Kernel Merging for Throughput-Oriented Accelerator Generation

IMPACT'23, Workshop on Polyhedral Compilation Techniques
January 16, 2023,
Toulouse, France

Nicolas Derumigny,
Colorado State University
Inria

Louis-Noël Pouchet,
Colorado State University

Fabrice Rastello,
Inria

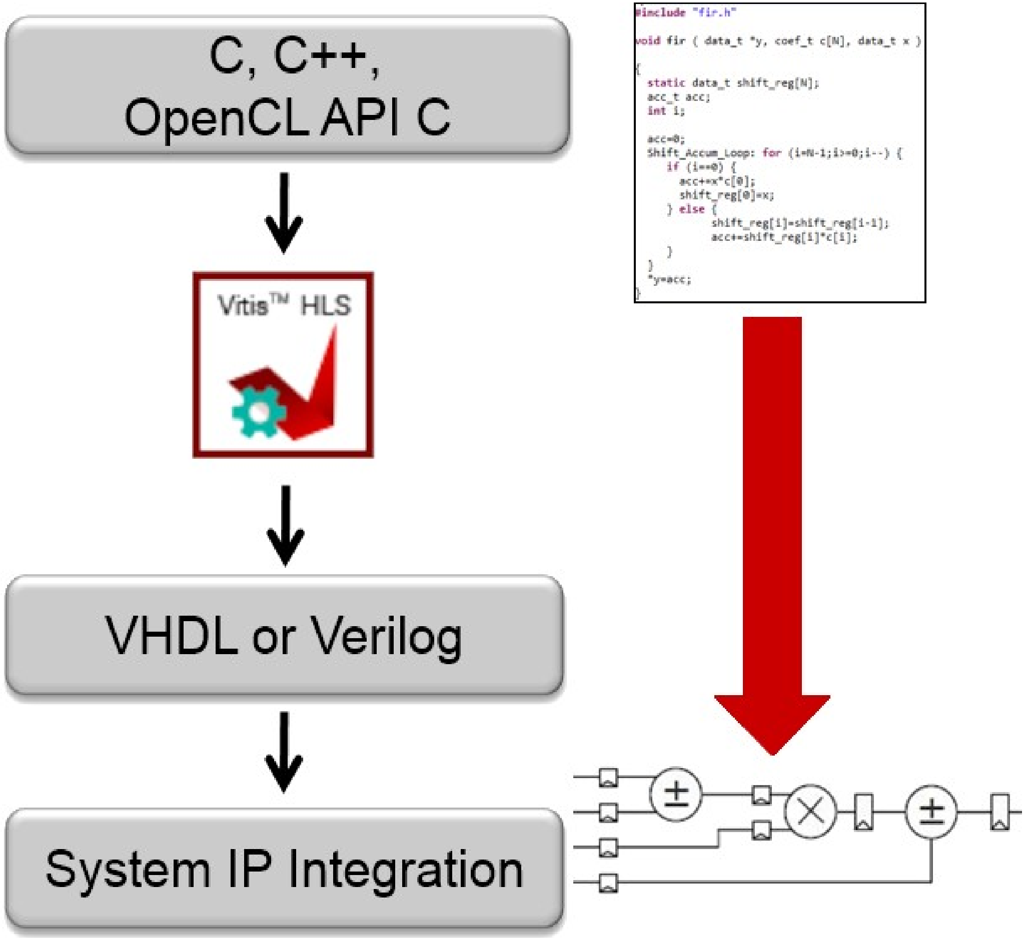
Introduction

- Context
- Motivation
- Overview

Context

C Code is used to design hardware accelerators

- **Architecture Layout is specified in C++**
  - High-Level Synthesis
  - Generates Hardware Description Langage:
    - **VHDL / Verilog**
- **Target dedicated chips**
  - FPGA (reprogrammable)
  - ASIC (fixed)
- **Automatized Generation of Hardware Design**
  - Specialised Accelerators (IP)
  - Target HPC applications
  - Evaluation on linear algebra and correlation
    - **On par with state of the art accelerators (GEMM, FP32)**
      - ScaleHLS[1]: 0.393 Op/Cycle/DSP
      - Us: 0.277 Op/Cycle/DSP



[1] Hanchen Ye, Cong Hao, Jianyi Cheng, Hyunmin Jeong, Jack Huang, Stephen Neuendorffer, and Deming Chen. ScaleHLS: A New Scalable High-Level Synthesis Framework on Multi-Level Intermediate Representation. In 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA).

Motivation #1

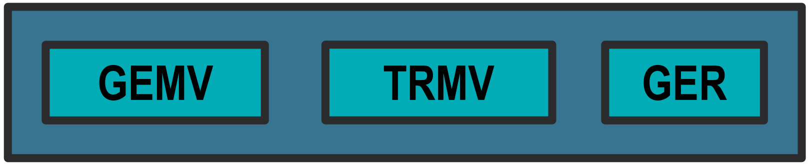
What is the area of a BLAS2 accelerator ?

