GStreamer-integrated HLS-based JPEG Encoder for Edge FPGA SoCs

Yuri Guimaraes Pereira Primo da Silva[1], Shinya Honda[1], Sugako Otani[12], Masato Edahiro[1], Abraham Monrroy Cano[3]

[1]Nagoya University, [2]Renesas Electronics, [3]Map IV Inc.

Motivation

1. Lack of freely available image encoder designed for edge use case.
2.

Approach

Architecture

Results & Comparisons

Future Work

GStreamer-integrated HLS-based JPEG Encoder for Edge FPGA SoCs

Yuri Guimaraes Pereira Primo da Silva[1], Shinya Honda[1], Sugako Otani[12], Masato Edahiro[1], Abraham Monrroy Cano[3]

[1]Nagoya University, [2]Renesas Electronics, [3]Map IV Inc.

E-mail: yuri_gpps1@ertl.jp

Motivation

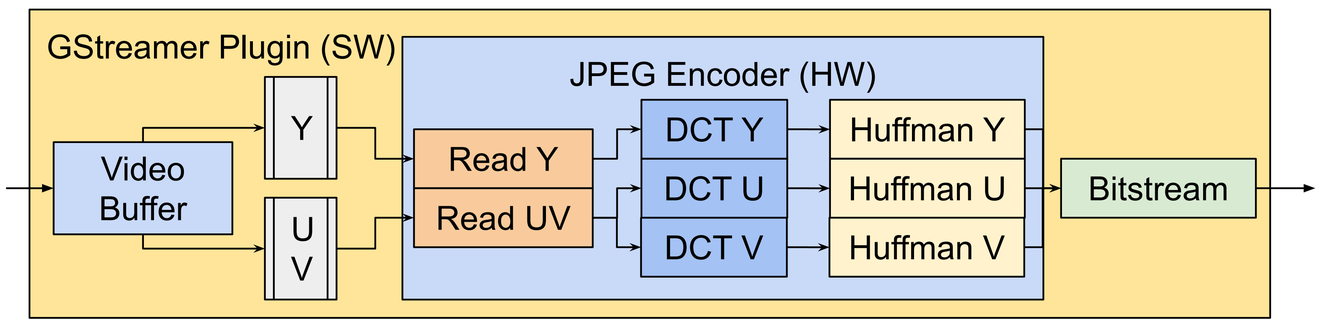
1. Lack of freely available image encoder designed for edge use case.
2.

Approach

JPEG HLS Encoder

GStreamer Plugin Integration

Architecture



Experiments

Future Work

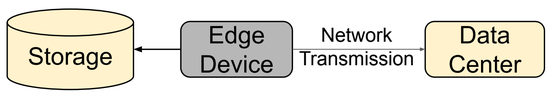
GStreamer-integrated HLS-based JPEG Encoder for Edge FPGA SoCs

Yuri Guimaraes Pereira Primo da Silva[1], Shinya Honda[1], Sugako Otani[12], Masato Edahiro[1], Abraham Monrroy Cano[3]

[1]Nagoya University, [2]Renesas Electronics, [3]Map IV Inc.

E-mail: yuri_gpps1@ertl.jp

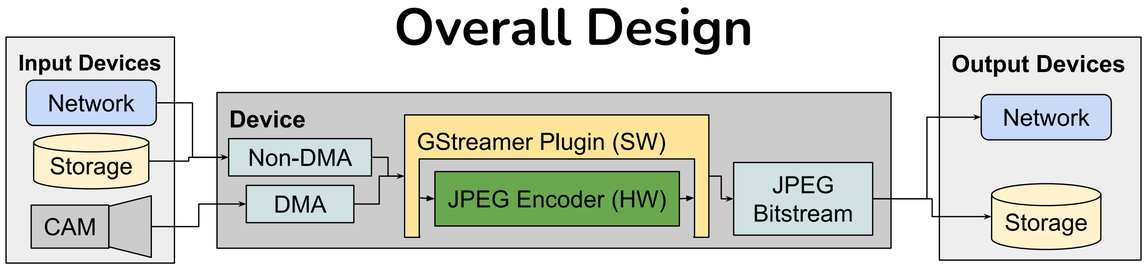
Motivation



- Increasing importance of image encoding at the edge.
- **Image encoding** efforts have been mostly concentrated on accelerator cards.
- This research focuses on the implementation and integration aspect of an **HLS-based JPEG encoder** on Kria KV260 SoC [1] integrated with **GStreamer**.

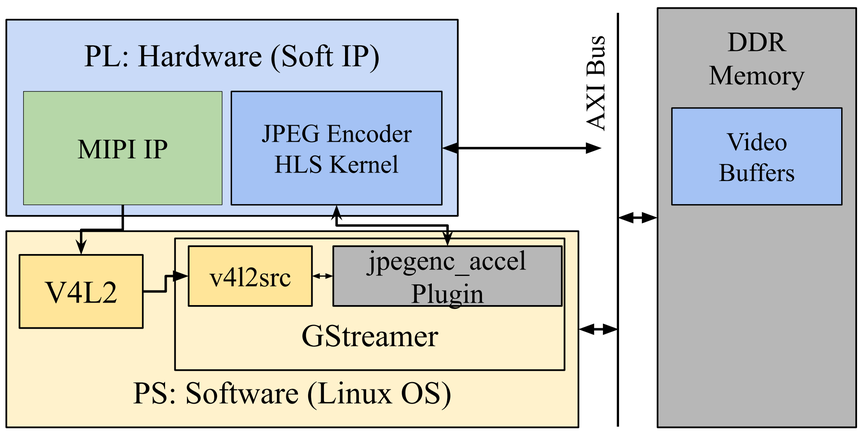
Approach

Overall Design



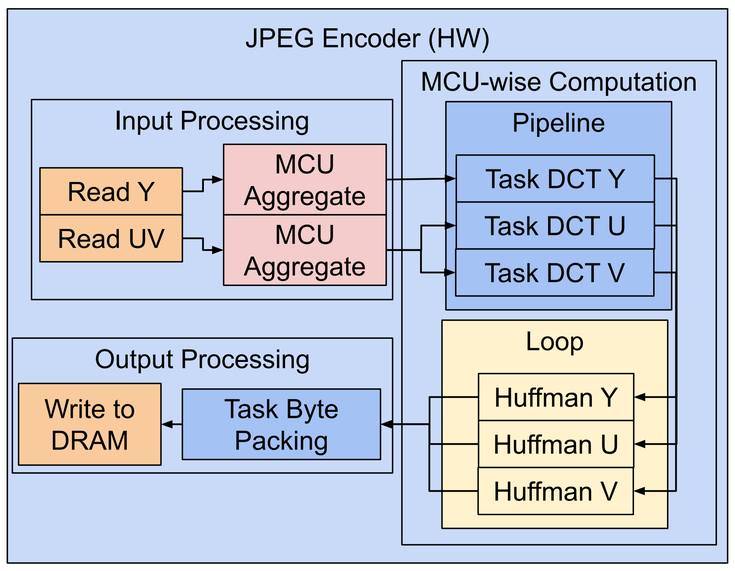
- HW JPEG Encoder implemented in *High-level Synthesis* [2].
- *GStreamer* plugin interfaces with encoder.
  - Allows for mix and matching of pipeline elements for arbitrary media processing.
  - Easy usage of DMA capabilities.

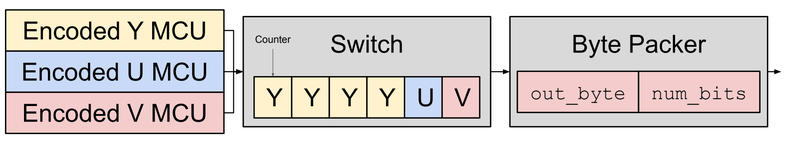
Top-level Block Design



Hardware Architecture

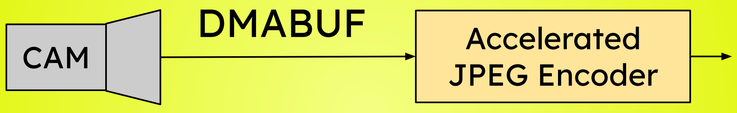
- Conventional JPEG algorithm design implemented C++ HLS.
- Streaming design:
  - Data is packed into MCU streams at input processing.
  - Each MCU in a stream goes through pipelined DCT and RLE pipelined, and sequential Huffman.
  - MCUs are packed into the expected JPEG bitstream format at the end.
- Encoded bits are packed via a stateful switch that shifts and ORs according to the encoding order (e.g. YYYYUV for YUV420).



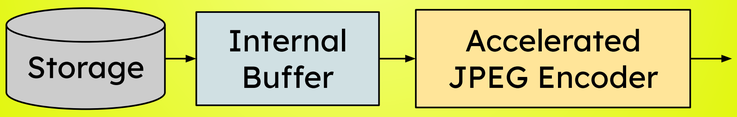


Pipelines

cam: From Camera Device (DMA-able)



file: From Storage (Requires Copy)



Performance Evaluation

FPGA vs. CPU (jpegenc)

		CPU	FPGA
cam	FPS	2.6	**19.5**
	CPU %	93	**5**
file	FPS	**22.6**	21.7
	CPU %	107	**14**

Resource Utilization

Generally, *Vitis Libraries* [4] codec kernels **exceed resource limitations** for edge boards, such as the **KV260**. Thus, employing these kernels on such boards often requires **lowering the target frequency**.
Our encoder is able to achieve **7~25%** of the resource usage of even the smallest Vitis encoding kernel (Webp at 100MHz), as shown below.
This is done possible by reducing the **HW functionality** as much as possible, and thus lowering complexity.

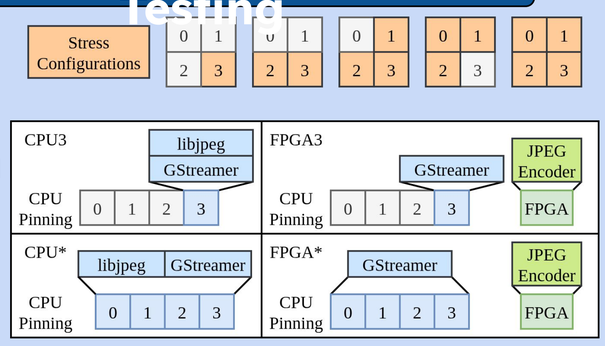
	Freq.	BRAM	URAM	DSP	FF	LUT
JPEG[1]	250MHz	**6**	**12**	**85**	**18150**	**13868**
Webp[2]	100MHz	81	46	834	71227	68906
Webp[2]	250MHz	229	10	414	92030	68755
JXL[2]	260MHz	584	122	644	270565	199537
PIK[2]	200MHz	614	473	2398	549987	449995

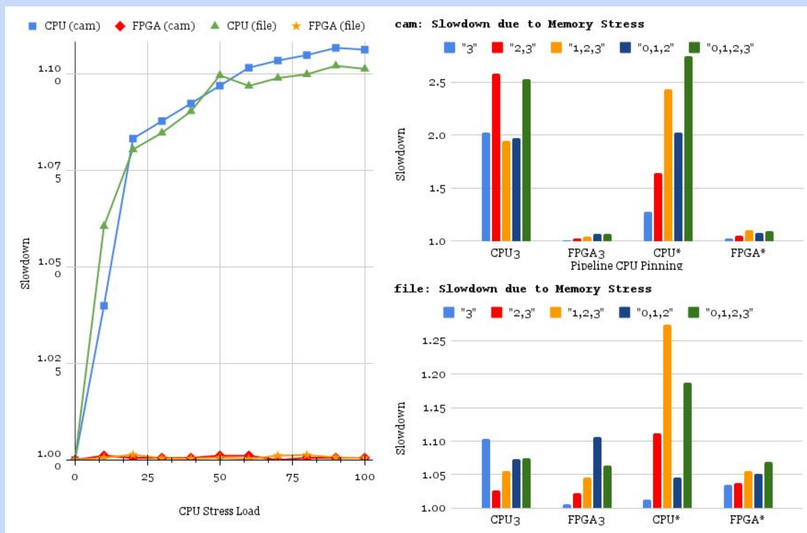
[1]Our work [2]Vitis Codec Libraries [4]

Experiments

CPU/Memory Stress Testing

Objective: Test the **slowdown** behavior of both the **jpegenc-based CPU encoder** and our **FPGA encoder** when the system is under load.





Discussion and Conclusions

1. Suitability for Edge
2.
3. Increased Predictability

References

[1] AMD. "Kria KV260 Vision AI Starter Kit". https://www.amd.com/en/products/system-on-modules/kria/k26/kv260-vision-starter-kit.html. Accessed 2025-08-20.
[2] AMD. "Vitis HLS". https://www.amd.com/en/products/software/adaptive-socs-and-fpgas/vitis/vitis-hls.html. Accessed 2025-05-30.
[4] AMD. "Vitis Libraries". https://github.com/Xilinx/Vitis_Libraries. Accessed 2025-05-30.

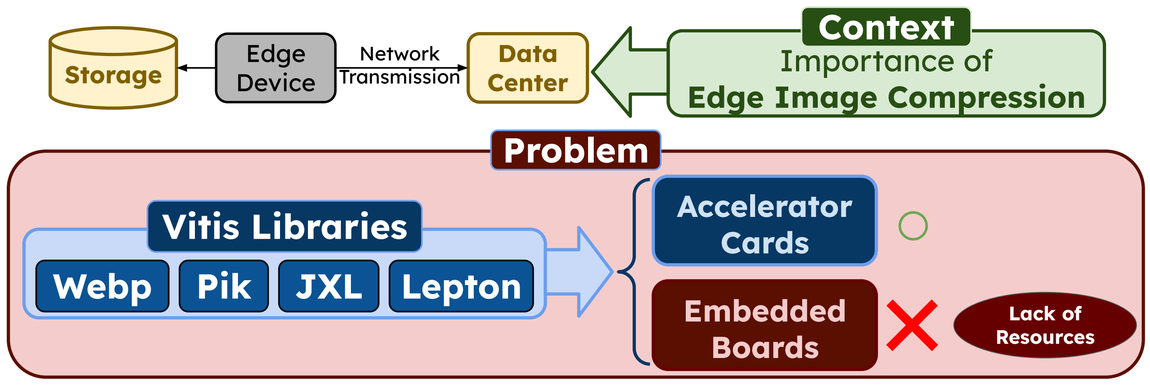
GStreamer-integrated HLS-based JPEG Encoder for Edge FPGA SoCs

Yuri Guimaraes Pereira Primo da Silva[1], Shinya Honda[1], Sugako Otani[12], Masato Edahiro[1], Abraham Monrroy Cano[3]

[1]Nagoya University, [2]Renesas Electronics, [3]Map IV Inc.

