The spinal cord is another brain

nature neuroscience

BRIEF COMMUNICA

https://doi.org/10.1038/s41593

Spinal stretch reflexes support efficient hand control

Jeffrey Weiler[1,2,3*], Paul L. Gribble[1,2,3] and J. Andr

Motor behaviour is most efficiently controlled by correcting only disturbances that influence task success. It is currently thought that such control is computed within a transcortical feedback pathway. Here we show that, for postural hand control, even the fastest spinal feedback pathway can produce efficient corrective responses, forcing a re-evaluation of how the nervous system derives the control laws that support motor behavior.

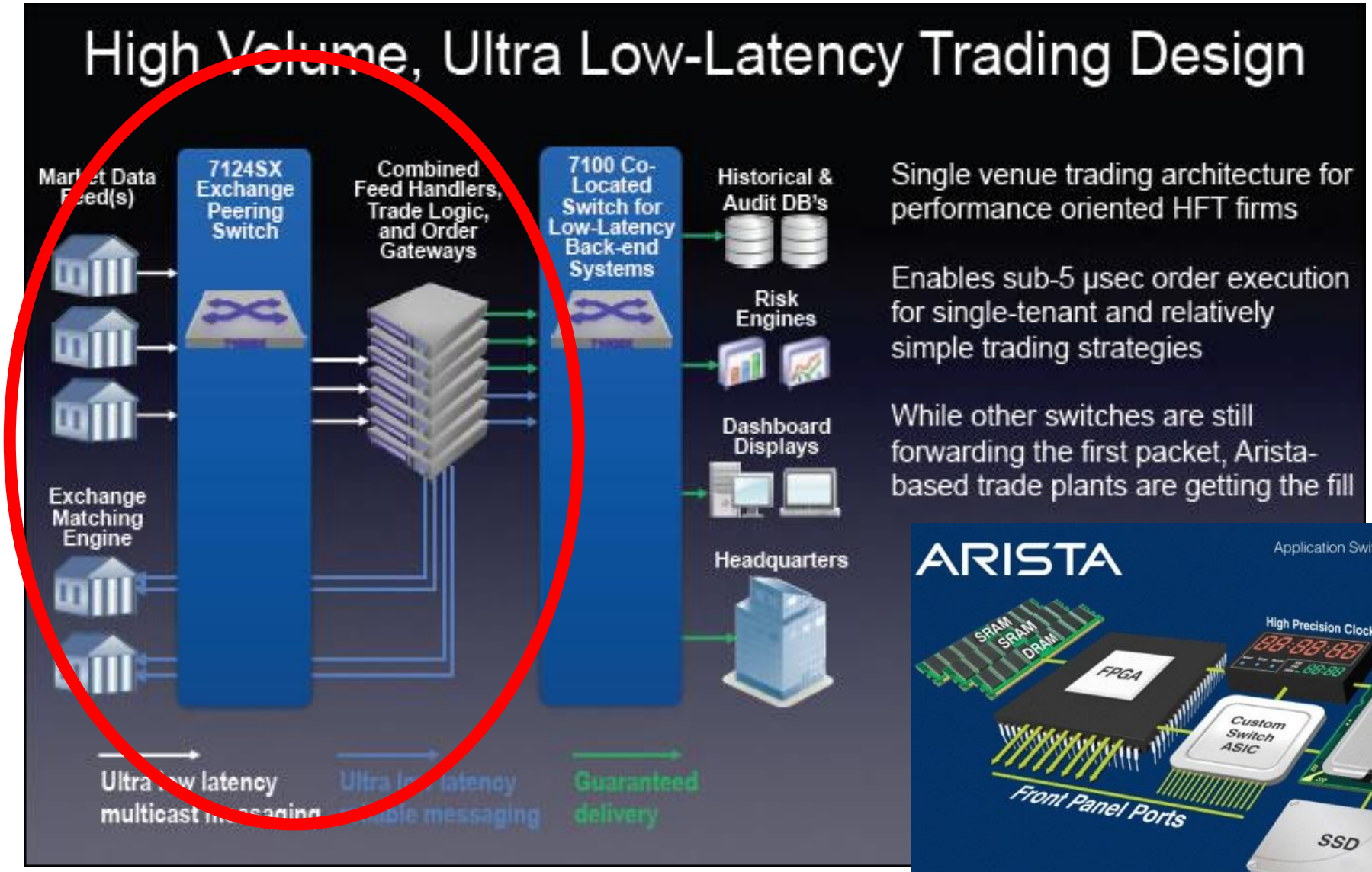
Real-world actions require active control of many joints in the presence of internal and external disturbances. The simplest way for the nervous system to counteract disturbances is to ensure that all

of a robotic exoskeleton and placed their hand at a while countering small flexion loads at the elbow a robot then mechanically flexed their elbow, stretchi

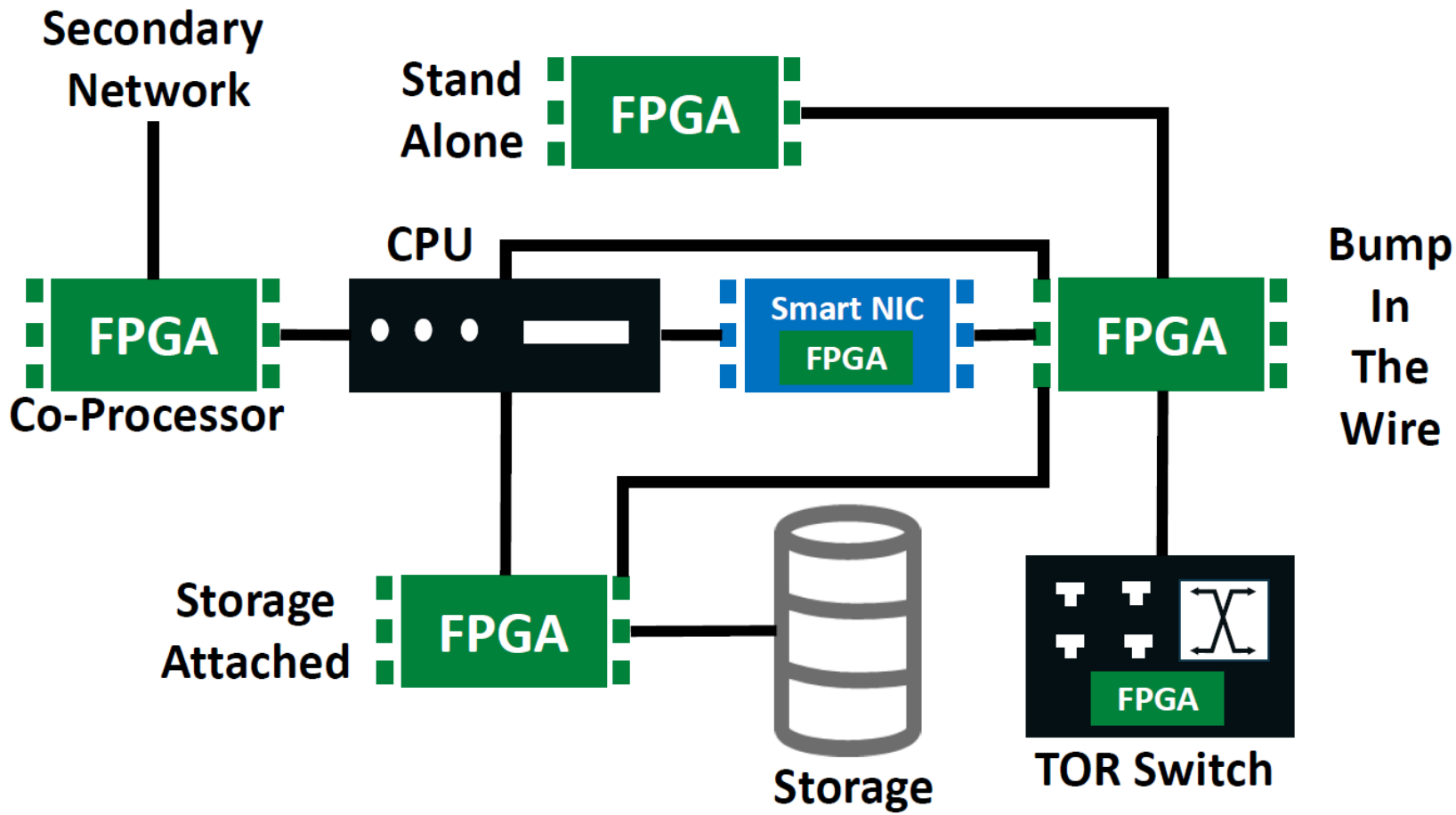
It is currently thought that such control is computed within [the brain].

Here we show that … the [spinal cord produces] efficient corrective responses, forcing a re-evaluation.

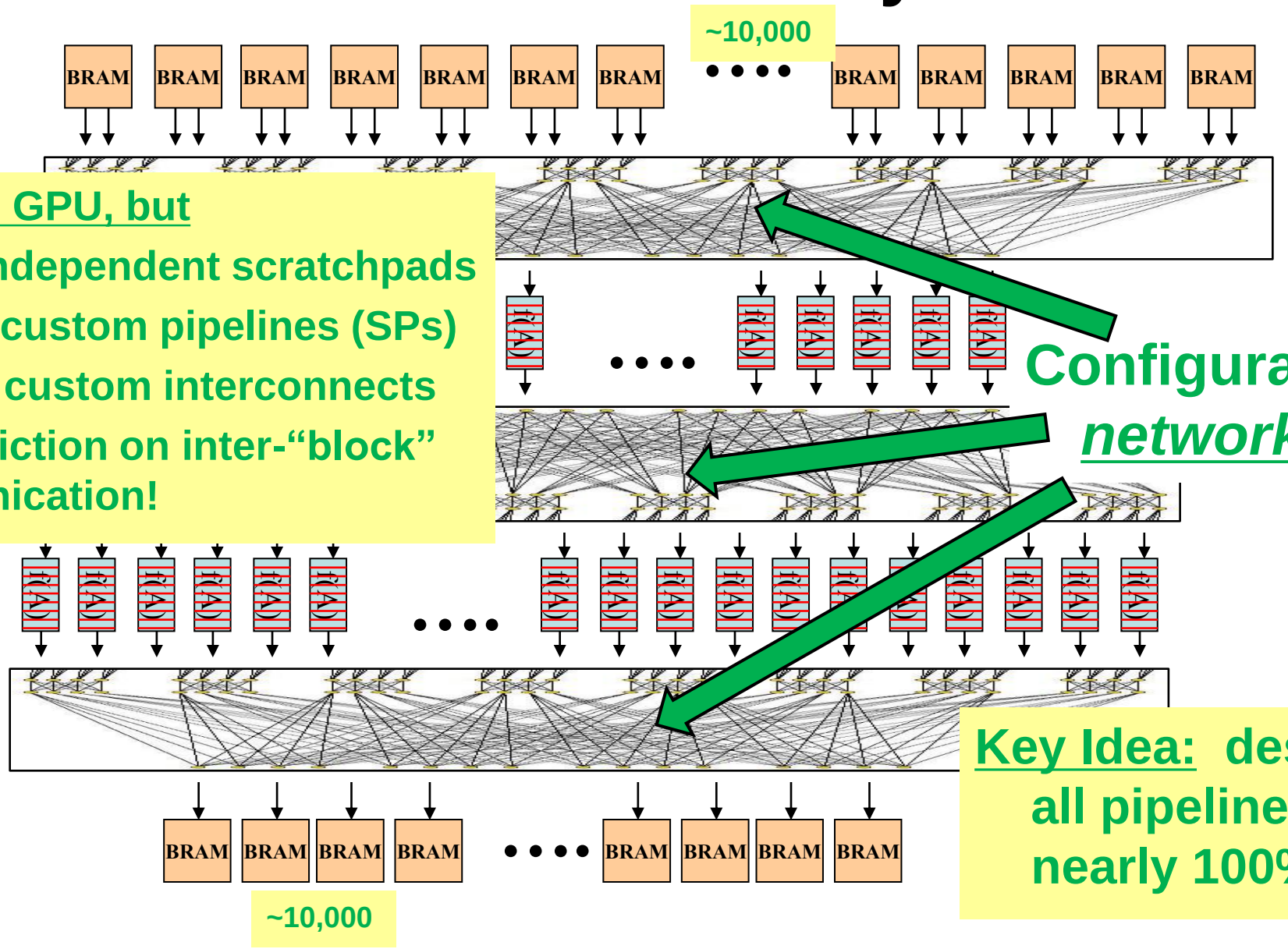
Example: *low-latency trading*



Communication & computation both done in the network switch


Secondary
Network
Stand
Alone
FPGA
CPU
FPGA
Co-Processor
Smart NIC
FPGA
FPGA
Bump
In
The
Wire
Storage
Attached
FPGA
Storage
FPGA
TOR Switch

Where does FPGA efficiency come from?



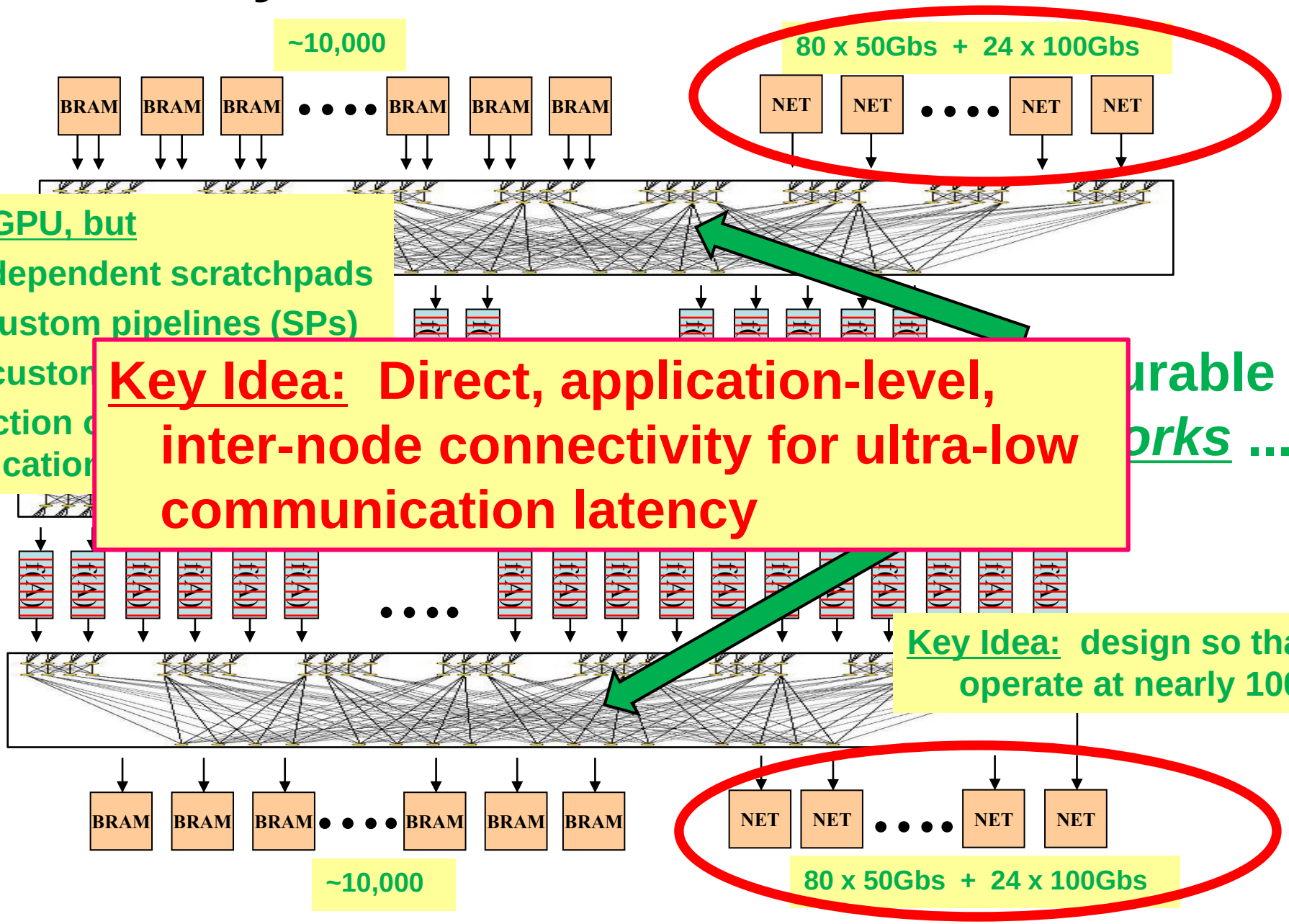
Resembles GPU, but

- → 10,000 independent scratchpads
- → Flexible custom pipelines (SPs)
- → Flexible custom interconnects
- → No restriction on inter-"block" communication!

Configurable *networks* ...

Key Idea: design so that all pipelines operate at nearly 100% efficiency

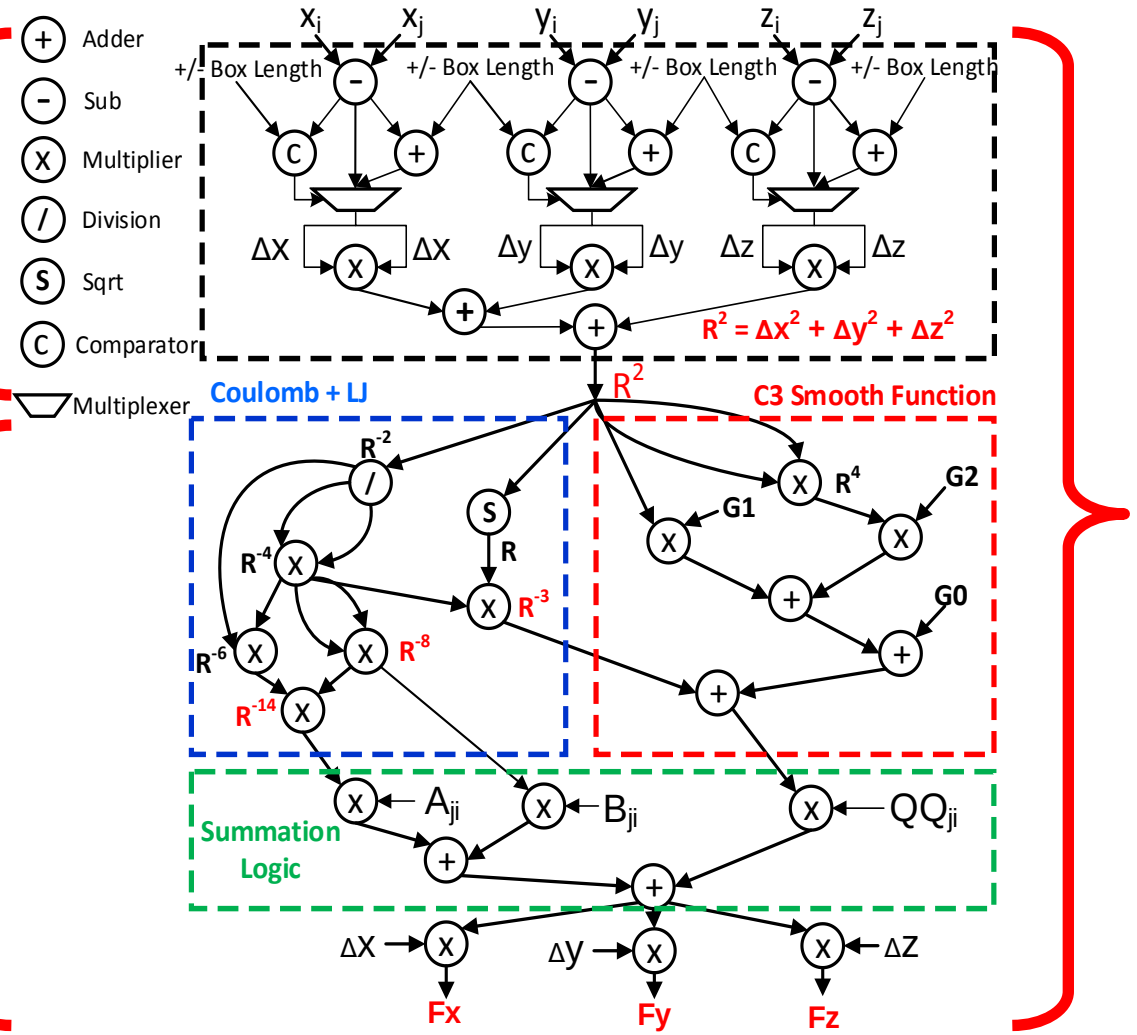
High efficiency extends to the network



Spatial Computing example – MD force pipeline

Distance computation

Force computation

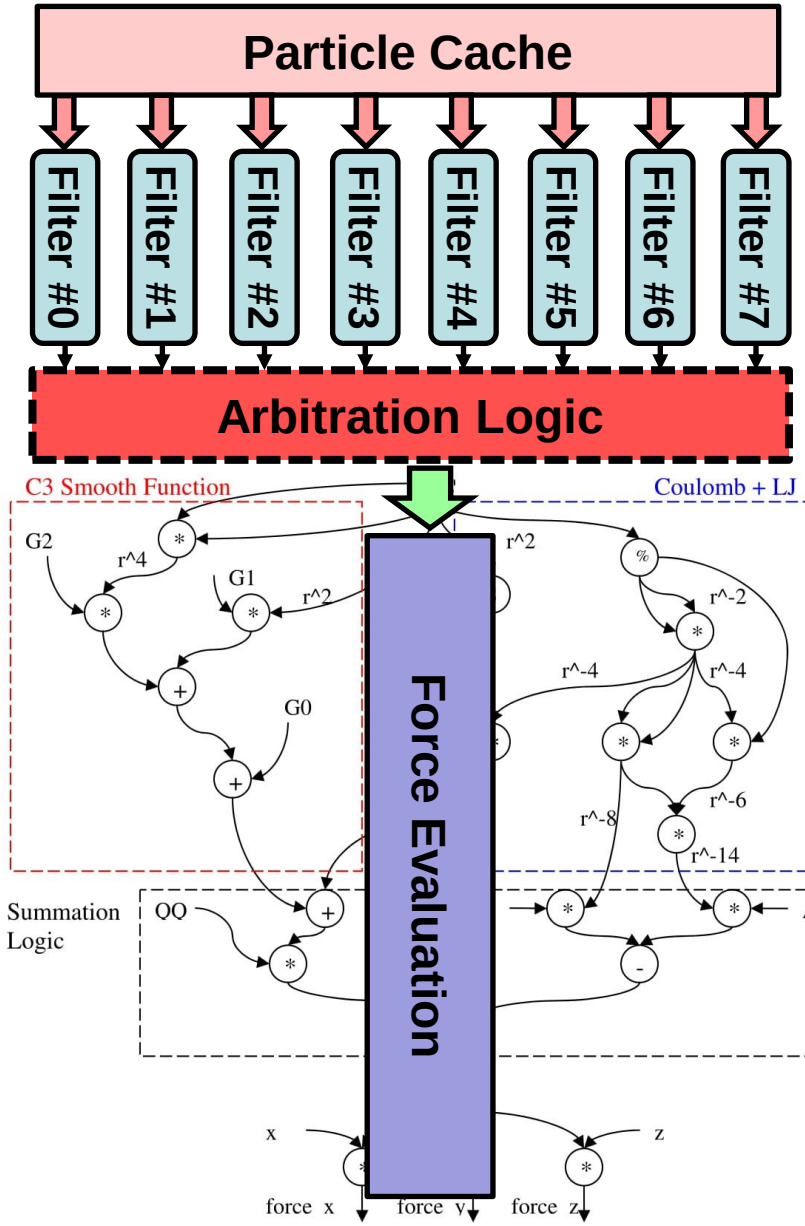


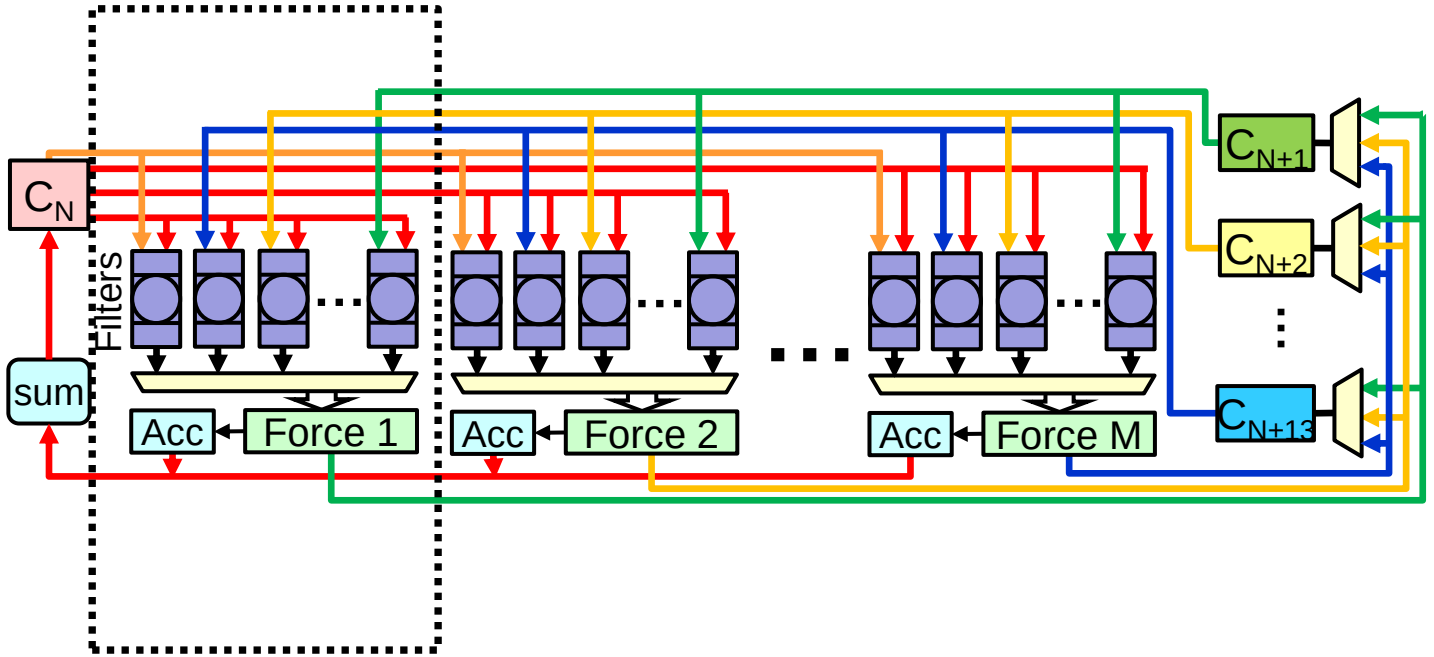
$$F_i^{LJ} = \sum_{j \neq i} \frac{\epsilon_{ab}}{\sigma_{ab}^2} \left\{ 48 \left(\frac{\sigma_{ab}}{|r_{ji}|} \right)^{14} - 24 \left(\frac{\sigma_{ab}}{|r_{ji}|} \right)^{8} \right\} \vec{r}_{ji}$$

$$F_i^{C} = \frac{q_i}{4\pi} \sum_{j \neq i} \frac{1}{\epsilon_{ab}} \left\{ \frac{1}{|r_{ji}|} \right\}^3 \vec{r}_{ji}$$

Full force pipeline

Spatial Computing – hierarchical replication





Compile statement →

a+b*c-d/(b*c)

3-address code

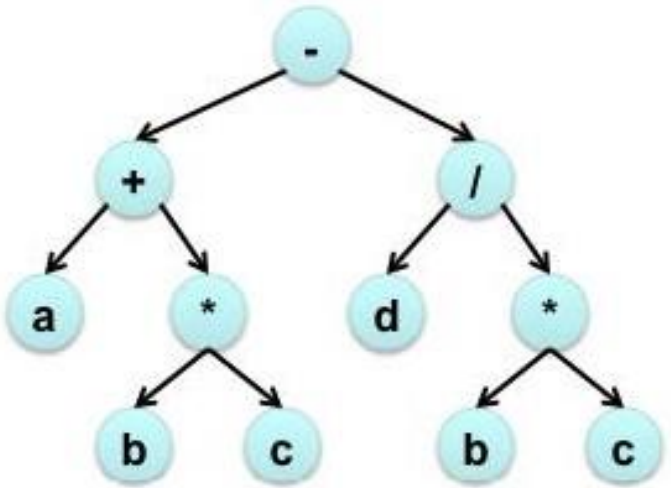
```
1  t1 = b*c
2  t2 = a+t1
3  t3 = b*c
4  t4 = d/t3
5  t5 = t2-t4
```

Quadruples

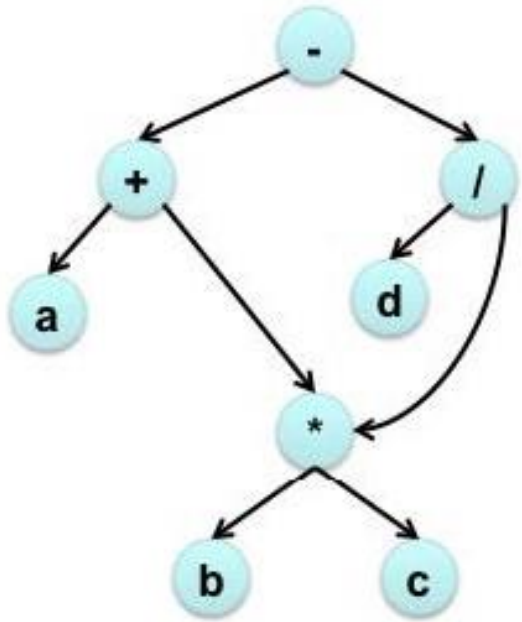
op	arg_1	arg_2	result
*	b	c	t1
+	a	t1	t2
*	b	c	t3
/	d	t3	t4
-	t2	t4	t5

Triples

	op	arg_1	arg_2
0	*	b	c
1	+	a	(0)
2	*	b	c
3	/	d	(2)
4	-	(1)	(3)



Syntax tree

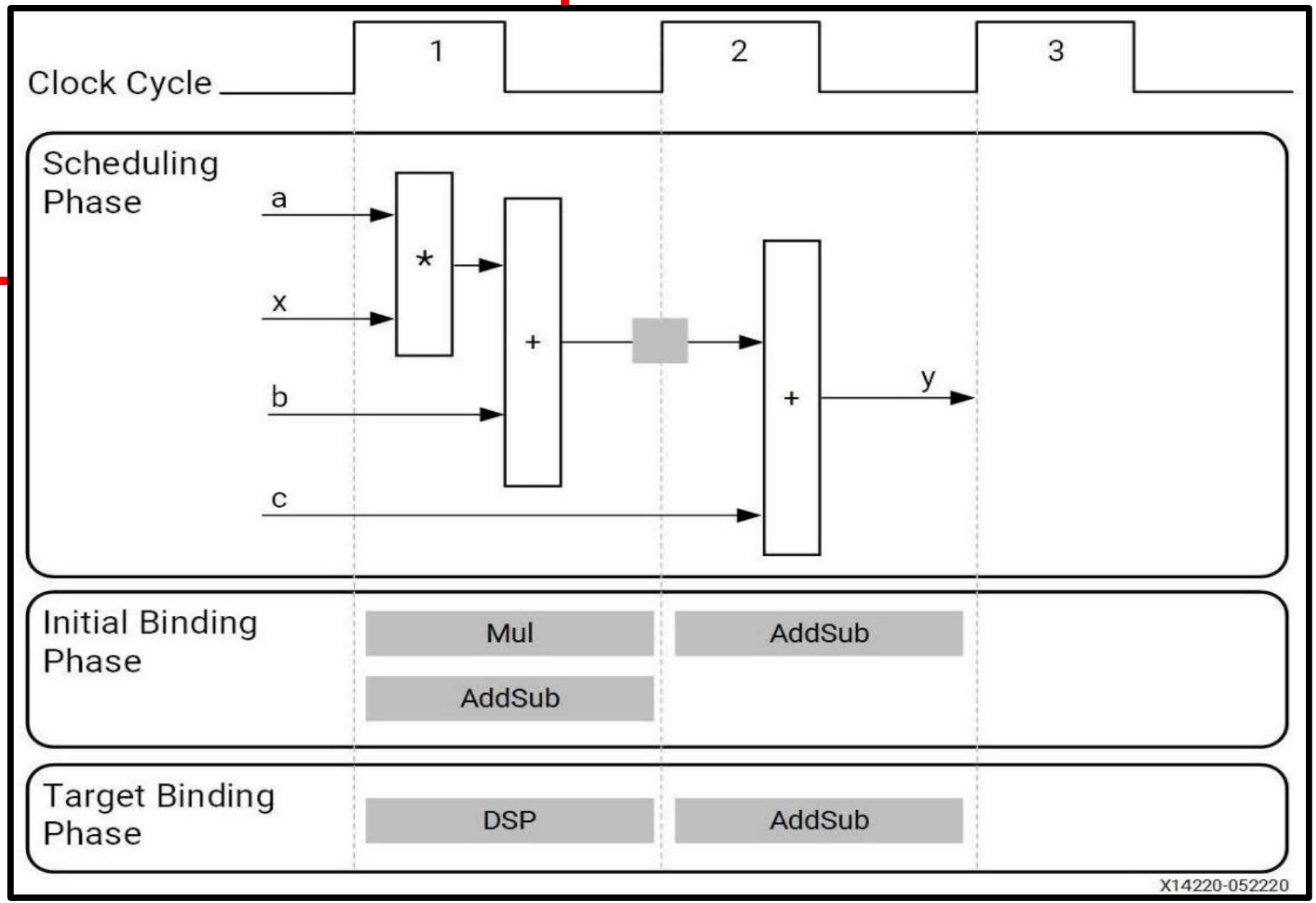


DAG

https://nptel.ac.in/content/storage2/courses/106108113/module5/Lecture17.pdf

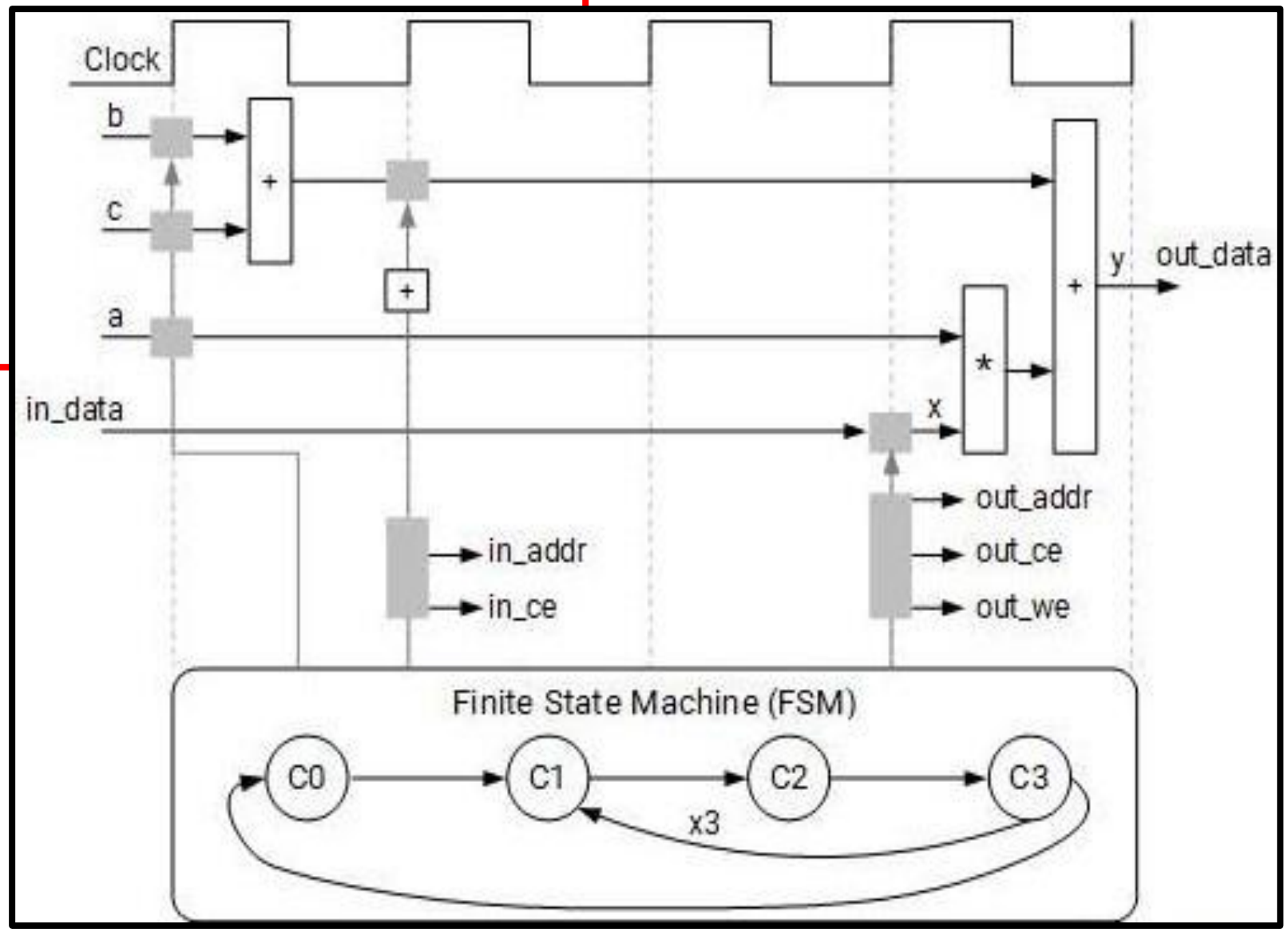
```
int foo(char x, char a, char b, char c) {
 char y;
 y = x*a+b+c;
 return y;
}
```

Xilinx Vitis documentation

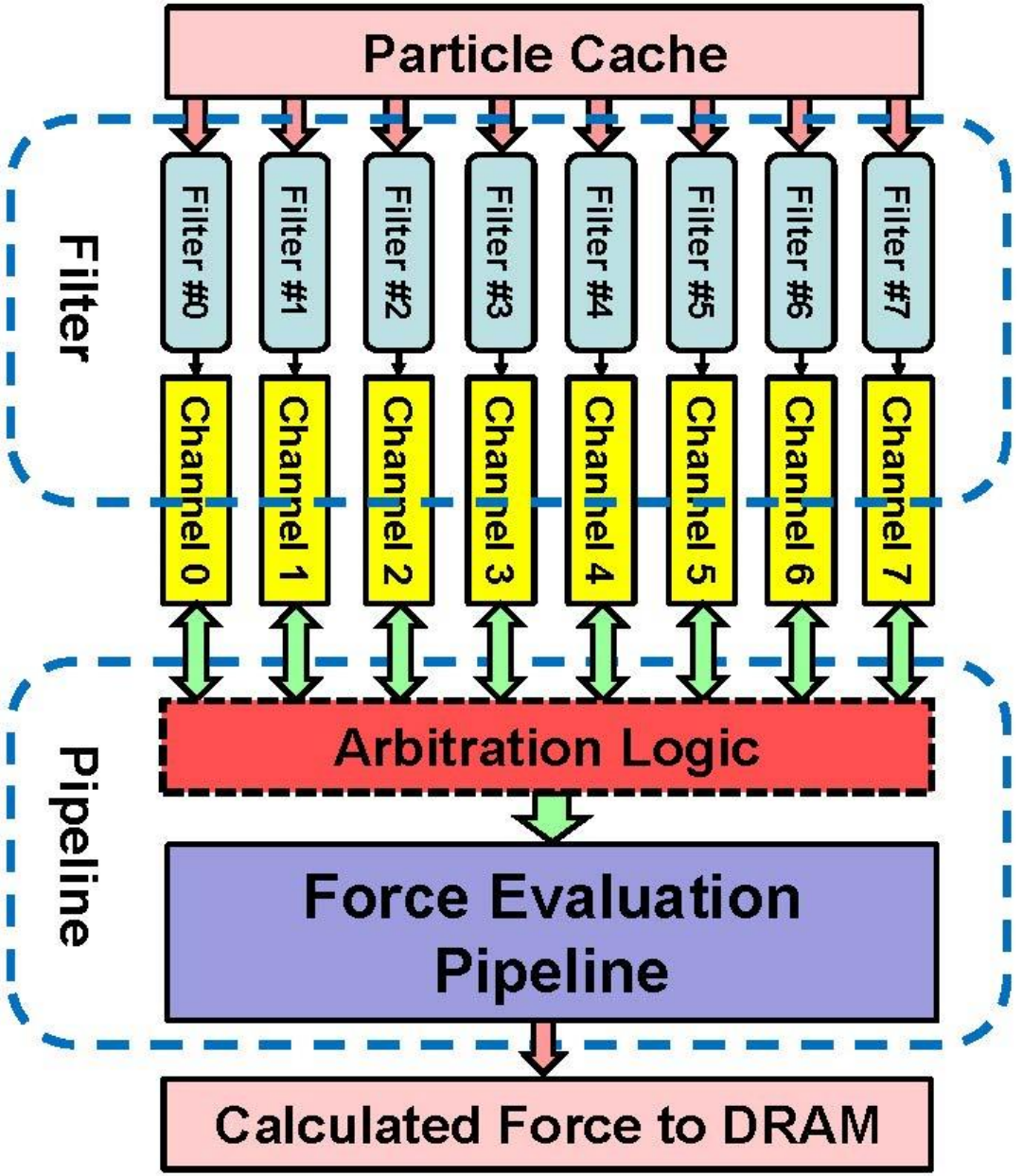


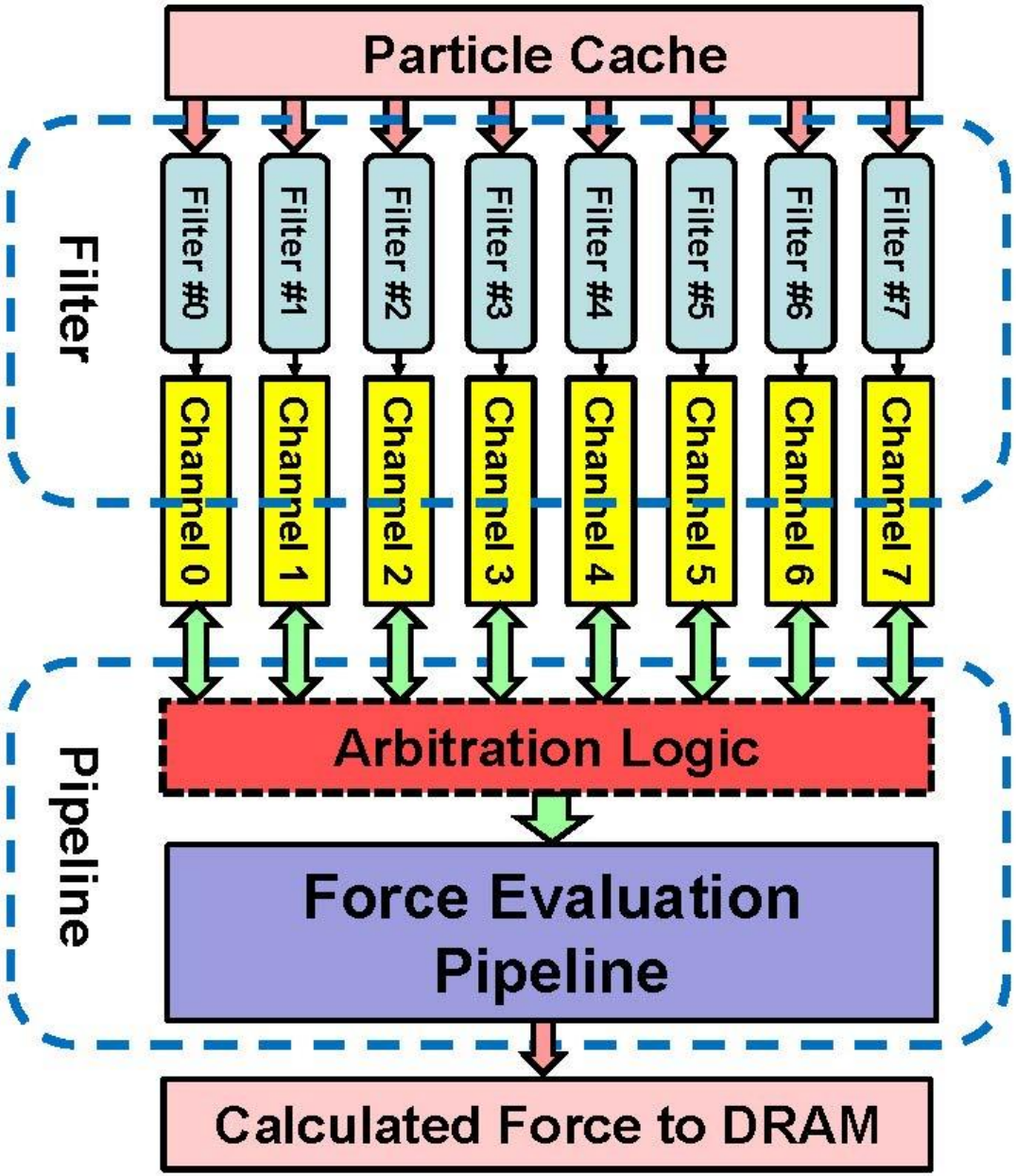
```
void foo(int in[3], char a, char b, char c, int out[3]) {
  int x,y;
  for(int i = 0; i < 3; i++) {
    x = in[i];
    y = a*x + b + c;
    out[i] = y;
  }
}
```

Xilinx Vitis documentation

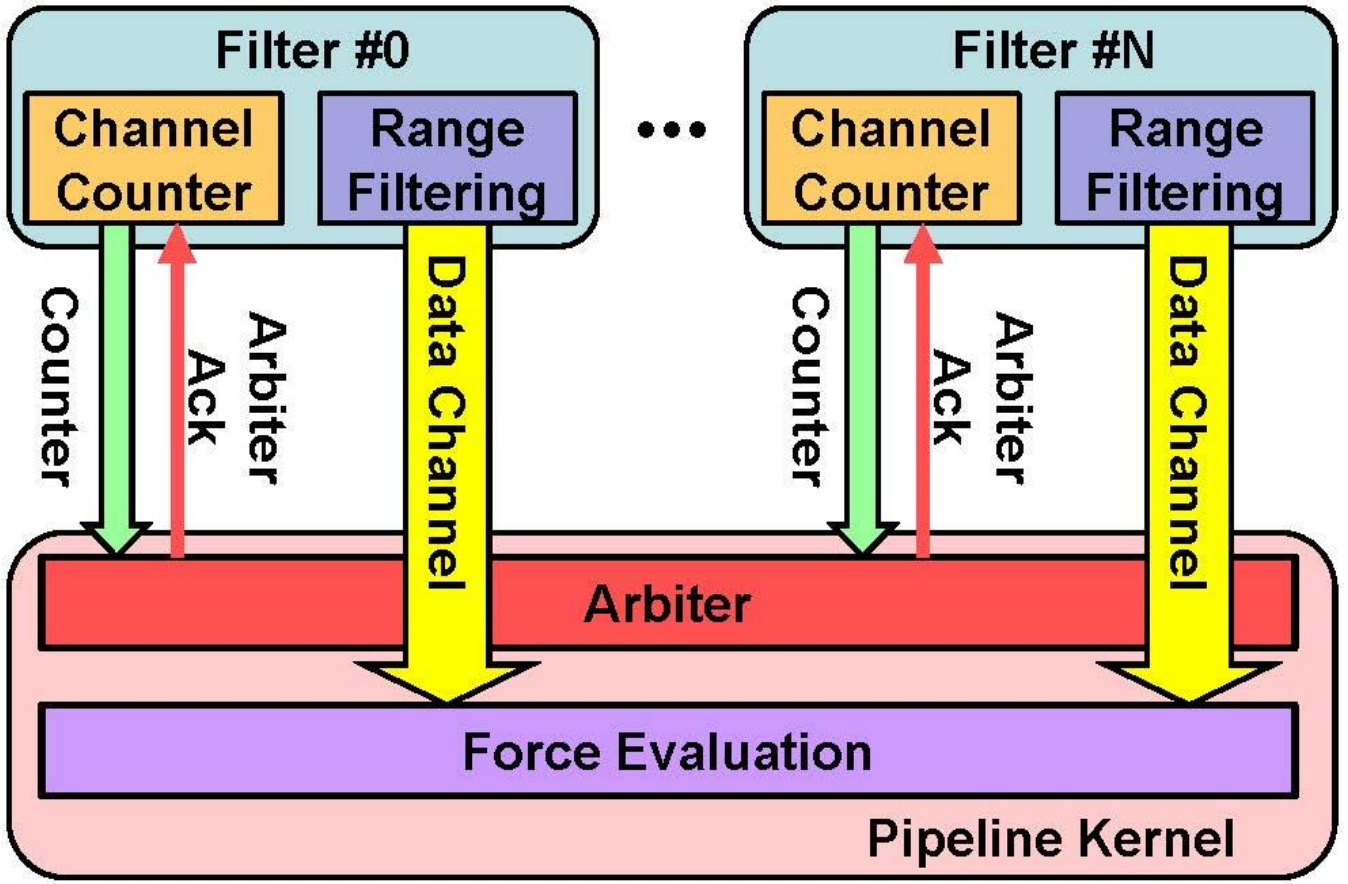


Original Design

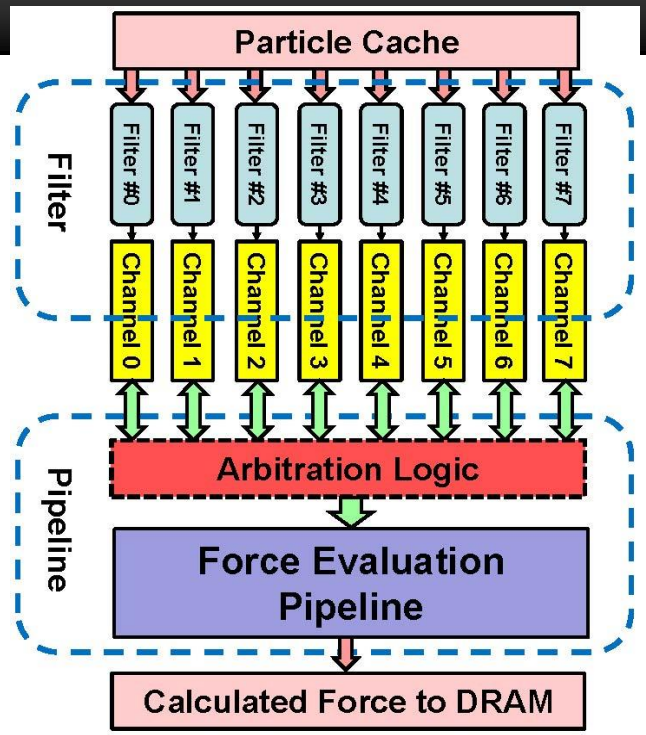


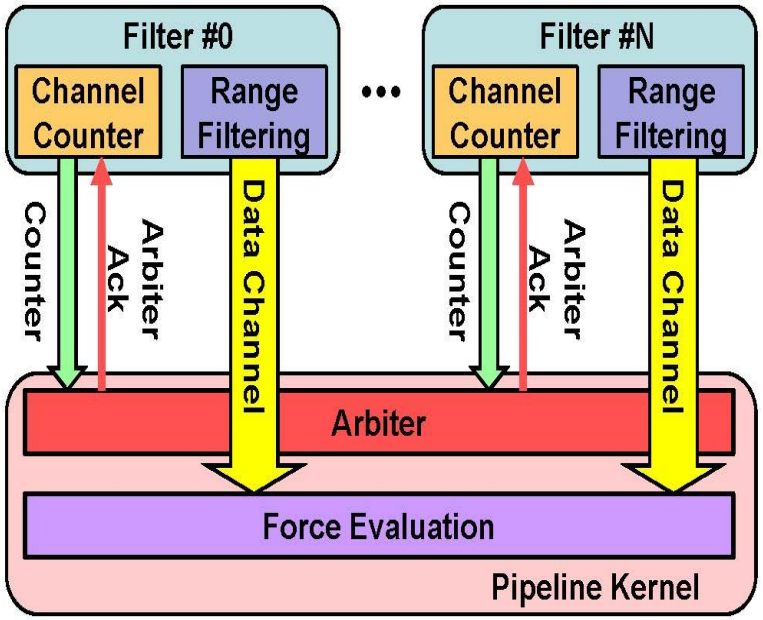


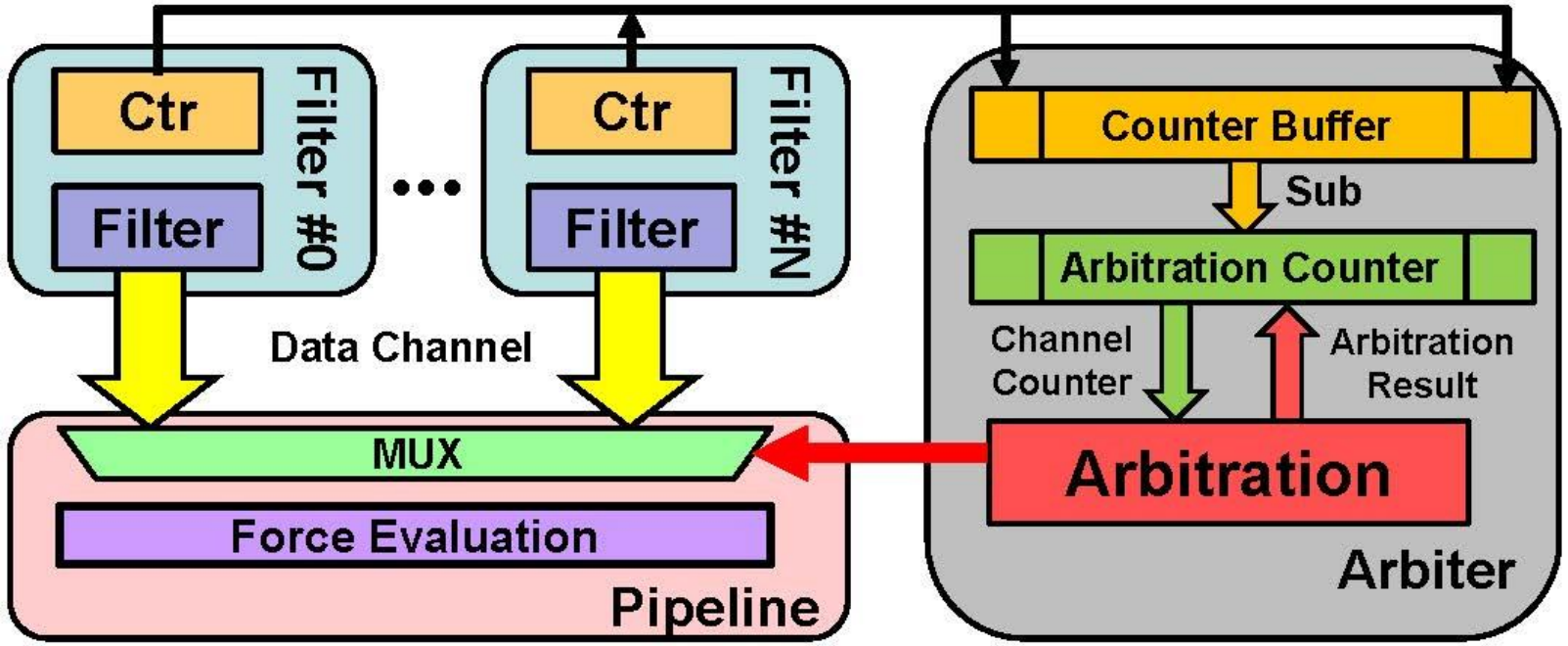
Result from HLS

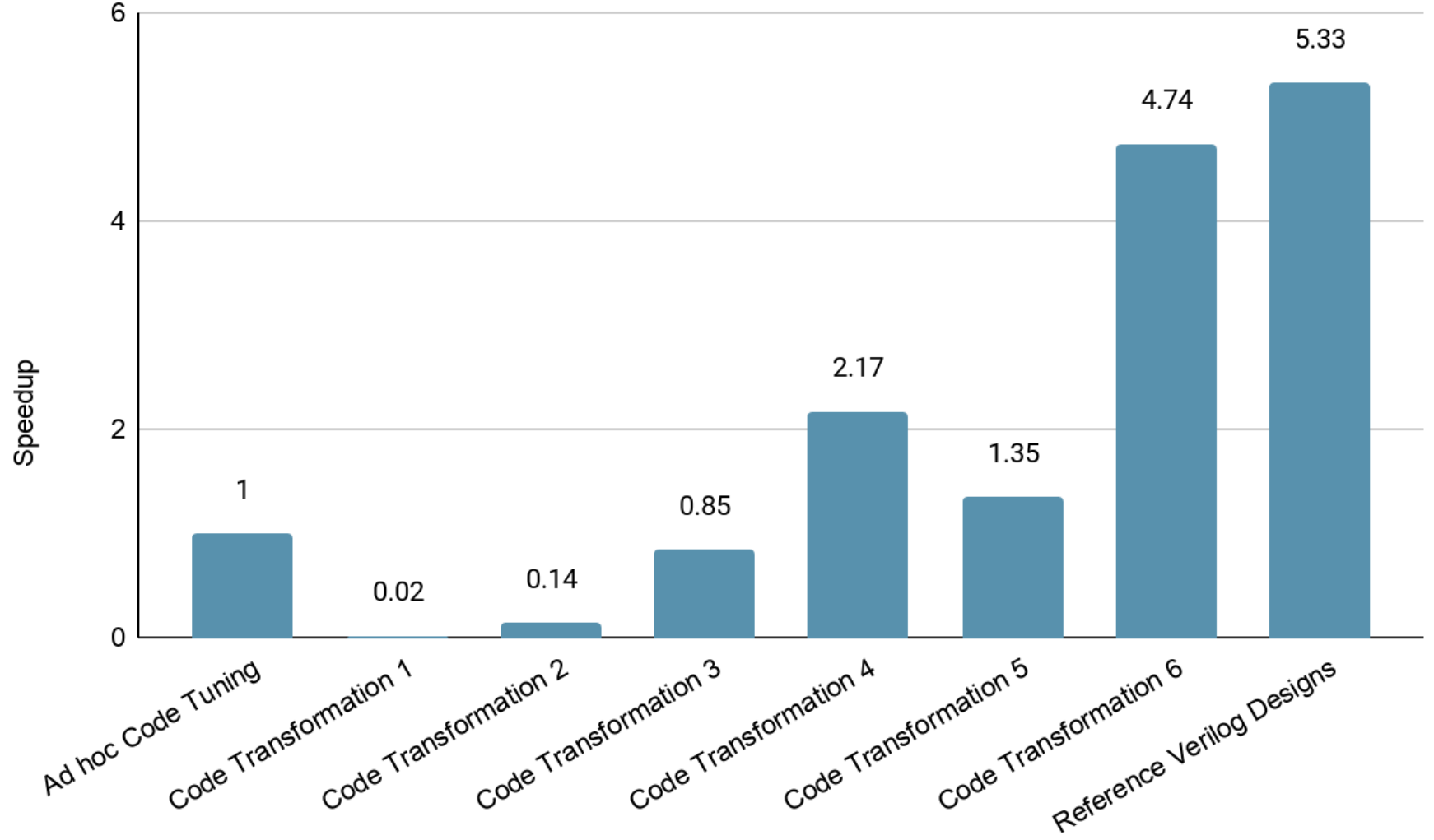


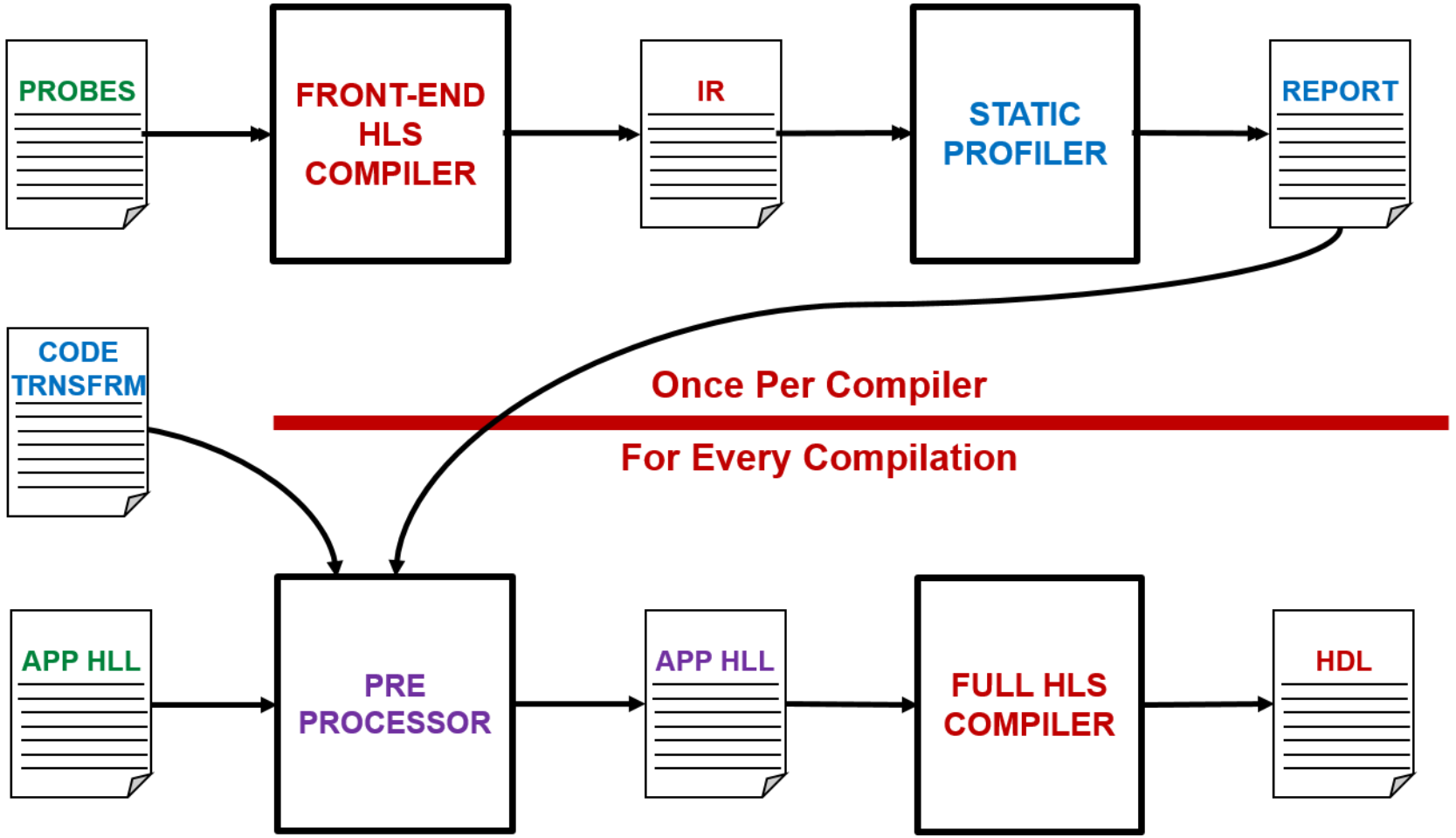
HLS result after refactoring input code

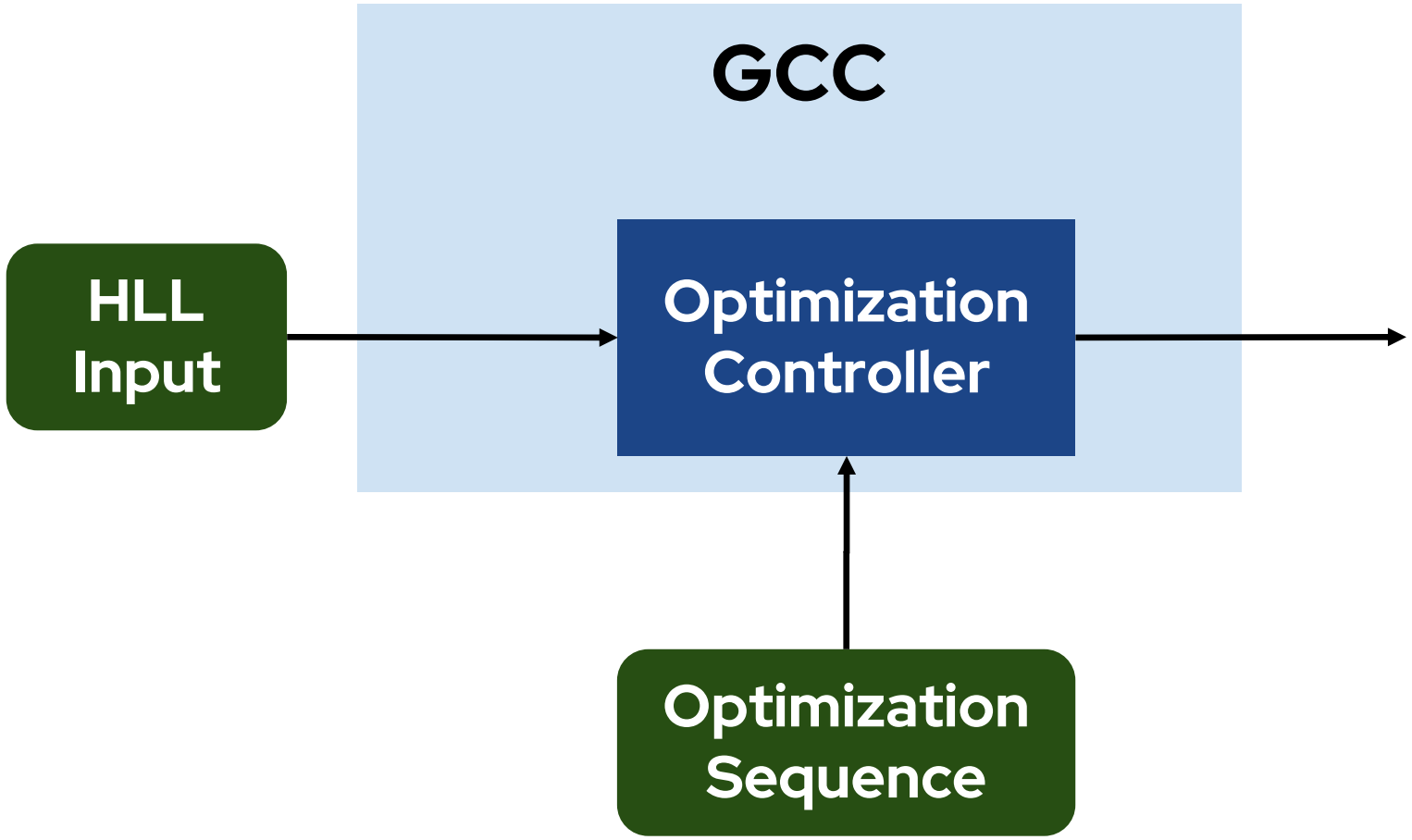




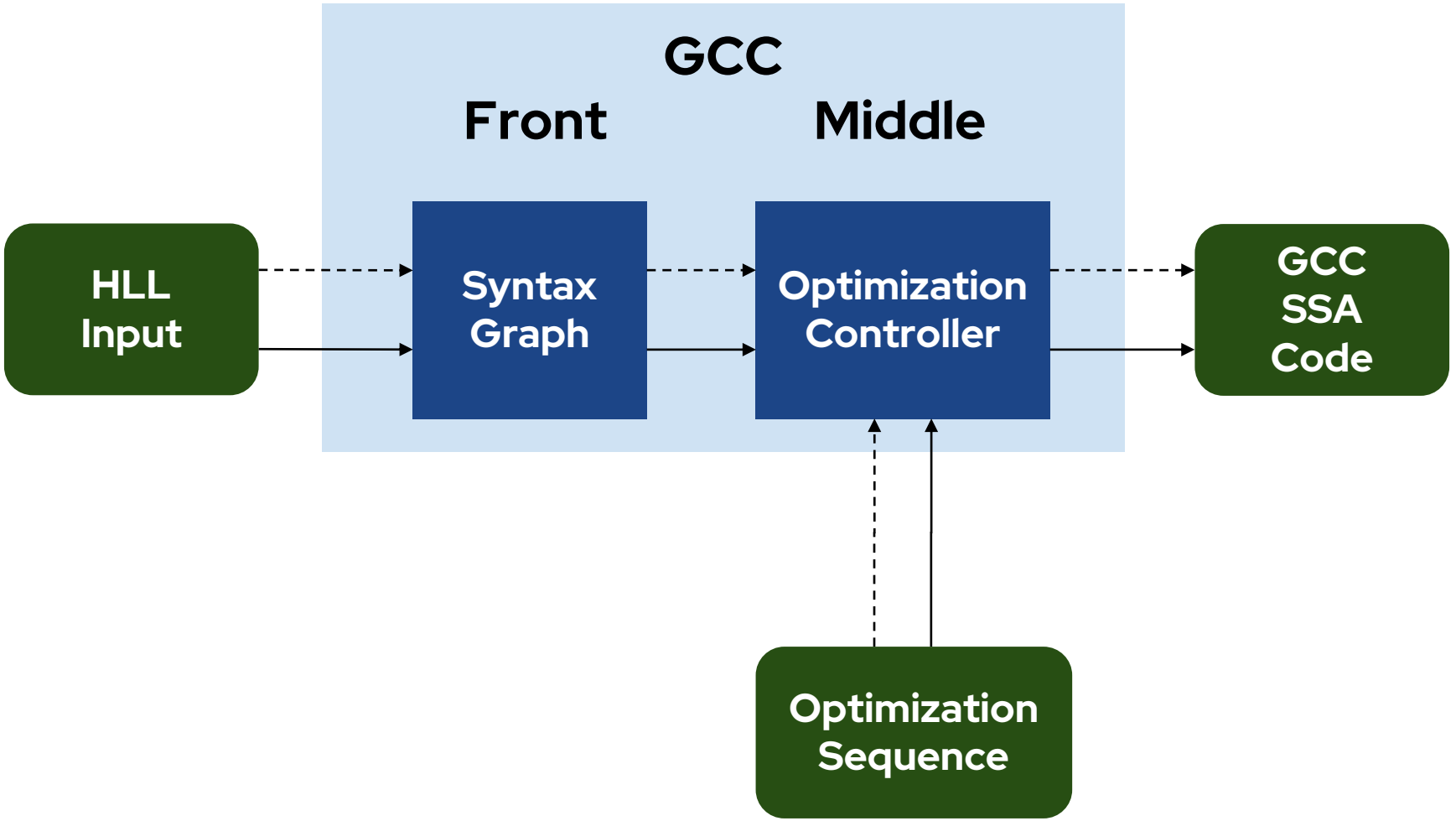



Speedup over ad hoc HLS code tuning
Speedup
6
4
2
0
1
0.02
0.14
0.85
2.17
1.35
4.74
5.33
Ad hoc Code Tuning
Code Transformation 1
Code Transformation 2
Code Transformation 3
Code Transformation 4
Code Transformation 5
Code Transformation 6
Reference Verilog Designs

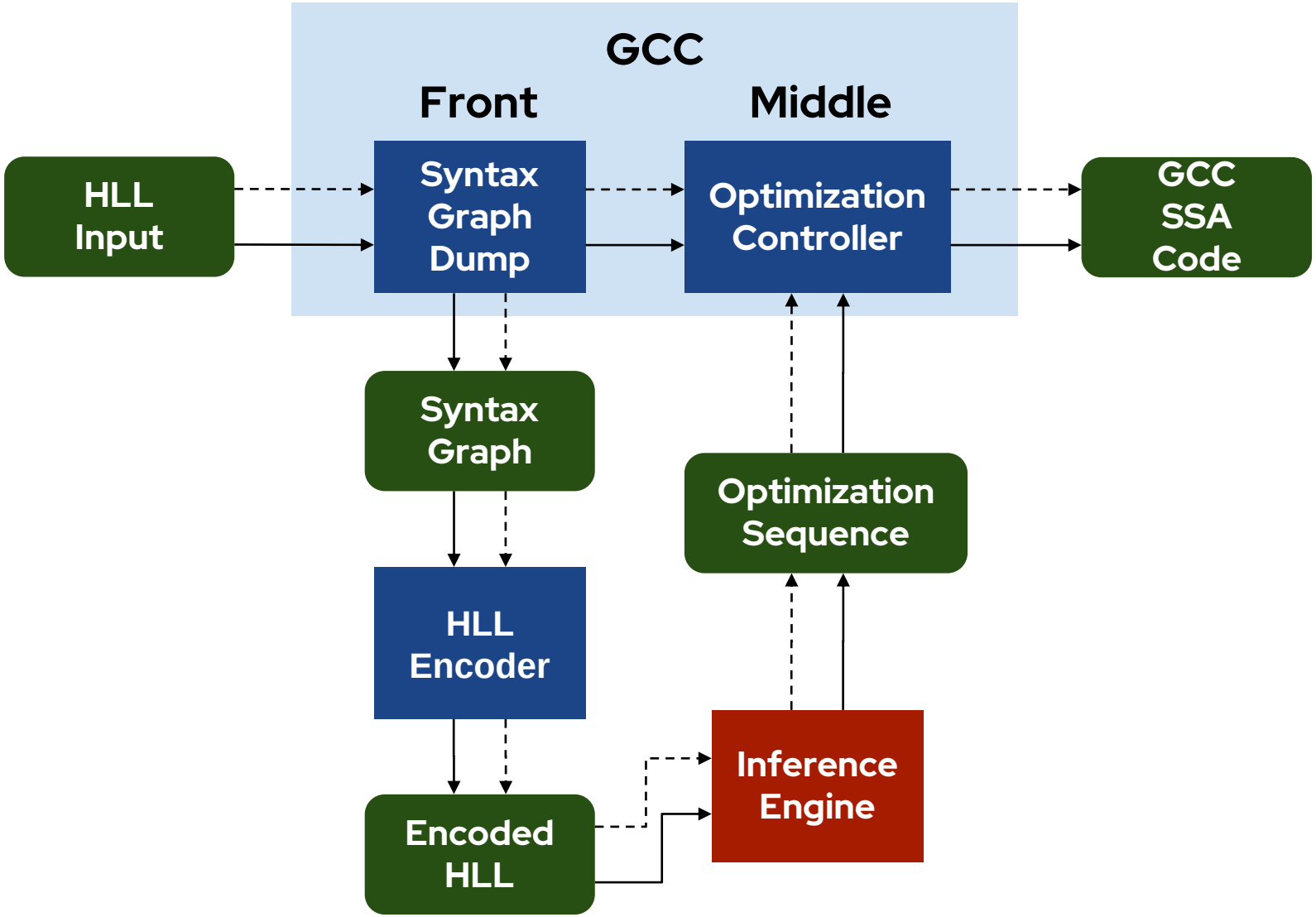

PROBES
FRONT-END HLS COMPILER
IR
STATIC PROFILER
REPORT
CODE TRNSFRM
Once Per Compiler
For Every Compilation
APP HLL
PRE PROCESSOR
APP HLL
FULL HLS COMPILER
HDL


GCC
HLL Input
Optimization Controller
Optimization Sequence

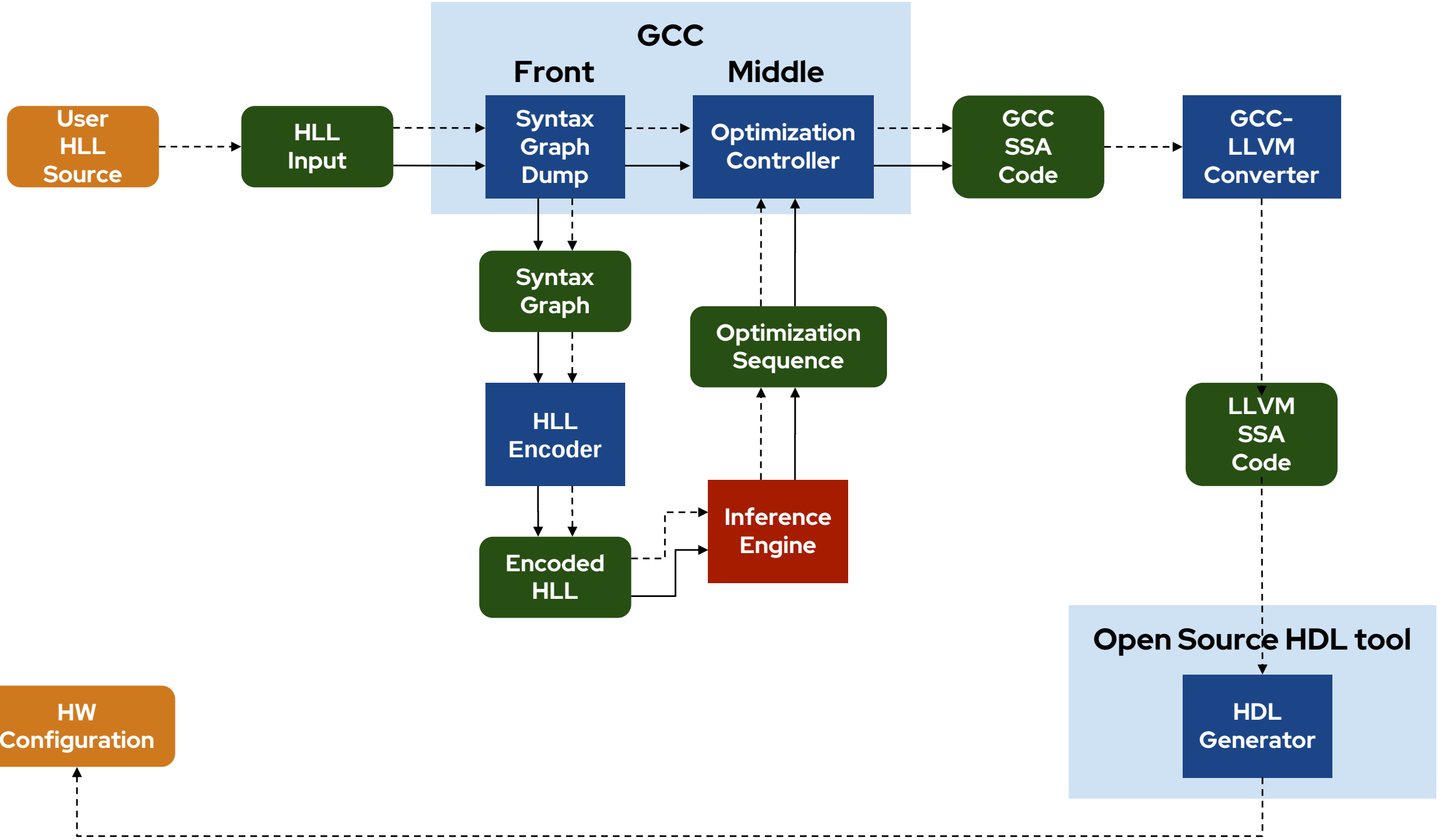
v1


GCC
Front
Middle
HLL Input
Syntax Graph
Optimization Controller
GCC SSA Code
Optimization Sequence

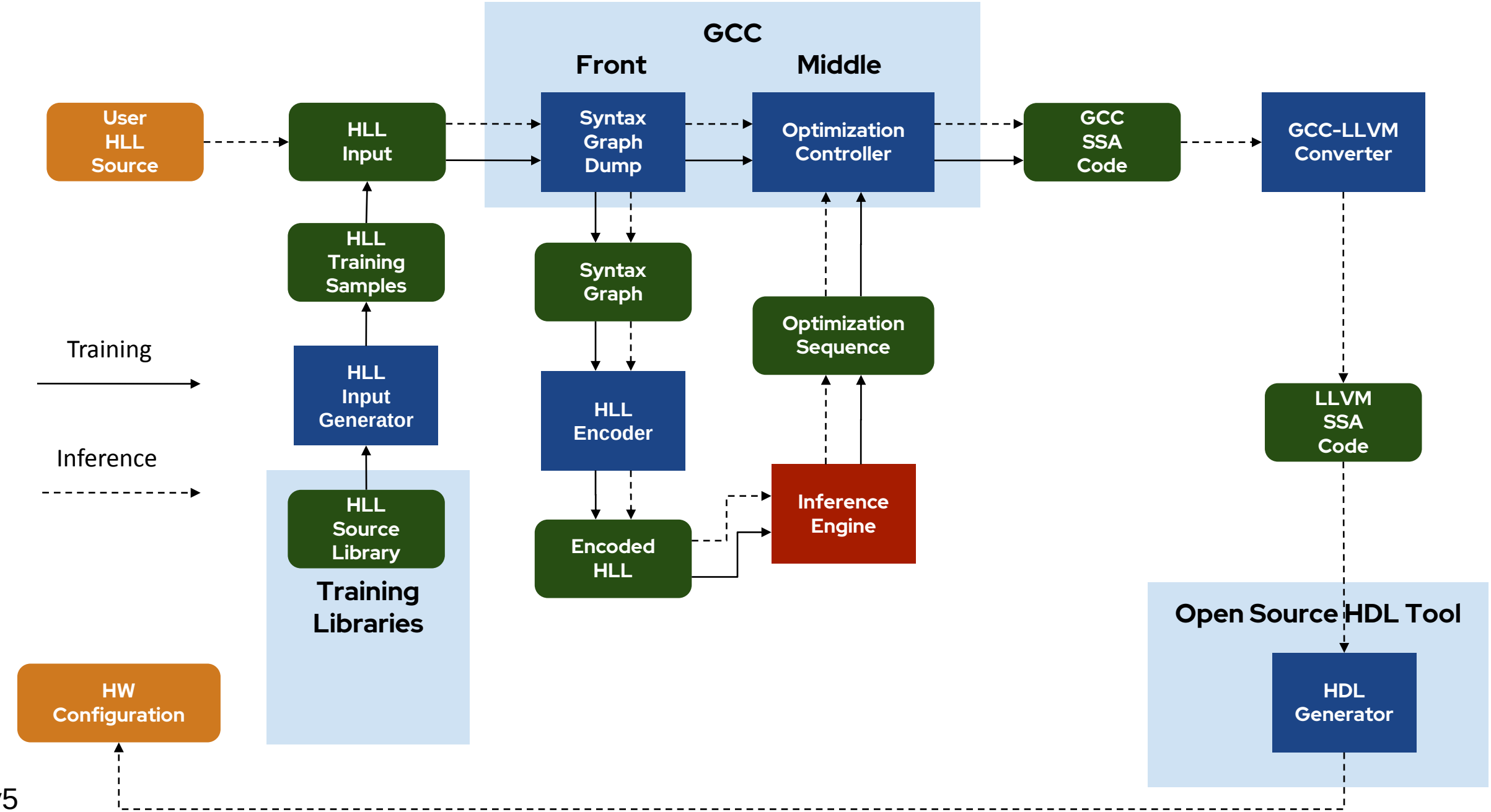
v2


GCC
Front
Middle
HLL Input
Syntax Graph Dump
Optimization Controller
GCC SSA Code
Syntax Graph
Optimization Sequence
HLL Encoder
Inference Engine
Encoded HLL

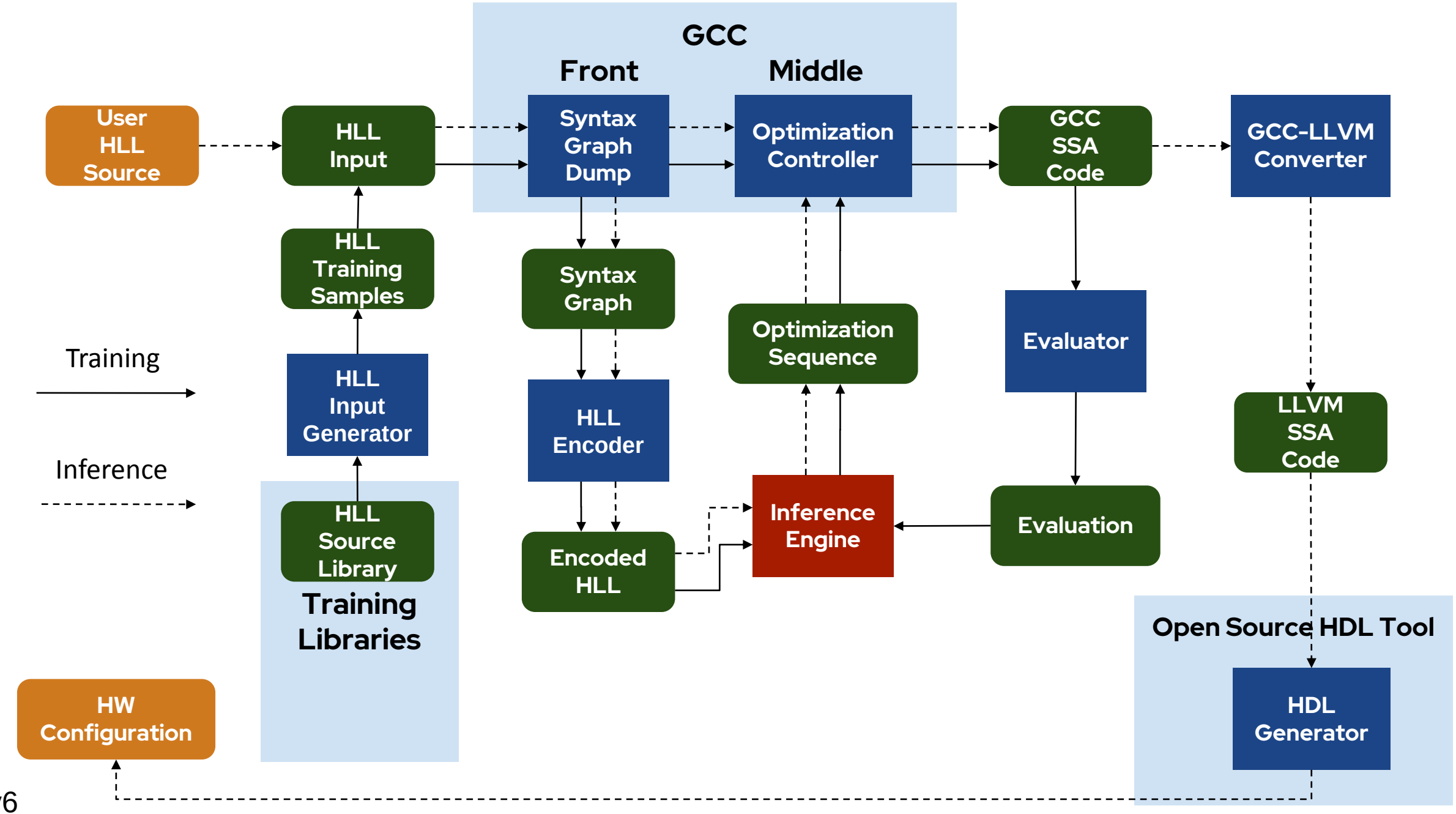
v3


GCC
Front
Middle
User HLL Source
HLL Input
Syntax Graph Dump
Optimization Controller
GCC SSA Code
GCC-LLVM Converter
Syntax Graph
Optimization Sequence
HLL Encoder
LLVM SSA Code
Inference Engine
Encoded HLL
Open Source HDL tool
HDL Generator
HW Configuration

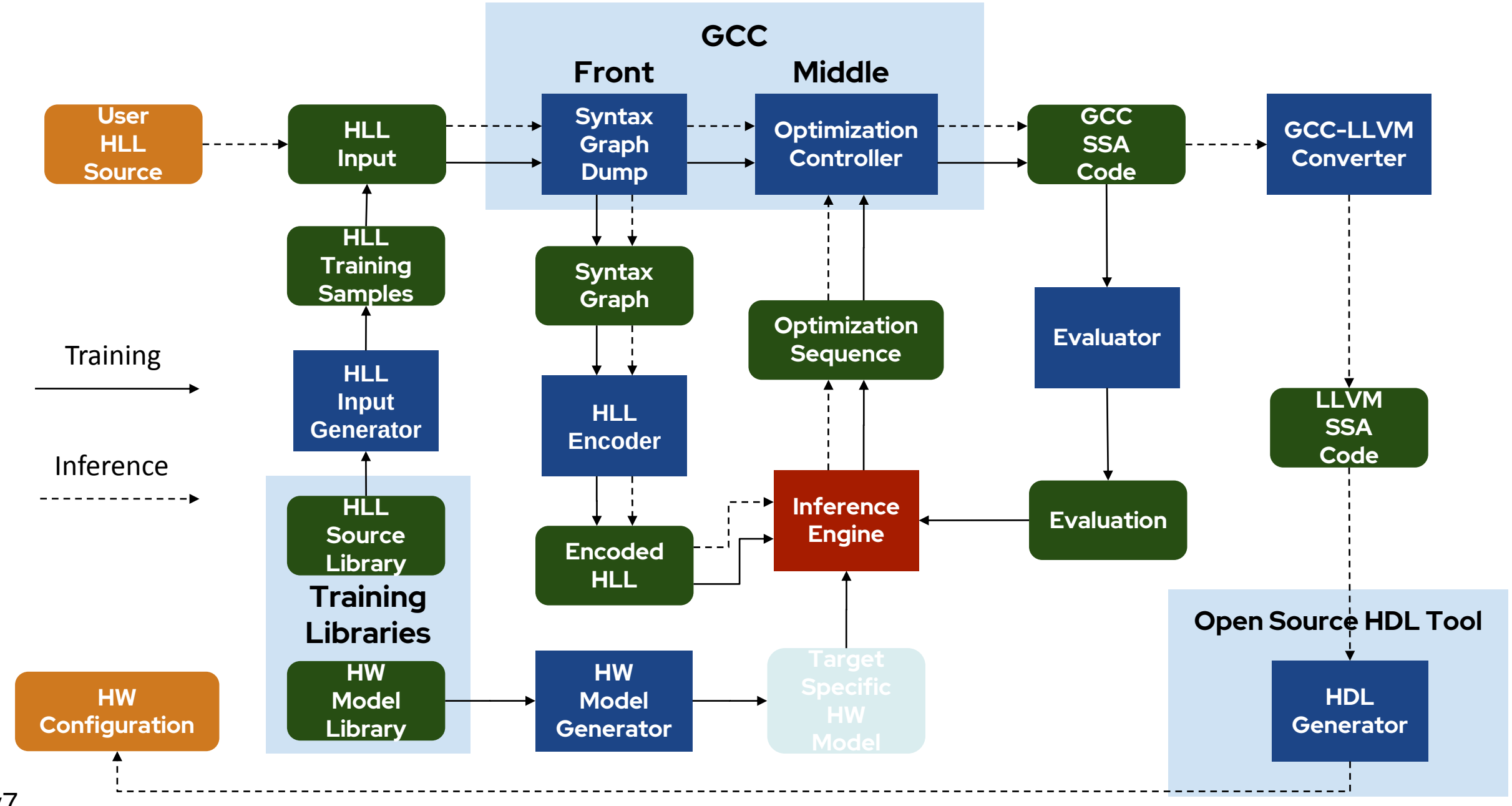
v4


GCC
Front
Middle
User HLL Source
HLL Input
Syntax Graph Dump
Optimization Controller
GCC SSA Code
GCC-LLVM Converter
HLL Training Samples
Syntax Graph
Optimization Sequence
Training
HLL Input Generator
HLL Encoder
LLVM SSA Code
Inference
HLL Source Library
Inference Engine
Encoded HLL
Training Libraries
Open Source HDL Tool
HDL Generator
HW Configuration

v5

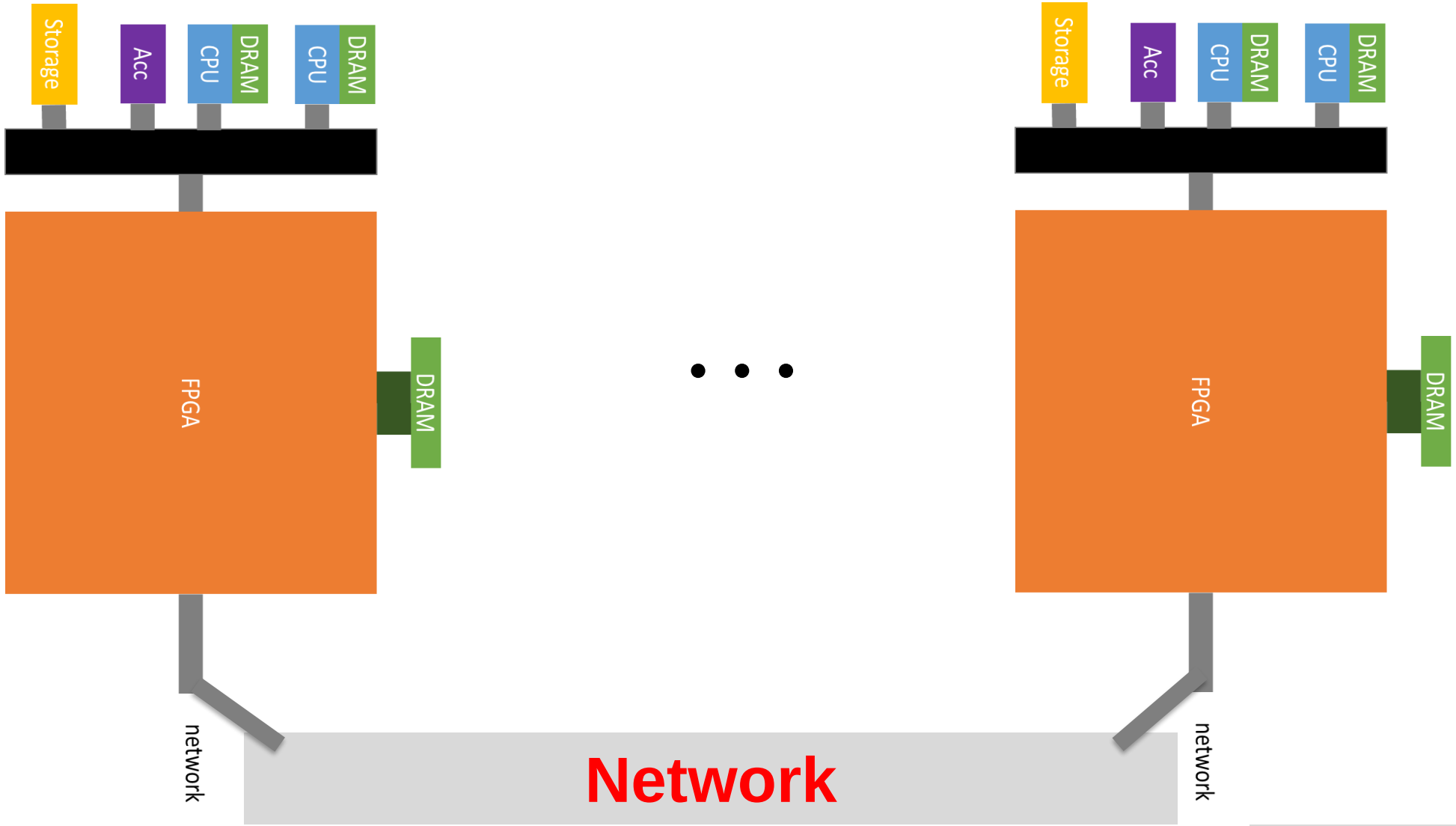

GCC
Front
Middle
User HLL Source
HLL Input
Syntax Graph Dump
Optimization Controller
GCC SSA Code
GCC-LLVM Converter
HLL Training Samples
Syntax Graph
Optimization Sequence
Evaluator
Training
HLL Input Generator
HLL Encoder
LLVM SSA Code
Inference
HLL Source Library
Inference Engine
Evaluation
Encoded HLL
Training Libraries
Open Source HDL Tool
HDL Generator
HW Configuration

v6


GCC
Front
Middle
User HLL Source
HLL Input
Syntax Graph Dump
Optimization Controller
GCC SSA Code
GCC-LLVM Converter
HLL Training Samples
Syntax Graph
Optimization Sequence
Evaluator
Training
HLL Input Generator
HLL Encoder
LLVM SSA Code
Inference
HLL Source Library
Inference Engine
Evaluation
Training Libraries
Encoded HLL
Open Source HDL Tool
HW Model Library
HW Model Generator
Target Specific HW Model
HDL Generator
HW Configuration

v7

What FPGA-centric clouds and clusters look like*



*Based on a Microsoft figure