Optimising Diverse Accelerator Designs

Jessica Vandebon, Gabriel Figueiredo, Ce Guo, **Wayne Luk**
Department of Computing
Imperial College London

WRC
21 January 2025

Cloud Computing

- rent computation/networking as needed
  - users: no need to acquire, maintain, decommission
  - dynamic allocation of resources
  - adapt to rapidly changing workloads

- based on large data centers
  - O(100K) servers, O(10MW) power

- benefits
  - economies of scale
  - improvements leveraged by all users
  - data, computation all in one place

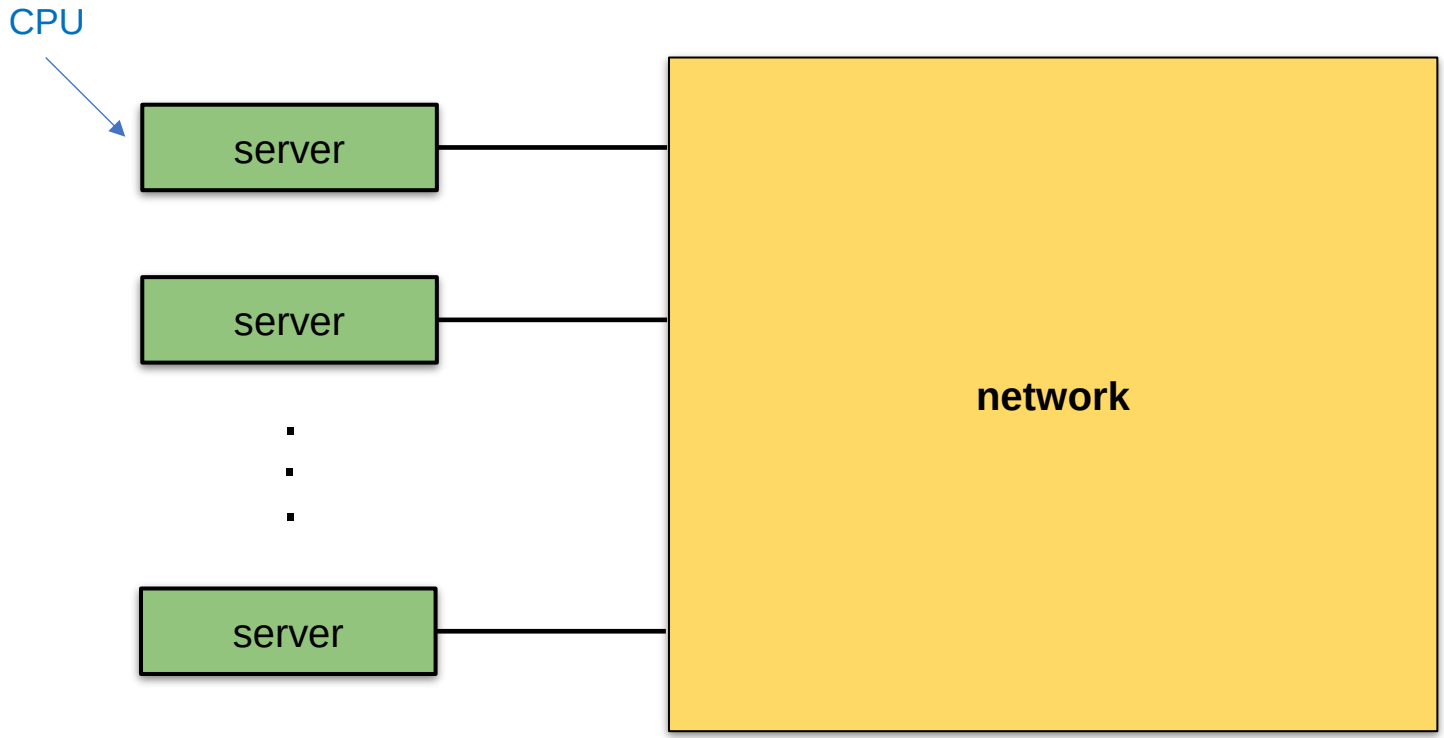


Source: D. Chiou 2018



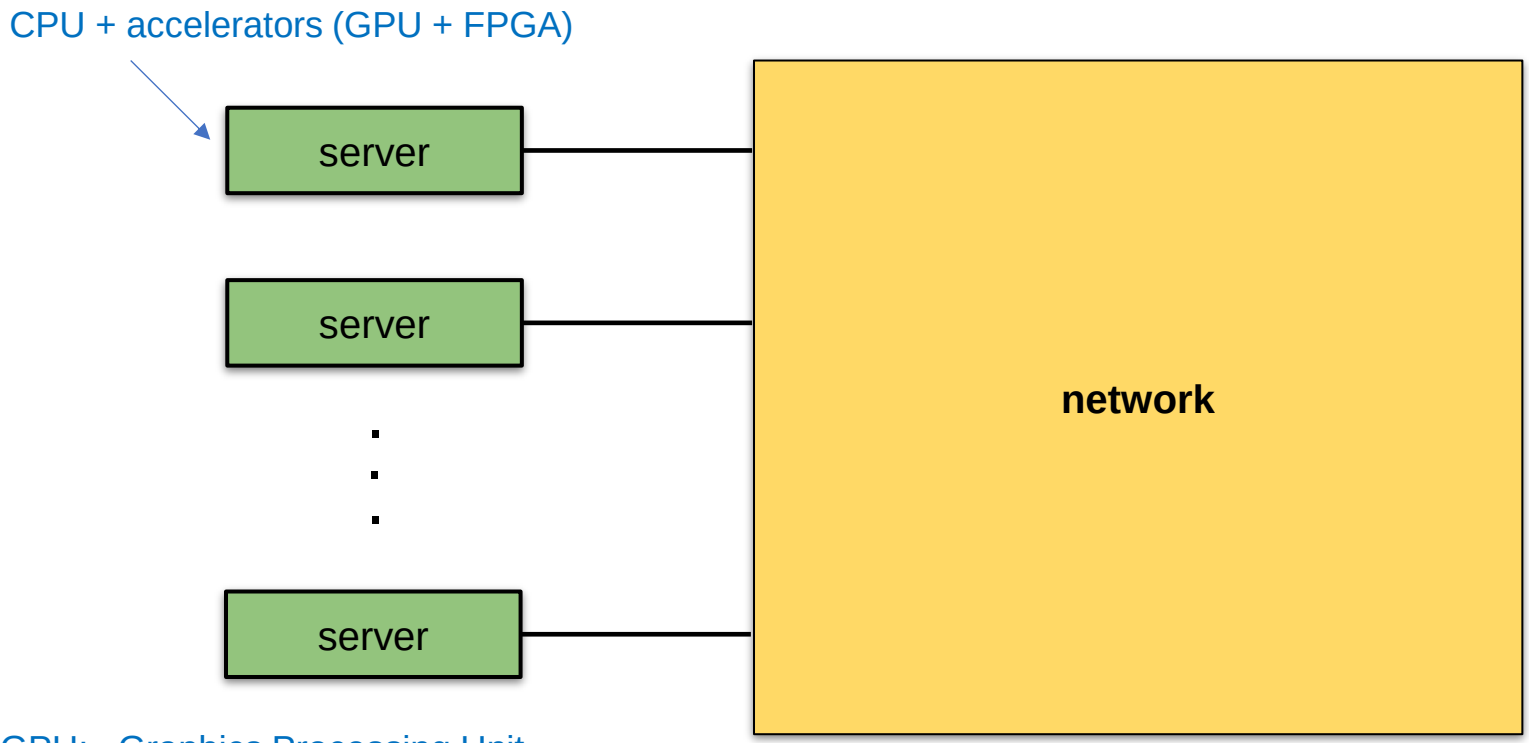


Cloud Computing System: Homogeneous



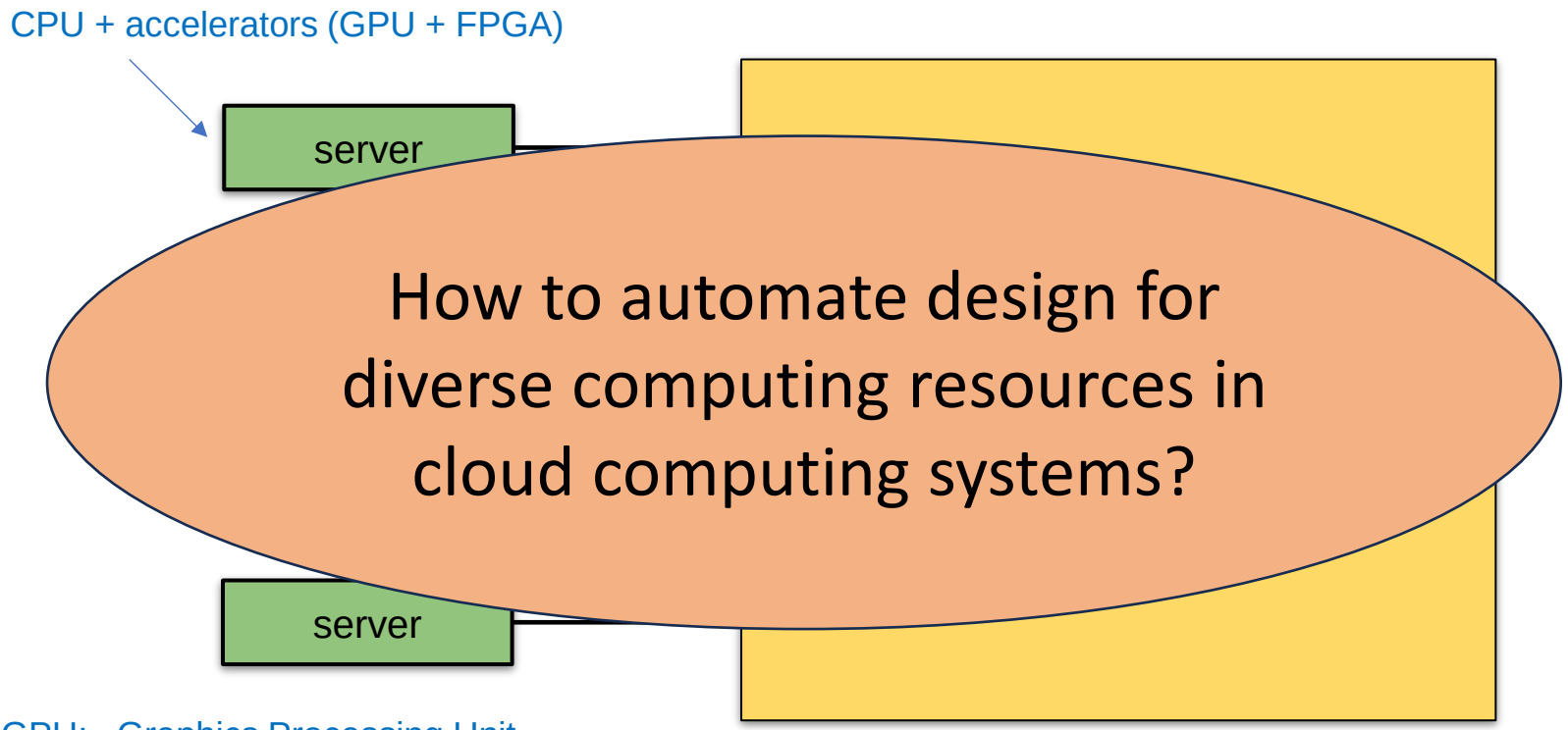
CPU: Central Processing Unit

Cloud Computing System: Heterogeneous



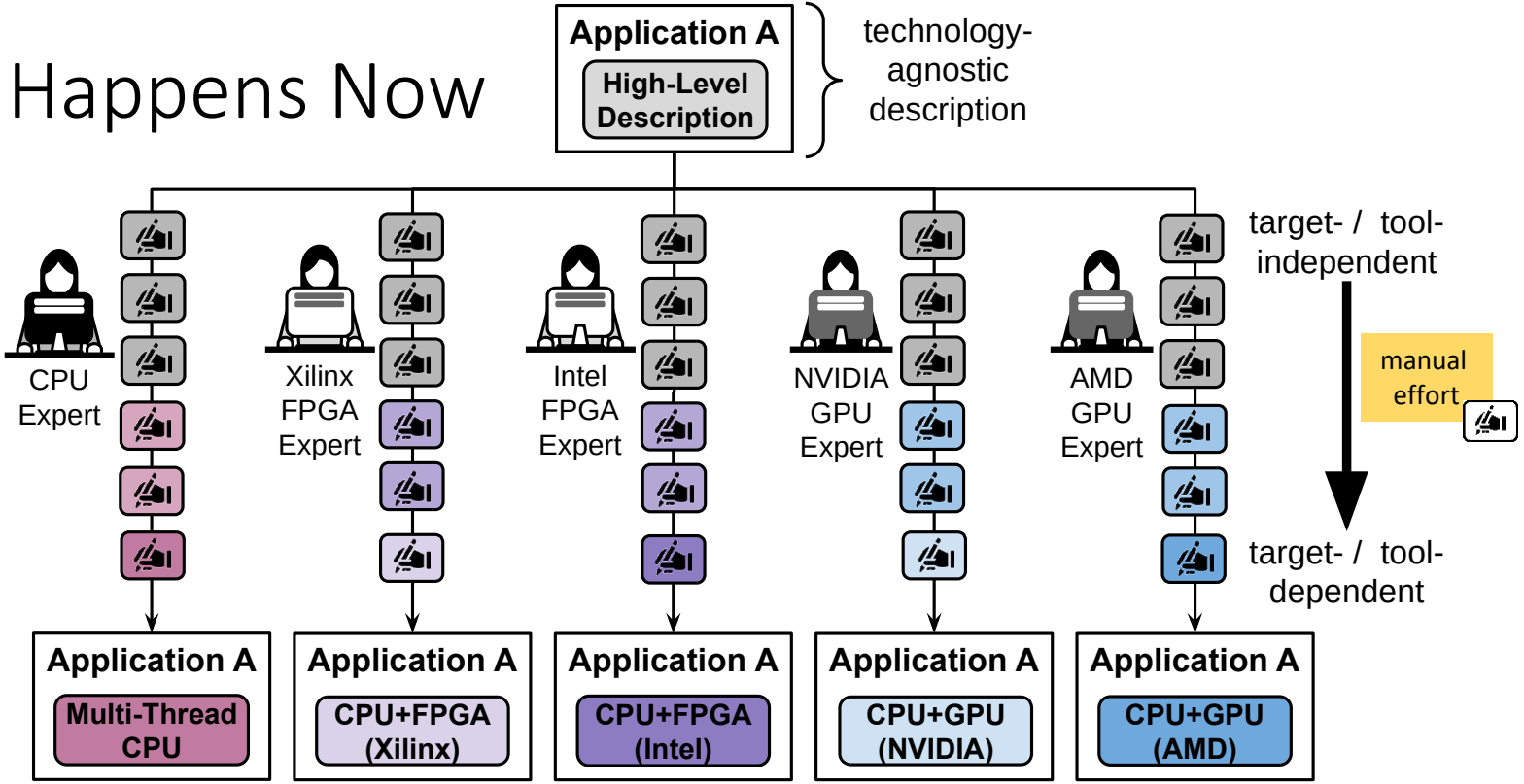
GPU: Graphics Processing Unit
FPGA: Field Programmable Gate Array