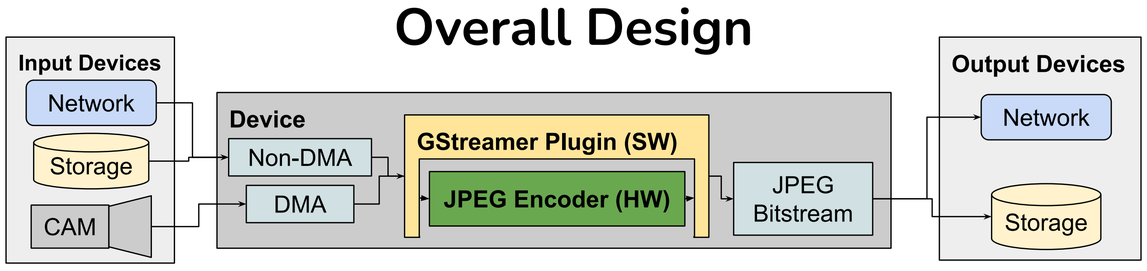
E-mail: yuri_gpps1@ertl.jp

Motivation



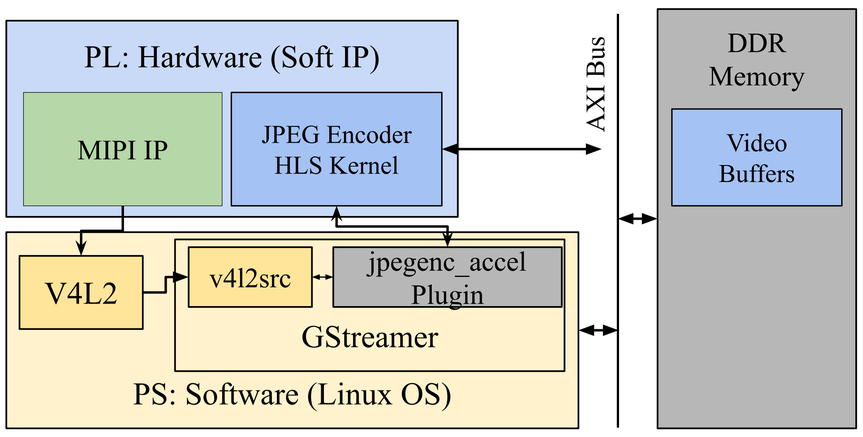
Approach

Overall Design



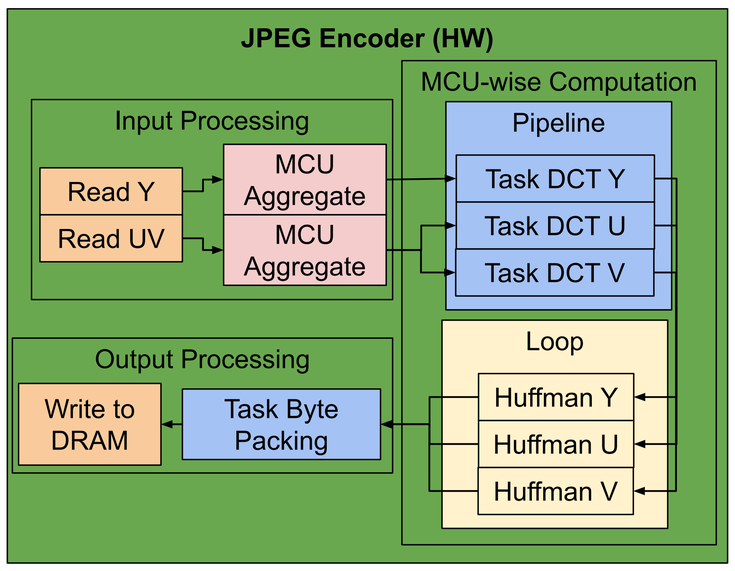
- HW JPEG Encoder implemented in *High-level Synthesis* [2].
- *GStreamer* plugin interfaces with encoder.
  - Allows for mix and matching of pipeline elements for arbitrary media processing.
  - Easy usage of DMA capabilities.

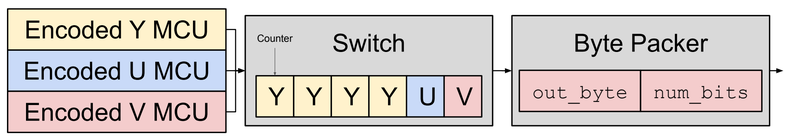
Top-level Block Design



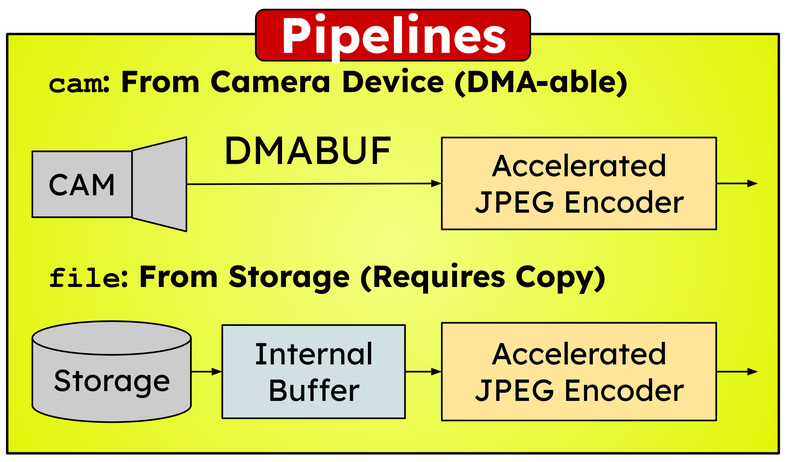
Hardware Architecture

- Conventional JPEG algorithm design implemented C++ HLS.
- Streaming design:
  - Data is packed into MCU streams at input processing.
  - Each MCU in a stream goes through pipelined DCT and RLE pipelined, and sequential Huffman.
  - MCUs are packed into the expected JPEG bitstream format at the end.
- Encoded bits are packed via a stateful switch that shifts and ORs according to the encoding order (e.g. YYYYUV for YUV420).





Pipelines



Performance Evaluation

FPGA vs. CPU (jpegenc)

		CPU	FPGA
cam	FPS	2.6	**19.5**
	CPU %	93	**5**
file	FPS	**22.6**	21.7
	CPU %	107	**14**

Resource Utilization

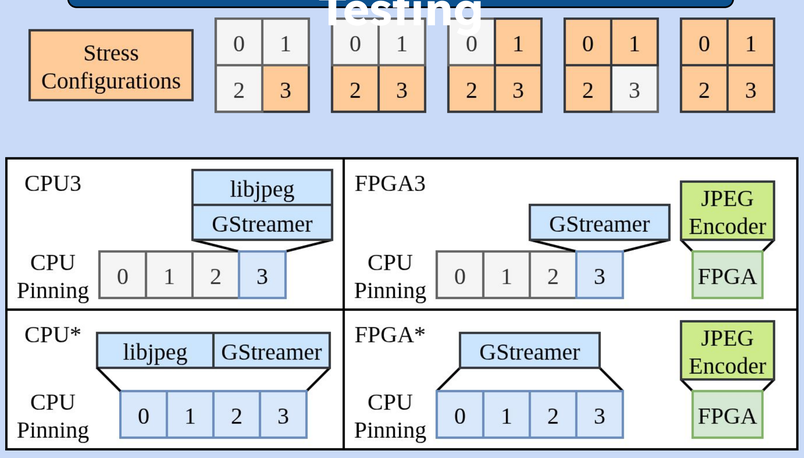
Generally, *Vitis Libraries* [4] codec kernels **exceed resource limitations** for edge boards, such as the **KV260**. Thus, employing these kernels on such boards often requires **lowering the target frequency**.
Our encoder is able to achieve **7~25%** of the resource usage of even the smallest Vitis encoding kernel (Webp at 100MHz), as shown below.
This is done possible by reducing the **HW functionality** as much as possible, and thus lowering complexity.

	Freq.	BRAM	URAM	DSP	FF	LUT
JPEG[1]	250MHz	**6**	**12**	**85**	**18150**	**13868**
Webp[2]	100MHz	81	46	834	71227	68906
Webp[2]	250MHz	229	10	414	92030	68755
JXL[2]	260MHz	584	122	644	270565	199537
PIK[2]	200MHz	614	473	2398	549987	449995

[1]Our work [2]Vitis Codec Libraries [4]

Experiments

CPU/Memory Stress Testing



Objective: Test the **slowdown** behavior of both the **jpegenc-based CPU encoder** and our **FPGA encoder** when the system is under load.

Discussion and Conclusions

1. Suitability for Edge
2.
3. Increased Predictability

References

[1] AMD. "Kria KV260 Vision AI Starter Kit". https://www.amd.com/en/products/system-on-modules/kria/k26/kv260-vision-starter-kit.html. Accessed 2025-08-20.
[2] AMD. "Vitis HLS". https://www.amd.com/en/products/software/adaptive-socs-and-fpgas/vitis/vitis-hls.html. Accessed 2025-05-30.
[4] AMD. "Vitis Libraries". https://github.com/Xilinx/Vitis_Libraries. Accessed 2025-05-30.

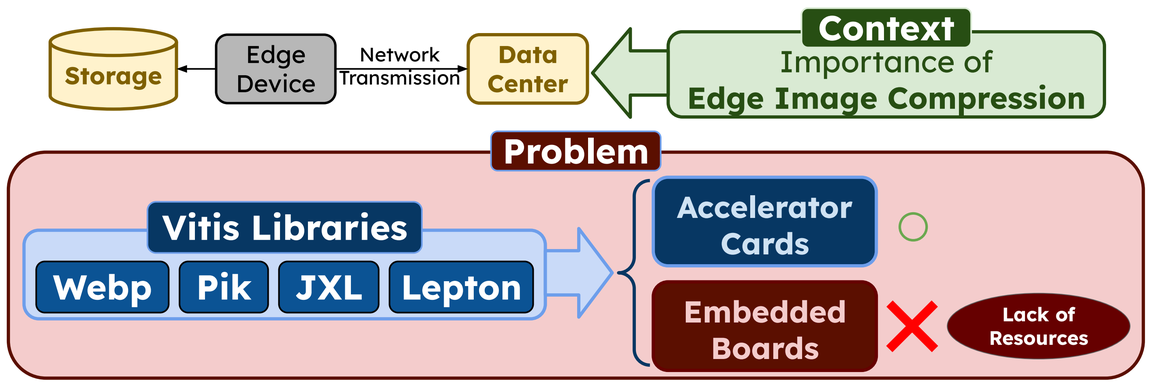
GStreamer-integrated HLS-based JPEG Encoder for Edge FPGA SoCs

Yuri Guimaraes Pereira Primo da Silva[1], Shinya Honda[1], Sugako Otani[12], Masato Edahiro[1], Abraham Monrroy Cano[3]

[1]Nagoya University, [2]Renesas Electronics, [3]Map IV Inc.

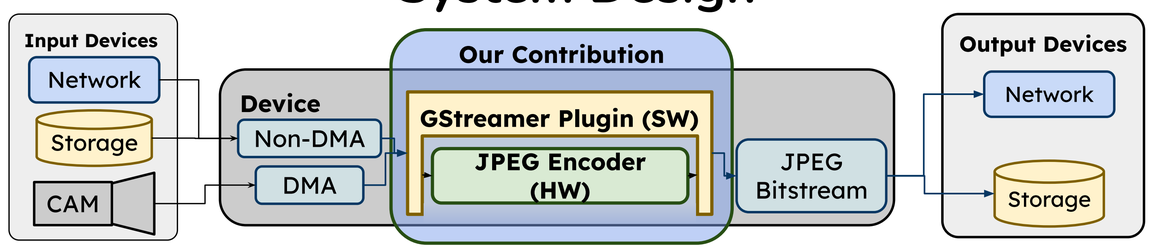
E-mail: yuri_gpps1@ertl.jp

Motivation



Approach

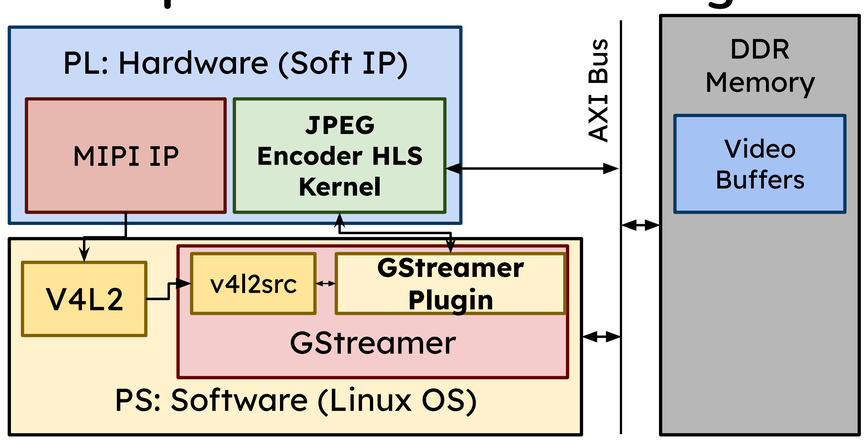
System Design



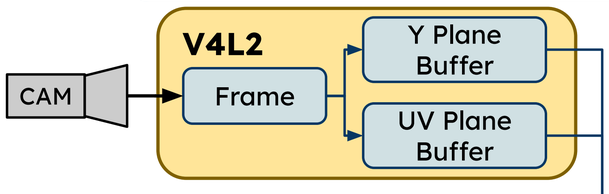
Contributions

- **HW JPEG Encoder** implemented in *High-level Synthesis.*
- ***GStreamer* plugin** interfaces with encoder.
  - Allows for mix and matching of pipeline elements for arbitrary media processing.
  - Easy usage of DMA capabilities.

