UNIVERSITÄT PADERBORN

Paderborn Center for Parallel Computing

CUSTONN2: CUSTOMIZING NEURAL NETWORKS ON FPGAS

supervised by
Prof. Dr. Christian Plessel and Dr. Tobias Kenter

23/09/2019

UNIVERSITÄT PADERBORN

Project Group Members

Aayush Suresh Bansal

Anshul Suresh Bansal

Nikhitha Shivaswamy

Suprajith Suresh Hakathur

Amay Churi

Adesh Shambhu

Rushikesh Vinay Nagle

Chiranjeevi Hongalli Revanna

Alina Egorova

Arathy Ajaya Kumar

UNIVERSITÄT PADERBORN

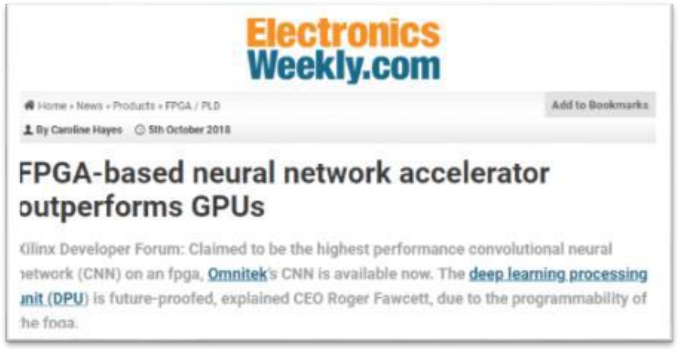
Content

- Motivation and goals
- Introduction to OpenCL and infrastructure
- Project planning and designing
- Tools
- Evaluation and Result
- Future tasks
- Summary

UNIVERSITÄT PADERBORN

Motivation

- Convolution neural networks (CNNs) are multi-layered architectures well suited for image classification.
- Field Programmable Gate Array (FPGA) is a hardware accelerator device.
- CNNs perform better with FPGAs as compared to other hardware platforms.
- The primary motivation is to investigate performance after scaling on multiple FPGAs.


Electronics Weekly.com

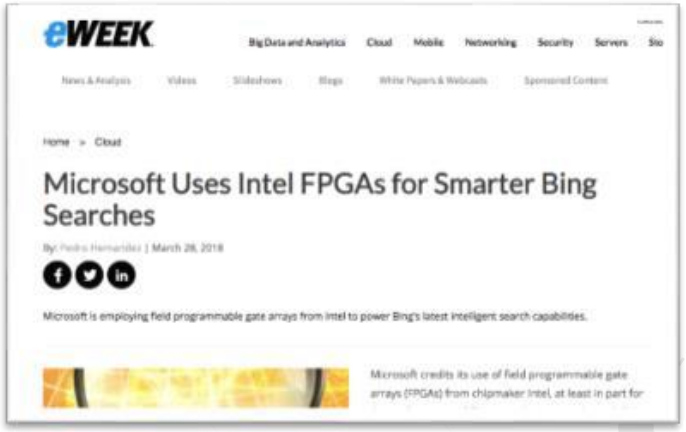
Home » News » Products » FPGA / PLD

Add to Bookmarks

By Caroline Hayes | 5th October 2018

FPGA-based neural network accelerator outperforms GPUs

Xilinx Developer Forum: Claimed to be the highest performance convolutional neural network (CNN) on an fpga, Omnitek's CNN is available now. The deep learning processing unit (DPU) is future-proofed, explained CEO Roger Fawcett, due to the programmability of the fpga.


eWEEK

Big Data and Analytics | Cloud | Mobile | Networking | Security | Servers | Sto

News & Analysis | Videos | Slideshows | Blogs | White Papers & Webcasts | Sponsored Content

Home > Cloud

Microsoft Uses Intel FPGAs for Smarter Bing Searches

By Pedro Hernandez | March 28, 2018

Microsoft is employing field programmable gate arrays from Intel to power Bing's latest intelligent search capabilities.

Microsoft credits its use of field programmable gate arrays (FPGAs) from chipmaker Intel, at least in part for

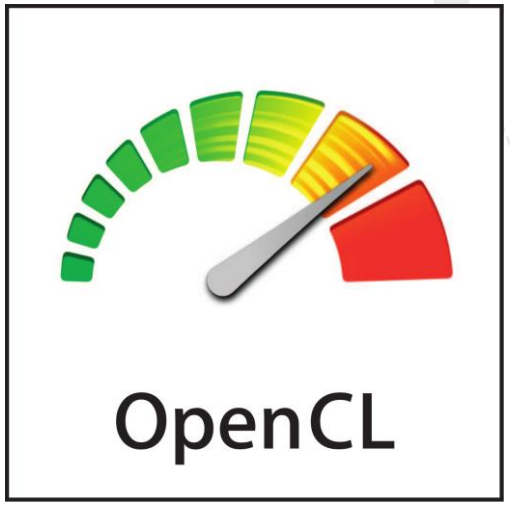
UNIVERSITÄT PADERBORN

Goals

- Deployment of state-of-the-art CNN models on FPGA
- Stretching the CNN model on the Noctua FPGA Infrastructure at PC^2 with 32 Stratix 10 FPGAs
- Quantization on CNN floating point weights
- Compare results with state-of-the-art benchmarks

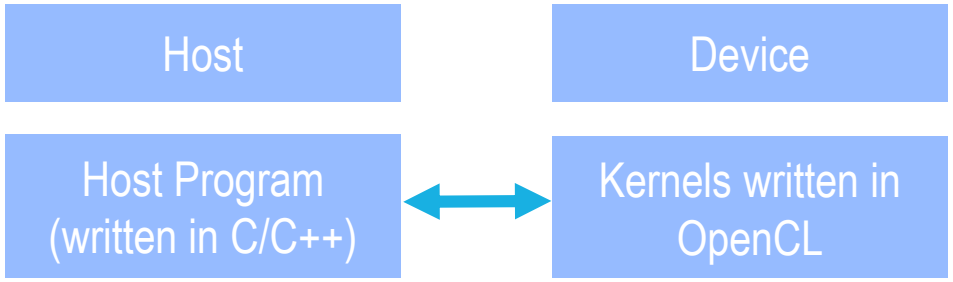
UNIVERSITÄT PADERBORN

Introduction to OpenCL



OpenCL (Open Computing Language)

- Parallel programming standard based on C/C++
- Used in heterogenous computing.
- Achieves high performance using task level and data level parallelism
- Supported platforms include CPUs, GPUs, DSPs, FPGAs



UNIVERSITÄT PADERBORN

Parallelism using OpenCL features

Loop Unrolling

Idea: Increase the number of operations per cycle thereby decreasing the number of iterations.

- Increases the hardware resource consumption
- OpenCL provides #pragma unroll extension for unrolling loops
- Syntax
  #pragma unroll [unroll-factor]

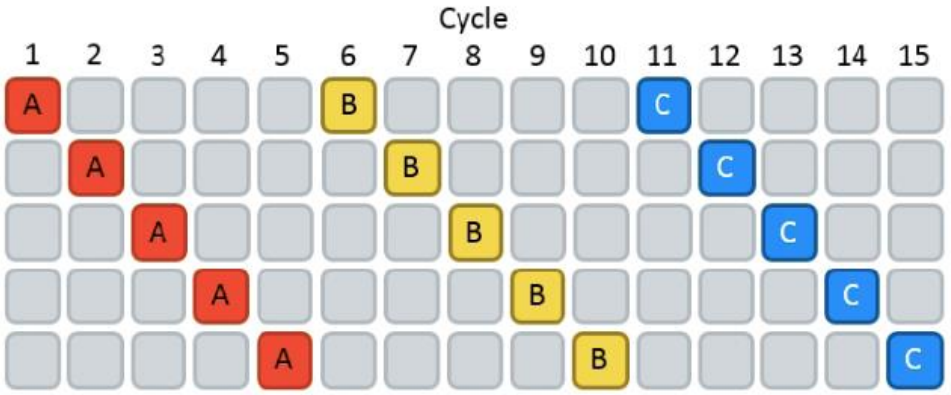
```
int m = 0;
#pragma unroll 1
for (int k = 0; k < N; k++) {
  m += out[k/3];
}
#pragma unroll
for (int k = 0; k.< 6; k++) {
  m += out[k];
}
#pragma unroll 2
for (int k = 0; k < 6; k++) {
  m += out[k];
}
out[2] = m;
```

UNIVERSITÄT PADERBORN

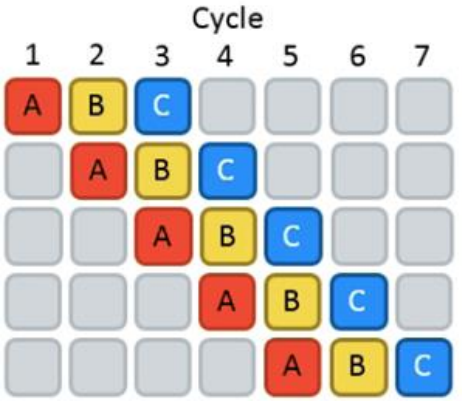
Pipelining

Allows the various hardware sections to process a different loop iteration simultaneously

NO LOOP PIPELINING



WITH LOOP PIPELINING



UNIVERSITÄT PADERBORN

Noctua Infrastructure

- 16 FPGA nodes, each with 2 Intel Stratix 10 FPGAs
- 256 Compute nodes
- Application specific interconnect between FPGAs
  - 4 x 40 Gbps links per FPGA
  - Configurable point to point connections between FPGAs
  - Accessible as serial (I/O) channels from OpenCL BSP

UNIVERSITÄT PADERBORN

Stratix 10

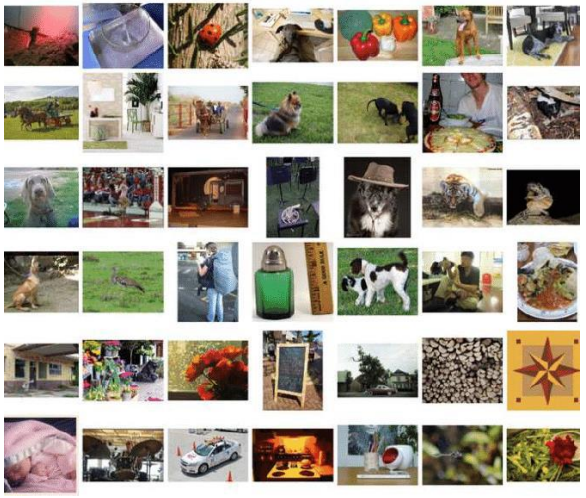
- 5500K Logic Elements
- 229 Mbits Embedded Memory
- 1640 GPIO pins
- 5760 DSP blocks

UNIVERSITÄT PADERBORN

Project plan / Datasets

Reasons for selecting ImageNet:

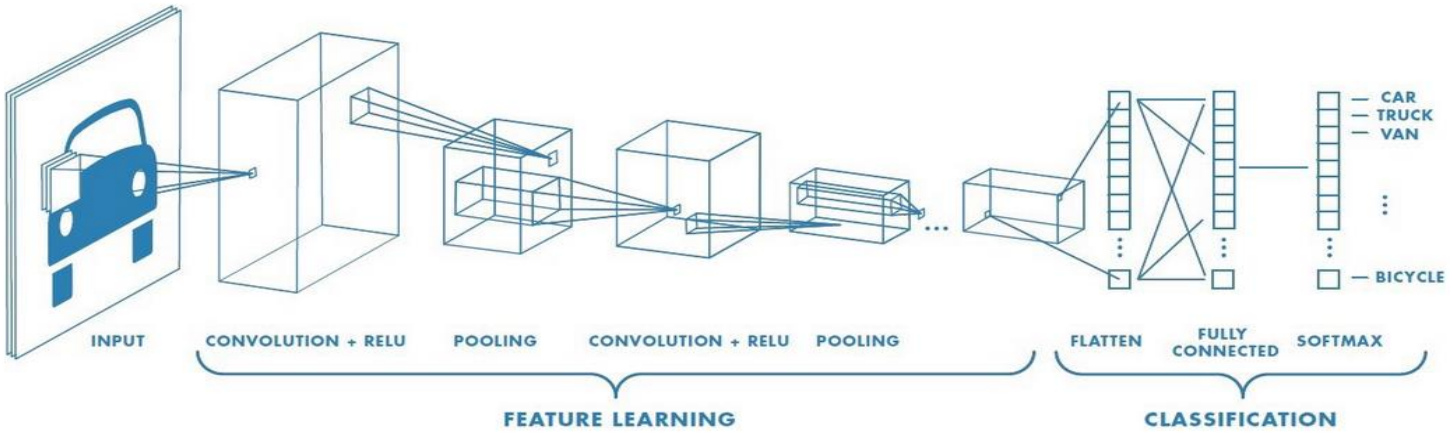
- DNN topologies in the project are well optimized for ImageNet.
- Their result can be used as a benchmarking performance.



ImageNet
14 million images
Over 20000 categories

UNIVERSITÄT PADERBORN

Convolutional Neural Networks (CNNs)

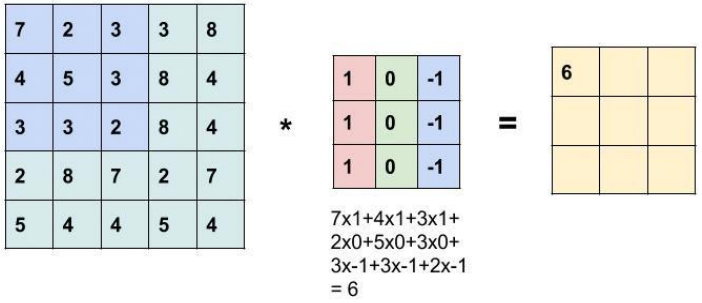


- Commonly used in image classification
- Can distinguish between images by capturing spatial and temporal dependencies using filter operations (Convolutions)

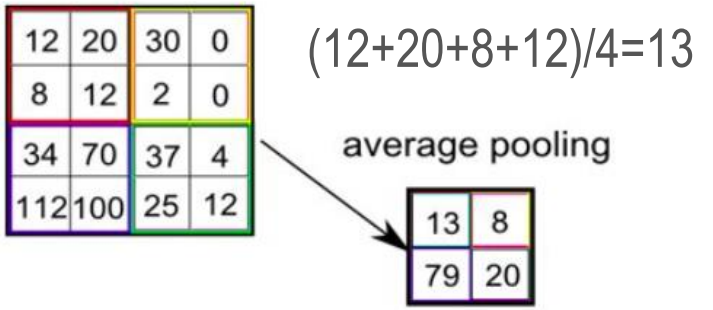
UNIVERSITÄT PADERBORN

CNN layers

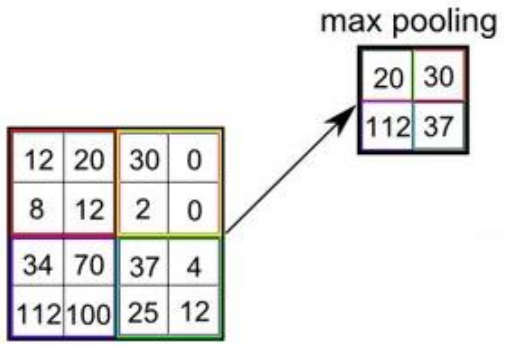
Convolution layer



Average Pooling



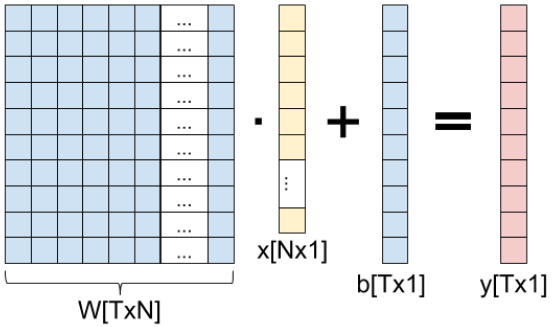
Max Pooling

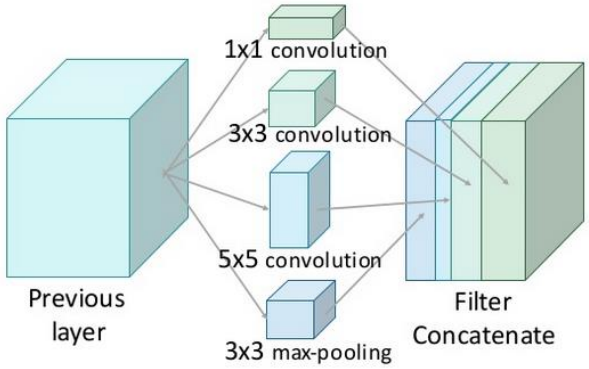


UNIVERSITÄT PADERBORN

CNN layers

Fully connected layer





Concatenation layer

UNIVERSITÄT PADERBORN

CNN layers

Eltwise

- Performs element-wise matrix addition
- Adds layer weights to the input

Scaleshift

- Performs a linear transformation on its input
- Typically used for batch normalization
- Y = Wx + b

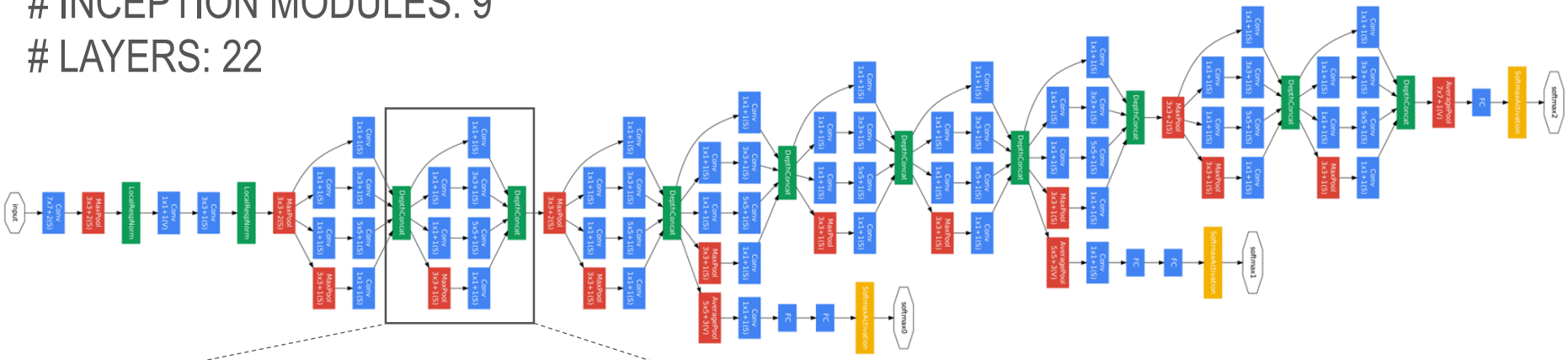
Softmax

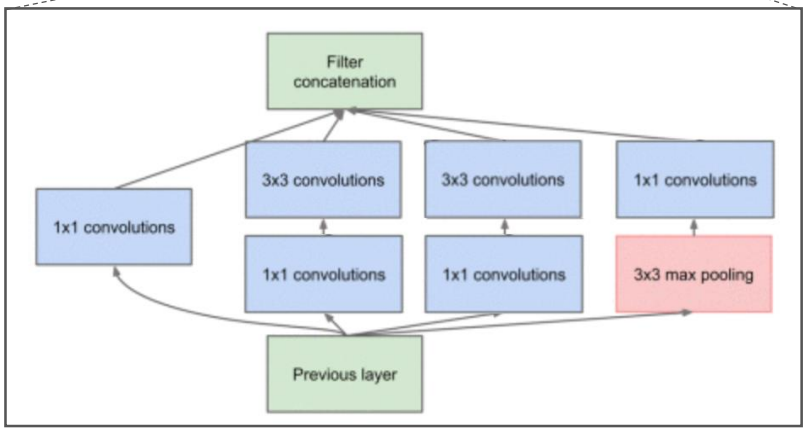
- Converts real valued scores from the previous layer to probabilities

UNIVERSITÄT PADERBORN

Project plan / GoogLeNet (Inception-v1)

INCEPTION MODULES: 9
LAYERS: 22





Inception module with dimension reductions

Total no of Ops: **2991M**

Convolution: 99%

Maxpool

Softmax 1%

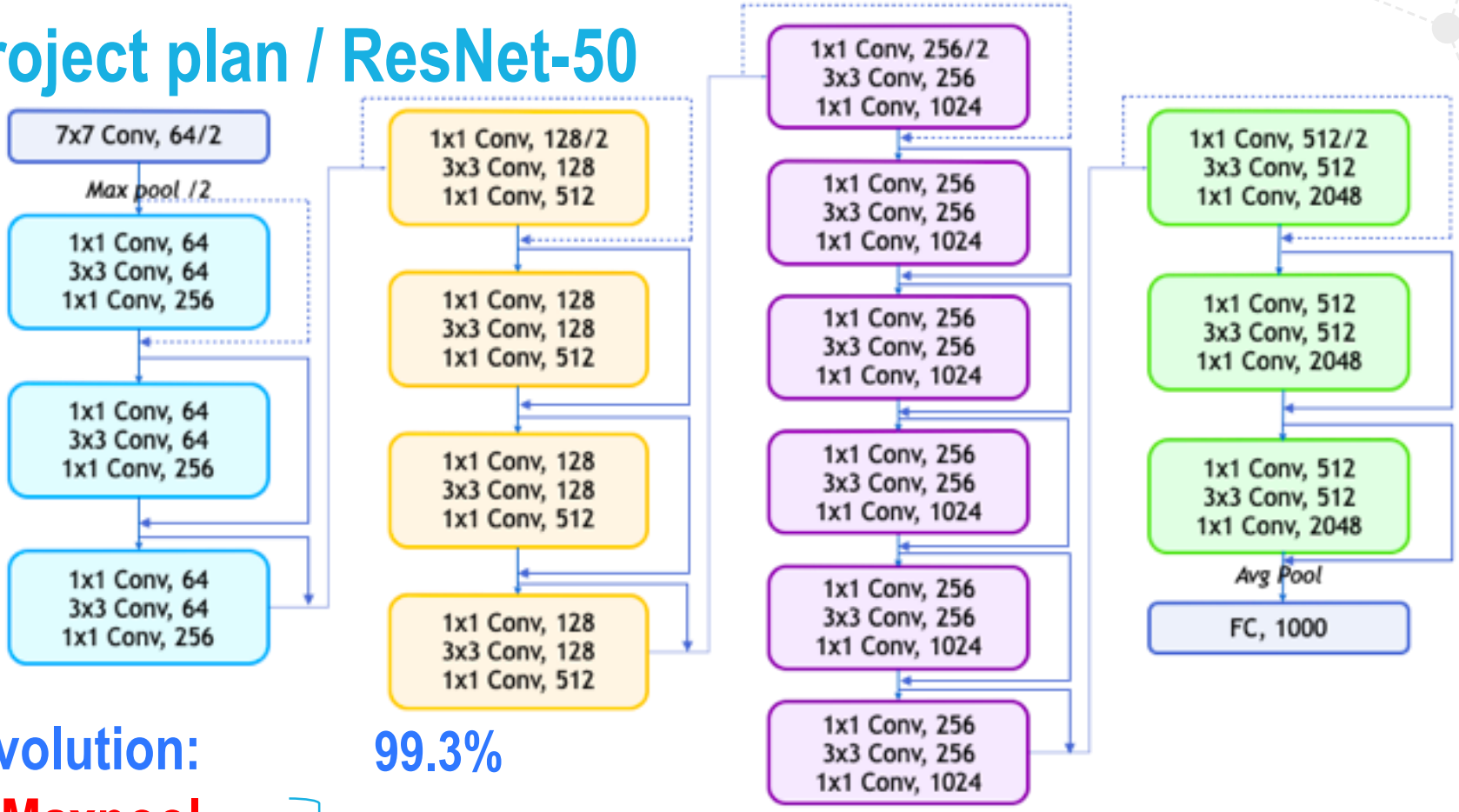
Concat

C. Szegedy et al., "Going deeper with convolutions", *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 1-9, Jun. 2015.

UNIVERSITÄT PADERBORN

Project plan / ResNet-50

BLOCKS: 4
LAYERS: 50



Convolution: 99.3%

Maxpool

Softmax 0.7%

Concat

Total no of Ops: **6940M**

Kordík, P. 2019. Introducing Conveiro, an open source library for visualizing convolutional neural networks. https://tech.showmax.com/2019/04/conveiro/

UNIVERSITÄT PADERBORN

Project Organisation

UNIVERSITÄT PADERBORN

Project Organisation

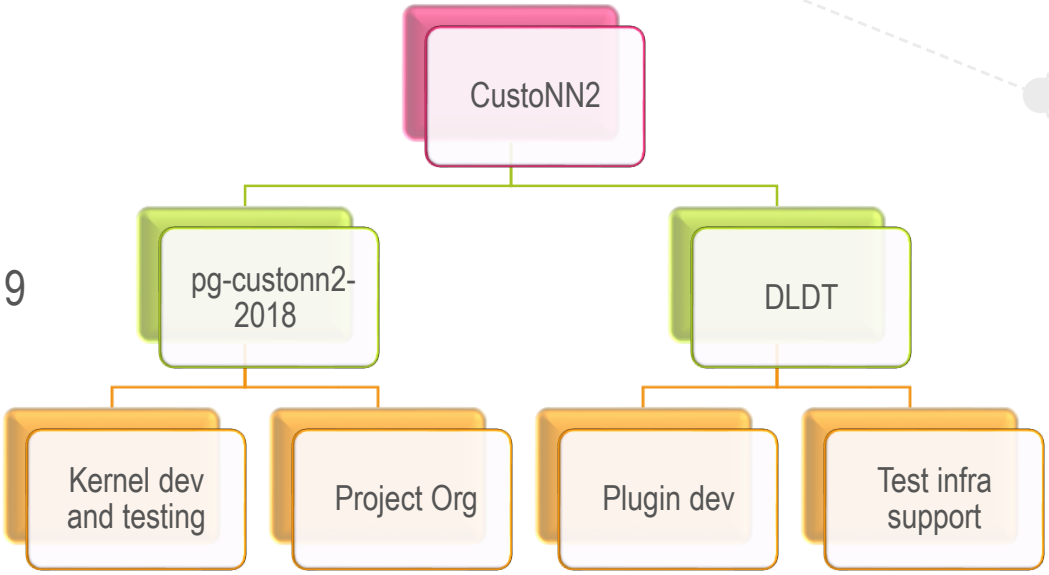
4 - 5 months	Tutorial phase
2 - 3 months	Research phase
rest time	Implementation phase

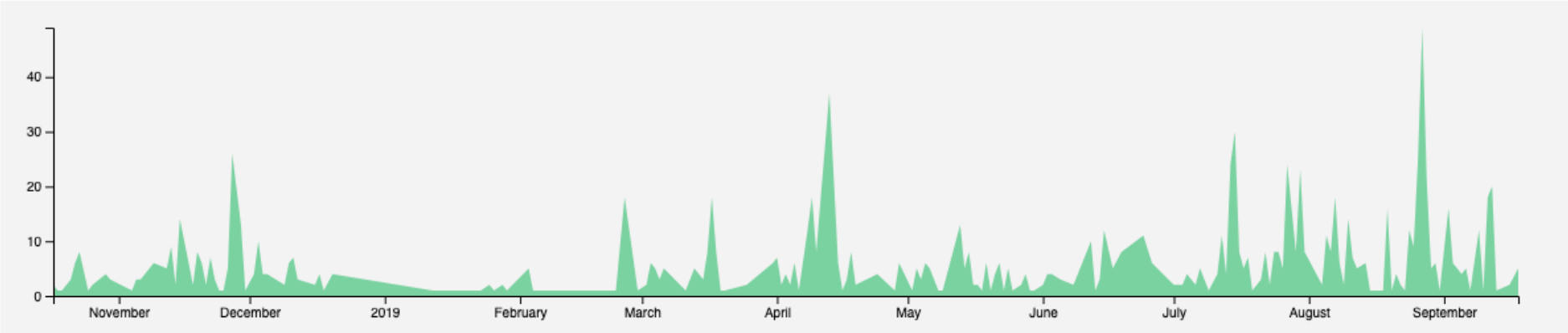
UNIVERSITÄT PADERBORN

Project Organisation

October 17, 2018 – September 18, 2019

- Total: +**1450 commits**
- Average per day: 4.4 commits





UNIVERSITÄT PADERBORN

Project Organisation

Google calendar

Meetings

- Status Updates
- Task creation
- Minutes of the meetings

Tasks

- Gitlab issue board
- Task distribution

Issue tracking

- Issue board with labels
- Issue owner

UNIVERSITÄT PADERBORN

Project Organisation

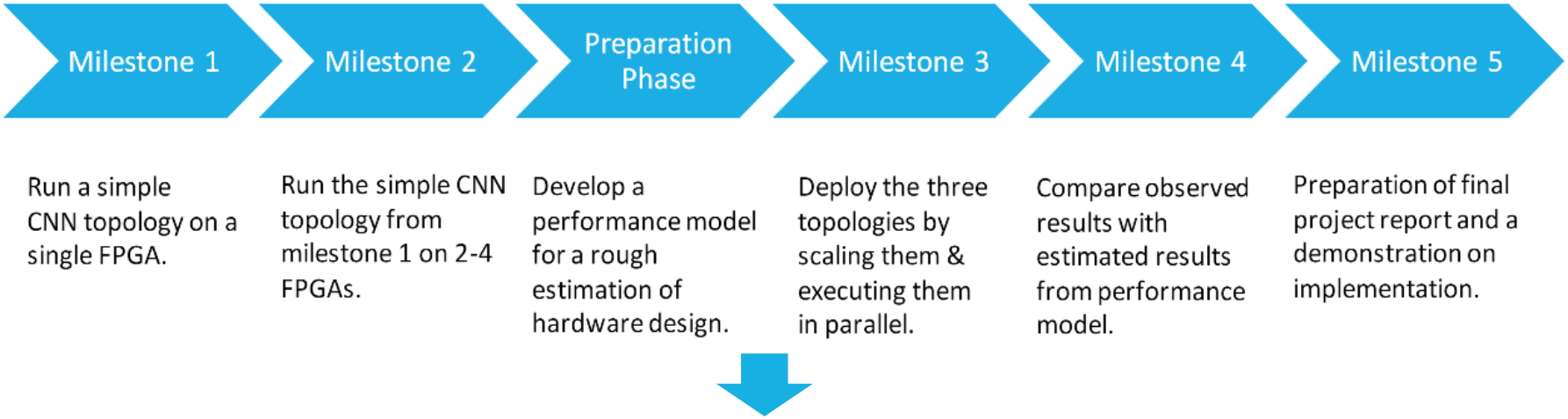
Initial plan:

Milestone 1	Milestone 2	Preparation Phase	Milestone 3	Milestone 4	Milestone 5
Run a simple CNN topology on a single FPGA.	Run the simple CNN topology from milestone 1 on 2-4 FPGAs.	Develop a performance model for a rough estimation of hardware design.	Deploy the three topologies by scaling them & executing them in parallel.	Compare observed results with estimated results from performance model.	Preparation of final project report and a demonstration on implementation.

