

A Methodology for Designing Scalable SDR Systems

New England Software Defined Radio Workshop '13
Worcester Polytechnic Institute, May 17, 2013

<http://sdr-boston.org>

Erich Whitney

**Group Leader, Lead Digital/Micro Hardware Engineer
The MITRE Corporation**

Contributors:

**Hieu Nguyen, Karl Wagner, Joseph Chapman, Chuck Mazzola
and Robert Reynolds
The MITRE Corporation**

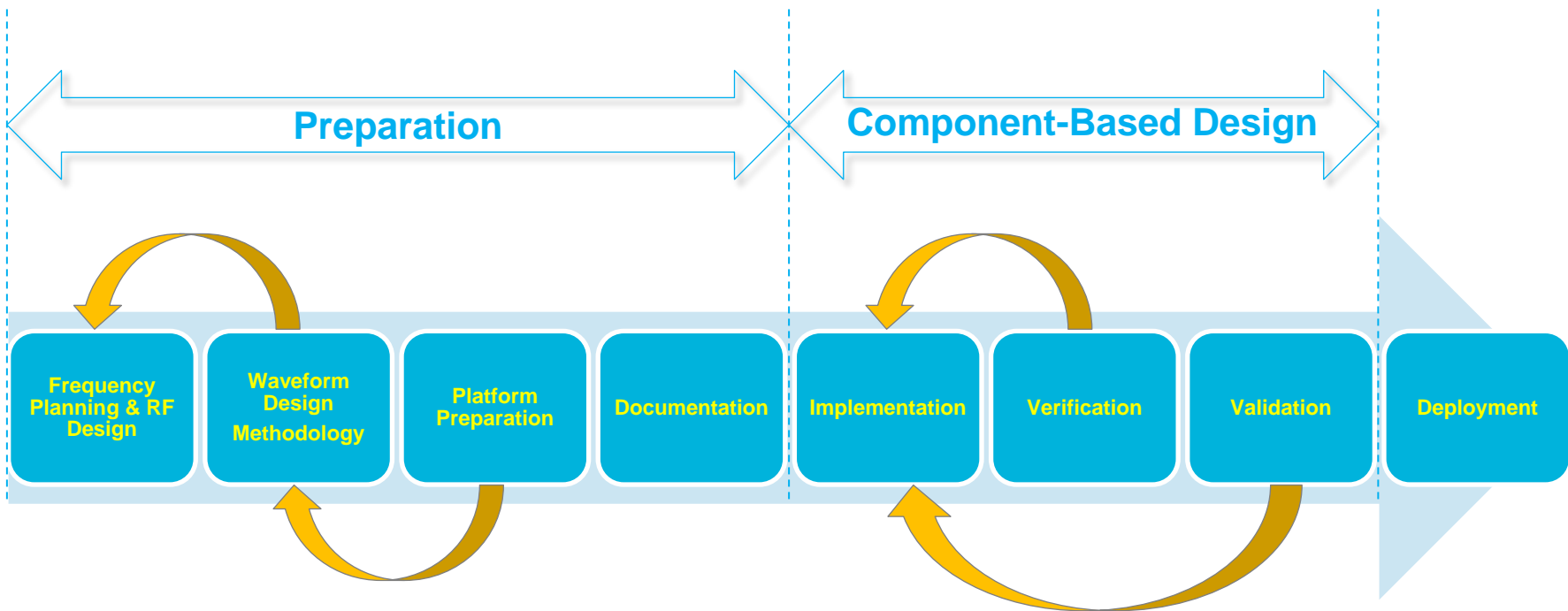
Abstract

MITRE's Software Defined Radio (SDR) development encompasses a wide variety of applications across communications, sensors, and navigation. This presentation discusses MITRE's methodology used to develop large, scalable, Field Programmable Gate Array (FPGA) based SDR systems. The process starts with up-front radio frequency (RF) planning, then explores analog and digital signal processing tradeoffs, and completes with component-based waveform design, implementation, verification, validation, and deployment. This methodology, developed through many years of experience, has resulted in multiple SDR systems successfully deployed to sponsor operations, hand-off to military contractors, and ground-breaking field experiments.

Outline

- **Introduction**
- **Preparation & Planning**
 - Frequency Planning and RF Design
 - Waveform Design Methodology
 - Platform Preparation
 - Documentation
- **Component-Based Design**
 - Waveform Integration
 - Embedded Software Considerations
 - Functional Verification
 - Performance Validation
- **Standard Release Process**
- **Conclusions**

