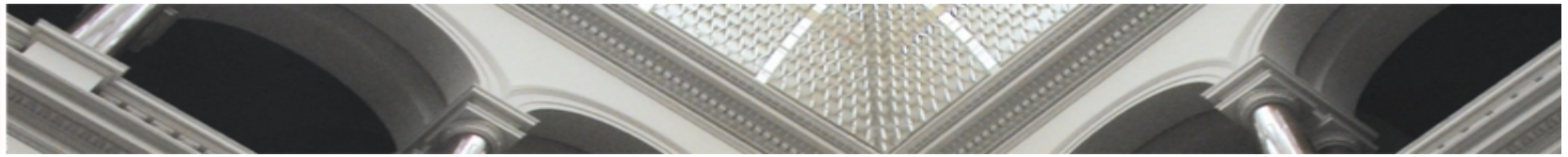
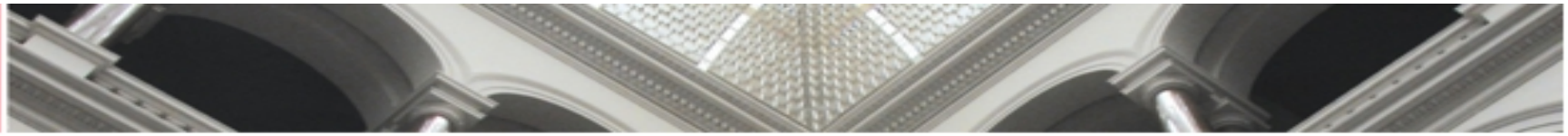


Technische Universität Berlin

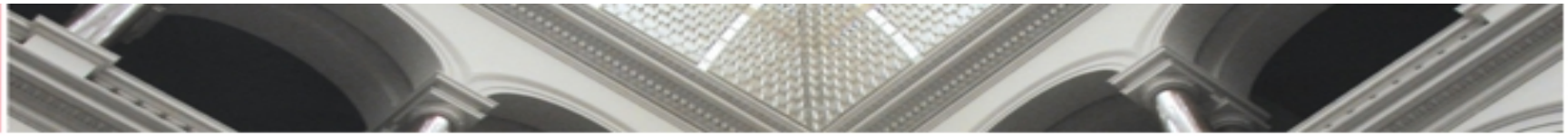
Shared Robo-Vision

Alberto Vaccari | Yu Liu | Dapeng Lan



Technische Universität Berlin

Introduction



Technische Universität Berlin

Introduction

Technische Universität Berlin

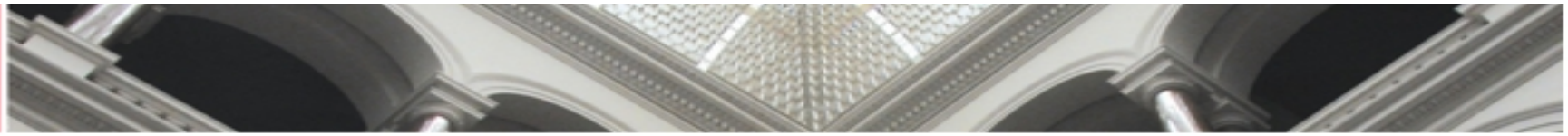
Introduction

RoboCup is an annual international robotics competition with the aim to promote robotics and AI research. This project is trying to create a tool for logging matches and the behavior of robots in the match.

The main idea is to position 2 cameras on each side of the match field. Cameras are connected to Zedboard, after pre-processing with FPGA, images captured by camera will be sent to PC for further handling to detect the positions of robots and ball, record the match and control specific robot.

Technische Universität Berlin

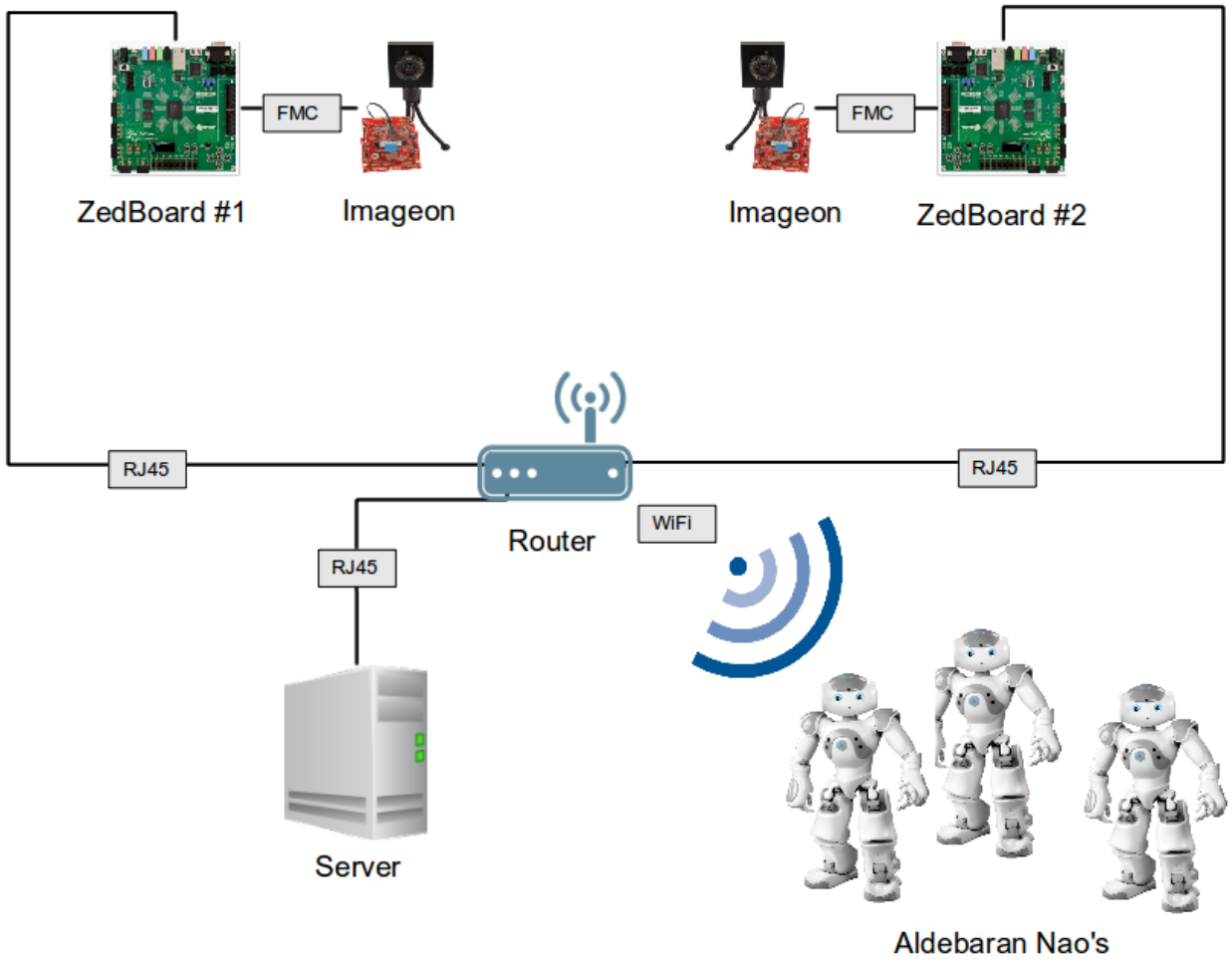
System Diagram

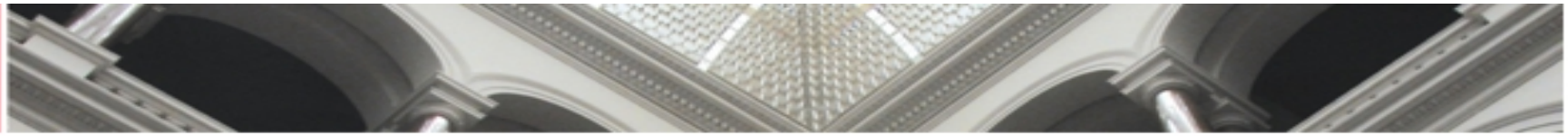


Technische Universität Berlin

System Diagram

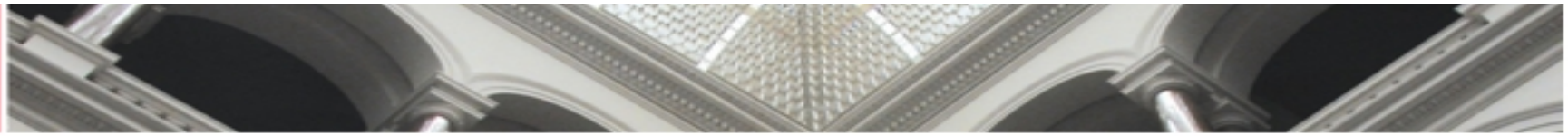
System Diagram





Technische Universität Berlin

Objectives



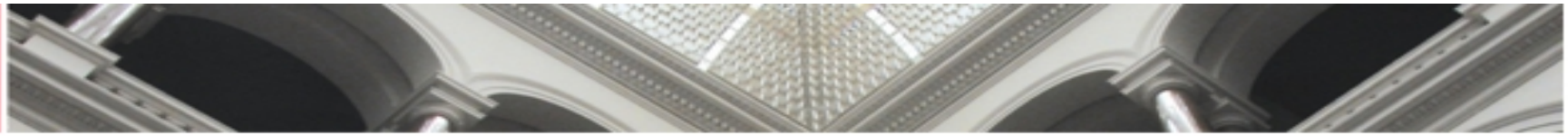
Technische Universität Berlin

Objectives

Technische Universität Berlin

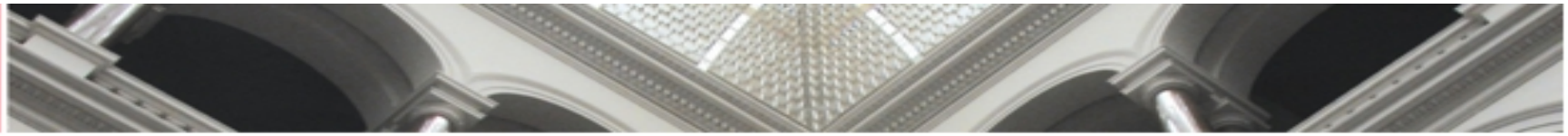
Objectives

- Transmit HD images (1920*1080) from 2 ZedBoards to a computer
- Perform some degree of image processing directly on the ZedBoards
- Track the position of the robots on the field
- Track the position of the ball on the field
- Record a match
- Replay a recorded match
- Control the movements of the robots from the server application



Technische Universität Berlin

Roles



Technische Universität Berlin

Roles

Roles

Alberto:

- ☐ Tracking of the robots
- ☐ Tracking of the ball
- ☐ Recording a match
- ☐ Replaying a saved match
- ☐ Communication and control of the robots

Yu:

- ☐ Receiving images from the ZedBoard
- ☐ Image pre-processing on the ZedBoards

Dapeng:

- ☐ Fetching HD images from FMC-Imageon V2000C camera
- ☐ Transmission of HD images from 2 ZedBoards to the server
- ☐ Image pre-processing on the ZedBoards

Technische Universität Berlin

Roles

Alberto:

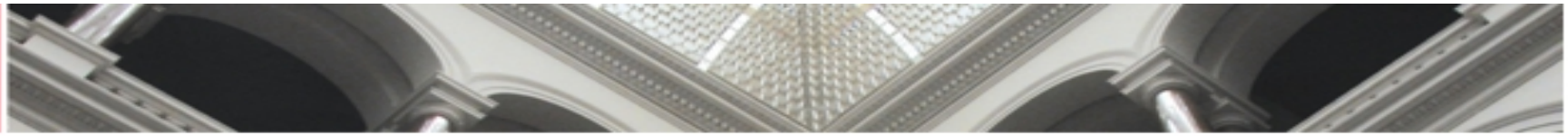
- ☑ Tracking of the robots
- ☑ Tracking of the ball
- ☑ Recording a match
- ☑ Replaying a saved match
- ☑ Communication and control of the robots

Yu:

- ☑ Receiving images from the ZedBoard
- ☐ Image pre-processing on the ZedBoards

Dapeng:

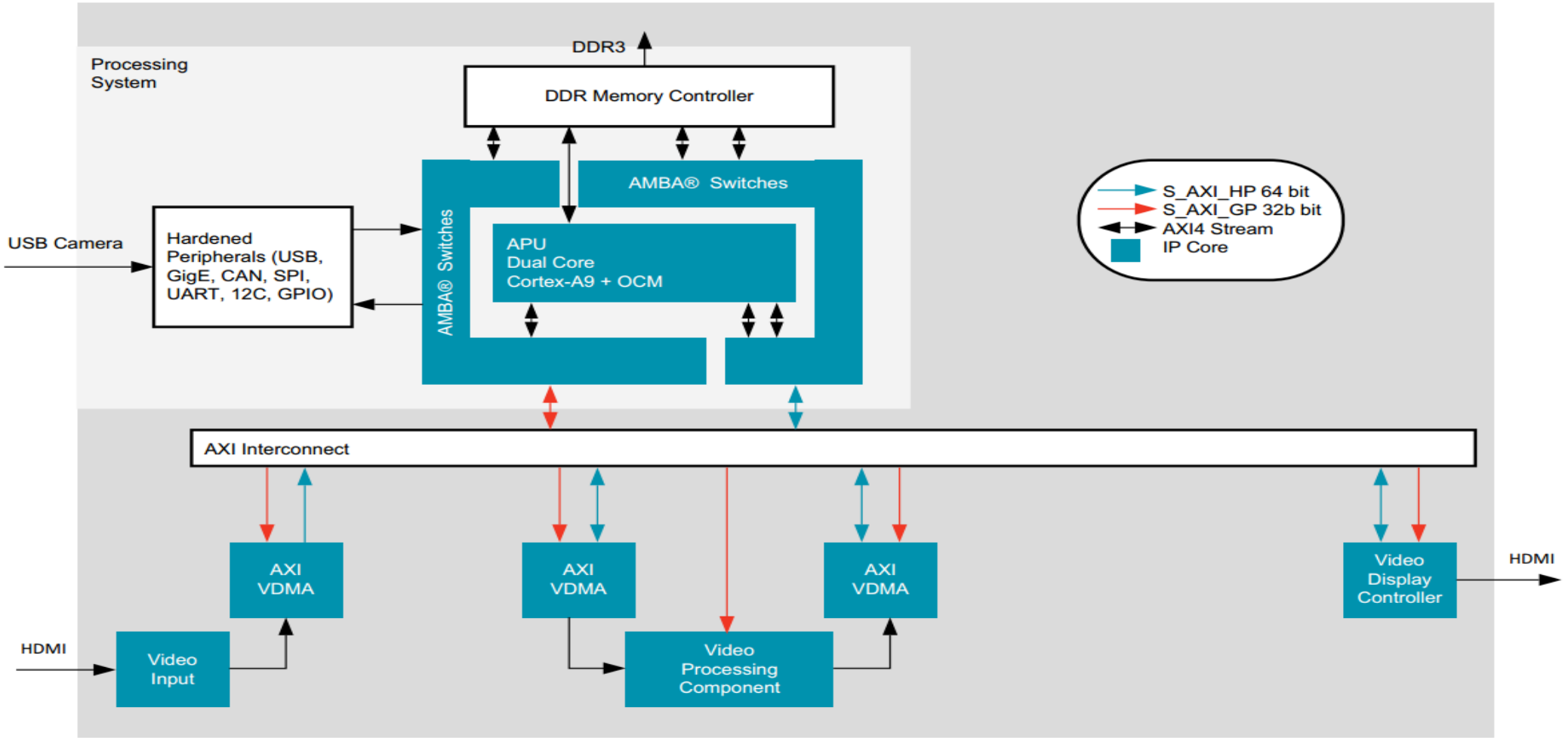
- ☑ Fetching HD images from FMC-Imageon V2000C camera
- ☒ Transmission of HD images from 2 ZedBoards to the server
- ☐ Image pre-processing on the ZedBoards



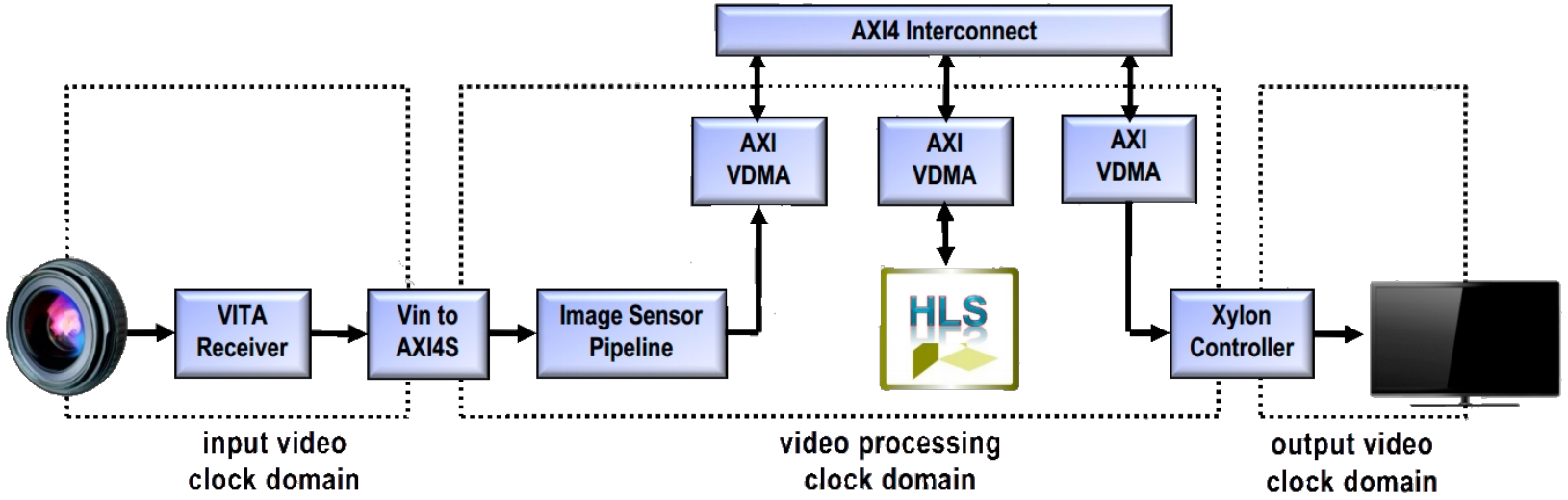
Technische Universität Berlin

Hardware

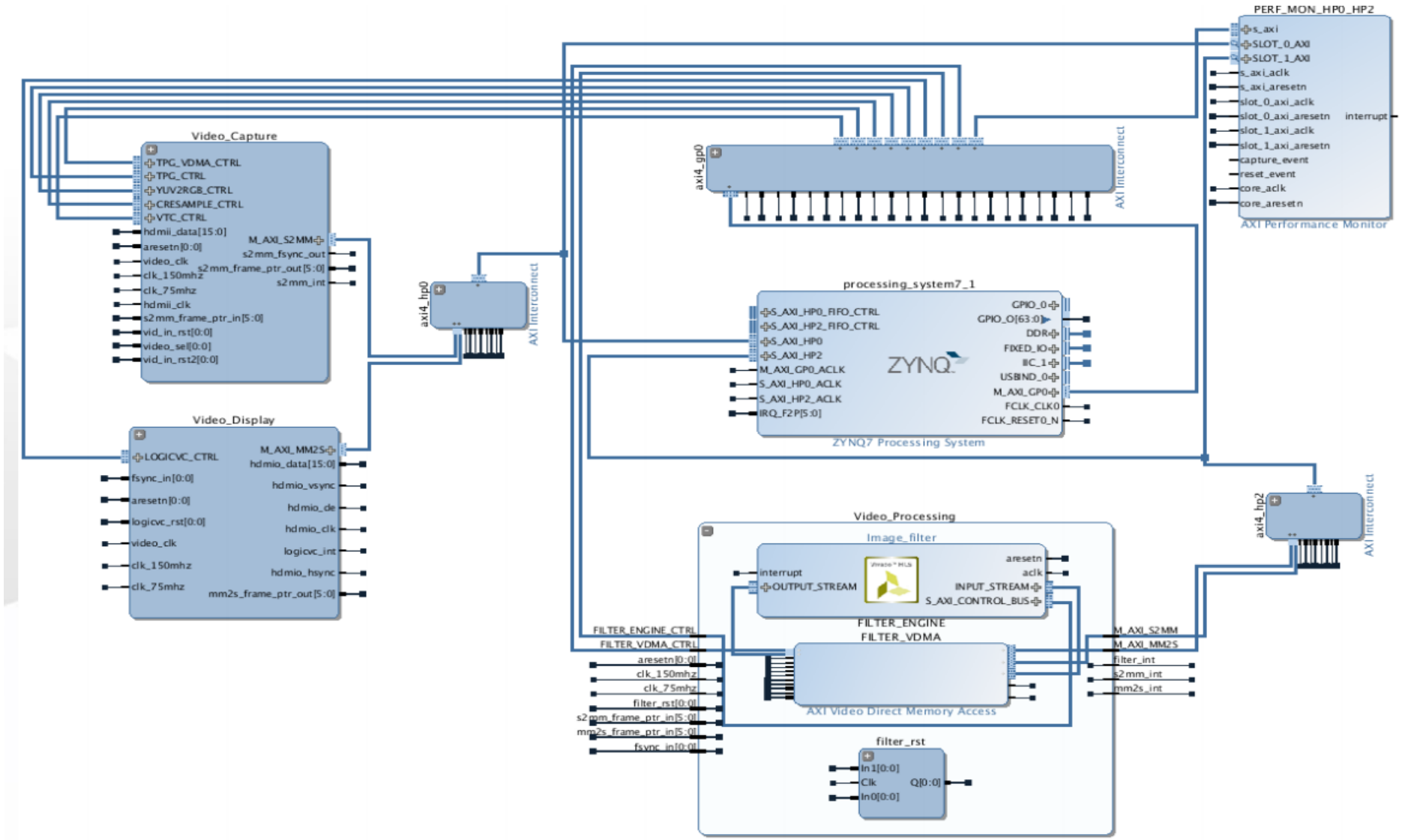
Hardware - structure



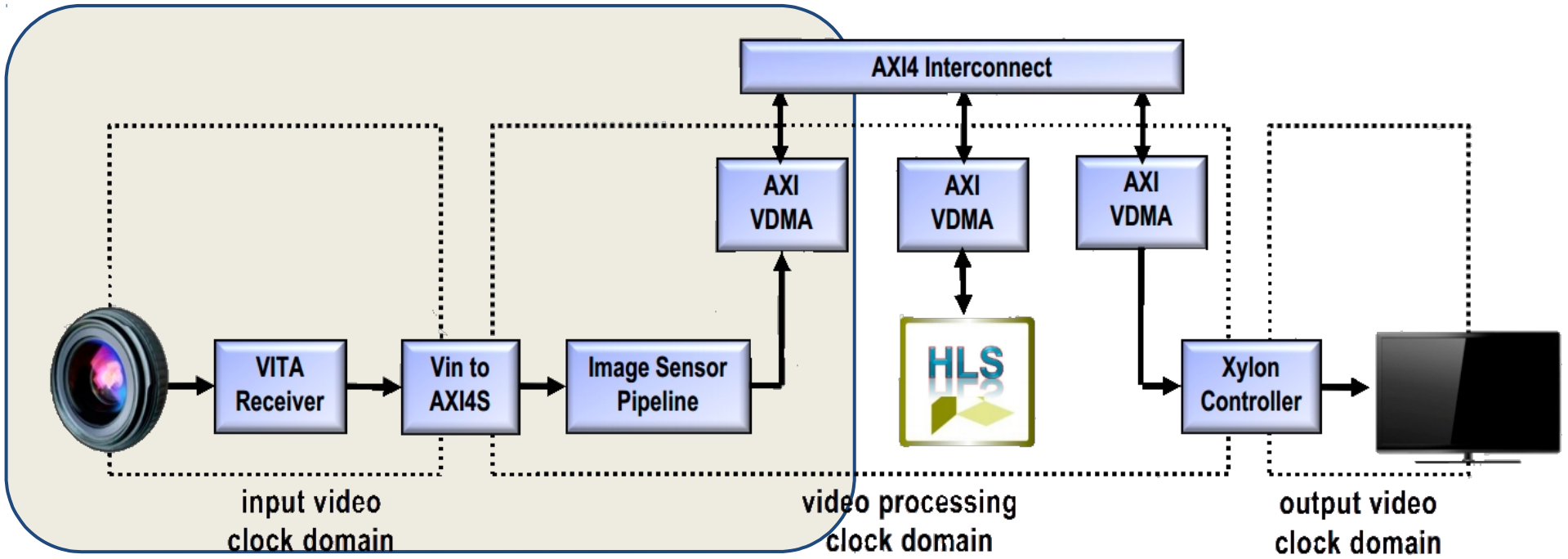
Hardware - structure



- AXI4-Lite for processor control of various IP cores (not shown): **50 MHz**
- AXI4-Stream for point-to-point video connections :**148.5MHz**
- AXI4 for high-performance access to external memory


PERF_MON_HP0_HP2
s_axi
SLOT_0_AXI
SLOT_1_AXI
s_axi_aclk
s_axi_aresetn
slot_0_axi_aclk
slot_0_axi_aresetn
slot_1_axi_aclk
slot_1_axi_aresetn
capture_event
reset_event
core_aclk
core_aresetn
interrupt
AXI Performance Monitor
Video_Capture
TPG_VDMA_CTRL
TPG_CTRL
YUV2RGB_CTRL
CRESAMPLE_CTRL
VTC_CTRL
hdmii_data[15:0]
aresetn[0:0]
video_clk
clk_150mhz
clk_75mhz
hdmii_clk
s2mm_frame_ptr_in[5:0]
vid_in_rst[0:0]
video_sel[0:0]
vid_in_rst2[0:0]
M_AXI_S2MM
s2mm_fsync_out
s2mm_frame_ptr_out[5:0]
s2mm_int
axi4_gp0
AXI Interconnect
axi4_hp0
AXI Interconnect
processing_system7_1
S_AXI_HP0_FIFO_CTRL
S_AXI_HP2_FIFO_CTRL
S_AXI_HP0
S_AXI_HP2
M_AXI_GP0_ACLK
S_AXI_HP0_ACLK
S_AXI_HP2_ACLK
IRQ_F2P[5:0]
ZYNQ
GPIO_0
GPIO_O[63:0]
DDR
FIXED_IO
IIC_1
USBIND_0
M_AXI_GP0
FCLK_CLK0
FCLK_RESET0_N
ZYNQ7 Processing System
Video_Display
LOGICVC_CTRL
fsync_in[0:0]
aresetn[0:0]
logicvc_rst[0:0]
video_clk
clk_150mhz
clk_75mhz
M_AXI_MM2S
hdmio_data[15:0]
hdmio_vsync
hdmio_de
hdmio_clk
logicvc_int
hdmio_hsync
mm2s_frame_ptr_out[5:0]
axi4_hp2
AXI Interconnect
Video_Processing
Image_filter
interrupt
OUTPUT_STREAM
aresetn
aclk
INPUT_STREAM
S_AXI_CONTROL_BUS
FILTER_ENGINE
FILTER_VDMA
AXI Video Direct Memory Access
FILTER_ENGINE_CTRL
FILTER_VDMA_CTRL
aresetn[0:0]
clk_150mhz
clk_75mhz
filter_rst[0:0]
s2mm_frame_ptr_in[5:0]
mm2s_frame_ptr_in[5:0]
fsync_in[0:0]
M_AXI_S2MM
M_AXI_MM2S
filter_int
s2mm_int
mm2s_int
filter_rst
In1[0:0]
Clk
In0[0:0]
Q[0:0]

