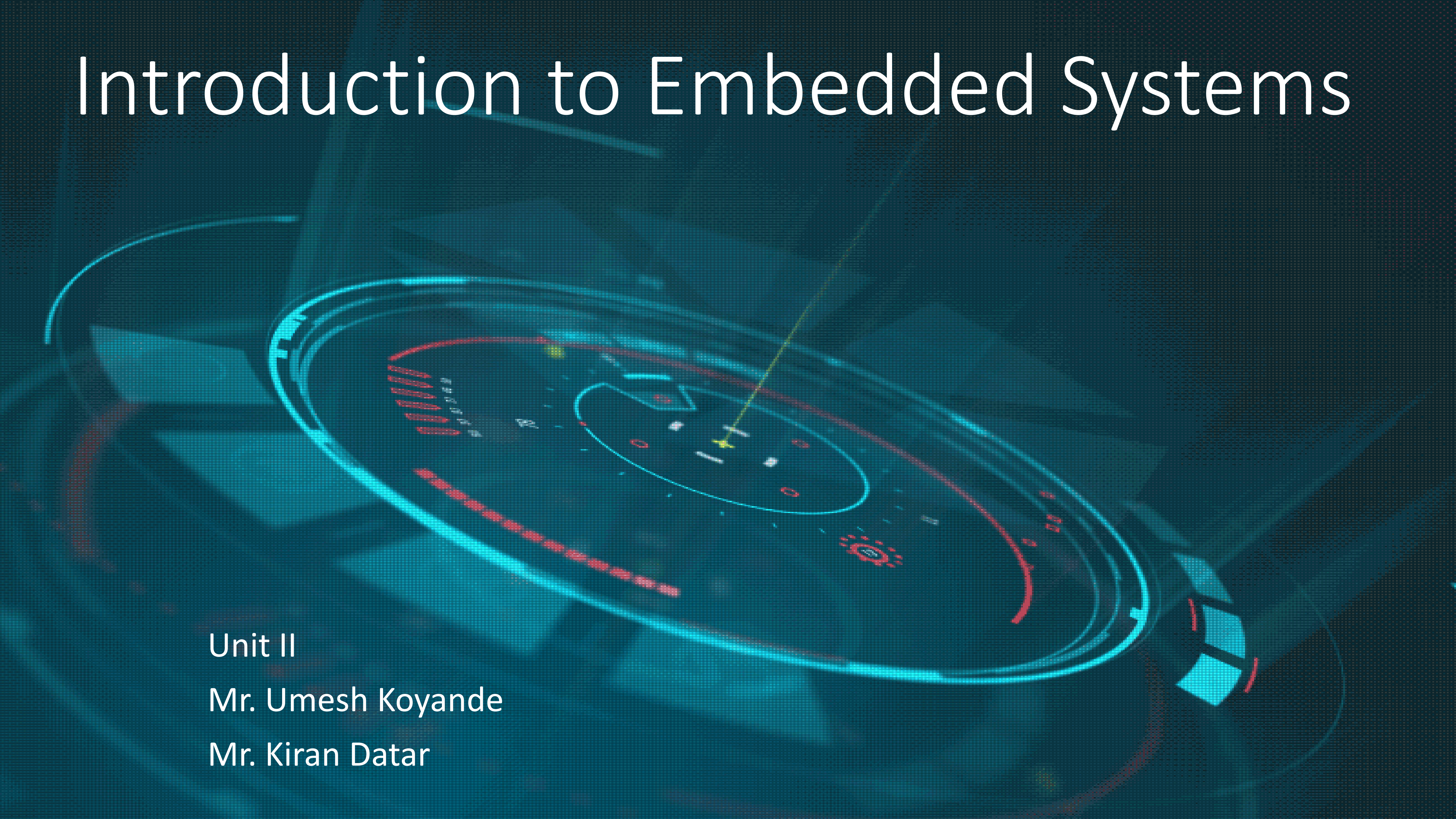
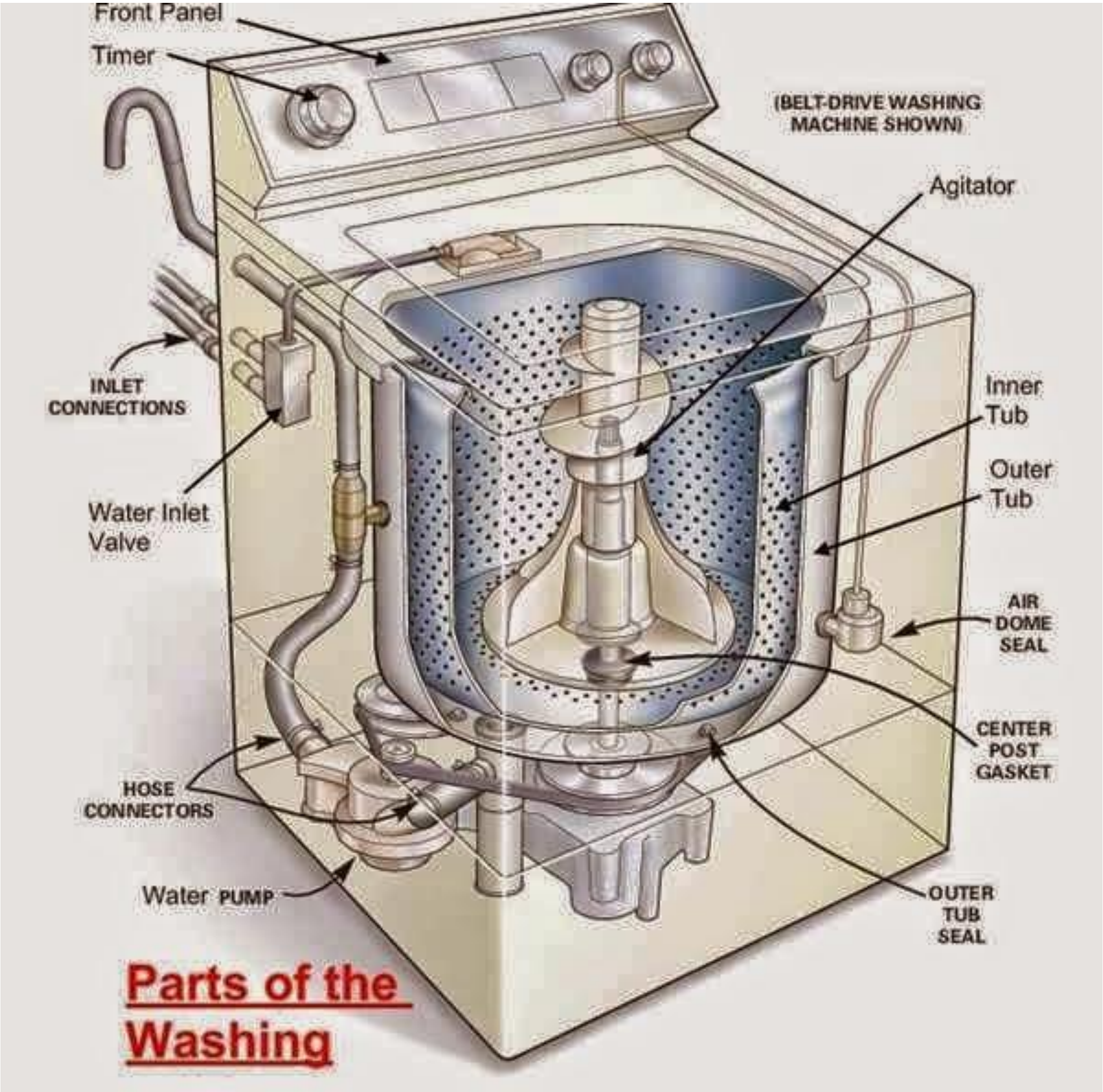
Introduction to Embedded Systems



