Mine
Date : / /20
Page :

Lec 12/13

Data Path

- Collection of fundamental units such as ALU or multipliers.
- Perform data processing operations, a set path is used, this is called Data path.

Definition :

- So, a data path is a collection of functional units such as ALU, registers and buses.
- Datapath is a set of functional units that carry out data processing operations.

The Control Unit + Data Path are the two major components of the CPU.

Control Unit

It generates control signals and components work according to these components.

- Control variable ⇒ Name of ~~a control~~ a control signal is know as control variable.
- Control Word ⇒ Collection of all the control signals is called control word.

Mine
Date : / /20
Page :

Control word is the signals generated by the CPU to do a single microoperation

Control Unit Organization

- Hardwired CU
- Microprogrammed CU

Hardwired C.U.

Control logic is implemented with Gates, flip-flops, decoders and other digital circuits.

- Can be optimized for faster operation
- Hard to rearrange ~~control~~ components.

A hardwired CU takes instruction as an input and a timing generator

The timing generator will generate * signals so that CU can identify which cycle it is in for a given instruction and will generate corresponding control word.

* Note: A single instruction can take multiple cycles / control words.

Mine
Date : / /20
Page :

Microprogrammed Control Unit

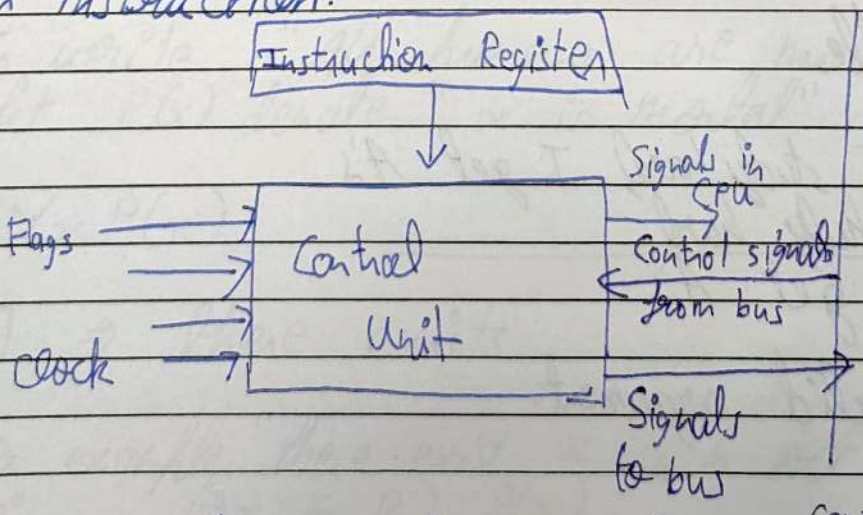
Rather than generating control word using hardware, keep all possible control words in some memory in form of micro-programs and send the corresponding control words when needed based on CU inputs.

- Updating logic is easy.
- Slower than hard wired CU

The ~~memory is sto~~ memory where control words are stored is called control memory. Control memory is in CPU.

All the control words for a instruction are stored in a sequence in control memory.

This makes it easier to generate words for an instruction.



A generic diagram for CU

Mine
Date : / /20
Page :

Lec 14

Control Word Sequencing

On each location in control memory :

Control Word	Next address information

is called Micro-instructions.
The collection of micro-instructions is called micro-program.

The control unit will send control words in sequence until the next address information tells the sequencer that instruction has ended in which case external input is taken.
This following of sequence is called control word sequencing.

The sequencer uses MUX to decide what to use as next address is, so a micro instruction uses the following structure.

control signals (control word)	MUX Select	Next Address

** Collection of control signals is called control word

The mux select will tell the MUX in sequences what to forward.

Vertical Control Unit

The CU that store control word as it is are called horizontal CU.

In a vertical CU, the bits are grouped such that we know that at a time there is only one bit high and others are zero.

We can group also in a way that almost one is zero

Eg, if a bus is connected to a group of registers we know only one has ~~IN~~input signal 1 and rest have zero.

