Embedded Systems Design: A Unified Hardware/Software Introduction

Memory

ESD_Cap5 ++

Outline

- Introduction
- Basic Concepts
- Write Ability and Storage Permanence
- Common Memory Types
- Composing Memory
- Memory Hierarchy and Cache
- Advanced RAM
- Scracth-Pad Memory

Introduction

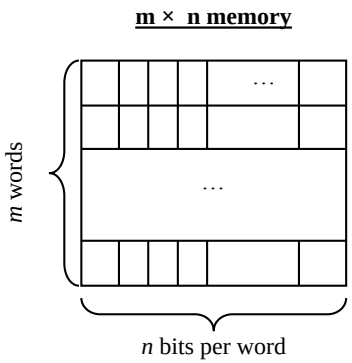
Introduction

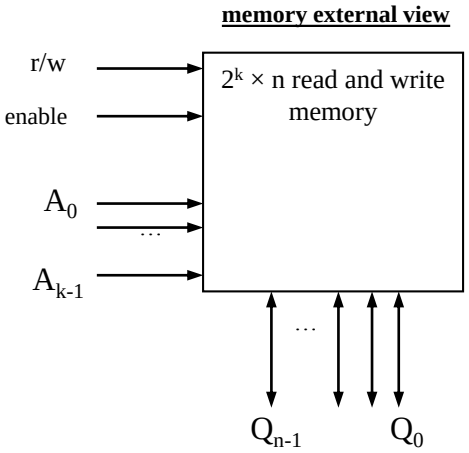
- Embedded system's functionality aspects
  - Processing
    - processors
    - transformation of data
  - Storage
    - memory
    - retention of data
  - Communication
    - buses
    - transfer of data

Basic Concepts

Memory: basic concepts

- Stores large number of bits
  - *m* x *n*: *m* words of *n* bits each
  - k = $\text{Log}_2(m)$ address input signals
  - or *m* = 2^k words
  - e.g., 4,096 x 8 memory:
    - 32,768 bits
    - 12 address input signals
    - 8 input/output data signals
- Memory access
  - r/w: selects read or write
  - enable: read or write only when asserted
  - multiport: multiple accesses to different locations simultaneously

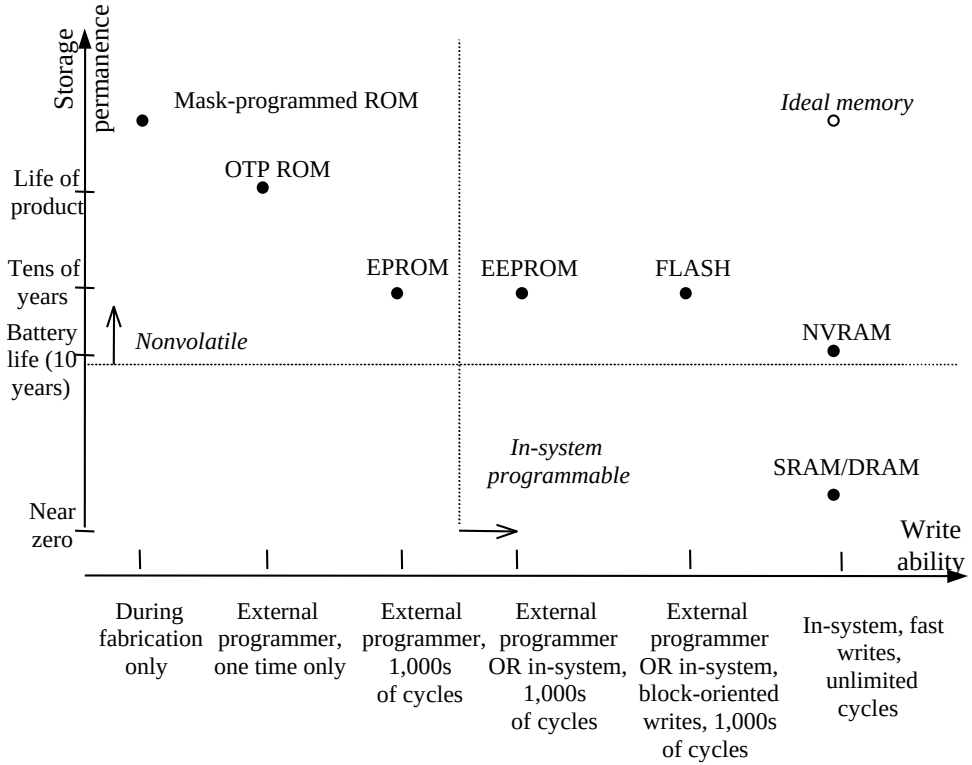




Write Ability and Storage Permanence

Write ability/ storage permanence

- Traditional ROM/RAM distinctions
  - ROM
    - read only, bits stored without power
  - RAM
    - read and write, lose stored bits without power
- Traditional distinctions blurred
  - Advanced ROMs can be written to
    - e.g., EEPROM
  - Advanced RAMs can hold bits without power
    - e.g., NVRAM
- Write ability
  - Manner and speed a memory can be written
- Storage permanence
  - ability of memory to hold stored bits after they are written



Write ability and storage permanence of memories, showing relative degrees along each axis (not to scale).

Write ability

- Ranges of write ability
  - High end
    - processor writes to memory simply and quickly
    - e.g., RAM
  - Middle range
    - processor writes to memory, but slower
    - e.g., FLASH, EEPROM
  - Lower range
    - special equipment, “programmer”, must be used to write to memory
    - e.g., EPROM, OTP ROM
  - Low end
    - bits stored only during fabrication
    - e.g., Mask-programmed ROM
- In-system programmable memory
  - Can be written to by a processor in the embedded system using the memory
  - Memories in high end and middle range of write ability

Storage permanence

- Range of storage permanence
  - High end
    - essentially never loses bits
    - e.g., mask-programmed ROM
  - Middle range
    - holds bits days, months, or years after memory's power source turned off
    - e.g., NVRAM
  - Lower range
    - holds bits as long as power supplied to memory
    - e.g., SRAM
  - Low end
    - begins to lose bits almost immediately after written
    - e.g., DRAM
- Nonvolatile memory
  - Holds bits after power is no longer supplied
  - High end and middle range of storage permanence

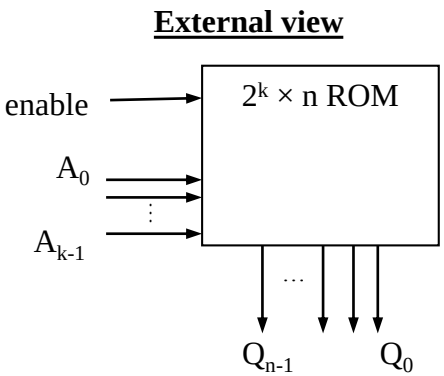
Common Memory Types

Common memory types

- ROM
  - Mask-programmed
  - OTP
  - EPROM
  - EEPROM
  - Flash
- RAM
  - Basic
  - Variations
- The future?
  - PCM

