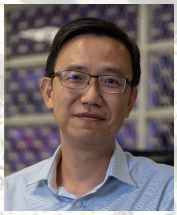
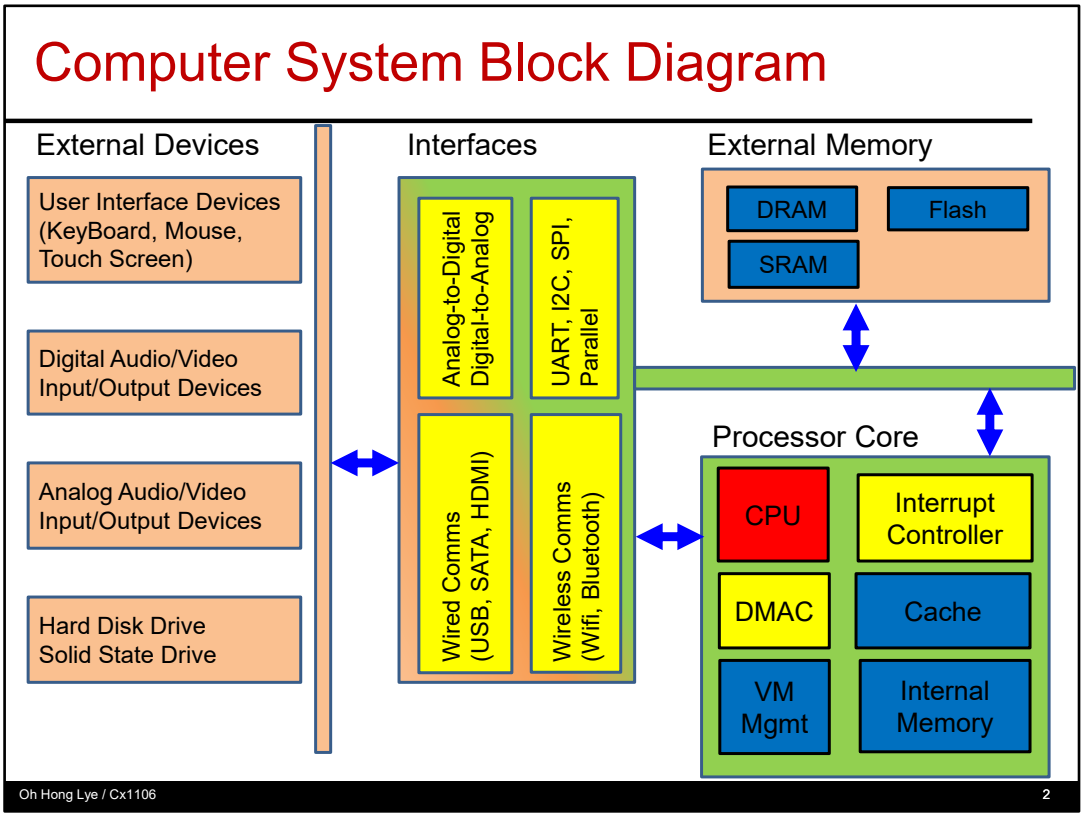
# Cx1106
# Computer Organization and Architecture

## Computer Systems Overview



Oh Hong Lye
Senior Lecturer
School of Computer Science and Engineering, Nanyang Technological University.
Email: hloh@ntu.edu.sg

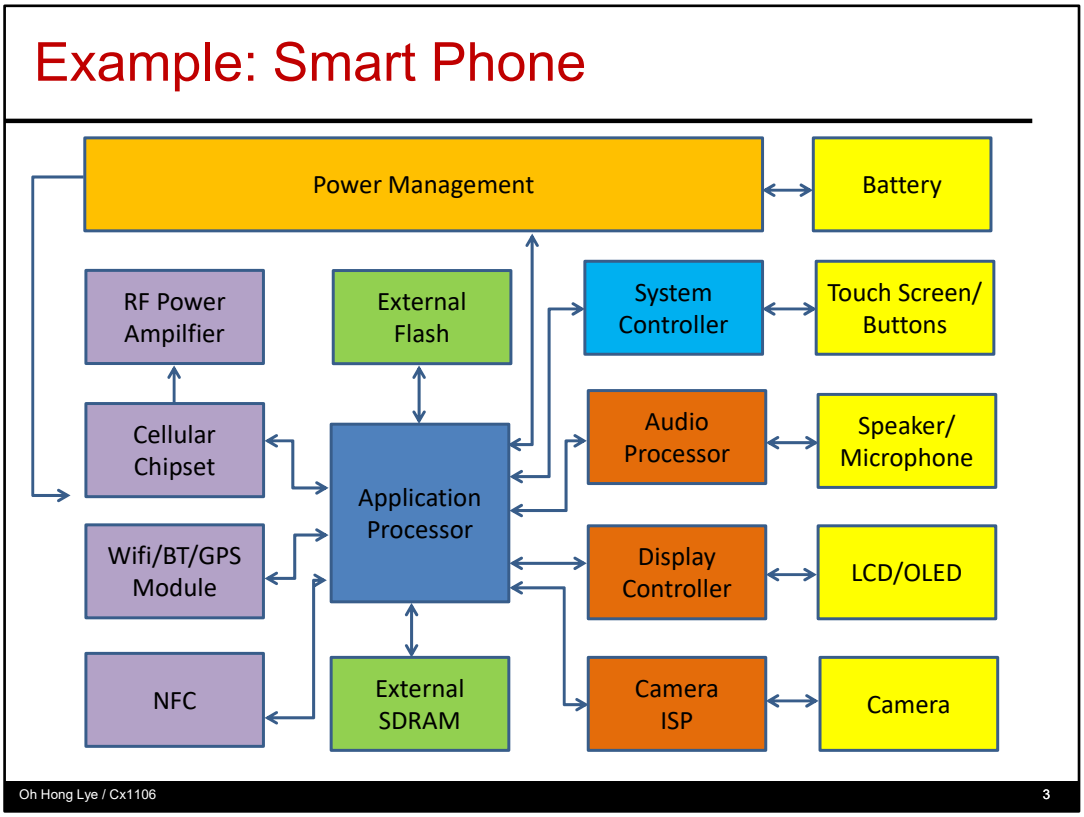
- Hi, Welcome to the second half of Cx1106 course.
- My name is Hong Lye, I'm your lecturer for this half of the course.
- If you have any questions after going through the slides and notes, please feel free to drop me an email at hloh@ntu.edu.sg




- In the first half of this course, you have been focusing mainly on the processor of a computer system.
- Now a computer system doesn't just consist of the processor alone, there are many other modules that formed up the system.
- There is the External Memory, which could be made of DRAM, SRAM or Flash memory.
    - This is where you store the code and data of your OS and applications.
- There is also the modules that enable the processor to interface to the external world.
    - It consist of external devices such as
        - Keyboard, Mouse and Touch Screen
        - Audio and Video devices
        - Mass storage such as the HDD and SSD.
    - These devices are connected to the processor via different interfaces
        - You have interface such as USB, ADC, UART, Wifi etc.
        - These functions could exist internal to the processor and is known as a peripheral, or it could exist as a separate module that is external to the processor.
- All these modules and devices are connected to the main processor via some sort of interfaces. Computer interface is a sub-topic which we will deal with in more details later.
- And the processor doesn't just consist of the CPU and internal memory as well, there are other modules such as
    - Direct Memory Controller, Interrupt controller, Cache, Virtual memory

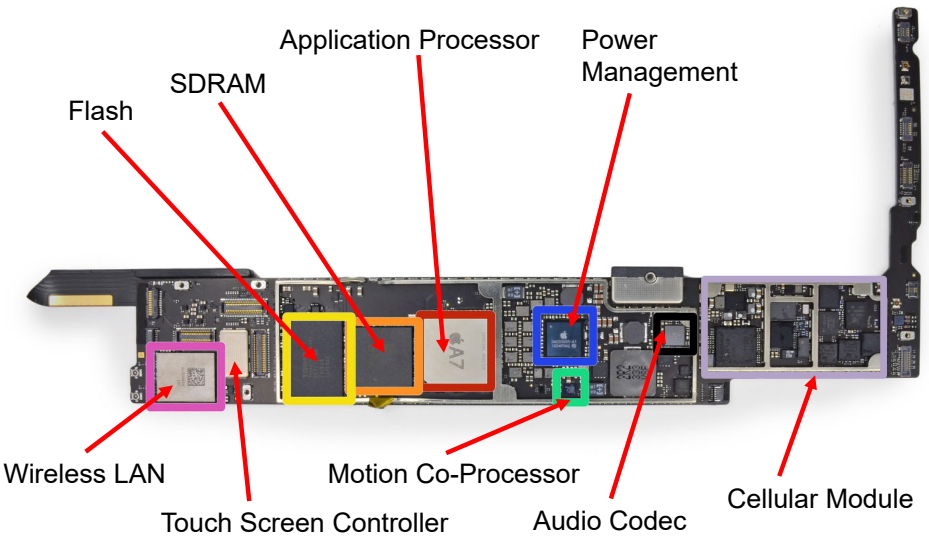
Management modules

- Essentially, we will be dealing with everything that can be found a typical computer system, topics that are beyond the CPU architecture that you learn during the first half.
- Don’t worry, we are only touching on the basics for most topics, you will be re-visiting a number of the topics again in your senior years.




- This slides shows the system block diagram of a smartphone, which is a very good example of a computer system.
- In the middle you have the application processor with its external memory
- There is a system controller which takes care of the touch screen and user inputs
- A few modules that deals with the audio, video and camera functions of the phone and the corresponding controllers for these modules.
- Over on the left side, these are the modules that is responsible for the data communication of the computer system, you have the Cellular, Wifi and NFC listed here.
- And lastly, you have the Power management module that manage the power rails and power source i.e. the battery of the system.
- If you ponder for a while, you will probably realise that modules such as the power management, wifi, camera ISP etc are actually quite complicated in design and probably would need their own processor to manage its function. You are right in that there is actually a processor in many of these modules.
- I did a count before and in a typical smart phone such as the one you see in this slide, there is easily 8-10 processors in there.

# iPad Air Main PCB

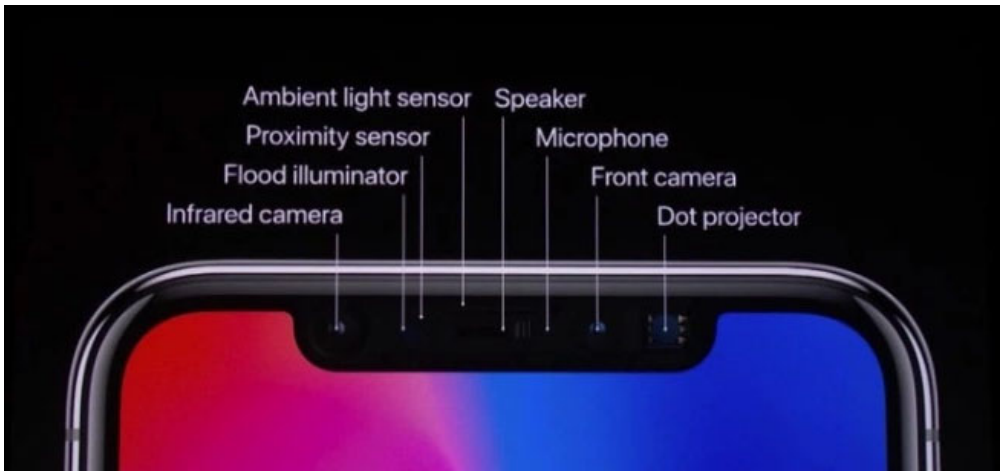



Source: https://www.ifixit.com/Teardown/iPad+Air+LTE+Teardown/18907



- Translating the system block diagram to an actual Printed Circuit Board, PCB in short, of a phone.
- What you see in this slide is the main PCB of a iPAD AIR.
- This is extracted from a website know as ifixit.com, you can visit the website to see other product teardowns if you are interested.
- The photo shows you the various modules that we mention in the previous slide, not very interesting unfortunately because they are mostly black in colour.
- Sorry to say that most IC chips will be in these colours though.
- Anyway, they are mostly hidden behind product casing so I guess no one will mind.

# iPhone front facing peripherals






- Smart phone these day uses many different types of sensors to enable many desired features.
- This slide here shows you a list of sensors you can find on a iphone, and this is not the latest iphone.
    - You have a proximity sensor that sense whether you face is near the phone so the screen will be blanked off
    - The ambient light sensor that would auto tune the display brightness based on the ambient lighting.
    - The Dot projector that projects thousands on IR dots in order to profile the face, and this is the basis of the FaceID feature that Apple use.
- Now the processor needs to interface to all these sensors and the various modules that we mentioned in the previous slides.
- As such, computer interface is a very important topic which we be will going into shortly under the Signal-Chain Sub-system.

## Sub-Systems in a Computer System

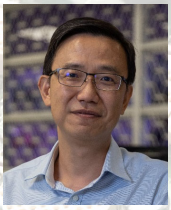
- Processor Core
- Signal Chain Sub-System
  - ADC-DAC
  - Parallel and Serial (UART/SPI) Digital Interfaces
  - Direct Memory Access Controller
  - Interrupt Controller
- Memory Sub-System
  - Semiconductor Memories (SRAM, DRAM, FLASH)
  - Flash memory based Solid State Drives
  - Magnetic Hard Disk Drives
  - Cache Memory Management
  - Virtual Memory Management
- Communication and User Interface Sub-System
  - Wired (USB, SATA, HDMI)
  - Wireless (Wifi, Bluetooth)
  - Input Devices (KBM, Capacitive Touch, Camera, Microphone)
  - Output Devices (Display, Speakers)



- I've divided the computer into a few sub-system
- There is the processor core which you are very familiar with by now
- The signal sub system deals with interfacing the computer to the real world, and a main part of it is the computer interface topic that I mention earlier.
  - This consist of the Analog-to-Digital Convertor, ADC in short, which brings the real world signal into the computer
  - the Digital-to-Analog Converter, DAC in short, which push the information out from the computer to the real world.
  - The various digital interfaces that the processor use to communicate with various modules
  - And modules that enable different type of transfer mechanism between modules.
- You have the memory sub-system where we will touch on the different memory types and how the processor manage the code and data in the computer using cache and virtual memory management.
- And lastly, I've grouped the more complicated interface and modules in a separate sub-system called the comms and UI.
  - They actually also deal with interfacing but the mechanism behind these interfaces are more complicated so we will only be scratching the basics in this course. That's the main reason why I put them in a separate group.

# Cx1106
# Computer Organization and Architecture

## Signal Chain Sub-System



Oh Hong Lye
Senior Lecturer
School of Computer Science and Engineering, Nanyang Technological University.
Email: hloh@ntu.edu.sg

- We will touch on the Signal Chain Sub System in the next few videos.
- I've divided the video into smaller chunks so you can have some rest in between. But some of the topics may take a longer time, in which case please feel free to exercise you right to press the pause or fast forward button.

# Signal Chain Sub-System

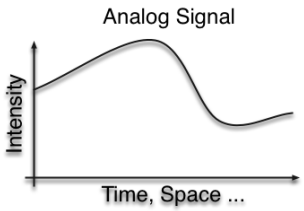

- Signal Chain Sub-System
  - ADC-DAC
  - Parallel and Serial (UART/SPI) Digital Interfaces
  - Interrupt Controller
  - Direct Memory Access Controller

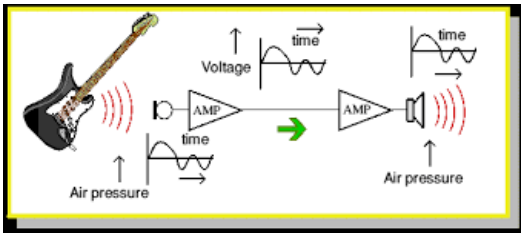


- The signal chain sub-system deals with the transfer of signals between the real-world and the processor.
- The analog real world signals, passes through the ADC to the processor, which operates in the digital domain, and gets out to the real world again via the DAC.
- There are many sub-topics in the signal chain, we are going to focus on only a few of them.
  - The ADC and DAC process that deals with conversion between analog and digital domain
  - The digital interfaces between digital modules
  - And the two main types of data transfer mechanism: interrupt and DMA.


- This section is about how analog real word signals are passed into the digital processor via the analog to digital converter, and how the digital output of the processor are subsequently passed back to the real world via the digital to analog converter.

# Analog and Digital Signal



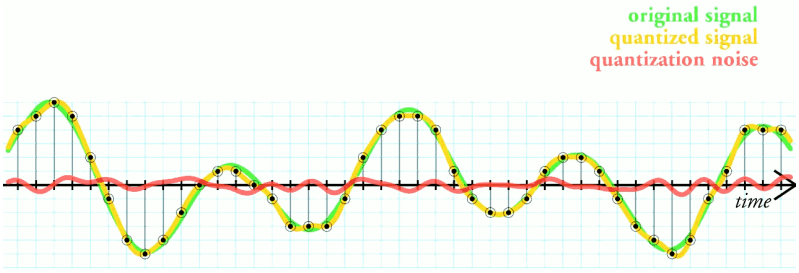




- Real world signals are Analog in nature.
  - Examples: Sound, light, heat, pressure etc
  - In the past, most processing are done in analog domain.
  - With the introduction digital processors and decreasing cost to build them, digital processing became increasingly popular as it offer more flexibility in implementation and are more tolerant to noise and component aging.
  - Analog signal has continuous voltage level and are continuous in time domain.



- Before we go into detail discussion of the various interfaces, let take some time look at the types of signal we need to deal with and their characteristics.
- All real world signals are actually analog, the sound we hear, the light we see, the wind we feel and the heat we sense.
- By analog, we mean that the signal magnitude is continuous. It is also continuous in time domain.
- So on a graph the intensity vs Time, an analog signal is represented by a continuous solid line.
- During the pre-digital processors days, analog signals are processed as it is in the analog domain.
  - For example, noise cancellation, audio equalization, TV brightness, audio loudness etc.
- With the enhancement in semiconductor technology in the last 40 years, digital processors become sufficiently economical to be deployed in large scale and overtook analog processing as the platform of choice for signal processing.
- Digital processing has the advantage of being more tolerant to noise interference and component aging.
- It also offer more flexibility in implementation as software can be easily modified to suit different scenarios.

# Digital Signal

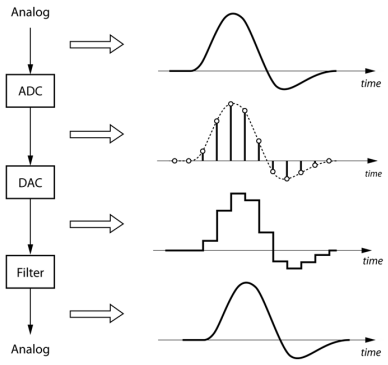



- Digital signals are discrete time representation of analog signal.
- Obtained through process of digitization, commonly know as Analog to Digital Conversion.
- Analog signals are digitized to its digital equivalent using a Analog-to-Digital Converter (ADC).
- Digital signal has discrete voltage levels.
- Similarly, analog signal can be reconstructed from digital signal via digital to analog conversion.



- The graph in this slide shows an analog sine wave being digitized.
- Digitization is also called analog to digital conversion. Typically done by passing the analog signal through an Analog-to-Digital Converter (ADC).
- What the ADC does is to sample the analog input at fixed interval, know as the sampling interval.
    - The value of the analog signal at each sampling instance is recorded and assigned to the closest discrete level that the particular digital system allows. More on that in the next slide.
- So these black points you see in the graph correspond to the digital equivalent of the analog sine wave.
- These digital data can be used subsequently to reconstruct the sine wave. The process is call digital-to-analog conversion and is done via a Digital-to-Analog Converter (DAC).

## Transformation between Digital and Analog Domain

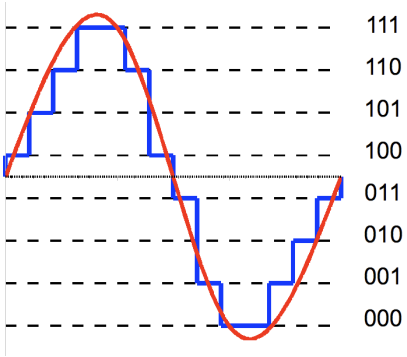



- Figure on the left shows conversion between analog and digital domain.
- Digital signal is obtained by sampling the analog signal level at discrete time (known as the sampling interval).
- Sampling frequency needs to be at least twice the signal frequency (Nyquist theorem).
- Typically, the analog signal level is assigned to the nearest discrete voltage level allowed in that particular digitization process.
- Analog signal can be reconstructed back by applying a filter on the digital signal.



- The diagram here shows the entire process of Analog-to-Digital followed by Digital-to-Analog conversion.
- Let's check out the Analog-to-Digital Conversion first
  - Analog signal is sampled at fixed interval known as sampling interval.
  - At each sampling point, the analog signal level is assigned a value that is closest to that allowed by the ADC used
- The digital equivalent of the analog signal is then processed in the digital processor to suit a particular purpose, for example changing the tone of the audio signal.
- In order for human to hear the processes audio, the digital signal needs to be converted back to its analog equivalent.
- This process is known as Digital to Analog Conversion
- It is achieved by passing the digital signal through a DAC followed by a filter to smooth out the edges of the DAC output.

## Digital Quantization




- Figure on the left shows a typical digital quantization process of a analog signal.
- A 3-bit system is used in this example. So all signal will be mapped to one of the 8 possible representations (000 to 111).
- At each sampling point, the analog signal level is approximated to the nearest digital equivalent.



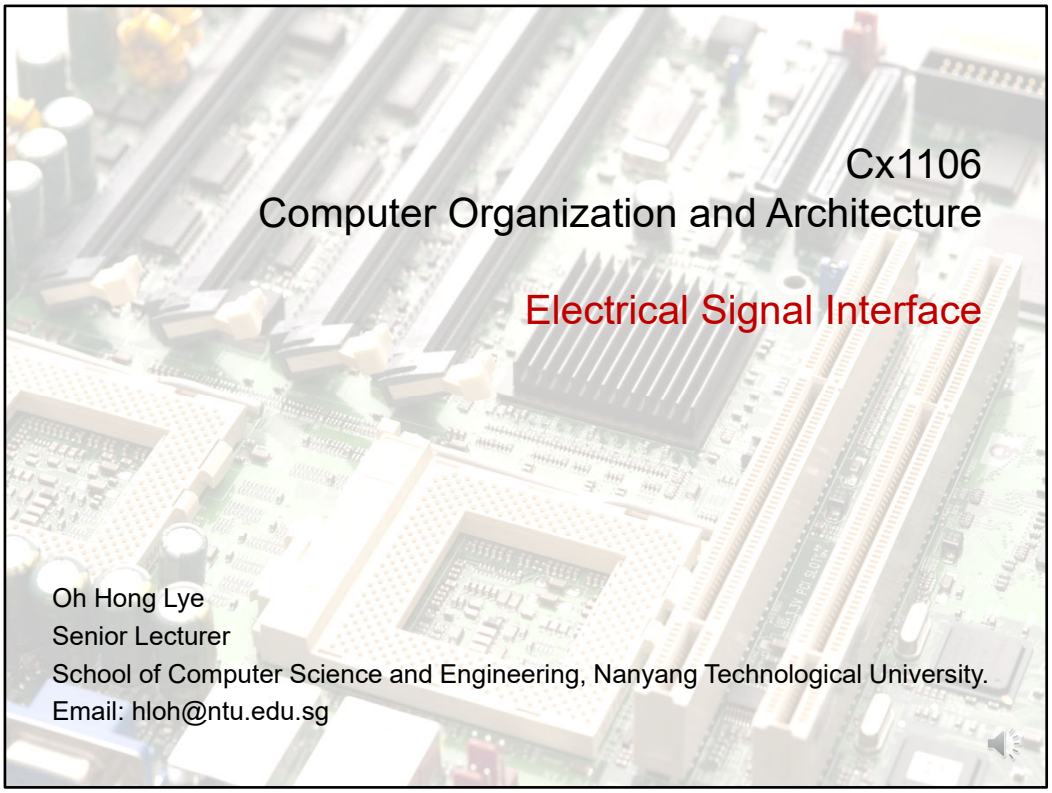
- The diagram in here illustrate the details of how a sine wave is digitized using a 3-bit ADC.
- For a 3-bit ADC, there are 8 discrete output levels 0-7.
- At each sampling point, the analog signal will be assigned to one of the 8 values, the one that is closest in magnitude to the analog magnitude sampled.

## Signal Chain between Processor and Real World


- Digital Processors works only in the digital domain so typical process these days is to convert the real world analog signal to digital signal, allow the processor to work on the digital data, and reconstruct back the analog signal to be output to the real world.
- So most of the interfaces we are dealing with in this course are in the digital domain.



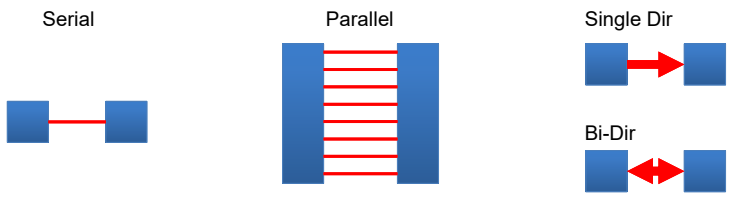
- As mentioned earlier, digital processors are the de facto platform for signal processing these days.
- Since digital processor only understand digital data, all real world signals would have to be digitized before they can be processed.
- Information exchange within the computer is typically digital in nature, they will only be converted to analog when the signals need to get into the real world.
- So naturally, most of the interface we'll touch on in this course is digital in nature.




- This section will cover the topic of electrical signal interface, where we will look at the various requirement for two devices or modules to be able to communicate with each other.

# Interface

- A boundary where two or more devices meet to exchange information. Some modes of connection below
  - Single-bit Data transfer (Serial)
  - Multiple-bits Data transfer (Parallel)
- For each mode above, the connection could be Single direction or Bi-direction.

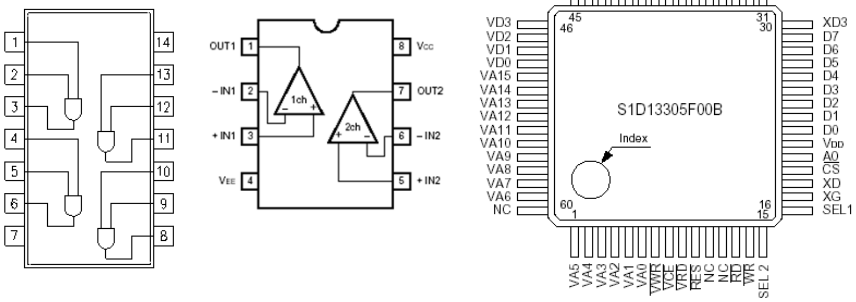





- As mentioned previously, Algorithm processing these days are predominantly done in the digital domain.
  - Which is why most of the interface we are going to discuss in this course are digital interfaces.
- Now, before that, what exactly is an interface?
  - In computer terms, interface is a boundary where two or more devices meet to exchange data.
- There as many different ways in which devices can be connected.
- The common ones are listed below.
  - Serial. In this connection, only one data line is available for each direction transfer. So data is transfer a bit at a time.
  - Parallel. Multiple data lines are available to transfer data. So multiple bits are transferred simultaneously.
  - In each way, the connection can be Single direction or Bi-Direction.