We use decoders to store these grouped signals.

~~We use n-bits to re~~
We can compress an n-bits signal group to ceil($\log_2(n)$) bits.

And latter later pass these compressed signals through a appropriate decoder.

Mine
Date : / /20
Page :

If all signals in group can be 0.
we use ceil(log2(n) +1) bits in compressed format to account for that possibility.

The extra outputs of decoder are discarded in this case.

** So basically,
a n-bit control group will have n combinations if one & only one bit is 1
and will have n+1 combitations if atmost 1 bit can be 1.

** And no. of bits in compressed signal are
~~ceil(log2(no. of signal control words in group~~
ceil(log2(no. of combinations))

Mine
Date : / /20
Page :

Lec-15

In Vertical CU if the ~~program~~ few signals cannot be ~~grouped~~ together, they are stored in a horizontal manner.

In terms of speed, from fastest to slowest

Hard wired CU, horizontal CU, vertical CU

~~Nano Programmed~~

Nano Programmed Control Unit

In a micro-programed CU we store the microinstructions in a sequence for easier sequencing.

But trying to maintain a sequence may ~~be~~ leads to a lot of repetition of ~~micro inst~~ control signals.

And control signals are long so keeping them in sequence ~~may be~~ takes a lot of memory.

To reduce memory usage we use ~~nano~~ nano programmed CU.

All unique control signals are kept in a seperate memory and in our micro instruction, we replace the

signal with the addres to that signal in our unique signals memory.

The main benefit is usage of less storage.

* This is slower than microprogrammed CU.

RISC (Reduced Instruction Set Computer)	CISC (Complex Instruction set computer)
1) Less number of instructions	More no of instructions
2) Fixed length instructions	Variable lenght instructions
3) Simple instructions	Complex instructions
4) Limited addressing modes	Mae & Complex Addressing modes
5) Easy to implement hard wired	Hard to implement hardwired
6) One cycle per instruction	Multiple cycles per instructions
7) More registers	Less registers. Any ALU input possible.
8) Register to Register ALU only	

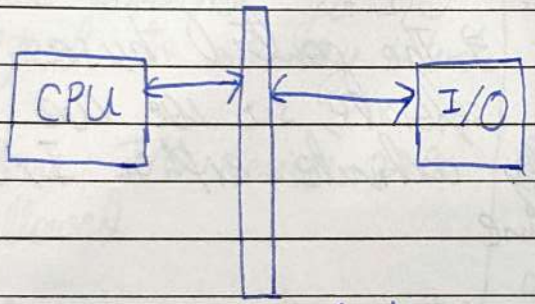
Mine
Date : / /20
Page :

Lec - 16

I/O Organization

Peripherals :
Other than main memory, all devices connected externally are known as peripheral devices.

Peripherals
- Input device
- Output device
- Storage devices



I/O interface
(Hardware / ~~soft~~ software)
↓
Device Drivers

I/O interface will help in ~~data~~ signal conversion. It also helps to synchronize the ~~data~~ signals sent between CPU and I/O.

I/O interface also help in data format conversion between peripherals & CPU. I/O interface also manages peripherals such

they don't interrupt CPU or other peripherals.

I O vs Memory Buses

- Seperate I/O and memory buses
  - common address & data bus but different control buses
  - Common buses for both I/O & memory.

We don't use completely seperate buses because it is not cost effective.

Common data/address → The control buses are seperate, so we use control bus to only activate either I/O or memory at a time.

~~Hence~~ We have seperate address for I/O and memory.
This config is called I/O mapped I/O or port mapped I/O.

Common data, address & control buses → In this config we take address from memory and assign that to an I/O device.

