University of Thessaly
Department of Electrical and Computer Engineering

Diploma Thesis

Implementation of the Tracking Algorithm of a Visual SLAM System on FPGAs

By Viktoria Biliouri

Supervisor : Prof. Nikolaos Bellas
2nd Committee member: Prof. Spyros Lalis
3rd Committee member: Associate Prof. Dimitrios Katsaros

AGENDA

- General Overview
- SLAM - Visual SLAM Systems
- KinectFusion - SLAMbench
- Implementation Tools & Devices
- Project Prerequisites
- Tracking SW & HW Implementation
- Optimizations & Diagrams
- Conclusion

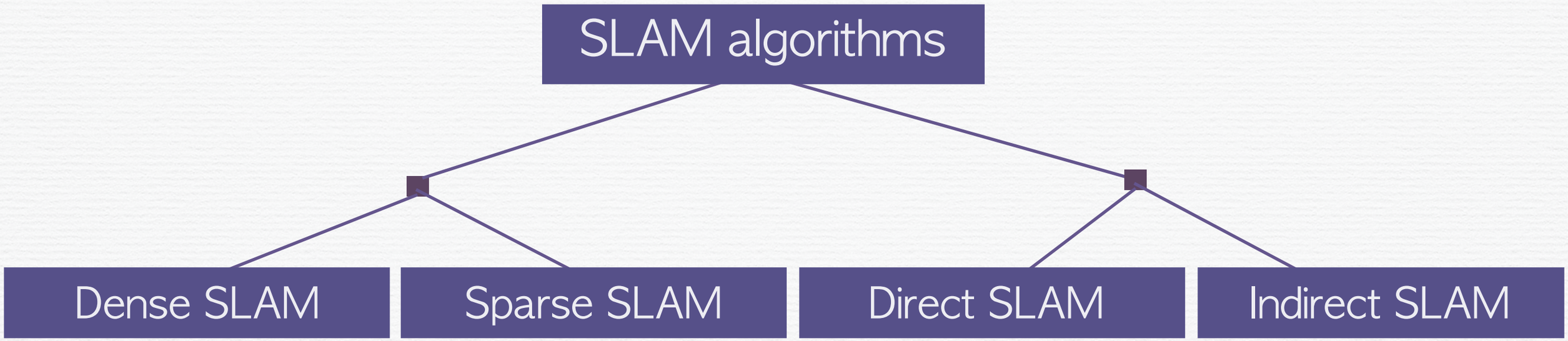
General Overview

KinectFusion is a SLAM algorithm, originally developed by Microsoft, which processes the data provided by input depth frames, and reconstructs a 3D view of an indoor scene by precisely placing the objects into it. This project is concentrated in a specific part of the KinectFusion algorithm, the tracking process, in particular the hardware execution of the track kernel on FPGA, and its optimization.

This project was conducted in terms of the curriculum of the Department of Electrical and Computer Engineering, University of Thessaly, Greece.

SLAM
(Simultaneous Localization and Mapping)

- Indoor scene 3D reconstruction by tracking and placing the objects in a 3D map



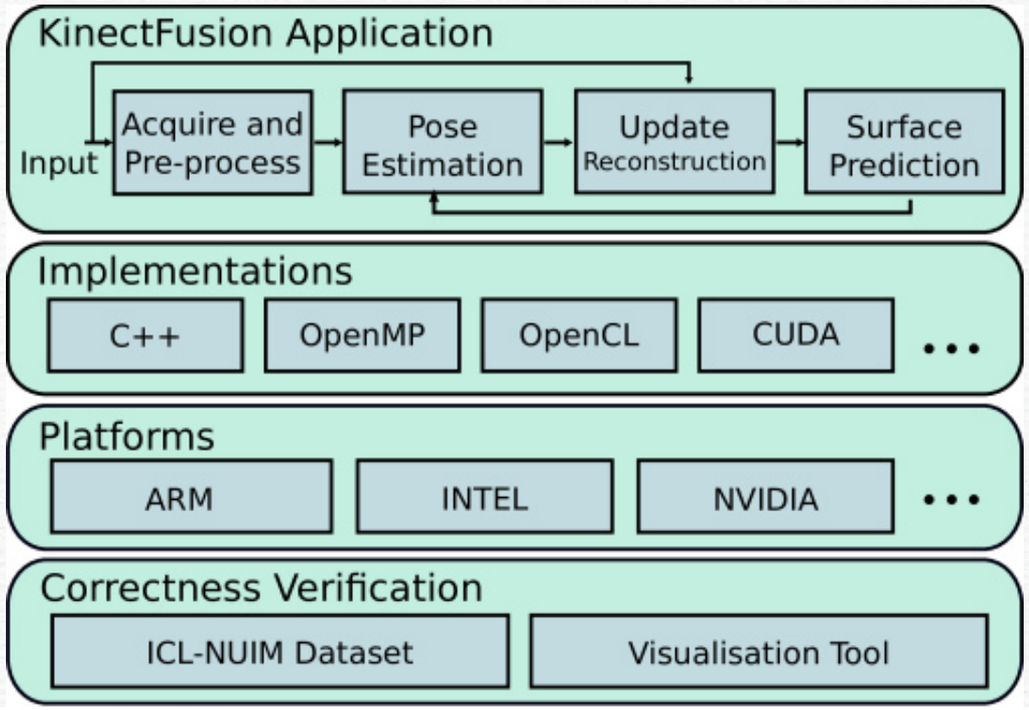
- Applications: self-driving cars, unmanned aerial vehicles, autonomous underwater cars, domestic robots, etc.

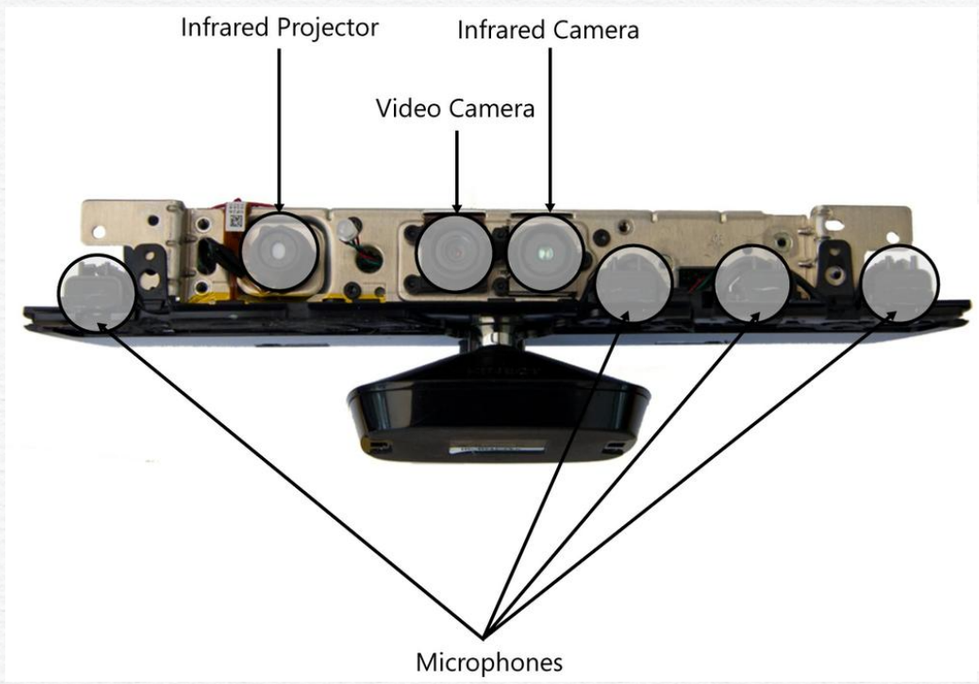
Visual SLAM

- Determine the exact position of a sensor regarding its surroundings, while mapping the space around it
- Use of the approximate projected position of a camera pose
- Single 3D vision camera to operate in real-time
- Modules:
  - Initialization
  - Tracking
  - Mapping
  - Relocalization
  - Global Map Optimization

KinectFusion

- Dense SLAM algorithm
- Designed by Microsoft
- Input Depth frames extracted by Kinect Motion Controller's camera

