赛灵思工业物联网研讨会
XILINX IIoT SEMINAR

Electric Drives & Motor Control with Predictive Maintenance

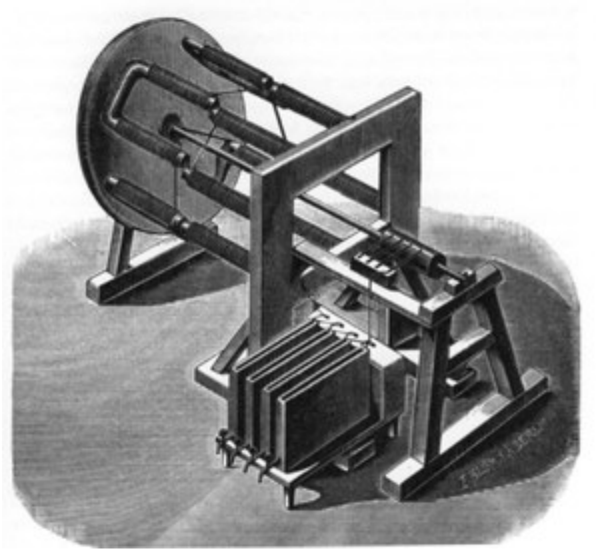
KV Thanjavur Bhaaskar
Industrial IoT Solution Architect & Product Marketing
KVT@Xilinx.com
May 2019

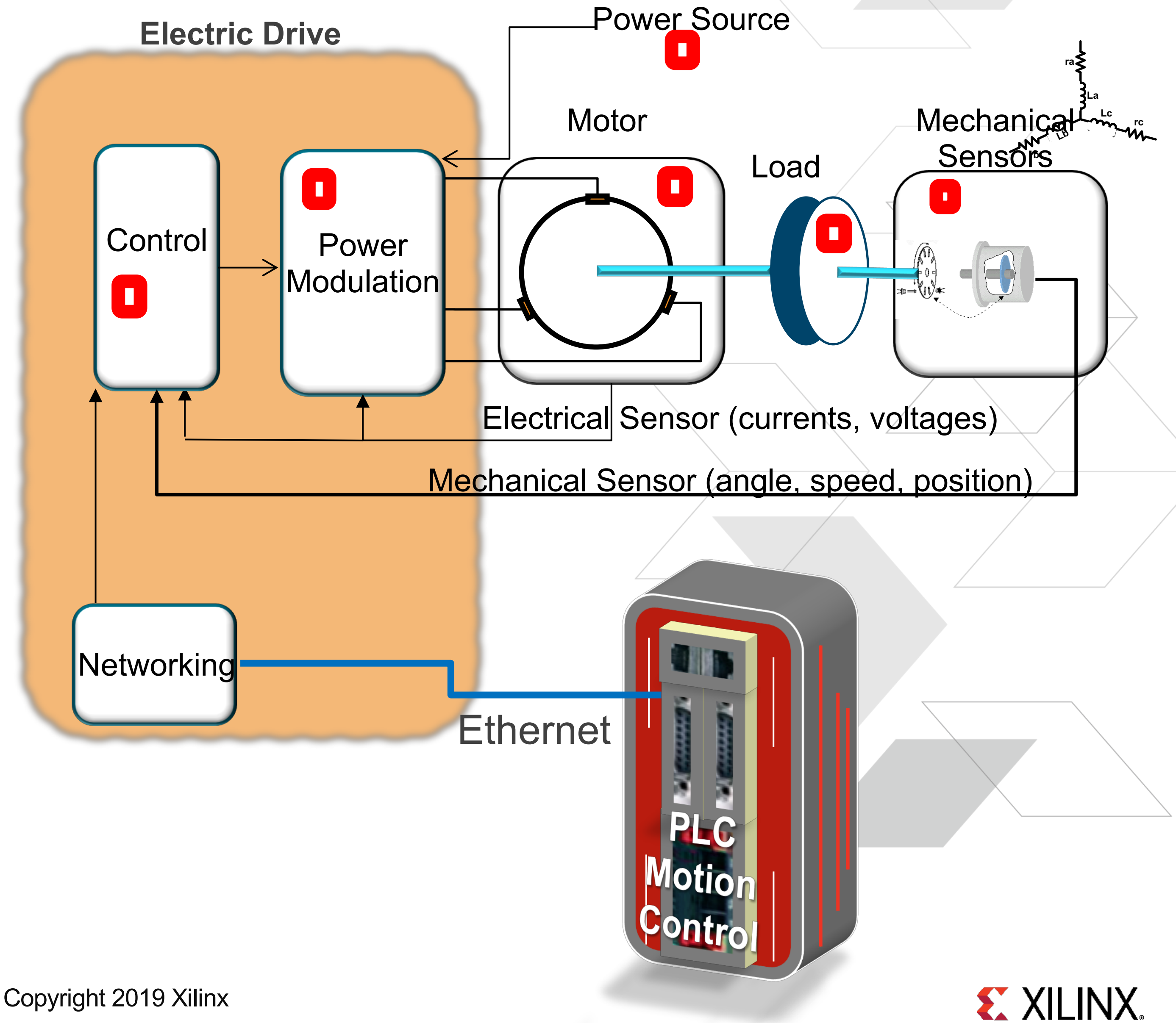
XILINX

Agenda

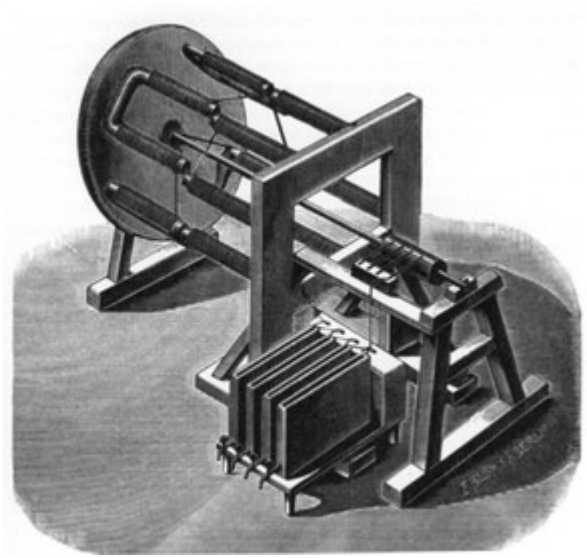
> Electric Drives & Motors in Industrie 4.0
> Electric Drives Demonstration Platform
> SPYN
> Predictive Maintenance
> SPYN AI
> Resources

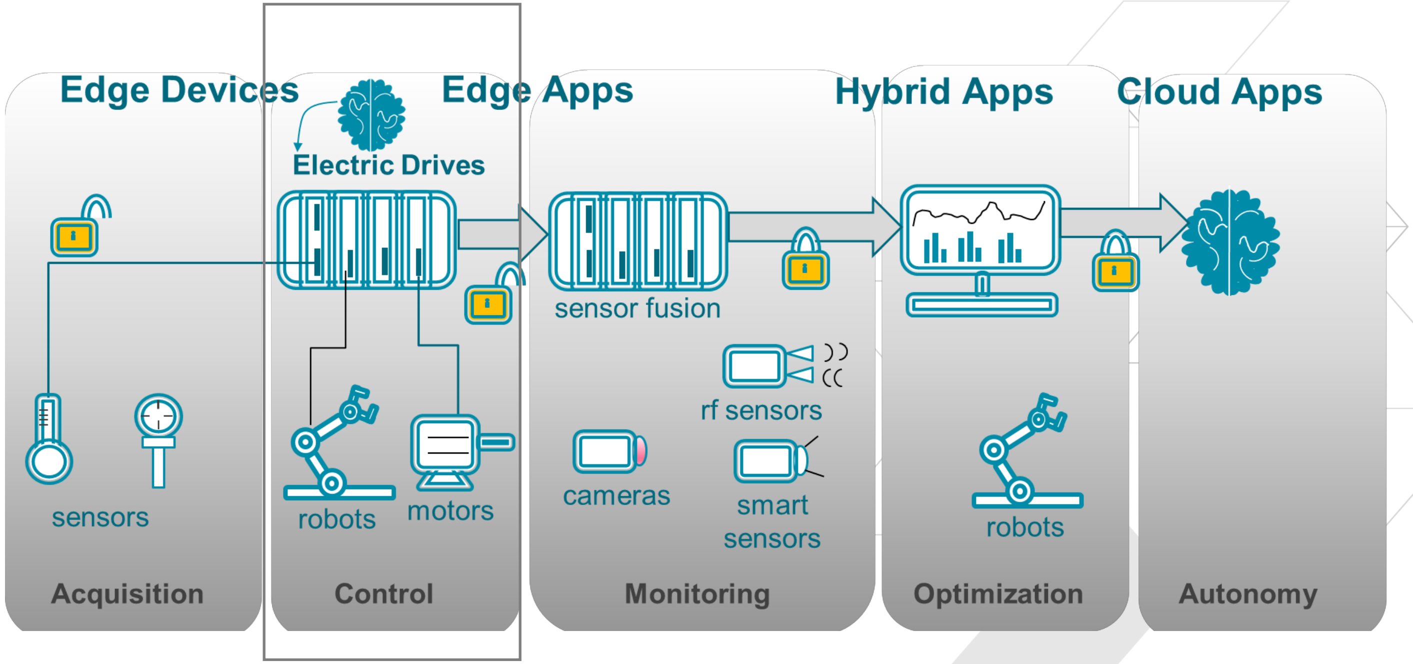
Six Key Elements of Traditional Electric Drives