Unit II

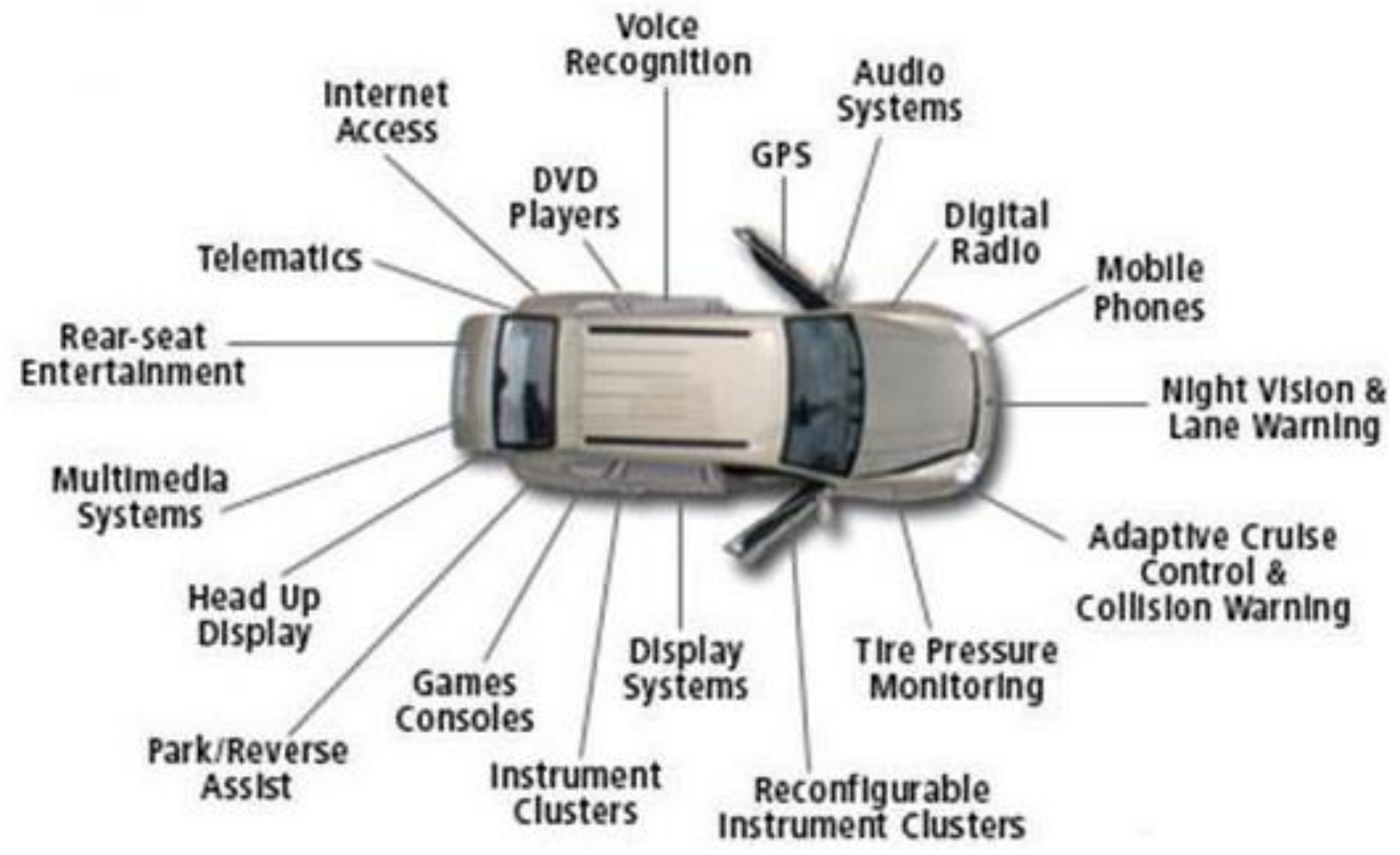
Mr. Umesh Koyande

Mr. Kiran Datar



Application-Specific Embedded Systems

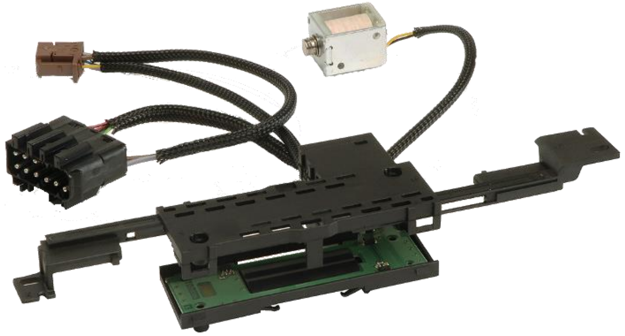
- https://www.youtube.com/watch?v=_EUxskkQjV8&ab_channel=eSpares



Domain Specific Embedded Systems

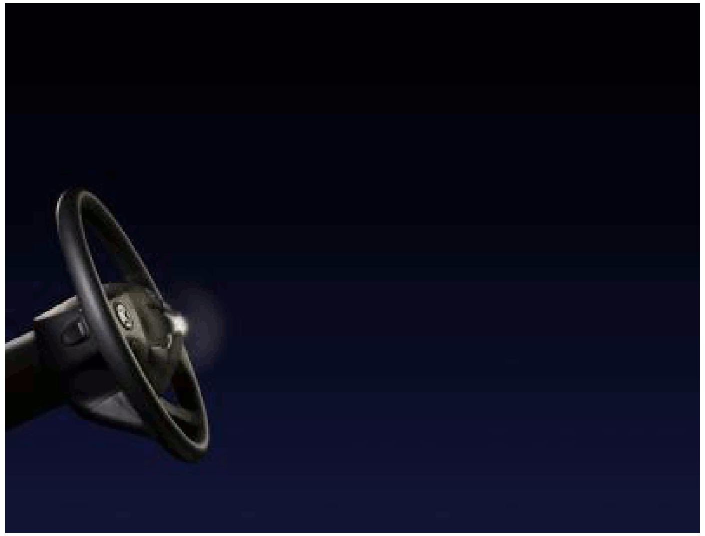
https://www.youtube.com/watch?v=oUf9_rS1fYg&ab_channel=tberth

Application And Domain Specific



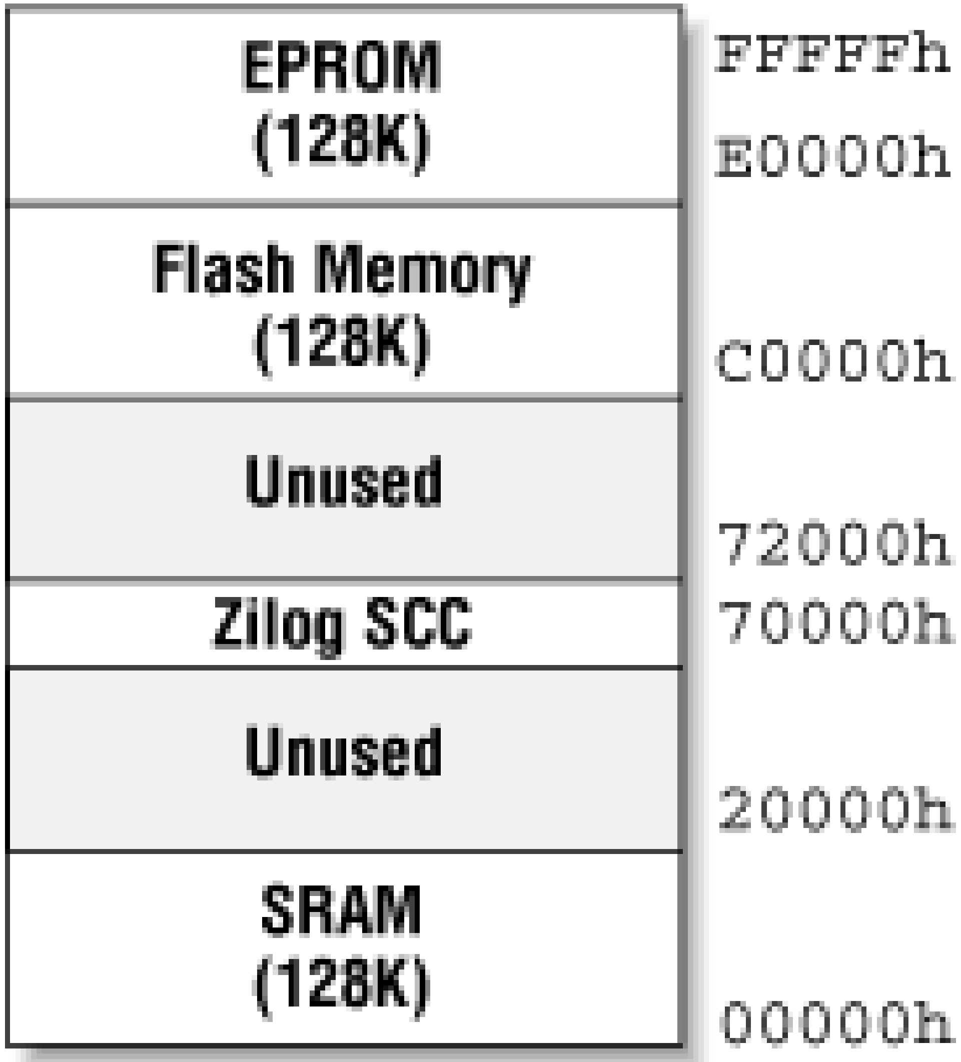
Electronic Control Units

- High–speed Electronic Control Units (HECUs)
- Low–speed Electronic Control Units (LECUs)



