CIT602 Computer Architecture (4 CH) – Week 4 Notes

Course: Computer Architecture

Credit Hours: 4 CH

Course Outline: Number systems, logic gates, digital circuits, combinational and sequential logic, CPU architecture, and memory systems, with applications in IT. Beginner course for Year 1, Semester 1 students, providing a foundation for Year 1 programming, computer systems, and later IT courses (e.g., Computer Security, Software Engineering).

Assessment: Labs/assignments (30%), midterm (30%), final exam/project (40%).

Resources:

- *Computer Organization and Architecture* by William Stallings (Chapter 4 for sequential logic).
- *Digital Design and Computer Architecture* by David Harris and Sarah Harris (Chapter 3 for sequential logic).
- Online tutorials: Search "Sequential Logic" or "Flip-Flops" on Coursera, YouTube (e.g., Neso Academy, Ben Eater), or Khan Academy.
- Tools: Logisim (free for circuit simulation), Multisim (for circuit design, trial available), Canva (free for diagrams), Google Docs (for documentation), GitHub (for project portfolio).

  Week 4 Topic: Sequential Logic

  Objective: Understand and apply sequential logic principles to design circuits with memory and state, critical for computer architecture in IT systems.

---

1. Recap of Previous Weeks

- **Week 1**: **Number Systems & Binary Arithmetic** covered binary, hexadecimal, and arithmetic operations (e.g., addition, subtraction).
- **Week 2**: **Logic Gates and Digital Circuits** explored basic gates (AND, OR, NOT) and simple circuits.
- **Week 3**: **Combinational Logic** introduced circuits like adders and multiplexers, where outputs depend only on current inputs.
- **Examples**: Designing a half-adder (Week 3), converting binary to decimal (Week 1).
- **Week 4 Focus**: Build on Week 3 by exploring sequential logic, where outputs depend on current inputs and past states, essential for memory and processing in computer architecture.

---

2. Overview of Sequential Logic

Sequential logic refers to digital circuits where outputs depend on both current inputs and previous states (memory), unlike combinational logic (Week 3). It is foundational for components like registers and counters in IT systems.

- **Core Idea**: Sequential logic circuits use memory elements (e.g., flip-flops) to store past states, enabling functions like counting or data storage. For example, a counter circuit tracks events by remembering previous counts.
- **Importance**:
    - Enables memory and state-based operations in computers (e.g., registers in CPUs).
    - Builds on combinational logic (Week 3) and logic gates (Week 2).
    - Prepares students for CPU architecture and memory systems in Year 1, Semester 2.
- **Real-World Applications**:
    - Designing a counter for a digital clock.
    - Creating registers for CPU data storage.
    - Building memory units for simple processors.

---

3. Key Concepts in Sequential Logic

This section details sequential logic components, their design, and applications in IT, minimizing code snippets (e.g., Verilog only if essential) and focusing on diagrams and truth tables.

3.1 Sequential Logic Basics

- **Definition**: Sequential logic circuits have outputs that depend on current inputs and stored states, using memory elements like flip-flops.
- **Key Features**:
    - **Memory Elements**: Store past states (e.g., flip-flops, latches).
    - **Clock Signals**: Synchronize state changes (e.g., clock pulses in CPUs).
    - **State**: Represents stored data (e.g., binary 1 or 0).
- **Types**:
    - **Synchronous**: Uses a clock to time operations (e.g., flip-flops).
    - **Asynchronous**: No clock, responds to input changes (e.g., latches).
- **Use Cases**:
    - Storing data in a CPU register.
    - Counting events in a digital system.
- **Advantages**:
    - Enables memory and sequential operations in computers.
    - Supports complex systems like processors (Week 2).
- **Disadvantages**:

  - More complex than combinational logic (Week 3).
  - Requires precise timing (clock signals).
- **Example**: Simple counter circuit.
  - **Scenario**: Design a circuit to count button presses.
  - **Process**: Use a flip-flop to store count, increment with each press, simulate in Logisim.
  - **Outcome**: Functional counter with memory.
- **Support Tasks**: Simulate a counter in Logisim, draw a state diagram.

