**Precision GPS for Agriculture**

- **Regular GPS has an accuracy of perhaps 20 meters**
  - Works well if you can “snap” your position to the nearest road
  - Not good enough for precision agriculture
    - Want to be within an inch

- **Precision GPS uses augmentation**
  - Ground stations monitor received GPS signals and broadcast correction
  - WAAS only gives 1 meter accuracy
  - Private correction service can give 1 inch position accuracy
  - Subscription service (how do you charge?)

- **Precision navigation saves money**
  - Minimal overlap between passes
  - Adaptive fertilizer, pesticide, irrigation
  - Tractor auto-pilot for poor evening operation and to reduce operator fatigue

[Link: http://precisionpays.com/topics/precision-in-practice/]

[John Deere Inc]
Where Are We Now?

◆ Where we’ve been:
  • Lectures on software techniques

◆ Where we’re going today:
  • Memory bus (back to hardware for a lecture)

◆ Where we’re going next:
  • Economics / general optimization
  • Debug & Test
  • Serial ports

  • Exam #1
    – Scope of coverage is indicated on course web page

Preview

◆ Memory types
  • Different types of memory and general characteristics (RAM, PROM, …)
  • Interfacing to memory (rows vs. columns)

◆ CPU memory bus
  • Connects CPU to memory
  • Connects CPU to I/O
  • DMA – direct memory access
  • Practicalities (fanout, etc.)

◆ Quick review of memory protection (15-213 material)
Reminder – the memory bus on a microcontroller

- Used to transfer data to and from processor
  - Various types of memory
  - I/O data as well
  - Carries: address, data, and control signals

**Figure 1.1**
The basic components of a computer system include processor, memory, and I/O.

[Valvano]

Various Types of Memory

- RAM = Random Access Memory
- ROM = Read Only Memory

**FIGURE 1** Common memory types in embedded systems

[Bar01]
### Memory Array Geometry

- **2-D array composed of identical memory cells**
  - Address **decoder** selects one row
  - Sense amps detect and amplify memory cell value
  - Word select takes a subset of columns that have the byte/word of interest (*mux* = multiplexor)

- **Memory cell construction varies**
  - Speed vs. density
  - Volatile vs. non-volatile

### TABLE 1 Memory type characteristics

<table>
<thead>
<tr>
<th>Type</th>
<th>Volatile?</th>
<th>Writeable?</th>
<th>Erase Size</th>
<th>Erase Cycles</th>
<th>Cost/byte</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM</td>
<td>Yes</td>
<td>Yes</td>
<td>Byte</td>
<td>Unlimited</td>
<td>*Expensive</td>
<td>Fast</td>
</tr>
<tr>
<td>DRAM</td>
<td>Yes</td>
<td>Yes</td>
<td>Byte</td>
<td>Unlimited</td>
<td>Moderate</td>
<td>Moderate</td>
</tr>
<tr>
<td>Masked ROM</td>
<td>No</td>
<td>No</td>
<td>n/a</td>
<td>n/a</td>
<td>Inexpensive</td>
<td>Fast</td>
</tr>
<tr>
<td>PROM</td>
<td>No</td>
<td>Once, with a programmer</td>
<td>n/a</td>
<td>n/a</td>
<td>Moderate</td>
<td>Fast</td>
</tr>
<tr>
<td>EPROM</td>
<td>No</td>
<td>Yes, with a programmer</td>
<td>Entire chip</td>
<td>Limited (see specs)</td>
<td>Moderate</td>
<td>Fast</td>
</tr>
<tr>
<td>EEPROM</td>
<td>No</td>
<td>Yes</td>
<td>Byte</td>
<td>Limited (see specs)</td>
<td>*Expensive</td>
<td>Fast to read, slow to write</td>
</tr>
<tr>
<td>Flash</td>
<td>No</td>
<td>Yes</td>
<td>Sector</td>
<td>Limited (see specs)</td>
<td>Moderate</td>
<td>Fast to read, slow to write</td>
</tr>
<tr>
<td>NVRAM</td>
<td>No</td>
<td>Yes</td>
<td>Byte</td>
<td>Unlimited</td>
<td>*Expensive</td>
<td>Fast</td>
</tr>
</tbody>
</table>

