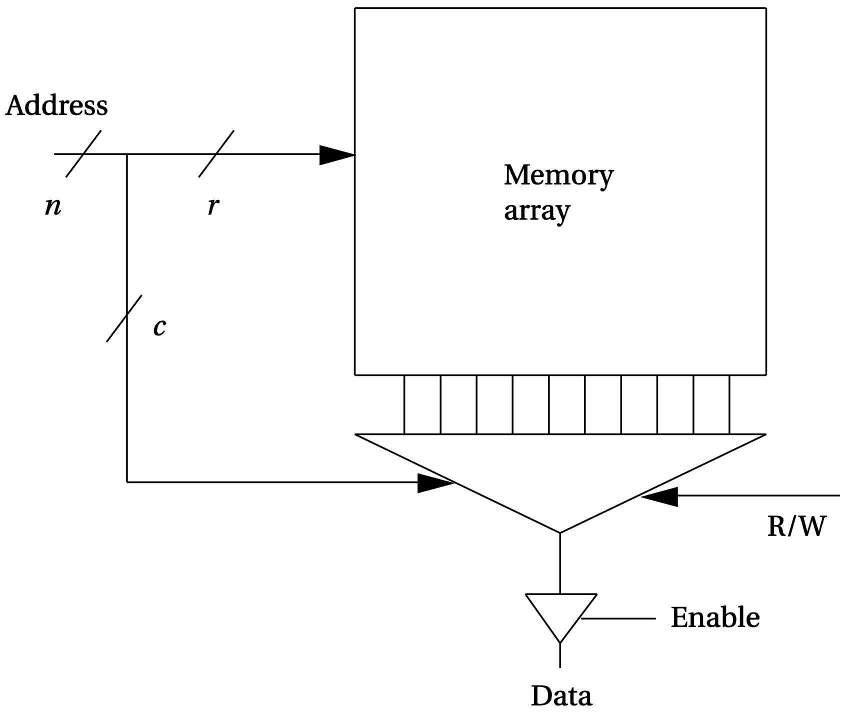
4.2 Memory components

- Several different types of memory:
  - DRAM.
  - SRAM.
  - Flash.
- Each type of memory comes in varying:
  - Capacities.
  - Widths.



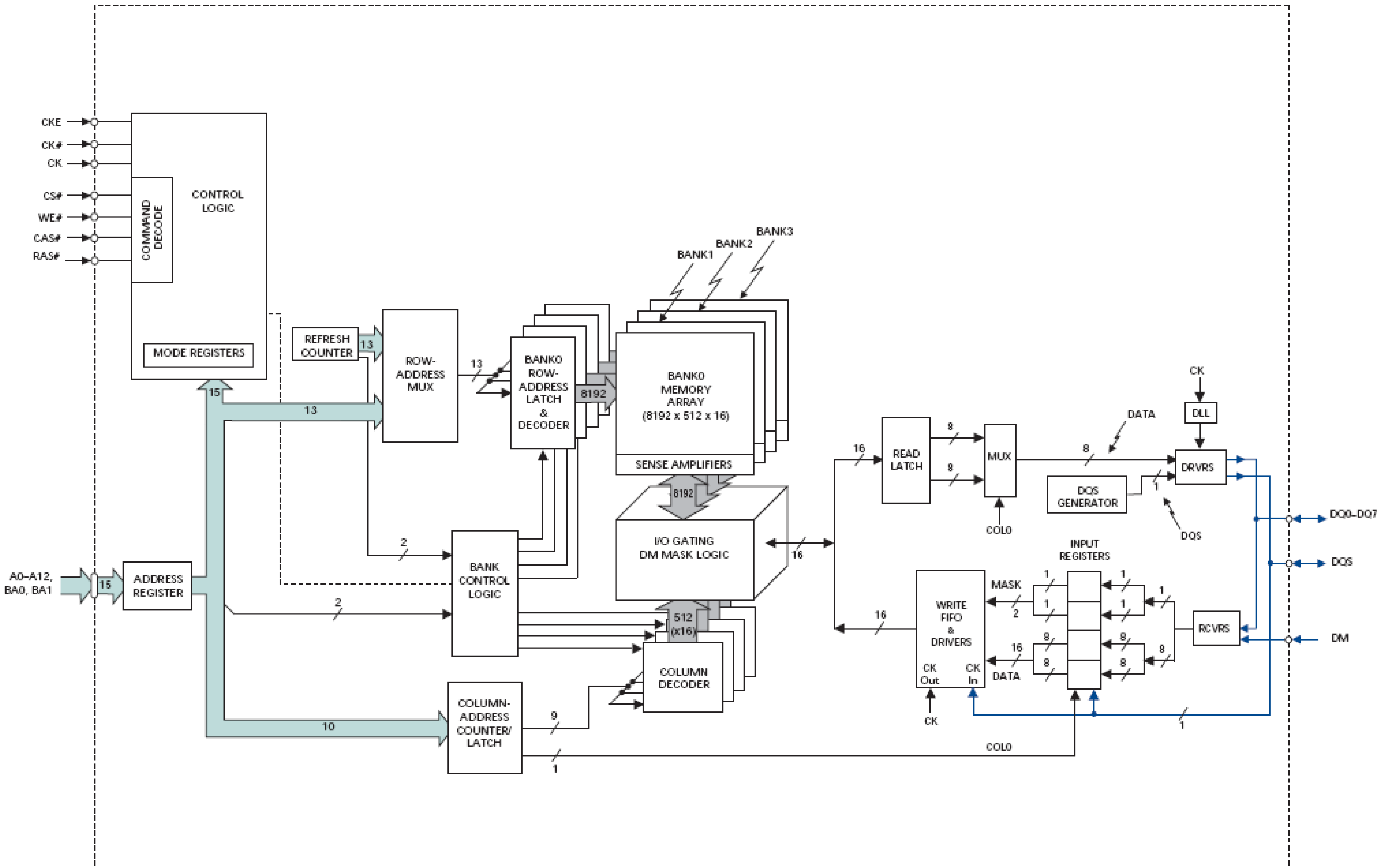
Random-access memory

- Dynamic RAM is dense, requires refresh.
  - Off-chip
  - Synchronous DRAM is dominant type.
  - SDRAM uses clock to improve performance, pipeline memory accesses.
- Static RAM is faster, less dense, consumes more power.
  - On-chip

DDR SDRAM operations

- Addressing (32M x 8)
  - 8M x 8 x 4 banks
  - Refresh count 8k
  - Row address 8k (A0 – A12)
  - Bank Address 4 (BA0, BA1)
  - Column address 1k (A0 – A9)
- CKE: clock enable
- RAS#, CAS#, WE#: command inputs
- DQ0 – DQ7: data input/output
- DQS: data strobe, edge aligned with read data and center in write data
- DM: input data mask

Functional block diagram



DDR SDRAM operations

Table 28: Truth Table 1 – Commands
CKE is HIGH for all commands shown except SELF REFRESH; All states and sequences not shown are illegal or reserved

Function	CS#	RAS#	CAS#	WE#	Address	Notes
DESELECT	H	X	X	X	X	1
NO OPERATION (NOP)	L	H	H	H	X	1
ACTIVE (select bank and activate row)	L	L	H	H	Bank/row	2
READ (select bank and column and start READ burst)	L	H	L	H	Bank/col	3
WRITE (select bank and column and start WRITE burst)	L	H	L	L	Bank/col	3
BURST TERMINATE	L	H	H	L	X	4
PRECHARGE (deactivate row in bank or banks)	L	L	H	L	Code	5
AUTO REFRESH or SELF REFRESH (enter self refresh mode)	L	L	L	H	X	6, 7
LOAD MODE REGISTER	L	L	L	L	Op-code	8

Table 29: Truth Table 2 – DM Operation
Used to mask write data, provided coincident with the corresponding data

Name (Function)	DM	DQ
Write enable	L	Valid
Write inhibit	H	X

DDR SDRAM operations

Notes:

1. DESELECT and NOP are functionally interchangeable.
2. BA0–BA1 provide bank address and A0–A*n* (128Mb: $n = 11$; 256Mb and 512Mb: $n = 12$; 1Gb: $n = 13$) provide row address.
3. BA0–BA1 provide bank address; A0–A*i* provide column address, (where A*i* is the most significant column address bit for a given density and configuration, see Table 2 on page 2) A10 HIGH enables the auto precharge feature (non persistent), and A10 LOW disables the auto precharge feature.
4. Applies only to READ bursts with auto precharge disabled; this command is undefined (and should not be used) for READ bursts with auto precharge enabled and for WRITE bursts.
5. A10 LOW: BA0–BA1 determine which bank is precharged. A10 HIGH: all banks are precharged and BA0–BA1 are "Don't Care."
6. This command is AUTO REFRESH if CKE is HIGH; SELF REFRESH if CKE is LOW.
7. Internal refresh counter controls row addressing while in self refresh mode, all inputs and I/Os are "Don't Care" except for CKE.
8. BA0–BA1 select either the mode register or the extended mode register (BA0 = 0, BA1 = 0 select the mode register; BA0 = 1, BA1 = 0 select extended mode register; other combinations of BA0–BA1 are reserved). A0–A*n* provide the op-code to be written to the selected mode register.

