PART IIA 2006
3F5: Computer and network systems
Principal Assessor: Dr A Gee

Datasheet: None

Engineering Tripos Part IIA THIRD YEAR

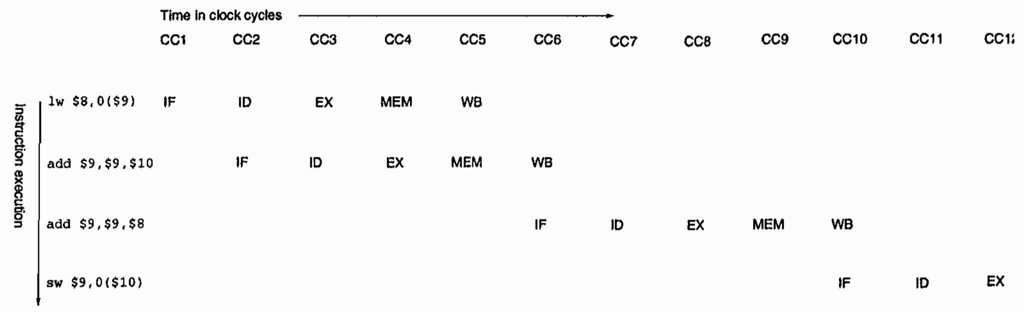
Module 3F5: Computer and Network Systems

Solutions to 2006 Tripos Paper

1. **Pipelining, superscalar operation, simultaneous multithreading**

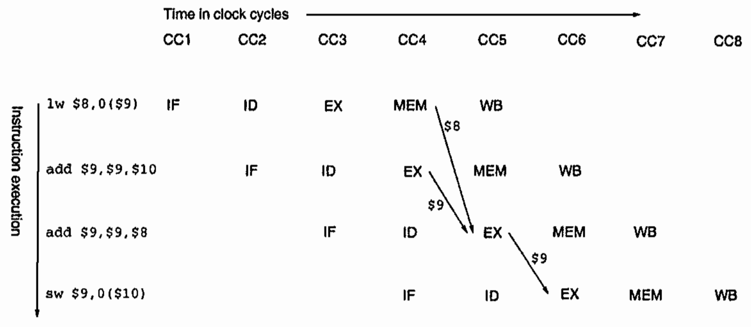
(a) The term *hazard* is used to describe dependencies between instructions which disrupt the operation of a pipelined datapath. *Data hazards* occur when an instruction requires data before a previous instruction has written it to the register file. *Branch hazards* occur when the address of the next instruction is required (for instruction fetching) before an earlier conditional branch instruction has been evaluated. [20%]

(b) (i) The critical data hazards are between the two `add` instructions and between the second `add` and the `sw`. If there is no data forwarding, the pipeline must stall for three clock cycles between these instructions, since data is read from the register file at pipe stage 2 but written at pipe stage 5.



Time in clock cycles
CC1 CC2 CC3 CC4 CC5 CC6 CC7 CC8 CC9 CC10 CC11 CC1:
Instruction execution
lw $8,0($9) IF ID EX MEM WB
add $9,$9,$10 IF ID EX MEM WB
add $9,$9,$8 IF ID EX MEM WB
sw $9,0($10) IF ID EX

The instructions therefore take 14 clock cycles to finish executing (or 13 ignoring the final WB, which doesn't do anything). (ii) With data forwarding, the second `add` can receive forwarded values for $8 and $9 from the appropriate pipeline registers. The `sw` can have the final value of $9 forwarded to its EX stage.

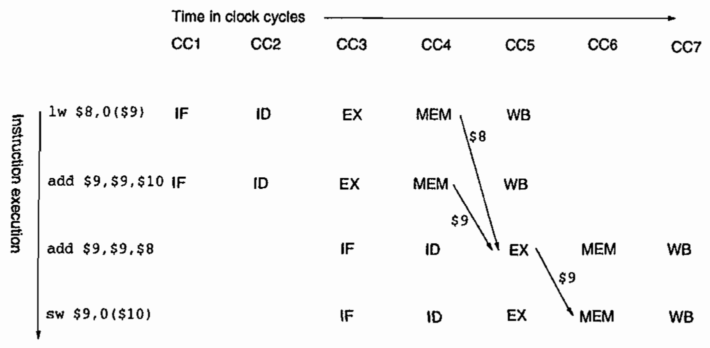


Time in clock cycles
CC1 CC2 CC3 CC4 CC5 CC6 CC7 CC8
Instruction execution
lw $8,0($9) IF ID EX MEM WB
$8
add $9,$9,$10 IF ID EX MEM WB
$9
add $9,$9,$8 IF ID EX MEM WB
$9
sw $9,0($10) IF ID EX MEM WB

There are no stalls and the instructions take only 8 clock cycles to finish executing (or 7 without the final WB). [20%]

