CPE/EE 421 Microcomputers:
Motorola 68000 –
The CPU Hardware Model

Instructor: Dr Aleksandar Milenkovic
Lecture Notes

Outline
- 68000 interface
- Timing diagram
- Minimal configuration using the 6800
- Extensions
- Exception Processing

68000 Interface
M68000: 64 pins, arranged in 9 groups:
- Address Bus: A01 – A23
- Data Bus: D00 – D15
- Asynchronous bus control: AS*, RW*, 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:
- XX: Type
- Input
- Output
- Input/Output

68000 Interface, cont’d
Classification of pins based on function
- SYSTEM SUPPORT PINS
  - Essential in every 68000 system (power supply, clock, …)
- MEMORY AND PERIPHERAL INTERFACE PINS
  - Connect the processor to an external memory subsystem
- SPECIAL-PURPOSE PINS
  (not needed in a minimal application of the processor)
  - Provide functions beyond basic system functions

Terminology
- Asterisk following a name: indicates the signal is active low
- “Signal is asserted” means signal is placed in its active state
- “Signal is negated” means signal is placed in its inactive state
**System Support Pins**
- **Power Supply**
  - Single +5V power supply: 2 Vcc pins and 2 ground pins
- **Clock**
  - Single-phase, TTL-compatible signal
  - Bus cycle: memory access, consists of a minimum 4 clock cycles
  - Instruction: consists of one or more bus cycles
- **RESET**
  - Forces the 68000 into a known state on the initial application of power:
    - Supervisor’s A7 is loaded from memory location $00 0000
    - Program counter is loaded from address $80 0000
  - During power-up sequence must be asserted together with the HALT* input for at least 100 ms.
  - Acts also as an output, when processor executes the instruction reset* (used to reset peripherals w/out resetting the 68000)

**System Support Pins, cont’d**
- **HALT**
  - In simple 68000 systems can be connected together with RESET*
  - Can be used:
    - By external devices to make the 68000 stop execution after current bus cycle (and to negate all control signals)
    - To single-step (bus cycle by bus cycle) through program
    - To rerun a failed bus cycle (if memory fails to respond correctly) in conjunction with the bus error pin, BERR*
  - It can be used as an output, to indicate that the 68000 found itself in a situation from which it cannot recover (HALT* is asserted)

**Memory and Peripheral Interface Pins**
- **Address Bus**
  - 23-bit address bus, permits 223 16-bit words to be addressed
  - Tri-state output pins (to permit devices other than the CPU to take a control over it)
  - Auxiliary function:
    - Supports vectored interrupts
    - Address lines A01, A02, A03 indicate the level of the interrupt being serviced
    - All other address lines are set to a high level
- **Data Bus**
  - Bi-directional 16-bit wide data bus
  - During a CPU read cycle acts as an input
  - During a CPU write cycle acts as an output
  - Byte operations: only D00-D07 or D08-D15 are active
  - Interrupting device identifies itself to the CPU by placing an interrupt vector number on D00-D07 during an interrupt acknowledge cycle

**Memory and Peripheral Interface Pins, cont’d**
- **AS**
  - When asserted, indicates that the content of the address bus is valid.
- **R/W**
  - Determines the type of a memory access cycle
    - CPU is reading from memory: R/W* = 1
    - CPU is writing to memory: R/W* = 0
    - If CPU is performing internal operation, R/W* is always 1
    - When CPU relinquishes control of its busses, R/W* is undefined
- **UDS** and **LDS**
  - Used to determine the size of the data being accessed
  - If both UDS* and LDS* are asserted, word is accessed
  - R/W* UDS* LDS*
    - 010: write lower byte (D00 – D07: data valid, replicated on D8-D15)
    - 011: write word (D00 – D15: data valid)
    - 101: read upper byte (D00 – D07: invalid, D8-D15: data valid)
Memory and Peripheral Interface Pins, cont’d

- **DTACK* (Data Transfer Acknowledge)**
  - Handshake signal generated by the device being accessed
  - Indicates that the contents of the data bus is valid
  - **If DTACK* is not asserted**, CPU generates waitstates until DTACK goes low or until an error state is declared.
  - **When DTACK* is asserted**, CPU completes the current access and begins the next cycle
  - DTACK* has to be generated a certain time after the beginning of a valid memory access (timer supplied by the system designer).

Special-Function Pins of the 68000

- **BERR* (Bus Error Control)**
  - Enables the 68000 to recover from errors within the memory system
- **BR*, BG*, BGACK* (Bus Arbitration Control)**
  - Used to implement multiprocessor systems based on M68000
- **FC0-FC2 (Function Code Output)**
  - Indicate the type of cycle currently being executed
  - Becomes valid approximately half a clock cycle earlier than the contents of the address bus
- **IPL0*-IPL2* (Interrupt Control Interface)**
  - Used by an external device to indicate that it requires service
  - 3-bit code specifies one of eight levels of interrupt request

Function Code Outputs

<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>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>
Special-Function Pins of the 68000 Using FC Outputs

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.

