GEMM-Like Convolution for Deep Learning Inference on the Xilinx Versal

Jie Lei, Héctor Martínez, José Flich, Enrique S. Quintana-Ortí
Universitat Politècnica de València, Spain
Universidad de Córdoba, Spain.

Presenter: Jie Lei
Universitat Politècnica de València, Spain

The authors gratefully acknowledge funding from
- European Union's Horizon2020 Research and Innovation program under the Marie Skłodowska Curie Grant Agreement No. 956090 (APROPOS, http://www.apropos-itn.eu/).
- European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955558.
- Junta de Andalucía research project PID2020-113656RBC22 of MCIN/AEI/10.13039/501100011033.

This presentation contain content written by Enrique S. Quintana-Orti

Some images are from: Flaticon.com

Motivations

➢ General Matrix multiplication accelerations

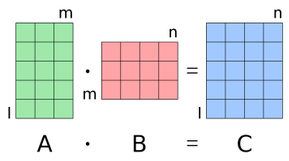
High Performance	High Commodity
Intel Xeon, AMD HPC	NVDIA GPU+ ARM
Low Power	**Accelerators**
Embedded Edge AI	SIMD Processer Xilinx Versal

Previous work:

Matrix Multiplication for Deep Learning Inference on the Xilinx Versal,

Jie Lei, Jose Flich, Enrique S. Quintana-Orti,
31st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing 03/2023

arxiv.org/abs/2302.07594



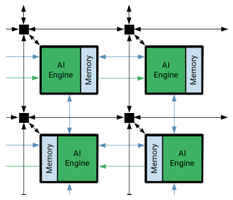
General Matrix Multiplication

AI Engine Local

FPGA RAM

DDR

Multi Level Memories



*adapted from AMD data sheet

Multiple AI Engines

INT16 Based GEMM, 28 MAC/Cycle out of 32

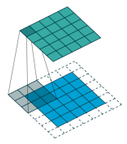
Mitigate Communication Overhead by Matrix reuse

Current work: NEW

GEMM-Like **Convolution** for Deep Learning **Inference** on the **Xilinx Versal**

Jie Lei, Héctor Martínez, José Flich, Enrique S. Quintana-Ortí

https://jiegh.github.io/about/H32023/H3_PAGE.html



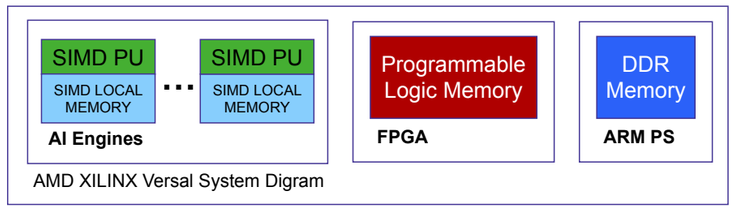
Extend the use case: applicable to direct convolution