Top-level Hardware Design

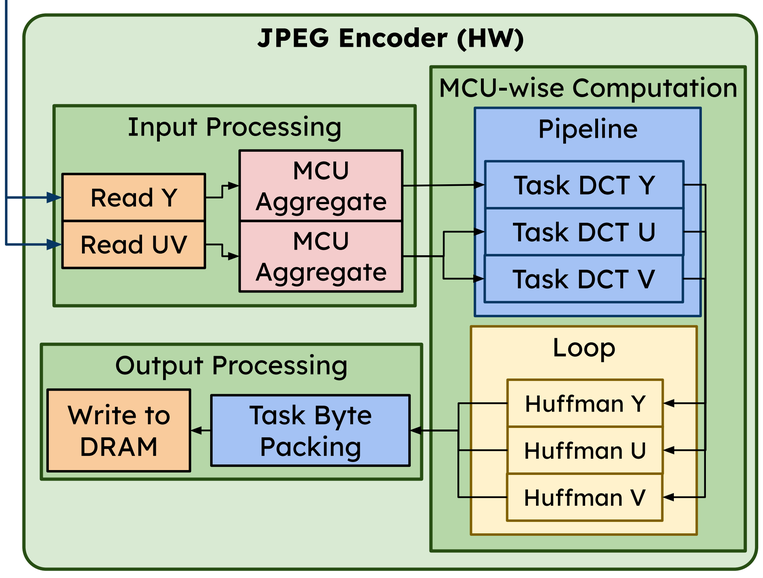


Hardware Architecture



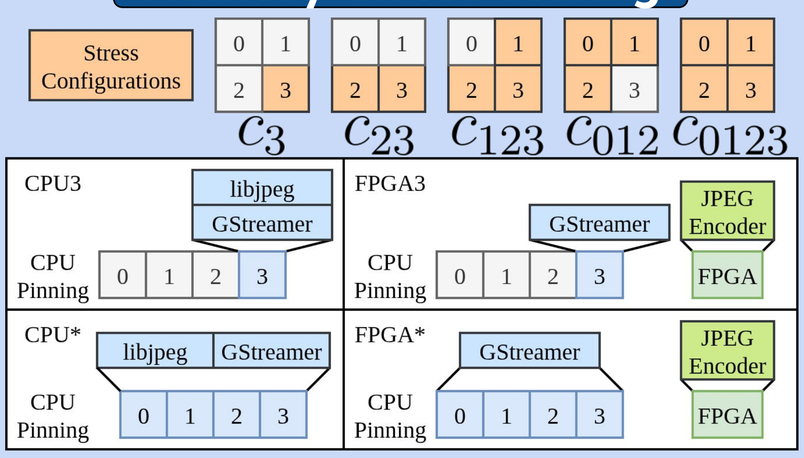
- Send each plane to the encoder via AXI
  - Avoid memory copy in case of DMA

- Conventional JPEG algorithm design implemented C++ HLS.
- Streaming design:
  - Data is packed into MCU streams at input processing.
  - Each MCU in a stream goes through pipelined DCT and RLE pipelined, and sequential Huffman.
  - MCUs are packed into the expected JPEG bitstream format at the end.

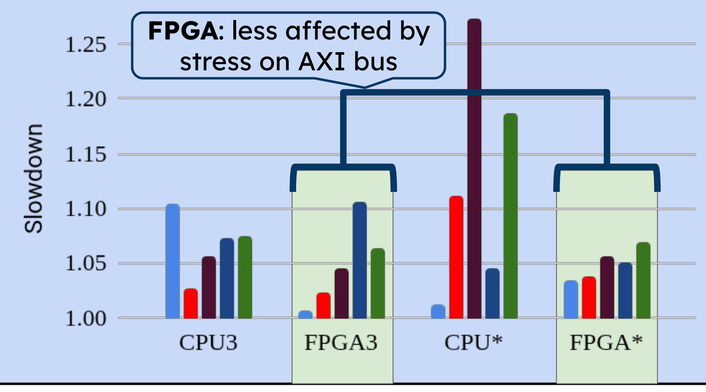


Experiments

Memory Stress Testing

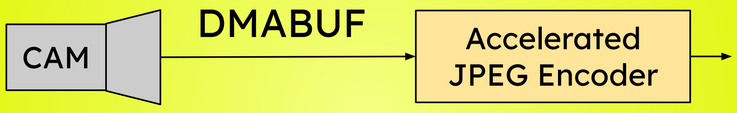


`file`: **Slowdown Comparison under Memory Stress**

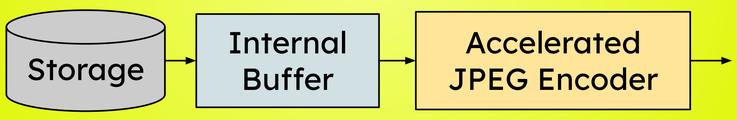


Definition: Pipelines

`cam`: From Camera Device (DMA-able)



`file`: From Storage (Requires Copy)



Performance Evaluation

FPGA vs. CPU (jpegenc)

		CPU	FPGA
cam	FPS	2.6	**19.5**
	CPU %	93	**5**
file	FPS	**22.6**	21.7
	CPU %	107	**14**

Resource Utilization

	Freq.	BRAM	URAM	DSP	FF	LUT
JPEG[1]	250MHz	**6**	**12**	**85**	**18150**	**13868**
Webp[2]	100MHz	81	46	834	71227	68906
Webp[2]	250MHz	229	10	414	92030	68755
JXL[2]	260MHz	584	122	644	270565	199537
PIK[2]	200MHz	614	473	2398	549987	449995

Webp Encoder[2]

- **High Resource Usage**
  - Requires lowering **target frequency** to 100Hz

JPEG Encoder[1]

- **7~25%** of Webp Resources
  - Can fit **four** JPEG encoders in place of a **single** Webp Encoder

[1]Our work [2]Vitis Codec Libraries

Discussion and Conclusions

1. Suitability for Edge
   - Low resource usage
   - Realtime performance
2. Usability
   - GStreamer integration
     - Easy composition of media pipelines
3. Increased Predictability
   - Resilience against memory stress

Future Work

- Streamlined flow for GStreamer integration with arbitrary HLS kernels

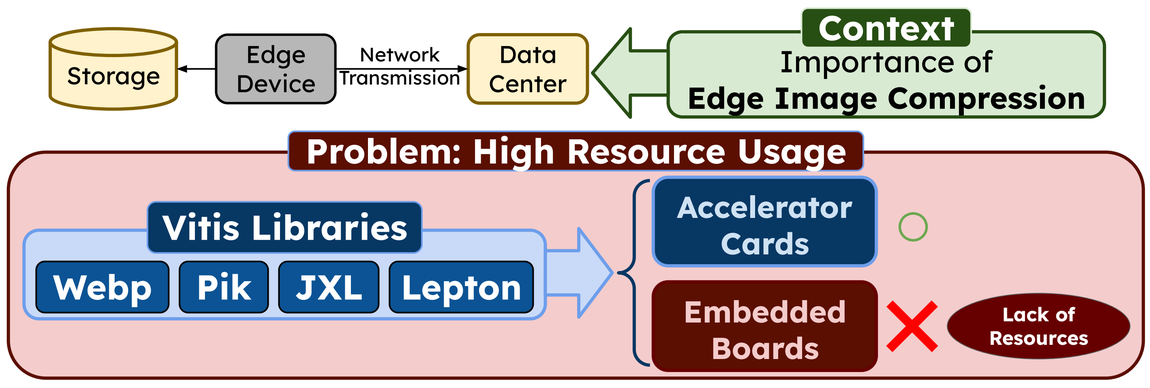
GStreamer-integrated HLS-based JPEG Encoder for Edge FPGA SoCs

Yuri Guimaraes Pereira Primo da Silva[1], Shinya Honda[1], Sugako Otani[12], Masato Edahiro[1], Abraham Monrroy Cano[3]

[1]Nagoya University, [2]Renesas Electronics, [3]Map IV Inc.

E-mail: yuri_gpps1@ertl.jp

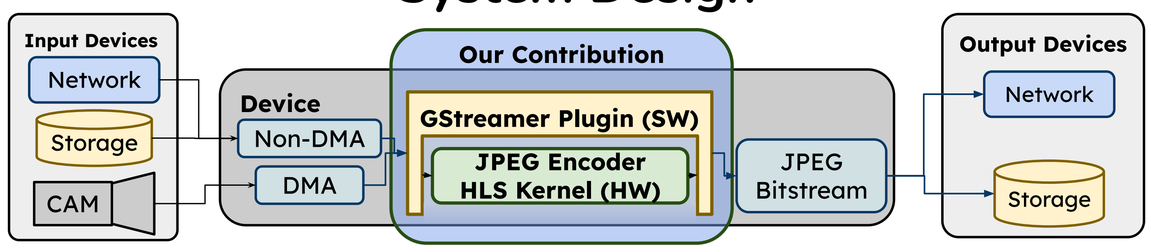
Motivation



Lack of low-resource HLS solutions for real-time FullHD image compression

Approach

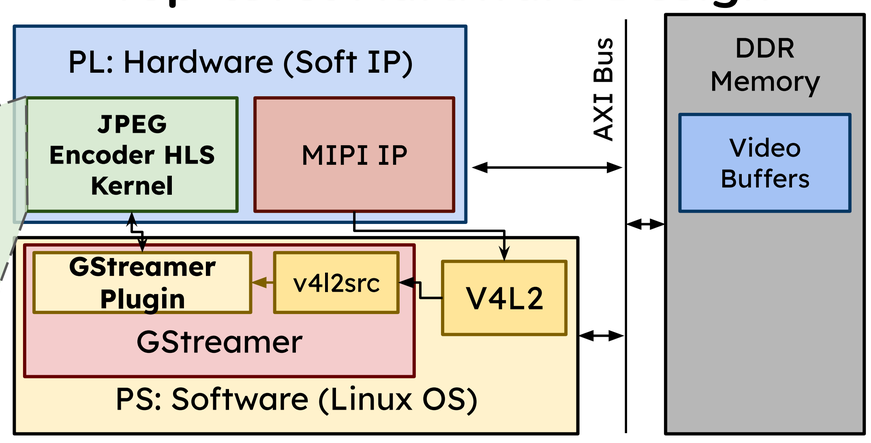
System Design



Contributions

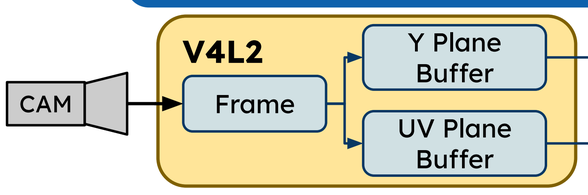
- **HW JPEG Encoder** implemented in *High-level Synthesis*.
- ***GStreamer* plugin** interfaces with encoder.
  - Allows for mix and matching of pipeline elements for arbitrary media processing.
  - Easy usage of DMA capabilities.

Top-level Hardware Design



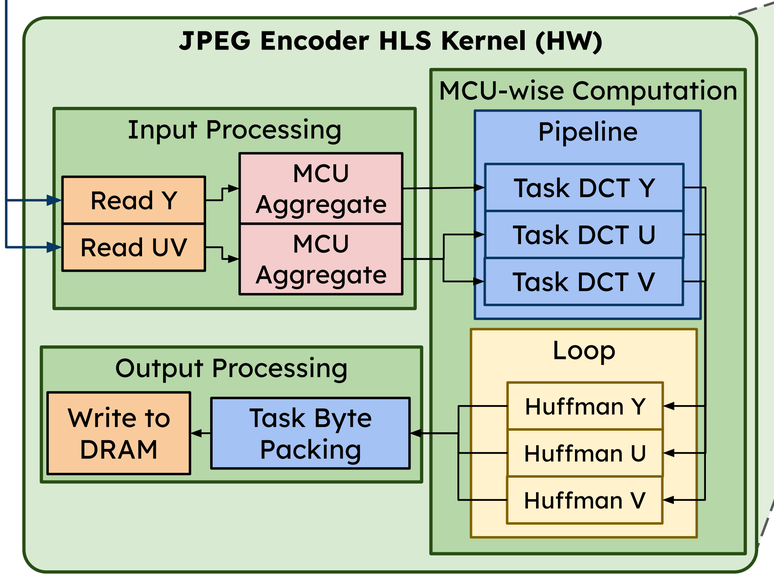
- Camera accessed through V4L2
  - Accesses must go through AXI bus

Hardware Architecture



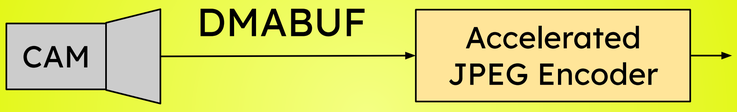
- Send each plane to the encoder via AXI
  - Avoid memory copy in case of DMA

- Conventional JPEG algorithm design implemented C++ HLS
- Streaming design:
  - Data is packed into **MCU** streams at input processing
  - Each **MCU** in a stream goes through **DCT and RLE** pipelined, and **sequential Huffman**
  - MCUs are packed into the expected JPEG bitstream format at the end.