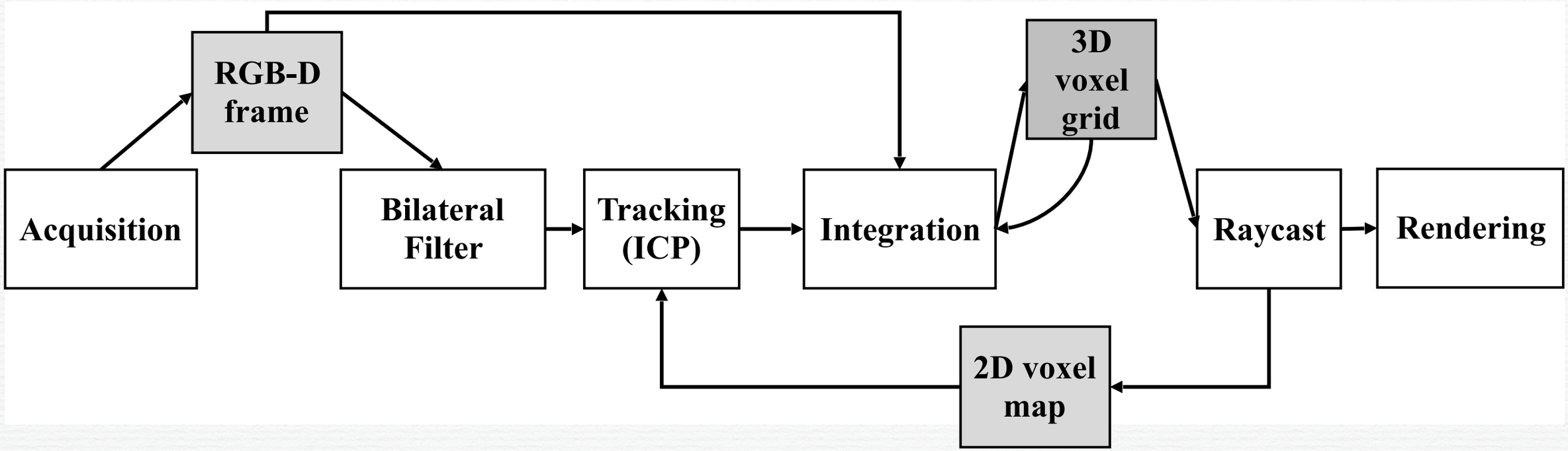




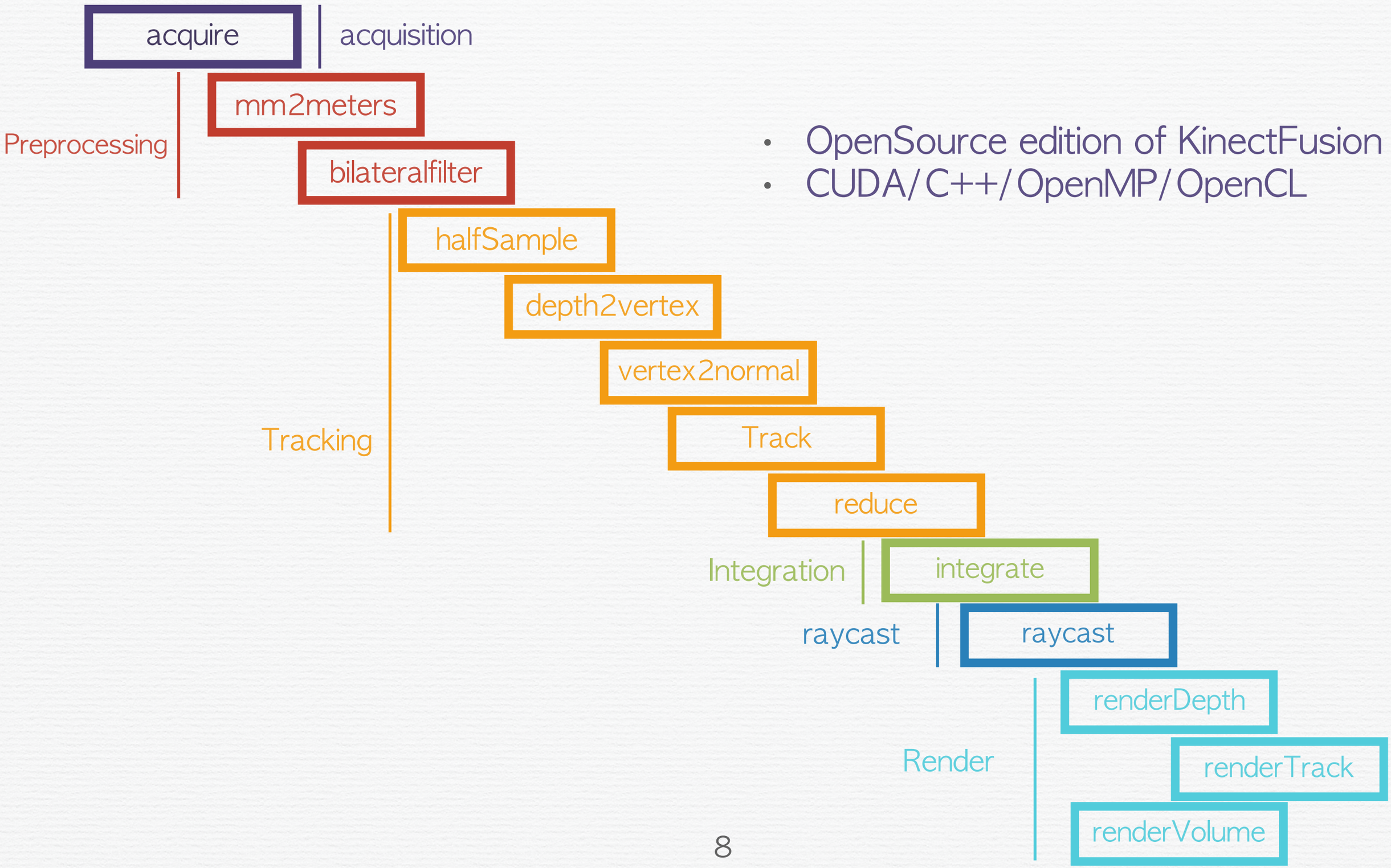
Depth Image

Microsoft Kinect

KinectFusion Stages



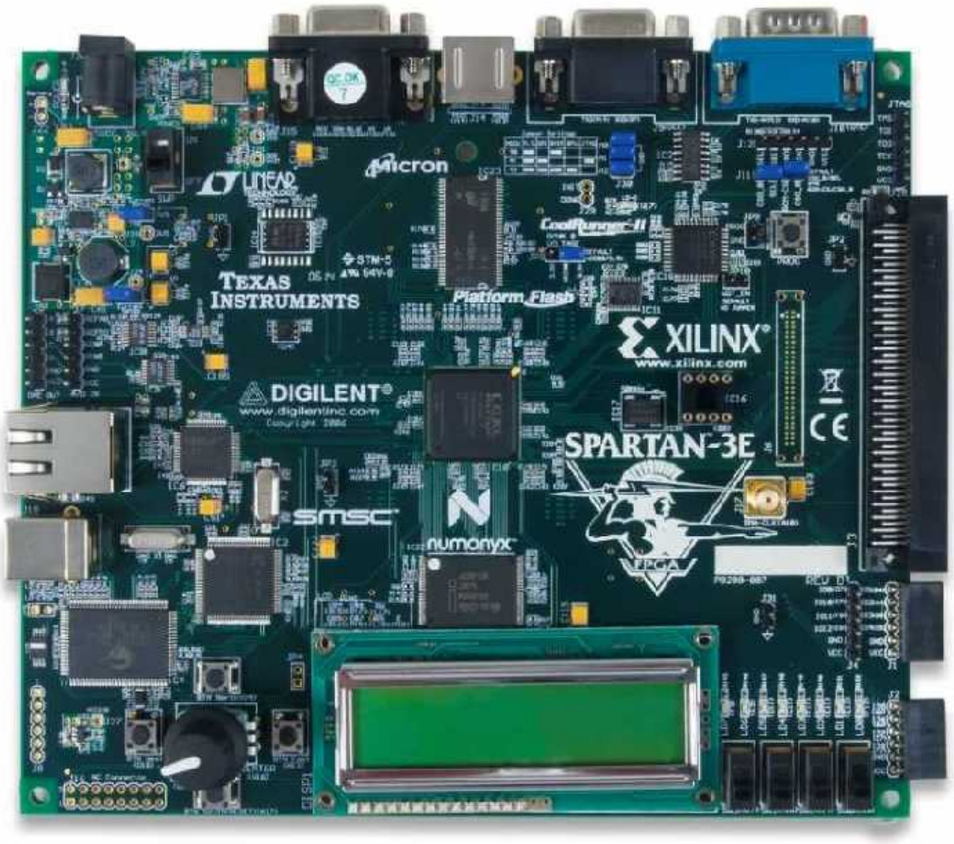
SLAMbench Kernels



- OpenSource edition of KinectFusion
- CUDA/C++/OpenMP/OpenCL

Implementation Devices
FPGA (Field-Programmable Gate Array)

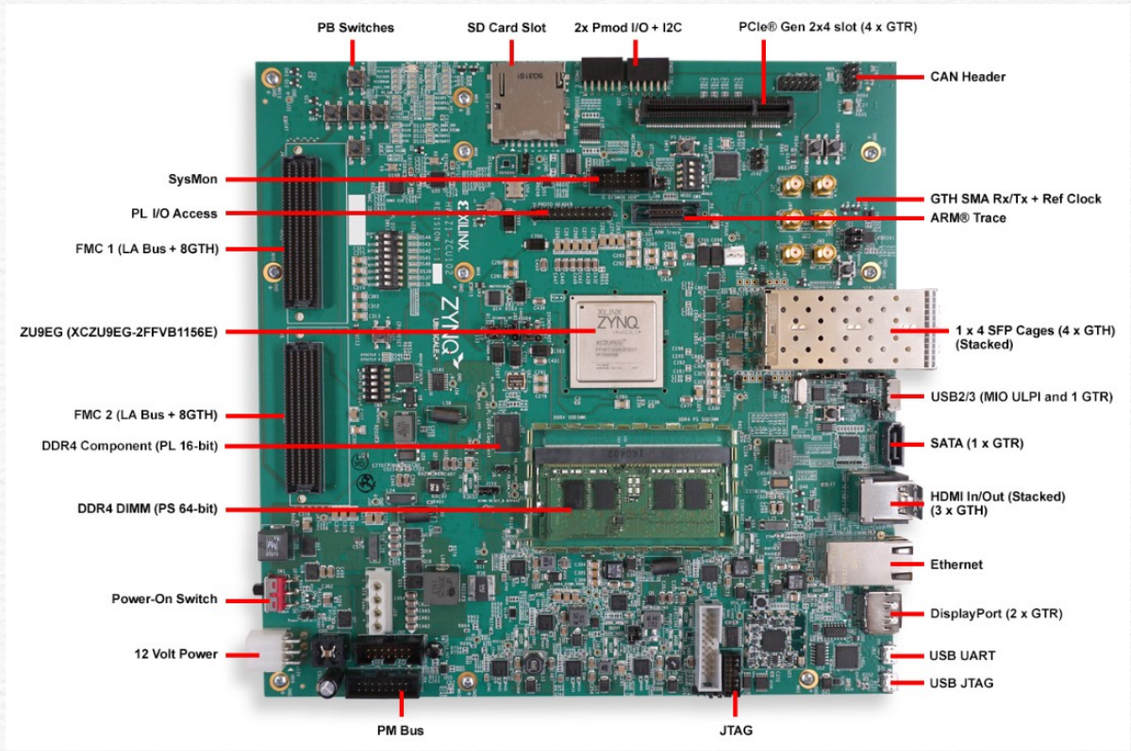
- User Programmable Integrated circuit
- Contents
  - Programmable Logic Blocks, LUTs
  - Interconnects
  - Flip Flops (FFs)
  - Block RAMs (BRAMs)
- Part of System-On-Chips (SoC)



Implementation Devices

Xilinx Zynq UltraScale+ MpSoC ZCU102

- General Purpose flexible reprogrammable SoC for rapid prototyping
- Contents
- quad-core ARM® Cortex-A53
- dual-core Cortex-R5 real-time processors
- FPGA
- Mali-400 MP2 graphics processing unit
- Block RAMs (BRAMs)
- Programmed using Xilinx Development Platforms



Implementation Tools

Vitis Unified Software Platform

- Software Design Tool
- Integration and Execution of Hardware Kernels, developed by Vivado Design Suite
- OpenCL API for the simultaneous execution and connection among Software and Hardware Kernels

Implementation Tools

Vivado HLS (High Level Synthesis)

- Hardware Development Tool
- Design and Execution of hardware IPs in C/C++ instead of HDL
- Programming Language automatically translated to RTL
- Synthesis Report: Profiling, Performance Control, Latency and Space Optimizations

OpenCL (Open Computing Language)

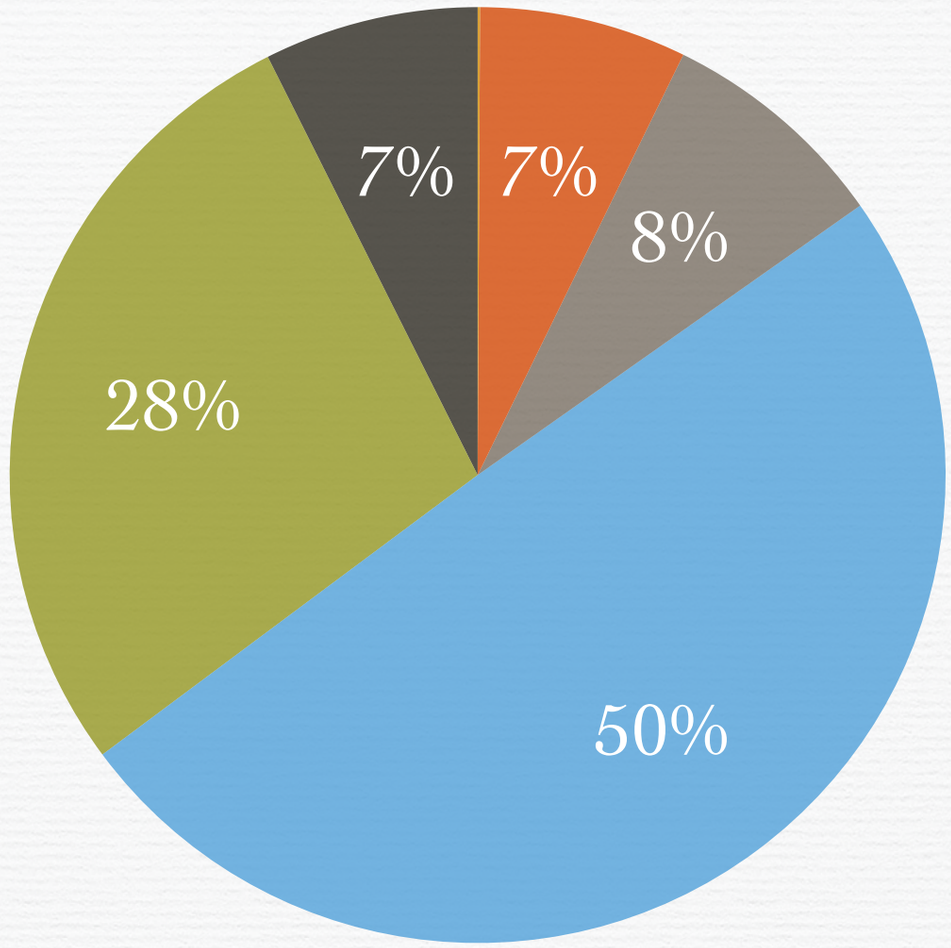
- Framework for Parallel Programming for devices as CPUs, DSPs, FPGAs
- C-language libraries, C compiler
- Multiple compute units, kernels

Project Prerequisites

- ICL - NUIM dataset
  - Input RGB-D images
  - Living Room Scene, Trajectory lr_kt2
  - Objects such as sofa, table, lamp etc.
  - 882 Depth frames Dataset
  - 640x480 resolution

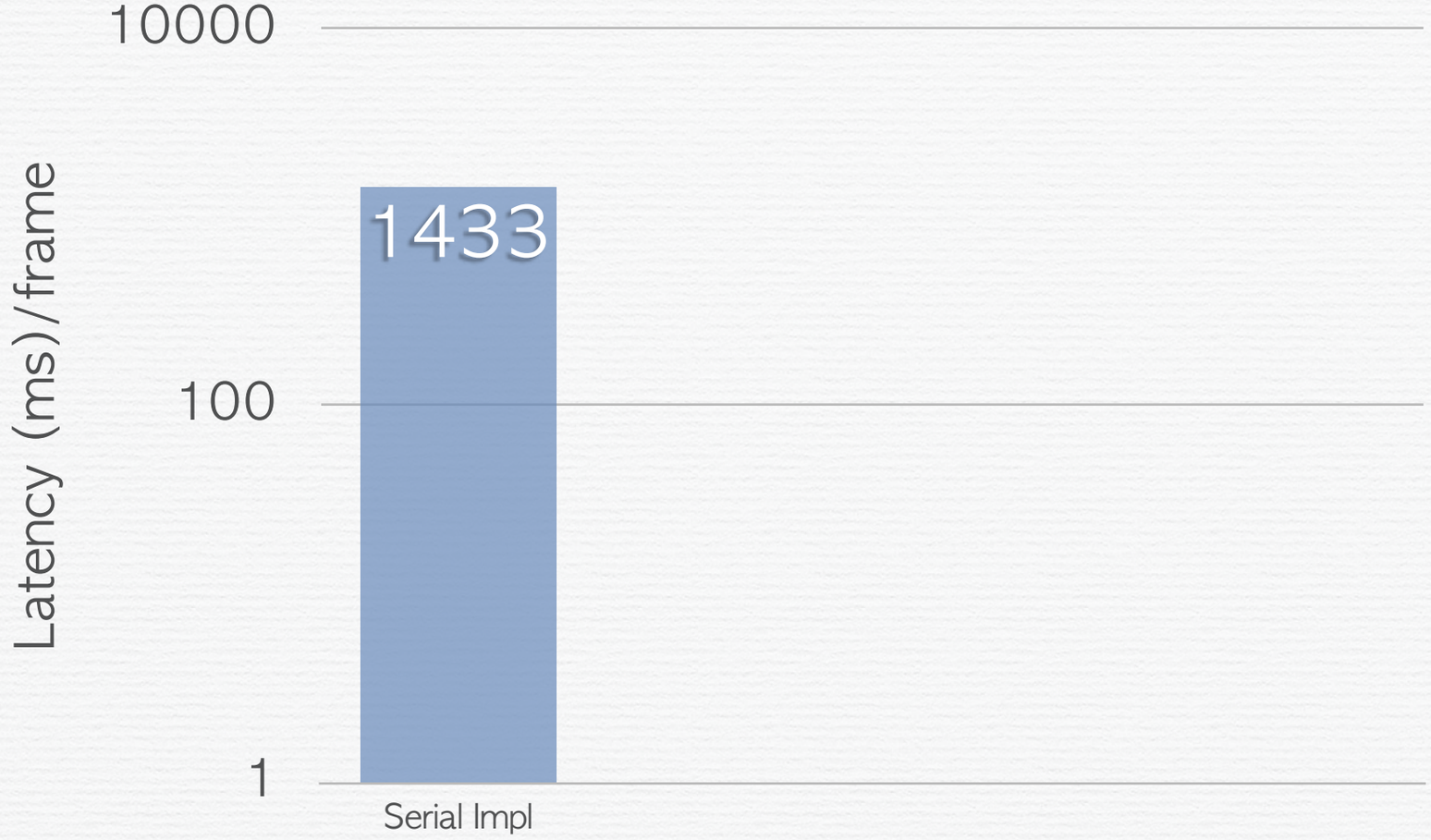
Software Execution

- Zynq Ultrascale+ MpSoc ZCU102 ARM processor
  - Quad core ARM Cortex-A53 processor, 8-stage pipelined
- Total latency = 535.7 ms/frame