DDR SDRAM operations

Table 30: Truth Table 3 – Current State Bank *n* – Command to Bank *n* (to the same bank)

Notes: 1–6 apply to the entire table; Notes appear below

Current State	CS#	RAS#	CAS#	WE#	Command/Action	Notes
Any	H	X	X	X	DESELECT (NOP/continue previous operation)	
	L	H	H	H	NO OPERATION (NOP/continue previous operation)	
Idle	L	L	H	H	ACTIVE (select and activate row)	
	L	L	L	H	AUTO REFRESH	7
	L	L	L	L	LOAD MODE REGISTER	7
Row active	L	H	L	H	READ (select column and start READ burst)	10
	L	H	L	L	WRITE (select column and start WRITE burst)	10
	L	L	H	L	PRECHARGE (deactivate row in bank or banks)	8
Read (auto precharge disabled)	L	H	L	H	READ (select column and start new READ burst)	10
	L	H	L	L	WRITE (select column and start WRITE burst)	10, 12
	L	L	H	L	PRECHARGE (truncate READ burst, start PRECHARGE)	8
	L	H	H	L	BURST TERMINATE	9
Write (auto precharge disabled)	L	H	L	H	READ (select column and start READ burst)	10, 11
	L	H	L	L	WRITE (select column and start new WRITE burst)	10
	L	L	H	L	PRECHARGE (truncate WRITE burst, start PRECHARGE)	8, 11

DDR SDRAM operations

Table 31: Truth Table 4 – Current State Bank *n* – Command to Bank *m* (to the different bank)

Notes: 1–6 apply to the entire table; Notes appear on page 45

Current State	CS#	RAS#	CAS#	WE#	Command/Action	Notes
Any	H	X	X	X	DESELECT (NOP/continue previous operation)	
	L	H	H	H	NO OPERATION (NOP/continue previous operation)	
Idle	X	X	X	X	Any command otherwise allowed to bank *m*	
Row activating, active, or precharging	L	L	H	H	ACTIVE (select and activate row)	
	L	H	L	H	READ (select column and start READ burst)	7
	L	H	L	L	WRITE (select column and start WRITE burst)	7
	L	L	H	L	PRECHARGE	
Read (auto precharge disabled)	L	L	H	H	ACTIVE (select and activate row)	
	L	H	L	H	READ (select column and start new READ burst)	7
	L	H	L	L	WRITE (select column and start WRITE burst)	7, 9
	L	L	H	L	PRECHARGE	
Write (auto precharge disabled)	L	L	H	H	ACTIVE (select and activate row)	
	L	H	L	H	READ (select column and start READ burst)	7, 8
	L	H	L	L	WRITE (select column and start new WRITE burst)	7
	L	L	H	L	PRECHARGE	
Read (with auto-precharge)	L	L	H	H	ACTIVE (select and activate row)	
	L	H	L	H	READ (select column and start new READ burst)	7
	L	H	L	L	WRITE (select column and start WRITE burst)	7, 9
	L	L	H	L	PRECHARGE	
Write (with auto-precharge)	L	L	H	H	ACTIVE (select and activate row)	
	L	H	L	H	READ (select column and start READ burst)	7
	L	H	L	L	WRITE (select column and start new WRITE burst)	7
	L	L	H	L	PRECHARGE	

SDRAM operations

DESELECT

The DESELECT function (CS# HIGH) prevents new commands from being executed by the DDR SDRAM. The DDR SDRAM is effectively deselected. Operations already in progress are not affected.

NO OPERATION (NOP)

The NO OPERATION (NOP) command is used to instruct the selected DDR SDRAM to perform a NOP (CS# is LOW with RAS#, CAS#, and WE# are HIGH). This prevents unwanted commands from being registered during idle or wait states. Operations already in progress are not affected.

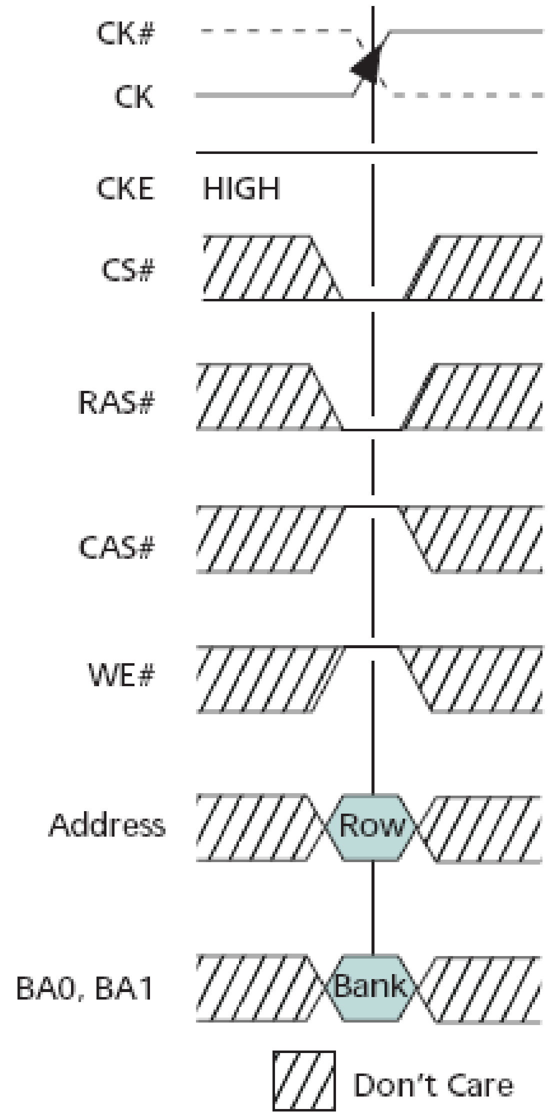
LOAD MODE REGISTER (LMR)

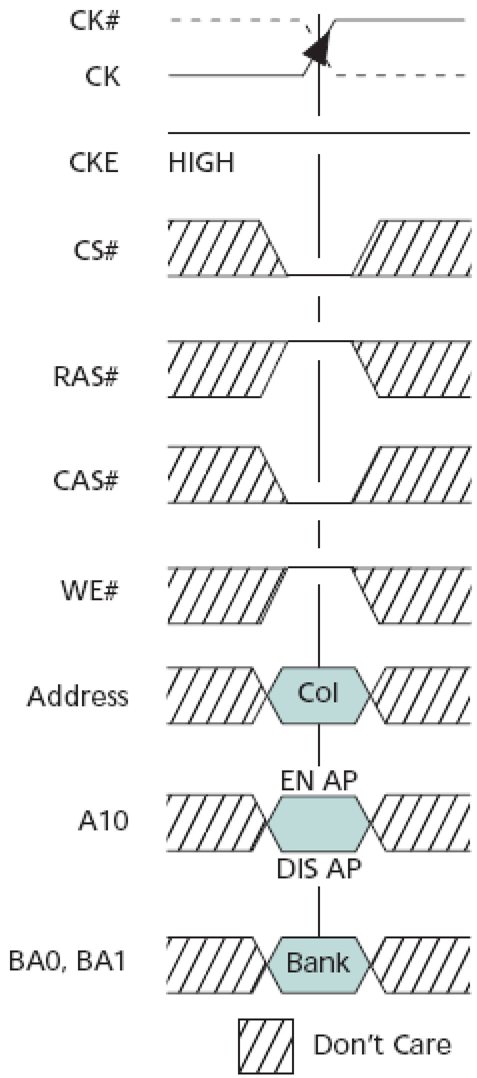
The mode registers are loaded via inputs A0–A*n* (see "REGISTER DEFINITION" on page 55). The LMR command can only be issued when all banks are idle, and a subsequent executable command cannot be issued until ^{t}MRD is met.

Activating a Specific Row in a Specific Bank

ACTIVE

- ⌘ An active command is used to open or active a row in a particular bank for a subsequent access, like a read or a write. The value of BA0, BA1 inputs selects the bank, and the address provided on inputs A0 ~A12 selects the row.



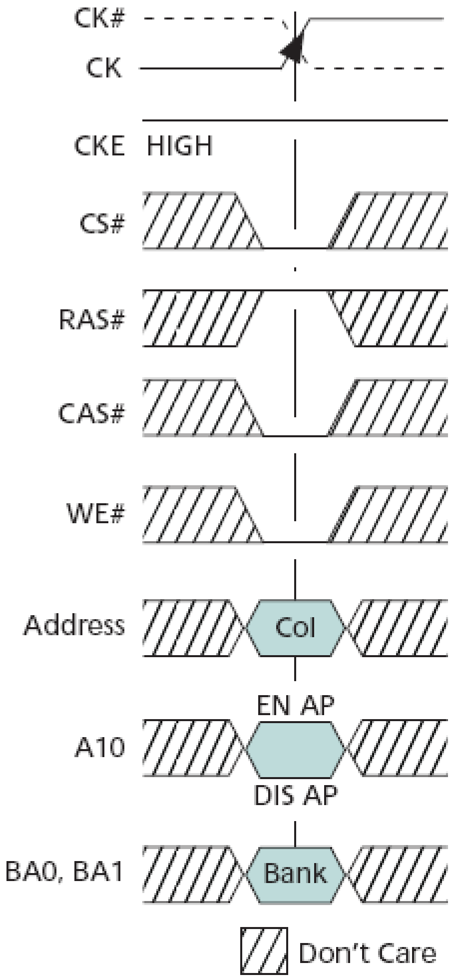


READ

- A read command is used to initiate a burst read access to an active row. The value of BA0, BA1 inputs selects the bank, and the address provided on inputs A0 ~ A12 selects the starting column location.

Note: EN AP = enable auto precharge; DIS AP = disable auto precharge.

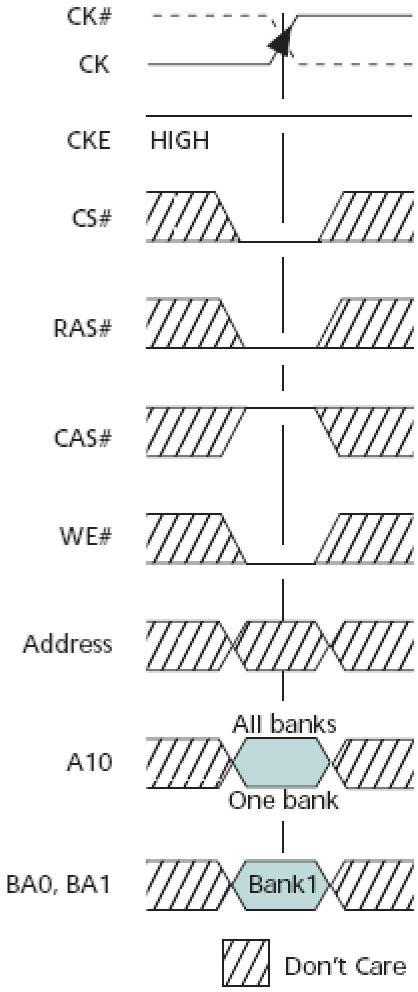
WRITE



Note: EN AP = enable auto precharge; and DIS AP = disable auto precharge.

