RISC 软核 CPU 设计

1，简介

FPGA 设计中在 IP 核的提供方式上，通常将其分为软核、固核和硬核这 3 类。软核(Soft IP Core)：软核在 EDA 设计领域指的是综合之前的寄存器传输级(RTL) 模型；通常遍是指以 HDL 代码（Verilog，VHDL...）为形式的可综合源代码；固核(Firm IP Core) :固核在 EDA 设计领域指的是带有平面规划信息的网表；硬核(Hard IP Core) :硬核在 EDA 设计领域指经过验证的设计版图。软核只经过功能仿真，需要经过综合以及布局布线才能使用。其优点是灵活性高、可移植性强，允许用户自配置。**软核处理器**是指利用 HDL 语言描述的处理器功能代码，用于实现处理器的所需要的各种功能。

通常的处理器架构由以下部分组成。指令寄存器、累加器、算术逻辑运算单元、数据控制器、状态控制器、程序计数器、地址多路器等基本部件。是用于实现根据特定指令集生成的汇编代码的硬件运行环境。

2，CPU 的流水线

常见的 CPU 内部有 5 级流水线组成。cpu 的工作大致分为以下几个步骤；

1：取指。该阶段从内存中读取指令，PC（程序计数器）制定指令的地址。

2：译码。该阶段将从内存读取的指令翻译为各种操作。并从寄存器中取出操作数。

3：执行。该阶段算数逻辑单元执行指令表示的操作。

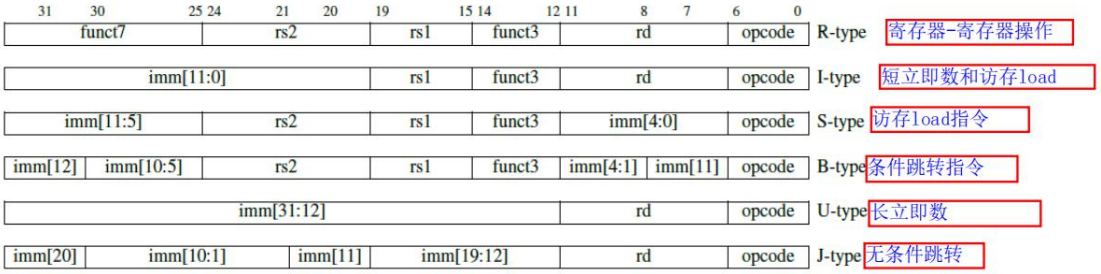
4：访存。该阶段将结果数据写入到内存中。

5：写回。将结果写会到寄存器文件中。

在 CPU 的工作流程中，首先读取 PC（程序计数器）指向的地址的指令，送入到译码模块，译码器对 opcode 指令进行译码，经过译码之后得到指令需要的操作数寄存器索引，可以使用此索引从通用寄存器组（Register File，Regfile）中将操作数读出。指令译码之后所需要进行的计算类型都已得知，并且已经从通用寄存器组中读取出了所需的操作数，那么接下来便进行指令执行。指令执行是指对指令进行真正运算的过程。譬如，如果指令是一条加法运算指令，则对操作数进行加法操作；如果是减法运算指令，则进行减法操作。

3，RISC-V 指令集介绍

下图显示了六种基本指令格式，分别是：用于寄存器-寄存器操作的 R 类型指令，用于短立即数和访存 load 操作的 I 型指令，用于访存 store 操作的 S 型指令，用于条件跳转操作的 B 类型指令，用于长立即数的 U 型指令和用于无条件跳转的 J 型指令。