# Input, Output and Bi-Directional Pins

- Semiconductor devices interface to the outside world via pins.
- Depending on the device design and configuration, these pins are either an input, output or bi-direction (input and output) pin.




Oh Hong Lye / Cx1106 11

- We talked about input and output signals in the earlier slides.
- The diagram you see here are IC pinout diagram, which illustrate the layout of the pins for a particular IC.
- Examples of actual ICs are shown in these photos.
- On the silicon chips, we often see pins or balls attached to their black casing.
- These are actually their Input/Output(I/O) interfaces to the external world.
- The pins/balls are electrical conductors and connects internally to the silicon DIE in the chip packaging.
- There are many different types of packaging, some can be as big as your palm, some smaller than your finger tip.
- In line with the types of interface connections we discussed in the previous slides, there are only 3 types of I/O where direction is concerned: Input, Output and Bi-Directional.

## Interface Compatibility

- Interfacing one electronic device to another requires compatibility in
  - Electrical signal level
  - Communication protocol (Handshaking and Data signals).



- This is a very simple slide, but it delivery two important information.
- In order for two devices to talk, the criteria has to be met
  - First, the electrical signal level of the two devices has to be compatible
  - Second, the devices has to speak in the same language. For our case, they have to use the same communication protocol.
- We will discuss the two points in more details in later slides.

## Electrical Signal level (Safety)

- Primary consideration when connecting two electrical device together.
- Ensure that the output voltage level of output device do not exceed the maximum allowable input voltage level of the input device.
- Electronic devices will either get 'fried' i.e. spoilt or have its reliability reduced if the input voltage is higher than what they are designed for.
- So do check the voltage level which the device input/output is operating on (1.8V, 3V, 5V, 15V etc) before connecting them together.



- There are two aspects where Electrical Signal Level is concerned.
- First is the safety consideration.
    - When connecting two electrical devices, the most important thing you need to remember is safety.
    - We are not dealing with high power electronics here but if you made the wrong connection, you may still see some smokes coming out from your equipment and boards.
    - So Make sure the output voltage and input voltage of the two devices are compatible. For example, don't connect a 5V device to a 3V device unless you know what you are doing.
- Electronics devices in general do not like to be electrical stressed. So it'll either get fried (spoilt) or its reliability will be reduced if a voltage higher than what it is designed for is applied to the device.
- So do check out the input/output voltage rating of each of the devices you are connecting.

- Second consideration is the digital logic level compatibility which we will discuss in the next slide.

## Electrical Signal Level (Digital Data Transfer)

- 5V => Logic '1'' or '0' ?
- Four parameters
  - VOH.
    - Transmitting a Logic '1' yield a signal level of >= VOH
  - VOL.
    - Transmitting a Logic '0' yield a signal level of =< VOL
  - VIH.
    - A received signal with level >= VIH will be recognized as a Logic '1'
  - VIL.
    - A received signal with level =< VIL will be recognized as a Logic '0'



- When a device receive a 5V signal at its input pin, is this considered Logic '1' and '0'?
  - Answer could be either actually.
  - It depends on a few device parameters shown here
- These four parameters dictate whether two devices are able to communicate with each other properly or not.
- Their definition are as follows
  - VOH determine the min voltage that a device will transmit if it is transmitting a logic '1'
  - Similarly, if a device transmit a logic '0', it will transmit a signal with magnitude no larger than VOL.
  - On the receive side, we have VIH and VIL.
  - For a signal to be recognised as a logic '1', its has to at least VIH.
  - For a signal to be Logic '0', its electrical level has to be less than VIL.

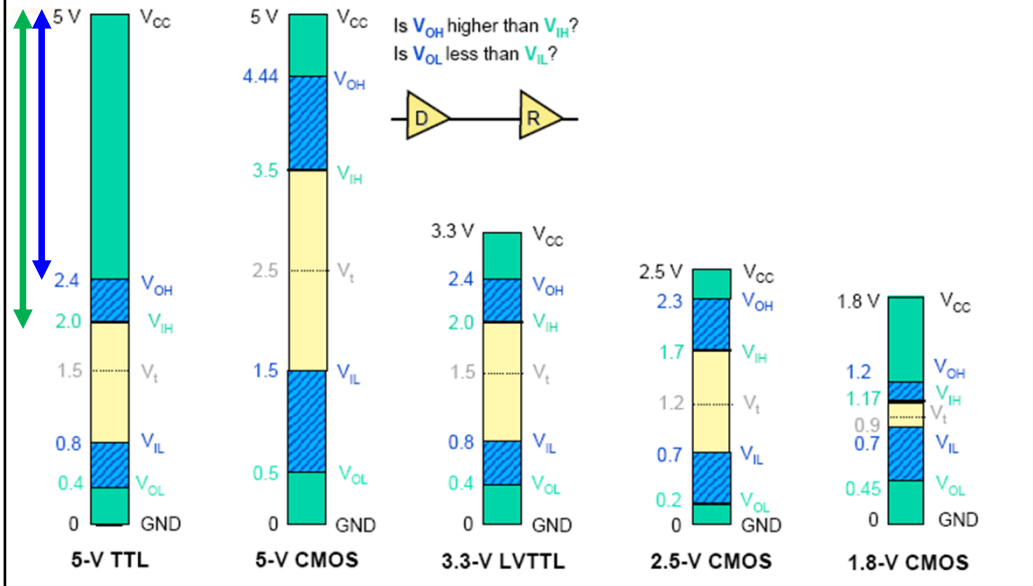
## Electrical Signal Level (Digital Data Transfer)


- Device #1
  - Output Voltage level for logic '1' = V1+ (e.g. 5V)
    For typical value of V1+, look for V(OH) parameter in device datasheets.
  - Output Voltage level for logic '0' = V1- (e.g. 0V)
    For typical value of V1-, look for V(OL) parameter in device datasheets.
- Device #2
  - Min Input voltage range to recognized as logic '1' = V(IH)
  - Max Input voltage range to recognized as logic '0' = V(IL)
- In order for Device #2 to sense the logic level correctly,
  - Condition 1: V1+ ≥ V(IH) (e.g. V1+ ≥ 2.0V)
  - Condition 2: V1- ≤ V(IL) (e.g. V1- ≤ 0.8V)

Oh Hong Lye / Cx1106 15

- This slide further illustrate what we discuss in the previous slide.
- Device #1 is the transmitter and V1+ correspond to Logic '1' and V1- correspond to Logic '0'.
  - So if we assume that VOH = 2.4V and VOL = 0.4V for Device #1, then V1+ will be greater or equal to 2.4V while V1- will be smaller or equal to 0.4V.
  - These parameters are design dependent and are specified in the device datasheets. You may have studied this in your digital logic module as well.
- When the signal reached Device #2,
  - If we assume that VIH = 2V and VIL=0.8V,
  - the signal will be recognized as a logic '1' if its signal level is larger than 2
  - It will be recognized as a logic '0' if its signal level is less than 0.8V
  - These parameters can be found in the device datasheets as well.
- In summary, two devices will be able to talk to each other if their VOH, VOL, VIH and VIL are compatible.

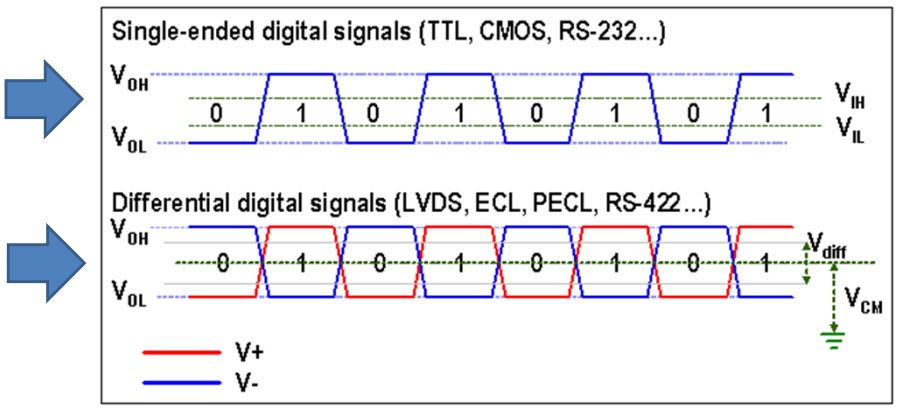
# Electrical Signal Level Standards




- Depending on the voltage rating and the chip design, the four parameters (VOH, VOL, VIH and VIL) can be quite different.
- This chart shows some of the devices out there. Its not the complete list, there are many others.
- Bottomline is, please check the device datasheets before you decided to hook them up together.
- VIH and VIL usually covers a larger range compared to VOH and VOL. This is to cater for noise or losses as the electrical signal propagate thru' the circuits.

# Differential Signals

- Differential signals has better noise tolerance so is able to be clocked at higher frequency.
- V(CM) = Common Mode Ground.




Source: http://www.ni.com/cms/images/devzone/tut/a/07c0be3031.gif

Oh Hong Lye / Cx1106 17

- What we have seen in the previous slides are single ended signal.
- Electrical signals are also be transmitted via differential signals.
- How it works is that if V+ > V-, than it is a logic '1', if V- > V+, then it'll be a logic '0'.
- In fact, many of the common transmission standard you are familiar with are based on differential signaling.
  E.g. USB, HDMI, SATA etc.
- Differential signals has better noise tolerance in general
  - Because they have a larger margin. You can see in the diagram that the V+ and V- are further apart.and can accommodate more noise.
  - Any external interference will also have the same effect on V+ and V-, so the difference between them remains unchanged.

## Communication Protocols

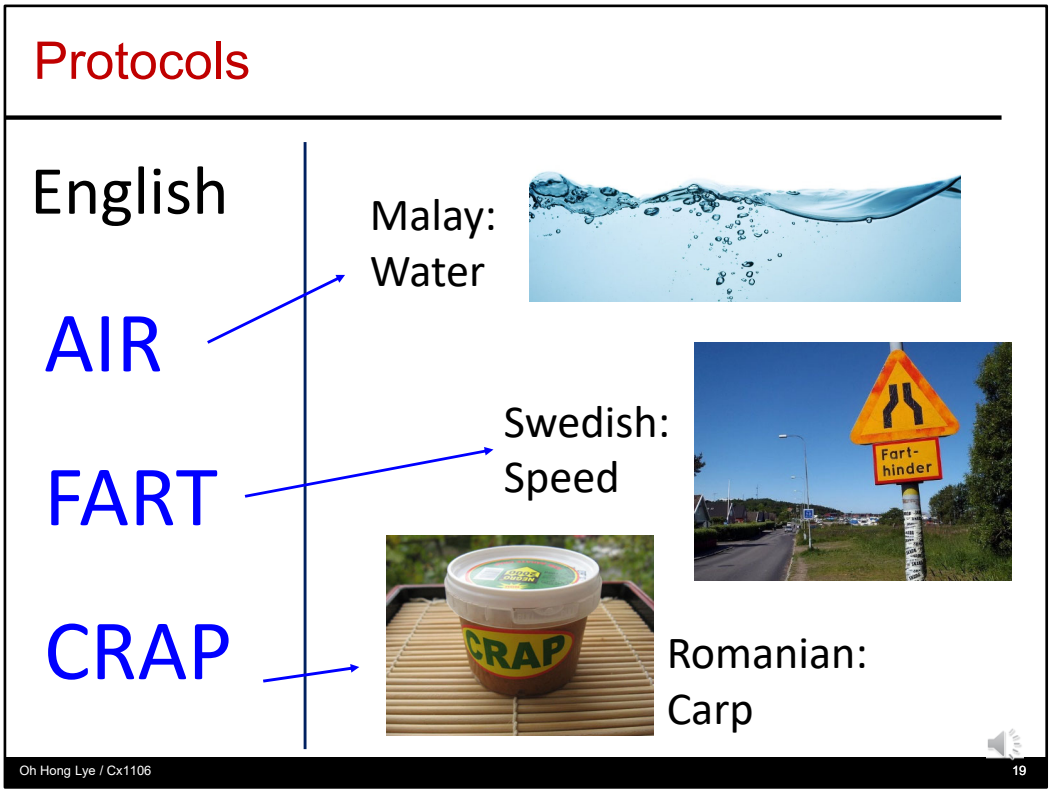
- Communication Protocols refers to how the data are formatted during transmission.
- Some examples
  - Number of bits in a transmission frame
  - What synchronization to use
  - Data width
  - Types of data and its formatting
- Examples of communication protocols are USB, UART, SPI etc.

| | |
|---|---|
| **Protocol 1** | **1001001001001000** |
| **Protocol 2** | **1001001001001000** |

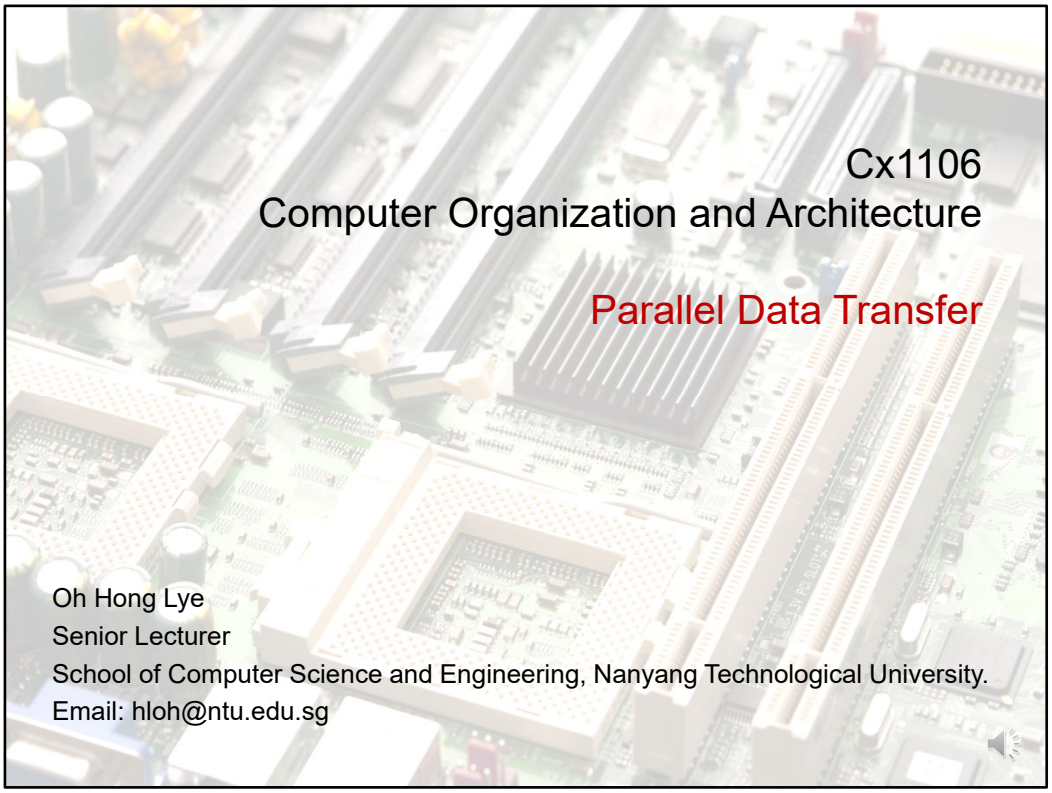
**Blue: Synchronization Bits**
**Green: Data Bits**
**Red: Error Correction Bits**



- The second criteria in order for two devices to communicate properly is that their communication protocol has to be compatible.
- Communication protocols refers to the how data are formatted during a transfer.
  - This could come in different form such as number of data bits, types of synchronisation bits used, error correction bits etc.
- In the two protocols shown here, you can see that although the information transfer is exactly the same, they are in fact two different communication with different synchronisation, data and error correction bits. Obviously, these two device will not be able to communication with each other as they speak different languages.



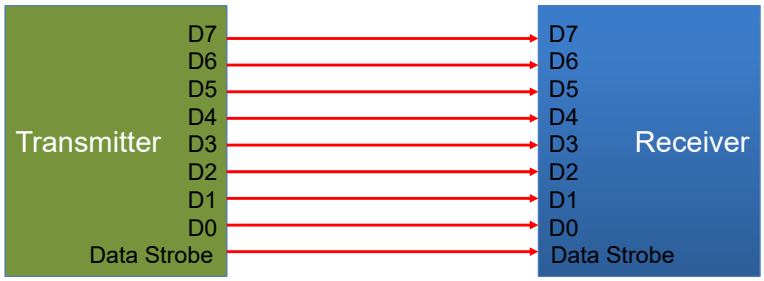

- Just a slide on some fun facts that illustrate examples of same same but different communications.
- You can skip this slide if you are tired.
- So on the left we have three English words: Air, Fart and Carp
- Now the same words in another language actually takes on a totally different meaning
    - Air in Malay is ‘A-eh’, which means water
    - Fart in Swedish is speed, so if you see a sign showing ‘Fart Hinder’ in Sweden, it just a speed bump, not asking you to hinder your fart.
    - And lastly, crap in Romanian is actually a type of fish.




- This section is about transferring data via the parallel interface.

# Parallel Data Transfer

- Multiple bits of data are transferred simultaneously between two devices.
- Synchronous in nature as some sort of strobe signal is needed to inform the receiver when to latch in the data. E.g. rising edge of strobe signal.
- Able to achieve higher transfer rate than Serial Interface (using same clock).
- But more prone to Signal Skew and Crosstalk (see later slides).




Oh Hong Lye / Cx1106 2

- Data can be transferred either one bit or multiple bits at a time.
- When multiple bits are transferred simultaneously, it is known as a Parallel interface.
- One example is shown in this slide, where the data is transmitted on multiple lines, collectively known as a data bus..
- The transfer is typically synchronized by a strobe signal, as indicated by the pin "Data Strobe" in the diagram. For example, data could be transferred on the rising edge of the strobe signal.
- The strobe signal may also be referred to as the clock signal of the parallel bus when we are comparing the data transfer rate with respect to the serial interface.

- Some examples of devices with parallel interface are Static Random Access Memory (SRAM), Dynamic RAM (DRAM), Read-Only-Memory (ROM) etc.

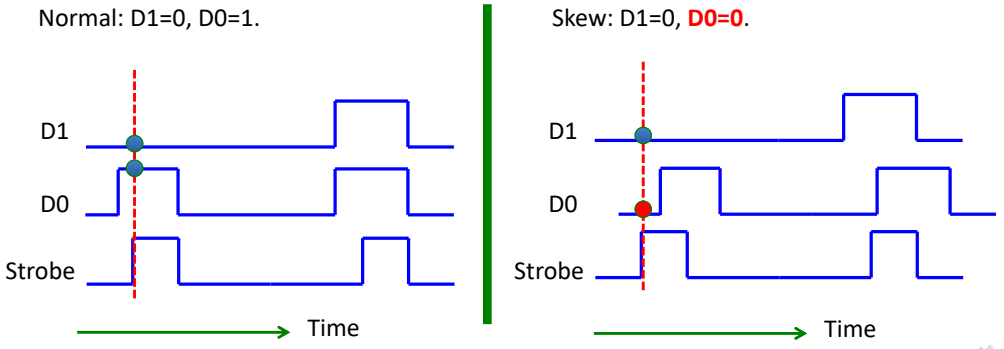
- Given the same clock rate, Parallel interface has a faster data rate than that achievable by the Serial interface.
- This is because Parallel interface has multiple data lines and therefore is able to transfer multiple data bits simultaneously.
- Serial interface, only the other hand, can only transfer one bit at a time since it has only one data line.

- However, because there are more signal lines that are closed to each other, parallel interface is more Prone to signal skew and crosstalk.

- These two phenomenon will result in wrong data being latched by the receiver.
- More details in the next few slides.

# Signal Skew

- If for some reason (due to circuit design, external interference), the signal in one or more data lines took different amount of time to reach the receiver, then there is a skew between the signals in the parallel data bus.
- Below example illustrate the effect of signal skew. Data is latched by the receiver on rising edge of the strobe signal.
- This result in wrong data (D0=0 instead of 1) being latched by the receiver.




Oh Hong Lye / Cx1106 3

- Signal skew is the phenomenon whereby signals in different data lines of a single multi-bit data transfer travels at different speed.
- This means these signals will arrive at the receiver at different time.
- This is illustrated in the diagrams in this slide.
- Suppose D0 and D1 forms a 2-bit data bus. If the transmitter output '01', i.e. D1=0, D0=1.
- Under normal circumstances when there is no signal skew, the receiver will receive '01' at its end.
- But if for some reason, which I will elaborate later, the signal on D0 gets delayed and reach the receiver later compared to D1 and the strobe signal. Then the receiver will see a '00' instead.
- If you look at the diagram on the right, when the strobe signal goes up, the value of D1 and D0 are both '0', which means the receiver will receive the wrong data.
- There are different factors contributing to signal skews and we will be touching on more details in the next few slides.

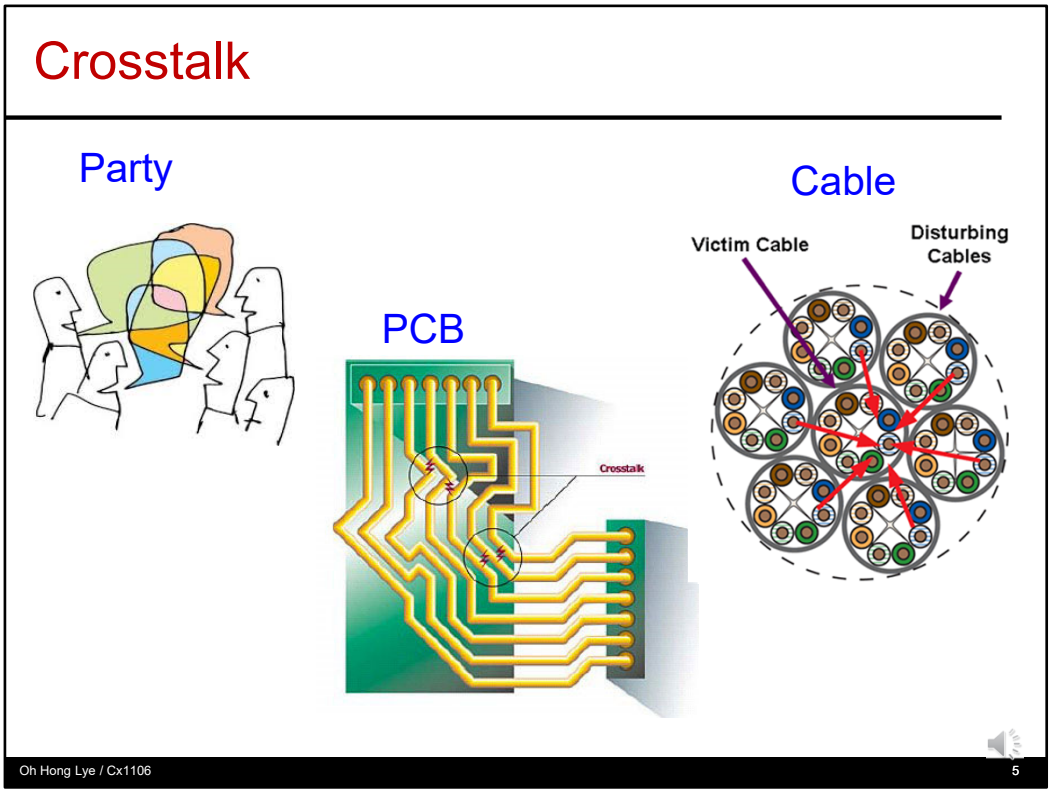
## Signal Skew – Contributing Factors

- Signal Skew is due to variation in propagation delay between signals from the same data bus.
- Propagation delay is the time taken for signal to travel between two points in a circuit.
- Capacitance and Resistance of the physical data line is a major contributor to circuit propagation delay.
  - The larger the resistance and capacitance, the larger the delay. Illustrated in the equation $\tau=RC$ where $\tau$ is the time constant dictating rate of change of voltage levels in the data line concerned.
- Variation in resistance and capacitance of the signal lines can be due to
  - Variation in PCB trace length/width (lead to change in impedance).
  - Connecting active components (capacitors, inductors, IC etc) to some of the signal lines.



- Signal skew is a result of variation in propagation delay between signals.
- So what is propagation delay?
  - Its the time taken by the electrical signal to travel from one point to another within the electrical circuit.
- Many factors can affect the propagation delay. One of the biggest contributor is capacitance and resistance of the wire that the electrical signal is travelling on.
  - The product of resistance and capacitance is proportional to the time it takes for any voltage change on the wire to reach its steady state.
- What do I mean by that?
  - Say the logic '1' of the device correspond to 3V, when the device tries to output a logic '1', the signal voltage will not reach 3V immediately but will take some time to get there.
  - How long it takes is dependent on the RC product value of the circuit it is connected to.
- And these two parameters change if the circuit environment changes.
  - Such as variation in length/width of PCB traces,
  - Connection of components such as capacitor, resistors, ICs on the circuit concerned.
- So when connecting high speed devices via parallel bus, it is very important to ensure that the propagation delay of the data line within the same bus is kept the same so data can be transferred correctly.
- One common practice is to try to keep the length of the PCB traces of the same data bus to be of similar length.

- You can see that some interesting patterns formed by the PCB traces connecting between the processor and high speed DRAM in this photo.
- These method of routing the PCB trasces is known as serpentine routing.
- For this case, the main objective being to ensure that the propagation delay of the data lines are similar to prevent signal skew.



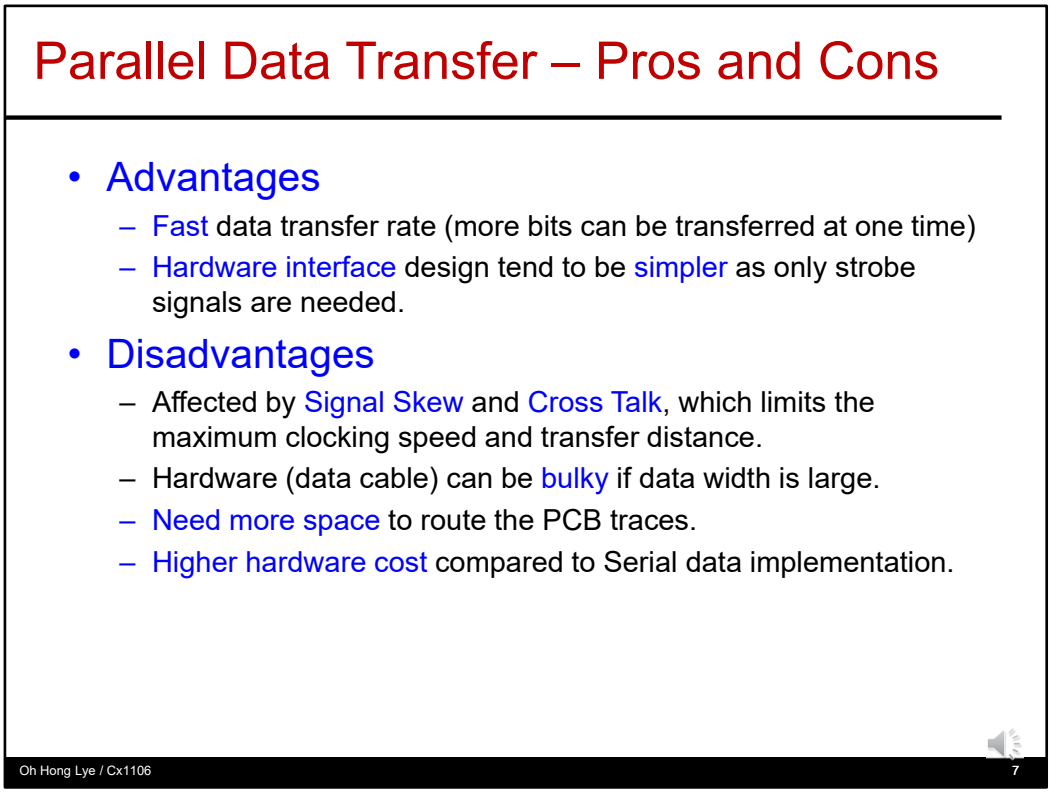

- Next is Cross Talk.
- In layman term, Crosstalk are unwanted interference from neighbours.
- In a party where many people is talking simultaneously, voices from people other than the person you are talking to will come across as Crosstalk to you and your partner.
- Similarly, in a PCB with many data lines closely packed together, there will be a lot of cross interference between these signals as they
- propagate through the circuits.
- Interference can be via
    - Electromagnetic domain where the wires behaves like antenna emitting Radio Waves.
    - Electrical domain as the electrical signal from one wire could potentially coupled to the other wires at the connection point.
      They could also be coupled via ground current returning via different paths on the PCB ground plane.

