Design, Verification, and Test of the ColdFire™ Embedded Microprocessors

Al Crouch
Jeff Freeman

Motorola, Inc.
6501 William Cannon Drive West
Austin, Texas 78735-8598

1997
Symposium

Design, Verification, and Test of the ColdFire Embedded Microprocessors

ABSTRACT

A new design methodology was developed and applied to the family of Motorola ColdFire Embedded Cores and Standard Products with particular application of new Design for Test techniques and tools. In addition to providing improved techniques for the integration of the ColdFire core in new applications, these methods provide new techniques to reduce overall product cycle time, increase quality measurement capability, and reduce the overall cost of test.

Author/Speaker

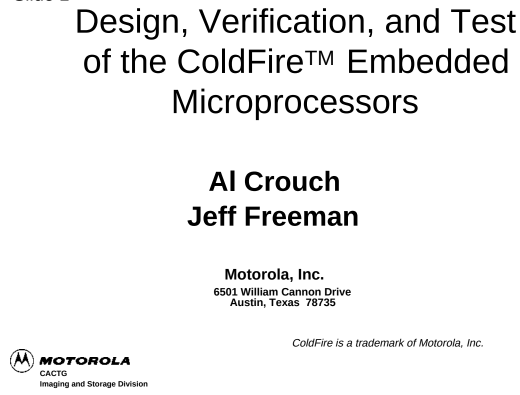
Author Background

Al has been involved with Test and Design-for-Test (DFT) for over 16 years.
He has been working at Motorola for over 5 years in the field of Microprocessor and Embedded Core DFT. He has been responsible for bringing aggressive DFT techniques to the 68060 Microprocessor and ColdFire Core-based products. At Motorola, he has largely been focused on JTAG, At-Speed Scan, Path Delay Scan, and Memory Built-in Self-Test (MBIST). He has published many articles on the same subjects, and holds several patents. Al received an M.S.E.E. from the University of Kentucky.

Jeff has 10 years of experience with both system design architectures and design process methodologies. For the first 4 years at Motorola, he was one of the architects on the 68040 pipeline unit. He did performance analysis and system design for this microprocessor. For the last 5 years, he served as a design methodology consultant and wrote many software tools to automate the design processes of the 68060 microprocessor and ColdFire microrocessors. He has written articles on test methodology, synthesis methodology, design methodology, and hardware emulation. The last year he has been part of the architecture design team. He received a B.S.E.E. from Texas A & M University.

Design, Verification, and Test of the ColdFire Embedded Microprocessors

Slide 1



Design, Verification, and Test of the ColdFire™ Embedded Microprocessors

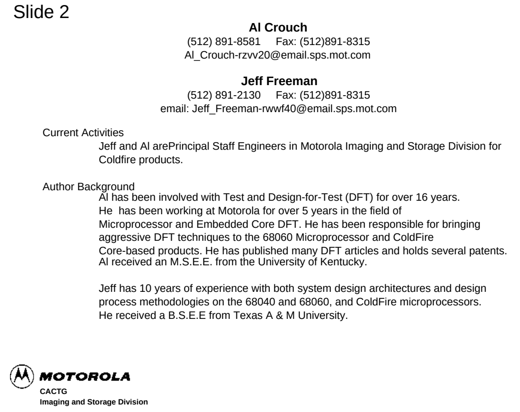
Al Crouch
Jeff Freeman

Motorola, Inc.
6501 William Cannon Drive
Austin, Texas 78735

ColdFire is a trademark of Motorola, Inc.

MOTOROLA
CACTG
Imaging and Storage Division

Slide 2



Al Crouch
(512) 891-8581 Fax: (512)891-8315
Al_Crouch-rzvv20@email.sps.mot.com

Jeff Freeman
(512) 891-2130 Fax: (512)891-8315
email: Jeff_Freeman-rwwf40@email.sps.mot.com

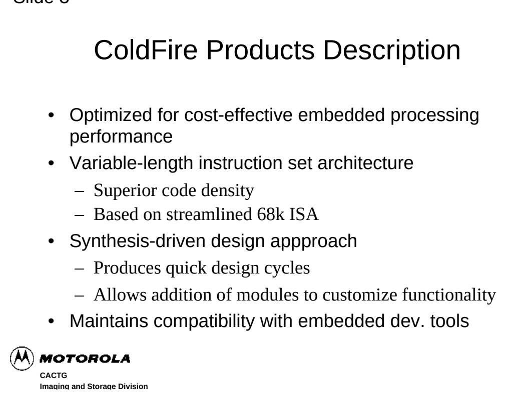
Current Activities
Jeff and Al arePrincipal Staff Engineers in Motorola Imaging and Storage Division for Coldfire products.

Author Background
Al has been involved with Test and Design-for-Test (DFT) for over 16 years. He has been working at Motorola for over 5 years in the field of Microprocessor and Embedded Core DFT. He has been responsible for bringing aggressive DFT techniques to the 68060 Microprocessor and ColdFire Core-based products. He has published many DFT articles and holds several patents. Al received an M.S.E.E. from the University of Kentucky.

Jeff has 10 years of experience with both system design architectures and design process methodologies on the 68040 and 68060, and ColdFire microprocessors. He received a B.S.E.E from Texas A & M University.

MOTOROLA
CACTG
Imaging and Storage Division

Slide 3



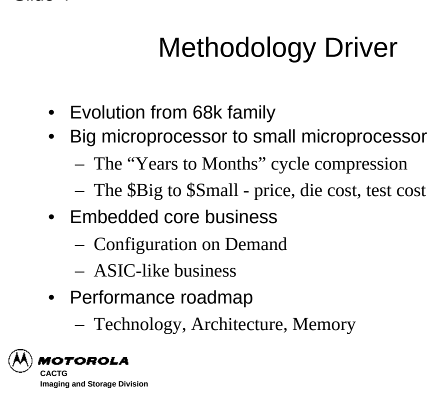
ColdFire Products Description

- Optimized for cost-effective embedded processing performance
- Variable-length instruction set architecture
  - Superior code density
  - Based on streamlined 68k ISA
- Synthesis-driven design appproach
  - Produces quick design cycles
  - Allows addition of modules to customize functionality
- Maintains compatibility with embedded dev. tools

MOTOROLA
CACTG
Imaging and Storage Division

The ColdFire products are targeted for embedded system applications. The variable length instruction set architecture (ISA) is optimized on a subset of the MC68000 ISA which maintains compatibility with embedded development tools.

Slide 4



Methodology Driver

- Evolution from 68k family
- Big microprocessor to small microprocessor
  - The “Years to Months” cycle compression
  - The $Big to $Small - price, die cost, test cost
- Embedded core business
  - Configuration on Demand
  - ASIC-like business
- Performance roadmap
  - Technology, Architecture, Memory

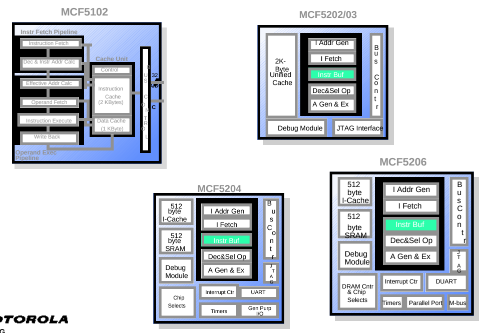
MOTOROLA
CACTG
Imaging and Storage Division

The ColdFire family of products came from the same design group that produced the 680X0 family of products. The target market for the 680x0 family was applications that used a general purpose microprocessor such as desktop computers. The ColdFire family is targeted for the embedded market. This market is extremely competitive and requires design cycle compression on the order of months instead of years. The ColdFire family is an ASIC-like design flow rather than the historical semi-custom flow.

Design, Verification, and Test of the ColdFire Embedded Microprocessors

Slide 5

Example of Standard Products in *ColdFire* Family



MOTOROLA
CACTG
Imaging and Storage Division

The ColdFire microprocessor architecture is optimized for high-performance low-cost embedded applications. The first ColdFire microprocessor, MCF5102, supports not only the ColdFire instruction set, but also the 680X0 instruction set. The MCF5202 is a 32-bit CPU core with 2k 4-way set associative unified cache and, as with all ColdFire microprocessors, contains real-time debug. The MCF5204 is comprised of a ColdFire core with a 512 direct mapped instruction cache and 512 SRAM. Integrated peripheral functions include a serial interface implemented as a programmable full-duplex UART, and two 16-bit general-purpose multimode timers. The MCF5206 is a highly integrated processor including a DRAM controller, timers, two independent UARTs, parallel port, and on-chip SRAM's.

Slide 6

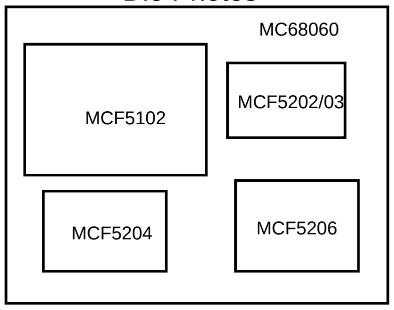
Roadmap

- Version 2: Standard
  - 5102, 5202, 5204, 5206
- Version 3: Clock Doubled : 8K Unified
- Version 4: Super Scalar 1 : Dual 8K
- Version 5: Super Scalar 2 : Dual 16K
- Version 6: Super Pipelined

MOTOROLA
CACTG
Imaging and Storage Division

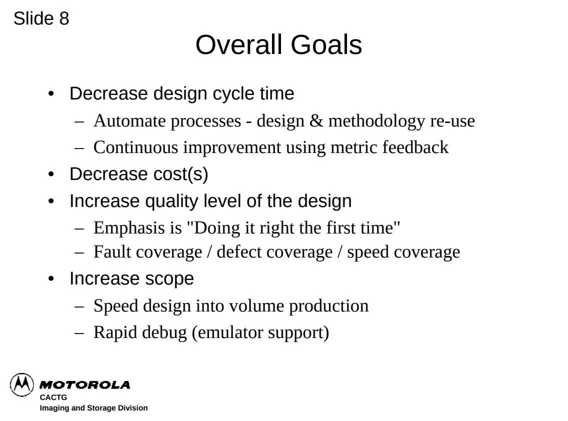
Slide 7

Die Photos



MOTOROLA
CACTG
Imaging and Storage Division

The slide presentation will compare actual die photos of the 68060 and the new ColdFire family. Notice the relative size of each of the example products.



