Convolution Processing Unit Featuring Adaptive Precision using Dynamic Reconfiguration

WF-IoT 2021 Paper

David Cain, Omar Eldash, Kasem Khalil, Magdy Bayoumi

UNIVERSITY of LOUISIANA LAFAYETTE

Presented by David Cain

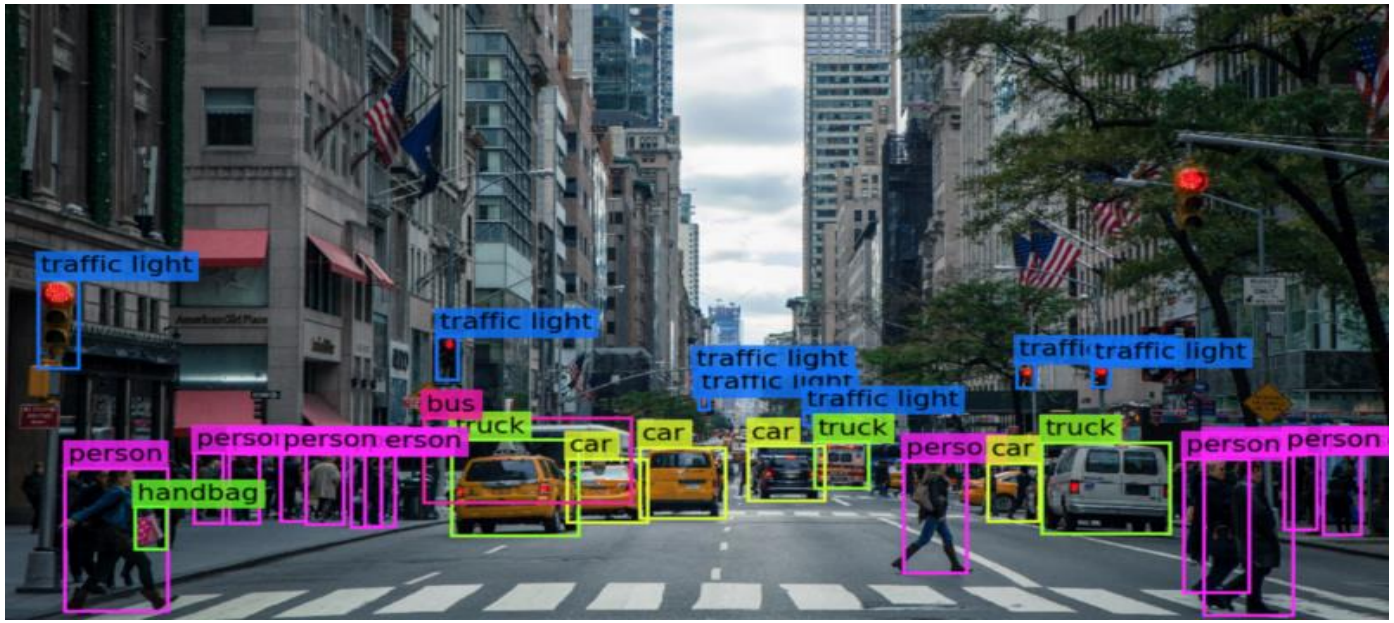
- **Introduction**
  - Computer Vision
  - Traditional IoT CNN Models
  - Reconfigurable Hardware
  - Dynamic Partial Reconfiguration
- Related Work
- Proposed Convolution Processing Element
- Hardware Implementation & Comparisons
- Conclusions
- Q&A

