embedded VISION SUMMIT

Designing a multi-core architecture tailored for pedestrian detection algorithms

Tom Michiels Manager R&D Synopsys

October 2, 2013

SYNOPSYS®
Accelerating Innovation

Synopsys—EDA Industry Leadership

- Business data
  - $1.76B in revenue (2012),
  - ~8400 employees (> 5100 R&D engineers)
  - ~80 offices worldwide
- Products for Designing Embedded Vision Systems
  - Synthesis and verification for SoCs and FPGAs
  - Semiconductor IP (USB, HDMI, ARC Processor, A/D, D/A, …)
  - FPGA-based prototyping system
  - Application Specific Processor (ASIP) design tools

Copyright © 2013 Synopsys

Pedestrian Detection and HOG

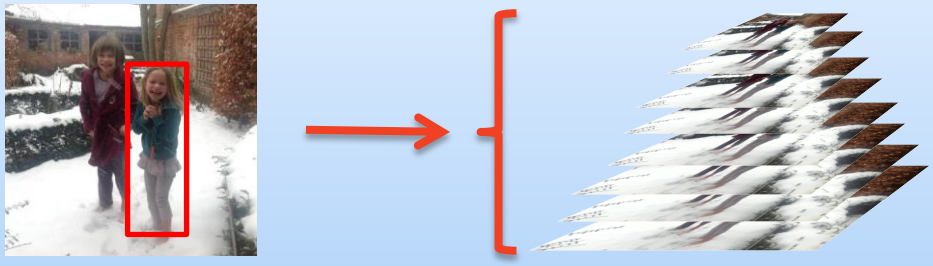
- Pedestrian detection
  - One of the most popular EV applications
  - Standard feature in luxury vehicles
  - Moving to mid-size and compact vehicles in the next 5-10 years, also due to legislation efforts
- Implementation requirements
  - Low cost
  - Low power (small form factor, and/or battery powered)
  - Programmable (to allow for in-field SW upgrades)
- Most popular algorithm for pedestrian detection is Histogram of Oriented Gradients (HOG)

Copyright © 2013 Synopsys

Histogram Of Oriented Gradients

- Grey scale conversion
- Scale to multiple resolutions
- Gaussian blur
- Gradient computation
- Histogram computation per block
- Normalization of the histograms
- SVM per window position
- Non-max suppression

Scale to Multiple Resolutions



Use a fixed 64x128-pixel detection window.

Apply this detection window to scaled frames.

Gaussian Blur

Apply two-dimensional Gauss filter: —

Gradient Computation

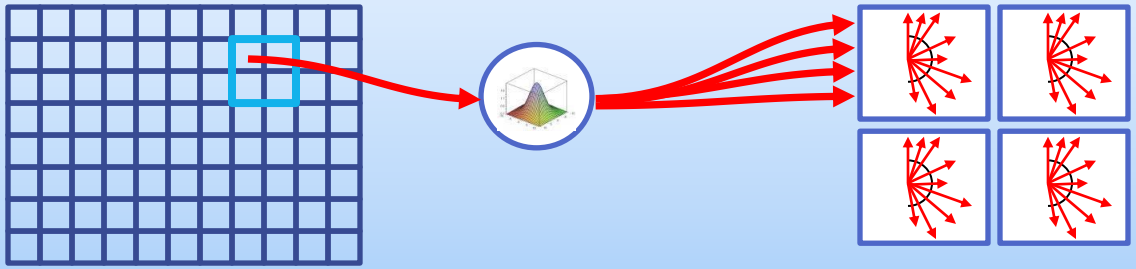
Apply Sobel operators: and

Copyright © 2013 Synopsys

Histogram Of Oriented Gradients

- Grey scale conversion
- Scale to multiple resolutions
- Gaussian blur
- Gradient computation
- Histogram computation per block
- Normalization of the histograms
- SVM per window position
- Non-max suppression

Histogram Computation



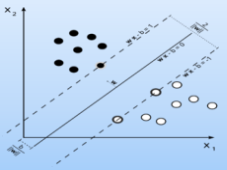
The image is divided in 8x8-pixel cells. For very block of 2x2 cells, apply Gaussian weights and compute 4 histograms of the orientation of the gradients.

Normalization of the Histograms

(1) L2 Normalization (2) clipping (saturation) (3) L2 Normalization

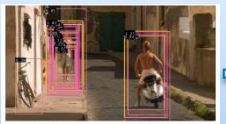
Support Vector Machine

Linear classification of histograms for every 64x128 windows position.

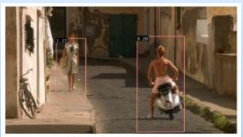


Non-Max Suppression

Cluster multi-scale dense scan of detection windows and select unique



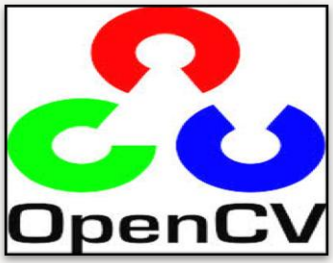
Multi-scale dense scan of detection window



Final detections

Copyright © 2013 Synopsys

Reference Data and Design Goals



2 VGA frames per second on Core i3@2.99GHz 80 Watt

20 VGA frames per second on: Nvidia GTX280: 236Watt

7 VGA frames per second on: ARM Cortex A15 ~ 8 Watt

Design Goals

- HOG in VGA at 20 frames per second
- ASIC design, TSMC 45nm LP
- 400 MHz @ < 500mW in < 500kGates
- HAPS FPGA-based system prototype @ 70 MHz

