GPU Technology Conference (GTC'19)
March 18-21, 2019

Can FPGAs compete with GPUs?

John Romein, Bram Veenboer

ASTRON
Netherlands Institute for Radio Astronomy

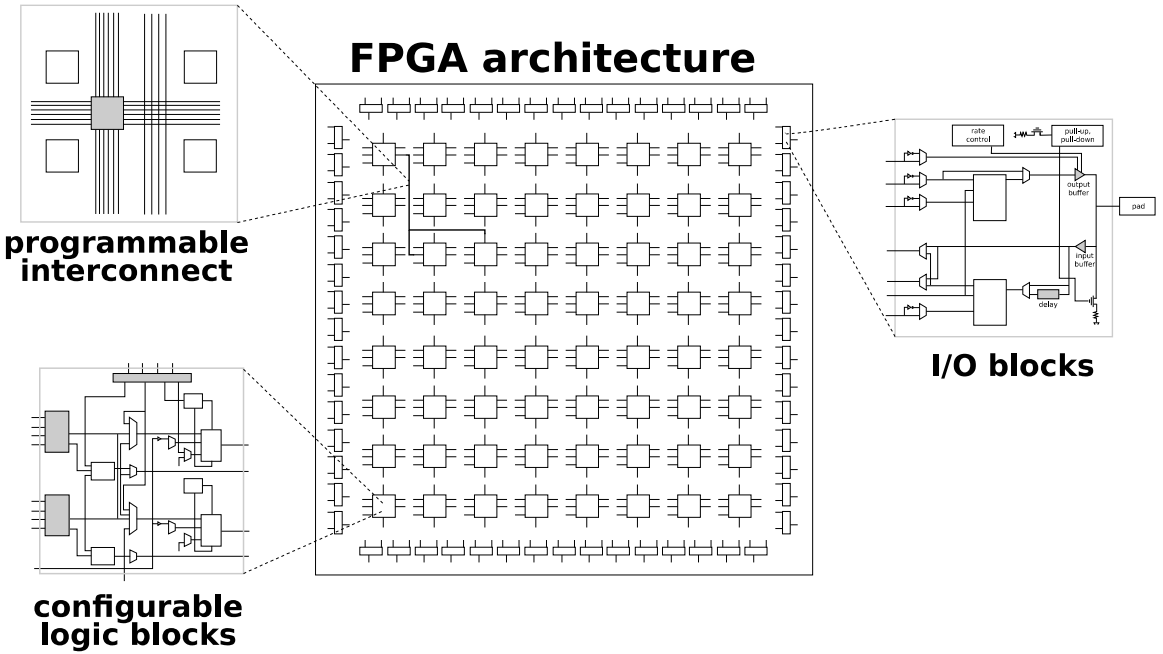
Outline

- explain FPGA
  - hardware
- FPGA vs. GPU
  - programming models (OpenCL)
  - case studies
    - matrix multiplication
    - radio-astronomical imaging

    } analyze performance & energy efficiency
  - lessons learned
- answer the question in the title

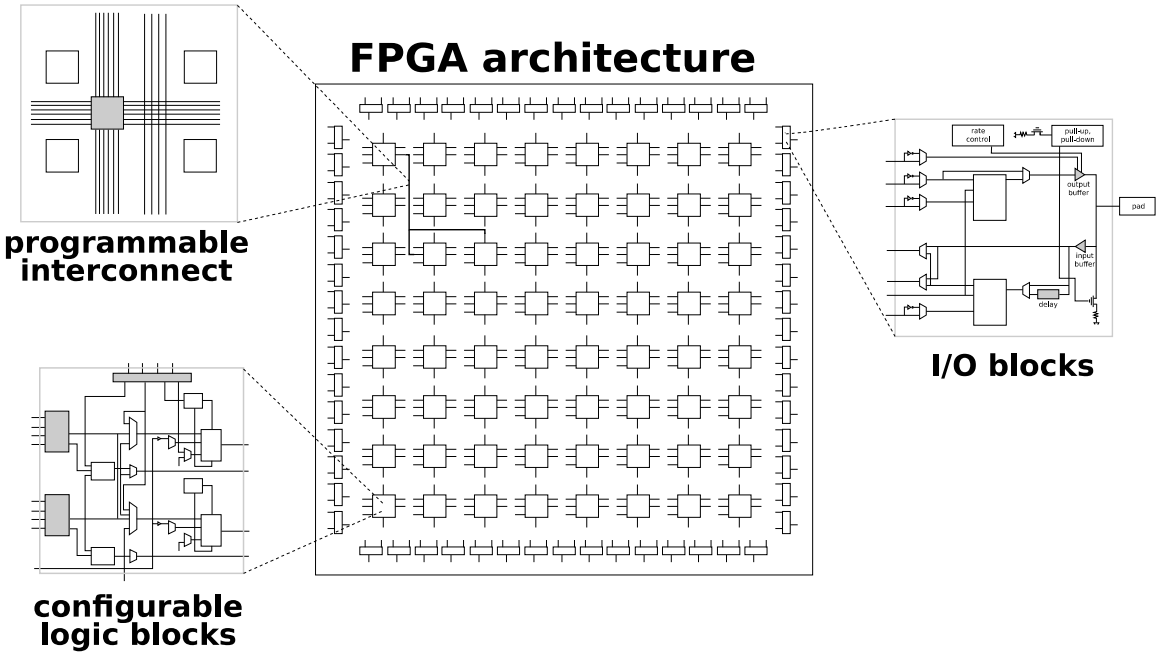
What is a Field-Programmable Gate Array (FPGA)?

- configurable processor
- collection of
  - multipliers / adders
  - registers
  - memories
  - logic
  - transceivers
  - clocks
  - interconnect



ASTRON
Netherlands Institute for Radio Astronomy

What is an FPGA program?



