赛 灵 思 工 业 物 联 网 研 讨 会
XILINX IIoT SEMINAR

Xilinx Value Proposition in Industrial Control

翁羽翔 Trevor
赛灵思工业与医疗市场经理
2019年5月

XILINX

Agenda

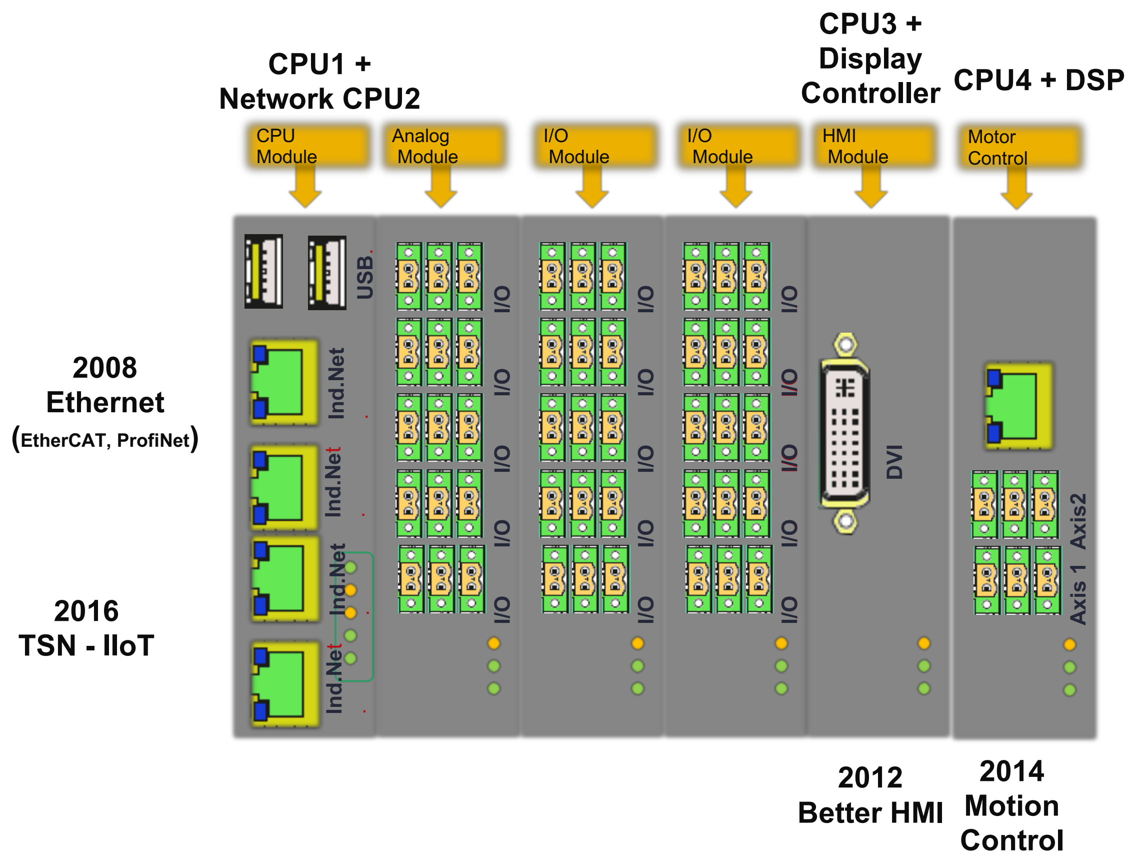
> Trends in Industrial Control

> Platform approach with Zynq Ultrascale+

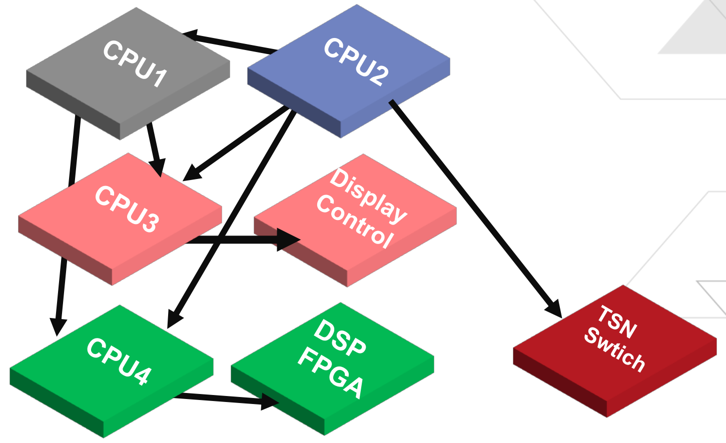
> Mixed Criticality result of integration

> Architectures and values

EVOLUTION OF TYPICAL PLC

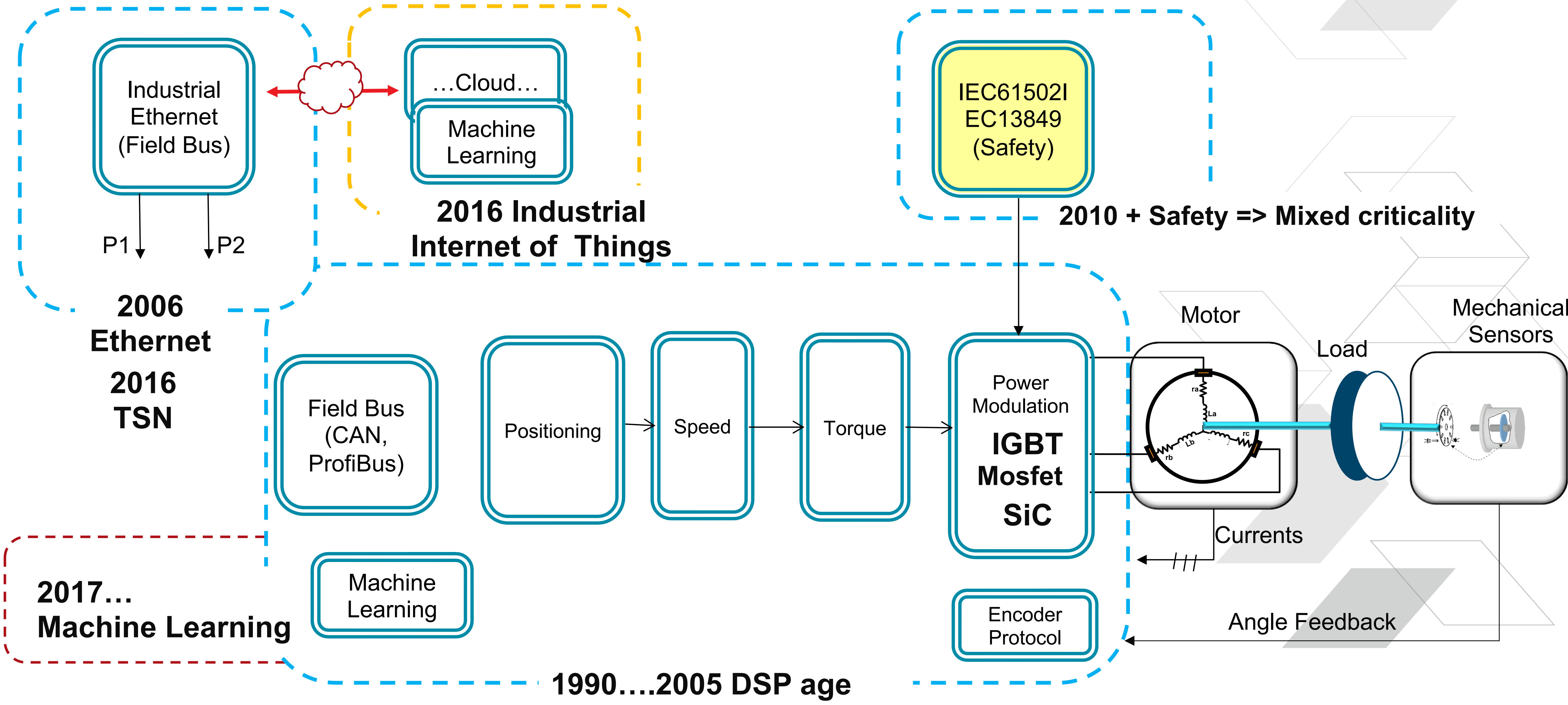


PLC

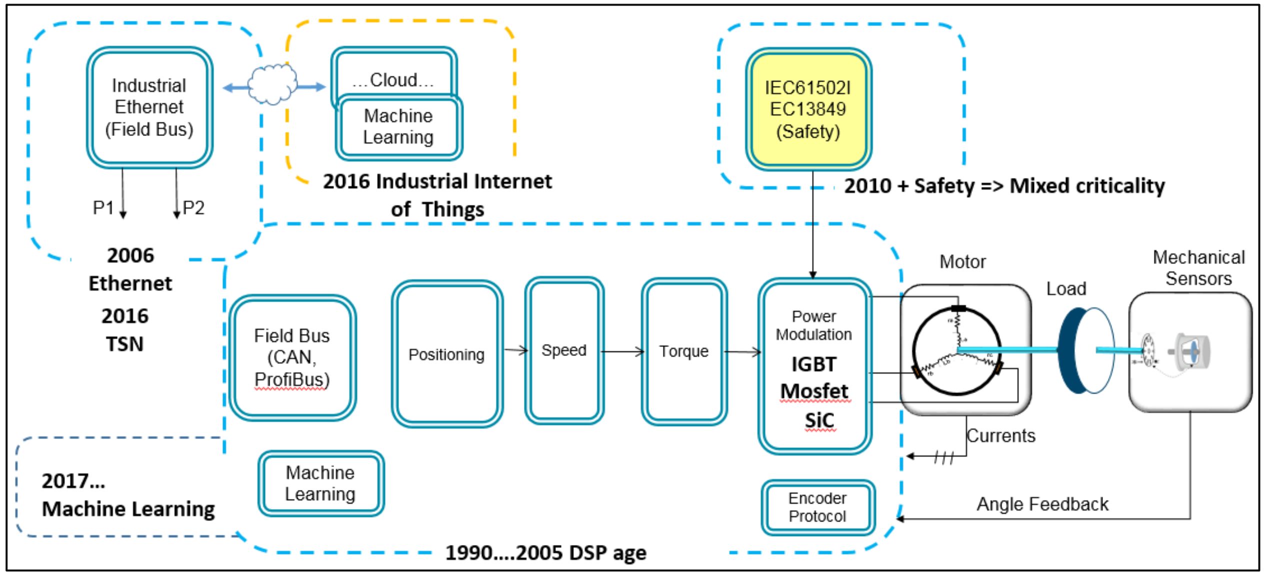


> Often 4 Different CPUs or ASICs
> Different Tools and Development
> Burden in internal data exchange
> Different and incompatible suppliers
> Difficult software maintenance

