embedded VISION SUMMIT

Targeting Computer Vision Algorithms to Embedded Hardware

Mario Bergeron, Avnet Electronics Marketing

October 2, 2013

AVNET electronics marketing

Accelerating Your Success™

Company Introduction - Avnet, Inc.

- **Who We Are:**
  - We are one of the world's largest global distributors of electronic components, computer products and embedded technology serving customers in more than 80 countries
- **What We Do**:
  - We connect the world's leading technology companies with more than 100,000 customers by providing cost-effective, value-added services and solutions
- **Financial Scope:**
  - For the fiscal year ended June 30, 2012, we generated revenue of $25.7 billion

Scope and Motivation

- Embedded Vision is growing at a tremendous pace
- Recent technology advancements make it feasible to deploy computer vision into embedded devices
  - Innovative computer vision algorithms and toolboxes
  - New hardware architectures optimized for vision
- What are the implications of turning a computer vision prototype into a product, on embedded hardware ?
- What does the engineer need to know in order to choose the best embedded vision platform for their needs ?

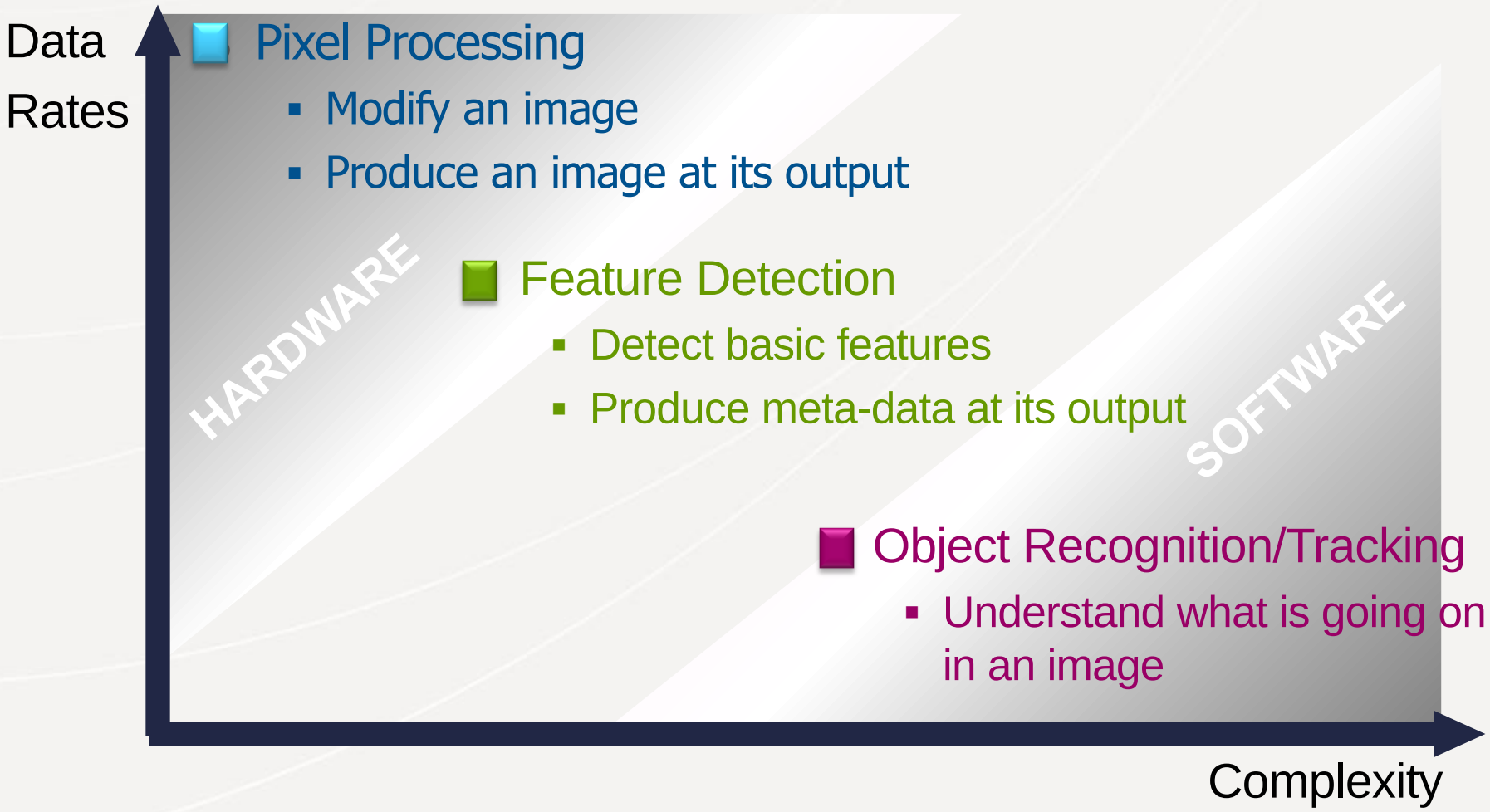
Overview

- Introduction
  - Implementing Embedded Vision on Embedded Hardware
- Pixel Processing Example
  - Define system requirements
  - Analyze challenges for an embedded implementation
  - Optimize the architecture to improve efficiency
- Use Cases

