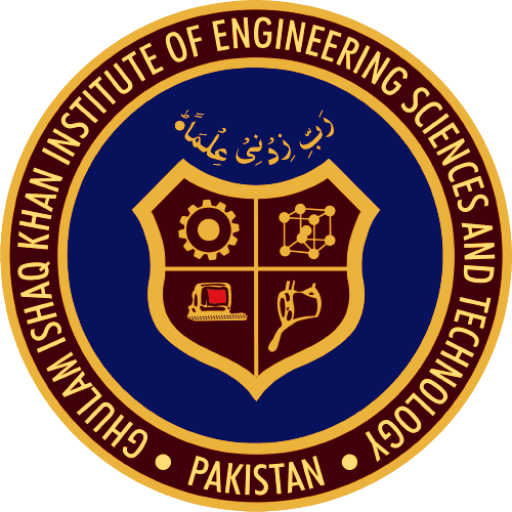
**CE222L – Project Proposal  
Enhanced SAP-1 Computer Using Verilog**



**Section - C**

**Faculty – CS**

|  |  |
| --- | --- |
| 2023330 | MOHAMMAD MUSA ALI |
| 2023640 | SAROSH ISHAQ |
| 2023538 | MUHAMMAD UMER |

**🔹 Project Description:**

This project aims to design, simulate, and analyze an enhanced version of the SAP-1 (Simple-As-Possible) computer architecture using Verilog HDL. The original SAP-1 includes basic functionality for instruction fetch, decode, and execute with a limited instruction set. In this project, we will extend its capabilities by introducing new instructions such as bitwise logic operations (AND, OR XOR) and conditional branching (JMP, JZ).

Each component of the SAP-1 (like the program counter, memory, instruction register, accumulator, ALU, and control unit) will be modularly implemented in Verilog. The simulation will demonstrate instruction-by-instruction execution using testbenches, and optionally a graphical frontend interface will be provided for visual demonstration.

**🔹 Goals and Objectives:**

* Understand and implement core digital components of a basic computer architecture.
* Develop Verilog modules for each SAP-1 block and interconnect them.
* Expand the instruction set to include logical operations and conditional control flow.
* Simulate instruction execution using testbenches and visualize waveform outputs.
* Document design decisions, implementation steps, and test results comprehensively.

**🔹 Technical Requirements and Tools:**

| **Requirement** | **Tool/Resource** |
| --- | --- |
| HDL Design | Verilog |
| Simulation | Vivado |
| Waveform Viewing | Vivado |
| Visualization (Optional) | HTML/CSS/JS frontend (SAP-1 UI Simulator) |
| Documentation | MS Word |

**🔹 Expected Outcomes:**

* A complete and functional SAP-1 architecture implemented in Verilog.
* Additional instructions like AND, OR XOR, JMP, and JZ are integrated into the processor.
* Simulated testbenches for programs running on the enhanced SAP-1
* Clear visualization of internal signal states through waveform tools
* A structured and well-documented final report and presentation

**🔹 Implementation Overview:**

1. **Instruction Set Architecture (ISA):**

| **Opcode** | **Mnemonic** | **Function** |
| --- | --- | --- |
| 0000 | LDA | Load value from memory |
| 0001 | ADD | Add value to accumulator |
| 0010 | SUB | Subtract value from accumulator |
| 0011 | AND | Bitwise AND with accumulator |
| 0100 | OR | Bitwise OR with accumulator |
| 0101 | XOR | Bitwise XOR with accumulator |
| 0110 | JMP | Jump to memory address |
| 0111 | JZ | Jump if accumulator is zero |
| 1110 | OUT | Output accumulator value |
| 1111 | HLT | Halt execution |

1. **Major Modules:**
   * program\_counter.v
   * memory.v
   * mar.v
   * instruction\_register.v
   * accumulator.v
   * b\_register.v
   * alu.v (with logic ops)
   * control\_unit.v (FSM)
   * output\_register.v
2. **Testing & Verification:**
   * Verilog testbenches simulating different instruction sequences.