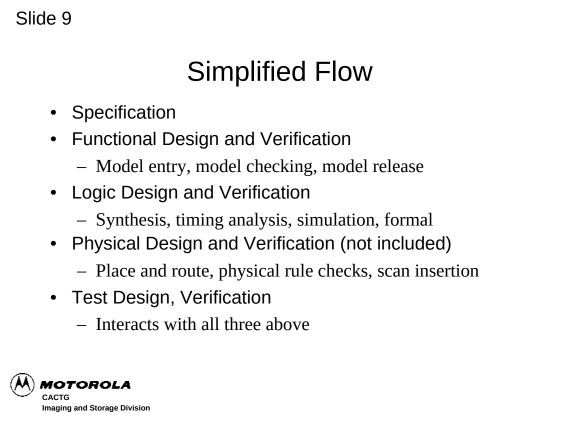
Slide 8

Overall Goals

- Decrease design cycle time
  - Automate processes - design & methodology re-use
  - Continuous improvement using metric feedback
- Decrease cost(s)
- Increase quality level of the design
  - Emphasis is "Doing it right the first time"
  - Fault coverage / defect coverage / speed coverage
- Increase scope
  - Speed design into volume production
  - Rapid debug (emulator support)

MOTOROLA
CACTG
Imaging and Storage Division

The design team was given a new set of challenges quite different from ones associated with the 680X0 family. The test costs must be minimized since the target selling price was an order of magnitude that of a desktop microprocessor. There is no room for design errors which means that processes must be in place to detect errors early in the design flow and give feedback to provide continuous improvement. The on-chip debug gives the customer the tools for debugging their embedded system.



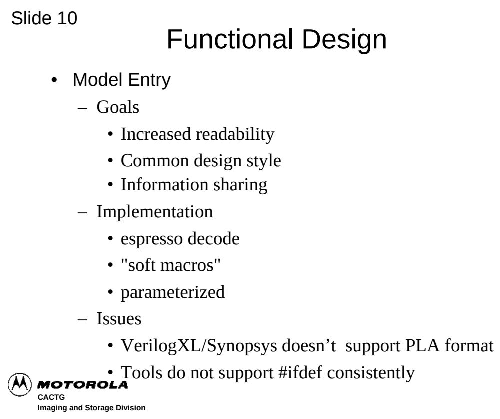
Slide 9

Simplified Flow

- Specification
- Functional Design and Verification
  - Model entry, model checking, model release
- Logic Design and Verification
  - Synthesis, timing analysis, simulation, formal
- Physical Design and Verification (not included)
  - Place and route, physical rule checks, scan insertion
- Test Design, Verification
  - Interacts with all three above

MOTOROLA
CACTG
Imaging and Storage Division

The simplified flow begins with the specification of a product based on customer and designer input. Once a specification has been drafted, work begins in creating a model of the design. Logic synthesis tools are used to translate this design description into logic based on Motorola designed standard cells. The first half of the proceedings describe the design and verification of ColdFire products and the last half describes the test and verification of these products. These proceedings will not include the physical design and verification of ColdFire cores.


Slide 10

Functional Design

- Model Entry
  - Goals
    - Increased readability
    - Common design style
    - Information sharing
  - Implementation
    - espresso decode
    - "soft macros"
    - parameterized
  - Issues
    - VerilogXL/Synopsys doesn't support PLA format
    - Tools do not support #ifdef consistently

MOTOROLA
CACTG
Imaging and Storage Division

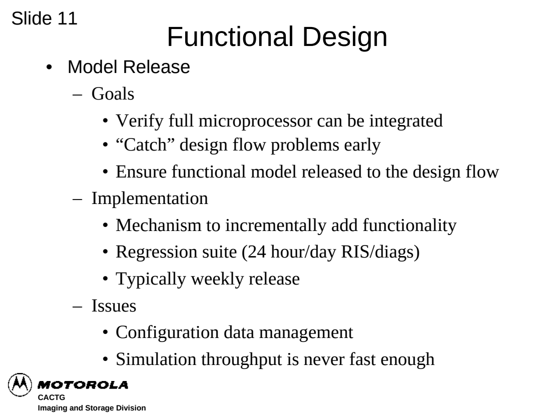
We use the Verilog-XL design language to describe our designs, more specifically, a synthesizeable subset of the Verilog-XL language. Third party tools were compatible because most of them supported the same common language subset we used. The internal cycle-based simulator we use (VCX) is optimized on this subset.

Portions of the decode logic are good candidates for PLA implementation. However, at least for now, the ColdFire architecture is fully-synthesizeable. Since Verilog does not support a PLA format, we chose to run some of our decode tables through espresso, and then insert the reduced logic equations into the Verilog-XL model. It may seem strange that we would do this, but we actually got better results during logic synthesis when these reduced logic equations were used by the logic synthesis tool, Synopsys.

The common building blocks such as registers, muxes, adders were coded into a centralized set of Verilog-XL files. All designers used the same common set of building blocks (soft macros). Thus, the readability of the design was increased since the model looked as if it were designed by one person. Designers could share information in a concise manner.

As the ColdFire family grew, we recognized that we needed a way to allow customers to optionally pick and choose hardware features. We use another internal preprocessor tool (MPP) that is more powerful than M4 or CPP (see Design Supercon'96 Rapid System Design Using a Parameterized-Synthesizable Module Library). For example, it allows us to insert a hardware divider, a MAC unit, or a different JTAG encoding by just setting a define.

Verilog-XL doesn't support a PLA format so implementing hardware tables is still challenging. We would like to be able to model at a higher-level of abstraction. Verilog-XL does not support a robust set of preprocessor constructs so we must continue to use our internal tool, MPP.



Slide 11

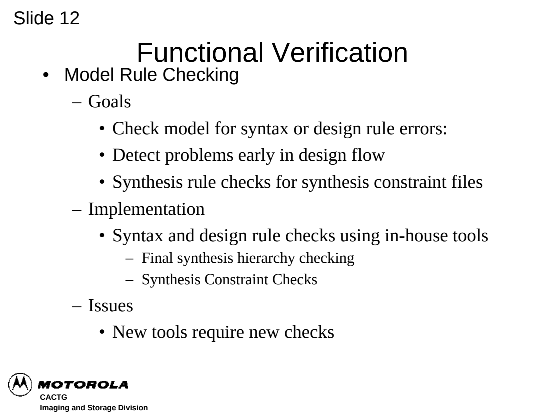
Functional Design

- Model Release
  - Goals
    - Verify full microprocessor can be integrated
    - "Catch" design flow problems early
    - Ensure functional model released to the design flow
  - Implementation
    - Mechanism to incrementally add functionality
    - Regression suite (24 hour/day RIS/diags)
    - Typically weekly release
  - Issues
    - Configuration data management
    - Simulation throughput is never fast enough

MOTOROLA
CACTG
Imaging and Storage Division

The full integration and release of the microprocessor encompasses pulling together all the sub-modules and performing model syntax checks and a simulation regression suite. This ensures that a syntactically and functionally correct design is released into the rest of the design flow steps. Typically, at the beginning of the design cycle, we try to release an initial baseline model as soon as possible, even if all the functionality is not there. We scaffold in incremental updates during weekly model releases.

We have a very good internally written configuration management system for tracking our design files and model releases. However, it lacks features that would really help. A few examples are links to a bug tracking system and automatic notification. The model release is typically done by one person. Designers fill out on-line model update entries to notify that person what should go in the next release.



Slide 12

Functional Verification

- Model Rule Checking
  - Goals
    - Check model for syntax or design rule errors:
    - Detect problems early in design flow
    - Synthesis rule checks for synthesis constraint files
  - Implementation
    - Syntax and design rule checks using in-house tools
      - Final synthesis hierarchy checking
      - Synthesis Constraint Checks
  - Issues
    - New tools require new checks

MOTOROLA
CACTG
Imaging and Storage Division

Checking the model for syntax errors provides a mechanism to catch syntax problems before releasing a model. Potentially, these errors may not be caught until much later in the flow (e.g. a Verilog-XL construct that may be allowed in simulation but cannot be synthesized in Synopsys).

The model checking step has proved very useful for adding checks at the front-end of the design flow to catch errors that otherwise would not be found until later. We continuously add new checks to this process. Multiple times this has saved several during the development of ColdFire products. Examples for design rule checks are bus contention during scan shifting, mutual exclusion checks, and multiple driver checks.

By introducing some synthesis checks as pre-requisites for releasing a model, simple errors can be caught in the model phase instead of the synthesis phase of the design process. In software terms, this is known as phase containment.

This first check is to make sure that the synthesis tool can read each model file without encountering any syntax or unsupported language constructs.

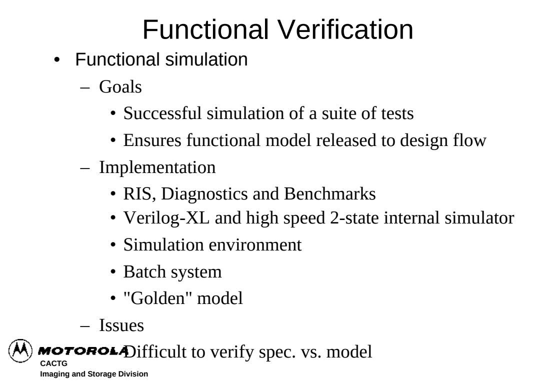
For Synopsys tools, we just use a script that reads in the Verilog-XL, and checks the status variable to validate the Verilog-XL design.

The second check is to ensure that the chip hierarchy is consistent and that there are no unconnected inputs. For this process, we have an internal Motorola tool that parses the Verilog-XL.

The third and final check is to validate the synthesis constraint files. We have a Perl script that parses the Verilog-XL and the corresponding synthesis constraint file checking that all ports have a constraint, and all port names referenced in the synthesis constraint file match the port names in the Verilog-XL model. To keep the methodology uniform across projects, this script reports errors for synthesis commands that are not recommended.

Any time new tools are added to the design flow, new design rule checks are required during the model rule checking step.

Slide 13



Functional Verification

- Functional simulation
  - Goals
    - Successful simulation of a suite of tests
    - Ensures functional model released to design flow
  - Implementation
    - RIS, Diagnostics and Benchmarks
    - Verilog-XL and high speed 2-state internal simulator
    - Simulation environment
    - Batch system
    - "Golden" model
  - Issues
    - Difficult to verify spec. vs. model

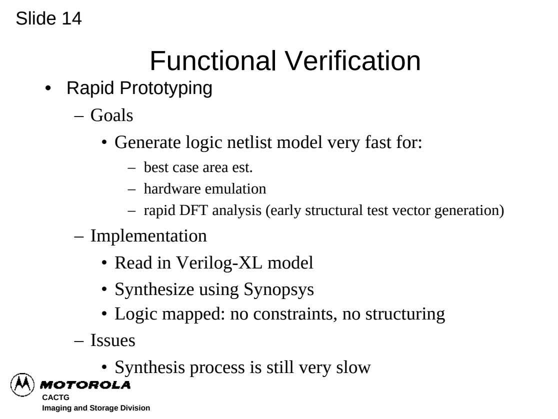
MOTOROLA CACTG Imaging and Storage Division

A suite of regression tests including benchmarks, diagnostics and random instruction sequences (RIS) are run on ColdFire cores before a model is released. This regression suite provides a limited set of tests to functionally verify the model release. After the regression suite has passed, RIS tests are continually run on the released model 24 hours per day on the entire network of workstations. This is accomplished via a batch system running primarily VCX. A smaller set of workstations are used to run Verilog-XL simulations. These RIS tests expose the model to more random situations and increase the overall confidence that the design is correct. The expected results are generated by a "golden model" which can be either a prior microprocessor in the family or an independently written software model.

The simulation environment used by the designers during simulation of ColdFire cores is a complete system simulation environment. This system model comprises, the microprocessor, external bus, peripherals, external memory, and an external development system model to stimulate the on-chip debug module. Programs consisting of the real code are loaded into the memory and executed. For the development system debug commands such as breakpoints, read/writes to registers or memory loaded and run. This environment can be used to fully create most application situations and has proved extremely valuable in the development of ColdFire products.

