[Public]

A High-Performance Pixel-Level Fully Pipelined Hardware Accelerator for CNNs

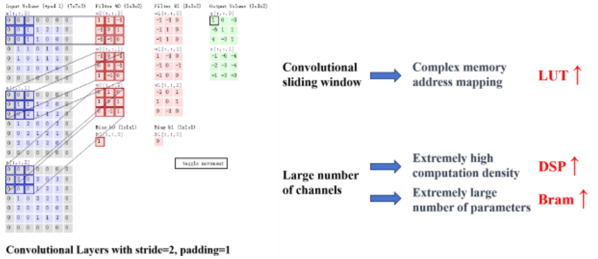
Li, Zhan，Xingyu Shi，Zhihan Zhang
Wuhan University, Hubei Province

OpenHW2023

On board test by AMD ZCU104

INTRODUCTION

The traditional multi-CE architecture consists of multiple **customized CEs** (each containing a PE array). **PE array** shows the schematic of the PE array section in a CE. Each PE array is flanked by a PE in/out Buffer. In the CE architecture, a complete computation cycle ***T*** includes the following delay components: Buffer Refresh Time (T_{br}) and PE Calculation Time (T_{pe}). The T_{pe} further includes PE Pipeline Time (T_{pip}) and Effective Time (T_e). Using T_e /T for accelerator efficiency. Every computation unit, the CE need to refresh its buffer and break the pipeline, which is **inefficient**. In pipeline architecture, T_e/T equals to **1**.



Convolutional Layers with stride=2, padding=1

Convolutional sliding window → Complex memory address mapping — LUT ↑

Large number of channels → Extremely high computation density — DSP ↑

→ Extremely large number of parameters — Bram ↑

Deploy CNN on FPGA

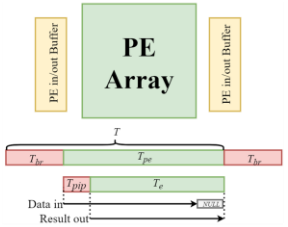


Fig. 1 PE Array

Buffer Refresh Time (T_{br})

PE Pipeline Time (T_{pip})

Effective Time (T_e)

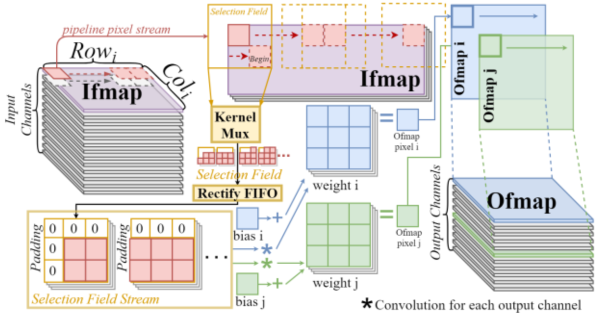
T_e / T → Accelerator efficiency

$T = T_{br} + T_{pip} + T_e$ CE

$T = T_e$ PIPELINE

Traditional CE Architecture

In pipeline architecture, caching (***Row***$_i$**+2**)×***In_C*** Ifmap pixels can solve the first pixel of Ofmap , and every new input pixel can determine a new selection field and compute the value of the next pixel. In CNNs, each receptive field undergoes a multiplication-addition operation, **resulting in a batch of output**.

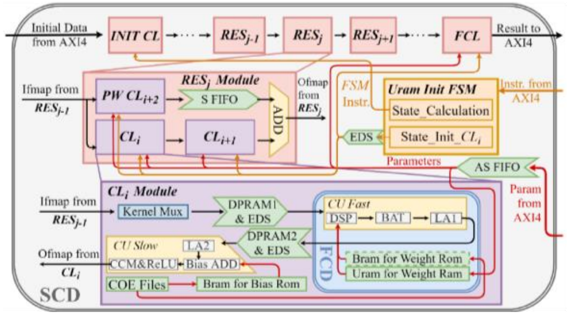


*Convolution for each output channel

Using a **slow clock domain (SCD)** to drive designs based on LUTs and FFs.

Using a **fast clock domain (FCD)** to drive designs based on DSPs and Brams/Urams.

Enhancing the efficiency of FPGA hard cores and further enhancing system performance.



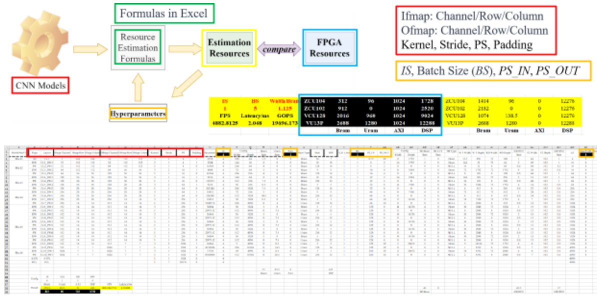
Initializing Uram through AXI4 bus input parameters before starting calculations.

Performing calculations in the FCD after determining the selection domain.

After calculations in the FCD, transitioning back to the SCD for post-processing.

CREATIVE DESING

RESULT



Hardware Design Toolchain

	[1]	[2]	[3]	[4]	[5]	**Our works**
Network	TNT-S	MobileNetV1	DSC	MobileNetV2	VGG16	MobileNetV1
Platform	VC707	XQRKU060	VC709	Arria 10 SoC	VX980T	ZCU104
Frequency	200MHz	54MHz	200MHz	133MHz	150MHz	100Mhz/400Mhz
Precision	8bit fixed	4-10bit fixed	4bit fixed	16bit fixed	8/16bit fixed	8bit fixed
LUT Util	156120	166924	107325	66127	335000	90762
FF Util	77223	58027	74430	251680		100556
DSP Util	2048	2338	1291	1687	3395	400
Bram Util	1024	642	381	2131	1492	Bram:269 Uram:92
Throughput	728.3GOPS	10.60 GOPS	413.2 GOPS	170.6 GOPS	1000 GOPS	571.82GOPS
Power (W)	12.49	3.22	6.35		14.36	5.104
Speed (fps)	67.6	9.30	2284	266.2		498.25
Latency	14.79ms	107.14ms	0.44ms	3.76ms		2.05ms
GOPS/W	58.31	3.29	65.07		69.64	112.03
GOPS/DSP	0.36	0.0045	0.32	0.10	0.29	1.42
GOPS/kLUT	4.67	0.064	3.03	0.68	2.99	6.30

High performance

High efficiency

References:
[1],[3],[4] : *IEEE Trans. Circuits Syst. II*
[2] : *IEEE Access*
[5] : *IEEE Trans. Neural Netw. Learn. Syst.*

Experiment Results