I/O Systems, MIMD Architecture & DMA Controller Analysis a) Comparison: Programmed I/O vs Interrupt Driven I/O Programmed I/O (Polling) **Aspect Interrupt Driven I/O CPU Utilization** Poor - CPU continuously polls device status Good - CPU free to do other tasks **Response Time** Fast - Immediate detection of device readiness Variable - Depends on interrupt latency **Implementation** Simple - No special hardware needed Complex - Requires interrupt handling mechanism **CPU Overhead** High - Continuous status checking Low - Only when interrupt occurs Multitasking Poor - CPU tied up in polling loop Excellent - Supports concurrent operations **Synchronization** Synchronous - CPU waits for completion Asynchronous - CPU notified when ready **Power Consumption** Lower - CPU can enter idle states High - Continuous CPU activity **Best Use Case** High-speed devices, simple systems Multiple devices, complex systems Programmed I/O Flow Interrupt Driven I/O Flow **CPU Issues I/O Command CPU Issues I/O Command Check Device Status CPU Continues Other Tasks** Device Ready? **Device Sends Interrupt** ↓ No ↓ Yes **Continue Polling** Transfer Data **CPU Services Interrupt Transfer Data & Resume Key Differences Summary: Programmed I/O Advantages:** Interrupt I/O Advantages: Simple implementation Better CPU utilization Predictable timing Supports multitasking No interrupt overhead Lower power consumption Full CPU control • Scalable to multiple devices b) Multiple Input, Multiple Data Stream (MIMD) Architecture Models **MIMD Definition:** MIMD (Multiple Instruction, Multiple Data) is a parallel computing architecture where multiple processors execute different instructions on different data simultaneously. It's the most flexible and widely used parallel architecture. **Model 1: Shared Memory MIMD (Tightly Coupled)** Symmetric Multiprocessing (SMP) Architecture Processor 1 Processor 2 Processor 3 Processor N Cache Cache Cache Cache **Shared System Bus** (Address, Data, Control) Shared Memory **Shared Memory** I/O I/O Module 1 Module 2 Controller Controller **Advantages** Disadvantages • Uniform Memory Access • Easy Programming Model Limited Scalability Cache Coherency • Fast Inter-processor Communication Bus Contention • Shared Address Space • Dynamic Load Balancing • Cache Coherency Overhead • Low Latency Communication • Shared Data Structures • Single Point of Failure **Model 2: Distributed Memory MIMD (Loosely Coupled)** Message Passing Multicomputer Architecture Local Memory 1 Processor 1 Processor 2 Local Memory 2 Network Interface **Network Interface** Local Memory 3 Local Memory N Processor 3 Processor N **Network Interface Network Interface Interconnection Network** (Mesh, Hypercube, Torus, etc.) Disadvantages Advantages • Highly Scalable • Complex Programming • Non-Uniform Memory Access • Message Passing Communication • High Communication Latency • No Cache Coherency Issues • Private Address Spaces Fault Tolerant • Load Balancing Challenges High Scalability Cost Effective • Data Distribution Overhead c) Direct Memory Access (DMA) - Definition and Operation (2+4=6) What is Direct Memory Access (DMA)? Direct Memory Access (DMA) is a feature that allows certain hardware components to access main system memory independently of the central processing unit (CPU). DMA enables high-speed data transfer between memory and I/O devices without continuous CPU intervention, significantly improving system performance. **Key Benefits of DMA:** • CPU Efficiency: Frees CPU from data transfer tasks • **High Throughput:** Direct memory-to-device transfer • Reduced Latency: Eliminates CPU bottleneck • Concurrent Processing: CPU can perform other tasks **DMA Controller Block Diagram** DMA Controller I/O Device CPU • Address Register • Word Count Register Disk Controller • Program Execution • Control Register Network Interface • DMA Programming • Status Register • Interrupt Handling • Address Generator Audio Controller • Bus Control Logic Main Memory • System RAM • Data Buffers • Device Drivers System Bus (Address, Data, Control) **DMA Operation Process Phase 3: Data Transfer Phase 4: Completion Phase 1: Initialization Phase 2: Bus Request** • I/O device signals DMA • CPU programs DMA controller • DMA releases bus control DMA controls system bus • Sets memory address controller • Direct transfer between • CPU resumes normal • Sets transfer count • DMA sends bus request to memory and I/O operation **CPU** • DMA sends interrupt to CPU • Sets transfer direction Address register incremented • CPU completes current Word count decremented • CPU handles completion tasks • Enables DMA operation Transfer continues until • Status updated in registers instruction • CPU grants bus control complete • CPU enters wait state **DMA Transfer Modes** Cycle Stealing Transparent Mode **Burst Mode** DMA takes complete control of the bus DMA transfers only when CPU is not until entire block is transferred using the bus Advantage: Highest transfer rate Advantage: Balance between speed and Advantage: No CPU interference **Disadvantage:** CPU completely blocked **Disadvantage:** Slowest transfer rate **Disadvantage:** Moderate transfer rate **DMA Controller Working with System Components Detailed DMA System Interaction CPU** Responsibilities **DMA Controller Functions** Memory Device Role I/O Device Interface Provide data storage Request DMA transfers • Initialize DMA controller • Generate memory addresses Program transfer Control bus signals • Respond to address signals Provide/accept data Manage data transfer Accept/provide data • Signal transfer readiness parameters Handle DMA interrupts timing Support burst transfers Handle device-specific protocols • Manage bus arbitration • Handle transfer completion Maintain data integrity • Interface with I/O devices Manage buffer states • Process completion status **DMA Transfer Timing Diagram DMA Operation Timeline** CPU Activity: [EXEC][EXEC][WAIT][WAIT][WAIT][EXEC][EXEC][EXEC][INT][EXEC] | DMA TRANSFER DMA Signals: - DREQ - DACK - Bus Request __ Legend: WAIT = CPU Waiting for Bus DREQ = DMA Request DMA = DMA Controller Active DACK = DMA Acknowledge = Signal Active ——= Signal Inactive **DMA System Integration Benefits:**

Performance Improvements: System Considerations: • Throughput: Up to 100x faster than programmed I/O • Bus Arbitration: Priority-based access control • CPU Utilization: 90%+ CPU availability for other tasks • Cache Coherency: Memory consistency maintenance • Latency: Reduced interrupt overhead • Error Handling: Transfer failure recovery • Bandwidth: Full bus bandwidth utilization Security: Memory protection mechanisms **Advanced DMA Features** Scatter-Gather DMA **Chained DMA Bus Mastering** Transfers data to/from multiple non-Links multiple DMA transfers together Allows devices to become bus masters contiguous memory regions in a single using descriptor chains and initiate their own transfers operation Use Case: High-performance storage, **Use Case:** Audio/video streaming, large file transfers graphics processing Use Case: Network packet processing, file system operations

Audio: Sound card buffers

Video: Frame buffer updates USB: High-speed device

communication

Networking: Ethernet, WiFi **Graphics:** GPU memory

transfers

Real-World DMA Applications:

adapters

Storage: Disk I/O, SSD

controllers