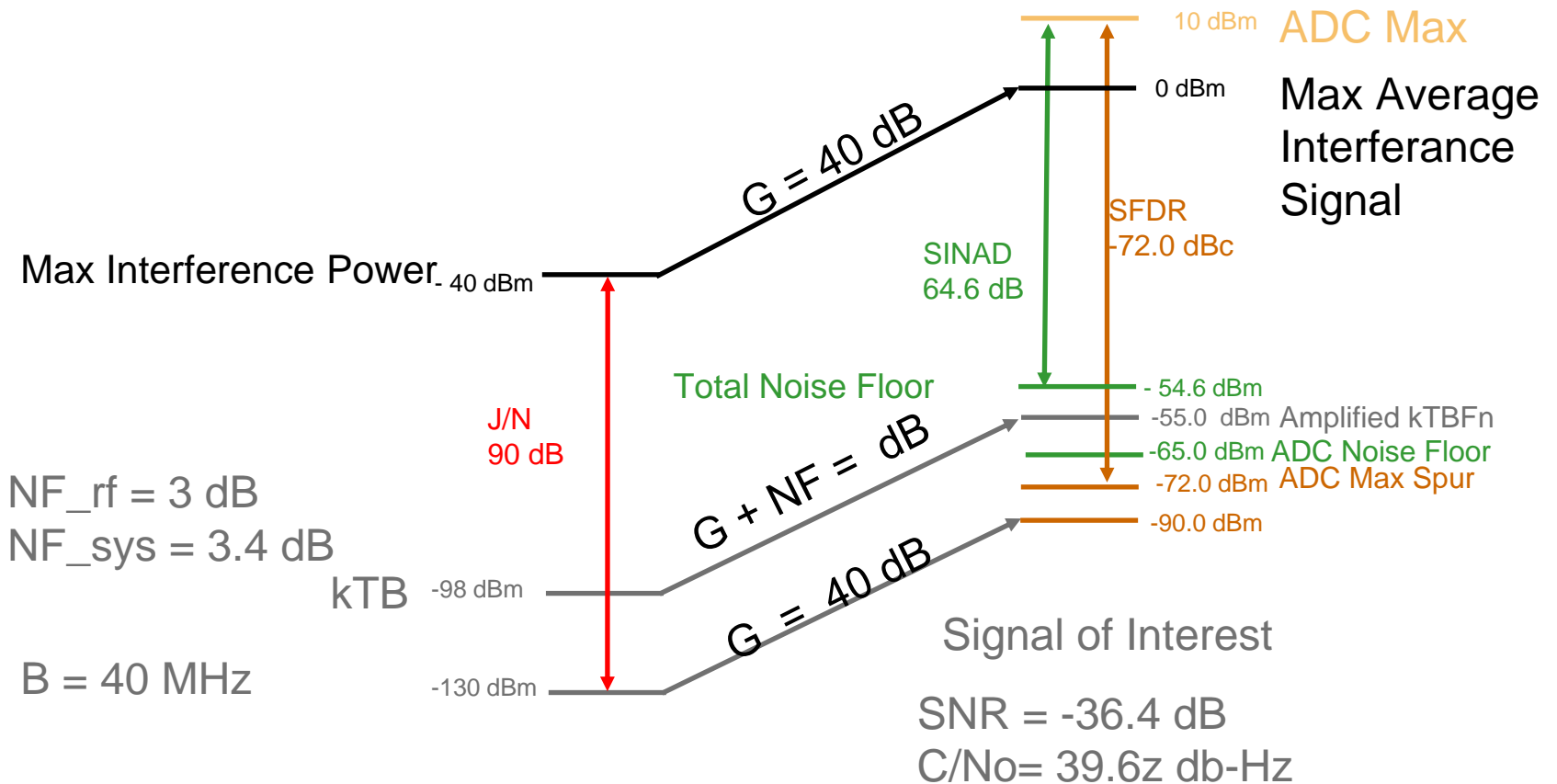
SDR System Design Process



Frequency Planning & RF Design

- **System Requirements**
 - Modulation
 - Data Rate
 - Link Parameters
 - Transmit Power
 - Signal Propagation Characteristics
- **RF System Cascade Analysis**
 - NF, IP3, Gain, etc.
- **RF to Bits**
 - ADC Sampling and Dynamic Performance
 - Sampling Clock Jitter
- **RF/Analog Circuit Design**
 - Replace System Blocks with component design
 - Filters, Amplifiers, Mixers, IQ Demodulators, ADC
 - ADC Interface
- **Need to know your environment!**
 - Expected Interferers
 - Channel Characteristics

Example: Receiver System Level Plan



System Tools Used for Examining Trade-Offs

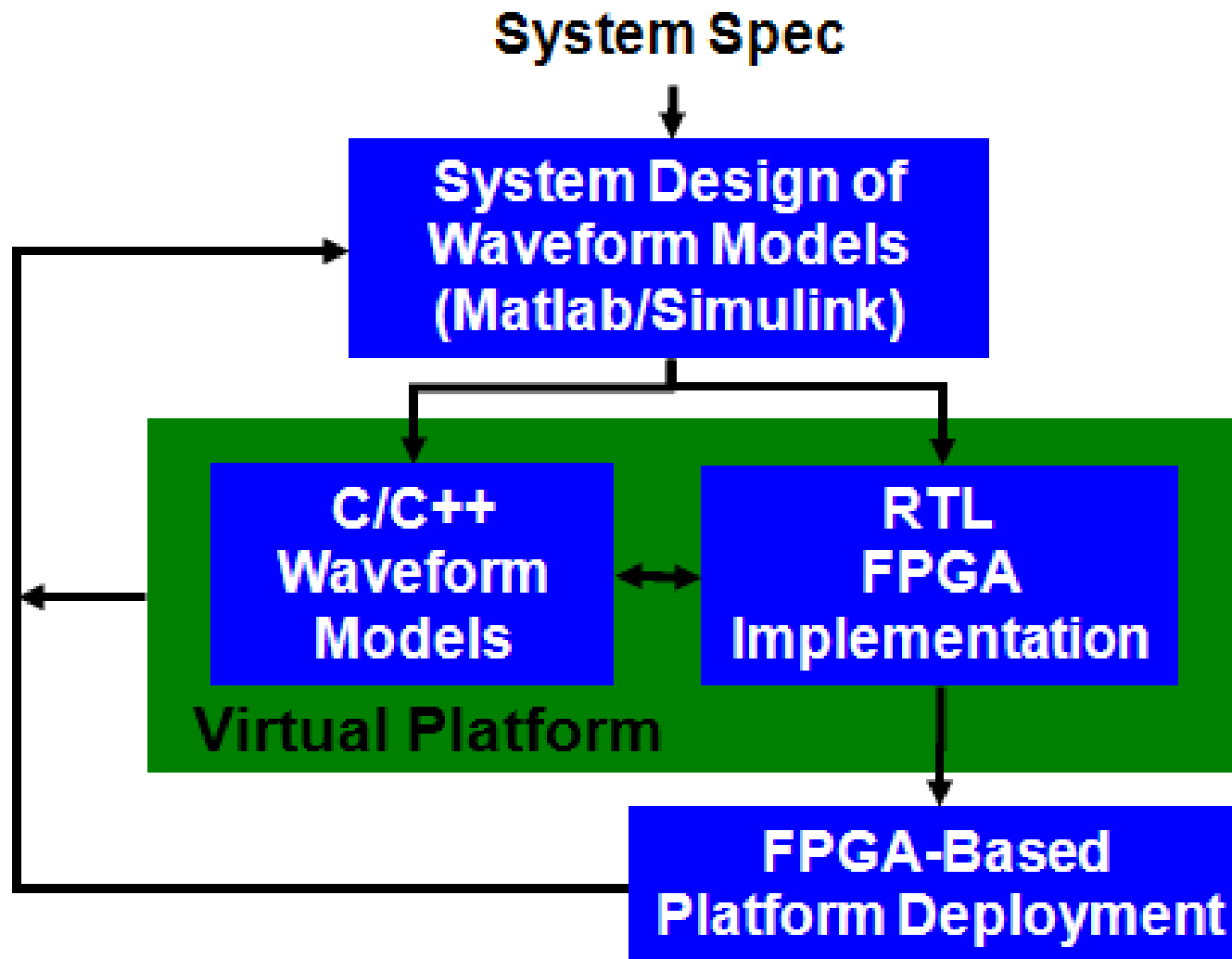


- End to End System Level Simulation
 - Define System Level Blocks
 - Examine Tradeoffs
 - Incorporate Circuit for Blocks
 - Test critical subsystems/components before PCB design
 - Don't take the manufacturer's word on it!

