Protium S1 – FPGA-based prototyping made easy

Volker Wegner, Staff Application Engineer
Tensilica Day
Hannover, Germany
February 07, 2018
30 years of innovation in emulation

Verification Computing Platforms

Processor-based Architecture

FPGA-based Architecture

2000 Quickturn
2005 Verisity/Axis

1987 to 2009

1995 to 2016

Moore’s Law is evolving - It’s about the software 😊
System & chip design in 2017 - 2020

- Time-to-market
- Development cost reduction
- Multi-core design and verification complexity
- Integration of new designs and derivatives
- Software stack development
- Hardware-software convergence
- More than 80% re-use
- More than 60% of effort in software
Customers Need the Fastest Engines

- Ever-increasing verification requirements driven by growing hardware and software complexity
- Fast time to results is essential to ensure projects can meet schedules
- Right tools for the right job: Combination of formal, simulation, emulation, and FPGA prototyping
There is no “One Size Fits All”
Verification and software platforms need to interoperate

<table>
<thead>
<tr>
<th>Platform Type</th>
<th>Key Features</th>
</tr>
</thead>
</table>
| SDK OS Simulation      | • Highest speed  
                         • Earliest in the flow  
                         • Ignore hardware |
| Virtual Platform       | • Almost at speed  
                         • Less accurate (or slower)  
                         • Before RTL  
                         • Great to debug (but less detail)  
                         • Easy replication |
| Formal Analysis        | • Non-scalable  
                         • Exhaustive  
                         • Early RTL  
                         • Great for IP  
                         • No SW execution |
| HDL Simulation         | • KHz range  
                         • Accurate  
                         • Excellent HW debug  
                         • Broadly available  
                         • Mixed abstractions  
                         • Limited SW execution |
| Acceleration Emulation | • MHz Range  
                         • RTL accurate  
                         • After RTL is available  
                         • Good to debug with full detail  
                         • Expensive to replicate |
| FPGA Prototype         | • 10’s of MHz  
                         • RTL accurate  
                         • After stable RTL is available  
                         • OK to debug  
                         • More expensive than software to replicate |
| Prototyping Board      | • Real time speed  
                         • Fully accurate  
                         • Post Silicon  
                         • Difficult to debug  
                         • Sometimes hard to replicate |
Cadence Verification Suite
Technology innovation leadership: Fast, Smart, and Optimized

- **Fast** Best-in-class engines
- **Smart** Flow-driven engine integrations
- **Optimized** comprehensive solutions
What is FPGA-Based Prototyping?

• **Primary platform for pre-silicon software development and validation**
  • Maps a digital ASIC, ASSP, SoC design or part thereof into one or more FPGAs

• **Allows software to execute in real world environments**
  • Provides pre-silicon execution speeds in the 10s of MHz
  • Enables connectivity to real peripherals
    • Runs real world traffic flows including interrupts and random events
    • Runs error conditions and handling errata with other system components
Case study: Amlogic

Leading SoC Supplier dedicated for Smart TV, OTT & Smart Home

California
Santa Clara
BD, R&D, Support

Shanghai:
Sales, R&D, Support

Beijing:
Sales, R&D, Support

Shenzhen:
Sales, R&D, Support

Taipei:
Sales, R&D, Support

Hong Kong:
Logistics
Case study: Amlogic (cont.)

Software Development Challenges

- **Increase in software complexity**
  - Multi-core, multi-task real-time system
  - Multiple layers of software (e.g. firmware, kernel driver, framework)
  - Millions line-of-code (e.g. Android, Linux kernel)

- **Time to market**
  - Software is required to ship SoC
  - Software delay == delay in $$$
  - Dependency on hardware availability
Case study: Amlogic (cont.)