~~Note when control bus~~

Mine
Date : / /20
Page :

Now when address bus refers to a address related to I/O, the I/O device is accessed, otherwise memory is accessed

This config is called Memory mapped I/O

Memory Mapped I/O	I/O Mapped I/O
i) I/O do not have own address space	i) I/O have their own address space
ii) Some memory waste	ii) No memory waste
iii) Memory access instruction can be used to access I/O also	iii) I/O and memory have different instructions.
iv) More instruction for I/O (equal to memory instruction)	iv) Less instructions for I/O
v) More I/O devices are allowed	vi) Only a limited number of I/O devices are allowed (depends on no. of ports).

Mine
Date : / /20
Page :

Lec 17

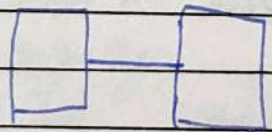
Asynchronous Data Transfer

If two devices work on same clock, they are in synchronous operation.
But in synchronous device clock has to be set on basis of the slowest component and hence causing bottleneck.

⇒ So we use asynchronous system where ~~each~~ component has its own separate clock.

So in async system, we need external synch.
The I/O interface will provide sync.

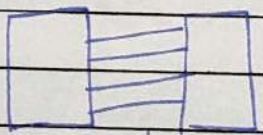
Serial data transfer



1-bit at a time

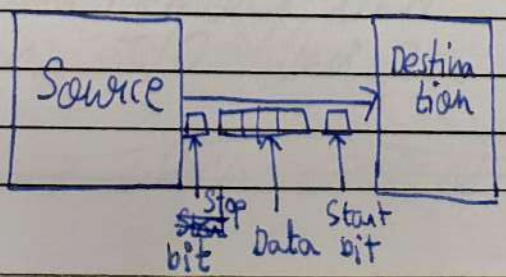
✗ cheaper

Parallel data Transfer



n-bit at a time
where, n = no. of lines
✗ Faster

Serial ~~data~~ asynchronous



To identify the start and end of data transfer, we use start bit and stop bit

* First a start bit is sent,
* Then the data is sent based on the memory word size (byte addressable / word addressable)
* Then stop bit is sent.

The start & stop bits are use to keep source & destination in sync.

Note: the memory is typically byte addressable so 8 bits of data are sent at a time.
Consider memory to be byte addressable unless specified.

Mode of transfer

- Programmed I/O
- Interrupt Initiated I/O
- Direct Memory Access (DMA)

Programmed I/O (Polling)

- There is no provision through which I/O can inform CPU of data transfer.
- I/O sets status and waits
- CPU runs program periodically to check

Mine
Date : / /20
Page:

status of each device
- If any device has status set, the CPU performs data transfer on it.

To set the status a flag register is used in I/O interface

Interrupt Initiated I/O

- I/O has provision (interrupt signal) to inform CPU about data transfer
- when CPU recieves interrupt,
  - ✱ It completes current instruction
  - ✱ Saves status (PC, PSW etc) of current process on stack
  - ✱ Branches to service interrupt
  - ✱ Resumes process by taking value from stack.

Mine
Date : / /20
Page :

Lec 18

Interrupt Service Routine

The program which is run, in response to a interrupt is called ISR.

Different interrupts will have different ISR.

Vectored Interrupt

The device which sends interrupt signal will also send a vector address which is the address to the proper ISR to that interrupt.

Non-vectored Interrupt (Scales interrupt)

The device only send interrupt signal.
The CPU will use Default Service Routine.
The default service routine will tell CPU the ISR address.

ISR's are a part of the device drivers.

Maskable interrupt

CPU can accept or reject the interrupt. CPU can also keep it in pending (based on CPU design)

Nonmaskable Interrupt

CPU cannot reject these interrupts.

Mine
Date : / /20
Page:

Internal interrupts

When interrupt occurs due to unexpected error during instruction execution.

Eg. page fault, divison by zero.

External interrupts

Interrupts recieved by external reasons. (ie not by the currently processing instruction error) by I/O devices mostly

** Internal / External interrupt are also called software / hardware interrupts. respectively

Simultaneous Interrupts

Interrupt from higer priority device will be serviced first.