- **Operations to support:**
  - Matrix-vector product (GEMV)

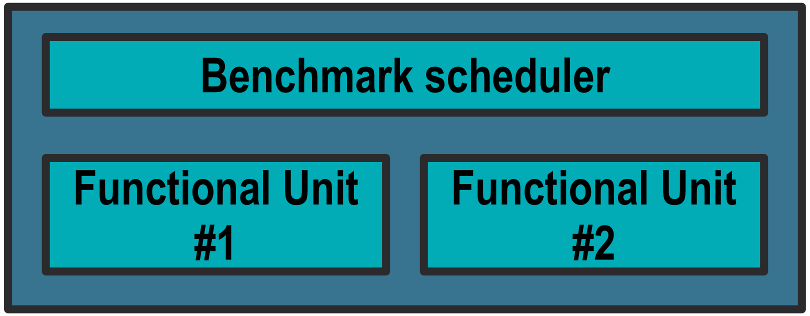
    y := alpha*A*x + beta*y
  - Triangular matrix-vector (TRMV)

    y := alpha*A*x + beta*y   with A triangular
  - Rank 1 operation (GER)

    A := alpha*x*y' + A
- **Using a juxtaposition of Fixed Functions (FF):**
  - FF with Maximum Throughput: **24 DSP**
  - FF with Max Sharing: **12 DSP**
- **Resulting design is:**

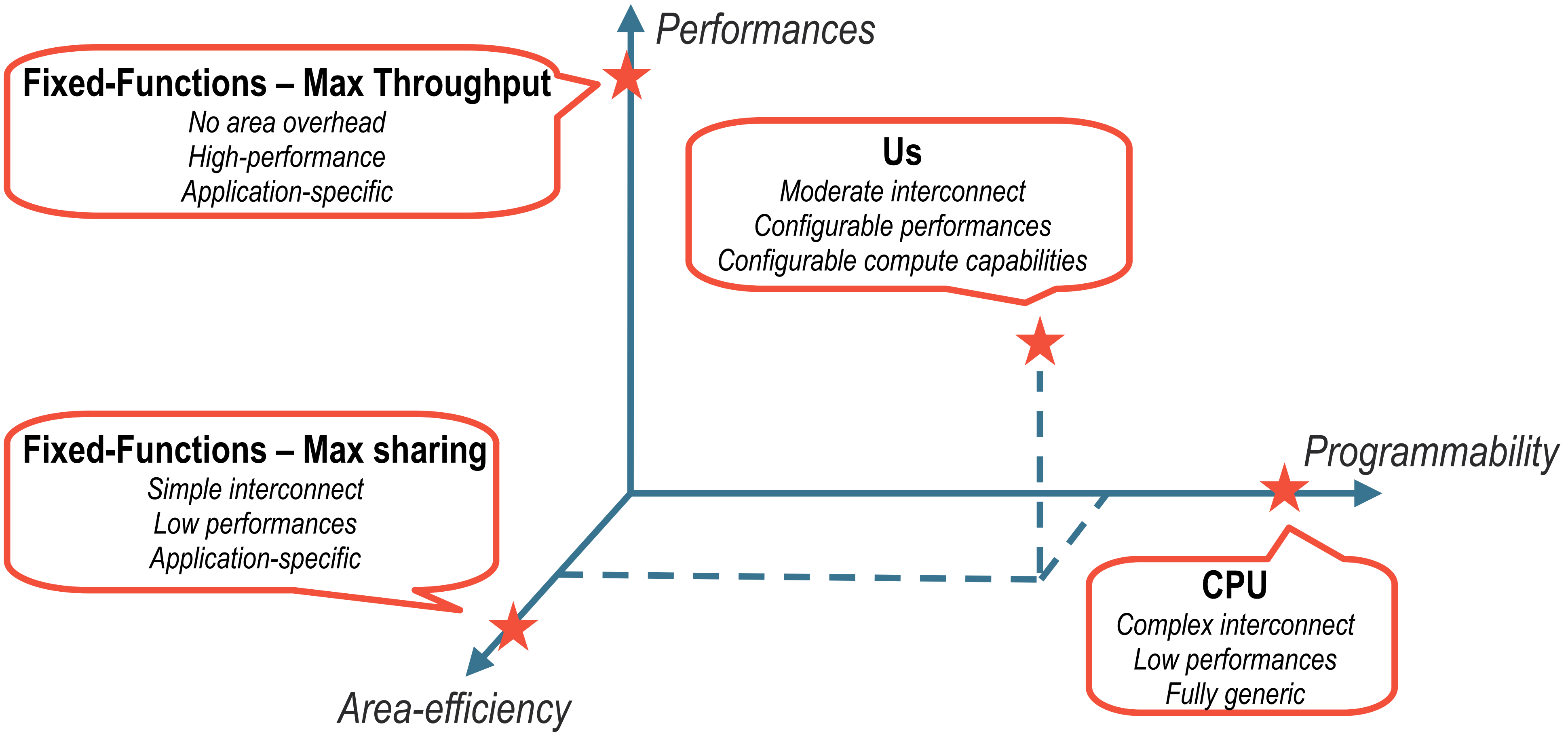


- **Area ~= Area_{GEMV} + Area_{TRMV} + Area_{GER}**
  - **No interconnect** between FFs
  - No resource reuse between FFs => **No sharing**
- **Reuse components beween FFs?**
  - Maximise **resource sharing** between FFs
  - Create Functional Units: **FU**
  - BLAS 2 + BLAS 3: **6 DSP**
- **Proposed design:**



