

QUALIS™

A Strategic Process for System-Level Verification

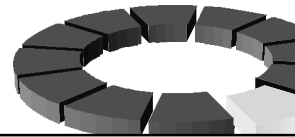
Janick Bergeron

Qualis Design Corporation

Lake Oswego, Oregon USA

<http://www.qualis.com>
+1-503-670-7200

Copyright © 1998 Qualis Design Corporation



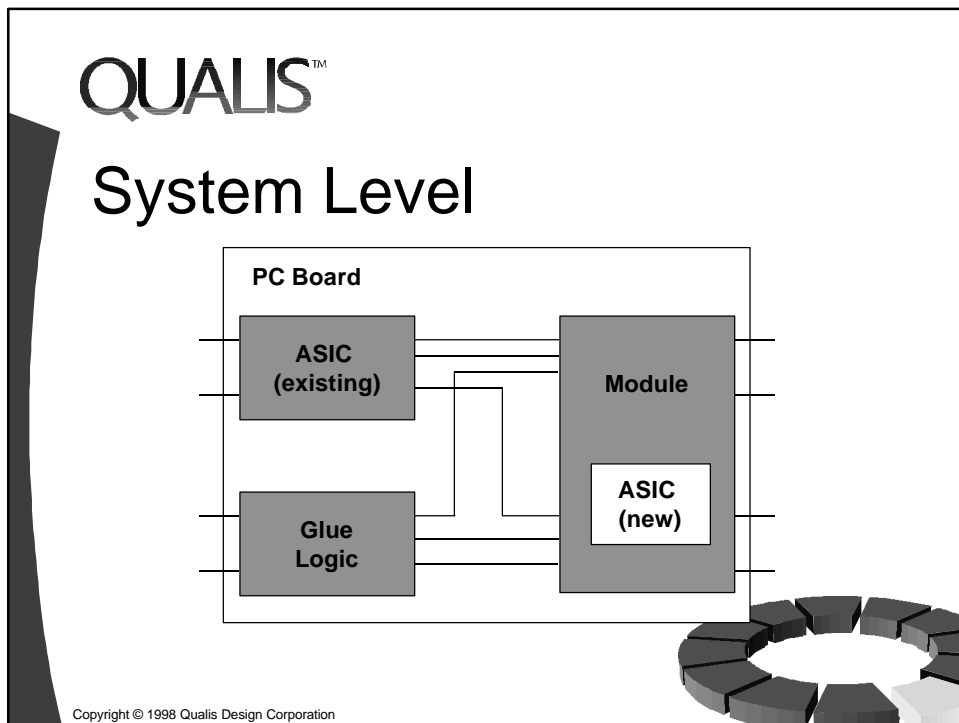
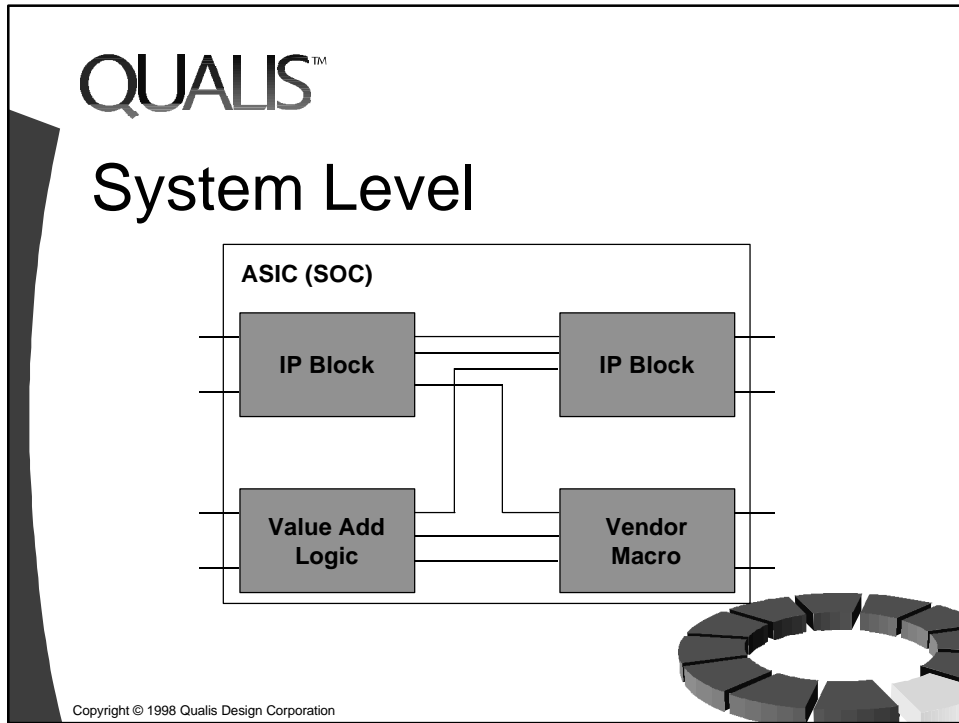
QUALIS™

What Is the System Level?

