预防性维护

*“在任何设计中，现在都必须考虑现场维护，因为从长期看，它往往超出所有其他成本。设计的系统越复杂，最终设计中现场维护的地位就越重要。只有将现场维护作为原始设计的一部分，才能对其进行安全控制；在后期想办法应付它不是明智的做法。*这既适用于机械，也适用于人类组织机构。

Richard W. Hamming，**1997 年** - 科研与工程设计的艺术：学会学习

Trevor Weng
ISM 市场经理

XILINX

关注赛灵思
获取更多研讨会资讯

本次网络研讨会的主题

- 预防性维护简介
- 在边缘实施预防性维护的步骤
- 用例示例
- 设置边缘系统 – ZYNQ® 和 ZYNQ Ultrascale+™
- 使用 PYNQ™ 框架开展快速原型设计
- 将赛灵思边缘 AI 解决方案用于神经网络
- 未来的边缘系统 – VERSAL™ A.I.

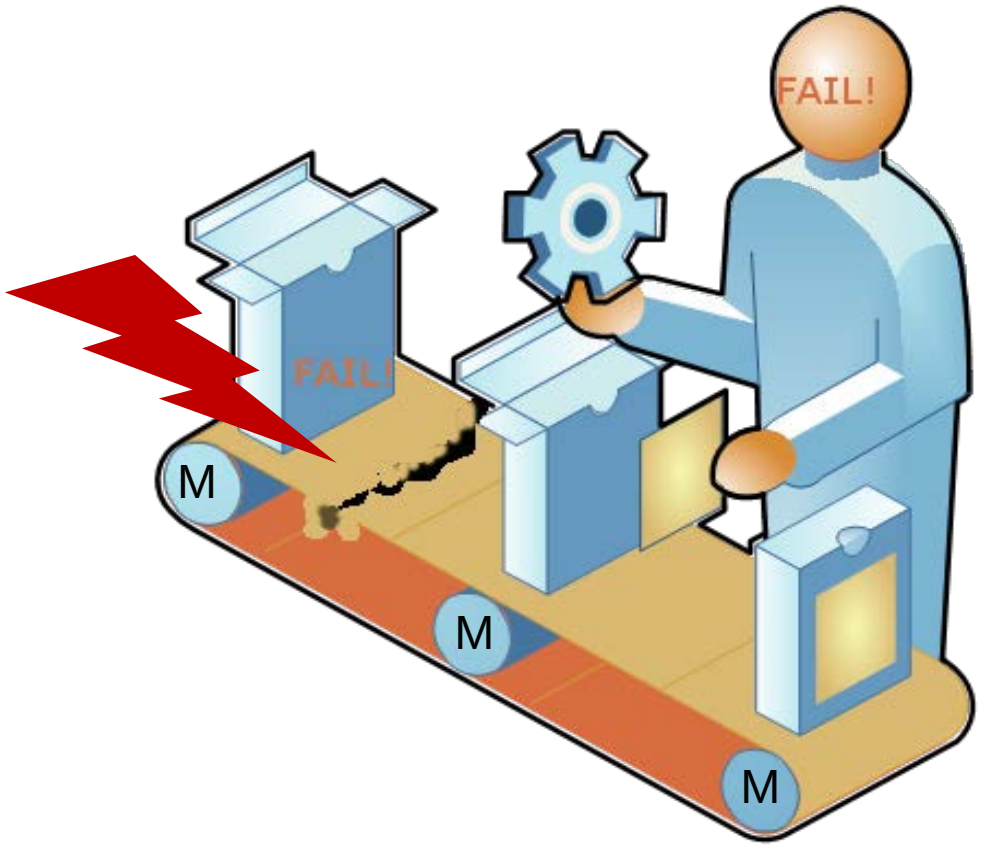
关注赛灵思
获取更多研讨会资讯

出现故障再维护

- 当机器设备发生故障时进行维修
- 如果没有发生故障就置之不顾

- 被动式管理方法“灭火”
- 最昂贵的维护管理
  - 计划外停工
  - 呼叫应急响应维修团队

很差

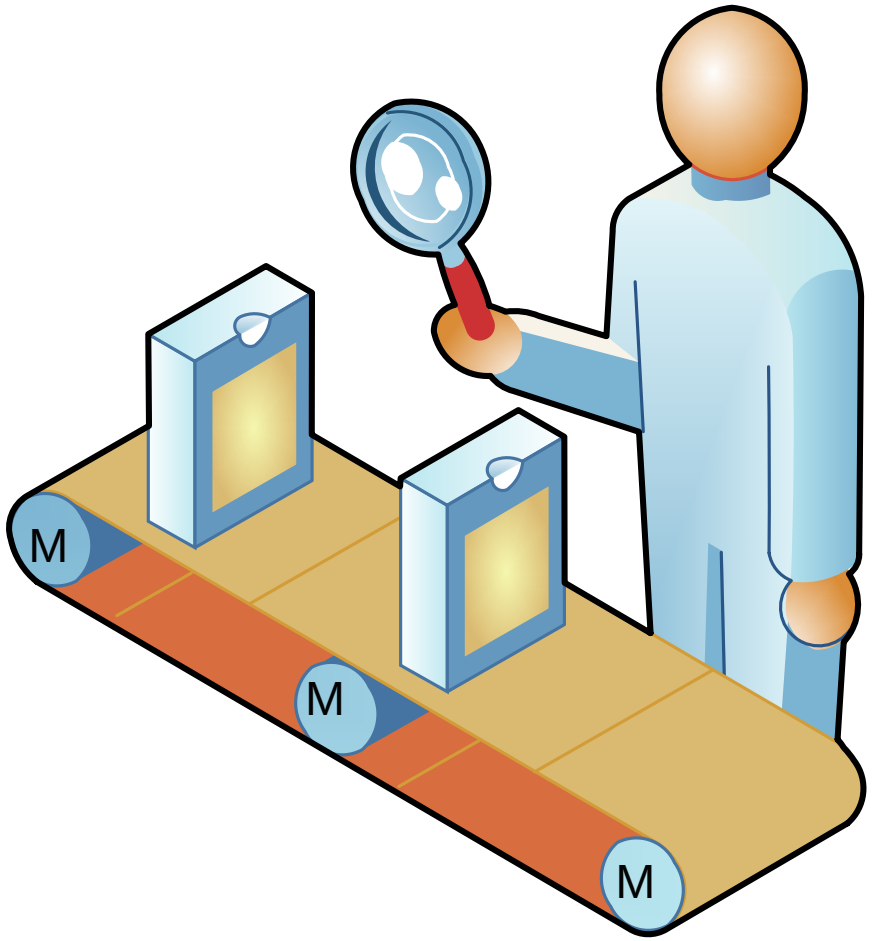


定期维护

- 使用**平均故障间隔时间 (MTTF)** 数据，根据时间进行维护
- 确定可能发生故障的关键资产并估算 MTTF
- 定期开展资产检验（保证试验）
- 产品的 MTTF 随用途变化：
  - 泵送水
  - 泵送盐水
  - 泵送脏水

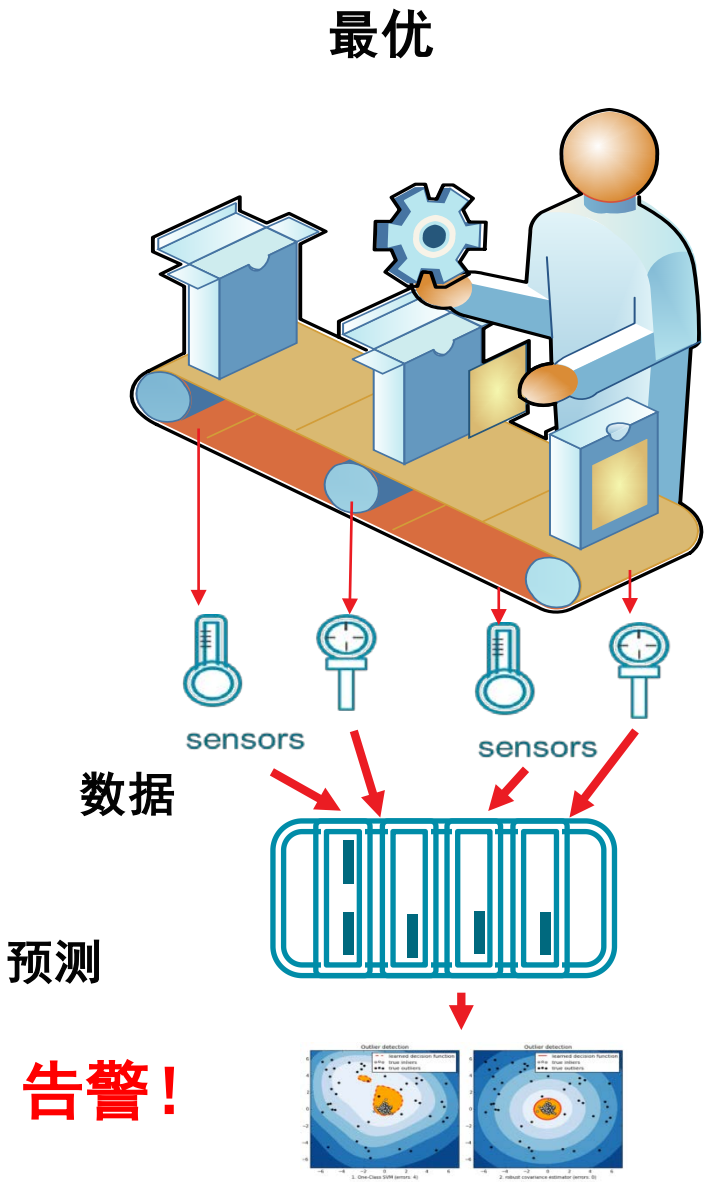
  不同的 MTBF
- 如果过早开展，可能因成本高昂导致停运
- 如果使用通用 MTBF，可能发生灾难性故障

成本高昂
非必要维修



预防性维护

- 及时执行维护的方法
- 额外效益
  - 最佳可用性
  - 最佳运行条件
  - 最佳维护资源利用率
  - 最低备品备件库存
- 使用传感器持续采集资产数据
  - 传感器已在系统中运行
  - 专为维护部署的传感器
- 使用预测算法估测维护
  - 基于模型
  - 基于规则
  - 基于机器学习



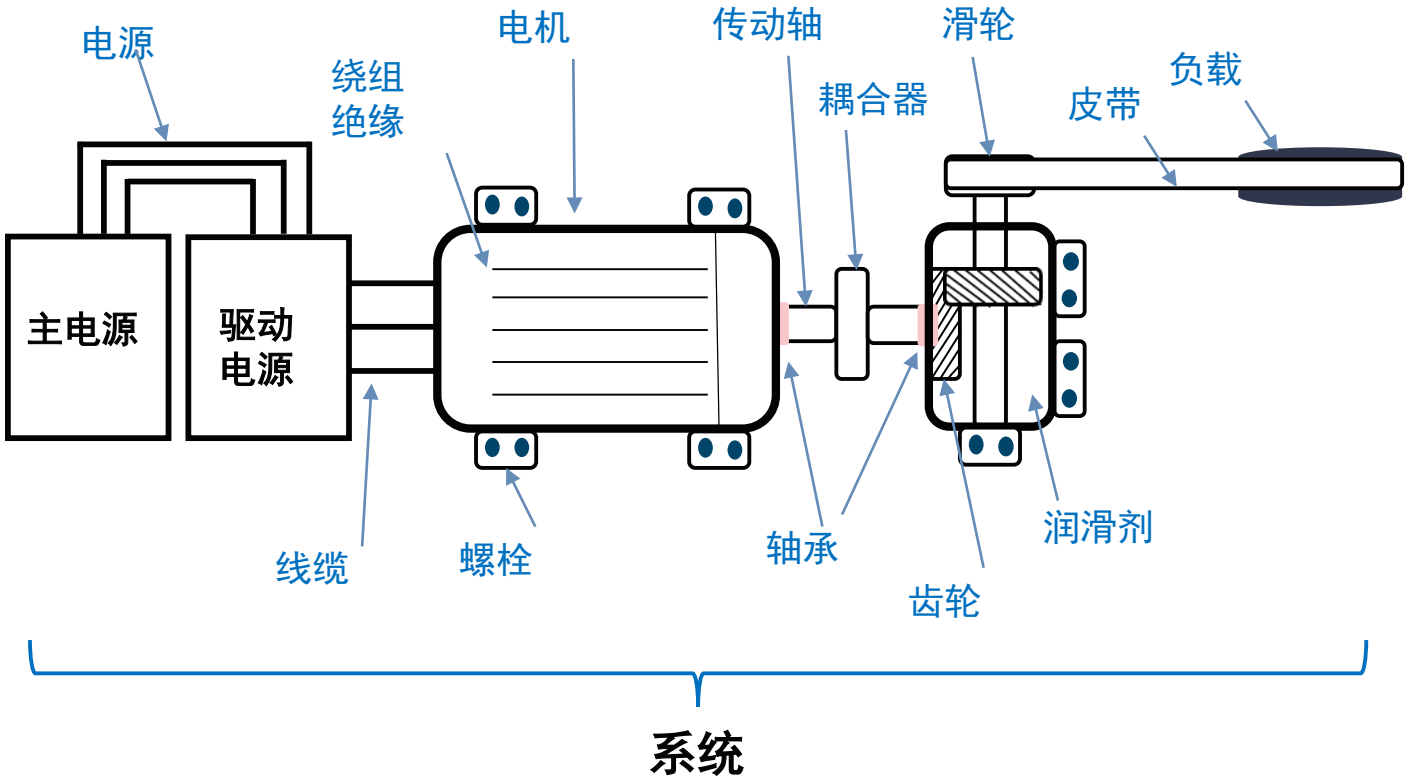
如何实现预测？- 7 个步骤

步骤
1. 确定系统
2. 确定故障 a) 故障源 b) 故障表现 = 故障 c) 故障模型
3. 确定所需数据及其可用性 a) 重点关注现有的可用数据源 b) 从系统提取数据
4. 采集选定数据 a) 梳理数据 b) 组织特性
5. 确定预测算法 a) 有监督/无监督
6. 选择分析数据的位置
7. 提取洞察并预测时间

