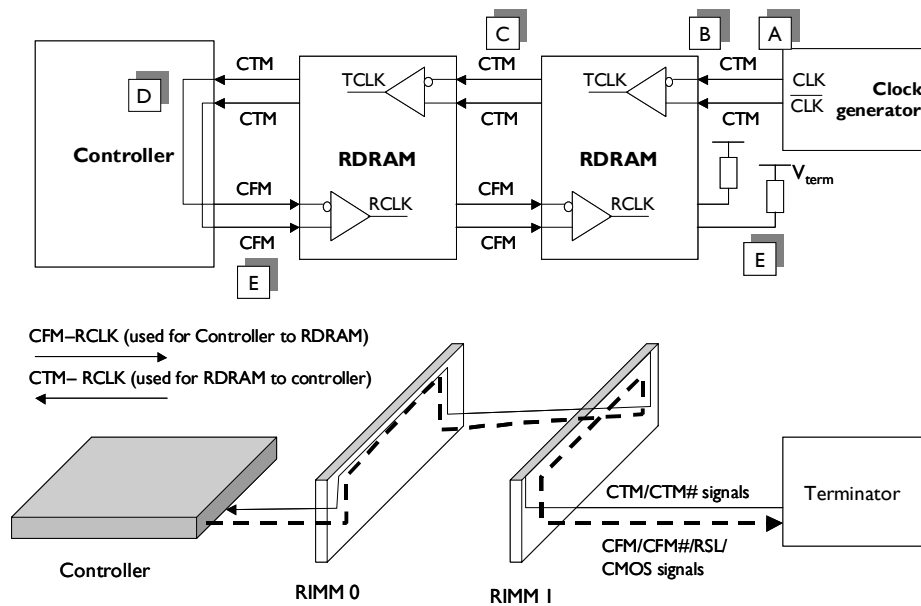


Figure Q2 RDRAM connections

3. (a) Contrast the architecture of hub-based architecture (such as the 840 chipset) with traditional north/south bridge architecture (such as with the 430 chipset). Outline the main signal lines for north/south bridge architecture.

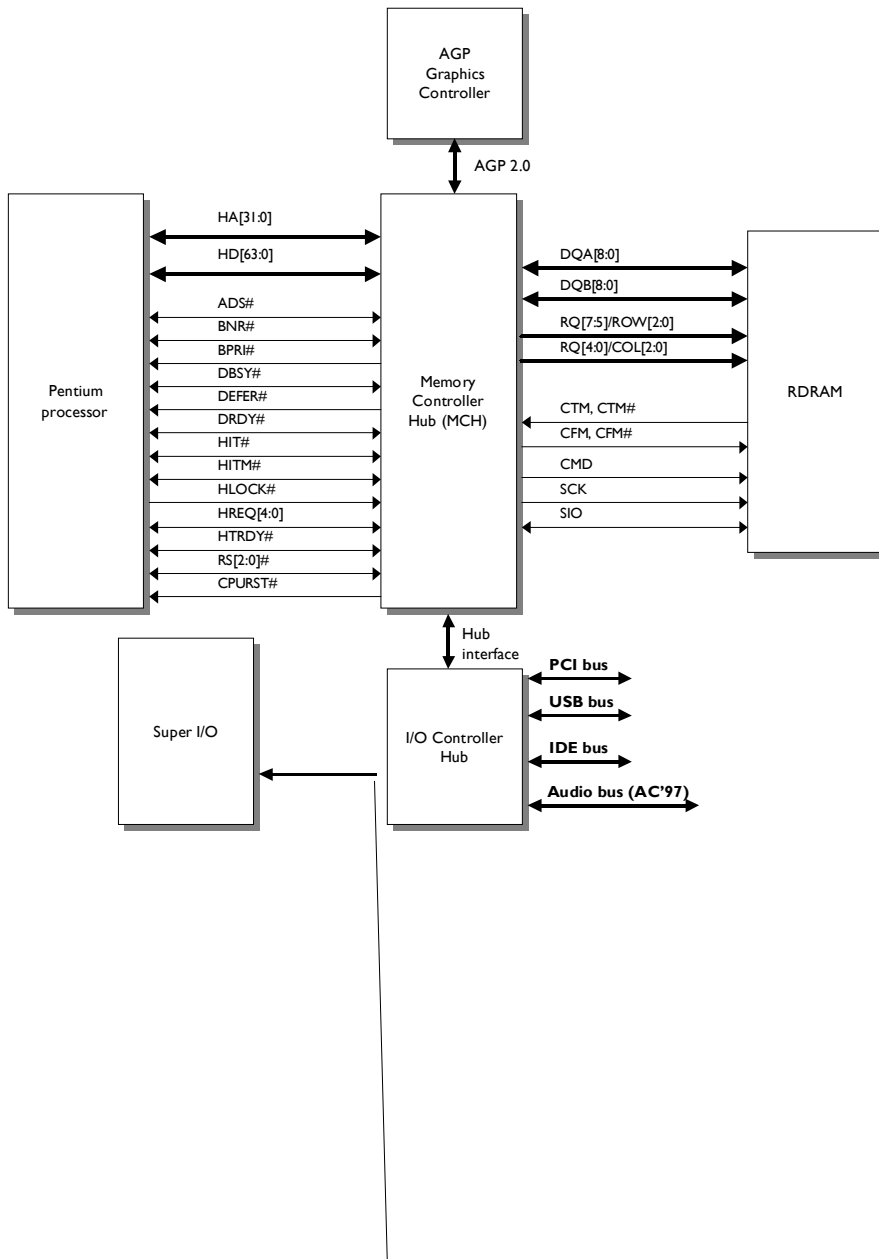
(10)