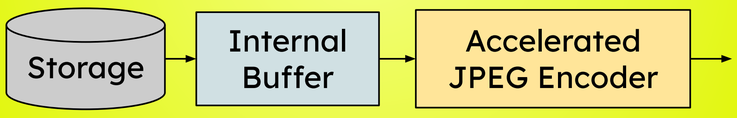


Definition: Pipelines

cam: From Camera Device (DMA-able)



file: From Storage (Requires Copy)



Performance

FPGA vs. CPU (jpegenc)

		CPU	FPGA	FPGA/CPU
cam	FPS	2.6	**19.5**	**7.5**
	CPU %	93	**5**	**0.054**
file	FPS	**22.6**	21.7	.96
	CPU %	107	**14**	**0.13**

Resource Utilization

JPEG (our) vs. Vitis Codec Libraries

	Freq.	BRAM	URAM	DSP	FF	LUT
JPEG[1]	250MHz	**6**	**12**	**85**	**18150**	**13868**
Webp[2]	100MHz	81	46	834	71227	68906
Webp[2]	250MHz	229	10	414	92030	68755
JXL[2]	260MHz	584	122	644	270565	199537
PIK[2]	200MHz	614	473	2398	549987	449995

[1]Our work
[2]Vitis Codec Libraries

Webp Encoder[2]

- **High Resource Usage**
  - Requires lowering **target frequency** to 100Hz

JPEG Encoder[1]

- **7~25%** of Webp Resources
  - Can fit **four** JPEG encoders in place of a **single** Webp Encoder

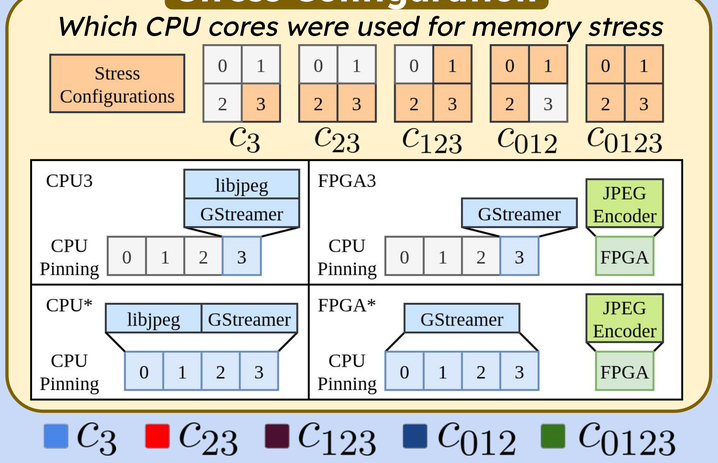
Experiments

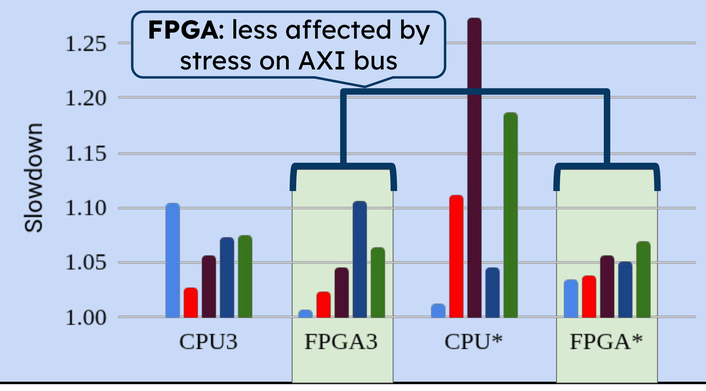
Memory Stress Testing

Slowdown Comparison under Memory Stress

Stress Configuration

Which CPU cores were used for memory stress





Discussion and Conclusions

1. Suitability for Edge
   - Low resource usage
   - Realtime performance
2. Usability
   - GStreamer integration
     - Easy composition of media pipelines
3. Increased Predictability
   - Resilience against memory stress

Future Work

- Streamlined flow for GStreamer integration with arbitrary HLS kernels

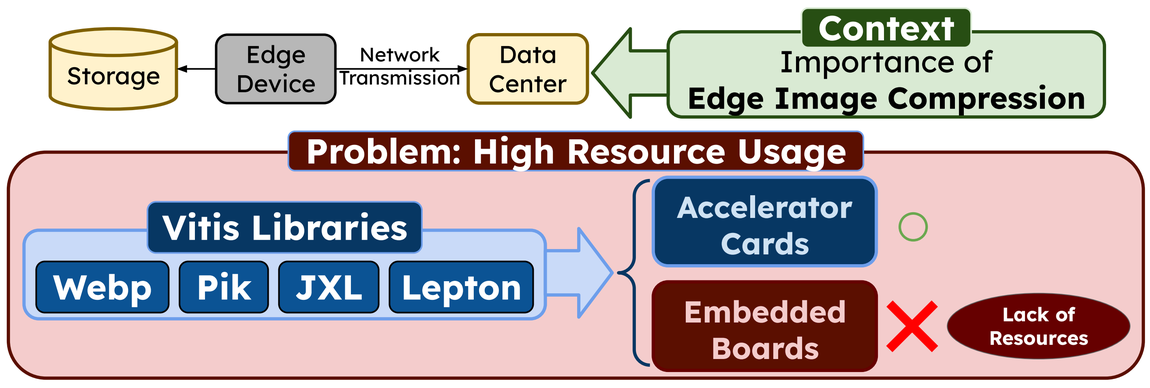
GStreamer-integrated HLS-based JPEG Encoder for Edge FPGA SoCs

Yuri Guimaraes Pereira Primo da Silva[1], Shinya Honda[1], Sugako Otani[12], Masato Edahiro[1], Abraham Monrroy Cano[3]

[1]Nagoya University, [2]Renesas Electronics, [3]Map IV Inc.

E-mail: yuri_gpps1@ertl.jp

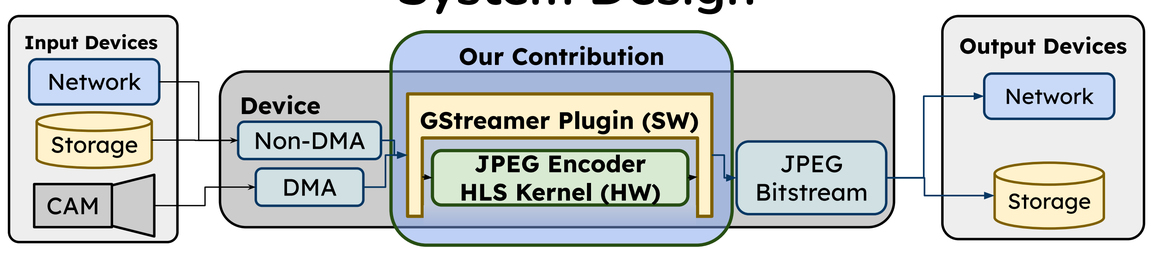
Motivation



Lack of low-resource HLS solutions for real-time FullHD image compression

Approach

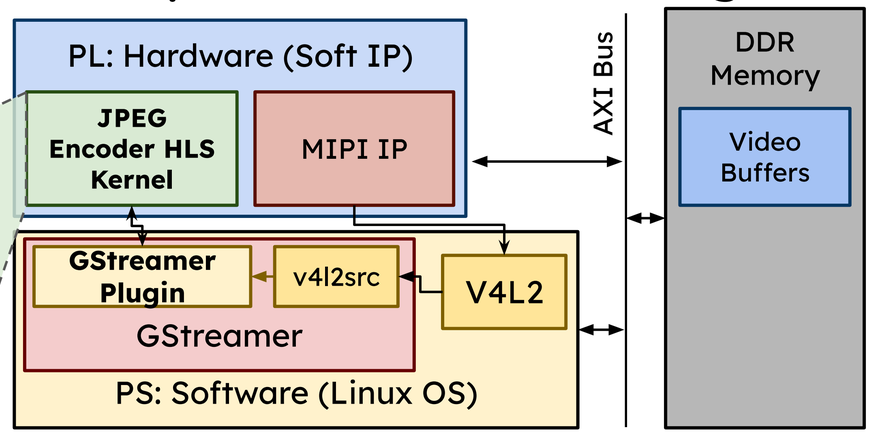
System Design



Contributions

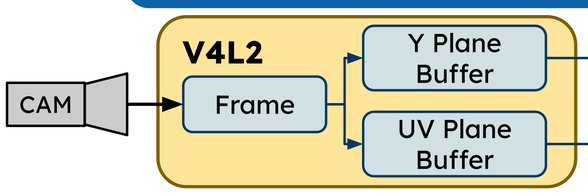
- **HW JPEG Encoder** implemented in *High-level Synthesis*.
- ***GStreamer* plugin** interfaces with encoder.
  - Allows for mix and matching of pipeline elements for arbitrary media processing.
  - Easy usage of DMA capabilities.

Top-level Hardware Design

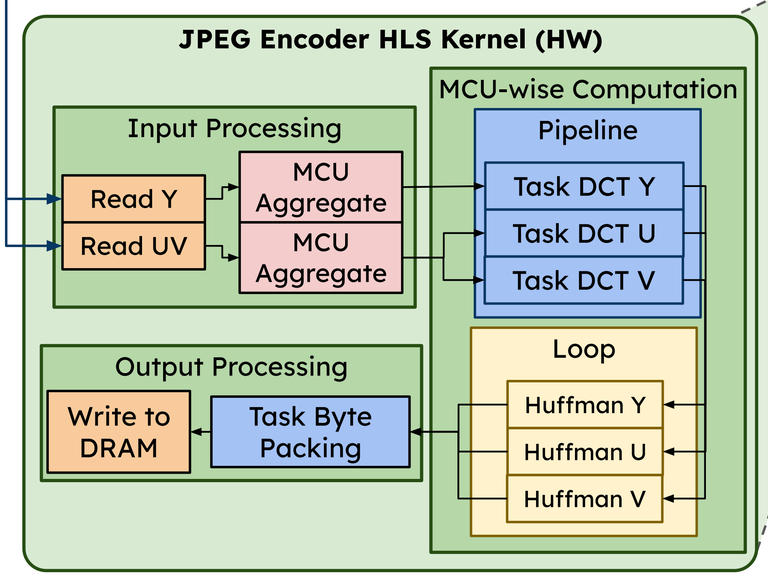


- Camera accessed through V4L2
  - Accesses must go through AXI bus

Hardware Architecture



- Send each plane to the encoder via AXI
  - Avoid memory copy in case of DMA

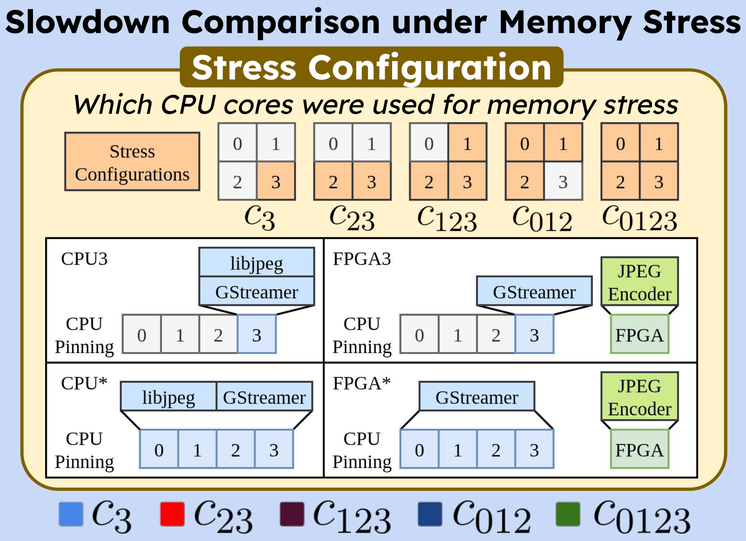


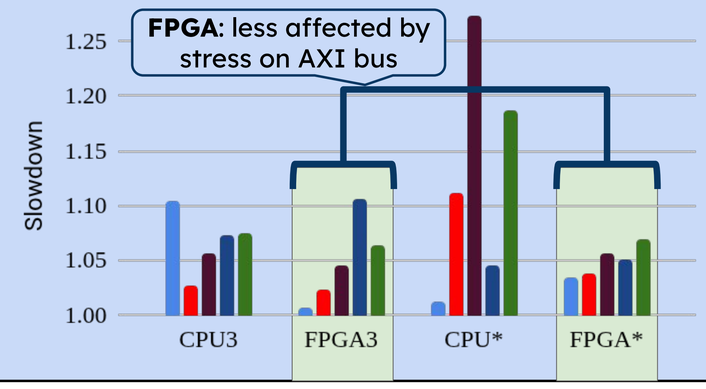
- Conventional JPEG algorithm design implemented C++ HLS
- Streaming design:
  - Data is packed into **MCU** streams at input processing
  - Each **MCU** in a stream goes through **DCT and RLE** pipelined, and **sequential Huffman**
  - MCUs are packed into the expected JPEG bitstream format at the end.

Experiments

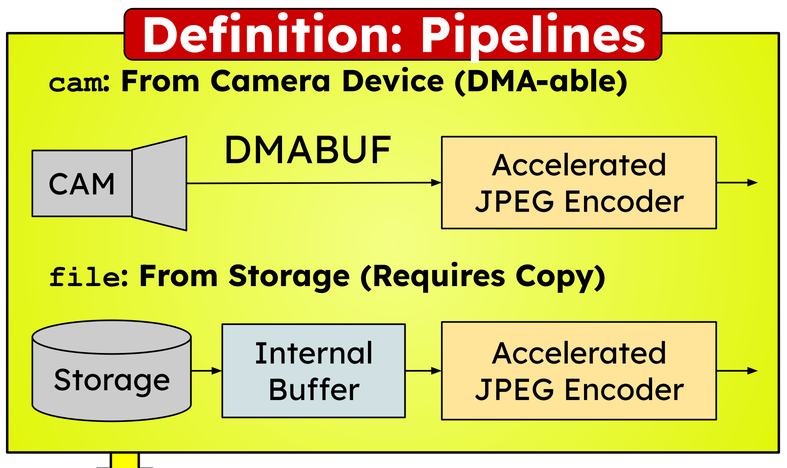
Memory Stress Testing

Slowdown Comparison under Memory Stress





Definition: Pipelines



Performance

FPGA vs. CPU (jpegenc)

		CPU	FPGA	FPGA/CPU
cam	FPS	2.6	**19.5**	**7.5**
	CPU %	93	**5**	**0.054**
file	FPS	**22.6**	21.7	.96
	CPU %	107	**14**	**0.13**

Resource Utilization

JPEG (our) vs. Vitis Codec Libraries

	Freq.	BRAM	URAM	DSP	FF	LUT
JPEG[1]	250MHz	**6**	**12**	**85**	**18150**	**13868**
Webp[2]	100MHz	81	46	834	71227	68906
Webp[2]	250MHz	229	10	414	92030	68755
JXL[2]	260MHz	584	122	644	270565	199537
PIK[2]	200MHz	614	473	2398	549987	449995

[1]Our work
[2]Vitis Codec Libraries

Webp Encoder[2]

- **High Resource Usage**
  - Requires lowering **target frequency** to 100Hz

JPEG Encoder[1]

- **7~25%** of Webp Resources
  - Can fit **four** JPEG encoders in place of a **single** Webp Encoder

Discussion and Conclusions

1. Suitability for Edge
   - Low resource usage
   - Realtime performance
2. Usability
   - GStreamer integration
     - Easy composition of media pipelines
3. Increased Predictability
   - Resilience against memory stress