- A write command is used to initiate a burst write access to an active row. The value of BA0, BA1 inputs selects the bank, and the address provided on inputs A0 ~ A12 selects the starting column location.

PRECHARGE



Notes: 1. If A10 is HIGH, bank address becomes "Don't Care."

- A precharge command is used to deactivate the open row in a particular bank or the open row in all banks. The value of BA0, BA1 inputs selects the bank, and the A10 input selects whether a single bank is precharged or whether all banks are precharged

SDRAM operations

BURST TERMINATE (BST)

The BURST TERMINATE command is used to truncate READ bursts (with auto precharge disabled). The most recently registered READ command prior to the BURST TERMINATE command will be truncated, as shown in “Operations” on page 52. The open page from which the READ burst was terminated remains open.

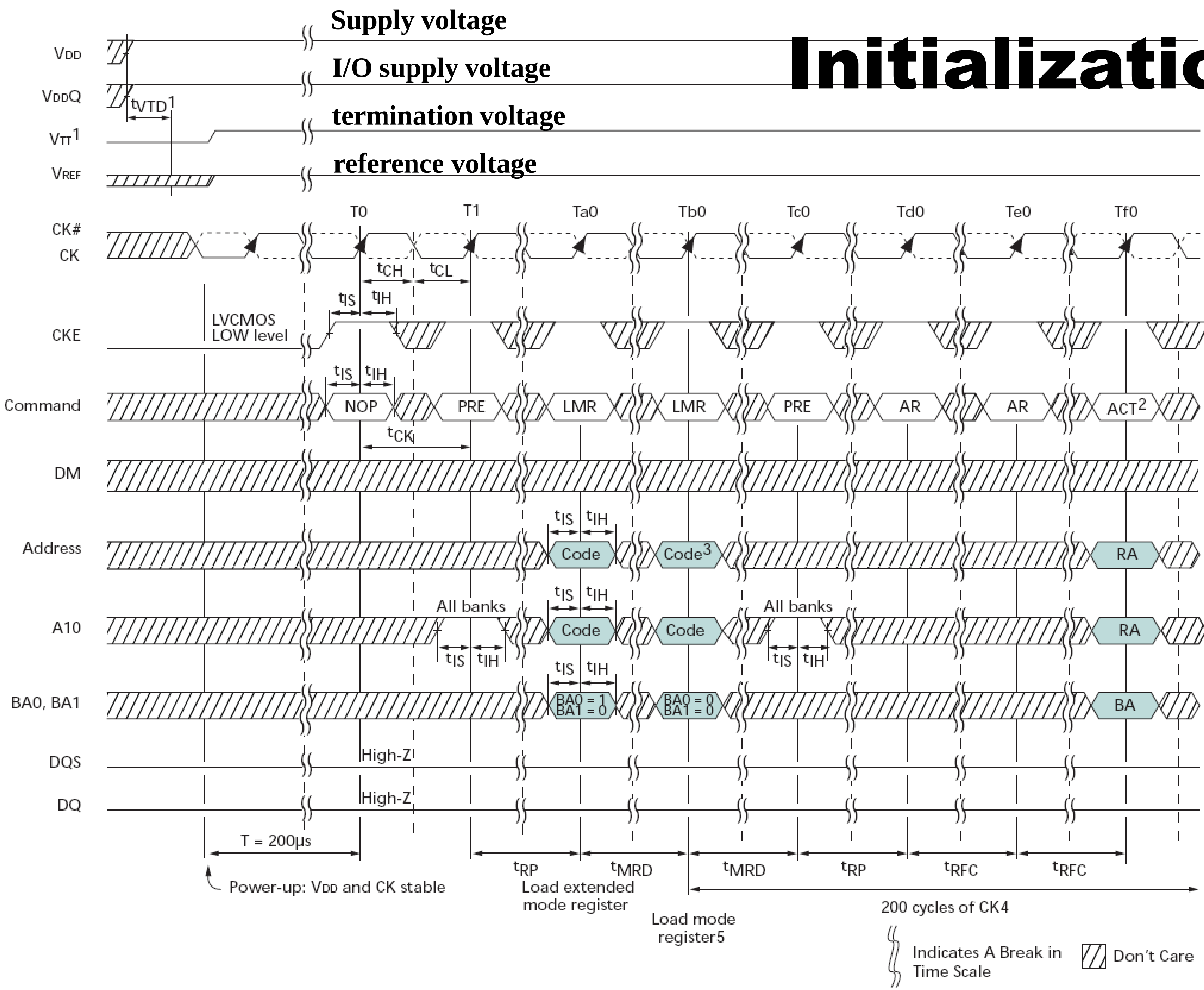
AUTO REFRESH (AR)

AUTO REFRESH is used during normal operation of the DDR SDRAM and is analogous to CAS#-before-RAS# (CBR) refresh in FPM/EDO DRAMs. This command is nonpersistent, so it must be issued each time a refresh is required. All banks must be idle before an AUTO REFRESH command is issued.

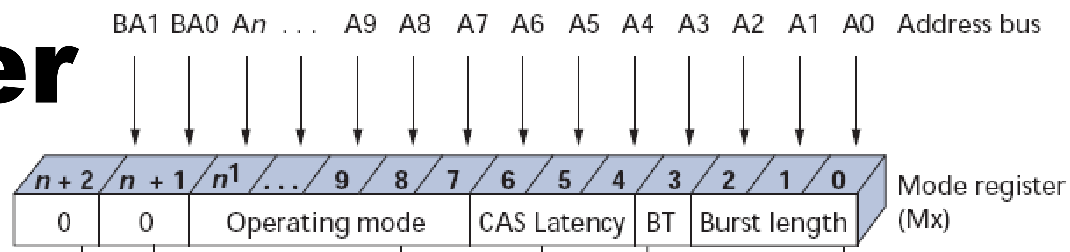
SELF REFRESH

The SELF REFRESH command can be used to retain data in the DDR SDRAM, even if the rest of the system is powered down. The SELF REFRESH command is initiated like an AUTO REFRESH command except CKE is disabled (LOW).

Initialization



Mode register



M*n* + 2	M*n* + 1	Mode Register Definition
0	0	Base mode register
0	1	Extended mode register
1	0	Reserved
1	1	Reserved

M3	Burst Type
0	Sequential
1	Interleaved

M2	M1	M0	Burst Length
0	0	0	Reserved
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

M*n*	. . .	M9	M8	M7	M6–M0	Operating Mode
0	0	0	0	0	Valid	Normal operation
0	0	0	1	0	Valid	Normal operation/reset DLL
–	–	–	–	–	–	All other states reserved

M6	M5	M4	CAS Latency
0	0	0	Reserved
0	0	1	Reserved
0	1	0	2
0	1	1	3 (-5B only)
1	0	0	Reserved
1	0	1	Reserved
1	1	0	2.5
1	1	1	Reserved

- The mode register is used to define a specific DDR SDRAM mode of operation.

