FPGA AND CEREBRAS TESTBEDS

Maurice Jamieson (EPCC)

m.jamieson@epcc.ed.ac.uk

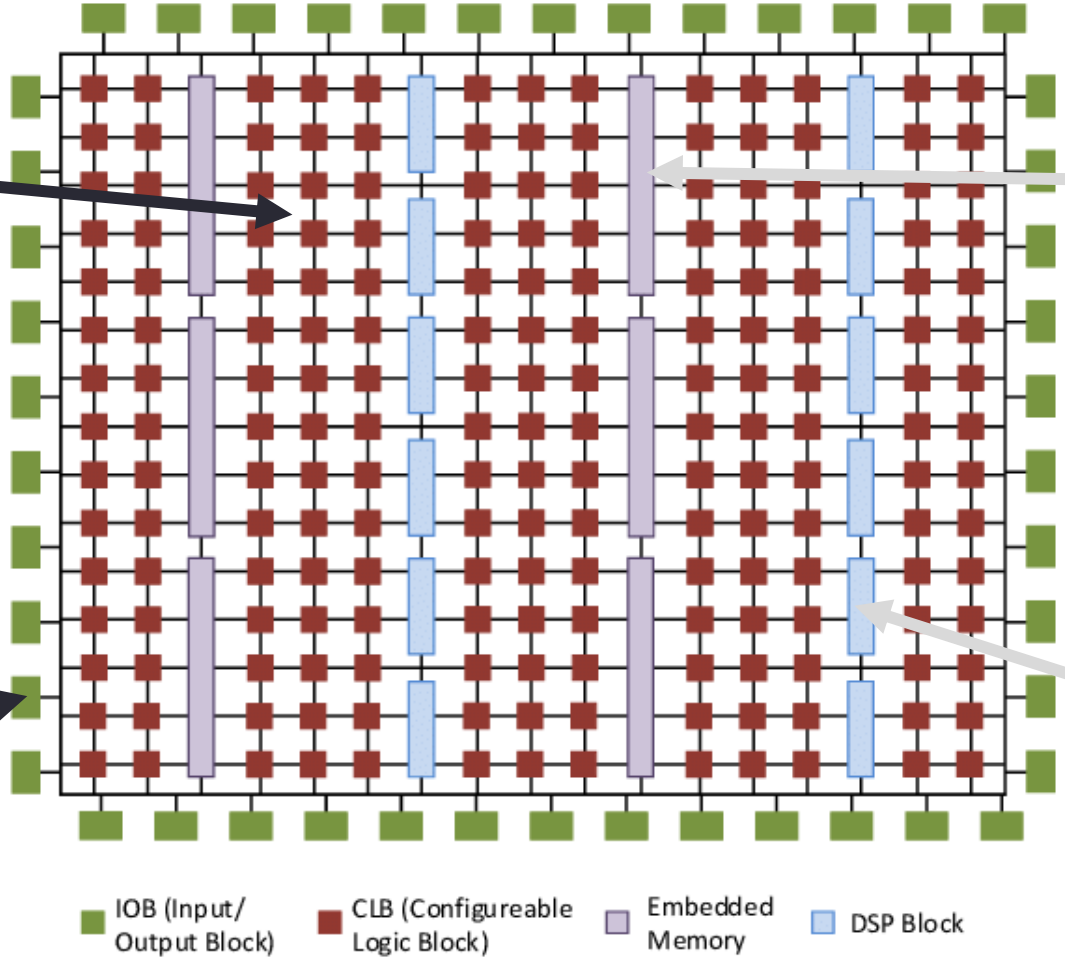
epcc

THE UNIVERSITY OF EDINBURGH

What is an FPGA?

Configurable Logic Block (CLB) contain look up tables which are configured with the application logic. These are sitting within a sea of configurable interconnect

Lots of I/O connections to the outside world, such as PCIe, HBM2, DDR, QSFP28 networking



Very fast on-chip memory, known as Block RAM (BRAM) and approx. 40 TB/s, similar to L1 cache and accessible in approx. 1 cycle

ASIC style components to perform arithmetic, used as the building blocks for floating point arithmetic as this saves a large amount of configurable logic

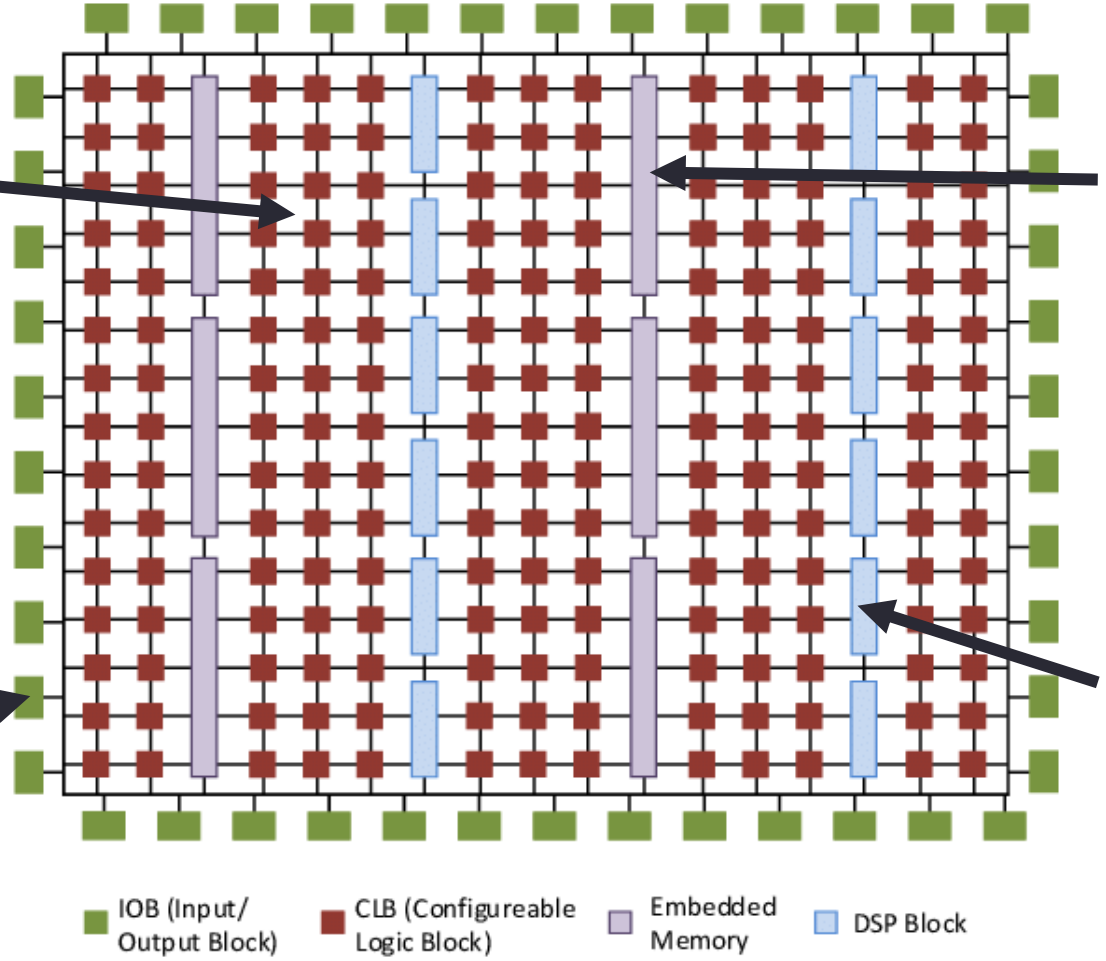
epcc

THE UNIVERSITY OF EDINBURGH

What is an FPGA?

Configurable Logic Block (CLB) contain look up tables which are configured with the application logic. These are sitting within a sea of configurable interconnect

Lots of I/O connections to the outside world, such as PCIe, HBM2, DDR, QSFP28 networking



Very fast on-chip memory, known as Block RAM (BRAM) and approx. 40 TB/s, similar to L1 cache and accessible in approx. 1 cycle

