Mid-term Group Defense Self Statement Document

1. Team work

|  |  |  |  |
| --- | --- | --- | --- |
| Name | SID | Lab\_class time | work（not MUST but suggested） |
| Sean Sovann | 12312526 |  |  |
| Wai Yan Kyaw | 12312638 |  | IO |
|  |  |  |  |

1. Code Specification: Structured Design（YES）

**Naming conventions**:

* Module names use PascalCase
* Signal names use camelCase
* Constants use UPPER\_SNAKE\_CASE
* Parameters use UPPER\_SNAKE\_CASE

**Annotation requirements**:

* Input/output ports are annotated with detailed descriptions
* Each component of code has a short description comment above
* Address mappings are fully documented with clear explanations

**Definition and use of symbolic constants**:

* I/O address ranges defined as constants (e.g., IO\_ADDRESS\_HIGH = 22'h3FFFFF)
* Peripheral-specific address ranges defined as localparam (e.g., SWITCH\_ADDR\_RANGE\_START = 8'h00)
* ALU operation codes defined as localparam (e.g., ALU\_ADD = 5'b00010)
* Display address mapping constants (e.g., NORMAL\_DISPLAY\_ADDR = 8'h60)
* Clock divider parameters (COUNTER\_MAX = 150000000)

3. CPU characteristics

**- CPU Type**: Single cycle RISC-V CPU

- **CPU Clock**: 100 MHz (configurable through clock wizard)

- **ISA**: RISC-V (32-bit, RV32I subset)

- **Storage scheme**: Harvard architecture (separate instruction and data memory)

- **I/O scheme**: Memory-Mapped I/O (MMIO)

The instruction set to be implemented in this major assignment (please list):

* R-type: add, sub, and, or, xor, sll, srl, sra, slt, sltu
* I-type (arithmetic): addi, andi, ori, xori, slti, sltiu, slli, srli, srai
* I-type (load): lw, lb, lbu
* S-type: sw
* B-type: beq, bne, blt, bge, bltu, bgeu
* J-type: jal
* I-type (jump): jalr
* U-type: lui

1. CPU architecture design (internal modules and wiring relationships)/interface design description

The single-cycle CPU architecture consists of several interconnected modules:

**Core Modules:**

* **CPU (Top Module)**: Integrates all components and provides external interface
* **IFetch**: Handles instruction fetch, PC update, and branch/jump control
* **Decoder**: Decodes instructions, manages register file, and generates immediate values
* **ALU**: Performs arithmetic and logical operations
* **Controller**: Generates control signals based on instruction type
* **DataMem**: Handles memory access for data storage and retrieval

**I/O Modules:**

* **MemOrIO**: Routes data between CPU and memory/I/O devices
* **Switch**: Interfaces with physical switches and buttons
* **Led**: Controls LED outputs
* **Tube**: Manages 7-segment display

**Clock System:**

* **Clock**: Generates clock signals for CPU and peripherals
* **ClockDivider**: Provides slower clock for display sequencing

**Internal Connections:**

* Instruction path: IFetch → Decoder → Controller → ALU → IFetch (for PC updates)
* Data path: Decoder → ALU → MemOrIO → Memory/I/O devices → Decoder
* Control signals: Controller → all modules

**External Interface:**

* Inputs: 16 DIP switches, 4 push buttons, FPGA clock, reset
* Outputs: 16 LEDs, 7-segment display (8 digits)

1. The relationship between CPU instructions and control signals (refer to the table below)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Instructoin | ALUOp | ALUsrc | MemRead | MemWrite | RegWrite | MemOrIOtoReg | Jump/Branch |
| lw | 00 | 1 | 1 | 0 | 1 | 1 | 0 |
| lb | 00 | 1 | 1 | 0 | 1 | 1 | 0 |
| lbu | 00 | 1 | 1 | 0 | 1 | 1 | 0 |
| sw | 00 | 1 | 0 | 1 | 0 | 0 | 0 |
| add | 10 | 0 | 0 | 0 | 1 | 0 | 0 |
| sub | 10 | 0 | 0 | 0 | 1 | 0 | 0 |
| slt | 10 | 0 | 0 | 0 | 1 | 0 | 0 |
| sltu | 10 | 0 | 0 | 0 | 1 | 0 | 0 |
| and | 10 | 0 | 0 | 0 | 1 | 0 | 0 |
| or | 10 | 0 | 0 | 0 | 1 | 0 | 0 |
| xor | 10 | 0 | 0 | 0 | 1 | 0 | 0 |
| sll/srl/sra | 10 | 0 | 0 | 0 | 1 | 0 | 0 |
| addi | 11 | 1 | 0 | 0 | 1 | 0 | 0 |
| slti/sltiu | 11 | 1 | 0 | 0 | 1 | 0 | 0 |
| andi/ori/xori | 11 | 1 | 0 | 0 | 1 | 0 | 0 |
| slli/srli/srai | 11 | 1 | 0 | 0 | 1 | 0 | 0 |
| beq/bne | 01 | 0 | 0 | 0 | 0 | 0 | Conditional |
| blt/bge | 01 | 0 | 0 | 0 | 0 | 0 | Conditional |
| bltu/bgeu | 01 | 0 | 0 | 0 | 0 | 0 | Conditional |
| jal | 00 | 1 | 0 | 0 | 1 | 0 | 1 |
| jalr | 00 | 1 | 0 | 0 | 1 | 0 | 1 |
| lui | 00 | 1 | 0 | 0 | 1 | 0 | 0 |

1. Implementation status of the project, completed project codes

|  |  |  |
| --- | --- | --- |
| Items | Completion percentage | Remarks (description on the work to be completed/resolved) |
| document | 50% | Only mid term defense document is finished. Final document needs to finish as well. |
| CPU sub-module design | 100% | All individual CPU modules implemented and tested |
| CPU top design | 70% | Top-level CPU module implemented with proper interconnections |
| CPU unit testing | 70% | Most unit tests complete; some edge cases still being tested |
| CPU Integration testing | 70% | Base CPU functionality verified |
| IO design | 70% | Memory-mapped I/O system partially implemented with Switch, LED, and 7-segment display |
| IO testing | 70% | Some test benches created for some I/O modules |
| Loading program | 10% | Basic program loading works; improving support for larger programs |
| Tools chain | 50% | Basic RISC-V assembly to machine code flow established; expanding automation |
| Basic Test Scenario 1 | 20% | Only 3 basic testcases are tested |
| Basic Test Scenario 2 | 0% |  |
| Additional testing scenarios | 0% | - |

1. Planned tool chain for use/development

Development Environment: Vivado 2016 for RTL design and synthesis

Hardware Platform: FPGA Development Board (Xilix Artix 7 ECO 1 board)

Simulation Tools: Vivado Simulator

Assembly Tools: RARS for assembly code development

Testing Tools: Custom test bench framework for both unit and integration testing

Version Control: Git for code management and collaboration

Documentation: Microsoft Word and Typora

1. Overall project progress - 70%
2. Current difficulties or issues – IO Mapping
3. Expected final defense time – Week-15
4. Follow-up plans – Left Basic Test case scenarios, Advanced test case scenarios, UART(optional), RISC-V extension(ecalls)(optional)