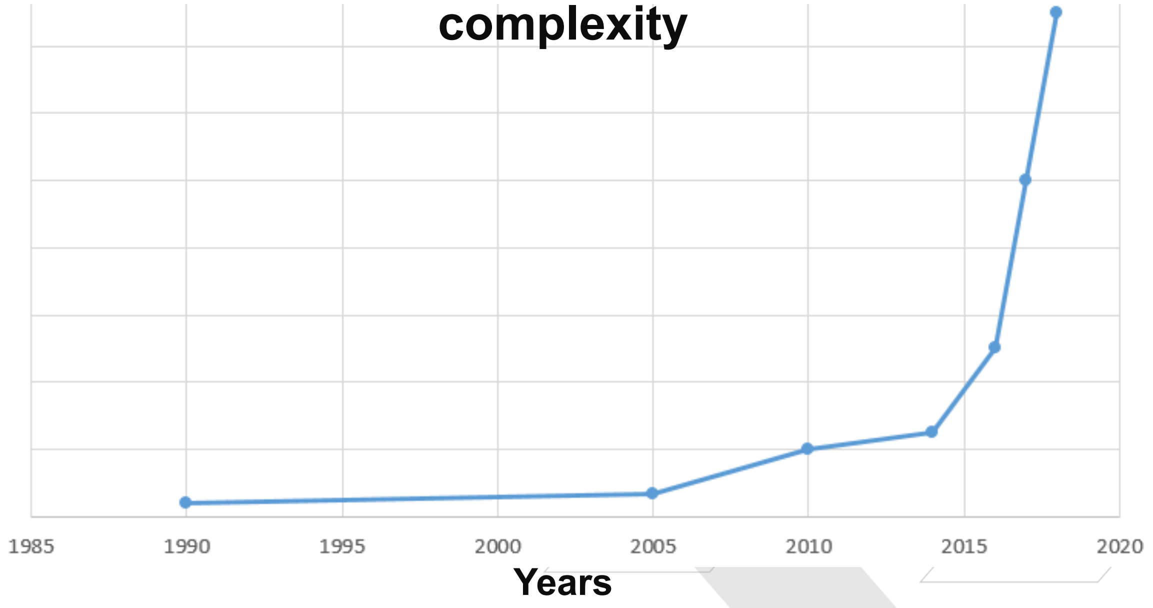
EVOLUTION OF TYPICAL ELECTRIC DRIVES

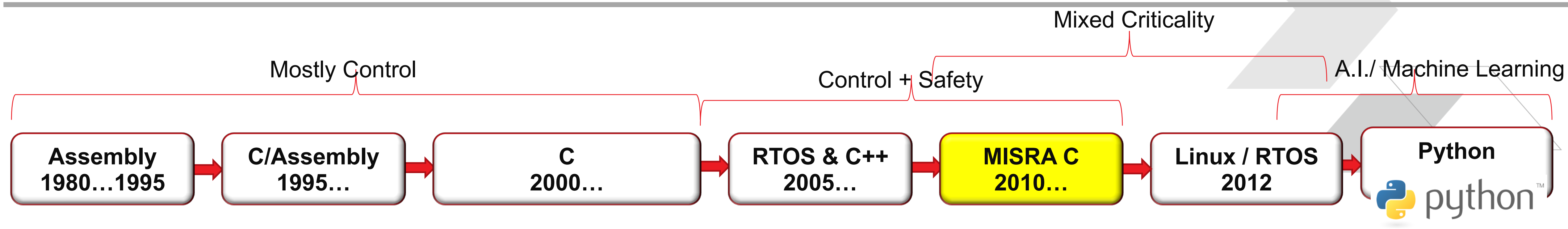


NEW SOFTWARE DEMAND ON DRIVES

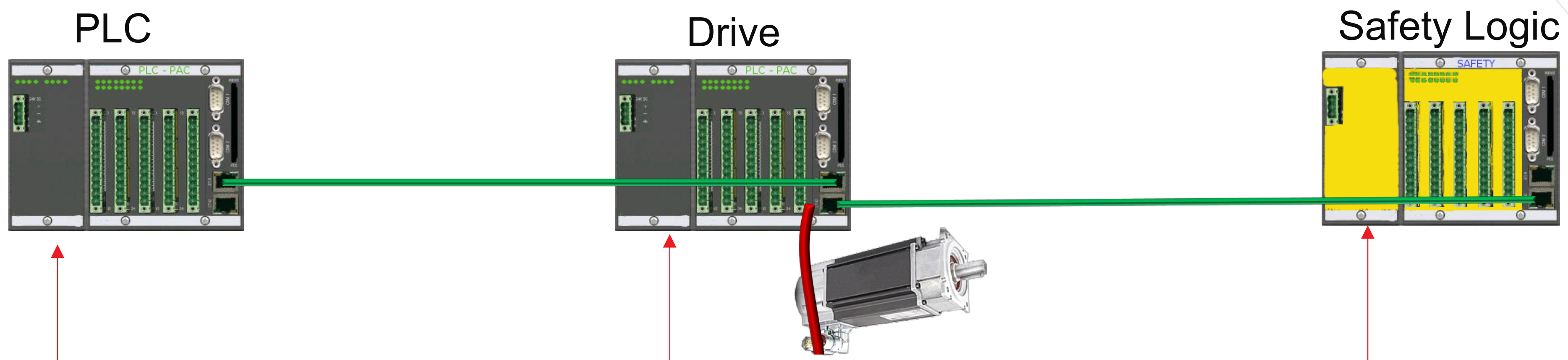


Speed of hardware technology changes and complexity





INTEGRATION INTO MIXED CRITICALITY SYSTEM



Typically "end user application dependent"

Typically "time critical"

"Safety critical"

PLC

- Application 1 | Application 2 | HMI
- RTOS | OS
- High Performance CPU

Drive

- Network I/F | Motion Control | Motor Control
- RTOS | Bare Metal
- Real Time | **Single DSP**

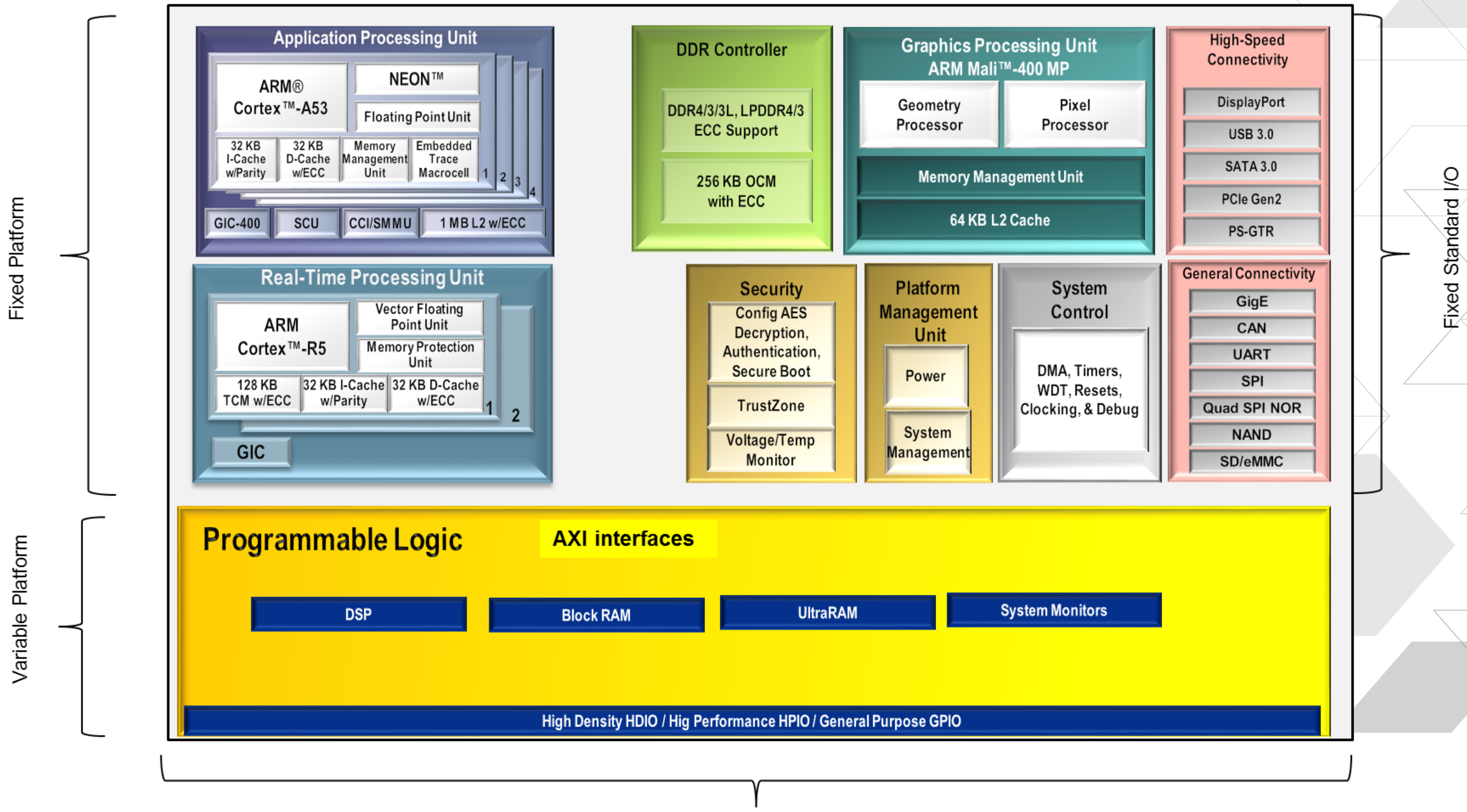
Safety Logic

- Safety Application | Safety Application
- Bare Metal | Bare Metal
- Single CPU | Single CPU
- Voter

Integration into Single Hardware Unit

The Platform

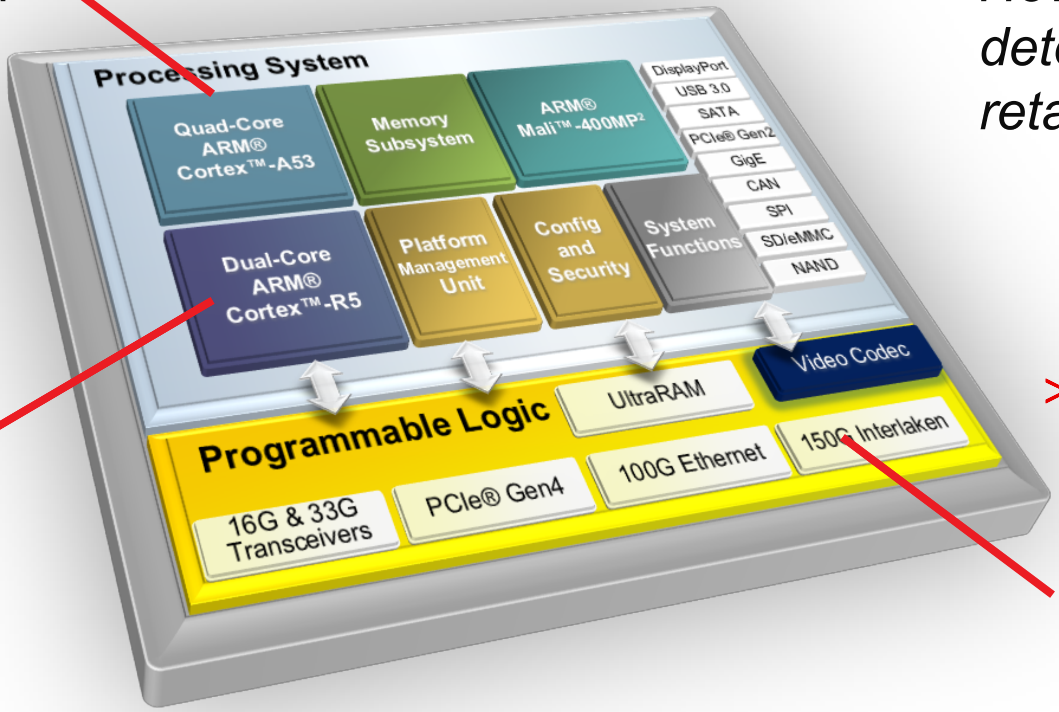
Zynq® UltraScale+™ MPSoC



Determinism vs. Flexibility and Features Tradeoff in MPSoC

> **Most Flexibility and Features:** Application Processors enable software engineers to operate at the highest levels of abstraction and convenience

> **Flexibility and Determinism with Limited Features:** RPU offers straight forward execution of critical code



