Hardware-Software Synthesis Project

Design and implementation of a parallel computing processor

University of Kaiserslautern
Embedded Systems Group
11.04.2014

Carsten Harms
Amir Kabouteh
Navid Dorosti
Ali Sabeghi

Overview

- Introduction
- Assembler and Instruction set
- Communication(Host\PowerPC)
- Design
- Implementation

Introduction

The goal for this project is to develop a Hardware/Software system for parallel numerical computation.

- Software Components
  - Assembler

    Translates human readable code to instructions.
  - Client software

    Communication between the host and PowerPC using TCP/IP
- Hardware Components
  - Parallel Computing Processor (PCP)

    The designed processor is executing 64 warps of 4 Threads in parallel.
  - Bus Communicator

    Sets the mode and manages the transaction between PCP and Device Memory (by Ali)

Assembler and Instruction set

- Assembler is written in Python, using regular expressions
  - Assembler instruction format:

label	*condition*	*operation*				
l1 :	c1 ?	add	r5	r3	r1	# comment
		eql	r3	r4	c2	c3
		jac	l1			

- Instruction format is inspired by MIPS and extended and modified for our needs
  - Makes instruction decoding very simple

Instruction Format

constrains:

- 3 bit condition register address
- 5 bit general purpose register address

R-Type	cond	opcode	target	src 1	src 2	unused	
MEM	cond	opcode	target	source	unused		
I-Type	cond	opcode	target & source	16-bit immediate			
J-Type	cond	opcode	unused	address			
C-Type	cond	opcode	unused	src 1	src 2	c target 1	c target 2
Special	cond	opcode	unused				