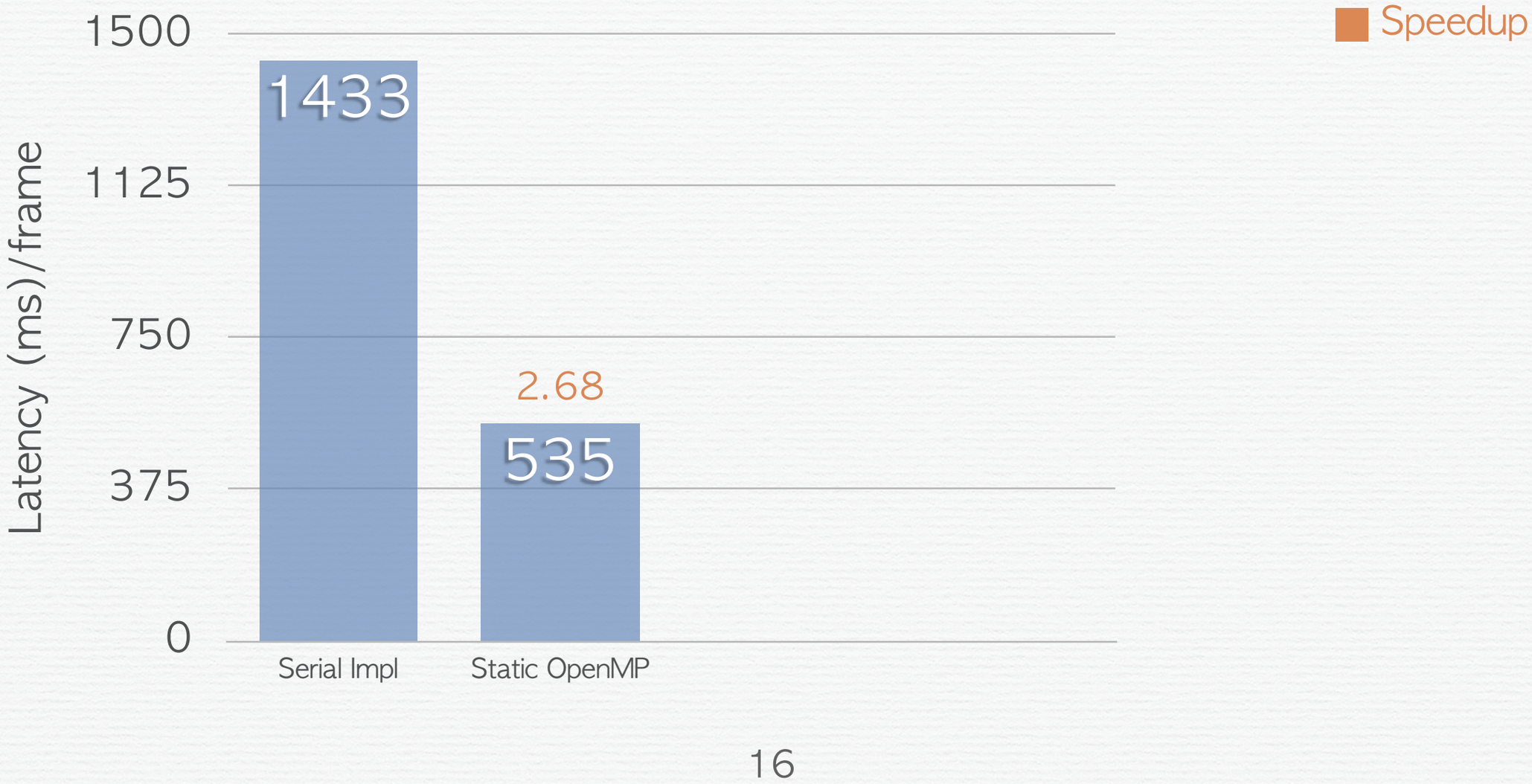
Software Execution

- OpenMP , parallel execution of the algorithm
- OpenMP static and OpenMP dynamic
  - Significant latency improvement



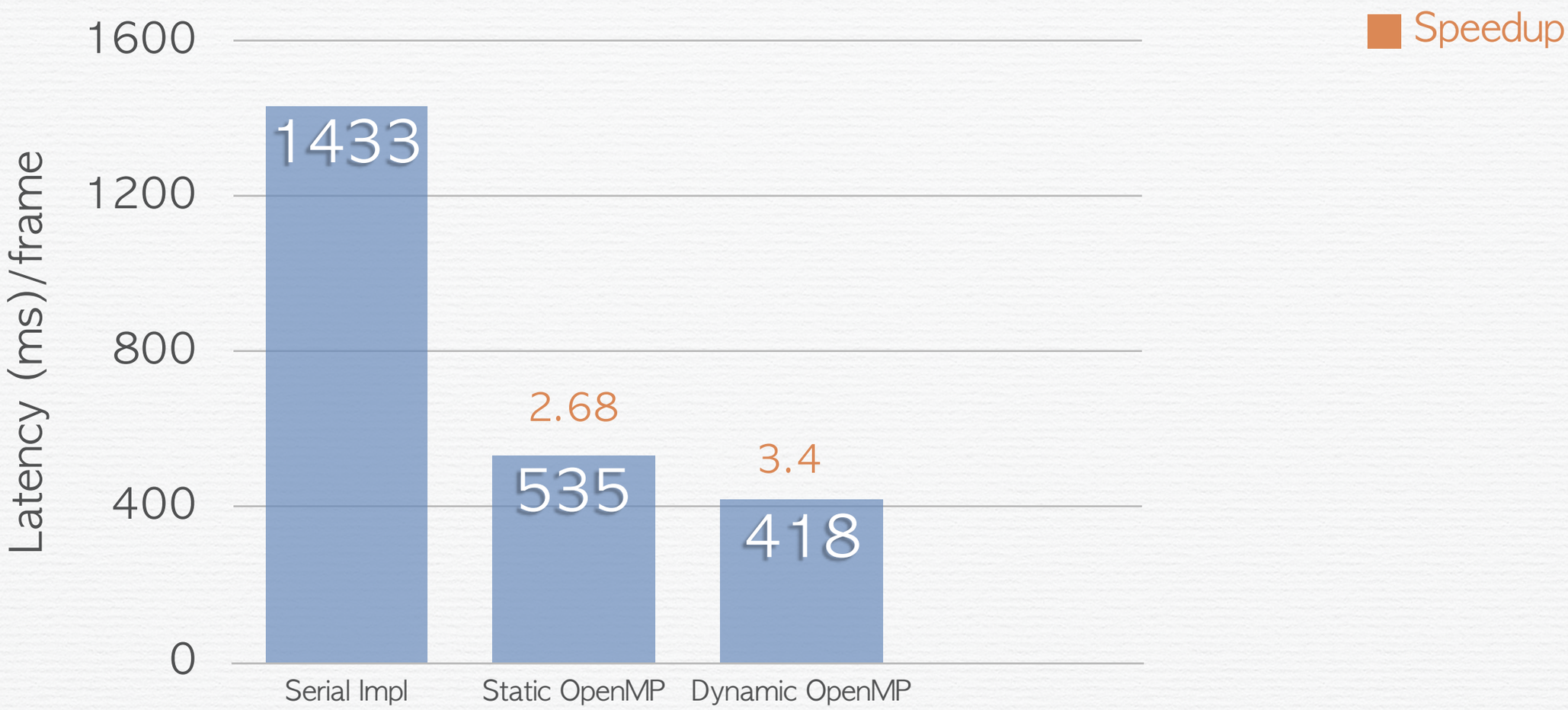
Software Execution

- OpenMP , parallel execution of the algorithm
- OpenMP static and OpenMP dynamic
  - Significant latency improvement



Software Execution

- OpenMP , parallel execution of the algorithm
- OpenMP static and OpenMP dynamic
  - Significant latency improvement

