IMEC: A Memory-Efficient Convolution Algorithm For Quantised Neural Network Accelerators

Eashan Wadhwa, Shashwat Khandelwal, Shreejith Shanker

Trinity College Dublin

Presentation for ASAP'22
14th July 2022

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

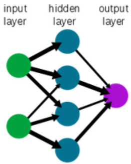
Overview

Neural Networks and their parameters

NNs are modelled after the human brain and are designed to recognize patterns by assigning each feature weights

How does it work?:

- Training / Learning
- Inference



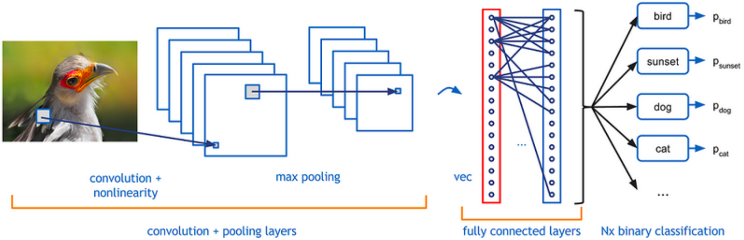
Factors:

- Accuracy
- Throughput
- Latency
- Power Requirements

Source: *[1]*

Convolution Neural Networks

Since NNs don't scale up well with images, CNN architectures constrain the 3D model into differentialable functions to make the model simpler



Source: *[2]*

Convolution Neural Networks

FPGAs provide flexibility, high throughput, fine- grain parallelism, and energy efficiency

- Diminishing effects from technology scaling
- Research now focuses on specialised acclerators
- FPGAs provide efficiency of general-purpose acclerators

Source: *[3]*

Convolution Neural Networks

Source: *[4]*

Why CNNs are **not** hardware-friendly?

- AlexNet has 650M parameters occupying 240MB (3:1 ratio)
- Inexpensive FPGAs have 1MB On-Chip-Memory

type	m	r	n	p	q	Par.	Mult.
conv	20	8	64	1	3	10.2K	27.7M
conv	10	4	64	1	1	164K	95.7M
lin	-	-	32	-	-	1.20M	1.20M
dnn	-	-	128	-	-	4.1K	4.1K
softmax	-	-	n_{labels}	-	-	1.54K	1.54K
Total	-	-	-	-	-	1.37M	125M

Source: *[4]*

Prerequisite

A few assumptions to make discussions simpler

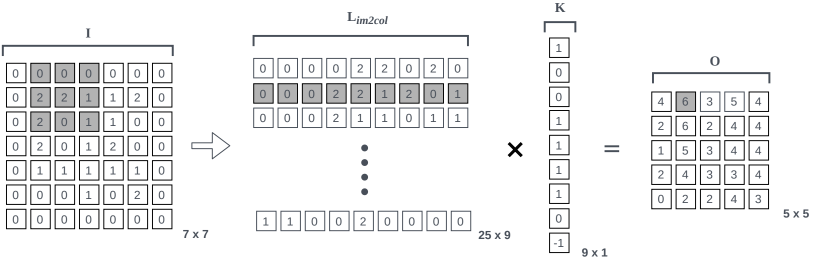
- Stride : `1`
- Precision : `4`
- Number of channels : `1`

How Convolution works in the training phase

$$
\underbrace{\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 2 & 1 & 1 & 2 & 0 \\
0 & 2 & 0 & 1 & 1 & 0 & 0 \\
0 & 2 & 0 & 1 & 2 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}}_{\mathbf{I}\ (7 \times 7)}
\times
\underbrace{\begin{bmatrix}
1 & 0 & 0 \\
1 & 1 & 1 \\
1 & 0 & -1
\end{bmatrix}}_{\mathbf{K}\ (3 \times 3)}
=
\underbrace{\begin{bmatrix}
4 & 6 & 3 & 5 & 4 \\
2 & 6 & 2 & 4 & 4 \\
1 & 5 & 3 & 4 & 4 \\
2 & 4 & 3 & 3 & 4 \\
0 & 2 & 2 & 4 & 3
\end{bmatrix}}_{\mathbf{O}\ (5 \times 5)}
$$

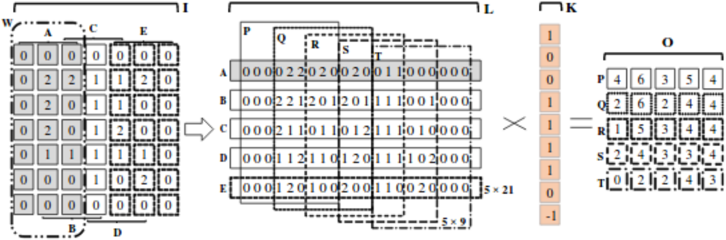
Standard convolution algorithm

How Convolution works in the inference phase



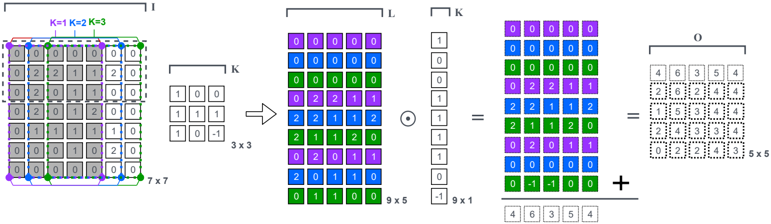
`im2col` algorithm

Memory Efficient Convolution



Source [5]

