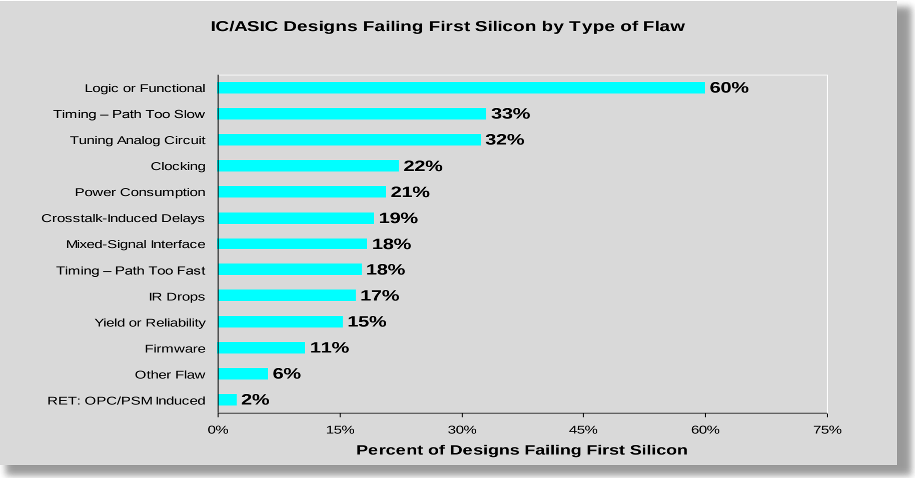
Confirma™

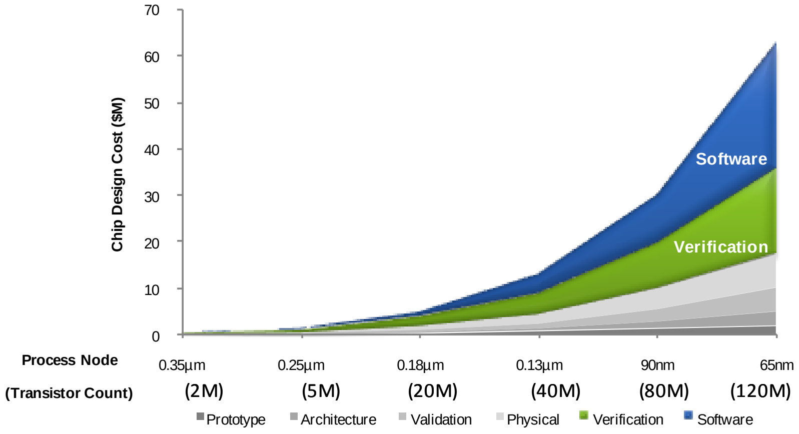
The Next Era of Rapid Prototyping

Verification Is Not Getting Any Easier

- **The Dominant Cause For Silicon Re-spins**
  - Functional errors
  - Bug count increasing exponentially with design size
  - Need to test with live stimulus



- **Software Content Is Increasing Dramatically**
  - Sufficient performance is key

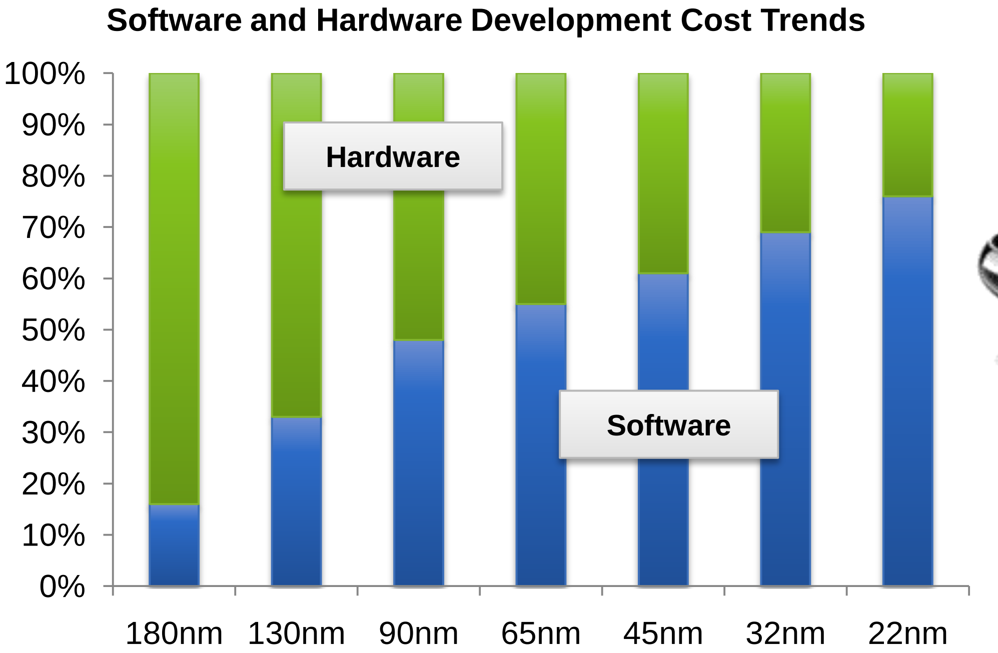


Source: IBS,2008

"Our biggest challenge in SoC development is software."

"Software cost is over 60% of the total cost of system design."

Software-Differentiated Hardware Era



Source: IBS, 2008

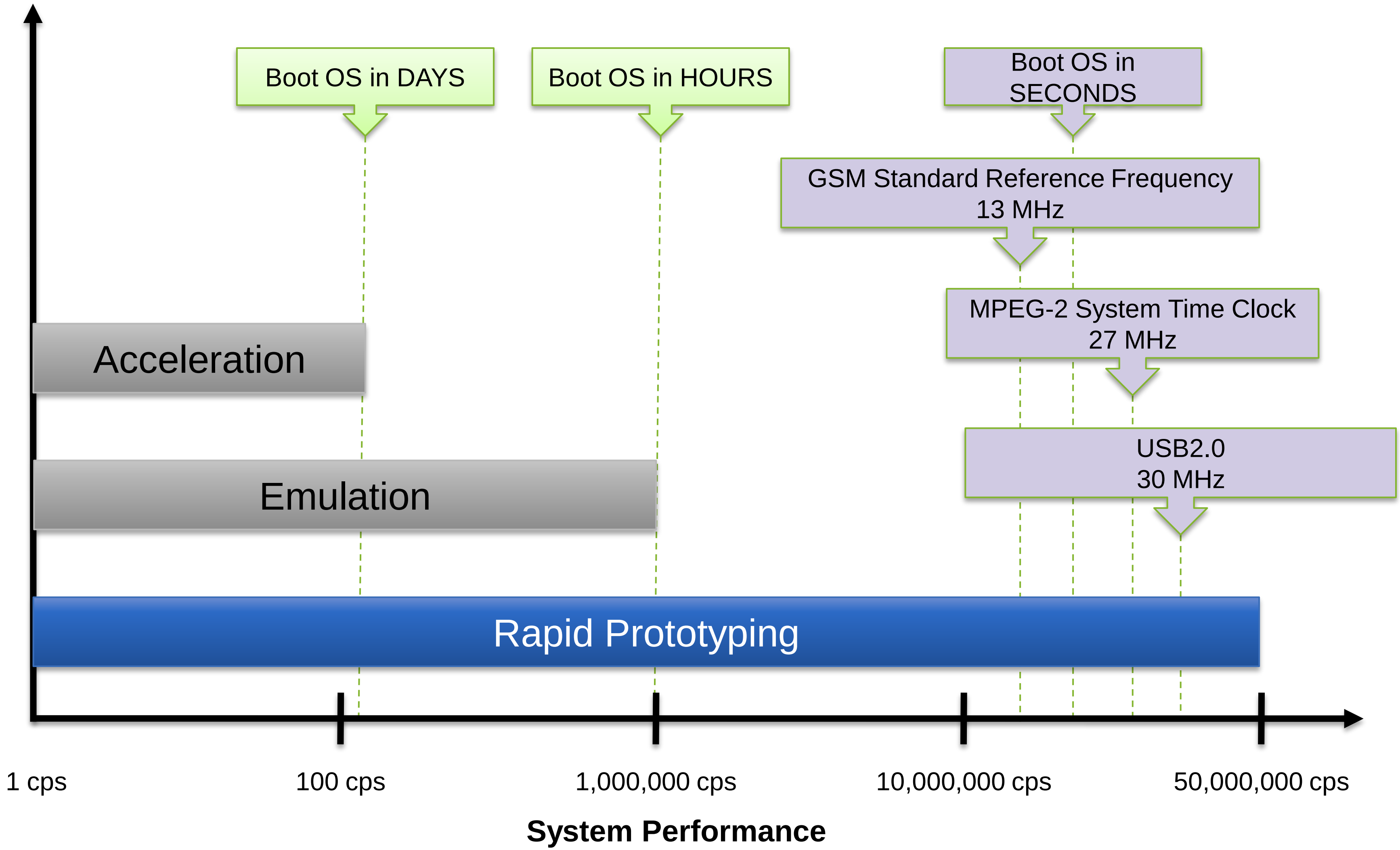
“Phone differentiation used to be about radios and antennas and things like that. We think, going forward, ***the phone of the future will be differentiated by software***.”

