DEEPHEALTH

Reconfigurable Architectures Support in EDDL

José Flich

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

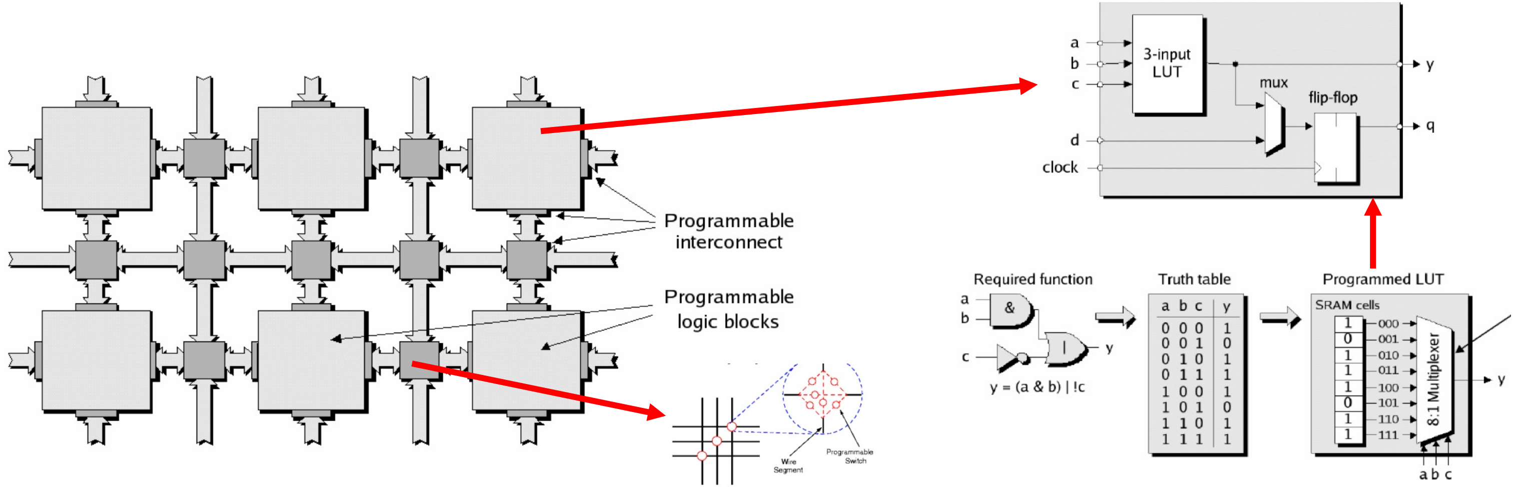
Outline

- Introduction to FPGAs
- High Level Synthesis
- HLSinf accelerator
- HLSinf – EDDL support
- Results
- Conclusion

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Introduction to FPGAs

- Field Programmable Gate Array
  - First introduced by Xilinx in 1985
  - Two major makers: Xilinx (part of AMD) and Altera (owned by Intel)
  - Large array of configurable logic blocks (CLB) connected via programable interconnects

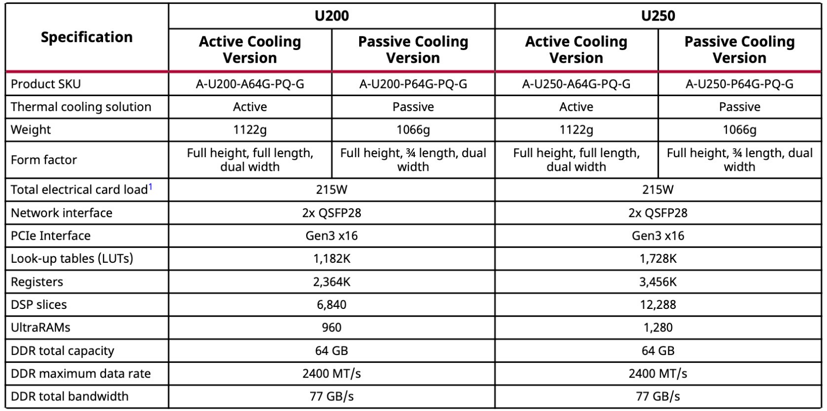


a	b	c	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

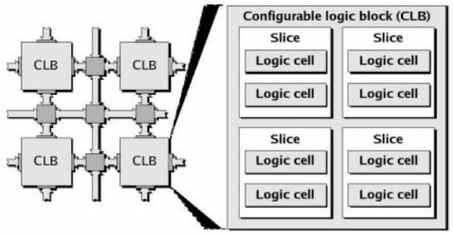
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

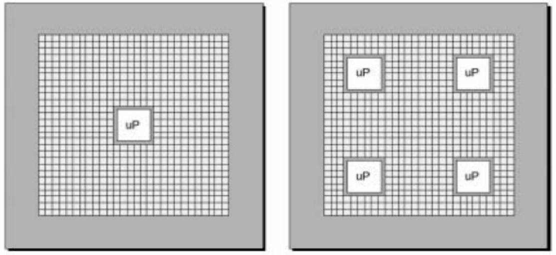
Introduction to FPGAs

- Hierarchical design
- Highly (evolving) heterogeneous system
  - DSPs, embedded cores, AI accelerators, real-time processors, …



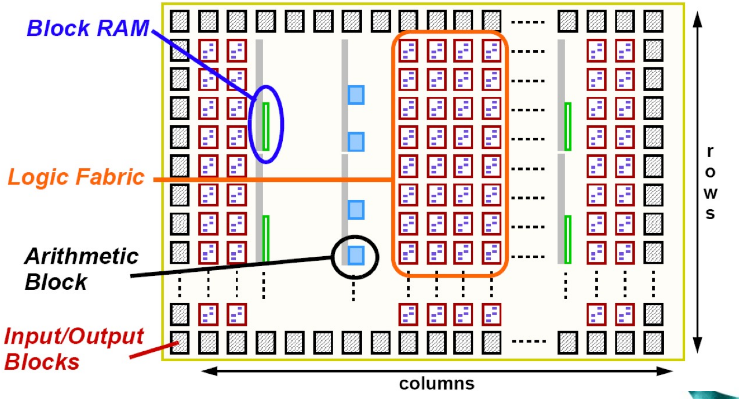
Specification	U200		U250	
	Active Cooling Version	Passive Cooling Version	Active Cooling Version	Passive Cooling Version
Product SKU	A-U200-A64G-PQ-G	A-U200-P64G-PQ-G	A-U250-A64G-PQ-G	A-U250-P64G-PQ-G
Thermal cooling solution	Active	Passive	Active	Passive
Weight	1122g	1066g	1122g	1066g
Form factor	Full height, full length, dual width	Full height, ¾ length, dual width	Full height, full length, dual width	Full height, ¾ length, dual width
Total electrical card load[1]	215W		215W	
Network interface	2x QSFP28		2x QSFP28	
PCIe Interface	Gen3 x16		Gen3 x16	
Look-up tables (LUTs)	1,182K		1,728K	
Registers	2,364K		3,456K	
DSP slices	6,840		12,288	
UltraRAMs	960		1,280	
DDR total capacity	64 GB		64 GB	
DDR maximum data rate	2400 MT/s		2400 MT/s	
DDR total bandwidth	77 GB/s		77 GB/s	





(a) One embedded core

(b) Four embedded cores



The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Introduction to FPGAs

- With an FPGA you can implement any circuit design
  - From a simple logic gate to a highly complex processor
  - Is a white paper and pencil approach for architecture design and deployment
- But…
  - Highly complex system may demand high expertise level for full exploitation
  - Hardware description programming languages (VHDL, Verilog)
  - Complex design flow
    - Synthesis, place & route, timing closure, debugging complexity, …
    - Design cycle time measured in hours/days
  - To the rescue we have High Level Synthesis

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

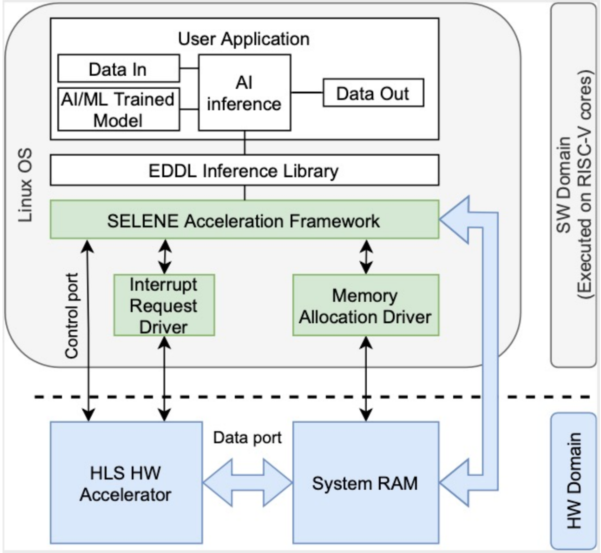
FPGAs in the DeepHealth project

- FPGAs are good devices for “irregular” applications
  - DeepLearning algorithms are known for being matrix-multiply intensize (thus, regular)
  - GPUs excel at regular applications (massive parallelism exploited by thousands of independent threads running concurrently)
- Training vs Inference
  - The training process typically requires high precision arithmetic (FP32)
  - The inference process can be simplified by using a lower precision format (Fixed Point or integers)
  - FPGAs have superior performance for reduced precision algorithms
  - Thus, FPGAs are good devices for inference processes
    - Models are quantized and compressed
- Power consumption and reusability
  - FPGAs exhibit much power consumption than GPUs and CPUs
  - FPGAs are good for latency-bound applications (e.g. infer an output in less than 2ms)
    - Latency vs Throughput

The project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 825111.

EDDL/FPGAs in other Projects

- EU SELENE project (https://www.selene-project.eu)
  - Goal: Deploy an embedded real-time predictable platform based on RISC-V processors and AI accelerators
    - Library selected: EDDL
    - Accelerator selected: HLSinf
  - All, processors, accelerator and EDDL running within an FPGA for different use cases:
    - Autonomous robot
    - Autonomous train
    - Satellites and deep space stations
  - Achievement:
    - Enabling HW/SW co-design with EDDL



SELENE platform

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

High Level Synthesis

- (Xilinx) Vitis HLS tool
  - Allows **C, C++, OpenCL functions to become hardwired** onto the device logic fabric and RAM/DSP blocks
  - Automates most of the code modifications required to implement and optimize C/C++ code in programable logic
  - Achieves low latency and high throughput
  - Based on **pragmas** to let produce right interfaces, pipeline loops and functions
- Vitis HLS design flow
  - Compile, simulate, debug C/C++ algorithm
  - View reports to analyse and optimize the design
  - Synthesize the C algorithm into RTL design
  - Verify the RTL implementation using RTL co-simulation
  - Package the RTL implementation into a compiled object file (.xo)

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

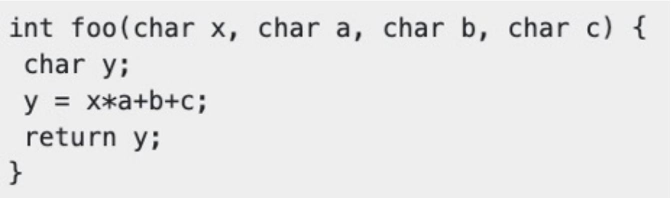
High Level Synthesis

- Benefits
  - Developing/Validating algorithms at C-level
  - C-simulation to validate the design (quick iterations)
  - Controlling the C-synthesis process using optimization pragmas
  - Creating multiple design solutions from the C source code and pragmas
  - Quickly recompile the C-source to target different platforms and hardware devices
- Stages
  - Scheduling
  - Binding
  - Control logic extraction

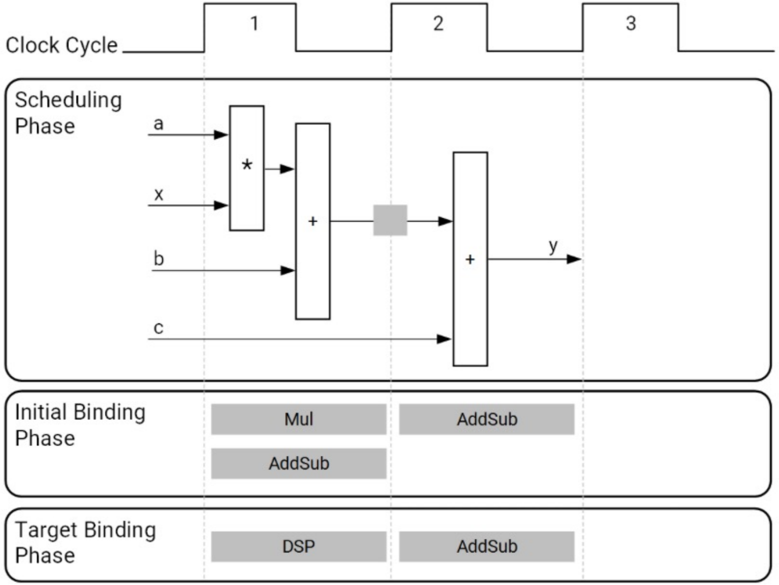
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