In addition to functionality checking, the modeled test logic is also simulated to ensure that it operates as expected. Most of the test logic is included in the model to ensure that it does not interfere with the normal functional mode of the chip.

But it is still a challenge to verify the model against the original specification.



Slide 14

Functional Verification

- Rapid Prototyping
  - Goals
    - Generate logic netlist model very fast for:
      - best case area est.
      - hardware emulation
      - rapid DFT analysis (early structural test vector generation)
  - Implementation
    - Read in Verilog-XL model
    - Synthesize using Synopsys
    - Logic mapped: no constraints, no structuring
  - Issues
    - Synthesis process is still very slow

MOTOROLA
CACTG
Imaging and Storage Division

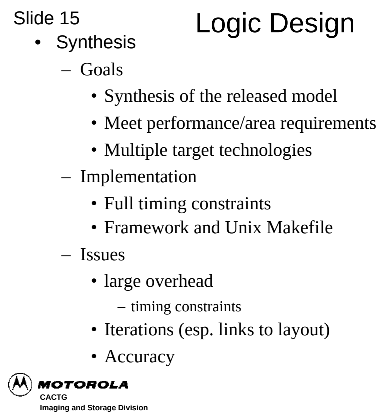
Rapid-prototyping generates a logic netlist from a Verilog-XL model very fast. The synthesis is not timing constrained so the final netlist may be twice as large as a timing constrained synthesis run, but this is not a problem. There are several applications to doing rapid-prototyping. First, a quick path is now possible by going from Verilog-XL to a hardware emulator. Another application is doing rapid design for test (DFT) analysis earlier in the design cycle, instead of waiting for a final physical netlist. Finally, this logic netlist can be used to flush out problems with back-end tools (early chip build).

The script we use reads in the Verilog-XL model using Synopsys. Each module in the design hierarchy is compiled if there is logic in that module. For modules that are wire-only, no compilation is needed. the fastest approach to getting a netlist is accomplished by compiling the Verilog-XL with a low Synopsys compile effort setting, no design rule fixing, and absolutely no synthesis timing constraints are applied.

Essentially, the Verilog-XL model is just mapped to a target technology with all synthesis algorithms turned off. A logic netlist is written out to be used by the various applications.

Rapid DFT analysis will verify all of the test architecture and design for test assumptions. This is done by modeling most of the test structures in parallel with the functional design and operating these structures in the simulator. The only test structures not modeled are the scan chain connections since these are physical placement specific (our scan chains are optimized for nearest-neighbor routing to reduce scan route/metal overhead and to ensure at-speed operation). The rapid prototype gate level netlist goes through design rule checking with a static and dynamic tool (more on this later).

The scan vector generation tool is used as the golden fault standard, so the vectors for JTAG and Memory Test are passed through the Mentor Flextest tool for fault measurement against the one fault database. Architectural problems that reduce fault coverage are identified and fixes or workarounds are proposed and
added to the next model update.


Slide 15

Logic Design

- Synthesis
  - Goals
    - Synthesis of the released model
    - Meet performance/area requirements
    - Multiple target technologies
  - Implementation
    - Full timing constraints
    - Framework and Unix Makefile
  - Issues
    - large overhead
      - timing constraints
    - Iterations (esp. links to layout)
    - Accuracy

MOTOROLA
CACTG
Imaging and Storage Division

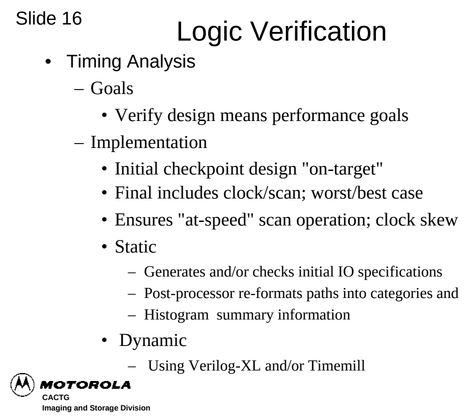
The synthesis phase of the design should not be anything new to designers. We try to write technology independent code in our models so that we can target multiple process technologies.

We continue to use the same synthesis framework tool that was described in the Design Supercon'95 presentation on 68060 microprocessor methodology. This framework uses a Unix Makefile and a highly structured approach to build a logic netlist based on a set of timing constraints from the designer. This framework allows the designer to select compile methodologies and select final design hierarchy of the chip. Once the timing constraints and the configuration file that controls the compile methodology and final chip hierarchy are available, the synthesis process is "push button". For example, a designer simply types "make DESIGN-map" where DESIGN is the name of the top level modules of the chip hierarchy. By keeping it simple, we have established a standardized and repeatable process for the product groups to use within Motorola.

An optional pre-layout in-place optimization is done using Synopsys, allowing gates to be resized and buffering to be added where needed. Sometimes the iterations between synthesis and layout are reduced using this method.

Timing constraints are required for a high-performance design, but a lot of work is required to create and maintain them. This is a very large overhead and we look forward to tools that will make this process easier. For now, we trade off the number of constraint files with the results we get in the synthesis process and the synthesis runtimes. On the MC68060 we had hundreds of constraint files, but for the ColdFire family we have tens of constraint files.

Iterations between the synthesis process and the layout process continue to exist. Depending on the performance goals, we may either use a conservative wire-load model (Synopsys capacitance/resistance estimate based on fanout), an aggressive wire-load model, or more time-consuming recent layout based wire-loads models.


Slide 16

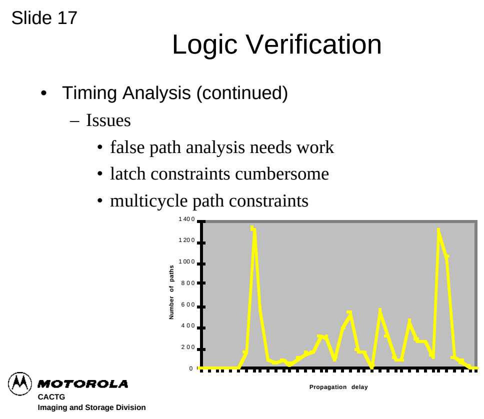
Logic Verification

- Timing Analysis
  - Goals
    - Verify design means performance goals
  - Implementation
    - Initial checkpoint design "on-target"
    - Final includes clock/scan; worst/best case
    - Ensures "at-speed" scan operation; clock skew
    - Static
      - Generates and/or checks initial IO specifications
      - Post-processor re-formats paths into categories and
      - Histogram summary information
    - Dynamic
      - Using Verilog-XL and/or Timemill

MOTOROLA
CACTG
Imaging and Storage Division

Static and dynamic timing analysis provide feedback as to whether the design is meeting the specifications (clock cycle time and IO specifications). Static timing analysis is faster and easier to manage than dynamic timing analysis. It is also invaluable to the design flow. However, the design team has found that when "design tricks" are used, because they are judged to be necessary for a given product, then static timing analysis can miss some problems and leave potential design flaws undetected in the design. In order to overcome these shortcomings we have added dynamic timing analysis to the design flow.

Timing analysis begins with wire-load table estimates, then after layout, parasitics are back-annotated to the logic netlist for timing. For final timing analysis, the clock and scan trees are timed (skew analysis) with the rest of the logic. This is done at both worst and best operating points. At-speed scan operation is verified. In the following slide, an example histogram is shown. We use this to gauge how much more timing work is required.



Slide 17

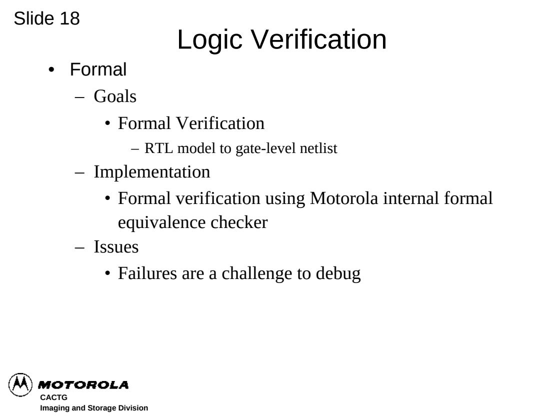
Logic Verification

- Timing Analysis (continued)
  - Issues
    - false path analysis needs work
    - latch constraints cumbersome
    - multicycle path constraints

1400
1200
1000
800
600
400
200
0
Number of paths
Propagation delay

MOTOROLA
CACTG
Imaging and Storage Division

One of the biggest obstacles in static timing analysis is dealing with false paths. This is especially true when doing “at-speed” scan pattern generation. A lot of iteration with the ATPG tool is required to get a “valid” path to test. Constraining latches and setting multicycle path constraints are cumbersome, at least with the timing tool we are using.



Slide 18

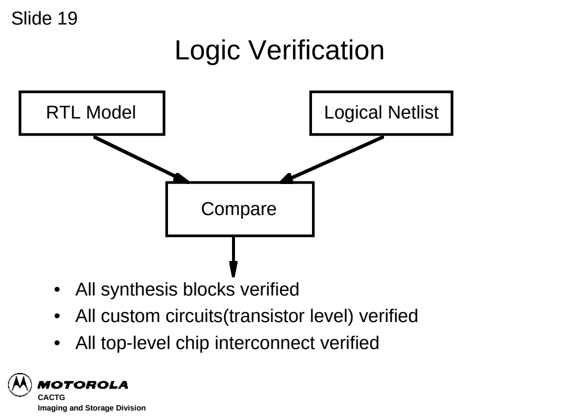
Logic Verification

- Formal
  - Goals
    - Formal Verification
      - RTL model to gate-level netlist
  - Implementation
    - Formal verification using Motorola internal formal equivalence checker
  - Issues
    - Failures are a challenge to debug

MOTOROLA
CACTG
Imaging and Storage Division

Formal equivalence checking is very useful in the design flow for proving that two descriptions are 100% logically equal.

We presented our use of formal equivalence at Design SuperCon’95, 68060 Microprocessor Logic Design Methodology. Since then we now use a more robust internal Motorola tool.



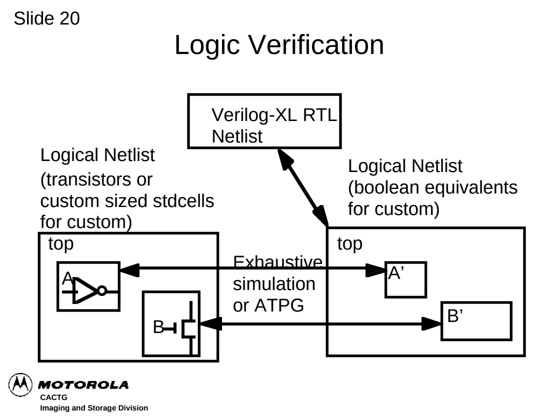
Slide 19

Logic Verification

- All synthesis blocks verified
- All custom circuits(transistor level) verified
- All top-level chip interconnect verified

All synthesized blocks are compared to the original model using the formal verification tool. We have in fact caught several synthesis tool logic errors recently using this verification. This methodology is also good for last-minute changes or incremental changes to the logic netlist.