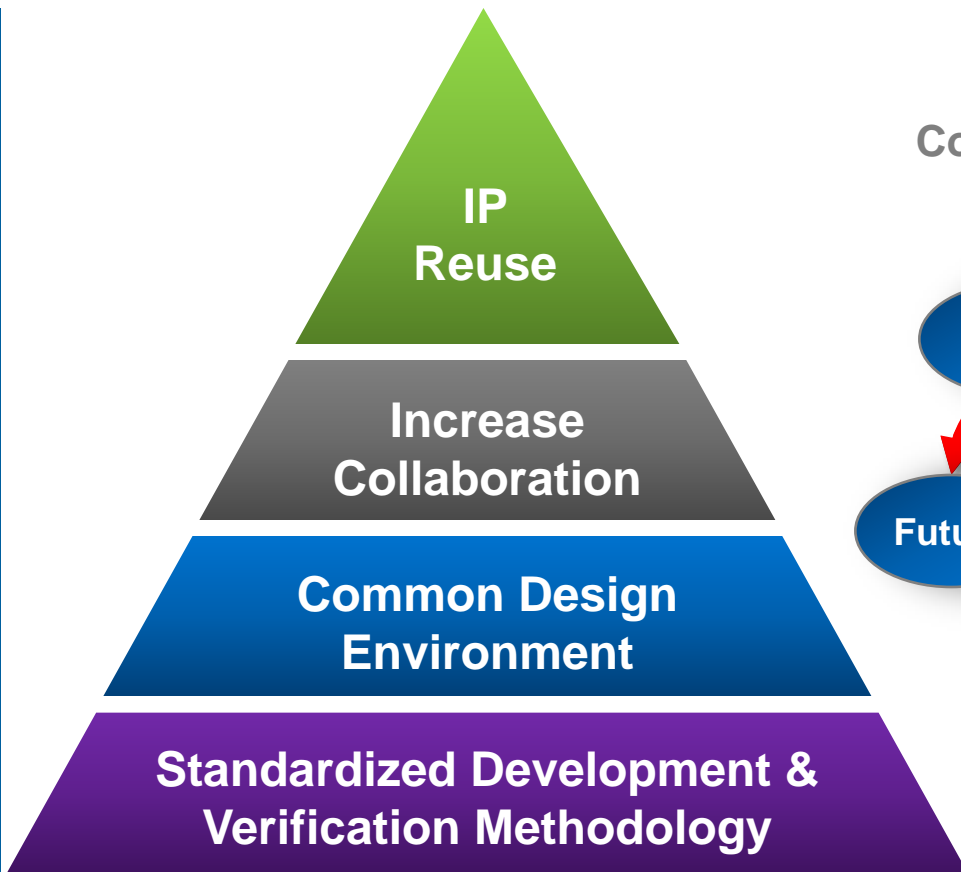
- System and circuit simulators
 - Time Domain
 - Frequency Domain
 - EM
- Math Packages



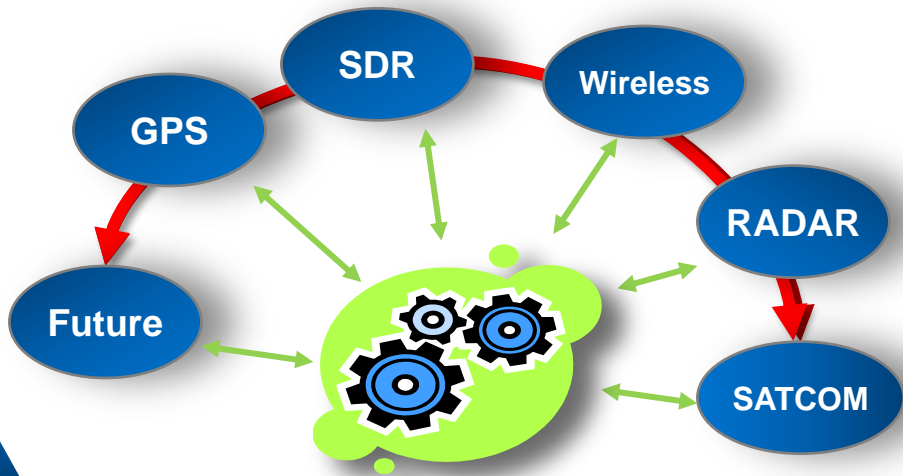
Waveform Design Methodology



IP Reuse is critical to our success



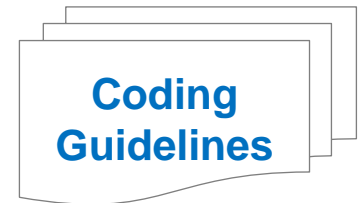
Component IP Reuse Across Projects.



Centralized IP Repository Provides Centralized Control & Flexibility for Change

Industry Best Practices

- Code Development Process
- RTL and C++ Coding Guidelines
- Distributed Version Control
- Online Code Reviews
- Bug Tracking
- Nightly Regressions