Motivation #2

Reuse common compute patterns accross a set of applications



Overview

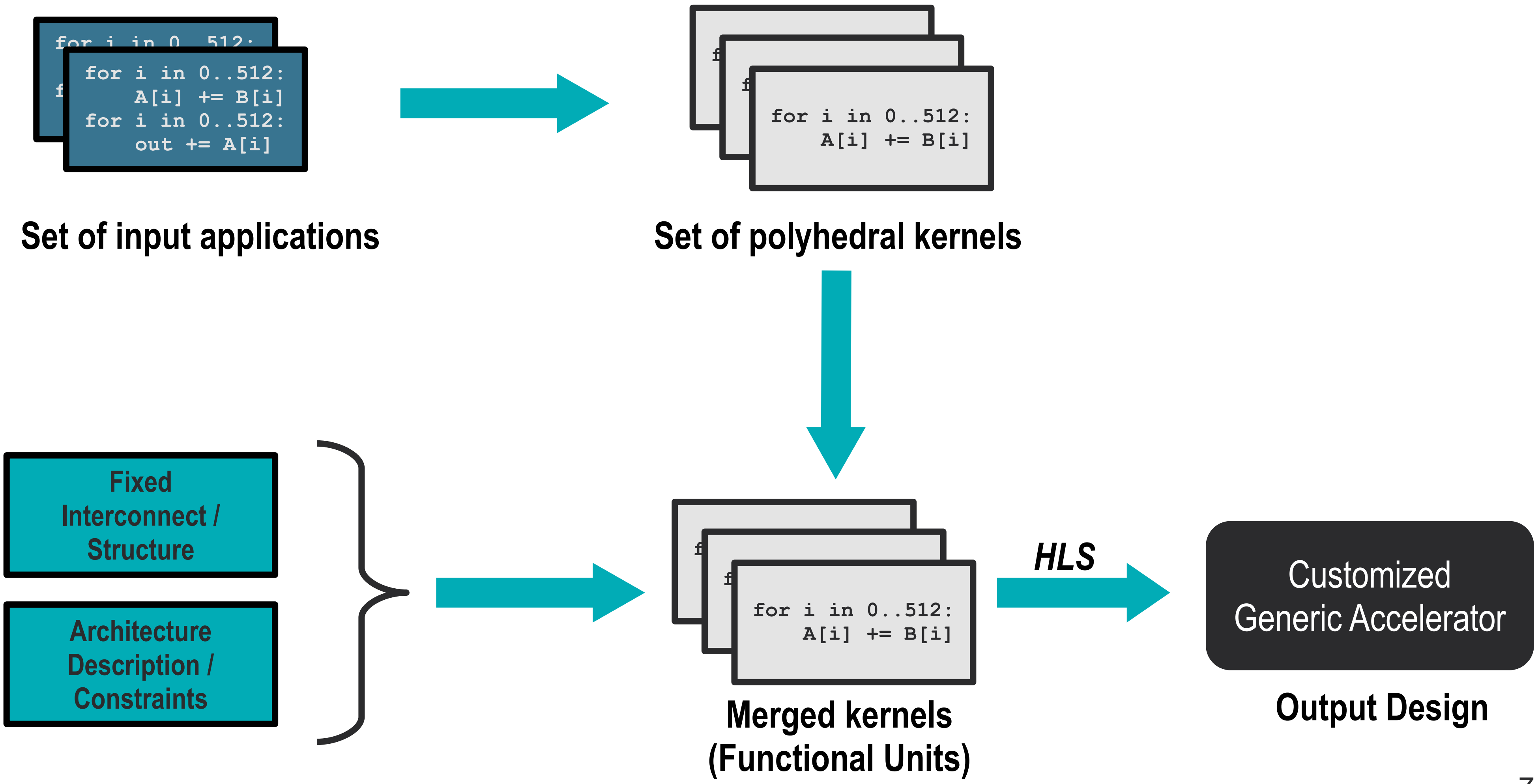
From common compute patterns to hardware generation

- **Creation of merged Fixed-Functions**
  - **Functional Units**
  - Polyhedral decomposition of application into **kernels**
  - Detection of identical **operators**
    - **Shareable components**

- **Implementation of hardware FUs**
  - Data routing through FUs
    - **Operator are shared...**
    - **...but control-flow remains kernel-specific**
  - Code generation
    - **Single merged/flattened loop**
    - **Hardware constraints**
      - C specification of the compute pipeline
      - Requires HLS-specific annotations

- **Integration into a real-life design**
  - Compromises
    - **Number of FU (~= performances) vs area**
    - **Supported kernels per FU vs area**
  - Interconnect
    - **Configuration of the accelerator**
    - **Data communications**
    - **On-accelerator data organisation**
    - **Scheduling kernels on FUs**
    - **=> Need of dedicated control units**