If the formal verification is being done on the final netlist, clock and scan tree connections are verified.



Slide 20

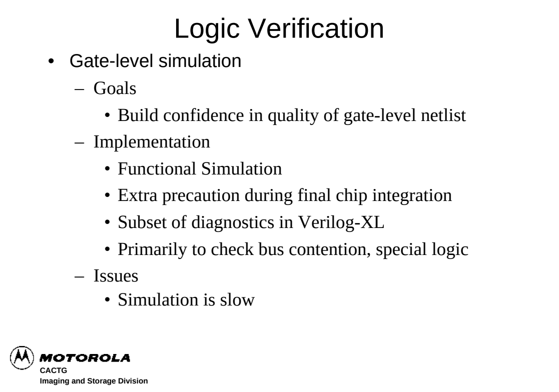
Logic Verification

Verilog-XL RTL Netlist
Logical Netlist (transistors or custom sized stdcells for custom)
Logical Netlist (boolean equivalents for custom)
top
top
A
B
A'
B'
Exhaustive simulation or ATPG
MOTOROLA
CACTG
Imaging and Storage Division

The transistor-based standard cells and custom cells are verified by exhaustive simulation or ATPG simulation. Thus, we have a 100% confidence that the logic netlist matches the original model. This process can also be applied to custom logic.

The primary issue with formal equivalence is debug. If there is a logic design error, the tool gives an input vector showing the difference. However, it would be better to narrow the scope by identifying the specific gates that have the problem.

Slide 21



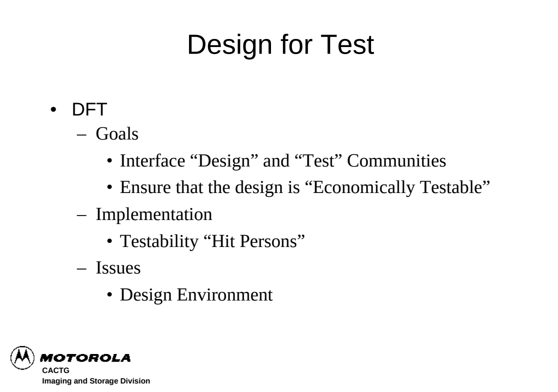
Logic Verification

- Gate-level simulation
  - Goals
    - Build confidence in quality of gate-level netlist
  - Implementation
    - Functional Simulation
    - Extra precaution during final chip integration
    - Subset of diagnostics in Verilog-XL
    - Primarily to check bus contention, special logic
  - Issues
    - Simulation is slow

MOTOROLA
CACTG
Imaging and Storage Division

Formal verification does not cover dynamic conditions such as bus contention. So, a subset of diagnostics are run on the final gate-level netlist to verify functionality and do timing checks (dynamic timing). This simulation is slow without the the aid of a hardware accelerator.

Slide 22



Design for Test

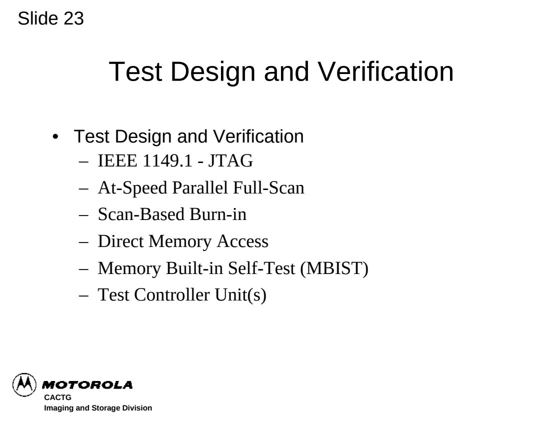
- DFT
  - Goals
    - Interface “Design” and “Test” Communities
    - Ensure that the design is “Economically Testable”
  - Implementation
    - Testability “Hit Persons”
  - Issues
    - Design Environment

MOTOROLA
CACTG
Imaging and Storage Division

The design methodology meet the overall design goals for reducing cycle time and optimizing design budgets (area, timing, power, etc.), but if it does not address test and the “cost of test”, then it is only half of a methodology as far as the small microprocessor and embedded core market is concerned. Our change of business from “Big Microprocessor” to cost-effective “Small Microprocessor or Embedded Core”, has made the “cost of test” and the ability to even apply test a more predominant issue.
The DFT problem has gotten worse in the Core business because a complex microprocessor, which is difficult to test in the first place, is made a bigger problem by embedding within a chip (surrounding the core by logic that is not part of the microprocessor, and not providing natural/functional access to package pins).

The way we ensured that test issues are addressed is to create a communication link between the design community/design methodology, and the test community. The communication link addresses the needs of the tester, design methodology limitations, and the cost issues involved. Another powerful technique we apply is to create a strategy and methodology for test similar to the standardized-optimized design methodology (e.g., we created a standardized-optimized test methodology).

In the case of ColdFire products, the communication link is a group of dedicated Design-for-Test (DFT) engineers. This works out well since the biggest issue with the design process is “design” (the constantly changing logic and tendency to stretch the technological limits).


Slide 23

Test Design and Verification

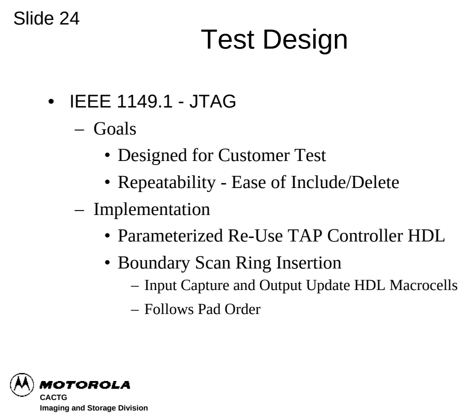
- Test Design and Verification
  - IEEE 1149.1 - JTAG
  - At-Speed Parallel Full-Scan
  - Scan-Based Burn-in
  - Direct Memory Access
  - Memory Built-in Self-Test (MBIST)
  - Test Controller Unit(s)

MOTOROLA
CACTG
Imaging and Storage Division

The ColdFire Cores and Standard Products are made with a standardized, repeatable test strategy-methodology. This test strategy addresses all test environments: manufacturing defect test for general combinational and sequential logic; manufacturing test for memory array circuitry; frequency, pin timing, and parametric verification; engineering debug and characterization; burn-in and life cycle testing; and customer or end user test.

The test strategy is implemented by including on-chip test architectures for JTAG (IEEE 1149.1), Scan, and Memory Test. All of the test features are organized, prioritized, and controlled by including multiple Test Controller units (multiple hierarchical units are needed to differentiate embedded cores from standard products).

The real challenge is to provide these test features in a silicon optimized form, without impacting the design schedule, and to make these features accessible when the core is embedded.



Slide 24

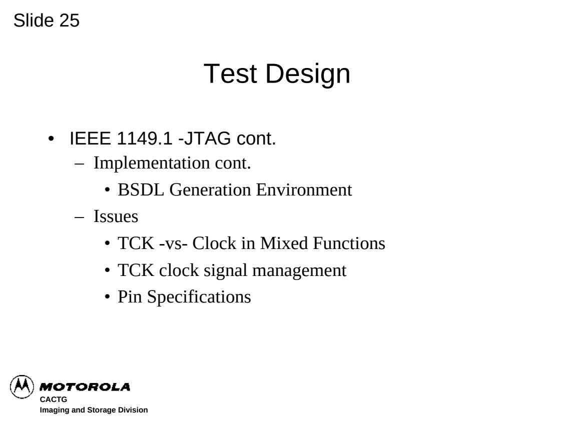
Test Design

- IEEE 1149.1 - JTAG
  - Goals
    - Designed for Customer Test
    - Repeatability - Ease of Include/Delete
  - Implementation
    - Parameterized Re-Use TAP Controller HDL
    - Boundary Scan Ring Insertion
      - Input Capture and Output Update HDL Macrocells
      - Follows Pad Order

MOTOROLA
CACTG
Imaging and Storage Division

For the customer test environment, standard products include logic that is compliant with the IEEE 1149.1 Standard (also known as JTAG). Since Core products may not include JTAG logic (it is generally only used at the package-pin-level), the methodology had to allow the ability to easily include or remove the JTAG logic from the model. The JTAG logic is a pre-existing re-use register transfer level (RTL) model of a TAP controller with preprocessor (MPP) parameters that allows the addition of instructions above and beyond the required Extest, Bypass, and Sample-Preload (e.g., idcode, hiz, clamp). This can be done because the JTAG is based on a standard and does not need to change from part to part.

The boundary scan ring is placement optimized, so the boundary scan cell order is the same as the pad order. To make boundary scan insertion and understanding of the boundary scan ring simple and easy, the boundary scan ring is made by stitching together input capture cells, and output update cells that are represented as single hardware description language (HDL) model entities (as opposed to loose or synthesized logic components).



Slide 25

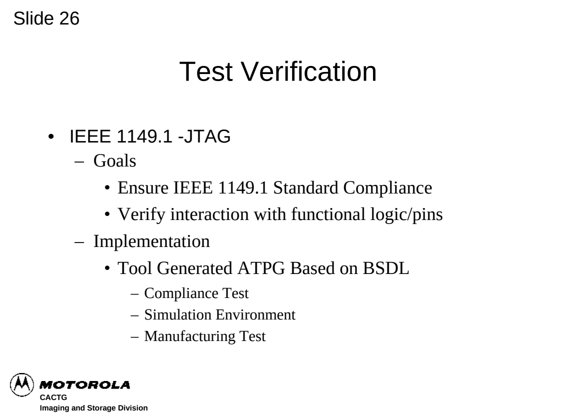
Test Design

- IEEE 1149.1 -JTAG cont.
  - Implementation cont.
    - BSDL Generation Environment
  - Issues
    - TCK -vs- Clock in Mixed Functions
    - TCK clock signal management
    - Pin Specifications

MOTOROLA
CACTG
Imaging and Storage Division

Once the JTAG part of the model is complete, the Boundary Scan Description Language (BSDL) file must be developed as a deliverable to the customer, and as a resource for pattern generation. The customer can then use this file to understand the configuration of the JTAG logic, or for ATPG. An internal tool known as MJTAG has an environment that is used to quickly generate the BSDL.

There are some issues with the current JTAG design methodology, but they are being addressed. One problem is in the ability to smoothly conduct chip-wide static timing analysis since TCK has no defined relationship to the system clock. Related to this is the fact that there is no “fixed” frequency defined in the standard. Not correctly constraining the Synthesis tool can result in a weak clock tree (too many loads -- not enough drive strength).
A specific effort must also be made to timing analyze the clock -vs- shift path (clock skew analysis) to ensure that no data race conditions will result. A fixed frequency and pin timing specification was developed mostly for the synthesis process to ensure that constraints were developed for the JTAG to meet “a” target frequency and the pin specifications (operating without a specification means leaving the JTAG logic unconstrained).



