A 32-bit CPU with RV32I ISA

Design Specification

Creation Date：2025.7.18

Version： 0406

University of Electronic Science and Technology of China

Chengdu, Sichuan

A 32-bit CPU with RV32I ISA

Abstract

本文档是研究 AI 辅助条件下的数字 IC 设计环节中的设计规格说明。

设计规格说明介绍

1. 设计背景 (Context)
   包括项目背景，指令集定义和设计应用场景。其中，项目背景包括架构的简单描述，为大模型指出项目的整体方向；指令集定义指出了设计的功能需求，而应用场景对设计的性能提出了更多的限制。将设计规格说明输入大模型时提供设计背景可能使其提出更加符合要求的解决方案
2. 设计内容 (Design)
   设计内容包括模块的功能说明，IO 说明以及架构说明
3. 验证 (Verification)
   1) 工程师提供测试原理的说明，若没有提供测试原理说明则大模型提供测试原理的建议。
   2) 大模型依据提供/生成的测试原理说明生成正确的 testbench
4. 规格验收表 (Specification Checklist)
   情况一：大模型根据“设计内容板块”提取规格要求并检验所设计电路是否满足要求
   情况二：大模型根据工程师提供的设计规格表格检验所设计电路是否满足要求

1 Context

1.1 Requirements

This project aims to design a 32-bit RISC-V CPU that supports RV32I instruction set architecture (ISA). This CPU is supposed to adopt five-stage pipelining techniques and being organized in the Harvard architecture.

It is assumed that there would only be user-level instructions, which means there is no requirement for the implementation of SYSTEM instructions in RV32I ISA, such as those regarding interrupts and memory ordering. Therefore, the CPU should contain 32 registers in addition to pipeline registers and a program counter register, which can implement all the user-level instructions in RV32I ISA. The control and status registers (CSRs) are not considered since they are used by potentially privileged instructions beyond the user level.

1.2 ISA definition

This section provides a detailed and rigorous introduction to the Instruction Set Architecture (ISA). Total of 37 instructions are included in the ISA, and they can be categorized into 5 different clusters (R-type, I-type, S-type, B-type, and U/J-type). This section introduces each instruction in the exact order.

1.2.1 R-type instruction

As shown in Table 1-1, R-type instructions perform logistic or arithmetic operations between two pieces of data in two source registers and store the result on the destination register. Each piece of data is 32 bits in length and is expected to be a signed integer.

Table 1-1 R-type instructions

Function [9:3]	rs2 5bits	rs1 5bits	Function [2:0]	rd 5bits	Opcode 7bits	instruction	Assembly representation
0000000	src_2	src_1	000	R_d	0110011	ADD	ADD rd, rs1, rs2
0000000	src_2	src_1	010	R_d	0110011	SLT	SLT rd, rs1, rs2
0000000	src_2	src_1	011	R_d	0110011	SLTU	SLTU rd, rs1, rs2
0000000	src_2	src_1	111	R_d	0110011	AND	AND rd, rs1, rs2
0000000	src_2	src_1	110	R_d	0110011	OR	OR rd, rs1, rs2
0000000	src_2	src_1	100	R_d	0110011	XOR	XOR rd, rs1, rs2
0000000	src_2	src_1	001	R_d	0110011	SLL	SLL rd, rs1, rs2
0000000	src_2	src_1	101	R_d	0110011	SRL	SRL rd, rs1, rs2
0100000	src_2	src_1	000	R_d	0110011	SUB	SUB rd, rs1, rs2
0100000	src_2	src_1	101	R_d	0110011	SRA	SRA rd, rs1, rs2

Apart from regular addition, subtraction, AND operations etc., the introduction of the R-type instruction focuses on the introduction of SLL, SRL, SRA, SLT, and SLTU, which are logistic and arithmetic shifting operations.

(1) SLL or SRL: Shift-Left/Right-Logistic.

(2) SRA: Shift-Right-Arithmetic.

These three instructions allow the software developers to perform both logistic and arithmetic bit-shift. It shall be noted that a leftward arithmetic shift is illegal in the current ISA since it is equivalent to the SLL.

(3) SLT or SLTU: Set if Less Than/Unsigned.

The SLT instruction compares the values respectively on rs1 between rs2. If the value on rs1 is lower than that on rs2, set the value on rd to 1. However, when executing SLTU instruction, it is the absolute values are compared.

1.2.2 I-type instructions

As shown in Table 1-2, I-type instructions comprise two subtypes of instructions: 1) arithmetic and logistic operations on immediate numbers, and 2) loading immediate numbers to destination registers.

