SymPLe: Design and Development of Verifiable FPGA-based PLC Architecture for Safety Critical Nuclear Power Applications

A US-DOE Funded Project under the NEET Program

11th International Workshop on the Application of FPGAs in NPPs

Matt Gibson, Program PI, EPRI

Dr. Carl Elks, Co-PI, Virginia Commonwealth University

Dr. Ashraf Tantawy, Research Professor, Virginia Commonwealth University

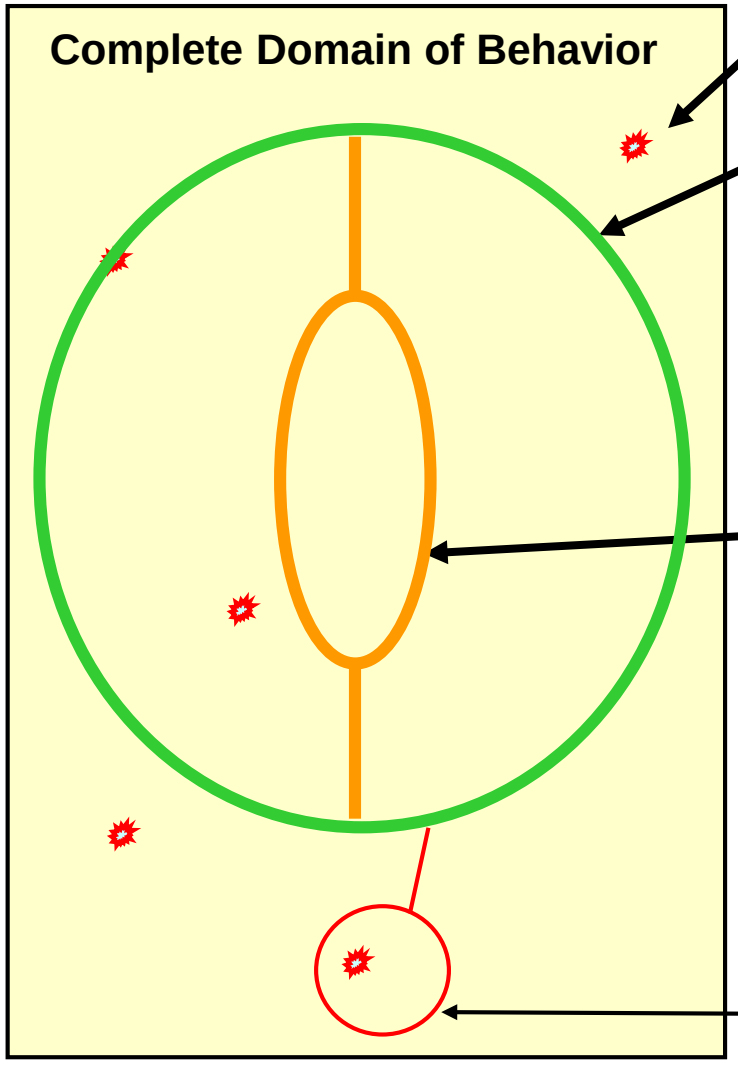
Rick Hite, Smitha Gautham, Chris Deloglos, Athira Jayakumar - Virginia Commonwealth University

Jason Moore, MathWorks

Andrew Nack, Paragon Inc.

Oct. 11th 2018

Software Driven Architectures: Cyclic Behavior with Non-Deterministic Characteristics



May contain residual digital faults

Path exercised continuously in normal situations

Influence factors during continuous operation:

- normal process inputs (validated before use)
- short-term memory (as little as possible)
- clock interrupts (thorough verification)
- *(process-related interrupts: none)*
- *(resource management: static)*

Path exercised in occasional but tested situations

Influence factors that could disrupt cyclic behavior:

- initialization (only once)
- operator requests (single channel)
- hardware failures (single channel)
- exceptions (very simple)
- *(particular dates & times: avoided)*
- plant transients: affect all channels

Path exercised in unanticipated or untested situations

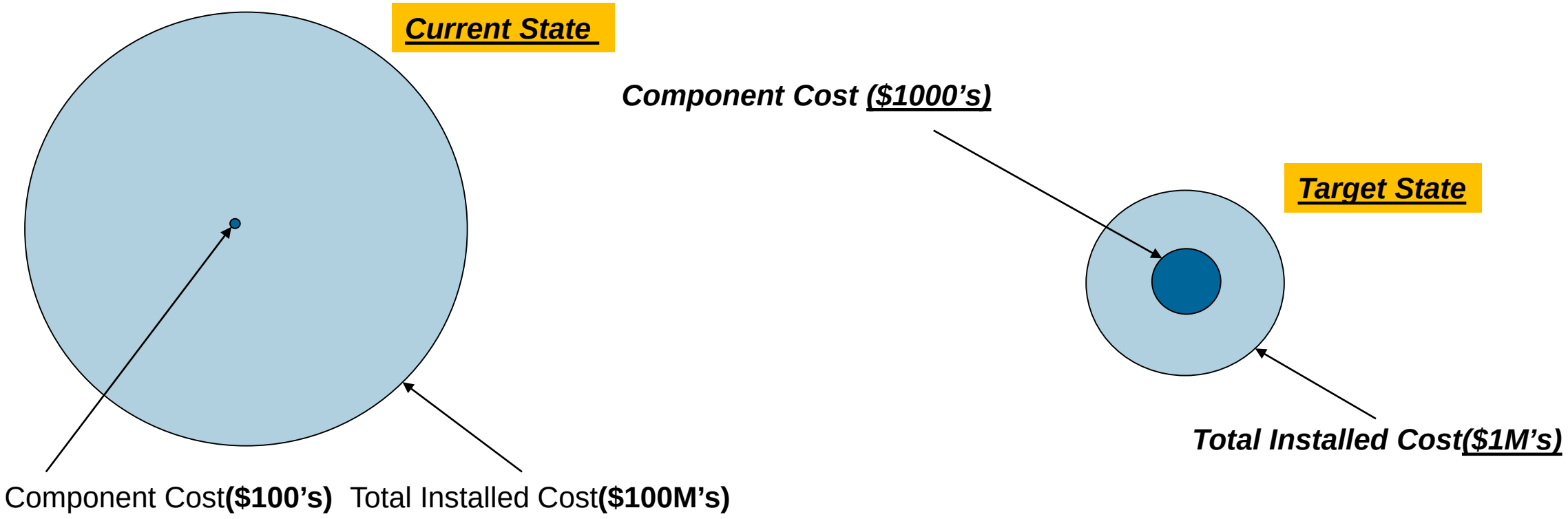
Can the System be *constrained* to well-understood and tested trajectories?

Research Perspective

- Premise:
  The general embedded computing CPU *is not optimized for high integrity applications* with modest functionality and high verification requirements, but rather for mass production commodity markets.
  - Large functional fault/attack surface makes them unpredictable if they exit their normal logic path due to software errors, misconfiguration, or malicious attack.
  - Maker Movement and low volume manufacturing of optimized components are now a reality and can be exploited to create components verified to function properly.
  - Digital components directly manufactured to 10CFR50 Appendix B are now within reach.

Can We Effectively Constrain?

- Use of general purpose, massed produced processors and FPGA modules has resulted in systems that are challenging to verify.
- While inexpensive, these parts contain orders of magnitude more capability than is needed for a specific application.

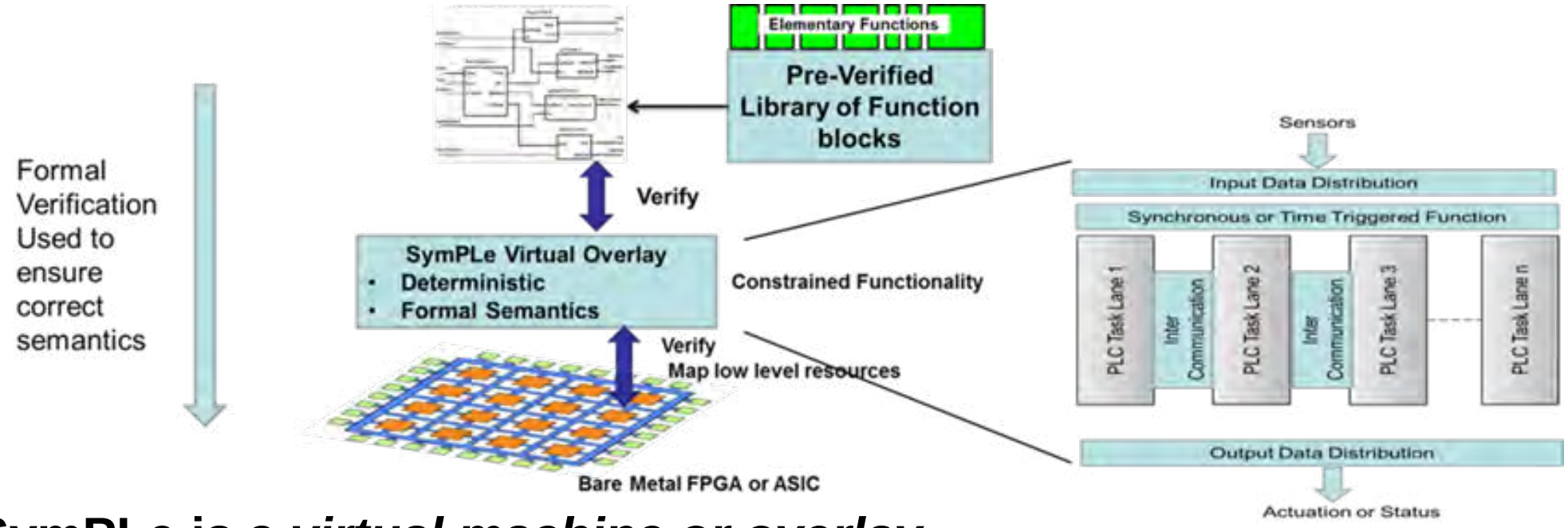


Research Perspective

- I&C systems in the context of nuclear power may not need to be derivatives of software intensive systems and by extension, not carrying the complexity associated with the SW intensive systems.