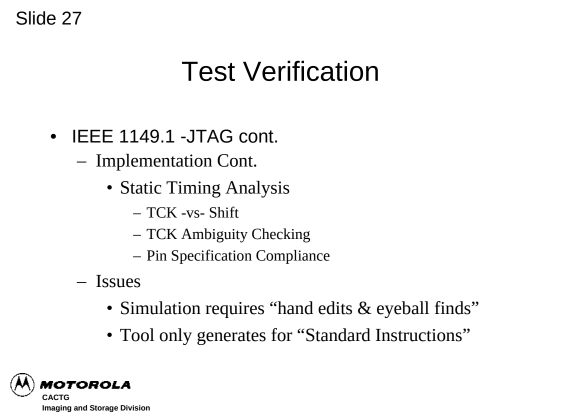
Slide 26

Test Verification

- IEEE 1149.1 -JTAG
  - Goals
    - Ensure IEEE 1149.1 Standard Compliance
    - Verify interaction with functional logic/pins
  - Implementation
    - Tool Generated ATPG Based on BSDL
      - Compliance Test
      - Simulation Environment
      - Manufacturing Test

MOTOROLA
CACTG
Imaging and Storage Division

After the JTAG logic is modeled (designed), it must be verified just as any other logic block included in the chip must be verified (functional operation and frequency). However, the JTAG is somewhat different in that an IEEE standard exists (1149.1) and functional verification means "standard compliance". At the chip-level, the JTAG controller and the boundary scan ring must also be shown to correctly interact with the chip logic and the package pins.
To meet these goals, the BSDL is used to generate several test patterns: a compliance pattern; a manufacturing-production test parametric pattern, tristate leakage pattern, and fault-defect coverage pattern. These patterns are then applied to the RTL model, and to the gate-level netlist after synthesis. These patterns are also delivered to the tester as possible production test patterns after passing through a translation step.



Slide 27

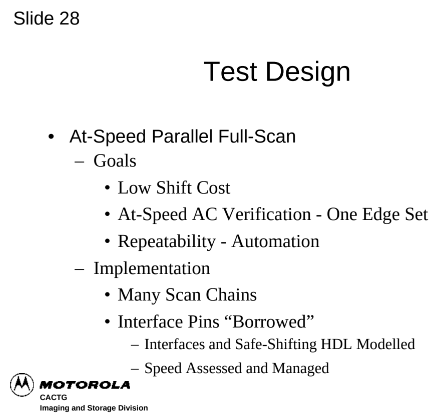
Test Verification

- IEEE 1149.1 -JTAG cont.
  - Implementation Cont.
    - Static Timing Analysis
      - TCK -vs- Shift
      - TCK Ambiguity Checking
      - Pin Specification Compliance
  - Issues
    - Simulation requires "hand edits & eyeball finds"
    - Tool only generates for "Standard Instructions"

MOTOROLA
CACTG
Imaging and Storage Division

Aside from simulation based verification, the JTAG must be timing analyzed. There are several reasons to do this: to conduct a clock skew assessment to ensure that there are no shift races; to ensure that the internal JTAG logic makes the design frequency goals; and to ensure that the pins meet their specifications (clock-to-out, input setup).

Although the verification is largely automated by relying on BSDL generation and ATPG against that BSDL, there are still manual, or scripted, tasks required from each different chip implementation. These are mostly from having to find certain internal nodes in the synthesized logic as display nodes for boundary scan ring verification. There is also a shortcoming in that the ATPG tool can only generate vectors for the "standard instructions" that are described in the IEEE standard. There is no method available to fully explain the intention or operation of a private or custom instruction with just the BSDL description.


Slide 28

Test Design

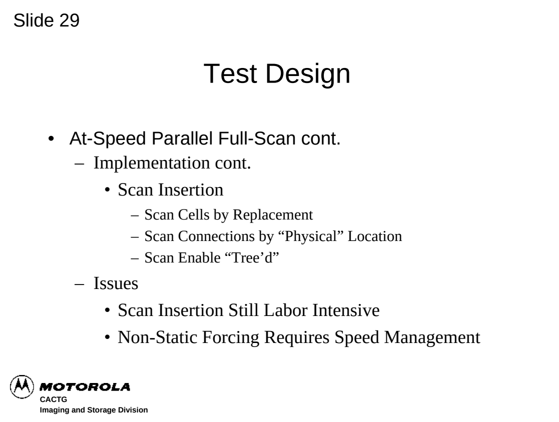
- At-Speed Parallel Full-Scan
  - Goals
    - Low Shift Cost
    - At-Speed AC Verification - One Edge Set
    - Repeatability - Automation
  - Implementation
    - Many Scan Chains
    - Interface Pins “Borrowed”
      - Interfaces and Safe-Shifting HDL Modelled
      - Speed Assessed and Managed

MOTOROLA
CACTG
Imaging and Storage Division

All general combinational and sequential logic is tested by scan. The key goal in supporting a scan environment is to make the “test cost” a lesser expense than using functional vectors (higher coverage and fewer clock cycles or tester memory space used). To this end we like to have optimized scan vectors with a shift depth of about 100 clock cycles.

Another test cost cutting technique is to support only one tester edge set. This is done by architecting the scan in such a way so that the tester pin timing is identical in scan mode to that used in functional mode. The scan interface and the internal scan must also support the ability to operate at, or above, the target functional frequency. The most key goal here is to make the ability to include “at-speed”, “one edge set” scan a simple exercise to the design staff with minimal impact to the design schedule.

These goals are implemented by “borrowing” many functional pins to be used as the scan interface, and maintaining the functional pin timing during scan mode. This has resulted in various numbers of scan chains being used on different ColdFire products (e.g., between 25 and 50 scan chains). To ensure that the scan interfaces will make “at-speed timing”, the scan input and output interfaces and the dynamic safe-shifting logic are modeled, synthesized, and timing analyzed with the rest of the chip.



Slide 29

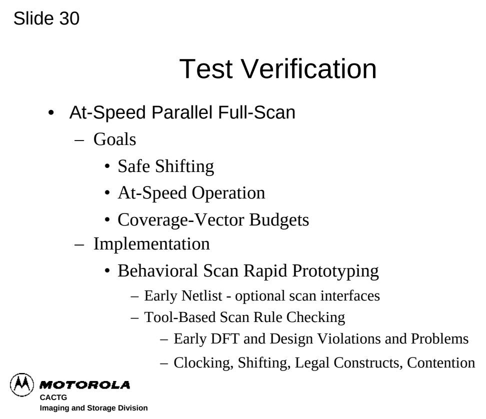
Test Design

- At-Speed Parallel Full-Scan cont.
  - Implementation cont.
    - Scan Insertion
      - Scan Cells by Replacement
      - Scan Connections by “Physical” Location
      - Scan Enable “Tree’d”
  - Issues
    - Scan Insertion Still Labor Intensive
    - Non-Static Forcing Requires Speed Management

MOTOROLA
CACTG
Imaging and Storage Division

The model passes through synthesis with only the scan interfaces and shift safety logic modeled. The scan cells, scan data connections (SDI-SDO), and scan control (SE) are netlist-physical scan inserted. To ensure that the scan connections will make speed, they are connected in an optimal fashion based on physical location. This minimizes metal usage and relieves route congestion. The scan enable also becomes a critical timing path for at-speed operation. The SE signal is treated as a clock tree to manage the skew and delay across the chip.

There are still some issues with this methodology. The scan insertion process requires extracting layout information and inserting it into the netlist prior to routing. Clock tree and SE tree insertion is also a labor intensive process.
One of the issues of using at-speed scan for AC verification has to do with the de-assertion of the scan shift state. A scan sample must occur when the logic is in an almost functional state. This prohibits the use of “static forcing functions” such as scan mode, and requires dynamic control signals. This is the reason that the scan architecture must be speed assessed and managed.


Slide 30

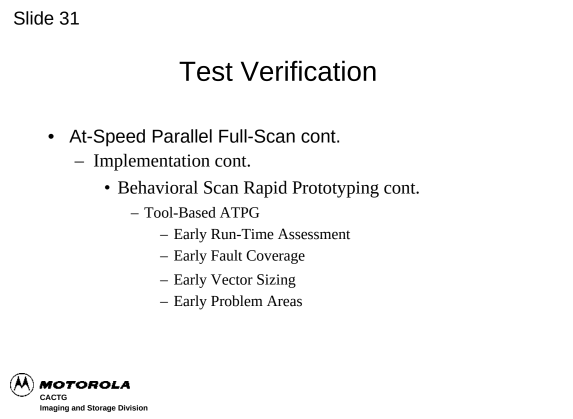
Test Verification

- At-Speed Parallel Full-Scan
  - Goals
    - Safe Shifting
    - At-Speed Operation
    - Coverage-Vector Budgets
  - Implementation
    - Behavioral Scan Rapid Prototyping
      - Early Netlist - optional scan interfaces
      - Tool-Based Scan Rule Checking
        - Early DFT and Design Violations and Problems
        - Clocking, Shifting, Legal Constructs, Contention

MOTOROLA
CACTG
Imaging and Storage Division

The verification of the scan architecture is required to prove that the shift process can operate smoothly and that there is no shift-blocking or driven contention. The verification process must also address whether all the scan cells used in the scan chain are legal scan cells, and that all sequential elements not in the scan chain are supposed to be non-scanned.
Since the scan architecture will be used for AC test as well as DC test, then the frequency of operation of scan chains must be determined. The verification process also generates the data or metrics on fault coverage, speed assessment, vector count, ATPG run-time, and the potential test time. These measurements are required to be available early in the design flow to allow the scan architecture to be changed if necessary.

The scan verification process is done in several stages on ColdFire products. Very early in the design process a netlist is created through rapid DFT analysis mentioned earlier. This netlist does not necessarily have to have the final scan architecture. The first time through the process is usually done by declaring only one scan chain and a dedicated scan input and output just to get an early netlist before any scan definition work is done. As the design matures this process is repeated and, eventually, the rapid prototype scan interface is identical to the final interface with multiple scan chains based on borrowed functional pins.



Slide 31

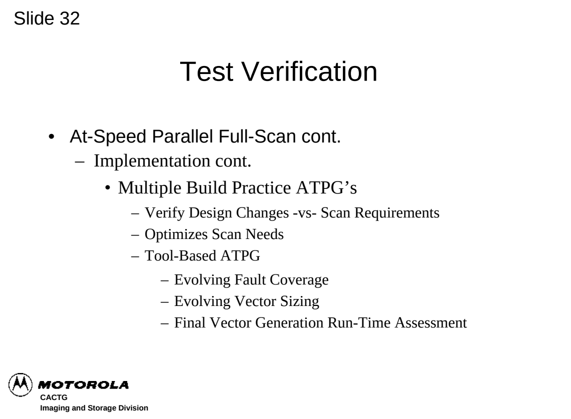
Test Verification

- At-Speed Parallel Full-Scan cont.
  - Implementation cont.
    - Behavioral Scan Rapid Prototyping cont.
      - Tool-Based ATPG
        - Early Run-Time Assessment
        - Early Fault Coverage
        - Early Vector Sizing
        - Early Problem Areas

MOTOROLA
CACTG
Imaging and Storage Division

The early prototype netlist with scan inserted allows the Mentor DFT tools (DFTAdvisor) to be used to check for compliance to scan design rules such as safe shifting, legal cells in the scan chain, no driven contention during scan test operation, no scannable flip-flops not in the scan chain, no inversions in the scan enable control signal, and no blocked shift paths due to disconnections, asynchronous clears or sets, or logic in the shift path. The netlist is also used for early vector generation. The Mentor Fastscan tool is used to verify that the process control scripts are valid and correct; that there are no fault coverage or design problems; that the vector sizing is within tester size and cost goals; and to get a feel for the ATPG tool runtime.