3.2 Flip-Flops and Latches

- **Definition**: Flip-flops and latches are memory elements storing one bit of data, the building blocks of sequential logic.
- **Types**:
  - **SR Latch**: Stores state using Set/Reset inputs (built with NOR gates, Week 2).
  - **D Flip-Flop**: Stores one bit, updates on clock signal (synchronous).
  - **JK Flip-Flop**: Versatile, toggles state based on inputs.
- **Key Features**:
  - **Inputs**: Control state (e.g., D input sets data).
  - **Outputs**: Q (current state), Q' (complement).
  - **Clock**: Synchronizes updates in flip-flops.
- **Use Cases**:
  - Storing a bit in a CPU register.
  - Toggling states in a counter.
- **Advantages**:
  - Reliable storage for sequential operations.
  - Supports synchronous circuit design.
- **Disadvantages**:
  - Requires careful clock management.
  - Complex to design for beginners.
- **Example**: D flip-flop for data storage.
  - **Scenario**: Store a bit in a register.
  - **Process**: Use a D flip-flop, input data (0 or 1), update on clock pulse, simulate in Logisim.

    - **Outcome**: Reliable bit storage.
- **Support Tasks**: Build a flip-flop circuit, create a truth table.

3.3 Sequential Logic Circuits

- **Definition**: Sequential logic circuits combine flip-flops and combinational logic (Week 3) to perform tasks like counting or state machines.
- **Examples**:
    - **Registers**: Store multiple bits (e.g., 8-bit register using D flip-flops).
    - **Counters**: Track events (e.g., binary counter).
    - **Finite State Machines (FSMs)**: Control sequences (e.g., traffic light controller).
- **Use Cases**:
    - Building a 4-bit counter for a timer.
    - Designing a simple state machine for a vending machine.
- **Advantages**:
    - Enables complex, state-based operations.
    - Critical for CPU and memory design (Year 1, Semester 2).
- **Disadvantages**:
    - Requires precise design and timing.
    - Debugging is challenging.
- **Example**: 4-bit binary counter.
    - **Scenario**: Design a counter for 0–15 counts.
    - **Process**: Use JK flip-flops, connect to increment on clock pulses, simulate in Logisim.
    - **Outcome**: Functional counter for digital systems.
- **Support Tasks**: Simulate a counter, draw a state transition diagram.

3.4 Integration with Computer Architecture

- **Definition**: Sequential logic integrates with combinational logic (Week 3) and logic gates (Week 2) to form core computer components like CPUs and memory.
- **How It Works**:
    - **Flip-Flops**: Store data in registers (e.g., CPU state).
    - **Counters**: Track cycles or events in processors.
    - **FSMs**: Control operations (e.g., instruction execution).
- **Example**: CPU register design.
    - **Sequential Logic**: Use D flip-flops for an 8-bit register.

    - **Combinational Logic**: Add logic gates for data input (Week 3).
    - **Integration**: Store and process data in a CPU (Week 2).
- **Support Tasks**: Design a simple register, simulate in Logisim.

3.5 Common Challenges

- **Timing Issues**: Incorrect clock signals cause errors.
    - Solution: Use synchronous design, verify clock in Logisim.
- **State Errors**: Flip-flops enter invalid states.
    - Solution: Use proper inputs, check truth tables.
- **Complexity**: Sequential circuits are hard to design.
    - Solution: Start with simple flip-flops, simulate in Logisim.
- **Debugging**: Hard to trace errors in sequential logic.
    - Solution: Use simulation tools, test step-by-step.

Visual (Text-Based Diagram):

text

```
[Sequential Logic]
  |--> [Basics: Memory Elements, Clock, State]
  |--> [Flip-Flops/Latches: SR, D, JK]
  |--> [Circuits: Registers, Counters, FSMs]
  |--> [Integration: Combinational Logic, CPU, Memory]
  |--> [Challenges: Timing, State Errors, Complexity, Debugging]
```

---

4. Applications in IT

Sequential logic is critical for:

- **CPU Design**: Storing data in registers.
    - Example: 8-bit register for a simple processor.
- **Digital Systems**: Counting events in timers or clocks.
    - Example: Binary counter for a digital clock.
- **Control Systems**: Managing sequences in FSMs.
    - Example: Traffic light controller.
- **Future IT Tasks**: Supporting processor design and memory systems (Year 1, Semester 2).
    - Example: Building a basic CPU.

5. Practical Examples

Example 1: D Flip-Flop Register

- **Scenario**: Design a 4-bit register to store data.
- **Process**:
    - Use four D flip-flops, connect data inputs, clock signal.
    - Simulate in Logisim, test with inputs (e.g., 1010).
    - Troubleshoot: Verify clock signal, check outputs.
- **Concepts**: Flip-flops, synchronous logic.
- **Outcome**: Functional register for data storage.
- **Tool**: Use Logisim for simulation, Canva for circuit diagram.