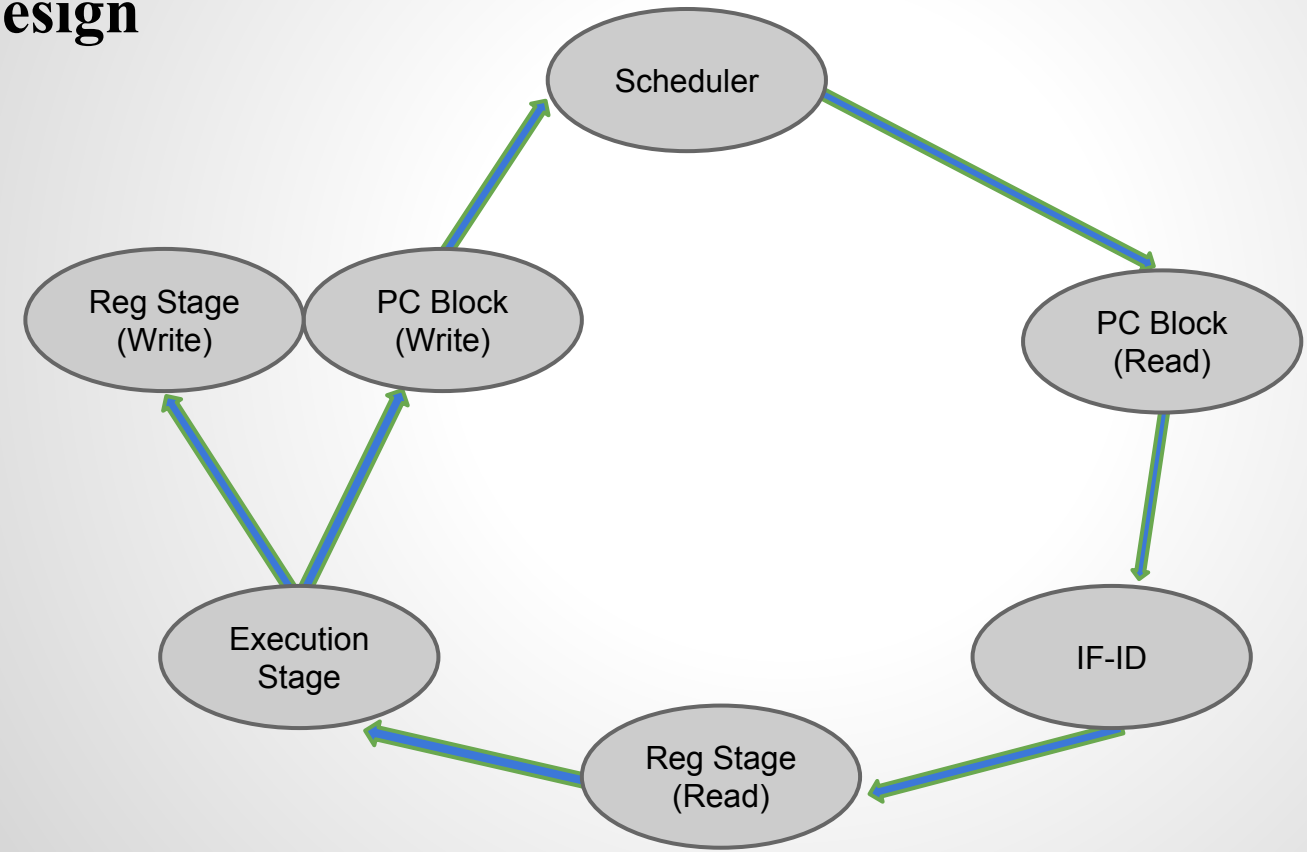
Communication Host-PowerPC

- Client is written in Python as a command line interpreter
- Two TCP connections are used, each dedicated to
  - commands
  - data
- Commands are transmitted as human readable ASCII
  - Examples: CONNECT, EXIT, INFO, PROGRAM
  - easily extendable and debugging friendly
- Data is send as a binary stream and is meant to be stored in previously allocated memory blocks on the device memory

Design

- System has three modes
  - Programming
  - Running
  - Reading
- 6 Stage Pipeline
  - Scheduler
  - PC Block Read
  - IF-ID
  - Register Read
  - Execution
  - PC Block-Register Write
- each stage executes one warp which consists of four threads

Design


Scheduler
PC Block (Read)
IF-ID
Reg Stage (Read)
Execution Stage
Reg Stage (Write)
PC Block (Write)

Scheduler

- Schedules the active warps
- One FIFO queue of size 64
- Enqueue active warps which are not in the pipeline
- Dequeue inactive warps or those which are currently in the pipeline
- Keeps the Warp Threads Status
- Changes made in the execution stage and will be pipeline to the scheduler
- This unit feeds the whole pipeline, in case it does not work the pipeline will do a NO_OP
- Being activated by the control unit

PC Block

- Stores the Program Counters for all threads of all warps
- Performs three steps in one clock cycle in parallel
  - Outputting the PC of active threads in the scheduled warp
  - Modifying the PC of active threads in the executed warp
  - Sending the updated values to the scheduler

IF-ID Stage

- IF-ID In programming mode
  - Instruction memory can be accessed by the Communicator Module-Write enable by Control unit
- IF-ID In running mode
  - IF fetches the instruction corresponding to the input PC and pass it to ID
  - ID decodes the instruction and sends different parts either to outputs or pipeline registers

Register Stage

- Register Read Contains two Sets of Registers.
  - Normal data registers, one mini Register file (32 Registers) for each thread of all warps.
  - Condition Registers, one Register file(8 1-bit Registers) for each thread of all warps.

For Normal Data Registers

- FPGA offers many dual port Block RAM, which are used to implement the register files
  - Dual port is not enough, we require two reads and one write simultaneously
  - Solution => duplicate the register sets for each thread. Write to both of Register files and read the first address from register No.1 and Second address from register No.2.

For Condition Registers

- Similar to Data Registers but we need one read and two writes simultaneously
  - For Writes: First read the value we want to change, make changes to the read value. Write it back to both Condition Register files. Both files will stay Consistent
  - For Read: Always read from the same condition register file

Execution Stage

- Consists of 4 execution Units and a Scratchpad Memory
- Handles the load and store instructions from and to the Scratchpad Memory
- Calculates thread control values according to the type of instruction and inactive signals which come from execution units and will send them to scheduler

- Execution units
  - Execute four threads in parallel
  - Non-intelligent and straightforward result computation
  - Compute New PC's and send them through pipeline to PC block