Common Interfaces

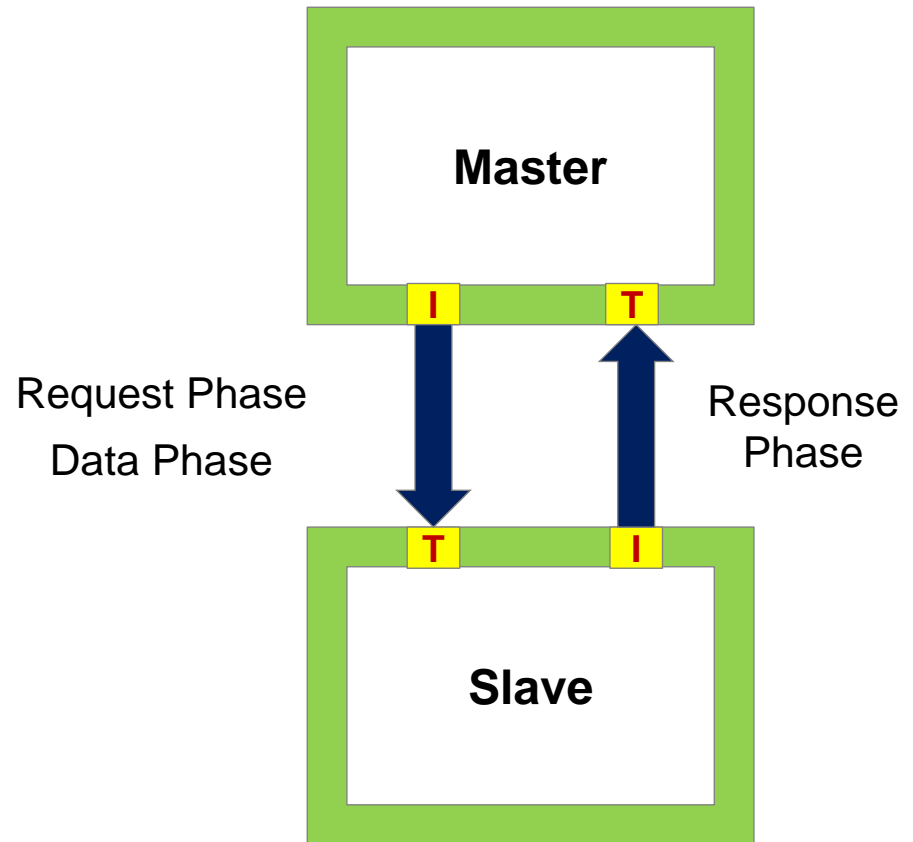
- Well defined interface at component boundary reduces ambiguity and increase porting efficiency
 - Take advantage of open non-proprietary interface defined by industry
 - IP porter only has to learn and understand one set of interfaces
 - Minimize number of gaskets needed during interfacing process



- Allow waveform porter to treat every component as a *black box*
 - Document all the component configuration parameters
 - Enable developer to protect the underlying design implementation
- Common interface also enables more automation in design flow

Open Core Protocol-International Partnership (OCP-IP)

- <http://www.ocpip.org>
- Defines a Master/Slave, Initiator/Target relationship between components
- **Dataflow**
 - Master
 - Data
 - Cmd
 - Slave
 - CmdAccept, ThreadBusy
- **Memory**
 - Master
 - Address, Data
 - Cmd
 - Slave
 - Data
 - CmdAccept, ThreadBusy



Component Metadata

- **XML is a great way to capture IP metadata**

- Interface Ports
- IP Source Files
- Design Parameters

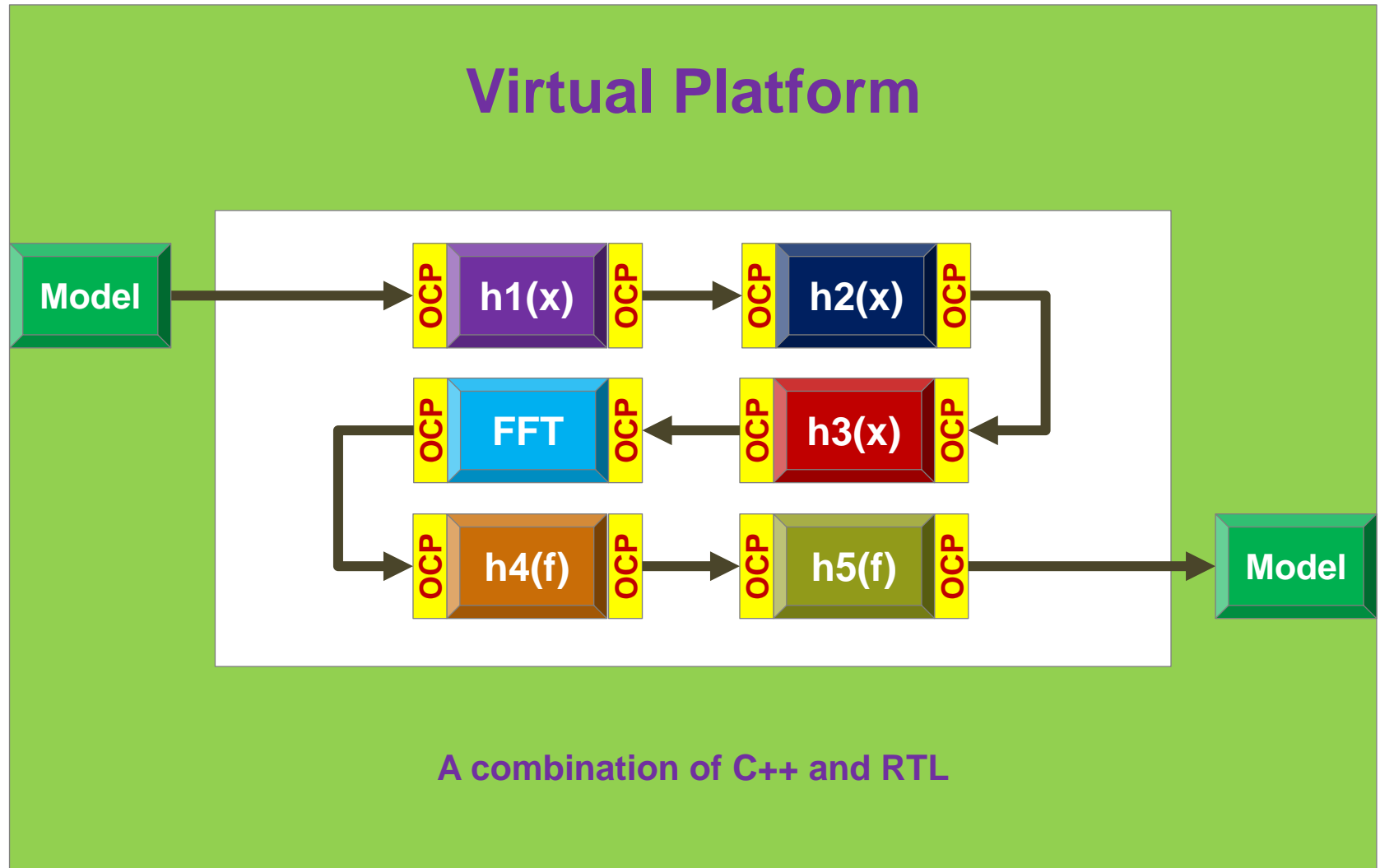


- **Open source and scripting support**

- **IP industry now embraces XML with IP-XACT**

- IEEE 1685 IP-XACT Standard, June 2010
- Accellera IP-XACT Schema Technical Committee, July 2010
- SPIRIT XML Schemas provide a mechanism to exchange IP across vendors and tools

