



SOPC
WORLD
2004

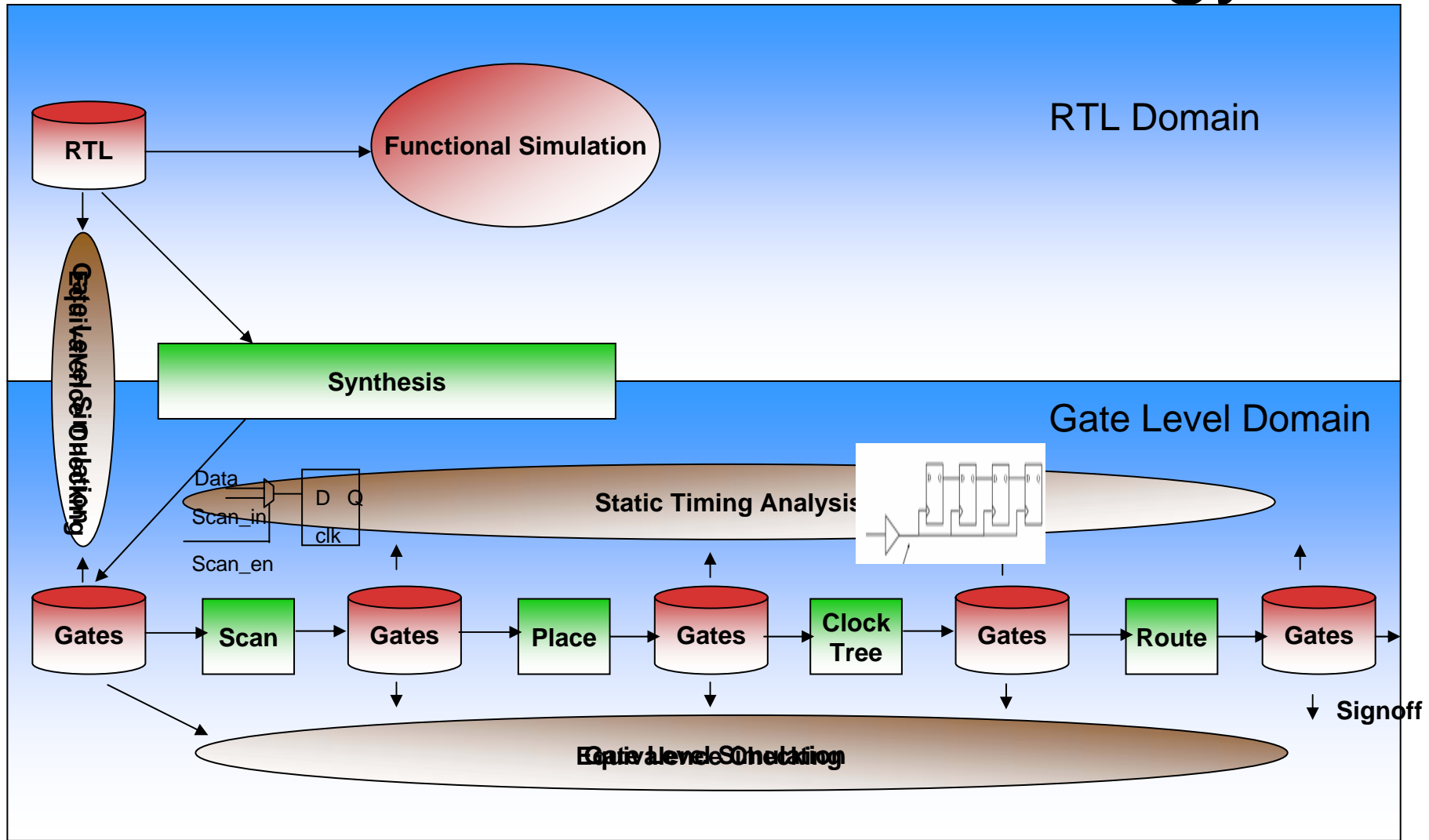
Formal Verification Seminar

Presented by SYCHOI

Agenda

- ASIC Verification Methodology
- What is the Formal Verification?
- Why Equivalence Checking?
- Equivalence Checking Flow
- How to use Conformal LEC
 - Libraries
 - Environment Variables
 - Quartus II settings
- Quiz