- ◆ Level at which a design is a collection of major functional blocks
- ◆ Can be thought of as the “IP level”
- ◆ Could occur in many forms:
 - Embedded IP blocks (SOC)
 - ASICs
 - Modules
 - Boards
 - Combinations of all the above

Copyright © 1998 Qualis Design Corporation

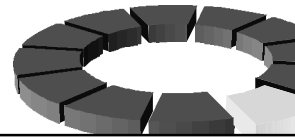




QUALIS™

What Is System-Level Verification?

- ◆ Checking that the entire system works as specified
- ◆ Verify that large circuit blocks interact correctly
- ◆ Natural extension of chip complexity, SOC, and IP blocks
 - Full systems are becoming collections of pre-designed functional blocks stitched together
 - Although blocks may be known good as standalone, they need to be verified in the target environment

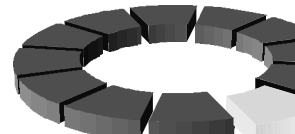


Copyright © 1998 Qualis Design Corporation

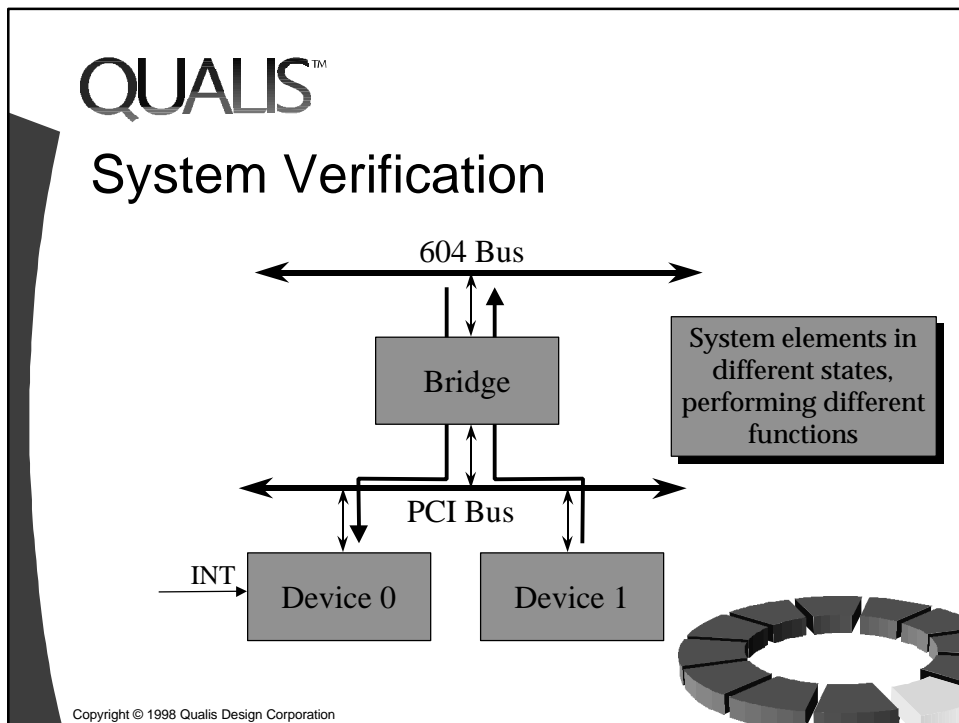
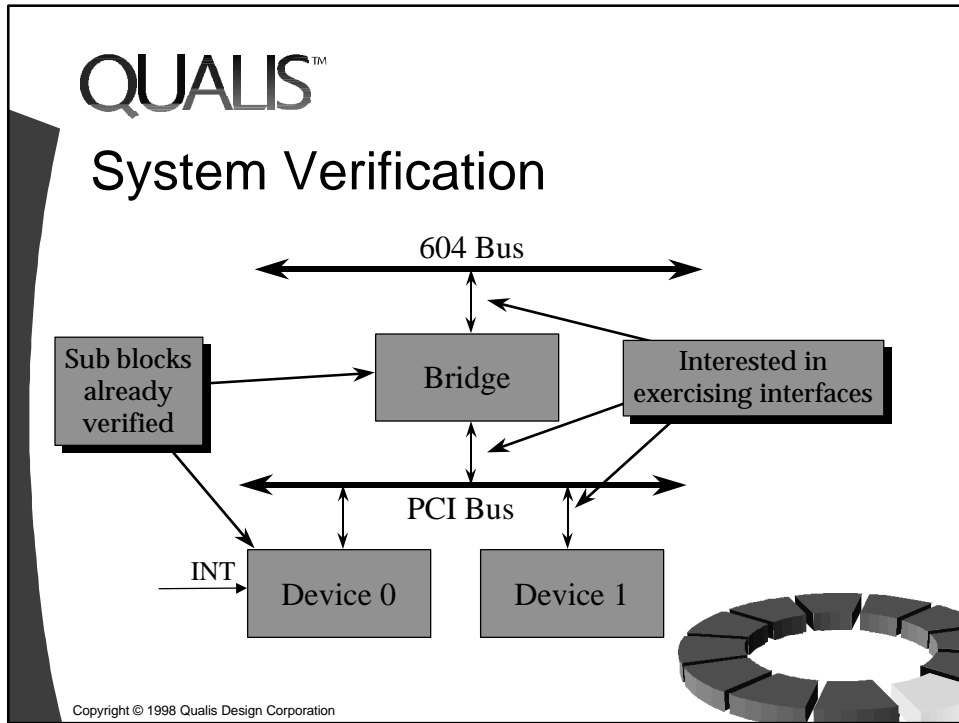
QUALIS™