Steve Jobs
CEO, Apple

http://online.wsj.com/article/SB121842341491928977.html

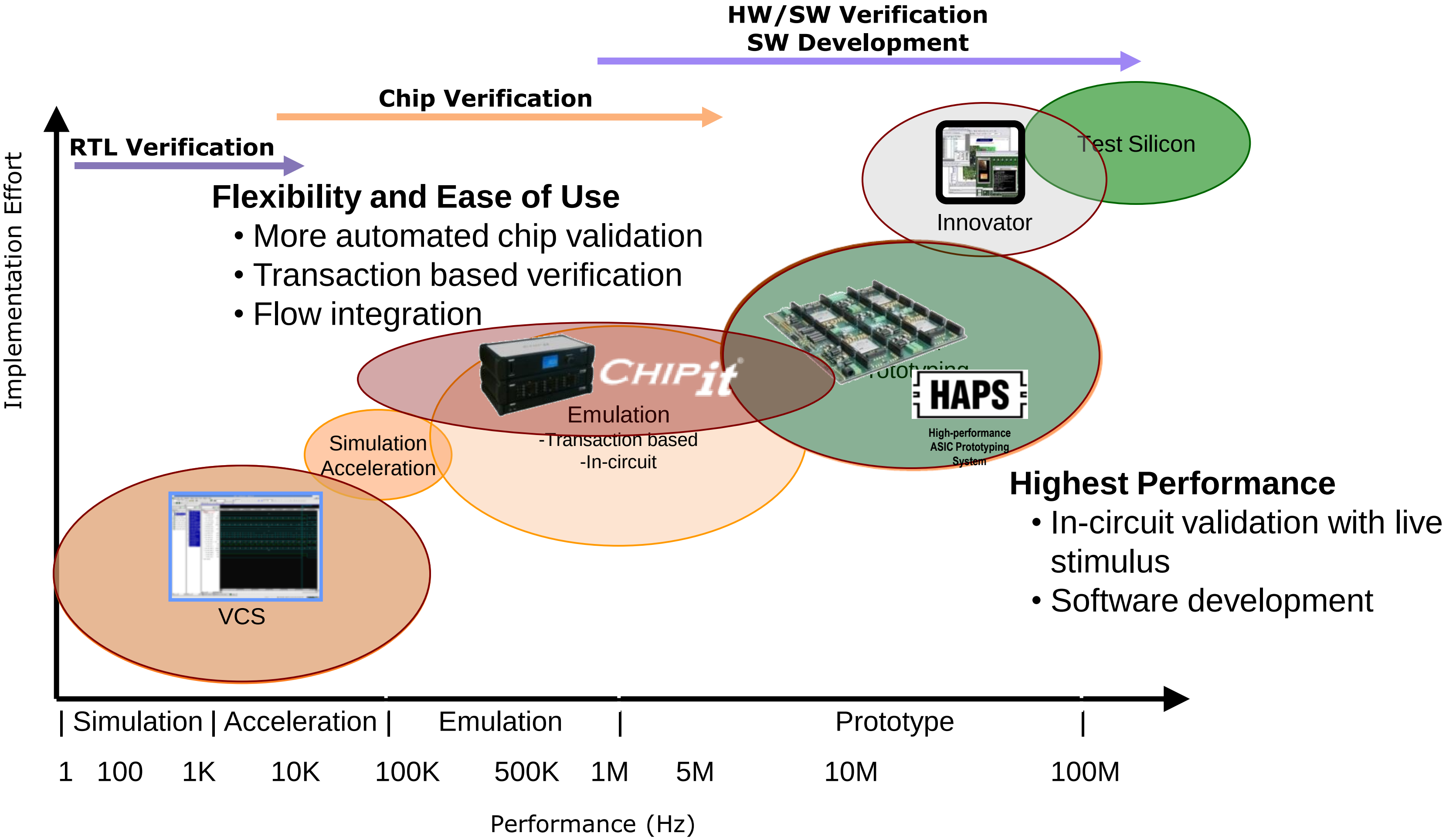
Performance Matters

Enabling System Validation And Software Development



SYNOPSYS® Predictable Success

Confirma Product Positioning



Flexibility and Ease of Use

- More automated chip validation
- Transaction based verification
- Flow integration

Highest Performance

- In-circuit validation with live stimulus
- Software development

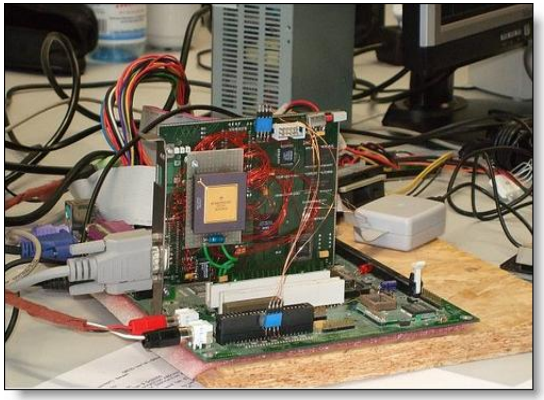
Confirma

Defining The Next Era In Rapid Prototyping

Confirma

Complete Prototyping

Traditional Prototype



+ Compact
+ Affordable
+ Fast

- Manual
- Difficult debug

Traditional Emulator

- Big
- Expensive
- Slow

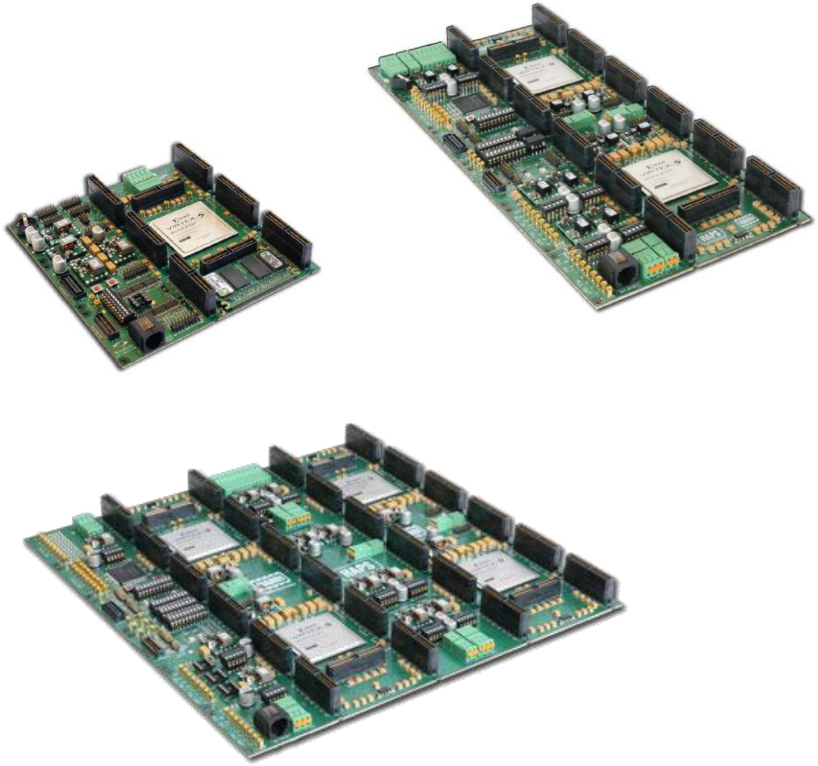
+ Automated
+ Easy debug

Prototyping Made Easy

- **System Solutions**
  - Single point for hardware and software support
  - Tightly Integrated Confirma Platform
- **High Performance**
- **Flexible Hardware**
  - Configurable
  - Over 30 daughter boards
  - Reuse for multiple projects
- **Over 500 companies worldwide**
- **Over 2000 boards shipped**

