ASIA SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE

ConvFIFO: A Crossbar Memory PIM Architecture for ConvNets Featuring First-In-First-Out Dataflow

Liang Zhao[1#], Yu Qian[1#], Fanzi Meng[1],

Xiapeng Xu[1], Xunzhao Yin[1] and Cheng Zhuo[2]

[1]College of Information Science and Electronic Engineering, Zhejiang University
[2]School of Micro-Nano Electronics, Zhejiang University, Hangzhou, China

Outline

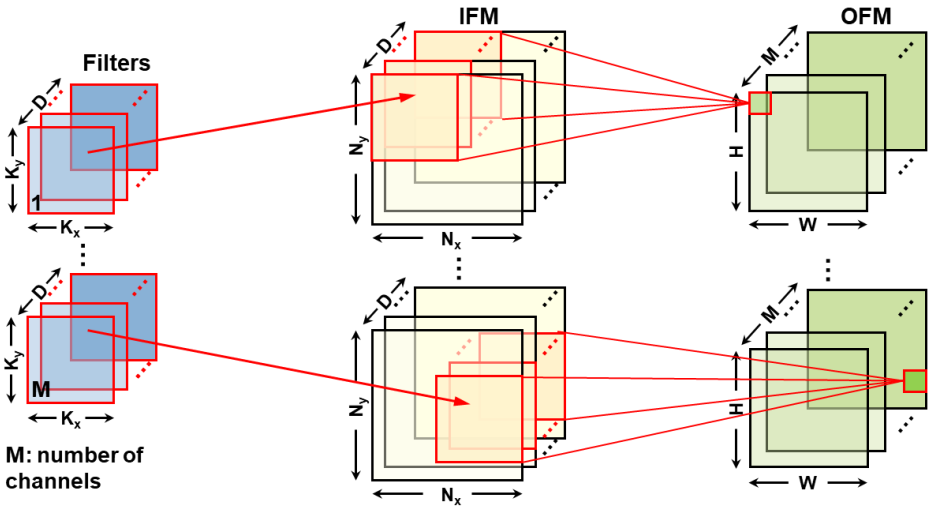
- Background and Motivations
- The ConvFIFO Architecture
- Evaluation and Benchmark of ConvFIFO
- Conclusions

Outline

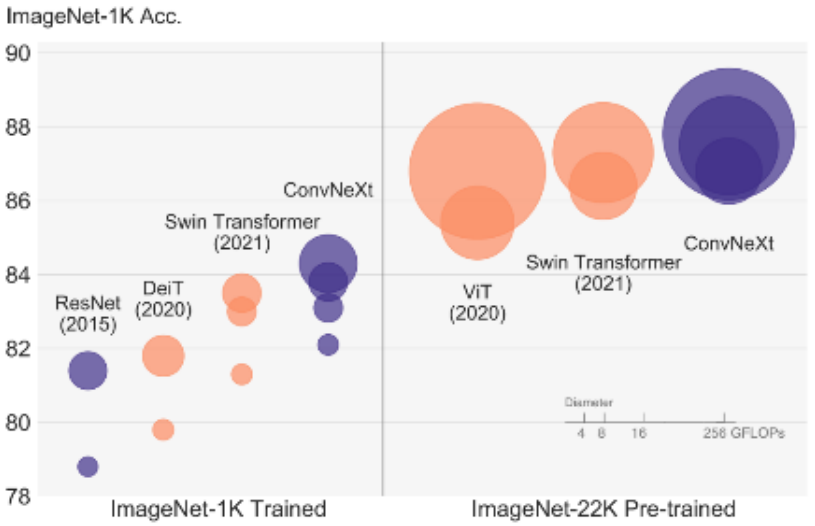
- **Background and Motivations**
- The ConvFIFO Architecture
- Evaluation and Benchmark of ConvFIFO
- Conclusions

Background

- Convolutional neural networks (**ConvNets**) saw a renaissance of interests in recent years, e.g. ConvNeXt (2022)
- For vision tasks, It can match state-of-the-art ViT in terms of accuracy, robustness and scalability, yet easier to implement



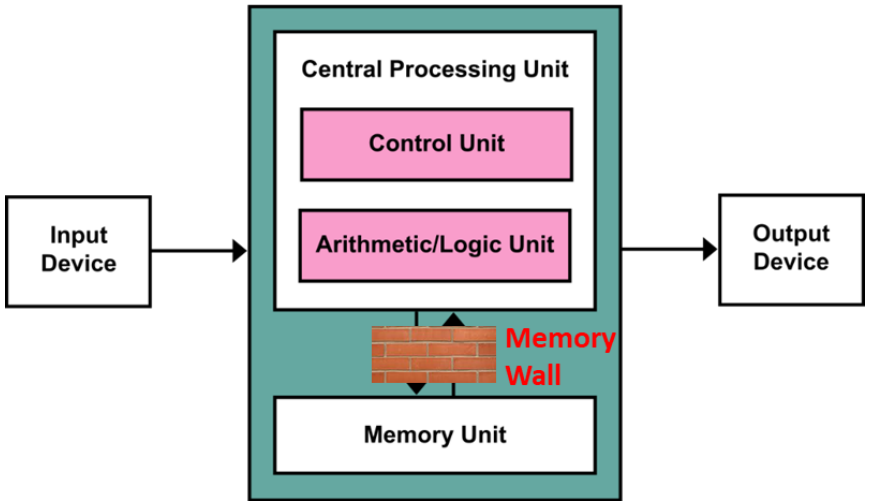
Structure of a single convolution layer in ConvNets



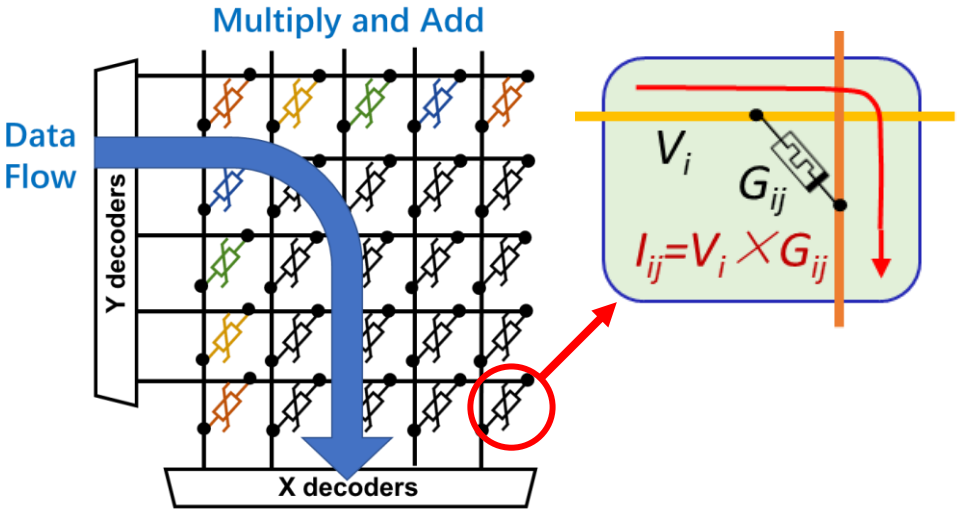
ConvNeXt accuracy benchmarks
Z. Liu et al., CVPR 2022

Background

- The hardware acceleration of ConvNets remains a crucial topic for edge computing, which suffers the “memory wall” problem in conventional von-Neumann architectures
- Process-in-memory (**PIM**) based on emerging memories (RRAM, PCM, MRAM, etc.) can circumvent the memory wall



“Memory wall” of von Neumann architectures

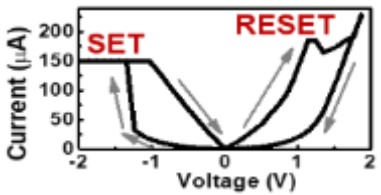


Crossbar-based PIM architecture

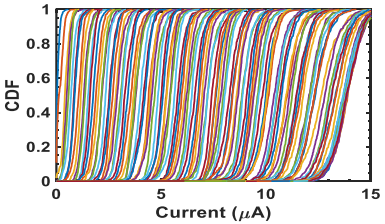
Motivation

- **RRAM-based PIM for ConvNets face challenges:**
  - PIM accuracy degradation due to device/circuit non-idealities
  - Limited throughput due to the sneak-path currents
  - RRAM buffer is not practical due to endurance concerns
  - High-precision ADCs for analog PIM is a resource bottleneck

