JULY 9–13, 2023

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

A Frequency Optimized Linear Scalable Architecture for CNN Implementation on FPGAs

Shubhayu Das* (Shubhayu.Das@iiitb.ac.in)
Nanditha Rao* (Nanditha.Rao@iiitb.ac.in)
Sharad Sinha# (sharad@iitgoa.ac.in)

*International Institute of Information Technology Bangalore, Karnataka, India
#Indian Institute of Technology, Goa, India

July 9-13 2023
Moscone West Center
San Francisco, CA, USA

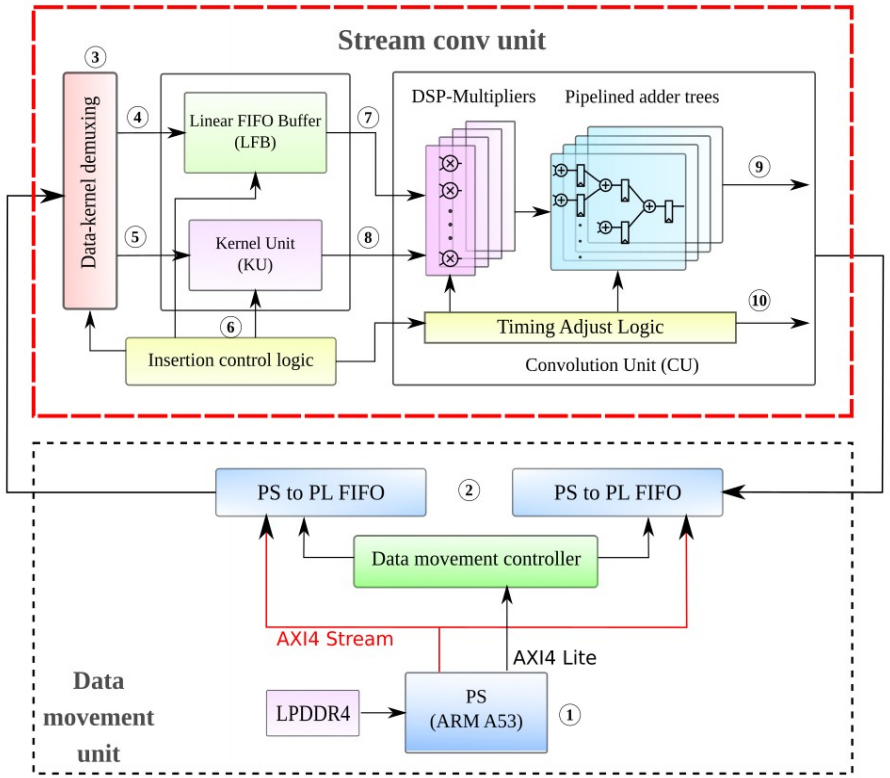
Overview

- Motivation
- Key idea- Pipelined DSP adder trees (PDATs)
- Architecture diagram on an AMD-Xilinx Zynq UltraScale+ device (Avnet Ultra96 v1)
- Explanation of the Stream-Conv Unit - the main computation unit
- Explanation of the existing data movement
- Results
- Future work

Motivation

- ➔ Context: FPGA-based hardware accelerators are suitable for CNN implementation due to the availability of **inherent parallel computing resources** in the FPGA fabric
- ➔ Our goal is to build an inexpensive, compact and light (physical) weight CNN accelerator
- ➔ We specifically ask the following questions:
  - ◆ How can we improve the **throughput** of an **image-kernel convolution** on an **FPGA**?
  - ◆ What is the most **efficient hardware structure on the FPGA** to map the convolution operation to?
- ➔ We map the **convolution** operation onto **pipelined DSP-adder tree (PDAT)** structures to improve the performance
- ➔ We explore the compromise between overall performance vs the size/cost of the accelerator
- ➔ Can be deployed is real-life applications like image segmentation/labelling applications on mobile robots

Key idea and architecture



1. Application Processor (PS)
2. CDC Logic across clock boundaries
3. Separate data/kernel
4. data bus
5. kernel bus
6. Insertion control lines
7. data to CU
8. kernel(s) to CU
9. convolution output
10. valid result flag

AXI4 Stream is used to tranfer data from the DDR memory connected to the PS, via DMA.