Copyright © 2013 Synopsys

HOG Platform Design Principles

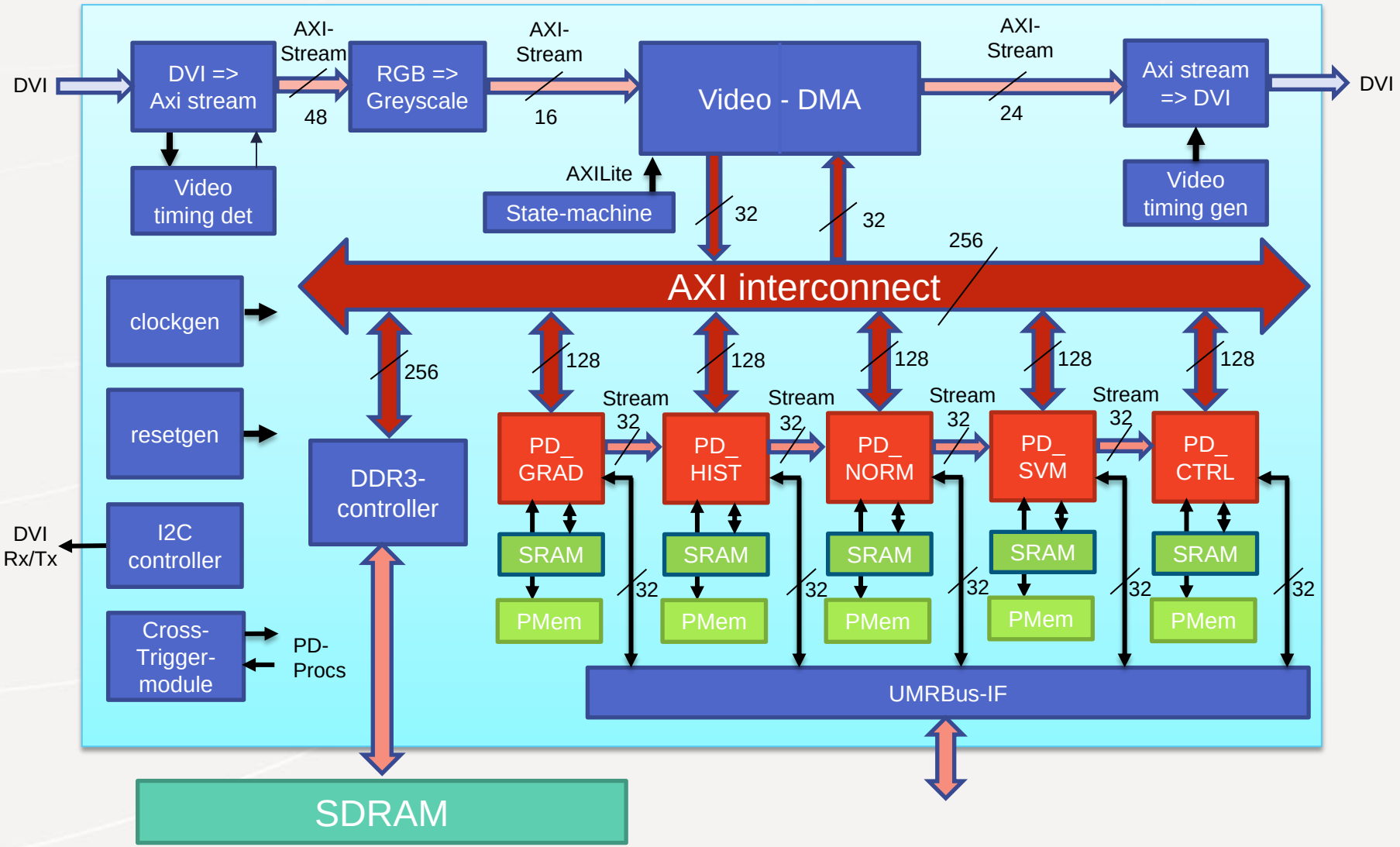
1. Multi-core streaming processors

2. Minimal memory usage
   - Only store full frame in SDRAM
   - Use FIFO's to communicate between streaming processors
   - Use minimal local SRAM

3. Full programmability of every processor
   - Use minimal RISC with FIFO interface
   - Add vector-slot with specialized instructions