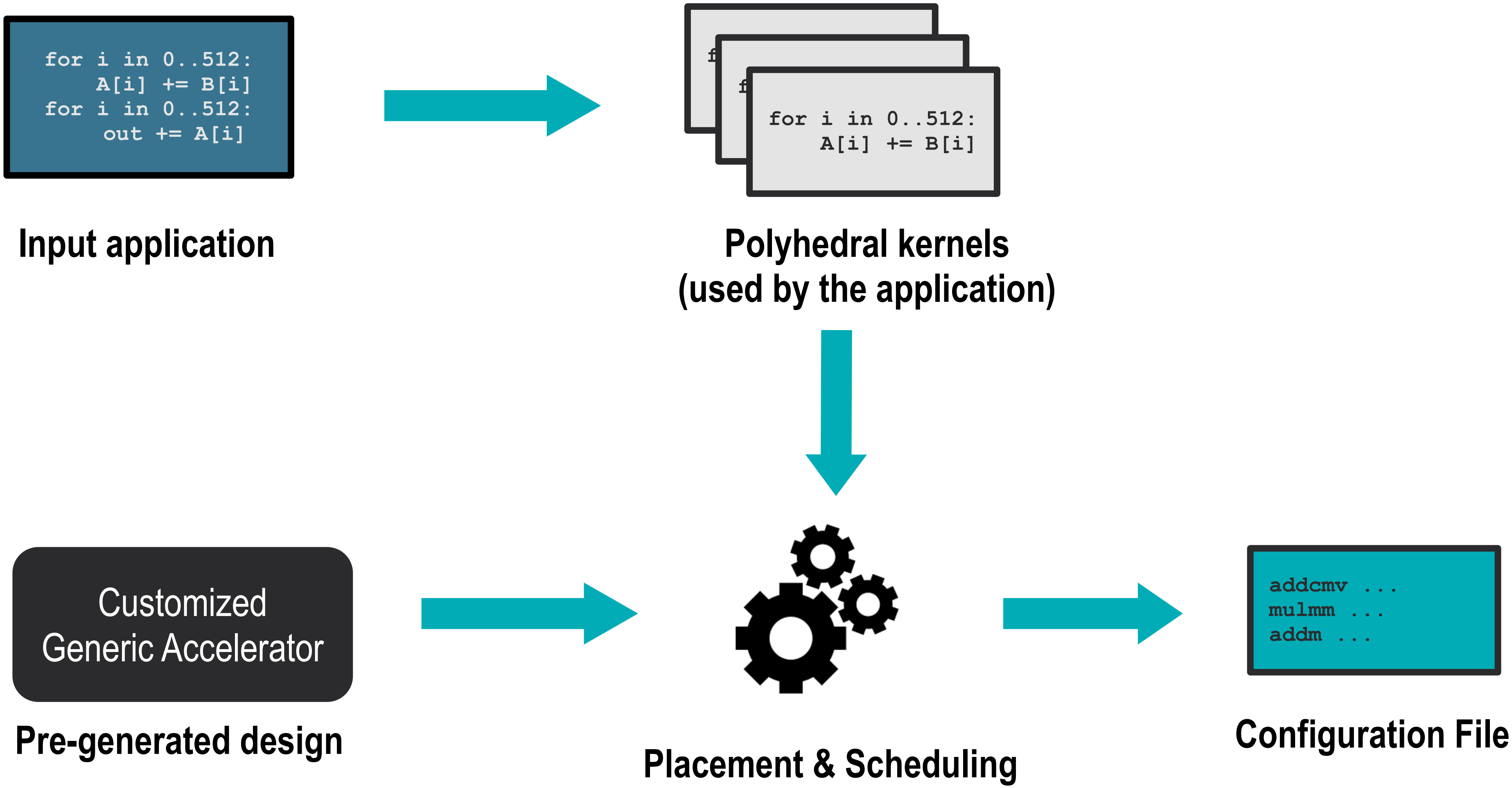
Flow of the work



Technical Details

- User Perspective
- Automatization
- Kernel Merging
- Structure of the accelerator
- Profitability criteria

User perspective



User perspective

```
for i in 0..512:
    A[i] += B[i]
for i in 0..512:
    out += A[i]
```

Input application

GEMV

```
for i in 0..N:
  y[i] *= beta

for j in 0..N:
    tmp = alpha * x[j]
    for i in 0..N:
        y[i] += A[i][j] * tmp
```

Customized Generic Accelerator

Pre-generated design

Placement & Scheduling

ration File

User perspective

mulsv

```
for i in 0..N:
  y[i] *= beta
```

Input application

GEMV

```
for i in 0..N:
  y[i] *= beta

for j in 0..N:
    tmp = alpha * x[j]
    for i in 0..N:
       y[i] += A[i][j] * tmp
```

