ESE: Efficient Speech Recognition Engine for Sparse LSTM on FPGA

Song Han[1,2], Junlong Kang[2], Huizi Mao[1], Yiming Hu[3], Xin Li[2], Yubin Li[2], Dongliang Xie[2], Hong Luo[2], Song Yao[2], Yu Wang[2,3], Huazhong Yang[2,3] and Bill Dally[1,4]

Stanford University[1], DeePhi[2], Tsinghua University[3], NVIDIA[4]

Feb 23, 2017
FPGA'17, Monterey, CA

Recurrent Neural Networks and LSTM

speech recognition

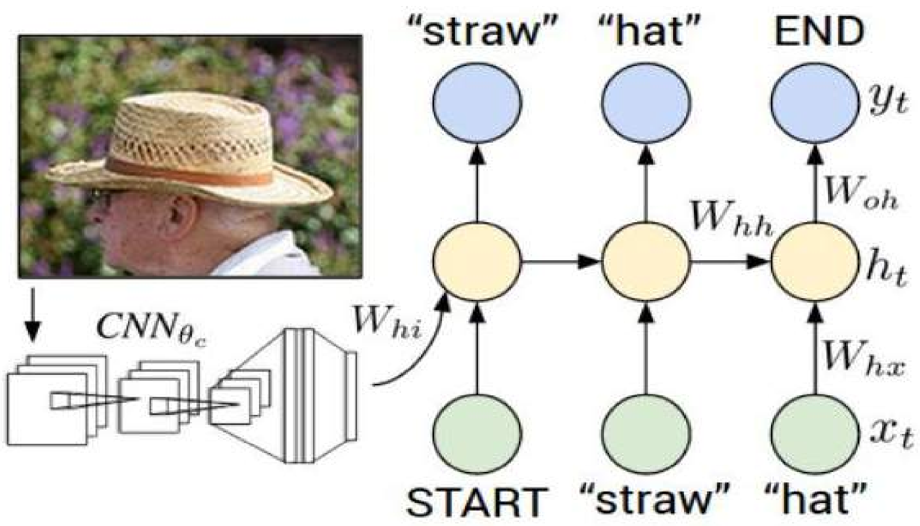
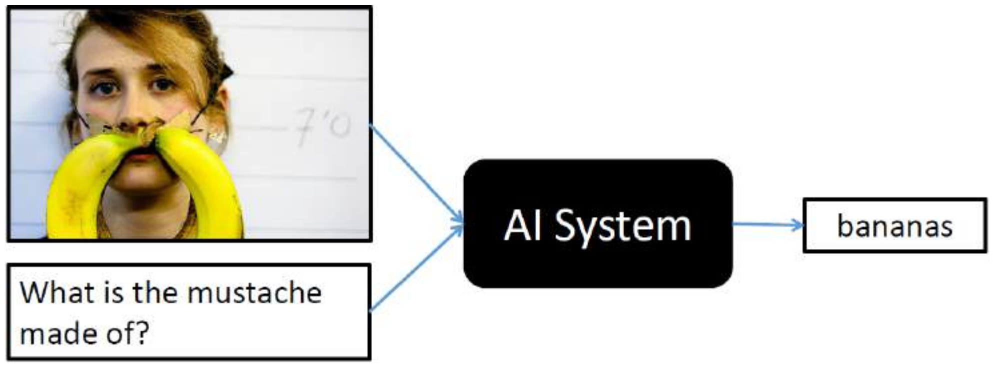


image caption

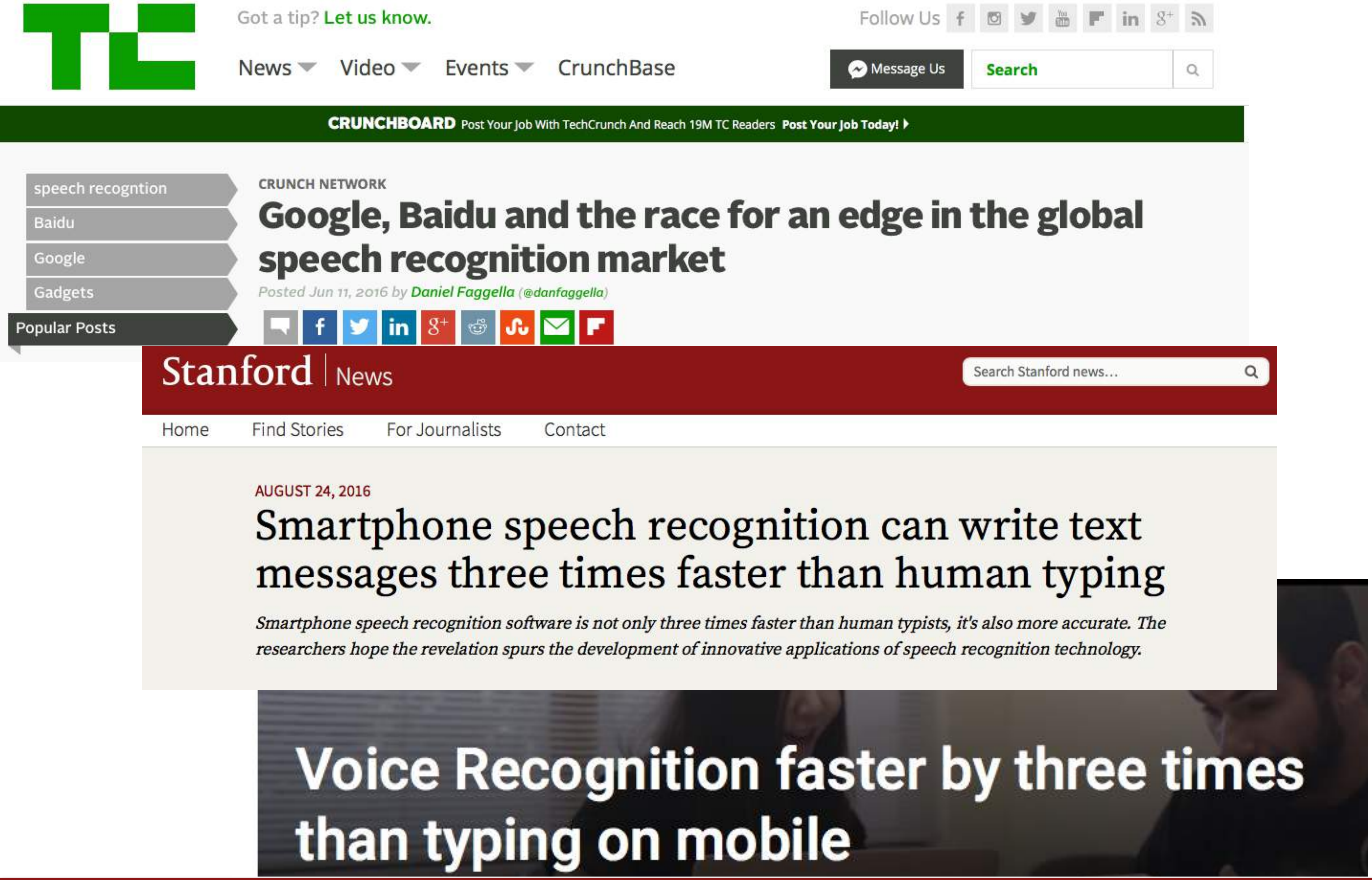
Google Translate

machine translation



visual question answering

Speech Recognition



Machine Translation

THE STACK NEWS | PARTNERS | EDUCATION | MAGAZINE | ABOUT US

WORLD

Google announces Neural Machine Translation

Nicky Cappella Wed 28 Sep 2016 11.52am

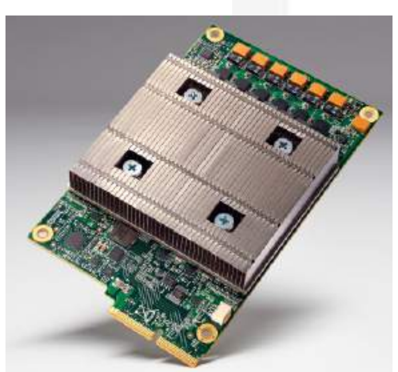
Search

CRUNCHBOARD Post Your Job With TechCrunch And Reach 19M TC Readers **Post Your Job Today!**

- Google
- neural networks
- artificial intelligence
- machine learning
- Artificial Intelligence

Google unleashes deep learning tech on language with Neural Machine Translation

Posted Sep 27, 2016 by Devin Coldewey, Contributor



The Tensor Processing Unit

Google is using Neural Networks for Chinese to English machine translation

Google Translate

Image Caption

DII	A group of people that are sitting next to each other.	Adult male wearing sunglasses lying down on black pavement.	The sun is setting over the ocean and mountains.
SIS	Having a good time bonding and talking.	[M] got exhausted by the heat.	Sky illuminated with a brilliance of gold and orange hues.

Figure 1: Example language difference between descriptions for images in isolation (DII) vs. stories for images in sequence (SIS).

Huang et al. “Visual Storytelling”

DII	A black frisbee is sitting on top of a roof.	A man playing soccer outside of a white house with a red door.	The boy is throwing a soccer ball by the red door.	A soccer ball is over a roof by a frisbee in a rain gutter.	Two balls and a frisbee are on top of a roof.
DIS	A roof top with a black frisbee laying on the top of the edge of it.	A man is standing in the grass in front of the house kicking a soccer ball.	A man is in the front of the house throwing a soccer ball up	A blue and white soccer ball and black Frisbee are on the edge of the roof top.	Two soccer balls and a Frisbee are sitting on top of the roof top.
SIS	A discus got stuck up on the roof.	Why not try getting it down with a soccer ball?	Up the soccer ball goes.	It didn't work so we tried a volley ball.	Now the discus, soccer ball, and volleyball are all stuck on the roof.

Figure 4: Example descriptions of images in isolation (DII); descriptions of images in sequence (DIS); and stories of images in sequence (SIS).

VQA: Visual Question Answering

http://vqa.daylen.com

which country is the flag of?

usa (0.72)
united states (0.13)
america (0.03)
canada (0.02)
us (0.02)

Took 0.267 sec

what is behind him?

flag (0.42)
curtain (0.32)
flags (0.22)
curtains (0.01)
chair (0.01)

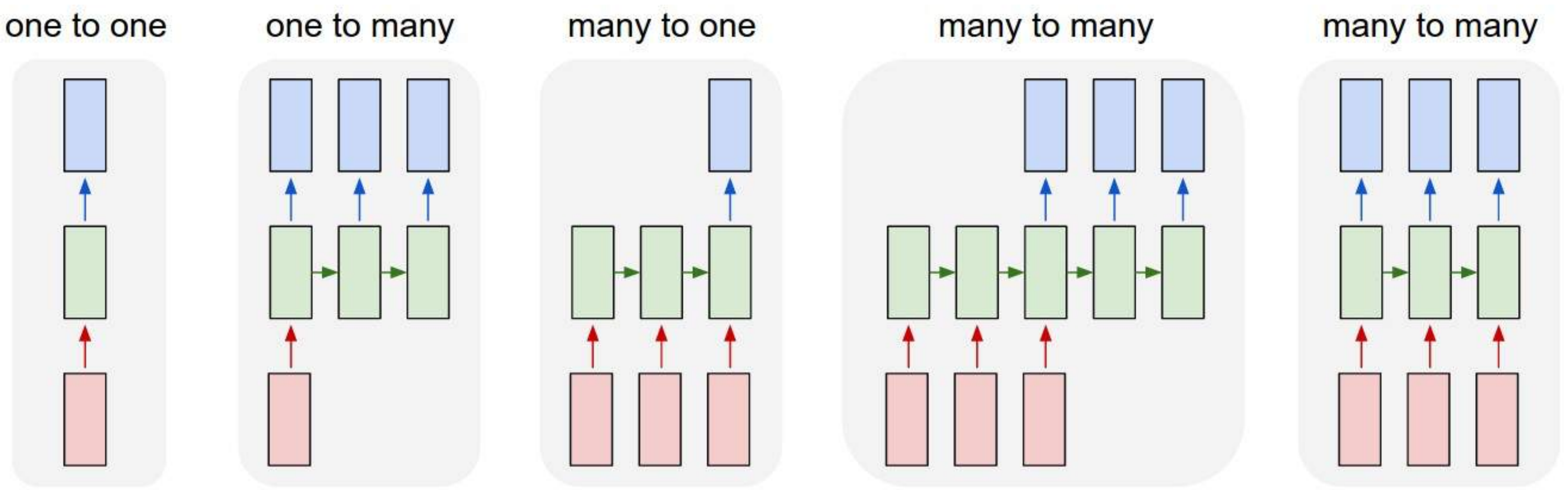
Took 0.266 sec

what is the color of his hair?

blonde (0.63)
gray (0.18)
red (0.09)
brown (0.08)
white (0.01)

Took 0.265 sec

Recurrent Neural Network



MLP

image caption

sentiment analysis

machine translation

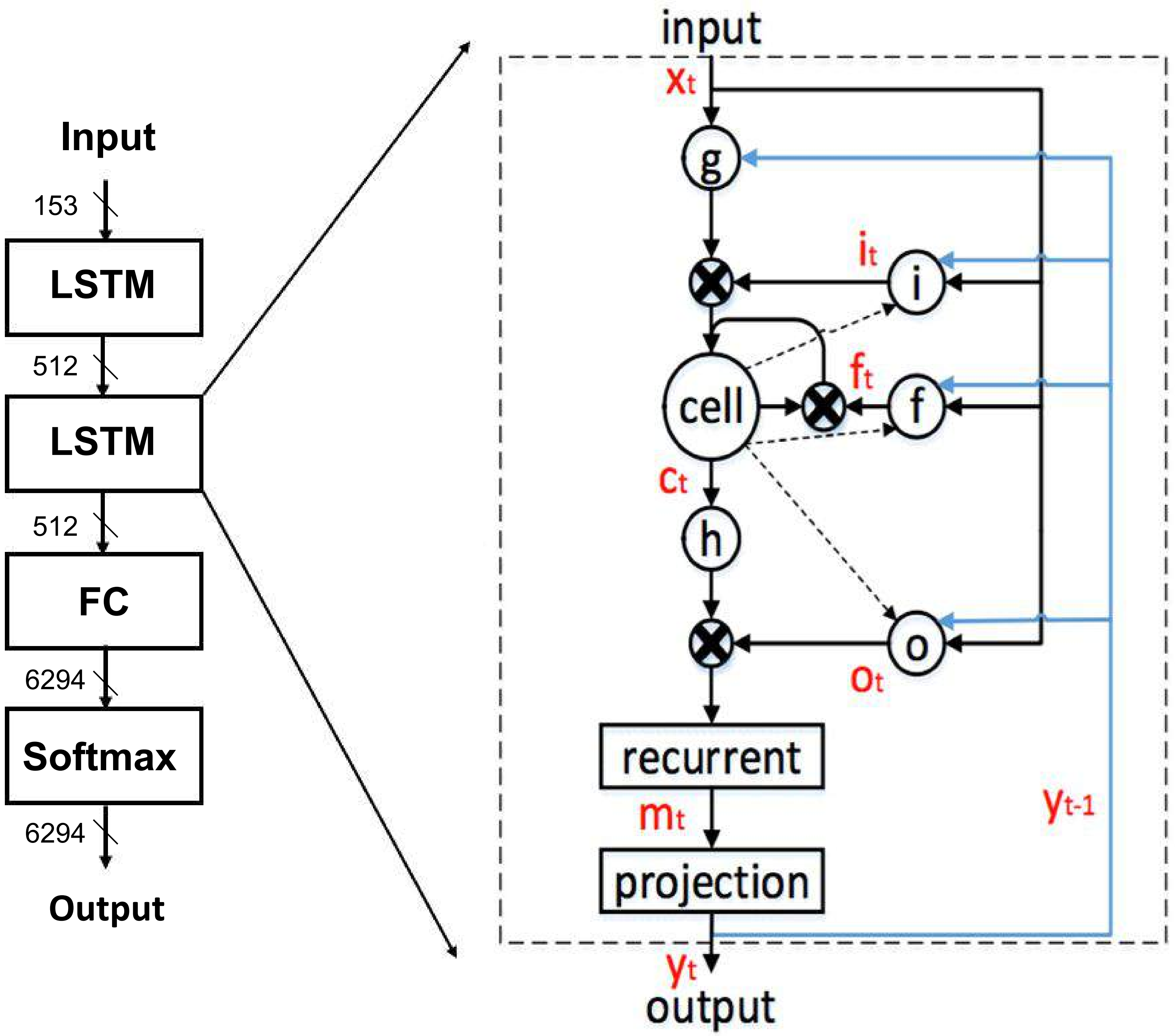
speech recognition

Stanford cs231n lecture notes

Comparing CNN / LSTM

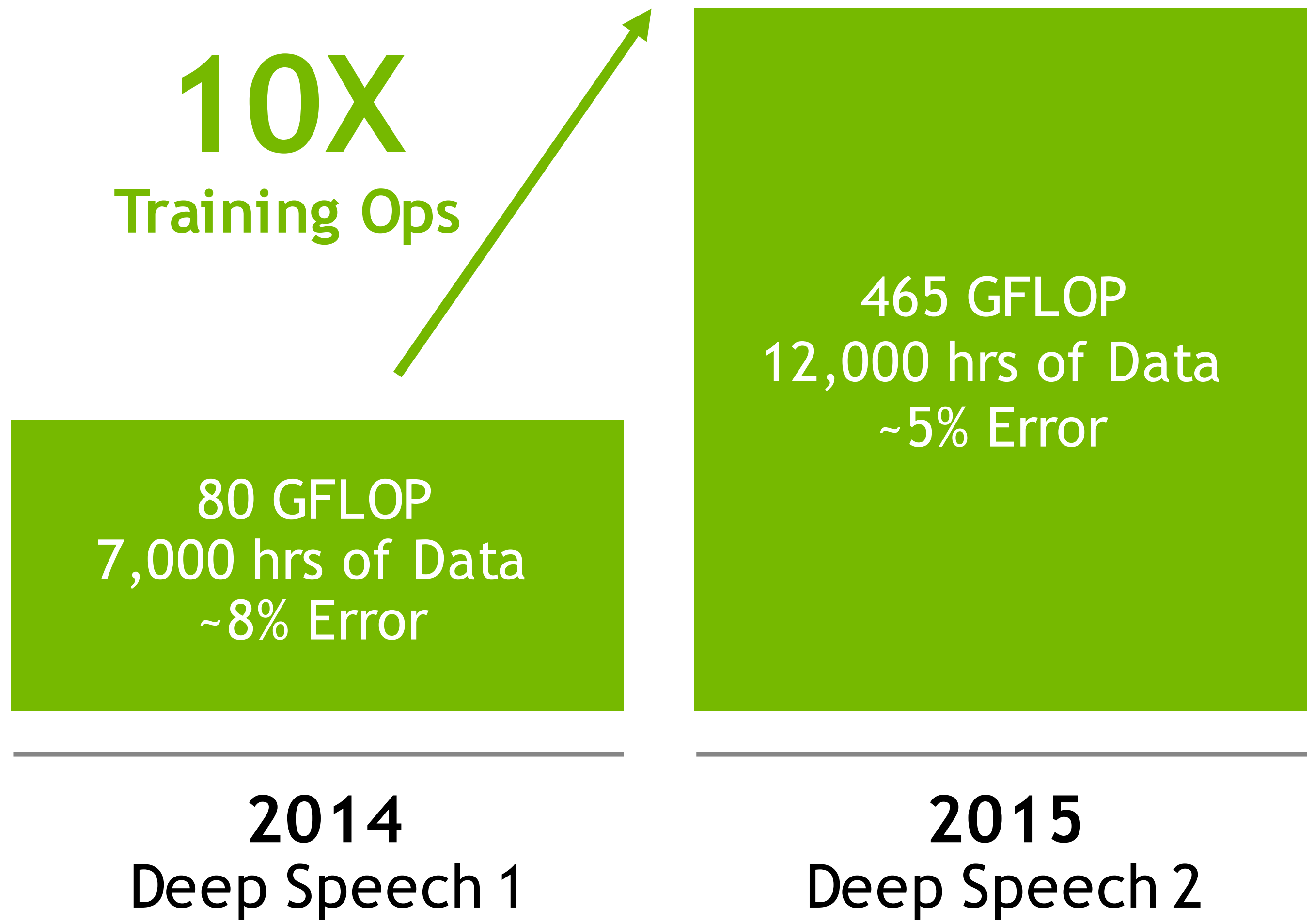
- CNN: weights shared in space
- RNN/LSTM: weights shared in time
- => Produces complicated <u>data dependency</u>
- => Making parallelization difficult

LSTM Structure



Models are Getting Larger

SPEECH RECOGNITION

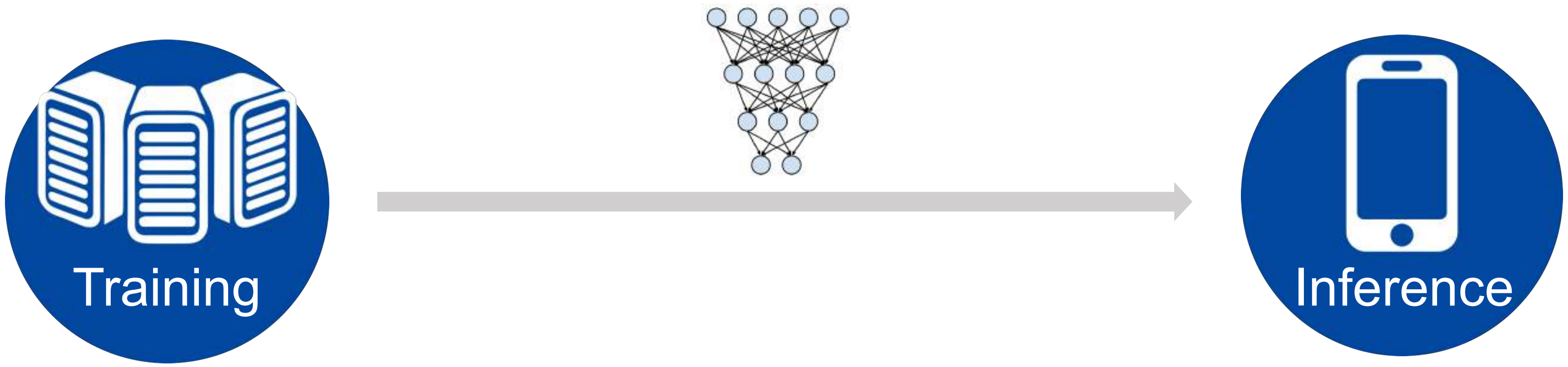


We Need more Computation

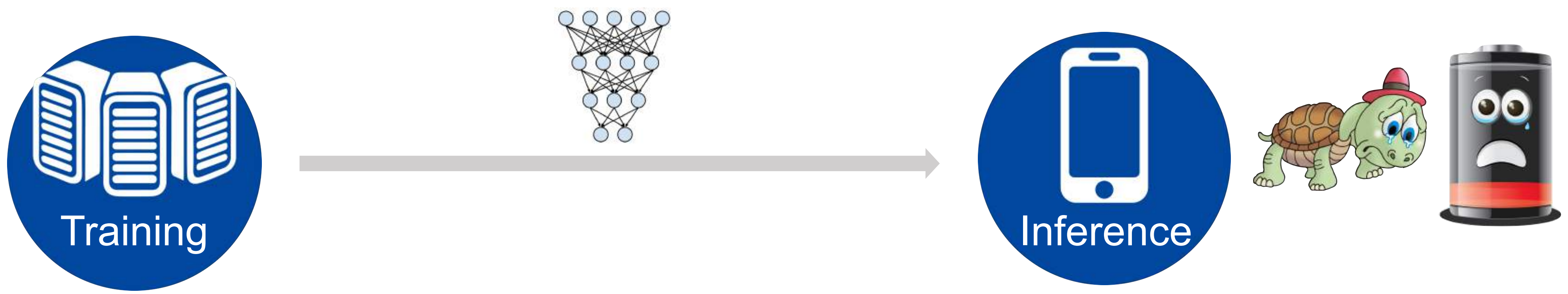
But Moore's law is no longer providing more compute…