(c) (i) The pipeline registers need to be widened to hold the intermediate states of two instructions at a time. The path to instruction memory needs be widened to deliver 64 bits (two instructions) at each clock cycle. The decoding logic at the ID stage needs to examine two opcodes at a time. The register file still holds the same number of registers, but needs further inputs and outputs: we might need to read four registers at a time (two for an arithmetic/logic operation and two for a `sw`) and write two registers (one for the arithmetic/logic result and one for a `lw`). We'll need an extra ALU at the EX stage, so we have one for branch target calculation, one for an arithmetic/logic operation or branch comparison and one for load/store address calculation. The MEM stage can stay as it is, since only one of the concurrent instructions is allowed to be a load/store. We'll also need extra hazard detection logic. [25%]

(ii) Superscalar operation brings more potential data hazards into play. If we execute the `lw` and first `add` concurrently, the $8 dependency between the `lw` and the second `add` becomes an issue: we'll need one stall. We can run the second `add` and the `sw` together, as long as we forward the `add` result straight into the memory input.



So the instructions now require only 7 clock cycles to finish executing. [10%]

(d) The modest performance gain in (c), admittedly contrived and considering a very small code segment, demonstrates how a superscalar pipeline's resources may not be fully exploited by a single process because of additional data hazards. If we run two independent threads simultaneously, though, there will be no hazards between the two threads and it should be much easier to issue two instructions at a time. For simultaneous multithreading (SMT), we need extra hardware to deal with each process's independent state. This means a separate program counter, a separate register file and a separate page table (implying, at the hardware level, a separate TLB) for each thread. It should be noted that SMT makes no difference if only one process is being run. Performance improvements become apparent when two or more CPU-hungry processes are run concurrently. One SMT performance worry comes from the memory system: the cache is now being shared concurrently (without time slicing) between multiple threads. If they're totally independent processes, we can expect more cache conflicts and more misses. If they're multiple threads cooperating on the same task in a shared memory framework, things might not be so bad. [25%]

2. **I/O hardware and software**

(a) In a synchronous bus, all data transfers happen at a set rate relative to a clock signal. The clock signal is distributed to all devices via a control line of the bus. In an asynchronous bus there is no clock, so every transfer must be acknowledged by the recipient before more data is sent.

Synchronous buses must run at the speed of the slowest device on the bus, since every device uses the same clock signal. Clock skew (the difference in time between seeing the same clock edge at different positions on the bus) limits the bus length. However, synchronous buses can run very fast, and are often used for processor-memory buses, since the devices communicating are close together and can run at high clock rates.

An asynchronous bus can cope with a wide variety of devices and data rates, and can be lengthened without worrying about clock skew. I/O buses are normally asynchronous: this includes Firewire and USB 2.0. To coordinate data transfers, an asynchronous bus uses a handshaking protocol. [25%]

(b) Polling, interrupt-driven I/O and direct memory access (DMA) are three different mechanisms for allowing the CPU to interact with I/O devices. Polling requires the least hardware: the CPU periodically checks to see whether the device is ready to send or receive more data, and handles the data transaction if necessary. The polling frequency must be high enough to satisfy the device's maximum data transfer rate. This can be tremendously wasteful of CPU time, especially for devices which are mostly idle. Polling may be used for low bandwidth devices which can tolerate low frequency polling, like mice.

Interrupt-driven I/O requires extra signal lines to interrupt the CPU whenever an I/O device requires attention. The CPU must still be involved in every bus transaction, so may still be heavily loaded when the device is active. But, in contrast to polling, there is no CPU load when the device is idle. Interrupt driven I/O may be used for relatively low bandwidth devices which are mostly idle, like printers.

DMA is the most expensive technique in terms of hardware, requiring a dedicated DMA controller on the bus. But it is the only viable technique for very high bandwidth devices, like hard disks, which might otherwise fully occupy the CPU with bus transfers. With DMA, the CPU hands control to the DMA controller, which deals with the individual bus transactions between the device and memory. Once the transfer is complete, the CPU is interrupted. The CPU then checks whether the transfer was completed successfully or whether there was an error. [25%]

(c) DMA poses problems for both virtual memory systems and caches. Starting with virtual memory, the DMA controller is supplied with a starting address and a number of bytes to transfer. But should the starting address be virtual or physical? If virtual, then the DMA controller will need extra hardware to store the necessary page table entries and perform the translations. If physical, then care must be taken to ensure that DMA transfers do not cross page boundaries, since contiguous virtual pages do not generally map to contiguous physical pages. Whichever approach is taken, the operating system must cooperate by not moving pages around while a DMA transfer involving that page is in progress.