What's Different About Systems?

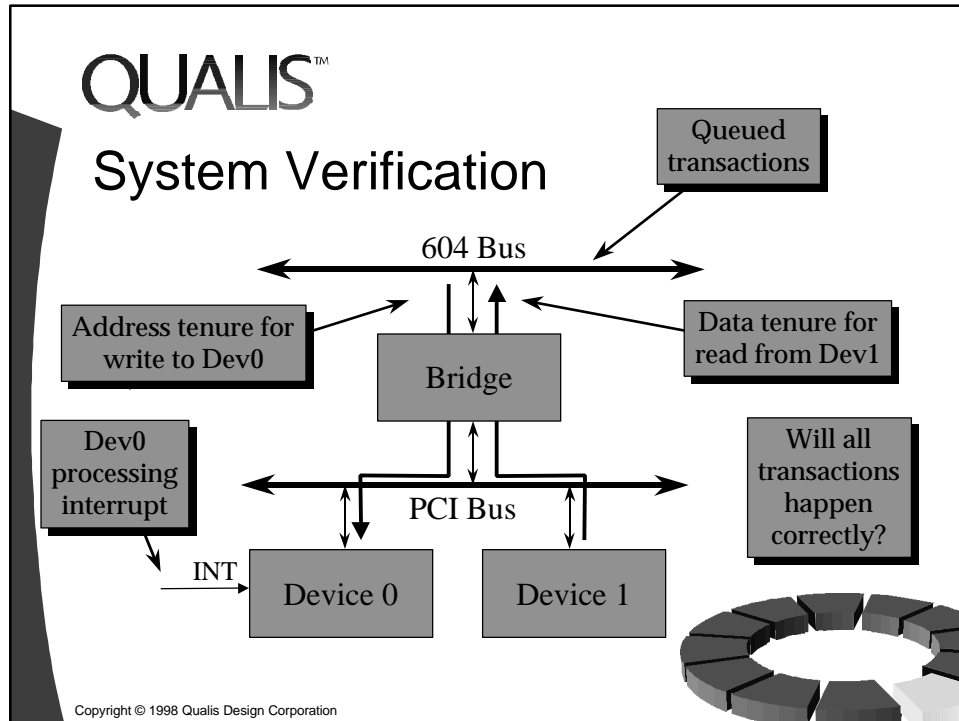
- ◆ Major functional blocks have already been verified
- ◆ Checking protocols/interfaces instead of sub blocks
- ◆ Checking interactions between conditions and states of elements
- ◆ Functional blocks often act independently
- ◆ Too many conditions to test with directed testing



Copyright © 1998 Qualis Design Corporation



A Strategic Process for System-Level Verification



QUALIS™

What's Different About Systems?

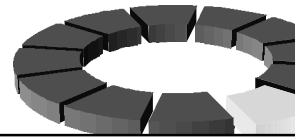
- ◆ Verification code likely to be more advanced:
 - Independent nature of elements
 - Creating and testing uncommon conditions (corner cases)
 - Random testing requires very robust code

Copyright © 1998 Qualis Design Corporation

QUALIS™

Why Do System-Level Verification?