- connect elements
  - performs fixed function
- data-flow engine
- Hardware Description Language (HDL)
  - Verilog, VHDL
  - difficult

FPGA vs GPU

- **FPGA advantages**
  - high energy efficiency
    - no instruction decoding etc.
    - use/move as few bits as possible
  - configurable I/O
    - high bandwidth
    - e.g., many times 100 GbE

- **GPU advantages**
  - easier to program
  - more flexible
  - short compilation times

New Intel (Altera) FPGA technologies

1) high-level programming language (OpenCL)
2) hard Floating-Point Units
3) tight integration with CPU cores

➔ simple tasks: less programming effort than HDL

➔ allows complex HPC applications

FPGA ≠ GPU

- hardware
  - GPU: use hardware
  - FPGA: create hardware
- execution model
  - GPU: instructions
  - FPGA: data flow

Common language: OpenCL

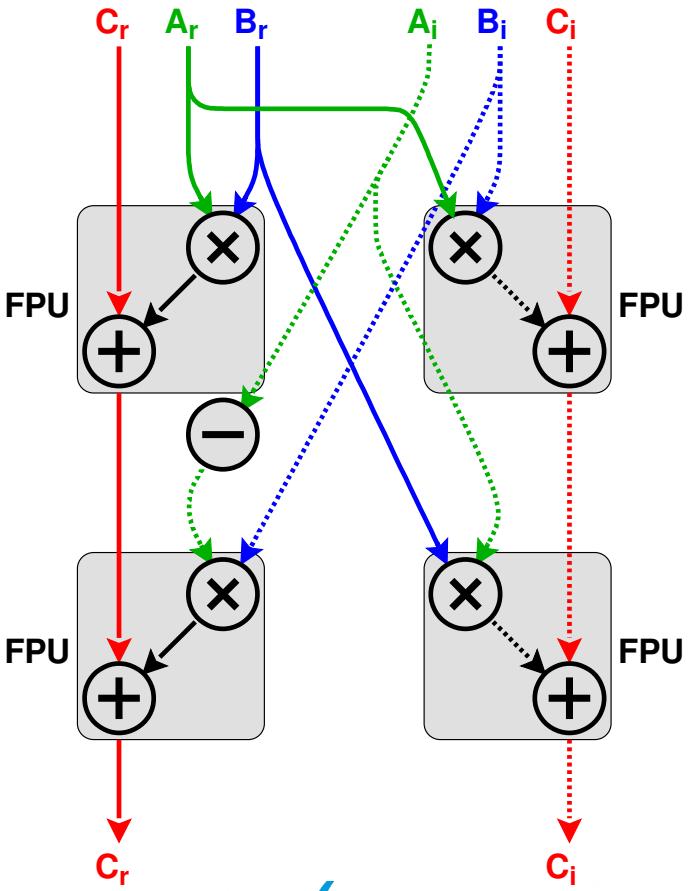
- OpenCL
  - similar to CUDA
  - C + explicit parallelism + synchronization + software-managed cache
  - offload to GPU/FPGA
- GPU programs not suitable for FPGAs
  - FPGA: data-flow engine

FPGA: data-flow pipeline example

- create hardware for complex multiply-add

```
C.real +=  A.real * B.real;
C.real += -A.imag * B.imag;
C.imag +=  A.real * B.imag;
C.imag +=  A.imag * B.real;
```

- needs four FPUs
- new input enters every cycle



ASTRON
Netherlands Institute for Radio Astronomy

Local memory

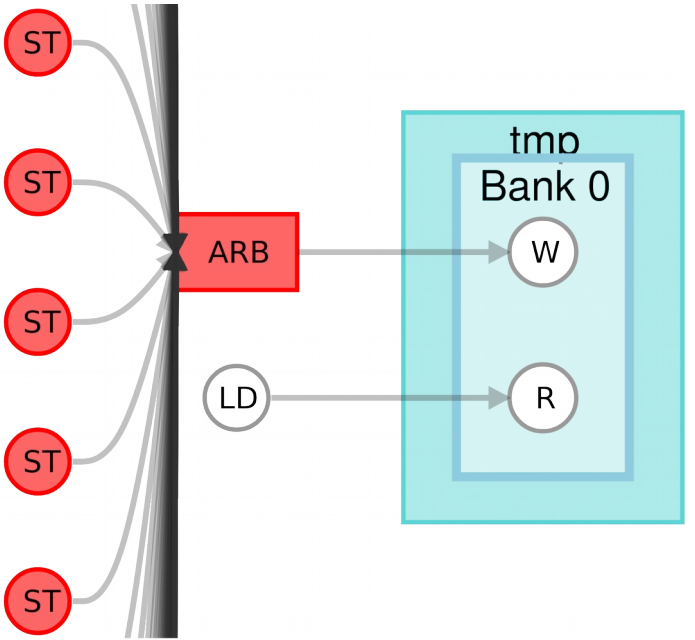
- GPU: use memory, FPGA: create memory

  float tmp[128] __attribute__((…));

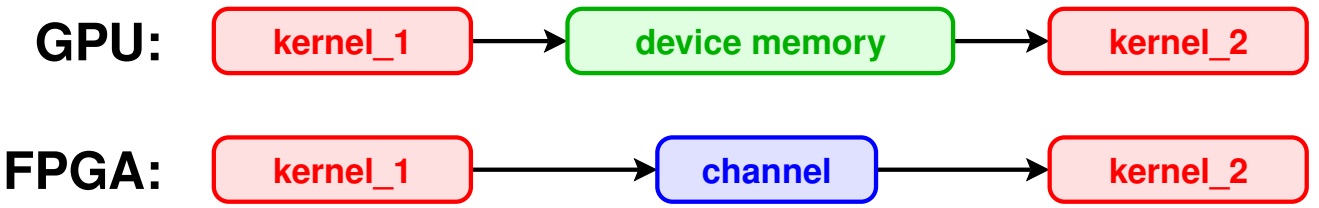
- properties:
  - registers or memory blocks
  - #banks
  - bank width
  - bank address selection bits
  - #read ports
  - #write ports
  - single/double pumped

  } attributes

  - with/without arbiter
  - replication factor
- well designed program: few resources, stall-free



Running multiple kernels

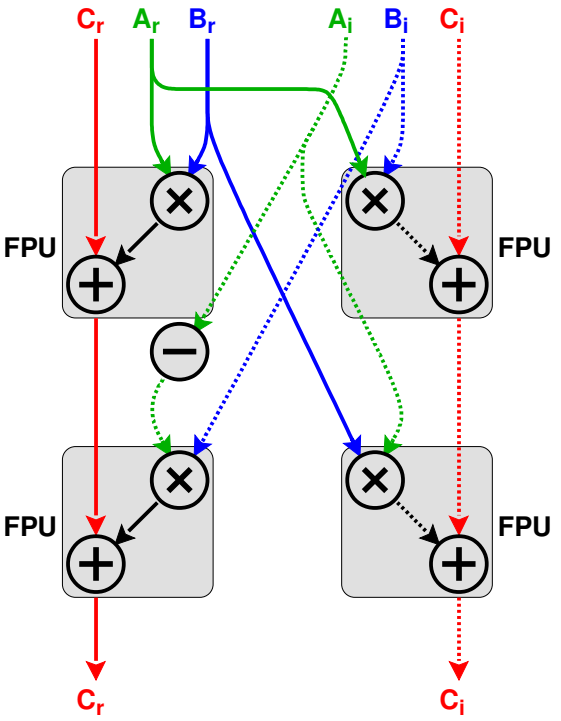


- GPU: consecutively
- FPGA: concurrently
  - channels (OpenCL extension)

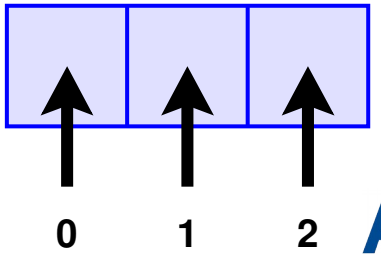
    channel float2 my_channel __attribute__((depth(256)));
  - requires less memory bandwidth
  - all kernels resident → must fit

Parallel constructs in OpenCL

- work groups, work items (CUDA: thread blocks, threads)
  - GPU: parallel
  - FPGA: default: one work item/cycle; not useful
- FPGA:
  - compiler auto-parallelizes whole kernel
  - #pragma unroll
    - create hardware for concurrent loop iterations
  - replicate kernels




```
#pragma unroll
  for (int i = 0; i < 3; i ++)
    a[i] = i;
```



ASTRON
Netherlands Institute for Radio Astronomy

More GPU/FPGA differences

- code outside critical loop:
  - GPU: not an issue
  - FPGA: can use many resources

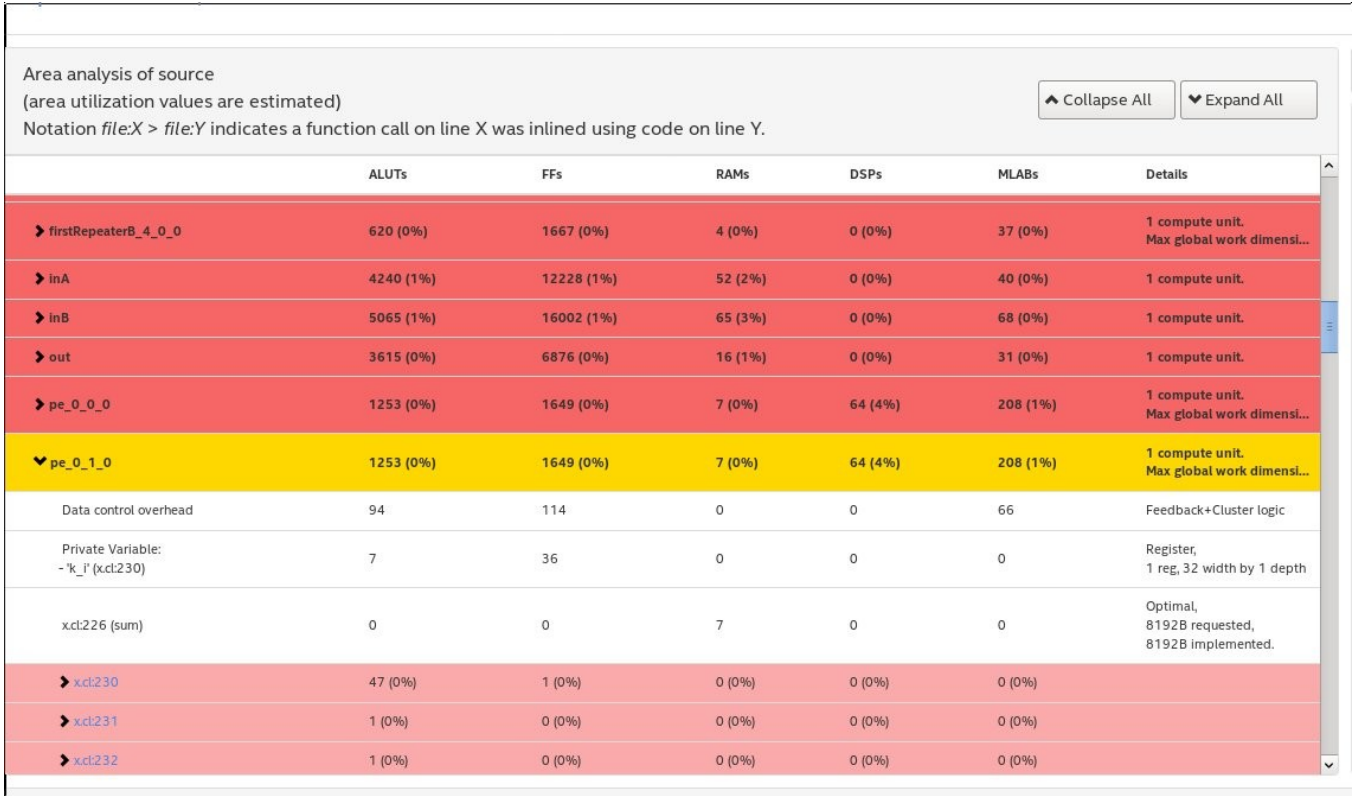
```
init_once();
for (int i = 0; i < 100000; i ++)
  do_many_times();
```

- Shift registers
  - FPGA: single cycle
  - GPU: expensive

```
for (int i = 256; i > 0; i --)
  a[i] = a[i – 1];
```

Resource optimizations

- compiler feedback
  - after minutes, not hours
  - HTML-based reports
- indispensible tool



Area analysis of source
(area utilization values are estimated)
Notation *file:X* > *file:Y* indicates a function call on line X was inlined using code on line Y.

Collapse All | Expand All

	ALUTs	FFs	RAMs	DSPs	MLABs	Details
firstRepeaterB_4_0_0	620 (0%)	1667 (0%)	4 (0%)	0 (0%)	37 (0%)	1 compute unit. Max global work dimensi...
inA	4240 (1%)	12228 (1%)	52 (2%)	0 (0%)	40 (0%)	1 compute unit.
inB	5065 (1%)	16002 (1%)	65 (3%)	0 (0%)	68 (0%)	1 compute unit.
out	3615 (0%)	6876 (0%)	16 (1%)	0 (0%)	31 (0%)	1 compute unit.
pe_0_0_0	1253 (0%)	1649 (0%)	7 (0%)	64 (4%)	208 (1%)	1 compute unit. Max global work dimensi...
pe_0_1_0	1253 (0%)	1649 (0%)	7 (0%)	64 (4%)	208 (1%)	1 compute unit. Max global work dimensi...
Data control overhead	94	114	0	0	66	Feedback+Cluster logic
Private Variable: - 'k_i' (x.cl:230)	7	36	0	0	0	Register, 1 reg, 32 width by 1 depth
x.cl:226 (sum)	0	0	7	0	0	Optimal, 8192B requested, 8192B implemented.
x.cl:230	47 (0%)	1 (0%)	0 (0%)	0 (0%)	0 (0%)	
x.cl:231	1 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	
x.cl:232	1 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	

Frequency optimization

- Compiler (place & route) determines F_{max}
  - Unlike GPUs
- Full FPGA: longest path limits F_{max}

- HDL: fine-grained control
- OpenCL: one clock for full design

- FPGA: 450 MHz; BSP: 400 Mhz; 1 DSP: 350 Mhz
- Little compiler feedback

- Recompile with random seeds
- Compile kernels in isolation to find frequency limiters
  - Tedious, but useful

Applications

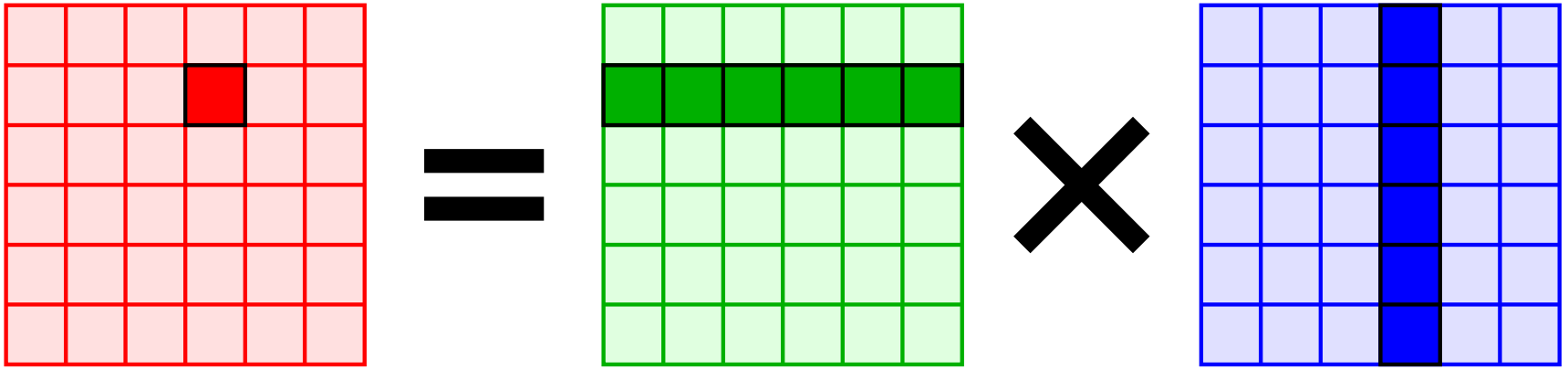
- dense matrix multiplication
- radio-astronomical imager

- Signal processing (FIR filter, FFT, correlations, beam forming)

ASTRON
Netherlands Institute for Radio Astronomy

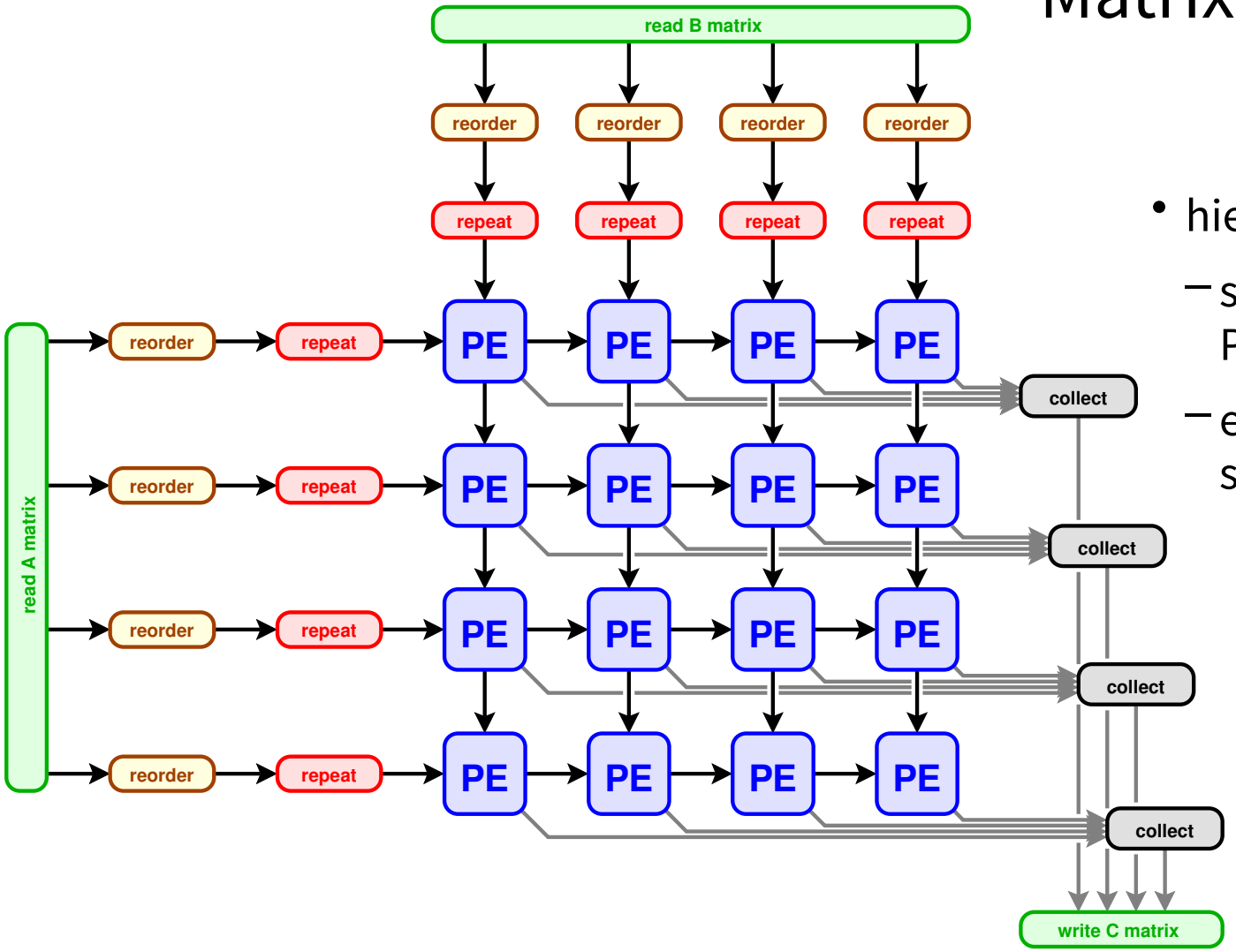
Dense matrix multiplication

- complex<float>
- horizontal & vertical memory accesses (→ coalesce)
- reuse of data (→ cache)



ASTRON
Netherlands Institute for Radio Astronomy

Matrix multiplication on FPGA



- hierarchical approach
  - systolic array of Processing Elements (PE)
  - each PE computes 32x32 submatrix

ASTRON
Netherlands Institute for Radio Astronomy

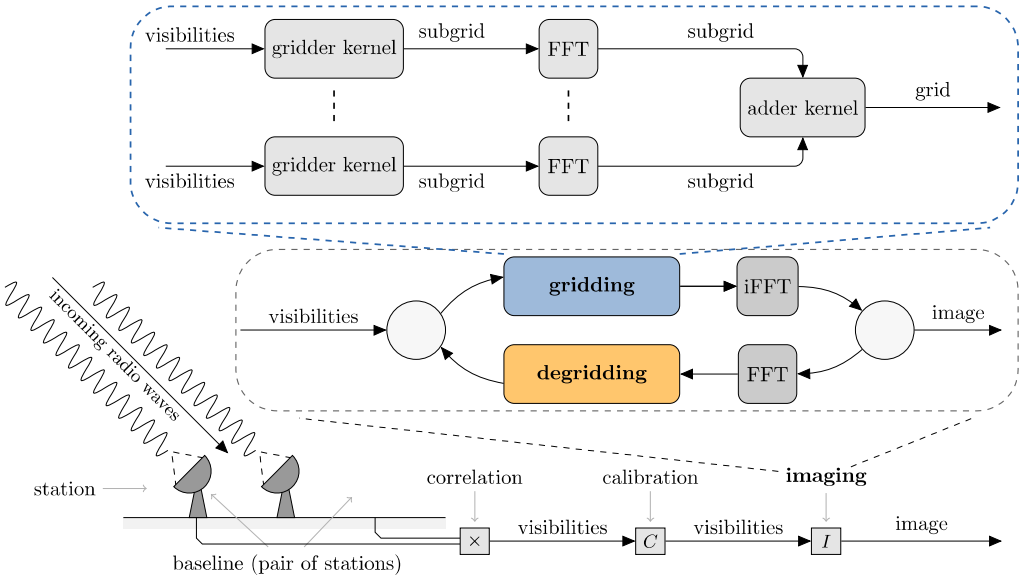
Matrix multiplication performance

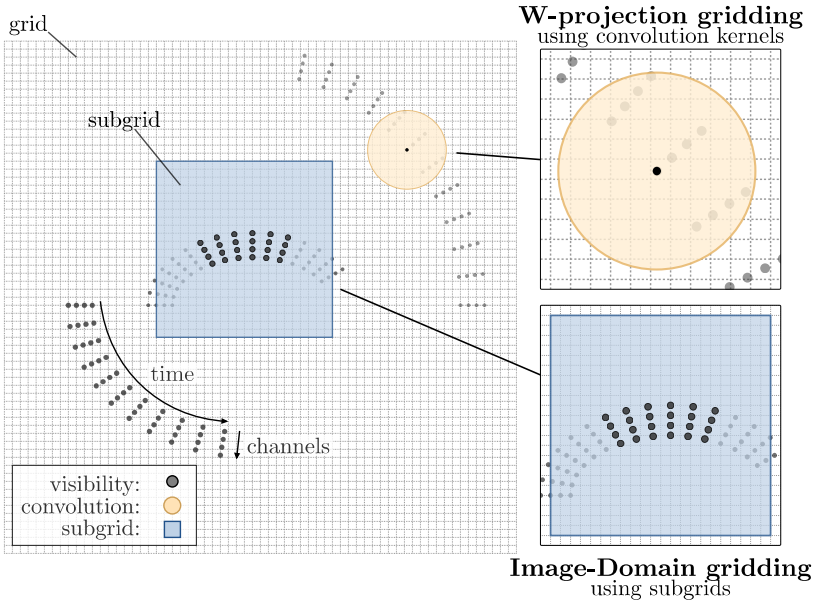
- Arria 10:
  - uses 89% of the DSPs, 40% on-chip memory
  - clock (288 MHz) at 64% of peak (450 Mhz)
  - nearly stall free

Type	Device	Performance (TFlop/s)	Power (W)	Efficiency (GFlop/W)
FPGA	Intel Arria 10	0.774	37	20.9
GPU	NVIDIA Titan X (Pascal)	10.1	263	38.4
GPU	AMD Vega FE	9.73	265	36.7

ASTRON
Netherlands Institute for Radio Astronomy

Image-Domain Gridding for Radio-Astronomical Imaging



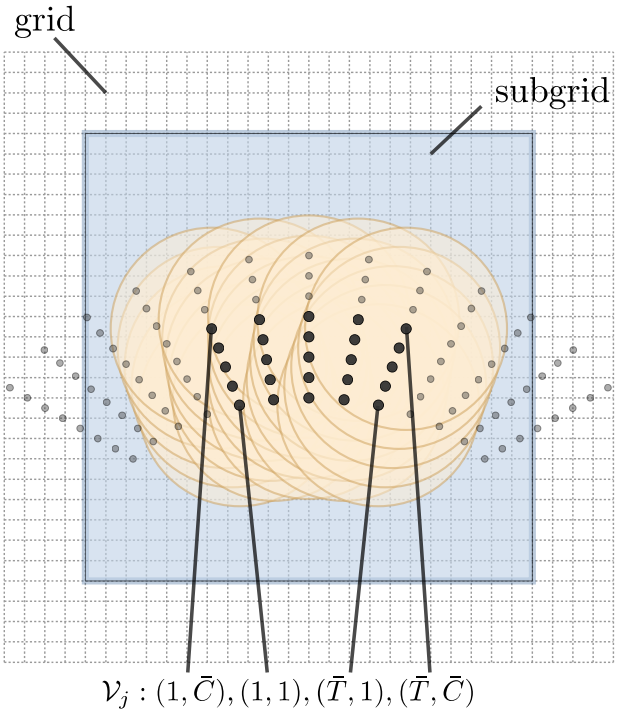


- see talk S9306 (Astronomical Imaging on GPUs)

ASTRON
Netherlands Institute for Radio Astronomy

Image-Domain Gridding algorithm

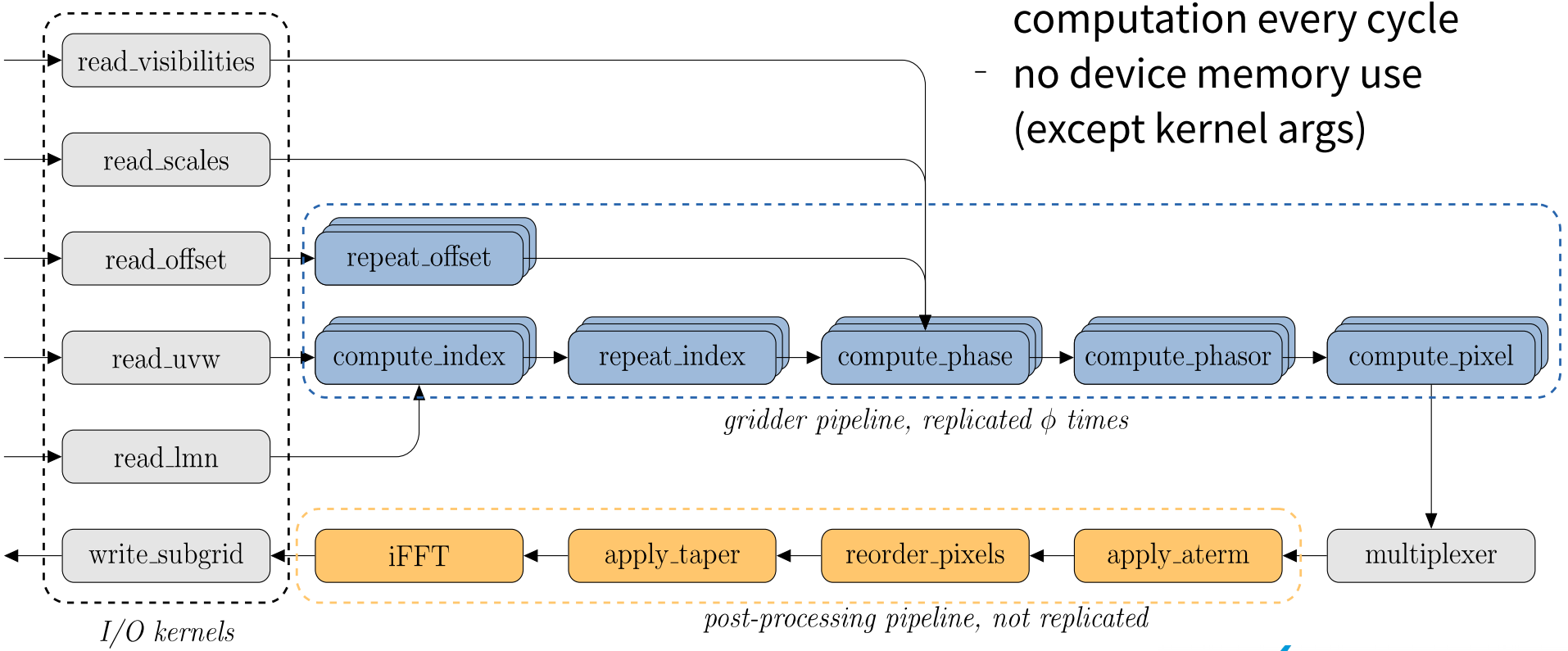
```
#pragma parallel
for s = 1...S :
   complex<float> subgrid[P ][N ×N ];
   for i = 1...N ×N :
      float offset = compute_offset(s, i);
      for t = 1...T :
         float index = compute_index(s, i, t);
         for c = 1...C :
            float scale = scales[c];
            float phase = offset - (index × scale);
            complex<float> phasor = {cos(phase), sin(phase)};
            #pragma unroll
            for p = 1...P : // 4 polarizations
               complex<float> visibility = visibilities[t][c][p];
               subgrid[p][i] += cmul(phasor, visibility);

   apply_aterm(subgrid);
   apply_taper(subgrid);
   apply_ifft(subgrid);
   store(subgrid);
```



ASTRON
Netherlands Institute for Radio Astronomy

FPGA gridding design

- Create a dataflow network:
  - Use all available DSPs
  - Every DSP performs a useful computation every cycle
  - no device memory use (except kernel args)



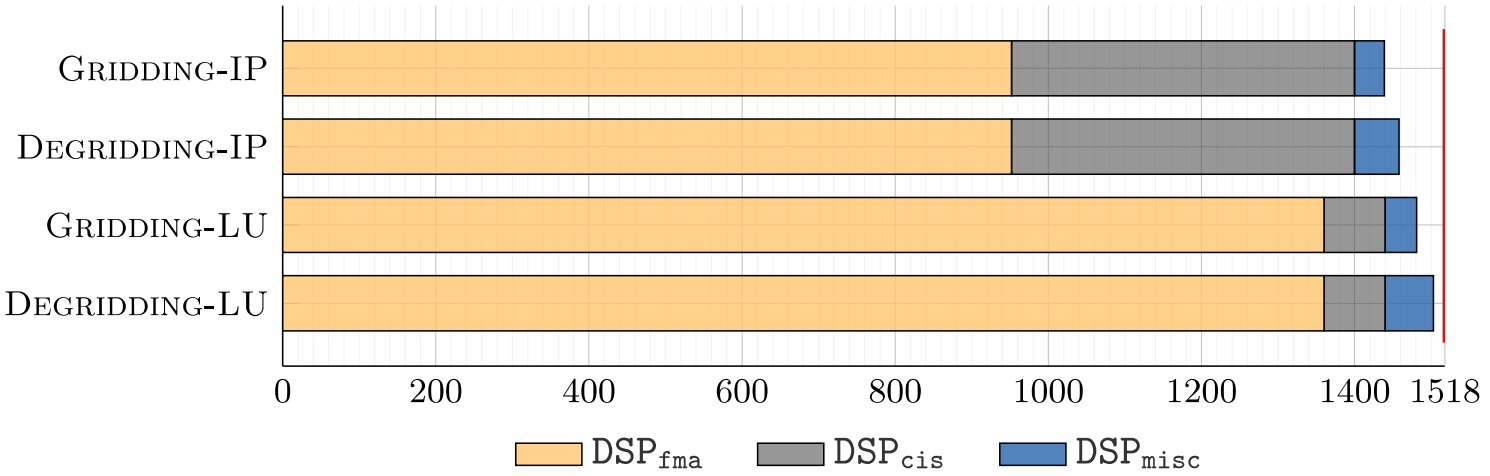
ASTRON
Netherlands Institute for Radio Astronomy

FPGA OpenCL gridder kernel

```
__attribute__((max_global_work_dim(0)))
__attribute__((autorun))
__attribute__((num_compute_units(NR_GRIDDERS)))
__kernel void gridder()
{
  int    gridder = get_compute_id(0);
  float8 subgrid[NR_PIXELS];

  for (unsigned short pixel = 0; pixel < NR_PIXELS; pixel ++) {
  | subgrid[pixel] = 0;
  }

  #pragma ivdep
  for (unsigned short vis_major = 0; vis_major < NR_VISIBILITIES; vis_major += UNROLL_FACTOR) {
  | float8 visibilities[UNROLL_FACTOR] __attribute__((register));
  |
  | for (unsigned short vis_minor = 0; vis_minor < UNROLL_FACTOR; vis_minor++) {
  | | visibilities[vis_minor] = read_channel_intel(visibilities_channel[gridder]);
  | }
  |
  | for (unsigned short pixel = 0; pixel < NR_PIXELS; pixel++) {
  | | float8 pixel_value = subgrid[pixel];
  | | float8 phasors     = read_channel_intel(phasors_channel[gridder]);   // { cos(phase), sin(phase) }
  | |
  | | #pragma unroll
  | | for (unsigned short vis_minor = 0; vis_minor < UNROLL_FACTOR; vis_minor++) {
  | | | pixel_value.even += phasors[vis_minor] * visibilities[vis_minor].even + -phasors[vis_minor] * visibilities[vis_minor].odd;
  | | | pixel_value.odd  += phasors[vis_minor] * visibilities[vis_minor].odd  +  phasors[vis_minor] * visibilities[vis_minor].even;
  | | }
  | |
  | | subgrid[pixel] = pixel_value;
  | }
  }

  for (unsigned short pixel = 0; pixel < NR_PIXELS; pixel ++) {
  | write_channel_intel(pixel_channel[gridder], subgrid[pixel]);
  }
}
```

Sine/cosine optimization

- compiler-generated: 8 DSPs
- limited-precision lookup-table: 1 DSP
  - more DSPs available → replicate Φ ~~14~~ 20x



ASTRON
Netherlands Institute for Radio Astronomy

FPGA resource usage + F_{max}

	ALUTs	FFs	RAMs	DSPs	MLABs	ϕ	F_{max}
gridding-ip	43%	31%	64%	1439 **(95%)**	71%	14	**258**
degridding-ip	47%	35%	72%	1441 **(95%)**	78%	14	**254**
gridding-lu	27%	32%	61%	1498 **(99%)**	57%	20	**256**
degridding-lu	33%	38%	73%	1503 **(99%)**	69%	20	**253**

- almost all of 1518 DSPs available used
- F_{max} < 350 Mhz

ASTRON
Netherlands Institute for Radio Astronomy

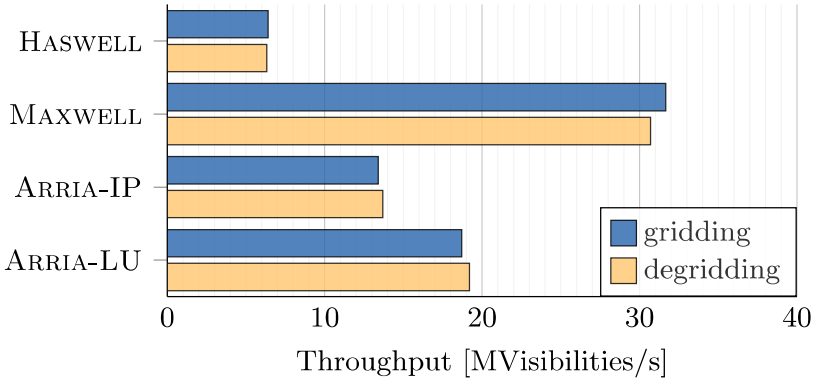
Experimental setup

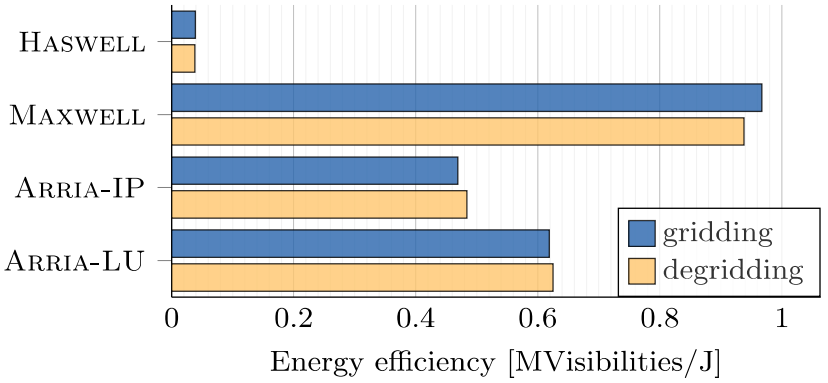
- compare Intel Arria 10 FPGA to comparable CPU and GPU
- CPU and GPU implementations are both optimized

Type	Device	#FPUs	Peak	Bandwidth	TDP	Process
CPU	Intel Xeon E5-2697v3	224	1.39 TFlop/s	68 GB/s	145W	28nm (TSMC)
FPGA	Nallatech 385A	1518	1.37 TFlop/s	34 GB/s	75W	20nm (TSMC)
GPU	NVIDIA GTX 750 Ti	640	1.39 TFlop/s	88 GB/s	60W	28nm (TSMC)

Throughput and energy-efficiency comparison

- **throughput**: number of visibilities processed per second (Mvisibilities/s)
- **energy-efficiency**: number of visibilities processed per Joule (MVisibilities/J)