Moving on to caches, there are two potential problems here. When transferring data from the I/O device to memory, there may be a copy of this chunk of memory in the cache. If the processor reads from this chunk, it will get the old value (from the cache) and not the new value (as updated by the DMA controller). Similarly, when DMA is used to transfer data from memory to the I/O device, and the cache is write-back, the DMA controller may transfer an old value from memory when there is a newer one in the cache. This is called the *stale data problem.*

There are three ways round this. One possibility is to route all DMA activity through the cache, though this is expensive and very wasteful of cache space, since the processor rarely needs to see the I/O data immediately and, in the meantime, useful data has been displaced from the cache. The second option is to have the operating system invalidate the cache for an I/O write or force write-backs for an I/O read. This sort of *cache flushing* requires minimal hardware support but is inefficient, since the whole cache is affected even if only one block overlaps with the DMA activity. The final, most complex option is to provide hardware mechanisms to selectively flush individual cache entries. [25%]

(d) The move to serial point-to-point networks arises because buses cannot keep up with the bandwidth of today's I/O devices. The clock rate of a parallel bus is limited by noise, stray capacitance, crosstalk and clock skew. Generally, if you want a very fast parallel bus, you'll need to make it physically short and limit the number of devices allowed to tap into it. This conflicts with the basic requirement than an I/O bus should be long and support as many I/O devices as necessary.

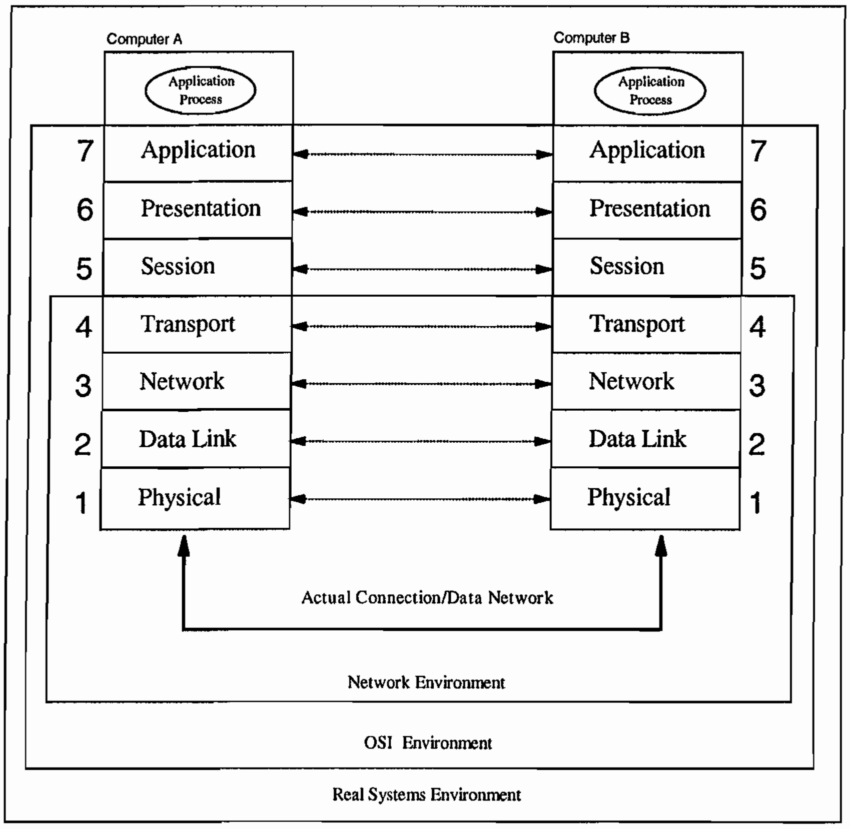
These problems are avoided by doing away with the shared, parallel bus, and replacing it with a switched point-to-point network. To keep the number of wires manageable, these networks are serial. But they can run very fast, since there are only two devices on each link — so less load, device noise and stray capacitance — and hardly any crosstalk. These new I/O networks also circumvent the need for bus mastering protocols. Data transfer is typically synchronous to an embedded clock.

Serial connections have the added advantage of requiring fewer wires. This means less clutter and hence better air flow/cooling inside computer cases.

The two obvious examples in recent PC hardware are the transition from parallel ATA to serial ATA for disk devices, and from PCI/AGP to PCI express for generic I/O devices and graphics cards. [25%]

3F5 2006 Cribs (answers are often longer then I expect from the candidates)

Q3. a) For successful data communication across a network, appropriate operating procedures must be established. They must be specified in detail and strictly adhered to by the sending data terminal (or computer) and any intervening switching centres. These procedures are called protocols. Many local area networks (LANs) interconnect data terminals (or computers) from the same manufacturer and operate using proprietary protocols. The need arose for communication between computers and terminals from different manufacturers. Open systems interconnect (OSI), to enable networks to be machine independent.
Computers may use different languages, data formats and operating systems; hence, the interface between user (application) programs, normally referred to as applications processes, and underlying communications services may be different. Development of the necessary specifications and protocols or OSI was undertaken by the international standards organisation (ISO). The ISO standards are based on a seven layer protocol known as the ISO reference model for OSI. Both the network dependent and application oriented (network independent) components of the OSI model are in turn implemented in the form of a number of protocol layers. The boundaries and processes for each layer have been selected based on experience gained from other standards in the past.