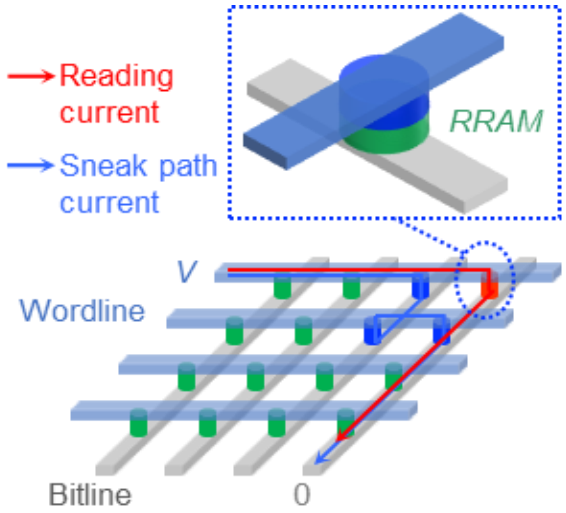
Non-linearity



Device variations



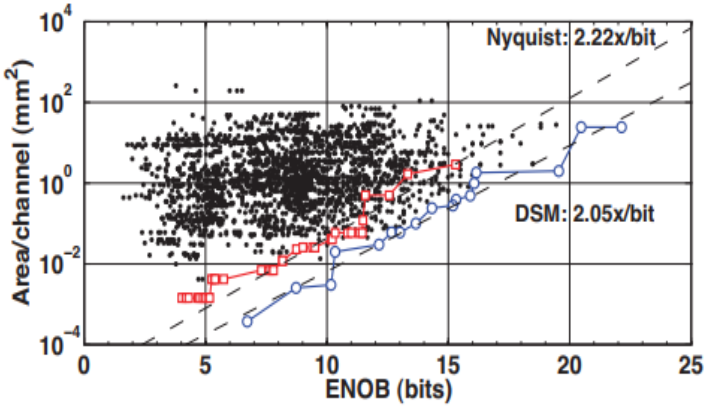
Device/circuits non-idealities



Sneak-path current in crossbar arrays

ADC area efficiency vs. ENOB

B. E. Jonsson, ECCTD 2011



High-res ADC area much larger than memory arrays

Outline

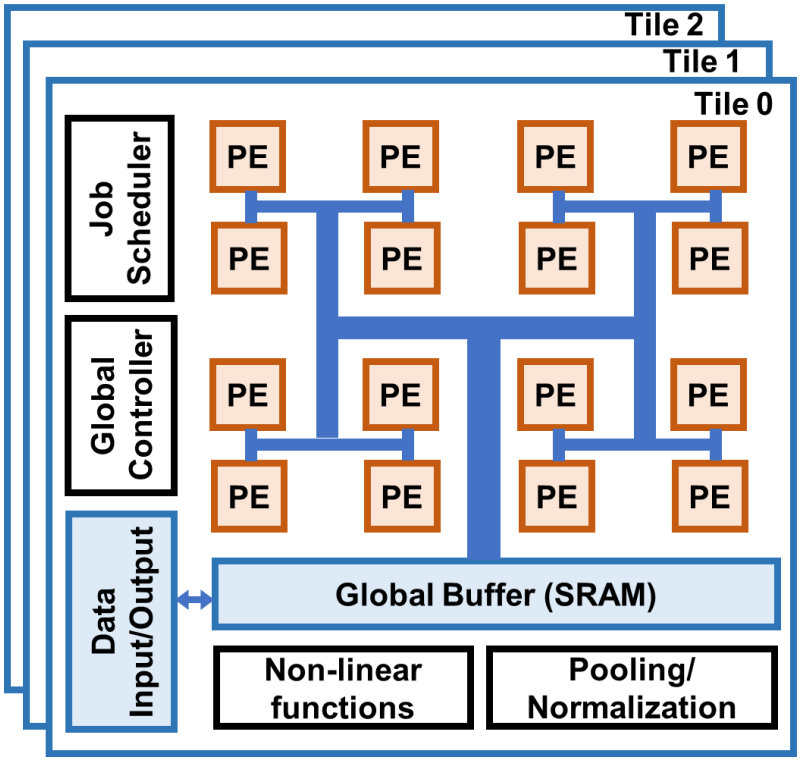
- Background and Motivations
- **The ConvFIFO Architecture**
- Evaluation and Benchmark of ConvFIFO
- Conclusions

Overview of ConvFIFO Architecture

- **Design philosophy of ConvFIFO**
  - Maximize the reuse rate of input/output data
  - Reduce the number of simultaneously-activated rows to minimize parasitic non-ideal effects (shorter BL)
  - Use **low-precision ADC** to boost area/power efficiency
  - Use **output FIFO** to store partial sums
  - Pixel-serial input/output for flexible pipeline design
- **Instead of sliding kernels, ConvFIFO slide the input images across the kernels to accelerate ConvNets**

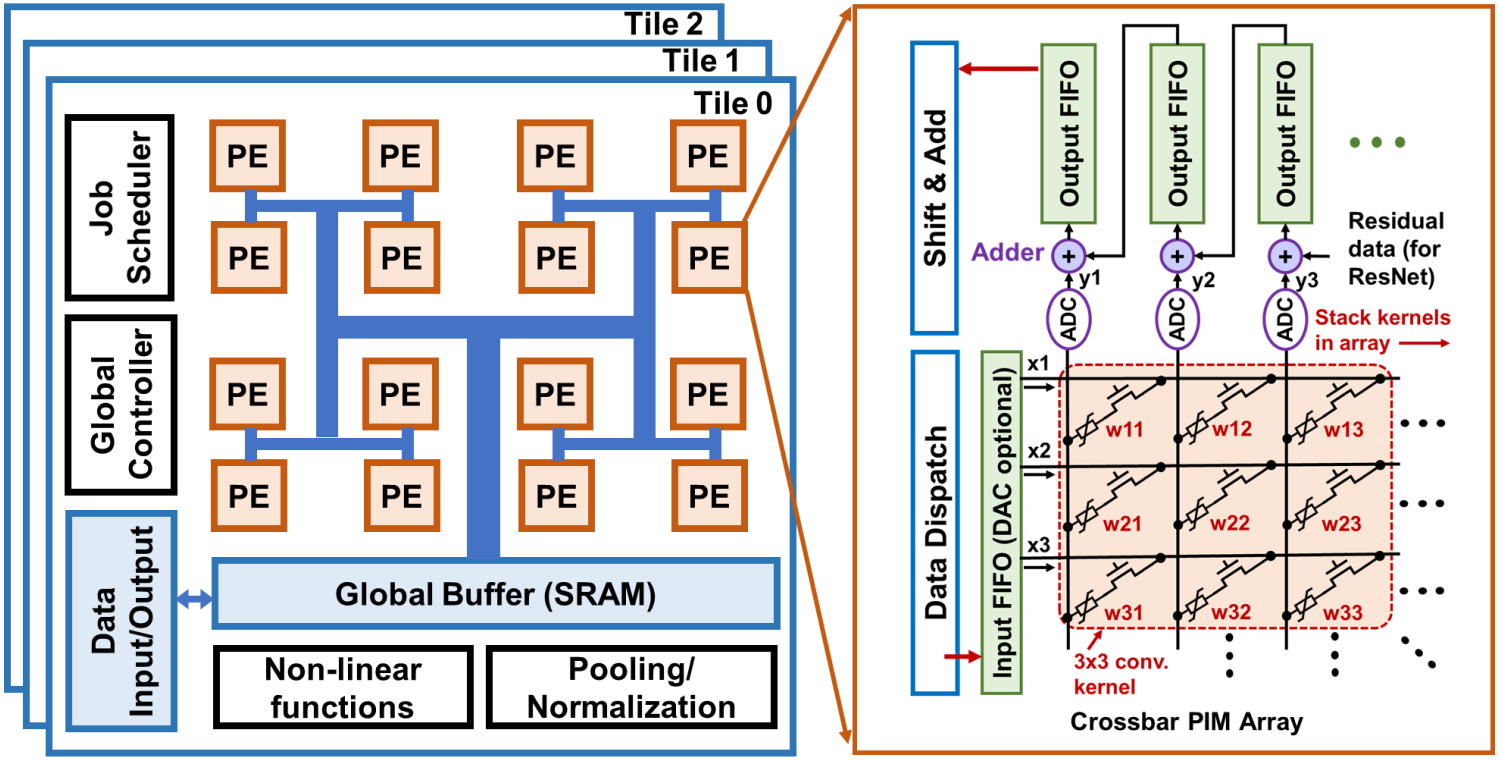
Overview of ConvFIFO Architecture

- **Tile design**
  - PIM process engines (**PEs**)
  - Global SRAM buffer
  - Control and math blocks
  - H-tree interconnects



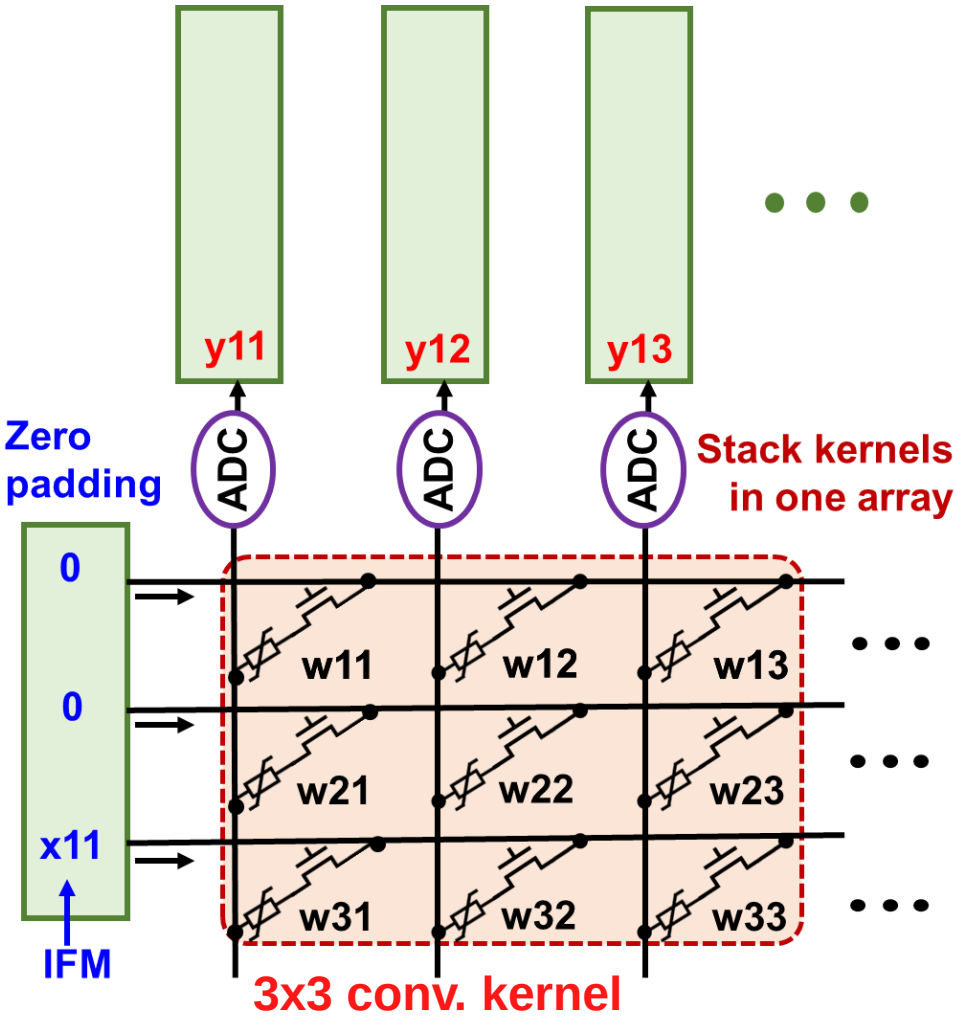
Overview of ConvFIFO Architecture

- **PE design**
  - Crossbar PIM array with input and output FIFO
  - ADC, decoders, peripherals, etc.