Memory map

- All processors store their programs and data in memory.
- Some cases memory resides on the very same chip, but more often it is located in external memory chips.
- Processor communicates with memory with the address bus and the data bus.
- To read or write a particular location in memory, the processor first writes the desired address onto the address bus.
- Data is then transferred over the data bus.



EPROM (128K)

Flash Memory (128K)

Unused

Zilog SCC

Unused

SRAM (128K)

FFFFFh

E0000h

C0000h

72000h

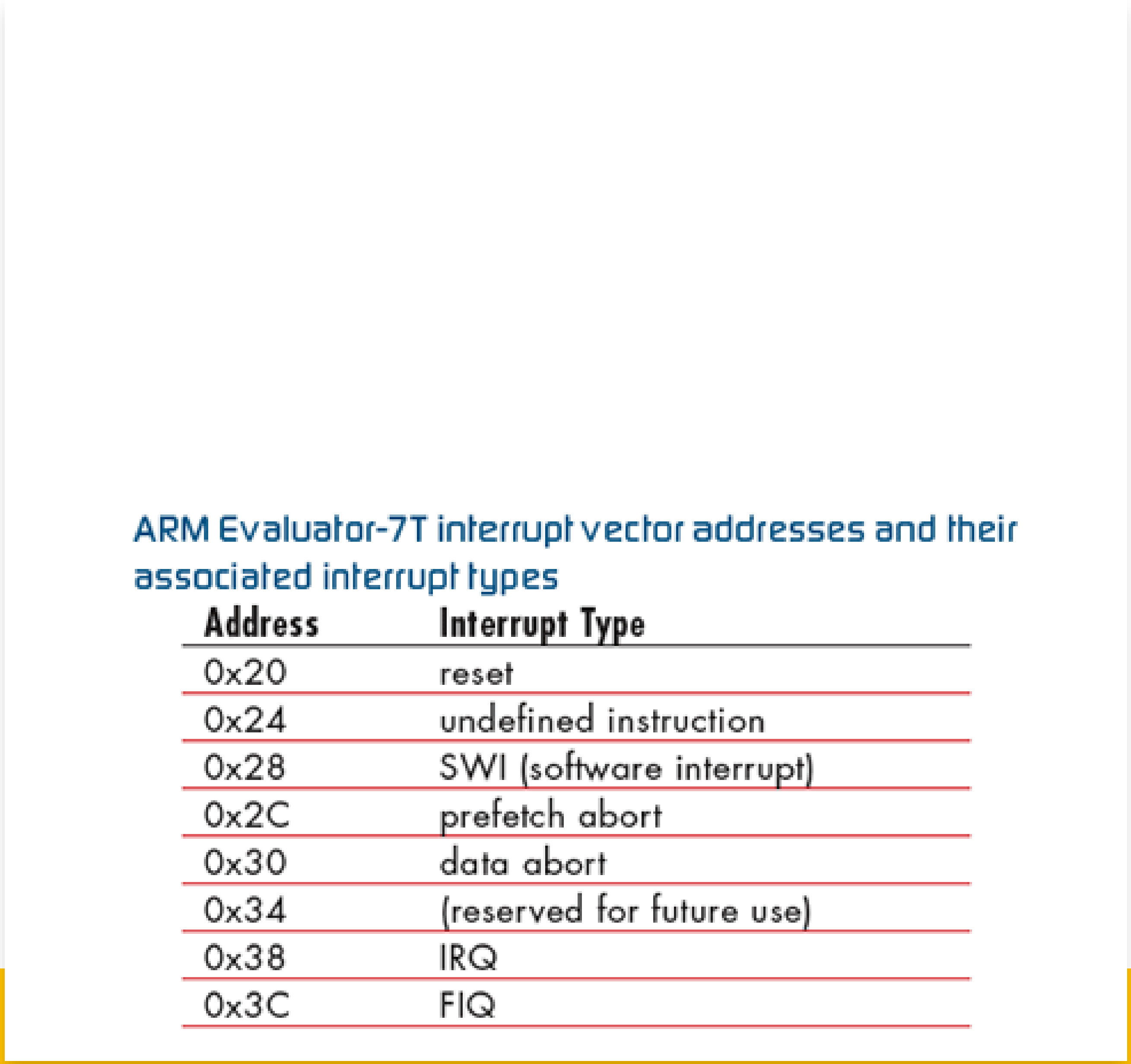
70000h

20000h

00000h

Interrupt map

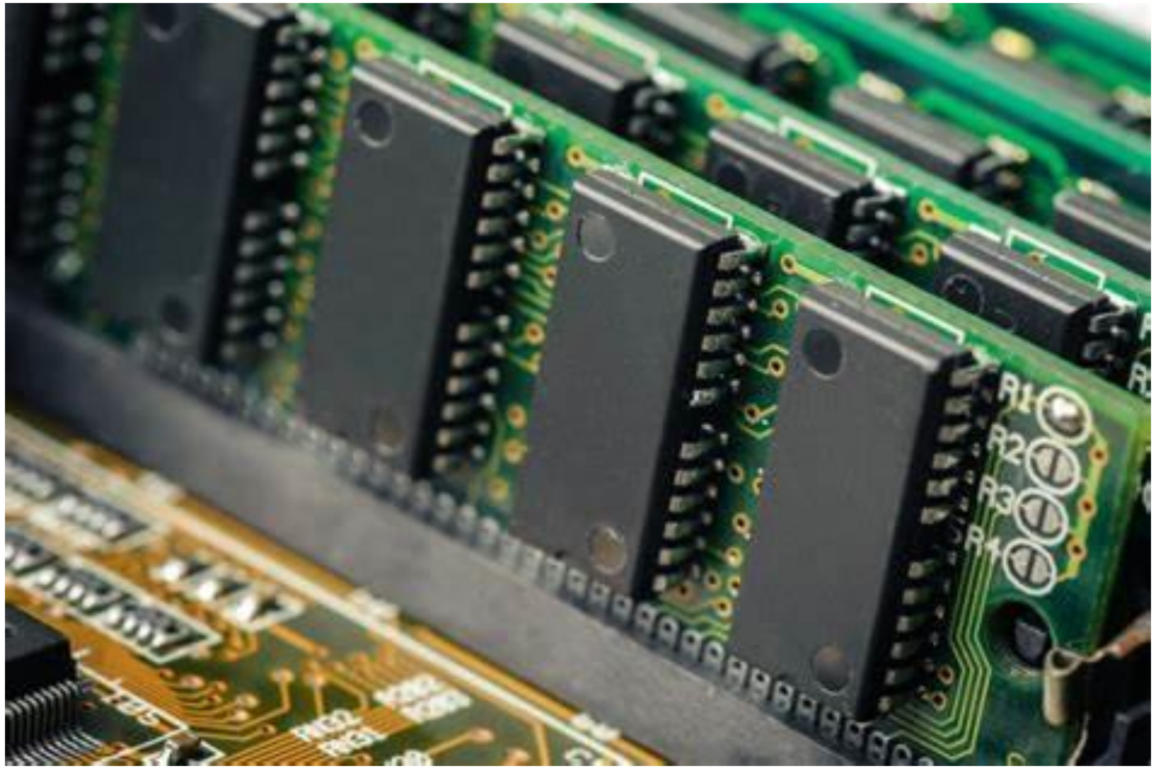
- Interrupts and associated with each of these are an interrupt pin and an ISR.
- To execute the correct ISR, a mapping must exist between interrupt pins and ISRs.
- Interrupt vector table.
- An interrupt map is a table that contains a list of interrupt types and the devices to which they refer.