Industrie 4.0 and IIoT: Electric Drives are Edge Devices





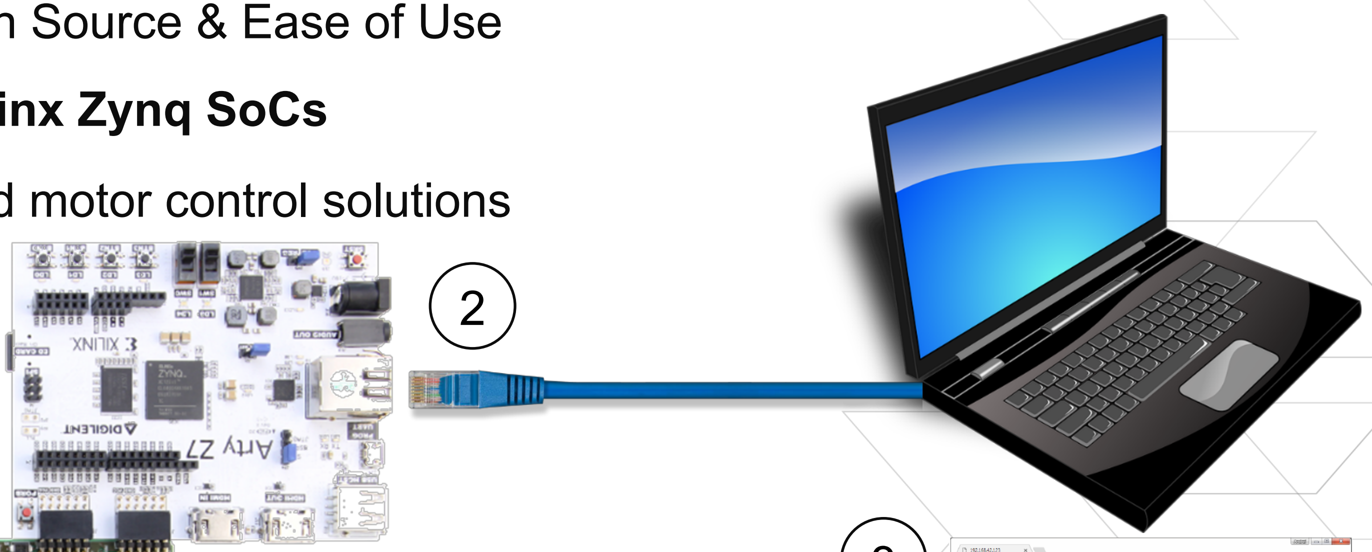
Today's Distributed Industrial Architectures
Industrie 4.0 / Industrial IoT

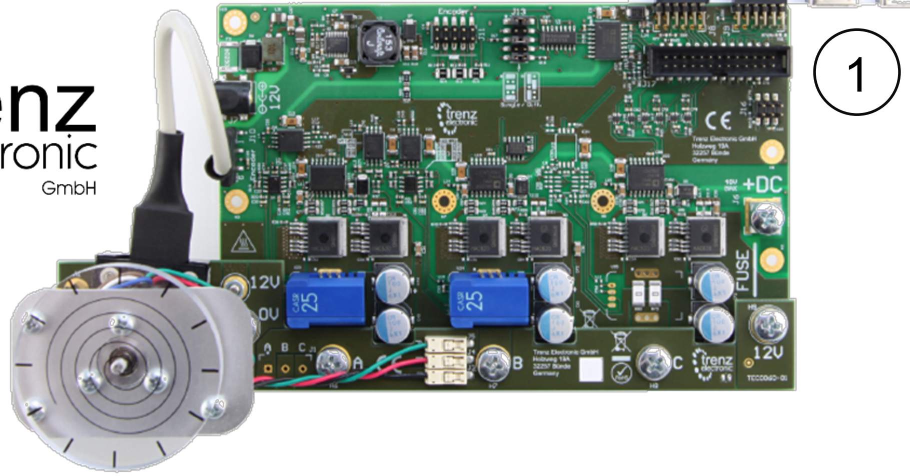
Electric Drives are expected to do more in Era of Industrie 4.0 / Industrial IoT:
IT-OT Convergence

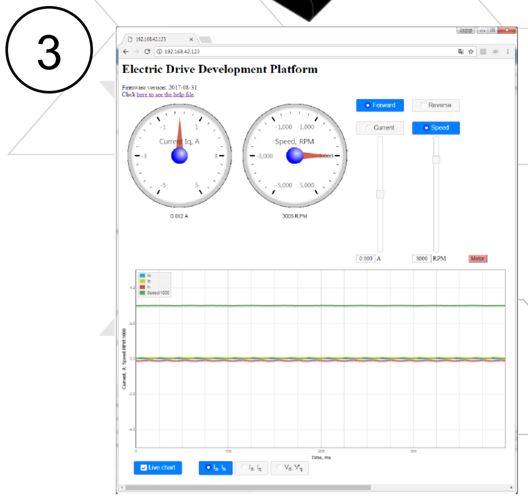
EDDP: Electric Drives Demonstration Platform

- Design Methodology Predicated on Open Source & Ease of Use
- EDDP takes complete advantages of **Xilinx Zynq SoCs**
- Platform offers two different flows to build motor control solutions
  - Xilinx SDSoC Design Flow
  - Xilinx Vivado HLS Design Flow

EDDP KIT







Three simple steps to get started!
[EDDP Demo Video]

Algorithm translates in C code and compiled in RTL

Algorithm in C/C++

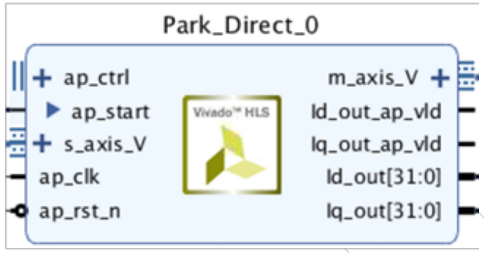
```c
// See the header file for the documentation.
void Park_Direct(hls::stream<int64_t> &s_axis, hls::stream<int64_t> &m_axis, int32_t *Id_out, int32_t *Iq_out){

#pragma HLS interface axis port=m_axis
#pragma HLS interface axis port=s_axis
    int64_t in_data, res;
    int16_t Ialpha, Ibeta, Theta, RPM;
    int32_t Id, Iq;
    int32_t cos_theta, sin_theta;
    int32_t Ia_cos, Ib_sin, Ib_cos, Ia_sin;

    // Decode Input stream
    in_data = s_axis.read();                    // Read one value from AXI4-Stream
    Ialpha = int16_t(in_data & 0xFFFF);             // Extract Ialpha - bits[15..0] from input stream
    Ibeta = int16_t((in_data >> 16) & 0xFFFF);      // Extract Ibeta - bits[32..16] from input stream
    RPM = int16_t((in_data >> 32) & 0xFFFF);        // Extract RPM - bits[47..32] from input stream
    Theta = int16_t((in_data >> 48) & 0xFFFF);      // Extract Angle - bits[63..48] from input stream

    // Process data
    cos_theta = (int32_t)cos_table[Theta];
    sin_theta = (int32_t)sin_table[Theta];
    Ia_cos = (int32_t)Ialpha * cos_theta;
    Ib_sin = (int32_t)Ibeta * sin_theta;
    Ib_cos = (int32_t)Ibeta * cos_theta;
    Ia_sin = (int32_t)Ialpha * sin_theta;
    Id = (Ia_cos + Ib_sin) >> 15;
    Iq = (Ib_cos - Ia_sin) >> 15;
    Id = (Id > MAX_LIM) ? MAX_LIM : Id;             // Clip max
    Id = (Id < MIN_LIM) ? MIN_LIM : Id;             // Clip min
    Iq = (Iq > MAX_LIM) ? MAX_LIM : Iq;             // Clip max
    Iq = (Iq < MIN_LIM) ? MIN_LIM : Iq;             // Clip min

    *Id_out = Id;
    *Iq_out = Iq;
    // Write output stream
    res =   (((int64_t)Theta << 48) & 0xFFFF000000000000) | // Put Angle bits[63:48]
            (((int64_t)RPM << 32)   & 0x0000FFFF00000000) | // Put RPM bits[47:32]
            (((int64_t)Iq << 16)    & 0x00000000FFFF0000) | // Put Iq bits[31:16]
            ( (int64_t)Id           & 0x000000000000FFFF);  // Put Id bits[15:0]
    m_axis.write(res);                              // Write result to the output stream
}
```

Xilinx Tools

SDSoC

Vivado™ HLS

RTL Design



Tools: SDSoC and Vivado HLS enabled motor control algorithm written in C/C++ code to be made into hardware implementation

