reVISION™

Responsive and Reconfigurable Vision Systems

MACHINE LEARNING | COMPUTER VISION | SENSOR FUSION | CONNECTIVITY

XILINX

ALL PROGRAMMABLE™

运行在 Zynq 上的 OpenCV：为 4k60 密集光流和立体视频提速

罗 霖（Andy Luo） — 赛灵思亚太区工业医疗市场高级经理

2017年8月

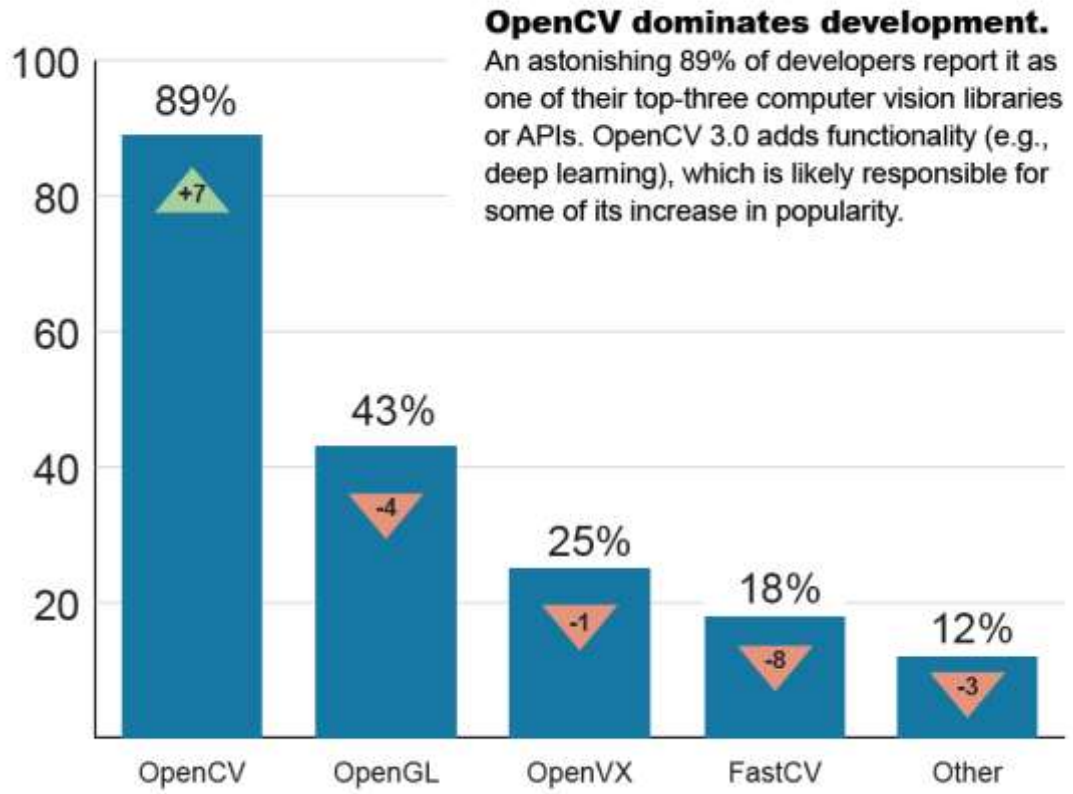
会议议程

- 为什么将 Zynq SoC 用于传统计算机视觉
- 用于 OpenCV 硬件加速的自动化流程
- 案例研究

OpenCV 需要嵌入式加速