Improve the Efficiency of Deep Learning by Algorithm-Hardware Co-Design

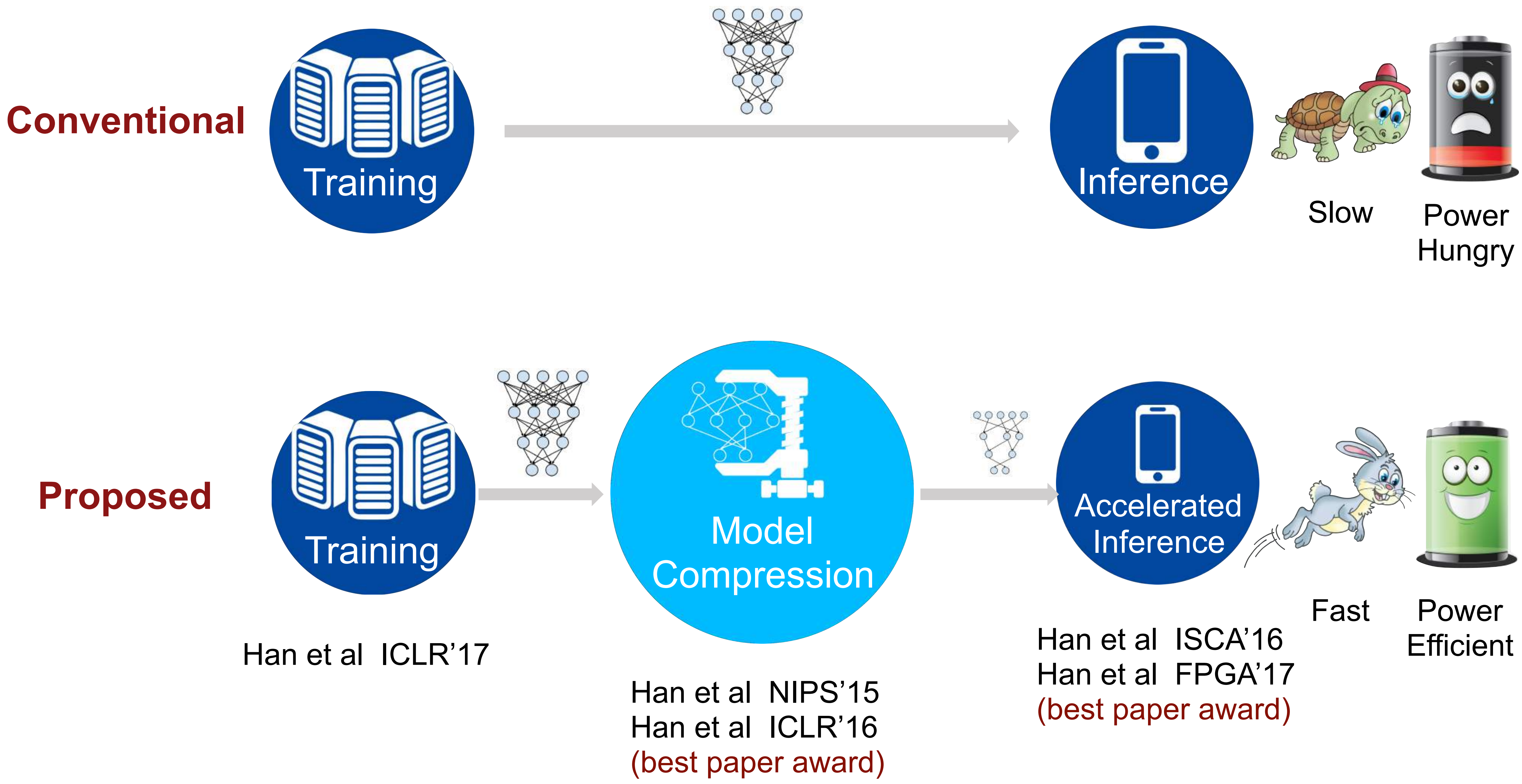
Conventional Paradigm



Conventional Paradigm

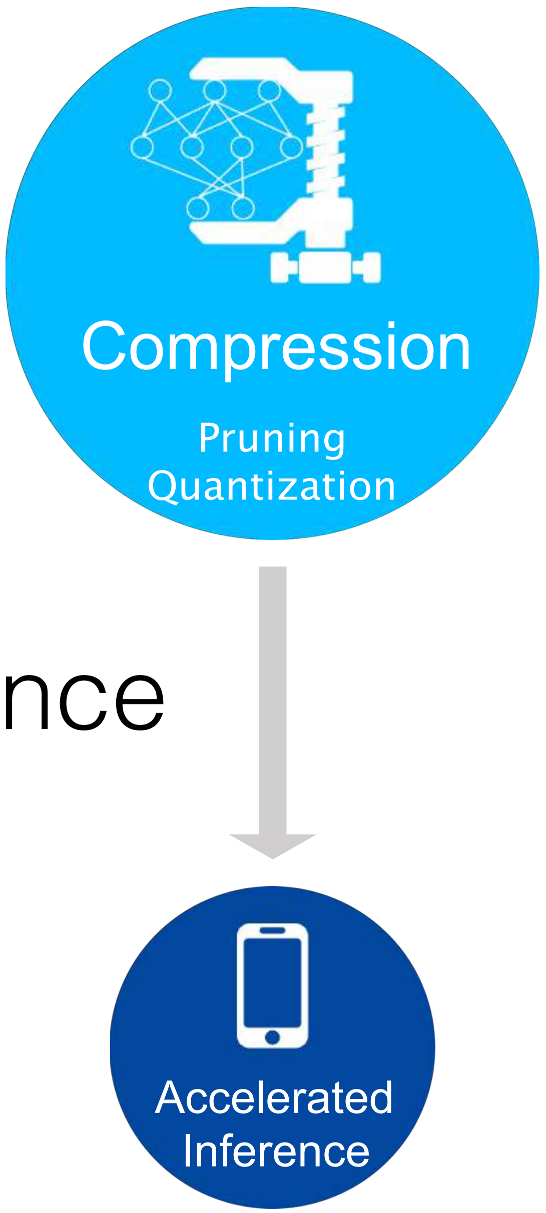


Proposed Paradigm



Agenda

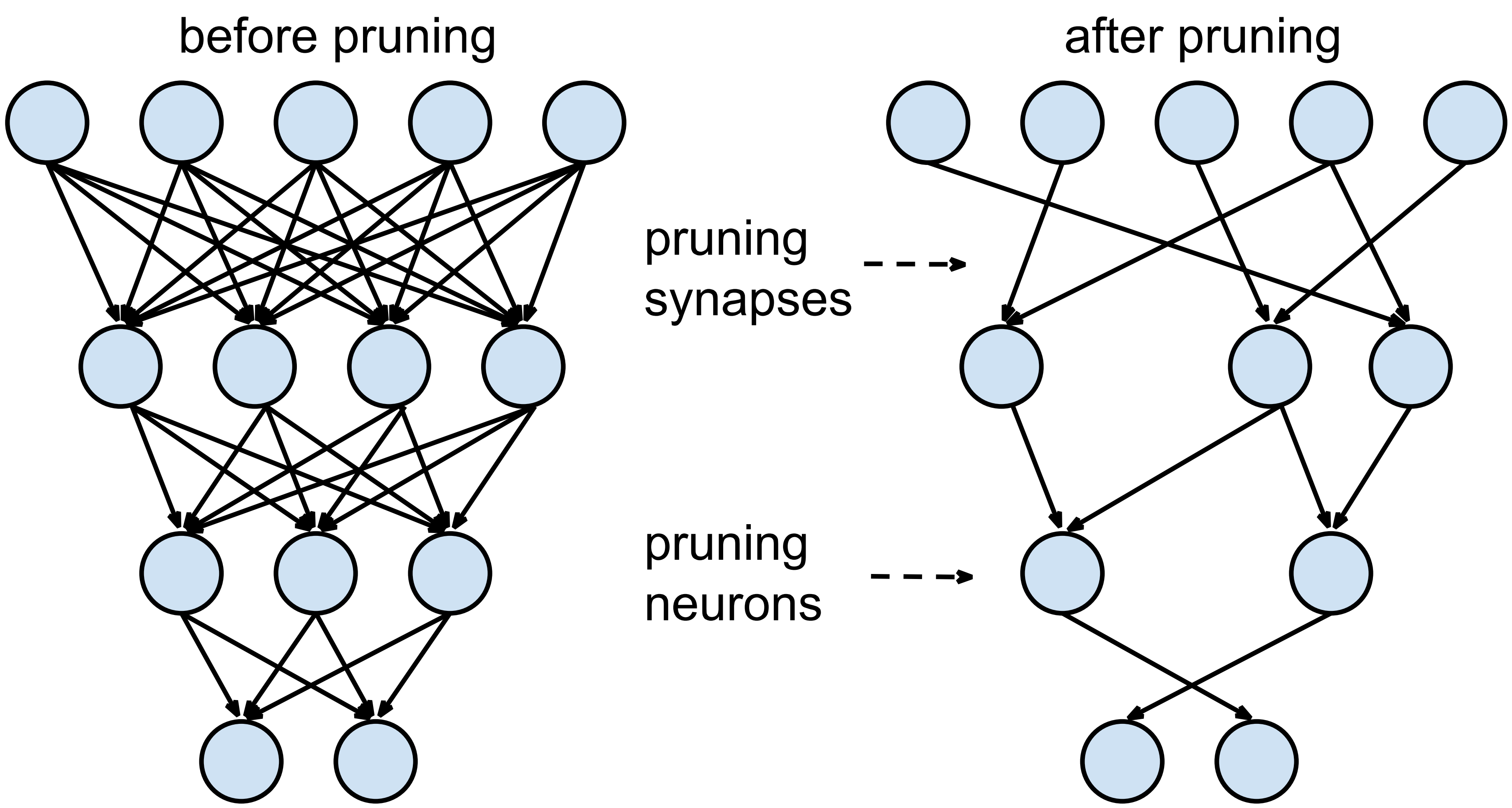
- **Compression**
  Load Balance-Aware Pruning
- **Scheduling**
  Overlap Computation and Memory Reference
- **Accelerated Inference**
  Efficient Architecture for Sparse LSTM
- **Results**



Agenda

- **Compression**
  Load Balance-Aware Pruning
- **Scheduling**
  Overlap Computation and Memory Reference
- **Accelerated Inference**
  Efficient Architecture for Sparse LSTM
- **Results**

Pruning Review



Han et al. Learning both Weights and Connections for Efficient Neural Networks, NIPS'15

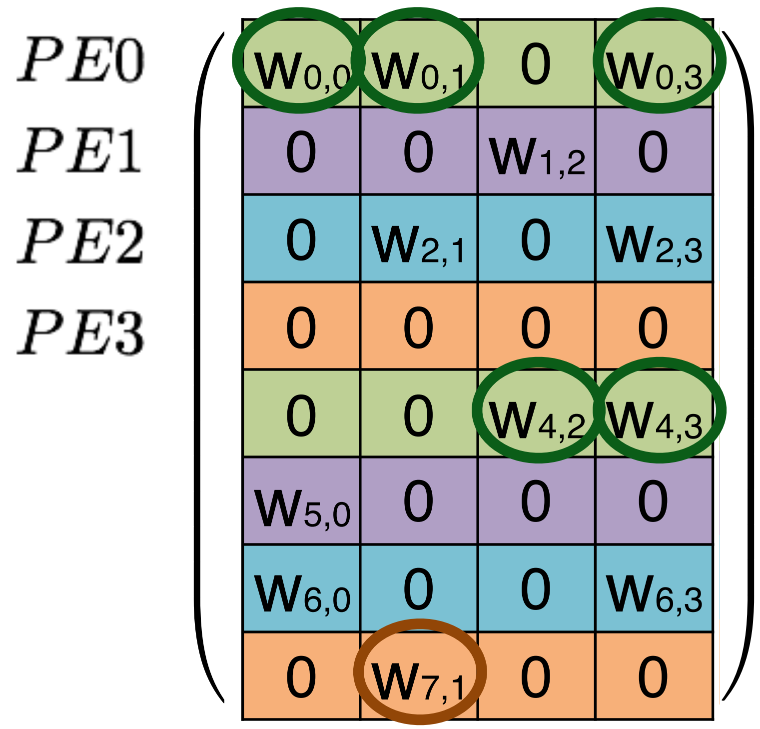
Pruning Lead to Load Imbalance

$PE0$	$w_{0,0}$	$w_{0,1}$	0	$w_{0,3}$
$PE1$	0	0	$w_{1,2}$	0
$PE2$	0	$w_{2,1}$	0	$w_{2,3}$
$PE3$	0	0	0	0
	0	0	$w_{4,2}$	$w_{4,3}$
	$w_{5,0}$	0	0	0
	$w_{6,0}$	0	0	$w_{6,3}$
	0	$w_{7,1}$	0	0

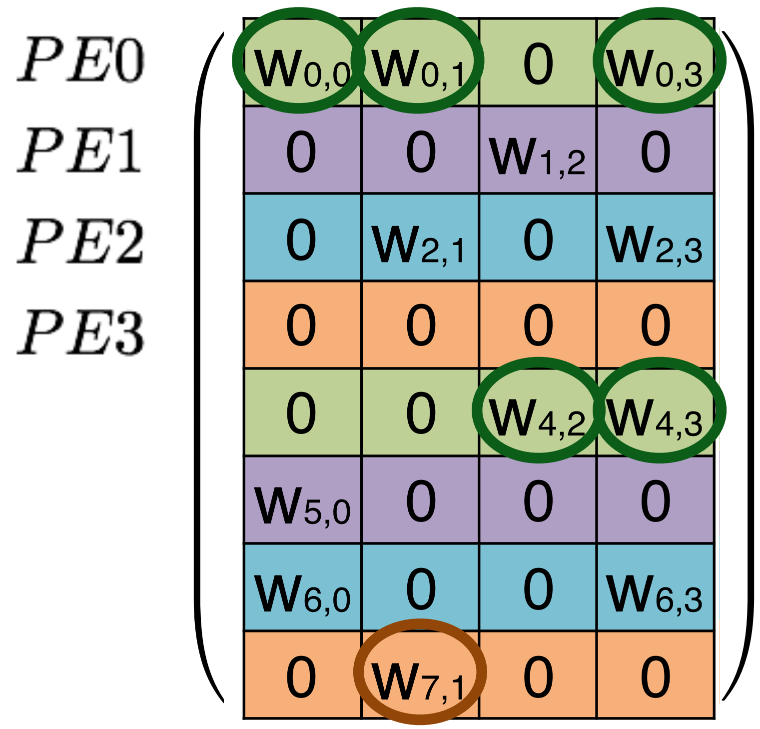
Pruning Lead to Load Imbalance

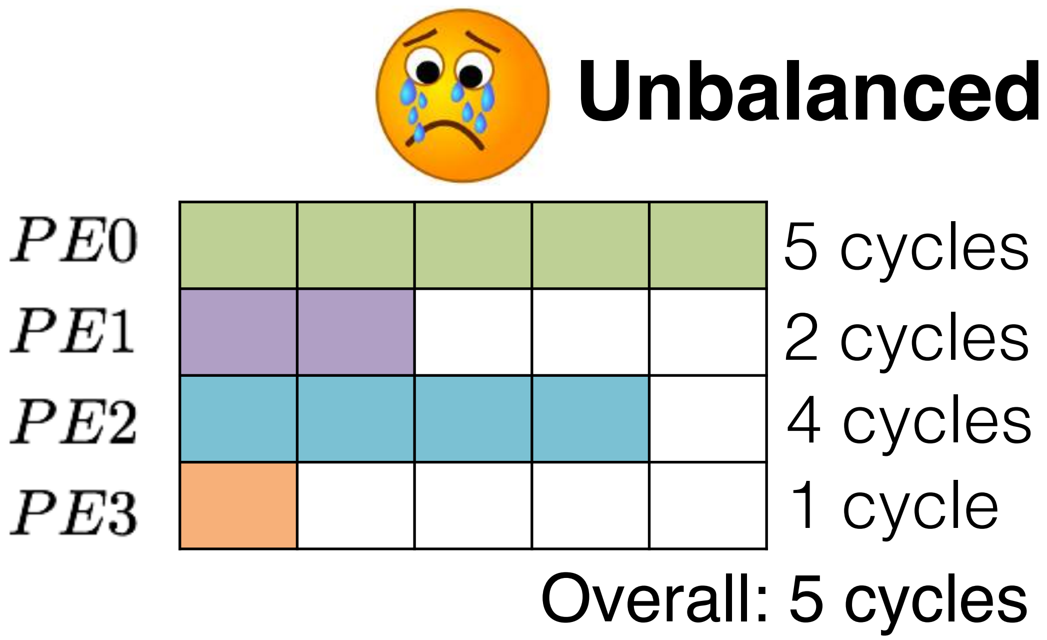
$PE0$	$w_{0,0}$	$w_{0,1}$	0	$w_{0,3}$
$PE1$	0	0	$w_{1,2}$	0
$PE2$	0	$w_{2,1}$	0	$w_{2,3}$
$PE3$	0	0	0	0
	0	0	$w_{4,2}$	$w_{4,3}$
	$w_{5,0}$	0	0	0
	$w_{6,0}$	0	0	$w_{6,3}$
	0	$w_{7,1}$	0	0

Pruning Lead to Load Imbalance

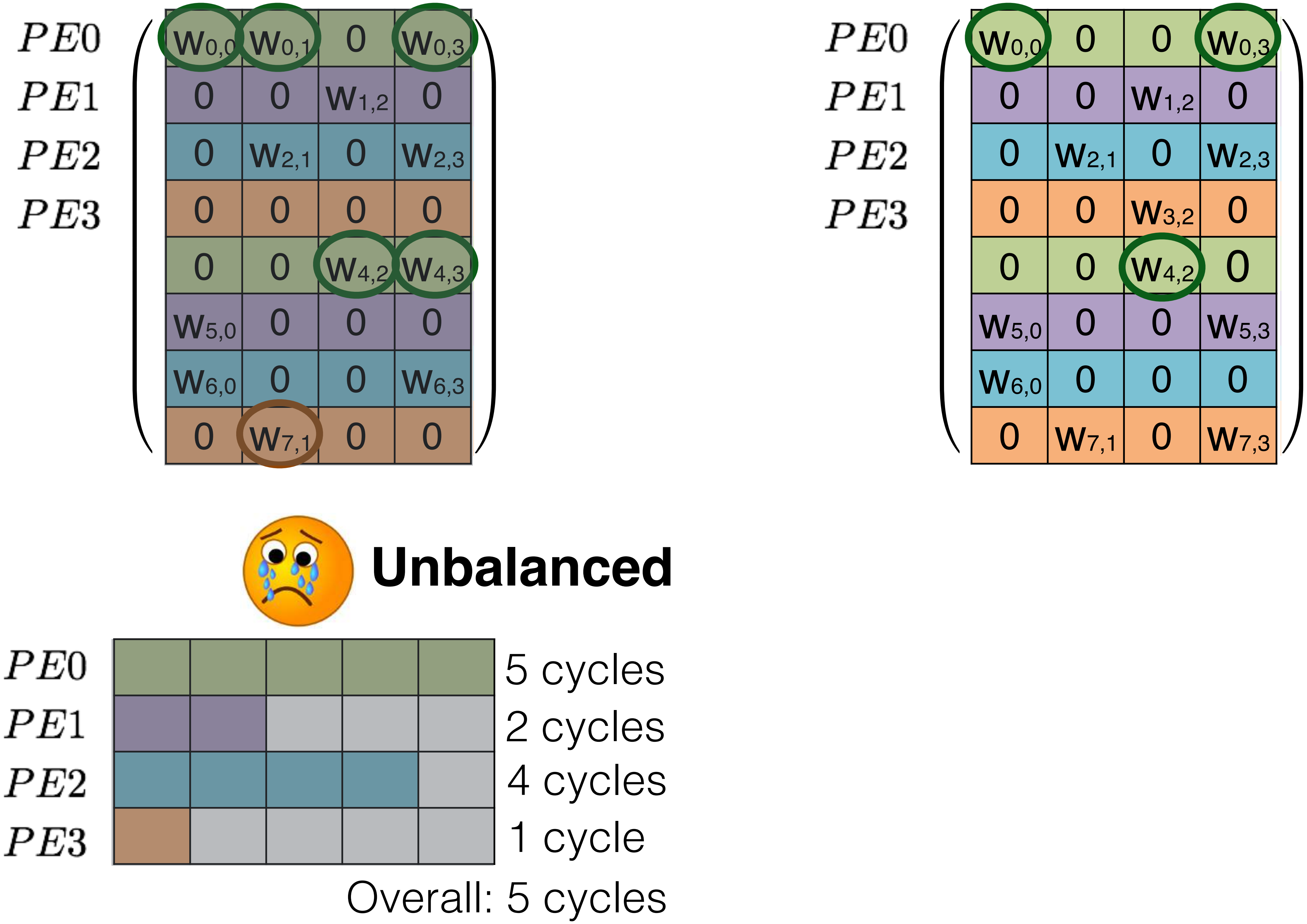


Pruning Lead to Load Imbalance





Load Balance Aware Pruning

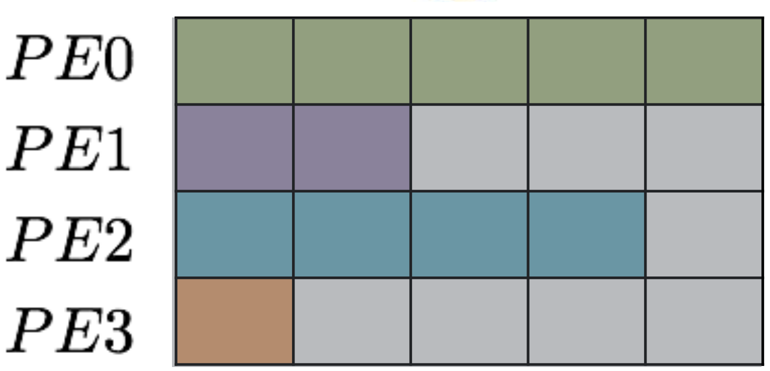


Load Balance Aware Pruning

$PE0$	$w_{0,0}$	$w_{0,1}$	0	$w_{0,3}$
$PE1$	0	0	$w_{1,2}$	0
$PE2$	0	$w_{2,1}$	0	$w_{2,3}$
$PE3$	0	0	0	0
	0	0	$w_{4,2}$	$w_{4,3}$
	$w_{5,0}$	0	0	0
	$w_{6,0}$	0	0	$w_{6,3}$
	0	$w_{7,1}$	0	0

$PE0$	$w_{0,0}$	0	0	$w_{0,3}$
$PE1$	0	0	$w_{1,2}$	0
$PE2$	0	$w_{2,1}$	0	$w_{2,3}$
$PE3$	0	0	$w_{3,2}$	0
	0	0	$w_{4,2}$	0
	$w_{5,0}$	0	0	$w_{5,3}$
	$w_{6,0}$	0	0	0
	0	$w_{7,1}$	0	$w_{7,3}$

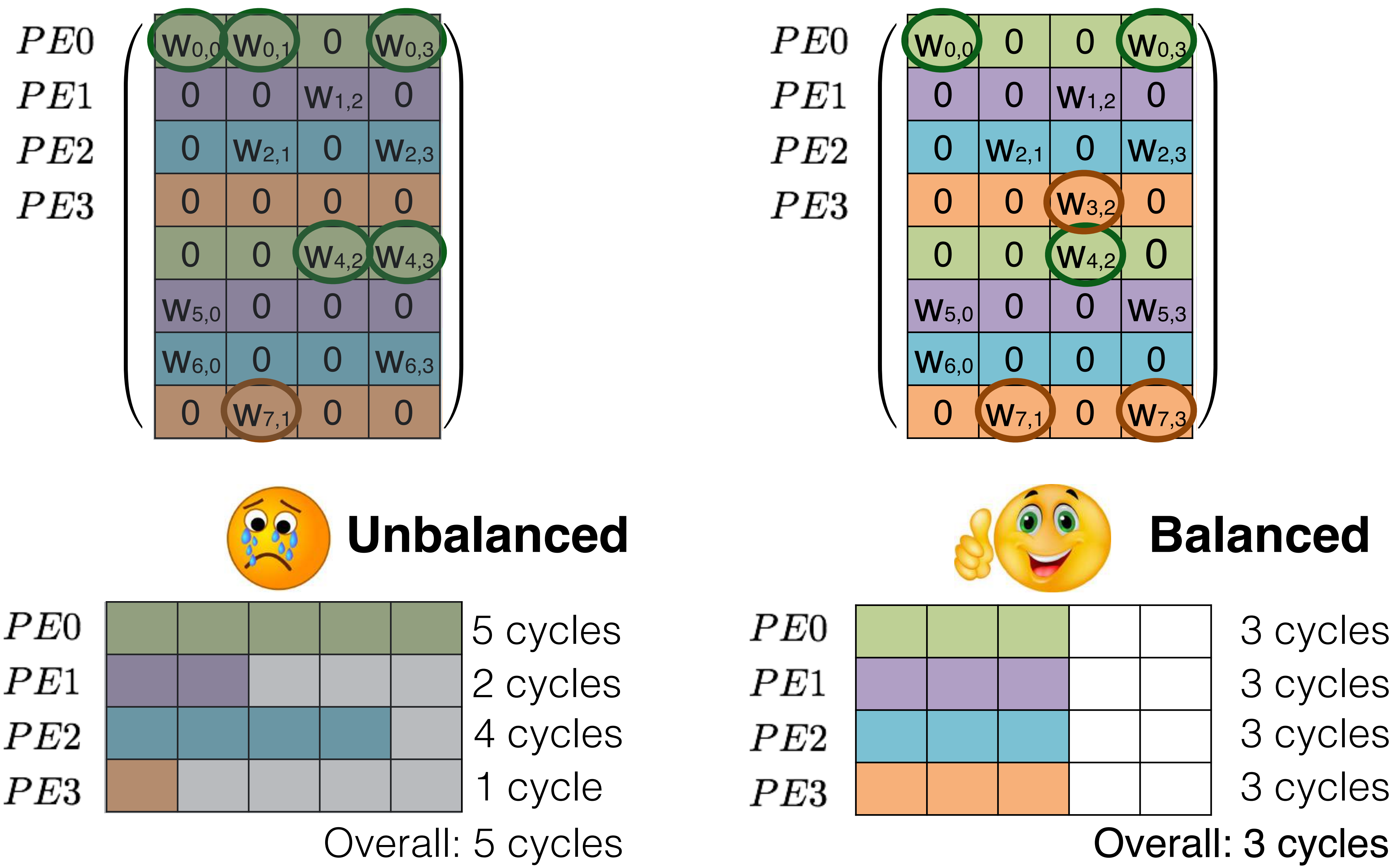
Unbalanced



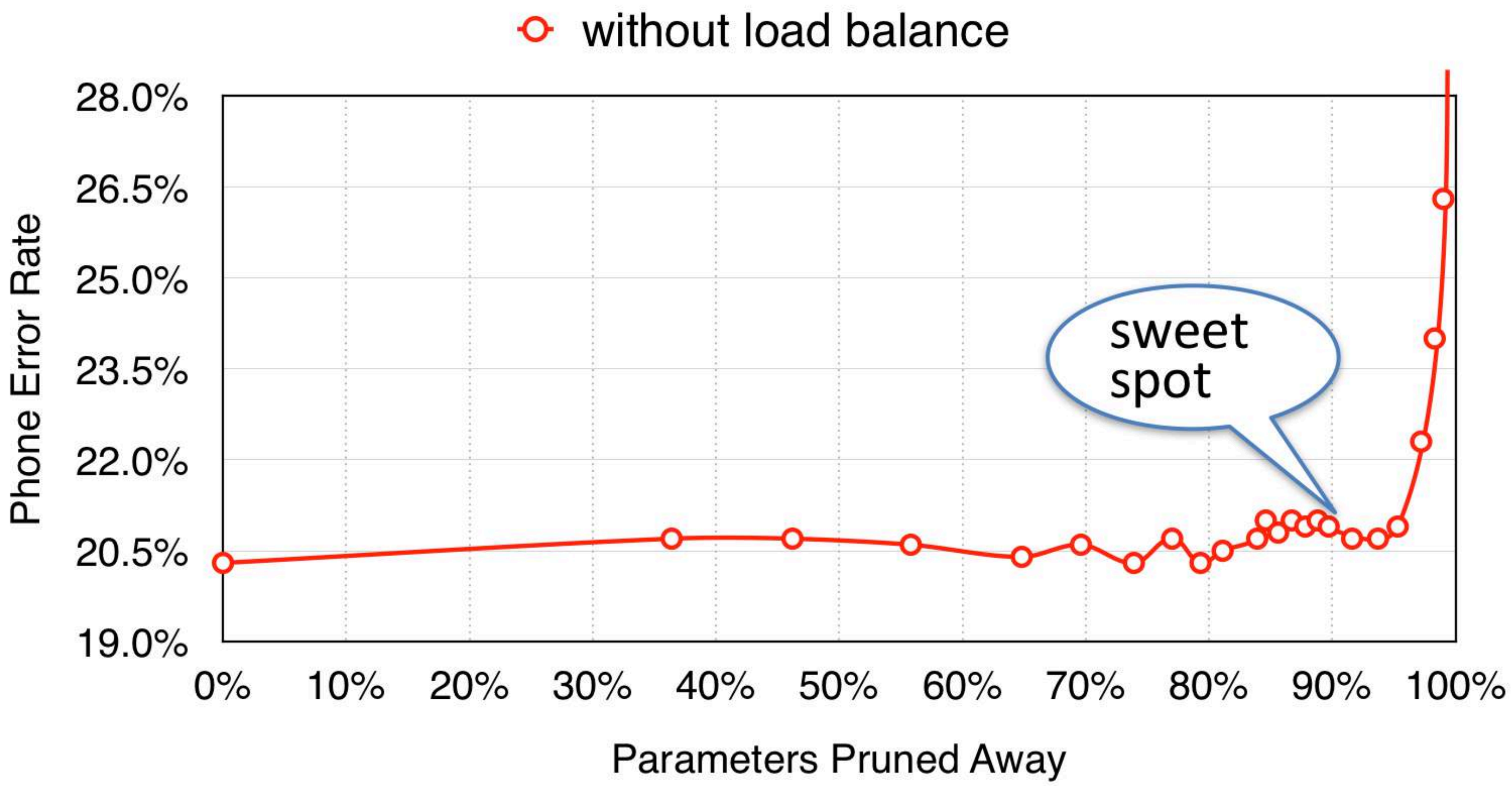
5 cycles
2 cycles
4 cycles
1 cycle

Overall: 5 cycles

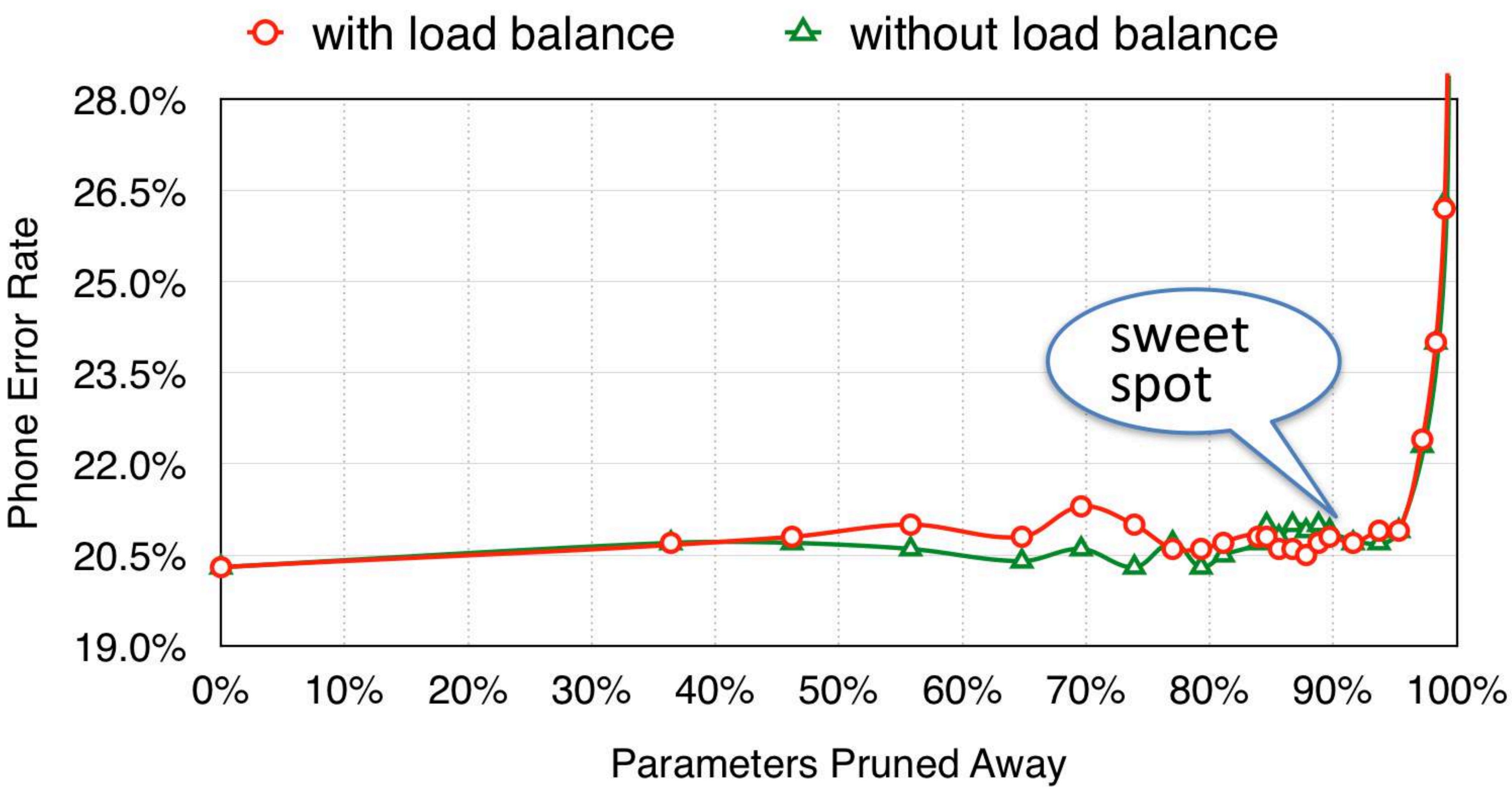
Load Balance Aware Pruning



Accuracy vs Sparsity



Accuracy vs Sparsity



Weight Quantization

Networks	WER
32bit floating original network	20.3%
32bit floating pruned network	20.7%
16bit fixed pruned network	20.7%
12bit fixed pruned network	20.7%
8bit fixed pruned network	84.5%

Agenda

- **Compression**
  Load Balance-Aware Pruning

- **Scheduling**
  Overlap Computation and Memory Reference

- **Accelerated Inference**
  Efficient Architecture for Sparse LSTM

- **Results**

FSM for LSTM

$$i_t = \sigma(W_{ix}x_t + W_{ir}y_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}y_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$

$$g_t = \sigma(W_{cx}x_t + W_{cr}y_{t-1} + b_c) \quad (3)$$

$$c_t = f_t \bigodot c_{t-1} + g_t \bigodot i_t \quad (4)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}y_{t-1} + W_{oc}c_t + b_o) \quad (5)$$

$$m_t = o_t \bigodot h(c_t) \quad (6)$$

$$y_t = W_{ym}m_t \quad (7)$$

Scheduling

$$i_t = \sigma(W_{ix}x_t + W_{ir}y_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}y_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$

$$g_t = \sigma(W_{cx}x_t + W_{cr}y_{t-1} + b_c) \quad (3)$$

$$c_t = f_t \bigodot c_{t-1} + g_t \bigodot i_t \quad (4)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}y_{t-1} + W_{oc}c_t + b_o) \quad (5)$$

$$m_t = o_t \bigodot h(c_t) \quad (6)$$

$$y_t = W_{ym}m_t \quad (7)$$

Memory

spMM

Elt-wise

Data Fetch	Sigmoid /Tanh	W_{ix}	W_{fx}	W_{cx}	W_{ir}	W_{fr}	W_{cr}	W_{ox}	W_{or}	N/A			W_{ym}		W_{ix}
		P	P	P	P	P	P	P	P	N/A			P		P
		x	b_i	W_{ic}	W_{fc}	b_f	b_c	b_o	W_{oc}	N/A			N/A		x
Computation	N/A		$W_{ix}x_t$	$W_{fx}x_t$	$W_{cx}x_t$	$W_{ir}y_{t-1}$	$W_{fr}y_{t-1}$	$W_{cr}y_{t-1}$	$W_{ox}x_t$	$W_{or}y_{t-1}$	N/A		N/A		y_t
	N/A		N/A		$W_{ic}c_{t-1}$	$W_{cf}c_{t-1}$	i_t	f_t	g_t	c_t	$W_{oc}c_t$	h_t	o_t	m_t	N/A
STATE	INITIAL		STATE_1			STATE_2			STATE_3	STATE_4			STATE_5		STATE_6

Sparse matrix-vector multiplication by *SpMV*

Accumulate operations by *Adder Tree*

Element-wise multiplication by *ElemMul*

Fetch data for the next operation

N/A Idle state

Scheduling

$$i_t = \sigma(W_{ix}x_t + W_{ir}y_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}y_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$

$$g_t = \sigma(W_{cx}x_t + W_{cr}y_{t-1} + b_c) \quad (3)$$

$$c_t = f_t \odot c_{t-1} + g_t \odot i_t \quad (4)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}y_{t-1} + W_{oc}c_t + b_o) \quad (5)$$

$$m_t = o_t \odot h(c_t) \quad (6)$$

$$y_t = W_{ym}m_t \quad (7)$$

Memory

spMM

Elt-wise

Data Fetch	Sigmoid /Tanh	W_{ix}	W_{fx}	W_{cx}	W_{ir}	W_{fr}	W_{cr}	W_{ox}	W_{or}	N/A			W_{ym}		W_{ix}
		P	P	P	P	P	P	P	P	N/A			P		P
		x	b_i	W_{ic}	W_{fc}	b_f	b_c	b_o	W_{oc}	N/A			N/A		x
Computation	N/A		$W_{ix}x_t$	$W_{fx}x_t$	$W_{cx}x_t$	$W_{ir}y_{t-1}$	$W_{fr}y_{t-1}$	$W_{cr}y_{t-1}$	$W_{ox}x_t$	$W_{or}y_{t-1}$	N/A		N/A		y_t
	N/A		N/A		$W_{ic}c_{t-1}$	$W_{cf}c_{t-1}$	i_t	f_t	g_t	c_t	$W_{oc}c_t$	h_t	o_t	m_t	N/A
STATE	INITIAL		STATE_1			STATE_2			STATE_3	STATE_4			STATE_5		STATE_6

Sparse matrix-vector multiplication by *SpMV*

Accumulate operations by *Adder Tree*

Element-wise multiplication by *ElemMul*

Fetch data for the next operation

N/A Idle state

Scheduling

Memory

spMM

Elt-wise

$$i_t = \sigma(W_{ix}x_t + W_{ir}y_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}y_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$

$$g_t = \sigma(W_{cx}x_t + W_{cr}y_{t-1} + b_c) \quad (3)$$

$$c_t = f_t \bigodot c_{t-1} + g_t \bigodot i_t \quad (4)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}y_{t-1} + W_{oc}c_t + b_o) \quad (5)$$

$$m_t = o_t \bigodot h(c_t) \quad (6)$$

$$y_t = W_{ym}m_t \quad (7)$$

Data Fetch	Sigmoid /Tanh	W_{ix}	W_{fx}	W_{cx}	W_{ir}	W_{fr}	W_{cr}	W_{ox}	W_{or}	N/A	W_{ym}	W_{ix}
		P	P	P	P	P	P	P	P	N/A	P	P
		x	b_i	W_{ic}	W_{fc}	b_f	b_c	b_o	W_{oc}	N/A	N/A	x
Computation	N/A		$W_{ix}x_t$	$W_{fx}x_t$	$W_{cx}x_t$	$W_{ir}y_{t-1}$	$W_{fr}y_{t-1}$	$W_{cr}y_{t-1}$	$W_{ox}x_t$	$W_{or}y_{t-1}$ / N/A	N/A	y_t
	N/A		N/A		$W_{ic}c_{t-1}$	$W_{cf}c_{t-1}$	i_t	f_t	g_t	c_t / $W_{oc}c_t$ / h_t	o_t / m_t	N/A
STATE	INITIAL		STATE_1			STATE_2			STATE_3	STATE_4	STATE_5	STATE_6

Sparse matrix-vector multiplication by *SpMV*

Accumulate operations by *Adder Tree*

Element-wise multiplication by *ElemMul*

Fetch data for the next operation

N/A Idle state

Scheduling

Memory

spMM

Elt-wise

$$i_t = \sigma(W_{ix}x_t + W_{ir}y_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}y_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$

$$g_t = \sigma(W_{cx}x_t + W_{cr}y_{t-1} + b_c) \quad (3)$$

$$c_t = f_t \odot c_{t-1} + g_t \odot i_t \quad (4)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}y_{t-1} + W_{oc}c_t + b_o) \quad (5)$$

$$m_t = o_t \odot h(c_t) \quad (6)$$

$$y_t = W_{ym}m_t \quad (7)$$

Data Fetch	Sigmoid /Tanh	W_{ix}	W_{fx}	W_{cx}	W_{ir}	W_{fr}	W_{cr}	W_{ox}	W_{or}	N/A			W_{ym}		W_{ix}
		P	P	P	P	P	P	P	P	N/A			P		P
		x	b_i	W_{ic}	W_{fc}	b_f	b_c	b_o	W_{oc}	N/A			N/A		x
Computation	N/A		$W_{ix}x_t$	$W_{fx}x_t$	$W_{cx}x_t$	$W_{ir}y_{t-1}$	$W_{fr}y_{t-1}$	$W_{cr}y_{t-1}$	$W_{ox}x_t$	$W_{or}y_{t-1}$	N/A		N/A		y_t
	N/A		N/A		$W_{ic}c_{t-1}$	$W_{cf}c_{t-1}$	i_t	f_t	g_t	c_t	$W_{oc}c_t$	h_t	o_t	m_t	N/A
STATE	INITIAL		STATE_1			STATE_2			STATE_3	STATE_4			STATE_5		STATE_6

Sparse matrix-vector multiplication by *SpMV*

Accumulate operations by *Adder Tree*

Element-wise multiplication by *ElemMul*

Fetch data for the next operation

N/A Idle state

Scheduling

Memory

spMM

Elt-wise

$$i_t = \sigma(W_{ix}x_t + W_{ir}y_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}y_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$

$$g_t = \sigma(W_{cx}x_t + W_{cr}y_{t-1} + b_c) \quad (3)$$

$$c_t = f_t \odot c_{t-1} + g_t \odot i_t \quad (4)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}y_{t-1} + W_{oc}c_t + b_o) \quad (5)$$

$$m_t = o_t \odot h(c_t) \quad (6)$$

$$y_t = W_{ym}m_t \quad (7)$$

Data Fetch	Sigmoid /Tanh	W_{ix}	W_{fx}	W_{cx}	W_{ir}	W_{fr}	W_{cr}	W_{ox}	W_{or}	*N/A*			W_{ym}		W_{ix}
		P	P	P	P	P	P	P	P	*N/A*			P		P
		x	b_i	W_{ic}	W_{fc}	b_f	b_c	b_o	W_{oc}	*N/A*			*N/A*		x
Computation	*N/A*		$W_{ix}x_t$	$W_{fx}x_t$	$W_{cx}x_t$	$W_{ir}y_{t-1}$	$W_{fr}y_{t-1}$	$W_{cr}y_{t-1}$	$W_{ox}x_t$	$W_{or}y_{t-1}$	*N/A*		*N/A*		y_t
	N/A		*N/A*		$W_{ic}c_{t-1}$	$W_{cf}c_{t-1}$	i_t	f_t	g_t	c_t	$W_{oc}c_t$	h_t	o_t	m_t	*N/A*
STATE	INITIAL		STATE_1				STATE_2		STATE_3	STATE_4			STATE_5		STATE_6

Sparse matrix-vector multiplication by *SpMV*

Element-wise multiplication by *ElemMul*

N/A Idle state

Accumulate operations by *Adder Tree*

Fetch data for the next operation

Scheduling

Memory

spMM

Elt-wise

$$i_t = \sigma(W_{ix}x_t + W_{ir}y_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}y_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$

$$g_t = \sigma(W_{cx}x_t + W_{cr}y_{t-1} + b_c) \quad (3)$$

$$c_t = f_t \odot c_{t-1} + g_t \odot i_t \quad (4)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}y_{t-1} + W_{oc}c_t + b_o) \quad (5)$$

$$m_t = o_t \odot h(c_t) \quad (6)$$

$$y_t = W_{ym}m_t \quad (7)$$

Data Fetch	Sigmoid /Tanh	W_{ix}	W_{fx}	W_{cx}	W_{ir}	W_{fr}	W_{cr}	W_{ox}	W_{or}	*N/A*			W_{ym}		W_{ix}
		P	P	P	P	P	P	P	P	*N/A*			P		P
		x	b_i	W_{ic}	W_{fc}	b_f	b_c	b_o	W_{oc}	*N/A*			*N/A*		x
Computation	*N/A*		$W_{ix}x_t$	$W_{fx}x_t$	$W_{cx}x_t$	$W_{ir}y_{t-1}$	$W_{fr}y_{t-1}$	$W_{cr}y_{t-1}$	$W_{ox}x_t$	$W_{or}y_{t-1}$	*N/A*		*N/A*		y_t
	N/A		*N/A*		$W_{ic}c_{t-1}$	$W_{cf}c_{t-1}$	i_t	f_t	g_t	c_t	$W_{oc}c_t$	h_t	o_t	m_t	*N/A*
STATE	INITIAL		STATE_1				STATE_2		STATE_3	STATE_4			STATE_5		STATE_6

Sparse matrix-vector multiplication by *SpMV*

Accumulate operations by *Adder Tree*

Element-wise multiplication by *ElemMul*

Fetch data for the next operation

N/A Idle state

Scheduling

Memory

spMM

Elt-wise

$$i_t = \sigma(W_{ix}x_t + W_{ir}y_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}y_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$

$$g_t = \sigma(W_{cx}x_t + W_{cr}y_{t-1} + b_c) \quad (3)$$

$$c_t = f_t \odot c_{t-1} + g_t \odot i_t \quad (4)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}y_{t-1} + W_{oc}c_t + b_o) \quad (5)$$

$$m_t = o_t \odot h(c_t) \quad (6)$$

$$y_t = W_{ym}m_t \quad (7)$$

Data Fetch	Sigmoid /Tanh	W_{ix}	W_{fx}	W_{cx}	W_{ir}	W_{fr}	W_{cr}	W_{ox}	W_{or}	N/A	W_{ym}	W_{ix}
		P	P	P	P	P	P	P	P	N/A	P	P
		x	b_i	W_{ic}	W_{fc}	b_f	b_c	b_o	W_{oc}	N/A	N/A	x
Computation	N/A		$W_{ix}x_t$	$W_{fx}x_t$	$W_{cx}x_t$	$W_{ir}y_{t-1}$	$W_{fr}y_{t-1}$	$W_{cr}y_{t-1}$	$W_{ox}x_t$	$W_{or}y_{t-1}$ / N/A	N/A	y_t
	N/A		N/A		$W_{ic}c_{t-1}$	$W_{cf}c_{t-1}$	i_t	f_t	g_t	c_t / $W_{oc}c_t$ / h_t	o_t / m_t	N/A
STATE	INITIAL		STATE_1			STATE_2			STATE_3	STATE_4	STATE_5	STATE_6

Sparse matrix-vector multiplication by *SpMV*

Accumulate operations by *Adder Tree*

Element-wise multiplication by *ElemMul*

Fetch data for the next operation

N/A Idle state

Scheduling

$$i_t = \sigma(W_{ix}x_t + W_{ir}y_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}y_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$

$$g_t = \sigma(W_{cx}x_t + W_{cr}y_{t-1} + b_c) \quad (3)$$

$$c_t = f_t \odot c_{t-1} + g_t \odot i_t \quad (4)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}y_{t-1} + W_{oc}c_t + b_o) \quad (5)$$

$$m_t = o_t \odot h(c_t) \quad (6)$$

$$y_t = W_{ym}m_t \quad (7)$$

Memory

spMM

Elt-wise

Data Fetch	Sigmoid /Tanh	W_{ix}	W_{fx}	W_{cx}	W_{ir}	W_{fr}	W_{cr}	W_{ox}	W_{or}	N/A			W_{ym}		W_{ix}
		P	P	P	P	P	P	P	P	N/A			P		P
		x	b_i	W_{ic}	W_{fc}	b_f	b_c	b_o	W_{oc}	N/A			N/A		x
Computation	N/A		$W_{ix}x_t$	$W_{fx}x_t$	$W_{cx}x_t$	$W_{ir}y_{t-1}$	$W_{fr}y_{t-1}$	$W_{cr}y_{t-1}$	$W_{ox}x_t$	$W_{or}y_{t-1}$	N/A		N/A		y_t
	N/A		N/A		$W_{ic}c_{t-1}$	$W_{cf}c_{t-1}$	i_t	f_t	g_t	c_t	$W_{oc}c_t$	h_t	o_t	m_t	N/A
STATE	INITIAL		STATE_1			STATE_2			STATE_3	STATE_4			STATE_5		STATE_6

Sparse matrix-vector multiplication by *SpMV*

Accumulate operations by *Adder Tree*

Element-wise multiplication by *ElemMul*

Fetch data for the next operation

N/A Idle state

Scheduling

Memory

spMM

Elt-wise

$$i_t = \sigma(W_{ix}x_t + W_{ir}y_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}y_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$

$$g_t = \sigma(W_{cx}x_t + W_{cr}y_{t-1} + b_c) \quad (3)$$

$$c_t = f_t \odot c_{t-1} + g_t \odot i_t \quad (4)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}y_{t-1} + W_{oc}c_t + b_o) \quad (5)$$

$$m_t = o_t \odot h(c_t) \quad (6)$$

$$y_t = W_{ym}m_t \quad (7)$$

Data Fetch	Sigmoid /Tanh	W_{ix}	W_{fx}	W_{cx}	W_{ir}	W_{fr}	W_{cr}	W_{ox}	W_{or}		N/A	W_{ym}	W_{ix}
		P	P	P	P	P	P	P	P		N/A	P	P
		x	b_i	W_{ic}	W_{fc}	b_f	b_c	b_o	W_{oc}		N/A	N/A	x
Computation	N/A		$W_{ix}x_t$	$W_{fx}x_t$	$W_{cx}x_t$	$W_{ir}y_{t-1}$	$W_{fr}y_{t-1}$	$W_{cr}y_{t-1}$	$W_{ox}x_t$	$W_{or}y_{t-1}$	N/A	N/A	y_t
	N/A		N/A		$W_{ic}c_{t-1}$	$W_{cf}c_{t-1}$	i_t	f_t	g_t	c_t	$W_{oc}c_t$ / h_t	o_t / m_t	N/A
STATE	INITIAL		STATE_1			STATE_2			STATE_3	STATE_4		STATE_5	STATE_6

Sparse matrix-vector multiplication by *SpMV*

Accumulate operations by *Adder Tree*

Element-wise multiplication by *ElemMul*

Fetch data for the next operation

N/A Idle state

Scheduling

$$i_t = \sigma(W_{ix}x_t + W_{ir}y_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}y_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$

$$g_t = \sigma(W_{cx}x_t + W_{cr}y_{t-1} + b_c) \quad (3)$$

$$c_t = f_t \odot c_{t-1} + g_t \odot i_t \quad (4)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}y_{t-1} + W_{oc}c_t + b_o) \quad (5)$$

$$m_t = o_t \odot h(c_t) \quad (6)$$

$$y_t = W_{ym}m_t \quad (7)$$

Memory

spMM

Elt-wise

Data Fetch	Sigmoid /Tanh	W_{ix}	W_{fx}	W_{cx}	W_{ir}	W_{fr}	W_{cr}	W_{ox}	W_{or}	N/A			W_{ym}		W_{ix}
		P	P	P	P	P	P	P	P	N/A			P		P
		x	b_i	W_{ic}	W_{fc}	b_f	b_c	b_o	W_{oc}	N/A			N/A		x
Computation	N/A		$W_{ix}x_t$	$W_{fx}x_t$	$W_{cx}x_t$	$W_{ir}y_{t-1}$	$W_{fr}y_{t-1}$	$W_{cr}y_{t-1}$	$W_{ox}x_t$	$W_{or}y_{t-1}$	N/A		N/A		y_t
	N/A		N/A		$W_{ic}c_{t-1}$	$W_{cf}c_{t-1}$	i_t	f_t	g_t	c_t	$W_{oc}c_t$	h_t	o_t	m_t	N/A
STATE	INITIAL		STATE_1			STATE_2			STATE_3	STATE_4			STATE_5		STATE_6

Sparse matrix-vector multiplication by *SpMV*
Element-wise multiplication by *ElemMul*
N/A Idle state
Accumulate operations by *Adder Tree*
Fetch data for the next operation

Scheduling

$$i_t = \sigma(W_{ix}x_t + W_{ir}y_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}y_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$

$$g_t = \sigma(W_{cx}x_t + W_{cr}y_{t-1} + b_c) \quad (3)$$

$$c_t = f_t \bigodot c_{t-1} + g_t \bigodot i_t \quad (4)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}y_{t-1} + W_{oc}c_t + b_o) \quad (5)$$

$$m_t = o_t \bigodot h(c_t) \quad (6)$$

$$y_t = W_{ym}m_t \quad (7)$$

Memory

spMM

Elt-wise

Data Fetch	Sigmoid /Tanh	W_{ix}	W_{fx}	W_{cx}	W_{ir}	W_{fr}	W_{cr}	W_{ox}	W_{or}	*N/A*			W_{ym}		W_{ix}
		P	P	P	P	P	P	P	P	*N/A*			P		P
		x	b_i	W_{ic}	W_{fc}	b_f	b_c	b_o	W_{oc}	*N/A*			*N/A*		x
Computation	*N/A*		$W_{ix}x_t$	$W_{fx}x_t$	$W_{cx}x_t$	$W_{ir}y_{t-1}$	$W_{fr}y_{t-1}$	$W_{cr}y_{t-1}$	$W_{ox}x_t$	$W_{or}y_{t-1}$	*N/A*		*N/A*		y_t
	N/A		*N/A*		$W_{ic}c_{t-1}$	$W_{cf}c_{t-1}$	i_t	f_t	g_t	c_t	$W_{oc}c_t$	h_t	o_t	m_t	*N/A*
STATE	**INITIAL**		**STATE_1**			**STATE_2**			**STATE_3**	**STATE_4**			**STATE_5**		**STATE_6**

Sparse matrix-vector multiplication by *SpMV*
Accumulate operations by *Adder Tree*
Element-wise multiplication by *ElemMul*
Fetch data for the next operation
N/A Idle state

Scheduling

Memory

spMM

Elt-wise

$$i_t = \sigma(W_{ix}x_t + W_{ir}y_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}y_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$

$$g_t = \sigma(W_{cx}x_t + W_{cr}y_{t-1} + b_c) \quad (3)$$

$$c_t = f_t \odot c_{t-1} + g_t \odot i_t \quad (4)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}y_{t-1} + W_{oc}c_t + b_o) \quad (5)$$

$$m_t = o_t \odot h(c_t) \quad (6)$$

$$y_t = W_{ym} m_t \quad (7)$$

Data Fetch	Sigmoid /Tanh	W_{ix}	W_{fx}	W_{cx}	W_{ir}	W_{fr}	W_{cr}	W_{ox}	W_{or}	*N/A*			W_{ym}		W_{ix}
		P	P	P	P	P	P	P	P	*N/A*			P		P
		x	b_i	W_{ic}	W_{fc}	b_f	b_c	b_o	W_{oc}	*N/A*			*N/A*		x
Computation	*N/A*		$W_{ix}x_t$	$W_{fx}x_t$	$W_{cx}x_t$	$W_{ir}y_{t-1}$	$W_{fr}y_{t-1}$	$W_{cr}y_{t-1}$	$W_{ox}x_t$	$W_{or}y_{t-1}$	*N/A*		*N/A*		y_t
	N/A		*N/A*		$W_{ic}c_{t-1}$	$W_{cf}c_{t-1}$	i_t	f_t	g_t	c_t	$W_{oc}c_t$	h_t	o_t	m_t	*N/A*
STATE	INITIAL		STATE_1			STATE_2			STATE_3	STATE_4			STATE_5		STATE_6

Sparse matrix-vector multiplication by *SpMV*

Accumulate operations by *Adder Tree*

Element-wise multiplication by *ElemMul*

Fetch data for the next operation

N/A Idle state

Scheduling

Memory

spMM

Elt-wise

$$i_t = \sigma(W_{ix}x_t + W_{ir}y_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}y_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$

$$g_t = \sigma(W_{cx}x_t + W_{cr}y_{t-1} + b_c) \quad (3)$$

$$c_t = f_t \bigodot c_{t-1} + g_t \bigodot i_t \quad (4)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}y_{t-1} + W_{oc}c_t + b_o) \quad (5)$$

$$m_t = o_t \bigodot h(c_t) \quad (6)$$

$$y_t = W_{ym}m_t \quad (7)$$

Data Fetch	Sigmoid /Tanh	W_{ix}	W_{fx}	W_{cx}	W_{ir}	W_{fr}	W_{cr}	W_{ox}	W_{or}	*N/A*			W_{ym}		W_{ix}
		P	P	P	P	P	P	P	P	*N/A*			P		P
		x	b_i	W_{ic}	W_{fc}	b_f	b_c	b_o	W_{oc}	*N/A*			*N/A*		x
Computation	*N/A*		$W_{ix}x_t$	$W_{fx}x_t$	$W_{cx}x_t$	$W_{ir}y_{t-1}$	$W_{fr}y_{t-1}$	$W_{cr}y_{t-1}$	$W_{ox}x_t$	$W_{or}y_{t-1}$	*N/A*		*N/A*		y_t
	N/A		*N/A*		$W_{ic}c_{t-1}$	$W_{cf}c_{t-1}$	i_t	f_t	g_t	c_t	$W_{oc}c_t$	h_t	o_t	m_t	*N/A*
STATE	**INITIAL**		**STATE_1**			**STATE_2**			**STATE_3**	**STATE_4**			**STATE_5**		**STATE_6**

Sparse matrix-vector multiplication by *SpMV*

Accumulate operations by *Adder Tree*

Element-wise multiplication by *ElemMul*

Fetch data for the next operation

N/A Idle state

Scheduling

Memory
spMM
Elt-wise

$$i_t = \sigma(W_{ix}x_t + W_{ir}y_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}y_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$

$$g_t = \sigma(W_{cx}x_t + W_{cr}y_{t-1} + b_c) \quad (3)$$

$$c_t = f_t \bigodot c_{t-1} + g_t \bigodot i_t \quad (4)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}y_{t-1} + W_{oc}c_t + b_o) \quad (5)$$

$$m_t = o_t \bigodot h(c_t) \quad (6)$$

$$y_t = W_{ym}m_t \quad (7)$$

<table>
<tr><td rowspan="3">Data Fetch</td><td rowspan="3">Sigmoid /Tanh</td><td>W_{ix}</td><td>W_{fx}</td><td>W_{cx}</td><td>W_{ir}</td><td>W_{fr}</td><td>W_{cr}</td><td>W_{ox}</td><td>W_{or}</td><td colspan="3">N/A</td><td colspan="2">W_{ym}</td><td>W_{ix}</td></tr>
<tr><td>P</td><td>P</td><td>P</td><td>P</td><td>P</td><td>P</td><td>P</td><td>P</td><td colspan="3">N/A</td><td colspan="2">P</td><td>P</td></tr>
<tr><td>x</td><td>b_i</td><td>W_{ic}</td><td>W_{fc}</td><td>b_f</td><td>b_c</td><td>b_o</td><td>W_{oc}</td><td colspan="3">N/A</td><td colspan="2">N/A</td><td>x</td></tr>
<tr><td rowspan="2">Computation</td><td colspan="2">N/A</td><td>$W_{ix}x_t$</td><td>$W_{fx}x_t$</td><td>$W_{cx}x_t$</td><td>$W_{ir}y_{t-1}$</td><td>$W_{fr}y_{t-1}$</td><td>$W_{cr}y_{t-1}$</td><td>$W_{ox}x_t$</td><td>$W_{or}y_{t-1}$</td><td colspan="2">N/A</td><td colspan="2">N/A</td><td>y_t</td></tr>
<tr><td colspan="2">N/A</td><td colspan="2">N/A</td><td>$W_{ic}c_{t-1}$</td><td>$W_{cf}c_{t-1}$</td><td>i_t</td><td>f_t</td><td>g_t</td><td>c_t</td><td>$W_{oc}c_t$</td><td>h_t</td><td>o_t</td><td>m_t</td><td>N/A</td></tr>
<tr><td>STATE</td><td colspan="2">INITIAL</td><td colspan="3">STATE_1</td><td colspan="3">STATE_2</td><td>STATE_3</td><td colspan="3">STATE_4</td><td colspan="2">STATE_5</td><td>STATE_6</td></tr>
</table>

Sparse matrix-vector multiplication by *SpMV*
Element-wise multiplication by *ElemMul*
N/A Idle state
Accumulate operations by *Adder Tree*
Fetch data for the next operation

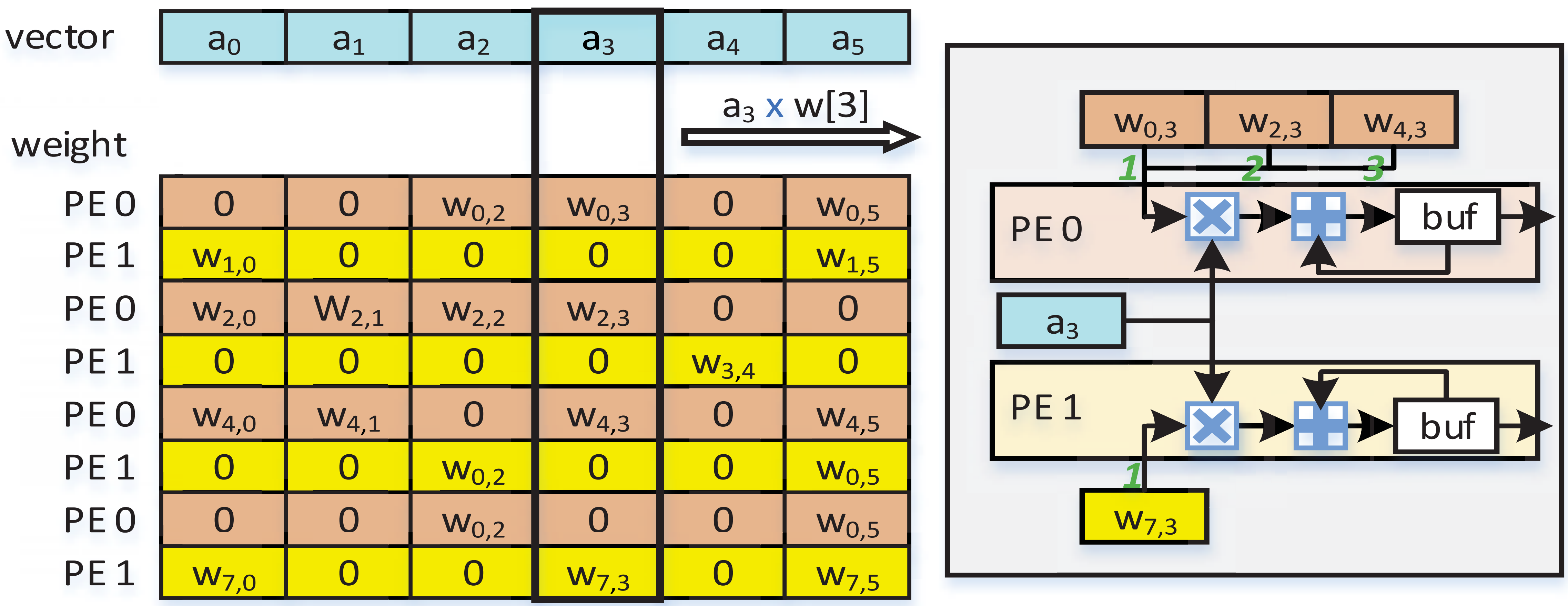
Agenda

- **Compression**
  Load Balance-Aware Pruning
- **Scheduling**
  Overlap Computation and Memory Reference
- **Accelerated Inference**
  Efficient Architecture for Sparse LSTM
- **Results**

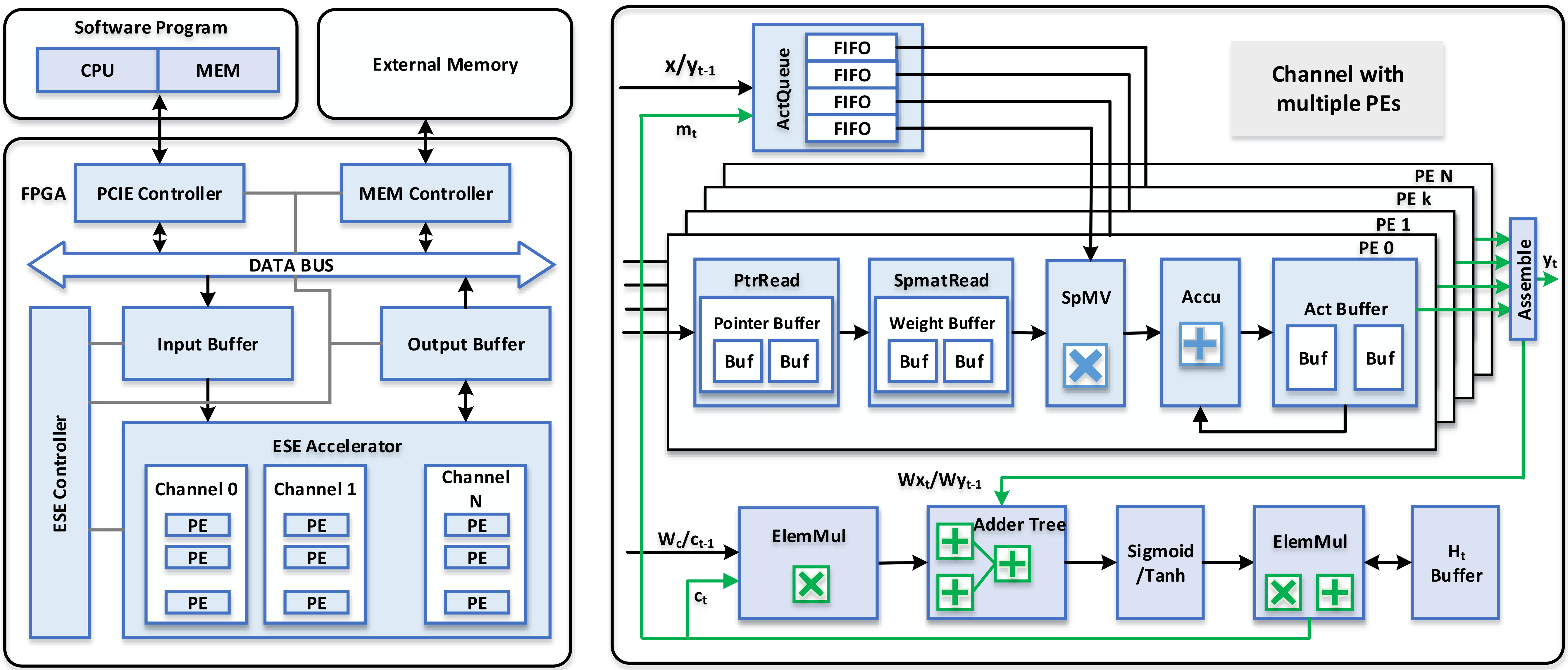
Challenges

- **Online de-compression while computing**
  - Special purpose logic
- **Computation becomes irregular**
  - Sparsity
- **Parallelization becomes challenging**
  - Load balance

Deal with Sparsity

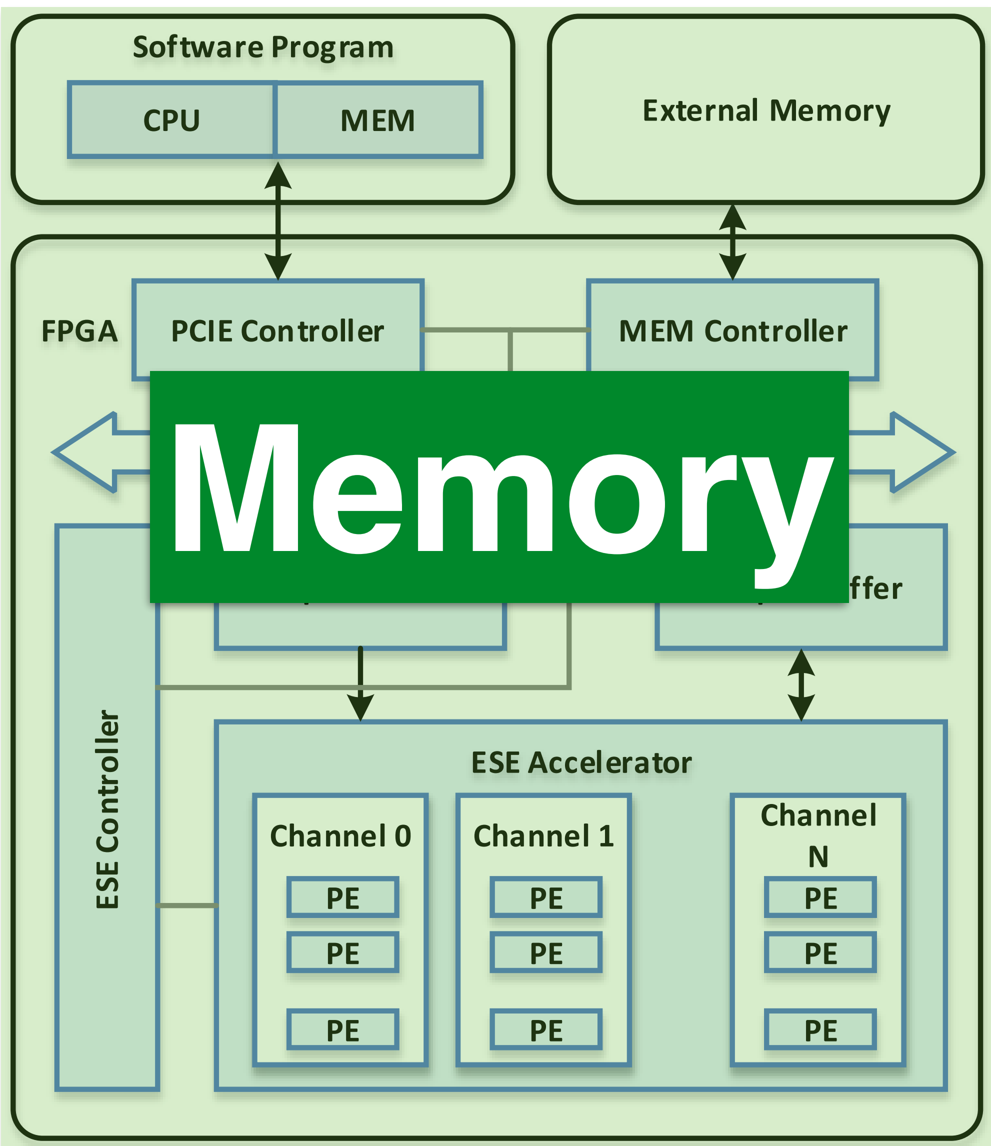


Hardware Architecture

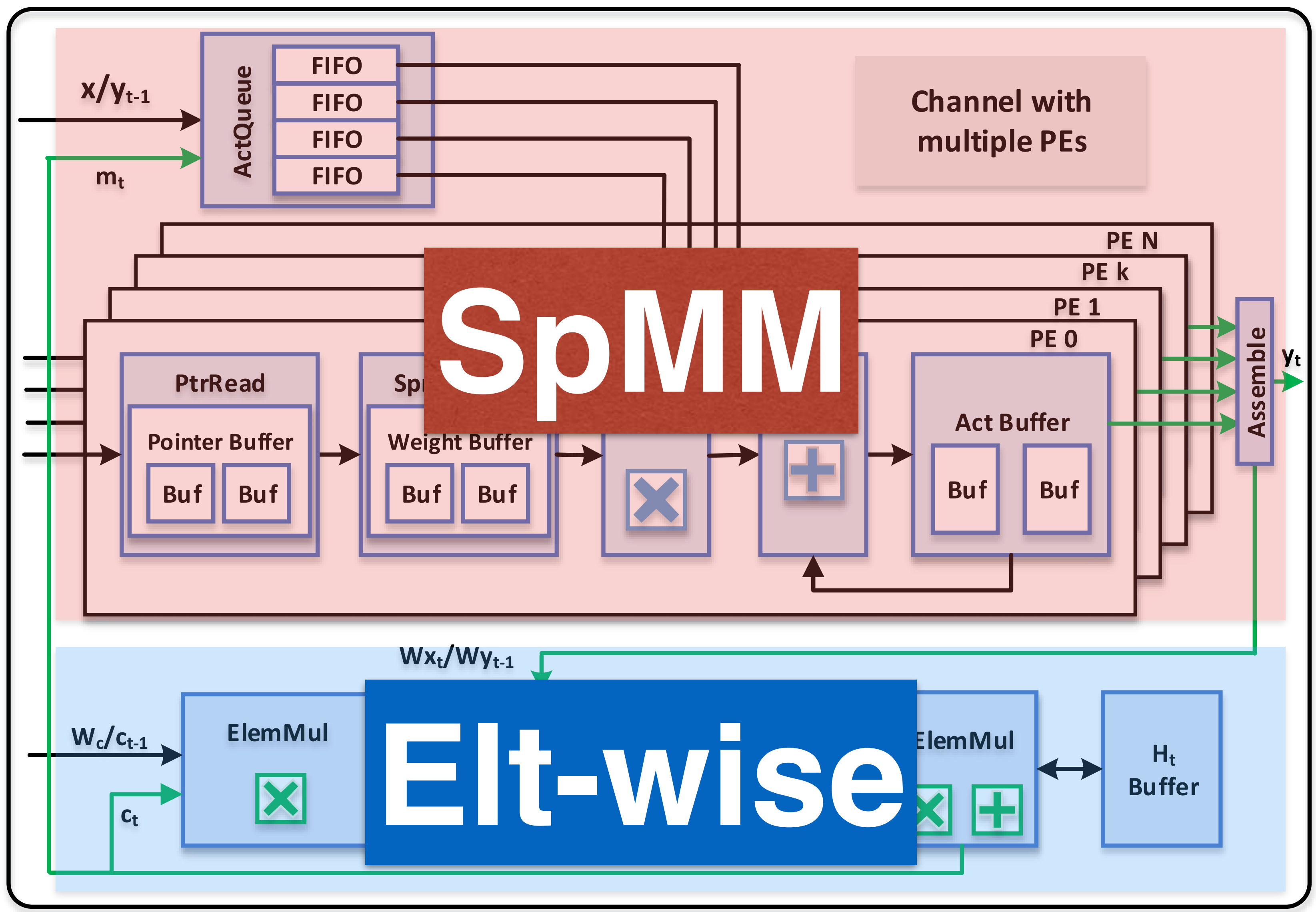


(a) (b)

Hardware Architecture

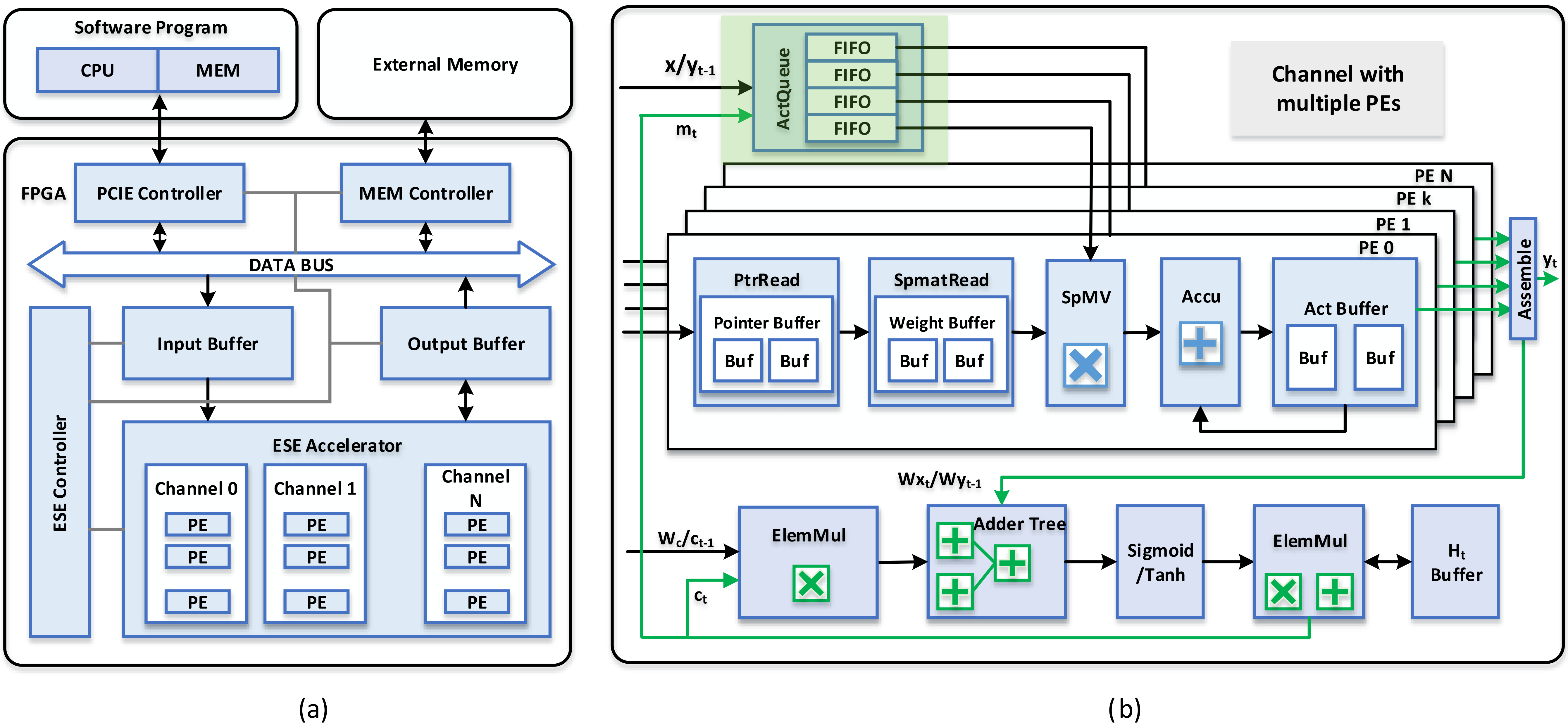


(a)

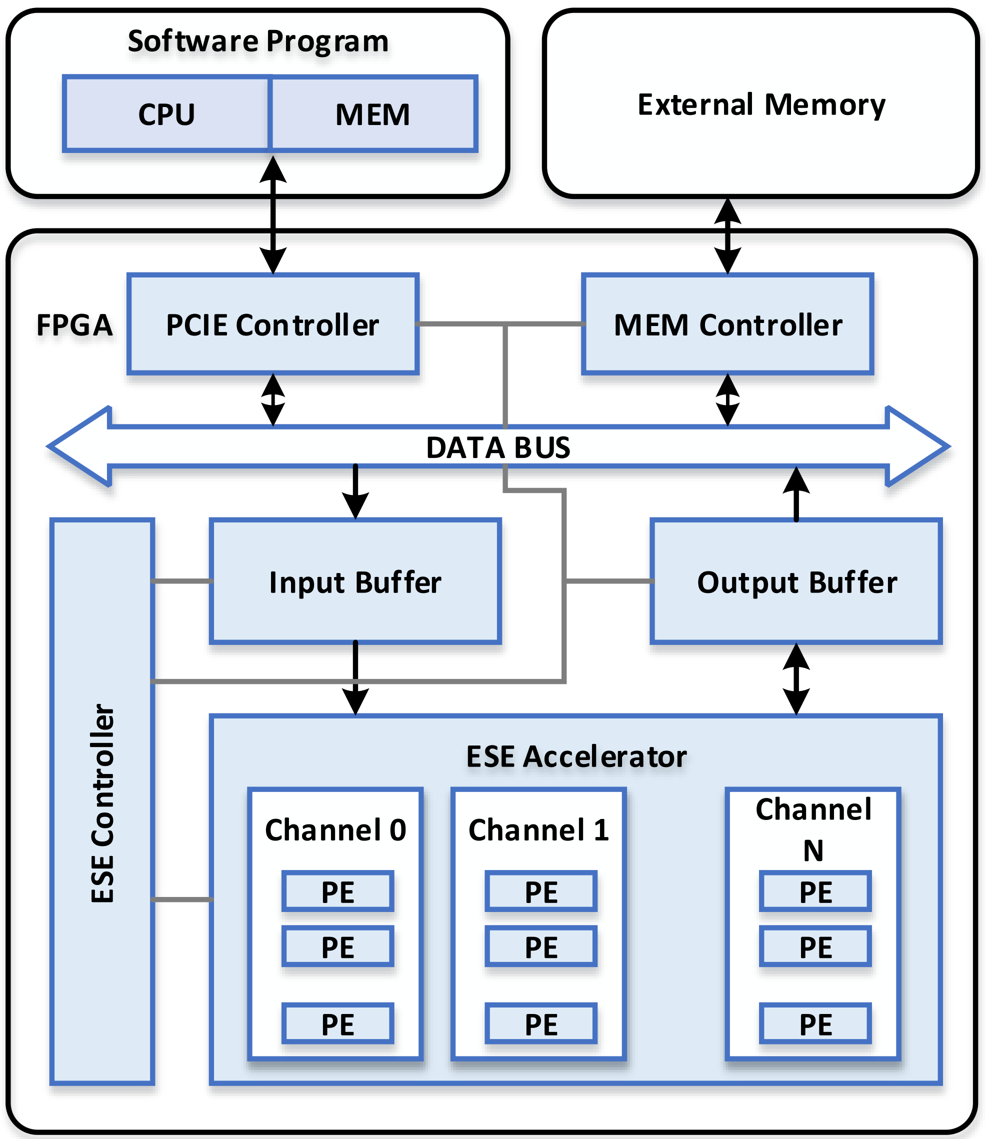


(b)

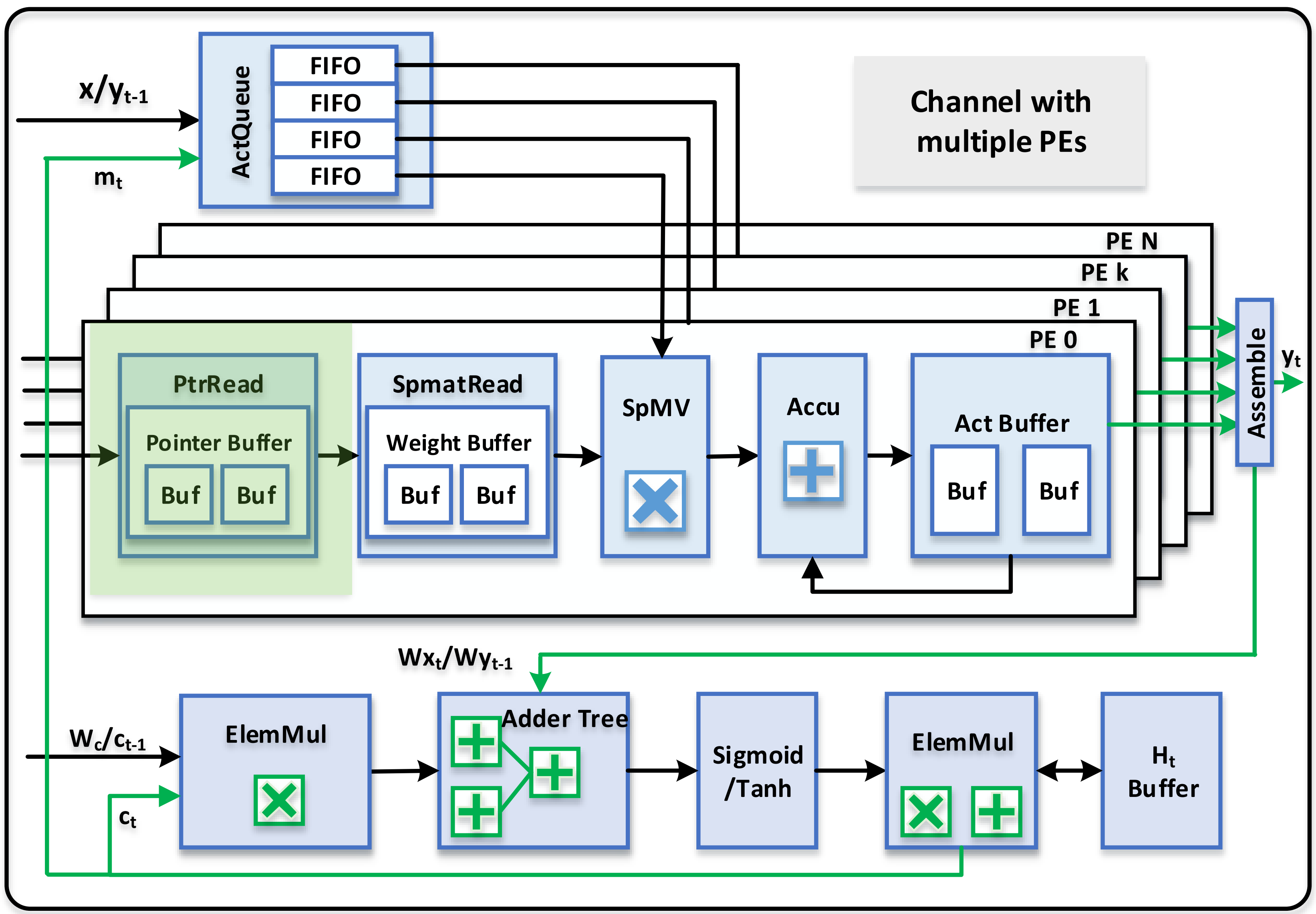
Hardware Architecture



Hardware Architecture

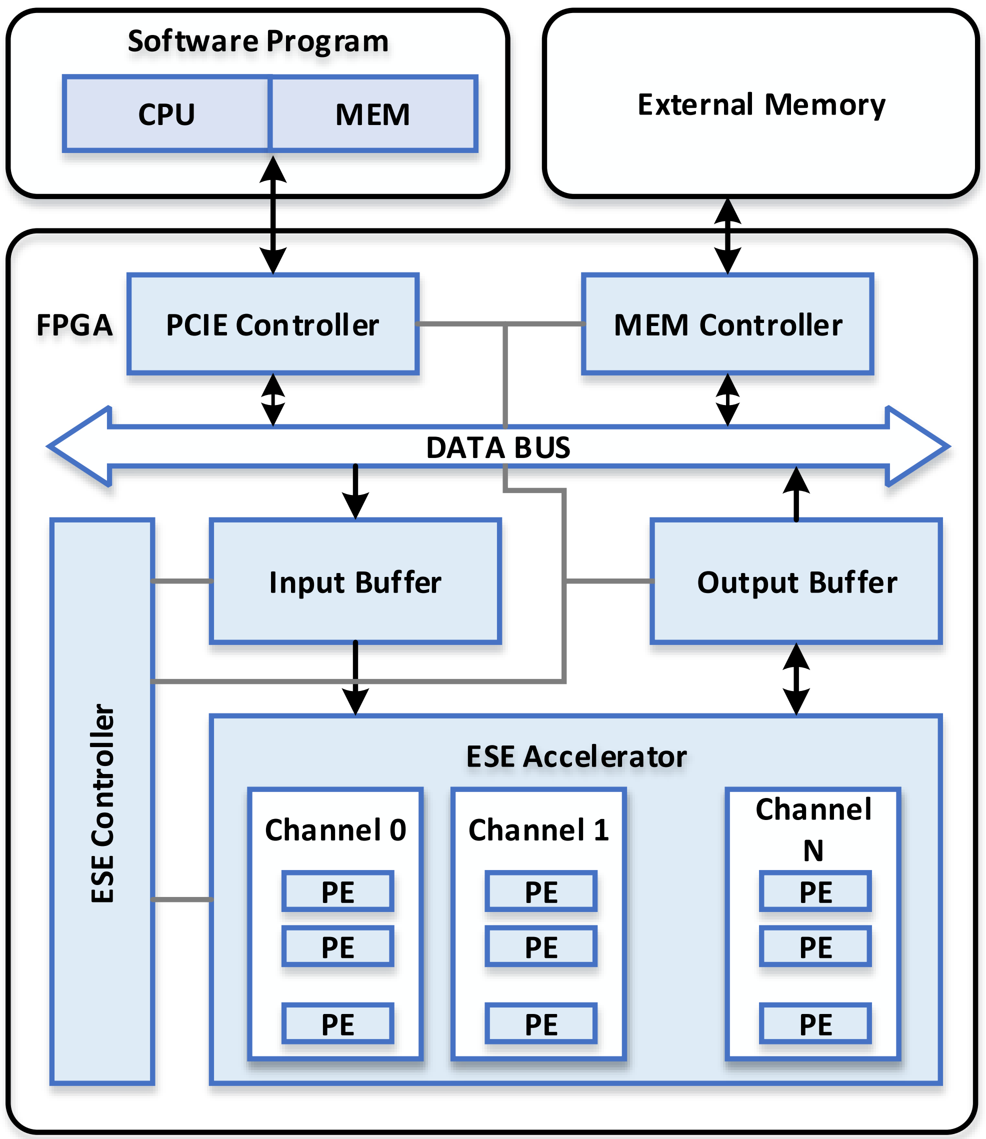


(a)

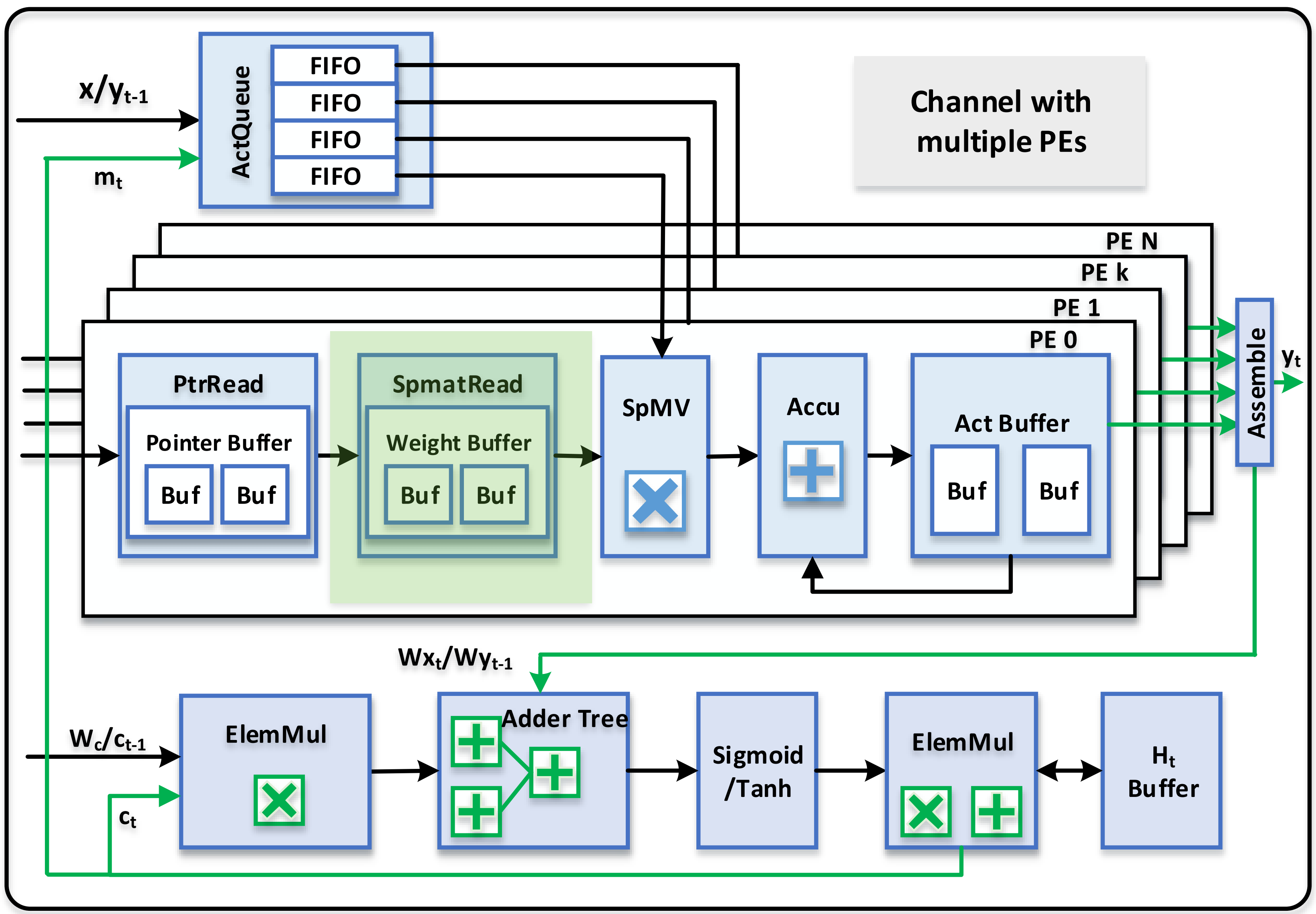


(b)

Hardware Architecture

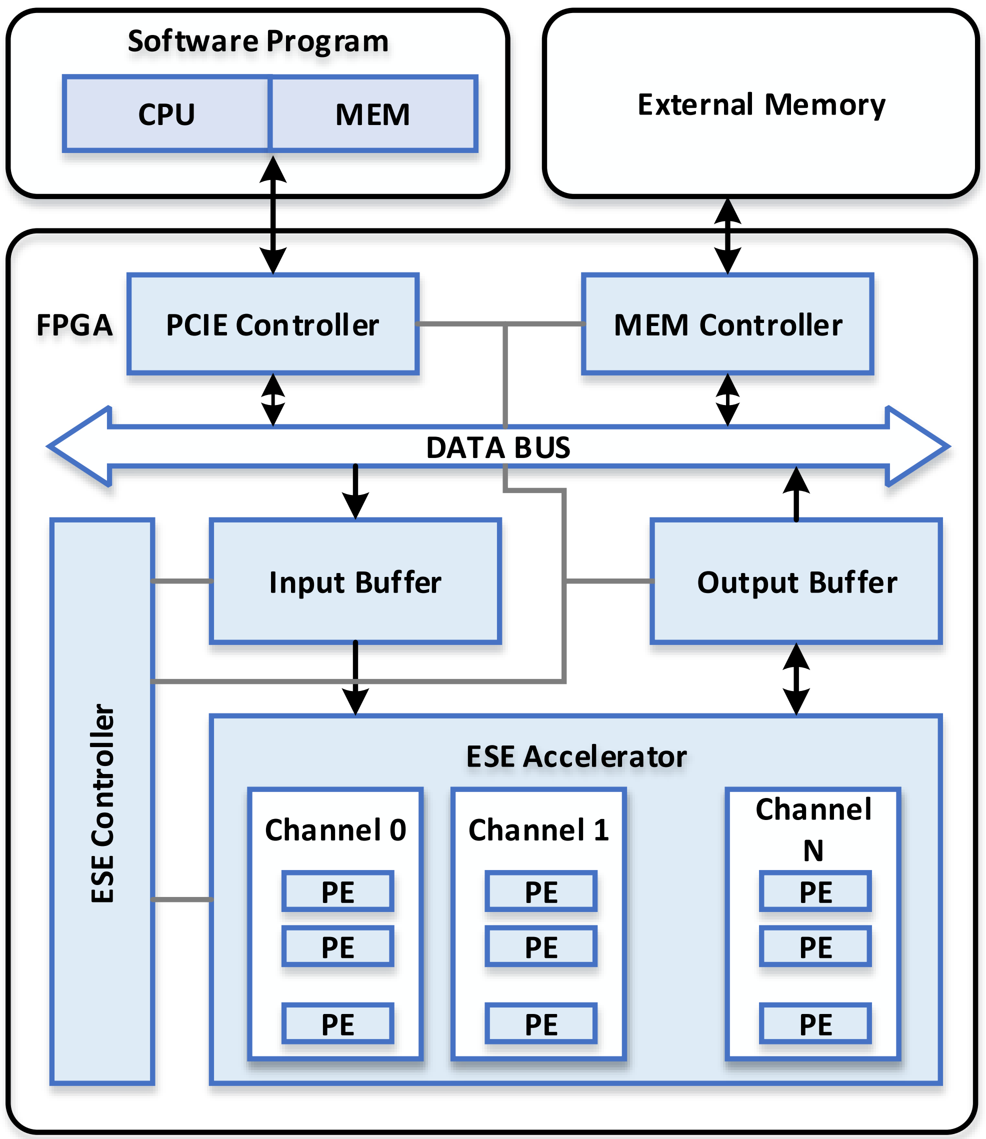


(a)

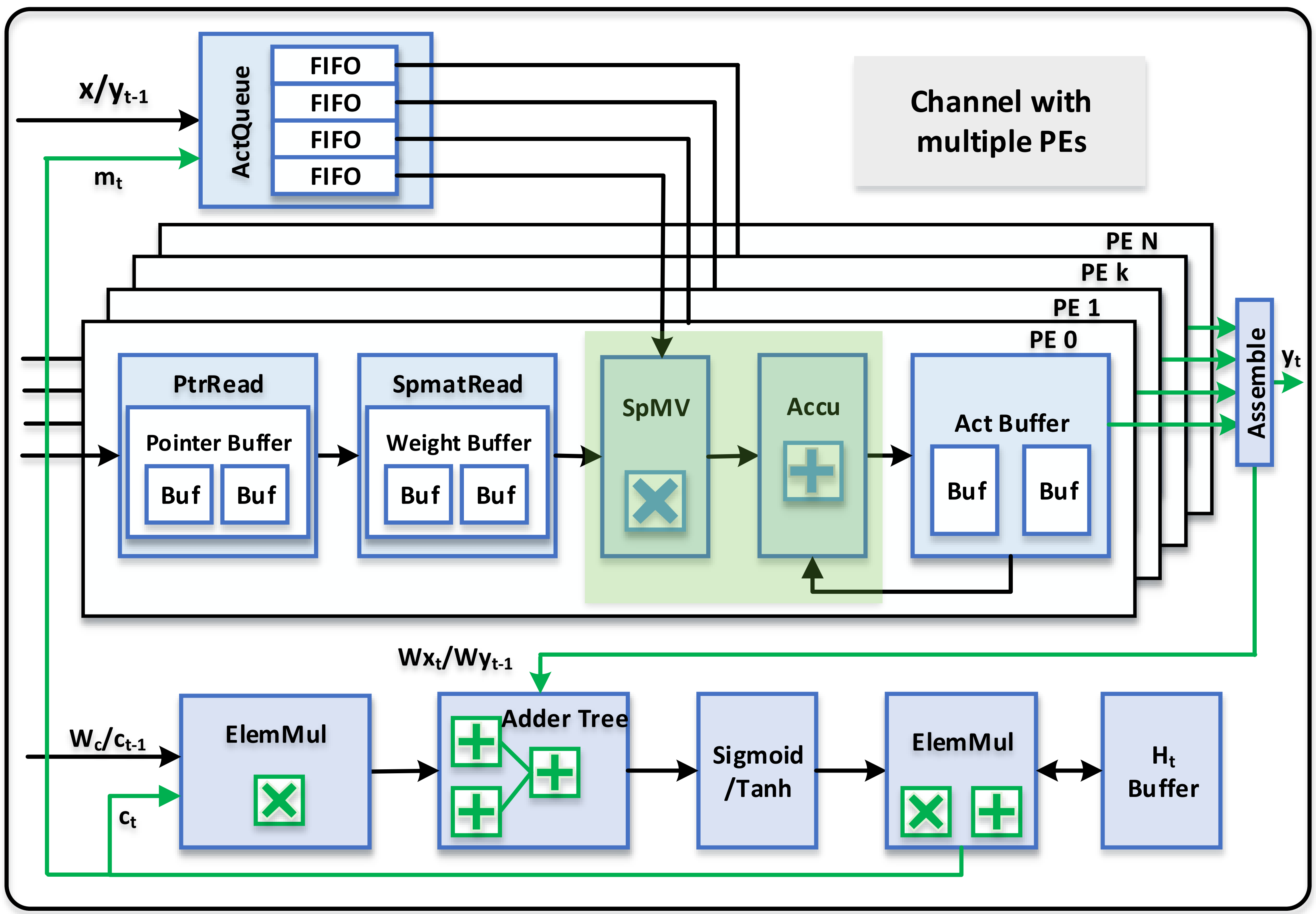


(b)

Hardware Architecture

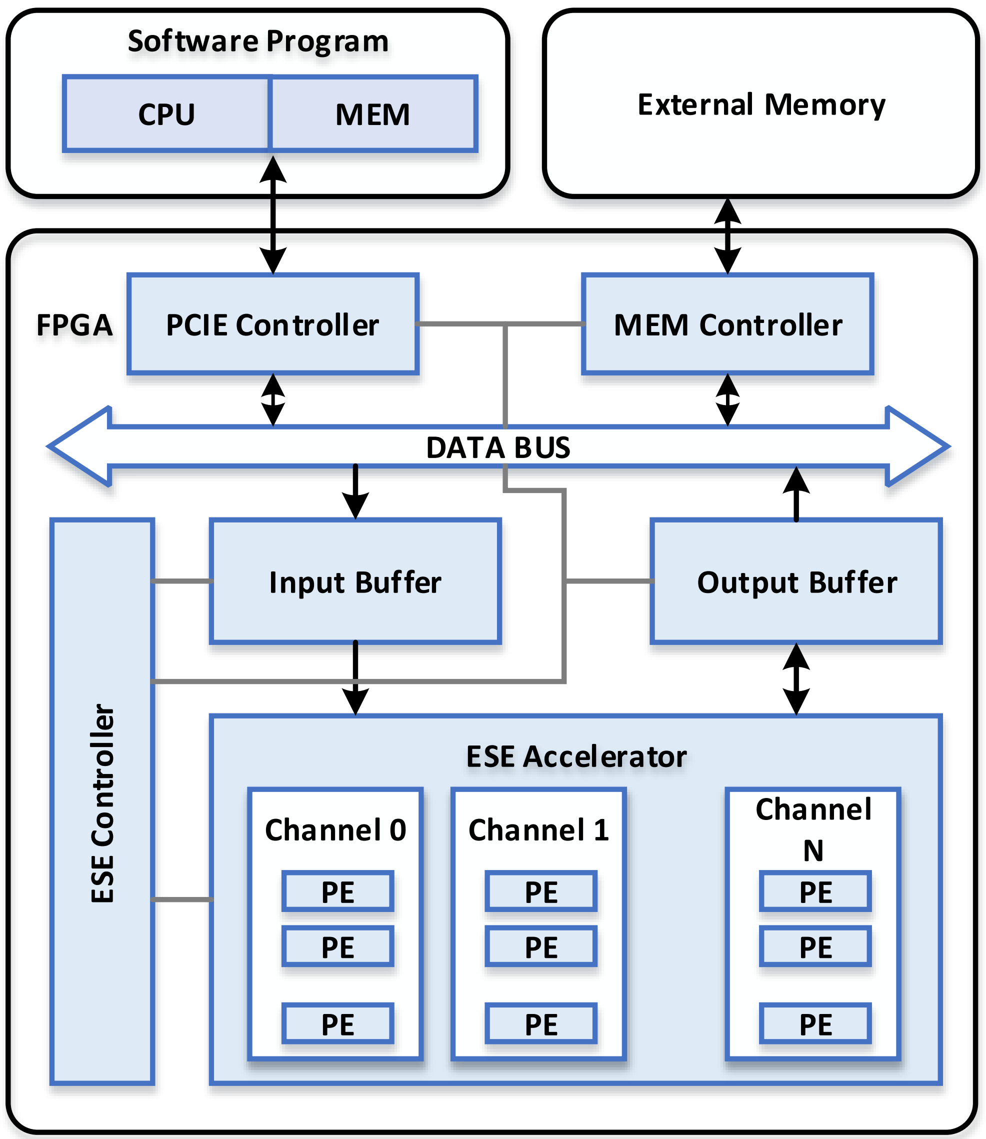


(a)

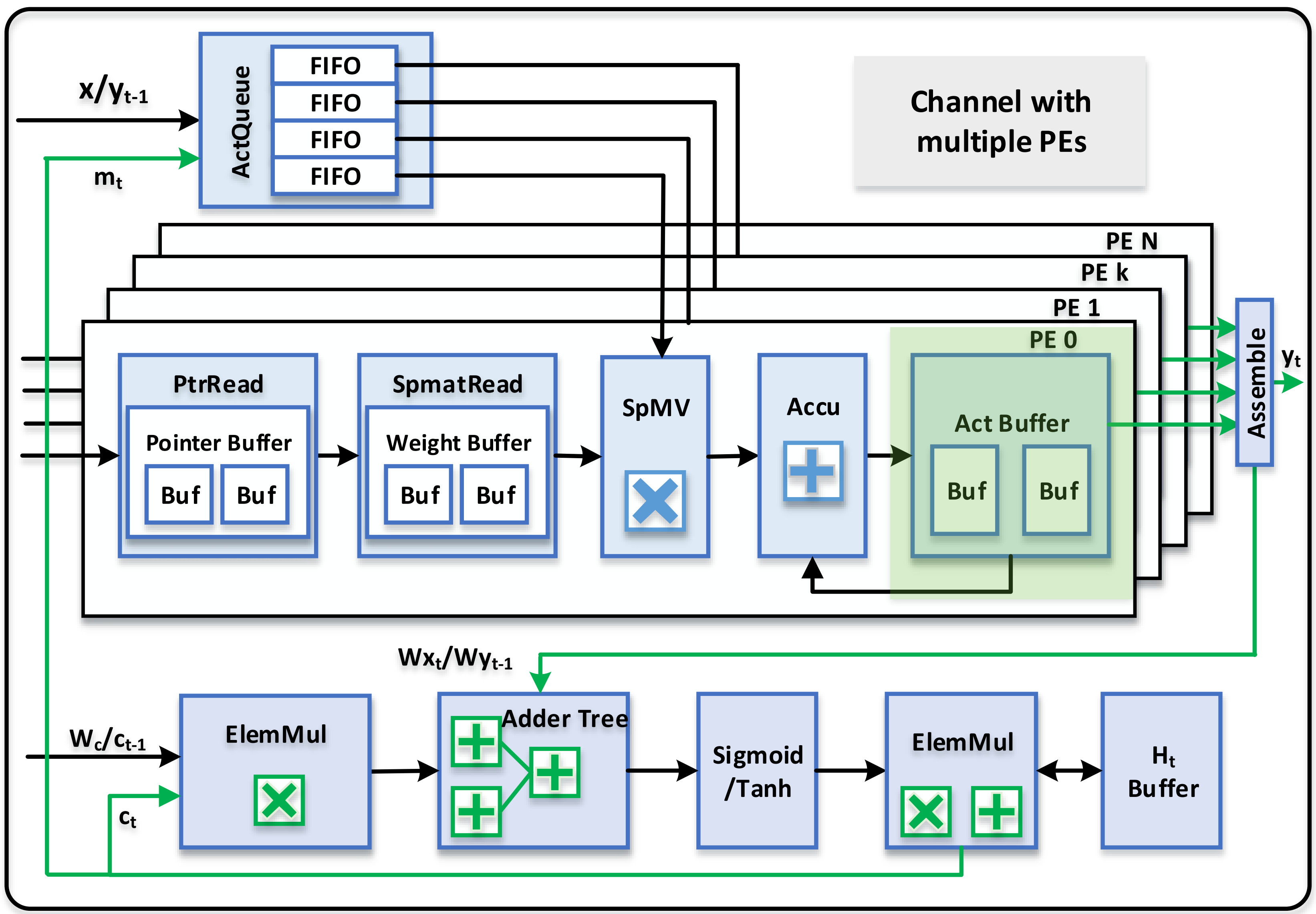


(b)

Hardware Architecture

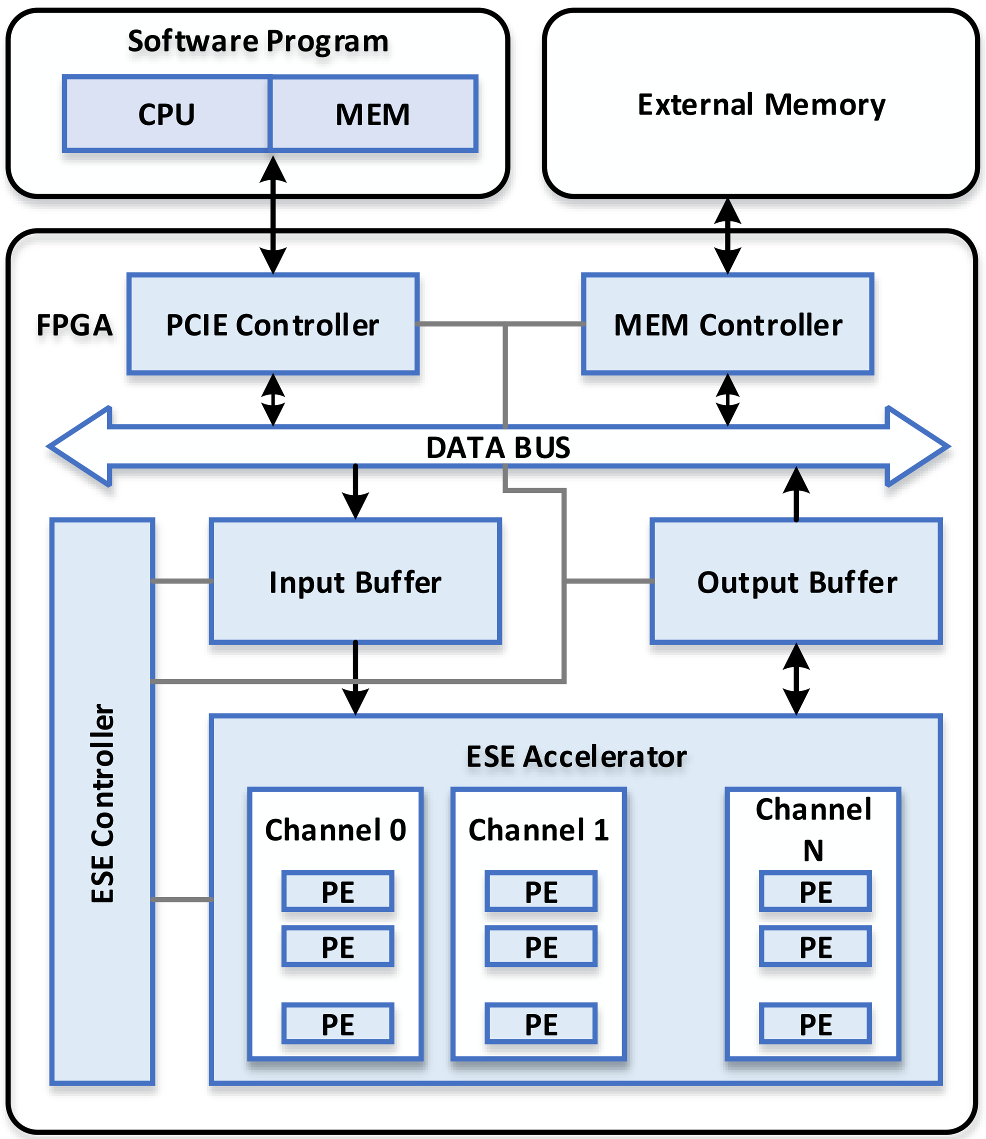


(a)

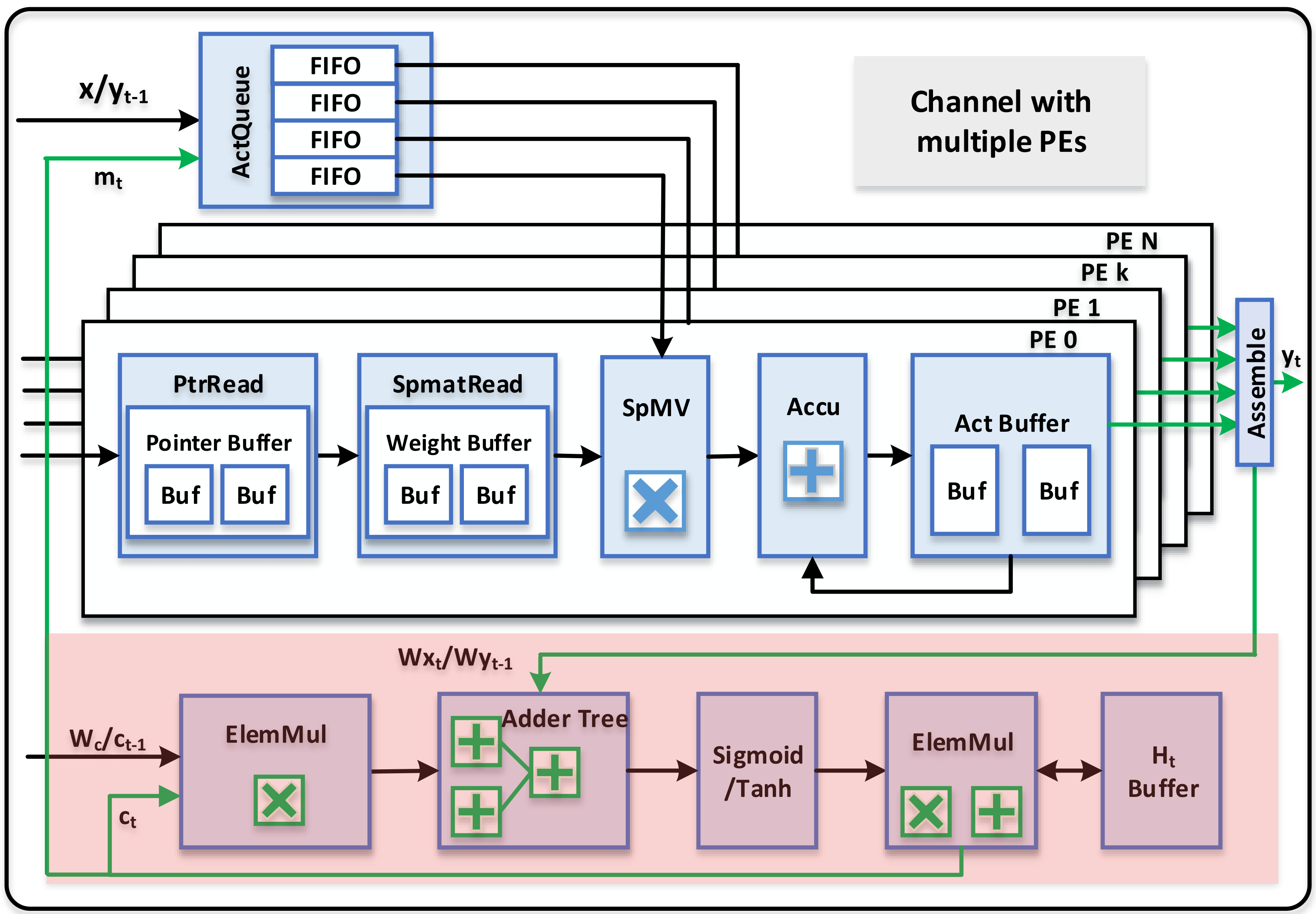


(b)

Hardware Architecture

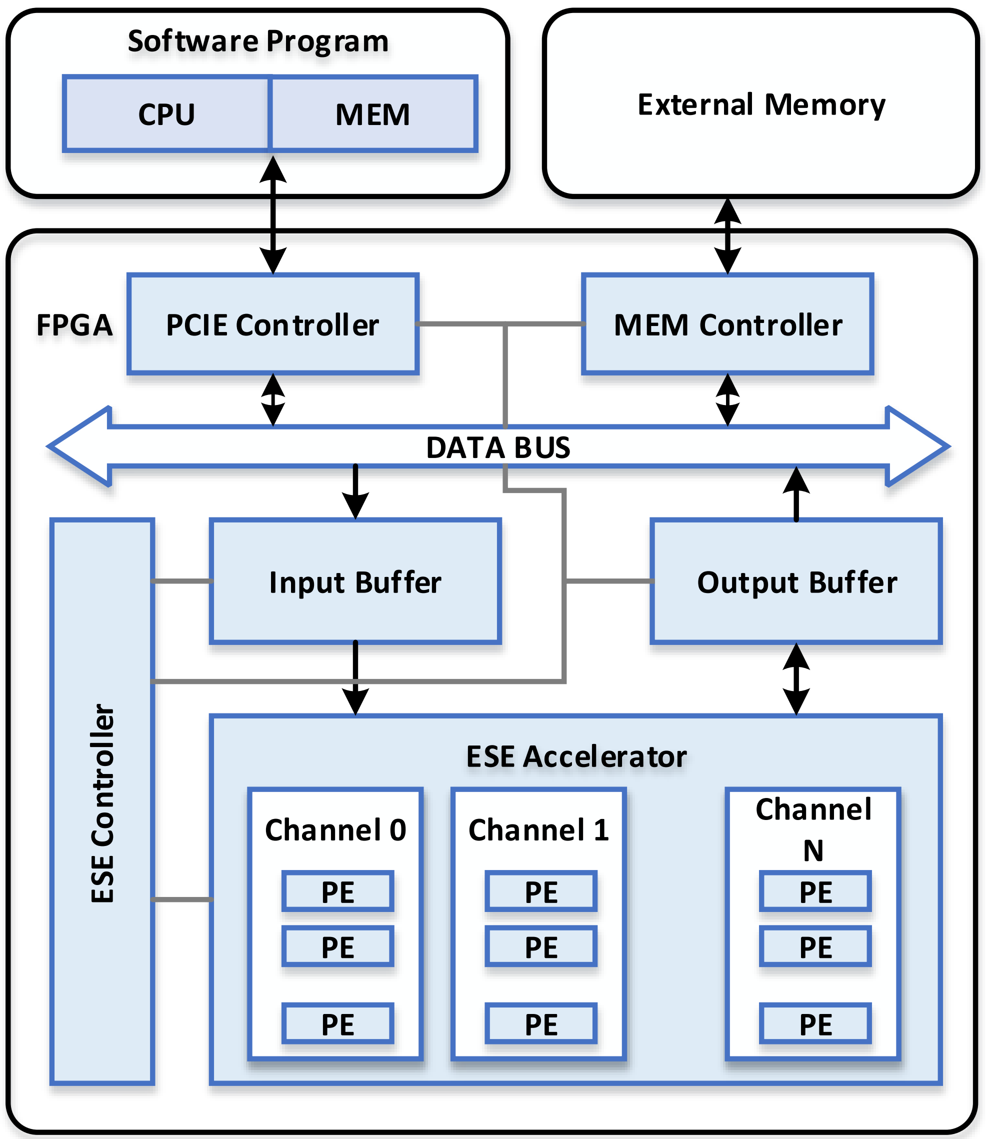


(a)

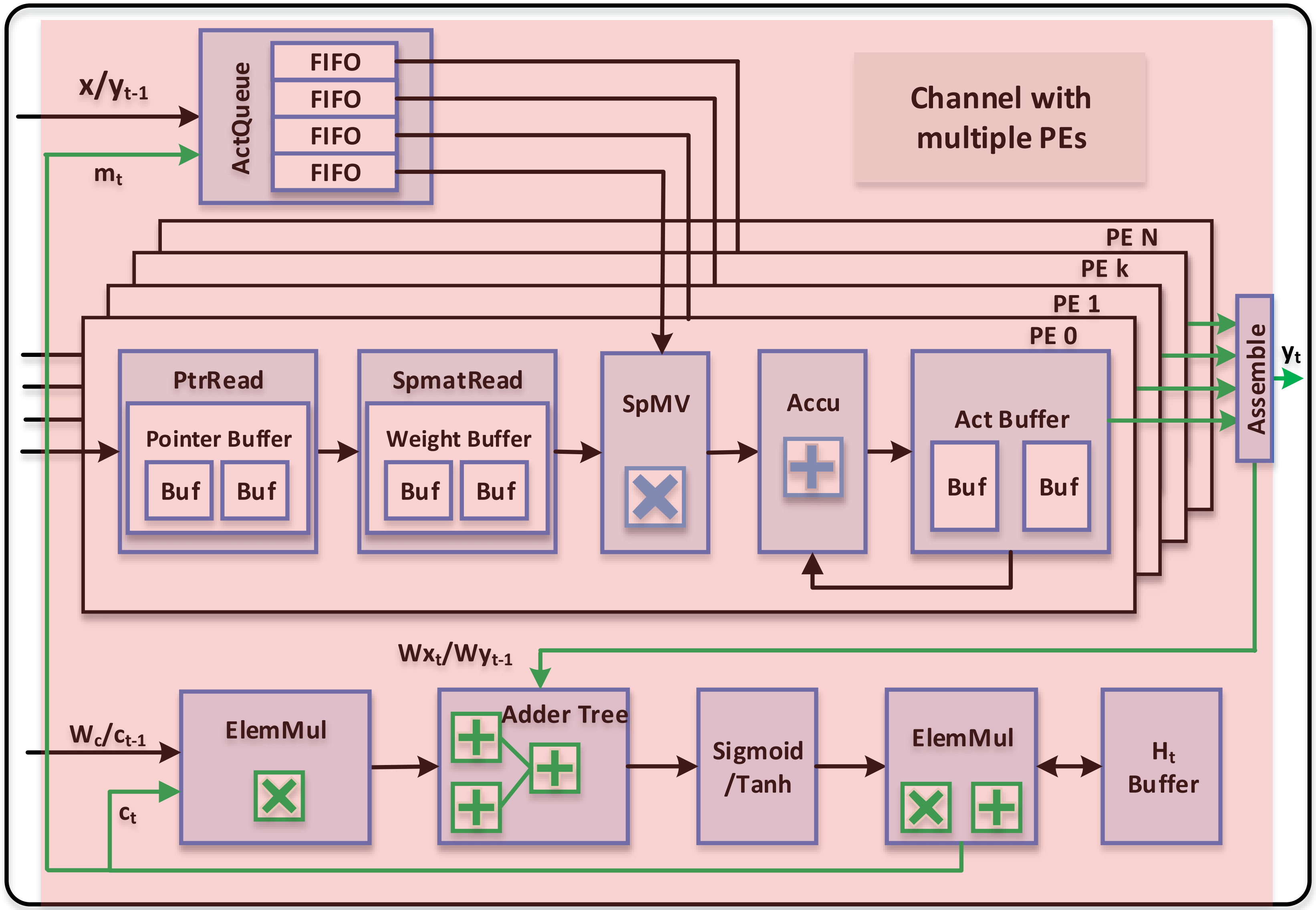


(b)

Hardware Architecture

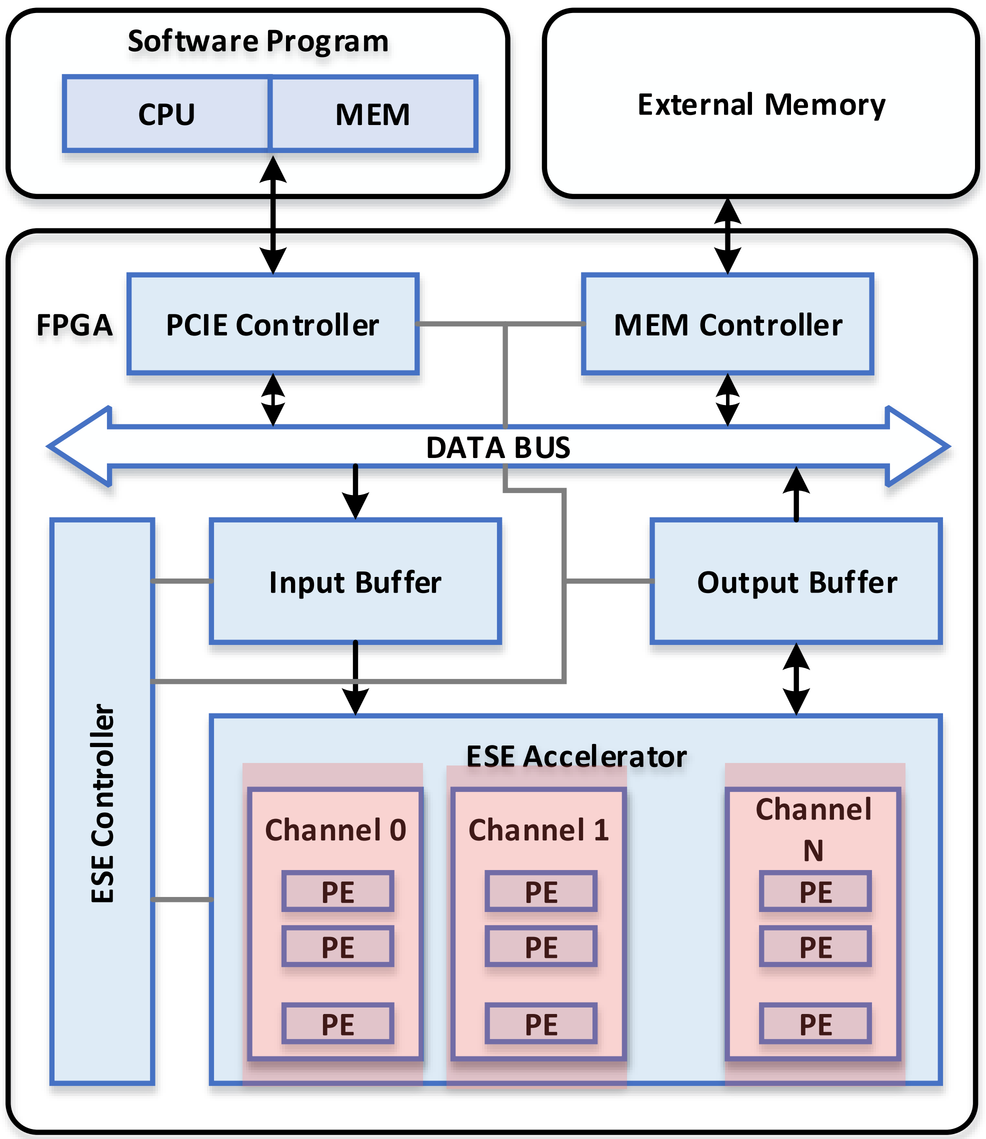


(a)

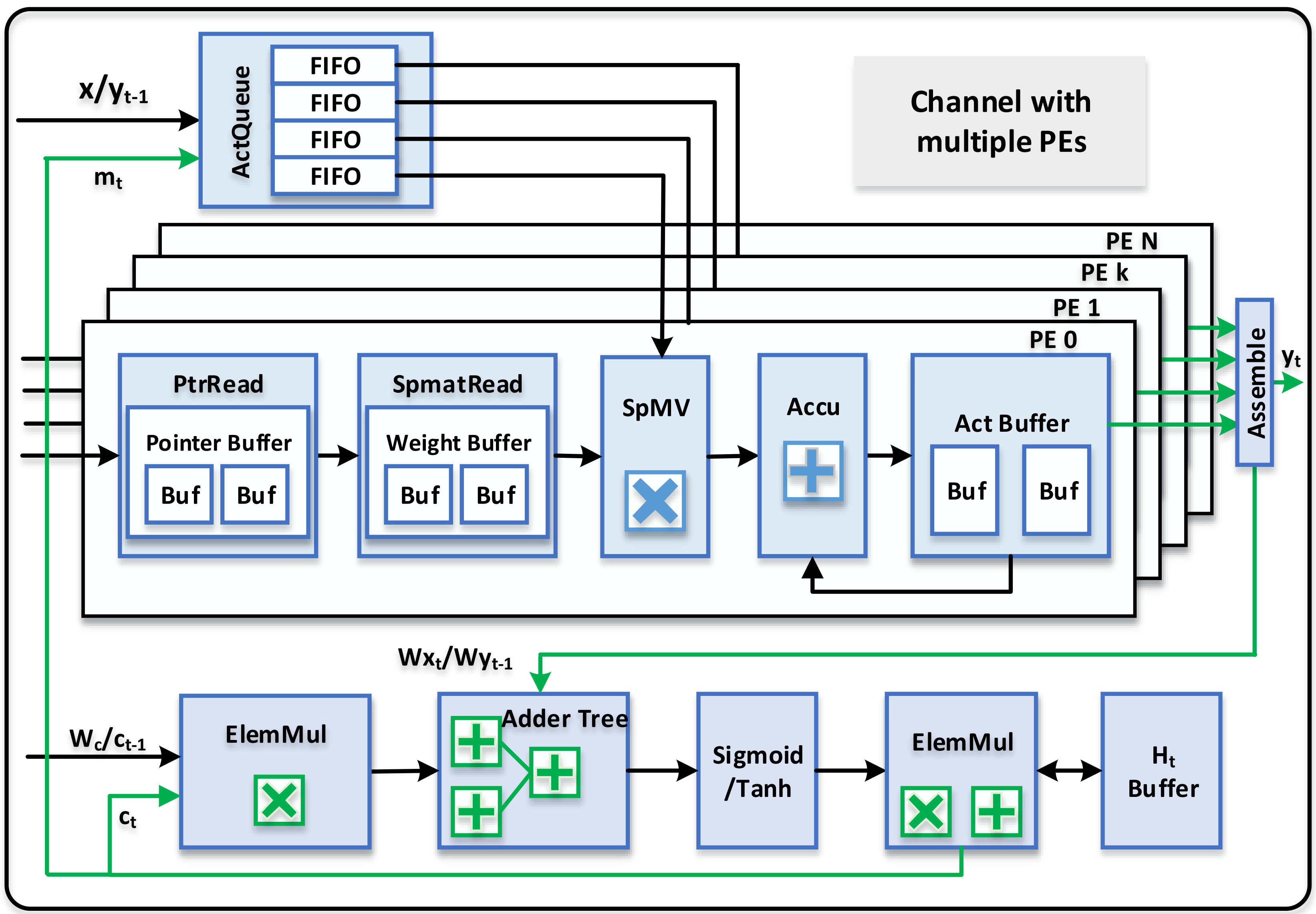


(b)

Hardware Architecture

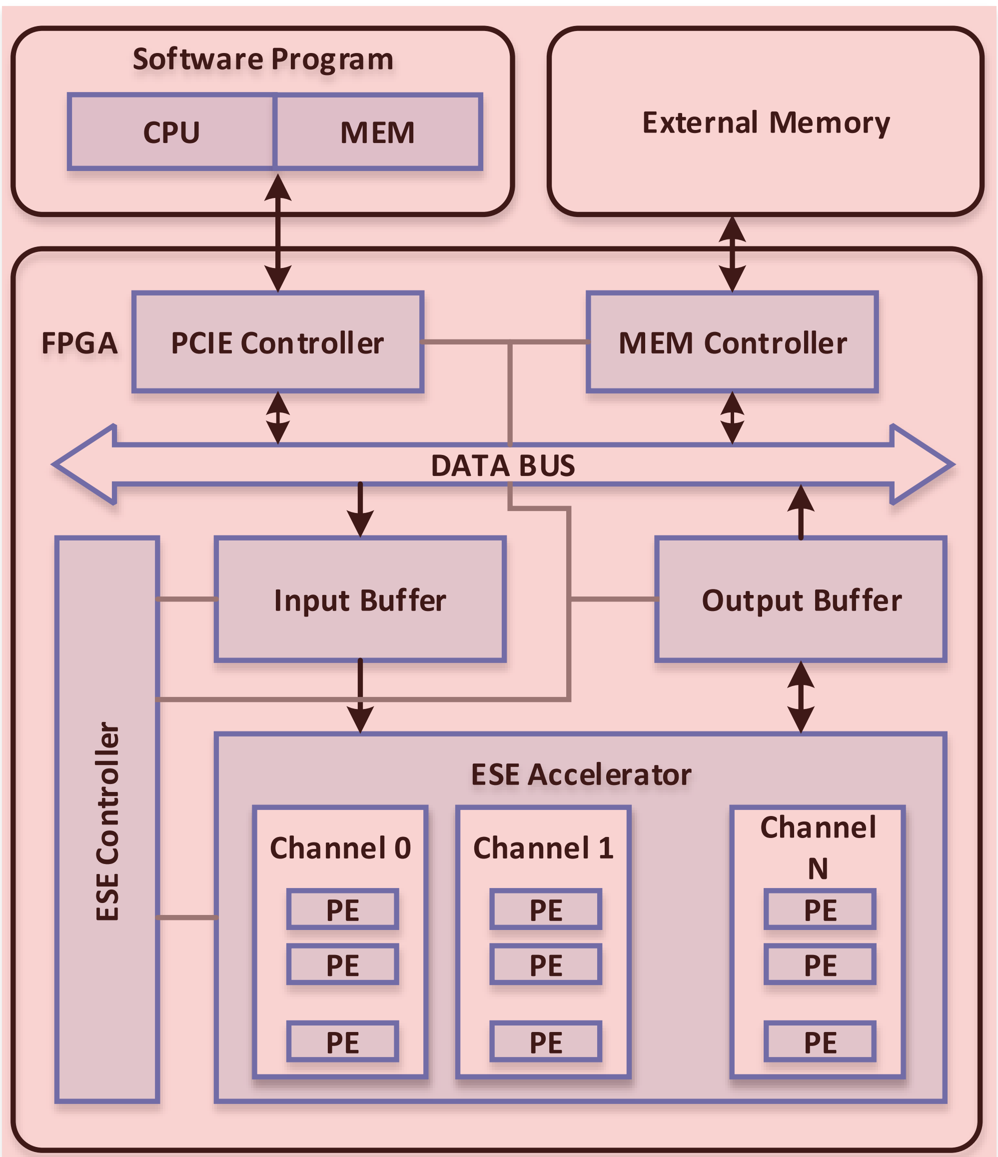


(a)

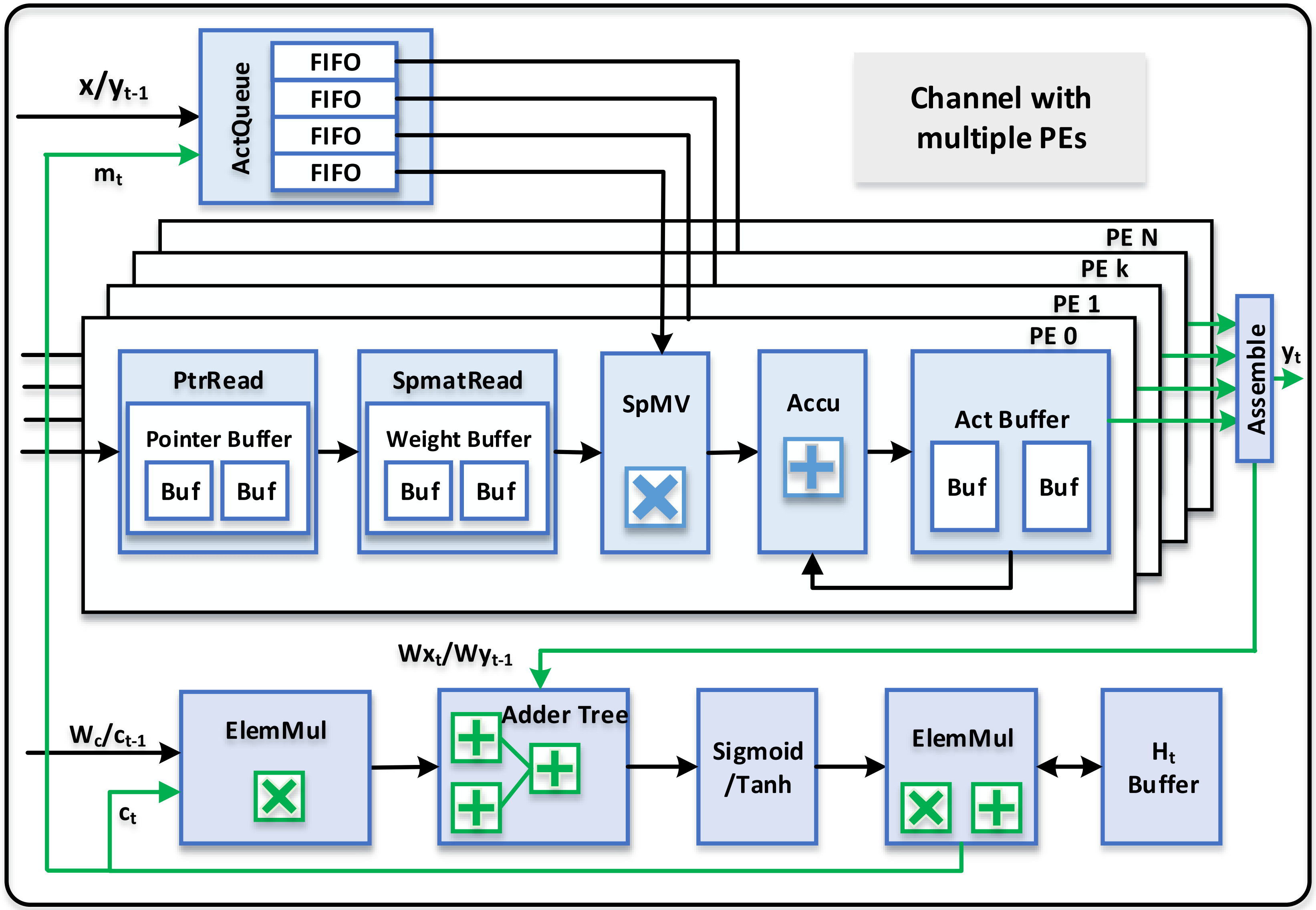


(b)

Hardware Architecture



(a)



(b)

Agenda

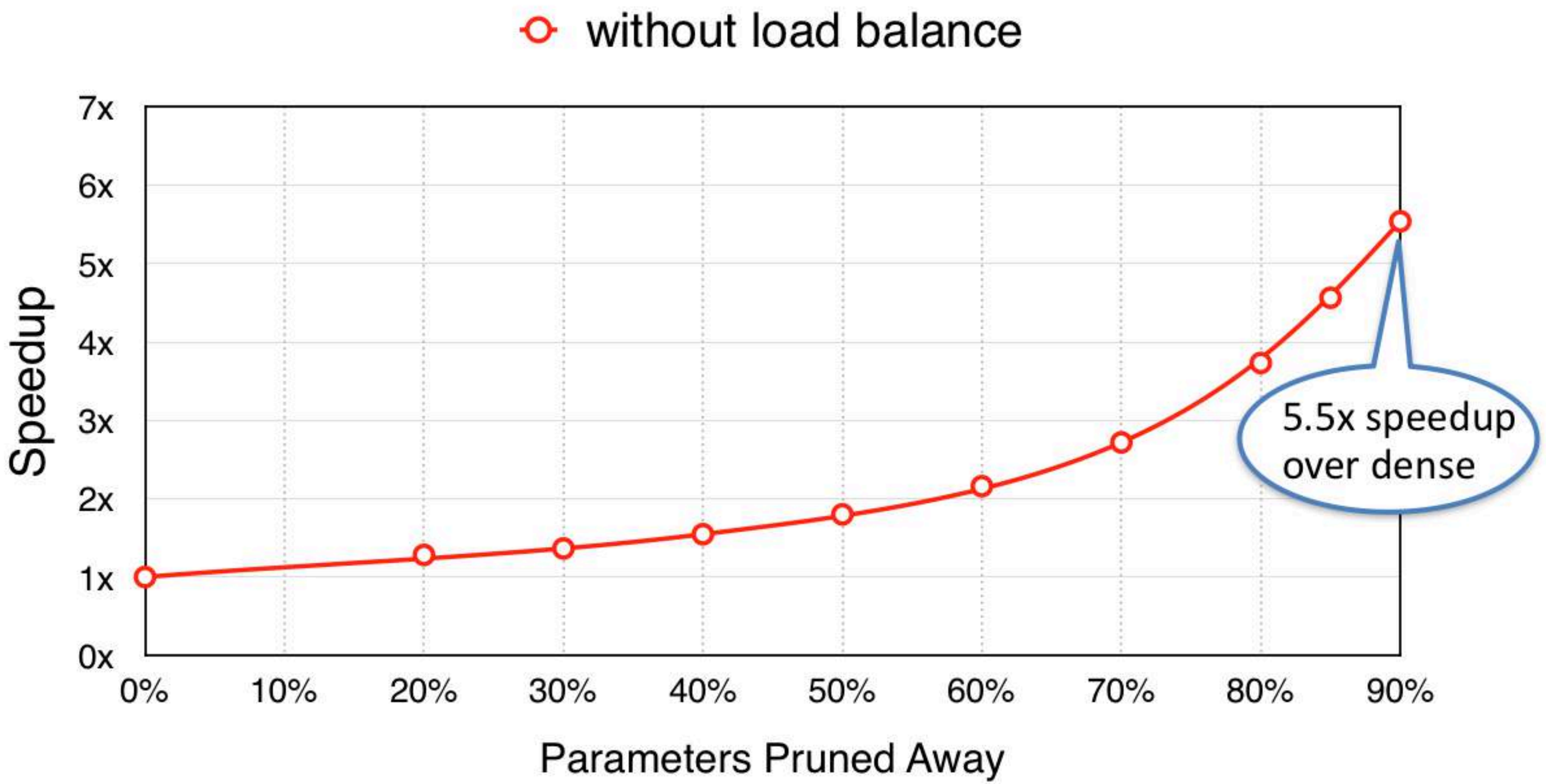
- **Compression**
  Load Balance-Aware Pruning

- **Scheduling**
  Overlap Computation and Memory Reference

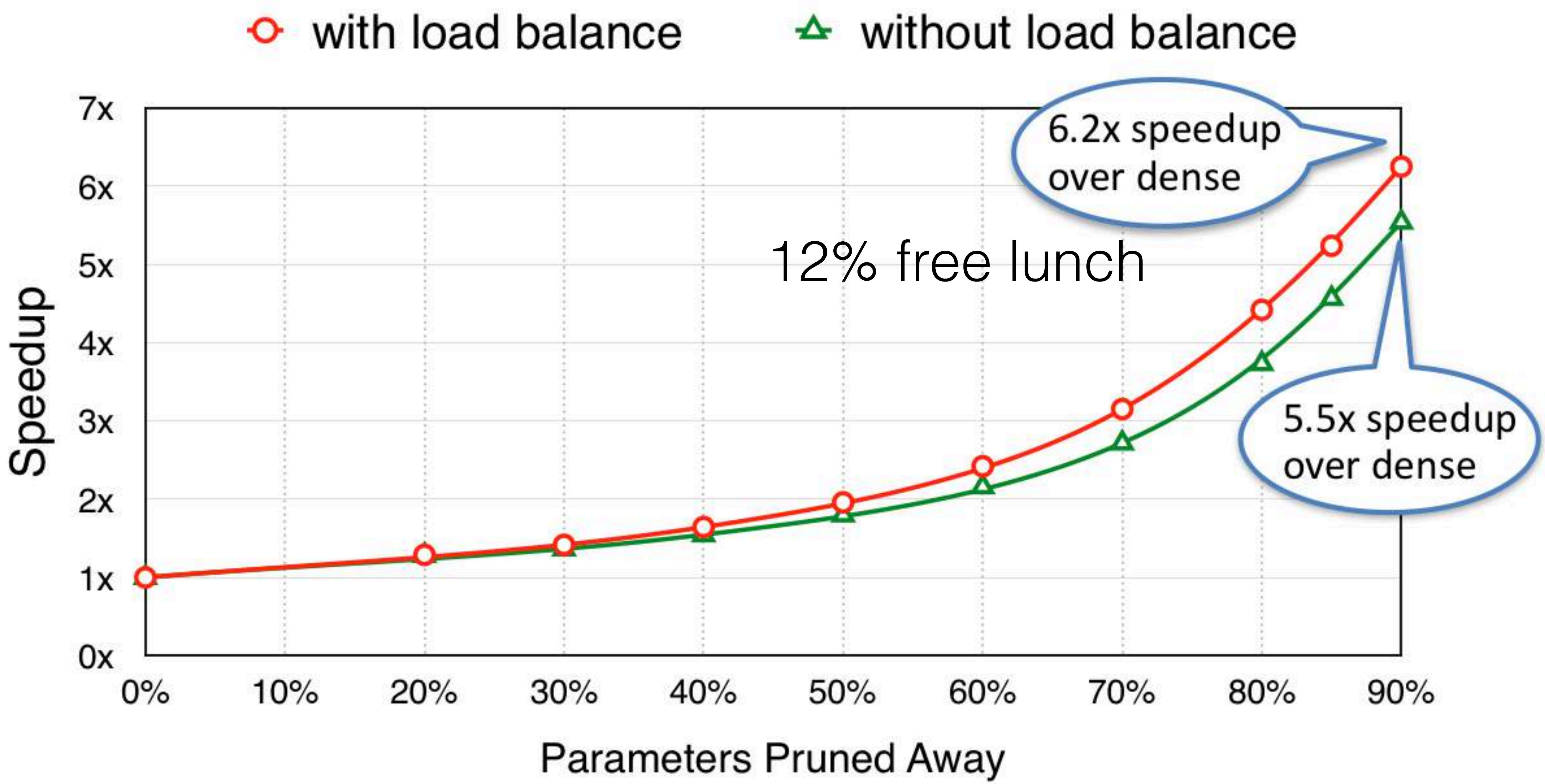
- **Accelerated Inference**
  Efficient Architecture for Sparse LSTM

- **Results**

Speedup vs Sparsity



Speedup vs Sparsity



Speedup and Energy Efficiency

Plat.	ESE on FPGA (ours)									CPU		GPU	
Matrix	Matrix Size	Sparsity (%)[1]	Compres. Matrix (Bytes)[2]	Theoreti. Comput. Time (µs)	Real Comput. Time (µs)	Total Operat. (GOP)	Real Perform. (GOP/s)	Equ. Operat. (GOP)	Equ. Perform. (GOP/s)	Real Comput. Time (µs)		Real Comput. Time (µs)	
										Dense	Sparse	Dense	Sparse
W_{ix}	1024×153	11.7	18304	2.9	**5.36**	0.0012	218.6	0.010	1870.7	**1518.4**[3]	**670.4**	**34.2**	**58.0**
W_{fx}	1024×153	11.7	18272	2.9	**5.36**	0.0012	218.2	0.010	1870.7				
W_{cx}	1024×153	11.8	18560	2.9	**5.36**	0.0012	221.6	0.010	1870.7				
W_{ox}	1024×153	11.5	17984	2.8	**5.36**	0.0012	214.7	0.010	1870.7				
W_{ir}	1024×512	11.3	59360	9.3	**10.31**	0.0038	368.5	0.034	3254.6	**3225.0**[4]	**2288.0**	**81.3**	**166.0**
W_{fr}	1024×512	11.5	60416	9.4	**10.01**	0.0039	386.3	0.034	3352.1				
W_{cr}	1024×512	11.2	58880	9.2	**9.89**	0.0038	381.2	0.034	3394.5				
W_{or}	1024×512	11.5	60128	9.4	**10.04**	0.0038	383.5	0.034	3343.7				
W_{ym}	512×1024	10.0	52416	8.2	**15.66**	0.0034	214.2	0.034	2142.7	1273.9	611.5	124.8	63.4
Total	**3248128**	**11.2**	**364320**	**57.0**	**82.7**	**0.0233**	**282.2**	**0.208**	**2515.7**	**6017.3**	**3569.9**	**240.3**	**287.4**

	ESE	CPU		GPU	
		Dense	Sparse	Dense	Sparse
Latency	82.7us	6017us	3569us	240us	287us
Power	41W	111W	38W	202W	136W
Performance	2.9x	0.039	0.067	1x	0.84
Energy Efficiency	14.3x	0.071	0.355	1x	1.25
Compression Ratio	20x	1	10	1x	10

Speedup and Energy Efficiency

Plat.	ESE on FPGA (ours)									CPU		GPU	
Matrix	Matrix Size	Sparsity (%)[1]	Compres. Matrix (Bytes)[2]	Theoreti. Comput. Time (µs)	Real Comput. Time (µs)	Total Operat. (GOP)	Real Perform. (GOP/s)	Equ. Operat. (GOP)	Equ. Perform. (GOP/s)	Real Comput. Time (µs) Dense	Real Comput. Time (µs) Sparse	Real Comput. Time (µs) Dense	Real Comput. Time (µs) Sparse
W_{ix}	1024×153	11.7	18304	2.9	**5.36**	0.0012	218.6	0.010	1870.7	**1518.4[3]**	**670.4**	**34.2**	**58.0**
W_{fx}	1024×153	11.7	18272	2.9	**5.36**	0.0012	218.2	0.010	1870.7				
W_{cx}	1024×153	11.8	18560	2.9	**5.36**	0.0012	221.6	0.010	1870.7				
W_{ox}	1024×153	11.5	17984	2.8	**5.36**	0.0012	214.7	0.010	1870.7				
W_{ir}	1024×512	11.3	59360	9.3	**10.31**	0.0038	368.5	0.034	3254.6	**3225.0[4]**	**2288.0**	**81.3**	**166.0**
W_{fr}	1024×512	11.5	60416	9.4	**10.01**	0.0039	386.3	0.034	3352.1				
W_{cr}	1024×512	11.2	58880	9.2	**9.89**	0.0038	381.2	0.034	3394.5				
W_{or}	1024×512	11.5	60128	9.4	**10.04**	0.0038	383.5	0.034	3343.7				
W_{ym}	512×1024	10.0	52416	8.2	**15.66**	0.0034	214.2	0.034	2142.7	1273.9	611.5	124.8	63.4
Total	**3248128**	**11.2**	**364320**	**57.0**	**82.7**	**0.0233**	**282.2**	**0.208**	**2515.7**	**6017.3**	**3569.9**	**240.3**	**287.4**

	ESE	CPU Dense	CPU Sparse	GPU Dense	GPU Sparse
Latency	82.7us	6017us	3569us	240us	287us
Power	41W	111W	38W	202W	136W
Performance	2.9x	0.039	0.067	1x	0.84
Energy Efficiency	14.3x	0.071	0.355	1x	1.25
Compression Ratio	20x	1	10	1x	10

Speedup and Energy Efficiency

Plat.	ESE on FPGA (ours)									CPU		GPU	
Matrix	Matrix Size	Sparsity (%)[1]	Compres. Matrix (Bytes)[2]	Theoreti. Comput. Time (µs)	Real Comput. Time (µs)	Total Operat. (GOP)	Real Perform. (GOP/s)	Equ. Operat. (GOP)	Equ. Perform. (GOP/s)	Real Comput. Time (µs)		Real Comput. Time (µs)	
										Dense	Sparse	Dense	Sparse
W_{ix}	1024×153	11.7	18304	2.9	**5.36**	0.0012	218.6	0.010	1870.7	**1518.4**[3]	**670.4**	**34.2**	**58.0**
W_{fx}	1024×153	11.7	18272	2.9	**5.36**	0.0012	218.2	0.010	1870.7				
W_{cx}	1024×153	11.8	18560	2.9	**5.36**	0.0012	221.6	0.010	1870.7				
W_{ox}	1024×153	11.5	17984	2.8	**5.36**	0.0012	214.7	0.010	1870.7				
W_{ir}	1024×512	11.3	59360	9.3	**10.31**	0.0038	368.5	0.034	3254.6	**3225.0**[4]	**2288.0**	**81.3**	**166.0**
W_{fr}	1024×512	11.5	60416	9.4	**10.01**	0.0039	386.3	0.034	3352.1				
W_{cr}	1024×512	11.2	58880	9.2	**9.89**	0.0038	381.2	0.034	3394.5				
W_{or}	1024×512	11.5	60128	9.4	**10.04**	0.0038	383.5	0.034	3343.7				
W_{ym}	512×1024	10.0	52416	8.2	**15.66**	0.0034	214.2	0.034	2142.7	1273.9	611.5	124.8	63.4
Total	**3248128**	**11.2**	**364320**	**57.0**	**82.7**	**0.0233**	**282.2**	**0.208**	**2515.7**	**6017.3**	**3569.9**	**240.3**	**287.4**

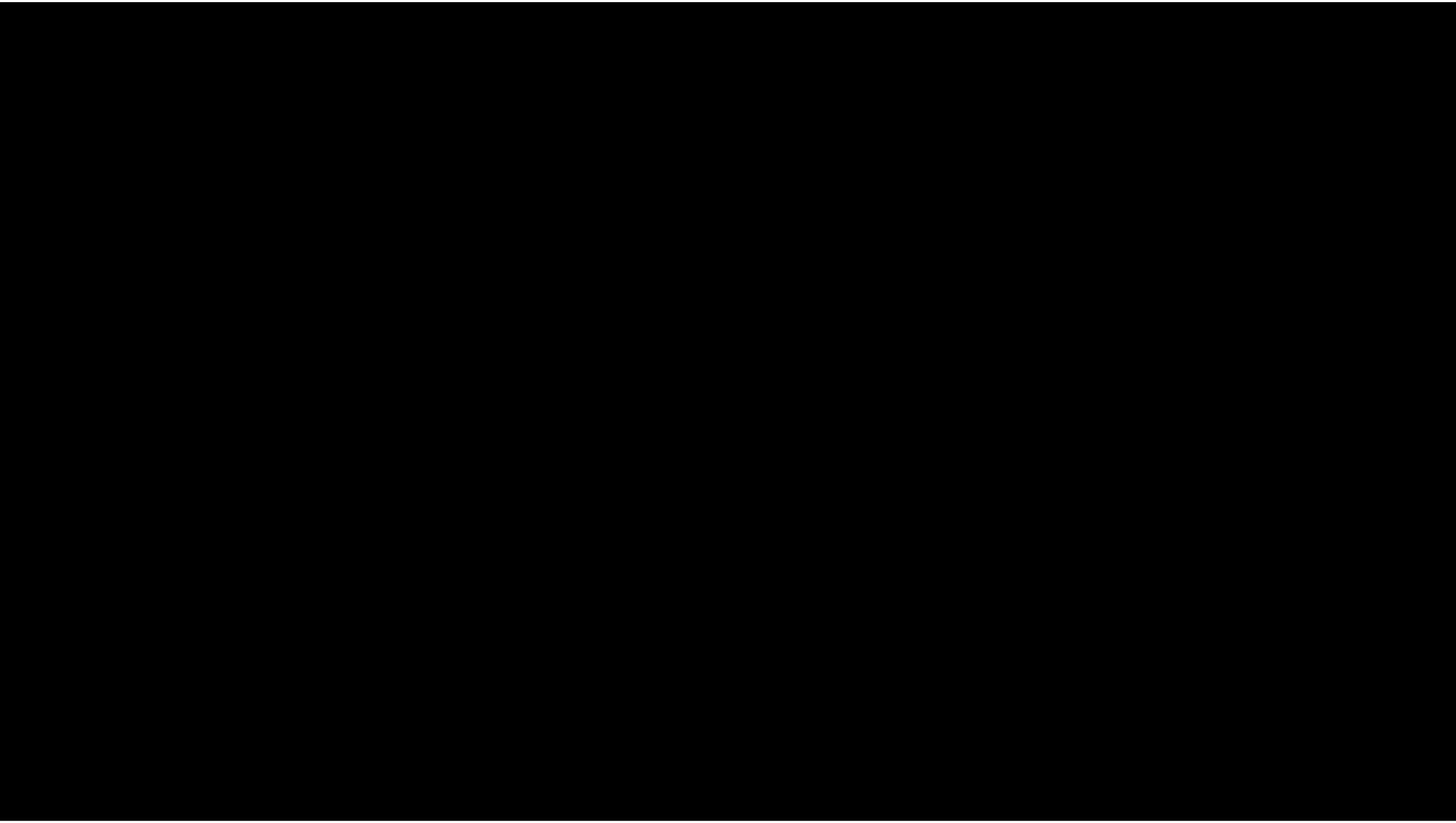
	ESE	CPU		GPU	
		Dense	Sparse	Dense	Sparse
Latency	82.7us	6017us	3569us	240us	287us
Power	41W	111W	38W	202W	136W
Performance	2.9x	0.039	0.067	1x	0.84
Energy Efficiency	14.3x	0.071	0.355	1x	1.25
Compression Ratio	20x	1	10	1x	10

Speedup and Energy Efficiency

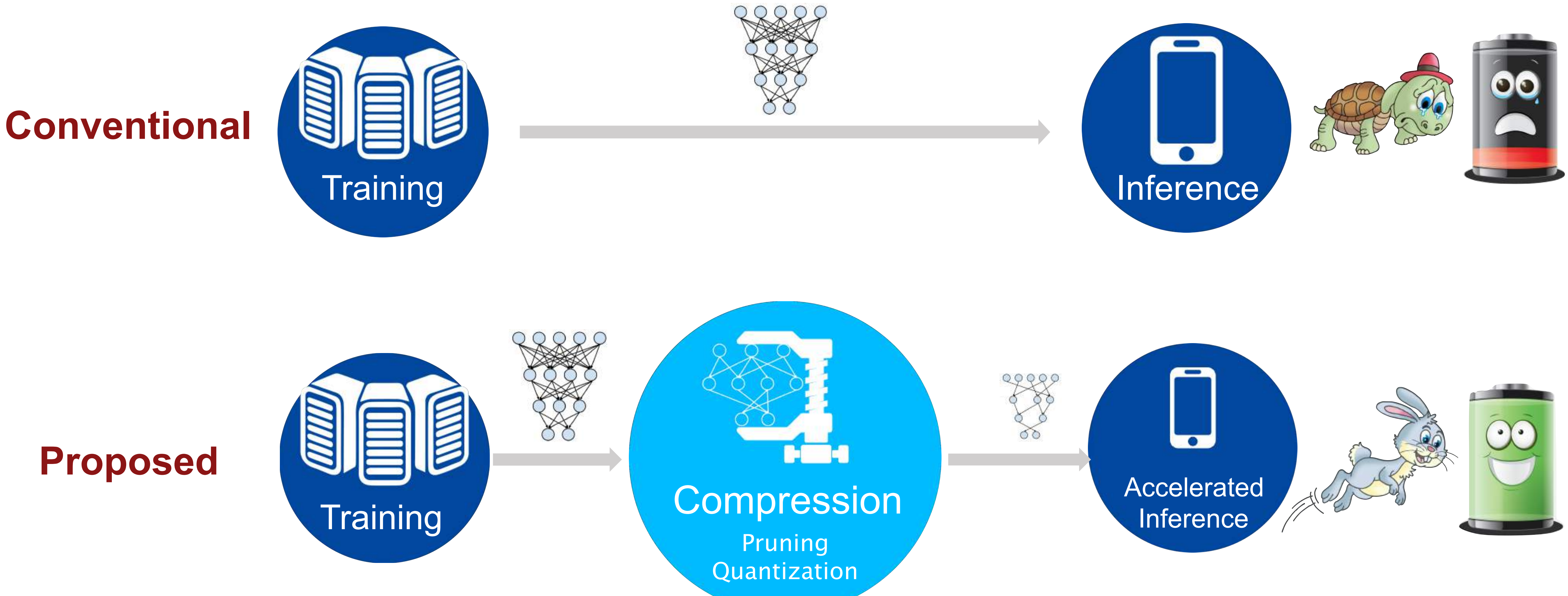
Plat.	ESE on FPGA (ours)									CPU		GPU	
Matrix	Matrix Size	Sparsity (%)[1]	Compres. Matrix (Bytes)[2]	Theoreti. Comput. Time (µs)	Real Comput. Time (µs)	Total Operat. (GOP)	Real Perform. (GOP/s)	Equ. Operat. (GOP)	Equ. Perform. (GOP/s)	Real Comput. Time (µs)		Real Comput. Time (µs)	
										Dense	Sparse	Dense	Sparse
W_{ix}	1024×153	11.7	18304	2.9	**5.36**	0.0012	218.6	0.010	1870.7	**1518.4[3]**	**670.4**	**34.2**	**58.0**
W_{fx}	1024×153	11.7	18272	2.9	**5.36**	0.0012	218.2	0.010	1870.7				
W_{cx}	1024×153	11.8	18560	2.9	**5.36**	0.0012	221.6	0.010	1870.7				
W_{ox}	1024×153	11.5	17984	2.8	**5.36**	0.0012	214.7	0.010	1870.7				
W_{ir}	1024×512	11.3	59360	9.3	**10.31**	0.0038	368.5	0.034	3254.6	**3225.0[4]**	**2288.0**	**81.3**	**166.0**
W_{fr}	1024×512	11.5	60416	9.4	**10.01**	0.0039	386.3	0.034	3352.1				
W_{cr}	1024×512	11.2	58880	9.2	**9.89**	0.0038	381.2	0.034	3394.5				
W_{or}	1024×512	11.5	60128	9.4	**10.04**	0.0038	383.5	0.034	3343.7				
W_{ym}	512×1024	10.0	52416	8.2	**15.66**	0.0034	214.2	0.034	2142.7	1273.9	611.5	124.8	63.4
Total	**3248128**	**11.2**	**364320**	**57.0**	**82.7**	**0.0233**	**282.2**	**0.208**	**2515.7**	**6017.3**	**3569.9**	**240.3**	**287.4**

	ESE	CPU		GPU	
		Dense	Sparse	Dense	Sparse
Latency	82.7us	6017us	3569us	240us	287us
Power	41W	111W	38W	202W	136W
Performance	2.9x	0.039	0.067	1x	0.84
Energy Efficiency	14.3x	0.071	0.355	1x	1.25
Compression Ratio	20x	1	10	1x	10

Demo



Thank you!



Han et al ICLR'17

Han et al NIPS'15
Han et al ICLR'16
(best paper award)

Han et al ISCA'16
Han et al FPGA'17
(best paper award)