Outline
- 68000 interface
- Timing diagram
- Minimal configuration using the 68000
- Extensions
- Exception Processing

Timing Diagram of a Simple Flip-Flop

Actual behavior of a D flip-flop
The Clock

- A microprocessor requires a clock that provides a stream of timing pulses to control its internal operations.
- A 68000 memory access takes a minimum of eight clock states numbered from clock state S0 to clock state S7.

Bus Cycle

A memory access begins in clock state S0 and ends in state S7.

The most important parameter of the clock is the duration of a cycle, \( t_{\text{Cyc}} \).
At the start of a memory access, the CPU sends the address of the location it wishes to read to the memory.

We are interested in when the 68000 generates a new address for use in the current memory access.

The next slide shows the relationship between the new address and the state of the 68000's clock.

Initially, in state S0 the address bus contains the old address. In state S1 a new address becomes valid for the remainder of the memory access.

The time at which the contents of the address bus change can be related to the edges of the clock.
Address Timing
- Let’s look at the sequence of events that govern the timing of the address bus.
- The “old” address is removed in state S0.
- The address bus is floated for a short time, and the CPU puts out a new address in state S1.

Bus Cycle
- The old address is removed in clock state S0 and the address bus floated.
- The designer is interested in the point at which the address first becomes valid. This point is $t_{CLAV}$ seconds after the falling edge of S0.
- The memory needs to know when the address from the CPU is valid. An address strobe, $A5^*$, is asserted to indicate that the address is valid.
Address and Address Strobe
- We are interested in the relationship between the time at which the address is valid and the time at which the address strobe, AS*, is asserted.
- When AS* is active-low it indicates that the address is valid.
- We now look at the timing of the clock, the address, and the address strobe.

Bus Cycle
- AS* goes active low after the address has become valid.
- AS* goes inactive high before the address changes.

The Data Strobes
- The 68000 has two data strobes LDS* and UDS*. These select the lower byte or the upper byte of a word during a memory access.
- To keep things simple, we will use a single data strobe, DS*.
- The timing of DS* in a read cycle is the same as the address strobe, AS*.
The data strobe, is asserted at the same time as AS* in a read cycle.

The Data Bus
- During a read cycle the memory provides the CPU with data.
- The next slide shows the data bus and the timing of the data signal.
- Note that valid data does not appear on the data bus until near the end of the read cycle.

Analyzing the Timing Diagram
- We are going to redraw the timing diagram to remove clutter.
- We aren’t interested in the signal paths themselves, only in the relationship between the signals.
We are interested in the relationship between the clock, AS*/DS* and the data in a read cycle.

The earliest time at which the memory can begin to access data is measured from the point at which the address is first valid.

Address becomes valid

Data becomes valid

The time between address valid and data valid is the memory’s access time, $t_{\text{acc}}$.

Calculating the Access Time

- We need to calculate the memory’s access time.
- By knowing the access time, we can use the appropriate memory component.
- Equally, if we select a given memory component, we can calculate whether its access time is adequate for a particular system.
Data from the memory is latched into the 68000 by the falling edge of the clock in state S6.

Data must be valid \( t_{DICL} \) seconds before the falling edge of S6.

We know that the time between the address valid and data valid is \( t_{ACC} \).

The address becomes valid \( t_{CLAV} \) seconds after the falling edge of S0.
From the falling edge of S0 to the falling edge of S6:
• the address becomes valid
• the data is accessed
• the data is captured

3 \text{tcyc} = \text{tCLAV} + \text{tacc} + \text{tDICL}

Timing Example

- 68000 clock 8 MHz \( t_{\text{CYC}} = 125 \text{ ns} \)
- 68000 CPU \( t_{\text{CLAV}} = 70 \text{ ns} \)
- 68000 CPU \( t_{\text{DICL}} = 15 \text{ ns} \)
- What is the minimum \( t_{\text{acc}} \)?
  - \( 3 \ t_{\text{CYC}} = t_{\text{CLAV}} + t_{\text{acc}} + t_{\text{DICL}} \)
  - 375 = 70 + \( t_{\text{acc}} \) + 15
  - \( t_{\text{acc}} = 290 \text{ ns} \)
A 68000 Read Cycle

Extended 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

Memory Timing Diagram, cont'd

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

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

Designer has to provide logic to control DTACK*

Data valid

Data floating

Address valid

Address valid
Connecting The 6116 RAM to a 68000 CPU

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0x</td>
<td>A11</td>
</tr>
<tr>
<td>A11</td>
<td>A1x</td>
</tr>
</tbody>
</table>

### Figure 4.19
Connecting The 6116 RAM to a 68000 CPU

Timing Diagram:
- No operation
- Lower byte read
- Upper byte read
- Word read
- No operation