Cloud Computing System: Heterogeneous



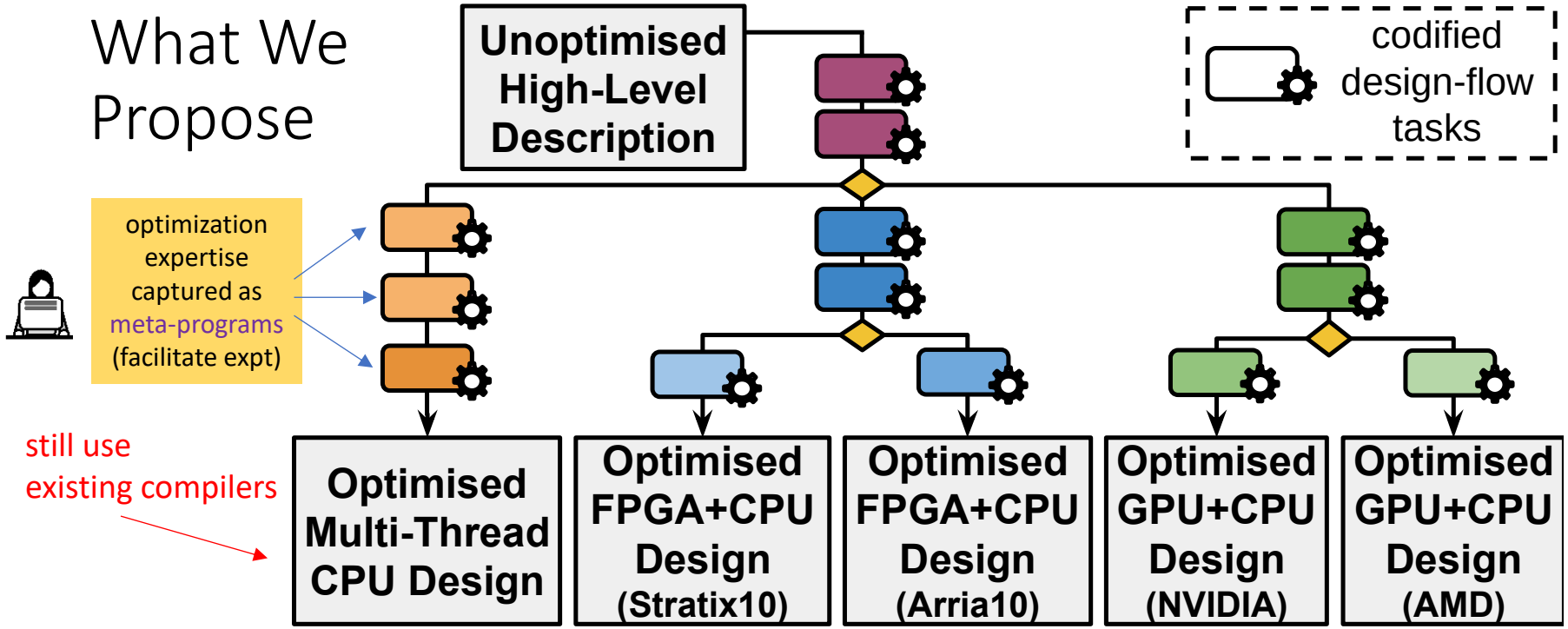
GPU: Graphics Processing Unit
FPGA: Field Programmable Gate Array

What Happens Now



- *disjoint design-flows* for diverse targets (i.e. CPUs, GPUs, FPGAs)
- optimization requires substantial *expertise*, *experience*, *manual effort*

What We Propose



- automate design-flows from high-level application description
- generate designs for diverse hardware targets automatically
- maximize re-use of designs and optimizations

Automated Design-Flow Requirements

R1) ***scope:*** cover full application (i.e. host and device)

R2) ***modularity:*** component building-blocks with uniform interfaces

R3) ***reusable:*** apply to multiple applications

R4) ***abstractions:*** mirror human-level artifacts

R5) ***automated decisions:*** specialization at design-flow branch points, support diverse targets and strategies

Key Idea



meta-program
optimization
expertise
encoded as
a program
(currently Python, covering
static and dynamic processing)

```
ast <- Ast('source.cpp');
loops <- ast.query('l:ForLoop',where=l.fixed_bound)
while report['LUT'] < 100:
  n <- n*2
  for l in loops:a
    l.instrument(before, '#pragma unroll $n')
  report = ast.exec(mode='fpga_report')
```

meta-program

code analysis
src-to-src
instrumentation

tool config.
tool reports

configure,
execute
profile,
monitor

C++ description
Source-Code

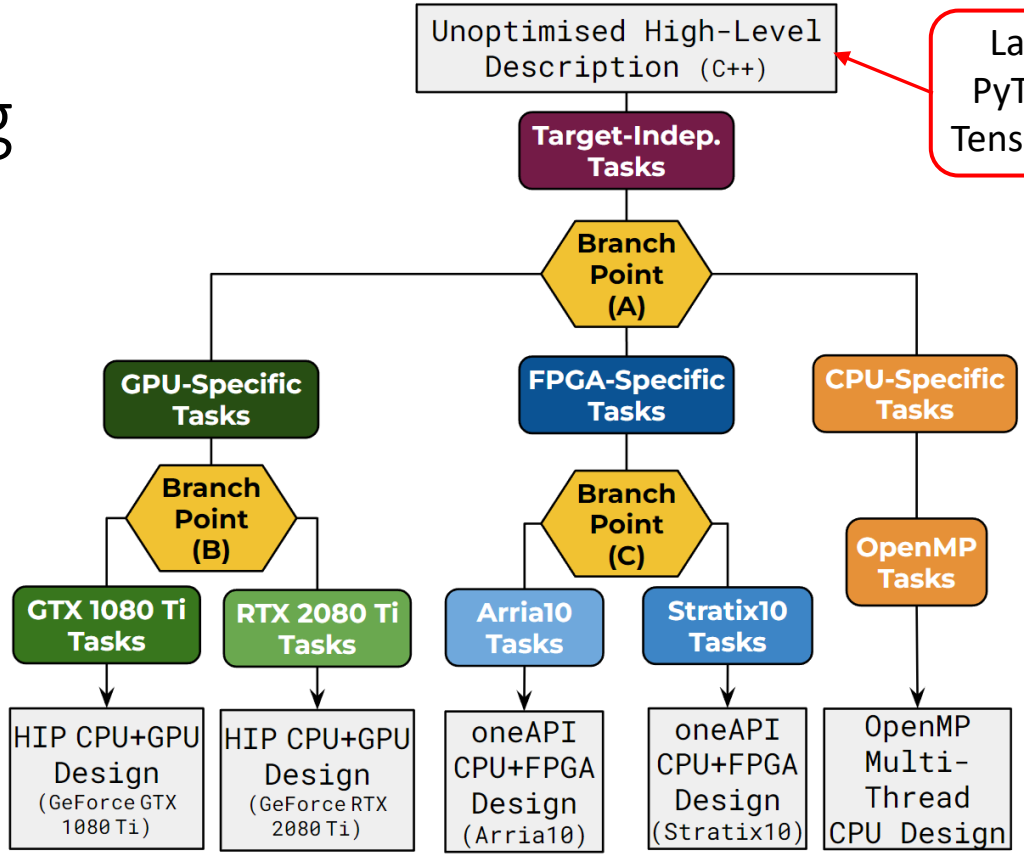
compiler | sim
Tools

OS | runtime | HW
Platforms

feedback

- **Artisan** meta-programs: programmatic access to source-code, tools
- enable codification of optimization strategies:
  - *transformations, analyses, instrumentation, tool / platform configuration, monitoring*