High Level Synthesis

- Scheduling determines which operations occur during each cycle based on:
  - When an operation's dependence has been satisfied or is available
  - The length of the clock cycle
  - Time the operation takes to complete
  - Available resource allocation
  - User-specified optimization directives
- Binding assigns hardware resources to implement each scheduled operation
  - Initial binding
  - Target binding



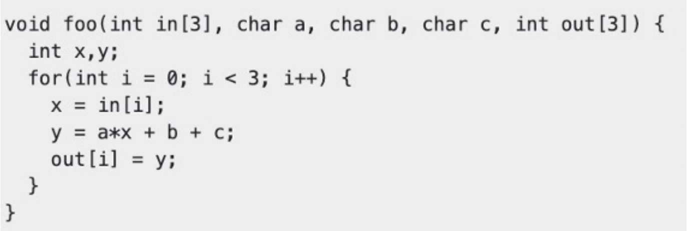
```
int foo(char x, char a, char b, char c) {
 char y;
 y = x*a+b+c;
 return y;
}
```



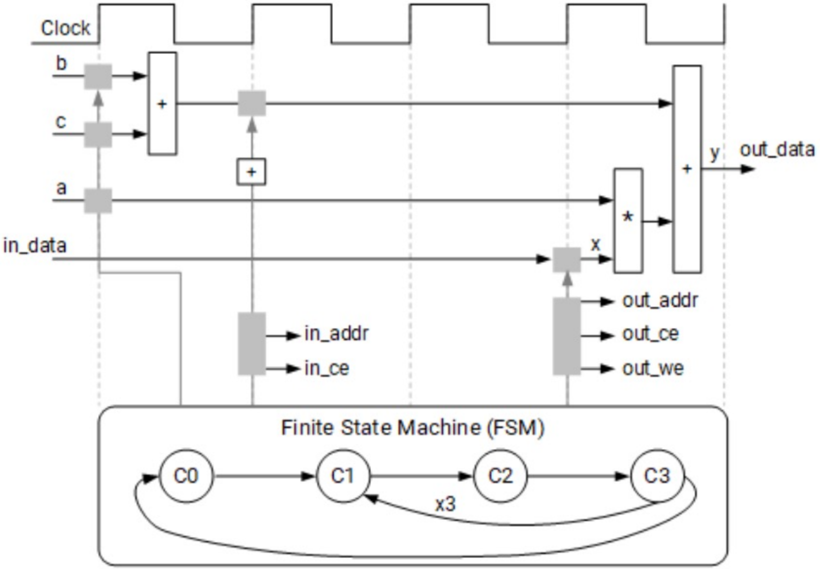
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

High Level Synthesis

- Control logic extraction
  - Operations inside a for-loop
  - Two of the arguments are arrays
  - Logic inside the for-loop three times
  - FSM created from the code (C0..C3)
  - Top-level function arguments become ports in the final RTL design
  - Arrays implemented as Block RAM (by default)



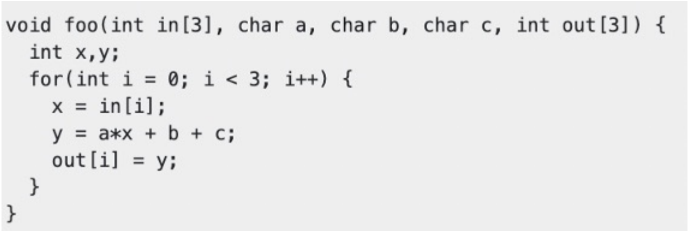
```
void foo(int in[3], char a, char b, char c, int out[3]) {
  int x,y;
  for(int i = 0; i < 3; i++) {
    x = in[i];
    y = a*x + b + c;
    out[i] = y;
  }
}
```



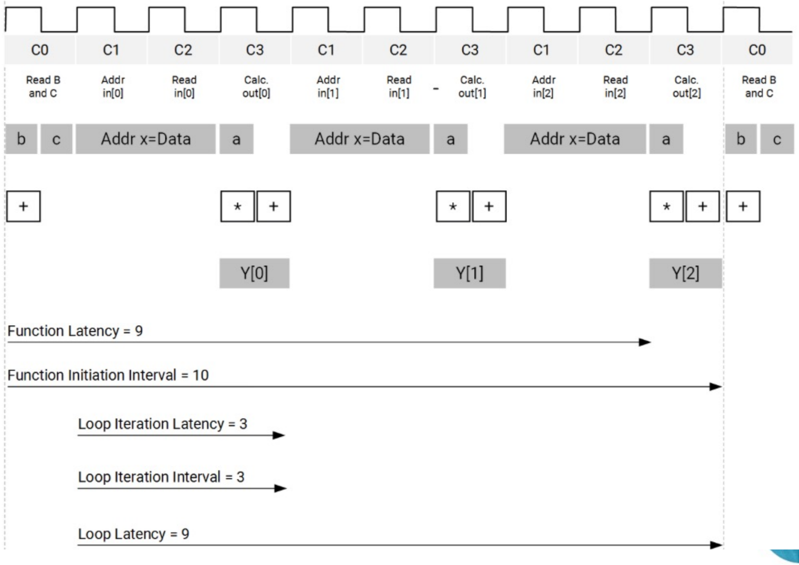
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

High Level Synthesis

- Performance metrics example
  - Latency: 9 cycles to output all values
  - **Initiation Interval** (II): 10
    - It takes 10 cycles before the function can initiate a new set of input reads and start to process the next set of input data
  - Loop iteration latency
  - Loop Initiation interval
  - Loop latency



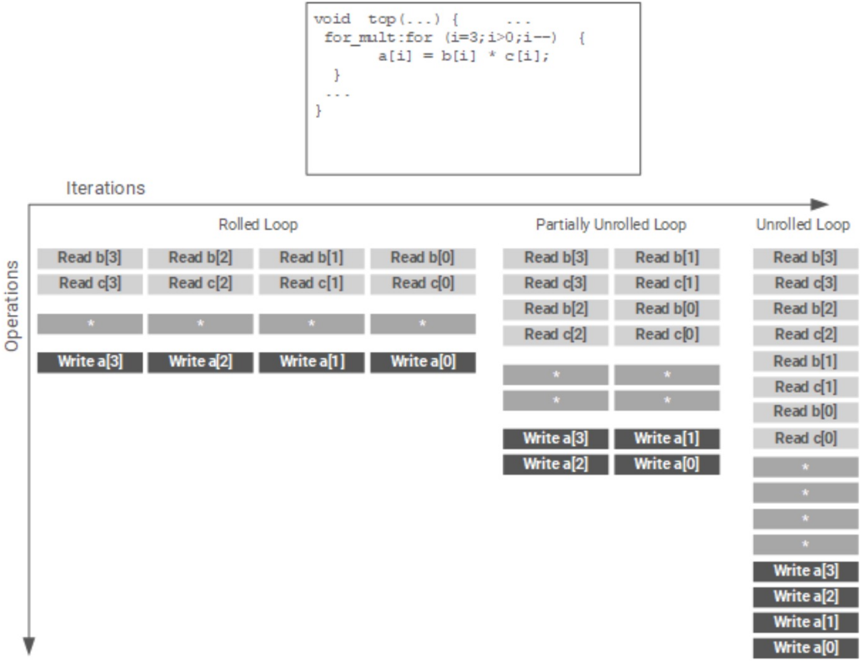
```
void foo(int in[3], char a, char b, char c, int out[3]) {
  int x,y;
  for(int i = 0; i < 3; i++) {
    x = in[i];
    y = a*x + b + c;
    out[i] = y;
  }
}
```



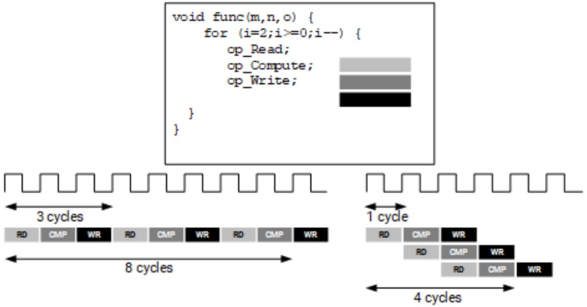
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

High Level Synthesis

- Optimizing for Throughput
  - PIPELINE directive (or pragma)
    - Loop pipelining and function pipelining
  - UNROLL directive (or pragma)
    - Partial or completely unroll factor
    - Rolled loop (one multiplier)
    - Unrolled loop (four multipliers)



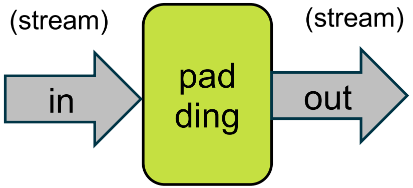
```
void  top(...) {      ...
 for_mult:for (i=3;i>0;i--)  {
      a[i] = b[i] * c[i];
  }
 ...
}
```

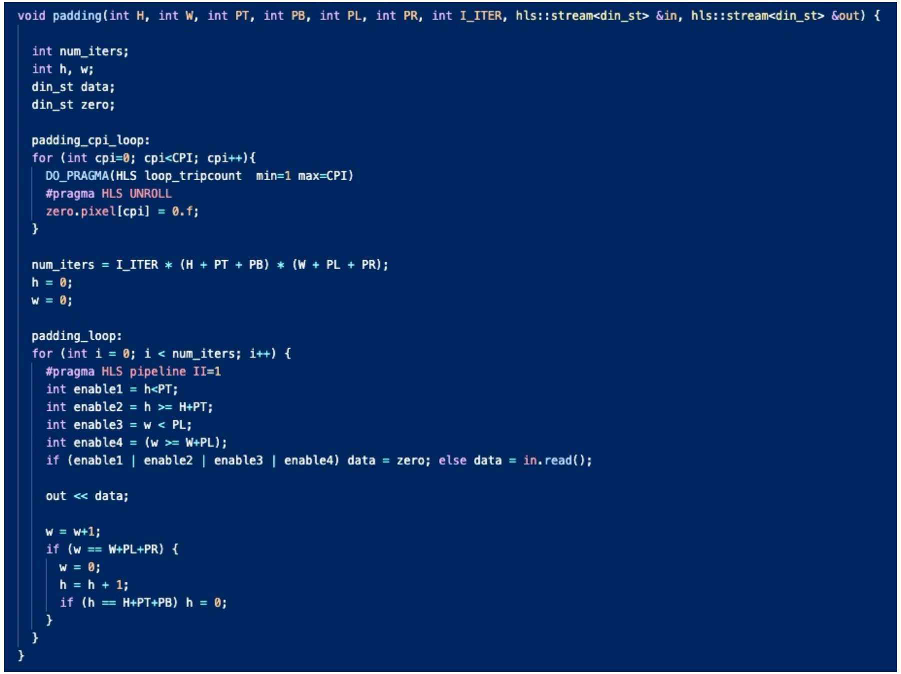


```
void func(m,n,o) {
    for (i=2;i>=0;i--) {
        op_Read;
        op_Compute;
        op_Write;

    }
}
```

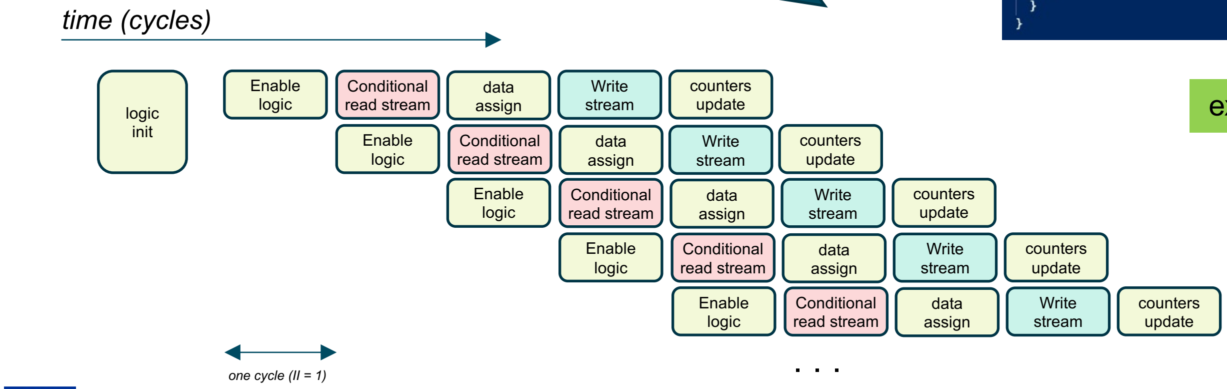
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