PYNQ harnesses the power of Programmable Logic

Two most common use models for PYNQ—SPYN Showcases Both

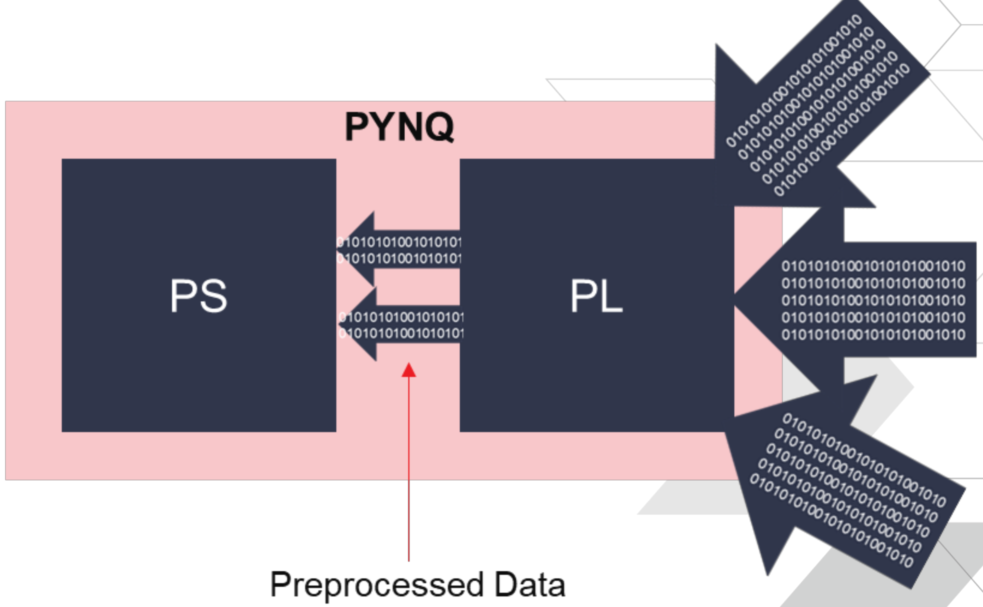
> **Control & Query Programmable Logic IP**

Steering wheel, pedals, gauges: **PYNQ Framework**

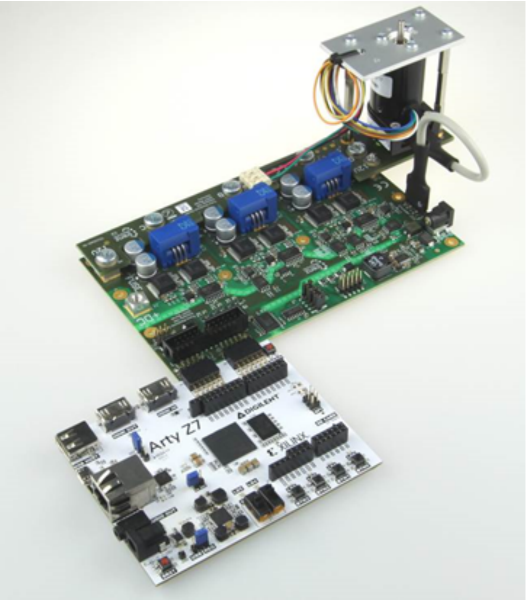
High Performance Engine: **Programmable Logic**

> **Division of labor between PS & PL**

- PL to rapidly preprocess raw, streaming IO data from sensors and passes to PS
- Processor leverages extensive libraries to execute wide range of operations on data



SPYN: Bridging two worlds for an Intelligent Drive



\+

PYNQ™ python powered

=

EDDP
Electric Drives Demo Platform

PYNQ
Python Productivity for Zynq

SPYN
Extreme Edge Analytics for Motor Control

- **SPYN takes advantage of both EDDP Kit & PYNQ framework**
- **EDDP kit can also be used to test, modify & build SPYN project at no additional charge**
- **The solution enables python powered machine learning & edge analytics for motor control**
- **Python libraries are leveraged to provide UI for control, data manipulation, analytics & visualization**

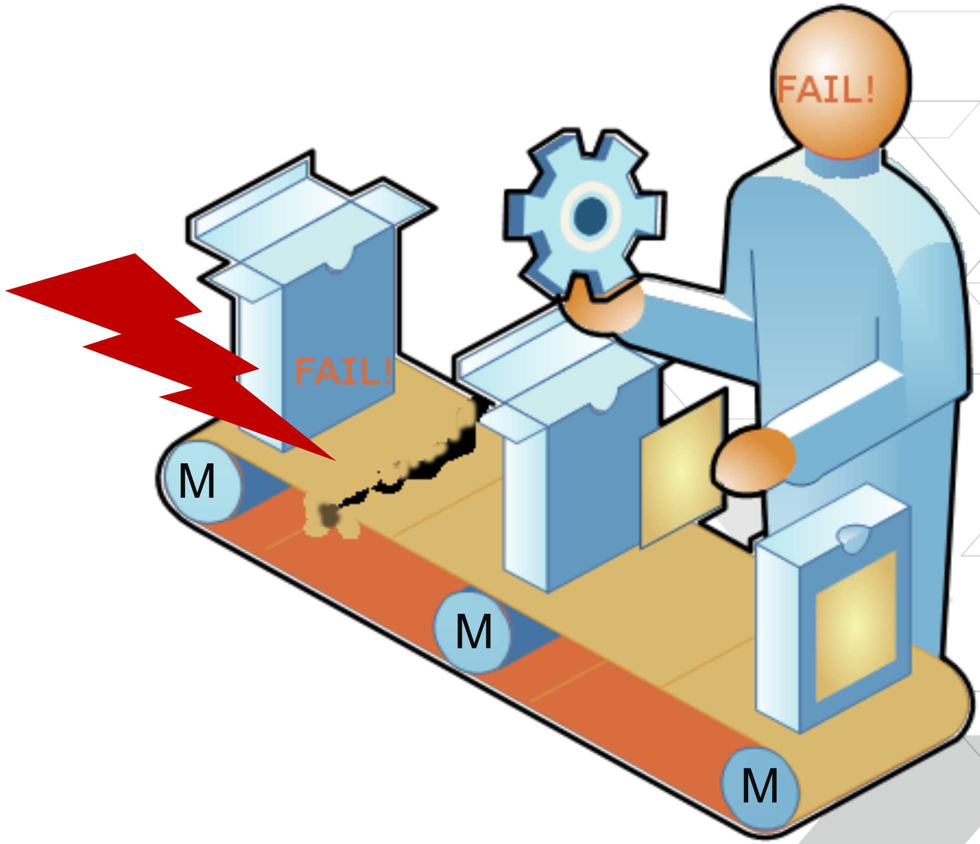
Predictive Maintenance

Run to Failure Maintenance – what is it?

- When a machine breaks down, fix it
- If it ain't broke, don’t fix it

- Reactive management technique “fire fighting”
- The most expensive maintenance management
  - Unscheduled shutdown
  - Emergency maintenance team calls

VERY BAD



Scheduled Maintenance - what is it?

COSTLY
UNNECESSARY REPAIRS

> Time based maintenance using **MTTF** (Mean Time To Failure) data

> Identified critical assets likely to fails and estimate MTTF

> Use asset inspection (Proof Test) at scheduled time

> MTTF of a product changes with its use:

