FD+FR with VART API

Seungjo Yea
8/13/2024

AMD
together we advance_

Agenda

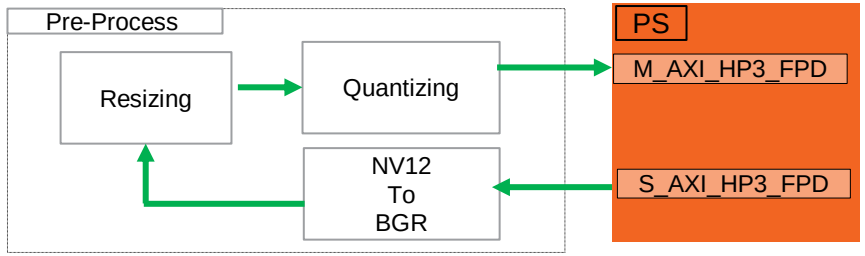
1. Block Diagram
    1. [HW]HW Platform Block Diagram
    2. [SW]Application Flow Block Diagram
    3. Improvement plan

Appendix

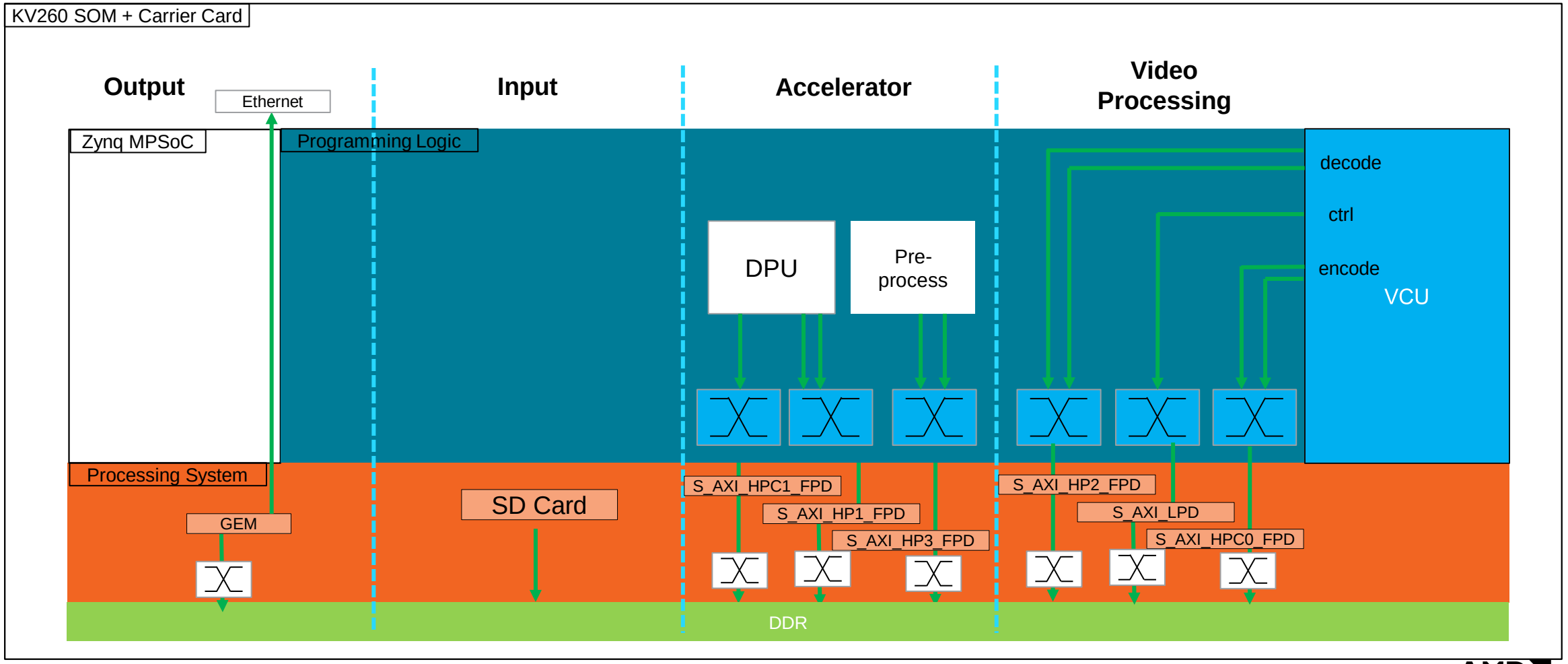
2. How I measured the latency
3. Profile result