Introduction

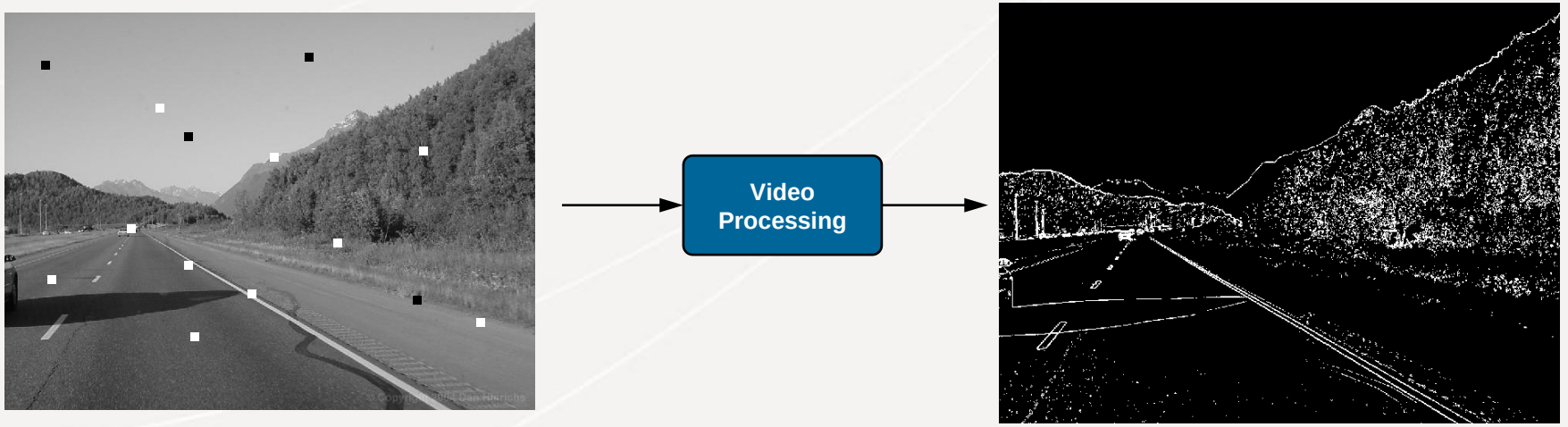
- Implementing Computer Vision on Embedded Hardware
  - Computer vision toolbox
    - Proprietary, Commercial, Open-Source
    - Goal : algorithm exploration
  - Embedded hardware platform
    - General purpose, specialized hardware
    - Goal : real-time execution, low cost/power

Different types of image/video processing algorithms



Pixel Processing - A simple example

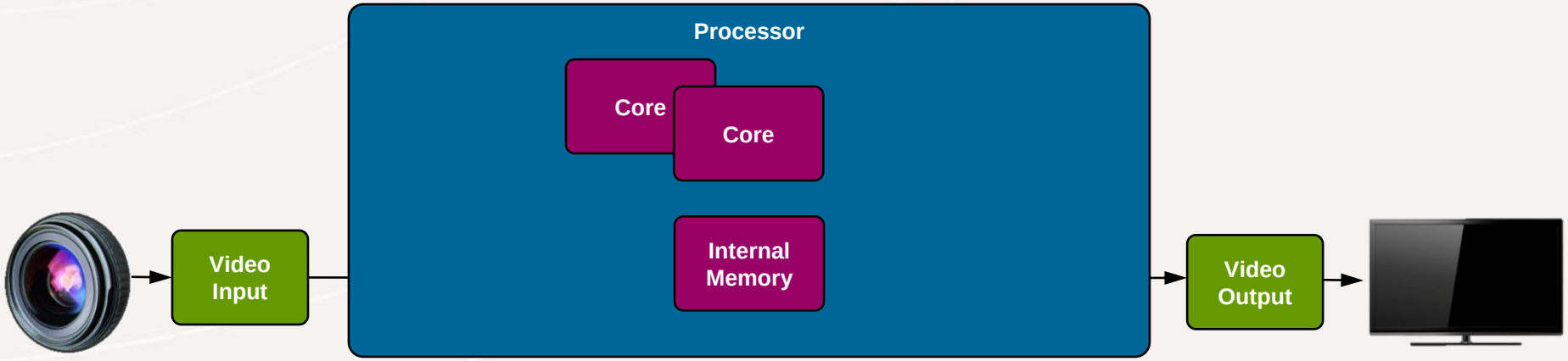
- cv::Canny
  - cv::GaussianBlur(gray, blurred, …)
  - cv::Sobel(blurred, grad_x, …)
  - cv::Sobel(blurred, grad_y, …)
  - cv::cartToPolar(grad_x, grad_y, magnitude, angle, …)
  - etc…



- Most Compute-Intensive functions
  - Gaussian Filter, Sobel Filters => 2D Filter