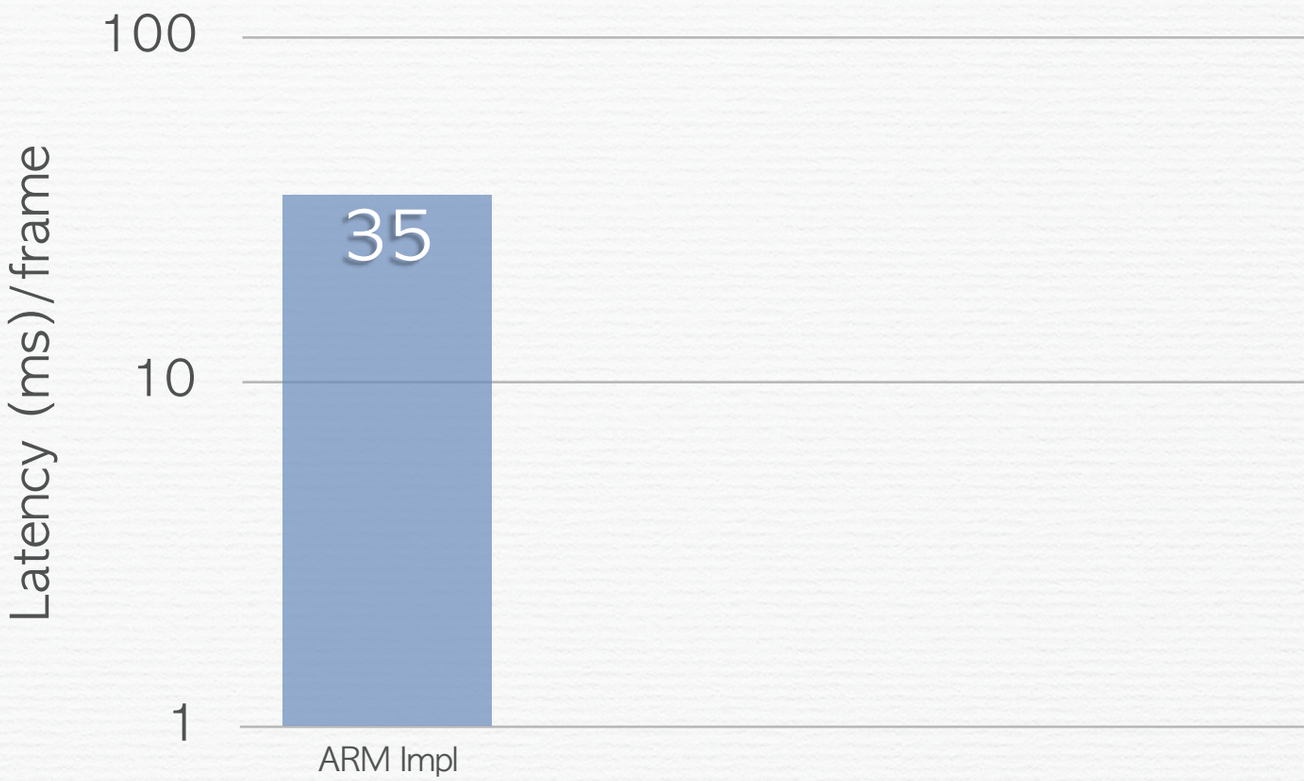


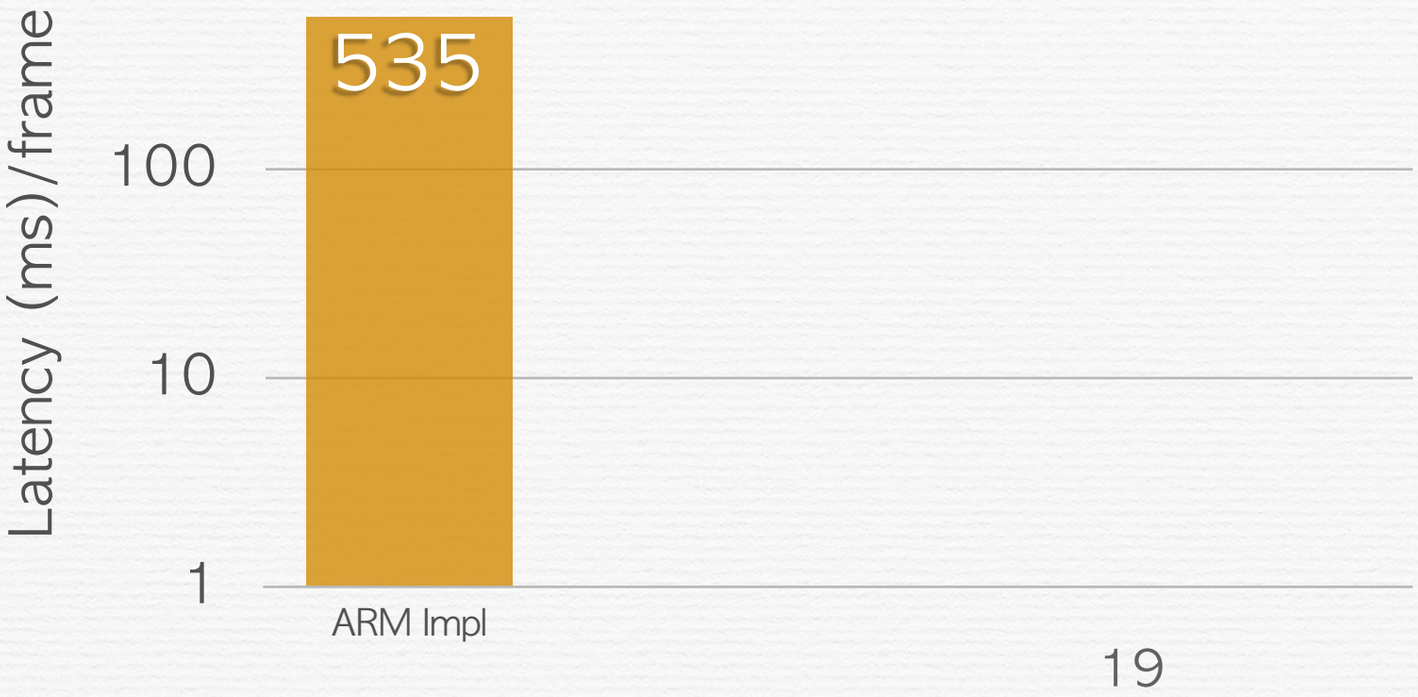
Track Kernel

- Tracking Stage
  - halfSampleRobustImage Kernel
  - Depth2vertex kernel
  - Vertex2normal kernel
  - Track kernel
  - Reduce kernel
- Process of Track Kernel
  1. Input vertices and normals used in previous kernels
  2. Projection of the possible position of each pixel
  3. Exact Position Calculation of each pixel
  4. Output data contain the position of all pixels
- Output data - 1 integer (error) and 7 floats (tracked data vector elements) for each pixel
- Depth frames resolution - 320 x 240

Track Kernel Hardware Accelerator

- Vivado HLS implementation of the Kernel
- Input and output data stored to memory buffers, accessed to every iteration
- Baseline Implementation:
  - Clock Frequency 100 MHz - 2797 ms/ frame
  - Clock Frequency 300 MHz - 1677 m/ frame



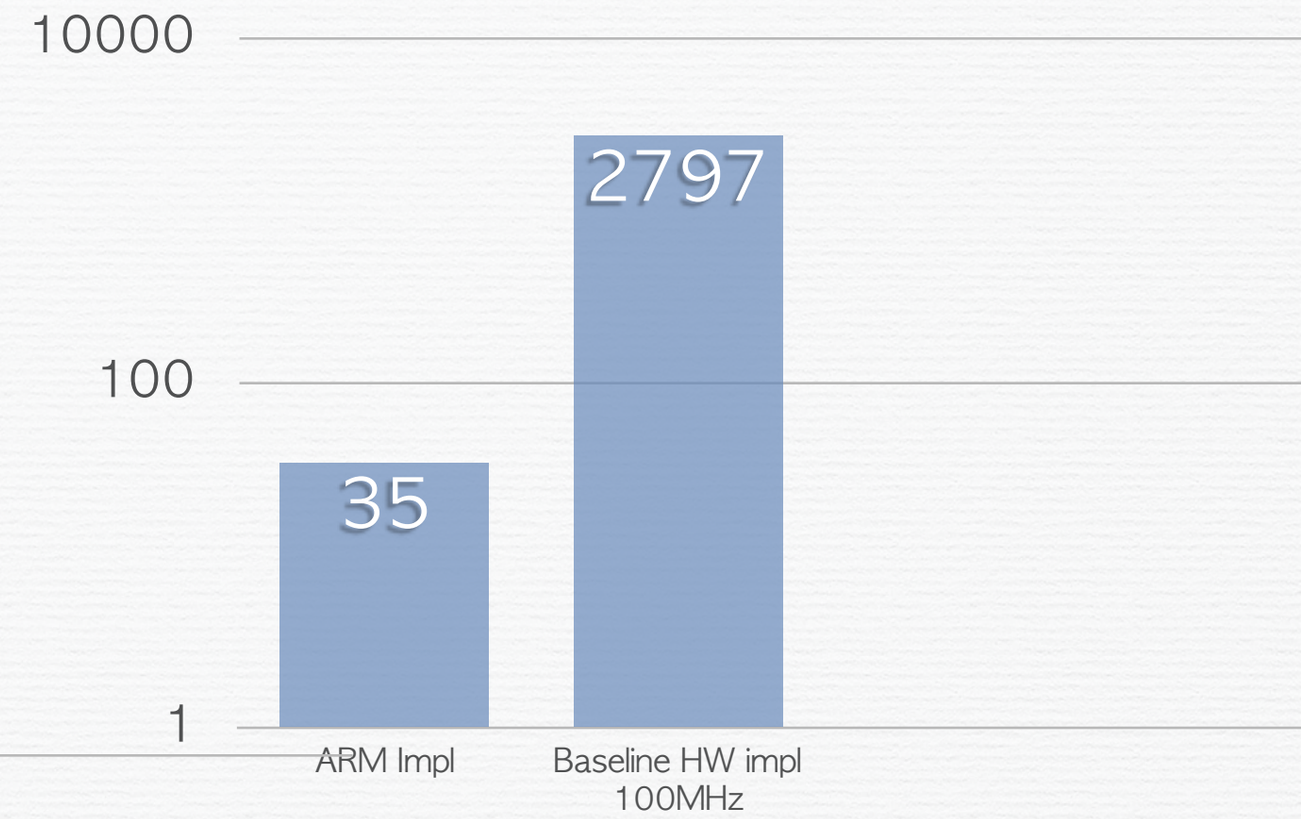


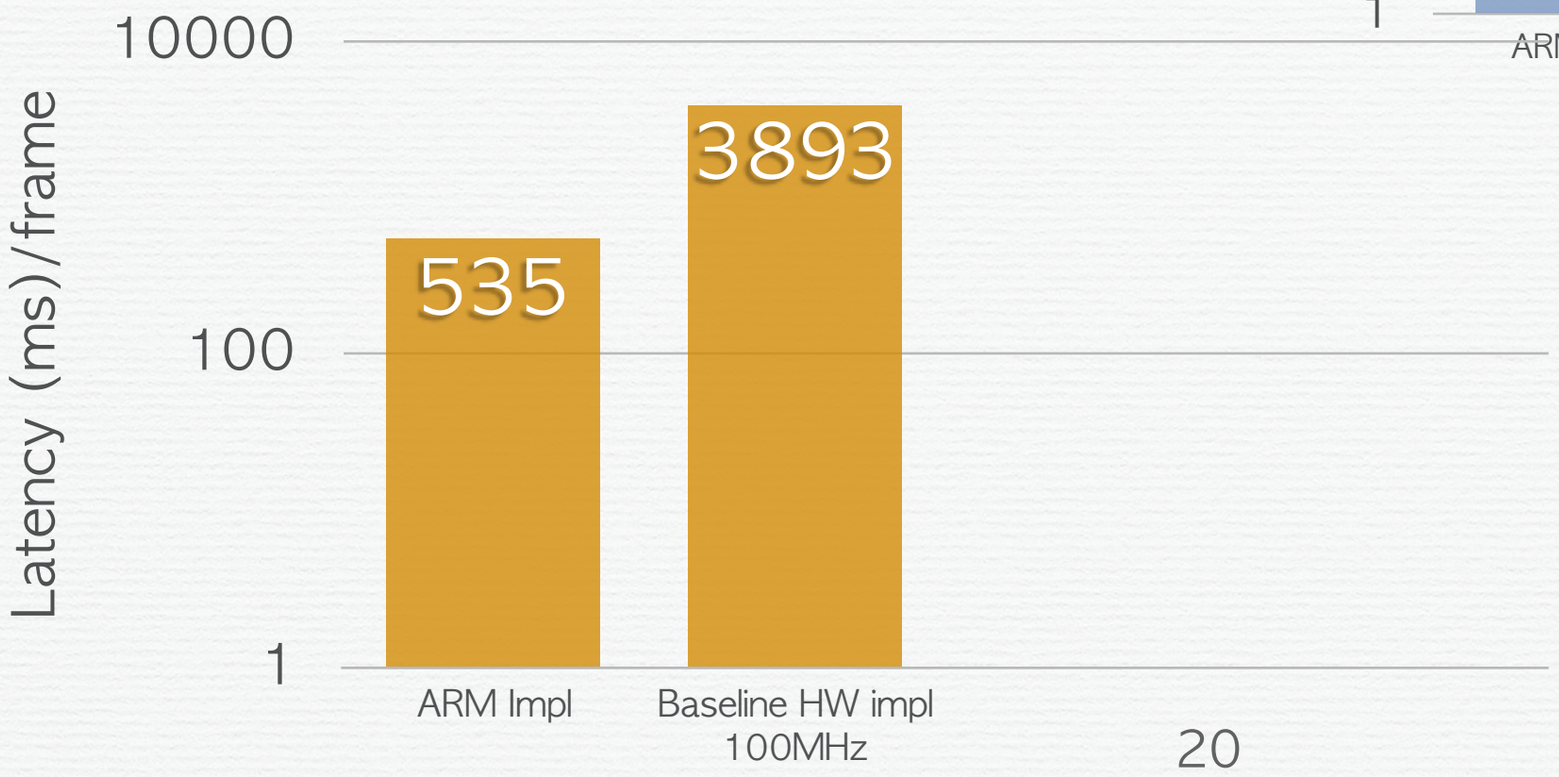
Track Kernel Latency

KinecFusion Algorithm Latency

Track Kernel Hardware Accelerator

- Vivado HLS implementation of the Kernel
- Input and output data stored to memory buffers, accessed to every iteration
- Baseline Implementation:
  - Clock Frequency 100 MHz - 2797 ms/ frame
  - Clock Frequency 300 MHz - 1677 m/ frame