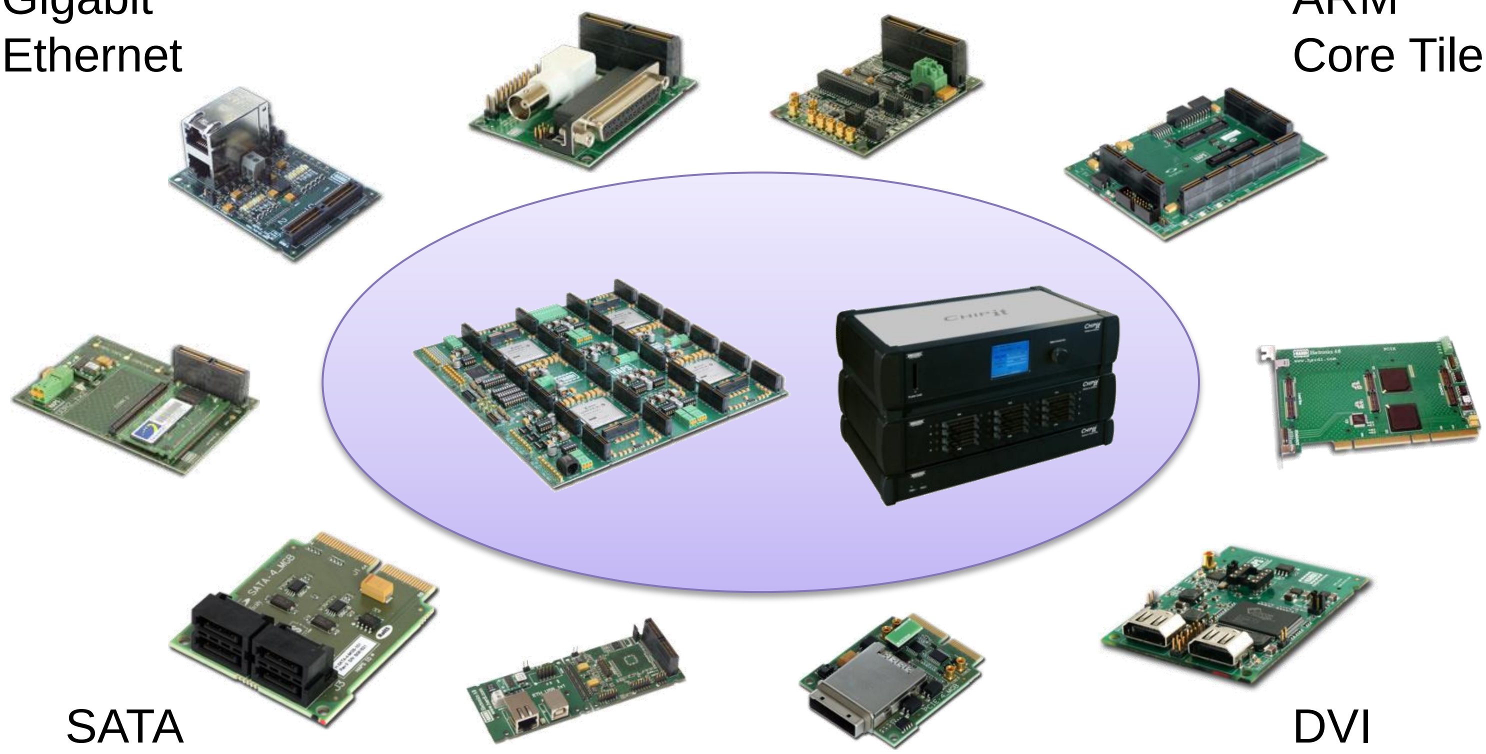
HAPS

High-performance ASIC Prototyping System



We Offer Real World IO

Common boards support both HAPS and CHIPit



Gigabit Ethernet

DVB

ADC / DAC

ARM Core Tile

DDR2

PCI-X

SATA

USB 2.0

PCIe

DVI

At-speed Connections to Real-world Interfaces Enable Realistic Validation

HAPS-60 Family

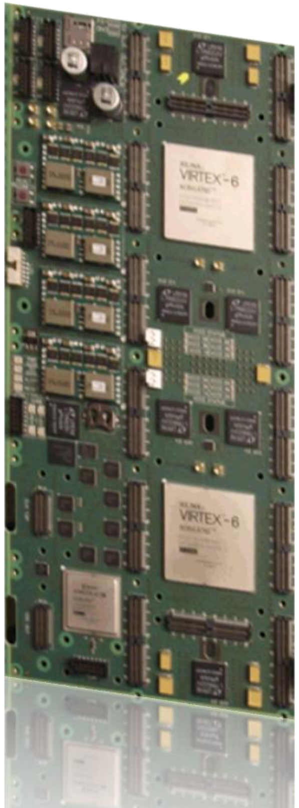
CAPACITY

HAPS-61

Up to 4.5M gates

HAPS-64

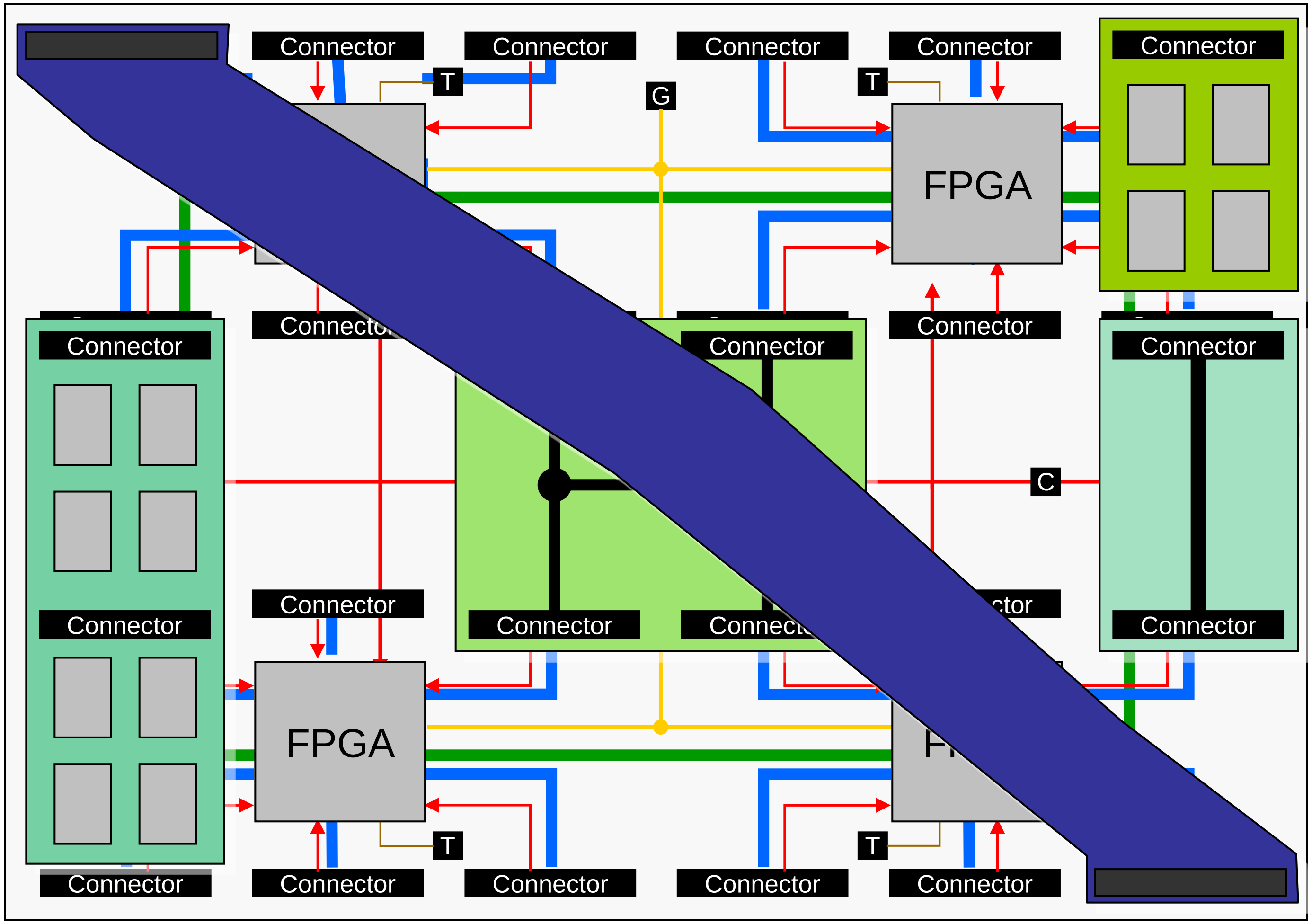
Up to 18M gates



HAPS-62

Up to 9M gates

HAPS Concept



I/Os or inter-FPGA connections

Fixed inter-FPGA connections

Local clocks

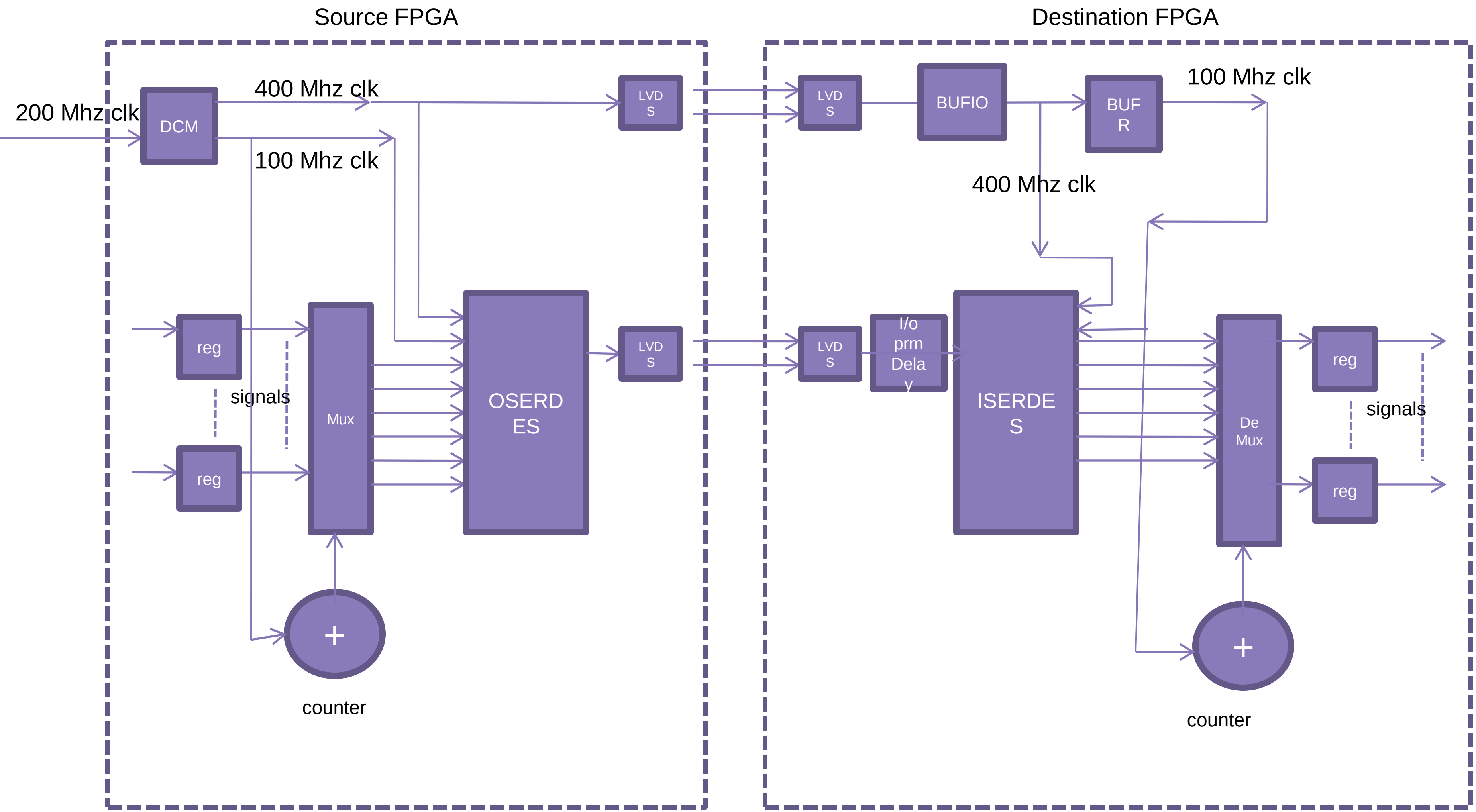
C Global clocks