Example 2: Binary Counter

- **Scenario**: Build a 3-bit counter (0–7).
- **Process**:
    - Use JK flip-flops, connect to increment on clock pulses.
    - Simulate in Logisim, test counting sequence.
    - Troubleshoot: Check flip-flop connections, verify counts.
- **Concepts**: Counters, sequential circuits.
- **Outcome**: Working counter for digital systems.
- **Tool**: Use Logisim for simulation, Canva for state diagram.

Example 3: Troubleshooting FSM

- **Scenario**: Fix a traffic light controller FSM.
- **Process**:
    - Design FSM with states (Red, Green, Yellow) using flip-flops.
    - Simulate in Logisim, test state transitions.
    - Troubleshoot: Correct invalid states, adjust clock timing.
- **Concepts**: Finite state machines, timing.
- **Outcome**: Functional traffic light controller.
- **Tool**: Use Logisim for simulation, Canva for state transition diagram.

---

6. In-Class and Self-Study Exercises

In-Class Exercises:

1. **Concept Analysis**:
    - In groups, list 3 sequential logic concepts (e.g., flip-flops, counters) and their roles in IT. Write a 50-word explanation.
2. **Tool Exploration**:
    - In pairs, explore Logisim for simulating a flip-flop. Discuss its use (50 words).
3. **Scenario Discussion**:
    - Discuss a sequential logic scenario (e.g., counter design). Write a 50-word summary.

Self-Study Exercises:

1. **Scenario Analysis**:
    - Write a 150-word description of a sequential logic scenario (e.g., register design).
2. **Concept Research**:
    - Research one concept (e.g., D flip-flop). Write a 100-word summary of its role.
3. **Tool Practice**:
    - Simulate a counter in Logisim or draw a truth table. Write a 100-word reflection.
4. **Diagram Design**:
    - Create a sequential logic diagram in Canva (e.g., flip-flop circuit). Write a 100-word explanation.

---

7. Week 4 Assignment (Contributes to Labs/Assignments – 30%)

Task: Create a Sequential Logic Analysis with Diagram and Report

- **Description**:
    - Analyze an IT sequential logic scenario (e.g., designing a 4-bit counter) using sequential logic concepts, tested with tools like Logisim.
    - Include:
        - Two concepts (e.g., flip-flops, counters).
        - Examples of their application (e.g., storing data, counting events).
        - A sequential logic diagram in Canva (e.g., counter circuit).
    - Write a **200-word explanation** of the scenario, concepts, tools, and their significance in IT.
    - Include a handwritten note (scanned/photographed) with your name and a 2-sentence summary of the analysis's purpose.
- **AI-Proof Measures**:

  - Handwritten note ensures originality.
  - Explanation must reflect personal understanding of sequential logic concepts.
- **Learning Outcomes**:
  - Reinforces understanding of sequential logic.
  - Develops skills in designing IT circuits.
- **Portfolio Value**:
  - Save the analysis, Logisim screenshot, diagram, explanation, and note as a PDF in a GitHub repository with a README (e.g., "Counter Circuit Analysis").
  - The explanation demonstrates computer architecture skills for academic or job applications.
- **Submission Format**:
  - PDF (analysis details + Logisim screenshot + diagram + 200-word explanation + handwritten note photo, 25% of lab/assignment grade).
  - Optional: MP3 (1-minute voice summary, 5% of lab/assignment grade, if required).
- **Grading Rubric**:
  - 40% Technical Accuracy: Correct application of sequential logic concepts.
  - 30% Clarity: Clear diagram, structure, and explanation.
  - 20% Originality: Unique scenario and handwritten note.
  - 10% Presentation: Neat diagram and readable text.
- **Tips**:
  - Use Logisim for simulation, Canva for diagrams, Google Docs for documentation.
  - Test your understanding with the counter example.
  - Ensure the handwritten note is legible and includes your name.

Sample Assignment Deliverable

4-Bit Counter Analysis Scenario: Design a 4-bit counter for a digital timer. **Concepts**:

1. **JK Flip-Flop**: Toggles state for counting events.
2. **Counter**: Tracks sequence using sequential logic. **Implementation**:

- **Flip-Flops**: Use JK flip-flops, connect for 0–15 counts.
- **Counter**: Simulate in Logisim, test counting sequence.
- **Troubleshoot**: Verify connections, check clock signal. **Tools**:
- **Logisim**: Simulate counter.

