The Heterogeneous Architecture Research Prototype (HARP)

Chad Kersey, Hyesoon Kim, S.Yalamanchili
Georgia Institute of Technology

Agenda

- Motivations
- Design Objectives
- The HARP Infrastructure
- The HARP ISA
- The HARP Compiler
- Harmonica Microarchitecture
- Assignment 4: Mini-Harp
- Questions?
Motivations

• Application performance and efficiency is largely constrained by the memory system
  - DRAM latency bottleneck
  - DRAM bandwidth constrained by pins saturation

• The influx of memory bound apps
  - Low compute to memory access ratio
  - Poor spatial and temporal locality
  - Irregular control flow

• Processing in Memory (PIM) challenges
  - Limited area in logic layer
  - Low power requirements

Design Objectives

• Area and power constraints
  - Compact and efficient microarchitecture
  - Reduce Instruction Set

• Effective memory bandwidth
  - Latency hiding via parallelism
  - Bandwidth saturation via concurrency

• Parametrization
  - Design space exploration (area vs power)
  - Scale the design to domain specific applications
  - Hardware prototyping (e.g. FPGA)
The HARP ISA

- Simple, RISC like
  - 64 opcodes (register and immediate operand)
  - General purpose and predicate registers
  - e.g. `addi %r0, %r0, #3`

- Full Predication
  - e.g. `@p0? addi %r0, %r0, #3`

- SIMT Oriented
  - Control divergence
  - Warp control instructions
  - Barrier synchronization

- Customizable
  - ArchID: 4w8/8/16/16

---

The HARP ISA (2)

- Instruction Encoding
  - Word/Byte encoding
  - Little endianness

- Privilege Instructions
  - Interrupts (ei, di, reti, halt)
  - Kernel (skep, jmpnu)
  - TLB (tlbadd, tlbrm, tlbflush)

- Memory Loads/Stores
  - ld/st

- Predicate Manipulations
  - andp, orp, xorpp, notp
The HARP ISA (3)

- **Value Tests**
  - rtop (!= 0), isneg (< 0), iszero (== 0)

- **Arithmetic Instructions**
  - Immediate Integer
  - Register Integer
  - Register Fixed/Floating-Point

- **Control Flow**
  - jmpi, jmpsr, jali, jalr

- **SIMD Control**
  - clone
  - Jalis, jalrs, jmprt
  - split/joint

The HARP ISA (4)

- **Warp Control**
  - wspan
  - bar

- **User/kernel**
  - trap

- **Interrupts**
  - Lane 0 is specialized for interrupts handling

- **ABI**
  - Stack pointer and link register are highest numbered registers
  - Frame pointer optionally follows
  - Calle manages the stack and frame pointers.
  - Function arguments use temp registers or stack
The HARP Infrastructure

- Harp Compiler
  - Clang Extension
  - LLVM Backend

- Harp Runtime

- Harp Tool
  - Assembler
  - Disassembler
  - Linker
  - Emulator

- Harmonica
  - FPGA / Simulation

The HARP Compiler

- Clang Front-End
  - C language extension for kernel parameters
  - e.g. `__attribute(harp_kernel) int foo(`
    `__attribute(warp_id) int wid,`
    `__attribute(lane_id) int lid, int val) {...}

- LLVM Back-End
  - Two targets: harp32 and harp64
  - LLVM IR to HARP assembly
The HARP Compiler (2)

- Instruction Selection
  - DAG Creation / Lowering / Scheduling
- Register Allocation
  - Restriction, Spilling, Frame index elimination
- Peephole Codegen Pass
  - Pseudo instructions to HARP instructions
- Control Divergence Pass
  - Predication, split-joint insertion
- Frame Lowering
- Asm Printer

The HARP Compiler (3)

- Predication
  - Converts control dependencies into data dependencies
  - Done during If-conversion pass after register allocation
  - e.g. if (%r1) %r2++ else %r2--:
    ```
    rtop $p0, $r1
    $p0 ? addi $r2, $r2, #1
    notp $p0, $p0
    $p0 ? subi $r2, $r2, #1
    ```
- Split-Joint
  - Hardware stack based control divergence
  - Divergent loops not supported
  - Done after If-conversion
- Decision framework
  - Static – operands def-use chains
  - Dynamic profiled guided
  - Split-join is best for unanimous branches
Harmonica Microarchitecture

• CHDL Implementation
  - C++ hardware modelling via template generators
  - Verilog codegen for FPGA

• Simplified Pipeline
  - No thread blocks
  - Single Issue Warp scheduler
  - SRAM based register file with single read/write ports
    - No bank parallelism

• Warps formation (nested parallelism)

Harmonica Microarchitecture (2)

• Pipeline stages
  - Schedule, Fetch, PredRegs, GPRgs, Exec, WriteBack

(13)

(14)
Harmonica Microarchitecture (3)

- Scheduler: FIFO circular queue

Harmonica Microarchitecture (4)

- Instruction Fetch
Harmonica Microarchitecture (5)

• Predicate Register File Access

Harmonica Microarchitecture (6)

• General Purpose Register File Access
Harmonica Microarchitecture (7)

- Functional Unit Output

Harmonica Microarchitecture (8)

- Register File Writeback
Harmonica Microarchitecture (9)

- Next Instruction Fetch

Harmonica Microarchitecture (10)

- Pipeline Stalls
Assignment 4: Mini Harp Emulator

• Minimal ISA
  - Word encoding
  - Integers only
  - A single predicate register
  - No Split-Join
  - No interrupts
  - No virtual addressing

• Instructions Set
  - Nop, Add, Sub, And, Or, Xor, Not, Shr, Shl, Ld, St, Clone, Bar

• Configuration
  - Register size, warp size, number of warps

Assignment 4: Mini Harp Emulator (2)

• Emulator baseline
  - System Core
  - Memory Subsystem
  - Instruction Decode
  - Instruction Fetch

• You Implement
  - Register File
  - Execute Stage
  - Print Stats

• Provided
  - Code base, sample programs
Questions?