Computer Memory

Textbook: Chapter 1

ARM Cortex-M4 User Guide (Section 2.2 – Memory Model)
STM32F4xx Technical Reference Manual:
  Chapter 2  – Memory and Bus Architecture
  Chapter 3  – Flash Memory
  Chapter 36 – Flexible Static Memory Controller
Computer Memory Systems

- Memory system hierarchy
  - Disk, ROM, RAM, Cache
- Memory module (chip) organization
  - On-chip (address) decoder, cell array
- Memory system interfacing
  - Address decoding
  - Bus timing
- Direct memory access (DMA)
  - Transfer data directly between memory and I/O devices
  - Coordinated by a DMA controller
Computer Memory Hierarchy

Memory Content: $M_C \subseteq M_M \subseteq M_D$

Memory Parameters:
- Access Time: increase with distance from CPU
- Cost/Bit: decrease with distance from CPU
- Capacity: increase with distance from CPU
Semiconductor Memory

- **RAM (Random Access Memory)**
  - Constant access time, independent of location
  - A unique address for each location (generally a byte)
  - The address is decoded by one or more address decoders

- **RAM (Read/Write Memory) vs. ROM (Read Only Memory)**
  - **RAM**
    - User’s application programs and data
    - Information is lost when the power is off
  - **ROM**
    - Embedded system program code and operating system
    - Information is retained even without power
    - Each ROM cell is simpler than a RAM cell
Read-only memory types

- Mask-programmed ROM
  - Programmed at factory
- PROM (Programmable ROM)
  - Programmable once by users
  - Electric pulses selectively applied to “fuses”
- EPROM (Erasable PROM)
  - Repeatedly programmable/reprogrammable
  - Electric pulses for programming (seconds)
  - Ultraviolet light for erasing (minutes)
- EEPROM (Electrically Erasable PROM)
  - Electrically erasable at the single-byte level (\(\text{msec}\)) & programmable
- Flash EPROM
  - Electrically programmable (\(\mu\text{sec}\)) & erasable (block-by-block: \(\text{msec} \sim \text{sec}\))
  - Most common program memory in embedded applications
  - Widely used in digital cameras, multimedia players, smart phones, etc.

ROM devices are “non-volatile” – they retain information, even when not powered.
Read-write memory types

- **Static RAM (SRAM)**
  - Each cell is a flip-flop, storing 1-bit information
  - Information is retained as long as power is on (lost when power off)
  - Faster than DRAM
  - Requires a larger area per cell (more transistors) than DRAM

- **Dynamic RAM (DRAM)**
  - Each cell is a capacitor, which needs to be refreshed periodically to retain the 1-bit information
  - A refresh consists of reading followed by writing back
  - Refresh overhead
Memory organization (RAM)

- RAM Structure
  - Memory Cell

A byte consists of 8 memory cells, with common control signals, \( \text{Select} \) and \( \text{R/W} \), and 8 bidirectional data lines.

Some RAMs have separate Din and Dout

- With \( n \)-bit address, the memory system can contain up to \( 2^n \) bytes.
  
  An n-bit address is decoded by one or more address decoders to generate the control signal, \( \text{Select} \).
ROM/RAM device organization

Memory “organization” = $2^n \times d$
(from system designer’s perspective)

- Size.
  - $2^n$ addressable words
  - Address width = $n = r + c$
- Aspect ratio.
  - Data width $d$. 
Address Decoding

- Selecting a sub-space of memory address
- A simple example
  - Microprocessor with 5 address bits \((A_4 \ldots A_0)\) \(\Rightarrow 2^5 = 32\) bytes addressable
  - Memory chip: \(4 \times 8\) (4 bytes) \(\Rightarrow\) Decodes two address bits \((A_1 A_0)\)
  - Can address up to 8 chips (decode address bits \((A_4 A_3 A_2)\) for chip enable

![Diagram of on-chip decoder and memory array with address bits A4 to A0 and byte enable signals](image-url)
CE’ = chip enable: initiate memory access when active
OE’ = output enable: drive Data lines when active
WE’ = write enable: update SRAM contents with Data
(May have one R/W’ signal instead of OE’ and WE’)
Multi-byte data bus devices have a byte-enable signal for each byte.
Timing diagrams
Generic SRAM timing

- **CE’**
- **OE’**
- **WE’**
- **Adrs**
- **Data**

**Time**
- **Chip enabled**
- **Read**
- **Write**
- **From SRAM**
- **From CPU**
- **CPU captures Data**
- **SRAM captures Data**
Microprocessor buses

- Mechanism for communication with memories and I/O devices
  - signal wires with designated functions
  - protocol for data transfers
  - electrical parameters (voltage, current, capacitance, etc.)
  - physical design (connectors, cables, etc.)