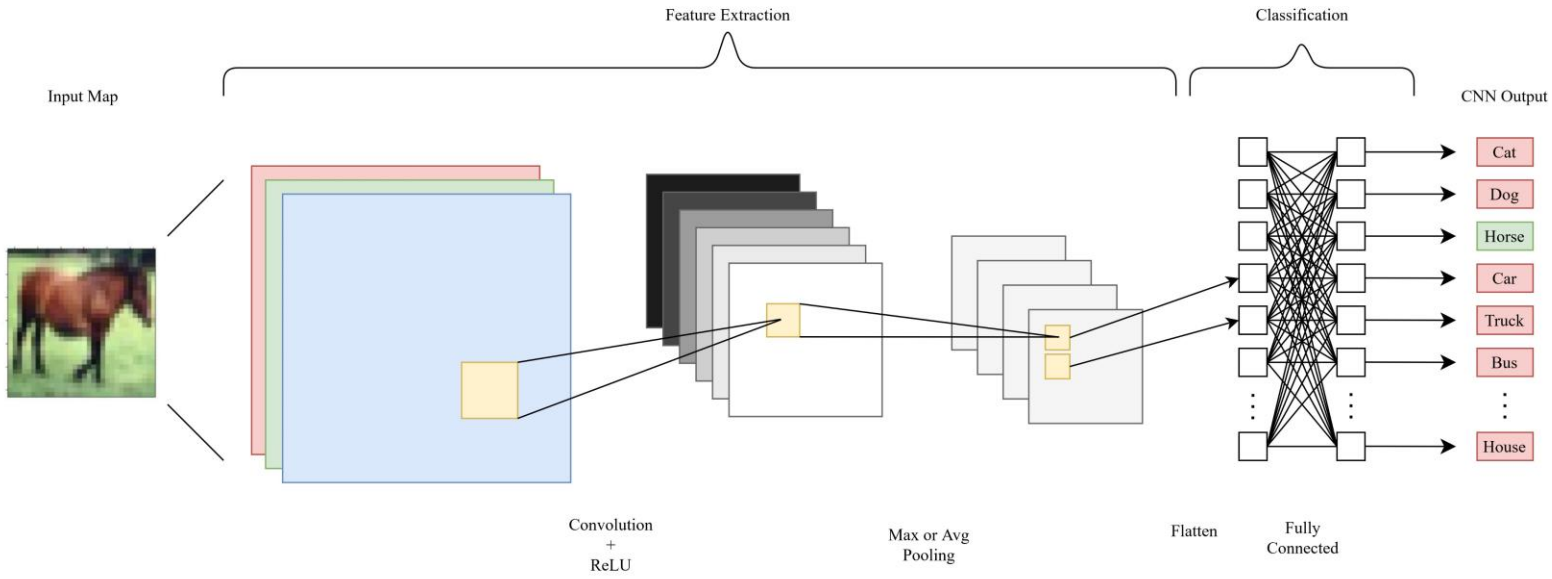
Overview

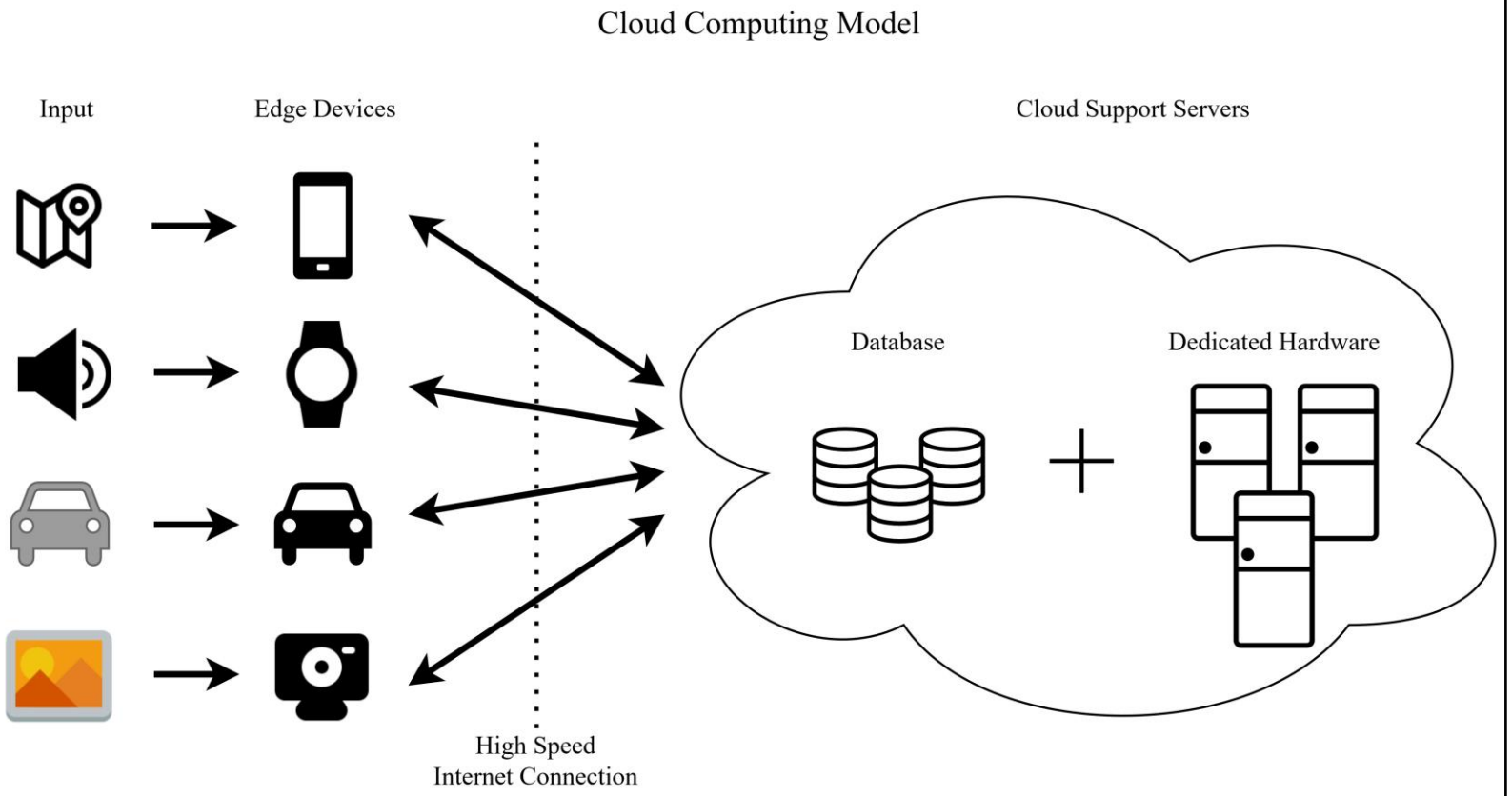
Computer vision is a deep learning application enabling object detection / classification

Unique edge devices with image and video data are being created for these applications in mind

The CNNs used are often complex and too computationally bulky for IoT edge devices