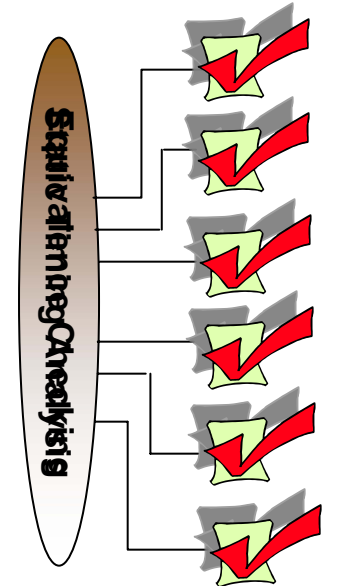
ASIC Verification Methodology



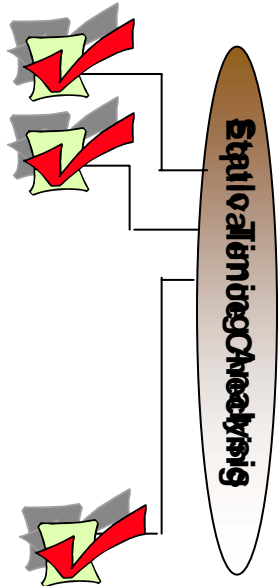
Verification Comparison

Compare

ASIC Flow



FPGA Flow



- Functional (RTL)
- Synthesized netlist
- Scan Insertion
- Placement
- Clock Tree Synthesis
- Routing



ASIC Verification Tools

■ Event Based Simulators

- 1000 Cycles/Sec
- VCS, NC-SIM, ModelSim
- \$5000 to \$50,000

■ Cycle Based Simulators

- 5000 Cycles/Sec
- Scirocco, SpeedSim
- \$50,000 to \$100,000

ASIC Verification tools

■ Hardware Accelerators

- 100,000 Cycles/Sec
- Hammer, CoBalt Plus
- > \$250,000

■ Hardware Emulation

- 1,000,000 Cycles/Sec
- VN-Cover
- > \$250,000

■ FPGA Prototyping



Emulators and Accelerators

- V-Station
- Co-modelling
- Celaro
- ARES RTL Acceleration
- Mercury Plus
- Palladium
- SpeedBridge



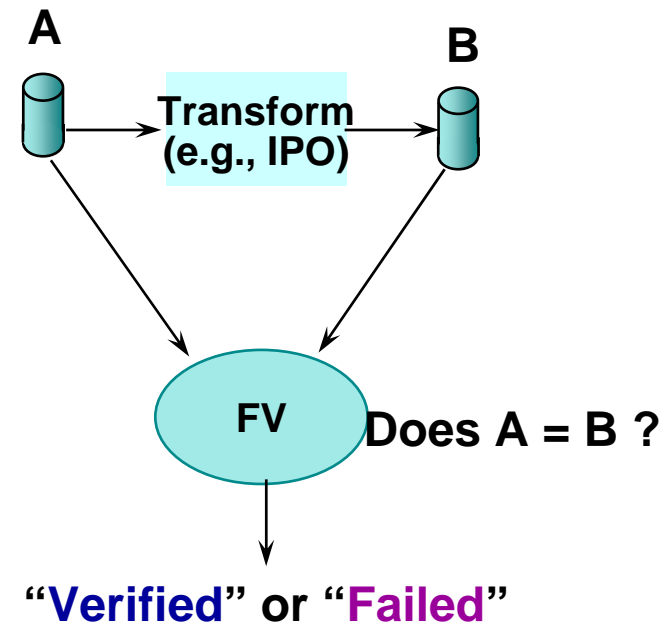
\$200,000 –
\$1,000,000

Agenda

- What is Formal Verification?
- Why Equivalence Checking?
- Equivalence Checking Flow
- How to use Conformal LEC
 - Libraries
 - Environment Variables
 - Quartus II settings

What is Formal Verification?