>> Pumping water

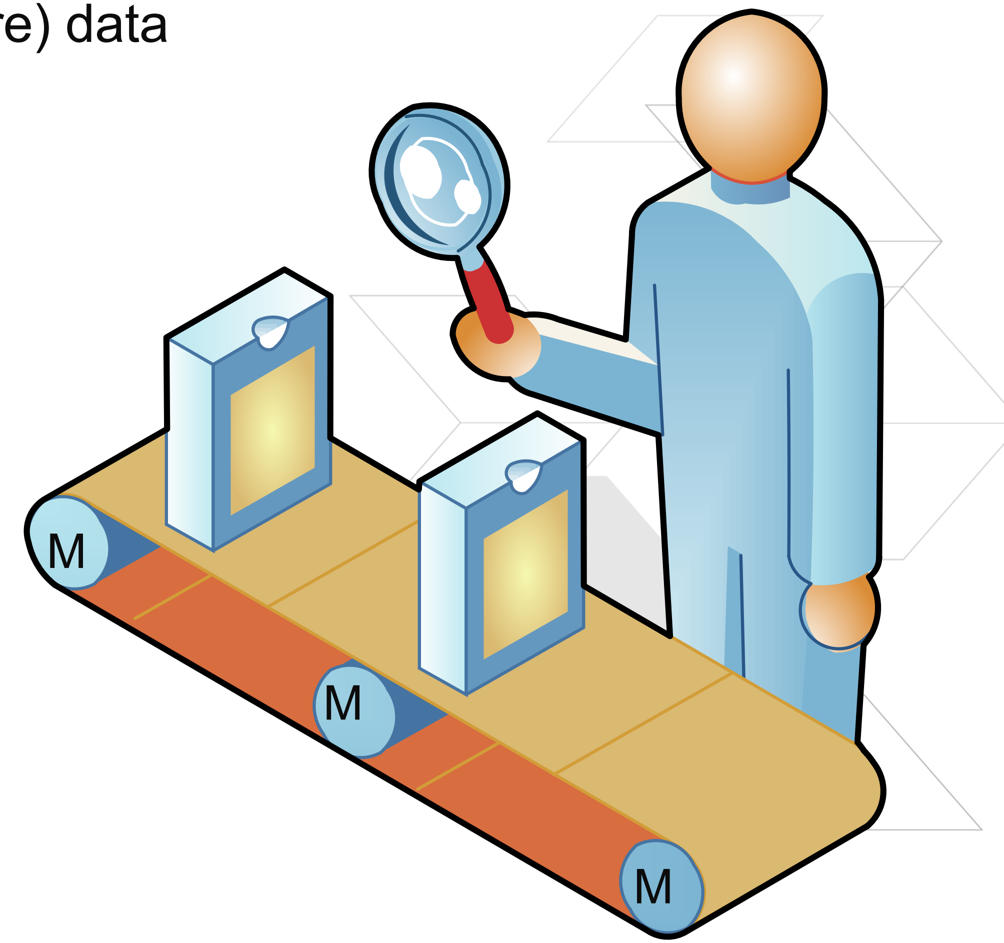
>> Pumping salt water

>> Pumping dirty water

Different MTBF

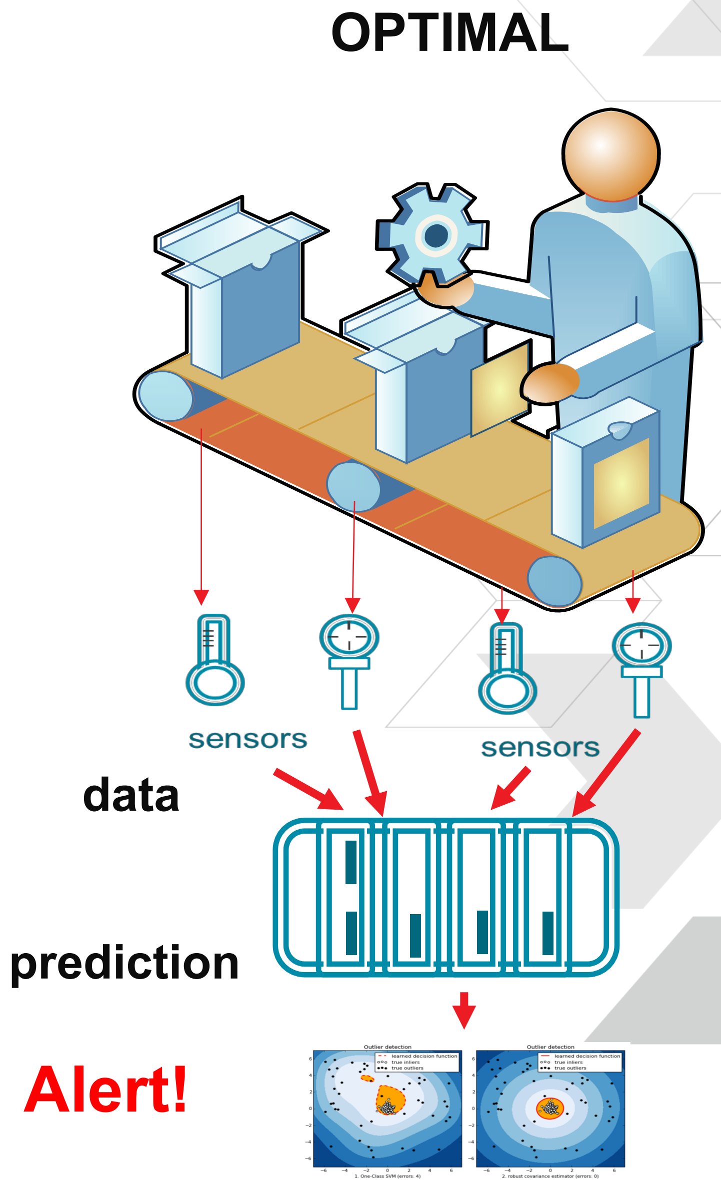
> Can produce costly shut-down if done too early

> Can fail catastrophically using generic MTBF



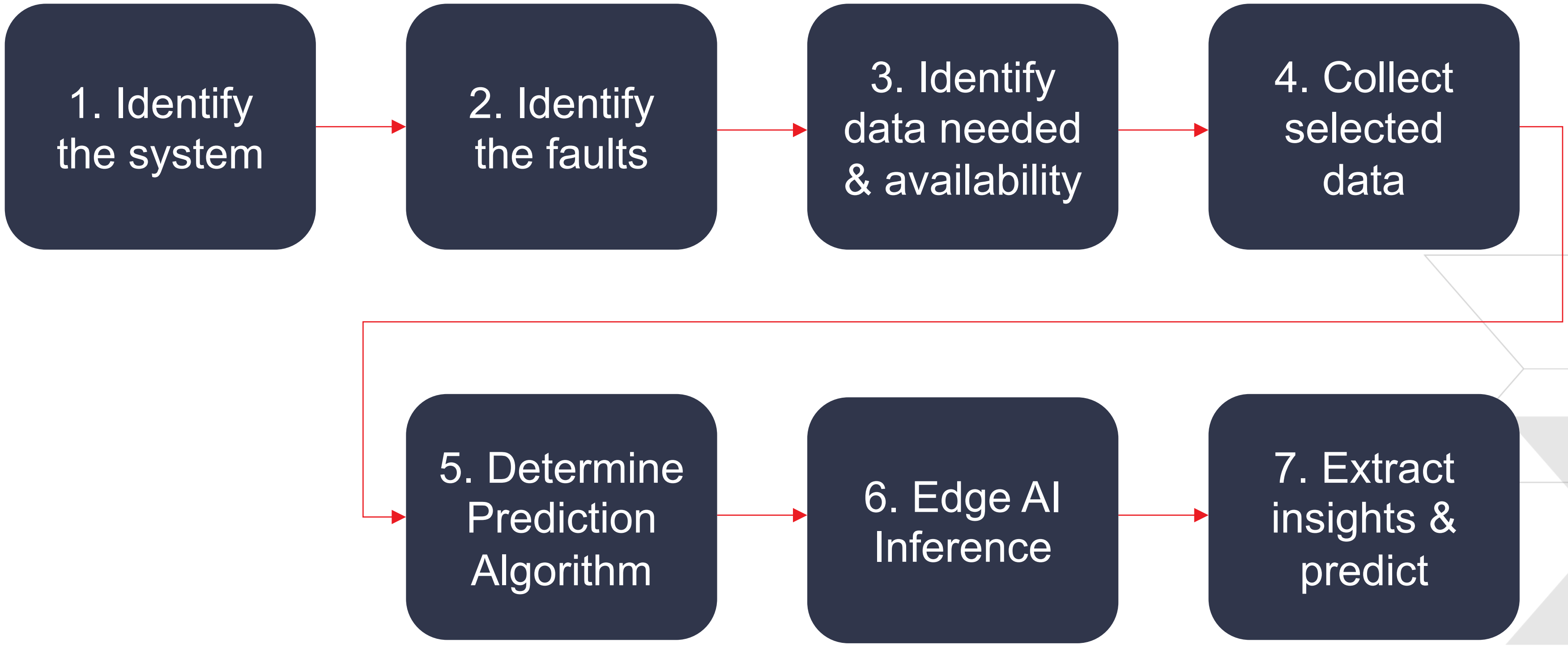
Predictive Maintenance – what is it?

- Method for timely maintenance execution
- Additional Benefits
  - Optimum Availability
  - Optimum Operating Conditions
  - Optimum Utilization of Maintenance Resources
  - Minimum spares of Inventory
- Uses sensors to continuously collect asset data
  - Sensors already available in the system
  - Sensors deployed for the maintenance
- Estimate maintenance with prediction algorithms
  - Model Based
  - Rule Based
  - Machine Learning Based