1. Block Diagram

- HW Platform
  - Next goal is using HW VCU and HW Pre-processor



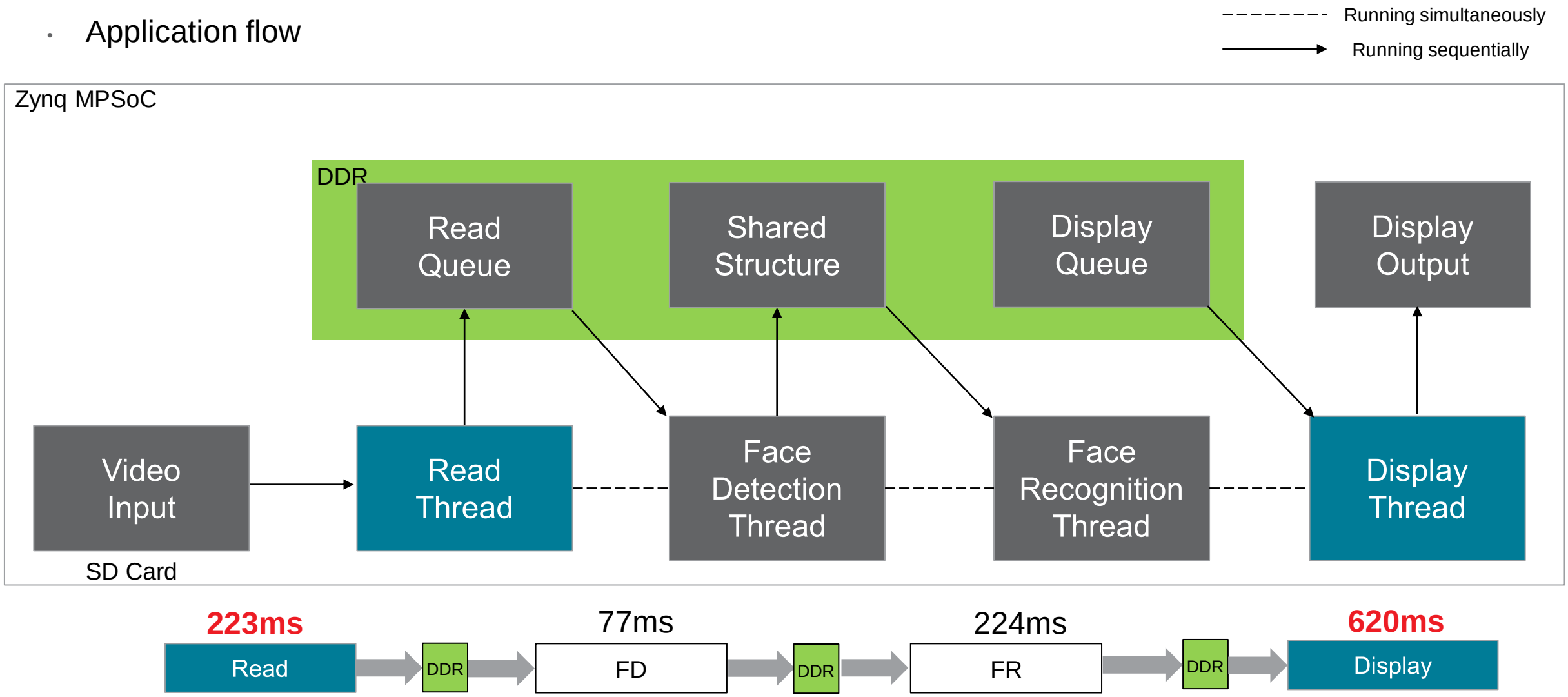
PL HW Block
PL Soft IP
Hard IP



*GEM : Gigabit ethernet interface

1. Block Diagram

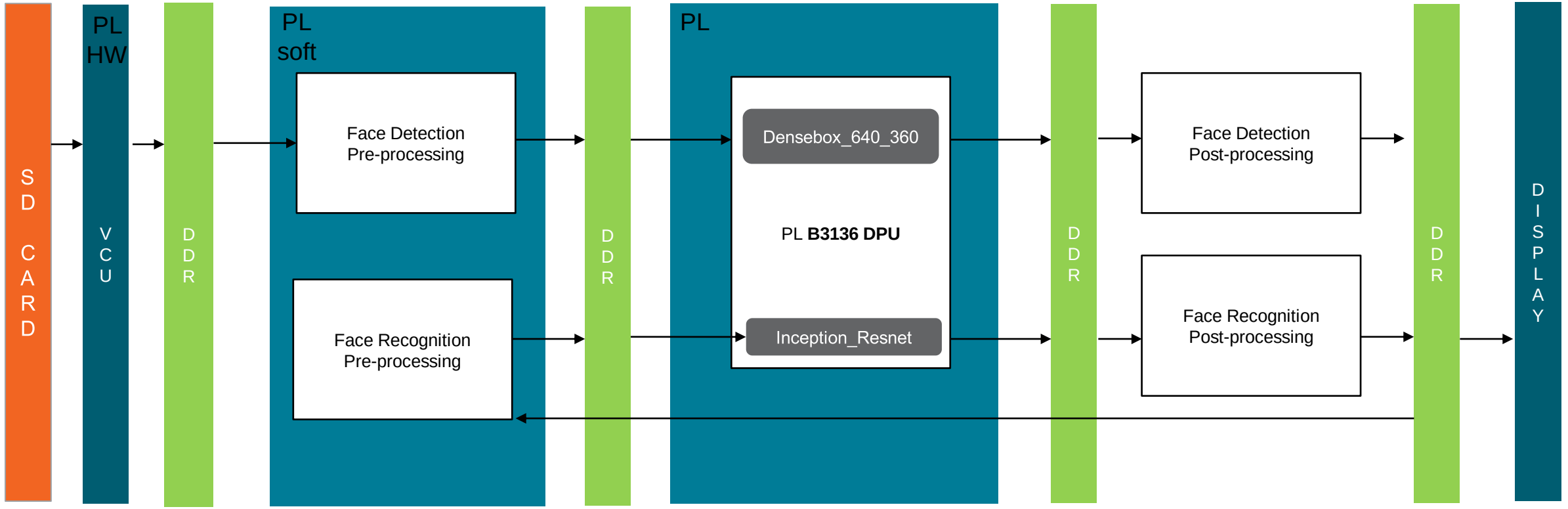
- Application flow



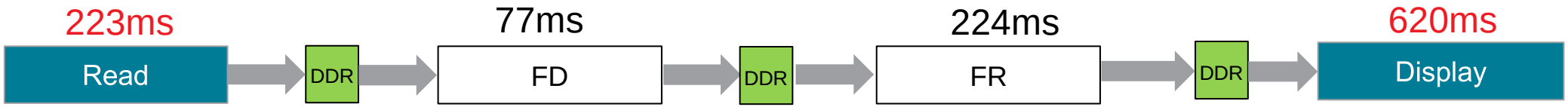