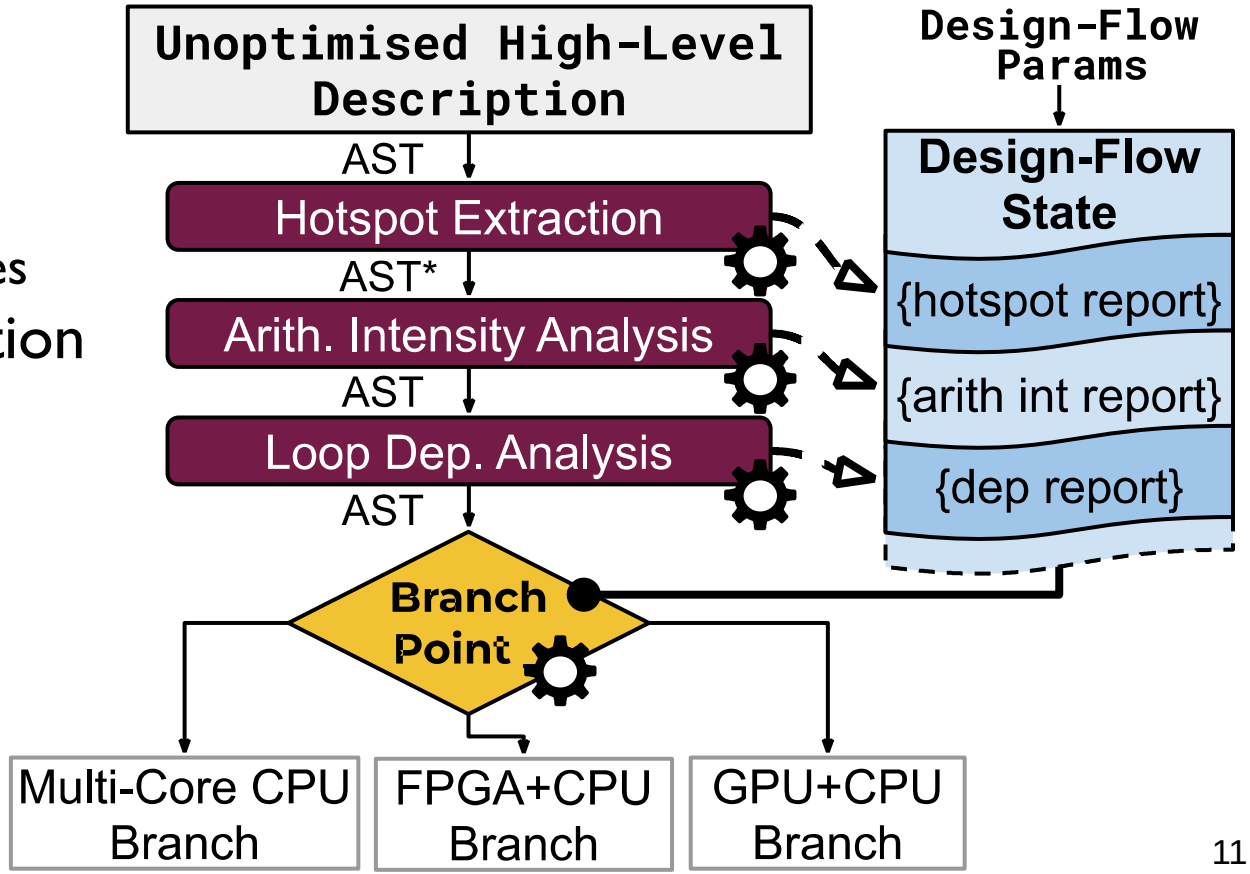
Meta-Programming for Diverse Design



- codify recurring ***design-flow patterns*** using Artisan meta-programs
  - reusable building blocks; target-independent → target-dependent
  - 3 pattern types: analysis, transform, optimization

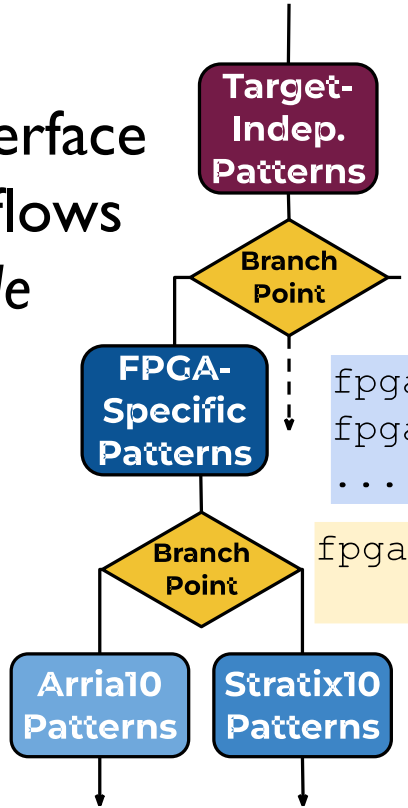
Path Selection Automation: PSA

- at branch points
- decisions based on
  - static/dynamic analyses
- multi-level specialisation
  - target type
  - vendor
  - optimization strategy



Automate: Compose Patterns + Branch Points

- high-level Python interface
- application-agnostic flows
- *arbitrary high-level code*




```
df = design_flow()
```

```
df.add_pattern(identify_hs_loops,
    {'filter':parallel,'thres':.6})
... # add target-indep patterns
```

```
df.add_branch(target_selection_fn,
    [fpga_df, gpu_df, cpu_df])
```

```
fpga_df = design_flow()
fpga_df.add_pattern(gen_oneapi_design)
... # add fpga-specific patterns
```

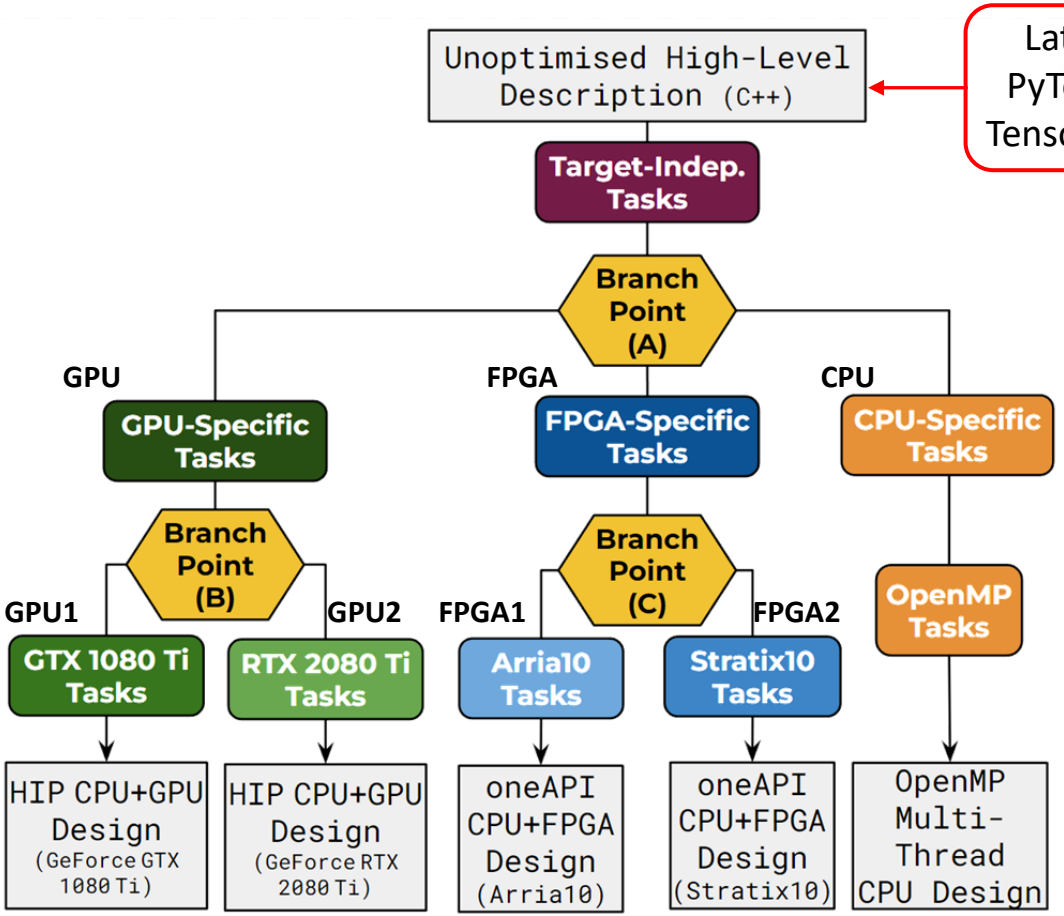
```
fpga_df.add_branch(device_selection_fn,
                   [s10_df, a10_df])
```

```
df.run('/path/to/src/main.cpp',
       '/path/to/dest/',
       params={'target':'FPGA',
               'device':'S10'})
```