Hardware – Image Capture

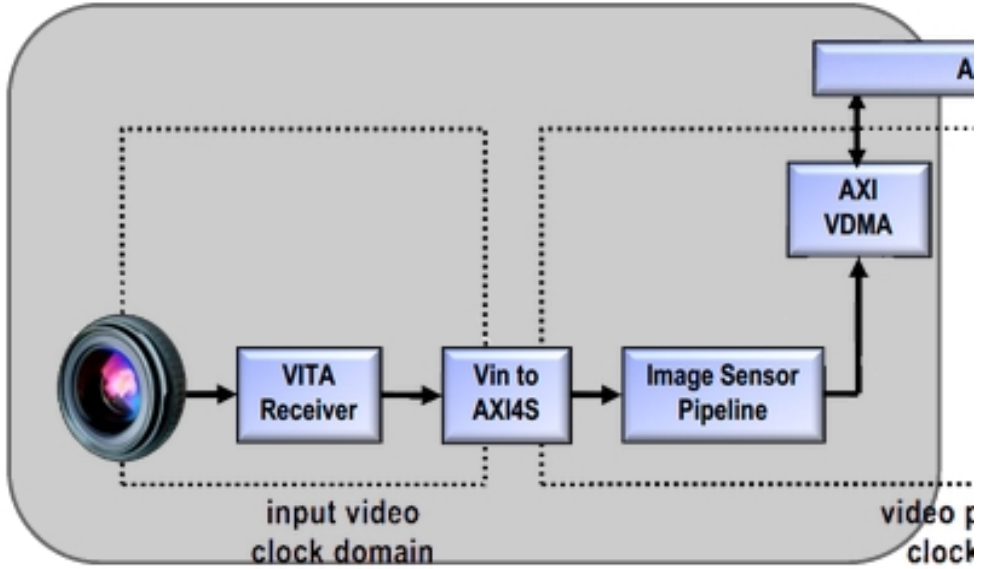


Hardware – Image Capture

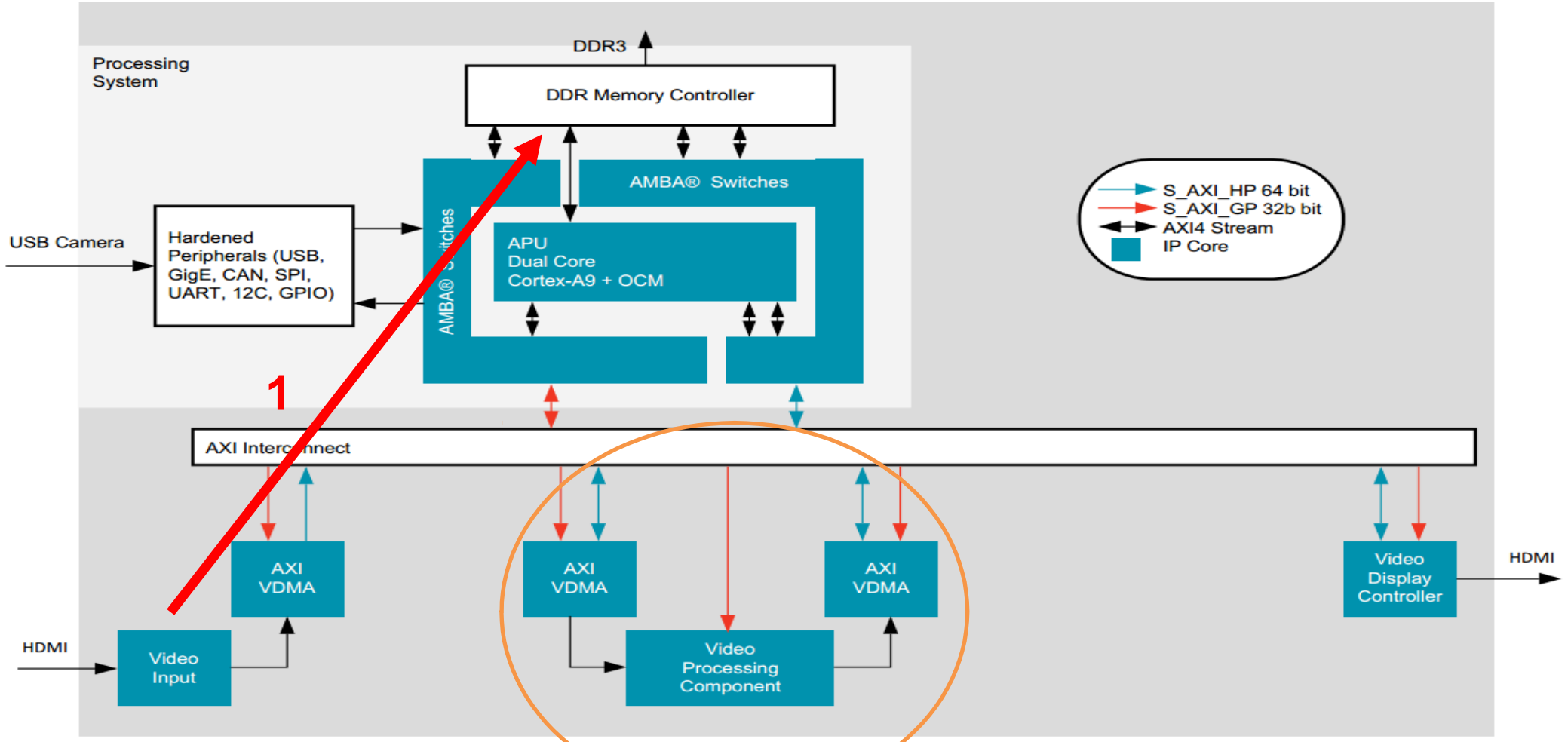
The function of this part is:

Add FMC-IMAGEON cores to IP Catalog
Create Video Pipeline
• VITA Receiver core + Video to AXI4-Stream bridge
• Image Processing cores : DPC, CFA
• AXI Video DMA

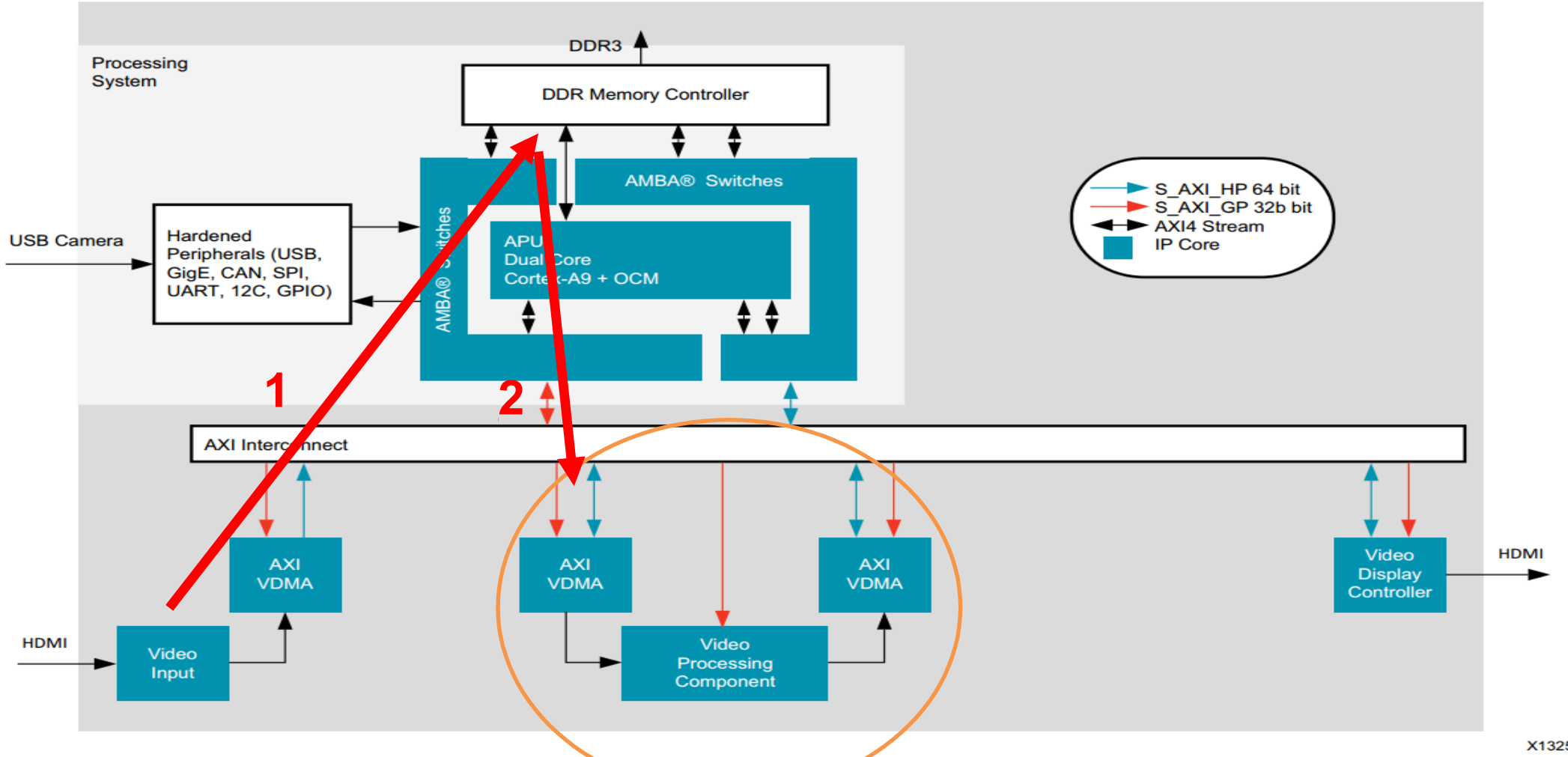
DPC: Defect Pixel Correction
CFA: Colour Filter Interpolation
VDMA: video direct memory access
• Three frame buffers in circular mode
• Tdata widths of 24 bits