mulsv

```
for j in 0..N:
  x[j] *= alpha
```

addv

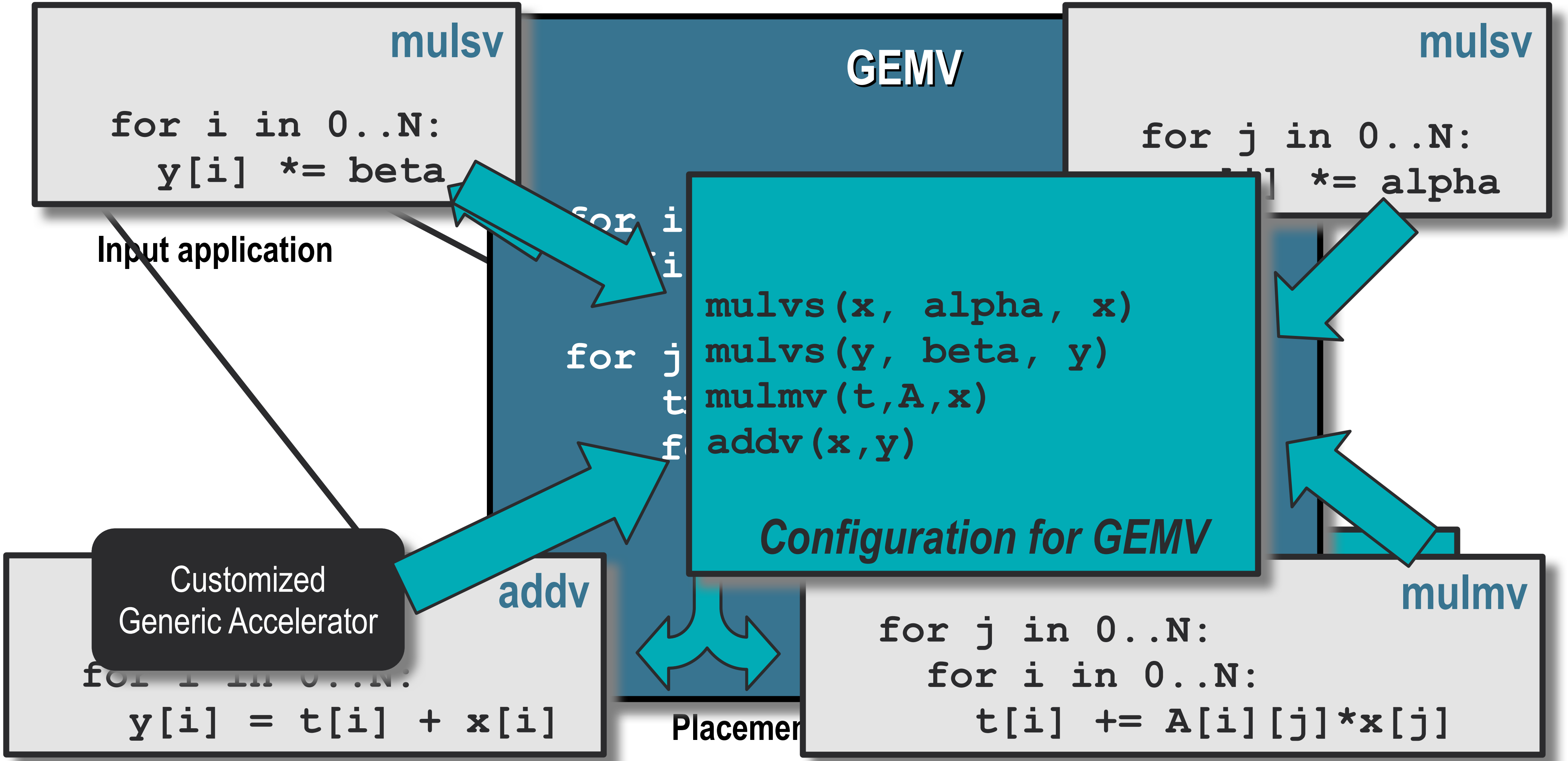
```
for i in 0..N:
  y[i] = t[i] + x[i]
```

Placemen

mulmv

```
for j in 0..N:
  for i in 0..N:
    t[i] += A[i][j]*x[j]
```

User perspective



Automatization

- **Polyhedral detection of kernels (Work-in-Progress)**
  - Decomposition of loop bodies in SSA 3-address code
  - Loop fission
- **Kernel merging**
  - Iteration domain extension
  - Loop fusion
- **Generation of the accelerator**
  - Fixed structure
    - **One main scheduling loop**
      - Flattenned version of the merged kernels loop nest
    - **Variable loop bound**
      - ~ execution time of the kernel
    - **Configurable number of FUs**
- **Compilation of the applications to the accelerator FUs**
  - ASAP scheduling on the FU
  - Greedy placement
    - **Longest kernels are scheduled in first**

Kernel Merging

mulsv

```
for i in 0..N:
  y[i] *= beta
```

mulmv

```
for i in 0..N:
   for j in 0..N:
      y[i] += A[i][k]*x[k]
```

Extension of the iteration domain

```
for j in 0..N:
   for i in 0..N:
      if (j==0):
         y[i] *= beta
```

Kernel Merging

mulsv

```
for i in 0..N:
  y[i] *= beta
```

mulmv

```
for i in 0..N:
  for j in 0..N:
    y[i] += A[i][k]*x[k]
```

Extension of the iteration domain

```
for j in 0..N:
  for i in 0..N:
    if (j==0):
      y[i] *= beta
```

Loop fusion

mulsv U_{fused} mulmv

```
for j in 0..N:
  for i in 0..N:
    if (kernel == mulmv):
      y[i] += A[i][j]*x[j]
    if (kernel == mulsv and j==0):
      y[i] *= beta
```

Kernel Merging

mulsv

```
for i in 0..N:
  y[i] *= beta
```

mulmv

```
for i in 0..N:
  for j in 0..N:
    y[i] += A[i][k]*x[k]
```

Extension of the iteration domain

```
for j in 0..N:
  for i in 0..N:
    if (j==0):
      y[i] *= beta
```