ROM: "Read-Only" Memory

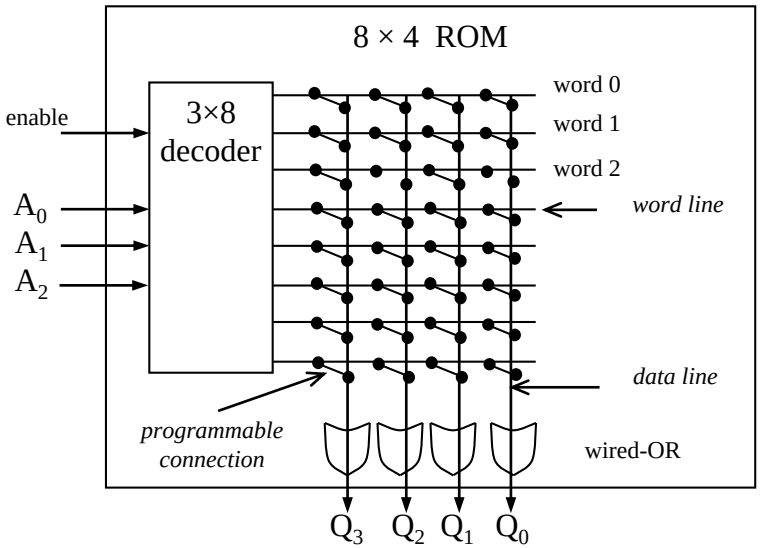
- Nonvolatile memory
- Can be read from but not written to, by a processor in an embedded system
- Traditionally written to, "programmed", before inserting to embedded system
- Uses
  - Store software program for general-purpose or application-specific processor
    - program instructions can be one or more ROM words
  - Store constant data needed by system
  - Implement combinational circuit



Example: 8 x 4 ROM

- Horizontal lines = words
- Vertical lines = data
- Lines connected only at circles
- Decoder sets word 2's line to 1 if address input is 010
- Data lines Q_3 and Q_1 are set to 1 because there is a "programmed" connection with word 2's line
- Word 2 is not connected with data lines Q_2 and Q_0
- Output is 1010

Internal view



Mask-programmed ROM

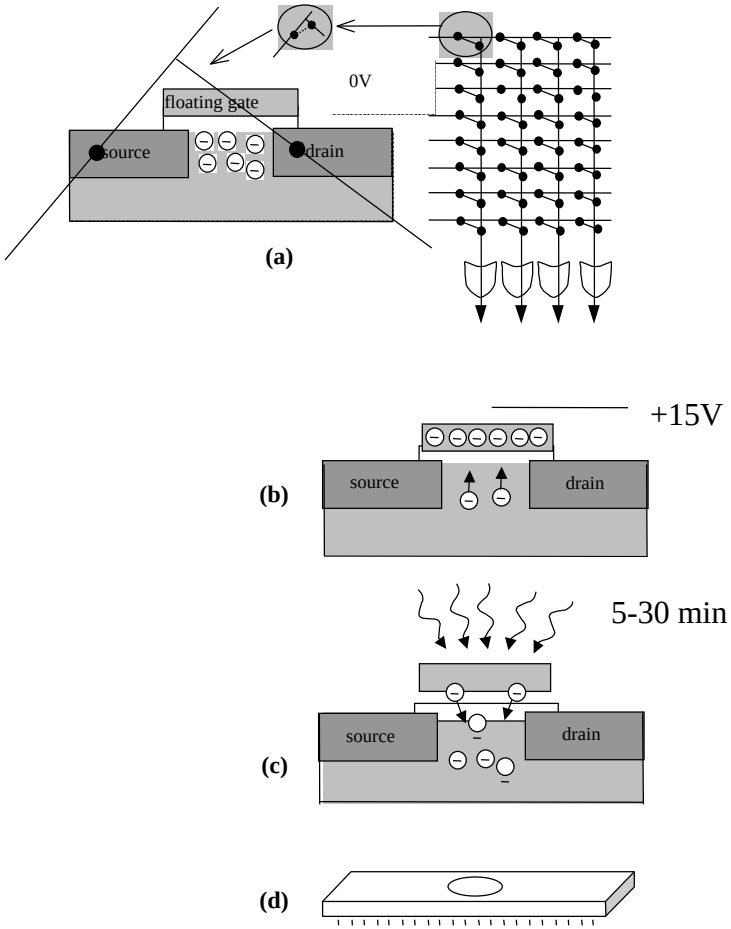
- Connections “programmed” at fabrication
  - set of masks
- Lowest write ability
  - only once
- Highest storage permanence
  - bits never change unless damaged
- Typically used for final design of high-volume systems
  - spread out NRE cost for a low unit cost

OTP ROM: One-time programmable ROM

- Connections "programmed" after manufacture by user
  - user provides file of desired contents of ROM
  - file input to machine called ROM programmer
  - each programmable connection is a fuse/antifuse
- Very low write ability
  - typically written only once and requires ROM programmer device
- Very high storage permanence
  - bits don't change unless reconnected to programmer and more fuses blown
- Commonly used in final products
  - cheaper, harder to inadvertently modify

EPROM: Erasable programmable ROM

- **Programmable component is a MOS transistor**
  - Transistor has “floating” gate surrounded by an insulator
  - **(a)** Negative charges form a channel between source and drain storing a logic 1
  - **(b)** Large positive voltage at gate causes negative charges to move out of channel and get trapped in floating gate storing a logic 0
  - **(c)** (Erase) Shining UV rays on surface of floating-gate causes negative charges to return to channel from floating gate restoring the logic 1
  - **(d)** An EPROM package showing quartz window through which UV light can pass
- **Better write ability**
  - can be erased and reprogrammed thousands of times
- **Reduced storage permanence**
  - program lasts about 10 years but is susceptible to radiation and electric noise
- **Typically used during design development**



EEPROM: Electrically erasable programmable ROM

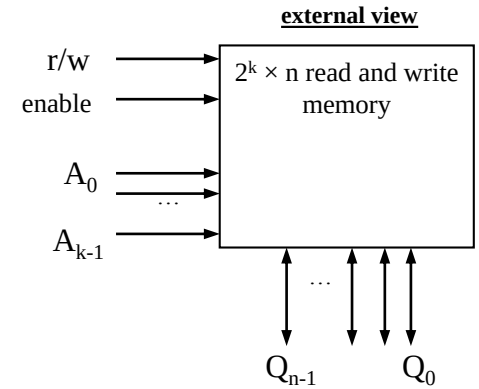
- Programmed and erased electronically
  - typically by using higher than normal voltage
  - can program and erase individual words
- Better write ability
  - can be in-system programmable with built-in circuit to provide higher than normal voltage
    - built-in memory controller commonly used to hide details from memory user
  - writes very slow due to erasing and programming
    - "busy" pin indicates to processor EEPROM still writing
  - can be erased and programmed tens of thousands of times
- Similar storage permanence to EPROM (about 10 years)
- Far more convenient than EPROMs, but more expensive