The OSI model can be broken up within this system into 7 separate layers, each with a clearly defined purpose and protocol.

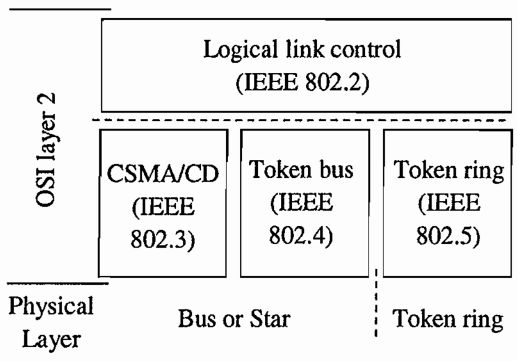
- Each layer is a service user to the layer below and a service provider for the layer above.
- Each layer is specified independently of the other layers;
- Each layer has a defined interface with the layer above and below.

As far as users are concerned, communication appears to take place across each layer. Each data exchange passes down to the bottom layer (the physical layer) at the sending terminal, crosses the network to the receiving terminal and then passes up again.

Data communication between layers is done through the addition and reading of headers on the data

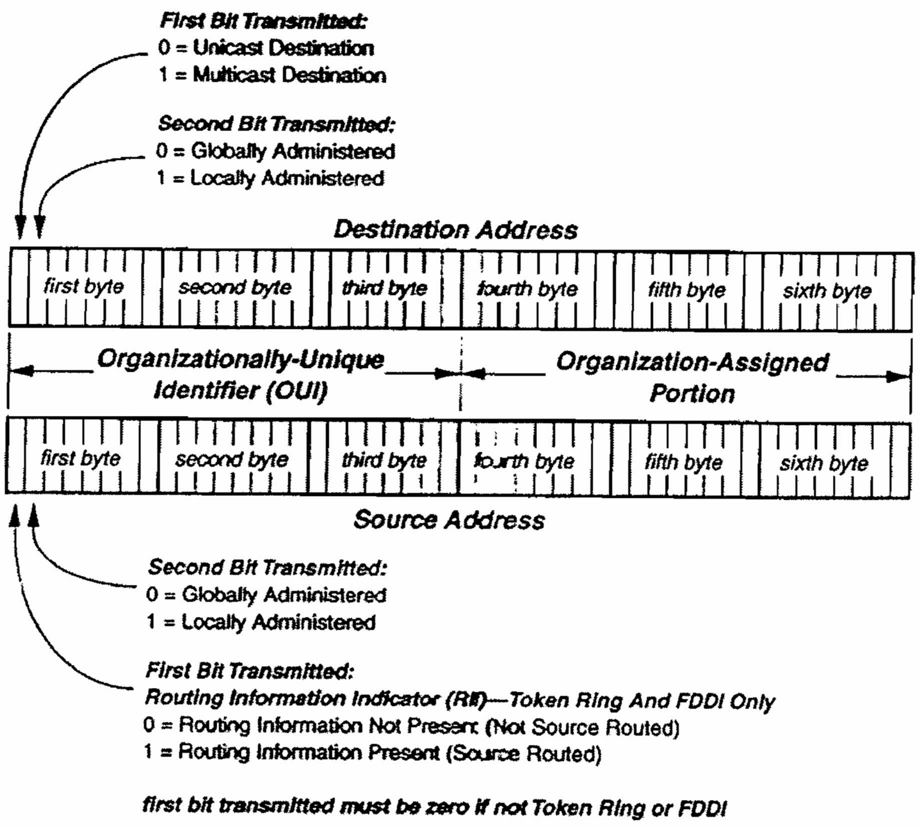
b) Layer 3: The network layer. This is concerned with the operation of the network between the terminals. It is responsible for establishing the correct connections between the appropriate network nodes, including network routing (addressing) and in some instances, flow control across the computer to network interface. Routing and network management at the packet level are done in this layer.

Layer 2: The data link layer. This provides error detection and correction for a link to ensure that the exchange of data is reliable. Provides error free data to the network layer. Adds control information to the data blocks and a frame check sequence (FCS) for error-checking or bits for synchronisation. Retransmits data if errors have occurred. The data-link layer is also responsible for determining the length of data segments to be sent to the physical layer. If the data provided by the network layer is too long, then it is broken into suitable sized packets. In the case of LANs, the data-link layer provides local area address management on top of that provided by the network layer through medium access protocols (MACs).