Total Latency: 223 + 77 + 224 + 620 = 1144ms

1. Block Diagram

1. Improvement plan – Add VCU HW, Pre-process HW



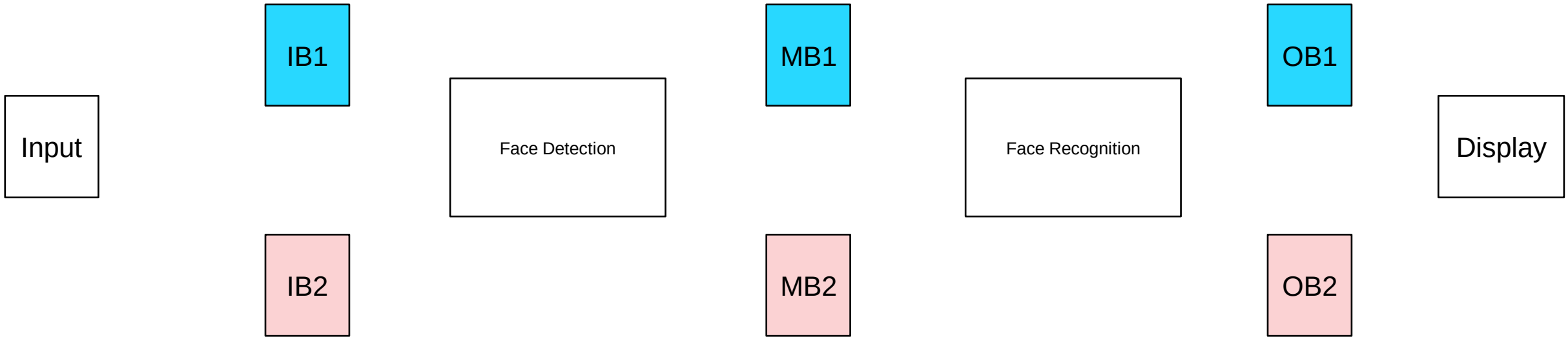
Read FD & FR Display



Total Latency: 223 + 77 + 224 + 620 = 1144ms

1. Block Diagram

Improvement plan
: Using double buffer to reduce read, write latency.