G Global I/Os

T Test signals

The size of a daughter board is strictly specified. There are no "wasted" connectors on the motherboard.

HSTDM (High Speed TDM)

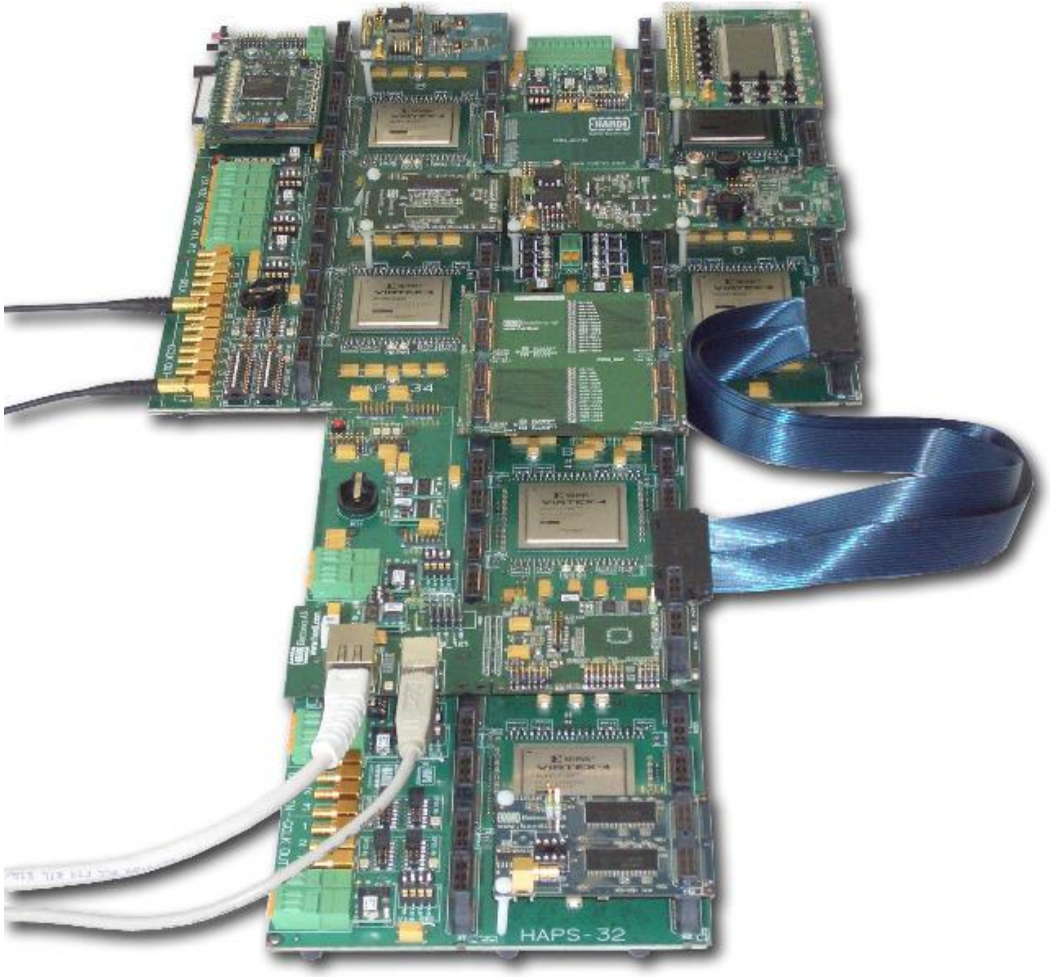


The Solution To Rent's Rule

- Number of pins = t * (Design Size) ^ p
  - ASIC 1.2 M gates – 253 pins
  - 2 FPGAs – 610K gates/FPGA – 322 pins

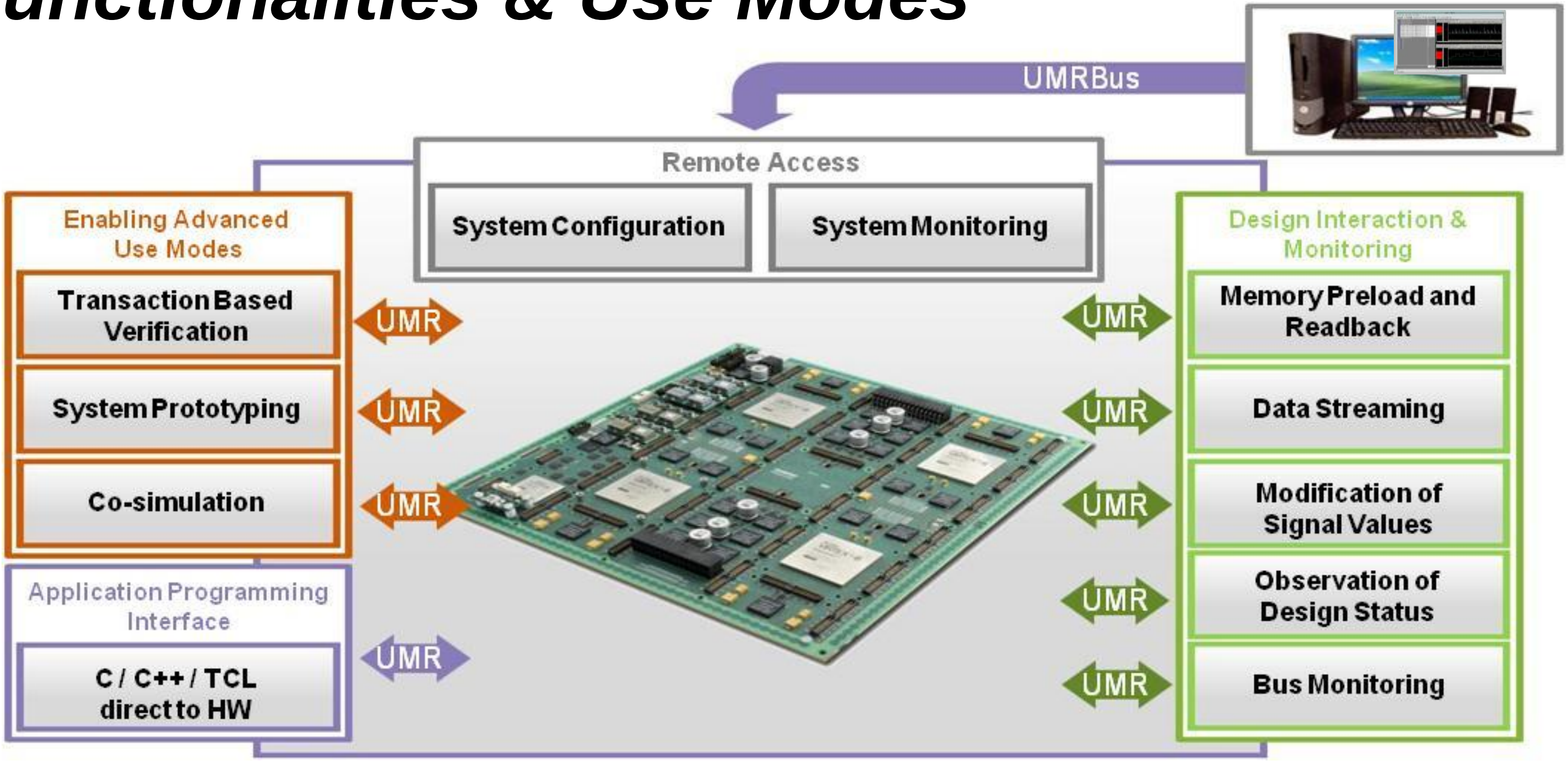
- Each FPGA on a HAPS 6X board has a 4.5 M gate capacity – No partitioning needed (For this case)!
- < 2800 flexible I/Os available on HAPS 64 boards for either ASIC I/O or interconnect.
- High Speed Pin Multiplexing using Xilinx ISERDES/OSERDES available if you need it!