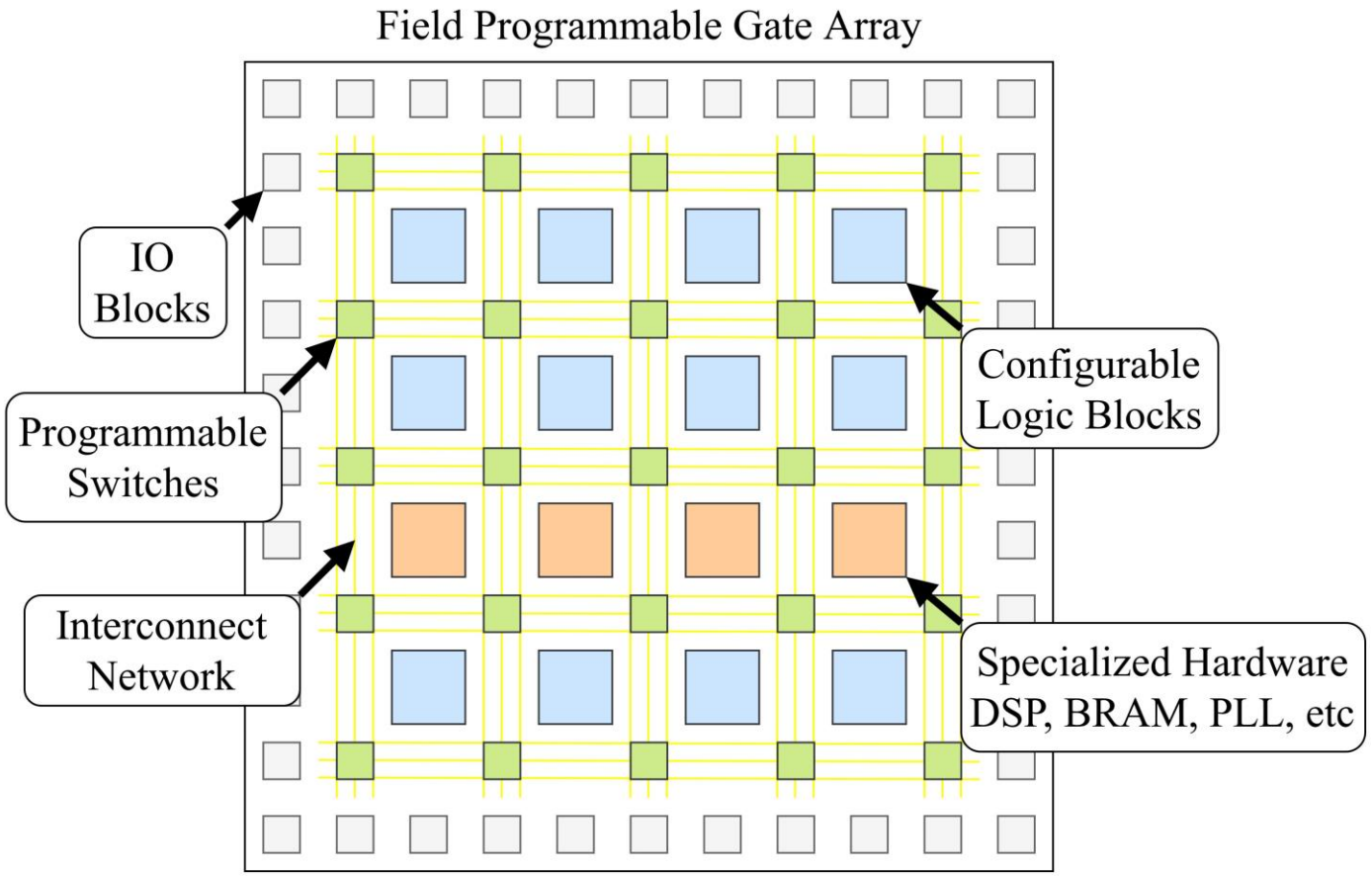


With traditional cloud computing models, edge devices are "dumb" data aggregators

Cloud servers perform CNN processes

Dedicated server hardware has much greater processing capabilities and relaxed power constraints compared to edge devices