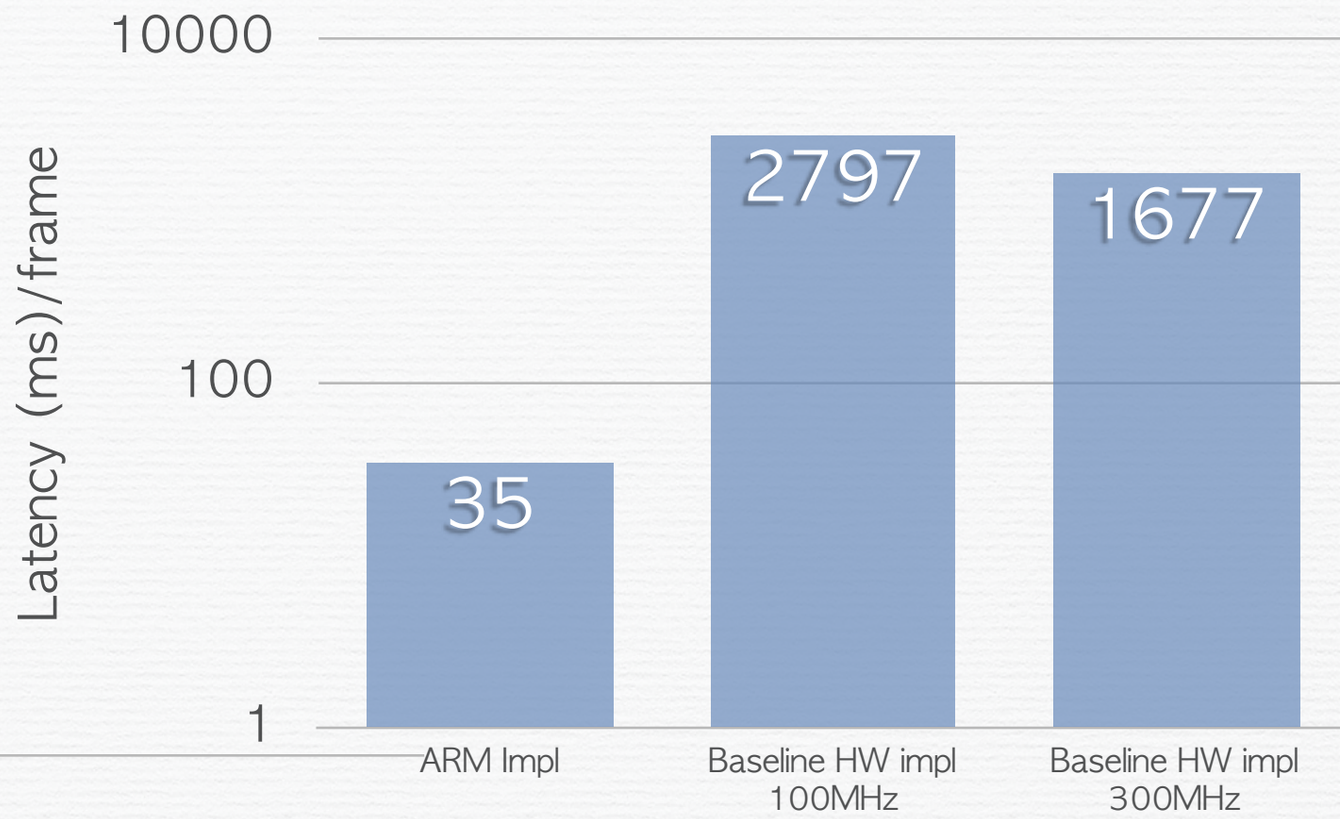


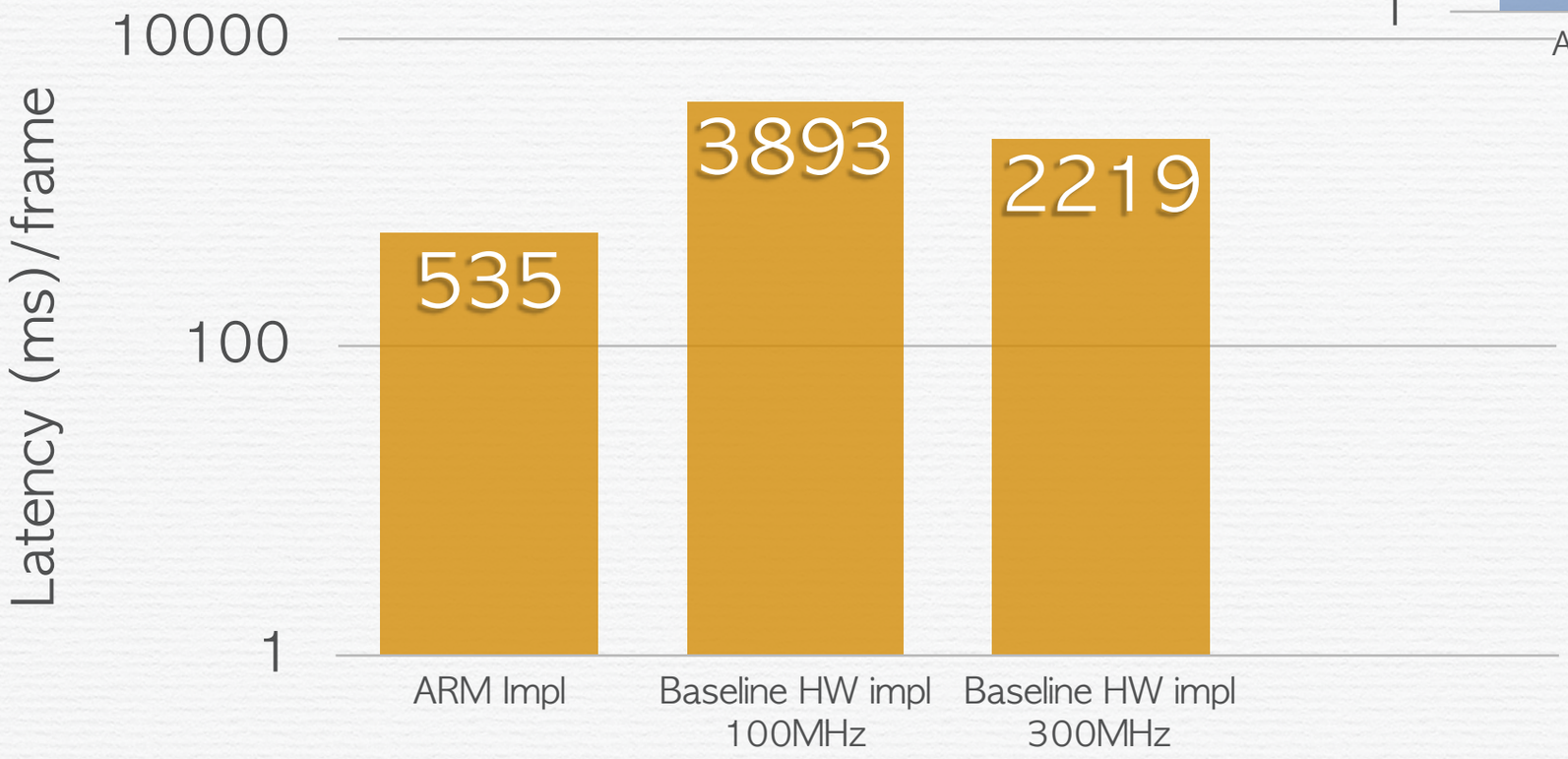
Track Kernel Latency

KinecFusion Algorithm Latency

Track Kernel Hardware Accelerator

- Vivado HLS implementation of the Kernel
- Input and output data stored to memory buffers, accessed to every iteration
- Baseline Implementation:
  - Clock Frequency 100 MHz - 2797 ms/ frame
  - Clock Frequency 300 MHz - 1677 ms/ frame





Track Kernel Latency

KinectFusion Algorithm Latency

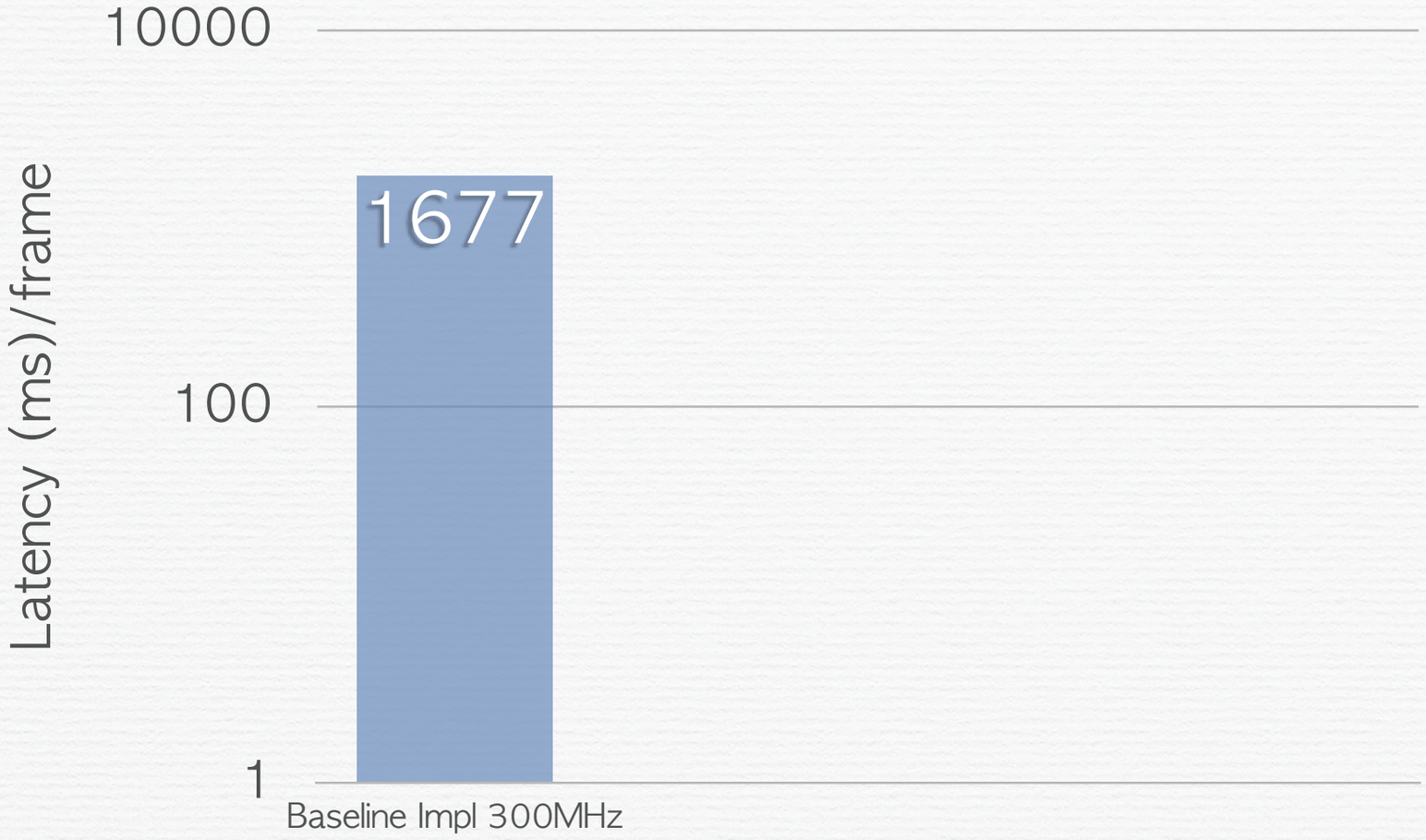
Optimizations
Precise Optimizations

- Pipelining II=1
  (+Computations Optimization)
- Unrolling factor = 2
- Multiple Output Lines Stored to BRAMs
- Output buffer's data structure alteration to 2xfloat4/pixel

