LATTE '21

From DSLs to Accelerator-Rich Platform Implementations: Addressing the Mapping Gap

Presenters: Steven Lyubomirsky*, Thierry Tambe*

Bo-Yuan Huang* Yi Li Mike He Gus Smith

Gu-Yeon Wei Aarti Gupta Sharad Malik Zachary Tatlock

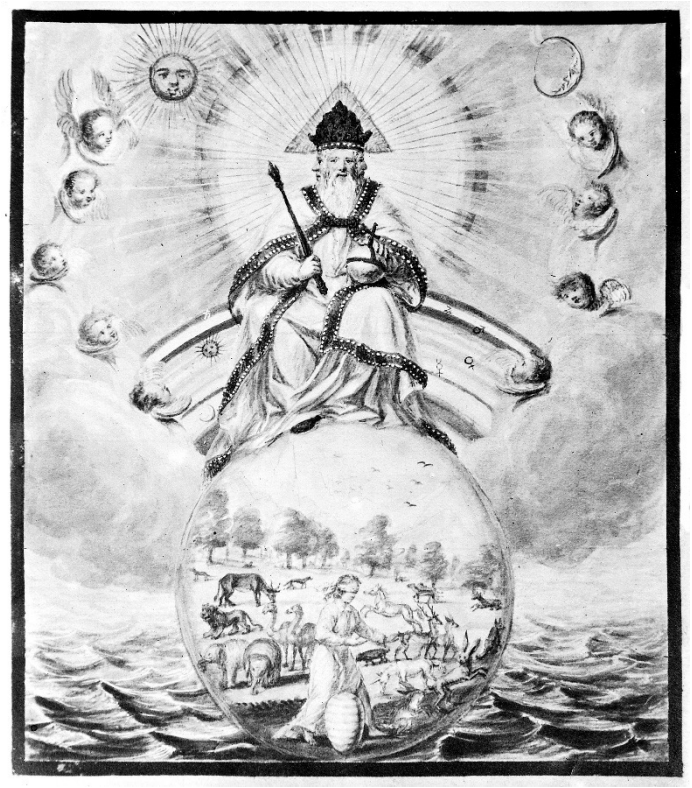
*Equal contribution

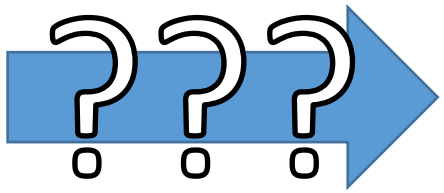
Harvard John A. Paulson School of Engineering and Applied Sciences

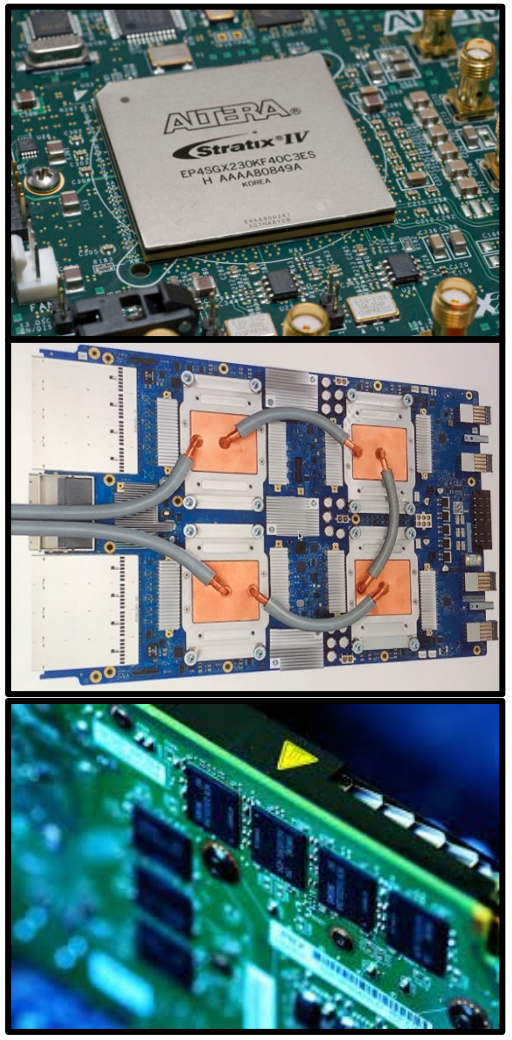
PRINCETON School of Engineering and Applied Science

PAUL G. ALLEN SCHOOL OF COMPUTER SCIENCE & ENGINEERING

The Mapping Gap



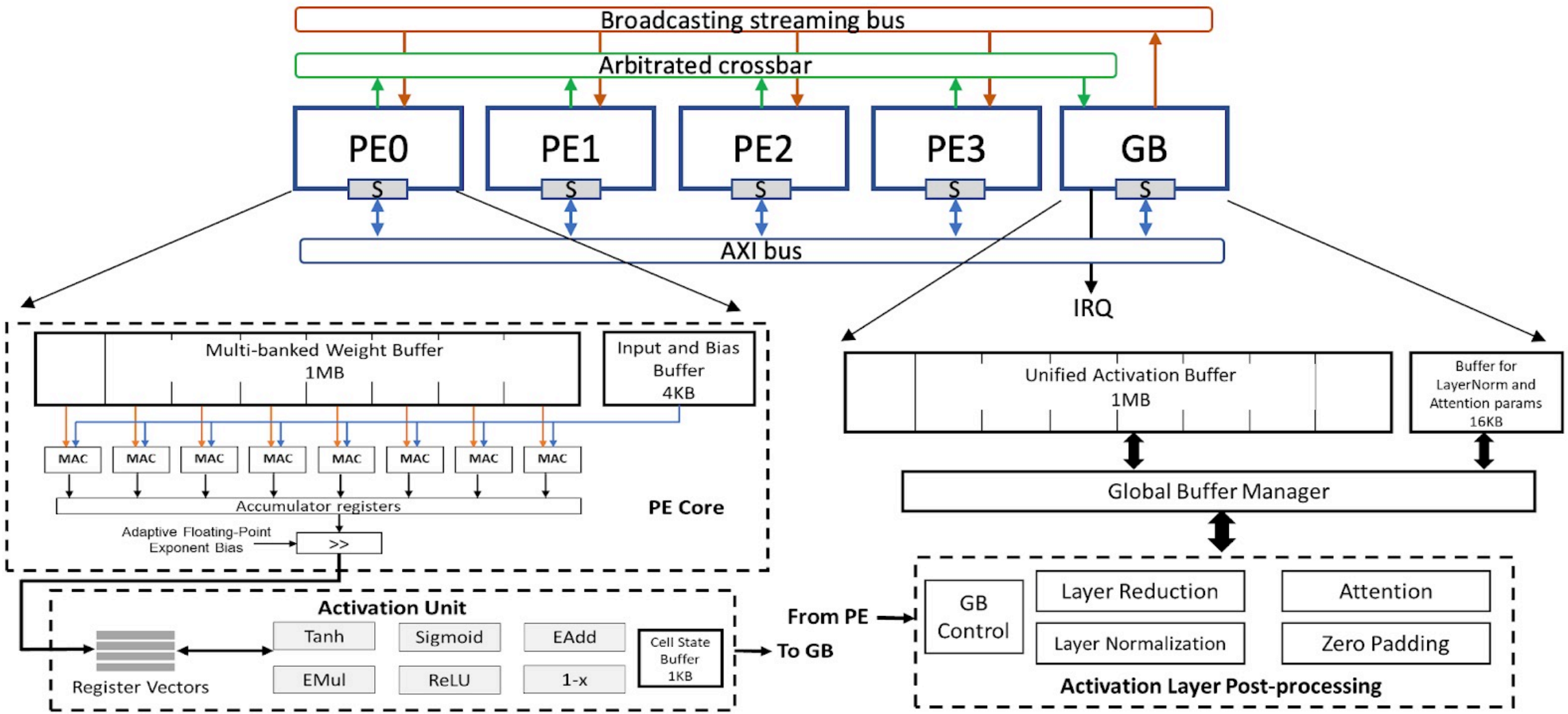




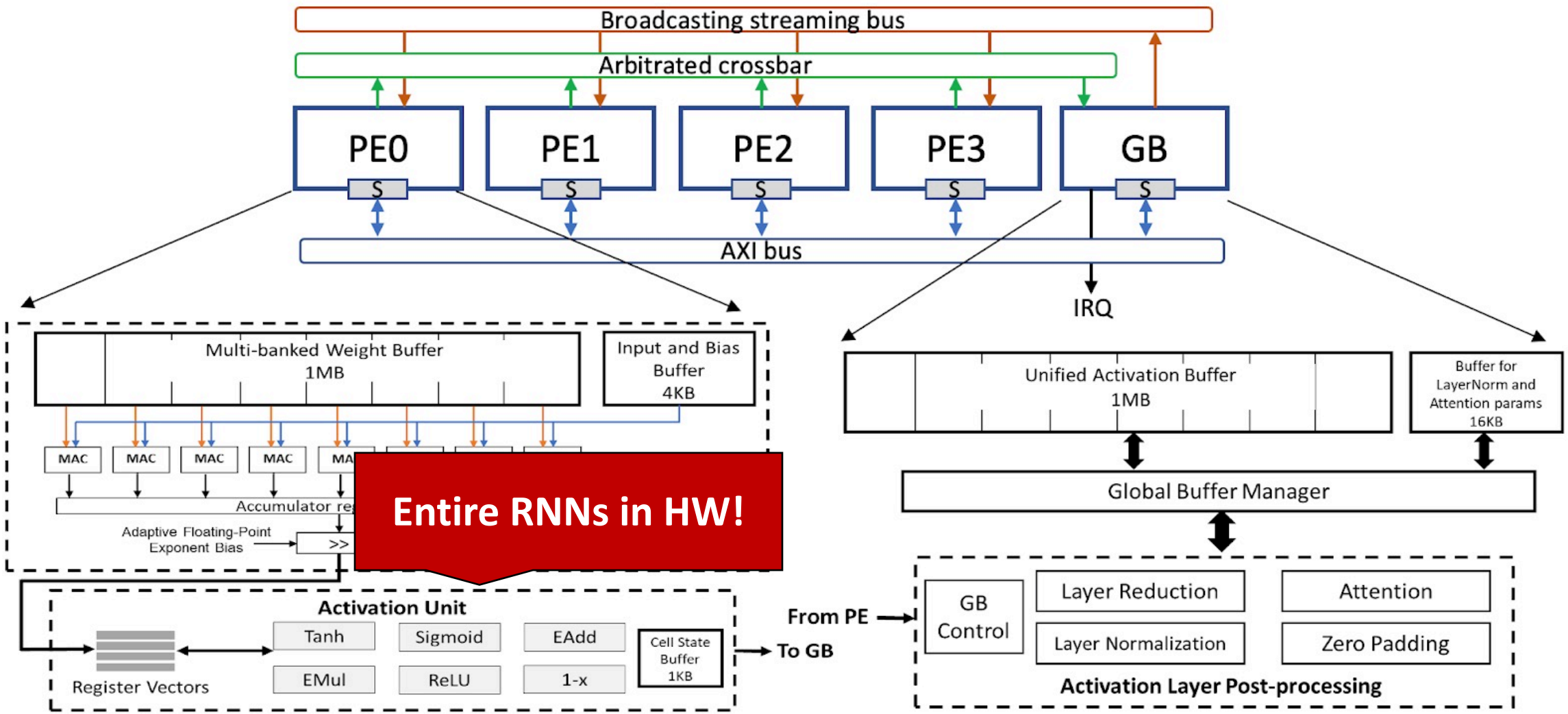
High-level application logic

Varied, low-level interfaces

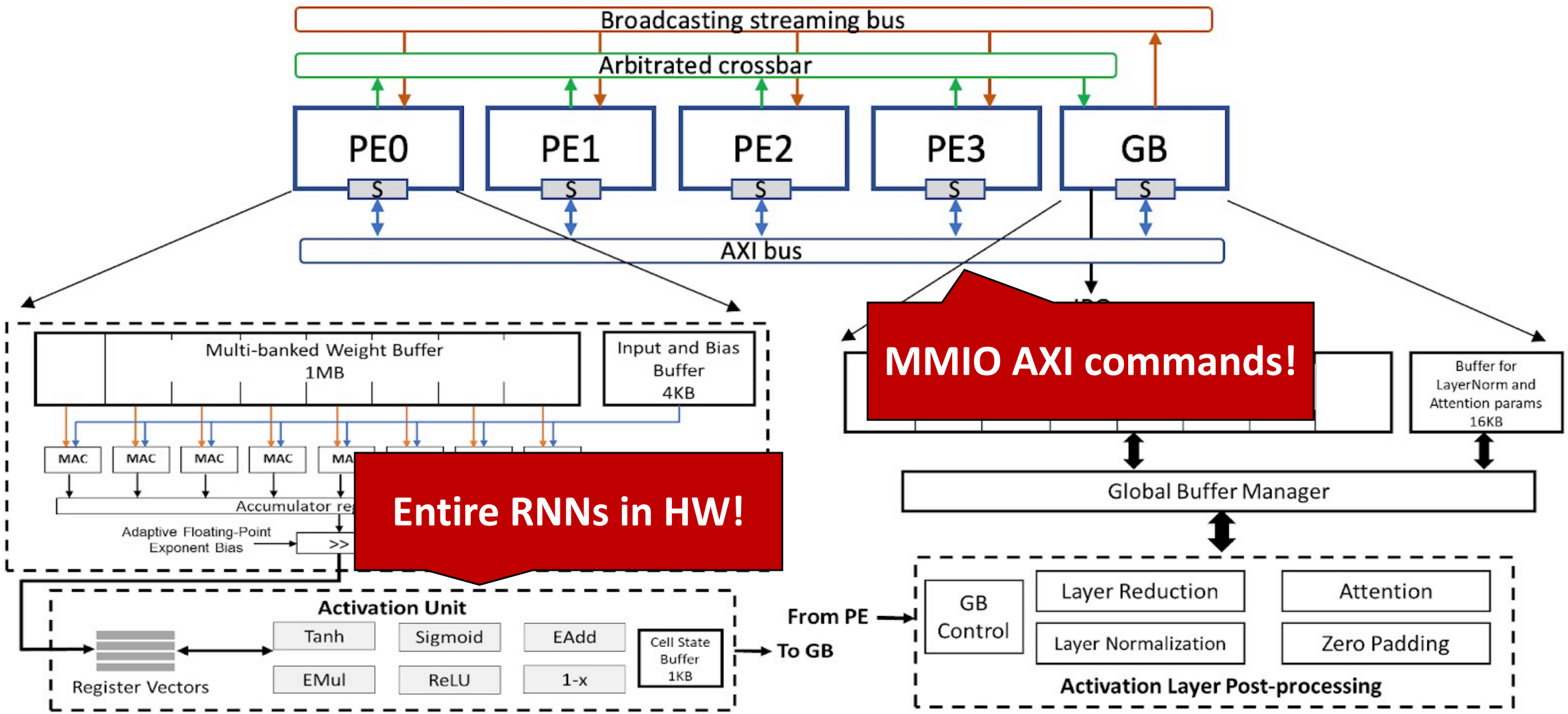
FlexNLP (Tambe *et al.*, 2020)



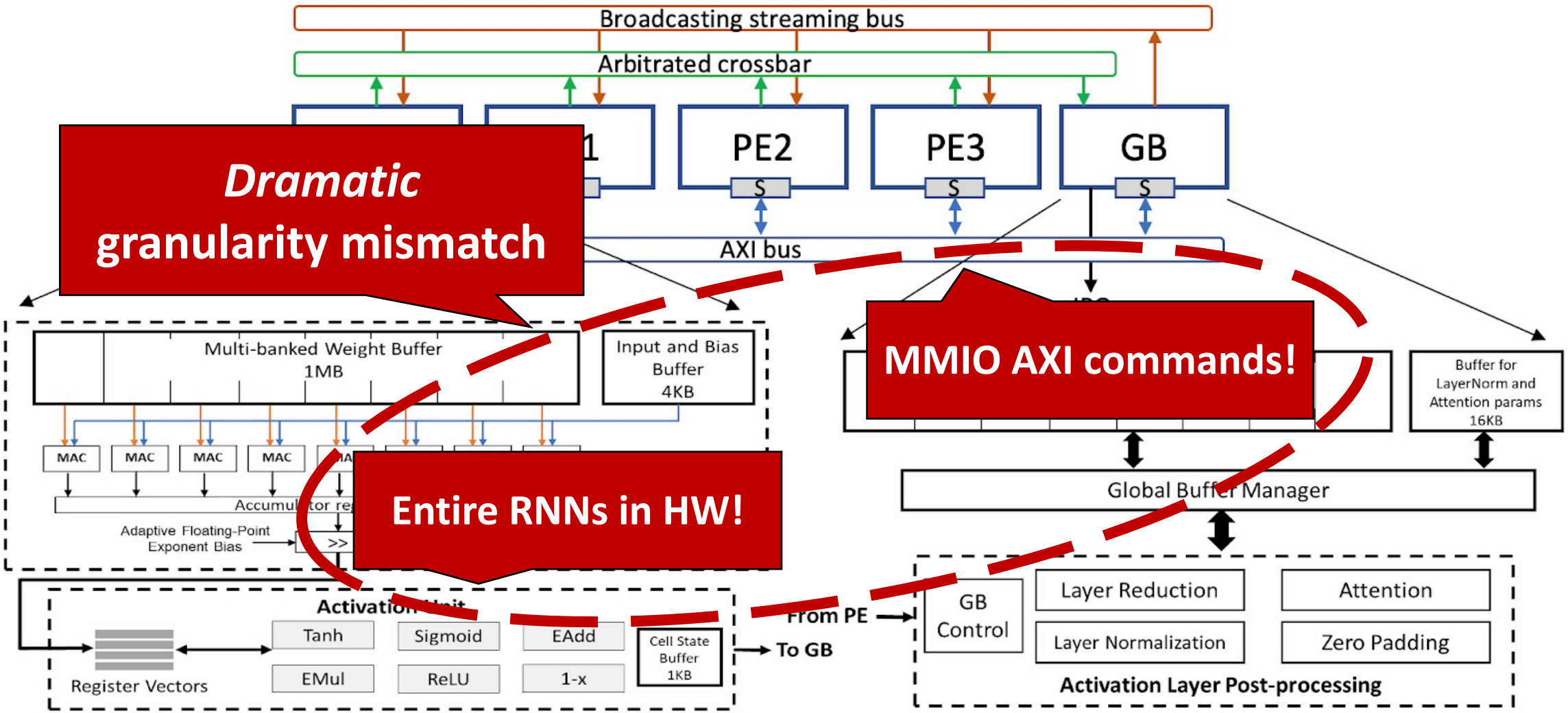
FlexNLP (Tambe *et al.*, 2020)



FlexNLP (Tambe *et al.*, 2020)



FlexNLP (Tambe *et al.*, 2020)



Compiler Support Challenges

- Lack of a unified hardware specification
  - No ISA!
  - Interfaces via MMIO commands
- Granularity mismatch
  - Source language is fine-grained
  - Accelerators operations are high-level
- Custom memory and numerical formats

Instruction-Level Abstraction (Huang *et al.*, 2019)

MMIO Access	Instruction
WR 0x33000010	GB_CONTROL_START
WR 0x33000050	GB_ATTENTION_START
WR 0x33700010	CONFIG_GB_CONTROL
WR 0x33700050	CONFIG_GB_ATTENTION
...	...

Architectural state variable:

$(cfgMode, cfgIsRnn, cfgNumTimestep, valid, memL, \ldots)$

Instruction CONFIG_GB_CONTROL:

$\delta_i \stackrel{\text{def}}{=} (inWr \wedge \neg inRd \wedge inAddr = 0x33700010)$

$N_i[cfgMode] \stackrel{\text{def}}{=} ITE(valid, inData[1:3], cfgMode)$

$N_i[cfgIsRnn] \stackrel{\text{def}}{=} ITE(valid, inData[4:4], cfgIsRnn)$

$\ldots$

Instruction-Level Abstraction (Huang *et al.*, 2019)

Formally specified interface!
Just like an ISA!

WR 0x33000050	GB_ATTENTION_START
WR 0x33700010	CONFIG_GB_CONTROL
WR 0x33700050	CONFIG_GB_ATTENTION
...	...

Architectural state variable:

$(cfgMode, cfgIsRnn, cfgNumTimestep, valid, memL, \ldots)$

Instruction CONFIG_GB_CONTROL:

$\delta_i \stackrel{\text{def}}{=} (inWr \wedge \neg inRd \wedge inAddr = 0x33700010)$

$N_i[cfgMode] \stackrel{\text{def}}{=} ITE(valid, inData[1:3], cfgMode)$

$N_i[cfgIsRnn] \stackrel{\text{def}}{=} ITE(valid, inData[4:4], cfgIsRnn)$

$\ldots$

Instruction-Level Abstraction (Huang *et al.*, 2019)

Easy lowering to MMIO commands!

MMIO Access	Instruction
WR 0x33000010	GB_CONTROL_START
WR 0x33000050	GB_ATTENTION_START
WR 0x33700010	CONFIG_GB_CONTROL
WR 0x33700050	CONFIG_GB_ATTENTION
...	...

$(cfgMode, cfgIsRnn, cfgNumTimestep, valid, memL, \ldots)$

Instruction CONFIG_GB_CONTROL:

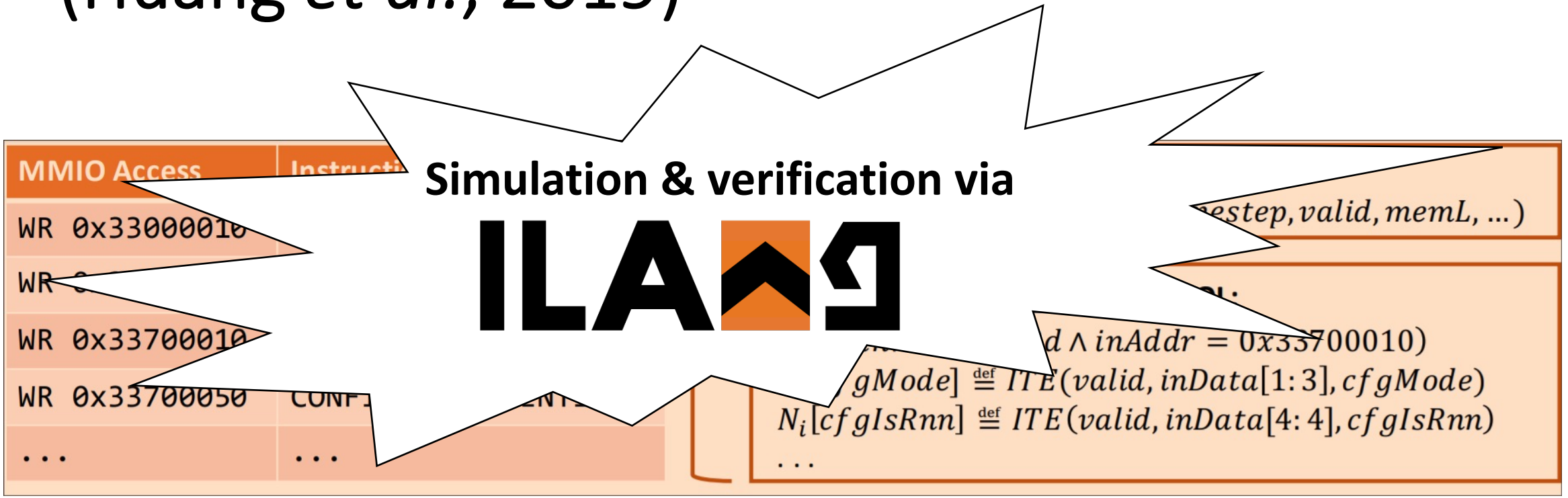
$\delta_i \stackrel{\text{def}}{=} (inWr \wedge \neg inRd \wedge inAddr = 0x33700010)$

$N_i[cfgMode] \stackrel{\text{def}}{=} ITE(valid, inData[1:3], cfgMode)$

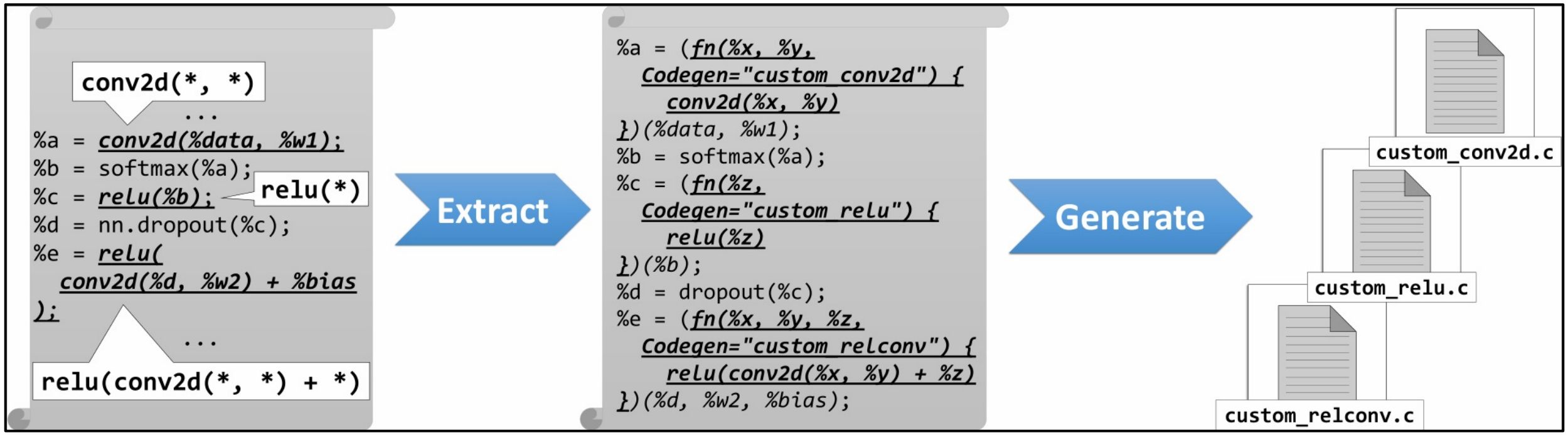
$N_i[cfgIsRnn] \stackrel{\text{def}}{=} ITE(valid, inData[4:4], cfgIsRnn)$

...

Instruction-Level Abstraction (Huang *et al.*, 2019)



TVM Bring Your Own Codegen

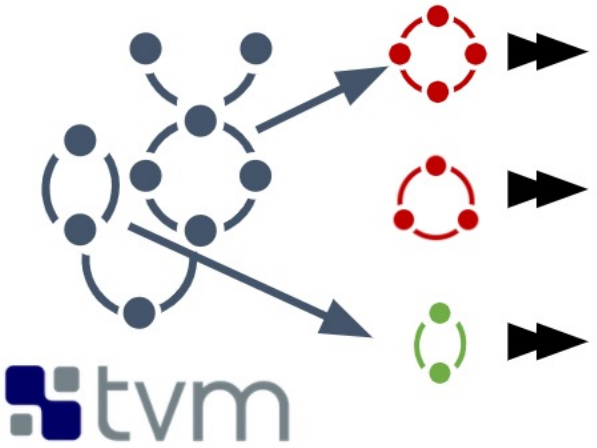


Syntactic pattern matching is key!

3LA: Codegen via ILA

Import from frameworks to TVM

Match Relay fragments



Map Relay fragments to ILA instructions

```
FlexNLP func A:
instr. 1 (STR r2, 0xffff0000)
instr. 2 (LDR r3, 0xffff0010)

FlexNLP func B:
instr. 3 (STR r2, 0xffffaa00)
instr. 4 (LDR r3, 0xffffaabb)

NVDLA func C:
instr. 5 (STR r4, 0xffff0100)
instr. 6 (LDR r5, 0xffff0110)
```

ILAng

Lower to accelerator interface

```
; CPU instructions
CMP   r0, r1
SUBGT r0, r0, r1
BNE   loop
; Access accel 1 (MMIO)
STR   r2, 0xffff0000
LDR   r3, 0xffff0010
; Access accel 2 (MMIO)
STR   r4, 0xffff0100
LDR   r5, 0xffff0110
; CPU instructions
MOV   r3, r2
SUBGT r0, r0, r1
B     lr
```

Heterogeneous hardware backends

intel

HLSCNN

NVDLA

VTA

FlexNLP

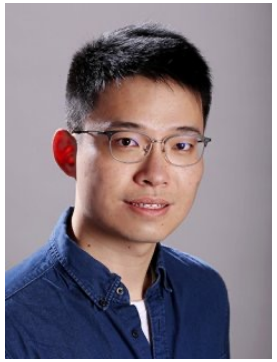
Further Possibilities

- End-to-end functional verification
- Flexible matching (semantic reasoning)
- Automating compiler stack generation

Thank You!

Bo-Yuan Huang

Thierry Tambe



Yi Li

Mike He

Gus Smith

Gu-Yeon Wei

Aarti Gupta

Sharad Malik

Zachary Tatlock

Thank You!

JUMP

Joint University Microelectronics Program

www.src.org/program/jump

Semiconductor Research Corporation

@srcJUMP

ada

Applications Driving Architectures

TVM (Chen *et al*., 2018)