## Crosstalk – Electrical Circuit

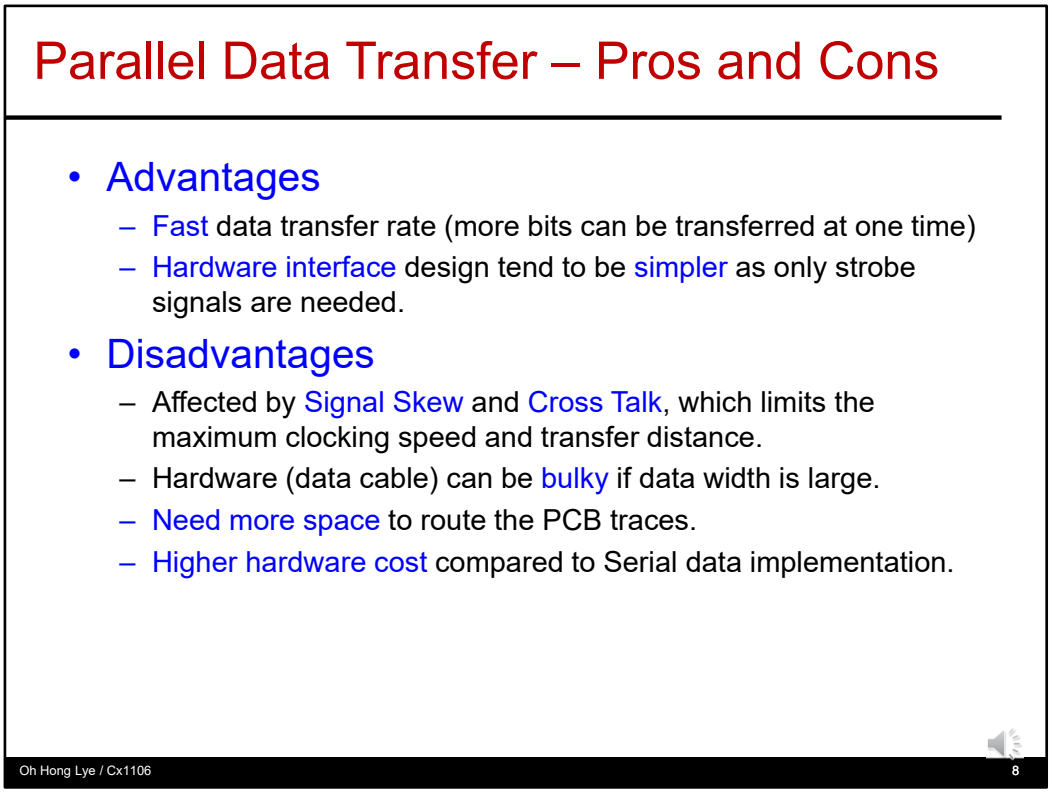
- Crosstalk are undesired coupling of signals from one circuit to another circuit.
- In a parallel bus context, the close placement of the data lines in PCB routing or cabling enables the effect of electrical signal in one trace/wire to be coupled over to the other. Creating undesired interference (crosstalk).
- Crosstalk can be transmitted electrically or via electromagnetic radiation (the trace/wire acts like an antenna).



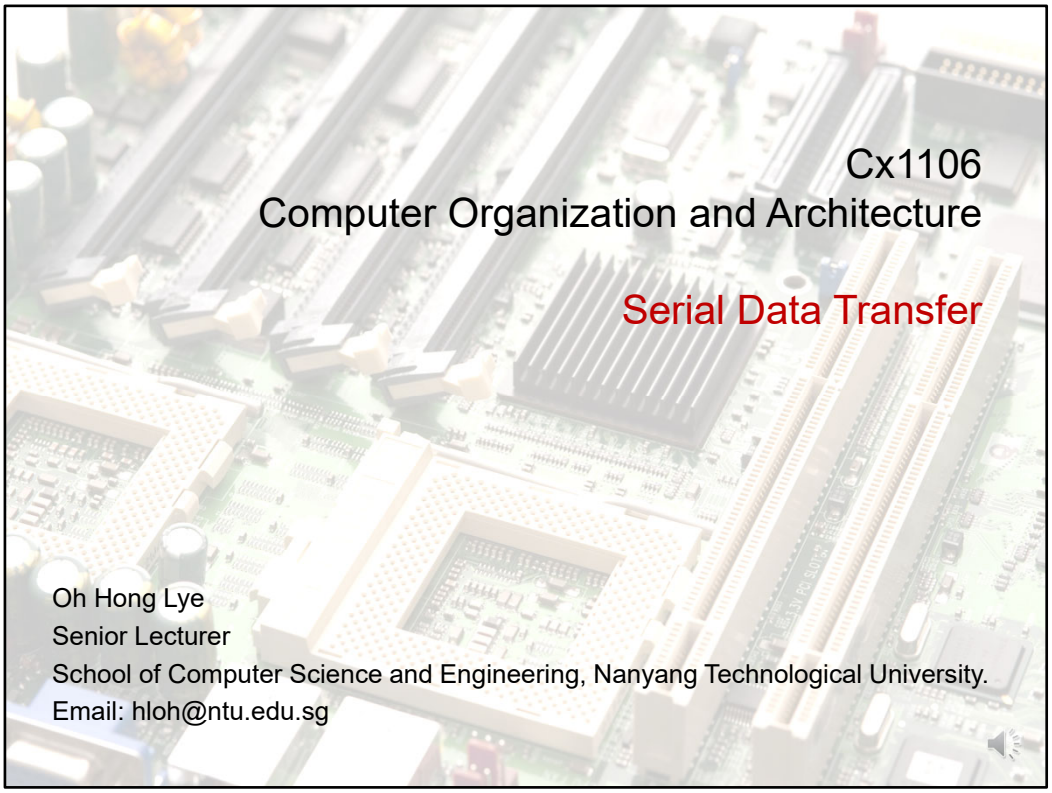
- This slide summarise what I have talked about in the previous slide.
- You can pause the video for a while to go through the information presented.




- Summarising the pros and cons of parallel interface
- Advantages
- Given the same clock rate, Parallel interface is able to transfer data at a faster rate compared to Serial Interface.
- However, because there are more signals lines (data and strobe) that are closed to each other, parallel interfaces are more prone to signal skew and crosstalk. These two phenomenon will result in wrong data being latched by the receiver. This also limit the maximum clock speed and maximum distance that the signal can travel without being affected by signal skew or cross talk.
- Parallel bus or cable also takes up more space compared to serial bus because more wires are involved.
    - As a result, it is more complex and costly to design the PCB for parallel bus.
- The cable is also more bulky so house keeping can be an issue.
- The forst photo shows difference in bulk comparing the IDE HDD cable many years back and the SATA HDD cable that we use today.
- The second photo shows you how messy it get be when you have a few of these IDE cables in your Computer Chasis. I used to have issue just trying to figure out which cable belongs to which device!

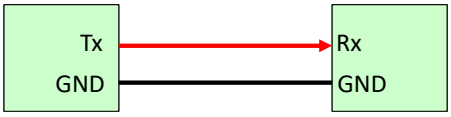



- Summarising the pros and cons of parallel interface
- Advantages
- Given the same clock rate, Parallel interface is able to transfer data at a faster rate compared to Serial Interface.
- However, because there are more signals lines (data and strobe) that are closed to each other, parallel interfaces are more prone to signal skew and crosstalk. These two phenomenon will result in wrong data being latched by the receiver. This also limit the maximum clock speed and maximum distance that the signal can travel without being affected by signal skew or cross talk.
- Parallel bus or cable also takes up more space compared to serial bus because more wires are involved.
    - As a result, it is more complex and costly to design the PCB for parallel bus.
- The cable is also more bulky so house keeping can be an issue.
- The first photo shows difference in bulk comparing the IDE HDD cable many years back and the SATA HDD cable that we use today.
- The second photo shows you how messy it get be when you have a few of these IDE cables in your Computer Chasis. I used to have issue just trying to figure out which cable belongs to which device!




- Next, we will be touching on the Serial Data Interface

# Serial Data Transfer




- Data is transferred one bit at a time over a single data line. Comparatively, parallel data interface transfer multiple bits simultaneously.
- Less affected by signal skew and crosstalk because there are less electrical wires involved compared to parallel data transfer. Hence, able to support higher frequency clocking.
- Data transfer rate lower (compared to parallel interface) given the same clock rate since only one data line is available.



- Serial Data interface, as mentioned earlier, transfer one bit of data at one time
- So given the same clock rate, its data transfer rate will be lower than that of the parallel interface.
- However, because only one data line per direction is involved,
  - it is less affected by signal skew since less wires are involved => chances of a skew happening is less.
  - It is also less affected by crosstalk since there are less neighbouring wires or cables as compared to a parallel interface..
- With less influence from signal skew and crosstalk, a serial interface bus, if well designed, could potentially achieve a higher clock rate compared to the parallel bus, this allows its transfer rate to be closer to that of the parallel bus.

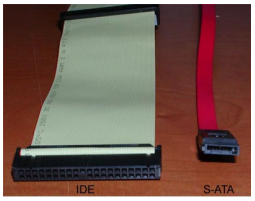
## Serial Data Transfer Pros and Cons

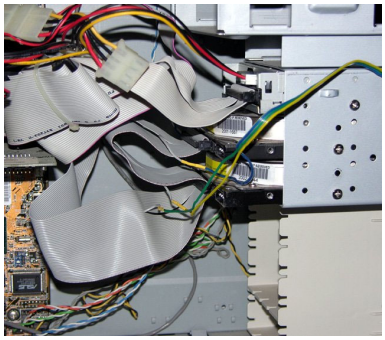
- Advantage
  - Less affected by signal skew and crosstalk because there are less electrical wires involved compared to parallel data transfer. Hence, able to support higher frequency clocking.
  - Able to transfer data reliably over a longer distance.
  - Lower cost since less wires and connectors are needed.
- Disadvantage
  - Data transfer rate lower given the same clock rate since only one data line is available.
  - Hardware interface design typically more complex as it need to handle serial to parallel conversion (Processor typically only process in bytes or multiple bytes).

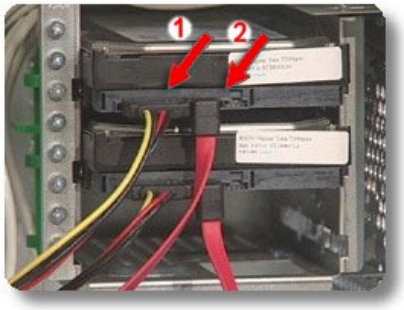


- To summarise
- The Advantages of Serial transfers are
  - It is less affected by signal skew and crosstalk as there are less wires involved in each interface
  - As the effect of signal skew and cross talk is not as significant when thre are only a few wires, serial interface can therefore be potentially clocked at a higher frequency and is able to transfer data over a longer distance.
  - It is also less costly since less wires and connectors are needed.
- The Disadvantages are
  - Given the same clock rate, Data Transfer rate for serial interface is lower than parallel interface.
  - Hardware interface design may be more complex because serialization and deserialization operation is needed to convert
    the serial bit data to multiple-bit data format such as word, byte used by the processor.

# Parallel and Serial Comparison



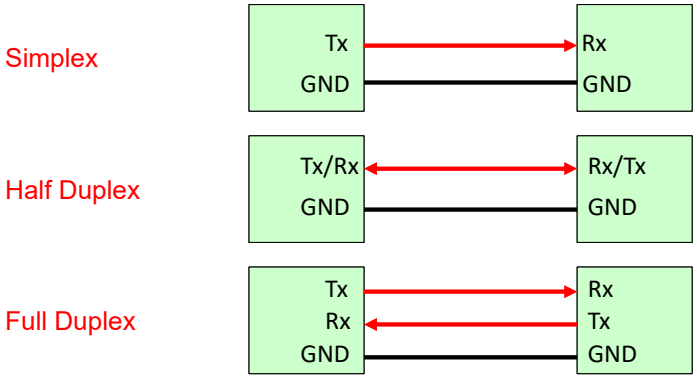



Oh Hong Lye / Cx1106 4

- This is an example of the space saving you can achieve by converting from parallel to serial interface.
- Example here shows the difference between the old IDE interface, which is a big mess and the new SATA interface for HDD.

# Serial Data Transfer Mode

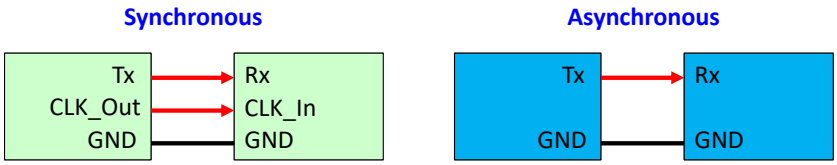
- Simplex: Data transfer in one direction only.
- Half-Duplex: Data transfer in both direction, but RX and TX is mutually exclusive.
- Full Duplex: Simultaneous RX and TX






- There are 3 types of serial transfer mode:
- In Simplex mode, One device is the transmitter while the other is the receiver. Data transfer is only in one direction.
- For Half-Duplex mode, Both devices can operate as transmitter as well as receiver. But at any point in time, there is only one transmitter and one receiver, with data flowing in one direction.
- Lastly, in Full Duplex mode, Both devices can operate as transmitter as well as receiver. There are separate data line for each direction transfer so each device can transmit and receive data at the same time.

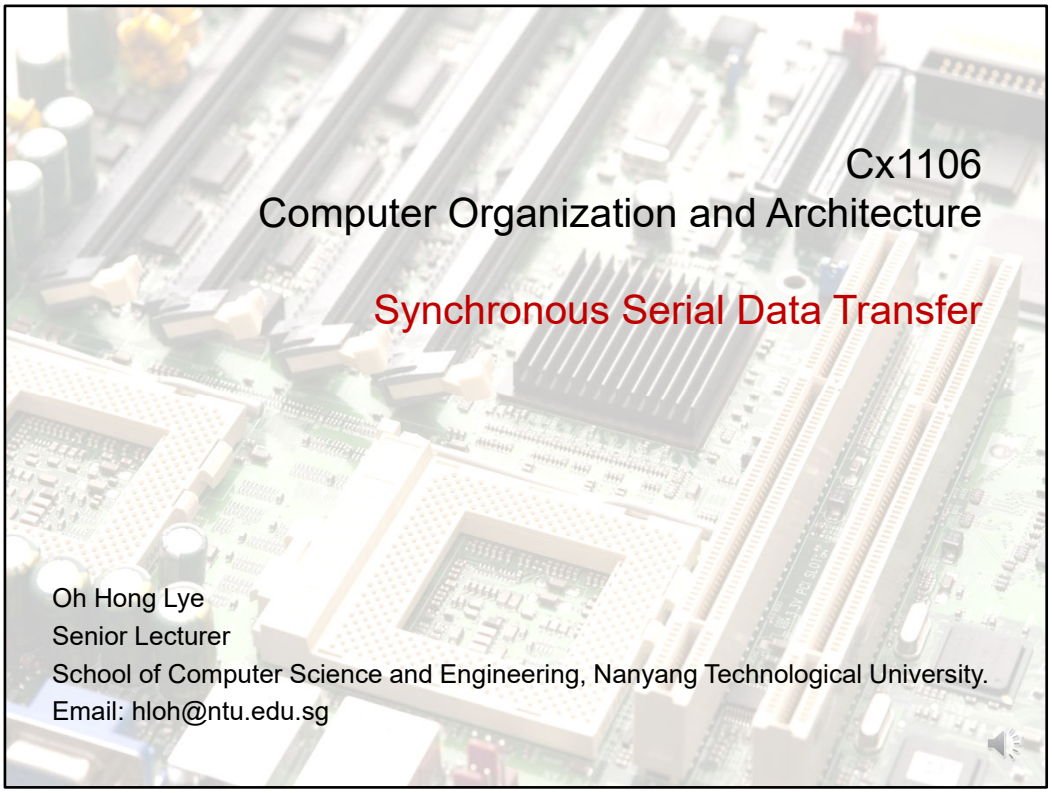
## Synchronous vs Asynchronous




- If there is a common clock signal between the Transmitter and Receiver, then the communication is termed synchronous. Else, the communication is termed asynchronous.
- In synchronous transmission, there is a common clock signal to synchronize the data transfer. E.g. receiver to latch in the data at every rising edge of the clock.
- In asynchronous transmission, there is no common clock signal so devices have to agree on a pre-fixed clock frequency to use for data transfer.

Oh Hong Lye / Cx1106 6

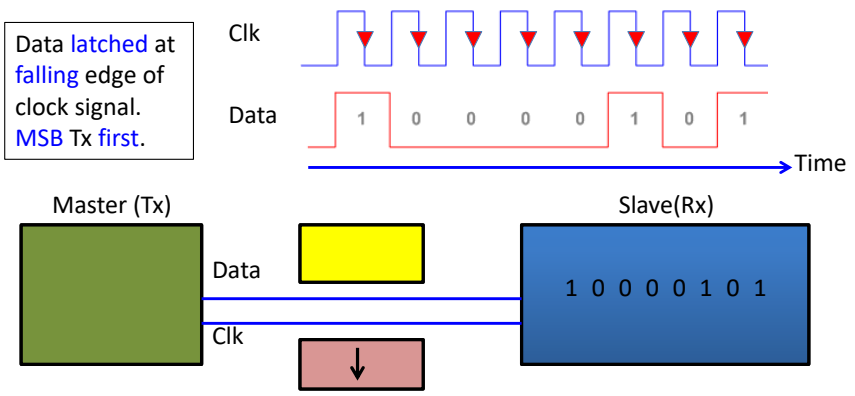
- Serial communications can be Synchronous or Asynchronous in nature and they have different design considerations during implementation.
- A serial communication is synchronous when there is a common clock signal between the transmitter and receiver.
- If a common clock signal is absent, then the transfer is asynchronous in nature.
- For synchronous transmission, the common clock signal is used to synchronize the data transfer, it tells the receiver when to latch in the data bits send by the transmitter.
- For Asynchronous transmission, since there is no common clock. But that doesn't mean that sychronisation is not required between the transmitter and receiver.
- In fact, synchronization is still needed for the data transfer to be successful.
    - Synchronization is achieved by sending synchronization information over the data line, and the devices will have to agree on the clock rate to be used for transferring data.
- This obviously will lead to some issues and we'll go into more detail later.
- Note that this concept of Synchronous and Asynchronous transfer that we discuss here is referring to the electrical signals layer.
- At higher protocol layer, the word synchronous may take on a slightly different meaning and it is possible to implement synchronous communication over an asynchronous physical link. You need not bother about that now but keep in mind that such implementation is possible.




- We will be covering synchronous serial transfer in the next few slides.

## Synchronous Serial Transfer

- Common clock signal between transmitter and receiver to synchronize the data transfer.
- Master-Slave configuration. With Master providing the clock signal.
- Potentially allows faster transfer rate since no data overhead is needed to synchronize the transfer.






- As mentioned, synchronous serial interface has a common clock signal that is used to synchronize the data transfer.
- The devices are usually configured in a Master-Slave configuration, with master providing the clock.
- What is shown here is a typical transmission.
  - The signal waveform is as shown. This is the typical way a signal transmission diagram is illustrated in documents.
  - What is means is that data on the left will be sent/receive first.
  - In this example, the data is latched on the falling edge of the clock.
  - The Master will first output a '1' and data will be latched by the slave on the falling edge of the clock signal.
  - This is followed by a '0' and corresponding clock edge.
  - Master will continue to send the rest of the data bits, which is latched into the receiver on the falling edge of the clock.
- Depending on the configuration, data can also be latched in at the rising edge instead.

# I2C and SPI Bus



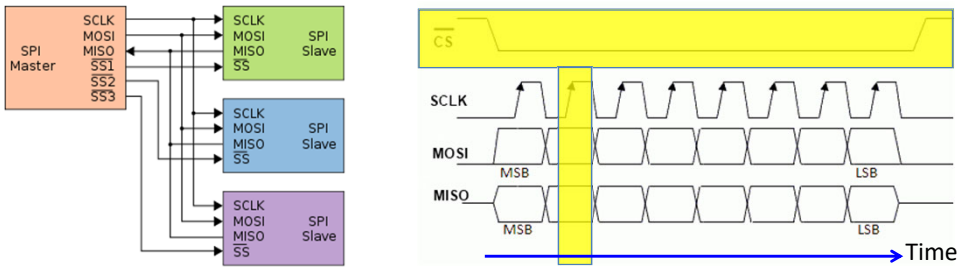
- Popular serial buses used by processor to transfer data and control many peripheral devices.
  - Accelerometers
  - Temperature Sensors
  - Touch Screen Controllers
  - Power Supply Modules Configuration
  - Audio/Video Codecs Configuration

I2C will not be covered in this module

Oh Hong Lye / Cx1106 9

- I2C ad SPI are two of the popular synchronous serial bus standards used by the processor for interfacing to other devices.
- Some of the examples are shown here.
- In this course, we will only be touching on SPI bus interface.

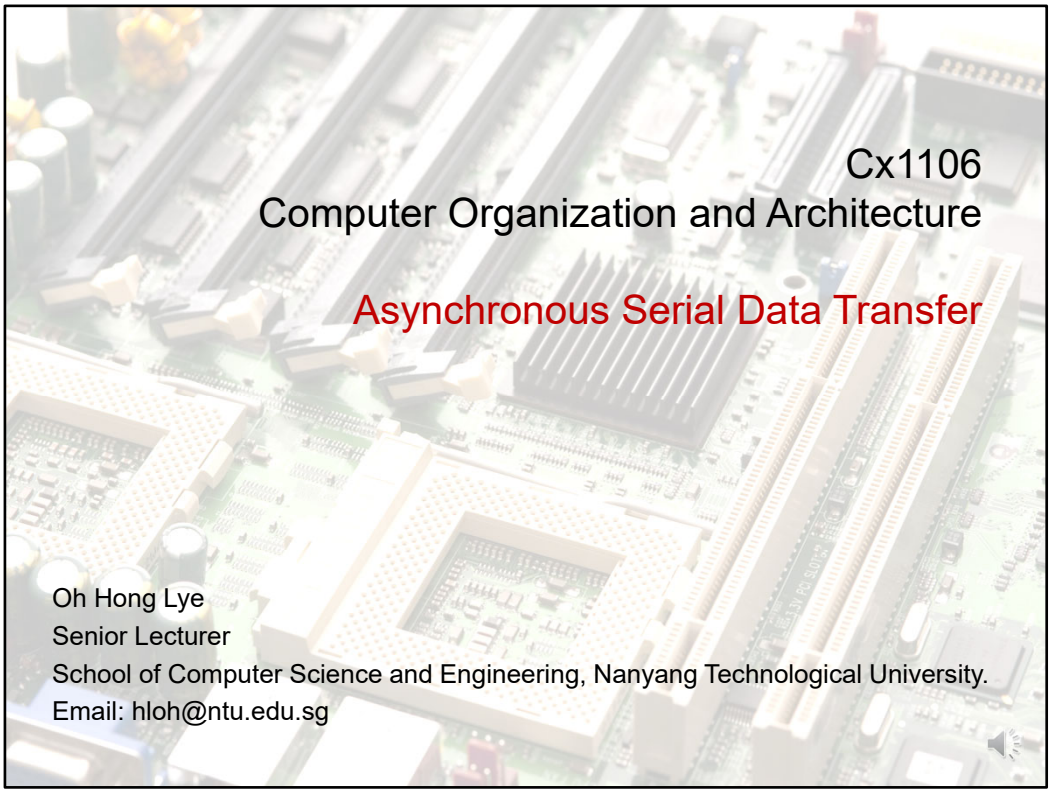
# Serial Peripheral Interface (SPI) Bus




- To start the transfer
  - Slave Select (SS) has to be pulled Low.
- Data transfer
  - Data on MOSI and MISO latched in on rising/falling clock edge (configurable)
  - MOSI: Master-Out-Slave-In (Master Output, Slave Input)
  - MISO: Master-In-Slave-Out (Slave Output, Master Input)
- Allow multipe slaves via use of multiple Slave Select.

Oh Hong Lye / Cx1106 10

- SPI bus is a synchronous serial bus. You can see the explicit clock line that the Master used to synchronize transfer over SPI interface.
- To initiate a transfer, the master will first pull the Slave Select pin low. This will enable the target SPI slave.
- Data on the MOSI and MISO pins are then latched on the rising edge of the SCLK signal.
- MOSI refers to Master Out Slave In, this allows master to output data to slave. Similarly, MISO refers to Master In Slave out.
- SPI bus allows multiple slaves to be connected to one master.




- This section is on asynchronous serial interface. This is the last section of the Serial Interface chapter.
- Yes, your agony is going to be over soon.

## Asynchronous Serial Transfer

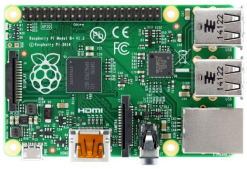
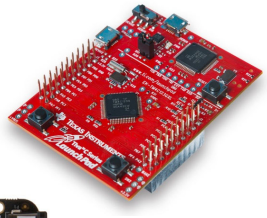
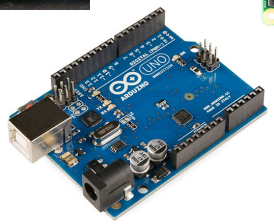
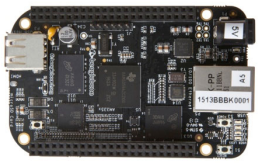
- No common clock is provided between transmitter and receiver.
- Prior to the transmission, the receiver needs to know the transmitting clock rate and the number of bits that are to be transferred with each data packet.
- Special SYNC words are used to indicate START/STOP condition.
- Upon receiving the START SYNC Word, the receiver then use its own local clock to track the timing.
- Potential skew issue between the two local clocks as transmission progress.
- Asynchronous Transmission typically also uses SYNC word/bits to provide occasional time stamp for receiver to synchronize its clock to the transmitter clock (helps to reduce clock skew between the two clocks).



