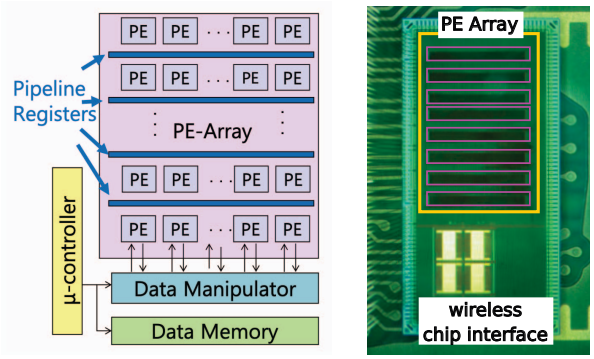
Demonstration of Low Power Stream Processing Using a Variable Pipelined CGRA

Takuya Kojima, Naoki Ando, Yusuke Matsushita, and Hideharu Amano
Dept. of Information and Computer Science, Keio University, Yokohama, Japan
Email: wasmii@am.ics.keio.ac.jp

***Abstract*—VPCMA (Variable Pipelined Cool Mega Array) is a low power CGRA (Coarse-Grained Reconfigurable Architecture) which we previously proposed in [1]. CC-SOTB2 is a real chip implementation of the VPCMA using Renesas 65-nm SOTB technology [2]. In this demonstration, we will show the power consumption of the CC-SOTB2 while performing a real image processing.**

I. The CGRA Architecture

Like typical CGRAs, the VPCMA has a large PE (Processing Element) array as described in Fig. 1(a). Instead of a host processor, execution of a compute-intensive part (mainly loops) is accelerated by leveraging the abundant computation resources of the PE array. Operation type of the PEs and interconnections between the PEs are reconfigurable depending on a target application.



(a) VPCMA diagram

(b) Chip photograph of CC-SOTB2

Fig. 1: VPCMA architecture and implementation

The PE does not contain registers, that is, it is composed of a pure combinational circuit. An immediate result provided by a PE is transferred to the adjacent PEs without latching. Therefore, a clock tree is not necessary for the PEs. It saves the dynamic power consumption of the PE array, yet may cause an extremely long critical path. Instead, a limited number of pipeline registers are placed and, a clock tree only for those registers is added. Each pipeline register is optionally activated so that they provide a variable pipeline structure. The clock signal of the unused pipeline registers is gated to cut off the dynamic power consumption. In this way, the VPCMA satisfies a wide range of performance requirement while managing the pipeline structure to minimize the power overhead.

II. Software Development Process

Although the VPCMA has a dedicated controller (μ-contoroller in Fig. 1(a)), it controls data transfer between data memory and the PE array. Thus, overall control including reconfiguration of the PE array has to be managed by a host processor. We are developing a programming environment as illustrated in Fig. 2. For a front-end compiler, we employ LLVM [3], which is a widely used compiler framework. A programmer writes a C code, and then, it is compiled to LLVM IR (Intermediate Representation). Loop kernels executed on the VPCMA's PE array are extracted from well optimized LLVM IR. Mapping the kernels to the PE array (configuration data) are generated by a genetic algorithm based optimizer [2]. Apart from the loop kernels, control part of the program is compiled for the host processor. Lastly, both programs are linked. So far, some part of this flow are processed manually. A fully automated compilation is left for future work.

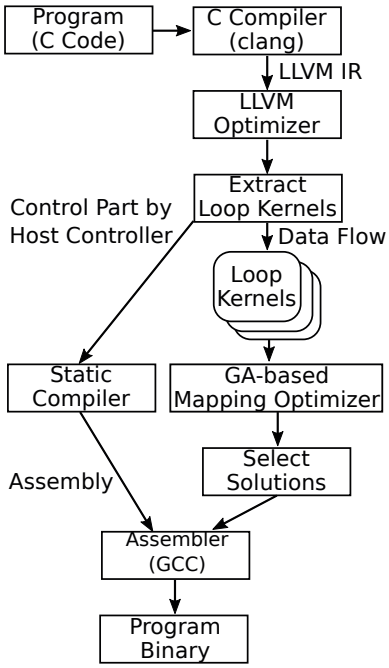


Fig. 2: Compilation Flow

1946-1488/19/$31.00 ©2019 IEEE
DOI 10.1109/FPL.2019.00071

III. Evaluation

We have implemented a real chip of the VPCMA named "CC-SOTB2" with Renesas SOTB 65-nm FD-SOI technology. Thus, body biasing can work efficiently. Fig. 1(b) is a chip photograph of the CC-SOTB2 [2]. To evaluate the power consumption and energy efficiency of the CC-SOTB2, we carried out some experiments with the CC-SOTB2 chip.

