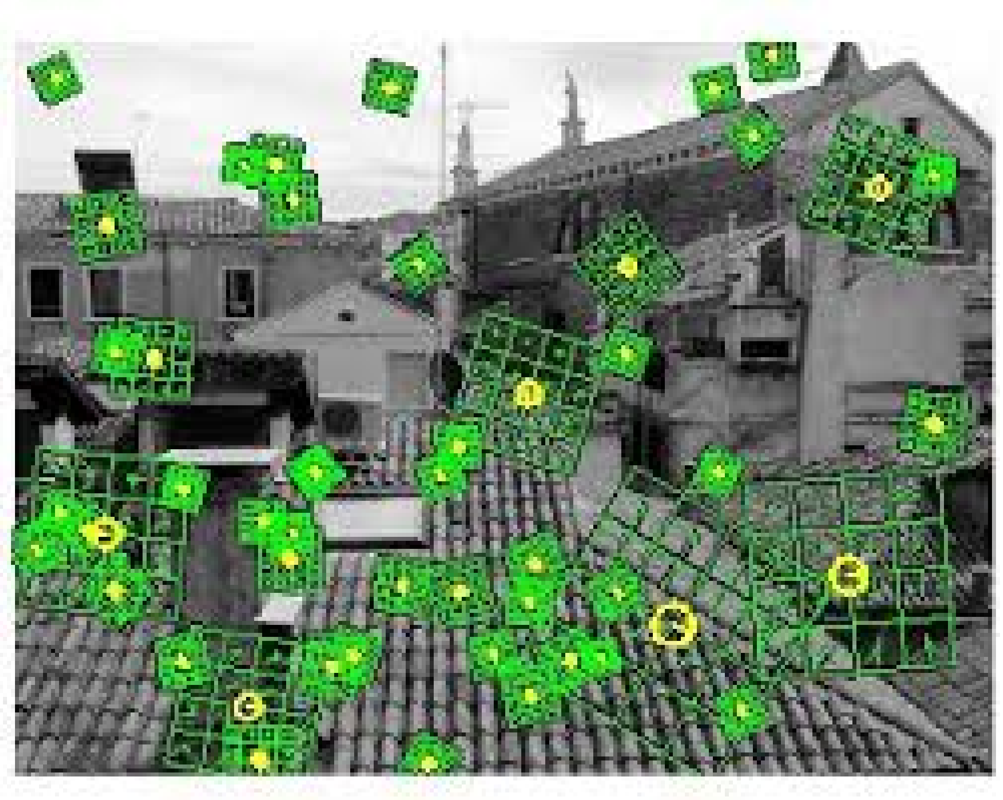
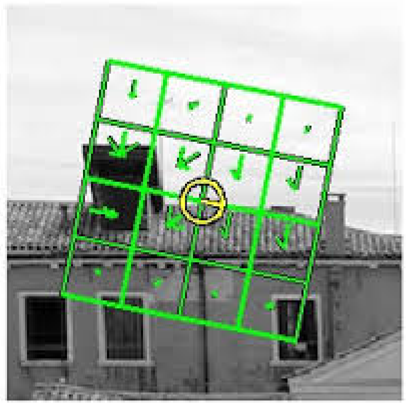
Design of Embedded Systems Extensions for Multimedia Processing

by Rafal Kapela

Local image descriptors – the idea

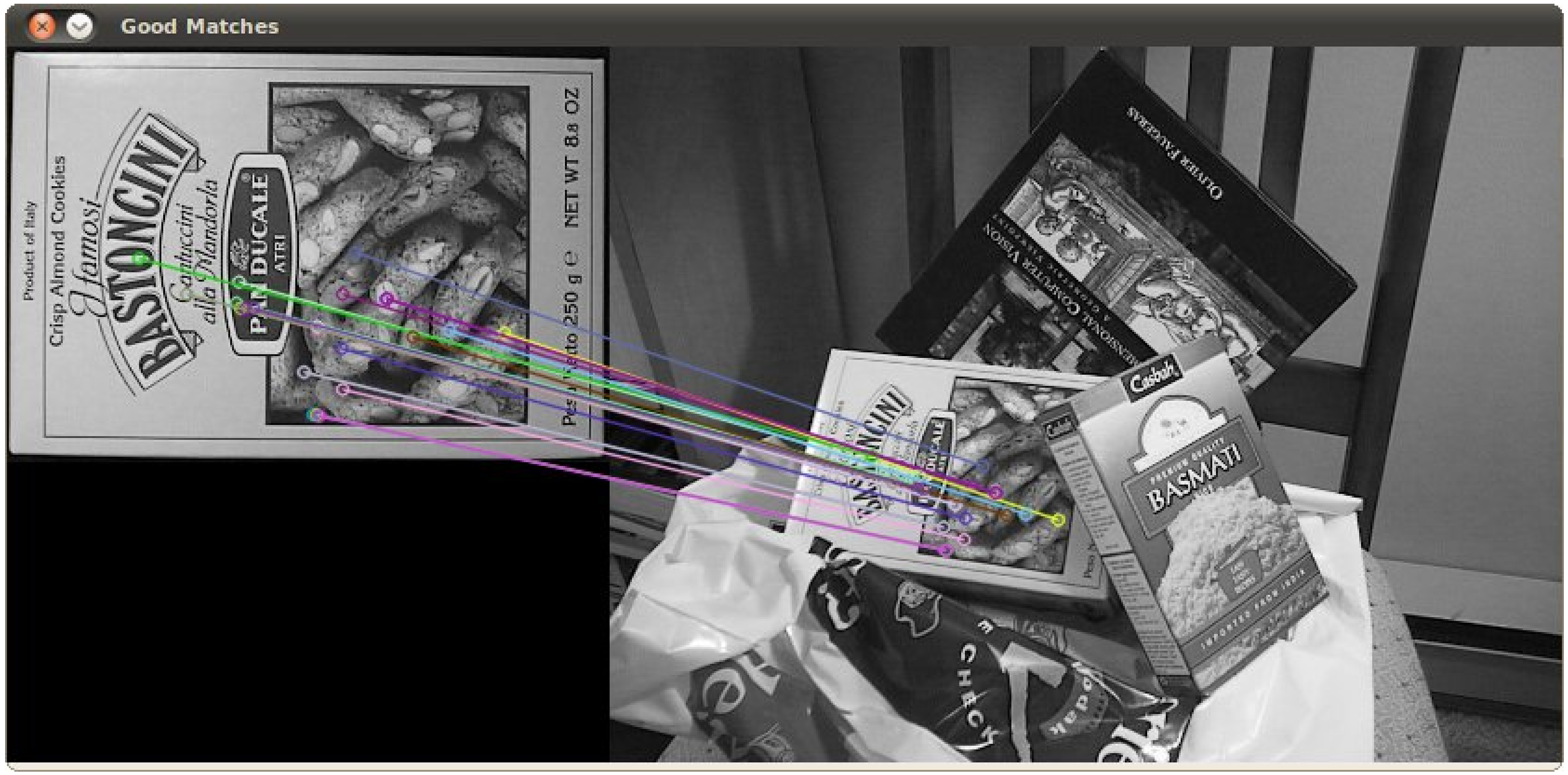
- Detect keypoints
- Produce descriptors around them





