Course Administration

- Instructor: Aleksandar Milenkovic
  milenko@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: hw3&lab5 due on 11/28/05.
- Text: Microprocessor Systems Design: 68000 Hardware, Software, and Interfacing
- Today: M68000 HW
- Review: Chapter 4, Chapter 5, Chapter 6 (M68K hw)

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: A31 - A0
  - Data Bus: D31 - D0
  - Asynchronous bus control: AS*, R/W*, UDS*, LDS*, DTACK*, BERR*
  - Synchronous bus control: E, VPA*, VMA*
  - Bus arbitration control: BR*, BG*, BGACK*
  - Function code: FC0, FC1, FC2
  - System control: CLK, RESET*, HALT*
  - Interrupt control: IPL0*, IPL1*, IPL2*
  - Miscellaneous: Vcc(2), Gnd(2)
- Legend: Type
  - XX Input
  - XX Output
  - XX Input/Output

Review: Memory and Peripheral Interface Pins

- Address bus: Address pins A23 to A0
- Data bus: Data pins D15 to D0
- Asynchronous bus control: Asynchronous signals such as R/W, UDS, LDS, DTACK, BERR
- Synchronous bus control: Synchronous signals such as E, VPA, VMA
- Bus arbitration control: Signals for bus arbitration including BR, BG, BGACK
- Function code: Different function codes for different processor cycles
- System control: Signals for system control including CLK, RESET, HALT
- Interrupt control: Signals for interrupt control including IPL0, IPL1, IPL2
- Miscellaneous: Power supply signals Vcc and ground Gnd

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 FC1 FC0</td>
<td></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>Undefined, reserved</td>
</tr>
<tr>
<td>1 0 0</td>
<td>Undefined, reserved</td>
</tr>
<tr>
<td>0 1 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

Review: Special-Function Pins of the 68000, cont’d
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.

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

The 68000 CPU

Address

Data in

3 \( t_{DY} = t_{CLAV} + t_{KOC} + t_{DBCL} \)

Memory device

A 68000 Read Cycle

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

Extended Read Cycle

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

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
Assumptions:

- R/W* is high for the duration of the read cycle
- OE* is low

Data is floating

Timing Example

- 68000 clock 8 MHz
  - t_CIC = 125 ns
  - t_CLAV = 70 ns
  - t_DICL = 15 ns
- What is the minimum t_\text{acc}? 
  - 3 \times t_CIC > t_CLAV + t_\text{acc} + t_DICL 
  - 375 > 70 + t_\text{acc} + 15 
  - t_\text{acc} < 290 ns (or t_\text{AA} from the timing diagram, access time)

- For the 12.5 MHz version of 68000
  - t_CIC = 80 ns
  - t_CLAV = 55 ns
  - t_DICL = 10 ns
  - 3 \times 80 > 55 + t_\text{acc} + 10
  - t_\text{acc} < 175 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 not be 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
Follow this sequence of events in a write cycle:
- Address stable
- AS* asserted
- R/W* brought low
- Data valid
- DS* asserted

Figure 4.23

Follow this sequence of events in a write cycle:
- Address stable
- AS* asserted
- R/W* brought low
- Data valid
- DS* asserted

Figure 4.23

Write Cycle Timing Diagram of a 6116 RAM

- 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

Figure 4.24

Designing a Memory Subsystem, an example

- Design a M68000 memory subsystem using
  - Two 32K \( \times 8 \) RAM chips residing at address \( \text{50000} \)
  - Two 8K \( \times 8 \) RAM chips residing in the consecutive window
  - LS 138 (3 to 8 decoder) and basic logic gates

Solution
- 32K is \( 4 \times 8 \)
  - 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
  - \( 2^1 \times 2^3 = 2^4 = 16 \) bits
  - 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
Interrupt Control Interface (details later)

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 $00 0000
- Chose the location of memory-mapped peripherals
- Control of DTACK* (is delay applied or not?)

The 68000's Reset Sequence

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]

![Memory Window Diagram](image)

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? Cont’d

- 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

![DTACK* Generation Diagram](image)

DTACK* Generation

Shift register and its timing diagram

![Timing Diagram](image)
DTACK* Generation
Shift register and its timing diagram

Interrupt Processing Mechanism
- Interrupt is an asynchronous event
- When an interrupt occurs, the computer can:
  - Service it
  - Ignore it (for the time being)

Interrupt processing mechanism, cont’d
- Sequence of actions when an interrupt is being serviced:
  1. The computer completes its current machine-level instruction
  2. The contents of PC is saved (on stack)
  3. The state of the processor (status word) is saved on the stack
  4. Jump to the location of the interrupt handling routine

Interrupt Control Interface (details later)
- The interrupt request:
  - Can be deferred or denied
  - When it is deferred, it is said to be masked
  - Special one: nonmaskable interrupt request (NMI)
  - The 68000 NMI: IRQ7 (MSP430: RST*/NMI pin)

Prioritized interrupts
- Vectored interrupts
  - Requesting peripheral identifies itself, CPU doesn’t have to poll the status of each device to discover the interrupter
Reset, bus error, address error, and trace exceptions take precedence over an interrupt.

A level 7 interrupt CAN interrupt level 7 interrupt.