UNIVERSITÄT PADERBORN

Project Organisation

Initial plan:



Too restrictive. Hard to factor unexpected issues.

UNIVERSITÄT PADERBORN

Project Organisation

Initial plan:

Milestone 1	Milestone 2	Preparation Phase	Milestone 3	Milestone 4	Milestone 5
Run a simple CNN topology on a single FPGA.	Run the simple CNN topology from milestone 1 on 2-4 FPGAs.	Develop a performance model for a rough estimation of hardware design.	Deploy the three topologies by scaling them & executing them in parallel.	Compare observed results with estimated results from performance model.	Preparation of final project report and a demonstration on implementation.

Too restrictive. Hard to factor unexpected issues.

free-flowing (on the fly) task scheduling
method keeping objectives of Milestones as reference.

UNIVERSITÄT PADERBORN

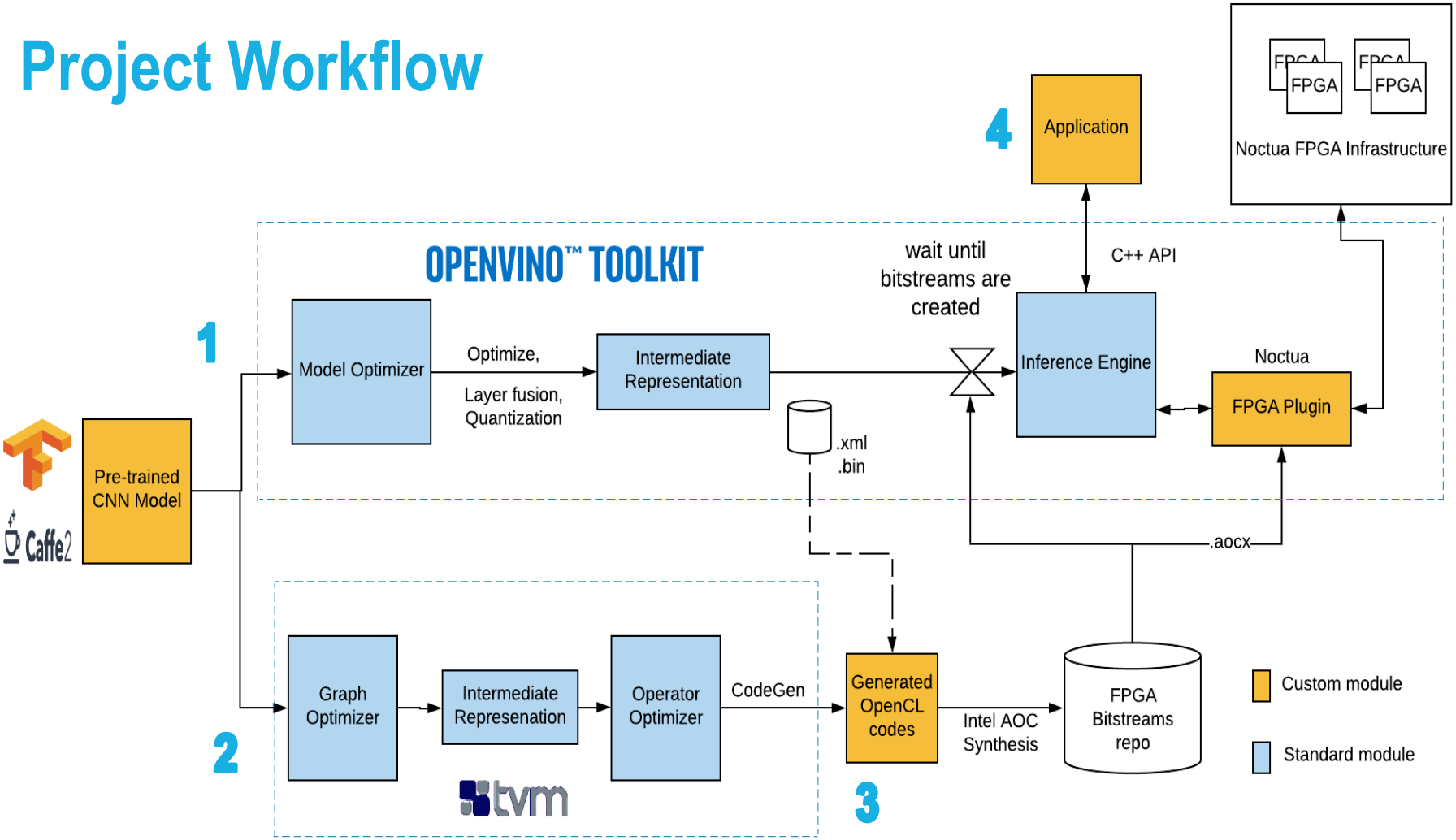
Project Organisation

Team Leader:

- Rotating Team Lead role
- Elected by team members by voting
- Liaison between the team and our supervisor
- Benevolent distributor of tasks!

UNIVERSITÄT PADERBORN

Project Workflow



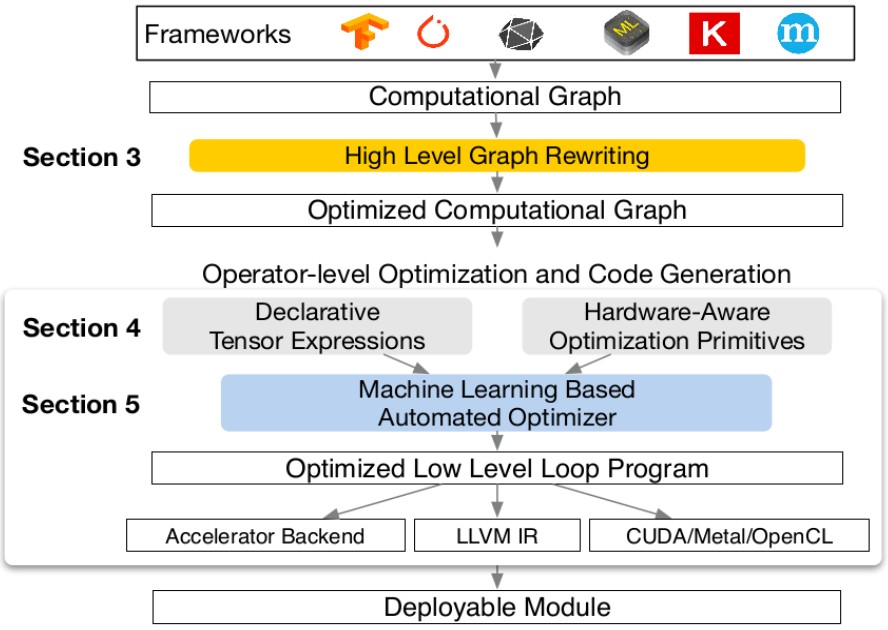
UNIVERSITÄT PADERBORN

Tools / Tensor Virtual Machine (TVM)

Tensor virtual machine (TVM) is an open deep learning compiler stack for CPUs, GPUs, and specialized accelerators.

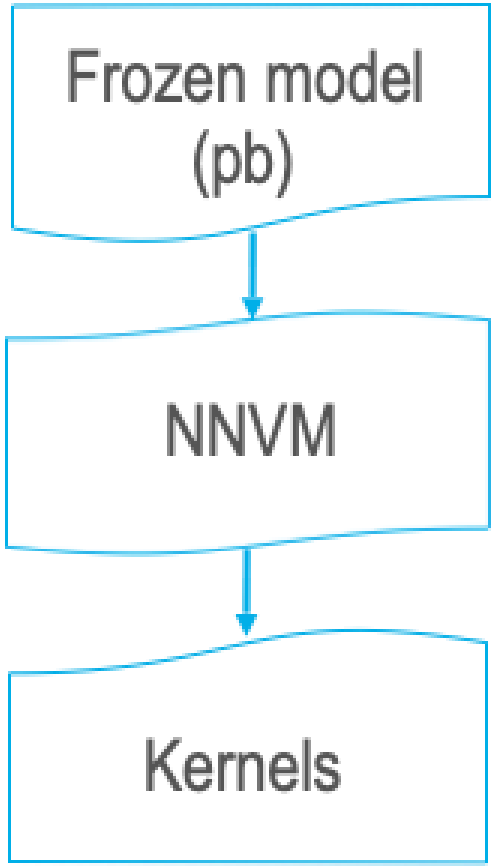
TVM is divided into two major component:

- TVM compiler, which does all the compilation and optimizations
- TVM runtime, which runs on the target devices.



UNIVERSITÄT PADERBORN

Tools / Tensor Virtual Machine (TVM)



Contains the graph definition, weights and bias of model

Neural Network Virtual Machine (NNVM):

- High-level optimizations during kernel generation support different hardware backend
- Various types of optimizations levels

Generated by TVM

UNIVERSITÄT PADERBORN

Tools / Tensor Virtual Machine (TVM)

NNVM Frontend

- NNVM API provides us with nnvm.frontend to parse the model from protobuf file.
- It provides us with nnvm expressions and parameters from the protobuf (tensors).

NNVM Compiler

- Expression and parameters obtained from frontend are passed to the compiler along with optimization level.
- The compiler provides us with a final graph (IR) after optimization and parameters after optimizations.

UNIVERSITÄT PADERBORN

Tensor Virtual Machine - Generated kernels

Padding kernel

```
__kernel void fuse_transpose_pad_kernel0(__global float* restrict T_pad, __global float* restrict input0) {
  for (int ax0_ax1_fused_ax2_fused_ax3_fused_inner = 0; ax0_ax1_fused_ax2_fused_ax3_fused_inner < 157323; ++ax0_ax1
    T_pad[ax0_ax1_fused_ax2_fused_ax3_fused_inner] = (float)(((((458 <= (ax0_ax1_fused_ax2_fused_ax3_fused_inner %
  }
}
```

Convolution kernel

```
__kernel void fuse_conv2d_kernel0(__global float* restrict compute, __global float* restrict input0, __global float
  for (int ff = 0; ff < 64; ++ff) {
    for (int yy = 0; yy < 112; ++yy) {
      for (int xx = 0; xx < 112; ++xx) {
        compute[(((ff * 112) + yy) * 112) + xx)] = 0.000000e+00f;
        for (int rc = 0; rc < 3; ++rc) {
          for (int ry = 0; ry < 7; ++ry) {
            for (int rx = 0; rx < 7; ++rx) {
              compute[(((ff * 112) + yy) * 112) + xx)] = (compute[(((ff * 112) + yy) * 112) + xx)] + (input0[((((
            }
          }
        }
      }
    }
  }
}
```

RELU

```
__kernel void fuse_transpose_broadcast_mul_broadcast_add_relu_transpose_kernel0(__global float* restrict T_transpos
  for (int ax0_ax1_fused_ax2_fused_ax3_fused_inner = 0; ax0_ax1_fused_ax2_fused_ax3_fused_inner < 802816; ++ax0_ax1
    T_transpose[ax0_ax1_fused_ax2_fused_ax3_fused_inner] = max(((input0[ax0_ax1_fused_ax2_fused_ax3_fused_inner] *
  }
}
```

UNIVERSITÄT PADERBORN

Tensor Virtual Machine - Custom kernels

Padding kernel

```
__kernel void Padding_Conv2d_1a_7x7_Conv2D(__global float *restrict T_pad, __global float *restrict input0)
{
    for (int ax0_ax1_fused_ax2_fused_ax3_fused_inner = 0; ax0_ax1_fused_ax2_fused_ax3_fused_inner < 157323; ++ax0_ax1_fused_ax2_fuse
    {
        T_pad[ax0_ax1_fused_ax2_fused_ax3_fused_inner] = (float)((((((458 <= (ax0_ax1_fused_ax2_fused_ax3_fused_inner % 52441)) && ((
    }
}
```

Convolution kernel + RELU

```
__kernel void Conv2d_1a_7x7_Conv2D(__global float *restrict compute, __global float *restrict input0, __global float *restrict input
{
    for (int ff = 0; ff < 64; ++ff)
    {
        for (int yy = 0; yy < 112; ++yy)
        {
            for (int xx = 0; xx < 112; ++xx)
            {
                compute[(((((ff * 112) + yy) * 112) + xx)] = input2[ff];
                for (int rc = 0; rc < 3; ++rc)
                {
                    for (int ry = 0; ry < 7; ++ry)
                    {
                        for (int rx = 0; rx < 7; ++rx)
                        {
                            compute[(((((ff * 112) + yy) * 112) + xx)] = (compute[(((((ff * 112) + yy) * 112) + xx)] + (input0[((((((rc
                        }
                    }
                }
                compute[(((((ff * 112) + yy) * 112) + xx)] = (compute[(((((ff * 112) + yy) * 112) + xx)] > 0) ? compute[(((((ff * 112)
}}}}
```

UNIVERSITÄT PADERBORN

Tensor Virtual Machine

GoogLeNet:

- Number of kernels generated by TVM **- 116**
- Required number of kernels according to OpenVINO IR – **104**

In case of GoogleNet, TVM generates additional transpose kernels which are handled by plugin.

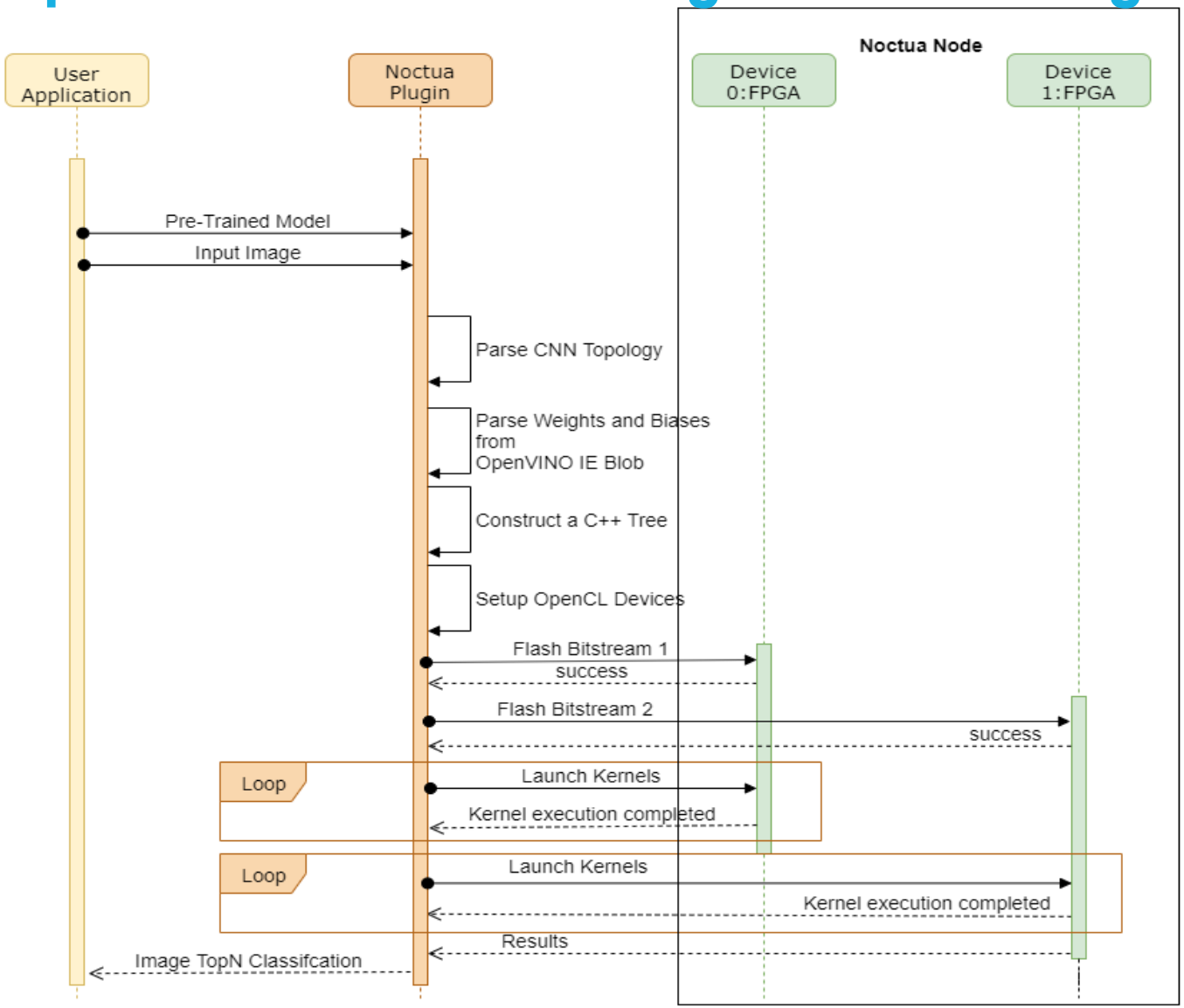
ResNet:

- Number of kernels generated by TVM **- 75**
- Required number of kernels according to OpenVINO IR **- 146**

In case of ResNet, TVM generates only 1 kernel for all the layers which have same I/O dimensions in OpenVINO IR.

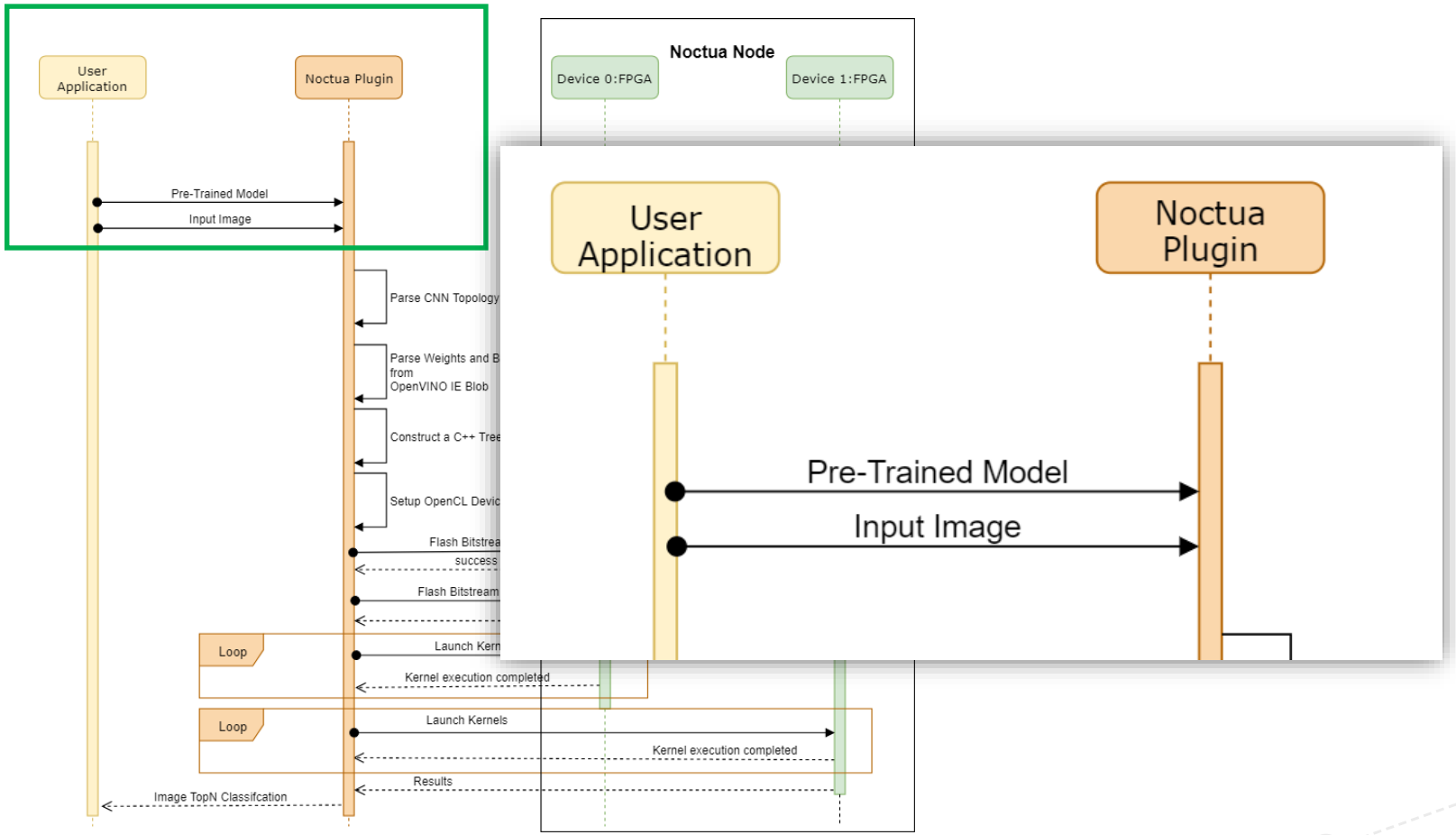
UNIVERSITÄT PADERBORN

Intel OpenVINO Inference Engine FPGA Plugin



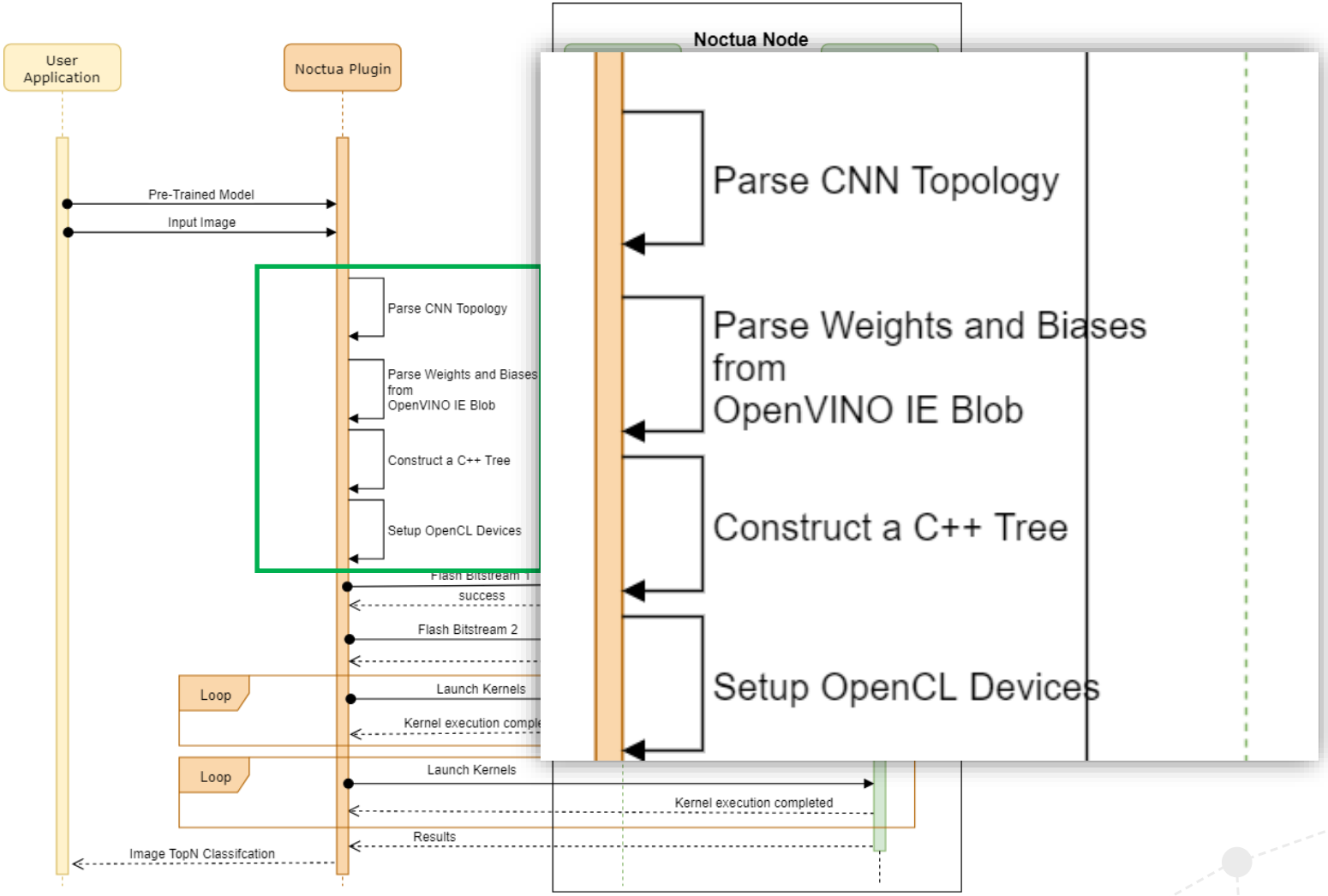
UNIVERSITÄT PADERBORN

Intel OpenVINO Inference Engine FPGA Plugin



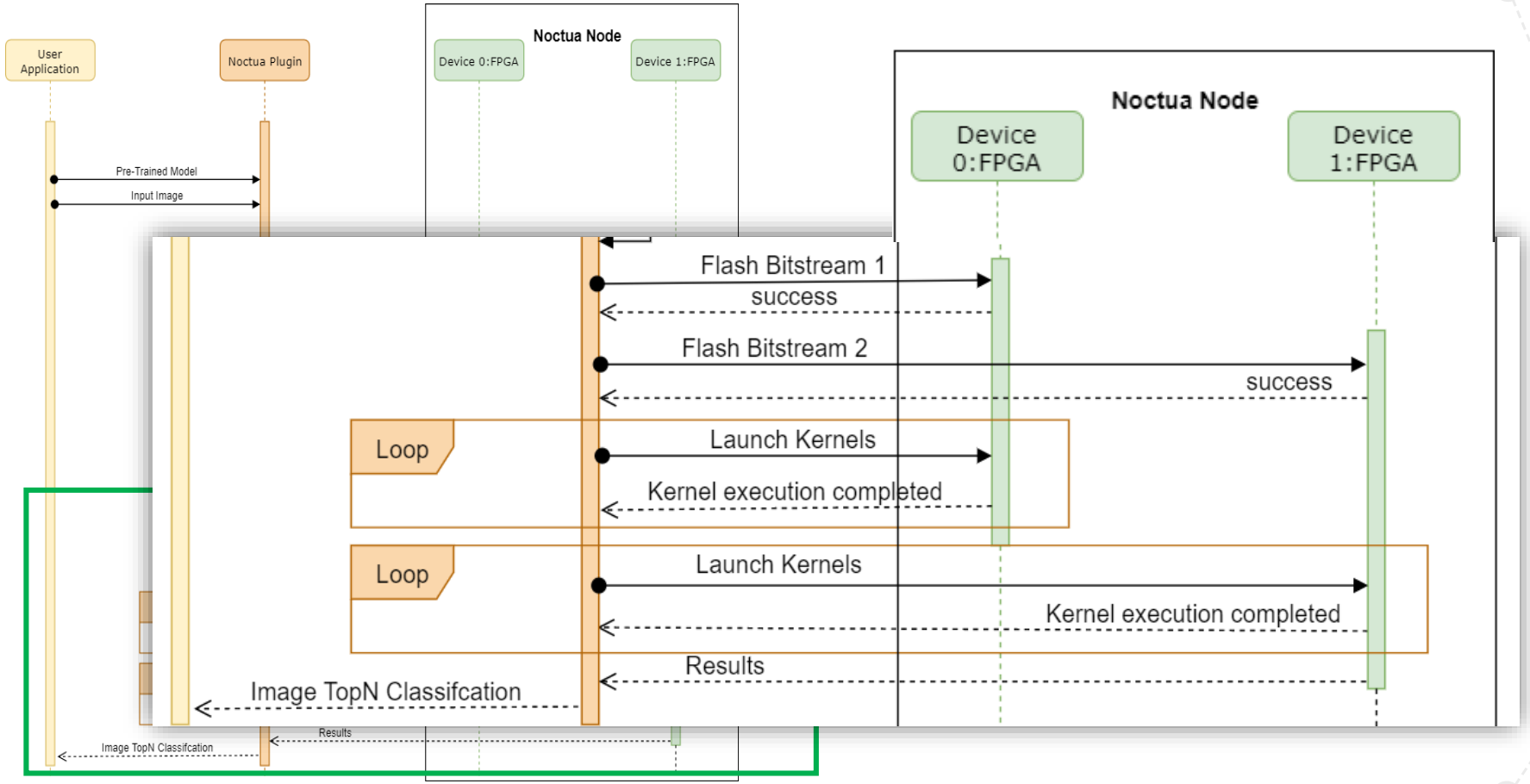
UNIVERSITÄT PADERBORN

Intel OpenVINO Inference Engine FPGA Plugin



UNIVERSITÄT PADERBORN

Intel OpenVINO Inference Engine FPGA Plugin



UNIVERSITÄT PADERBORN

Challenges

- OpenVINO FPGA plugin is not open source
- Understanding the complex OpenVINO Inference Engine framework
- Understanding IE Data structure (for ex: Blobs, CNNLayer, Network)
- Many functions are deprecated methods in the OpenVINO codebase
- OpenVINO does not support scaling
- Layout mismatch between OpenVINO and TVM (NCHW and NHWC)