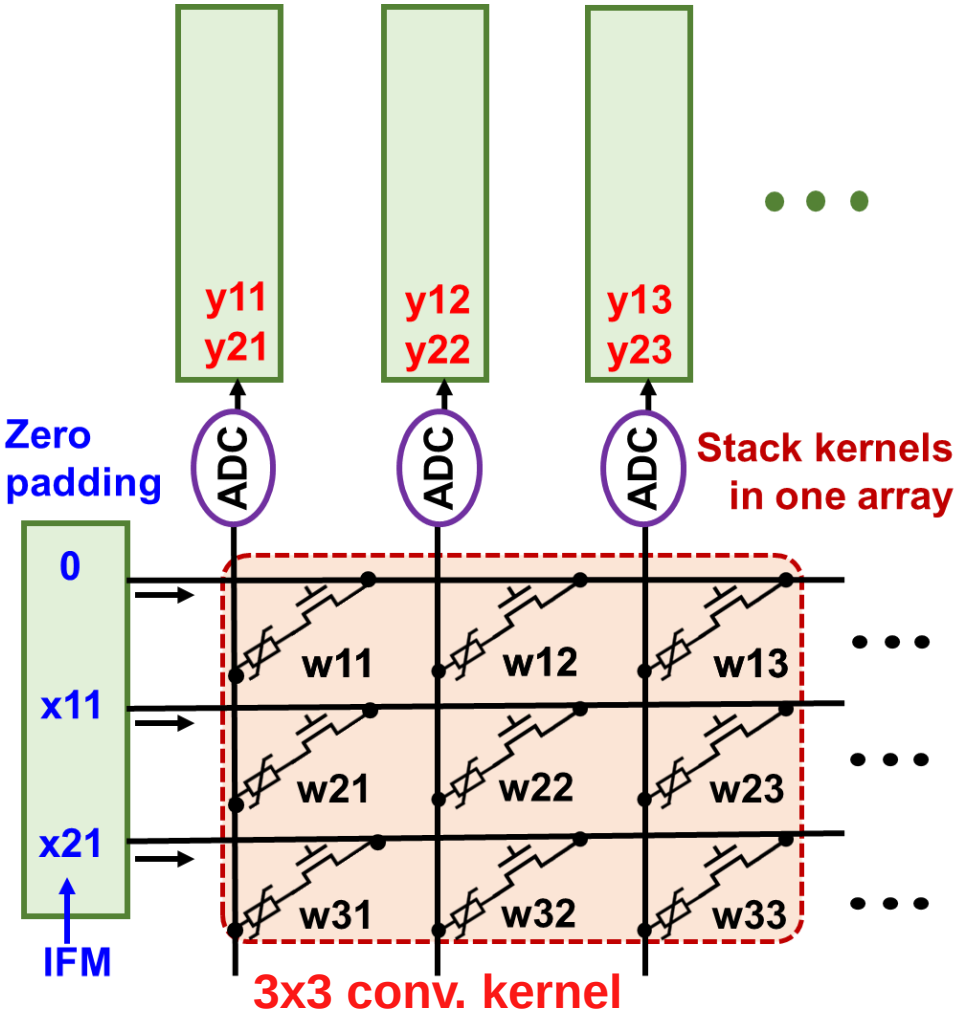
ConvFIFO Dataflow

- The weight data of the kernel are mapped into the array with the original 2D order
- The **IFM data** is fetched from previous layer into the input FIFO (starting with the first column of the IFM)
- The **partial sums** are computed by ADC and pushed into the output FIFO of the associated bitline



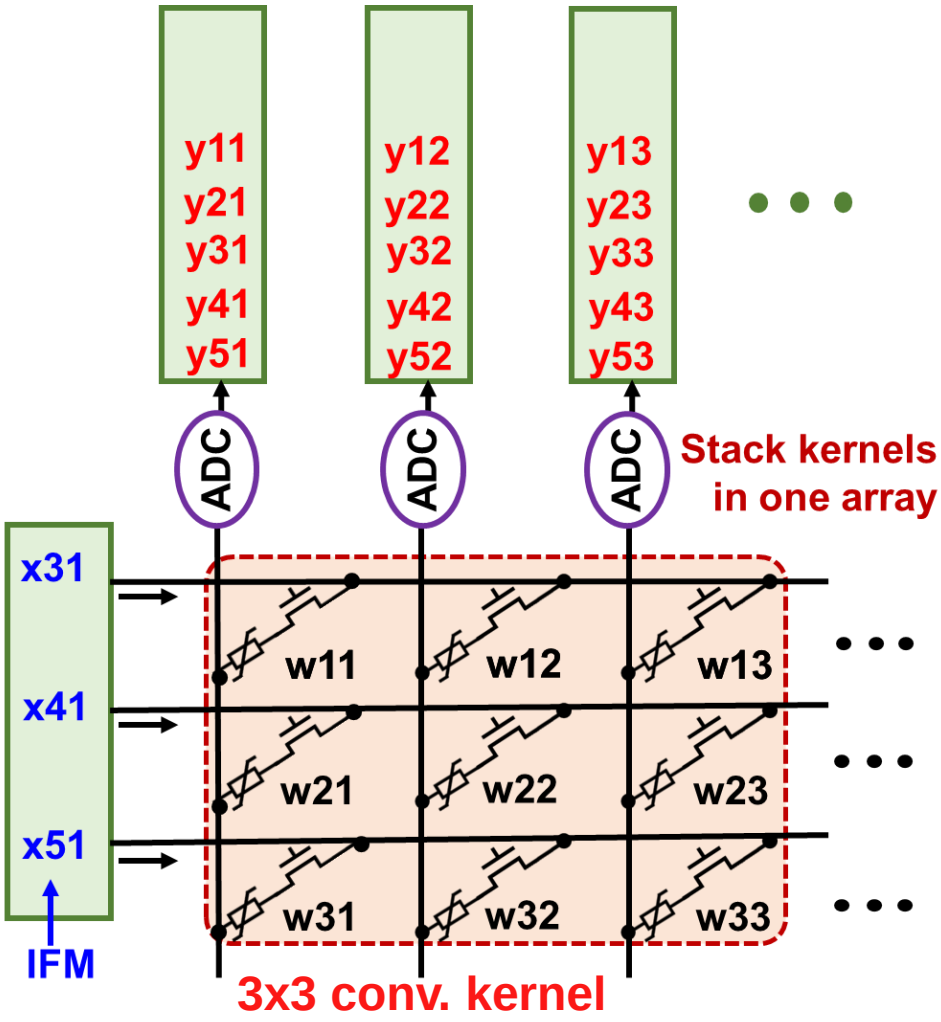
ConvFIFO Dataflow

- Fetch one more input from the **IFM** and push into the input FIFO
- The previous input in the FIFO are slided upwards
- Generate new **partial sums** and push into output FIFO



