DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

accellera SYSTEMS INITIATIVE™

Fast Track RISC-V System Validation Using Hardware Assisted Verification Platforms

DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

accellera SYSTEMS INITIATIVE™

Prasad Kadookar
Director Applications Engineering
Synopsys

Prasad is responsible to make customers successful with Synopsys Hardware Based Verification Solutions primarily HAPS Prototyping and Synplify Products. Prasad leads application engineering team responsible for supporting global customer base. He is based out of Bangalore. Prasad have been with Synopsys and working for FPGA and FPGA prototyping close to 20 years and have successfully deployed various HAPS & Synplify technologies at multiple customers. Prasad holds bachelor's degree in Electronics & Instrumentation from Pune University.

DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

accellera SYSTEMS INITIATIVE™

Mohan Singh
Staff Application Engineer
Synopsys

Mohan is HAPS Prototyping expert at Synopsys. His primary responsibility is to support customers on complex SOC prototype bring up on HAPS. Mohan comes with around 20 years of experience of FPGA Design, Prototyping and Emulation. His expertise includes RISC & ARM based SOCs Architecture, DDR and LPDDR memories and PCIE, USB, Ethernet, protocols. He is with Synopsys for 6 years and prior to joining Synopsys, Mohan have worked with Qualcomm on emulation. Mohan holds bachelor's degree in Electronics and Telecommunication from MVJ College of Engineering, Bangalore.

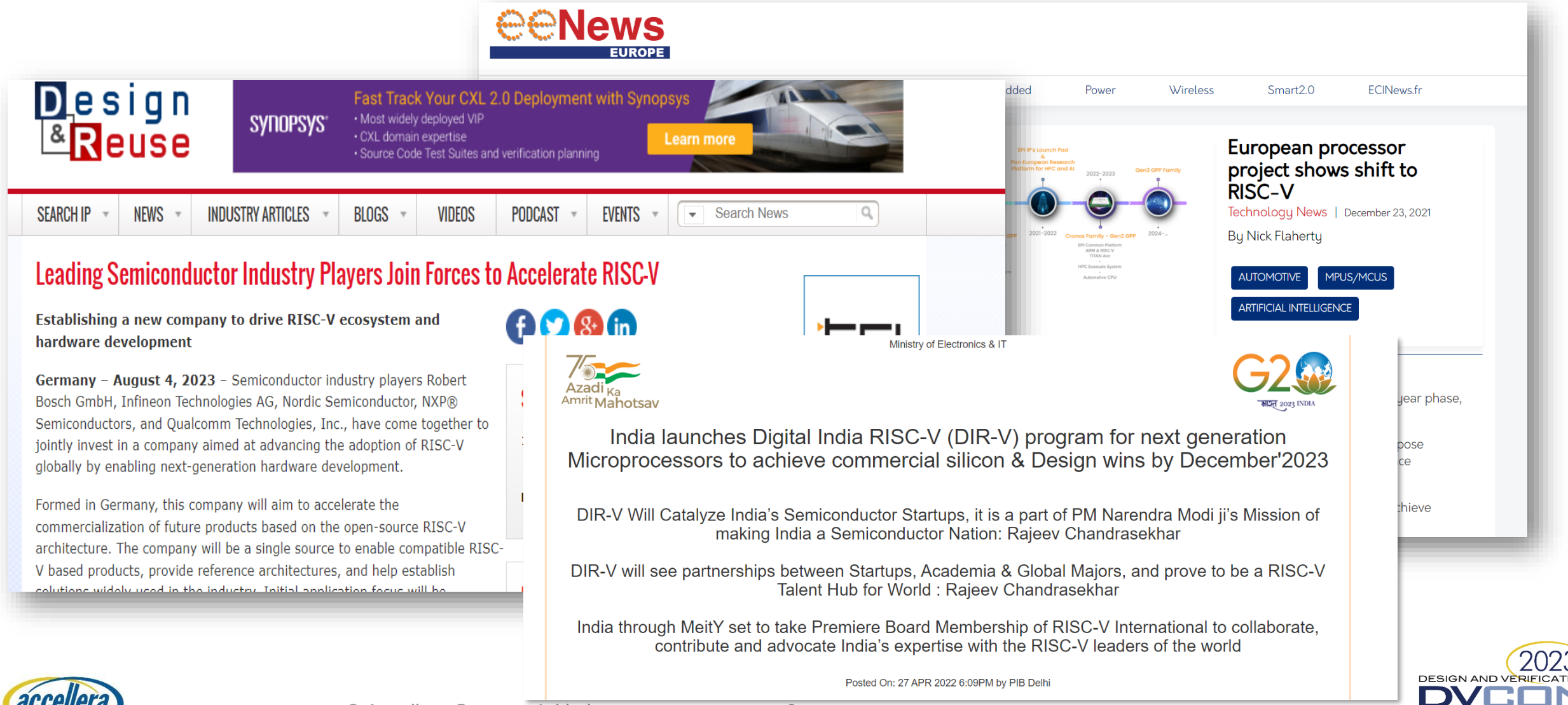
Agenda

- RISC-V Landscape
- High-performance FPGA-based Prototyping Landscape
- 5 Key Area for **Fast Track RISC-V System Validation**
- RISC-V on HAPS Demo

RISC-V Landscape

- Advances in Semiconductor and Computer
  - AI , 5G and Sensors
  - Even Quicker Uptick AI SoCs powered by RISC-V
- Device and System Complexity - > Design Complexity
- SOC: Changing Definitions, Increasing Complexities
  - Basic SOC (1+ Bus Structure , 1+ Complex Interconnect, 100-200 Discrete IP Blocks)
  - Value Multi Core SOC (3+ sub system, 4+ Complex Interconnect , 200-275 Discrete IP Blocks)
  - Advance Performance Multi Core SOC (4+ sub system, 5+Complex Interconnect , >275 Discrete IP Blocks
- Smaller nodes needs more software Development
- Verification & Validation includes:
  - RTL Verification with Simulations
  - RTL Verification with Emulations
  - Software Development Enablement with Prototyping
  - HW <-> SW Validation with Prototyping

RISC-V Landscape (Adaption)



Addressing Needs of the RISC-V Design Community

- Synopsys is a strategic member of RISC-V International
- Partnership with key RISC-V core providers, foundries and universities
- Interoperability of Synopsys IP with RISC-V solutions
- Availability of customized and adaptive flows for implementation and verification
- Availability of collaterals, user guides, training, cloud-based solution and design services

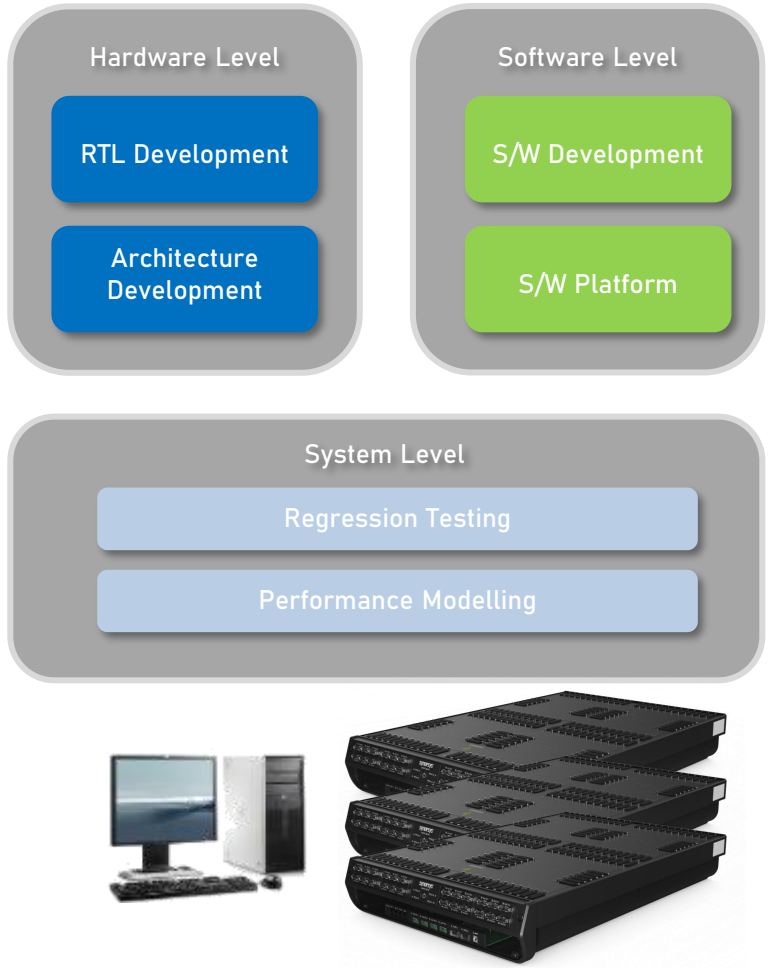
High Performance FPGA-Based Prototyping Landscape

Integral Part of IP and SoC verification

- From early RTL debugging to SoC performance modelling …. and everything in between

Primary requirements for FPGA Prototyping

- High Performance
  - Peta cycles for verification coverage
- High Capacity, Scalable and Flexible
  - Growing IP and SoC sizes
  - Expanded verification tasks
- Visibility Capabilities
  - High capacity and at-speed debug
  - Highest visibility
  - Flexibility to locate the hardest to find bugs



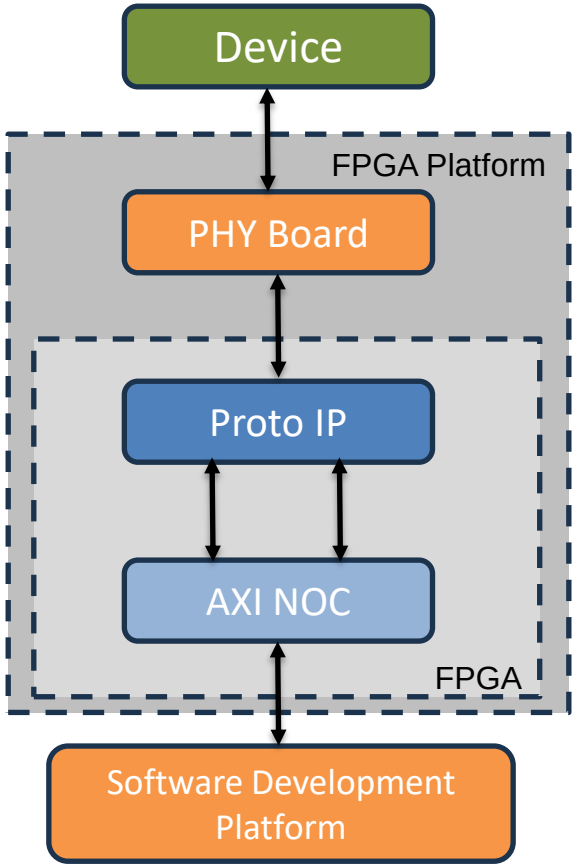
Fast Track RISC-V System Validation

- Manage the growing RISC-V SOC Complexity
- RTL Readiness
- Achieve High Performance
- Deep Cycle Debug Management
- Dynamics Of Data Center Prototyping