- Our approach called SymPLe is to rethink digital I&C from a perspective of three views: Simplicity, Extensibility and Verifiability.

SymPLe Concept



SymPLe is a ***virtual machine or overlay***

- *SymPLe* is a architectural viewpoint that seeks to maximize reasoning, transparency and safety evidence while avoiding unnecessary complexity.
- *Engineer Accessible*: By adopting overlay architecture, we hope to make SymPLe extensible like a CPU based architecture – function blocks are the execution functions.
- SymPLe is limited in what it can do – it trades computational power for verifiability.

SymPLe Architectural Concepts

- Constrain design to favor verifiable execution behavior
- Constrain program composability rules to favor testing and comprehension
- Design for predictability
  - Well understood behaviors
  - Well formed semantics – no side effects
  - *Engineer Accessible*: SymPLe is explicitly specified in a manner (e.g., language; structure) that is comprehensible to the community of its users and reviewers.
  - Reusable and verified Hardware Function Blocks

Basic Tenants of SymPLe

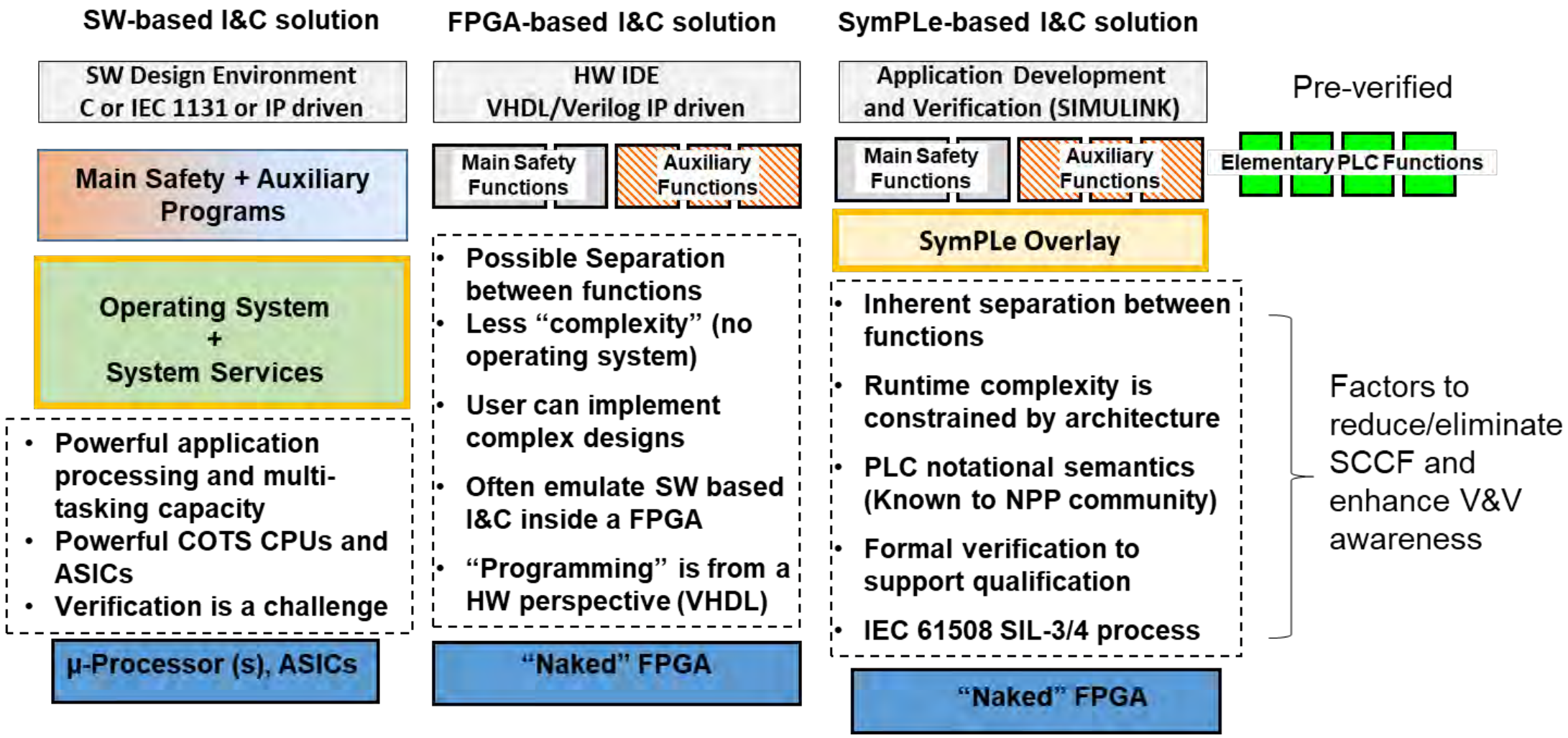
- *Composability* - The behavior of "composed" element is a composition of the behaviors of its constituent elements, with well-defined, unambiguous rules of composition.
  - –Interfaces of elements are unambiguously specified, including behavior.
  - –Interactions across elements occur only through their specified interfaces
- *Orthogonal* - The system is modularized using principles of information hiding and separation of concerns, considering orthogonality[1] of functions and data.
  - –Only required interactions are allowed. The architecture precludes unwanted interactions and unwanted, unknown hidden coupling or dependencies.
  - –Each element (e.g., a FB unit) is internally well-architected and relatively simple.
- *Determinism* - The system is architected (satisfying conditions above) to be predictable and synchronous in it's execution behavior.

These Tenants along with a formal design and verification methodology allows any CCFs to be identified before deployment, and enhances the ability to reason about system during qualification.

1 = a relatively small set of primitive constructs can be combined in a relatively small number of ways to build the control and data structures of program behavior

Comparative View of I&C "Stacks"

What's the difference: SW vs. FPGA vs. SymPLe based I&C

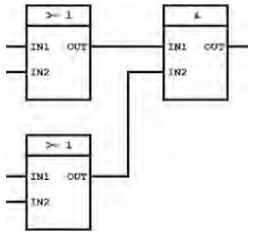


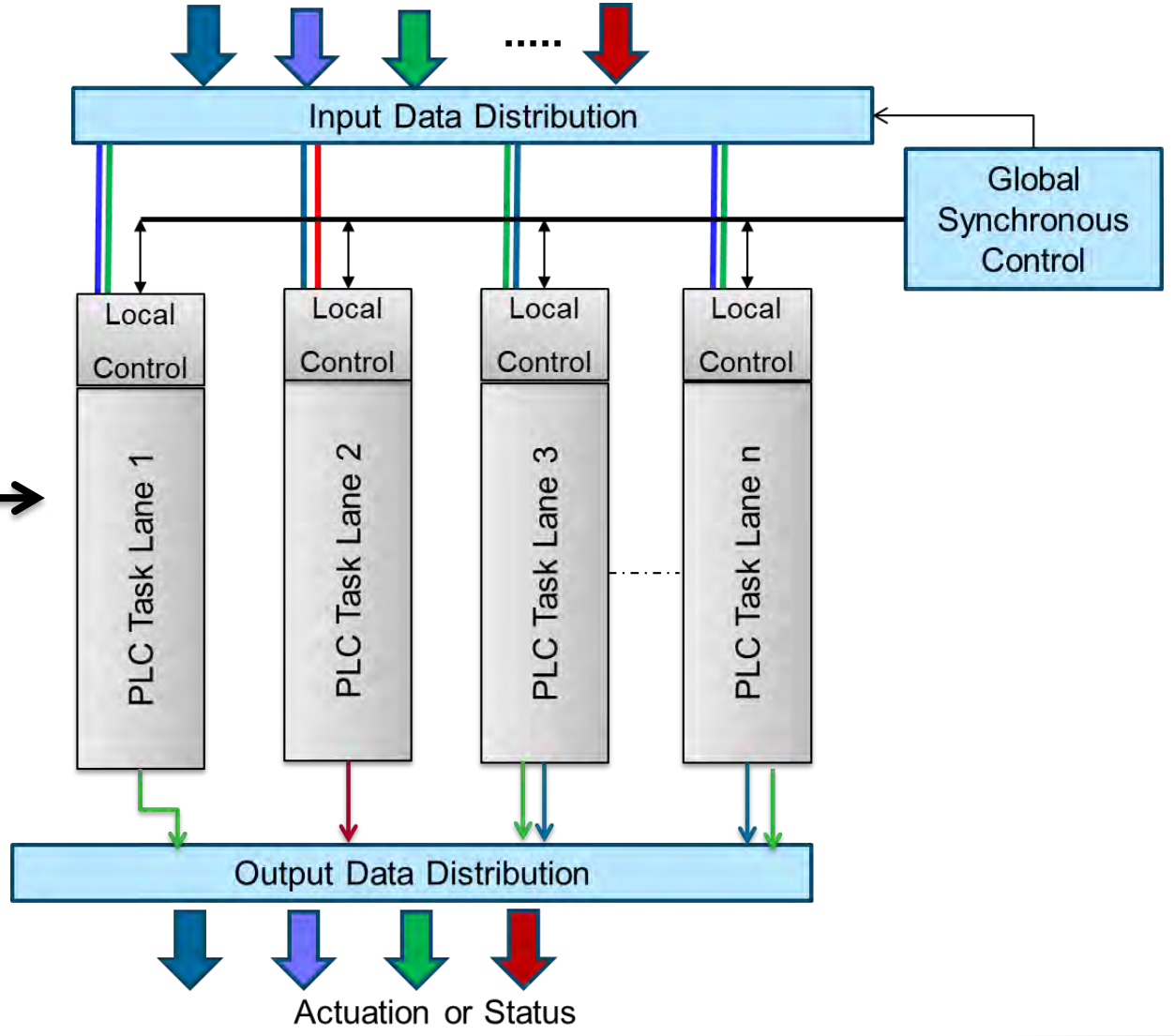
High Level Architecture Model of SymPLe

MATLAB SIMULINK

SymPLe Pre-verified FB Libraries

HDL CODER





VCU College of Engineering

EPRI ELECTRIC POWER RESEARCH INSTITUTE

SymPLe Function Blocks

Most I&C applications in NPP are not computationally challenging

Elementary FBs

Instruction	Description
AND, OR, NOT, XOR, NAND, NOR	Logical Operators
AND, OR, NOT, XOR, NAND, NOR	Bitwise Logical Operators
MAX, MIN, MUX	Selection Operators
GT, GE, EQ, LT, LE, NE	Comparison Operators
ADD, SUB, MUL, DIV	Arithmetic Operators
SLL, SLR	Bit-shift Operators
MOVE	System Operators