- Formal Verification is Equivalence Checking:
 - Very fast replacement of gate-level simulation for regression testing
 - 100% verification of all functionality without vectors
- Formal Verification proves mathematically that two designs have the same functionality:
 - RTL-to-gates
 - Gates-to-gates
 - RTL-to-RTL
- Verifies that design function has not changed

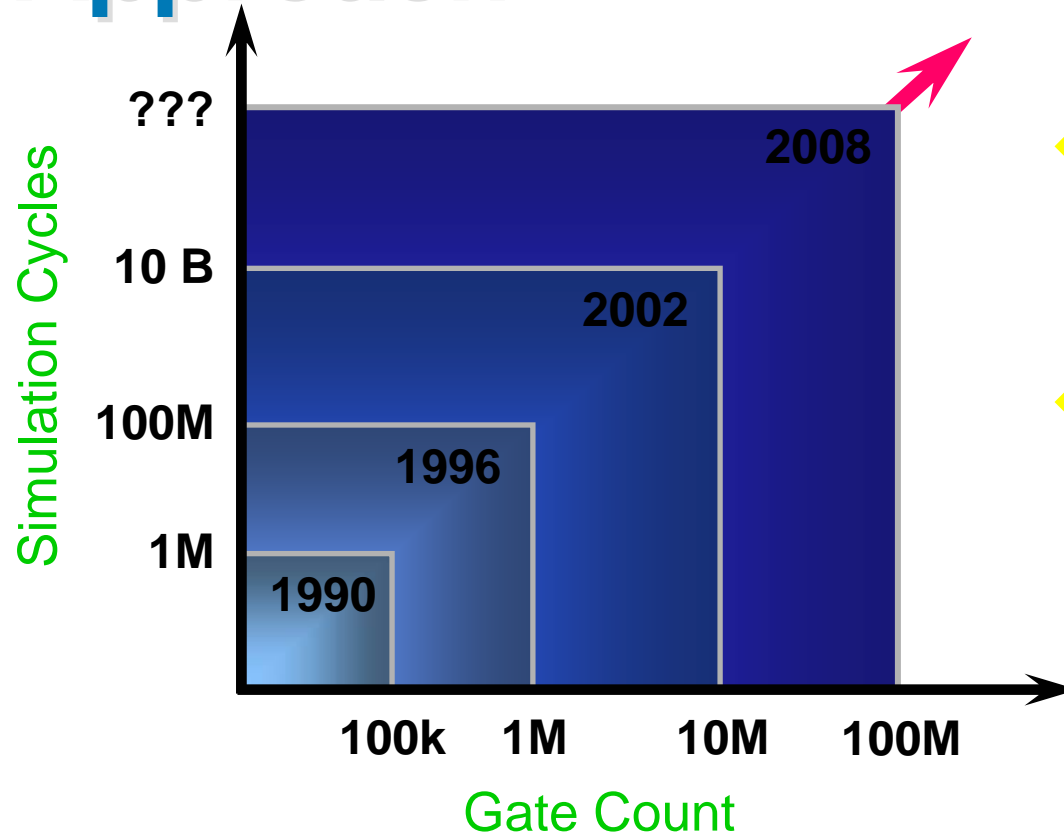


Equivalence Checking Using Conformal LEC

Agenda

- Why Equivalence Checking?
- Equivalence Checking Flow
- How to use Conformal LEC
 - Libraries
 - Environment Variables
 - Quartus II settings

Problems with Simulation Approach



- ◆ Designs continue to grow in accordance with Moore's law
- ◆ Effort required to verify these new designs doubles every 6 to 9 months

Simulation simply cannot fill the verification gap

What are the Components of Verification ?

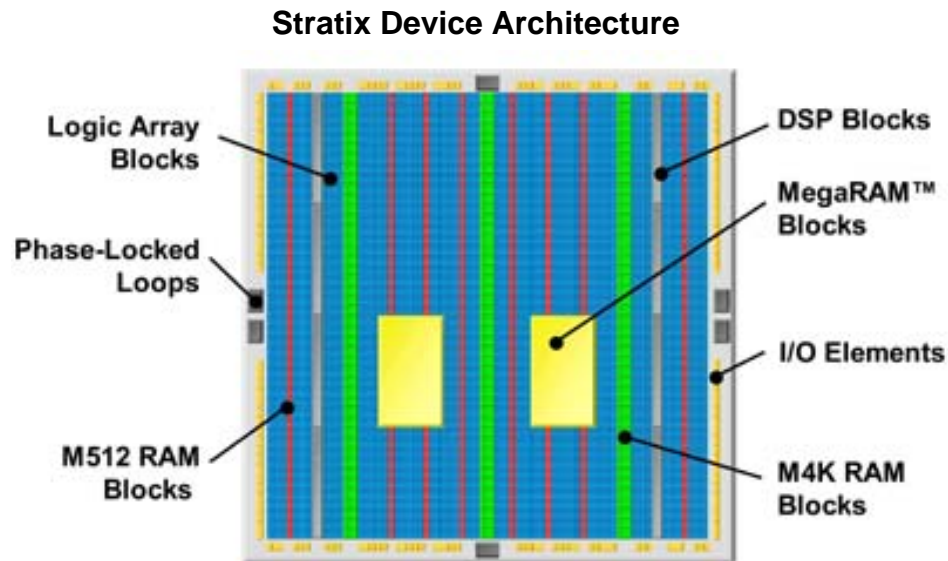
Engineers need to catch functional bugs associated with