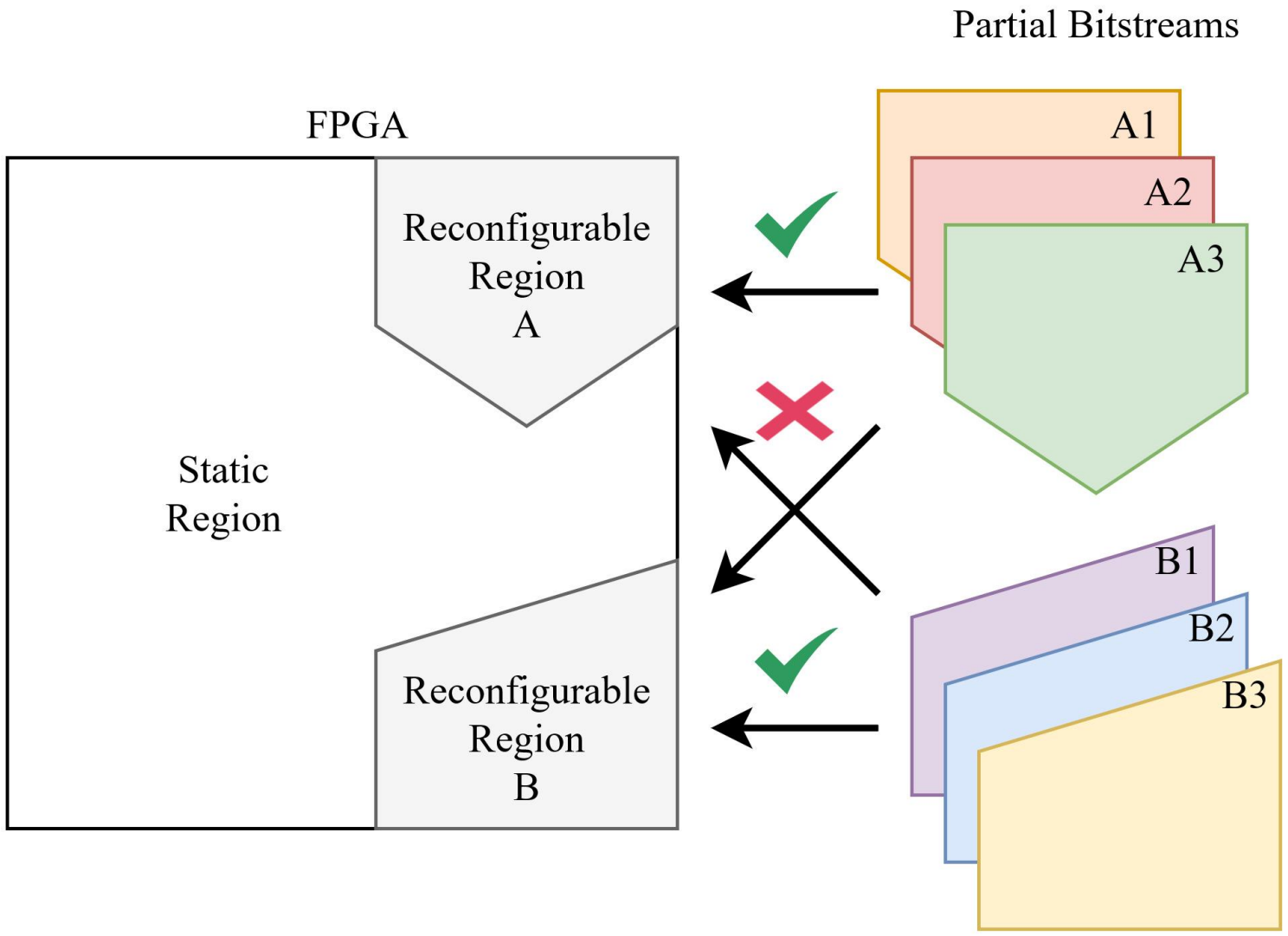
High connectivity required



Due to low development overhead, FPGAs are being investigated for accelerating CNN process at the edge devices

Reconfigurable computing devices change behavior or functionality at the hardware logic level

FPGAs are a member of reconfigurable hardware and offer **Dynamic Partial Reconfiguration**



DPR allows the device to partially reconfigure while the rest of the system remains unchanged

- **Offline** Reconfiguration
- **Online** Reconfiguration

Modern DPR design flows use reconfigurable regions with partition definitions to be loaded after full configuration

The partial bitstream file size directly correlates with the reconfigurable region size

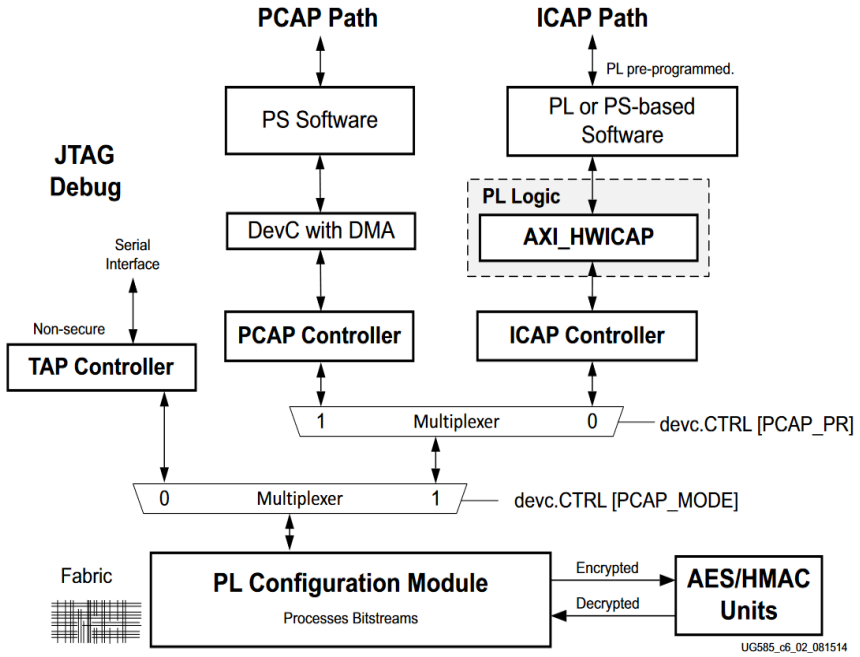


Figure 6-2: **PL Configuration Paths**

Ref: Xilinx UG585 Zynq-7000 SoC Technical Reference Manual

Table 8-4: **Maximum Bandwidths for Configuration Ports in 7 Series Architectures**