- In Asynchronous transfer, as mentioned earlier, there is no common clock between the transmitter and receiver so they need to rely on other means to achieve synchronization.
- First, both devices needs to agree on a certain clock frequency and the number of data bits in a data packet.
- They also need a special SYNC word to indicate the start and stop of transmission. This SYNC word is embedded in the data packet. Remember again that there is no explicit clock line so all information has to be transferred via the data line.
- Upon receiving the Start SYNC word, the receiver will start to track the data using its own local clock. Since the transfer clock frequency is known beforehand, the receiver can roughly guess when the data bits are coming in and latch it.
- The fact that the transmitter and receiver are using two different clocks means that over time, these two clocks will drift relative to each other since there is no way these two clock will run at exactly the same frequency.
- If the clock skew kept increasing, it'll eventually lead to wrong data being latched by the receiver.
- One way to mitigate this issue is for the transmitter to send SYNC word periodically so that the receiver can use that info to re-align their local clock.
- The re-alignment of clock here means getting the clock edge of the two local clock to occur at the same time stamp.

## Universal Asynchronous Receiver Transmitter (UART)

- One of the most commonly used serial interface.
  - PC Serial COM Port (RS232) uses UART protocol.
  - Many USB devices uses a Virtual COM Port implementation to connect to the PC.
  - Communication interface between PC and many processor development boards e.g. Arduino Board, TIVA-C Launchpad etc

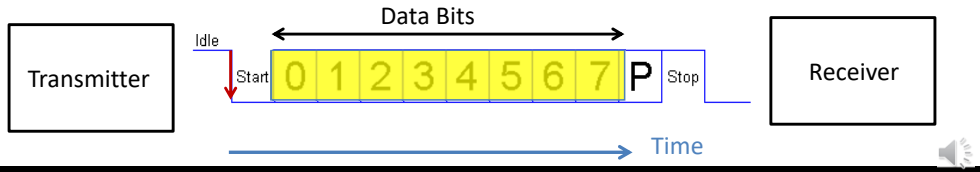






Oh Hong Lye / Cx1106 13

- One of the most commonly used Asynchronous Serial bus standard is the UART.
- It stands for Universal Asynchronous Receiver Transmitter.
- The PC serial COM Port you see here used to be found in every PC but has since been replaced by USB.
- But UART as a protocol is still a popular protocol for various processor development boards like Arudino and Rasberry Pi.
- You don't see a physical COM Port on these boards though, what you have is the USB interface.
- What happened is that a Virtual COM Port will be enumerated over the USB when processor board such as Arduino is connected to the PC. And the Virtual COM Port uses the UART protocol for communication.

# UART Transmit

- During **IDLE** State, the data line is in 'powered' state i.e. Logic '1'.
- The transmitter send a **START** pattern (logic '0') to alert the receiver.
- Sending a logic '0' from idle state (logic '1') will result in a falling edge on the data line. This is typically used by the receiver to detect the start of transmission.
- This is followed by the actual **DATA** at a frequency known to the receiver. The transmitting clock rate is also known as the **baud rate**, and determines the number of bits transmitted per second.
- A **PARITY** bit (optional) may also be sent for the receiver to check the integrity of the data packet.
- A **STOP** bit (Logic '1') terminates the transmission.




Oh Hong Lye / Cx1106 14

- When the UART is IDLE, its data line will be at a logic HIGH.
- When the transmitter want to send data, it'll first send a STAR bit by pulling the data line LOW.
- This falling edge of the data line will alert the receiver to get ready for data reception.
- The transmitter will then output the actual data at a known frequency, also know as the baud rate.
- At the end of the data transfer, the transmitter can optionally output a parity bit.
- Function of a Parity bit is to enable error detection at the receiver side. We discuss this in more details in the next slide.
- After that, the transmitter will send a STOP bit to end the transmission.
- If the transmitter wants to send another word, it needs to restart the whole process by transmitting the start bit again.
- You can't send multiple words with only one set of Start and Stop bit.
- For UART protocol, a few of the symbols or status has a fixed logic level
    - IDLE state is always Logic '1'
    - START bit is always Logic '0'
    - STOP bit is always Logic '1'

## Parity Bit

- If parity scheme is enabled, the receiver will also sample the parity bit and checks for parity error.
- If even parity scheme is used, there should be an even number of '1's in the data field and the parity field.
- Hence, if there is an odd number of '1's in the data field, then the parity bit transmitted should be a '1' to make the total even.
- If receiver receives odd number of '1's in a even parity scheme, it'll flag a Parity Error, meaning one or more bits in the transmission is wrong.
- Vice versa for odd parity scheme.
- The receiver then samples the **STOP** bit(s), if a '0' is detected instead of '1', then receiver will flag a Framing Error.



- UART parity scheme is designed to detect errors during transmission
- There are 3 different options: Even, Odd or No Parity.
- If Even Parity Scheme is employed, the transmitter will count the number of '1's when it is transmitting the data. Depending on the number of '1's recorded, it will then transmit a '0' or '1' as the Parity bit, so that the total number of '1's in the Data field and the Parity Field is an Even number.
- For example, if the data transmitted is 1110000, which has three '1's, then the transmitter will transmit a '1' as the parity bit, so that the total number is 4, which is an even number.
- Similarly, Odd Parity scheme implies the transmitter will set or clear the Parity bit so that total number of '1's in data and parity field is ODD.
- No parity means parity bit will not be generated.
- On the receiver side, after it received the data and parity bits from the transmitter, it will count the number of '1's in there, if it doesn't match the parity scheme used, then it implies that one or more bits in the transmission is wrong. But it won't be able to tell the exact bit or bits that are wrong.
- Another type of error is the framing error. This will be flagged if the receiver detects a logic '0' in the bit position where the STOP bit is expected.

# UART Tx Example

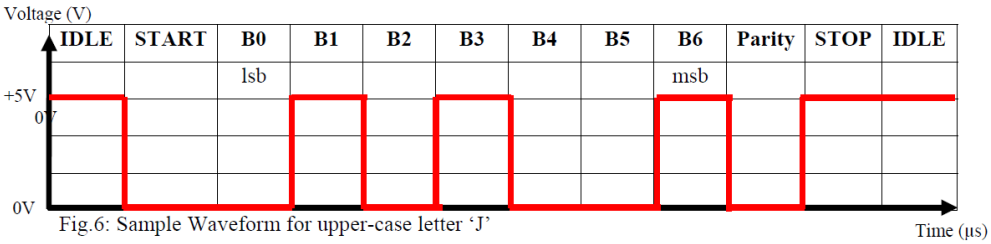


Fig.6: Sample Waveform for upper-case letter 'J'

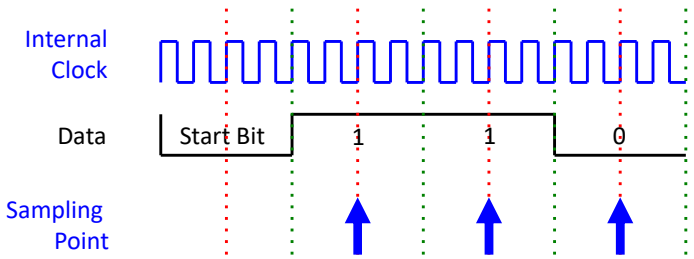
- Figure 6 above correspond to the configuration 7O1 (7 Data, 1 STOP and Odd parity).
- Capital Letter 'J' => Ascii value = 0x4A
- 0x4A => 100 1010 (binary)
- Presented in LSB first => 0101 001
- IDLE=1, START=0, STOP=1.
- Parity bit logic depends on number of 1's in Data Field and the Parity Scheme (Odd or Even) used.



- One example to illustrate the UART transmit operation.
- Here we have a 7O1 configuration, which implies 7 Data bits, Odd Parity and 1 STOP bit. There is always only one START bit in each UART frame.
- Notice that for UART transmission, LSB is transmitted first.
- In this example, the data transmitted is 0x4A or 100 1010 in binary. But you'll notice that the waveform shows that the LSB is transmitted first so the logics on the UART signal waveform shows 0101001.
- There are 3 '1's in the data, so the transmitter will clear the parity bit. You can see that Parity bit = 0 in the diagram.

# UART Receive

- Receiver monitors the Data line for the Start Bit. In real world design, the falling edge on the data line will trigger an interrupt in the microprocessor to start the receiving process.
- Needs to know the baud rate in order to sample the data bits correctly.
- Internal clock of the UART typically run at a multiple e.g. 16X of the baud rate so as to timed the sampling closed to the middle of each data bit.
- Below is an example of UART receive with internal clock running at 4X baud rate (for illustration only). Internal clock rate is typically faster than 4X baud rate in real world implementation.

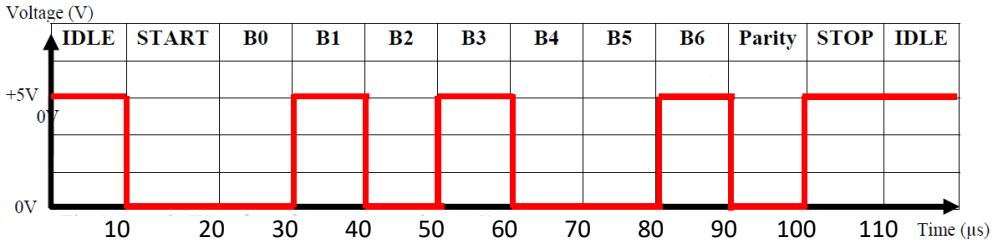



Oh Hong Lye / Cx1106 17

- Ext, let's take a look at the UART receiver operation, which is more complicated than the transmitter.
- The receiver needs to constantly monitor the data line to see if transmitter has anything for him.
- In a real world design, interrupt mechanism is used to make the process more efficient.
- The receiver needs to know the baud rate in order to sample the data correctly.
- And the internal clock of the UART receiver is typically much faster than the baud rate. One value is 16X.
- The receiver will use this faster clock to derive the best location to sample the incoming data.
- The example below illustrates the process. I'm using a 4X sampling here to simplify things. Actual internal clock is usually much faster.
- We have the first logic '0' as the start bit. This is followed by the data bits
- Since the internal clock is running at much faster frequency, the receiver will able to estimate the mid point of each data bit and perform the sample at that instance.

# UART Rx Example

**Tx baud rate = 1/(10 us) = 100000 bps. Configuration = 7O1.**



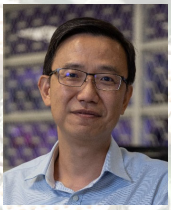

- RX Data = 1001010b
- RX Data = 0011000b ?
- **Information Sampled @ 200000 bps: 0000110011000011011 1**

Oh Hong Lye / Cx1106 18

- Similarly, let take a look at the timing diagram example of a UART receiver to illustrate the concepts learnt.
- The first Logic '0' is the START bit,
- it is followed by the data bits and the bits that comes into the UART receiver in chronological order are 0101001.
- Since LSB is transmitted first, the actual data received is 1001010 or 0x4A.
- Odd parity scheme is used so Parity bit as transmitted by the transmitter is '0' so that total number of '1's is an odd number.
- Now, looking at the waveform, is it possible for the receiver to receive 0011000 instead?
- Answer is yes and that happen when the transmitter and receiver baud rate is different.
- In this case, the receiver baud rate is twice that of the transmitter.
- So receiver will be sampling 20 points instead of 10.
    - It'll then use the UART configuration of 7O1 to decode these 20 samples, as shown in the slide.
    - The first data received is 0001100. Its LSB first so the actual value is 0011000.

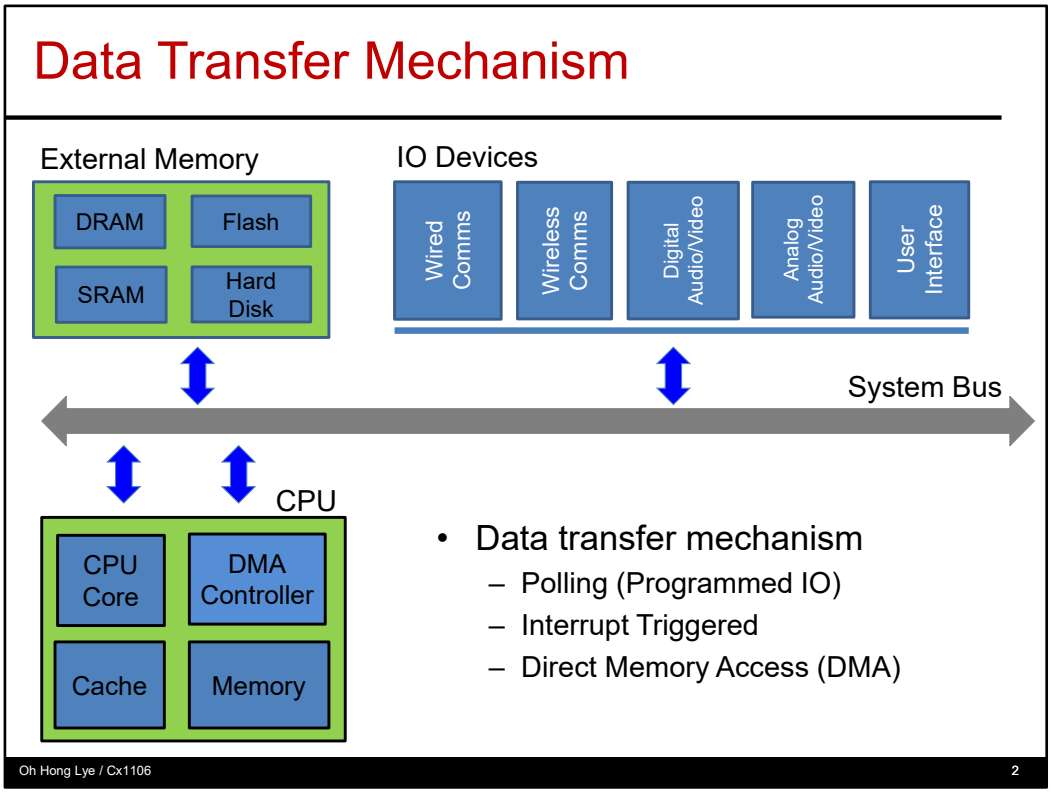
# Cx1106
# Computer Organization and Architecture

## Data Transfer Mechanism



Oh Hong Lye
Senior Lecturer
School of Computer Science and Engineering, Nanyang Technological University.
Email: hloh@ntu.edu.sg

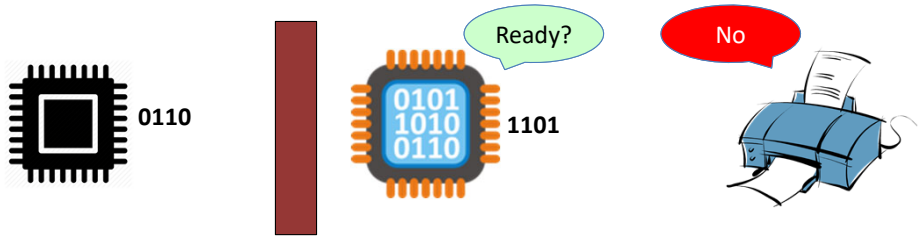
- This section is on Polling and Interrupts, which are two data transfer mechanism commonly used in computer system.




- A computer system consist of the CPU, memory and IO devices.
- These blocks are connected by a system bus, which is used to transfer data between the blocks.
- The two controllers that can initiate these transfers are the CPU core and the DMA Controller.
- The 3 transfer mechanisms that we'll touch on in this lecture are
    - Polling
    - Interrupt Triggered
    - DMA – stands for Direct Memory Access (we'll go into more details later).

# Polling Technique

- CPU polls a certain I/O port continuously (using software) for data or readiness of the port to perform a data transaction.
  - For example, the CPU polls the printer port continuously to see if the printer is ready to accept data.
  - If it is ready, the CPU writes a data byte to the printer port. Else, it waits.
- CPU has full control and dedicate 100% of its resource in the whole data transfer process and does nothing else.

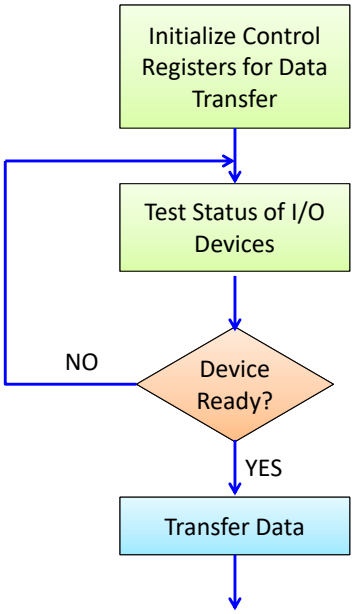





- First, Polling. Its also known as Programmed IO.
- In this method, the CPU poll a certain IO port continuously for data.
- One example is CPU polling the printer port to see if the printer is ready to accept data.
- If the printer is ready, the CPU will send the data to the printer for printing. Else, it'll wait.
- AS CPU uses 100% of its resource to do the polling, it is not able to entertain request from any other peripherals until the polling is done.
- The following animation explains how the process works.
  - Here we have a processor, printer and an external device
  - When the processor wanted to send some information to the printer, it'll first poll the printer to see if the printer is ready.
  - If the printer says 'No', the processor will continue to poll. Note that during this time, no other device will be able to get the processor's attention since it dedicate 100% of its attention on the polling.
  - Eventually, the printer will be ready and the processor can send the information over.
  - Once that is done, the process will remove the wall installed and allow other device to communicate with it.

# Polling Technique - Flowchart

- CPU performs all necessary initialization.
- CPU polls the I/O device for its readiness to perform data transfer.
- If I/O device is not ready, CPU continue to wait in the loop to check if device is ready.
- If device is ready, CPU make the data transfer and exit the loop.






- This sides shows the flow chart of the process that the CPU goes through when handling a data transfer suing polling technique.
- CPU first perform some system initialisation
- It then poll the status of the device it wants to talk to
- If the device is not ready, it'll continue to poll the device status in a loop.
- Once the device is ready, CPU will transfer the data and exit the loop

## Pros/Cons of Polling Technique

- Advantages
  - Programmer has complete control over the entire process.
  - Easiest method to test and debug.
- Disadvantages
  - Since the CPU waits in a loop, it cannot perform any other task until data transfer is completed.
  - Program execution of CPU held up while waiting for I/O device to get ready.
  - Inefficient use of CPU resources.



- Advantages of using polling technique to transfer data is that
  - Since it is a pure software approach, the programmer has complete control over the entire process, this makes testing and debugging simple.
- However, it result in very inefficient use of the CPU resources because
  - The CPU needs to dedicate 100% of its resource to do the polling, as such it would not be able to perform any other task.
  - Specifically, program execution will be held up while CPU is wating for the devices to be ready.

## Interrupts

- A signalling mechanism that allows internal and external peripherals to alert the CPU that attention is needed.
- Could be in the form of a external/internal electrical pulse or change in internal register status.
- Once the CPU receive the signal, known as interrupt request, it would then need to decide if it wanted to service the request.
- If CPU decides to service interrupt request, it will follow up with the series of procedures to handle the interrupt event.
- Interrupt mechanism is typically used trigger the CPU to start some operation, e.g. data transfer to memory, status registers, control registers etc.



- Interrupt mechanism is another common data transfer mechanism used and will result in a more efficient usage of CPU resources.
- Interrupt is a signal that is used to alert the CPU that its attention is needed.
- This signal can be an electrical pulse or some change in register status.
- When the CPU receive an interrupt, it can choose to service it, or not to service it.
- If the CPU decides to service an interrupt request, it will proceed to execute some pre-designed sequences. We will discuss more of that in later slides.
- In general, interrupt is typically used to trigger the CPU to start some operation, which could be some data transfer to memory, registers or other devices.

# Polling vs Interrupt

- Boiling Water Analogy

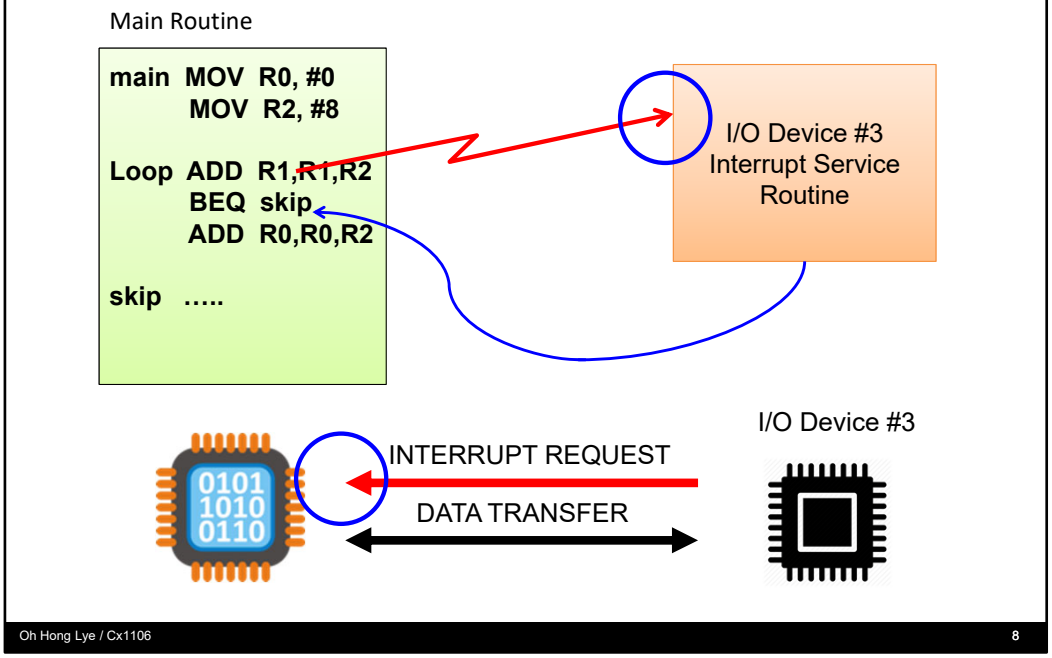
**Polling**:
Check every few minutes

**Interrupt**:
Listen for the whistle



- This slide use the example of boiling water to give a simply analogy to the difference between polling and interrupt.
- If you boil water using an open pot, you will likely need to monitor the status of the water continuously or periodically to see if the water is boiled. This is similar to the polling mechanism.
- But if you boil water in a kettle with whistle, you would not need to monitor the kettle and is free to go about doing your work until you hear the whistle from the kettle. The whistle sound is similar to the interrupt request that inform the processor, which is you in this case, that your attention is needed.

# Interrupt Triggered Data Transfer






- This slide will illustrate the interrupt mechanism with actual program.
- Supposed the processor is running some code in the main routine
  - After two instructions,
  - IO device #3 interrupted the CPU.
- If the CPU decides to service the interrupt,
  - the program will then branched off to a special routine known as the interrupt service routine.
  - For this case, it's a routine to transfer data between CPU and Device #3.
- After the necessary data is transferred, the control is passed back to the main routine where the
  code will continue where it has previously interrupted.
- One key timing parameter you need to take note is delay between
  - the point where CPU receive the interrupt request
  - To the point where it enter the ISR
- This delay is known as the interrupt latency, you will need to use this in your tutorial.

# Interrupt Vector Table

**Interrupt Vector Table**

| 0 | Reset ISR Addr |
|---|---|
| 1 | … |
| 2 | … |
| 3 | … |
| 4 | … |
| 5 | … |
| 6 | … |
| 7 | Ext INT 3 ISR Addr |
| 8 | SPI ISR Addr |
| 9 | UART ISR Addr |
| A | |

- How does the CPU know the location of the corresponding interrupt service routine for each interrupts?
- The starting address of each interrupt is stored in a table known as the interrupt vector table
- Each interrupt has a unique index to the vector table, e.g. UART interrupt could be in index 9, so if a UART interrupt occurs, CPU will know where to branch to by checking index 9 in the vector table.
- The values and interrupt source in the table on the left are for illustration purpose only, different processor has different indexing for their interrupts.



- In the previous slide, the CPU will branch off to the ISR if it decides to service the interrupt.
- But how does the CPU known where to branch to? Especially when there are many interrupt sources in the system.
- The answer is in a table known as the Interrupt Vector Table.
- This table contains the starting address of each ISR that is present in the system.
- Each interrupt has a unique location within the table so the CPU knowns exactly where to look for it when searching for the starting address of the corresponding ISR.
- For example, in the table shown, UART interrupt is stored at index 9, so if a UART interrupt occur, CPU will be looking at location corresponding to index 9 for the starting address of the ISR.
- Note however, that the information in the table here is for illustration purpose only.

# Interrupt Control Flow

- A signal from external/internal peripheral, or a change in status of some special registers notifies the CPU that some event has occur and ask for CPU's (immediate) attention.
- If the CPU decide to service the interrupt, it will suspend its current program temporarily.
- CPU look up the interrupt vector table to check the starting address of the interrupt service routine (ISR) for the corresponding interrupt.
- CPU would save a copy of the processor context, i.e. the current value of various registers, this is to allow the interrupted routine to continue its execution after returning from the ISR.
- CPU then proceed to execute the ISR linked to the interrupt that was triggered.
- Once the ISR is completed, CPU restore the save context, returns to the interrupted routine and continue from where it had left previously.



- This slide summarizes the interrupt control flow we discussed in the previous slides.
- You can pause the video to go through the points again.
- Note that CPU will save a copy of the processor context before branching to the ISR. This so call 'context' is basically the status of the key registers.
- This context will be restored after the ISR complete and the control is passed back to the interrupted routine, so that the interrupted routine can proceed from where it was interrupted.

# Interrupt Service Routine

- The ISR will perform some operation e.g. Data Transfer.
- ISR is typically a very short routine so as not to suspend the main program for too long.
- After servicing the ISR, CPU will return to the previous program and continue from it branched off.
- It is possible for CPU to receive multiple interrupt requests simultaneously since it typically interface to multiple devices.
- In this case, some arbitration scheme has to be designed to decide which interrupt to service first (priority, first-come-first-serve etc).



- This slides summarizes what I mentioned in the previous slide.
- You can pause the video to go through the points again.
- Some additional points that we didn't touch on in previous slides:
  - ISR tends to be a short routine so as not to hold the CPU for extended duration.
  - It is possible for CPU to receive multiple interrupt requests at the same time,
    so CPU has to be some scheme in place to decide which interrupt to service first.

## Pros/Cons of Interrupt Triggered Technique

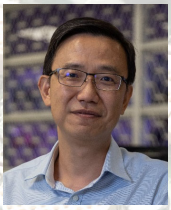
- Advantages
  - Efficient use of CPU resources as it does not need to monitor I/O device status.
  - CPU can continue with other tasks between interrupts.
  - Allows prioritization and pre-emption.
- Disadvantages
  - More hardware interface circuitry required between I/O device and processor.
  - Program is slightly more complex and difficult to debug.



- And here we have the advantages and disadvantages of interrupt triggered IO.
- Advantage
  - It leads to a more efficient use of CPU resources
  - And allow prioritization and pre-emption. This is One of the key enablers for high level OS as it allows
    a more critical software task to pre-empt other tasks to get the system going. Pre-empt here mean to
    halt the current task temporarily and run the higher priority task.
- Disadvantages
  - More hardware is needed to implement an interrupt mechanism, the control flow from interrupt request to the fetching of the ISR starting address are all handled in hardware.
  - Since hardware is involved, there is less visibility to the underlying operation so resultant program is more complex and more difficult to debug.