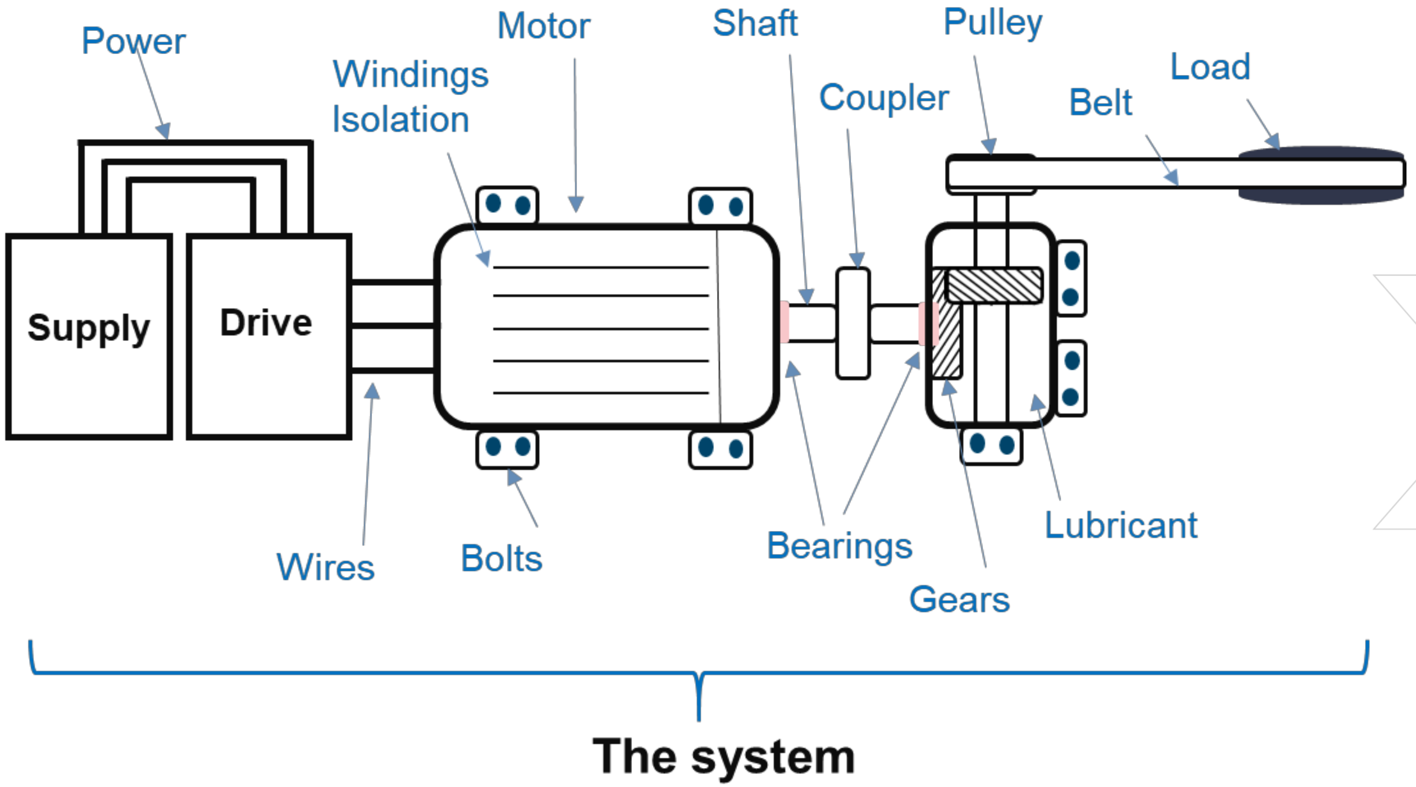


XILINX

7 Steps to Prediction



Step 1. Identify the system
Use Case – Power Train

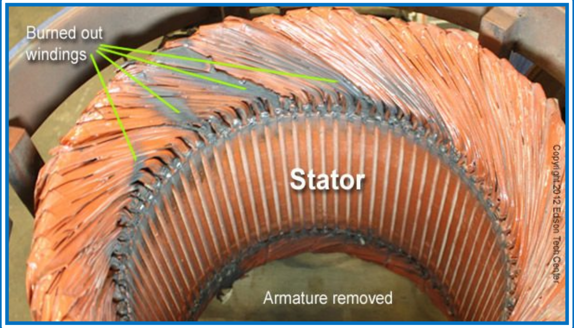


Step 2. - Sources of Faults

> **Electrical faults**
>> Open or short circuit in motor windings
>> Isolation degradation
>> High resistance contact to conductor
>> Wrong or unstable ground

> **Mechanical faults**
>> Broken rotor bars or magnet
>> Cracked end-rings
>> Bent shaft
>> Bolt loosening
>> Bearing failure
>> Gearbox failure

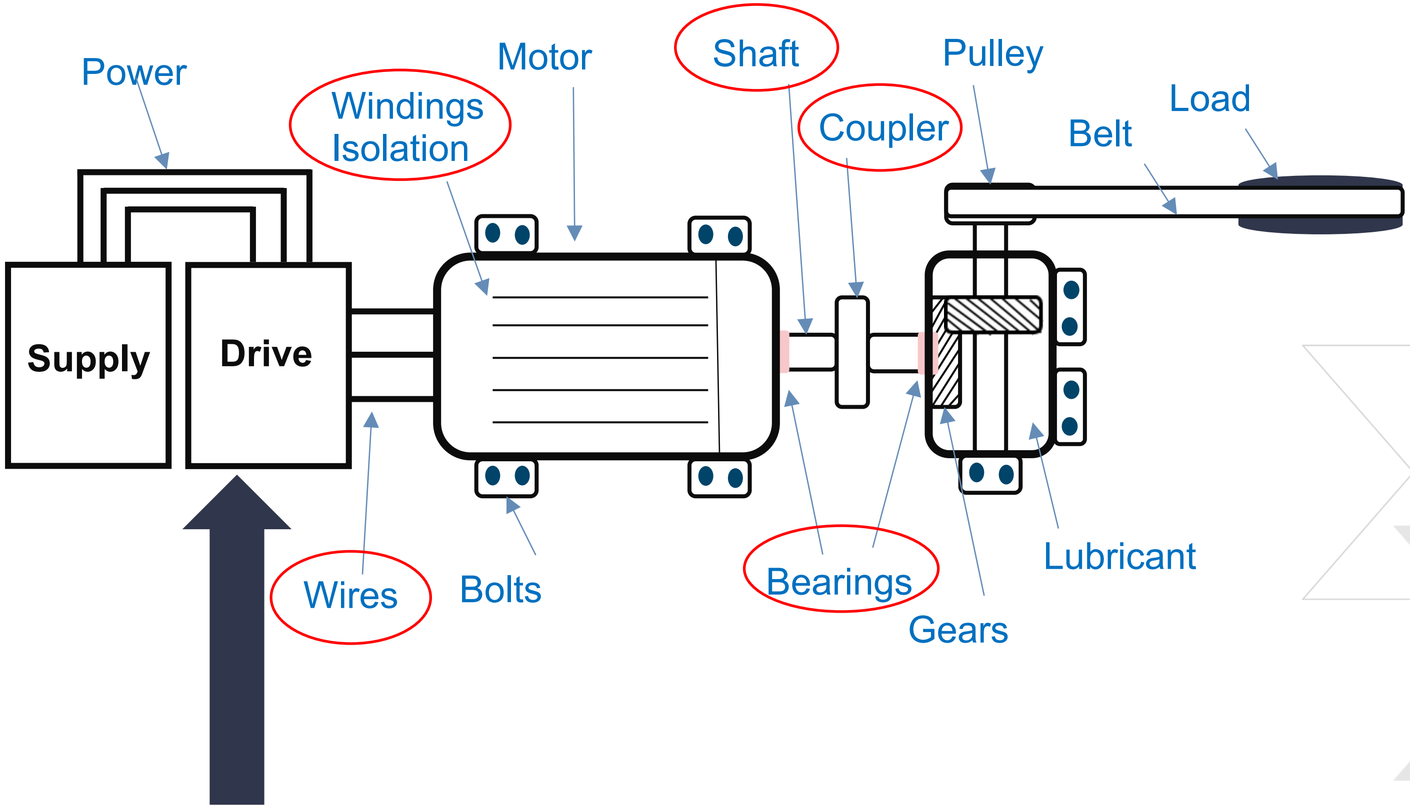
> **Outer motor drive system failures**
>> Inverter system failure
>> Unstable voltage/current source
>> Shorted or opened supply line



MANY FAILURE MODES

MANY DATA SETS FOR FAILURES AND NORMAL BEHAVIOUR

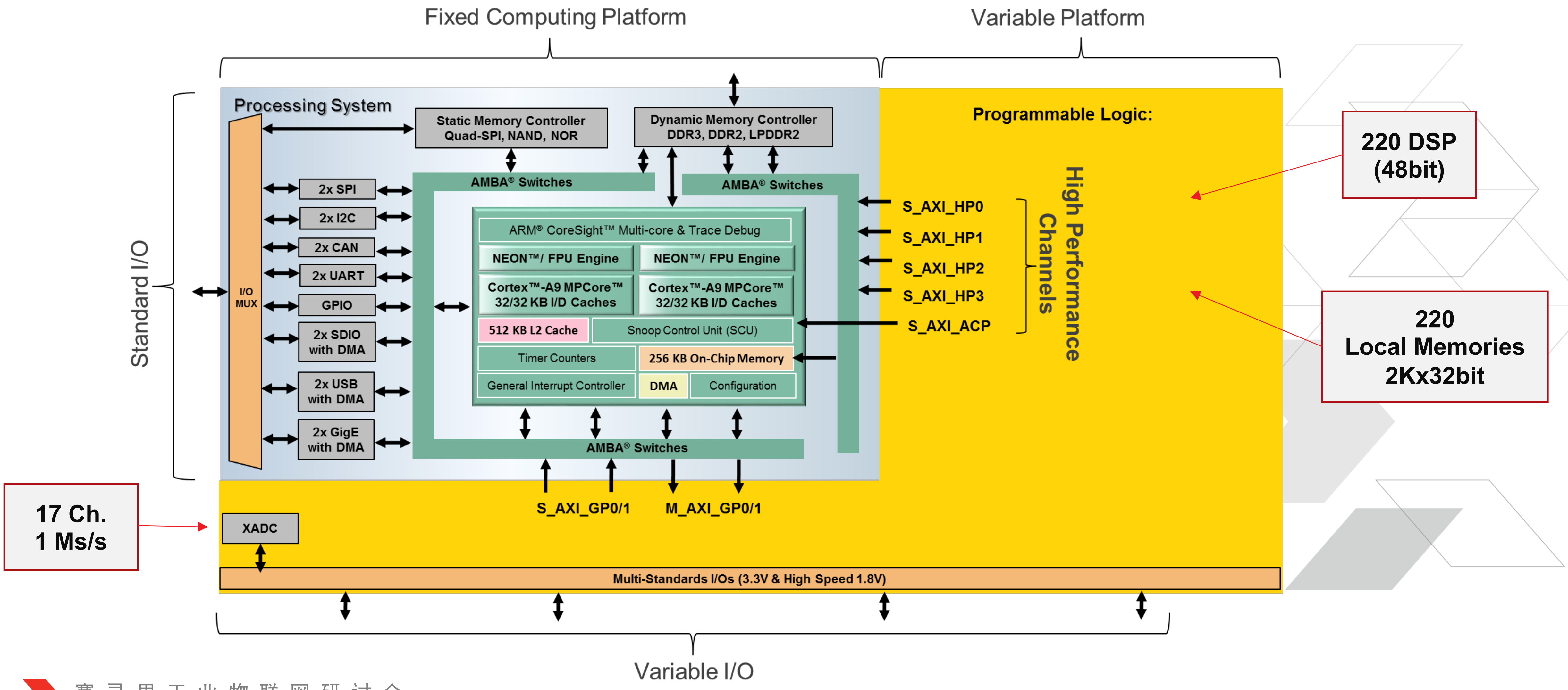
STEP 3. Identify data needed and data availability