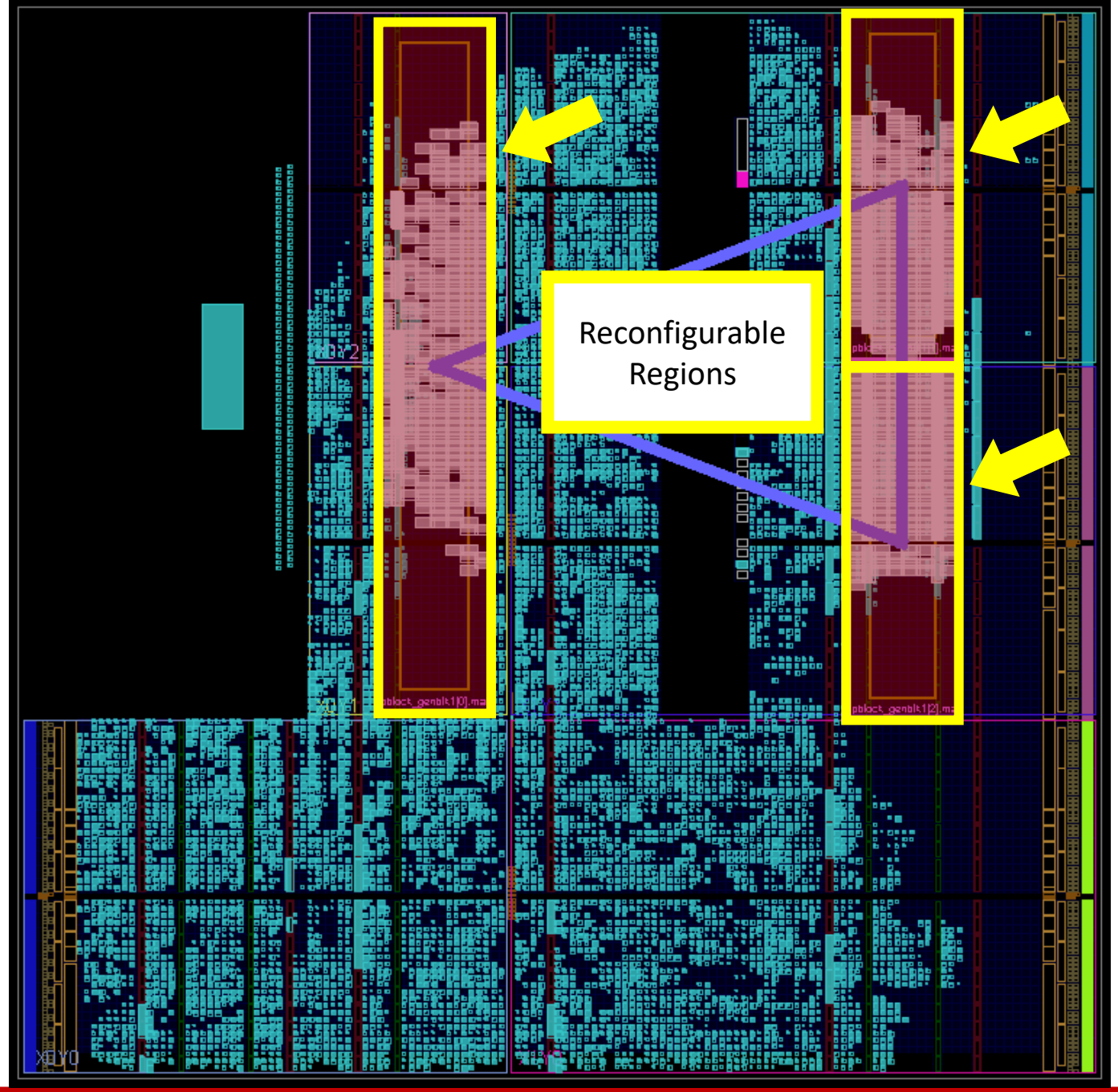
Configuration Mode	Max Clock Rate	Data Width	Maximum Bandwidth
ICAP	100 MHz	32 bit	3.2 Gb/s
SelectMAP	100 MHz	32 bit	3.2 Gb/s
Serial Mode	100 MHz	1 bit	100 Mb/s
JTAG	66 MHz	1 bit	66 Mb/s

Ref: Xilinx UG909 (v2018.1) Partial Reconfiguration

FPGA has several ways to programmed on boot that vary with device family

The Zynq device used in implementation allows programming over Serial, JTAG, ICAP, and PCAP

The port bandwidth **varies drastically**



Reconfigurable regions are manually mapped to the FPGA floorplan during Synthesis

If the partition definitions need special elements like DSP, the region must be mapped to include this

3 channel convolution processor with RR mapped to FPGA floorplan

Related Work

FINN and DNNWeaver two frameworks for generating fully synthesized CNN models to deploy on an FPGA target

- **Hardware Utilization**

FINN is developed by a Xilinx research group and specializes in low latency streaming applications

- **HLS**

DNNWeaver is an open-source framework that employs groups universal processing elements and a state machine to assign data to one layer of the CNN at a time

- **Caffe**

FINN Ref: https://xilinx.github.io/finn/about.html
DNNWeaver Reg: http://dnnweaver.org/

Ref: E. Youssef, H. A. Elsemary, M. A. El-Moursy, A. Khattab and H. Mostafa, "Energy Adaptive Convolution Neural Network Using Dynamic Partial Reconfiguration," 2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS), Springfield, MA, USA, 2020, pp. 325-328, doi: 10.1109/MWSCAS48704.2020.9184640.