Local image descriptors – the idea

- Match descriptors to find objects - homography

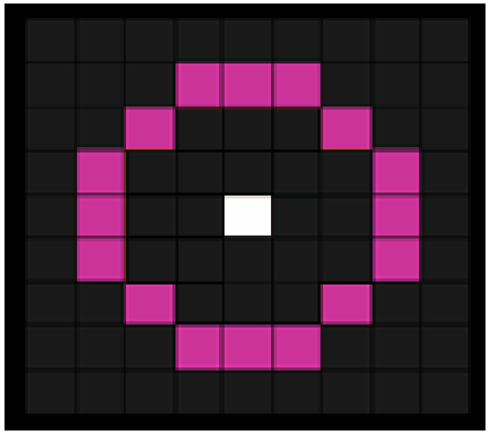


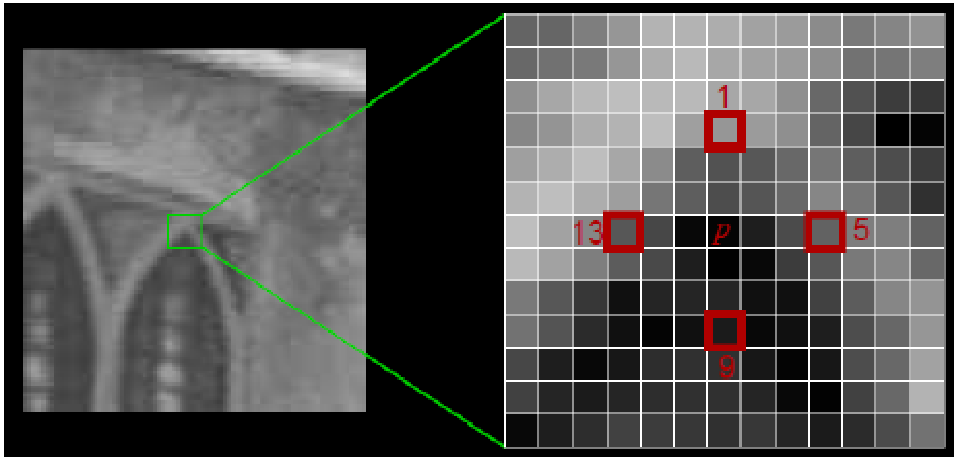
FREAK – Keypoint detector

- FREAK uses AGAST as a keypoint detector

  Adaptive and Generic Accelerated Segment Test

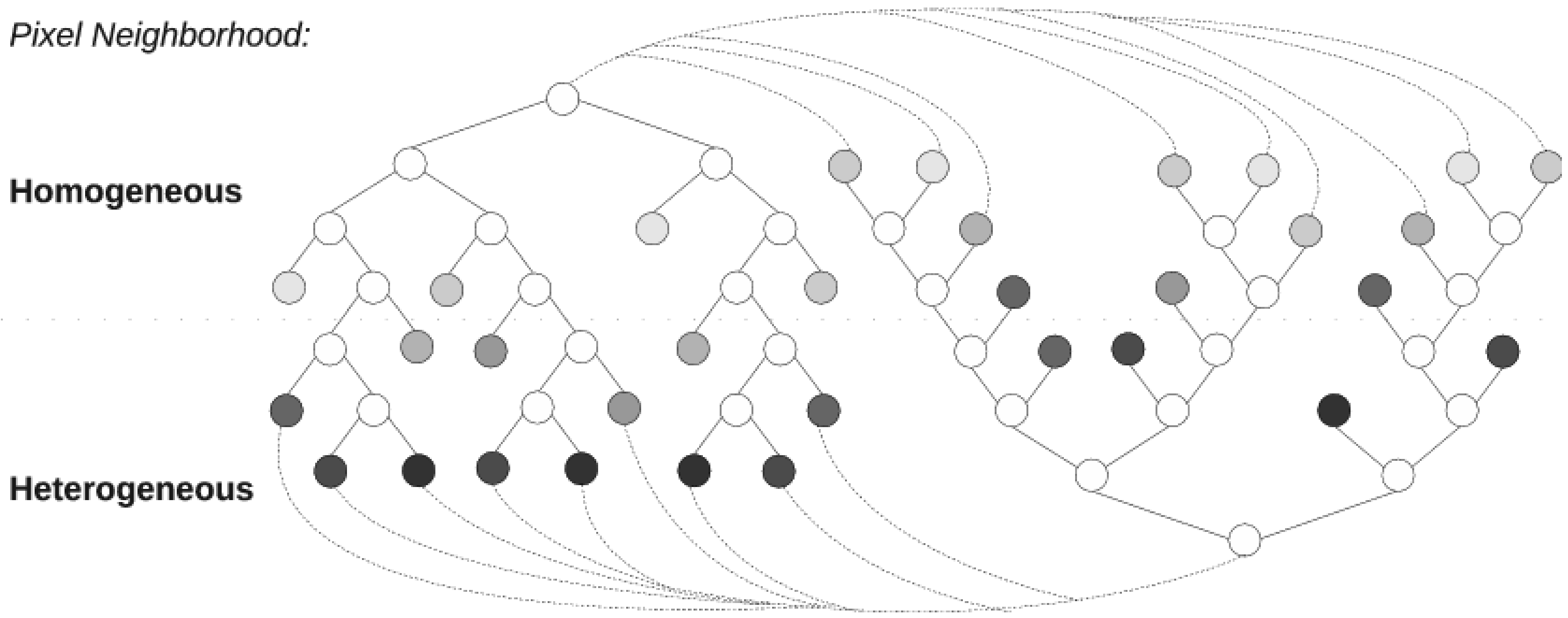
- Corner criteria:
  - based on a Bersehnam's circle (different sizes)
  - N connected pixels (of very similar luminance) either brighter or darker than the nucleus





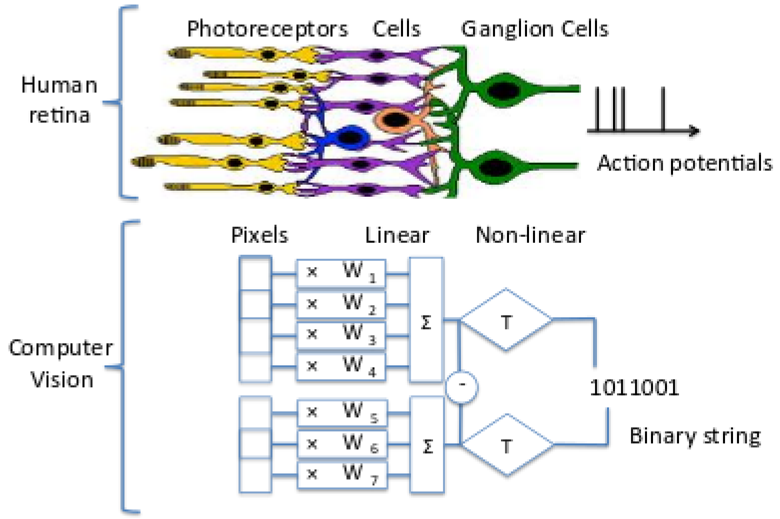
FREAK – Keypoint detector

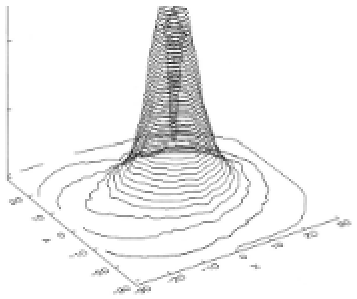
- AGAST
  - Builds a decision tree optimized for processing costs
  - Uses adaptive switching between
    - a tree for homogeneous areas
    - a tree for textured regions

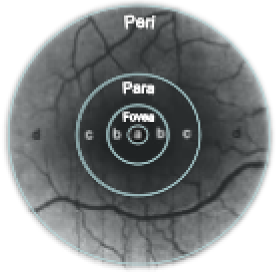


FREAK – Motivation

- Descriptor is built based on the topology of human retina:
  - extracts information using DoGs (diffrent sizes);
  - the spatial distribution of ganglion cells reduces expotentially with the radial distance from the foveola;
  - the size of the receptive field increases with radial distance from the foveola;







http://www.ivpe.com/papers/freak.pdf

FREAK – Sampling pattern

- The architecture of the descriptor was chosen experimentally:
  - sizes of the Gussian kernels change with respect to the log-polar retinal pattern;
  - neighboring receptive fields overlap eachother;

A>B, B>C ... A>C

$$T(P_a) = \begin{cases} 1 \; if \; \left(I\left(P_a^{r_1}\right) - I\left(P_a^{r_2}\right)\right) > 0 \\ 0 \; otherwise \end{cases}$$

$$F = \sum_{0 \leqslant a < N} 2^a T(P_a)$$

FREAK – which pairs?