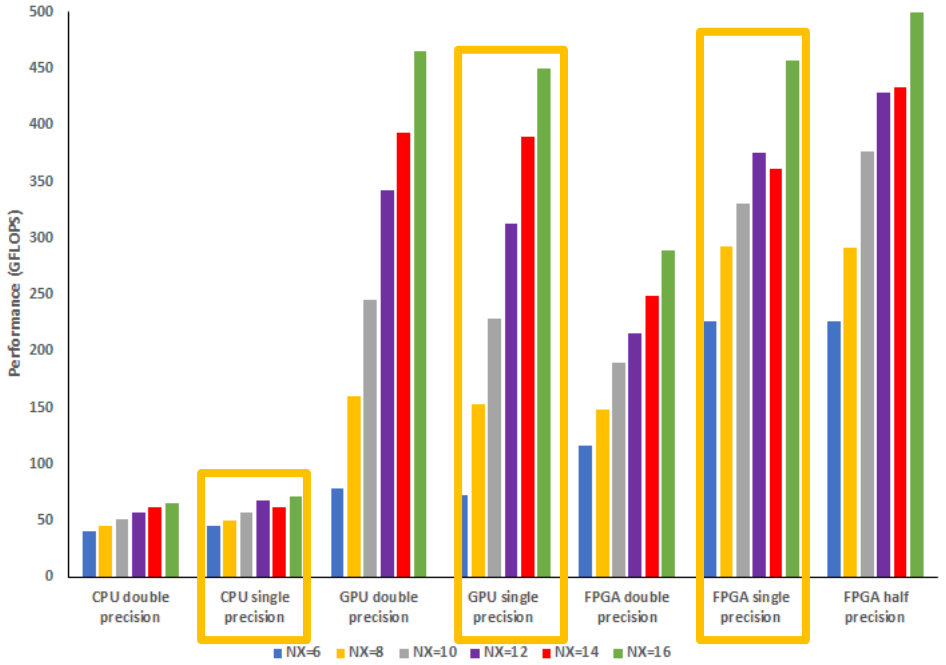
ASIC style components to perform arithmetic, used as the building blocks for floating point arithmetic as this saves a large amount of configurable logic

epcc

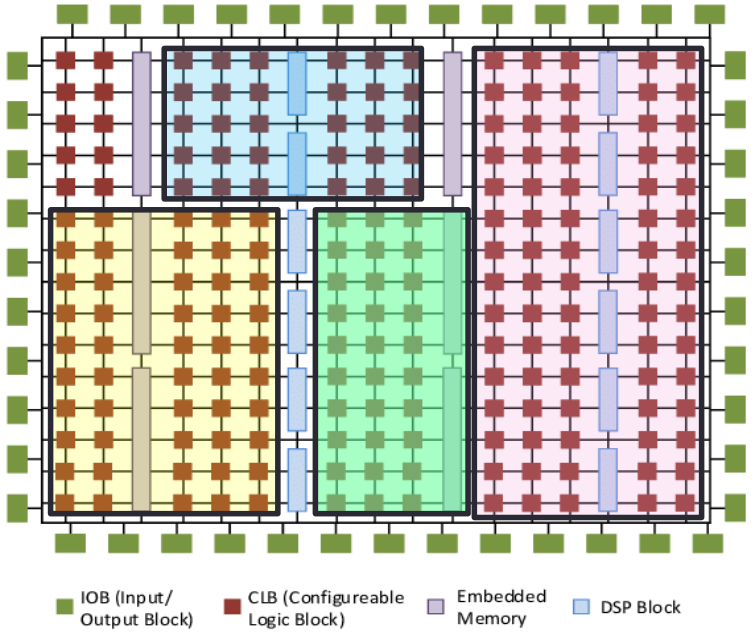
THE UNIVERSITY OF EDINBURGH

Why for HPC?

- For FPGAs, if your workload is compute-bound then a GPU is probably a better option!



Nekbone performance. A Xilinx Alveo U280, CPU is 24-core Xeon Platinum Cascade Lake, GPU is NVIDIA V100



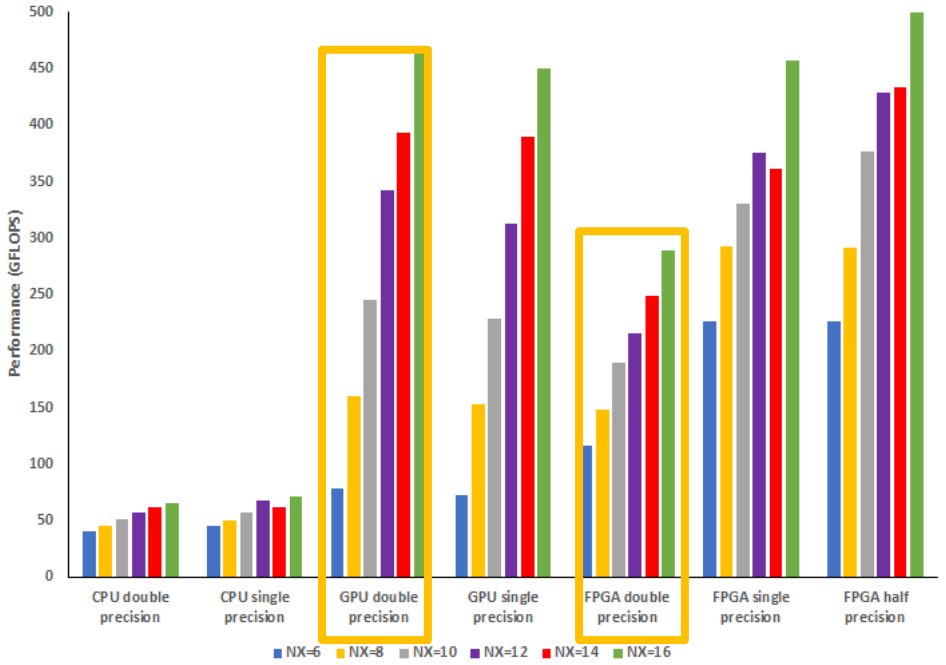
- However, if your code is memory bound or bound by other core issues, this is when you could get benefits
  - Tailoring how you use that fast L1-style BRAM memory
  - Exploiting high-bandwidth off-chip connections and HBM2

epcc

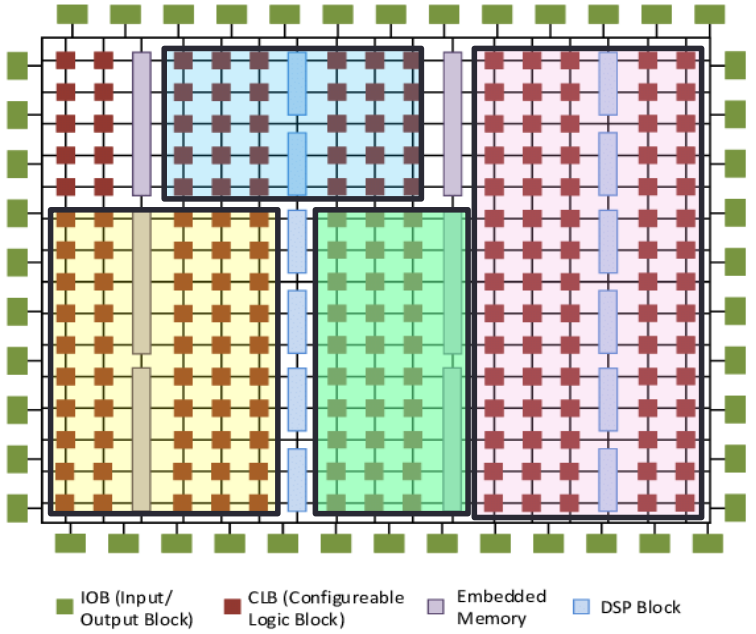
THE UNIVERSITY OF EDINBURGH

Why for HPC?

- For FPGAs, if your workload is compute-bound then a GPU is probably a better option!



Nekbone performance. A Xilinx Alveo U280, CPU is 24-core Xeon Platinum Cascade Lake, GPU is NVIDIA V100



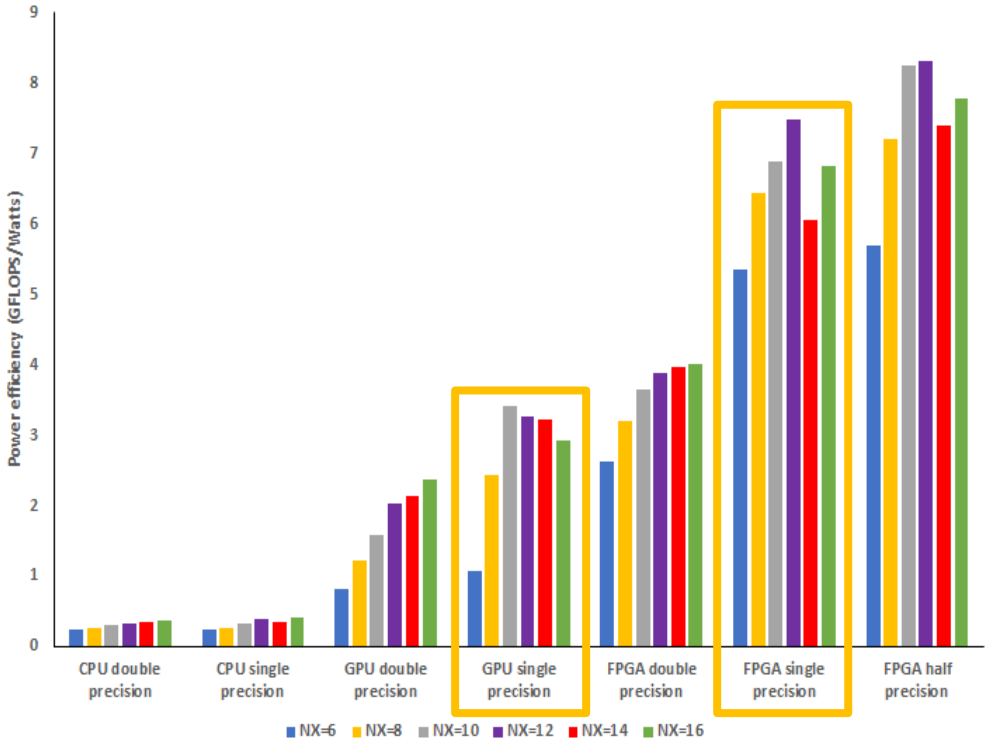
- However, if your code is memory bound or bound by other core issues, this is when you could get benefits
  - Tailoring how you use that fast L1-style BRAM memory
  - Exploiting high-bandwidth off-chip connections and HBM2