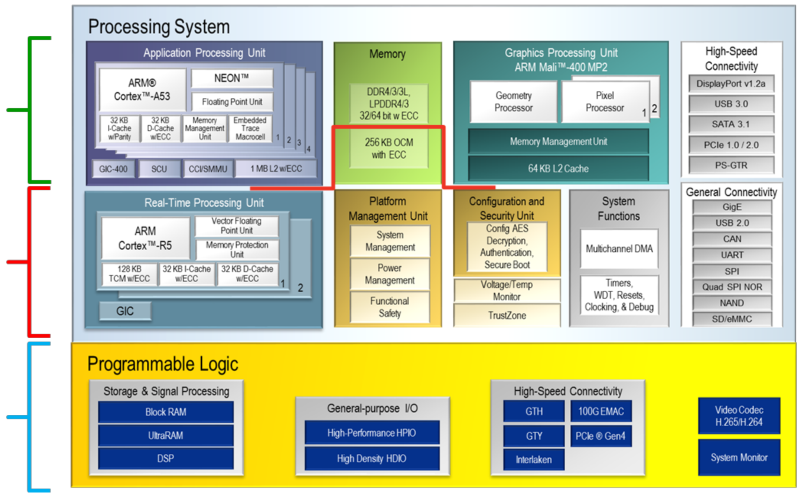
> **Key Question:**
> *How to maximize determinism of APU to retain convenience?*

> **Most Deterministic:** PL brings inherent parallelism and isolation

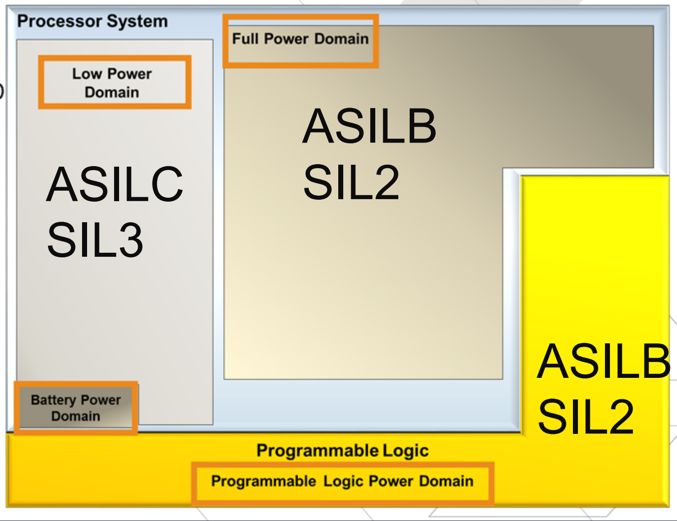
DEVICE STRUCTURE AND PHYSICAL ISOLATION OF DOMAINS

Device Domains

- Full Power Domain (FPD)
- Low Power Domain (LPD)
- Programmable Logic (PL)

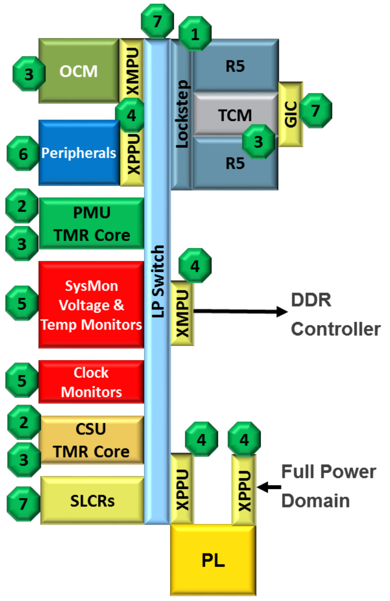


- **Isolated Domains**
  - Battery Power Domain (BPD
  - Low Power Domain (LPD)
  - Full Power Domain (FPD)
  - Programmable Logic (PL)
- **Each power domain is separated**
  - By 50x the min spacing
  - ~10 um

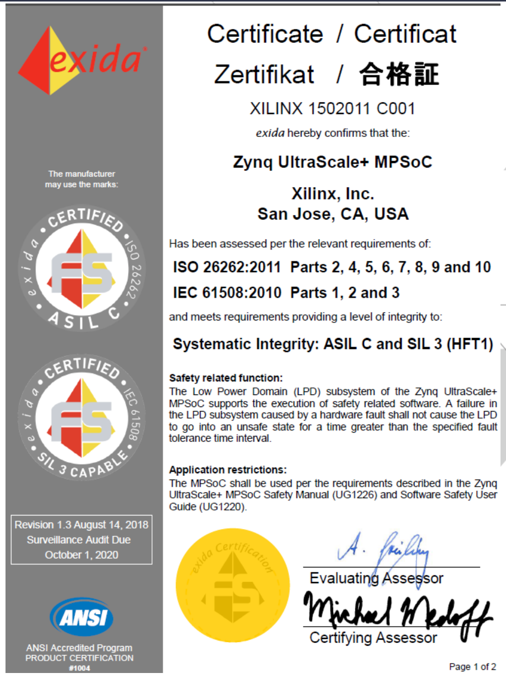


LOW POWER DOMAIN

1. Lockstep for R5s
2. Triple Modular Redundancy (TMR) for Platform Management Unit (PMU) and Configuration & Security Unit (CSU)
3. ECC for TCM, OCM, CSU and PMU RAMs
4. Memory & Peripheral Protection Units provide functional isolation
5. CCF coverage by clock, voltage, and temperature monitors
6. Logic Built In Self Test (LBIST) for checkers & monitors at power-on
   - Peripherals coverage by end-to-end software protocols
7. Software Test Library (STL) for GIC, interconnect, SLCRs & error injection



SIL3 HFT = 1


exida

The manufacturer may use the marks:

CERTIFIED • ISO 26262 • ASIL C • exida

CERTIFIED • IEC 61508 • SIL 3 CAPABLE • exida

Revision 1.3 August 14, 2018
Surveillance Audit Due
October 1, 2020

ANSI
ANSI Accredited Program
PRODUCT CERTIFICATION
#1004

Certificate / Certificat
Zertifikat / 合格証

XILINX 1502011 C001

exida hereby confirms that the:

Zynq UltraScale+ MPSoC

Xilinx, Inc.
San Jose, CA, USA

Has been assessed per the relevant requirements of:

ISO 26262:2011 Parts 2, 4, 5, 6, 7, 8, 9 and 10

IEC 61508:2010 Parts 1, 2 and 3

and meets requirements providing a level of integrity to:

Systematic Integrity: ASIL C and SIL 3 (HFT1)

Safety related function:
The Low Power Domain (LPD) subsystem of the Zynq UltraScale+ MPSoC supports the execution of safety related software. A failure in the LPD subsystem caused by a hardware fault shall not cause the LPD to go into an unsafe state for a time greater than the specified fault tolerance time interval.

Application restrictions:
The MPSoC shall be used per the requirements described in the Zynq UltraScale+ MPSoC Safety Manual (UG1226) and Software Safety User Guide (UG1220).

exida Certification

[illegible]
Evaluating Assessor

[illegible]
Certifying Assessor

Page 1 of 2

XILINX

赛灵思工业物联网研讨会
XILINX IIoT SEMINAR

The system its challenges and solutions

XILINX

THE PARTITION PROBLEM – WHERE AND WHAT

LEGACY Applications

~ 1ms..100ms

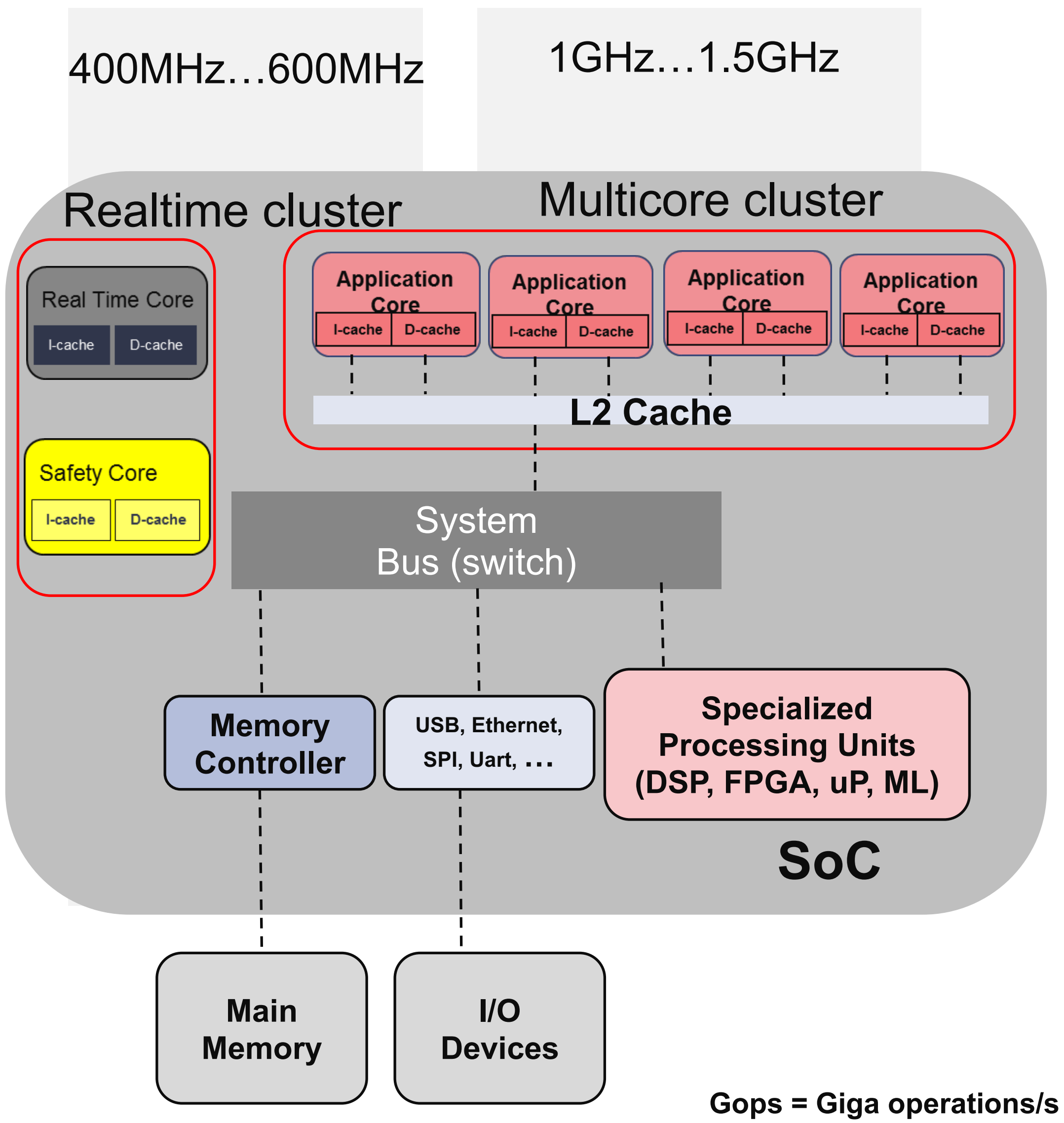
PLC Application
- User dependent
- Time sensitive
- Safety Critical (IEC61508)

~ 10us..10ms

Motor + Motion Control Application
- Time Critical
- Hard real time
- Safety Critical (ISO13849)

~ 64ms..1ms

Field bus Application
- Protocol Dependent
- Time critical



NEW Applications

User's Application
- Sandbox
- Security critical
- Rich OS demand
  - Linux

IoT Gateway Application
- IT dependent
- Security critical

OT/IT Gateway Application
- TSN dependent
- Security critical
- Time critical (synchronization)

~ 10Gops…Tops

Machine Learning Application
- User dependent
- Data intensive
- Performance intensive

Gops = Giga operations/s
Tops = Tera operations/s

MAXIMIZING DETERMINISM FROM APU

- **Don’t put any critical tasks in APU**
  - Use the PL and RPU, that’s what they are there for
- **Use a Real-time Operating System**
  - Significant task scheduling, synchronization, and interrupt services