The rapid DFT analysis is valid because the “rapid synthesis” process generally produces more gates than a “timing and area” constrained synthesis. This makes for a pessimistic analysis (more faults than in the mature design).



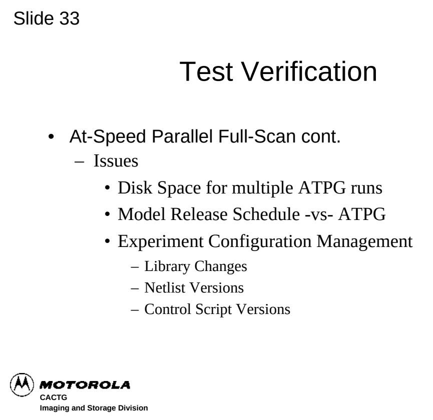
Slide 32

Test Verification

- At-Speed Parallel Full-Scan cont.
  - Implementation cont.
    - Multiple Build Practice ATPG’s
      - Verify Design Changes -vs- Scan Requirements
      - Optimizes Scan Needs
      - Tool-Based ATPG
        - Evolving Fault Coverage
        - Evolving Vector Sizing
        - Final Vector Generation Run-Time Assessment

MOTOROLA
CACTG
Imaging and Storage Division

The design matures as a regularly scheduled model release process occurs. Several of the model releases are synthesized and even passed through the physical mapping process. Some of these netlists are used as “practice” netlists in the same way the rapid prototyping was used early in the design process. This practice is done to ensure that “bug fixes” and “design changes” do not violate the scan rules, and to assess the various sizing (vectors, fault coverage, runtime). Since the design will slowly evolve toward the final form, the measurements approach their final form.
The measurements may be used during the design process to adjust the scan architecture itself. As the design matures, the number of flip-flops to be included in the final design will become apparent. The number of flip-flops and the number of scan vectors may result in the increasing or decreasing of the number of scan chains.


Slide 33

Test Verification

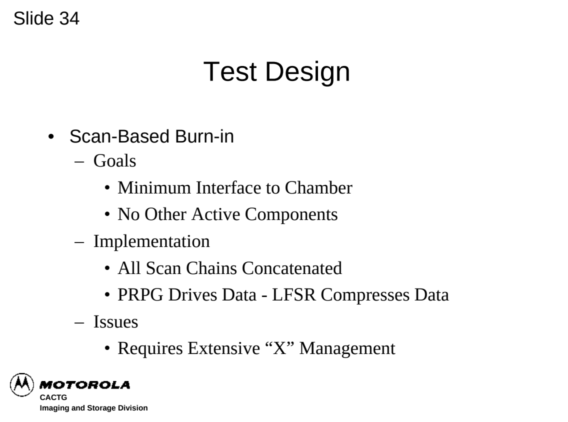
- At-Speed Parallel Full-Scan cont.
  - Issues
    - Disk Space for multiple ATPG runs
    - Model Release Schedule -vs- ATPG
    - Experiment Configuration Management
      - Library Changes
      - Netlist Versions
      - Control Script Versions

MOTOROLA
CACTG
Imaging and Storage Division

There are still several issues with the at-speed scan verification methodology. The biggest problem we encounter on a regular basis is the amount of disk space that is required to store ATPG data for multiple-chips and multiple runs per chip. This is loosely tied to the model release schedule. If the design team releases a new model each week, then there is a potential to have a new verification netlist each week. The whole “rule-check to vectors to fault coverage” process may take more than a week, so “which netlists do we choose to check?”.

The amount of data and disk space may be a problem, but the keeping track of many experimental netlists for multiple chips, netlist versions, library element changes/fixes, vectors (experimental -vs- final), and control or run scripts is a full-time job in itself. We have an internal configuration management environment, and it can track all of the normal “chip build” items, but storing vectors is a whole different problem.

Vectors are “large” files. When new vector file versions replace old vector file versions, the files are generally completely changed, and the previous version is rendered invalid (except in a few cases). Also, the files must be stored in some sort of compressed format. Because of this, we store our vector files apart from our configuration management tool-environment, and have the individual DFT engineers store them with the ZCS utility.

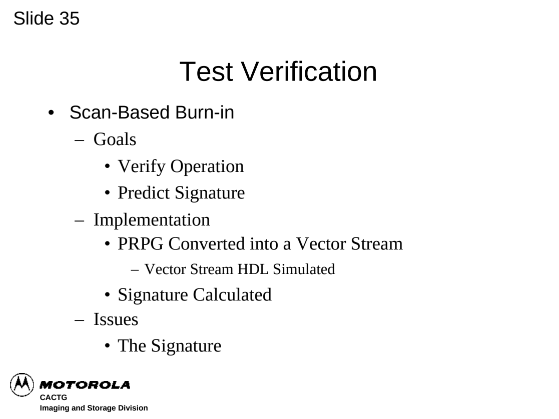

Slide 34

Test Design

- Scan-Based Burn-in
  - Goals
    - Minimum Interface to Chamber
    - No Other Active Components
  - Implementation
    - All Scan Chains Concatenated
    - PRPG Drives Data - LFSR Compresses Data
  - Issues
    - Requires Extensive “X” Management

MOTOROLA
CACTG
Imaging and Storage Division

The same scan architecture can be used for burn-in testing. Since the environmental chamber has a limited electrical interface passing through the door, and since there is a preference to not have active components other than the ColdFire products under test inside the chamber, the scan architecture was designed in such a way to meet these goals.
The minimum number of signals passing through the door requirement was met by externally connecting (outside the package) all of the scan inputs and outputs together into one single scan chain. This scan chain can be driven from an initializer and a Linear Feedback Shift Register (LFSR) used as a Pseudo-Random Pattern Generator (PRPG) located external to the chamber, and the results of the scan testing are compressed by another LFSR used as a signature analyzer. An issue discovered during the implementation of this methodology was the need for “absolute” X-management inside the part. Any non-scan device, or non-controlled memory array that allows non-deterministic values (logic X’s) to be sampled into the scan chains causes test result errors. Each non-deterministic value that can be sampled multiplies the number of possible signatures by two.


Slide 35

Test Verification

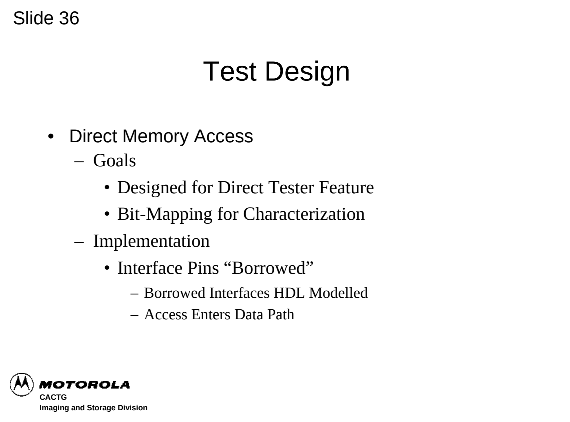
- Scan-Based Burn-in
  - Goals
    - Verify Operation
    - Predict Signature
  - Implementation
    - PRPG Converted into a Vector Stream
      - Vector Stream HDL Simulated
    - Signature Calculated
  - Issues
    - The Signature

MOTOROLA
CACTG
Imaging and Storage Division

The scan-based burn-in verification is not for chip build verification. The verification steps are that the X-management must be checked and that the chip can operate with all of it’s scan chains tied together into one single scan chain. This verification is done by taking the data stream created by the initializer and the PRPG LFSR and applying it to the chip during a simulation run. The simulation is run until functionality is generally verified, since the simulation could take a long time.

The ability to operate all the scan chains when tied together is done by conducting timing analysis on the input and output pins used for the scan interfaces, and looking at the data with respect to the clock-to-out and input setup time specifications needed for the burn-in frequency (which is slower than the operational frequency due to the burn-in equipment limitations).

The other major goal of verification is to calculate the signature. Currently, we have not found a way to successfully “quickly” find the signature other than a long and laborious simulation. There is a shortcut, but it only works for shifting vectors through the part with no sampling being done (scan enable is never de-asserted - so only the flip-flops are checked for errors - however, the part is well exercised even if the combinational logic is not fault-detected). This can be done by applying the PRPG output to the Signature Analyzer directly as a mathematical division operation.



Slide 36

Test Design

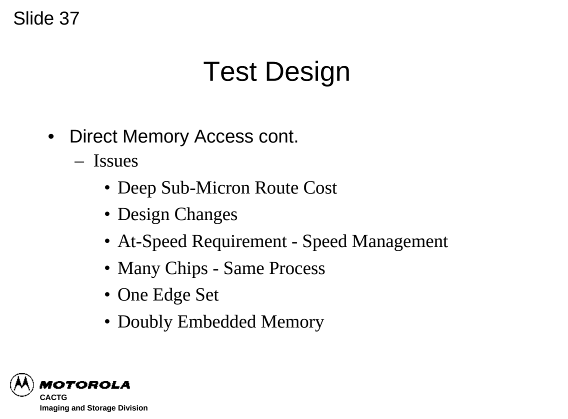
- Direct Memory Access
  - Goals
    - Designed for Direct Tester Feature
    - Bit-Mapping for Characterization
  - Implementation
    - Interface Pins "Borrowed"
      - Borrowed Interfaces HDL Modelled
      - Access Enters Data Path

MOTOROLA
CACTG
Imaging and Storage Division

Some of the ColdFire Standard Products include embedded memory arrays. The in-house established method for production test and characterization, was to use a built-in tester function that generated algorithmic memory tests. This kept memory test vectors from consuming tester vector space/memory, but required implementing some form of direct memory access.

Cost of test is as critical for memory testing as it is for scan. So, similarly to the scan, the interface is borrowed from functional pins, and the edge set is preserved across this test mode as well. The problem with direct memory access, however, is the number of pins that must be borrowed. Generally, there is the Data Bus (32), the Address Bus (24), the Read/Write control signal (1-2), and any extra signals such as "which memory is being tested".

Since direct memory access is used for DC and AC testing, the test architecture must be designed to operate at-speed. The architecture is basically implemented by placing multiplexors in the memory data, address, and control path to take over these paths during a test mode. Since memory access time may be a frequency limiting factor, it is not wise to place the multiplexors in the cone of logic nearest to the memory array. The tester function has the ability to adjust to pipeline sequentiality, so the access to the memory array is designed further up the data path (to the tester extra pipeline stages just make the memory look like a 33-bit or 34-bit word).