用例 – 动力传动装置

XILINX

第 1 步：确定系统

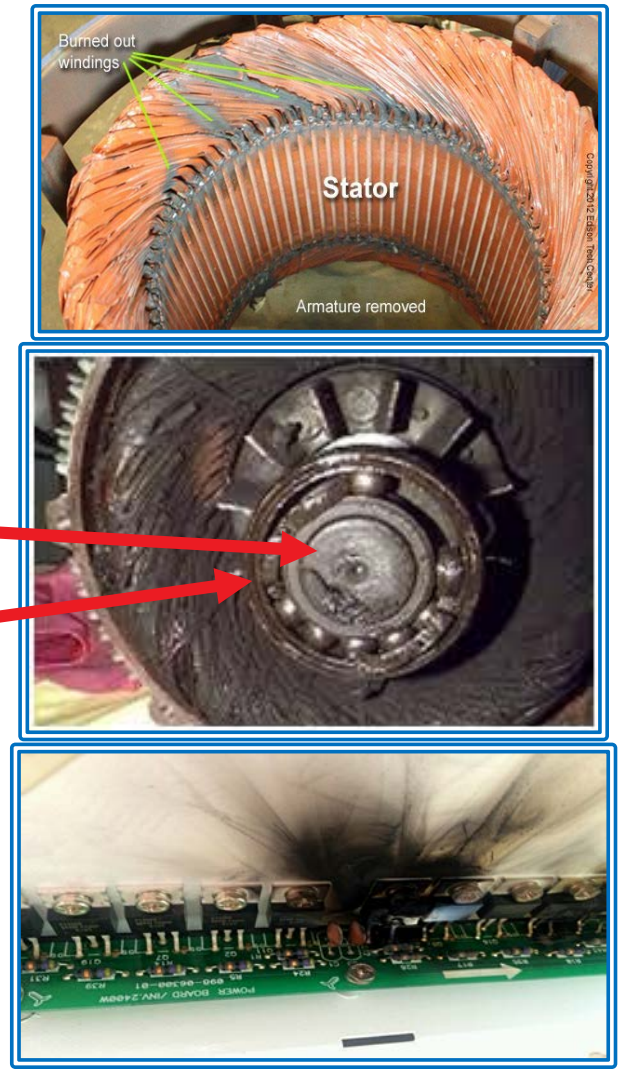


关注赛灵思
获取更多研讨会资讯

XILINX

第 2 步：- 故障源

- **电气故障**
  - 电机绕组断路或短路
  - 绝缘性能劣化
  - 导体接触电阻大
  - 错误接地或不稳定接地
- **机械故障**
  - 转子断条或磁铁断裂
  - 短路环开裂
  - 传动轴弯曲
  - 螺栓松动
  - 轴承故障
  - 齿轮箱故障
- **外部电机驱动系统故障**
  - 反相器系统故障
  - 电源电压/电流不稳
  - 电源线短路/断路



众多故障模式

大量数据集针对故障行为与正常行为

XILINX

第 3 步：确定所需数据及其可用性

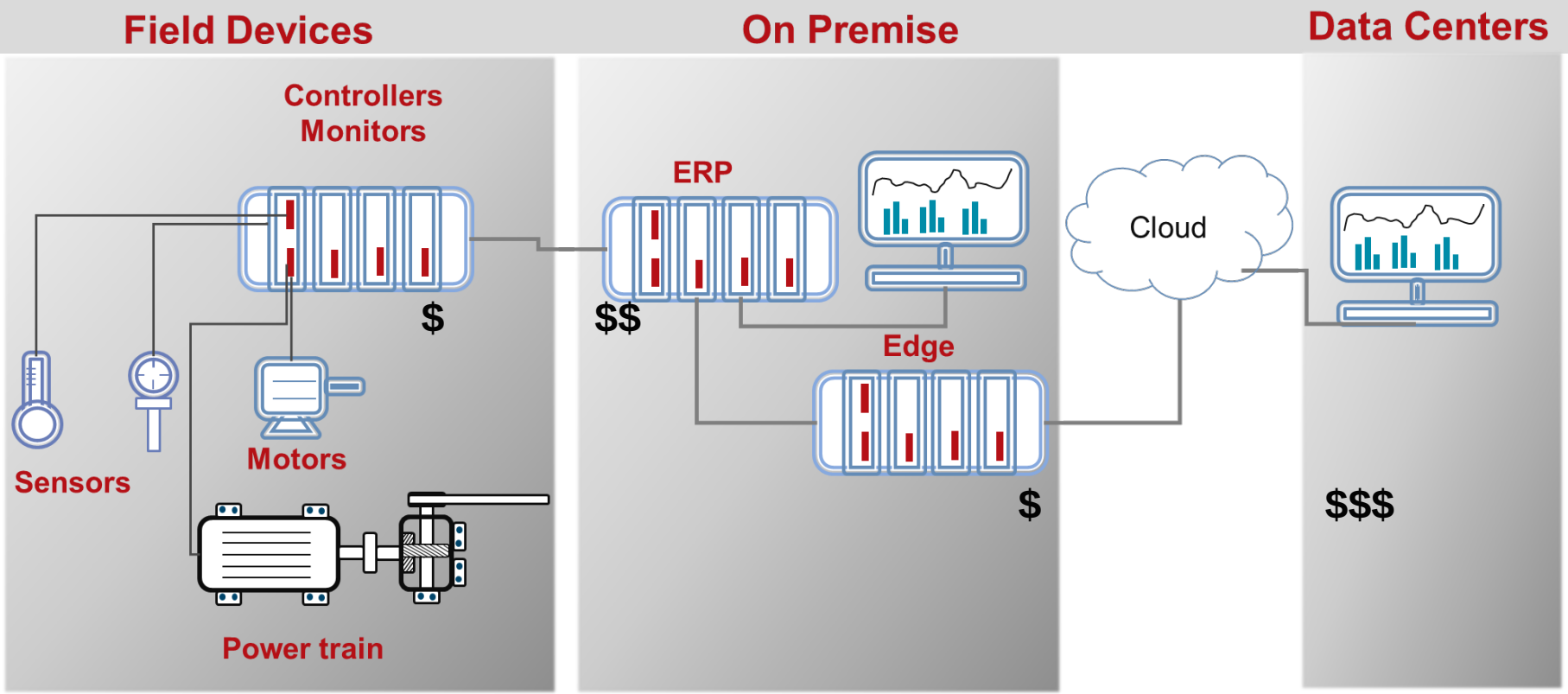
实时数据
可用且成本低

数据可用且成本低
实时数据
可用但成本较高

数据可用但成本较高

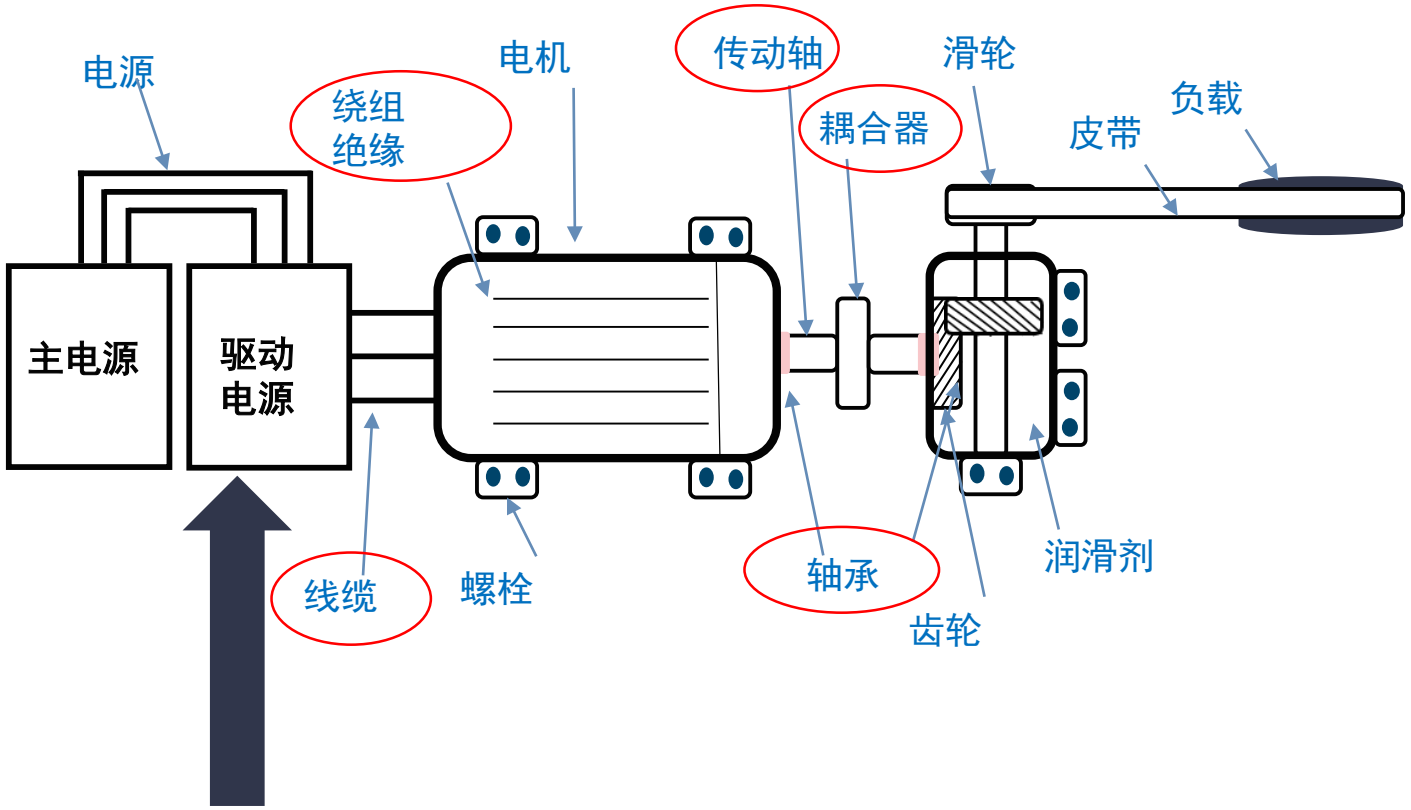
类型	实时 – 大量 结构化与非结构化	批量 – 大数据 结构化与非结构化	异构 – 大数据 结构化与非结构化