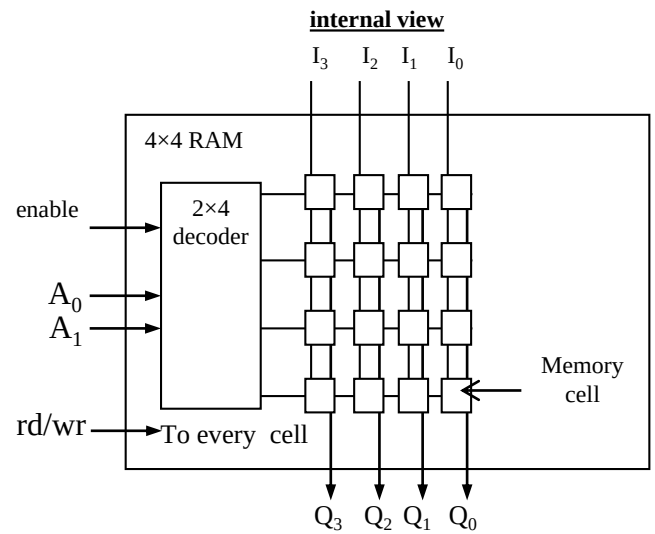
Flash Memory

- Extension of EEPROM
  - Same floating gate principle
  - Same write ability and storage permanence
- Fast erase
  - Large blocks of memory erased at once, rather than one word at a time
  - Blocks typically several thousand bytes large
- Writes to single words may be slower
  - Entire block must be read, word updated, then entire block written back
    - NAND (read block)/NOR (read byte)
- Used with embedded systems storing large data items in nonvolatile memory
  - e.g., digital cameras, TV set-top boxes, cell phones (NAND)
  - BIOS, Smarthphone OSs (NOR)

RAM: "Random-access" memory

- **Typically volatile memory**
  - bits are not held without power supply
- **Read and written to easily by embedded system during execution**
- **Internal structure more complex than ROM**
  - a word consists of several memory cells, each storing 1 bit
  - each input and output data line connects to each cell in its column
  - rd/wr connected to every cell
  - when row is enabled by decoder, each cell has logic that stores input data bit when rd/wr indicates write or outputs stored bit when rd/wr indicates read



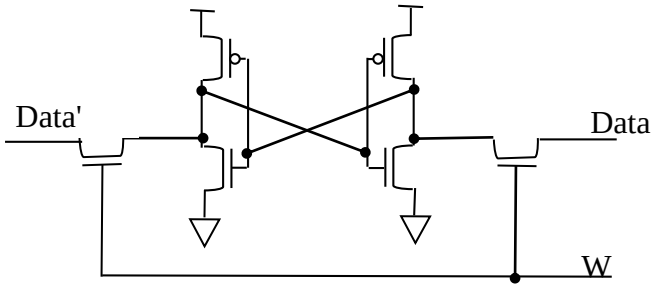


Basic types of RAM

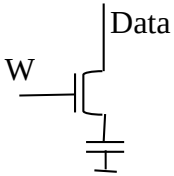
- SRAM: Static RAM
  - Memory cell uses flip-flop to store bit
  - Requires 6 transistors
  - Holds data as long as power supplied
- DRAM: Dynamic RAM
  - Memory cell uses MOS transistor and capacitor to store bit
  - More compact than SRAM
  - "Refresh" required due to capacitor leak
    - word's cells refreshed when read
  - Typical refresh rate 15.625 microsec.
  - Slower to access than SRAM

memory cell internals

SRAM



DRAM



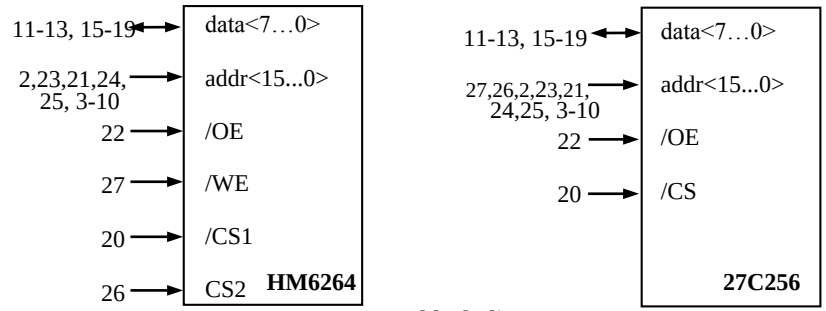
Ram variations

- PSRAM: Pseudo-static RAM
  - DRAM with built-in memory refresh controller
  - Popular low-cost high-density alternative to SRAM
- NVRAM: Nonvolatile RAM
  - Holds data after external power removed
  - Battery-backed RAM
    - SRAM with own permanently connected battery
    - writes as fast as reads
    - no limit on number of writes unlike nonvolatile ROM-based memory
  - SRAM with EEPROM or flash
    - stores complete RAM contents on EEPROM or flash before power turned off

Example: HM6264 & 27C256 RAM/ROM devices

- Low-cost low-capacity memory devices
- Commonly used in 8-bit microcontroller-based embedded systems
- First two numeric digits indicate device type
  - RAM: 62
  - ROM: 27
- Subsequent digits indicate capacity in kilobits

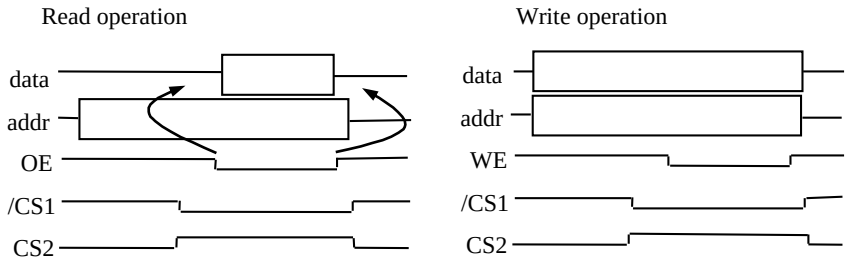
- Memory controllers?