Built-up FBs

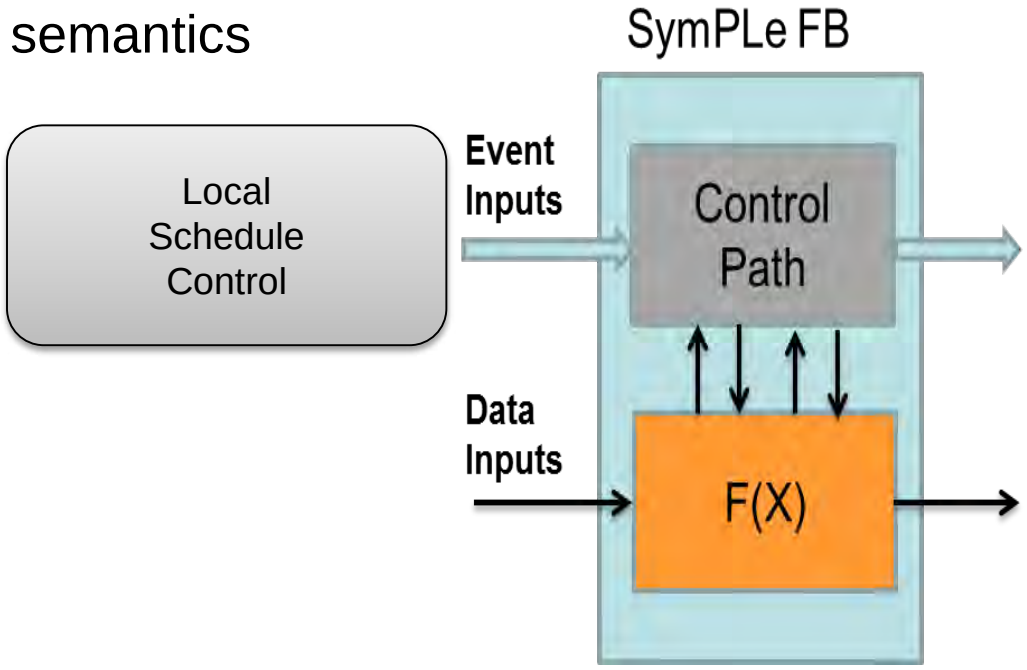
Instruction	Description
PID	Control operator
TON, TOF	Timer operator
FXTOI, ITOFX	Data conversion operator
AVOTE	Analog voting
BVOTE	Digital voting
MEM	Memory operator
LOG	Logging operator

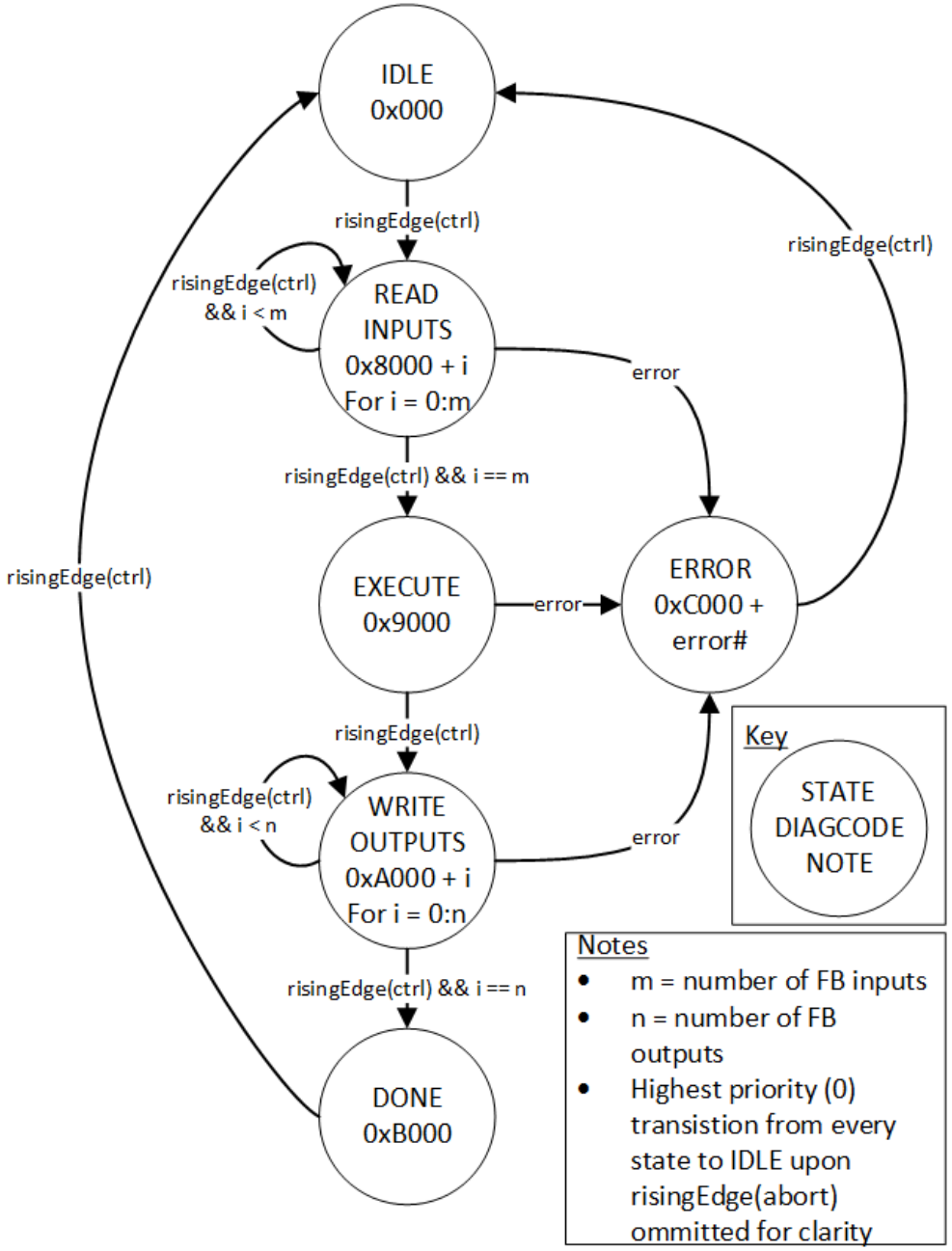
All Function blocks V1.0 were formally proven.

Function Block Architecture

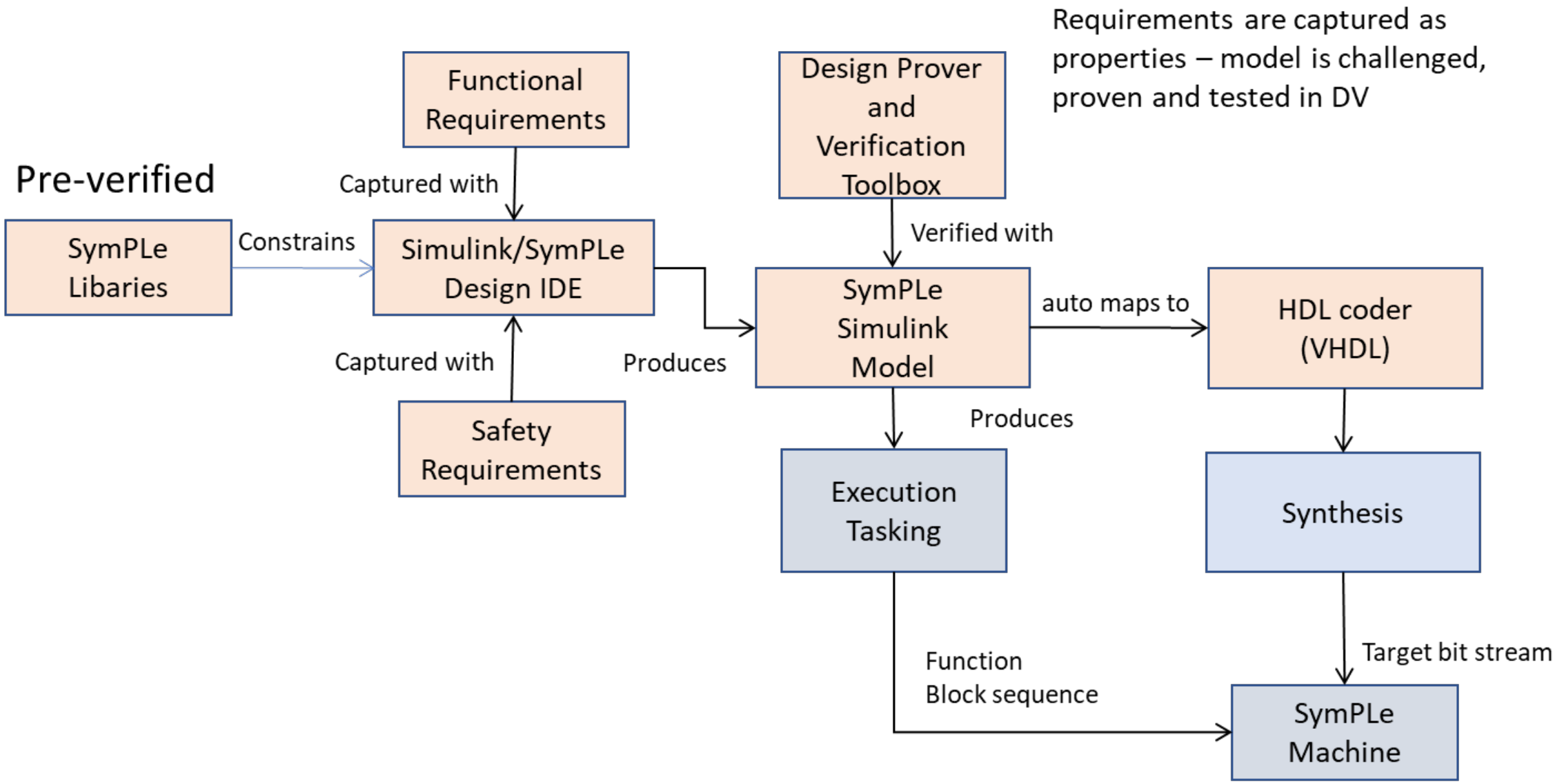
Common architecture for all SymPLe function blocks