**Marks awarded:**

- Sketch the main component parts of the hub-based systems [2].
- Sketch the main component parts of the north/south bridge-based systems [2].
- Address and data bus sizes should be shown [1].
- Hub interface to memory lines should be outlined and contrasted with DRAM interface [2].
- LPC interface should be highlighted, and the reasons for it [2].
- Outline north/south bridge usage of legacy interfaces [1].

**SAMPLE ANSWER**

Figure Q3.1 and Figure Q3.2 shows an outline of the two architectures. *There is no need for the student to go into any great detail on the actual signal lines which connect between the devices.*



- (b) For a PC with PCI, AGP, SCSI, ISA and RDRAM interfaces, derive the data transfer requirements for each of the interfaces. State all assumptions made. How does the data rate vary for different implementation of these interfaces

(15)

Total Marks [25]

**Marks awarded:**

- Define ISA is 8MHz [1]. Define it transfers 16 bits at a time [1].
- Define PCI transfers at 33MHz and 32 bits [1]. Calculate 132MB/s [1].
- Define PCI uses a multiplexed bus [1]. Define that the rate is likely to be only 67MB/s [1]. Define burst mode [1], and how it operates [1]. Define 64 bit transfer and calculate [1]
- Define AGP based on PCI bus, but with address pipelining [1]. Calculate basic rate [1]. Define  $\times 2$  and  $\times 4$  rates [1].
- Define basic transfer at 8 bits giving 5MB/s [1]. Define that SCSI now uses 10MHz clock, which gives 10MB/s [1]; Define SCSI -II (Wide) which uses 16-bits at 10MHz [1]. Define SCSI-II (Fast/Wide) to give 40MB/s [1].

**SAMPLE ANSWER**

1. ISA, 8MHz at 16 bits gives 16MB/s [1]. No variation [1].
2. PCI, 33MHz at 32 bits gives 132MB/sec [2]. This is not really possible because of the multiplexing of the address and data lines, thus a rate of 67MB/sec is more likely [2]. It is possible to transfer 64 bits at 50MHz giving 400MB/s [1]. A faster rate is possible in burst mode where the address is placed on the address bus, and then the data is bursted sequentially [2].
3. AGP. Based on the PCI bus, but uses address pipelining [1] to transfer the address thus the transfer rate will be 132MB/s [1]. In  $\times 2$  and  $\times 4$  mode, twice and four times the transfer rate is achieved as the system clock is multiplied [1].
4. SCSI. SCSI-I transfers at 8 bits at a time giving 5MB/s [1], whereas SCSI -II uses a 10MHz clock and transfers 8 bits at a time thus the rate is 10MB/s [1]. SCSI-II (Wide) uses a 16-bit address bus at 10MHz giving a rate of 20MB/s [1]. SCSI-II (Fast/Wide) uses a 16-bit address bus with a 20MHz clock giving 40MB/s [1].

4. (a) Explain how a SCSI device uses the free-bus, arbitration and selection phases to gain access to the bus.

(12)

**Marks awarded:**

- Free-bus. Define the free-bus state [1]. Identify signals that define the state [1]. Define that any unit can capture the bus [1].
- Arbitration. Define the activation signal [1]. Define how unit puts its address on the bus [1]. Define delay [1]. Define the selection signal activation [1].
- Selection. Define the OR operation [1]. Give example of it [1]. Show activation of bus and select signals [3].