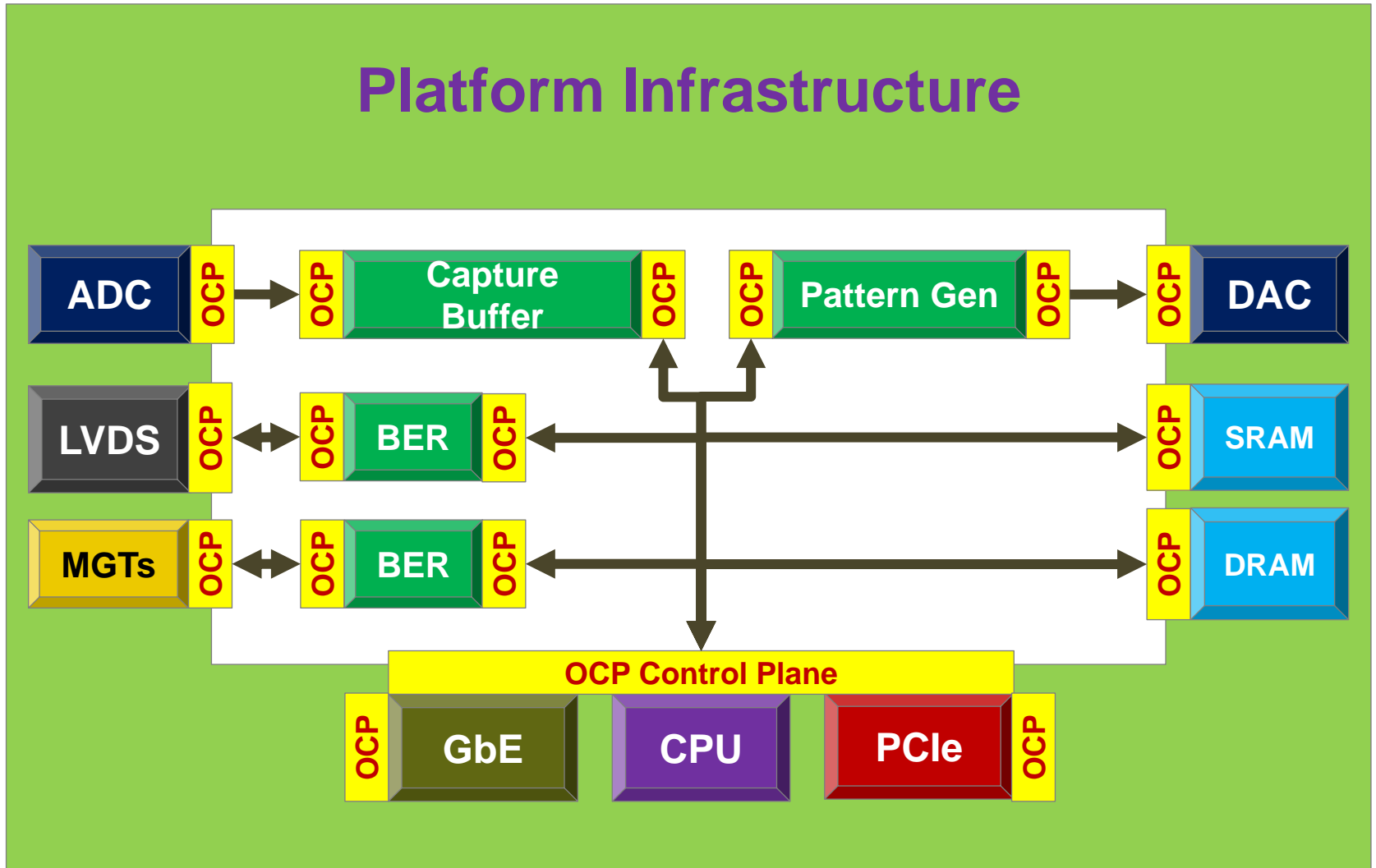
Waveform Design



Platform Preparation

- **Platform is the hardware support for the waveform**
- **All hardware interfaces adapted to OCP-IP**
- **Control plane is part of the platform design**
- **External memory interfaces**
- **High-speed serial data interfaces (chip-to-chip/board-to-board)**
- **ADC/DAC and other I/O interfaces**
- **Control plane infrastructure tied to a processor or processors**
- **Exception handling (i.e. interrupts and error handling)**
- **Debug interfaces (i.e. data capture buffers, read-back registers)**
- **Clock and reset architecture**
- **System bring-up support and hardware synchronization**
- **Chassis/Backplane Support and Configuration**