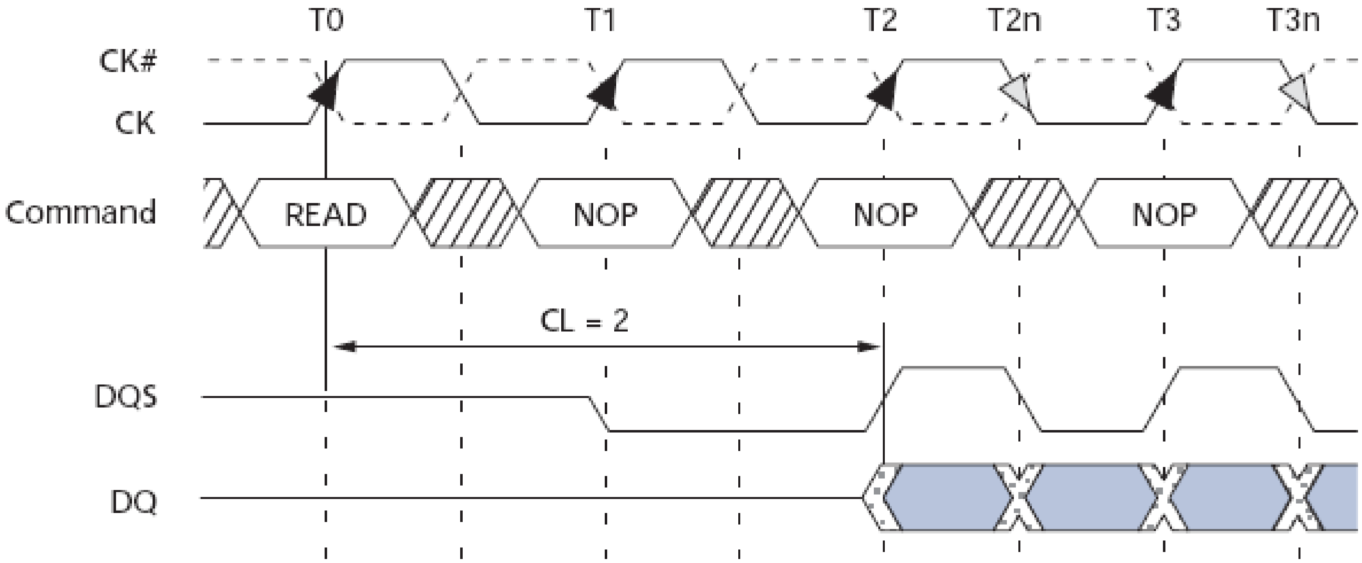
Burst length and Burst type

Table 34: Burst Definition

Burst Length	Starting Column Address			Order of Accesses Within a Burst	
				Type = Sequential	Type = Interleaved
2	–	–	**A0**	–	–
	–	–	0	0-1	0-1
	–	–	1	1-0	1-0
4	–	**A1**	**A0**	–	–
	–	0	0	0-1-2-3	0-1-2-3
	–	0	1	1-2-3-0	1-0-3-2
	–	1	0	2-3-0-1	2-3-0-1
	–	1	1	3-0-1-2	3-2-1-0
8	**A2**	**A1**	**A0**	–	–
	0	0	0	0-1-2-3-4-5-6-7	0-1-2-3-4-5-6-7
	0	0	1	1-2-3-4-5-6-7-0	1-0-3-2-5-4-7-6
	0	1	0	2-3-4-5-6-7-0-1	2-3-0-1-6-7-4-5
	0	1	1	3-4-5-6-7-0-1-2	3-2-1-0-7-6-5-4
	1	0	0	4-5-6-7-0-1-2-3	4-5-6-7-0-1-2-3
	1	0	1	5-6-7-0-1-2-3-4	5-4-7-6-1-0-3-2
	1	1	0	6-7-0-1-2-3-4-5	6-7-4-5-2-3-0-1
	1	1	1	7-0-1-2-3-4-5-6	7-6-5-4-3-2-1-0

CAS latency

- The CAS latency is the delay, in clock cycles, between the registration of a READ command and the availability of the first bit of output data, which can be set 2, 2,5, or 3 clocks.



Read-only memory

- ROM may be programmed at factory.
- Flash is dominant form of field-programmable ROM.
  - Electrically erasable, must be block erased.
  - Random access, but write/erase is much slower than read.
  - NOR flash is more flexible.
  - NAND flash is more dense.

Flash memory

- Non-volatile memory.
  - Flash can be programmed in-circuit.
- Random access for read.
- To write:
  - Erase a block to 1.
  - Write bits to 0.

Flash writing

- Write is much slower than read.
  - 1.6 μs write, 70 ns read.
- Blocks are large (approx. 1 Mb).
- Writing causes wear that eventually destroys the device.
  - Modern lifetime approx. 1 million writes.

Types of flash

- NOR:
  - Word-accessible read.
  - Erase by blocks.
- NAND:
  - Read by pages (512-4K bytes).
  - Erase by blocks.
- NAND is cheaper, has faster erase, sequential access times.

숙제

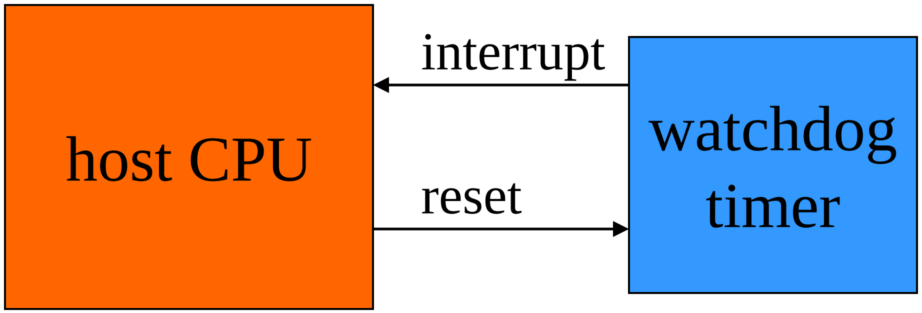
- 중간고사 문제를 남의 도움을 받지 않고 정답을 작성하여 4/26까지 제출
- NAND flash memory에 대해서 조사하여 자세한 보고서를 5/3까지 제출

4.3 I/O devices

- Timers and counter are very similar:
  - a timer is decremented by a periodic signal;
  - a counter is incremented by an asynchronous, occasional signal.
- When the watchdog timer rolls over. It generates an interrupt to the system.

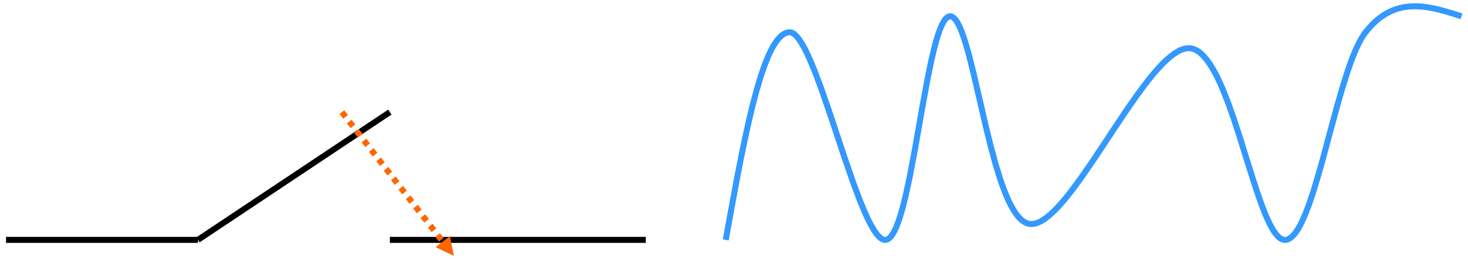
Watchdog timer

- Watchdog timer is periodically reset by system timer before it reaches this time-out limit.
- If the watchdog timer reaches this limit , it generates an interrupt to reset the host.



Switch debouncing

⌘A switch must be debounced to multiple contacts caused by eliminate mechanical bouncing:

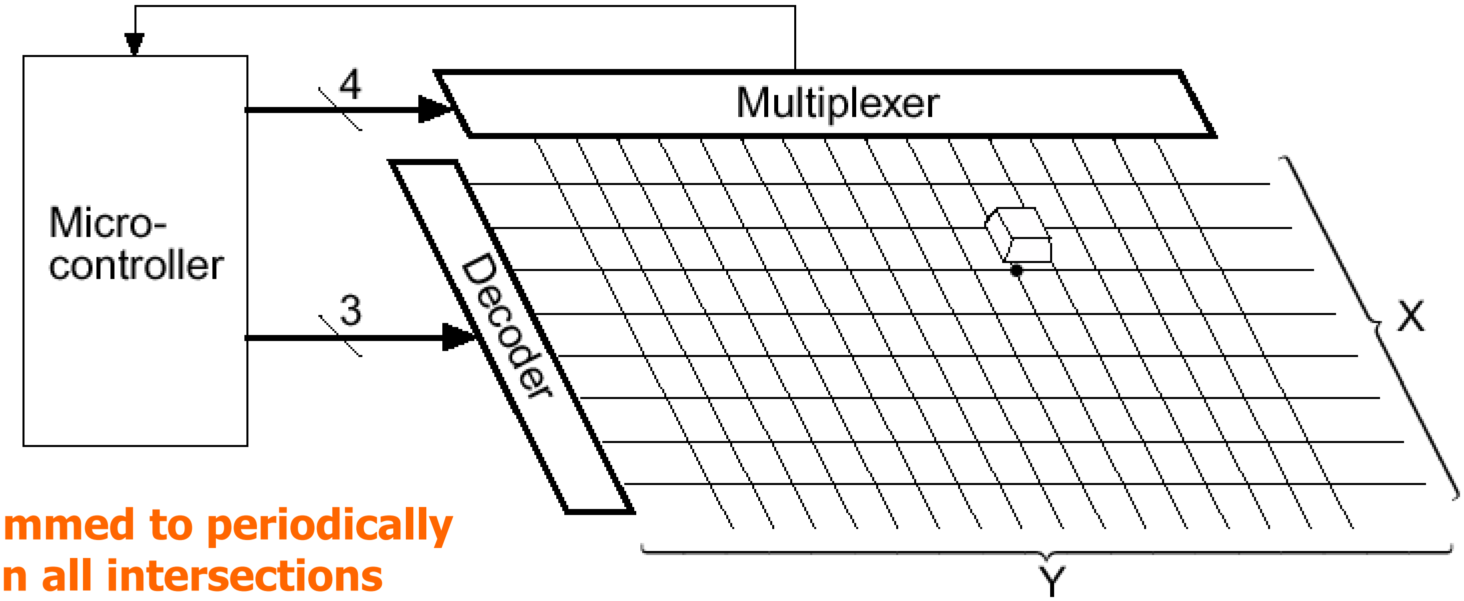


Encoded keyboard

- An array of switches is read by an encoder.
- A 4-bit microprocessor on a keyboard
  - Debouncing: wait and see (waits for 10~ 20 ms)
  - ASCII code for each key is stored in a LUT.
  - Scanned keyboard: one row at a time
    - Control-Q
    - Rollover (pressing another before releasing a key) may not be allowed
- N-key rollover can be programmed, which remembers multiple key depressions.

