RISC-V 自定义指令加速器

Version 1.0

1 项目概述

本项目基于 RISC-V 指令集架构，借助 RISC-V 开源项目 Rocket Chip 平台设计 SoC, 并通过协处理器(Rocket Custom Coprocessor，RoCC) 接口，采用高层次语言 Chisel 定制 Cordic 算法片上加速器，实现自定义指令以完成正余弦函数值的计算。

2 系统架构

本系统 Rocket Core、SoC 总线、Bootrom、UART 和 Cordic 算法加速器五部分构成，如图 1 所示。其中，Rocket Core 是由伯克利大学基于 RISC-V 架构，采用高层次语言 Chisel 开发的一款开源处理器核生成器，具有良好的可配置性，提供协处理器接口，可供用户定制加速器。Cordic 算法加速器具备正余弦函数计算功能。

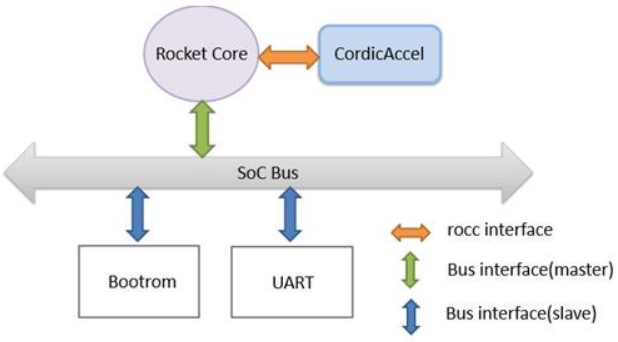

图 1 系统架构

3 电路设计基本原理

3.1 Cordic 算法

Cordic 算法以坐标旋转为基础，将旋转角度θ分成若干个连续的小偏转角，摇摆逼近目标旋转角度，通过简单的加减和移位操作来实现整个旋转过程。旋转角度的正余弦值可以由一点的 x,y 坐标求得。

3.2 自定义指令

RISC-V 架构定义了 4 组自定义指令：custom0、custom1、custom2 和 custom3。指令格式如图 2 所示，定制加速器实现自定义指令需要遵循此格式。

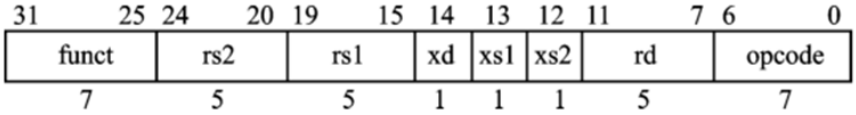

图 2 Custom-X 指令格式

其中，rs1、rs2 是源操作数寄存器号，rd 是目的寄存器号。funct 是具体的操作类型，opcode 是 Custom-X 指令的操作码。xd、xs1、xs2 分别表示 rd、rs1、rs2 对应通用的寄存器是否需要访问。本项目中的加速器实现了 custom0 指令。

3.3 RoCC 接口在系统中的位置

Cordic 算法运算单元通过 RoCC 接口与主处理器 Rocket Core 连接，以协处理器的形式充当加速器的角色。CPU 在译码的时候根据 opcode 判断指令是否为自定义指令，如果是，处理器 Rocket core 就将自定义指令通过 RoCC 接口发送给加速器。加速器执行完这条指令时，将结果又通过 RoCC 接口发送给处理器。因而，RoCC 接口有两种模式：command 模式（如图 3）和 response 模式（如图 4）。command 模式下，CPU 发送指令和数据；response 模式下，CPU 接收数据。CPU 与加速器之间的数据交互时序如图 5 所示。