[Barr01]
SRAM – Static RAM

- Uses “6T” cell design to reduce power consumption -- static CMOS
  - Used for on-chip RAM and small off-chip RAMs
  - Uses same process technology as CPU logic
  - Faster, less dense, more expensive than DRAM

IBM’s 6-Transistor Memory Cell

---

DRAM Cells

- DRAM optimized for small size, not speed
  - Uses different process technology than SRAMs or CPUs
    - Integrated DRAM + CPU chips can be inefficient to create – more process steps

Figure 1: IBM Trench Capacitor Memory Cell
Basics of DRAM Cells [18-240]

◆ The DRAM cell
  - Dynamic memory — the memory element is not active
  - Even with power on, the memory will eventually forget

◆ Memory mechanism is a capacitor
  - Charge is stored in it to represent a logic 1
  - No charge represents a logic 0
  - When you read it, you drain the capacitor — must rewrite it
  - Real life hits! The capacitor has a leak — the logic 1 eventually decays to a logic 0

Dram refresh [18-240]

◆ The charge exponentially decays
  - The capacitor must be refreshed (recharged), typically every 4 milliseconds
  - Every bit of the memory must be refreshed!
  - Typically one memory array row is refreshed at a time
Multiplexed Addresses [18-240]

- SRAM chips have a pin for every address line
  - Gives fast access, which is what SRAM is all about
  - For example, 64K bit x 1 chip has 16 address lines
  - For example, 256K bit x 8 (2 Mbit chip) has 18 address pins; 8 data pins

- DRAMS split the address in half (multiplex high and low bits)
  - The top 8 bits were the row address
  - Then bottom 8 bits selected one column (the column address)
  - This organization reduces the DRAM pin count – same pins for both Row & Col
    - 8 address bits can be sent at a time, in sequence
    - Only 8 pins and two strobe signals
    - vs. 16 pins and a strobe signal
    - Also ties in with the internal memory organization

Address

- row addr
- col addr
A 64K-bit DRAM Example [18-240]

- **Aspect ratio of chip**
  - Needs to be closer to square — here 256x256
  - Thus rows contain more than one “word”

- **External**
  - One bit in/out (“word”)
  - 16-bit address

- **Internal storage**
  - Top eight bits of address select the word
  - 256:1 mux (bottom 8 bits of address) selects bit to read/write
  - 256 bits refreshed at a time

A “word” is how many bits go in/out at a time (1 here)

A 64K-bit DRAM — Read [18-240]

- **Read access**
  - One of 256 rows selected
  - The read is destructive — it drains the capacitor.
  - The 256 bits are later written back into the memory to refresh it
**Timing Diagram Notation**

**Figure 9.18**
Nomenclature for drawing timing diagrams.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The input must be valid</td>
<td>The output will be valid</td>
</tr>
<tr>
<td></td>
<td>If the input were to fall</td>
<td>Then the output will fall</td>
</tr>
<tr>
<td></td>
<td>If the input were to rise</td>
<td>Then the output will rise</td>
</tr>
<tr>
<td></td>
<td>Don’t care, it will work regardless</td>
<td>Don’t know, the output value is indeterminate</td>
</tr>
<tr>
<td></td>
<td>Nonsense</td>
<td>High impedance, tristate, HiZ, Not driven, floating</td>
</tr>
</tbody>
</table>

[18-240]

**DRAM Read Cycle**

READ CYCLE

RAS# $V_{IH}$ –

CAS# $V_{IH}$ –

ADDR $V_{IH}$ –

WE# $V_{IL}$ –

DQ $V_{IH}$ –

OE# $V_{IL}$ –

VALID
DRAM Read Cycle [18-240]

- Sequence of events for reading a memory
  - Note – it is pretty complex
  - Usually “small” embedded systems avoid DRAM to keep things simple

Address

ras_l

cas_l

Dout

Load row-address register (latch), read selected row and store in row latch.