Thread							
T1	Input	IB1	FD	MB1	FR	OB1	Display
T2	Input	IB1	FD	MB2	FR	OB2	Display
T3	Input	IB2	FD	MB1	FR	OB1	Display
T4	Input	IB2	FD	MB2	FR	OB2	Display

Requirements

- **Pure VART APIs? No VVAS?**

: Yes.

- **Have you run profiling?**

: Yes, I have the data in Appendix

- **Which video format does the VAI Library use? Does the VAI Library use OpenCV?**

: Not specified. Reference design has Format convert HW that convert NV12 to BGR.

- **Is using MP4 or AVI a user requirement?**

: What if user requirement is not MP4 or AVI?

- **Waitkey(25) rather than (1)?**

: Need to reduce the latency significantly.. Need to improve waitkey.

Appendix

2. How I measured the latency

- std::chrono::high_resolution_clock

```
    auto preprocess_start = std::chrono::high_resolution_clock::now();
    setImageBGR({resized}, runner, imageInputs, mean, scale);
    auto preprocess_end = std::chrono::high_resolution_clock::now();
    auto preprocess_duration = std::chrono::duration_cast<std::chrono::microseconds>(preprocess_end -
         preprocess_start);

    std::cout << "  Postprocess: " << postprocess_duration.count() << " µs" << std::endl;
```

3. Profile result

(CPU : Pre-processing, Post-procesing, DPU : Inference)
Container : MPEG-4(Base Media): 498 KiB, 6s0ms, 1 video stream: AVC
First video stream : 673 kb/s, 1280*720(16:9) at 60.000FPS, AVC

Metric	Average Latency(ms)	Comments
Read	223.28	**Frame read**
Face Detection	77.44	**Detect face on the frame**
Face Recognition	224.18	**Compare with embedding data**
Display	619.55	**Display frame**

3. Profile result : Read

(CPU : Pre-processing, Post-procesing, DPU : Inference)
Container : MPEG-4(Base Media): 498 KiB, 6s0ms, 1 video stream: AVC
First video stream : 673 kb/s, 1280*720(16:9) at 60.000FPS, AVC

Metric	Average Latency(ms)	Comments
Read	223	Video Read
Others	0	
Total latency	223.28	

3. Profile result : Display

(CPU : Pre-processing, Post-procesing, DPU : Inference)
Container : MPEG-4(Base Media): 498 KiB, 6s0ms, 1 video stream: AVC
First video stream : 673 kb/s, 1280*720(16:9) at 60.000FPS, AVC

Metric	Average Latency(ms)	Comments
Frame processing	0.15	
ImShow	2.88	
WaitKey	616.52	waitKey(1)
Total latency	619.55	

3. Profile result : Face Detection

(CPU : Pre-processing, Post-procesing, DPU : Inference)
Container : MPEG-4(Base Media): 498 KiB, 6s0ms, 1 video stream: AVC
First video stream : 673 kb/s, 1280*720(16:9) at 60.000FPS, AVC

Metric	Average Latency(ms)	Comments
Queue	0.04	
Resize	3.67	
Preprocess	27.81	setInputBGR, Quantize
DPU execution	4.09	
Postprocess	40.57	De-Quantize, FilterBox, NMS
Store results	1.56	
Total latency	77.44	

3. Profile result : Face Recognition

(CPU : Pre-processing, Post-procesing, DPU : Inference)
Container : MPEG-4(Base Media): 498 KiB, 6s0ms, 1 video stream: AVC
First video stream : 673 kb/s, 1280*720(16:9) at 60.000FPS, AVC

Metric	Average Latency(ms)	Comments
Crop faces	0.80	
Preprocessing	3.09	Resize, Quantizing
DPU execution	10.09	
Postprocessing	1.23	De-Quantizing
Drawing	0.20	
Wait for result	208.33	
Draw bounding box and label	14.89	
Add to display queue	0.01	
Total latency	224.18	


AMD