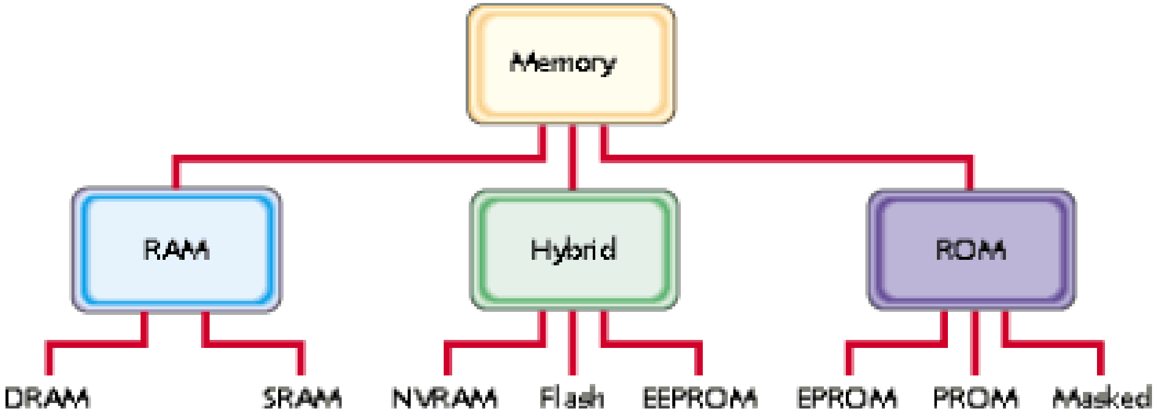


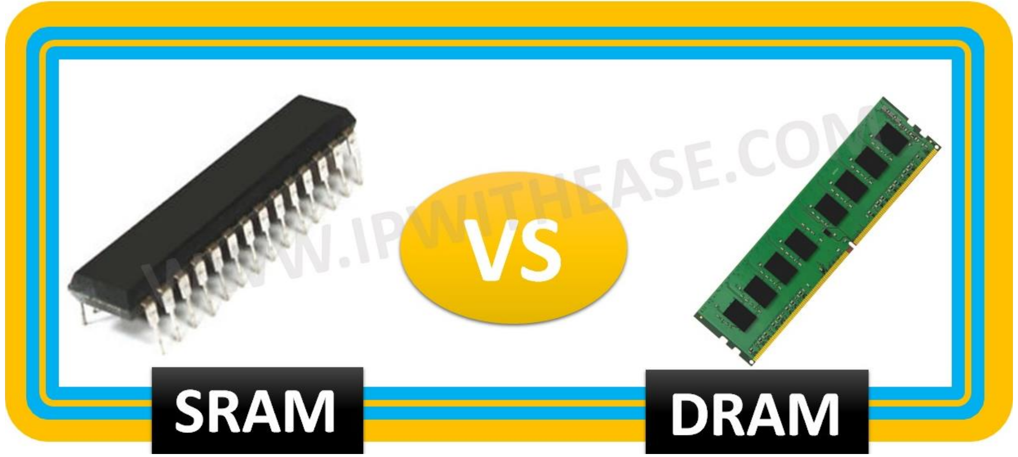
ARM Evaluator-7T interrupt vector addresses and their associated interrupt types

Address	Interrupt Type
0x20	reset
0x24	undefined instruction
0x28	SWI (software interrupt)
0x2C	prefetch abort
0x30	data abort
0x34	(reserved for future use)
0x38	IRQ
0x3C	FIQ

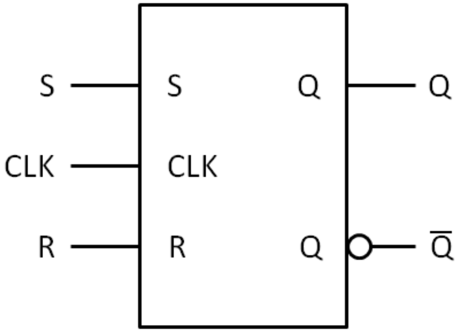
Types of Memory

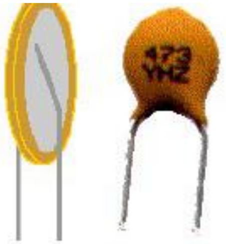




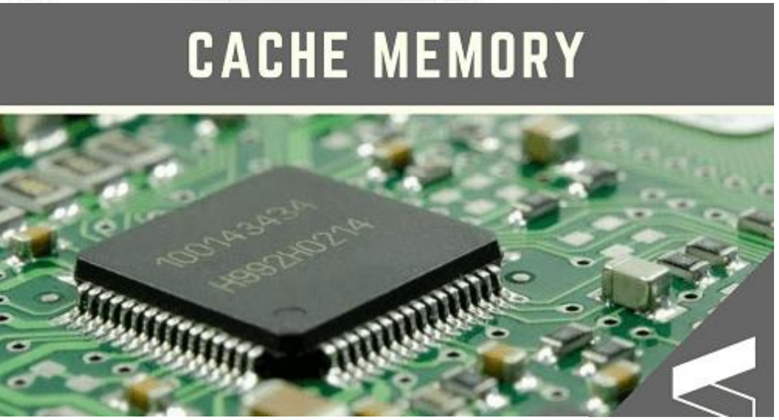

WWW.IPWITHEASE.COM
VS
SRAM
DRAM

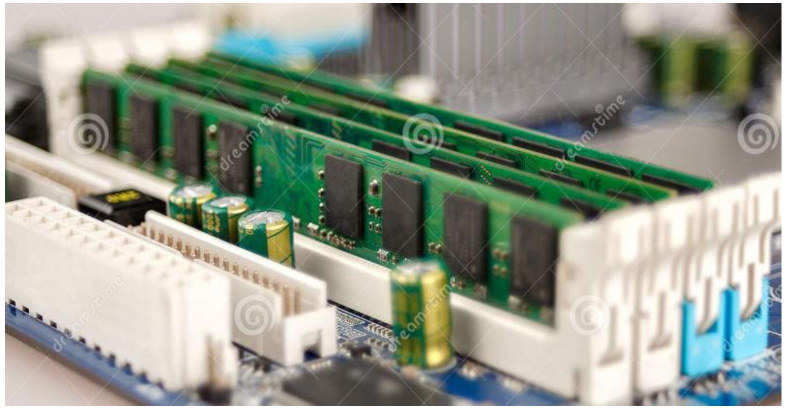
Expensive


S
CLK
R
S
CLK
R
Q
Q
Q
Q̄


473
YHZ

Cheap


CACHE MEMORY
100143434
H9S2H0214



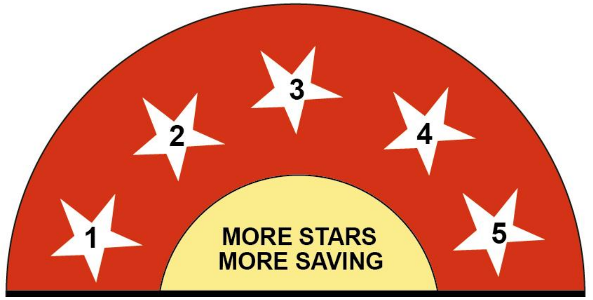
SD HC
Lock
10
U1
32GB
Digi-chip
Speed-Pro

SD XC
Lock
10
U1
64GB
Digi-chip
Speed-Pro

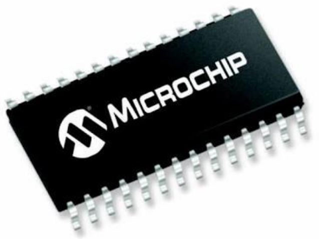
The more
stars the more
energy efficient

