Course Administration

- **Instructor:** Aleksandar Milenkovic
  - milenka@ece.uah.edu
  - www.ece.uah.edu/~milenka
  - EB 217-L
  - Mon. 5:30 PM – 6:30 PM,
  - Wed. 12:30 – 13:30 PM
- **URL:** http://www.ece.uah.edu/~milenka/cpe421-05F
- **TA:** Joel Wilder
- **Labs:** Lab #5 is on.
- **Text:** Microprocessor Systems Design: 68000 Hardware, Software, and Interfacing
- **Review:** M68K (Chapter 1; Chapter 2; Chapter 3), MSP430 (Introduction, Arch., Basic Clock System, WDT, Low Power Modes, Digital I/O), Timers, MSP 430 UART;
- **Today:** M68000 Hw (Chapter 4)
# THE 68000 CPU HARDWARE MODEL

Chapter 4

- 68000 interface
- Timing diagram
- Minimal configuration using the 68000

## Review: 68000 Interface

- M68000: 64 pins, arranged in 9 groups:
  - Address Bus: $A_{01} - A_{23}$
  - Data Bus: $D_{00} - D_{15}$
  - Asynchronous bus control:
    - AS*, R/W*, UDS*, LDS*, DTACK*, BERR*
  - Synchronous bus control:
    - E, VPA*, VMA*
  - Bus arbitration control:
    - BR*, BG*, BGACK*
  - Function code:
    - FCO, FC1, FC2
  - System control:
    - CLK, RESET*, HALT*
  - Interrupt control:
    - IPL0*, IPL1*, IPL2*
  - Miscellaneous:
    - Vcc(2), Gnd(2)

- **Legend:**
  - Type
    - Input
    - Output
    - Input/Output
Review: Memory and Peripheral Interface Pins

![Diagram of memory and peripheral interface pins]

Review: Special-Function Pins of the 68000

<table>
<thead>
<tr>
<th>Function Code Output</th>
<th>Processor Cycle Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC2</td>
<td>FC1</td>
</tr>
<tr>
<td>0 0 0</td>
<td>Undefined, reserved</td>
</tr>
<tr>
<td>0 0 1</td>
<td>User data</td>
</tr>
<tr>
<td>0 1 0</td>
<td>User program</td>
</tr>
<tr>
<td>0 1 1</td>
<td>Undefined, reserved</td>
</tr>
<tr>
<td>1 0 0</td>
<td>Undefined, reserved</td>
</tr>
<tr>
<td>1 0 1</td>
<td>Supervisor data</td>
</tr>
<tr>
<td>1 1 0</td>
<td>Supervisor program</td>
</tr>
<tr>
<td>1 1 1</td>
<td>CPU space (interrupt acknowledge)</td>
</tr>
</tbody>
</table>
Review: Special-Function Pins of the 68000, cont’d
Using the 68000’s function code outputs

Asynchronous Bus Control

- The 68000 is not fully asynchronous because its actions are synchronized with a clock input.
- It can prolong a memory access until an ACK is received, but it has to be in increments of one clock cycle.
Review: Bus Read Cycle

\[ 3 \times t_{\text{cyc}} = t_{\text{CLAV}} + t_{\text{acc}} + t_{\text{DICL}} \]

Figure 4.14

A 68000 Read Cycle

CPE/EE 421/521 Microcomputers
Figure 4.15

Extended Read Cycle

One bus cycle with four wait states

CLK

A<sub>00</sub> - A<sub>03</sub>

AS* UDS* LDS*

R/W*

1

0

DTACK* from memory

D<sub>00</sub> - D<sub>15</sub>

Data valid

DTACK* did not go low at least 20ns before the falling edge of state S4

- Designer has to provide logic to control DTACK*

Figure 4.18

Memory Timing Diagram

- The 6116 static memory component
  - 2K x 8bit memory – **byte-oriented**!
  - Two 6116’s configured in parallel to allow word accesses
  - Eleven address inputs

- 6116 2K x 8-bit RAM

- RAM 8-bit data bus

- RAM control bus

- +5 V

- 0 V

- RAM 11-bit address bus
Assumptions:
- \( R/W^* \) is high for the duration of the read cycle
- \( OE^* \) is low

Data is floating (max 50ns)
- Usually derived from \( UDS^*/LDS^* \)