16 BIT ⇩ 8 BIT

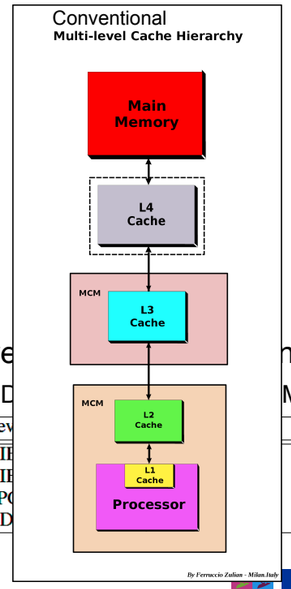
Compute throughput leap

Evaluating throughput with layers of Resnet

Differences of our Versal based design compares

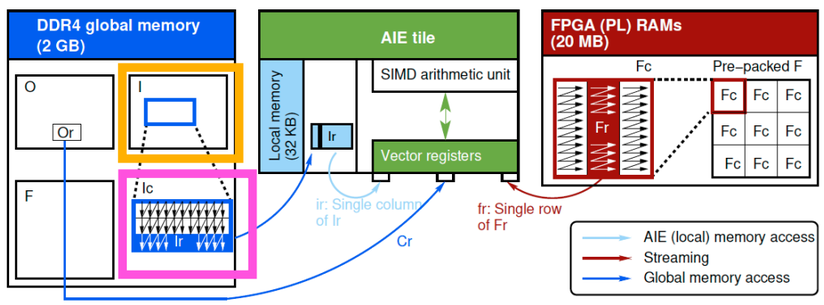


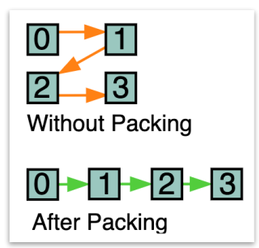
AMD XILINX Versal System Digram



- No conventional multi-level caches system
  - Yet, 3 types with different speed and capacity
- Most of the data fetching/storing are made explicitly.
  - Streaming interface
  - Global memory access IO interface
- Matrix F repacked into FPGA memory

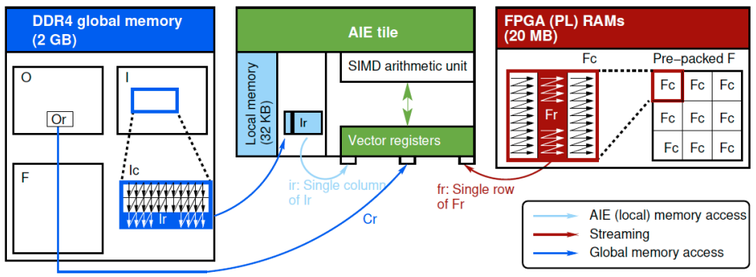
Approach: Matrix tiling and packing





- Partition Matrix into smaller blocks
- Packing into a continuous memory location

Approach: Tiled matrix transferring

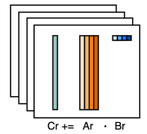


Capacities of different memory units in Versal

Level	Capacity
AIE vector registers	2 KB
AIE tile local memory	32 KB
FPGA RAMs	20 MB
DDR4 (global) memory	2 GB

- For an O = I * F, data movement
  - Output matrix O: From DDR -> Vector register
  - Input image matrix I: From DDR. -> SIMD Local memory. -> Vector register
  - Trained filter matrix F/Fc: From FPGA Memory -> Vector register

Approach: Micro-kernel design

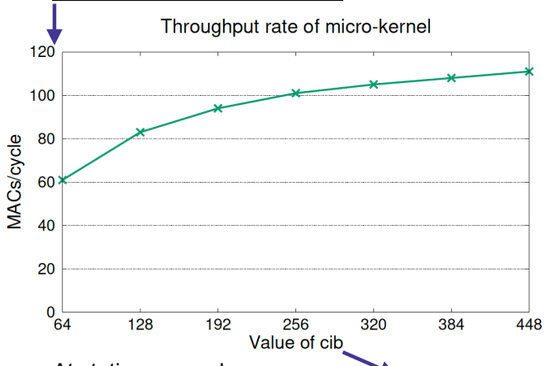



```
1  for (jc=0; jc<n; jc+=nc) // Loop L1
2   for (pc=0; pc<k; pc+=kc){    // L2
3    // Pack B
4    Bc:=B(pc:pc+kc-1, jc:jc+nc-1);
5    for (ic=0; ic<m; ic+=mc){   // L3
6     // Pack A
7     Ac:=A(ic:ic+mc-1, pc:pc+kc-1);
8     for (jr=0; jr<nc; jr+=nr)  // L4
9       for(ir=0; ir<mc; ir+=mr) // L5
10       // Micro-kernel
11       C(ic+ir:ic+ir+mr-1,
12         jc+jr:jc+jr+nr-1)
13         += Ac(ir:ir+mr-1, 0:kc-1)
14         *  Bc(0:kc-1, jr:jr+nr-1);
15  }}
```

- In platform code
  - Utilize intrinsic mac16(…)
    - Performing 128 MAC (multiply and accumulate) operations per clock cycle.
  - Loop L5 unrolls as a factor of 16
    - Split compute into smaller sections, better compute and communication overlaps
    - Utilise more vector and accumulator registers, avoid register spilling
- Matrix tiling and packing: Lines 1 - Line 9
- Micro-kernel: Lines 10 - 14
  - Where matrix small-scale multiplication happened
  - Computation critical region

