COMP7300 Fall 2021 HW03

Libo Sun lzs0101@auburn.edu
Compute Science and Software Engineering
Auburn University, Auburn, AL, U.S.

1. **At what point during an instruction cycle an external interrupt is recognized by the CPU? When more than one interrupt occurs, what approaches can you use to service them? What approaches should be taken when an interrupt occurs within an interrupt service routine?**

   - Because the external devices are much slower than the processor, if the processor must pause and wait for the external device's completion, therefore, an external interruption is recognized.
   - Two approaches can be taken to dealing with multiple interrupts:
     - **Disabled Interrupts:** The processor can and will ignore that interrupt request signal. If an interrupt occurs during this time, it generally remains pending and will be checked by the processor after the processor has enabled interrupts. Thus, when a user program is executing and an interrupt occurs, interrupts are disabled immediately. After the interrupt handler routine completes, interrupts are enabled before resuming the user program, and the processor checks to see if additional interrupts have occurred. This approach is nice and simple, as interrupts are handled in strict sequential order.
     - **Nested Interrupts:** This approach is to define priorities for interrupts and to allow an interrupt of higher priority to cause a lower-priority interrupt handler to be itself interrupted. Then current state of Interrupt Service Routine (ISR) for low priority device will be saved on stack and the processor executes the ISR of higher priority device. After completion, the processor executes the ISR of the current device. It can be happening multiple times and the execution sequence is only based on the priority.
   - The **Nested Interrupts** should be taken when an interrupt occurs within an ISR.

2. (a) **Discuss the merits and demerits of using a single bus.**
   - **Merits**:
     –Multiple devices can be connected by a shared transmission medium
     –A simple design and easy to add more devices to the bus
     –Low complexity and low costs
   - **Demerits**:
     –High latency
     –Low data rate

–Less scalability
–Only one component can transmit data at a time

(b) **How will the use of a mezzanine bus architecture solve this problem?**

- By building a high speed bus that is closely integrated with the rest of the system, a mezzanine architecture requirs only a bridge between the processor's bus and the high-speed bus.
- There is a local bus that connects the processor to a cache controller, which is in turn connected to a system bus that supports main memory. The cache controller is integrated into a bridge, or buffering device, that connects to the high-speed bus.
- This bus supports connections to high-speed LANs, such as Fast Ethernet at 100 Mbps, video and graphics workstation controllers, as well as interface controllers to local peripheral buses, including SCSI and FireWire.
- The high-speed bus arrangement specifically is designed to support high-capacity I/O devices. Lower-speed devices are still supported off an expansion bus, with an interface buffering traffic between the expansion bus and the high-speed bus.
- The advantage of this arrangement is that the high-speed bus brings high demand devices into closer integration with the processor and at the same time is independent of the processor. Thus, differences in processor and high-speed bus speeds and signal line definitions are tolerated. Changes in processor architecture do not affect the high-speed bus, and vice versa.

(c) **Discuss various methods of bus arbitration.**
Because all the components are exchanging data on a shared medium, it need arbitration to assure only one component transmit data at a time. There are two approaches to bus arbitration: Centralized and distributed.
By implementing either of distribution, a master device can manage the data transmission on the bus without collision.

- Centralized arbitration:
  - In centralized bus arbitration, a single bus arbiter performs the required arbitration. The bus arbiter may be the processor or a separate controller connected to the bus.
  - There are three different arbitration schemes that use the centralized bus arbitration approach. There schemes are:
    - ∗ Daisy chaining
    - ∗ Polling method
    - ∗ Independent request
- Distributed arbitration:
  - No single arbiter, all devices participate in the selection of the next bus master.
  - Each module may claim the bus, each device on the bus is assigned a4 bit identification number
  - When one or more devices request control of the bus, they assert the start arbitration signal and place their 4-bit identification numbers on arbitration lines through ARB3.
  - Each device compares the code and changes its bit position accordingly.

  - It does so by placing a 0 at the input of their drive.
  - The distributed arbitration is highly reliable because the bus operations are not dependant on devices.