- Inspired by IEC 61499
- Deterministic, synchronous behavior.
- Separation of control and dataflow with clear and defined interconnections
- Formal semantics

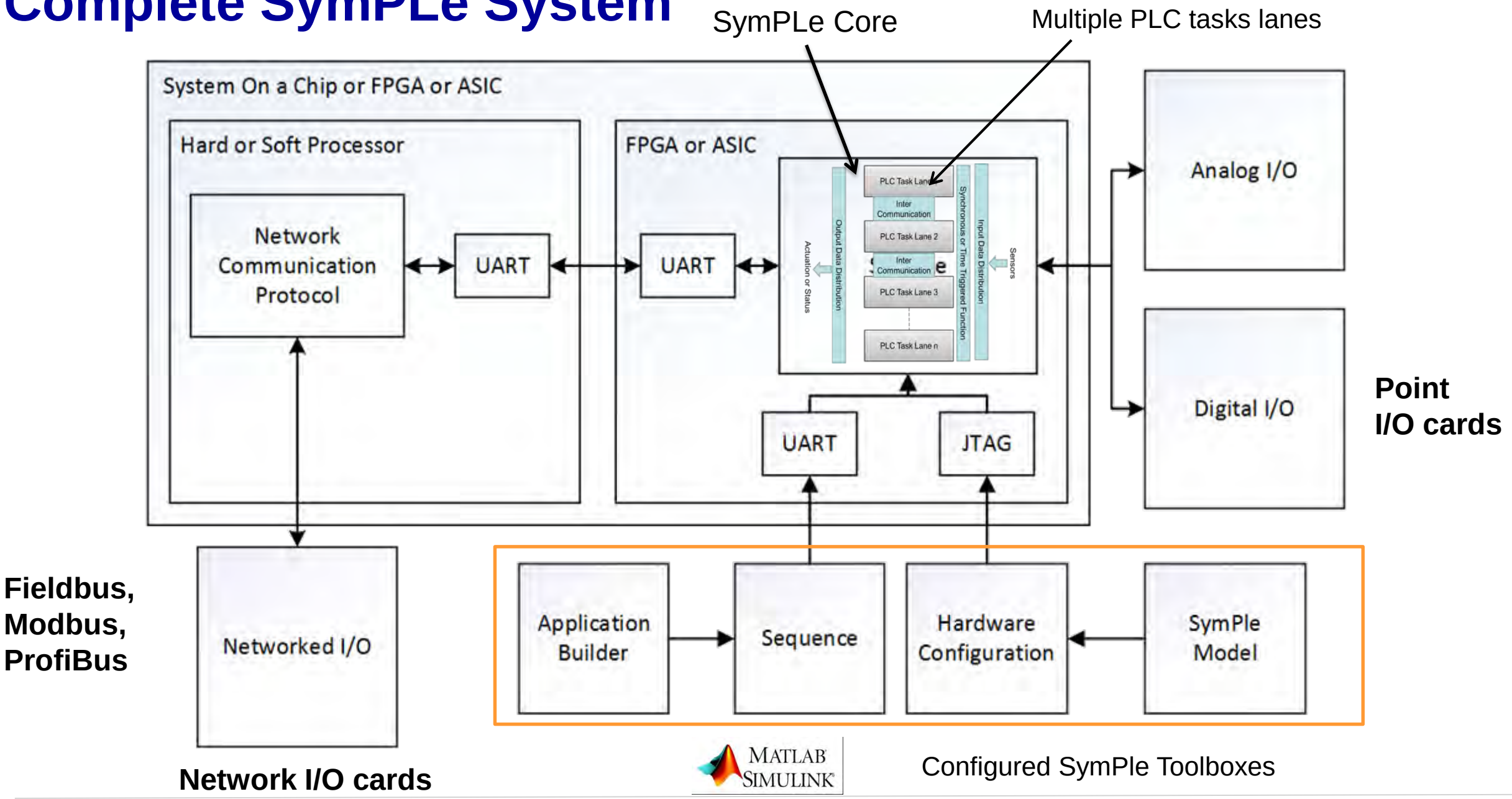




SymPLe Application Capture Workflow



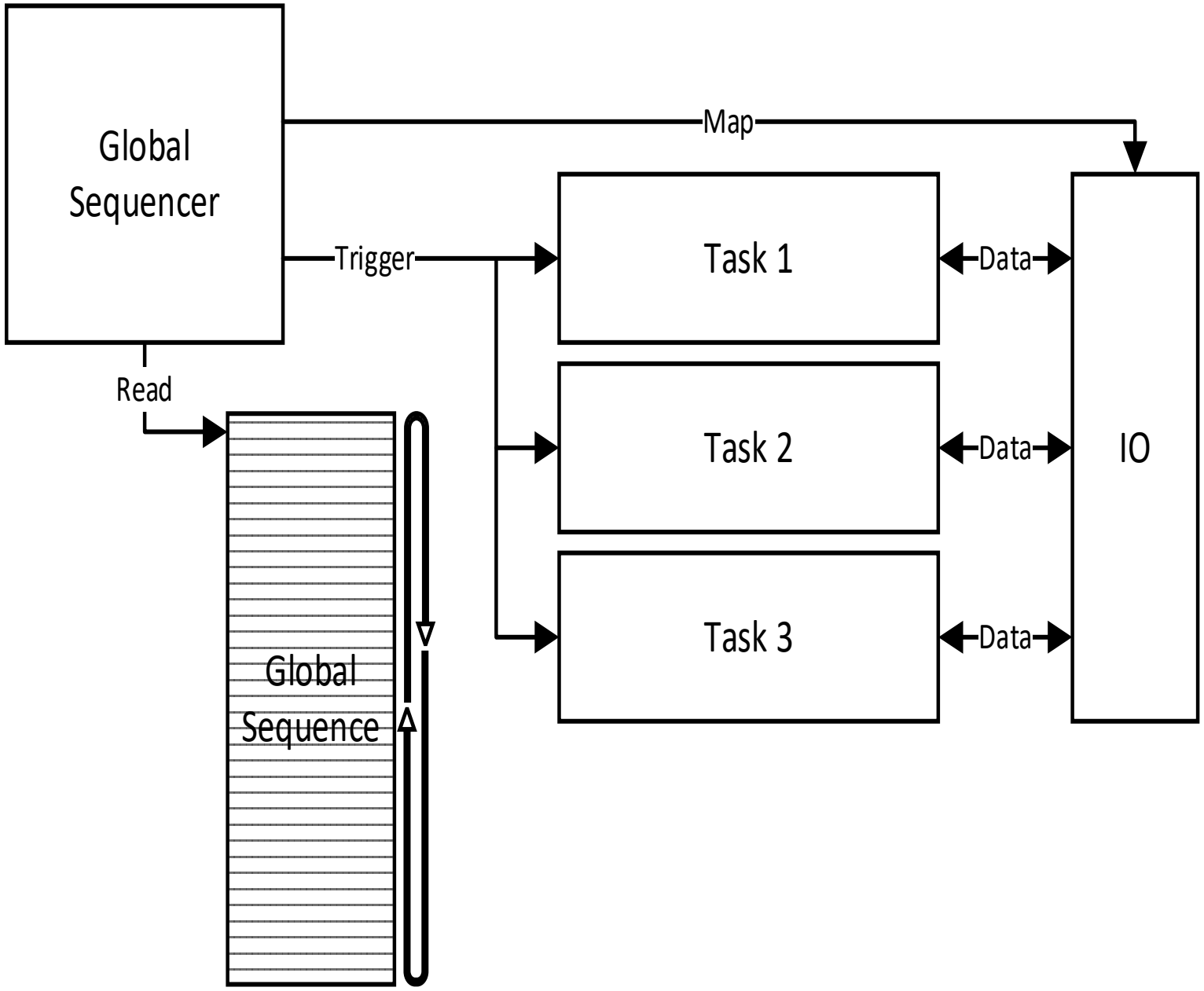
Complete SymPLe System



Global Operation

Execution Cycle

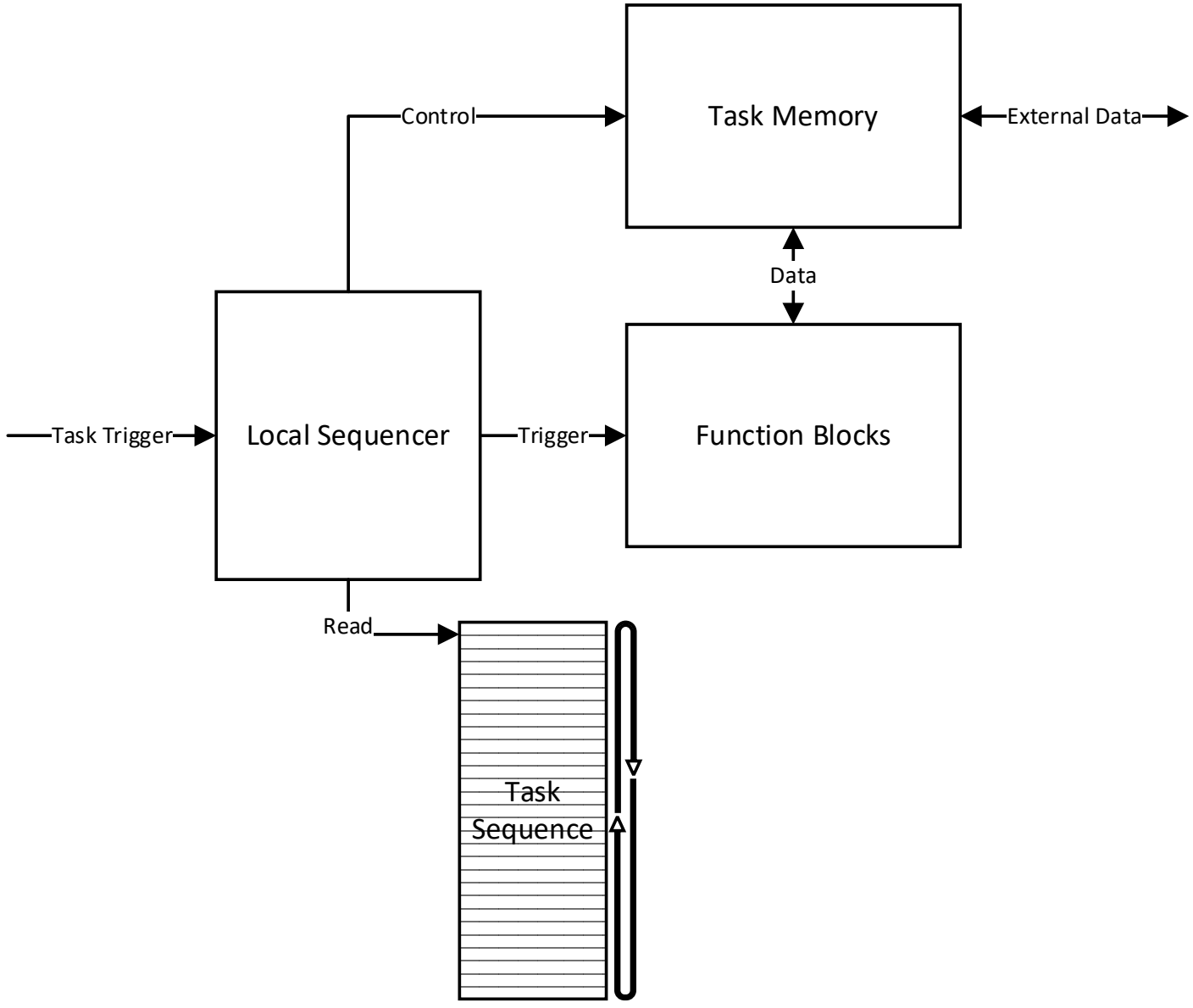
1. Map inputs to task data.
2. Trigger task execution.
3. Map task data to outputs.



Task Operation

Execution Cycle

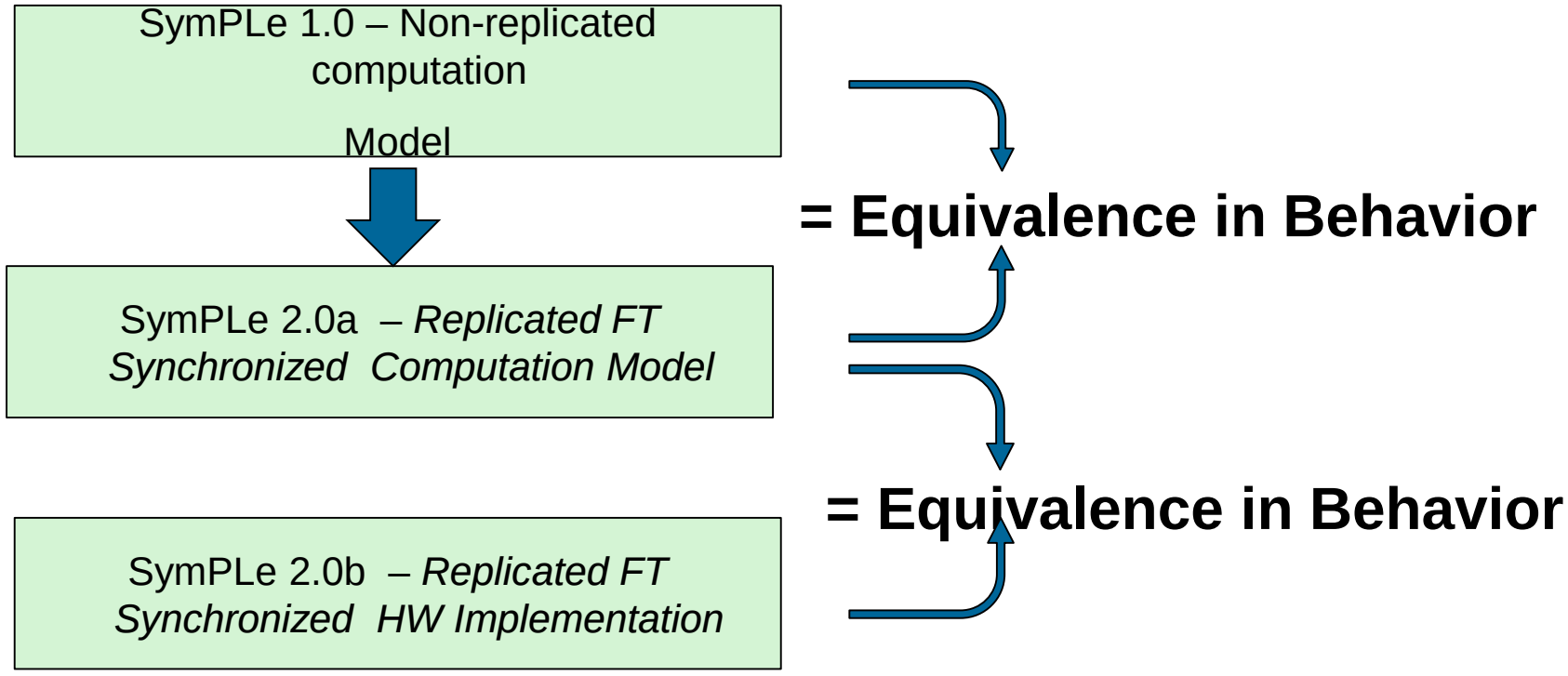
1. Triggered by Global Sequencer.
2. Trigger function blocks sequentially until end of task sequence.
3. Store result from each function block in task memory.



SymPLe 2.0 – A Reliable Computing Platform

A very general way to express fault tolerant processes in a system is through *transformational methods.* When something is transformed it has all of it's old properties, but it takes on new properties as well.

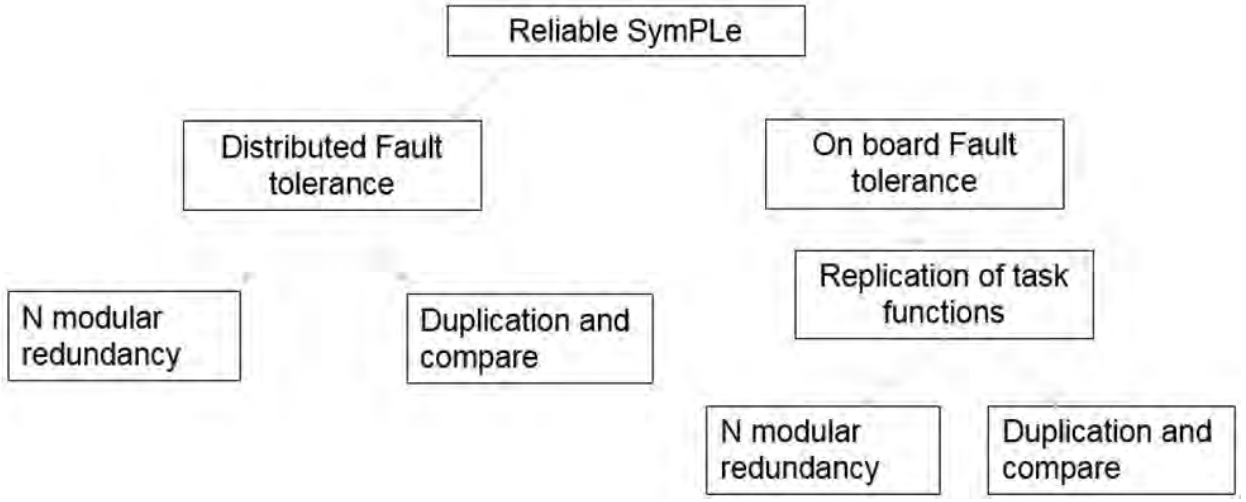
- In the design of SymPLe 2.0 we specified the FT design as a *transformation process* from a non-replicated computation machine to replicated fault tolerant machine.
- SymPLe 2.0 has all of the operational semantics of SymPLe 1.0 , but it has new fault tolerance properties.



SymPLe 2.0 Fault Tolerance Features (in Progress)