Layer 2 is split into the LLC and the MAC. This was to split the control of the LAN from the media dependant functions. The different types of LAN are characterised by their distinctive topologies. They all comprise a single transmission path interconnecting all the data terminal devices, together with the appropriate protocols (called the logical link control (LLC) and the medium access control (MAC)) to enable data transfer. The LLC and MAC split the data-link layer (Layer 2)

c) The key to MAC layer protocols is a uniform addressing structure so that LAN hardware can easily identify stations on the network and transmit packets between them.



First Bit Transmitted:
0 = Unicast Destination
1 = Multicast Destination
Second Bit Transmitted:
0 = Globally Administered
1 = Locally Administered
Destination Address
first byte
second byte
third byte
fourth byte
fifth byte
sixth byte
Organizationally-Unique Identifier (OUI)
Organization-Assigned Portion
first byte
second byte
third byte
fourth byte
fifth byte
sixth byte
Source Address
Second Bit Transmitted:
0 = Globally Administered
1 = Locally Administered
First Bit Transmitted:
Routing Information Indicator (RII)—Token Ring And FDDI Only
0 = Routing Information Not Present (Not Source Routed)
1 = Routing Information Present (Source Routed)
first bit transmitted must be zero if not Token Ring or FDDI

A MAC address is 48 bit long (6 bytes) and the address space is split into two halves:

- A *unicast address* identifies a single device or network interface. The source address is always unicast. Often referred to as *individual, physical* or *hardware addresses.*
- A *multicast address* identifies a group of logically related devices. Often referred to as *group* or *logical addresses.*

The first bit of the destination address defines if it is a unicast (0) or a multicast (1) address. Source addresses are always unicast so the first bit is zero except in token ring or FDDI, where is describes how the packet is routed. The second bit of the address defines whether the address is globally unique (0) or locally unique (1) to the LAN. Globally unique addresses are assigned by the manufacturer, whereas locally unique addresses are assigned by the LAN administrator, carefully! The 48-bit MAC address is divided into two parts. The first 24 bits constitute the organisationally unique identifier (OUI), which indicates which organisation (typically the manufacturer) is responsible for assigning the remaining 24 bits of the address. A MAC address is usually expressed in canonical byte form *aa-bb-cc-dd-ee-ff* where the first 3 pairs of bytes are the OUI and the last 3 pairs of bytes are set by the manufacturer. An address is read *aa* byte first.

The address evolved in layer 2 as LANs were originally only designed for local connection so a simple hardware address was used to identify all end users. Connection between LANs was originally managed by bridges which switch purely on the basis of the MAC address structure. As LANs got more complex and started to connect globally, then the MAC address became globally unique, but also restricted by the layer 2 functionality. This was alleviated by the inclusion of layer 3 functions which allow more network oriented processes and can be used to manage large networks. This also included the layer 3 address which is where addressing should be handled. Hence we have been left with the MAC address as a method of locating users locally. This is not a true OSI definition, however it allows us to uniquely identify hardware as well as manage network addressing in a separate manner.

d)

OSI								
7 6 5	X - windows	TELNET	FTP file transfer protocol	SMTP Simple mail transfer protocol	SNMP Simple network manage protocol	TFTP Trivial file transfer protocol	RCP Remote procedure call	NFS Network file server
4	TCP				UDP			
3	ARP	ICMP IP			RARP	Gateway protocols BGP EGP		
2	SNAP LLC (eg Ethernet LAN)			SLIP (PPP) Serial Line			Frame relay etc	
1	Physical network							

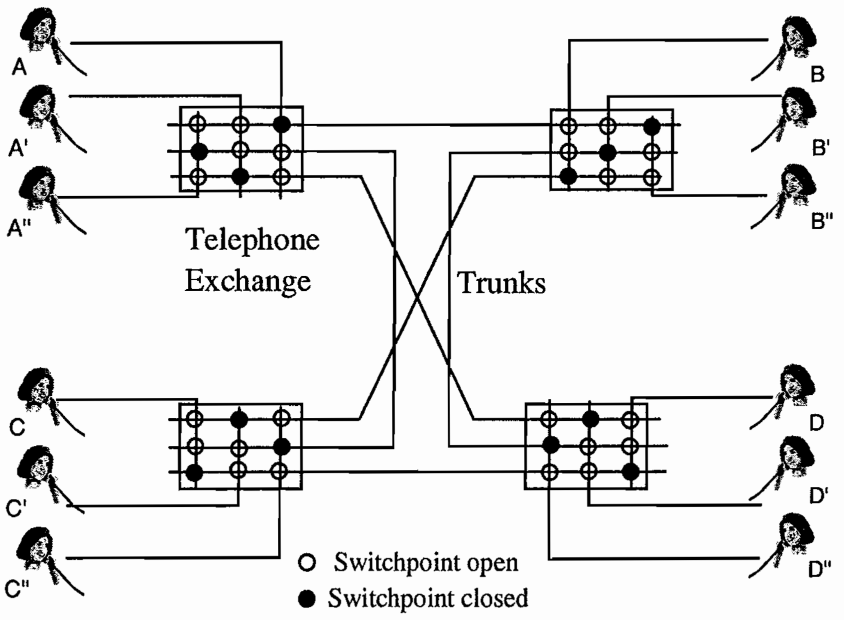
Mapping the destination to a local data-link address (ARP mapping) – The structure of the station identifier section of the IP address does not provide a simple mapping onto 48 bit data link addresses. It is not possible to determine the 48 bit data link address solely from the station identifier. Hence for packets destined for a locally connect network (such as the last hop port) must undergo a second look up process to find the destination station. In some instances this could be a separate look up operation in the ARP cache or continuation of the router look up process. Either way it will comprise one of three classes:

1. The packet is destined for the router itself. ie the destination IP address corresponds to one of the IP addresses of the router. This packet will be passed the higher layer entities in the router.
2. The ARP mapping for the indicated station is unknown. In this case, the router must initiate a discovery procedure (ARP request). This may take some time (it is a form of layer 3 flooding) and could result in the packet being dropped. This is not part of the fast path. ARP requests are not normally part of the steady state operation of a router.
3. The station is destined for a known station on the directly attached network. In this, very common case, the router successfully determines the mapping from the ARP cache and continues the routing process

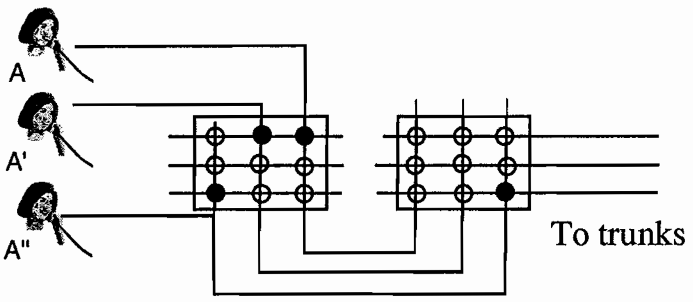
Q4 a) i) A circuit switched network is a good solution to a telecoms voice network with low numbers of subscribers. The trunks allow 1 call per connection to be made and offer full guaranteed connected service at the required latency and quality of service. The access to the trunks is done via circuit switches which connect subscribers as they request a connection. Each subscriber is connected to each trunk via a crossbar switch which makes the desired connection, whilst at the same time preventing other users from accessing the same trunk. The protocol used in the hardwiring of the switch will send the engaged tone to the second requester who is accessing the same trunk. This protocol is called SS7.

The problem with this network is that it does not allow full connectivity, as trunks can only take one call each. This is a circuit switched network with full availability. To have full connectivity, a mesh of trunks is required to connect all subscribers to all of the others at any time. This would require a lot of trunks (23) which is complicated and each subscriber must be capable of taking up to 11 calls simultaneously. This is expensive, cumbersome and inefficient.

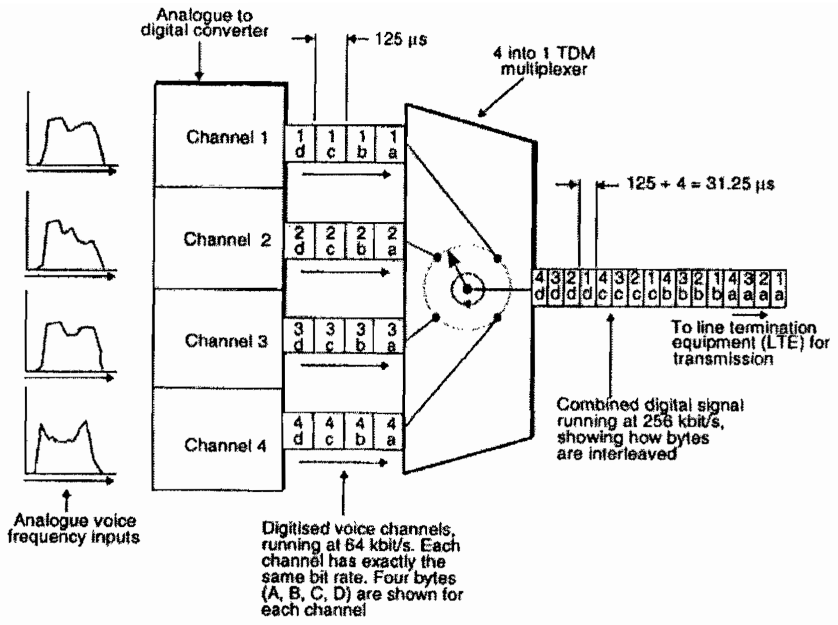
ii)