TABLE I: Evaluated application

Application	Description
gray	24 bit (RGB) gray scale
af	24 bit (RGB) alpha blender
sf	24 bit (RGB) sepia filter
sepia	8 bit sepia filter

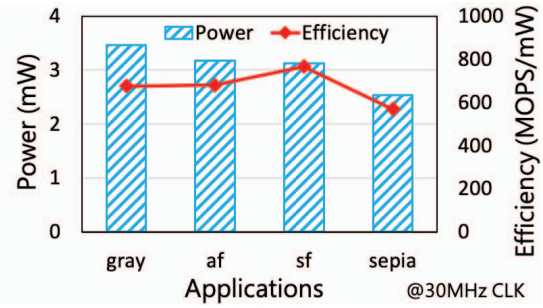


Fig. 3: Power consumption and energy efficiency

The experimental results are shown in Fig. 3. Four simple image processing applications listed in Table. I are selected as a benchmark and, operation frequency is set to 30MHz. For all applications, the CC-SOTB2 chip consumes only a few milliwatts. In the best case (*sf*), 766 MOPS/mW of the energy efficiency is achieved. Compared to a previous (no pipelined) version which we showed a demonstration[4], about 4x higher peak performance is achieved while keeping almost the same energy efficiency thanks to the variable pipeline structure.

(a) VDD=0.55 (b) VDD=0.45

Fig. 4: Example of processed real images

Fig. 4 shows real processed images (*gray*) at different two supply voltage and at 20MHz clock frequency. In case of 0.55 V VDD (standard supply voltage of the SOTB process), the PE array works correctly. However, if lower VDD (0.45 V) is supplied intentionally, timing fault occurs. Note that the mapping optimizer [2] decides the mapping and the pipeline structure to satisfy the timing constraint.

IV. Demonstration Environment

We built a demonstration system illustrated in Fig. 5. Zynq FPGA is employed as the host processor to control the CCSOTB2 chip and to handle an image captured by a camera. PL (FPGA) part of the Zynq plays a role of an interface between the CCSOTB2 and PS (CPU) part. Host program provided by the development environment runs on Xilinx PetaLinux 2018.2.

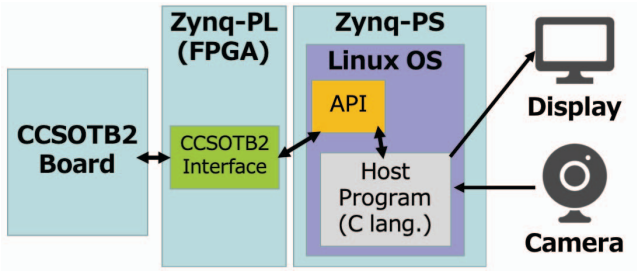


Fig. 5: System overview

We also developed a mother board for the above system as shown in Fig.6. To demonstrate the low power consumption, we will use an ordinary solar battery as the power source of the CC-SOTB2 chip.

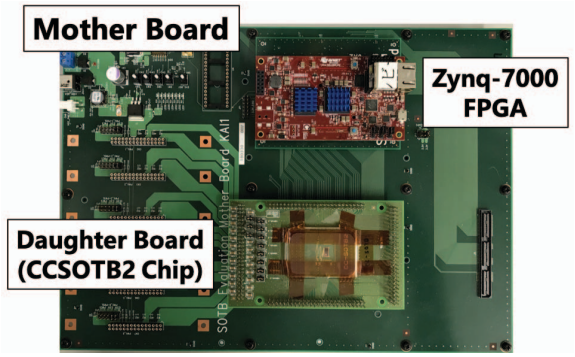


Fig. 6: Demonstration environment

References

[1] N. Ando *et al.*, "Variable Pipeline Structure for Coarse Grained Reconfigurable Array CMA," in *2016 International Conference on Field-Programmable Technology*, 2016, pp. 231–238.

[2] T. Kojima *et al.*, "Real chip evaluation of a low power cgra with optimized application mapping," in *Proceedings of the 9th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies*. ACM, 2018, p. 13.

[3] C. Lattner and V. Adve, "LLVM: A compilation framework for lifelong program analysis & transformation," in *Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization*. IEEE Computer Society, 2004, p. 75.

[4] K. Masuyama *et al.*, "7mops/lemon-battery image processing demonstration with an ultra-low power reconfigurable accelerator cma-sotb-2," in *2015 25th International Conference on Field Programmable Logic and Applications (FPL)*, Sep. 2015, pp. 1–1.