The 68000 CPU Hardware Model

- 68000 interface
- Timing diagram
- Minimal configuration using the 68000
4.1 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: $FC0, FC1, FC2$
  - System control: $CLK, RESET^*, HALT^*$
  - Interrupt control: $IPL0^*, IPL1^*, IPL2^*$
  - Miscellaneous: $Vcc(2), Gnd(2)$

4.1 68000 Interface (continued)

- 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
4.1 68000 Interface – 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 $000000$  
    - Program counter is loaded from address $000004$
  - 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)

---

4.1 68000 Interface – System Support Pins (continued)

- **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 situation from which it cannot recover (HALT* is asserted)
**4.1 68000 Interface – Memory and Peripheral Interface Pins**

- **Address Bus**
  - 23-bit address bus, permits $2^{23}$ 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 $A_{01}, A_{02}, A_{03}$ 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 $D_{00}-D_{07}$ or $D_{08}-D_{15}$ are active
  - Interrupting device identifies itself to the CPU by placing an *interrupt vector number* on $D_{00}-D_{07}$ during an interrupt acknowledge cycle

---

**4.1 68000 Interface – Memory and Peripheral Interface Pins (continued)**

- **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 ($D_{00} - D_{07}$: data valid, replicated on $D_8-D_{15}$)
    - 011: write word ($D_{00} - D_{15}$: data valid)
    - 101: read upper byte ($D_{00} - D_{07}$: invalid, $D_8-D_{15}$ – data valid)
4.1 68000 Interface – Memory and Peripheral Interface Pins (continued)

- **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 wait-states 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).
4.1 68000 Interface – Special Function Pins

- **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 Output

<table>
<thead>
<tr>
<th>FC2</th>
<th>FC1</th>
<th>FC0</th>
<th>Processor Cycle Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Undefined, reserved</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>User data</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>User program</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Undefined, reserved</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Undefined, reserved</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Supervisor data</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Supervisor program</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>CPU space (interrupt acknowledge)</td>
</tr>
</tbody>
</table>
### 4.1 68000 Interface – Special-Function Pins: 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.**

Note: Control signals shown are active-high.
### 4.2 Timing Diagram – D Flip-Flop

**Idealized form of the timing diagram**

![Idealized timing diagram for D flip-flop](image)

**Actual behavior of a D flip-flop**

![Actual behavior of D flip-flop](image)

- **Data hold time**: Max time for output to become valid after clock.
- **Data setup time**: Time required for the input data to be stable.
- **Max time for output to become valid after clock**: The maximum time allowed for the output to be valid.

### 4.2 Timing Diagram – General

**General form of the timing diagram**

![General timing diagram](image)

**An alternative form of the timing diagram**

![Alternative timing diagram](image)
4.2 Timing Diagram – 68000 Memory Cycle

- A microprocessor requires a clock that provides a stream of ________ to control its internal operations.
- A 68000 memory access takes a minimum of ______ clock states numbered from clock state ___ to clock state ___.

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.
4.2 Timing Diagram – 68000 Memory Cycle: Address Timing

- We are interested in _____ the 68000 generates a new address for use in the current memory access
- 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
- The next slide shows the relationship between the ________ and the state of the 68000’s ________

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.

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 $S_0$. 
The memory needs to know when the address from the CPU is valid. An address strobe, AS*, is asserted to indicate that the address is valid.

4.2 Timing Diagram – 68K Memory Cycle: How Does Memory see a Valid Address?

- We are interested in the relationship between the time at which the __________ and the time at which the address strobe, AS*, is _________
- When AS* is _________ it indicates that the address is valid
- We now look at the timing of the clock, the address, and the address strobe
4.2 Timing Diagram – 68K Memory Cycle: Address Strobe Timing

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

4.2 Timing Diagram – 68K Memory Cycle: AS Relative to Clock

- **AS*** goes low in clock state S2.
• The 68000 has two data strobes _______ and _______. These select the _____ byte or the _______ byte of a word during a memory access.
• To keep things simple, we will use a single ____________, DS*.
• The timing of DS* in a read cycle is the same as the address strobe, AS*.
During a read cycle the ____________ provides the ____________ with data.
The next slide shows the ____________ and the timing of the ____________.
Note that valid data does not appear on the data bus until near the end of the read cycle.
4.2 Timing Diagram – 68K Memory 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 Address valid and the Data out in a read cycle
4.2 Timing Diagram – 68K Memory Cycle: \( t_{acc} \)

- Address becomes valid
- Data becomes valid

---

4.2 Timing Diagram – Calculating \( t_{acc} \)

- 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.
4.2 Timing Diagram – 68K Memory Cycle: Putting it all Together

From the falling edge of S0 to the falling edge of S6:
- the address becomes valid
- the data is accessed
- the data is captured

4.2 Timing Diagram – 68K Memory Cycle: Timing Constraint

3 \times t_{cyc} = t_{CLAV} + t_{acc} + t_{DICL}
4.2 Timing Diagram – 68K Memory
Cycle: Timing Example

- 68000 clock 8 MHz \( t_{CYC} = 125 \text{ ns} \)
- 68000 CPU \( t_{CLAV} = 70 \text{ ns} \)
- 68000 CPU \( t_{DICL} = 15 \text{ ns} \)
- What is the maximum \( t_{acc} \)?