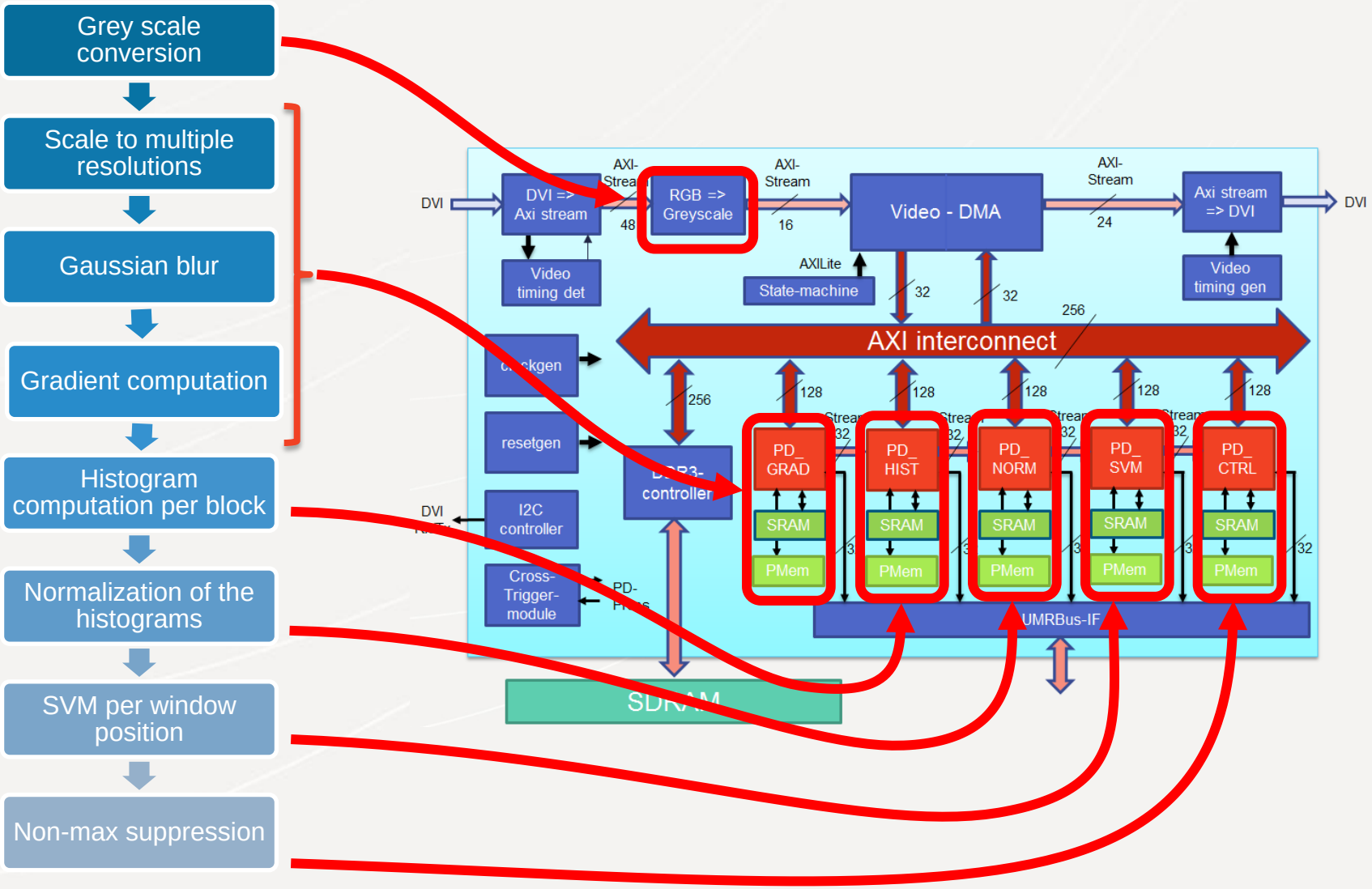
Copyright © 2013 Synopsys

HOG Detection Platform



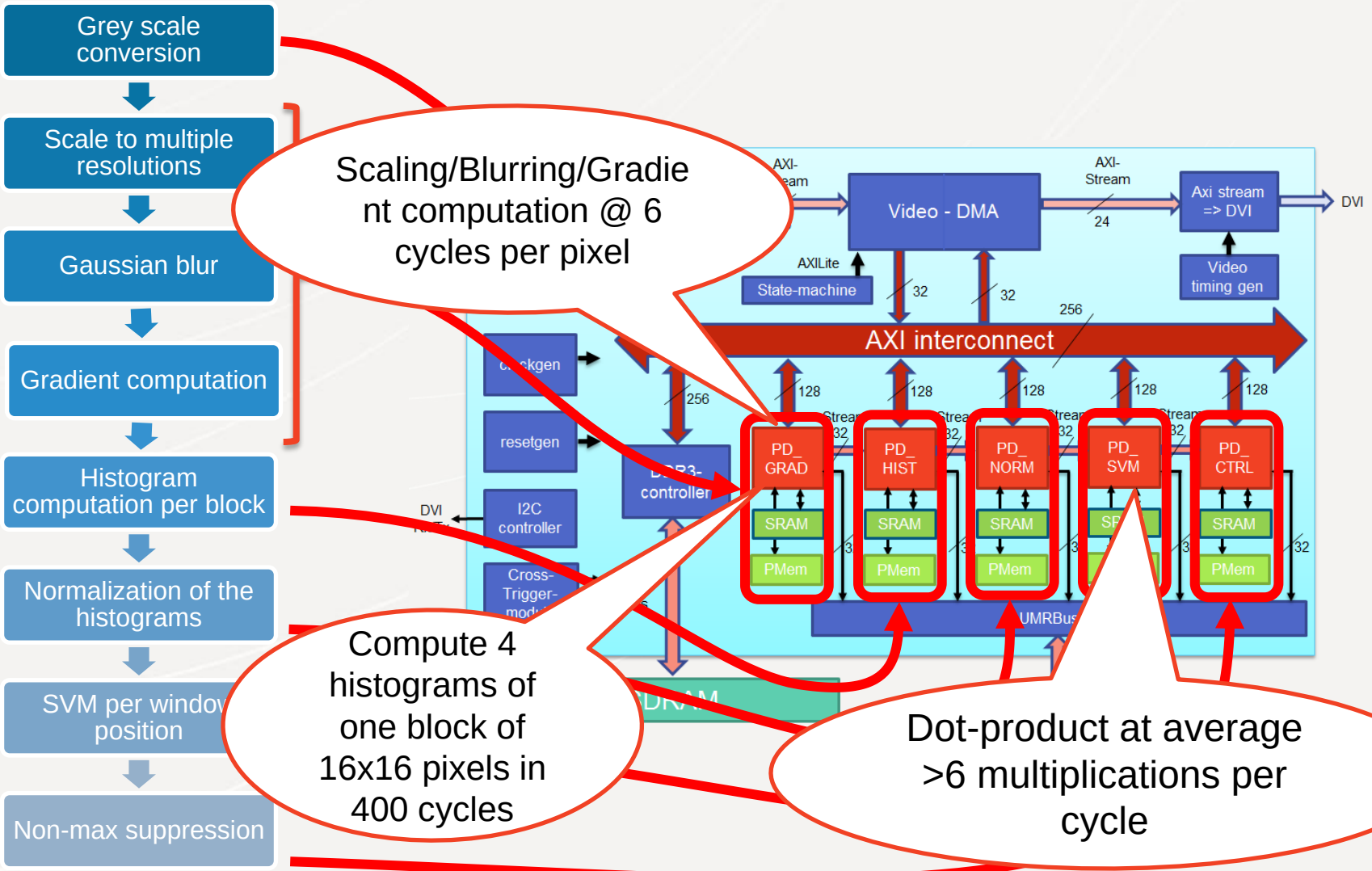
Copyright © 2013 Synopsys

Mapping HOG to 5-Core Design



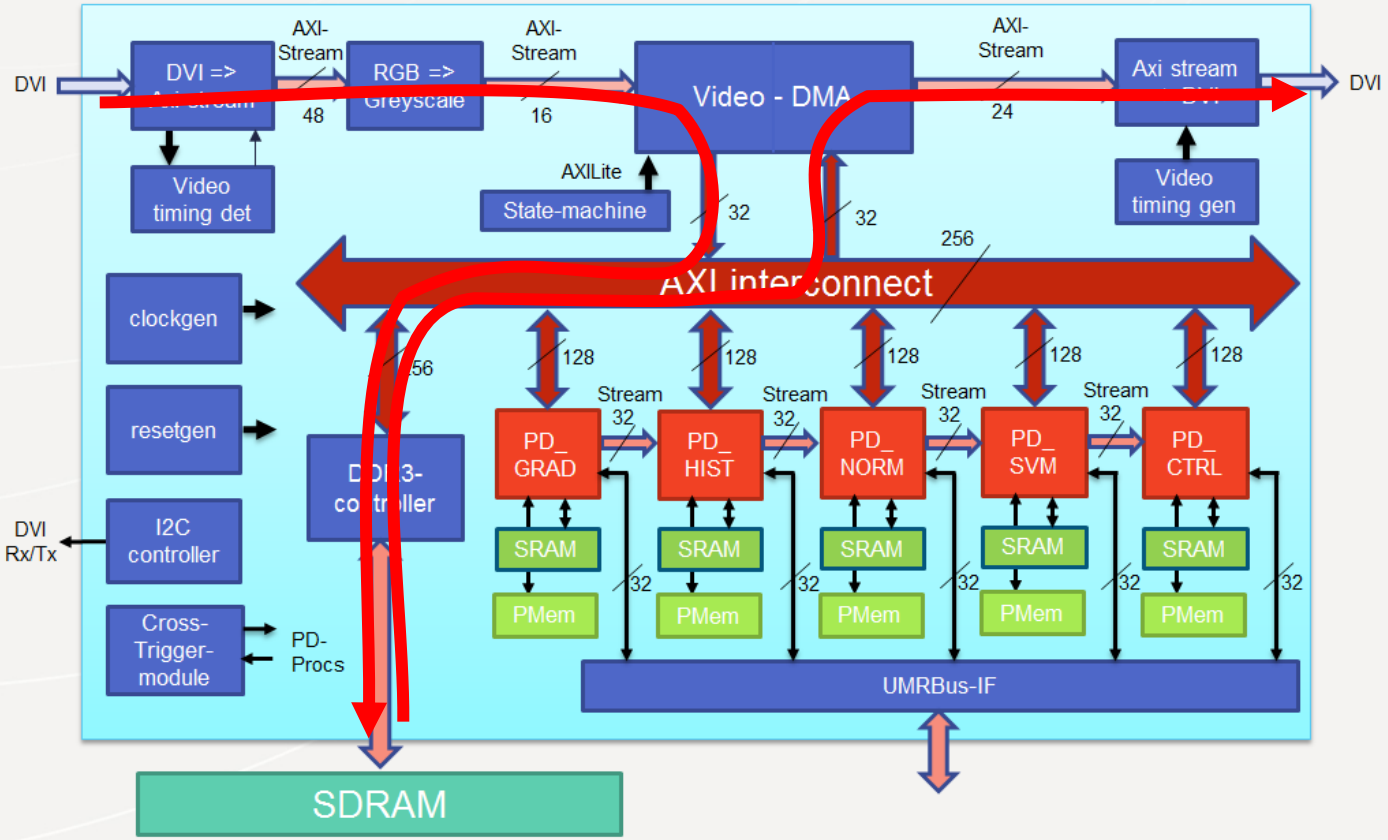
Copyright © 2013 Synopsys

Performance Requirements



Copyright © 2013 Synopsys

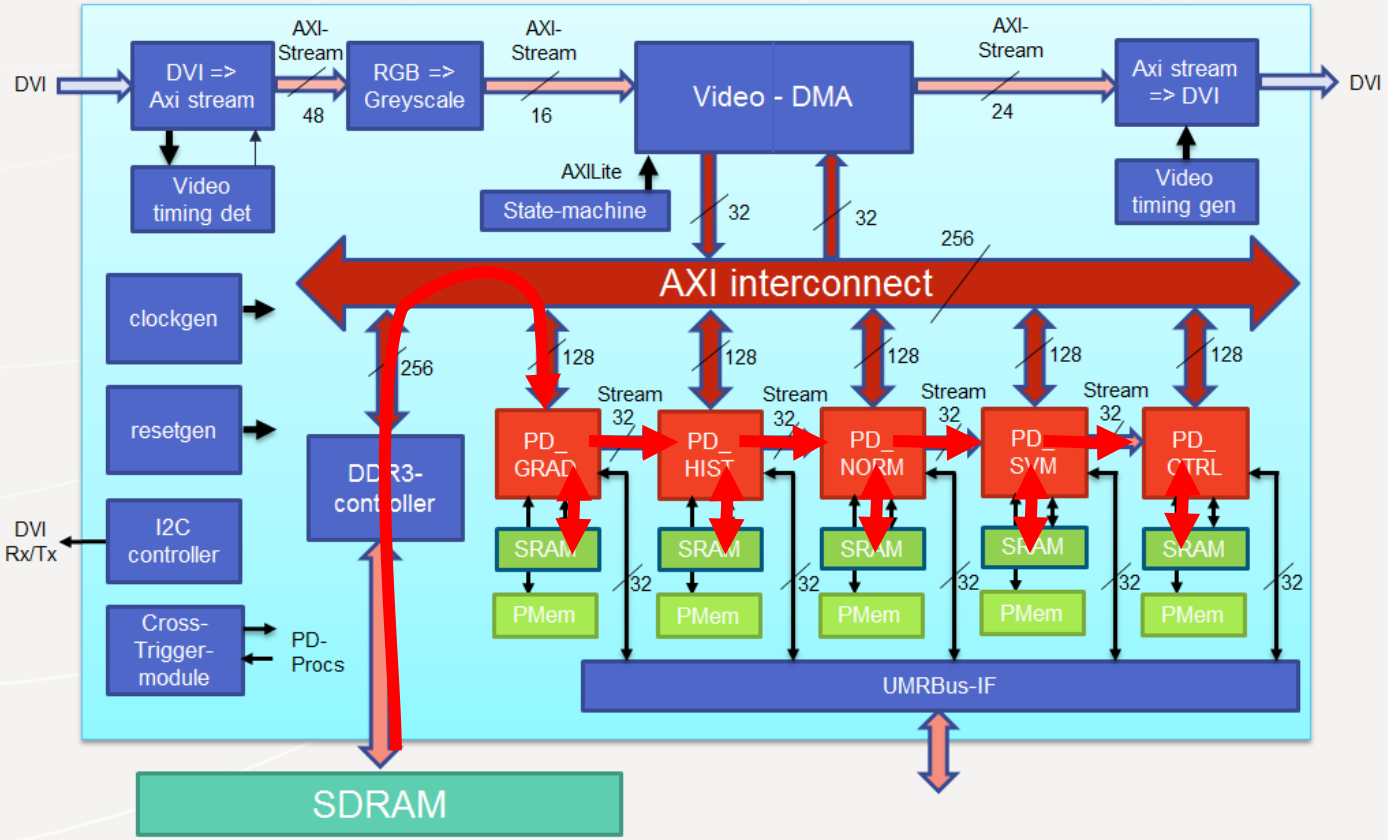