Platform Preparation

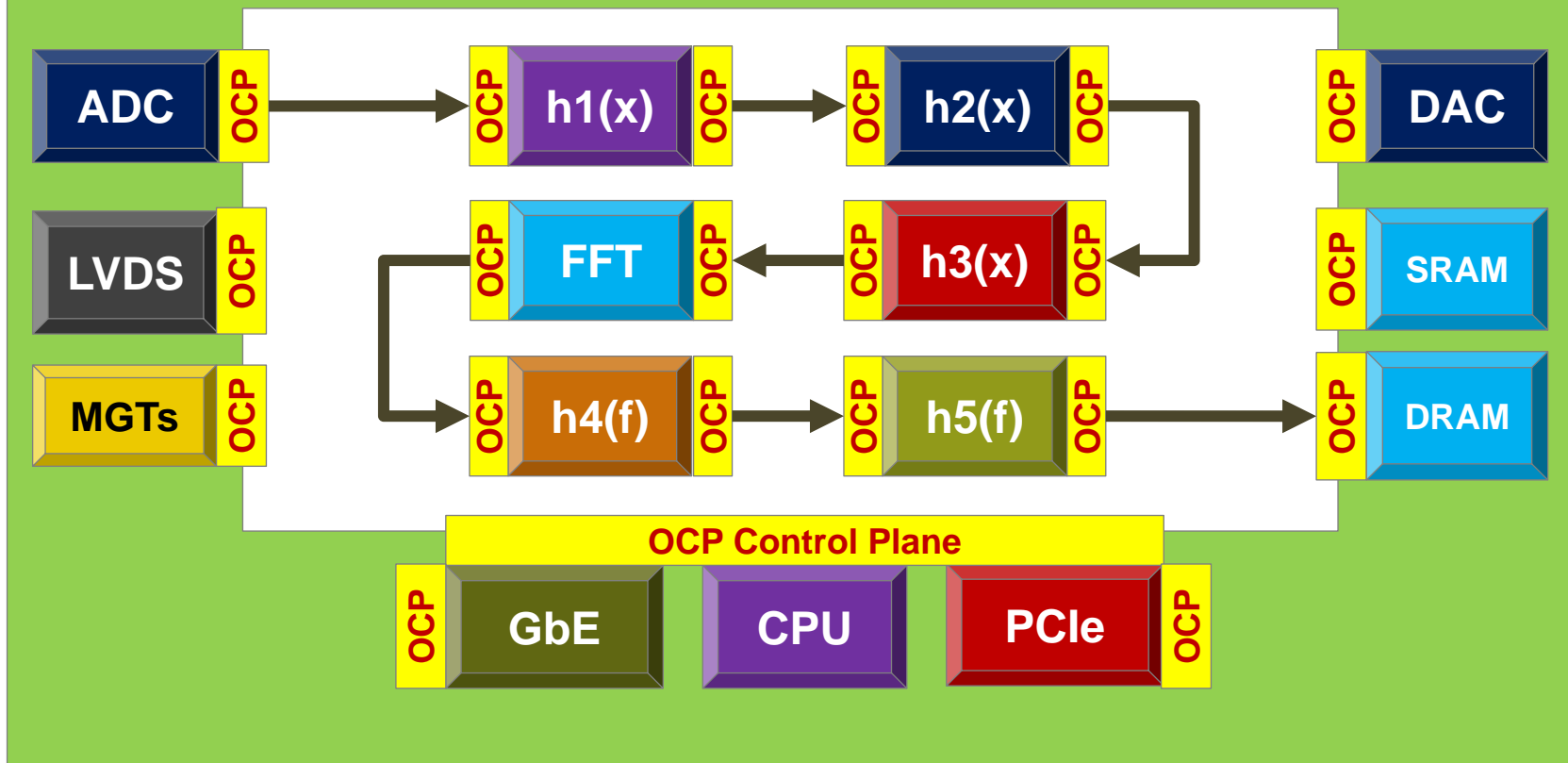


Waveform Integration

- Platform preparation provides a “pipe-cleaner” application to validate all necessary I/O interfaces
- Waveform development may be done in stages where some FPGAs are used in pass-through initially
- Waveform partitioning requires breaking the data path at appropriate places with high-speed data interconnect
- Control plane device map created for software control
- Waveform component initialization validation procedure

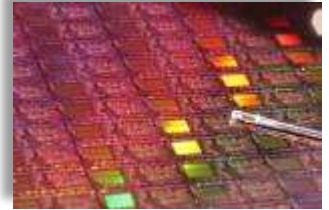
Waveform Integration

Complete Waveform

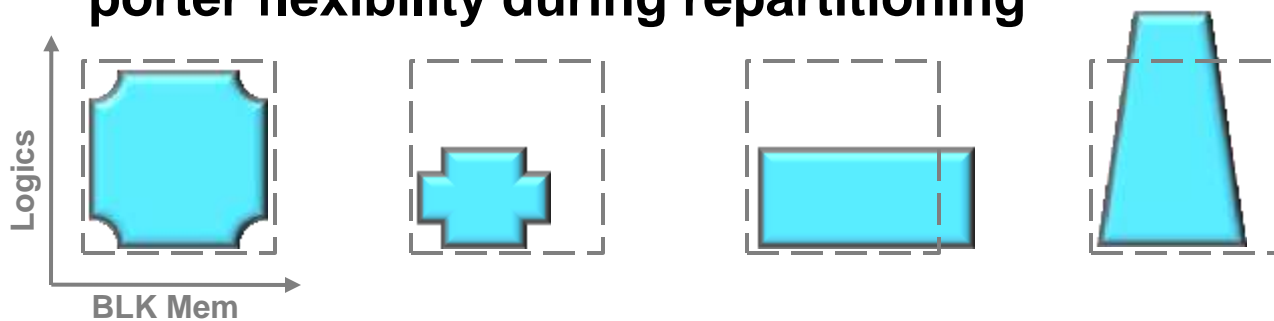


Waveform Partitioning

- ❑ New FPGA devices pack more resources (logic cells, Block RAMs, etc.)



- ❑ Need to impose limit on resources per component to give porter flexibility during repartitioning



- ❑ Oversized components may fit into larger FPGAs but not into smaller devices

- ❑ Taking middle ground makes the waveform more portable



Documentation

- **Specifications**

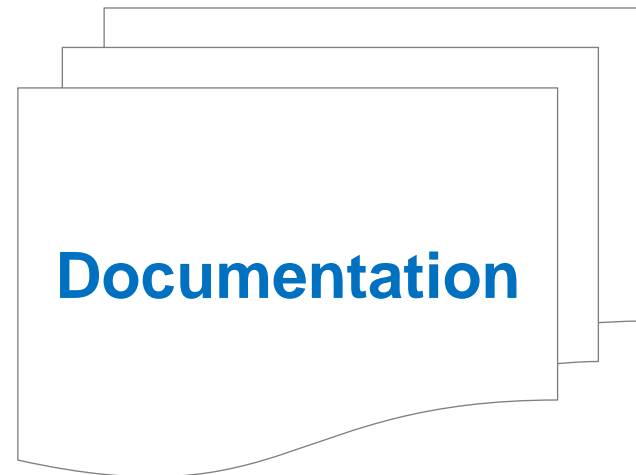
- Interface Definitions
- Architectural
- Functional
- Implementation (reference performance & utilization)
- Test Plan