Future Work

- Streamlined flow for GStreamer integration with arbitrary HLS kernels

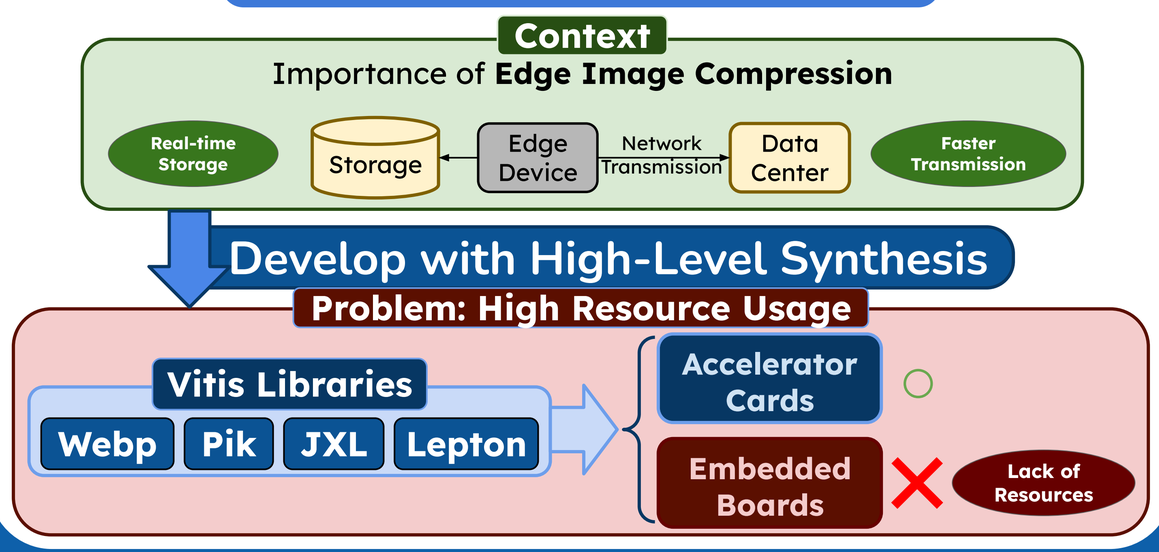
GStreamer-integrated HLS-based JPEG Encoder for Edge FPGA SoCs

Yuri Guimaraes Pereira Primo da Silva[1], Shinya Honda[1], Sugako Otani[12], Masato Edahiro[1], Abraham Monrroy Cano[3]

[1]Nagoya University, [2]Renesas Electronics, [3]Map IV Inc.

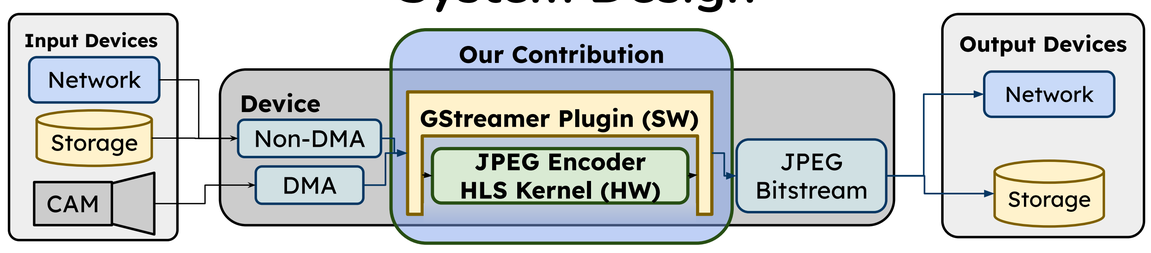
E-mail: yuri_gpps1@ertl.jp

Motivation



Approach

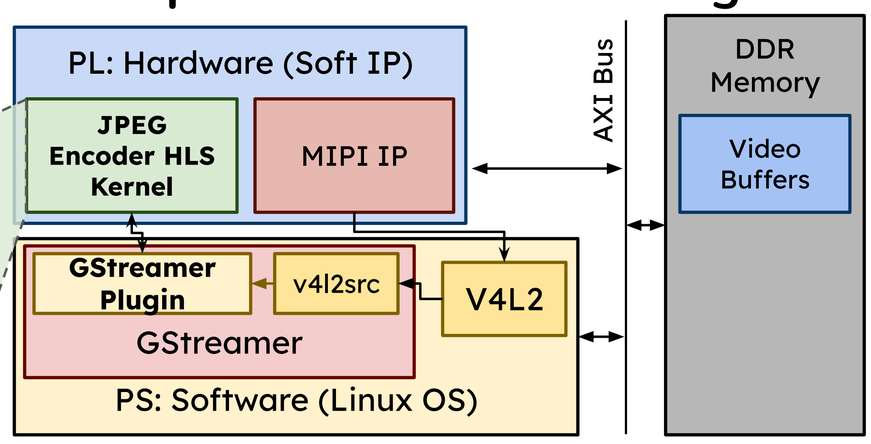
System Design



Contributions

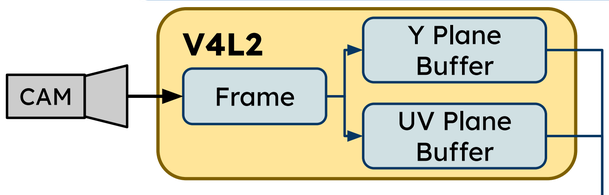
- **HW JPEG Encoder** implemented in *High-level Synthesis*.
- ***GStreamer* plugin** interfaces with encoder.
  - Allows for mix and matching of pipeline elements for arbitrary media processing.
  - Easy usage of DMA capabilities.

Top-level Hardware Design



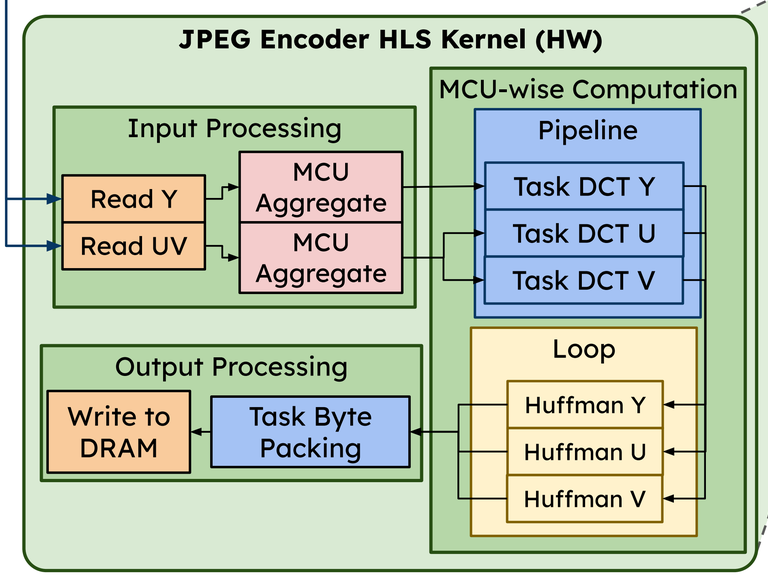
- Camera accessed through V4L2
  - Accesses must go through AXI bus

Hardware Architecture



- Send each plane to the encoder via AXI
  - Avoid memory copy in case of DMA

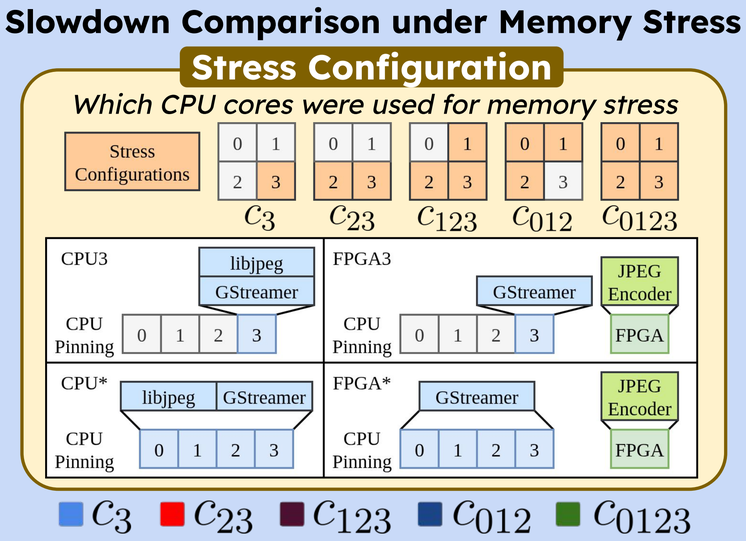
- Conventional JPEG algorithm design implemented C++ HLS
- Streaming design:
  - Data is packed into **MCU** streams at input processing
  - Each **MCU** in a stream goes through **DCT and RLE** pipelined, and **sequential Huffman**
  - MCUs are packed into the expected JPEG bitstream format at the end.

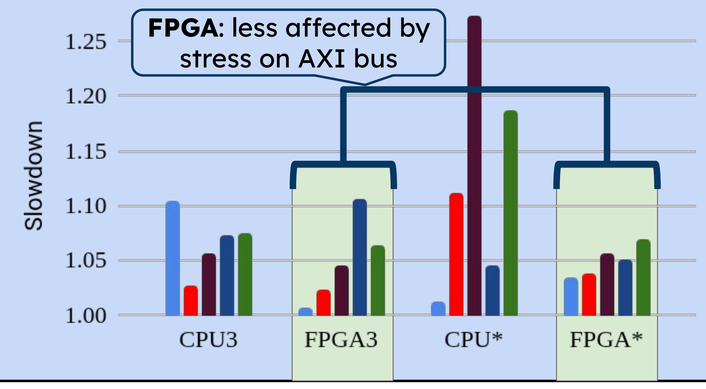


Experiments

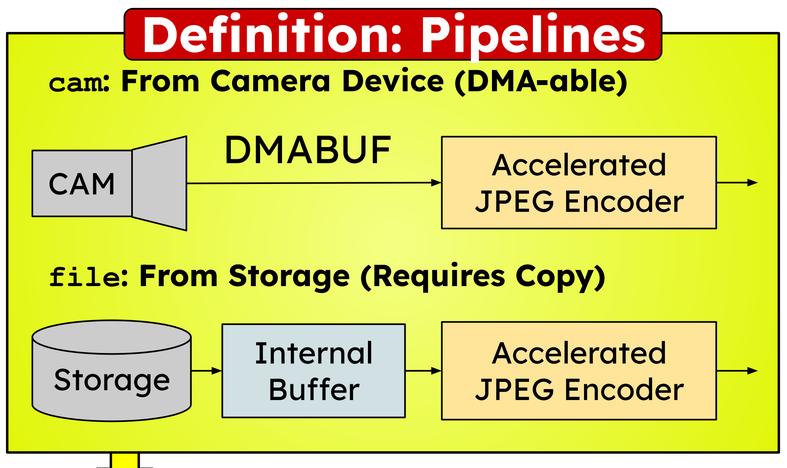
Memory Stress Testing

Slowdown Comparison under Memory Stress





Definition: Pipelines



Performance

FPGA vs. CPU (jpegenc)

		CPU	FPGA		FPGA/CPU	
		Single Core	1 CU	2 CU	1 CU	2 CU
cam	FPS	2.6	**19.5**	**30.0**	7.5	**15**
cam	CPU %	93	**5**	**10**	**0.054**	0.108
file	FPS	**22.6**	21.7	30.0	.96	**1.38**
file	CPU %	107	**14**	**28**	0.13	0.26

Resource Utilization

JPEG (our) vs. Vitis Codec Libraries

	Freq.	BRAM	URAM	DSP	FF	LUT
JPEG[1]	250MHz	**6**	**12**	**85**	**18150**	**13868**
Webp[2]	100MHz	81	46	834	71227	68906
Webp[2]	250MHz	229	10	414	92030	68755
JXL[2]	260MHz	584	122	644	270565	199537
PIK[2]	200MHz	614	473	2398	549987	449995

[1]Our work
[2]Vitis Codec Libraries

Webp Encoder[2]

- **High Resource Usage**
  - Requires lowering **target frequency** to 100Hz

JPEG Encoder[1]

- **7~25%** of Webp Resources
  - Can fit **four** JPEG encoders in place of a **single** Webp Encoder

d Conclusions

1. S
- L
- R

lity
amer integration
/ composition of
lia pipelines

3. Increased Predictability
- Resilience against memory stress

Future Work

- Streamlined flow for GStreamer integration with arbitrary HLS kernels

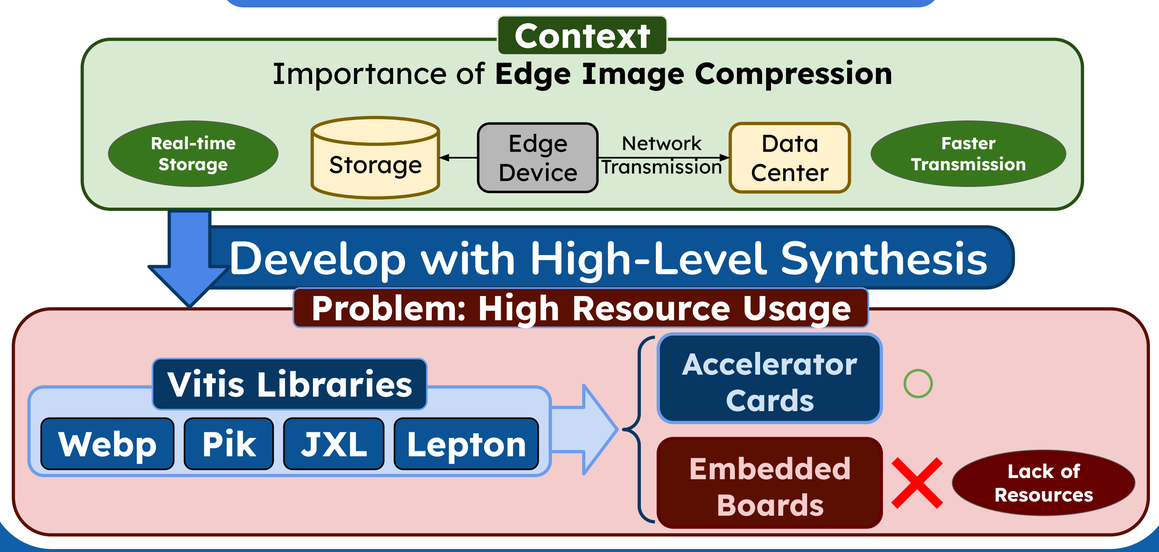
GStreamer-integrated HLS-based JPEG Encoder for Edge FPGA SoCs

Yuri Guimaraes Pereira Primo da Silva[1], Shinya Honda[1], Sugako Otani[12], Masato Edahiro[1], Abraham Monrroy Cano[3]

[1]Nagoya University, [2]Renesas Electronics, [3]Map IV Inc.