Memory Architecture



Frames are copied directly from/to DVI to/from SDRAM by the DMA

Copyright © 2013 Synopsys

Memory Architecture



All ASIPs avoid AXI access where possible.

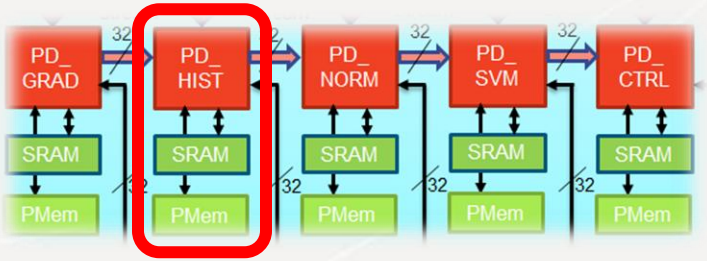
All intermediate data is stored in a local small SRAM (64KB)

ASIPs exchange data via point-to-point connections.

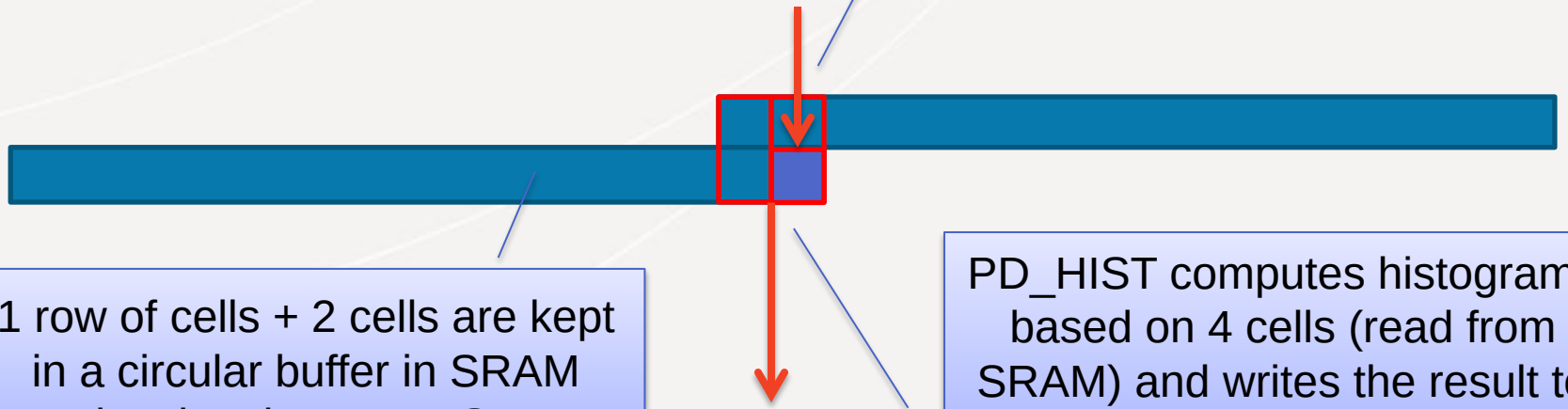
Copyright © 2013 Synopsys

Minimal Local Memory Usage

Example: Memory usage of PD_HIST



PD_HIST reads 1 cell (8x8 pixel) of gradients from the PD_GRAD via the FIFO

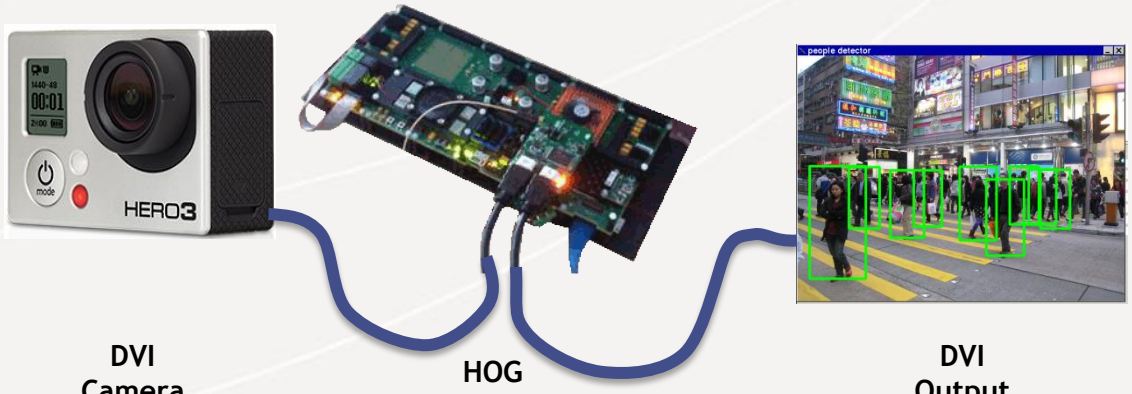


1 row of cells + 2 cells are kept in a circular buffer in SRAM local to the PD_HIST

