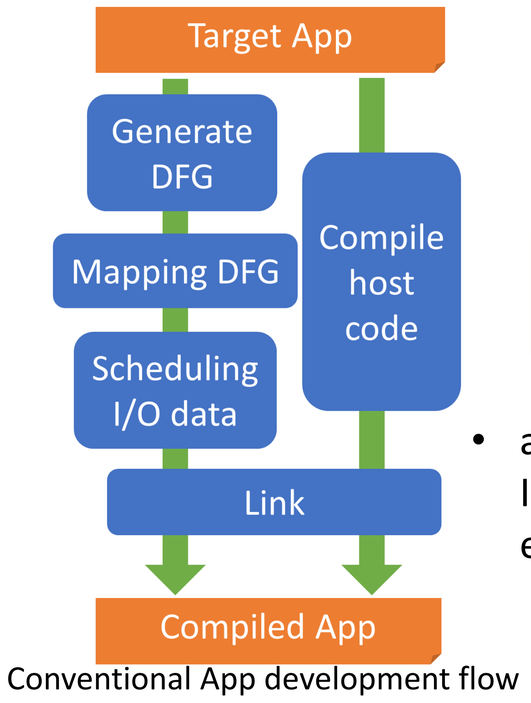
Compiler Framework for Spatial Mapping CGRA using LLVM

Ayaka Ohwada*, Takuya Kojima*, and Hideharu Amano*
*Keio Univ. JAPAN
E-mail: ohwawa@am.ics.keio.ac.jp

Introduction

Coarse-grained reconfigurable architecture (CGRA) is a type of reconfigurable device. There are several steps in CGRA application development, such as converting the target application to a data flow graph (DFG), mapping the DFG to a PE array, etc. In this study, we implemented and evaluated an application development environment using LLVM for the CGRAs which covers the entire system.

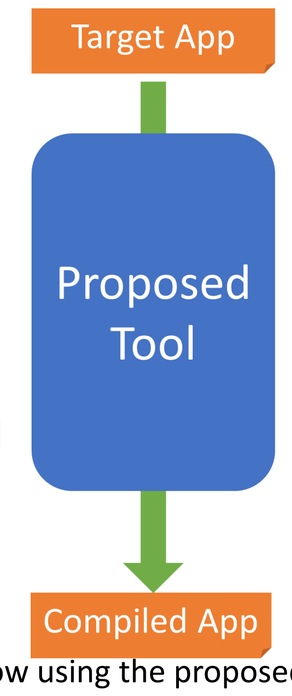
Background & Motivation



Conventional App development flow

Each step is separate.
Some are manual.

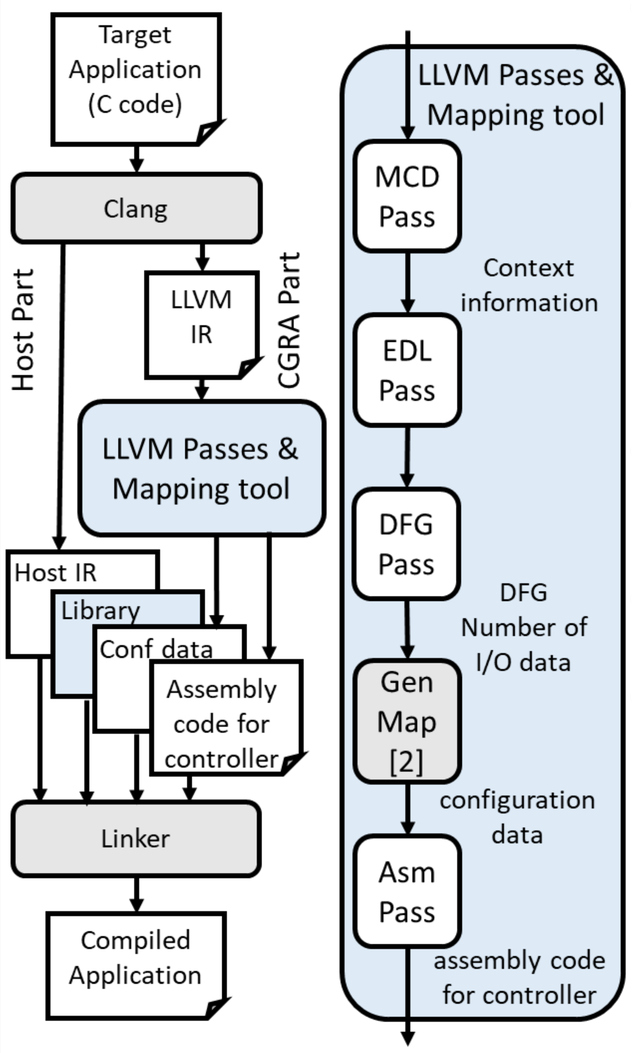
- automation and Integration of each process