epcc

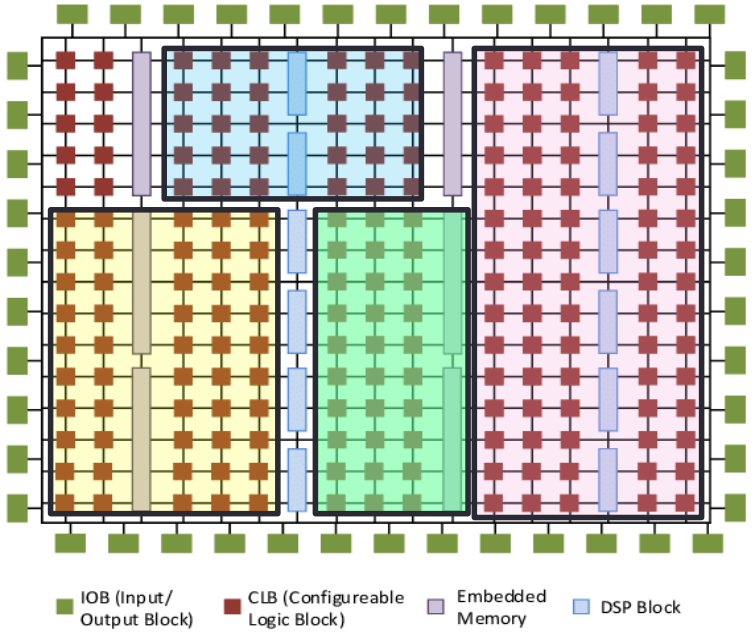
THE UNIVERSITY OF EDINBURGH

Why for HPC?

- For FPGAs, if your workload is compute-bound then a GPU is probably a better option!



Nekbone power efficiency. A Xilinx Alveo U280, CPU is 24-core Xeon Platinum Cascade Lake, GPU is NVIDIA V100



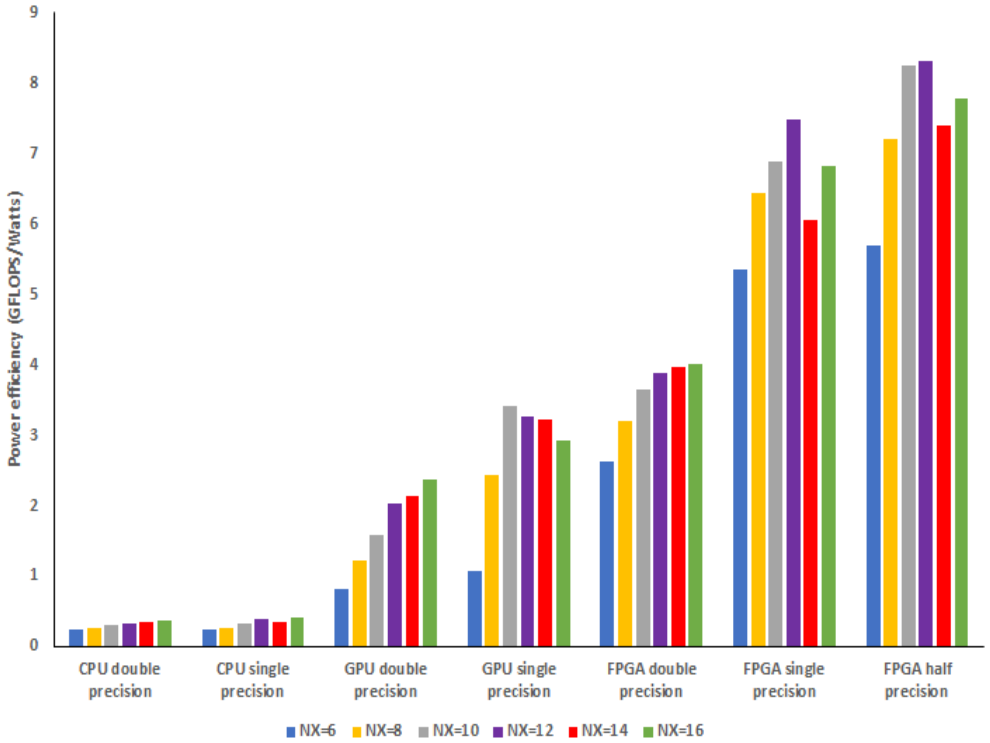
- However, if your code is memory bound or bound by other core issues, this is when you could get benefits
  - Tailoring how you use that fast L1-style BRAM memory
  - Exploiting high-bandwidth off-chip connections and HBM2

epcc

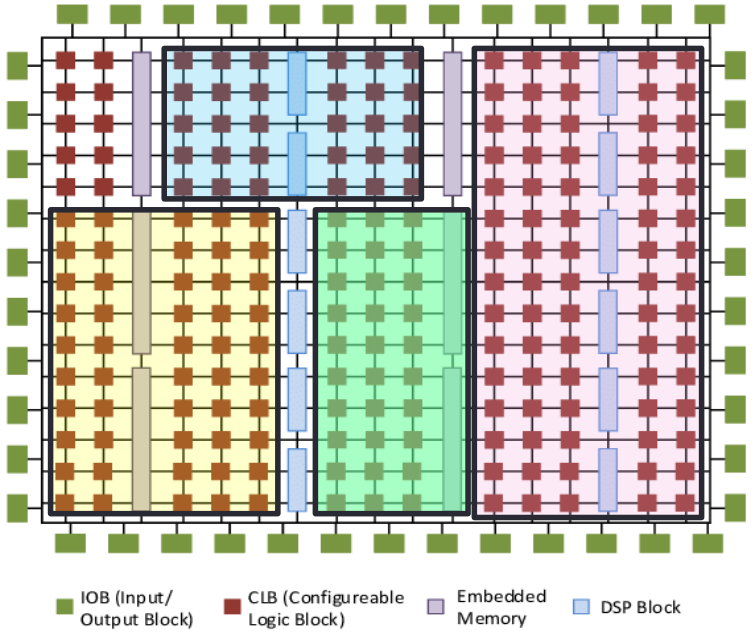
THE UNIVERSITY OF EDINBURGH

Why for HPC?

- **For FPGAs, if your workload is compute-bound then a GPU is probably a better option!**



Nekbone power efficiency. A Xilinx Alveo U280, CPU is 24-core Xeon Platinum Cascade Lake, GPU is NVIDIA V100



- **However, if your code is memory bound or bound by other core issues, this is when you could get benefits**
  - Tailoring how you use that fast L1-style BRAM memory
  - Exploiting high-bandwidth off-chip connections and HBM2

epcc

THE UNIVERSITY OF EDINBURGH

FPGA testbed

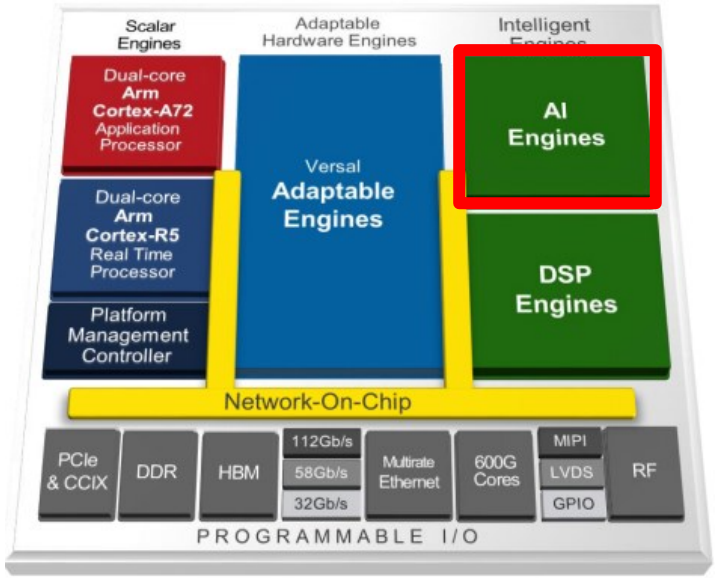
- Intended to provide a range of FPGAs across vendors
  - Make these available as a service to users, along with node for building and general development
- Considerable number of users, mainly in the UK but some from USA too