**SAMPLE ANSWER**

- **Free-bus.** In this state, there are no units that either transfer data or have control of the bus. It is identified by deactivate  $\overline{SEL}$  and  $\overline{BSY}$  (both will be high). Thus, any unit can capture the bus. [3]
- **Arbitration.** In this state, a unit can take control of the bus and become an initiator. To do this, it activates the  $\overline{BSY}$  signal and puts its own ID address on the data bus. After a delay, it tests the data bus to determine whether a high-priority unit has put its own address on the bus. If it has, then it will allow the other unit access to the bus. If its address is still on the bus, then it asserts the  $\overline{SEL}$  line. After a delay, it then has control of the bus. [4]
- **Selection.** In this state, the initiator selects a target unit and gets the target to carry out a given function, such as reading or writing data. The initiator outputs the OR value of its SCSI-ID and the SCSI-ID of the target onto the data bus (for example, if the initiator is 2 and the target is 5 then the OR-ed ID on the bus will be 00100100). The target then determines that its ID is on the data bus and set the  $\overline{BSY}$  line active. If this does not happen within a given time, then the initiator deactivates the  $\overline{SEL}$  signal, and the bus will be free. The target determines that it is selected when the  $\overline{SEL}$  signal and its SCSI ID bit are active and the  $\overline{BSY}$  and  $\overline{I/O}$  signals are false. It then asserts the  $\overline{BSY}$  signal within a selection abort time. [5]

- (b) Outline, giving an example, how arbitration is implemented on the PCI bus. Explain how the PCI support the locking of targets, and also why are locks necessary?

(13)

Total Marks [25]

**Marks awarded:**

- Define the operation of the REQ and GNT signals for arbitration [2].
- Define the operation of the FRAME and IRDY signals [3].
-

- Define that a resource lock, allows a master to ‘lock’ a specific target until the master has completed its accesses [1]. By locking a target no other master will be able to access the target until the master that locked it is finished with its series of accesses. [1]
- Show the operation of the LOCK and FRAME signals [4].

### SAMPLE ANSWER

Figure Q4 shows that two devices are requesting the buses at the same time. They are referred to as device-a, and device-b. The sequence is:

1. PCI clock – All signals are sampled on the rising edge of the PCI clock.
2. In this example REQ#-a was asserted prior to the 1st clock.
3. Device-a is granted access to the PCI bus when GNT#-a is asserted.
4. Device-a’s transaction begins when FRAME# is sampled active , (low), on the rising edge of clock. Device-a leaves REQ#-a asserted since it wants to perform another transaction.
5. The arbiter decides that device-b should get the buses next, it deasserts GNT#-a, and asserts GNT#-b.
6. Device-a completes its transaction, FRAME# and IRDY#, (not shown), are inactive, and device-b now owns the buses.
7. Device-b completes its transaction. The arbiter has granted the buses to device -a again since its REQ# signal is still active.

[7]

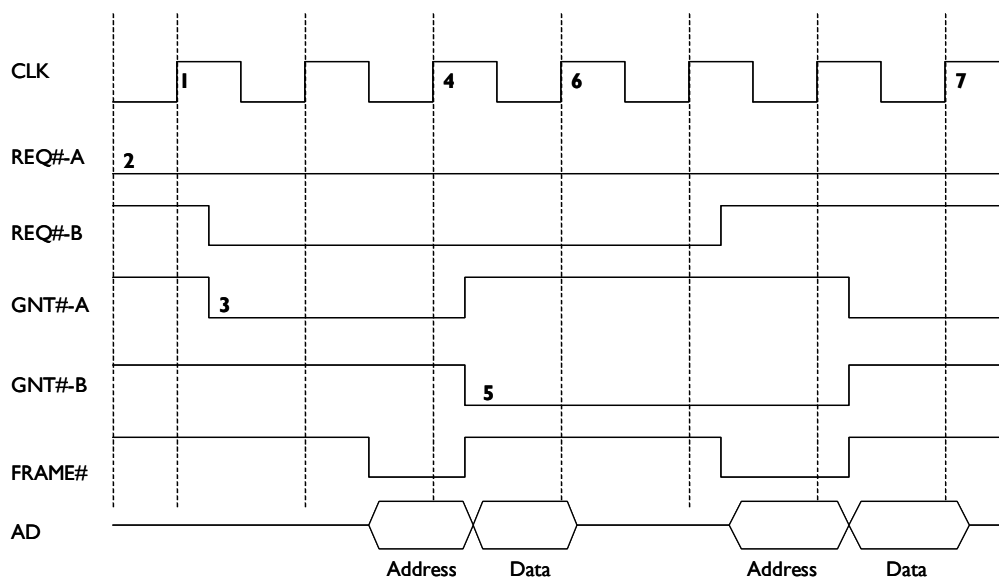


Figure Q4 Arbitration

A resource lock, allows a master to ‘lock’ a specific target until the master has completed its accesses. By locking a target no other master will be able to access the target until the master that locked it is finished with its series of accesses.

PCI allows exclusive access while allowing other masters to access targets other than the locked targets. The sequence is:

1. PCI targets that support exclusive accesses sample LOCK# when sampling the address.
2. If LOCK# is asserted during the address phase, the target of the transaction will not mark itself as locked (since there is already another exclusive access taking place).
3. If LOCK# is deasserted during the address phase, then the target of the transaction marks itself as locked.
4. The target stays locked until FRAME# and LOCK# are deasserted.

A true lock, (a target is locked for several transactions), occurs when LOCK# is de-asserted during the address phase and asserted on the following clock.

[6]