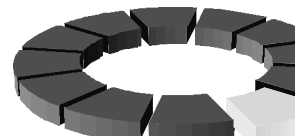
- ◆ Even though the sub blocks are verified, the system is still broken if the blocks don't work together
- ◆ Systems are too complex to skip organized verification - avoid integration woes with simulation
- ◆ Too many conditions to test with directed testing - lack of coverage leaves the design exposed
- ◆ No design is an island



Copyright © 1998 Qualis Design Corporation

QUALIS™

General Verification Issues/Methods

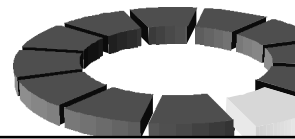


Copyright © 1998 Qualis Design Corporation

QUALIS™

Design Culture

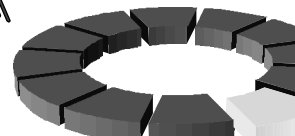
- ◆ Verification needs to share top billing with design
- ◆ Verification is another view on the design, not a side effort
- ◆ Verification provides check and balance



Copyright © 1998 Qualis Design Corporation

QUALIS™

Planning for Verification



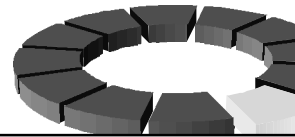
Copyright © 1998 Qualis Design Corporation

QUALIS™

Planning for Verification

- ◆ Need to know what kind of verification will be done (e.g. block-level, system-level, random)
- ◆ Verification is now a larger task than design
- ◆ Have to plan ahead or verification may be inefficient
- ◆ Keep in mind the different emphasis and goal of system-level testing
 - Check element interactions
 - Check corner cases
 - Skip functional testing of individual elements

Copyright © 1998 Qualis Design Corporation

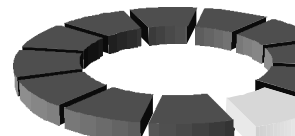


QUALIS™

Planning for Verification

- ◆ Plan for random testing if directed testing is incomplete or impractical
- ◆ Architect system verification structure just as much as a design - avoid unpleasant surprises...

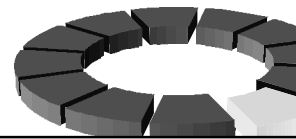
Copyright © 1998 Qualis Design Corporation



QUALIS™

Planning for Verification

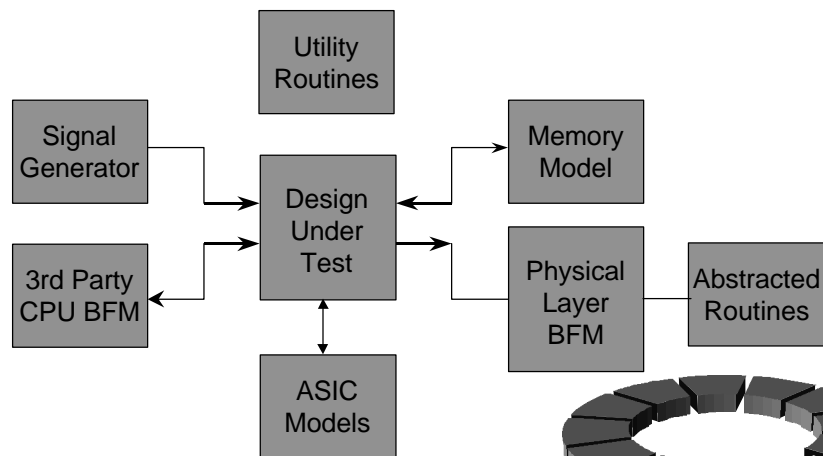
- ◆ Make sure all needed elements can be included in verification
 - Boundaries for different activities may be different (e.g. packaging, board design, verification)
 - Needed functionality may be on another board or in another ASIC
 - Simulation models - are they available?
- ◆ Look for ways to develop verification in parallel



Copyright © 1998 Qualis Design Corporation

QUALIS™

Test Environment Elements



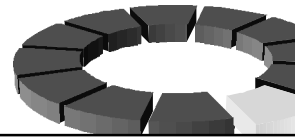
Copyright © 1998 Qualis Design Corporation

QUALIS™

Test Plan

- ◆ Describe test strategy and philosophy
- ◆ List all functions and features that should be tested
 - Keep in mind focus on interfaces and interactions
 - Emphasize stress tests and random tests
- ◆ Assign individual tests to cover the functions
- ◆ List assumptions, simplifications, and omissions for reference