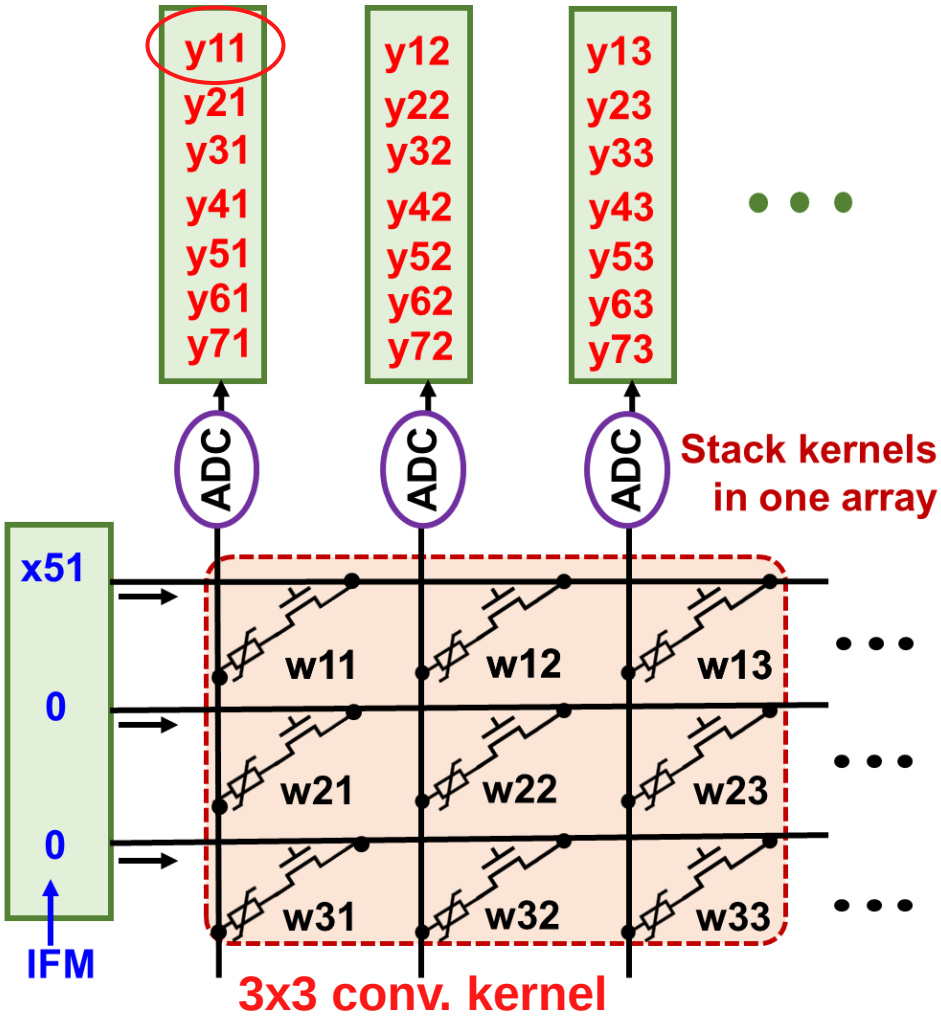
ConvFIFO Dataflow

- Repeat the above steps until …



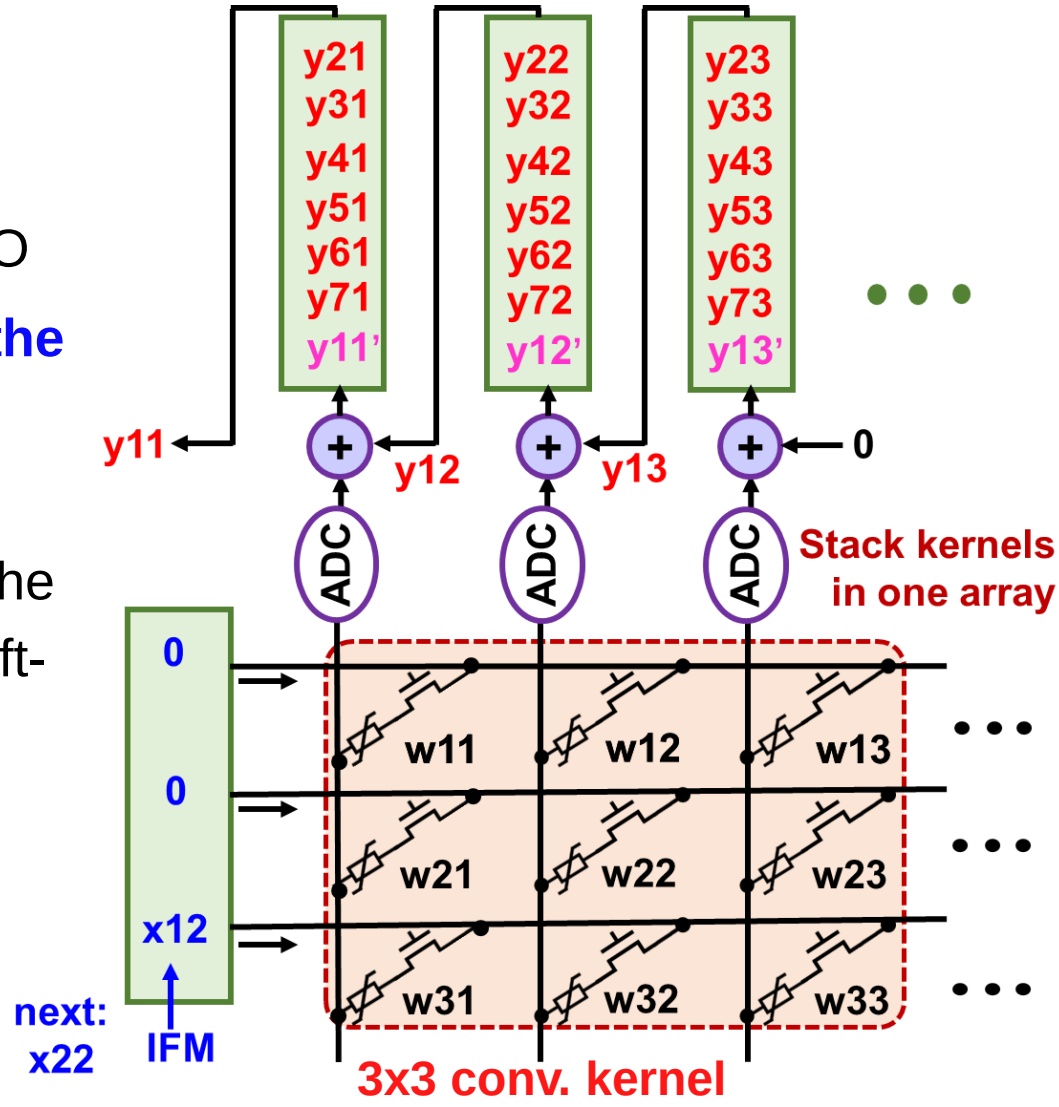
ConvFIFO Dataflow

- Repeat the above steps until the **first column of the IFM** has slided through the input FIFO
- Now, **the first data of the OFM** is at the top of the left-most FIFO, ready to be popped out for further processing



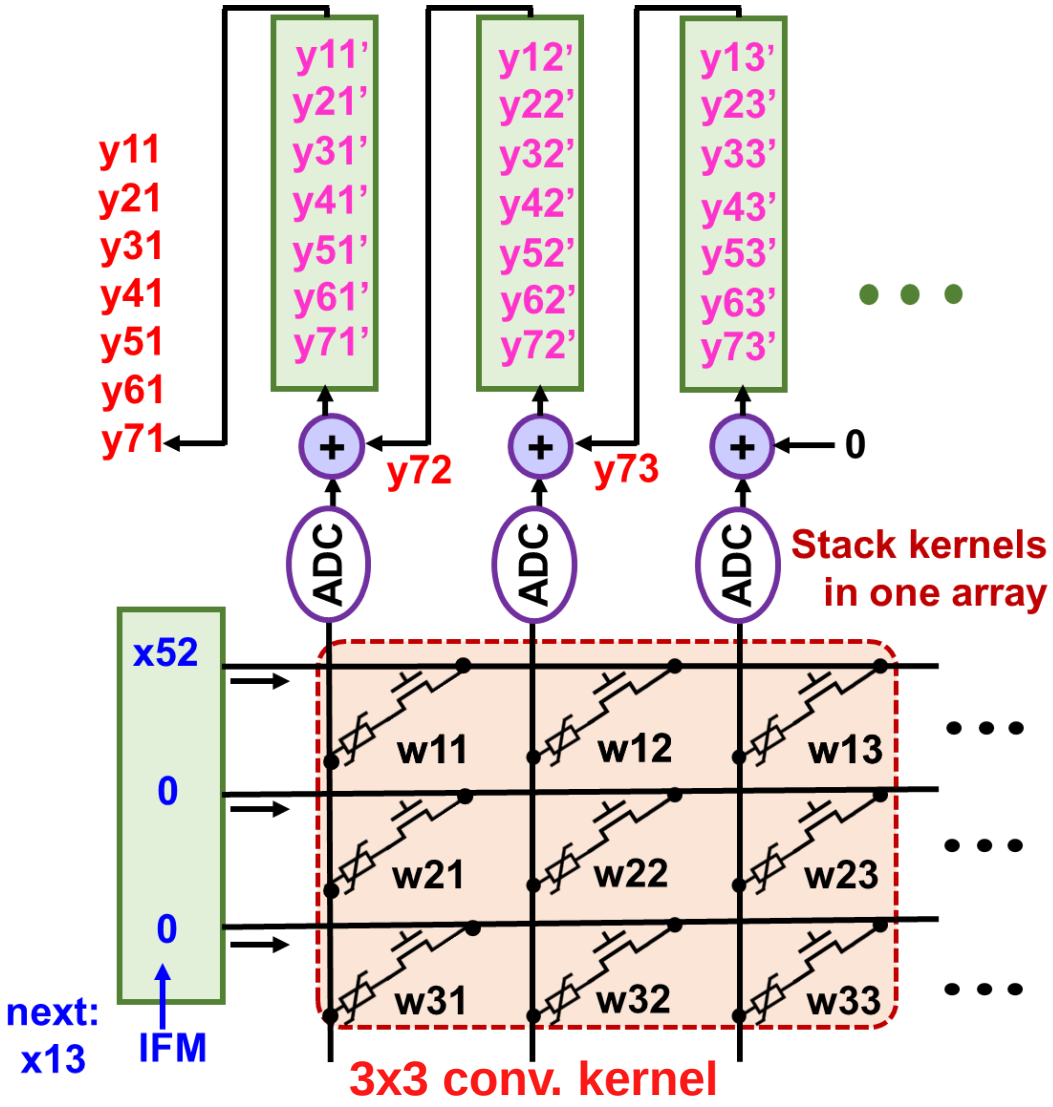
ConvFIFO Dataflow

- To achieve multi-column convolution, insert adders between ADC & output FIFO
- Fetch the **next column of the IFM** and slide through the input FIFO
- **Partial sums** at the top of the output FIFOs (except the left-most) are popped out and added to the left column
- Left-most FIFO: data at the top can be outputted



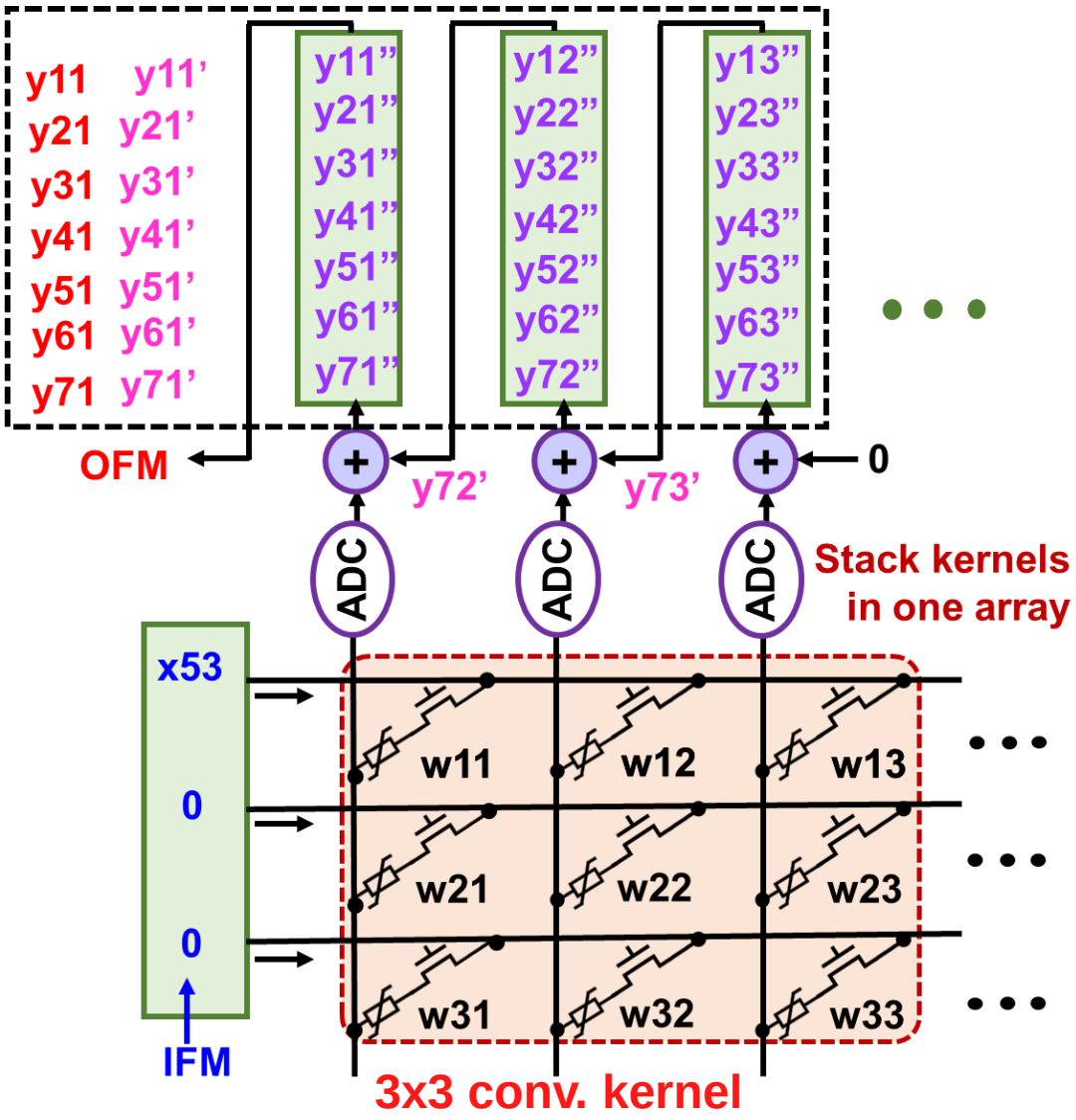
ConvFIFO Dataflow

- Repeat until **the second column of the IFM** has been slided through the input FIFO
- **First column of the OFM** is now fully outputted