Redundancy Employed – Space (HW), Time, and Information

- Lowest level of FT is the Function Block. Each Function Block type is fault-aware.
- Task lanes can be organized into synchronous replica groups – duplex, TMR, QMR, self-checking pairs.
- Multiple SymPLe machines across different FPGAs.
- Single source inputs are distributed via interactive consistency network
- Goal is to achieve fail-stop semantics for function blocks and task lanes.
- Design Faults:
  - use of runtime verification checkers. Encode temporal logic (PSL) assertions (safety properties) as checking automata in VHDL or Verilog.
  - Design time formal verification of critical functions
  - Constrained behavior of architecture helps with formal verification



IDF: Place and route for enhanced reliability

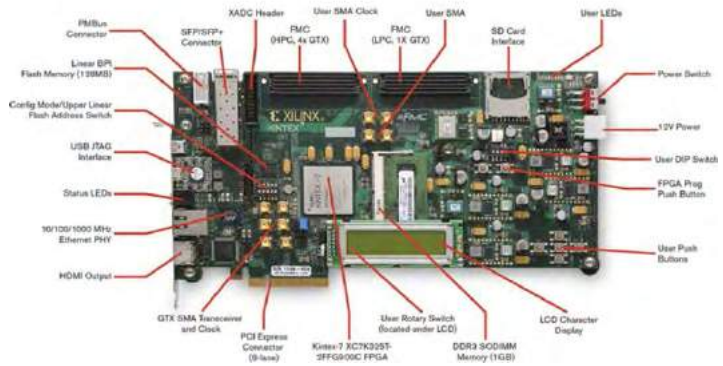
- To use SymPLe in safety critical applications or applications where high availability/reliability are needed – fault tolerance is needed..
- One of the important design tools is the Xilinx Isolation Design Flow (IDF).
- IDF helps define fault containment regions in FPGA during place and route.
- IDF is IEC 61508 certified and verifies that the isolated regions meets Safety Integrity Level(SIL) 3 safety requirements.
- The isolated regions are defined by drawing regions called PBlocks around them.
- Constraints were defined to isolate the PBlocks.