QNX WIND RIVER Green Hills SOFTWARE

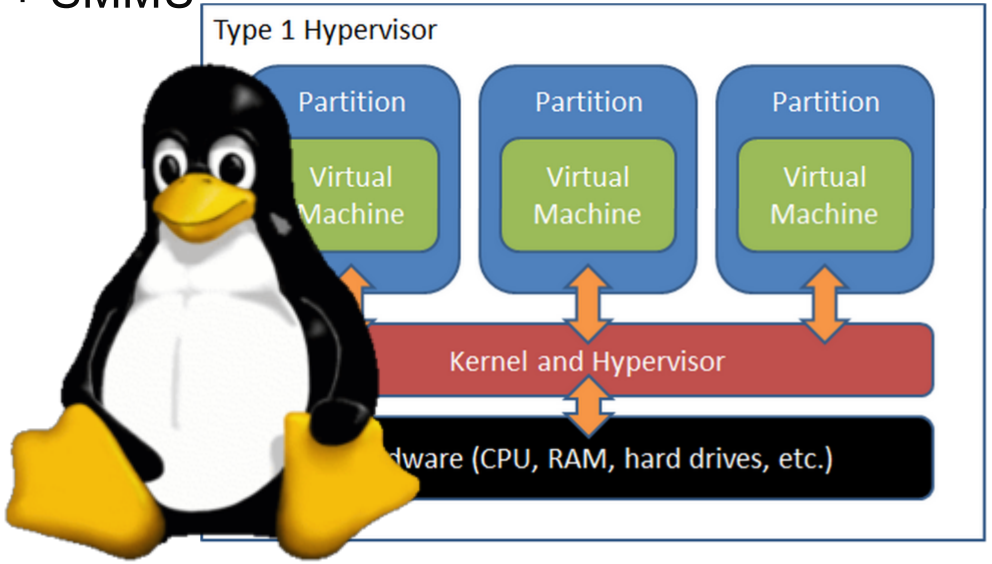
aws mentor embedded expresslogic Micrium Embedded Software

- **Asymmetric Multiprocessing (AMP) in APU for predictability**
  - Use Hypervisors
  - Make use of cache coloring

MORE MAXIMIZING DETERMINISM FROM APU

> **Install PREEMPT_RT patch to Linux**
>> Preempt Lower Criticality Tasks when in conflict with Higher Criticality Tasks

> **Hypervisor + Hardware Virtualization in Arm v8 Architecture**
>> Enforces Isolation at OS-level and allocates specific processor, peripherals, and memory regions
>> Allows Virtualization of processors, interrupts, memory, timers
>> Supported in HW through exception levels + SMMU

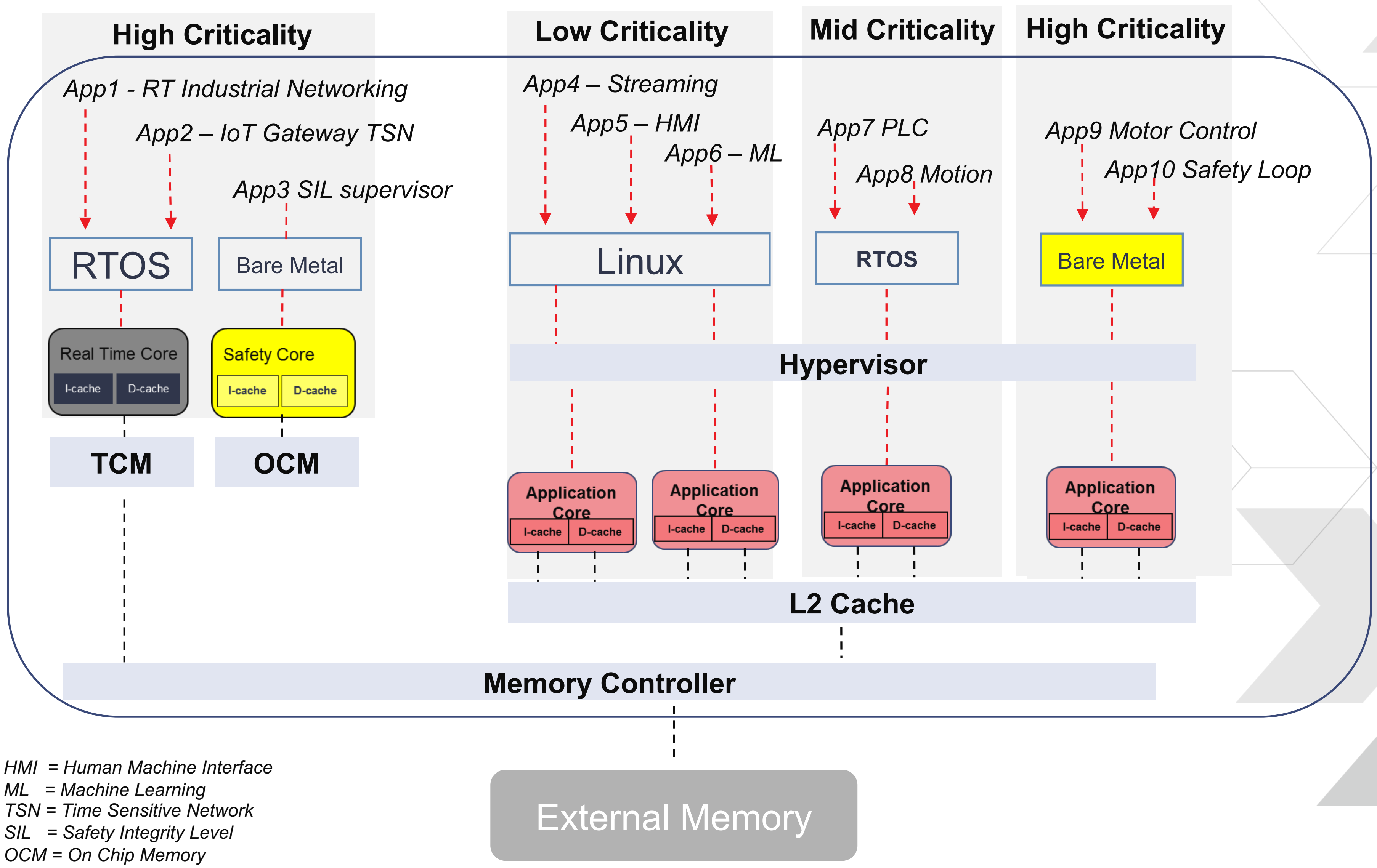


赛 灵 思 工 业 物 联 网 研 讨 会
XILINX IIoT SEMINAR

Hypervisor and Cache Coloring improve predictability

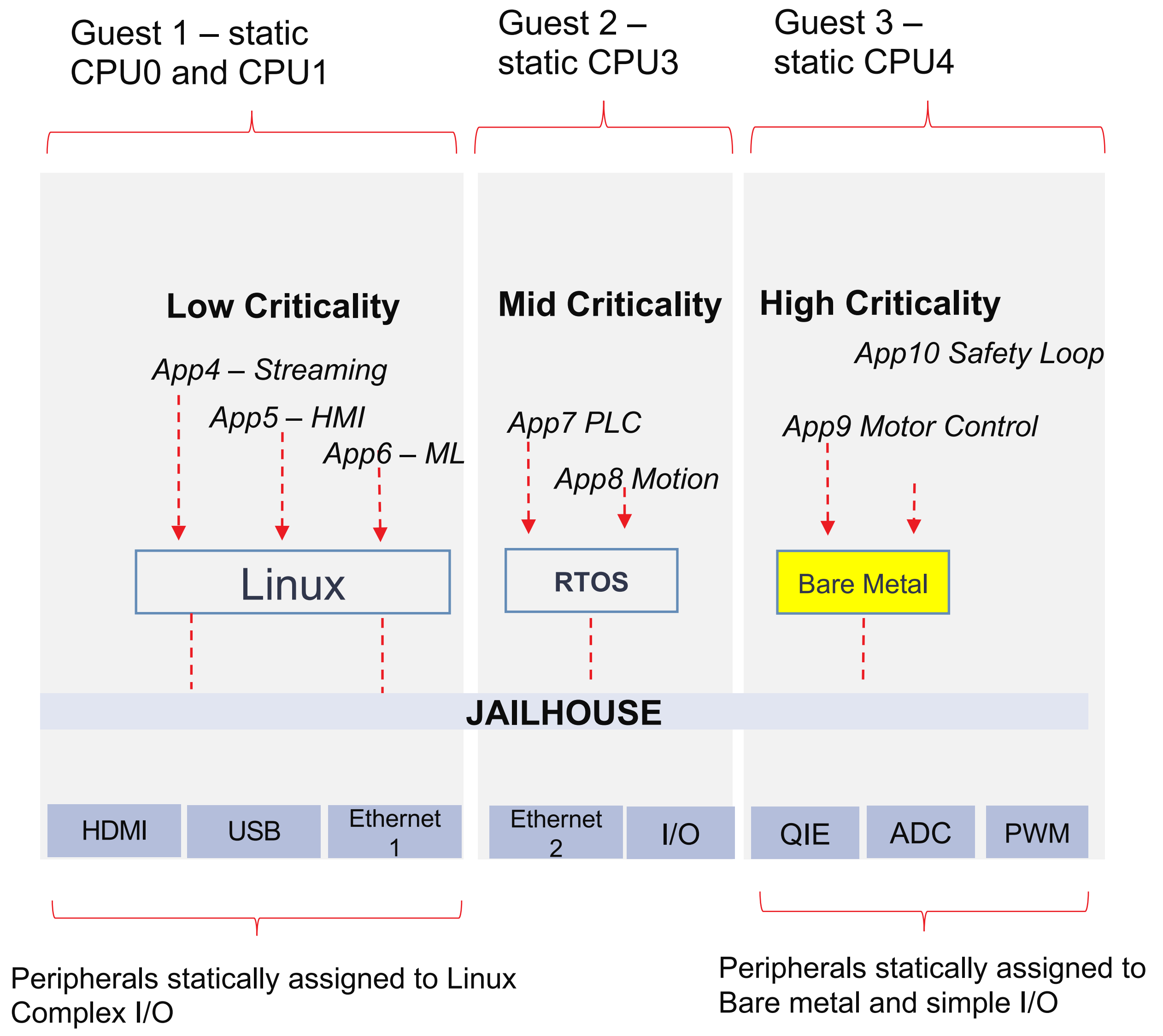
XILINX

CONCEPTUAL ALLOCATION OF APPLICATIONS – CASE #1



HMI = Human Machine Interface
ML = Machine Learning
TSN = Time Sensitive Network
SIL = Safety Integrity Level
OCM = On Chip Memory
TCM = Tightly Coupled Memory