ConvFIFO Dataflow

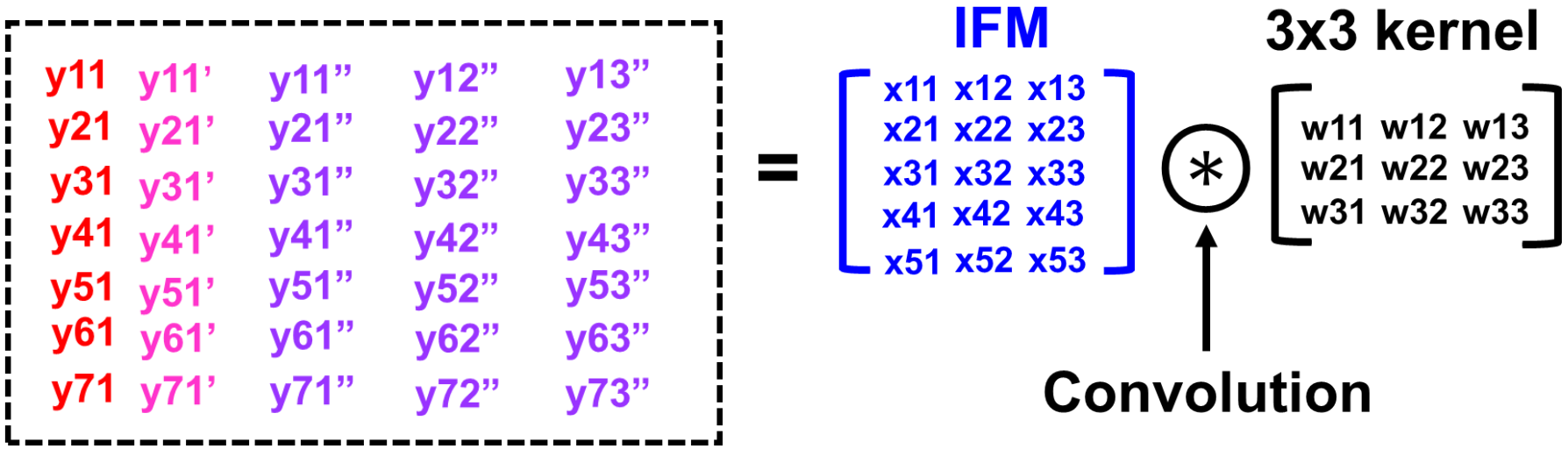
- Repeat until **the third column of the IFM** has been slided through the input FIFO
- **Second column of the OFM** is now fully outputted
- If all columns of IFM have been fetched, then the rest of the OFM is already in the output FIFO



ConvFIFO Dataflow

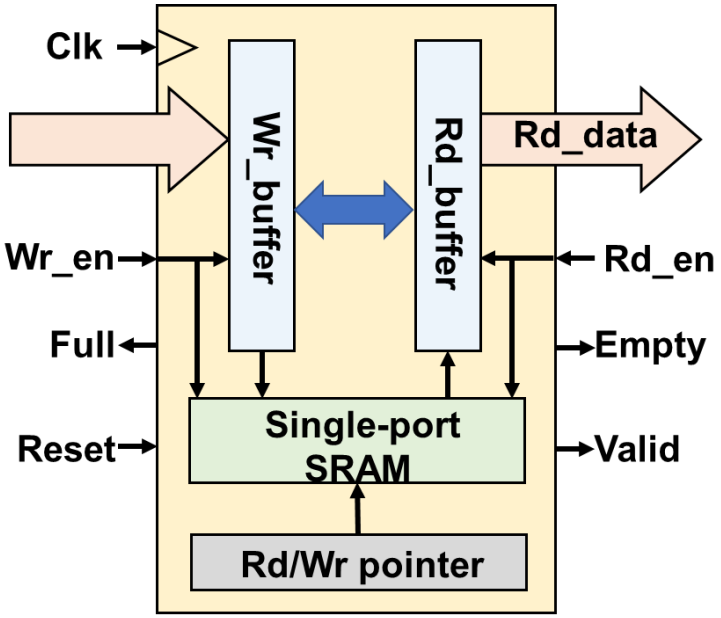
- Repeat until **all pixels of the IFM** are slided through the input FIFO
- Once done, **all pixels of the OFM** can be popped out from the output FIFO sequentially to achieve one full convolution operation

- ***The equation below is proved to be rigorously correct, can be applied to arbitrary-shaped IFMs and kernels***

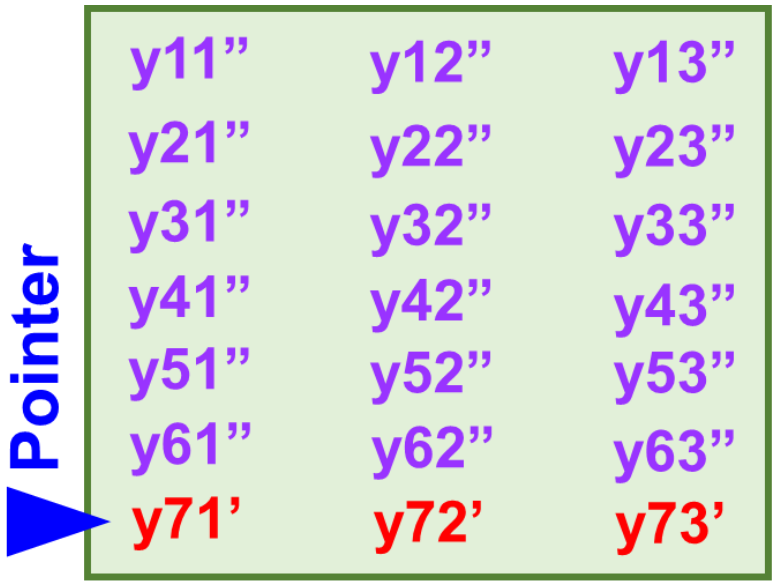


SRAM-based Output FIFO

- **Synchronous SRAM-based FIFO** used as output FIFO
- With the help of pointers, no physical data movement occurred in the output FIFO, improving energy efficiency