MANAGE THE GROWING RISC-V SOC COMPLEXITY

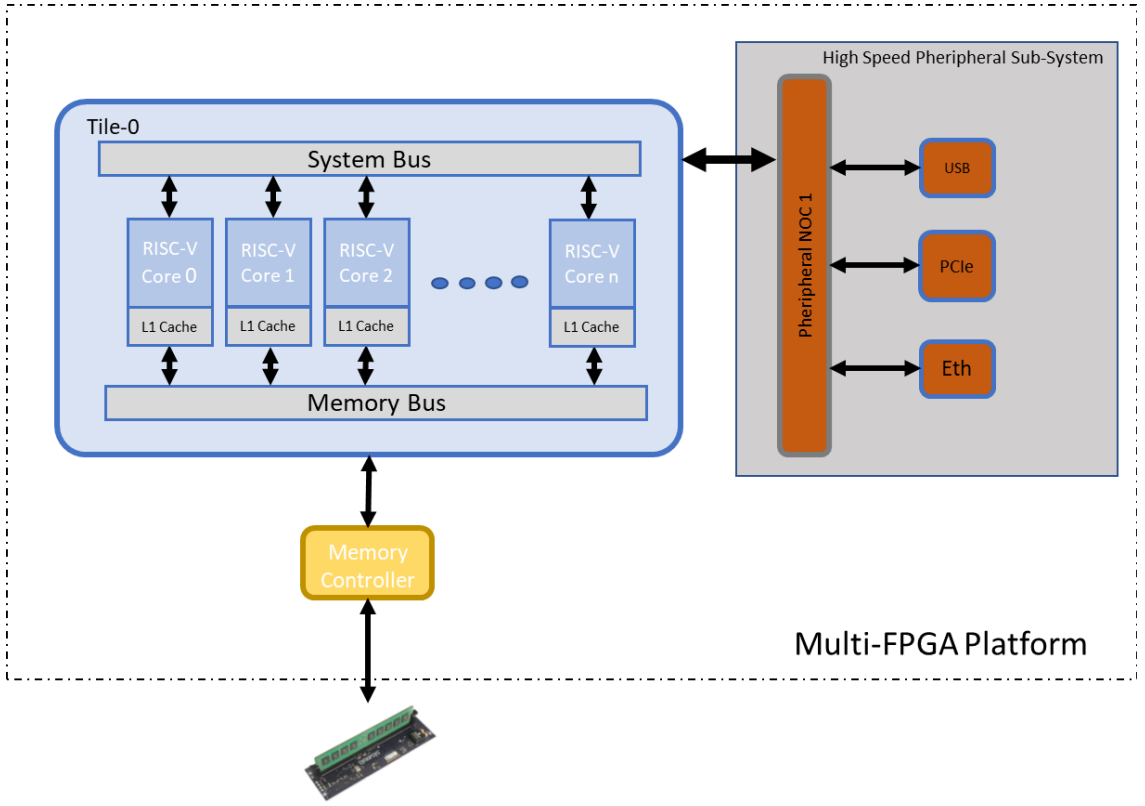
IP Prototyping

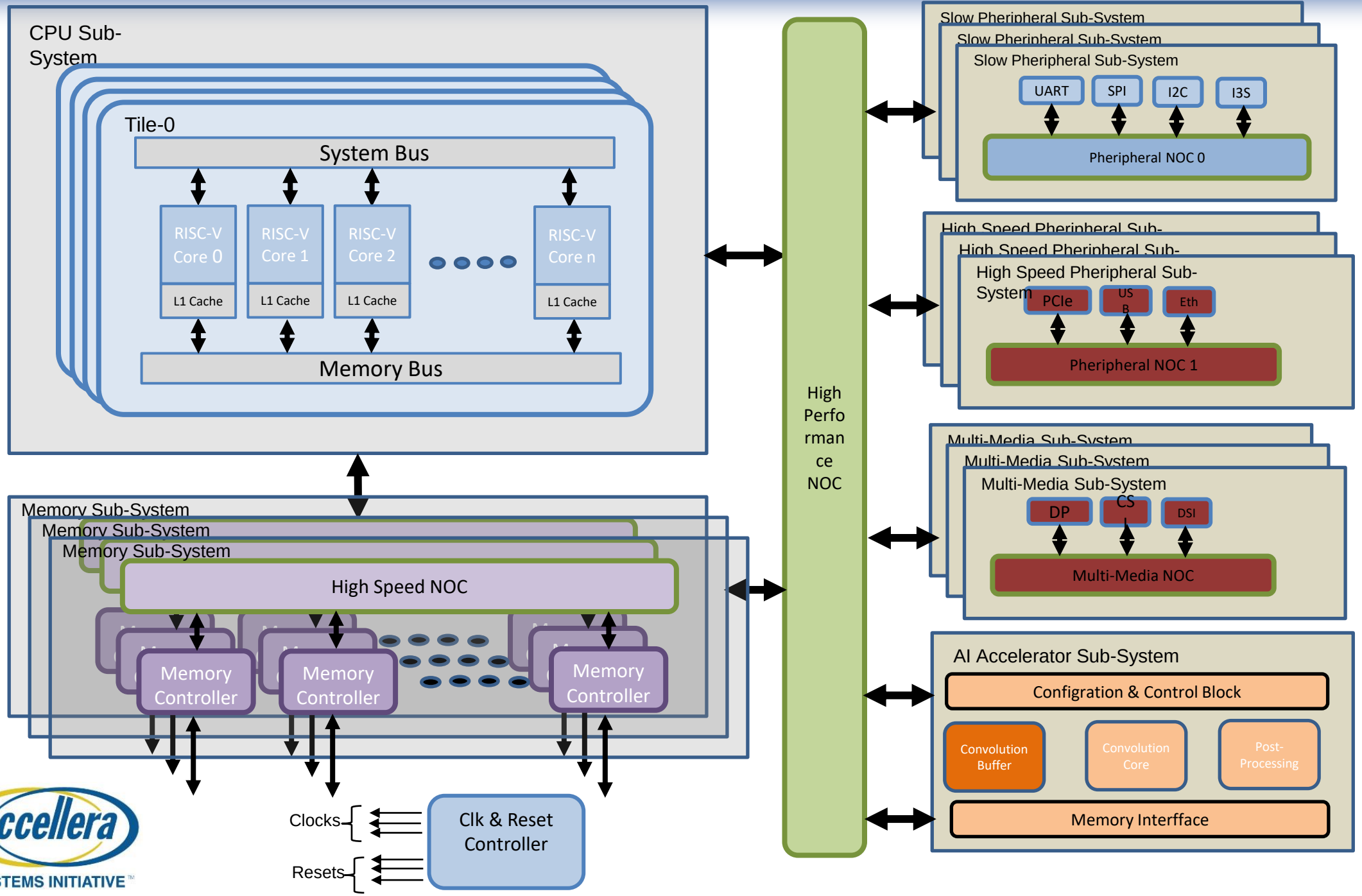
- Prototyping Development cycle
  - Target specific IP
  - Utilize 1 - 2 FPGA
  - Create Test Environment
  - Prototype IP for FPGA platform
  - Interface SW development platform
  - Develop Test scenarios
  - Monitor test results
- Prototyping Challenges
  - Converting ASIC models for FPGA
  - Adding extra IPs to create test environment
  - Interfacing External Platforms
  - Creating Platforms to monitor the results



Subsystem Prototyping

- Prototyping Development cycle
  - Target Sub-System
  - Utilize 4-8 FPGA
  - Create Test Environment
  - Partition Sub-System in Multi-FPGA platform
  - Interface SW development platform
  - Develop Test scenarios
  - Monitor test results
- Prototyping Challenges
  - Converting ASIC models for FPGA
  - Partitioning the Sub-System
  - Adding extra Transactor IPs to create test environment
  - Interfacing External Daughter cards on Platform
  - Creating Platforms to monitor the results



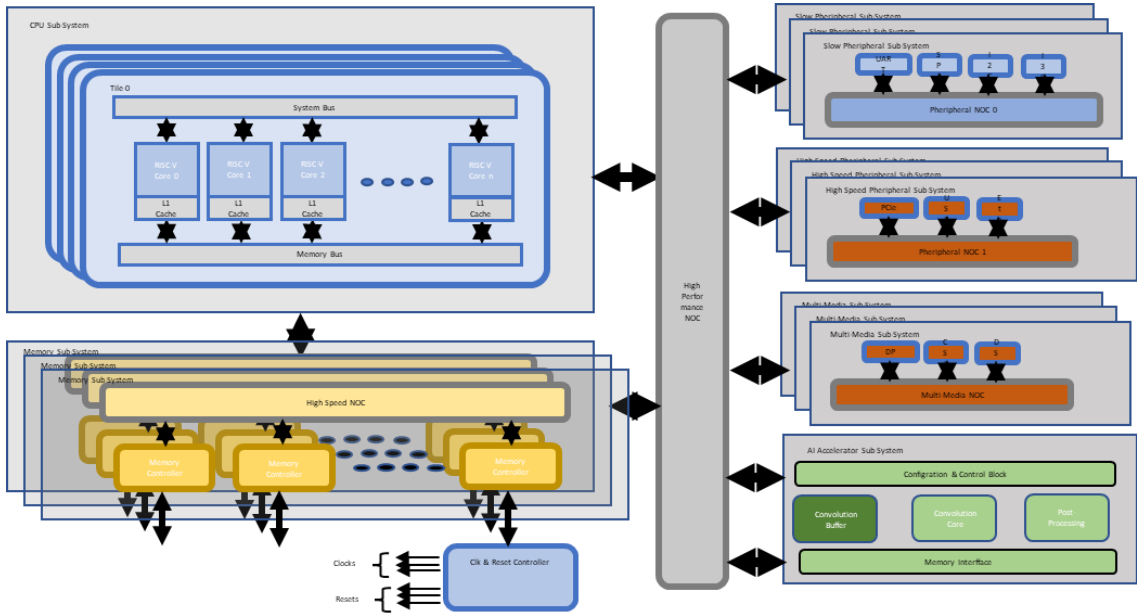


accellera SYSTEMS INITIATIVE™

2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

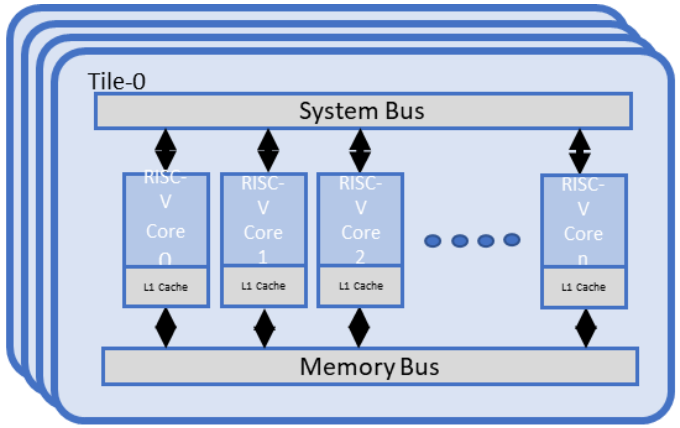
SoC Prototyping

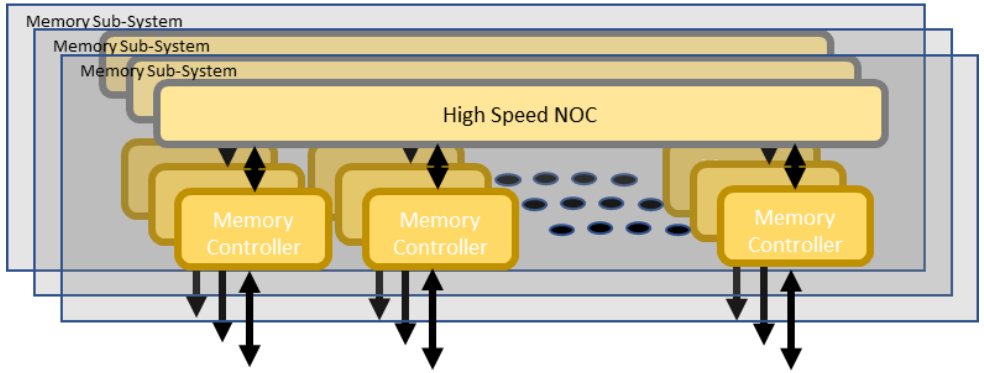
- Prototyping Development cycle
  - Target System On Chip (SoC)
  - Utilize 10-100 FPGAs
  - Create Test Environment
  - Partition SoC in Multi-FPGA platform
  - Identify Replication & Non-Replication Modules
  - Team Based Design
  - Interface SW development platform
  - Develop Test scenarios, Monitor test results
- Prototyping Challenges
  - Converting ASIC models for FPGA
  - Partitioning the SoC
  - Identifying Replication & Non-Replication Modules
  - Interfacing External Daughter cards on Platform
  - Creating Platforms to monitor the results



SOC Prototyping (continued)