block diagrams

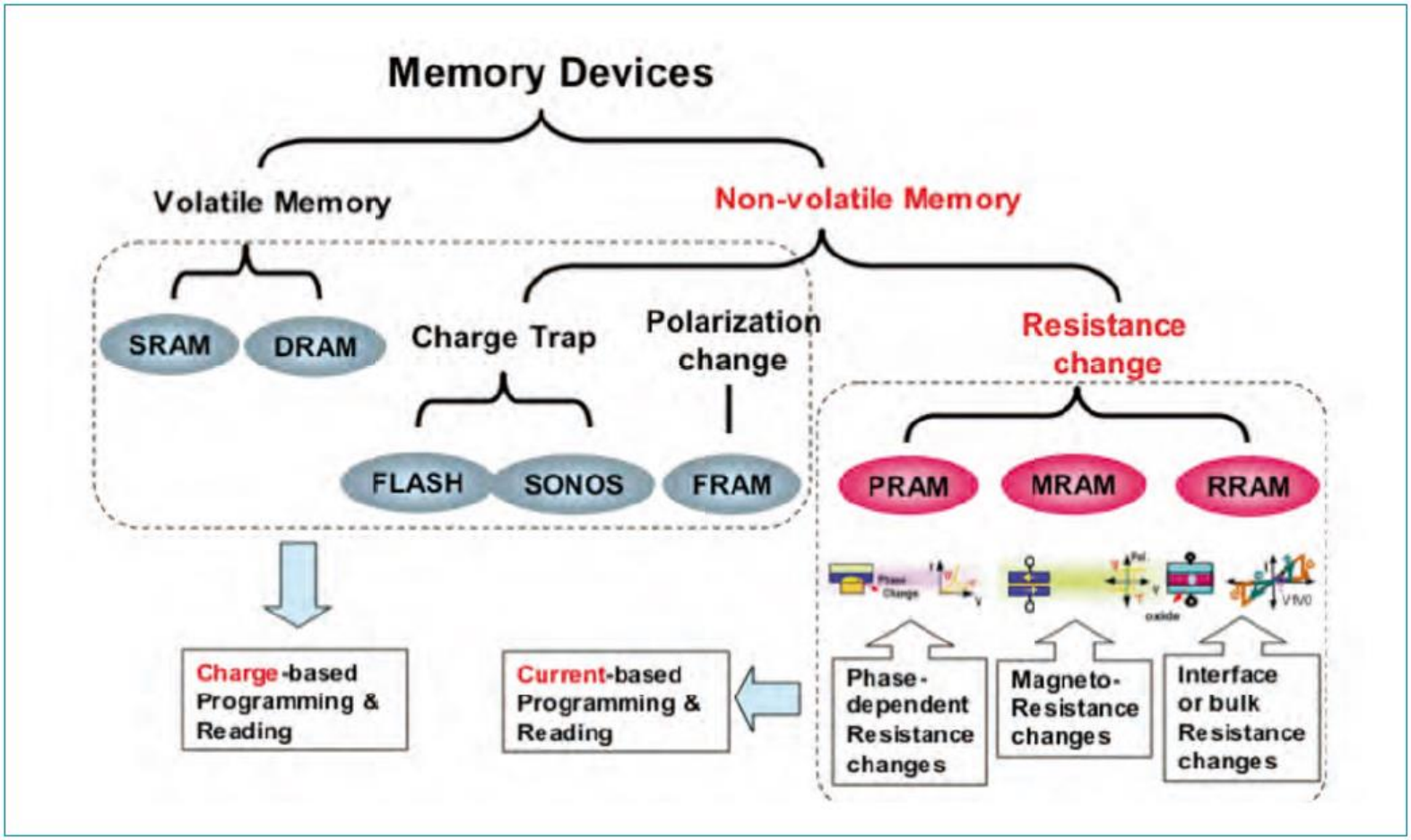
Device	Access Time (ns)	Standby Pwr. (mW)	Active Pwr. (mW)	Vcc Voltage (V)
HM6264	85-100	.01	15	5
27C256	90	.5	100	5

device characteristics



timing diagrams

The future?


Memory Devices
Volatile Memory
Non-volatile Memory
SRAM
DRAM
Charge Trap
Polarization change
Resistance change
FLASH
SONOS
FRAM
PRAM
MRAM
RRAM
Charge-based Programming & Reading
Current-based Programming & Reading
Phase-dependent Resistance changes
Magneto-Resistance changes
Interface or bulk Resistance changes

The future?

Phase Change Memory

- Overview

Features

- Scaling capability to <10nm
- Small cell size ($5.5F^2$)
- Bit alterability, Byte/Bit Write, NO erase required
- High endurance: 10^{6+} cycles
- High write throughput (5 ÷15+MB/sec)
- Very low read latency and high read throughput

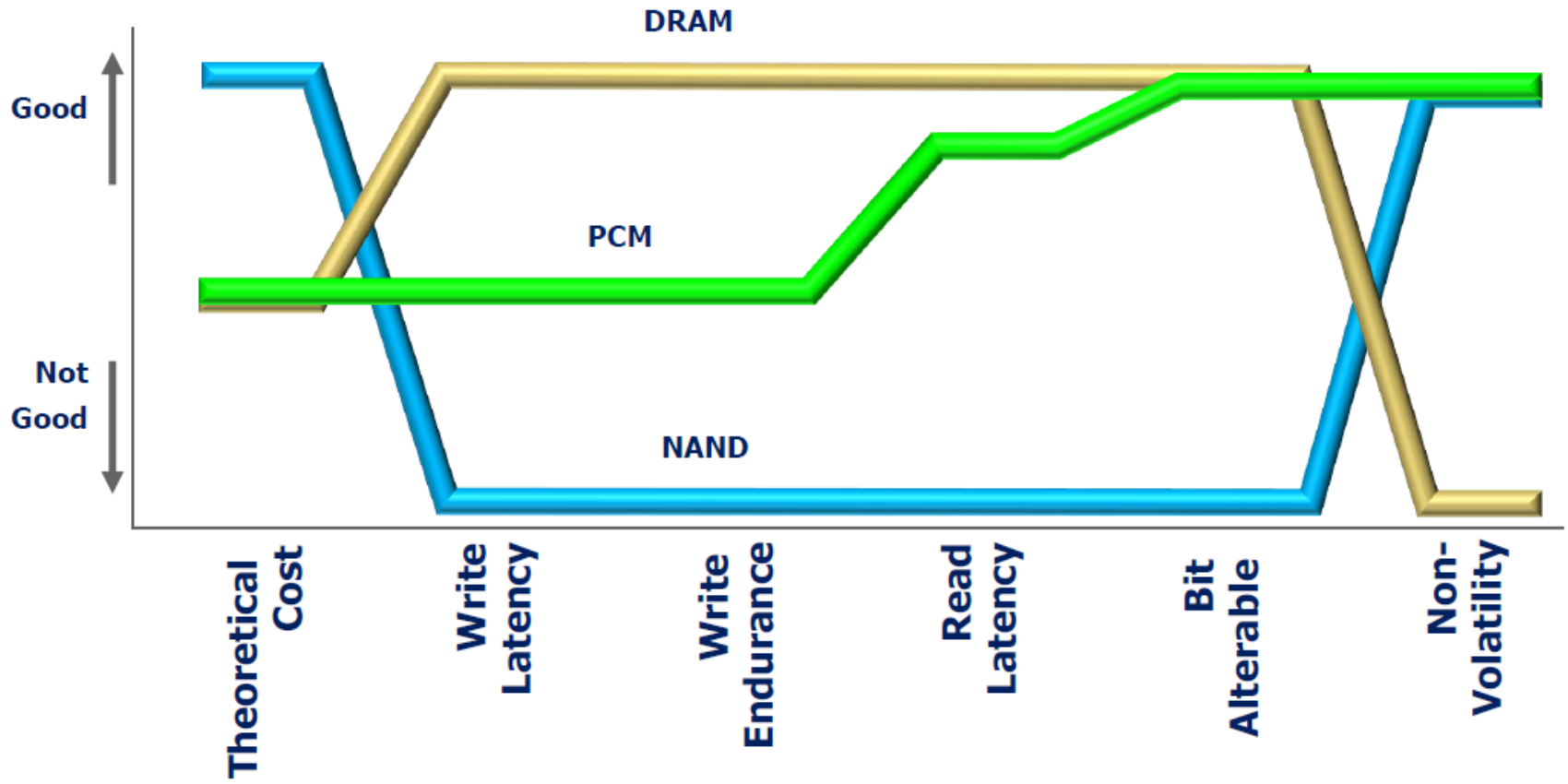
Benefits

- **Improve System performance**
  - Lower boot time
  - SW improvement using PCM Overwrite
- **Superior Quality**
  - Improved cycling performance
  - Better Reliability for code storage/execution
- **Save Battery Life**
  - Reduce overall power consumption in a system
- **Design optimization**
  - Lower BOM Cost by reducing required RAM density

The future?