Identify bottleneck, micro-kernel at stationary mode

Theoretical peak: 128 MACs/cycle

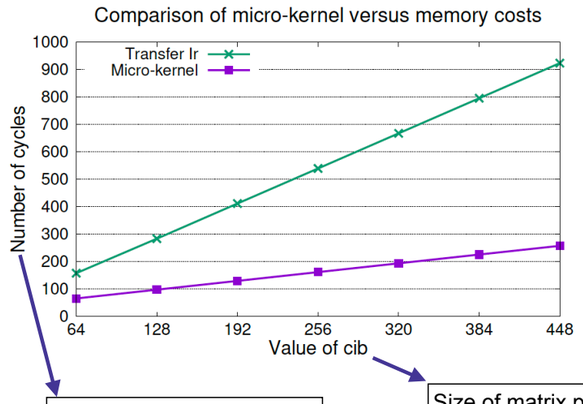


At stationary mode

Size of matrix problem within the micro-kernel

- In this loop:
  - Ir transfer is not considered
  - Fr transfer happen within the loop
- Throughput profiling at stationary mode
  - Only considering the computations at the micro-kernel level
  - The theoretical peak performance is 128 MACs/cycle
- A smaller workload may result in poor loop pipelining

Identify bottleneck, micro-kernel considering some data transfer



Measuring the cycles costs

Size of matrix problem within the micro-kernel

- Transfer of Ir:
  From DDR. -> SIMD Local memory.
  -> Vector register

Outside MAC Loop:
- Transferring of Ir from DDR into the local memory of the AI Engine

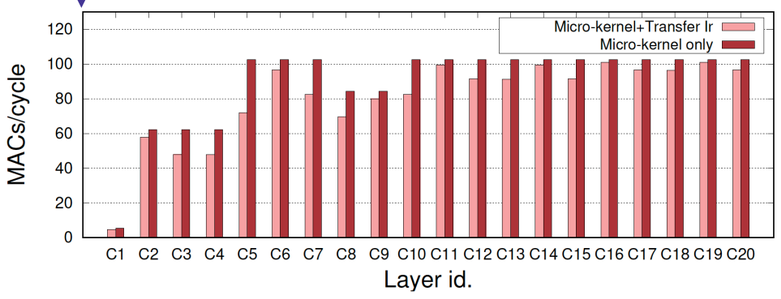
Facts:
- Slow
- Difficult to avoid

To amortize this cost::
- Reuse Ir more, key approach
- > Change the partition parameter of Ir

➢ The theoretical peak performance is 128 MACs/cycle

Evaluating computation throughput with layers in ResNet-50

Theoretical peak: 128 MACs/cycle

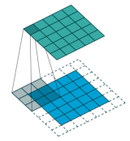


> With larger layers: deliver optimal performance

> Low number of input channel (Ci) cause poor overall throughput

Recap

Alternative GEMM approach for AMD Versal



Extend the use case: applicable to direct convolution

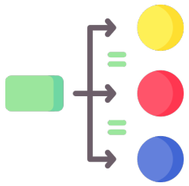
16 BIT
⇩
8 BIT

Compute throughput leap

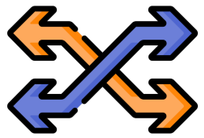
Tackle bottleneck: Throughput analyse

70% of peak performance for some layers of the ResNet

Future work



Utilise Multiple AI Engines



Mixed precisions support

Exploring heterogeneous architecture

http://apropos-itn.eu

This project has received funding from the European Union's Horizon 2020 (H2020) Marie Sklodowska-Curie Innovative Training Networks H2020-MSCA-ITN-2020 call, under the Grant Agreement no 956090.

The authors gratefully acknowledge funding from
- European Union's Horizon2020 Research and Innovation program under the Marie Skłodowska Curie Grant Agreement No. 956090 (APROPOS, http://www.apropos-itn.eu/).
- European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955558.
- Junta de Andalucía research project PID2020-113656RBC22 of MCIN/AEI/10.13039/501100011033.