- Clock for synchronization.
- R/W true when reading ** (R/W’ false when reading).
- Address = bundle of \( a \) address lines.
- Data = bundle of \( n \) data lines.
- “Data ready” => addressed device ready to complete the read/write

** Instead of R/W’ some CPUs have separate RD’ and WR’
Typical bus read and write timing
Bus wait state

Insert additional clock cycle(s) before completing read/write.
IS61LV51216-12T: 512K x 16 SRAM

2^{19} 1024K Bytes = 1M Byte

Byte Lane Select
- Upper byte D15-8
- Lower byte D7-0

Decoded A_{31-24}
Timing Parameters: Max data valid times following activation of Address, CE, OE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>-8 Min.</th>
<th>-8 Max.</th>
<th>-10 Min.</th>
<th>-10 Max.</th>
<th>-12 Min.</th>
<th>-12 Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>trc</td>
<td>Read Cycle Time</td>
<td>8</td>
<td></td>
<td>10</td>
<td></td>
<td>12</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>tAA</td>
<td>Address Access Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>toHA</td>
<td>Output Hold Time</td>
<td>3</td>
<td></td>
<td>3</td>
<td></td>
<td>3</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>tACE</td>
<td>CE Access Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>tDOE</td>
<td>OE Access Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ns</td>
</tr>
</tbody>
</table>
Flash memory devices

- Available in NAND or NOR structures
  - NOR flash system interface similar to SRAM (random access)
  - NAND flash system interface typically “serial” (indirect access)
- Read operations are the default, and similar to other memory devices
- Writing/erasing is initiated by writing “commands” to the Flash memory controller
  - Flash is programmed at system voltages.
  - Erasure time is long, and must be erased in blocks.

<table>
<thead>
<tr>
<th></th>
<th>SLC NAND Flash (x8)</th>
<th>MLC NAND Flash (x8)</th>
<th>MLC NOR Flash (x16)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Density</strong></td>
<td>512 Mbits(^1) – 4 Gbits(^2)</td>
<td>1Gbit to 16Gbit</td>
<td>16Mbit to 1Gbit</td>
</tr>
<tr>
<td><strong>Read Speed</strong></td>
<td>24 MB/s(^3)</td>
<td>18.6 MB/s</td>
<td>103MB/s</td>
</tr>
<tr>
<td><strong>Write Speed</strong></td>
<td>8.0 MB/s</td>
<td>2.4 MB/s</td>
<td>0.47 MB/s</td>
</tr>
<tr>
<td><strong>Erase Time</strong></td>
<td>2.0 mSec</td>
<td>2.0mSec</td>
<td>900mSec</td>
</tr>
<tr>
<td><strong>Interface</strong></td>
<td>I/O – indirect access</td>
<td>I/O – indirect access</td>
<td>Random access</td>
</tr>
<tr>
<td><strong>Application</strong></td>
<td>Program/Data mass storage</td>
<td>Program/Data mass storage</td>
<td>eXecuteInPlace</td>
</tr>
</tbody>
</table>
Ex: SST39VF1601- 1M x 16 NOR Flash
SST39VF3201=2M x 16 (4Mbyte: $2^{22}$) / SST39VF3201=4M x 16 (8Mbyte: $2^{23}$)

- Byte lane selects NBL[1:0] not used: all operations are “words”
- SST39VF3201 uses A[21..1], SST39VF6401 uses A[22..1]

2Mbyte ($2^{21}$) used of 64Mbyte ($2^{26}$) addr. space

Therefore, 5 address bits not decoded: A[25..21]**

A[0] is part of NBL[1:0]*

---

* Byte lane select not used
** Additional address bit
SST39VF1601 characteristics

- Organized as 1M x 16
  - 2K word sectors, 32K word blocks

- Performance:
  - Read access time = 70ns or 90ns
  - Word program time = 7us
  - Sector/block erase time = 18ms
  - Chip erase time = 40ms

- Check status of write/erase operation via read
  - DQ7 = complement of written value until write complete
  - DQ7=0 during erase, DQ7=1 when erase done
## SST39VF1601 command sequences
(assert WE# and CE# to write commands)