Functional Inconsistency	Unintended and unexpected design behavior
Semantic Inconsistency	Introduced by unsafe RTL code
Logical Inconsistency	Introduced by design implementation process
Structural Inconsistency	Bus contention, bus floating, tri-state stuck-at
Initialization	Start-up state problems
Test Logic	Boundary scan, internal scan, test logic
Clock Synchronization	Signals cross clock domains w/o proper synch.

Today, most engineers still depend on **simulation** to catch these bugs .

Growing Need for Equivalence Checking in FPGA

- FPGA devices approaching ASIC complexity
 - Speed, Capacity, SOPC style (embedded memories, Intellectual Property, DSPs, CPU)
- ASIC-like design verification challenges in FPGA's
 - Implementation process involves many netlist changes



© 2004 Altera Corporation

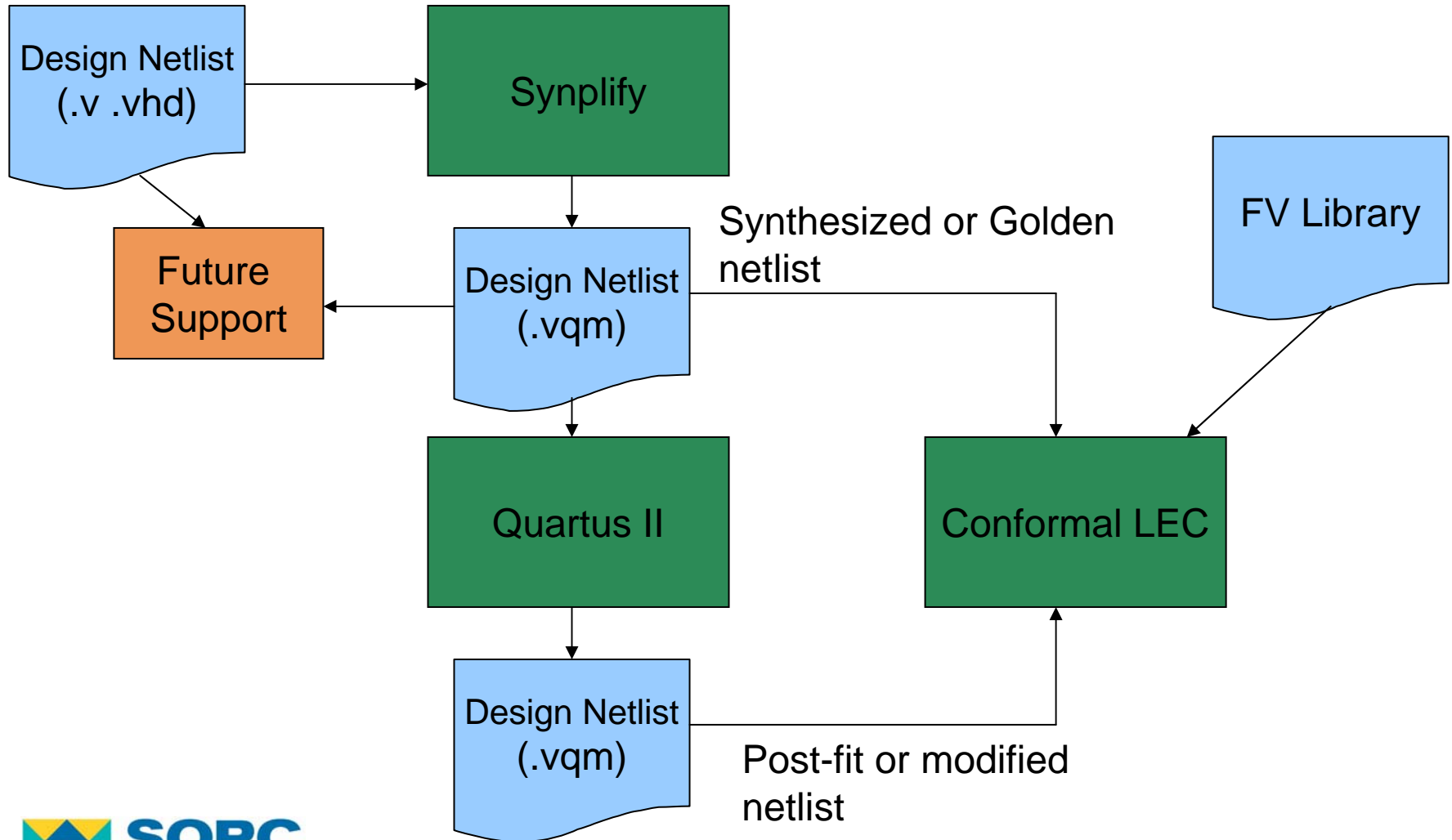
Equivalence Checking Advantages

- Very high capacity and performance:
 - Orders of magnitude faster than simulation
- Best assurance of design correctness:
 - 100% complete functional verification without using test vectors
- Easy to adopt and use:
 - Integrates smoothly in existing flows
 - Effective debugging capabilities

Agenda

- Why Equivalence Checking?
- Equivalence Checking Flow
- How to use Conformal LEC
 - Libraries
 - Environment Variables
 - Quartus II settings

Equivalence Checking – Altera Supported



Agenda

- Why Equivalence Checking?
- Equivalence Checking Flow
- How to use Conformal LEC
 - Libraries
 - Environment Variables
 - Quartus II settings

Formal Verification Library

■ <quartus_install>/eda/fv_lib

Supported families

1. apex20ke_atoms.v apex20ke_bbox.v
2. apexii_atoms.v apexii_bbox.v
3. cyclone_atoms.v cyclone_bbox.v
4. stratix_atoms.v stratix_bbox.v
5. stratixgx_atoms.v stratixgx_bbox.v
mfs_hssi_bbox.v