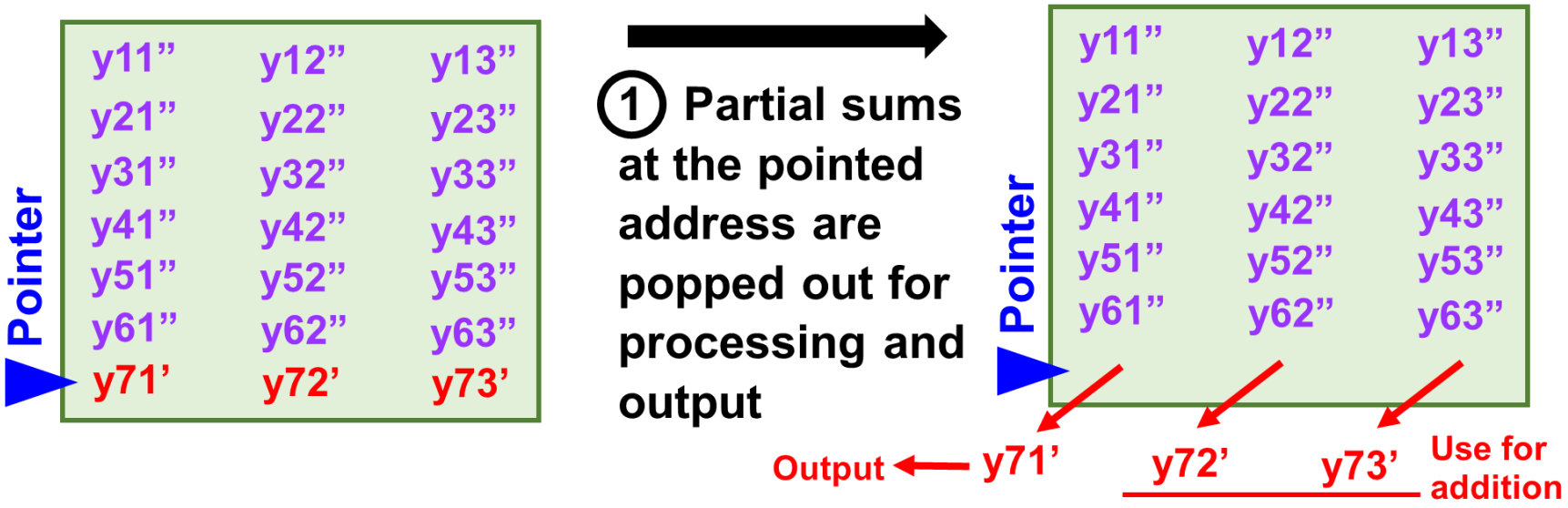
SRAM-based synchronous FIFO



Partial sums stored in the output FIFO

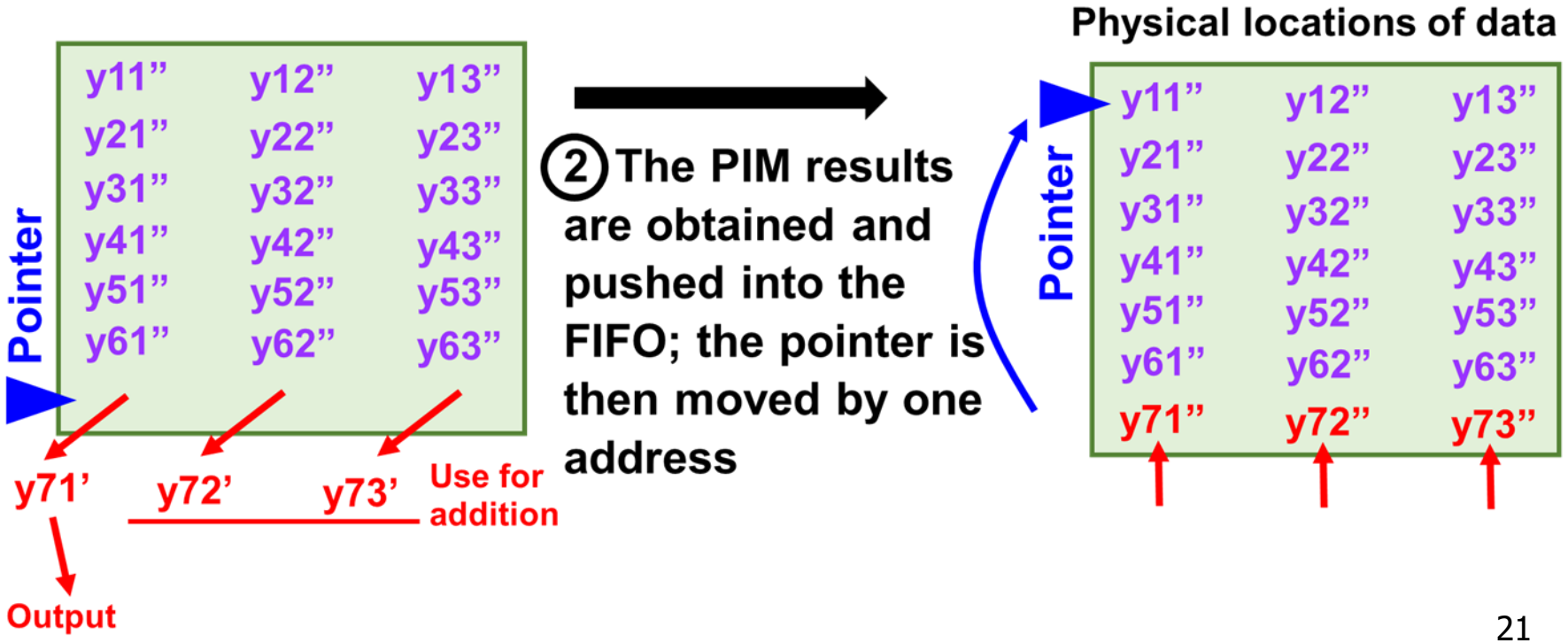
SRAM-based Output FIFO

- During one cycle, partial sums at the pointed address are first popped out for further addition or output



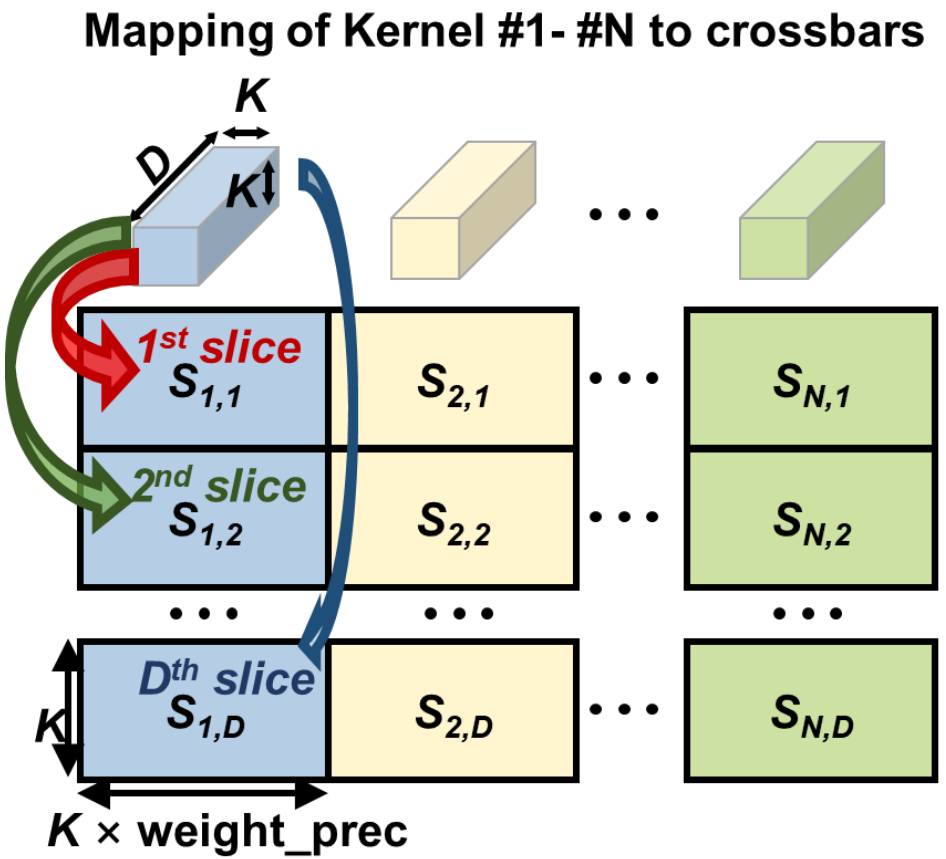
SRAM-based Output FIFO

- The new partial sums containing PIM results are available after one cycle, pushed into FIFO at the pointed address
- The pointer is then moved to the next address
- ***No physical movement of partial sums needed (systolic style)***



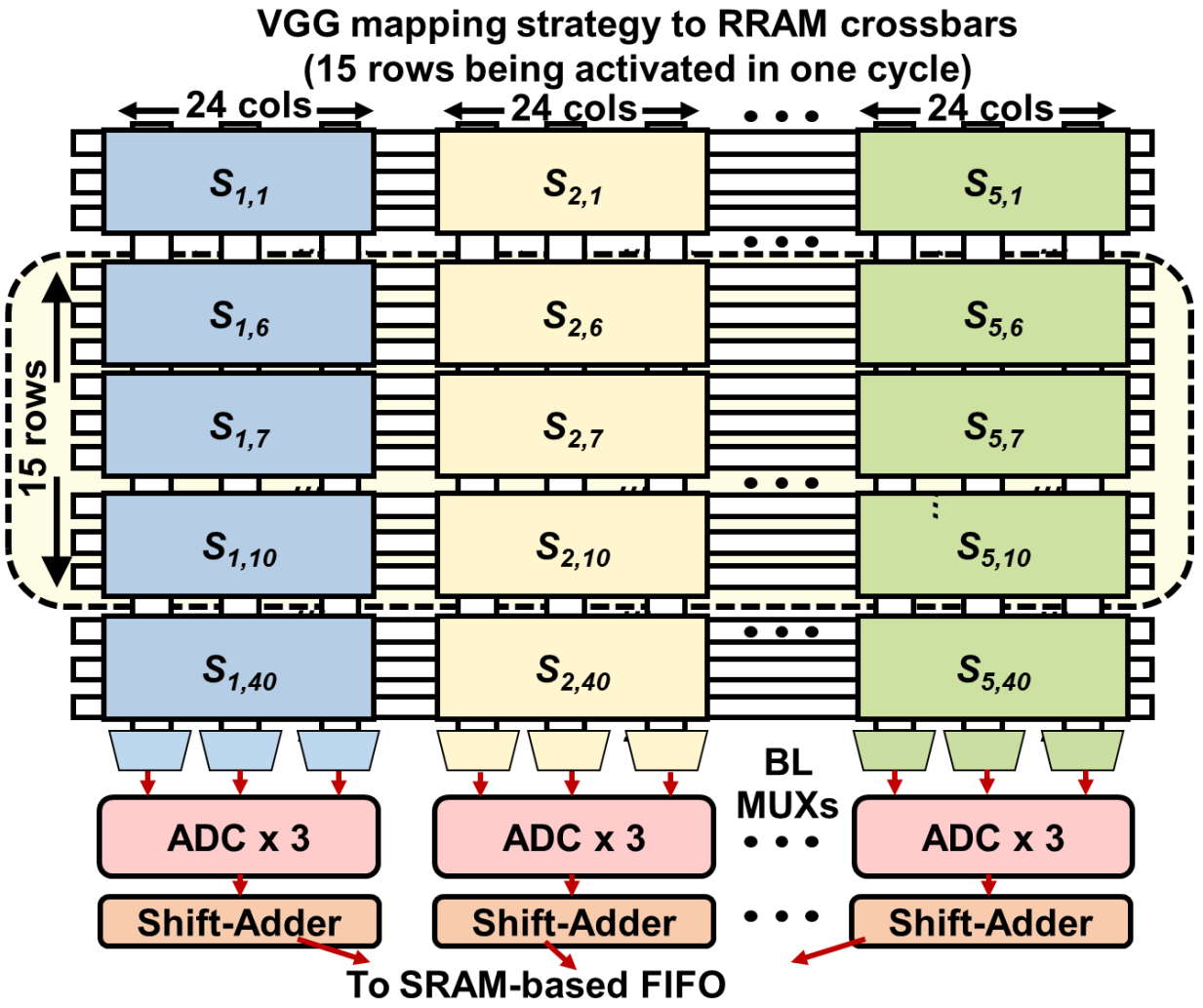
ConvFIFO Mapping Strategy

- In ConvFIFO, different **kernels are mapped along the horizontal direction**
- In order to improve input parallelism, multiple **slices of the same kernel are mapped along the vertical direction** (can be calculated in one cycle)