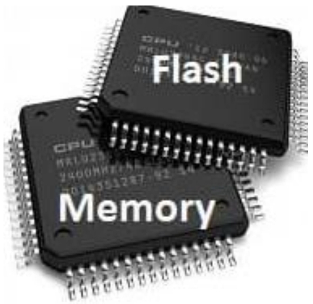
REFRESH

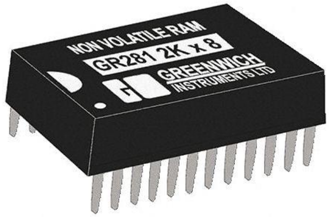
REFRESH

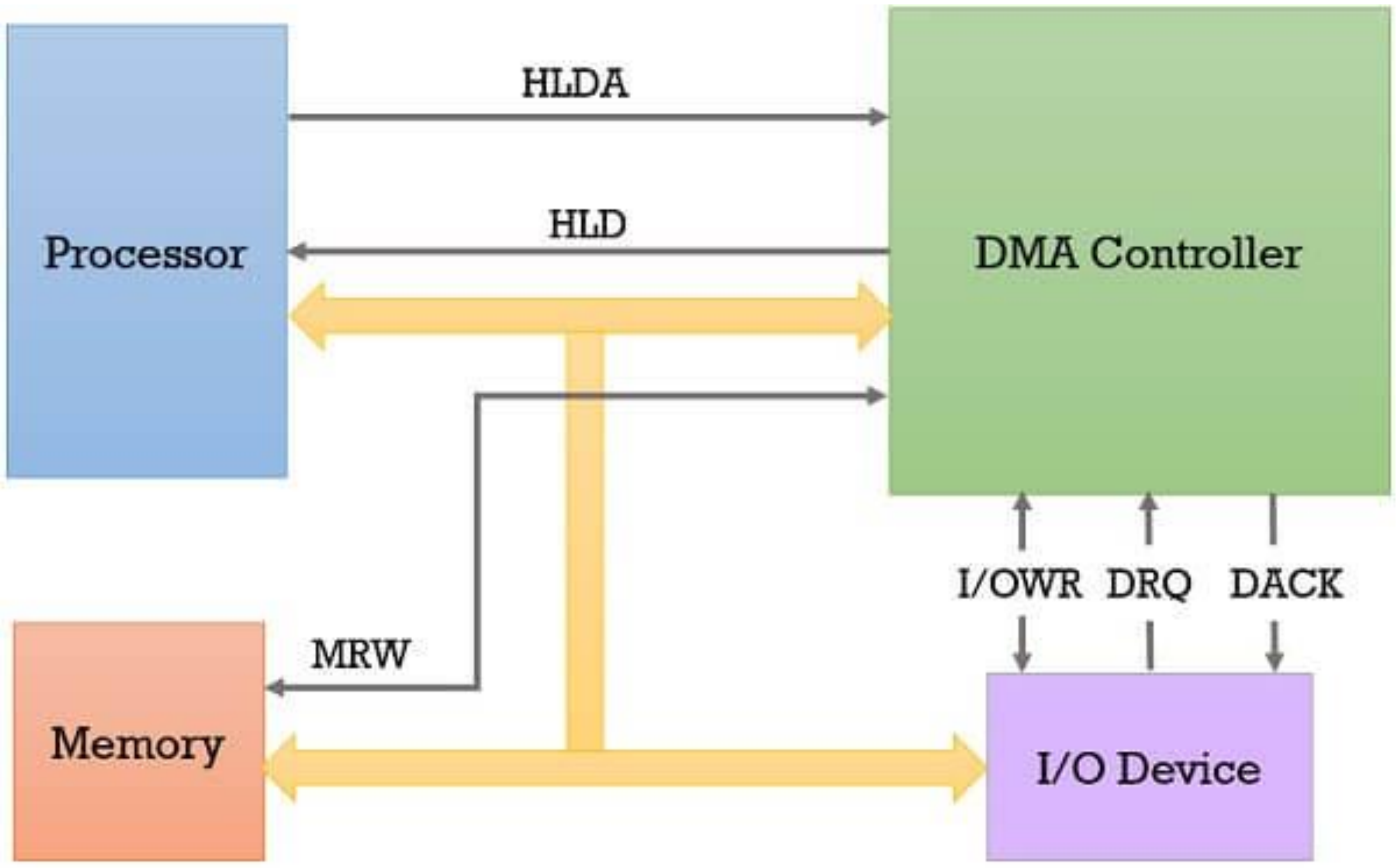

1
2
3
4
5
MORE STARS
MORE SAVING

Memory Type	Masked ROM	PROM	EPROM	EEPROM	Flash	NVRAM
Description	Preprogrammed	One Time Progamable	Erased using UV light	Electrically Reprogrammable	Electrically Reprogrammable	SRAM with a battery backup
Writeable	No	Once, with programmer	Yes, with programmer	Yes	Yes	Yes
Erase Size	n/a	n/a	Entire chip	Byte	Sector	Byte
Relative Cost	Inexpensive	Moderate	Moderate	Expensive	Moderate	Expensive
Relative Speed	Fast	Fast	Fast	Fast to read, slow to write	Fast to read, slow to write	fast