HAPS Configuration – An example



Universal Multi-Resource Bus (UMRbus)

Functionalities & Use Modes


UMRBus
Remote Access
System Configuration
System Monitoring
Enabling Advanced Use Modes
Transaction Based Verification
System Prototyping
Co-simulation
Application Programming Interface
C / C++ / TCL direct to HW
Design Interaction & Monitoring
Memory Preload and Readback
Data Streaming
Modification of Signal Values
Observation of Design Status
Bus Monitoring
UMR
UMR
UMR
UMR
UMR
UMR
UMR
UMR
UMR

FAST UMRBus

What It Is

- High-performance, low-latency communication bus
- Connections to every FPGA, memories, registers, etc.

Customer Benefits

- Remote prototype management
- Application-level programming
- Co-simulation
- Transaction-based verification

Co-simulation

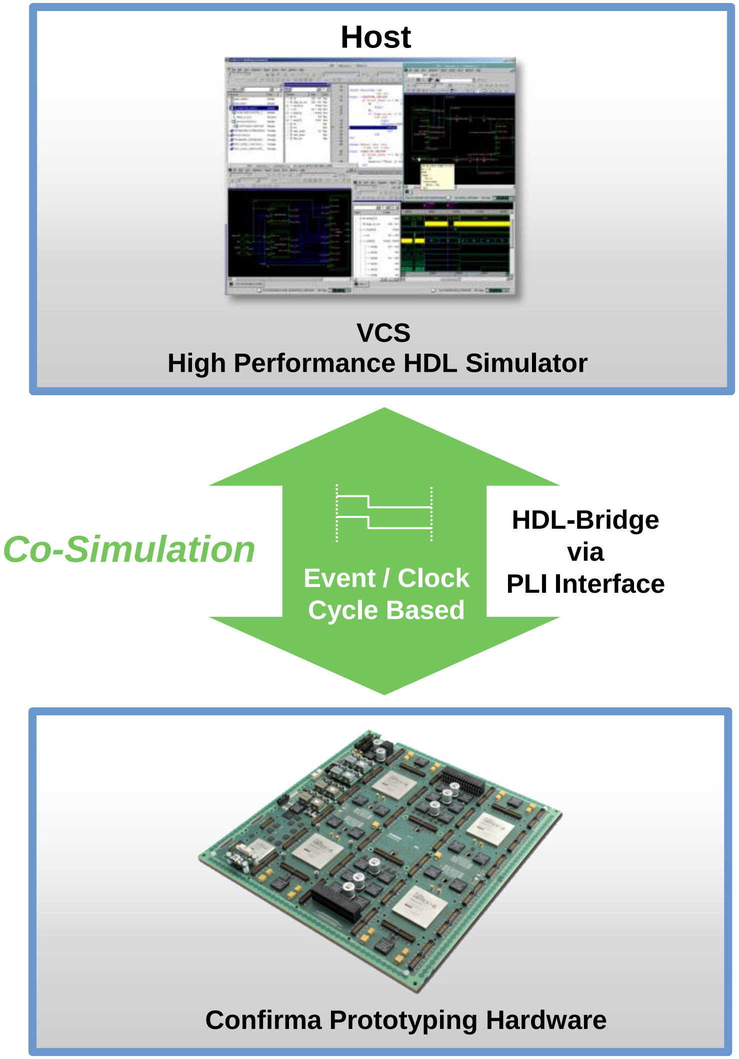
CO-SIMULATION

HAPS Features

- Supports VCS (and other simulators)
- Off-line debug with VCS
- Interactive debug with Identify Pro
- Built on UMRBus

Customer Benefits

- Easy transition from simulation to prototype
- Re-use existing regression tests
- Validate prototype implementation
- Continue verification in hardware



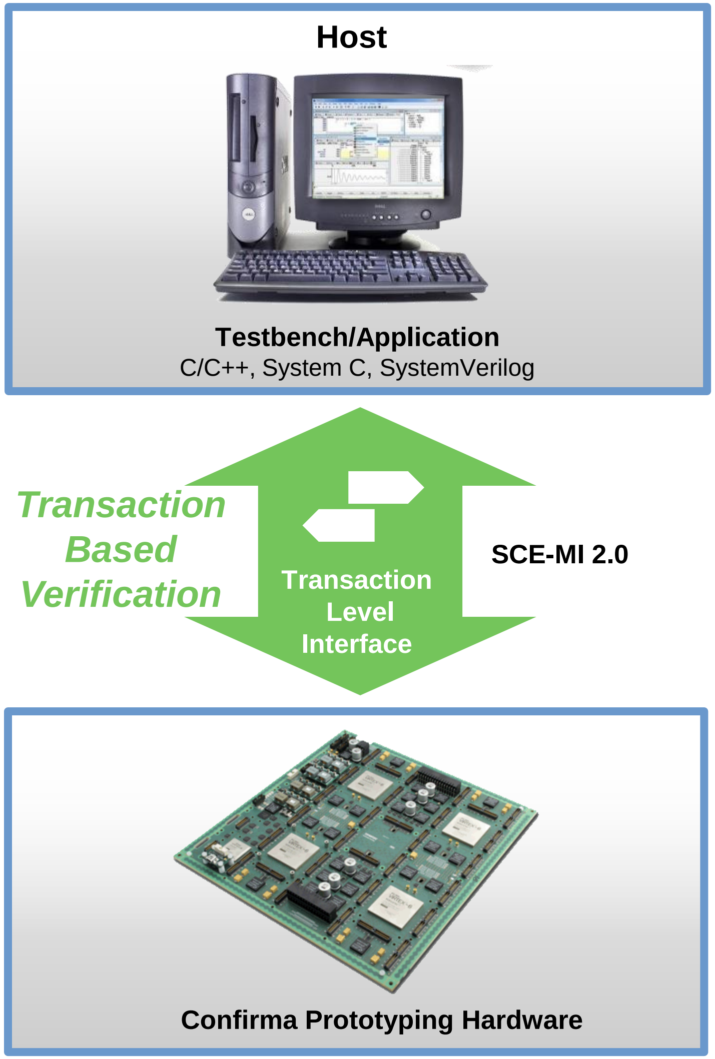
Transaction-based Verification

HAPS Features

- Industry-standard SCE-MI 2.0 support
- Uses transaction-based testbenches (C/C++/VCS/Innovator)
- VMM-HAL support
- Built on UMRBus

Customer Benefits

- Up to 10,000x faster than simulation*
- Interface to live devices
- Re-use existing regression tests
- Accurate modeling
- Leverage prototype hardware earlier