Table 1-2 I-type instructions

I-immediate [11:0]	rs1 5bits	Function [2:0]	rd 5bits	Opcode 7bits	instruction	Assembly representation
imm	src_1	000	R_d	0010011	ADDI	ADD rd, rs1, imm
imm	src_1	010	R_d	0010011	SLTI	SLT rd, rs1, imm
imm	src_1	011	R_d	0010011	SLTIU	SLTU rd, rs1, imm
imm	src_1	111	R_d	0010011	ANDI	AND rd, rs1, imm
imm	src_1	110	R_d	0010011	ORI	OR rd, rs1, imm
imm	src_1	100	R_d	0010011	XORI	XOR rd, rs1, imm
imm	src_1	000	R_d	0010011	SLLI	SLL rd, rs1, imm
imm	src_1	001	R_d	0010011	SRLI	SRL rd, rs1, imm
imm	src_1	101	R_d	0010011	SRAI	SRA rd, rs1, imm

Comparing Table 1-2 and Table 1-3, developers are allowed to directly write immediate numbers to participate in both numerical and logistic operations.

Table 1-3 I-type instructions for load purpose

I-immediate [11:0]	rs1 5bits	Function [2:0]	rd 5bits	Opcode 7bits	instruction	Assembly representation
imm	src_1	000	R_d	0000011	LB	LB rd, rs1, imm
imm	src_1	010	R_d	0000011	LH	LH rd, rs1, imm
imm	src_1	010	R_d	0000011	LW	LW rd, rs1, imm
imm	src_1	100	R_d	0000011	LBU	LBU rd, rs1, imm
imm	src_1	101	R_d	0000011	LHU	LHU rd, rs1, imm

(1) LW, LH, or LB: Load-Word/Half/Byte (U: unsigned).

These instructions allow the data to be loaded from the main memory to the destination register. The address of the exact data equals to the addition of the data in rs1 and the input immediate value. The lower half/8-bit of data is loaded to the rd when using LH/LB instructions.

It shall be noticed that the immediate numbers specified in Table 1-2 and Table 1-3, will be sign-extended to 32bits. In addition, when executing LB/HU instruction, only the absolute value will be loaded to the register.

1.2.3 S-type instructions: store operations

As shown in Table 1-4, the S-type instructions allow the developer to store data in the main memory at a designated address.

Table 1-4 S-type instructions

S-imm [11:5]	rs2 5bits	rs1 5bits	Function [2:0]	S-imm [4:0]	Opcode 7bits	instruction	Assembly representation
#imm [11:5]	src_2	src_1	000	#imm [4:0]	0100011	SB	SB rd, rs1, #imm
#imm [11:5]	src_2	src_1	001	#imm [4:0]	0100011	SH	SH rd, rs1, #imm
#imm [11:5]	src_2	src_1	010	#imm [4:0]	0100011	SW	SW rd, rs1, #imm

(1) SW: Store-Word
(2) SH or SB: Store Half/Byte

1.2.4 B-type instructions: conditional branching operations

Software developers can jump between different code blocks using conditional branching instructions. Immediate numbers in the B-type instructions are divided into two separate segments. Two pieces of data on rs1 and rs2 are the sources of the conditional operations. Once the specified condition is met, the program counter will be diverted to address PC + imm.

Table 1-5 B-type instructions

B-immediate [12] [10:5]	rs2 5bits	rs1 5bits	Function [2:0]	B-imm [4:1][11]	Opcode 7bits	instruction	Assembly representation
#imm [12] [10:5]	src_2	src_1	000	#imm [4:1][11]	1100011	BEQ	BEQ rs1, rs2, #imm
#imm [12] [10:5]	src_2	src_1	001	#imm [4:1][11]	1100011	BNE	BNE rs1, rs2, #imm
#imm [12] [10:5]	src_2	src_1	100	#imm [4:1][11]	1100011	BLT	BLT rs1, rs2, #imm
#imm [12] [10:5]	src_2	src_1	110	#imm [4:1][11]	1100011	BLTU	BLTU rs1, rs2, #imm
#imm [12] [10:5]	src_2	src_1	101	#imm [4:1][11]	1100011	BGE	BGE rs1, rs2, #imm
#imm [12] [10:5]	src_2	src_1	111	#imm [4:1][11]	1100011	BGEU	BGEU rs1, rs2, #imm

(1) BEQ or BNE: Branch-Equal/Branch-Not-Equal
(2) BLT or BGT: Branch-Less/Greater-Than (U: unsigned)

In BEQ or BNE cases, if the value on rs1 is (not, in the later) equal to the value on rs2, the program counter will point to the current address plus #imm. Likewise, when performing BLT or BGT operations, the

same branching operations will be executed if the value on rs1 is less/greater than that on rs2. When implementing unsigned operations, the highest positions (sign digits) are not considered. It shall be noted that one zero shall be appended to the LSB of the input immediate number otherwise, the malfunction of the PC.