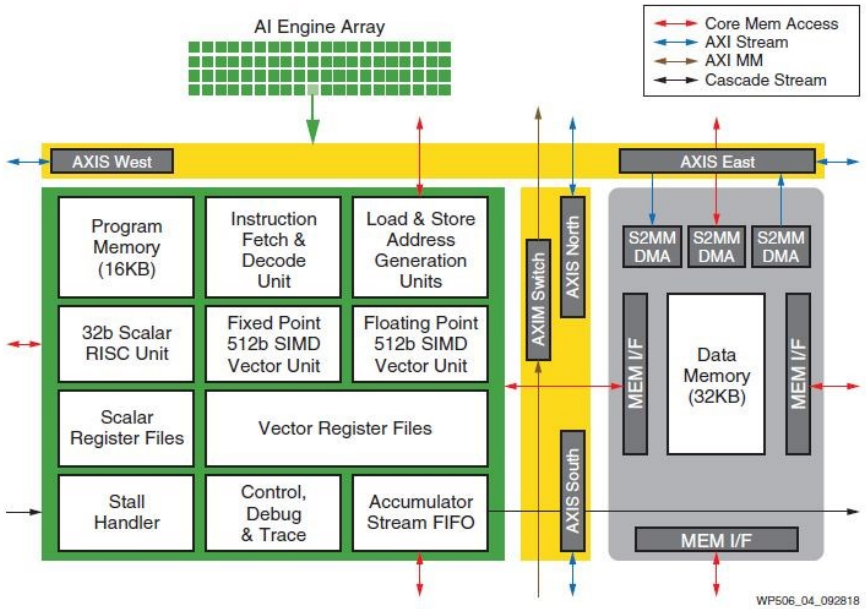
https://fpga.epcc.ed.ac.uk

epcc | THE UNIVERSITY OF EDINBURGH

AMD AI engines

- Began as compute accelerators in AMD's Versal architecture
  - In 2022/2023 undertook the first study of using AMD's AI engines for scientific computing workloads by exploring Met Office advection kernel on them



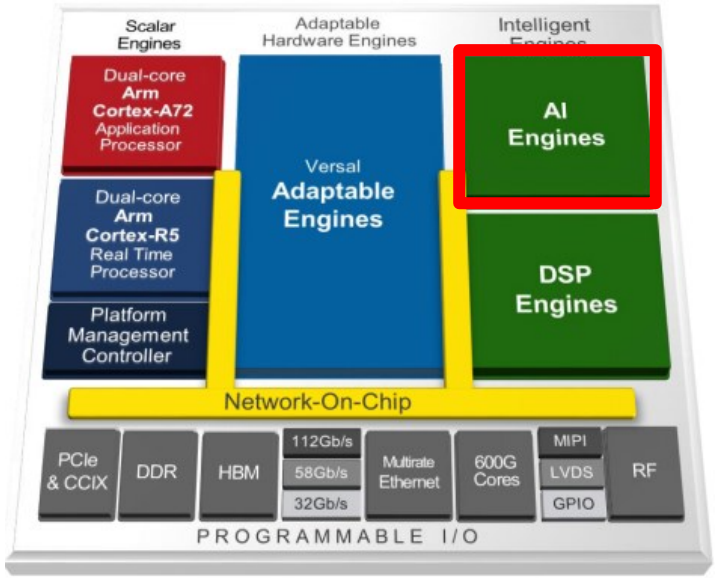


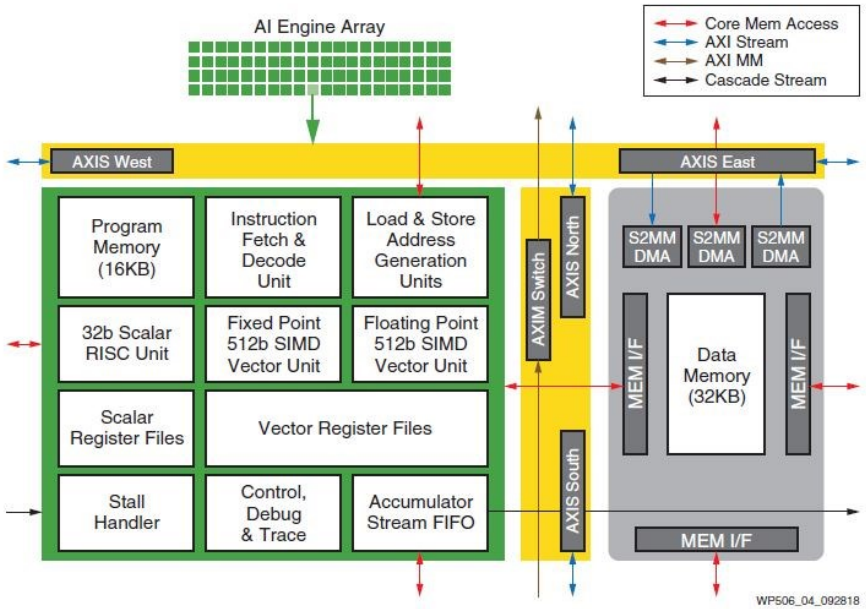
- Recently have been integrated by AMD into the Ryzen AI CPUs in their Neural Processing Unit (NPU)
  - These mainline CPUs contain an array of twenty AI engines which can be used for accelerating workloads running on the CPU
  - Not yet in server grade AMD CPUs, but might in the future

epcc | THE UNIVERSITY OF EDINBURGH

AMD AI engines

- Began as compute accelerators in AMD's Versal architecture
  - In 2022/2023 undertook the first study of using AMD's AI engines for scientific computing workloads by exploring Met Office advection kernel on them

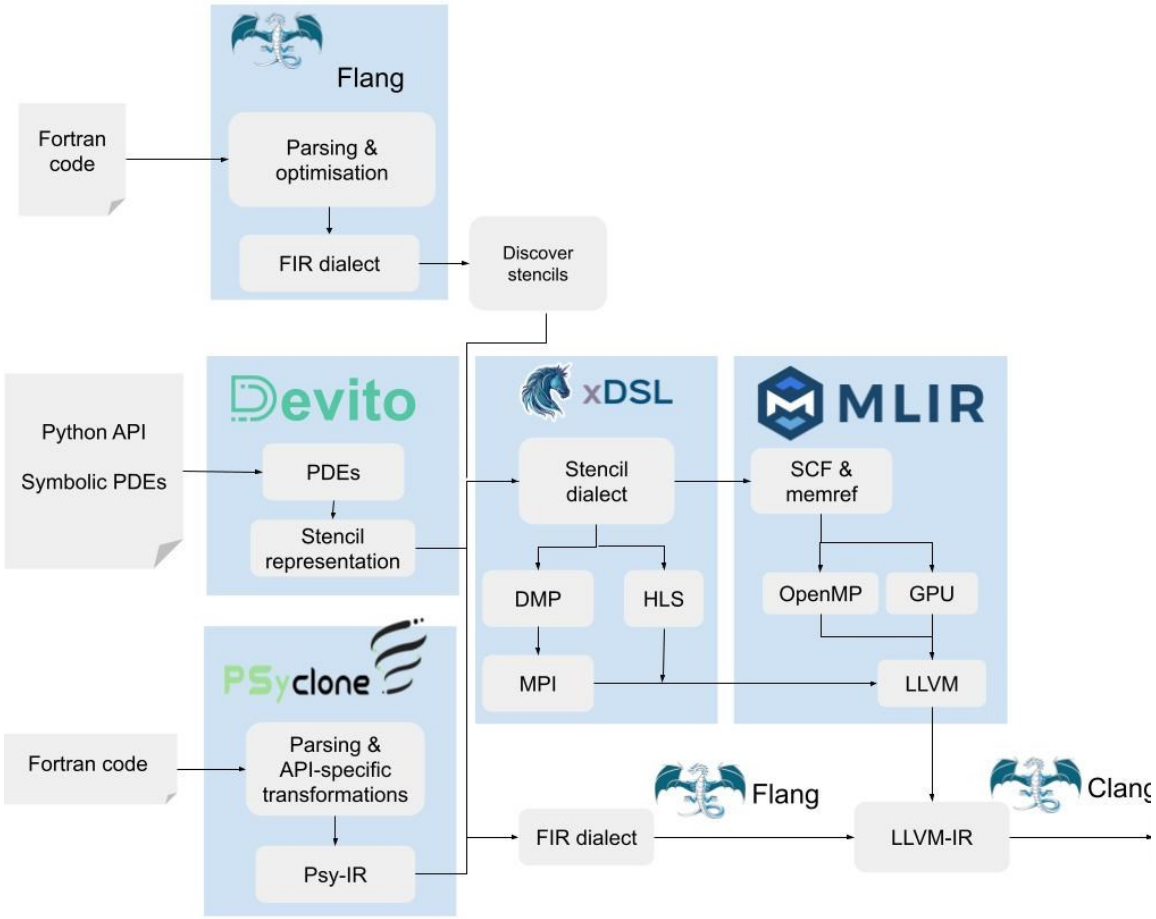




- Recently have been integrated by AMD into the Ryzen AI CPUs in their Neural Processing Unit (NPU)
  - These mainline CPUs contain an array of twenty AI engines which can be used for accelerating workloads running on the CPU
  - Not yet in server grade AMD CPUs, but might in the future