UNIVERSITÄT PADERBORN

Classification results

Top-10 Image Classification for elephant.png // **GoogLeNet**

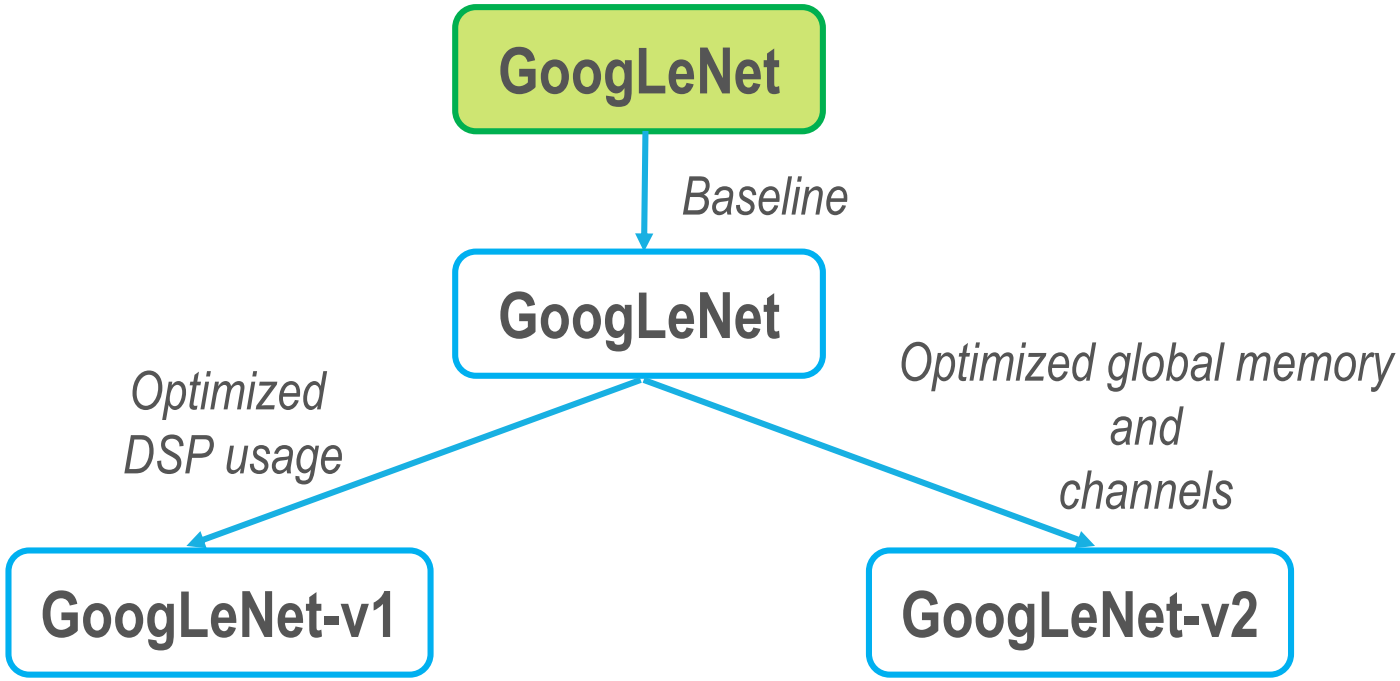
Label index **102** (**51.6359%**) - Label : **tusker**
Label index 387 (40.6516%) - Label : African elephant, Loxodonta africana
Label index 386 (4.2946%) - Label : Indian elephant, Elephas maximus
Label index 150 (0.0435517%) - Label : dugong, Dugong dugon
Label index 344 (0.0421621%) - Label : warthog
Label index 367 (0.0354225%) - Label : gorilla, Gorilla gorilla
Label index 491 (0.027498%) - Label : chain mail, ring mail, mail, chain armor, chain armour, ring armor, ring armour
Label index 234 (0.027498%) - Label : Bouvier des Flandres, Bouviers des Flandres
Label index 996 (0.0257801%) - Label : earthstar
Label index 269 (0.0255324%)- Label : Mexican hairless

Top-10 Image Classification for elephant.png // **ResNet-50**

Label index **387** (**79.871%**) - Label : **African elephant, Loxodonta africana**
Label index 102 (19.3072%) - Label : tusker
Label index 386 (0.820408%) - Label : Indian elephant, Elephas maximus
Label index 52 (8.094444e-05%) - Label : triceratops
Label index 491 (5.10283e-05%) - Label : chain mail, ring mail, mail, chain armor, chain armour, ring armor, ring armour
Label index 367 (2.77504e-05%) - Label : gorilla, Gorilla gorilla
Label index 907 (2.72284e-05%) - Label : Windsor tie
Label index 49 (1.95787e-05%) - Label : Granny Smith
Label index 948 (1.76418e-05%) - Label : mushroom
Label index 835 (1.73147e-05%) - Label : suit, suit of clothes

UNIVERSITÄT PADERBORN

GoogLeNet Optimization sequence



UNIVERSITÄT PADERBORN

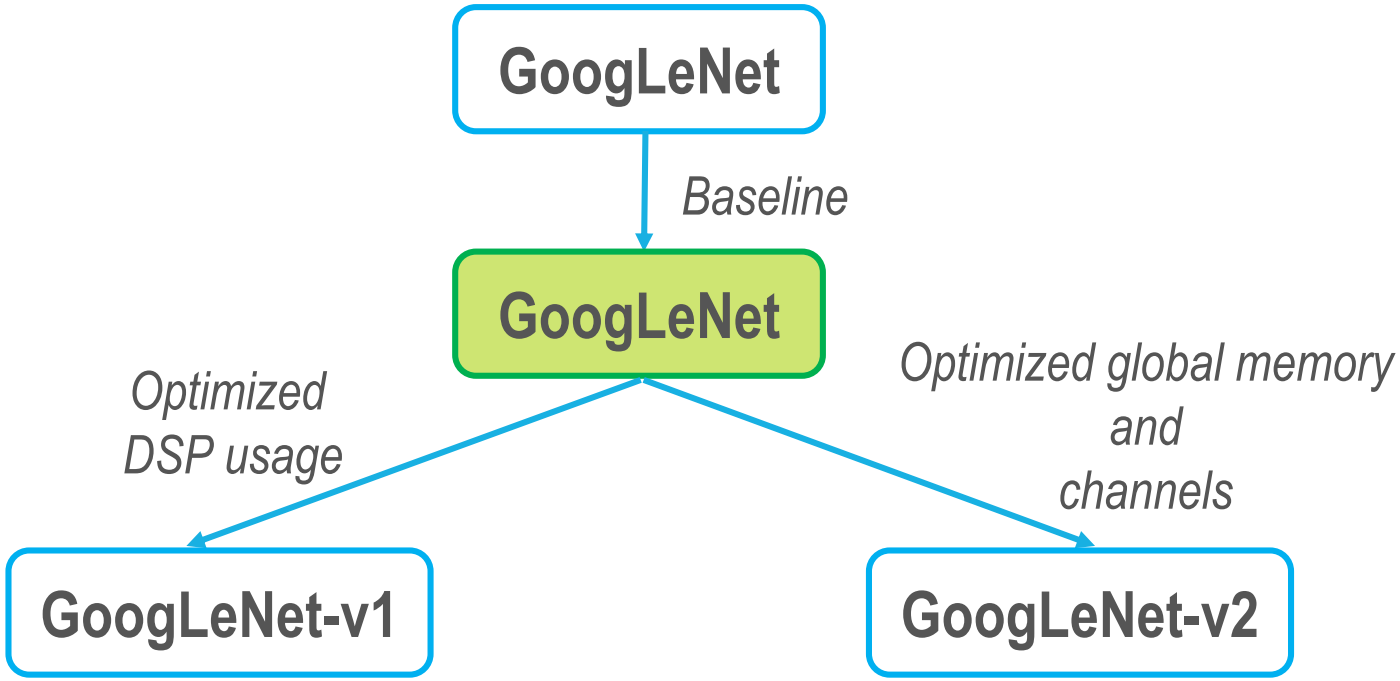
Performances / GoogLeNet (Inception-v1)

Whole network without optimizations requires:

Resource	Usage
Logic utilization	178%
ALUTs	97%
Dedicated logic registers	88%
Memory blocks	138%
DSP blocks	24%

UNIVERSITÄT PADERBORN

GoogLeNet Optimization sequence



UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Inception-v1)

Step 1: Run one file in emulation mode for checking the correctness

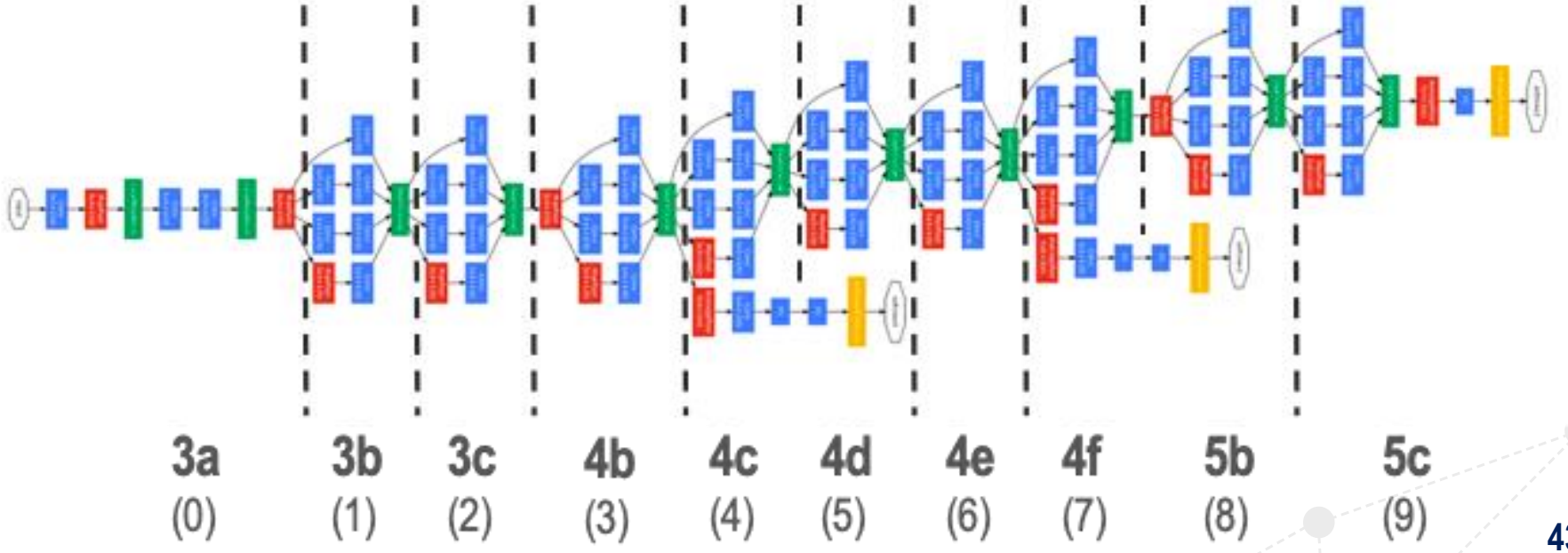
Step 2: Divide all kernels on several files(.aocx) which fit FPGA

UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Inception-v1)

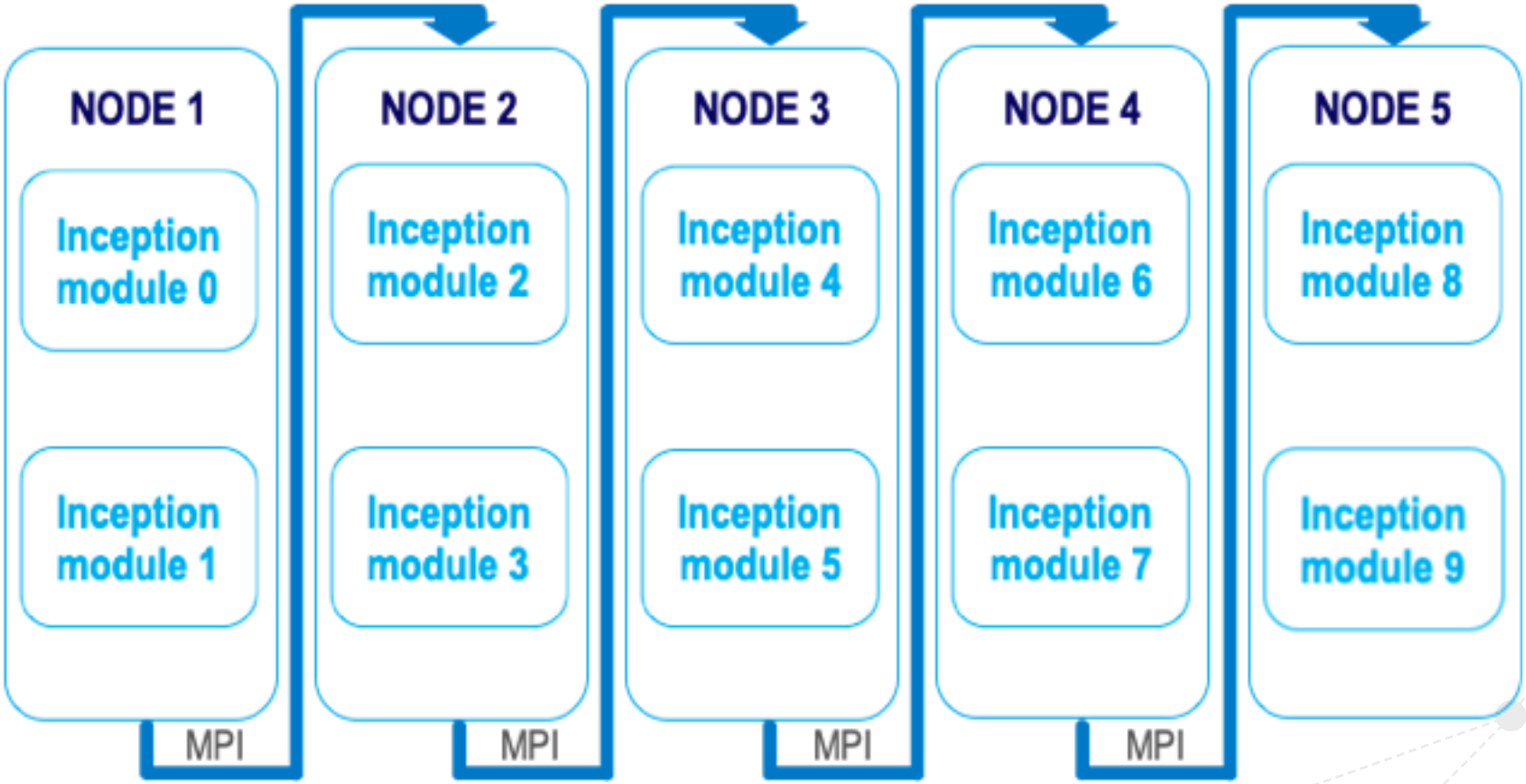
Each block is called an Inception Module and each block ends with concat layer

Convolution
Maxpool
Softmax
Concat



UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Baseline)



UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Baseline)

/ test_plugin / **main.cpp**

```
#include "mpi.h"

MPI_Init(NULL, NULL);
int rank;
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
fpga_launcher(network,inputModel,imageNames,
                                    cnn_model,rank);
```

UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Baseline)

/ noctua_plugin / **fpga_plugin.cpp**

```
#include "mpi.h"

std::string file1 = GoogLeNet_DIR+"inception"+std::to_string(2*rank)+".aocx";
std::string file2 = GoogLeNet_DIR+"inception"+std::to_string(2*rank+1)+".aocx";

std::string file1_xml = GoogLeNet_DIR+"inception"+std::to_string(2*rank)+".xml";
std::string file2_xml = GoogLeNet_DIR+"inception"+std::to_string(2*rank+1)+".xml";

char f1_xml[file1_xml.length()];
strcpy(f1_xml,file1_xml.c_str());

std::vector<std::string> first_kernels = xml_parser1(f1_xml);
```

UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Baseline)

/ noctua_plugin / **fpga_plugin.cpp**

```
// MPI write to the next host instance
if(rank<com_sz-1&&program_number == 2)
{
        std::string concat_layer_name = p->layerName;
        MPI_Send(concat_layer_name.c_str(), concat_layer_name.size(), MPI_CHAR,
        int dims1 = p->outH * p->outW * p->outDepth;
        MPI_Send(&dims1, 1, MPI_INT, rank+1, 0, MPI_COMM_WORLD);
        float concat_out[p->outH * p->outW * p->outDepth];
        cmd_queues[p->layerID]->enqueueReadBuffer(*buffers[p->layerOutBufferInde
        MPI_Send(concat_out, p->outH * p->outW * p->outDepth, MPI_FLOAT, rank+1,
}
```

UNIVERSITÄT PADERBORN

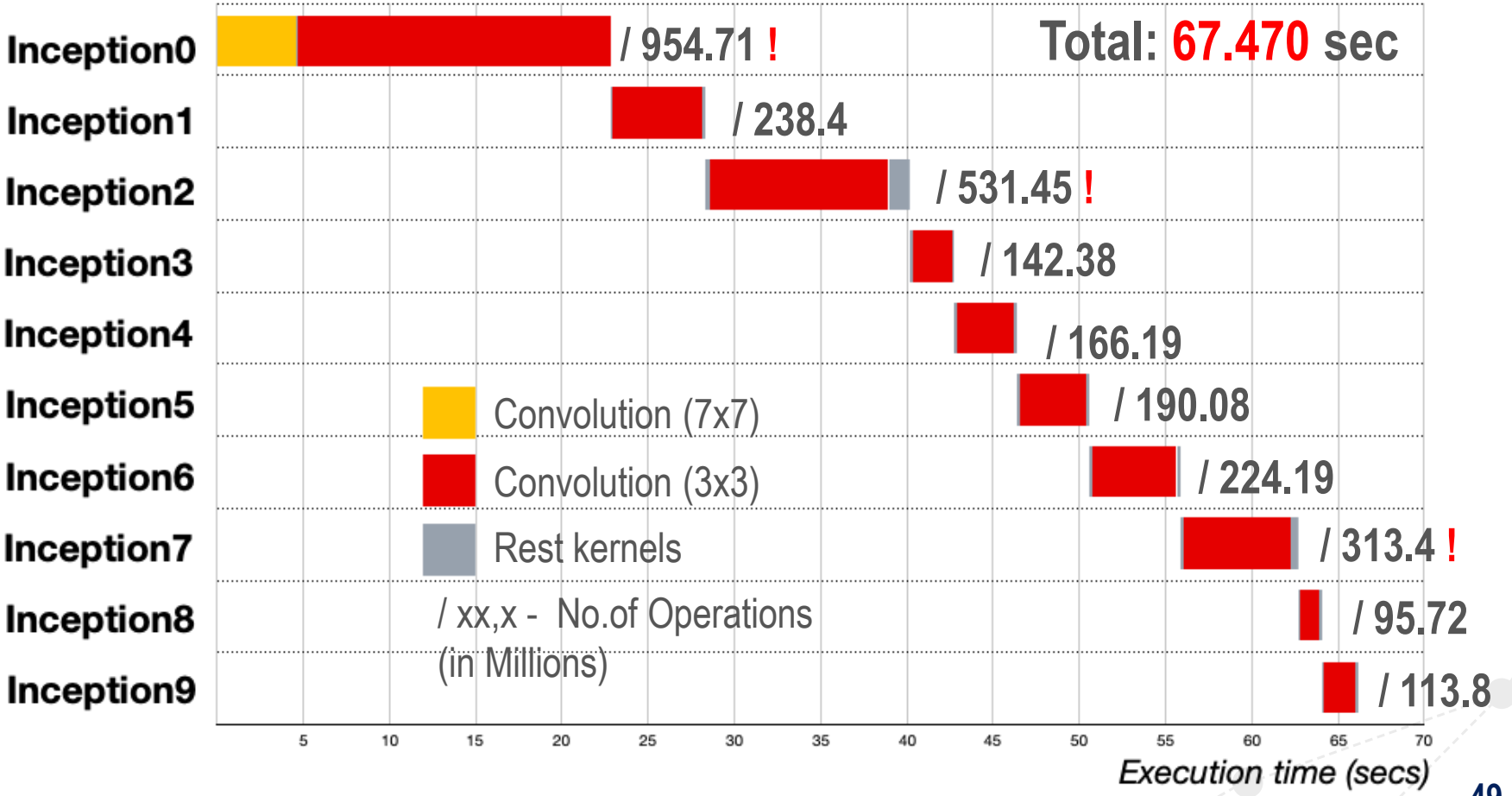
Performances / GoogLeNet (Baseline)

/ noctua_plugin / **fpga_plugin.cpp**

```
if(rank!=0)
{
        char layer_name[50];
        MPI_Recv(layer_name, 50, MPI_CHAR, rank - 1, 0, MPI_COMM_WORLD,MPI_STATUS_IGNORE)

        std::string previous_l_name(layer_name);
        int dims_prev;
        MPI_Recv(&dims_prev, 1, MPI_INT, rank - 1, 0, MPI_COMM_WORLD,MPI_STATUS_IGNORE);

        float prev_data[dims_prev];
        MPI_Recv(prev_data, dims_prev, MPI_FLOAT, rank - 1, 0,MPI_COMM_WORLD,MPI_STATUS_I
```

UNIVERSITÄT PADERBORN

Performance modelling - GoogLeNet (Baseline)



UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Baseline)

Li...	Source: inception_3a.cl	Attributes	Stall%	Occupa...	Bandwidth
79	}				
80	}				
81					
82	__kernel void Conv2d_2c_3x3_Conv2D(__...				
83	{				
84	for (int ff = 0; ff < 192; ++ff)				
85	{				
86	for (int yy = 0; yy < 56; ++yy)				
87	{				
88	for (int xx = 0; xx < 56;...				
89	{				
90	compute[((((ff * 56) ...	1: __glob...	0: 0.0%	0: 0.1%	1: 2.1MB/s, 9.52%Efficiency
91	for (int rc = 0; rc <...				
92	{				
93	for (int ry = 0; ...				
94	{				
95	for (int rx =...				
96	{				
97	compute[(...	0: __glob...	0: 0.0%	0: 8.3%	0: 19.6MB/s, 100.00%Effic...
98	}				
99	}				
100	}				
101	compute[((((ff * 56) ...	(__global...	(0.0%)	(0.1%)	(2.1MB/s, 9.52%Efficiency)
102	}				
103	}				
104	}				
105	}				
106					
107	__kernel void MaxPool_3a_3x3_MaxPool(...				
108	{				
109	for (int ax1 = 0; ax1 < 192; ++ax1)				
110	{				
111	for (int ax2 = 0; ax2 < 28; +...				