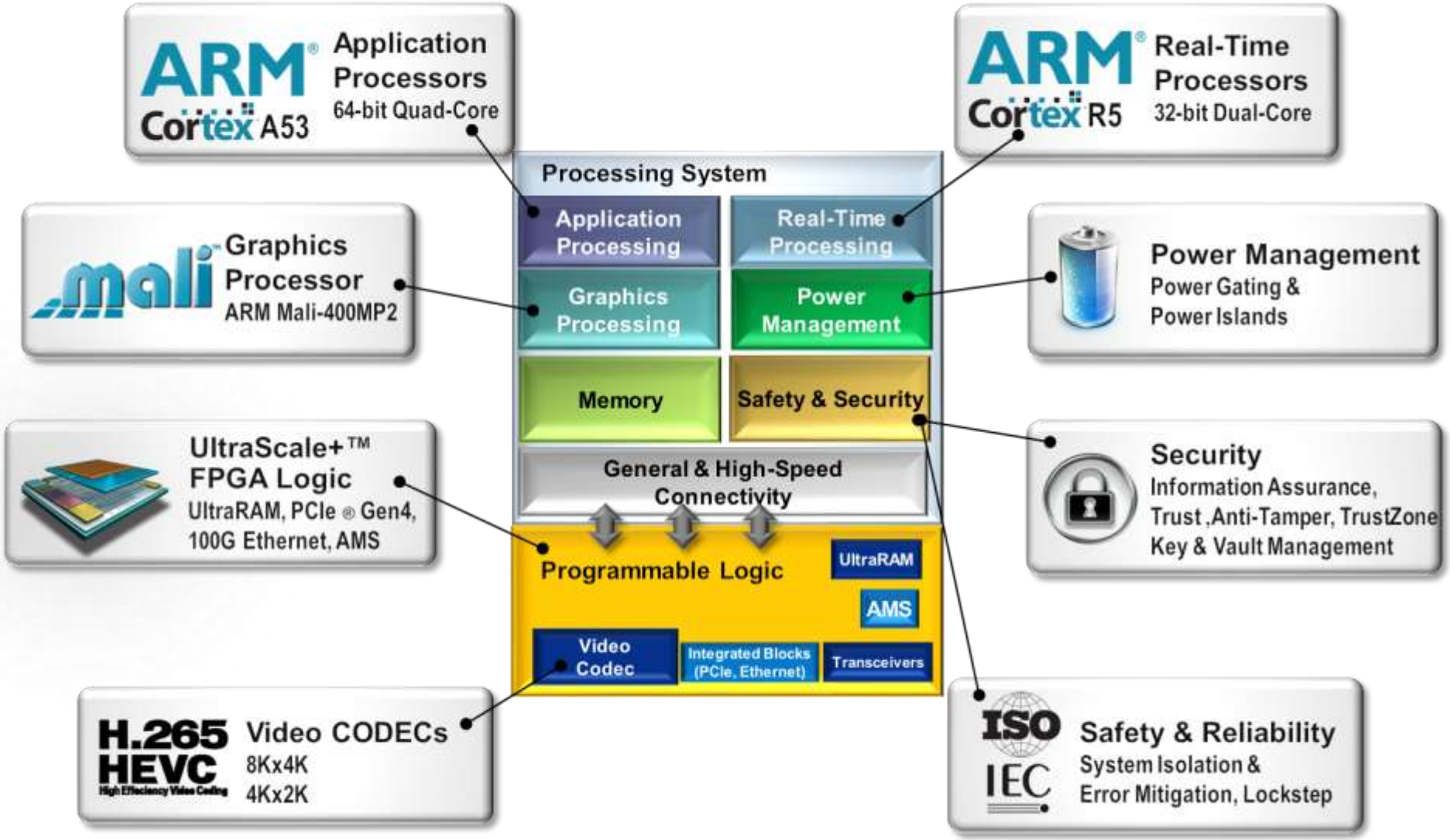
- 典型 ARM Cortex-A53

典型要求	> 30 FPS
Harris角点	2.4 FPS
立体深度图	2.1 FPS
密集光流	0.1 FPS



来源：嵌入式视觉联盟，
《嵌入式视觉开发人员调查》，2017 年 1 月

Zynq提供最有效的CV加速



Zynq 具有更高的性能和更低的时延

赛灵思基准测试

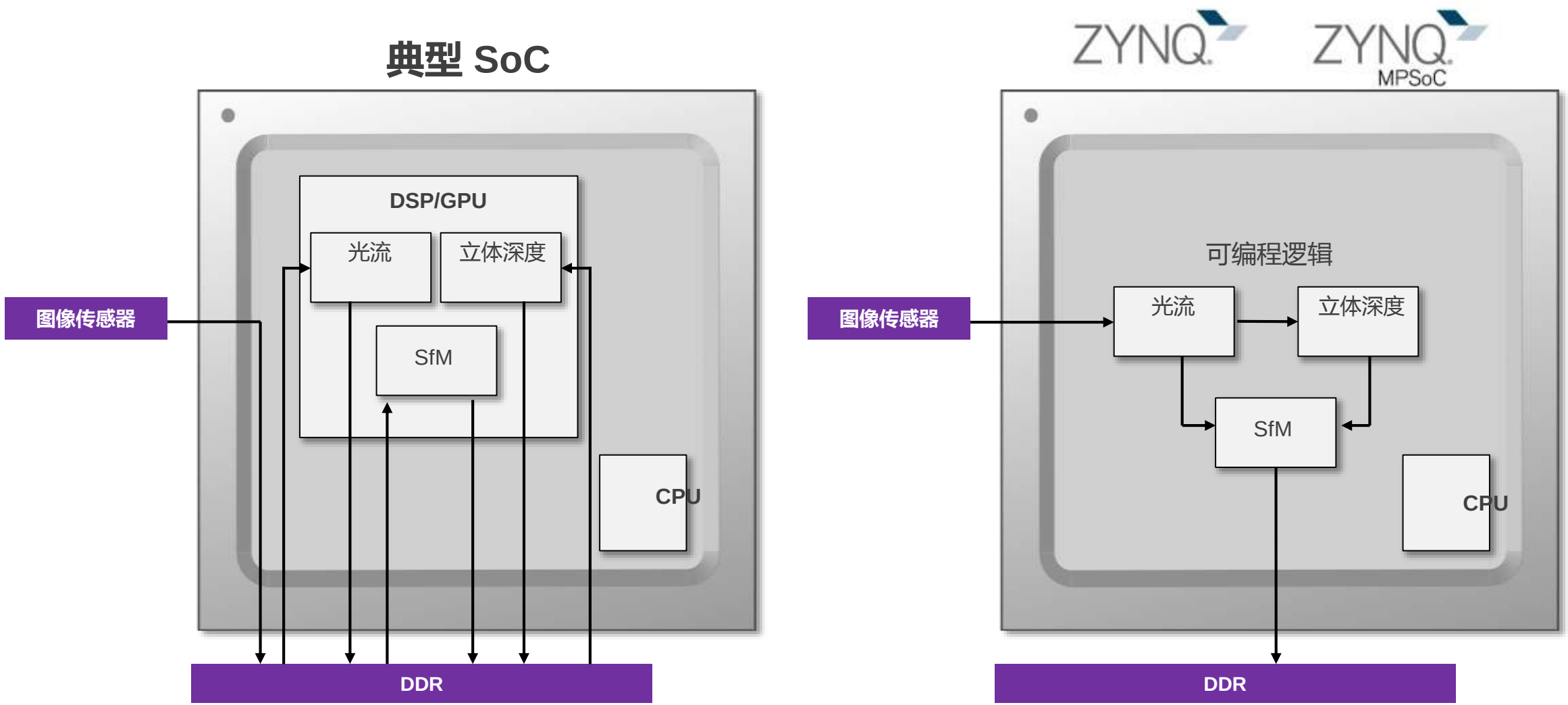
赛灵思基准测试

		赛灵思ZU9	赛灵思ZU5	eGPU*
CV:: StereoLBM @1080p	帧数/秒	**700**	**296**	43
	功耗 (瓦)	**4.8**	**3.3**	7.9
	帧数/秒/瓦	**145.8**	**89.7**	5.4