Load column-address register (latch), output-enable Dout.

Valid

Store row latch into selected row (like a refresh)

Output disable dout

we_l not asserted

ras_l, row address strobe

Fast Page Mode

(Micron MT4LC16M4A7)

FAST-PAGE-MODE READ CYCLE

RAS#

CAS#

ADDR

WE#

DQ

OE#
A 64K-bit DRAM — Write [18-240]

- **Write access**
  - First read 256 bits into latches (like previous read)
  - Change single bit in latches

- **Write 256 bits back into array**

![Diagram of DRAM Write Cycle]

**DRAM Write Cycle** [18-240]

- Load row-address register, read selected row and store in row latch
- Load column-address register, merge Din into selected column of column latches.
- Store row latches into selected row

Lots of details not shown!
**A 64K-bit DRAM — Refresh [18-240]**

- **Write access**
  - First read 256 bits into latches
  - Write 256 bits back into array
  - Then do next word

-Sometimes this is done by a controller on the chip, sometimes by an off-chip one.
**Refresh Cycle [18-240]**

- Each 4 ms, every word must be refreshed
  - Every ~15 µsec a 256-bit word is refreshed (4ms/256)
  - There is an on-chip controller to do this — it generates the row address and ras_l

- Notes
  - More happens in this memory than is easily accountable for with two edges (load register, load latches, write memory!)

**Non-Volatile RAM Technologies**

- **Sometimes memory has to survive a power outage**
  - On desktop machines this is (mostly) done by hard disk
  - Many embedded systems don’t have magnetic storage (cost, reliability, size)

- **Battery backed SRAM (fairly rare now that EEPROM is cheap)**
  - Mold a battery right into the SRAM plastic chip case
  - Just as fast & versatile as SRAM
  - Typically retains data for 4-7 years (usually limited by battery shelf life)
  - Cost includes both SRAM and a dedicated battery

- **FRAM**
  - Relatively new technology – in the marketplace, but not mainstream (yet)
  - Ferroelectric RAM
  - Unlimited read/write cycles
  - Intended as non-volatile drop-in replacement for SRAM (still expensive)
**ROM – Read Only Memory**

- **Masked ROM – pattern of bits built permanently into silicon**
  - Historically the most dense (least expensive) NV memory
  - BUT – need to change masks to change memory pattern ($$$$, lead time)
  - Every change means building completely new chips!
    - It also means throw the old chips away … they can’t be changed

- **Masked ROM seldom used in low-end embedded systems**
  - Too expensive to make new chips every time a change is needed
  - Takes too long (multiple weeks) to get the new chips

- **Corollary: many high volume embedded systems don’t use ASICs!**
  (Application-Specific ICs and semi-custom chips)
  - Design tools are too expensive and have too steep a learning curve
  - Changes come frequently, obsoleting inventory
  - ASICs usually only worthwhile for high-end embedded systems
    ($50 to $100 chips might be sensible ASICs – not $1 to $10 chips!)

---

**PROM Types**

- **PROM: Programmable Read-Only Memory**
  - Generic term for non-volatile memory that can be modified

- **OTPROM – “One Time” PROM**
  - Can only be programmed a single time (think “blowing fuses” to set bit values)
  - Holds data values indefinitely

- **EPROM – “Eraseable” PROM**
  - Entire chip erased at once using UV light through a window on chip
  - Mostly obsolete and replaced by flash memory

- **EEPROM – “Electrically Eraseable” PROM**
  - Erasure can be accomplished in-circuit under software control
  - Same general operation as flash memory EXCEPT…
    - …EEPROM can be erased/rewritten a byte at a time
      - Often have both flash (for bulk storage) and EEPROM (for byte-accessible writes) in same system