Phase Change Memory

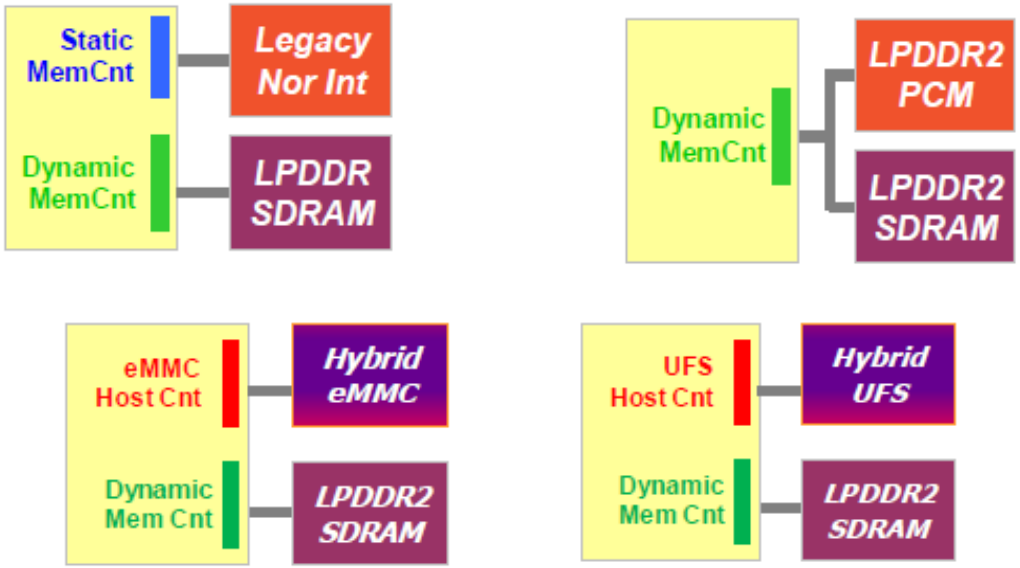
- Comparison



The future?

Phase Change Memory

- Possible use
  - EEPROM replacement
  - Hybrid architectures for mobile platforms

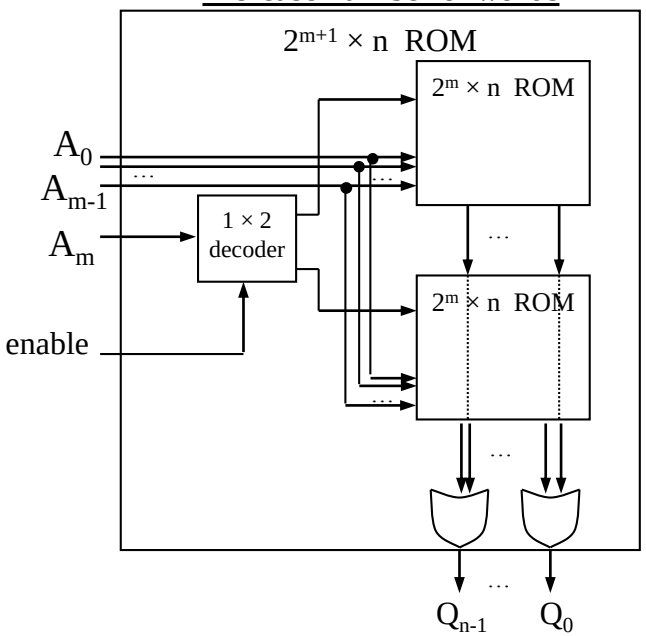


Composing Memory

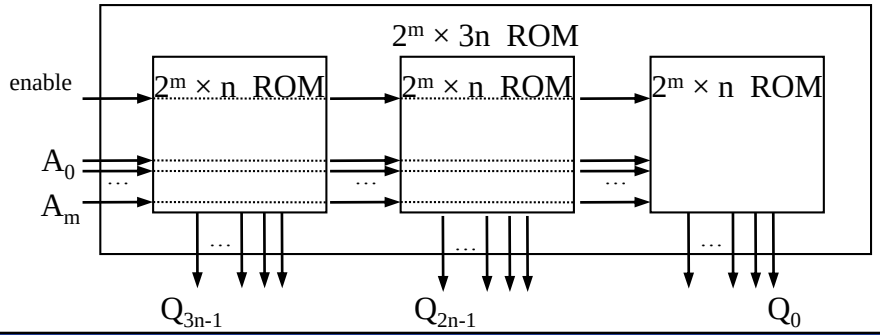
Composing memory

- Memory size needed often differs from size of readily available memories
- When available memory is larger, simply ignore unneeded high-order address bits and higher data lines
- When available memory is smaller, compose several smaller memories into one larger memory
  - Connect side-by-side to increase width of words
  - Connect top to bottom to increase number of words
    - added high-order address line selects smaller memory containing desired word using a decoder
  - Combine techniques to increase number and width of words