Meta-Programming with Path Selection Automation

Repository of Codified Design-Flow Tasks

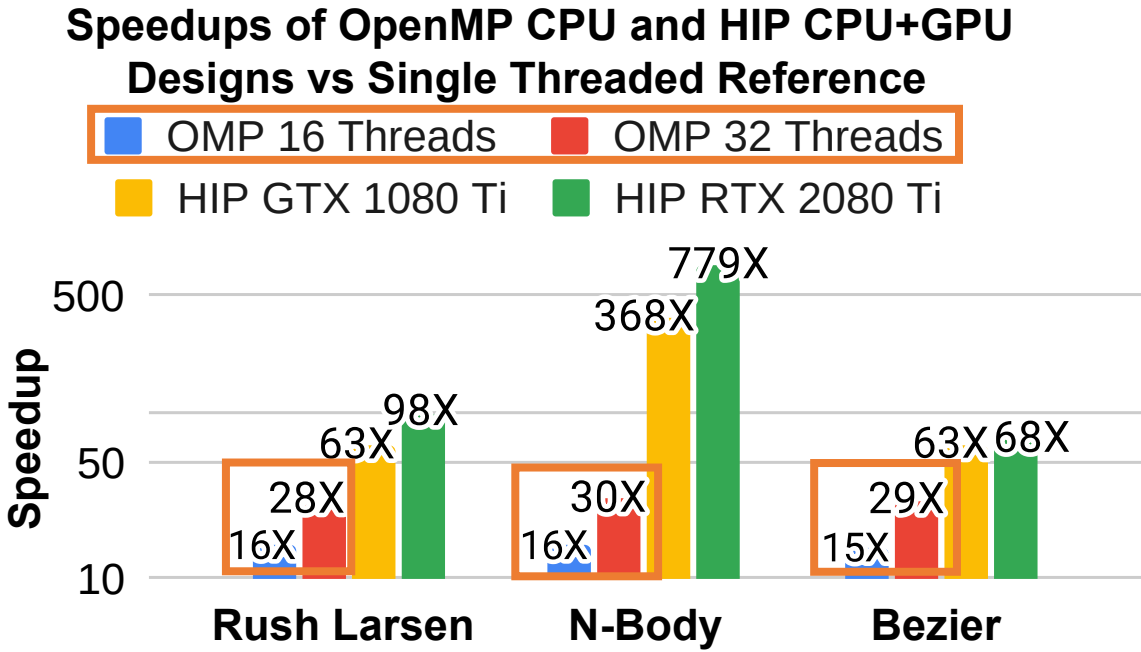
T-INDEP	Identify Hotspot Loops	A
T-INDEP	Hotspot Loop Extraction	T
T-INDEP	Pointer Analysis	A
T-INDEP	Arithmetic Intensity Analysis	A
T-INDEP	Data In/Out Analysis	A
T-INDEP	Loop Dependence Analysis	A
T-INDEP	Loop Trip-Count Analysis	A
T-INDEP	Remove Array += Dependency	T
FPGA	Generate oneAPI Design	CG
FPGA	Unroll Fixed Loops	T
FPGA	Employ SP Math Fns*	T
FPGA	Employ SP Numeric Literals*	T
FPGA-A10	A10 Unroll Until Overmap DSE	O
FPGA-S10	Zero-Copy Data Transfer	T
FPGA-S10	S10 Unroll Until Overmap DSE	O
GPU	Generate HIP Design	CG
GPU	Employ HIP Pinned Memory	T
GPU	Employ SP Math Fns*	T
GPU	Employ SP Numeric Literals*	T
GPU	Introduce Shared Mem Buf	T
GPU	Employ Specialised Math Fns	T
GPU-1080	GTX 1080 Blocksize DSE	O
GPU-2080	RTX 2080 Blocksize DSE	O
CPU-OMP	Multi-Thread Parallel Loops	T
CPU-OMP	OMP Num. Threads DSE	O



Results: CPU, GPU Performance

- applied implemented design-flow to 5 AI/HPC applications
- 3 designs for each: multi-thread CPU, CPU+FPGA, CPU+GPU

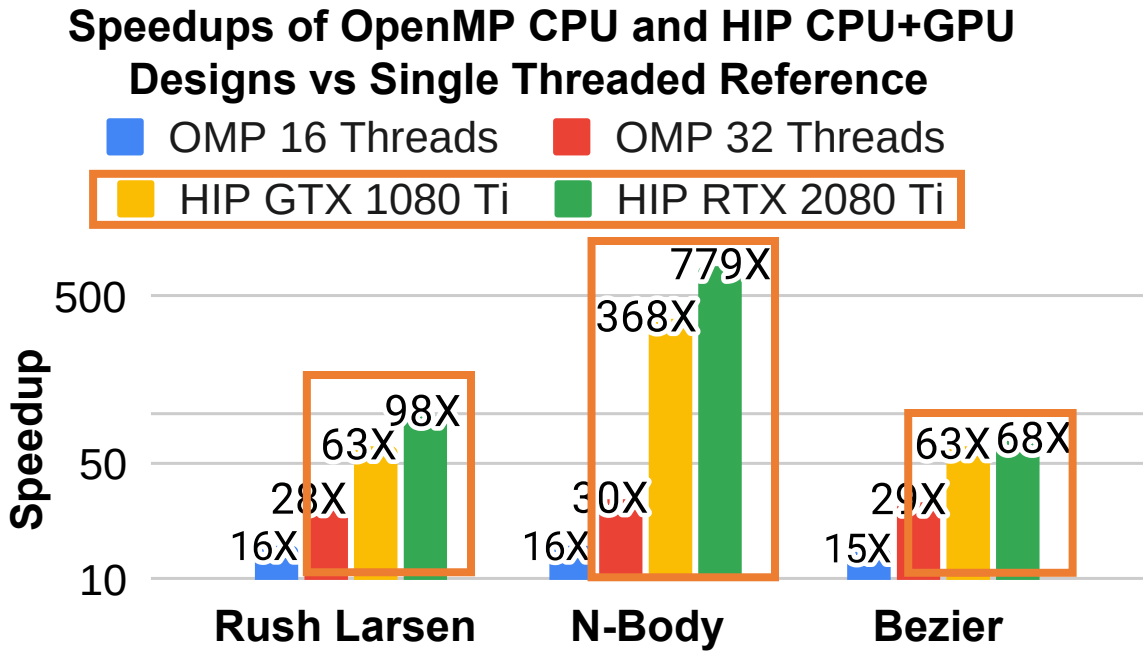
- *CPU, GPU performance:*
  - *up to 30X for OpenMP multi-thread CPU (32 cores)*



Results: CPU, GPU Performance

- applied implemented design-flow to 5 AI/HPC applications
- 3 designs for each: multi-thread CPU, CPU+FPGA, CPU+GPU