High Level Synthesis (pipeline)





```
void padding(int H, int W, int PT, int PB, int PL, int PR, int I_ITER, hls::stream<din_st> &in, hls::stream<din_st> &out) {

  int num_iters;
  int h, w;
  din_st data;
  din_st zero;

  padding_cpi_loop:
  for (int cpi=0; cpi<CPI; cpi++){
    DO_PRAGMA(HLS loop_tripcount  min=1 max=CPI)
    #pragma HLS UNROLL
    zero.pixel[cpi] = 0.f;
  }

  num_iters = I_ITER * (H + PT + PB) * (W + PL + PR);
  h = 0;
  w = 0;

  padding_loop:
  for (int i = 0; i < num_iters; i++) {
    #pragma HLS pipeline II=1
    int enable1 = h<PT;
    int enable2 = h >= H+PT;
    int enable3 = w < PL;
    int enable4 = (w >= W+PL);
    if (enable1 | enable2 | enable3 | enable4) data = zero; else data = in.read();

    out << data;

    w = w+1;
    if (w == W+PL+PR) {
      w = 0;
      h = h + 1;
      if (h == H+PT+PB) h = 0;
    }
  }
}
```



time (cycles)

logic init

Enable logic | Conditional read stream | data assign | Write stream | counters update

Enable logic | Conditional read stream | data assign | Write stream | counters update

Enable logic | Conditional read stream | data assign | Write stream | counters update

Enable logic | Conditional read stream | data assign | Write stream | counters update

Enable logic | Conditional read stream | data assign | Write stream | counters update

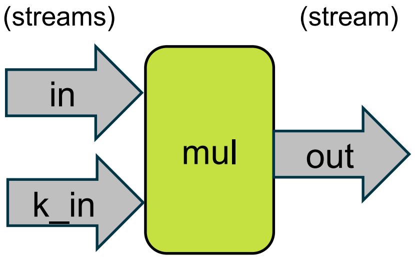
one cycle (II = 1)

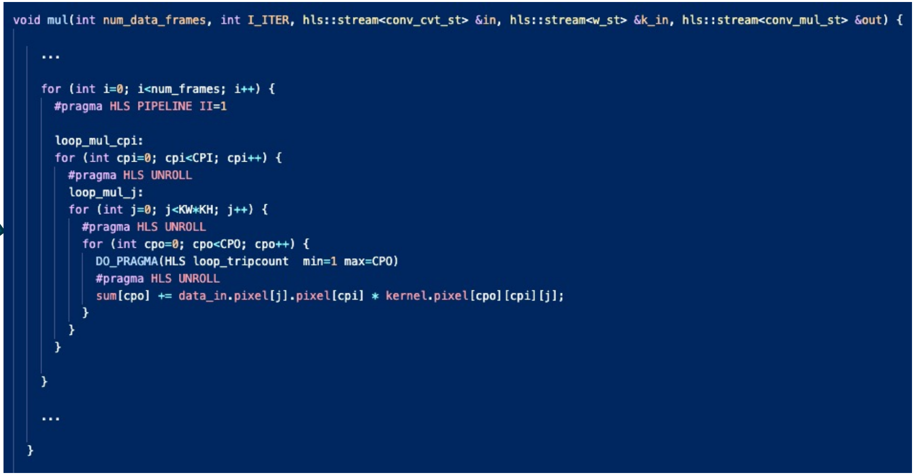
. . .

execution time = (H+PT+PB) x (W+PL+PR) cycles

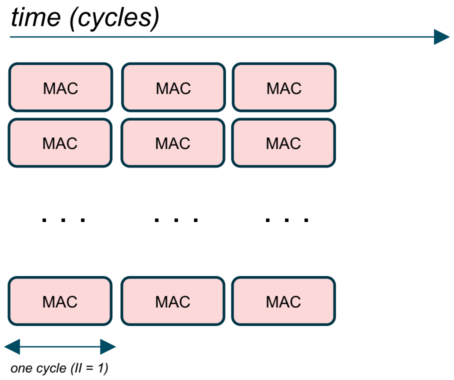
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

High Level Synthesis (unroll)





```
void mul(int num_data_frames, int I_ITER, hls::stream<conv_cvt_st> &in, hls::stream<w_st> &k_in, hls::stream<conv_mul_st> &out) {

    ...

    for (int i=0; i<num_frames; i++) {
      #pragma HLS PIPELINE II=1

      loop_mul_cpi:
      for (int cpi=0; cpi<CPI; cpi++) {
        #pragma HLS UNROLL
        loop_mul_j:
        for (int j=0; j<KW*KH; j++) {
          #pragma HLS UNROLL
          for (int cpo=0; cpo<CPO; cpo++) {
            DO_PRAGMA(HLS loop_tripcount  min=1 max=CPO)
            #pragma HLS UNROLL
            sum[cpo] += data_in.pixel[j].pixel[cpi] * kernel.pixel[cpo][cpi][j];
          }
        }
      }

    }

    ...

  }
```

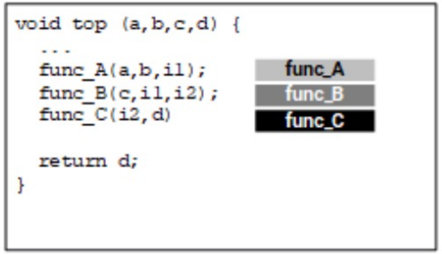


execution time = `num_frames` cycles

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

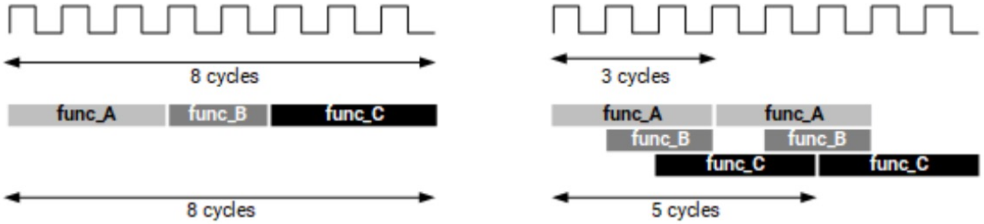
High Level Synthesis

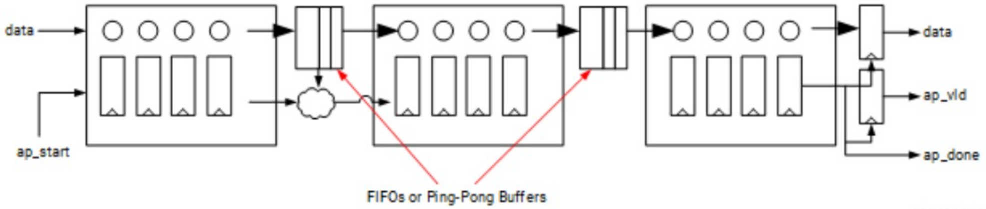
- Optimizing for Throughput
  - DATAFLOW directive (or pragma)
  - Exploiting Task Level Parallelism
  - Dataflow optimization creates an architecture of concurrent processes connected with channels
  - Added overhead
    - FIFO channels as BRAM
  - Each process goes at its own pace
  - Empty and Full Channels are blocking for Reading and writing, respectively
  - Applicable to a DAG (not simply chain of processes)



```
void top (a,b,c,d) {
  ...
  func_A(a,b,i1);
  func_B(c,i1,i2);
  func_C(i2,d)

  return d;
}
```

func_A
func_B
func_C

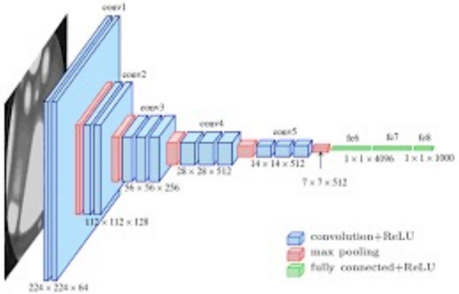




The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

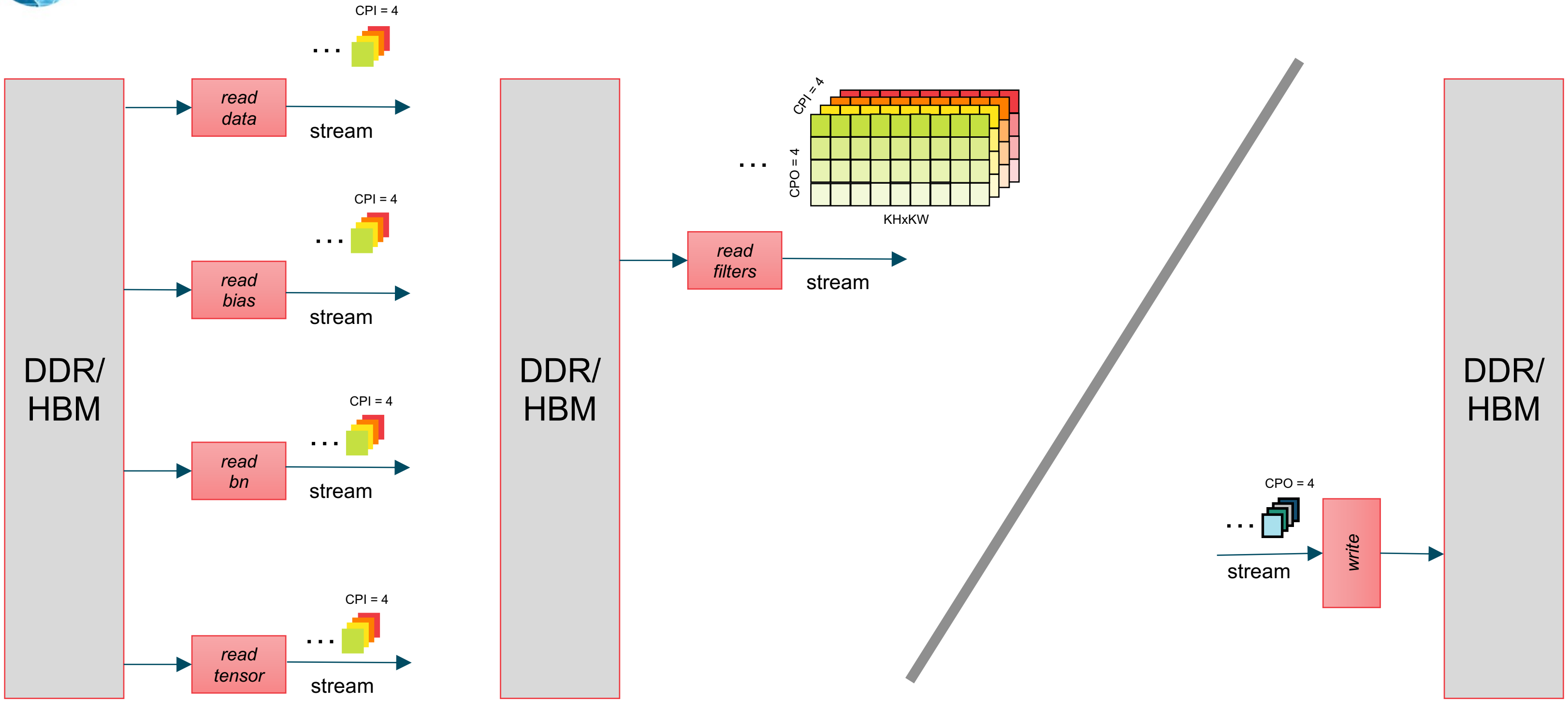
HLSinf accelerator

- Inference of complex neural network models on FPGA
- Open source: https://github.com/PEAK-UPV/HLSinf
- Written in HLS
- Dataflow model, pipelined loops, unrolled loops (maximum throughput)
- Input/Output channel parallelism (CPI/CPO)
- Data precision formats (FP32, AP_FP<n>, APINT<n>)
- Multifunction support
  - Conv2D, ReLu, Pooling, Softplus, Tanh, TensorMult, TensorAdd, …
- Conv types: Direct, Winograd-based, DepthWise Separable (DWS)
- Deterministic performance, Energy efficient (<30Watt Alveo U200)
- Multidevice:
  - Xilinx Alveo U200, U280
  - Xilinx Kintex (MANGO prototype)
  - Intel Stratix 10 (DeepHealth board by PRODESIGN)