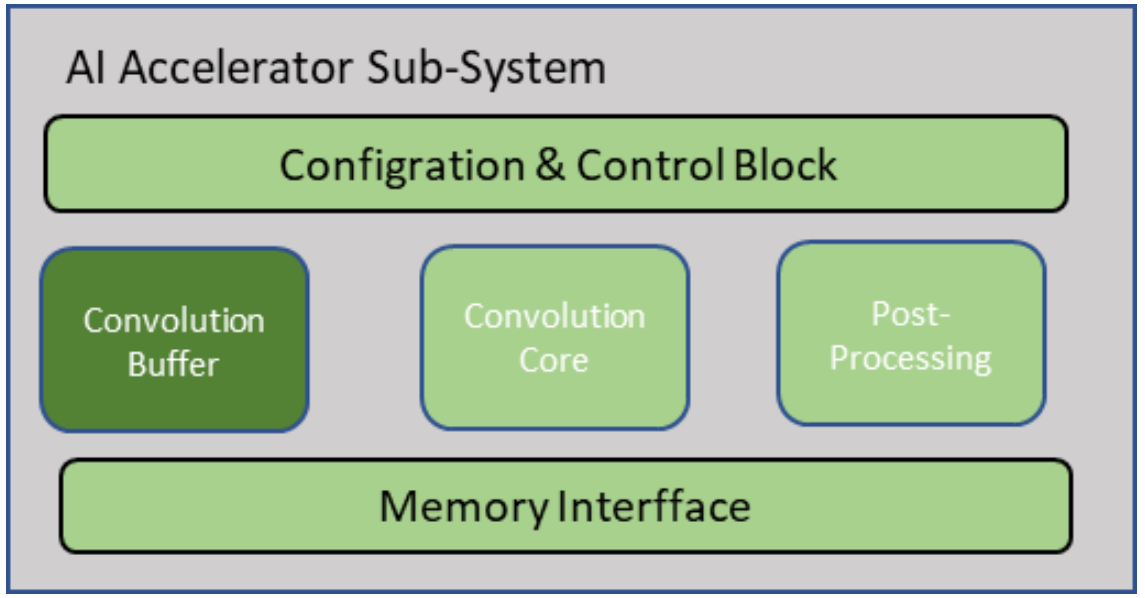
- Partition Replication Modules
  - Identify Replication Modules & their size.
  - Group FPGAs with Identical on-board interconnections
  - Finalise on the IO assignments with care
  - Partition the Module within a Group.
  - Identify on board Clock sources and map to all groups
  - Map resets to all groups
  - Generate bit files of a group & use the same to others
  - Interface SW development platform
  - Develop Test scenarios
  - Monitor test results





SOC Prototyping (continued)

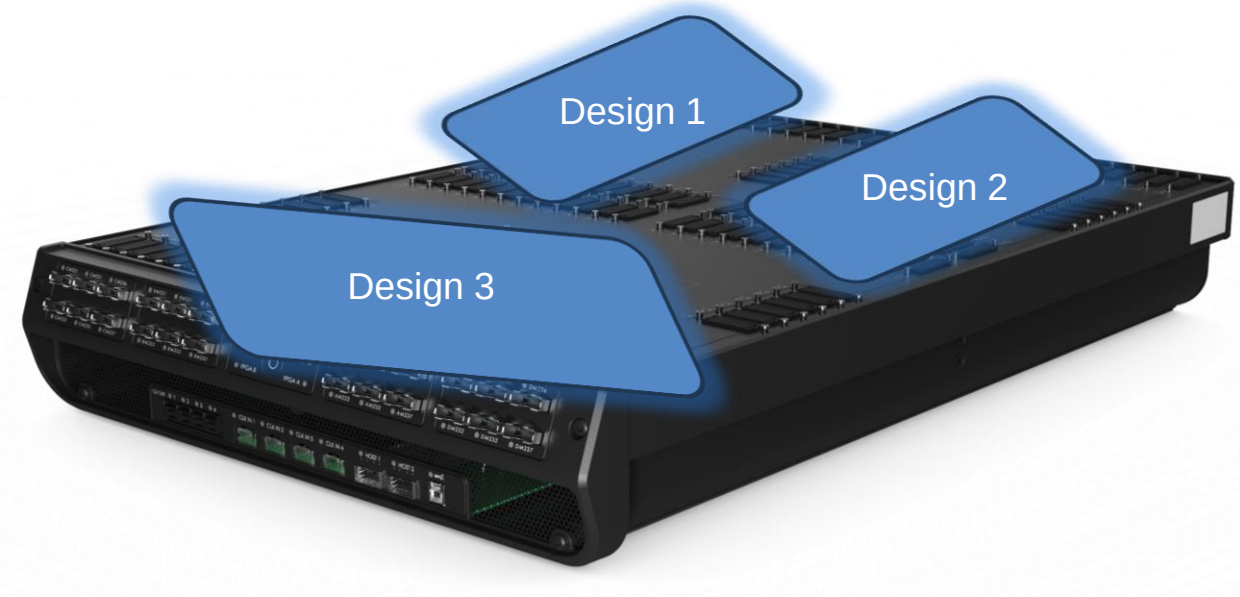
- Partition Non-Replication Modules
  - Convert ASIC models for FPGA
  - Partition the Sub-System
  - Add extra Transactor IPs to create test environment
  - Interface External Daughter cards on Platform
  - Create Platforms to monitor the results



Synopsys HAPS® Solutions

- HAPS Hardware
  - Flexible and Scalable Hardware (IP to SOC Prototyping)
    - Desktop -> Lab Table-Top -> Rack Mount
  - Off-the-shelf Synopsys DesignWare Intellectual Property Kit & Speed Adaptors

- HAPS® ProtoCompiler Software
  - Team Based Design
  - Multi Design Mode
  - Farm Management



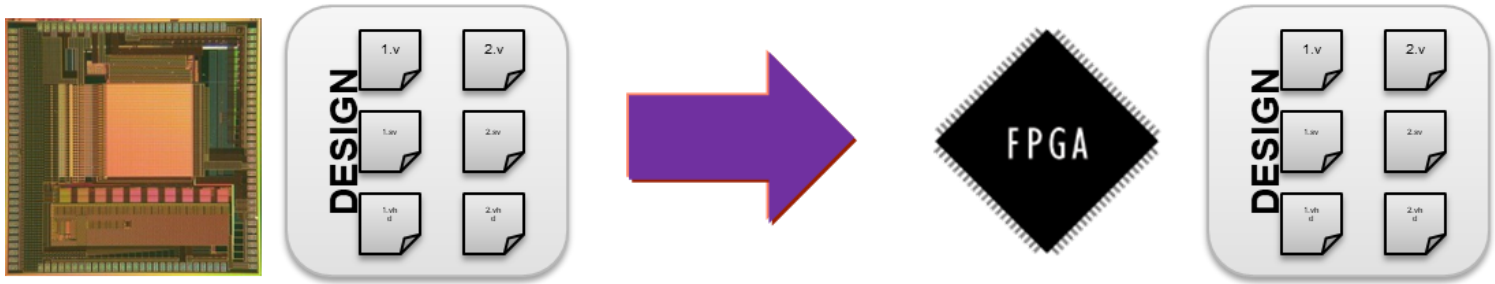
RTL READINESS

RTL Readiness (RTL Change)

- **ASIC RTL must be modified for efficient FPGA prototyping due to the following reasons:**
  - ASIC design elements not suited for FPGA technology
  - Tweaking for higher performance
  - Efficient Debugging
  - Modularizing RTL for FPGAs

```
`ifdef FPGA_PROTO

     ... Modified FPGA RTL
`else
     ... Original ASIC RTL
`endif
```

```
`ifdef FPGA_PROTO

// Downward XMR read

assign d = inst1.a ;
....

// Upward XMR Write
assign top.d = a

//remove BIST Logic
Assign bist.clk = 1'b0

`endif
```



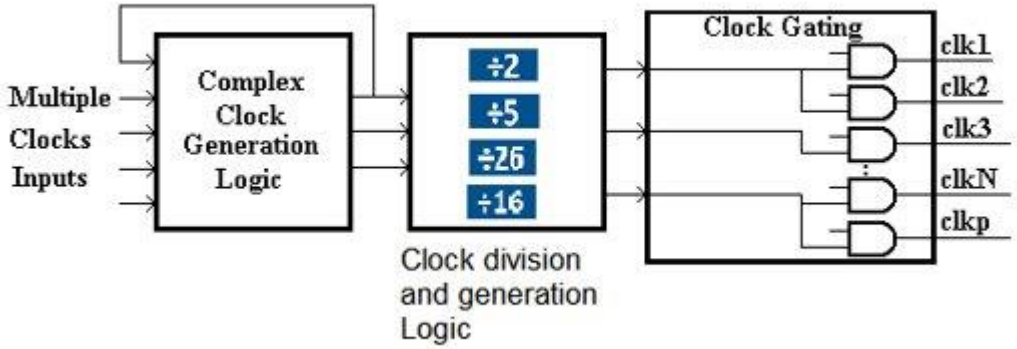
RTL Readiness (Clocks)

- **Problems**
  - Timing violations due to clock skew can be caused by clock-gating logic
  - Number of clocks in a design exceeds the number of global clocks available in the FPGAs/FPGA prototyping Platforms

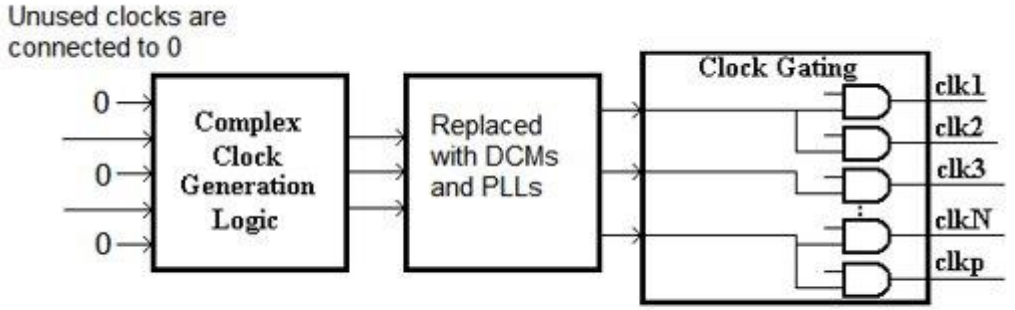
- **Solution**
  - To successfully prototype an ASIC into an FPGA, make sure that all the ASIC design clocks fit into the clock resources of the FPGA prototyping platforms and FPGAs by doing the following:
  - Simplifying the clock networks
  - Removing/reducing Skew across FPGA
  - Using the clock-conversion feature to convert the remaining gated and generated clocks

RTL Readiness (Clocks)

- **Replace clock generation logic with MMCMs and PLLs**



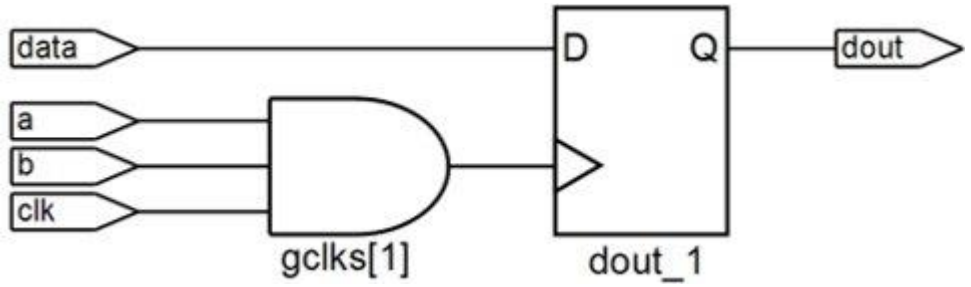
- **Tie constants to clock inputs not being used for prototyping**



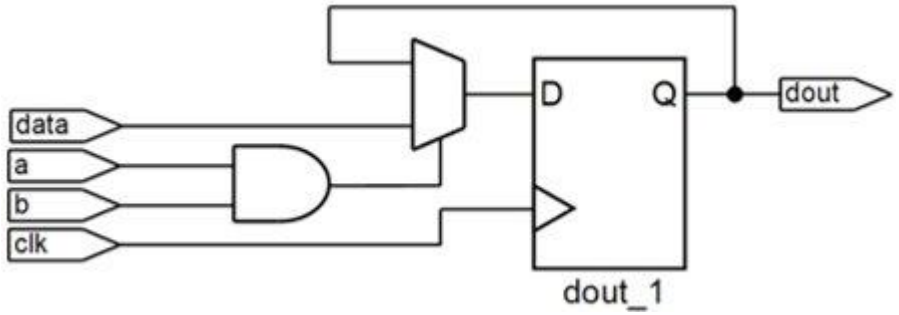
RTL Readiness (Clocks)

- After simplifying the clock networks, gated clocks may still exist in the design

- To eliminate these clocks, use the clock conversion feature in the synthesis tool, which moves the gating clock logic from the clock pin of the sequential elements to the enable pins



Without Gated Clock Conversion



With Gated Clock Conversion

RTL Readiness (Memories)