Copyright © 1998 Qualis Design Corporation

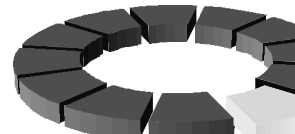


QUALIS™

Test Plan

- ◆ Just as important as the functional specification
- ◆ Re-statement of the functionality from another perspective
- ◆ System-level verification has no method to guarantee full functional coverage
 - Need to extract designer knowledge and extrapolate to system level
 - Code coverage can be used at block level, but not at system level due to black-box nature of testing

Copyright © 1998 Qualis Design Corporation

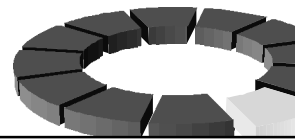


QUALIS™

Impact on Design

- ◆ May require access to internal nodes to aid black-box testing (e.g. diagnostic pins, status bits, internal self-test)
- ◆ Requires shift in emphasis to promote verification

Copyright © 1998 Qualis Design Corporation



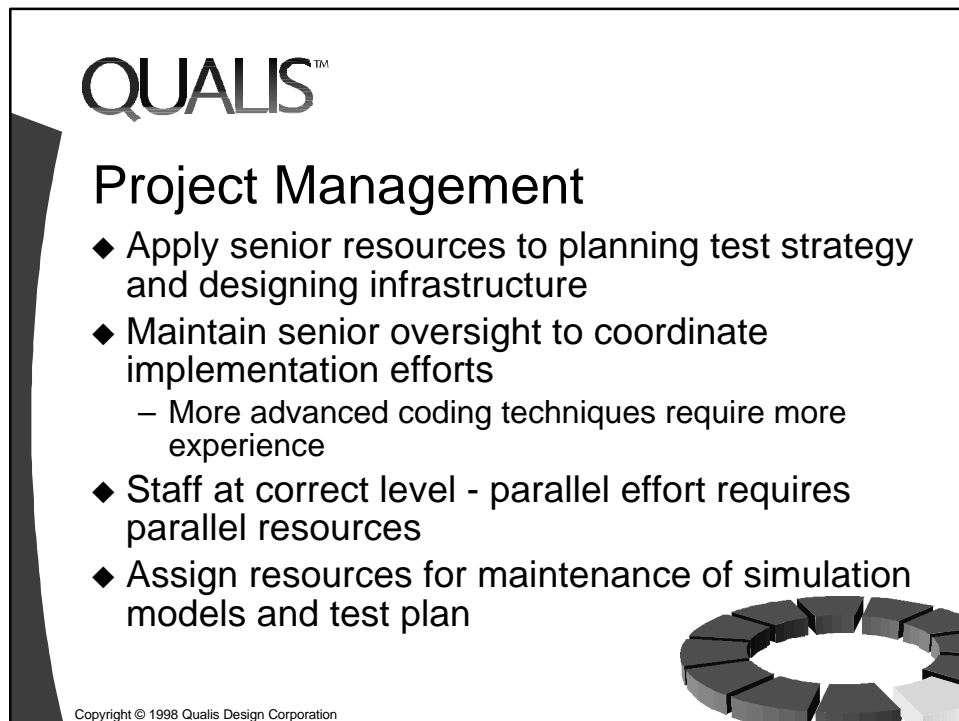
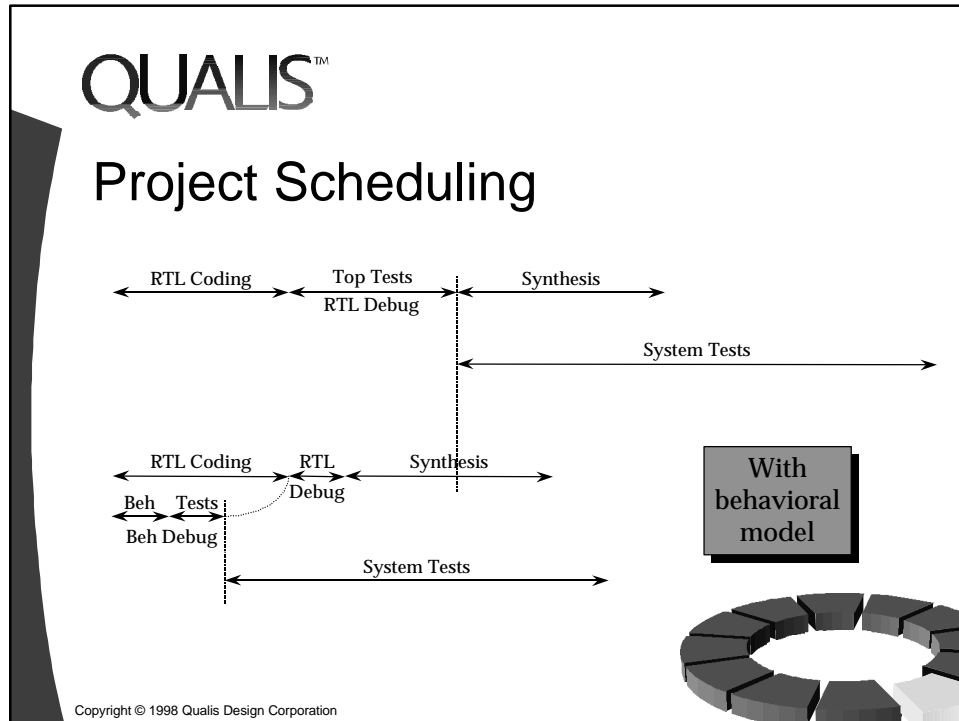
QUALIS™