Loop fusion

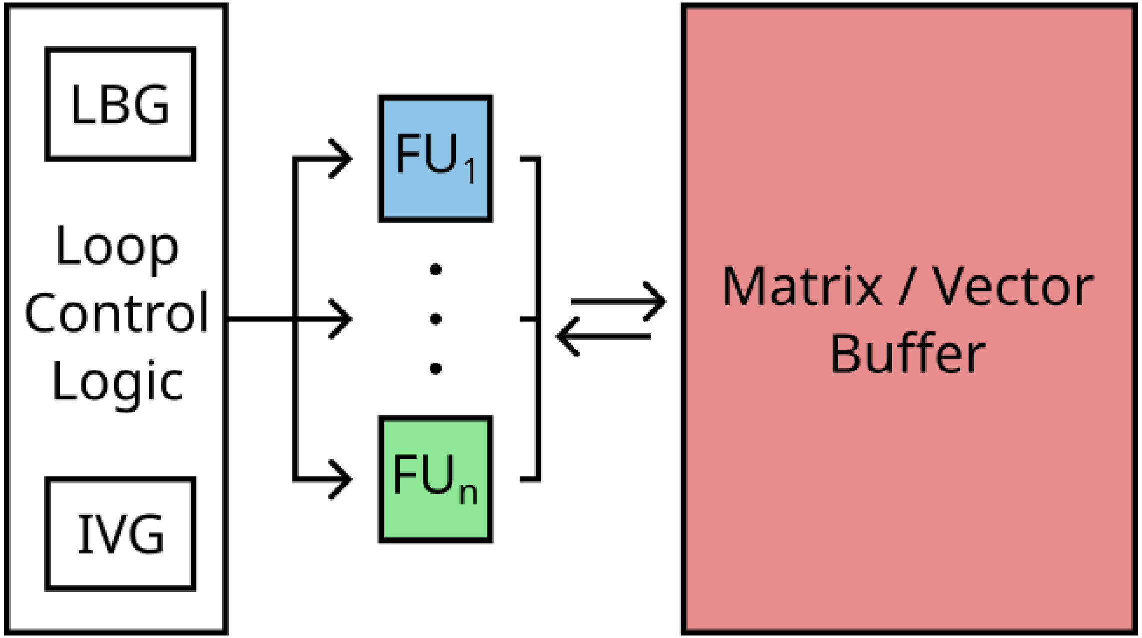
mulsv $\mathrm{U}_{\text{fused}}$ mulmv

```
for j in 0..N:
  for i in 0..N:
    if (kernel == mulmv):
      y[i] += A[i][j]*x[j]
    if (kernel == mulsv and j==0):
      y[i] *= beta
```

- **Minimal dependence distance**
  - **Hardware constraint:** a value computed by the pipeline must exit the pipeline before being used (no bypass)
  - The minimal Read-after-Write distance must be greater that the **pipeline latency**
  - In practice:
    - **Ensure that the inner-most loop is synchronisation-free**

Structure of the accelerator

- **Fonctional Units (FU)**
  - Execute elementary operations of the applications
  - Result of kernel merging
- **Loop Control Logic**
  - Schedule the kernels on the FUs
  - **Loop Bound Generator (LBG)**
    - **Generates the trip count of the execution loop**
  - **Iteration Vector Generator (IVG)**
    - **Generates the values of the iteration vector**
- **Matrix / Vector Buffer**
  - Scratchpad containing accessible data
  - Transfered to main memory before and after execution



- **Degrees of freedom**
  - Compute capabilities of each FU
    - **Add, Mul, div, sqrt, abs, ...**
  - Number of FUs

Profitability Criteria

What is the best FU topology (functionnalities/replications) ?

- **Maximise performance under area constraint**

$$\text{minimize} \quad \sum_{K \in Kernels} \lceil \#calls(K) / \#FU(K) \rceil * card(\mathcal{D}_K) * IL_K$$

$$\text{subject to} \quad \sum_{i \in FUs} Area(FU_i) * \#FU_i < max\_area$$

- **With:**
  - $Kernels$ the set of supported kernels
  - IL_K the iteration domain of kernel K
  - $\mathcal{D}_K$ the iteration latency of kernel K
- **Instantiate FUs to match the proportion of their execution time in the input workload**
  - **Do not replicate FUs that account for a low part of the workload execution time**

Profitability Criteria

What is the best FU topology (functionnalities/replications) ?

- **Maximise performance under area constraint**

$$\begin{aligned} \text{minimize} \quad & \sum_{K \in Kernels} \lceil \#calls(K) / \#FU(K) \rceil * card(\mathcal{D}_K) * IL_K \\ \text{subject to} \quad & \sum_{i \in FUs} Area(FU_i) * \#FU_i < max\_area \end{aligned}$$

- **Requires knowledge of FU resource consumption**
- **Do not take glue logic into account**
  - Constant overhead
  - Does not use DSP
- **Specialised for either one application, or a family of applications**

Evaluation

- Supported Primitives
- Benchmark: Linear Algebra-Generic Accelerator
- Batching Linear Algebra Computation?
- Benchmark: Correlation-Generic Accelerator

Supported Kernels

- **Primitives extracted from**
  - Linear algebra (BLAS) level 2 and 3
  - Polybench's Correlation
  - 31 different kernels supported
- **Evaluation on 2 different accelerators**
  - FP16 data type

	Number of operators				Nb. of FU
	$a \pm b$	$a * b$	a/b	$\sqrt{a}$	
BLAS	2	1	0	0	2
CORR	3	3	1	1	4

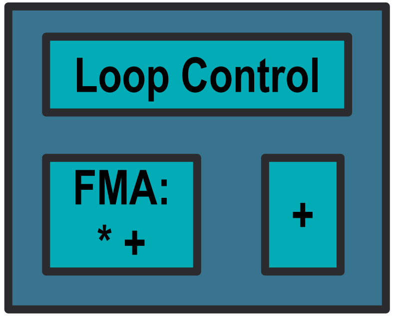
- **Hardware primitives:**
  - Add, mul, div, sqrt
  - Different routing/iteration spaces combination creates 31 kernels