PD_HIST computes histograms based on 4 cells (read from SRAM) and writes the result to the PD_NORM via the FIFO

Copyright © 2013 Synopsys

Design Methodology



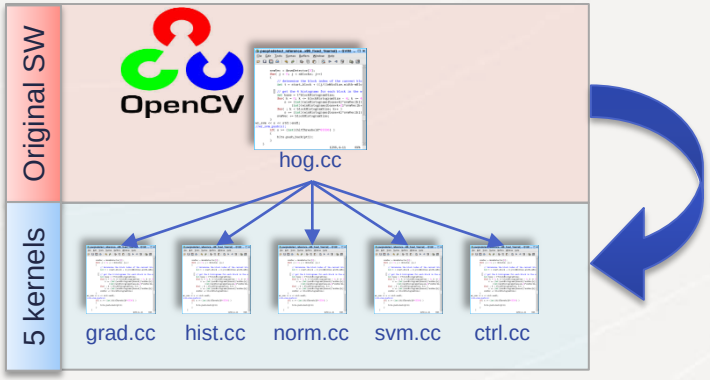
DVI Camera

HOG Detection

DVI Output

Copyright © 2013 Synopsys

Design Methodology



Design Tasks

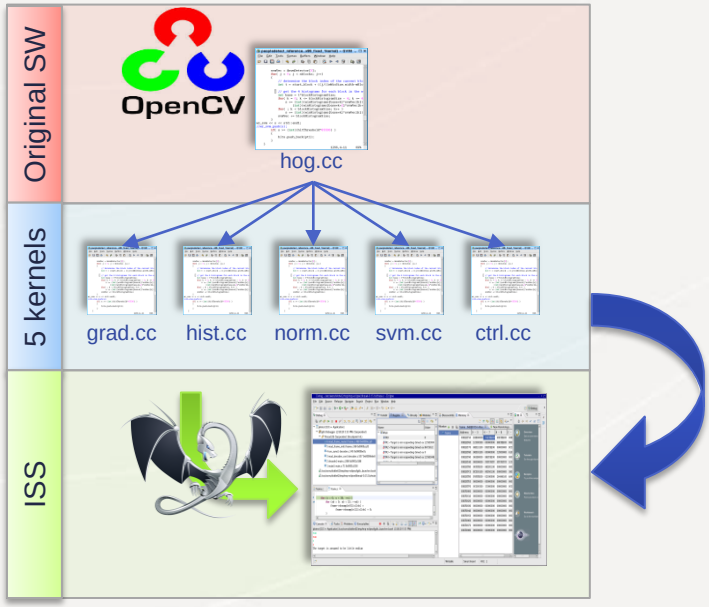
1. Start from algorithms in C++ running on the host.
2. Split the kernel into 5 parts communicating via FIFO's
3. Test the 5 kernels on the host.

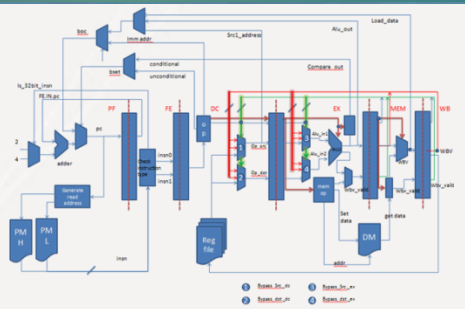
Implications

- Quick start with OpenCV or existing software
- Functional reference implementations of the ASIPs can be used for verification
- Fast software model of HOG pipeline for SVM training

Copyright © 2013 Synopsys

Design Methodology





Design Tasks

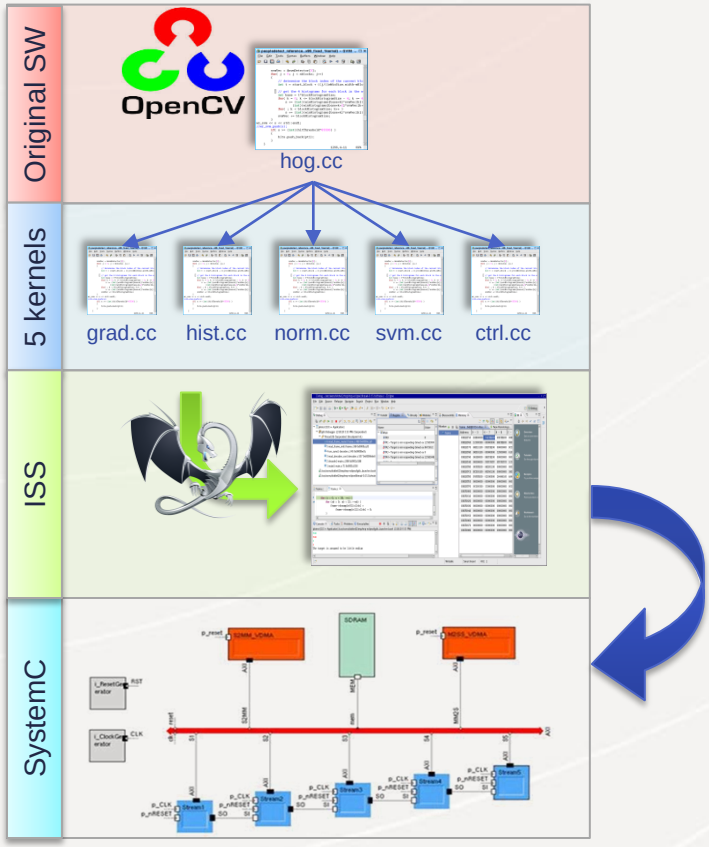
1. Compile the kernels for a simple embedded RISC architecture
2. Make link-scripts to fit the program/data memory constraints
3. Verify the kernels on standalone instruction set simulator