Host

Testbench/Application

C/C++, System C, SystemVerilog

Transaction Based Verification

Transaction Level Interface

SCE-MI 2.0

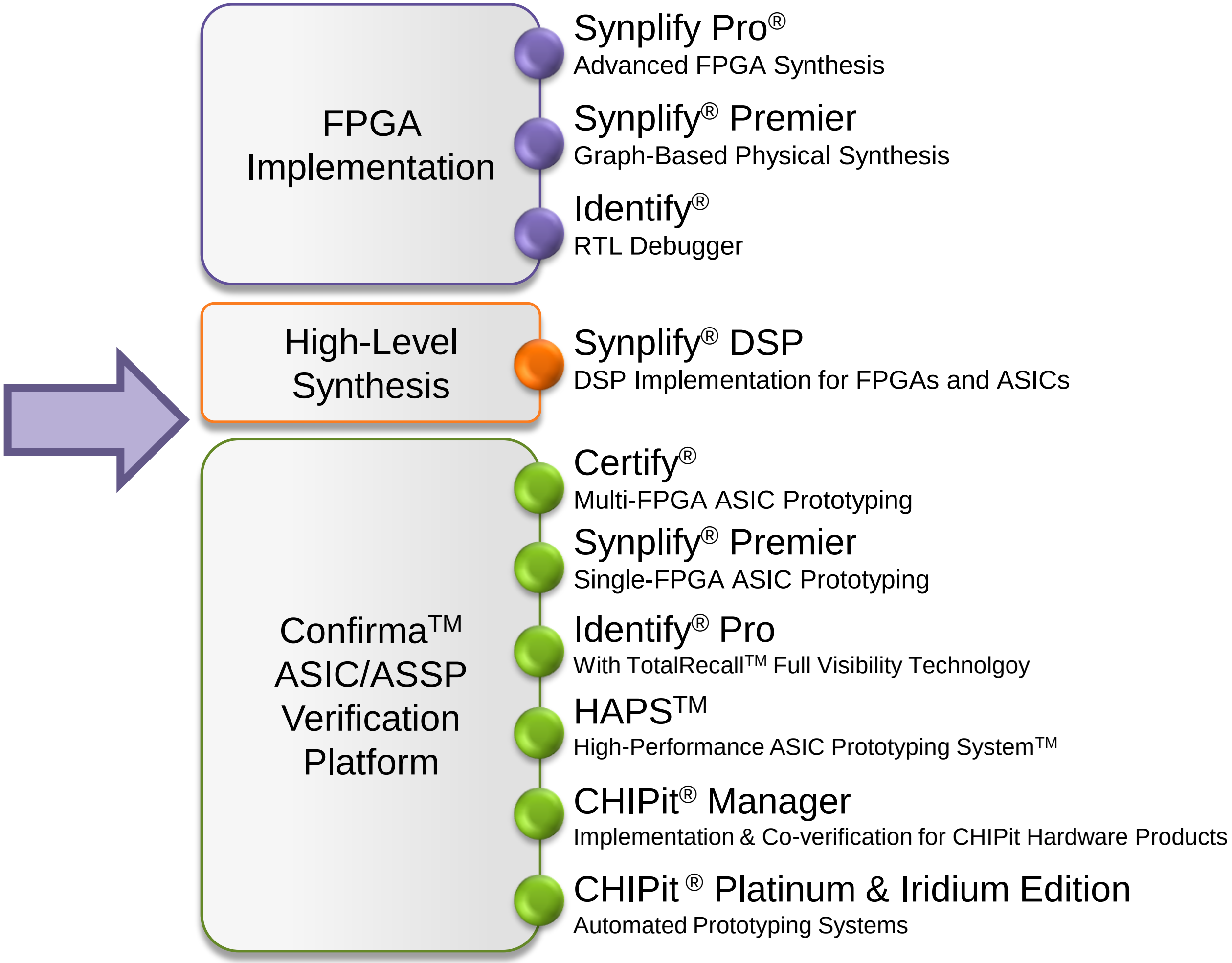
Confirma Prototyping Hardware

* Appropriate testbench required

TBV

Solutions Overview

Provide industry leading FPGA Implementation, High Level Design, and Hardware Assisted Verification solutions



FPGA Implementation

- Synplify Pro®
  Advanced FPGA Synthesis
- Synplify® Premier
  Graph-Based Physical Synthesis
- Identify®
  RTL Debugger

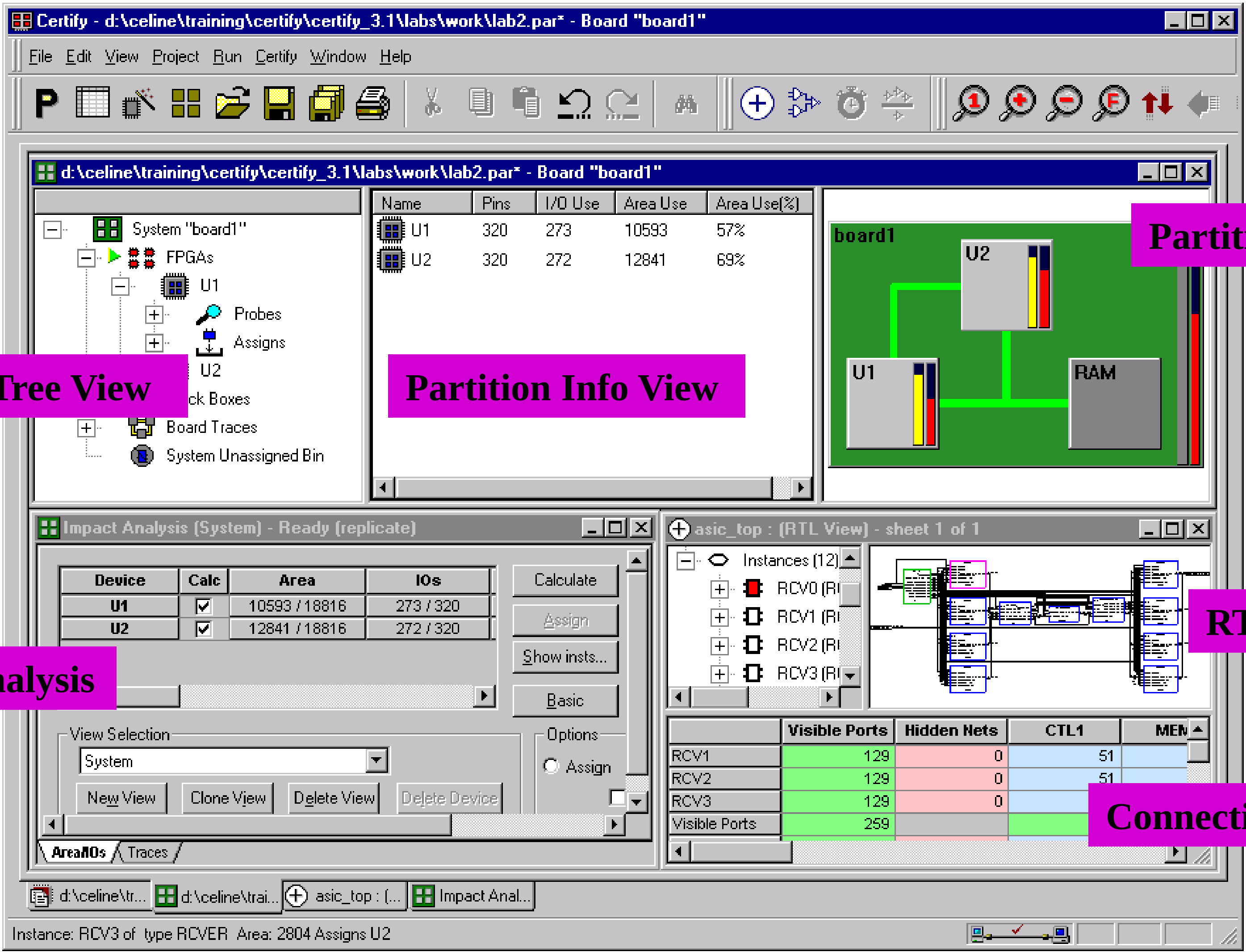
High-Level Synthesis

- Synplify® DSP
  DSP Implementation for FPGAs and ASICs

Confirma™ ASIC/ASSP Verification Platform

- Certify®
  Multi-FPGA ASIC Prototyping
- Synplify® Premier
  Single-FPGA ASIC Prototyping
- Identify® Pro
  With TotalRecall™ Full Visibility Technolgoy
- HAPS™
  High-Performance ASIC Prototyping System™
- CHIPit® Manager
  Implementation & Co-verification for CHIPit Hardware Products
- CHIPit ® Platinum & Iridium Edition
  Automated Prototyping Systems

Certify Multi-FPGA Partitioning



Name	Pins	I/O Use	Area Use	Area Use(%)
U1	320	273	10593	57%
U2	320	272	12841	69%

Device	Calc	Area	IOs
U1	☑	10593 / 18816	273 / 320
U2	☑	12841 / 18816	272 / 320

	Visible Ports	Hidden Nets	CTL1	MEM
RCV1	129	0	51	
RCV2	129	0	51	
RCV3	129	0		
Visible Ports	259			

Instance: RCV3 of type RCVER Area: 2804 Assigns U2

Partition View

Partition Tree View

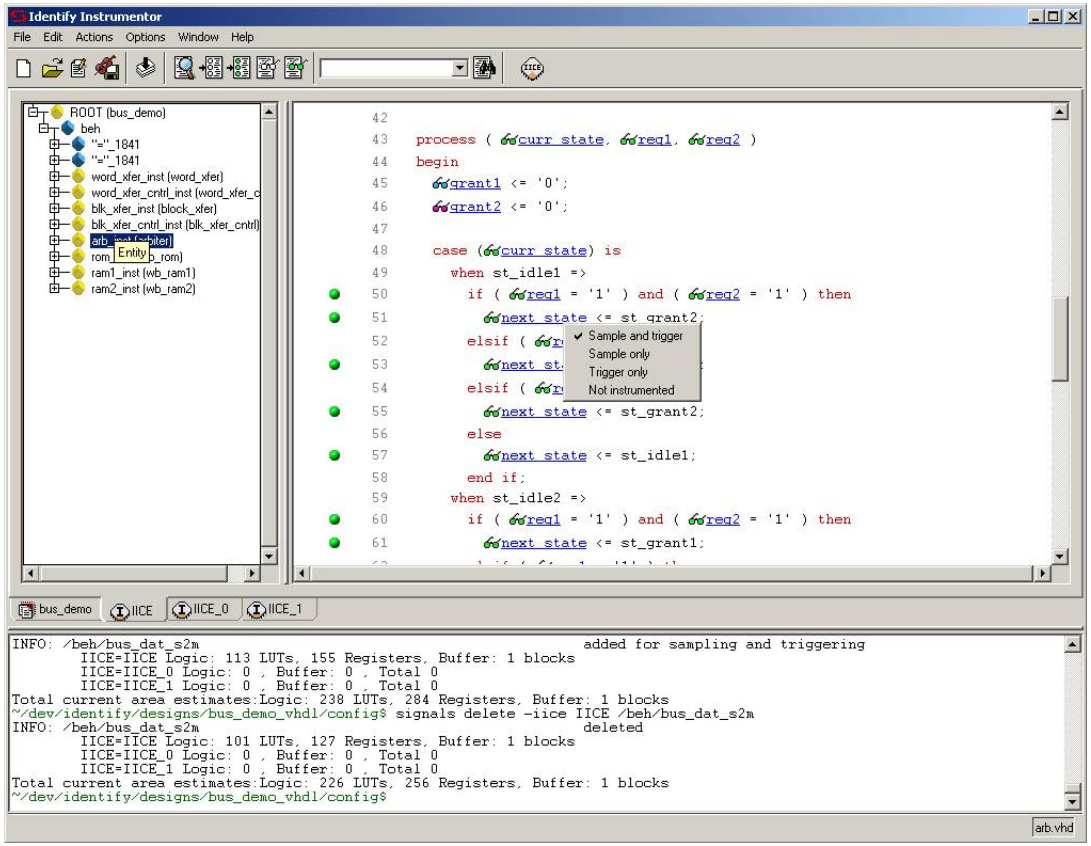
Partition Info View

RTL View

Impact Analysis

Connectivity Matrix

Identify Instrumentor



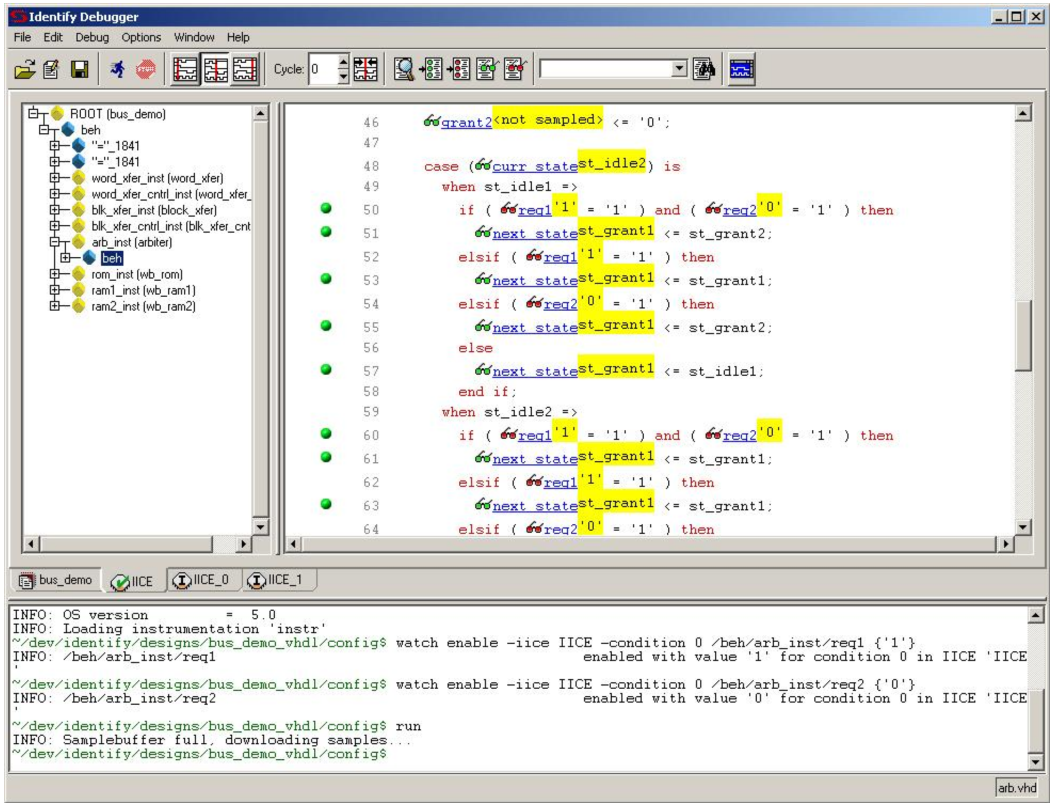
```
process ( curr_state, req1, req2 )
begin
  grant1 <= '0';
  grant2 <= '0';

  case (curr_state) is
    when st_idle1 =>
      if ( req1 = '1' ) and ( req2 = '1' ) then
        next_state <= st_grant2;
      elsif ( r
        next_st
      elsif ( r
        next_state <= st_grant2;
      else
        next_state <= st_idle1;
      end if;
    when st_idle2 =>
      if ( req1 = '1' ) and ( req2 = '1' ) then
        next_state <= st_grant1;
```

```
INFO: /beh/bus_dat_s2m                                          added for sampling and triggering
       IICE=IICE Logic: 113 LUTs, 155 Registers, Buffer: 1 blocks
       IICE=IICE_0 Logic: 0 , Buffer: 0 , Total 0
       IICE=IICE_1 Logic: 0 , Buffer: 0 , Total 0
Total current area estimates:Logic: 238 LUTs, 284 Registers, Buffer: 1 blocks
~/dev/identify/designs/bus_demo_vhdl/config$ signals delete -iice IICE /beh/bus_dat_s2m
INFO: /beh/bus_dat_s2m                                          deleted
       IICE=IICE Logic: 101 LUTs, 127 Registers, Buffer: 1 blocks
       IICE=IICE_0 Logic: 0 , Buffer: 0 , Total 0
       IICE=IICE_1 Logic: 0 , Buffer: 0 , Total 0
Total current area estimates:Logic: 226 LUTs, 256 Registers, Buffer: 1 blocks
~/dev/identify/designs/bus_demo_vhdl/config$
```

- Automatically displays signals and branches
- Control sampling & triggering on each node
- State Machine Triggering
- Provides area estimate of debug resources
- Automate with TCL scripts

Identify Debugger



```
    grant2<not sampled> <= '0';

    case (curr_statest_idle2) is
      when st_idle1 =>
        if ( req1'1' = '1' ) and ( req2'0' = '1' ) then
          next_statest_grant1 <= st_grant2;
        elsif ( req1'1' = '1' ) then
          next_statest_grant1 <= st_grant1;
        elsif ( req2'0' = '1' ) then
          next_statest_grant1 <= st_grant2;
        else
          next_statest_grant1 <= st_idle1;
        end if;
      when st_idle2 =>
        if ( req1'1' = '1' ) and ( req2'0' = '1' ) then
          next_statest_grant1 <= st_grant1;
        elsif ( req1'1' = '1' ) then
          next_statest_grant1 <= st_grant1;
        elsif ( req2'0' = '1' ) then
```

```
INFO: OS version           =  5.0
INFO: Loading instrumentation 'instr'
~/dev/identify/designs/bus_demo_vhdl/config$ watch enable -iice IICE -condition 0 /beh/arb_inst/req1 {'1'}
INFO: /beh/arb_inst/req1                                  enabled with value '1' for condition 0 in IICE 'IICE
'
~/dev/identify/designs/bus_demo_vhdl/config$ watch enable -iice IICE -condition 0 /beh/arb_inst/req2 {'0'}
INFO: /beh/arb_inst/req2                                  enabled with value '0' for condition 0 in IICE 'IICE
'
~/dev/identify/designs/bus_demo_vhdl/config$ run
INFO: Samplebuffer full, downloading samples...
~/dev/identify/designs/bus_demo_vhdl/config$
```

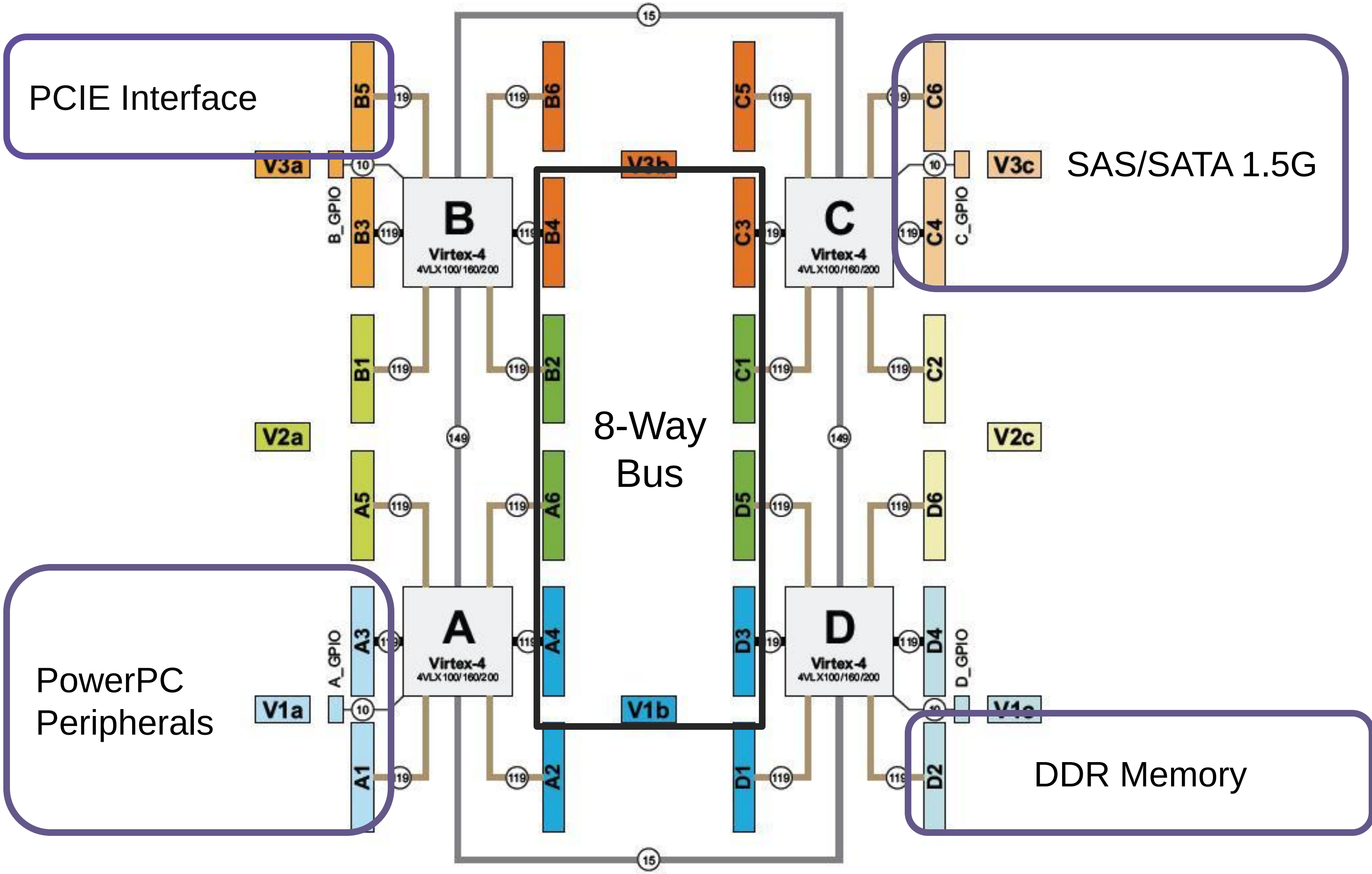
- Dynamically sets trigger values
- Displays captured data symbolically
- Tabbed view of multiple clock regions
- Multiple debuggers per scan chain supported
- Full TCL scripting

Accelerating Hardware Prototype Development

Chuck Cruse

Prototype/Emulation Team Lead

LSI Prototyping System – HAPS 34



PCIE Interface

SAS/SATA 1.5G

8-Way Bus

PowerPC Peripherals

DDR Memory

Hardware Prototypes Provide

- “Real World” exercise of RTL
- Hardware/Software integration vehicle
- Detection and debug of subtle system-level “Real Time” problems
- Specification verification amongst various compatible products
- Customer demo

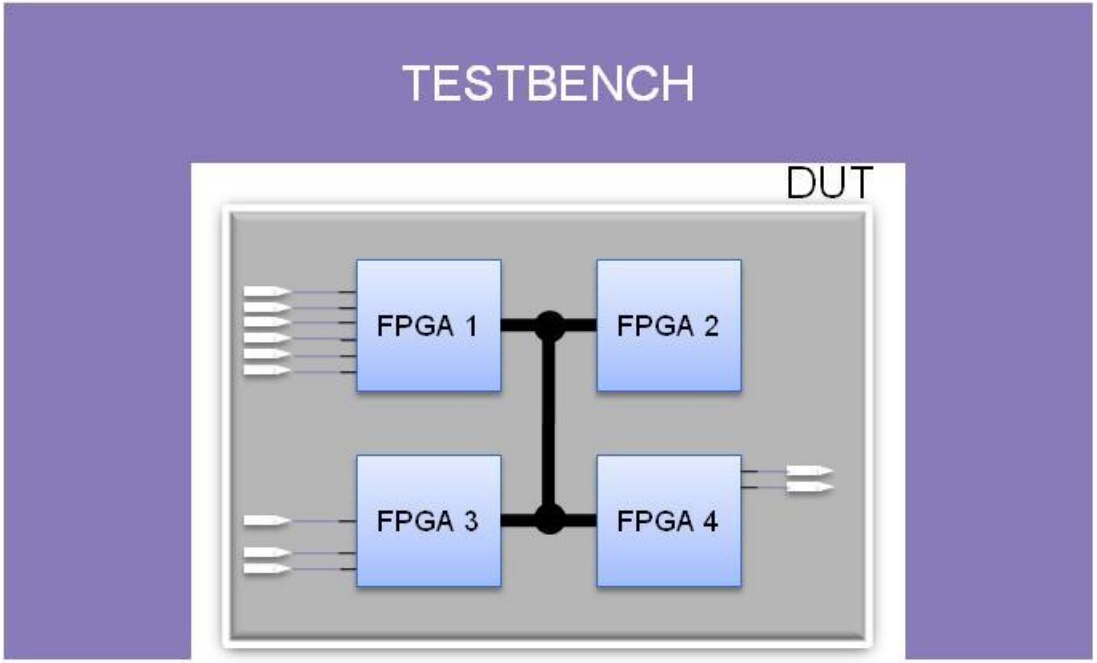
Issues for the Hardware Prototype

- Must hit a narrow window of opportunity
- Delays in the development of the prototype reduce effectiveness
- Visibility of internal signals, while improving, remains a challenge
- Quick turnaround of incremental changes improving, but could be better

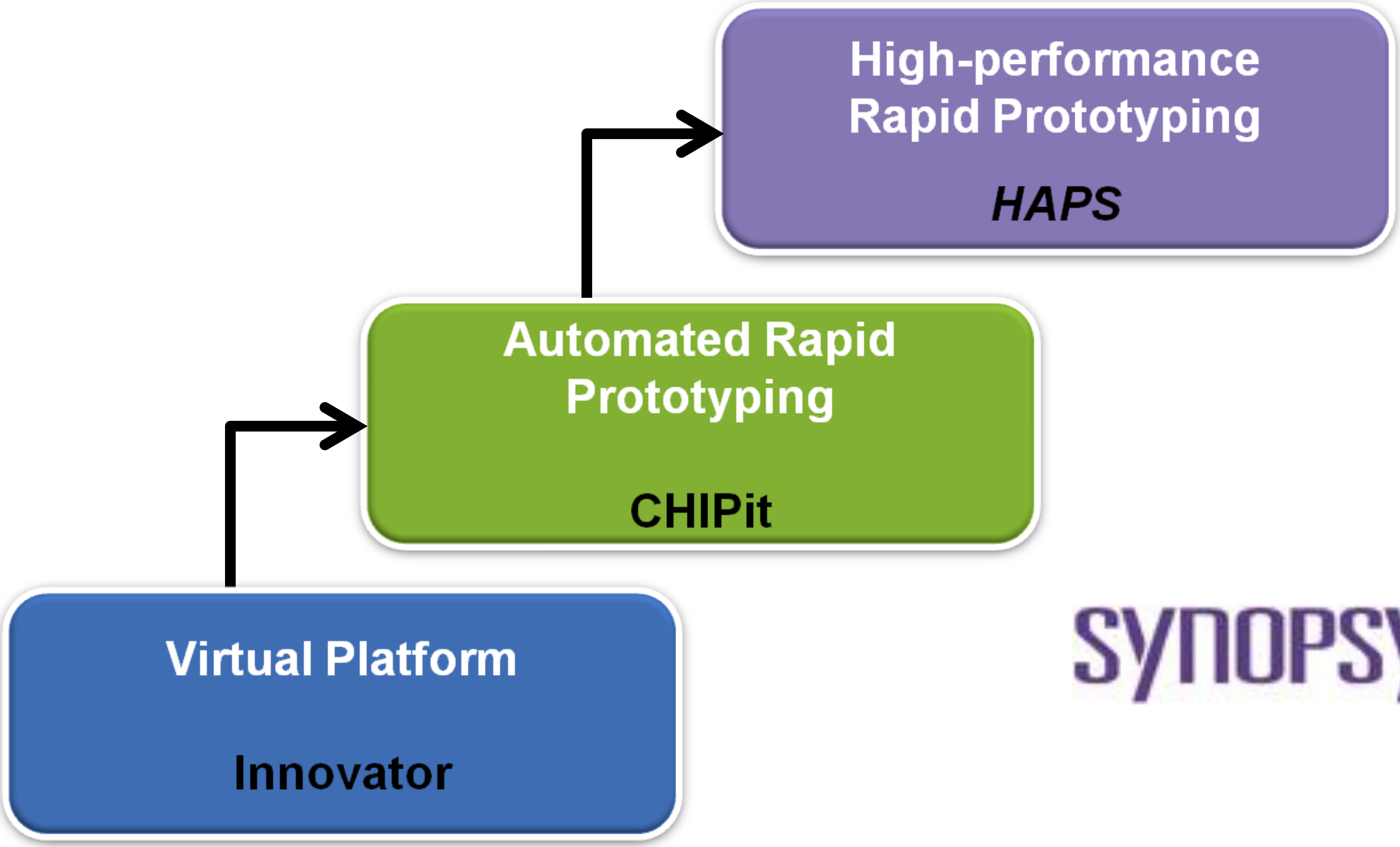
Moving Up the Food-Chain

- In the past, FPGA-specific RTL has been simulated in order to demonstrate basic functionality before downloading bitstreams into the lab.
- For the future, the hope is that this approach could be expanded to include Virtual Platforms

Design Example



Future Feasibility Study



High-performance Rapid Prototyping

HAPS

Automated Rapid Prototyping

CHIPit

Virtual Platform

Innovator

SYNOPSYS®

Additional Product Development Goals

- Retain the benefits of software simulation
  - Cycle-accuracy will suffer, but the hardware prototype will catch these issues
- Begin hardware/software integration very early in the development process
- Hardware/software effort will be more transparent when “crossing over” from the virtual world to the hardware prototype
  - Hardware prototypes always resulted in a “step function” effort
  - “Step function” effort should reduce when the virtual platform is incorporated

End-Of-Day Benefits

- Reduced spins of the design
- Much faster chip-evaluation when silicon arrives
  - Software ready to go
  - Integration/System-Engineering team already up to speed on the device
- Effective widening of the pre-silicon testing window

HAPS 52 Debug Demo

SYNOPSYS®

Predictable Success