Keyboard scan matrix

To allow for "rollover" identifies the depressing(*make code*) and release(*break code*)



programmed to periodically scan all intersections

8 X 16 = 128 intersections

Standard Keyboard Layout

- **A standard computer keyboard has about 100 keys.**
- **Most keyboards use the QWERTY layout, named for the first six keys in the top row of letters.**

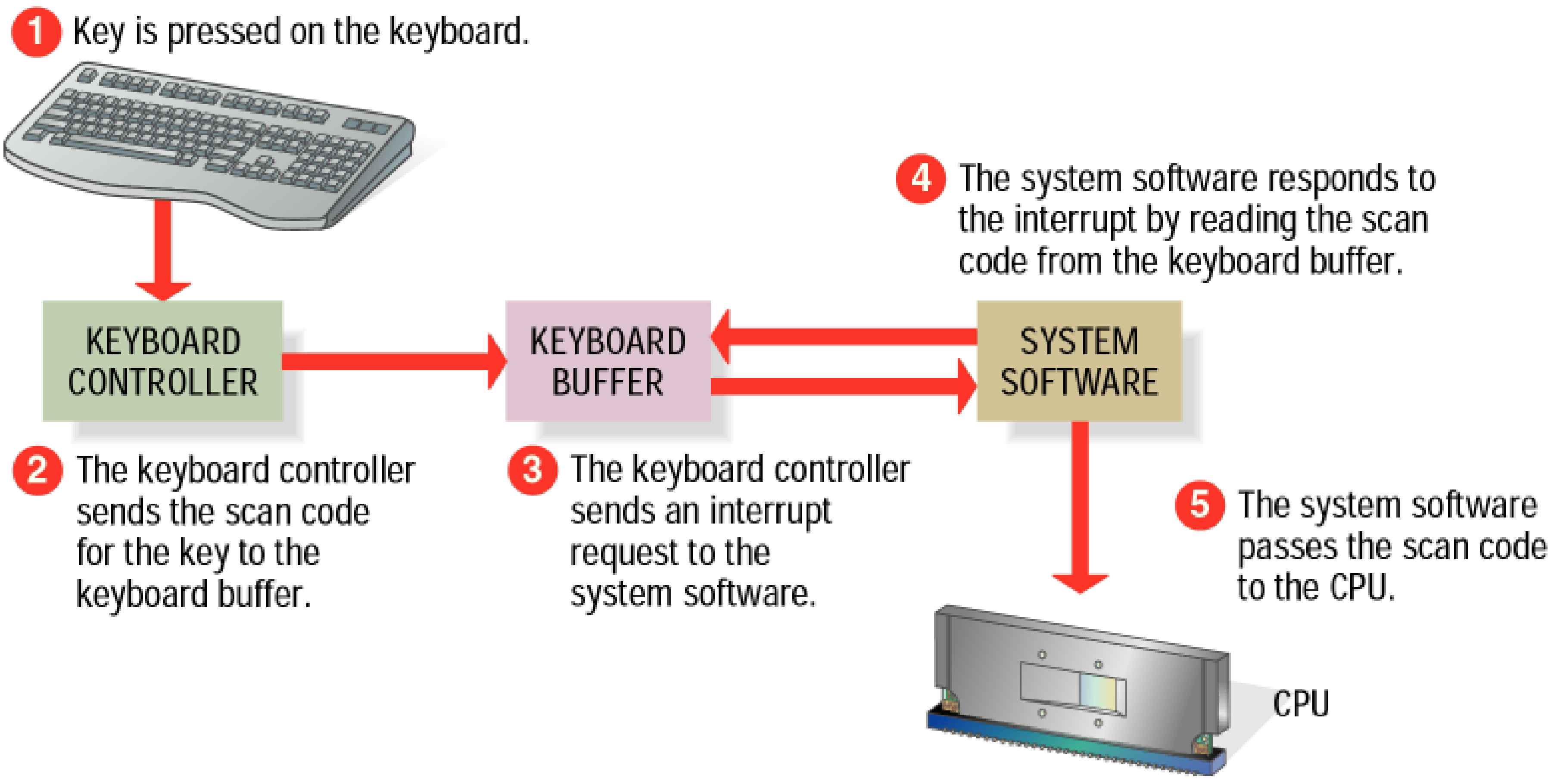
How a Keyboard Works

- A keyboard is a lot like a miniature computer.
- It has its own processor and circuitry that carries information to and from that processor.
- A large part of this circuitry makes up the key matrix.

Parts of Keyboard Circuitry

- **Keyboard controller**
- **Keyboard buffer**
- **Scan code**
- **Interrupt request**

How the Computer Accepts Input from the Keyboard



4.5.1 System architectures

- Architectures and components:
  - software;
  - hardware.
- Some software is very hardware-dependent (HdS).

Hardware platform architecture

- Contains several elements:
  - CPU;
  - bus;
  - memory;
  - I/O devices: networking, sensors, actuators, etc.
- How big/fast much each one be?
- How are they connected?

Software architecture

- Functional description must be broken into pieces:
  - division among people;
  - conceptual organization;
  - performance;
  - testability;
  - maintenance.

HW/SW architectures

- Hardware and software are intimately related:
  - software doesn't run without hardware;
  - how much hardware you need is determined by the software requirements:
    - speed;
    - memory.

Evaluation boards

- Designed by CPU manufacturer or others.
- Includes CPU, memory, some I/O devices.
- May include prototyping section.
- CPU manufacturer often gives out evaluation board netlist---can be used as starting point for your custom board design.

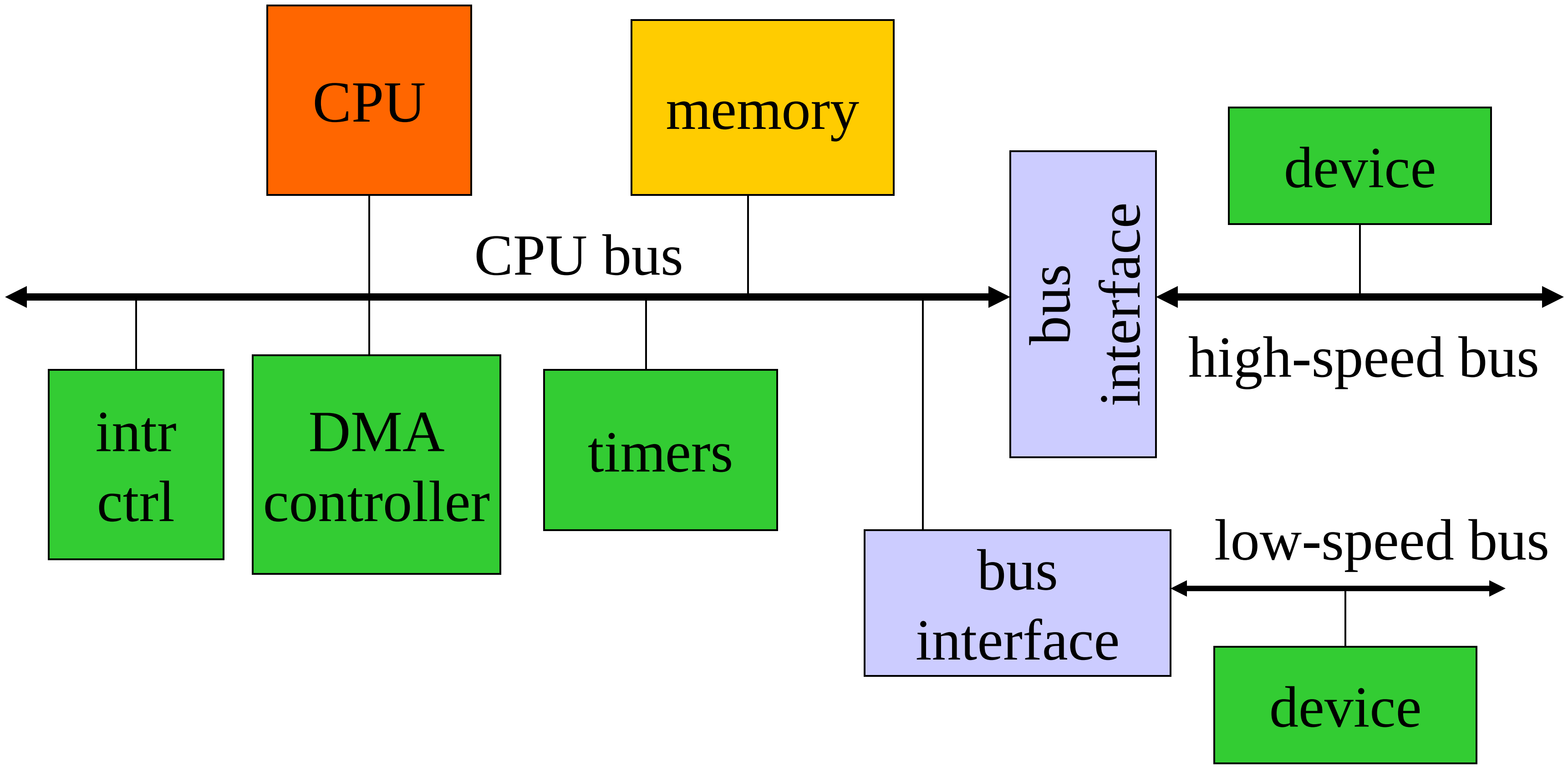
Adding logic to a board

- Programmable logic devices (PLDs) provide low/medium density logic.
- Field-programmable gate arrays (FPGAs) provide more logic and multi-level logic.
- Application-specific integrated circuits (ASICs) are manufactured for a single purpose.

4.5.3 The PC as a platform

- Advantages:
  - cheap and easy to get;
  - rich and familiar software environment.
- Disadvantages:
  - requires a lot of hardware resources;
  - not well-adapted to real-time.

Typical hardware platform



Typical busses

- PCI: standard for high-speed interfacing
  - 33 or 66 MHz.
  - PCI Express (PCIe): serial link.
    - 4 data wires per lane,
    - V1.x: 250 MB/s per lane
    - V2.0: 500 MB/s per lane
    - V3.0: 1GB/s per lane
- USB (Universal Serial Bus), Firewire (IEEE 1394): relatively low-cost serial interface with high speed.

Software elements

- IBM PC uses BIOS (Basic I/O System) to implement low-level functions:
  - boot-up;
  - minimal device drivers.
- BIOS has become a generic term for the lowest-level system software.
- Boot firmware
  - designed to be the first code run by a PC when powered on.
  - identify, test, and initialize system devices such as the video display card, hard disc, and floppy disc and other hardware.
  - prepare the machine into a known state, so that software stored on compatible media can be loaded, executed, and given control of the PC
  - This process is known as booting, or booting up, which is short for bootstrapping.