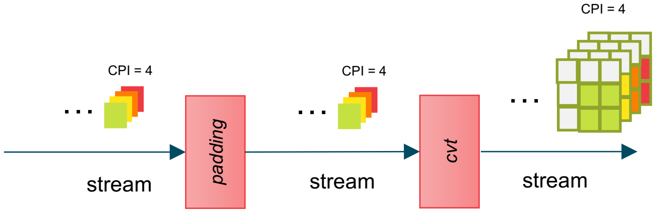
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

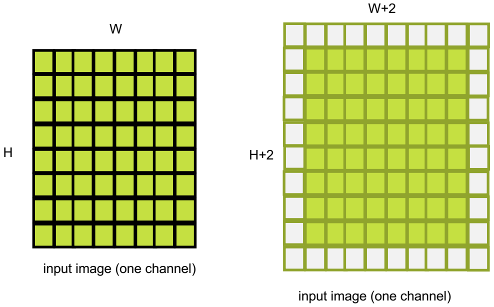
HLSinf accelerator (read & write)



The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

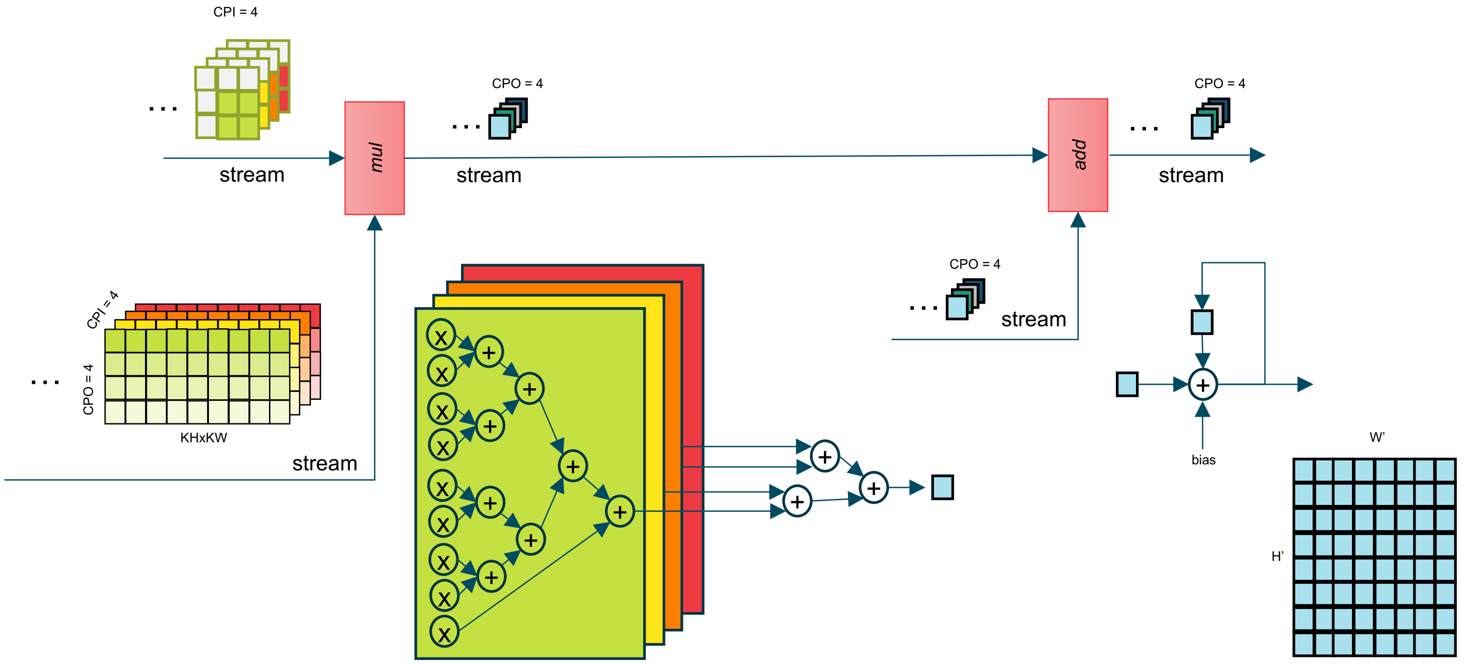
HLSinf accelerator (transformations)





The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

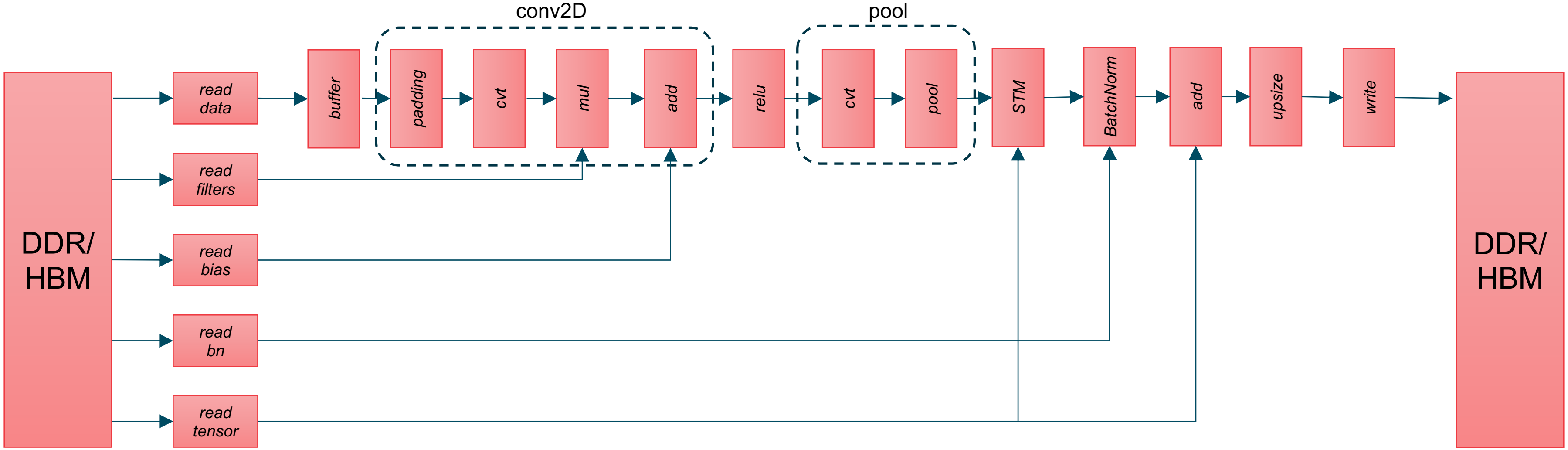
HLSinf accelerator (compute)



(operations for one output channel)

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

HLSinf accelerator (dataflow)

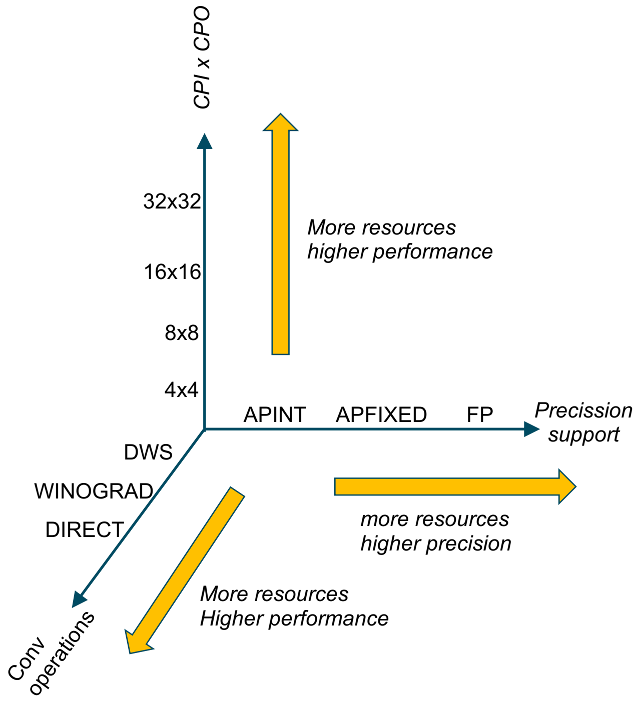


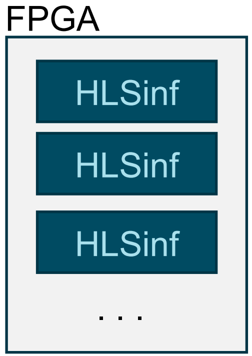
(streamed dataflow, fully pipelined, Initial Interval = 1)

Execution Time = L + (H x W x (I / CPI) x (O / CPO)) cycles
Frequency = 300 MHz

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

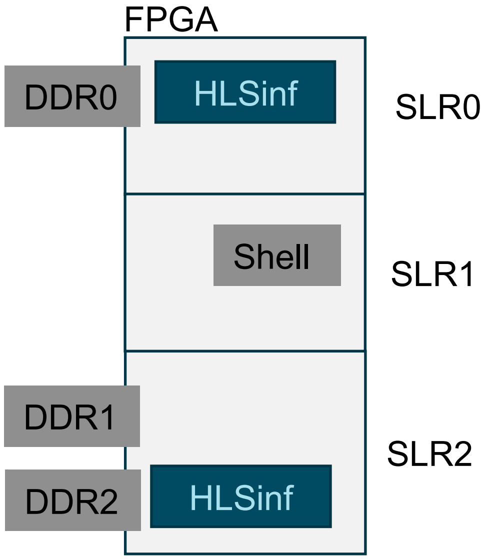
HLSinf accelerator (scalability)



