Birzeit University

Department of Electrical & Computer Engineering

Summer - 2024/2025

ENCS5337 Chip Design Verification

Dr. Ayman Hroub

Course Project

Design and Verification of a Simplified Interrupt Controller Using SystemVerilog/UVM

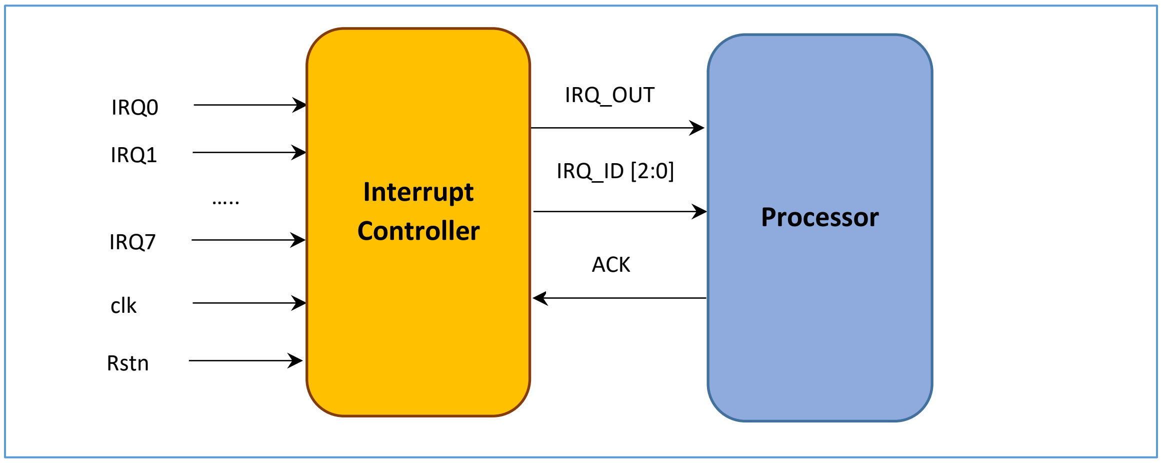
Deadline August 8, 2025

1. Project Objectives

1. A simple interrupt controller design using SystemVerilog
2. Developing a comprehensive verification plan of this interrupt controller
3. UVM based design verification environment of this interrupt controller

2. Interrupt Controller Specifications

An interrupt controller is a hardware unit that receives external interrupts and generates an interrupt request (IRQ) to the processor. In this project, you will design an interrupt controller that receives eight external interrupt requests (IRQ0 to IRQ7) from different peripherals, such as, timer, UART, and sensor. Then the interrupt controller discovers the highest priority request, in the form of an interrupt request ID, and sends it to the processor for servicing as shown in the block diagram below.



Input Ports:

- **IRQ0 to IRQ7** represent the interrupt request signals from different eight peripherals. IRQ0 has the highest priority while IRQ7 has the lowest priority. Use priority encoder to implement the prioritization, and picks the unmasked interrupt with the highest priority to send to the processor.
- **Rstn** is asynchronous active low reset signal. It clears all interrupt requests and internal registers inside the interrupt controller.
- **ACK** is the CPU acknowledge signal after receiving an interrupt. When the interrupt controller receives it, it clears the interrupt pending bit.

Output Ports

- **IRQ_OUT** is the global interrupt output
- **IRQ_ID** is the ID of the highest-priority active, unmasked interrupt (0–7)

Internal Registers

- **Pending Register** is an 8-bit register, with each bit corresponding to one external peripheral interrupt request line. When a peripheral asserts its interrupt line, the associated bit in the Pending Register is set to indicate that an interrupt request is pending.

- **Mask Register** is an 8-bit register, with each bit corresponding to one external peripheral interrupt request line. It allows the CPU to temporarily disable specific interrupts, even if they are pending. A bit value of 1 indicates that the corresponding interrupt is masked (disabled), while a value of 0 means the interrupt is unmasked (enabled). The Mask Register is configured and maintained by the CPU through software. In the verification environment, you can assign random values to the Mask Register to test the interrupt controller's behavior under various masking scenarios.

3. Team Work

You can work on this project in teams of up to three students. Teams exceeding three members will not be permitted. All team members are expected to contribute equally and participate actively in all phases of the project, including design, implementation, verification, and documentation.

4. Project Deliverables

1. RTL Design SystemVerilog Source Code
2. UVM TestBench Source Code
3. Golden Reference Model which could be a simple SystemVerilog function in the scoreboard
4. Report, which includes the following components in a single document:
    - Design and implementation detailed description and block diagrams
    - Detailed verification plan
    - Simulation snapshots for all tests
    - Coverage report

5. Submission Guidelines

- Submissions must be made by replying to the assignment on Ritaj. Submissions through any other means will not be accepted.
- Late submissions will not be accepted under any circumstances.
- Only one team member should submit the project, i.e., there must be one submission per team only.
- The full names and university IDs of all team members must be included both in the message body of the submission and on the cover page of the report.
- The submission must include the project report and source code, packaged together in a single compressed folder.

6. Grading Criteria

Evaluation Item	**Weight (%)**
RTL design	25%
Complete UVM-based Testbench (including functional coverage and golden reference model)	35%
Verification plan	15%
The rest of the report	10%
The way of discussion and answering questions	10%
Code organization and documentation	5%
Total	**100%**