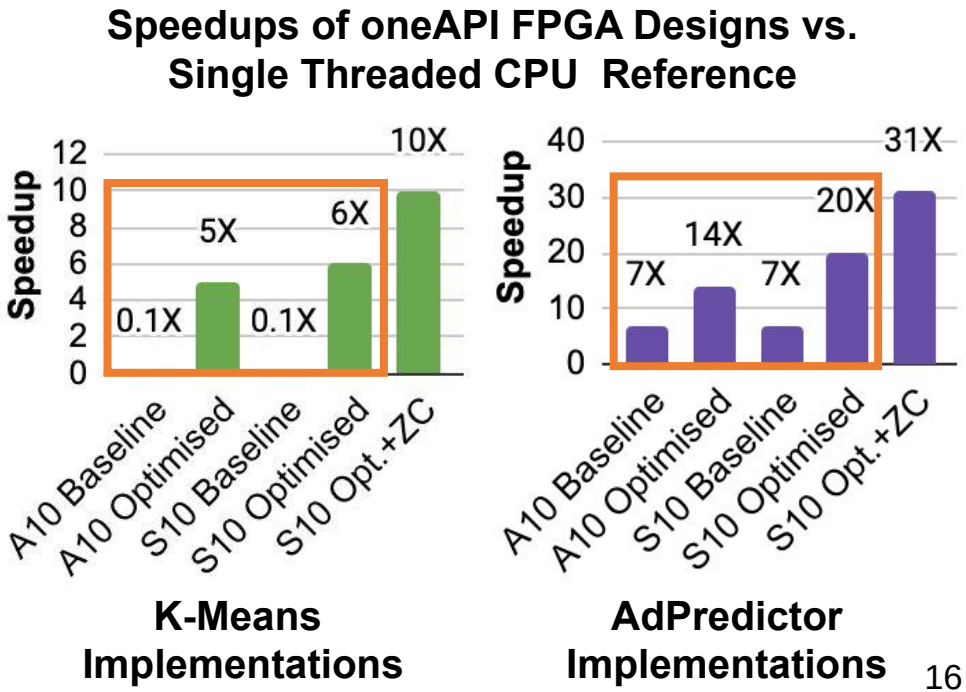
- *CPU, GPU performance:*
  - *up to 30X for OpenMP multi-thread CPU (32 cores)*
  - *up to 779X for HIP CPU+GPU (GeForce RTX 2080 Ti)*



Results: FPGA Performance

- applied implemented design-flow to 5 AI/HPC applications
- 3 designs for each: multi-thread CPU, CPU+FPGA, CPU+GPU

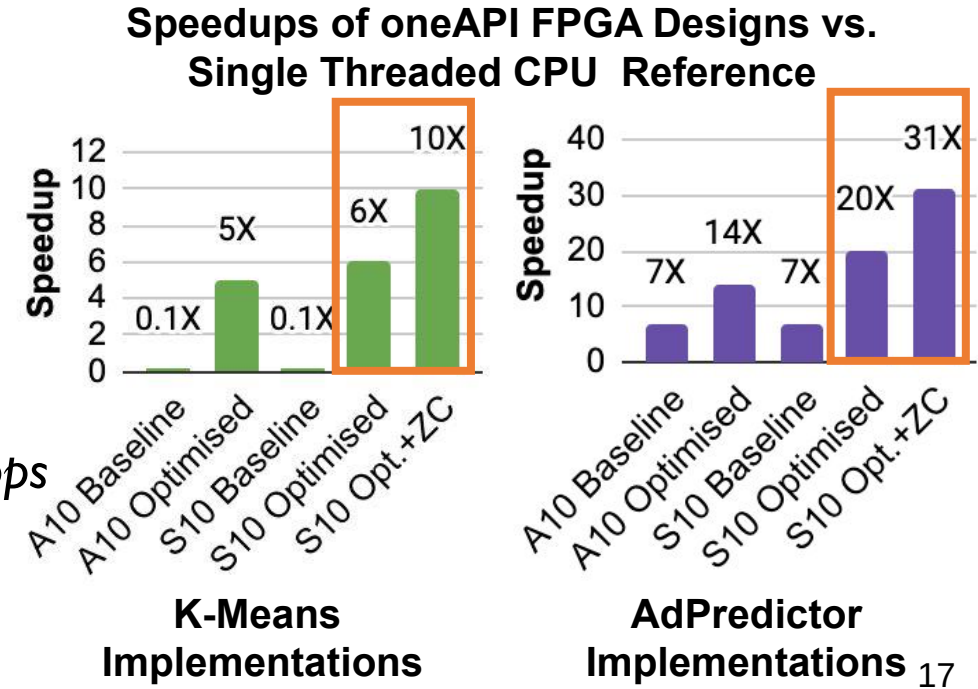
- *FPGA performance:*
  - *baseline A10, S10 versions with unmodified code vs generated designs:*
    - *50-60X speed for k-means*
    - *2-3X speed for AdPredictor*



Results: FPGA Performance

- applied implemented design-flow to 5 AI/HPC applications
- 3 designs for each: multi-thread CPU, CPU+FPGA, CPU+GPU

- *FPGA performance:*
  - *baseline A10, S10 versions with unmodified code vs generated designs:*
    - *50-60X speed for k-means*
    - *2-3X speed for AdPredictor*
  - *S10 branch + zero-copy data transfer*
    - *1.6-1.7X improvement across both apps*
    - *zero-copy: transfer from source to des*



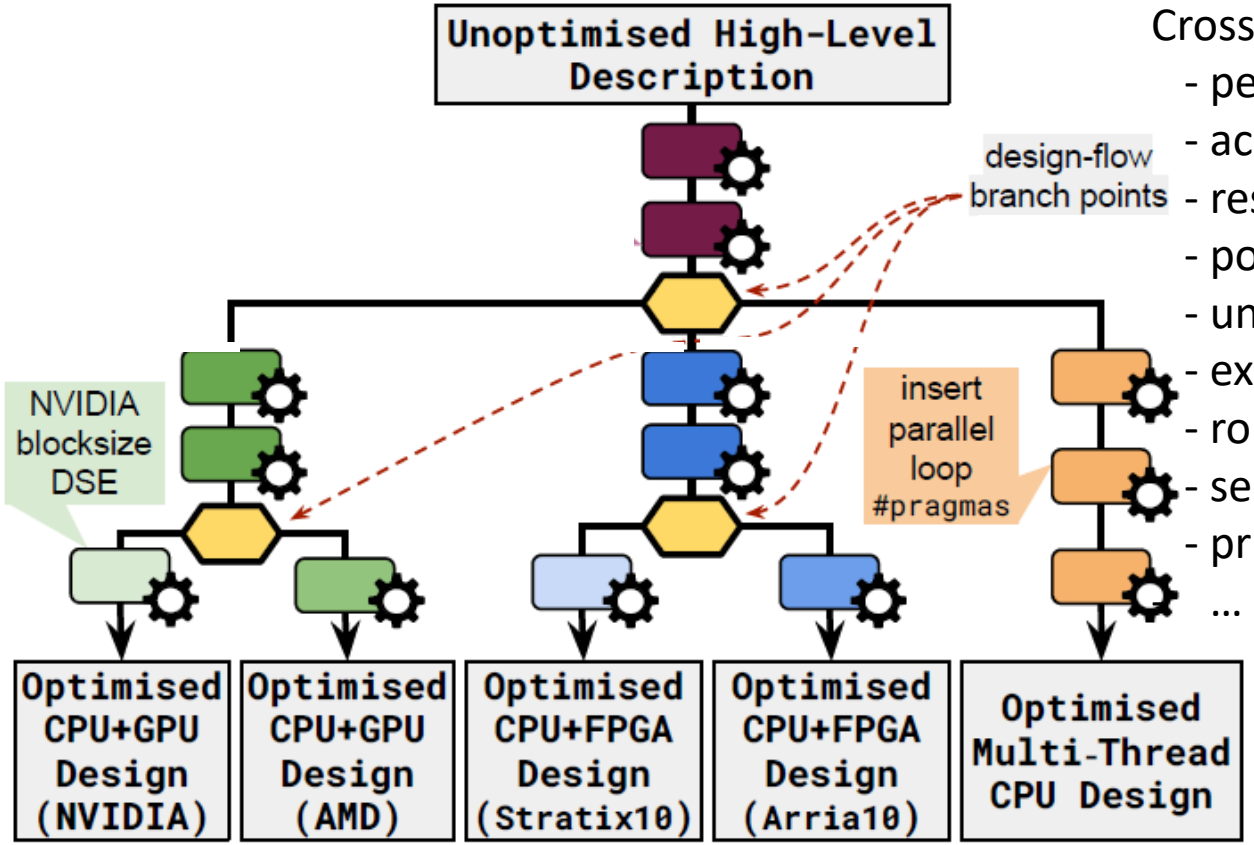
Productivity Analysis

- applied implemented design-flow to 5 AI/HPC applications
- 3 designs for each: multi-thread CPU, CPU+FPGA, CPU+GPU