# Cx1106
# Computer Organization and Architecture

## Direct Memory Access



Oh Hong Lye
Senior Lecturer
School of Computer Science and Engineering, Nanyang Technological University.
Email: hloh@ntu.edu.sg

- This section is on Direct Memory Access, DMA in short. We will go through the operation of a DMA controller and how it can be used to relief the CPU from data transfer task.

## Optimizing CPU resources in Data Transfer

- Till now, we have been using CPU to perform the data transfer.
- This becomes increasing inefficient as amount of data increase as CPU has to spend most of its time moving data.
- As such, CPU has less time to perform algorithm processing.
- DMA controller (DMAC) is added to relief CPU of the data transfer task.
- DMAC has dedicated hardware that could move data more efficiently than CPU in scenarios where complex address manipulation is required. E.g. de-interleaving Left and Right Channel Audio data.
- If there are no conflict in hardware resources used, it is possible for data transfer via DMA and CPU execution to occur simultaneously.
- If there is a conflict in hardware resources used, e.g. both DMAC and the CPU needs the system bus, then access will be given to the one with higher priority.
- Who has the higher priority and whether the priority is configurable depends on processor design and is thus processor specific.

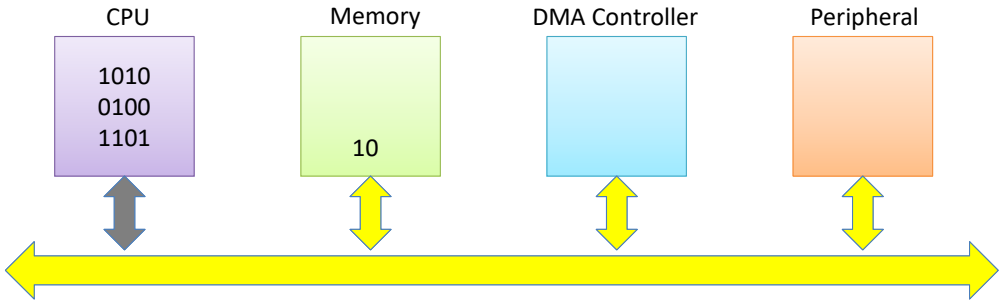


- Up till this point, our discussion on data transfer mechanism has always involve CPU as the controller that perform the data transfer task.
- As the amount of data increases, the time needed to transfer data increases as well. So it would appear that we are not using the CPU effectively as the main purpose of a CPU should be number crunching rather than transfer data from point A to point B.
- If we are able to relief the processor from the data transfer task and have it focus on algorithm processing, the overall system will be better utilised and efficient. This is where the DMA controller comes in.
- DMA Controller is a controller that is able to control the system bus to perform data transfer without help from CPU.
    - In doing so, it relief the CPU from doing data transfer so that it could focus on processing data.
- DMA controller has dedicated hardware that if well designed, would be more efficient than CPU when transferring data which requires complex address manipulation.
    - For example, de-interleaving the Left and Right channel of audio data.
- Operation wise, it is possible to have both the CPU execution and DMAC data transfer to happen at the same time if both of them are not using any common resources.
- But if they are using the same resource, e.g. system bus, then one of them will have to wait.
- With common resources, there will be a need for arbitration, and one way of

doing it is via priority. Whoever has the higher priority will get to use the system bus. In real word processor, this priority can be fixed, either CPU has higher priority compared to DMAC, or vice versa. Or it could very well be configurable.

- For our course, we will assume that there is only one system bus available in the processor. That means the CPU and DMA will have to fight of the usage of the system bus.

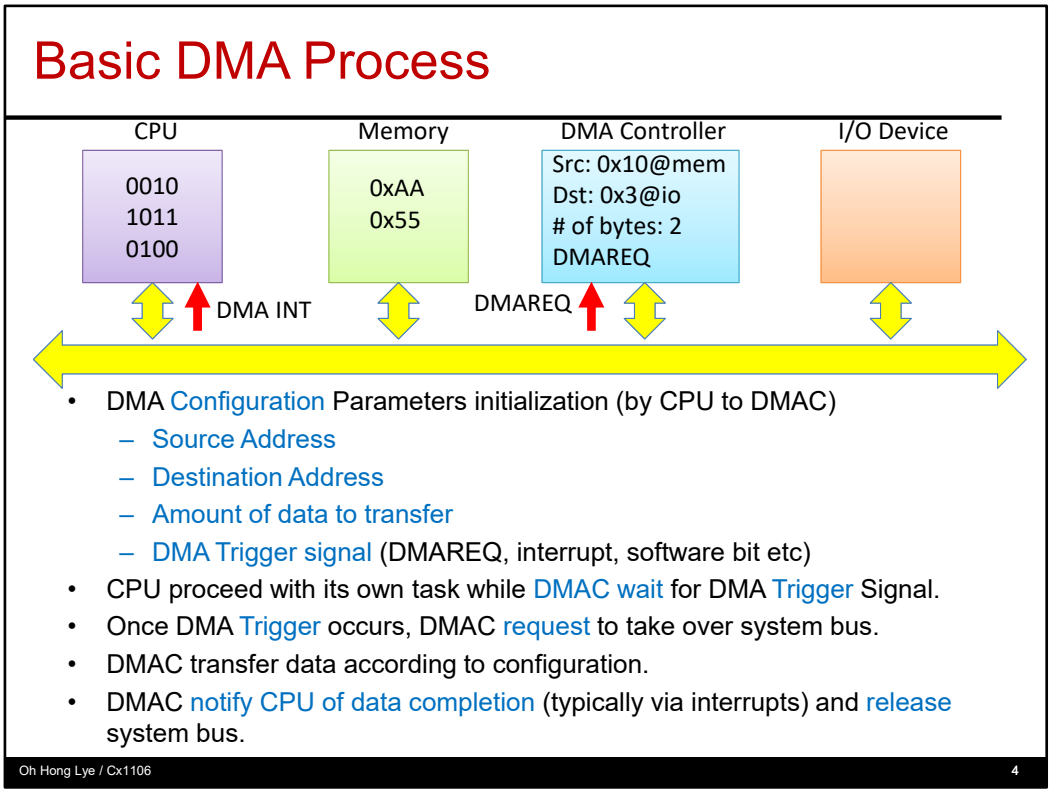
## DMA Controller (DMAC)




- DMAC is a Data Bus Controller module that performs data transfer independent of CPU, between Memory and memory, I/O Peripheral and I/O peripheral, Memory and I/O Peripheral.
- Generates address and initiates read/write operation between devices mentioned above.
- Note that 'Peripheral' here could be internal or external peripherals
- The DMAC shown here uses the Fetch and Deposit DMA mechanism where the data goes into the internal buffer of the DMAC before being transferred to the Destination.



- A DMA controller, DMAC in short, is able to transfer data between various modules within a computer system.
  - It does this by generating the addresses and initiate the read/write controls.
- The DMAC shown here is known as a Fetch-and-Deposit type of DMA, where the data from the source will be transferred into the DMAC before it is transported to the destination.
- The following animation illustrates how bus is shared between CPU and DMAC.
  - The CPU may be doing some operations that doesn't require the system bus.
  - So DMAC is able to perform data transfer between the memory and IO device.
  - As and when the CPU needs the system bus, control will be given to the CPU.
  - And after the CPU completes its data transfer, the DMAC can continue its operation.
  - In real world design, the bus priority between the CPU and DMAC may differ from the illustration.
  - In this example, we are assuming that CPU has a higher priority than DMAC.




- Now that we have some idea of what a DMA is, let's walk thru a DMA process to understand
  how a DMAC is configured and the various control signals involved.
- In order for DMAC to transfer data, it needs a few key parameters. Typically, these parameters are configured by the CPU to the DMAC.
  - Source Address of Data
  - Destination Address of Data
  - Number of words of Data to transfer
  - And the trigger signal to tell DMAC when to start the transfer
- After the DMAC is configured, it'll just wait in IDLE state for the trigger signal.
- Once the trigger signal comes in, it'll request for the system bus and start the data transfer.
- When all the data has been transferred, DMAC will release the system bus and issue a DMA interrupt signal to the CPU, informing CPU that the data is ready to be processed.

# DMA Mode of Operation

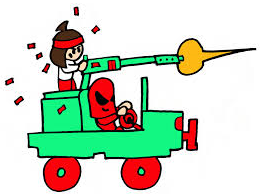
- Different processors has different DMAC design, typically with slight variance in terms of how they transfer data and the data type (Audio, Video etc) they are optimized for.
- In general, DMAC could transfer the block of data they are tasked to transfer in the following mode
  - Burst
  - Cycle Stealing
  - Transparent
- Do note that you may encounter DMAC design that varies from the basic mode of operation discussed here but general concept will still apply.
- In fact, majority of the time, you'll encounter DMAC design that uses a combination of the modes described



- Different DMAC may differ in the way they transfer data.
- There are three basic data transfer modes that DMAC typically use, namely, Burst, Cycle Stealing and Transparent. We will get to the details of each mode in later slides.
- Note, however that in actual processor, it is likely that you will encounter DMAC that uses a combination of these modes.

# Burst Mode

- The DMA controller gains control of the system bus and transfer multiple units of data before returning control of the bus to the CPU.
- Note that the burst could be the entire block of data DMAC is tasked to transfer, or a subset of the block, i.e. it may take a few burst to complete the transfer of the entire block.
- CPU may continue to operate as long as it does not need access the particular system bus that DMAC is using.
- If CPU needs to access the ssytem bus, CPU may be suspended till DMAC has completed its data transfer.
- Fast data transfer rate but may render the CPU inactive for longer period of time.






- First, let's take a look at the Burst Mode.
- In Burst Mode, DMAC will transfer multiple units of data when they take control of the system bus.
  - This could be the entire block of data the DMAC is tasked to transfer, OR
  - It could be a subset of the block, that means it probably take the DMAC a few burst to complete the block transfer.
- CPU can continue with its operation as long as there are no conflict in resource utilisation, for example the system bus.
- However, if the CPU needs system bus during the time when DMAC is doing the transfer, the CPU may need to wait till DMAC complete its transfer.
- Burst mode would result in faster data transfer rate but may potentially cause the CPU to be suspended for extended period of time.

# Cycle Stealing

- DMAC releases the system bus after transferring one unit of data.
- Depending on processor design, DMAC tends to execute the data transfer
  - Between CPU instructions
  - Between Pipeline stages
- CPU may be suspended if it need to access the system bus but suspend time is shorter as only one unit is transferred at one time.
- Transfer rate is slower than in Burst Mode but will CPU will only be inactive for very short period of time.
- Favoured in application which requires CPU to be responsive e.g. real-time security status monitoring.

Between CPU Instr

| CPU | Instr1 | Instr2 | | Instr3 | | Instr4 |
|---|---|---|---|---|---|---|
| DMAC | | | Data | | Data | |

Between Pipeline Stages

| CPU | Fetch Instr | Decode Instr | | Fetch Operand | Exe Instr | |
|---|---|---|---|---|---|---|
| DMAC | | | Data | | | Data |



- If DMAC is using Cycle Stealing mode, it will only transfer one unit of data after taking control of the system bus, after which it will release the bus back to the CPU.
- The name cycle stealing comes about as the DMAC will try to adjust its use of system bus by virtue of the processor operation.
  - For example, it will tend to execute the data transfer
    - Between CPU instructions, OR
    - Between Processor Pipeline stages
- Any resultant suspension of CPU execution is kept short because only one unit of data will be transferred so CPU will be more responsive to other events.
- The time available for data transfer is shorter than that of the Burst Mode, therefore, data transfer rate will be lower as well.
- As this method is less disruptive to CPU operation compared to Burst Mode, it is preferred in system that required CPU to be more responsive, for example, real-time monitoring.

# Transparent Mode

- DMAC transfer data only when CPU is not using the data bus.
- Zero impact to CPU performance in terms of data bus access.
- Potentially the slowest transfer rate among the three modes.
- More Complex hardware needed to detect when CPU is not using the data bus.
- The CPU activity detection technique could also be used by some processors as a cue to trigger a burst transfer only when CPU is not using the system bus.



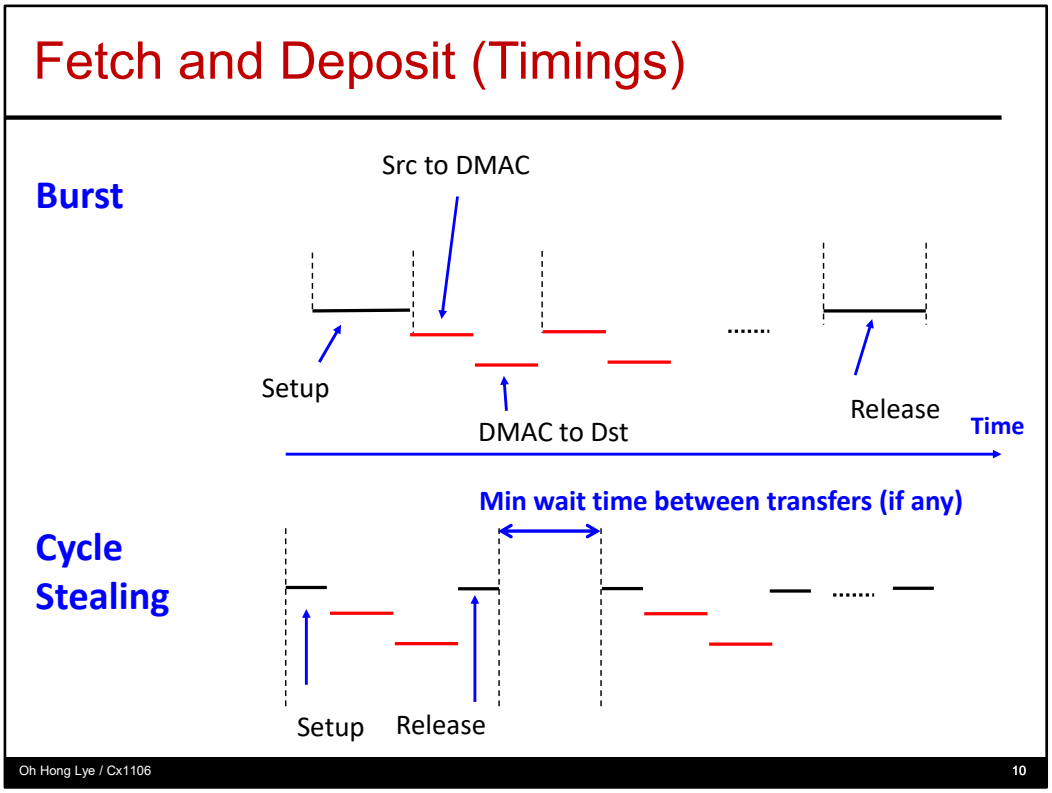
- Lastly is the Transparent Mode, which can be view as the extreme od the cycle stealing.
- In this ode, the DMAC will only start a transfer if it detected that the system bus is not being used by the CPU.
- This incidentally result in the least disruption to the CPU but also have the slowest data transfer rate among the three DMA transfer mode.
- In addition, it would require some complex hardware that allows the DMAC to snoop the system bus in order to determine the activity level on the bus.

## Comparison of DMA Transfer Modes

- **Burst**
  - Allow faster data transfer
  - CPU may be suspended for longer period of time
  - May not be suitable for Real-time application
  - Suitable for application where transfer is bursty e.g. HDD file transfer.
- **Cycle Stealing**
  - Interleaving DMA data transfer with CPU instructions allow CPU to continue executing its program while DMAC is doing the data transfer.
  - CPU performance lower due to interleaving of DMA transfers, but would still be more responsive than in Burst mode.
  - Slower data transfer rate than Burst mode.
  - Suitable for Real-time application.
- **Transparent**
  - Potentially would not affect CPU performance at all.
  - Slowest data transfer rate but best CPU respond
  - More complex hardware needed to detect when CPU is not using the bus.



- To summarise.
- Burst Mode allows a high transfer rate but may potentially suspend the CPU for extended period of time.
  - Suitable for application where data transfer is bursty in nature, such as HDD transfer.
- Cycle Stealing mode interleave its data transfer with CPU instructions so it has less impact to the CPU performance.
  - This allows the CPU to be more responsive, but at a trade off of lower DMA transfer rate.
- Transparent mode DMA does not affect the CPU performance at all as it only work when CPU is not using the system bus.
  - However, implementation may be more complex as it now has to predict/detect when the system bus is free.



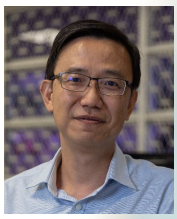

- This is an additional slide to illustrate, from timing diagram perspective, the DMA operation under different modes.
- The slide shows the timing diagram for the burst and the cycle stealing mode.
- In the burst mode
    - Recall that once the DMAC have the control for the system bus, it'll transfer the entire block of data to the destination.
    - Timing wise
        - Setup correspond to the time DMAC takes to request control of the system bus from the CPU
        - This is followed by transfer of data
            - Each unit of data is transfer in two stages,
                - first from Source to DMAC internal buffer
                - Second from DMAC to the destination.
        - After the last unit of data is transferred to the destination, the DMAC will release the bus control back to the CPU.
- In Cycle Stealing mode
    - The DMAC will request control of the system bus from the CPU, transfer one unit of data and release the bus back to the CPU.
    - The DMAC may also be required to wait for a certain time duration before it can request for the bus control again.
    - The timing is as shown in the diagram
        - The DMAC will request for the bus control, which is 'Setup' in the diagram, it will than transfer the data, release the bus back to CPU

and wait for a fixed duration before requesting for the bus again.

- Take note of these timing diagrams as you will need them to complete your tutorial questions.
- This is the last slide of the DMA section.

# Cx1106
# Computer Organization and Architecture

## Computer Memory Introduction



Oh Hong Lye
Senior Lecturer
School of Computer Science and Engineering, Nanyang Technological University.
Email: hloh@ntu.edu.sg

- We will be touching on the topic of computer memory in the next few videos.
- Memory is an important sub system in the computer and has a big impact to the overall system performance.

# Computer Memory

**Semiconductor-based**

Solid-State Drive

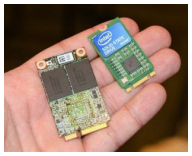
SODIMM

Memory IC Chips

CF, SD, miniSD, microSD, MMC

Solid-State Drive

Solid-State Drive



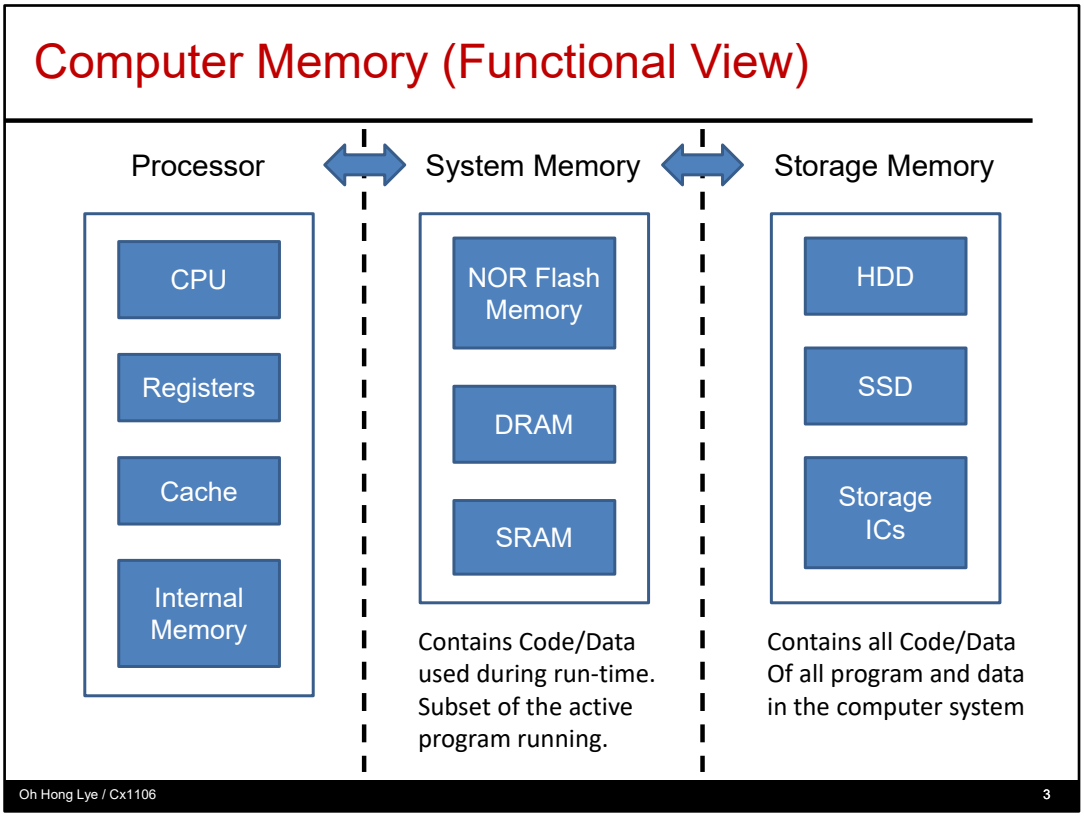
Thumb Drive

**Magnetic-based**

Magnetic HD

Oh Hong Lye / Cx1106 2

- Computer memory is a key component in the computer system as this is where all the information and code are stored.
- There are many types of memory
    - Semiconductor based
    - And the Good old Magnetic HDD
    - Although HDD is fast being replaced by SSD (solid state drive) in laptop these days, you may want to know that HDD is still very much involved in your daily life, in the form of cloud storage.
        - The iCloud, Google Drive, drop box cloud storge that you use stores information in Data Centers, and HDD is still the main storage element in Data Centers. We will get to more of that in later slides.
- One question, among these memories, which are volatile and which are non-volatile?
- To know the answer, we have to first understand what is volatile and non-volatile memory.

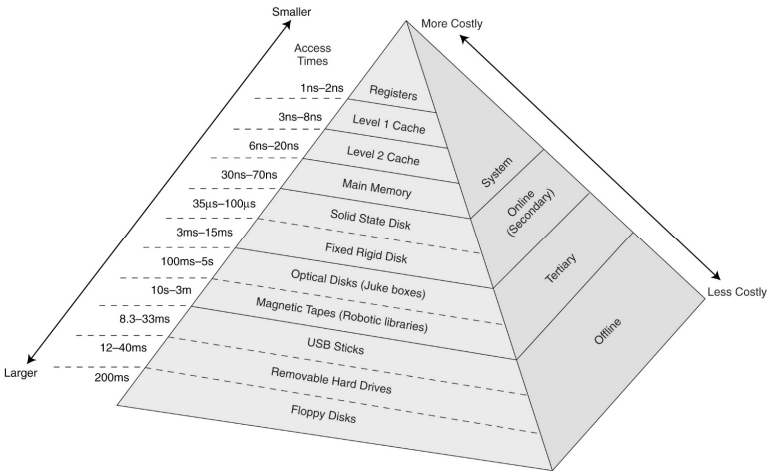



- Before we go into the memory type details, let's go through some basics on the function of a memory in a computer system.
- When we look at memory from functional point of view, there are two types of memory in a computer system.
    - System and Storage memory
        - System memory is where you run the code directly from. But it is likely that system memory will not be large enough to contain all the application program and data you have, as such you will need a larger space for these information.
        - This is where Storage memory comes in. The entire collection of Program and Data is stored in the Storage memory.
        - Another common attribute of storage memory is that it is typically non-volatile as it needs to preserve the program and data even after the computer power down.
        - For this course, where system memory is concern, we only have three memory types to select from: NOR Flash, DRAM and SRAM, the primary reason behind is that these three memories support XIP, meaning you can run code directly from these memory. No worries if you are not familiar with these names and terms, we will get to more details in later slides.
    - The processor itself will also have its own internal memory and other memory elements such as Cache and Registers.
    - Operation wise, the computer will power up with its OS and applications resided in the Storage memory, these are then transferred over to the System

memory so that the processor can execute the code. The various internal memory elements may also be used.

# Memory – Cost vs Function Trade Off

- The memory and storage devices may be organized like a pyramid.
- The pinnacle has the fastest access time, but is also more costly.
- Memory Access Time below are only for illustration. These changes with improvement in technology.




Oh Hong Lye / Cx1106 4

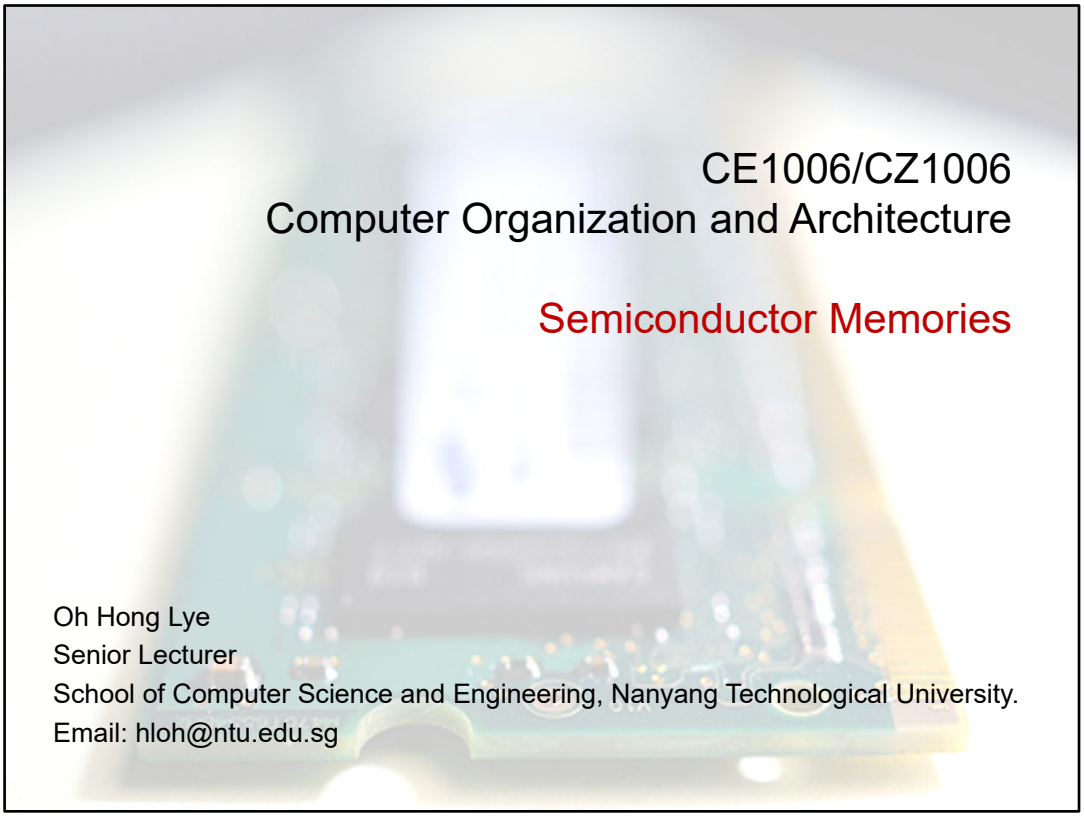
- When choosing which type of memory to use, we typically have to consider the cost vs function trade off.
- IN this slide, different memory types are organised in the form of a pyramid.
    - The fastest memories are at the top but they are also the most expensive so their size are typically smaller
      as compared to those at the bottom.
- Note that the memory access timing are for illustration only, these changes with improvement in technology.

