Assignment Solution:-

1. The verification Plan: -

Planning an exhaustive plan was a challenging task, A simple program interrupt controller that is Simple PIC will be behaving different under different circumstances.
The following testcases have been taken into role for making sure that the PIC behaviour can be checked accordingly

Test Case	Objective	Expected outcome
Reset Verification	Making ensure that all internal registers are correctly initialized after getting reset	Register like **edgen, pol & mask** should be initialized correctly as follows (**edgen = 0x00, pol = 0x00, mask = 0xFF**)
Write & Read Registers	Checking if registers are being written and read correctly	A written test value should be given i.e., stored and read back
Level-Sensitive Interrupt Request (IRQ)	Verifying the level-sensitive IRQs are detected and setting them into pending register	If an interrupt is asserted, it should be seen in pending register until it's completed.
Edge-Sensitive IRQ Handling	Verify that edge-sensitive IRQs are detected correctly	If an IRQ signal toggles (0 → 1 → 0), it should trigger an interrupt in the pending register
Clear Pending IRQs	Checking if pending interrupts can be cleared via register write.	Writing '1' to a bit in the 'pending' register should clear that interrupt
Interrupt Output Behavior	Ensure that "int_o" is asserted correctly when an enabled interrupt occurs.	"int_o" should be high if an unmasked pending interrupt is present.

2a. Directed Test Vectors (Verilog Testbench Implementation)

The testbench **(simple_pic_tb)** performs directed testing based on the verification plan:

Testbench Breakdown:

1. **Instantiation of the Device Under Test (DUT)**
   - The **Simple Interrupt Controller** module is instantiated using simple_pic uut(...) with appropriate connections.
2. **Clock Generation**
   - The clock signal (clk_i) toggles every 5-time units (#5 clk_i = ~clk_i;).

3. **Initial Setup & Reset Verification (TC_01)**
   - Signals are initialized, and a reset pulse is generated (rst_i = 1;).
   - Registers (edgen, pol, mask) are checked for correct initialization.
4. **Register Write & Read Test (TC_02)**
   - A value (dat_i = 8'b00001111) is written to address adr_i = 2'b00 (e.g., edgen register).
   - The value is read back and compared to verify correct register behavior.
5. **Level-Sensitive IRQ Handling (TC_03)**
   - An IRQ is asserted (irq = 8'b00000001).
   - The pending register should reflect this (uut.pending[1] == 1).
6. **Edge-Sensitive IRQ Handling (TC_04)**
   - The IRQ signal is toggled (0 → 1 → 0) to simulate an edge-sensitive interrupt.
   - The pending register should correctly capture the event.
7. **Clearing Pending IRQs (TC_05)**
   - The pending register is cleared by writing 1 to the corresponding bit.
   - The test verifies that the pending interrupt is successfully cleared.
8. **Interrupt Output (int_o) Behavior (TC_06)**
   - An IRQ is asserted (irq = 8'b00000100), and the int_o signal is checked.
   - If an interrupt is active, int_o should be set (1).
9. **Waveform Dumping ($dumpfile) for Debugging**
   - The simulation data is stored in a .vcd file for further analysis.

2b. The testbench code for “simple program interrupt controller”

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////
//////////
// Name: Amit
// Roll No: KVLSI2501033
// College:- Govt. S.K.S.J Technological Institute
// Create Date: 29.03.2025 10:25:53
// Design Name: Testbench code for Simple PIC
// Module Name: simple_pic_tb
// Project Name: Simple PIC testbench code
// Target Devices: Simulated device in Xilinx Vivado
// Tool Versions: 2024.2
//
```