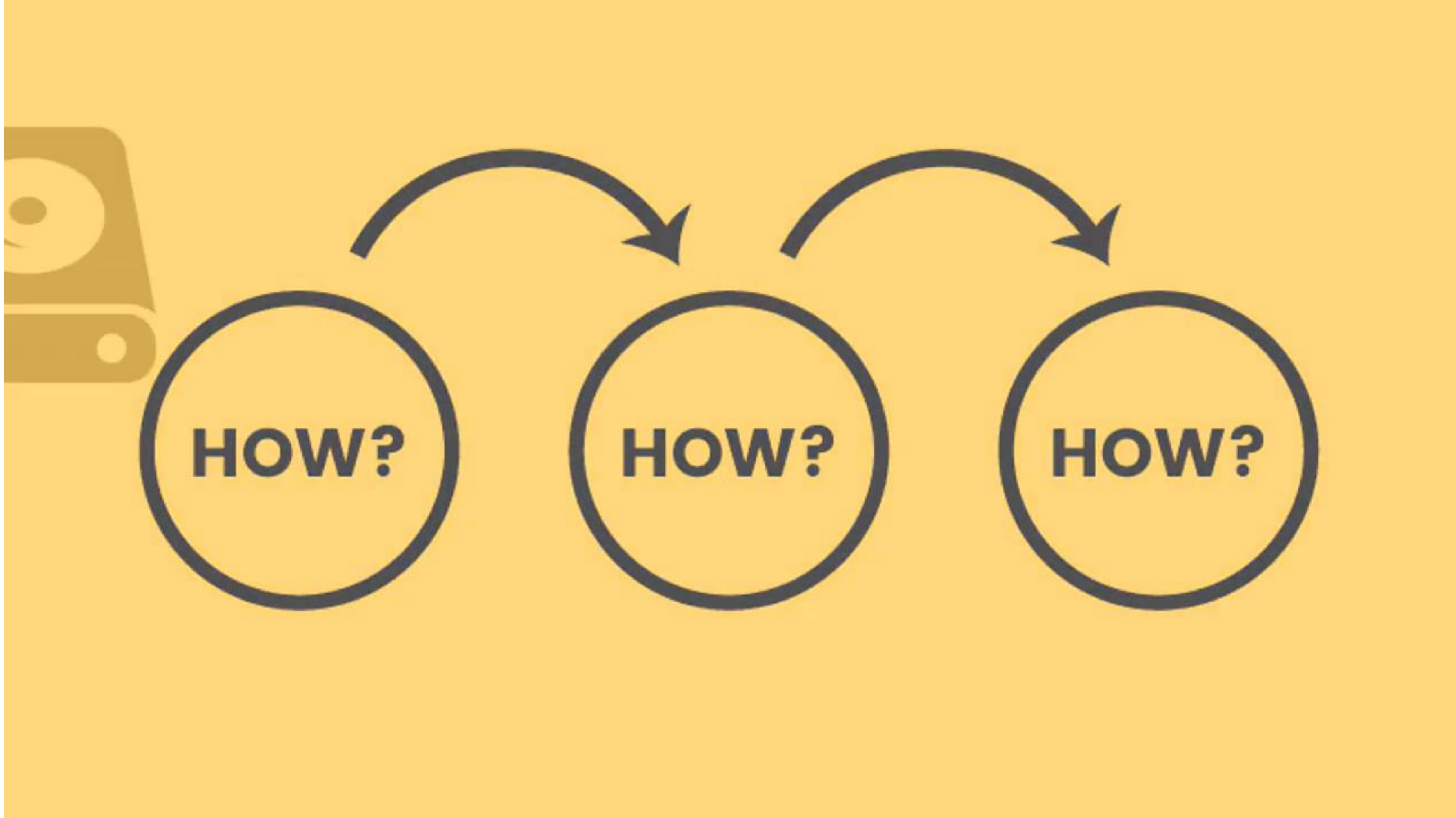


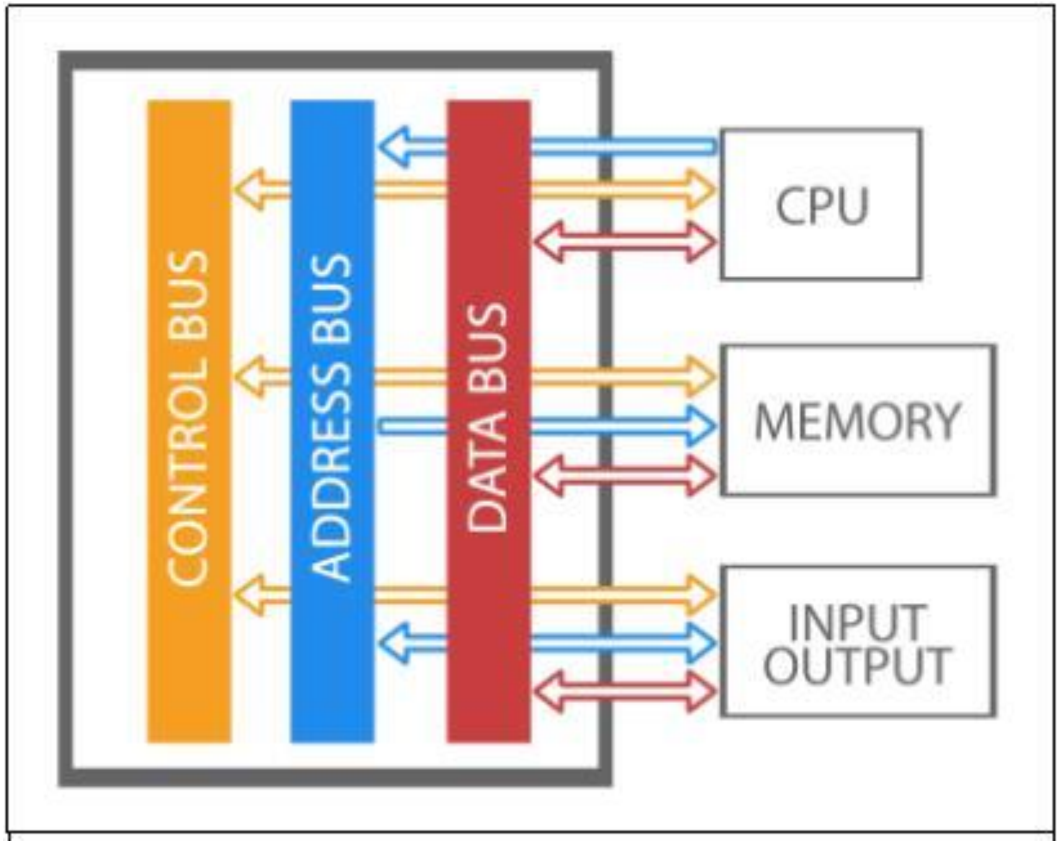
DMA Controller Data Transfer

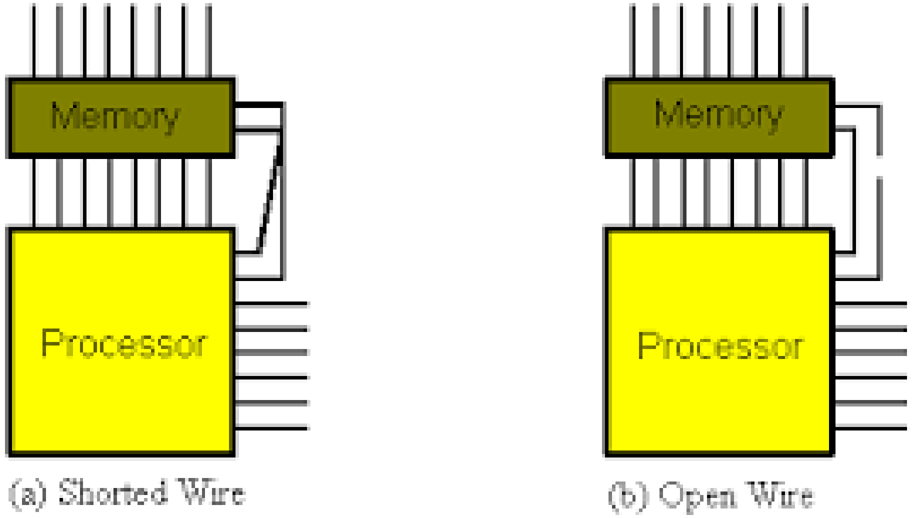
Direct Memory Access

- Mode of data transfer between the memory and I/O devices, without the involvement of the processor
- The DMA controller transfers the data in three modes
  - Burst Mode
  - Cycle Stealing Mode
  - Transparent Mode

Direct Memory Access







- **Data bus test**
- **Address bus test**
- **Device test**

Memory Testing

Data Bus Test

00000001
00000010
00000100
00001000
00010000
00100000
01000000
10000000

Need to check value placed on the data bus by the processor is correctly received by the memory device

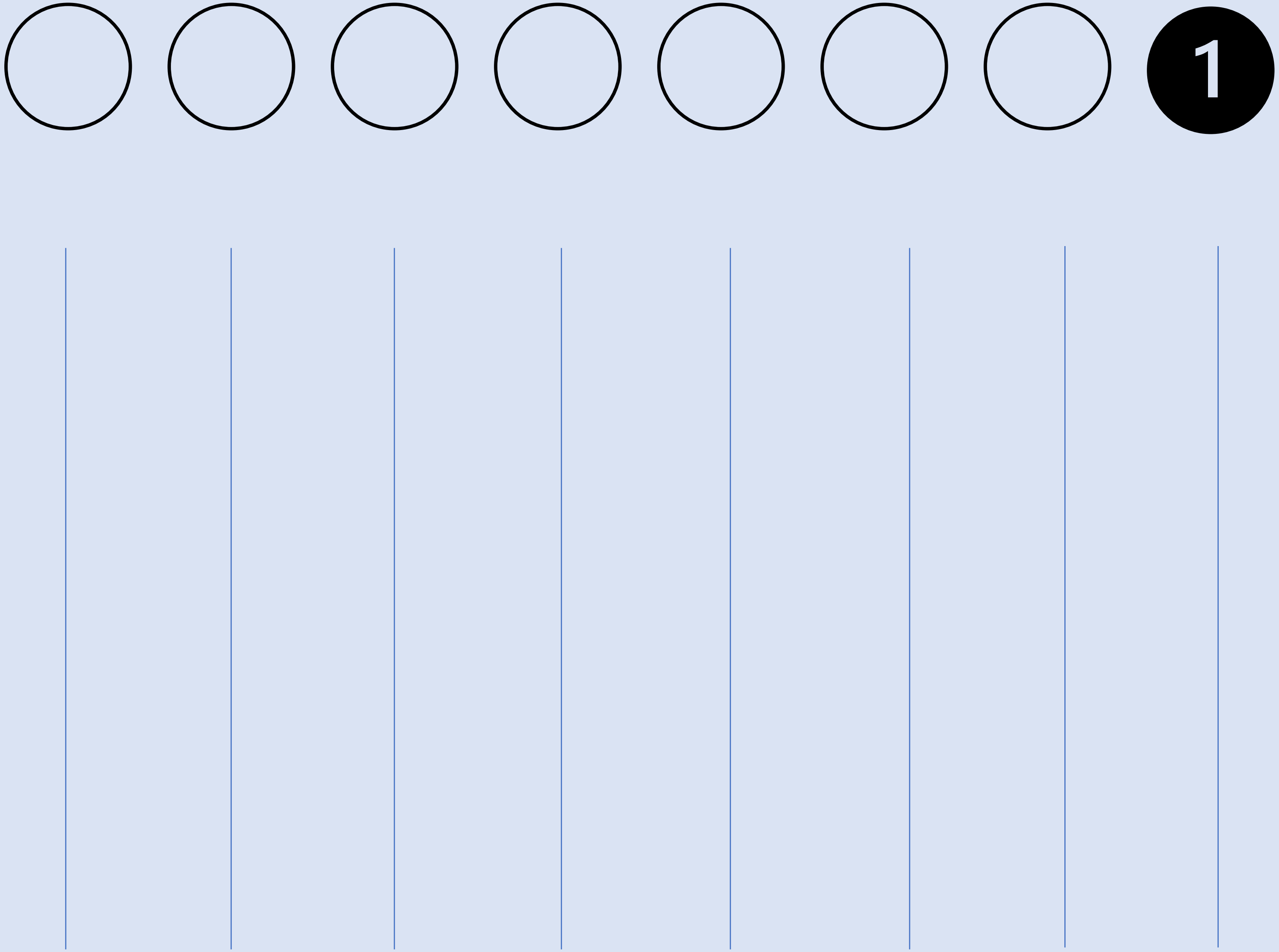
Write all possible data values and verify that the memory device stores each one successfully