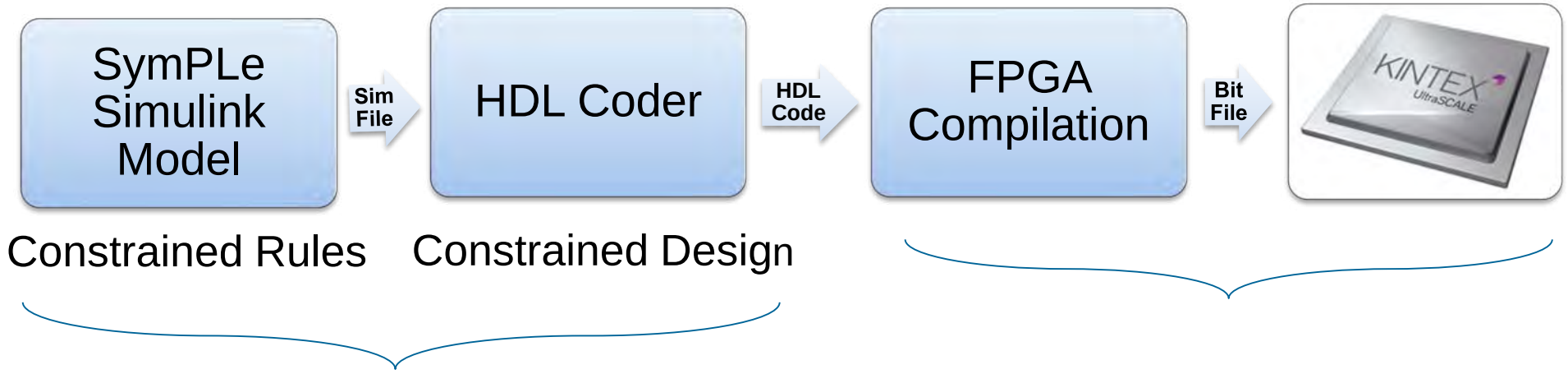
Implemented design: Dual modular redundancy(DMR)

Two Task Lanes as a DMR

SymPLe Initial Implementation – Xilinix FPGA

- Initial design was implemented on Xilinx KINTEX 7





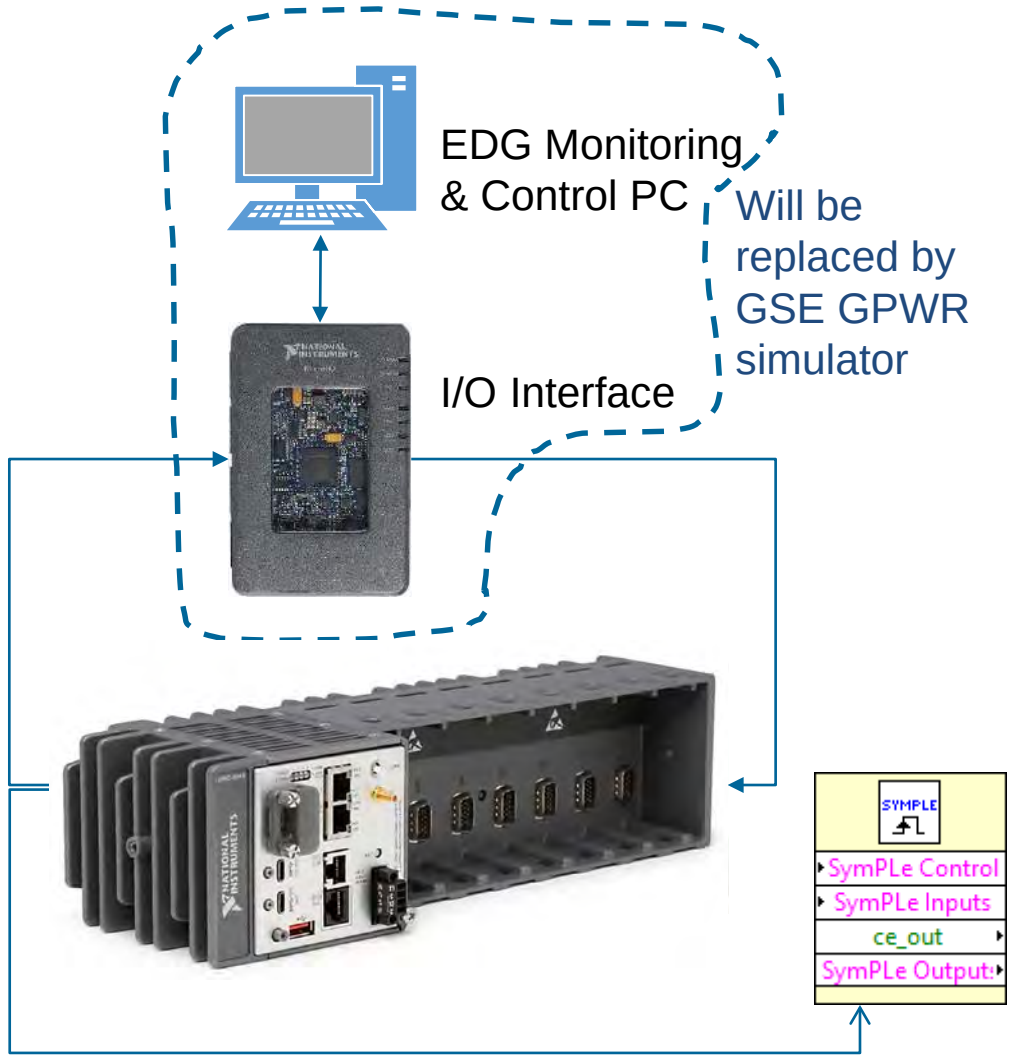
SymPLe Industrial Implementation

- Moving Beyond Xilinx Kintex 7 development board – we chose to implement SymPLe on a industrial controller to explore issues with portability, I/O interfacing, behavior in a real world platform.
- We chose the National Instruments cRIO Industrial controller with modular I/O capability.

➢Built-in FPGA to enable the implementation of SymPLe architecture using HDL code.
➢Flexibility in I/O programming to interface with SymPLe.

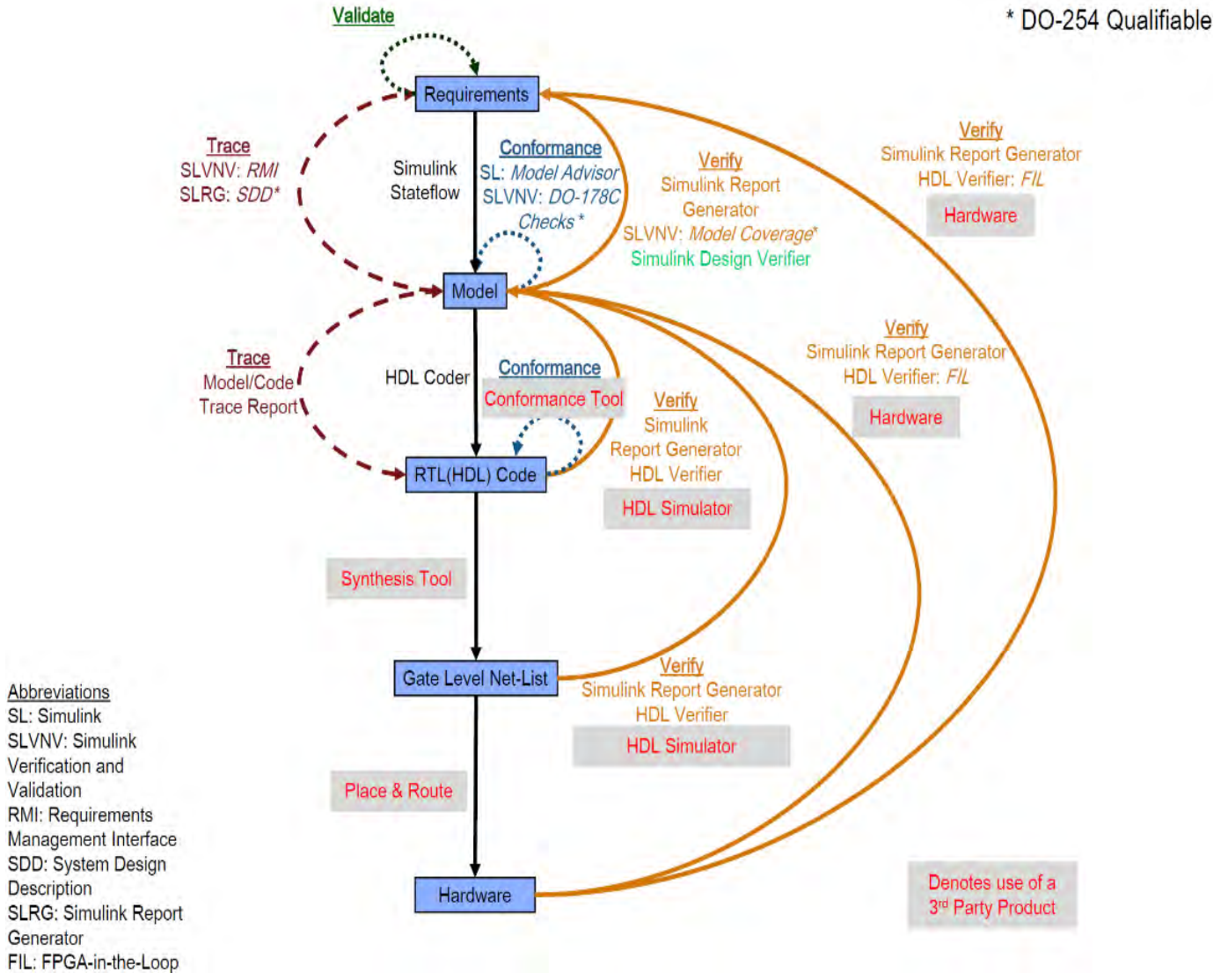
VCU Hardware In the Loop (HIL) Implementation

- HIL was setup for the Emergency Diesel Generator (EDG) application.
- A simple user interface was built using National Instruments LabVIEW software to manipulate EDG inputs and monitor the outputs.
- SymPLe architecture, implementing EDG logic, was downloaded to National Instruments cRIO controller FPGA.
- Using modular design, a SymPLe subsystem block was developed in LabVIEW, where it could be reused in other test programs.