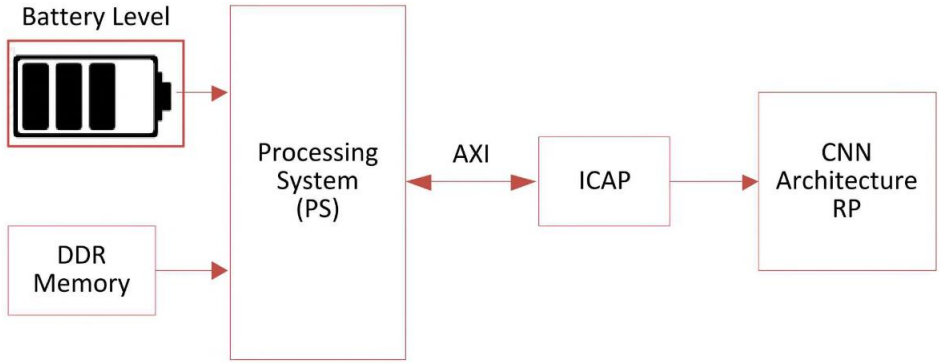


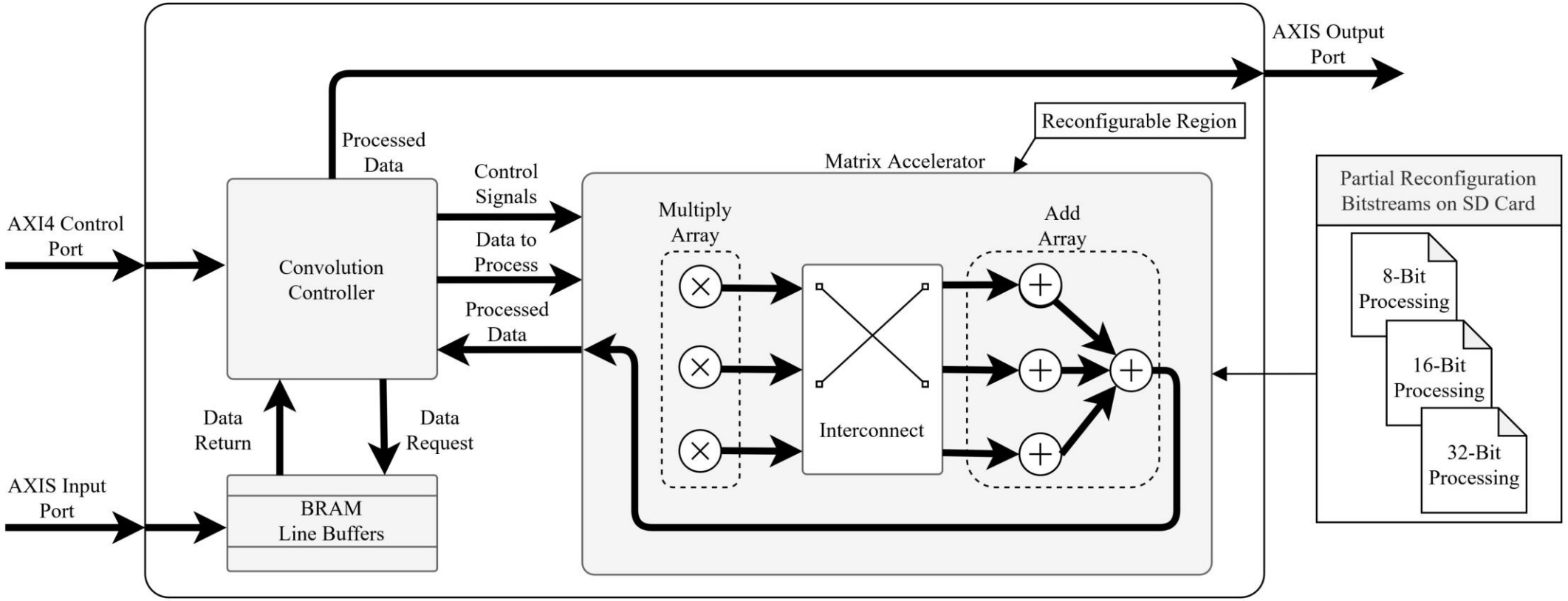
Fig. 5. DPR system block diagram.

Fig. 8. Accuracy loss vs. energy reduction with approximated CNN.

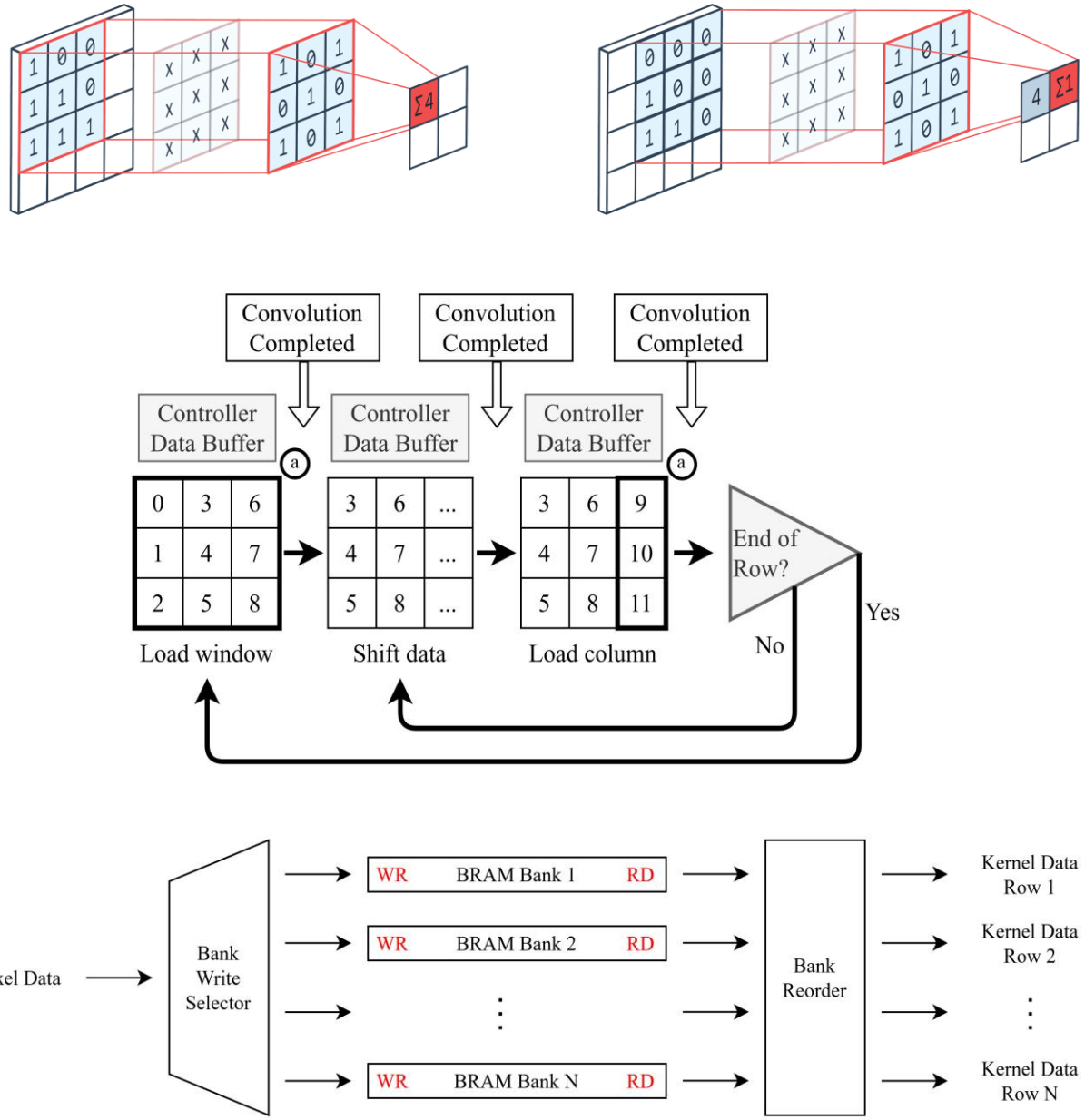
Youssef et. al developed an MNIST CNN architecture to support DPR for reducing bit precision which enables lower dynamic power consumption