首先，指令只有六种格式，并且所有的指令都是 **32 位长**，这简化了指令解码。第二，RISC-V 指令提供**三个寄存器操作数（rs1,rs2,rd）**，而不是像 x86-32 一样，让源操作数和目的操作数共享一个字段。当一个操作天然就需要有三个不同的操作数，但是 ISA 只提供了两个操作数时，编译器或者汇编程序程序员就需要多使用一条 move（搬运）指令，来保存目的寄存器的值。第三，在 RISC-V 中对于所有指令，**要读写的寄存器的标识符总是在同一位置**，意味着在解码指令之前，就可以先开始访问寄存器。第四，这些格式的立即数字段总是符号扩展，**符号位总是在指令中最高位**。这意味着可能成为关键路径的立即数符号扩展，可以在指令解码之前进行。

31 25	24 20	19 15	14 12	11 7	6 0	
imm[31:12]				rd	0110111	U lui
imm[31:12]				rd	0010111	U auipc
imm[20|10:1|11|19:12]				rd	1101111	J jal
imm[11:0]		rs1	000	rd	1100111	I jalr
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	B beq
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	B bne
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	B blt
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	B bge
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	B bltu
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	B bgeu
imm[11:0]		rs1	000	rd	0000011	I lb
imm[11:0]		rs1	001	rd	0000011	I lh
imm[11:0]		rs1	010	rd	0000011	I lw
imm[11:0]		rs1	100	rd	0000011	I lbu
imm[11:0]		rs1	101	rd	0000011	I lhu
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	S sb
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	S sh
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	S sw
imm[11:0]		rs1	000	rd	0010011	I addi
imm[11:0]		rs1	010	rd	0010011	I slti
imm[11:0]		rs1	011	rd	0010011	I sltiu
imm[11:0]		rs1	100	rd	0010011	I xori
imm[11:0]		rs1	110	rd	0010011	I ori
imm[11:0]		rs1	111	rd	0010011	I andi
0000000	shamt	rs1	001	rd	0010011	I slli
0000000	shamt	rs1	101	rd	0010011	I srli
0100000	shamt	rs1	101	rd	0010011	I srai
0000000	rs2	rs1	000	rd	0110011	R add
0100000	rs2	rs1	000	rd	0110011	R sub
0000000	rs2	rs1	001	rd	0110011	R sll
0000000	rs2	rs1	010	rd	0110011	R slt
0000000	rs2	rs1	011	rd	0110011	Rsltu
0000000	rs2	rs1	100	rd	0110011	R xor
0000000	rs2	rs1	101	rd	0110011	R srl
0100000	rs2	rs1	101	rd	0110011	R sra
0000000	rs2	rs1	110	rd	0110011	R or
0000000	rs2	rs1	111	rd	0110011	R and
0000 pred	succ	00000	000	00000	0001111	I fence
0000 0000	0000	00000	001	00000	0001111	I fence.i
000000000000		00000	00	00000	1110011	I ecall
000000000000		00000	000	00000	1110011	I ebreak
csr		rs1	001	rd	1110011	I csrrw
csr		rs1	010	rd	1110011	I csrrs
csr		rs1	011	rd	1110011	I csrrc
csr		zimm	101	rd	1110011	I csrrwi
csr		zimm	110	rd	1110011	I cssrrsi
csr		zimm	111	rd	1110011	I csrrci

图 2.3：RV32I 带有指令布局，操作码，格式类型和名称的操作码映射。（此图基于[Waterman and Asanovi'c 2017]的表 19.2。）

RV32I 带有指令布局，操作码，格式类型和名称的操作码映射。

4，设计总览

该设计由以下的模块组成。

- 取指模块。用于从内存读取指令。
- 译码模块。用于指令解码。
- 执行模块。用于执行指令。
- 数据访问模块。用于从内存的数据读取。
- 寄存器模块。用于 CPU 工作时的寄存器，储存临时数据。
- 内存接口模块。用于生成访问内存读写的接口和时序。
- 控制模块。控制 CPU 工作时的各个模块之间的协调配合。
- CSR 模块。控制状态寄存器，使我们可以轻松地访问一些程序性能计数器，便于清楚处理器所处的状态。这些计数器包括了系统时间, 时钟周期以及执行的指令数目。