<table>
<thead>
<tr>
<th>Command Sequence</th>
<th>1st Bus Write Cycle</th>
<th>2nd Bus Write Cycle</th>
<th>3rd Bus Write Cycle</th>
<th>4th Bus Write Cycle</th>
<th>5th Bus Write Cycle</th>
<th>6th Bus Write Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Addr¹ Data²</td>
<td>Addr¹ Data²</td>
<td>Addr¹ Data²</td>
<td>Addr¹ Data²</td>
<td>Addr¹ Data²</td>
<td>Addr¹ Data²</td>
</tr>
<tr>
<td>Word-Program</td>
<td>5555H AAH</td>
<td>2AAAH 55H</td>
<td>5555H A0H</td>
<td>WA³ Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sector-Erase</td>
<td>5555H AAH</td>
<td>2AAAH 55H</td>
<td>5555H 80H</td>
<td>5555H AAH</td>
<td>2AAAH 55H</td>
<td>SA₄ 30H</td>
</tr>
<tr>
<td>Block-Erase</td>
<td>5555H AAH</td>
<td>2AAAH 55H</td>
<td>5555H 80H</td>
<td>5555H AAH</td>
<td>2AAAH 55H</td>
<td>BA₄ 50H</td>
</tr>
<tr>
<td>Chip-Erase</td>
<td>5555H AAH</td>
<td>2AAAH 55H</td>
<td>5555H 80H</td>
<td>5555H AAH</td>
<td>2AAAH 55H</td>
<td>5555H 10H</td>
</tr>
<tr>
<td>Erase-Suspend</td>
<td>XXXXH B0H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Erase-Resume</td>
<td>XXXXH 30H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Query Sec ID⁵</td>
<td>5555H AAH</td>
<td>2AAAH 55H</td>
<td>5555H 88H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Security ID</td>
<td>5555H AAH</td>
<td>2AAAH 55H</td>
<td>5555H A5H</td>
<td>WA⁶ Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Word-Program</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Security ID</td>
<td>5555H AAH</td>
<td>2AAAH 55H</td>
<td>5555H 85H</td>
<td>XXH⁶ 0000H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Program Lock-Out</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software ID Entry⁷,⁸</td>
<td>5555H AAH</td>
<td>2AAAH 55H</td>
<td>5555H 90H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CFI Query Entry</td>
<td>5555H AAH</td>
<td>2AAAH 55H</td>
<td>5555H 98H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software ID Exit⁹,¹⁰ /CFI Exit</td>
<td>5555H AAH</td>
<td>2AAAH 55H</td>
<td>5555H F0H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/Sec ID Exit</td>
<td>XXH F0H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Generic DRAM device

RAS’ = Row Address Strobe: row# on Address inputs
CAS’ = Column Address Strobe: column# on Address inputs
Generic DRAM timing

CE’
R/W’
RAS’
CAS’
Adrs
Data

Row adr
Col adr
data

Time
Page mode access

CE’
R/W’
RAS’
CAS’
Adrs
Data

time

row adrs

col adrs

col adrs

col adrs

data

data

data
Bus burst read (if supported)
Dynamic RAM refresh

- Value decays in approx. 1-4 ms.
- Refresh value by reading it
  - Read row of bits and then copy back
  - Can’t access memory during refresh.
- RAS-only refresh
- CAS-before-RAS refresh.
- Hidden refresh.

4 Mbyte DRAM: Refresh every 4 msec

Organized as **2048 rows** x 2048 columns \( \rightarrow \) 2048 refreshes

Assume 1 row refresh \( \rightarrow \) 80 nsec

\[
\frac{2048 \times 80 \times 10^{-9}}{4 \times 10^{-3}} \approx 0.041 \quad \rightarrow \quad 4.1\% \text{ of time spent refreshing}
\]
Other DRAM forms

- Extended data out (EDO): improved page mode access.
- Synchronous DRAM: clocked access for pipelining.
- Double Data Rate (DDR) – transfer on both edges of clock
  - DDR-1, DDR-2, DDR-3 support increasingly higher bandwidths
- Rambus: highly pipelined DRAM.
Cortex-M4 memory map

uC decodes $A_{31-28}$ for 0.5 GB blocks

Add external memory in this address range
STM32F407 Microcontroller

AHB 168MHz

APB 142MHz

APB 84MHz

External Bus
STM32 Flexible Static Memory Controller (FSMC)

- Control external memory on AHB bus in 4 - 256K banks
- Upper address bits decoded by the FSMC