- The procedure to choose pairs:
  - take large number of keypoints (50k in the paper);
  - create D matrix:
    - each row represents to a keypoint;
    - each column represents a pair (43 fields – 1k columns)
  - calculate a mean of each column (0.5 leads to the highest variance in binary distribution);
  - sort the columns with respect to their variance;
  - take the best column (the one of the highest variance);
  - check if the correlation between the chosen column and the ones already included to the descriptor;
  - low correlation – include the column; high correlation – do not include the column.

FREAK – saccadic search

- Humans do not look at a scene with fixed steadiness.
- Human's eye does saccades (discontinuous individual movements).
- The movements are very small, thus receptive regions located mainly around the foveola are stimulated (the ones with the highest density).
- To mimic this we can propose cascades of pairs of Gaussians (e.g., 1st cascade is composed of pairs that are close to the nucleus).
- More than 90% of candidates are discarded with the first 16 bytes of FREAK descriptor.

FREAK – orientation

$$O=\frac{1}{M}\sum_{P_o\in G}\left(I\left(P_o^{r_1}\right)-I\left(P_o^{r_2}\right)\right)\frac{P_o^{r_1}-P_o^{r_2}}{\|P_o^{r_1}-P_o^{r_2}\|}$$

M – number of pairs in *G*
P – 2D vector indicating the centre of the Gaussian
I – intensity function

FREAK – useful links

FREAK, ORB (oriented BRIEF), BRISK and FAST are implemented in openCV (since 2.4.2)

http://docs.opencv.org/modules/features2d/doc/feature_detection_and_description.html

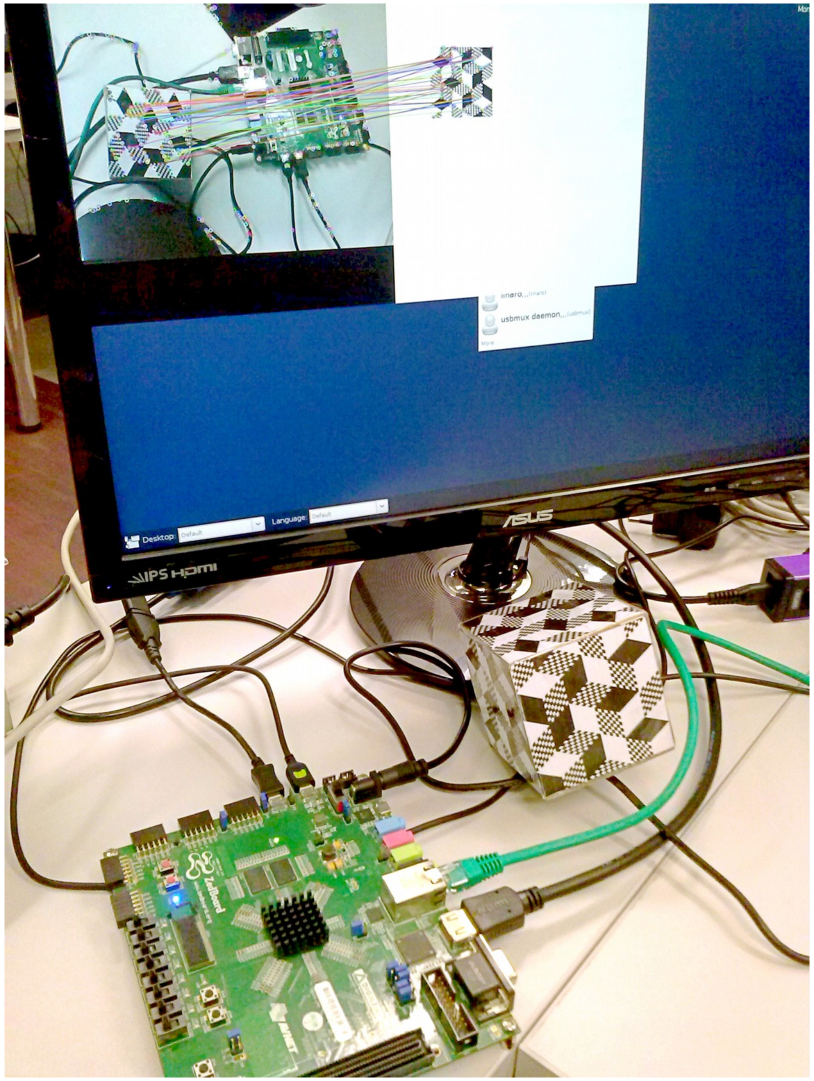
DEMO C++ code:
http://fossies.org/dox/OpenCV-2.4.3/freak__demo_8cpp_source.html

FREAK the article
infoscience.epfl.ch/record/175537/files/2069.pdf

FREAK in MATLAB (using openCV)
http://dovgalecs.com/blog/freak-descriptor-in-matlab/

FREAK – hardware

1. Project requirements
 and assumptions
2. Linux requirements
3. Hardware requirements



Project overview
the ugly truth

1. HW/SW co-design is **hard** - plenty of things to know:
   - software: OS'es specyfics (kernel & rootfs), driver specyfics, application specyfics;
   - hardware: VHDL/Verilog knowledge, hard to debug, some knowledge of the platform you design for is needed
2. It's **not always** reasonable.
3. Hard to get any good **support**.

4. Once it's done it rocks - there is <u>Something</u> to show
5. It shows you're not just another programmer
5. This knowledge is well paid!

Project overview

knowledge coverage - full project

1. Software - Xilinx
    - XPS (professional level)
    - Xilinx IP cores (AXI, FIFO)
2. Software - Linux
    - device drivers
    - kernel porting, rootfs design
3. Software - application
    - IO operations
    - some libs (OpenCV, gstreamer)
3. Hardware
    - Xilinx IP cores (AXI, FIFO)
    - VHDL (quite advanced)
    - simulaiton - Modelsim

Project requirements and assumptions

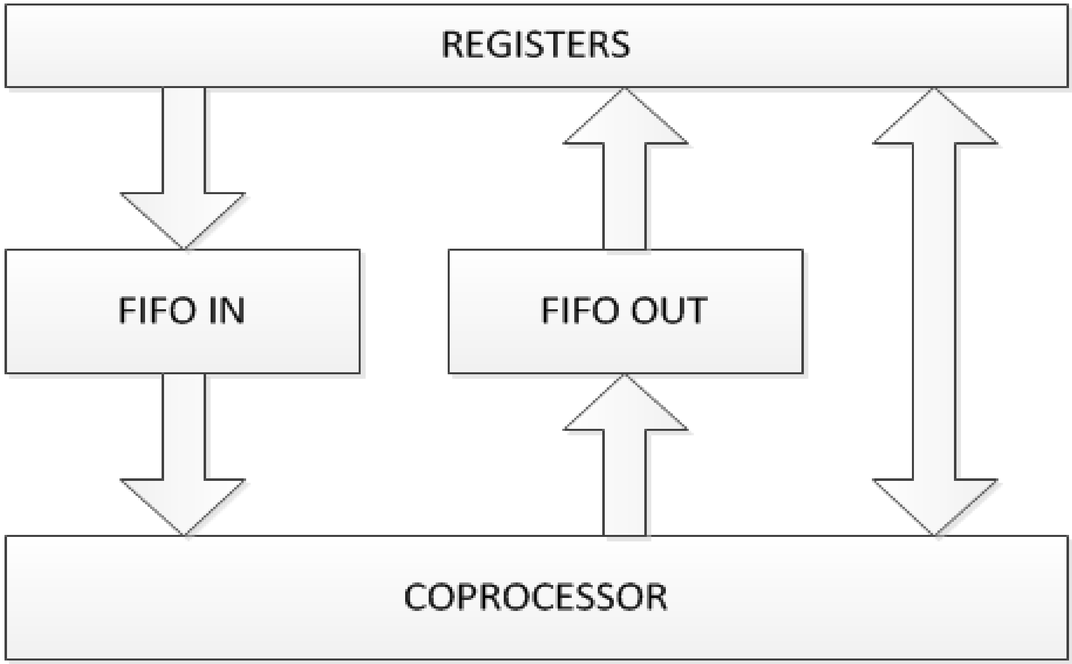
Project requirements

1. ARM processor and hardware accelerator will work simultaneously and simlessly
   - data can be passed back and forth between sw/hw
   - hardware can work simultaneuosly to the software
   - software can be multithreaded

2. ARM processor will run full OS
3. OS will implement hardware driver
4. Driver is supposed to be loadable kernel module
5. Mimimal FPGA area shall be taken for communication channel
6. HW/SW will communicate through the additional signals (control dataflow)

Project - data exchange concepts

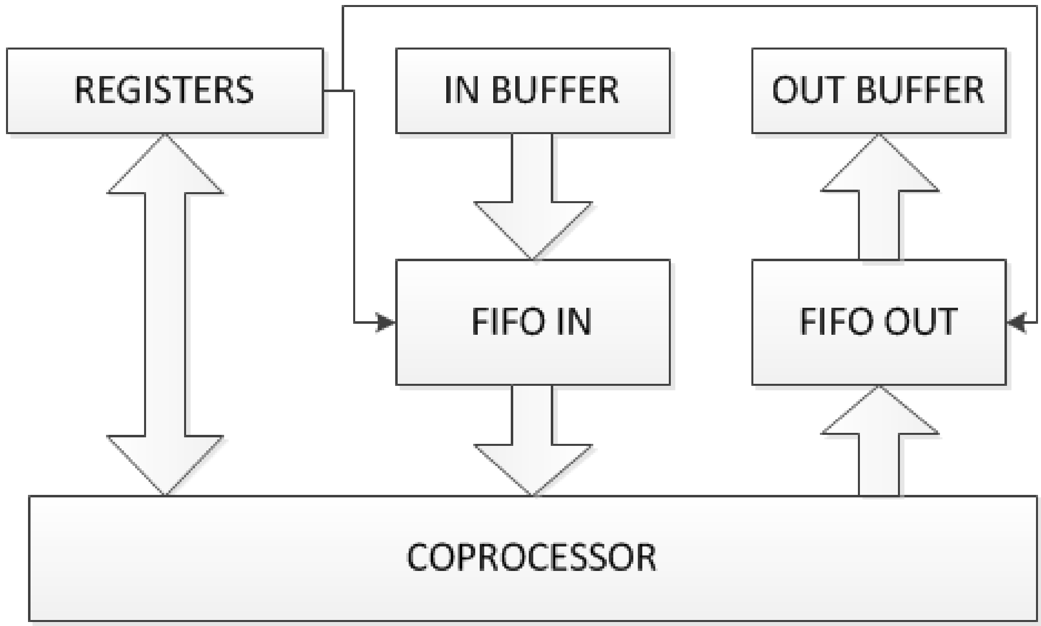
1. **Communication through the processor registers**:
   - processor registers are available from both ARM and FPGA
   - FIFO queues are implemented in FPGA's block RAMs.
   - processor has to handle almost whole communication (i.e., fill and empty regs)
   - ARM is 32-bit (4B) so are the regs -> FIFO width



Project - concepts

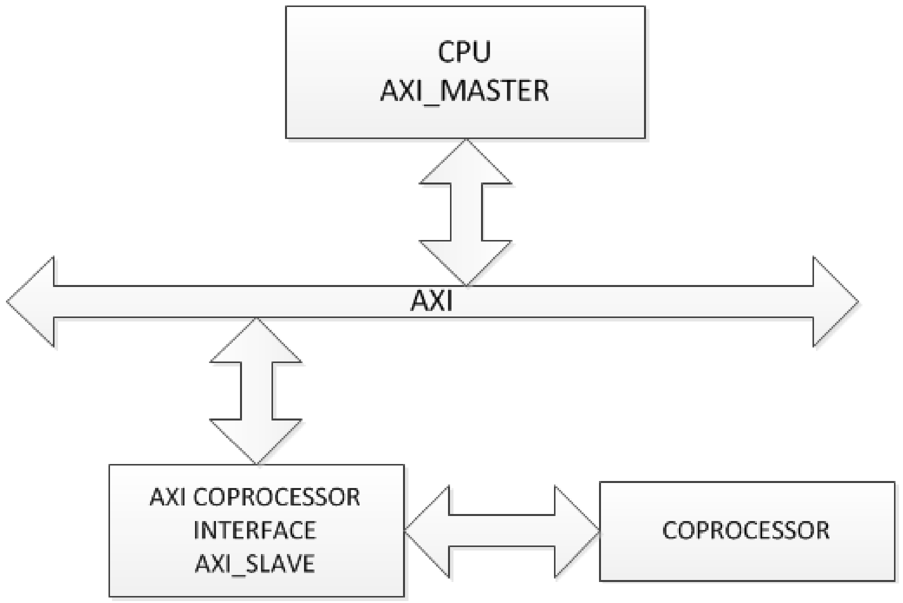
2. **Communication through the processor registers and mapped memory buffers**:

- processor registers are available from both ARM and FPGA
- FIFO queues are implemented in FPGA's block RAMs.
- RAM memory reserved to double FIFO in FPGA
- processor doesn't have to handle the whole communication since it can use DMA now
- ARM is 32-bit (4B) so are the regs -> FIFO width

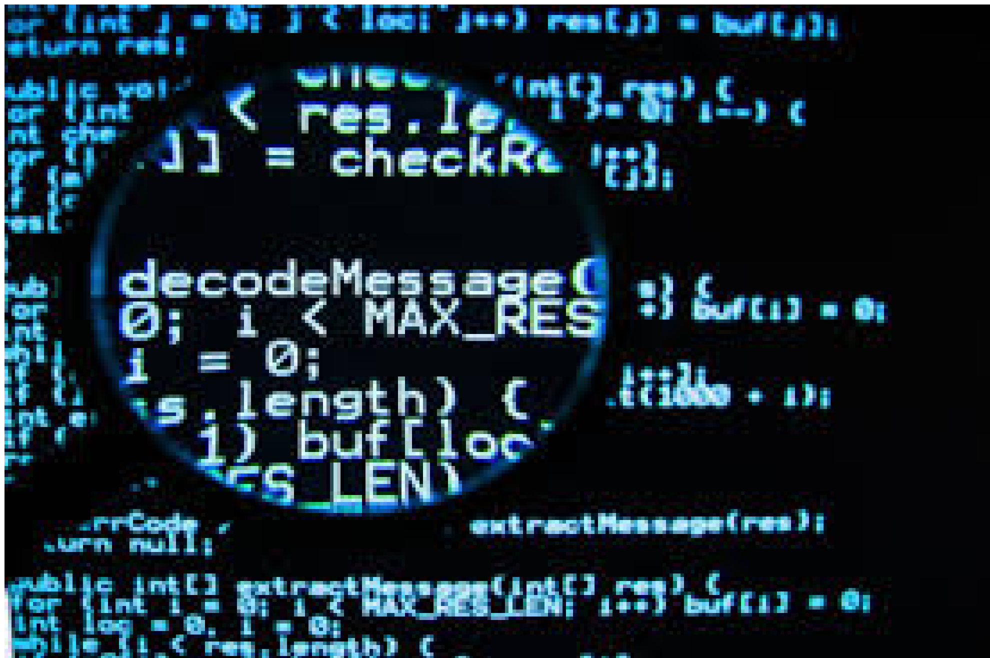


Project - concepts

3. **Coprocessor as the processor's periferial module**:
   - FIFO queues are implemented in FPGA's block RAMs.
   - communication between the cores is done through the AXI (AMBA (Advanced Microcontroller Bus Architecture) eXternal Interface) bus
   - ARM is 32-bit (4B) so are the regs -> FIFO width