		赛灵思ZU9	赛灵思ZU5	eGPU*
CV:: LK密集光流 @720p	帧数/秒	**170**	**73**	7
	功耗(瓦)	**4.8**	**3.3**	7.9
	帧数/秒/瓦	**35.4**	**22.1**	0.9

- eGPU = nVidia Tegra X1：StereoLBM使用VisionWorks；OpticalFlow使用OpenCV4Tegra
- 所有基准测试都尽可能量多地使用 GPU 资源 (~99%) 和可编程逻辑(~70%)

为何如此棒？高效率基于窗口有效串流



reVISION Stack

OpenVX Vision Acceleration | Caffe

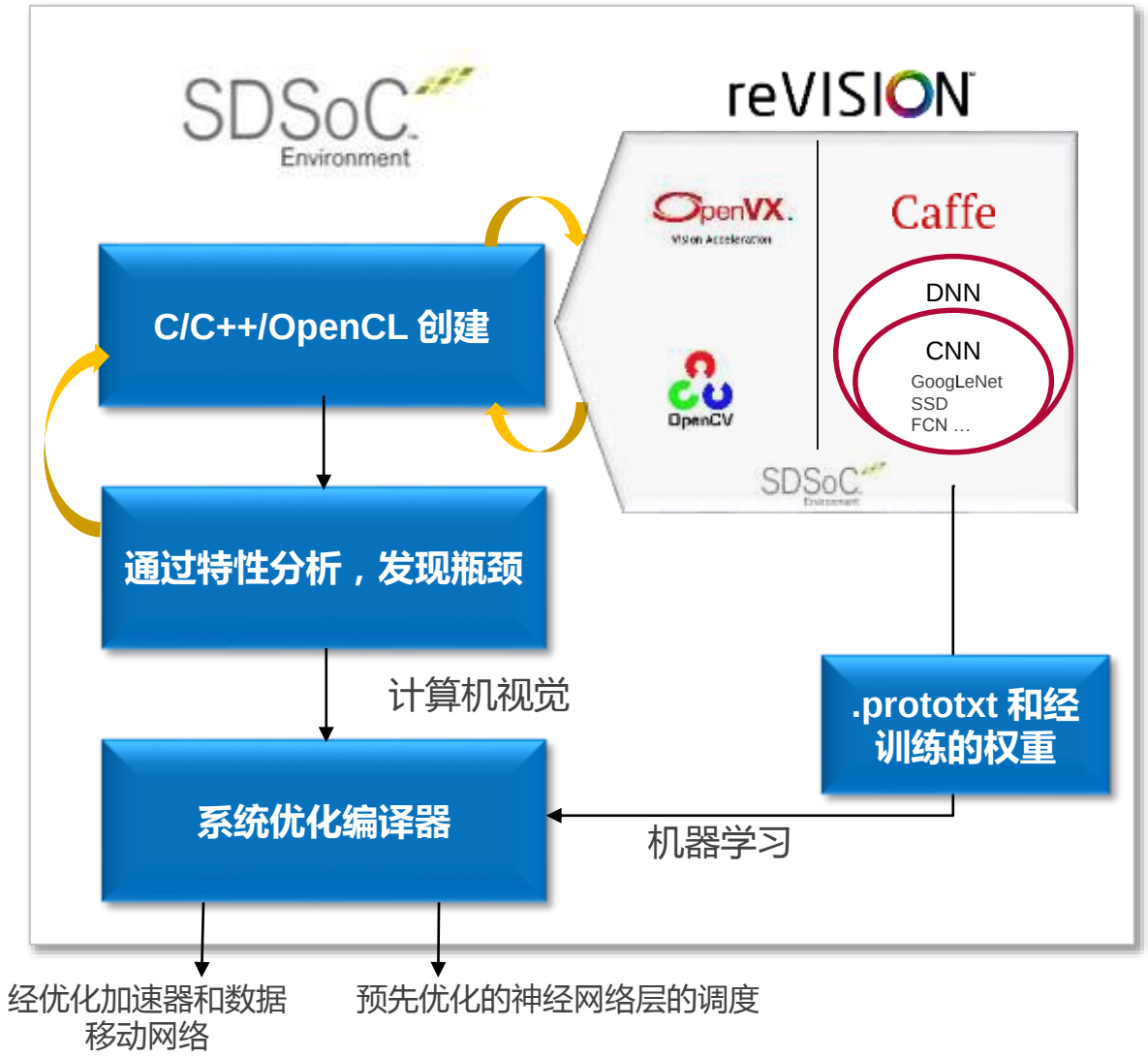
OpenCV | SDSoC Environment | DNN, CNN, GoogLeNet, SSD, FCN …

