Memory-Aware Compilation for CGRAs

Aviral Shrivastava
Compiler Microarchitecture Lab
Arizona State University

Work done in collaboration with SO&R Lab, SNU, South Korea
The Road of Power-Efficiency

- Need for high performance
  - Cannot increase power
  - Power and thermal issues
- Exa-scale Computing
  - $10^{18}$ Ops/20 MW = 50 Gops/W
- Power-Efficiency is the key design metric
  - UNIVAC: 0.015 ops/W
  - Dual core Athlon 64 x2: 58 Mops/W
  - IBM's Roadrunner: 376 Mops/Watt
  - GeForce 9800 GX2: 6 Gops/W
  - Intel 80-core: 16 Gops/W
- Coarse Grain Reconfigurable Arrays
  - Up to 100s of Gops/W
  - Power-efficiency scales to a wider set of applications
Coarse-Grained Reconfigurable Array

- 2D array of PEs
- PE operates on the result of neighboring PEs
- Pipelining, routing, scheduling, everything in software
  - Minimal power overhead

10/27/2010
http://www.public.asu.edu/~ashriva6
CGRA as a coprocessor

- Traditionally streaming applications
  - Samsung TVs
- Offload the computationally intensive innermost loop kernels onto the CGRA
  - Like we use GPUs for computing
  - Has direct link to the memory
Compilation for CGRAs 101*

• Assume ideal memory
  – Assume all data needed is in the DM
  – Assume a large unified memory
  – Assume any PE can perform load/store

*SPKM: ASPDAC 2008
1. PE Constraint

- Only a small subset of PEs can perform memory operations
  - Given a grid of PEs, only certain PEs have the hardware necessary to access local memory
    - Example: One PE in each column is allowed access to local memory
  - Memory operations should receive highest priority to these PEs during mapping
  - Current techniques are able to effectively work with this constraint
2. Finite Ports

- Local memory has finite ports
  - While multiple PEs are capable of performing memory operations, only a subset can be serviced at a time
    - Memory area and power ~ no. of ports
  - CGRA with 16 PEs may provide 4 PEs each with 2 load and 1 write port(s), but the memory can only handle 4 loads and 2 writes

- DMQ
  - DMQ of depth $K$ can tolerate up to $K$ instantaneous conflicts
  - Increases load latency to $K$ cycles
  - DMQ cannot help if average conflict rate > 1

- Expose to Compiler
  - Spread of memory operations
2. Finite Ports

Data-Dependency Graph

Exposed to Compiler Mapping

DMQ Mapping

Memory Port Arbitration Logic

Local Memory Buffer

http://www.public.asu.edu/~ashriva6
3. Bank Constraints

- Local memory is divided into separate banks
  - In order to provide more access ports, multiple banks of local memory are created, each with independent data and resources
- Example: A CGRA can provide 12 access ports through 4 separate memory stores
3. Bank Constraints*

• Execution Time
  – Tc: Computation time
    • Assuming all data is in local memory
    • Proportional to II
  – Td: Data transfer time
    • Increases with data duplication
  – Sum(Tc, Td) or at best Max(Tc, Td)

Data transfer time is the bottleneck for parallel loops

*Operation and Data … LCTES 2010
3. Bank Constraints

- Memory Bank Arbitration Logic
  - DMQ of depth K can tolerate up to K instantaneous conflicts
  - DMQ cannot help if average conflict rate > 1
  - Increases load latency to K cycles

- Expose to Compiler

Data-Dependency Graph:

\[ A[i] \rightarrow \text{Ld } 0 \rightarrow \text{Op } 3 \rightarrow \text{Ld } 2 \rightarrow \text{A}[i+1] \]

\[ \text{Ld } 1 \rightarrow \text{Op } 4 \]

Aware Mapping:

- Bank 1
- Bank 2
- Bank 3
- Bank 4

*Metal-Aware .., HIPEAC 2010*
4. Finite Size Constraint

- Local memory is of finite size
  - Local memory can only store a finite amount of data, and therefore buffer only a part of main memory
    - Example: A CGRA may have 4 banks of memory, each with 128KB storage
  - Data duplication may cause the actual amount of data storable in local memory to be less than the total amount of memory available
4. Finite Size Constraint

- Double Buffering

```plaintext
for (i, 0, N) {
    s += A[i];
}

b=0;
DMA (A, 0, T, L+bT);
for (j, 1, N/T-1) {
    DMA (A, j*T, (j+1)*T, L+(1-b)*T);
    for (i, (j-1)*T, j*T) {
        s += L+bT+4*i;
    }
    b = 1-b;
}
for (i, (j-1)*T, j*T) {
    s += L+bT+4*i;
}
```

System Memory
Summary

• CGRAs are a promising platform for highly power-efficient computing
  – 100s of Gops/W

• Real memories are complicated
  – Only some PEs may perform load/store
  – Only a few memory ports
  – Multi-bank memory
  – Limited size of local memory

• Performance limited by memory bandwidth
• Memory-aware compilation is important