A “LEAN” HYPERVISOR THE JAILHOUSE CASE

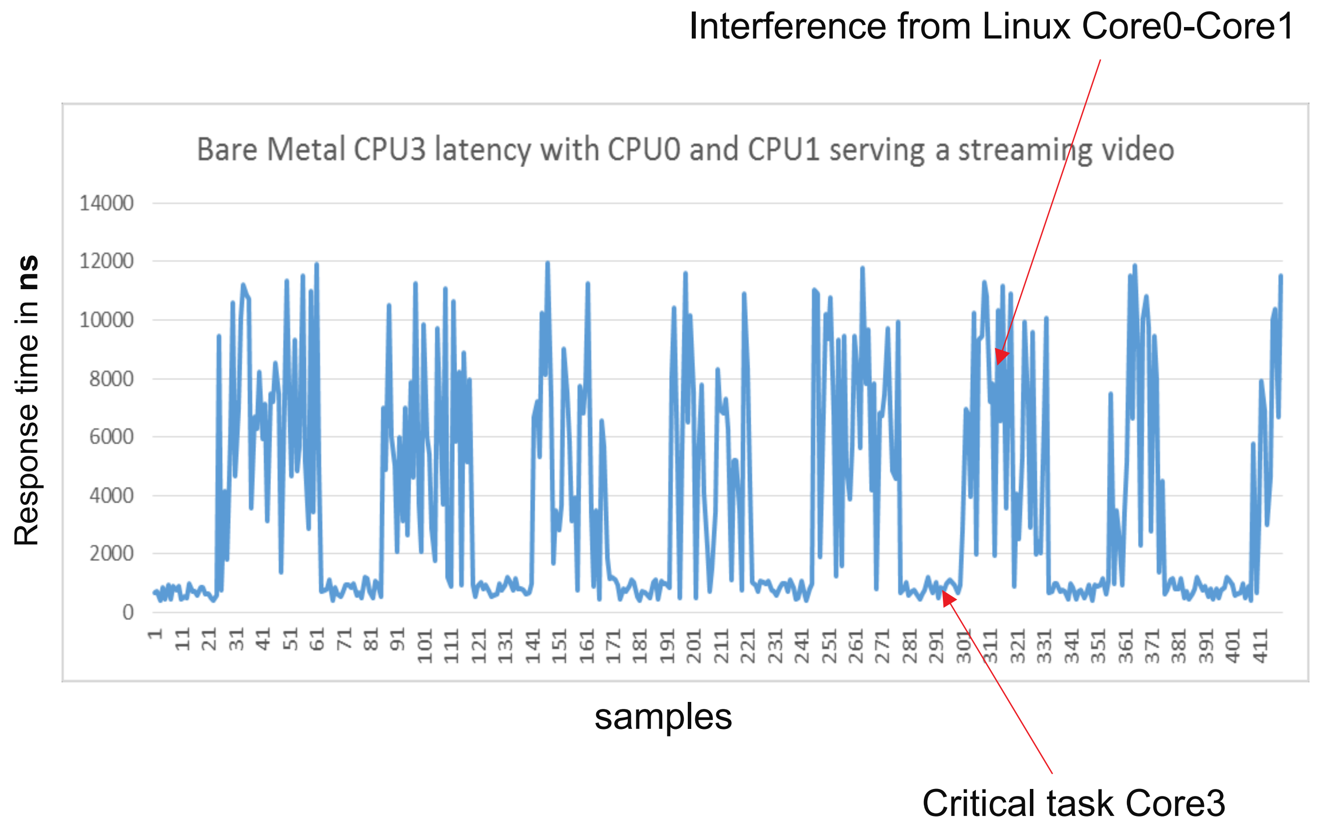


> **Jailhouse is a partitioning hypervisor**

- Transforms Symmetric Multiprocessing into Asymmetric Multiprocessing
- Bootstrapped via a Linux as type 2 hypervisor
- Changes into a type 1 hypervisor after booting
- Provides consistent isolation among cores using a “root” and “inmate” cells.
- Assigns peripherals to specific cores statically with no reallocation
- Minimum impact on latency – no scheduling
- It is an Open Source project conceived by SIEMENS https://github.com/siemens/jailhouse

Keep it simple and straightforward!

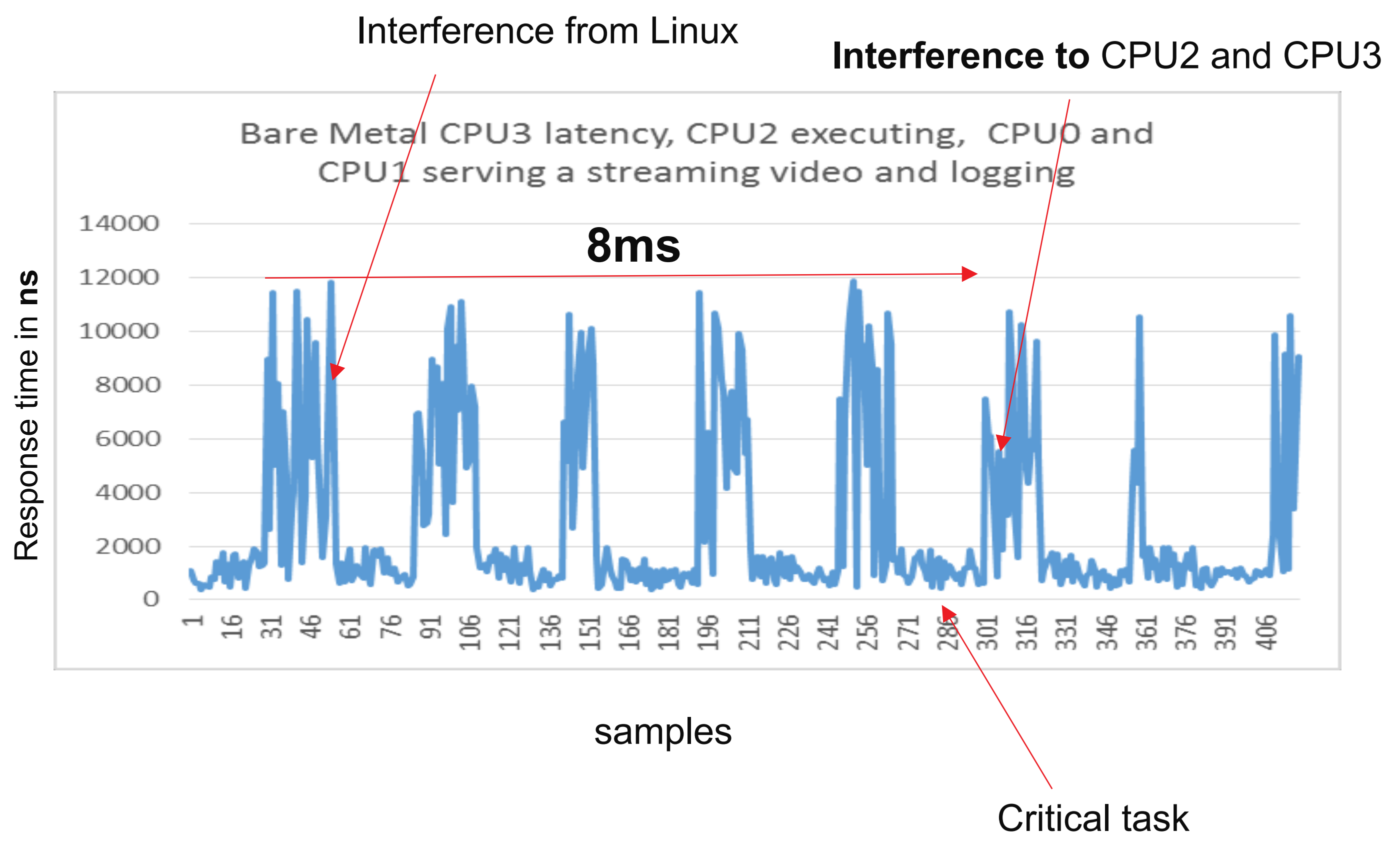
DO WE HAVE ABSENCE OF INTERFERENCE? BENCHMARK #1



> **Conditions**
- CPU3 executing a 48Kbyte code every **256us** as motor control task + safety loop
- CPU2 dormant
- CPU1 executing streaming of 2 Megabytes of data
- CPU0 executing DDR access for data logging with 256 Kilo bytes stream

> **Measurements**
- CPU3 executing with response time between **~400ns and ~12000ns** a 30x deviation!
- **Clear and significant interference!**
- **Likely to be the L2 cache**

DO WE HAVE ABSENCE OF INTERFERENCE?



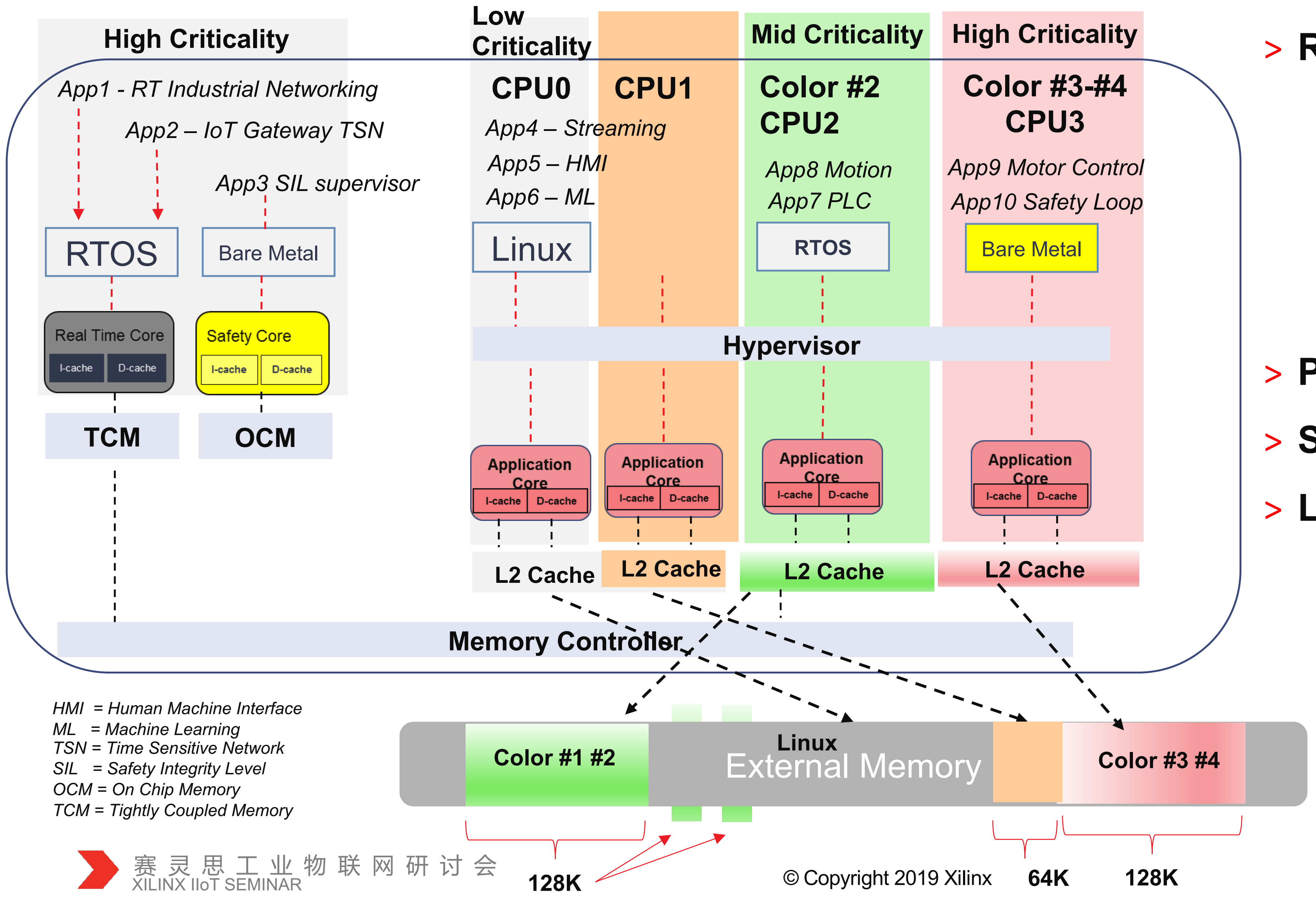
> **Conditions**

- CPU3 executing a 48Kbyte code every **256us** as motor control task + safety loop
- CPU2 executing a "PLC" like 48K byte code every **8ms**
- CPU1 executing streaming of 2 Megabytes of data
- CPU0 executing DDR access for data logging with 256 Kilo bytes stream

> **Measurements**

- CPU3 executing with response time between **~400ns and ~12000ns** a 30x deviation!
- CPU2 executing with response time between **~2000 and ~12000 ns**
- **Clear and significant interference!**
- **Likely to be the L2 cache again**

CACHE COLORING + JAILHOUSE RUNNINGIN COLORED PARTITION



HMI = Human Machine Interface
ML = Machine Learning
TSN = Time Sensitive Network
SIL = Safety Integrity Level
OCM = On Chip Memory
TCM = Tightly Coupled Memory

