面向FPGA的软硬件结合异步状态机接口设计与实现

英文标题：

Design and implementation of software-hardware combined asynchronous state machine interface for FPGA

# 一、选题依据

## 1. 背景和研究意义

在片上系统（System-on-Chip，SoC）的发展历程中，异构计算一直是一个重要的课题。片上系统的异构计算是指在一个芯片上集成多个不同类型的处理器和加速器，以实现高效、灵活和优化的计算。这种异构计算架构可以充分发挥各种处理器和加速器的优势，针对不同的计算任务进行分配和协同工作，以提高系统的性能和效能。

在异构计算中，常见的处理器和加速器包括以下几方面：一是中央处理器（CPU），CPU是片上系统中最常见的处理器，负责通用计算任务。它具有较强的单线程性能和复杂的控制逻辑，适用于运行操作系统、控制流程和执行串行任务等。二是图形处理器（GPU），GPU主要用于图形渲染和图像处理任务。它具有高并行计算能力和大规模并行处理单元，适合于同时处理大量数据的并行计算，如图像处理、计算机视觉和科学计算等。三是张量处理器（TPU），TPU是专门用于加速人工智能和机器学习任务的处理器。它具有高度优化的矩阵乘法和向量计算单元，能够高效执行深度神经网络的推断和训练任务。四是数字信号处理器（DSP），DSP主要用于数字信号处理和实时控制任务。它具有高效的乘法和累加运算单元，适合于音频处理、视频编解码和通信信号处理等领域。五是硬件加速器（例如FPGA、ASIC），硬件加速器是定制化的处理器，根据特定的计算需求进行设计和优化。FPGA具有可重构的逻辑和数据通路，适合于快速原型开发和灵活的定制化计算。ASIC是专门设计和定制的应用特定集成电路，具有更高的性能和能效，但开发和生产成本较高。

不同处理器和加速器之间通过高速互联通道进行通信和数据传输。根据具体的计算任务和需求，系统可以动态地分配任务给不同的处理器和加速器，以实现最优的计算效果。其中，集成了FPGA与CPU的SoC可同时具备FPGA的⼤规模并发式处理能⼒以及CPU的通⽤计算能⼒。但其也面临着一些问题，在FPGA程序设计中很难⾼效的使⽤到CPU的计算能⼒造成资源浪费。在需要用到大量浮点数及乘除法运算的步骤中，FPGA上计算元器件数量成为瓶颈。因此，可以通过编写FPGA高效使用CPU计算存储资源的异步接口，来提高对CPU计算资源的利用率。同时，CPU对FPGA方面也存在一定的局限性，如FPGA功能较为固定不够灵活。若采用库函数调用封装的方式，使得CPU对FPGA功能的使用更加多变。

本课题的目标是实现一种基于软硬件的FPGA与CPU通信接口，它在保留了CPU与FPGA通信的优势的前提下，一方面可以提高对CPU计算资源的利用率，节省FPGA资源损耗；另一方面可以增强CPU对FPGA调用的灵活程度，提高计算性能。同时，定义一个具备可移植性的接口规范，便于软件与硬件程序员的使用。

## 2. 国内外研究概况

目前，基于FPGA的加速器设计主要关注性能和效率[1]。FPGA程序员可以进行如下操作：( 1 )创建由简单(通常是低位宽)计算单元组成的深度定制流水线，而不是完全成熟的ALU；( 2 )构建高度并行和分布式的控制逻辑和片上存储；( 3 )以显式的方式调度数据流，以尽量减少片外存储器访问，而不使用缓存。这与编程微处理器(即CPUs)和通用图形处理单元(也就是说, GPU)形成鲜明对比，后者的底层硬件架构(指令流水线、存储器层次结构等。)是固定的，软件程序的控制流驱动硬件基于指令的执行。FPGA可以针对特定的应用或应用领域进行重新配置/定制，来利用其大规模的细粒度并行性和片上带宽。与CPU和GPU相比，这往往提供了更高的计算吞吐量，更低的能耗和更可预测的延迟。