Software elements

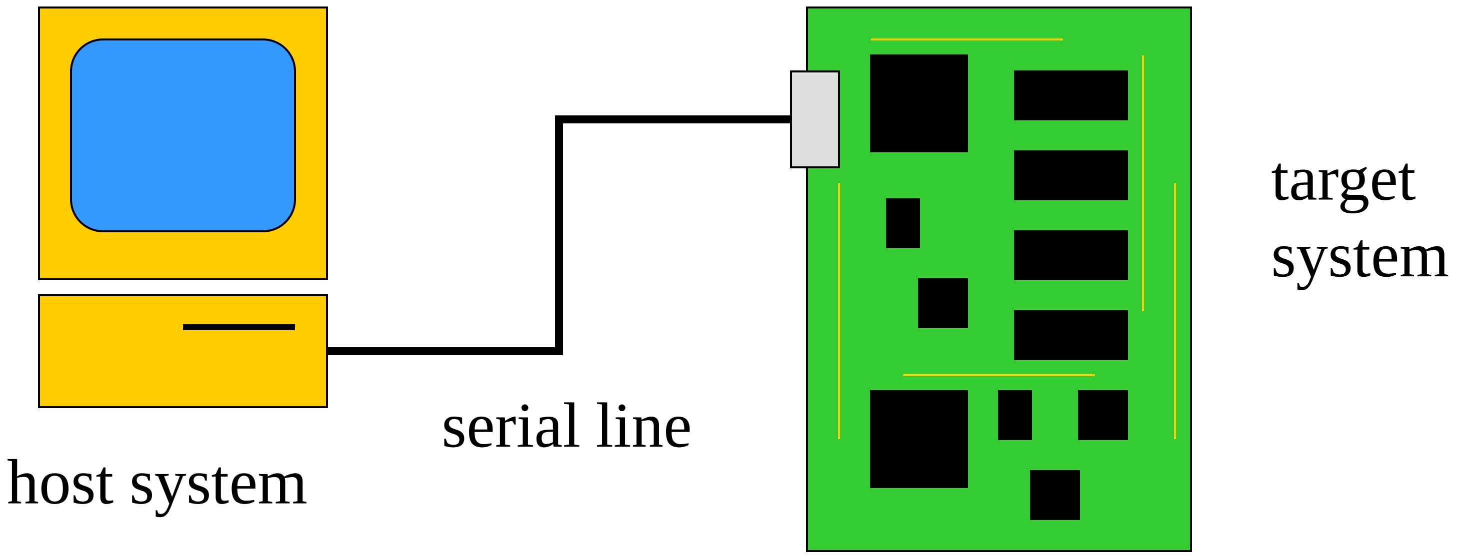
- BIOS programs are stored on a flash ROM and are built to work with various devices that make up the complementary chipset of the system.
- They provide a small library of basic input/output functions that can be called to operate and control the peripherals such as the keyboard, text display functions and so forth.
- In the IBM PC and AT, certain peripheral cards such as hard-drive controllers and video display adapters carried their own BIOS extension ROM, which provided additional functionality.
- OS and executive software, designed to supersede this basic firmware functionality, will provide replacement software interfaces to applications.

4.6 Debugging embedded systems

- Challenges:
  - target system may be hard to observe;
  - target may be hard to control;
  - may be hard to generate realistic inputs;
  - setup sequence may be complex.

Host/target design

⌘Use a host system to prepare software for target system:



Host-based tools

- Cross compiler:
  - compiles code on host for target system.
- Cross debugger:
  - displays target state, allows target system to be controlled.
  - by establishing a debug message protocol and using an interface like TCP/IP for communication between host development system and the target system, where the application to be debugged actually runs.

Software for debuggers

- A monitor, which is a small debug handler application, should run in user space on the target. It usually idles in user space memory and gets triggered by a dedicated debug interrupt.
- This is when it starts sending status information via a dedicated TCP/IP port to the host system where the debugger itself is waiting to pick up the data it receives.

Software for debuggers

- The debug interrupt could be caused by a breakpoint or data watchpoint being hit. It could also be triggered by an explicit action on the debug host.
- The developer telling the debugger to attach to a specific running process or telling the debugger to stop a specific thread.

Breakpoints

- A breakpoint allows the user to stop execution, examine system state, and change state.
- Replace the breakpointed instruction with a subroutine call to the monitor program.
- Can you set breakpoints in programs running out of ROM? No

ARM breakpoints

uninstrumented code		code with breakpoint	
0x400	MUL r4,r6,r6	0x400	MUL r4,r6,r6
0x404	ADD r2,r2,r4	0x404	ADD r2,r2,r4
0x408	ADD r0,r0,#1	0x408	ADD r0,r0,#1
0x40c	B loop	0x40c	BL bkpoint

uninstrumented code → code with breakpoint

Breakpoint handler actions

- Save registers.
- Allow user to examine machine.
- Before returning, restore system state.
  - (when the breakpoint is erased) Safest way to continue execution is to replace back the original instruction while fixing the return address.
  - (when the breakpoint is to remain) Put another temp breakpoint after replacing back the original instruction. When reached to the temp breakpoint after executing the original instruction, replace back the original breakpoint, remove the temp breakpoint, and resume execution.

In-circuit emulators (ICE)

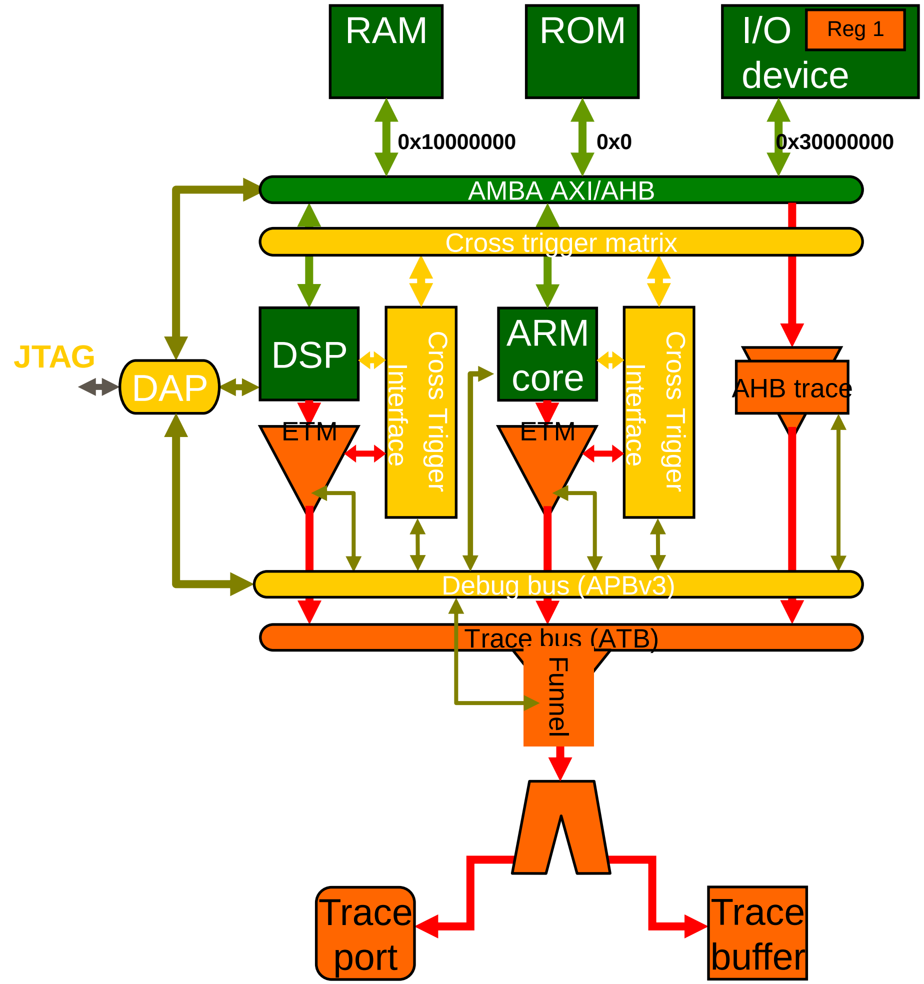
- A microprocessor in-circuit emulator is a specially-instrumented microprocessor.
- Allows you to stop execution, examine CPU state, modify registers.
- the emulator is a bridge between your target and your PC, giving you both an interactive terminal peering deeply into the target, while providing a rich set of debugging resources.

History

- In the beginning, there was the ROM debug monitor.
- After that the in-circuit emulator (ICE) came. By using special bond-out versions of processors, an ICE provides capabilities far beyond those of a simple ROM monitor.
- Now, dedicated debug circuitry is integrated into their chips. Or, simply software debug capabilities are added to their existing JTAG ports. Collectively, we'll call these technologies on-chip debug. Such hardware-based capabilities take the place of a software debug monitor, yet offer some additional features previously associated only with emulators.