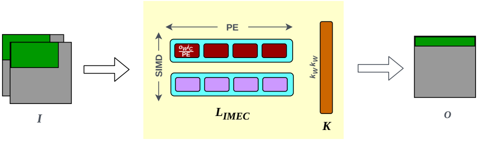
Inverse Memory Efficient Convolution



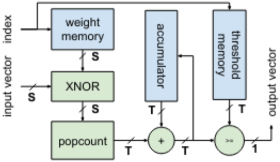
`IMEC` algorithm

Verification

- Implementation in Vivado HLS
- The modifier headers are then used as part of the FINN library
- by changing the input and output dimenstions of the dataflow convolver we implemented in this in the larger BNN-PYNQ framework



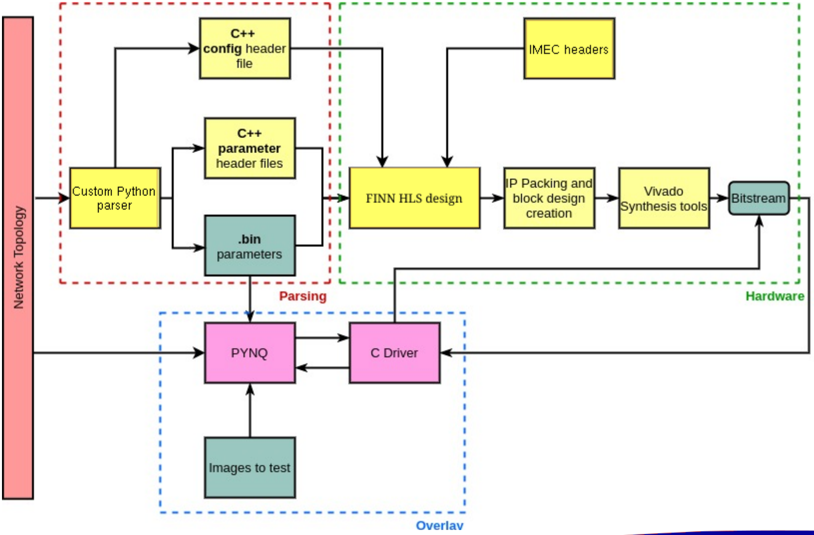
Sliding window with matrix-accumulate in IMEC



Matrix-accumulate datapath

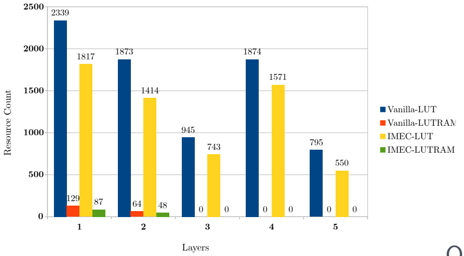
Source: *[6]*

BNN-PYNQ (IMEC) Framework

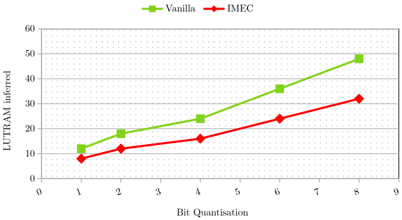


Results

Accelerator	Framework	Model	Platform	Frequency (MHz)	Resource consumption				Resource saving v/s corr. FINN				Power (Watt)
					LUTs	LUTRAMs	FFs	DSPs	LUTs	LUTRAMs	FFs	DSPs	
LUTNET [9]	Tiled-LUTNET	CNV	Kintex XCKU115	200	106,776	3,786	216,513	184	-	-	-	-	6
FINN (1-bit) [11]	BNN-PYNQ	CNV	Zynq XC7Z020	200	29,635	2,438	42,053	24	1.0	1.0	1.0	1.0	1.793
This Work (1-bit)	BNN-PYNQ	CNV	Zynq XC7Z020	200	23,744	2,322	38,110	24	0.8	0.95	0.91	1.0	1.764
FINN (2-bit) [11]	BNN-PYNQ	CNV	Zynq XC7Z020	200	40,022	7,598	51,321	32	1.0	1.0	1.0	1.0	1.863
This Work (2-bit)	BNN-PYNQ	CNV	Zynq XC7Z020	200	35,001	7,273	43,738	32	0.87	0.96	0.85	1.0	1.828



Resource level comparasion for convolution layers



Quantisation w.r.t. LUTRAMs (single layer only!)

Future work

- There could be even more massive gains compared to the vanilla `im2col` implementations, given we find a compute-intensive application/framework for it (currently limited to only BNN-PYNQ)
- Implement a simpler framework to test such algorithms
- Implement the IMEC algorithm in GPUs to see performance

Always open to any other suggestions / questions (email me `wadhwae@ieee.org` or any of the other authors)!

Bibliography

[1] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. "CIFAR-10 (Canadian Institute for Advanced Research)". In: (). URL: http://www.cs.toronto.edu/~kriz/cifar.html (cit. on p. 3).

[2] Adit Deshpande. *Convolution Neural Network*. 2015. URL: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/ (visited on 2020) (cit. on p. 4).

[3] URL: https://www.missinglinkelectronics.com/www/www/index.php?option=com_content&view=category&layout=blog&id=141&Itemid=310 (visited on 2022) (cit. on p. 5).

[4] Raphael Tang et al. "An Experimental Analysis of the Power Consumption of Convolutional Neural Networks for Keyword Spotting". In: *CoRR* abs/1711.00333 (2017). arXiv: 1711.00333. URL: http://arxiv.org/abs/1711.00333 (cit. on p. 6).

[5] Minsik Cho and Daniel Brand. "MEC: Memory-efficient Convolution for Deep Neural Network". In: *CoRR* abs/1706.06873 (2017). arXiv: 1706.06873. URL: http://arxiv.org/abs/1706.06873 (cit. on p. 10).

[6] Yaman Umuroglu et al. "Finn: A framework for fast, scalable binarized neural network inference". In: *Proceedings of the 2017 ACM/SIGDA international symposium on field-programmable gate arrays*. 2017, pp. 65–74 (cit. on p. 12).