Background

Related work

Research scheme

Current progress

Conclusion

Loop Acceleration For Tightly-Coupled CPU+FPGA System

Cheng Liu
Supervisor: Dr. Hayden Kwok-Hay So
Co-supervisor: Dr. Ngai Wong

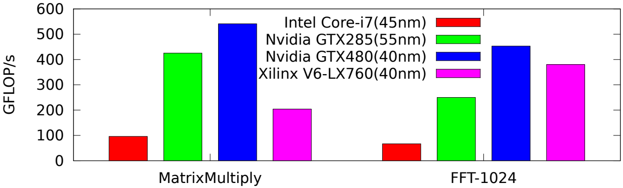
Department of Electrical and Electronic Engineering
The University of Hong Kong

November 27, 2014

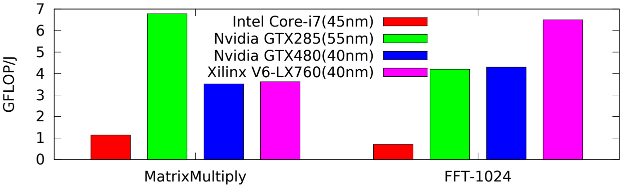
FPGA vs. CPU vs. GPU

FPGA has competitive computation capability and energy efficiency.

Performance comparison [1]



Energy efficiency comparison [1]



[1] Eric S. Chung, etc., Single-Chip Heterogeneous Computing: Does the future include customized logic, FPGA and GPGPUs?, IEEE International Symposium of Microarchitecture, 2010

Challenges and progress on FPGA computing

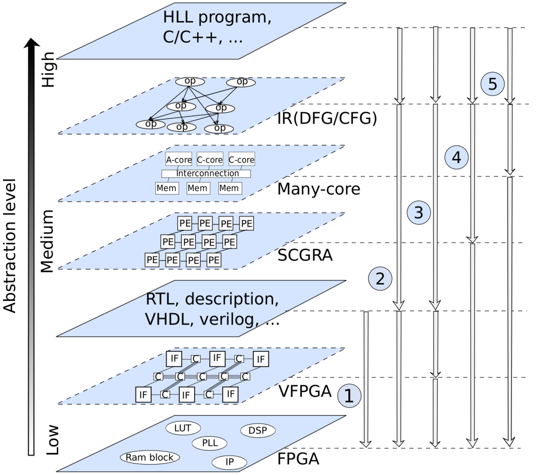
Challenges

- High barrier-to-entry (Hardware knowledge, ...)
- Low design productivity (Low abstarction level, long compilation time, ...)

Progress

- High level synthesis (HLS), e.g., LegUp, AotoESL, Impulse-C, ROCCC, ...
- Virtual overlays
  - ✓ Reconfigurable many-core, e.g., MARC, WPPA(Weakly programmable processor array), ...
  - ✓ Coarse-grained reconfigurable array, e.g.,QUKU, SCGRA, Heterogeneous CGRA, ...
  - ✓ Virtual FPGA, e.g., Intermediate Fabric, ZUMA, CARBON, MALIBU, ...
- Other techniques, e.g., Partial reconfigurable technique, Hard Macros, Communication library, ...

Differences and relations of the overlays



1 RTL design flow 2 Conventional HLS 3 VFPGA based HLS

4 SCGRA based HLS 5 Reconfigurable many-core architecture

SCGRA work in our group

What have been done?

- Introduced the SCGRA layer for HLS,
- showed potential design productivity improvement,
- and proved its energy efficiency using an application specific SCGRA topology

What are still missing?

- The relationship between a holistic loop and its kernel data flow graph,
- influence of the communication between CPU and FPGA on the SCGRA based HLS.

Focus of my work

- Automatic loop acceleration on a tightly-coupled CPU+FPGA using the SCGRA overlay

Why loop acceleration?

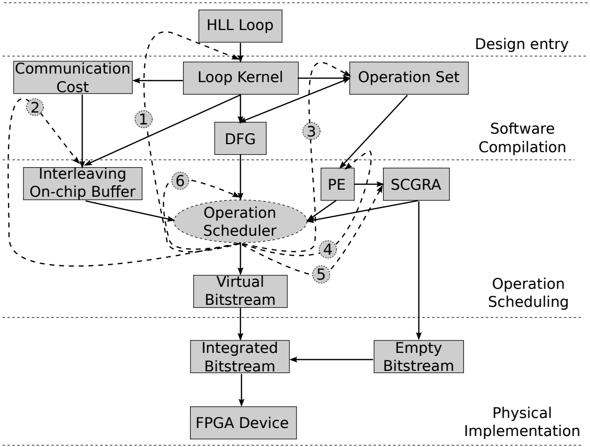
Loop and computation kernel

- Loops usually form the most computationally intensive kernel of a program
- Regularity of loops provide ample of data parallelism
- Loops are important optimization targets for the parallel computing architecturs including Multi-core processor, GPU, CGRA and FPGA.

Difference from previous work

- Hardware infrastructure (SCGRA and communication) is changing with the loop optimization
  - ✓ Not possible with hard CGRA
  - ✓ Take advantage of the softness of the FPGA
  - ✓ Application-specific buffering, loop unrolling, and scheduling

SCGRA based accelerator design flow



1 loop unrolling factors

2 On-chip buffer size, interleaving scheme, data fetching scheme

3 Primitive operations supported by the hardware infrastructure

4 PE pipeline depth, local memory port number and allocation

5 Topology of the computation array, array size

6 Scheduling algorithm, scheduling strategies

Optimal loop unrolling

Why loop unrolling and why not fully unroll the loop?

