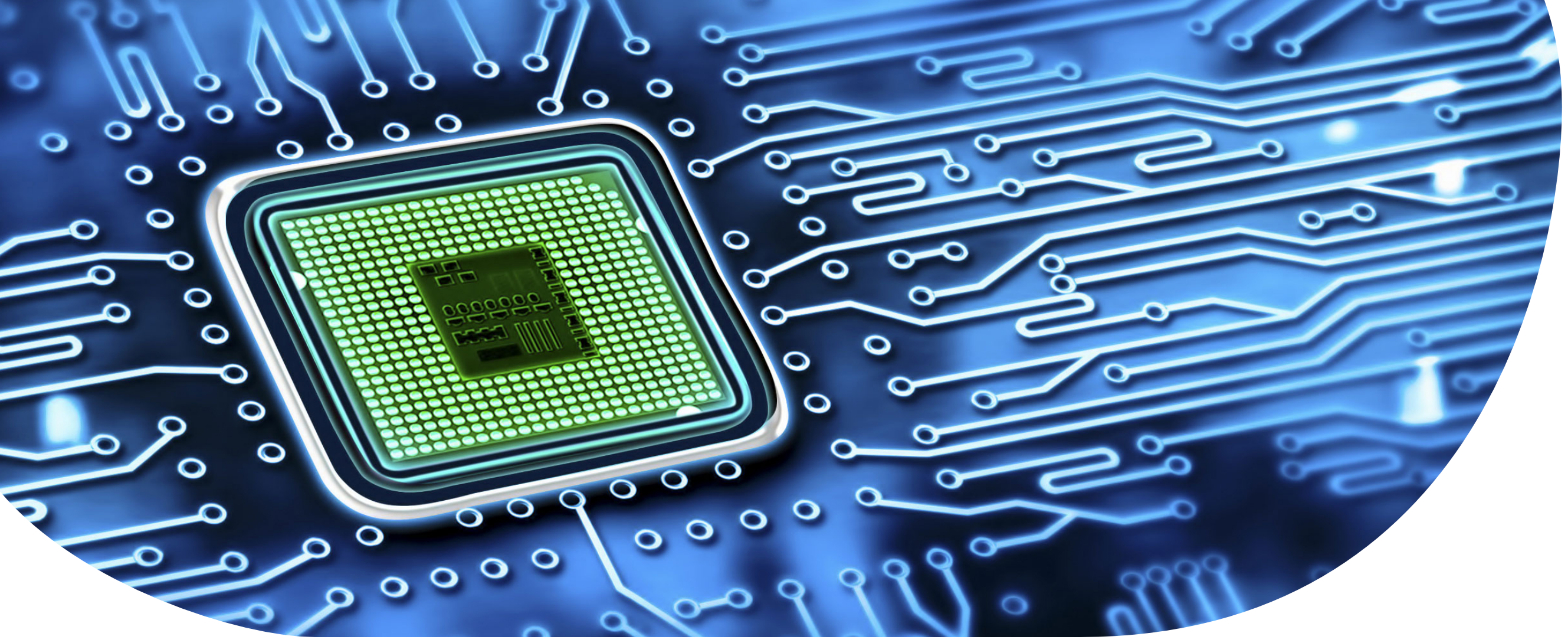


جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

It's all about the primitives! Designing for high performance on FPGAs

NANDA Workshop, Imperial, London, 9 Sep 2024

Suhaib A Fahmy
suhaib.fahmy@kaust.edu.sa

The prototyping trap

- *“FPGAs are the ideal platform for prototyping architectural ideas”*

The prototyping trap

- *“FPGAs are the ideal platform for prototyping architectural ideas”*
- *“We demonstrate a prototype of our new NIC design on an FPGA”*

The prototyping trap

- *“FPGAs are the ideal platform for prototyping architectural ideas”*
- *“We demonstrate a prototype of our new NIC design on an FPGA”*
- *“Our architecture achieves a clock frequency of 200 MHz on ___ FPGA”*

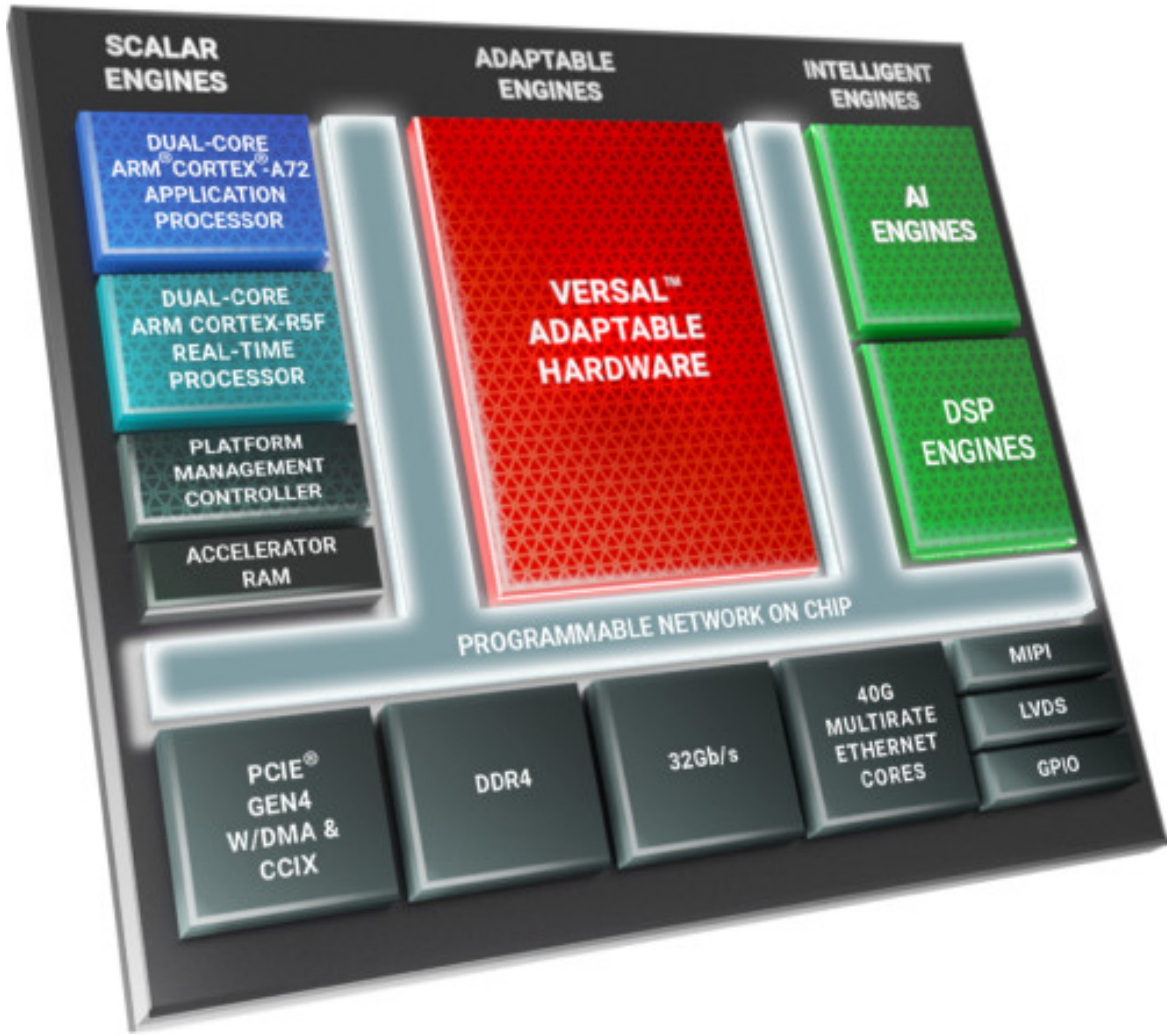
The prototyping trap

- *"FPGAs are the ideal platform for prototyping architectural ideas"*
- *"We demonstrate a prototype of our new NIC design on an FPGA"*
- *"Our architecture achieves a clock frequency of 200 MHz on ___ FPGA"*

- **What are we missing?**

These feisty architectures

- Modern FPGAs are a **complex mix** of capable computing resources, flexible routing, and high bandwidth I/O
- Perhaps we can leverage the design effort put into them to build more performant systems
- Let's also think about embodied carbon…

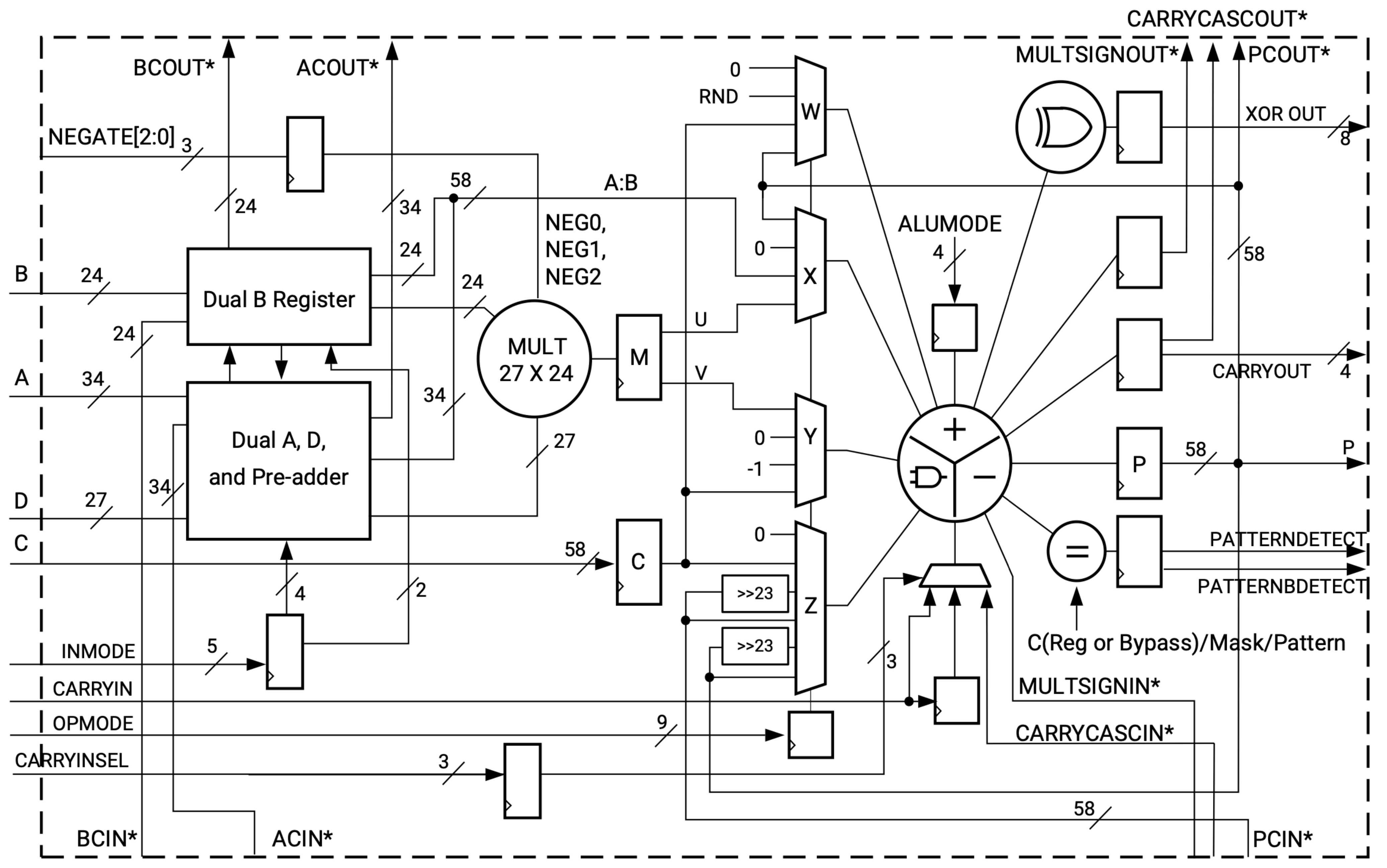

SCALAR ENGINES
ADAPTABLE ENGINES
INTELLIGENT ENGINES
DUAL-CORE ARM® CORTEX®-A72 APPLICATION PROCESSOR
DUAL-CORE ARM CORTEX-R5F REAL-TIME PROCESSOR
PLATFORM MANAGEMENT CONTROLLER
ACCELERATOR RAM
VERSAL™ ADAPTABLE HARDWARE
AI ENGINES
DSP ENGINES
PROGRAMMABLE NETWORK ON CHIP
PCIE® GEN4 W/DMA & CCIX
DDR4
32Gb/s
40G MULTIRATE ETHERNET CORES
MIPI
LVDS
GPIO

The evolution of the DSP Block

- FPGAs found widespread use in DSP applications so architectures evolved to support this
- Xilinx Virtex-II first introduced 18×18-bit multiplier at 105 MHz
- Virtex-4 added a 48-bit adder/subtractor/accumulator at 500MHz and programmable input muxes to ALU
- Virtex-5 expanded to 25×18-bit multiplier and added logic functions and dynamic programmability to ALU at 550 MHz
- Virtex-6 and all 7-series added programmable pre-adders and input registers up to 600 MHz , the DSP48E1

The evolution of the DSP Block

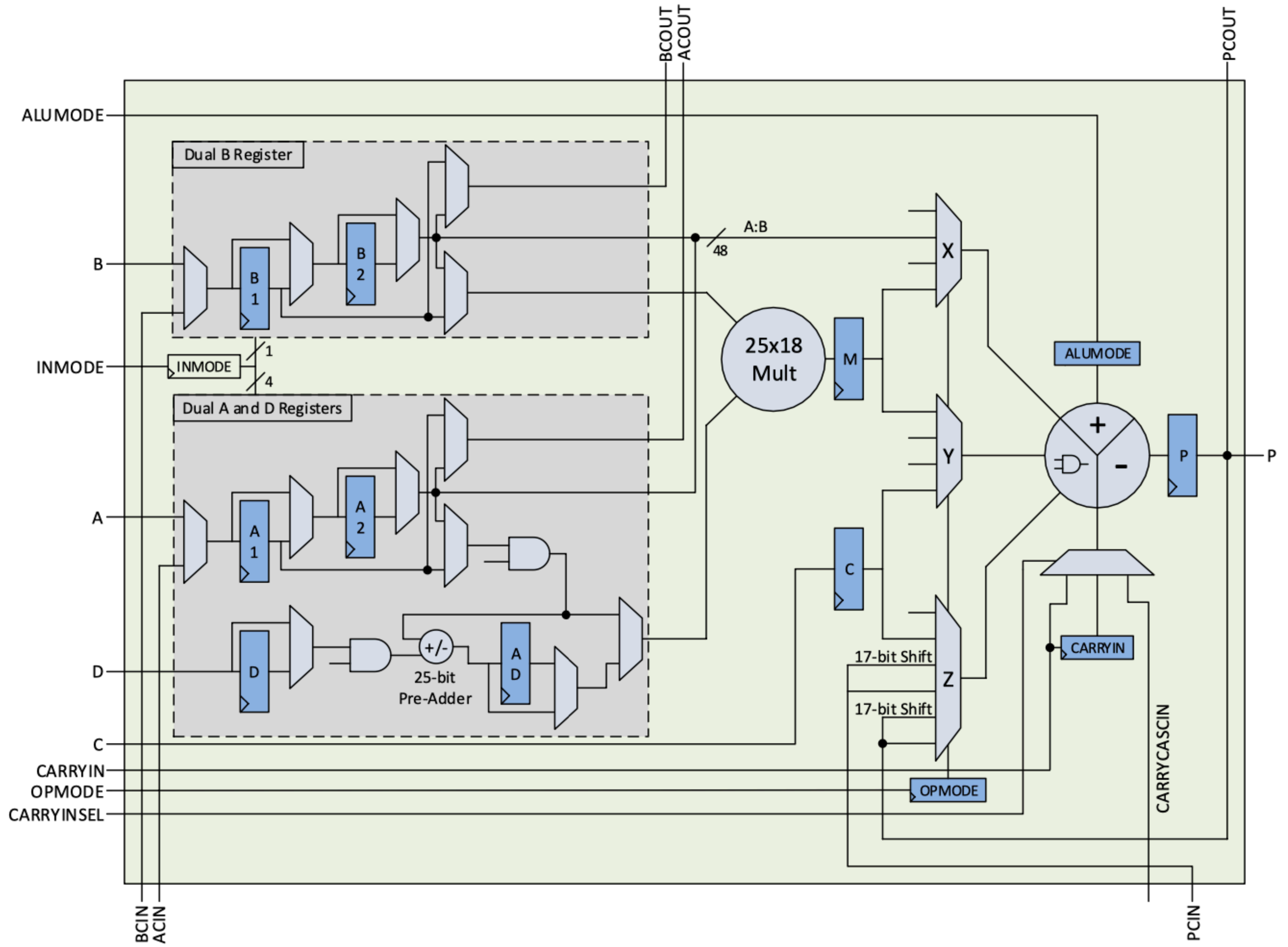
- Virtex UltraScale+ introduced the DSP48E2, multiplier 27×18-bit, added fourth ALU input and inline XOR, 775 MHz
- Versal ACAP introduced the DSP58, multiplier to 27×24-bit, FP32 mode, and complex multiplier support, 1000+ MHz
- Crucially, these primitives are **backward compatible**



*These signals are dedicated routing paths internal to the DSP58 column. They are not accessible through general-purpose routing resources.