Slide 37

Test Design

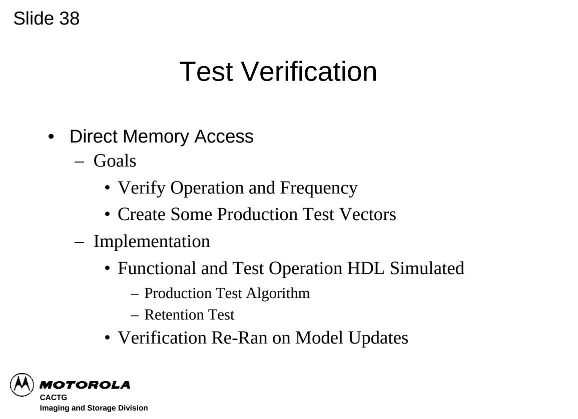
- Direct Memory Access cont.
  - Issues
    - Deep Sub-Micron Route Cost
    - Design Changes
    - At-Speed Requirement - Speed Management
    - Many Chips - Same Process
    - One Edge Set
    - Doubly Embedded Memory

MOTOROLA
CACTG
Imaging and Storage Division

There are many issues with using a direct memory access test architecture. One of the main complaints is that there must be "wire" routes brought from each memory array out to the "many" pins. As designs move into the "deep sub-micron" area from .5 micron and below, routing dominates timing, and layout becomes more route-limited. If there are not enough package pins, then internal logic must be added to allow partial words and partial addresses to be time multiplexed.

One key concern is the instability of the design during development. Since the test architecture is related directly to the memory architecture, every small design change require a test architecture change (we don't have the luxury of waiting until the final version of the memory architecture exists before creating the test architecture). Variations of memory array sizes and types have kept a single "insertable" architecture from being developed.

A good portion of the design process involves making timing. It is difficult to ensure that the edge-set used for functional operation is maintained because of the "wire routing" from the pins to the memory array datapath.
The cost of this memory architecture is high in design time, and chip area (routing in a deep-submicron process), and pins. In the old days, when used on the larger, long development cycle chips, each new chip was with a new process, so characterization was important. However, in the new business with many cores/products made from a single process, this architecture may be overkill.
The biggest issue became evident when the ColdFire architecture was to be embedded. The Core itself was to be embedded within a chip with minimal access of the core pins to the package pins. This resulted in the memory arrays being doubly embedded -- no access to the pins to be used as memory test pins.



Slide 38

Test Verification

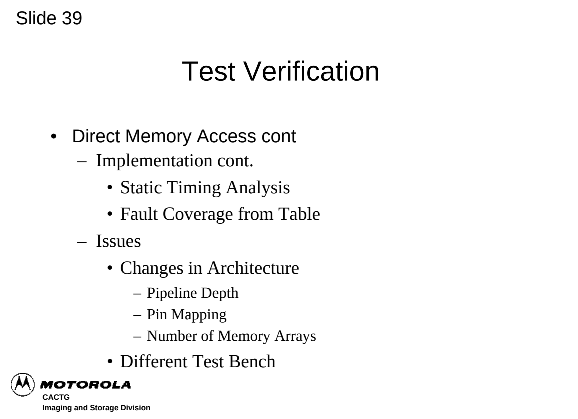
- Direct Memory Access
  - Goals
    - Verify Operation and Frequency
    - Create Some Production Test Vectors
  - Implementation
    - Functional and Test Operation HDL Simulated
      - Production Test Algorithm
      - Retention Test
    - Verification Re-Ran on Model Updates

MOTOROLA
CACTG
Imaging and Storage Division

The verification of a direct memory access architecture is generally straightforward. The goal is to ensure that all the logic related to the memory arrays are operating correctly, and that the memory array itself meets the design goals. The functional verification is done by applying any of several memory test algorithms to a "whole chip" simulation" environment (which exists as part of our overall verification strategy).

The goal is not so much to get "fault coverage" as it is to provide some simulation and operational test "code" to launch the test sequence on the tester, and to provide specific test operations such as "retention testing".

Once the operational vectors have been created, the code is run on each model update (model release).



Slide 39

Test Verification

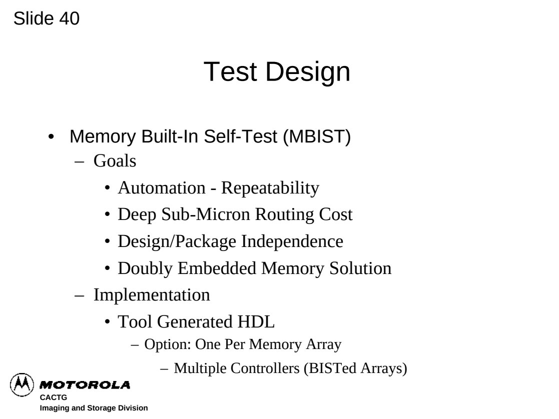
- Direct Memory Access cont
  - Implementation cont.
    - Static Timing Analysis
    - Fault Coverage from Table
  - Issues
    - Changes in Architecture
      - Pipeline Depth
      - Pin Mapping
      - Number of Memory Arrays
    - Different Test Bench

MOTOROLA
CACTG
Imaging and Storage Division

The "fault coverage" of the memory array is assessed by referring to a table of coverage by the algorithm applied (e.g., a March C+ algorithm has a specific set of defects that are targeted, and a related coverage of those defects). We do not actually expect a fault simulator to provide the actual coverage based on vector application.

To ensure timing goals are met, the pathways and control logic for the memory are timing analyzed, and the constraints are hand written to reflect the "cycle" or "timing" differences between a memory array and general logic (e.g., memory access time may not be single cycle).

The issues that exist with memory test verification have to do with the constant changes done to the model for each model release. This requires that the test bench or the applied stimulus must be changed on a regular basis. Changes that affect the ability to apply tests are items like "pipeline depth" changes, modifications to the "pin mapping" (which external pins are used for the test interface), and the number of memory arrays, or the segmentation of memory arrays.



Slide 40

Test Design

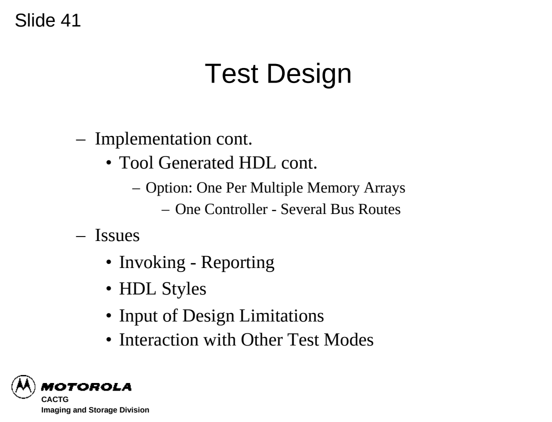
- Memory Built-In Self-Test (MBIST)
  - Goals
    - Automation - Repeatability
    - Deep Sub-Micron Routing Cost
    - Design/Package Independence
    - Doubly Embedded Memory Solution
  - Implementation
    - Tool Generated HDL
      - Option: One Per Memory Array
        - Multiple Controllers (BISTed Arrays)

MOTOROLA
CACTG
Imaging and Storage Division

The main goal of the Memory Built-in Self-Test (MBIST) was to solve the shortcomings identified with the direct memory access architecture. This methodology change was done to eliminate the “design intensive” methodology; to minimize the routing cost associated with bringing the internal data, address, and control busses out to the pins; to minimize the required large pin interface; to allow test to be supported no matter how many package pins are supported; and to allow testing of “doubly embedded” memory arrays.

Meeting these goals was done by adopting a tool generated Memory BIST methodology, the Mentor MBISTArchitect tool. This involves allowing a software tool to create an HDL model that could go through the synthesis process.

The MBIST tool has two basic ways of implementing the test architecture. One method is to support one controller per memory array -- which results in many controllers associated one with each memory array -- this is known as having BISTed arrays. No matter what is done with a memory array, the test circuitry is associated with it and moves with it as a single entity. This method requires that several logic blocks exist, but minimizes the number of busses that have to traverse the chip.


Slide 41

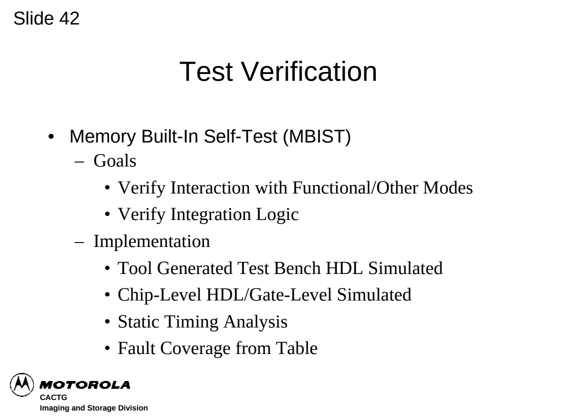
Test Design

- Implementation cont.
  - Tool Generated HDL cont.
    - Option: One Per Multiple Memory Arrays
      - One Controller - Several Bus Routes
- Issues
  - Invoking - Reporting
  - HDL Styles
  - Input of Design Limitations
  - Interaction with Other Test Modes

MOTOROLA CACTG Imaging and Storage Division

The other method of implementing the architecture has to do with making one single chip-level controller, and letting this operate and test many memories. This method has a similar cost to the direct memory access method in that a single design entity will be the source of many busses and control signals. However, the large pin interface will not be necessary.
The issues with either MBIST method have to do with what is not provided. There is not a “tool” to decide which method to use. It requires an extensive tradeoff analysis per chip design that involves: how many memory arrays; relative sizing of each array; the size of the busses used; and the location of the memory arrays on the final chip.

There is no acceptable reporting logic. Each chip has different needs such as “fail flags per each memory”, “a fail flag for all/any memory”, or “a bitmap output”. Currently, there is no menu or selection of these.
One other big issue has to do with HDL styles. The tool generated HDL implies asynchronous elements (which we do not allow), and, for our case, was not “non-blocking”. We had to hand edit the output code to fix these “style” differences. We recommend that some form of design limitations be an input to the tool to allow some of these cases to be handled.
In the future we also need to address the interaction of the MBIST with other test modes such as IDDQ, scan, and JTAG modes -- what is the interaction, are there one-cycle mode changes, are mode changes destructive, and can other modes be run simultaneously.



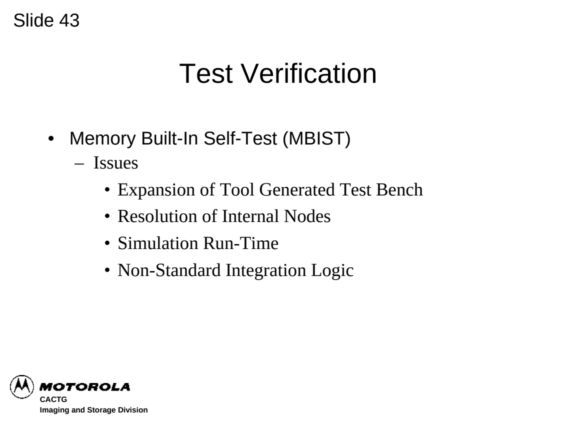
Slide 42

Test Verification

- Memory Built-In Self-Test (MBIST)
  - Goals
    - Verify Interaction with Functional/Other Modes
    - Verify Integration Logic
  - Implementation
    - Tool Generated Test Bench HDL Simulated
    - Chip-Level HDL/Gate-Level Simulated
    - Static Timing Analysis
    - Fault Coverage from Table

MOTOROLA
CACTG
Imaging and Storage Division

The verification of MBIST has the same goals as the direct memory access architecture, to provide some vector components for the tester, and to ensure operation and frequency. As mentioned earlier, the interaction with scan and JTAG modes must be defined and verified. In addition, there is also the “extra” logic that was added to place the part into MBIST mode, or to report the result of the MBIST test. Some of this logic is not provided by the tool and yet the entire test design must be shown to operate correctly.
The Mentor MBISTArchitect tool creates the logic, and also creates a local testbench. This is useful for verifying the tool output, and can be used to get any signature (if used). However, the verification requires a whole chip operation and interaction. The MBIST logic must also be static timing analyzed since it must be shown to operate at the rated frequency. In our methodology, we decided to allow the MBIST controller to be scanned so that the logic could be “fault tested” by the scan mode. The memory fault coverage comes from a table similarly to the direct memory access method (the algorithm is embedded in the chip instead of coming from a tester, though).