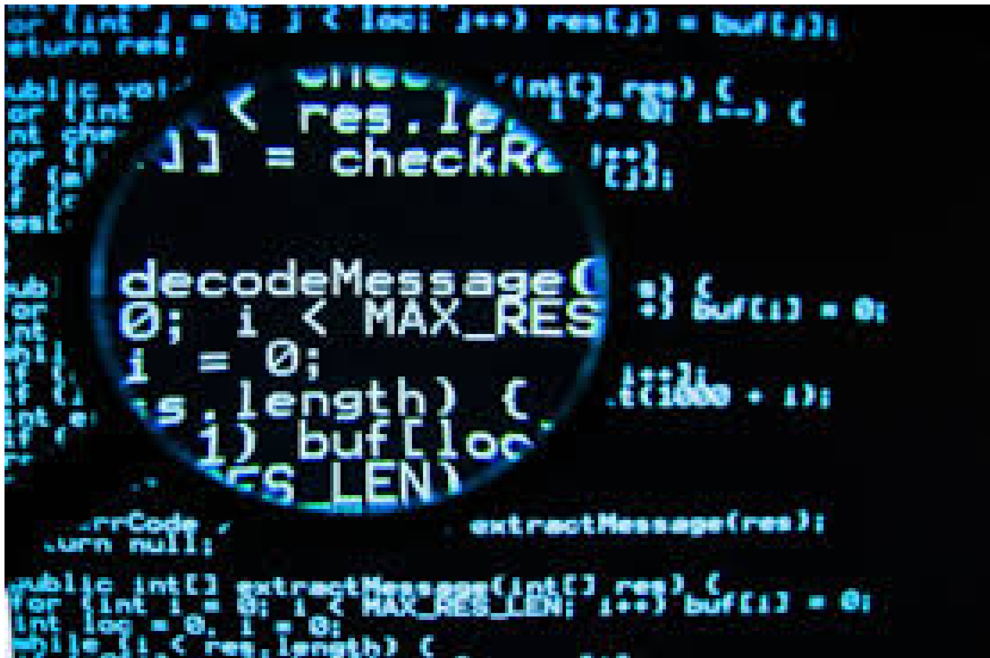


SOFTWARE



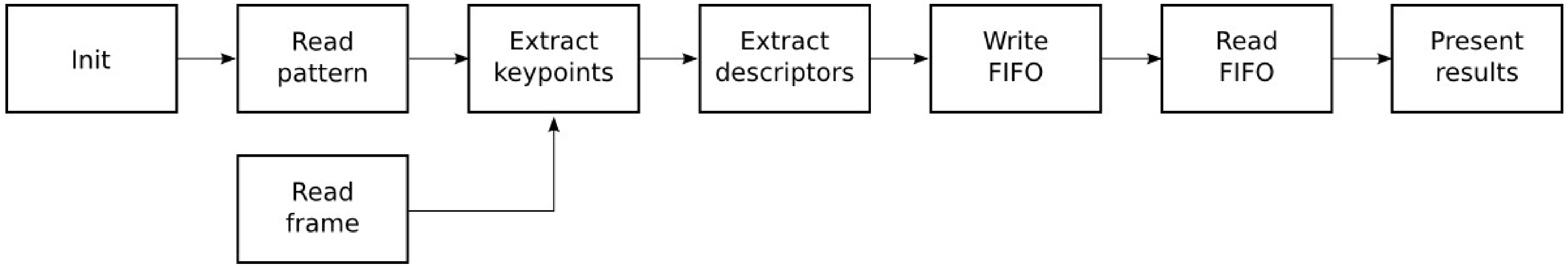
Project - software part (overview)

1. Zynq 7020 has dual core Cortex A9 processor (space for multithreaded applications)
2. Open libs like OpenCV and gstreamer could be used for handling the video frame extraction and preprocessing.
3. FPGA hardware can be used to accelerate some parts of this process.



Project - software part (application)

1. *main.cpp* handles parallelism of the application, gstreamer calls and different operation modes
2. *opencv_alg.cpp* handles all the stuff related to the actual algorithm and communication with the hardware accelarator



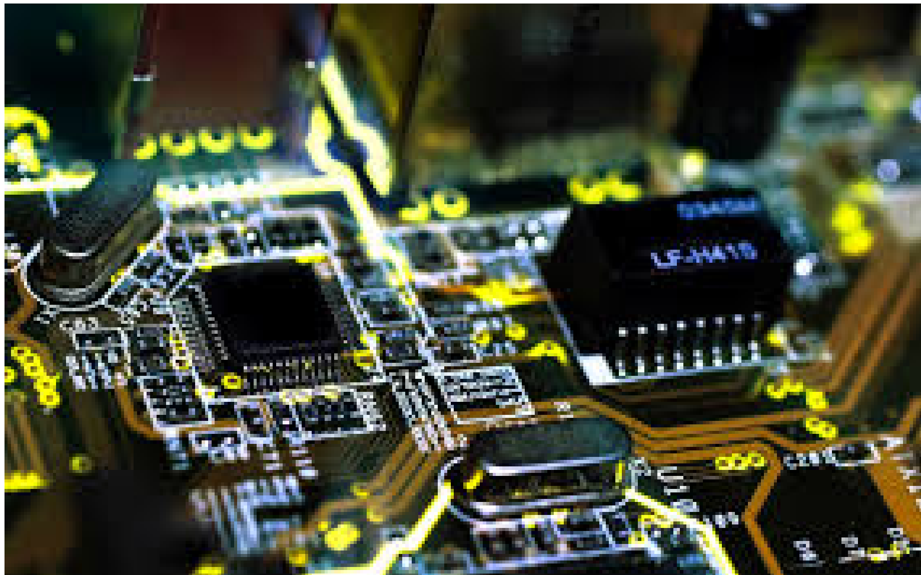
see main.cpp
opencv_alg.cpp

Project - software part (driver)

1. Linux as an OS (open, scalable, secure, etc.)
2. Device configuration taken from device tree.
3. Character device
4. Accessible through the virtual file in /dev/
5. Data transfer through the RW commands to this file
6. State monitoring through the ioctl calls

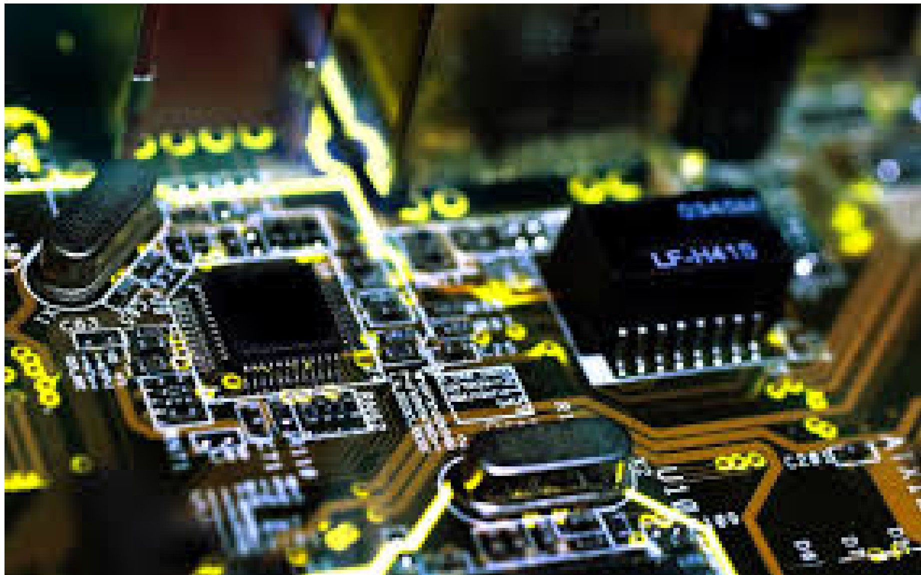
see axi_coprocessor_interface.c
axi_coprocessor_interface.h

HARDWARE



Project - hardware part (overview)

1. Project designed in XPS (Xilinx Platform Studio)
2. AXI and FIFO are Xilinx IP cores
3. User logic is instantiated in the AXI interface on the FPGA end of the AXI bus.
4. AXI bus is “memory mapped” so it is visible as set of adresses for the ARM processor
5. OS kernel needs to see it as well so we need to design a valid DTS



Project - AMBA

based on UG761 by Xilinx

There are three types of AXI4 interfaces:

• AXI4—for high-performance memory-mapped requirements.