Test one bit at a time

Data bus passes the test if each data bit can be set to 0 and 1, independently of the other data bits.

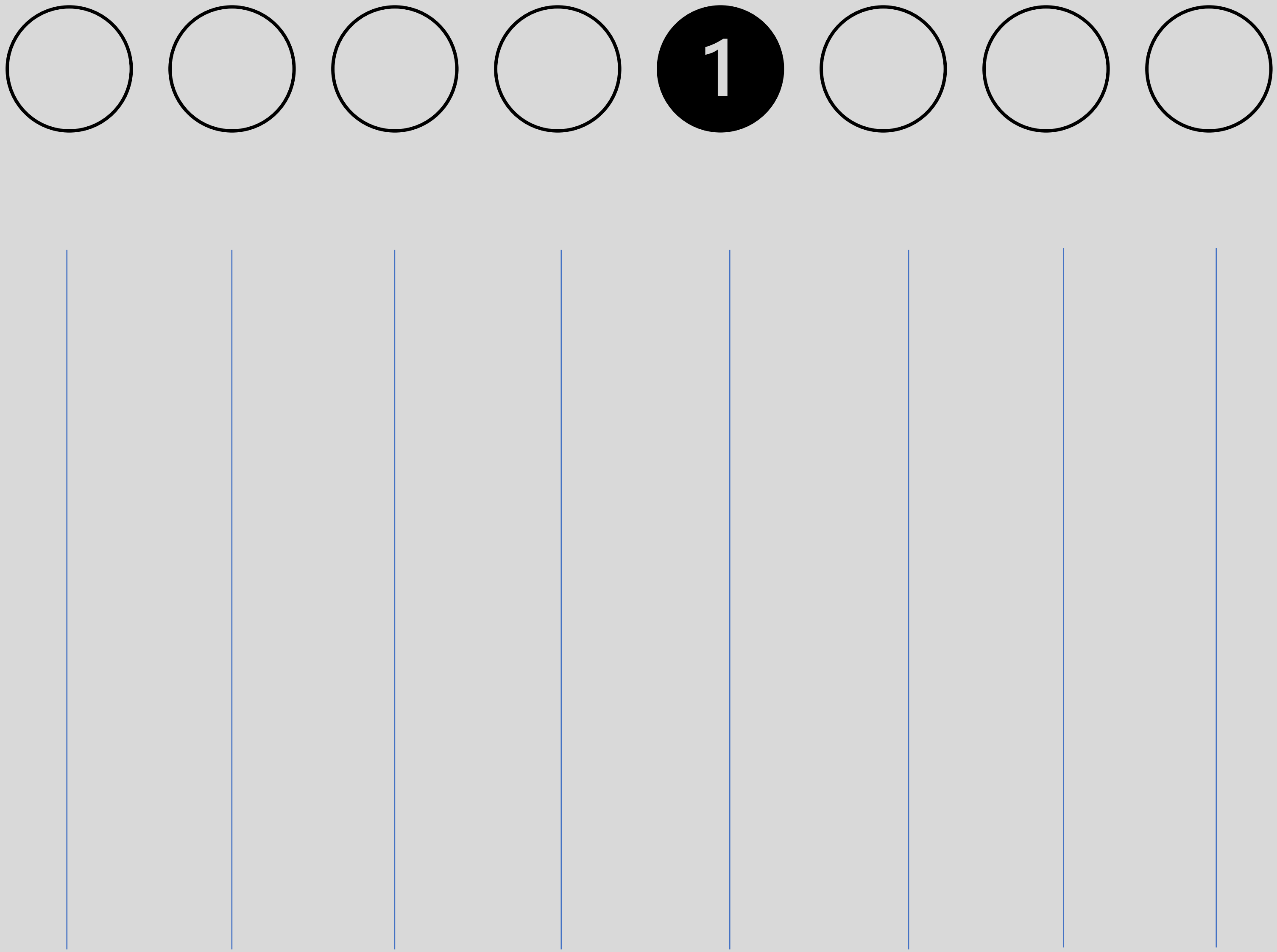
Walking 1's test

Reduces the number of test patterns from 2^n to n, where n is the width of the data bus.