来源



XILINX

第 3.1 步 - 能否使用现有数据？

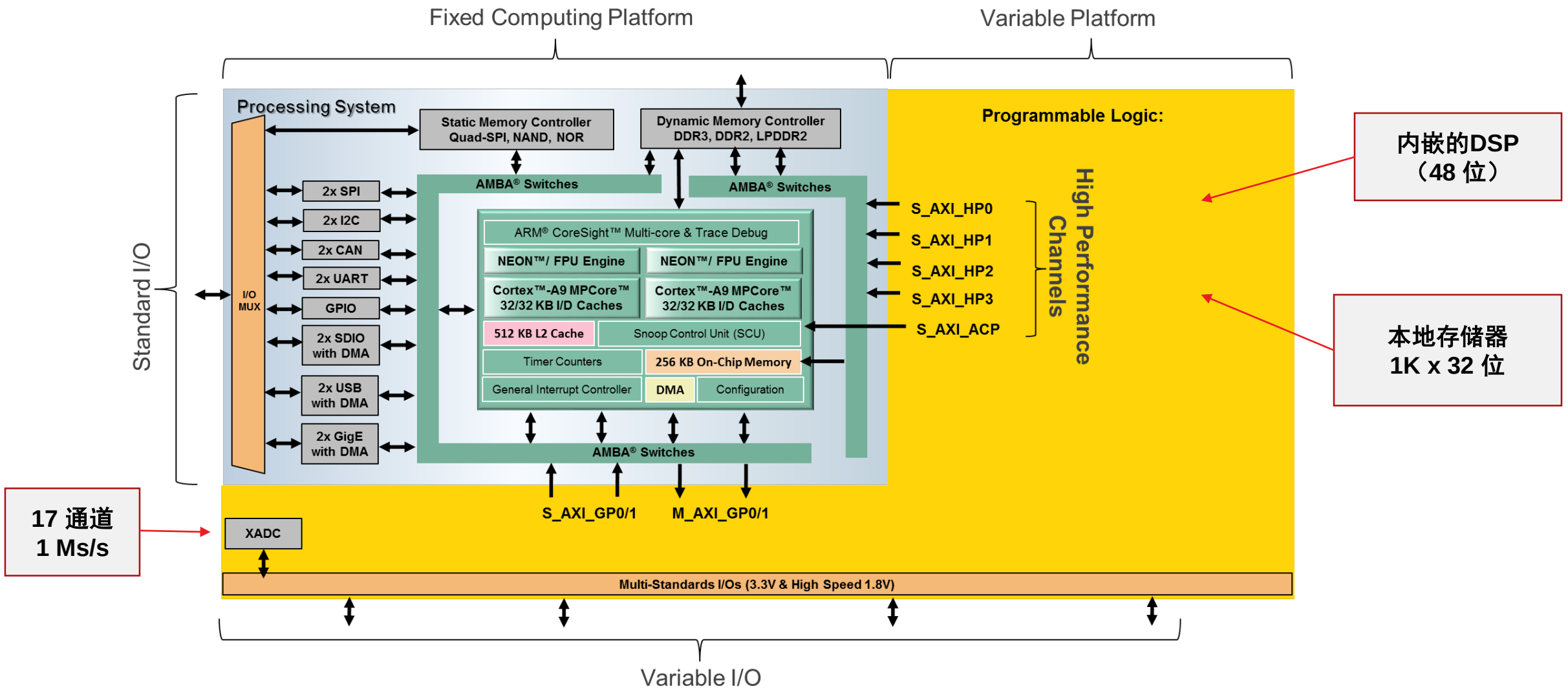


使用驱动变量开展预测

- ***电机是一种可逆机械***
  - *电力变动力（电动机）*
  - *动力变电力（发电机）*
- ***能否使用电机的测量数据开展预测？***
  - *可以（红圈标出的是可检测项）*
  - *机械扰动产生电气扰动*
  - *在用于控制的测量电流中可见电气扰动*
  - *在适当的时间内收集电机电流，为预测提供数据集。*

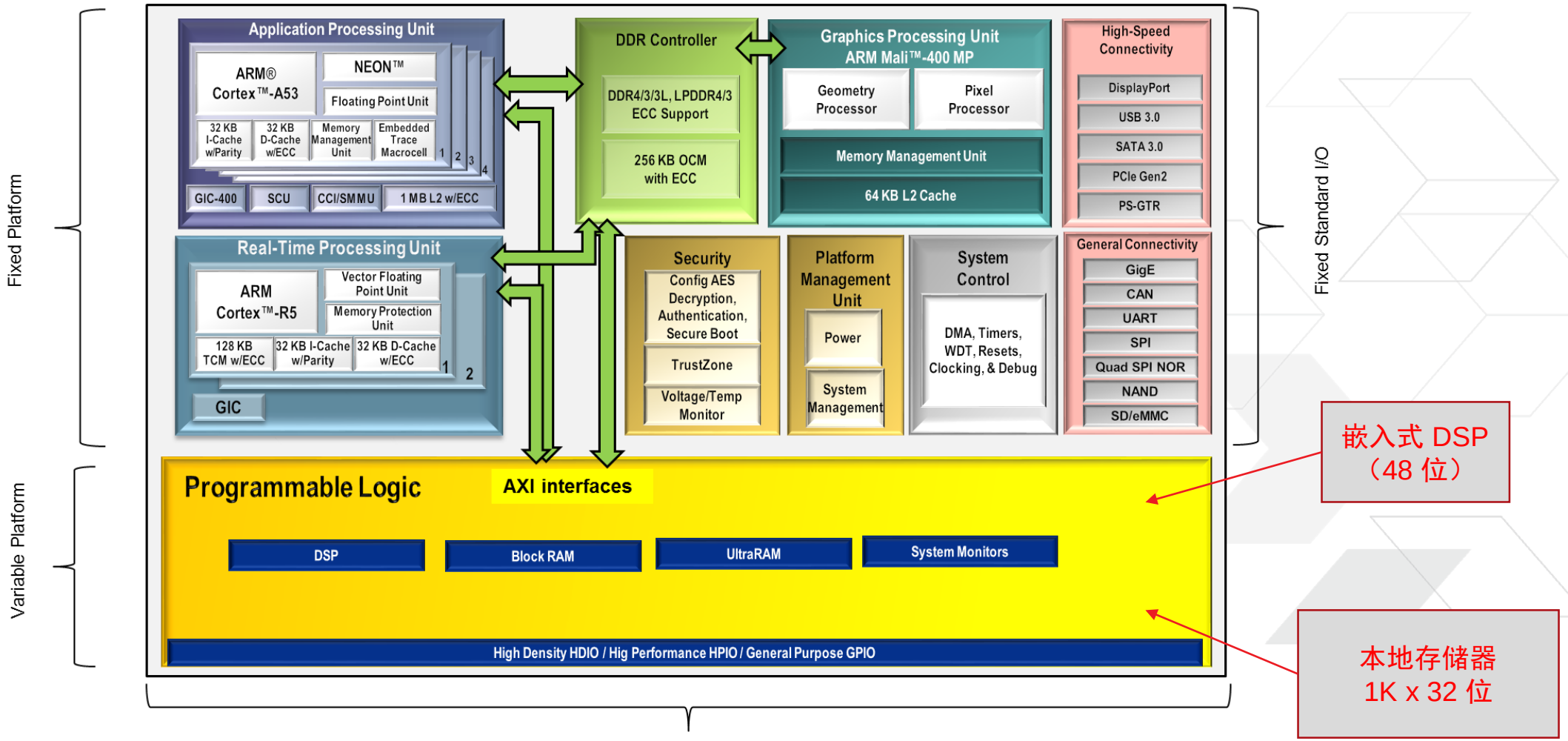
第 4 步：- 如何采集数据？

采用哪个平台？Zynq-7000 All Programmable SoC *1 代*



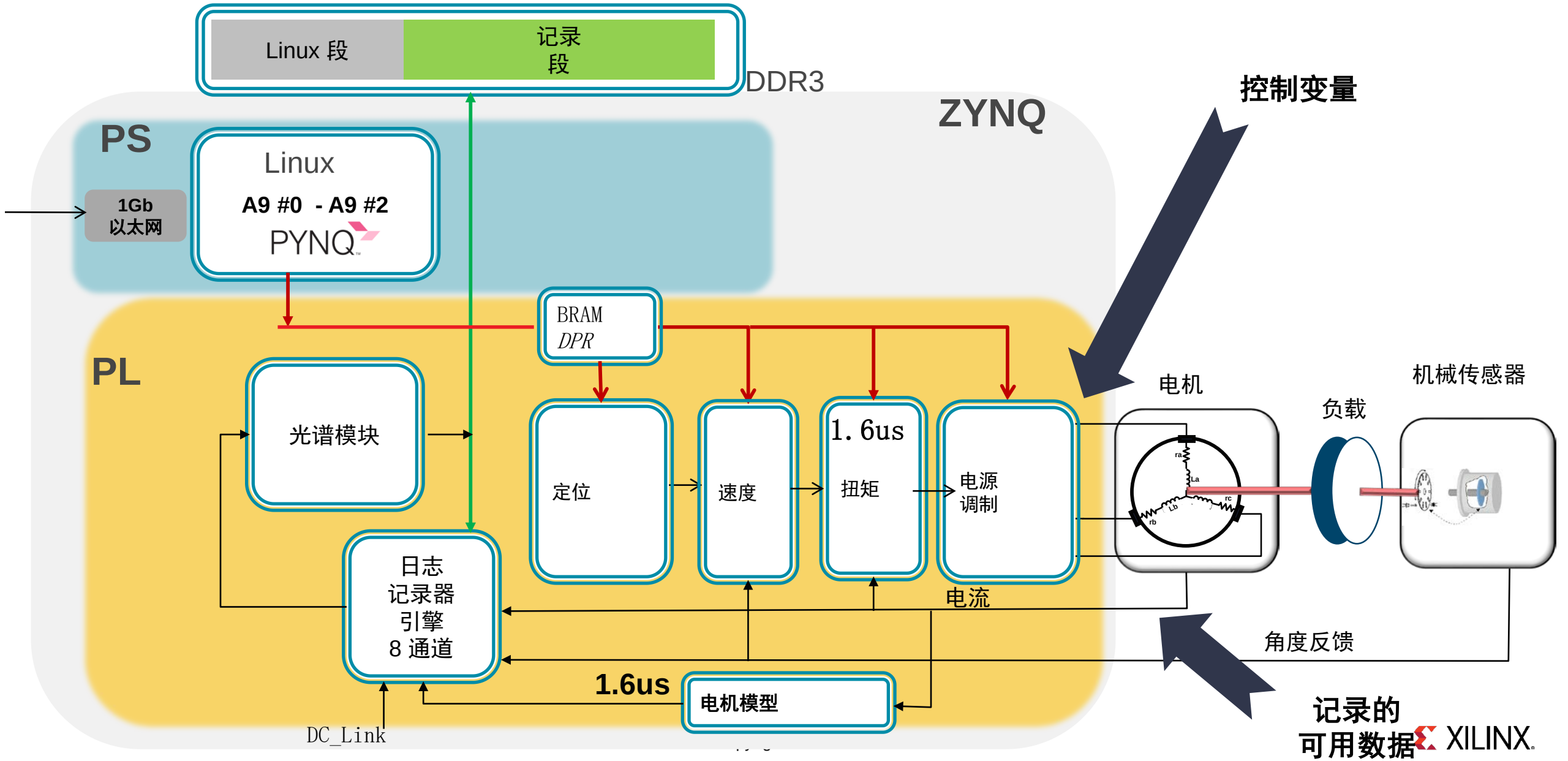
XILINX

采用哪个平台？ZYNQ Ultrascale + *2 代*

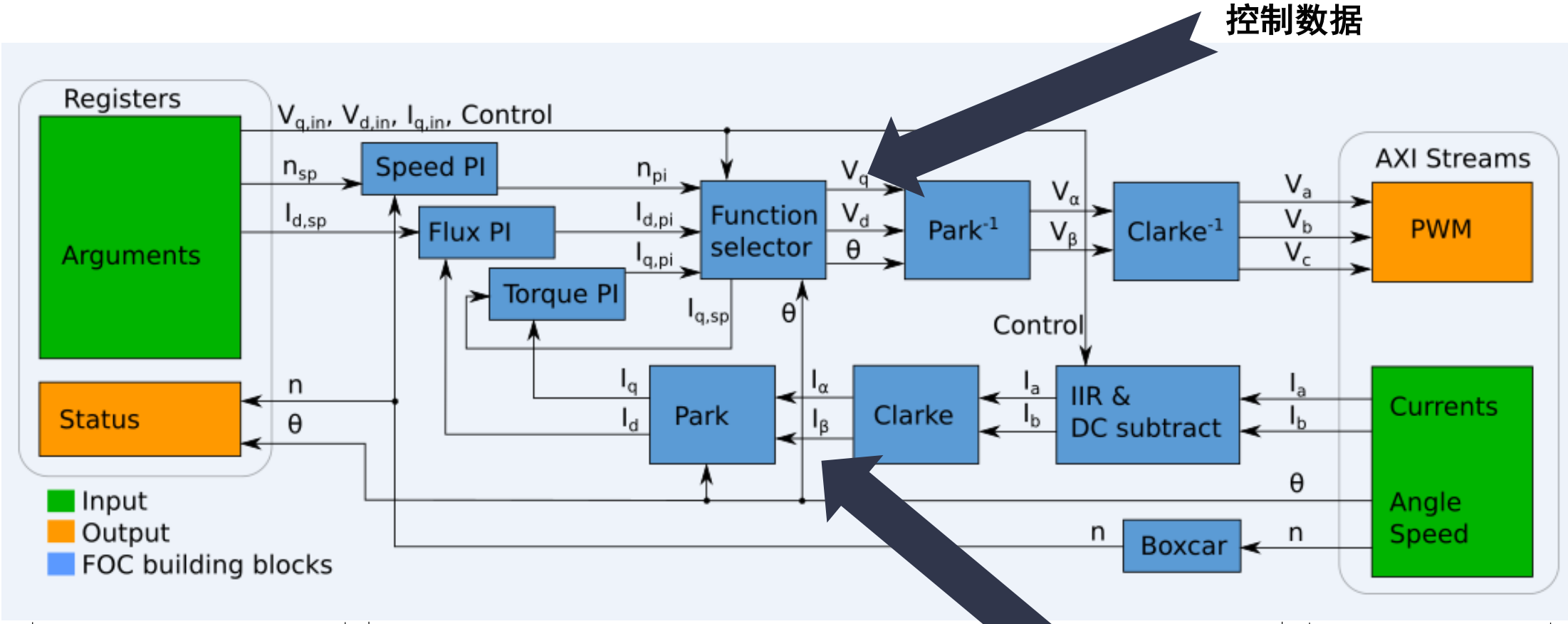


XILINX

电气驱动获得的变量 + 推断



硬件电机控制场定向算法



XILINX.

