The Veloce Emulator
and its Use for Verification and System Integration of Complex Multi-node SOC Computing System

Laurent VUILLEMIN
Platform Compile Software Manager
Emulation Division
Agenda

- What is Emulation
- Use models
- Veloce Architecture Overview
- Veloce Software
- Practical: Use Veloce to verify Veloce
Agenda

- What is Emulation
- Use models
- Veloce Architecture Overview
- Veloce Software
- Practical: Use Veloce to verify Veloce
Verification Challenges

Systems

Software

SPEED

Debug
Typical Development Cycle

- Hardware Development
  - Block-level Verification
  - Chip-level Verification
- Fab
- System Integration
  - System Verification
- Software Development

Time to Market
Typical System Development

Design Cycle

Hardware Development

Fab

System Integration

Software Development

Block-level Verification

Chip-level Verification

System Verification

Time to Market

© 2010 Mentor Graphics Corp. Company Confidential
www.mentor.com
Software Simulation

- Model is represented in Data Structures
- Compute and propagate signal values

RTL Model

```verilog
define module counter (ck, en, step, dout);
  input ck, en;
  input [3:0] step
  output [3:0] dout
  reg [3 : 0] dout;

  always @ (posedge ck) begin
    if ( en ==1 )
      dout = dout+step;
  end
endmodule
```

Test bench

```verilog
define module counter (ck, en, step, dout);
  input ck, en;
  input [3:0] step
  output [3:0] dout
  reg [3 : 0] dout;

  always @ (posedge ck) begin
    if ( en ==1 )
      dout = dout+step;
  end
endmodule
```
Clock Speed Scaling Stalls

Emulation Required to Extend Performance

Source: Recording Microprocessor History 4/6/2012 Andrew Danowitz, Kyle Kelley, James Mao, John P. Stevenson, Mark Horowitz http://queue.acm.org/detail.cfm?id=2181798
Emulation

• Multiples FPGA reproduce Model behavior from high level description

module counter (ck, en, step, dout);
  input ck, en;
  input [3:0] step
  output [3:0] dout
  reg [3 : 0] dout;

always @ (posedge ck)
begin
  if ( en ==1 )
    dout = dout+step;
end
endmodule

Map RTL model in Programmable Logic

Control execution and get results
Start Early – Continue for Entire SoC Life

- Architecture Phase
- Block-level Verification
- Fullchip/SoC Verification
- Software, Firmware & Device Drivers
- Systems Validation
- Post Silicon Bringup and Validation
Agenda

- What is Emulation
- Use models
- Veloce Architecture Overview
- Veloce Software
- Practical: Use Veloce to verify Veloce
ICE use Model

- The emulator is connected to actual Hardware
Ethernet Verification With ICE
Co Emulation Testbench Xpress (TBX)

- Testbench divided in 2 parts one in emulator the other in Station
- Communication via transactor
- Software sends commands that are interpreted by transactor to generate DUT stimulus

Testbench

Software

Hardware

High-speed Interface
Veloce Use Models

Simulation Acceleration
- OVM/UVM
- SystemVerilog
- C/SystemC

Accelerated Transactors

Software Debug
- Codelink
- VProbe
- Fast ISS
- QEMU

Virtual Protocol Solutions
- USB
- SATA
- Video
- Ethernet
- PCIe

VirtuaLAB Solutions

Physical Protocol Solutions
- PCIe
- SATA
- Video
- USB
- Ethernet

iSolve Solutions

Co-Model Channels

Testbench Xpress

Physical I/O
Agenda

- What is Emulation
- Use models
- Veloce Architecture Overview
- Veloce Software
- Practical: Use Veloce to verify Veloce
Veloce 2 Architecture

Crystal2 Chip

- 16 X
- + 40 SXB (Switch boards)
- + 4 CXB (clock board)

AVB2

- 64 X
- 16 AVB2
- + 6 SXB (Switch boards)
- + 1 CXB (clock board)

Maximus2

Quattro2
Crystal 2 IC

- Programmable Logic Array
  - Set of LUT and Sequential elements
  - Interconnect Network
- Memories:
  - User Memories model
- Virtual Wire Logic
  - Transport signals between chips
- Debug Resources
  - Trace every Sequential element and memories
  - Triggers
- Control
  - Load configuration
  - Control emulation
Agenda

- What is Emulation
- Use models
- Veloce Architecture Overview

- Veloce Software
  - Compile Software
  - Runtime Software

- Practical: Use Veloce to verify Veloce
Software Overview

PC Farm

Compile Servers

Emulator host

User design

Compile SW

Transform design in bitstream

Configuration bitstream

Runtime SW

Low Level SW

User Interface
Control Emulator
Collect debug data
Compile software

1. Design + Testbench

2. TBX
   - Partition Testbench between SW and Emulator

3. RTL
   - Perform RTL Synthesis

4. RTLC

5. Gate netlist

6. Platform Compile
   - Partitioning in Crystal System Routing
   - Crystal Place and Route Ressource Allocation

7. Bitstream
Runtime Software

- Emulator Message Bus
  - Maintenance Server
  - Resource Server
  - Runtime Server
  - Emulator Host Server
  - LLSW

- User Message Bus
  - Visibility Server
  - UI
Agenda

- What is Emulation
- Use models
- Veloce Architecture Overview
- Veloce Software

Practical: Use Veloce to verify Veloce
  - Challenges
  - The verification infrastructure
  - Verifying ASIC
  - Verifying the Compilation software
  - Low level software integration
  - Firmware validation debug
  - Runtime software integration
Challenges

- Complex system
  - ASIC
  - FPGAs Firmware
  - Multiple software components

- Verifying all component of the system and their interactions

- Time to get a bug

- Size:
  - Detailed model of a full Emulator of next generation will not fit in current generation
Addressing the challenges

- Use a comodel approach for more abstraction level and connection with the software

- Divide verification in steps based on functionality

- Simplify the model by using different abstraction level depending on what functionality is tested
Verification Infrastructure

More details on Veloce

LLSW on host → Control Chip → Control → Crystal → Control → Crystal

Veloce Control Bus

Control Commands
Nature of the Software, Transactor and Model in emulator depend on abstraction level
ASIC verification: example of Trace

Trace Capture, Trace Control and DDR Controller are Accurate models
Design is a gate level netlist generating random data
Data are either manually generated or come from actual compile
Run million cycles on multiple designs
Verification of compile SW

Macro and Configuration blocks are Accurate models
Bitstream is the output of actual compile flow
Verify behavior of design
Virtual Wire need a training/Calibration sequence
This sequence is controlled by Low Level Software
Control block in Crystal and VW block are accurate model
Actual LLSW communicate with the model through comodel SW and transactor
Verification of Firmware 1/2

Example: Trigger Reduction

A trigger expresses a condition on values coming from the design. At AVB and System level, it is implemented in FPGA. A binary is generated by runtime SW to express the condition.
Verification of Firmware

Design is modeled as gate level netlist
Trigger binary is generated by actual runtime SW
Verify behavior of trigger in multiple design sequences
Actual Runtime Server is used
High level model for the HW
QUESTIONS