- Software solution
- Hardware solution

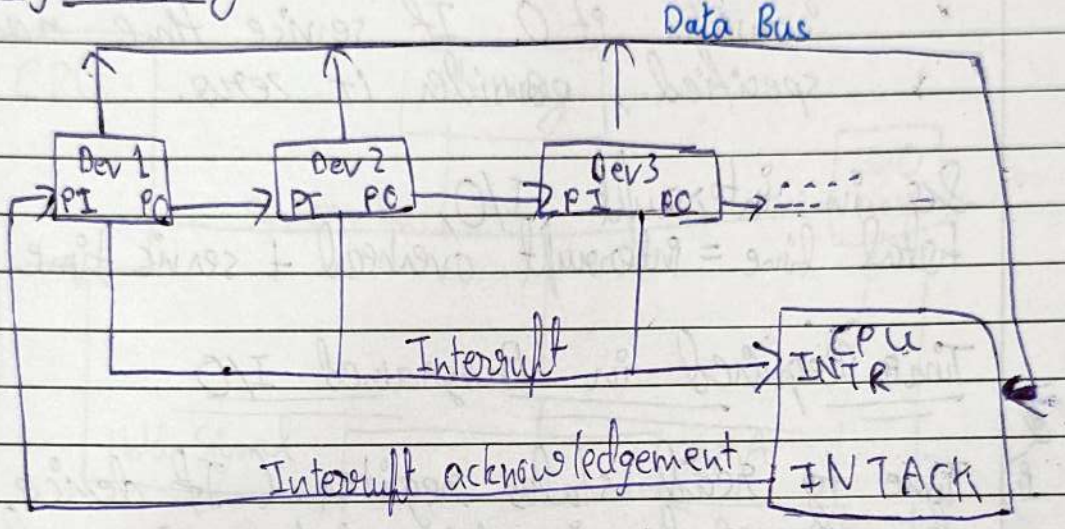
Software solution : Upon recieving a simultaneous interrupt, a software is executed to decide device with higher priority. (It uses polling)

Mine
Date : / /20
Page :

Hardware Solution

- Serial (Daisy chaining)
- Parallel (Not in syllabus 😊)

Daisy chaining



- If interrupt is on from a device, signal is recieved in INTR
- CPU generates achknowledgement and send it to serial chain
- In priority Dev 1 > Dev 2 > Dev 3 ...
- If the dev was causing interrupt recieves ~~signal~~ achknowledgement, it can retun its vector address
- Else device forwards acknowledgement.
- The INTR in CPU will remain enabled until all interrupts are resolved.

Usually, devices which is nearest to CPU is given highest priority.

Mine
Date : / /20
Page:

Time Required in Interrupt I/O

i) Interrupt overhead (completing current inst, storing in stack etc)
ii) Service time (remaining all).

Note: if overhead not specified in ques, consider it 0. If service time not specified, consider it zero.

Eg in interrupt I/O,
total time = interrupt overhead + servic time

Time Required in Programed I/O

i) Time to read status register. (If device transfer rate is given, take size of register in consideration. If size not specified consider what size-addressable machine is using, else 1 byte)
ii) Transfer time between components

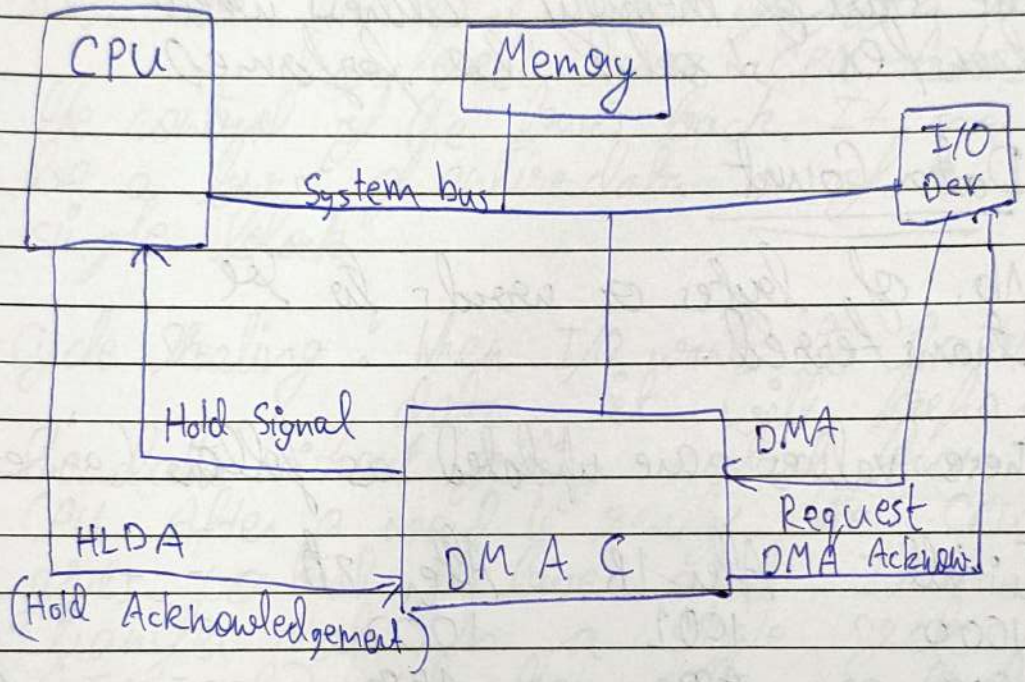
Transfer size

Mine
Date : / /20
Page :

Lec-19

DMA I/O

A hardware named DMA Controller is used to transfer data without CPU.



CPU sends two data to DMAC before starting transfer,

8) Starting address
20) Data ~~conte~~ count

The complete process is

1) I/O device sends DMA Request to DMAC
2) DMAC sends Hold signal to CPU
3) CPU sends data to DMAC using system bus
4) CPU sends HLDA signal to DMAC
5) ~~DMC~~ A DMAC send DMA Acknowledgement to I/O.

Mine
Date : / /20
Page :

6) Data transfer between device and memory using system bus.

I Starting Address

The start of memory address where transfer should be performed

II Data Count

No. of bytes or words to be transferred.

These values are updated as follows (an example)

	Initial	After 1B	After 2B	---	
S.A.	1000	1001	1002	---	
D.C	500	499	498	---	0

During DMA Transfer, CPU can perform only those operation which do not require system bus.
So mostly CPU is blocked

If CPU is blok for too long, performance will suffer.
So we use DMA Transfer modes to solve this issue

Mine
Date : / /20
Page :

Modes of DMA Transfer

- Burst Mode
- Cycle Stealing
- Interleaving DMA

Burst Mode : The entire burst of data is transfered before CPU takes the control of the buses back. It can be a burst of entire data or of a single block.

Cycle Stealing : When I/O wants to transfer data, it will prepare the data without getting control from CPU. After a word is ready, the CPU gives control of buses and I/O will transfer data in a single cycle (I/O cycle) and then return control.

If time required to prepare data = t_x
" " transfer data = t_y

% of time CPU blocked = $\frac{t_y}{t_x + t_y} \times 100$ (Burst mode)

If transfer time is overlapped with preparation time

% of time CPU is blocked = $\frac{t_y}{t_x} \times 100$ (Cycle stealing)

Mine
Date : / /20
Page:

Lec-20

Interleaving DMA

Only when CPU is not using system bus, DMAC will use the system bus.

CPU will not be blocked due to DMA.

* DMAC is a special purpose processor in this, as it can control data transfer between memory & I/O (can generate address & control signals for memory).
(DMAC is often refered to as special purpose processor).

* DMA service can be done independent of CPU ~~instruction~~ state, so unlike Interrupt service, it can be done in the middle of an instruction.

⇒ * If data size is very small, we use interrupt I/O

⇒ * If data size is very large, we use DMA.