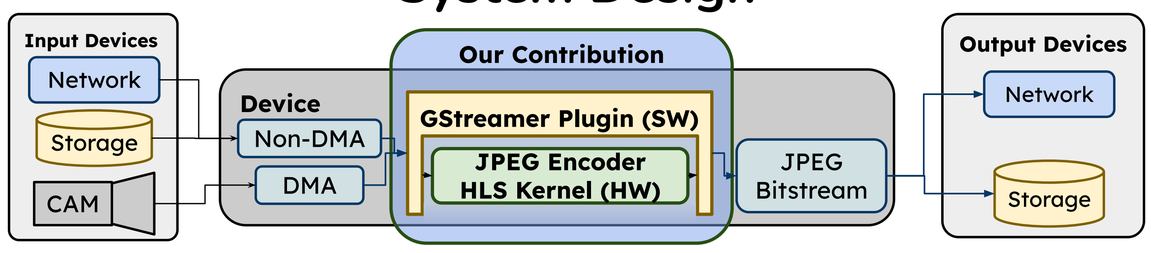
E-mail: yuri_gpps1@ertl.jp

Motivation



Approach

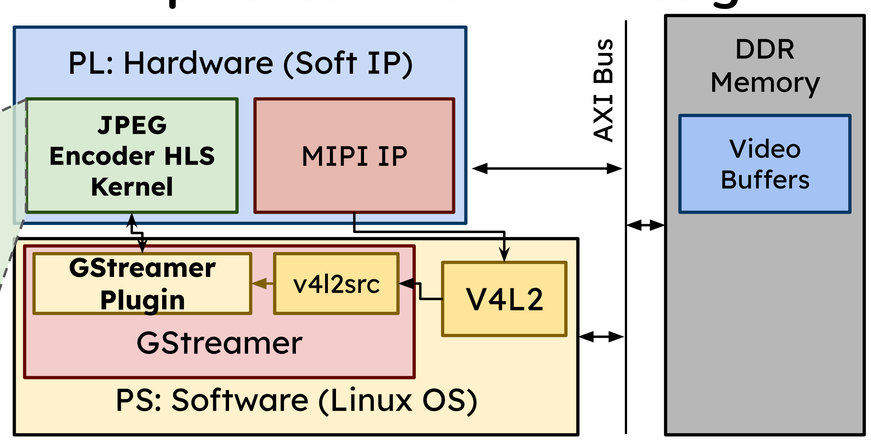
System Design



Contributions

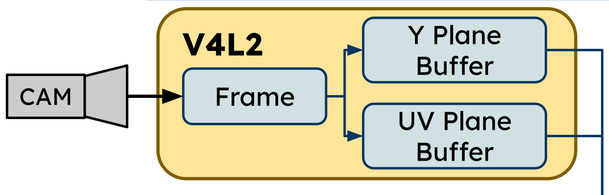
- **HW JPEG Encoder** implemented in *High-level Synthesis*.
- ***GStreamer* plugin** interfaces with encoder.
  - Allows for mix and matching of pipeline elements for arbitrary media processing.
  - Easy usage of DMA capabilities.

Top-level Hardware Design

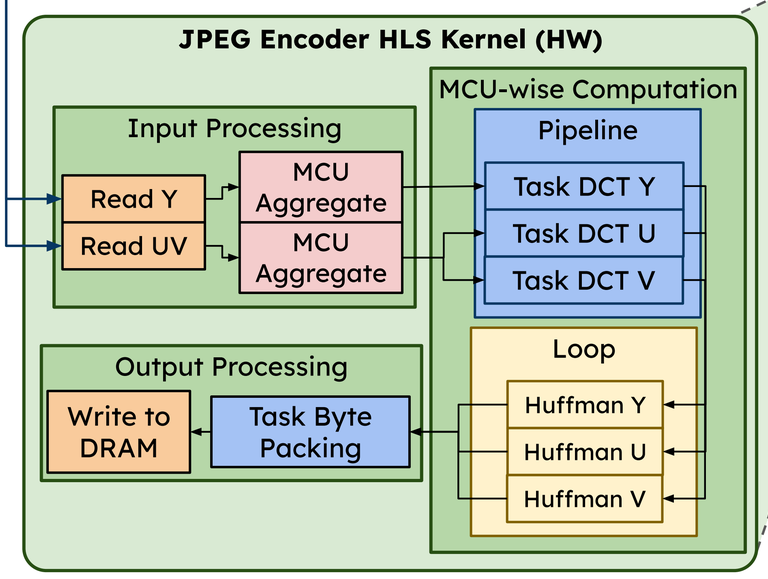


- Camera accessed through V4L2
  - Accesses must go through AXI bus

Hardware Architecture



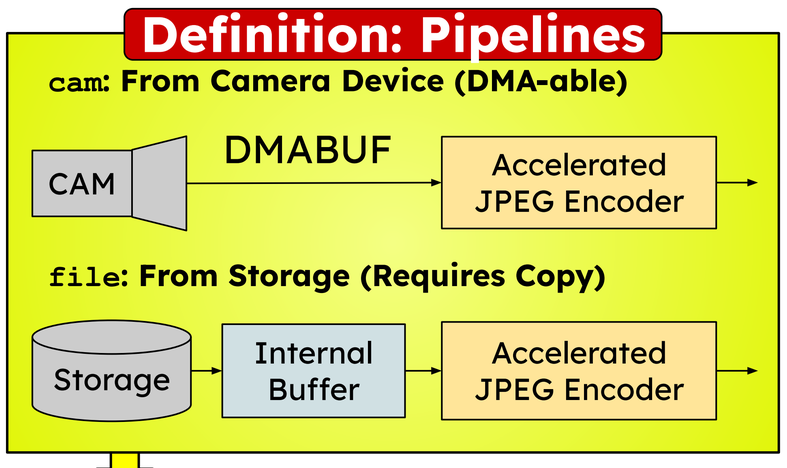
- Send each plane to the encoder via AXI
  - Avoid memory copy in case of DMA



- Conventional JPEG algorithm design implemented C++ HLS
- Streaming design:
  - Data is packed into **MCU** streams at input processing
  - Each **MCU** in a stream goes through **DCT and RLE** pipelined, and **sequential Huffman**
  - MCUs are packed into the expected JPEG bitstream format at the end.

Experiments

Definition: Pipelines



Performance

FPGA vs. CPU (jpegenc)

		CPU	FPGA (Our)	FPGA/CPU
cam	FPS	2.6	**19.5**	7.5
	CPU %	93	**5**	0.054
file	FPS	**22.6**	21.7	.96
	CPU %	107	**14**	0.13

Resource Utilization

JPEG (our) vs. Vitis Codec Libraries

	Freq.	BRAM	URAM	DSP	FF	LUT
JPEG[1] (Our)	250MHz	**6**	**12**	**85**	**18150**	**13868**
Webp[2]	100MHz	81	46	834	71227	68906
Webp[2]	250MHz	229	10	414	92030	68755
JXL[2]	260MHz	584	122	644	270565	199537
PIK[2]	200MHz	614	473	2398	549987	449995

[1]Our work
[2]Vitis Codec Libraries

Webp Encoder[2]

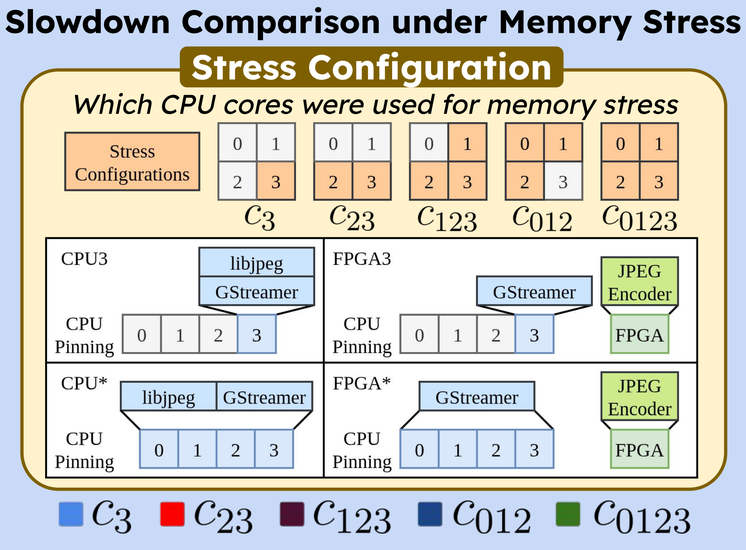
- **High Resource Usage**
  - Requires lowering **target frequency** to 100Hz

JPEG Encoder[1]

- **7~25%** of Webp Resources
  - Can fit **four** JPEG encoders in place of a **single** Webp Encoder

Memory Stress Testing

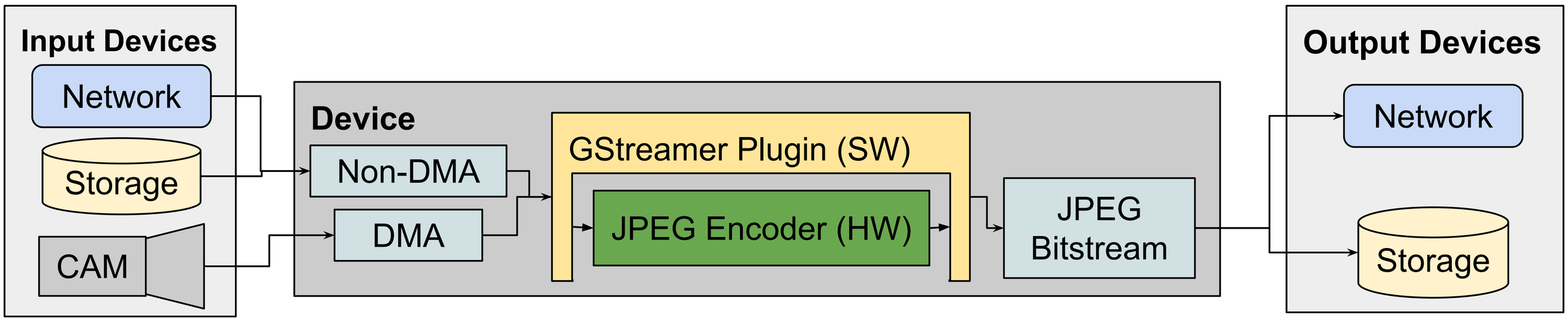
Slowdown Comparison under Memory Stress

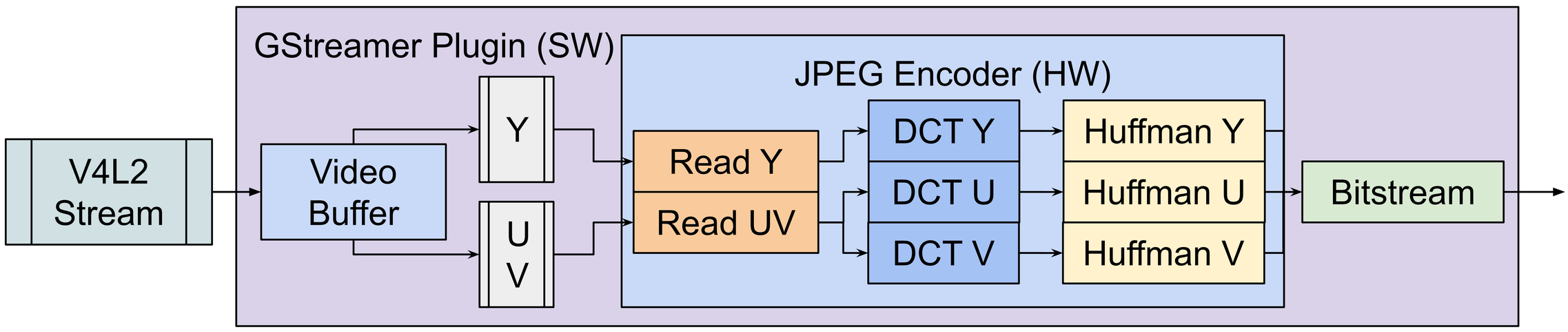


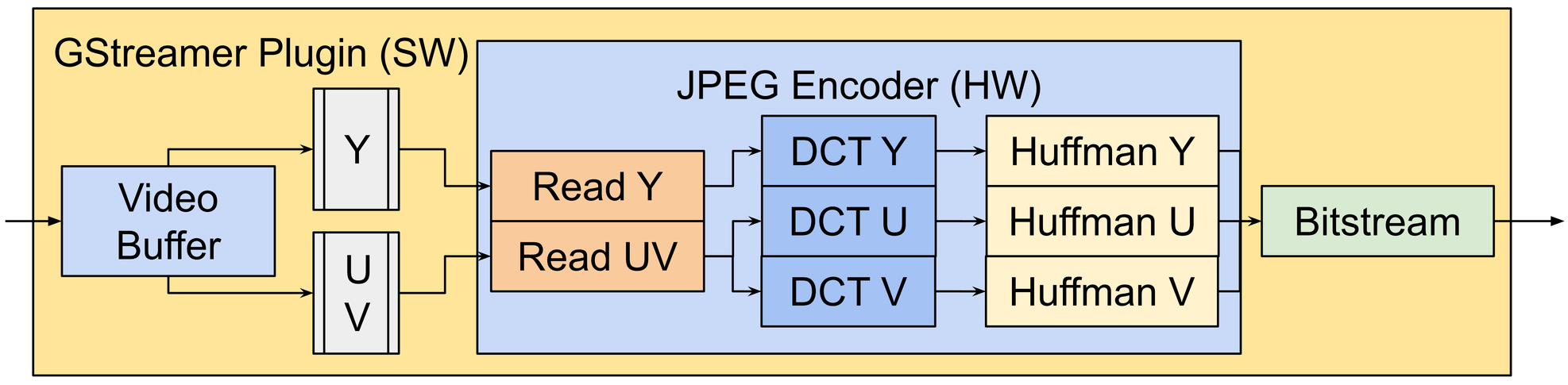
Discussion and Conclusions

1. Suitability for Edge
   - Low resource usage
   - Realtime performance
2. Usability
   - GStreamer integration
     - Easy composition of media pipelines
3. Increased Predictability
   - Resilience against memory stress

Download this Poster!