Project Scheduling

- ◆ Make use of parallel development streams whenever possible
 - Behavioral/simulation models can allow test development in parallel with RTL coding


Copyright © 1998 Qualis Design Corporation





QUALIS™

A Proven
System Verification Method




Copyright © 1998 Qualis Design Corporation

QUALIS™

Proven System Verification

- ◆ Test plan
- ◆ Behavioral modeling
- ◆ Parallel effort
- ◆ Automated tests using test infrastructure
- ◆ Source control
- ◆ Directed and random tests



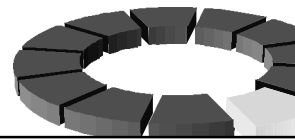
Copyright © 1998 Qualis Design Corporation

QUALIS™

Role of a Test Plan

- ◆ Coordinates verification effort
- ◆ Serves as the 'spec' for verification

Copyright © 1998 Qualis Design Corporation

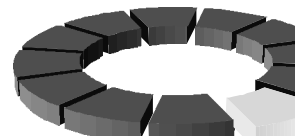


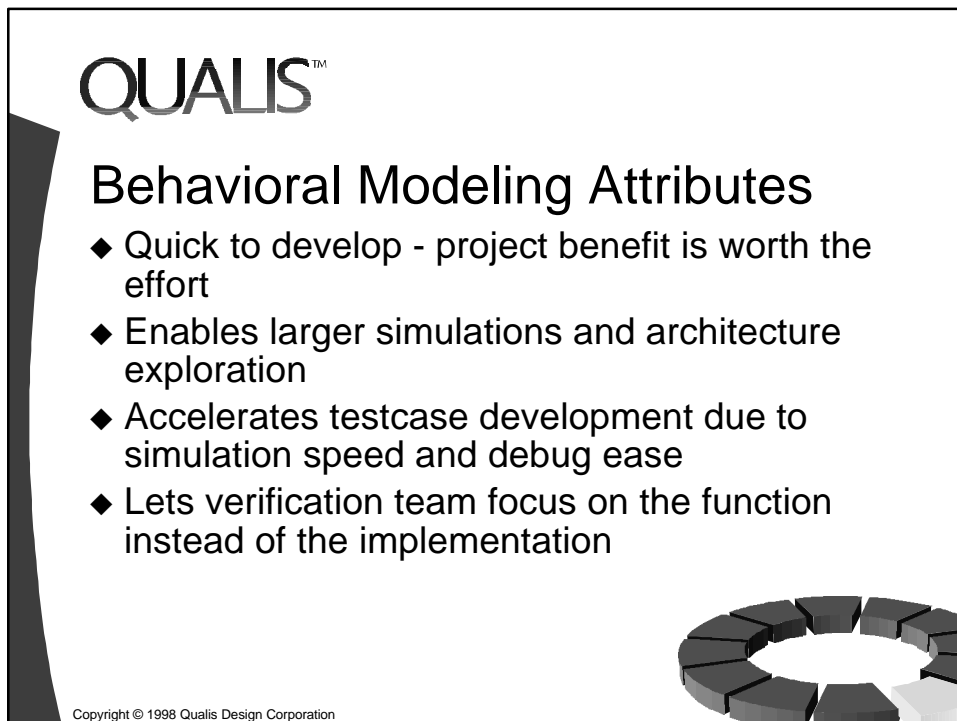
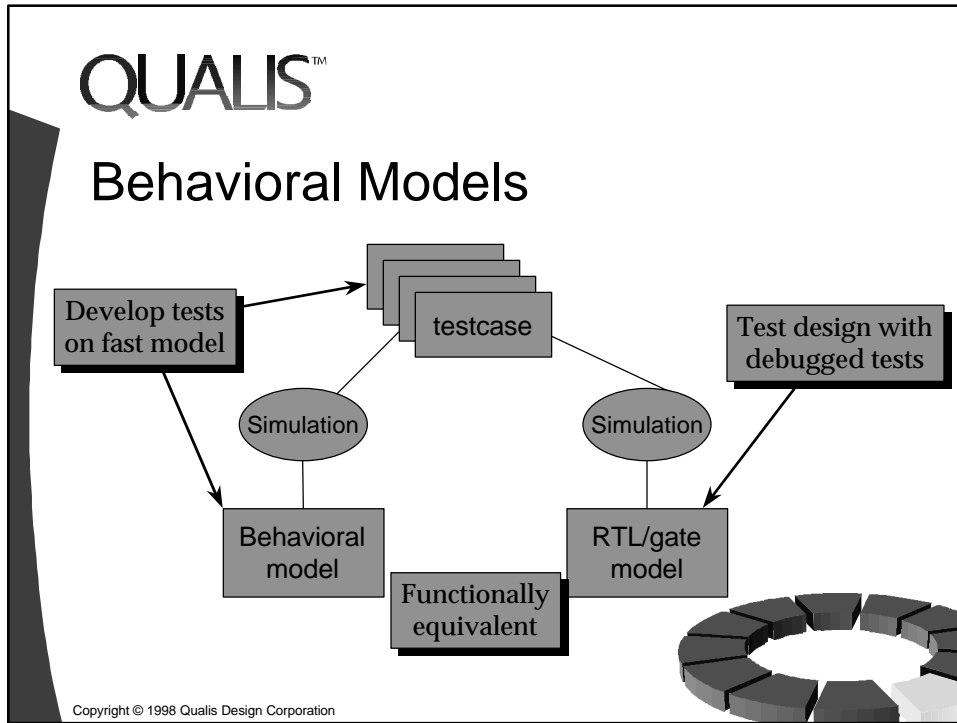
QUALIS™