DMAC has 3 registers

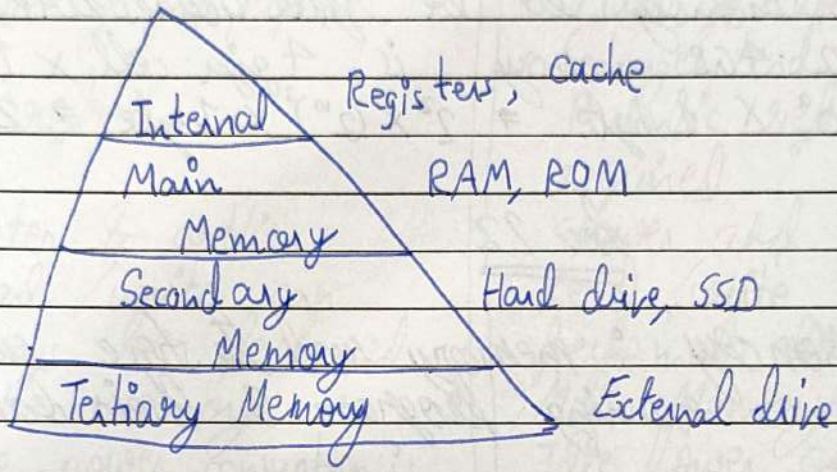
* Address Register - Address to destination in memory
* Word Count Register - The no. of words to be transferred
* Control Register - Specifies the transfer mode (Interleaving, cycle stealing, Burst)

Mine
Date : / /20
Page :

Lec - 21

Memory ⇒ Physical device, used for storage.

Goal of Memory hierarchy
1. Reduce access time
2. Reduce cost per bit

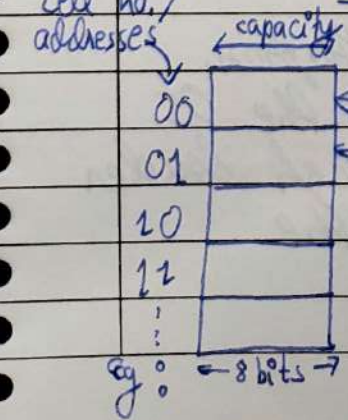


The goal is to have system access memory more on the higher level (top level).

Average cost of memory = $\frac{\text{Cost of total memory in system}}{\text{Size of total memory in system}}$

Memory Representation

Memory is represented by
= No. of cells × 1 cell capacity



Mine
Date : / /20
Page :

⇒ So a 8X16 bits memory has 8 cells each of size 16 bits.

⇒ Each cell has a unique number we call memory address

⇒ 128 X 8 bit can be written as 128 X 1 byte or 128 bytes. If not specified, we can consider memory to be byte addressable.

⇒ So a 4GB memory is 4 giga cell X 1 byte ⇒ $2^2 \times 10^9 \times 1$ byte ⇒ $2^2 \times (2^{10})^3 \times 1$ byte ⇒ $2^{32} \times 1$ byte

Lec 22

Main Memory : memory used to store current running programs & their data.

Types → ROM
→ RAM

ROM ⇒ It stores initial ~~function~~ instructions when CPU is started. The first three functions of these instructions is

i) Hardware check (Power On Self Test)(POST)
ii) Booting (Bootstrap Program performs booting)

⇒ We can also ~~store~~ store Bootstrap Program (BSP) in secondary memory with the OS and store a ~~Bootstrap~~ Bootstrap Loader in our ROM which will run the BSP.

Mine
Date : / /20
Page :

RAM ⇒ used to store current running programs.