AXI4-Lite is used for configuring the movement of data and other timing related information

Key ideas:

Design an efficient Stream-convolution unit:

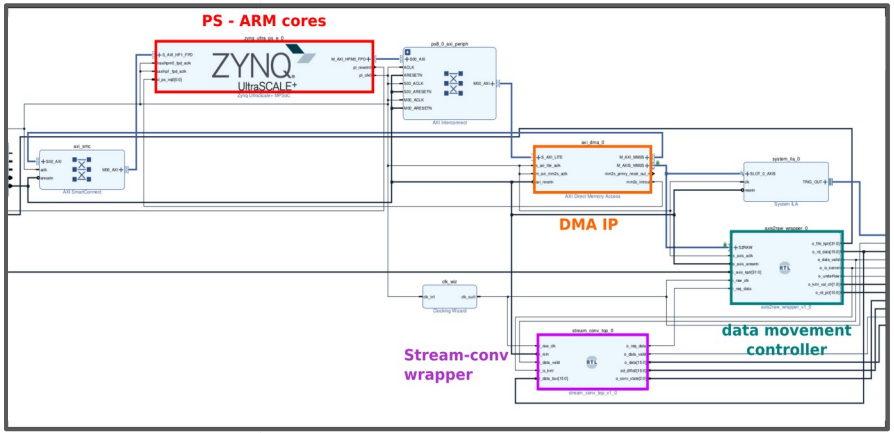
The RTL for the Stream-conv unit is designed such that it maps to specific Xilinx FPGA blocks to improve the performance

For example:

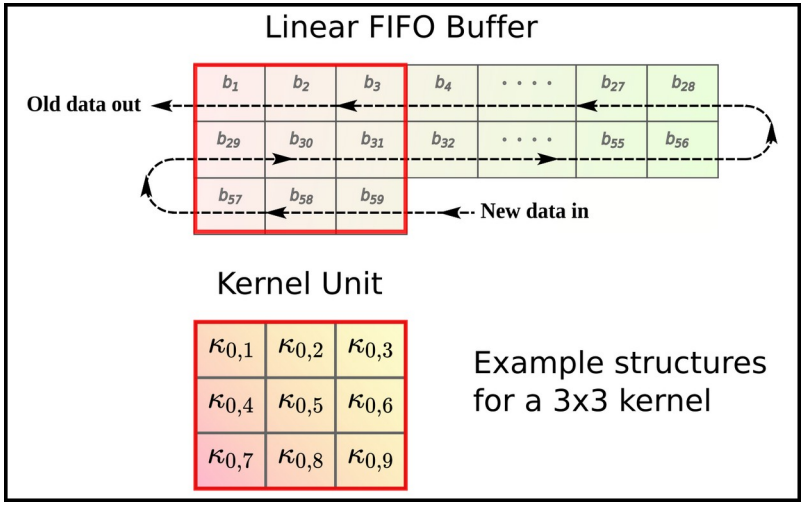
1. Map the **convolution** operation onto **pipelined DSP-adder tree (PDAT)** structures

2. Map the LFB to shift registers (SRLC32E)

Explanation of the Data movement



We utilize the high performance AXI ports (AXI4-HP) ports along with Xilinx's DMA IP to transmit data between the PS [ARM core(s)] and the PL (FPGA fabric)



The linear movement of data inside the stream-conv unit (kernel static, after initial load). The elements highlighted in red are multiplied with the kernel (element-wise), at every timestep.