Application	OpenMP LOC Δ	GPU+CPU LOC Δ	FPGA+CPU LOC Δ	Total LOC Δ
Rush Larsen	+.4%	+6%	n/a	n/a
N-Body Sim.	+2%	+44%	+58%	+104%
Bezier Blend	+2%	+43%	+49%	+94%
AdPredictor	+2%	+51%	+59%	+112%
K-Means	+4%	+103%	+124%	+231%
Average LOC Δ	**+2%**	**+49%**	**+72%**	**+135%**

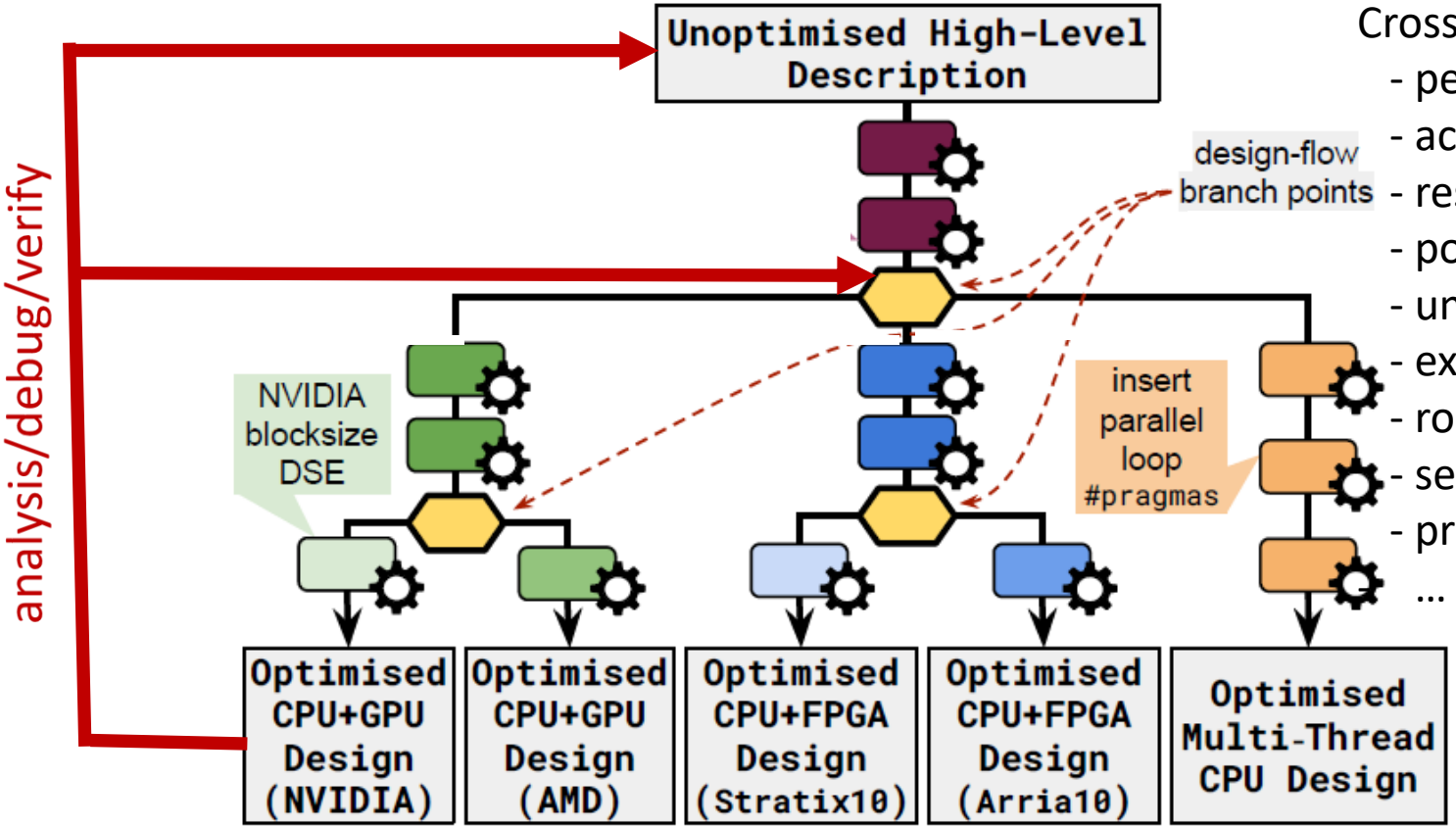
- *Productivity in LOC: Lines of Code*
  - average LOC for 3 new designs: +135%
  - total LOC added: ~5X design-flow code LOC

Future: High-Level Description to Optimized Designs



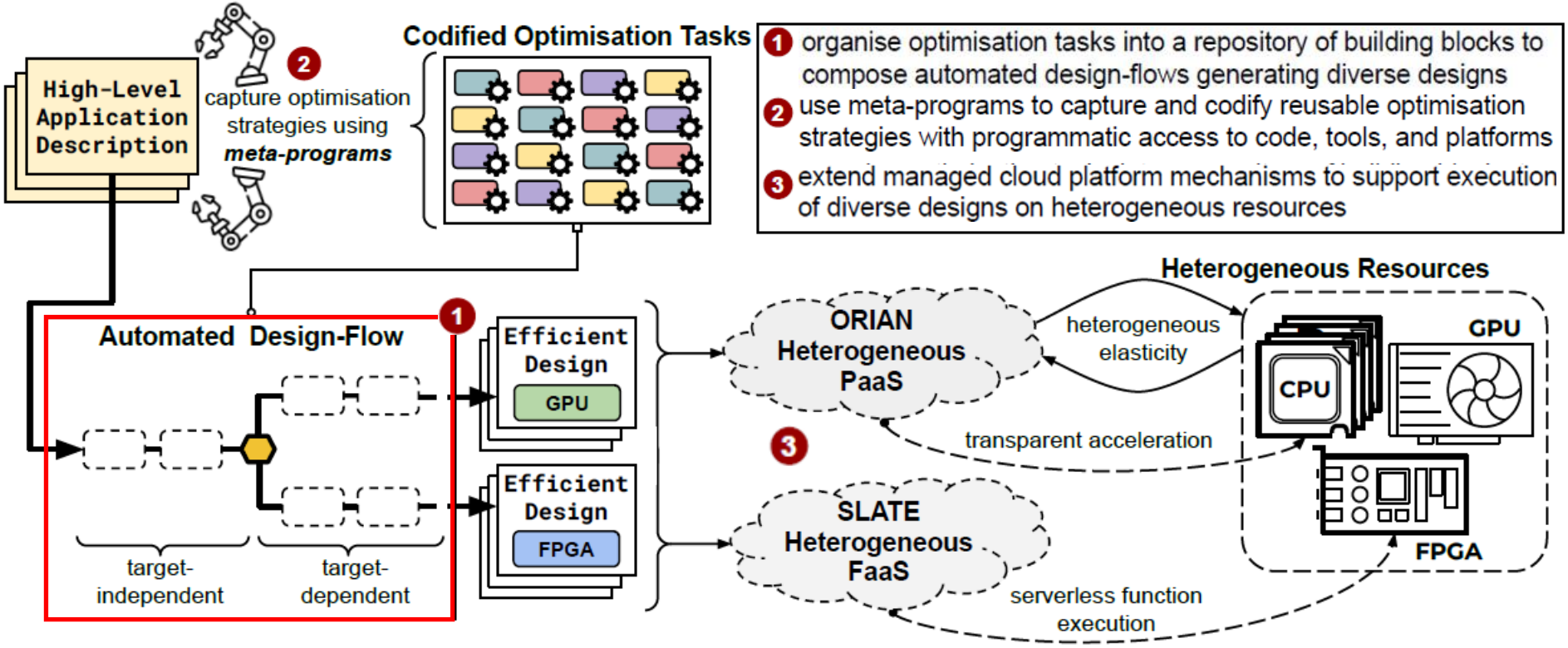
Cross-stack trade-offs:
- performance
- accuracy
- resource usage
- power/energy
- uncertainty estim.
- explainability
- robustness
- security
- privacy
- …