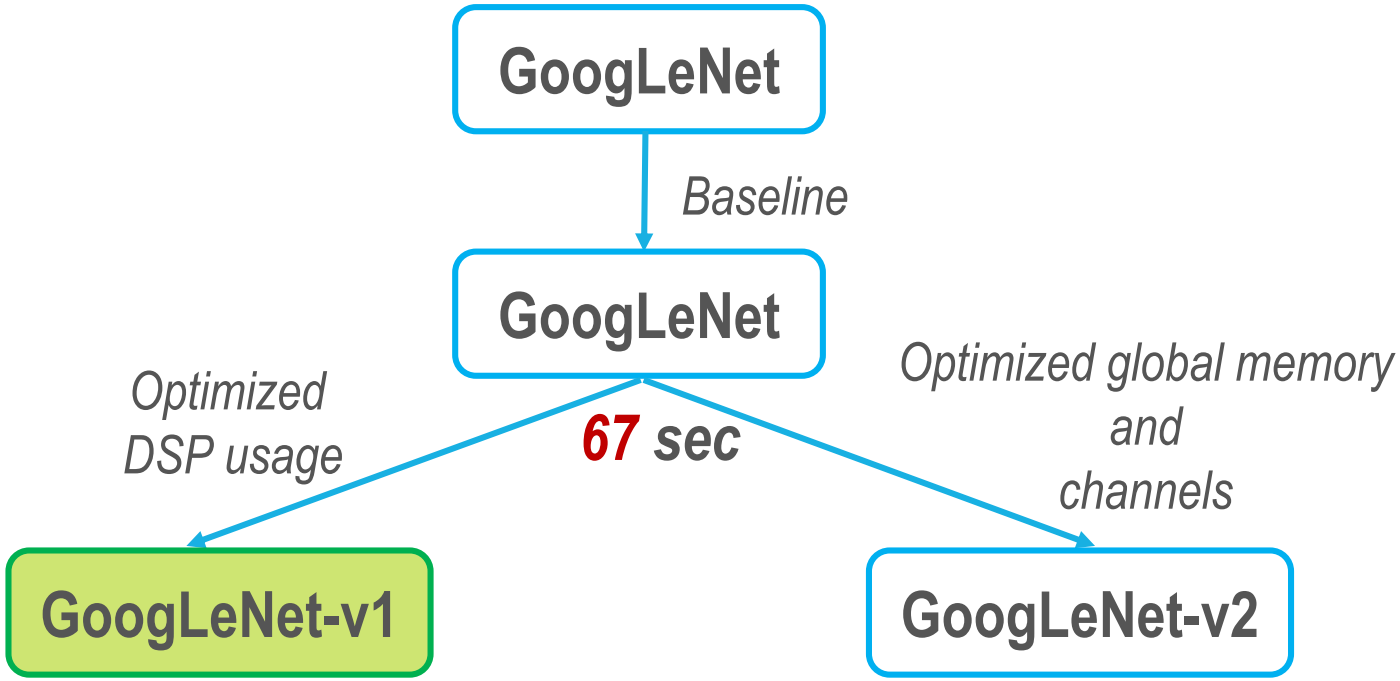
This convolution kernel (3x3) takes 18 sec

Write to global memory has only 9.52% efficiency

Computations have 8.3 % of occupancy

UNIVERSITÄT PADERBORN

GoogLeNet Optimization sequence



UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Inception-v1)

Initial report:

- Usage of DSP ~ 1% (the rest is required by BSP)

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	61%
ALUTs	32%
Dedicated logic registers	31%
Memory blocks	31%
DSP blocks	23%

Next step:

Apply optimizations:

- Loop Unrolling
- Loop Pipelining
- Shift Registers
- Using Local Memory

UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Inception-v1)

Intermediate report:

- Usage of DSP ~ 2% (the rest is required by BSP)
- Logic utilization is high

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	96%
ALUTs	39%
Dedicated logic registers	57%
Memory blocks	54%
DSP blocks	25%

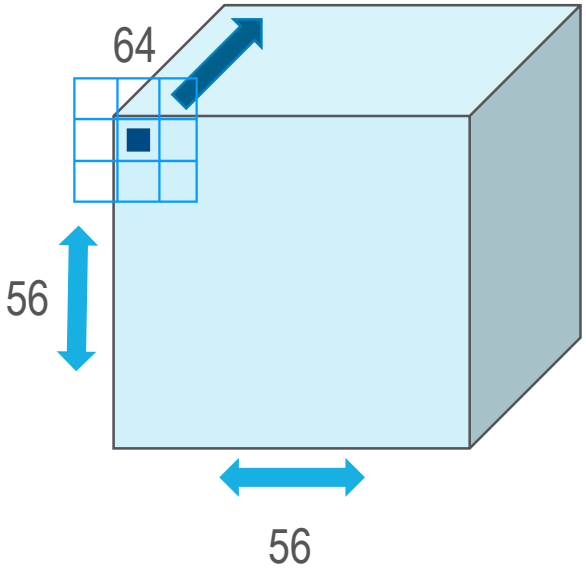
Next step:

Devise new logic of computations to increase DSP usage (change the loop order)

UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Inception-v1)

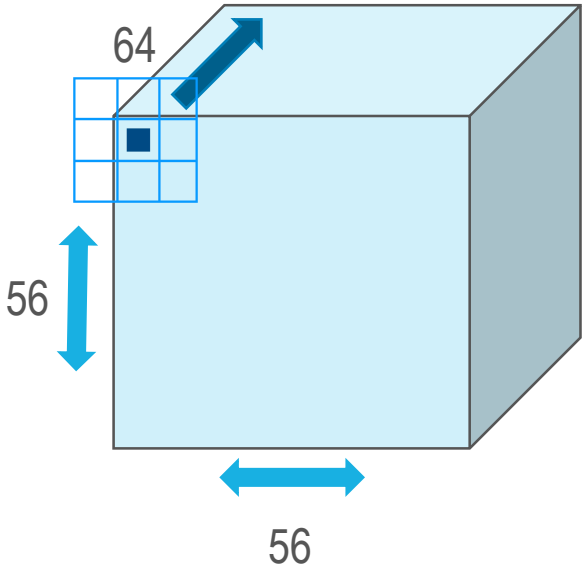
```
for (int ff = 0; ff < 192; ++ff)
{
    #pragma unroll 64
    for(int m = 0 ; m < 3*3*64 ; m++){
        local_weight[m] = input1[((ff * 3*3*64) + m)];
    }

    for (int yy = 0; yy < 56; ++yy)
    {
        #pragma unroll 2
        for (int xx = 0; xx < 56; ++xx)
        {
            float temp_0 = input2[ff];
            float temp_1 = 0.0;
            for (int rc = 0; rc < 64; ++rc)
            {
                float temp_2 = 0;
                #pragma unroll
                for (int ry = 0; ry < 3; ++ry)
                {
                    float temp_3 = 0;
                    #pragma unroll
                    for (int rx = 0; rx < 3; ++rx)
                    {
                        temp_3 += (l_input0[((((((rc * 58) + yy) + ry) * 58) + xx) + rx)] * local_weight[(((rc * 3) + ry) * 3 + rx]);
                    }
                    temp_2 +=temp_3;
                }
                temp_1 += temp_2;
            }
            temp_0 += temp_1;
            temp_0 = (temp_0 > 0) ? temp_0 : 0.000000e+00f;
            compute[((((ff * 56) + yy) * 56) + xx)] = temp_0;
        }
    }
}
```



UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Inception-v1)

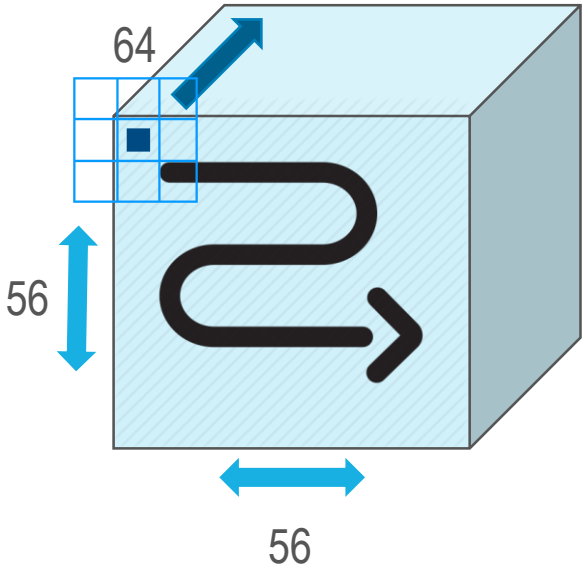
```
for (int ff = 0; ff < 192; ++ff)
{
    #pragma unroll 64
    for(int m = 0 ; m < 3*3*64 ; m++){
        local_weight[m] = input1[((ff * 3*3*64) + m)];
    }

    for (int yy = 0; yy < 56; ++yy)
    {
        #pragma unroll 2
        for (int xx = 0; xx < 56; ++xx)
        {
            float temp_0 = input2[ff];
            float temp_1 = 0.0;
            for (int rc = 0; rc < 64; ++rc)
            {
                float temp_2 = 0;
                #pragma unroll
                for (int ry = 0; ry < 3; ++ry)
                {
                    float temp_3 = 0;
                    #pragma unroll
                    for (int rx = 0; rx < 3; ++rx)
                    {
                        temp_3 += (l_input0[((((((rc * 58) + yy) + ry) * 58) + xx) + rx)] * local_weight[(((rc * 3) + ry) * 3 + rx]);
                    }
                    temp_2 +=temp_3;
                }
                temp_1 += temp_2;
            }
            temp_0 += temp_1;
            temp_0 = (temp_0 > 0) ? temp_0 : 0.000000e+00f;
            compute[(((ff * 56) + yy) * 56) + xx)] = temp_0;
        }
    }
}
```



UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Inception-v1)

```
for (int ff = 0; ff < 192; ++ff)
{
    #pragma unroll 64
    for(int m = 0 ; m < 3*3*64 ; m++){
        local_weight[m] = input1[((ff * 3*3*64) + m)];
    }

    for (int yy = 0; yy < 56; ++yy)
    {
        #pragma unroll 2
        for (int xx = 0; xx < 56; ++xx)
        {
            float temp_0 = input2[ff];
            float temp_1 = 0.0;
            for (int rc = 0; rc < 64; ++rc)
            {
                float temp_2 = 0;
                #pragma unroll
                for (int ry = 0; ry < 3; ++ry)
                {
                    float temp_3 = 0;
                    #pragma unroll
                    for (int rx = 0; rx < 3; ++rx)
                    {
                        temp_3 += (l_input0[((((((rc * 58) + yy) + ry) * 58) + xx) + rx)] * local_weight[(((rc * 3) + ry) * 3 + rx]);
                    }
                    temp_2 +=temp_3;
                }
                temp_1 += temp_2;
            }
            temp_0 += temp_1;
            temp_0 = (temp_0 > 0) ? temp_0 : 0.000000e+00f;
            compute[(((ff * 56) + yy) * 56) + xx)] = temp_0;
        }
    }
}
```



UNIVERSITÄT PADERBORN

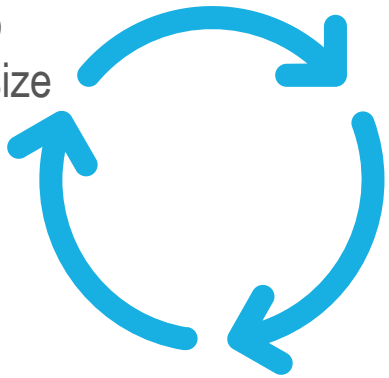
Performances / GoogLeNet (Inception-v1)

Intermediate report:

- Usage of DSP ~ 47% (the rest is required by BSP)
- Logic utilization is high

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	97%
ALUTs	51%
Dedicated logic registers	50%
Memory blocks	65%
DSP blocks	70%

Send to synthesize



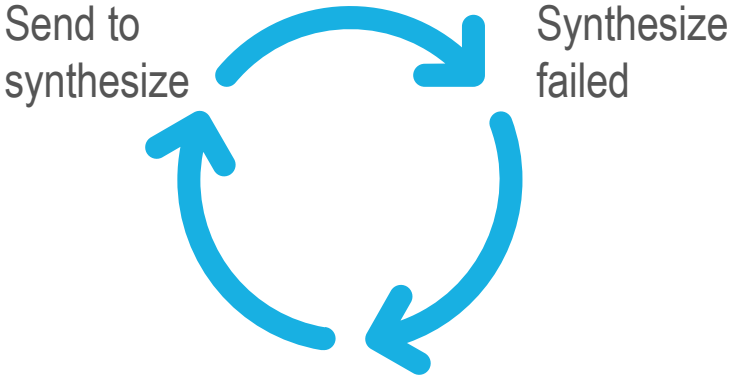
UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Inception-v1)

Intermediate report:

- Usage of DSP ~ 47% (the rest is required by BSP)
- Logic utilization is high

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	97%
ALUTs	51%
Dedicated logic registers	50%
Memory blocks	65%
DSP blocks	70%



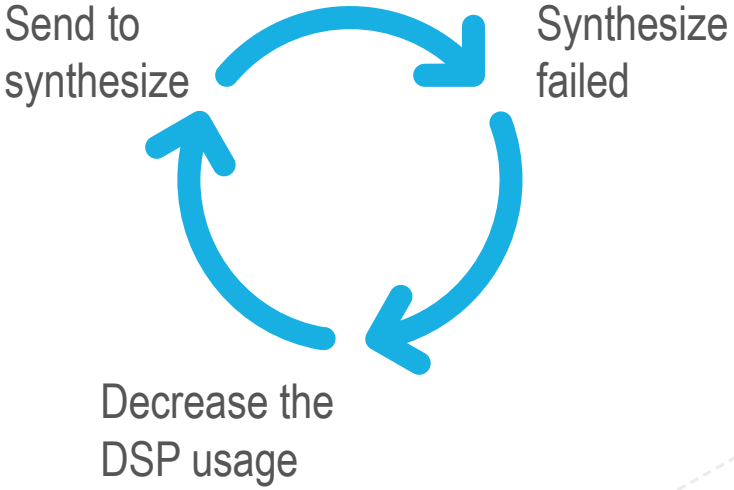
UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Inception-v1)

Intermediate report:

- Usage of DSP ~ 47% (the rest is required by BSP)
- Logic utilization is high

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	97%
ALUTs	51%
Dedicated logic registers	50%
Memory blocks	65%
DSP blocks	70%



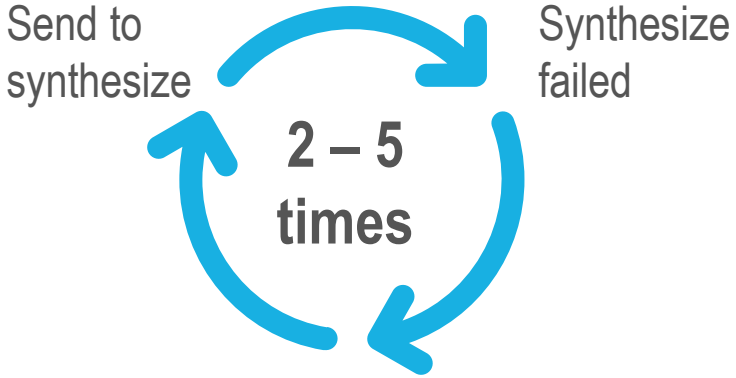
UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Inception-v1)

Intermediate report:

- Usage of DSP ~ 47% (the rest is required by BSP)
- Logic utilization is high

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	97%
ALUTs	51%
Dedicated logic registers	50%
Memory blocks	65%
DSP blocks	70%



UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Inception-v1)

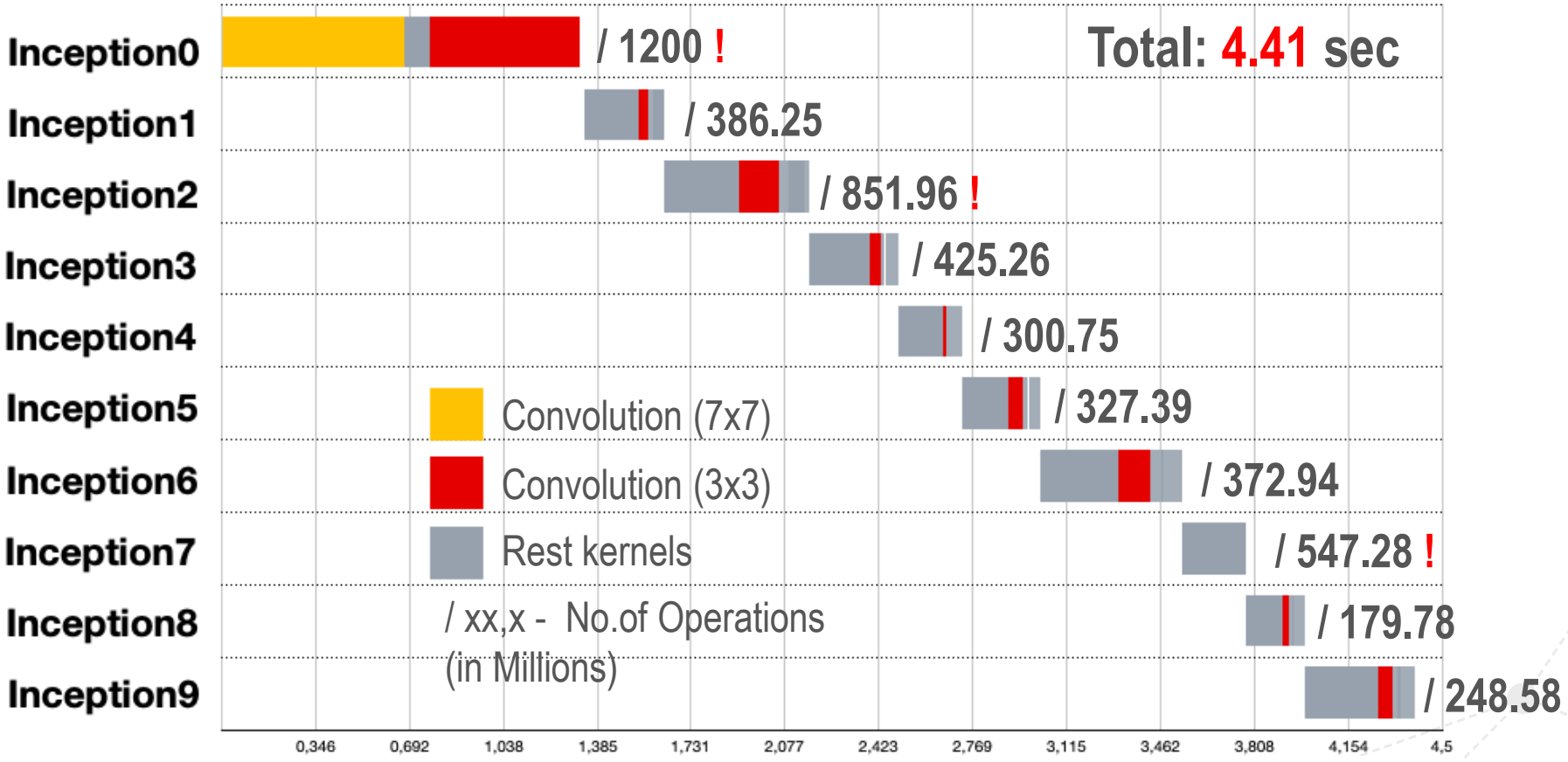
Final report:

- Usage of DSP ~ 7% (the rest is required by BSP)

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	65%
ALUTs	35%
Dedicated logic registers	33%
Memory blocks	71%
DSP blocks	30%

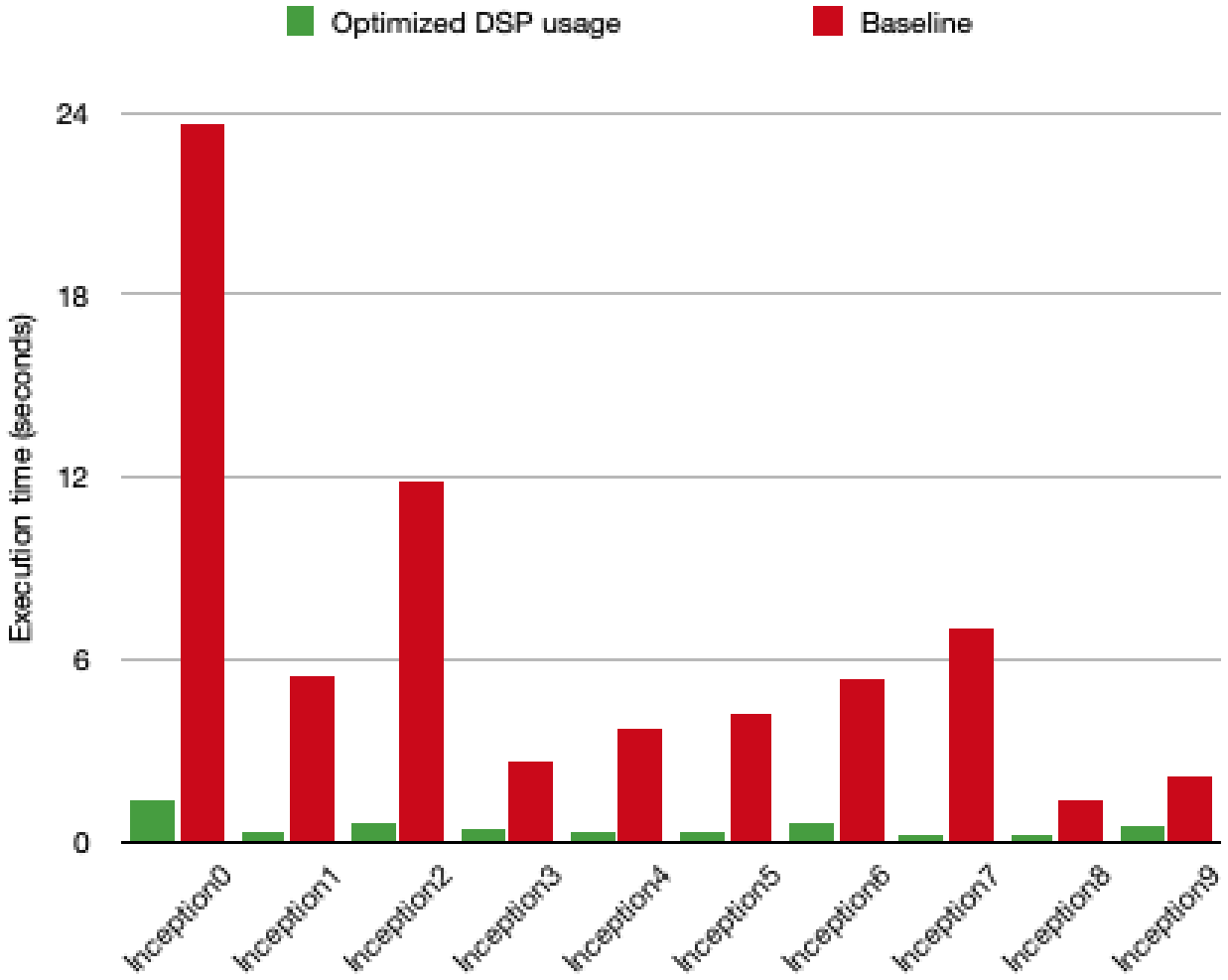
UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Inception-v1)



UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Inception-v1)



67 secs

4.41 secs

Can we do better ?

UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Inception-v1)

Stalling problem: reading one slice of data needs to be unrolled

Line #	Source: inception_5b.cl	Attributes	Stall%	Occupa...	Bandwi...
48	}				
49	for (int rc = 0; rc < 832; rc++)				
50	{				
51	for (int i = 0; i < 7*7; i++){				
52	l_input[i] = input0[7*7*rc+i];	0: __g...	0: 48.94%	0: 73.2%	0: 49...
53	}				
54					
55	#pragma unroll 2				
56	for (int yy = 0; yy < 7; ++yy)				
57	{				
58	#pragma unroll				
59	for (int xx = 0; xx < 7; ++xx)				
60	{				
61	temp_out[yy][xx] += (l_input[yy * 7 + xx] * input_weights...	0: __l...	0: 0.0%	0: 6.0%	0: --
62	}				
63					
64	}				
65	}				

UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Inception-v1)

Memory dependency: all computations have a low occupancy

Line #	Source: inception_5b.cl	Attributes	Stall%	Occupa...	Bandwi...
48	}				
49	for (int rc = 0; rc < 832; rc++)				
50	{				
51	for (int i = 0; i < 7*7; i++){				
52	l_input[i] = input0[7*7*rc+i];	0: __g...	0: 48.94%	0: 73.2%	0: 49...
53	}				
54					
55	#pragma unroll 2				
56	for (int yy = 0; yy < 7; ++yy)				
57	{				
58	#pragma unroll				
59	for (int xx = 0; xx < 7; ++xx)				
60	{				
61	temp_out[yy][xx] += (l_input[yy * 7 + xx] * input_weights...	0: __l...	0: 0.0%	0: 6.0%	0: --
62	}				
63					
64	}				
65	}				

UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Inception-v1)

Memory dependency: all computations have a low occupancy

Line #	Source: inception_5b.cl	Attributes	Stall%	Occupa...	Bandwi...
48	}				
49	for (int rc = 0; rc < 832; rc++)				
50	{				
51	for (int i = 0; i < 7*7; i++){				
52	l_input[i] = input0[7*7*rc+i];	0: __g...	0: 48.94%	0: 73.2%	0: 49...
53	}				
54					
55	#pragma unroll 2				
56	for (int yy = 0; yy < 7; ++yy)				
57	{				
58	#pragma unroll				
59	for (int xx = 0; xx < 7; ++xx)				
60	{				
61	temp_out[yy][xx] += (l_input[yy * 7 + xx] * input_weights...	0: __l...	0: 0.0%	0: 6.0%	0: --
62	}				
63					
64	}				
65	}				

Future task ?

UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Inception-v1)

Memory dependency: fully unroll the reading from global memory and nested loops with computations

Line #	Source: inception_5c_v2.cl	Attributes	Stall%	Occupa...	Bandwi...
19	`for (int l = 0; l < 7; l++ ){`				
20	`for (int j = 0; j < 7; j++){`				
21	`temp_out[l][j] = 0.0;`				
22	`}`				
23	`}`				
24	`for (int rc = 0; rc < 832; rc++)`				
25	`{`				
26	`#pragma unroll`				
27	`for (int i = 0; i < 7*7; i++){`				
28	`l_input[i] = input0[7*7*rc+i];`	(__glo...	(0.91%)	(96.3%)	(94.4...
29	`}`				
30					
31	`#pragma unroll`				
32	`for (int yy = 0; yy < 7; ++yy)`				
33	`{`				
34	`#pragma unroll`				
35	`for (int xx = 0; xx < 7; ++xx)`				
36	`{`				
37	`temp_out[yy][xx] += (l_input[ yy * 7 + xx] * input_weights...`	(__loc...	(0.0%)	(96.3%)	(--)
38	`}`				
39					
40					
41	`}`				
42	`}`				

UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Inception-v1)

Memory dependency: fully unroll the reading from global memory and nested loops with computations

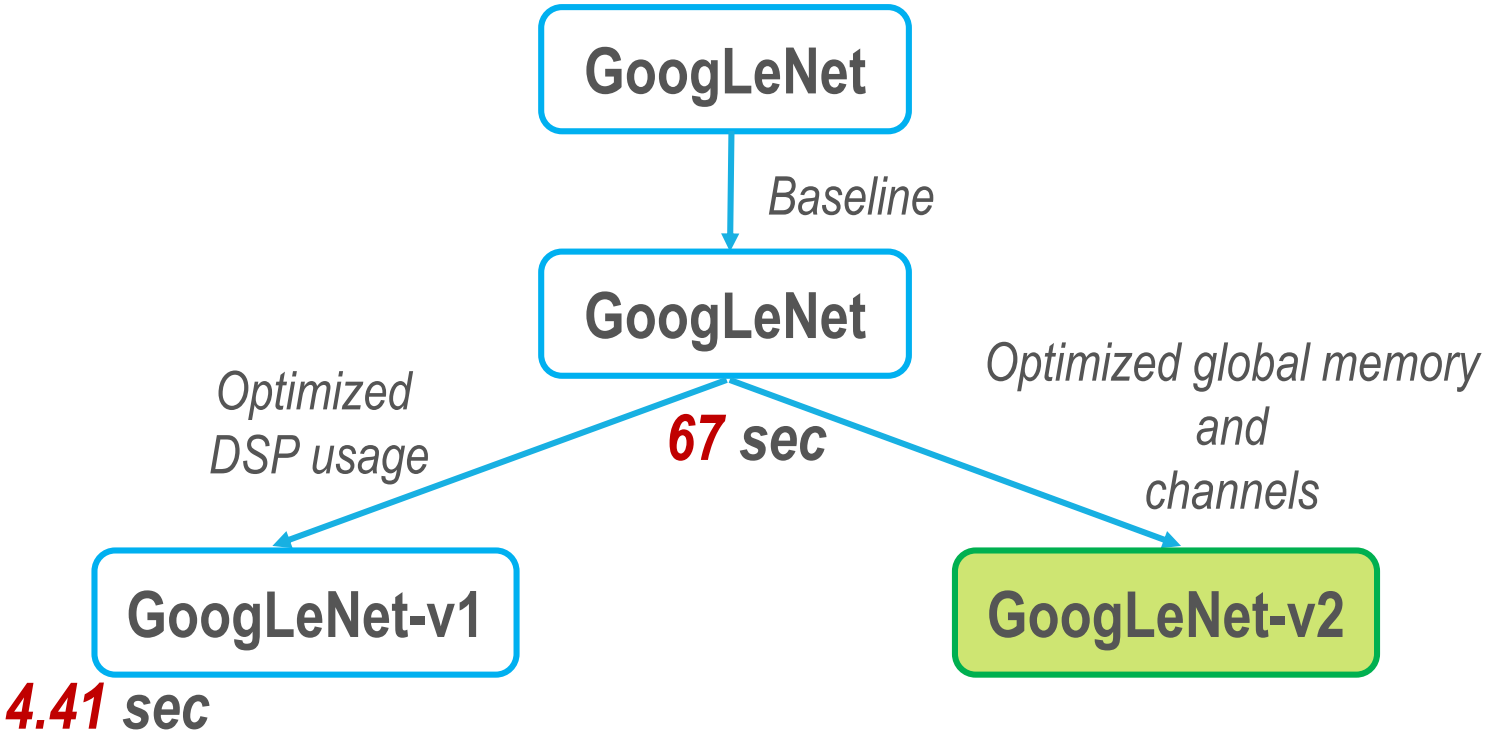
Line #	Source: inception_5c_v2.cl	Attributes	Stall%	Occupa...	Bandwi...
19	`for (int l = 0; l < 7; l++ ){`				
20	`for (int j = 0; j < 7; j++){`				
21	`temp_out[l][j] = 0.0;`				
22	`}`				
23	`}`				
24	`for (int rc = 0; rc < 832; rc++)`				
25	`{`				
26	`#pragma unroll`				
27	`for (int i = 0; i < 7*7; i++){`				
28	`l_input[i] = input0[7*7*rc+i];`	(__glo...	(0.91%)	(96.3%)	(94.4...
29	`}`				
30					
31	`#pragma unroll`				
32	`for (int yy = 0; yy < 7; ++yy)`				
33	`{`				
34	`#pragma unroll`				
35	`for (int xx = 0; xx < 7; ++xx)`				
36	`{`				
37	`temp_out[yy][xx] += (l_input[ yy * 7 + xx] * input_weights...`	(__loc...	(0.0%)	(96.3%)	(--)
38	`}`				
39					
40					
41	`}`				
42	`}`				

79.06 ms

1.75 ms

UNIVERSITÄT PADERBORN

GoogLeNet Optimization sequence



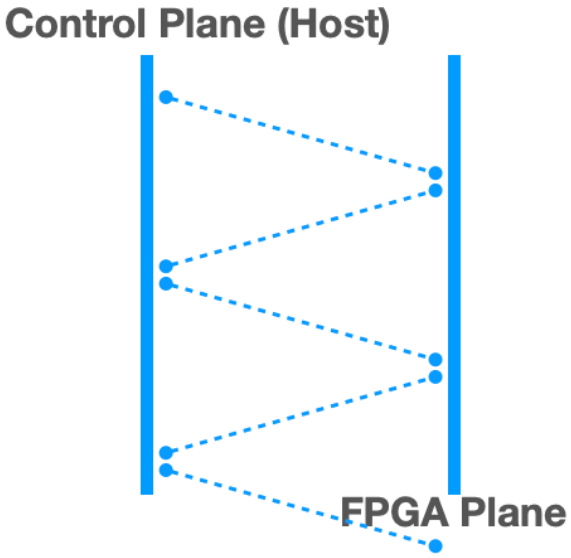
UNIVERSITÄT PADERBORN

Performances / GoogLeNet (with Channels)

Global memory

still slow, high latency, low throughput

Ping-pong model – data flow controlled by host



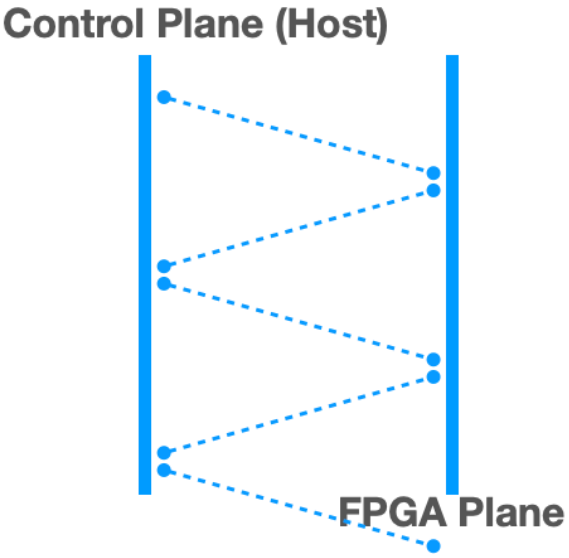
UNIVERSITÄT PADERBORN

Performances / GoogLeNet (with Channels)

Global memory

still slow, high latency, low throughput

Ping-pong model – data flow controlled by host



Big Idea: Make use of OpenCL Channels (internal and external) between kernels

- Reduce global memory data transfers
- Bulk of data transfers without involving the Host

UNIVERSITÄT PADERBORN

Performances / GoogLeNet (with Channels)

Earlier : Send data between nodes using global memory + MPI

Now : External (IO) Channels
Challenge **:** How to route data between nodes ? Node topology could change at any time.