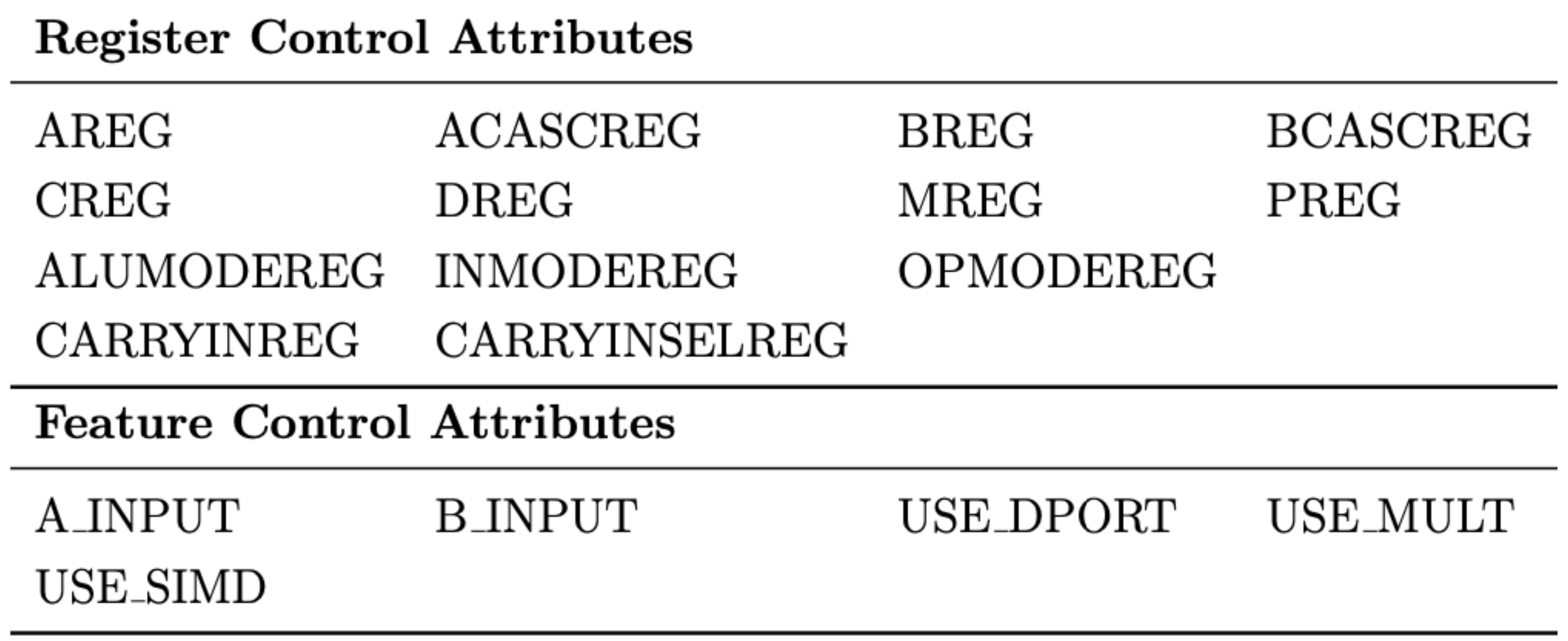
The DSP48E1

- Across whole 7-series families, compatible with the later DSP48E2 and DSP58
- Multiplier with adjustable registered inputs
- ALU for addition/logic/ accumulation
- Optional pre-adder



The DSP48E1

- Various “attributes” are set at **compile** time when instantiating the primitive:
  - Number of register stages to enable
  - Direct or cascaded (from neighbours) inputs
  - Use of D-port
  - Multiplier disabled, enabled, or dynamic
  - ALU in scalar or SIMD mode



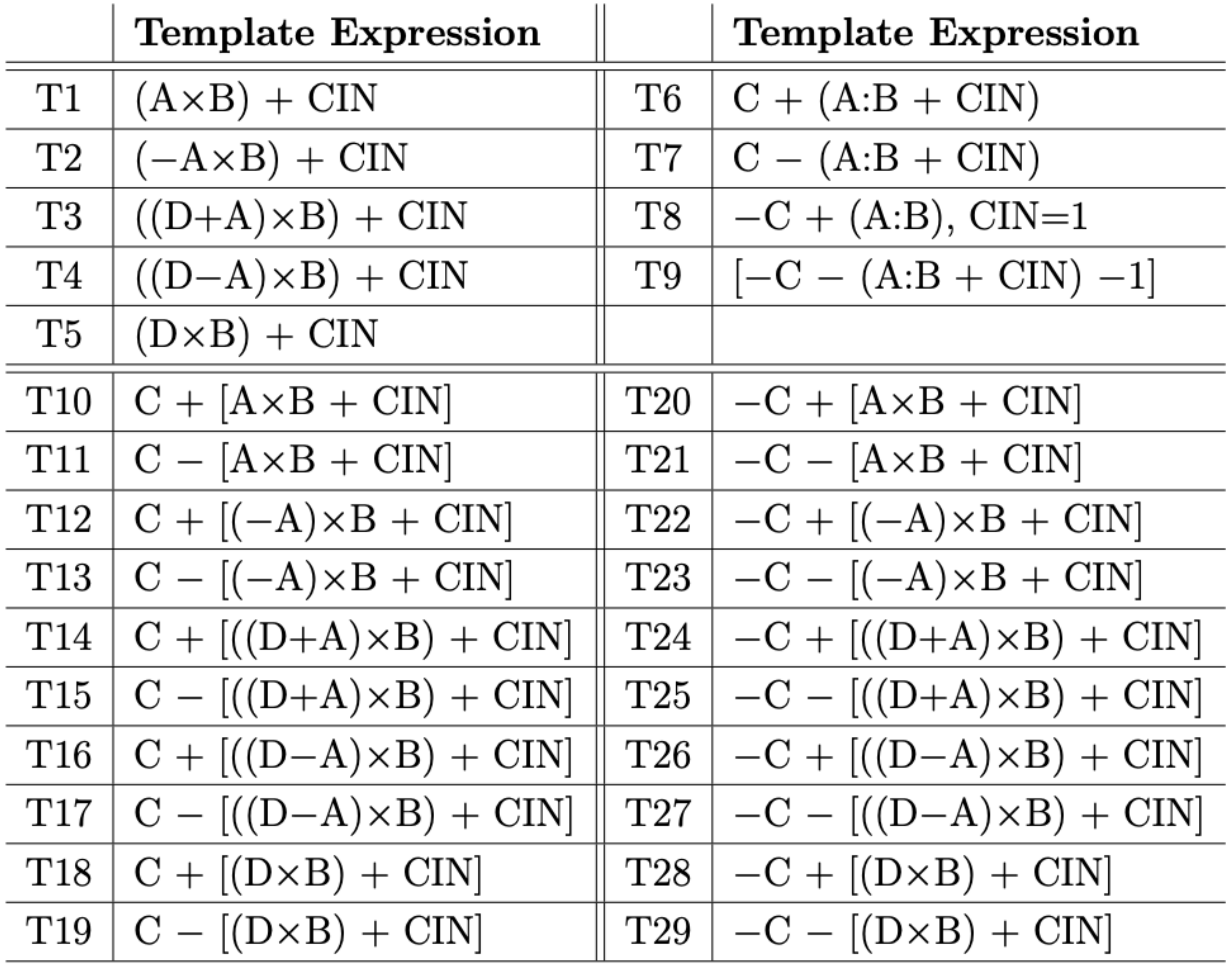
Register Control Attributes			
AREG	ACASCREG	BREG	BCASCREG
CREG	DREG	MREG	PREG
ALUMODEREG	INMODEREG	OPMODEREG	
CARRYINREG	CARRYINSELREG		
Feature Control Attributes			
A_INPUT	B_INPUT	USE_DPORT	USE_MULT
USE_SIMD			

The DSP48E1

- Three **dynamic** control signals allow functionality to change at runtime
  - INMODE: determines the functionality of the input registers and pre-adder
  - OPMODE: controls the X, Y, and Z multiplexers that feed the ALU
  - ALUMODE: controls the function performed in the ALU
- These signals open up a significant opportunity to further exploit DSP blocks

DSP48E1 supported expressions

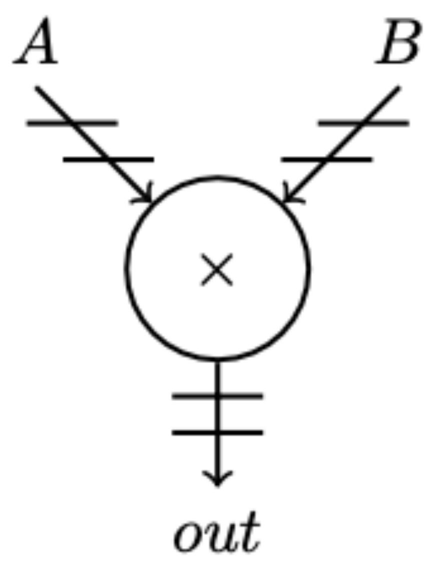
- Combining different attributes and dynamic signals allow the DSP block to support a wide range of different computational expressions *on a per cycle basis*
- Here T1–T5 use only the pre-adder and multiplier, T6–T9 do not use the multiplier, while T10–T29 use all functions



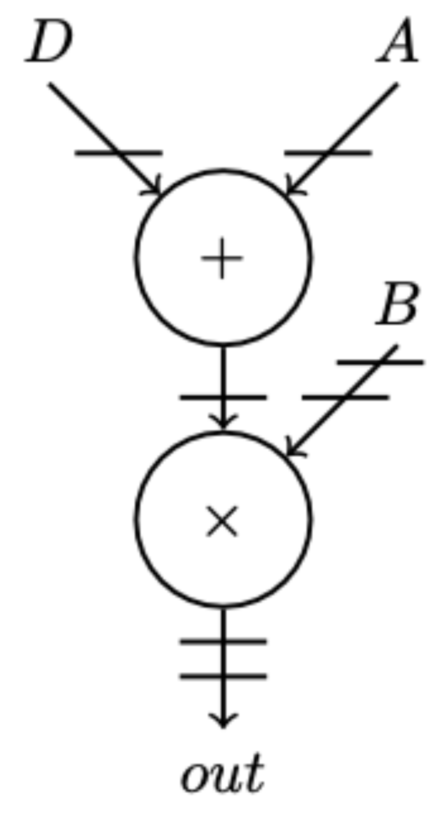
	Template Expression		**Template Expression**
T1	$(A \times B) + CIN$	T6	$C + (A{:}B + CIN)$
T2	$(-A \times B) + CIN$	T7	$C - (A{:}B + CIN)$
T3	$((D+A) \times B) + CIN$	T8	$-C + (A{:}B)$, $CIN=1$
T4	$((D-A) \times B) + CIN$	T9	$[-C - (A{:}B + CIN) -1]$
T5	$(D \times B) + CIN$		
T10	$C + [A \times B + CIN]$	T20	$-C + [A \times B + CIN]$
T11	$C - [A \times B + CIN]$	T21	$-C - [A \times B + CIN]$
T12	$C + [(-A) \times B + CIN]$	T22	$-C + [(-A) \times B + CIN]$
T13	$C - [(-A) \times B + CIN]$	T23	$-C - [(-A) \times B + CIN]$
T14	$C + [((D+A) \times B) + CIN]$	T24	$-C + [((D+A) \times B) + CIN]$
T15	$C - [((D+A) \times B) + CIN]$	T25	$-C - [((D+A) \times B) + CIN]$
T16	$C + [((D-A) \times B) + CIN]$	T26	$-C + [((D-A) \times B) + CIN]$
T17	$C - [((D-A) \times B) + CIN]$	T27	$-C - [((D-A) \times B) + CIN]$
T18	$C + [(D \times B) + CIN]$	T28	$-C + [(D \times B) + CIN]$
T19	$C - [(D \times B) + CIN]$	T29	$-C - [(D \times B) + CIN]$

DSP48E1 supported expressions

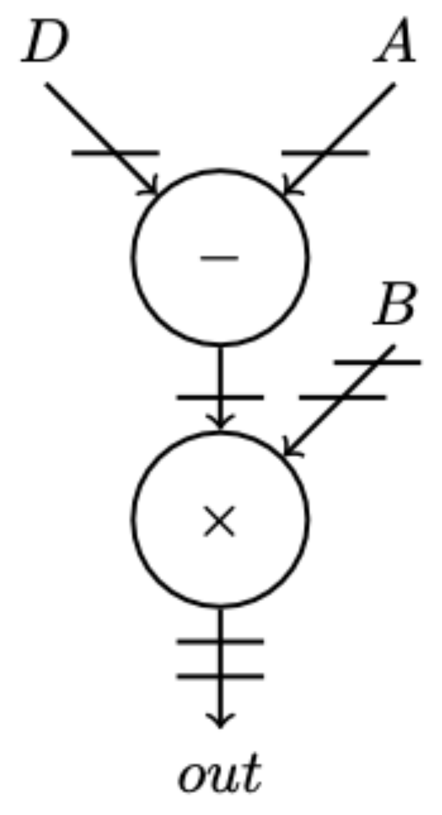
- These supported expressions can be expressed as graphs which can be matched to during compilation
- It is important to note that the register positions in any code must match those of the DSP block to be able to map successfully
- This presents a challenge to the mapping tool with explicitly scheduled designs



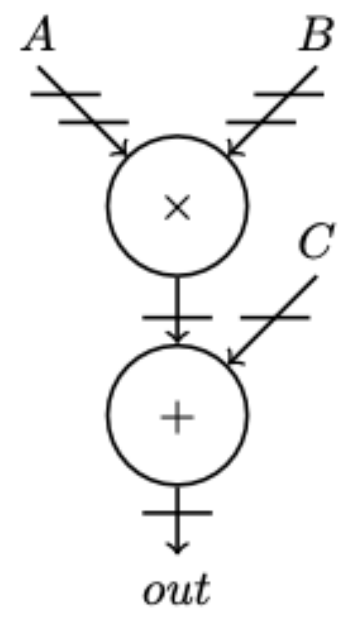
(a)



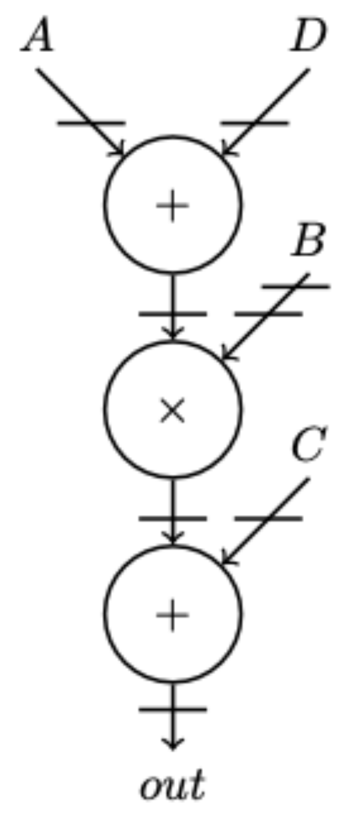
(b)



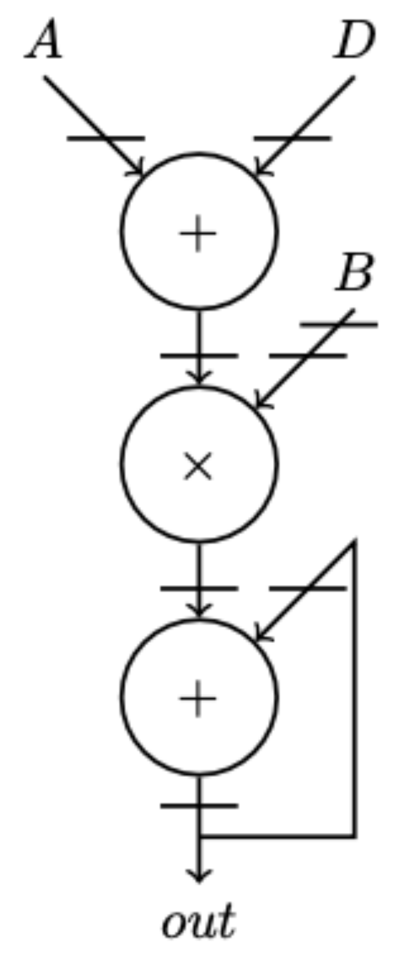
(c)



(d)