第 4b 步：如何结构化数据？

```
"meta" : {
    "view" : {
      "id" : "dq-tlimot-1",
      “Name" : "Alpaha Beta logging",
      "averageRating" : 0,
      "category" : "Diagnostic Motor",
      "description" : "This dataset contains currents logging of Rimfire .15 Motor
      "Date" : "Wednesday, July 27, 2016"
      "Time" : "6:29:10 PM"
      "Manufacturer" : "ACME"
      "Motor" : "EF150"
      "windings" : "bldc"
      "Paipol" : "2"
      "enc/p" : "False"
      "enc/r" : "0"
      "rpm max" : "38000"
      "DC_link" : "24.35"
      "Pwm Freq Hz" : "42000.00"
      "Pwm Mod %" : "85.75:
      "Pwm gear" : "False"
      "RPFM PW" : "1.6E-06"
      "RPFM 0/7 Nice" : "True"
      "RPFM   7 Hold" : "True"
      "RPFM Mod %" : "83.08"
      "RPFM gear" : "True"
      "RPM" : "38401.023"
      "Ph1 0->1/sec" : "43491.4"
      "Hz" : "4030.335"
      "tSample" : "1.6E-06"
      "n-sample" : "819200"
      "displayType" : "table",
 "data" : [ [ 1.76561, 0.63945, 0.29209, 0.63155, 0.30788, 24.12],
            [ 1.76753, 0.64221, 0.31204, 0.63455, 0.30932, 24.14],
            [ 1.76912, 0.64424, 0.32945, 0.63945, 0.31045, 24.07],
```

元数据

数据

- *用下列元素组织数据*
  - *元数据*
    - *描述符*
    - *日期*
    - *时间*
  - *数据（示例）*
    - *电流 (ia,ib,id,iq)*
    - *电压 (Dc_link)*
    - *角度（机械）*

第 5 步：采用哪种流程？

- ***有监督***
  - *我们了解特性*
    - *有预期的结果*
    - *已添加数据标签*
    - *发生时间*
  - *我们了解系统*
    - *有可用模型*
    - *可推断模型*
  - *我们可以使用.....*
    - *DNN / CNN*
    - *决策树*
    - *分类器*

- ***无监督***
  - *不了解输出*
    - *确定模式或分组*
    - *未添加数据标签*
    - *时间可能未知*
  - *自主引导算法.....*
    - *K-学习*
    - *自动编码器*
    - *生成式对抗网络*

第 5 步：我们采用有监督方法

取电流 iα、iβ

↓

在直角坐标系内绘制 iα、iβ

↓

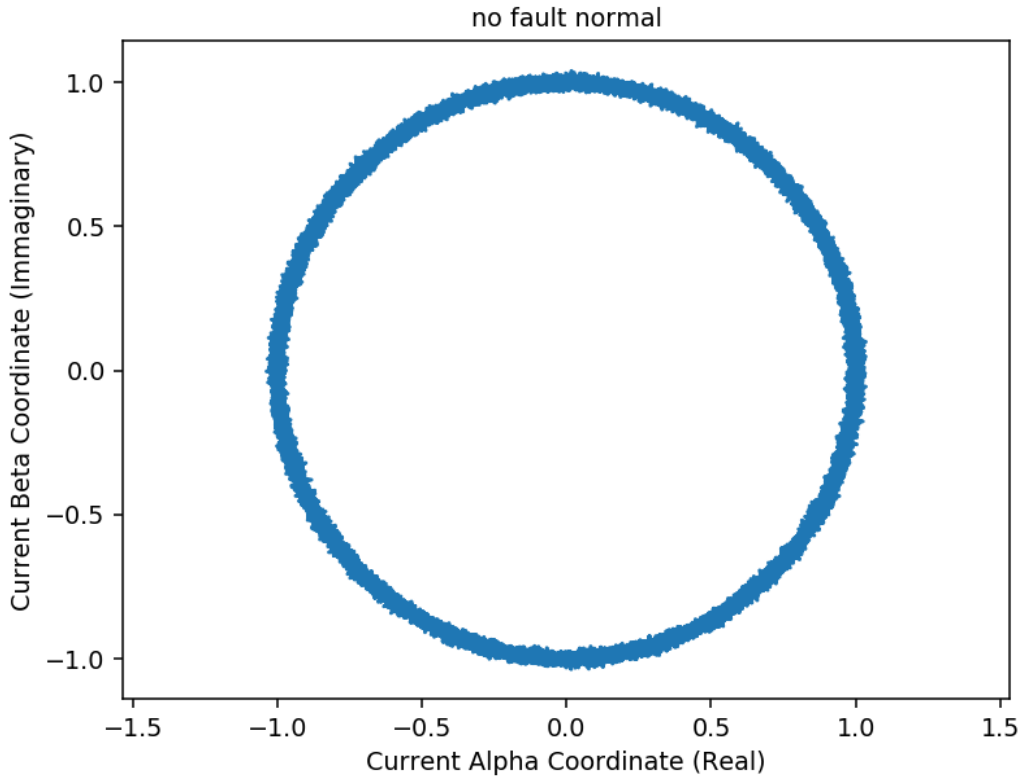
直角坐标系图体现正常/扰动条件

↓

保存直角坐标系图

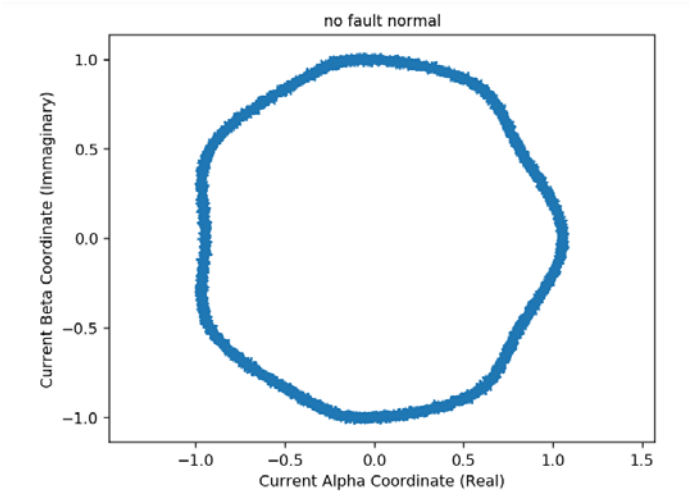
↓

使用 CNN 开展“图像”分类

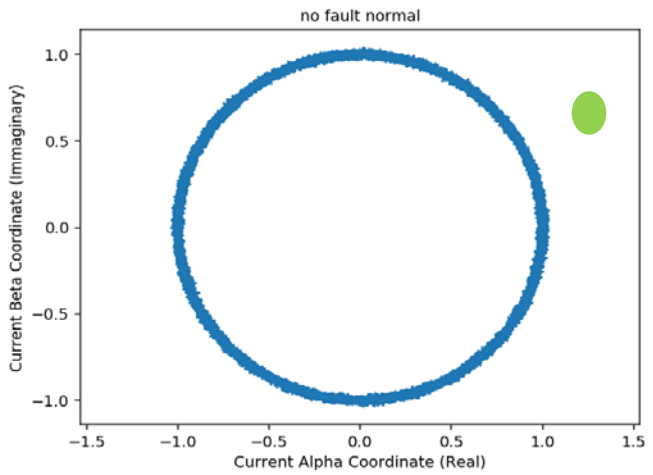


使用 CNN 将“无视觉”转化为“有视觉”

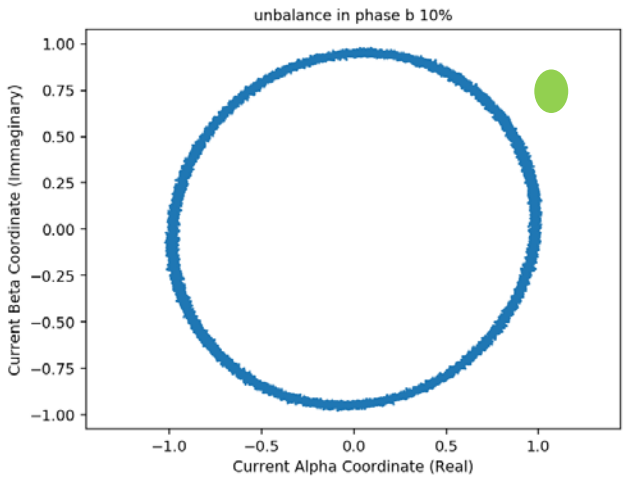
第 5 步 – 部分 α、β 图示电流 –（分辨率 512 x 512）



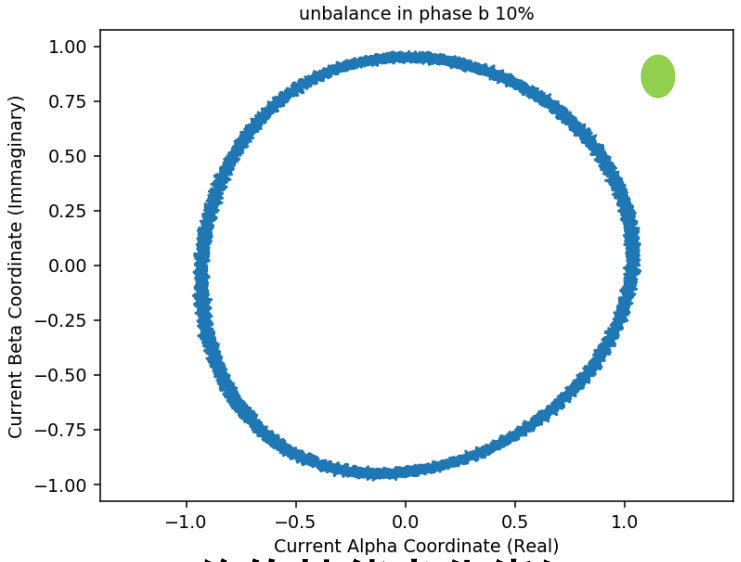
有凸极效应的正常电机



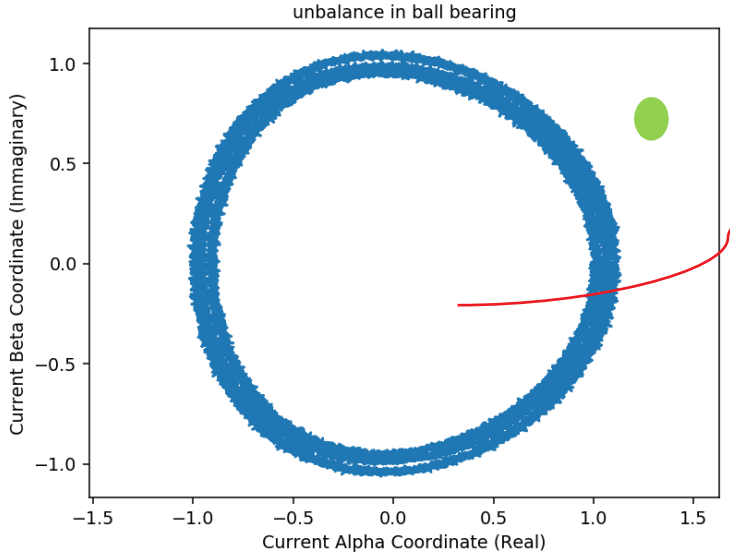
无凸极效应的正常电机



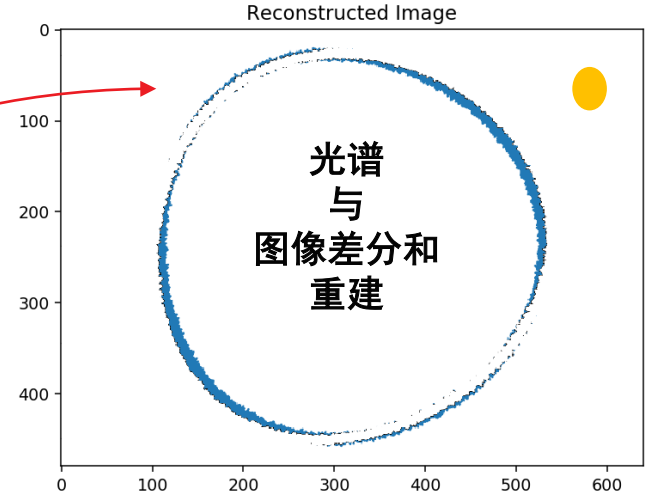
相 b的接触电阻较高



绝缘性能劣化绕组 B



齿轮性能劣化与绝缘性能劣化绕组 B



齿轮性能劣化与绝缘性能劣化绕组 B

XILINX

数据是否齐备？
不，我们人为的增强数据

关注赛灵思
获取更多研讨会资讯

XILINX

第 5 步：生成约 35 维的数据集（举例）

电机故障模式 → 滚珠轴承故障模式 → 传动故障模式 → 光谱 / α、β 图形

$$\begin{bmatrix} v_\alpha \\ v_\beta \\ 0 \end{bmatrix} = \begin{bmatrix} R_s & 0 & -R'_{a2} \\ 0 & R_s & 0 \\ -R'_{a2} & 0 & R'_f \end{bmatrix} \begin{bmatrix} I_\alpha \\ I_\beta \\ I_f \end{bmatrix} + \begin{bmatrix} L_s & 0 & M_{f\alpha} \\ 0 & L_s & M_{f\beta} \\ M_{f\alpha} & M_{f\beta} & L_{a2} \end{bmatrix} \frac{d}{dt} \begin{bmatrix} I_\alpha \\ I_\beta \\ I_f \end{bmatrix} + \begin{bmatrix} e_\alpha \\ e_\beta \\ -e_f \end{bmatrix}$$

