Digital IC Design Methodology

Felipe Prado Yonehara
Digital IC Designer
Decisions, Decisions

- Take a CEO, a manager, and an individual contributor for a simple company.
  - The CEO’s “executive” decisions can have a great affect on the company, its partners, its products, and its customers.
  - The manager’s decision can have an affect on the managed team, and possibly other teams.
  - The individual contributor (IC) may have an affect on a specific project or task.
Decisions, Decisions  (continued)

- Just like the company organization, decisions made for IC design also have very different effects:
  - Decisions made at the specification level
  - Decisions made at the microarchitectural level
  - Decisions made at the RTL level
Objectives (of this module)

- State the considerations in moving from specification to microarchitecture to RTL
- State the considerations for selecting a process technology
Topics for Digital IC Design Methodology

• Day 1
  – Specification
  – Microarchitecture
  – RTL coding
  – CMOS process selection

• Day 2
  – Implementation flow overview
  – Synthesis
  – Static timing analysis
  – Test
  – Floorplanning
  – Timing closure
Topics in This Module

- Specification
- Microarchitecture
- RTL coding
- CMOS process selection
Basic Flow: Implementation and Verification

1. **Specification**
   - System Simulation
   - Designer

2. **Micro-Architecture**
   - System Simulation
   - Designer

3. **RTL**
   - Logic simulation
   - Logic Synthesis

4. **Synthesizable Gates**
   - Gate Level Simulation
   - Place/Route
   - Gates
   - Placed/Route Gates
   - Gates
   - Place/Route
   - GDSII

5. **Implementation**
   - Verification
   - Implementation
What Is a Specification?

- Ideas begin with a specification, which can be a textual, graphical, or even a software representation.

- **Definition**: A specification is an explicit set of requirements to be satisfied by a material, product, or service.

- Example: The specification for the latest chip called for 250 MHz core clock with a 6G SERDES interface, able to process 1M streams of data per second at less than 10W total power.
Discussion

• Who creates the specification?
• What information is required to come up with the details of the specification?
# Specification Contents and Example

<table>
<thead>
<tr>
<th>Title</th>
<th>Specification for XYZ Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Reviewers</td>
<td>HW, SW, Test, Manufacturing, Customer, etc.</td>
</tr>
<tr>
<td>Modification History</td>
<td>1.0 - 12/2007 – Initial Revision, 2.0 – 1/2008 – Changes to ...</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>Section 1 Overview - Section 2 Block 1 ...</td>
</tr>
<tr>
<td>Glossary</td>
<td>XYZ = Codename for project, etc.</td>
</tr>
<tr>
<td>Overview</td>
<td>The XYZ chip is a new product targeted for consumers...</td>
</tr>
<tr>
<td>Performance Targets</td>
<td>125 MHz, 1W Total Power, 1M streams/s</td>
</tr>
<tr>
<td>Block Diagrams</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>Graphs</td>
<td></td>
</tr>
<tr>
<td>Tables</td>
<td>Pin D</td>
</tr>
<tr>
<td>Detailed Description</td>
<td>Block A is input block, it receives signals ...</td>
</tr>
</tbody>
</table>
For chip design, the specification is the reference model the team uses to:

- Design the overall chip
- Specify the intellectual property used
- Specify the “new logic” to be created
- Specify the block-level and chip-level interfaces
- Partition the chip into functional blocks
- Communicate interfaces and requirements with other teams
- Measure actual performance versus specified targets
Specification Snippet

- Section 2.1 Block A Interface
- Block A is the input which receives serial data from the I/O and transfers to Block B.

<table>
<thead>
<tr>
<th>PORT NAME</th>
<th>DIRECTION</th>
<th>SOURCES/DESTINATION</th>
<th>SIZE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>INPUT</td>
<td>I/O</td>
<td>1 bit</td>
<td>Clock at 125 MHz</td>
</tr>
<tr>
<td>Din</td>
<td>Input</td>
<td>I/O</td>
<td>32 bits</td>
<td>Input data</td>
</tr>
<tr>
<td>Dout</td>
<td>Output</td>
<td>A and B</td>
<td>32 bits</td>
<td>Output data</td>
</tr>
</tbody>
</table>
High-Level Decisions

In creating or modifying the specification, high-level decisions that effect the system and its environment are made.

• For example, the choice between external SRAM or DRAM:
  • Control for each one is drastically different
  • I/Os for the chip will be affected
  • Board itself will be affected, components plus signal routing

• Another example is the choice to run the design at 125 MHz or 250 MHz
  • Chip-level clock input has to change
  • Software might have to change because the performance could be 2X different
Discussion

- What other decisions would be considered specification level?
- How would we validate these decisions?
Topics in this Module

• Specification
✓ Microarchitecture
• RTL coding
• CMOS process selection
What Is Microarchitecture?

- Step between the specification and RTL, the microarchitecture defines how the block will be implemented.

- **Definition:** The microarchitecture implements the specification and defines specific mechanisms and structures for achieving that implementation.

- **Example:** For Block A, the designer created a microarchitecture and partitioned his block into several smaller modules.
Discussion

• Who creates the microarchitecture?
• What information is required to come up with the details of the microarchitecture?
Microarchitecture

- The microarchitecture is typically based on a block in the specification.
For chip design, the micro-architecture is the reference model the designers uses to

• Design the block
• Specify the intellectual property used
• Specify the “new logic” to be created
• Specify the block-level interfaces
• Partition the block into more functional blocks
• Communicate interfaces and requirements with other block designers
• Measure actual performance versus specified targets
Microarchitecture Snippet

• This is the input block for Block A. Its function is to process and slice the data into 16-bit segments, based on control signals from the FSM, and send it to block B2.

<table>
<thead>
<tr>
<th>PORT NAME</th>
<th>DIRECTION</th>
<th>SOURCE/DESTINATION</th>
<th>SIZE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>input</td>
<td>I/O</td>
<td>1 bit</td>
<td>Clock at 125MHz</td>
</tr>
<tr>
<td>bus1</td>
<td>input</td>
<td>I/O and Sync FIFO</td>
<td>32 bits</td>
<td>Input data</td>
</tr>
<tr>
<td>bus1a</td>
<td>output</td>
<td>Slicer</td>
<td>16 bits</td>
<td>Output data</td>
</tr>
</tbody>
</table>
Mid-Level Decisions

In creating or modifying the microarchitecture, mid-level decisions that affect the block itself are made.

• For example, the choice between internal SRAM or register array
  • Interface to the outside environment is the same.
  • Performance of the block may vary slightly, but functionality is the same.
• The choice to use multiple datapath versus a single datapath
  • Area is tradeoff versus performance.
  • As long as the performance targets are met, how the design is actually implemented is a microarchitectural decision.
Discussion

• What other decisions would be considered microarchitecture level?
• How would we validate these decisions?
Topics in this Module

• Specification
• Microarchitecture
  ✓ RTL coding
• CMOS process selection
Basic Flow: Implementation and Verification

- Specification
  - System Simulation
  - Designer
  - Micro-Architecture
- RTL
- Synthesizable Gates
  - Logic Synthesis
  - Gates
- Gate Level Simulation
- Place/Route
- Placed/Route Gates
  - Physical verification
  - Gates
  - Place/Route
  - GDSII
- Formal Verification
- Timing Signoff
- Verification
What Is RTL?

RTL (register transfer level)

- **Definition:** A way of describing the operation of digital circuit where the behavior is defined in terms of the flow of signals between registers and the operations performed

- **Example:** The translation of a system specification to RTL is a difficult and time-consuming task.
Discussion

- Who creates the RTL?
- What information is required to come up with the details of the RTL?
The RTL is typically based on a block in the microarchitecture.
For chip design, the RTL is the reference model the designer uses to

- Design the block for final implementation
- Instantiate and connect intellectual property
- Code the “new logic”
- Create the block-level interfaces
- Partition the block into sub-blocks
- Verify the interfaces to other blocks
- Run simulations to measure actual performance versus specified targets
RTL Snippet

// File: block_a1.v
module block_a1 (...);
input clk;
input [31:0] bus1;
output [15:0] bus1a;
sync_fifo u1 (...);
slicer u2 (...);
fsm u3 (...);
endmodule

// File: sync_fifo.v
module sync_fifo (clk, bus1, ta, tb);
...
endmodule

// File: slicer.v
module slicer (clk, tb, tc, bus1a);
...
endmodule

// File: fsm.v
module fsm.v (clk, ta, tc);
...
endmodule
Low-Level Decisions

• In creating or modifying the RTL, low level decisions that effect the implementation of the block itself are made.
  – For example, the choice to use a particular coding style
    • Designer has previous knowledge of an optimal style for implementation or verification
    • Designer is more comfortable with a particular style
  – The choice to add pipeline stages versus forcing more logic into a single cycle
    • Cycle time is traded off for sequential area
    • As long as the performance targets are met, how the design is actually implemented is a microarchitectural decision
    • It is possible the latency of the top-level block is “flexible”
Discussion

• What other decisions would be considered RTL level?
• How would we validate these decisions?
## Specification to Microarchitecture to RTL

- **Who creates the ___?**
- **What information is required to come up with the details of the ___?**
- **What other decisions would be considered ___?**
- **How would we validate these decisions?**

<table>
<thead>
<tr>
<th></th>
<th>Specification</th>
<th>Microarchitecture</th>
<th>RTL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Who?</strong></td>
<td>CEO, CTO, Marketing,</td>
<td>Chip Lead, Block-Level</td>
<td>Block-Level Designer</td>
</tr>
<tr>
<td></td>
<td>Chip Lead, etc.</td>
<td>Designer</td>
<td></td>
</tr>
<tr>
<td><strong>What Information?</strong></td>
<td>Customers, Market Data,</td>
<td>Performance Targets,</td>
<td>Performance Targets,</td>
</tr>
<tr>
<td></td>
<td>Competitive Data, etc.</td>
<td>Block I/O, etc.</td>
<td>Block I/O, etc.</td>
</tr>
<tr>
<td><strong>Other decisions?</strong></td>
<td>Overall Architecture,</td>
<td>Block Partitioning, Memory</td>
<td>Instantiate vs. Infer,</td>
</tr>
<tr>
<td></td>
<td>Use of specific IP</td>
<td>size, and Quantity</td>
<td>Re-use code vs. Create</td>
</tr>
<tr>
<td><strong>Validate?</strong></td>
<td>System Simulation</td>
<td>System Simulation</td>
<td>RTL Simulation</td>
</tr>
</tbody>
</table>
Topics in this Module

• Specification
• Microarchitecture
• RTL coding
✓ CMOS process selection
Definitions

• ASIC foundry
  – **Definition:** Manufacturer of the process technology to enable the creation of integrated circuits
  – **Example:** IBM and TSMC are two of the largest ASIC foundries

• Technology node
  – **Definition:** The geometry or level of the process technology
  – **Example:** The technology node for integrated circuits have evolved from 10 μm in the early 1970s to 45 nm today

• CMOS process selection
  – **Definition:** The process of selecting a particular foundry and technology node to implement an integrated circuit
  – **Example:** The current chip is targeted for an IBM 90-nm process
ASIC Foundries

• Among others:
  – IBM
  – Freescale
  – Texas Instruments
  – Agere
  – Fujitsu
  – NEC
  – Toshiba
  – Hitachi
  – ST Microelectronics
  – TSMC
  – UMC
  – Chartered
Technology Node
What Are the Popular Choices Today?

- 180 nm: Very mature technology
- 130 nm: Mature technology
- 90 nm: Mainstream
- 65 nm: Advanced, but many are migrating from 90 nm
- 45 nm: Cutting edge
Discussion

• As a design team, what would be considered when choosing an ASIC foundry?
• What would be considered when choosing a technology node?
ASIC Foundry and Process Technology

• Considerations:
  – NRE
  – Mask costs
  – Yield
  – Die size
  – IP
  – Packaging
  – Tool complexity
• NRE (non-recurring engineering cost)
  – Up-front payment to an ASIC vendor to design a chip
  – The more advanced the technology process, the higher the NRE
Mask Cost

- Part of the NRE is the mask cost, or the price of creating the custom mask sets to produce the chip. The more advanced the technology process, the higher the NRE.
- The more advanced the technology process, the higher the mask cost.
- When performing a late-stage ECO (engineering change order), the cost of a new set of masks are charged to the customer.

<table>
<thead>
<tr>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRE</td>
</tr>
<tr>
<td>Mask Cost</td>
</tr>
<tr>
<td>Yield</td>
</tr>
<tr>
<td>Die Size</td>
</tr>
<tr>
<td>IP</td>
</tr>
<tr>
<td>Packaging</td>
</tr>
<tr>
<td>Tool Complexity</td>
</tr>
</tbody>
</table>
Yield

- Yield is the percentage of working die per wafer.
- The more advanced the technology process and the larger the die, the lower the yield.
- The more mature the technology process and the smaller the die, the higher the yield.
- Yield affects the overall costs of producing the chip. With a higher yield, the cost is lower.
Die Size

- For a similar design in different process technologies, the more advanced the technology, the smaller the die size.
- Smaller die size reduces cost, since more die can be produced per wafer.
- Smaller die size also makes it possible to integrate more functionality into the same space.

<table>
<thead>
<tr>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRE</td>
</tr>
<tr>
<td>Mask Cost</td>
</tr>
<tr>
<td>Yield</td>
</tr>
<tr>
<td>Die Size</td>
</tr>
<tr>
<td>IP</td>
</tr>
<tr>
<td>Packaging</td>
</tr>
<tr>
<td>Tool Complexity</td>
</tr>
</tbody>
</table>

IP (Intelectual Property)

- IP choices are dependent on the process technology.
- Some IP have been validated and tested and are available in some process technologies and not others.
- IP choice and availability is often one of the major factors in deciding a project’s technology process.

<table>
<thead>
<tr>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRE</td>
</tr>
<tr>
<td>Mask Cost</td>
</tr>
<tr>
<td>Yield</td>
</tr>
<tr>
<td>Die Size</td>
</tr>
<tr>
<td>IP</td>
</tr>
<tr>
<td>Packaging</td>
</tr>
<tr>
<td>Tool Complexity</td>
</tr>
</tbody>
</table>
Packaging

- Because the process technology choice affects power and die size, it also affects the packaging selection.
- High power and large chips require ceramic or custom packages.
- Lower power and smaller chips can use cheaper, plastic packages.

<table>
<thead>
<tr>
<th>Device</th>
<th>NRE</th>
<th>Mask Cost</th>
<th>Yield</th>
<th>Die Size</th>
<th>IP</th>
<th>Packaging</th>
<th>Tool Complexity</th>
</tr>
</thead>
</table>

Tool Complexity

• The more advanced the process technology, the more complex the tool methodology.

• Very cutting-edge process technologies typically have much more complex design and signoff rules.

• Since more advanced processes scan mean much larger designs,

• the tool’s capacity limits are often tested.

<table>
<thead>
<tr>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRE</td>
</tr>
<tr>
<td>Mask Cost</td>
</tr>
<tr>
<td>Yield</td>
</tr>
<tr>
<td>Die Size</td>
</tr>
<tr>
<td>IP</td>
</tr>
<tr>
<td>Packaging</td>
</tr>
</tbody>
</table>

Tool Complexity
Summary

• Specification
  – The reference model that defines the overall scope of the design and its performance goals
  – The specification is created by the product specialists or sometimes the leaders of the company

• Microarchitecture
  – The model that defines the implementation plan of the blocks of the specification
  – The microarchitecture is created by the chip leads or block-level designers
Summary (continued)

• RTL
  – The model that defines the actual implementation of the blocks of the microarchitecture and, hence, the specification
  – The RTL is created by the block-level designers

• CMOS Process selection
  – With a variety of ASIC foundries and technology nodes, there are many choices for design teams
  – Many considerations must be taken into account before deciding on a foundry and node
Digital IC Methodology

Part 2
Basic Flow

• At the highest level, what tasks need to be performed when creating a chip from an idea?
  – Implementation
    • Transform the design from idea or specification to various representations of logical and physical hardware
  – Verification
    • Ensure the functionality, timing, and integrity of the changing design through the process
Basic Flow (continued)

• What about RTL?
  – Verification
    • Covered in the “Design Specification” module
  – Implementation
    • From system to microarchitecture to RTL
  – Discussion questions
    • How do we determine the “quality” of the RTL going forward?
    • How do we know the RTL code meets our specifications?
    • When do we change the RTL?

Implement (Design Methodology part 1)

Verify

RTL

Implement (This Module)

Verify
Module Objectives

• You will be able to
  – Articulate the overall design methodology in more detail
  – State the considerations for the quality of the RTL implementation
Topics for Digital IC Design Methodology

• Day 1
  – Specification
  – Microarchitecture
  – RTL coding
  – CMOS process selection

• Day 2
  – Implementation flow overview
  – Synthesis
  – Static timing analysis
  – Test
  – Floorplanning
  – Timing closure
Topics in This Module

✓ Implementation flow overview
• Synthesis
• Static timing analysis
• Test
• Floorplanning and placement
• Timing closure
Basic Flow: Implementation

1. Specification
2. Designer
3. Micro-Architecture
4. Designer
5. RTL
6. Logic Synthesis
7. Gates: Synthesizable Gates
8. Place/Route
9. Gates: Placed/Route Gates
10. Place/Route
11. GDSII
Implementation Flow Overview

- Logic Synthesis
- Gates
- Floorplanning
- Placement
- Clock Tree
- Route
- GDSII
- RTL
- Timing Closure
- Static Timing Analysis
- Test

Place and Route
Topics in This Module

• Implementation flow overview
  ✓ Synthesis
• Static timing analysis
• Test
• Floorplanning and placement
• Timing closure
What Is Logic Synthesis?

• **Definition:** The process of parsing, translating, and optimizing RTL code into

• **Example:** To determine the feasibility of the design, we need to synthesize the RTL code into gates and measure timing, power, and area. Discrete logic gates.
Implementation Flow Overview

- Logic Synthesis
- Gates
- Floorplanning
- Placement
- Clock Tree
- Route
- GDSII
- RTL
- Timing Closure
- Static Timing Analysis
- Test
Discussion Questions

• What are the inputs and outputs of logic synthesis?
• How does this help us determine the “quality” of the RTL?
How Do We Measure the Quality of the RTL?

- **By synthesizing to our target technology...**
  - **Timing**
    - We can measure the number of logic levels in the design.
    - We can “time” the design against its requirements to see if it meets timing.
  - **Area**
    - We can measure the number of gate units used in the design.
    - We can compare the size of the design against its requirements to see if it meets its area goals.
  - **Power**
    - We can measure the dynamic and leakage power used in the design
    - We can compare the power numbers against
**Inputs and Outputs**

- Fundamentally, synthesis produces a gate-level netlist from RTL code.
  - **RTL**
    
    ```
    always @ (a or b) begin
    z = a && b
    end
    ```
  - **Netlist**
    
    ```
    AND2x1 u0 (.A(a), .B(b),
    .Z(z));
    ```
The technology library determines the specific process technology for which the design is targeted.

For example, if we changed the library, we could have netlists targeted for various technology with the same RTL.

- 90 nm IBM
- 65 nm IBM
- 90 nm TSMC
- 65 nm TSMC
• The constraints are the goals that the synthesis tool tries to meet.

• For example, we may have a design that is targeted for several different applications with different requirements.
  – Timing is most important; area is not a concern.
  – Power and area are more important; and timing is less so.
  – Timing, area, and power are all important, and we have specific measures for each.
• The reports from the synthesis tool let us know if we met all of our constraints.
  – Timing reports
  – Area reports
  – Power reports
When Do We Change the RTL?

• Based on the constraints and the synthesis reports, we will know...
  – The timing of the design, and which specific paths violate
  – The overall area of the design
  – The overall power of the design
• If we violate timing, area, or power by a large margin, we may have to modify the RTL accordingly
Topics in This Module

• Implementation flow overview
• Synthesis
  ✓ Static timing analysis
• Test
• Floorplanning and placement
• Timing closure
What Is Static Timing Analysis?

• Static timing analysis (STA) is the preferred method for timing signoff.
  – **Definition**: A method of computing the expected timing of a digital circuit without using Spice (transistor level) simulation.
  – **Example**: To determine the timing of the design, we ran static timing analysis after synthesis, and saw several paths violating their setup time requirements.
Implementation Flow Overview
• **SPICE or circuit simulation**
  – Very accurate
  – Very small capacity
  – Very slow

• **Dynamic simulation**
  – Accurate
  – Requires test vectors
  – Larger capacity than SPICE
  – Slow compared to static

• **Static**
  – Accurate
  – No test vectors
  – Highest capacity
  – Very fast
Inputs and Outputs

• Like synthesis, the STA tool requires a technology library and constraints.
• It also takes in the gate-level netlist from logic synthesis.
• STA has a delay calculator that computes the timing for every path in the design.
• STA then produces reports showing timing violators, for example.
How Does STA Affect the RTL?

• After logic synthesis, STA is used to determine if the timing goals of the design are met.
• If the timing goals are not met, then the RTL may have to be modified.
• During the physical design (for example, floorplanning), STA is again used to determine if the timing goals of the design are met.
• If the timing goals are not met, then the RTL may have to be modified.
Example

• Assume the following RTL code:
  
  ```verilog
  always @ (posedge clk) begin
    z <= a + b;
  end
  ```

• Depending on the constraints, the netlist could be synthesized to:
  – A small, but slow ripple-carry-adder
  – A larger sized, but faster carry-look-ahead adder

• STA is used during synthesis to determine which adder meets the constraints of the design.
STA Usage During Logic Synthesis

- Let’s assume the logic synthesis tool has two adder architectures to choose from (RPL and CLA).
- The logic synthesis tool first implements a RPL adder, then performs static timing analysis.
  - If the design meets timing, it creates a netlist based on the RPL adder.
  - If the design does not meet timing, it modifies the architecture and creates a netlist based on the CLA adder.
• In physical design, STA is also used for optimization decision, just as it is done for logic synthesis.

• Consider the signal A driven by two or three buffers. STA is used during this optimization to determine if the extra buffer is needed in physical design, depending on the
  – Placement of the buffers
  – Connections between the buffers
  – Timing calculated through the path from A to DFF1
Topics in This Module

- Implementation flow overview
- Synthesis
- Static timing analysis
- Test
- Floorplanning and placement
- Timing closure
What Are DFT, BIST, and JTAG?

• Design for test (DFT)
  – Design techniques to add testability features in integrated circuits to make it easier to apply manufacturing tests

• Built-in self-test (BIST)
  – Extra logic in a design to verify all or a portion of the internal functionality

• Joint test action group (JTAG), boundary scan
  – Standard used to test board connectivity using boundary scan
Implementation Flow Overview

- Logic Synthesis
- Gates
- Floorplanning
- Placement
- Clock Tree
- Route
- GDSII
- RTL
- Static Timing Analysis
- Test
- Timing Closure
- Place and Route
DFT

- In short, the registers in a design are swapped with "scannable" registers or registers with muxed inputs.
  - In normal mode, the registers perform their normal operation.
  - In test mode, the registers for scan-chains used by the automated testers for manufacturing tests.
How Does DFT Affect the RTL?

• Timing
  – The additional mux in front of the RTL requires the designer to take this extra level of logic into account when writing the RTL.

• Area and power
  – The additional mux also increases the overall area and power of the design.

• DFT rules
  – Registers inside of the design must be observable and controllable, so the RTL code may need modification to make sure the DFT rules are met.
BIST

• In short, BIST structures are added to memory (MBIST) or logic (LBIST) to help improve the overall testing efficiency of the integrated circuit.
How Does BIST Affect the RTL?

• Area and power
  – The addition of the MBIST and LBIST structures should be considered when calculating the overall area and power goals for the design.

• Timing
  – Timing is affected by the additional logic, so this should be considered when writing the RTL.
In short, JTAG allows for a standard method to check the traces of a board using boundary scan.

- Each chip has standard ports and a tap controller (TC).
- Each chip has special scannable IOs that form a chain.
- Using boundary scan, patterns are shifted into the IOs of Chip1 and received on the IOs of Chip2.
- The tester can determine if there are shorts or open in the board traces by comparing the data shifted into Chip1 with the data shifted out of Chip2.
How Does JTAG Affect the RTL?

• Not very much. Since JTAG is a very mature standard and many chips require it, the TC and IOs are often taken into account at the chip level already.

• Chip architects must budget for JTAG at the chip and board level.
Topics in This Module

• Implementation flow overview
• Synthesis
• Static timing analysis
• Test
✓ Floorplanning and placement
• Timing closure
What Are Floorplanning and Placement?

- **Floorplanning**
  - Process of laying out the physical partitions of a design to determine the size and connectivity of each partition relative to the chip

- **Placement**
  - Process of finding specific locations for each discrete component of an integrated circuit
Implementation Flow Overview

Logic Synthesis

Gates

Floorplanning

Placement

Clock Tree

Route

GDSII

RTL

Place and Route

Timing Closure

Static Timing Analysis

Test

Timing Closure
Inputs and Outputs

- The gate-level netlist from logicsynthesis is the main input to floorplanning.
- Constraints are needed so that timing with STA can be accurate and measured against the specifications of the design.
- Technology library is needed, which contains the timing information for each discrete logic gate or macro.
- Physical library is needed, which contains information about the shape and connectivity of the technology library cells.
- The output is a floorplan, which is used to drive the rest of the implementation flow.
Floorplanning

• In short, floorplanning is where the RTL designer gets a more realistic view of the design in physical terms.
  – The chip architect partitions the design, and the physical design engineer allocated various areas and aspect ratios for each block.
  – The RTL designer uses the information for the given block and modifies the code accordingly.
Floorplanning (continued)

- Inside the block, the RTL designer can investigate the sub-partitions or the instantiated RAMs.
  - RAMs can be swapped here, possibly for smaller, slower, more power efficient RAMs.
  - Sub-partitions can be relocated, resized, etc.
  - Pins at the top level may be moved.
How Does Floorplanning Affect the RTL?

• Floorplanning has a significant affect on the constraints of a design and, therefore, has a significant affect on the RTL.
  – The size and the aspect ratio of the block may dictate changes to the RTL.
  – RAM sizes, aspect ratios, timing, and power characteristics may dictate changes to the RTL.
  – Pin placement may cause the input and output delays to the block to change significantly, requiring attention at the RTL level.
Placement

- In short, placement gives a more accurate picture of the timing and area versus logic synthesis.
- Consider the following example:
  - Gates U1 and U2 and part of the critical path.
  - In logic synthesis, U1 and U2 are “placed” close together, have minimal power and size, and the delay estimated for them is very small.
  - In placement, U1 and U2 are placed very far apart, U1 is upsized significantly, so the delay through U1 and through the connection is vastly different.
How Does Placement Affect the RTL?

• Placement has a significant affect on the accurate implementation of the design; therefore, it has a significant affect on the RTL.
  – The critical path may be very different between synthesis and placement, so the designer needs to be careful which path in the RTL to fix.
  – Because placement (and CTS) can add a significant amount of logic for placement optimization, buffering, etc., the designer needs an accurate area/power estimate to determine is the area/power goals will be met
Topics in This Module

• Implementation flow overview
• Synthesis
• Static timing analysis
• Test
• Floorplanning and placement
✓ Timing closure
What Is Timing Closure?

• One the most challenging aspects of chip design is achieving timing closure.

• **Definition:** