2017 ACM/SIGDA International Symposium on
Field-Programmable Gate Arrays

Optimizing Loop Operation and Dataflow in FPGA Acceleration of Deep Convolutional Neural Networks

Yufei Ma, **Yu Cao**, **Sarma Vrudhula†**, **Jae-sun Seo**

School of Electrical, Computer and Energy Engineering

†School of Computing, Informatics, Decision Systems Engineering

Arizona State University, Tempe, USA

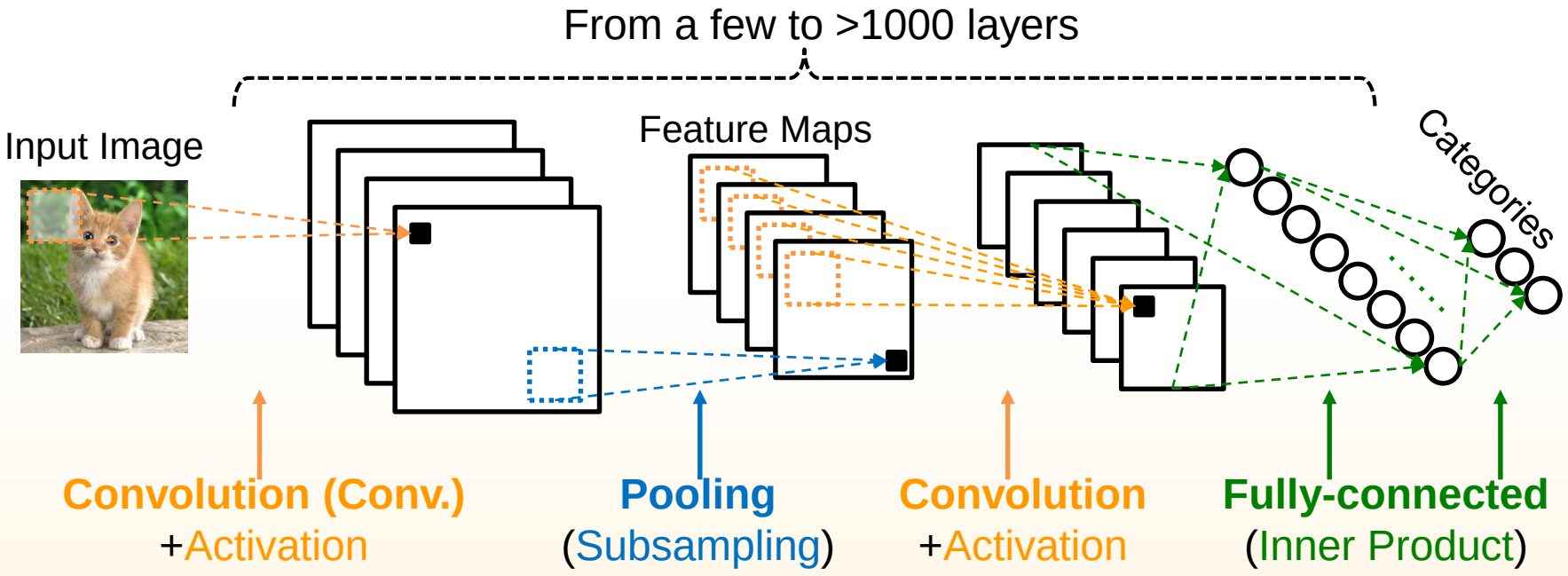
2017/02/22

Outline

- Overview of CNN Algorithm and Accelerator
- Convolution Loop Optimization
- Design Objectives of CNN Accelerator
- Loop Optimization in Related Works
- Proposed CNN Accelerator
- Experimental Results
- Conclusion

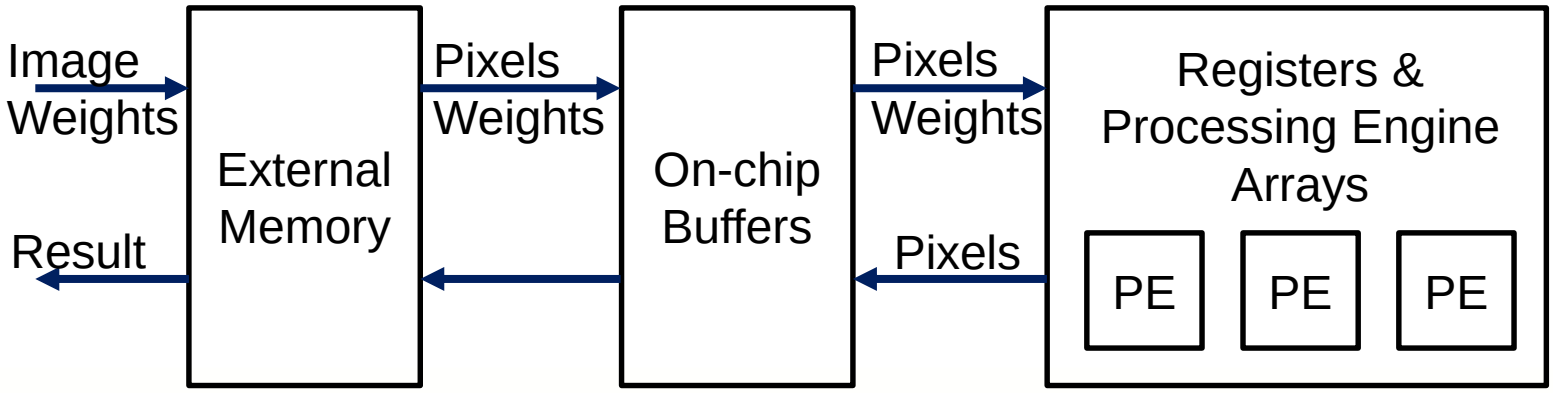
Convolutional Neural Networks (CNN)

- Dominant approach for recognition and detection tasks
- Require large # of operations (>1G) and parameters (>50M).
- High variability in the sizes of different convolution layers.
- Evolving rapidly with more layers to achieve higher accuracy.



General CNN Acceleration System

- Three levels of hardware accelerator hierarchy
  - External memory (DRAM, ~GB)
  - On-chip buffers (RAM, ~MB)
  - Registers and processing engines (PEs)

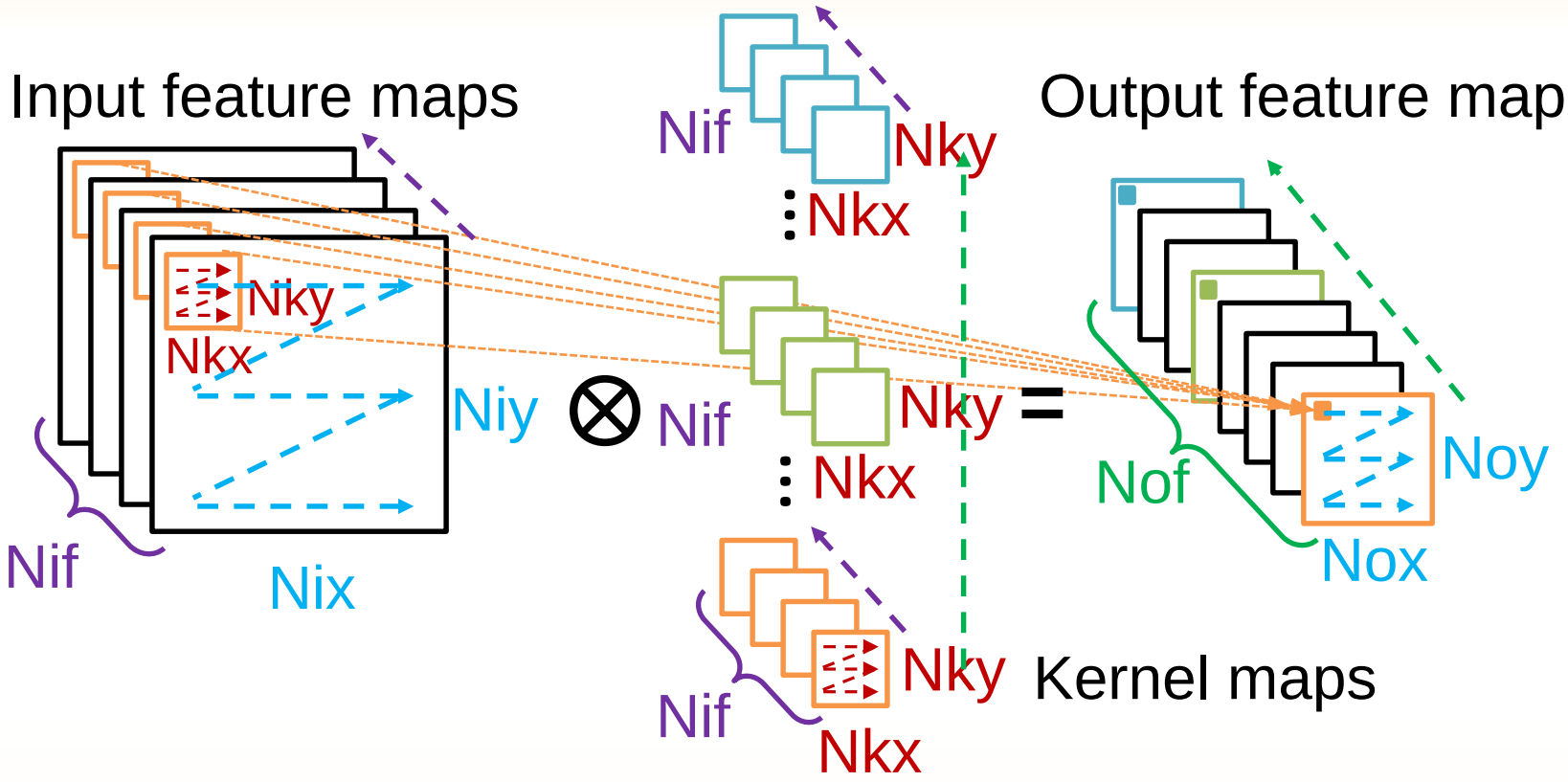


- Our contribution
  - In-depth analysis of convolution loop optimization techniques.
  - CNN accelerator with low communication and high performance

Outline

- Overview of CNN Algorithm and Accelerator
- Convolution Loop Optimization
  - Loop Unrolling
  - Loop Tiling
  - Loop Interchange
- Proposed CNN Accelerator
- Experimental Results
- Conclusion

Convolution Parameters and Loops

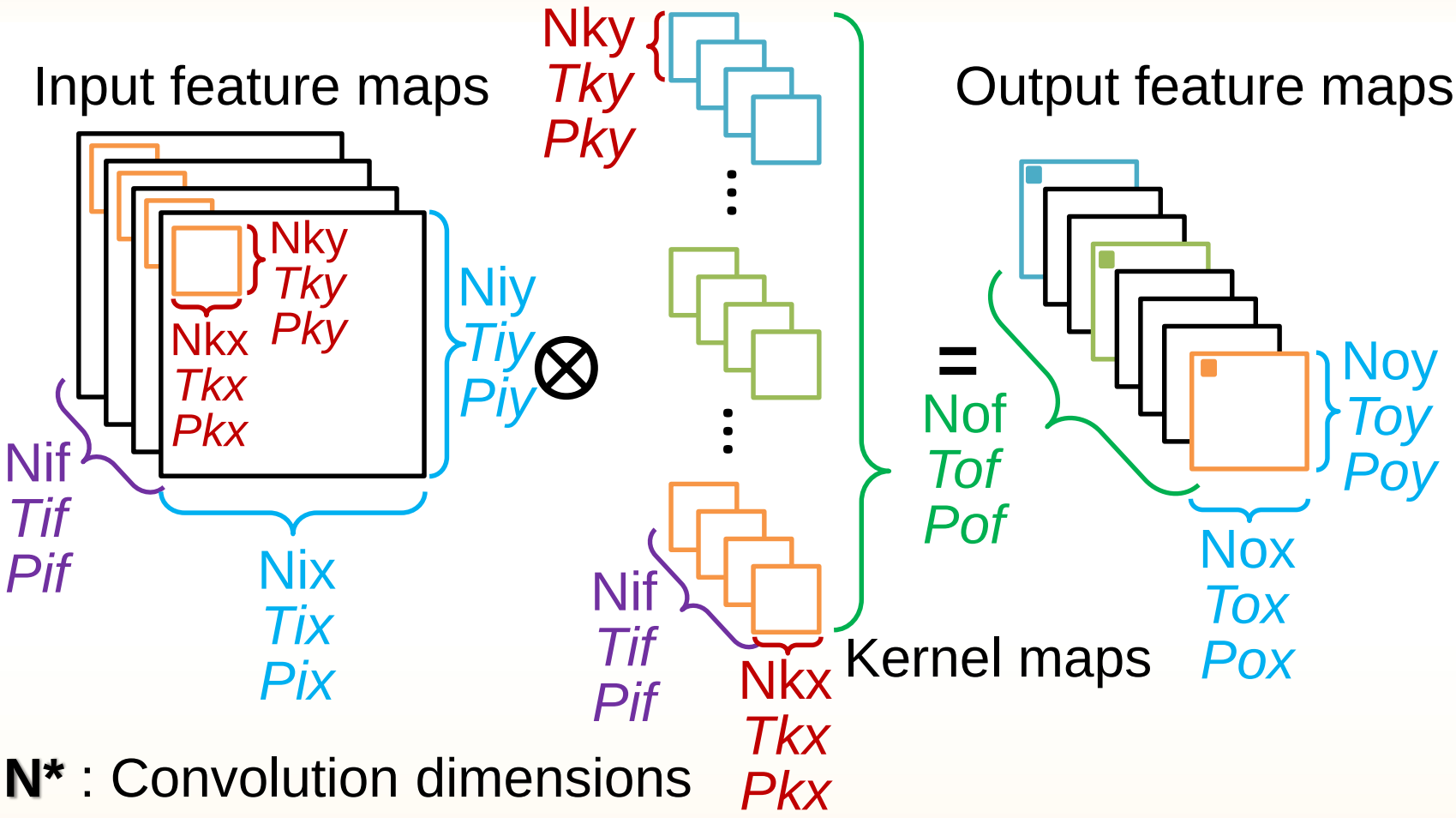


- - - →	**Loop-4**	Across the output feature maps of Nof
- - - →	**Loop-3**	Scan within one input feature map along Nix×Niy
- - - →	**Loop-2**	Across the input feature maps of Nif.
- - - →	**Loop-1**	MAC within a kernel window of Nkx×Nky

Loop Optimization Techniques

- Loop Unrolling
  - parallel computation of conv. MACs
  - register arrays and PE architecture
- Loop Tiling
  - increase data locality
  - determine on-chip buffer size
- Loop Interchange
  - computation order of four conv. loops

Parameters and Design Variables



N* : Convolution dimensions

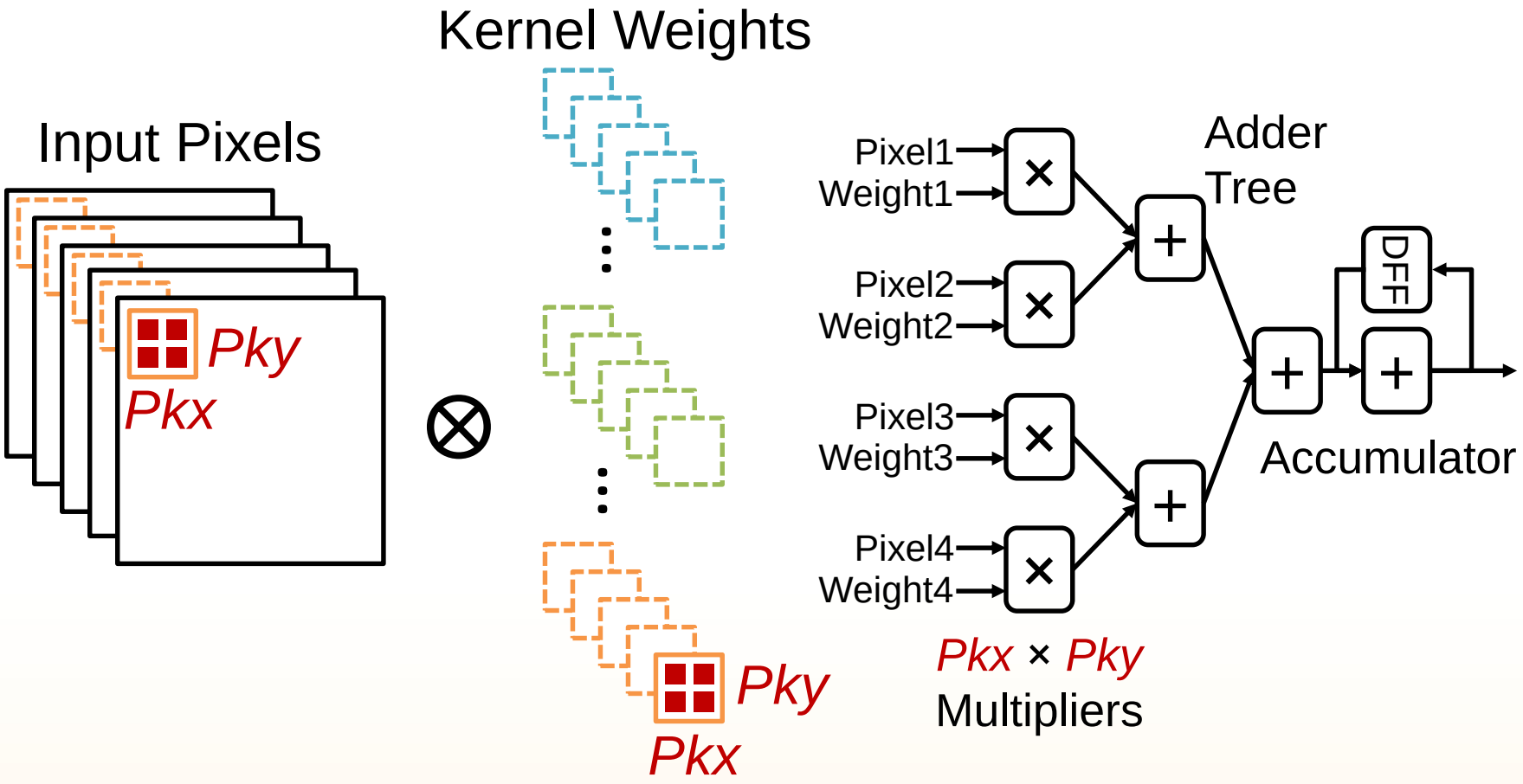
T* : Loop tiling design variables

P* : Loop unrolling design variables

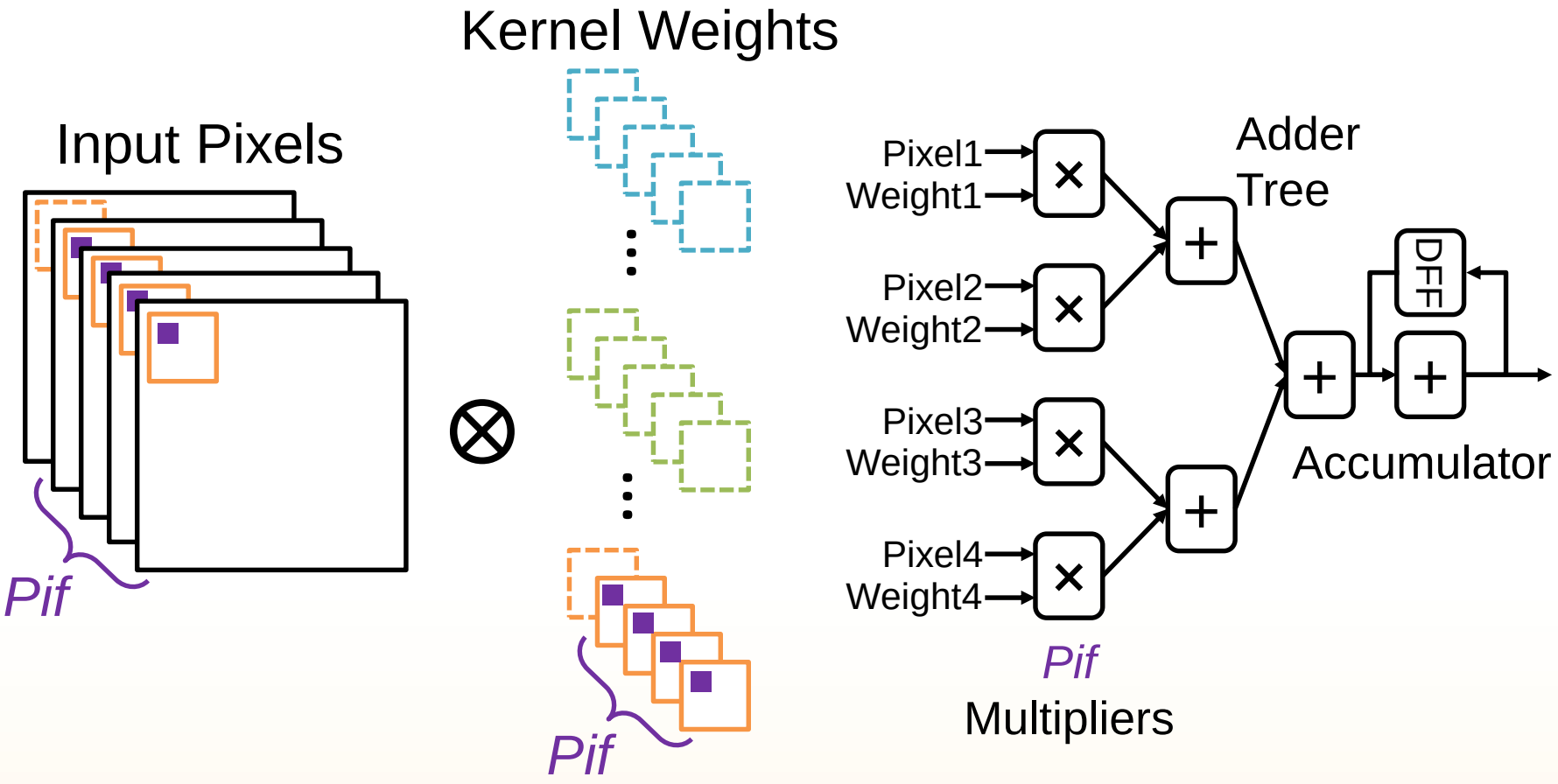
$1 \le P^* \le T^* \le N^*$, e.g.

$1 \le Pif \le Tif \le Nif$

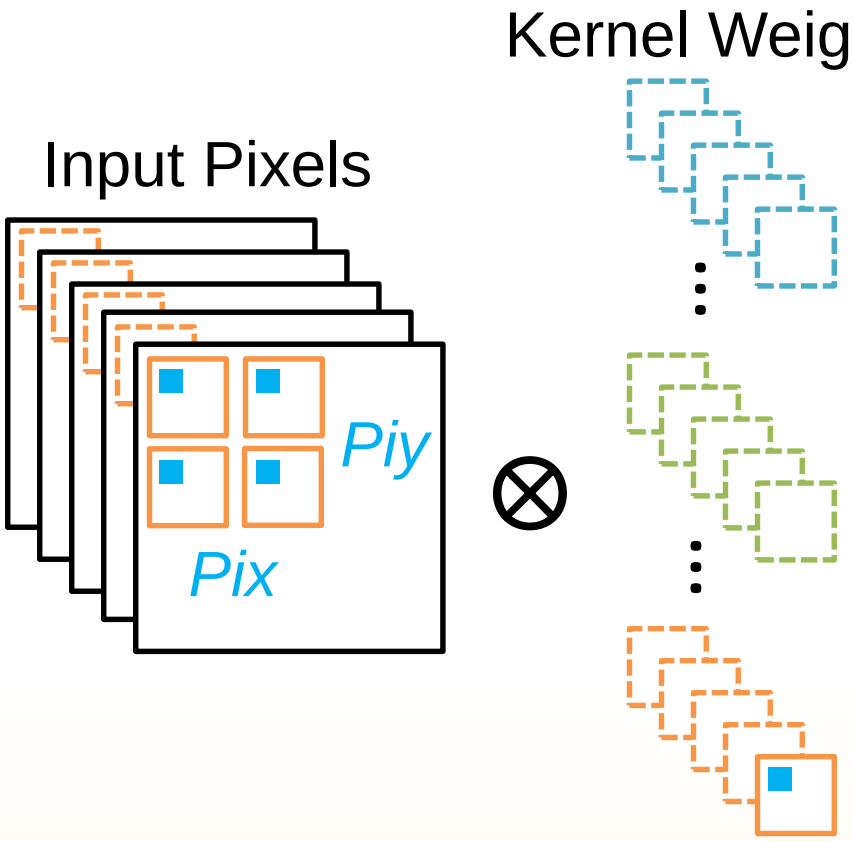
Loop Unrolling – Unroll Loop-1

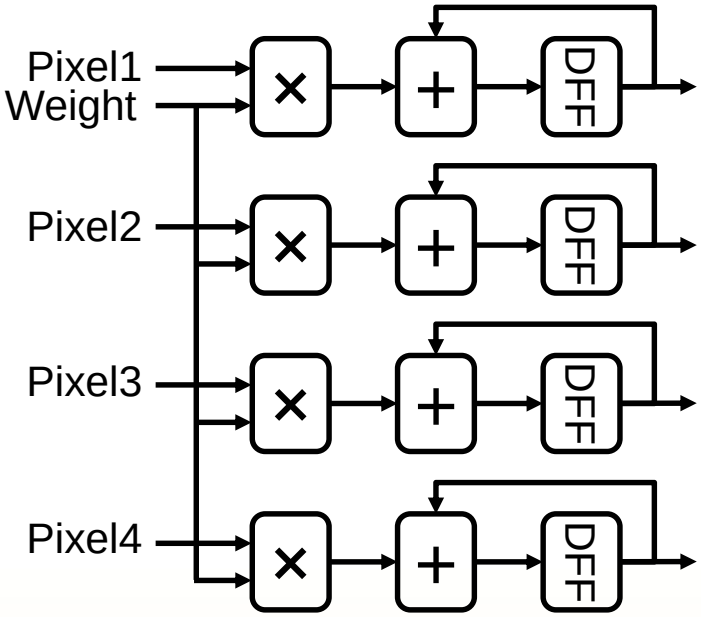


Loop Unrolling – Unroll Loop-2



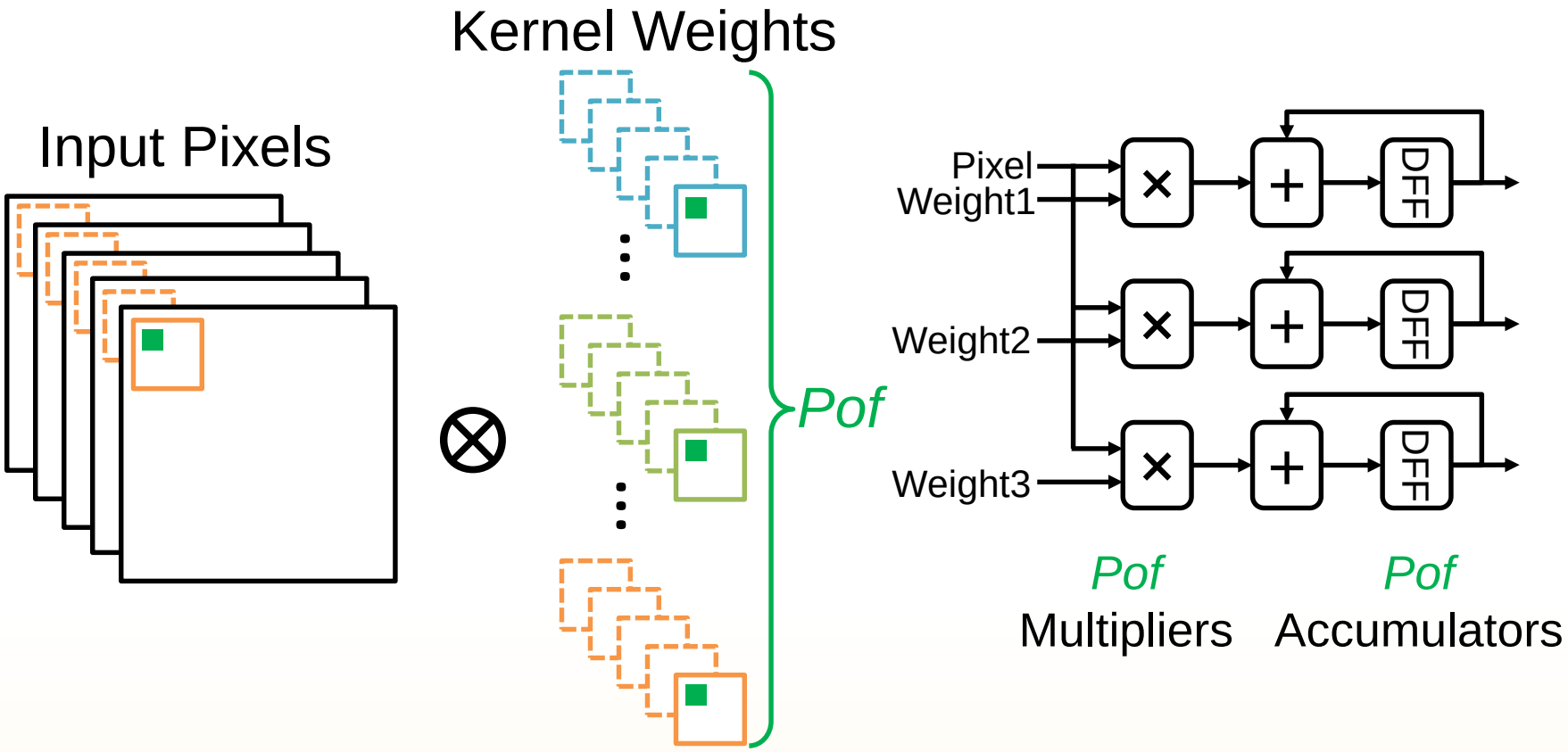
Loop Unrolling – Unroll Loop-3





The weight can be **reused** by *Pix* × *Piy* times.

Loop Unrolling – Unroll Loop-4



The pixel can be **reused** by *Pof* times.

Loop Optimization Techniques

- Loop Unrolling (*P**)
  - total # of parallel MACs or multipliers (*Pm*) is

    $Pm = Pkx \times Pky \times Pif \times Pix \times Piy \times Pof$

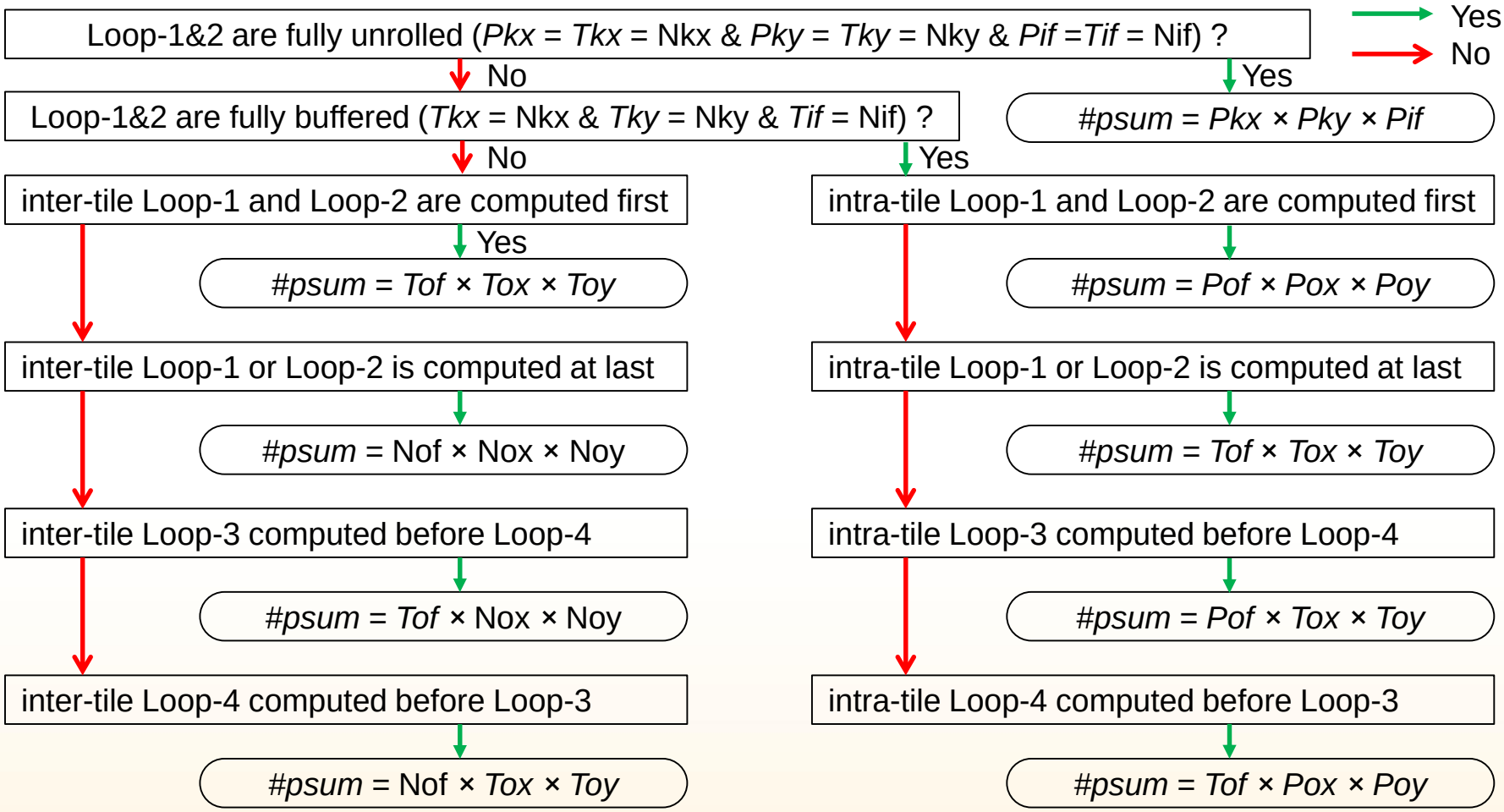
- Loop Tiling (*T**)
  - Input pixel buffer: *Tif* ×*Tix* × *Tiy* × pixel_datawidth
  - Weight buffer: *Tkx* ×*Tky* ×*Tif* ×*Tof* × weight_datawidth
  - Output pixel buffer: *Tox* × *Toy* × *Tof* × pixel_datawidth
- Loop Interchange
  - Intra-tile order: buffers to registers or PEs
  - Inter-tile order: external memory to buffers

Outline

- Overview of CNN Algorithm and Accelerator
- Convolution Loop Optimization
- Design Objectives of CNN Accelerator
  - Minimize Partial Sum Storage
  - Minimize On-chip Buffer Access
  - Minimize Off-chip DRAM Access
- Experimental Results
- Conclusion

Design Space Exp. – Minimize #*psum*

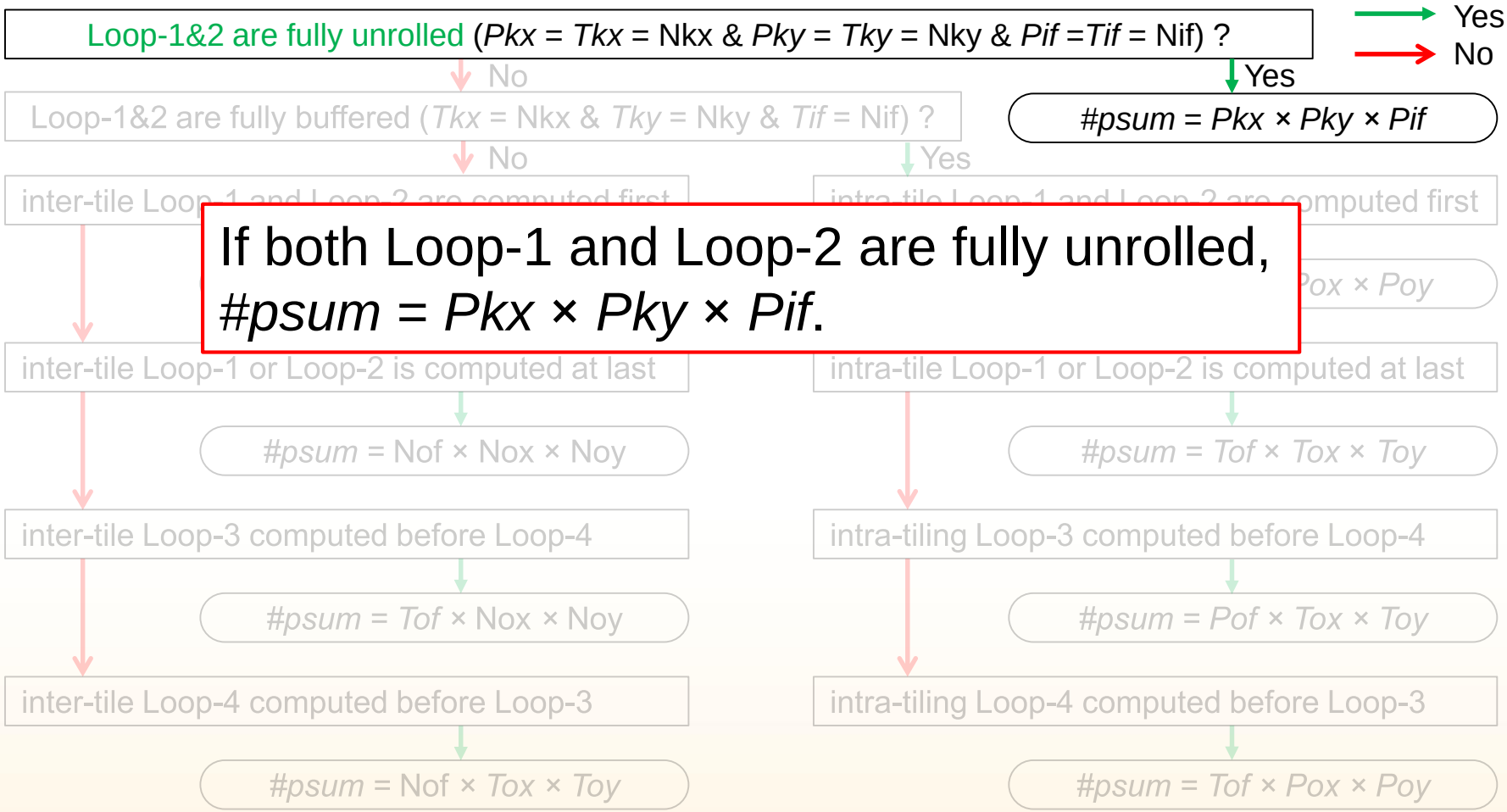
- ***#psum*** = # of partial sums need to be stored in memory
- To obtain one pixel, need to finish Loop-1 and Loop-2.



$1 \le P^* \le T^* \le N^*$, e.g. $1 \le Pof \le Tof \le Nof$

Design Space Exp. – Minimize *#psum*

- ***#psum*** = # of partial sums need to be stored in memory
- To obtain one pixel, need to finish Loop-1 and Loop-2.

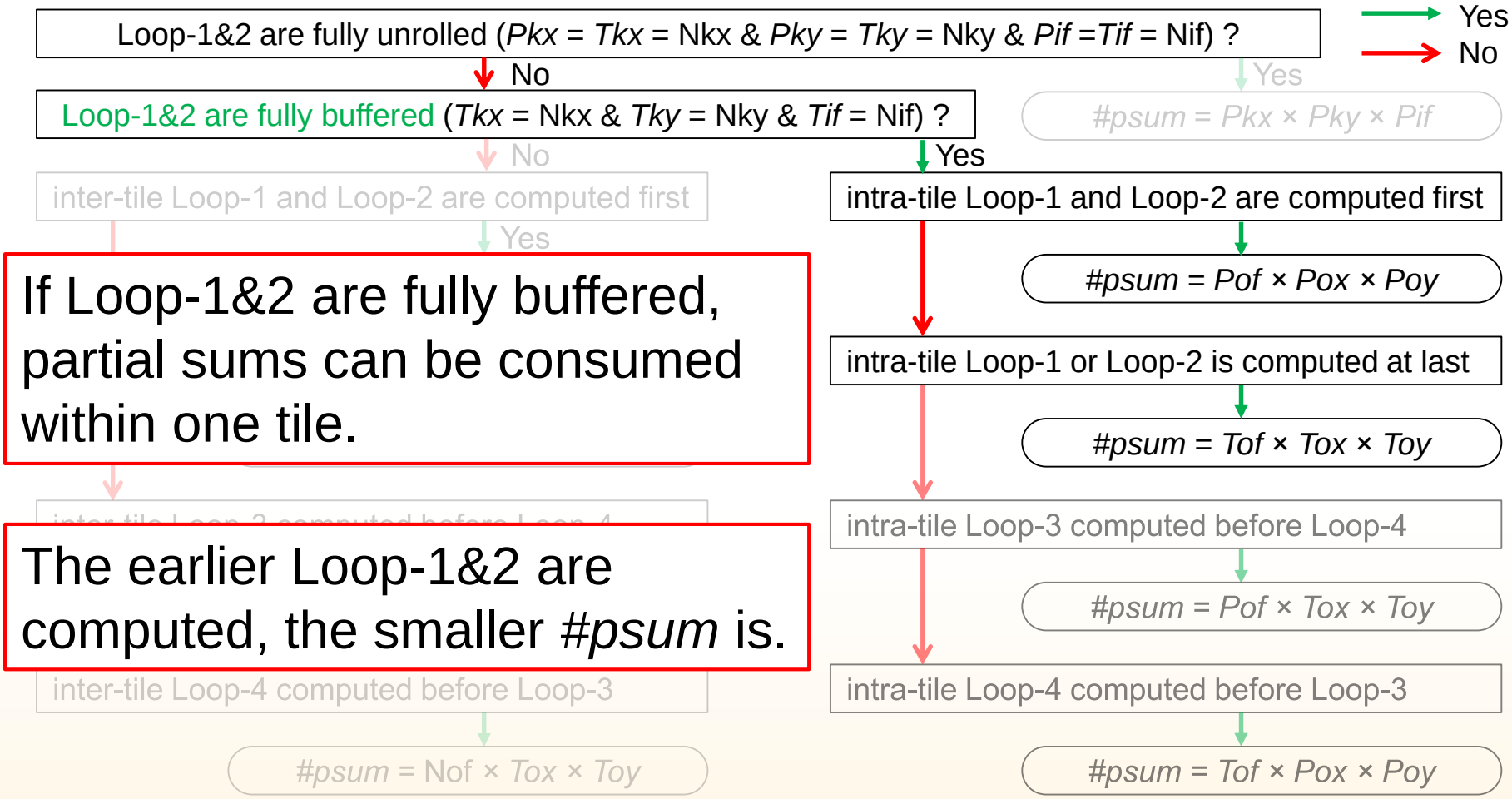


If both Loop-1 and Loop-2 are fully unrolled, *#psum = Pkx × Pky × Pif*.

$1 \le P^* \le T^* \le N^*$, e.g. $1 \le Pof \le Tof \le Nof$

Design Space Exp. – Minimize *#psum*

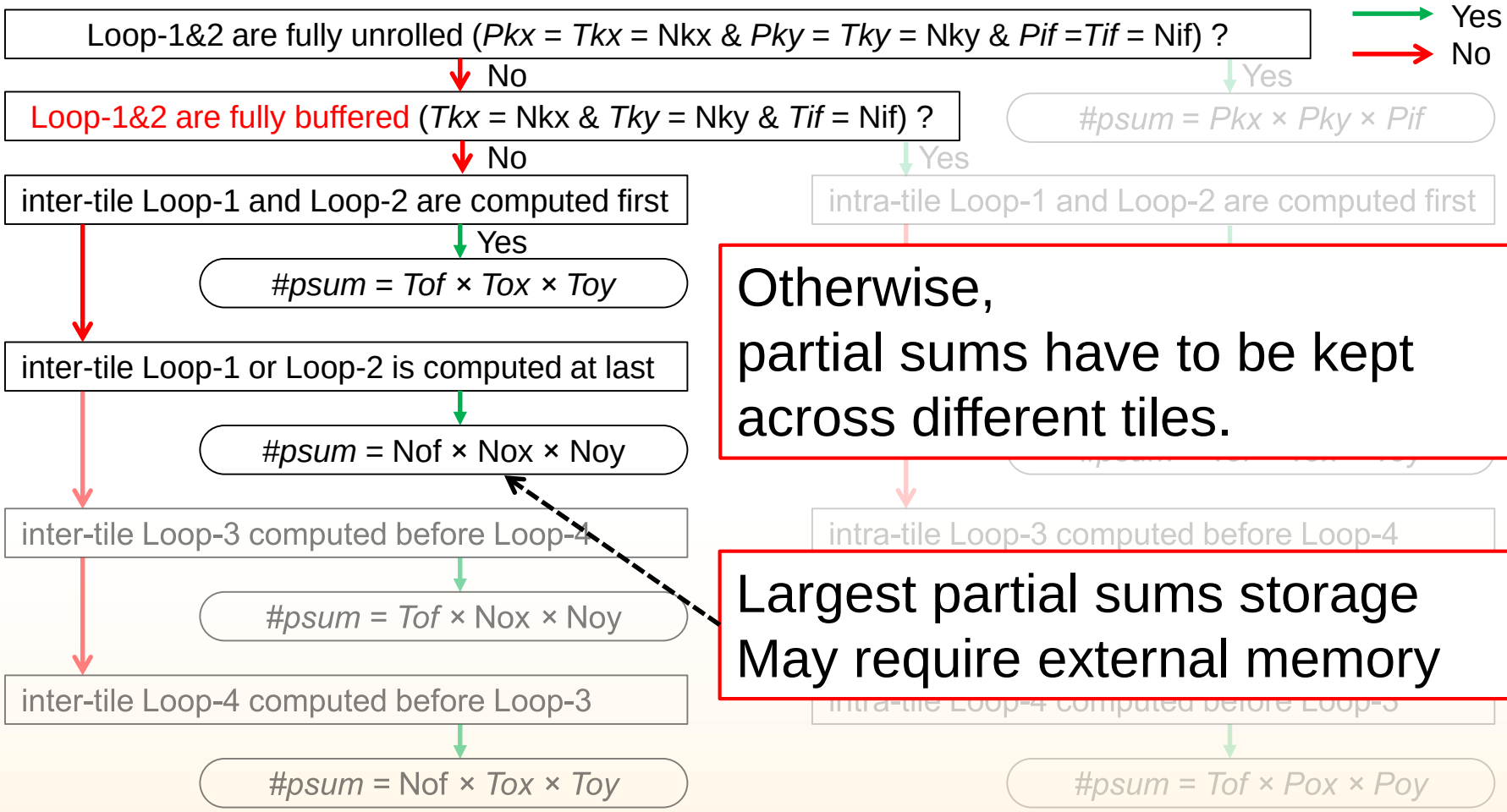
- ***#psum*** = # of partial sums need to be stored in memory
- To obtain one pixel, need to finish Loop-1 and Loop-2.



$1 \le P^* \le T^* \le N^*$, e.g. $1 \le Pof \le Tof \le Nof$

Design Space Exp. – Minimize *#psum*

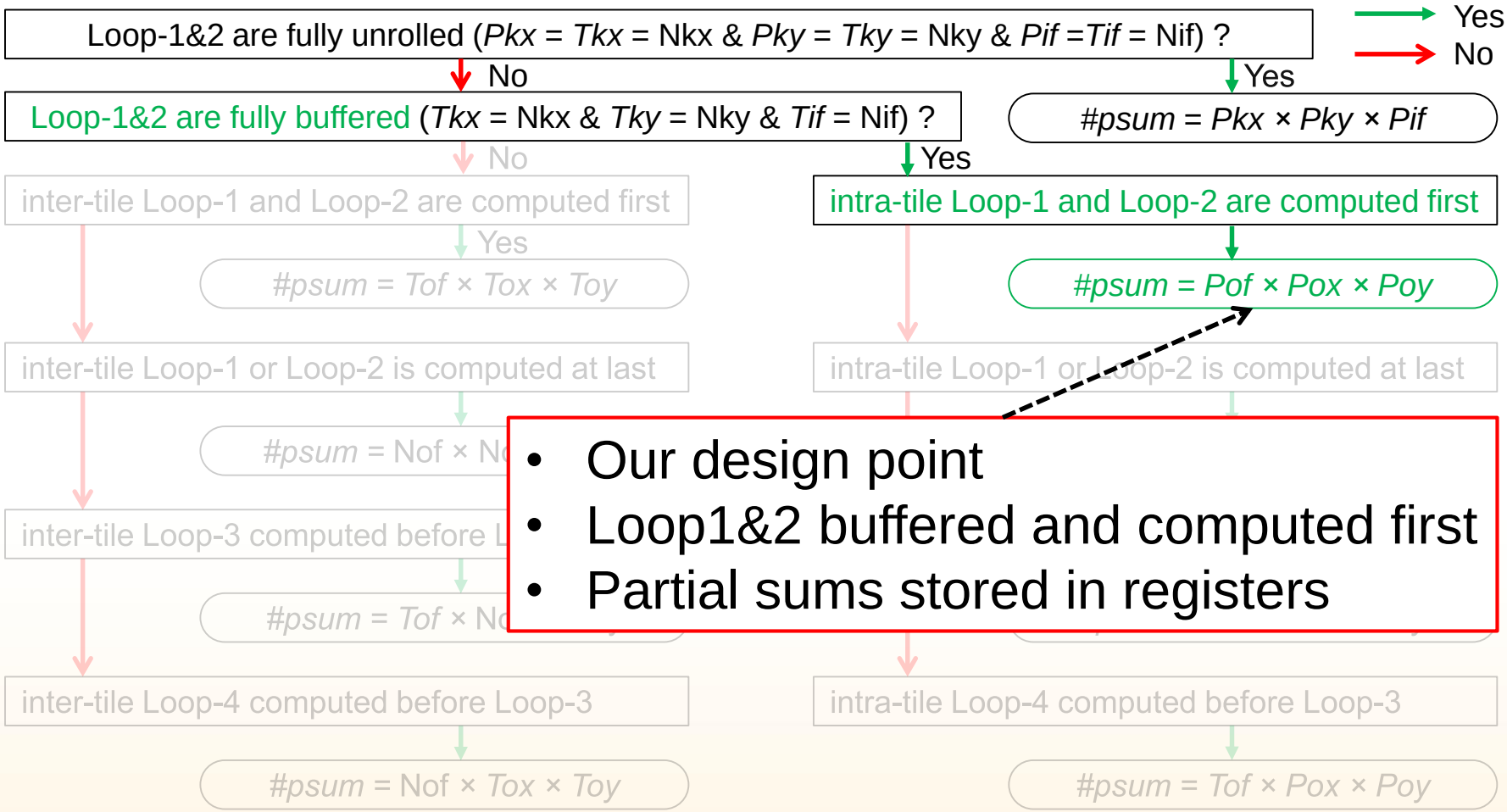
- ***#psum*** = # of partial sums need to be stored in memory
- To obtain one pixel, need to finish Loop-1 and Loop-2.



$1 \le P^* \le T^* \le N^*$, e.g. $1 \le Pof \le Tof \le Nof$

Design Space Exp. – Minimize *#psum*

- ***#psum*** = # of partial sums need to be stored in memory
- To obtain one pixel, need to finish Loop-1 and Loop-2.

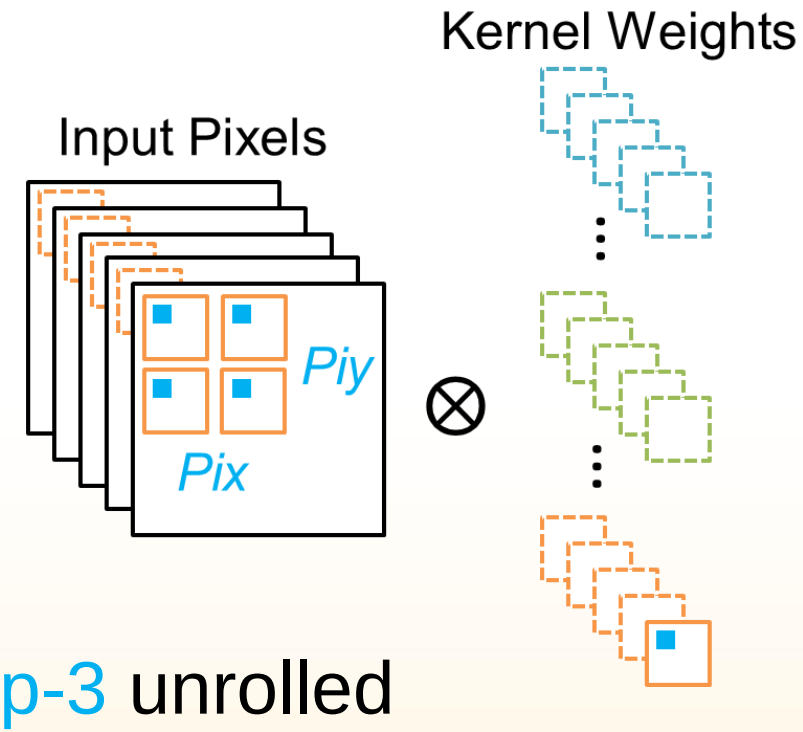


- Our design point
- Loop1&2 buffered and computed first
- Partial sums stored in registers

$1 \le P^* \le T^* \le N^*$, e.g. $1 \le Pof \le Tof \le Nof$

Reduce Buffer Access by Data Reuse

- A single pixel or weight can be reused for multiple multipliers after fetched out of on-chip buffers.
- Reuse times of a weight: *Reuse_wt* = *Pix* × *Piy*

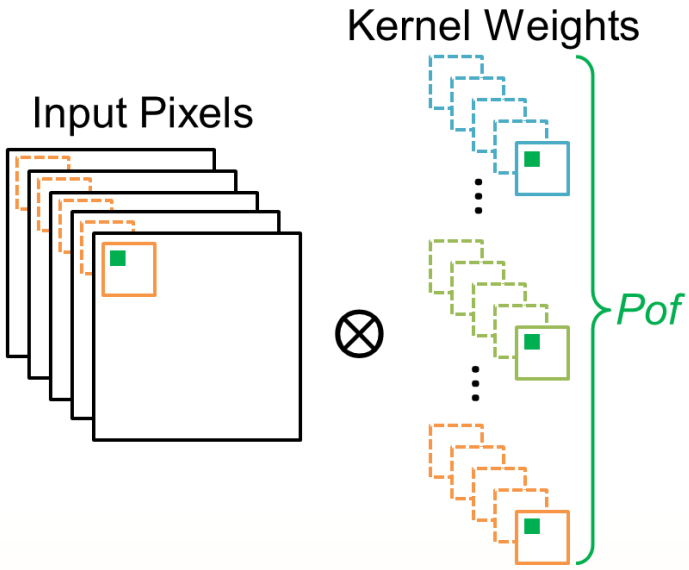


Loop-3 unrolled

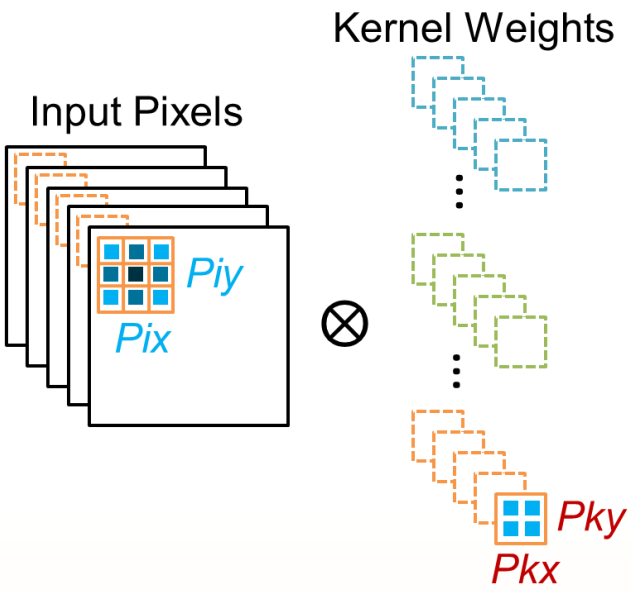
Reduce Buffer Access by Data Reuse

- Reuse times of a pixel: *Reuse_px*

Loop-4 unrolled



Both Loop-1 and Loop-3 unrolled



Reuse_px = *Pof*,
if *Pkx* = 1, *Pky* = 1

$$Reuse\_px = \frac{Pof \times Pkx \times Pky \times Pix \times Piy}{((Pix-1)S + Pkx) \times ((Piy-1)S + Pky)}$$

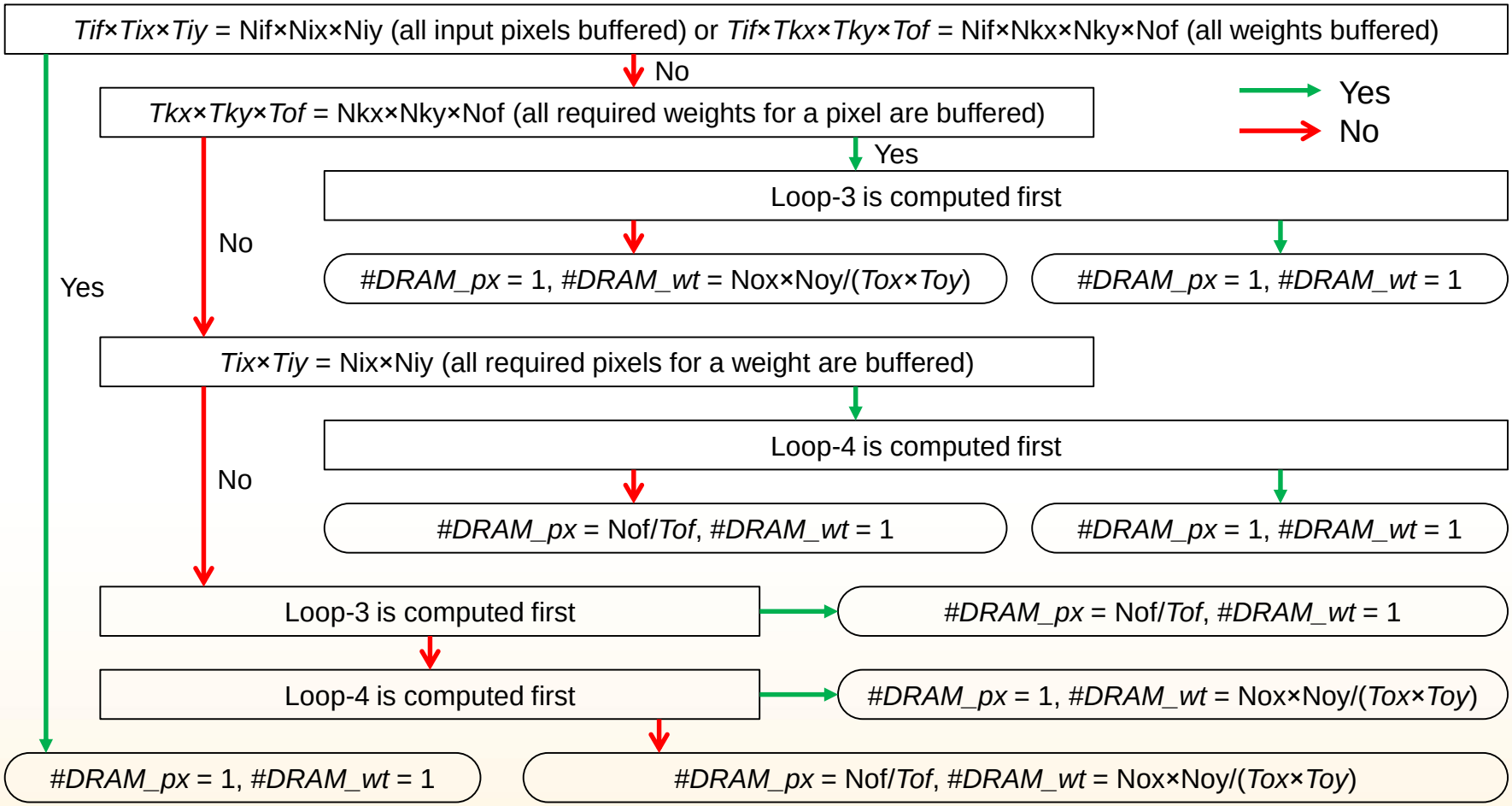
S: sliding stride

Design Spa. Exp. – Min DRAM Access

- Both weights and intermediate results of pixels are stored in external memory
  - large-scale CNNs (> 50 MB)
  - limited FPGA RAM capacity (< 50 Mbits)
- Minimum access of external memory
  - read every pixel and weight only once
  - maximal data reuse with on-chip buffer
  - proper loop computing orders

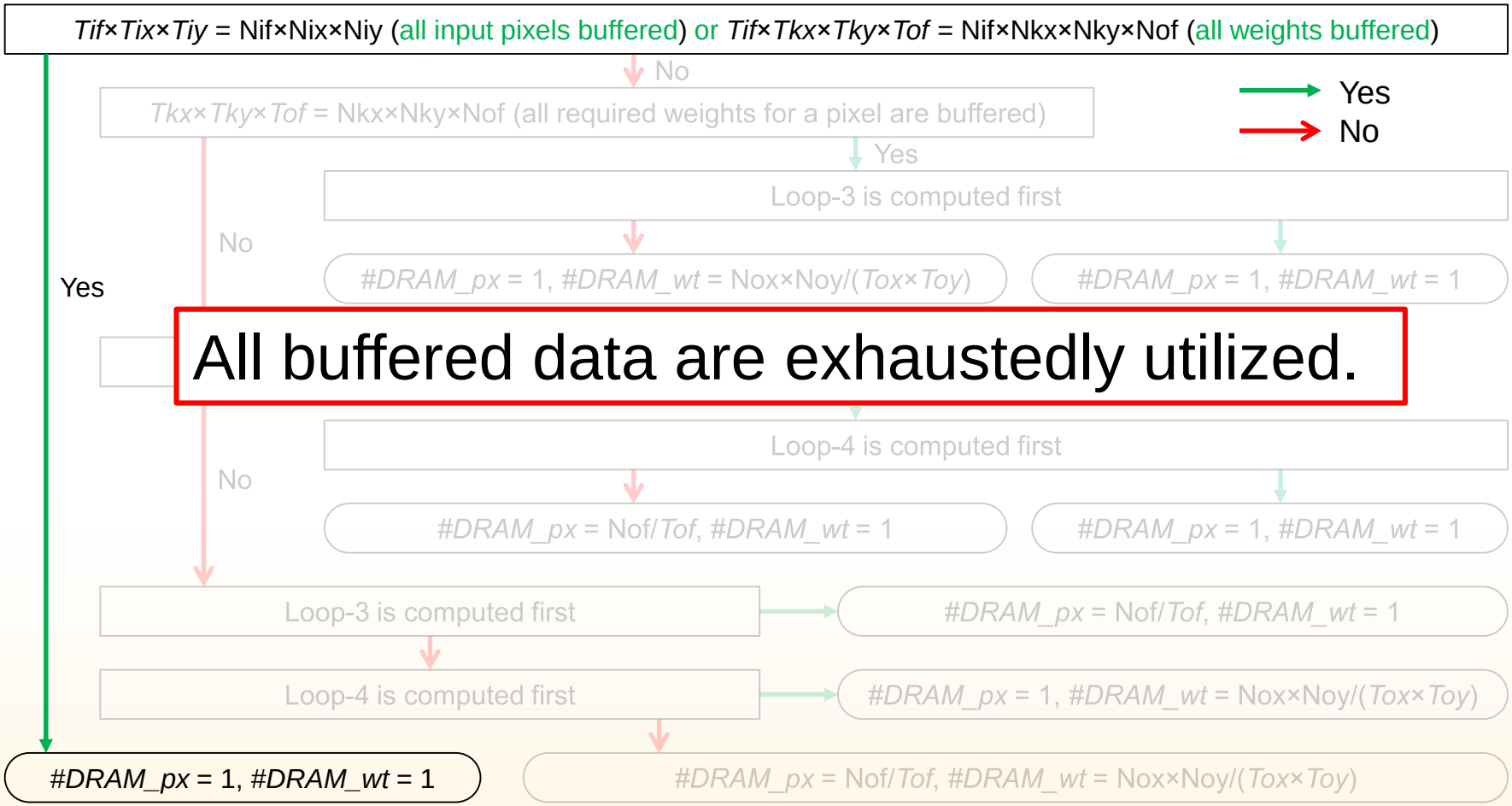
Design Spa. Exp. – Min DRAM Access

- *#DRAM_px* : # of DRAM reads of one input pixel in one layer
- *#DRAM_wt* : # of DRAM reads of one weight in one layer



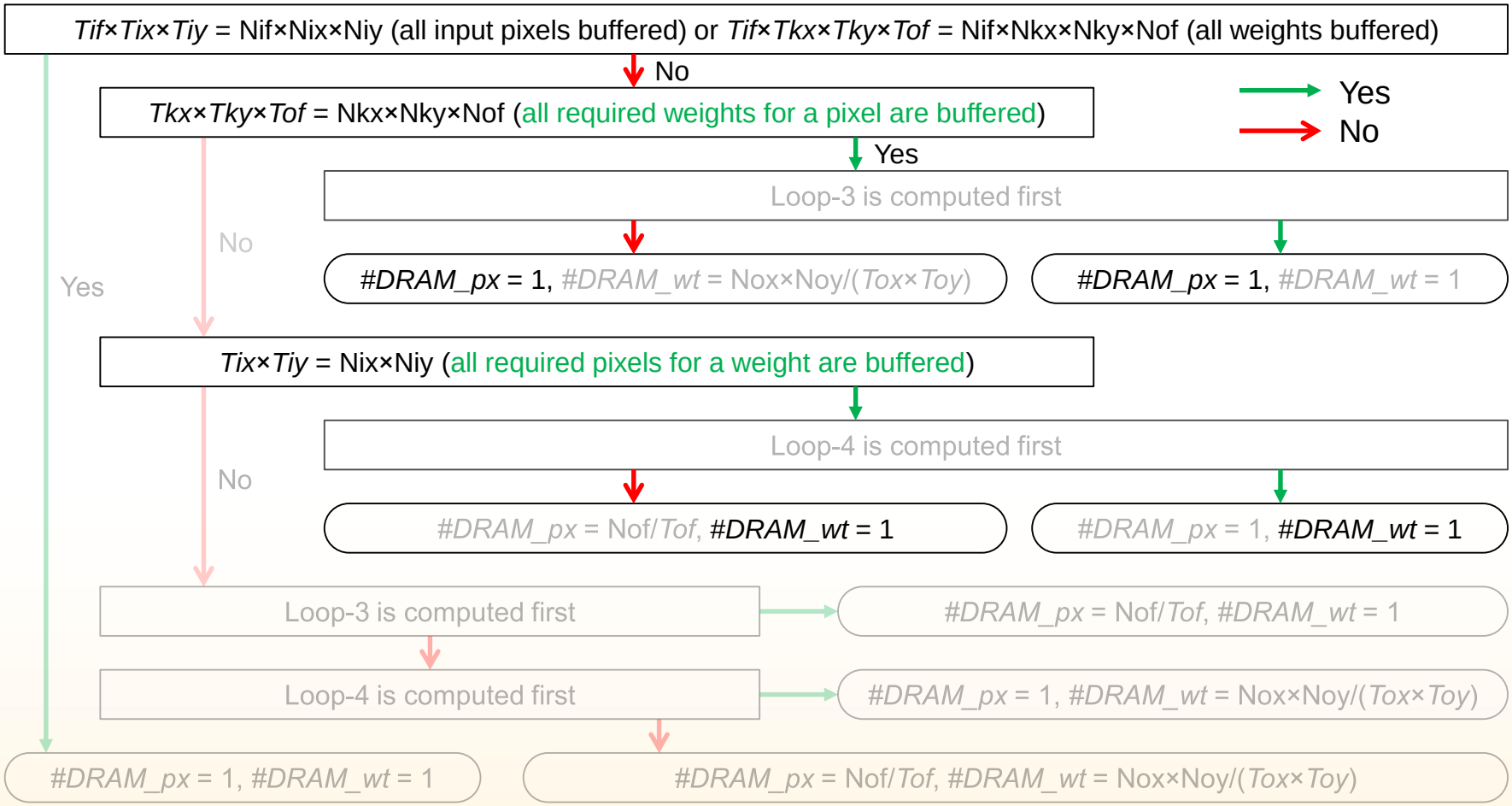
Design Spa. Exp. – Min DRAM Access

- *#DRAM_px* : # of DRAM reads of one input pixel in one layer
- *#DRAM_wt* : # of DRAM reads of one weight in one layer



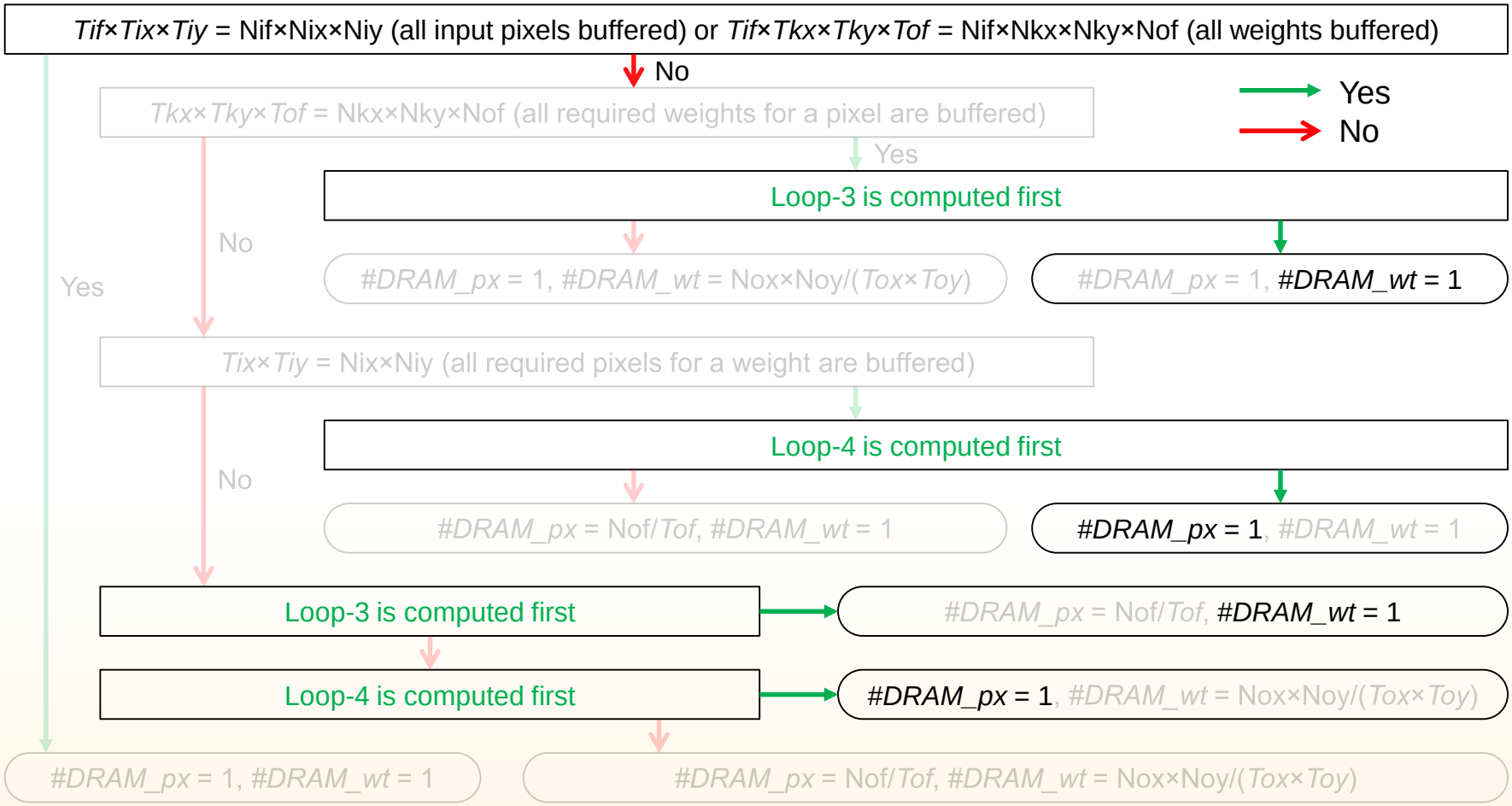
Design Spa. Exp. – Min DRAM Access

- *#DRAM_px* : # of DRAM reads of one input pixel in one layer
- *#DRAM_wt* : # of DRAM reads of one weight in one layer



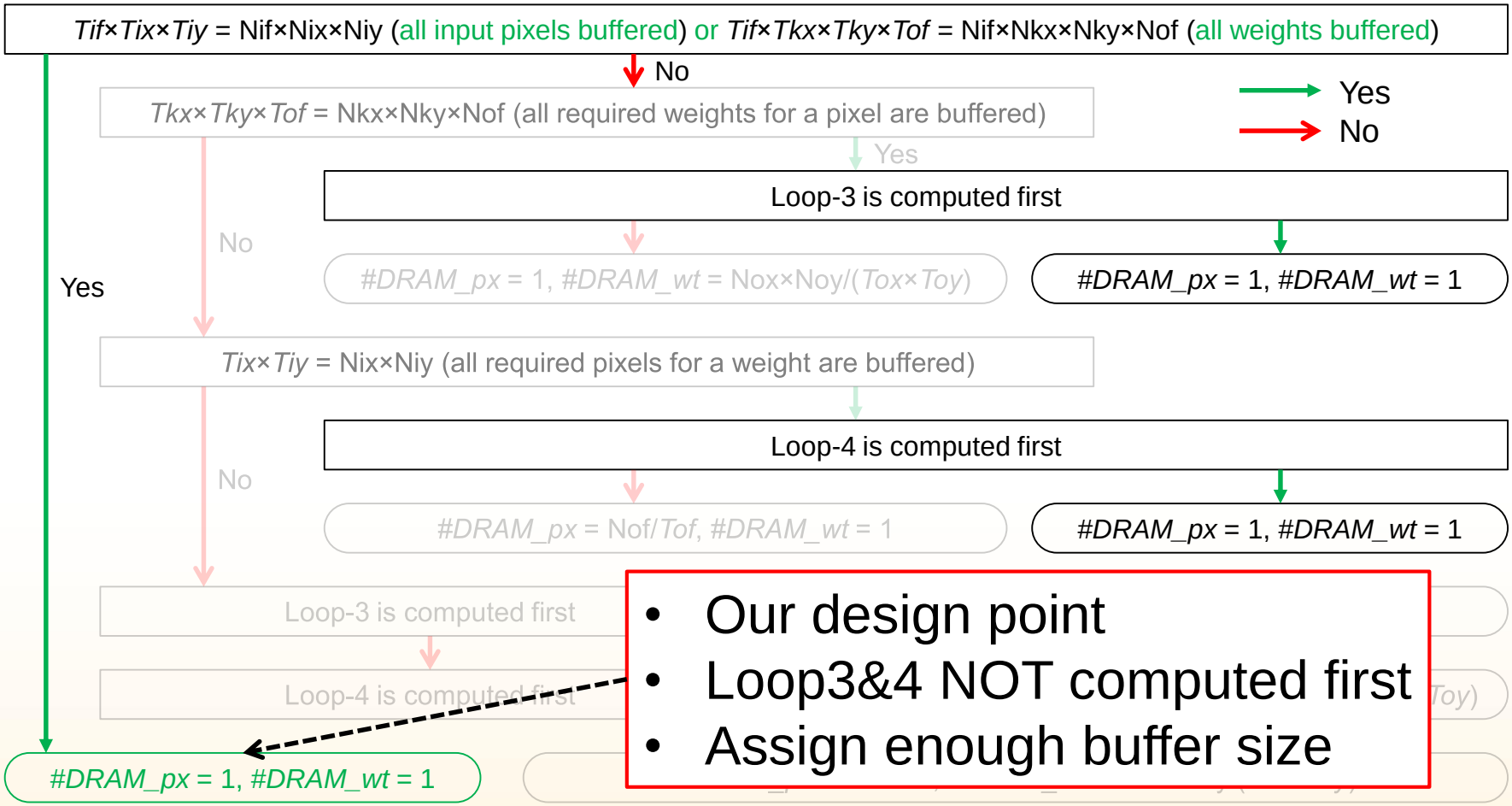
Design Spa. Exp. – Min DRAM Access

- *#DRAM_px* : # of DRAM reads of one input pixel in one layer
- *#DRAM_wt* : # of DRAM reads of one weight in one layer



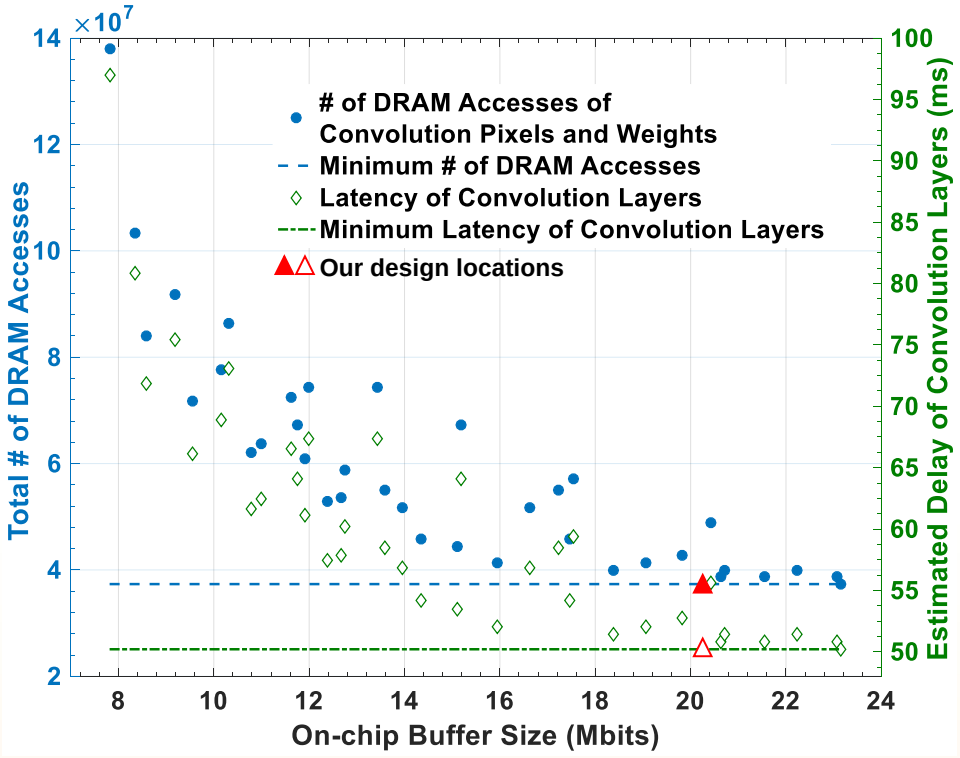
Design Spa. Exp. – Min DRAM Access

- *#DRAM_px* : # of DRAM reads of one input pixel in one layer
- *#DRAM_wt* : # of DRAM reads of one weight in one layer



Design Spa. Exp. – Min DRAM Access

- Buffer size vs. # of DRAM access by tuning *Toy* and *Tof*
- *Tkx* × *Tky* = Nkx × Nky & *Tif* = Nif : buffer both Loop-1&2
- *Tox* = Nox : benefit DMA transactions with continuous data



~50Mbits RAM in our FPGA

On-chip buffer size vs. # of DRAM access and conv. delay

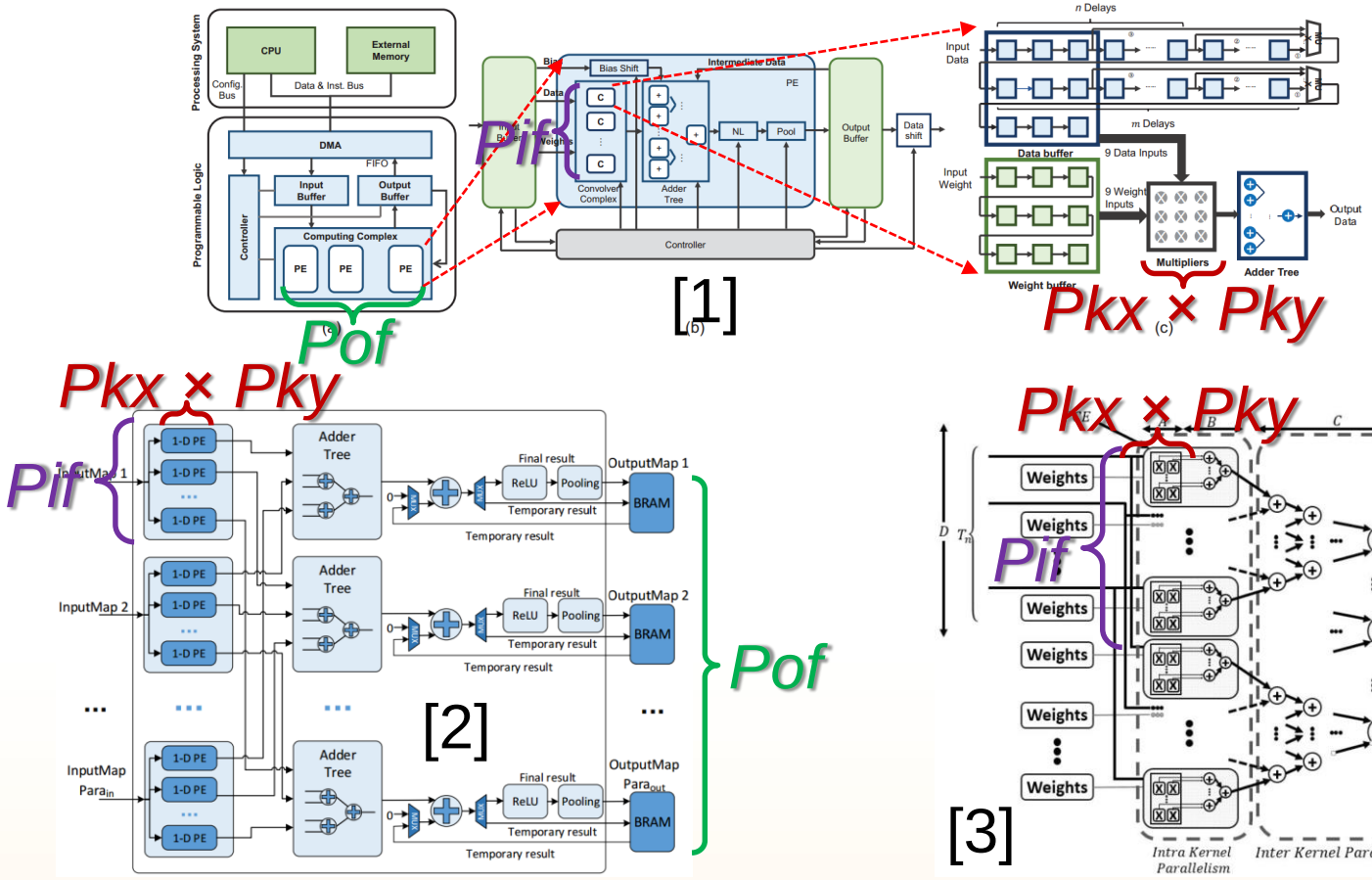
Estimated conv. delay = computing time + DRAM delay

Outline

- Overview of CNN Algorithm and Accelerator
- Convolution Loop Optimization
- Design Objectives of CNN Accelerator
- Loop Optimization in Related Works
  - Type-(A): unroll Loop-1, Loop-2, Loop-4
  - Type-(B): unroll Loop-2, Loop-4
  - Type-(C): unroll Loop-1, Loop-3
  - Type-(D): unroll Loop-3, Loop-4

Loop Optimization in Related Work

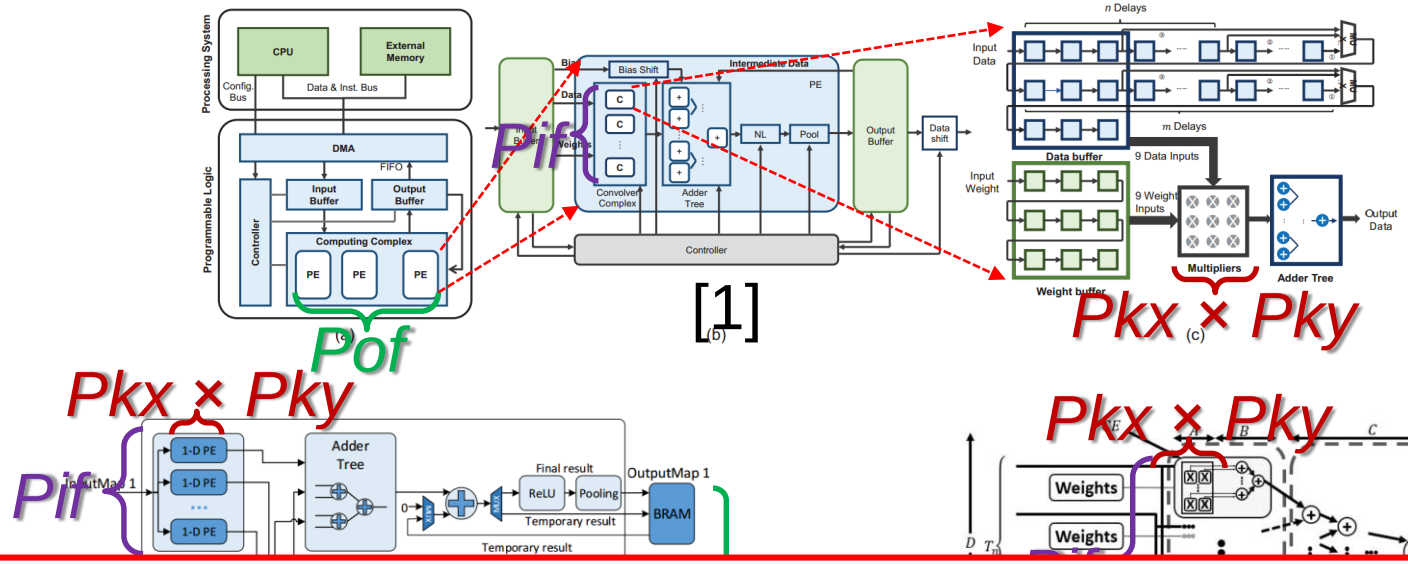
- Type-(A): Unroll Loop-1, Loop-2, Loop-4 [1][2][3]



[1] J. Qiu, et al. Going deeper with embedded FPGA platform for convolutional neural network. In ACM *FPGA* 2016.
[2] H. Li, et al. A high performance FPGA-based accelerator for large-scale convolutional neural networks, In *FPL* 2016.
[3] M. Motamedi, et al. Design space exploration of FPGA based deep convolutional neural networks. In *ASP-DAC* 2016.

Loop Optimization in Related Work

- Type-(A): Unroll Loop-1, Loop-2, Loop-4 [1][2][3]

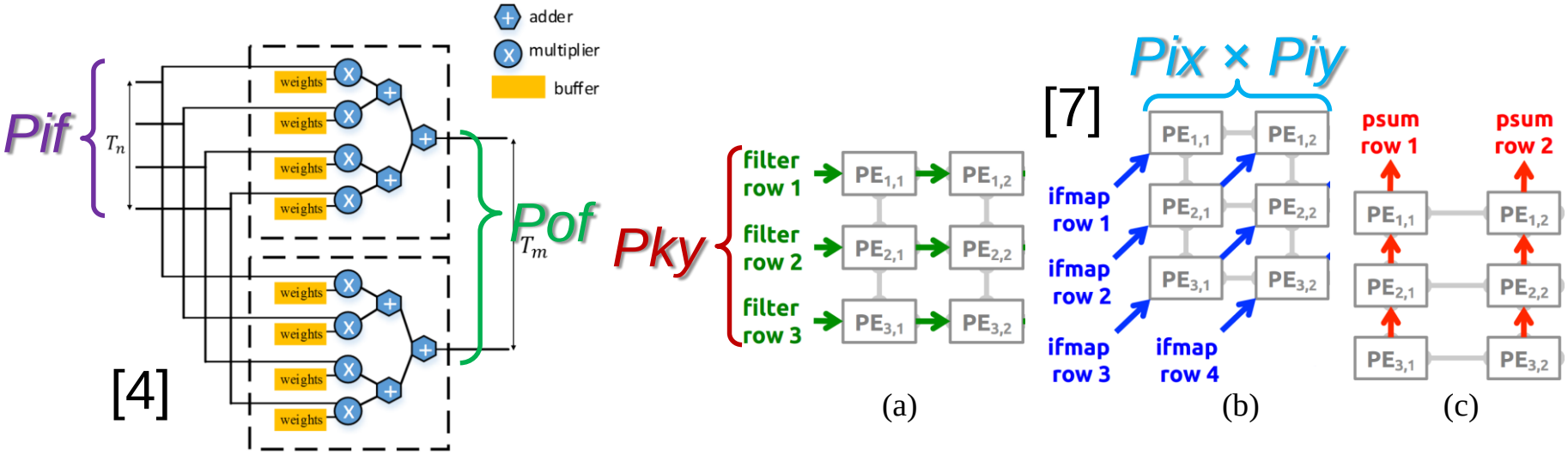


- Unroll Loop-1 employs parallelism within kernel maps
- Kernel sizes (Nkx × Nky) are small
  - Cannot provide sufficient parallelism
- Kernel sizes vary considerably across different conv. layers
  - Workload imbalance and PE mapping difficulty

[1
[2
[3
6.

Loop Optimization in Related Work

- Type-(B): Unroll Loop-2, Loop-4 [4][5]
- Type-(C): Unroll Loop-1, Loop-3 [6][7]



[4] C. Zhang, et al. Optimizing FPGA-based accelerator design for deep Convolutional Neural Networks. In ACM *FPGA* 2015.

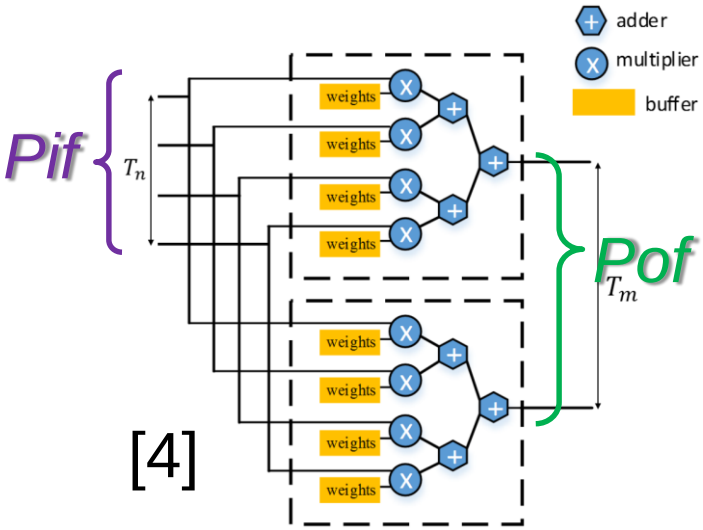
[5] Y. Ma, et al. Scalable and modularized RTL compilation of Convolutional Neural Networks onto FPGA. In *FPL* 2016.

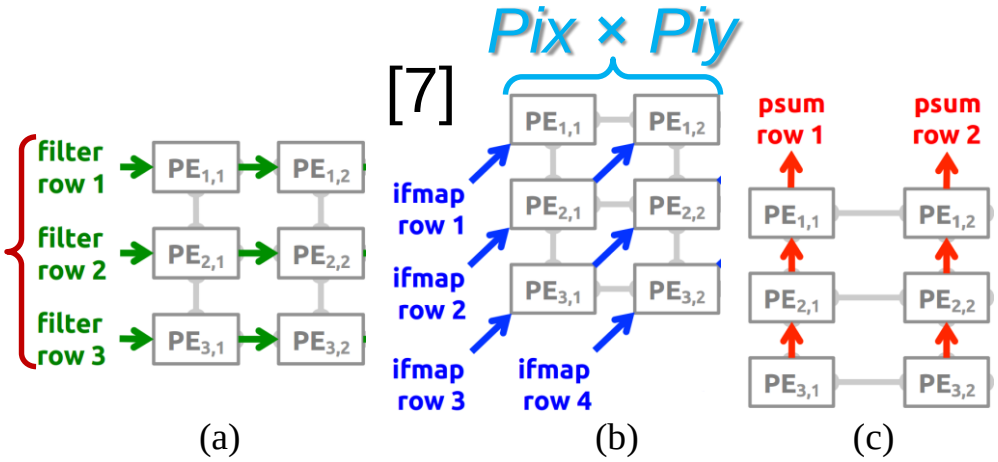
[6] Y.-H. Chen, et al. Eyeriss: An energy-efficient reconfigurable accelerator for deep Convolutional Neural Networks. In *ISSCC* 2016.

[7] Y.-H. Chen, et al. Eyeriss: A spatial architecture for energy-efficient dataflow for Convolutional Neural Networks. In *ISCA* 2016.

Loop Optimization in Related Work

- Type-(B): Unroll Loop-2, Loop-4 [4][5]
- Type-(C): Unroll Loop-1, Loop-3 [6][7]



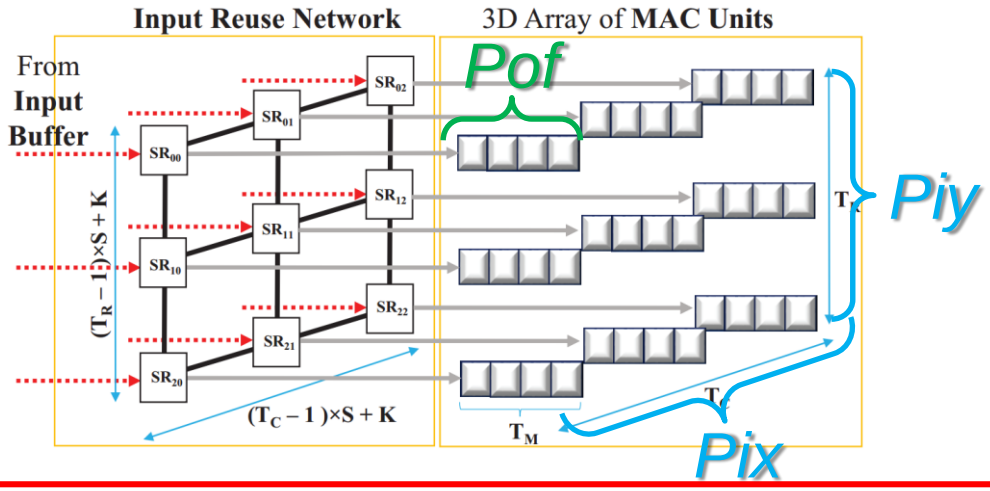


- Type-(A)&(B) only reuses pixels by unrolling Loop-4.
- Type-(C) reuses pixels by the overlapping of Loop-1 and Loop-3, and reuses weights by unrolling Loop-3.
- Type-(C) is also affected by the issue of unrolling Loop-1.

Loop Optimization in Related Work

- Type-(D): Unroll Loop-3, Loop-4 [8]

[8]



- Type-(D) reuses weights by unrolling Loop-4.
- Type-(D) reuses pixels by unrolling Loop-3.
- Nix × Niy × Nof is large to provide sufficient parallelism.
- Data tiles in [8] do NOT cover Loop-2.
  - increase movements and storage of partial sums.

[8] A. Rahman, et al. Efficient FPGA acceleration of Convolutional Neural Networks using logical-3D compute array. In *DATE* 2016.

Outline

- Overview of CNN Algorithm and Accelerator
- Convolution Loop Optimization
- Design Objectives of CNN Accelerator
- Loop Optimization in Related Works
- Proposed CNN Accelerator
  - Acceleration Scheme
  - Convolution Dataflow
  - Convolution PE Architecture

Proposed Acceleration Scheme

- **Loop Unrolling (*P**)**
  - *Pkx* = *Pky* = 1, *Pif* = 1, *Pox* = *Poy* = 14, *Pof* = 16 for all conv. layers
  - Reuse pixels/weights by unrolling Loop-3/Loop-4, respectively
  - Total # of PEs = 3,136 (≈ # of DSP multipliers in our FPGA)
  - Uniform PE mapping by constant unrolling factors (*P**) for all conv.
- **Loop Tiling (*T**)**
  - Loop-1 & Loop-2 are buffered: *Tkx* × *Tky* = Nkx × Nky, *Tif* = Nif
    - Only *Pof* × *Pox* × *Poy* partial sums stored in separate registers
  - Either *Tix* × *Tiy* = Nix × Niy or *Tof* = Nof for every conv. layer
    - Ensure every pixel and weight are read only **once** from DRAM
- **Loop Interchange**
  - Serially compute Loop-1&2 first to consume partial sums ASAP

Proposed Convolution Dataflow

- Serially compute Loop-1: 3 × 3 Conv kernel illustration

Input feature map with zero padding

0	0	0	0	0	0	0	0
0	11	12	13	14	15	16	0
0	21	22	23	24	25	26	0
0	31	32	33	34	35	36	0
0	41	42	43	44	45	46	0
0	51	52	53	54	55	56	0
0	61	62	63	64	65	66	0
0	0	0	0	0	0	0	0

: output pixel location

Kernel Map

1	2	3
4	5	6
7	8	9

M* denotes MAC units
Green pixels are in registers (**R**)

clk	M21
0	0
1	11
2	12
3	0
4	21
5	22
6	0
7	31
8	32

×

WT	Kernel Weights
1	
2	
3	
4	
5	
6	
7	
8	
9	

Proposed Convolution Dataflow

- Serially compute Loop-1: 3 × 3 Conv kernel illustration
- Adjacent output pixel computation

M* denotes MAC units
Green pixels are in registers (**R**)

Input feature map with zero padding

0	0	0	0	0	0	0	0
0	11	12	13	14	15	16	0
0	21	22	23	24	25	26	0
0	31	32	33	34	35	36	0
0	41	42	43	44	45	46	0
0	51	52	53	54	55	56	0
0	61	62	63	64	65	66	0
0	0	0	0	0	0	0	0

: output pixel location

Kernel Map

1	2	3
4	5	6
7	8	9

clk	M22
0	11
1	12
2	13
3	21
4	22
5	23
6	31
7	32
8	33

×

WT	Kernel Weights
1	
2	
3	
4	
5	
6	
7	
8	
9	

Proposed Convolution Dataflow

- Serially compute Loop-1: 3 × 3 Conv kernel illustration
- Adjacent output pixel computation

M* denotes MAC units
Green pixels are in registers (**R**)

Input feature map with zero padding

0	0	0	0	0	0	0	0
0	11	12	13	14	15	16	0
0	21	22	23	24	25	26	0
0	31	32	33	34	35	36	0
0	41	42	43	44	45	46	0
0	51	52	53	54	55	56	0
0	61	62	63	64	65	66	0
0	0	0	0	0	0	0	0

: output pixel location

Kernel Map

1	2	3
4	5	6
7	8	9

clk	M23
0	12
1	13
2	14
3	22
4	23
5	24
6	32
7	33
8	34

×

WT	Kernel Weights
1	
2	
3	
4	
5	
6	
7	
8	
9	

Proposed Convolution Dataflow

- Same weight is used (multiplied) at all MACs in a given clock cycle

Input feature map with zero padding

0	0	0	0	0	0	0	0
0	11	12	13	14	15	16	0
0	21	22	23	24	25	26	0
0	31	32	33	34	35	36	0
0	41	42	43	44	45	46	0
0	51	52	53	54	55	56	0
0	61	62	63	64	65	66	0
0	0	0	0	0	0	0	0

: output pixel location

Kernel Map

1	2	3
4	5	6
7	8	9

M* denotes MAC units
Green pixels are in registers (**R**)

clk	M21	M22	M23
0	0	11	12
1	11	12	13
2	12	13	14
3	0	21	22
4	21	22	23
5	22	23	24
6	0	31	32
7	31	32	33
8	32	33	34

×

Kernel Weights

WT
1
2
3
4
5
6
7
8
9

Proposed Convolution Dataflow

- Pixels are reused by movements among register arrays

Input feature map with zero padding

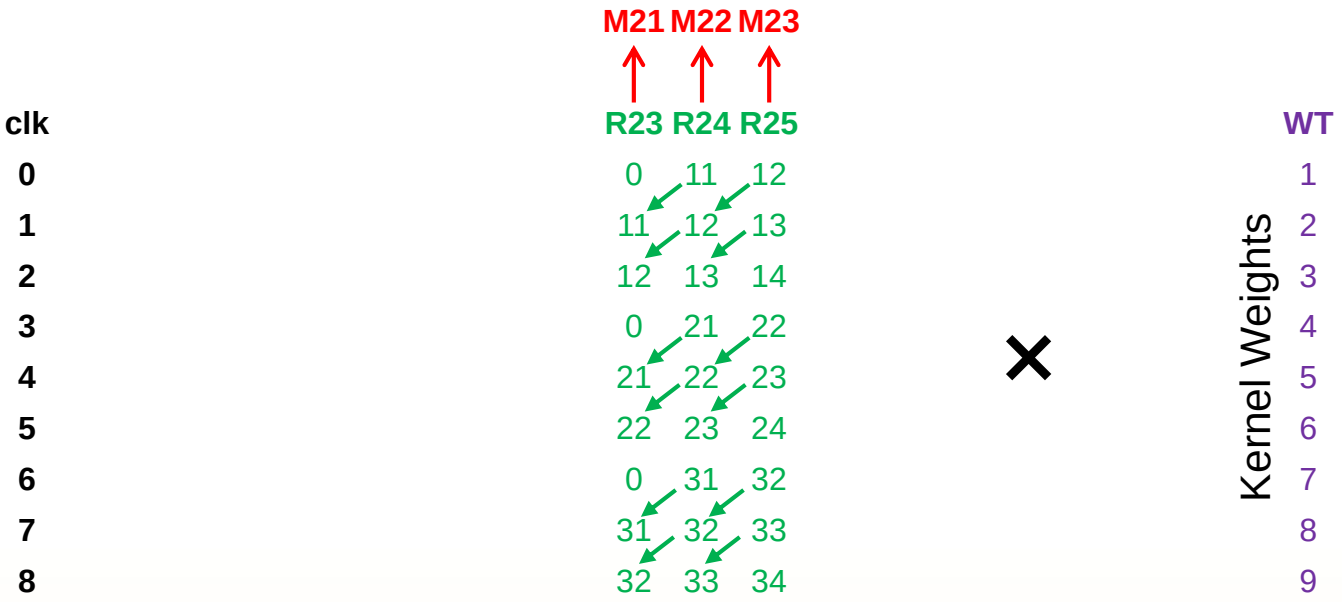
0	0	0	0	0	0	0	0
0	11	12	13	14	15	16	0
0	21	22	23	24	25	26	0
0	31	32	33	34	35	36	0
0	41	42	43	44	45	46	0
0	51	52	53	54	55	56	0
0	61	62	63	64	65	66	0
0	0	0	0	0	0	0	0

: output pixel location

Kernel Map

1	2	3
4	5	6
7	8	9

M* denotes MAC units
Green pixels are in registers (**R**)



Proposed Convolution Dataflow

- Same weight is used (multiplied) at all MACs in a given clock cycle
- Unroll Loop-3: parallelism within one input feature map

Input feature map with zero padding

0	0	0	0	0	0	0	0
0	11	12	13	14	15	16	0
0	21	22	23	24	25	26	0
0	31	32	33	34	35	36	0
0	41	42	43	44	45	46	0
0	51	52	53	54	55	56	0
0	61	62	63	64	65	66	0
0	0	0	0	0	0	0	0

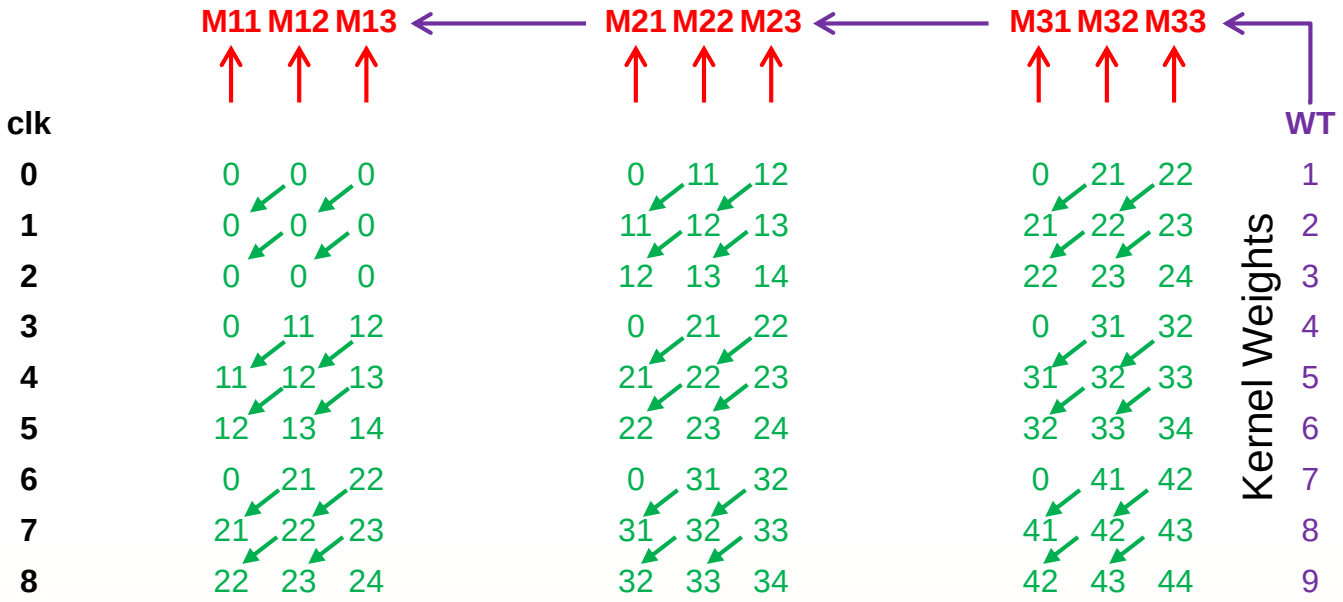
: output pixel location

Kernel Map

1	2	3
4	5	6
7	8	9

M* denotes MAC units

Green pixels are in registers (**R**)



Proposed Convolution Dataflow

- Same weight is used (multiplied) at all MACs in a given clock cycle
- Pixels are reused by movements among register arrays.

Input feature map with zero padding

0	0	0	0	0	0	0	0
0	11	12	13	14	15	16	0
0	21	22	23	24	25	26	0
0	31	32	33	34	35	36	0
0	41	42	43	44	45	46	0
0	51	52	53	54	55	56	0
0	61	62	63	64	65	66	0
0	0	0	0	0	0	0	0

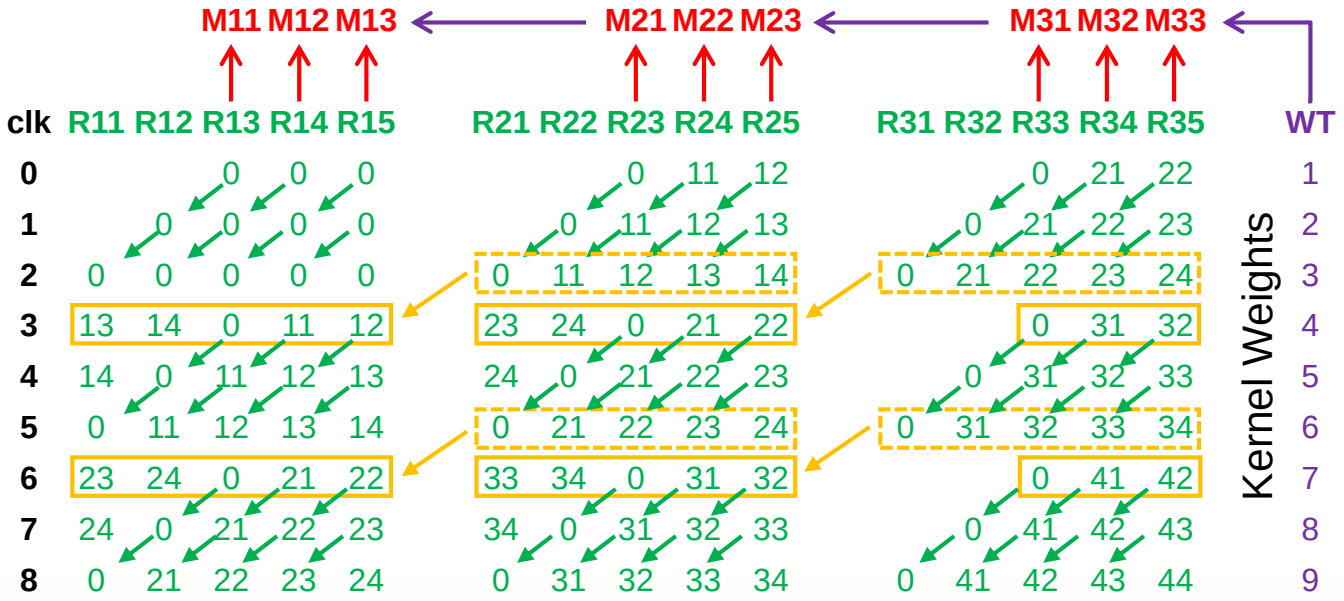
: output pixel location

Kernel Map

1	2	3
4	5	6
7	8	9

M* denotes MAC units

Green pixels are in registers (**R**)



Proposed Convolution Dataflow

- Same weight is used (multiplied) at all MACs in a given clock cycle
- Pixels are reused by movements among register arrays.

Input feature map with zero padding

Pix

Piy								
	0	0	0	0	0	0	0	0
	0	11	12	13	14	15	16	0
	0	21	22	23	24	25	26	0
	0	31	32	33	34	35	36	0
	0	41	42	43	44	45	46	0
	0	51	52	53	54	55	56	0
	0	61	62	63	64	65	66	0
	0	0	0	0	0	0	0	0

×

Kernel Map

1	2	3
4	5	6
7	8	9

Nkx = Nky = 3

Pkx = *Pky* = 1

Pix = *Piy* = 3

M* denotes MAC units

Green pixels are in registers (**R**) Blue pixels are in buffers (**BUF**)

	M11	M12	M13			M21	M22	M23			M31	M32	M33			
clk	**R11**	**R12**	**R13**	**R14**	**R15**	**R21**	**R22**	**R23**	**R24**	**R25**	**R31**	**R32**	**R33**	**R34**	**R35**	**WT**
0			0	0	0			0	11	12			0	21	22	1
1		0	0	0	0		0	11	12	13		0	21	22	23	2
2	0	0	0	0	0	0	11	12	13	14	0	21	22	23	24	3
3	13	14	0	11	12	23	24	0	21	22			0	31	32	4
4	14	0	11	12	13	24	0	21	22	23		0	31	32	33	5
5	0	11	12	13	14	0	21	22	23	24	0	31	32	33	34	6
6	23	24	0	21	22	33	34	0	31	32			0	41	42	7
7	24	0	21	22	23	34	0	31	32	33		0	41	42	43	8
8	0	21	22	23	24	0	31	32	33	34	0	41	42	43	44	9

Kernel Weights

BUF address

0	0	0	0			16	11	12			26	21	22
1	0	0	0			13	14	15			23	24	25
2	36	31	32			46	41	42			56	51	52
3	33	34	35			43	44	45			53	54	55

offset by zero padding

Input Pixel BUF 1 **Input Pixel BUF 2** **Input Pixel BUF 3**

Proposed Convolution Dataflow

- Same weight is used (multiplied) at all MACs in a given clock cycle
- Pixels are reused by movements among register arrays.

Input feature map with zero padding

Pix

Piy								
	0	0	0	0	0	0	0	0
	0	11	12	13	14	15	16	0
	0	21	22	23	24	25	26	0
	0	31	32	33	34	35	36	0
	0	41	42	43	44	45	46	0
	0	51	52	53	54	55	56	0
	0	61	62	63	64	65	66	0
	0	0	0	0	0	0	0	0

×

Kernel Map

1	2	3
4	5	6
7	8	9

Nkx = Nky = 3

Pkx = *Pky* = 1

Pix = *Piy* = 3

M* denotes MAC units

Green pixels are in registers (**R**) Blue pixels are in buffers (**BUF**)

			M11	**M12**	**M13**			**M21**	**M22**	**M23**			**M31**	**M32**	**M33**	
clk	**R11**	**R12**	**R13**	**R14**	**R15**		**R21**	**R22**	**R23**	**R24**	**R25**	**R31**	**R32**	**R33**	**R34**	**R35**	**WT**
0			0	0	0				0	11	12			0	21	22	1
1		0	0	0	0			0	11	12	13		0	21	22	23	2
2	0	0	0	0	0		0	11	12	13	14	0	21	22	23	24	3
3	13	14	0	11	12		23	24	0	21	22			0	31	32	4
4	14	0	11	12	13		24	0	21	22	23		0	31	32	33	5
5	0	11	12	13	14		0	21	22	23	24	0	31	32	33	34	6
6	23	24	0	21	22		33	34	0	31	32			0	41	42	7
7	24	0	21	22	23		34	0	31	32	33		0	41	42	43	8
8	0	21	22	23	24		0	31	32	33	34	0	41	42	43	44	9

Kernel Weights

BUF address

0	0	0	0				16	11	12				26	21	22	
1	0	0	0				13	14	15				23	24	25	
2	36	31	32				46	41	42				56	51	52	
3	33	34	35				43	44	45				53	54	55	

offset by zero padding

Input Pixel BUF 1 **Input Pixel BUF 2** **Input Pixel BUF 3**

Proposed Convolution Dataflow

- Same weight is used (multiplied) at all MACs in a given clock cycle
- Pixels are reused by movements among register arrays.

Input feature map with zero padding

Pix

Piy								
	0	0	0	0	0	0	0	0
	0	11	12	13	14	15	16	0
	0	21	22	23	24	25	26	0
	0	31	32	33	34	35	36	0
	0	41	42	43	44	45	46	0
	0	51	52	53	54	55	56	0
	0	61	62	63	64	65	66	0
	0	0	0	0	0	0	0	0

×

Kernel Map

1	2	3
4	5	6
7	8	9

Nkx = Nky = 3

Pkx = *Pky* = 1

Pix = *Piy* = 3

M* denotes MAC units

Green pixels are in registers (**R**) Blue pixels are in buffers (**BUF**)

	M11	M12	M13			M21	M22	M23			M31	M32	M33			
clk	**R11**	**R12**	**R13**	**R14**	**R15**	**R21**	**R22**	**R23**	**R24**	**R25**	**R31**	**R32**	**R33**	**R34**	**R35**	**WT**
0			0	0	0			0	11	12			0	21	22	1
1		0	0	0	0		0	11	12	13		0	21	22	23	2
2	0	0	0	0	0	0	11	12	13	14	0	21	22	23	24	3
3	13	14	0	11	12	23	24	0	21	22			0	31	32	4
4	14	0	11	12	13	24	0	21	22	23		0	31	32	33	5
5	0	11	12	13	14	0	21	22	23	24	0	31	32	33	34	6
6	23	24	0	21	22	33	34	0	31	32			0	41	42	7
7	24	0	21	22	23	34	0	31	32	33		0	41	42	43	8
8	0	21	22	23	24	0	31	32	33	34	0	41	42	43	44	9

Kernel Weights

BUF address

0	0	0	0	16	11	12	26	21	22
1	0	0	0	13	14	15	23	24	25
2	36	31	32	46	41	42	56	51	52
3	33	34	35	43	44	45	53	54	55

offset by zero padding

Input Pixel BUF 1 **Input Pixel BUF 2** **Input Pixel BUF 3**

Proposed Convolution Dataflow

- Same weight is used (multiplied) at all MACs in a given clock cycle
- Pixels are reused by movements among register arrays.

Input feature map with zero padding

Pix

Piy								
	0	0	0	0	0	0	0	0
	0	11	12	13	14	15	16	0
	0	21	22	23	24	25	26	0
	0	31	32	33	34	35	36	0
	0	41	42	43	44	45	46	0
	0	51	52	53	54	55	56	0
	0	61	62	63	64	65	66	0
	0	0	0	0	0	0	0	0

×

Kernel Map

1	2	3
4	5	6
7	8	9

Nkx = Nky = 3

Pkx = *Pky* = 1

Pix = *Piy* = 3

M* denotes MAC units

Green pixels are in registers (**R**) Blue pixels are in buffers (**BUF**)

clk	R11	R12	R13	R14	R15	R21	R22	R23	R24	R25	R31	R32	R33	R34	R35	WT
0			0	0	0			0	11	12			0	21	22	1
1		0	0	0	0		0	11	12	13		0	21	22	23	2
2	0	0	0	0	0	0	11	12	13	14	0	21	22	23	24	3
3	13	14	0	11	12	23	24	0	21	22			0	31	32	4
4	14	0	11	12	13	24	0	21	22	23		0	31	32	33	5
5	0	11	12	13	14	0	21	22	23	24	0	31	32	33	34	6
6	23	24	0	21	22	33	34	0	31	32			0	41	42	7
7	24	0	21	22	23	34	0	31	32	33		0	41	42	43	8
8	0	21	22	23	24	0	31	32	33	34	0	41	42	43	44	9

Kernel Weights

BUF address

	Input Pixel BUF 1			Input Pixel BUF 2			Input Pixel BUF 3		
0	0	0	0	16	11	12	26	21	22
1	0	0	0	13	14	15	23	24	25
2	36	31	32	46	41	42	56	51	52
3	33	34	35	43	44	45	53	54	55

offset by zero padding

Proposed Convolution Dataflow

- Same weight is used (multiplied) at all MACs in a given clock cycle
- Pixels are reused by movements among register arrays.

Input feature map with zero padding

Pix

Piy								
	0	0	0	0	0	0	0	0
	0	11	12	13	14	15	16	0
	0	21	22	23	24	25	26	0
	0	31	32	33	34	35	36	0
	0	41	42	43	44	45	46	0
	0	51	52	53	54	55	56	0
	0	61	62	63	64	65	66	0
	0	0	0	0	0	0	0	0

×

Kernel Map

1	2	3
4	5	6
7	8	9

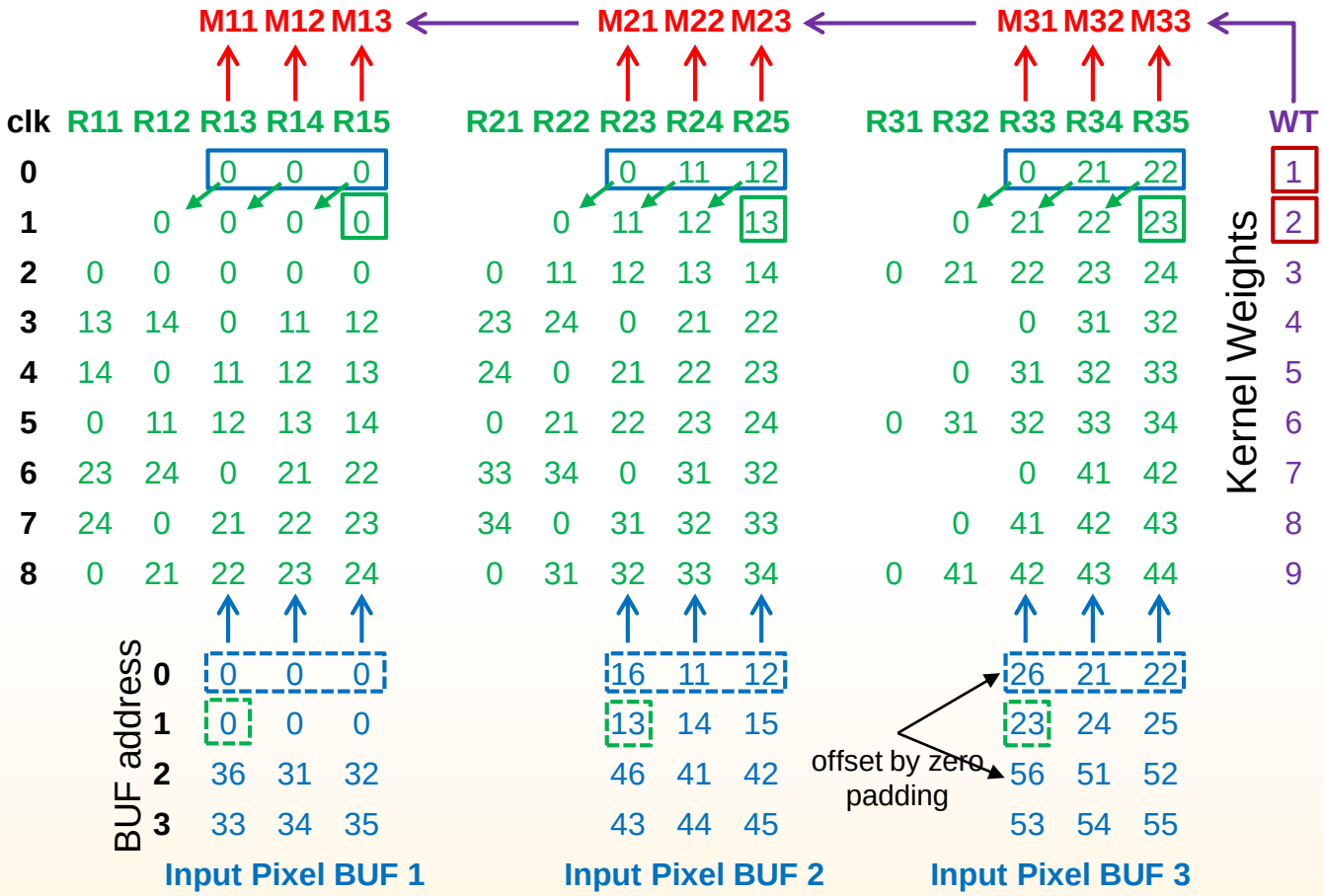
Nkx = Nky = 3

Pkx = *Pky* = 1

Pix = *Piy* = 3

M* denotes MAC units

Green pixels are in registers (**R**) Blue pixels are in buffers (**BUF**)



Proposed Convolution Dataflow

- Same weight is used (multiplied) at all MACs in a given clock cycle
- Pixels are reused by movements among register arrays.

Input feature map with zero padding

Pix

Piy								
	0	0	0	0	0	0	0	0
	0	11	12	13	14	15	16	0
	0	21	22	23	24	25	26	0
	0	31	32	33	34	35	36	0
	0	41	42	43	44	45	46	0
	0	51	52	53	54	55	56	0
	0	61	62	63	64	65	66	0
	0	0	0	0	0	0	0	0

×

Kernel Map

1	2	3
4	5	6
7	8	9

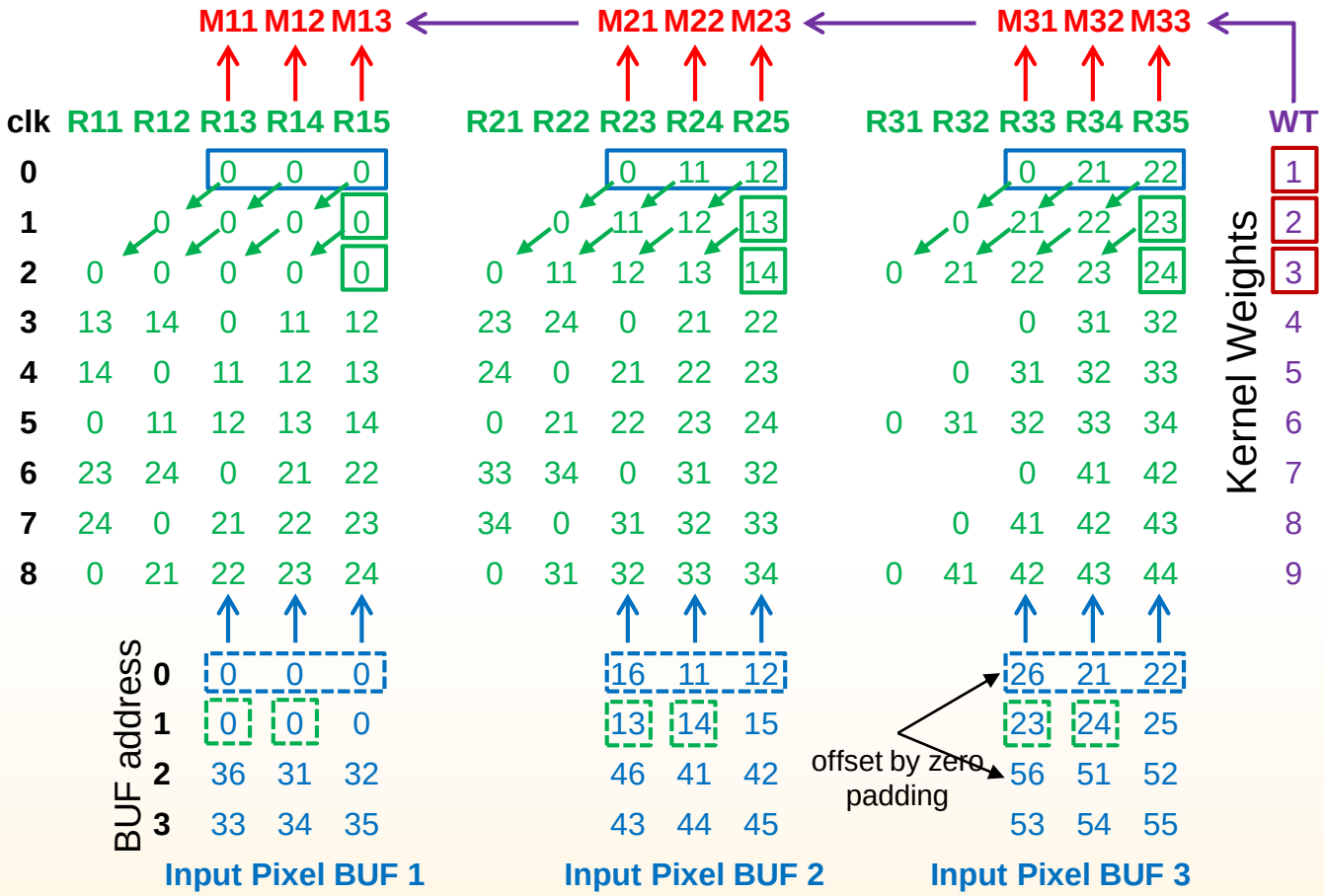
Nkx = Nky = 3

Pkx = *Pky* = 1

Pix = *Piy* = 3

M* denotes MAC units

Green pixels are in registers (**R**) Blue pixels are in buffers (**BUF**)



Proposed Convolution Dataflow

- Same weight is used (multiplied) at all MACs in a given clock cycle
- Pixels are reused by movements among register arrays.

Input feature map with zero padding

Pix

Piy								
	0	0	0	0	0	0	0	0
	0	11	12	13	14	15	16	0
	0	21	22	23	24	25	26	0
	0	31	32	33	34	35	36	0
	0	41	42	43	44	45	46	0
	0	51	52	53	54	55	56	0
	0	61	62	63	64	65	66	0
	0	0	0	0	0	0	0	0

×

Kernel Map

1	2	3
4	5	6
7	8	9

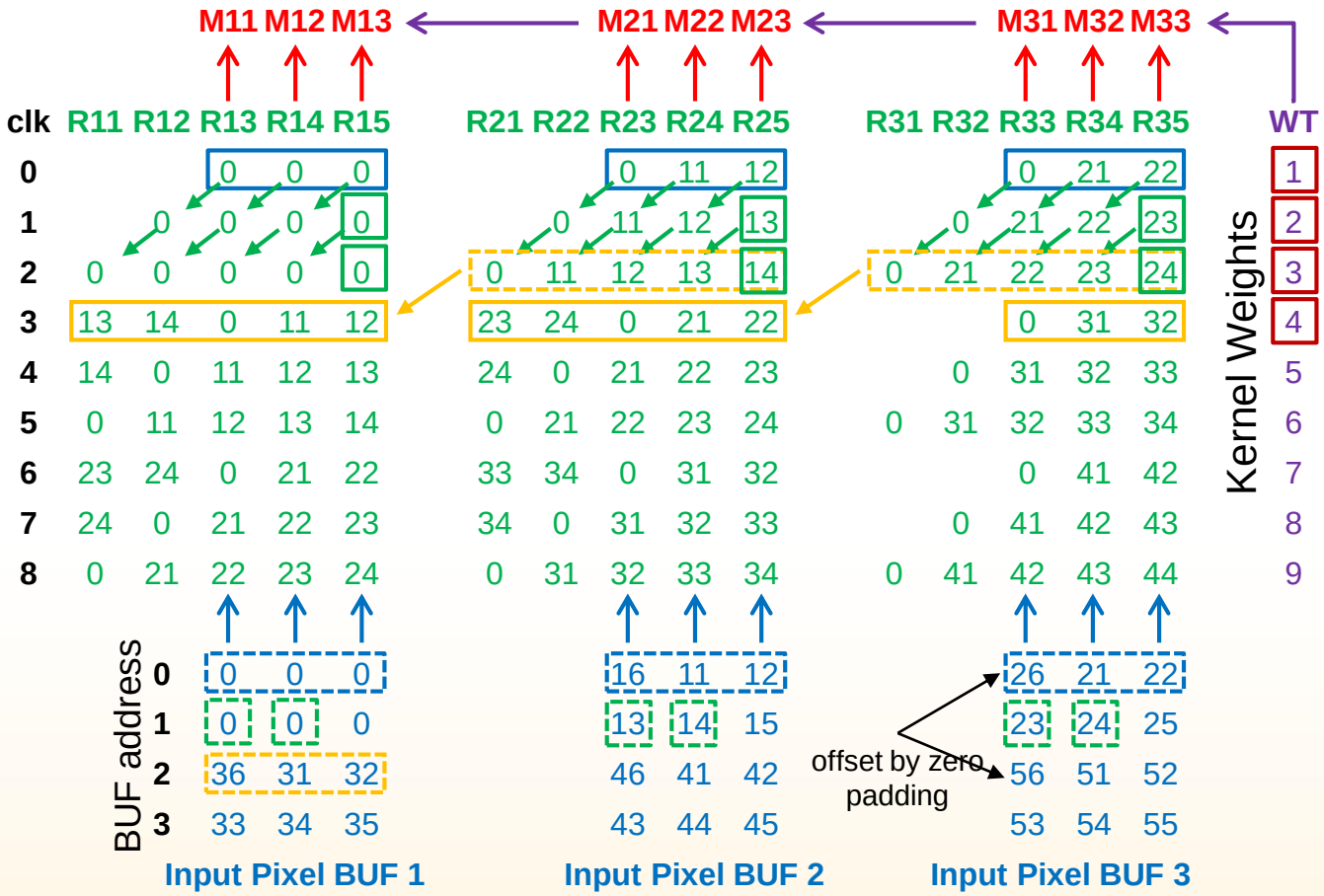
Nkx = Nky = 3

Pkx = *Pky* = 1

Pix = *Piy* = 3

M* denotes MAC units

Green pixels are in registers (**R**) Blue pixels are in buffers (**BUF**)



Proposed Convolution Dataflow

- Same weight is used (multiplied) at all MACs in a given clock cycle
- Pixels are reused by movements among register arrays.

Input feature map with zero padding

Pix

Piy								
	0	0	0	0	0	0	0	0
	0	11	12	13	14	15	16	0
	0	21	22	23	24	25	26	0
	0	31	32	33	34	35	36	0
	0	41	42	43	44	45	46	0
	0	51	52	53	54	55	56	0
	0	61	62	63	64	65	66	0
	0	0	0	0	0	0	0	0

×

Kernel Map

1	2	3
4	5	6
7	8	9

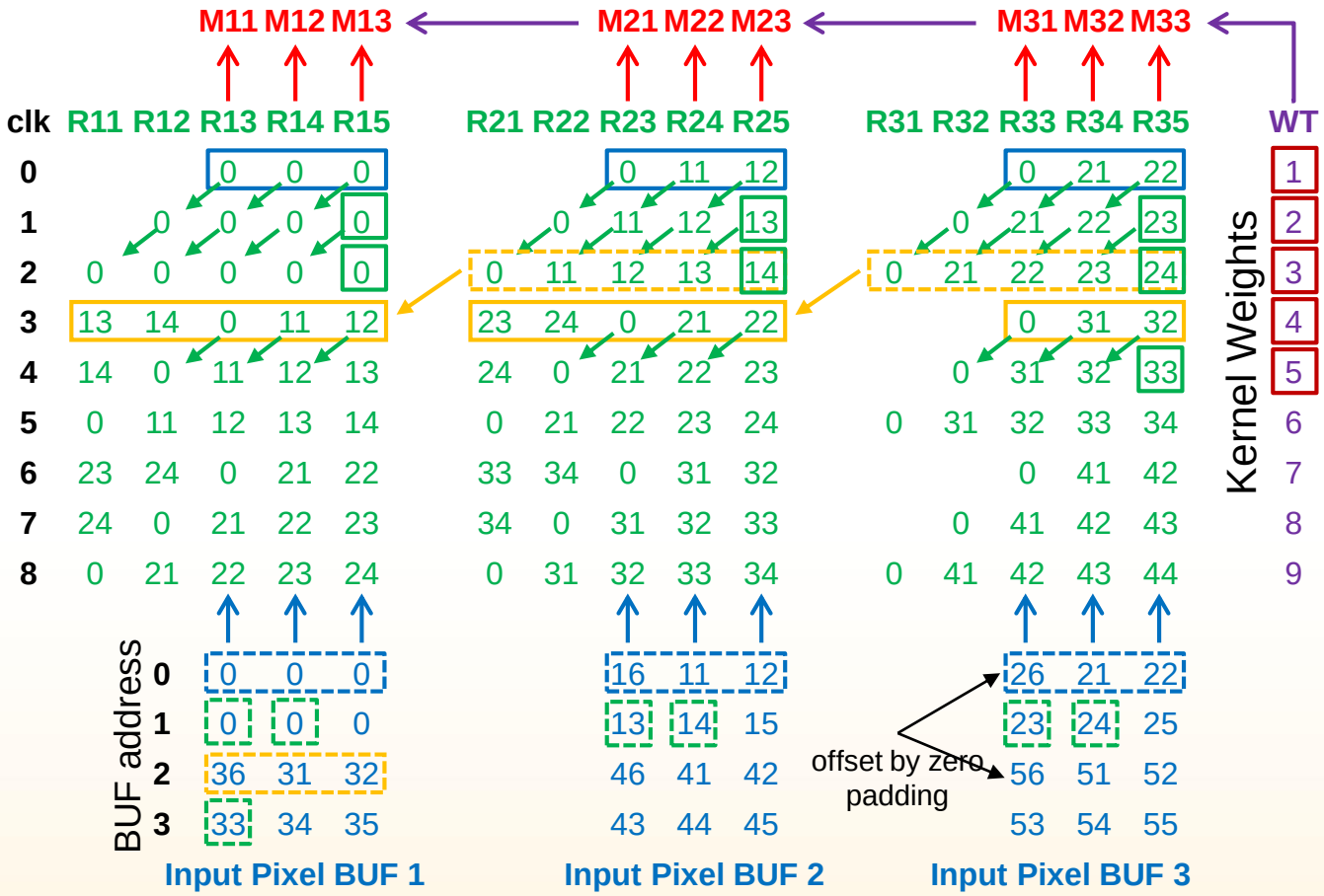
Nkx = Nky = 3

Pkx = *Pky* = 1

Pix = *Piy* = 3

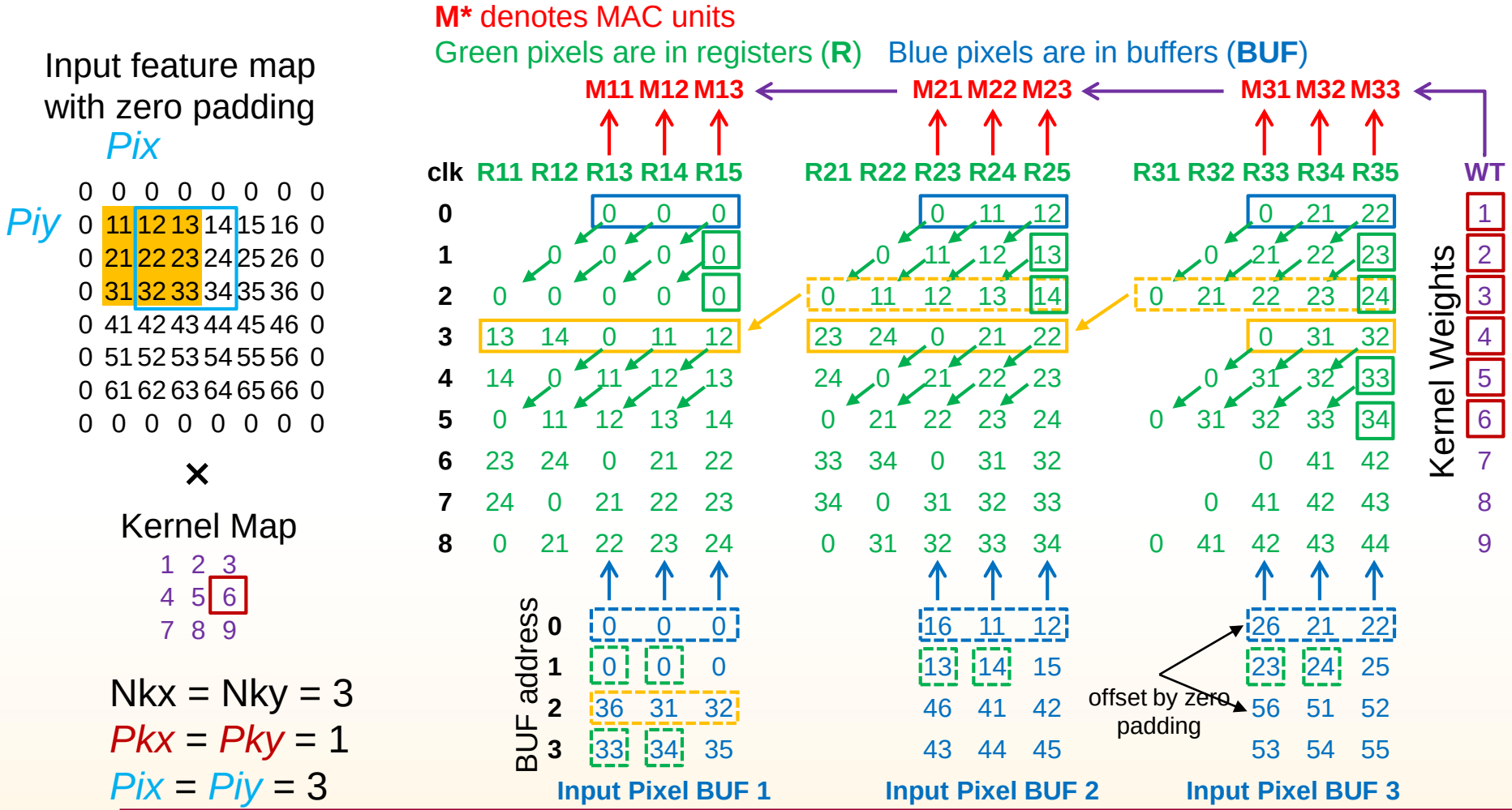
M* denotes MAC units

Green pixels are in registers (**R**) Blue pixels are in buffers (**BUF**)



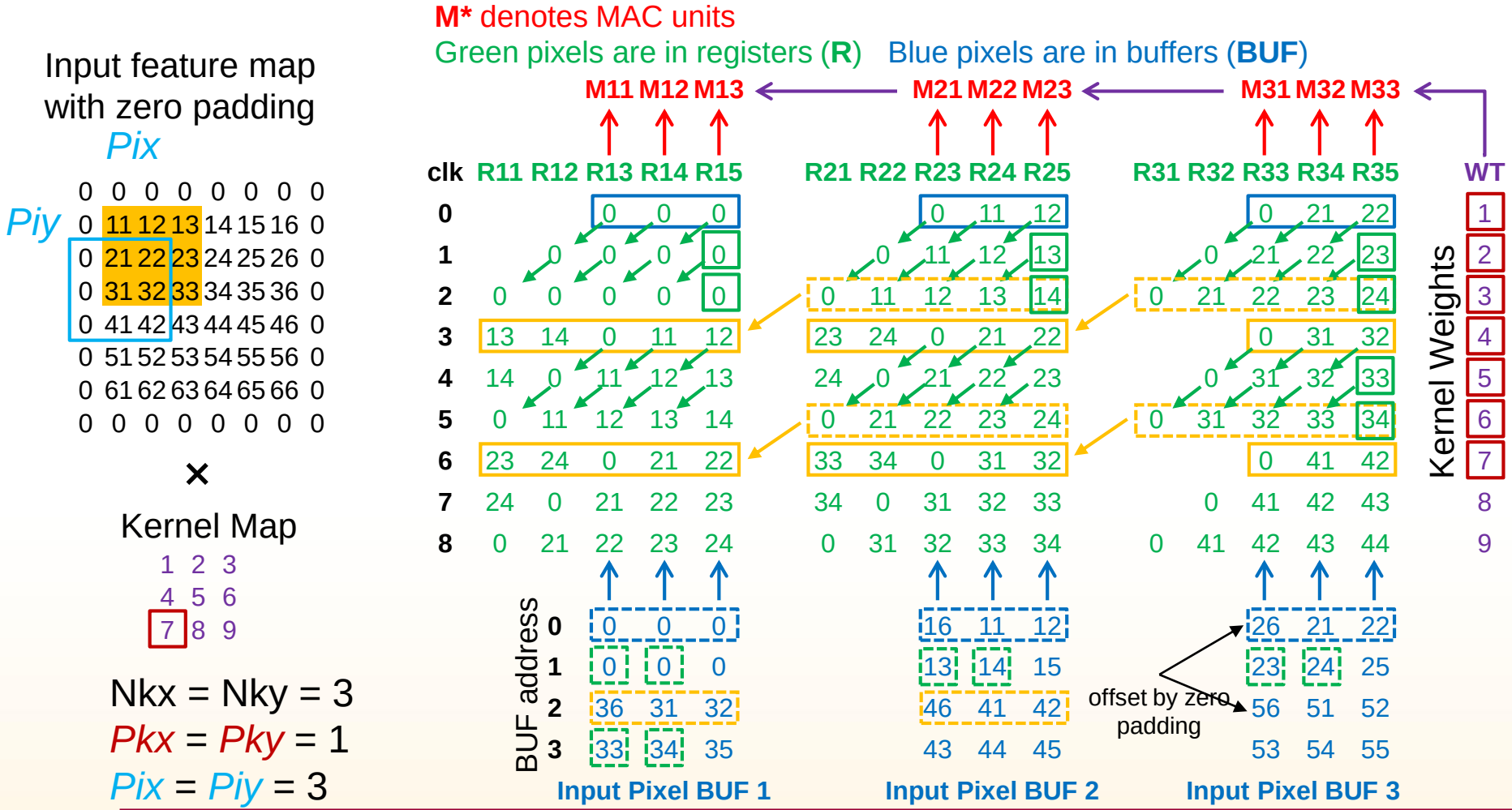
Proposed Convolution Dataflow

- Same weight is used (multiplied) at all MACs in a given clock cycle
- Pixels are reused by movements among register arrays.