Simple Memory Handling

- Recode Memories based on FPGA Synthesis

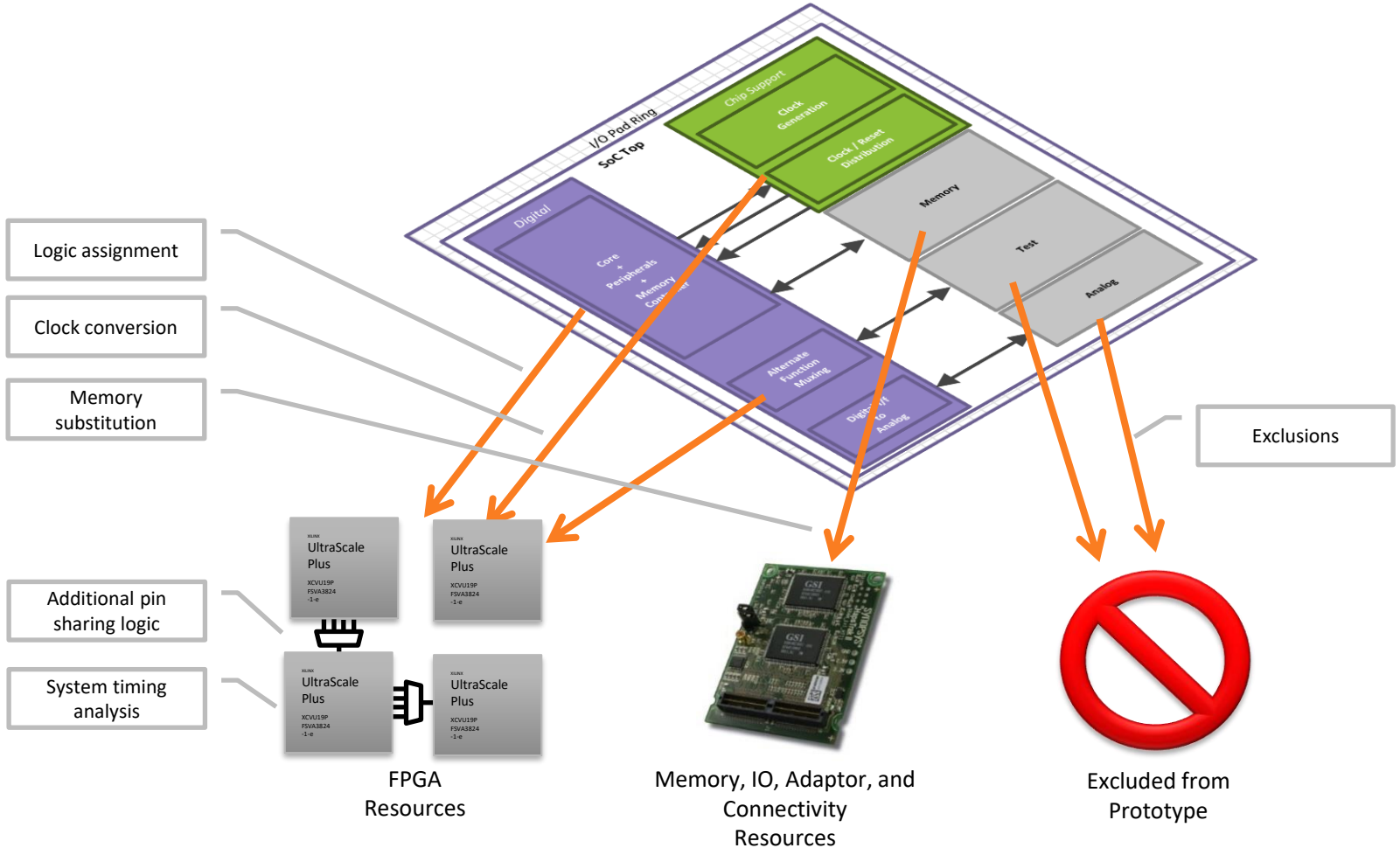
Memories	Types	Purpose
ROMs	Single Port Dual Port	Stores boot image, the constant co-efficient in digital filters, etc.
High-Speed SRAMs	Single Port Multi Port	Used as cache memory in processor cores and graphics processing units (GPUs)
High-Density SRAMs	Single Port Multi Port	Stores data in basic peripherals such as UART, I2C, and GPIO
High-Density Register Files	Single Port Multi Port	Stores data in camera interfaces
FIFO	Asynchronous Synchronous	Stores data in networking applications
CAM	Multi Port	Used in network switches and routers

Complex Memory Handling

- Virtual Memory Models
  - HBM family of memory models
  - DDR family of memory models
- External Daughter Cards
  - DDR3
  - DDR4
  - LPDDR3
  - LPDDR4

RTL Readiness (Summary)

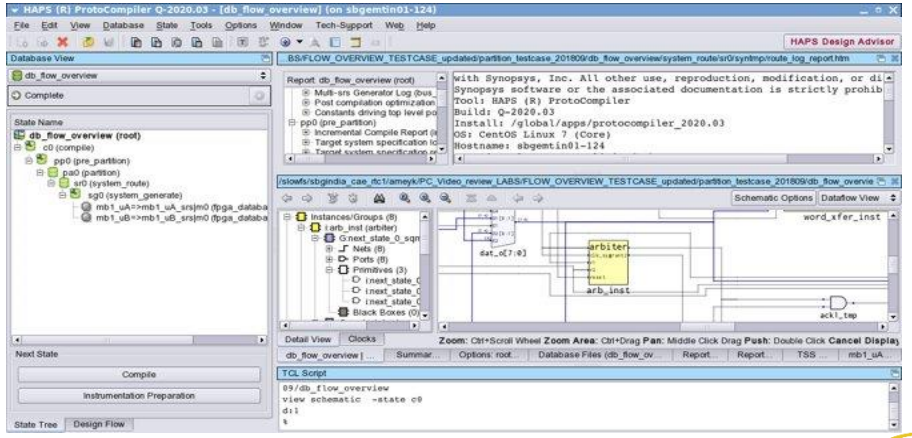
- Stub out Analog & Test Logic not required for prototypes
- Optimize Clocks
- Handle Memory
- Account for additional Mux Logic



Synopsys HAPS® Solutions

- HAPS® Hardware
  - Daughter Cards (Off the shelf)
  - IPK Solutions (Off the shelf)
  - Inbuilt Checks for HAPS Daughter Cards and Cables

- HAPS® ProtoCompiler Software
  - Easy Mapping of Global Clocks on Platforms
  - Automatic Gated Clock Conversion
  - Efficient Memory Mapping to URAM/BRAM