The energy consumption of this model at 16 bits was shown to be 11.25μJ per image and at 5 bits, 3.02 μJ per image while retaining 81.74% accuracy

Proposed Convolution Processing Element



The proposed Convolution Processing Element utilizes DPR at a processing element level whereas traditional methods must either fully reconfigure or use DPR at a pipeline level

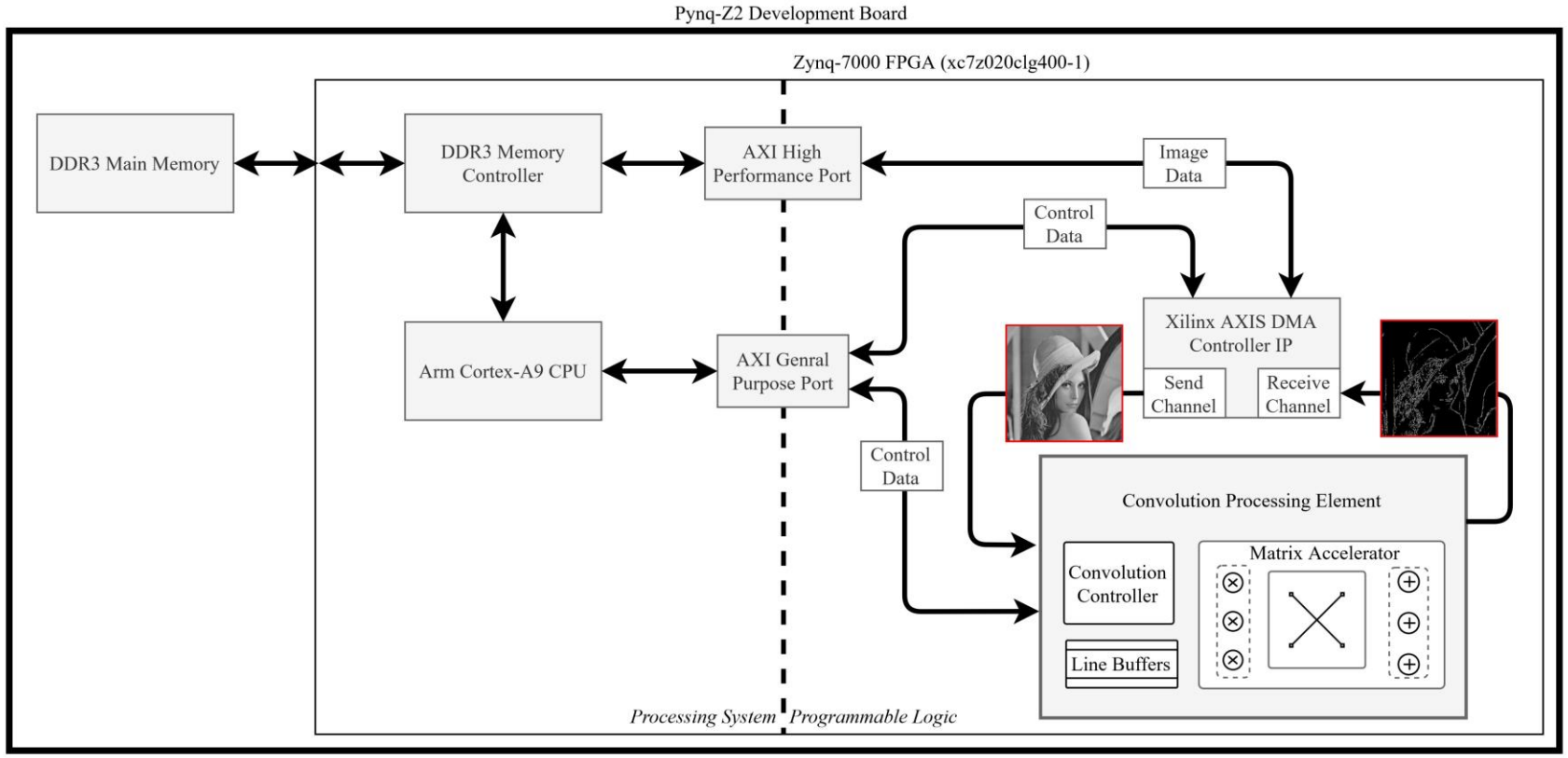


The convolution controller handles the data stream with an internal state machine

For a kernel of size N, the line buffers temporarily store N rows of input map as each convolution requires NxN data points from N rows and N columns