- **Results**
  - Interference amongst Core 3 and Core 2 is eliminated
  - Contiguous memory map in function of the number of color assigned to CPU
  - Cache "lockdown" same size of number assigned colors
- **Predictability improved**
- **Separation improved**
- **Linux re-incarnated**
  - Coloring no interference

XILINX

Could we do even better?
Use of PL and more processors

MicroBlaze 32/64 bit soft processor

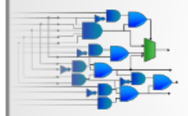
Multiple Configurations*

Microcontroller
- 188 MHz, 197 DMIPs
- 1175 LUTs, 811 FFs

Real-Time Processor
- 161 MHz, 213 DMIPs
- 2461 LUTs, 2125 FFs

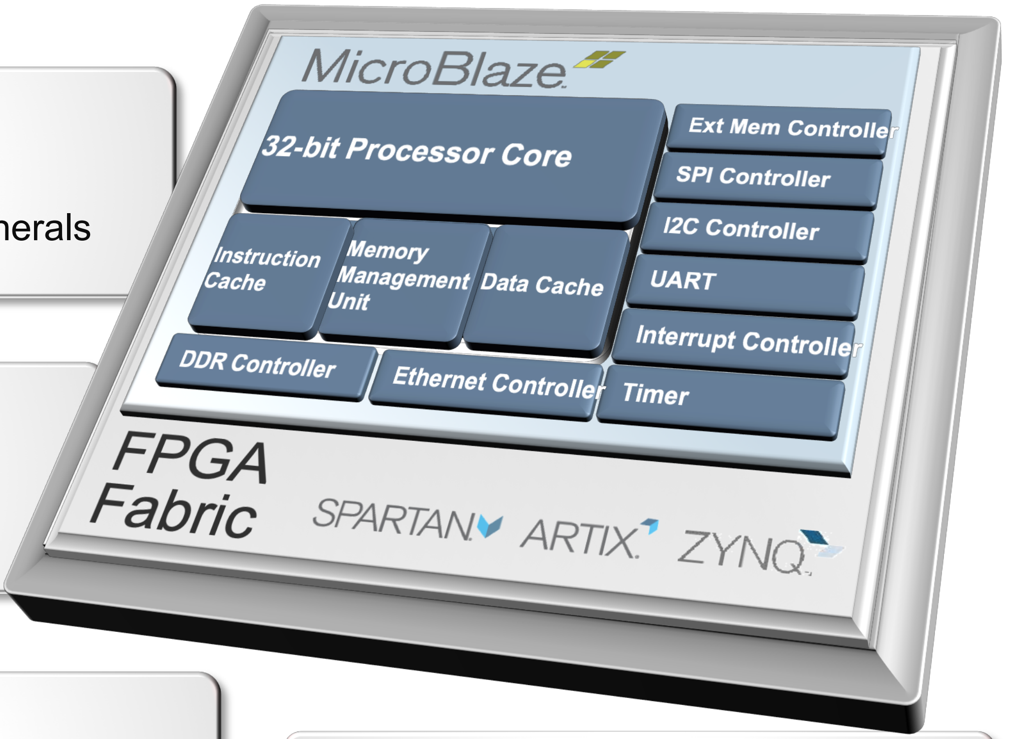
Application Processor
- 130 MHz, 172 DMIPs
- 4342 LUTs, 3812 FFs

…

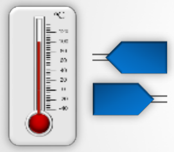


FPGA Fabric
- Custom Engines
- Extendable Peripherals

AXI Interconnect
- Plug-and-Play IP
- Standards Compliant

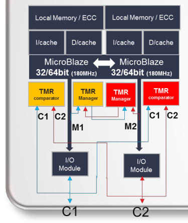


Security
- HMAC
- Config AES

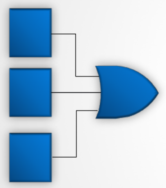


Integrated Analog
- Temp & Power Monitor

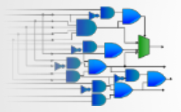
IP Sub-System



Lockstep Capable
- Full 1oo2 (one out of two)
- Safety and Security



TMR Capable
- SEU Mitigation
- Voter Circuit



MCS
- Lean configuration
- Integrated peripherals

* Performance and utilization for a Spartan-7 -2 Speed Grade Device, using 32-bit mode

64-bit MicroBlaze Implementation

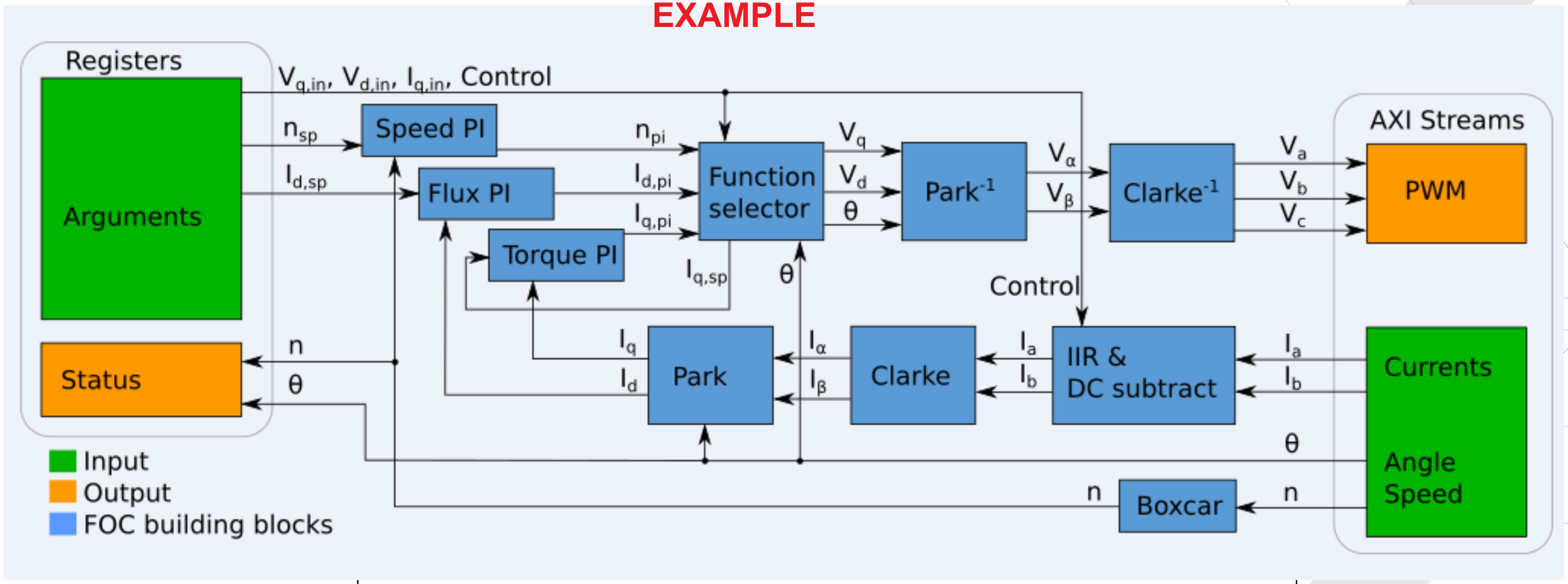
> **MicroBlaze now supports 64-bit addressing (available in 2018.3)**

- Available in v11.0 (2018.3 release) as a configuration option
- Default is 32-bit implementation so existing designs are not impacted when upgrading
- Allows access to up to 16Exaybtes of system memory
- The full Zynq UltraScale+ address maps are now usable by MicroBlaze
- All registers and supporting IP have also been updated
- Works with XSCT – manual configuration in XSDK GUI
- No change to performance, only impacts resource utilization
- Benchmarks for all current Xilinx device families are posted in the MicroBlaze Reference Guide – UG984.