1.2.5 U/J -format instructions: upper immediate numbers / unconditional jump

The U/J-type instructions load/add upper 20-bit immediate numbers to the destination registers or the Program Counter. These can be used to obtain large integers or jump to another instruction.

Table 1-6 U-type instructions

U-immediate [31:12]	rd 5bits	Opcode 7bits	instruction	Assembly representation
#imm	R_d	0110111	LUI	LUI rd, #imm
#imm	R_d	0100011	AUIPC	AUIPC rd, #imm

(1) LUI: Load-Upper-Immediate.
(2) AUIPC: Add-Upper-Immediate-to-PC.

Both immediate numbers specified in Table 1-6 are extended to 32bits by adding zeros to their lower twelve digits. The LUI instruction loads the extended immediate number to the destination register. By synthesizing LUI and ADDI, developers can write 32-bit immediate numbers to a desired register. The AUIPIC instruction adds the extended number to the address on the program counter. This combination of the address and the offset immediate number will be further loaded to the destination register.

Table 1-7 J-type instructions

J-immediate [20] [10:1][11] [19:21]			rd 5bits	Opcode 7bits	Instruction	Assembly representation
#imm			R_d	1101111	JAL	JAL rd, #imm
I-immediate [11:0]	rs 5bits	Function [2:0]	rd 5bits	Opcode 7bits	Instruction	Assembly representation
#imm	src_1	000	R_d	1101111	JALR	JALR rd, rs, #imm

(1) JAL: Jump-and-Link.
(2) JALR: Jump-and-Link-Register.

The JAL instruction performs an unconditional jumping operation to the address, which equals to PC + offset (J-immediate), and saves the address of the next instruction to the rd. The JALR instruction loads the address of the next instruction to rd and executes the instruction at the rs + offset on the main memory. One zero is appended to the LSB of the J-immediate, which will be sign-extended to 32 bits.

1.3 Application Scenarios

Figure 1-1 shows a possible application scenario for this CPU, where it is connected to two memory modules (IMEM refers to Instruction Memory; DMEM refers to Data Memory).

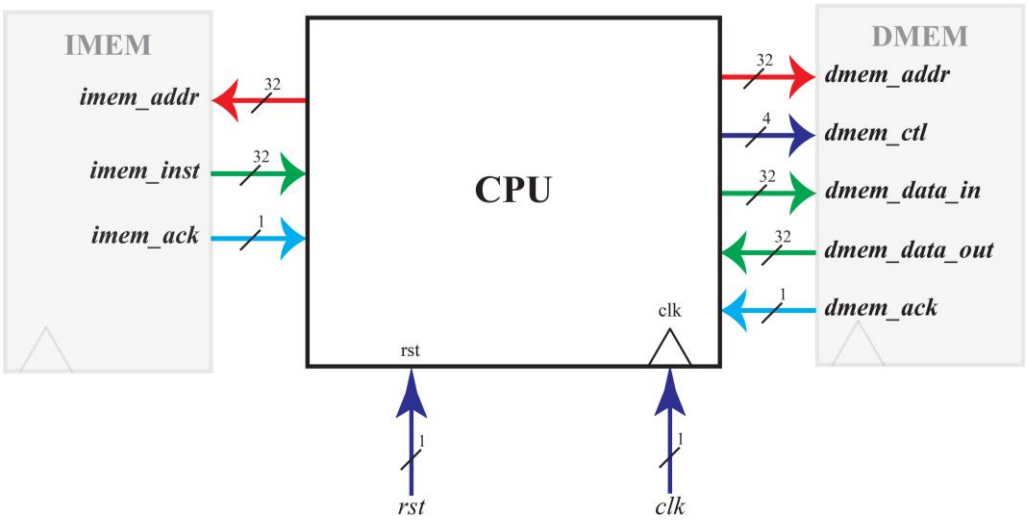

Figure 1-1 The application scenarios diagram for the CPU

It is worth noting that the memory module shown in Figure 1-1 can be replaced by caches or memory controllers. The DMEM module can also contain input/output ports by allocating them to different addresses.

2 Design：XXXX

2.1 Function

2.2 Input/Output

2.3 Architecture

3 Verification

测试用例名		测试用例标识	
追踪关系			
测试原理			
初始化要求			
前提约束			
终止条件			

测试规程			
步骤	输入及操作说明	预期结果	实际结果
1.			
2.			
3.			

4.			
5.			

结果评价准则	如果每一步的实际结果和预期结果相符，则测试通过，否则不通过		
测试人员		测试日期	
验收人员		验收日期	

4 Specification Checklist

SPEC No. 设计规格序号	SPEC Name 设计规格名称	Verification Principle 验证原理	Termination Condition 终止条件		Verification Result 测试结果
1.	XXX				
2.	XXX				
3.	XXX				