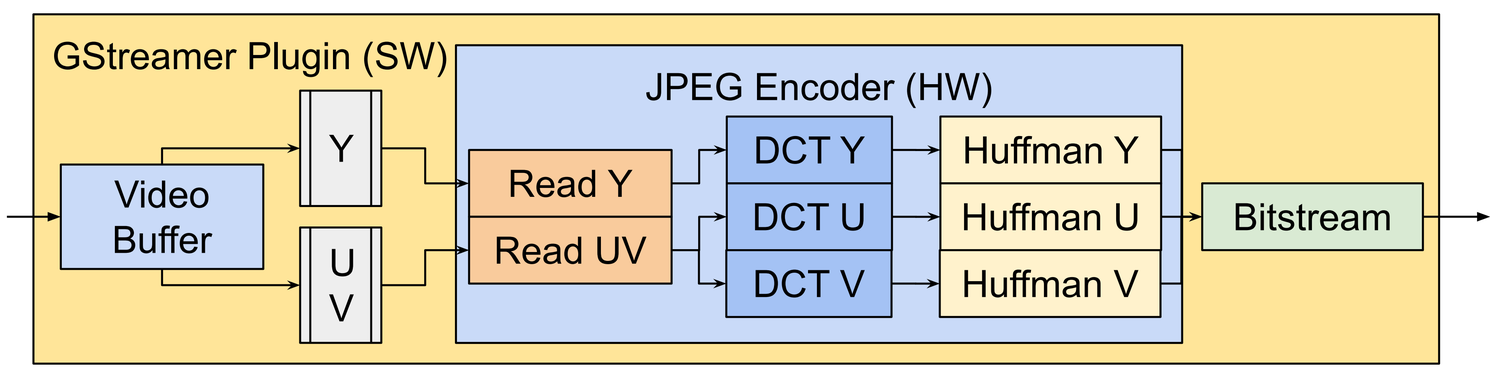

Input Devices
Network
Storage
CAM
Device
Non-DMA
DMA
GStreamer Plugin (SW)
JPEG Encoder (HW)
JPEG
Bitstream
Output Devices
Network
Storage


V4L2
Stream
GStreamer Plugin (SW)
Video
Buffer
Y
U
V
JPEG Encoder (HW)
Read Y
Read UV
DCT Y
DCT U
DCT V
Huffman Y
Huffman U
Huffman V
Bitstream

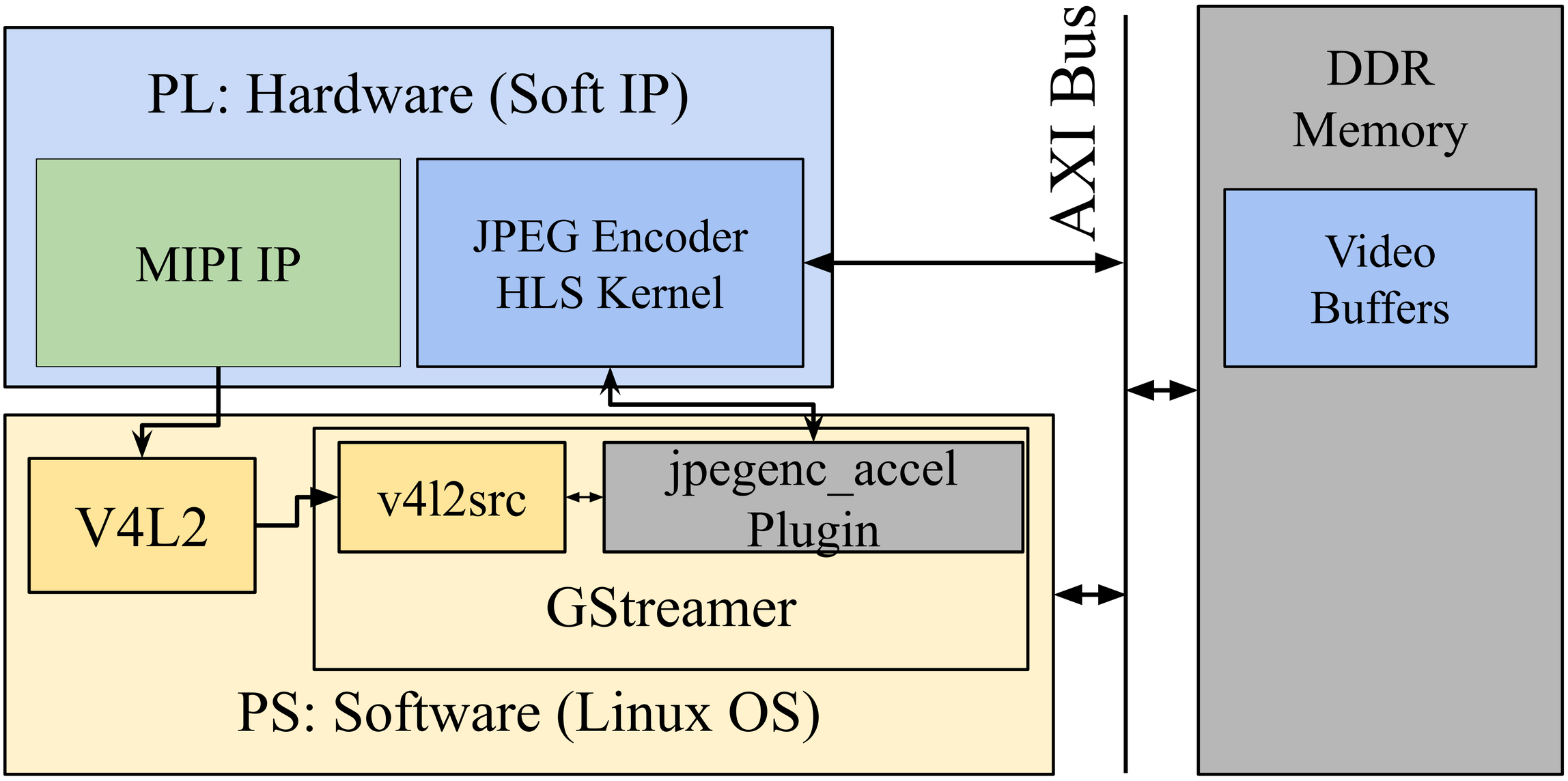

GStreamer Plugin (SW)
Video
Buffer
Y
U
V
JPEG Encoder (HW)
Read Y
Read UV
DCT Y
DCT U
DCT V
Huffman Y
Huffman U
Huffman V
Bitstream

Overall Architecture

The design consists of a HW component integrated with a *GStreamer* plugin.



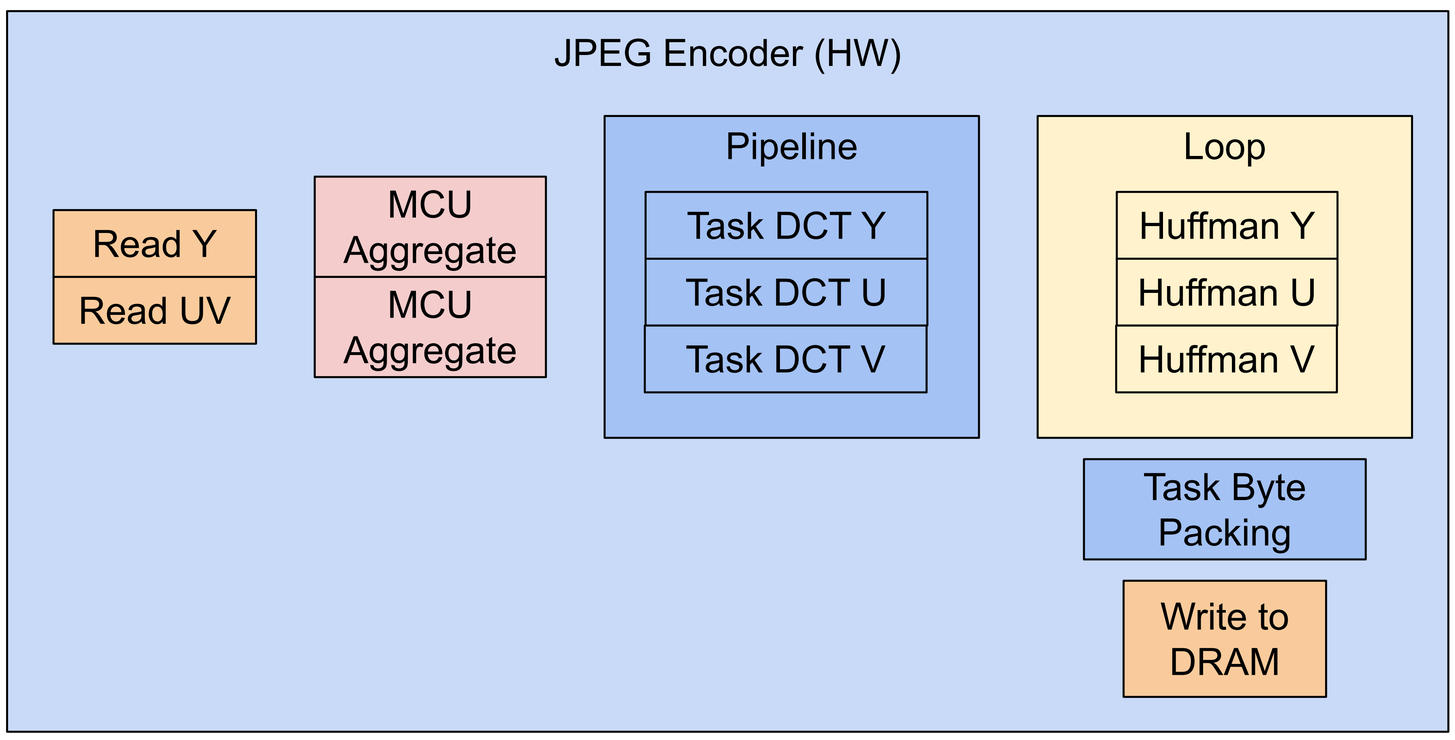
GStreamer allows for easy and quick composition of media pipelines, making the design easily testable in a wide range of scenarios.

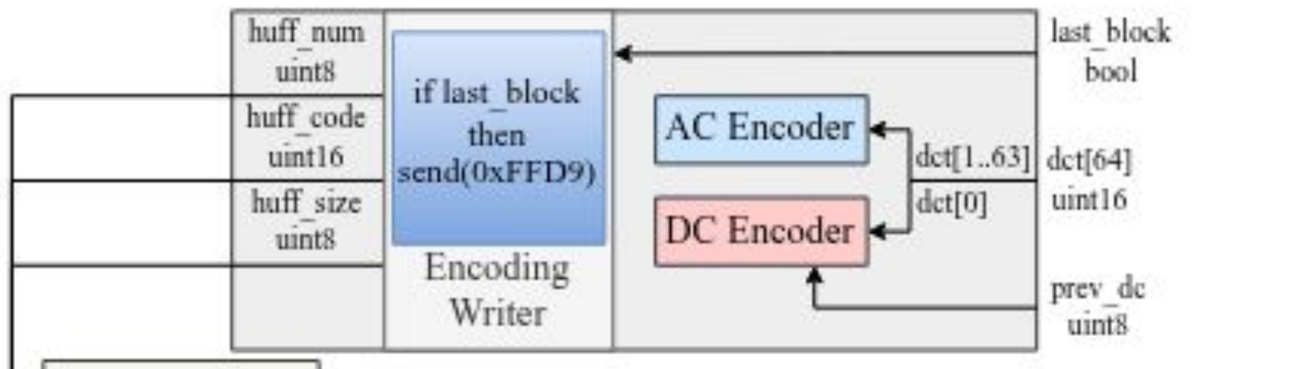


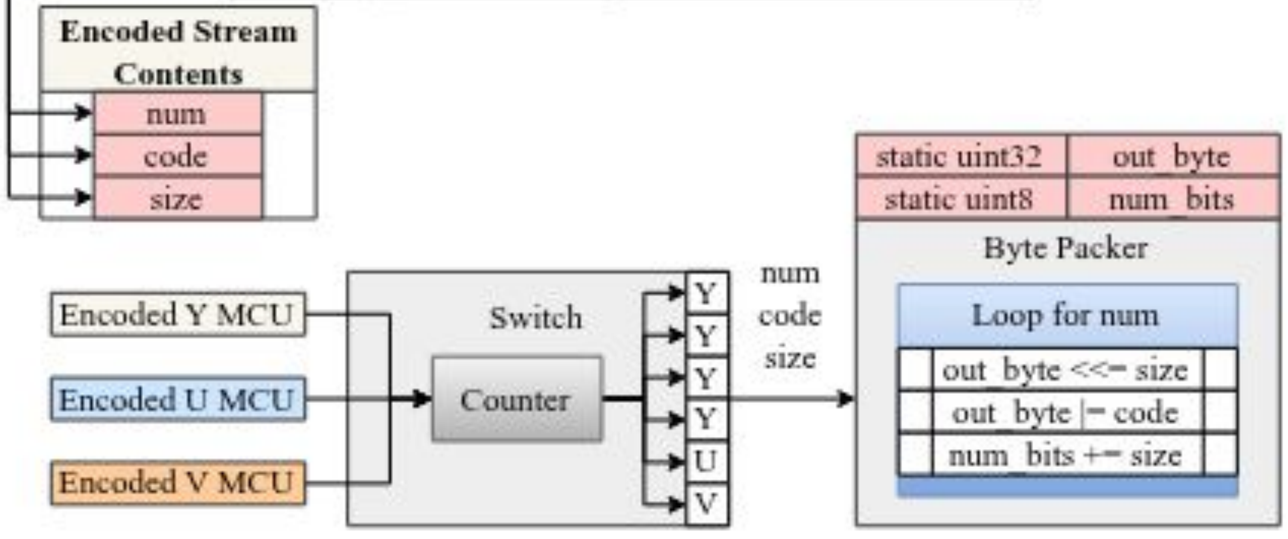
The camera-based pipeline consists of an AMD MIPI IP accessed through the v4l2src GStreamer plugin node. Separate buffer planes can be accessed directly and processed as individual channels on the JPEG IP, improving throughput.