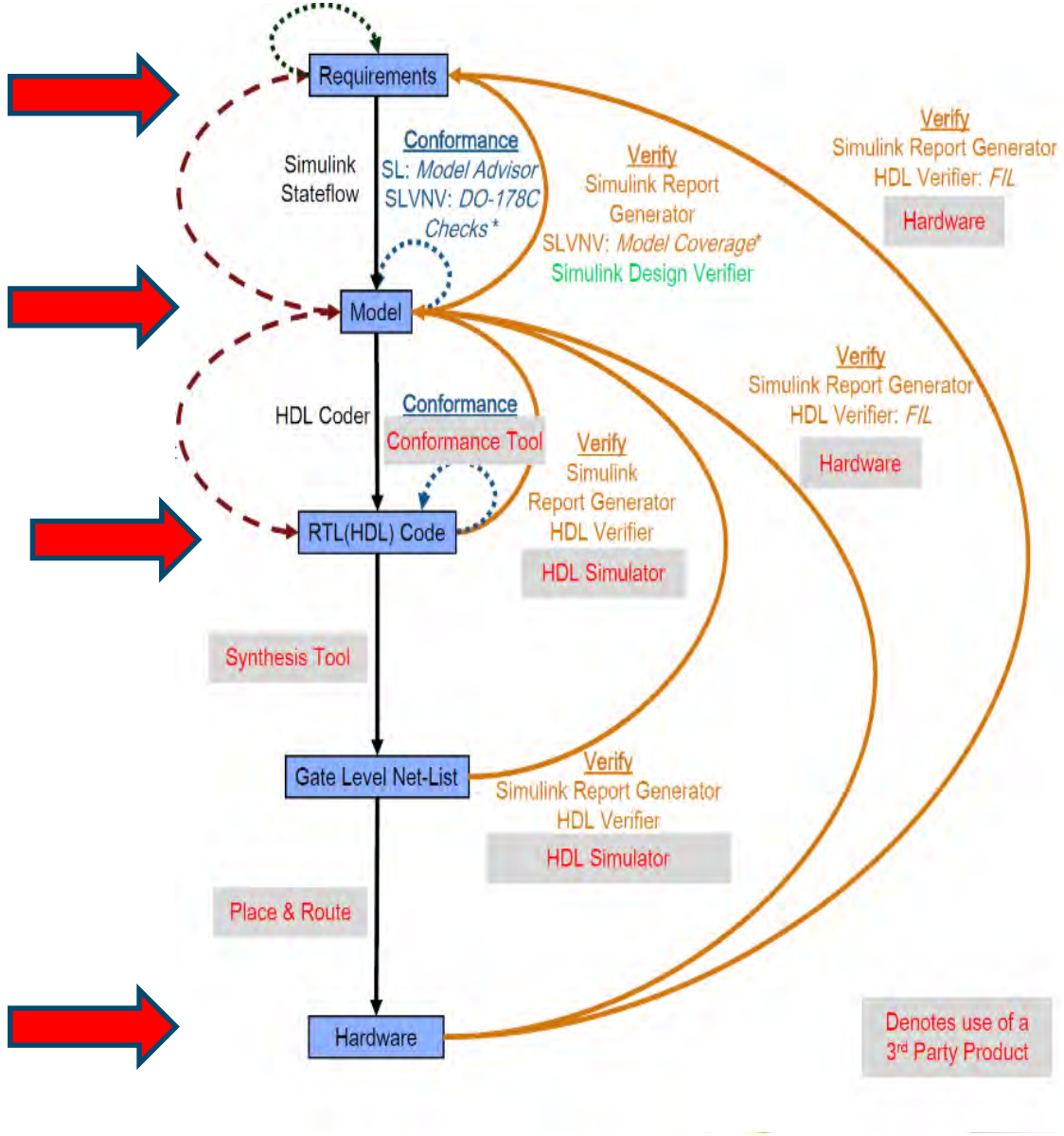
SymPLe Development and Design Assurance Workflow

- Currently following IEC 61508 SIL 3/4
- Verification at each step from requirements to implementation
- Math works Tools used throughout V&V workflow
  - **Prevents gaps in V&V from oversight or misinterpretation**
- Tools IEC 61508 certified

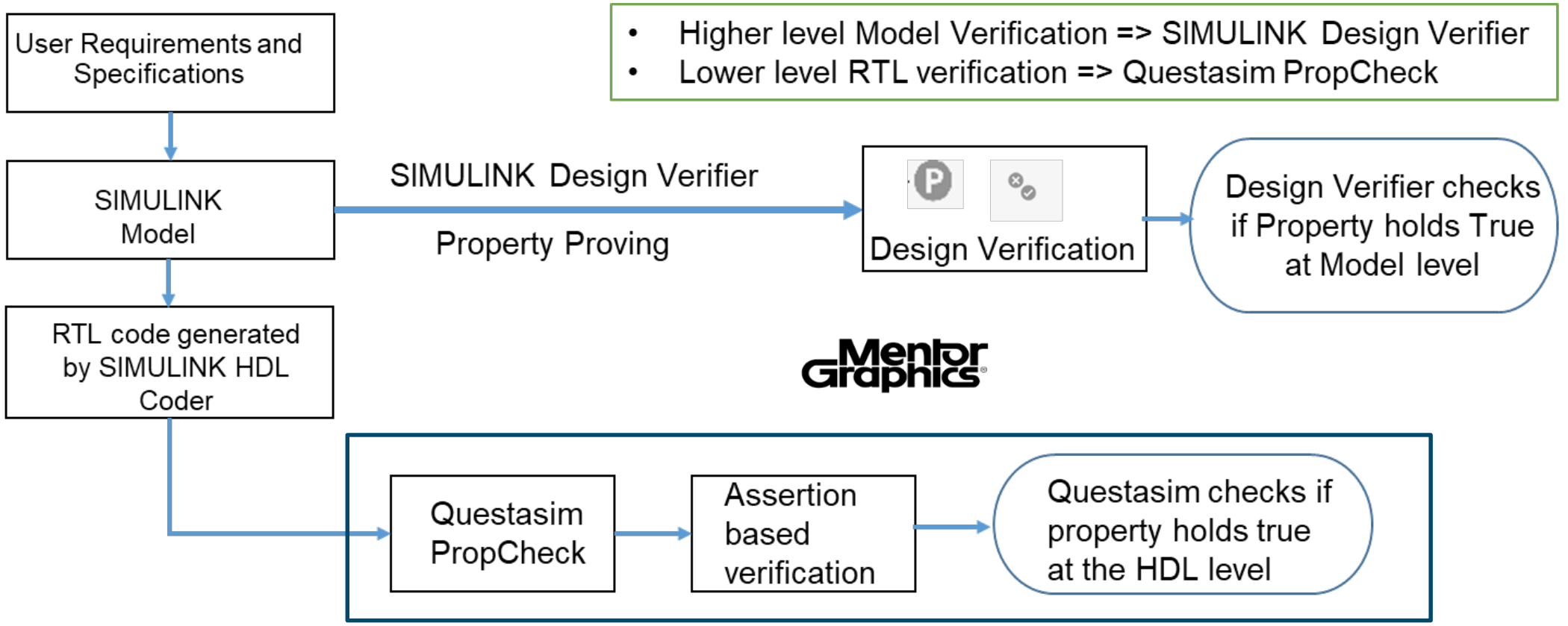


SymPLe V&V - Test Case Generation

- Why?: Formal verification needs to be balanced with test cases - generate corner cases.
- Bridges gaps in formal methods
- Develop specific test cases for:
  - Execution sequences
  - Edge cases
    - Errors commonly found here
  - Mid range cases
    - Verify normal operation