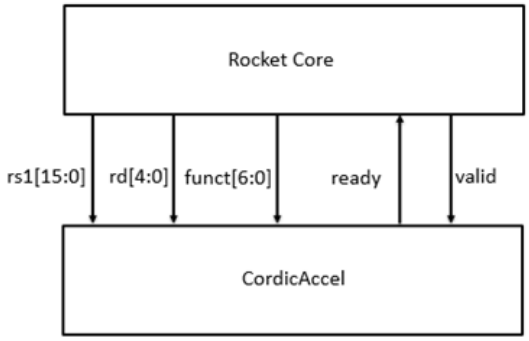

图 3 RoCC command 接口

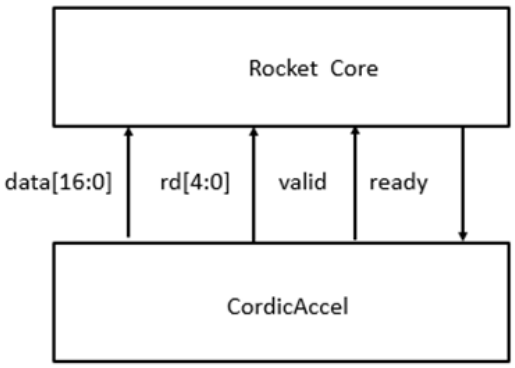

图 4 RoCC response 接口

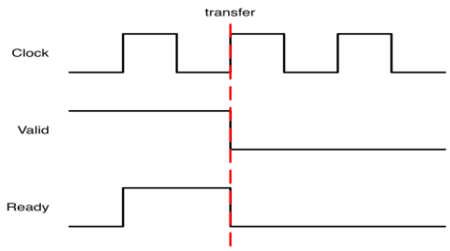

图 5 握手信号时序图

4 RoCC 电路结构

4.1 RoCC 接口信号

表 1 RoCC command 模式信号定义

信号名	方向	说明
reset	输入	复位信号
clock	输入	时钟信号
io.cmd.bits.rs1[15:0]	输入	源寄存器编号
io.cmd.inst.rd[4:0]	输入	目的寄存器编号
io.cmd.funct	输入	功能码，选择计算类型
io.cmd.valid	输入	控制信号
io.cmd.ready	输出	控制信号

表 2 RoCC response 模式信号定义

信号名	方向	说明
reset	输入	复位信号
clock	输入	时钟信号
io.resp.bits.rd	输出	目的寄存器编号
io.resp.data	输出	计算结果信号
io.resp.valid	输出	控制信号
io.resp.ready	输入	控制信号

4.2 RoCC 内部结构

Cordic 算法运算单元采用循环 16 次的反馈结构，由象限预处理单元，循环单元和符号后处理单元三部分组成，如图 6 所示。其中，最主要的循环单元结构

如图 7 所示。

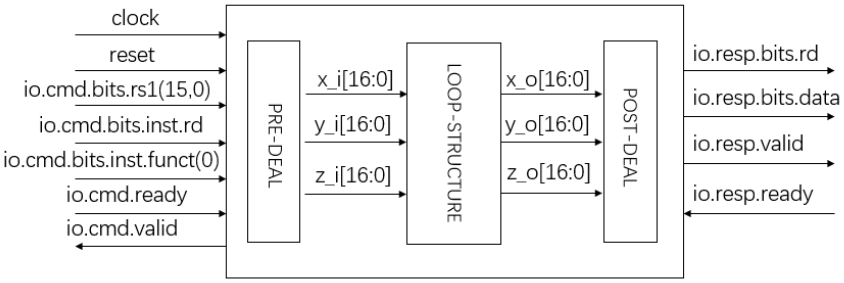

图 6 RoCC 电路结构

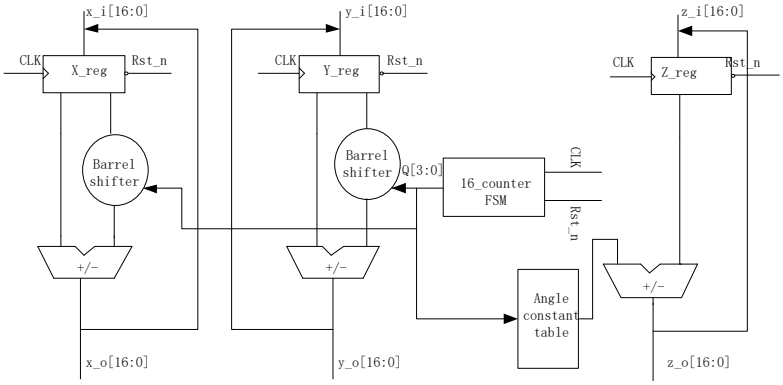

图 7 RoCC 循环模块结构

4.3 计算示例

输入相位角采用 16 位二进制数均匀量化，量化范围为 0 到 360 度。正余弦值的计算结果采用 17 位二进制补码表示，最高位表示符号位，低 16 位表示数值大小。如表 2 的计算示例,分别在四个象限选取一个角度，并给出其对应的预期计算结果。