<table>
<thead>
<tr>
<th>Address</th>
<th>Banks</th>
<th>Supported memory type</th>
</tr>
</thead>
<tbody>
<tr>
<td>6000 0000h</td>
<td>Bank 1</td>
<td>4 x 64 MB</td>
</tr>
<tr>
<td>6FF FFFFh</td>
<td></td>
<td>NOR / PSRAM</td>
</tr>
<tr>
<td>7000 0000h</td>
<td>Bank 2</td>
<td>4 x 64 MB</td>
</tr>
<tr>
<td>7FF FFFFh</td>
<td></td>
<td>NAND Flash</td>
</tr>
<tr>
<td>8000 0000h</td>
<td>Bank 3</td>
<td>4 x 64 MB</td>
</tr>
<tr>
<td>8FF FFFFh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9000 0000h</td>
<td>Bank 4</td>
<td>4 x 64 MB</td>
</tr>
<tr>
<td>9FF FFFFh</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Static memory-mapped devices:
  * SRAM
  * Pseudo-Static RAM
  * NOR flash

- 2 banks NAND flash

- 16-bit PC-Card devices

Bank 1 addresses:
A[31:28] = 0110
A[27:26] = 64MB bank select
A[25:0]   = 64MB bank offset
Example SRAM address decoding

SRAM/NE4 Addresses: [ 0x6C00 0000 … 0x6F00 0000]

Ext SRAM bank
2-to-4 decoder

Flash
Ext SRAM

Cortex peripherals

4Gbyte address space
4-to-16 decoder

A25-A0
CE
OE
WE

SRAM

2^{26} = 64 Mbytes

Within the microcontroller
“N” = “negative” (active low)

NE[4:1] = NOR/PSRAM enable
- NE[1]: A[27:26]=00

NL = address latch/advance
NBL = byte lane
CLK for sync. Burst

A[25:0] = Address bus
D[15:0] = Data bus**
NOE = output enable
NWE = write enable
NWAIT = wait request

** Data bus = 8 or 16 bits
FSMC “Mode 1” memory read

Other modes:

* Provide ADV (address latch/advance)

* Activate OE and WE only in DATAST

* Multiplex A/D bits 15-0

* Allow WAIT to extend DATAST

ADDSET/DATAST programmed in chip-select timing register (HCLK = AHB clock)
Example: 512K x 16 SRAM (1 Mbyte)

1Mbyte \(2^{20}\) used of this 64Mbyte \(2^{26}\) address space for NEx

Therefore, 6 address bits not decoded: A[25..20]

A[0] is part of NBL[1:0]

Microcontroller decodes upper address bits – ADDR[31..26] – for NEx
CPU Bus Types

- Synchronous vs. Asynchronous
  - Sync: all op’s synchronized to a clock
  - Async: devices signal each other to indicate start/stop of operations
    - May combine sync/async (80x86 “Ready” signal)
- Data transfer types:
  - Processor to/from memory
  - Processor to/from I/O device
  - I/O device to/from memory (DMA)
- Data bus types
  - Parallel (data bits transferred in parallel)
  - Serial (data bits transferred serially)
Hierarchical Bus Architecture
Example ARM System

High Performance ARM processor

AHB

APB Bridge

UART
Timer
Keypad
PIO

Low Power
Non-pipelined
Simple Interface

High Bandwidth External Memory Interface

High-bandwidth on-chip RAM

DMA Bus Master

High Performance
Pipelined
Burst Support
Multiple Bus Masters

Joe Bungo (ARM): CPU Design Concept to SoC
DMA

- Direct memory access (DMA) performs data transfers without executing instructions.
- CPU sets up transfer by programming the DMA controller.
- DMA engine fetches, writes.
- DMA controller is a separate unit – can become bus master.

![DMA Diagram]

**Diagram:**
- CPU
  - Bus request
  - Bus grant
- DMA controller
- Device
  - Clock
  - R/W
  - Address
  - Date ready
  - Data
- Memory
Bus mastership

- Bus master controls operations on the bus.
- By default, CPU is bus master and initiates transfers.
- Other devices may request bus mastership.
  - Separate set of handshaking lines.
  - CPU can’t use bus when it is not master.

- Bus mastership protocol:
  - Bus request – a device requests bus mastership from CPU
  - Bus grant – CPU relinquishes and grants mastership to device

- Situations for multiple bus masters:
  - DMA data transfers
  - Multiple CPUs with shared memory
    - One CPU might be graphics/network processor
DMA operation

- CPU configures DMA controller registers for:
  - peripheral address, memory start address, #xfers, direction (P->M or M->P)
- Peripheral issues DMA request to DMA controller.
- DMA controller takes bus mastership from CPU
- Once DMA is bus master, it transfers automatically.
  - Memory address incremented and count decremented for each transfer.
  - May run continuously until complete.
  - May use every n\textsuperscript{th} bus cycle.