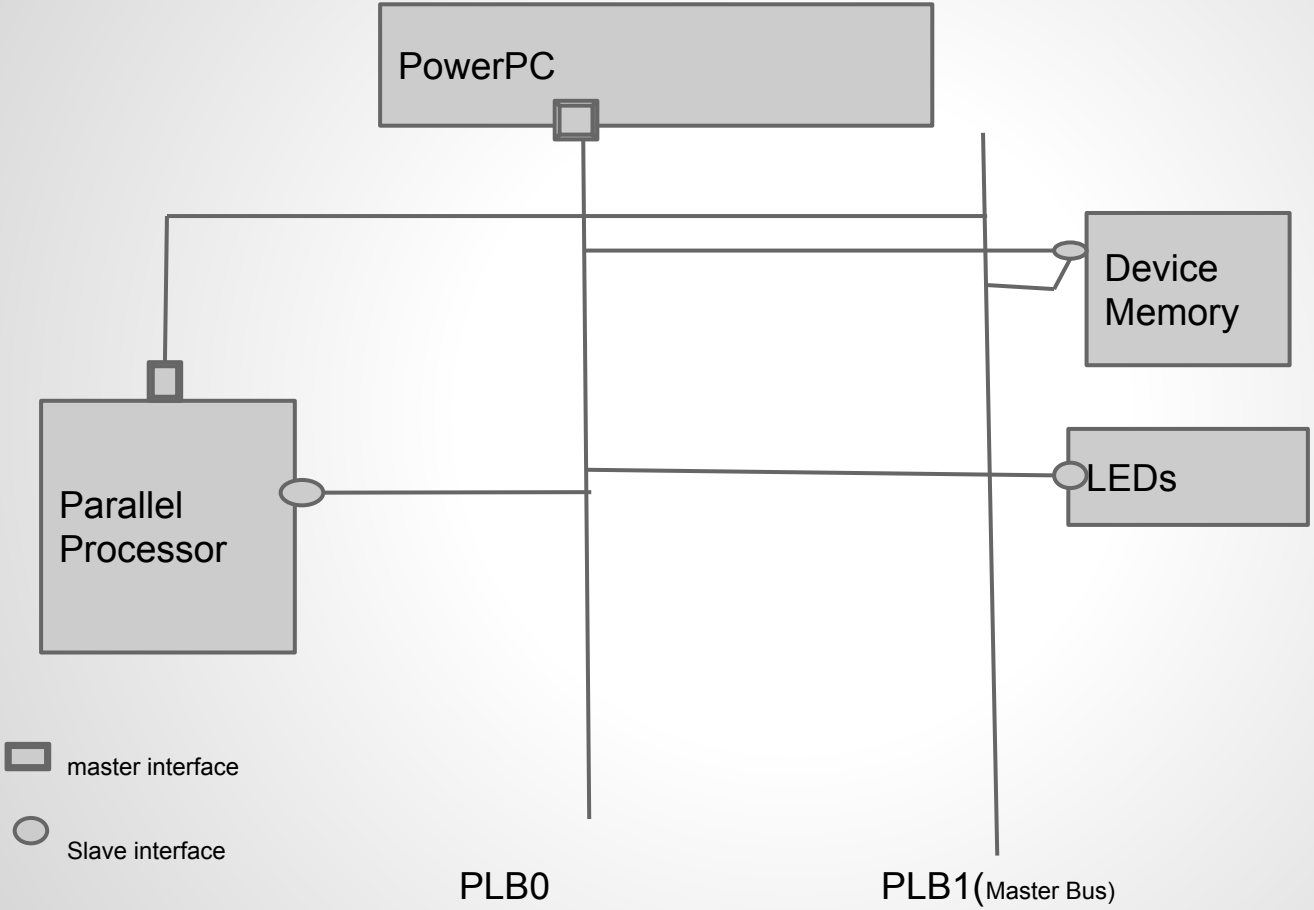
Write back to Register and PC-Block

- Updating PC values
- Writing to the register files for data and condition registers
- Sending back the finished warp address and New PC's so that the Scheduler can check the active threads and enqueue or dequeue it.

PowerPC-PCP Communication

- Bus configuration of the system
- Communicator unit .
- Phases of PowerPC-PCP communication

PowerPC-PCP Communication . Bus Configuration



PowerPC-PCP Communication .Communicator Module

Communicator modules Tasks:

- Sets the working mode of PCP (Programming,Running ,Result)

- Manages the Communications between PCP and other components of system(PowerPC ,Device Memory)

PowerPC-PCP Communication,PCP Programming Mode

Parallel Processor works in 3 different modes

- *In Programming mode:*

➢ In Programming mode , the Software sets the Software_Accesible_Register of Communicator Module via slave bus

➢ The Software Sets the Instruction memory start and end registers which show the the start and End of the Instruction memory block in Device memory via slave bus

➢ The Communicator unit, start to send sequence of ***read*** requests on master bus and puts the data on Instruction memory of Parallel Processor .

PowerPC-PCP Communication, PCP Running Mode

❖ *In Running Mode:*

➢ If there is a Store instruction ,Communication unit gets the data and its address from the execution unit and sends a write request on bus (master bus)

➢ If there is Load instruction , Communicator gets the address from execution unit and send the read request on bus and waits for validation and then puts the data on Scratch pad.

PowerPC-PCP Communication,PCP Result Mode

- *In Result Mode:*
  - The Communicator ,Starts to read the data of Scratch pad and sends of sequence of write request on bus

The Registers in Communicator which show the start and end of data block on Device memory are set by the Software via slave bus

End

Thank you for your Attention