Can we improve even further determinism? HLS and Model Composer

DETERMINISTIC MOTOR CONTROL FIELD ORIENTED ALGORITHM

EXAMPLE

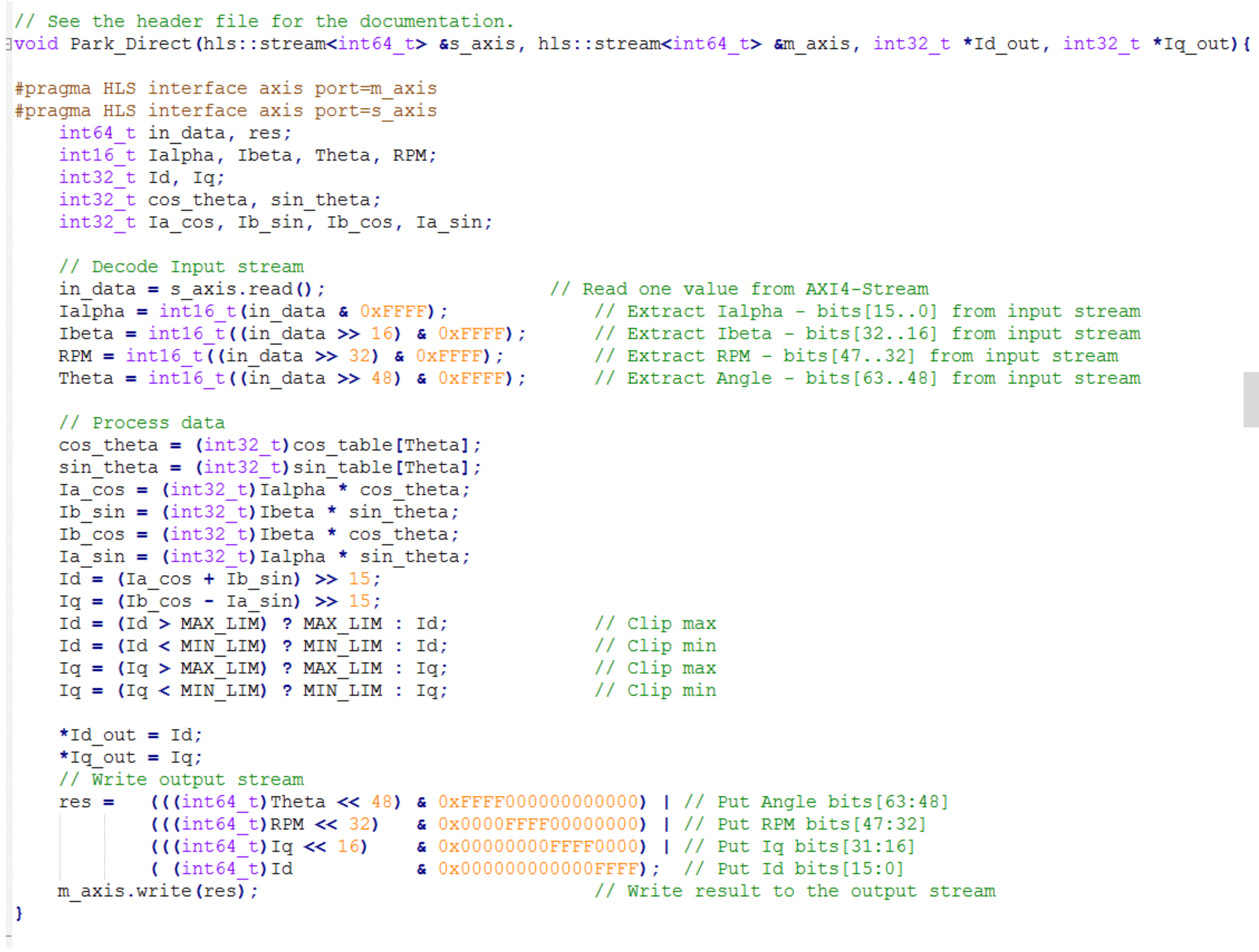


1 Processor interface

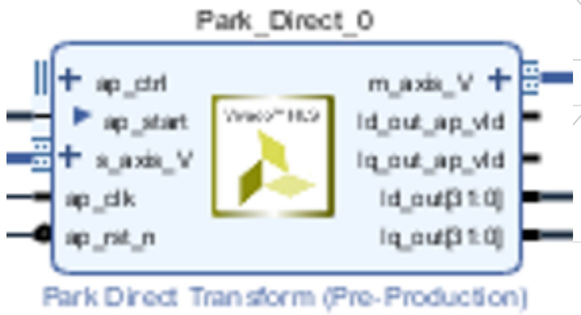
7 algorithmic components

2 I/O components

FOR EACH HW ALGORITHM YOU HAVE A C/C++ DESCRIPTION

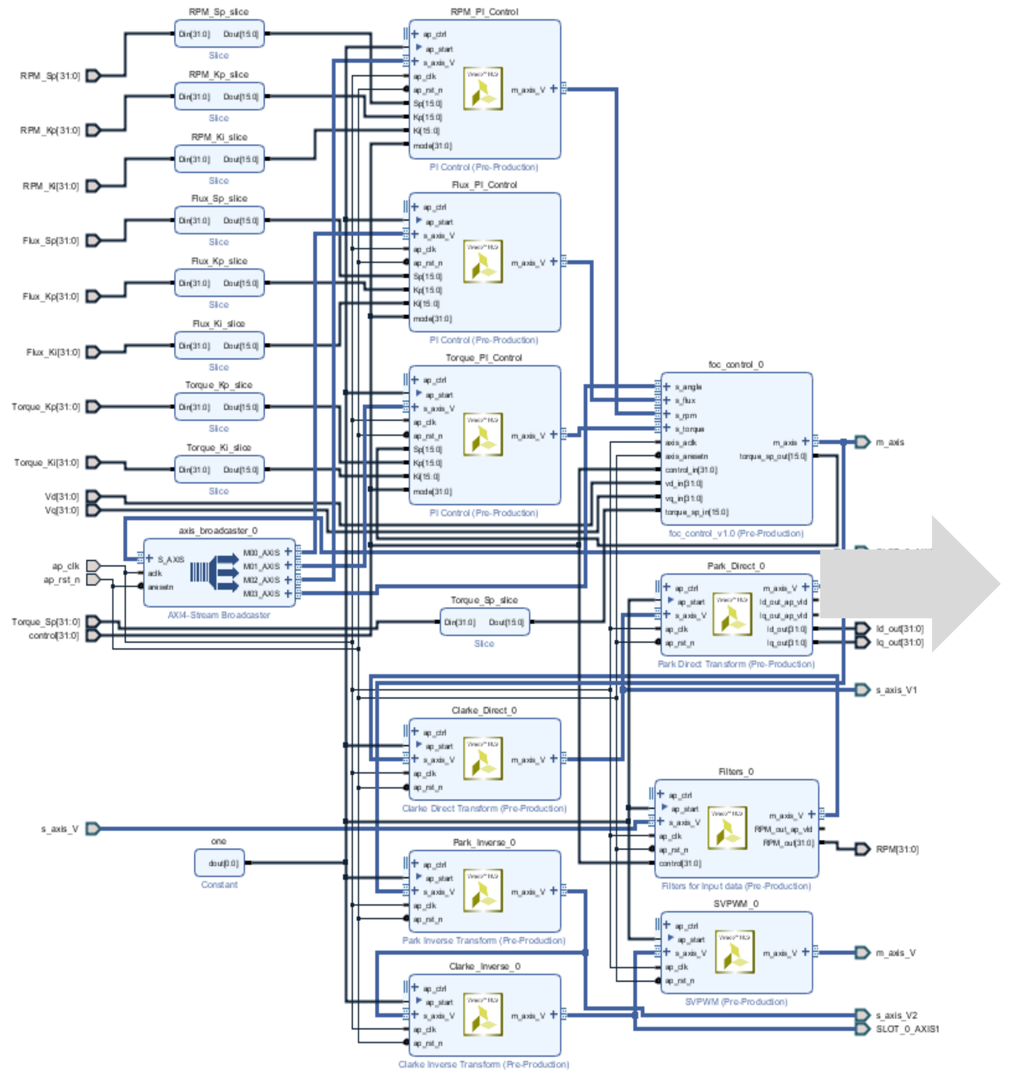


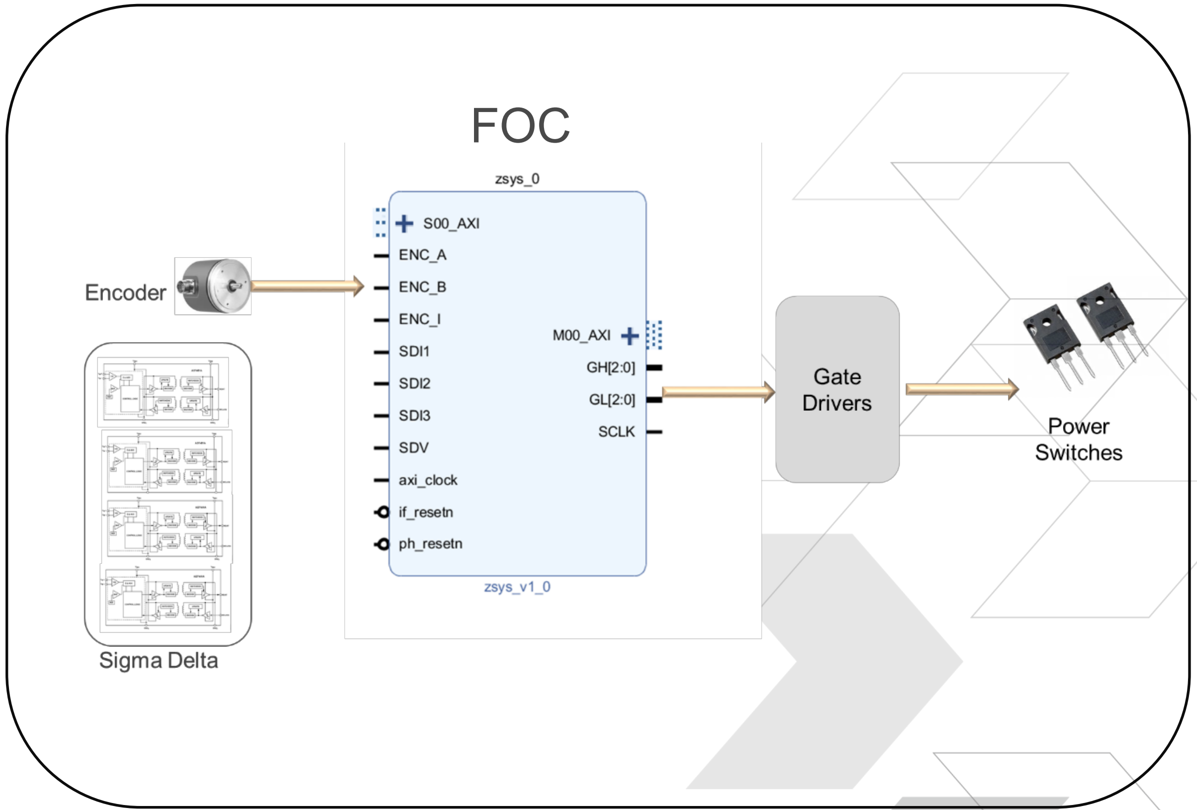
```
// See the header file for the documentation.
void Park_Direct(hls::stream<int64_t> &s_axis, hls::stream<int64_t> &m_axis, int32_t *Id_out, int32_t *Iq_out){

#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

THE FOC BLOCK BECOMES AN HW LIBRARY COMPONENT

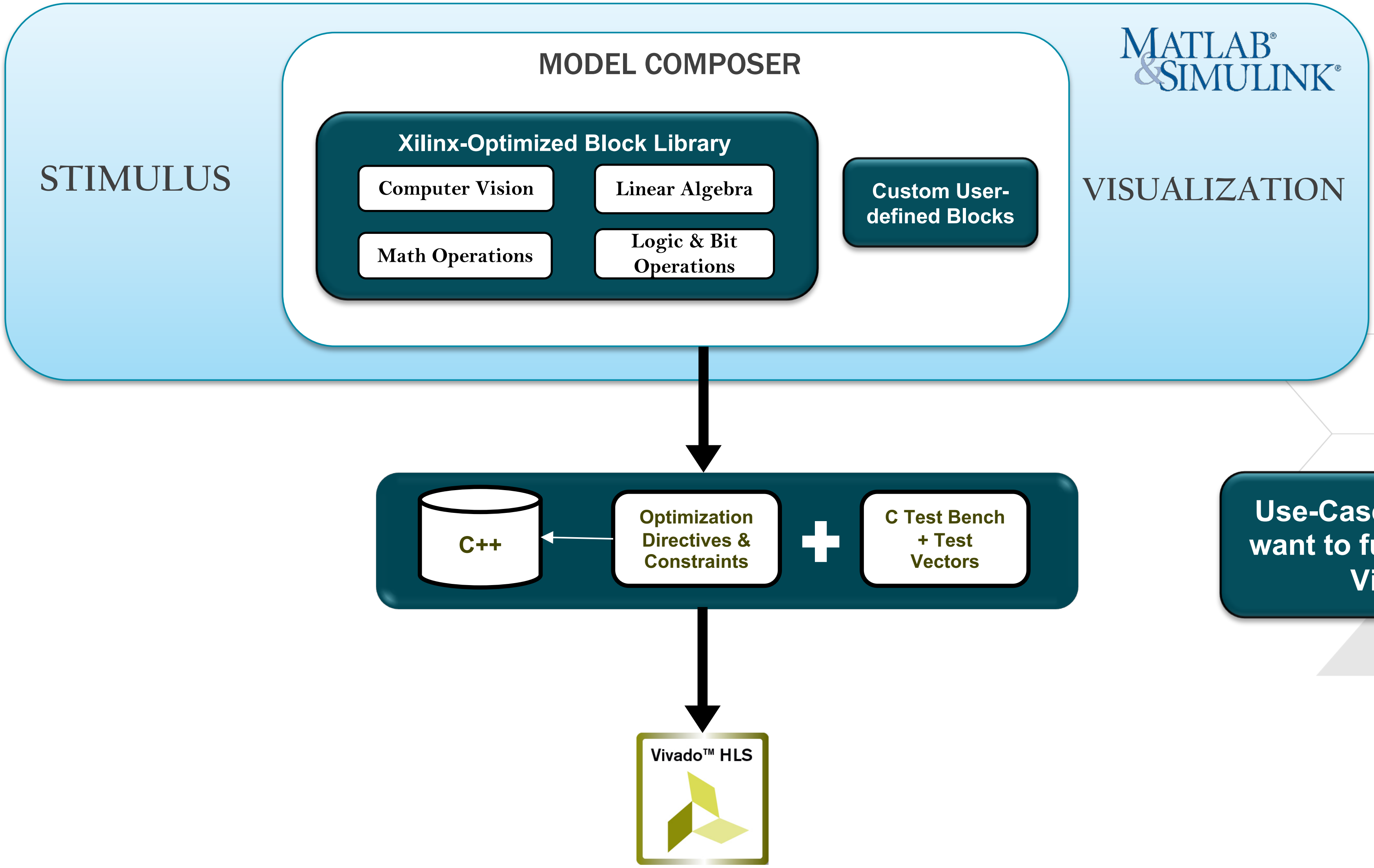




XILINX

Directly from Simulink…

Model Composer to Vivado HLS



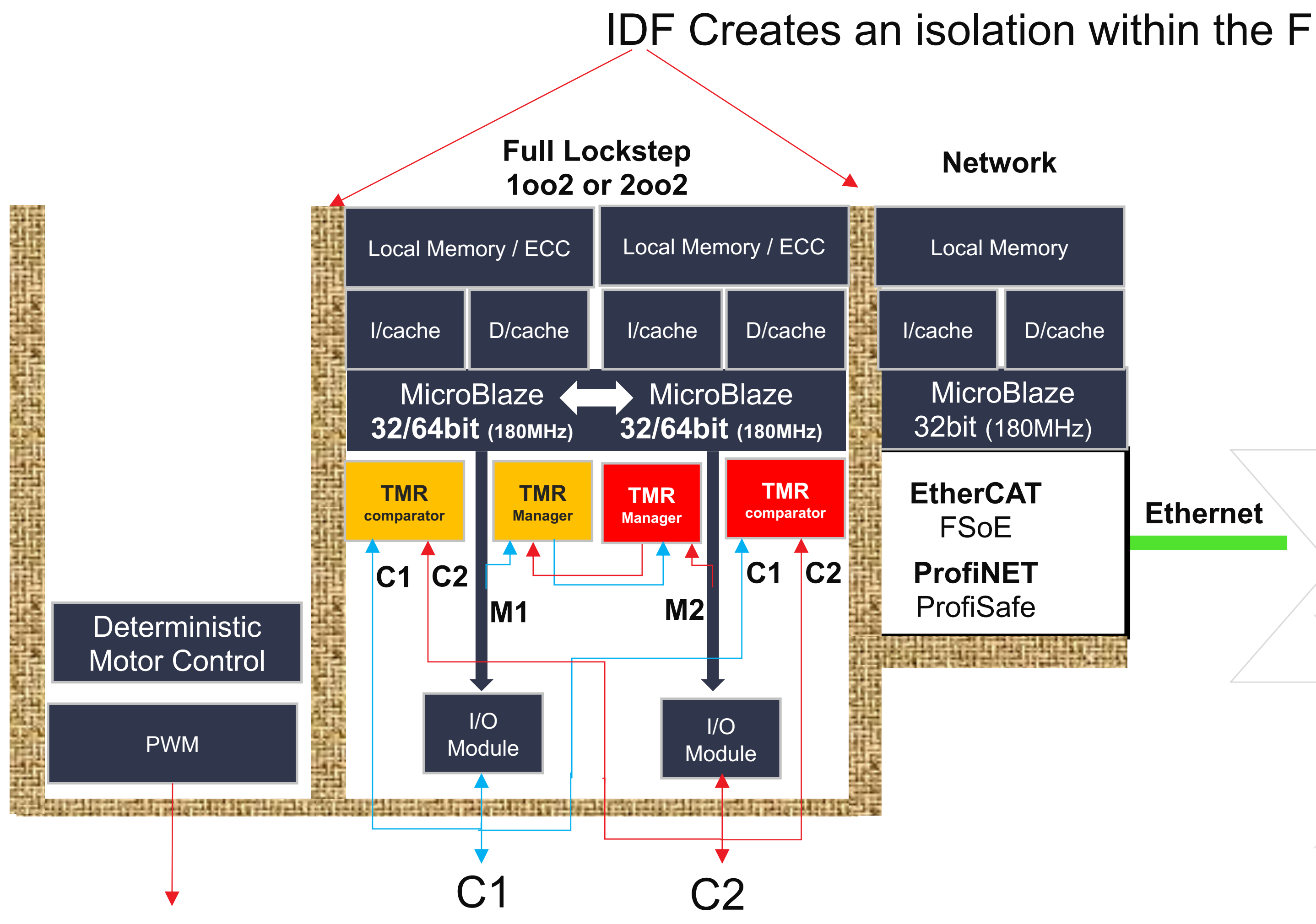
Use-Case: For Users who want to further optimize in Vivado HLS

赛灵思工业物联网研讨会
XILINX IIoT SEMINAR

Final result

XILINX

EXTENSION TO PROGRAMMABLE LOGIC DETAIL



THE ISOLATION DESIGN FLOW

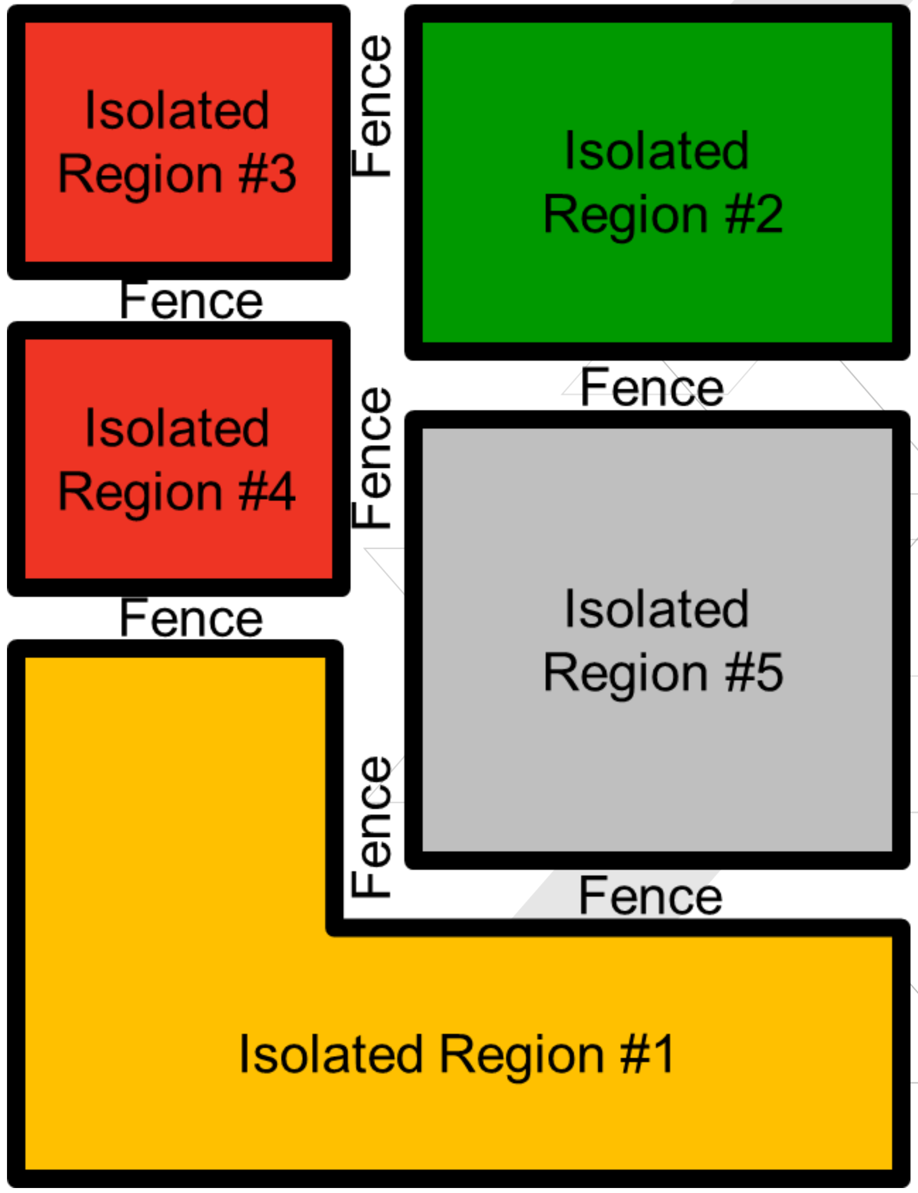