• AXI4-Lite—for simple, low-throughput memory-mapped communication (for example, to and from control and status registers).

• AXI4-Stream—for high-speed streaming data.

Project - AMBA

based on UG761 by Xilinx

<u>Productivity</u>

By standardizing on the AXI in terface, developers need to learn only a single protocol for IP.

Project - AMBA

based on UG761 by Xilinx

<u>**Flexibility**</u>

Providing the right protocol for the application:

- **<u>AXI4</u>** is for memory mapped interfaces and allows burst of up to 256 data transfer cycles with just a single address phase.
- **<u>AXI4-Lite</u>** is a light-weight, single transaction memory mapped interface. It has a small logic footprint and is a simple interface to work with both in design and usage.
- **<u>AXI4-Stream</u>** removes the requirement for an address phase altogether and allows unlimited data burst size. AXI4-Stream interfaces and transfers do not have address phases and are therefore not considered to be memory-mapped.

Project - AMBA
based on UG761 by Xilinx

Availability

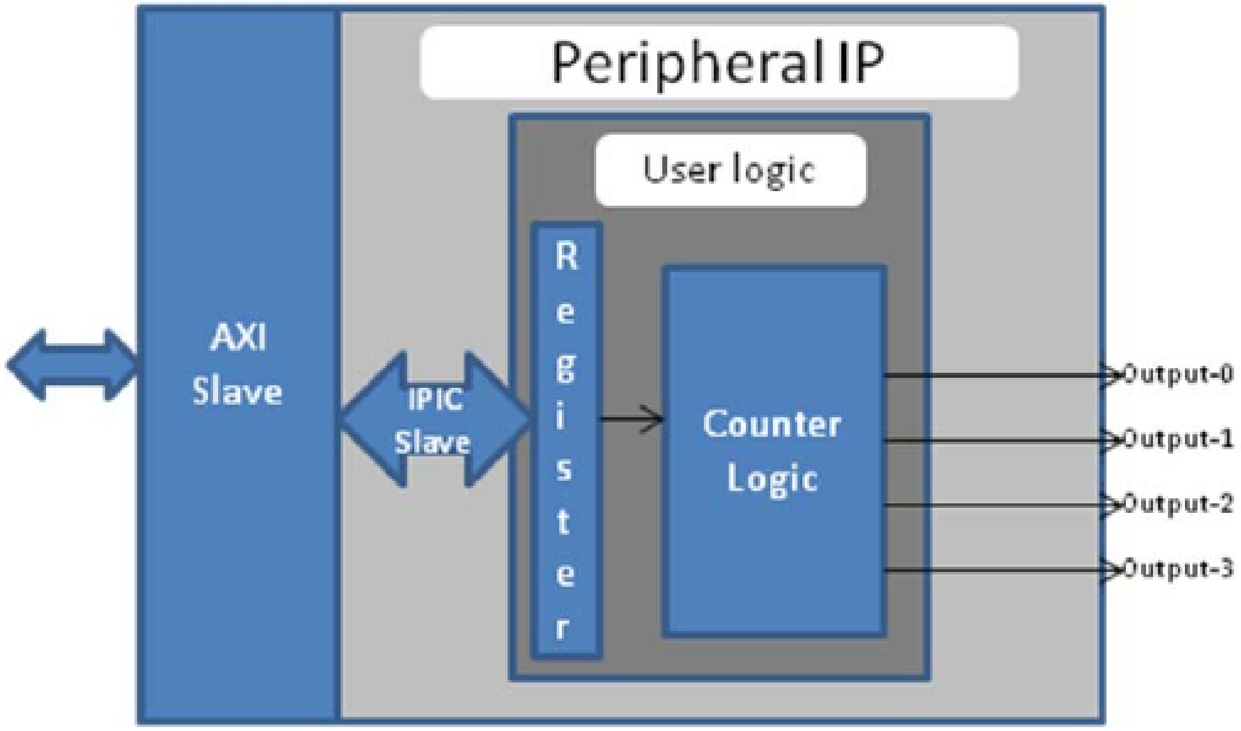
By moving to an industry-standard, you have access not only to the Xilinx IP catalog, but also to a worldwide community of ARM Partners.

- Many IP providers support the AXI protocol.
- A robust collection of third-party AXI tool vendors is available

Project - XPS (example)

based on UG873 by Xilinx

Exemplary IP is AXI4-Lite compliant Slave IP. It includes a 28 bit counter. 4 MSB bits of the counter are driving the 4 output ports of the peripheral IP.



Project - XPS (example)

based on UG873 by Xilinx

Configuration register:

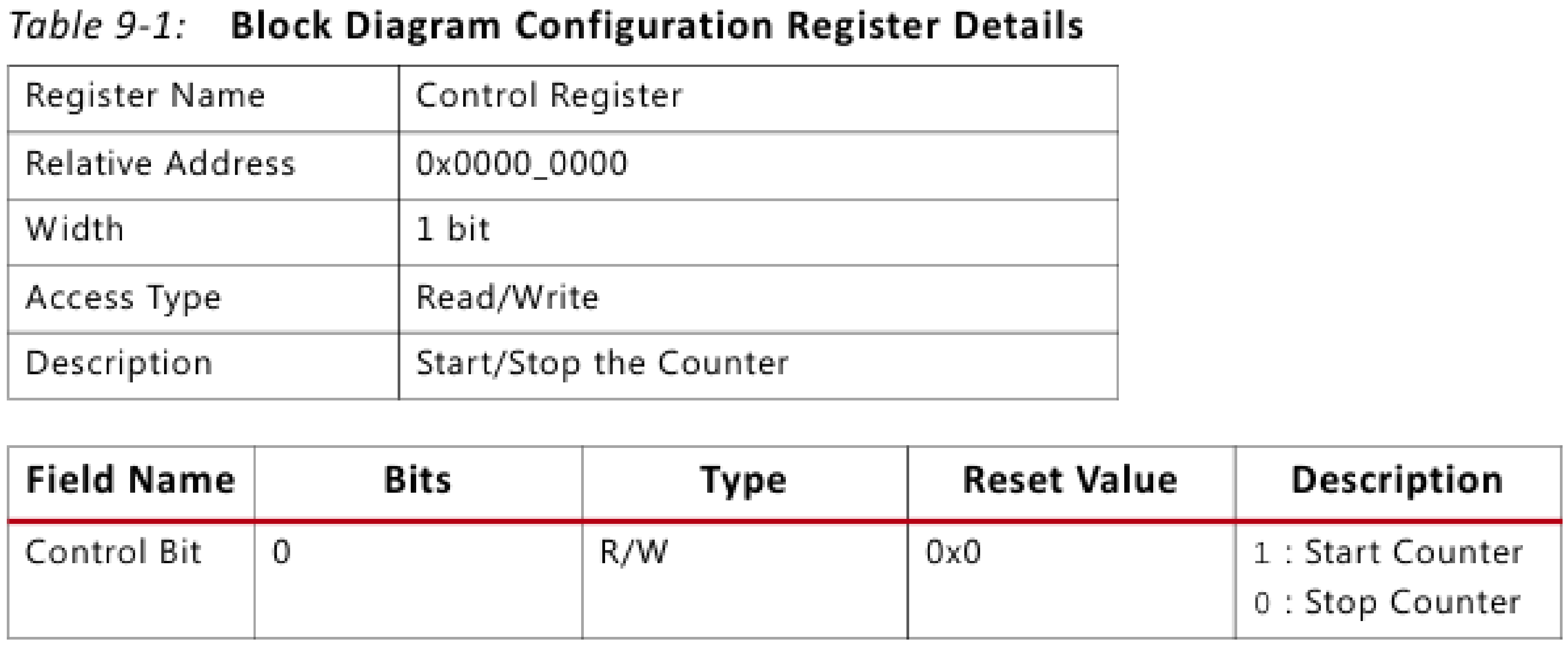


Table 9-1: **Block Diagram Configuration Register Details**

Register Name	Control Register
Relative Address	0x0000_0000
Width	1 bit
Access Type	Read/Write
Description	Start/Stop the Counter

Field Name	**Bits**	**Type**	**Reset Value**	**Description**
Control Bit	0	R/W	0x0	1 : Start Counter 0 : Stop Counter

Project - XPS (example)

based on UG873 by Xilinx

The system covers the following connections:

- Peripheral IP connected to PS General Purpose master port 0 (M_AXI_GP0). This connection is used by the PS CPU to configure Peripheral IP register configurations.
- Four output ports of Peripheral IP connected to DS15, DS16, DS17, and DS18 on-board LEDs.