lpms.v
prims.v
lpms_bbox.v
mfs_bbox.v

**Common set of files
that have to be read
with all the families**

Platforms & Tools

- Platforms

- Solaris, HP Unix and Linux

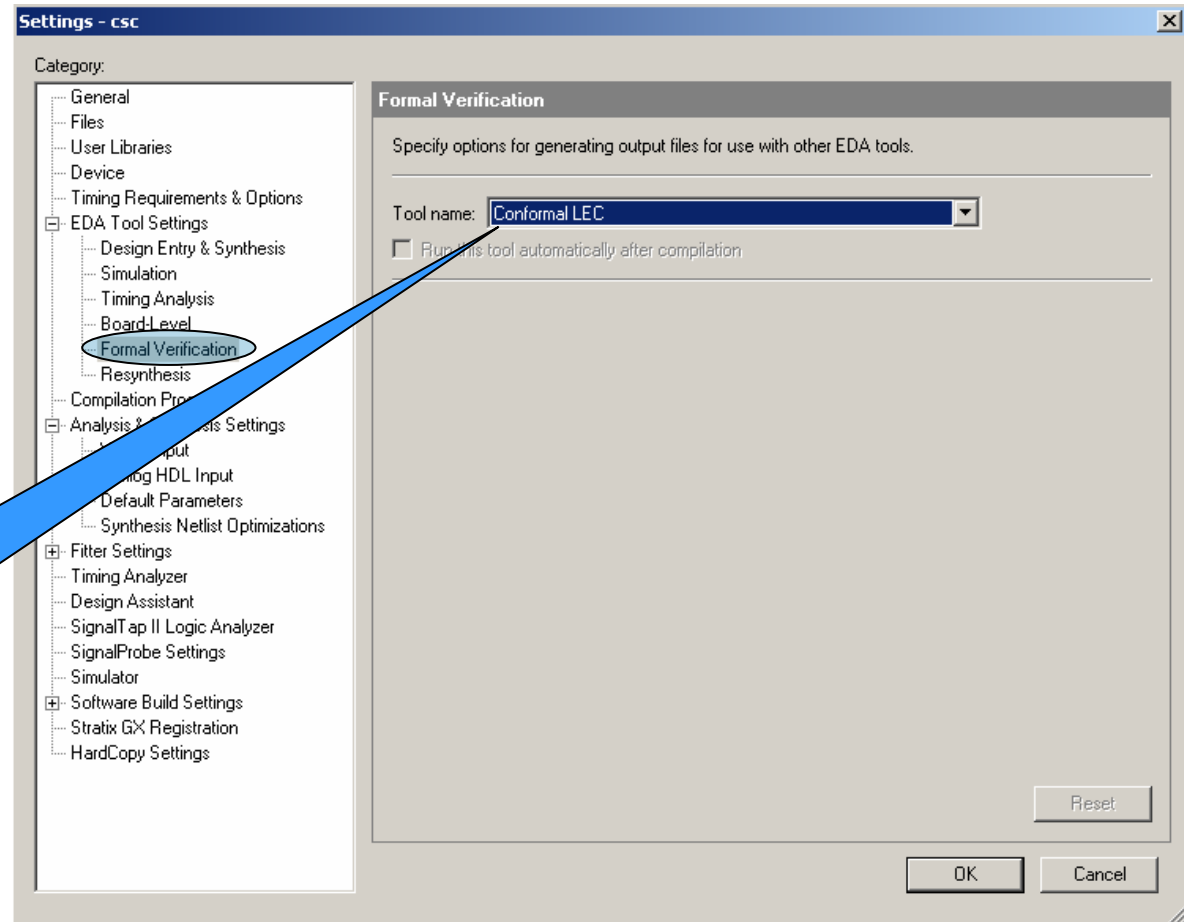
- Tools

- Synplify, Conformal LEC & Quartus II

Quartus II Settings

- Assignment -> EDA Tools Settings

**Set
Formal Verification
= Conformal LEC**



Quartus II settings

Settings - csc

Category:

- General
- Files
- User Libraries
- Device
- Timing Requirements & Options
- EDA Tool Settings
- Compilation Process
- Analysis & Synthesis Settings
 - VHDL Input
 - Verilog HDL Input
 - Default Parameters
 - Synthesis Netlist Optimizations
- Fitter Settings
 - Physical Synthesis Optimizations
- Timing Analyzer
- Design Assistant
- SignalTap II Logic Analyzer
- SignalProbe Settings
- Simulator
- Software Build Settings
 - Stratix GX Registration
 - HardCopy Settings

Synthesis Netlist Optimizations

Specify options for performing netlist optimizations during synthesis. Note: The availability of these options depends on the current device family.

- Perform WYSIWYG primitive resynthesis (using optimization technique specified in Analysis & Synthesis settings)
- Perform gate-level register retiming
- Allow register retiming to trade off Tsu/Td

Settings - csc

Category:

- General
- Files
- User Libraries
- Device
- Timing Requirements & Options
- EDA Tool Settings
- Compilation Process
- Analysis & Synthesis Settings
 - VHDL Input
 - Verilog HDL Input
 - Default Parameters
 - Synthesis Netlist Optimizations
- Fitter Settings
 - Physical Synthesis Optimizations
- Timing Analyzer
- Design Assistant
- SignalTap II Logic Analyzer
- SignalProbe Settings
- Simulator
- Software Build Settings
 - Stratix GX Registration
 - HardCopy Settings

Physical Synthesis Optimizations

Specify options for performing physical synthesis optimizations during fitting. Note: The availability of these options depends on the current device family.

