OpenCL-Based High-Performance 3D Stencil Computation on FPGAs

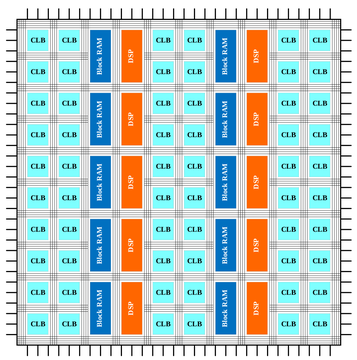
Hamid Reza Zohouri*, Artur Podobas*, Naoya Maruyama†, and Satoshi Matsuoka*
*Tokyo Institute of Technology, †RIKEN Advanced Institute for Computational Science
Contact: zohouri.h.aa@m.titech.ac.jp

Summary

Due to the architectural advantage of FPGAs for stencil computation, temporal blocking achieves better scaling than GPUs. This allows FPGAs to achieve comparable performance to that of high-end GPUs, despite having multiple times lower memory bandwidth.
Multiple previous work on FPGAs [1,2] achieved such level of performance by employing temporal blocking *without* spatial blocking, which requires restricting the size of input dimensions. We show that it is possible to combine spatial and temporal blocking to avoid such restrictions, and still achieve a high level of performance.

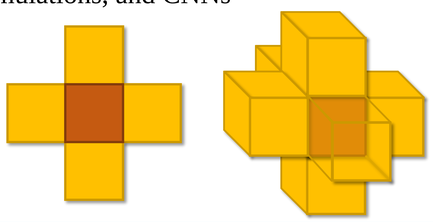
Field-Programmable Gate Arrays (FPGA)

- Reconfigurable devices that consist of:
  - Configurable Logic Cells (CLB)
  - Hard blocks like Block RAMs and DSPs
  - Controllers like DDR, PCI-E, Ethernet
  - Configurable interconnect
- High flexibility and low power consumption



Stencil Computation

- Grid cells are updated based on the value of their neighboring cells
- Used in solving PDEs, weather and seismic simulations, and CNNs

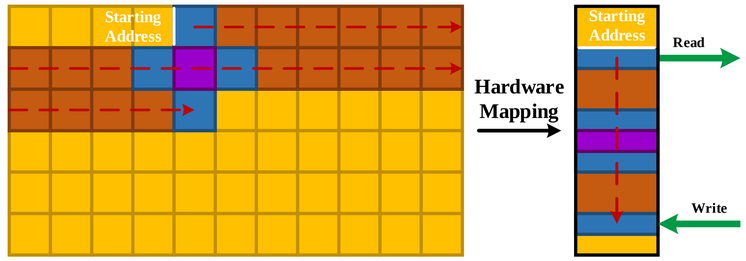


Typical Optimizations for Stencil Computation

Spatial Blocking: Stencil computation exhibits good spatial locality. This can be exploited by caching neighboring cells in on-chip memory to avoid redundant off-chip memory accesses.
Temporal Blocking: Stencil computation also has good temporal locality. Intermediate data between subsequent iterations can be saved in on-chip memory and reused, rather than passed through off-chip memory.

Spatial Blocking on FPGAs

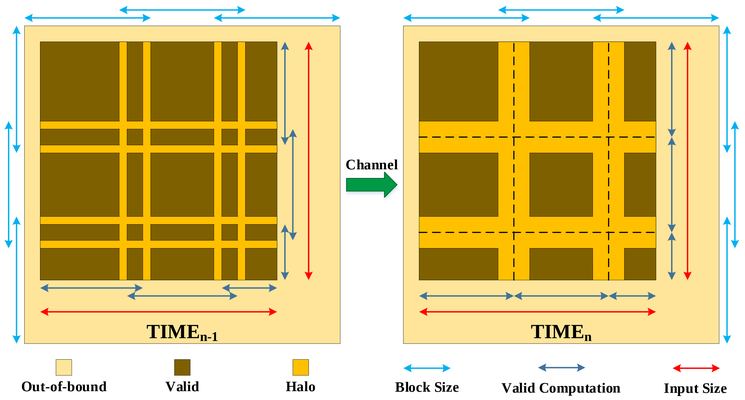
- Neighbors are accesses sequentially
- Can be cached using a shift register-type storage
  - Rather than multi-ported RAM
- Maps well to FPGA Block RAMs
- Incrementing the buffer's starting address shifts the stencil forward
- New cells are loaded to the head of the shift register, and old ones are evicted from the tail



Compared to spatial blocking on CPUs and GPUs which involves keeping a full rectangular block in on-chip memory, this architectural advantage of FPGAs allows achieving the same goal with a much lower on-chip memory requirement.