Griffin Lacey等人指出[2]，对于卷积神经网络(CNN)的推理阶段，微软团队利用FPGA(Stratix V D5)实现了高性能的加速。目前，一些国内外知名公司，如亚马逊、微软、腾讯以及阿里巴巴等，也逐步尝试将FPGA部署到数据中心中，FPGA与CPU协同开发已成为发展的趋势。

在SoC FPGA中CPU和FPGA是紧耦合的，即CPU和FPGA是封装在一个芯片中，通过总线连接，其中CPU通常分为两种：硬核和软核．目前FPGA供应商如Xilinx和Altera提供的最新SoC开发板使用的CPU通常为ARM硬核。SoC FPGA架构的主要特点是集成度高、功耗低及通信带宽较高，通常用于嵌入式应用中。

与SoC FPGA不同，标准FPGA与CPU是松 耦合结构，即FPGA和CPU不在一个芯片中，两者 一般通过PCIe(Peripheral Component Interconnect express)接口等进行连接．由于标准FPGA的芯片 中未嵌入微处理器，因此在同等芯片面积的情况下，标准FPGA相比于SoC FPGA中的FPGA逻辑部 分拥有更丰富的硬件资源，能实现更为复杂的应用。对于标准FPGA中CPU和FPGA之间的数据交互，目前常用的方式有两种：基与PCIe总线的数据交互和基于QPI(Quick Path Interconnect)总线的数据交互。

传统的FPGA设计工具主要针对硬件设计专家，而不是软件编程人员。使用传统的寄存器传输级( RTL )方法手动创建和优化加速器架构需要花费大量的精力。人们必须与低级硬件描述语言( HDL )描述和计算机辅助设计( CAD )工具进行斗争，以实现丰富的硬件定制，如定点运算、流水线、存储单元和双缓冲。更糟糕的是，将RTL设计综合为比特流通常需要数小时，甚至数天。这种漫长的编译周期使得FPGA上的设计空间探索( DSE )变得非常昂贵。如何降低FPGA与CPU协同开发的门槛，让软件与硬件程序员能使用友好的接口进行编程是一个需要重点关注的问题。

# 二、研究目标和内容

## 1. 研究目标和主要内容

本研究拟分为接口设计和性能测试两部分。首先在zedboard上设计并实现CPU与FPGA的异步调用，并验证其正确性，再将其和现有的异构计算机制进行性能测试，对比分析测试结果。

在接口设计方面在CPU与FPGA之间设计一个透明的互相调用接口。在CPU调用FPGA时，希望对FPGA的调用是一个函数调用的形式，并能够异步执行；在FPGA调用CPU时，希望FPGA以调用子模块的方式调用CPU来完成计算任务，并通过数据交互形式直接获得计算结果。

本研究所实现的软硬件接口需要达到的目标为：可以正常工作并给出期望结果；能够发挥异构计算的性能优势；具有可扩展性并具备可移植性。本研究的对比测试部分需要达到的目标为：设计科学、完善的对比实验并得出可靠结果；分析产生实验结果的原因；分析本研究实现的各个机制对结果的影响。

## 2. 拟解决的关键问题

FPGA与CPU之间通常采用总线传输协议，即AXI、PCIE等。FPGA内的模块通常作为外设模块挂载在总线上，每个模块分配一个物理地址。在CPU中运行的程序直接通过物理地址对FPGA内模块进行读写访问。FPGA要访问内存时，通常采用同步访问方式，要实现较为复杂的状态机控制每一个读写信号，比较麻烦。另外，因为没有Cache因此访问速度慢。

 CPU有内置Cache机制，访问速度快；且有高级语言封装，对于程序员来说不用考虑同步和异步问题，访问方便。FPGA内部通常集成了存储空间和DSP模块（用于浮点计算），但资源较少，无法与CPU的三级存储结构效率和易用性相提并论。

在使用FPGA进行计算的过程中，对于大规模的数据计算，FPGA实现需要消耗的计算资源量较多，受制于其计算元器件数量。而CPU的计算资源得到的利用较少，产生了计算资源的浪费。通过实现FPGA调用CPU的接口，解决了CPU计算资源利用率较少问题，节省了计算资源。同时，CPU对于FPGA调用的灵活性较低，通过实现CPU调用FPGA的函数式封装，提高了调用的灵活性。

# 三、研究方案

**1. 研究方法**

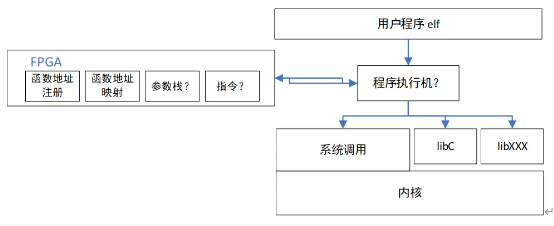
本项研究在不同阶段采用的研究方法如下：

* + 在调研阶段，采用文献研究法。学习zynq上硬件开发、CPU与FPGA通信、soc异构计算的相关知识。同时，调研现有的异步通信接口的设计方案，从中学习值得借鉴的设计思路。
  + 在设计开发和对比测试阶段，采用实验研究法和定性分析法。设计对比实验，测试本项研究设计的接口和现存几个接口在不同外部条件下的性能。并且，采用定性分析的方式，根据不同接口的实现机制，分析实验结果。

**2. 开发阶段的技术路线**

可以从两个角度进行分析，分别是（1）FPGA端访问CPU资源；（2）CPU访问FPGA资源。

基本结构：



* + **CPU访问FPGA资源**

基本思路：

1. 用户程序在调用库函数时，自动对被调函数入口地址进行匹配：匹配成功则自动调用FPGA内部模块，否则调用原函数地址。
2. FPGA内的模块动态加载和进行函数注册与映射
3. 要使本结构正常工作，需要对系统执行程序的机制进行修改（或者用一个显式的程序对可执行文件单独进行一次扫描，替换入口地址）

程序编译运行过程：

1. 编译器不做修改，程序正常流程进行编译链接生成可执行文件
2. 程序运行开始，将程序读入进程，进行相关解释运行
3. 对于每一个程序段，根据FPGA中保存的函数地址映射关系，动态修改入口地址
4. FPGA和软件中的库不需要了解其功能是由谁调用的。
   * **FPGA端访问CPU资源**

所谓的CPU资源即使用高级程序语言实现的各类功能，功能以函数的形式体现。对于软件开发者来说，还是以原来的方式编写程序。可以实现如下两种方式，即面向异步执行的函数调用，以及面向同步执行的函数调用:

**面向异步执行的函数调用**:

流程：

1. FPGA中模块进行初始化时，向CPU进行函数注册，即注册需要调用的函数。CPU将函数与模块进行匹配注册
2. 在运行中，需要调用函数时，模块向CPU发送中断请求
3. CPU响应中断，进行程序查询操作，即查询模块的函数调用请求。并取得对应输入参数
4. 调用对应函数，把函数预期运行时间写入FPGA模块寄存器
5. 函数运行结束，将运行结果和状态写回FPGA模块寄存器

**3. 实验阶段的实验方案**

在实验阶段，将对接口在FPGA端与CPU端分别进行性能测试。通过编写多个不同的例程来测试接口的性能，可靠性以及接口使用的便宜程度。在FPGA与CPU通信的接口端，主要测试接口的性能，期望在发挥FPGA可编程性优势的情况下，充分利用CPU计算资源，达到合适的性能。在CPU与FPGA通信端，主要测试接口调用的可靠性以及使用的简易度，期望接口达到较高的封装程度，并具有透明性，能够实现CPU调用FPGA的简便规范性接口。

实验例程将采用由简单到复杂例程的方式，测试接口在不同工作压力下的表现性能，以及封装的可靠性。

**4. 可行性分析**

在前期调研阶段，已经进行了初步的学习工作，包括：完成了用verilog语言实现软核与FPGA通信的实验，学习了zedboard开发的相关知识，学习已有的CPU调用FPGA接口实现方式等。目前对FPGA调用CPU接口已有初步构思，未遇到特别大的障碍。在完成FPGA调用CPU接口之后，CPU调用FPGA接口的封装要更加复杂一些，但总体上的开发思路和开发流程与传统的开发大致相同，预估在设计和开发过程中不会遇到太大的问题。