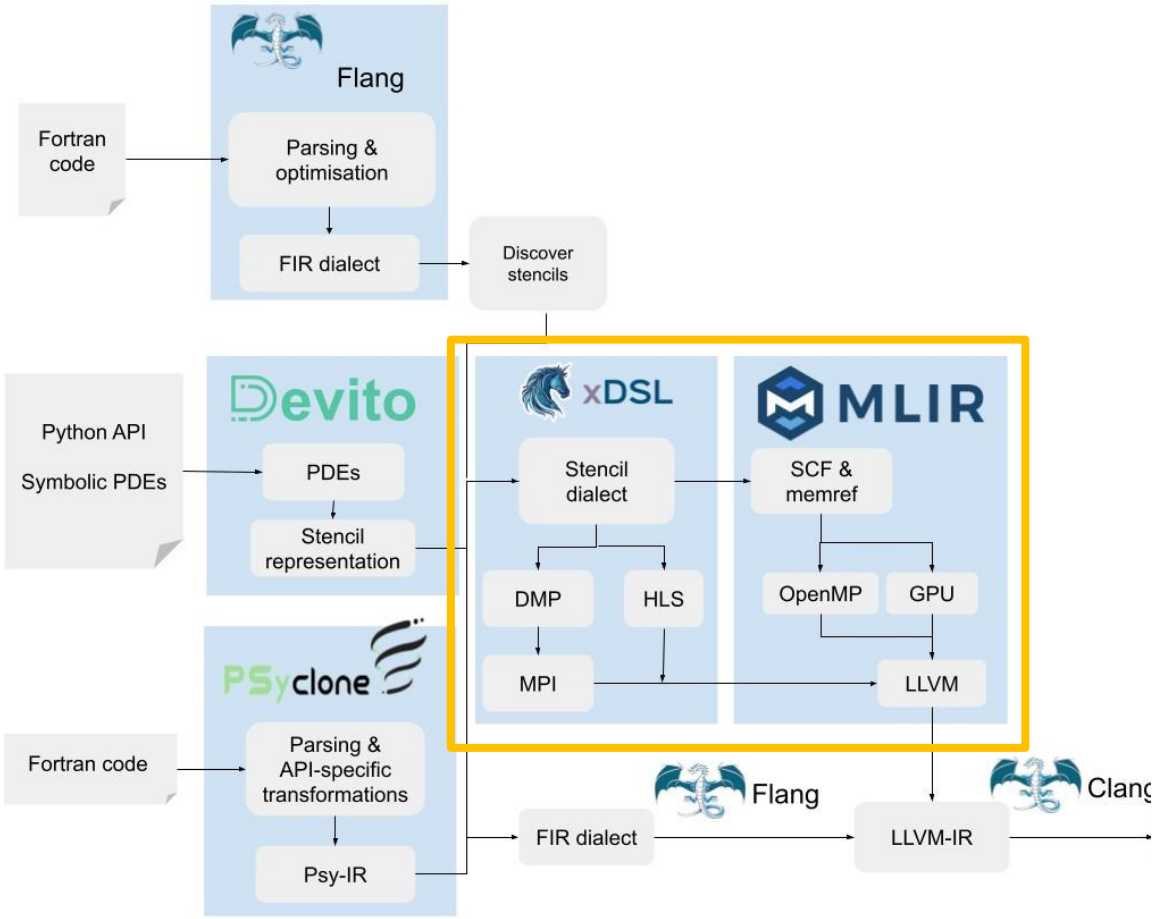
Programmability: The elephant in the room

- Programming FPGAs is hard, this requires algorithmic changes and deep understanding of the **underlying dataflow architecture** by the programmer
  - It is not realistic that scientific developers will have this expertise, or the time to port their codes to the architecture.
- Worked with the xDSL ExCALIBUR project to address this via MLIR
  - They provide the compiler expertise
  - We provide the hardware/architecture knowledge
- Extract stencils from codes and use this to drive automatic transformation/optimisation for FPGAs



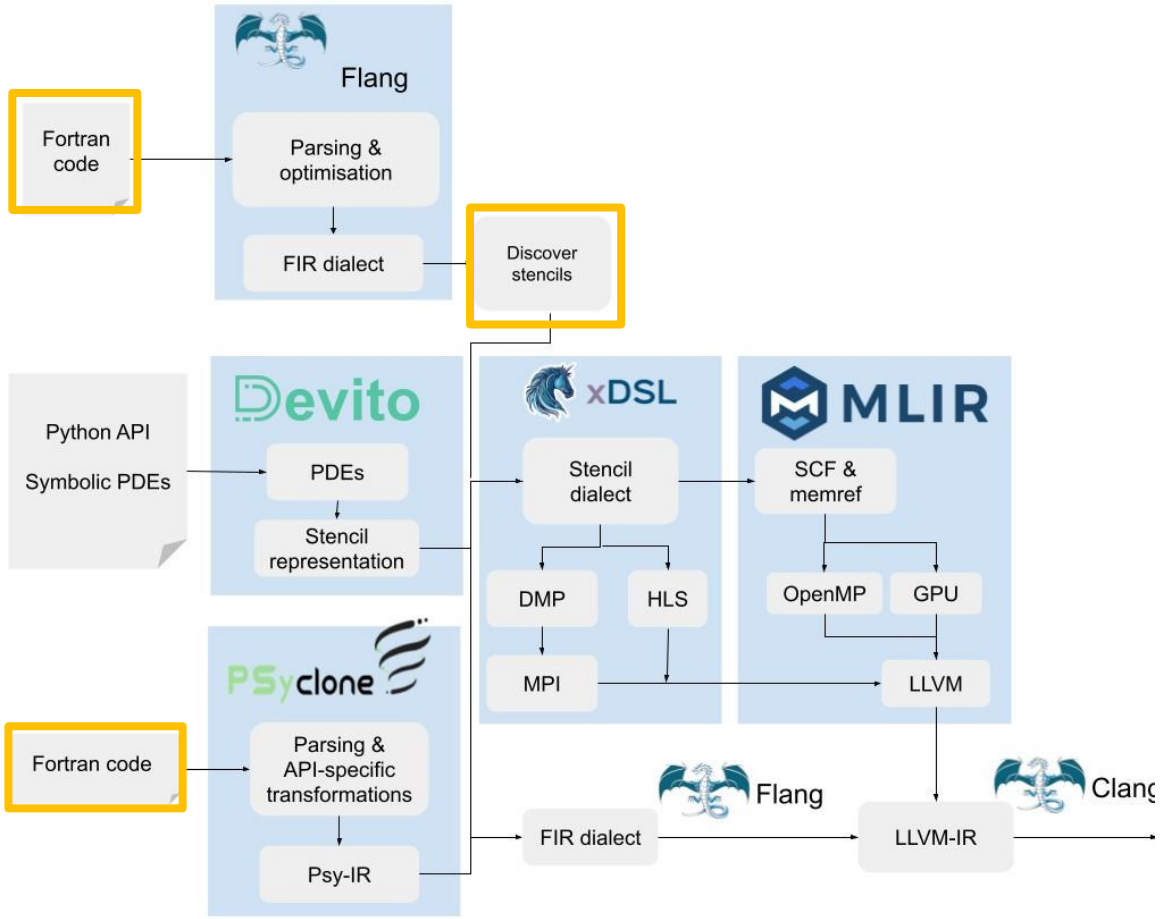
Programmability: The elephant in the room

- Programming FPGAs is hard, this requires algorithmic changes and deep understanding of the underlying dataflow architecture by the programmer
  - It is not realistic that scientific developers will have this expertise, or the time to port their codes to the architecture.
- **Worked with the xDSL ExCALIBUR project to address this via MLIR**
  - They provide the compiler expertise
  - We provide the hardware/architecture knowledge
- Extract stencils from codes and use this to drive automatic transformation/optimisation for FPGAs



Programmability: The elephant in the room

- Programming FPGAs is hard, this requires algorithmic changes and deep understanding of the underlying dataflow architecture by the programmer
  - It is not realistic that scientific developers will have this expertise, or the time to port their codes to the architecture.
- Worked with the xDSL ExCALIBUR project to address this via MLIR
  - They provide the compiler expertise
  - We provide the hardware/architecture knowledge
- **Extract stencils from codes and use this to drive automatic transformation/optimisation for FPGAs**

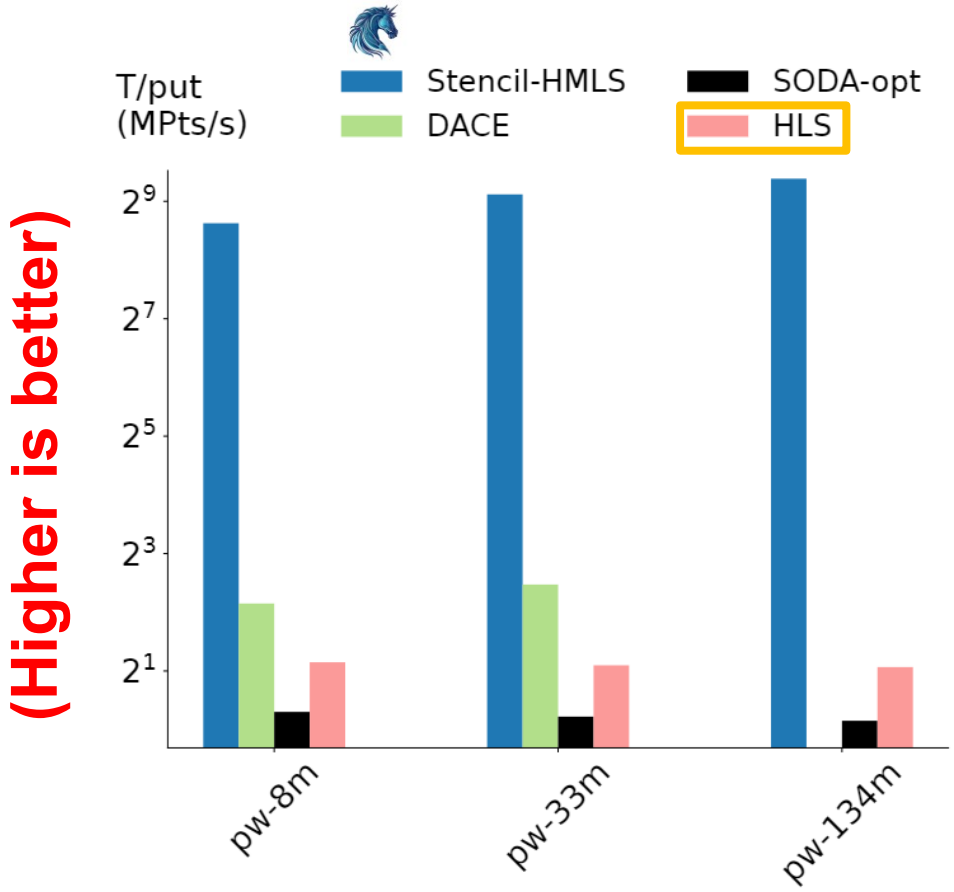


epcc

THE UNIVERSITY OF EDINBURGH

Programmability: The elephant in the room

- Piacsek and Williams (PW) advection kernels running on FPGA
  - Stencil-HMLS: Stencil High-Multi Level Synthesis
  - SODA: Stencil with Optimized Dataflow Architecture
  - DaCe: Data-Centric Parallel Programming
  - HLS: High-Level Synthesis