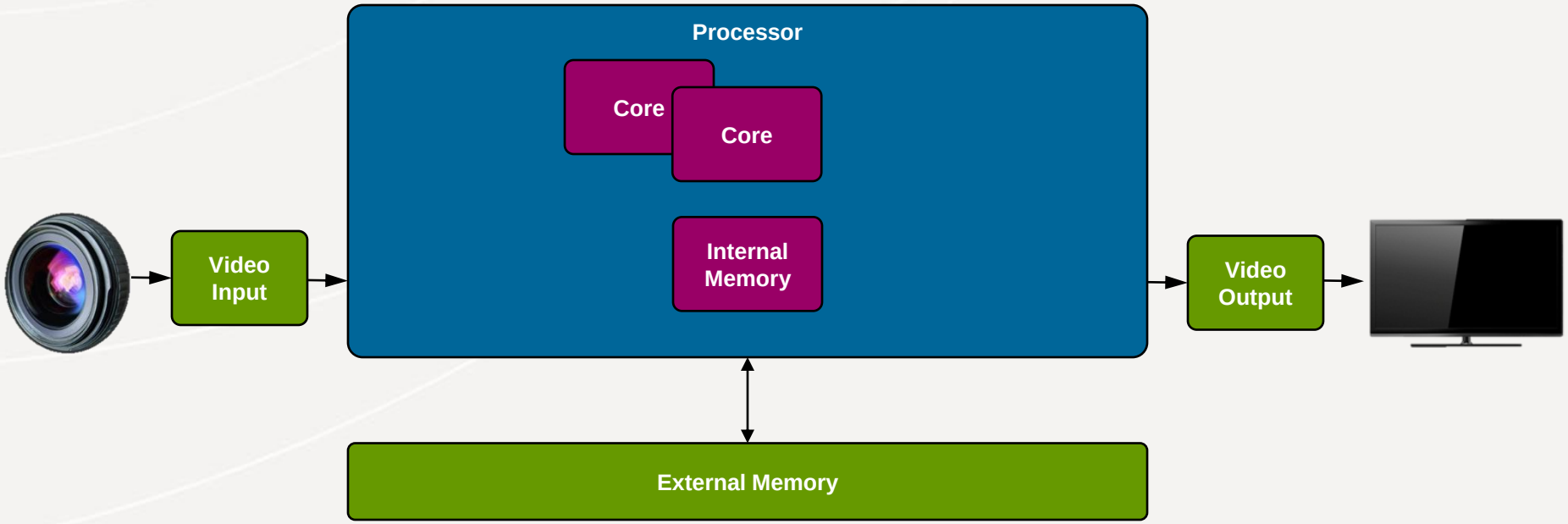
Pixel Processing System

- Let's Build a Simple Example
  - Video Input = HD image sensor (1080P @ 60FPS)
  - Video Output = HDMI interface (1080P @ 60FPS)
  - Video Processing = 2D Filter



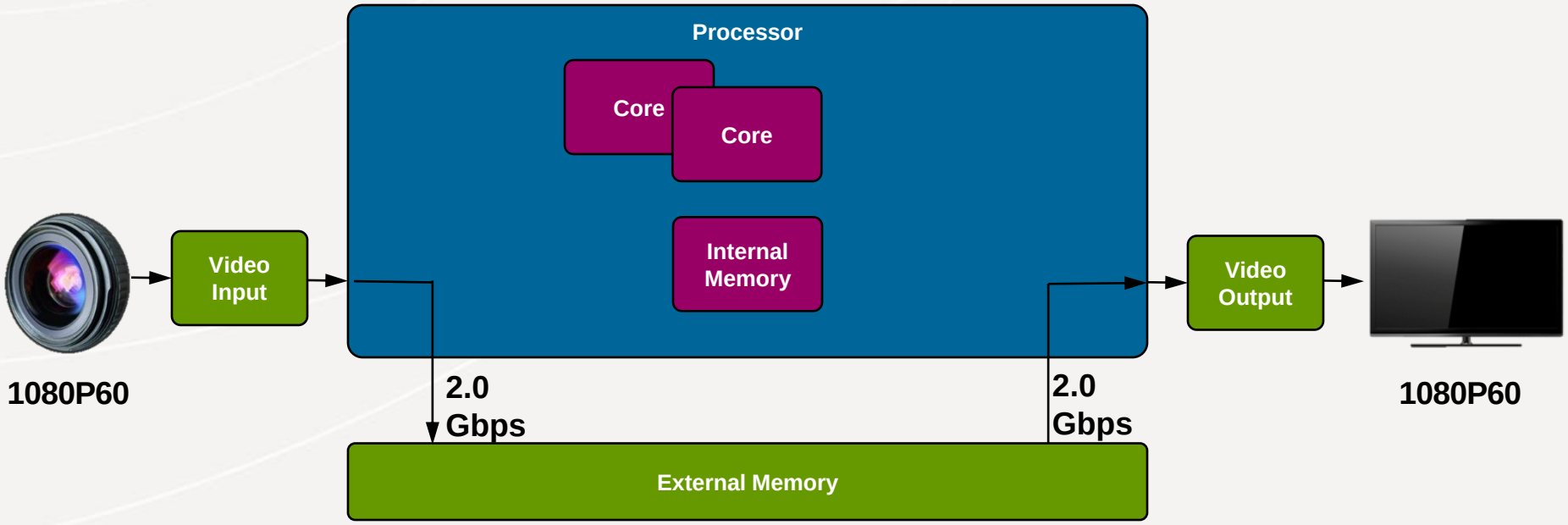
Video Storage Requirements

- 1080P Resolution
  - Frame Size = (1920 x 1080) pixels * 16 bits/pixel = 33 Mbits
- Challenge => Internal memory is typically 1-10 Mbits
- External memory required for HD video