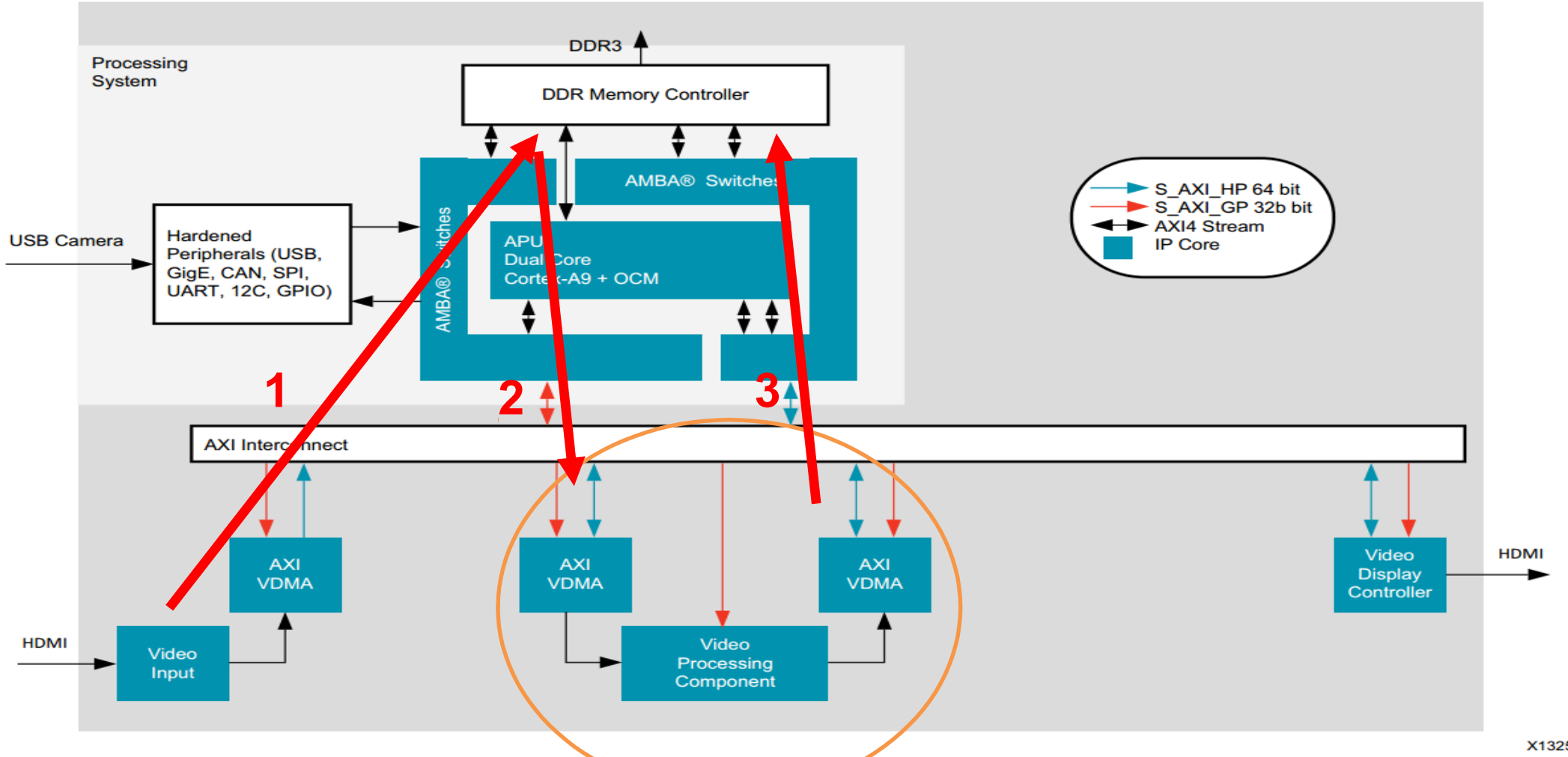
Hardware – Image process



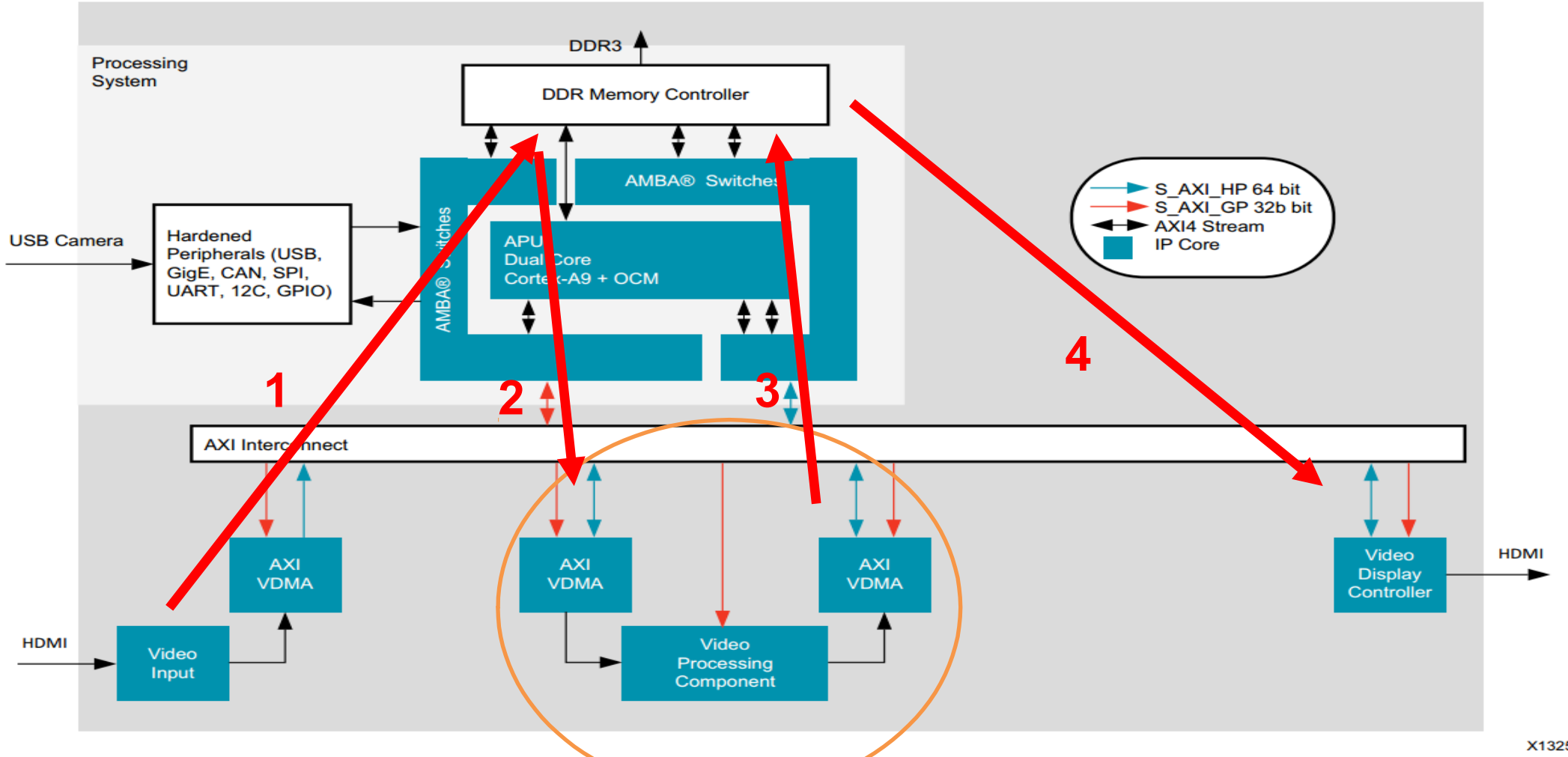
Hardware – Image process

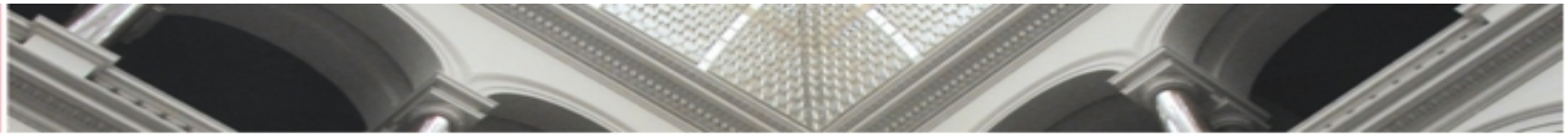


Hardware – Image process



Hardware – Image process





Technische Universität Berlin

Hardware

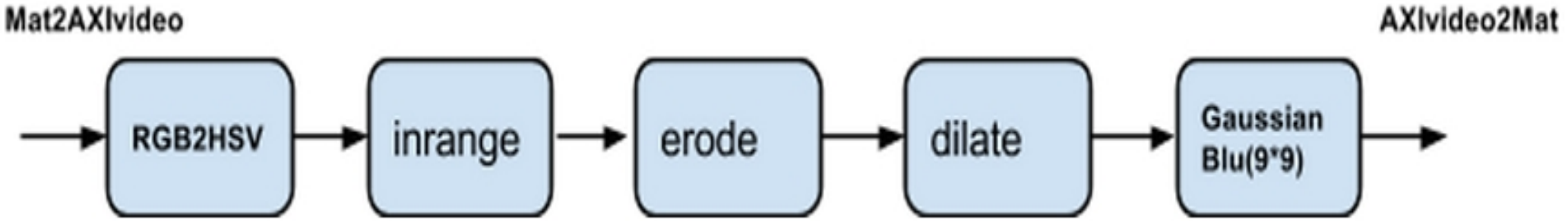
Video Processing in Vivado HLS

These libraries are implemented as synthesizable C++ code and roughly correspond to video processing functions and data structures implemented in OpenCV.

OpenCV	HLS Video Library
`cv::Point_<T>, CvPoint`	`hls::Point_<T>, hls::Point`
`cv::Size_<T>, CvSize`	`hls::Size_<T>, hls::Size`
`cv::Rect_<T>, CvRect`	`hls::Rect_<T>, hls::Rect`
`cv::Scalar_<T>, CvScalar`	`hls::Scalar<N, T>`
`cv::Mat, IplImage, CvMat`	`hls::Mat<ROWS, COLS, T>`
`cv::Mat mat(rows, cols, CV_8UC3);`	`hls::Mat<ROWS, COLS, HLS_8UC3> mat (rows, cols);`
`IplImage* img = cvCreateImage(cvSize(cols,rows), IPL_DEPTH_8U, 3);`	`hls::Mat<ROWS, COLS, HLS_8UC3> img (rows, cols);`
	`hls::Mat<ROWS, COLS, HLS_8UC3> img;`
	`hls::Window<ROWS, COLS, T>`
	`hls::LineBuffer<ROWS, COLS, T>`

Hardware

Pre-image processing



Hardware

BGR2HSV algorithm

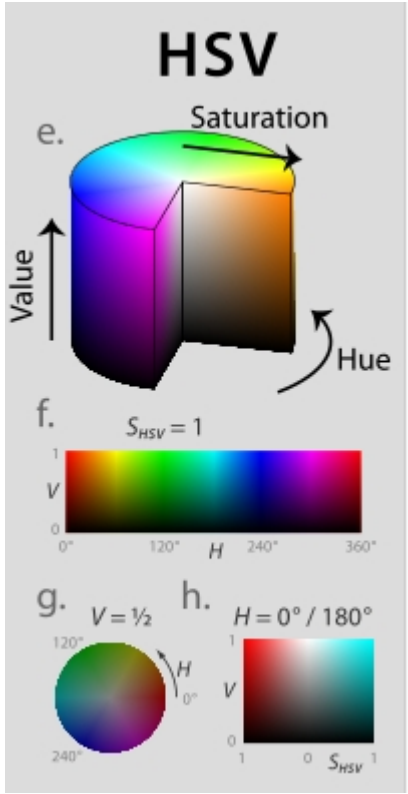
$$MAX = \max(Red, Green, Blue)$$

$$MIN = \min(Red, Green, Blue)$$

$$H = \begin{cases} 0 & if \quad MAX = MIN \\ 42 \times \frac{Green - Blue}{MAX - MIN} + 42 & if \quad MAX = Red \\ 42 \times \frac{Blue - Red}{MAX - MIN} + 127 & if \quad MAX = Green \\ 42 \times \frac{Red - Green}{MAX - MIN} + 213 & if \quad MAX = Blue \end{cases}$$

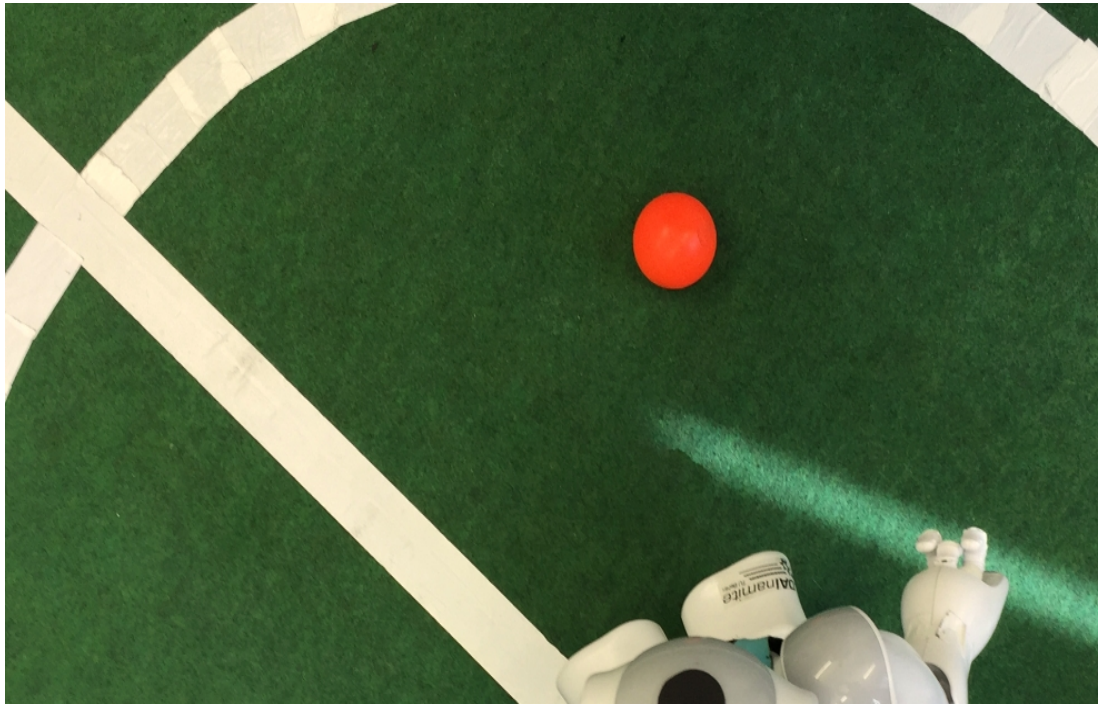
$$S = \begin{cases} 0 & if \quad MAX = 0 \\ 255 \times \frac{MAX - MIN}{MAX} & Otherwise \end{cases}$$

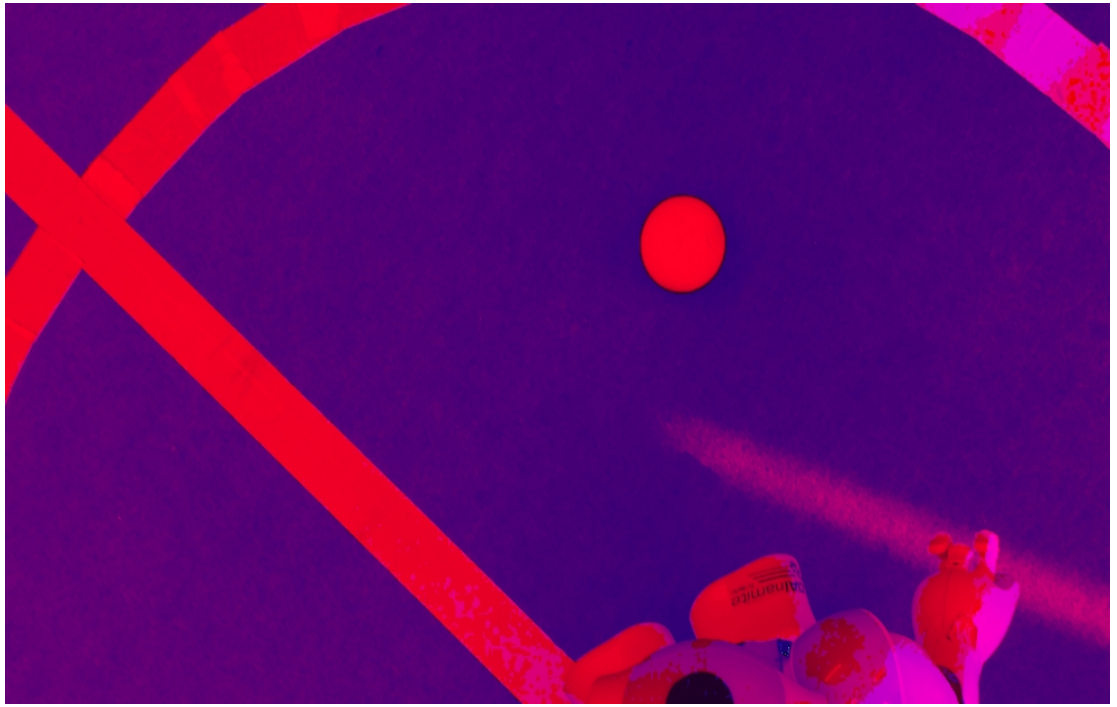
$$V = MAX$$



https://en.wikipedia.org/wiki/HSL_and_HSV

Hardware - Results



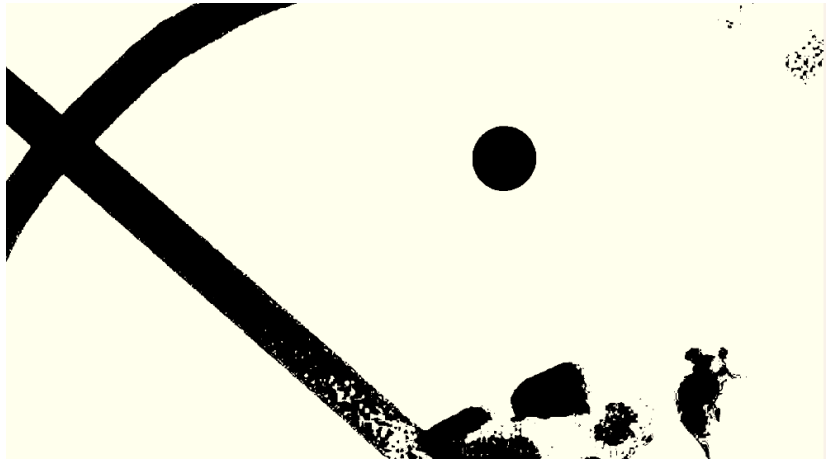


Video1: rgb2hsv https://youtu.be/gzL8fef96NY

Hardware - Results

inRange

```
unsigned int xmin_temp = cols − 1;
unsigned int xmax_temp = 0;
unsigned int ymin_temp = rows − 1;
unsigned int ymax_temp = 0;
temp = H − Hin;
if(S > 63 &V > 127){
   if (ABS(temp) < Threshold){
      if (i < xmin_temp) xmin_temp = i;
      if (i > xmax_temp) xmax_temp = i;
      if (j < ymin_temp) ymin_temp = j;
      if (j > ymax_temp) ymax_temp = j;}
}
```



We **compare** the HSV values of the pixel with **an input value (Hin)** chosen by the user. If H is close enough to that input value and S and V are big enough, then the pixel will be tracked.

Here we choose the color red

Hardware

Following using the IP core from Vivado HLS:

Erode

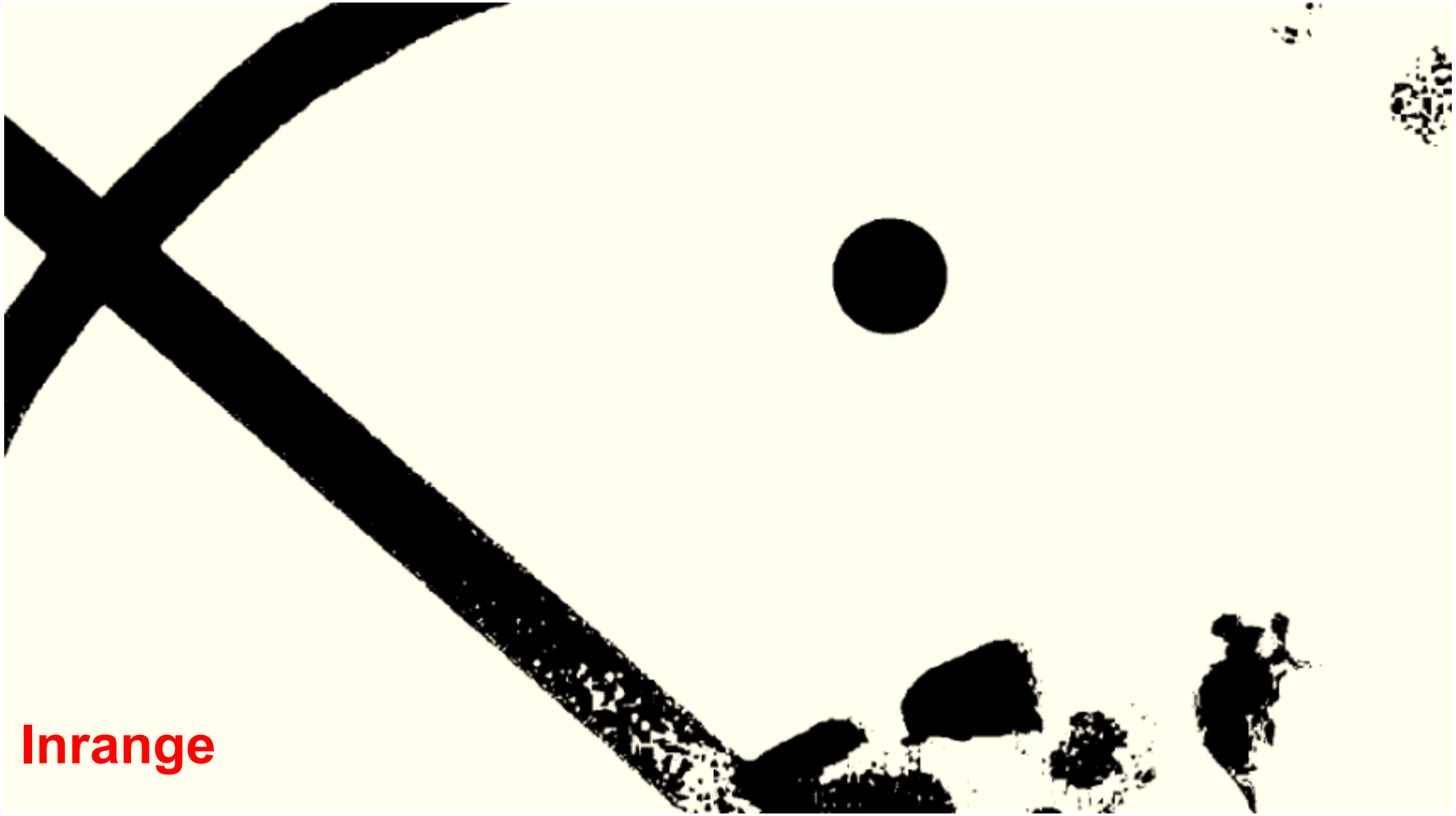
The basic idea of erosion is like soil erosion only.
***simply white region decreases** in the image.*
It is useful for removing small white noises.

Dilate

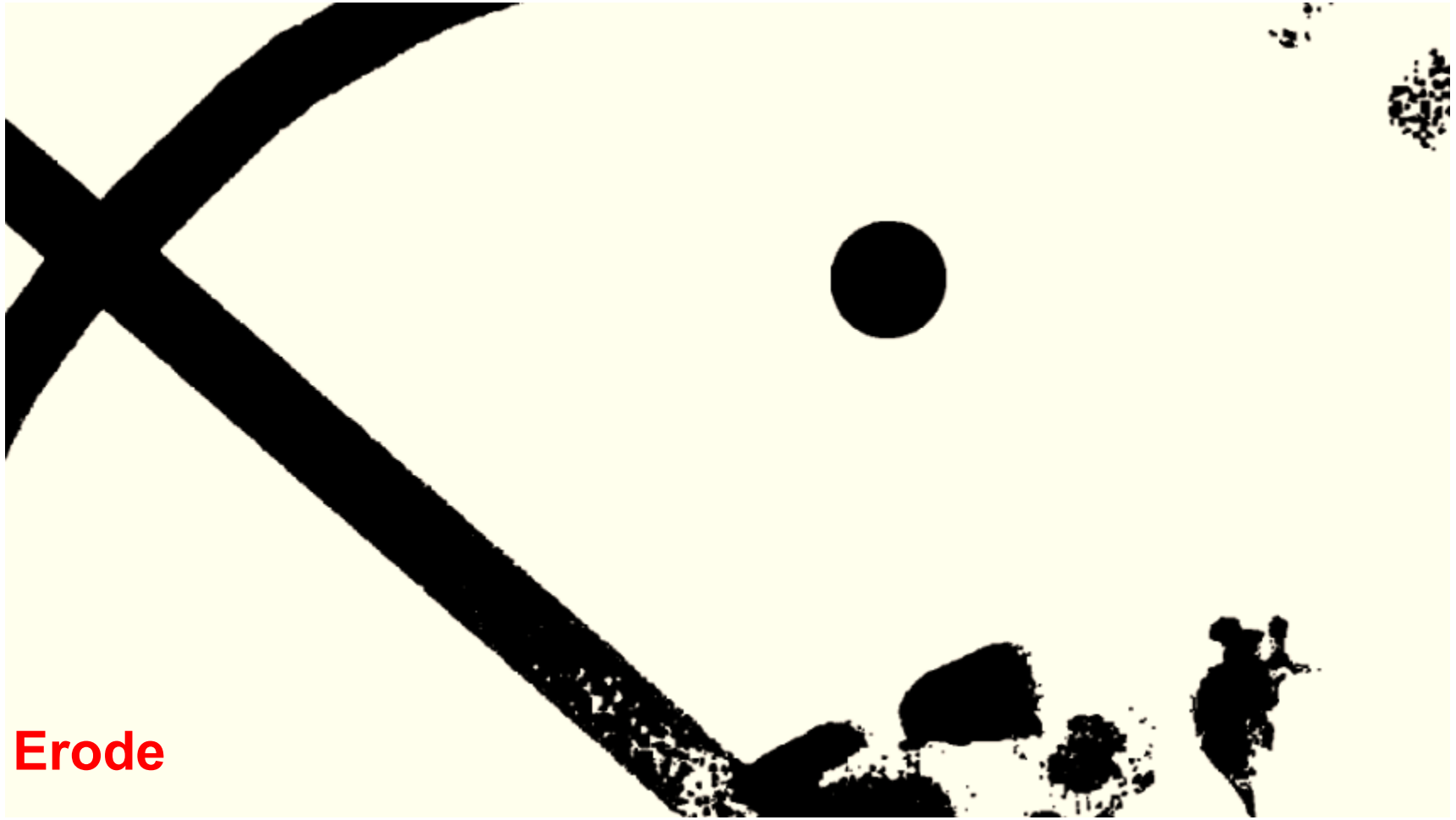
It is just opposite of erosion.
it increases the white region in the.
Normally, in cases like noise removal, erosion is followed by dilation.
Because, erosion removes white noises, but it also shrinks our object. So we dilate it.

Gaussian blur to reduce image noise and reduce detail.