#### Figure 4.20
Connecting The 6116 RAM to a 68000 CPU Timing Diagram

- Turnoff time: 70 + 10 + 60 = 140 ns

### Timing Example

68000 clock 8 MHz

- 68000 CPU
- 68000 CPU
- 68000 CPU
- What is the minimum t_{min}?
  - 3 × t_{VC} > t_{CLAV} + t_{CLCL} + t_{DICL}
  - 375 > 70 + 15
  - t_{min} < 290 ns (or t_{AA} from the timing diagram, access time)

- For the 12.5 MHz version of 68000
  - t_{VC} = 80 ns
  - t_{CLAV} = 55 ns
  - t_{CLCL} = 10 ns
  - 3 × 55 > 55 + t_{min} + 10
  - t_{max} < 175 ns

- Remember, maximum t_{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

68000 Write Cycle Diagram
In a write cycle, UDS*/LDS* is asserted one cycle after AS*.

Write Cycle

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

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.

Write Cycle Timing Diagram of a 6116 RAM, cont'd
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 × 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 (2^13 = 8K)
- To address five modules we need 3 bits
- Don’t forget that there is no A0, we will use LDS/UDS

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 the buses.
- In the system with only one bus master, 68000 would have permanent control of the address and data buses.

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.

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.
**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**

**Outline**

- 68000 interface
- Timing diagram
- Minimal configuration using the 68000
- Extensions
- Exception Processing

**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?)

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]

Outline

- 68000 interface
- Timing diagram
- Minimal configuration using the 68000
- Extensions
- Exception Processing

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

Shift register and its timing diagram
DTACK* Generation

Shift register and its timing diagram

DTACK* Generation

DTACK* generator based on a counter

Figure 4.74

Outline

- 68000 interface
- Timing diagram
- Minimal configuration using the 68000
- Extensions
- Exception Processing

Interrupt Processing Mechanism

- Interrupt is an asynchronous event
- When an interrupt occur, the computer can:
  - Service it
  - Ignore it (for the time being)
Interrupt processing mechanism, cont’d

- The interrupt is transparent to the interrupted program
- 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

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
The 68000 Interrupt Interface

The 68000 Interrupt Interface

Reset, bus error, address error, and trace exceptions take precedence over an interrupt
A level 7 interrupt CAN interrupt level 7 interrupt

Interrupt Timing Diagram

Interrupt acknowledge timing and the IACK cycle
Exception Vectors

- A vector is associated with each type of exception
  - Vector is the 32-bit absolute address of the appropriate exception handling routine
  - 256 exception vectors, 32 bits (4 bytes) each, extending from address $000000$ to $00003FF$
  - Vectors 0-63 : EXCEPTIONS
  - Vectors 64-255 : INTERRUPT HANDLING ROUTINES

- Difference between the reset vector and all other exceptions:
  - It requires 2 longwords
  - Located in SP space (FC = 110); others are in SD space (FC = 101)

Privileged States and the 68000

User programs operate only

- Figure 6.13 illustrates the status byte of the 68000’s status register. The definitions of the bits in the status byte are as follows:
  - S: When set, the supervisor state bit indicates that the 68000 is in its supervisor state. When clear, S indicates that the 68000 is in user state.
  - T: When the trace bit in the trace register is 1, the 68000 generates a trace exception after the execution of each instruction. The trace exception is used in debugging programs.
  - I, E, L: The interrupt mask bits. I, E, and L indicate the level of the current interrupt mask (i.e., 0 to 3).

- Figure 6.13 Format of the status bits of the 68000’s status register

- An exception always forces the 68000 into the supervisor state
The 68000 Exceptions

Interrupts and Real-time Processing

- Multitasking (multiprogramming)
  - concurrent execution
  - multiple tasks (processes)
  - resource sharing (multiple users using the same printer)

- Multiprocessing
  - parallel execution
  - multiple PROCESSORS!

Multitasking

- Operating system (to schedule activities)
- Interrupt mechanism (to switch between tasks)

Real-Time Operating System

- Real time - meaningful time
  - fast enough to influence the system at that moment
    - space shuttle / chemical plant

- Real-time system
  - Optimizes the response time to events
  - Tries to use resources efficiently

- Multitasking system
  - Optimizes resource utilization
  - Tries to provide a reasonable response time

---

Alex Milenkovich
Real-Time Kernel

- Scheduler is the kernel, nucleus, of a real-time OS
- Functions
  - A first-level interrupt handler
  - Scheduler - the sequence in which tasks are executed
  - Interprocess communication
- Task States
  - Ready
  - Running
  - Blocked (dormant)

Tasks

- Volatile portion (PC, status, registers)
- Task control block (TC)
  - Task ID
  - Task block pointer
  - PC
  - SP
  - Status register
  - Other registers
  - Task status
    - Run / ready / blocked
  - Task priority
  - Task time allocation
  - How many slots

Exception Handling and Tasks

- Preemptive real-time OS:
  - RTC generates periodic interrupts
  - Used by the kernel to locate and run the next task
- How to deal with other interrupts?
  - Service them independently, subject to priority
  - Integrate them into the real-time task structure

Figure 6.25