- **For all PROMS, ask about data retention**
  - Bits “rot” over time, 10 years for older technology; 100 years for newer technology
  - 10 year product life is often too short for embedded systems!
  - Also ask about wearout for values that are updated frequently
**Flash Memory Operation**

- Flash memory (and EEPROM, etc.) hold data on a floating transistor gate
  - Gate voltage turns transistor on or off for reading data
    - Usually, erasure results in all “1” values
  - **Erase/program cycles wear out the gate**
    - E.g., max 100K cycles for NOR flash
    - E.g., max 1M cycles for NAND flash
  - Data retention can be 100 years+
  - Cheaper than EEPROM; not byte modifiable

**Don’t Update EEPROM Every Minute!**

- 1M cycle EEPROM can only be updated every 5-10 minutes
  - Assuming 5-10 year product life
Flash Memory Update & Integrity

- Flash memory can be used as a “solid state hard drive”
  - Supports erase/reprogram of blocks of memory (not bytes as with EEPROM)
  - Technology used in USB “thumb drives” and solid state MP3 players
  - Hardware supports wear leveling and sector remapping to mitigate write hot-spots

- Flash/EEPROM update is complex
  - Requires significant time and repeated operations to set good bit values
  - Writing both flash and EEPROM is slow

- Common flash problem – “weak writes”
  - What happens if machine crashes during flash update?
  - Gate can be at a marginal voltage → unreliable data values
  - Usual solution: keep flag elsewhere in flash indicating write in progress
    - “System has started a flash update”
    - “System has completed a flash update”
    - If reboot finds “started” flag set, you know a weak write took place
  - Some flash-based file systems to have vulnerabilities in this area
    - Sometimes even the ones that say they are protected against power outages
    - If you use one, try about 100 power cycle tests to see if it suffers corruption

How Does Memory Connect To CPU?

- Processor bus (“memory bus”) connects CPU to memory and I/O
  - Data lines – actually transfers data
  - Address lines – feed memory address and I/O port number
  - Control lines – provides timing and control signals to direct transfers
  - Sometimes these lines are shared to reduce hardware costs

Figure 1.2
A memory read cycle copies data from RAM, ROM, or an input device into the processor.

[Valvano]
**Bus Transactions**

- **Bus serves multiple purposes**
  - Memory read and write
  - I/O read and write
  - Bulk data transfers (DMA – discussed later in lecture)

**Figure 1.3**
A memory write cycle copies data from the processor into RAM or an output device.

[Valvano]

---

**Address Decoding**

- **Every device on bus must recognize its own address**
  - Must decide which of multiple memory chips to activate
  - Each I/O port must decide if it is being addressed
  - High bits of addressed decoded to “select” device; low bits used within device

- **“Memory Mapped” I/O**
  - I/O devices and memory share same address space (e.g., Freescale)
  - Alternative: separate memory and I/O control lines (e.g., Intel)
  - What address does this decode?

[Valvano]
Read And Write Timing

- **Usually two edges involved**
  - One edge means “address valid now” – starts memory cycle
  - Second edge means “read or write data valid now” – ends memory cycle

**Figure 9.24**
Synchronized bus timing.

[Valvano]

MC9S12C32 Bus Timing

**Figure 9.40**
Simplified bus timing for the MC9S12C32 in expanded mode.

[Valvano]
Typical Bus Lines

- **Clock**
  - System clock so other devices don’t have to have their own oscillators
  - Drives bus timing for synchronous transfers

- **Address & Data**
  - Used for memory R/W, I/O, and DMA
  - Sometimes multiplexed, sometimes separate
  - Sometimes address is multiplexed (high/low) to make DRAM interface simpler

- **Control signals**
  - Read/write – which way is data moving?
  - Memory vs. I/O – if they are separate address spaces (Intel, not Freescale)
  - Byte vs. word – is it a whole word, or just a byte?
  - Device controls – interrupt request/grant; DMA request/grant; etc.