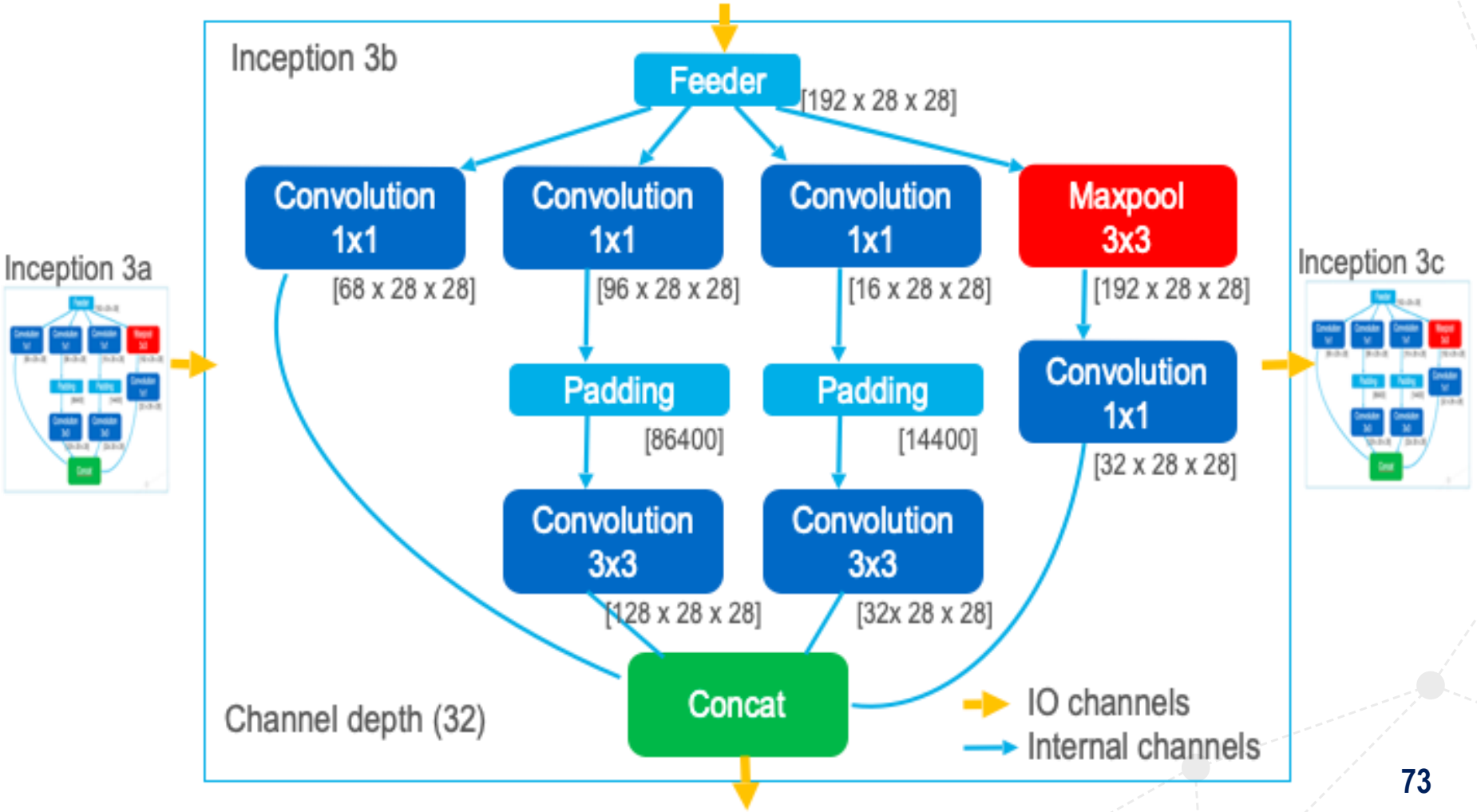
Big Idea

- Control the output port into which the last kernel in file writes to
- Control using parameters loaded from a config file -> route.xml
- Parsed during run time

Thus, our kernels are dynamic in nature – can be made to work on any node topology !

UNIVERSITÄT PADERBORN

Performances / GoogLeNet (with Channels)



UNIVERSITÄT PADERBORN

Performances / GoogLeNet (with Channels)

Solution ?

We hit a roadblock !
Channels were stalling !

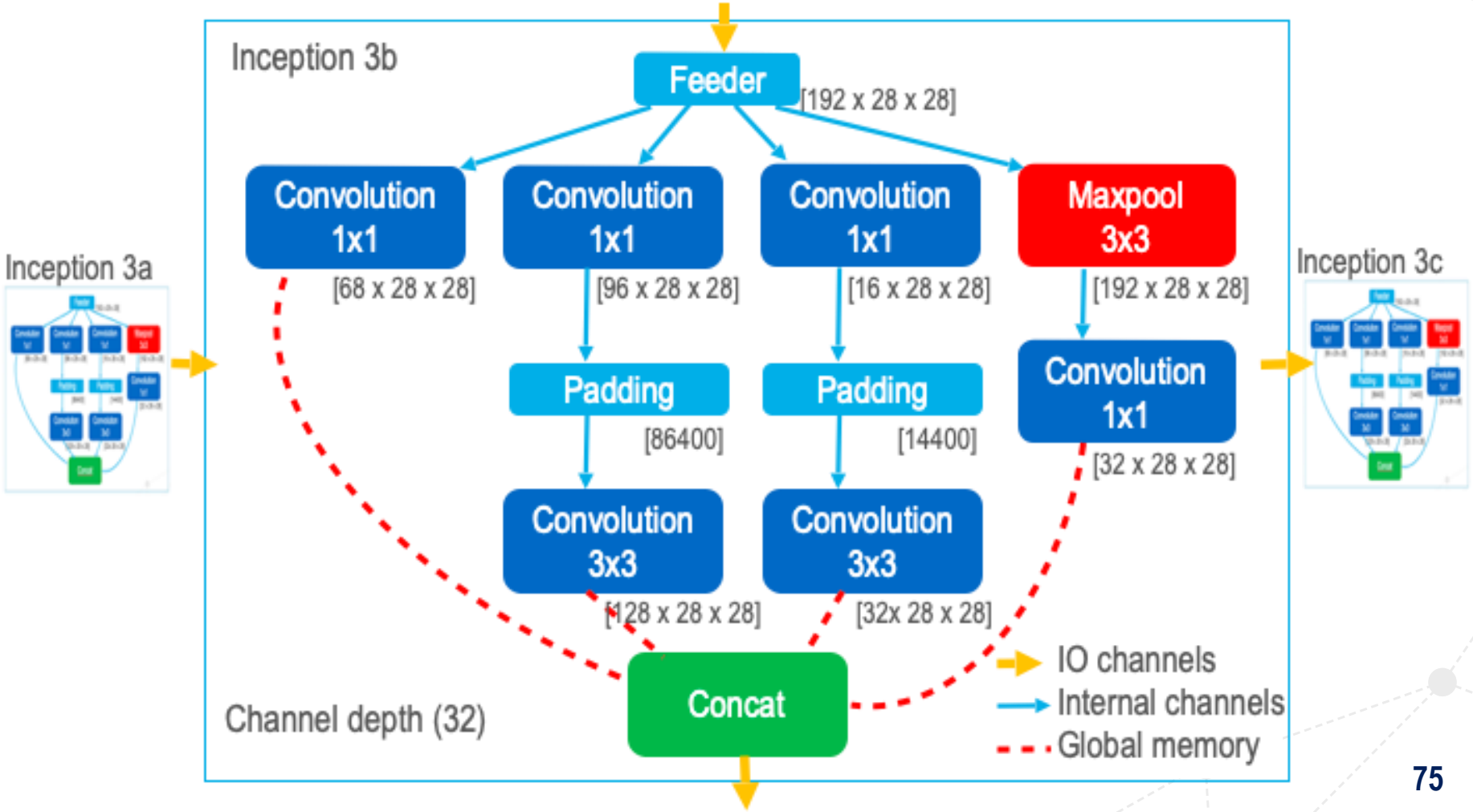
Debugging:

- Printf statements
- Error codes from OpenCL

Problem:
Channels between Concat kernels and data feeders of Concat were stalling

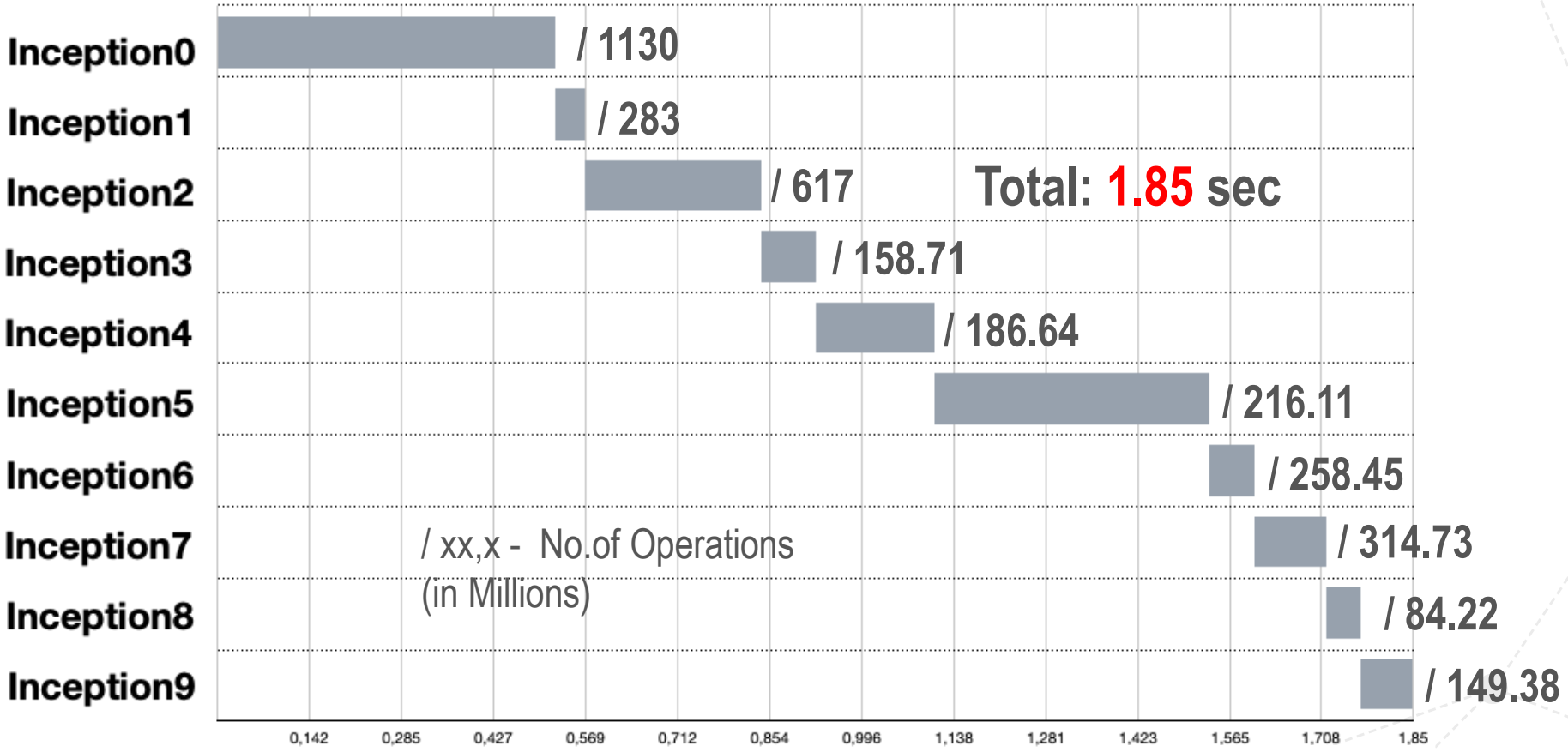
UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Hybrid Design)



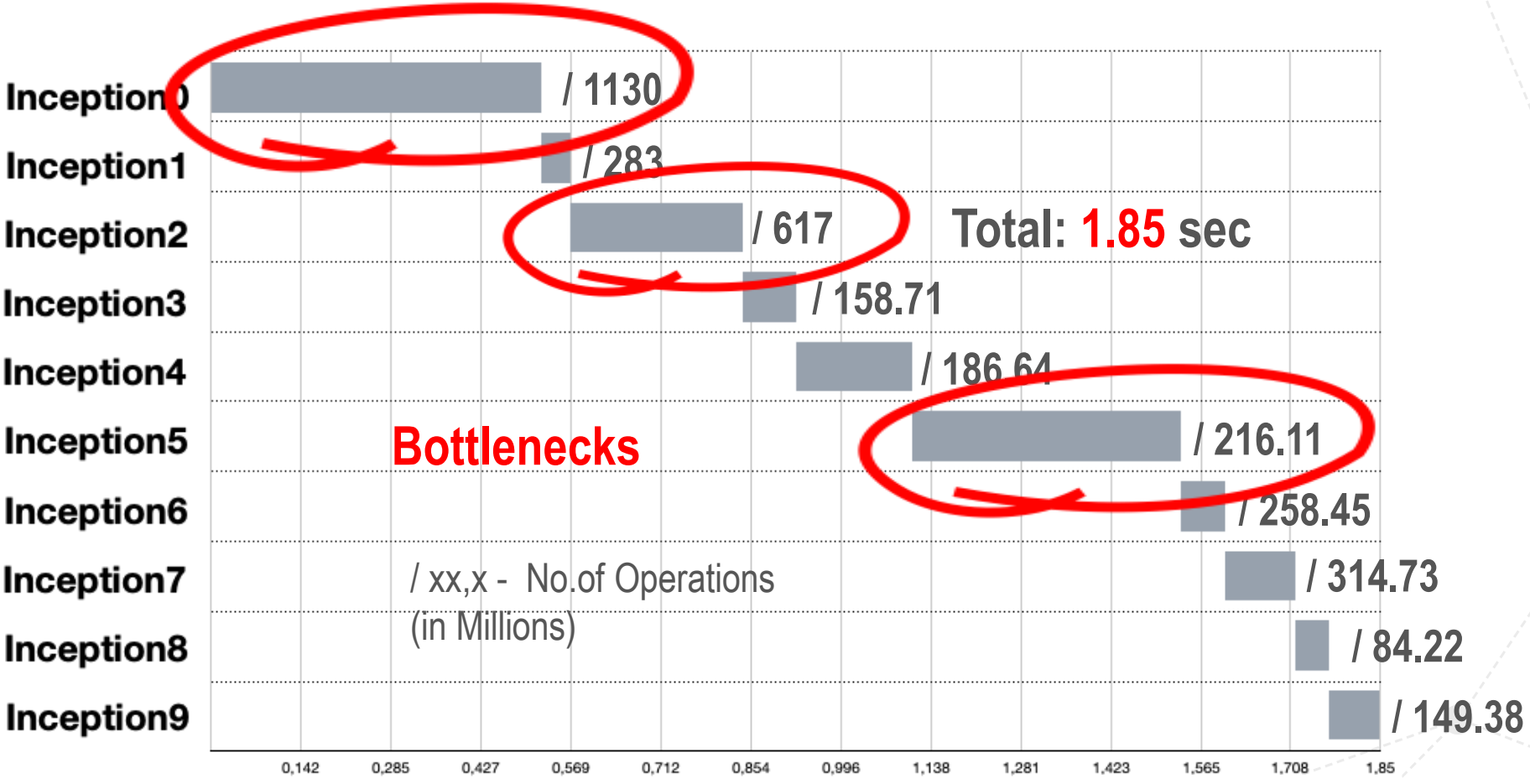
UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Hybrid Design)



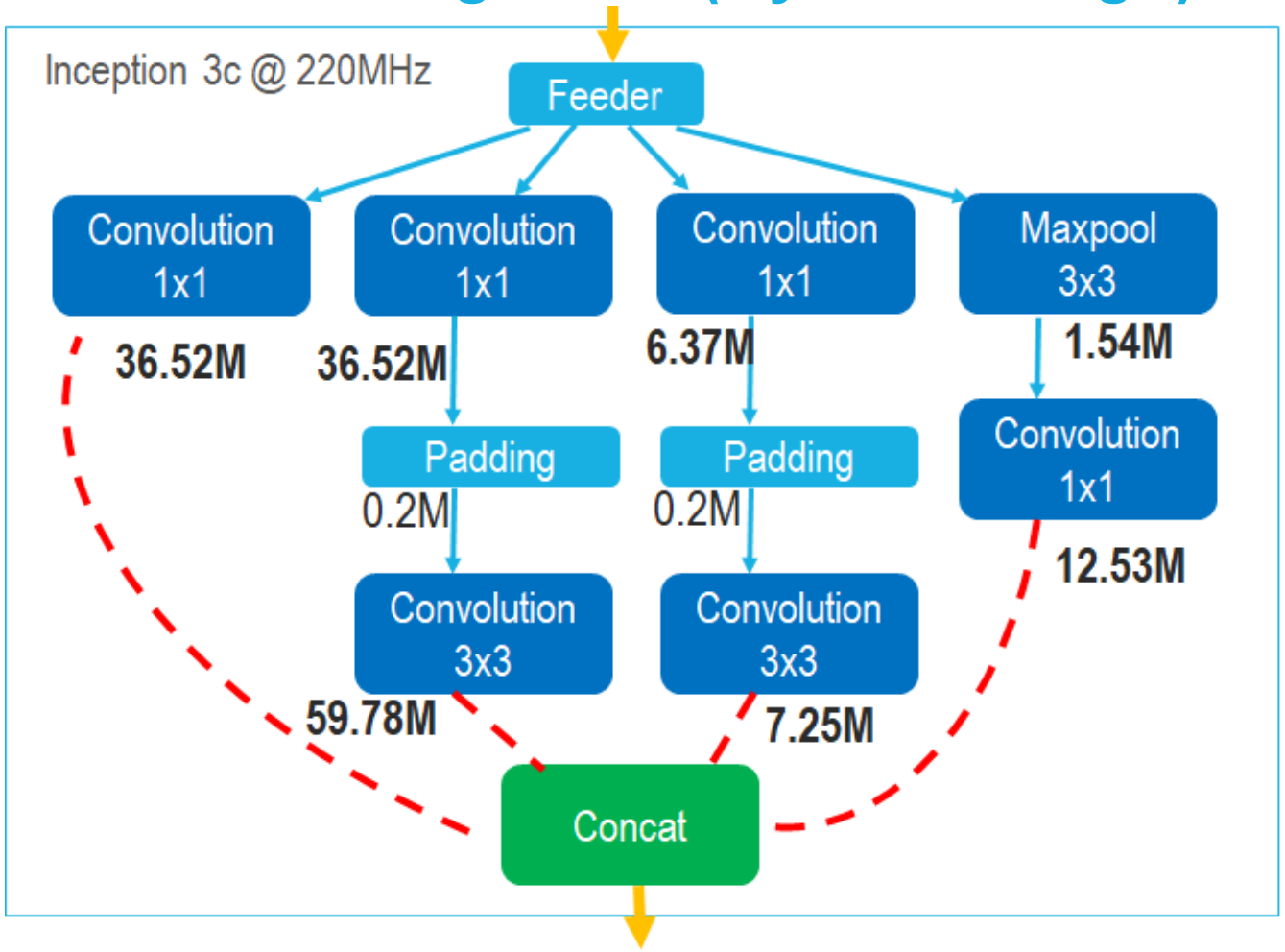
UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Hybrid Design)



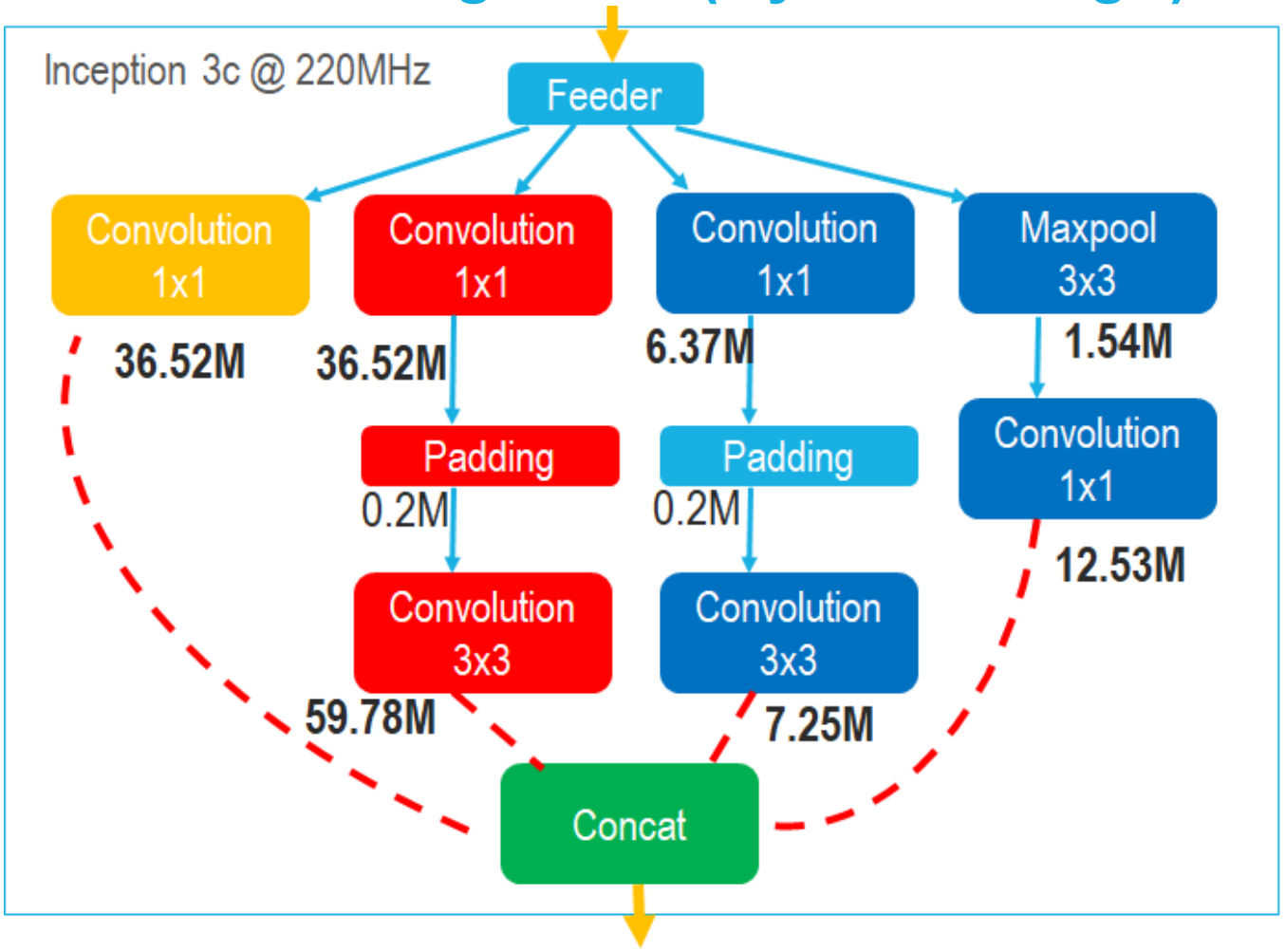
UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Hybrid Design)