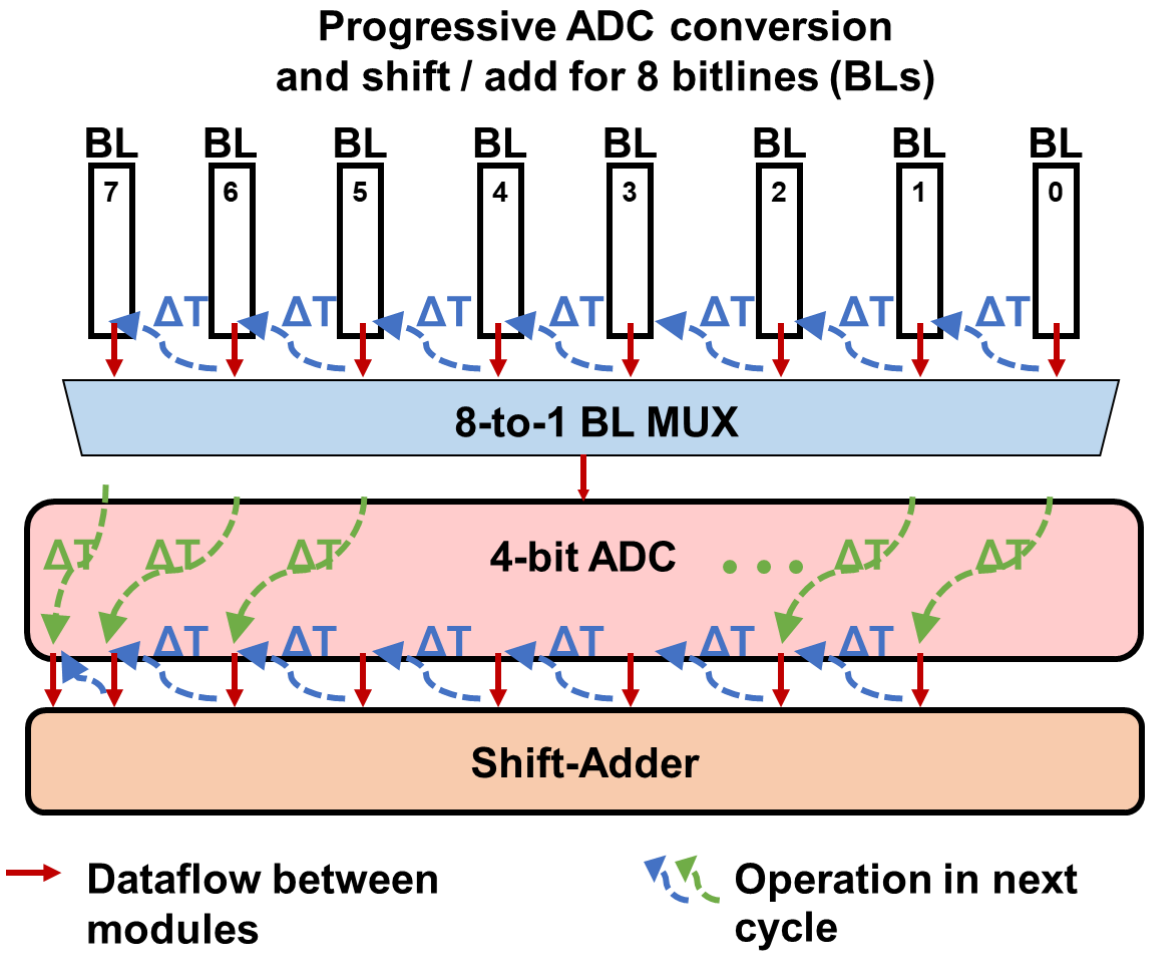
ConvFIFO Mapping Strategy

- Consider 8-bit weights, each 3×3 depth slice is mapped to a 3×24 subarray
- Each 120×120 crossbar array can store 200 depth slices
- Each ADC is shared among 8 bitlines



Progressive ADC Conversion

- ADCs consume significant chip area
- To save area or improve parallelism, **progressive ADC conversion** is proposed, e.g. 8-bit weights are stored across 8 bitlines
- The operation is done from LSB to MSB



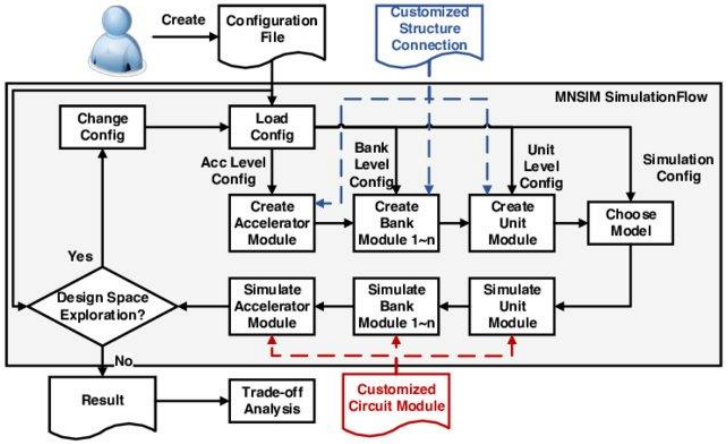
Outline

- Background and Motivations
- The ConvFIFO Architecture
- **Evaluation and Benchmark of ConvFIFO**
- Conclusions

Evaluation Methodology

- Evaluations done by the **MNSim** simulator, ISAAC as reference
- System contains 256 tiles, each containing 64 PEs
- HRS/LRS resistances of 1T1R are set as 200K/10KΩ
- All simulations based on **40nm**

MNSim software flow
(L. Xia et al., TCAD 2017)

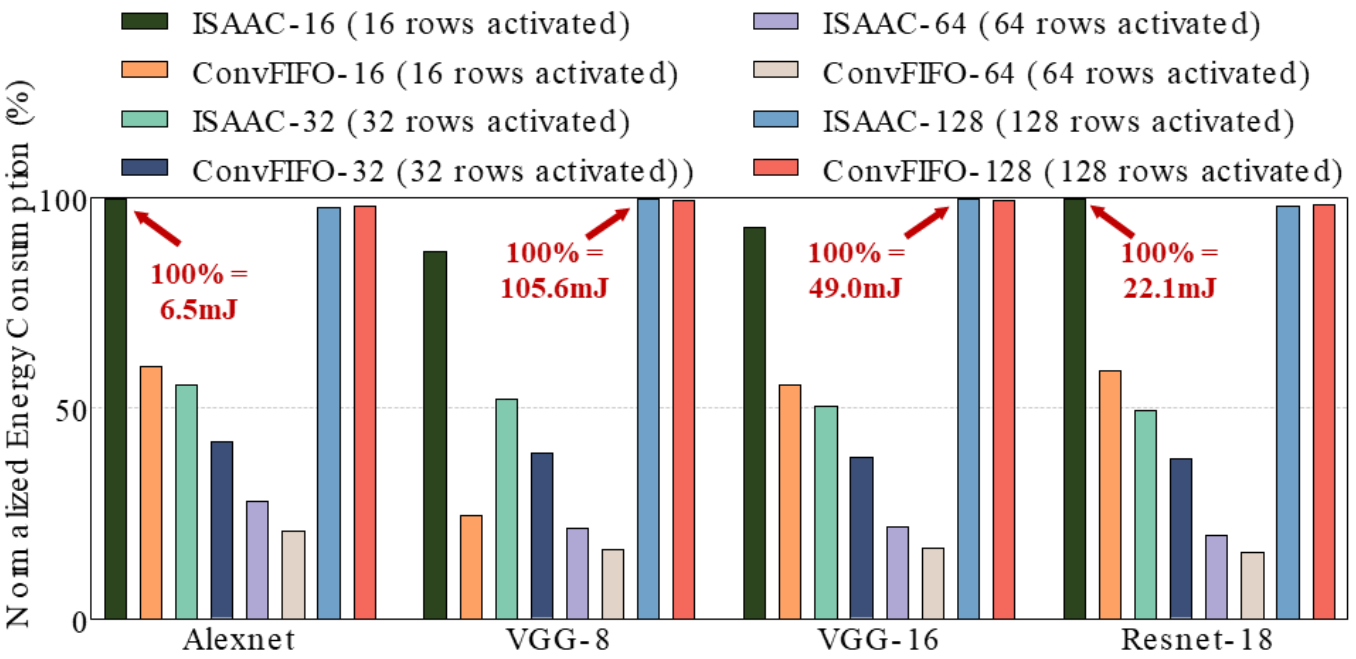


CONVFIFO-16 SETUP (16 ROWS ACTIVATED) WITHIN ONE PE

Component	Params.	Power(mW)	Area(μm^2)
Crossbar	1-bit per Cell 128×128 Size one Crossbar per PE 64 PEs per tile	51.2	146800.64
ADC [34]	4-bit Resolution 16 ADCs per Crossbar 1 Gbit/s for Sampling	535.21	455111.11
DAC	1-bit Resolution 16 DACs per Crossbar 1 Gbit/s for Sampling	7.99	339.97
SRAM FIFO [35]	30×210 Size per Crossbar	40.3	97574.4

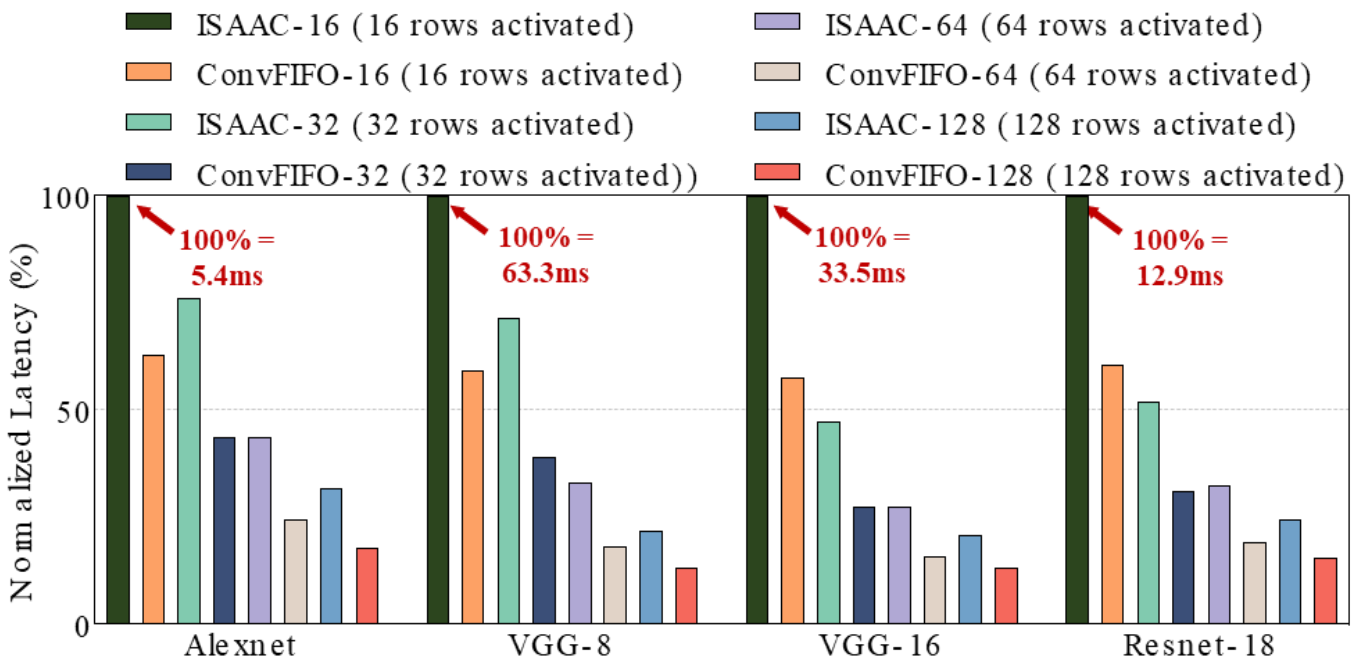
Evaluation Results – Energy Consumption

- ConvFIFO-x: x is the number of simultaneously-activated rows
- ConvFIFO-64 : the lowest total energy consumption
- Trade-off between row parallelism and ADC energy