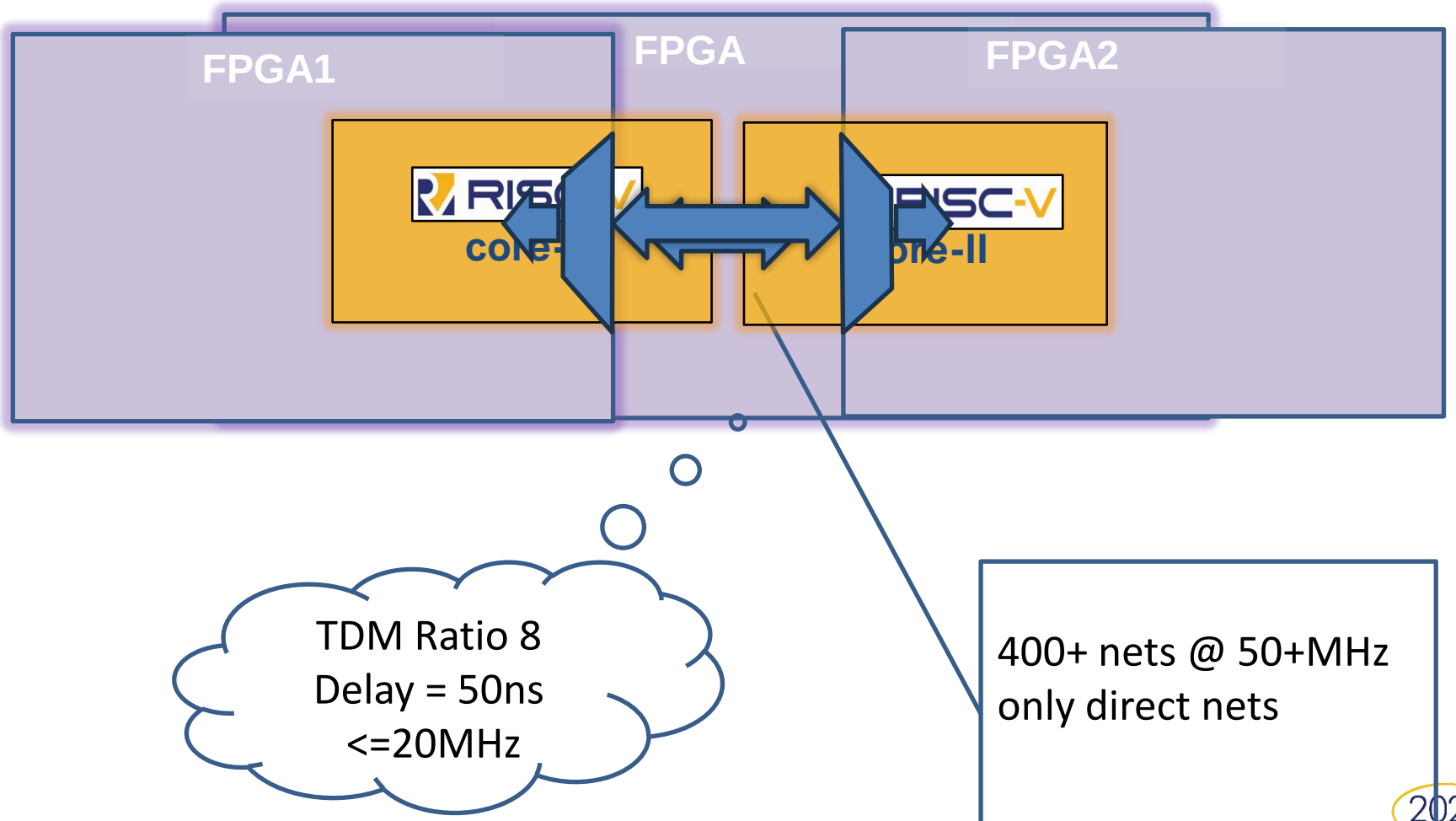


ACHIEVING HIGH PERFORMANCE

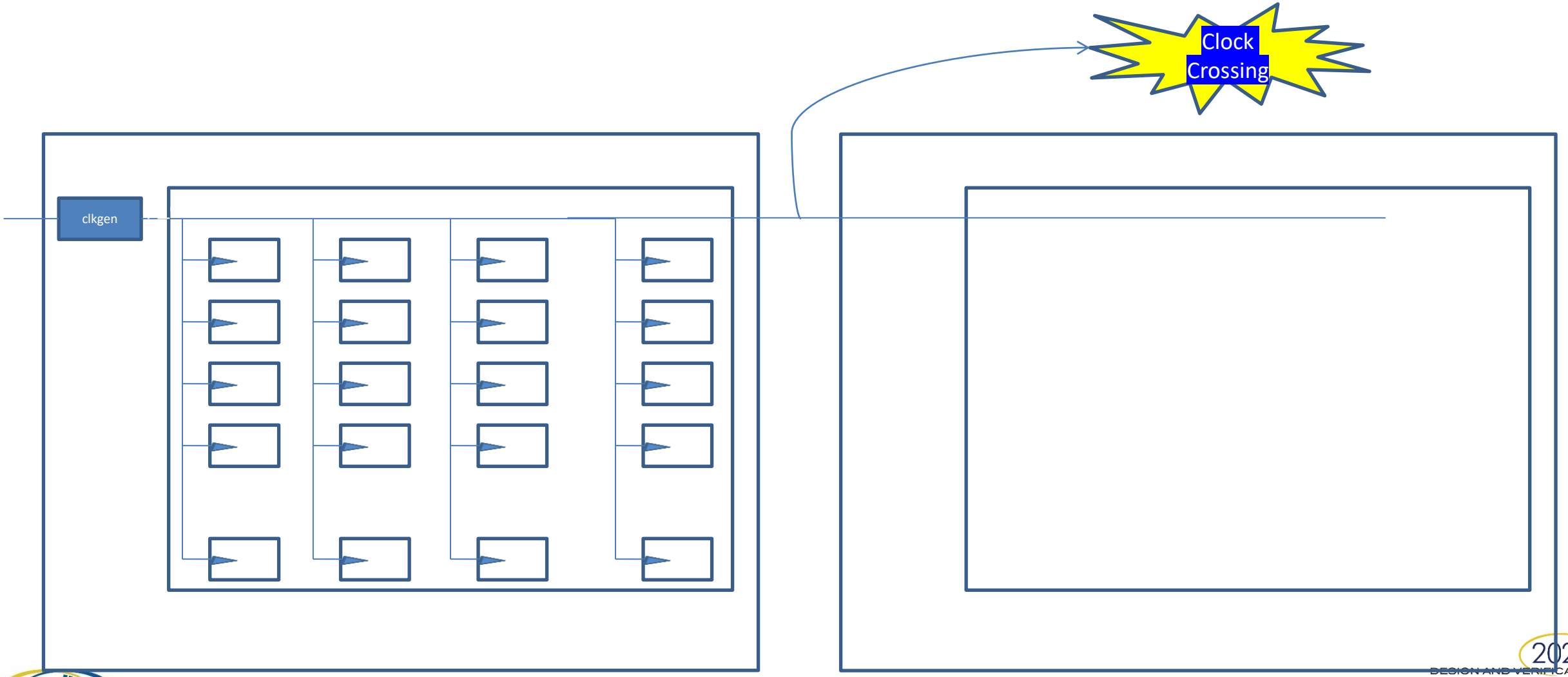
Timing Aware Partitioning with core clk

Partitioning Core
- CPU cores

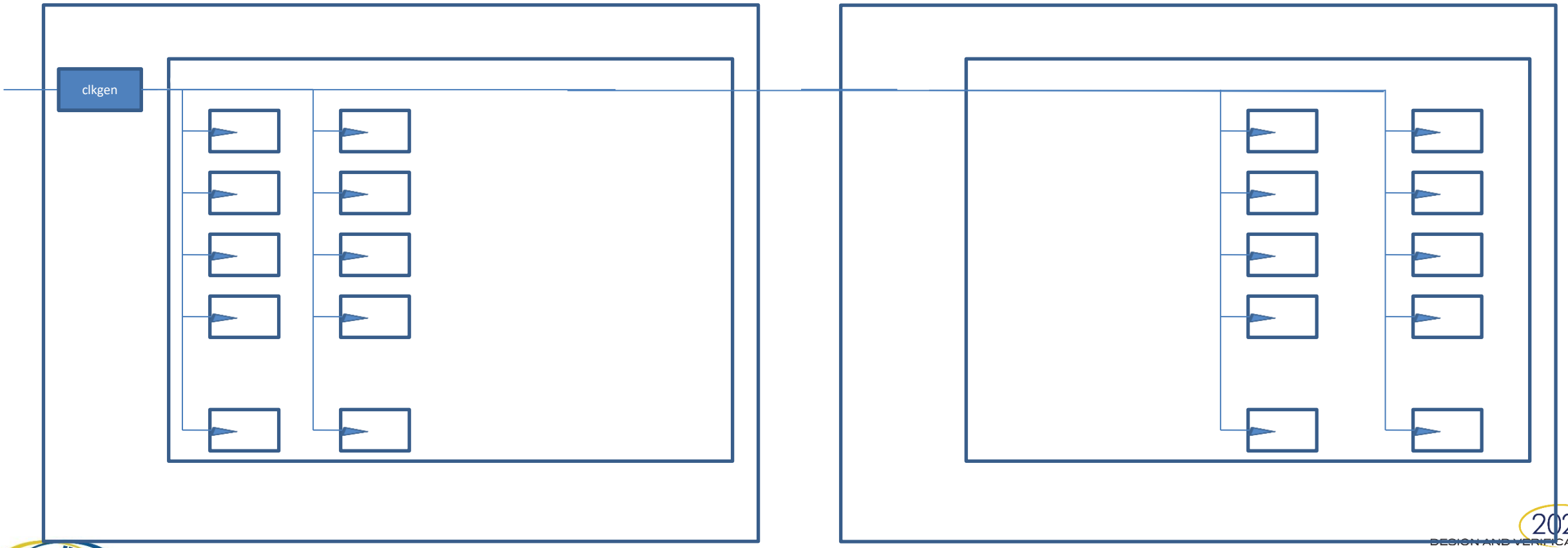
assign to FPGAs
- Identify nets between these two cores
- Make sure enough traces are there between two FPGAs
- Use lowest TDM based on slack



Resolving - Clock Crossings/Cut Clocks

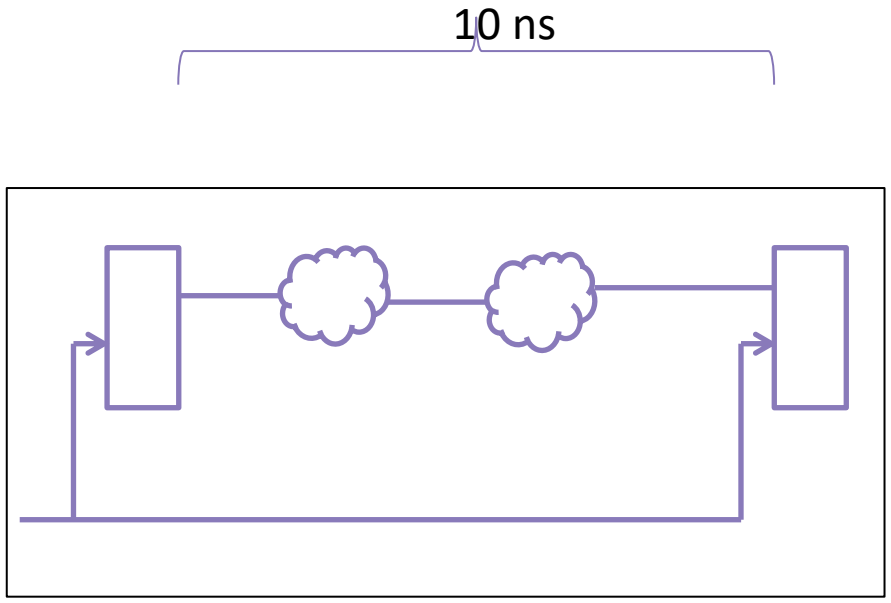


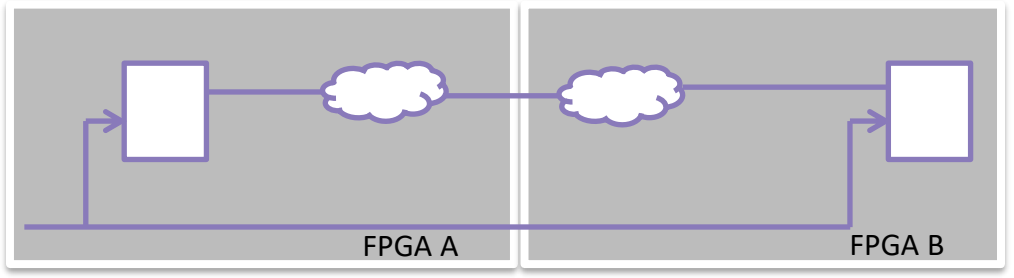
Resolving - Clock Crossings / Cut Clocks



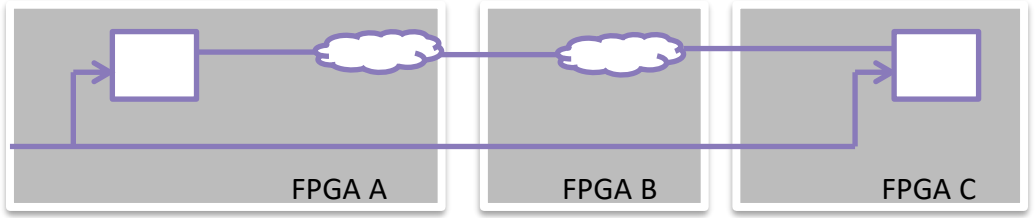
Resolving Hops

- What is a Hop?
- An FPGA boundary that a path crosses between start and end points
- Optimizing to minimize hops is important!

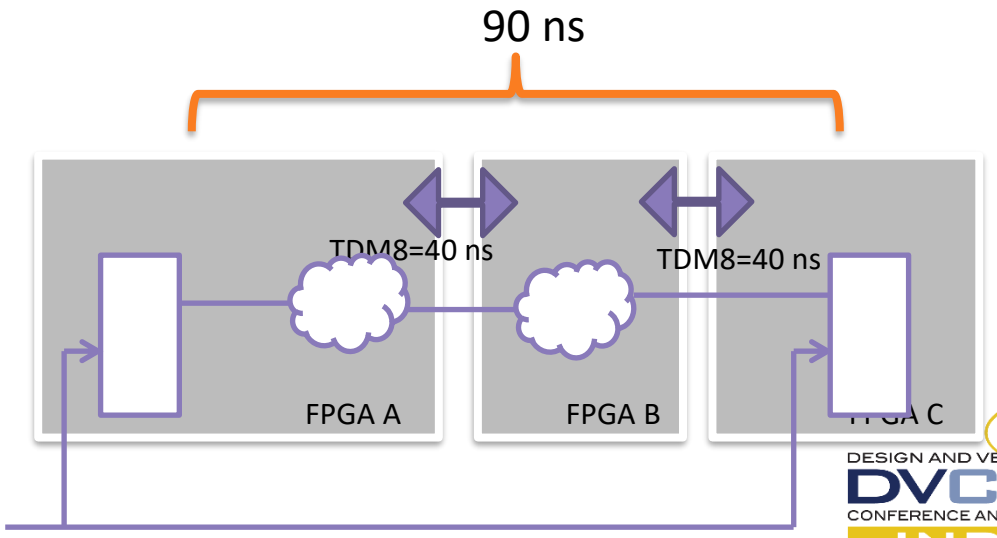




Single “Hop”



Two “Hops”



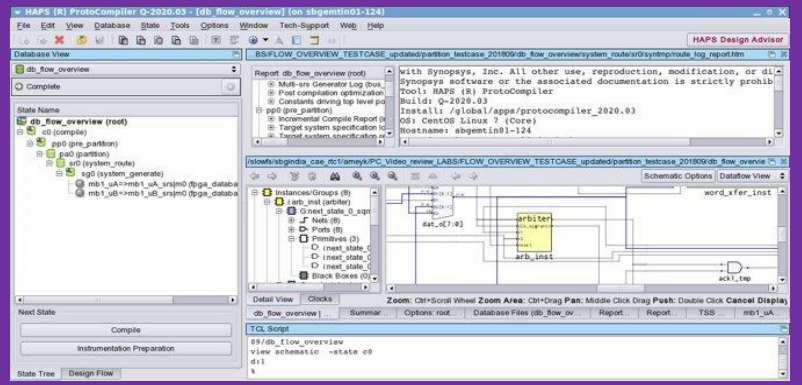
Synopsys HAPS Solutions

- HAPS Hardware
  - Flexible & Scalable Hardware
  - 18-24 Global Clocks

- HAPS® ProtoCompiler Software
  - Highly Flexible Partitioner
  - Timing Aware Partitioning
  - Automatic Cut Clock Resolving
  - High Speed Time Division Multiplexing (HSTDM)
  - Multi Gigabit Time Division Multiplexing (MGTDM)
  - Reduce Multi Hops

HAPS-100 Hardware

HAPS® ProtoCompiler Software



DEEP CYCLE DEBUG MANAGEMENT

The Need for Deep-Cycle Visibility

Deep cycle visibility a prerequisite for high performance prototyping

- Visibility for debug
  - Fault finding at all levels
- Visibility for performance measurement
  - Data traffic characteristics due to S/W interaction with external memories, data caches
  - Data bus transactions when running on H/W
- Visibility for off-line verification
  - Capture for RTL and S/W Debugging
  - Capture “real-world” stimuli for off-line module simulation

Continued debug innovation expands the scope

- High performance and complex event detection
  - Peta cycles of cycles to get near the point of Interest
- High capacity and wide captures
  - Full state captures over 100,000s cycles

	Simulation	Emulation	Prototyping
Speed	Slow (~0.1 – 10KHz)	Medium (~1MHz)	Fast (~10 – 100MHz)
Visibility	High	Medium	Prototyping Debug Innovation

The Need for Deep-Cycle Visibility

Case Study: Trapping the Hardest to Find Bug

Might only happen after days or even weeks of prototyping run

- H/W and S/W nearing maturity
  - Error happens late in the verification cycle
- Need peta cycles to get to the point of interest
  - Only possible with a FPGA Prototyping platform operating at the highest speed