# 研究计划及进度安排

2023年12月 确定研究选题

2024年1月 进行开题工作

2024年1月~2024年2月 调研，学习相关知识；设计

2024年2月~2024年3月 开发，正确性测试

2024年3月~2024年4月 性能测试，实验

2024年4月 进行中期检查工作

2024年4月~2024年5月 撰写和完善论文

2024年5月 进行论文提交

2024年6月 进行毕业设计答辩

# 五、创新点及预期研究成果

**1. 创新点**

本研究的创新点在于以下三点：

第一，将FPGA对CPU的调用子模块化，减少了FPGA资源的使用及提高对CPU计算资源的利用率。

第二，将CPU对FPGA的调用函数化，便利了CPU对FPGA调用的灵活性。

第三，该研究将以开源库的形式发布，有利于soc异构计算生态。

**2. 预期研究成果**

* 完成研究内容，达成研究目的。
* 完成异步调用开源库的开发。
* 对本研究设计的调度器开展性能测试和对比实验，得出实验结果。

提交开题报告、中期报告、毕业论文等文字成果。

# 六、参考文献

[1]吴艳霞,梁楷,刘颖,等. 深度学习FPGA加速器的进展与趋势[J]. 计算机学报,2019,42(11):2461-2480. DOI:10.11897/SP.J.1016.2019.02461.

[2]Lacey G, Taylor G W, Areibi S. Deep learning on fpgas: Past, present, and future[J]. arXiv preprint arXiv:1602.04283, 2016.

[3]Wang T, Wang C, Zhou X, et al. A survey of FPGA based deep learning accelerators: Challenges and opportunities[J]. arXiv preprint arXiv:1901.04988, 2018.

[4]刘焰强, 戚正伟, 管海兵. FPGA加速系统开发工具设计:综述与实践. 软件学报, 2020, 31(10): 3087-3099. <http://www.jos.org.cn/1000-9825/6065.htm>

[5]Lai Y H, Ustun E, Xiang S, et al. Programming and synthesis for software-defined FPGA acceleration: status and future prospects[J]. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 2021, 14(4): 1-39.

[6]Zhou Z, Bi Y, Wan J, et al. Userspace Bypass: Accelerating Syscall-intensive Applications[C]//17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23). 2023: 33-49.

[7][Rehan Hameed](https://s.wanfangdata.com.cn/paper?q=%E4%BD%9C%E8%80%85:"Rehan Hameed"" \t "https://d.wanfangdata.com.cn/periodical/_blank),[Wajahat Qadeer](https://s.wanfangdata.com.cn/paper?q=%E4%BD%9C%E8%80%85:"Wajahat Qadeer"" \t "https://d.wanfangdata.com.cn/periodical/_blank),[Megan Wachs](https://s.wanfangdata.com.cn/paper?q=%E4%BD%9C%E8%80%85:"Megan Wachs"" \t "https://d.wanfangdata.com.cn/periodical/_blank),等.[Understanding Sources of Inefficiency in General-Purpose Chips](https://d.wanfangdata.com.cn/conference/CC0211563322" \t "https://d.wanfangdata.com.cn/periodical/_blank)[C].2010.

[8][Doug DiSabello](https://s.wanfangdata.com.cn/paper?q=%E4%BD%9C%E8%80%85:"Doug DiSabello"" \t "https://d.wanfangdata.com.cn/periodical/_blank),[Tom VanCourt](https://s.wanfangdata.com.cn/paper?q=%E4%BD%9C%E8%80%85:"Tom VanCourt"" \t "https://d.wanfangdata.com.cn/periodical/_blank),[Josh Model](https://s.wanfangdata.com.cn/paper?q=%E4%BD%9C%E8%80%85:"Josh Model"" \t "https://d.wanfangdata.com.cn/periodical/_blank),等.[Achieving High Performance with FPGA-Based Computing](https://d.wanfangdata.com.cn/periodical/457947b75fb1dceffc73873e4ab7dfcc" \t "https://d.wanfangdata.com.cn/periodical/_blank)[J].Computer.2007,40(3).