FPGA Setup, layout and schematics

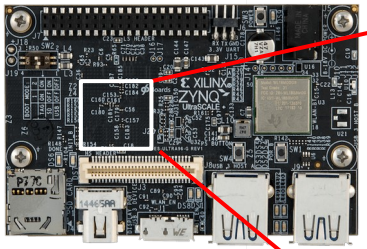
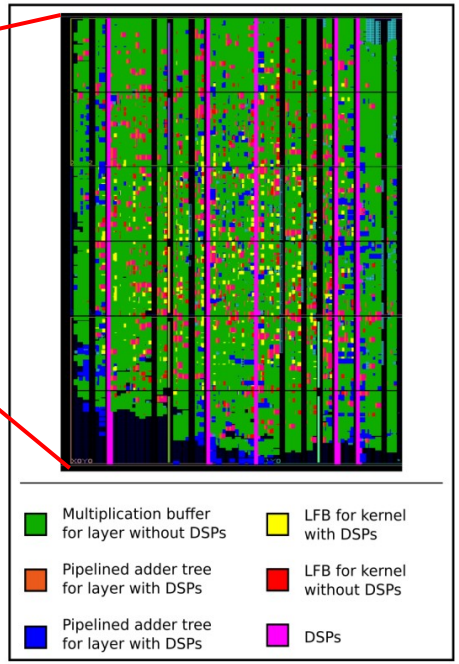
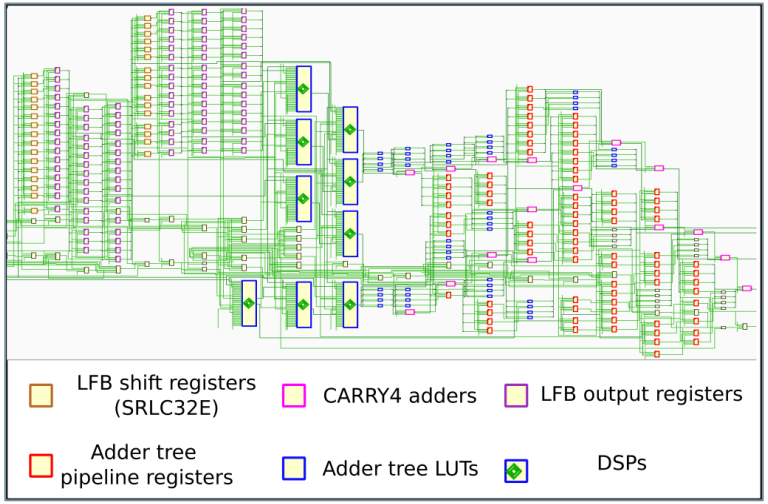


Image c
redit

Board Name	Ultra96 V1
SoC	Xilinx Zynq UltraScale+ MPSoC ZU3EG A484
RAM	512Mb x 32 LPDDR4





Overall structure of the stream-conv unit after mapping onto the Xilinx UltraScale+ FPGA

Results

Setup:
A: Zedboard (Zynq-7000)
B: Ultra96 v1 (UltraScale+ MPSoC)

Design 1: Cascaded DSPs with a SISO Linear FIFO buffer (LFB) – *lower clock freq baseline*

Design 2: Pipelined DSPs with SISO LFB

Design 3: Pipelined DSPs and pipelined adder trees with a PISO LFB – *improved performance, but inconsistent*

Design 4: Pipelined DSP with pipelined adder-tree with SISO LFB results in the best performance – *slightly higher latency, but very consistent*