3. **Consider a hypothetical microprocessor having 64-bit instructions composed of two fields: the first 16-bits contains the op-code and the remainder the immediate operand or an operand address. Assume memory is organized in 32-bit words, i.e. one r/w access can yield a maximum of 32 bits.**

   - **What is the maximum directly addressable memory capacity (in bytes)?**
     - Memory is 32-bits = 4 Bytes
     - Immediate address: 64-16 = 48 bits
     - Maximum Addressable memory capacity for 2^{48} bits:
       2^{48} bytes = **256 TB**
   - **How many bits are needed for the program counter and the instruction register?**
     - The addressable memory size(also operand address length) is 48 bits, hence, the program counter should be **48 bits** in order to have the possible memory address can be stored.
     - The instruction register stores the instruction which will be executed. Since the instruction size is 64 bits, it requires the instruction register must be **64 bits** as well.
   - **Discuss the impact on the system speed if the microprocessor has:**
     - A 64-bit local address bus and a 16-bit local data bus
       * Memory addressable : 48 bits
       * Instruction: 64 bits
       * Memory is organized : 32 bits
       * For reading or writing a 32 bits data from or to memory, it will require 2 circle to complete on a 16 bit data bus
       * For a 64-bit instruction, it will need **4 clock cycles** to get the instruction on a 16 bit data bus.
     - A 32-bit local address bus and a 64-bit local data bus.
       * for 48 bits address: **2 circle** to read or write with 32 bit address bus
       * 64-bit instruction: **1 clock cycle** to read it on 64-bit data bus
       * 32-bit data in memory: **1 clock cycle** on a 64-bit data bus/

4. **Consider a hypothetical microprocessor generating a 32-bit address (assume that the program counter and the address registers are 32-bit wide) and having a 32-bit data bus. Assume memory is byte addressable.**

   - **What is the maximum memory address space that the processor can access directly if connected to a "32-bit memory"?**
     Because it is 32bit address memory, and memory is byte addressable, hence, the directly addressable is 2^{32} Bytes = **4 GB**
   - **What is the maximum memory address space that the processor can access directly if connected to an "16-bit memory"?**
     Because it is 32bit address memory, and memory is byte addressable, hence, the directly addressable is 2^{32} Bytes = **4 GB**

- **What architectural features will allow this microprocessor to access a separate "I/O space"?**
  I/O devices have a separate address space from general memory. Separate I/O instructions are needed because during its execution will generate separate its own signals I/O signals. That signals will be different from the memory signals which is generated during the execution for memory instructions. Therefore, **one more output pin** will be needed to carry I/O signals.
- **If an input and an output instruction can specify a 16-bit I/O port number, how many 16-bit I/O ports can the microprocessor support? How many 32-bit I/O ports?**
  I/O port number is 16-bit, so it can have 2^{16} = **65536 addresses**.
  For 16-bit I/O ports, the microprocessor supports **65536 ports**
  For 32-bit I/O ports, the microprocessor supports the same **65536 ports** but at 32-bit data.

5. **Consider a microprocessor, with a 64-bit external data bus, driven by a 4 GHz input clock. Assume this microprocessor has a bus cycle whose minimum duration equals five (5) input clock cycles. What is the maximum data transfer rate that this microprocessor can sustain? To increase its performance, would it be better to make its external data bus 128-bits or to double the external clock frequency supplied to the microprocessor? Discuss and state any other assumptions you make, and explain. Hint: Determine the number of bytes that can be transferred per bus cycle.**

   - Maximum data transfer rate:
     - clock rate: 4 GHz
     - 5 cycles: 4/5 GHz = 0.8×10^9 Hz
     - 64-bits data bus: $64 \times 0.8 \times 10^9$ /8 = **6.4 GB/Sec**
   - Double the external data bus to 128-bits.
     Increase the data bus size from 64-bits to 128-bits, it doubles the data bits per time, thereby the throughput has doubled as **12.8 GB/Sec**. However, this improvement also will require the memory size to increase to 128-bits as well, otherwise the performance will not achieve.
   - Double the external clock frequency.
     The doubled clock frequency can absolutely double the performance theoretically at **12.8 GB/Sec**, because at the same time, the doubled clock frequency can execute twice times of instructions. But in this case, the bottle-neck is the memory speed. The processor require the memory speed can be faster to catch up the doubled clock frequency. If not, the processor will have to wait for memory accessing.