Results

• Software is READY when silicon returns
  Basic Android running in 30 minutes
  after silicon is back
  Full Android demo to customer in 3 days

• Run software exactly same way as real silicon
  Compile, “flash” and run
  Boot from SD card or eMMC/NAND

• Rich debug capability
  JTAG & UART
  Memory capture and restore
  Stop clock and set/reset signals
  Trace any signal on Palladium XP
  Capture pre-defined signal for offline viewing on Protium
FPGA-Based Prototyping is Hard to Do …
FPGA-Based Prototyping Is Fragmented
Disjointed, lacking integrated flow and automation

- FPGA-based prototyping

Challenges:
- Fragmented
  - Requires RTL modifications
- Lack auto compilation
  - Memory and clocks
  - Partitioning
- Lack of flow integration
  - Emulation and prototyping
  - Configuration reuse
  - FPGA P&R
Really Hard to Do ... Or is It?
Protium S1 – Addressing the prototyping challenges

- No RTL modifications needed
  - Clocking / number of clocks
  - Automated memory compilation and modeling

- Fully automatic, multi-FPGA partitioning
  - Optional manual optimization

- FPGA Timing Closure
  - Multiple design integrations per day
  - Avoids time-consuming FPGA P&R

- Fully integrated FPGA P&R
  - Automatic constraint generation
  - Guaranteed P&R success
Fast Time-to-Prototype (TTP)

Note: Sample customer bring-up gains over traditional FPGA-based Prototyping solutions.
No RTL Modifications – Clocking

- Traditional imitations:
  - Gated clock, multiplexed clocks
  - # of clocks
  - Difficult to achieve FPGA timing closure
  - Long iteration times / long FPGA P&R times
  - Unpredictable results & prototype behavior

- Protium S1 benefits:
  - No hold-time violations in user clock domains
  - Removes any FPGA-specific clock limitations
  - Supports unlimited # of design clocks
  - Improves FPGA timing closure
  - Accelerates FPGA P&R times

Protium is “cycle-based”

- Protium updates each net in the design once per cycle of a conceptual clock called FCLK.
- FCLK is generated automatically by the compiler. Its frequency is determined by the compiler.
- Depending on the clocking mode, CAKE1x or CAKE2x, the fastest design clock changes once or twice per FCLK cycle.

FCLK and Step Clock

- In Protium hardware, FCLK is a conceptual clock, but step clock really exists.
- Step clock is ideally 150MHz, but may be slower.
- In each compile, the compiler determines both the step clock frequency and the step count
  - Step count is the number of step clock cycles per FCLK cycle
  - Typical step count is between 10 and 50
No RTL Modifications - Memories

• No ASIC RTL changes
  • Automatic conversion of latches and tri-states
  • Automatic memory compilation and modeling
  • Fully automated clock tree transformation
    • Automatic conversion of gated and multiplexed clocks
Innovative XDRAM & XSRAM Solution

• XSRAM
  – Benefits:
    – Increases FPGA internal memory from 80Mbits to 128MBytes (>10x)
    – Automatic mapping of any memory type
    – Support for multi-port memories
    – Support for backdoor upload/download

• XDRAM
  – Benefits:
    – Adds DDRx bulk memories
    – Supports LPDDR2/3/4; DDR3/4; HBM
    – No change to design memory controller and firmware
    – Support for backdoor upload/download
    – Acts as memory SpeedBridge (timing, refresh, etc.)
Comprehensive, automated memory support

Conversion and implementation of memories is one of the most challenging and time-consuming steps in bring-up of an FPGA-based prototype (often taking many weeks to complete).

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Palladium MMP</th>
<th>Upload/ Download</th>
<th>Perform.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA-internal</td>
<td>~50Mbits / FPGA</td>
<td>Yes</td>
<td>Yes</td>
<td>Full design speed</td>
<td>• Fully automatic compile</td>
</tr>
</tbody>
</table>
| XSRAM (automated small external memory)| 128 Mbytes per memory card         | Some          | Yes              | Full design speed | • Fully automatic compile  
• Extends ‘FPGA-internal’ memory to external SRAM  
• Useful for Serial Parallel Interface (SPI)-flash and other small memories (e.g. boot ROM) |
| XDRAM (automated bulk memory)         | 8/16 GBytes per XDRAM card         | DDR family models | Yes              | <8MHz       | • semi automatic compile  
• Leverages XDRAM hardware Support for DDR3/4, LPDDR3/4                                                                                     |
| DCMC (Direct Connected Memory Card)   | x GBytes (depending on memories used) | No            | No               | Full design speed | • Design change may be required, depending on memory type  
• App notes available                                                                                                                        |
| FCMC (Full-custom Memory Card)        | Custom                             | No            | No               | Full design speed | • Fully custom development                                                                                                                |