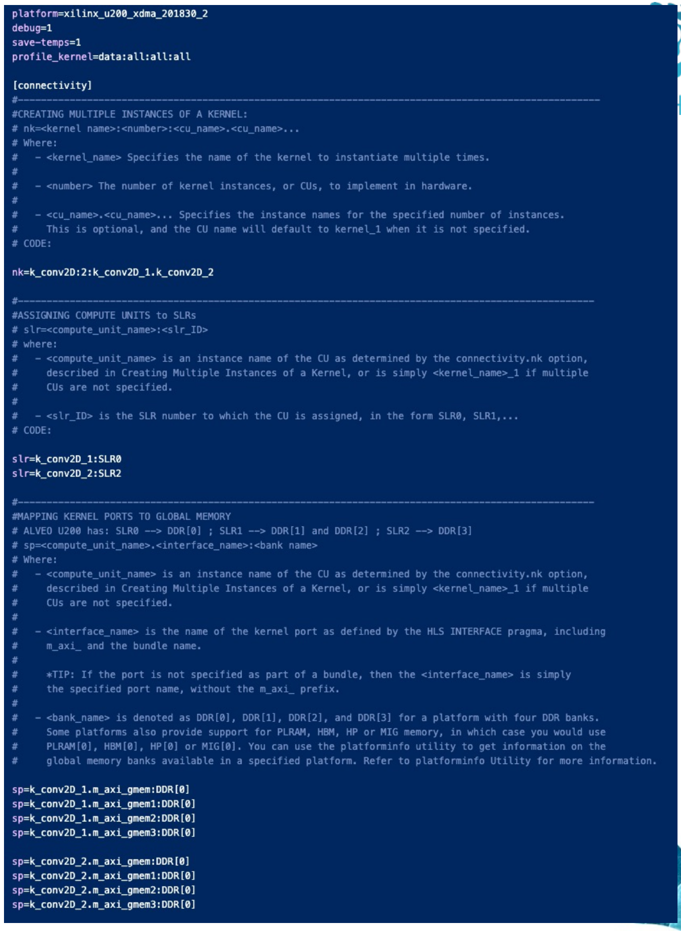


The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

HLSinf accelerator



(just an example)



```
platform=xilinx_u200_xdma_201830_2
debug=1
save-temps=1
profile_kernel=data:all:all:all

[connectivity]
#----------------------------------------------------------------------------------------------------------
#CREATING MULTIPLE INSTANCES OF A KERNEL:
# nk=<kernel name>:<number>:<cu_name>.<cu_name>...
# Where:
#   - <kernel_name> Specifies the name of the kernel to instantiate multiple times.
#
#   - <number> The number of kernel instances, or CUs, to implement in hardware.
#
#   - <cu_name>.<cu_name>... Specifies the instance names for the specified number of instances.
#     This is optional, and the CU name will default to kernel_1 when it is not specified.
# CODE:

nk=k_conv2D:2:k_conv2D_1.k_conv2D_2

#---------------------------------------------------------------------------------------------------------
#ASSIGNING COMPUTE UNITS to SLRs
# slr=<compute_unit_name>:<slr_ID>
# where:
#   - <compute_unit_name> is an instance name of the CU as determined by the connectivity.nk option,
#     described in Creating Multiple Instances of a Kernel, or is simply <kernel_name>_1 if multiple
#     CUs are not specified.
#
#   - <slr_ID> is the SLR number to which the CU is assigned, in the form SLR0, SLR1,...
# CODE:

slr=k_conv2D_1:SLR0
slr=k_conv2D_2:SLR2

#---------------------------------------------------------------------------------------------------------
#MAPPING KERNEL PORTS TO GLOBAL MEMORY
# ALVEO U200 has: SLR0 --> DDR[0] ; SLR1 --> DDR[1] and DDR[2] ; SLR2 --> DDR[3]
# sp=<compute_unit_name>.<interface_name>:<bank name>
# Where:
#   - <compute_unit_name> is an instance name of the CU as determined by the connectivity.nk option,
#     described in Creating Multiple Instances of a Kernel, or is simply <kernel_name>_1 if multiple
#     CUs are not specified.
#
#   - <interface_name> is the name of the kernel port as defined by the HLS INTERFACE pragma, including
#     m_axi_ and the bundle name.
#
#     *TIP: If the port is not specified as part of a bundle, then the <interface_name> is simply
#     the specified port name, without the m_axi_ prefix.
#
#   - <bank_name> is denoted as DDR[0], DDR[1], DDR[2], and DDR[3] for a platform with four DDR banks.
#     Some platforms also provide support for PLRAM, HBM, HP or MIG memory, in which case you would use
#     PLRAM[0], HBM[0], HP[0] or MIG[0]. You can use the platforminfo utility to get information on the
#     global memory banks available in a specified platform. Refer to platforminfo Utility for more information.

sp=k_conv2D_1.m_axi_gmem:DDR[0]
sp=k_conv2D_1.m_axi_gmem1:DDR[0]
sp=k_conv2D_1.m_axi_gmem2:DDR[0]
sp=k_conv2D_1.m_axi_gmem3:DDR[0]

sp=k_conv2D_2.m_axi_gmem:DDR[0]
sp=k_conv2D_2.m_axi_gmem1:DDR[0]
sp=k_conv2D_2.m_axi_gmem2:DDR[0]
sp=k_conv2D_2.m_axi_gmem3:DDR[0]
```

conf file for Alveo U200 (2 kernels)

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

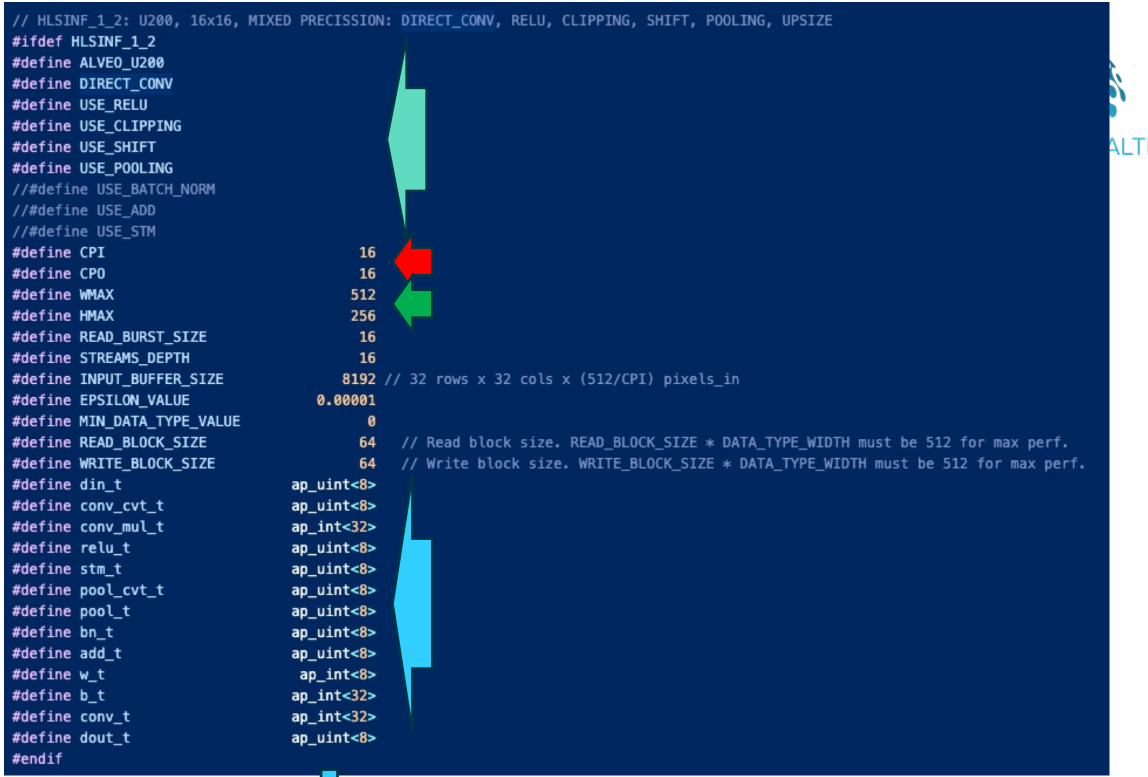
HLSinf conf

functionality

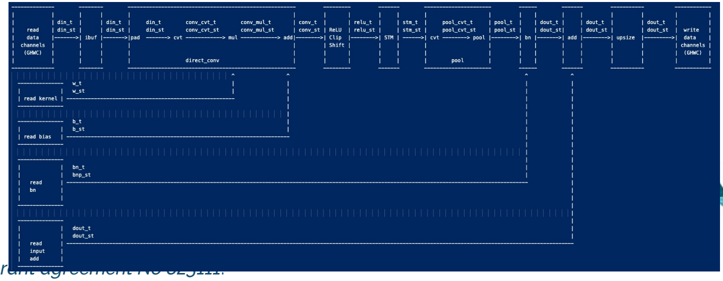
Input paralelism and output paralelism

maximum column and row of images

data path widths configuration



```
// HLSINF_1_2: U200, 16x16, MIXED PRECISSION: DIRECT_CONV, RELU, CLIPPING, SHIFT, POOLING, UPSIZE
#ifdef HLSINF_1_2
#define ALVEO_U200
#define DIRECT_CONV
#define USE_RELU
#define USE_CLIPPING
#define USE_SHIFT
#define USE_POOLING
//#define USE_BATCH_NORM
//#define USE_ADD
//#define USE_STM
#define CPI                                  16
#define CPO                                  16
#define WMAX                                512
#define HMAX                                256
#define READ_BURST_SIZE                      16
#define STREAMS_DEPTH                        16
#define INPUT_BUFFER_SIZE                  8192 // 32 rows x 32 cols x (512/CPI) pixels_in
#define EPSILON_VALUE                   0.00001
#define MIN_DATA_TYPE_VALUE                   0
#define READ_BLOCK_SIZE                      64   // Read block size. READ_BLOCK_SIZE * DATA_TYPE_WIDTH must be 512 for max perf.
#define WRITE_BLOCK_SIZE                     64   // Write block size. WRITE_BLOCK_SIZE * DATA_TYPE_WIDTH must be 512 for max perf.
#define din_t                        ap_uint<8>
#define conv_cvt_t                   ap_uint<8>
#define conv_mul_t                   ap_int<32>
#define relu_t                       ap_uint<8>
#define stm_t                        ap_uint<8>
#define pool_cvt_t                   ap_uint<8>
#define pool_t                       ap_uint<8>
#define bn_t                         ap_uint<8>
#define add_t                        ap_uint<8>
#define w_t                           ap_int<8>
#define b_t                          ap_int<32>
#define conv_t                       ap_int<32>
#define dout_t                       ap_uint<8>
#endif
```



HLSinf 1.2 configuration

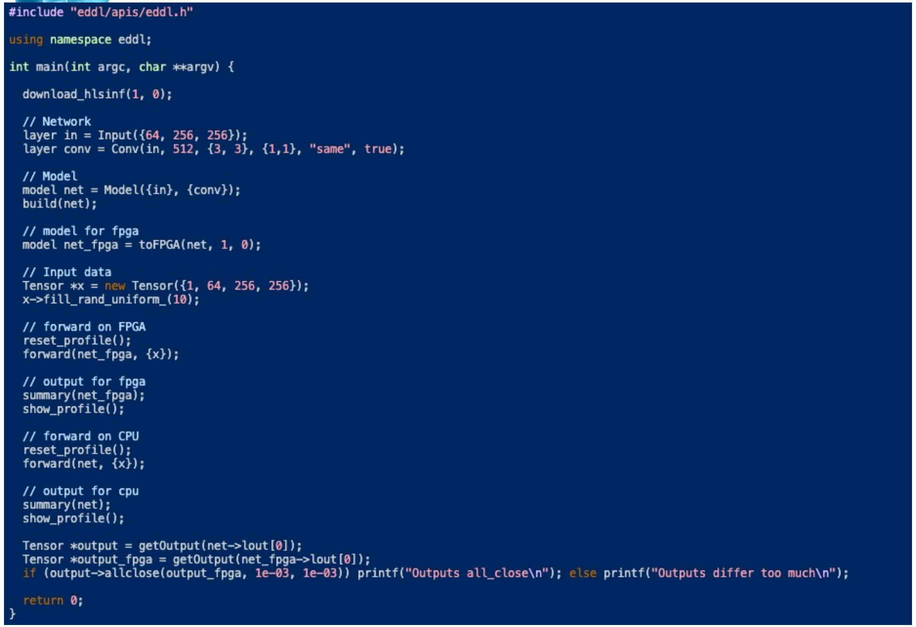
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

HLSinf – EDDL support

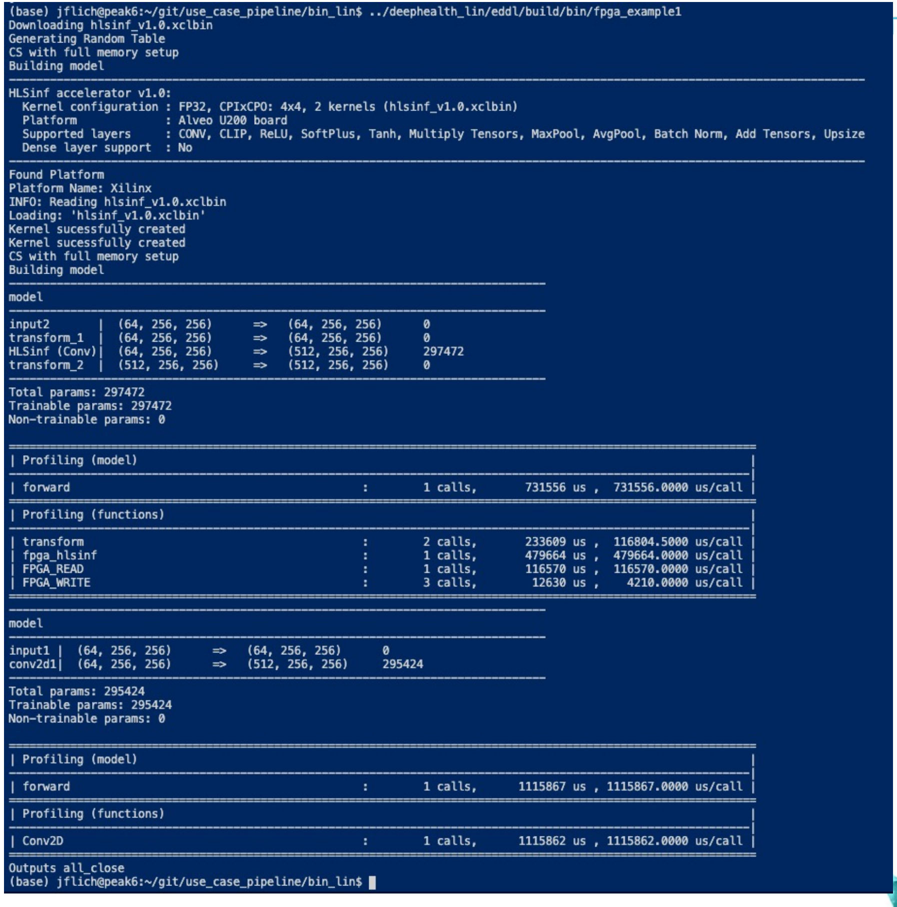
- OpenCL support to access any FPGA device
  - Hidden to the end user
- Model adaptation to HLSinf
  - Layer fusion
  - New EDDL layer for HLSinf (HLSinf)
- Tensor
  - Data format adaptations (Transformations)
  - Data precision conversions
    - FP->APINT, APINT->FP, …
  - All data transformations implemented through a new automatic EDDL layer (Transform)
- Timing statistics & profiling