Flow using the proposed tool

Our tool covers entire flow.

Proposed CGRA App Development Tool



Application development flow

```
int main() {
   int index =
      get_kernel_index("gray");
   cgra_setup(index);
   send_host_to_cgra(index, 1, data,
      48, 0);
   cgra_run(index, 0, 0, 0);
   wait_cma();
   send_cgra_to_host(index, 1, result,
      48, 0);
}
```

Sample code using CGRA Library

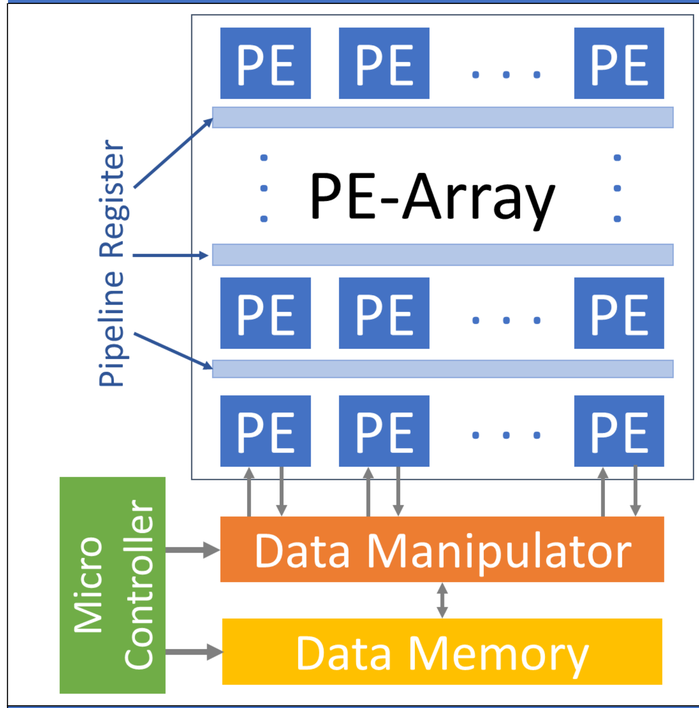
- **MCDPass**
  (Multi Context Data Pass)
  - Extracts each context information.
- **EDLPass**
  (Eliminate Duplicate Load Pass)
  - Eliminates duplicate load instructions which load from the same memory address.
- **DFGPass**
  - Analyses LLVM IR and outputs DFG in the *dot* format.
- GenMap[2]
  - Maps the DFG to the PE array.
- **AsmPass**
  - Generates assembly code for the memory access controller in CGRA.

```
    DELAY 2
    LDI r3, #48
    SET_LD #0x0, #2
    SET_ST #0x30, #1
LP: LDST_ADD #0, #0
    BNZD r3, LP
    DONE
```

Sample assembly code

- **CGRA Library**
  - Used to write code that runs on the CPU side.
  - Provides functions for controlling the CPU and the CGRA.
  - Encapsulates calculation of data transfer destination/source address.

VPCMA(Variable Pipelined Cool Mega Array)[1]

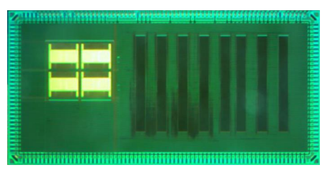


- PE(Processing Element)
  - Composed of
    1. Simple ALU
    2. Switching Element
  - No register file
  →No need of clock signal
- PE Array
  - 12 cols x 8 rows PEs
  - 7 pipeline registers
- Micro-controller
  - Controls data transfer b/w data memory & PE array

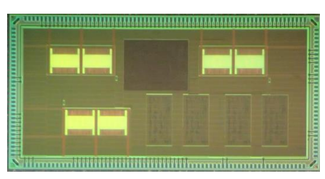
Evaluation

Evaluation Target

- CCSOTB2@20MHz
  - A real chip implementation of a VPCMA
- GeyserTT@20MHz
  - An embedded processor with a MIPS R3000 compatible CPU core.