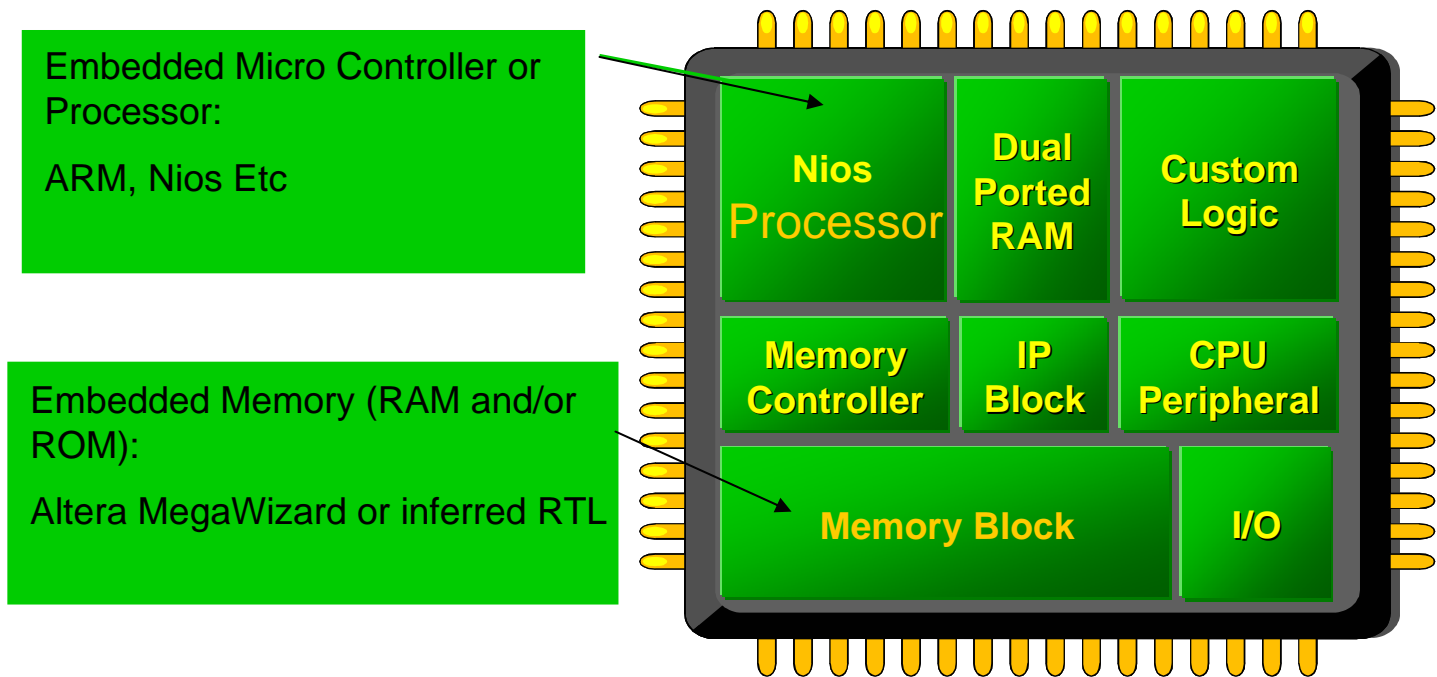
- Perform physical synthesis for combinational logic
- Physical synthesis for registers
 - Perform register duplication
 - Perform register retiming