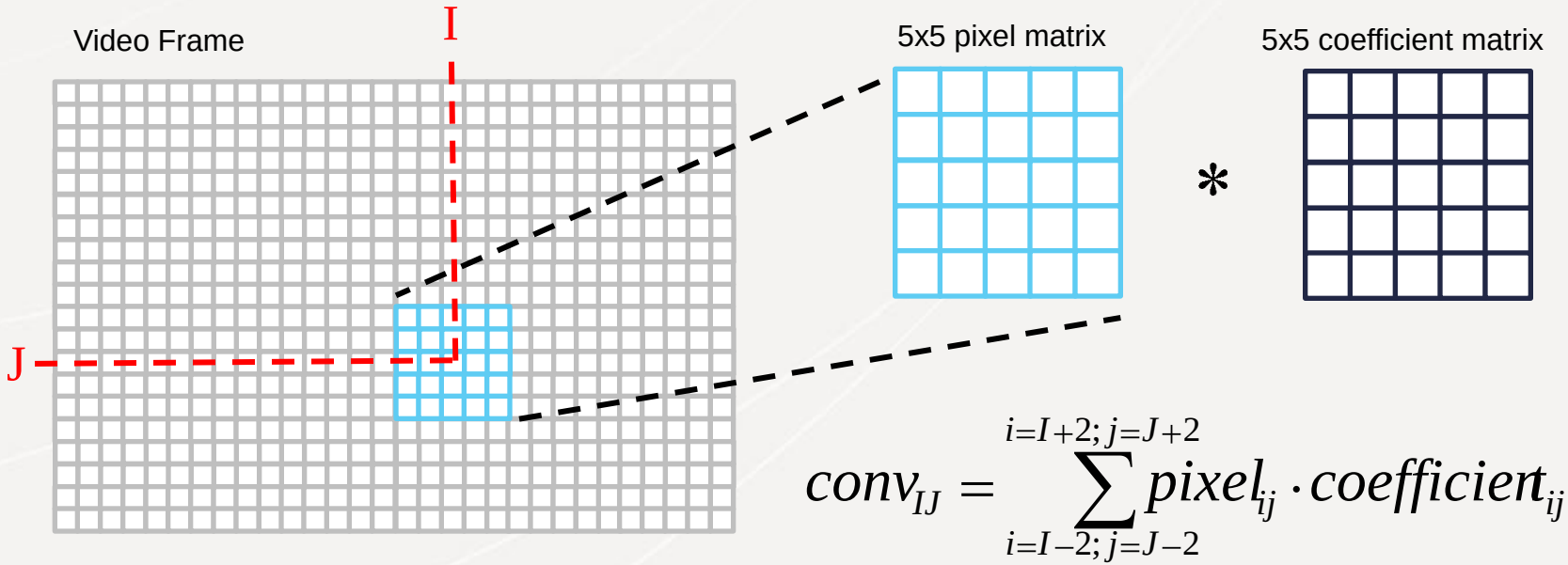
Video I/O Requirements

- 1080P60 Resolution
  - Video Bandwidth = 33 Mbits * 60 FPS = 2.0 Gbps
- Challenge => Processor needs dedicated high speed video interfaces



Video Processing – 2D Filter (5x5)

- 2D Convolution of pixels with coefficient matrix

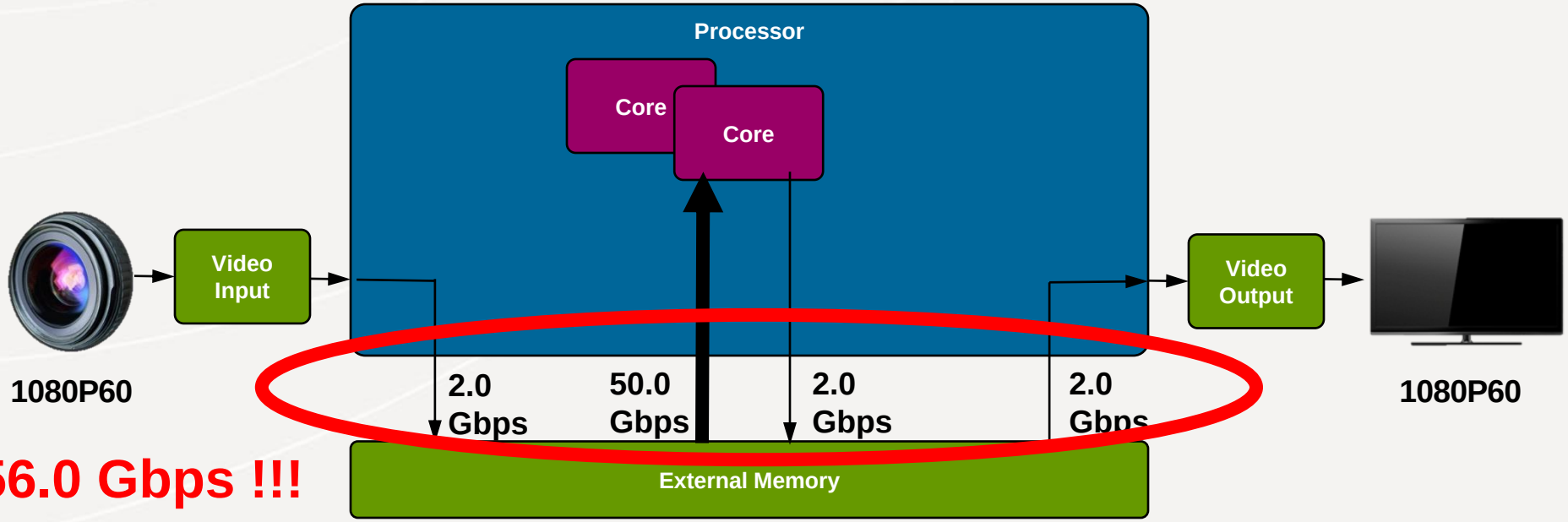


$$conv_{IJ} = \sum_{i=I-2;\, j=J-2}^{i=I+2;\, j=J+2} pixel_{ij} \cdot coefficient_{ij}$$

- For each input pixel
  - 25 Memory Read operations
  - 25 Multiply-Accumulate operations
  - 1 Memory Write operations