框架

库与工具

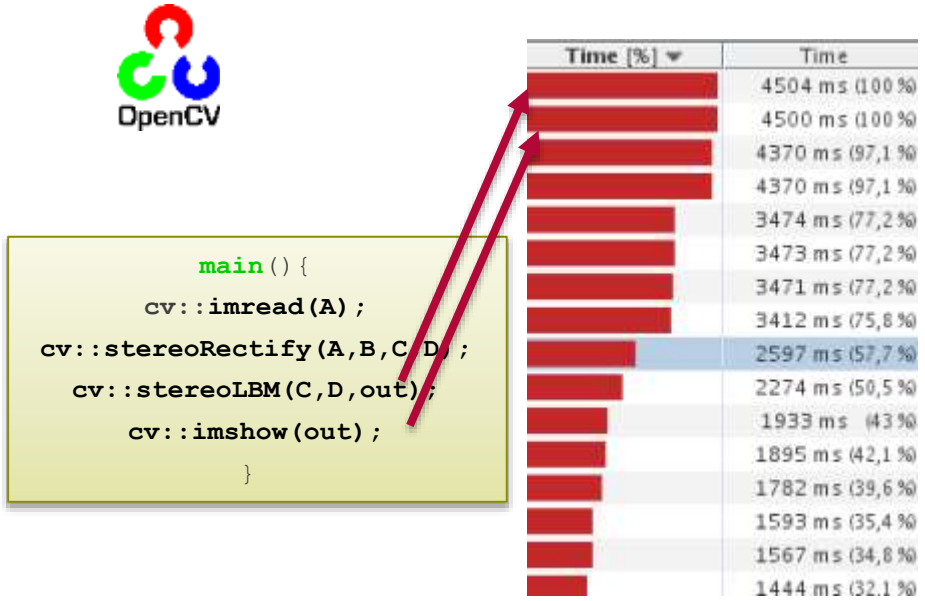
开发套件

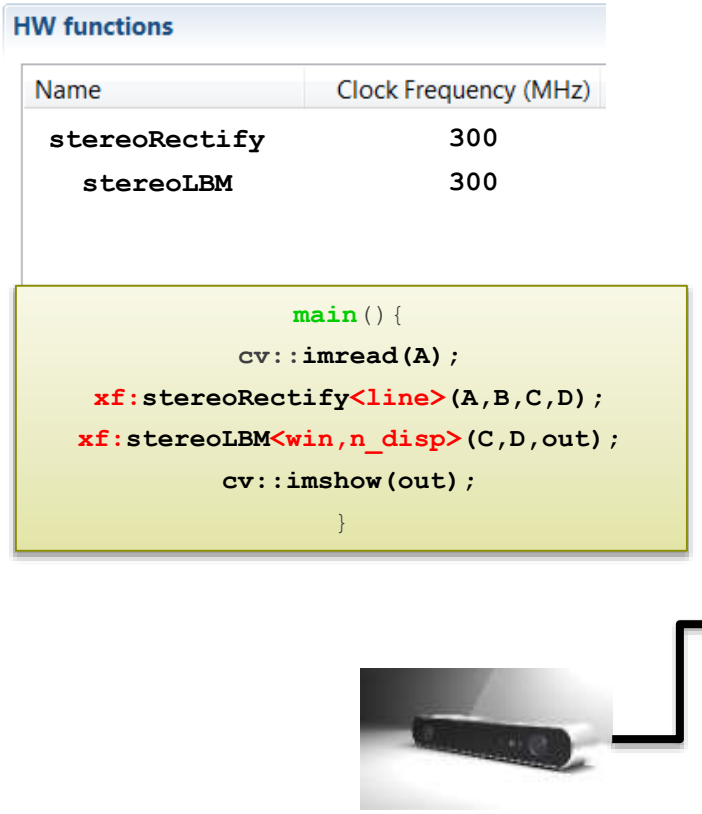
推翻“Zynq SoC 难以编程”

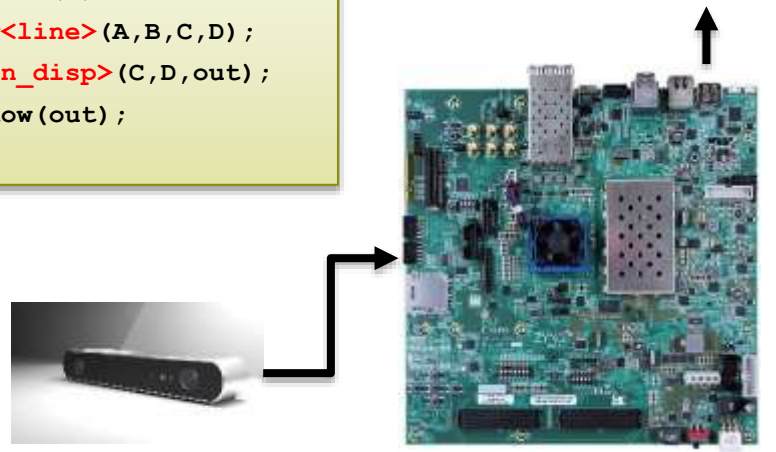


利用用自动硬件加速提供 OpenCV 支持

1. 交叉编译 OpenCV 应用到 Zynq (ARM A9/A53)
2. 特性分析和发现瓶颈功能
3. 对代码做最小改动，设置功能到硬件。使用 SDSoC编译。
4. 在 Zynq 开发板上运行







xfOpenCV： 硬件加速 OpenCV 功能

1级		2级		3级
绝对差	通道合并	矩形窗	缩放/调整尺寸	取向梯度直方图（HOG）
累计	通道抽取	高斯	立体矫正	
累计开平方	颜色转换	中值	仿射变换	SVM（二进制）
累计加权	转换位深度	索贝尔	弯曲透视	OTSU 阈值设定
算术加	查找表	定制卷积	快速角点	均值移位跟踪 (MST)
算术减	直方图			LK 密集光流
位运算： AND, OR, XOR, NOT	梯度相位	膨胀	Harris 角点	Canny 边缘检测
像素相乘	最小/最大位置	侵蚀	重映射	影像金字塔
积分图像	平均和标准偏差	双边	均衡直方图	颜色检测
梯度幅度	阈值设定			立体 LBM