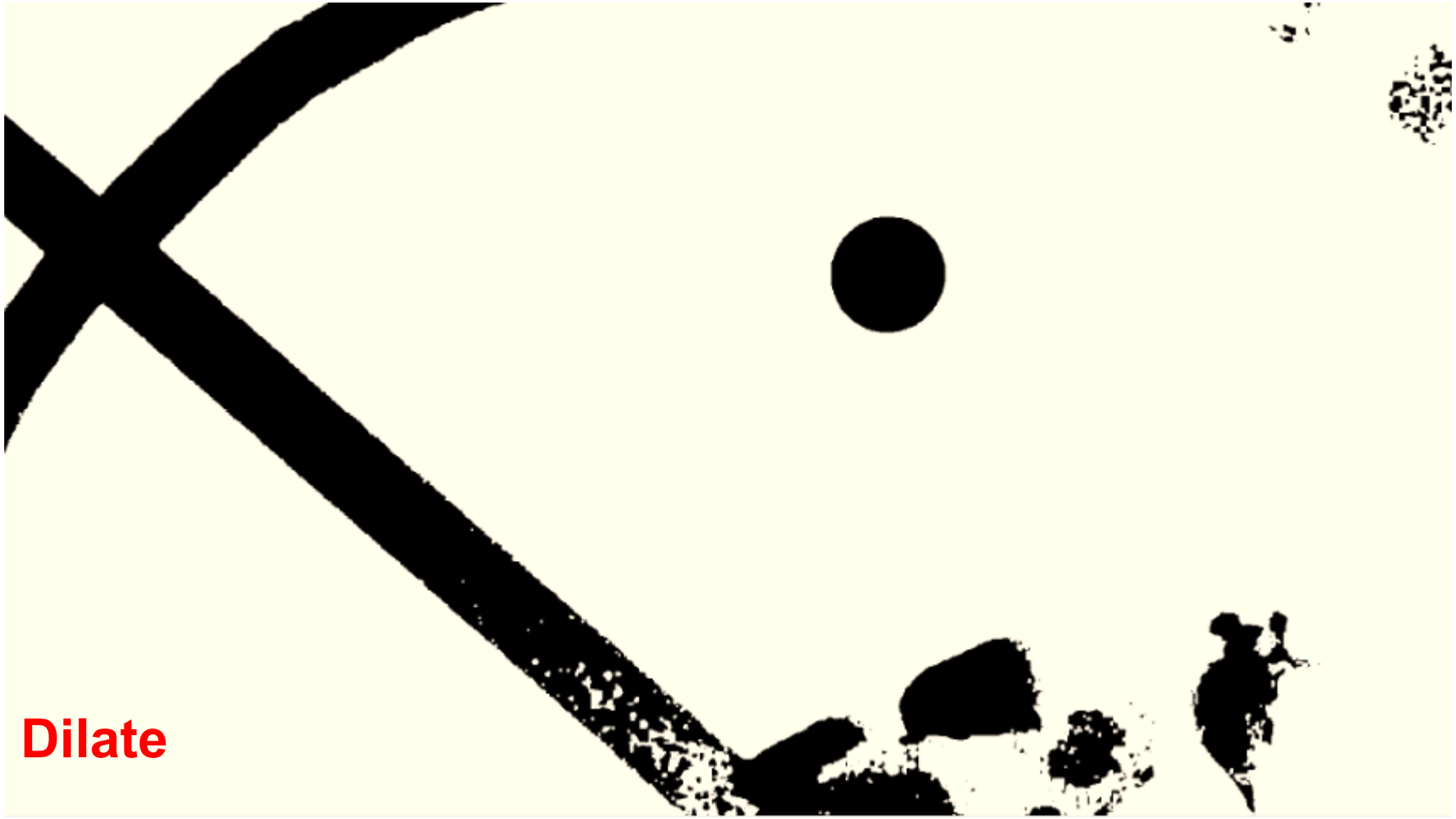
Hardware - Results



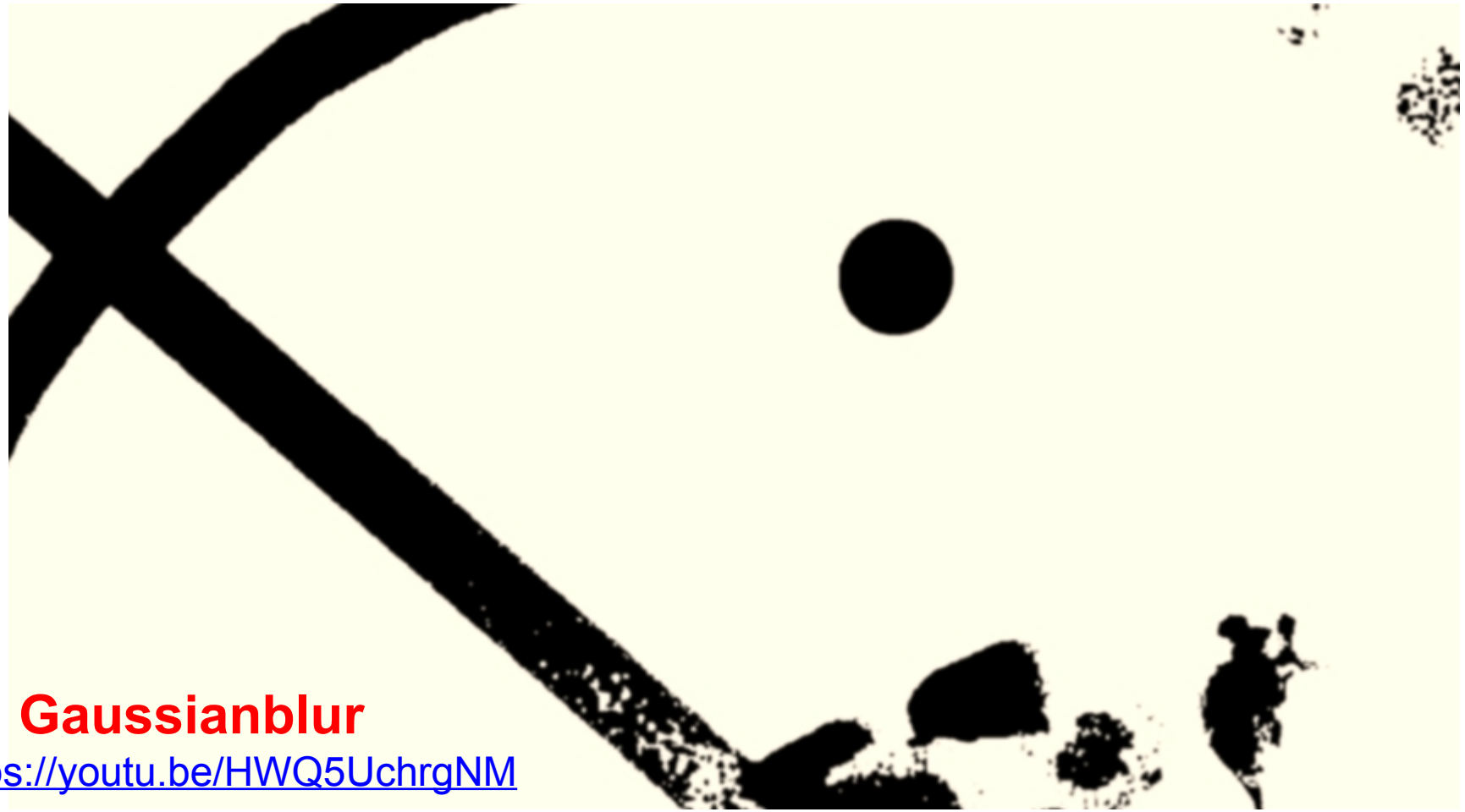
Hardware - Results



Hardware - Results



Hardware - Results



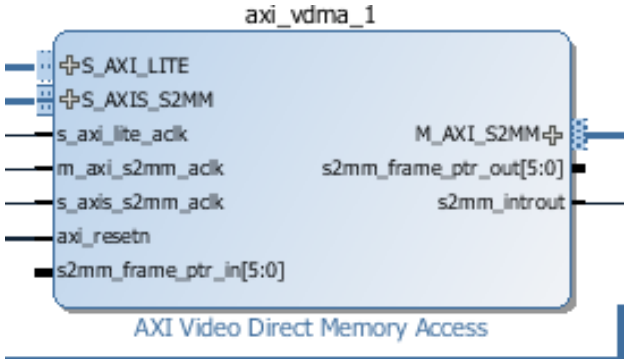
Gaussianblur

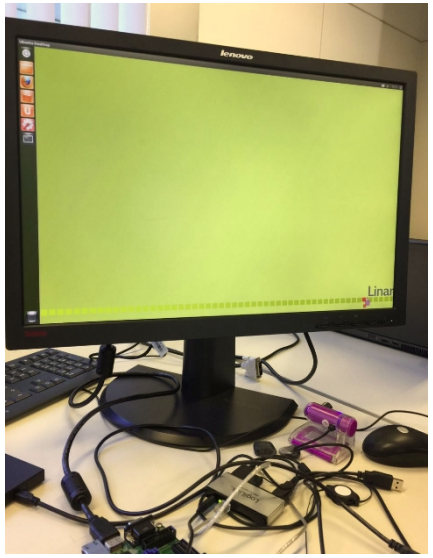
https://youtu.be/HWQ5UchrgNM

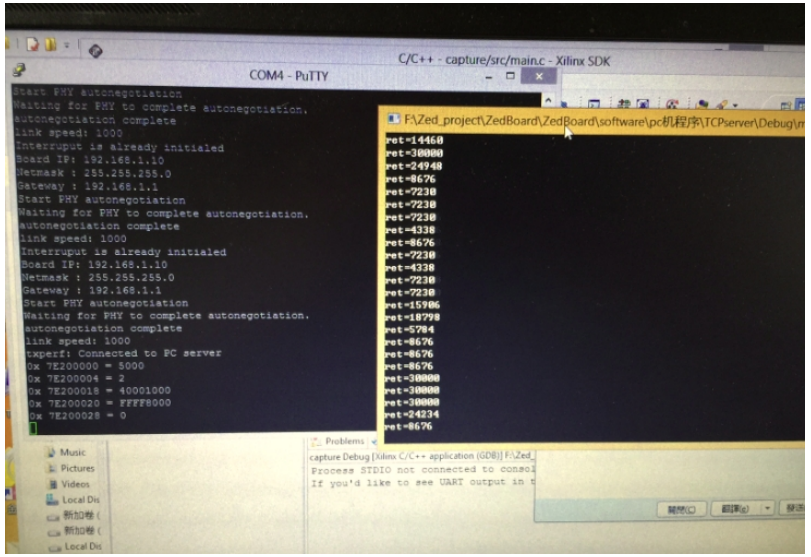
Hardware – Problems

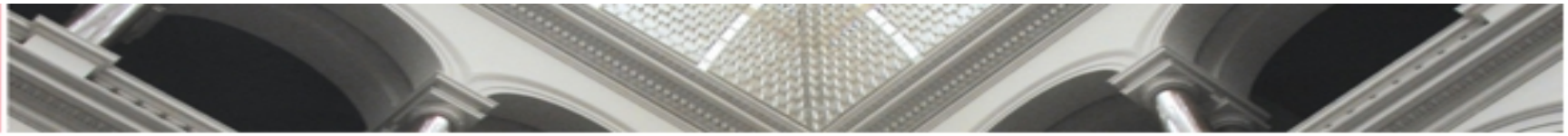
Hough circle implementation in hardware
- if want to using shape recognizing, have to use Hough circle
- next step

UDP or TCP sending
- Can send some strings
- using Linux.
  cannot use high resolution camera.









Technische Universität Berlin

Software

Technische Universität Berlin

Software – Robot Tracking

Technische Universität Berlin

Software – Robot Tracking

Initial intuition:

- Use already existing patterns on Nao

Software – Robot Tracking

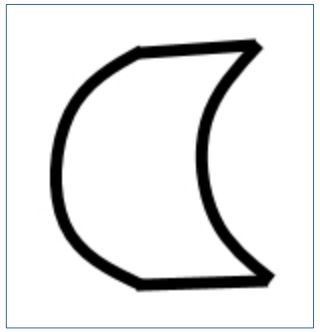
Initial intuition:

- Use already existing patterns on Nao

Software – Robot Tracking

Initial intuition:

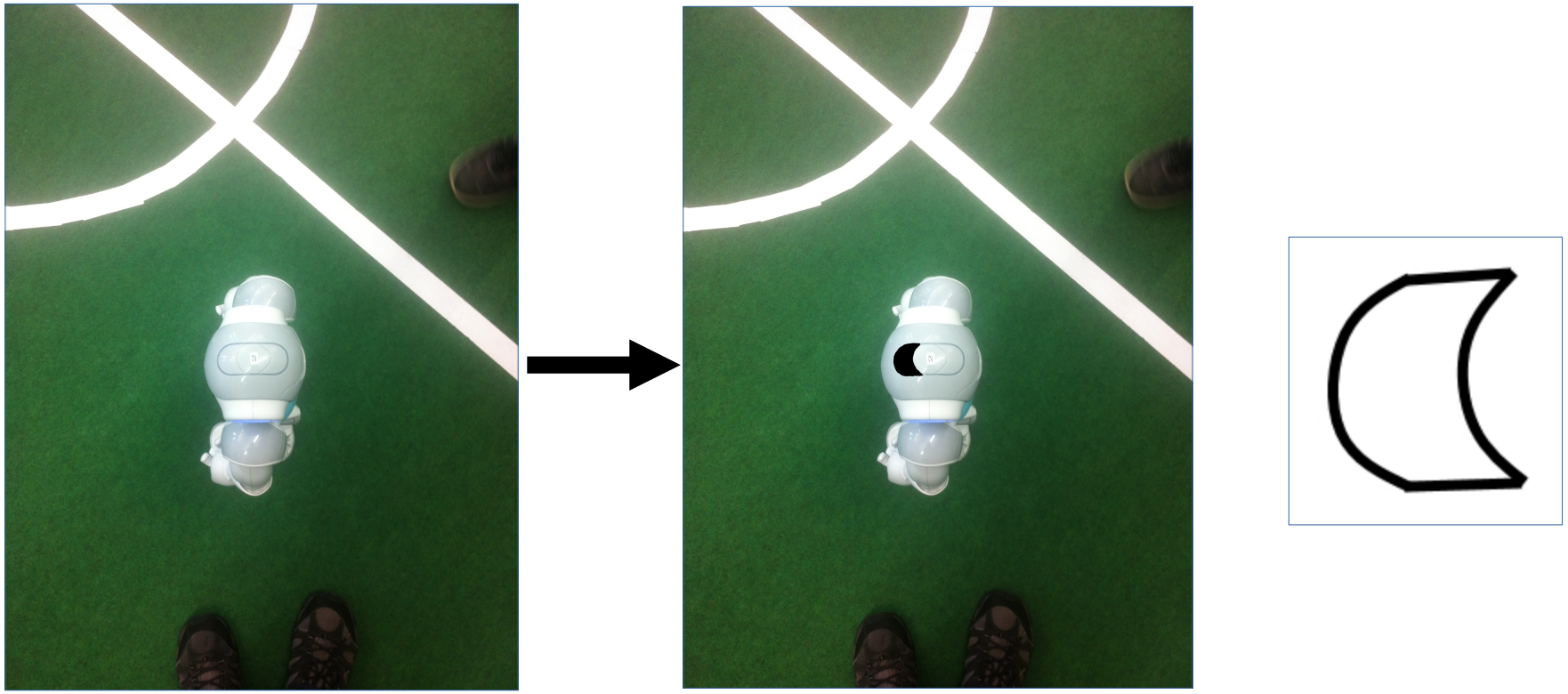
- Use already existing patterns on Nao



Software – Robot Tracking

Initial intuition:

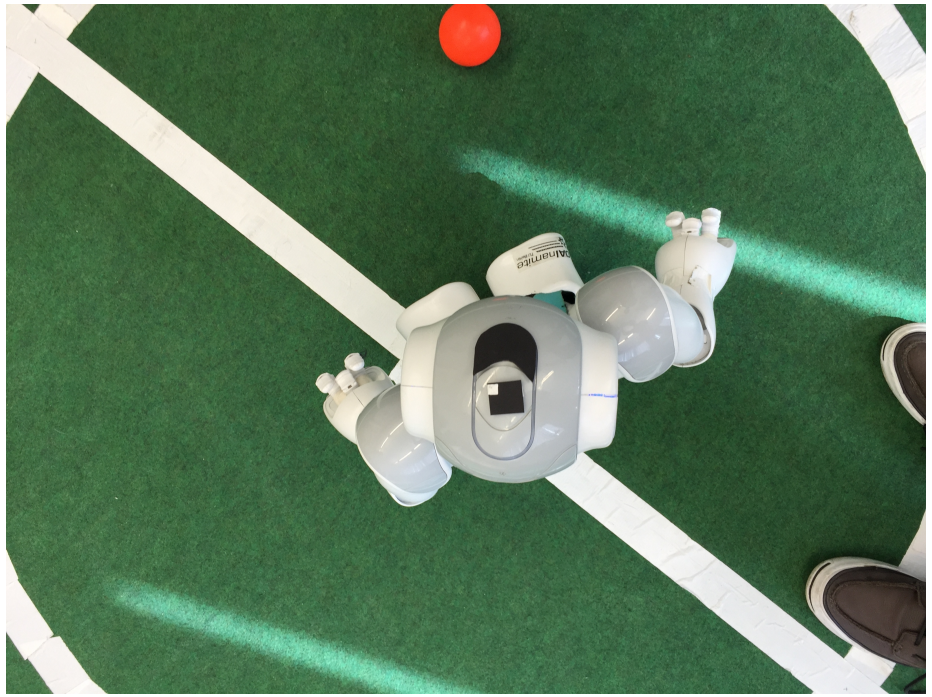
- Use already existing patterns on Nao



Software – Robot Tracking

Initial intuition:

- Use already existing patterns on Nao
- Add marker to uniquely identify each robot



Software – Robot Tracking

Problems with initial intuition:

- Custom shape matching not directly providing angles
- Additional processing for the custom ID marker

Software – Robot Tracking

Problems with initial intuition:

- Custom shape matching not directly providing angles
- Additional processing for the custom ID marker

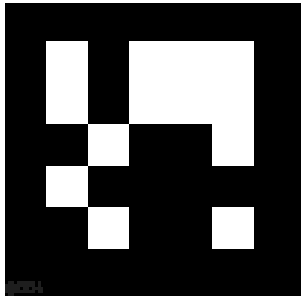
Solution:

Software – Robot Tracking

Problems with initial intuition:

- Custom shape matching not directly providing angles
- Additional processing for the custom ID marker

Solution: **ArUco markers**

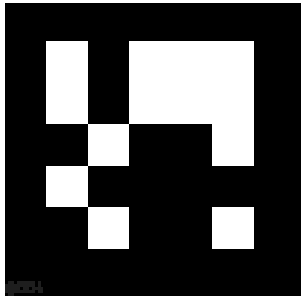


Software – Robot Tracking

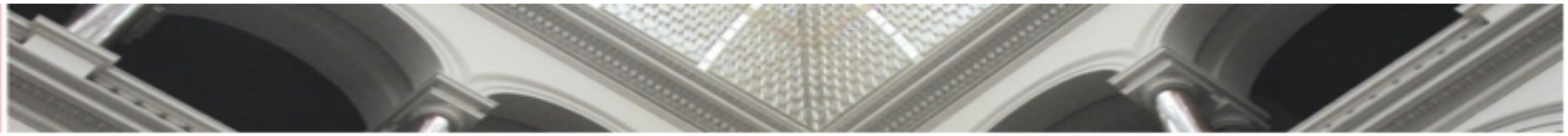
Problems with initial intuition:

- Custom shape matching not directly providing angles
- Additional processing for the custom ID marker

Solution: **ArUco markers**



- Provide 3D coordinates and marker rotation
- Provide up to 1023 unique ID's
- Easy implementation
- Rapid markers detection



Technische Universität Berlin

Software – Ball Tracking

Technische Universität Berlin

Software – Ball Tracking

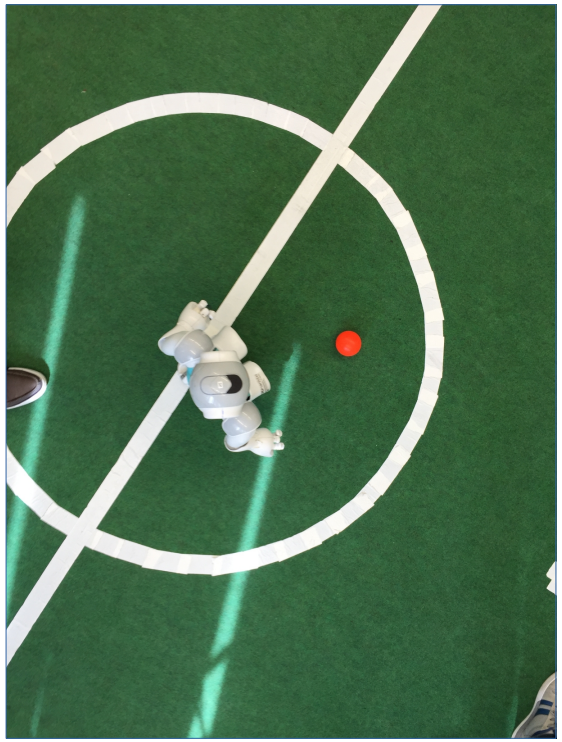
Initial intuition:

- Filter non-red elements from the frame
- Use HoughCircles to find circles with specific size

Software – Ball Tracking

Initial intuition:

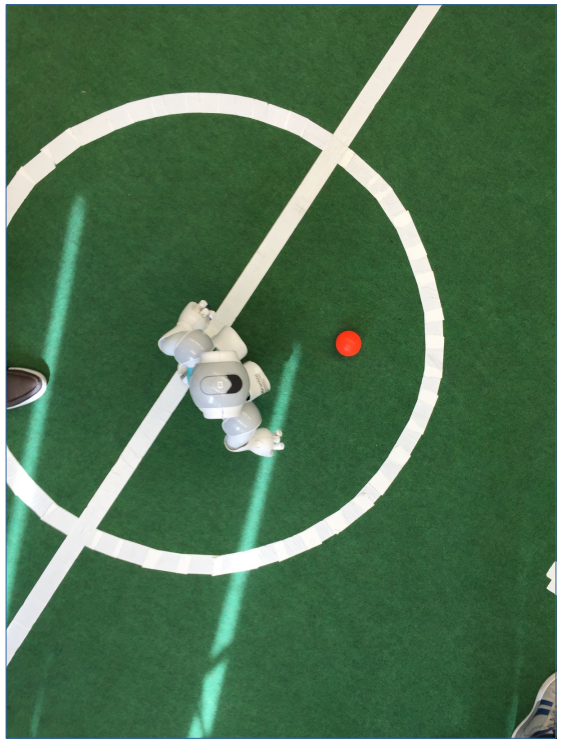
- Filter non-red elements from the frame
- Use HoughCircles to find circles with specific size

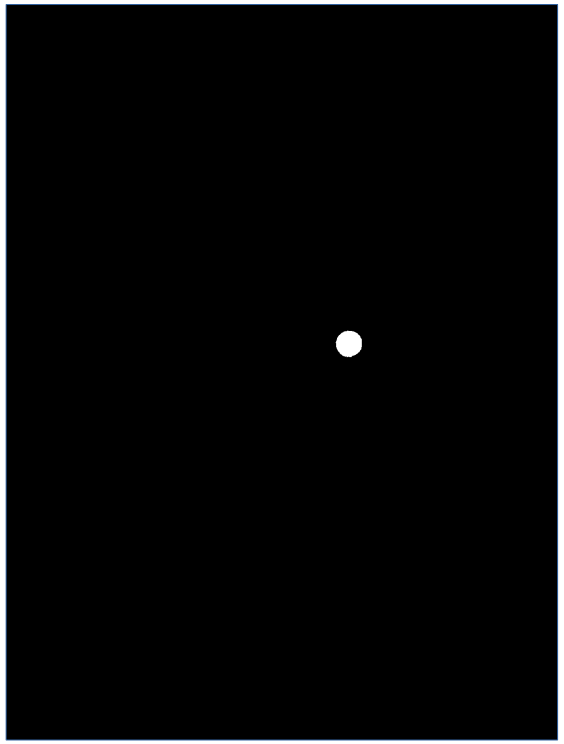


Software – Ball Tracking

Initial intuition:

- Filter non-red elements from the frame
- Use HoughCircles to find circles with specific size

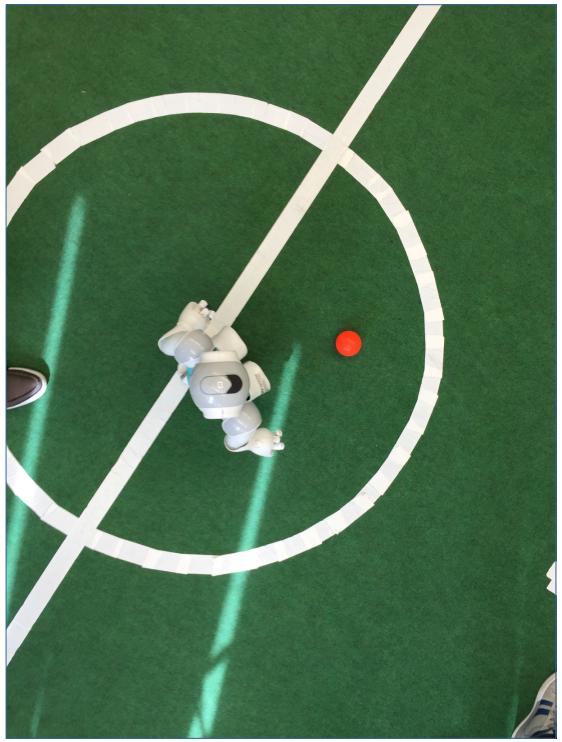


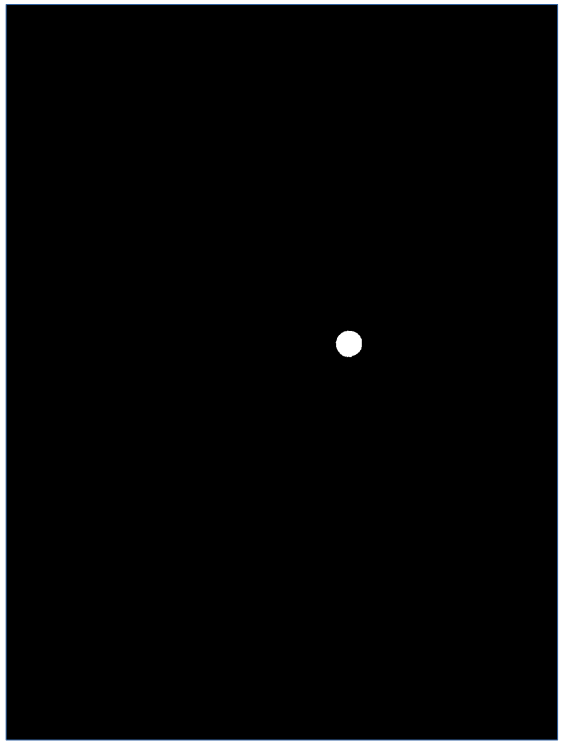


Software – Ball Tracking

Initial intuition:

- Filter non-red elements from the frame
- Use HoughCircles to find circles with specific size