Performance of the stream-conv unit on 2 different FPGA architectures

Setup						Performance				Resource utilization			
FPGA setup	Design version	Model name	Input size	Kernel size	Data width	Observed number of clock cycles	Clock period (ns)	Max clock frequency (MHz)	Latency (ms)	LUT	FF	DSP	LUTRAM
Setup A	Design 1	LeNet5	28x28	3x3	16	788	10.5	95	8.274	214	110	10	140
Setup A	Design 2	LeNet5	28x28	3x3	16	788	6	167	4.728	318	95	9	140
Setup A	Design 3	LeNet5	28x28	3x3	16	736	4.35	230	3.202	1381	1198	9	0
Setup A	Design 4	LeNet5	28x28	3x3	16	792	4.35	230	3.445	174	242	9	32
Setup A	Design 1	Alexnet	224x224	9x9	18	50180	19	53	953.42	1357	684	85	1178
Setup A	Design 2	Alexnet	224x224	9x9	18	50180	11.9	84	597.142	2353	2737	81	1008
Setup A	Design 3	Alexnet	224x224	9x9	18	50064	8.0	125	400.512	40108	33994	81	0
Setup A	Design 4	Alexnet	224x224	9x9	18	50186	4.35	230	218.309	1897	2320	81	1008
Setup A	Design 1	VGG16	224x224	3x3	18	50180	10.5	95	526.89	433	137	10	360
Setup A	Design 2	VGG16	224x224	3x3	18	50180	6.35	157	318.643	553	137	9	360
Setup A	Design 3	VGG16	224x224	3x3	18	49761	4.6	217	228.901	10932	9115	9	0
Setup A	Design 4	VGG16	224x224	3x3	18	50183	4.35	230	218.296	402	345	9	252
Setup B	Design 4	LeNet5	28x28	3x3	16	792	2.00	500	1.584	178	351	9	32
Setup B	Design 4	AlexNet	224x224	9x9	16	50186	2.00	500	100.372	1852	2468	81	928
Setup B	Design 4	VGG16	224x224	3x3	16	50183	**2.00**	500	100.366	370	556	9	224

Comparison with related work

	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	**Our design**
FPGA	Virtex 7	Virtex 7	XC7Z020	Arria10 GX1150	Virtex VCU440	Zynq-7000	Zynq XC7Z045	Arria10 GX1150	MPSoC ZU3EG
Process node	28nm	28nm	28nm	20nm	16nm	28nm	28nm	20nm	16nm
Frequency (MHz)	200	100	200	303	200	-	150	200	500, 300
LUTs	229K (52.9%)	186K(61%)	36384 (68.4%)	246K(58%)	175K (40%)	70K (75%)	303K	138K(32%)	61861 (87.67%)
FFs	253K (29.3%)	205K (34%)	28326 (26.6%)	30K	681K (%)	30%	607K	NA	31932 (22.63%)
BRAMs	1188 (40.4%)	1024 (50%)	280 (12.5%)	2487 (92%)	1232 (42%)	21 (15%)		2232 (82%)	1 (0.4%)
DSPs	1307 (36.3%)	2240 (80%)	–	1476 (97%)	1476 (97%)	–	–	1518 (100%)	360 (100%)
Data type	float32	float32	-	float 16	float 16	float	16 bit fixed	16 bit fixed	16 bit fixed
Throughput (GOPs)	45.8	61.62	–	1382	821	0.76 iter/s	123.76	715.9	544

This is not a 1:1 comparison. We have not implemented an entire model yet; our FPGA is smaller than previous works, but has a newer fabric architecture,

[1] DiCecco, FPT'16; [2] Zhang, FPGA'15; [3] Wang, TCAD, 2017; [4] Aydonat, FPGA'17;
[5] Shen, FPGA'18; [6] Sanchez, HPEC' 18; [7] Jiantao Qiuet, FPGA'16; [8] Y.Ma, TVLSI'18

Summary and Future Work

- Summary:
  - We explored both SISO and PISO formats of a linear FIFO buffer and found that the SISO structure is better than the PISO when combined with a pipelined DSP-adder tree for Convolution.
  - We mapped the convolution operation onto pipelined DSP-adder tree structures on the FPGA, which resulted in a 2.4× improvement in frequency compared to the baseline
  - We replicated the PDATs to entirely populate the FPGA by implementing tiled PDATs. We achieved a competitive throughput of 544 GOP/s on the resource-constrained FPGA Ultra96 V1
- Future work:
  - Buffer the data in the programmable logic of the FPGA to retain and loop-back data without the need for the PL-to-PS communication
  - Improve on the configuration logic(over AXI4-Lite) to support alternative movement of data
  - Explore alternate memory connections to the PL, use an OS (like Linux) to schedule data movement