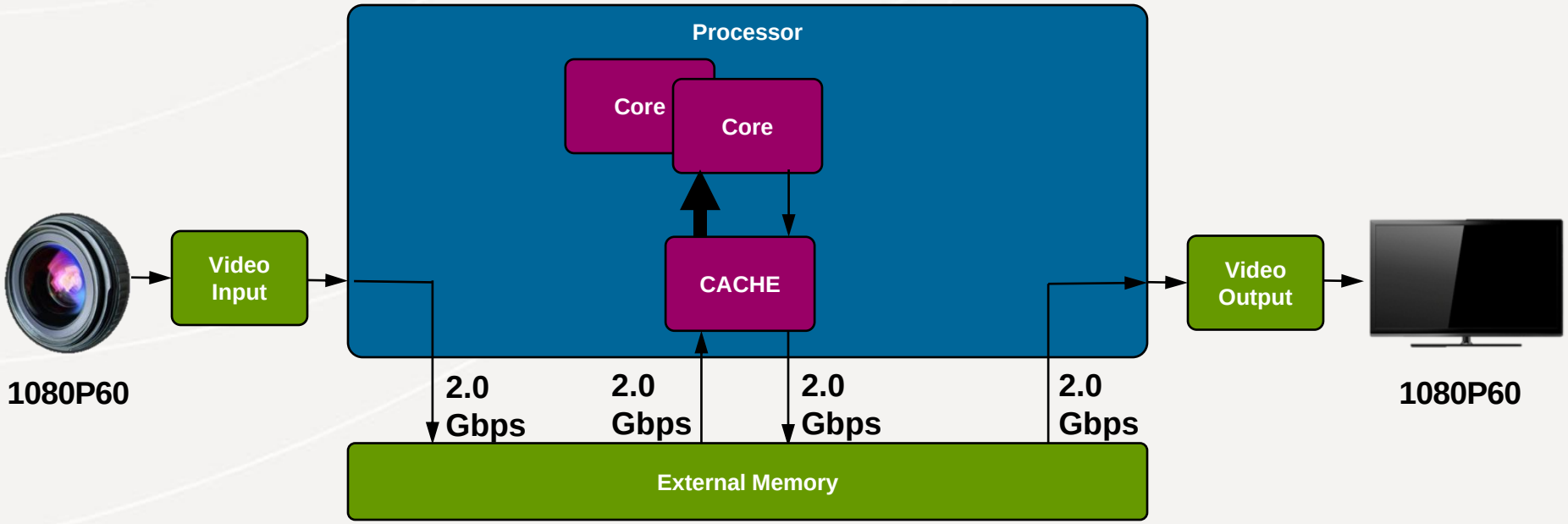
Video Processing Requirements

- 2D Filter (5x5) - 1080P @ 60FPS
  - MACs = 25 MAC * (1920x1080) * 60 fps = 3.1 GMACs/sec
  - MEM Read bandwidth = 2.0 Gbps * 25 = 50.0 Gbps
  - MEM Write bandwidth = 2.0 Gbps * 1 = 2.0 Gbps
- Challenge => External memory accesses are expensive



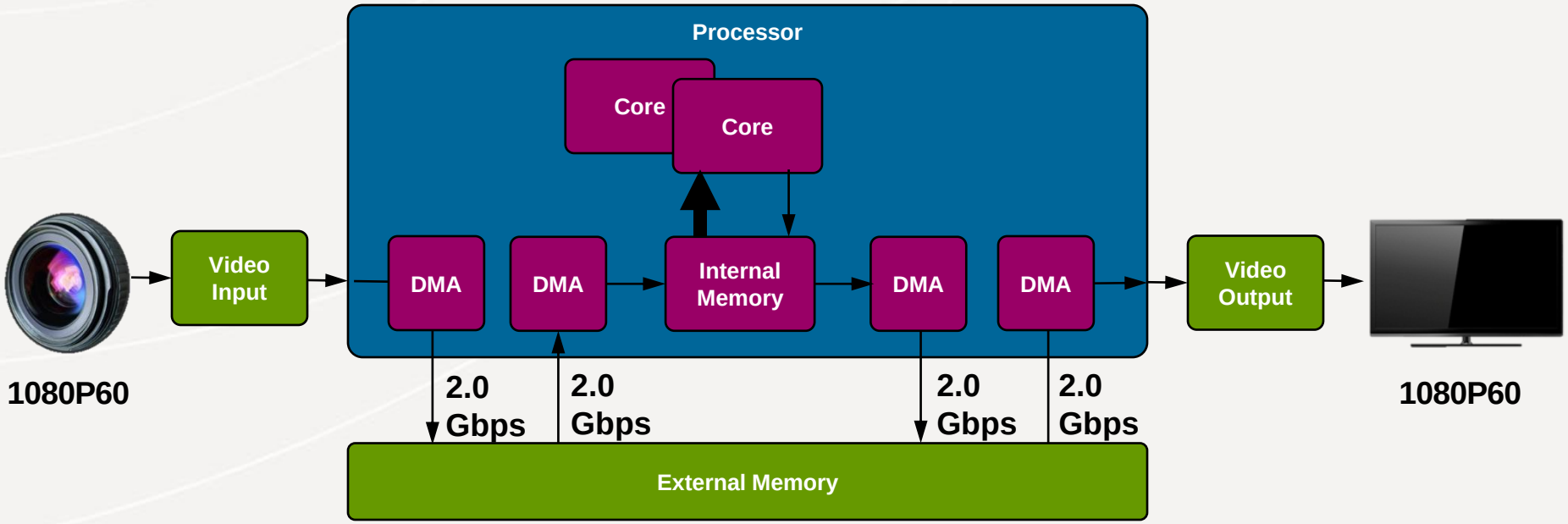
Accelerating with Cache

- "Cache" the pixels
  - Read each pixel only once
  - Explicitly by re-writing the source code
  - Enabling the processor cache
- Challenge => processor cache not dedicated to video, results may vary