CCSOTB2



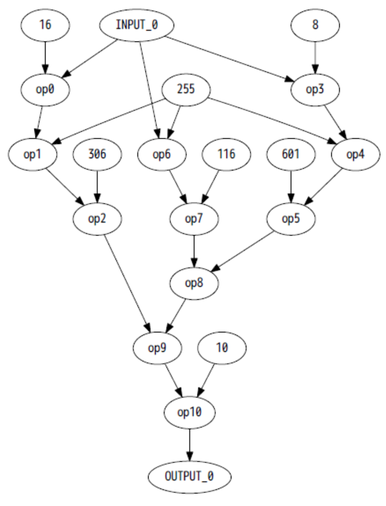
GeyserTT

Evaluation Environment

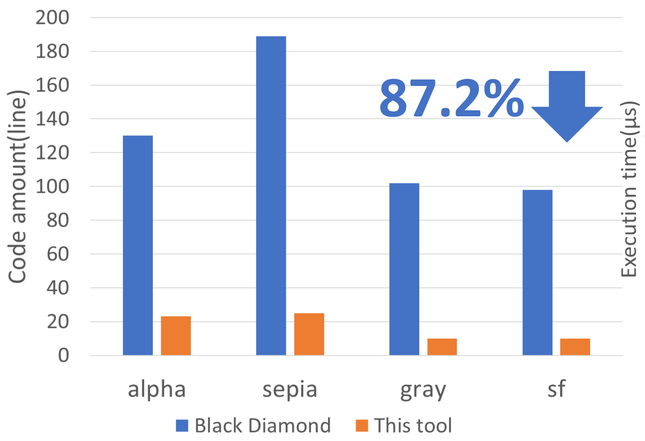
Design	Verilog HDL
Process	Renesas SOTB 65nm
Logic Synthesis	Synopsys Design Compiler 2016.03-SP4
Place and Route	Synopsys IC Compiler 2016.03-SP4
HDL Simulation	Cadence NC-Verilog 15.20-s020
Power-Supply Voltage	CCSOTB2:0.55V GeyserTT: 0.75V

Result

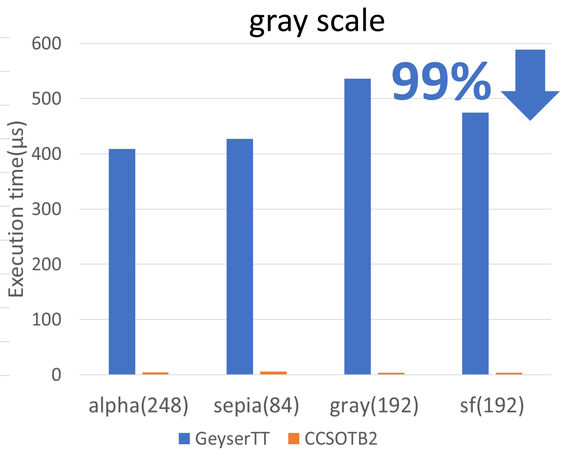
- Four applications were chosen for the evaluation.
  - alpha: 24bit alpha blender
  - gray: 24bit gray scale
  - sepia: 24bit sepia filter
  - sf: 8bit sepia filter
- All applications were confirmed to work properly.



The DFG to convert an RGB image to gray scale



Code amount for each kernel compared to [3]



Comparison of execution time b/w CCSOTB2 and GeyserTT

Reference

[1] Ando Naoki, et al. "Variable pipeline structure for coarse grained reconfigurable array CMA." FPT 2016.
[2] Takuya Kojima, et al. "Real chip evaluation of a low power cgra with optimized application mapping," HEART 2018.
[3] Vasutan TUNBUNHENG, et al. "A Retargetable Compiler Based on Graph Representation for Dynamically Reconfigurable Processor Arrays", IEICE Transactions on Information and Systems, 2008.