定制 CV 功能/库创建流程

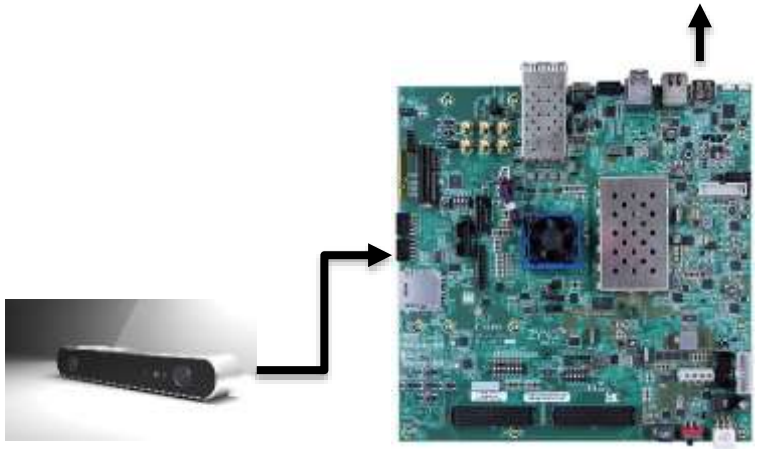
1. 交叉编译到 Zynq (ARM A9/A53)
2. 在 C、C++ 或 OpenCL 中编写定制 CV 功能。使用 HLS 为硬件优化。
3. 分配功能到硬件。使用 SDSoC 编译。
4. 在 Zynq 开发板上运行。

```
main(){
cv::imread(A);
xF:stereoRectify<line>(A,B,C,D);
xF:stereoLBM<win,n_disp>(C,D,E);
CUSTOM_CV(E,out);
cv::imshow(out);
}
```

```
CUSTOM_CV(E,out){
  #pragma HLS PIPELINE
  for(…){
   #pragma HLS UNROLL
   for(…){ …
   }
  }
}
```

HW functions

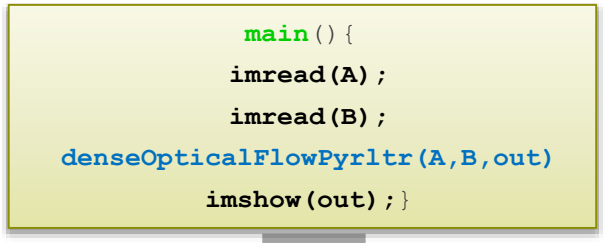
Name	Clock Frequency (MHz)
stereoRectify	300
stereoLBM	300
CUSTOM_CV	300

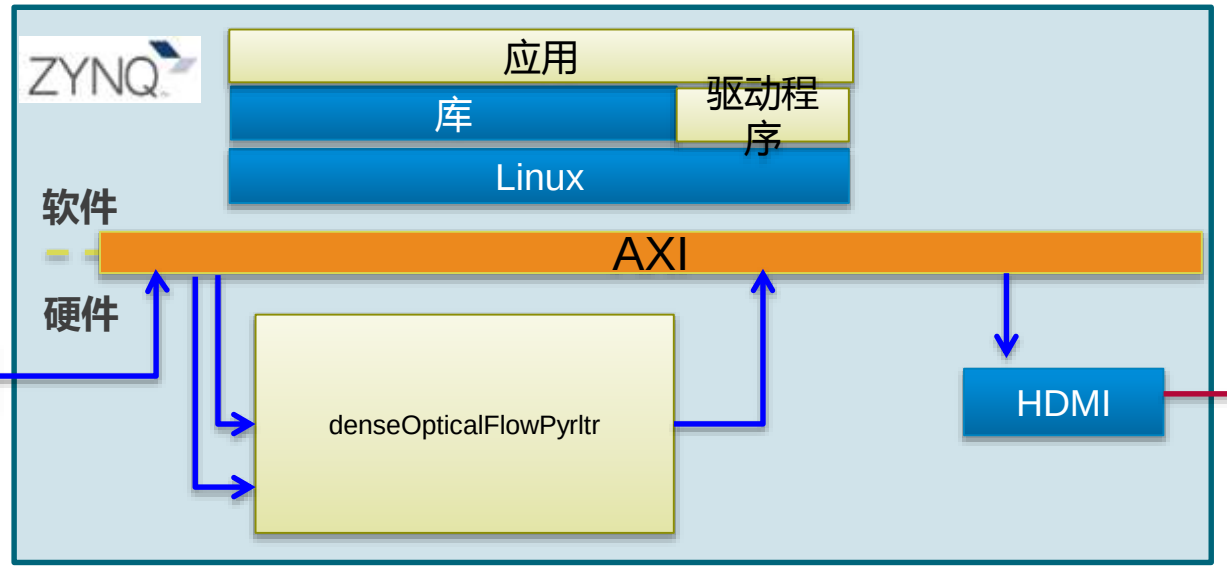


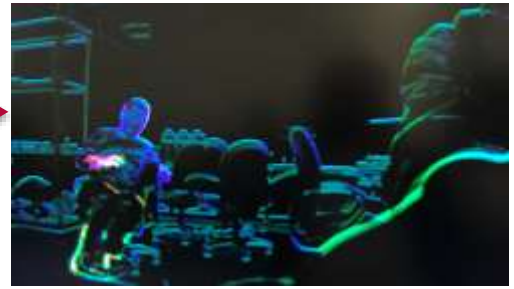
实例： 4K60 LK 密集光流

	赛灵思ZU9
帧数/秒	60
功耗(瓦)	4.8
时延(毫秒)	16.7
利用率	15%

- 使用 CUDA OpenCV 的 nVidia 数量
- 赛灵思和 nVidia 基准测试均未包含摄像头输入和 HDMI/DP
- LK 密集光流，非金字塔，非迭代，窗口大小 53x53