- **User Documentation**

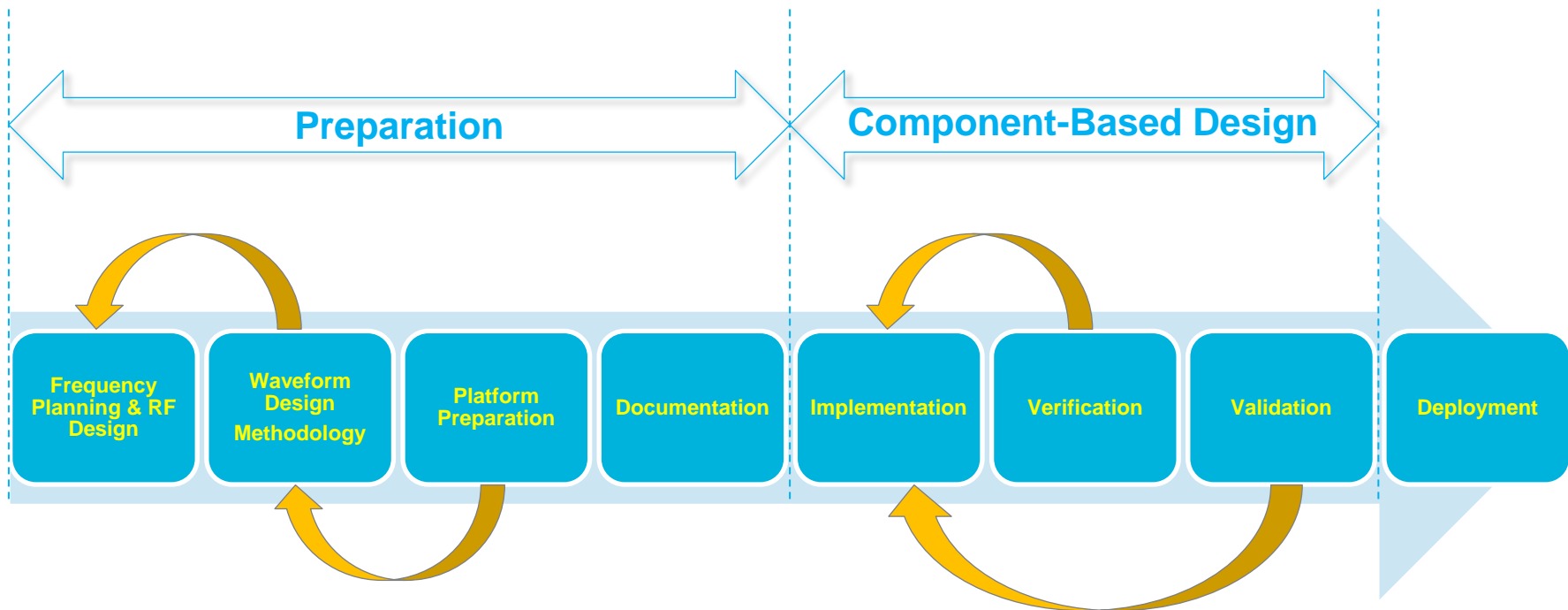
- Provided to the end-user

- **Design Reviews**

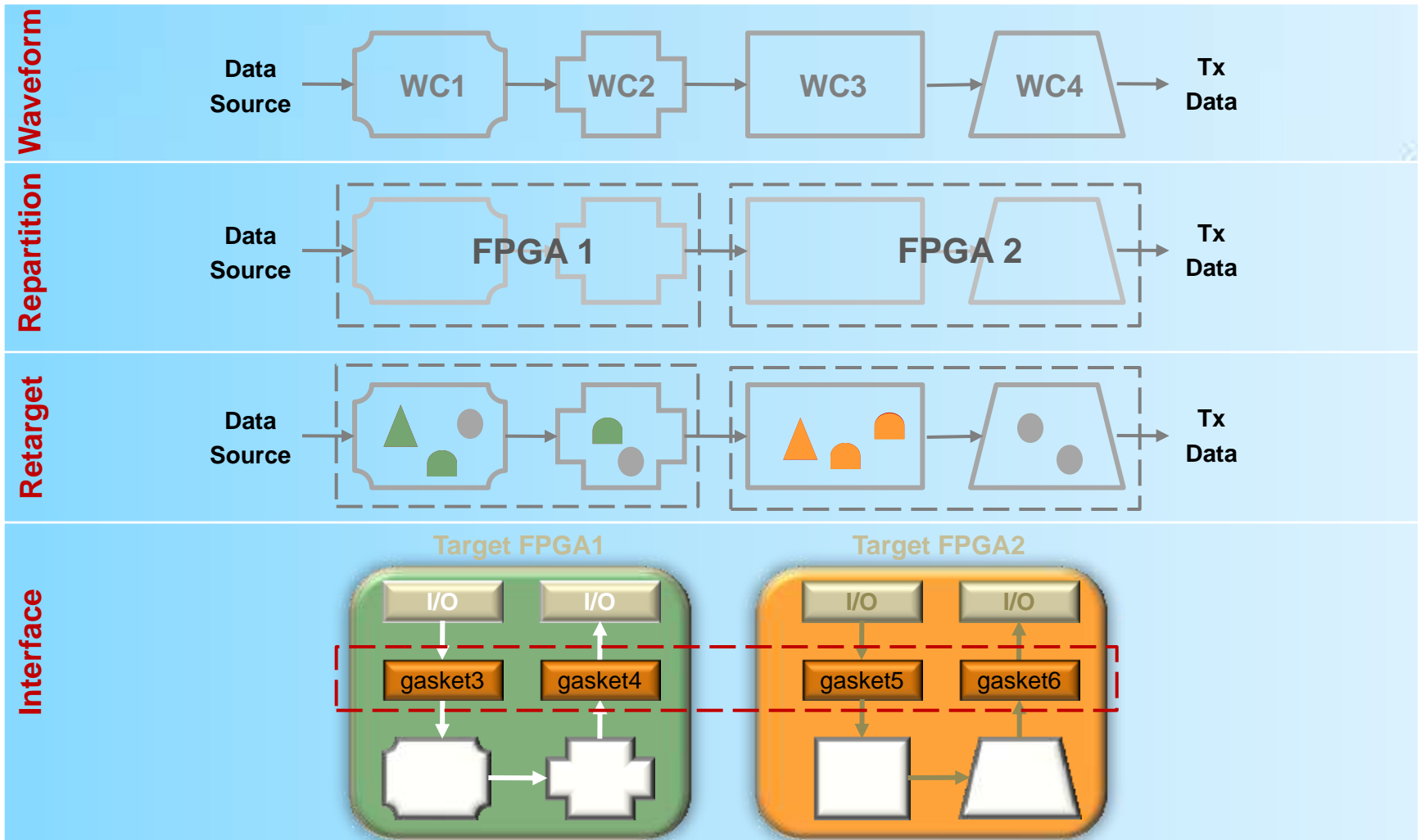
- Specification Reviews
- Code Reviews
- Verification Reports



SDR System Design Process



Component-Based Design



Embedded Software Considerations

- **Control and data flow**

- Most data is processed by the FPGAs and moved between them without CPU intervention (if possible)
- CPUs run control software which manages component initialization and any real-time hardware interaction

- **API, user interface, and debug facilities**

- API should provide a relatively architecture-neutral way to control hardware and data flows.
- API used to create UI and allow for custom UI

Embedded Software Considerations (cont.)