Technische Universität Berlin

Software – Robot Control

Technische Universität Berlin

Software – Robot Control

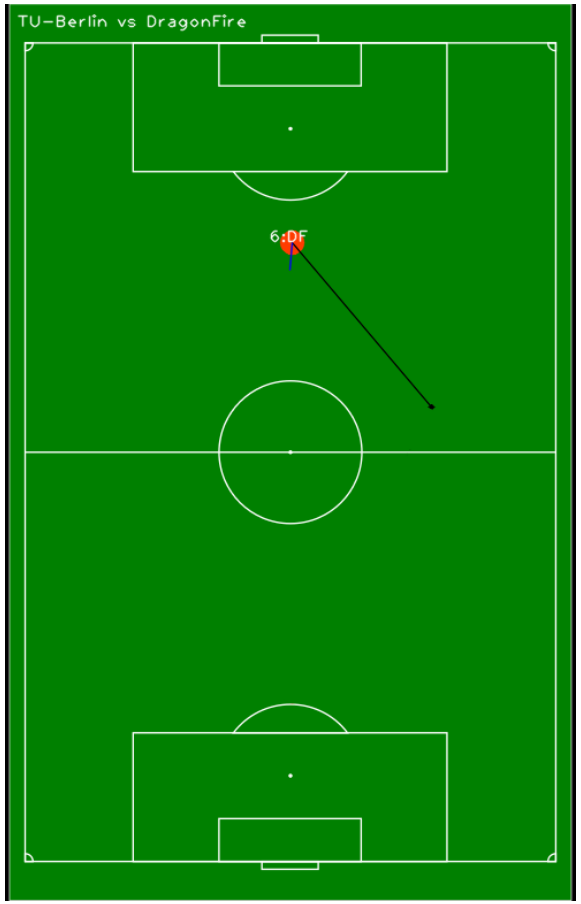
Control robots from the application through the network (WiFi, using UDP):

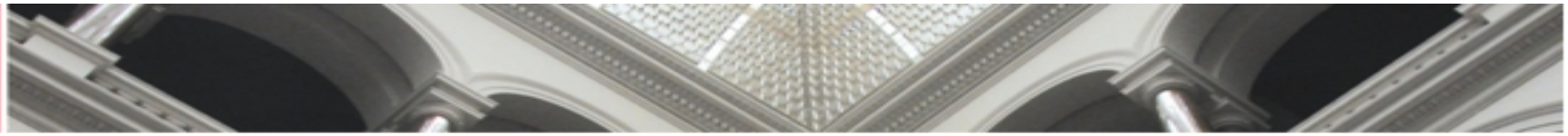
Software – Robot Control

Control robots from the application through the network (WiFi, using UDP):

Software – Robot Control

Control robots from the application through the network (WiFi, using UDP):



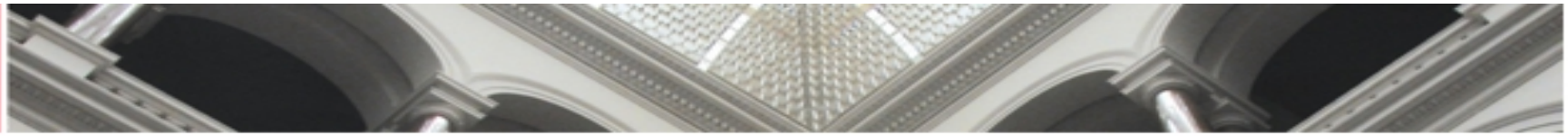


Technische Universität Berlin

Software – Robot Control

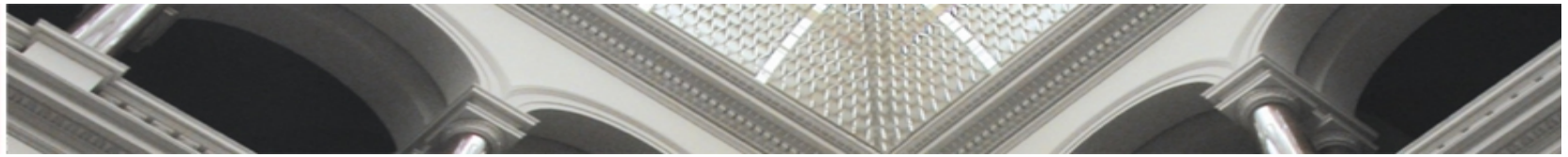
Robot information (Team Name, Team Colour, Players and IPs) stored in configuration file:

```
1  team_1=TU-Berlin
2  short_1=TUB
3  colour_1=255,255,0
4  players_1=792,4,127.0.0.1;849,2,127.0.0.2;1009,5,127.0.0.3
5  #
6  team_2=DragonFire
7  short_2=DF
8  colour_2=0,60,255
9  players_2=466,6,127.0.0.4;650,1,127.0.0.5;354,10,127.0.0.6;785,11,127.0.0.7
10 ##
11
```



Technische Universität Berlin

DEMO



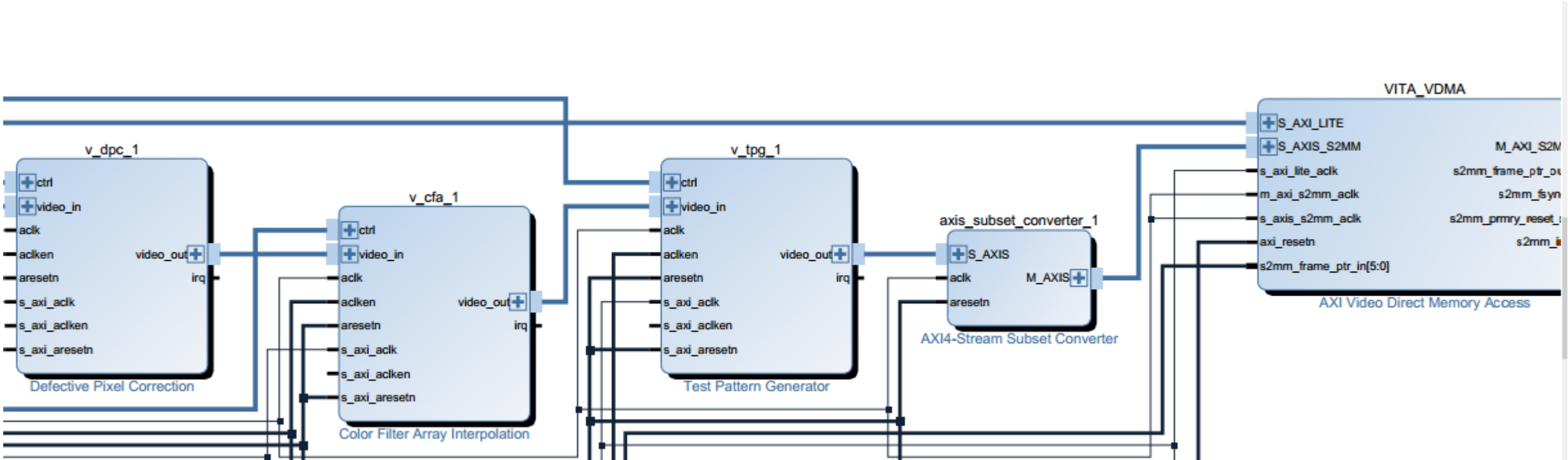
Technische Universität Berlin

The End

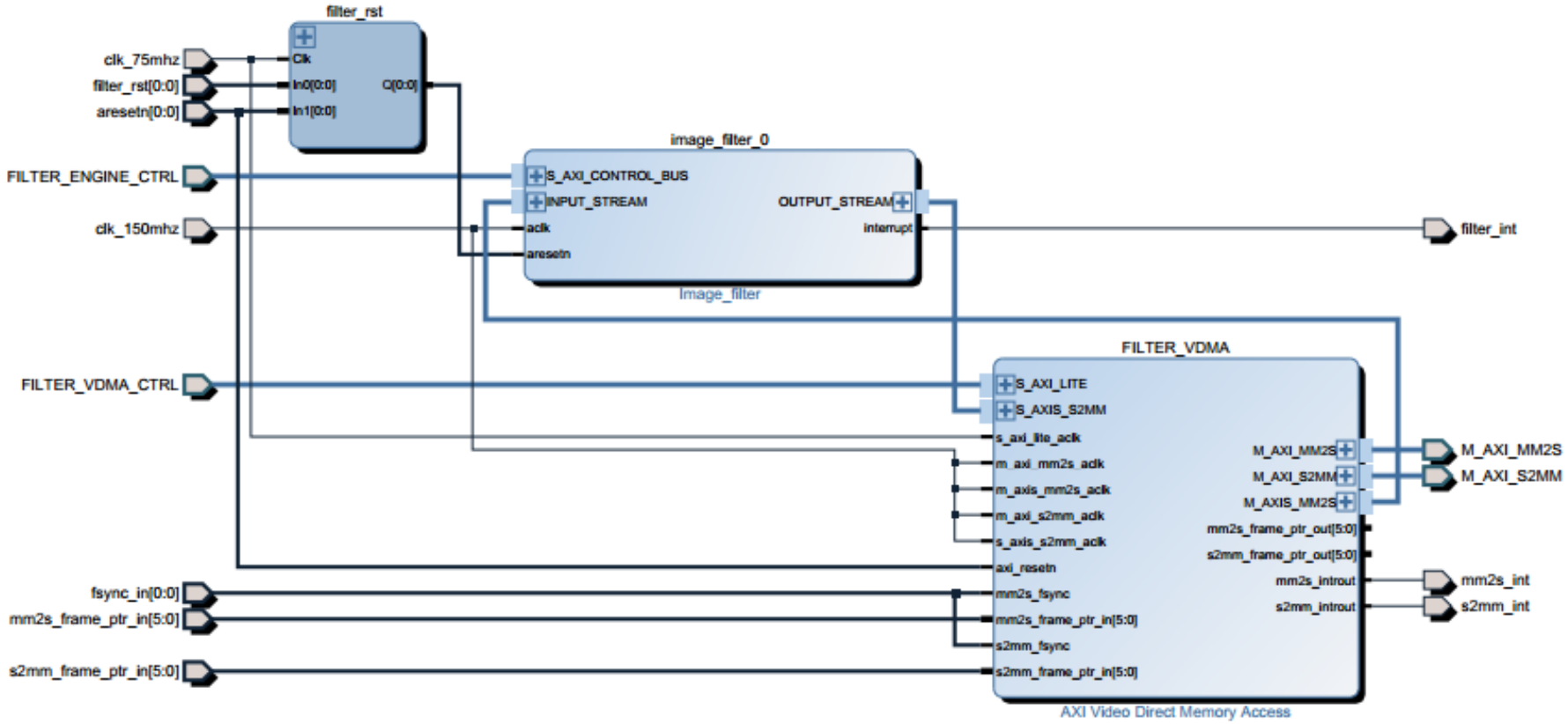
Alberto Vaccari | Yu Liu | Dapeng Lan

Backup slides

System-Level Analysis with AXI Perf. Monitoring
Access the memory speed:
HP0: Read 3.98 Gb/s, Write 3.98 Gb/s, Total 7.97 Gb/s;
HP2: Read 3.98 Gb/s, Write 3.98 Gb/s, Total 7.96 Gb/s



Backup slides



```
9     //Create AXI streaming interfaces for the core
10 #pragma HLS INTERFACE axis port=INPUT_STREAM
11 #pragma HLS INTERFACE axis port=OUTPUT_STREAM
12
13 #pragma HLS RESOURCE core=AXI_SLAVE variable=rows metadata="-bus_bundle CONTROL_BUS"
14 #pragma HLS RESOURCE core=AXI_SLAVE variable=cols metadata="-bus_bundle CONTROL_BUS"
15 #pragma HLS RESOURCE core=AXI_SLAVE variable=return metadata="-bus_bundle CONTROL_BUS"
16
17 #pragma HLS INTERFACE ap_stable port=rows
18 #pragma HLS INTERFACE ap_stable port=cols
19 //hls::Mat<MAX_HEIGHT,MAX_WIDTH,HLS_8UC3>       _dst(rows,cols);
20     RGB_IMAGE img_0(rows, cols);
21     RGB_IMAGE img_1(rows, cols);
22     RGB_IMAGE img_2(rows, cols);
23 #pragma HLS dataflow
24     hls::AXIvideo2Mat(INPUT_STREAM, img_0);
25     assert(rows <= MAX_HEIGHT);
26     assert(cols <= MAX_WIDTH);
27     for(int row = 0; row < rows; row++) {
28 #pragma HLS loop_flatten off
29         for(int col = 0; col < cols; col++) {
30 #pragma HLS pipeline II=1
31             RGB_PIXEL p;
32             img_0 >> p;
33              dRG = p.val[2]-p.val[1];
34             dGB = p.val[1]-p.val[0];
35             dBR = p.val[0]-p.val[2];
36             MIN2_t = MIN2(p.val[2],p.val[1]);
37             MAX2_t = MAX2(p.val[2],p.val[1]);
38
39             MIN2i = MIN2(p.val[0],MIN2_t);
40             MAX2i = MAX2(p.val[0],MAX2_t);
41             diff = MAX2i - MIN2i;
```

Backup slides

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
Expression	-	-	0	8
FIFO	0	-	95	540
Instance	-	36	7327	9817
Memory	-	-	-	-
Multiplexer	-	-	-	-
Register	-	-	25	-
ShiftMemory	-	-	-	-
Total	0	36	7447	10365
Available	280	220	106400	53200
Utilization (%)	0	16	6	19

Only by color