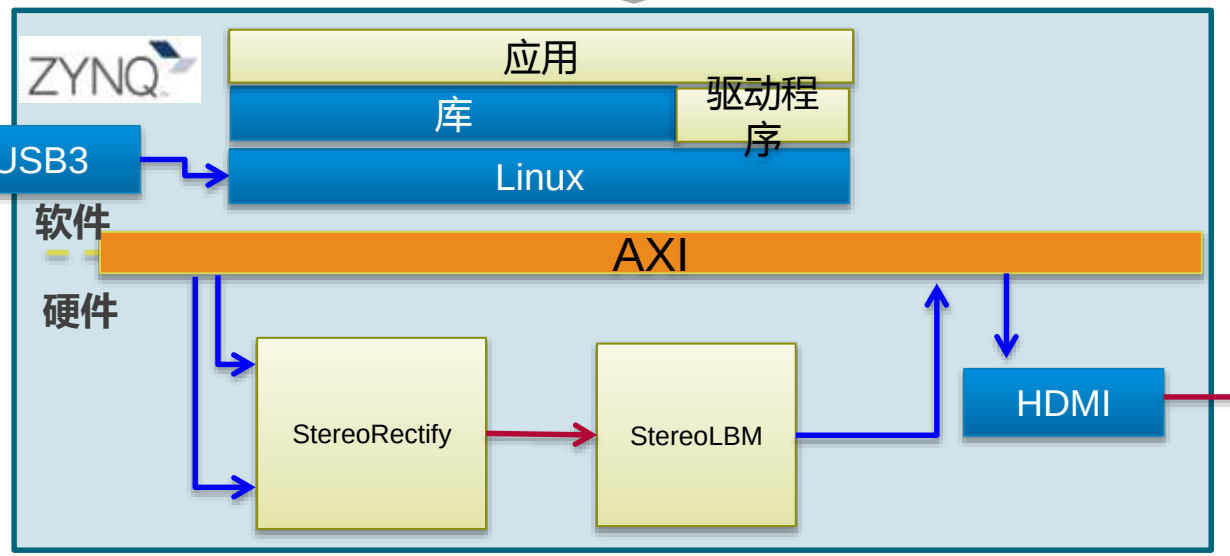


实例：立体深度映射

	赛灵思ZU9
帧数/秒	140
功耗 (瓦)	4.8
时延 (毫秒)	7.1
利用率	14%

```
main(){
imread(A);
imread(B);
stereoRectify(A,B,C,D);
stereoLBM(C,D,out);
imshow(out);}
```