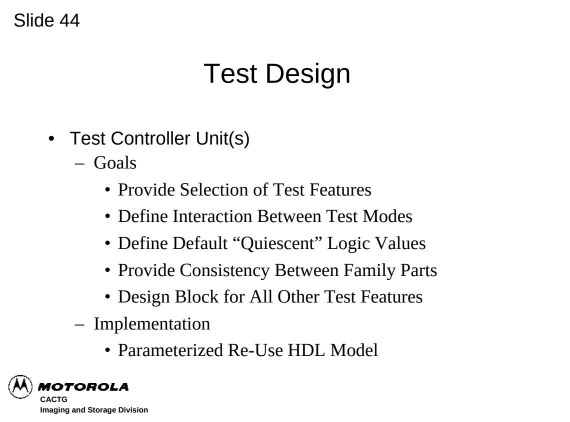
Slide 43

Test Verification

- Memory Built-In Self-Test (MBIST)
  - Issues
    - Expansion of Tool Generated Test Bench
    - Resolution of Internal Nodes
    - Simulation Run-Time
    - Non-Standard Integration Logic

MOTOROLA
CACTG
Imaging and Storage Division

There are many issues with the change from direct memory access method to the MBIST method. However, we must say that this change is still fairly new and we expect it to mature in time.
The issues are that we must expand the capability of the tool generated testbench to be useful at the chip-level. When we do the verification on the gate-level netlist, we must “find” the internal nodes in a synthesis netlist that we wish to observe (model names are not always preserved).
The simulation for verification of the MBIST, and all associated memory logic, is the application of the entire memory test. If multiple memory arrays are being tested, and the controllers were designed for sequential operation (one after the other), then run-time can be an issue.
The MBIST has turned out to be as “design specific” as the direct memory architecture in that we have to “hand craft” the integration logic for invoke and report.



Slide 44

Test Design

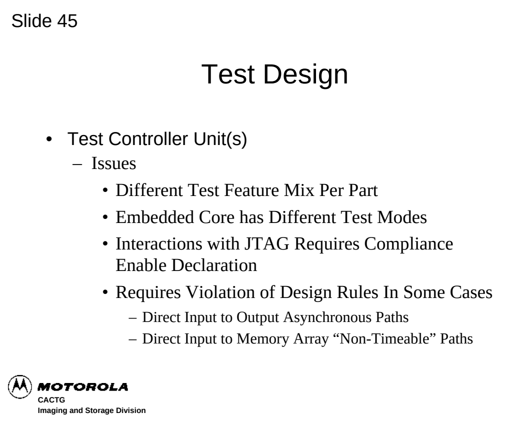
- Test Controller Unit(s)
  - Goals
    - Provide Selection of Test Features
    - Define Interaction Between Test Modes
    - Define Default “Quiescent” Logic Values
    - Provide Consistency Between Family Parts
    - Design Block for All Other Test Features
  - Implementation
    - Parameterized Re-Use HDL Model

MOTOROLA
CACTG
Imaging and Storage Division

There is a wealth of test features and capabilities on each ColdFire product (Embedded Core or Standard Product). Organizing, prioritizing, and defining the interaction of these test features and modes requires a dedicated test controller. When test features are being used, other test features must remain quiescent -- when no test features are being used, all test features must remain quiescent.

Having the on-chip Test Controller design unit also allows for consistency between family parts. The scan mode or the memory test mode is always invoked in a similar fashion. The Test Controller is also a design block for other features that may not be test modes per se. For example, there is a single pin dedicated to tristating all outputs for IDDQ and tristate leakage tests. This feature operates in all functional and test modes except for JTAG modes (this is a compliance enable issue).
The Test Controllers are re-use HDL models with parameters included. A preprocessor (MPP) configures the models for each different part based

on the mix of test features that are to be included.


Slide 45

Test Design

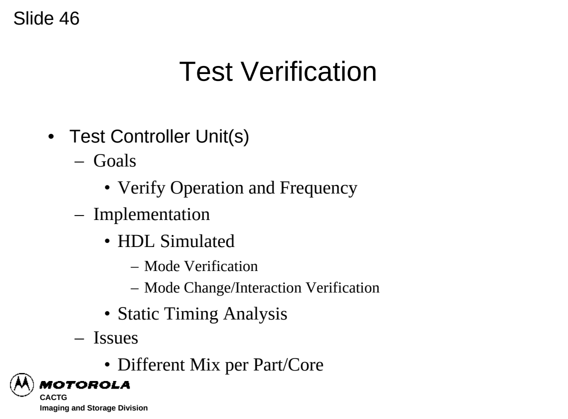
- Test Controller Unit(s)
  - Issues
    - Different Test Feature Mix Per Part
    - Embedded Core has Different Test Modes
    - Interactions with JTAG Requires Compliance Enable Declaration
    - Requires Violation of Design Rules In Some Cases
      - Direct Input to Output Asynchronous Paths
      - Direct Input to Memory Array "Non-Timeable" Paths

MOTOROLA
CACTG
Imaging and Storage Division

Even though the Test Controller seems to be a simple concept, there are many issues with it's implementation. Each standard product and embedded core does have a different mixture of features. For example, a Core does not have JTAG whereas a standard product does. Another example is that a first part in a new process may require the direct memory access to allow easy characterization, where the next product may use the MBIST methodology.

One of the other big issues is the interaction of test modes with JTAG. Any test mode that can override what JTAG is doing requires declaration as a compliance enable in the BSDL, and requires some architecture changes or support. All of these interactions cannot be seen beforehand, so some hand edits are required to adjust to each condition.

The biggest issue (especially with the design community) is that we, the DFT group, may create logic that does not meet the design rules applied to the design community. For example, the JTAG logic requires an asynchronous reset if the trstB signal is supported. Another example is the "multi-cycle" or "non-timeable" paths into the memory arrays from input pin for the direct memory access test. The signal that gives us the most problem, however, is the "tristate all" signal that come directly from an input pin and affects every output pin tristate driver. This signal is fully asynchronous and does not ever meet a register.



Slide 46

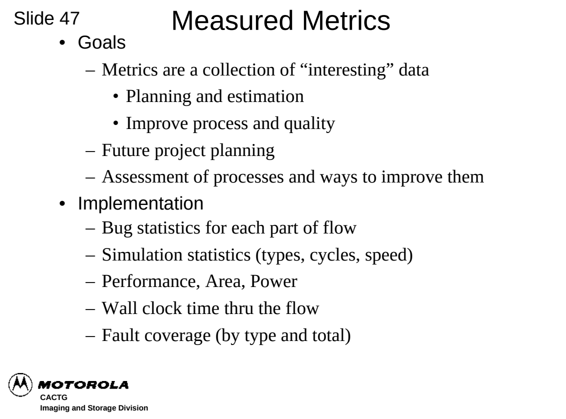
Test Verification

- Test Controller Unit(s)
  - Goals
    - Verify Operation and Frequency
  - Implementation
    - HDL Simulated
      - Mode Verification
      - Mode Change/Interaction Verification
    - Static Timing Analysis
  - Issues
    - Different Mix per Part/Core

MOTOROLA
CACTG
Imaging and Storage Division

Since the Test Controller blocks are design blocks, they must also meet the requirements of all other design blocks. They must be verified for operation and frequency. To this end, we simulate the HDL early in the design cycle. The Test Controllers must also be timing analyzed since several test modes and signals are critical (such as the at-speed scan control).

The Test Controller units different for each chip/core, so we must craft the verification for each one.



Slide 47

Measured Metrics

- Goals
  - Metrics are a collection of “interesting” data
    - Planning and estimation
    - Improve process and quality
  - Future project planning
  - Assessment of processes and ways to improve them
- Implementation
  - Bug statistics for each part of flow
  - Simulation statistics (types, cycles, speed)
  - Performance, Area, Power
  - Wall clock time thru the flow
  - Fault coverage (by type and total)

MOTOROLA
CACTG
Imaging and Storage Division

Metrics are essential in assessing current processes and coming up with ways to improve them. They are used for future project planning as well. We tracked bug statistics for each section of the chip, for each design discipline, and categorized them for future reference. We can assess our verification coverage by keeping track of simulation statistics. We can assess whether our design is progressing toward our design goals by measuring the area and timing results for each pass through the methodology flow. We can assess fault coverage before we finish the design, guaranteeing that we can get our product into volume manufacturing much faster. Measuring the wall clock time through the methodology flow enables us to continuously make refinements.

Slide 48

Summary

- Model release methodology
- Rapid-prototyping methodology
  - Decreases design cycle time
  - Target embedded core or standard product
- Functional verification methodology
- Logic verification methodology
- Design/methodology reuse across multiple products
- DFT methodology
  - High quality
  - Target embedded core or standard product
  - Test access for embedded core

MOTOROLA
CACTG
Imaging and Storage Division

Slide 49

Recommended Resources

- Motorola's ColdFire Embedded Core Products
  - Technical Resource Center 1-800-521-6274 (U.S only)
  - http://www.motorola.com/ColdFire
- HP Class C180 workstations
- Synopsys for synthesis
- Aquarius for place/route
- Mentor
  - Design Architect for custom cache design
  - Fastscan ATPG tool
  - MBIST Architech (Memory BIST Tool)
  - DFT Insight, DFT Advisor
  - Checkmate for physical verification

MOTOROLA
CACTG
Imaging and Storage Division

The HP Class C180 workstations are used primarily for CPU intensive synthesis and timing jobs, place and route jobs, and physical verification. During idle cycles (rarely the case), these machines are used for model regression testing.

There is quite an array of Mentor tools used in the design, test, and verification of ColdFire products, the newest member being the MBIST tool.

Slide 50

Recommended Resources

- Resources required per ColdFire product

– Job Function	Disk	Mem	CPU
– Model (Verilog)	20M	-	-
– Functional Verification	200M	32M	100's
– Synthesis/timing	200M	128M	1-3 (fast)
– Logic Verification	100M	300M	1-20
– ATPG	1G	128M	1

MOTOROLA
CACTG
Imaging and Storage Division

The disk and memory resources are shown for typical ColdFire products.

Slide 51

Acknowledgments

- Customer demand created need for the ColdFire family

MOTOROLA
CACTG
Imaging and Storage Division