What we do and don't know

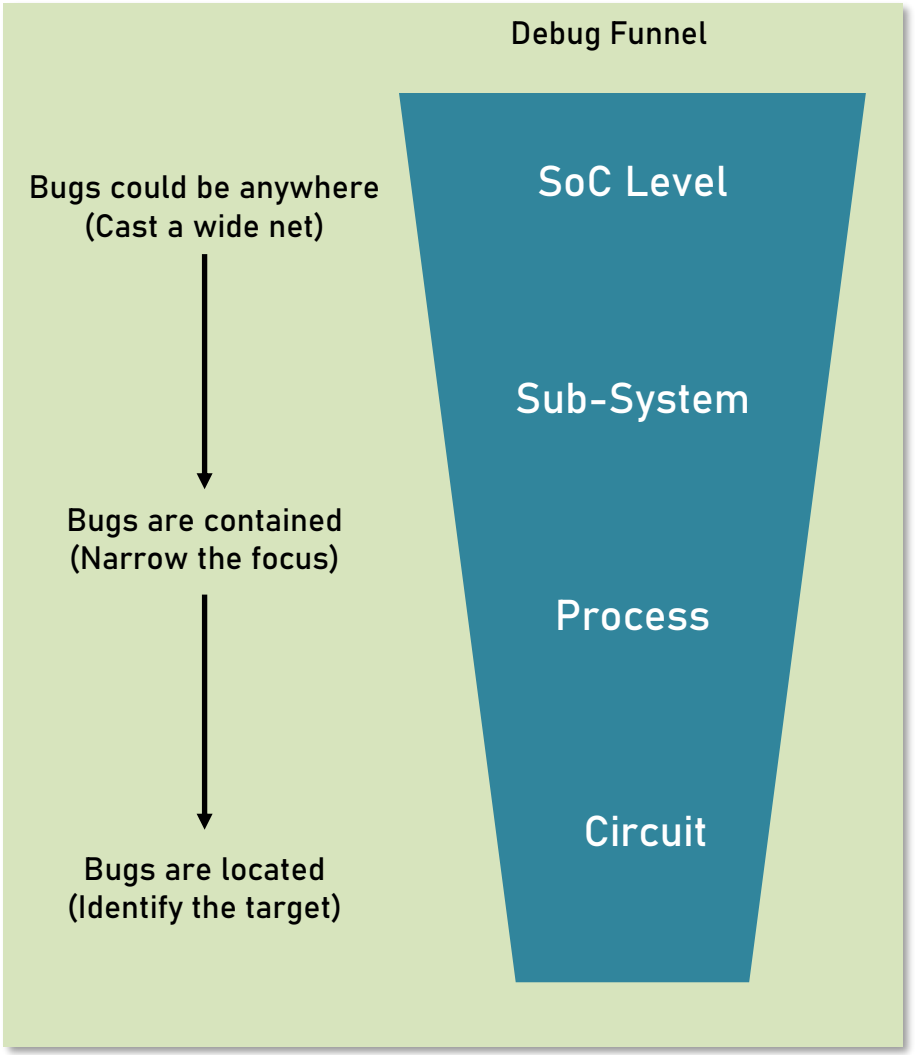
- S/W error reported via higher-level transaction (UART with line count)
  - Maybe pointing us to a specific process
- Something happened leading up to S/W error
  - We might need to detect a complex sequence of events to help isolate that
- Maximum visibility required to locate and fix the error
  - We are not completely sure what to look for

The Need for Deep-Cycle Visibility

Case Study: Trapping the Hardest to Find Bug

What we need to do

- Re-run the platform at the highest speed to a point prior the line counter value
- Switch to a lower speed to increase the visibility and look for primary events
- Once detected, capture the full state of the platform to help locate and fix
  - Off-Line debugging at both the S/W and RTL level



Synopsys HAPS - DUT Debug

Multiple visibility options

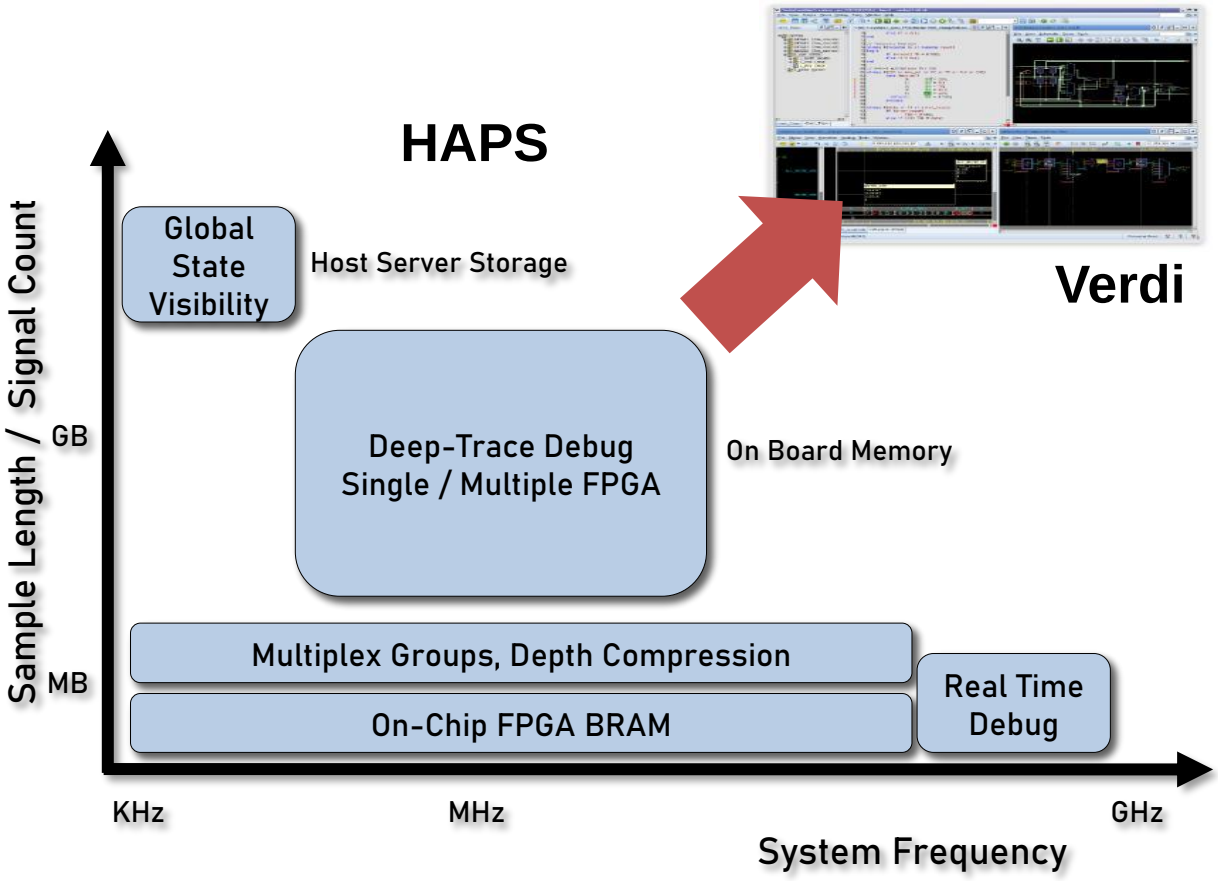
- Performance, capacity and intrusion into the DUT
- Real-time, at-speed debugging, full state capture

Increased visibility capacity and flexibility

- Greater event detection and capture resolution
- Eliminate re-spins to capture the necessary scope

Different technologies combine to great effect

- First event detection at full speed
- Higher capacity visibility at reduced speeds
- Complete state capture at controlled speeds



DYNAMICS OF DATA CENTER PROTOTYPING

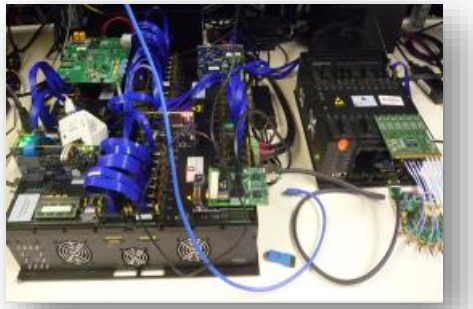
The Dynamics Of Data Center Prototyping

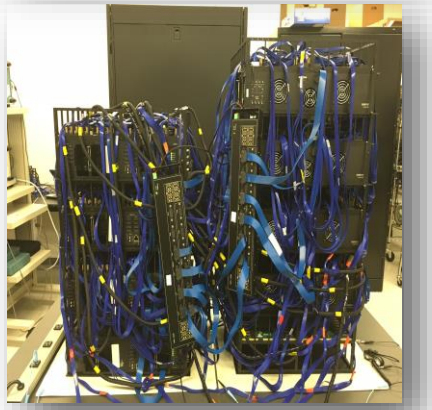
FPGA prototyping installations exist in multiple environments

- Desktop Systems for I/P and S/W development
  - Interactive configuration changes, test access, visual output
- Lab-based systems for large system set-ups, exploration and development
  - Initial system configuration testing
- High-Capacity prototyping farms with remote access
  - Access for all world-wide users and for all verification requirements

Data Center based prototyping farms becoming the normal

- Secure remote data centers with limited physical access
  - High system reliability with extensive remote system test and diagnostic routines
- Remote global access
  - System configuration, test deployment and high-speed result data extraction



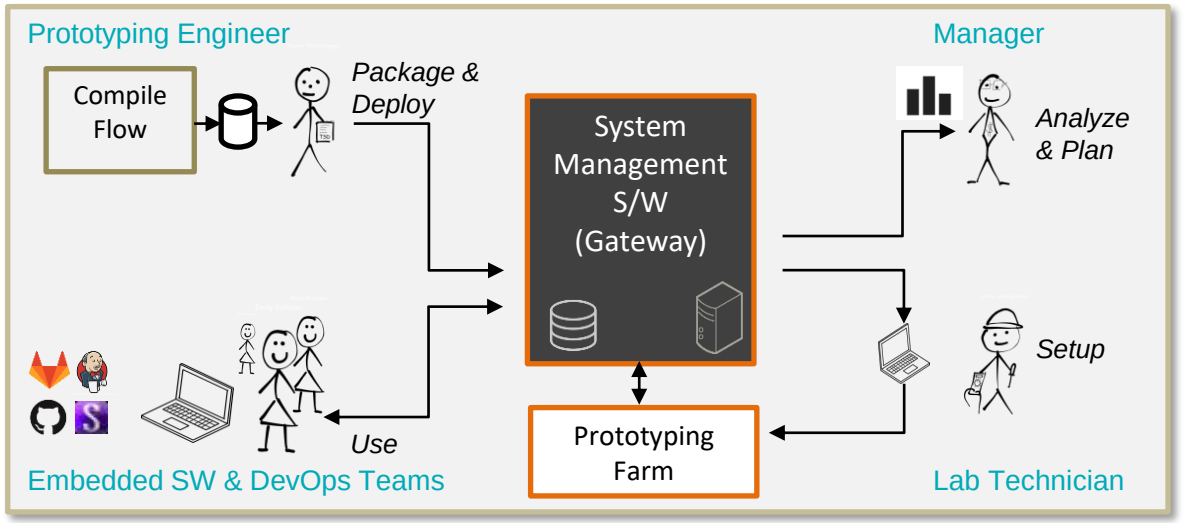


The Dynamics Of Data Center Prototyping

Challenge to monitor and maintain to maximize the return

- Global System Resource Management
  - Resource allocation
    - Queue based system allocation
    - The right system for the right job
  - System Monitoring
    - Which Configurations are bottlenecked
    - Reallocation of systems of maximize return
    - Tracking deep-cycle events over time

- Requires S/W Infrastructure
  - To automate and manage access for all teams
  - To analyze usage to ensure maximum productivity