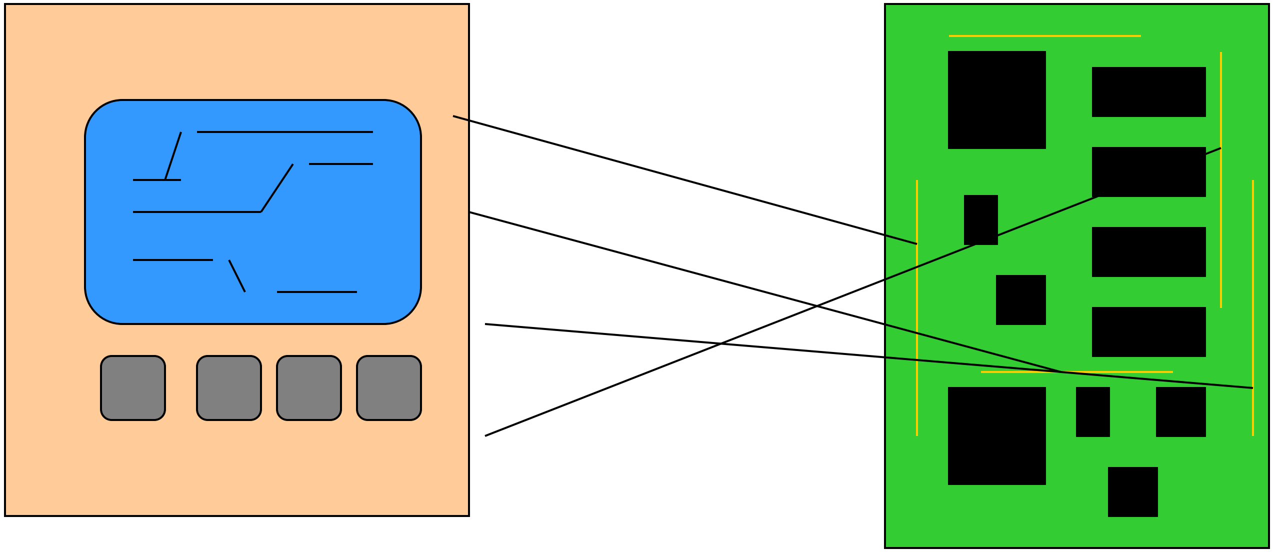
What does the debugger need to know?

- Programmers' model:
  - System components
  - System busses
  - Base addresses
  - Device registers
- Debug access description:
  - Debug access to processors
  - Other debug devices
  - Debug interconnections

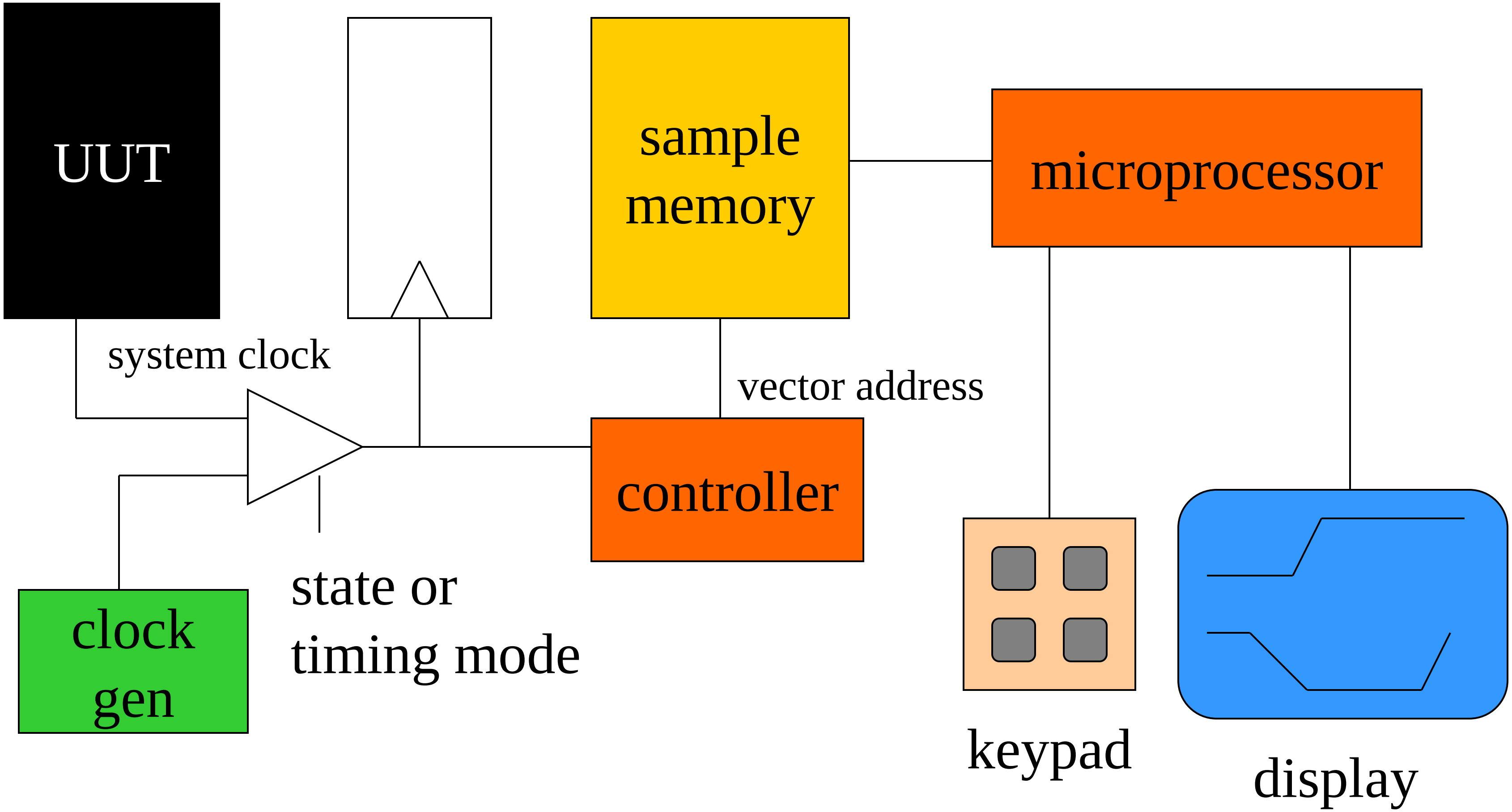


Logic analyzer

- A logic analyzer can be regarded as an array of low-grade oscilloscopes:



Logic analyzer architecture

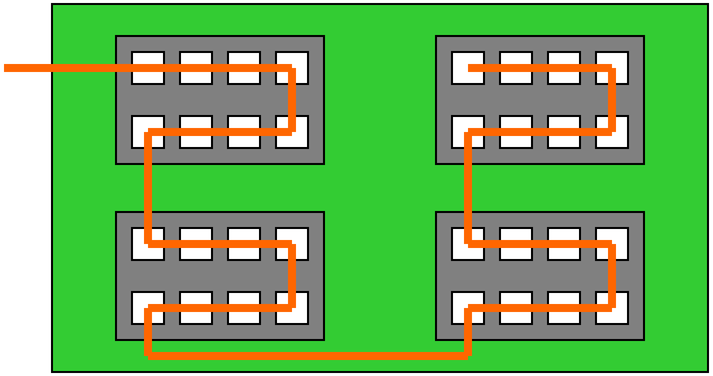


State and timing modes

- Timing mode: several samples per period
  - For glitch oriented debugging
  - more memory
- State mode: one sample per period
  - For sequential oriented problem

Boundary scan

- Simplifies testing of multiple chips on a board.
  - Registers on pins can be configured as a scan chain.
  - Used for debuggers, in-circuit emulators.



How to exercise code

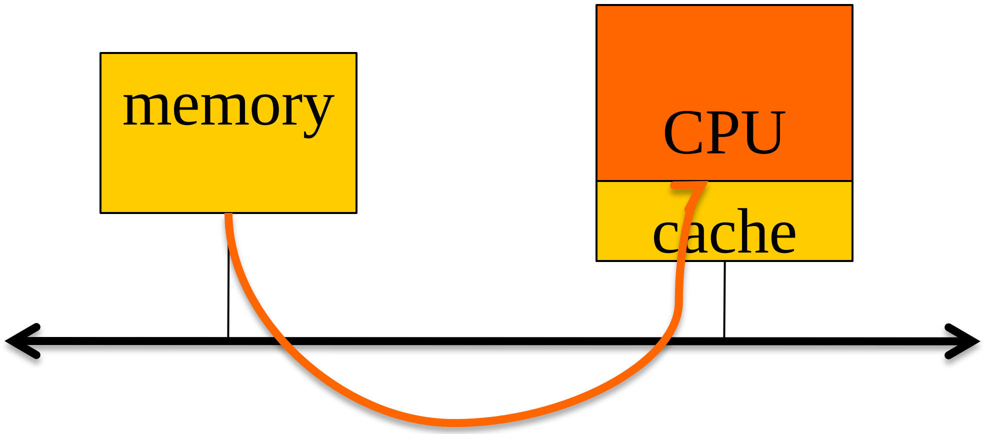
- Run on host system.
- Run on target system.
- Run in instruction-level simulator.
- Run on cycle-accurate simulator.
- Run in hardware/software co-simulation environment.

Debugging real-time code

- Bugs in drivers can cause non-deterministic behavior in the foreground problem.
- Bugs may be timing-dependent.

4.7 System-level performance analysis

- Performance depends on all the elements of the system:
  - CPU.
  - Cache.
  - Bus.
  - Main memory.
  - I/O device.