电感与电阻变化

$$f_{BPFO} = \frac{n}{2} f_r \left(1 - \frac{BD}{PD} \cos\beta\right)$$

$$f_{BPFI} = \frac{n}{2} f_r \left(1 + \frac{BD}{PD} \cos\beta\right)$$

$$f_{BSF} = \frac{PD}{BD} f_r \left[1 - \left(\frac{BD}{PD} \cos\beta\right)^2\right]$$

$$f_{FTF} = \frac{1}{2} f_r \left(1 - \frac{BD}{PD} \cos\beta\right)$$

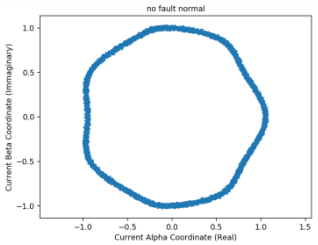
- PD 轴承节圆直径，
- BD 滚珠轴承直径
- β 接触角，
- n：滚动元件数，
- Fr：旋转速度

$$f_{pulley} = \frac{D_{motor\_pulley} \times f_r}{D_{load\_pulley}}$$

$$f_b = \frac{D_{motor\_pulley} \times \pi \times f_r}{L_{belt}}$$

- 皮带长度
- 电机滑轮

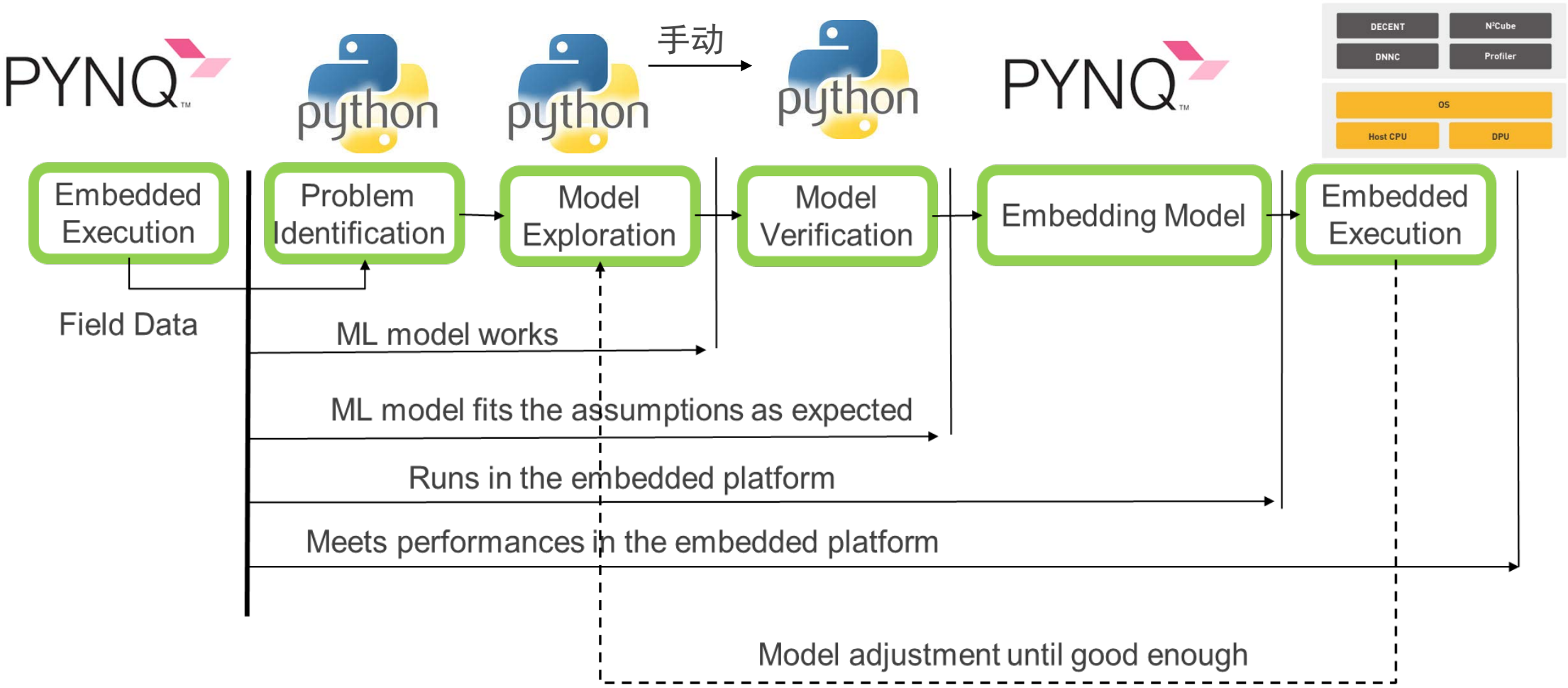
- 特性增强
- 离散化
  512 x 512



XILINX

第 5 步：何种方法？

硬件



XILINX

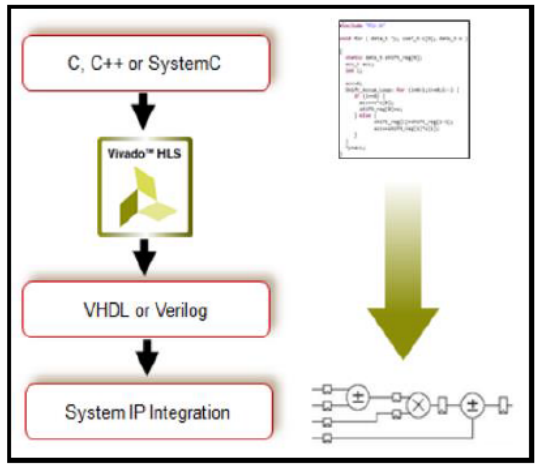
在边缘器件上使用 Python

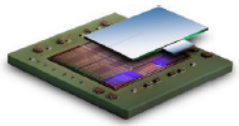
关注赛灵思
获取更多研讨会资讯

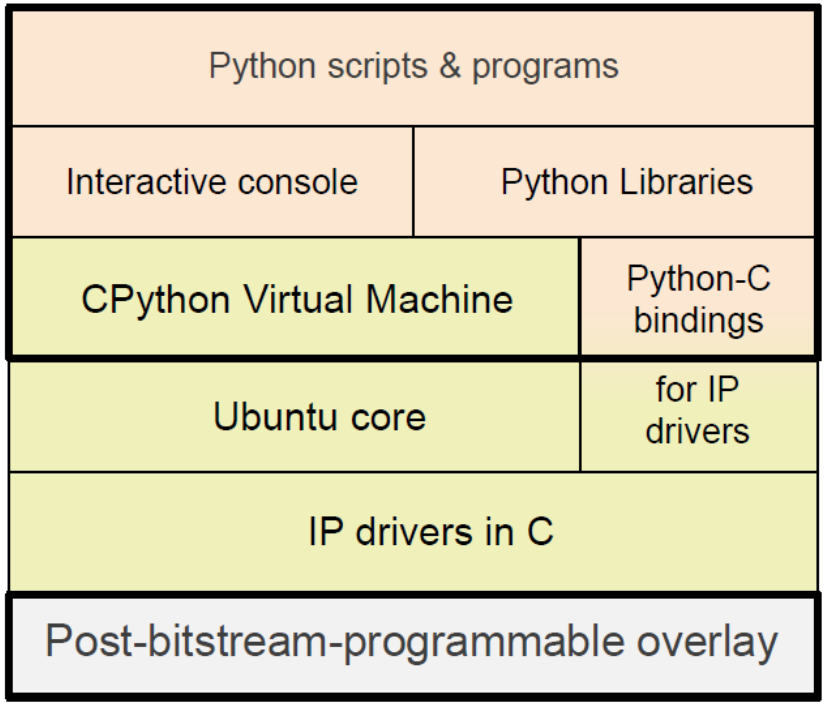
XILINX

生产力方法 PYNQ

PYNQ™ Productivity level tools for Zynq

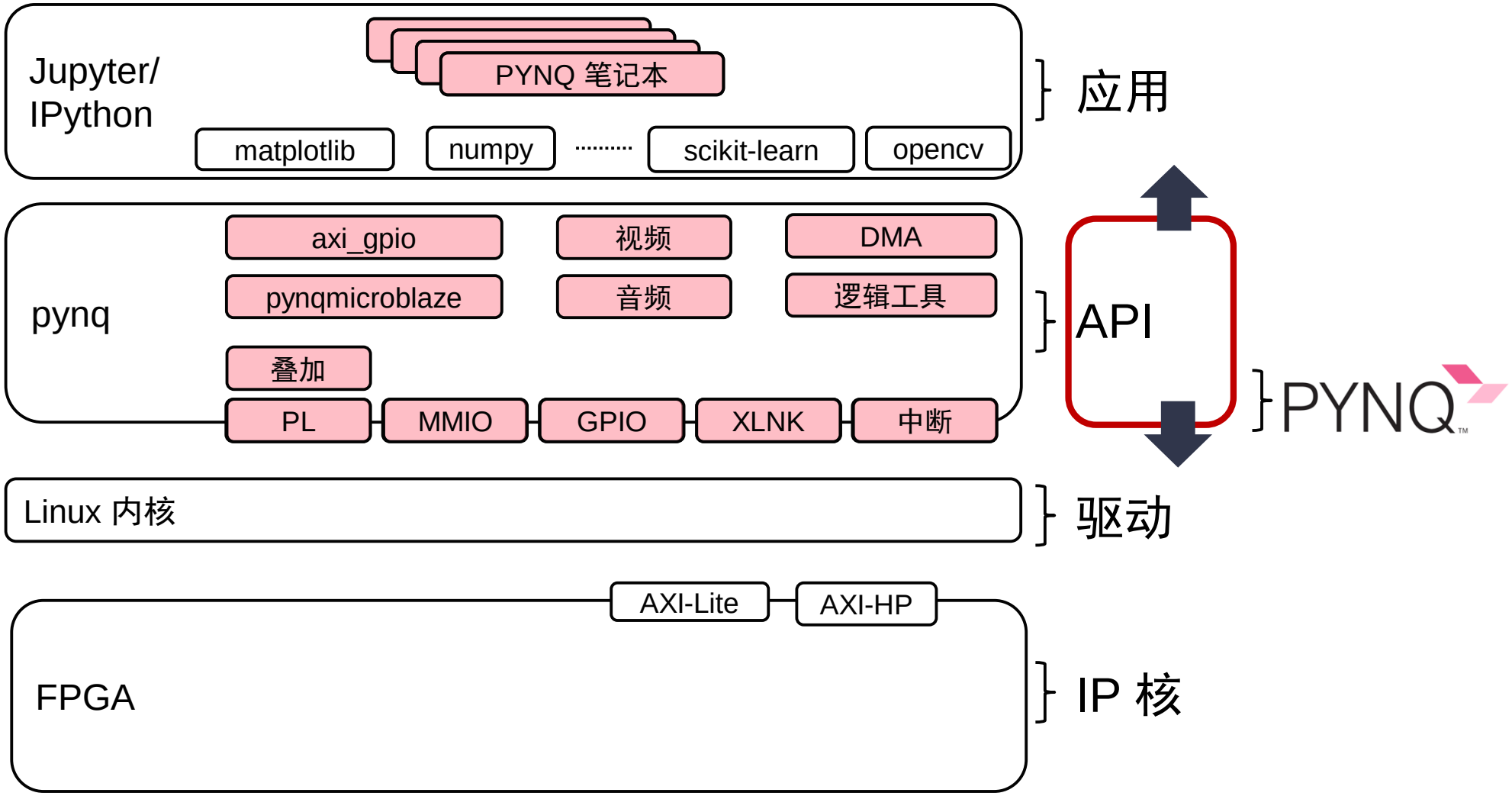






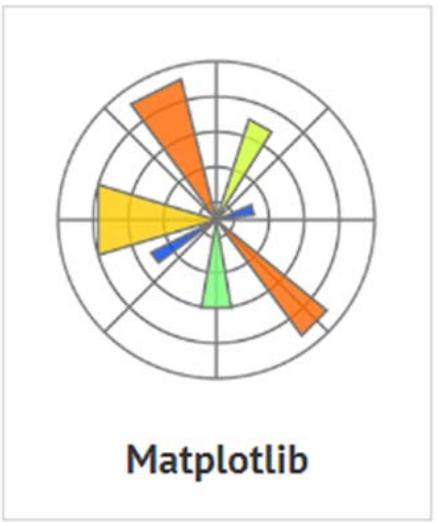
XILINX.