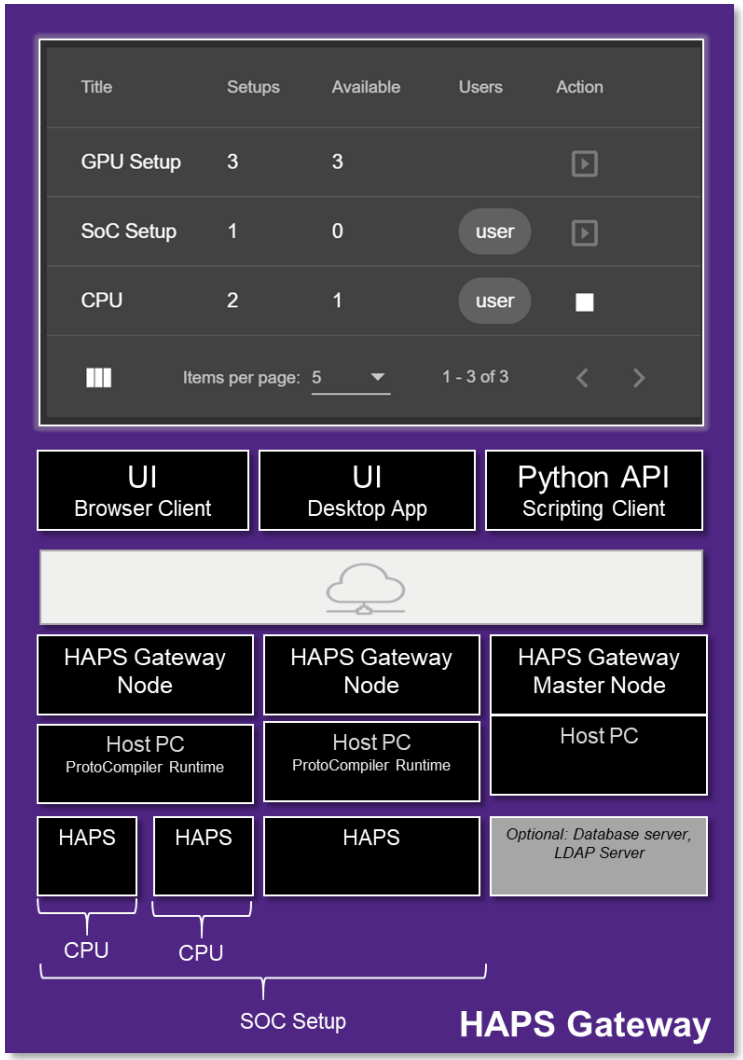
HAPS® Gateway for Prototyping Resource Management

HAPS Gateway

- A SW framework for global control and analysis of HAPS prototyping systems
- Ease of access to all available HAPS prototyping systems
  - From anywhere, at any time, from any location with instant access

Cloud native SW architecture

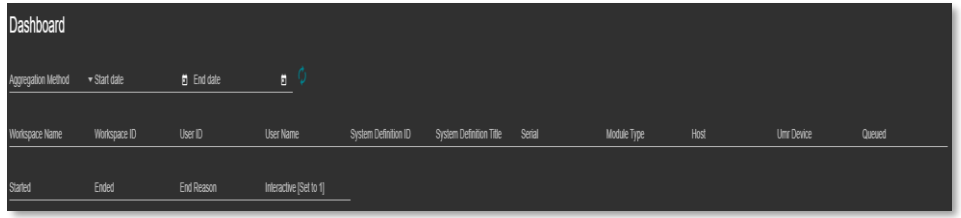
- Secure web APIs, access controls
- Latest Google UI technology and global scripting



HAPS Gateway for Prototyping Resource Management

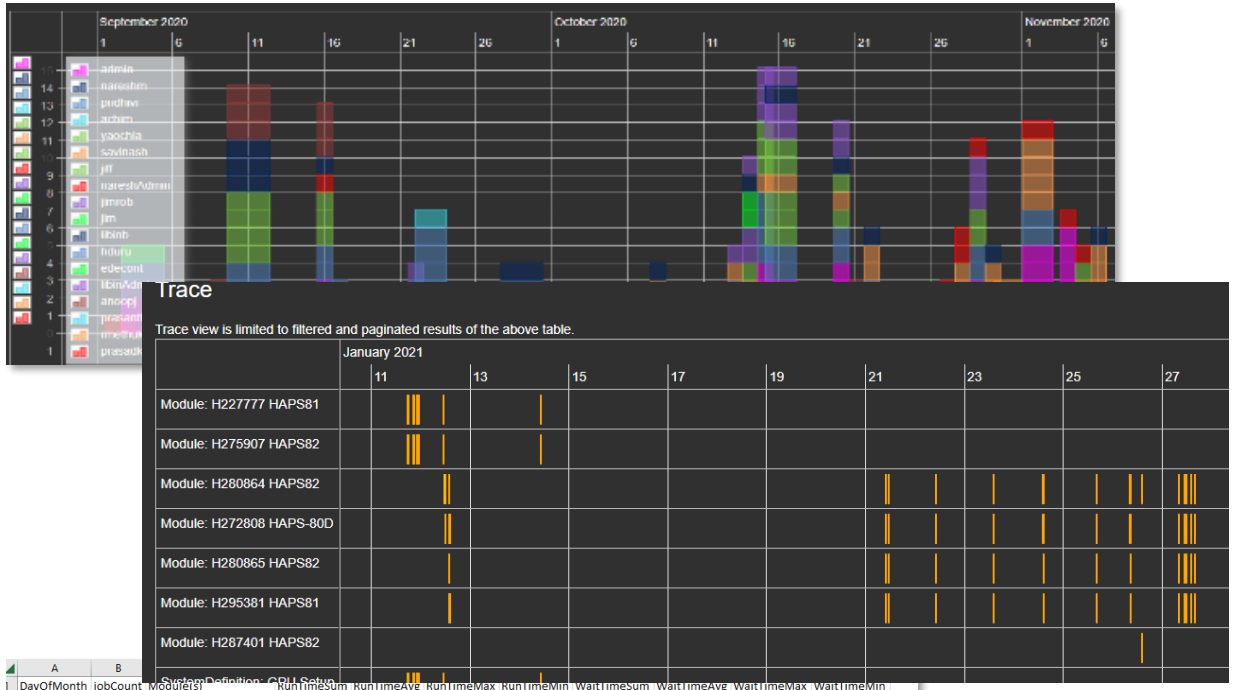
Analytics: Chart view, Trace and Dashboard

Filters



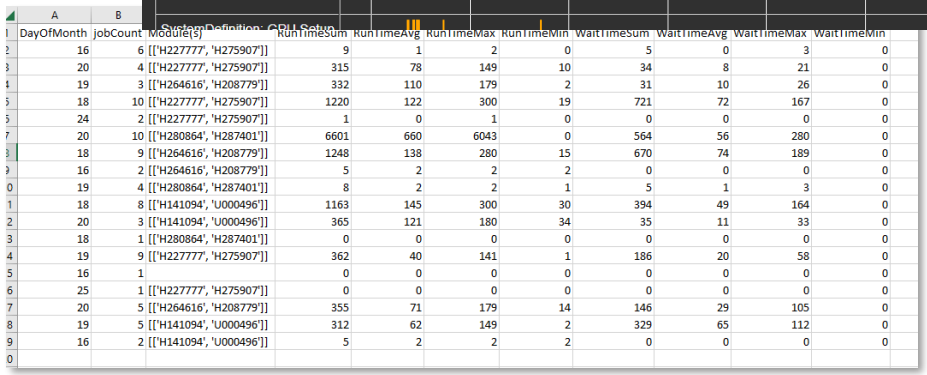
Timeline View

- Max, Min, Average, Sum and Job Count
- Use and wait time per system, module, and user



Database

- MongoDB aggregation exported to CSV
- Copy and paste from UI for refined use



DayOfMonth	jobCount	Module(s)	RunTimeSum	RunTimeAvg	RunTimeMax	RunTimeMin	WaitTimeSum	WaitTimeAvg	WaitTimeMax	WaitTimeMin
16	6	[['H227777', 'H275907']]	9	1	2	0	5	0	3	0
20	4	[['H227777', 'H275907']]	315	78	149	10	34	8	21	0
19	3	[['H264616', 'H208779']]	332	110	179	2	31	10	26	0
18	10	[['H227777', 'H275907']]	1220	122	300	19	721	72	167	0
24	2	[['H227777', 'H275907']]	1	0	1	0	0	0	0	0
20	10	[['H280864', 'H287401']]	6601	660	6043	0	564	56	280	0
18	9	[['H264616', 'H208779']]	1248	138	280	15	670	74	189	0
16	2	[['H264616', 'H208779']]	5	2	2	2	0	0	0	0
19	4	[['H280864', 'H287401']]	8	2	2	1	5	1	3	0
18	8	[['H141094', 'U000496']]	1163	145	300	30	394	49	164	0
20	3	[['H141094', 'U000496']]	365	121	180	34	35	11	33	0
18	1	[['H280864', 'H287401']]	0	0	0	0	0	0	0	0
19	9	[['H227777', 'H275907']]	362	40	141	1	186	20	58	0
16	1		0	0	0	0	0	0	0	0
25	1	[['H227777', 'H275907']]	0	0	0	0	0	0	0	0
20	5	[['H264616', 'H208779']]	355	71	179	14	146	29	105	0
19	5	[['H141094', 'U000496']]	312	62	149	2	329	65	112	0
16	2	[['H141094', 'U000496']]	5	2	2	2	0	0	0	0

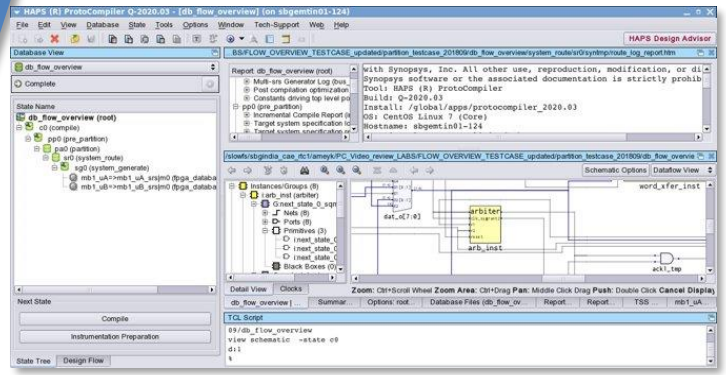
Conclusion: Fast Track RISC-V System Validation

Manage the growing RISC-V SOC Complexity

Scalability and Capacity of Hardware and Software
Modular Approach to Prototype Bring up

RTL Readiness

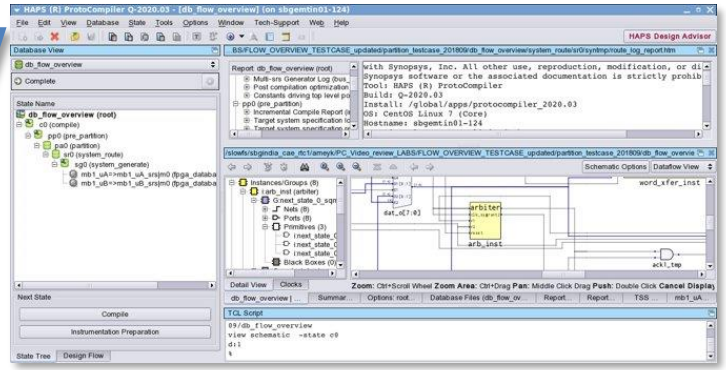
Design the RTL/HDL for Considering Prototype Environment/Platforms