Proposed Convolution Dataflow

- Same weight is used (multiplied) at all MACs in a given clock cycle
- Pixels are reused by movements among register arrays.



Proposed Convolution Dataflow

- Same weight is used (multiplied) at all MACs in a given clock cycle
- Pixels are reused by movements among register arrays.

Input feature map with zero padding

Pix

Piy								
	0	0	0	0	0	0	0	0
	0	11	12	13	14	15	16	0
	0	21	22	23	24	25	26	0
	0	31	32	33	34	35	36	0
	0	41	42	43	44	45	46	0
	0	51	52	53	54	55	56	0
	0	61	62	63	64	65	66	0
	0	0	0	0	0	0	0	0

×

Kernel Map

1	2	3
4	5	6
7	8	9

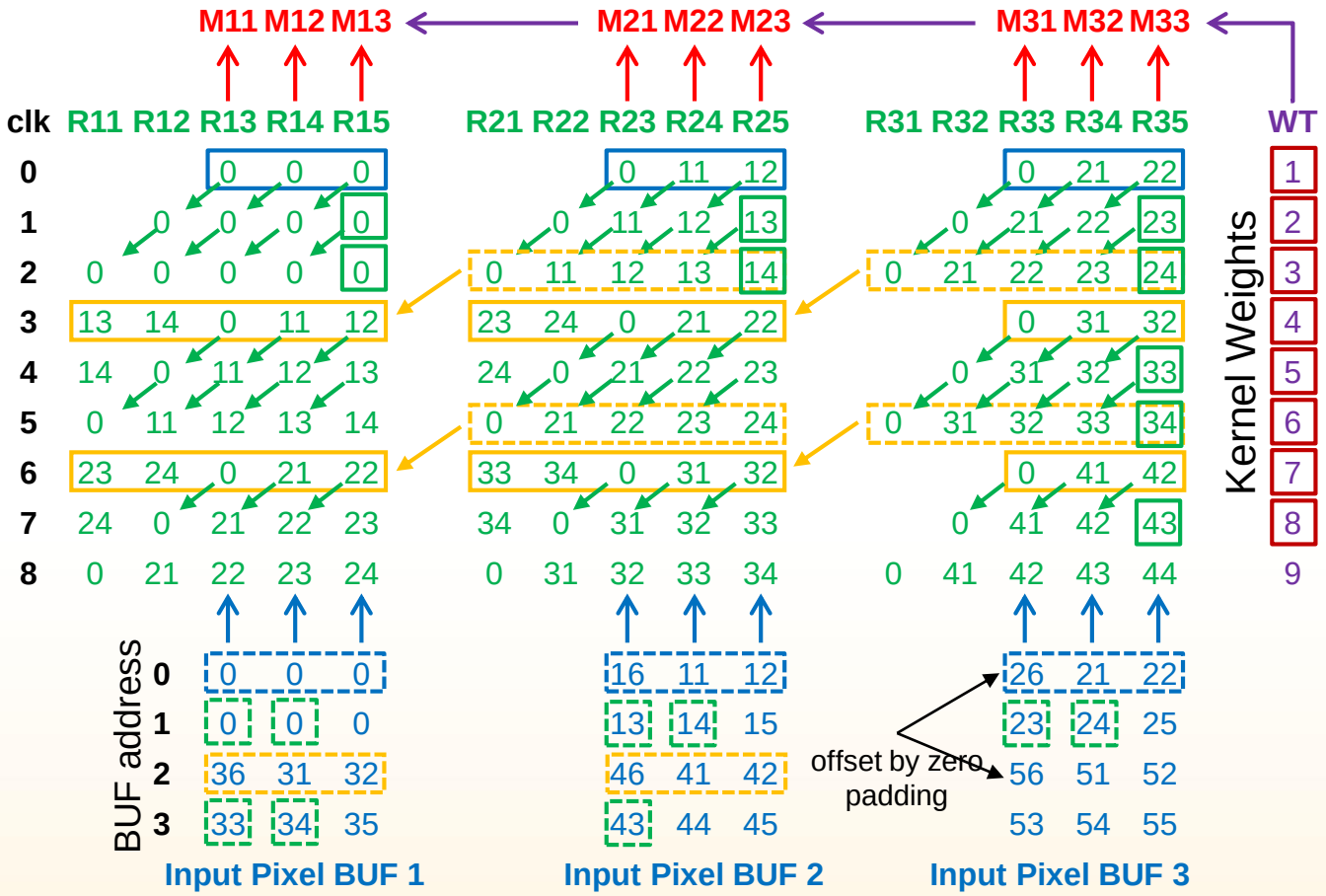
Nkx = Nky = 3

Pkx = *Pky* = 1

Pix = *Piy* = 3

M* denotes MAC units

Green pixels are in registers (**R**) Blue pixels are in buffers (**BUF**)



Proposed Convolution Dataflow

- Same weight is used (multiplied) at all MACs in a given clock cycle
- Pixels are reused by movements among register arrays.

Input feature map with zero padding

Pix

Piy								
	0	0	0	0	0	0	0	0
	0	11	12	13	14	15	16	0
	0	21	22	23	24	25	26	0
	0	31	32	33	34	35	36	0
	0	41	42	43	44	45	46	0
	0	51	52	53	54	55	56	0
	0	61	62	63	64	65	66	0
	0	0	0	0	0	0	0	0

×

Kernel Map

1	2	3
4	5	6
7	8	9

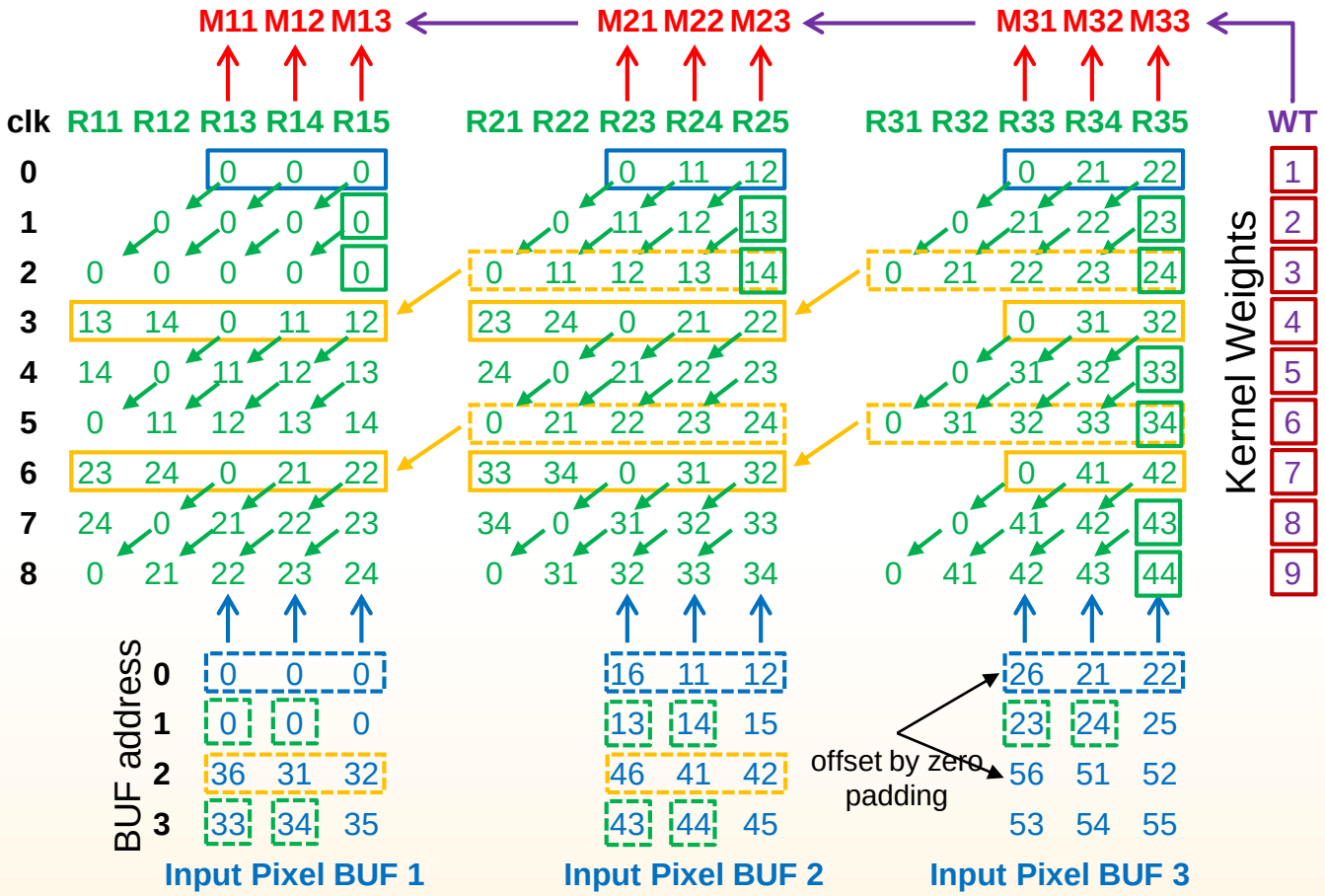
Nkx = Nky = 3

Pkx = *Pky* = 1

Pix = *Piy* = 3

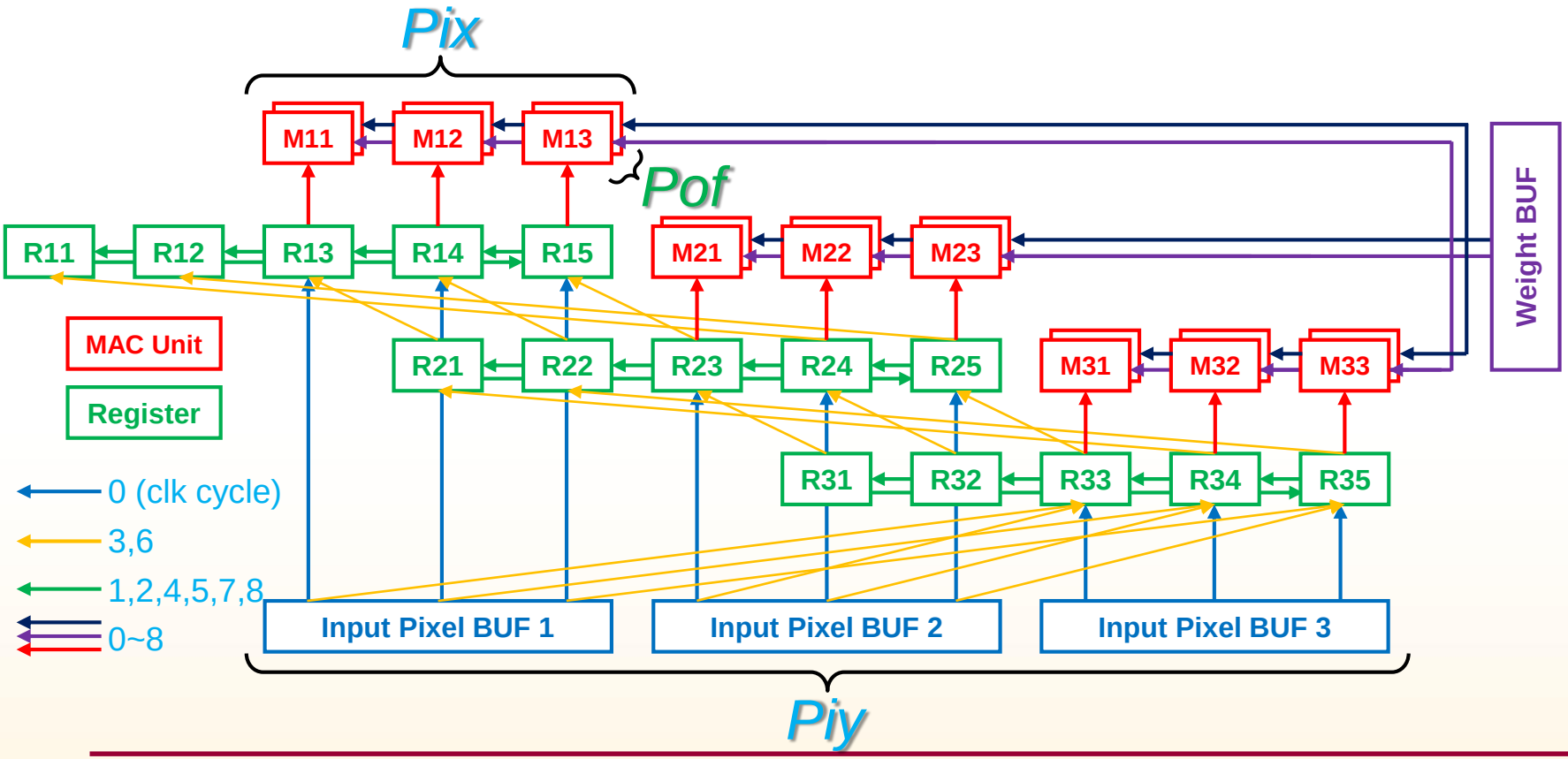
M* denotes MAC units

Green pixels are in registers (**R**) Blue pixels are in buffers (**BUF**)



Convolution PE Architecture

- *Pix* (14) × *Piy* (14) × *Pof* (16) independent PEs (MAC unit).
- Pixels / Weights shared by *Pof* / *Pix*×*Piy* PEs, respectively.
- Partial sum is consumed inside each MAC unit.

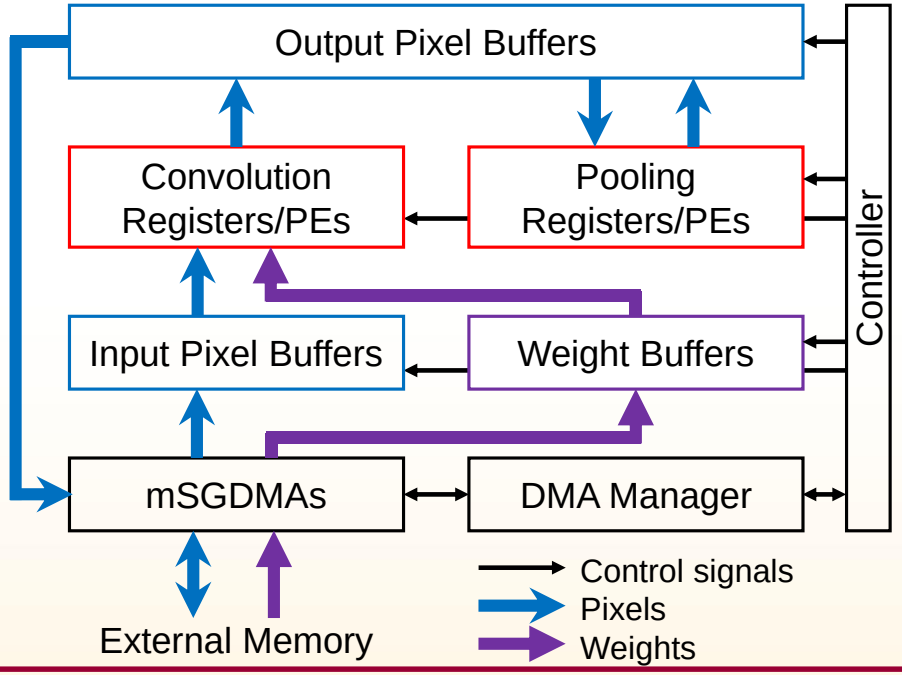


Outline

- Overview of CNN Algorithm and Accelerator
- Convolution Loop Optimization
- Design Objectives of CNN Accelerator
- Loop Optimization in Related Works
- Proposed CNN Accelerator
- Experimental Results
- Conclusion

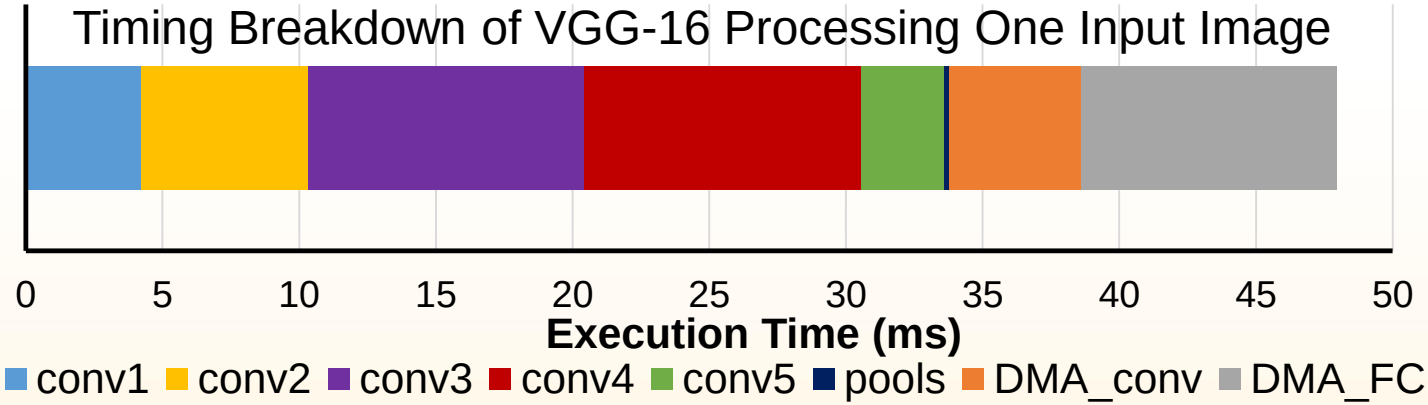
Experiment System Setup

- Altera Arria 10 GX 1150 FPGA + 2 × 4GB DDR3L SDRAM
  - 1,150K logic elements, 1,518 DSP blocks, 2,713 M20K RAMs
- VGG-16 CNN Model
  - 13 convolution, 5 pooling, 3 FC layers, 138.3 million parameters
- Fixed point data with dynamically adjusted decimal points
  - 16-bit pixels, 8-bit weights, 30-bit partial sums.



Experimental Results

- End-to-end latency per image is 47.97 ms
  - Conv. computation (70%), DMA_conv (10%), DMA_FC (20%)
- 1,900 M20K RAMs are used
  - Conv. buffers (70.1%), FC buffers (11.9%), FIFO (3.1%), DMA (11.7%)
- Simulated thermal power is 21.2 W from PowerPlay
  - DSP (3.6%), M20K (15.0%), logic cells (16%), clock (12.9%), transceiver (15.7%), I/O (13.2%), static consumption (23.5%)



Comparison with Prior Works

	J. Qiu *FPGA'16*	N. Suda *FPGA'16*	C. Zhang *ISLPED'16*	**This Work**
CNN Model	VGG-16	VGG-16	VGG-16	**VGG-16**
FPGA	Zynq XC7Z045	Stratix-V GSD8	Virtex-7 VX690t	**Arria-10 GX 1150**
Frequency (MHz)	150	120	150	**150**
# Operations (GOP)	30.76	30.95	30.95	**30.95**
# of Weights	50.18 M	138.3 M	138.3 M	**138.3 M**
Precision	16 bit	8-16 bit	16 bit	**8-16 bit**
DSP Utilization	780 (89%)	-	-	**1,518 (100%)**
Logic Utilization[a]	183K (84%)	-	-	**161K (38%)**
On-chip RAM[b]	486 (87%)	-	-	**1,900 (70%)**
Latency/Image (ms)	**224.6**	**262.9**	**151.8**	**47.97**
Throughput (GOPS)	**136.97**	**117.8**	**203.9**	**645.25**

a. Xilinx FPGAs in LUTs and Altera FPGAs in ALMs
b. Xilinx FPGAs in BRAMs (36 Kb) and Altera FPGAs in M20K RAMs (20 Kb)

Outline

- Overview of CNN Algorithm and Accelerator
- Convolution Loop Optimization
- Design Objectives of CNN Accelerator
- Loop Optimization in Related Works
- Proposed CNN Accelerator
- Experimental Results
- Conclusion

Conclusion

- In-depth analysis of convolution acceleration strategy.
  - Numerically characterize the loop optimization techniques, including loop unrolling, tilling and interchange.
  - Quantitatively investigate the relationship between accelerator objectives and design variables.
- Propose efficient dataflow and architecture to minimize data communication and enhance throughput.
  - Implement VGG-16 on Arria 10 FPGA
  - End-to-end 645.25 GOPS of throughput and 47.97 ms of latency.
  - Outperform all prior VGG FPGA implementations by 3.2×.

Thanks!

Questions?