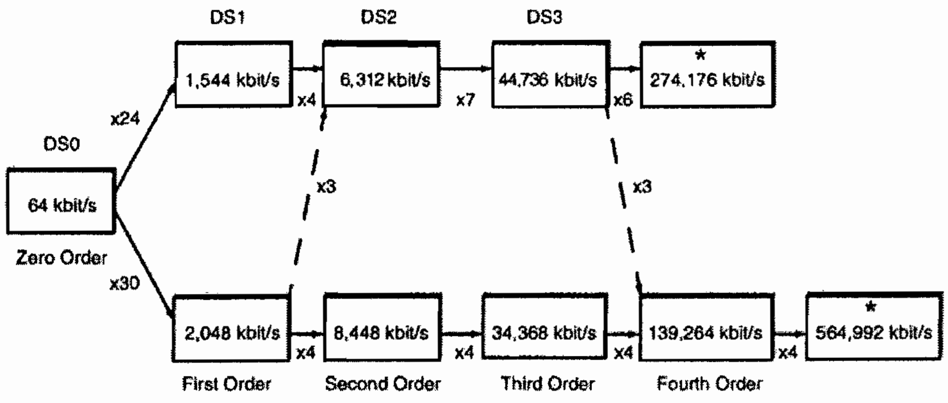
iii) Local switching between the groups of three subscribers can be done by modifying the switches at the exchange to allow connections between A and A' etc. This is done by adding extra switch elements to the crossbar which effectively isolate or bypass the local connection. What you want to avoid is having to block access to a trunk when a local call is being made.



b) Circuit switched networks work well and are ideally suited to voice communications, however they are limited in scale. As the number of subscribers and trunks increases so does their physical size and space requirements. Wire density is limited by space and Strowger exchanges became huge in order to house the switches. Hence it became necessary to multiplex data streams to allow multiple subscribers on each physical wire. This was further enhanced by invention of pulse code modulation (digital circuits) and the optical fibre (huge bandwidth). PCM converts 4kHz BW to 64kbit/sec digital stream, which avoids noise and other analogue problems. More importantly it allowed many PCM channels to be multiplexed together using Time Division Multiplexing (TDM) to increase the BW used in the transmission medium. Many 64kbit/sec channels could be byte interleaved onto a high capacity link.


Analogue to digital converter
125 µs
4 into 1 TDM multiplexer
Channel 1
Channel 2
Channel 3
Channel 4
125 ÷ 4 = 31.25 µs
To line termination equipment (LTE) for transmission
Combined digital signal running at 256 kbit/s, showing how bytes are interleaved
Analogue voice frequency inputs
Digitised voice channels, running at 64 kbit/s. Each channel has exactly the same bit rate. Four bytes (A, B, C, D) are shown for each channel

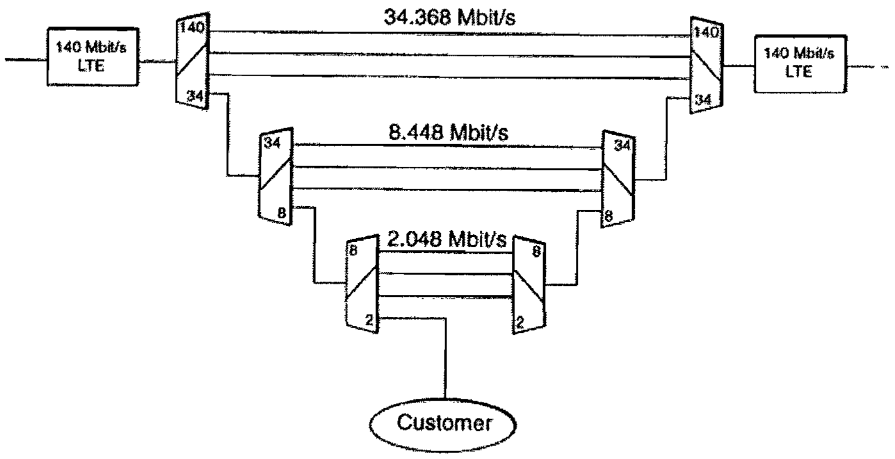
From TDM, a European standard arose where 30x64kbit/sec channels plus 2 extra channels were combined into one 2.048Mbit/sec link by byte interleaving. (1st order MUX). Higher orders of Mux were created as they became needed. These higher order Mux were created by bit interleaving the 2Mbit/sec channels.