4.2 Timing Diagram – 68K Memory
Cycle: A 68000 Read Cycle
4.2 Timing Diagram – 68K Memory Cycle: Read Cycle with Wait States

- Designer has to provide logic to control DTACK*

4.2 Timing Diagram – Memory Timing Diagram: The 6116 SRAM Schematic

- 2K x 8bit memory – byte-oriented!
- Two 6116’s configured in parallel to allow word accesses
- Eleven address inputs
4.2 Timing Diagram – Memory Timing

Diagram: The 6116 SRAM Timing

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

Diagram: Connecting the 68K and 6116

Electrical and Computer Engineering
4.2 Timing Diagram – Memory Timing

Diagram: Read Cycle Timing

- 68000 clock 8 MHz
  - \( t_{CYC} = 125 \text{ ns} \)
  - \( t_{CLAV} = 70 \text{ ns} \)
  - \( t_{DICL} = 15 \text{ ns} \)
  - What is the minimum \( t_{acc} \)?
    - \( 3 \times t_{CYC} > t_{CLAV} + t_{acc} + t_{DICL} \)

- For the 12.5MHz version of 68000
  - \( t_{CYC} = 80 \text{ ns} \)
  - \( t_{CLAV} = 55 \text{ ns} \)
  - \( t_{DICL} = 10 \text{ ns} \)
  - \( 3 \times 80 > 55 + t_{acc} + 10 \)
  - \( t_{acc} < 175 \text{ ns} \)

- Remember, maximum \( t_{AA} \) for the 6116 RAM was 200 ns
4.2 Timing Diagram – Memory Timing

Diagram: 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 the 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

Diagram: Simplified Write Cycle Timing

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

4.2 Timing Diagram – Memory Timing
Diagram: Write Cycle Events

4.2 Timing Diagram – Memory Timing
Diagram: Write Cycle Timing

Address setup time (min 20ns)
Address valid to end of write (min 120ns)
Write pulse width (min 90ns)
Write recovery time (min 10ns)
4.2 Timing Diagram – Memory Timing

Diagram: Write Cycle Timing Details

- 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

<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>90</td>
<td>150</td>
</tr>
<tr>
<td>Chip select low to end of write</td>
<td>$t_{CW}$</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Write recovery time</td>
<td>$t_{WR}$</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>Address valid to end of write</td>
<td>$t_{AV}$</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Write pulse width</td>
<td>$t_{WP}$</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>Data setup time</td>
<td>$t_{SU}$</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>

4.2 Timing Diagram – Memory Timing

Diagram: Memory Subsystem Design

- Design a M68000 memory subsystem using
  - Two 32K $\times$ 8 RAM chips residing at address $000000_8$
  - Two 8K $\times$ 8 RAM chips residing in the consecutive window
  - LS 138 (3 to 8 decoder) and basic logic gates

- Solution
  - 32K is $4 \times 8K$
    - 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 A_0**, we will use LDS/UDS
4.2 Timing Diagram – Memory Timing

Diagram: Memory Subsystem Addressing

Diagram: Memory Subsystem Connections
4.2 Timing Diagram – Memory Timing

Diagram: Data Bus Contention Scenario

- 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
4.2 Timing Diagram – Memory Timing

Diagram: 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)

Diagram: Dynamic Bus Contention

- Write-to-Read Data-Bus-to-Data-Bus Contention
4.4 Minimal 68K Configuration – 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

4.4 Minimal 68K Configuration – 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)
4.4 Minimal 68K Configuration – 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 $000000$
- Chose the location of memory-mapped peripherals
- Control of DTACK* (is delay applied or not?)

4.4 Minimal 68K Configuration – Block Diagram
4.4 Minimal 68K Configuration – Memory and Peripheral Decisions

- 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 $000000$, let’s situate 16 KB of ROM at $000000$
4.4 Minimal 68K Configuration – Control Section Decisions

- 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?

4.4 Minimal 68K Configuration – Control Section Schematic
4.4 Minimal 68K Configuration – Memory Management Approaches: Gaps

- Largest memory window (16 KB) [MEMORY GAPS]

4.4 Minimal 68K Configuration – Memory Management Approaches: No Gaps

- Smallest memory window (4 KB) [NO MEMORY GAPS]
4.4 Minimal 68K Configuration – Memory Management Improvements

- 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

4.4 Minimal 68K Configuration – Further Considerations

- 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)
4.4 Enhanced Minimal 68K Configuration

- DTACK* generator based on a shift register
4.4 Minimal 68K Configuration – Improvements: DTACK* Generation

Function table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEAR</td>
<td>CLOCK</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
</tr>
</tbody>
</table>

Typical clear, shift, and clear sequences

4.4 Minimal 68K Configuration – DTACK* Generation Timing

CPE/EE 421/521
4.4 Minimal 68K Configuration – Improvements: DTACK* Generation

- DTACK* generator based on a counter

4.4 68K Configuration Enhancement – Bus Arbitration Control

- When 68000 controls the address and data buses, we call it the ___________
- The 68000 may allow ______________ or __________ to take control over the buses
- In the system with only one bus master, 68000 would have __________ control of the address and data buses
4.4 68K Enhanced Configuration – Bus Arbitration Control Timing

- 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 the 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