What Is a Behavioral Model?

- ◆ Simulation model that captures the external functionality of a block
- ◆ Designed for simulation speed and debug ease
- ◆ May be organized and implemented completely differently from RTL because goals are different

Copyright © 1998 Qualis Design Corporation



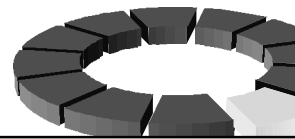


QUALIS™

Behavioral Modeling Attributes

- ◆ Allows parallel verification effort - models can be done early and quickly
- ◆ Can make use of speed-enhancing 'tricks' - not limited to synthesis subset
- ◆ Requires maintenance - models need to be kept up to date

Copyright © 1998 Qualis Design Corporation

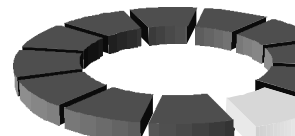


QUALIS™

Behavioral Modeling Tips

- ◆ Should overlap with design architecture
 - Think about behavioral model while design architecture is being developed
 - Code behavioral model as soon as architecture is stable. At this point, an efficient model can be developed quickly and will be the most useful to the project.
 - Starting too early may cause rewrites of the model
 - Starting too late will lose some of the scheduling benefit

Copyright © 1998 Qualis Design Corporation

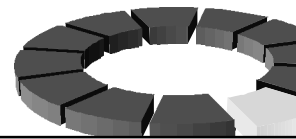


QUALIS™

Behavioral Modeling Tips

- ◆ Keep separate from design effort
 - Provides check and balance
 - Lets the model be implemented completely differently from the RTL
- ◆ Partition along lines that are likely to change and make it configurable to accommodate maintenance

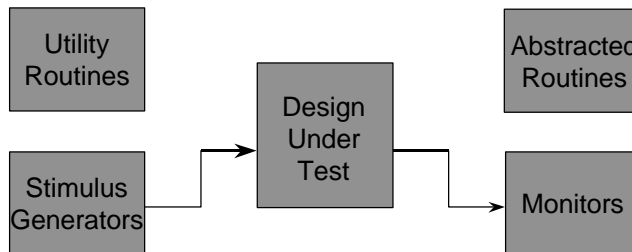
Copyright © 1998 Qualis Design Corporation



QUALIS™

Test Infrastructure Guidelines

- ◆ A collection of routines, models, and BFM's needed to test the system



Copyright © 1998 Qualis Design Corporation



QUALIS™

Test Infrastructure Guidelines

- ◆ Use top-down approach in designing test infrastructure to ensure that all needed functions will be supported
- ◆ Provide general purpose routines to facilitate test writing
- ◆ Keep routines simple - the test infrastructure should not be a major source of bugs!
- ◆ Should be robust enough to handle random cases - if you can handle random tests, you can handle any directed test