Using the Drive's variables for prediction

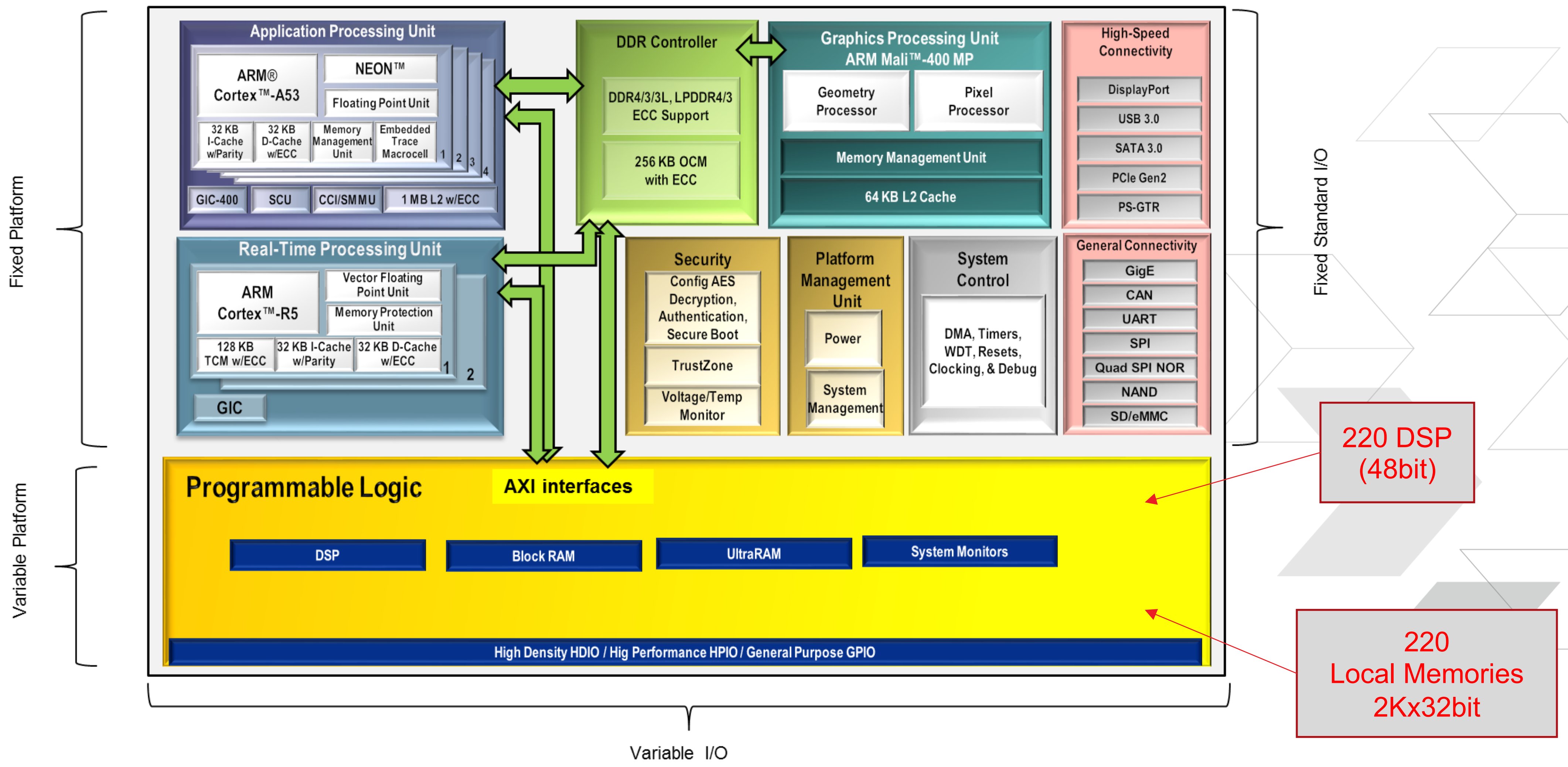
Step 4. - How do we collect the data?

Platform 1: Zynq-7000 All Programmable SoC



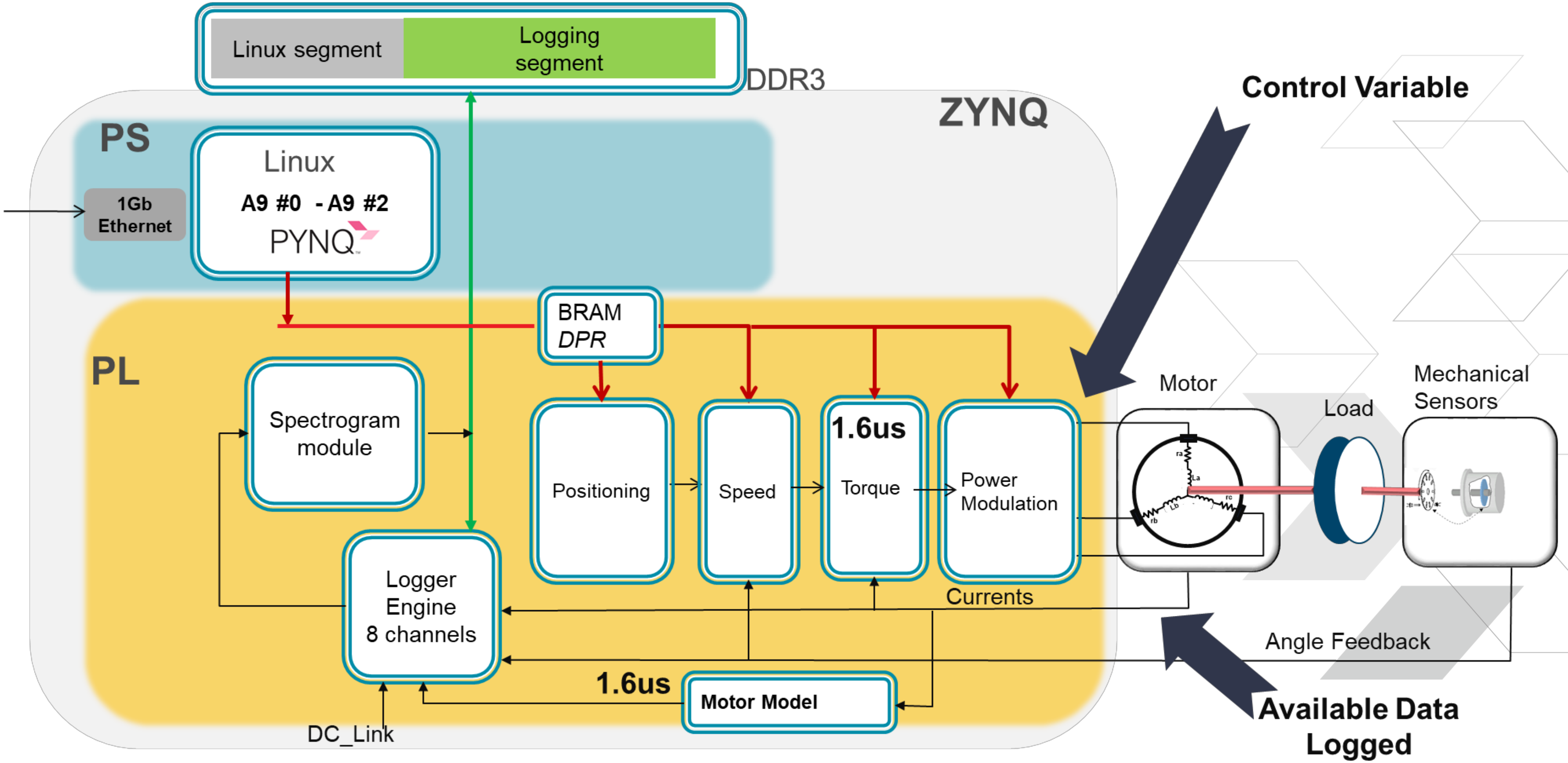
Step 4. - How do we collect the data?

Platform 2: Zynq UltraScale +



Step 4. - How do we collect the data?

Drivers control & available data is logged!



Step 5. Determine Prediction Algorithm & Approach

> ***SUPERVISED***

- *We posses knowledge of the features*
  - *There is expected outcome*
  - *Data is labeled*
  - *Time of occurrence*
- *We posses knowledge of the system*
  - *A model is available*
  - *A model can be inferred*
- *We can use….*
  - *DNN / CNN*
  - *Decision Trees*
  - *Classifiers*

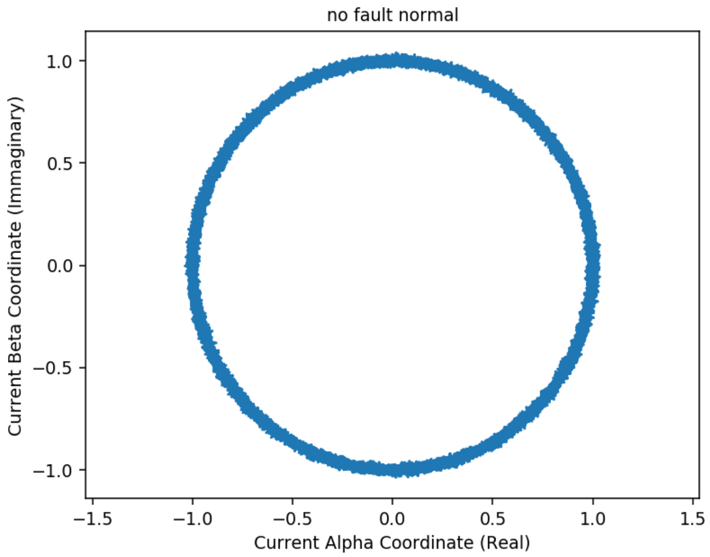
> ***UN-SUPERVISED***

- *No knowledge of the output*
  - *Determine pattern or grouping*
  - *Data is unlabeled*
  - *Time may be unknown*
- *Self Guided Algorithm….*
  - *K-Learn*
  - *Autoencoders*
  - *Generative adversarial networks*