Connecting The 6116 RAM to a 68000 CPU

---

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>( AS^* )</td>
<td>( RAMCS^* )</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

---

Figure 4.17: Memory Timing Diagram, cont’d

Figure 4.19: Connecting The 6116 RAM to a 68000 CPU
Timing Example

- 68000 clock 8 MHz
  - 68000 CPU
  - 68000 CPU
  - What is the minimum $t_{\text{acc}}$?
    - $3 \times t_{\text{CYC}} > t_{\text{CLAV}} + t_{\text{acc}} + t_{\text{DICL}}$
    - $375 > 70 + t_{\text{acc}} + 15$
    - $t_{\text{acc}} < 290 \text{ ns}$ (or $t_{\text{AA}}$ from the timing diagram, access time)

- For the 12.5MHz version of 68000
  - 68000 CPU
  - 68000 CPU
  - $3 \times 80 > 55 + t_{\text{acc}} + 10$
  - $t_{\text{acc}} < 175 \text{ ns}$

- Remember, maximum $t_{\text{AA}}$ for the 6116 RAM was 200 ns
68000 Write Cycle

- 68000 transmits a byte or a word to memory or a peripheral
- Essential differences:
  - The CPU provides data at the beginning of a write cycle
  - One of the bus slaves (see later) reads the data
- In a read cycle DS* and AS* were asserted concurrently **This will be not a case here!**
- Reason for that: 68000 asserts DS* only when the contents of data bus have stabilized
  - Therefore, memory can use UDS*/LDS* to latch data from the CPU

Simplified write cycle timing diagram

Figure 4.22

In a write cycle: UDS*/LDS* is asserted one cycle after AS*
Follow this sequence of events in a write cycle:

- Address stable
- AS* asserted
- R/W* brought low
- Data valid
- DS* asserted
Write Cycle Timing Diagram of a 6116 RAM, cont’d

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Symbol</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write cycle time</td>
<td>$t_{WC}$</td>
<td></td>
<td>150</td>
</tr>
<tr>
<td>Chip select low to end of write</td>
<td>$t_{CW}$</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>Write recovery time</td>
<td>$t_{WR}$</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Address valid to end of write</td>
<td>$t_{AW}$</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>Address setup time</td>
<td>$t_{AS}$</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Write pulse width</td>
<td>$t_{WP}$</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>Data setup time</td>
<td>$t_{DW}$</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Data hold time</td>
<td>$t_{DH}$</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

- Write cycle ends with either CS* or WE* being negated (CS* and WE* internally combined)
- An address must be valid for at least $t_{AS}$ nanoseconds before WE* is asserted
- Must remain valid for at least $t_{WR}$ nanoseconds after WE* is negated
- Data from the CPU must be valid for at least $t_{DW}$ nanoseconds before WE* is negated
- Must remain valid for at least $t_{DH}$ nanoseconds after the end of the cycle

Designing a Memory Subsystem, an example

- Design a M68000 memory subsystem using
  - Two 32K × 8 RAM chips residing at address $\$00 \ 8000$
  - Two 8K × 8 RAM chips residing in the consecutive window
  - LS 138 (3 to 8 decoder) and basic logic gates

- Solution
  - 32K is $4 \times 8K$
    - Let’s split the address space into 8K modules
  - In total, we have five (4+1) 8K windows
  - To address each line in 8K window
    - $= 13$ bits ($2^3 \times 2^{10} = 2^{13} = 8K$)
  - To address five modules we need 3 bits
  - Don’t forget that there is no $A_0$, we will use LDS/UDS
Designing a Memory Subsystem, an example

![Diagram of memory subsystem](image)

CPE/EE 421/521 Microcomputers 23

---

Designing a Memory Subsystem, an example

![Diagram of memory subsystem](image)

CPE/EE 421/521 Microcomputers 24
Interrupt Control Interface (details later)

![Diagram of Interrupt Control Interface](image)

**Figure 4.9**

**Bus Arbitration Control**

- When 68000 controls the address and data buses, we call it the **bus master**
- The 68000 may allow another 68000 or DMA controller to take control over buses
- In the system with only one bus master, 68000 would have permanent control of the address and data buses
Bus Arbitration Control, cont’d