Bandwidth as performance

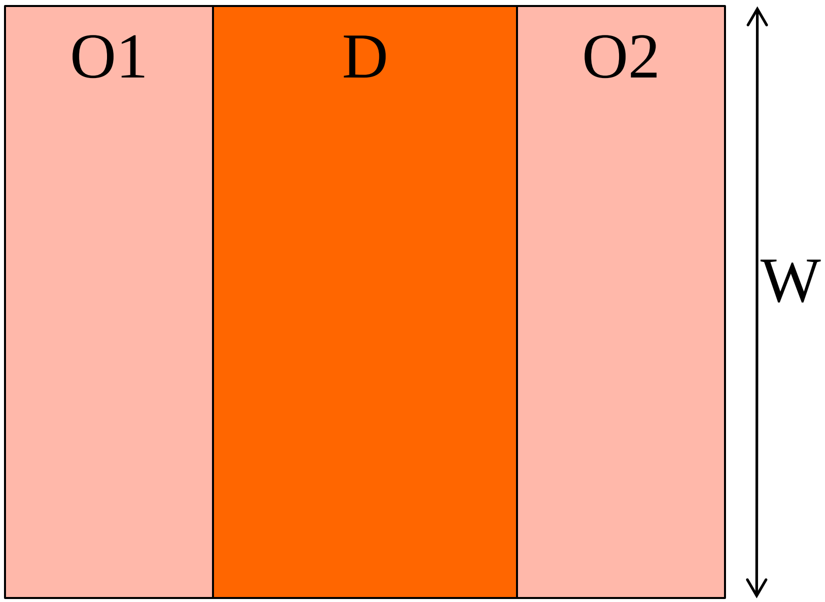
- Bandwidth applies to several components:
  - Memory.
  - Bus.
  - CPU fetches.
- Different parts of the system run at different clock rates.
- Different components may have different widths (bus, memory).

Bandwidth and data transfers

- Per video frame: 320 x 240 x 3 = 230,400 bytes.
  - Transfer in 1/30 sec.
- Transfer 1 byte/μsec, 0.23 sec per frame.
  - Too slow.
- Increase bandwidth:
  - Increase bus width.
  - Increase bus clock rate.

Bus bandwidth

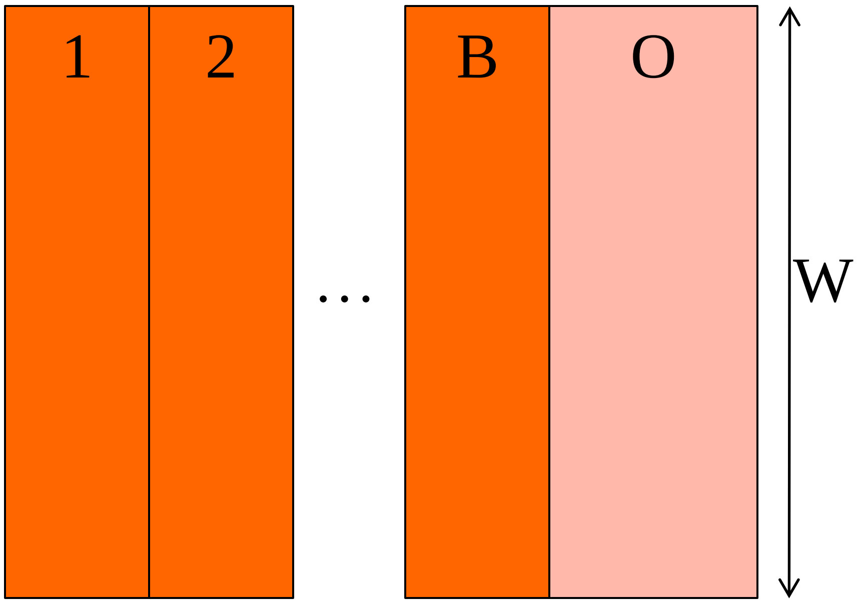
- T: # bus cycles.
- P: time/bus cycle.
- Total time for transfer:
  - t = TP.
- D: data payload length.
- O1 + O2 = overhead O.
  - Address, handshaking
- N bytes to be transferred
- Bus width: W bytes



$$T_{basic}(N) = (D+O)N/W$$

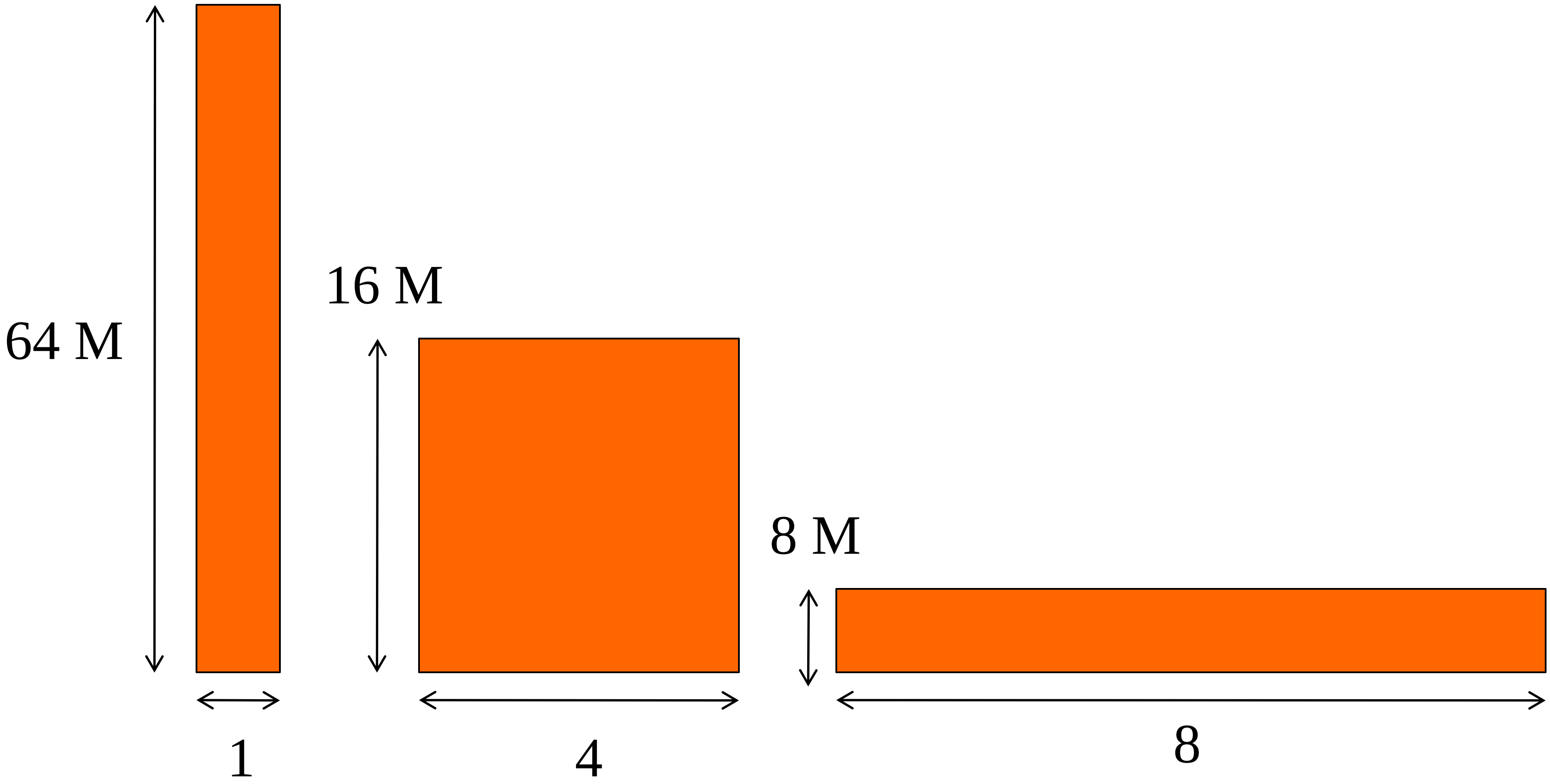
Bus burst transfer bandwidth

- T: # bus cycles.
- P: time/bus cycle.
- Total time for transfer:
  - t = TP.
- D: data payload length.
- O1 + O2 = overhead O.



$$T_{burst}(N) = (BD+O)N/(BW)$$

Memory aspect ratios

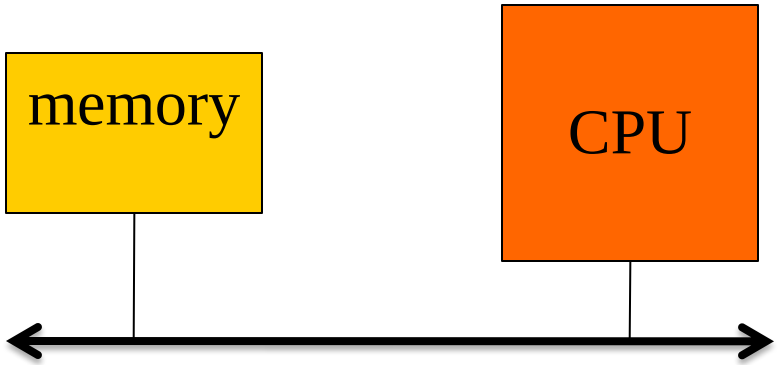


Memory access times

- Memory component access times comes from chip data sheet.
  - Page modes allow faster access for successive transfers on same page.
- What if data doesn't fit naturally into physical words:
- A pixel: RGB 24-bit
  - an access for 24-bit-wide memory
  - 3 accesses for 8-bit wide memory
  - how about 32-bit wide memory
    - waste one byte for each access
    - packing

Bus performance bottlenecks

- Transfer 320 x 240 video frame @ 30 frames/sec = 612,000 bytes/sec.
- Is performance bottleneck bus or memory?



Bus performance bottlenecks, cont'd.

- Bus: assume 1 MHz bus, D=1, O=3:
  - T_{basic} = (1+3)612,000/2 = 1,224,000 cycles = 1.224 sec.
- Memory: try burst mode B=4, width w=0.5. (assume 10MHz)
  - T_{mem} = (4*1+4)612,000/(4*0.5) = 2,448,000 cycles = 0.2448 sec.

Performance spreadsheet

bus					memory				
clock period		1.00E-06			clock period		1.00E-08		
W		2			W		0.5		
D		1			D		1		
O		3			O		4		
					B		4		
N		612000			N		612000		
T_basic		1224000			T_mem		2448000		
t		1.22E+00			t		2.45E-02		

4.7.2 Parallelism

- Speed things up by running several units at once.
- DMA provides parallelism if CPU doesn't need the bus:
  - DMA + bus.
  - CPU.