Kernel	Description	Op.	LA-GA	CORR-GA
`noop`	Do nothing	None	✓	✓
`mulmm`	Matrix-matrix multiplication	$\pm$ and $*$	✓	✓
`mulmv`	Matrix-vector multiplication	$\pm$ and $*$	✓	✓
`multrmm`	Triangular matrix-matrix multiplication	$\pm$ and $*$	✓	
`multrmv`	Triangular matrix-vector multiplication	$\pm$ and $*$	✓	
`mulsm`	Scalar-matrix multiplication	$*$	✓	✓
`multrsm`	Scalar-triangular matrix multiplication	$*$	✓	
`mulsv`	Scalar-vector multiplication	$*$	✓	✓
`muls`	Scalar-scalar multiplication	$*$	✓	✓
`trm`	Matrix transposition	None	✓	✓
`addm`	Matrix addition	$\pm$	✓	✓
`addv`	Vector addition	$\pm$	✓	✓
`adds`	Scalar addition	$\pm$	✓	✓
`addtrm`	Triangular matrix addition	$\pm$	✓	
`subm`	Matrix subtraction	$\pm$	✓	✓
`subcmv`	Column-wise matrix subtraction	$\pm$	✓	✓
`subv`	Vector subtraction	$\pm$	✓	✓
`subs`	Scalar subtraction	$\pm$	✓	✓
`pmulm`	Point-wise matrix multiplication	$*$	✓	✓
`pmulv`	Point-wise vector multiplication	$*$	✓	✓
`oprodv`	Outer (vector) product	$*$	✓	✓
`sqrtv`	Point-wise vector square root	$\sqrt{\cdot}$		✓
`sqrts`	Scalar square root	$\sqrt{\cdot}$		✓
`accsumcm`	Columns-wise accumulation of a matrix	$\pm$	✓	✓
`cutminv`	Vector round to 1 low values	None	✓	✓
`divms`	Pointwise division of matrices	$/$		✓
`divvs`	Pointwise division of vectors	$/$		✓
`divcmv`	Point-wise division with column-wise value	$/$		✓
`set0m`	Initialisation of a matrix to 0	None	✓	✓
`setidm`	Initialisation of a matrix to Id	None	✓	✓
`setd1`	Initialisation of the diagonal of a matrix to 1	None	✓	✓

Benchmark: Linear Algebra-Generic Accelerator

Bench name	Arithmetic expression	Execution Time (cycles) MS	MT	LA-GA	FLOP/C/DSP MS	MT	LA-GA
SCALE	$A = \alpha \cdot A + B$	5572	2059	8258	0.368	0.497	0.165
GEMV	$y = \alpha \cdot A \cdot x + \beta \cdot y$	4553	2126	4396	0.457	0.391	0.315
TRMV	$y = \alpha \cdot A \cdot x + \beta \cdot y$	2339	2435	2380	0.458	0.293	0.300
GER	$A = \alpha \cdot x \cdot y^t + A$	4738	2057	8343	0.436	0.401	0.165
GEMM	$C = \alpha \cdot A \cdot B + \beta \cdot C$	307586	134018	274540	0.433	0.397	0.323
TRMM	$C = \alpha \cdot A \cdot B + \beta \cdot C$	149696	155840	145516	0.458	0.293	0.314

- **Accelerator: Linear Algebra-GA**



- **Raw Performance similar to Max Sharing designs**
- **Performance-per-area similar to Max Throughput designs**
  - 4/6 benchmarks
  - Cases when MS operation schedule is identical to GA
- **But semi-generic!**

Batching Linear Algebra Computation?

- **Batching with factor 5**
- **Better use of the LA-GA**
  - More (exploited) parallelism opportunities
  - Better occupancy of the LA-GA FUs
- **Execution time comparable to MS on every benchmark**

Bench name	Exec. Time (cycles)		
	MS	MT	CORR-GA
SCALEx5	27860	10295	24726
GEMVx5	22765	10630	21544
TRMVx5	11695	12175	11379
GERx5	23690	10285	41279
GEMMx5	1537930	670090	1356136
TRMMx5	748480	779200	711016

Batching Linear Algebra Computation?

- **Batching with factor 5**
- **Better use of the LA-GA**
  - More (exploited) parallelism opportunities
  - Better occupancy of the LA-GA FUs
- **Execution time comparable to MS on every benchmark**

Bench name	Exec. Time (cycles) MS	MT	CORR-GA
SCALEx5	27860	10295	24726
GEMVx5	22765	10630	21544
TRMVx5	11695	12175	11379
GERx5	23690	10285	41279
GEMMx5	1537930	670090	1356136
TRMMx5	748480	779200	711016

What about other works?

Batching Linear Algebra Computation?