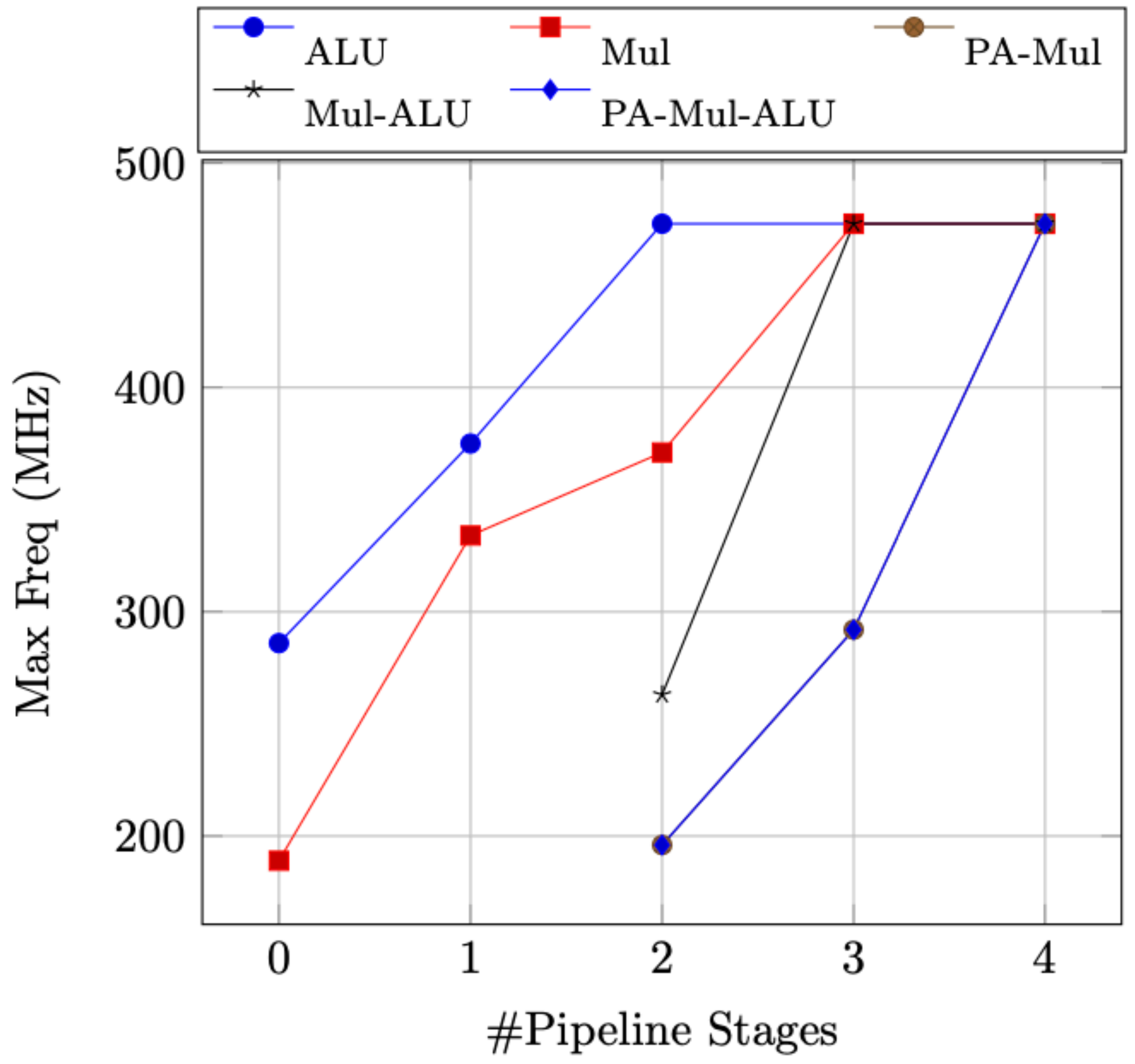
(e)



(f)

Maximising frequency

- Depending on the features used, the number of pipeline stages required to achieve maximum frequency changes
- (Note these experiments were on V6, and frequencies are higher on VU+/Versal)
- Enabling all features to use with dynamic programmability requires a pipeline depth of 4 cycles

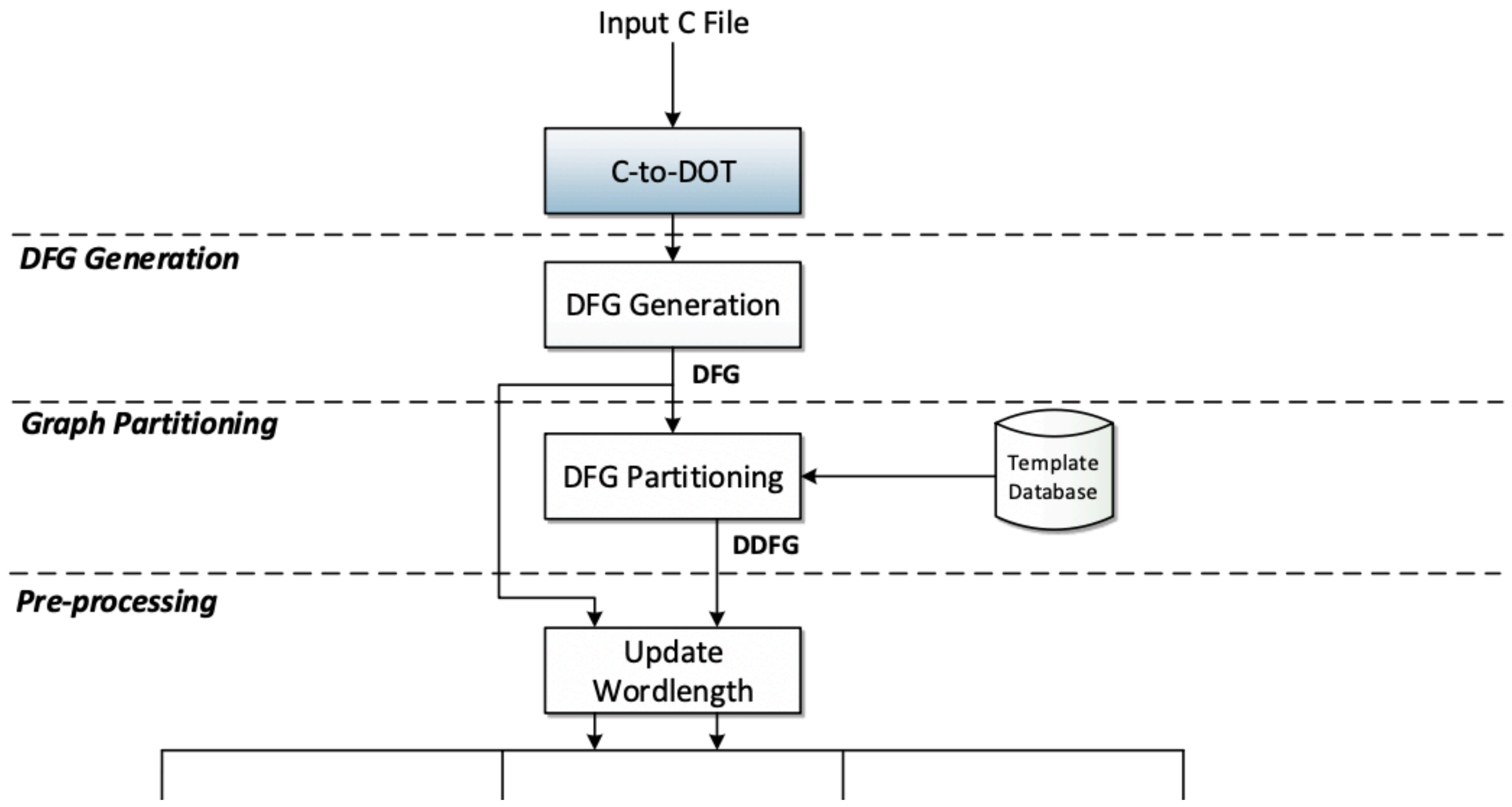


Mapping DFGs to DSP blocks

- We found that synthesis tools tend to mostly only utilise the multipliers in the DSP blocks when inferring, often additional operations are not absorbed into the DSP block
- Frequency of inferred circuits well below the capabilities of the DSP blocks due to mismatched pipelining
- *What if we feed architectural information up the design flow?*

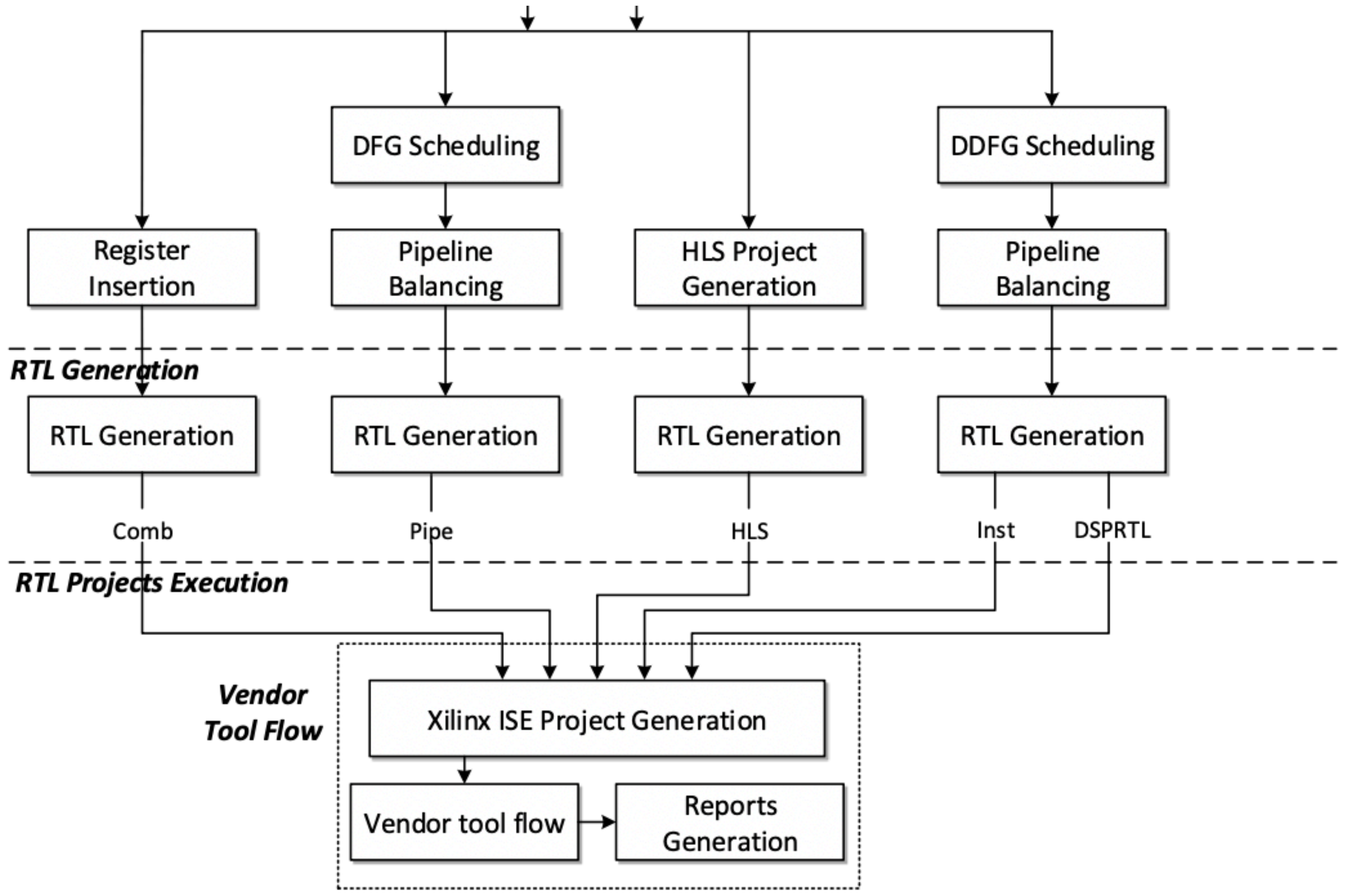
Mapping DFGs to DSP blocks

- Our flow reads a C representation of an arithmetic expression (mult, add, sub) and generates a DFG
- We analyse this DFG, matching against the templates the DSP block supports
- We update intermediate wires to the correct wordlengths to allow inference



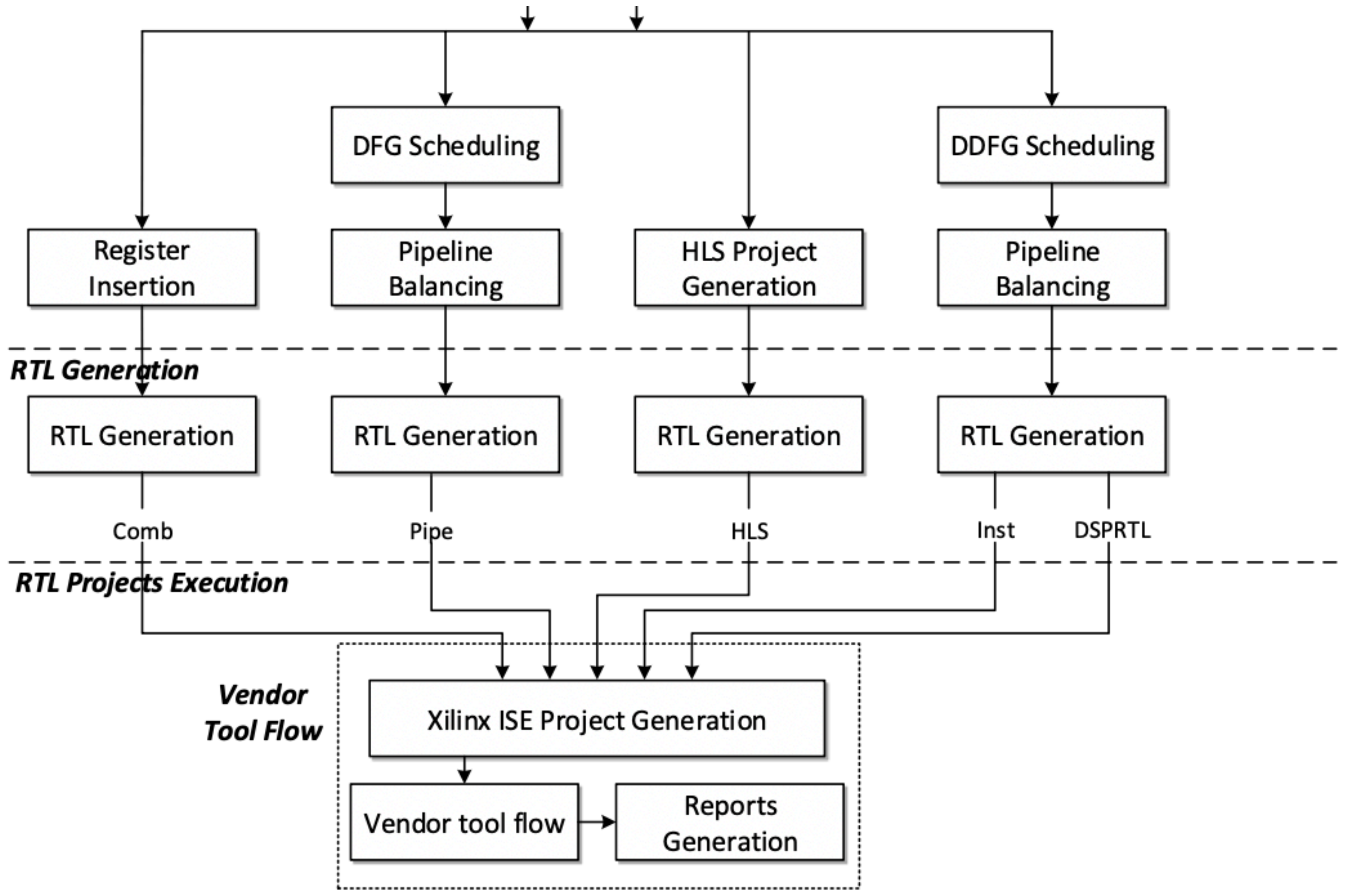
Mapping DFGs to DSP blocks

- Two traditional designs are generated
  - Combinational implementation with ample output registers for retiming by the tool
  - Pipelined implementation following traditional approaches (with suitable balancing of branches)