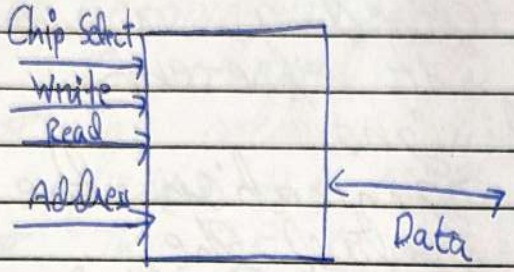
↳ Static RAM
↳ Dynamic RAM

	Static RAM	Dynamic RAM
i)	Made of flip flops	Made of capacitors (charges tend to discharge)
ii)	No refresh	Periodic refresh is required
iii)	Faster & costlier	slower and cheaper
iv)	Read/write can always be performed	Read/write cannot be performed during refresh
v)	Cache memory	Main Memory
vi)	Idle power consuption is less compared to DRAM	Idle power consuption is more compared to SRAM
vii)	Operational power is more compared to DRAM	Operational power is less compared to SRAM

Mine
Date : / /20
Page :

Lec 23

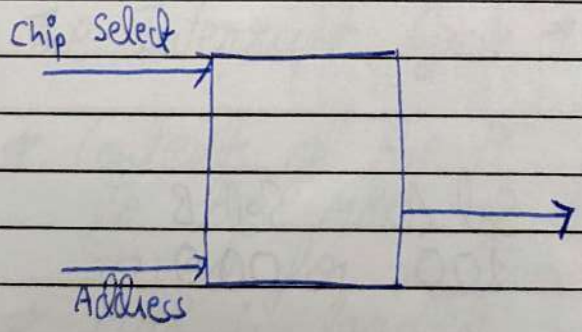
RAM Chip



Chip Select ⇒ Use by CPU to select either RAM chip or ROM chip.

CS	RD	WR	Operation
0	X	X	no op.
1	0	0	no op.
1	0	1	Write
1	1	X	Read

ROM Chip



CS	Operation
0	no op.
1	Read

Mine
Date : / /20
Page :

Multiple Chip Support

Assume we have two chips in RAM

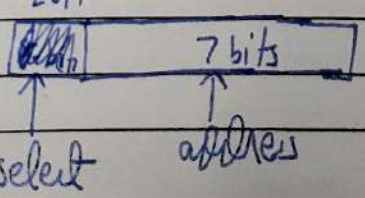
RAM 0	⇒	128 X 8 bits	address 7 bits
RAM 1	⇒	128 X 8 bits	address 7 bits
Total	⇒	256 X 8 bits	address 8 bits

So we need 7+7 bits for both, but CPU only generates address for the total memory. Hence only 8 bits.
So we use the MSB of 8 bit address to split them to two 7 bit signals.

They are divided as follows

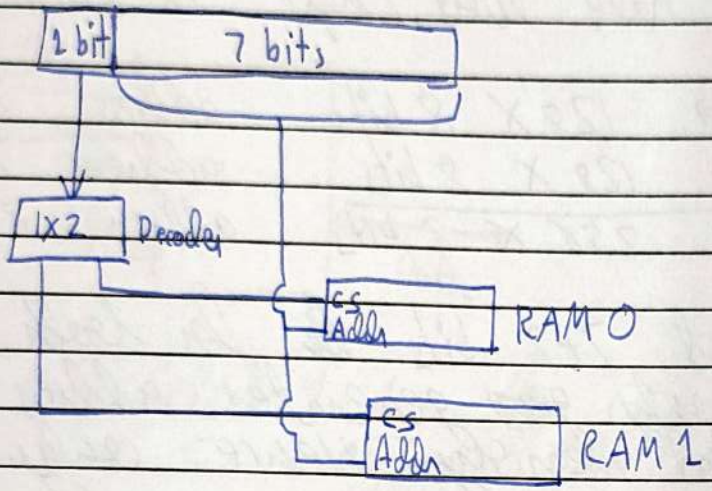
Decimal	Binary
0	0 0 0 0 0 0 0 0
⋮	⋮
127	0 1 1 1 1 1 1 1
128	1 0 0 0 0 0 0 0
⋮	⋮
255	1 1 1 1 1 1 1 1

So for CPU generated address in our example



Mine
Date : / /20
Page:

This can be drawn in hardware as CPU address



Similarly we can expand this idea for more chips.