Conclusion: Fast Track RISC-V System Validation

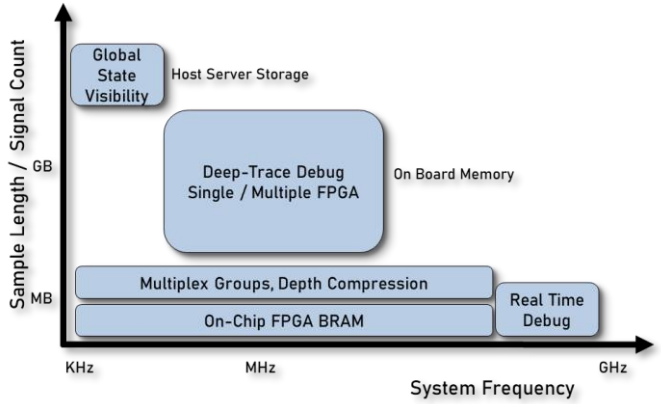
Achieve High Performance

Planning and Timing Awareness in both Hardware and Software tools



Deep Cycle Debug Management

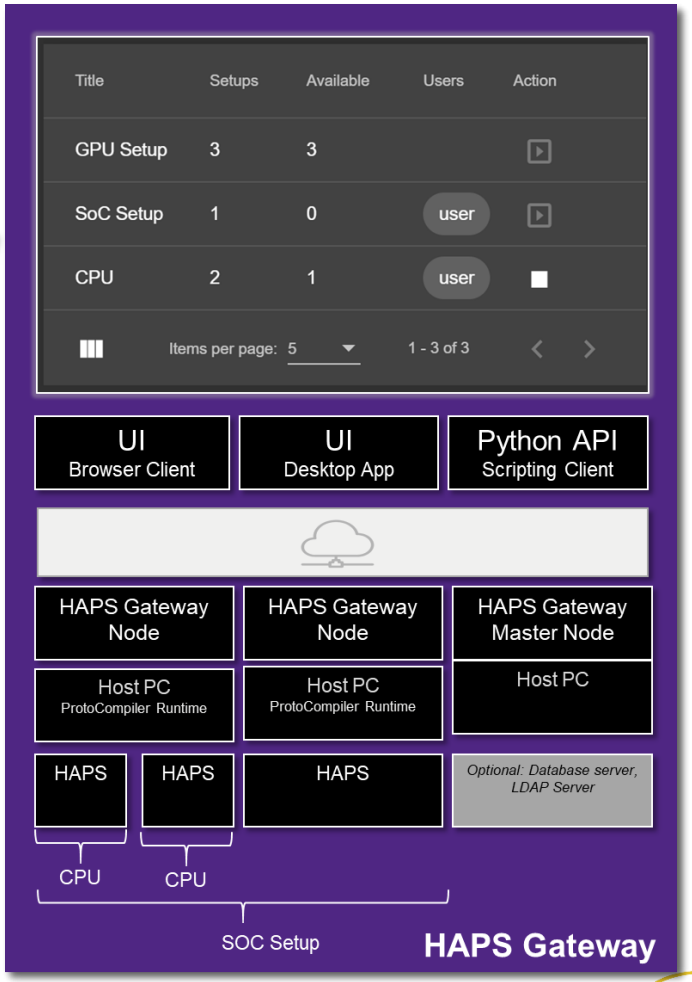
Capacity and at Speed Debug



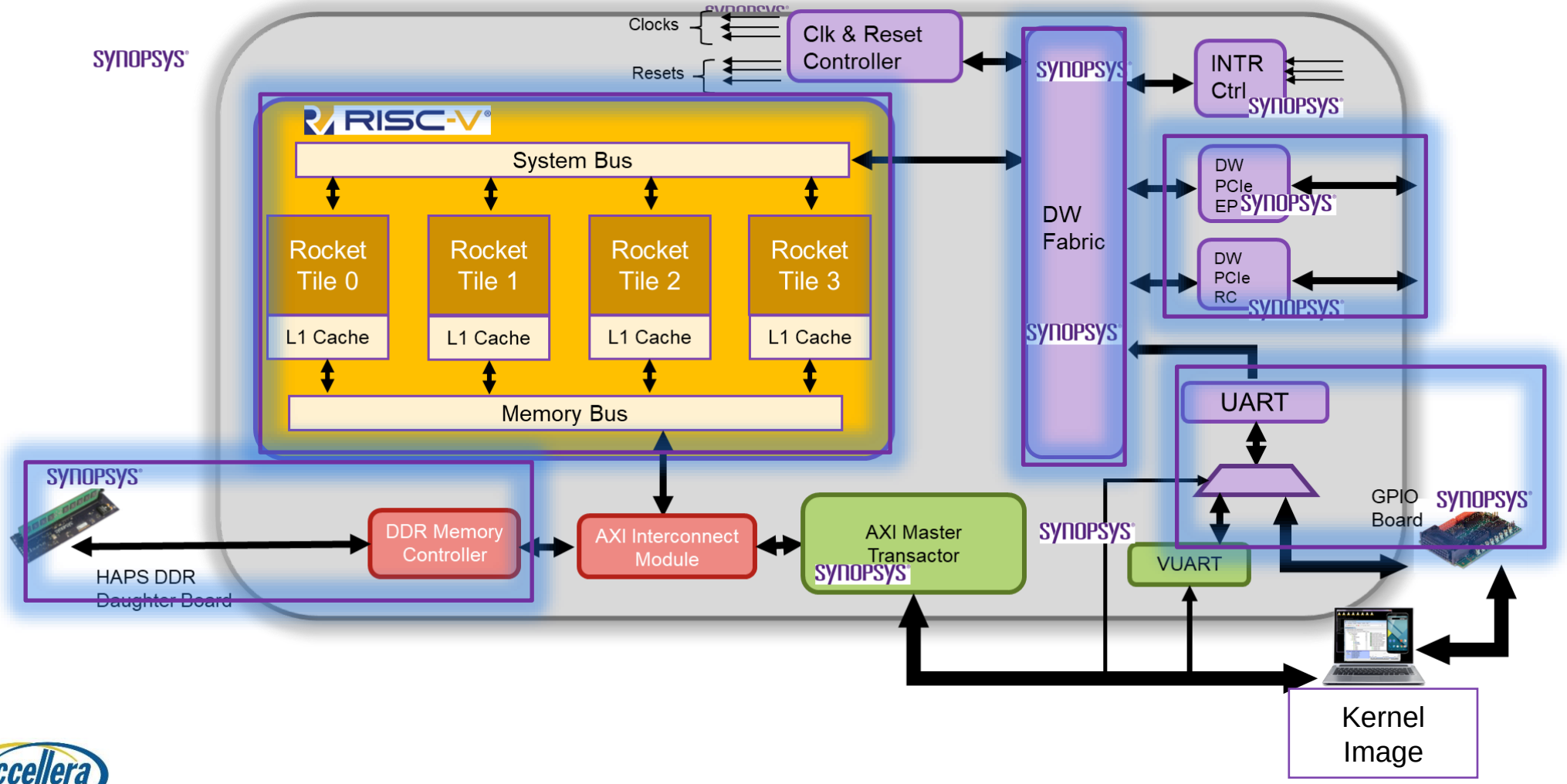
Conclusion: Fast Track RISC-V System Validation

Dynamics Of Data Center Prototyping

– Effective use of Hardware Platforms



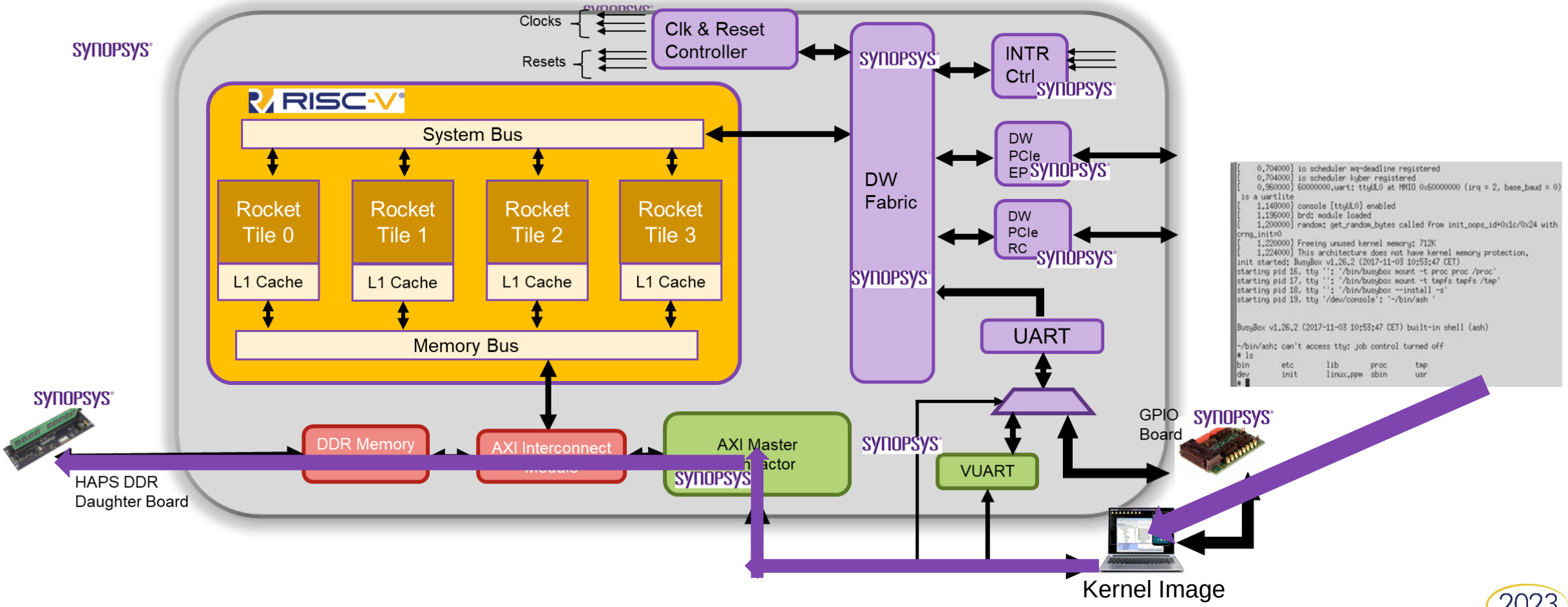
Demo : RISC-V System Validation with HAPS

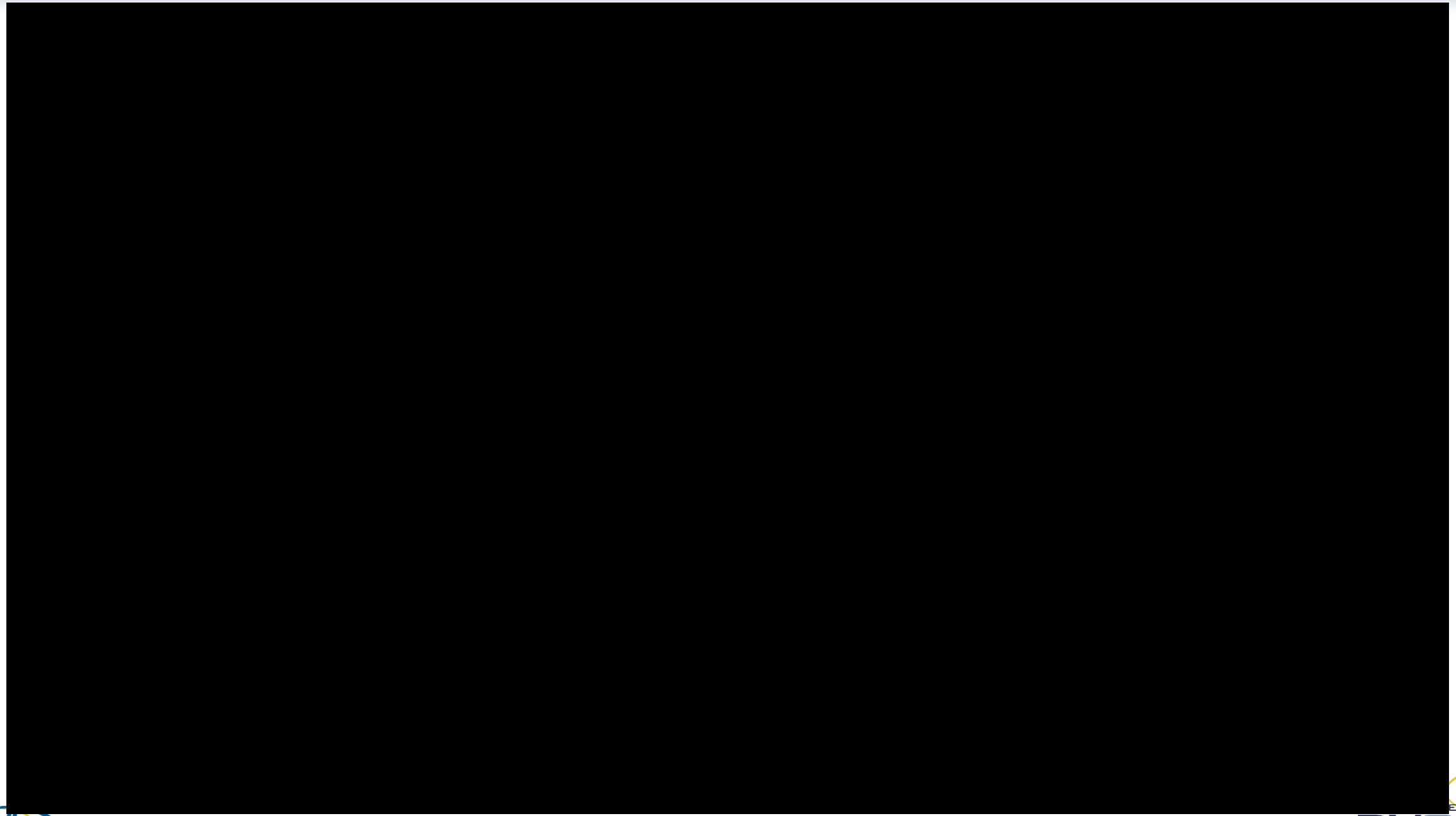


accellera SYSTEMS INITIATIVE™

2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Demo : RISC-V System Validation with HAPS





Thank You