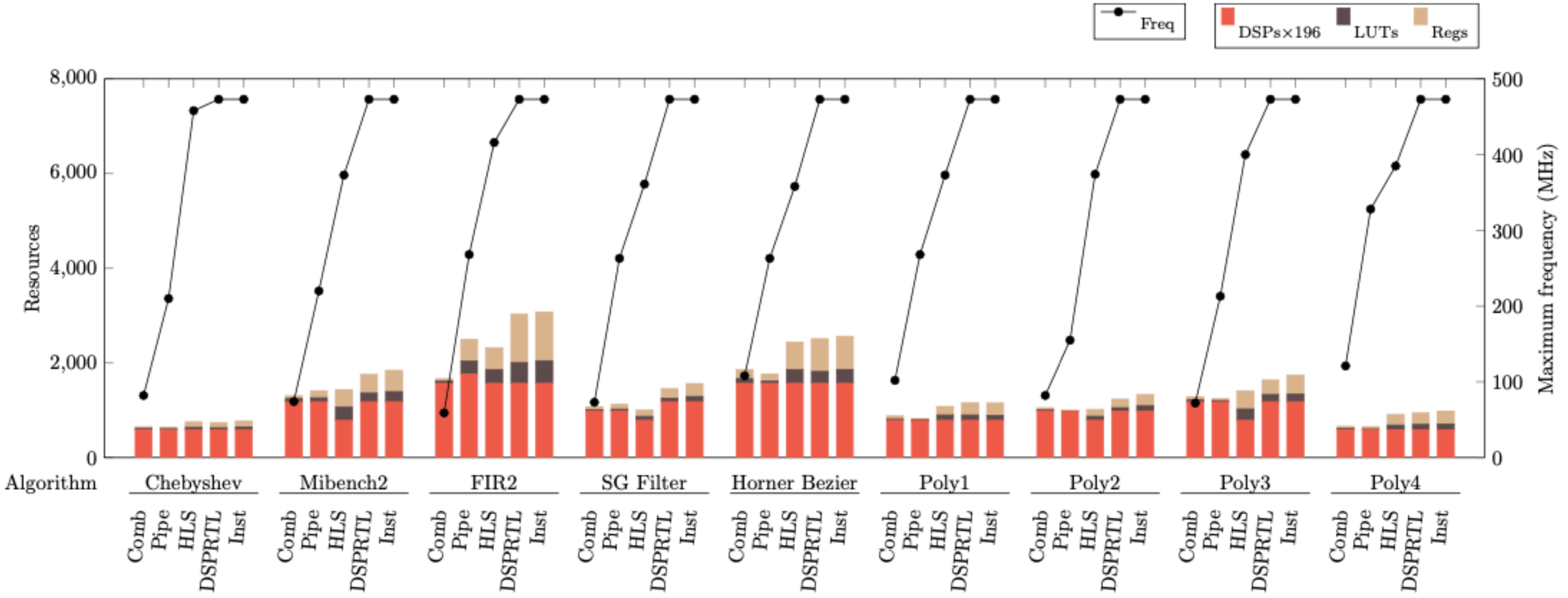


Mapping DFGs to DSP blocks

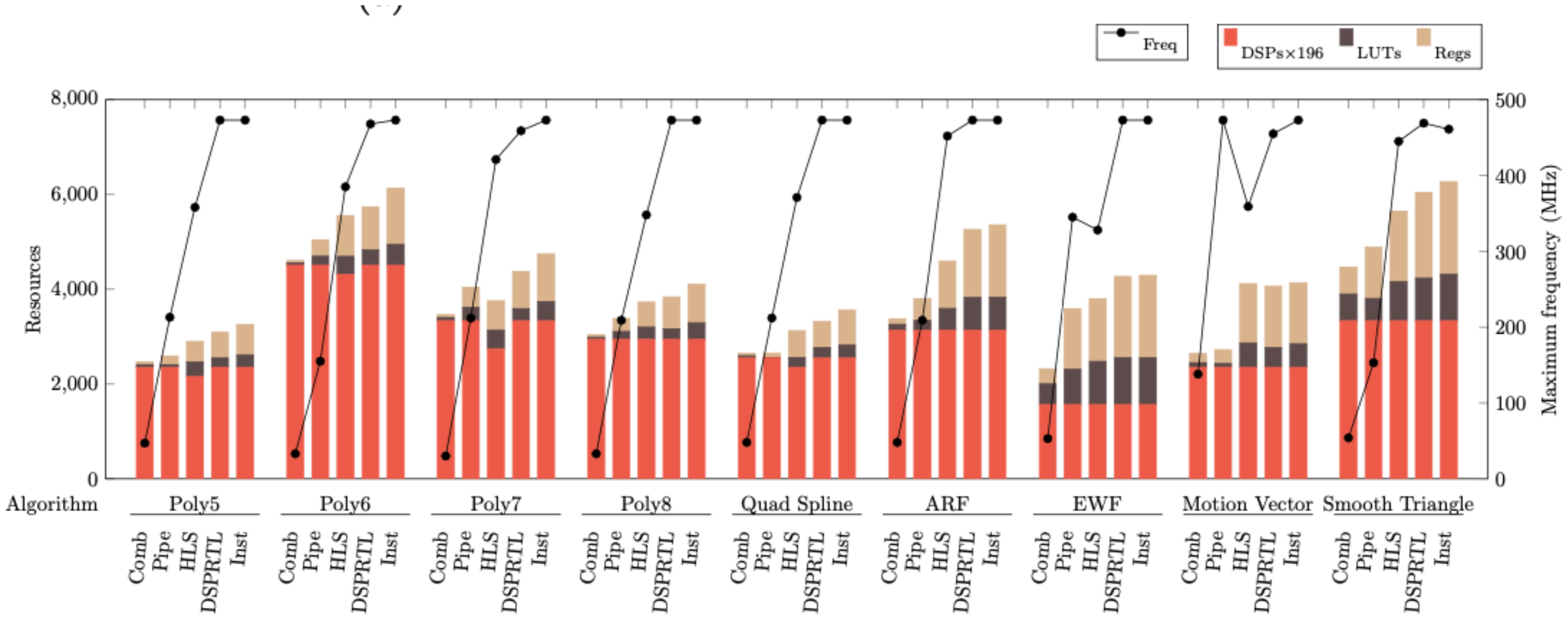
- An HLS implementation with suitable directives is generated
- Our approach uses the matched templates to generate both a low-level instantiated graph of DSP blocks (Inst) as well as an RTL implementation suitably pipelined (DSPRTL)



Mapping results

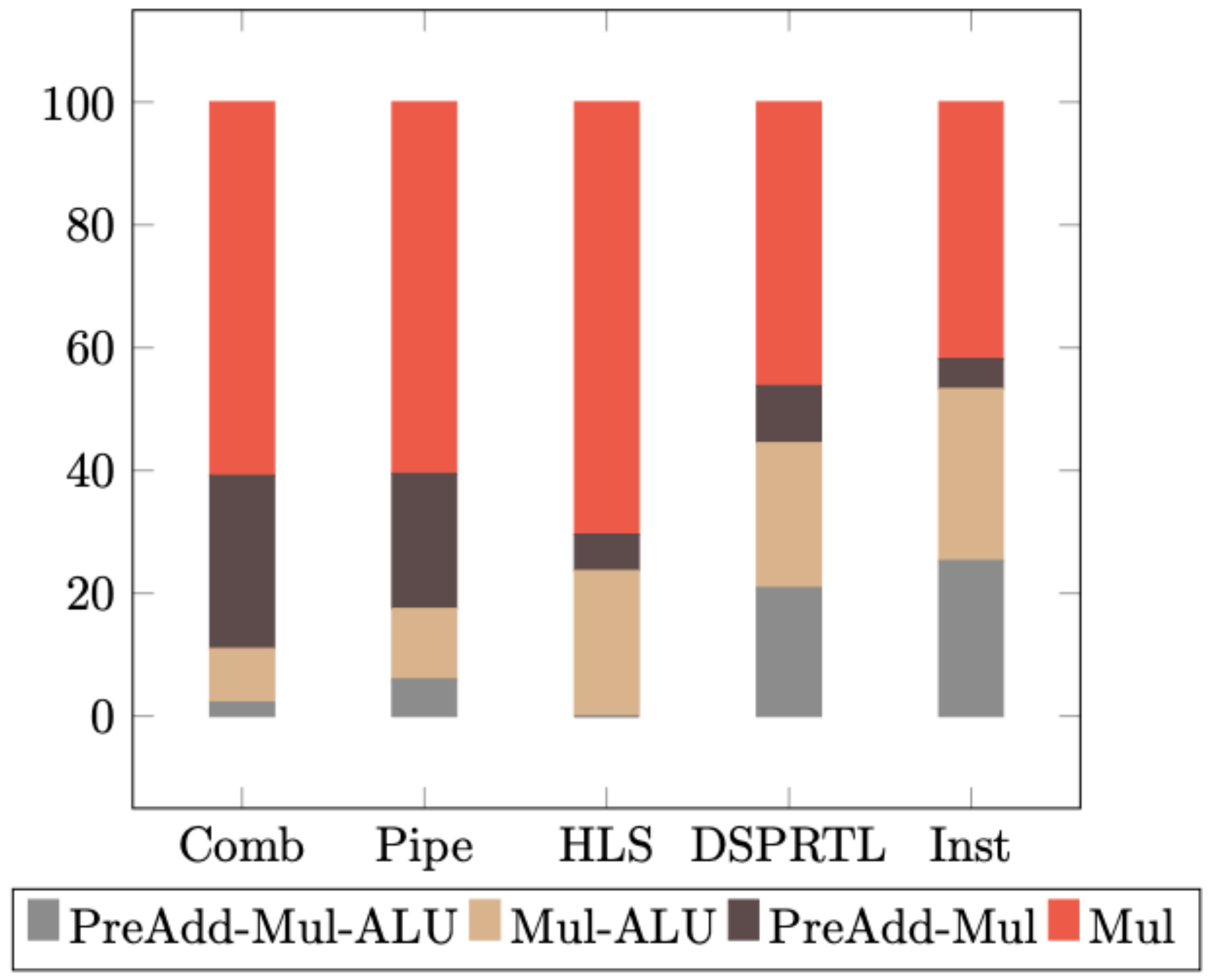


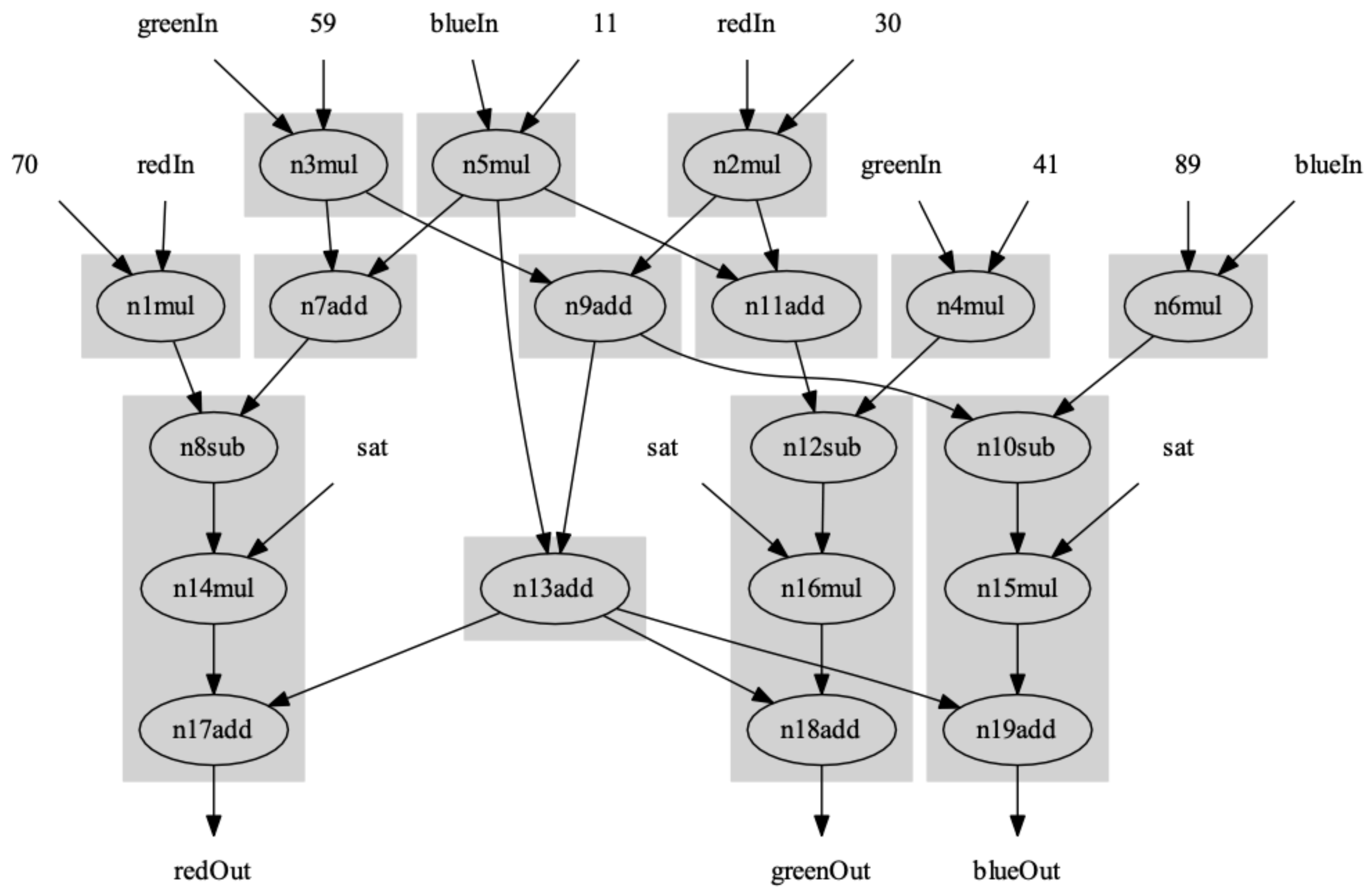
Mapping results



Key takeaways

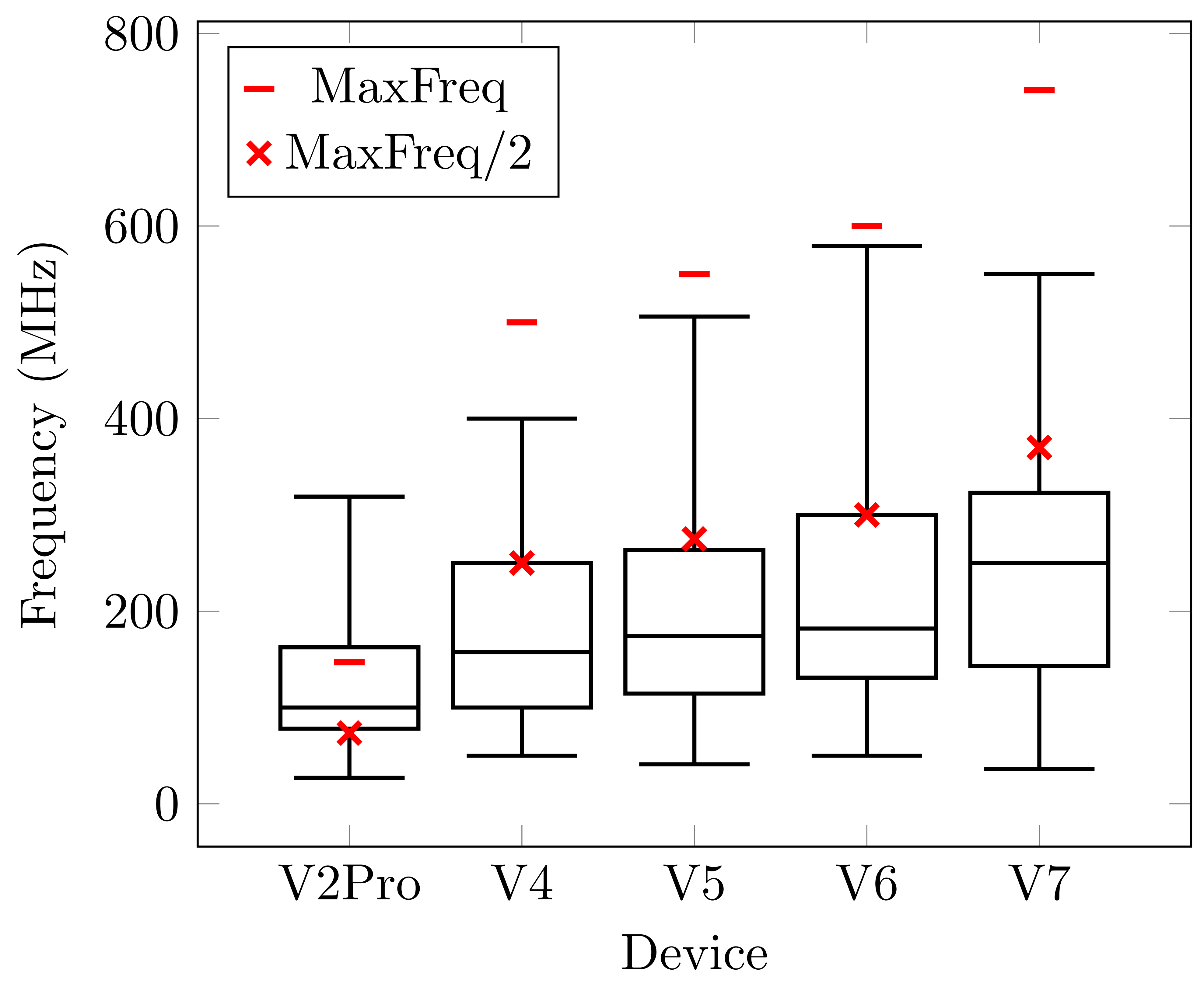
- Consideration of DSP block structure and pipelining is essential in fully exploiting them at maximum performance
- It **is** possible to achieve the highest performance from portable RTL descriptions, as long as the DSP block structure has been used to guide pipelining
- Overall latency (and FF usage) may increase due to the enforced 4-cycle DSP block pipeline