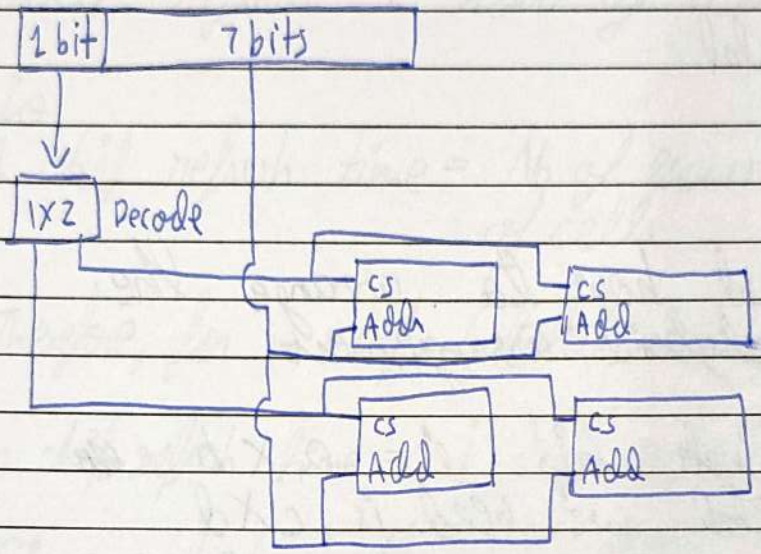
⇒ So when we keep the amount of addressable data same and increase no. of cells, it is called vertical arrangement.

⇒ We could also rather keep number of cells constant and increase addressable size, this is called horizontal arrangement

⇒ Using both together is called horizontal arrangement.

Mine
Date : / /20
Page :

Eg, ~~256~~ 256 x 16 bits using 128 x 8 bits



Mine
Date : / /20
Page :

Lec 24

Multiple chip Support

- Horizontal
- Vertical
- Hybrid

If we want how to arrange the chips in hybrid arrangement

If a single chip is $a \times b$ and total we need is $c \times d$

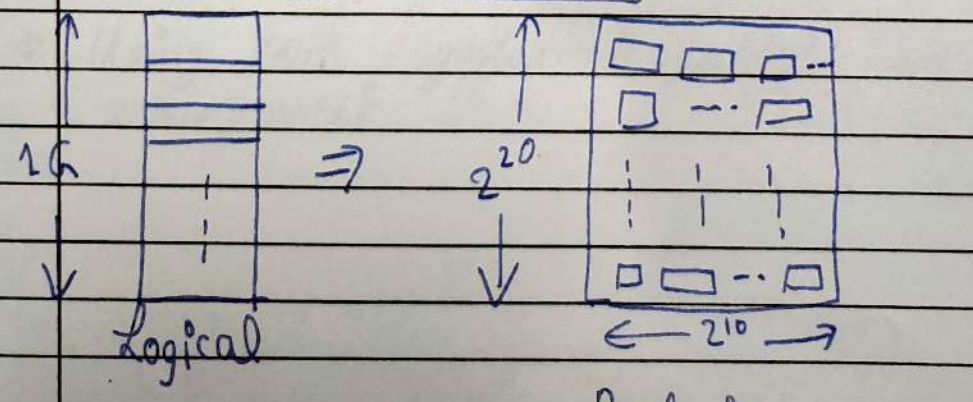
The arrangenet is

$$\left(\frac{c}{a}\right) \times \left(\frac{d}{b}\right)$$

Vertical size

Horizontal size

DRAM Chip Cell Arrangement



Mine
Date : / /20
Page :

DRAM Chip Refresh

In a DRAM chip, one refresh operation will refresh 1 row of cells.

So
1 chip refresh time = No of rows of cells × 1 refresh operation

Therefore, for n chips (total memory)

n chip refresh time = 1 chip refresh time

Since, refresh happens parallely.

Note: We are referring to rows as in a row in the practical design of a chip as shown in previous page.