## Volatile and Non-Volatile Memory

- Volatile
  - Data is lost when electric power is removed.
  - Temporary storage.
  - Typically used as system memory.
  - We will look at Random Access Memories such as Static-RAM (SRAM) and Dynamic-RAM (DRAM).
- Non-volatile
  - Data is retained even if electric power is removed.
  - Permanent storage.
  - Typically used as main storage.
  - We will look at FLASH, magnetic hard-disk specifically.

Oh Hong Lye / Cx1106 5

- Back to our discussion on Volatile and Non-Volatile memory
- Volatile memories has the attribute that
  - The data will be lost when power supply to the memory is cut off.
  - So the information can only last if there is power and is therefore used as a temporary storage.
  - System memory are typically implemented using volatile memories. For this course, we will look at SRAM and DRAM
- Non-volatile memory, on the other hand, is able to preserve the data even when the power is cut.
  - So it is used as a permanent storage. We will look at Flash and Magnetic Hard Disk in our course.




- The first group of memories we are going to check out is the Semiconductor memories.

# Semiconductor Memories

- Memories based on semiconductor integrated circuits (IC).
- Used as processor internal memories and system memory in a computer system
- Processor internal memories
  - Registers
  - Buffers
  - Cache
  - Internal System and Storage Memory (SRAM, DRAM, Flash based)
- Types of memory
  - SRAM
  - DRAM
  - Flash Memory
  - Solid State Drives (based on Flash Memories)



- Semiconductor memories are based on semiconductor IC, short for integrated circuits.
- They are used as processor internal memory elements such as registers, buffers, cache and internal system or storage memory.
- Types of semiconductor memories we will touch on in this course are SRAM, DRAM, Flash Memory and SSD, which are bulk storage memory mainly consist of Flash memory ICs.

# VOLATILE MEMORY



This section is on Volatile memories, which are memories that lose its data if the power is shut down.

# Random Access Memory (RAM)

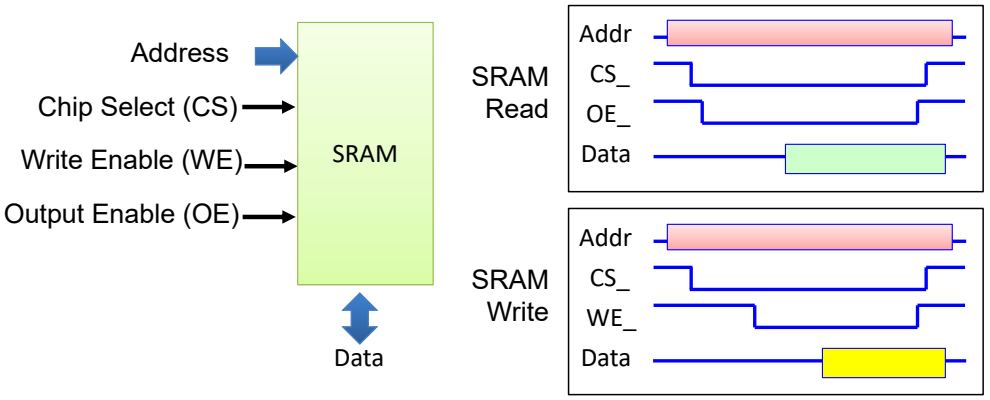
- Static RAM (SRAM)
  - Static Random Access Memory
  - Data stored as long as supply is applied.
  - Large (4 to 6 transistors per cell).
  - Fast.
  - Low power consumption (active and standby)

- Dynamic Random Access Memory (DRAM)
  - Periodic refresh required.
  - Small (1 to 3 transistors per cell).
  - Slower.
  - Higher power consumption due to need for periodic refresh operation to maintain data integrity of memory cells.



- In this course, we will look at two types of volatile memory, SRAM and DRAM.
- Static Random Access Memory, SRAM in short
  - Is able to retain its data integrity as long as power supply is maintained.
  - It has a relatively larger layout consisting of 4-6 transistors per cell, typically.
  - It access time is fast
  - And power consumption for both active and standby mode is low.
  - All the above are wrt DRAM
- Dynamic RAM, DRAM in short is another type of volatile memory you will frequently encounter.
  - It consist of a capacitor and one transistor.
  - This give rise to a smaller layout compared to SRAM, cost per bit of DRAM is therefore lower than SRAM
  - The use of capacitor as a storage element, however, requires the cell to be refreshed periodically in order to maintain data integrity.
  - That leads to a higher power consumption and slower access time when compared to SRAM.

# Static RAM (SRAM) Access




| | | |
|---|---|---|
| **Addr** | Specifies Address of memory location. | |
| **Data** | Data to be read/written. | Typically 8,16 or 32 bits. |
| **CS** | Control signals | Chip Select (Enables or disables the chip). |
| **WE** | | Write Enable (Allows data to to be written) |
| **OE** | | Output Enable (Allows SRAM to output data) |

Oh Hong Lye / Cx1106 3

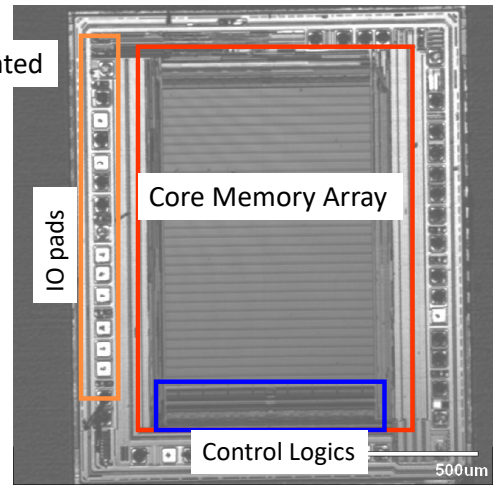
- To perform read and write operation on the SRAM, we need the following signal lines.
  - Address, which specify which memory location will be read or written
  - Chip select which is the main switch for the SRAM chip
  - The Write Enable pin which tells the SRAM that the processor is performing a write operation
  - The Output Enable pin which allows the SRAM to output data from its data bus.
  - And lastly the data bus.
- During a SRAM read, the uP will first output the address info, followed by the CS and OE. The SRAM will then output the data requested on the data bus.
- I'm showing these signaling sequentially to walk you thru the whole process. In actual case, these signaling happen simultaneously.
- Similarly for SRAM write, uP wil first output the Address, followed by the CS and WE so that SRAM can prepare for a write operation. uP will then output the data on the data bus and SRAM will latch the information in.
- And again, these signals are output simultaneously.

# SRAM Chip DeCap

Commercial SRAM Chip
Cypress CY62128, 1 Mbit


Decapsulated

Decapsulated silicon die
Laser scanned image, 5x magnification

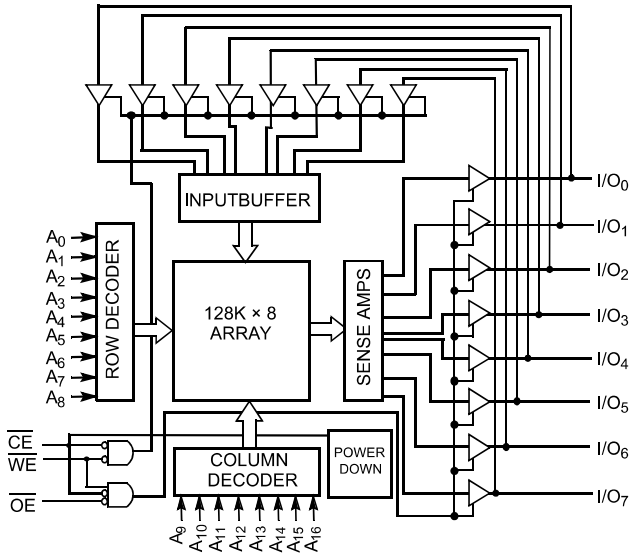



- **Memory** array takes up **most** of the **real estate**.
- Leading Edge Semiconductor Process is usually first tested with Memory Design.

Oh Hong Lye / Cx1106 4

- Next, let's dig deeper into the SRAM chip.
- This show what it looks like under the packaging. You can see rows and rows of uniform design, which is actually the SRAM cells.
- In most system or uP, memory usually takes up the most real estate.
- As such, they are usually the target candidate to test the leading edge semiconductor process technology.
- Which is why leading memory makers such as Samsung are usually the leaders in semiconductor process technology.
- The other group of companies that leads in the process technology are the processor makers such as Intel and AMD.

# SRAM Internal Circuitry



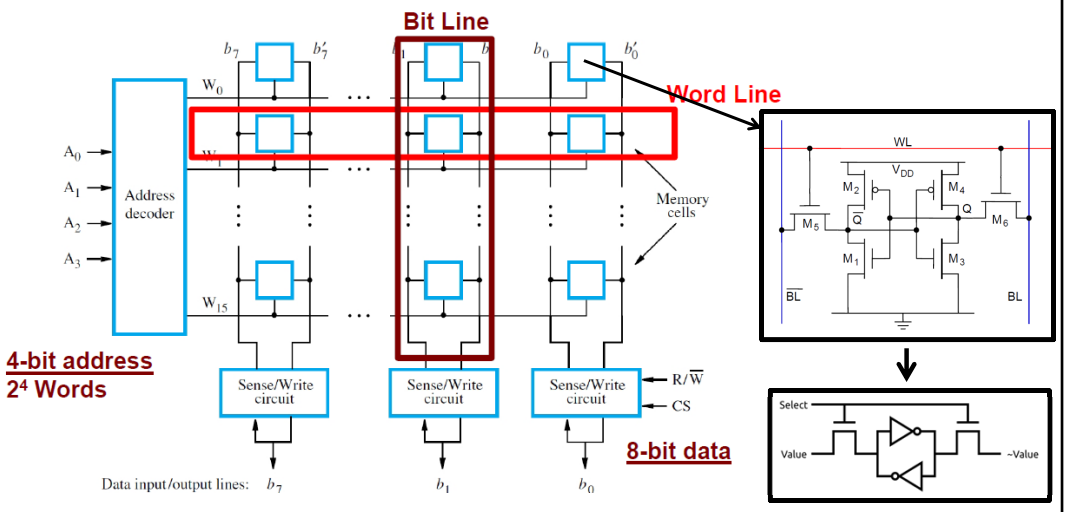

- Memory array
- 1M SRAM cells for the chip shown.
- Control circuitry
- Decoders
- Sense amplifiers
- Input/Output multiplexers



- This is how SRAM internal looks like functionally.
- The example shown here is a 128KByte SRAM chip
  - It consist of an array of 1 Mega (2^20) SRAM memory cells, each cell contains 1 bit.
  - The address are fed into decoders which will enable 8 of the cells so that read or write operation can be carried out.
- For SRAM, you don't have to be too concern about the detail design such as the role of decoders, multiplexers and sense amplifiers, its not in the syllabus.

# SRAM Cell

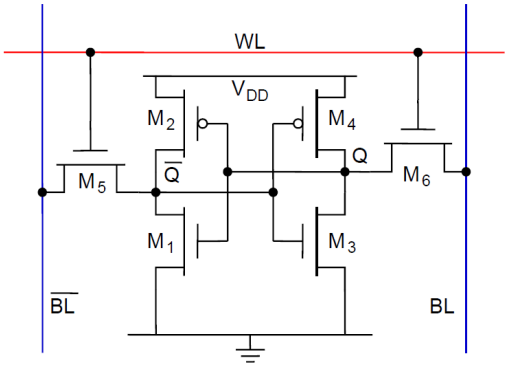
- Memory cells are organized in arrays (rows), and are accessed via the word lines and bit lines.
- Each individual SRAM Bit consist of 6 Transistors (typical design).





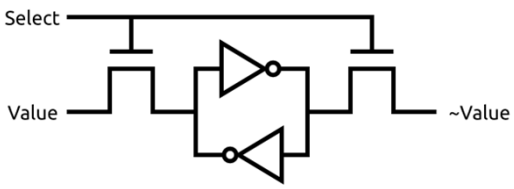

- Diving in further, this is how the SRAM cells are arranged.
- The address info is decoded to enable a particular word of data, each word consist of one row of cells, each cell corresponding to one bit of data.
- Each SRAM cell has six transistors. It has a differential output.
- The digital logic equivalence of the transistor design is shown here.
  - Its basically two inverters with output of one connected to the input of the other, and vice versa.
  - This ensure that the logic state that is stored will remain unchanged as long as the gates are powered.
- I do not expect you to know the detail transistor level layout design but you should at least know the function of the transistors and the logic equivalent of the 6-transistor design.

# SRAM Cell




- There are six transistors
- M1, M2, M3, M4, M5 and M6
- Word line (WL) is derived from Address Decoder Output. It controls the Read/Write process
- The actual data are placed on the differential bit lines (BL and BLB). This is connected to the Data Bus.

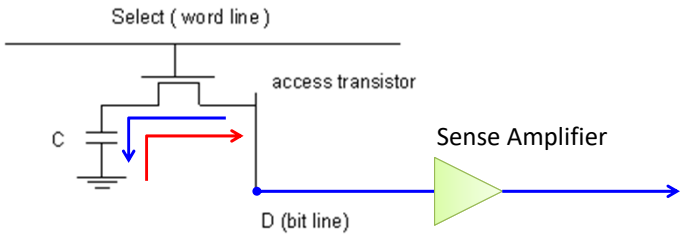
- The data bit is stored in M1, M2, M3 and M4 (which is equivalent to two invertors connected as shown on the right).
- M5 and M6 are pass transistors.




Oh Hong Lye / Cx1106 7

- As mentioned in the previous slide, SRAM cell consist of 6 transistors.
- The word line is derived from the address info and controls which sets of bit cells are enabled
- The bit line output the actual data.
- M1 to M4 forms the invertor logic we mentioned earlier.
- M5 and M6 controls the data in and out of the cell.

# Dynamic RAM (DRAM)




- Single Transistor Design
- DRAM uses a capacitor as its storage element.
- The Transistor is used to control charges flowing in and out of the capacitor during the Read and Write process.
- Write Process
  - To store a Logic '1': Enable the Transistor, transfer charge into capacitor.
  - To store a Logic '0': Enable the Transistor, discharge the capacitor.
- Read Process
  - Enable the Transistor. Measure the capacitor charge using a sense amplifier.



- Next we are going to look at Dynamic RAM. DRAM in short.
- DRAM cell only consist of one single transistor and a capacitor.
- The capacitor is used to store the charge while the transistor controls the flow of charges in and out of the capacitor.
- First, let's look at the Write Process
  - To store a logic '1', we first enable the pass transistor and transfer some charge to the capacitor to charge it up.
  - If we want to write a logic '0' instead, then we would need to enable the pass transistor and discharge the capacitor.
- Similarly, for the Read Process
  - The pass transistor has to be enabled and the capacitor charge status is detected using a sense amplifier, in order to determine the logic stored in the DRAM cell.

## DRAM – Maintaining Data Integrity

- DRAM Read Process destroys information stored on capacitor
- The process of measuring charges on a capacitor also effectively discharged it i.e. data is destroyed.
- Hence, the original data has to be re-written back after every read.
- Periodic refresh is needed as the stored charge "leaks" with time.
- The basic DRAM is more or less obsolete in the market today. It is replaced by its synchronous version called Synchronous DRAM (SDRAM).
- Difference between SDRAM and DRAM is that the former make use of a clock signal from the host to synchronize data transfer, enabling faster transfer rate. SDRAM also has a pipeline architecture that allow faster, overlapping operations.
- Other enhancements of SDRAM includes its double date rate versions DDR, DDR2, DDR3 SDRAM, which could reach transfer speed of more than 2G transfers per second.



- DRAM is called Dynamic RAM because the charges stored in the DRAM cell tends to change, this compared to the SRAM,
  which can hold its charges indefinitely as long as there is power supplied.
- Let's take a look at this behaviour in more detail.
- In DRAM, a read process involves discharging the capacitor so in a way, the data stored is destroyed when performing the read operation.
    - That's why the original data has to be re-written after every read.
- On top of that, charges in a capacitor tend to leak with time. Hence, DRAM cells requires periodic refresh operation to hold the data.
- Note that the basic DRAM is more or less obsolete in the market today. It is replaced by the synchronous version called
  Synchronous DRAM, or SDRAM in short.
    - But basic concepts and features of the DRAM remains, such as the use of capacitor as main storage element and the need to have period refresh to maintain data integrity.
- The difference between SDRAM and DRAM is that
    - SDRAM use a clock signal from host to synchronise the data transfer
    - SDRAM also employs a pipeline architecture that allow faster, overlapping operations compared to conventional DRAM.
- The SDRAM variants you see today are enhancement over the single rate SDRAM. Known as DDR2, DDR3, DDR4 etc which is able to sustain higher data transfer rate.
- That conclude the discussion on volatile memory.

# DRAM – Maintaining Data Integrity

- DRAM Read Process destroys information stored on capacitor
- The process of measuring charges on a capacitor also effectively discharged it i.e. data is destroyed.
- Hence, the original data has to be re-written back after every read.
- Periodic refresh is needed as the stored charge "leaks" with time.
- The basic DRAM is more or less obsolete in the market today. It is replaced by its synchronous version called Synchronous DRAM (SDRAM).
- Difference between SDRAM and DRAM is that the former make use of a clock signal from the host to synchronize data transfer, enabling faster transfer rate. SDRAM also has a pipeline architecture that allow faster, overlapping operations.
- Other enhancements of SDRAM includes its double date rate versions DDR, DDR2, DDR3 SDRAM, which could reach transfer speed of more than 2G transfers per second.



- DRAM is called Dynamic RAM because the charges stored in the DRAM cell tends to change, this versus the SRAM,
  which can hold its charges indefinitely as long as there is power supplied.
- Let take a look at this behaviour in more detail.
- In DRAM, a read process involves discharging the capacitor so in a way, the data stored is destroyed in the process.
    - That's why the original data has to be re-written after every read. This process is called pre-charging.
- Also, charges in a capacitor tend to leak with time. Hence, DRAM requires periodic refresh operation to hold the data.
- The basic DRAM is more or less obsolete in the market today. Being replaced by its synchronous version called
  Synchronous DRAM, or SDRAM in short.
- Other enhancement

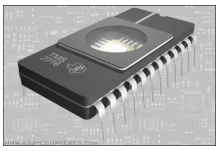
# NON-VOLATILE MEMORY



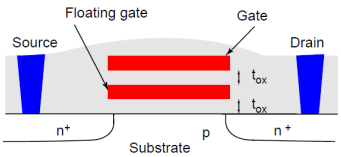
- This section is on non-volatile memory.
- These memory is able to retain the data even when its power is cut off.

# EPROM and EEPROM

- EPROM (Erasable Programmable ROM)
  - Earliest floating gate transistors are implemented as Erasable Programmable ROM (EPROM) devices.
  - Need to put device under ultra-violet (UV) light to erase the stored program.



[Source] www.old-computers.com



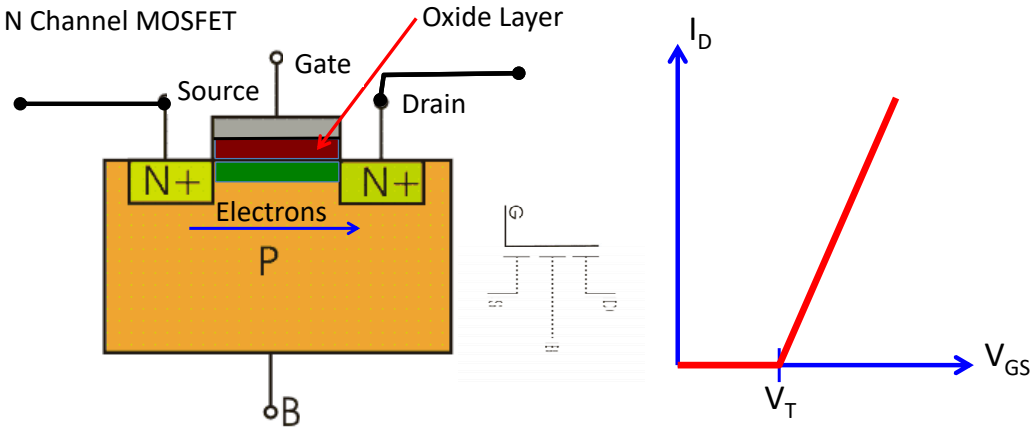

Device cross-section

- EEPROM (Electrically Erasable PROM)
  - Advancement in process technologies make it possible to reduce the oxide thickness ($t_{ox}$).
  - Can electrically program or erase device.
  - Hence, named Electrically Erasable Programmable ROM ($E^2$PROM).

Oh Hong Lye / Cx1106 2

- Two common non-volatile memories you can find in the market are EPROM and EEPROM
  - Which are short form for
    - Erasable Programmable ROM and
    - Electrically Erasable PROM respectively
- EPROM is one of the earliest erasable non-volatile memory that uses a floating gate design.
  - This device can be programmed electrically but requires UV light to erase its data.
  - The packaging has a small transparent window that allows UV to get to the floating gate to remove the charges trapped there.
- As technology advances, we are able to reduce the thickness of the gate oxide.
- This led to the creation of EEPROM, which allows the charges to be erased electrically.

# MOSFET

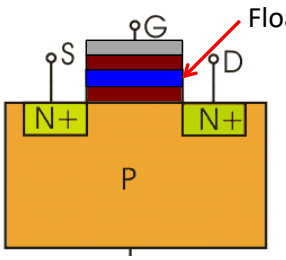



- MOSFET (Metal-Oxide-Semiconductor Field Effect Transistor)
- For N-Channel MOSFET, if Gate-Source Voltage (Vgs) > Threshold Voltage (Vt), a conductive channel of electrons (inversion layer) will be formed and current will flow if a positive voltage is applied across Drain and Source.

Oh Hong Lye / Cx1106 3

- We mentioned a number of times this term called floating gate in the previous slide.
- So what is a floating gate?
- To know that, let's start by looking at a MOSFET, or Metal-Oxide-Semicon Field Effect transistor.
- For a N-channel MOSFET, it remains non-conductive until the Gate-Source voltage exceeds a certain positive voltage threshold.
- When the gate voltage is sufficiently positive, it will be able to attract more electrons which are negatively charged.
- A layer of electrons will be formed under the oxide and current will flow between the Source and Drain terminal of the MOSFET, the electron layer is illustrated in this slide by a green bar below the oxide layer.
- You can see from the graph that current is zero till Vgs exceed Vt. Afterwhich current will flow.

# Floating Gate Transistor (FGT)

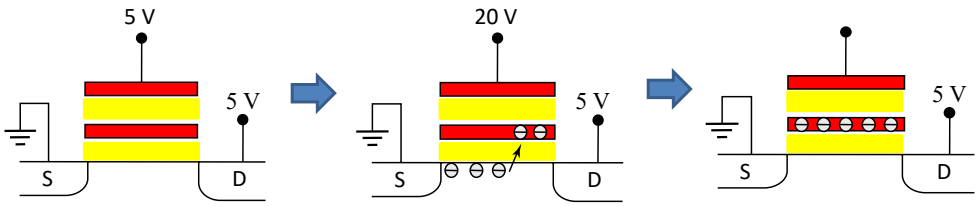



Floating Gate

**Erased State**
No Excess Electrons in Floating Gate

**Programming**
A large positive voltage applied at the gate, causing excess electrons to tunnel through the oxide and embed in the floating gate

**Programmed State**
Excess Electrons are trapped in Floating Gate even after power off




Erased State | Programming | Programmed State

Oh Hong Lye / Cx1106 4

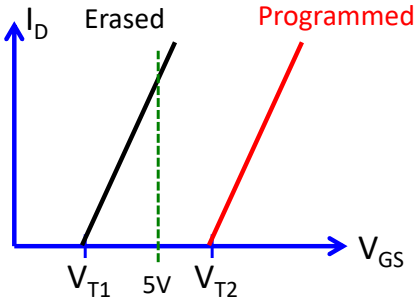
- Now that you know how a MOSFET works, let's take a look at the floating gate transistor, FGT in short.
- In the FGT, an additional layer of the metal gate and the oxide is inserted. So the gate that is sandwiched between the two oxide layers appears to be floating as its not electrically connected to anywhere.
- There are two states that the FGT may be in.
- One is the erased state.
    - This correspond to the state where there are no excess electrons in the FGT, i.e. it is electrically neutral.
- Then there is the Programmed State
    - Which is the state when there are excess electrons in the FGT.
- In order to program the FGT, a large positive voltage e.g. 20V is applied to the gate.
    - This will cause some electrons in the substrate of the FGT to tunnel through the oxide and embed into the floating gate.
    - Note that the voltage value used in this slide are for illustration purpose only, actual value may differ depending on the semiconductor process.
- After programming, the FGT is in the programmed state and the excess electrons in the floating gate will continued to be trapped there even if the power is removed.
- The presence of excess electrons in the floating gate has effect on the threshold voltage of the FGT and this attribute can be used to implement memory bit cell with non-volatile property. More details in the next slide.

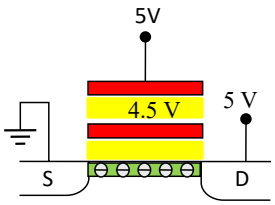
# FGT Threshold Voltage

As an illustration, let's say a 5V input at the Gate of a FGT in erased state is sufficient to turn ON the FGT.

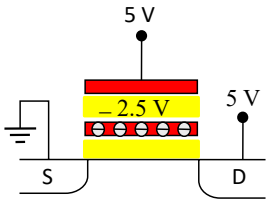
"Programming" increases the threshold voltage of Floating Gate Transistor so the same 5V is now unable to turn ON the FGT.

Voltage values used are for illustration purpose only.




**Erased State**




**Programmed State**

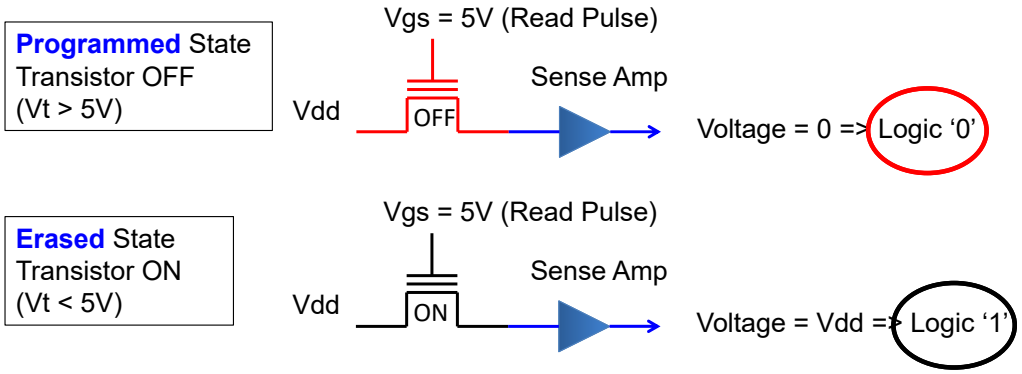
Oh Hong Lye / Cx1106 5

- First, let's look at how the presence of excess electrons affects the threshold voltage of the FGT.
- Remember that a MOSFET becomes conductive when the gate source voltage or Vgs is larger than a certain threshold voltage Vt.
- Similarly, for FGT, It'll be conductive when its gate voltage crosses a certain threshold.
    - As an illustration, let's say that when the FGT is in erased state, i.e. the floating gate has no excess electrons, a 5V input at the gate is able to attract enough electrons to form the conductive layer. So the FGT is turned ON.
- Now, if the FGT is in the programmed state instead,
    - Which means excess electrons are present in the floating gate, and these negatively charged electrons will lower the resultant voltage potential, meaning user now has to apply a larger voltage in order to attract the same number of electrons to form the conductive layer.
    - Applying the same 5V will therefore not be able to do the job, as illustrated in the diagram, the same 5V input at the gate will result in a much lower voltage potential at the floating gate when the FGT is in programmed state. This as mentioned, is due to effect of the negative charges of the excess electrons.
- So to summarise, FGT in erase state has a lower Vt than when it is in programmed state.
- This attribute of the FGT can be used to implement a memory element.
    - We have previously discussed about the way to detect whether a FGT is in erased or programmed state.
        - That can be done by injecting a voltage between the two threshold, VT1 and VT2 in the graph.