- 68000 must respond to BR* request (it cannot be masked)
- Assertion of BG* indicates that the bus will be given up at the end of present bus cycle
- Requesting device waits until AS*, DTACK*, and BGACK* have been negated, and only then asserts its own BGACK* output
- Old master negates its BG*, and BR* can be asserted by another potential master

---

Data Bus Contention in Microcomputers

- Situation where more than one device attempts to drive the bus simultaneously
- Example: Two memory modules, M1 selected during read cycle 1, M2 selected during read cycle 2
- Assumption:
  - M1 has data bus drivers with relatively long turn-off times
  - M2 has data bus drivers with relatively short turn-on times
Data Bus Contention in Microcomputers, cont’d

Bus Contention and Data Bus Transceivers

- **Data bus transceiver** – consists of a transmitter (driver) and a receiver
- Driver – tristate output, can be driven high, low, or internally disconnected from the rest of the circuit
- Two control inputs: **Enable** (active low) and **DIR** (direction)
- **Dynamic data bus contention**
Minimal Configuration Using The 68000

DESIGN CONSTRAINTS

- Used in stand-alone mode
- Classroom teaching aid
- 16 KB EPROM-based monitor
- Speed is not important
- At least 4 KB RAM
- 1 serial and 1 parallel port
- Memory expandable
- No interrupts and multiple processors
MAJOR COMPONENTS

- ROM – Two 8K × 8 components
- RAM – Two 2K × 8 components
- Parallel – 6821 Peripheral Interface Adapter (PIA)
- Serial – 6850 Asynchronous Comm. Interface Adapter (ACIA)

DESIGN CHOICES

- Chose the location of ROM (16KB) and RAM (8 KB) within the address space (16 MB)
  - Unimportant, as long as the reset vectors are located at $00000000$
- Chose the location of memory-mapped peripherals
- Control of DTACK* (is delay applied or not?)
The 68000’s Reset Sequence

RESER SEQUENCE

- Set SR S bit to 1
- Set SR T bit to 0
- Set SR mask to 111

- Fetch SSP from address 0
- Transfer longword to SSP

- Fetch initial PC from address 4
- Transfer longword to PC

- Begin processing in the supervisor state

- Bus error occurs?
- Double bus error
- FATAL ERROR

REMEMBER

- When the RESET* pin is asserted for the appropriate duration:
  - SR = $2700
  - SSP is loaded with the longword @ $00 0000
  - PC is loaded with the longword @ $00 0004
Memory and Peripheral Components

- We assigned address lines to address pins, and data lines to data pins.
- Before designing logic that will generate chip select signals, we have to decide about RAM/ROM location.
- To assure that the reset vector location is at $00 0000$, let's situate $16$ KB of ROM at $00 0000$
Control Section

- We will divide the memory space $00\ 0000 - \$01\ FFFF$ into eight blocks of 16 KB (IC1a,b, IC2a, IC3)
- 16 KBytes of ROM are at $00\ 0000$ to $00\ 3FFF$
- Where is the RAM situated? Peripherals?
- Note: there is no delay applied to DTACK*.
- What will happen if we access non-decoded memory?
Different approaches to memory arrangement

- Largest memory window (16 KB) [MEMORY GAPS]
Different approaches to memory arrangement, cont’d

- Smallest memory window (4 KB) [NO MEMORY GAPS]

![Diagram of memory window and decoder]

- ROM (16 KB) 4 Windows (Blocks)

How can we make it better?

- ROM is EPROM-based, and thus slower
- With EPROMs from the same generation, we’ll need wait states, maybe even with RAM components
- Watchdog for non-decoded memory addresses
How can we make it better?

- CONTROL OF INTERRUPTS
  - Use 74LS148 priority encoder to provide 7 levels of interrupt

- EXTERNAL BUS INTERFACE
  - CPU can supply only the limited current to drive the bus
  - SOLUTION: Bus drivers (buffers)
DTACK* Generation

- DTACK* generator based on a shift register

![Diagram of DTACK* generator](image)

Figure 4.72

Shift register and its timing diagram

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEAR, CLOCK</td>
<td>A, B, Q3, Q2, Q1, Q0, Qn</td>
</tr>
</tbody>
</table>

Function table

- Typical clear, shift, and clear sequences
DTACK* Generation

Shift register and its timing diagram

Figure 4.74

DTACK* Generation

- DTACK* generator based on a counter