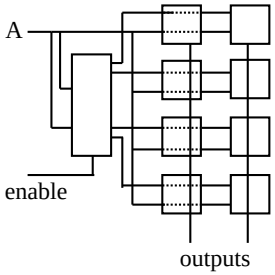
Increase number of words



Increase width of words



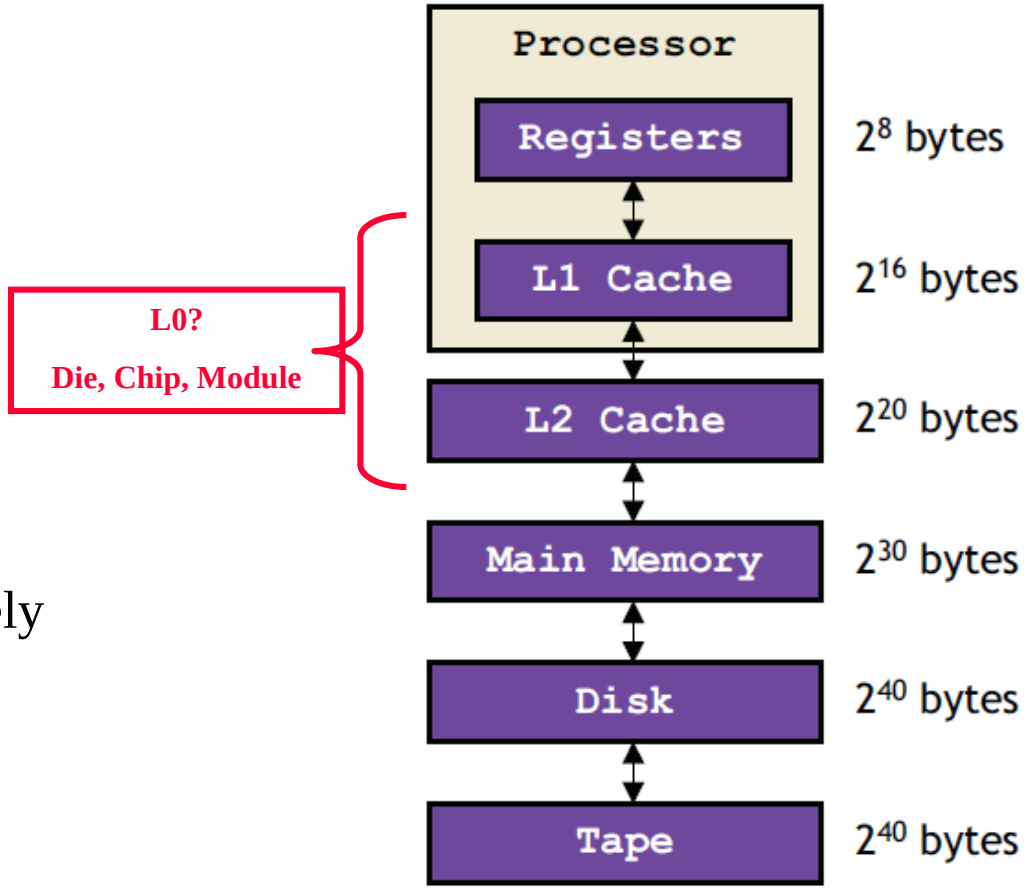
Increase number and width of words



Memory Hierarchy & Cache Memory

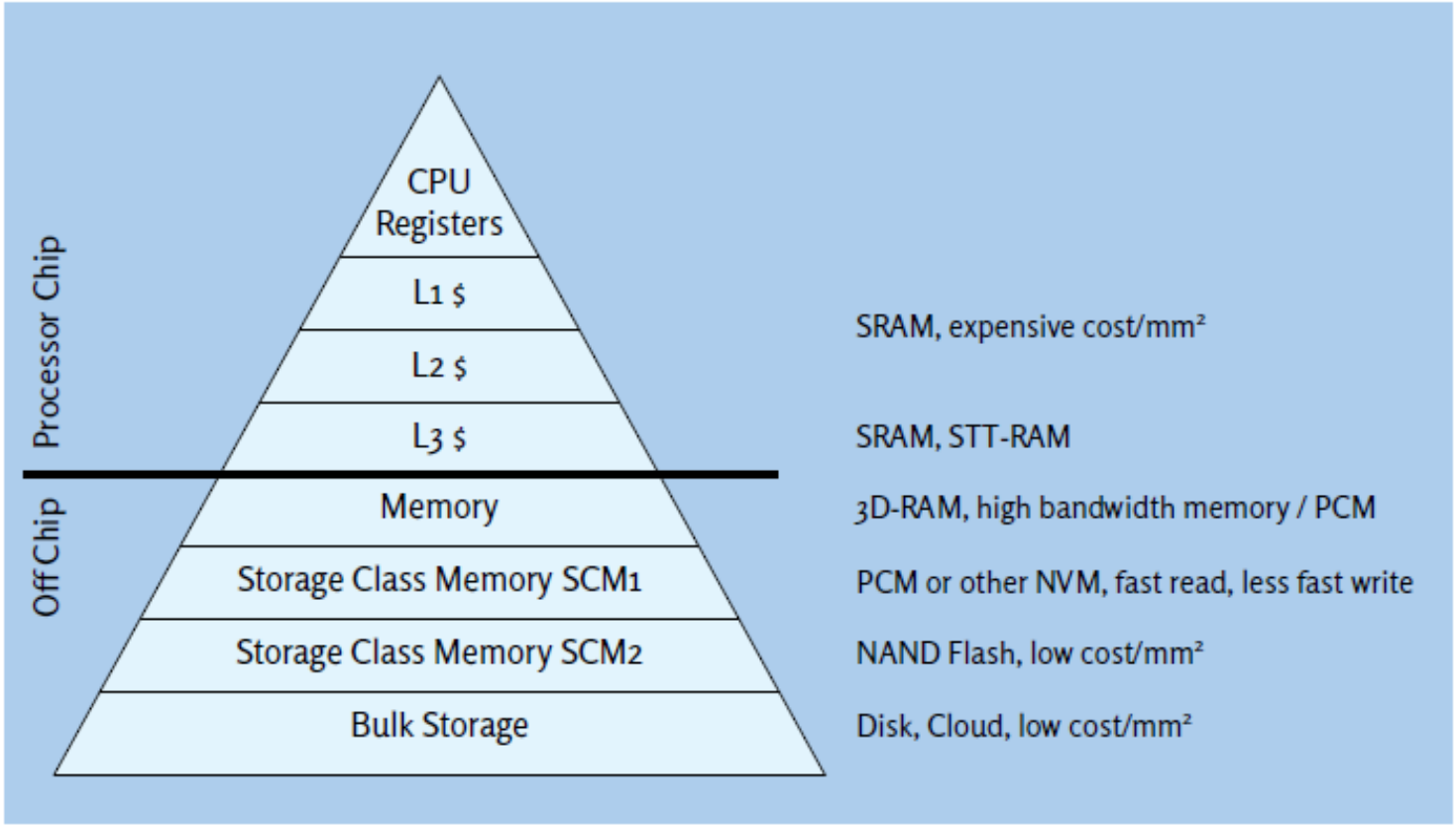
Memory hierarchy

- Want inexpensive, fast memory
- Main memory
  - Large, inexpensive, slow memory stores entire program and data
- Cache
  - Small, expensive, fast memory stores copy of likely accessed parts of larger memory
  - Can be multiple levels of cache



Memory hierarchy

- The future?



More info:

https://www.hipeac.net/vision/2019/

https://www.eurolab4hpc.eu/media/public/vision/D2-2--final-vision.pdf

Why cache memory?

- Empirical spatial/temporal locality principle
  - If a memory location with address *i* is used at a time *t*, it is very probable that the same location and the near ones will be used in a few time
    - This is valid both for instructions and data
    - The principle is statistically verified by the major part of the programs
- Empirical "90/10" law
  - 90% of the execution time is related to the 10% of code
    - The law is statistically verified by several programs

Cache

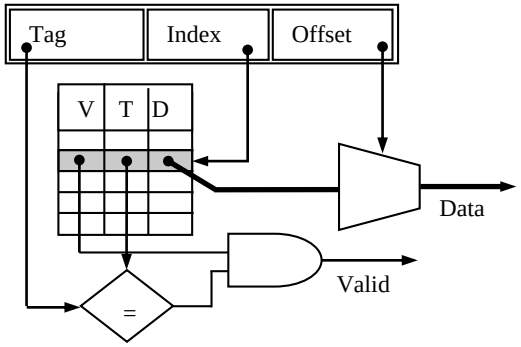
- **Usually designed with SRAM**
  - faster but more expensive than DRAM
- **Usually on same chip as processor**
  - space limited, so much smaller than off-chip main memory
  - faster access (1 cycle vs. several cycles for main memory)
- **Cache operation**
  - Request for main memory access (read or write)
  - First, check cache for copy
    - cache hit: copy is in cache, quick access
    - cache miss: copy not in cache, read address and possibly its neighbors into cache
- **Several cache design choices**
  - Separated or unified data/instructions cache
  - Cache mapping, replacement policies, and write techniques

Cache mapping

- Far fewer number of available cache addresses
  - Are address' contents in cache?
- Cache mapping used to assign main memory address to cache address and determine hit or miss
- Three basic techniques:
  - Direct mapping
  - Fully associative mapping
  - Set-associative mapping
- Caches partitioned into indivisible blocks or **lines** of adjacent memory addresses
  - usually 4 or 8 addresses per line