        - If the FGT in ON, its in erased state, if it is OFF, it is in programmed state.
    - If we further defined that the FGT stores a Logic ‘1’ and Logic ‘0’ when it is in erased and programmed state respectively,
        - then we have a 1-bit memory cell in the form of the FGT.
            - To write a ‘1’, we erase the FGT
            - To write a ‘0’, we programmed the FGT.
            - To read the content, we apply a 5V.
- Note again the 5V here and other values are for illustration purpose only.
- This concept forms the basis of many of the  non-volatile memory discussed in this course, e.g. EPROM, EEPROM and Flash.

## Reading of Stored Data in Floating Gate

- With a positive Gate Voltage (Vgs) = 5V
  - Transistor OFF if floating gate is programmed (contains charges).
  - Transistor ON if floating gate is erased (no charges).
- 5V value below is for illustration only. Actual Vt value in real world may varies depending on the doping of the transistors.

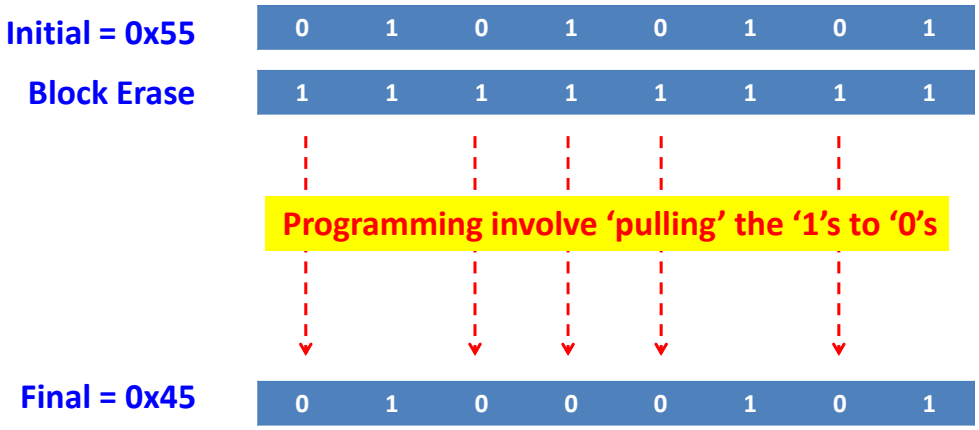



Oh Hong Lye / Cx1106 6

- Just to recap from the discussion in previous slide,
  - With a positive gate voltage, e.g. 5V, you find that FGT will remain OFF if it is in the programmed state.
  - Since FGT is off, the sense amplifier will detect a logic '0'.
- But if the FGT is in erased state, the threshold voltage will be lower and the FGT will be turned ON, leading to sensing of logic '1' at the output.
- Which is why, after a flash is erased, it'll contain all '1's in its memory. During programming, we are actually storing '0's into the various memory bits.
- Note that the 5V value here is just for illustration, actual Vt value may change depending on the doping of the transistors.

## Example of Programming a FGT based memory

- Flash 'programming' can only modify the cell content from '1' to '0'
- To modify the cell content from '0' to '1', an erase operation has to be done at block level
- Example: To Program a value of '0x45' when initial value in flash memory is '0x55'




Oh Hong Lye / Cx1106 7

- This slide runs through the process of programming a number into a FGT based memory such as Flash memory.
- Programming in FGT based memory is really modifying the cell content from '1' to '0'.
- If you need to write a '1' instead, that correspond to the erase operation.
- In FGT based memory, erasure are done at block level, where one block consist of multiple bytes.
- For example, if we want to write a value '0x45' to a particular memory location whose initial value was '0x55' in a Flash memory, we first have a erased the block in which the byte location is.
- This will set all the bits to '1's.
- After which, the actual programming involves setting to some of the '1's to '0's to form the value 0x45.

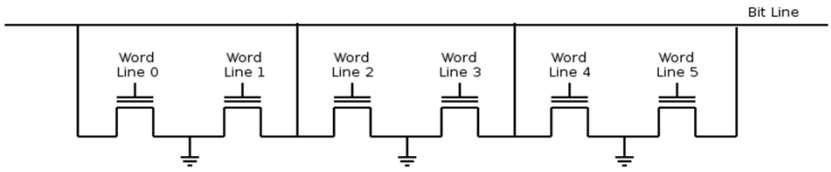
# Flash

- Based on similar floating gate technology as EEPROM.
- Can be erased in larger blocks size compared to EEPROM (which typically has smaller page/block size).
- Since Erase cycle is comparatively slower than other operations (Read/Write), Flash memory has a faster speed than EEPROM when performing write operations for large block of data.
- Flash also cost less than EEPROM.
- Suitable for system requiring large amount of non-volatile memory.
- Two main types of Flash in the market
  - NAND Flash
  - NOR Flash



- One of the most popular non volatile memory based on FGT is the Flash Memory.
- Flash memory can be erased in larger block size compared to EEPROM
- Since erase operation is typically slower than other operations, Flash memory has a faster access speed compared to EEPROM, especially when writing large blocks of data.
- Flash memory also cost less since each overhead module for erasing or programming cater for a larger block of memory.
- There are two main types of Flash Memory in the market today, namely the NOR and NAND flash. We will discuss in more details in later slides.

# NOR Flash

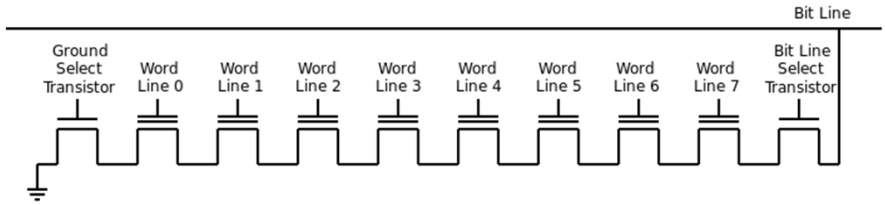



- Cell behaves like a NOR gate.
  When any one of the Word Line > Vt(Prog), Bit Line output = 0.
- Supports Execute in Place, i.e. Program code stored in Parallel NOR Flash can be executed directly without the need to transfer to internal RAM first.
  - Allows Random Reading of memory data using only Address information (no additional Commands needed).
- Need Special Commands (issue in Write mode) in order to perform operations other than Data Read. E.g. Program, Erase etc.
- Allows random word/byte programming. But erasure has to be done at block level. Typical Block size: 64KByte, 128KByte, 256KByte
- Use as system memory to store program code or general storage memory



- In NOR flash, the individual cell are connected with logics similar to that of the NOR gates.
- Meaning that if any of the wordline is higher than the threshold voltage, the bit line output will be LOW.
  - Similar to the NOR gate property that if any of the input is a '1', then output will be a '0'.
- NOR flash supports Execture-In-Place, XIP in short.
- This means you can store your program code in the NOR flash and execute the instruction directly from there without having to transfer the code to RAM.
  - This is possible because the NOR flash allows random reading of its data using only Address information.
  - Recall that in a processor, it only uses the Program Counter as the pointer to access instruction code, no other commands are involved.
- Note that the previous discussion refers only to NOR Read operation.
- For other operations such as write operation, procedure is more complicated and will require special commands and multiple steps operations.
- Note that even though NOR flash allows random word or byte programming, erase operation needs to be done at block level.
- Lastly, its ability to support XIP enable NOR Flash to be used as the system memory of a product. It could still be as a storage memory if user wants to.

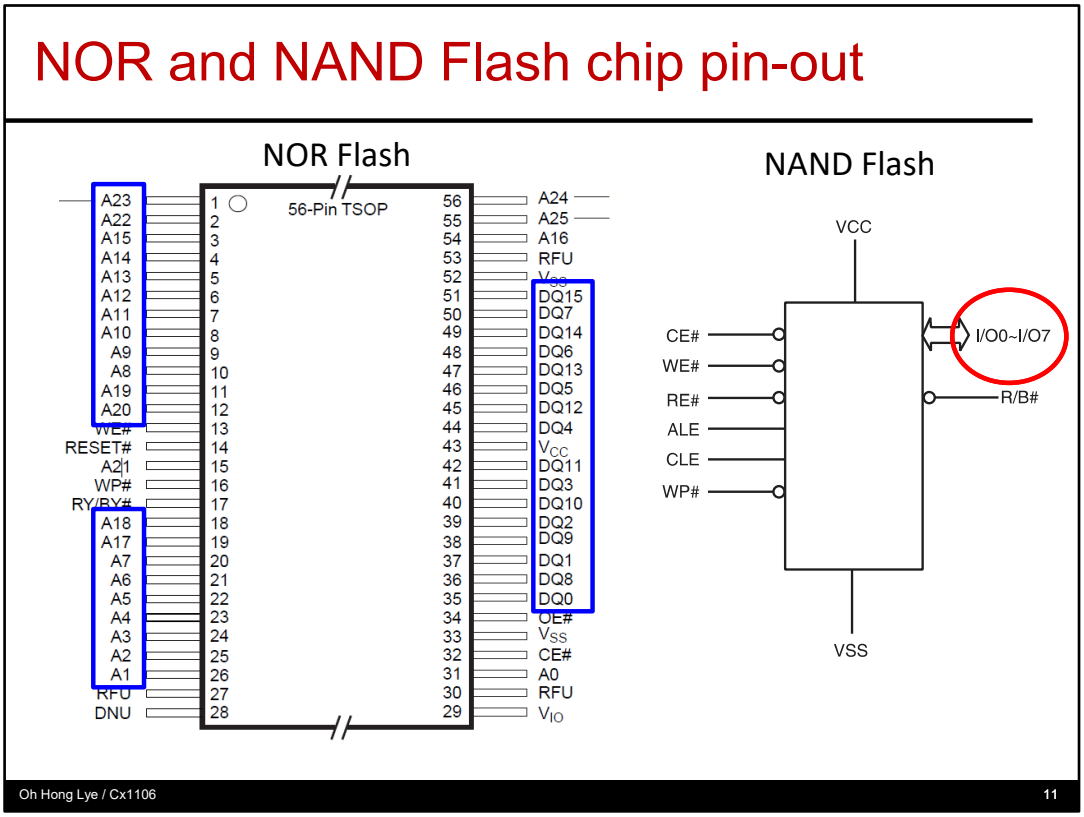
# NAND Flash




- Cell behaves like NAND gate.
  - Bit line output = '0' only when ALL Word Line > Vt(Prog).
- Does not support execute in place operation.
  - Data has to be accessed one page at a time.
  - Command issued to open a particular page, followed by which byte(s) is/are needed in the page.
  - NAND chips uses a single bus to carry Address and Data.
- Lower Cost per Bit than NOR. Used mainly as Main Storage Memory.
  - On Board Main Storage, USB Flash Drive, SSD etc



- Similarly, Nand flash got its name from the fact that the behaviour of its design has a lot similarilty with the property of a NAND gate.
  - The Bit line output is '0' only when all Word Line input > Vt.
- Unlike NOR Flash, NAND flash does not support XIP.
  - Data is accessed at page level. Each page consist of multiple bytes of data, multiple pages formed a block.
  - NAND flash uses a single IO bus to transfer Command, Address and Data information.
    - It's a multi-step process where the Processor will first need to send a COMMAND to the NAND Flash telling the memory what operation will be carried out, e.g. Read. It then followed up transferring the Address information and lastly the target data will be transferred.
    - As seen, the process is much more complicated does not support XIP.
- NAND flash's layout, however, has less connecting wires and therefore can be compact to a higher density, compared to NOR Flash.
  - This lower cost per bit advantage result in NAND flash being the memory of choice for mass storage devices.
  - Examples include USB Flash Drive, SSD, Phones and Tablets.




- This slide illustrate the difference between the NOR and NAND Flash interface.
- NOR Flash has a very simple interface consisting of Address and Data lines. The processor provide the Address of the target location and the NOR Flash supplies the data.
- NAND Flash, on the other hand, has only one IO bus so all the COMMANDS, ADDRESS and DATA information have to go through this bus.

# System vs Storage Memory

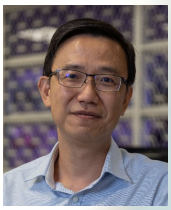
- System Memory
  - Used to store runtime program and code is executed directly from the system memory
  - This typically refers to
    - Internal SRAM/DRAM or NOR Flash
    - External memory that supports XIP (NOR Flash, SRAM, DRAM)
  - DRAM technically speaking does not support XIP by itself but processors that has a DRAM interface will have a DRAM controller that handle the necessary translation to allow execution of code directly from the DRAM.
- Storage memory
  - Used to store all program and data in a computer system
  - Cannot run code directly from storage memory, needs to be transferred to system memory before code execution
  - All memory types can be used as storage memory



- This is the last slide of the semiconductor memory chapter.
- I'll use this slide to recap the key points for System and Storage Memory.
- System memory
  - Store runtime program code and data that the processor use.
    - Note that the code is executed directly from system memory.
  - System memory could be internal or external memory.
    - For this course, there are three memory types that supports XIP and can be used to as a system memory, namely, SRAM, DRAM and NOR Flash.
    - DRAM, technically speaking does not support XIP as its access is also multi-steps, but typically processor that can be connected to a DRAM will have a DRAM controller module that handles the necessary translation to allow executing code directly from DRAM.
- Storage memory, on the other hand, stores all the program code/data in a computer system.
  - However, the processor do not run code directly from the storage memory, these needs to be transferred to the system memory for execution.
  - Technically speaking, all memory types can be used as storage memory.
  - If you need the storage memory to be able to retain information when the power is cut off, then non-volatile memory has to be used.
- That comes to the end of the semiconductor memory chapter.
  - We will touch on the HDD and SDD in next video.

# Cx1106
# Computer Organization and Architecture

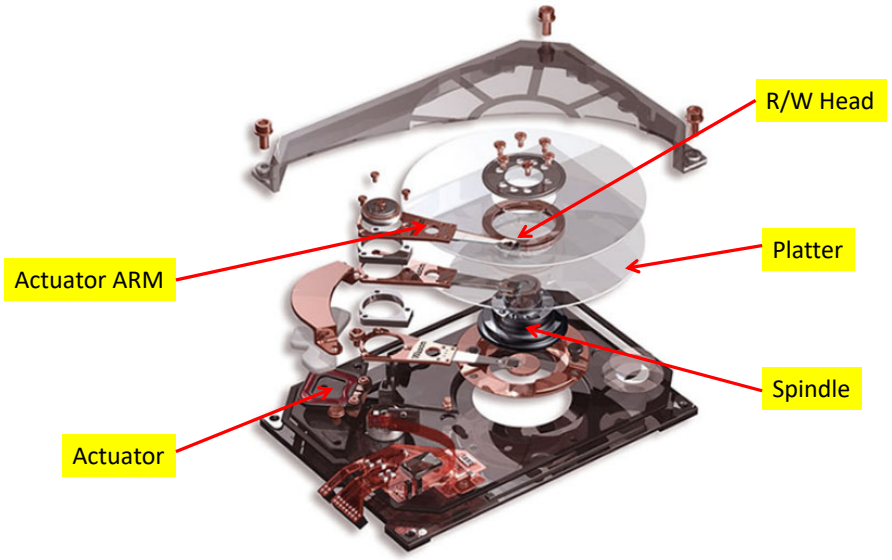
## HDD and SSD



Oh Hong Lye
Senior Lecturer
School of Computer Science and Engineering, Nanyang Technological University.
Email: hloh@ntu.edu.sg

- This chapter is on the two main storage devices for bulk storage, the Magnetic Hard disk drives (HDD) and the Solid State Drives (SSD).

# Magnetic Hard Disk

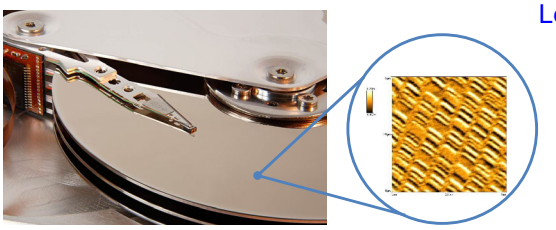



Source: Backblaze

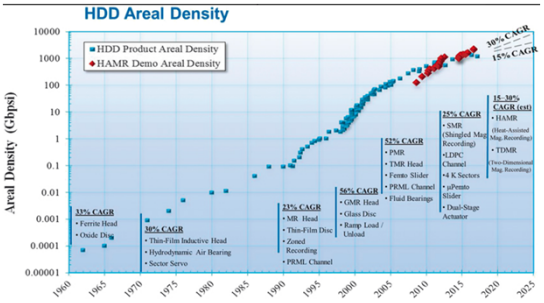


- This diagram shows an exploded view of the hard disk internal.
  - The Actuator ARM which carries the R/W head around different parts of the HDD media
  - The Actuator is an assembly of strong magnets and electrical windings. The resultant force produced when electric current flow through the wires is used to propel the actuator ARM.
  - The R/W head which picks up the changes in magnetic field on the media and converts these fluctuation into data bits.
  - The Platter which is the storage media. Each Platter consist of two surfaces, top and bottom surfaces. These day, data is stored on both surfaces.
  - Lastly, the spindle which is a motor used to spin the platter at very high speed of up to 15K rpm.

# Magnetic Hard Disk

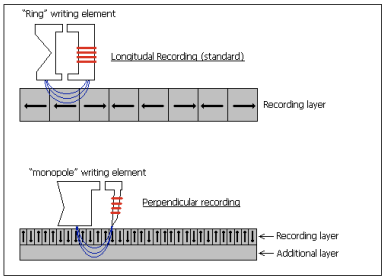


[Source] http://en.wikipedia.org/wiki/Disk_read-and-write_head




Source: Cambridge

Longitudinal vs Perpendicular Recording



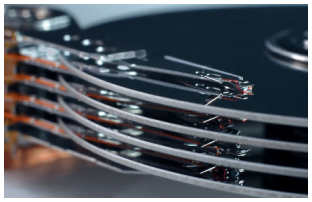

[Source] Robert Fontana et al, IBM Areal Density Comparison Paper, 2010

- Stores data by magnetizing a thin film of ferromagnetic media on the circular disk known as platter.
- Video on Introduction to Hard Disk. https://www.youtube.com/watch?v=kdmLvl1n82U

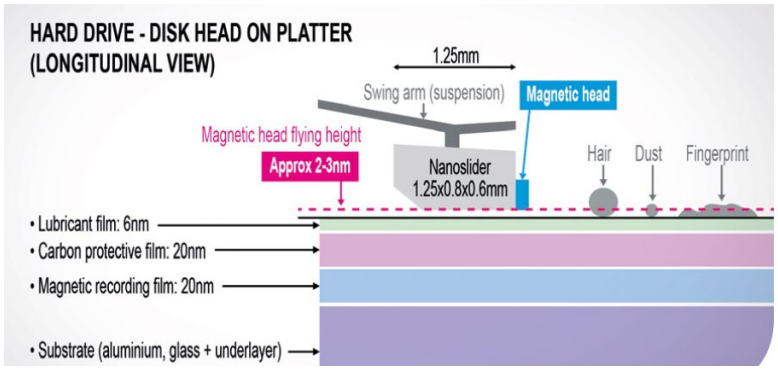


- Magnetic hdd is still the main mass storage for Data Centers, Servers, Desktop PC. However, it is fast being replaced by semiconductor memories such as SSD in the notebook products.
- Magnetic HDD stores data by changing the orientation of the tiny magnetic elements in the circular disc platter.
- There are Two types of recording method: Longitudinal and the more advanced perpendicular recording.
    - Difference between these two types of recording is in the orientation of the magnetic elements: as the name implies, longitudinal is along the surface of the platter while the other is perpendicular to the surface (which offers higher data storage density)
    - The different orientation of the magnetic element will be interpreted as ones and zeroes by the RW head.
- On the bottom left corner, we have a plot of the HDD areal density vs time. The graph shows how hard the HDD maker is working to enable a continuous improvement in HDD technology. Which is essential in enabling HDD to continue to be the memory element used in huge storage use cases such as the Data Centers.

# HDD Technology



- The magnetic head uses the wind pressure generated by the high speed rotation of the disk to fly above the disk surface, similar to the principles employed by the aircraft.
- Fly height is lower than a finger print mark.

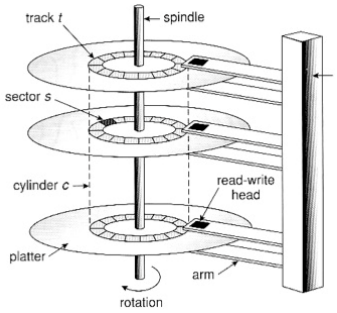



Source: datarecoverydublin



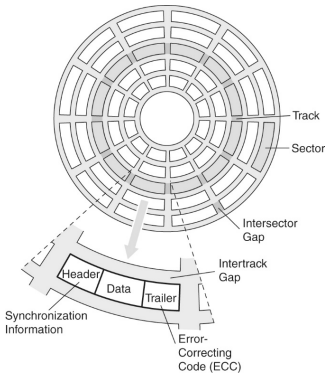
- Flying the Magnetic head is often compared to flying a jumbo jet.
- On the web, you'll find many different figures when people made this comparison. This is because it was done at different timeline, the technology they reference to is therefore different. And since HDD technology changed so quickly, you'll find that the number quoted differ by a large amount.
- There is one source quoting that flying the RW head is like flying the jumbo jet a few mm above the ground.
- With the actual height the RW head flies, even a finger print mark is taller than the fly height and will crash the RW head.
- Bottomline, it shows the huge amount of technology and precision that has gone into the tiny HDD.

# Magnetic HDD Data Organization




- Computers often use magnetic hard disks for large secondary storage devices.
  - One or more platters on a common spindle.
  - Platters are covered with thin magnetic film.
  - Platters rotate on spindle at constant rate.

- Data is stored on the surface of the platter in concentric rings called tracks.
  - Gaps between tracks to minimize interferences from adjacent tracks.
- Tracks are divided into sectors.
  - Minimum data block size is a sector.
- Stored data consists header, data and trailer.

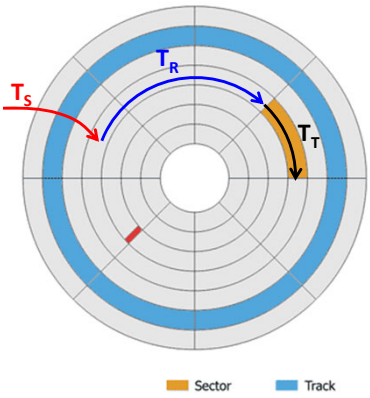





- Now let's go thru some definition of parts in detail.
- HDD stores its data in the circular disk known as platters.
- The reading and writing of info go thru the read/write head. There is one head to each surface of the platter.
  - RW head is attached to an actuator arm whose motion is initiated by electromagnetic forces.
- In terms of data organisation, the smallest container is known as a sector.
  - Multiple sectors will form a track, which is basically a circular track around the disc.
- Within each sector, the data is formatted to provide synchronisation info, actual data payload and error correction code.

# HDD Transfer Rate

- Seek time ($T_S$)
  - Time taken for the head to move to the correct track.
- Rotational Delay ($T_R$)
  - Time taken for the disk to rotate until the read/write head reaches the starting position of the target sector.
- Access Time ($T_A$)
  - Time from request to the time the head is in position ($T_S + T_R$)
- Transfer Time ($T_T$)
  - Time required to transfer the required data after the head is positioned.




Oh Hong Lye / Cx1106 6

- Now for some definition of timing parameters used in HDD performance measurement.
- First the Seek Time
  - This is the time it takes for the ARM to move from its resting position to the track that contains the target sector to be read.
    - The HDD vendor will typically supply a parameter known as the Track-to-Track seek time, since each movement from one track to another really takes different time, the parameter supplied by the HDD vendor is the average seek time to move from one track to another. That is, it is a statistical average value based on some test cases.
- Then we have the Rotational Delay
  - Which is the time it takes to move to the start of the targeted sector, after the RW head reaches the correct track.
- Adding the Seek Time and Rotational Delay will give you the Access Time.
- Lastly, we have the Transfer Time, which is the time it takes to read the target data after the RW head reached the start of the sector.

# HDD Transfer Rate

- $T_R$ is dependent on the rotational speed, Revolutions Per Minute (RPM), of the disk. For calculations, RPM is usually converted to Revolutions Per Second (RPS). i.e. RPS = RPM/60
- For a random section, average rotational delay $T_{R,AV}$ may be calculated as

$$T_{R,AV} = \frac{0.5}{RPS} \text{ seconds}$$

- $T_T$ is dependent on the rotational speed of the disk, the Track Density $D_T$ (number of sectors per track), Sector Density $D_S$ (number of bytes per sector) and the number of bytes N for the transfer.

$$T_T = \frac{N}{RPS * D_T * D_S}$$



- Rotational delay depends on the rotation speed. The faster the rotation speed, the shorter the rotational delay.
- Assuming we start from any random section, then the following formula will give the average rotational delay.
- So how did we derive this formula?
  - Since the RW head could land on any sector when it reaches the correct track,
    - Under the best case scenario, the RW head is just above the target sector, which means we have zero delay.
    - Under the worst case scenario, the RW head had just past the target sector, which means it need to wait for a full revolution of the disk before it can read the data. In this case the delay is equal to time it take to rotate one round, which is 1/RPS.
    - So if we take the average, it'll be 0.5/RPS, the formula given in the slide.
- For the formula on Data Transfer Time,
  - The assumption here is that the RW head will be able to read the data once it past over the corresponding sectors, so transfer time is equal to the time needed to transverse the targeted sectors.
  - Dt*Ds gives the total number of bytes per track and N is the number of bytes to read.
    - So N/(Dt*Ds) gives you the ratio of data needed vs total data in one track.

- Since the angular velocity of the disk is a constant, this ratio is also the ratio of the time it takes to transverse the targeted data vs the time it take to revolve one round, which is 1/RPS.
- That is how we end up with the expression shown in the slide for Tt.

# HDD Transfer Rate Example

- A magnetic hard disk rotates at 15000 RPM, with the following properties:
  - Average Seek Time, $T_S$ = 4ms
  - Track density, $D_T$ = 500 sectors per track
  - Sector density, $D_S$ = 512 bytes per sector
- Calculate the total time $T_{TOTAL}$ it takes to read a 3 KB file stored in consecutive sectors on the same track?