Step 5. Determine Prediction Algorithm & Approach

In this use case: Supervised Learning

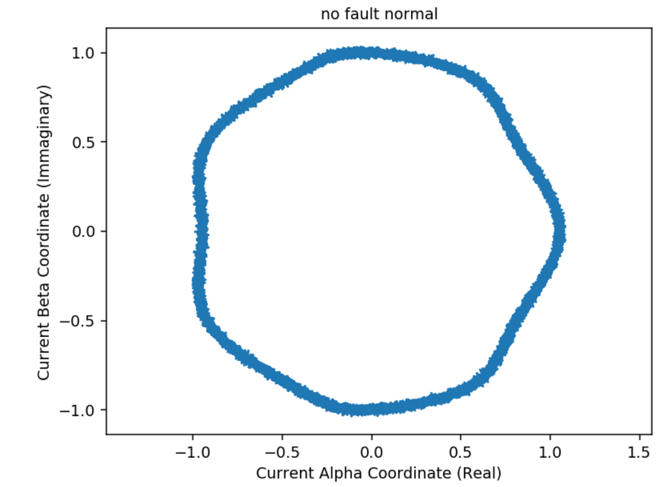
Collect Currents Iα, Iβ → Plot Iα, Iβ in X, Y → Plot Shows Normal or Perturbed → Save the X, Y plot as image → Use CNN for “image” classification



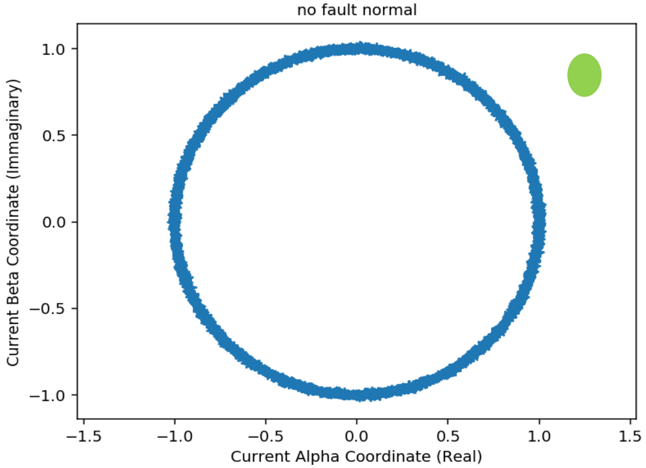
Transformed from “non vision” into “vision” to use CNN