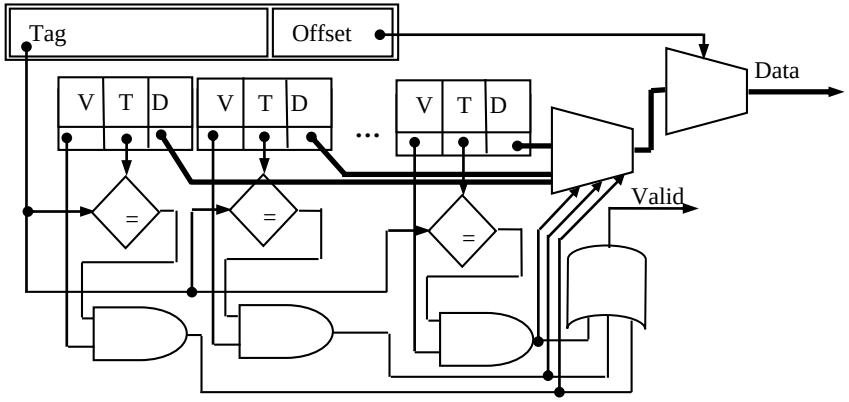
Direct mapping

- Main memory address divided into 2 fields
  - Index
    - cache address
    - number of bits determined by cache size
  - Tag
    - compared with tag stored in cache at address indicated by index
    - if tags match, check valid bit
- Valid bit
  - indicates whether data in slot has been loaded from memory
- Offset
  - used to find particular word in cache line



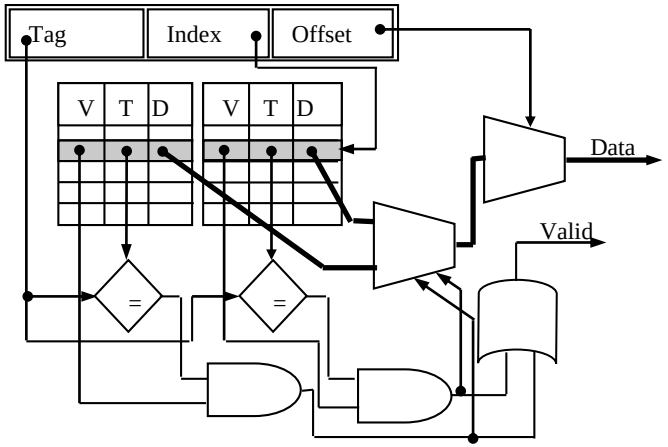
Fully associative mapping

- Complete main memory address stored in each cache address
- All addresses stored in cache simultaneously compared with desired address
- Valid bit and offset same as direct mapping



Set-associative mapping

- Compromise between direct mapping and fully associative mapping
- Index same as in direct mapping
- But, each cache address contains content and tags of 2 or more memory address locations
- Tags of that **set** simultaneously compared as in fully associative mapping
- Cache with set size N called **N-way** set-associative
  - 2-way, 4-way, 8-way are common



Cache-replacement policy

- Technique for choosing which block to replace
  - when fully associative cache is full
  - when set-associative cache's line is full
- Direct mapped cache has no choice
- Random
  - replace block chosen at random
- LRU: least-recently used
  - replace block not accessed for longest time
- FIFO: first-in-first-out
  - push block onto queue when accessed
  - choose block to replace by popping queue

Cache write techniques

- When written, data cache must update main memory
- Write-through
  - write to main memory whenever cache is written to
  - easiest to implement
  - processor must wait for slower main memory write
  - potential for unnecessary writes
- Write-back
  - main memory only written when “dirty” block replaced
  - extra dirty bit for each block set when cache block written to
  - reduces number of slow main memory writes

Cache impact on system performance

- Most important parameters in terms of performance
  - Total size of cache
    - Total number of data bytes cache can hold
    - Tag, valid and other house keeping bits not included in total
      - But they can be a lot!
  - Degree of associativity
  - Data block size
- Larger caches achieve lower miss rates but higher access cost
  - Improving cache hit rate without increasing size
    - Increase line size
    - Change set-associativity
- Big problem!
  - Cache coherence in multi processor/core systems!

Examples

	AMD Opteron	ARM 7D	Intel Xscale
CPU	CISC, 64 bit	RISC, 32 bit	RISC, 32 bit
Application	High-End Desktop	GPS, PDA, Games	Embedded
L1 organization	Separated	Unified	Separated
L1 dimension	64 KB, 64 KB	2 KB	32 KB, 32 KB
L1 associativity	2-way	4-way	32-way
L1 replacement	LRU	LRU	Roud-Robin
L1 writing	write-back	write-back	configurable
L2 organization	Unified		
L2 dimension	1 MB		
L2 associativity	16-way		
L2 replacement	Approximated LRU		
L2 writing	write-back		

Advanced RAM

Advanced RAM

- DRAMs commonly used as main memory in processor based embedded systems
  - high capacity, low cost
- Many variations of DRAMs proposed
  - need to keep pace with processor speeds
    - FPM DRAM: fast page mode DRAM
    - EDO DRAM: extended data out DRAM
    - SDRAM/ESDRAM: synchronous and enhanced synchronous DRAM
    - RDRAM: rambus DRAM
    - Double-Data Rate RAM (DDR, DDR2, DDR3, LP–DDR)
    - Fully-Buffered DRAM (FB-DRAM)
  - a key role is played by the memory controller

DRAM integration problem

- SRAM easily integrated on same chip as processor
- DRAM more difficult
  - Different "chip making process" between DRAM and conventional logic
    - But similar to CMOS sensors!
  - Goal of conventional logic (IC) designers
    - minimize parasitic capacitance to reduce signal propagation delays and power consumption
  - Goal of DRAM designers
    - create capacitor cells to retain stored information
  - Integration processes beginning to appear

Scratch-Pad Memory

Scratch-Pad Memory

- A combination of small memories containing frequently used data and instructions and a larger memory containing the remaining data and instructions is generally also more energy efficient than a single, large memory
  - Caches were initially introduced in order to provide good run-time efficiency but it is obvious that caches potentially also improve the energy-efficiency of a memory system
    - Accesses to caches are accesses to small memories and therefore may require less energy per access than large memories

Scratch-Pad Memory

- However, for caches it is required that the hardware checks whether or not the cache has a valid copy of the information associated with a certain address
  - This check involves comparing the tag fields of caches, containing a subset of the relevant address bits
    - Reading these tags requires additional energy
    - Also, the predictability of the real-time performance of caches is frequently low