Uncheck Synthesis And Fitter Optimizations options

OK Cancel

IP and Megafunction Support

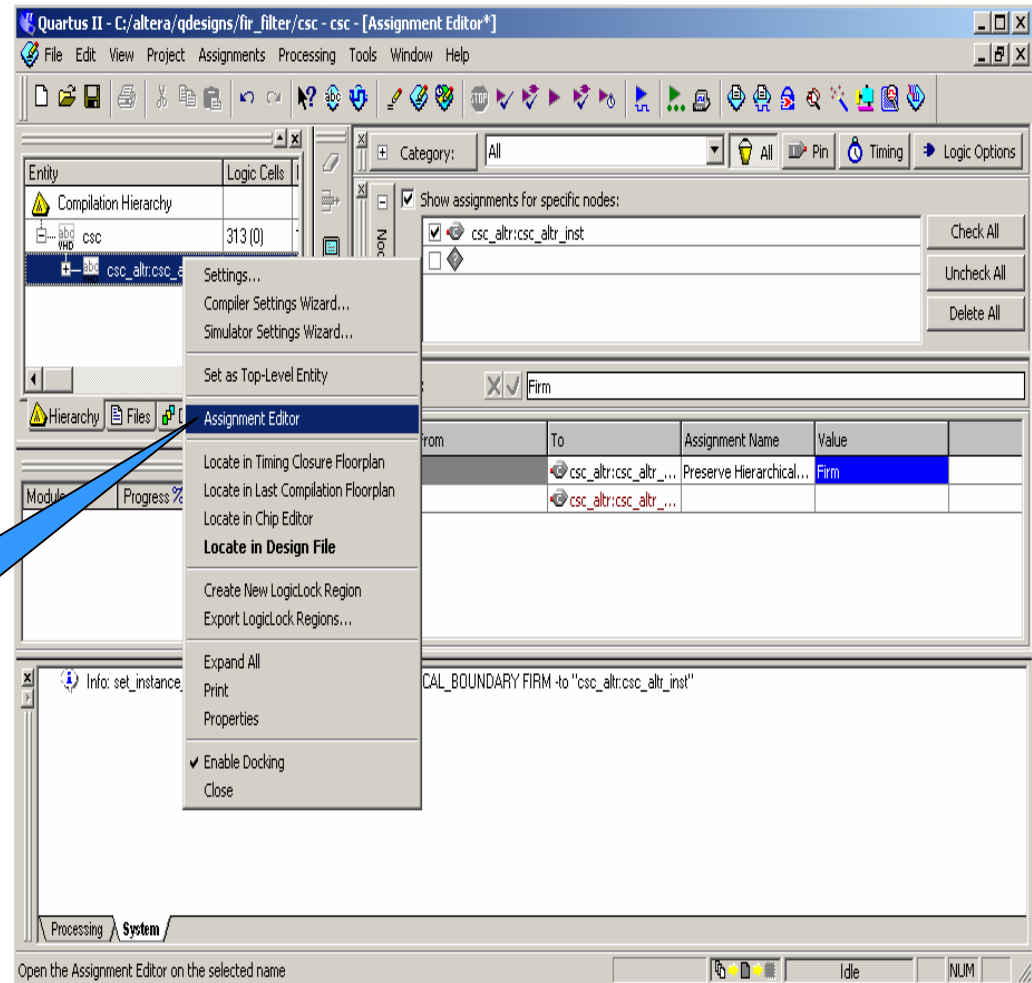
■ SOPC Design Elements



IP and Megafunction Support

- Encrypted IP and Mega functions are treated as black boxes

Selecting the black box to set the preserve hierarchy property



IP and Megafunction Support

- Assignment Editor -> preserve hierarchy property -> Firm

The screenshot shows the Quartus II Assignment Editor interface. The 'Entity' table on the left lists 'csc' and 'csc_altr:csc_altr_inst'. The 'Node Filter' section shows 'csc_altr:csc_altr_inst' selected. The 'Edit' table below shows two rows with the 'Firm' value assigned to 'Preserve Hierarchical...'. The status bar at the bottom displays the command: `_assignment -name PRESERVE_HIERARCHICAL_BOUNDARY FIRM -to "csc_altr:csc_altr_inst"`.

From	To	Assignment Name	Value
1	csc_altr:csc_altr_...	Preserve Hierarchical...	Firm
2	csc_altr:csc_altr_...		