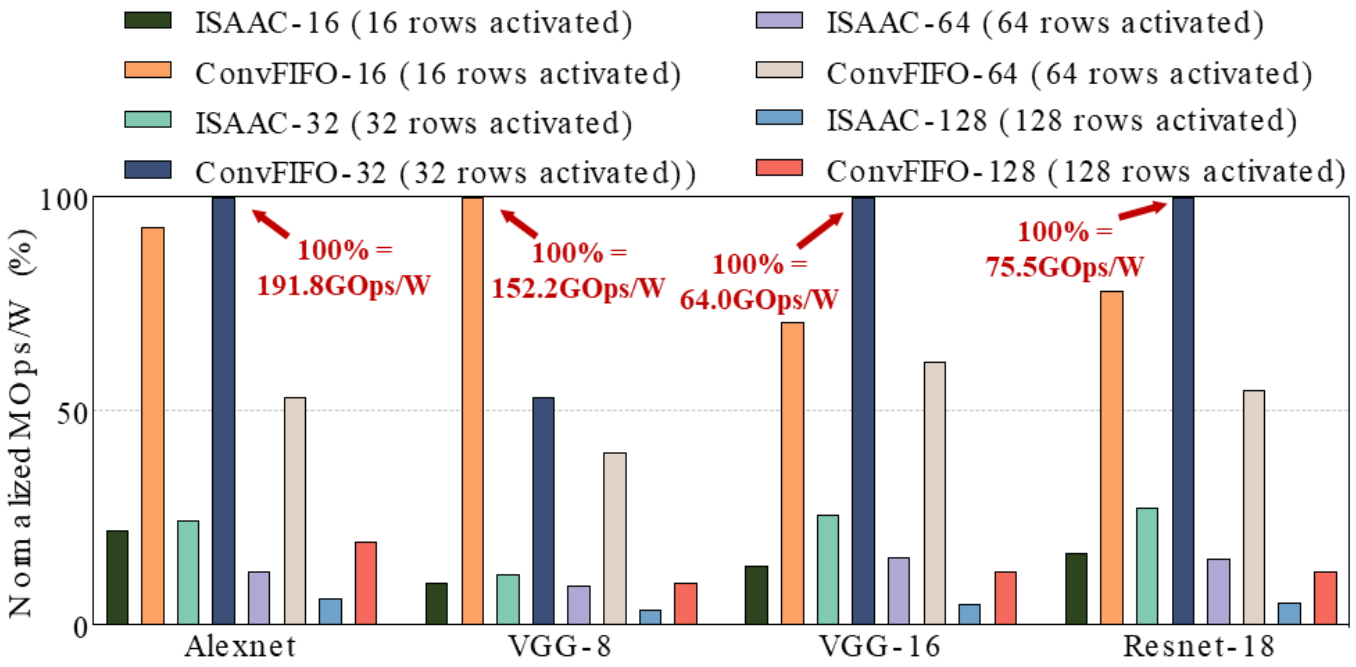
Evaluation Results – Latency

- ConvFIFO-128 : the lowest latency
- Higher row parallelism leads to higher throughput, at the cost of higher energy and lower accuracy



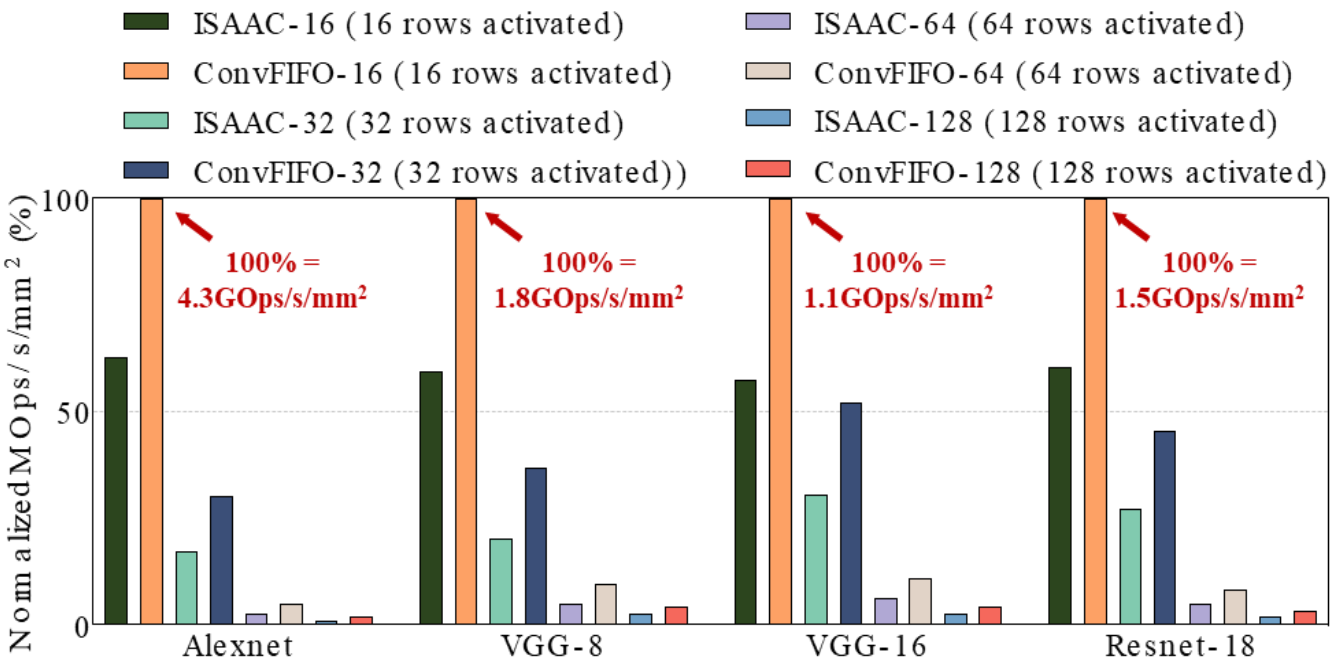
Evaluation Results – Ops/W

- ConvFIFO systematically better than ISAAC in terms of Ops/W
- ConvFIFO-32 : the highest Ops/W
- Achieve good balance between energy efficiency and throughput



Evaluation Results – Ops/s/mm2

- ConvFIFO-16 : the highest Ops/s/mm2
- Only 4-bit ADC needed, which leads to the highest area efficiency; also, lower precision implies better accuracy



Benchmarks vs. ISAAC

- Benchmarked with various ConvNets, ConvFIFO show improvements of total energy (**1.66-3.56×**), latency (**1.69-1.74×**), Ops/W (**4.23-10.17×**) and Ops/s×mm^2 (**1.59-1.74×**) compared to ISAAC

COMPARISON OF ISAAC-x AND CONVFIFO-x IN TERMS OF ENERGY, LATENCY, OPS/W AND OPS/S/MM^2 ON DIFFERENT CONVNETS

ConvNet	Performance	ISAAC-16	ConvFIFO-16	Imp.*	ISAAC-32	ConvFIFO-32	Imp.*	ISAAC-64	ConvFIFO-64	Imp.*	ISAAC-128	ConvFIFO-128	Imp.*
Alexnet	Energy (mJ)	6.52	3.92	1.66×	3.64	2.75	1.32×	1.83	1.38	1.33×	6.39	6.41	1.00×
	Latency (ms)	5.35	3.35	1.59×	4.08	2.33	1.75×	2.34	1.31	1.78×	1.68	0.95	1.77×
	MOps/W	42240.02	178501.37	4.23×	47265.81	191843.32	4.06×	24157.50	102553.76	4.25×	11943.18	37525.45	3.14×
	MOps/s/mm^2	2714.65	4327.39	1.59×	744.88	1305.06	1.75×	118.48	211.40	1.78×	48.29	85.70	1.77×
VGG-8	Energy (mJ)	92.45	25.94	3.56×	55.33	41.73	1.33×	22.84	17.49	1.31×	105.60	105.34	1.00×
	Latency (ms)	63.25	37.45	1.69×	45.26	24.69	1.83×	20.86	11.49	1.82×	13.70	8.32	1.65×
	MOps/W	14974.09	152215.64	10.17×	18207.91	81147.91	4.46×	14225.42	61222.77	4.30×	5549.20	15098.64	2.72×
	MOps/s/mm^2	1044.03	1763.30	1.69×	355.54	651.81	1.83×	92.45	167.83	1.82×	46.86	77.20	1.65×
VGG-16	Energy (mJ)	45.82	27.27	1.68×	24.92	18.93	1.32×	10.78	8.29	1.30×	48.98	48.92	1.00×
	Latency (ms)	33.52	19.25	1.74×	15.81	9.23	1.71×	9.17	5.33	1.72×	6.91	4.37	1.58×
	MOps/W	8912.04	45392.72	5.09×	16578.35	64020.19	3.86×	10209.13	39307.10	3.85×	3245.55	8106.94	2.50×
	MOps/s/mm^2	628.06	1093.53	1.74×	332.72	569.86	1.71×	70.12	120.68	1.72×	30.45	48.09	1.58×
Resnet-18	Energy (mJ)	22.12	13.12	1.69×	10.98	8.47	1.30×	4.43	3.55	1.25×	21.69	21.84	0.99×
	Latency (ms)	12.91	7.81	1.65×	6.70	4.00	1.67×	4.17	2.46	1.69×	3.15	2.01	1.57×
	MOps/W	12640.55	58178.01	4.60×	20501.03	74488.77	3.63×	11462.89	40929.18	3.57×	3790.82	9314.49	2.46×
	MOps/s/mm^2	899.77	1486.81	1.65×	403.72	675.85	1.67×	73.52	124.43	1.69×	30.43	47.86	1.57×

$\star$: Imp. denotes the improvement of ConvFIFO compared to ISAAC.

Improvements

Outline

- Background and Motivations
- The ConvFIFO Architecture
- Evaluation and Benchmark of ConvFIFO
- **Conclusions**

Conclusions

- **ConvFIFO: a crossbar-based PIM architecture for ConvNets featuring a unique first-in-first-out dataflow.**
- With the help of input/output FIFOs, ConvFIFO maximizes the reuse rates of inputs and partial sums.
- SRAM-based FIFO is used to achieve a systolic architecture without need to constantly move weights and partial sums.
- ConvFIFO bypasses some limitations due to NVM non-idealities (e.g. endurance, multi-level variations and parasitics).
- **Compared to ISAAC, ConvFIFO show improvements in terms of total energy (1.66-3.56×), latency (1.69-1.74×), Ops/W (4.23-10.17×) and Ops/s×mm^2 (1.59-1.74×).**

ASIA SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE

ConvFIFO: A Crossbar Memory PIM Architecture for ConvNets Featuring First-In-First-Out Dataflow

Thank you!

Q&A