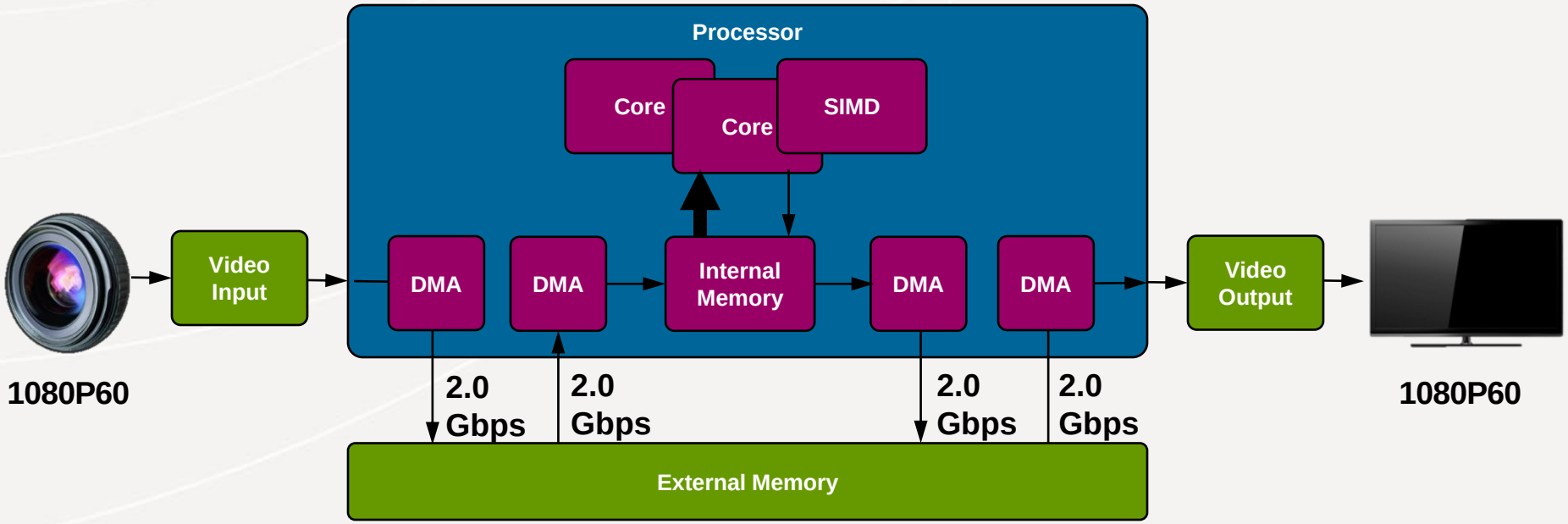
Accelerating with Direct Memory Access

- Using Direct Memory Access
  - Explicitly transfer video to internal memory for processing
  - Transfers occur in parallel with processing
  - Also used for video input/output interfaces
- Challenge => processor is not the ideal engine for pixel processing



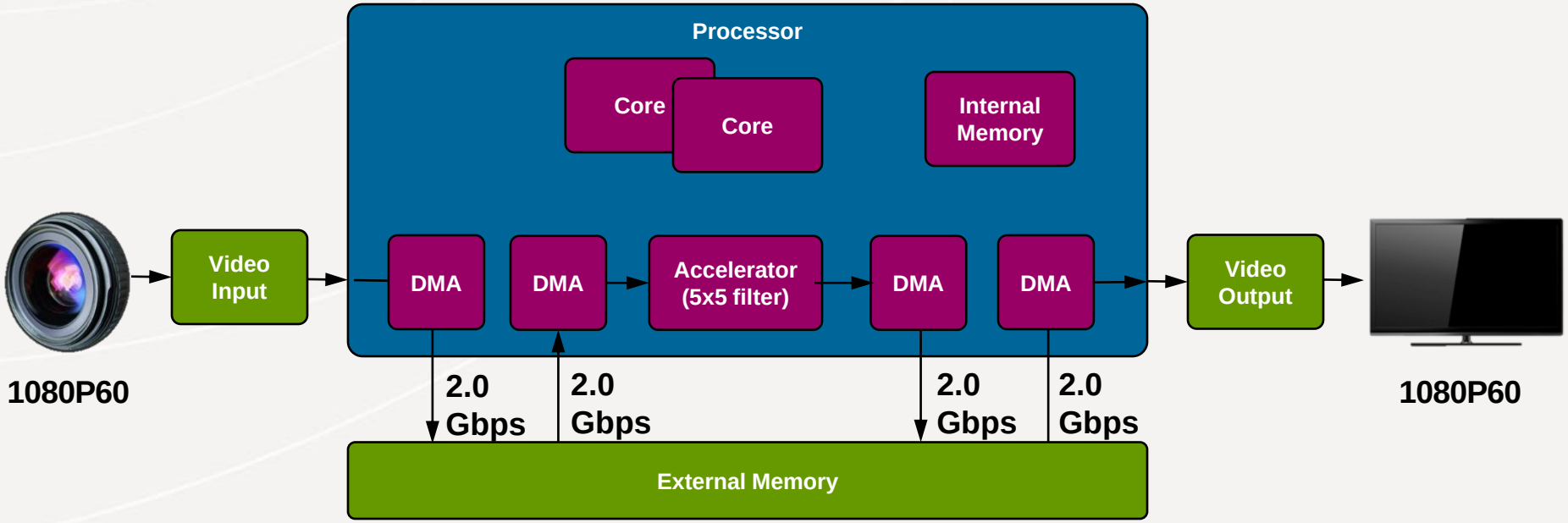
Accelerating with Dedicated Hardware

- SIMD (Single Instruction Multiple Data)
  - Acceleration provided via compiler intrinsics or explicit coding
  - Intel : SSE provides 5X acceleration (via IPP libraries)
  - ARM : NEON provides 2X (–mfpu=neon) to 5X (explicit coding)
- Challenge => May require explicit coding, Upper limit to acceleration