Temporal Blocking on FPGAs

- Single work-item deep-pipelined design
- Decoupled memory operations and compute
- Multiple replicated compute modules running in parallel
  - Defined as *autorun* kernel for easy replication
  - Each processes the same spatial block of a different time step
  - Each is one plane behind the previous one
  - Connected using on-chip channels
- Computation starts from top-left in *XY* plane and is streamed through *Z* dimension
- Overlapped blocking is used to avoid synchronization
- Redundant computation increases towards the last compute block



Due to lack of threads, barriers, and fixed warps in this design, the problem of "thread divergence" does not exist and warp specialization is not required. This allows better scaling with temporal blocking on FPGAs, compared to CPUs and GPUs.

Advanced FPGA-Specific Optimizations

Loop Collapse: Nested loops are merged to avoid area waste for keeping variable states.

```
for y from 0 to m
  for x from 0 to n
    compute(x,y);
```

```
while y != m {
  compute(x,y);
  x = (x+1)%n;
  if (x == 0)
    y++; }
```

Exit Condition Optimization: Critical path is shortened by using a global index as loop exit condition, instead of a chain of dimension and block variable updates

```
while index != m*n {
  compute(x,y);
  x = (x+1)%n;
  if (x == 0) y++;
  index++; }
```

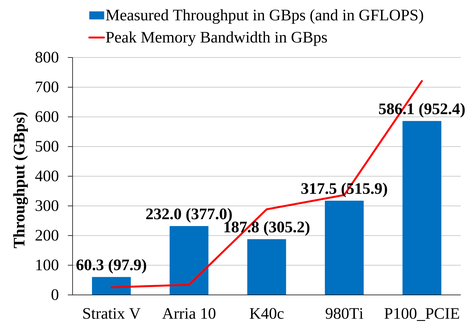
Padding: Negative starting address and block overlapping results in non-aligned memory accesses. Padding device buffers allows improving alignment and reducing memory bandwidth waste.

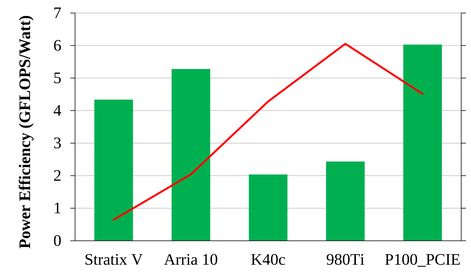
[1] Hasitha Muthumala Waidyasooriya, Yasuhiro Takei, Shunsuke Tatsumi, and Masanori Hariyama. 2017. OpenCL-Based FPGA-Platform for Stencil Computation and Its Optimization Methodology. IEEE Trans. Parallel Distrib. Syst. 28, 5 (May 2017), 1390-1402.
[2] Kohei Nagasu, Kentaro Sano, Fumiya Kono, and Naohito Nakasato. 2017. FPGA-based tsunami simulation: Performance comparison with GPUs, and roofline model for scalability analysis. J. of Parallel and Distrib. Comput. 106 (Aug. 2017), 153-169.
[3] Naoya Maruyama, Takayuki Aoki. 2014. Optimizing stencil computations for NVIDIA Kepler GPUs. In Proceedings of the 1st Int'l Workshop on High-Performance Stencil Comp. (HiStencils '14). Vienna, Austria, 89-95.

Methodology

- 7-point 3D single-precision floating-point stencil
  - FLOP to byte ratio of $^{13}/_{8}$
- Two Intel FPGAs and three NVIDIA GPUs
- Best throughput is selected per device
  - FPGAs: input dimensions align with block size
  - GPUs: input dimensions are a multiple of 512
- Highly-optimized GPU code [3]
- At least 1 GB input size and 2 seconds run time
- Host/Device transfer time is ignored
- Board power is estimated for Stratix V and measured for others

Results





- ***The FPGAs can achieve higher computation throughput than their memory bandwidth***
- ***Arria 10 achieves better performance than K40c, despite 8 times lower memory bandwidth***
- ***Both FPGAs have better power efficiency than 980Ti***

Future Outlook

Addition of HBM to the upcoming Stratix 10 FPGAs, coupled with better scaling with temporal blocking compared to GPUs, could allow these devices to achieve better performance compared to their same-generation GPUs for stencil computation.

This work was supported by MEXT, and JST CREST under Grant Number JPMJCR1303, and performed under the auspices of the Real-world Big-Data Computation Open Innovation Laboratory, Japan.