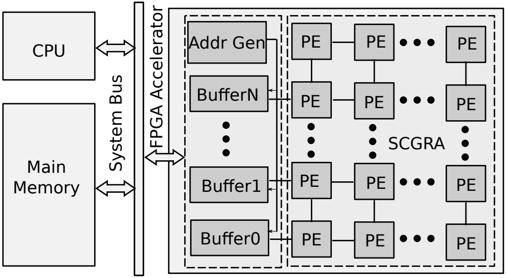
- Increases parallel operations and improves performance
- Induces larger hardware overhead
- Benefit may be limited by system constrains.

Loop unrolling problem

- Assumptions: Bounded loop, and data dependency known at compiling
- Input: Sequential program proportition, kernel DFG, loop iteration bound, ...
- Optimization target: Min(loop execution time/communication cost)
- Constrain: hardware overhead, IO bandwidth
- Model: Operation Scheduling model+Data prefetching model

Hardware infrastructure

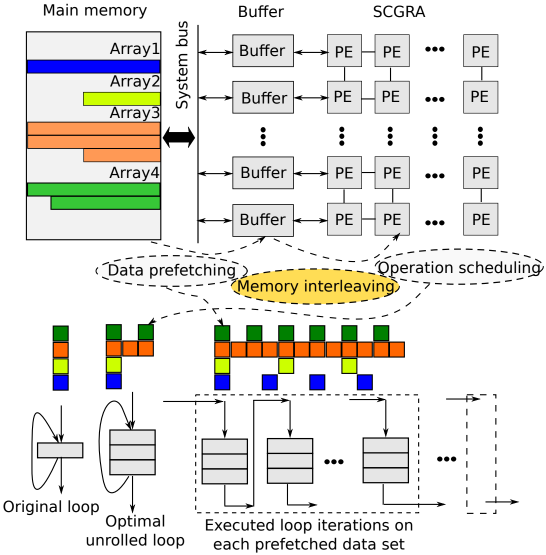
SCGRA based CPU+FPGA accelerator



Softness of the accelerator

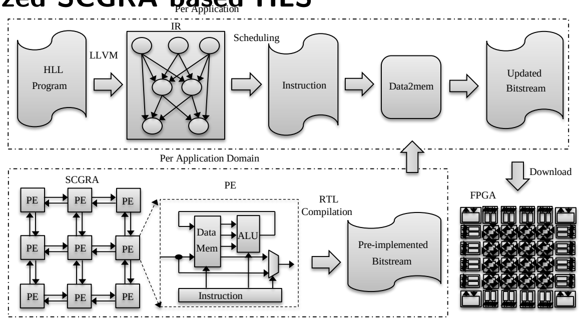
- SCGRA structure could be reconfigurable
- On chip buffer could be reconfigurable

On-chip buffering

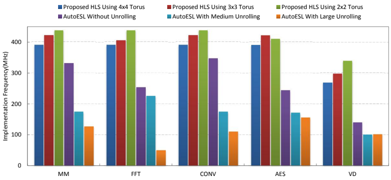


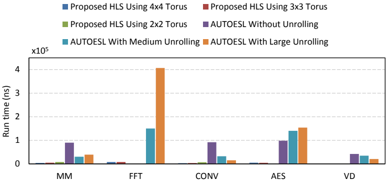
SCGRA based HLS optimization for both design productivity and frequency

Optimized SCGRA based HLS



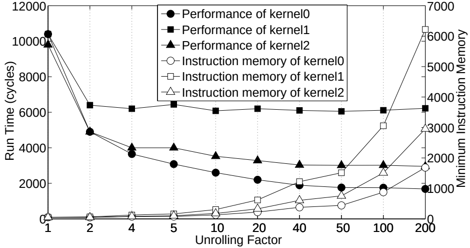
Experiment results



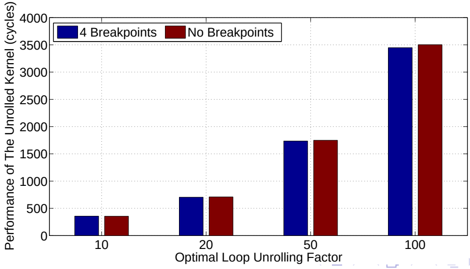


Preliminary loop unrolling analysis

loop unrolling influence on performance and overhead

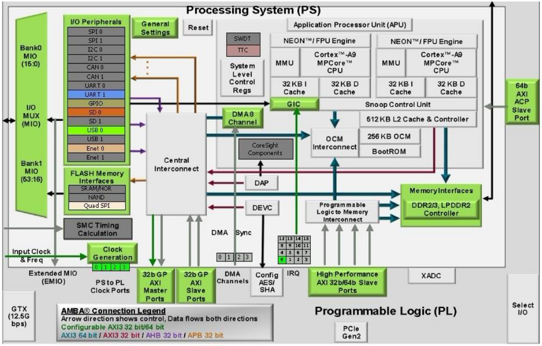


Irregular loop bound



HW/SW communication on Zedboard

Zedboard platform



Different communication methods

- Accelerator coherence port
- Central DMA, Video DMA, XDMA
- GPIO

Conclusion

Potential contribution

- Analyze the relationship between loop and its kernel data flow graph. Hopefully, an optimal partial loop unrolling may help resolve the BRAM-consuming problem in previous work.
- Provide a systemic solution to loop acceleration on a CPU+FPGA system and therefore a more friendly high level interface to the end users.