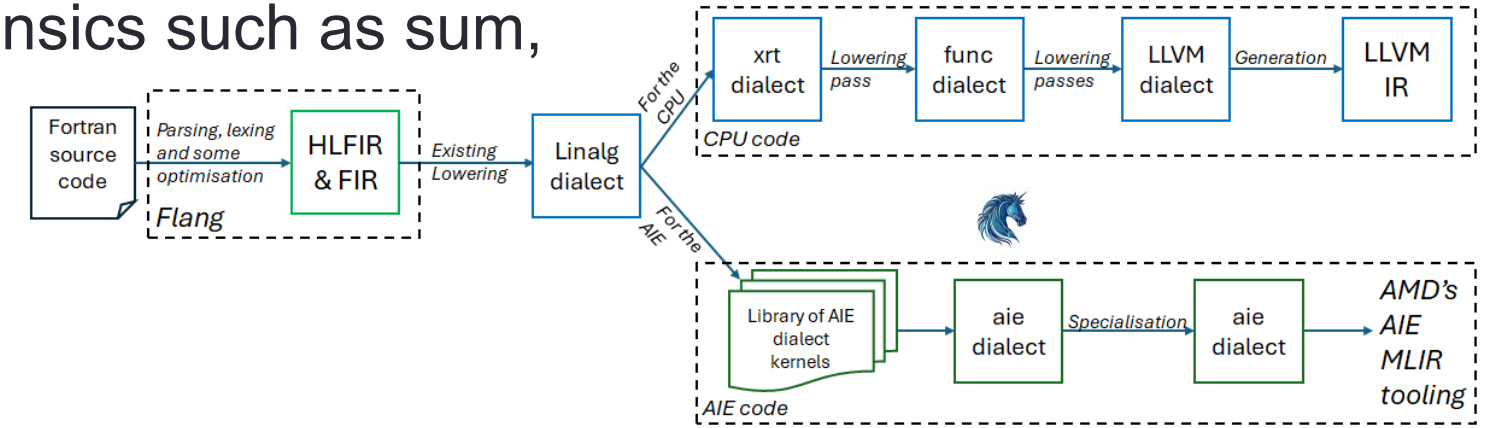


More details at https://arxiv.org/pdf/2310.01914

epcc | THE UNIVERSITY OF EDINBURGH

A similar approach for AI Engines

- We used a similar approach to seamlessly accelerate codes on the AI Engines too
  - In this instance Fortran intrinsics such as sum, maxval, matmul



Data type	CPU runtime (us)	NPU first runtime (us)	NPU subsequent runtime (us)
int16	5473	2572	1353
int32	14032	2635*	1503*
bfloat16	815194	2626	1357
float32	17566	3901*	1471*

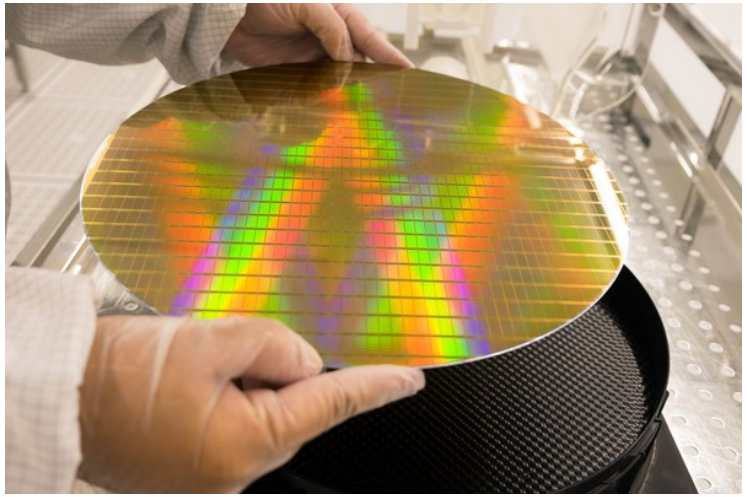
Performance of matmul intrinsic of 256x256x512 elements

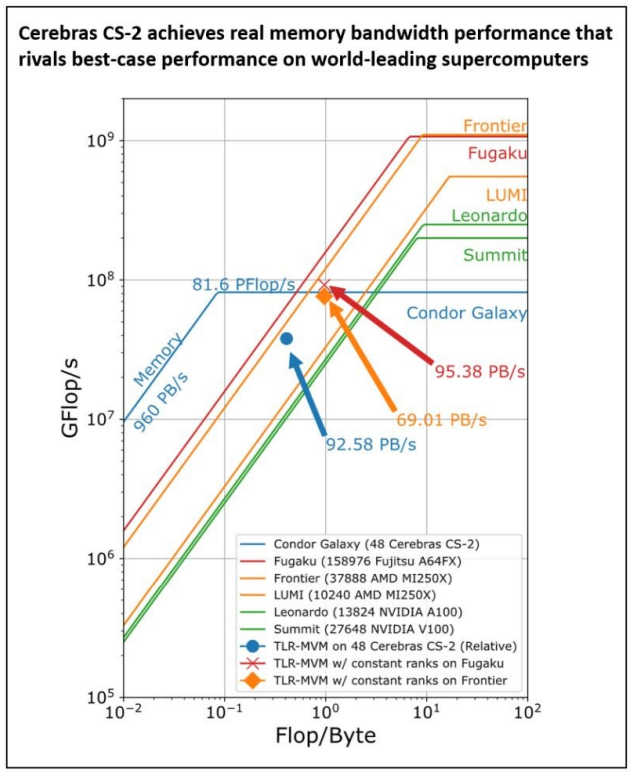
- The same idea, to lower the barrier to entry in accelerating codes on these architectures by leveraging the compiler

epcc | THE UNIVERSITY OF EDINBURGH

Cerebras CS-2

- A single wafer, Wafer Scale Engine (WSE)
  - CS-2 has around 850,000 individual cores, along with 40GB of SRAM (distributed amongst the cores), and memory BW of 20 PB/s



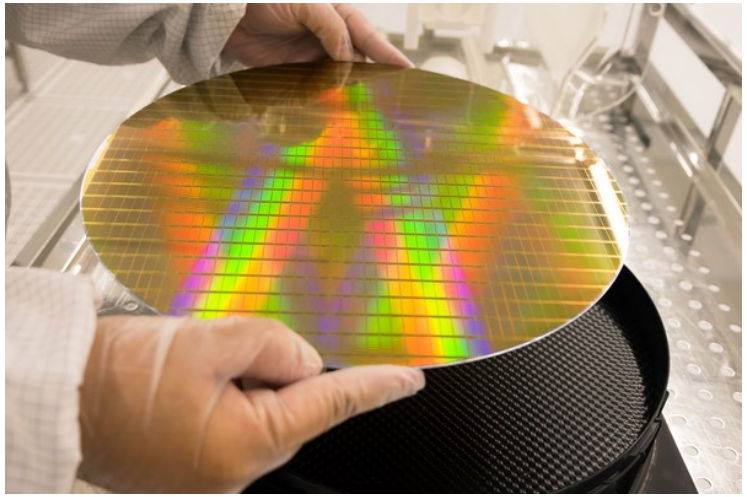


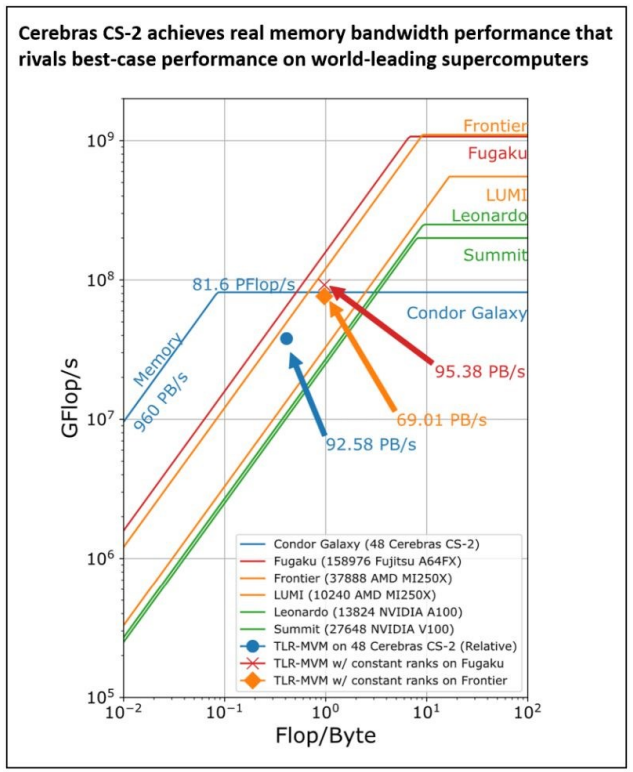
- Designed for AI workloads, but can program directly via CSL and we have developed a CSL tutorial
- Memory bandwidth makes this architecture especially useful for memory bound workloads
  - Transforming these from being memory to compute bound

epcc | THE UNIVERSITY OF EDINBURGH

Cerebras CS-2

- A single wafer, Wafer Scale Engine (WSE)
  - CS-2 has around 850,000 individual cores, along with 40GB of SRAM (distributed amongst the cores), and memory BW of 20 PB/s