DMA – Direct Memory Access

- **For block memory transfers, can we keep data from the CPU bottleneck?**
  - In software, each byte read requires Device => CPU; CPU => Memory
  - Instead, directly transfer data from I/O device to memory (and reverse too)
  - Requires separate DMA controller hardware to perform transfer

*Figure 1.4*
A DMA read cycle copies data from RAM, ROM, or an input device into an output device.

[Valvano]
DMA Read Operation

CPU sets up DMA controller and I/O device before starting DMA

Where does the I/O address come from?
- For a CPU read from I/O device it would be the address on the bus
- But here, the address is the memory address

DMA Write Operation

DMA Controller signals CPU when DMA is done
- CPU keeps executing programs in parallel with DMA (they alternate bus access)

Does the memory “know” if it is doing DMA or CPU-directed accesses?
- Does the I/O device “know” if it is doing DMA or CPU-directed accesses?
Case Study: Original PC ISA Bus Pinout (PC-104)

```
CHIP SIDE        SOLDER SIDE
A1: IOCHK#        B1: GND
A2: SD7         B2: RESETDRV#
A3: SD6        B3: +5V
A4: SD5        B4: IRQ2
A5: SD4        B5: -5V
A6: SD3        B6: DRQ2
A7: SD2        B7: -12V
A8: SD1        B8: (unused)
A9: SD0        B9: +12V
A10: IOCHRDY    B10: GND
A11: AEN        B11: SMEMW#
A12: SA19       B12: SMEMR#
A13: SA18       B13: IOW#
A14: SA17       B14: IOR#
A15: SA16       B15: DACK3#
A16: SA15       B16: DRQ3
A17: SA14       B17: DACK1#
A18: SA13       B18: DRQ1
A19: SA12       B19: REFRESH=1=DACK#0
A20: SA11       B20: BCLK (4.77 MHz)
A21: SA10       B21: IRQ7
A22: SA9        B22: IRQ6
A23: SA8        B23: IRQ5
A24: SA7        B24: IRQ4
A25: SA6        B25: IRQ3
A26: SA5        B26: DACK2#
A27: SA4        B27: TC
A28: SA3        B28: BALE
A29: SA2        B29: +5
A30: SA1        B30: OSC (14.3 MHz)
A31: SA0        B31: GND
```

(Eggebrecht Figure 8-1)

ISA (PC/104) I/O Bus Read Operation

- Still used in embedded systems as the PC-104 bus standard
- Read from port
  - Note: Intel chips have separate I/O and Memory control lines (shared A & D)
ISA (PC/104) Direct Memory Access (DMA) Operation

- **Separate DMA controller**
  - Counter to track number of words remaining
  - "Cycle steals" bus bandwidth, transparent to programs

- **Data moves from memory to I/O**
  - I/O card asserts DRQx
  - I/O eventually receives DACKx from DMA controller
  - DMA controller asserts MEMR and IOW to accomplish a concurrent memory read and I/O write operation

---

**Practicalities – Fanout**

- **Sometimes a CPU has to drive many loads on a bus**
  - Multiple banks of memory
  - Multiple I/O devices

- **Fanout = number of loads being driven**
  - By address bus
  - By data bus
  - By control lines
  - Limited by drive current $I_{OH}$ and $I_{OL}$ (chip I/O speed rated at limited current)
  - Common limit for fanout is 5-10 loads

- **If fanout limit is exceeded need a buffer**
  - Especially common for address lines on memory wider than 8 bits
  - For example, 74LS245 is a bidirectional data buffer; 74LS244 is a unidirectional buffer
  - Buffer adds delay; slows down maximum system speed; increases fanout limit
  - Usually need to buffer DRAM memory address lines
    - Address lines drive *all* the chips (e.g., drives 8 chips for 4 chips x 32 bits x 2 banks)
    - Data lines only drive one chip in each bank (e.g., drives 2 chips for 2 banks)
Practicalities – Conflicting Bus Devices

- What happens if address decoding has a hardware bug?
  - One device might drive a bit to high
  - One device might drive that same bit to low
  - Is that OK?