生产力的关键是统一的 API

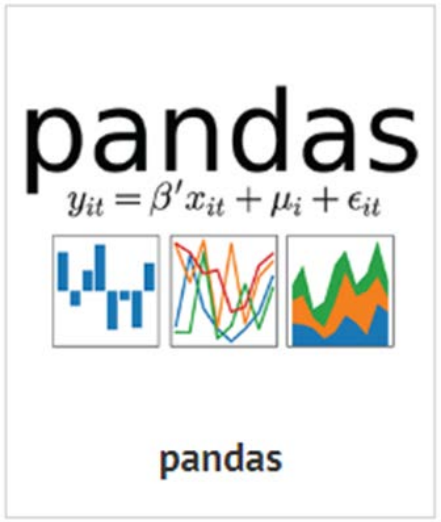


适用于所有用例的 Python 基础库

对*数值计算*使用类 Matlab™ 框架



对*静态和交互数据可视化*使用 2D 绘图程序库



对数据*摄取、转换和导出功能*使用易于使用的数据整理

获取

转换

组织

显示

设计网络

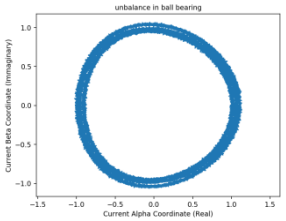
关注赛灵思
获取更多研讨会资讯

XILINX

第 6 步 – 生成 CNN

前端 Keras

后端 TensorFlow



512 x 512

Maxpooling(2,2)

Conv2D(64,(5,5))

Maxpooling(2,2)

Conv2D(10,(5,5))

Dense(100,(2,2))

Dense(100,(2,2))

Dense(50,(2,2))

Dropout(0.5)

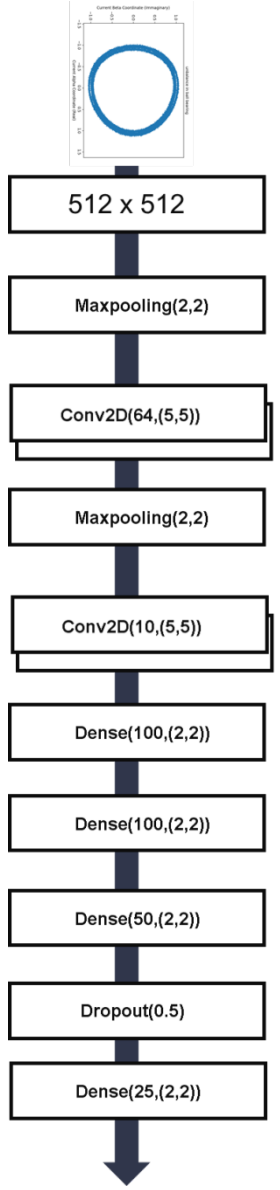
Dense(25,(2,2))

25 种故障可能 98%…99% 准确率

XILINX

使用 KERAS 框架在 Jupyter Notebook 中建模

```
model = Sequential()
model.add(Conv2D(5, KERNEL_SIZE,
                 input_shape=IMAGES_SHAPE,
                 data_format='channels_last',
                 kernel_initializer=KERNEL_INITIALIZER,
                 padding=PADDING))
model.add(LeakyReLU(LEAK_ALPHA))

model.add(MaxPooling2D(pool_size=MAX_POOLING_POOL_SIZE))
model.add(Conv2D(64, KERNEL_SIZE,
                 kernel_initializer=KERNEL_INITIALIZER,
                 padding=PADDING))
model.add(LeakyReLU(LEAK_ALPHA))

model.add(Conv2D(10, KERNEL_SIZE,
                 kernel_initializer=KERNEL_INITIALIZER,
                 padding=PADDING))
model.add(LeakyReLU(LEAK_ALPHA))
model.add(MaxPooling2D(pool_size=MAX_POOLING_POOL_SIZE))

model.add(Conv2D(10, KERNEL_SIZE,
kernel_initializer=KERNEL_INITIALIZER,
padding=PADDING))
model.add(LeakyReLU(LEAK_ALPHA))

model.add(MaxPooling2D(pool_size=MAX_POOLING_POOL_SIZE))

model.add(Flatten())

model.add(Dense(100))
model.add(LeakyReLU(LEAK_ALPHA))

model.add(Dense(50))
model.add(LeakyReLU(LEAK_ALPHA))
model.add(Dropout(DROPOUT))
model.add(Dense(NUMBER_OF_CLASSES))
model.add(Activation(ACTIVATION_LAYER_FUNCTION))

model.compile(loss=LOSS_FUNCTION,
              optimizer=OPTIMIZER,
metrics=[metrics.categorical_accuracy])
model.summary()
```