ATE ~= 18m

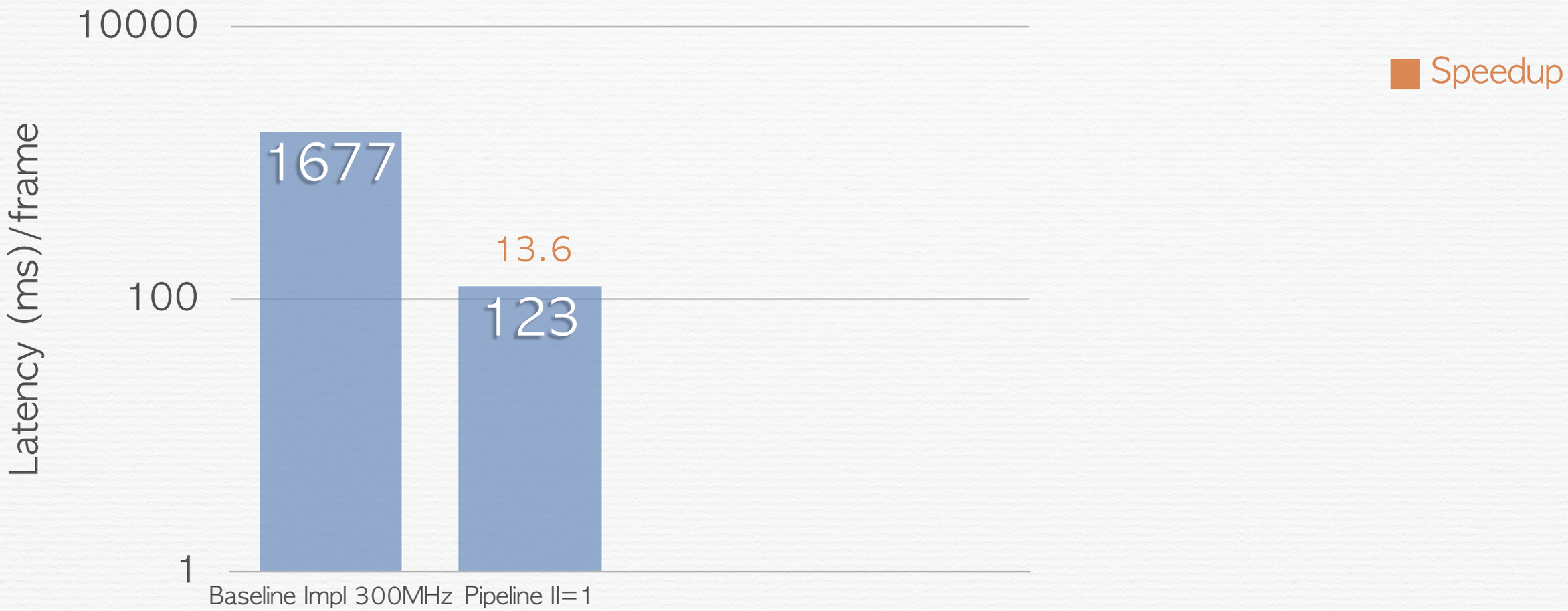
Optimizations Diagram

Precise Optimizations



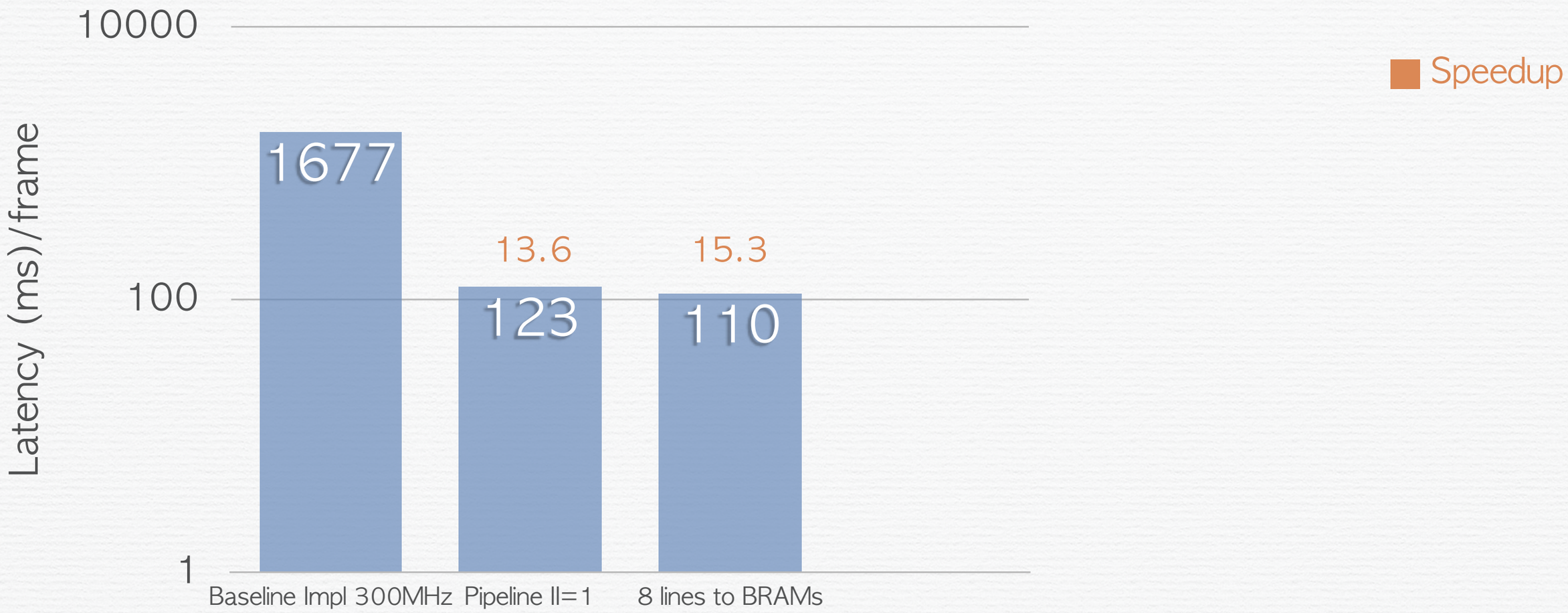
Optimizations Diagram

Precise Optimizations



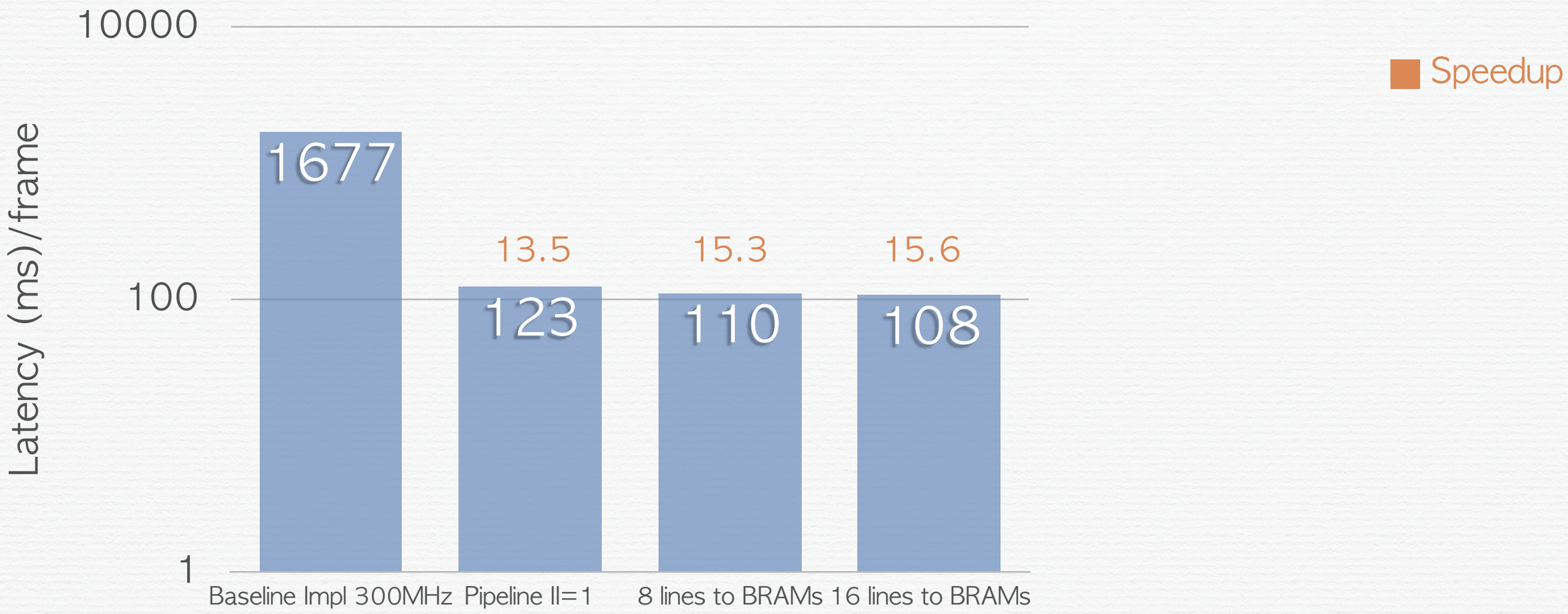
Optimizations Diagram

Precise Optimizations



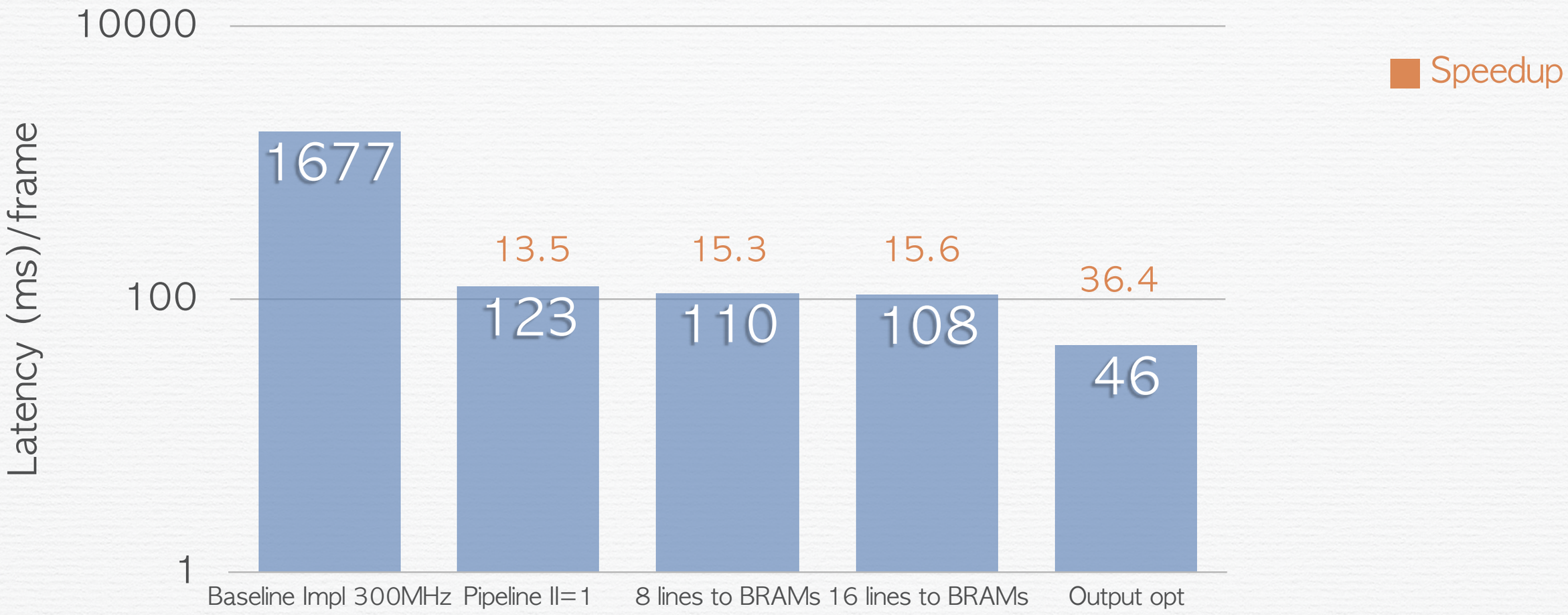
Optimizations Diagram

Precise Optimizations



Optimizations Diagram

Precise Optimizations



Optimizations
Approximate Optimizations

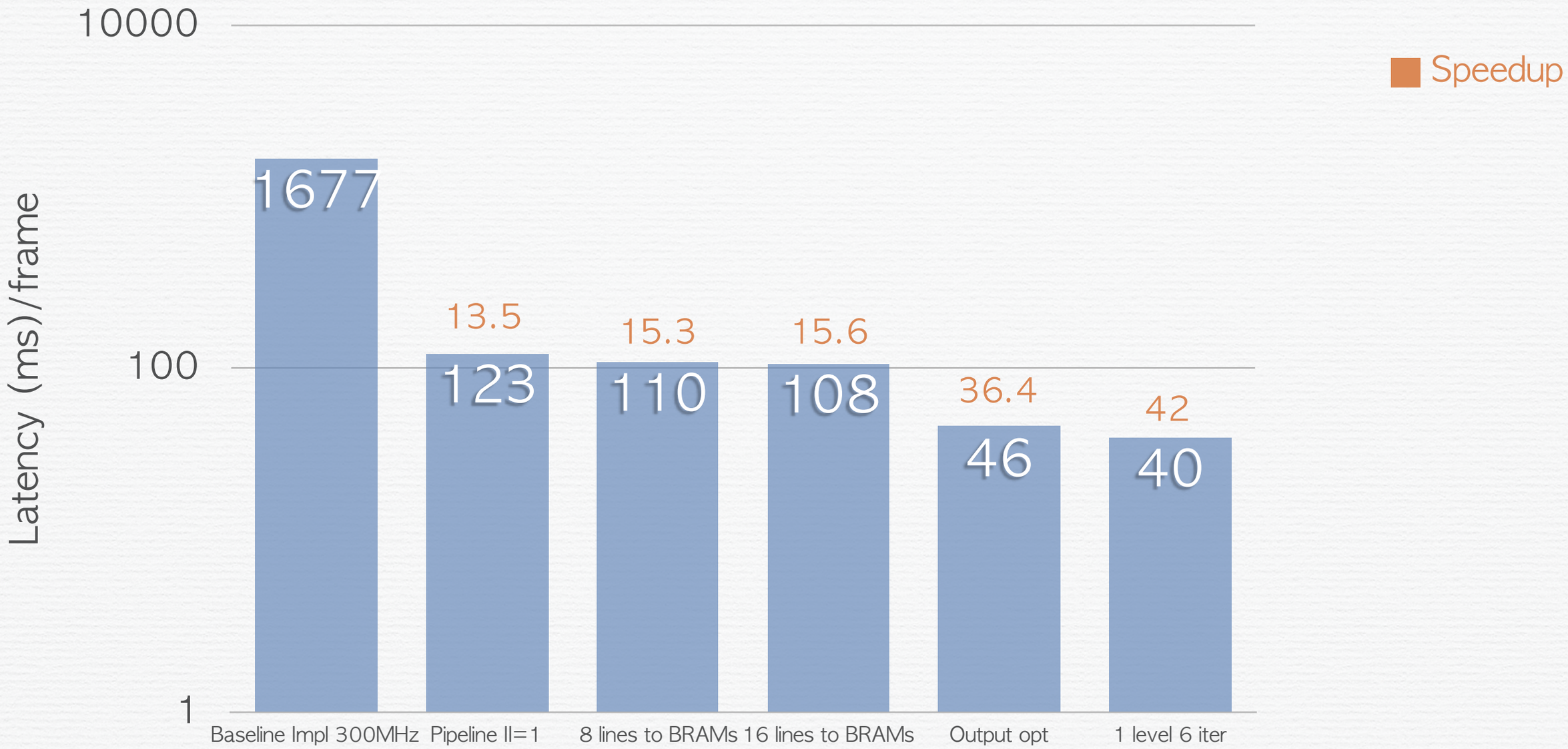
- 1 pyramid level
- Track kernel's iterations reduction (for the 1 level left)
  - 6 iterations
  - 4 iterations
  - 2 iterations

Initially 3 levels:
level 1 - 10 iterations
level 2 - 5 iterations
level 3 - 4 iterations

ATE ~= 18 - 19 m

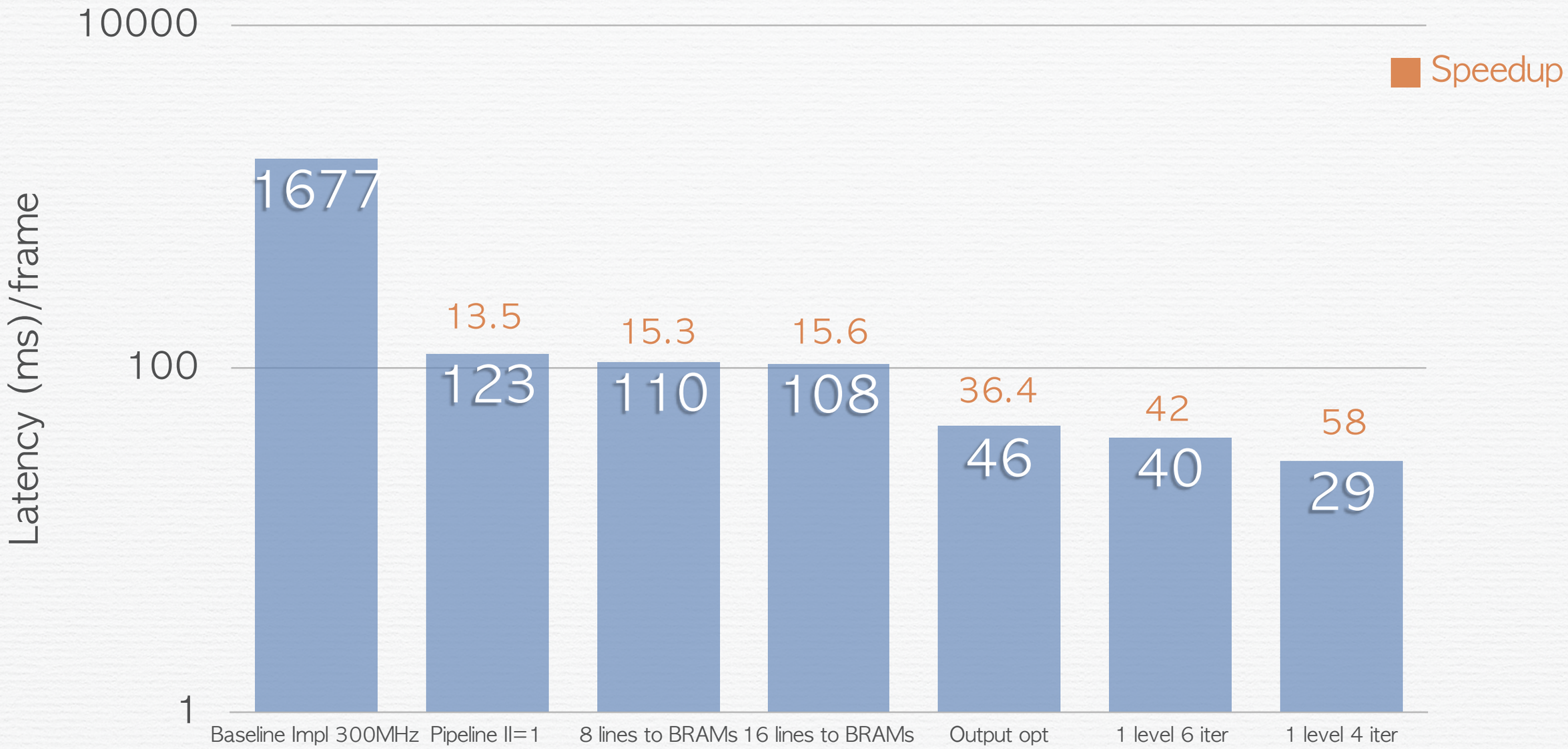
Optimizations Diagram

Approximate Optimizations



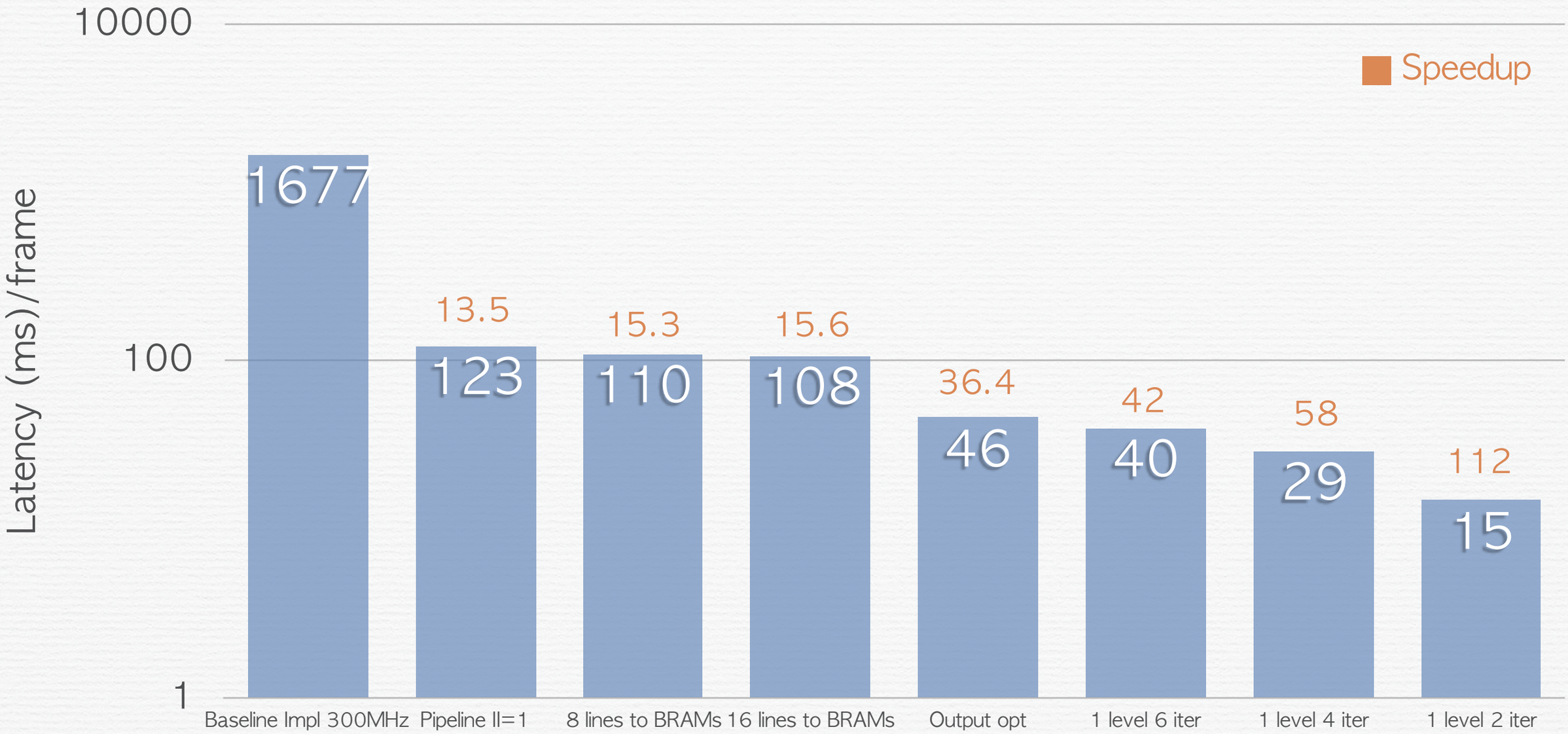
Optimizations Diagram

Approximate Optimizations



Optimizations Diagram

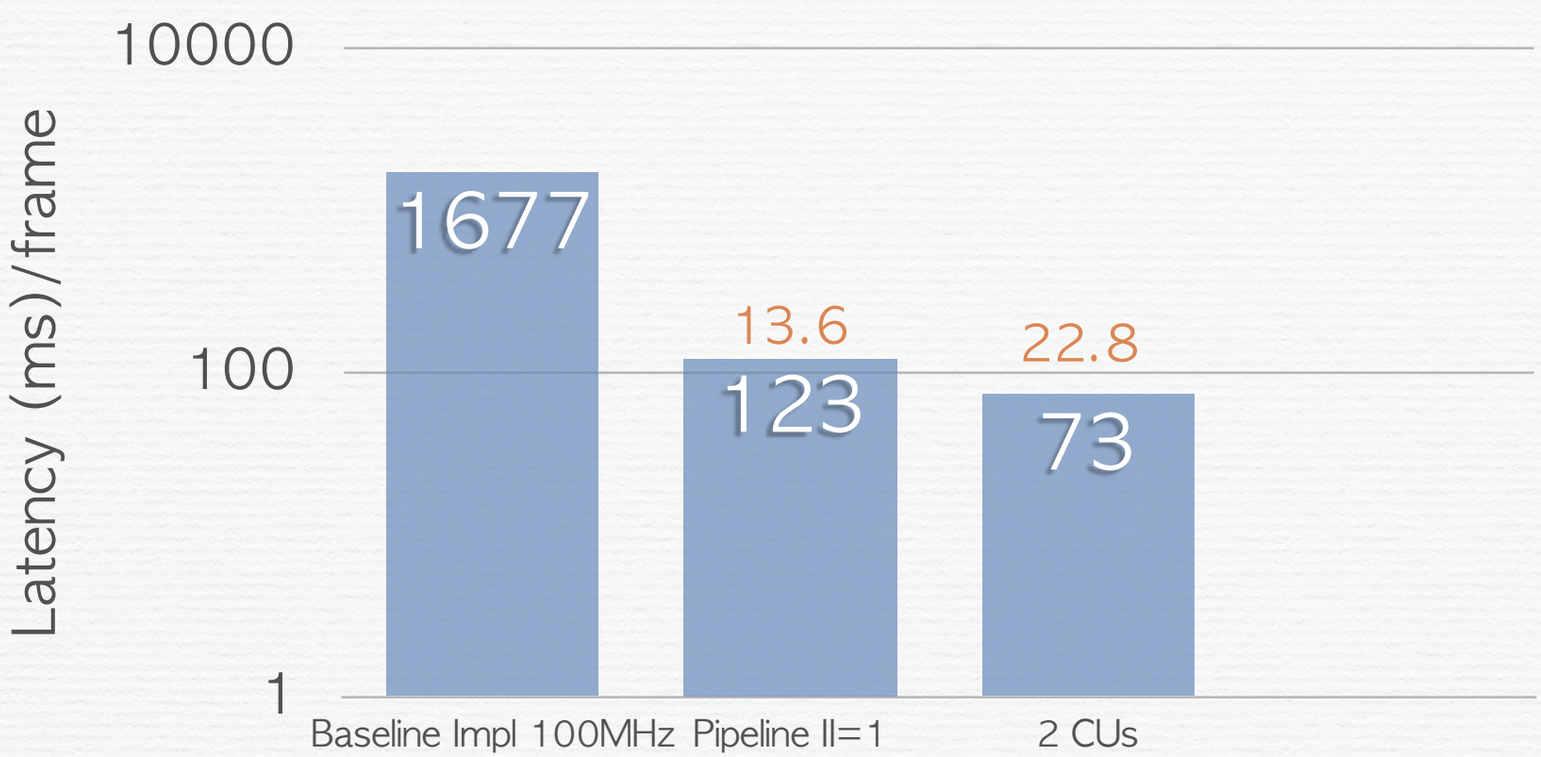
Approximate Optimizations



Optimizations

Multiple Compute Units

2 CUs after Pipeline Optimization - Process half frame each CU

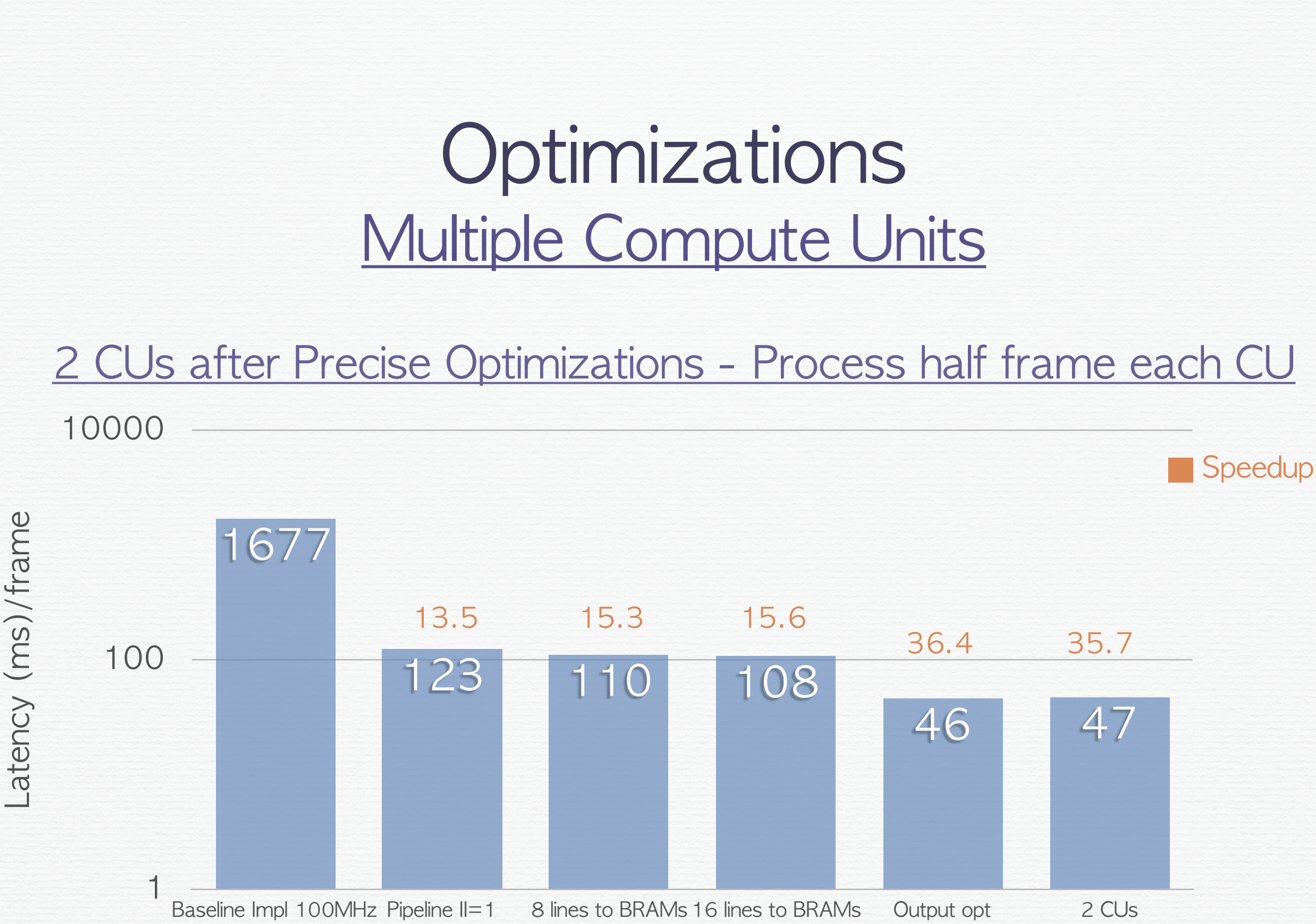


ATE ~=19 m

Optimizations

Multiple Compute Units

2 CUs after Precise Optimizations - Process half frame each CU

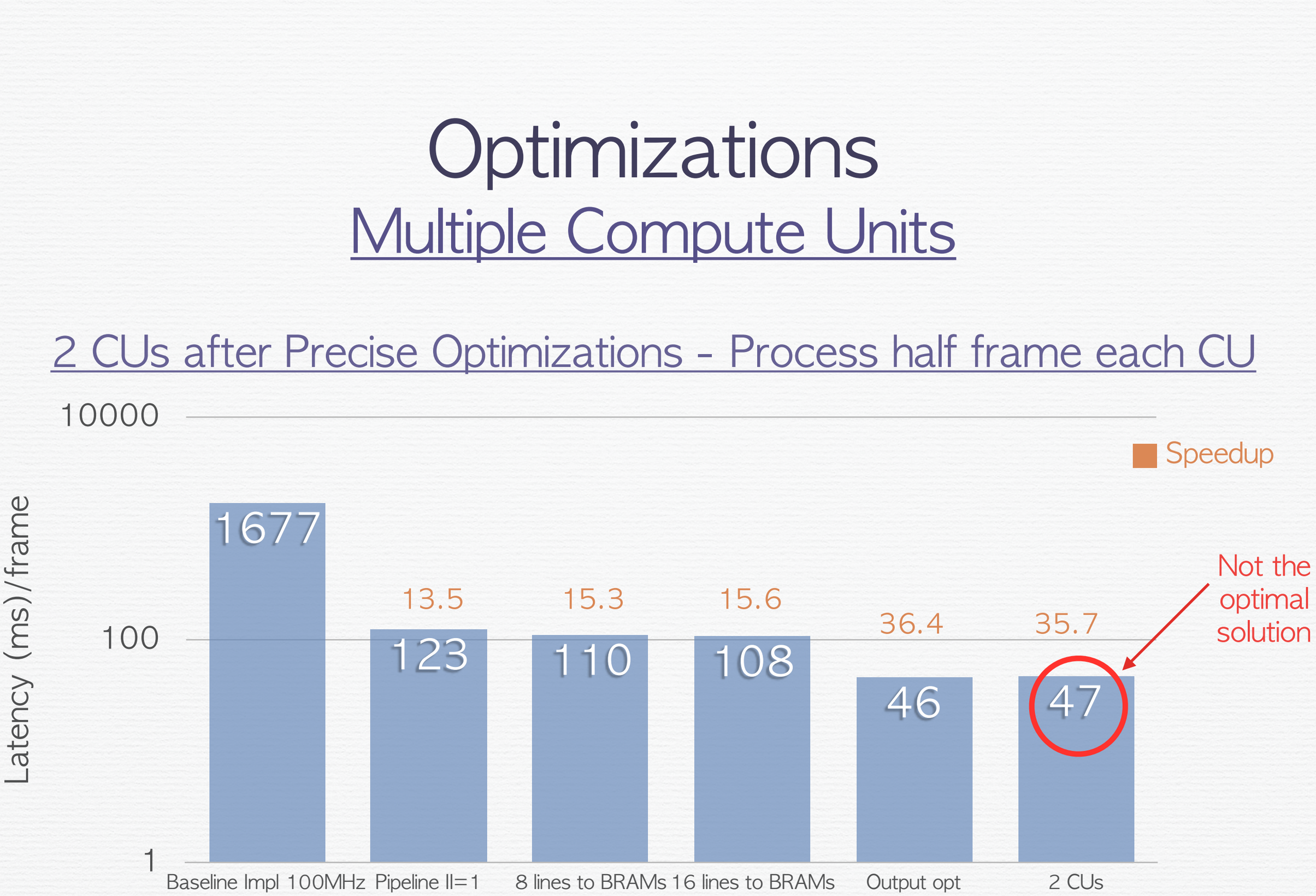


ATE ~= 18m

Optimizations

Multiple Compute Units

2 CUs after Precise Optimizations - Process half frame each CU



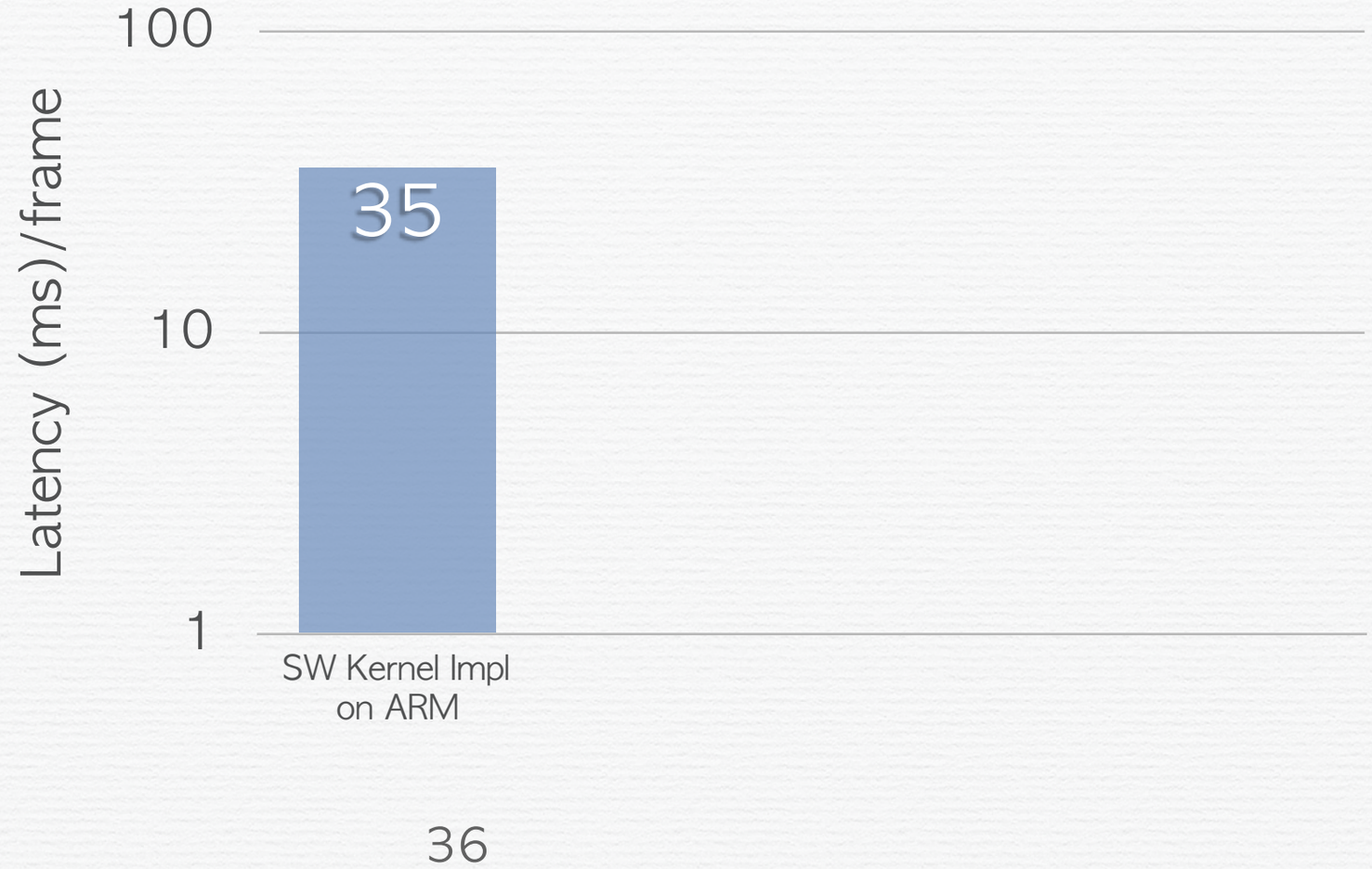
ATE ~= 18m

Area utilization

	BRAM(%)	DSP(%)	LUT(%)	FF(%)
Baseline Implementation 300 MHz	1.14	3.02	6.81	5.03
Optimal Precise Implementation	6.04	8.25	11.94	8.44
Optimal Solution	6.04	8.25	11.94	8.44
2 CUs Implementation	8.74	16.75	24.38	17.3

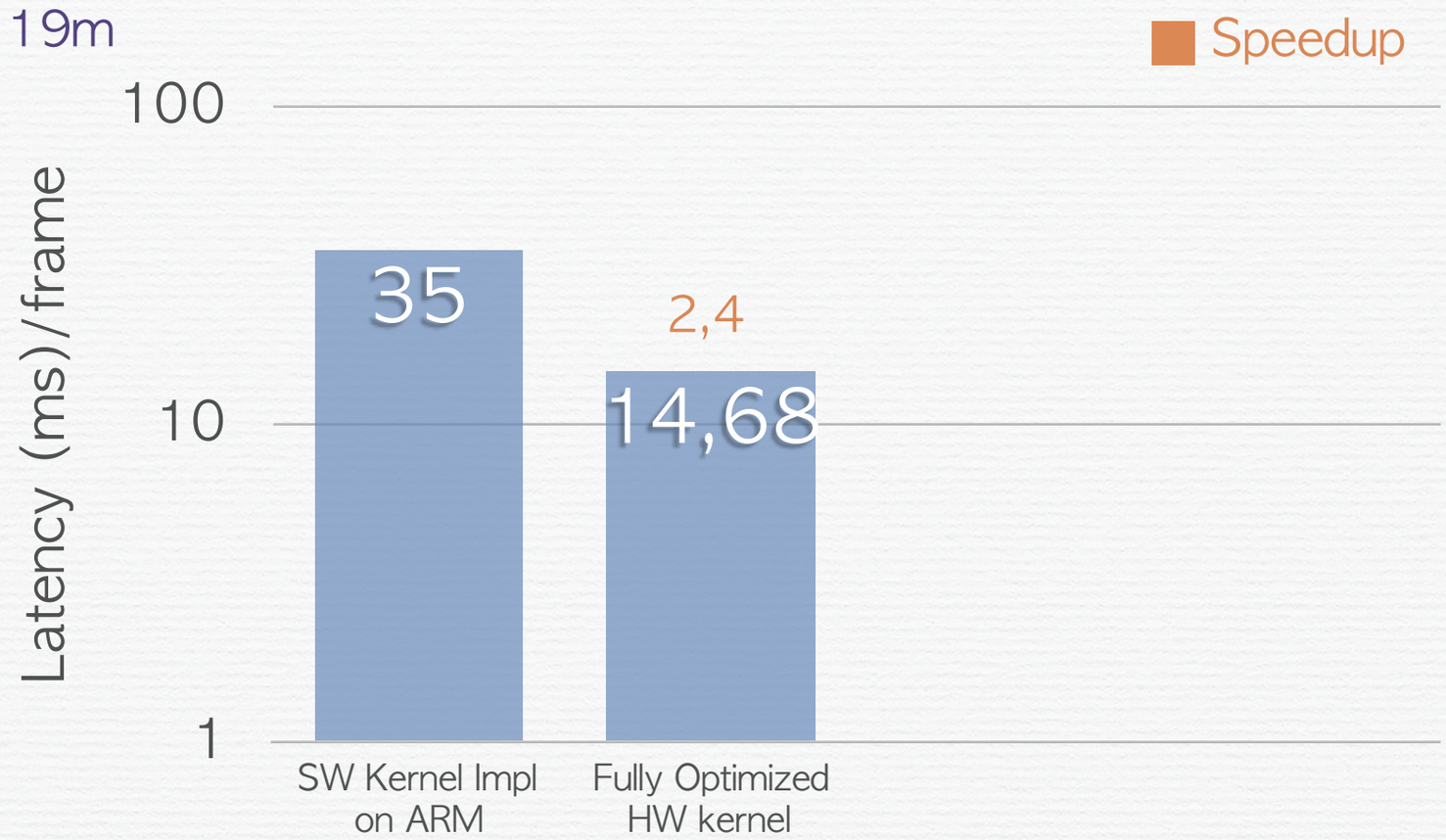
Conclusion

- Optimal Implementation: Latency = 14.68
- Almost half kernel latency of the ARM implementation.
- Faster track kernel
- Stable Error: ATE ~=18-19m



Conclusion

- Optimal Implementation: Latency = 14.68
- Almost half kernel latency of the ARM implementation.
- Faster track kernel
- Stable Error: ATE ~=18-19m



Thank you for your attention