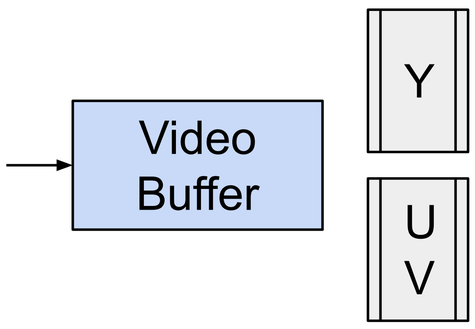
Hardware Components

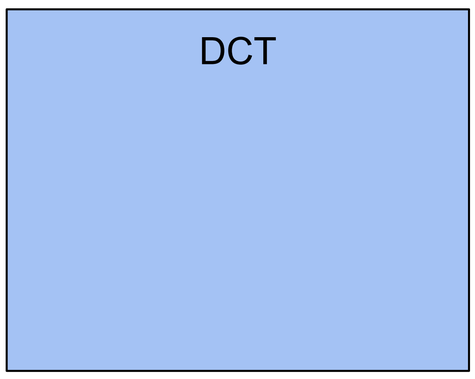
Y and UV channels are processed separately in order to aggregate MCUs into streams.
Afterwards, each MCU goes through DCT, RLE, Huffman encoding.
Finally, all MCU channels are merged together acording to the YYYYUV order for YUV420.

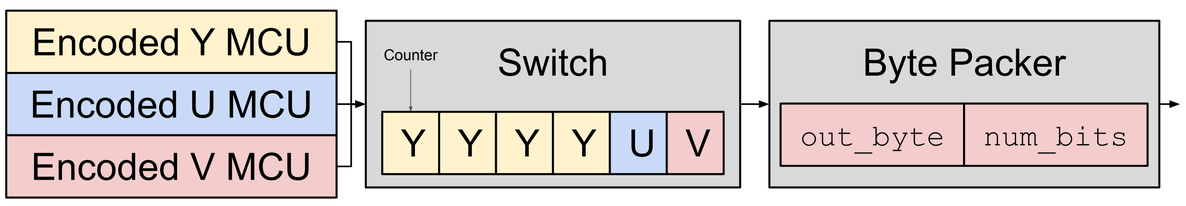


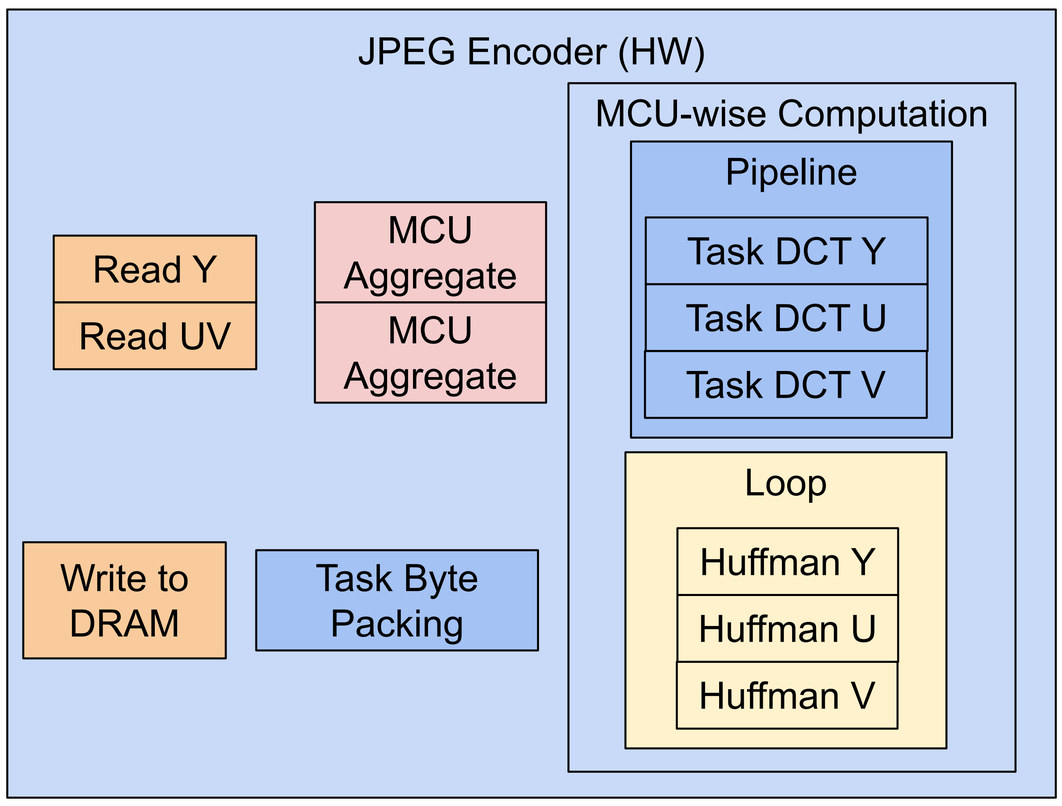




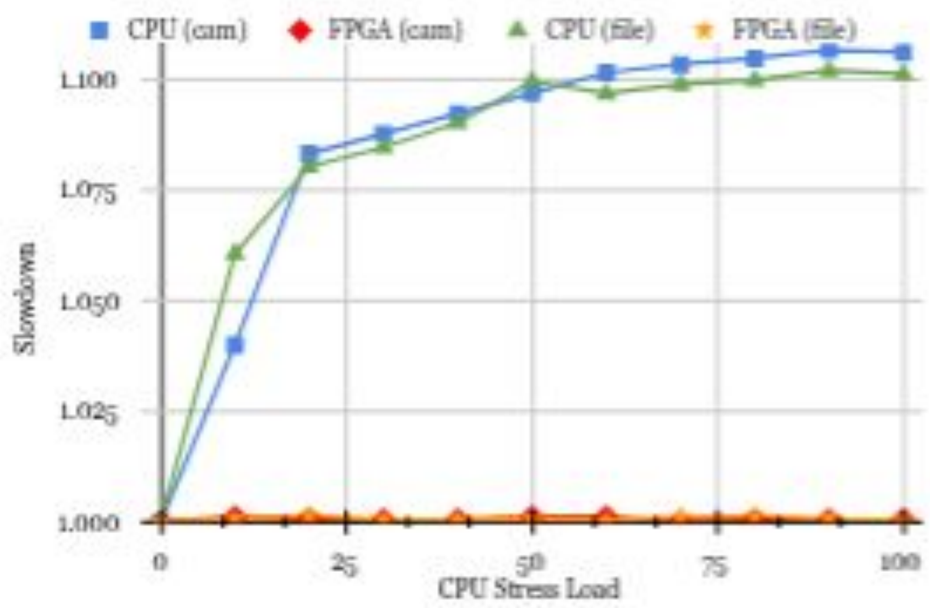


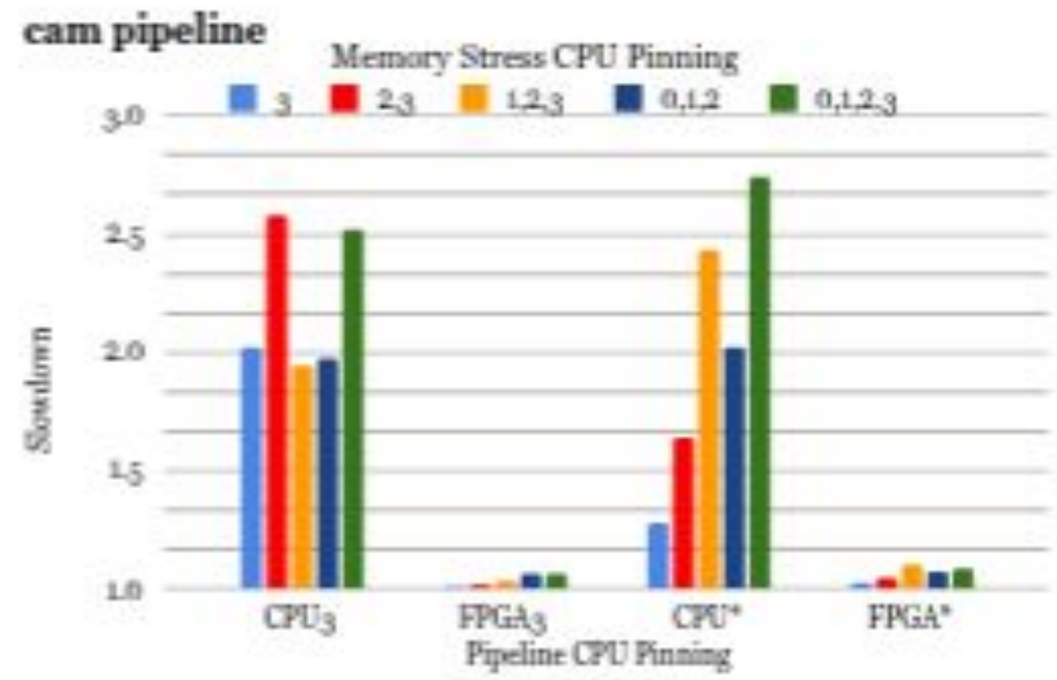


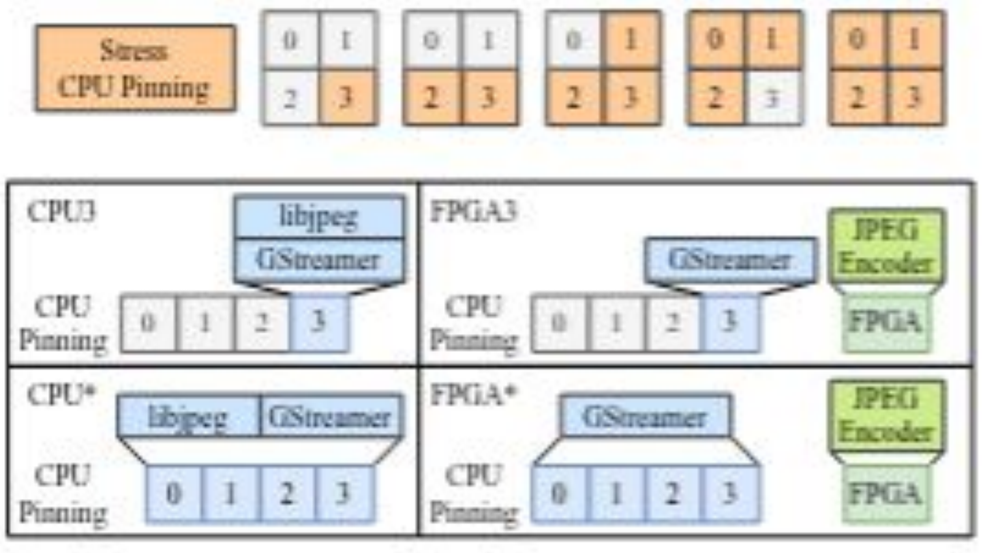




Experiments







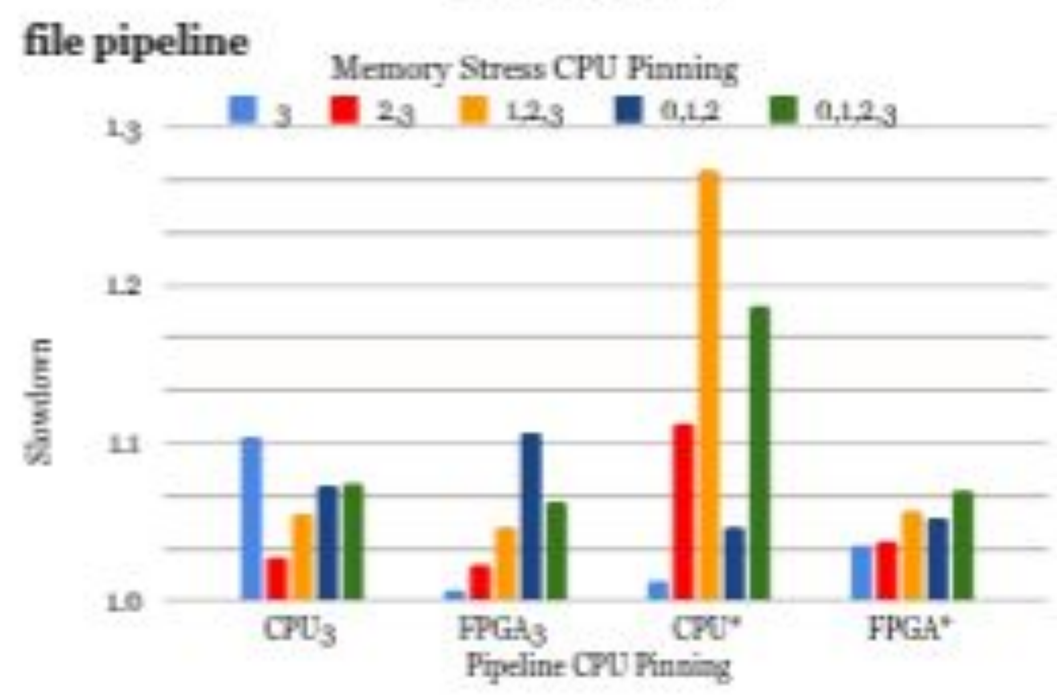


Fig. 6. **Memory Stress Configuration**: Four configurations for single-core or multi-core affinity of the pipeline (CPU3, CPU*, FPGA3, FPGA*), as well as five CPU affinity configurations for the memory stress.

TABLE II
RESOURCE UTILIZATION COMPARISON OF JPEG AND WEBP ENCODERS ON KV260

	JPEG (our work)	WebP [3]	Total Available
Frequency	250 MHz	100 MHz	
Throughput	4.49 MPps	1.88 MPps	
LUT	**11,513**	62,296	92,832
LUTAsMem	**2,355**	6,610	55,336
REG	**18,150**	71,227	191,197
BRAM	**6**	81	106
URAM	**12**	46	56
DSP	**85**	834	1248

TABLE III
PERFORMANCE EVALUATION OF THE JPEG ENCODER

		CPU	FPGA (Our)
`cam`	Throughput (fps)	2.6	**19.5**
	CPU Usage (%)	93	**5**
`file`	Throughput (fps)	**22.6**	21.7
	CPU Usage (%)	107	**14**
	Compression Ratio	3.202%	**3.219%**
	PSNR (dB)	**27.72**	27.45