> Enables mixed criticality designs to be co-located on the same fabric

> Limits FMEDA (Failure Modes Effects, and Diagnostic Analysis) to the Safety-Related function of interest instead of the entire fabric.

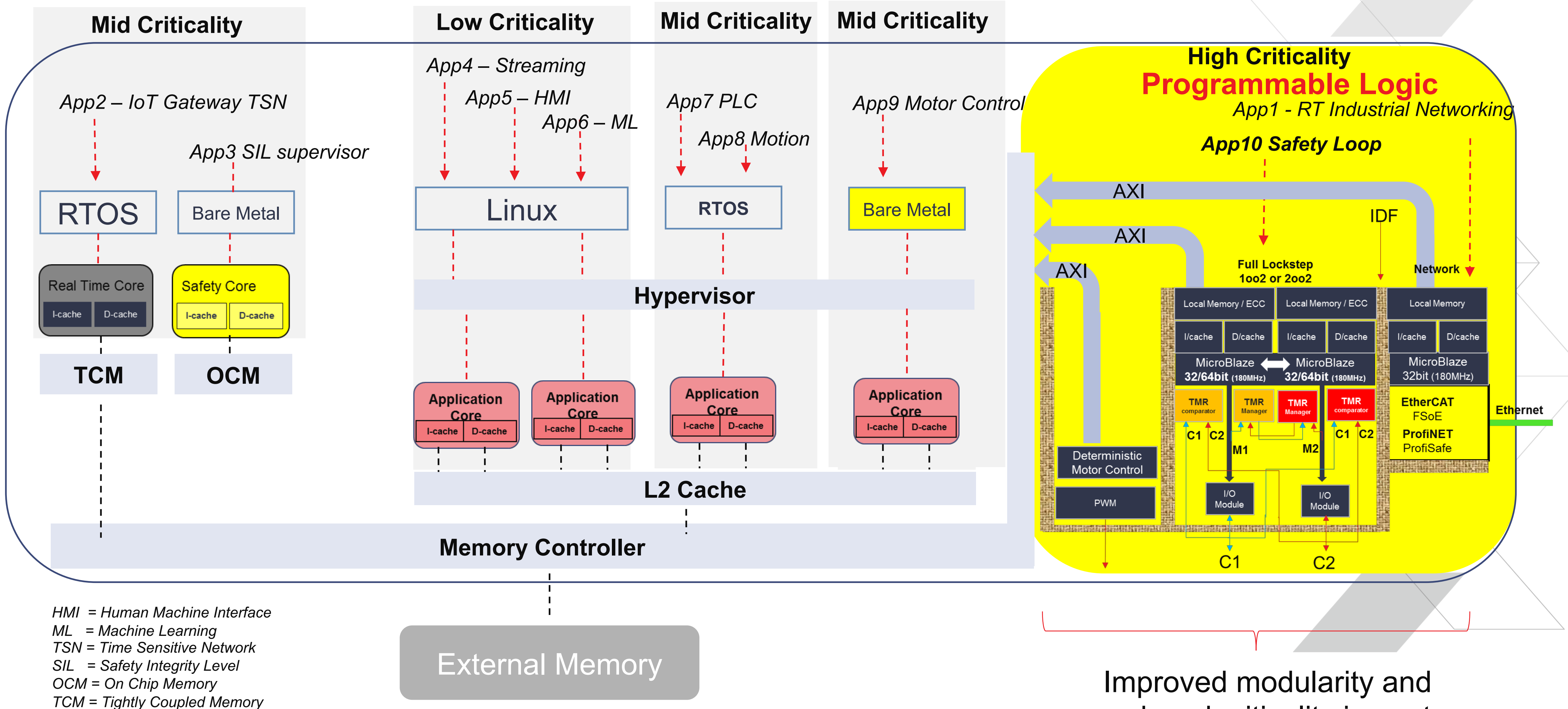
> Mitigates Common Cause Failures

> Separates Safety and Non-Safety functions

> Enabler for HFT=1 under IEC 61508 Part 2 Annex E



CONCEPTUAL ALLOCATION OF APPLICATION + BENEFIT OF PL– CASE #2



HMI = Human Machine Interface
ML = Machine Learning
TSN = Time Sensitive Network
SIL = Safety Integrity Level
OCM = On Chip Memory
TCM = Tightly Coupled Memory

Improved modularity and reduced criticality impact

CONCLUSION

> Zynq Ultrascale + as platform

> Safety by design

> Highest performance and enhanced predictability

> Scalability with dedicated processing units in PL

> Scalability with dedicated hardware accelerators designed with HLS or Matlab/Simulink

> Isolation design flow for reduced interference

> Other interesting options exploring Xilinx website.

Adaptable.
Intelligent.

赛灵思工业物联网研讨会
XILINX IIoT SEMINAR

XILINX