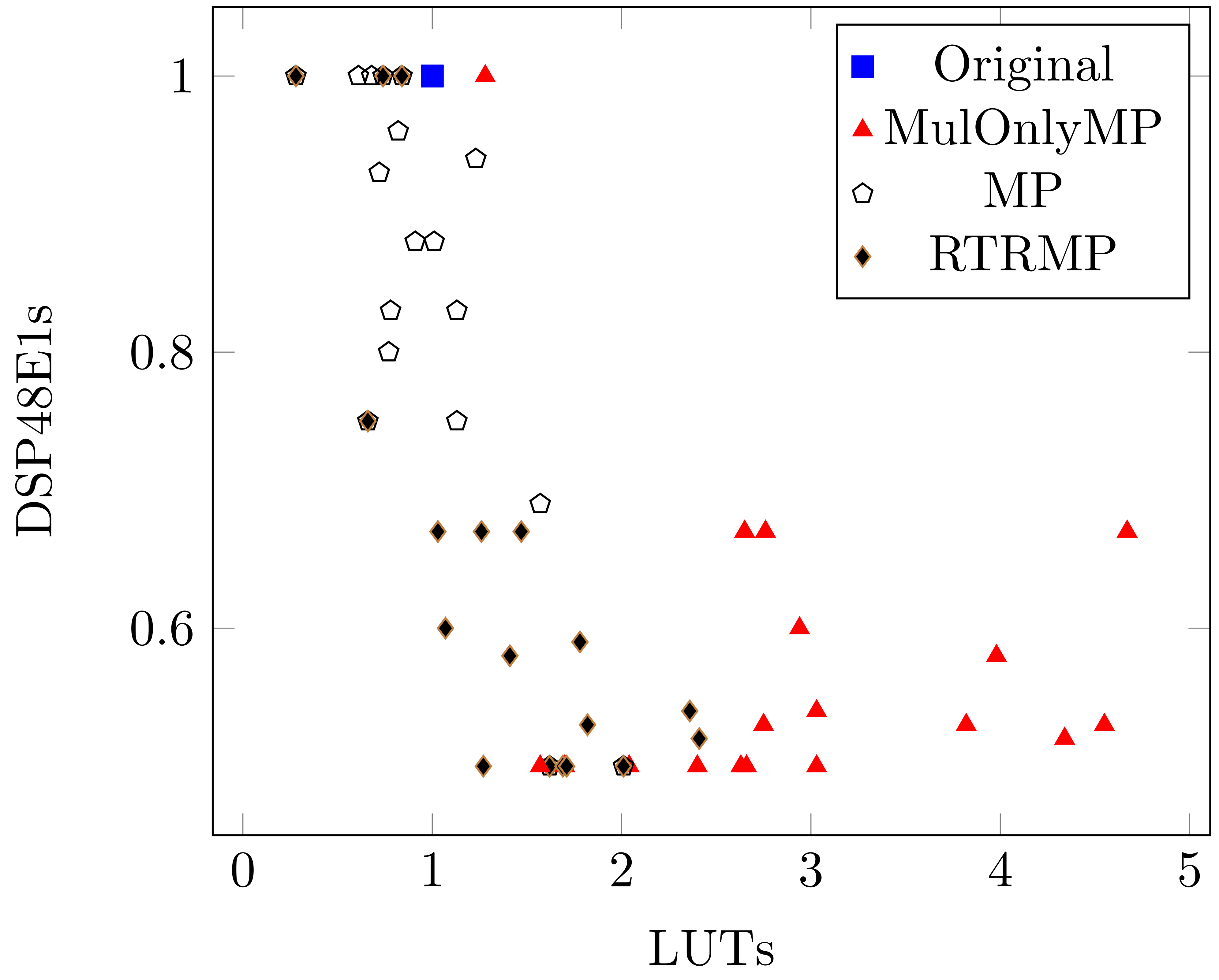
DSP blocks in larger designs

- DSP block frequencies have scaled significantly across FPGA generations
- Meanwhile, general design frequency has not scaled as well
- A 300MHz design is still considered "fast" though DSP blocks on current devices support three times that frequency
- We proposed a flow that multi-pumps DSP blocks to reduce usage in large designs



DSP blocks in larger designs

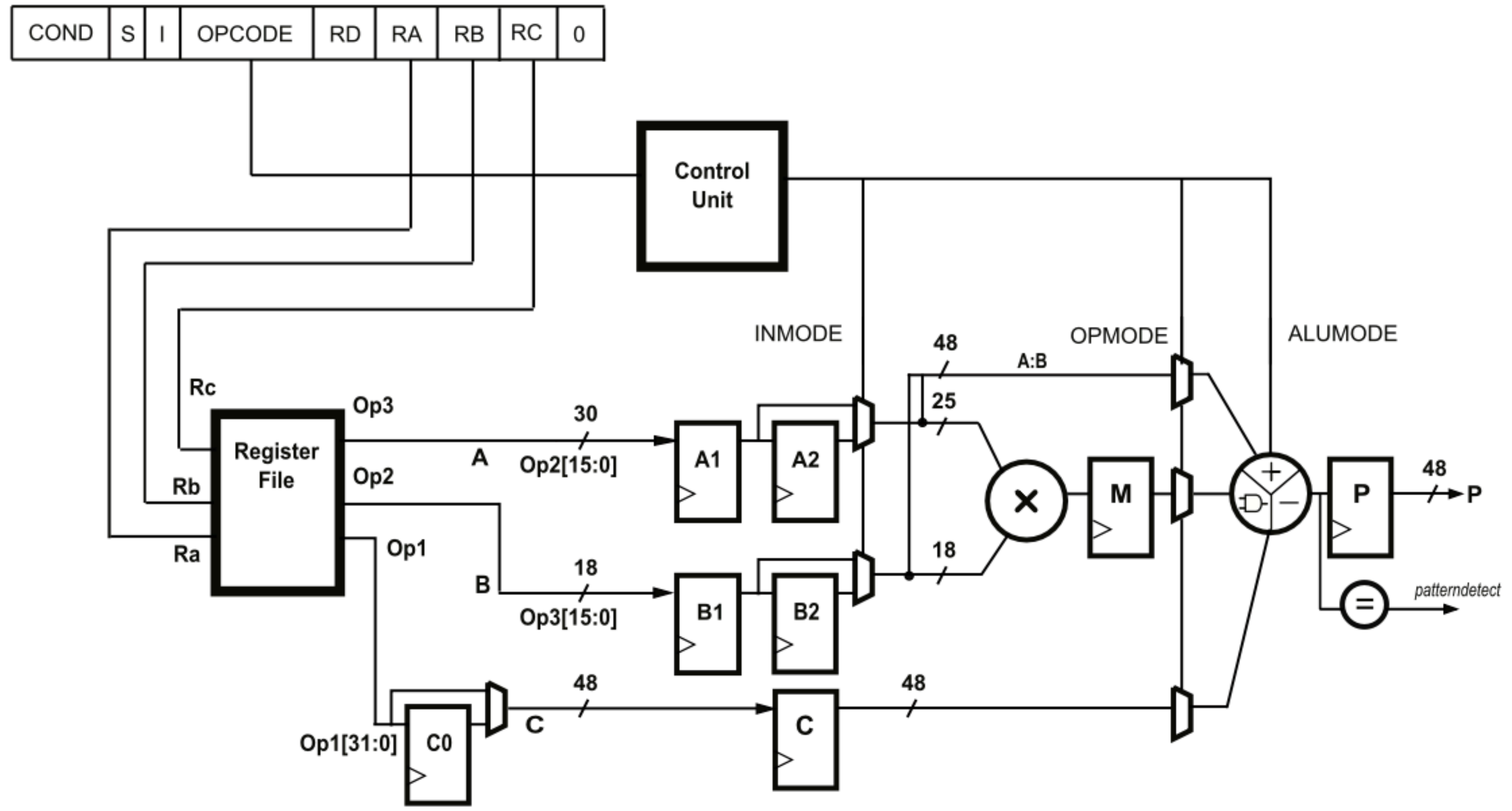
- Multi-pumping runs a resource at a multiple of the clock frequency of surrounding logic
- Multi-pumping multipliers can be beneficial but uses more LUTs *(MulOnlyMP)*
- Using more DSP block features reduces the chances for resource sharing *(MP)*
- Using dynamic programmability expands this further and achieves significant improvements *(RTRMP)*



Building a soft processor

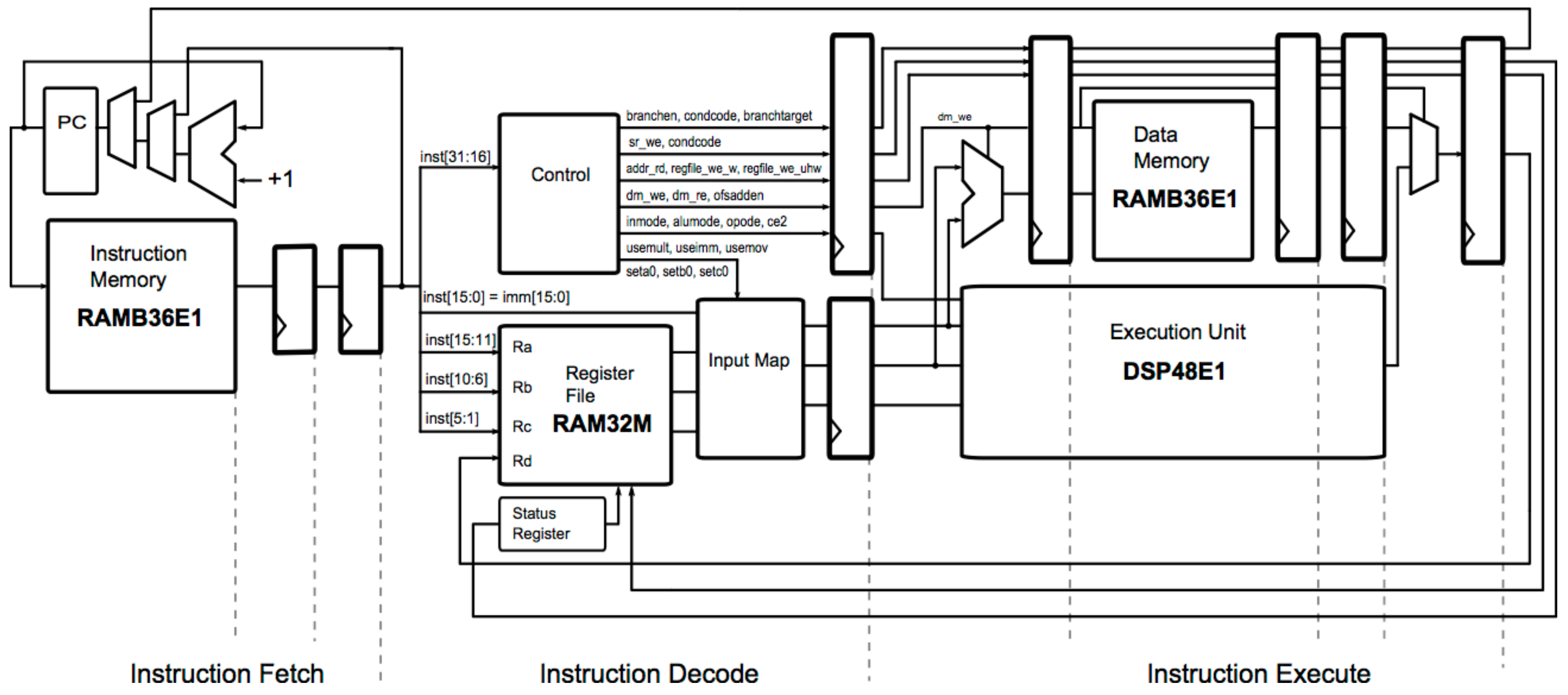
- Recognising that the DSP block can implement all the functionality of a processor ALU, we set about building a soft processor that uses it

- Control unit translates instructions into INMODE, OPMODE, ALUMODE control signals



Building a soft processor

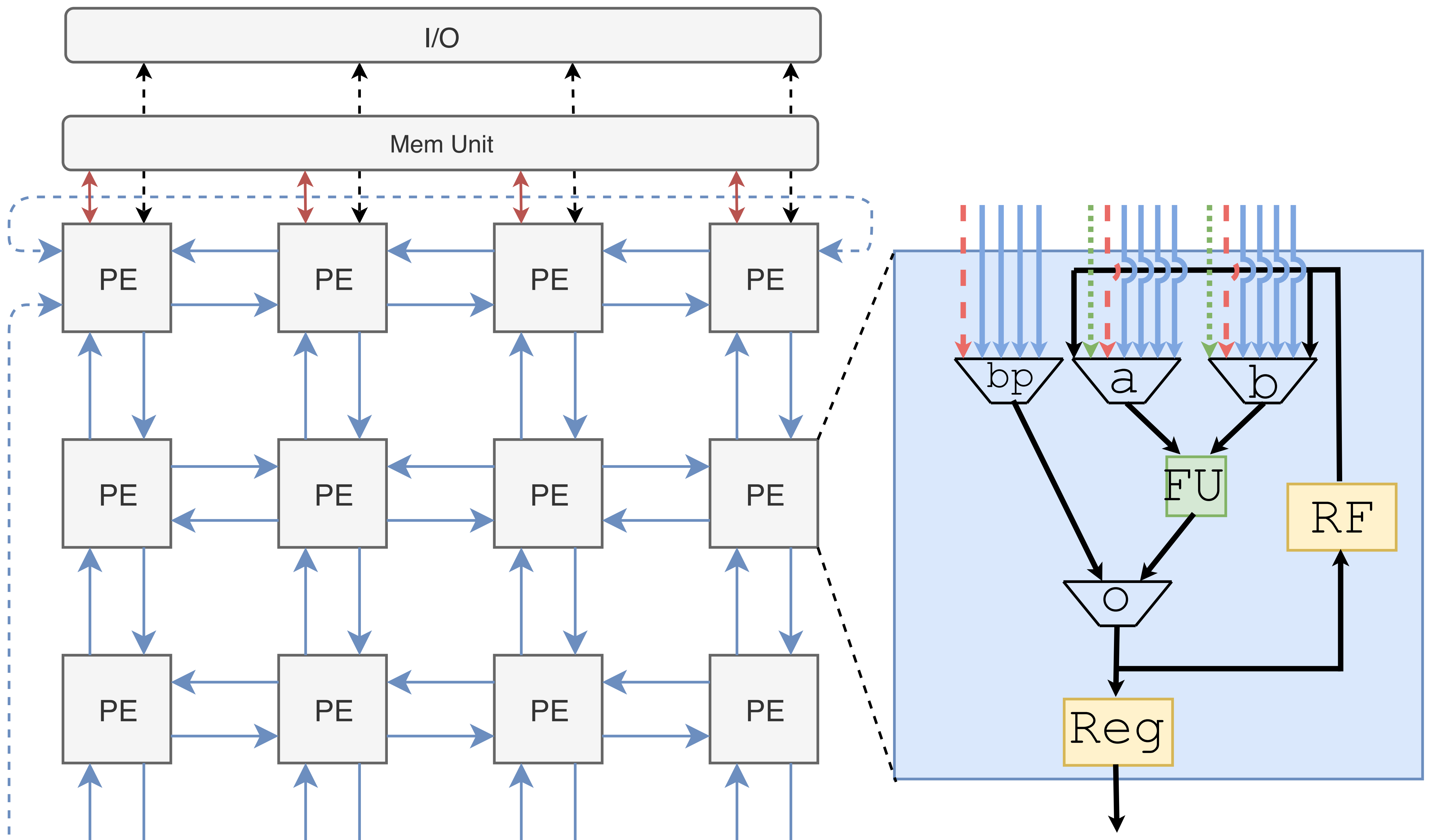
- iDEA FPGA soft processor: over 400MHz on Xilinx Virtex 6 and all 7-series devices, half the size of other soft cores and double frequency



Stages	Freq. (MHz)	Registers	LUTs	DSP48E1
iDEA				
5	193	273	255	1
6	257	256	254	1
7	266	308	263	1
8	311	319	293	1
9	405	413	321	1
9 *LUTs*	173	571	864	1
MicroBlaze				
3	189	276	630	3
5	211	518	897	3

Coarse grained reconfigurable arrays

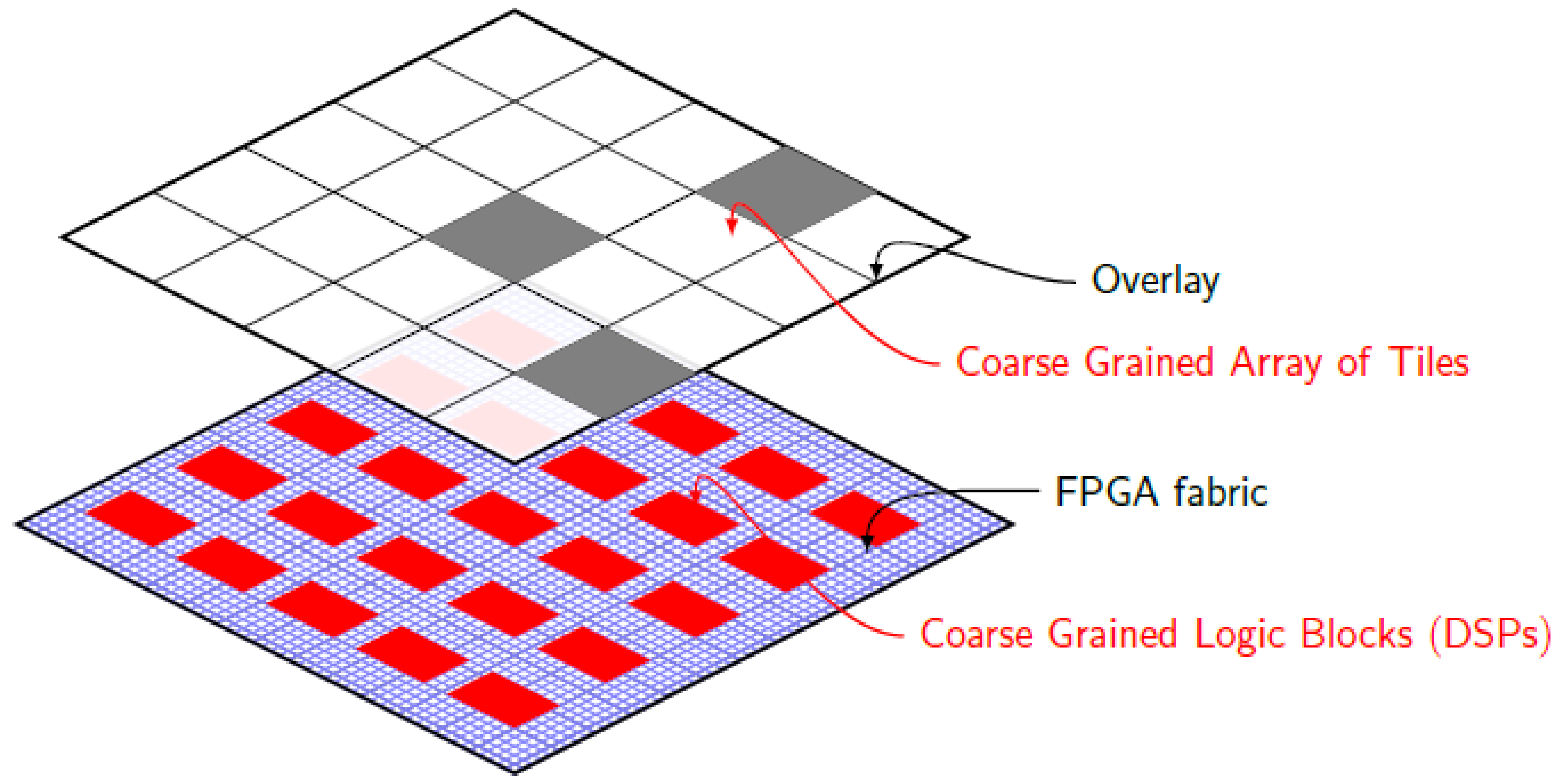
- Arrays of processing elements (PEs) interconnected by a programmable interconnect
- Coarser grained than FPGAs, leading to simpler compilation and configuration
- Dataflow arrangement allows PEs to pass data between them for enhanced parallelism