Copyright © 1998 Qualis Design Corporation

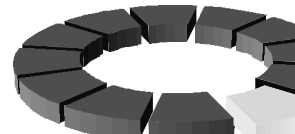


QUALIS™

Test Infrastructure Guidelines

- ◆ Tame the complexity of the system being verified with the test infrastructure
- ◆ Partition blocks so that general purpose routines can be reused
- ◆ Use layers and abstraction - build abstractions for common functions on top of physical layer routines
- ◆ Model low-level details so tests can be written logically

Copyright © 1998 Qualis Design Corporation

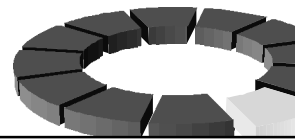


QUALIS™

Directed Tests

- ◆ Set up explicit conditions and perform a test for a particular function
- ◆ For system-level testing, detailed functional testing of sub blocks has been done
 - Use directed tests for basic sanity checks on major functional blocks
 - Use directed tests for basic system testing

Copyright © 1998 Qualis Design Corporation

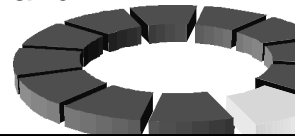


QUALIS™

Random Tests

- ◆ Generate 'random' stimulus to emulate real-world conditions
- ◆ Random testing is a superset of directed testing (a directed test is just one of the conditions that a random test might create)
- ◆ Great for exercising corner cases when there are too many conditions to use directed tests
- ◆ Let simulation time work for you - the longer the test runs, the more combinations and conditions are exercised

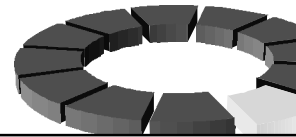
Copyright © 1998 Qualis Design Corporation



QUALIS™

Random Tests

- ◆ Require more advanced coding
 - Watch out for systematic omissions
 - Code cleanly - avoid side affects
 - Beware of simulation timing issues (e.g. event order dependency, concurrency, multiple task activation)
- ◆ Require more robust test infrastructure
- ◆ Require more advance planning and architecting - a poorly planned infrastructure is buggy, insufficient, and hard to use

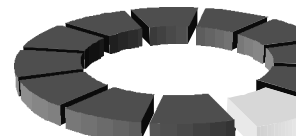


Copyright © 1998 Qualis Design Corporation

QUALIS™

Tool Requirements

- ◆ A lot can be done with HDLs - effective verification is achieved with advanced coding techniques
- ◆ Test languages provide an alternative to using HDLs
 - Another tool to buy and learn
 - Pre-bundled verification functions handle code 'complexities' for you



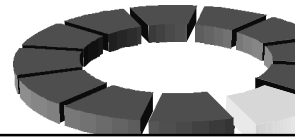
Copyright © 1998 Qualis Design Corporation

QUALIS™

Emulation as an Alternative

- ◆ Large overhead - setup and \$\$\$
- ◆ Execution speed advantage - use for very long tests, such as video
- ◆ Long time loop for iterations/edits - involves synthesis run
- ◆ Capacity issue - can you emulate a full system?
- ◆ Best to split test stimulus - BFM's in emulation and high-level control in simulation

Copyright © 1998 Qualis Design Corporation

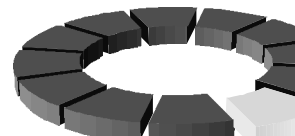


QUALIS™

Summary

- ◆ System-level verification is a necessity
- ◆ Driven by time-to-market, complexity, SoC, and IP-based designs
- ◆ Focused on protocols, interfaces, and interactions
- ◆ Cultural shift needed to recognize verification requirements throughout the design process
- ◆ Can be effective and efficient

Copyright © 1998 Qualis Design Corporation

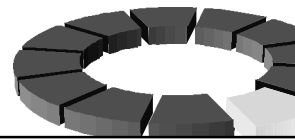


QUALIS™

Questions

- ◆ Can an emulator be integrated with an event-driven simulator?
- ◆ How do you handle analog blocks?

Copyright © 1998 Qualis Design Corporation



QUALIS™

More Information

- ◆ Check out our web site www.qualis.com:
 - Qualis Library pages have articles on system verification, system design, and design for reuse
 - Qualis Learning pages describe classes on design, verification, and synthesis of ultra complex systems
- ◆ Check out our training partner's web site, Mentor Graphics Customer Education and Training, www.mentorg.com/cet/
- ◆ Contact Janick Bergeron at Qualis:
 - Phone 503-670-7200 ext 544
 - Email janick@qualis.com

Copyright © 1998 Qualis Design Corporation