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

HLSinf – EDDL example



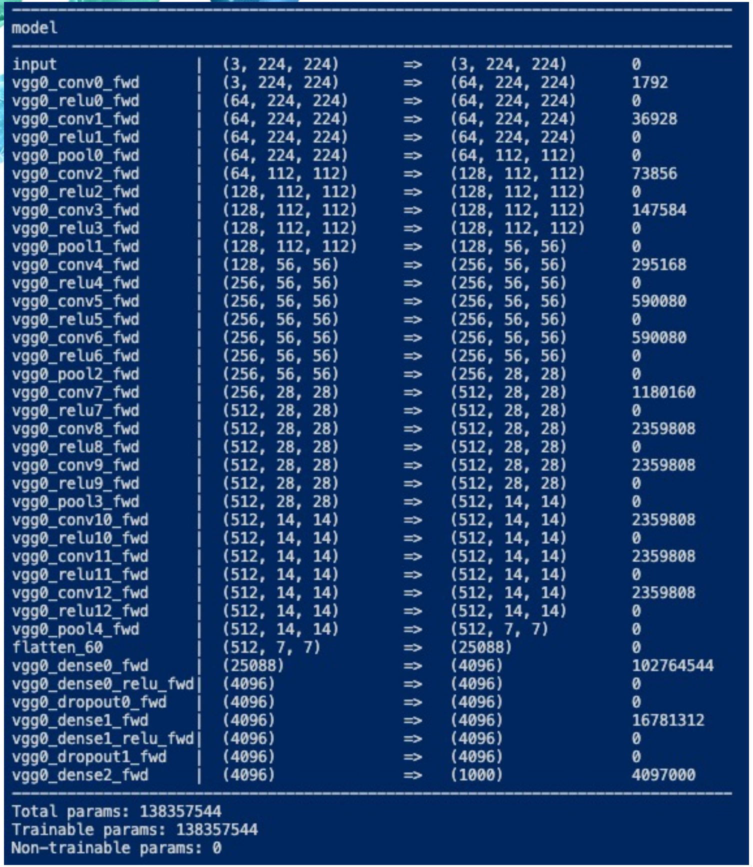
```cpp
#include "eddl/apis/eddl.h"

using namespace eddl;

int main(int argc, char **argv) {

  download_hlsinf(1, 0);

  // Network
  layer in = Input({64, 256, 256});
  layer conv = Conv(in, 512, {3, 3}, {1,1}, "same", true);

  // Model
  model net = Model({in}, {conv});
  build(net);

  // model for fpga
  model net_fpga = toFPGA(net, 1, 0);

  // Input data
  Tensor *x = new Tensor({1, 64, 256, 256});
  x->fill_rand_uniform_(10);

  // forward on FPGA
  reset_profile();
  forward(net_fpga, {x});

  // output for fpga
  summary(net_fpga);
  show_profile();

  // forward on CPU
  reset_profile();
  forward(net, {x});

  // output for cpu
  summary(net);
  show_profile();

  Tensor *output = getOutput(net->lout[0]);
  Tensor *output_fpga = getOutput(net_fpga->lout[0]);
  if (output->allclose(output_fpga, 1e-03, 1e-03)) printf("Outputs all_close\n"); else printf("Outputs differ too much\n");

  return 0;
}
```



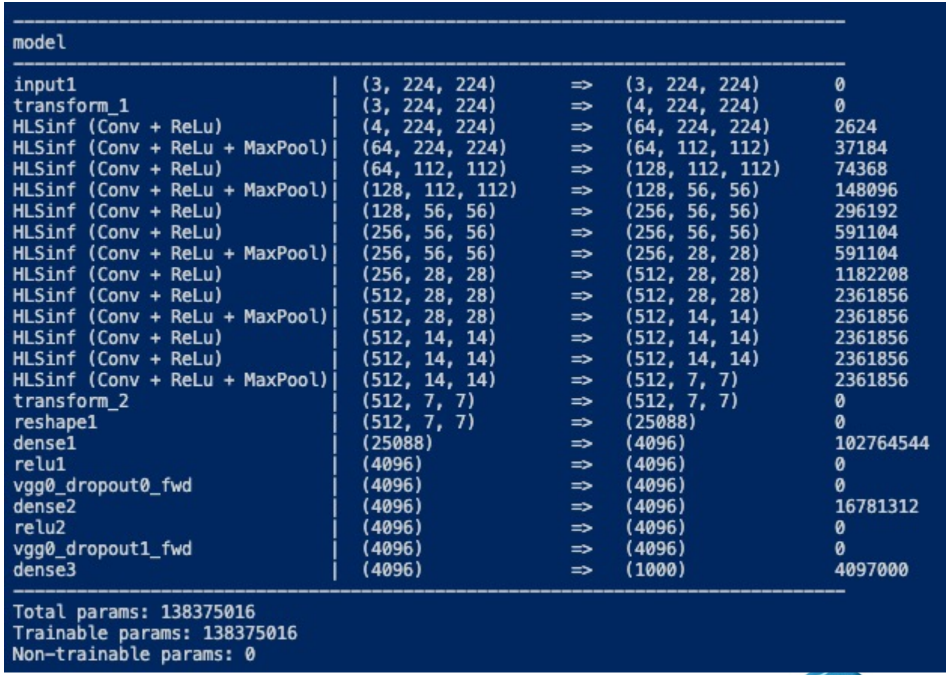
```
(base) jflich@peak6:~/git/use_case_pipeline/bin_lin$ ../deephealth_lin/eddl/build/bin/fpga_example1
Downloading hlsinf_v1.0.xclbin
Generating Random Table
CS with full memory setup
Building model
----------------------------------------------------------------------------------------------------------------------------
HLSinf accelerator v1.0:
  Kernel configuration : FP32, CPIxCPO: 4x4, 2 kernels (hlsinf_v1.0.xclbin)
  Platform             : Alveo U200 board
  Supported layers     : CONV, CLIP, ReLU, SoftPlus, Tanh, Multiply Tensors, MaxPool, AvgPool, Batch Norm, Add Tensors, Upsize
  Dense layer support  : No
----------------------------------------------------------------------------------------------------------------------------
Found Platform
Platform Name: Xilinx
INFO: Reading hlsinf_v1.0.xclbin
Loading: 'hlsinf_v1.0.xclbin'
Kernel sucessfully created
Kernel sucessfully created
CS with full memory setup
Building model
-------------------------------------------------------------------------------
model
-------------------------------------------------------------------------------
input2        |  (64, 256, 256)      =>   (64, 256, 256)      0
transform_1   |  (64, 256, 256)      =>   (64, 256, 256)      0
HLSinf (Conv)|  (64, 256, 256)      =>   (512, 256, 256)     297472
transform_2   |  (512, 256, 256)     =>   (512, 256, 256)     0
-------------------------------------------------------------------------------
Total params: 297472
Trainable params: 297472
Non-trainable params: 0

=================================================================================================================
| Profiling (model)                                                                                               |
-----------------------------------------------------------------------------------------------------------------|
| forward                                                  :        1 calls,       731556 us ,  731556.0000 us/call |
=================================================================================================================
| Profiling (functions)                                                                                           |
-----------------------------------------------------------------------------------------------------------------|
| transform                                                :        2 calls,       233609 us ,  116804.5000 us/call |
| fpga_hlsinf                                              :        1 calls,       479664 us ,  479664.0000 us/call |
| FPGA_READ                                                :        1 calls,       116570 us ,  116570.0000 us/call |
| FPGA_WRITE                                               :        3 calls,        12630 us ,    4210.0000 us/call |
=================================================================================================================
-------------------------------------------------------------------------------
model
-------------------------------------------------------------------------------
input1 |  (64, 256, 256)      =>   (64, 256, 256)      0
conv2d1|  (64, 256, 256)      =>   (512, 256, 256)     295424
-------------------------------------------------------------------------------
Total params: 295424
Trainable params: 295424
Non-trainable params: 0

=================================================================================================================
| Profiling (model)                                                                                               |
-----------------------------------------------------------------------------------------------------------------|
| forward                                                  :        1 calls,      1115867 us , 1115867.0000 us/call |
=================================================================================================================
| Profiling (functions)                                                                                           |
-----------------------------------------------------------------------------------------------------------------|
| Conv2D                                                   :        1 calls,      1115862 us , 1115862.0000 us/call |
=================================================================================================================
Outputs all_close
(base) jflich@peak6:~/git/use_case_pipeline/bin_lin$ 
```

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

HLSinf – EDDL example



```
-------------------------------------------------------------------------------
model
-------------------------------------------------------------------------------
input               | (3, 224, 224)       =>   (3, 224, 224)      0
vgg0_conv0_fwd      | (3, 224, 224)       =>   (64, 224, 224)     1792
vgg0_relu0_fwd      | (64, 224, 224)      =>   (64, 224, 224)     0
vgg0_conv1_fwd      | (64, 224, 224)      =>   (64, 224, 224)     36928
vgg0_relu1_fwd      | (64, 224, 224)      =>   (64, 224, 224)     0
vgg0_pool0_fwd      | (64, 224, 224)      =>   (64, 112, 112)     0
vgg0_conv2_fwd      | (64, 112, 112)      =>   (128, 112, 112)    73856
vgg0_relu2_fwd      | (128, 112, 112)     =>   (128, 112, 112)    0
vgg0_conv3_fwd      | (128, 112, 112)     =>   (128, 112, 112)    147584
vgg0_relu3_fwd      | (128, 112, 112)     =>   (128, 112, 112)    0
vgg0_pool1_fwd      | (128, 112, 112)     =>   (128, 56, 56)      0
vgg0_conv4_fwd      | (128, 56, 56)       =>   (256, 56, 56)      295168
vgg0_relu4_fwd      | (256, 56, 56)       =>   (256, 56, 56)      0
vgg0_conv5_fwd      | (256, 56, 56)       =>   (256, 56, 56)      590080
vgg0_relu5_fwd      | (256, 56, 56)       =>   (256, 56, 56)      0
vgg0_conv6_fwd      | (256, 56, 56)       =>   (256, 56, 56)      590080
vgg0_relu6_fwd      | (256, 56, 56)       =>   (256, 56, 56)      0
vgg0_pool2_fwd      | (256, 56, 56)       =>   (256, 28, 28)      0
vgg0_conv7_fwd      | (256, 28, 28)       =>   (512, 28, 28)      1180160
vgg0_relu7_fwd      | (512, 28, 28)       =>   (512, 28, 28)      0
vgg0_conv8_fwd      | (512, 28, 28)       =>   (512, 28, 28)      2359808
vgg0_relu8_fwd      | (512, 28, 28)       =>   (512, 28, 28)      0
vgg0_conv9_fwd      | (512, 28, 28)       =>   (512, 28, 28)      2359808
vgg0_relu9_fwd      | (512, 28, 28)       =>   (512, 28, 28)      0
vgg0_pool3_fwd      | (512, 28, 28)       =>   (512, 14, 14)      0
vgg0_conv10_fwd     | (512, 14, 14)       =>   (512, 14, 14)      2359808
vgg0_relu10_fwd     | (512, 14, 14)       =>   (512, 14, 14)      0
vgg0_conv11_fwd     | (512, 14, 14)       =>   (512, 14, 14)      2359808
vgg0_relu11_fwd     | (512, 14, 14)       =>   (512, 14, 14)      0
vgg0_conv12_fwd     | (512, 14, 14)       =>   (512, 14, 14)      2359808
vgg0_relu12_fwd     | (512, 14, 14)       =>   (512, 14, 14)      0
vgg0_pool4_fwd      | (512, 14, 14)       =>   (512, 7, 7)        0
flatten_60          | (512, 7, 7)         =>   (25088)            0
vgg0_dense0_fwd     | (25088)             =>   (4096)             102764544
vgg0_dense0_relu_fwd| (4096)              =>   (4096)             0
vgg0_dropout0_fwd   | (4096)              =>   (4096)             0
vgg0_dense1_fwd     | (4096)              =>   (4096)             16781312
vgg0_dense1_relu_fwd| (4096)              =>   (4096)             0
vgg0_dropout1_fwd   | (4096)              =>   (4096)             0
vgg0_dense2_fwd     | (4096)              =>   (1000)             4097000
-------------------------------------------------------------------------------
Total params: 138357544
Trainable params: 138357544
Non-trainable params: 0
```



```
-------------------------------------------------------------------------------
model
-------------------------------------------------------------------------------
input1                           | (3, 224, 224)     =>   (3, 224, 224)     0
transform_1                      | (3, 224, 224)     =>   (4, 224, 224)     0
HLSinf (Conv + ReLu)             | (4, 224, 224)     =>   (64, 224, 224)    2624
HLSinf (Conv + ReLu + MaxPool)   | (64, 224, 224)    =>   (64, 112, 112)    37184
HLSinf (Conv + ReLu)             | (64, 112, 112)    =>   (128, 112, 112)   74368
HLSinf (Conv + ReLu + MaxPool)   | (128, 112, 112)   =>   (128, 56, 56)     148096
HLSinf (Conv + ReLu)             | (128, 56, 56)     =>   (256, 56, 56)     296192
HLSinf (Conv + ReLu)             | (256, 56, 56)     =>   (256, 56, 56)     591104
HLSinf (Conv + ReLu + MaxPool)   | (256, 56, 56)     =>   (256, 28, 28)     591104
HLSinf (Conv + ReLu)             | (256, 28, 28)     =>   (512, 28, 28)     1182208
HLSinf (Conv + ReLu)             | (512, 28, 28)     =>   (512, 28, 28)     2361856
HLSinf (Conv + ReLu + MaxPool)   | (512, 28, 28)     =>   (512, 14, 14)     2361856
HLSinf (Conv + ReLu)             | (512, 14, 14)     =>   (512, 14, 14)     2361856
HLSinf (Conv + ReLu)             | (512, 14, 14)     =>   (512, 14, 14)     2361856
HLSinf (Conv + ReLu + MaxPool)   | (512, 14, 14)     =>   (512, 7, 7)       2361856
transform_2                      | (512, 7, 7)       =>   (512, 7, 7)       0
reshape1                         | (512, 7, 7)       =>   (25088)           0
dense1                           | (25088)           =>   (4096)            102764544
relu1                            | (4096)            =>   (4096)            0
vgg0_dropout0_fwd                | (4096)            =>   (4096)            0
dense2                           | (4096)            =>   (4096)            16781312
relu2                            | (4096)            =>   (4096)            0
vgg0_dropout1_fwd                | (4096)            =>   (4096)            0
dense3                           | (4096)            =>   (1000)            4097000
-------------------------------------------------------------------------------
Total params: 138375016
Trainable params: 138375016
Non-trainable params: 0
```

```
Model_fpga = toFPGA(model, 1, 0);
```

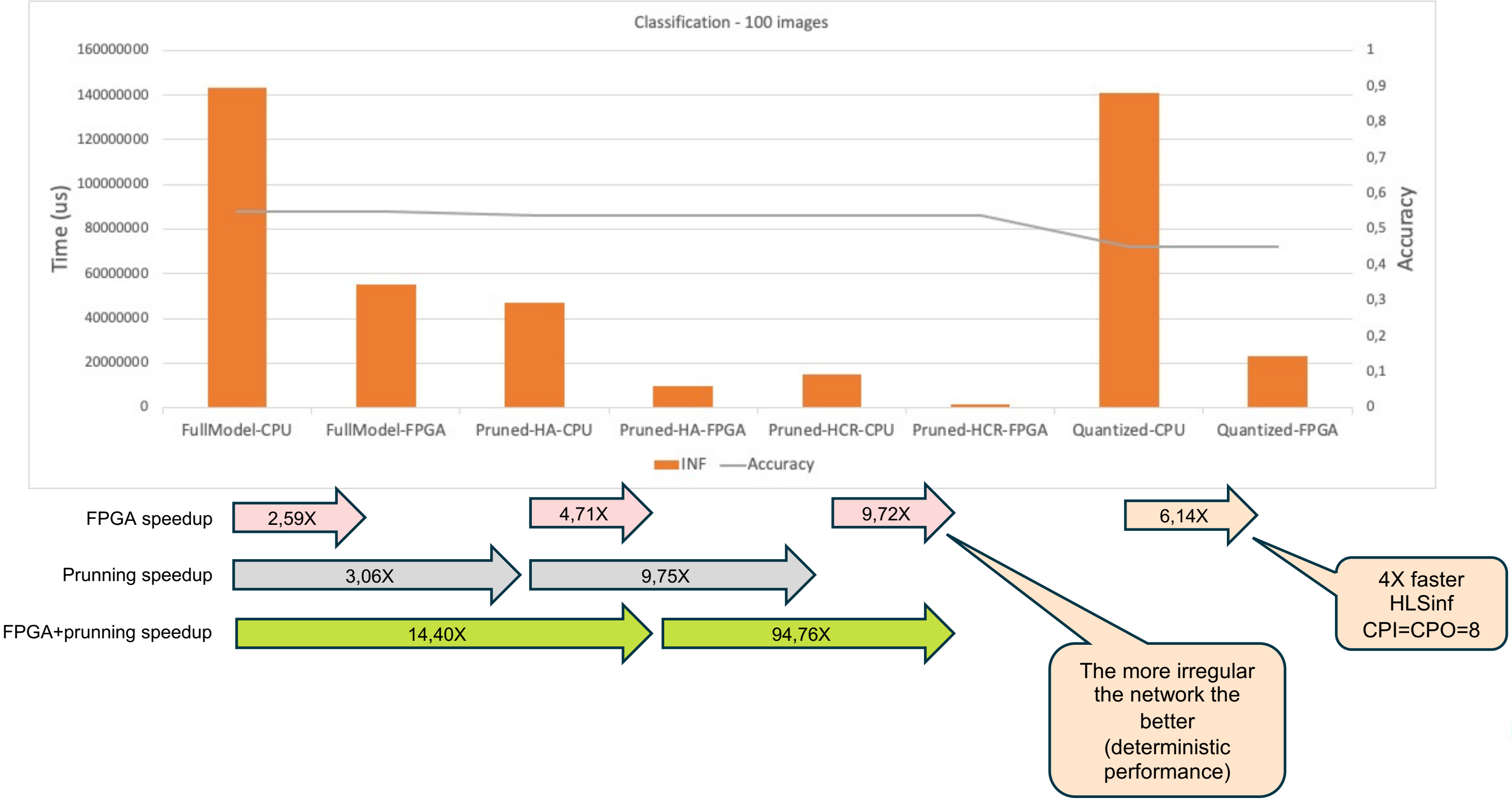
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Results

- Skin cancer classification
  - Full model (VGG16) – FP32
  - Pruned (compressed) - High Accuracy – FP32
  - Pruned (compressed) - High Compression Rate – FP32
  - Quantized – Weights (APINT<8>), Bias and Activations (APINT<32>)
- Skin cancer segmentation
  - Full model (SegNet) – FP32
  - Pruned (compressed) – FP32
- All models (FP32)
  - VGG16, VGG16 + BN, VGG19, VGG19 + BN, ResNet18, ResNet34, ResNet50, ResNet101, ResNet152, DenseNet121
- COVID19 (FP32)
  - DeepHealth use case, Kaggle use case (VGG16)
- All cases compared to high-end CPU with EDDL with HPC option enabled
- CPI=CPO=4 for FP32, CPI=CPO=8 for APINT

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Results (skin cancer classification)

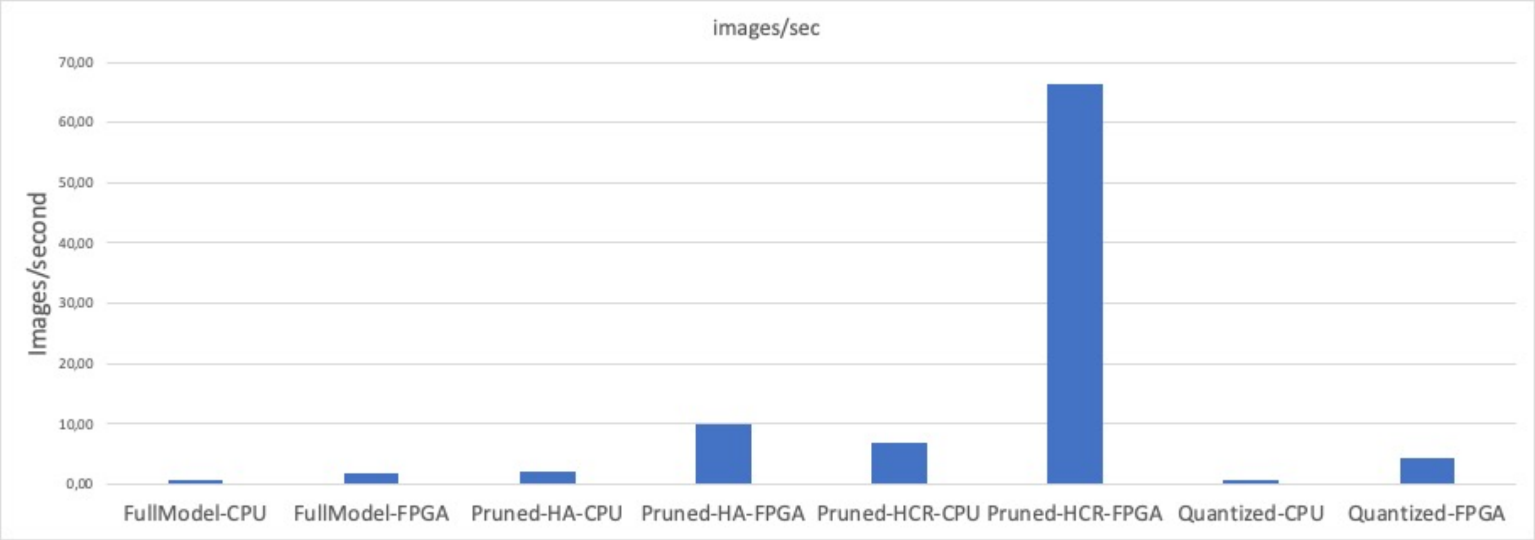


The more irregular the network the better (deterministic performance)

4X faster HLSinf CPI=CPO=8

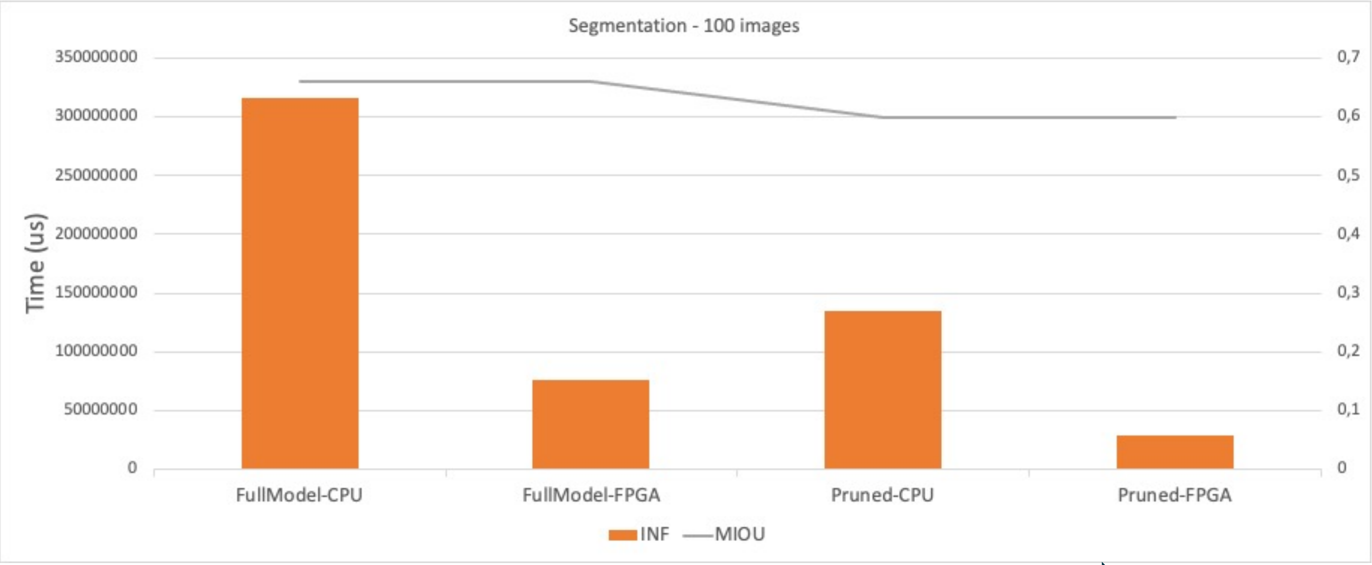
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