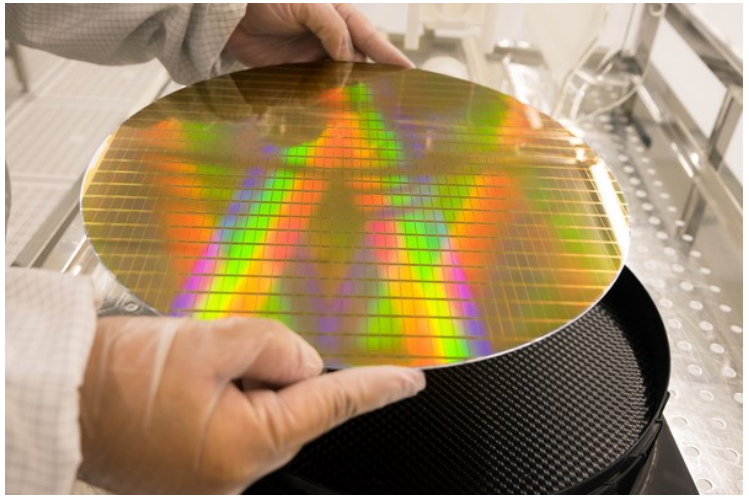


- Designed for AI workloads, but can program directly via CSL and we have developed a CSL tutorial
- Memory bandwidth makes this architecture especially useful for memory bound workloads
  - Transforming these from being memory to compute bound

epcc | THE UNIVERSITY OF EDINBURGH

But again: Programmability

- First solution is to port libraries to the CS-2
  - Focussed on PETSc on the CS-2
  - The interface works, but it's hard to write general high performance reusable code in this manner due to hardware constraints
  - Software stack is also developing very quickly
  - Best for long running codes rather than shorter kernels

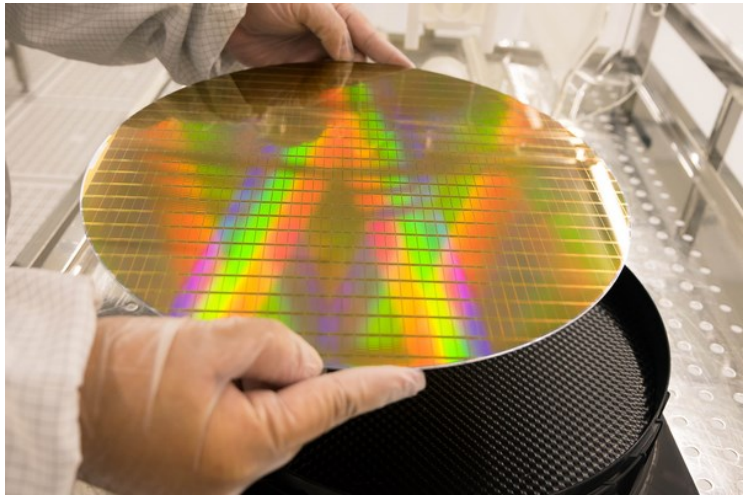


```python
#from petsc4py import PETSc
from petsc4wse import PETSc

...
x = PETSc.Vec().createSeq(m)
y = PETSc.Vec().createSeq(m)
A = PETSc.Mat().createDense([M,N])
x.setValues(range(m), range(m), addv=None)
...
result = x.sum()
A.mult(x,y) # y=A*x
...
```

```python
from petsc4py import PETSc

# Subclass the Vec class and override its sum function
# to do the work using the WSE
class Vec(PETSc.Vec):
  def sum(self):
    x = self.array

    calc_wse_distribution()
    compile_for_wse()
    initialise_wafer()
    memcpy_h2d(x, ...)
    wse_remote_launch("vec_sum")
    memcpy_d2h(x_sum, ...)

    return x_sum

# Assign modified functions to the original module
PETSc.Vec = Vec
```

But again: Programmability

- First solution is to port libraries to the CS-2
  - Focussed on PETSc on the CS-2
  - The interface works, but it's hard to write general high performance reusable code in this manner due to hardware constraints
  - Software stack is also developing very quickly
  - Best for long running codes rather than shorter kernels

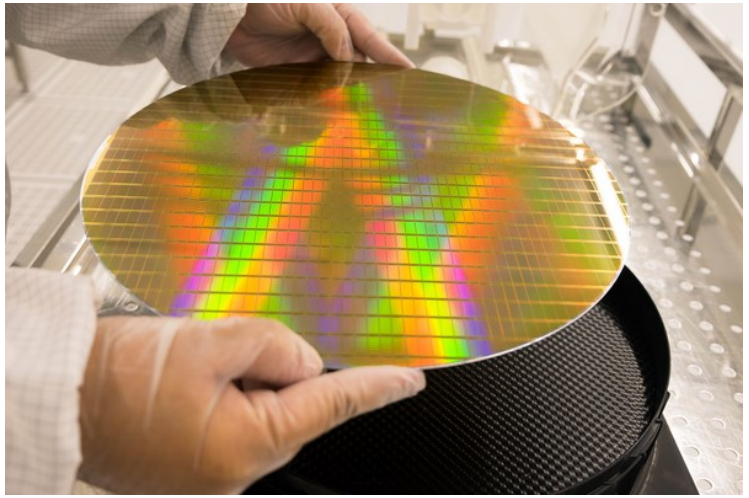


```python
#from petsc4py import PETSc
from petsc4wse import PETSc

...
x = PETSc.Vec().createSeq(m)
y = PETSc.Vec().createSeq(m)
A = PETSc.Mat().createDense([M,N])
x.setValues(range(m), range(m), addv=None)
...
result = x.sum()
A.mult(x,y) # y=A*x
...
```

```python
from petsc4py import PETSc

# Subclass the Vec class and override its sum function
# to do the work using the WSE
class Vec(PETSc.Vec):
  def sum(self):
    x = self.array

    calc_wse_distribution()
    compile_for_wse()
    initialise_wafer()
    memcpy_h2d(x, ...)
    wse_remote_launch("vec_sum")
    memcpy_d2h(x_sum, ...)

    return x_sum

# Assign modified functions to the original module
PETSc.Vec = Vec
```

But again: Programmability

- First solution is to port libraries to the CS-2
  - Focussed on PETSc on the CS-2
  - The interface works, but it's hard to write general high performance reusable code in this manner due to hardware constraints
  - Software stack is also developing very quickly
  - Best for long running codes rather than shorter kernels