End-to-End Property Verification using Simulink DV and MENTOR GRAPHICS Questasim



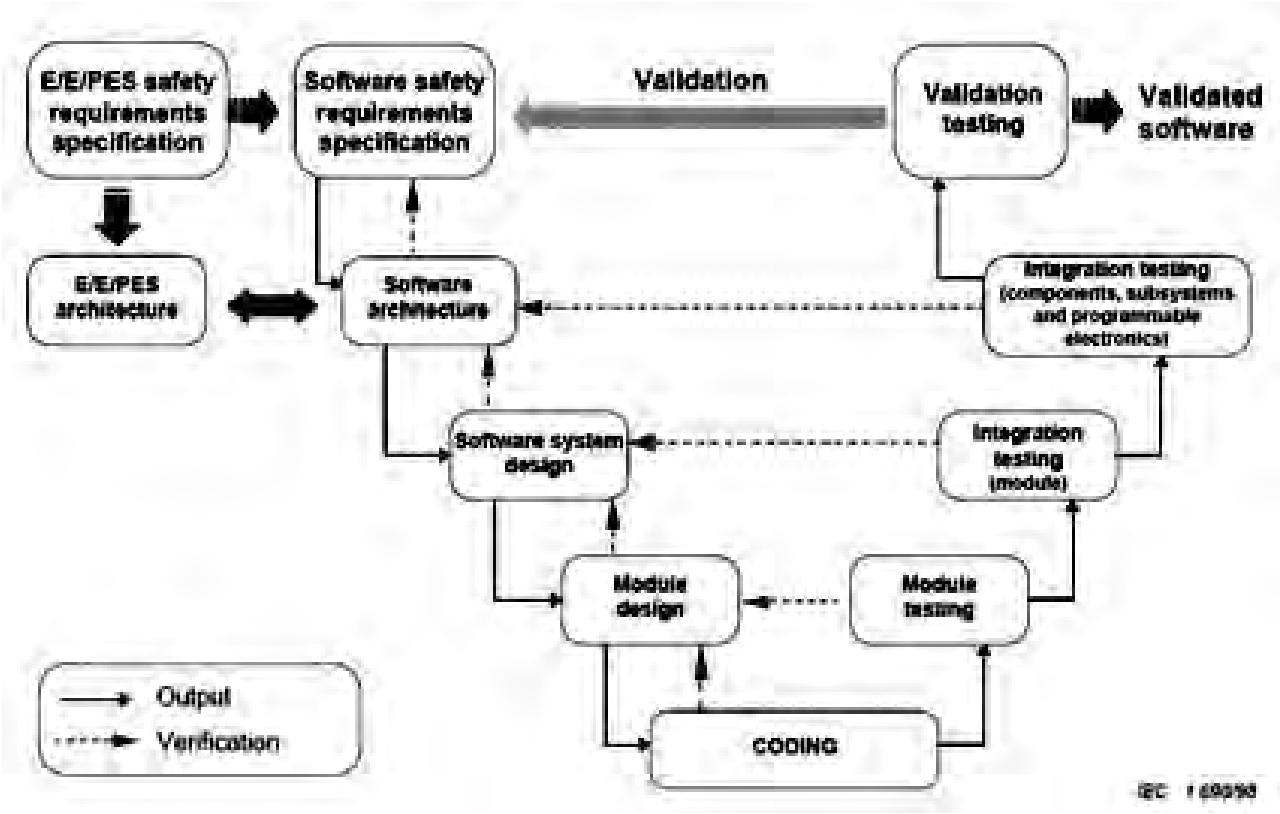
24

Verifies that actual SymPLe HW has inherited the proven property of models

Commercial Grade Dedication Exercise

To Stress the SymPLe concept – It is undergoing a limited and focused Commercial Grade Dedication(CGD) based on IEC 61508

- CGD will focus mostly section 3 of IEC 61508-Software Process
  - SymPLe was designed, developed, verified, and implemented with a Model-based design process
  - Our functional models of SymPLe are “system” oriented with no relation to HW or SW, but our process ultimately produces HDL code
  - Paragon Tech will perform the CGD
- The Section 3 “V” model along with the corresponding annexes are closest to existing CGD activities.
- To our knowledge this is one of the first “model based design” efforts with respect to a Nuclear I&C to be CGD reviewed

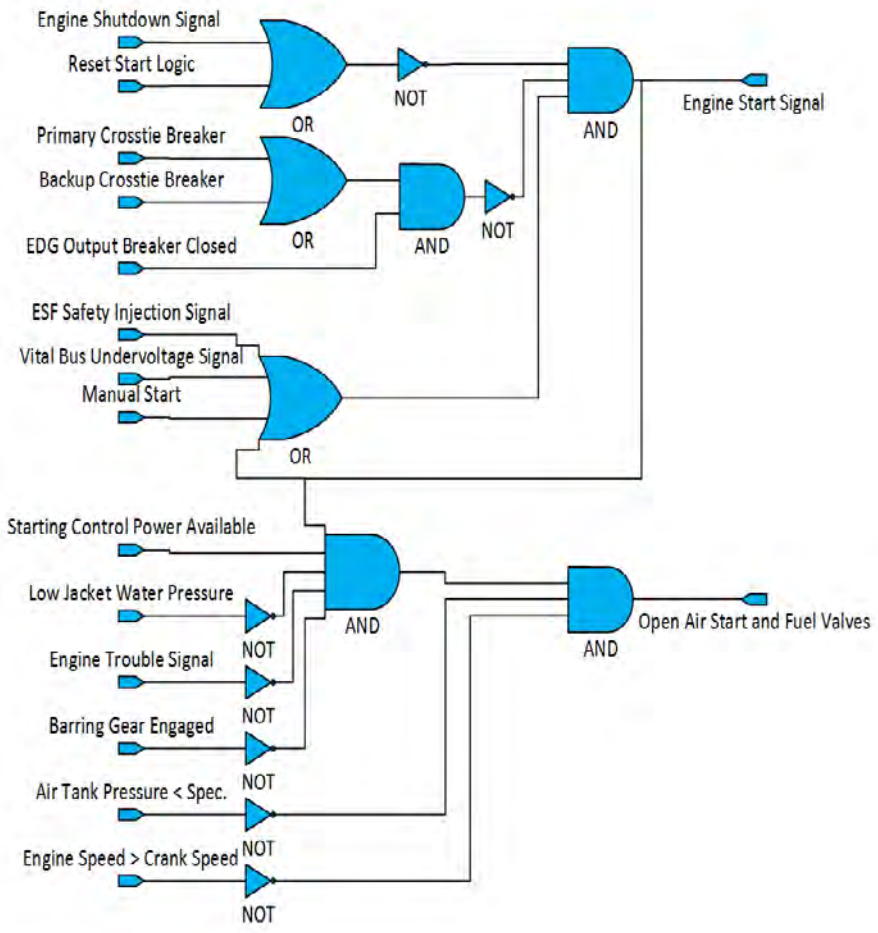


VCU College of Engineering

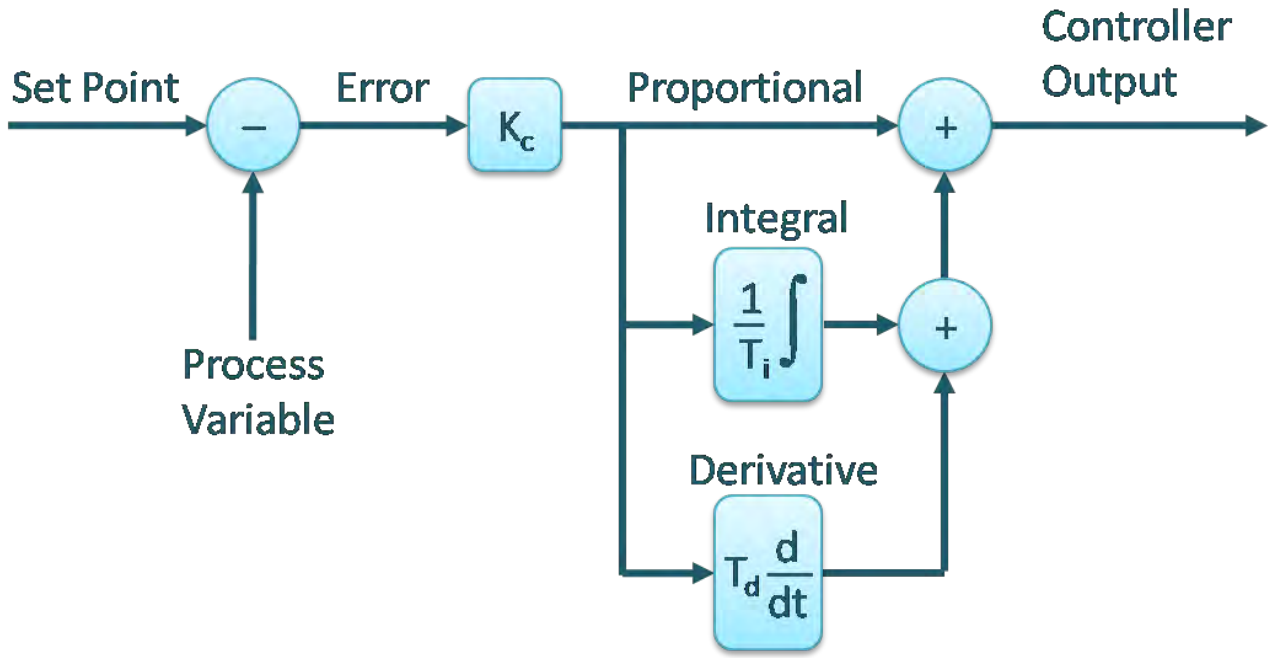
EPRI ELECTRIC POWER RESEARCH INSTITUTE

Applications Ported to SymPLe to Date

Emergency Diesel Generator Startup Controller



Proportional, Integral, and Differential Controller



VCU College of Engineering

EPRI ELECTRIC POWER RESEARCH INSTITUTE

2018 Summary and Take Away

- Have designed and realized a practical verifiable PLC Instantiable architecture that addresses CGD and SCCF via:
  - Constrained architecture operations
  - Formal deterministic FB semantics
  - Extensive use of model based design and analysis
  - Complementary Formal verification and Testing
- Have tested Hardware in the Loop in the lab, working toward HIL for GSE GPWR.
- Have Demonstrated SymPLe concept with Basic Emergency Diesel Generator (EDG) start controller, PID loops, etc…
- Developed design, models, and beta versions for ultra safe versions of SymPLe (Safety Integrity Level 3/4)
- Presented two papers at ANS NPIC 2017, more papers in preparation for IEEE Journal on Control Systems Technology.

2018/2019 planned activities

- Conduct IEC 61508 (SIL 3)Commercial Grade Dedication (CGD) Exercise with Paragon Energy Solutions Nuclear and EPRI.
- Use Model Based Tools to provide CGD data to Paragon for CGD review.
- VCU purchased IEC 61508 tools from MathWorks
- Continue refining architecture, complete critical review of Property Proving with MathWorks experts
- Complete design and testing of I/O architecture
- Select a new “more complex” application to port and test on SymPLe
  - Feedwater
- More HW testing, and more HW testing….
- Document final project findings and results

Discussion

Together…Shaping the Future of Electricity