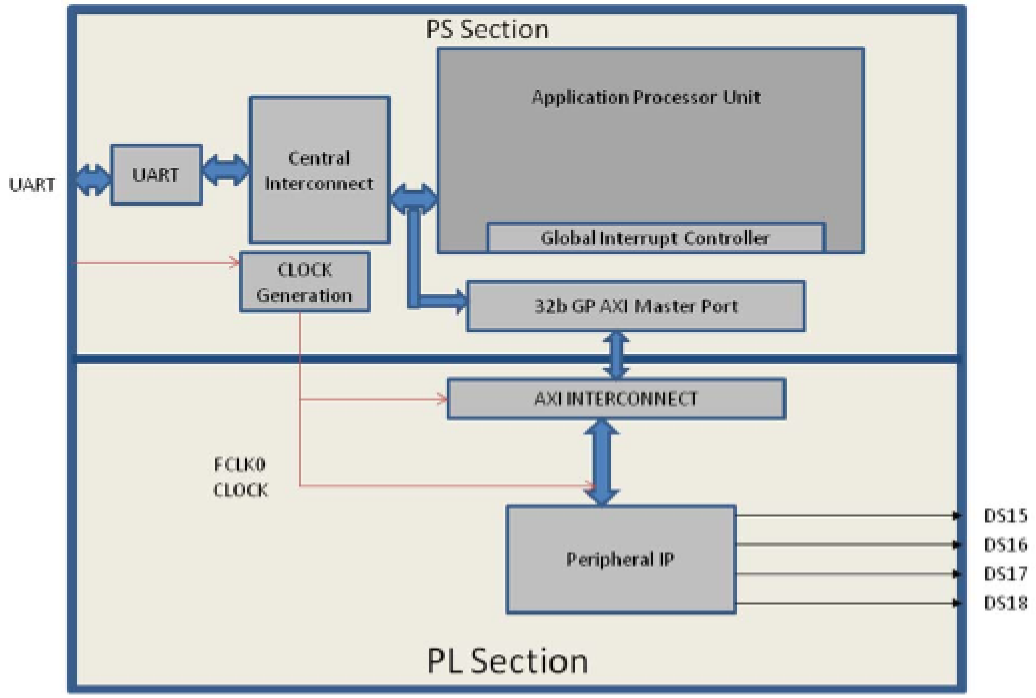
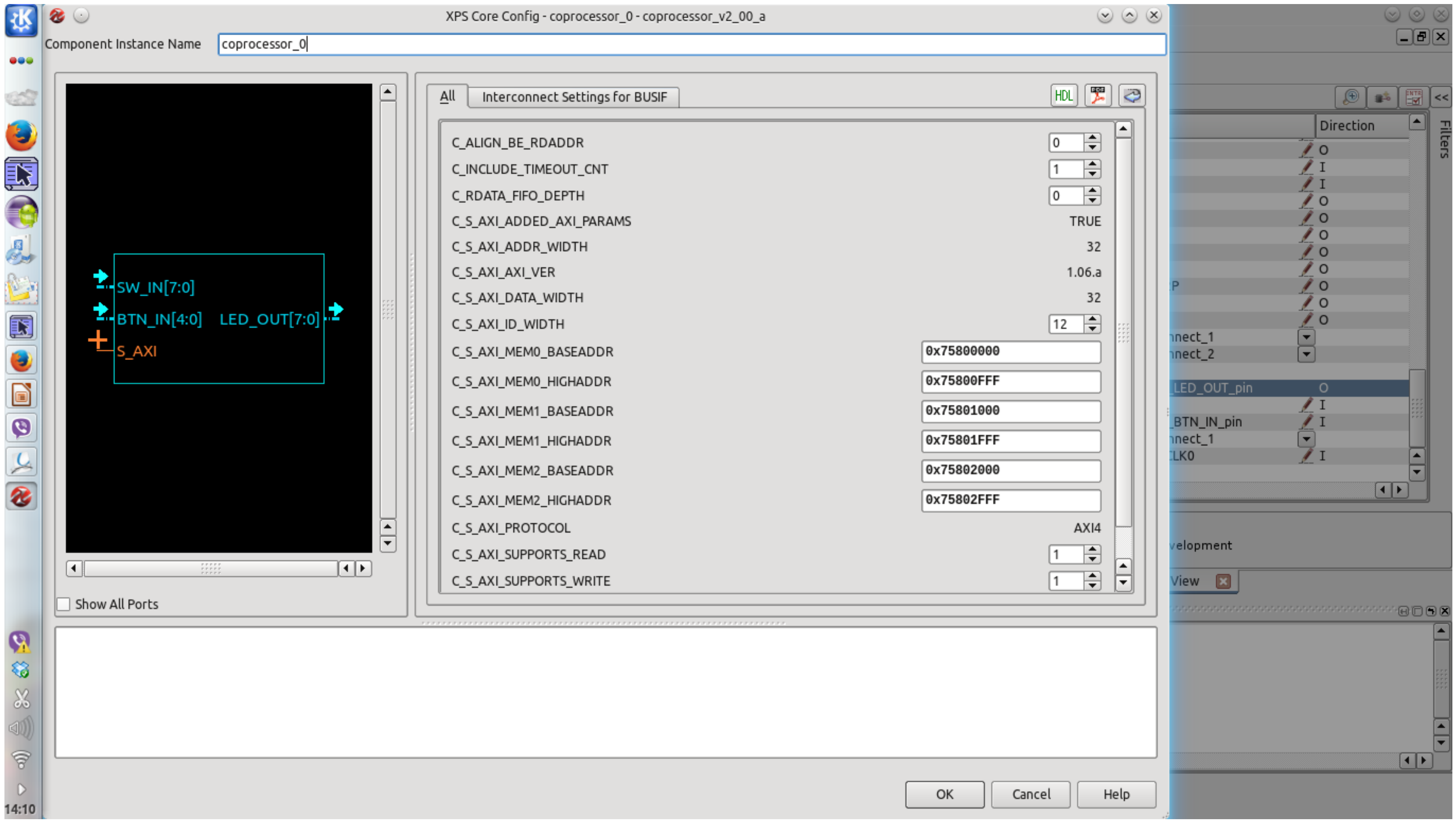


Figure 9-2: **Block Diagram**

Project - XPS (real case)

Port assignment:



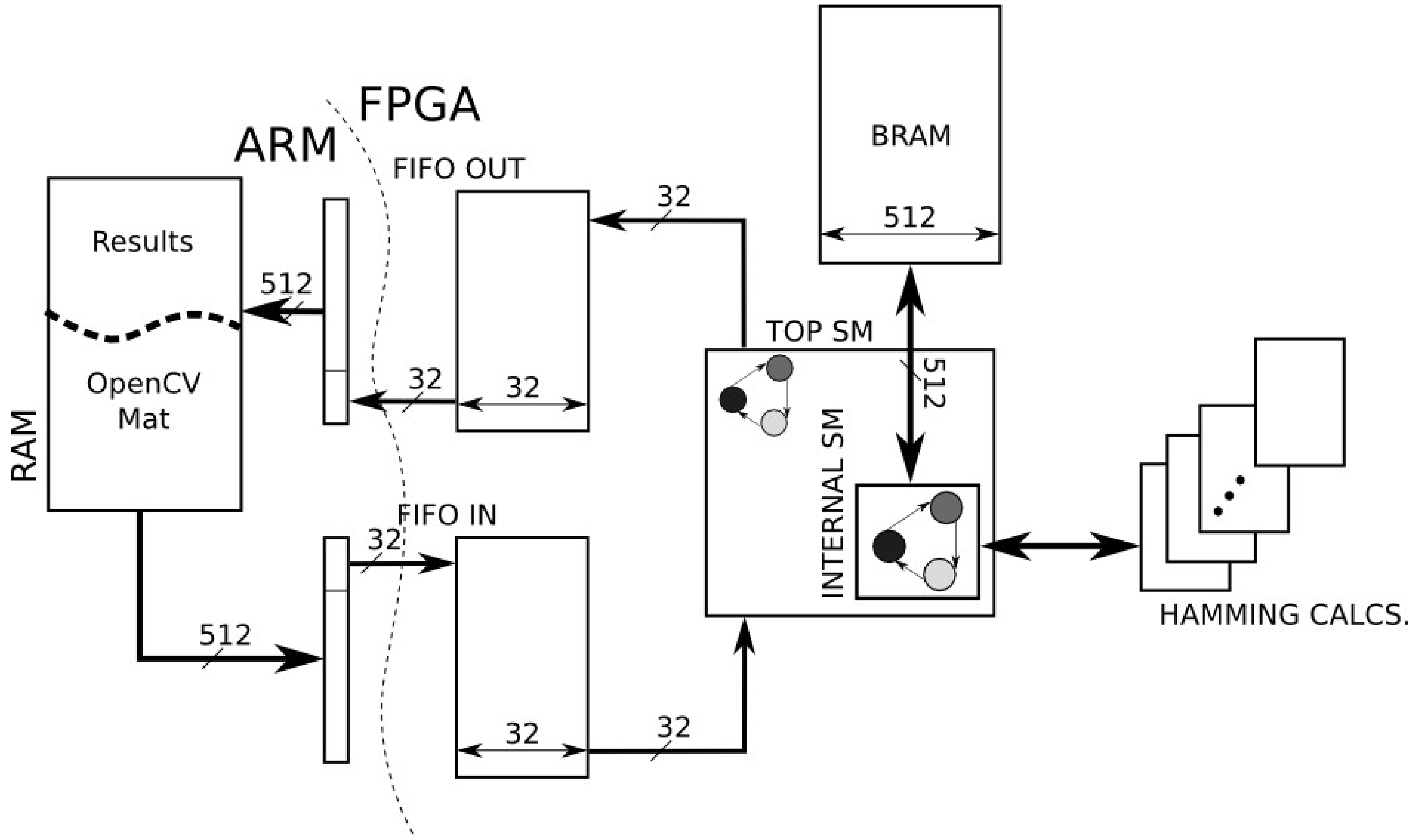
Component Instance Name: coprocessor_0

C_ALIGN_BE_RDADDR	0
C_INCLUDE_TIMEOUT_CNT	1
C_RDATA_FIFO_DEPTH	0
C_S_AXI_ADDED_AXI_PARAMS	TRUE
C_S_AXI_ADDR_WIDTH	32
C_S_AXI_AXI_VER	1.06.a
C_S_AXI_DATA_WIDTH	32
C_S_AXI_ID_WIDTH	12
C_S_AXI_MEM0_BASEADDR	0x75800000
C_S_AXI_MEM0_HIGHADDR	0x75800FFF
C_S_AXI_MEM1_BASEADDR	0x75801000
C_S_AXI_MEM1_HIGHADDR	0x75801FFF
C_S_AXI_MEM2_BASEADDR	0x75802000
C_S_AXI_MEM2_HIGHADDR	0x75802FFF
C_S_AXI_PROTOCOL	AXI4
C_S_AXI_SUPPORTS_READ	1
C_S_AXI_SUPPORTS_WRITE	1

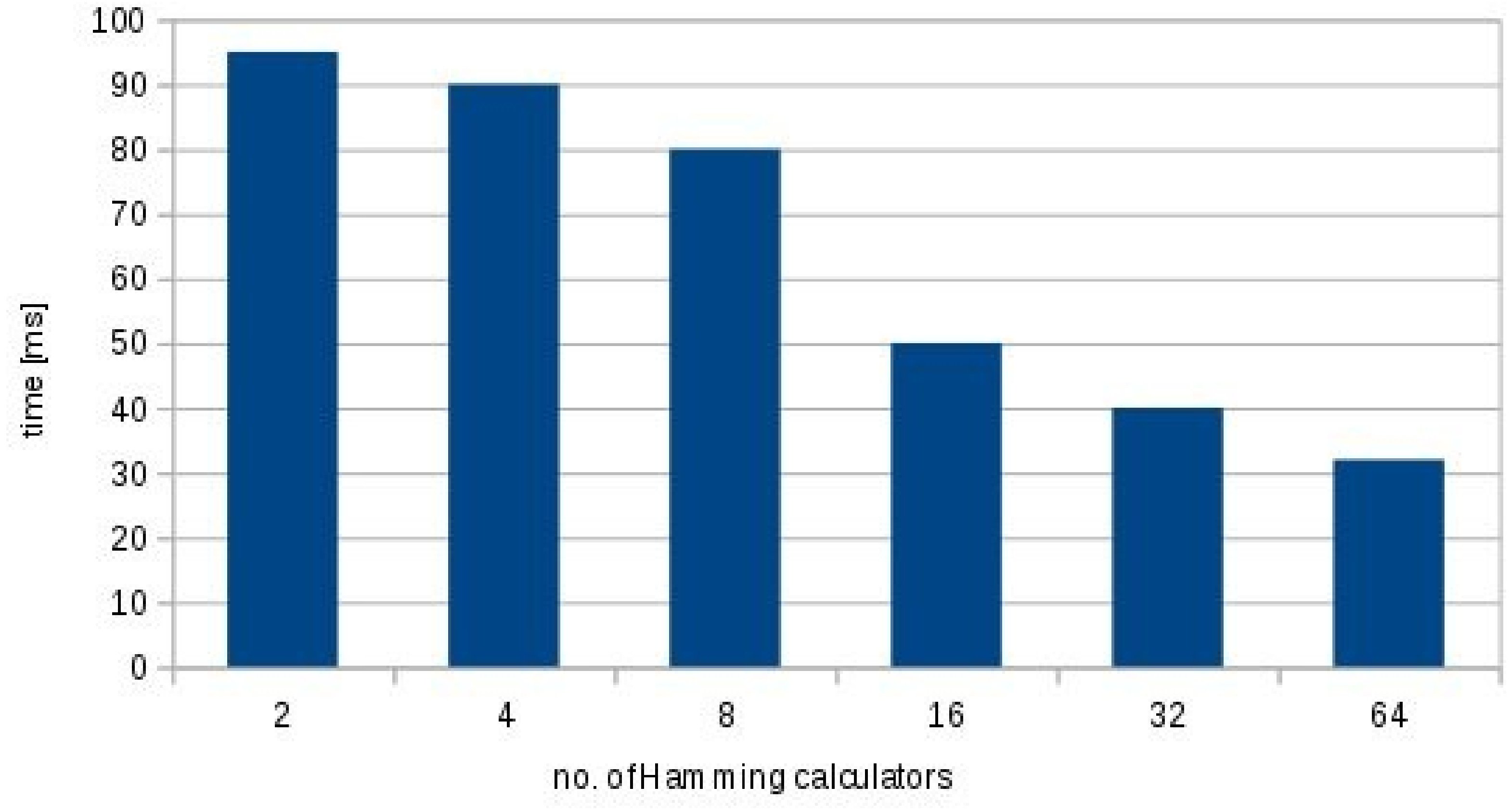
Project - XPS

HDL - IP core

Modelsim



Project - summary
calculation times


time [ms]
no. of Hamming calculators