在工作站或云端进行训练

XILINX

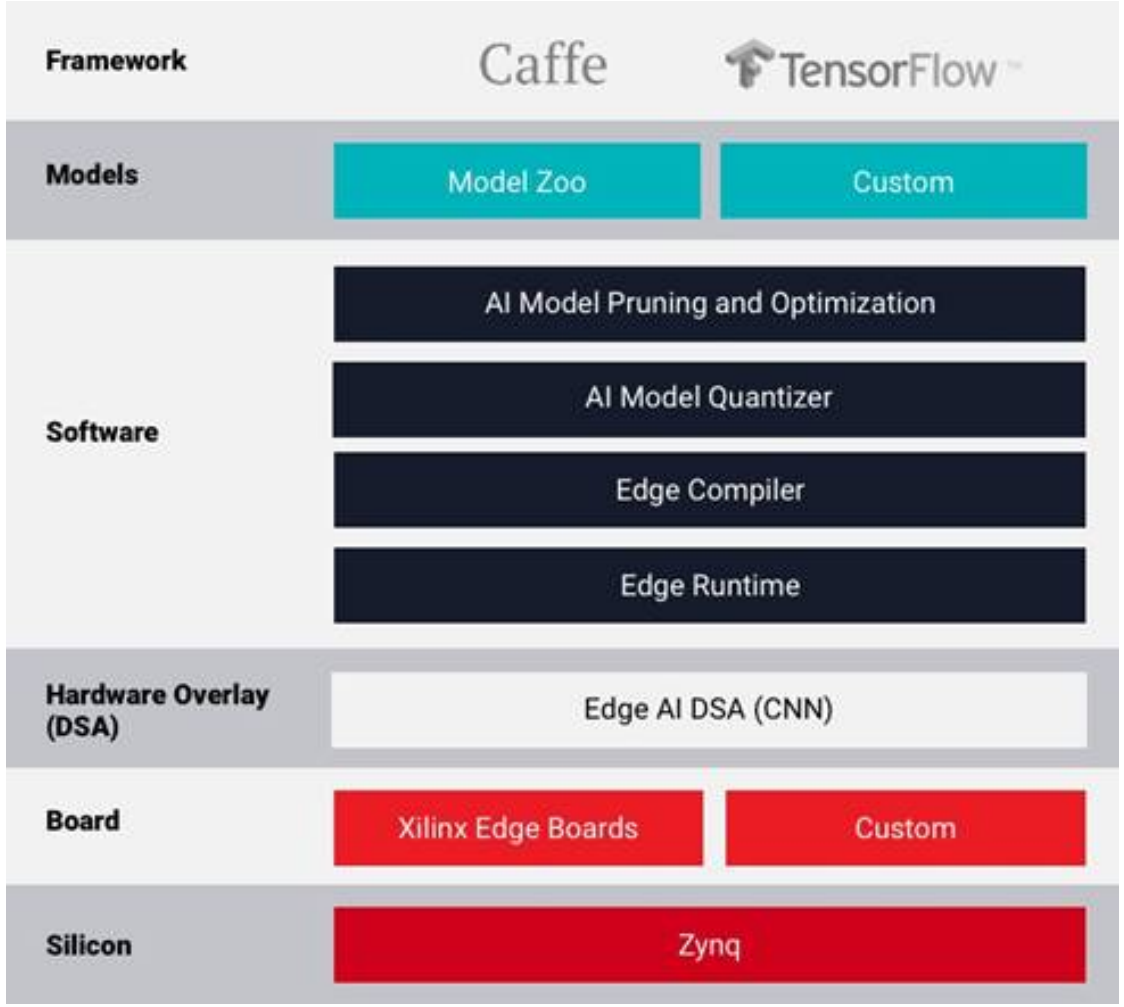
边缘器件上的推断引擎

关注赛灵思
获取更多研讨会资讯

XILINX

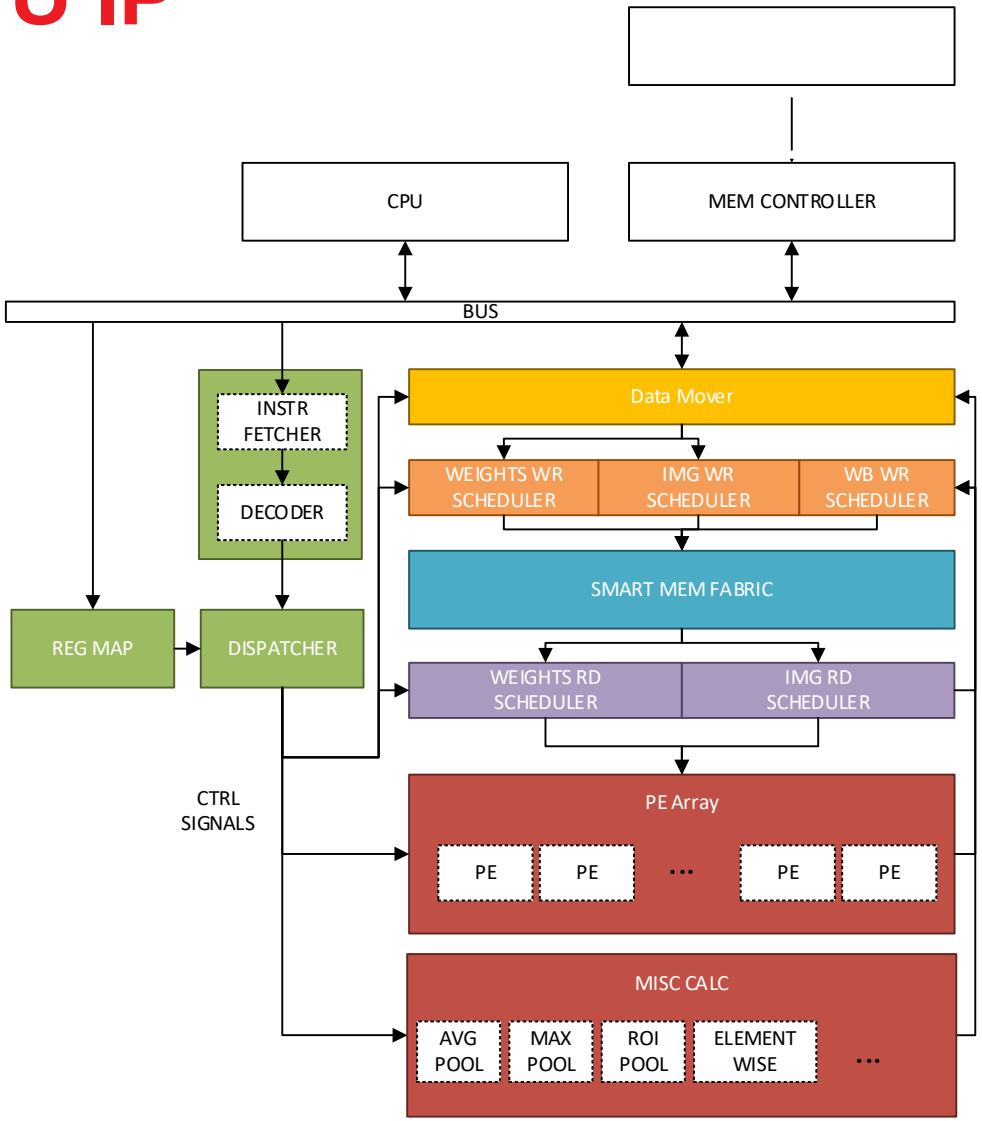
DNNDK – 深度神经网络开发套件

赛灵思边缘 AI 平台



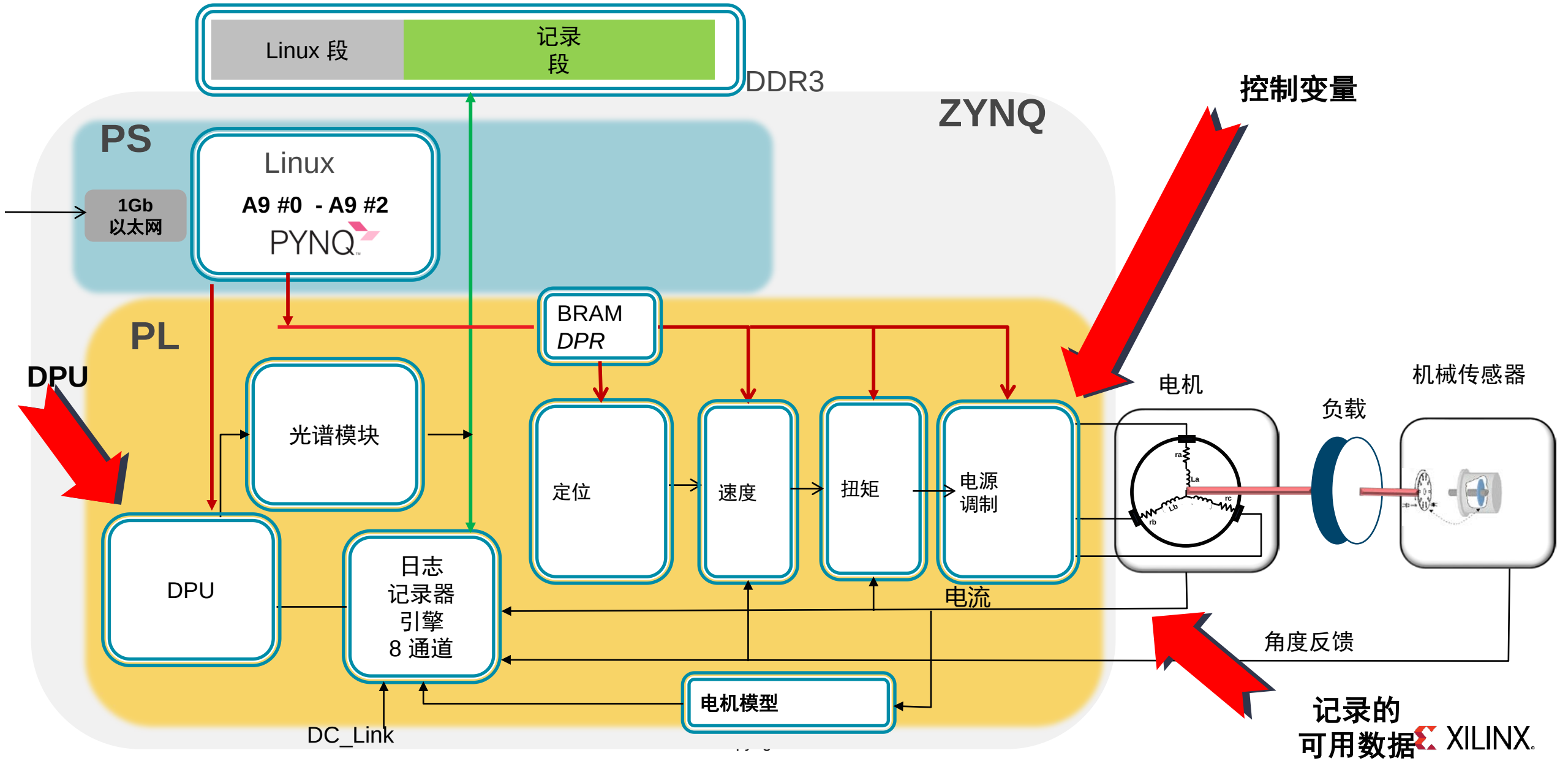
XILINX.

高效率 DPU IP



XILINX

电气驱动 + 推断 DPU


Linux 段
记录段
DDR3
ZYNQ
PS
Linux
A9 #0 - A9 #2
PYNQ
1Gb 以太网
PL
DPU
BRAM
DPR
光谱模块
DPU
日志记录器引擎
8 通道
定位
速度
扭矩
电源调制
电流
电机模型
DC_Link
控制变量
电机
负载
机械传感器
角度反馈
记录的可用数据
XILINX

所支持的运算符

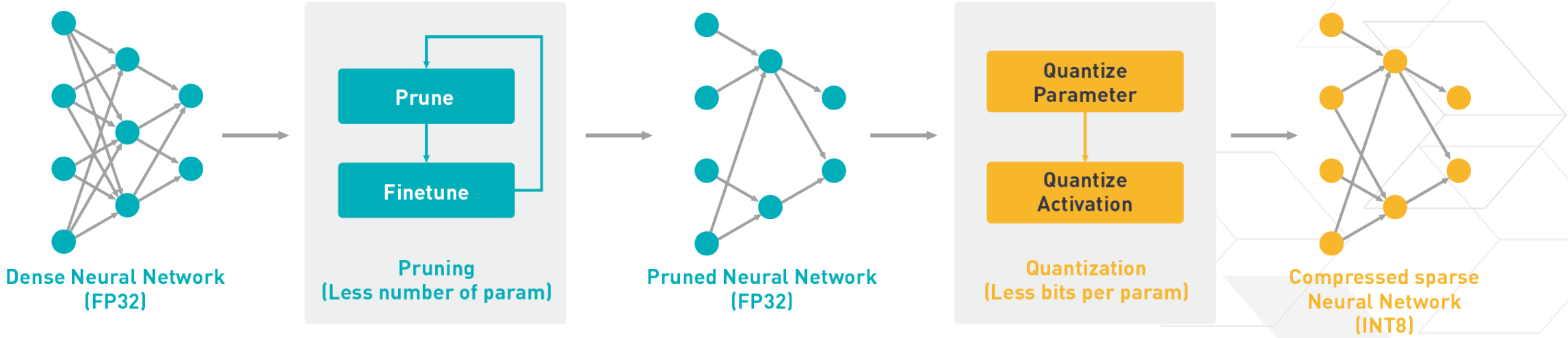
- Conv
  - Dilation
- Pooling
  - Max
  - Average
- ReLU / Leaky Relu/ Relu6
- Full Connected (FC)
- Batch Normalization
- Concat
- Elementwise
- Deconv
- Depthwise conv
- Mean scale
- Upsampling
- Split
- Reorg
- Resize (Optional)
- Softmax (Optional)
- Sigmoid (Optional)

推断引擎工具

关注赛灵思
获取更多研讨会资讯

XILINX

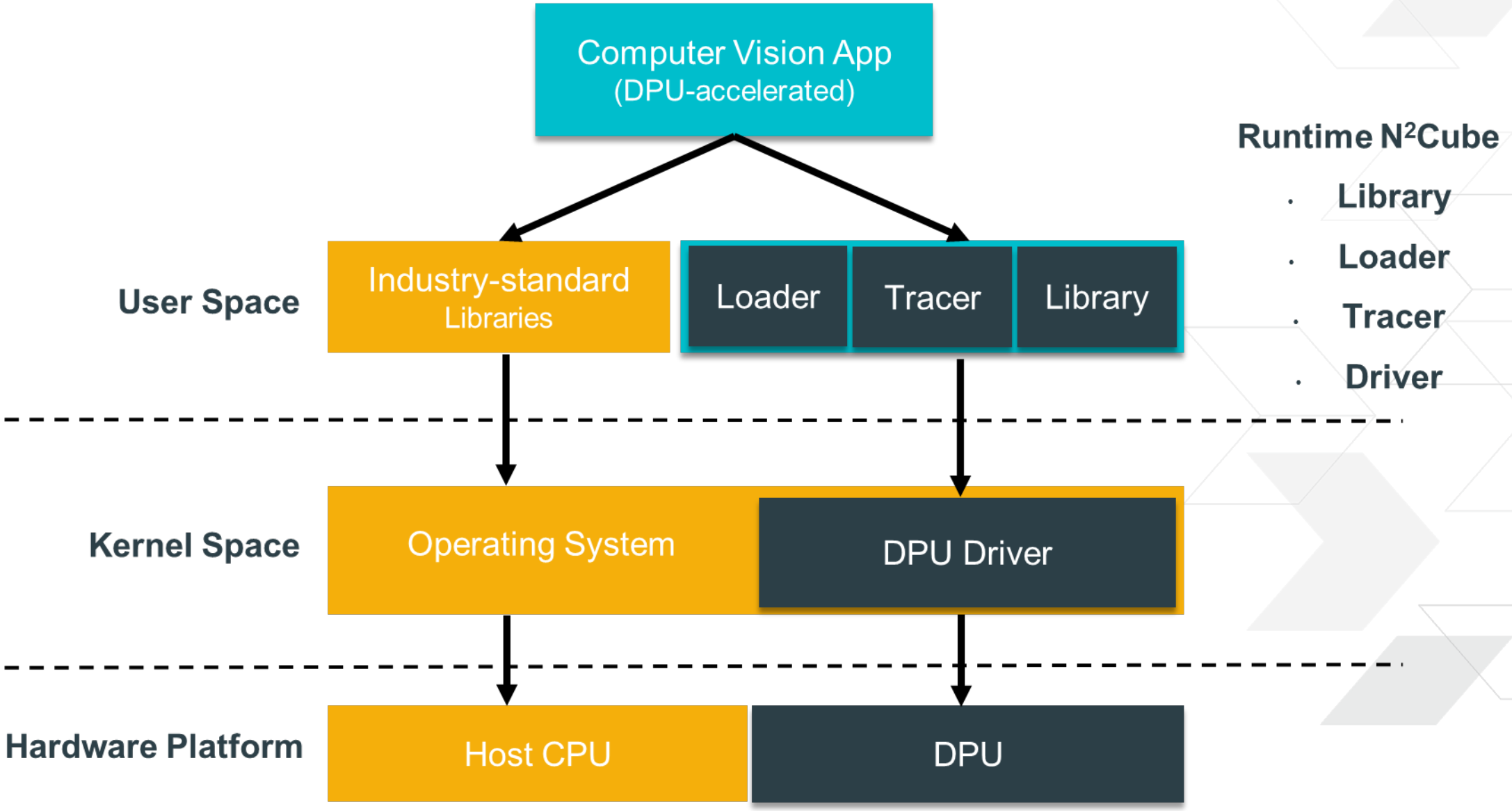
赛灵思边缘 AI 平台工具流程



HTTPS://WWW.XILINX.COM/PRODUCTS/DESIGN-TOOLS/AI-INFERENCE.HTML#FRAMEWORKS

XILINX.

赛灵思边缘 AI 平台运行时引擎



XILINX.

基础版与专业版

- **公开供货时间：**
  - 基础版：现在
  - 带有 Tensorflow 的基础版：2019 年 4 月
  - 专业版：2019 年 5 月
- **AWS 云端基础版 – 2019 年 4 月**
- **补充设计服务 – SoW**

可获取信息

关注赛灵思
获取更多研讨会资讯

XILINX

赛灵思边缘 AI

- **DNNDK 与 DPU**
  - DNNDK 基础版 - 从 Xilinx.com 下载
  - 剪枝工具单独申请
  - 可申请 DPU 用于评估与系统集成
- **演示与参考设计**
  - 通用：Resnet50、Googlenet、VGG16、SSD、Yolo v2/v3、Tiny Yolo v2/v3、Mobilenet v1/v2 等。
  - 视频监控：人脸检测与交通结构
  - ADAS/AD：多通道检测及语义分割
  - 基于ZCU102 DPU TRD
- **文档**
  - DNNDK 用户指南 – UG1327
  - 面向 SDSoC 的 DNNDK 用户指南 – UG1331
  - 边缘 AI 教程 - https://github.com/Xilinx/Edge-AI-Platform-Tutorials
  - DPU 产品指南 – PG338
- **申请或咨询**
  - 请联系 Andy Luo，andy.luo@xilinx.com

PYNQ™

- **PYNQ 框架**
  - 主站点 www.pynq.io
  - 社群互助 http://www.pynq.io/community.html
  - SPYN：用于电机控制的分析框架 https://github.com/Xilinx/IIoT-SPYN
- **平台支持**
  - PYNQ-Z1 v2.4 PYNQ 镜像
  - PYNQ-Z2 v2.4 PYNQ 镜像
  - ZCU104 v2.4 PYNQ 镜像
  - ZCU111 v2.4 PYNQ 镜像
  - Avnet Ultra96 v2.4 PYNQ 镜像
  - 其他电路板 http://www.pynq.io/board.html