Step 5. Determine Prediction Algorithm & Approach

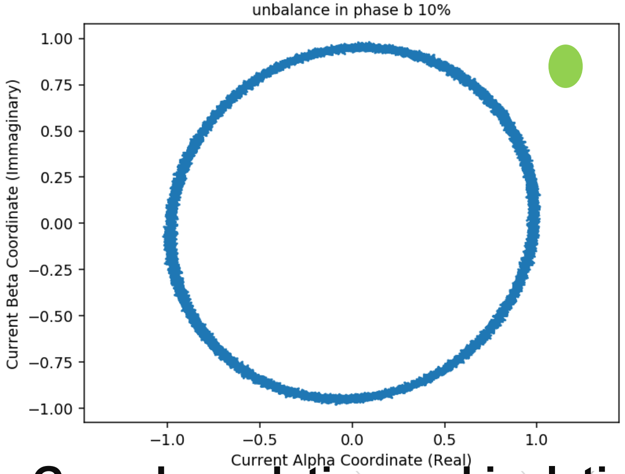
Normal Motor with Saliency



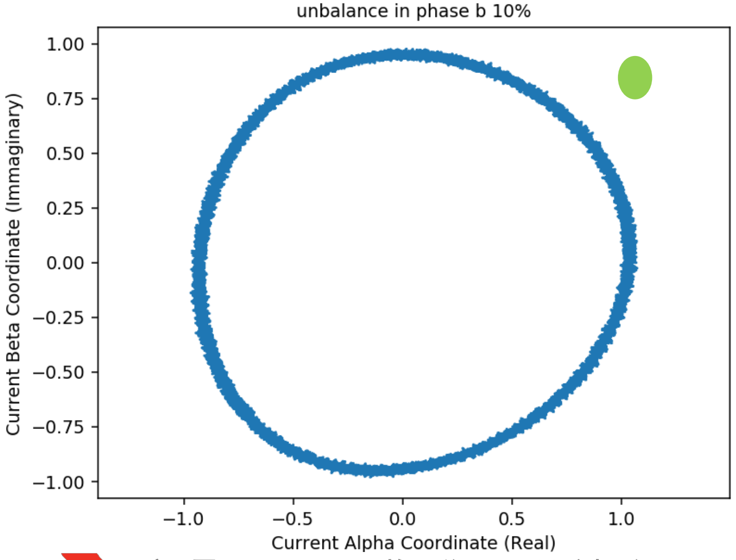
Normal Motor No Saliency



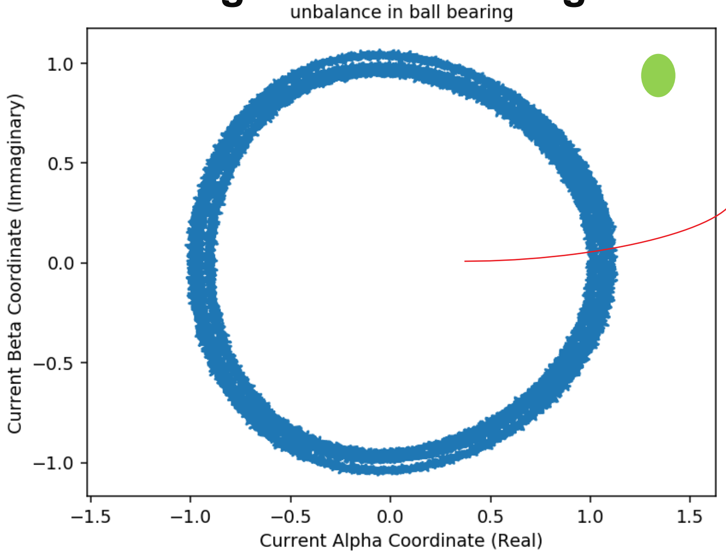
Higher contact resistance Phase b



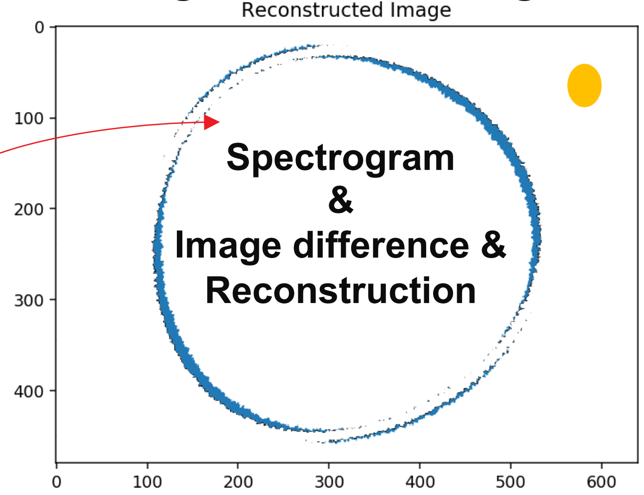
Isolation degradation winding B



Gear degradation and isolation degradation winding B



Gear degradation and isolation degradation winding B



Step 6. Edge AI Inference

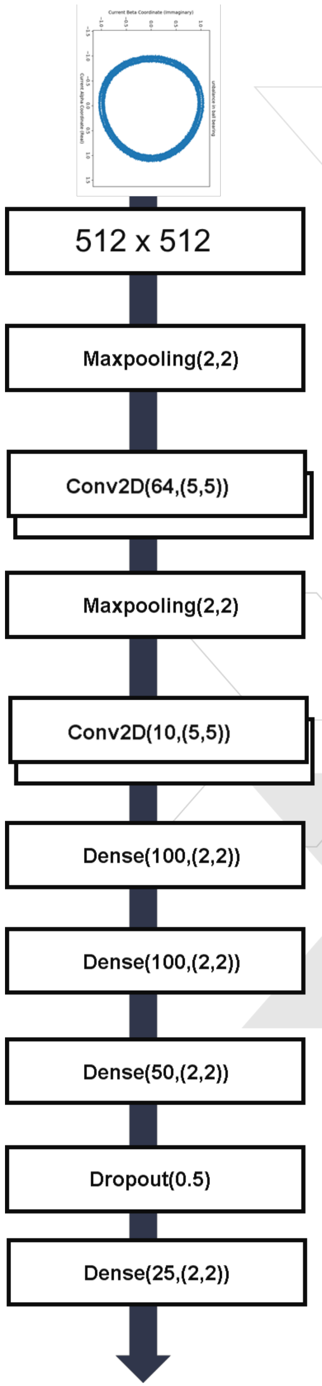
Creating Trained Model

Front End

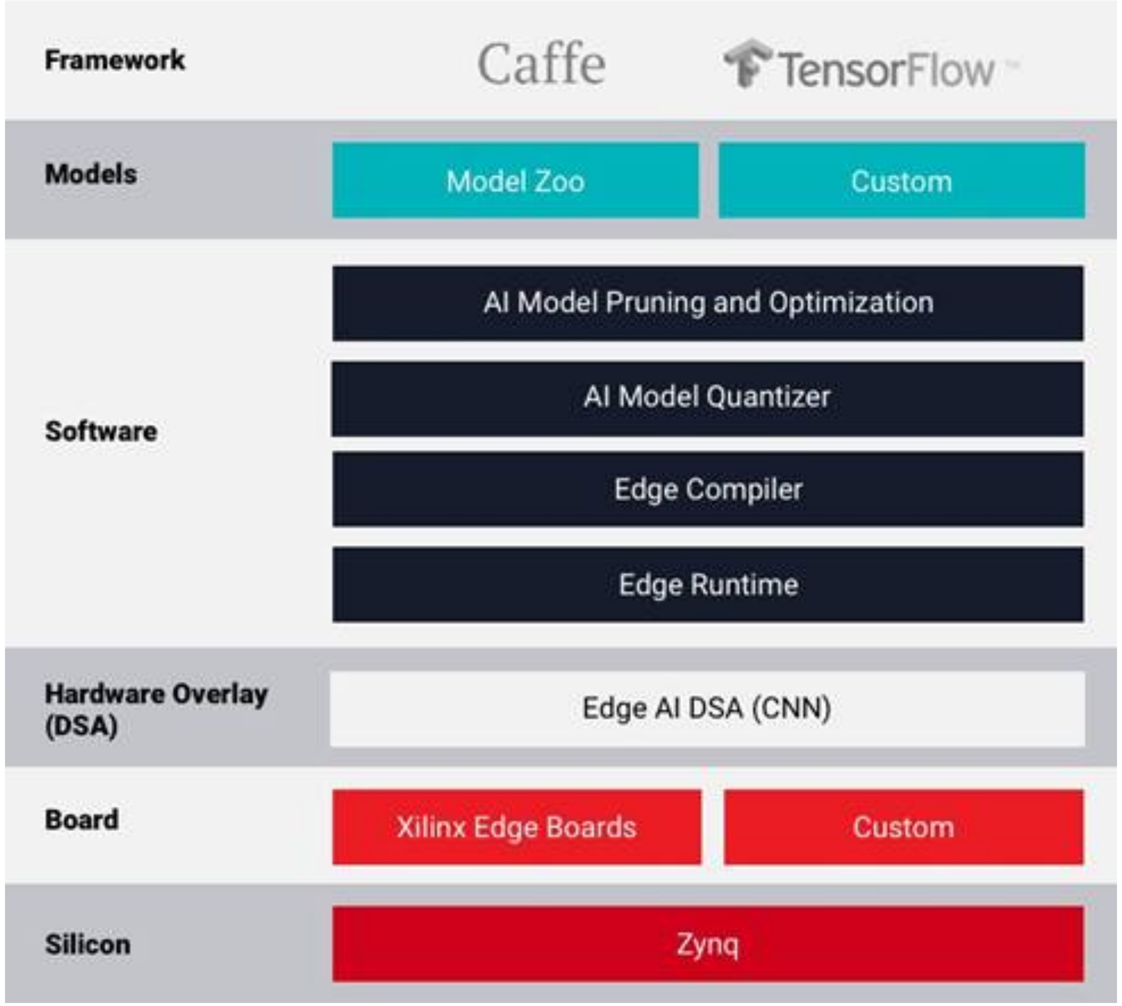
Keras

Backend

TensorFlow

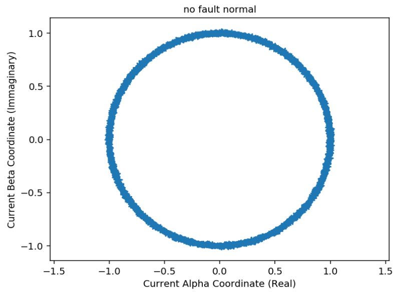


Step 6. Edge AI Inference
Using Xilinx Edge AI Platform



Step 7. Extract Insights & Predict

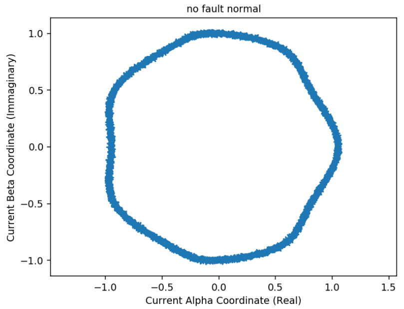
Inputs Image



Output Prediction

Normal Motor

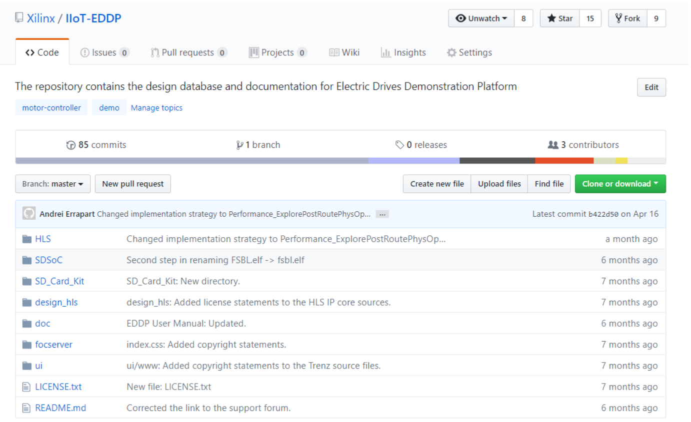
Inputs Image



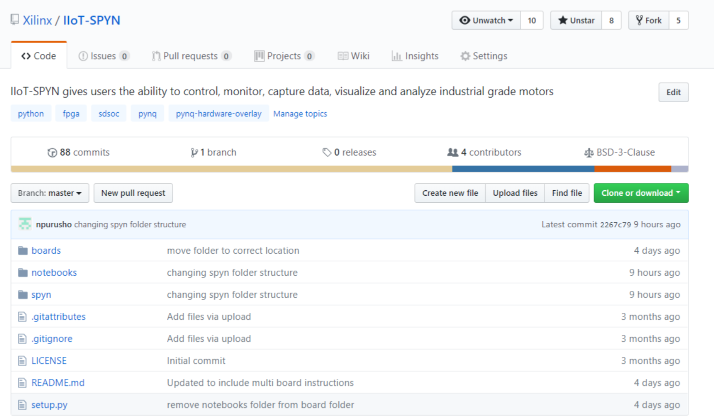
Output Prediction

Motor with Saliency

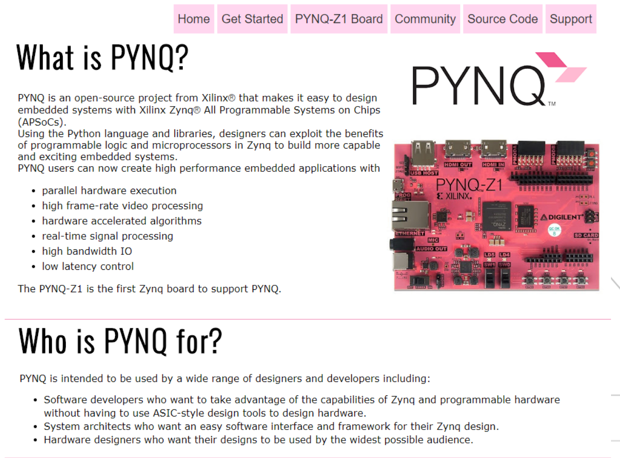
Explore EDDP and SPYN Resources



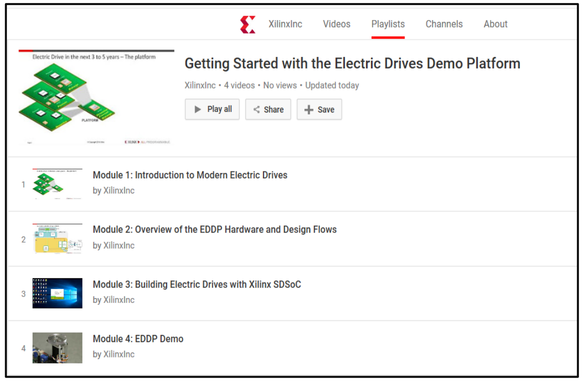
EDDP GitHub
https://github.com/Xilinx/IIoT-EDDP



SPYN GitHub
https://github.com/Xilinx/IIoT-SPYN



PYNQ GitHub
https://github.com/Xilinx/pynq

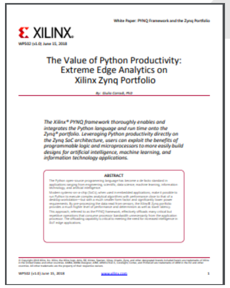


Watch Webinar
ON DEMAND

YouTube Videos:
Getting Started with the Electric Drives Demo
SPYN Quick Take Video on YouTube

Xilinx.com Videos:
Available in English (xilinx.com)
Chinese (china.xilinx.com)
Japanese (japan.xilinx.com)
SPYN Quick Take Video on Xilinx.com

Hardware Kit for
EDDP and SPYN



White
Paper

Explore Xilinx Edge AI

- **DNNDK & DPU**
  - DNNDK basic edition - Download from Xilinx.com
  - Pruning tool, separate upon request
  - DPU available for evaluation & system integration upon request
- **Demos & Ref Designs**
  - General: Resnet50, Googlenet, VGG16, SSD, Yolo v2/v3, Tiny Yolo v2/v3, Mobilenet v1/v2 etc..
  - Video surveillance: face detection & traffic structure
  - ADAS/AD: multi-channel detection & segmentation
  - DPU TRD (Work in progress)
- **Documentation**
  - DNNDK user guide – UG1327
  - DNNDK for SDSoC user guide – UG1331
  - Edge AI tutorials - https://github.com/Xilinx/Edge-AI-Platform-Tutorials
  - DPU product guide & tutorial (Work in progress)
- **Request or Inquiry**
  - Please contact Andy Luo, andy.luo@xilinx.com

Adaptable.
Intelligent.

赛灵思工业物联网研讨会
XILINX IIoT SEMINAR

XILINX