- **Canva**: Create counter circuit diagram. **Significance in IT**: Sequential logic enables counters and registers, critical for CPU and digital systems. **Diagram**: [Canva diagram showing JK flip-flops → counter → output workflow]. **Handwritten Note**: [Photo with name and summary: "This analysis designs a 4-bit counter using JK flip-flops for IT systems."]

---

9. Connection to Course Assessments

- **Labs/Assignments (30%)**: The Week 4 assignment contributes to the 30% lab/assignment component by teaching sequential logic design. Future assignments will involve CPU or memory design.
- **Midterm (30%)**: The Week 4 quiz prepares students for midterm questions on logic circuits. Later midterm tasks will include analyzing circuit designs.
- **Final Exam/Project (40%)**: Week 4 concepts (sequential logic) will inform the development of simple processor projects.

---

10. Glossary of Key Terms

- **Sequential Logic**: Circuits with outputs depending on inputs and past states.
- **Flip-Flop**: Memory element storing one bit (e.g., D, JK).
- **Latch**: Asynchronous memory element (e.g., SR latch).
- **Clock Signal**: Synchronizes sequential logic operations.
- **Counter**: Circuit tracking events (e.g., binary counter).
- **Register**: Stores multiple bits using flip-flops.
- **Finite State Machine (FSM)**: Controls sequences with states.
- **Synchronous**: Logic timed by a clock signal.
- **Asynchronous**: Logic without a clock.
- **State**: Stored data in a sequential circuit.

---

11. Frequently Asked Questions

- **What's the difference between sequential and combinational logic?**
  - Sequential logic uses memory for past states; combinational logic depends only on current inputs (Week 3).
- **How do I design a flip-flop circuit?**
  - Use Logisim, connect inputs and clock. Review the D flip-flop example.
- **What if my counter doesn't work?**
  - Check flip-flop connections, verify clock. Test with the counter example.

- **How do I prepare for the quiz?**
    - Study these notes, referenced textbook chapters, and simulate in Logisim.
- **Why is sequential logic important in IT?**
    - It enables memory and state-based operations in computers.

---

12. Troubleshooting Tips

- **Problem**: Counter skips counts.
    - **Solution**: Verify flip-flop connections, check clock signal. Review the counter example.
- **Problem**: Flip-flop enters invalid state.
    - **Solution**: Check inputs, use truth tables. Test with the D flip-flop example.
- **Problem**: My explanation is too short.
    - **Solution**: Describe the scenario, concepts, tools, and impact (e.g., "Counters enable timing"). Use the FSM example.
- **Problem**: I'm stuck on the application question.
    - **Solution**: Think of register or counter scenarios. Explain how sequential logic supports IT systems.

---

13. Self-Study and Portfolio Tips

- **Practice**:
    - Simulate a flip-flop or counter in Logisim.
    - Create a sequential logic diagram in Canva.
    - Analyze a circuit scenario (e.g., register design).
- **Resources**:
    - Read Chapter 4 of *Computer Organization and Architecture* and Chapter 3 of *Digital Design and Computer Architecture*.
    - Watch Neso Academy's "Sequential Logic" on YouTube.
    - Explore Khan Academy's digital logic resources.
    - Use Logisim documentation for simulation.
- **Portfolio**:
    - Create a GitHub repository for architecture assignments.
    - Upload the analysis, Logisim screenshot, diagram, explanation, and note as a PDF with a README (e.g., "Sequential Logic Analysis").

    - Include a Logisim screenshot to show circuit design skills.
- **Applications**: Consider how sequential logic supports IT tasks like CPU design or timing to appreciate its impact.

---

14. Additional Notes for Success

- **Engagement**: Participate in class discussions to clarify sequential logic concepts. Ask questions during tutorials to deepen understanding.
- **Practice**: Simulate circuits in Logisim to build confidence.
- **Preparation for Future Weeks**: Week 4 prepares students for CPU architecture (Week 5) and memory systems (Week 6). Mastering sequential logic now will simplify later topics.
- **Portfolio Building**: Treat the analysis and diagram as professional artifacts. A clear design and thorough explanation can showcase computer architecture skills to academic programs