Practicalities – Noise And Termination

- Real Hardware buses act as a transmission line
  - Signals take non-zero time to propagate
  - Signal waves reflect, superimpose, interfere, etc.
  - Noise issues are dominated by edge steepness – not just MHz!
    - Spectral components of edge are the culprit, not transitions per second

- Termination is used in physically large or complex buses
  - Put terminating resistors at one (or better, both) ends of bus lines
  - Especially if cabling or mechanical connectors are involved

[Ethirajan98]
Memory Address Space Extension

- **How does a 16-bit CPU address more than 64KB?**
  - Ever wonder how a 16-bit CPU can have 128KB of memory?
  - To do this, need to change “memory model”

- **Page register**
  - A register that holds top 8 or 16 bits of memory address
  - Memory address pre-pended with page register value
  - Might have “long” instructions that take full size memory address
  - Might have multiple page registers to allow copying between pages
  - If you have a problem with load and store instructions not working, check that you have the right memory model – we’re using the “tiny” memory model which ignores page register

- **Segment registers (e.g., 808x – original IBM PC CPU)**
  - A 24-bit or 32-bit base register that is added to each memory address
  - Flexible, but hardware addition adds latency to memory path
  - Might have multiple segment registers (e.g., program, stack, data)

- **Virtual memory …. (coming right up)**

Course CPU Uses A Page Register

- **Version 5 uses “far” addresses for subroutine calls**
  - Uses CALL instructions instead of JSR/BSR
  - Uses RTC instead of RTS

- **PPG = Program Page register**
  - 8 bit register that holds the top 8 bits of program address
  - Programs operate in a 64 K-byte fixed address space for programs
  - Switch between pages using CALL and RTC
  - CALL pushes PPG onto stack; RTC pulls PPG from stack
Virtual Memory (Remember This?)

- Two-level page table example showing address 0x12345678

Memory Protection

- Many small CPUs have unlimited access to memory
  - Any task can corrupt RAM
  - Fortunately, a wild pointer can’t corrupt Flash memory
    - Flash requires a complex procedure to modify

- Virtual memory provides excellent memory protection
  - Each task has its own distinct memory space starting at address 0
  - Only the OS can access other tasks’ memory spaces
  - Can enable sharing on a page by page basis

- Virtual memory hardware “lite” = MMU
  - Memory Management Unit
  - Big MMU might provide hardware support for virtual memory
  - But, a “small” MMU might just protect memory from other tasks
    - Usually a per-task base register that is added to memory addresses

- What if you don’t have an MMU?
  - Good practice is at least putting error code information of blocks of RAM values
  - If a wild pointer changes values, the error code has a chance to detect it
Lab Skills

- **Built a memory bus interface**
  - The module we use doesn’t have the real memory bus pinned out to proto-board
  - So we created software to emulate a simple memory bus for you

Review

- **Memory types**
  - Different types of memory and general characteristics
    - Should know names, general construction, characteristics of each
    - General idea behind NV memory (flash operation/EPPROM use)
  - Interfacing to memory (rows vs. columns)
    - Should know, e.g., what “RAS” and “CAS” do on DRAMs at level presented
    - Should understand how “read,” “write,” and “refresh” signals work

- **CPU memory bus**
  - General signals on a bus and what they are for
  - How to read a timing diagram
  - General bus operations – read, write, DMA, I/O
  - General practicalities (fanout, conflicts, noise, termination)
  - Memory address space protection

- **BUT we don’t expect you to memorize or do these things:**
  - Memorize timing numbers on specific buses
  - Draw bus timing diagrams or recall bus signal names from memory
  - Draw or interpret what each individual transistor does in a memory cell