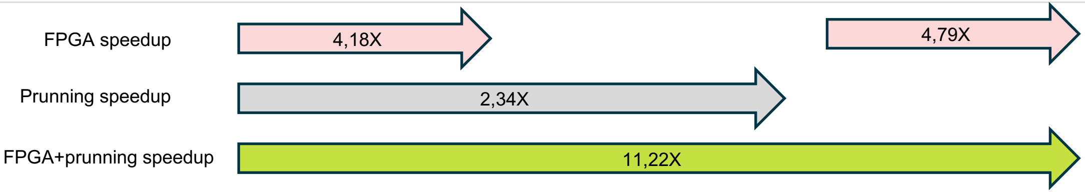
Results (skin cancer classification)



The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Results (skin cancer segmentation)




FPGA speedup

4,18X

4,79X

Prunning speedup

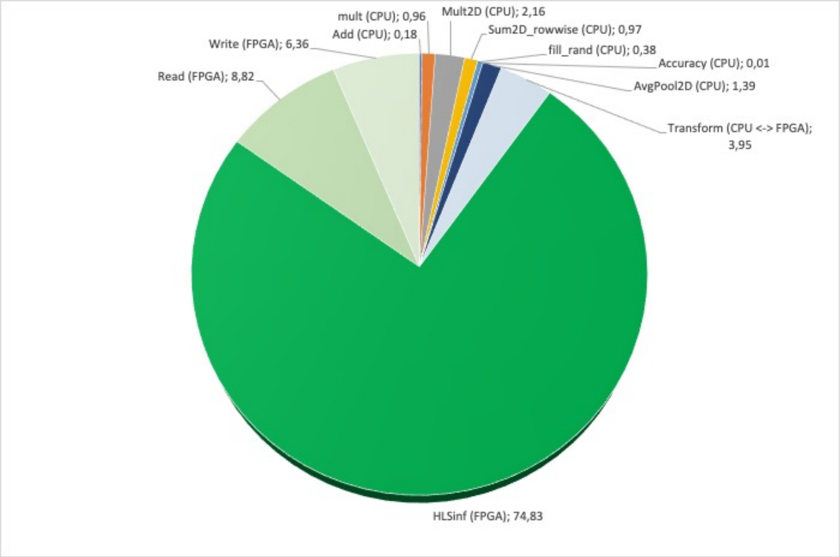
2,34X

FPGA+prunning speedup

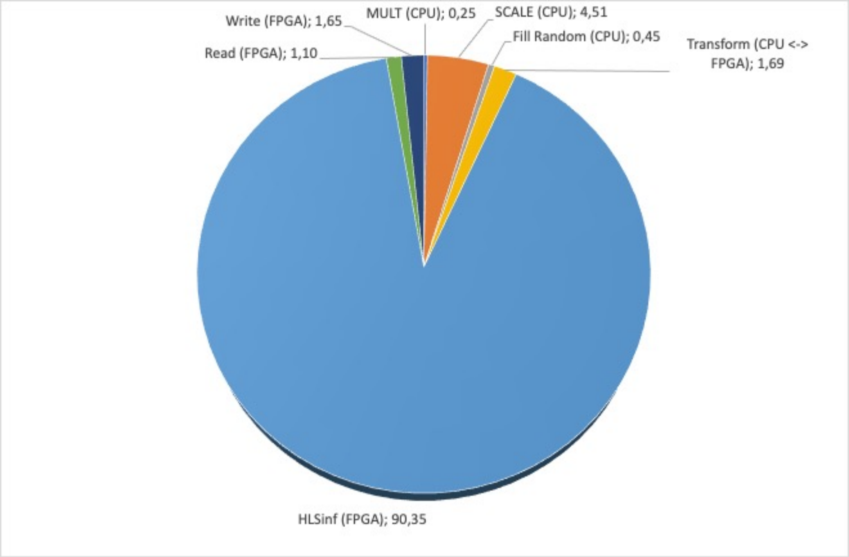
11,22X

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Results (classif. Vs segm.)



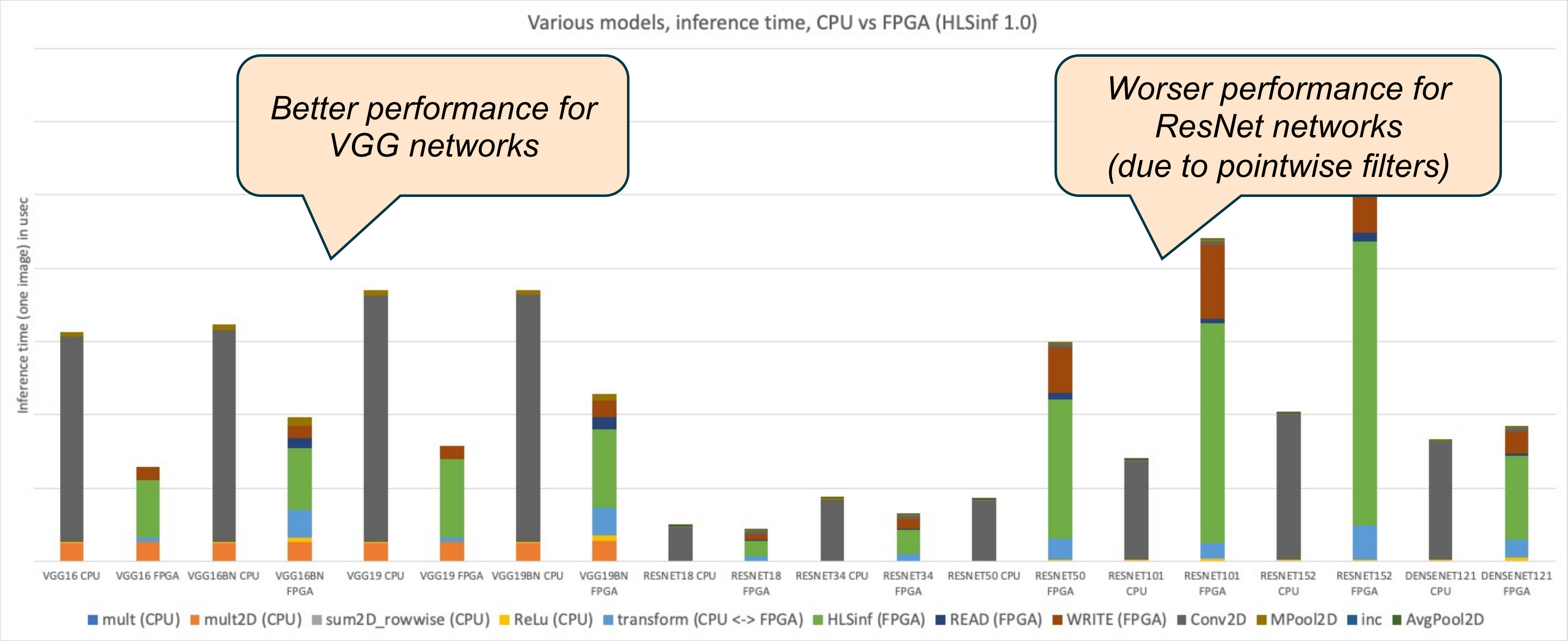
classification



segmentation

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Results (all models)

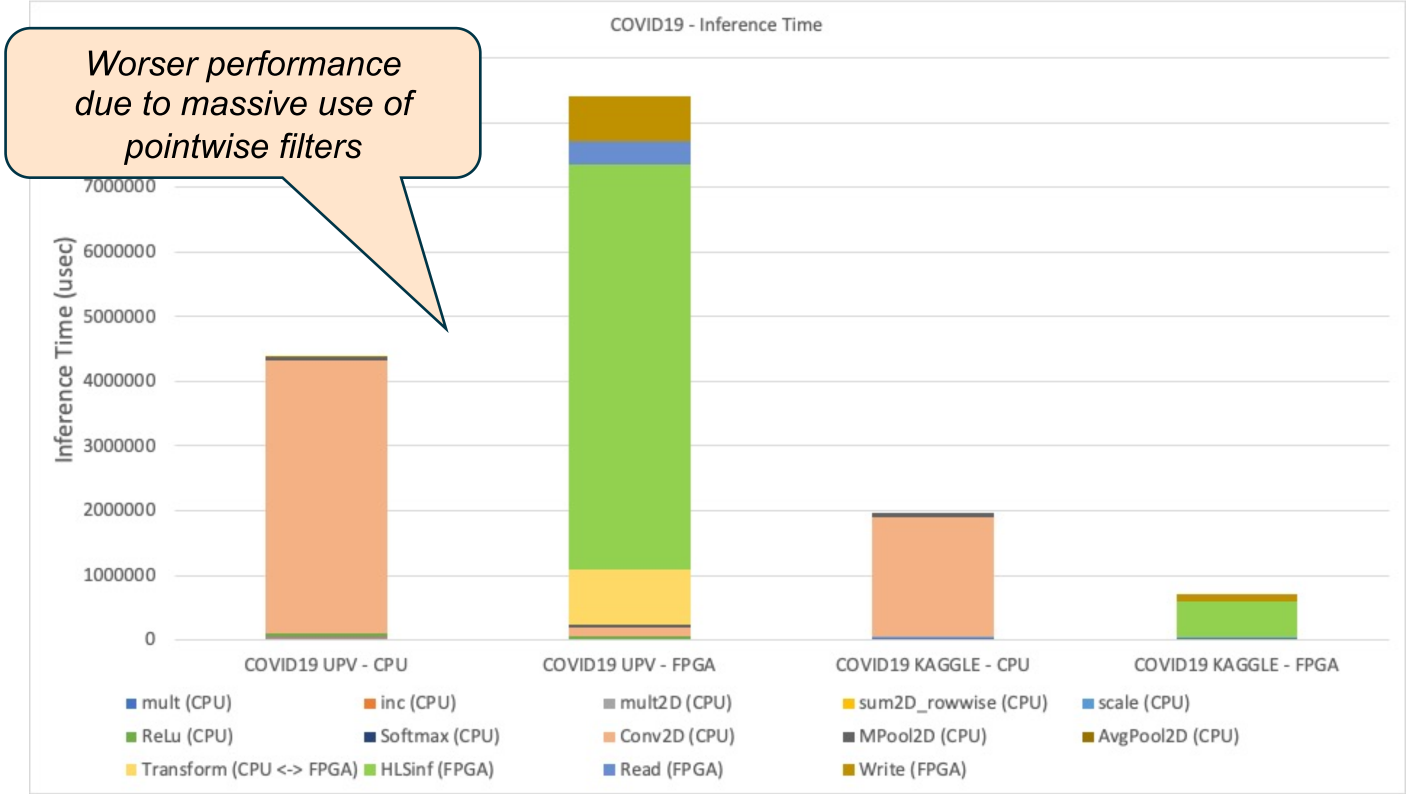


Better performance for VGG networks

Worser performance for ResNet networks (due to pointwise filters)

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Results (COVID19)



Worser performance due to massive use of pointwise filters

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Conclusions

- EDDL has been provisioned with FPGA support for inference processes
- FPGA use completely hidden to the end user
- Use of Open Source HLSinf accelerator
  - Customizable in size, data precission and functionality
  - Compression and Quantization support
  - Highly configurable
- Improves CPU inference time while achieving low power consumption profiles
- ***Full native integration of FPGAs in EDDL***

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

DEEPHEALTH

Thank you!

José Flich (jflich@disca.upv.es)

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.