Line buffers are implemented to infer into BRAM as to not waste available device logic

Hardware Implementation & Comparisons



Implemented on a Pynq-Z2 which features a Zynq-7000 FPGA

Reconfigurable between 8-, 16-, and 32-bit precision with 1 or 3 processing channels

Benchmarked using PYNQ Python Jupyter Notebook and the Vitis C environment

TABLE II
THREE CHANNEL HARDWARE UTILIZATION BREAKDOWN

	Component	LUTs	DSPs
	Controller	5503	0
32-bit	Matrix Accelerator	271	27
	CPE	6316 (11.87%)	81 (36.82%)
16-bit	Matrix Accelerator	365	9
	CPE	6610 (12.42%)	27 (12.27%)
8-bit	Matrix Accelerator	721	0
	CPE	7741 (14.55%)	0 (0.00%)

TABLE III
HARDWARE UTILIZATION COMPARED TO OTHER DESIGNS

	[14] (Data reuse)	[16]	CPE	CPE	CPE
Data Type	N/A	N/A	Integer	Integer	Integer
Precision	16-bit	N/A	32-bit	16-bit	8-bit
Kernel Size	3×3	3×3	3×3	3×3	3×3
LUTs	684	38069	5291	5398	5785
FFs	672	18557	3762	3597	3661
BRAMs	N/A	72	6	6	6
DSPs	18	3	27	9	0

The proposed 3-channel CPE hardware utilization was compared against other designs

At the 32-bit configuration, DSP utilization reaches 36% of total available for the device

Rupani et. al use a design which buffers the full image which causes high BRAM utilization

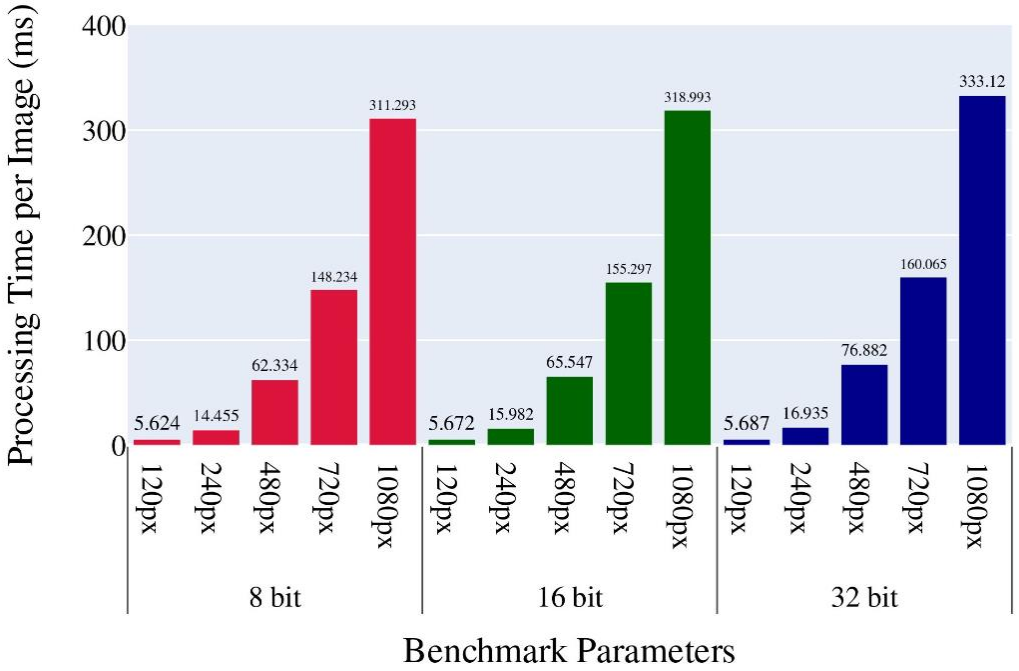


Fig. 4. CPE Average Processing Times for Single Channel Images

TABLE IV
CPE Runtime Analysis on 64x64 Input Map

CPE Channels	Input Channels	Conv. per Frame	Cycles per Frame
1	1	3969	31190
1	3	11907	93570
3	3	11907	47070
CPE Channels	Input Channels	Process FPS	Process Latency
1	1	3206	N/A
1	3	1068	N/A
3	3	2124	N/A

When testing with Jupyter Notebooks (Left), reconfiguration was completed using JTAG

Benchmarking was moved to Vitis (Right) which enabled use of the ICAP for reconfiguration

$$Reconfiguration\ Time = \frac{Bitstream\ Size}{Port\ Bandwidth}$$

(Traditional) Full Reconfiguration Using JTAG	**Partial Reconfiguration Using JTAG**	**Partial Reconfiguration Using ICAP**
Bitstream Size: 32,364,512 b	Bitstream Size: 1,766,976 b	Bitstream Size: 1,766,976 b
Port bandwidth: 66 Mb/s	Port bandwidth: 66 Mb/s	Port bandwidth: 3.2 Gb/s
Reconfiguration Time: 490.371 ms	Reconfiguration Time: 26.772 ms	Reconfiguration Time: 0.552 ms

As traditional methods would reconfigure the full device via JTAG,
using DPR at a processing element level reduces configuration time by 800x

Conclusions

With the reduced development time compared to other hardware solutions, **reconfigurable hardware enables power efficient processors to be developed for application specific needs**

Reconfigurable hardware offers **Dynamic Partial Reconfiguration** which **greatly suits the constrained hardware systems found in embedded devices**

The proposed design provides insight to a design flow which enables system downtime to be reduced if pipelines are to be strategically reconfigured

The proposed design can be elaborated further to reuse hardware for process parallelization at lower precision, etc.

Questions?

Thank You