Supporting CUDA for an extended RISC-V GPU architecture

Reporter: Ruobing Han
Date: 05/30/2021

co-author: Blaise Tine, Jaewon Lee, Jaewoong Sim, Hyesoon Kim
Motivation

High-level programmers

Low-level programmers
Pipeline for running CUDA on the RISC-V GPU

Features:
• Lightweight
• Scalable
• Open-Source

Reference: "Bringing OpenCL to Commodity RISC-V CPUs" CARRV 2021
Background: code migration

- HIPIFY: Only for CUDA->HIP
- SYCL: Re-implement CUDA programs by SYCL libraries

Background: SPIR

The SPIR-V ecosystem includes a rich variety of language front-ends, tools and run-times

source: https://www.khronos.org/spir/
OVERVIEW OF THE PIPELINE: CUDA->NVVM IR

```c
__global__ void vecadd (int *a, int *b, int *c) {
    int gid = threadIdx.x;
    c[gid] = a[gid] + b[gid];
}
```

**Code 2: NVVM VectorAdd IR**

```c
target triple = "nvptx64-nvidia-cuda"

define dso_local void @vecadd(
    i32* nocapture readonly %a,
    i32* nocapture readonly %b,
    i32* nocapture %c)
{
    entry:
    ; see Fig. 2(a) for detail code
}

declare i32 @llvm.nvvm.read.ptx.sreg.tid.x()

!nvvm.annotations = !{"\n!3 = !\{void (i32*, i32*, i32*) *,@vecadd, !"kernel", i32 1}\n```
OVERVIEW OF THE PIPELINE: NVVM IR->SPIR-V IR

- NVVM IR
  - Device independent instruction
  - NVVM built-in function declaration
  - Meta-data information (For NVVM)

- SPIR-V IR
  - Device independent instruction
  - SPIR-V built-in function declaration
  - Meta-data information (For SPIR-V)

Github link:
Example for NVVM-SPIR-V translation

For device independent instructions (load, store, binaryOp...): use OpenCL-SPIR-V translator
For NVVM special built-in function (llvm.nvvm.read.ptx.sreg.tid.x(...)...): use NVVM-SPIR-V translator
OVERVIEW OF THE PIPELINE: SPIR-V->OpenCL IR

SPIR-V

define spir_kernel void @\_Z6vecaddPiS_S_i32* %a, i32* %b, i32* %c) {
    entry:
    %0 = call spir_func i64 @\_Z12get_local_id\_l(i32 0) #1
    %1 = trunc i64 %0 to i32
    %idxprom8 = zext i32 %1 to i64
    %arrayidx = getelementptr inbounds i32, i32* %a, i64 %idxprom8
    %2 = load i32, i32* %arrayidx, align 4
    %arrayidx2 = getelementptr inbounds i32, i32* %b, i64 %idxprom8
    %3 = load i32, i32* %arrayidx2, align 4
    %add = add i32 %3, %2
    %arrayidx4 = getelementptr inbounds i32, i32* %c, i64 %idxprom8
    store i32 %add, i32* %arrayidx4, align 4
    ret void
}

declare spir_func i64 @\_Z12get_local_id\_l(i32) #1

OpenCL IR

SPIRV-LLVM-Translator: https://github.com/KhronosGroup/SPIRV-LLVM-Translator
OVERVIEW OF THE PIPELINE: OpenCL->Object file

POCL for Vortex: https://github.com/vortexgpgpu/pocl
Vortex: https://github.com/vortexgpgpu/vortex
Experiments

<table>
<thead>
<tr>
<th>application</th>
<th>feature</th>
<th>support?</th>
</tr>
</thead>
<tbody>
<tr>
<td>b+tree</td>
<td>-</td>
<td>yes</td>
</tr>
<tr>
<td>bfs</td>
<td>-</td>
<td>yes</td>
</tr>
<tr>
<td>cfd</td>
<td>double3 type</td>
<td>yes</td>
</tr>
<tr>
<td>huffman</td>
<td>atomic</td>
<td>yes</td>
</tr>
<tr>
<td>pathfinder</td>
<td>memory hierachy</td>
<td>yes</td>
</tr>
<tr>
<td>gaussian</td>
<td>-</td>
<td>yes</td>
</tr>
<tr>
<td>hotspot</td>
<td>-</td>
<td>yes</td>
</tr>
<tr>
<td>hotspot3D</td>
<td>-</td>
<td>yes</td>
</tr>
<tr>
<td>lud</td>
<td>memory hierachy</td>
<td>yes</td>
</tr>
<tr>
<td>nw</td>
<td>-</td>
<td>yes</td>
</tr>
<tr>
<td>streamcluster</td>
<td>-</td>
<td>yes</td>
</tr>
<tr>
<td>particlefilter</td>
<td>d2i</td>
<td>on going</td>
</tr>
<tr>
<td>backprop</td>
<td>__log2f</td>
<td>on going</td>
</tr>
<tr>
<td>lavaMD</td>
<td>d2i</td>
<td>on going</td>
</tr>
<tr>
<td>kmeans</td>
<td>texture</td>
<td>no</td>
</tr>
<tr>
<td>hybrid sort</td>
<td>texture</td>
<td>no</td>
</tr>
<tr>
<td>leukocyte</td>
<td>texture</td>
<td>no</td>
</tr>
</tbody>
</table>

Table 2: Translating applications in Rodinia benchmark

Version:
Vortex-pocl: commit log 6272e2
Vortex: v0.2.2
NVPTX-SPIR-V translator: v0.1.0

Analysis:
• Support built-in function is the key to support general programs
• Well support for primitive instructions
• For new features, need to modify both translators and RISC-V library for Vortex

Rodinia: https://github.com/yuhc/gpu-rodinia
Support feature (NVPTX-SPIR-V translator: version v0.1.0, Vortex: version v0.2.0)

- Memory hierarchy
- Synchronization
- Some mathematics operations (square root, log, absolute value)

<table>
<thead>
<tr>
<th>function name</th>
<th>detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>llvm.nvvm.read.ptx.sreg.ctaid</td>
<td>get the block index</td>
</tr>
<tr>
<td>llvm.nvvm.read.ptx.sreg.ntid</td>
<td>get the block dimension</td>
</tr>
<tr>
<td>llvm.nvvm.read.ptx.sreg.tid</td>
<td>get the thread index</td>
</tr>
<tr>
<td>llvm.nvvm.barrie</td>
<td>synchronize threads within a block</td>
</tr>
<tr>
<td>llvm.sqrt</td>
<td>calculate the square root</td>
</tr>
<tr>
<td>llvm.fabs</td>
<td>calculate the absolute value</td>
</tr>
<tr>
<td>llvm.nvvm.d2i</td>
<td>narrowing conversions</td>
</tr>
<tr>
<td>llvm.fma</td>
<td>fused multiply–add</td>
</tr>
</tbody>
</table>
Conclusion

• We propose and implement a lightweight, scalable pipeline for running CUDA source code on an open-source RISC-V GPU
• We build a translator to convert from NVVM IR to SPIR-V IR
• We can execute most of sample in a widely used benchmark

contact information: vortexgpu@cc.gatech.edu