SDSoC Environment

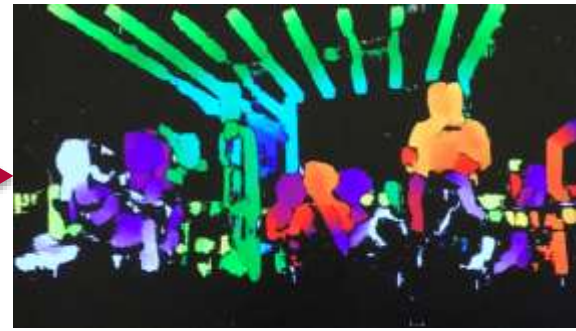


SDSoC生成

平台

DMA

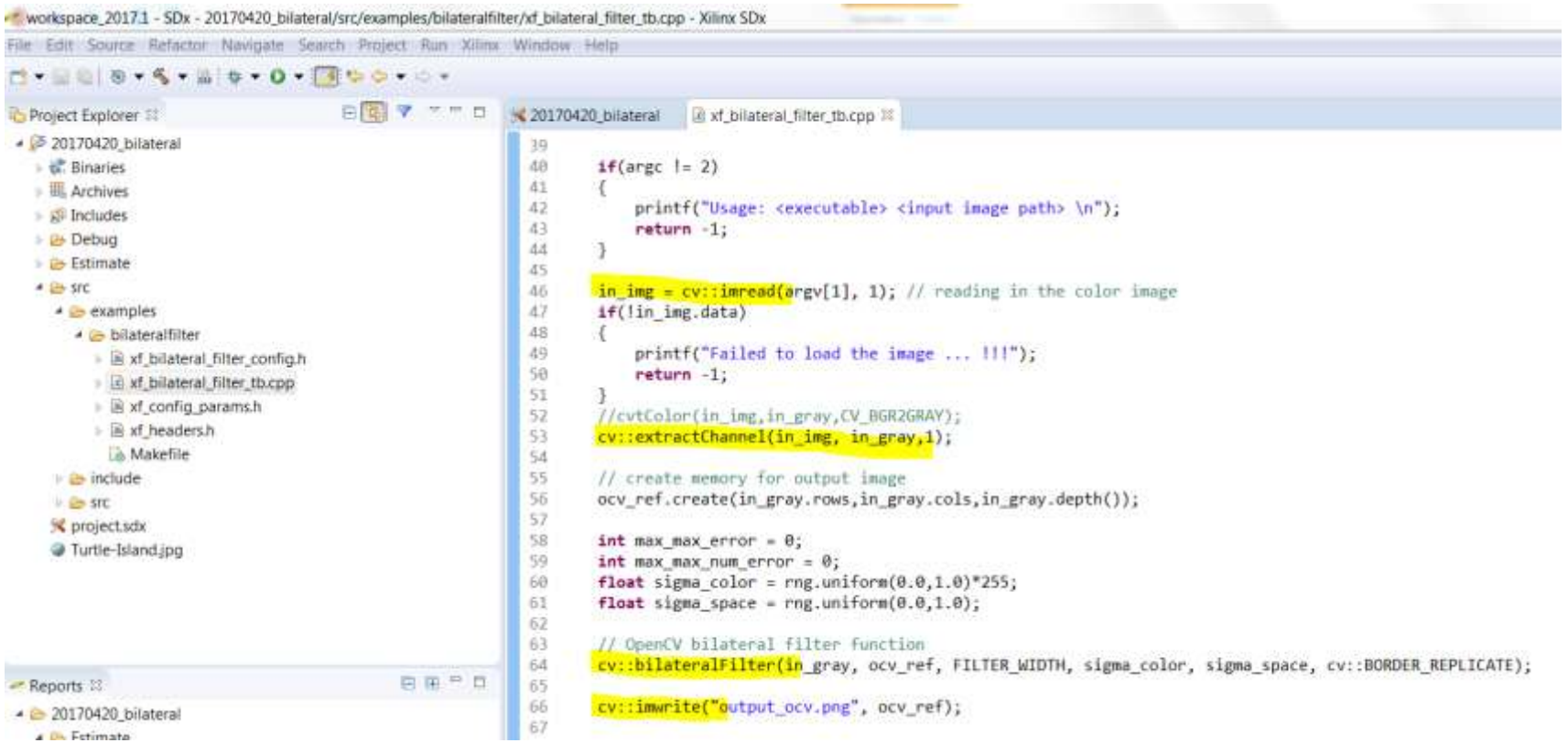
AXI-S



- 使用 CUDA OpenCV 的 nVidia 数量
  - 基于 SAD 的立体localBM
- 赛灵思和 nVidia 基准测试均未包含摄像头输入和 HDMI/DP 输出

第 1 步：导出桌面 OpenCV 应用到 Zynq

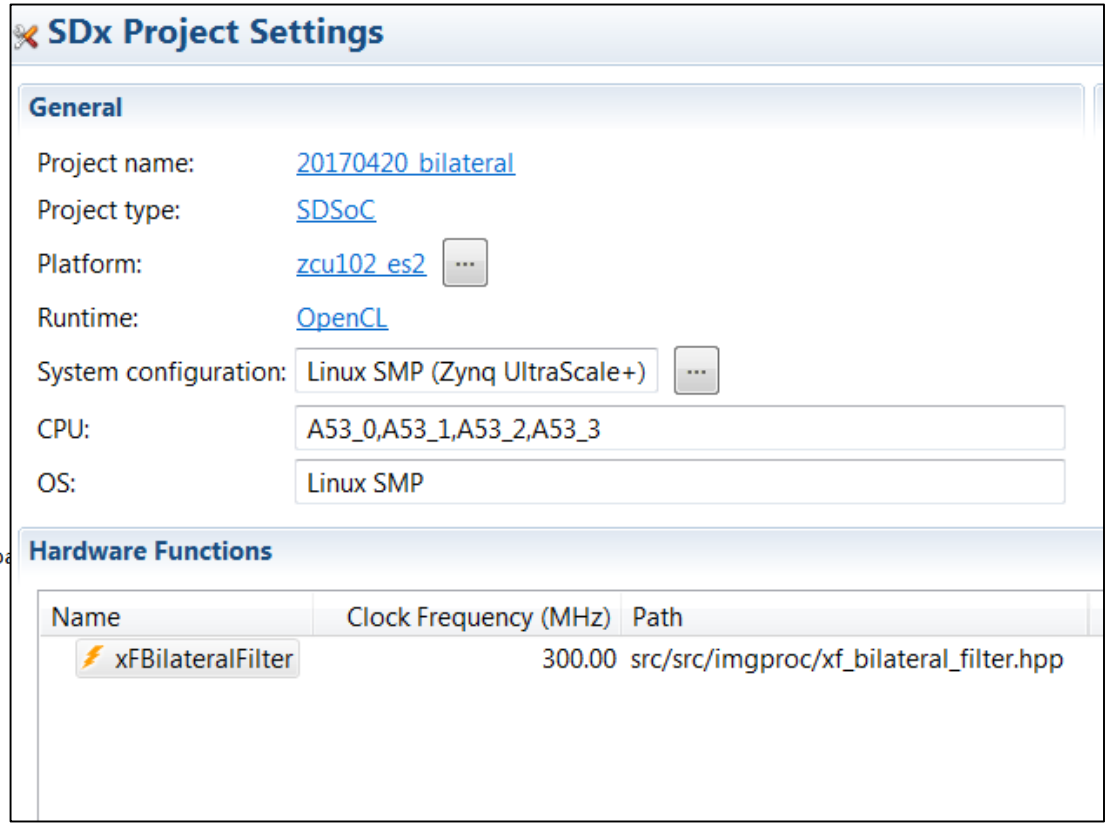
- 使用 OpenCV API 直接导出 C/C++ 项目到 SDSoC 中
- 为 ARM 提供所有必要的 OpenCV 编译/ 链接环境
- 编译准备就绪 ！



第 2 步：分配功能到硬件加速

- 将 OpenCV 库用于硬件加速需要辅模式
  - 命名域修改： “cv::” 改为 “xF::”
  - 添加模板参数，生成最佳硬件
- 直接分配关键功能给硬件

```
uint16_t width = in_gray.cols;
uint16_t height = in_gray.rows;

xF::Mat<XF_8UC1, HEIGHT, WIDTH, NPC1> _src(height,width);
xF::Mat<XF_8UC1, HEIGHT, WIDTH, NPC1> _dst(height,width);

_src.copyTo(in_gray.data);

xF::BilateralFilter<FILTER_WIDTH, XF_BORDER_REPLICATE, XF_8UC1, HEIGHT, WIDTH, NPC1>(_src,_dst, sigma_spa

out_img.data = _dst.copyFrom();
imwrite("output_hls.png", out_img);

absdiff(ocv_ref, out_img, diff); // Compute absolute difference image

//  Find minimum and maximum differences.
```