表 3 计算示例

原始角度(deg)	45	135	225	315
换算角度	0x2000	0x6000	0xA000	0xE000
正弦值	0.0701	0.0701	-0.0701	-0.0701
	0xB504	0xB504	0x14AFC	0x14AFC
余弦值	0.0701	-0.0701	-0.0701	0.0701
	0xB504	0x14AFC	0x14AFC	0xB504

注：$1\deg = \frac{2^{16}}{360}$

5 FPGA 原型验证

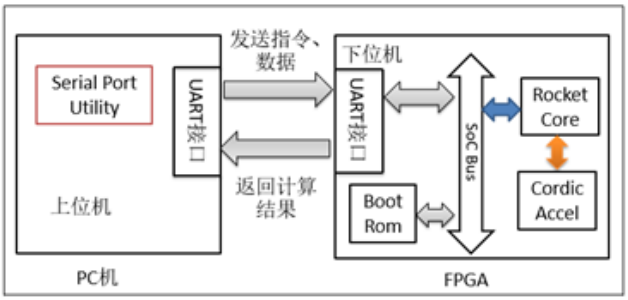

图 8 FPGA 原型验证平台

如图 8 所示，将 UART 驱动程序以硬编码的形式固化在 Bootrom 里面，此程序用于查询上位机发送过来数据和指令，当 CPU 接收到指令并译码之后，如果判断这条指令是自定义指令，便将该指令发送给 Cordic 算法加速器计算。计算完成之后，计算结果又通过 UART 接口返回到上位机。

若规定上位机发送数据为四个字节，高 8 位表示计算类型，即求正弦值或者求余弦值。定义当高 8 位发送 0，表示计算正弦值；高 8 位发送 1，表示计算余弦值。规定上位机接收数据也为 4 个字节，低 16 位表示数值大小，第 17 位表示符号，0 为正，1 为负。

图 9 FPGA 原型验证效果展示

如图 9 所示，上位机发送指令分别计算 45°、135°、225°、315° 的正弦值和余弦值，对比表 3 的预期结果，在误差允许范围内是正确的。

6. 文件列表

（1）组织工程相关的文件

build.sbt　　#工程配置文件
makefrag　　#编译工程的脚本

（2）源代码（src/main/scala）

TYOD_SoC.scala　　#SoC 的顶层文件
MyConfig.scala　　#SoC 的配置文件
Generator.scala　　#SoC 生成器文件，与生成 Verilog 相关
TestHarness.scala　　#生成 Verilog 相关文件
CordicRoCC.scala　　#基于 RoCC 接口实现的 Cordic 算法加速器代码

（3）Sifive-blocks　　#Sifive 开源项目，包含 UART 接口 IP

（4）Bootrom

将程序添加至 Bootrom.S，生成.img 文件，在生成 Verilog 时，将程序硬化。

（5）Vsim　　#该目录下生成 Verilog

7. 改进方案

改进方案：增加指令中“精度”配置。

8. 课题组简介

本项目由电子科技大学低功耗集成电路与系统研究所（Institute of Integrated Circuits and Systems, UESTC）数字系统设计研究小组创建并维护。

电子科技大学低功耗集成电路与系统研究所，前身为通信学院“无线通信与嵌入式系统实验室”（111 教研室/李广军教授团队）的模拟集成电路研究组，2014 年调整进入原微电子与固体电子学院成立集成系统实验。2016 年 7 月在学校创新平台培育计划支持下，挂牌成立电子科技大学低功耗集成电路与系统研究所，具有独立对外学术活动权，由李强教授负责。研究所研究领域集中在集成电路与集成系统设计，重点关注集成电路与集成系统设计中的功耗优化问题。研究方向包括模拟与数模混合集成电路、射频 IC、计算机系统结构与大规模集成电路设计等，尤其是 ADC、无线通信芯片/SoC、传感器接口电路、多核/多处理器 SoC 等目前我国与国际先进水平差距较大的研究方向。

电子科技大学低功耗集成电路与系统研究所数字系统设计研究小组由黄乐天副教授负责，重点关注以多核/多处理器 SoC 为代表的系统级集成电路设计方法学研究。目前已在多个领域内顶级期刊和会议上发表了论文。小组自 2016 年初开始从事基于 RISC-V 指令集的领域专用处理器与 SoC 研究，力求探索面向低功耗、高能效的领域专用处理器通用设计方法。小组此后将持续更新本项目，并陆续上线基于 RISC-V 指令集的多核处理器、基于 RISC-V 指令集的人工智能

SoC 等开源项目。欢迎各位关注。