- **Batching with factor 5**
- **Better use of the LA-GA**
  - More (exploited) parallelism opportunities
  - Better occupancy of the LA-GA FUs
- **Execution time comparable to MS on every benchmark**

Bench name	Exec. Time (cycles) MS	MT	CORR-GA
SCALEx5	27860	10295	24726
GEMVx5	22765	10630	21544
TRMVx5	11695	12175	11379
GERx5	23690	10285	41279
GEMMx5	1537930	670090	1356136
TRMMx5	748480	779200	711016

Data Type	Implementation	OP/Cycle/DSP
INT32	ResNet-18 ScaleHLS [1]	1.343
INT32	ResNet-18 TVM-VTA [2]	0.344
INT32	LA-GA GEMM	0.646
FP32	GEMM ScaleHLS	0.393
FP32	LA-GA GEMM	0.277

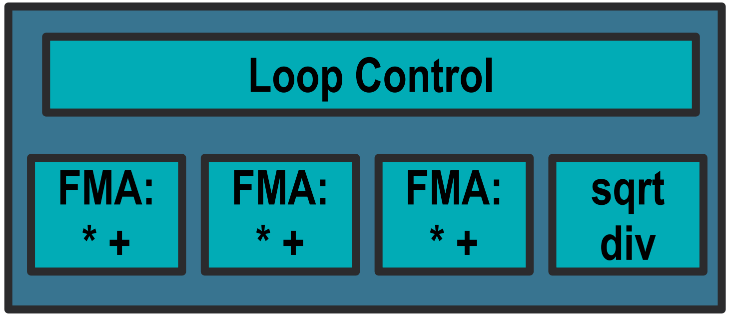
LA-GA is comparable with state-of-the art design in terms of performance-per-area (GEMM, FP32)

[1] Hanchen Ye, Cong Hao, Jianyi Cheng, Hyunmin Jeong, Jack Huang, Stephen Neuendorffer, and Deming Chen. ScaleHLS: A New Scalable High-Level Synthesis Framework on Multi-Level Intermediate Representation. In 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA).
[2] Thierry Moreau, Tianqi Chen, Luis Vega, Jared Roesch, Eddie Yan, Lianmin Zheng, Josh Fromm, Ziheng Jiang, Luis Ceze, Carlos Guestrin. A hardware–software blueprint for flexible deep learning specialization. IEEE Micro (2019)

Benchmark: Correlation-Generic Accelerator

Bench name	Arithmetic expression	Execution Time (cycles) MS	MT	CORR-GA	FLOP/C/DSP MS	MT	CORR-GA
CENTER	$X^C_{ij} = X_{ij} - (\sum_{i'} X_{i'j})/n$	8343	4166	12480	0.495	0.495	0.055
STDDEV	$\sigma^X_j = \sqrt{\sum_i (X^C_i)^2/n}$	16691	8370	29053	0.247	0.247	0.047
CENTER-REDUCE-DIV	$X^{CR}_{ij} = (X_{ij} - \sum_{i'} X_{i'j})/(\sigma^X_j \cdot \sqrt{n})$	20935	10486	33352	0.247	0.164	0.052
CORR	$(X^{CR})^t \cdot X^{CR}$	291221	144614	303763	0.468	0.314	0.150
CORRx3	$3\times$CORR	873663	433842	320603	0.468	0.314	0.425

- **Accelerator: CORR-GA**



- **Raw Performance lower than Max Sharing designs on CORR subexpressions**
  - Dedicated accelerators
  - Sub-optimality in the choice of kernels
- **Performance similar Max Sharing for CORR**
  - Dominated by the matrix multiplication
  - Better than MT on batched Correlation
- **Still semi-generic!**

Final Words

- Limitations
- Future Work
- Conclusion

Limitations

- **Interconnect size**
  - Op/LUT and Op/FF are **2 to 20x off** dedicated accelerators
  - Due to
    - **Organisation of the on-chip buffer**
    - **Scheduling on the FUs**
- **Fixed matrix size**
  - Implementation limitation: the **Iteration Vector Generator** and the **Loop Bound Generator** use a constant size
  - Matrix / vector size could be send as part of the accelerator configuration
- **Data reuse**
  - Possible optimisation: Loop-invariants values are currenty fetched from the accelerator buffer **each cycle**
- **Vectorisation of the FUs**
  - BRAM limitation: only 2 load/store operations per cycle are possible
- **DSP repartition**
  - FMA are implemented as a mul-add sequence: only matrix-matrix or matrix-vector max their usage
  - Work in progress!

Future Work

- **Support BLAS 1 primitives**
  - Support reductions in the inner-most loop
    - **Temporary buffer for loop-carried accumulation**
    - **Hardware-compliante schedule**
      - Stalls of the execution pipeline
- **Reduce DSP/operation**
  - Use of “floating point” Vivado primitives
    - **Heavier interfaces (AXIStream)**
    - **FP16 FMA using only one DSP**
    - **Basic hardware unit for any FU using add/sub/mul**
- **Formalisation of the kernel detection algorithm**
  - Transformation of the loop bodies to 3-address code
  - Loop fission
  - Legality of the operation
- **Caracterisation of the supported applications**
  - Any combination of the accelerated kernels can be accelerated

Conclusion

- **Generation of a programmable accelerator**
  - For a family of applications
  - Focused on **performance-per-area**
- **Relying on polyhedral kernel merging**
  - Iteration domain extension
  - Loop fusion
  - Custom Functional Units generation
- **Performance in par with dedicated designs**
  - Evaluation on Linear Algebra (BLAS2-3) and Correlation
  - Batching favors performance-per-area
  - **Op/Cycle/DSP** comparable to state-of-the-art designs
- **Improved version of the Generic Accelerator under submission**

Interested in an academic full-time research job?

Interested in an academic full-time research job?

→ Inria hires!

Interested in an academic full-time research job?

Inria hires!

- ~40 opened (junior) tenured positions each year

Interested in an academic full-time research job?

Inria hires!

- ~40 opened (junior) tenured positions each year
- Senior positions, hundreds of post-docs, ...