第 3 步：性能估算与构建

- 数分钟内完成快速构建，取得系统级性能和硬件利用率
- 一键点击构建完整系统

Details

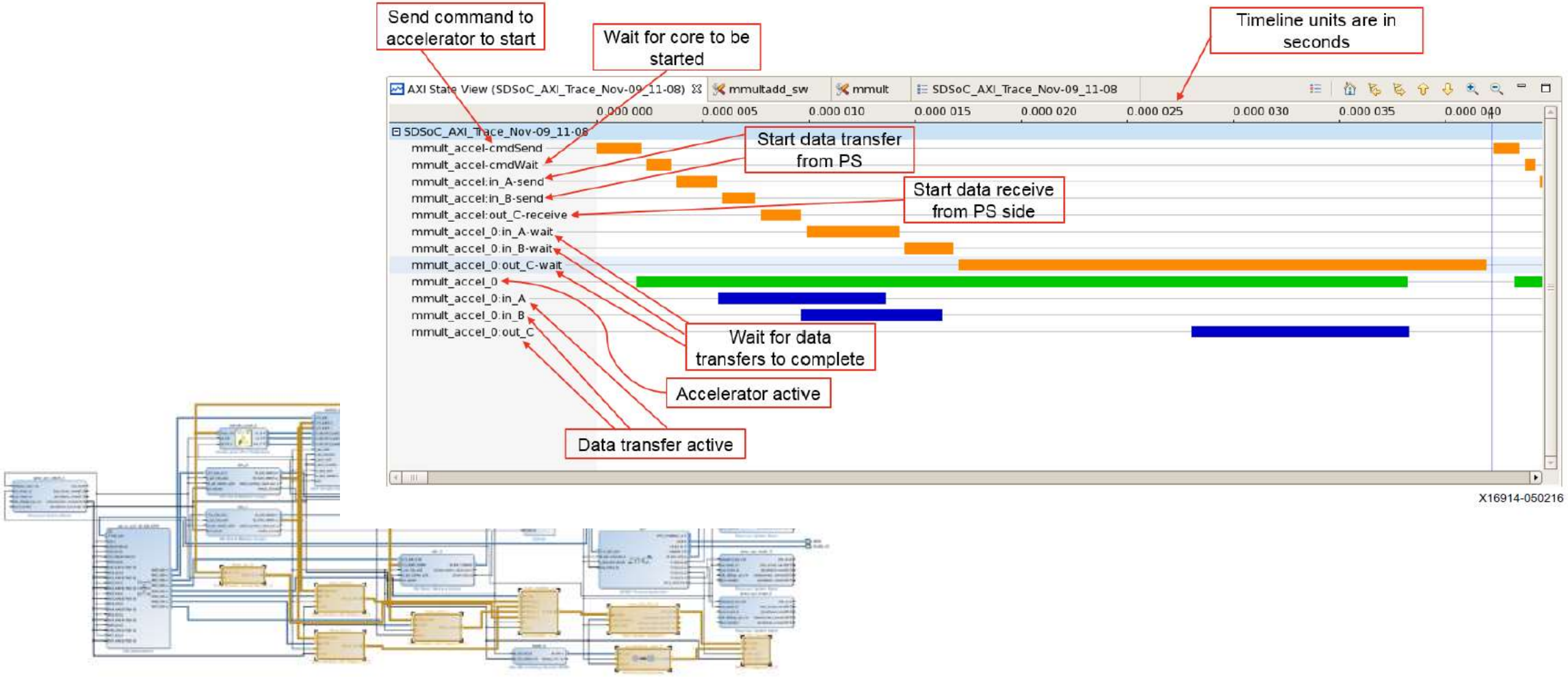
Performance estimates for 'xFBilateralFilter_3_1_0_1080_1 ...

Hardware accelerated (Estimated cy	30185245

Resource utilization estimates for Hardware functions

Resource	Used	Total	% Utilization
DSP	22	2520	0.87
BRAM	3	912	0.33
LUT	7061	274080	2.58
FF	6627	548160	1.21

第 4 步：在开发板上运行并采集踪迹



总结

- 与其他 SoC 解决方案相比，Zynq SoC 具有更高的性能和更低的时延
- SDSoC 上的 reVISION 堆栈使用预先优化的库引入熟悉的软件环境
- 现已开始供货
- 欢迎访问 reVISION 开发人员专区：
  https://china.xilinx.com/products/design-tools/embedded-vision-zone.html#computer

设计实例详见：Xilinx.com/revision

Computer Vision Design Examples

Design Example Provided by Xilinx	Latest SDSoC Version Supported	Board & SOM Supported	Provider
LK Dense Optical Flow iterative and pyramidal based implementation doing motion segmentation	2017.1	ZCU102, ZC702, ZC706	Xilinx
Stereo Disparity Map Calculates disparity map from two sensor inputs using local block matching	2017.1	ZCU102, ZC702, ZC706	Xilinx
Warp Transform	2017.1	ZCU102, ZC702, ZC706	Xilinx
Harris Corner	2017.1	ZCU102, ZC702, ZC706	Xilinx
Bilateral Filter	2017.1	ZCU102, ZC702, ZC706	Xilinx

资源

- INT8 白皮书
- 机器学习白皮书
- reVISION 背景资料
- 更多白皮书与辅导资料
- 赛灵思嵌入式视觉视频
- 论坛

如需获取所有这些及更多资料，敬请访问：china.xilinx.com/reVISION