6. **Discuss with a diagram the operation of a synchronous read/write from/to memory. Look up the web for synchronous r/w operation.**

   With synchronous timing, the occurrence of events on the bus is determined by a clock. The bus includes a clock line upon which a clock transmits a regular sequence of alternating 1s and 0s of equal duration. A single 1–0 transmission is referred to as a clock cycle or bus cycle and defines a time slot. All other devices on the bus can read the clock line, and all events start at the beginning of a clock cycle.

In this simple example, the processor places a memory address on the address lines during the first clock cycle and may assert various status lines. Once the address lines have stabilized, the processor issues an address enable signal. For a read operation, the processor issues a read command at the start of the second cycle. A memory module recognizes the address and, after a delay of one cycle, places the data on the data lines. The processor reads the data from the data lines and drops the read signal. For a write operation, the processor puts the data on the data lines at the start of the second cycle, and issues a write command after the data lines have stabilized. The memory module copies the information from the data lines during the third clock cycle.

Below Figure 1 shows the timing of synchronous bus operations.

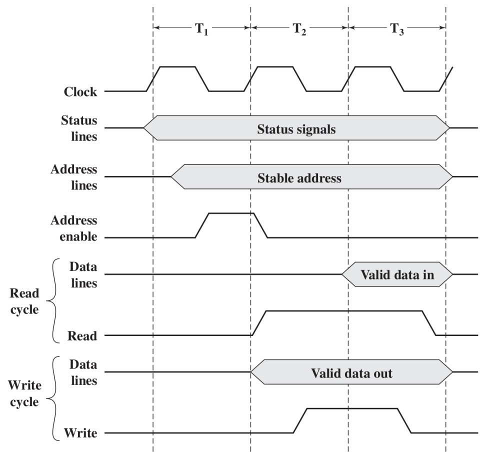

Figure 1: Timing of Synchronous Bus Operations

7. **Figure 2 (see below) indicates a distributed arbitration scheme that can be used with an obsolete bus scheme known as Multibus I. Agents are daisy cha-ined physically in priority order. The left-most agent in the diagram receives a constant bus priority in (BPRN) signal indicating that no higher-priority agent desires the bus. If the agent does not re- quire the bus, it asserts its bus priority out (BPRO) line. At the beginning of a clock cyce, any agent can request control of the bus by lowering its BPRO line. This lowers the BPRN line of the next agent in the chain, which is in turn required to lower its BPRO line. Thus, the signal is propagated the length of the chain. At the end of this chain reaction, there should be only one agent whose BPRN is asserted and whose BPRO is not. This agent has priority. If, at the beginning of a bus cycle, the bus is not busy (BUSY inactive), the agent that has priority may seize control of the bus by asserting the BUSY line.**

   It takes a certain amount of time for the BPR signal to propagate from the highest-priority agent to the lowest. Must this time be less than the clock cycle? Explain.

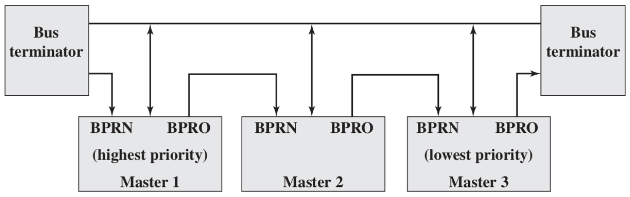

Figure 3.26 Multibus I Distributed Arbitration

Figure 2: Distributed Arbitration

- It needs a mount of time for propagating the BPR signal from the highest-priority agent to the lowest
- If the propagating is not in a suitable time range, a lower priority agent may claim the control when it doesn't know any higher priority has already claimed therefore a collision might occur. e.g., if the propagating time will be 2 clock cycles, there can be two agents try to control the bus and send the data because at the beginning of the two cycles, two agents will seize control of the bus by asserting the BUSY line accordingly.
- In order to avoid collision, the propagating must be done within one clock cycle.