- Alternatively…

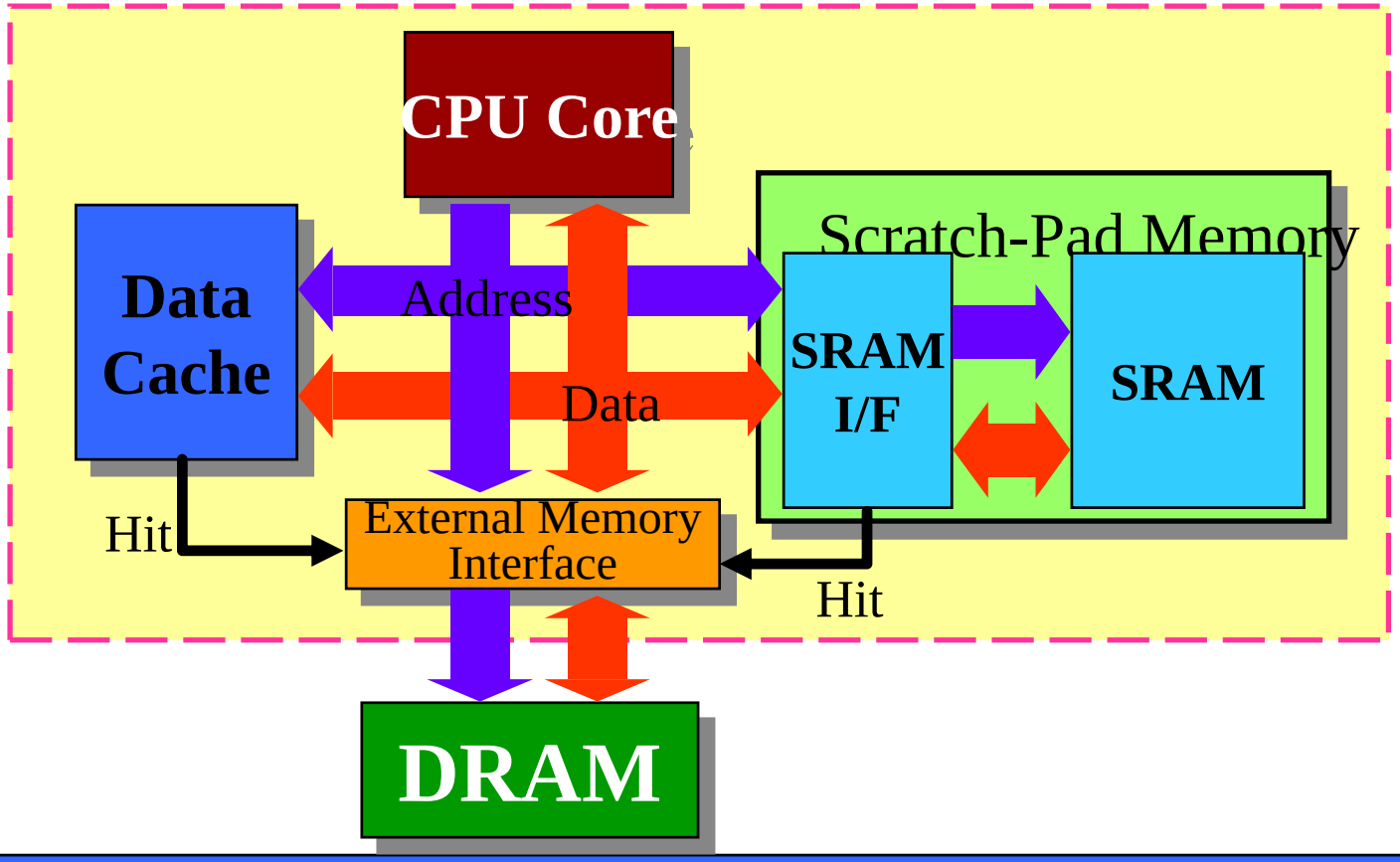
  …small memories can be mapped directly into the address space!

Scratch-Pad Memory

- Such memories are called **Scratch Pad Memories**
  - Frequently used variables and instructions should be allocated to that address space and no checking needs to be done in hardware
- As a result, the energy per access is reduced
  - SPMs can improve memory access times and predictability, if the compiler is in charge of keeping frequently used variables in the SPM. But…
    - **SPM is visible to the programmer/compiler while cache is not!**
    - **In a multi-tasking system the SPM could be erased (or managed in some way) at each context switch…**

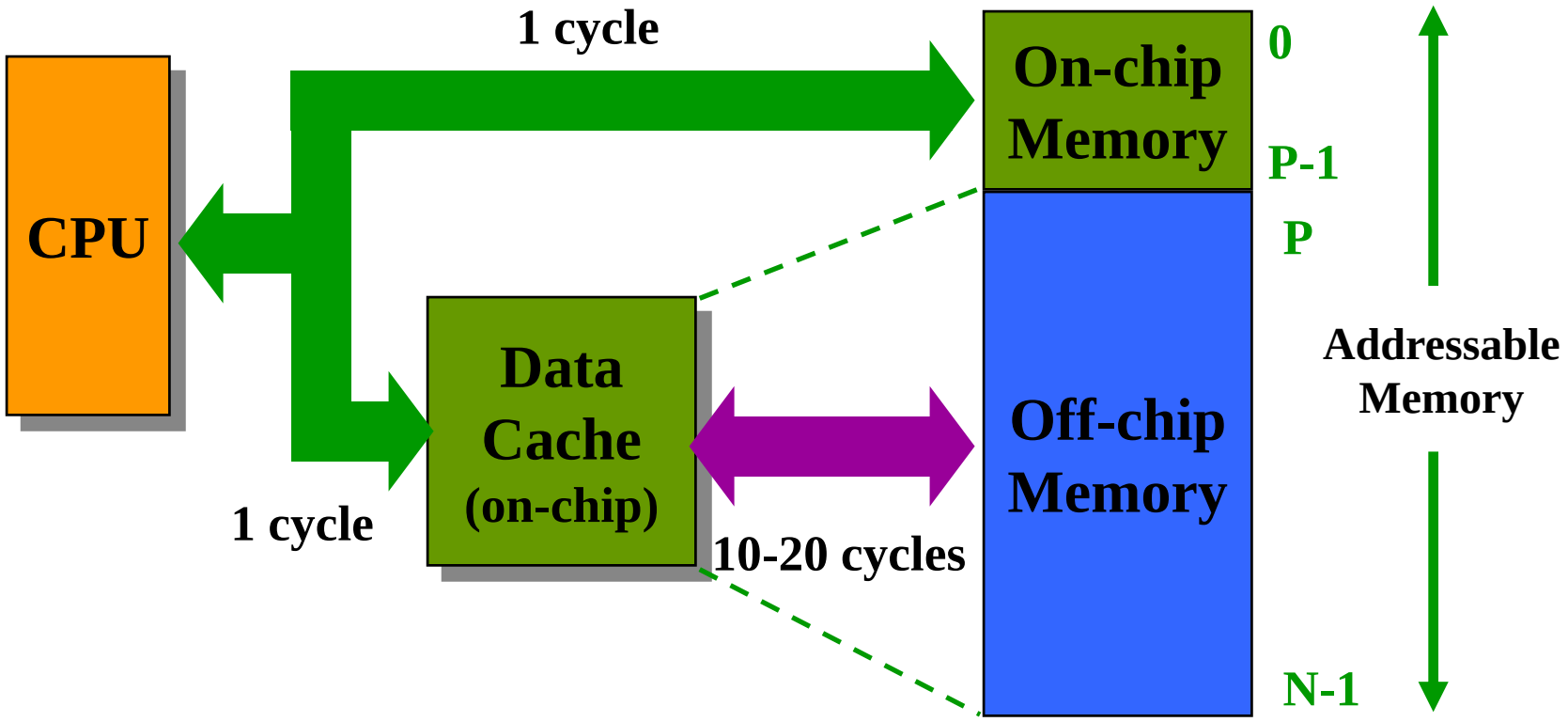
Scratch-Pad Memory

- Structural view



Scratch-Pad Memory

- Address space view



Scratch-Pad Memory

- Scratchpad usage in embedded systems could reduce the memory energy consumption, improve the performance and reduce the occupied area
  - **To maximize the benefit of using scratchpads, compiler has a primary role in the memory allocation…**

Cache

- Larger
- Subject to conflict, capacity and compulsary misses
- Unpredictable data access time
- Runtime assignment

Scratchpad

- Smaller
- Always result in a hit if requested within the data range
- Mapping by user or compiler directed