Protium S1 Memory compile capabilities:

- Smaller memories are **automatically compiled** into FPGA-internal resources
- For larger, off-FPGA memories, the Protium platform offers **several automated solutions**, see table
Hardware and Software Debug

**Hardware Debug: RTL**
- Waveforms across partitions
  - Design-centric view vs. FPGA-centric
- Force/release
  - Predefined signals (at compile time) to “0” or “1” during runtime
- Monitor signal
  - Real-time monitoring of predefined (at compile time) signals

**Software Debug: C Code**
- Backdoor memory access
  - Quickly change boot code, software, etc.
- Clock control
  - Start/stop the clock on demand
- Fully scriptable runtime environment
- Remote access
  - Network resource anytime from anywhere
  - High-performance link to software model

**Software**
- Applications
- Middleware
- Operating Systems (OS)
- Drivers
- Firmware / HAL

**SoC, Sub-System or IP**
- Compute Sub System
- Customer’s Application-Specific Components
- SoC interconnect fabric
- High-Speed, Wired Interface Peripherals
- General-Purpose Peripherals
- Low-Speed Peripherals
- JTAG
- Probes
- Daughter-cards & peripherals
- Software Debug: C Code
- Hardware Debug: RTL
Scalable performance

Performance (single board, multi-FPGA)

Automatic mode

Further Optimization

Phase 1
- Automatic for quick functionality

Phase 2
- Higher effort performance optimization

Phase 3
- Design-based user manual refinement
Scalable hardware

**Protium S1-G**
- Single board
  - 1 FPGA
  - Up to 25M ASIC gates
  - Affordable and scalable
  - Highest performance
  - Early software development
  - IP verification

**Protium S1-SC**
- Single chassis system
  - 2-8 FPGAs
  - Up to 200M ASIC gates
  - Flexible and scalable
  - Fastest bring-up
  - Unique SW debug capabilities
  - Early software development
  - HW/SW integration

**Protium S1-MC (2H 2017)**
- Multi-chassis configuration
  - 8-24 FPGAs
  - Up to 600M ASIC gates
  - Highest capacity
  - Flexible use modes
  - Advanced debug
  - High performance regression
  - Full system validation
Palladium Z1 + Protium S1

Palladium® Z1
• Best debug
• SoC integration & accel
• Rich use models

Protium™ S1
• Highest performance
• SW development
• Regressions

Congruency and common environment
Protium S1 Fully Integrated Implementation Flow

- Automated prototyping flow reduces time-to-prototype (TTP) from months to weeks
  - Design changes have much lesser time impact on iterations
  - Simpler single pass flow iterations are run in hours not days
  - Software development gets a head start measured in months not days

<table>
<thead>
<tr>
<th>Months</th>
<th>Weeks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional FPGA-based Prototyping Flow</td>
<td>Protium S1 FPGA-based Prototyping Flow</td>
</tr>
<tr>
<td>ASIC RTL (Verilog/VHDL/SV)</td>
<td>ASIC RTL (Verilog/VHDL/SV)</td>
</tr>
<tr>
<td>Edit RTL</td>
<td>FPGA Partitioning and Synthesis</td>
</tr>
<tr>
<td>Prototype RTL (Verilog/VHDL/SV)</td>
<td>FPGA Vendor P&amp;R</td>
</tr>
<tr>
<td>FPGA Partitioning and Synthesis</td>
<td>FPGA bit Files</td>
</tr>
<tr>
<td>FPGA Vendor P&amp;R</td>
<td>Independent re-run of selected tests for debug</td>
</tr>
<tr>
<td>FPGA bit Files</td>
<td>Optional</td>
</tr>
<tr>
<td>Test</td>
<td>Functional Assurance Before P&amp;R</td>
</tr>
</tbody>
</table>

† 1-2 days per iteration
† 1-2h per iteration

Case study: Microsemi

- Microsemi corporation is a Global provider of semiconductor solutions for applications focused on delivering power, reliability, security, and performance

- High-value, high-barrier entry markets:
  - Communications
  - Enterprise storage
  - Defense and security
  - Aerospace
  - Industrial

Corporate headquarters in Aliso Viejo, CA
Case study: Microsemi (cont.)

- Mapping design to multiple FPGAs is not automated: design partitioning
- Clocking issues
- Manual interventions and several iterations for timing closure
- ASIC RTL modifications
- FPGA debug tools improved, but still not good enough to tackle complexity
- Peripheral availability: daughter cards for PCIe, USB, DDR,
- Memory mapping
- Reusability: most of the time, custom platforms are not reusable

Why explore new prototyping platforms?
The traditional FPGA-prototyping flow has broken up, it is hard to debug, and its performance is not always what we expect.

Why Choose Protium?

- Shares a common front-end flow with Palladium, which results in easier builds and maintenance!
- Protium HW can be used with any custom flow
- Requires no ASIC RTL modifications
- Does not need manual intervention for timing closure
- Users can easily generate as many clocks as they need
- Large variety of peripherals and memories to quickly connect the DUT to a realistic environment
- Can use traditional on-chip debug tools or port the design back to Palladium
- Forces and monitors
- Memory backdoor access for debug and quick FW deposits: compatible with Z1 scripts

- Protium has incredibly simplified our prototyping flow
- It allowed us to significantly improve prototyping bring-up time
- It helped to off-load Palladium and use it in a more efficient way
Cadence SpeedBridge Adapter Solutions
Palladium® Series

Protium S1

Video/Audio
Data Streaming
HD interfaces

SATA

SAS

USB 3.0/2.0/1.1 Device

USB 2.0 Host

Ethernet 10/100, 1G, 10G, 40G

PCIe 4.0/3.0/2.0/1.1/1.0a
SpeedBridge Configuration Manager (SCM)

- Error injection during System-level test*
- The SpeedBridge™ Configuration Manager components
  - Hardware: SCM; Software/GUI: SpeedBridge view
- Benefits of Remote Configurability
  - Confirmation of setup and configuration
    - A complete environment for remote monitoring & use
    - Protocol specific debug and SpeedBridge information
    - DIP switches, LED status
    - Remote SpeedBridge reset capability
  - Ability to monitor multiple SpeedBridges from a single GUI

*New! – (EA) capability available on some of the SpeedBridge adapters
Comprehensive portfolio of accessories and interfaces

• Please see datasheet for comprehensive list of what is natively supported.
Protium S1 – the Most Efficient Way to Prototype Your SoC

- **Fast time-to-prototyping** (months to 1-2 weeks)
  - No RTL changes
  - Automatic partitioning/memory compilation
  - Fully integrated FPGA place-and-route

- **Scalable performance** (3-100MHz)
  - From fully automatic to fully manual
  - Advanced black-box methodology

- **Advanced software debug**
  - Memory upload/download
  - Force and release
  - Data Capture card
  - Assertion checkers
  - State read-back
Protium S1 Prototyping Solution
Industry's first comprehensive, fully integrated solution
“The Cadence Protium S1 platform ensures scalability to hundreds of software developers at the earliest possible point during the development flow, and allows developers to focus on design validation and software development rather than prototype bring-up. The common flow with the Cadence Palladium Z1 emulation platform enables a smooth transition from emulation to prototyping, which greatly improves productivity.”

*Peter Ryser, Senior Director for System Software, Integration, and Validation, Xilinx*