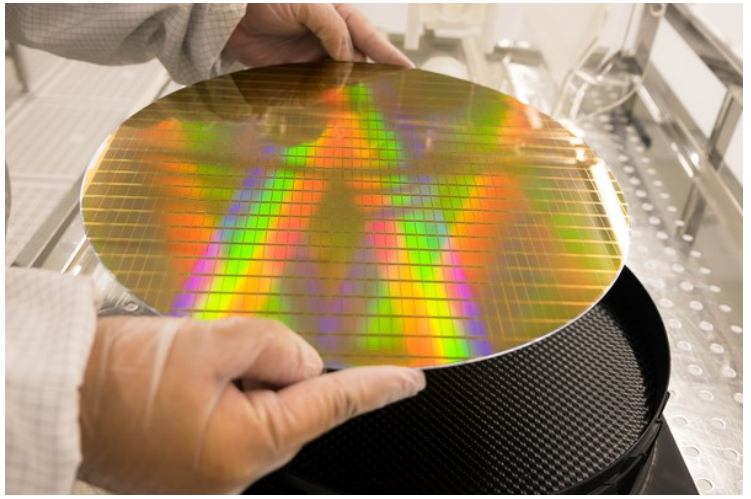


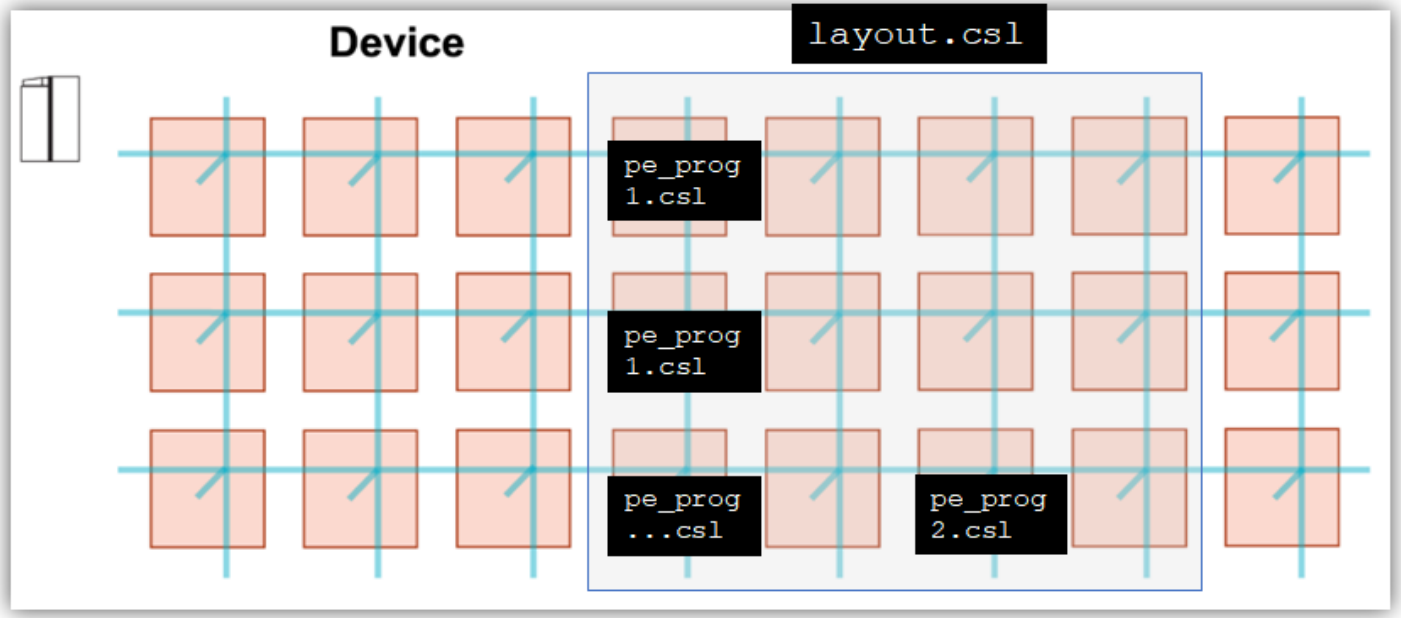
```python
#from petsc4py import PETSc
from petsc4wse import PETSc

...
x = PETSc.Vec().createSeq(m)
y = PETSc.Vec().createSeq(m)
A = PETSc.Mat().createDense([M,N])
x.setValues(range(m), range(m), addv=None)
...
result = x.sum()
A.mult(x,y) # y=A*x
...
```

```python
from petsc4py import PETSc

# Subclass the Vec class and override its sum function
# to do the work using the WSE
class Vec(PETSc.Vec):
  def sum(self):
    x = self.array

    calc_wse_distribution()
    compile_for_wse()
    initialise_wafer()
    memcpy_h2d(x, ...)
    wse_remote_launch("vec_sum")
    memcpy_d2h(x_sum, ...)

    return x_sum

# Assign modified functions to the original module
PETSc.Vec = Vec
```

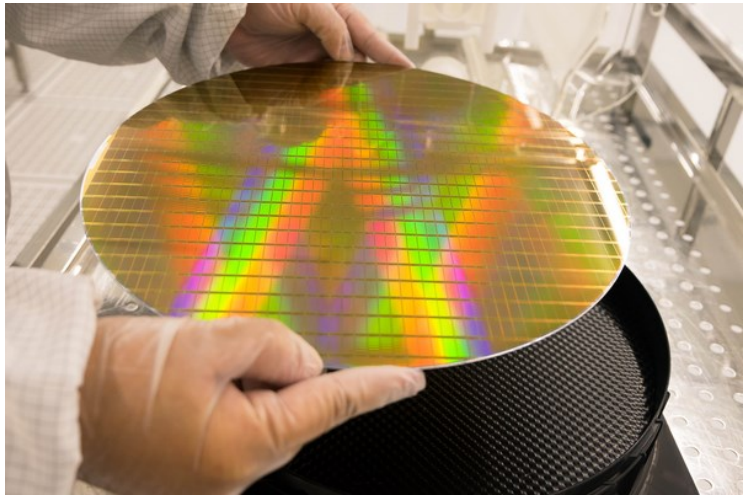
But again: Programmability

- Second solution is to handle in compiler
  - Similar to FPGAs and AIEs, have developed support in MLIR for the CS-2 in collaboration with xDSL
  - Generates the individual kernels and routing from source in a DSL or Fortran
  - This is currently work in progress, but is running end to end and gives performance comparable to hand crafted code

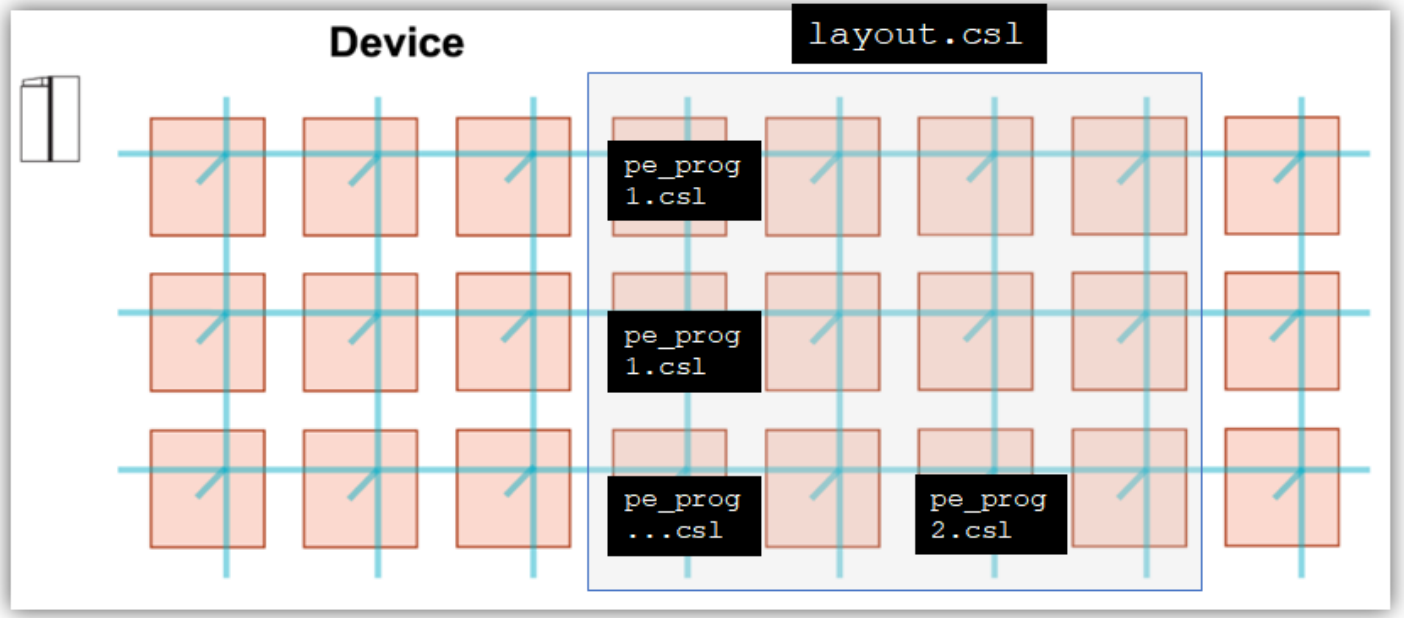




But again: Programmability



- Second solution is to handle in compiler
  - Similar to FPGAs and AIEs, have developed support in MLIR for the CS-2 in collaboration with xDSL
  - Generates the individual kernels and routing from source in a DSL or Fortran
  - This is currently work in progress, but is running end to end and gives performance comparable to hand crafted code

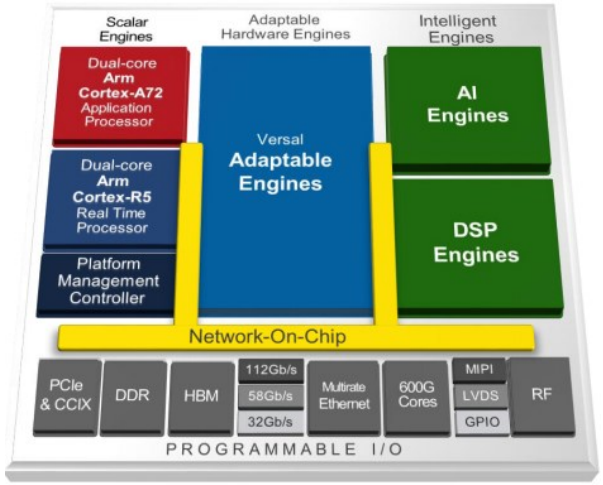


Summary

- Interesting architectures, fundamentally all built around the **paradigm of dataflow**
  - Fine-grained for FPGAs, course-grained for AIEs and CS-2

xDSL

https://xdsl.dev



- Can provide access to these machines if people are interested

epcc | THE UNIVERSITY OF EDINBURGH