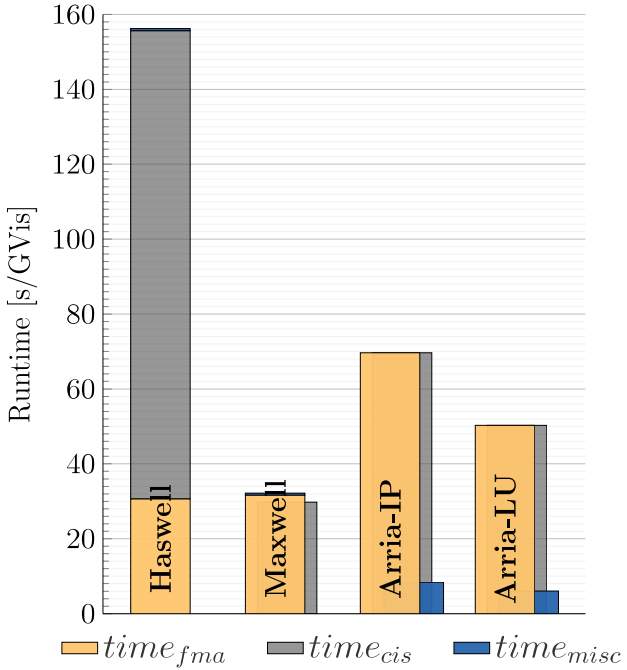


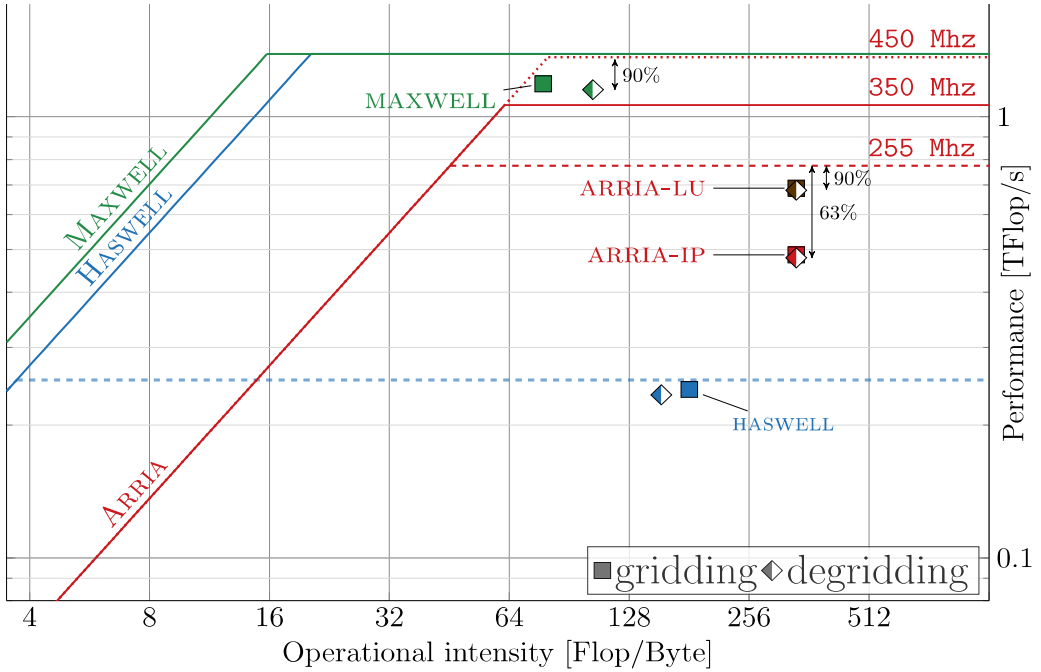


- *similar peak performance, what causes the performance differences?*

ASTRON
Netherlands Institute for Radio Astronomy

Performance analysis





- **CPU**: sin/cos in software takes 80% of total runtime
- **GPU**: sin/cos overlapped with other computations
- **FPGA:** sin/cos overlapped, but cannot use all DSPs for FMAs

ASTRON
Netherlands Institute for Radio Astronomy

FPGAs vs GPUs: lessons learned (1/3)

- dataflow vs. imperative
  - rethink your algorithm
  - different program code
- on FPGAs:
  - frequent use of OpenCL extensions
  - think about resource usage, occupancy, timing
  - less use of off-chip memory
- parallelism:
  - both: kernel replication and vectorization
  - FPGA: pipelining and loop unrolling

FPGAs vs GPUs: lessons learned (2/3)

- FPGA ≠ GPU, yet: same optimizations …
  - exploit parallelism
  - maximize FPU utilization
    - hide latency
  - optimize memory performance
    - hide latency → prefetch
    - maximize bandwidth → avoid bank conflicts, unit-stride access (coalescing)
    - reuse data (caching)
- … but achieved in very different ways!
- + architecture-specific optimizations

FPGAs vs GPUs: lessons learned (3/3)

- much simpler than HDL
  - FPGAs accessible to wider audience
    - long learning curve
  - small performance penalty
- yet not as easy as GPUs
  - distribute resources in complex dataflow
  - optimize for high clock
- tools still maturing

Current/future work

- Stratix 10 versus Volta/Turing
  - Stratix 10 imager port not trivial ← different optimizations for new routing architecture
- 100 GbE FPGA support
  - send/receive UDP packets in OpenCL
  - BSP firmware development
- streaming data applications
  - signal processing (filtering, correlations, beam forming, ...)
- OpenCL FFT library

Conclusions

- complex applications on FPGAs now possible
  - high-level language
  - hard FPUs
  - tight integration with CPUs
- GPUs vs FPGAs
  - 1 language; no (performance) portability
  - very different architectures
  - yet many optimizations similar

So can FPGAs compete with GPUs?

- depends on application
  - compute → GPUs
  - energy efficiency → GPUs

  } FPGA not far behind; CPU way behind
  - I/O → FPGA
  - flexibility → CPU/GPU
  - programming ease → CPU, then GPU, then FPGA

Acknowledgements

- This work was funded by
  - Netherlands eScience Center (Triple-A 2)
  - EU H2020 FETHPC (DEEP-EST, Grant Agreement nr. 754304)
  - NWO Netherlands Foundation for Scientific Research (DAS-5)
- Other people
  - Suleyman Demirsoy (Intel), Johan Hiddink (NLeSC), Atze v.d. Ploeg (NLeSC), Daniël v.d. Schuur (ASTRON), Merijn Verstraaten (NLeSC), Ben v. Werkhoven (NLeSC)