- **Distributed processors for large systems**
 - Central CPU on SBC for primary control of system, some signal processing (esp. altering data formats)
 - Embedded CPUs on FPGA boards used for control of the FPGAs, relaying commands from the central CPU
 - Extra CPUs on SBCs for additional data processing (if needed)
 - Proper coordination between CPUs is paramount.

Embedded Software Considerations (cont.)

- **Embedded system typically contains:**
 - Ethernet infrastructure (GbE/10GbE, etc...)
 - Control and/or data channel
 - Serial RapidIO (SRIO) and/or PCI-Express (PCIe)
 - Each has its benefits and drawbacks
 - Data channel
 - Can be used for control channel also
 - Flash file system/boot loader
 - Embedded OS, not always real time (RTOS)
 - VxWorks
 - Linux



VxWorks



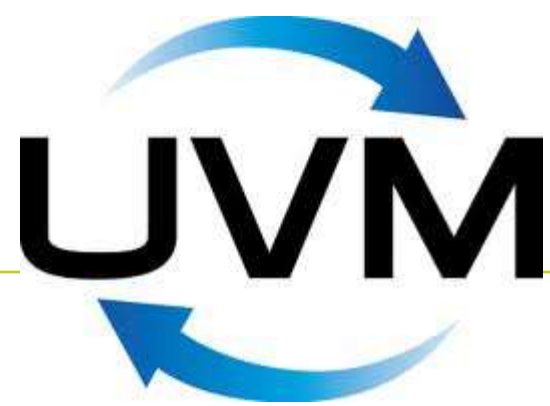
Embedded Software Considerations (cont.)

- **Boot time considerations**
 - Make booting as fast and reliable as possible

- **Middleware**
 - Multiple layers of middleware may be needed:
 - Communicate from CPU to FPGA in an architecture-agnostic manner
 - Data transmission between FPGAs

- **Robustness and error handling**
 - Reduce user interaction

Functional Verification



- **Components can be very complex**
- **Constrained-random verification methodology provides a way to run continuous regressions for improved code quality**
- **Functional tests are designed to detect a wide range of defects**
- **Large waveform verification can be compute intensive**
- **Platform verification creates additional challenges from complex interfaces**
- **Comprehensive lab validation is key to verifying complex waveforms**
- **Complete verification is a typically a hybrid approach where simulation and lab validation are used together**


Performance Validation

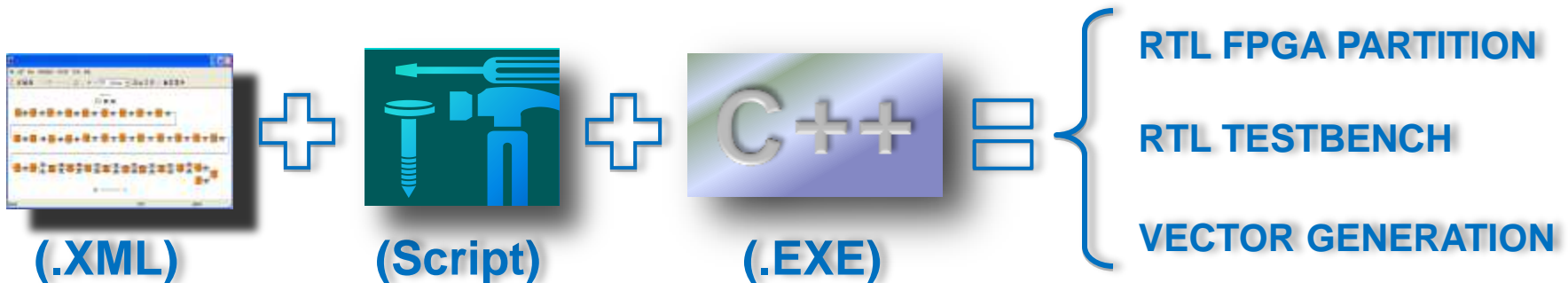
- **Performance validation is a process to characterize the overall system performance across a range of operational modes**

- **Digital Loopback Testing**
 - Characterize and validate the digital performance
 - Bit error rate
 - Implementation loss
 - Noise performance

- **Analog Loopback Testing**
 - Characterize and validate the analog performance
 - Bit error rate
 - Implementation loss
 - Noise performance

Standard Release Process

- ❑ Core of each waveform component can be encrypted to protect Intellectual Property (IP) 
- ❑ Automated repartition process using MITRE developed CAD tool
 - Allows porter to define FPGA boundaries and configure the waveform
- ❑ Automatic RTL code generation and additional verification facility
 - Unlimited test vectors enable porter to exercise all different modes and configurations



Standard Release Process

- ❑ **Waveform delivered to contractor on Virtual Machine (VM)**
 - VM behaves like virtual computer that runs on a host machine
 - Setup with user account (username and password)
 - VM environment is pre-configured by MITRE Lab (OS, tools, scripts, etc.)

- ❑ **Three Layers of documentations**
 - Environment : Introduce tools, script, general usage
 - Application : Overview of waveform settings and performance results
 - Component : Functional description and interface parameters

- ❑ **The VM copied to external mini-drive with AES encryption**
 - Instructions to access mini-drive given after contractors receive package



SDR System Components (OpenVPX Example)

- SBC
- FPGA Boards
- ADC, DAC, and Fiber Interconnect
- 19" Rack Chassis
 - High speed backplane
 - Power supply
- Transit Case



Conclusions

- **Large SDR systems are complex**
- **Employing a structured, systematic methodology greatly increases success**
- **A “divide-and-conquer” strategy can be used to improve schedule**
- **Up-front planning and verification pays off in decreasing the amount of time and money spent on back-end debug and validation**

Questions?

- **Thank You!**