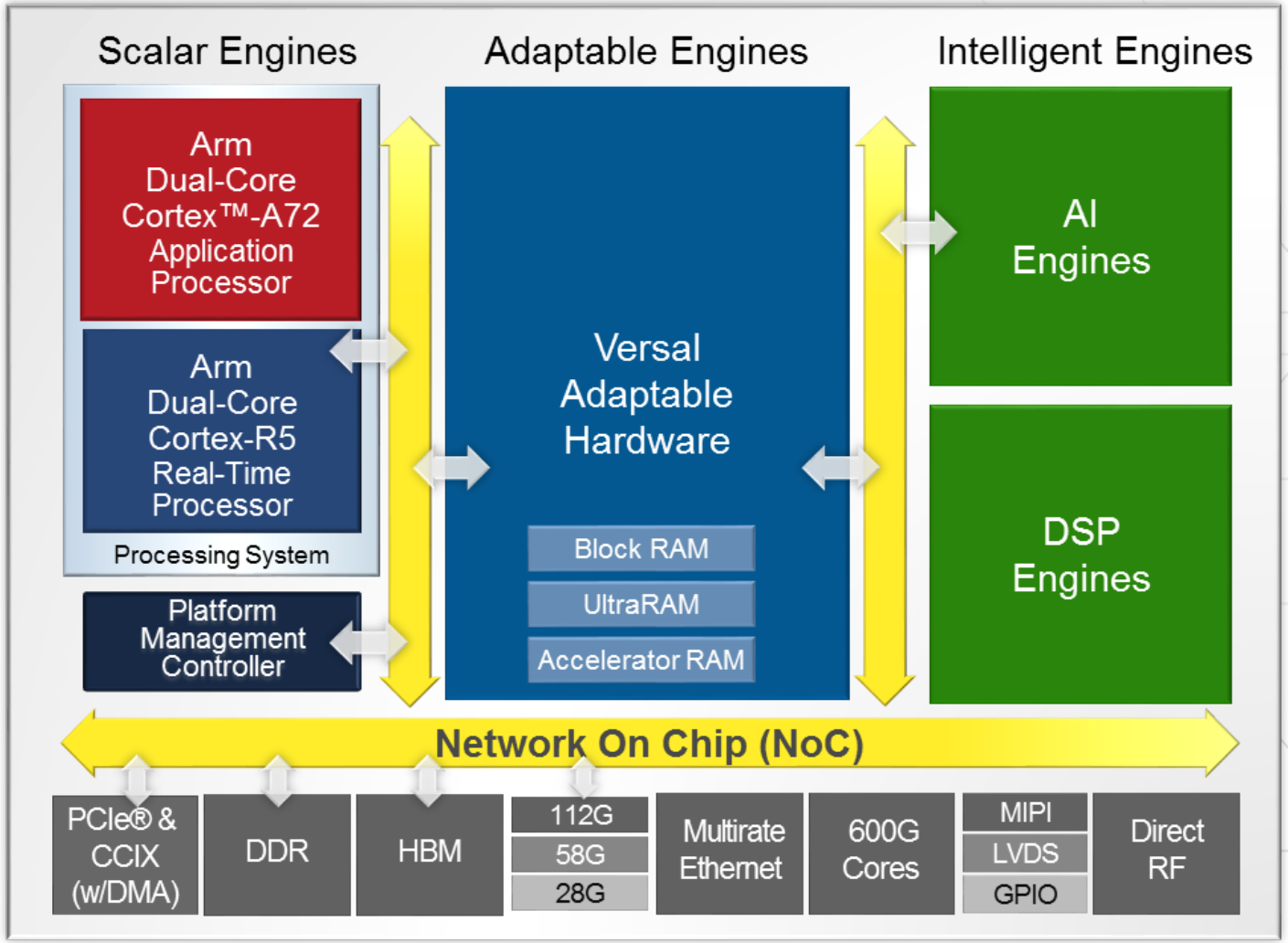
关注赛灵思
获取更多研讨会资讯

XILINX

边缘 AI 后续步骤.....

通过 NoC 连接的自适应架构

- **标量引擎**
  - Arm® Cortex™-A72 APU
  - Arm Cortex-R5 RPU
- **自适应引擎**
  - CLB
  - 内存
- **智能引擎**
  - AI 引擎
  - DSP 引擎
- **连接**
  - 带有 CCIX 的 PCIe
  - 以太网
  - DDR 存储器控制器
  - 收发器
  - I/O
- **平台资源**
  - 片上网络
  - 平台管理控制器



XILINX

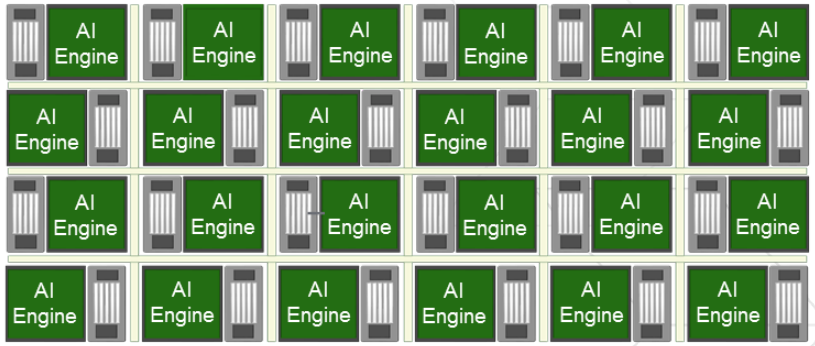
数字信号处理功能

功能	DSP48E2	DSP58
DSP 块/Slice 类型	DSP48E2	DSP58
乘法器与 MACC	27x18	27x24
32 位/16 位单精度浮点乘法-加法	软	√
复数 18b x 复数 18b	不适用	2 x DSP58
3 x Int8 点积	不适用	√

AI 引擎 2D 阵列

VLIW 和 SIMD 架构

C/C++ 可编程

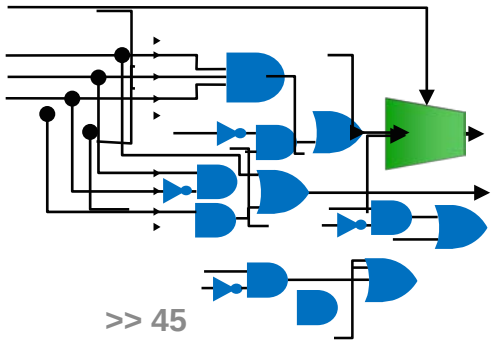


功能

FPGA 架构 DSP

LUT 与存储器

RTL 输入



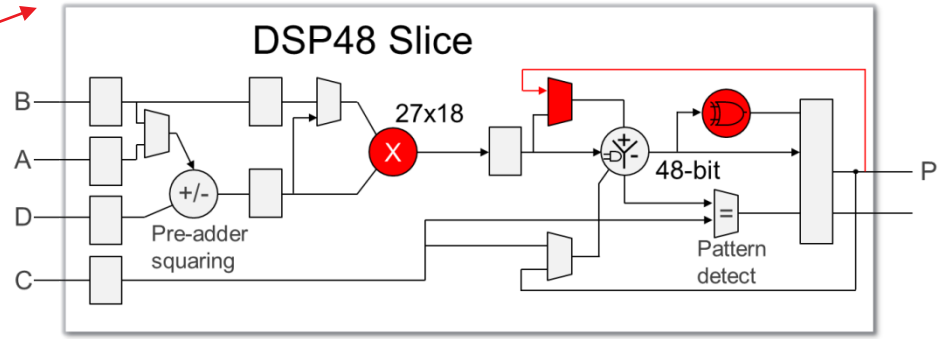
功能

DSP48E2 Slice

硬化乘法器和加法器

ACC = ACC + (A × B)

RTL 输入

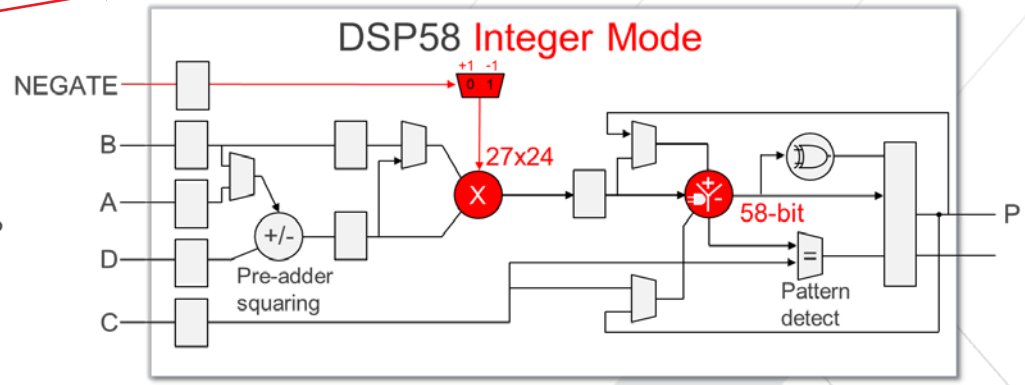


功能

DSP58

补充特性

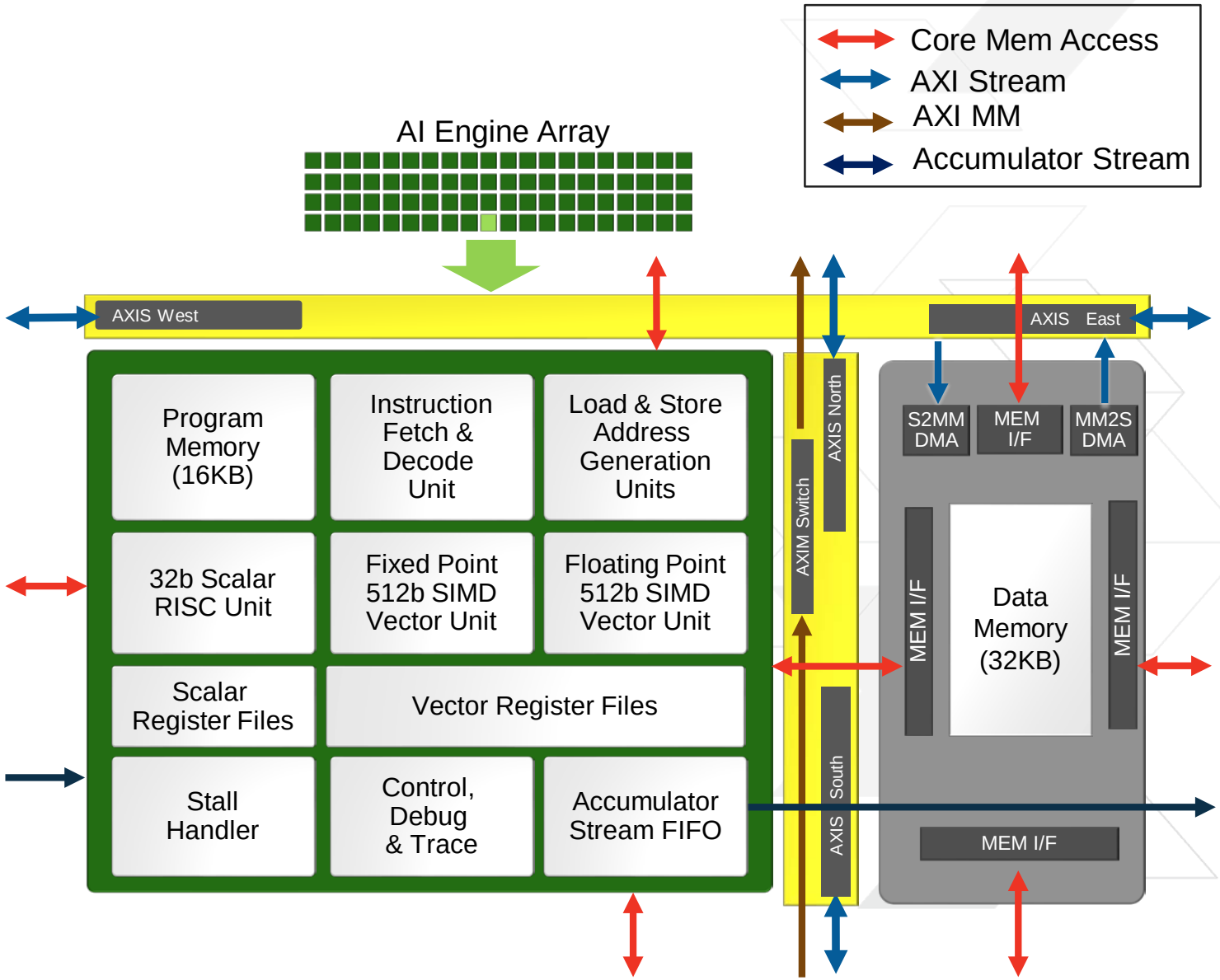
RTL 输入



XILINX

AI 引擎架构

- **AI引擎单元**
  - AI 引擎, 数据存储和互联
- **1+ GHz VLIW/SIMD AI 引擎**
  - 带有支持浮点和定点的矢量单元的32位标量RISC处理器
- **每个AI引擎可以访问4个内存模组(N,E,S,W)做为一个连续的内存**
- **通过AXI-MM接口完成配置，控制以及调试功能**
- **通过AXI-Stream crossbar交换完成对于N/E/S/W流的路由**



XILINX.

总结

- **VERSAL™ 包含一些革命性的新功能**
  - 高带宽NoC
  - 配备专用 AI 引擎，具备突破性的加速与计算功能
  - 为器件集成与互联提供集成“Shell”

- **如需了解更多信息，请访问 www.xilinx.com/versal**

- **问题解答请咨询：**
  - yuxiangw@Xilinx.com
  - Giulio.Corradi@Xilinx.com
  - KVT@Xilinx.com

关注赛灵思
获取更多研讨会资讯

灵活应变
万物智能

XILINX