Source: I Taras and J Anderson, FCCM 2019

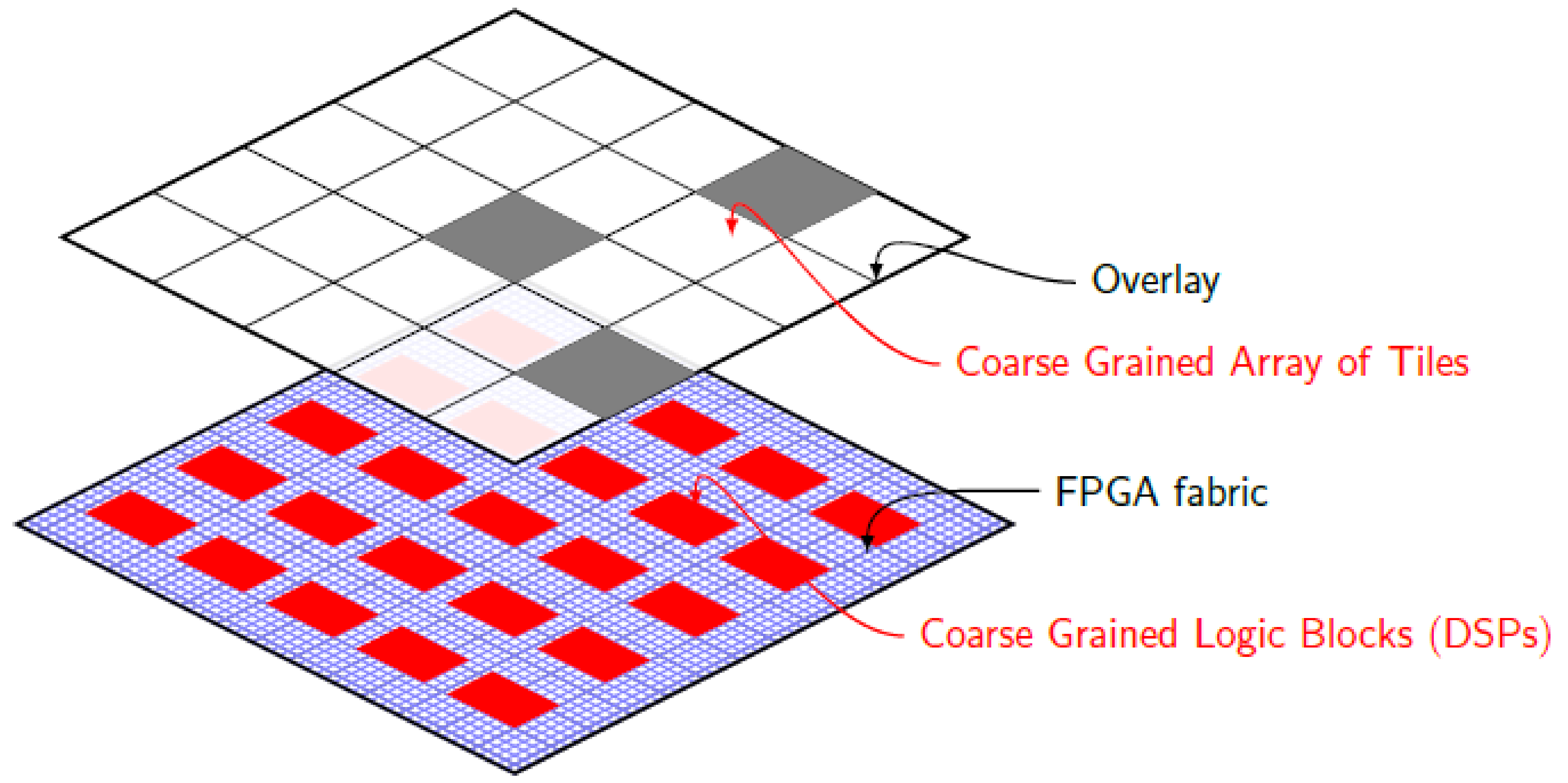
FPGA overlays

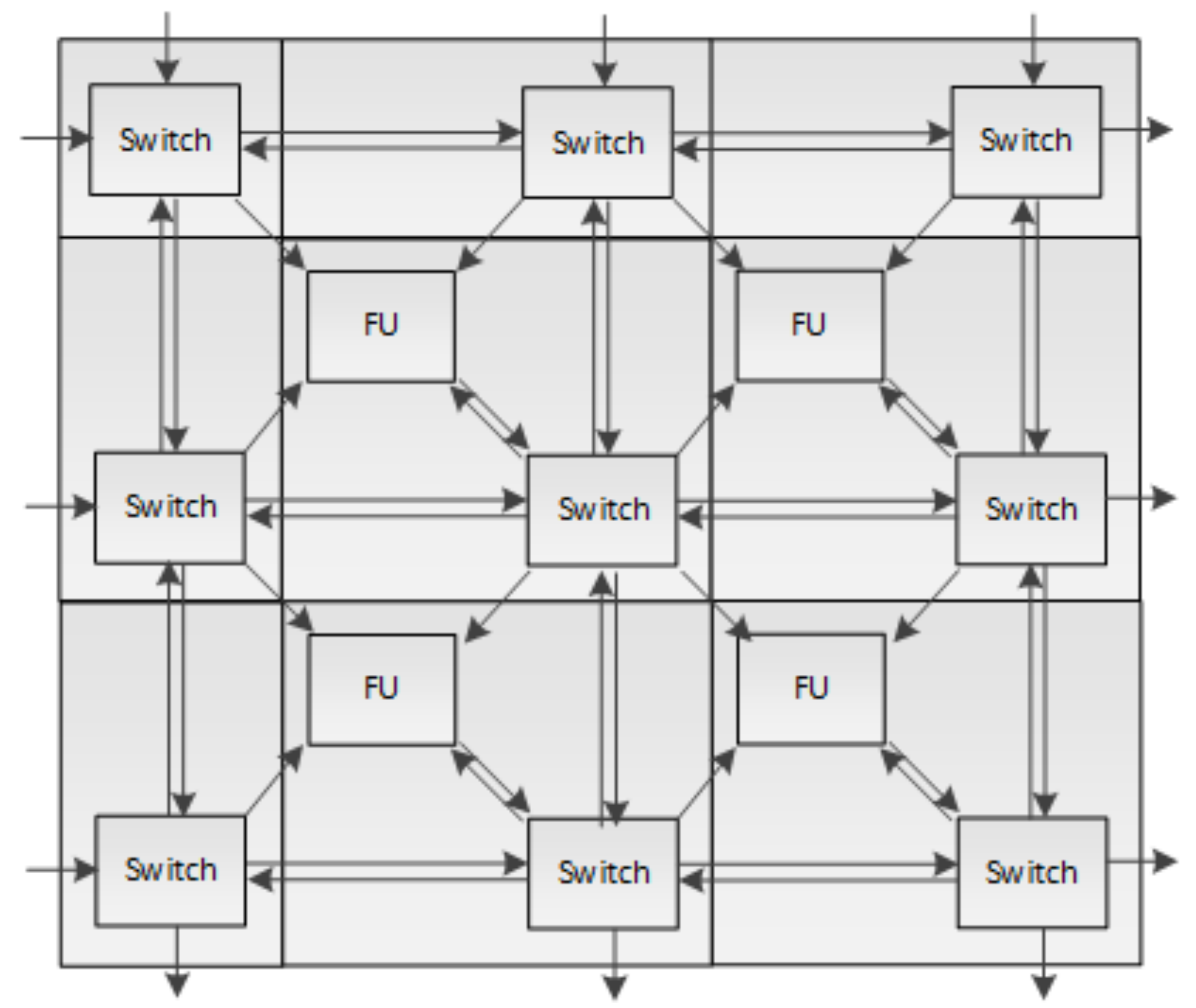
- Rather than target LUT-based FPGA architecture, build a coarse grained architecture on top and target that
- Design flow now deals with larger word-level operations, so simpler and faster
- Circumvent the (slow, cumbersome) backend flow, except to build the overlay (once)
- Retain flexibility over an ASIC CGRAs through having different, reconfigurable overlays


Overlay
Coarse Grained Array of Tiles
FPGA fabric
Coarse Grained Logic Blocks (DSPs)

FPGA overlays

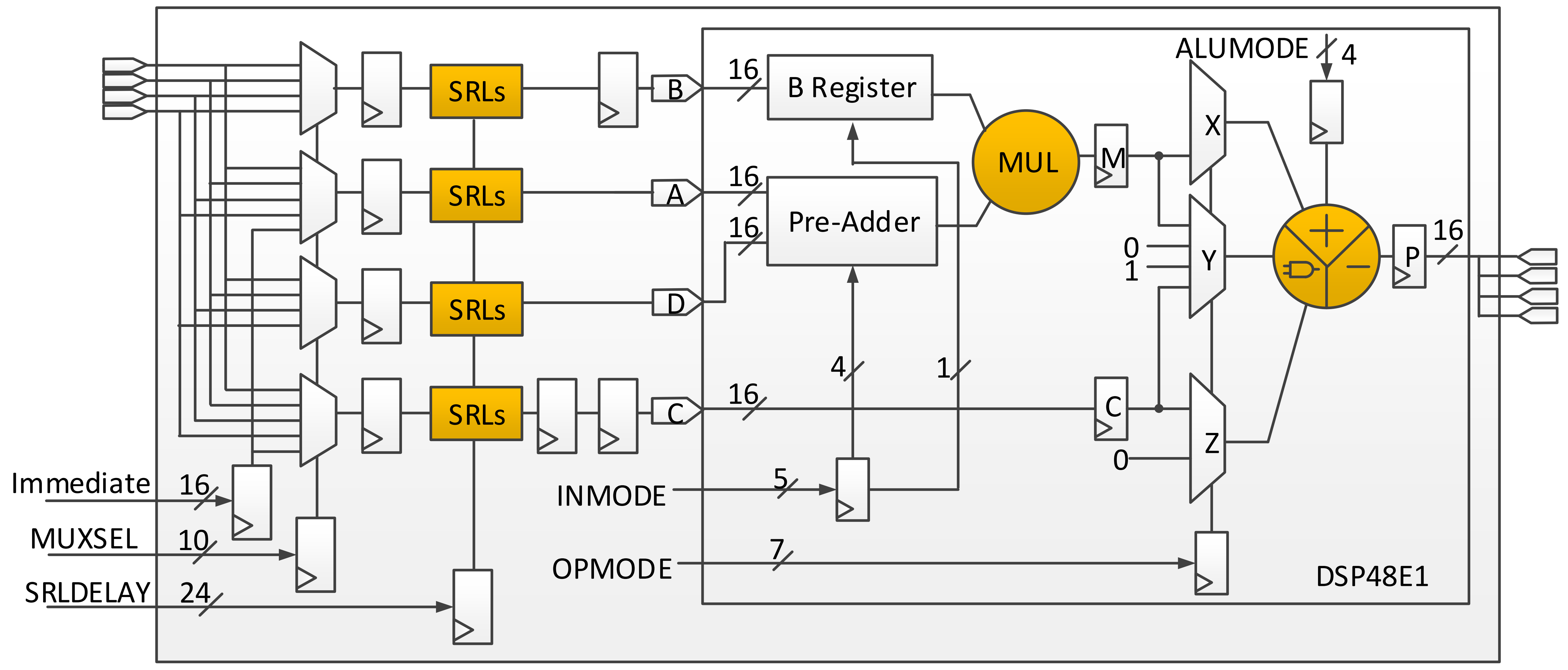
- Rather than target LUT-based FPGA architecture, build a coarse grained architecture on top and target that
- Design flow now deals with larger word-level operations, so simpler and faster
- Circumvent the (slow, cumbersome) backend flow, except to build the overlay (once)
- Retain flexibility over an ASIC CGRAs through having different, reconfigurable overlays


Overlay
Coarse Grained Array of Tiles
FPGA fabric
Coarse Grained Logic Blocks (DSPs)



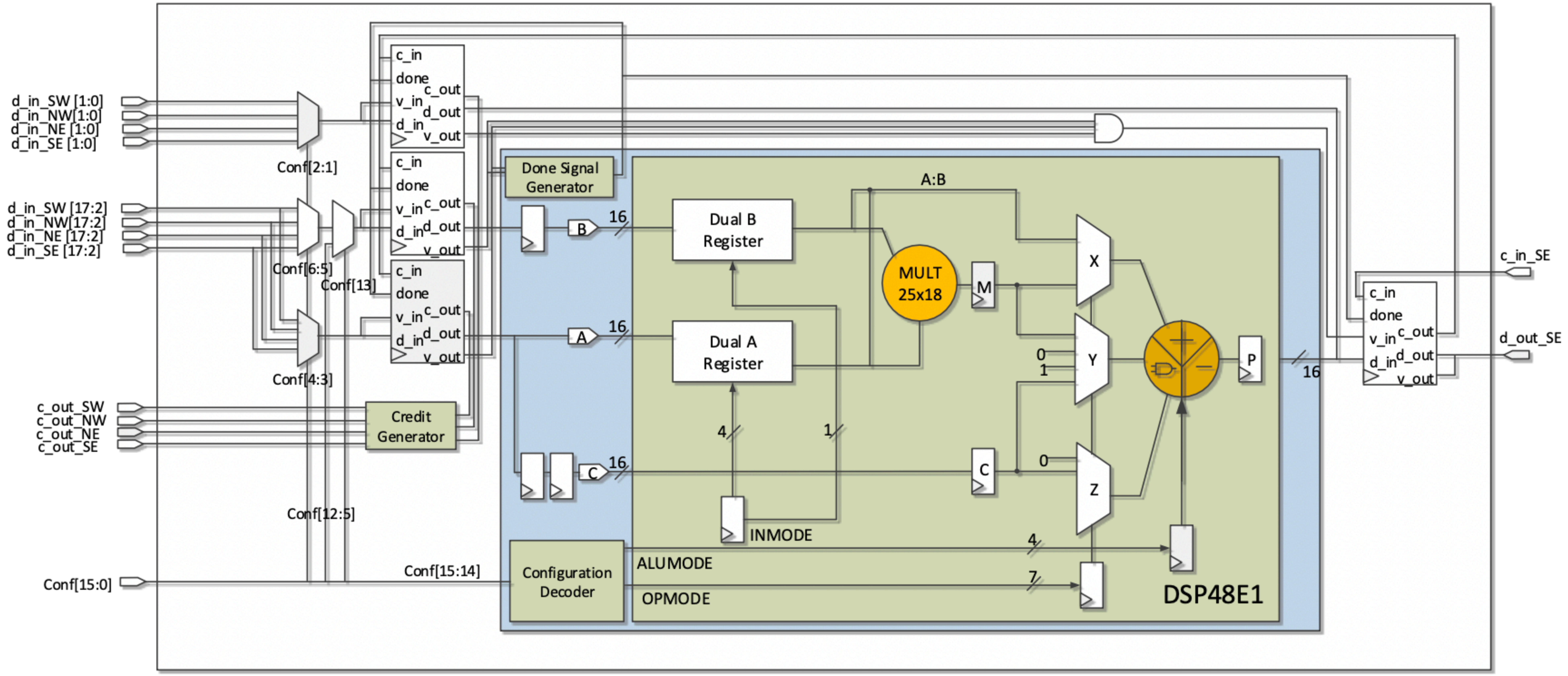
Overlays on FPGAs

- If we build an overlay without considering the FPGA, we are unlikely to reach reasonably frequency or efficient area utilisation
- Consider the DySER CGRA mapped to an FPGA: replace the functional unit with one built around the DSP block
- 25% area reduction and 150% throughput increase compared to original design
- Hence: need architectural optimisation!