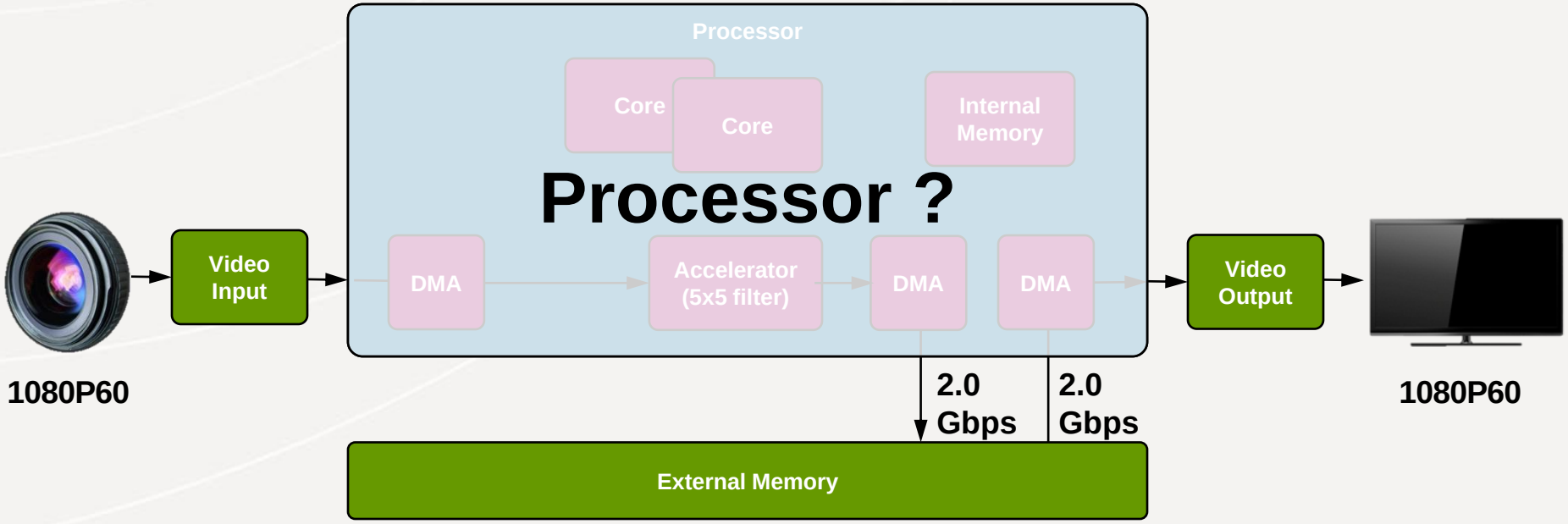
Accelerating with Dedicated Hardware

- Application Specific Accelerator : 2D Filter (5x5)
  - Dedicated internal memory for pixel cache and coefficients
  - Dedicated MAC engines for parallel execution
  - Acceleration : 25X for case of 5x5 2D Filter
- Challenge => Minimize unnecessary data paths in system