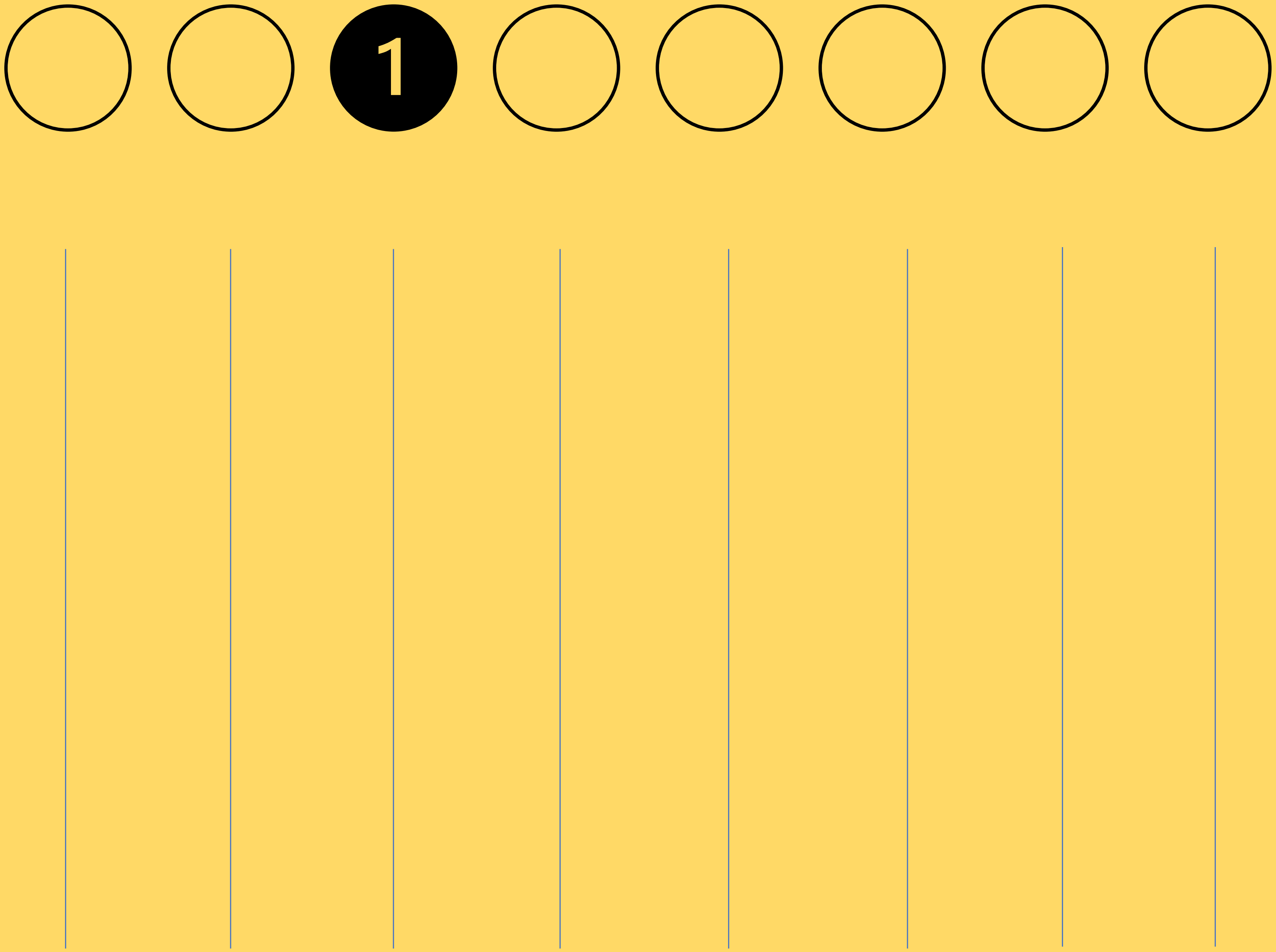

1

○ ○ ○ ○ ○ ○ 1 ○

○ ○ ○ ○ ○ ❶ ○ ○


1

○ ○ ○ 1 ○ ○ ○ ○


1

○ ❶ ○ ○ ○ ○ ○ ○

❶ ○ ○ ○ ○ ○ ○ ○

Address Bus Test

00001 h
00002 h
00004 h
00008 h
00010 h
00020 h
00040 h
00080 h

Overlapping memory locations

Try to isolate each address bit during testing

It is very important to confirm that each of the address pins can be set to 0 and 1 without affecting any of the others

Power-of-two addresses

After writing to one of the addresses, one must check that none of the others has been overwritten

Device Test

Memory Offset	Binary Value	Inverted Value
000h	00000001	11111110
001h	00000010	11111101
002h	00000011	11111100
003h	00000100	11111011
⋮	⋮	⋮
0FEh	11111111	00000000
0FFh	00000000	11111111

To test is that every bit in the device is capable of holding both 0 and 1

Easy to implement, longer to execute

Write and verify every memory location twice

Any data value can be selected for the first pass, so long as invert that value during the second

Increment test

Checksums

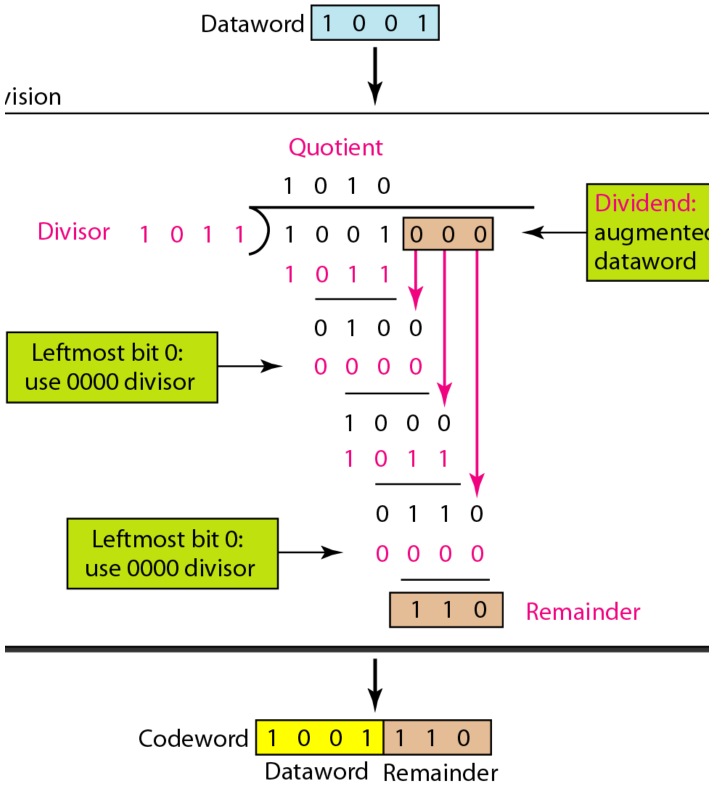
Simple sum-of-data checksum

Cannot detect many of the most common data errors

Storing Checksum

- Define it as a constant
- Store the checksum at some fixed location in memory
- Store the checksum is in another nonvolatile memory device

Cyclic Redundancy Checksum



Designed to detect the most common data errors.

When computing a CRC

- Set of data to be a very long string of 1's and 0's (called the message).
- Binary string is divided-in a rather peculiar way-by a smaller fixed binary string called the generator polynomial.
- Remainder of this binary long division is the CRC checksum.

By carefully selecting the generator polynomial, resulting checksum can detect most (but never all) errors within the message.

In addition, greater than 99.99% of all burst can be detected.

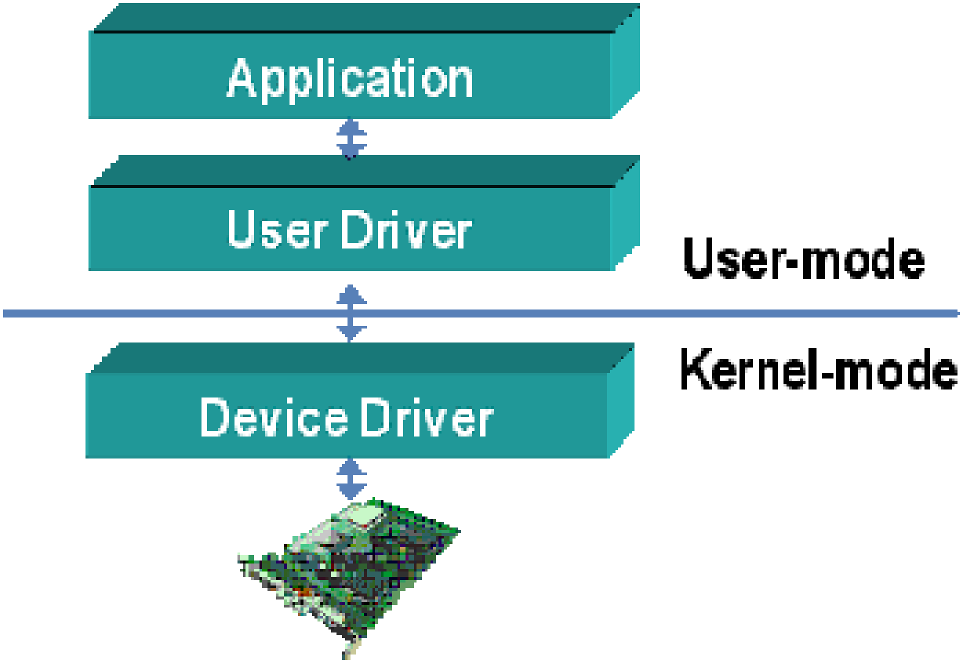
Working with Flash Memory

Writing data to a Flash is much harder

Three factors make writes difficult:

- Each memory location must be erased before it can be rewritten
- Only one sector, or block, of the device can be erased at a time
- Process of erasing the old data and writing the new varies from one manufacturer to another and is usually rather complicated

DEVICE DRIVER



- Goal: hide the hardware completely
- Programming interface that would not need to be changed if the underlying peripheral were replaced with another
- Good device driver design
  - Modularization - the structure of the overall software is easier to understand
  - Because there is only one module that ever interacts directly with the peripheral's registers, the state of the hardware can be more accurately tracked
  - Software changes that result from hardware changes are localized to the device driver

Driver implementation

- A data structure that overlaps the memory–mapped control and status registers of the device
- A set of variables to track the current state of the hardware and device driver
- A routine to initialize the hardware to a known state
- A set of routines that, taken together, provide an API for users of the device driver
- One or more interrupt service routines