Overlays on FPGAs

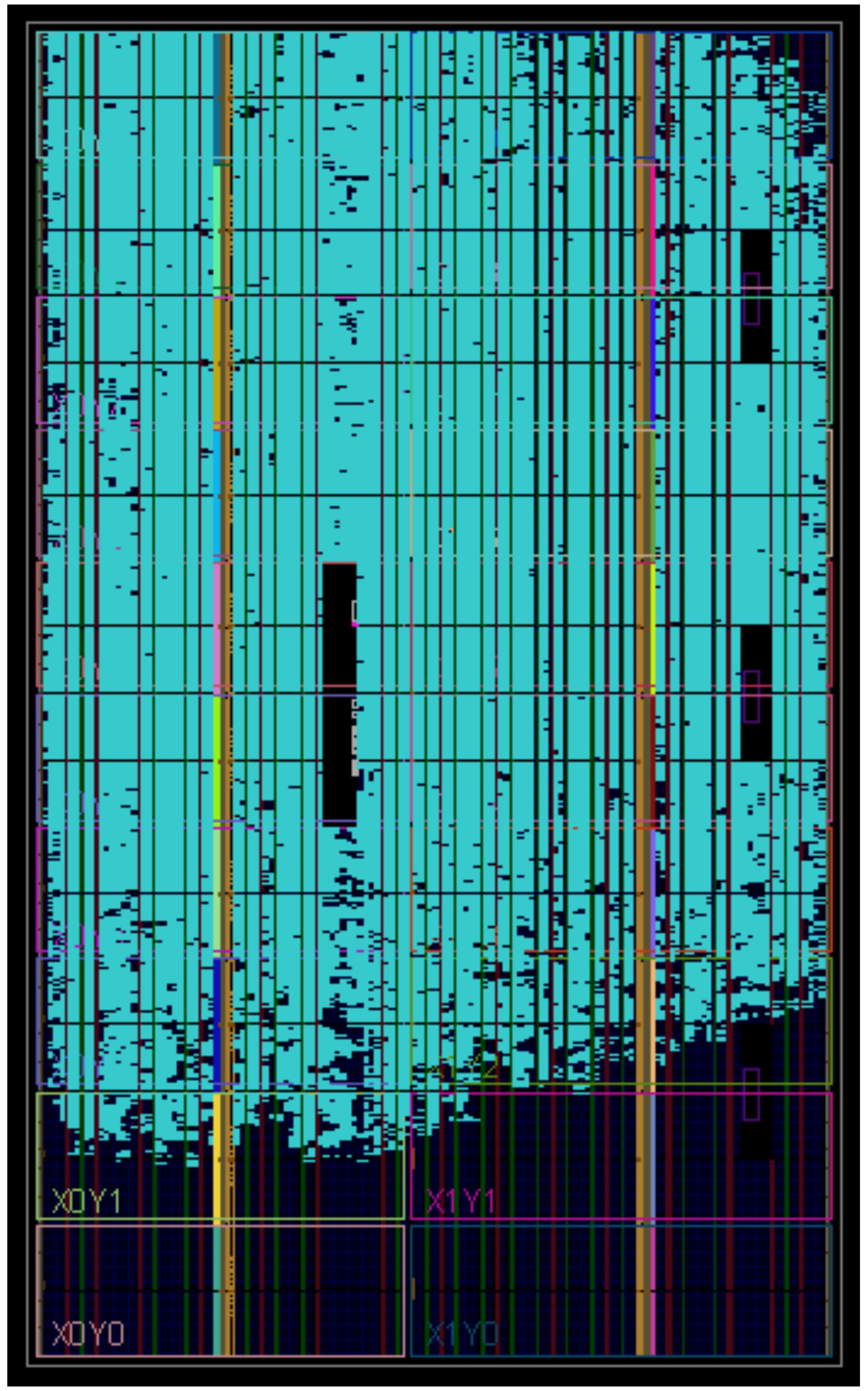
- Some effort required to address the non-commutativity of inputs and to increase pipeline depth



Scaling overlay designs

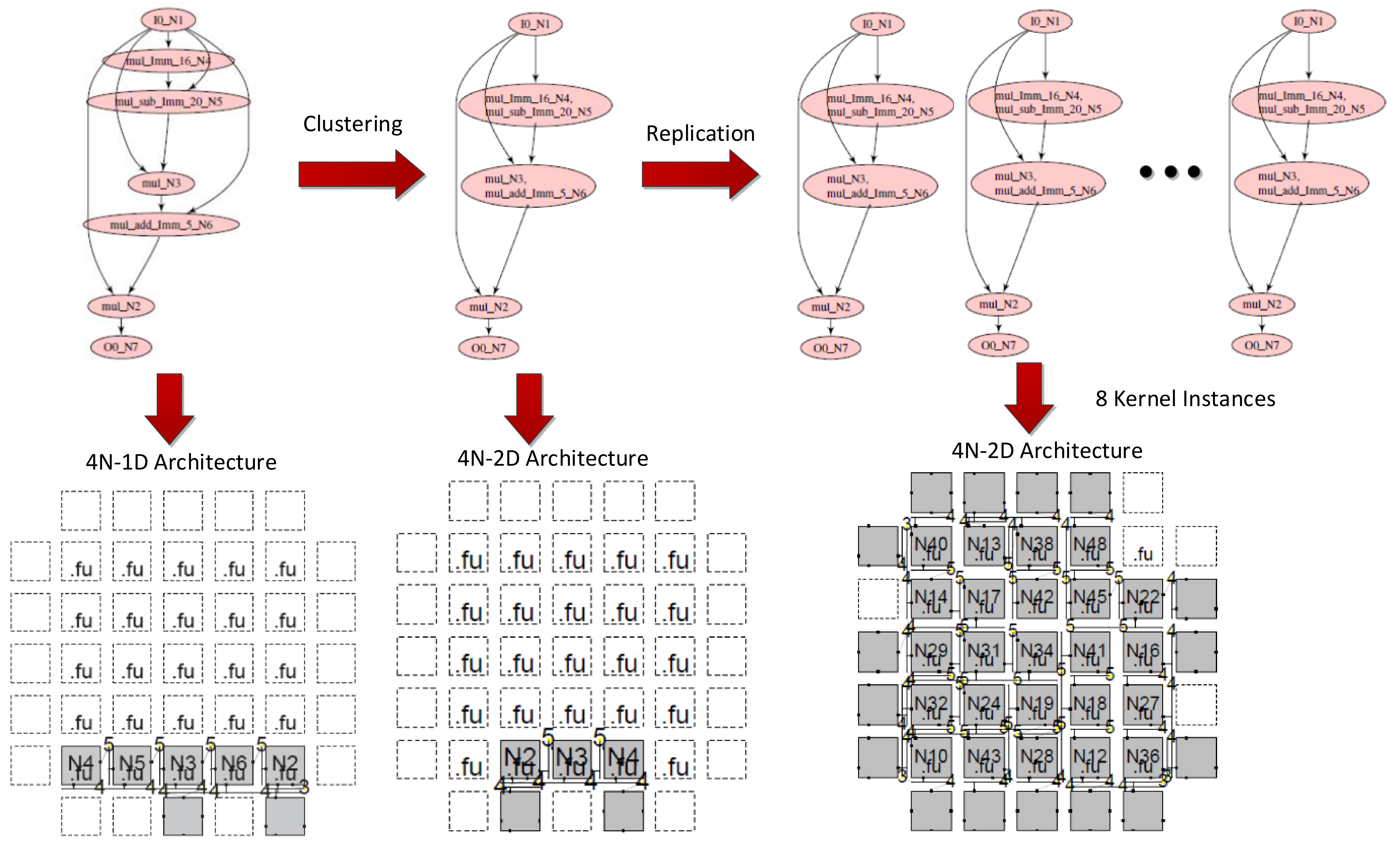
- Clustering more functionality in FUs reduces routing overhead
- 20×20 Overlay (800 DSP Blocks) mapped on Virtex-7 (XC7VX690T) at 380 MHz
- Can support up-to 2400 operation nodes, peak throughput of 912 GOPS
- Scales with newer architectures

FPGA	Size	DSPs	Freq.	Peak GOPS
Zynq 7Z020	8×8	128	300	115
Virtex 7VX690T	20×20	800	380	912
Alveo XCU280 (3 SLR)	40×40	3200	500	4800
Alveo XCU280 (1 SLR)	22×22	968	650	1887



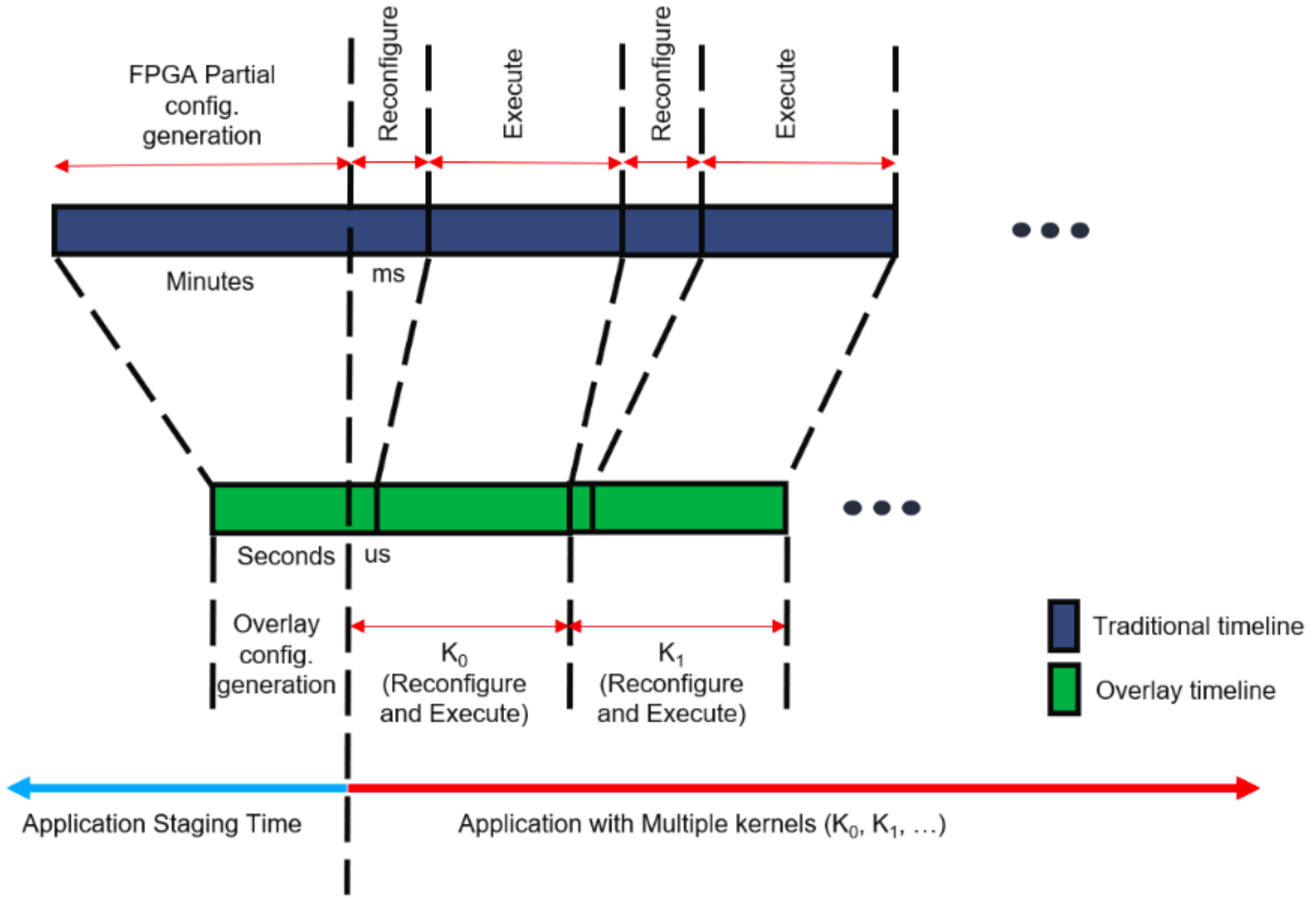
On-chip just-in-time compilation

- Simpler compilation for overlays can run on embedded processors in under a second vs minutes for traditional flow on workstation
- Processor awareness of overlay state can allow dynamic runtime allocation of resources
- New kernels can be compiled and mapped at runtime



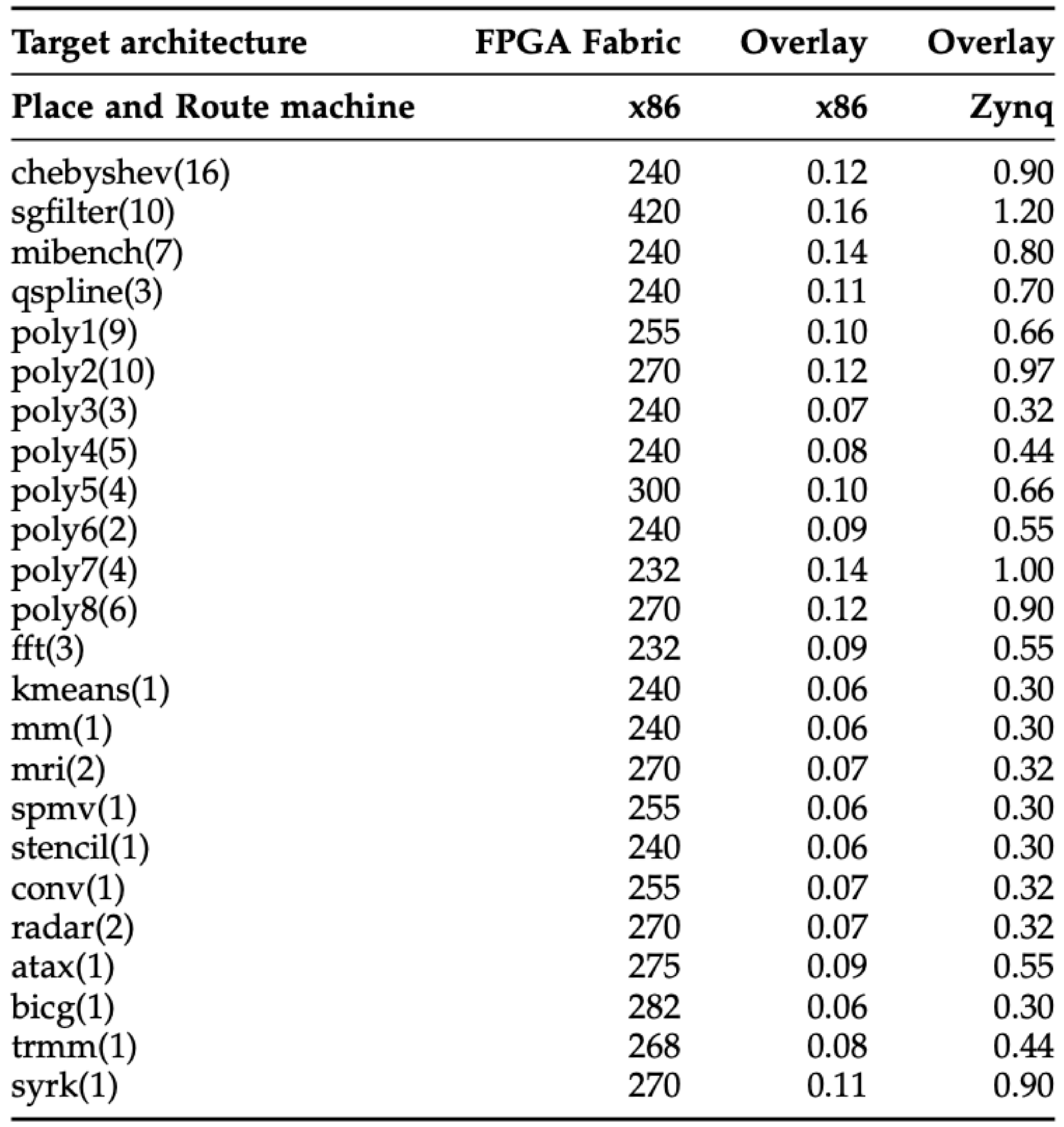
On-chip just-in-time compilation

- We propose a workflow where individual kernels are prepared in advance during application staging
- Overlay makes this rapid and the runtime switching between kernels an order of magnitude faster than traditional PR


FPGA Partial config. generation
Reconfigure
Execute
Reconfigure
Execute
Minutes
ms
Seconds
us
Overlay config. generation
K_0 (Reconfigure and Execute)
K_1 (Reconfigure and Execute)
Traditional timeline
Overlay timeline
Application Staging Time
Application with Multiple kernels (K_0, K_1, …)