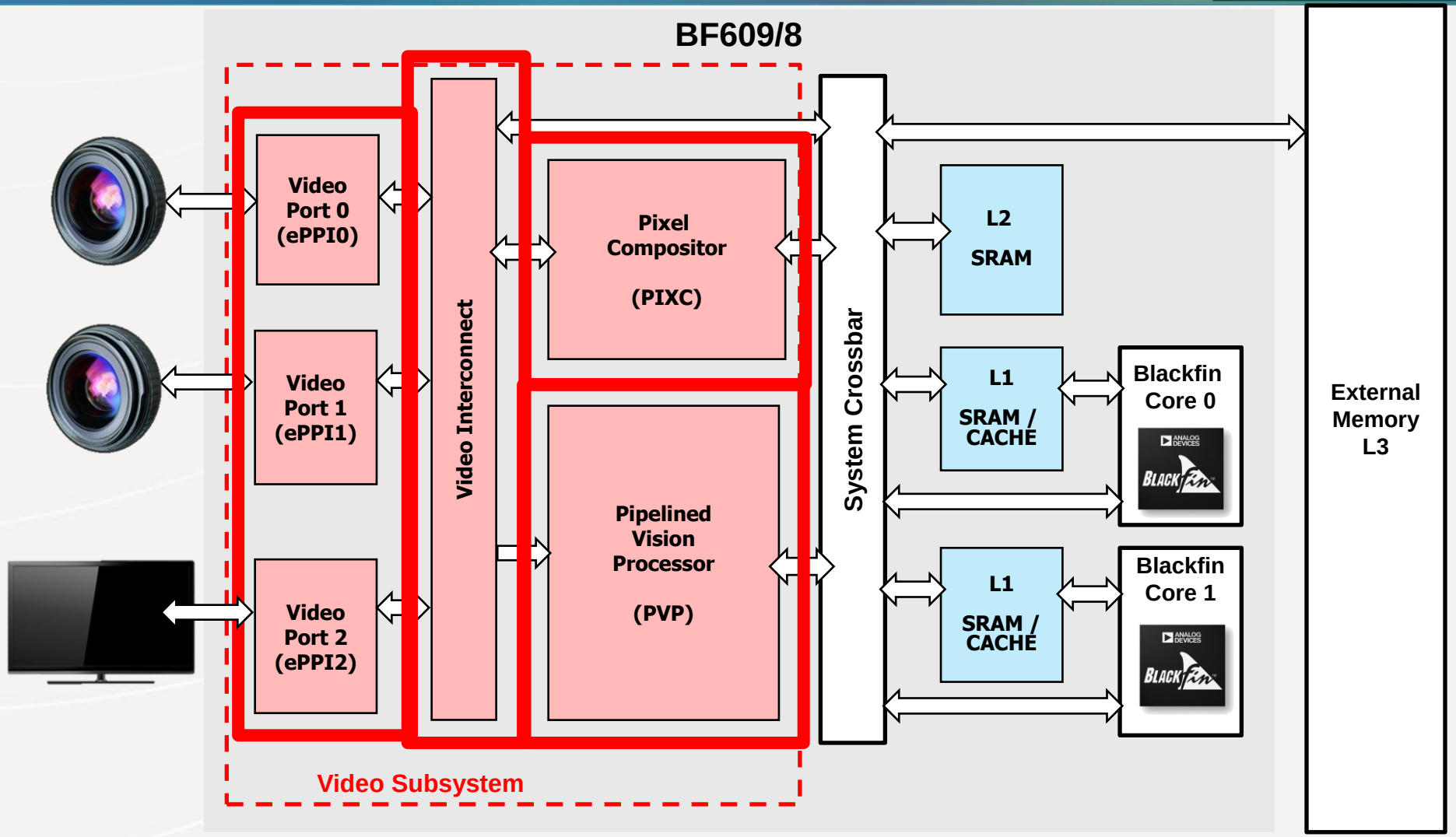


Moving Processing Inline with Data Path

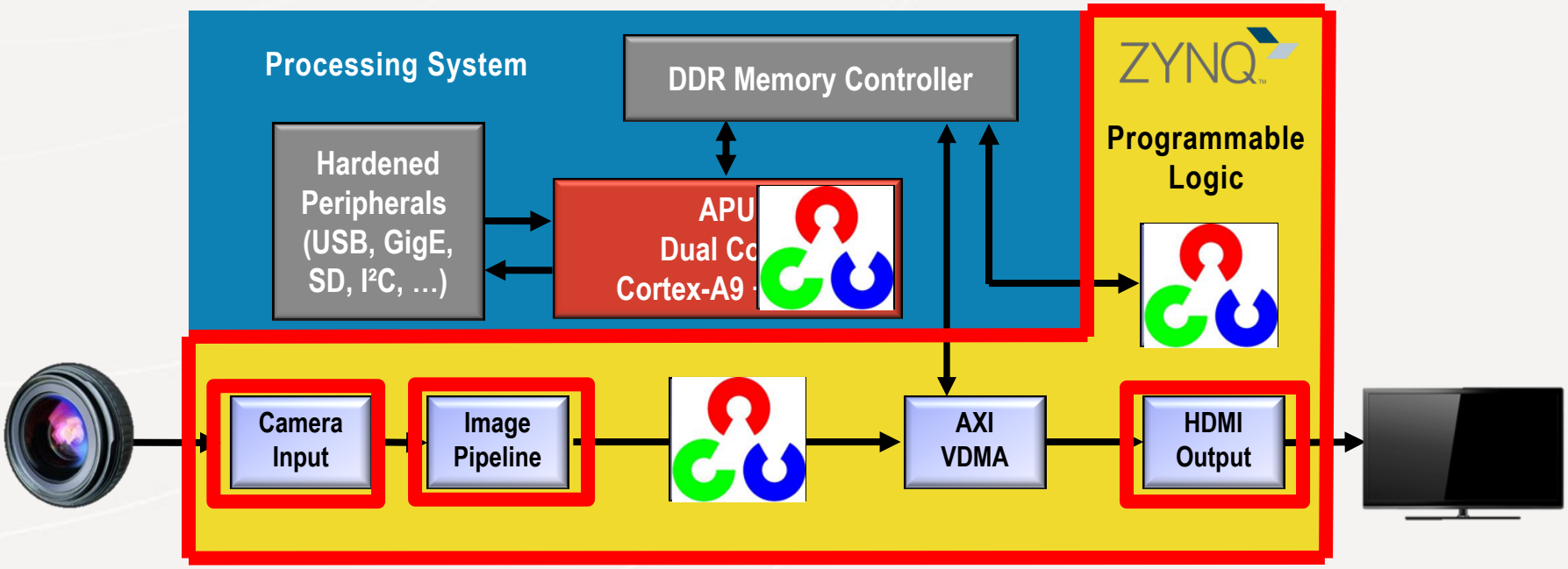
- Hardware Accelerators Inline with Data Path
  - Removes unnecessary external memory accesses
  - Reduces power consumption
  - Frees up processor(s) to perform more complex tasks
- Challenge => Build a processor with required accelerators ?

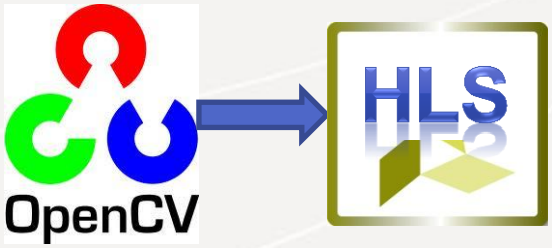


Use Case 1
BF609 - Dual-Core Blackfin Vision Processor



Use Case 2
Xilinx Zynq-7000 All Programmable SoC





- XAPP 1167 – Accelerating OpenCV Applications with Zynq using High-Level Synthesis (HLS)

Conclusions

- Computer Vision
  - Requires specialized hardware for real-time execution
- Embedded Hardware
  - Manufacturers are creating platforms optimized for vision
- What to look for in an embedded vision platform ?
  - Dedicated high-speed video interfaces
  - Hardware Accelerators
    - SIMD (for 2x to 5X acceleration)
    - Application specific (for 10x to 100x or more acceleration)
  - Flexible Video Data Paths
    - Reduce external memory bandwidth (save power !)

Resources for Further Investigation

- Embedded Vision Alliance resources (web site)
  - http://embedded-vision.com
- Visit the community web sites
  - http://FinBoard.org => Analog Devices BF609
  - http://ZedBoard.org => Xilinx Zynq-7000 All Programmable SoC

- Come Visit the Avnet Demos !
  - FinBoard – Finding Objects in Live Video
    - Dice Counting demonstration
  - ZedBoard – Your Programmable SoC Vision Platform
    - 1080P60 camera with optimized OpenCV functions