Future: High-Level Description to Optimized Designs



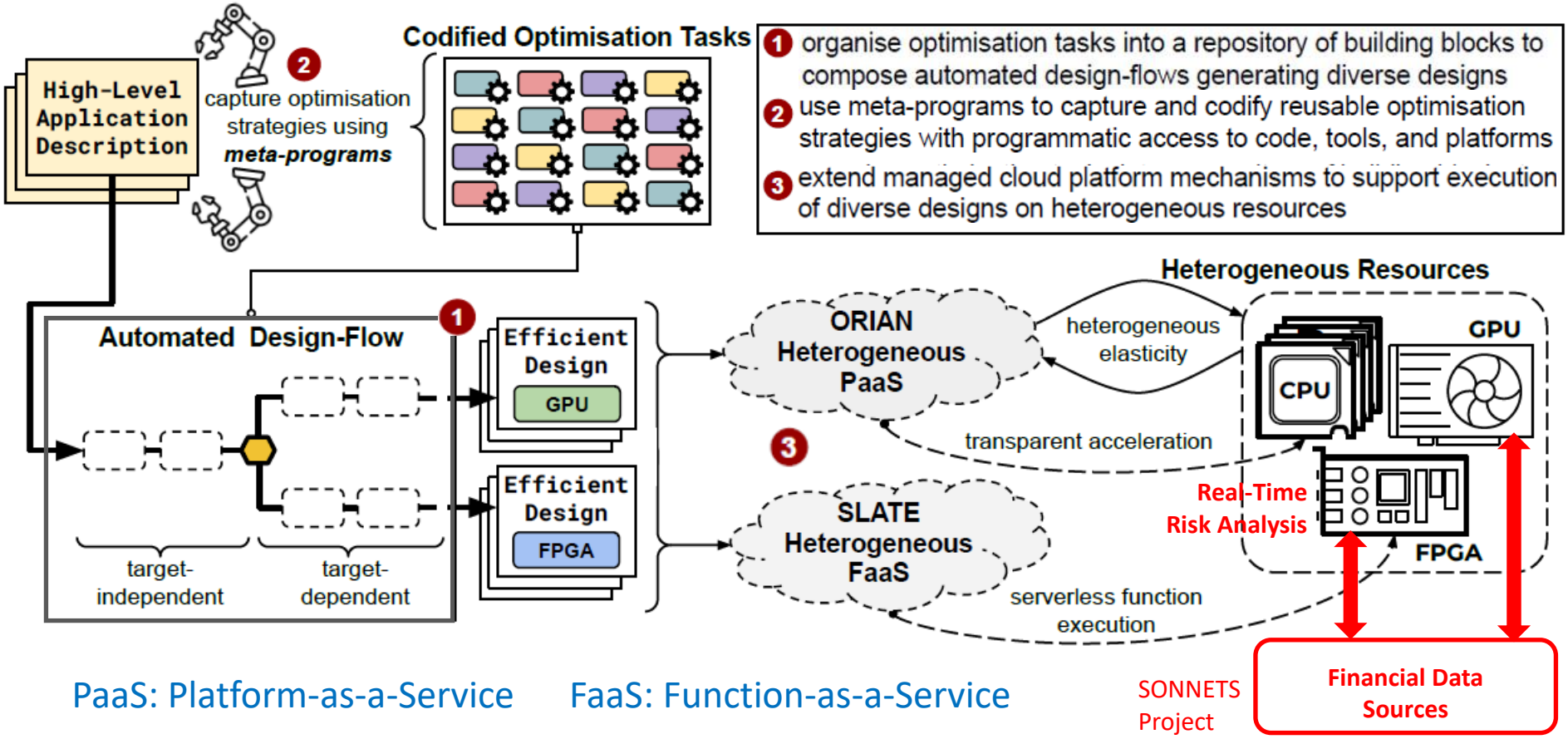
Cross-stack trade-offs:
- performance
- accuracy
- resource usage
- power/energy
- uncertainty estim.
- explainability
- robustness
- security
- privacy
- …

Future: Heterogeneous PaaS and FaaS Cloud Systems



PaaS: Platform-as-a-Service FaaS: Function-as-a-Service

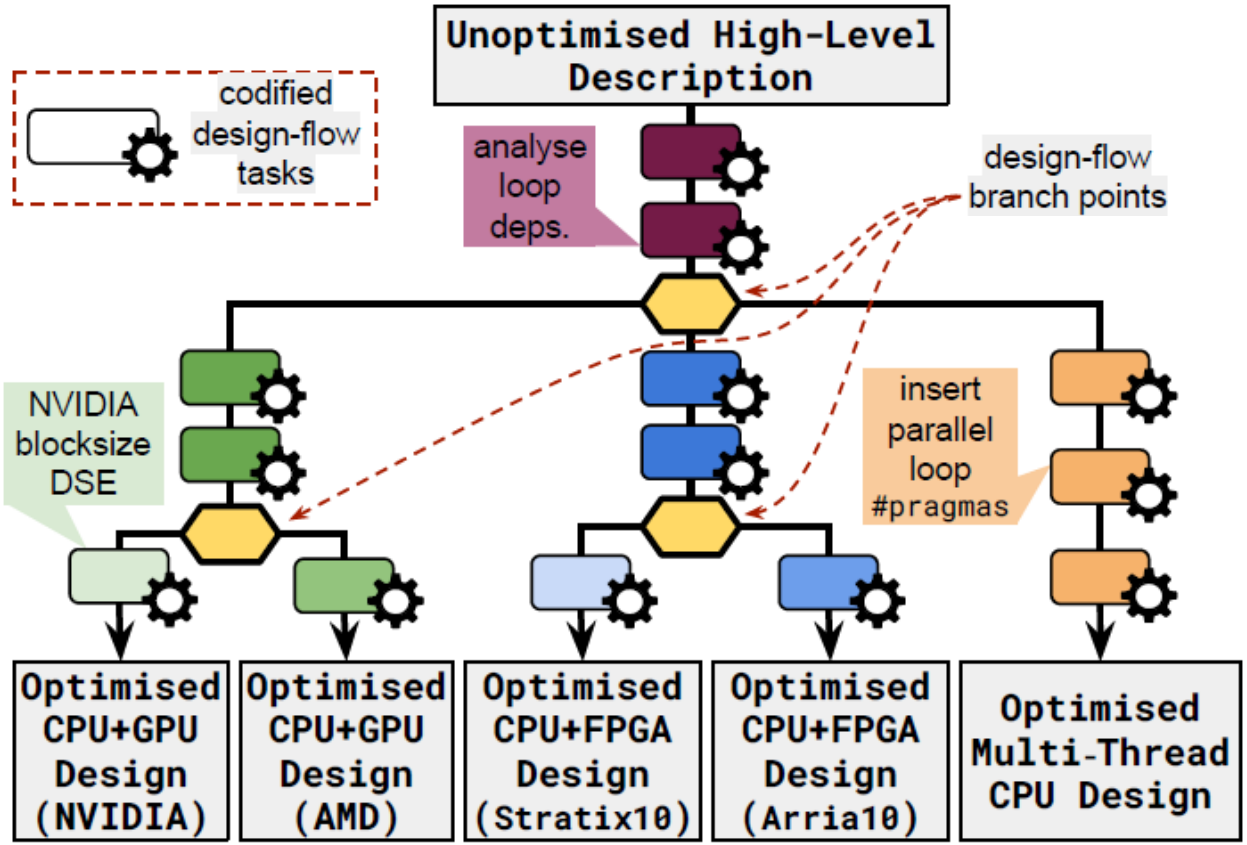
Future: Heterogeneous PaaS and FaaS Cloud Systems



PaaS: Platform-as-a-Service FaaS: Function-as-a-Service

SONNETS Project

Summary



- meta-program in Python controls design-flow
- codify optimisation strategies for diverse targets
- support static and dynamic analysis and transformation
- for heterogeneous cloud computing