Implications

- Working base model of the Application Specific Processor early on in the design flow

Copyright © 2013 Synopsys

Design Methodology



Design Tasks

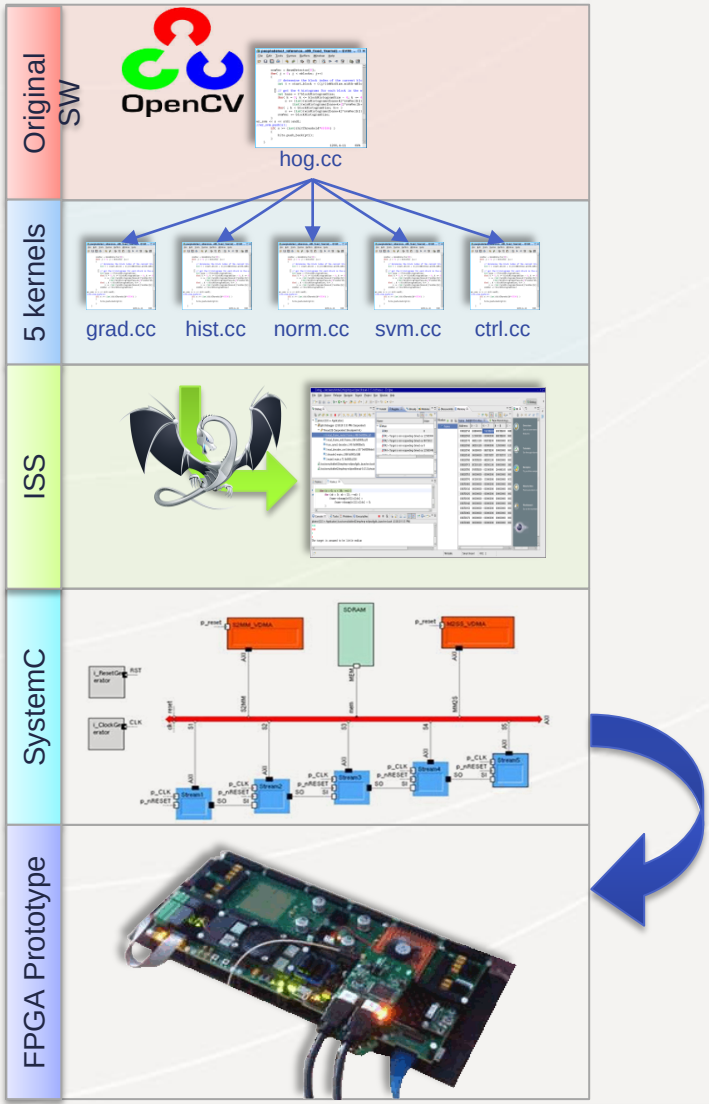
1. SystemC model with Synopsys Virtualizer
2. Compile the five kernels with LLVM compiler for RISC core
3. Simulate and validate system behavior

Implications

- Multi-core debug environment
- Early validation of system behavior
  - Multi-core interaction
  - Usage of SDRAM, FIFO's, local SRAM
  - DMA and Frame synchronization
- Slow, but functionally correct reference system model

Copyright © 2013 Synopsys

Design Methodology



Design Tasks

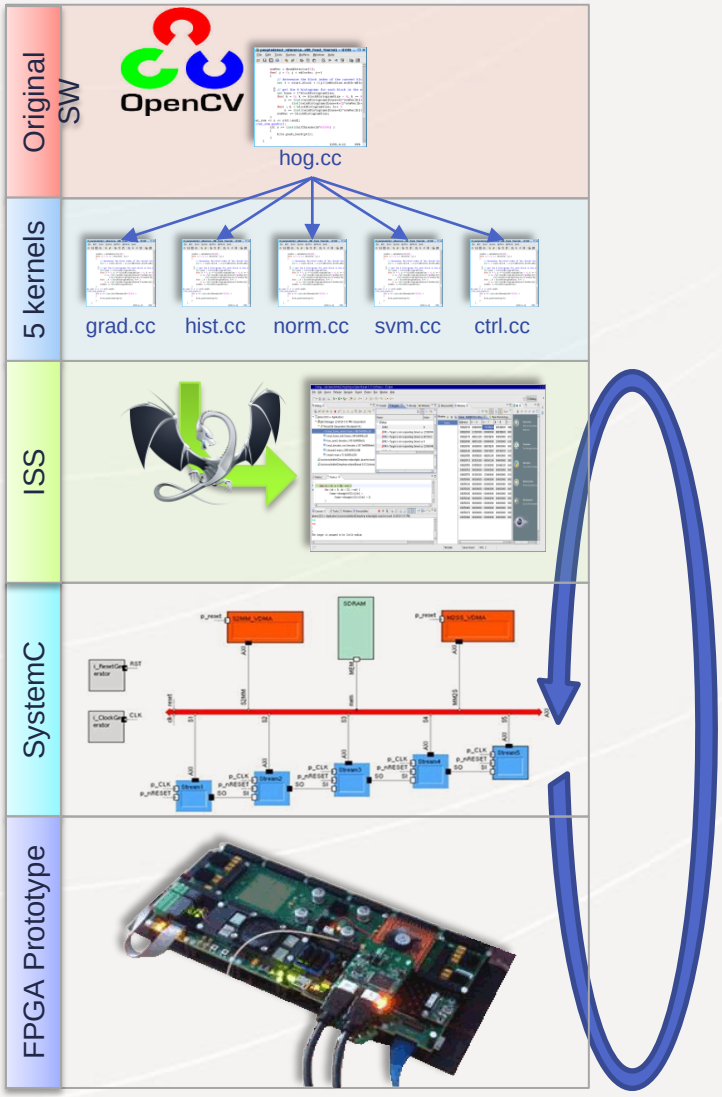
1. Assemble HDL system with RISC cores
2. Interface with SDRAM and DVI
3. Run HOG on FPGA prototype with 5 RISCs

Implications

- Early working prototype on FPGA
- Validate SW debugging flow
  - UMR-bus of JTAG for software debugging
- Base-line for performance validation & measurement