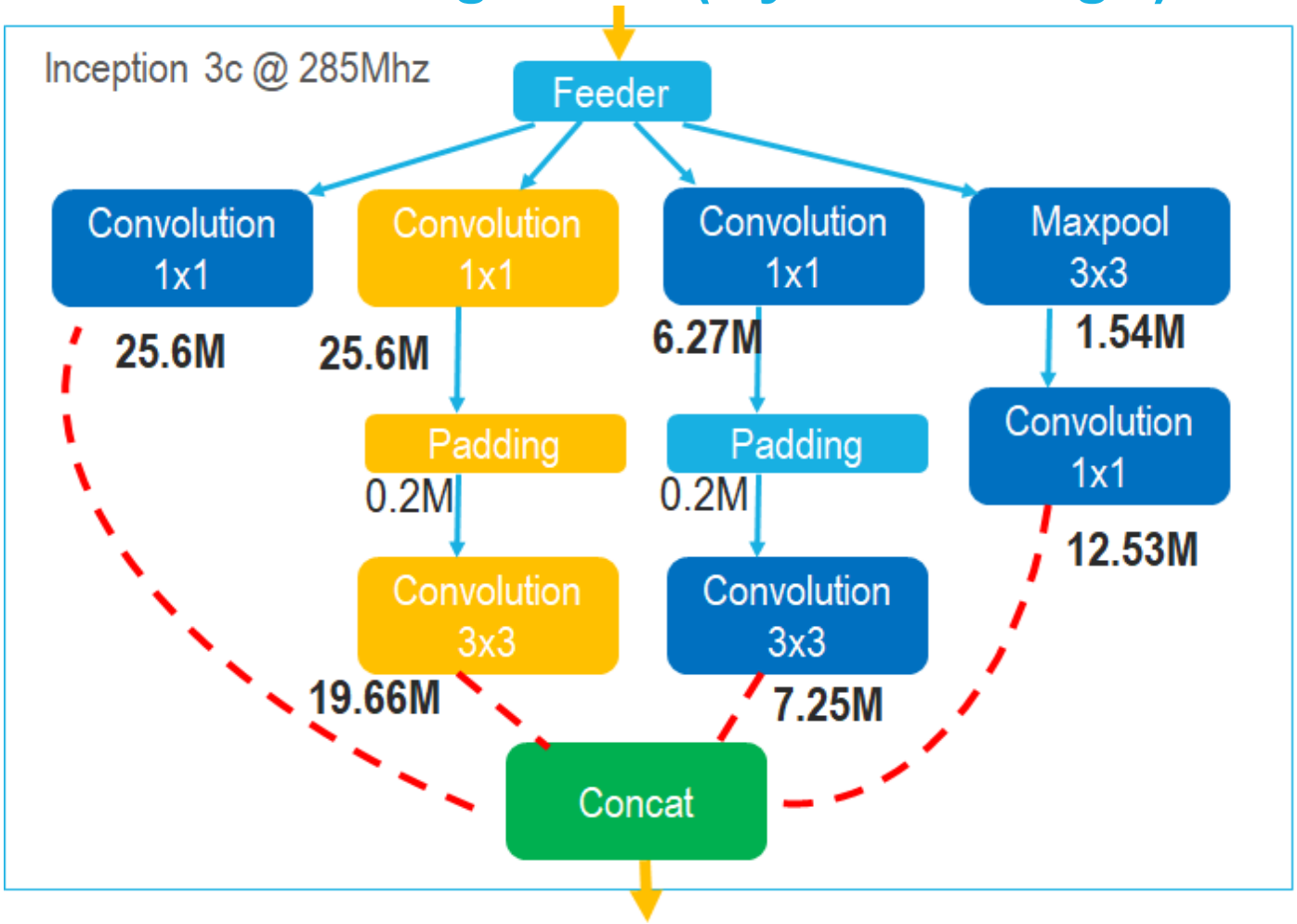
UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Hybrid Design)



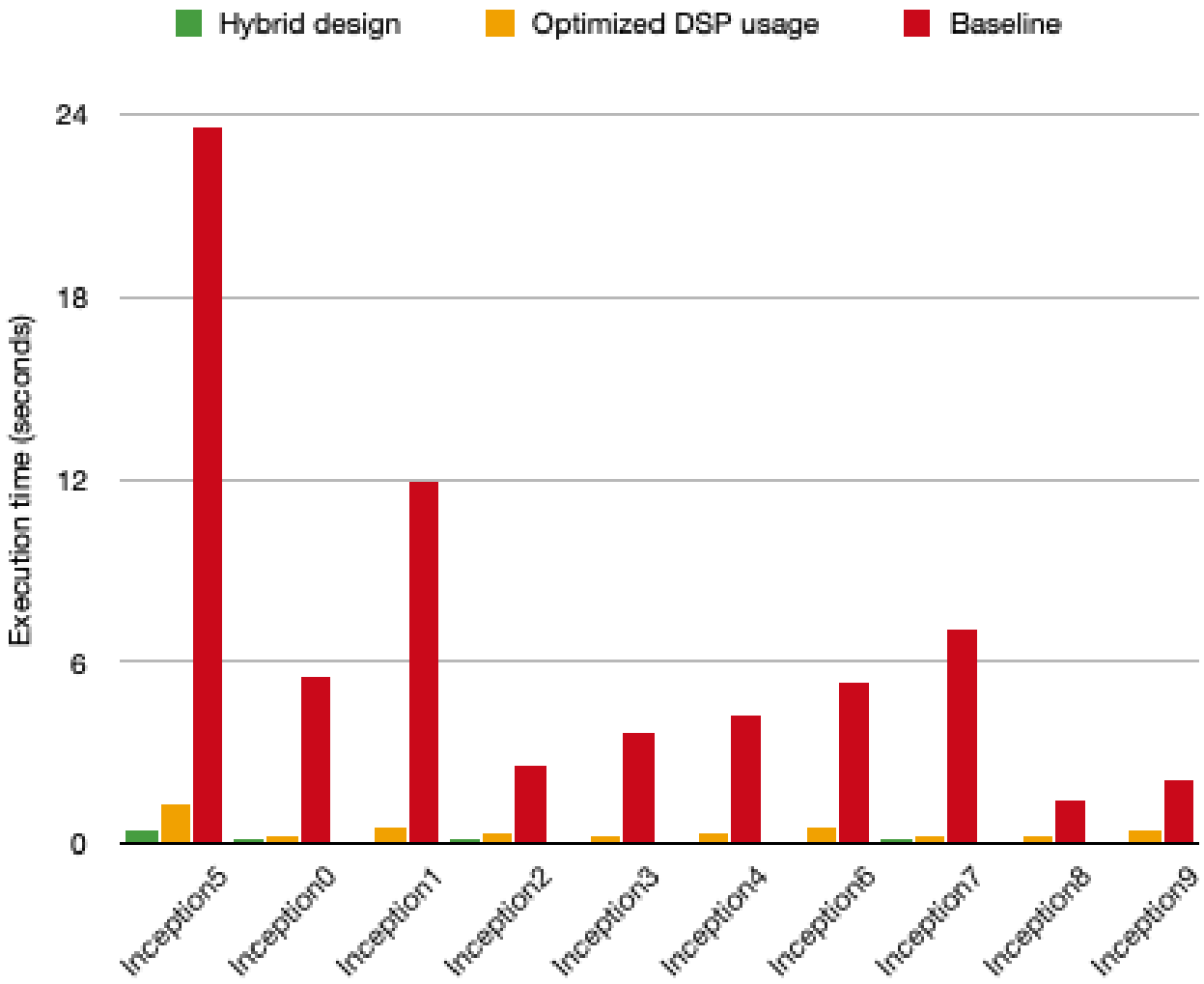
UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Hybrid Design)



UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Hybrid Design)



67 sec

⬇

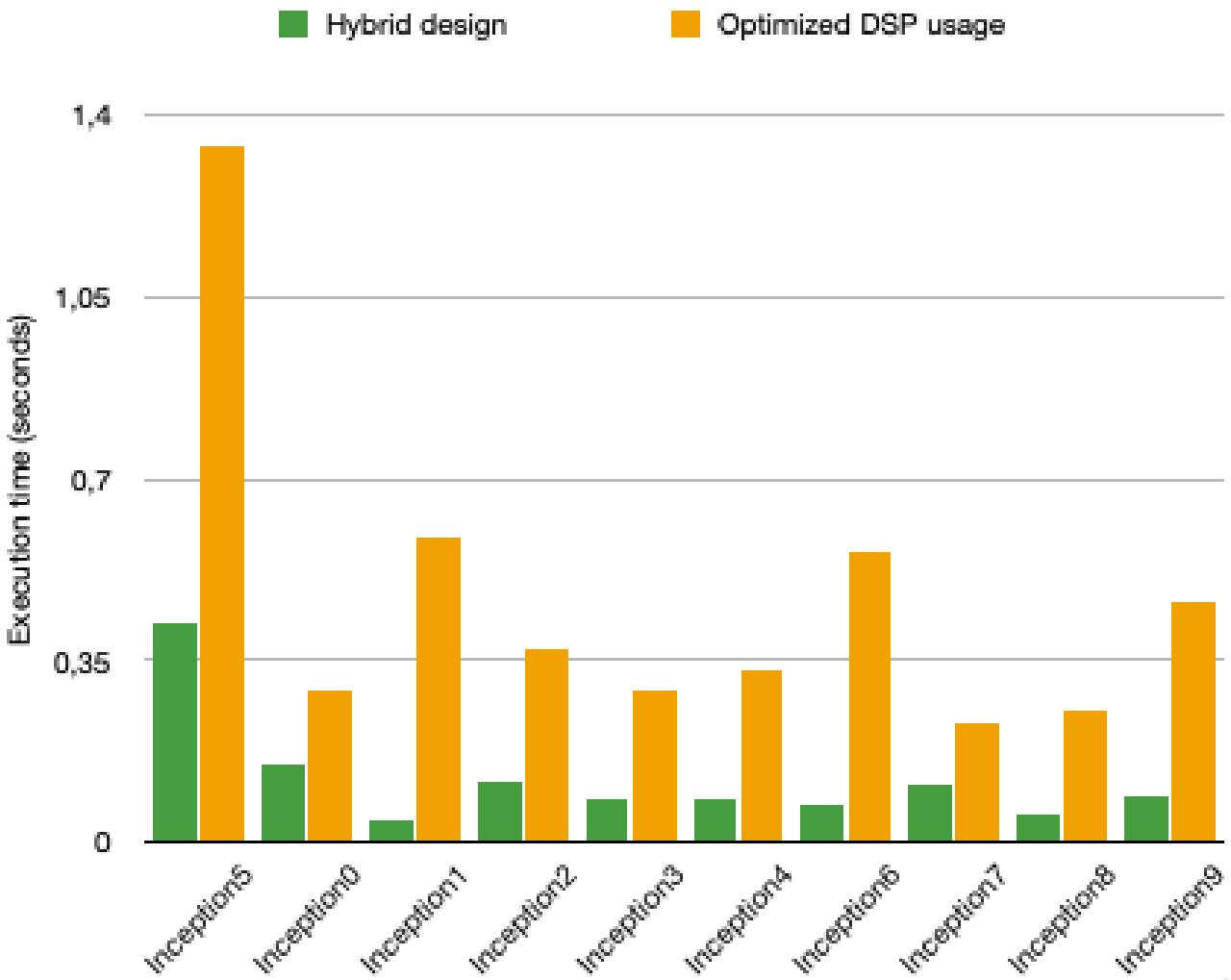
4.6 sec

⬇

1.18 sec

UNIVERSITÄT PADERBORN

Performances / GoogLeNet (Hybrid Design)

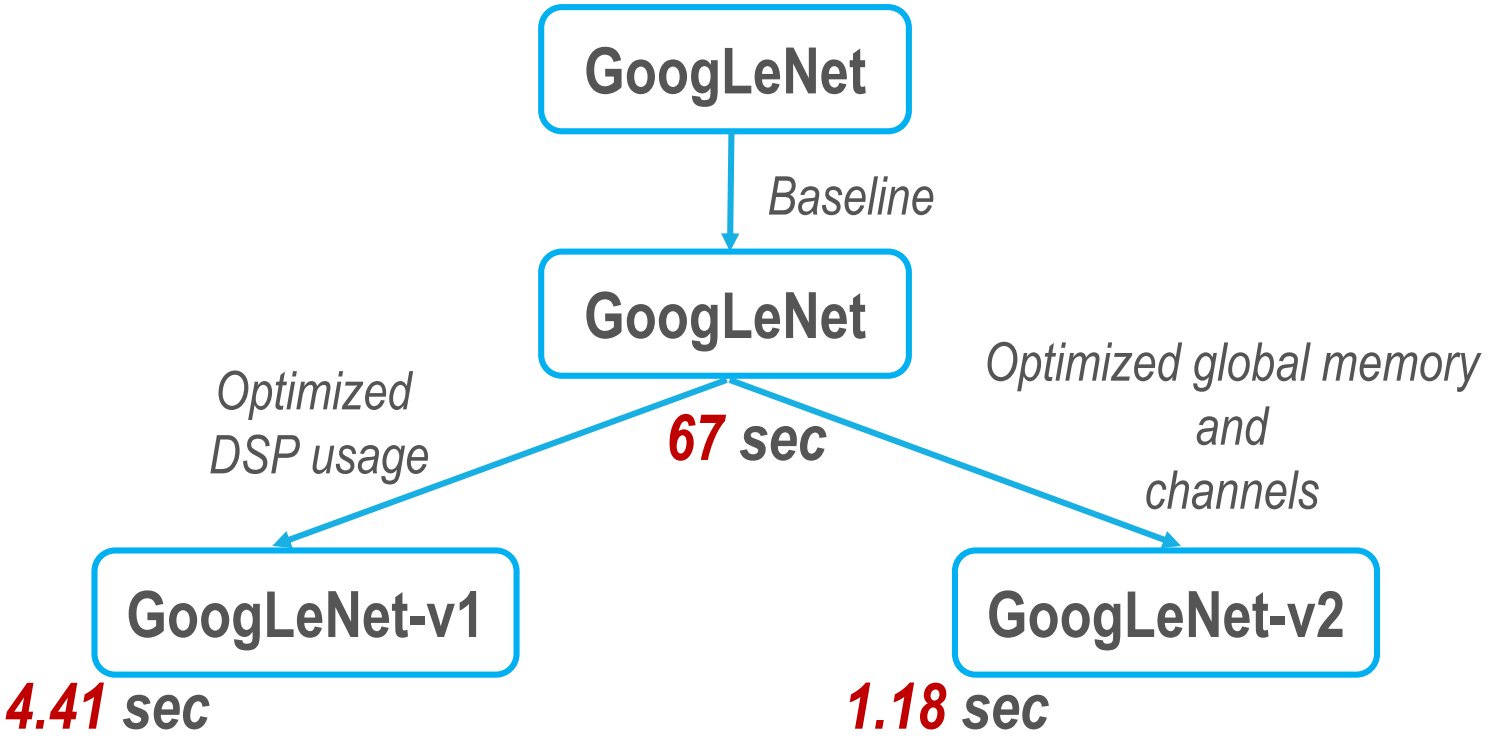


4.6 sec

1.18 sec

UNIVERSITÄT PADERBORN

GoogLeNet Optimization sequence

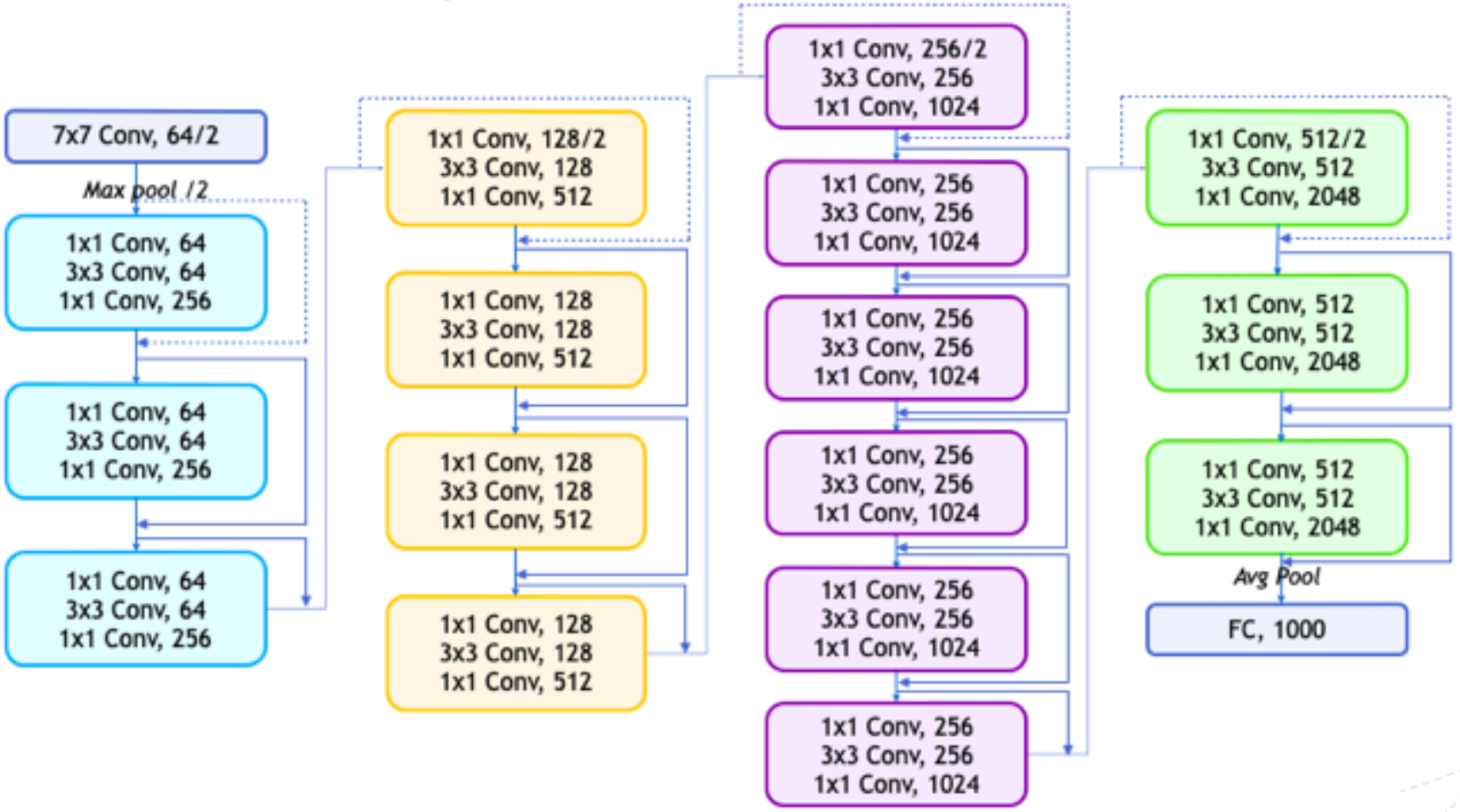


UNIVERSITÄT PADERBORN

RESNET - 50

UNIVERSITÄT PADERBORN

ResNet-50 Design



Kordík, P. 2019. Introducing Conveiro, an open source library for visualizing convolutional neural networks. https://tech.showmax.com/2019/04/conveiro/

UNIVERSITÄT PADERBORN

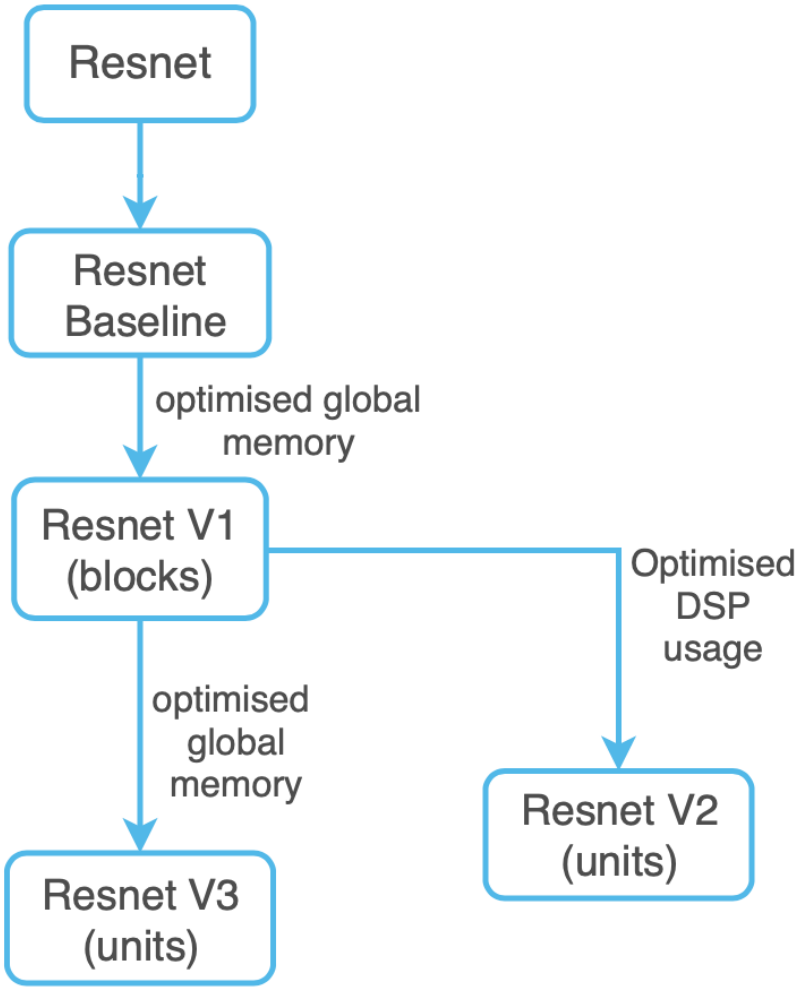
ResNet-50 Design

Whole network without optimizations requires:

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	178%
ALUTs	96%
Dedicated logic registers	88%
Memory blocks	148%
DSP blocks	24%

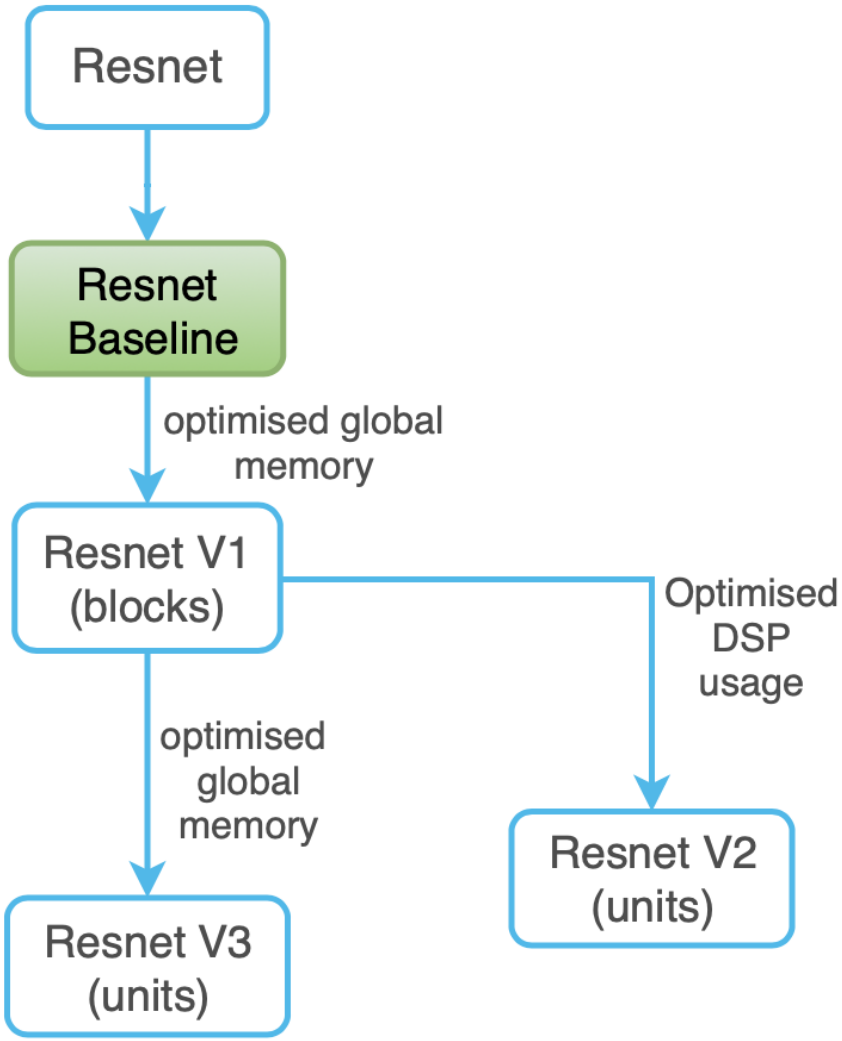
UNIVERSITÄT PADERBORN

ResNet Optimization sequence



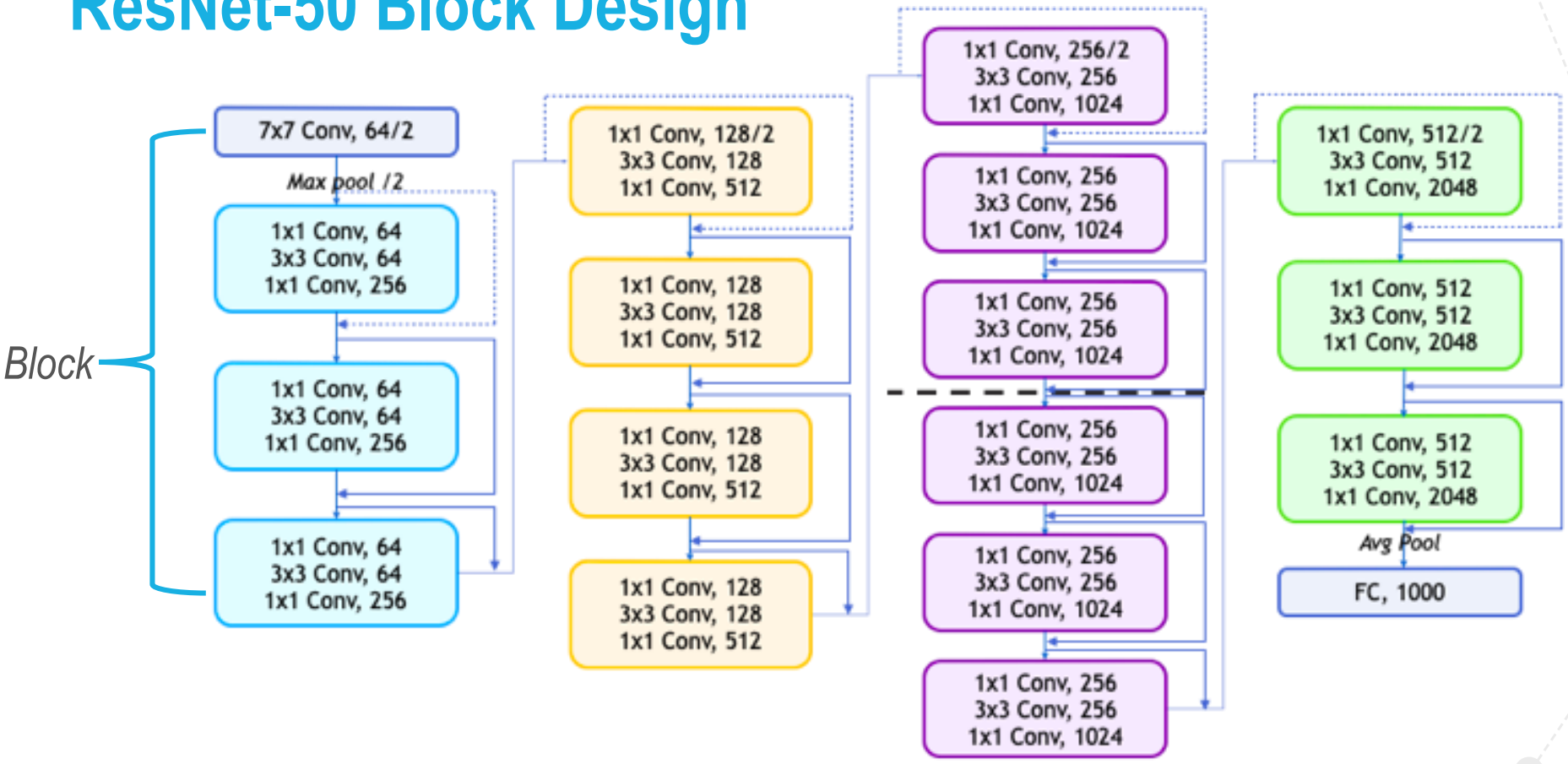
UNIVERSITÄT PADERBORN

ResNet Optimization sequence



UNIVERSITÄT PADERBORN

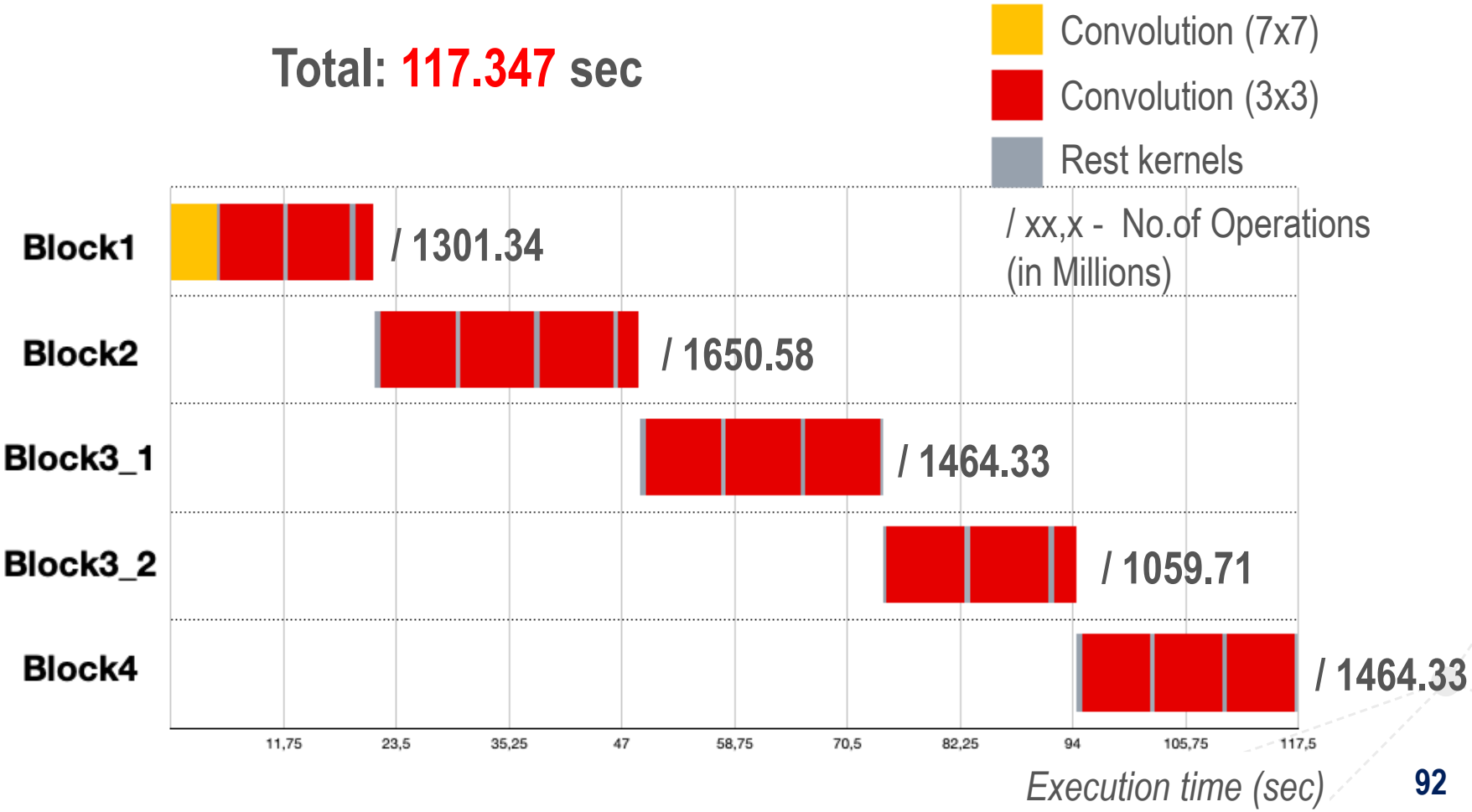
ResNet-50 Block Design



Kordík, P. 2019. Introducing Conveiro, an open source library for visualizing convolutional neural networks. https://tech.showmax.com/2019/04/conveiro/