On-chip just-in-time compilation

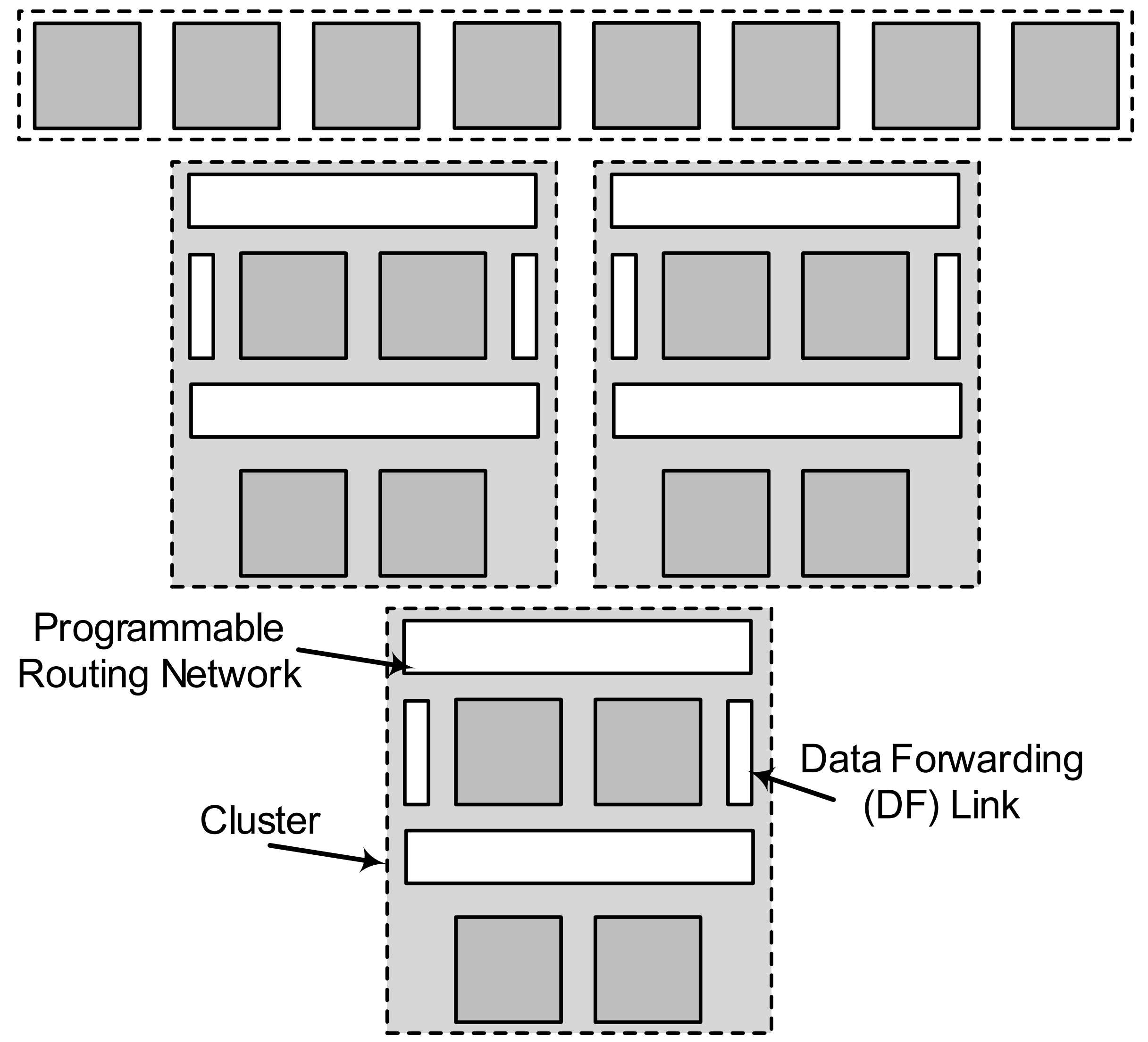
- The simpler compilation flow can run on embedded processor, e.g. Arm in Zynq in under a second
- Processor is aware of the state of the overlay at any point in time –dynamic allocation including replication, placement and routing on the overlay
- New kernels can be compiled and mapped at runtime



Target architecture	FPGA Fabric	Overlay	Overlay
Place and Route machine	**x86**	**x86**	**Zynq**
chebyshev(16)	240	0.12	0.90
sgfilter(10)	420	0.16	1.20
mibench(7)	240	0.14	0.80
qspline(3)	240	0.11	0.70
poly1(9)	255	0.10	0.66
poly2(10)	270	0.12	0.97
poly3(3)	240	0.07	0.32
poly4(5)	240	0.08	0.44
poly5(4)	300	0.10	0.66
poly6(2)	240	0.09	0.55
poly7(4)	232	0.14	1.00
poly8(6)	270	0.12	0.90
fft(3)	232	0.09	0.55
kmeans(1)	240	0.06	0.30
mm(1)	240	0.06	0.30
mri(2)	270	0.07	0.32
spmv(1)	255	0.06	0.30
stencil(1)	240	0.06	0.30
conv(1)	255	0.07	0.32
radar(2)	270	0.07	0.32
atax(1)	275	0.09	0.55
bicg(1)	282	0.06	0.30
trmm(1)	268	0.08	0.44
syrk(1)	270	0.11	0.90

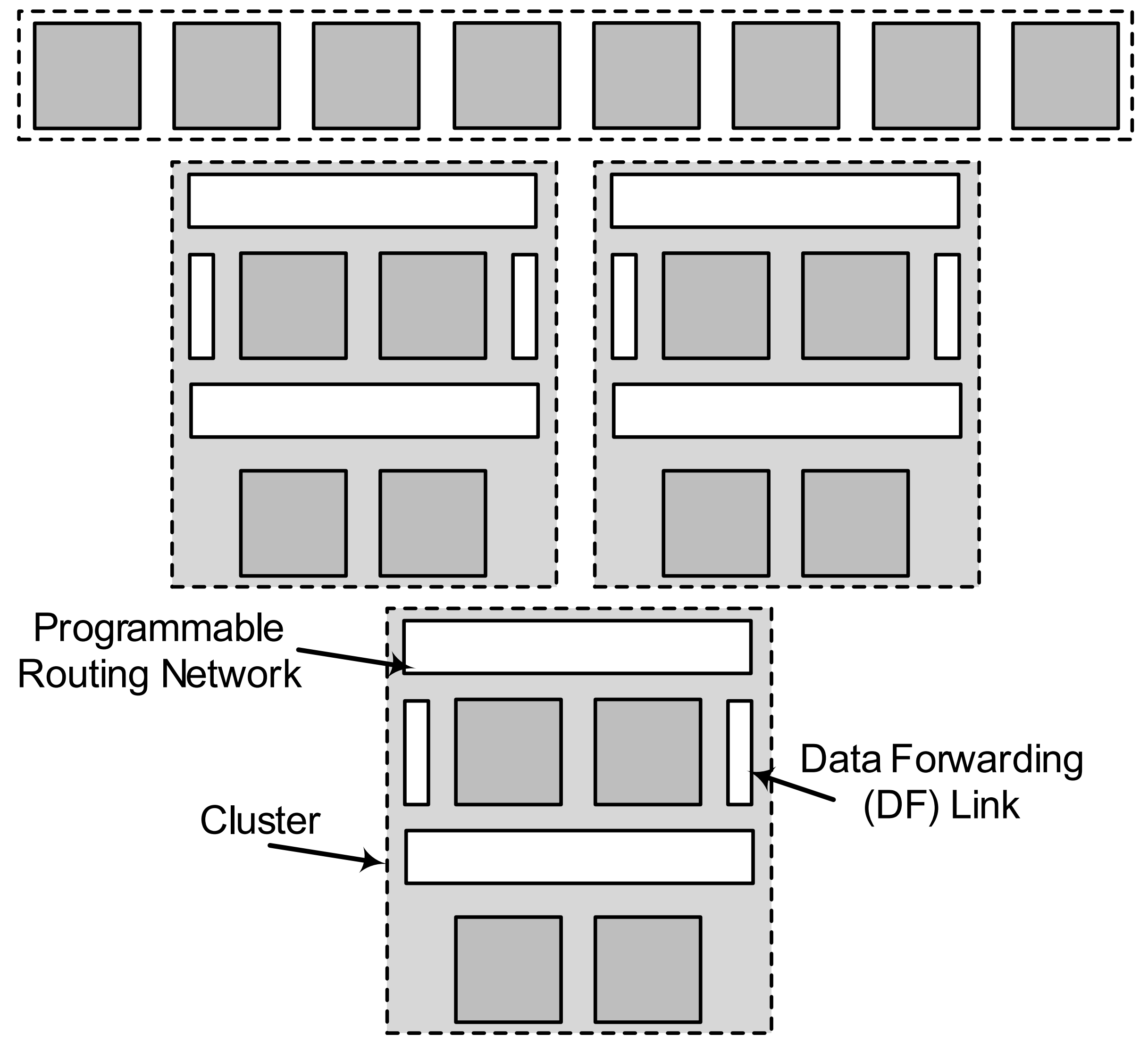
Optimising overlay routing

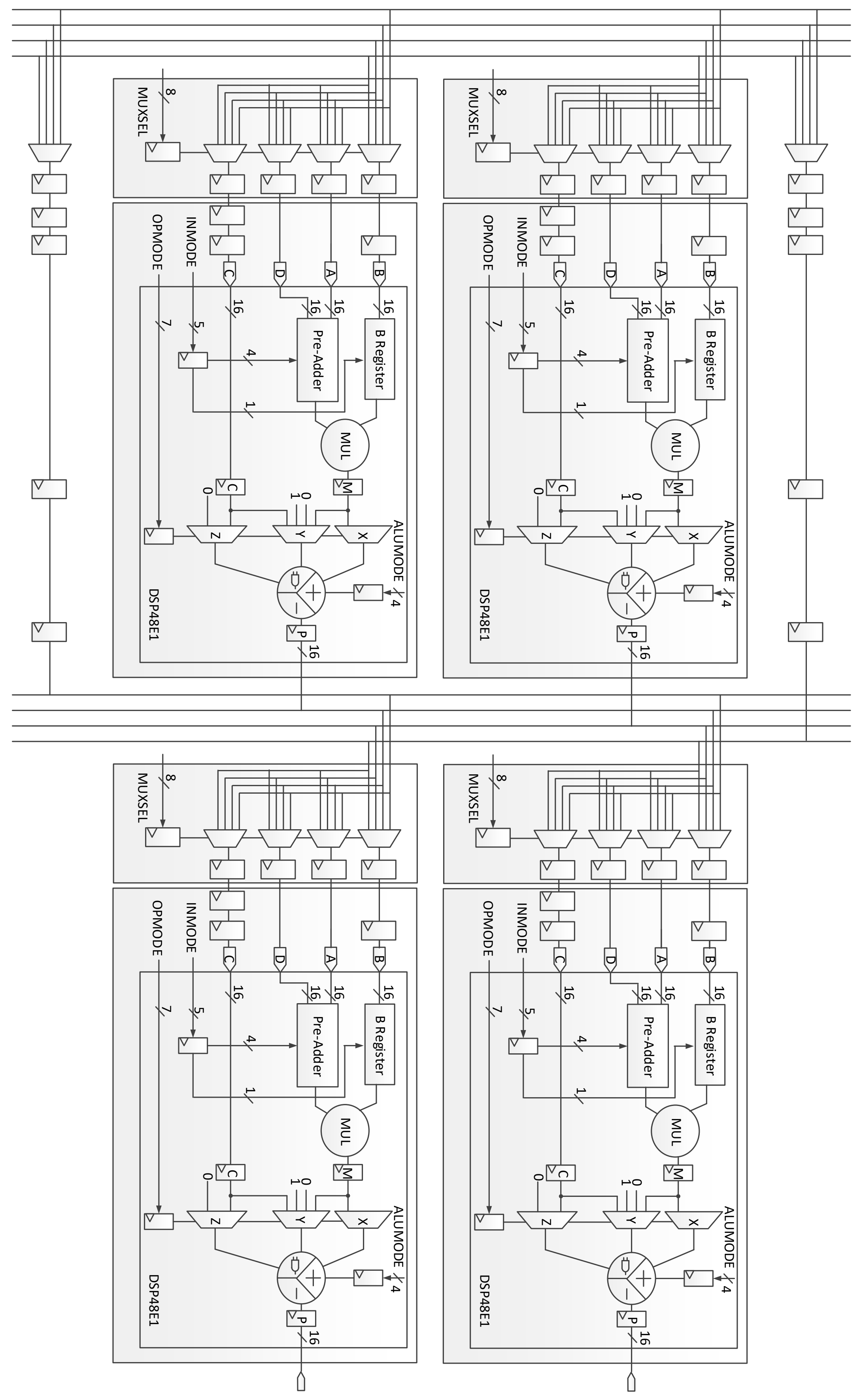
- Flexible island-style interconnect consumes significant area
- Most arithmetic functions are reductions of some sort – tailor the interconnect to reduce its overhead: DeCO
- Optimised around a set of benchmark feed forward dataflow graphs but more general



Optimising overlay routing

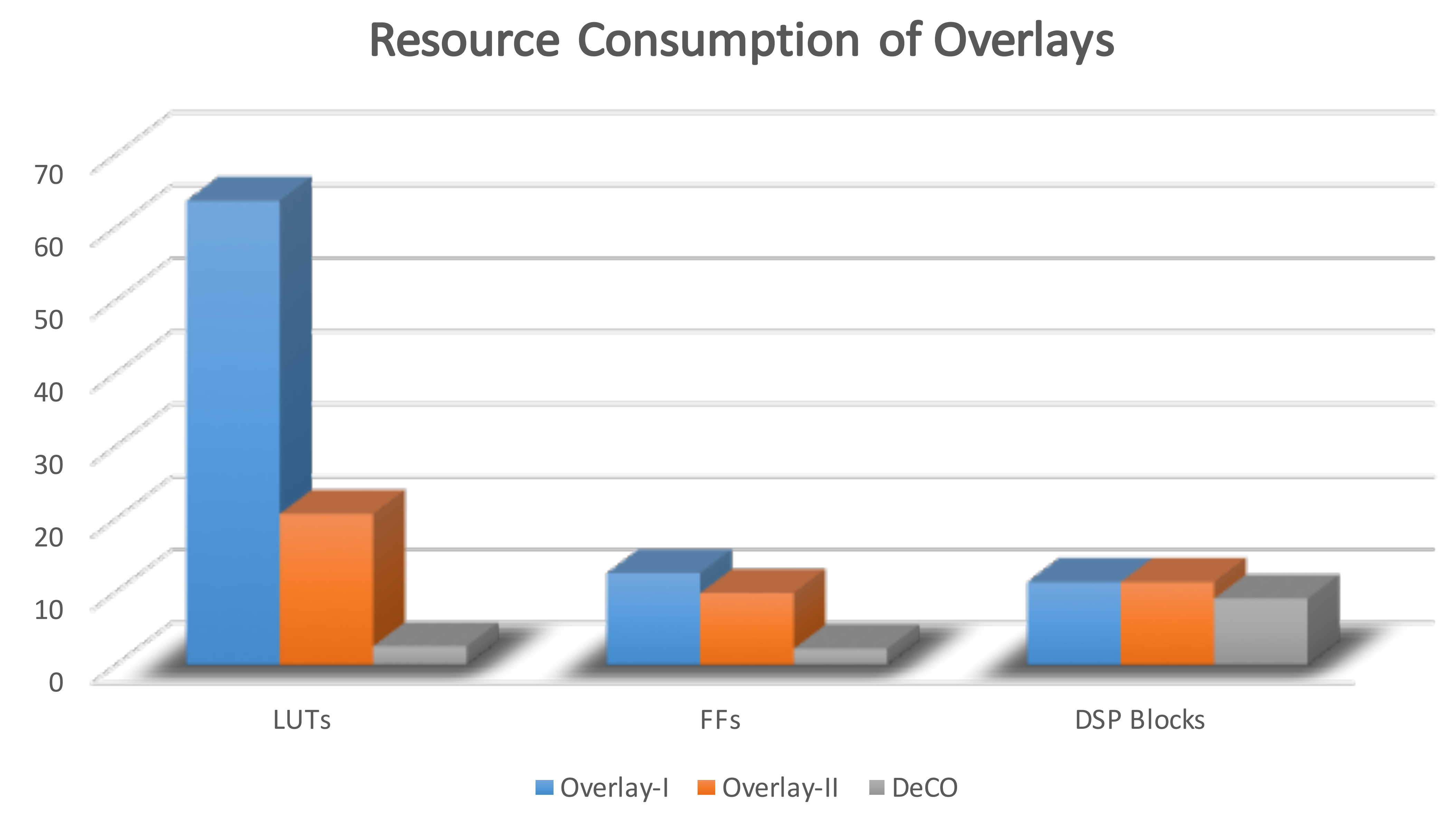
- Flexible island-style interconnect consumes significant area
- Most arithmetic functions are reductions of some sort – tailor the interconnect to reduce its overhead: DeCO
- Optimised around a set of benchmark feed forward dataflow graphs but more general





Optimising overlay routing

- Compared DeCO against two overlays:
  - 5x5 DSP block based DySER overlay (Overlay-I)
  - 5x5 DSP block based island-style overlay (Overlay-II)
- Significant LUT savings of 87–96% compared to Overlay-II and I
- Reconfiguration in 2us vs 382us for standard FPGA flow



References

- **Mapping for Maximum Performance on FPGA DSP Blocks**
  B. Ronak and S. A. Fahmy
  IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, vol. 35, no. 4, Apr 2016

- **Multi-pumping Flexible DSP Blocks for Resource Reduction on Xilinx FPGAs**
  B. Ronak and S. A. Fahmy
  IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, vol. 36, no. 9, Sep 2017

- **Coarse Grained FPGA Overlay for Rapid Just-In-Time Accelerator Compilation**
  A. K. Jain, D. L. Maskell, S. A. Fahmy
  IEEE Transactions on Parallel and Distributed Systems, vol. 33 no. 6, Jun 2022.

- **Streaming Overlay Architecture for Lightweight LSTM Computation on FPGA SoCs**
  L. Ioannou, S. A. Fahmy
  ACM Transactions on Reconfigurable Technology and Systems, vol. 16 no. 1, Mar 2023

Other uses of DSP blocks

- Multiple wide multiplexers within can be used for routing word-level signals around architectures – HopLiteDSP
- Cascade connections and proximity to block RAMs can build large GEMM structures at very high frequency – Cascades, SuperTile
- Any compile flow that deals with graphs of mul/sub/add can be more efficient by mapping to DSPs with architecture information

Current areas of research

- Integrating DSP blocks into the FU designs of overlay/CGRA generators
  - Requires adapted scheduling to accommodate deeper pipeline depths
- Use in dynamically-scheduled dataflow circuits
  - Since dynamic control is per arithmetic node, clustering arithmetic can reduce control overhead
- Building large multipliers that achieve maximum DSP block frequency
  - Using e-graphs to capture primitive structure

Opportunities

- Today's more structurally aware compilers can capture architectural features and map to them efficiently
- Let's throw away RTL as the IR for high level architecture design – flexible pipelining is key
- FPGAs are a compute platform missing better tooling, not a prototyping platform!

About KAUST

- One the shores of the Red Sea, an hour from the port city of Jeddah
- A graduate-only research university: MS and PhD programs
- All students receive scholarships
- Postdocs and Research Scientists receive good salaries
- Access to world-class facilities