preserve hierarchy
Property -> Firm

Agenda

- Why Equivalence Checking?
- Equivalence Checking Flow
- How to use Conformal LEC
 - Libraries
 - Environment Variables
 - Quartus II settings

POP Quiz

- Formal Verification is performed to verify
 1. Equivalence between RTL to Gate
 2. Functional equivalence between two netlists
 3. Real timing simulation

Quiz Answer

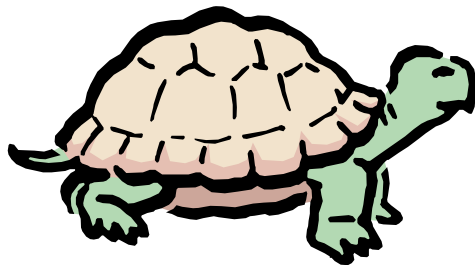
- Formal Verification is performed to verify
 - Functional equivalence between two netlists
 - Gate level timing
 - Functional RTL Verification
 - Synthesis results

Summary

- Advances In Verification Technology Make The Entire Design Cycle Time Shorter

→ Effectively Reducing The Time To Market

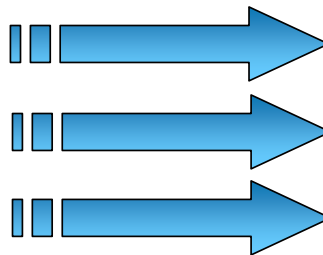
Before



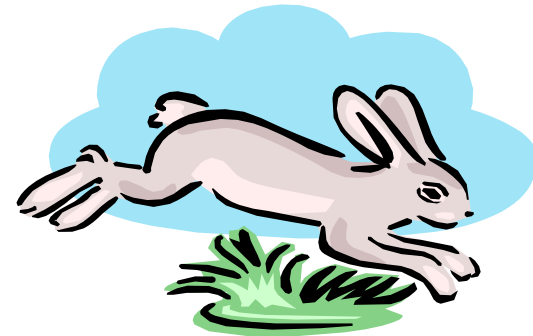
•HDL Simulation



Advancement In
Verification Technology



After



•Hardware Emulation/Acceleration

•Formal Verification



© 2004 Altera Corporation