**[Solution]**

RPS = 15000/60 = 250 per second

Access time $T_A = T_S + T_R$ = 4 ms + (0.5/250) = 6 ms

Transfer time $T_T$ = 3072/(250*500*512) = 48 µs

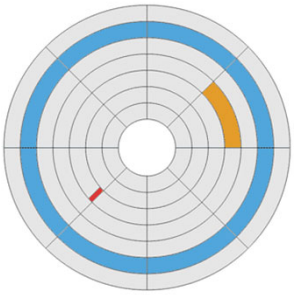
Total time, $T_{TOTAL} = T_A + T_T$ = 6.048 ms

Notice $T_A >> T_T$. This example shows that accessing file whose data is distributed across sectors on different tracks on the magnetic hard disk would potentially incur more time.

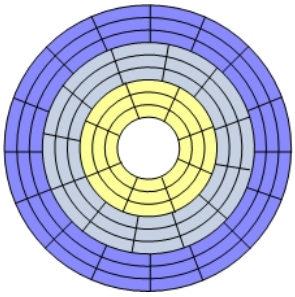


- Let's look at one work example to apply what we have learnt so far.
  - With a HDD having the parameters shown in the slide, what is the total time taken of read a 3KByte file stored in consecutive sectors?
- From the previous slide, we know that the total time taken is equal to Ta+Tt.
  - Where Ta = Ts +Tr, the seek time and the rotational delay
  - Tt is obtained using the expression in the previous slide.
  - With that, we have a final value of 6.048ms
- One point, notice that Ta>>Tt in this case.
- What this means is that if the data of one file is spread across different sectors in different track, then you'll find that the effective HDD transfer rate will be much lower compared to the case where the data are all within the same track. This is because every time the RW head completes reading one sector, it has to move to the start of the next sector which is located in a different track, and that is going to take another Ta duration.
- Not sure if you people still do disk defragmentation or not, but that is a process where you get the computer to organise the data so that data from the same file is allocate to consecutive sectors/track. Effect is that the HDD transfer rate should increase compared to before defragmentation.

# Physical Layout



- Early HDD
  - Physical Layout refers to the actual layout design on the HDD.
  - Equal number of sectors per track and each sector has the same data size.
  - Same-numbered Tracks from different surfaces formed a cylinder.
  - The early hard disks were implemented using this topology to simplify the controller design.
- Modern HDD
  - Having equal number of sectors per track means that sectors at the outer tracks are wider.
  - Waste physical space as bit density of those sectors are not optimal.
  - Zone bit recording technique addresses space wastage.
  - Tracks are divided into zones, with differing number of sectors per track for different zones.

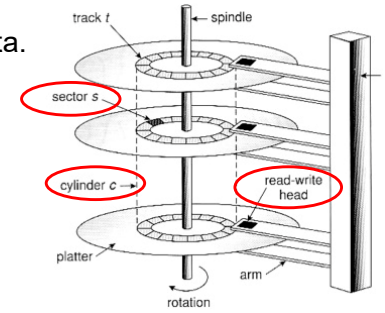


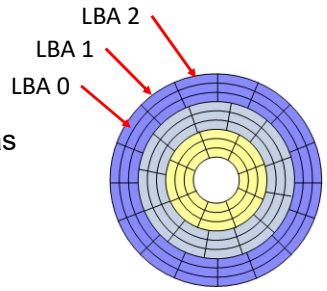


- Physical layout refers to how data is organized and formatted on the HDD media.
- In the past,
  - Every track has equal number of sectors and each sector contain the same amount of data.
  - This constant sector per track type of layout is similar to what you saw in earlier slides when we do the access time computation.
  - Early HDD design used this type of formatting for both logical and physical layout to simplify the controller design.
- But the physical layout of the HDD today has adopted a slightly different format.
  - Primary reason is that the old method of formatting results in a lot of space wastage since the sector length on the outer track are much larger than those in the inner tracks, meaning they could potentially store more data.
  - Hence HDD today uses the zone bit recording formatting. Where the sector length is kept approximately the same to fully utilize the recording media.
  - Outcome of this is that the number of sectors on the outer track is now more than those on the inner tracks.

# Logical Layout

- How the software see and address the HDD data.
- Address translations are needed to map the physical to logical locations. Two addressing scheme are CHS and LBA.
- Cylinder-Head-Sector (CHS)
  - Legacy scheme using the old HDD physical structure.
  - Made Obsolete in recent standards due to addressing limitation and more complex formatting.
- Logical Block Addressing (LBA)
  - Simple linear addressing starting from LBA=0 as first block, LBA=1 as second block and so on.
  - 48-bit LBA standard allows addressing up to 128PByte. 1PByte=2^50 Bytes.

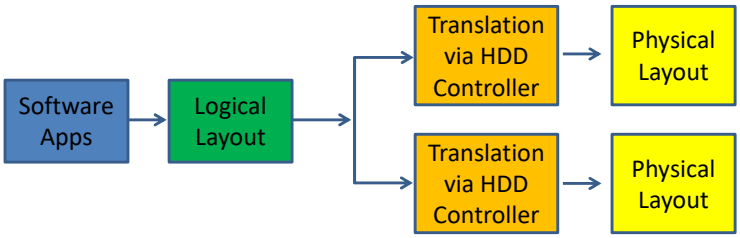



Oh Hong Lye / Cx1106 10

- Logical layout and the corresponding addressing, is what your program code used when accessing the HDD.
- There are two types of logical layout addressing used,
  - CHS, which stands for Cylinder-Head-Sector, and
  - LBA, short form for Logical Block Addressing.
- CHS is the legacy scheme that uses the old constant sector per track layout and formatting convention i.e. Cylinder, Head and Sector.
  - This scheme is more or less obsolete these days due to limit in the HDD capacity it can support.
- CHS is replaced by the LBA, which uses a simple linear addressing scheme.
  - As shown in the diagram, the first block in the HDD is given the address LBA0, next block = LBA1 and so on until the last block of the HDD.
  - 48 bits is allocated to specify the LBA number so this scheme have a lot of headroom, it can address up till 2^50 bytes.

## Logical vs Physical Layout




- Logical layout is needed in order to provide a common interface to all software developer
- If only physical layout is available, software has to be tuned for every HDD that uses a different physical layout
- Having a standard such as LBA allows the software developer to write application that can be used in any HDD.
- HDD vendor will provide the necessary firmware in their controller to perform the logical to physical layout address translation.



- So why do we need two different layout format?
- If only physical layout is available, a piece of software has to be tweaked everytime it needs to interface to a new HDD.
    - Because there is not standard way of designing a physical layout.
    - In fact, the physical layout of HDD from different vendors will be different, and is a way these vendors could differentiate their products.
- So in order to resolve this issue, we need a standard way for a software program to talk to the HDD, and that is where logical layout comes in.
- With a standard such as LBA, the software only needs to communicate according to this standard.
- Whenever a HDD vendor develop a new product, they will need to ensure that proper translation of logical to physical layout can be done via their HDD controller.

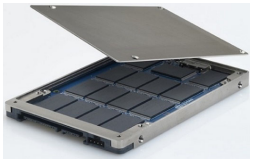
## Modern Day HDD

- Detail physical layout not given due to complex zone bit recording.
- Only the average transfer rate is given these days.
- Use of Cache and Buffers to speed up data transfer.
- However, concept and limitation of the magnetic HDD's physical design still valid
  - Actuator ARM is fixed and access has to be sequential, once data is missed, it has to wait for a disc revolution.
  - Seeking from track to track takes time as the R/W head needs to align to the starting sector.
  - More efficient to read in blocks rather than random access
  - Vulnerable to motion



- If you buy any HDD these days, you will realise that not much information of the HDD is provided beyond the capacity of the HDD, RPM valaue and the Average Data Transfer Rate.
- The detail physical layout information is not given because of the complexity of Zone Bit Recording, giving detail physical layout information will cause more confusion than useful information.
- The HDD these days employ use of Cache and Buffers to speed up the data transfer.
- However, the concepts and limitation we learn here is still applicable
  - The mechanics of the HDD and its associated limitation remains, e.g. the actuator ARM is fixed and access has to be sequential.
  - Seeking from track to track consume more time
    - So it is more efficient to read/write in blocks rather than random sectors
    - Which means HDD defragmentation is still applicable and helps to improve performance
  - And HDD, due to its mechanical nature, is vulnerable to motion.

# Solid State Drive (SSD)

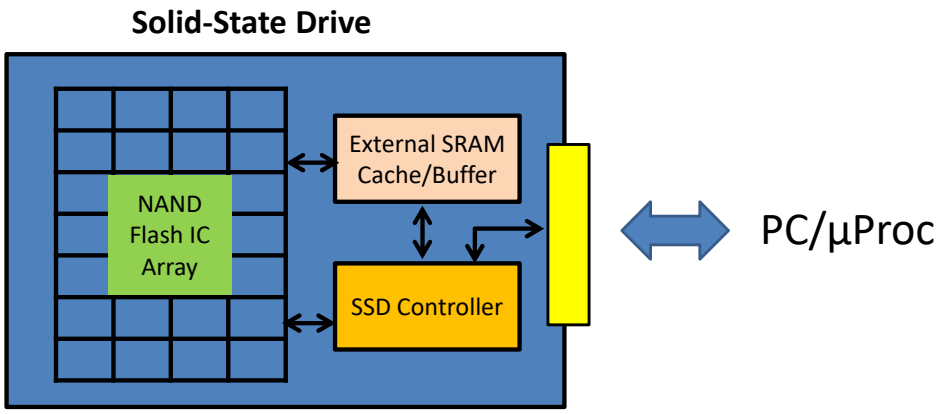


- Solid-state drives (SSD) are becoming popular
  - Memory array based on NAND-FLASH or NOR-FLASH.
  - Still more expensive than magnetic hard-disk.
- Flash memory has limited program-erase cycles (3,000 to 1,000,000).
- Various techniques used to extend the life of SSD
  - Wear levelling is a technique used to extend the life of the SSD disk by distributing data erase/write operations evenly over the entire disk.
  - Use External RAM as buffer to minimize the number of writes to Flash in SSD.
  - Error Correction Code. Ability to recover from one or more bits of error in media.
- MTBF (Mean Time To Failure) of recent SSD is comparable to that of HDD.

Oh Hong Lye / Cx1106 13

- Next we come to the solid state drives.
- These drive are getting more and more popular as they are more robust to handling, since they don't have any moving parts compared to magnetic HDD. And also, primarily because the cost of NAND flash memory, which is the main memory used in SSD, has dropped drastically over the last few years.
- It is, however still more expensive than magnetic HDD so many of the computers, especially the desktops and servers, are still using magnetic HDD. HDD is also still the main storage element found in Data Centers.
- One of the main issue with SSD is the limited program-erase cycles of flash memory that it uses.
  - What it means is that there is only limited number of times user can perform erase operation on a particular memory block location.
- But there are many techniques to extend the life of SSD
  - Wear levelling which spreads out the write/erase operation evenly over the entire disk
  - Using external RAM as buffer to reduce the number of memory access to Flash memory.
- With these enhancement and mitigation, the mean time to failure benchmark of SSD these days are comparable to magnetic HDD.

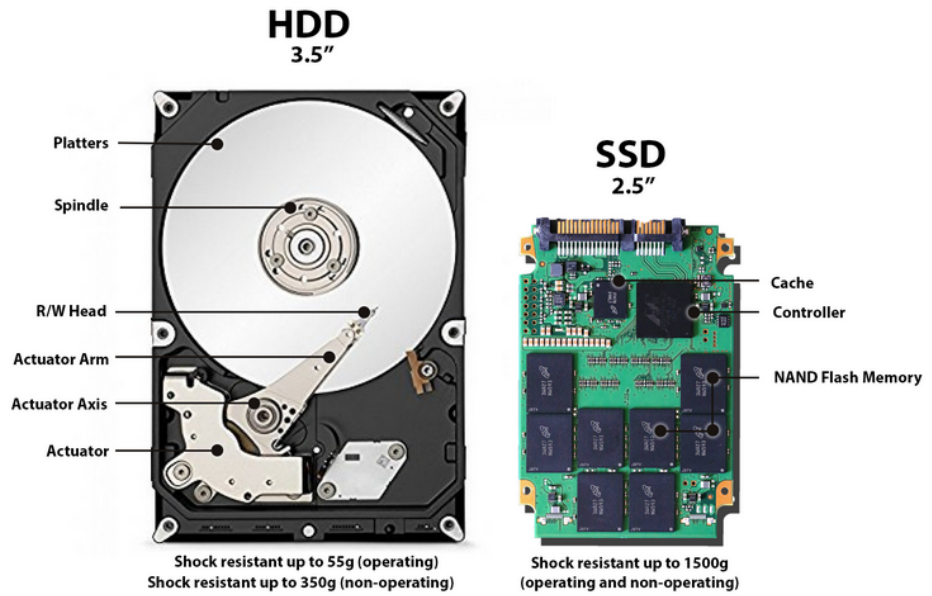
# SSD System Block Diagram






- This slides shows the system block diagram of a SSD.
- The main storage elements are NAND flash.
- There is a SSD controller which a basically a microprocessor that manages the operation of the SSD, this includes
  - Addressing translation between Logical to Physical Layout
  - Caching and buffering of data in and out of the SSD
  - Wear levelling algorithm
- There are external SRAM and Cache to improve the transfer rate for both read and write operation
- So in a way, SSD itself is a computer system.

# HDD vs SSD






- This is a photo comparison between a HDD and a SSD.
- SSD form factor can be made smaller actually, such as the mSATA and NVME form factor which is much smaller than the legacy 2.5inch HDD form factor.
- The earlier SDD adopted the 2.5 inch form factor so that they can be a drop in replacement for the 2.5inch HDD found on notebook computers.

# HDD vs SSD

- HDD
  - Pros
    - Lower Cost Per Bit,
    - Almost infinite Erasure cycles
  - Cons
    - Consist of Moving Mechanical Parts so more prone to crashing if HDD is dropped or shaken.
    - Heavier and larger physical profile.
    - Slower transfer rate
- SSD
  - Pros
    - No moving parts. More robust to movement.
    - Lighter and occupy less space
    - Higher Transfer rate
  - Cons
    - More Costly compared to HDD
    - Finite number of Erasure cycles

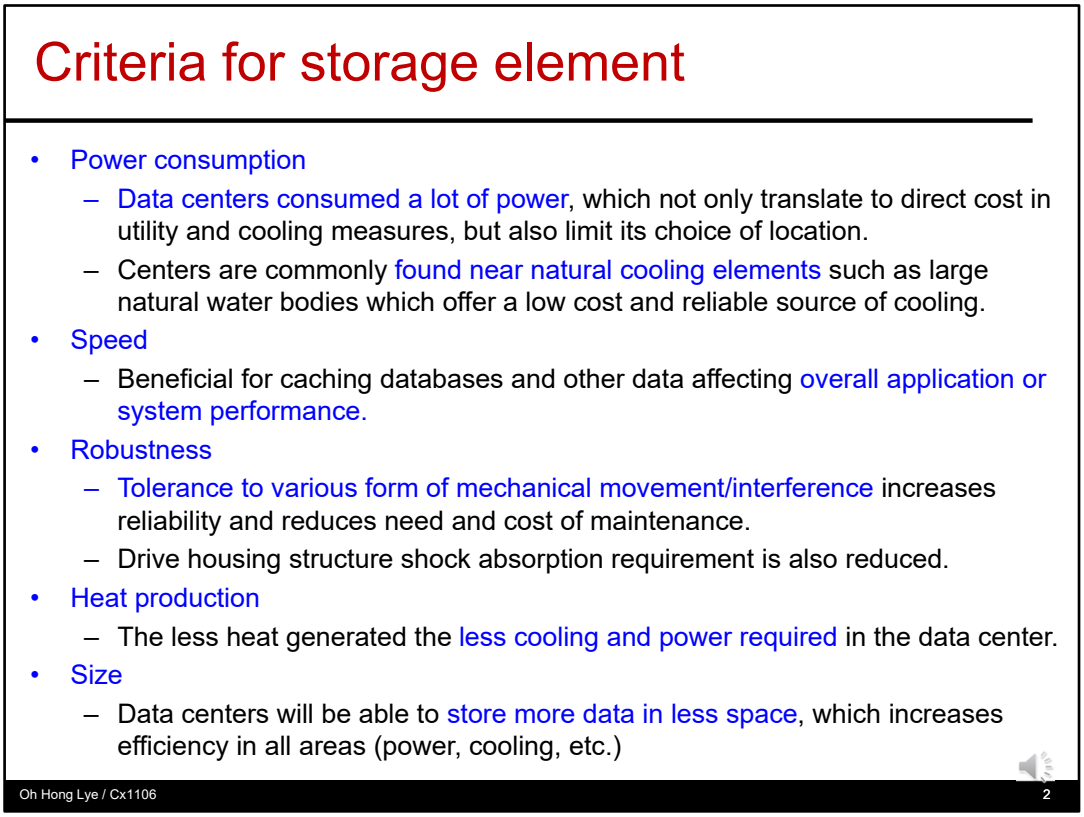


- Last slide of the HDD/SSD video.
- Advantage of the HDD
  - It has lower cost per bit and
  - Almost infinite erasure cycles.
- Disadvantage of HDD
  - There are many moving parts in HDD, it is therefore prone to crashing if there are huge movements
  - Its form factor are larger and more bulky
  - And has a slower transfer rate.
- Advantage of SSD
  - Opposite of HDD, there are no moving part so it is very robust against movement.
  - It is very light, can be made really small and
  - Has a higher transfer rate
- Disadvantages of SSD
  - Cost. Its more expensive cost per bit wise when compared with HDD
  - There is a limit to the number of erasure cycles user can apply on the SSD.
- We have come to the end of this video, next video will be on Data Center related information.

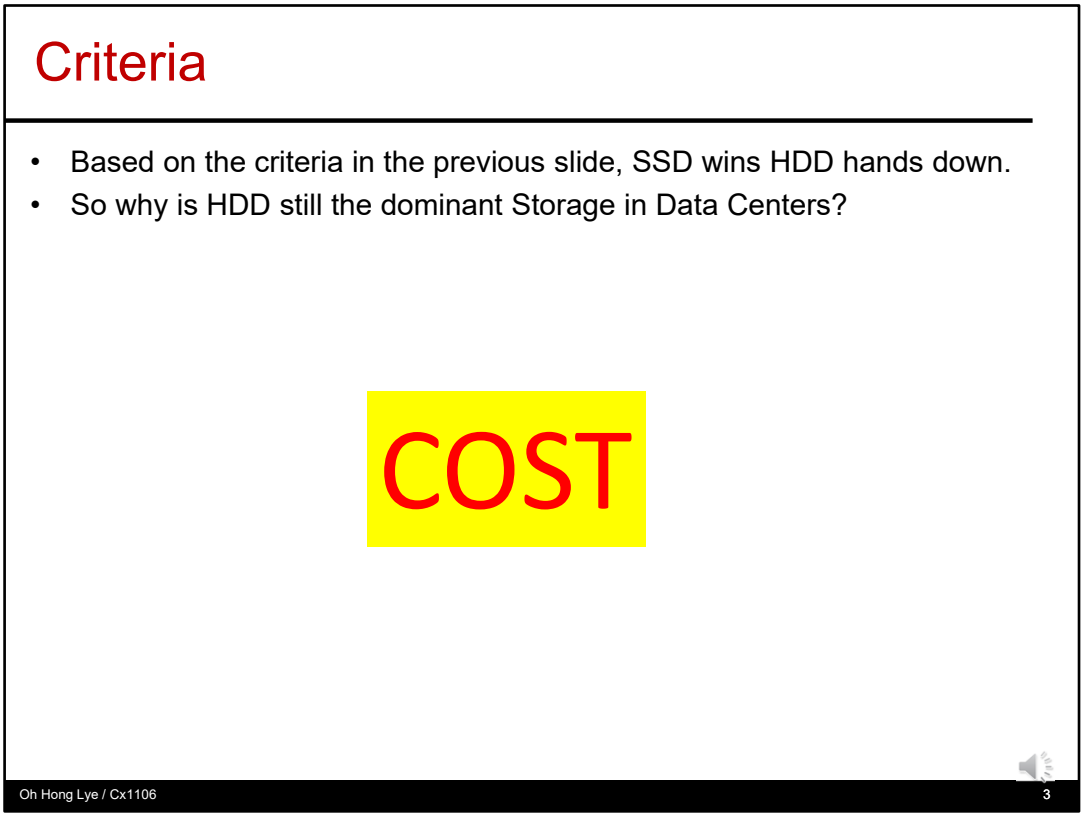
# DATA CENTER STORAGE



- In this section, we will touch on some HDD specific considerations under the context of Data Center Operation.

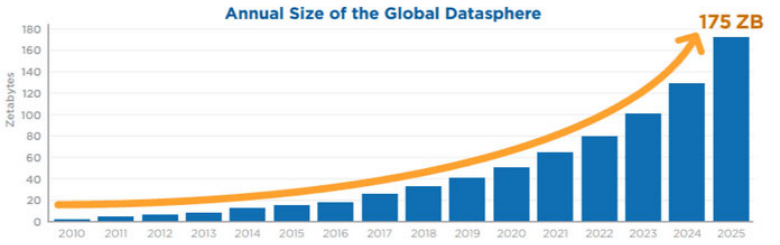



- This slide shows some of the key consideration when selecting storage elements to use in a Data Center.
- First is the Power Consumption
    - Data Center uses many storage elements so power consumption of these storage elements is very important.
    - Its translate to direct cost of electrical bill and also the indirect cost of requirement for cooling measures.
    - The need for cooling measures may also limit the choice of Data Center Locations, with many of the Centers located natural cooling sources such as natural water bodies.
- Next is the speed factor
    - Having storage element with higher speed will improve the overall performance of the system hosted in the Data Centers. Data Centers these days are increasingly being used to host virtual machines and other processing related functions, rather than just a simple cloud storage.
- The storage element needs to be robust as well too since reliability is one of the most important attribute of a Data Center.
    - One of the desired attributes are tolerance of various form of mechanical movement and interference. This will improve the reliability and reduces the need and cost for maintenance as the storage element is less likely to fail.
- Cooling measures, as mentioned above, incur cost and limits the location choices. So having low power consumption will be very beneficial.
- Lastly, would be good for the size to be small to be able to pack into a smaller building.

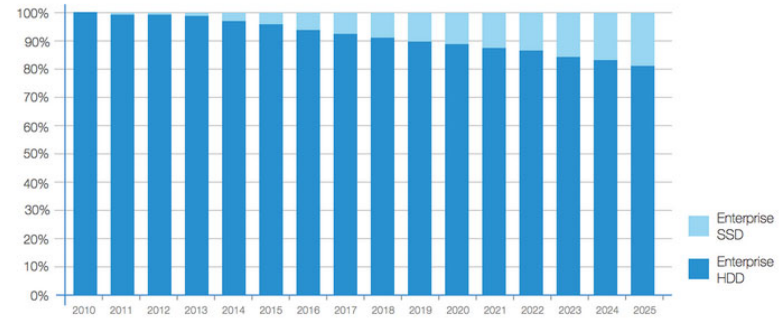



- Based on the selection criteria discussed in the previous slide, the obvious choice of storage element is the SSD.
    - But HDD is still the dominant storage for Data Center because of one most important reason – COST.
- The cost per bit of SSD today is still higher than that of HDD, this translate to huge cost difference due to the huge storage size of a Data Center.

## Global Storage Consumption and Shipment Projection




1 ZB = $10^{21}$ bytes

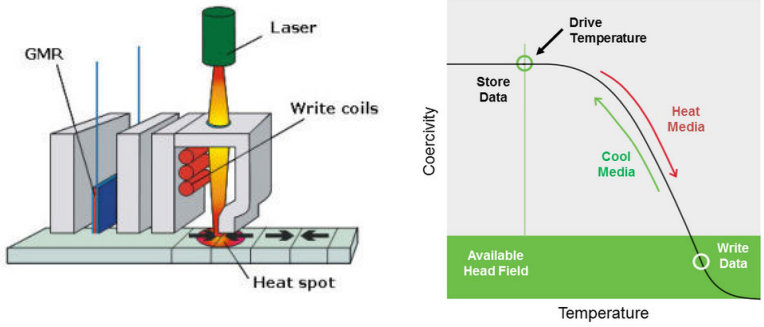



Source: IDC, Seagate.

Oh Hong Lye / Cx1106 4

- Other the Data Centers, other use case than requires huge storage also see HDD as the preferred choice, such as in the enterprise servers.
- As seen in the chart here, although SSD is slowing creeping into the market share of HDD, the total size of shipment for HDD is still much larger than that of SSD.

## HAMR (Heat Assisted Magnetic Recording)




- The areal density of the HDD was stagnant for a while and manufactures had been shipping drives with 2TByte/Platter.
- But with the HAMR, the areal density is expected to increase again.
- A small laser is attached to a recording head, designed to heat a tiny spot on the disk where the data will be written. This allows a smaller bit cell to be written as either a 0 or a 1.
- Current projections are that HAMR can achieve 5 Tbpsi, enabling hard drives with capacities higher than 100 TB.

Oh Hong Lye / Cx1106 5

- Cost is the single most important reason why HDD is still surviving to this day, so naturally, the HDD maker is always looking at new technologies to increase their cost advantage.
- One of these technologies is the HAMR, which stands for Heat Assisted Magnetic Recording.
    - There was a time where the area density of HDD was stagnant for a while and manufacturers had been shipping drives which has 2TB/platter.
- But with HAMR, the areal density is expected to increase again.
    - The recording media used in HAMR drive allows more data to be packed into the same area reliably, but it is also more difficult to change the orientation of the magnetic element on these media.
    - So for HAMR, a smaller laser is attached to the recording head.
    - The laser is designed to heat a tiny spot on the disk, increasing the temperature allows data to be written more easily to the disk.
- Current projection are that the HAMR drives can achieve 5Tbpsi, enabling drive with capacities higher than 100TB.

# HAMR

- The major problem with packing bits so closely together on conventional magnetic media is that the data bits become unstable and may flip (0 → 1 or 1 → 0).
- To make the media maintain their stability to store bits over a long period of time, the recording media needs to have a higher coercivity.
- Higher coercivity implies the media is magnetically more stable during storage, but it would also be more difficult to change the magnetic characteristics of the media when writing.
- For that, a laser is employed to heat a tiny region of several magnetic grains for a very short time (~1 ns) to a temperature high enough to lower the media's coercive field to below that of the write head's magnetic field. This is the write process.
- Immediately after the heat pulse, the region quickly cools down and the bit's magnetic orientation is frozen in place. The data is stored in the media and would be stable due to the high coercivity of the recording media.
- See the graph in the previous slide for the visual illustration.



- This slides describe the principle between the HAMR drives.
- One main problem of increasing the areal density of a HDD media is that the data bits will become unstable, meaning it has higher chance of flipping from 1 to 0 or vice versa.
- To make the media more stable, the recoding media needs to have high coercivity.
    - Higher coercivity means the magnetic media is more stable but it also means it is more difficult to change its magnetic characteristics, i.e. more difficult to write to the media.
    - To counter that, a laser is used to heat up the spot where the head is writing to, this increases the temperature and lower the coercivity of that surrounding area. The reduced coercivity allow data write to be carried out properly.
    - After the write operation, the laser is turned off, the area cools down quickly and regain its coercivity and stability.
    - You can revisit the graph in the previous slide for illustration of this process.