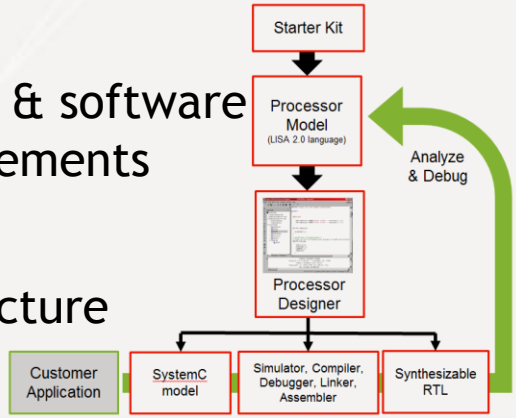
Copyright © 2013 Synopsys

Design Methodology



Design Tasks

1. Profile application code
2. Enhance the architecture & software to fit performance requirements
   - Add register files
   - Modify memory architecture
   - Add SIMD/VLIW
   - Add special instructions
3. Validate system behavior on SystemC model
4. Validate performance on FPGA

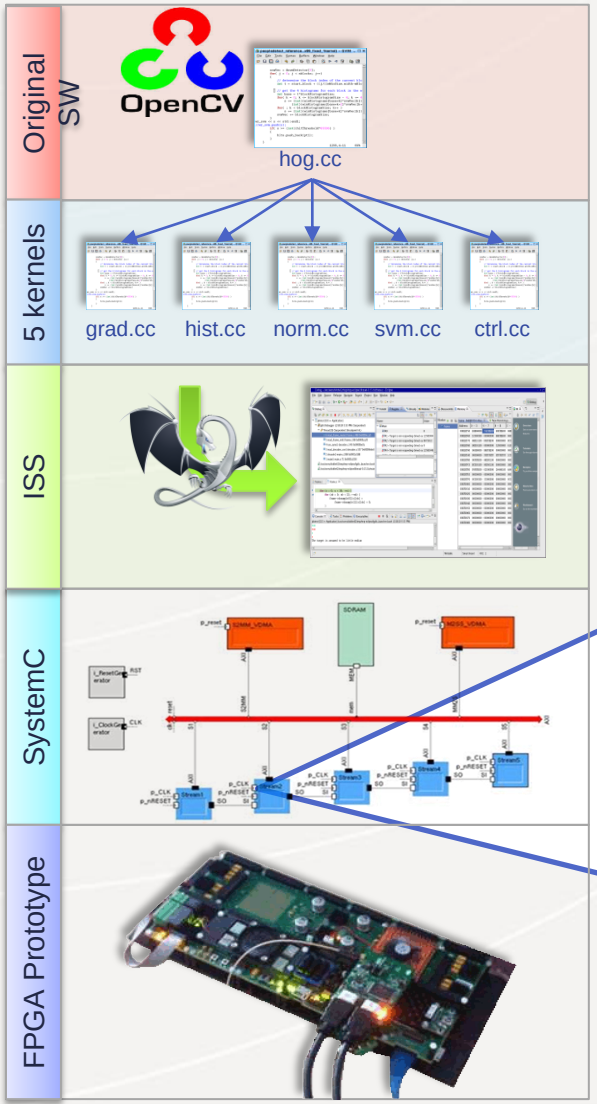


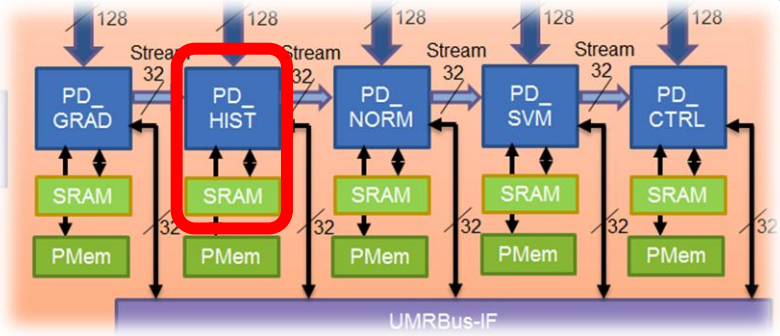
Implications

- Fully working prototype from the start
- Agile development process for processor architecture

Copyright © 2013 Synopsys

Design Methodology





Example: Gradient Computation by PD_HIST

- Vector-slot next to existing scalar instructions (VLIW)
- 16x(8/16)-bit vector register files
- 16x8-bit SRAM interface
- 16x8-bit FIFO interfaces
- Vector arithmetic instructions
- Special registers and instructions to compute histograms

4X size increase & 200X speedup

Copyright © 2013 Synopsys

Preliminary Results

- Prototype running at 60 MHz on HAPS
- 23 frames/second on 400 MHz
- 5 processors (without AXI, local SRAM)
  - TSMC 45nm GS “typical” condition
  - Synopsys Reference Methodologies, using DC Topographical (DCT)
  - Gate count: total gate count: 328.7 kGates
  - Power consumption: 42mW

Demonstration available at our booth

Copyright © 2013 Synopsys

Conclusions

- Embedded vision applications combine complex algorithms and high data rates with a need for low power
- For applications requiring ultimate efficiency, custom (multi-core)-processors on silicon are ideal
- Performance gains and power efficiency due to tailored instruction sets and dedicated memory architecture
- Key challenge is simplifying the design process while retaining flexibility to change the algorithms

Copyright © 2013 Synopsys

Resources

- HOG algorithm: en.wikipedia.org/wiki/Histogram_of_oriented_gradients
- Synopsys Processor Designer: www.synopsys.com/pd
- Embedded Vision Development System:
  - Video: http://www.synopsys.com/Systems/BlockDesign/ProcessorDev/Pages/Videos.aspx
  - White paper: http://www.synopsys.com/cgi-bin/sld/pdfdla/pdfr1.cgi?file=pd_smb_vision_wp.pdf
- Synopsys web page: www.synopsys.com

Copyright © 2013 Synopsys