* not recognised by ITU-T

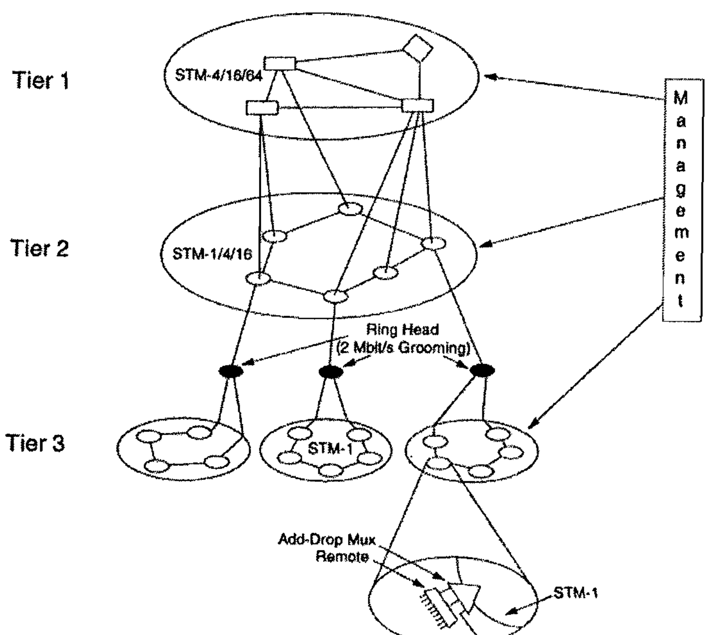
Each of the 2Mbit channels will be generated by different equipment each with its own clock. Hence they will have slightly different data rates. Before they are interleaved, they must undergo bit rate adaptation. All of the channels are brought to the same data rate by adding dummy information known as 'justification bits'. This is known as 'bit stuffing'.

c) The problem occurs when a single 2Mbit channel needs to be dropped or added from a higher order Mux. (Drop and Insert). All of the channels must be demuxed and then muxed again as the position of a channel is unknown in the higher orders. ie 140 to 34, 34 to 8 and then 8 to 2Mbit/sec. This is very expensive as it requires a lot of equipment.



There is also inadequate provision of network management, which is important when combining POTS and computer data. The aim of SDH was to correct the defects of PDH.

- Synchronisation. All elements of the system are synchronised to the same master clock. All tributaries are at a common rate before Mux.
- Pointers. Information about the muxed signals is transmitted at fixed intervals, which indicate the positions of units within the mux process. Hence any unit within the mux process can be identified and drop and insert processes can be done dynamically.
- Control and management. Time slots are put aside for a variety of tasks in ensuring the synchronicity of voice and data services.



1st basic SDH mux unit is the STM-1 (synchronous transport module) frame which is at a data rate of 155Mbits/sec. Higher mux is done by byte interleave by 4 (STM-4, STM-16). STM-1 signal is a repeated series of 125usec frames of 270x9 (2430) bytes. (Same as 1x64kbit/sec byte to avoid delays).

The key to SDH is that it is externally managed. All connections and bandwidth is allocated externally by pointers through management channels. This makes it possible to manage end to end connections without adding delays.

Assessors' remarks for question 1: This question was about pipelining, hazards, data forwarding, superscalar operation and simultaneous multithreading (SMT). In (a), all candidates were able to define the terms *data hazard* and *branch hazard.* In (b), the most common mistake was assuming that a register file would always forward a value written and read at the same clock cycle: this is not necessarily the case when there is no data forwarding. In (c)(i), candidates tended to write vague passages about multi-issue scheduling logic, instead of concentrating on the principal pipeline components. In (c)(ii), the most common mistake was overlooking the possibility of forwarding to the data memory input for the `sw` instruction. In (d), many candidates correctly identified thread independence, and hence fewer hazards, as the key point. Several also realised that cache conflicts would be a problem, though some seemed to think that SMT required multiple caches and hence some sort of cache coherency protocol.

Assessors' remarks for question 2: A popular essay-style question covering computer input/output. Most candidates had evidently prepared this topic well, producing detailed and accurate answers. As usual with essay questions, the best answers were distinguished by sound editorial judgement: they highlighted the key points and, just as importantly, omitted irrelevancies and trivia.

Assessors' remarks for question 3: This was a well answered question in most sections with a few missing the role of the layer split in the context of LANs. The assessor was a bit disappointed by the responses to section (d) as there was a bit of scope for interpretation. Most avoided that section with a few getting the role of ARP.

Assessors' remarks for question 4: This was a bit easy as it was mostly bookwork. Most candidates did quite well in the PDH and SDH sections. There were some very unusual answers to the switching section, but almost all missed the role and complexity of local switching.

Andrew Gee & Tim Wilkinson
May 2006

Part IIA 2006

Module 3F5: Computer and Network Systems

Numerical Answers

1. (b) (i) 14 clock cycles, (ii) 8 clock cycles.
   (c) (ii) 7 clock cycles.