UNIVERSITÄT PADERBORN

ResNet-50 Baseline Design



UNIVERSITÄT PADERBORN

ResNet-50 Baseline Design

Line #	Source: block1.cl	Attributes	Stall%	Occupancy%	Bandwidth
95					
96					
97	__kernel void block1_unit_1_b...				
98	for (int ff = 0; ff < 64; ++...				
99	for (int yy = 0; yy < 56; ...				
100	for (int xx = 0; xx < 56...				
101	compute[(((ff * 56) +...	0: __globa...	0: 0.0%	0: 0.1%	0: 0.1MB/s, 1...
102	for (int rc = 0; rc < ...				
103	for (int ry = 0; ry ...				
104	for (int rx = 0; r...				
105	compute[(((ff *...	0: __globa...	0: 0.0%	0: 8.3%	0: 18.8MB/s, ...
106	}				
107	}				
108	}				
109	compute[(((ff * 56) + yy) * 5...	(__global{...	(0.0%)	(0.1%)	(2.0MB/s, 10....
110	}				
111	}				
112	}				
113	}				
114					
115					

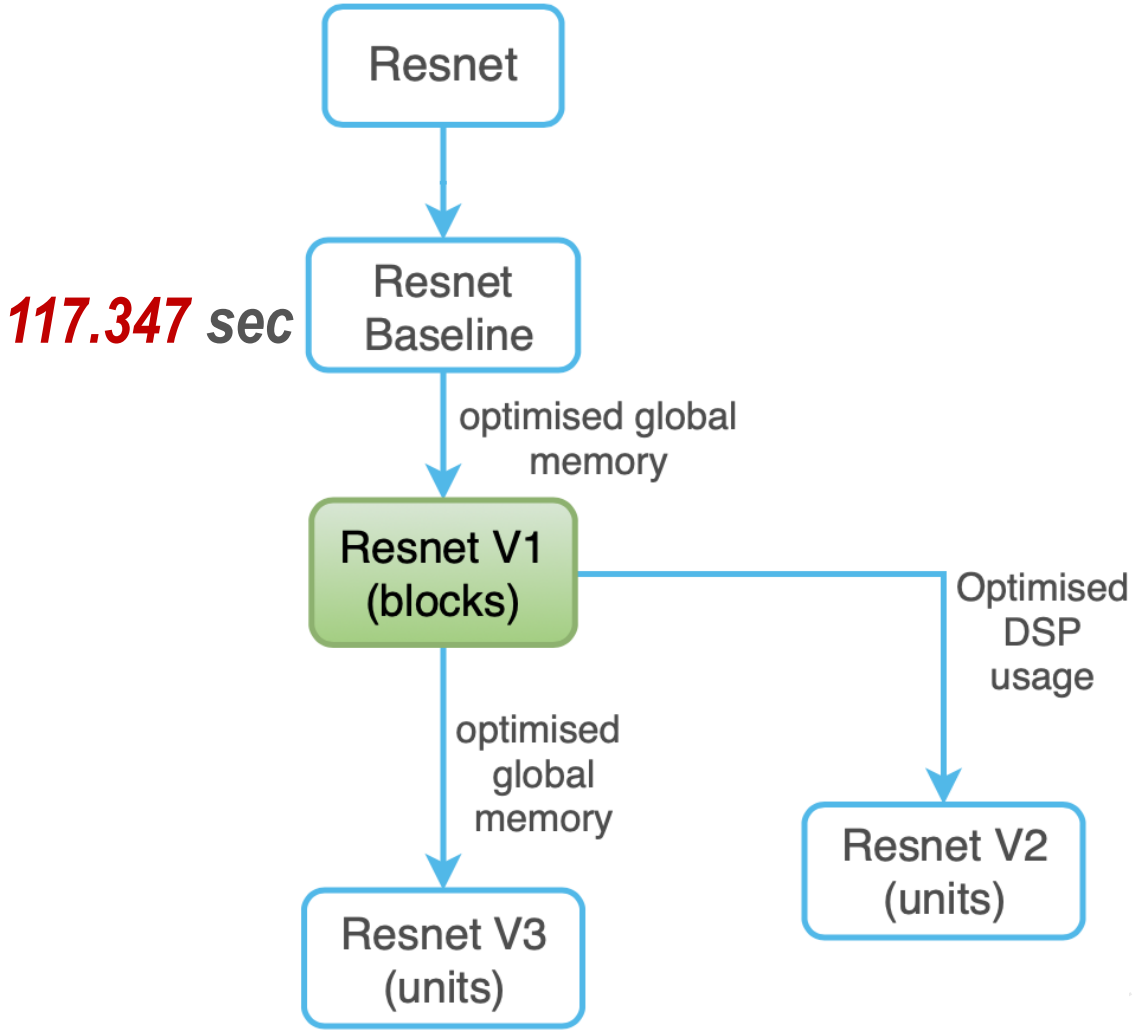
This convolution kernel (3x3) takes 6.58 sec

8.3% occupancy for Computations

10% efficiency
Write to global memory

UNIVERSITÄT PADERBORN

ResNet Optimization sequence



UNIVERSITÄT PADERBORN

ResNet-50 Baseline Design

Initial report:

- Usage of DSP ~ 1% (the rest is required by BSP)

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	78%
ALUTs	41%
Dedicated logic registers	39%
Memory blocks	47%
DSP blocks	23%

Next step:

Apply optimizations:

- Loop Unrolling
- Loop Pipelining
- Using Local Memory

UNIVERSITÄT PADERBORN

ResNet-50 Baseline Design

Final report:

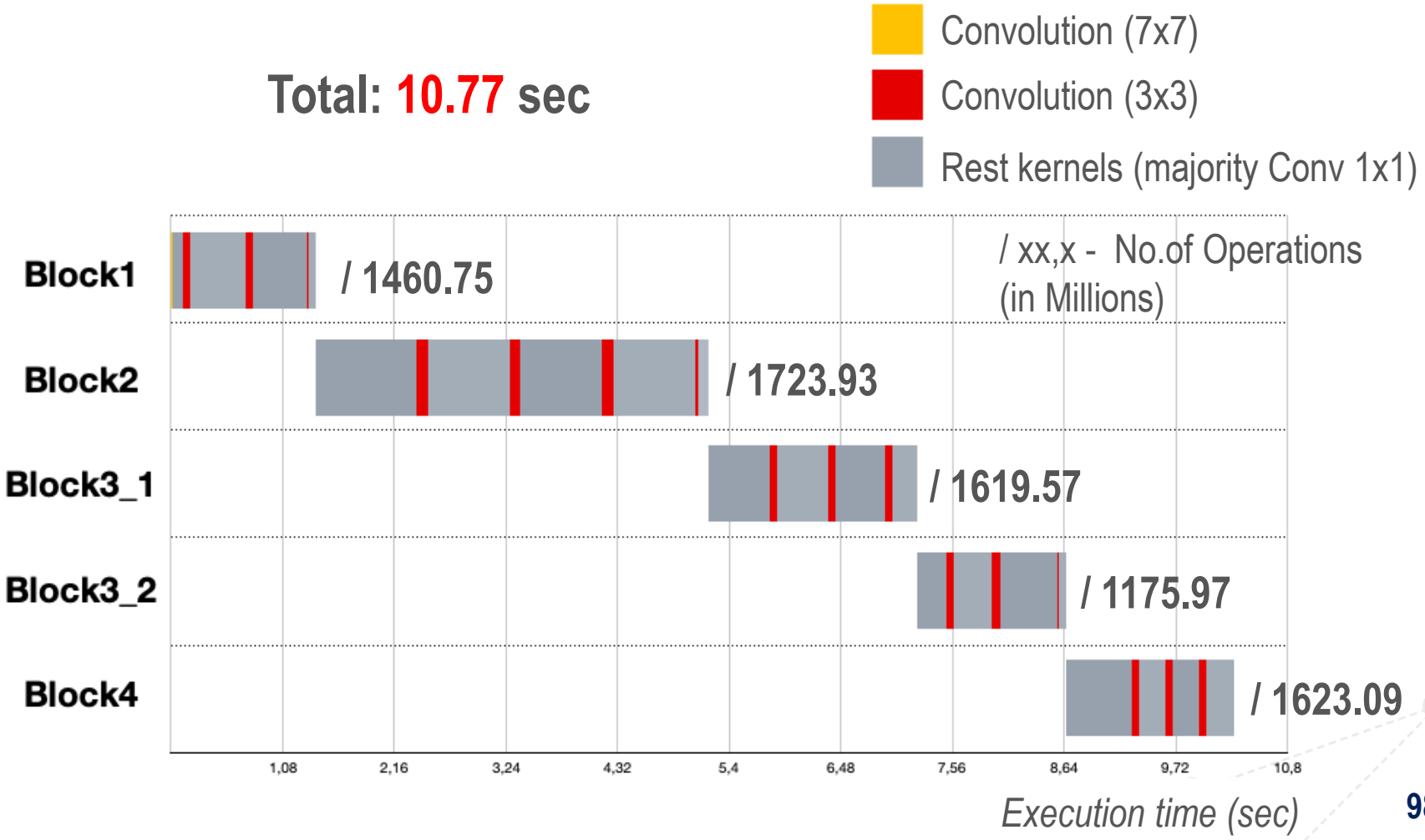
- Usage of DSP ~ 2% (the rest is required by BSP)

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	112%
ALUTs	52%
Dedicated logic registers	62%
Memory blocks	83%
DSP blocks	25%

The initial design does not allow to increase DSP usage by more than 2%

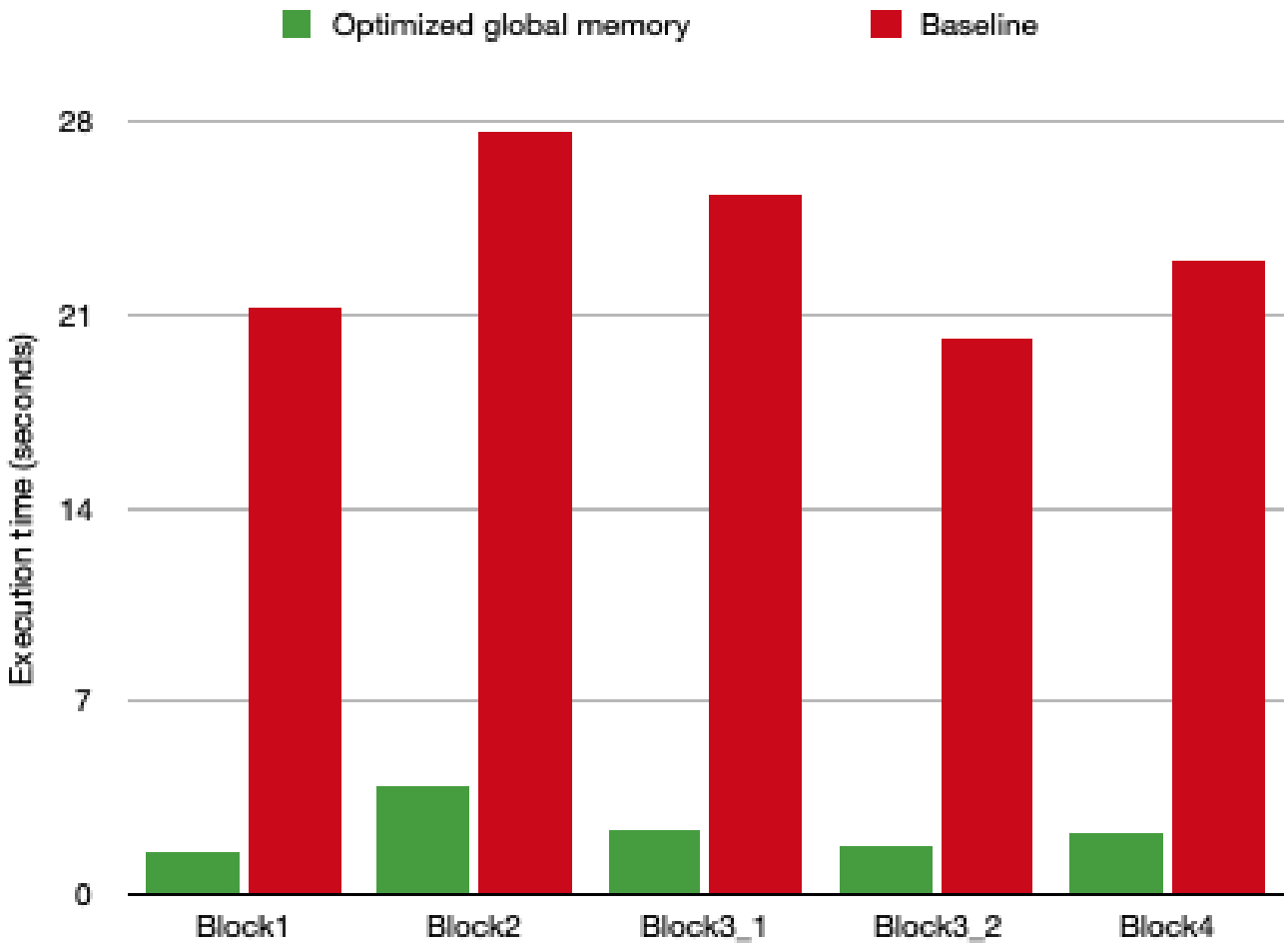
UNIVERSITÄT PADERBORN

Performances / ResNet (Opt-v1)



UNIVERSITÄT PADERBORN

Performances / ResNet (Opt-v1)



117.347 sec

10.77sec

UNIVERSITÄT PADERBORN

ResNet-50 Opt-v1

This convolution kernel (1x1) took 783.46 ms when 3x3 which have more number of operations took 110 ms

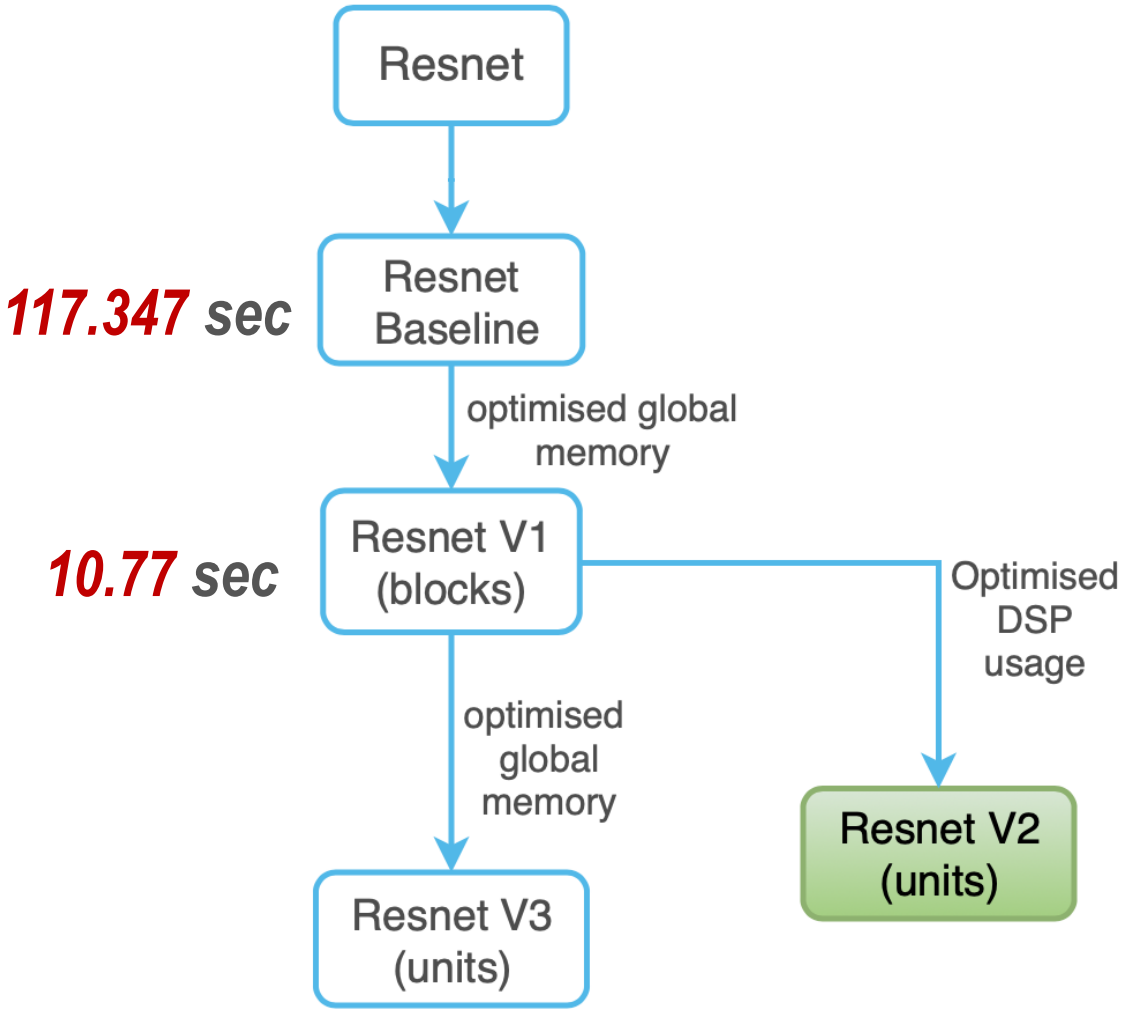
Line #	Source: block2.cl	Attribut...	Stall%	Occupa...	Bandwidth
7					
8	__kernel void block2_unit_1_bt_v2_shortcut_Conv2D(__g...				
9	__local float input_bias[512];				
10	#pragma unroll 64				
11	for (int b = 0; b < 512; ++b) {				
12	input_bias[b] = input2[b];	0: __...	0: 0.0%	0: 0.1%	0: 0.1MB/s, 100.0...
13	}				
14					
15	for (int ff = 0; ff < 512; ++ff) {				
16	float input_weights[256];				
17	#pragma unroll 32				
18	for (int w = 0; w < 256; ++w) {				
19	input_weights[w] = input1[((ff * 256) + w)];	0: __...	0: 0.0%	0: 0.1%	0: 0.7MB/s, 100.0...
20	}				
21	for (int yy = 0; yy < 28; ++yy) {				
22	for (int xx = 0; xx < 28; ++xx) {				
23	float temp_0 = input_bias[ff];	(__lo...	(0.0%)	(0.1%)	(--)
24	float temp_1 = 0.0;				
25	for (int rc = 0; rc < 256; ++rc) {				
26	temp_1 += (input0[((((rc * 28) + y...	0: __...	0: 0.0%	0: 99.7%	0: --
27	}				
28	temp_0 += temp_1;				
29	compute[((((ff * 28) + yy) * 28) + xx)...	(__gl...	(0.0%)	(0.4%)	(2.1MB/s, 100.00%...
30	}				
31	}				
32	}				
33	}				

Computations have 99.7% of occupancy

Write to global memory has 0.4% of occupancy

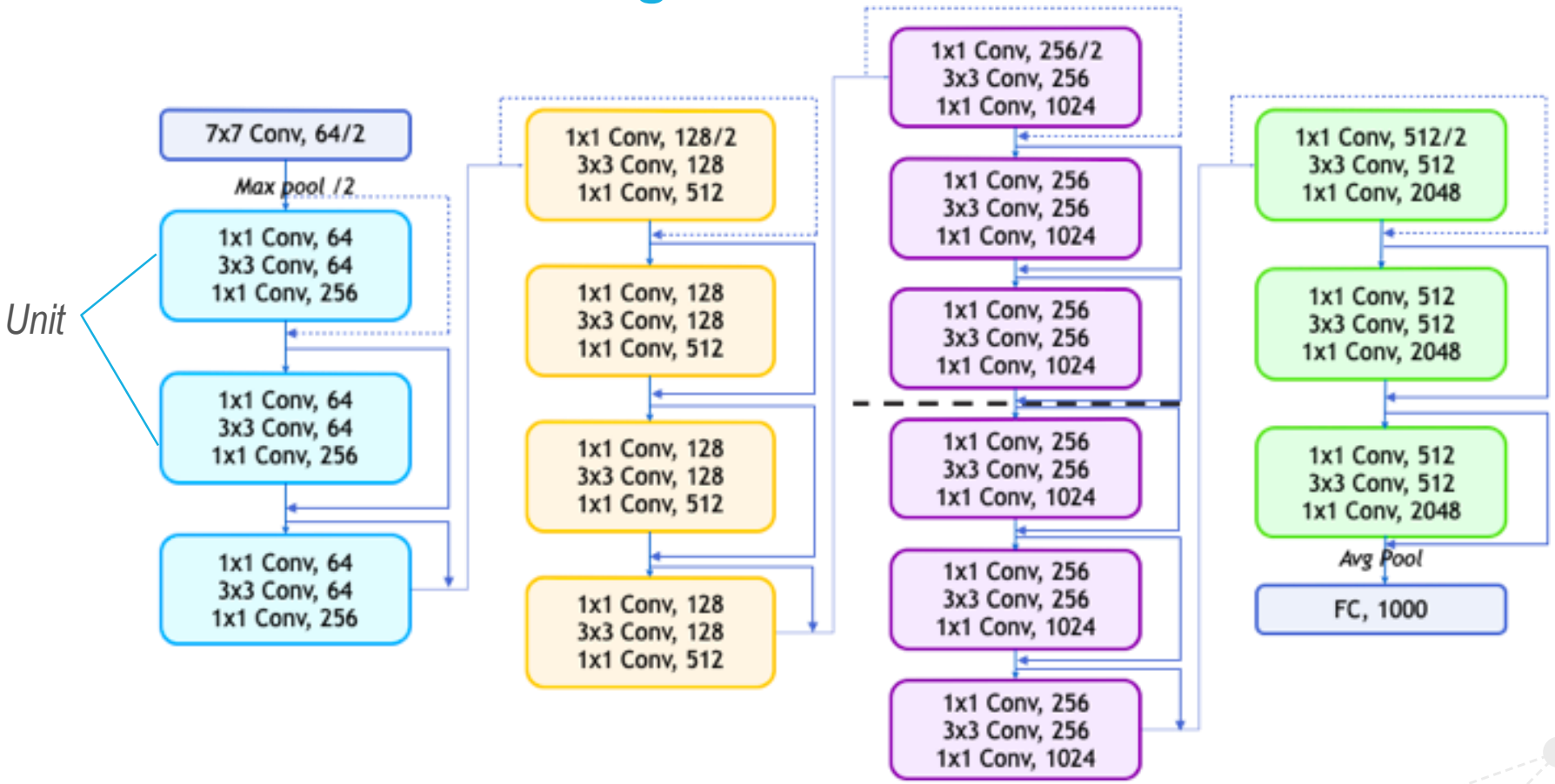
UNIVERSITÄT PADERBORN

ResNet Optimization sequence



UNIVERSITÄT PADERBORN

ResNet-50 Unit Design



Units : 1 - 3 4 - 7 8 –13 14 - 16

Kordík, P. 2019. Introducing Conveiro, an open source library for visualizing convolutional neural networks. https://tech.showmax.com/2019/04/conveiro/

UNIVERSITÄT PADERBORN

ResNet-50 Opt-v2

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	75%
ALUTs	40%
Dedicated logic registers	37%
Memory blocks	39%
DSP blocks	37%

~ 14%

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	69%
ALUTs	37%
Dedicated logic registers	35%
Memory blocks	52%
DSP blocks	50%

~ 27%

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	69%
ALUTs	35%
Dedicated logic registers	36%
Memory blocks	38%
DSP blocks	25%

~ 2%

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	64%
ALUTs	35%
Dedicated logic registers	32%
Memory blocks	38%
DSP blocks	30%

~ 7%

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	64%
ALUTs	35%
Dedicated logic registers	32%
Memory blocks	38%
DSP blocks	30%

~ 7%

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	73%
ALUTs	42%
Dedicated logic registers	35%
Memory blocks	51%
DSP blocks	30%

~ 7%

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	68%
ALUTs	38%
Dedicated logic registers	33%
Memory blocks	46%
DSP blocks	30%

~ 7%

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	71%
ALUTs	38%
Dedicated logic registers	36%
Memory blocks	41%
DSP blocks	49%

~ 26%

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	69%
ALUTs	38%
Dedicated logic registers	33%
Memory blocks	47%
DSP blocks	30%

~ 7%

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	68%
ALUTs	38%
Dedicated logic registers	33%
Memory blocks	46%
DSP blocks	30%

~ 7%

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	70%
ALUTs	36%
Dedicated logic registers	36%
Memory blocks	39%
DSP blocks	27%

~ 27%

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	83%
ALUTs	40%
Dedicated logic registers	45%
Memory blocks	40%
DSP blocks	37%

~ 14%

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	83%
ALUTs	45%
Dedicated logic registers	41%
Memory blocks	46%
DSP blocks	29%

~ 4%

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	69%
ALUTs	37%
Dedicated logic registers	34%
Memory blocks	39%
DSP blocks	35%

~ 12%

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	68%
ALUTs	38%
Dedicated logic registers	33%
Memory blocks	46%
DSP blocks	30%

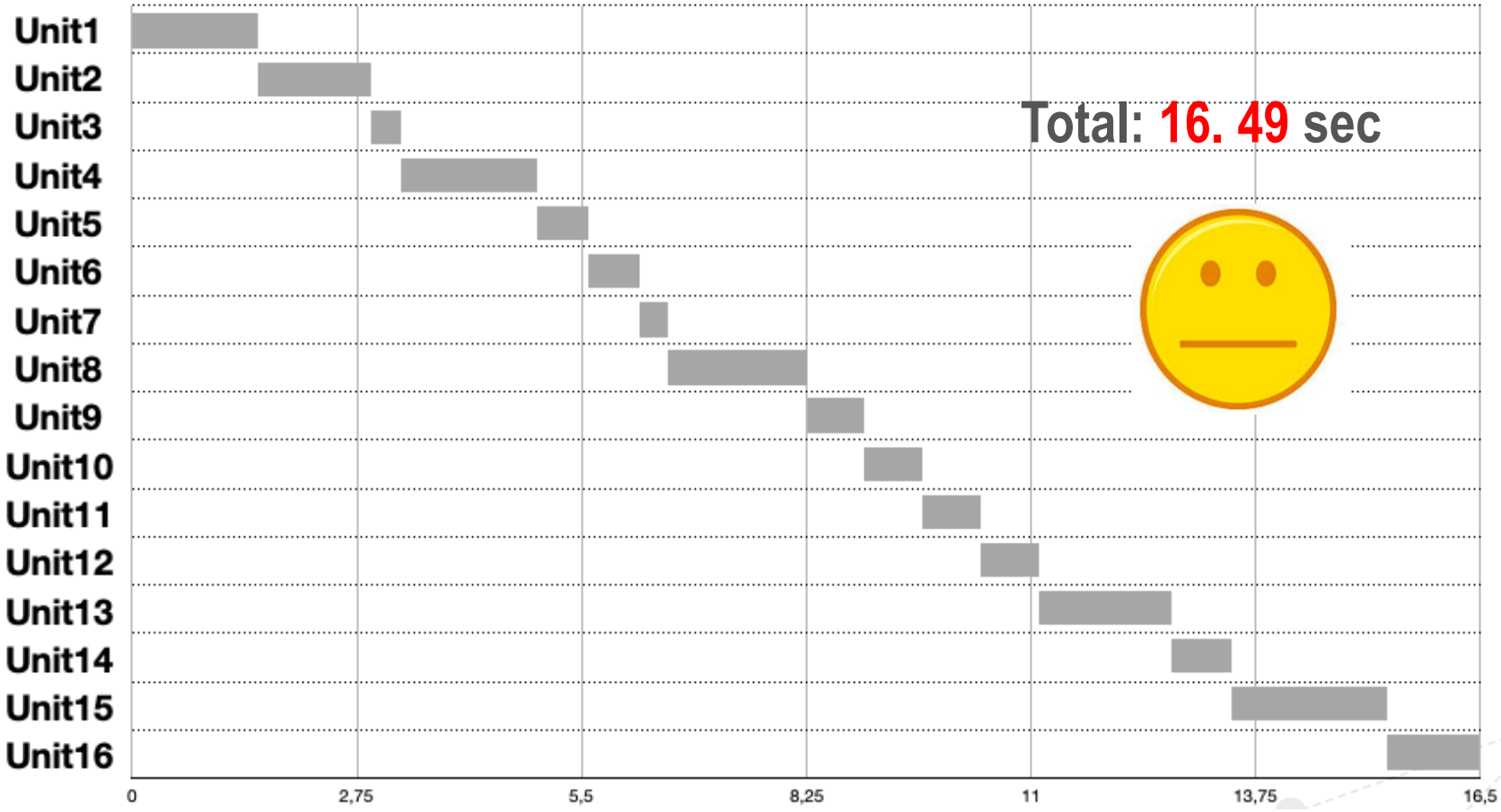
~ 7%

Estimated Resource Usage Summary	
Resource	Usage
Logic utilization	77%
ALUTs	42%
Dedicated logic registers	37%
Memory blocks	71%
DSP blocks	33%

~ 10%

UNIVERSITÄT PADERBORN

ResNet-50 Opt-v2



UNIVERSITÄT PADERBORN

ResNet-50 Opt-v2

The problem are still the same:

- Stalling problem
- Memory dependency

Next step:

- Revert computation logic
- Reduce Memory Stalls

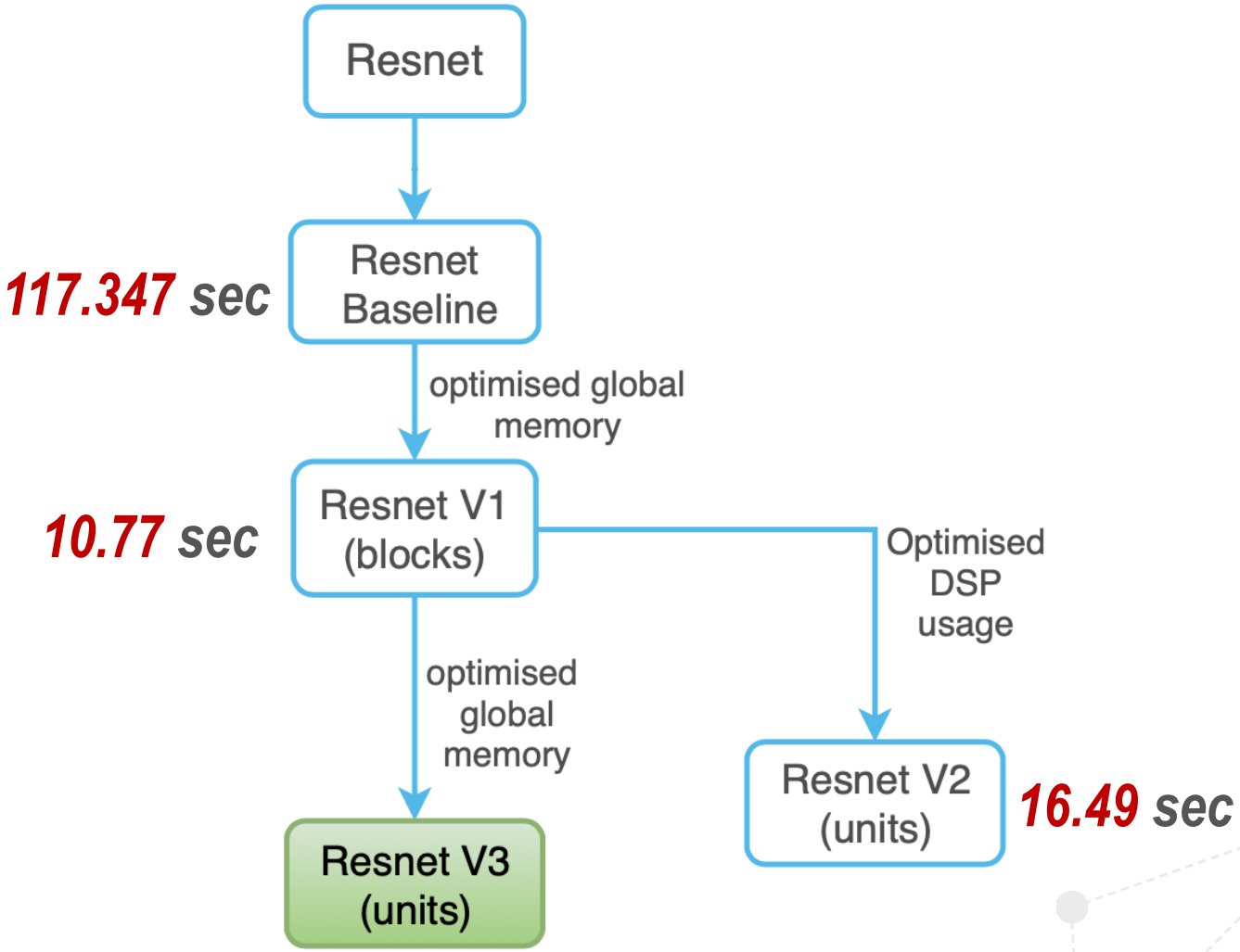
```
        temp_out[l][j] = 0;                                                                  (__loc...  (0.04%)   (1.6%)   (--)
      }
    }
    for (int rc = 0; rc < 64; ++rc) {
      for (int i = 0; i < 56*56; i++){
        l_input[i] = input0[56*56*rc+i];                                                     0: __g...  0: 66.85%  0: 99.7%  0: 675...
      }
#pragma unroll 2
      for (int yy = 0; yy < 56; ++yy) {
#pragma unroll
        for (int xx = 0; xx < 56; ++xx) {
  temp_out[yy][xx] += (l_input[yy * 56 + xx] * input_weights[rc]);                           0: __l...  0: 0.0%   0: 0.9%   0: --

        }
      }
    }
    for (int yy = 0; yy < 56; ++yy){

     for (int xx = 0; xx < 56; ++xx){
        temp_out[yy][xx] += input_bias[ff];                                                  0: __l...  0: 0.0%   0: 0.1%   0: --
        compute[((((ff * 56) + yy) * 56) + xx)] = temp_out[yy][xx];                          (__glo...  (0.05%)   (1.6%)   (10.6M...
```

Read from global memory has 66.85% stalls.

Computations have 0.1% of occupancy

UNIVERSITÄT PADERBORN

ResNet Optimization sequence



UNIVERSITÄT PADERBORN

ResNet-50 : Opt-v3 V/S Opt-v2 comparison

Device Id	Kernel	Compute Unit	0.00ms	1.46s
Device 0	block1_unit_1_bt_v2_add	Compute Unit 0		
Device 0	block1_unit_1_bt_v2_conv3_Conv2D	Compute Unit 0		
Device 0	block1_unit_1_bt_v2_conv2_Conv2D	Compute Unit 0		
Device 0	P_block1_unit_1_bt_v2_conv2_Conv2D	Compute Unit 0		
Device 0	block1_unit_1_bt_v2_conv1_Conv2D	Compute Unit 0		
Device 0	block1_unit_1_bt_v2_shortcut_Conv2D	Compute Unit 0		
Device 0	Mul1_1520_Fused_Mul__FusedScaleShift	Compute Unit 0		
Device 0	pool1_MaxPool	Compute Unit 0		
Device 0	conv1_Conv2D	Compute Unit 0		
Device 0	P_conv1_Conv2D	Compute Unit 0		

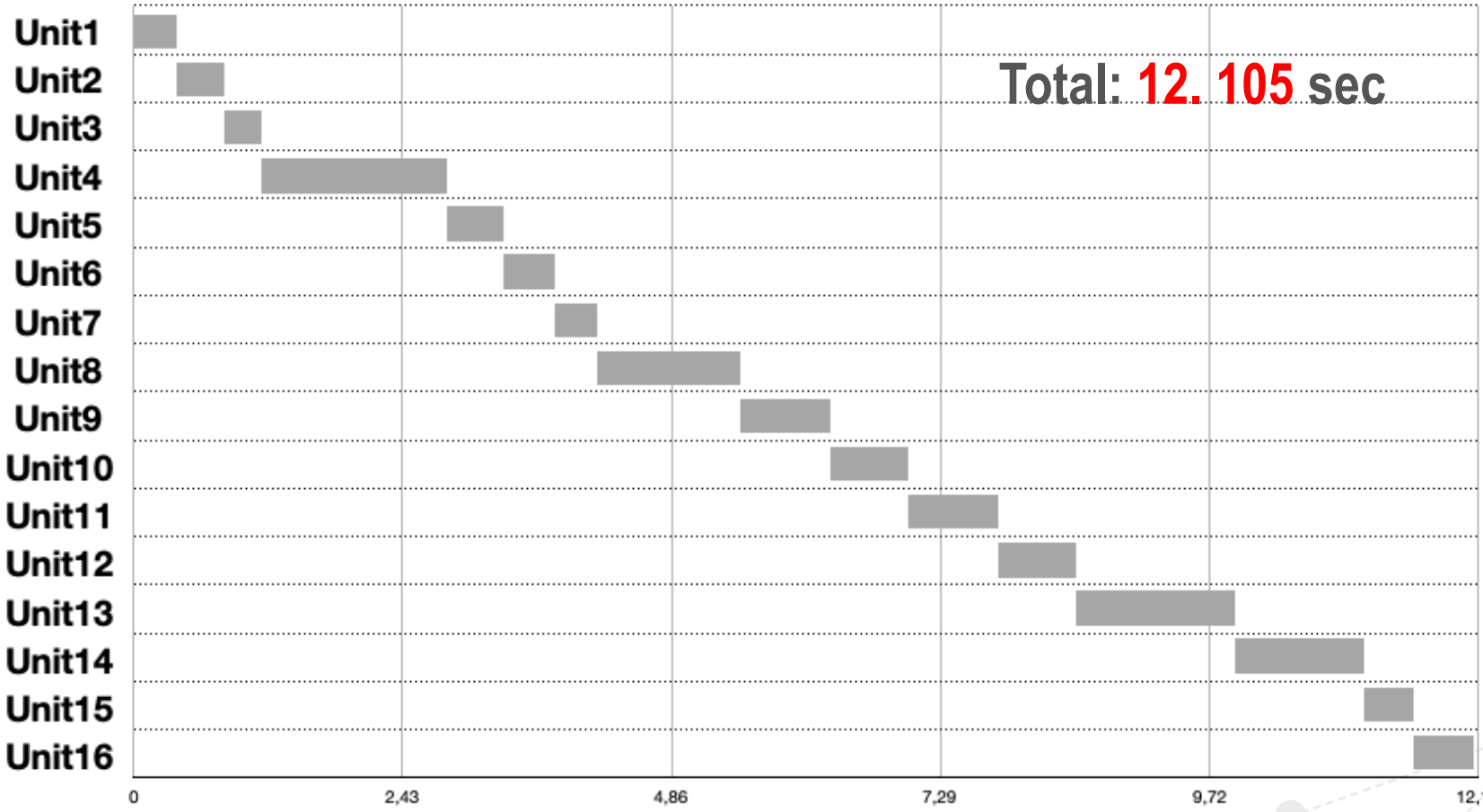
Device Id	Kernel	Compute Unit	0.00ms	410.64ms
Device 0	block1_unit_1_bt_v2_add	Compute Unit 0		
Device 0	block1_unit_1_bt_v2_conv3_Conv2D	Compute Unit 0		
Device 0	block1_unit_1_bt_v2_conv2_Conv2D	Compute Unit 0		
Device 0	P_block1_unit_1_bt_v2_conv2_Conv2D	Compute Unit 0		
Device 0	block1_unit_1_bt_v2_conv1_Conv2D	Compute Unit 0		
Device 0	block1_unit_1_bt_v2_shortcut_Conv2D	Compute Unit 0		
Device 0	Mul1_1520_Fused_Mul__FusedScaleShift	Compute Unit 0		
Device 0	pool1_MaxPool	Compute Unit 0		
Device 0	conv1_Conv2D	Compute Unit 0		
Device 0	P_conv1_Conv2D	Compute Unit 0		

1.46 seconds

~400 milliseconds

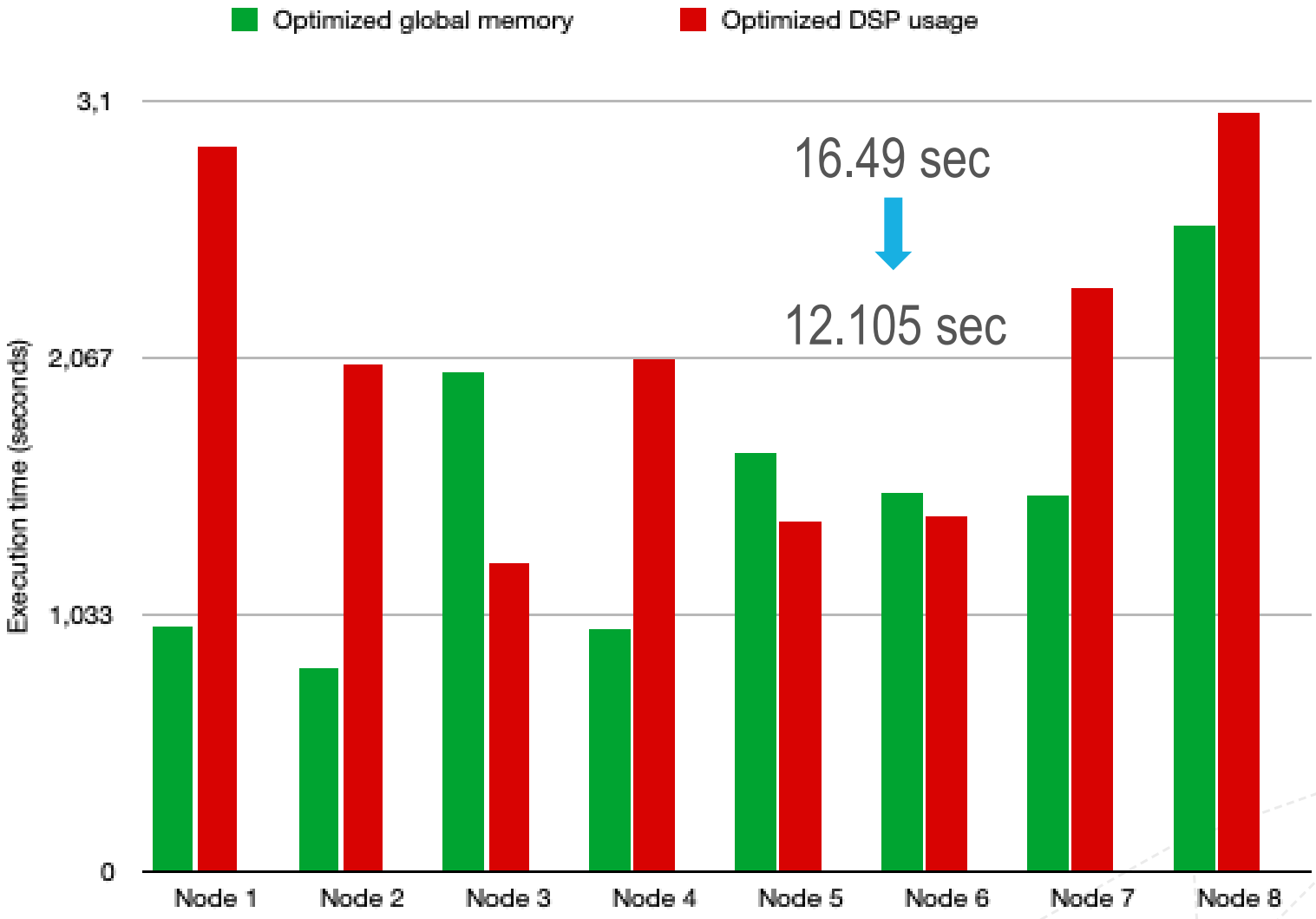
UNIVERSITÄT PADERBORN

ResNet-50 Opt-v3



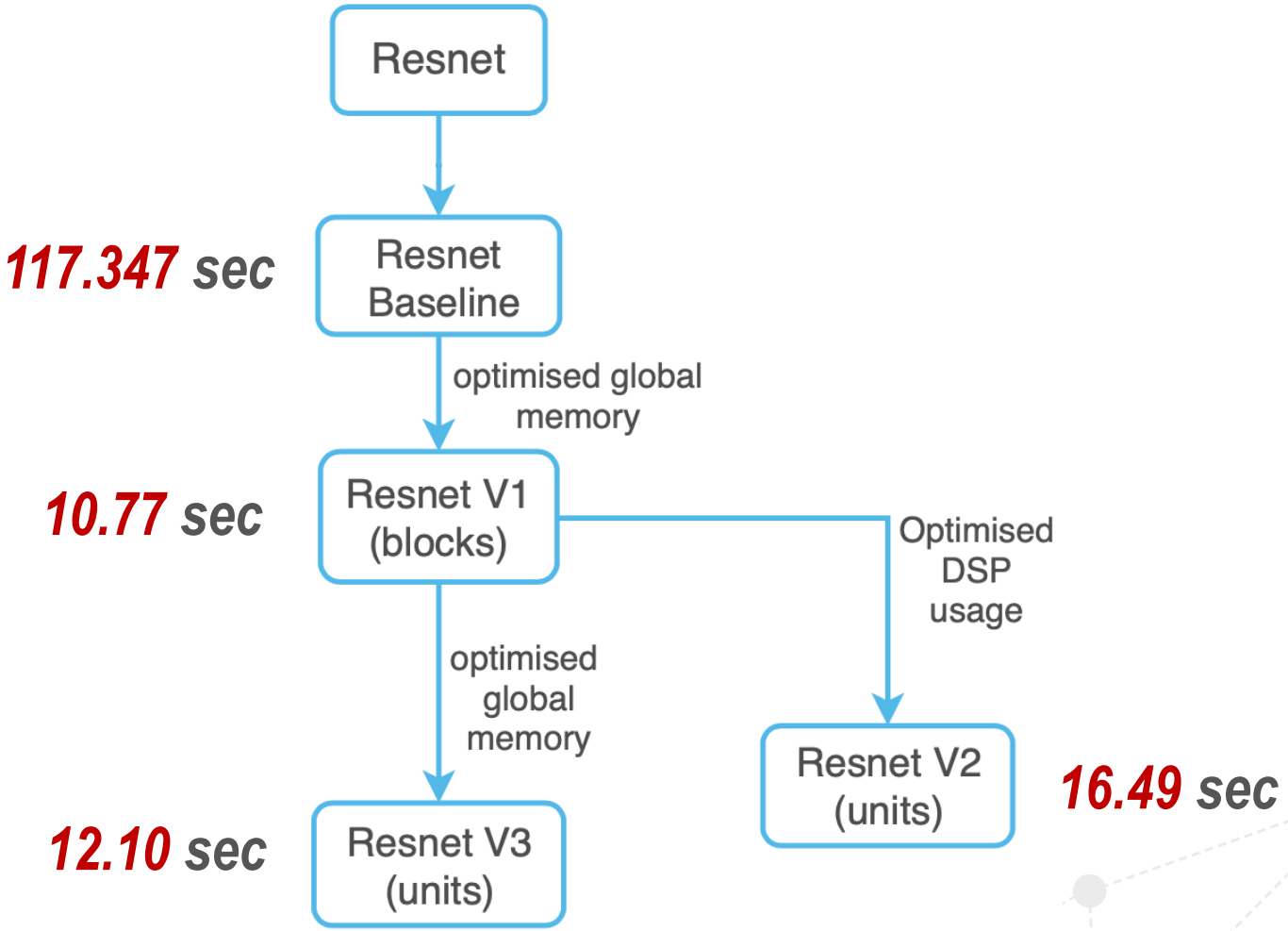
UNIVERSITÄT PADERBORN

ResNet-50 Opt-v3



UNIVERSITÄT PADERBORN

ResNet Optimization sequence



UNIVERSITÄT PADERBORN

Synthesis Statistics

	Number of Synthesis attempts
Successful Synthesis	103
Failed attempts	38
Synthesized designs used for final measurements	71
Unused synthesized designs	32

Compute hours ~ 1,400 hours

More time needed to synthesize all the designs than what it took India to send its mission Chandrayaan 2 to the moon.

UNIVERSITÄT PADERBORN

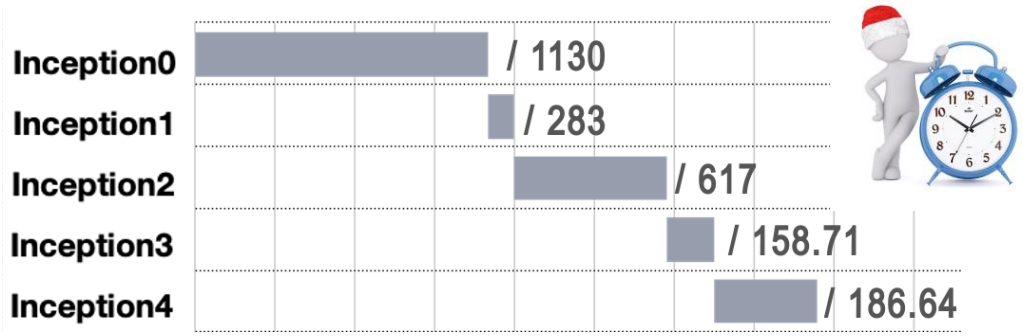
Comparison with state-of-the-art benchmarks

Topology	FPGA execution time	OpenVINO CPU execution time	Microsoft Brainwave
GoogLeNet	1.17 sec	8.79 ms	-
ResNet – 50	10.77 sec	19.42 ms	4 ms

UNIVERSITÄT PADERBORN

Reasons for Sub Optimal Performance

1. FPGAs are idle most of the time



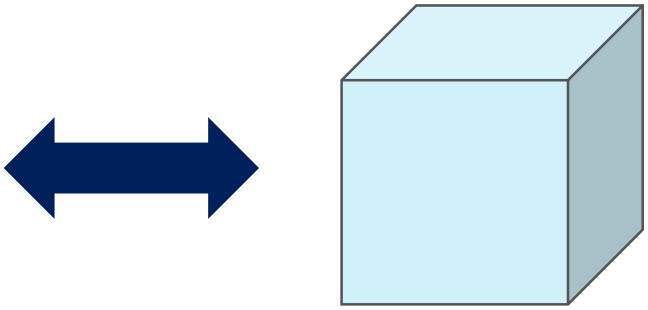
Solution : Batch processing

UNIVERSITÄT PADERBORN

Reasons for Sub Optimal Performance

2. Kernels work on “accumulated data” through channels

```
__kernel void Conv2d_2c_3x3_Conv2D(__global float *restrict input1, __global float *restrict input2)
{

    float input0[215296];
    for (int i = 0; i < 215296; i++)
    {
        input0[i] = read_channel_intel(padding_2c_out_channel);
    }
    for (int ff = 0; ff < 192; ++ff)
    {
        for (int yy = 0; yy < 56; ++yy)
        {
            for (int xx = 0; xx < 56; ++xx)
            {
```

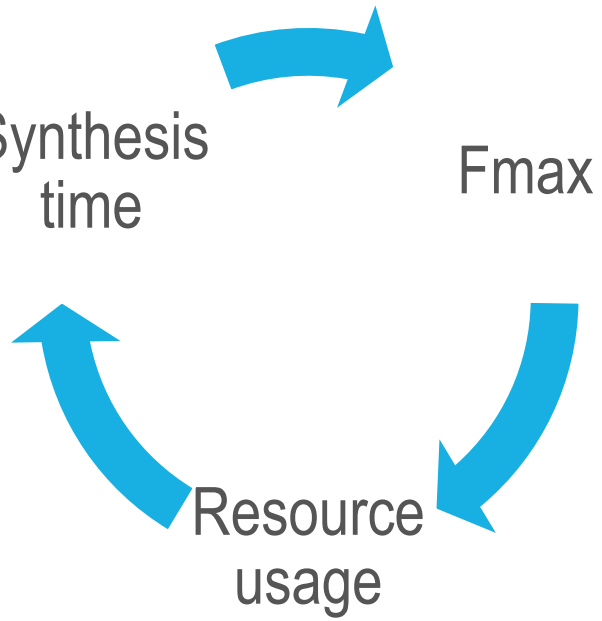


Solution : Streaming design will help

UNIVERSITÄT PADERBORN

Reasons for Sub Optimal Performance

3. Difficult to find trade-off between following factors



4. Lack of Experience

UNIVERSITÄT PADERBORN

Goals

- ✓ Deployment of state-of-the-art CNN models on FPGA
- ✓ Stretching the CNN model on the Noctua FPGA Infrastructure at PC2 with 32 Stratix 10 FPGAs
- ✓ Compare results with state-of-the-art benchmarks
- ○ Quantization on CNN floating point weights

UNIVERSITÄT PADERBORN

Reasons for not doing Quantization

- Quantization only supported by OpenVINO version 2019.1 (latest)
- Required to migrate the project from 2018.5 version to the new one, late in development
- Many branches of the repository simultaneously under development made migration difficult

UNIVERSITÄT PADERBORN

Future tasks

- Solve memory dependency in the kernels
- Exploring the performance by changing the width of the channels in Googlenet channel design
- Batch processing of image classification
- Generalised OpenVINO FPGA plugin to support different CNNs

Summary

- State-of-the-art CNNs were deployed on FGPAs.
- Machine Learning Toolkits were customized to fit our PG scope.
- Scaling was achieved with the help of MPI and FPGA IO channels.
- Sharpened project management and teamwork skills.
- Quantization could not be done due to the limited support from OpenVINO Model Optimizer.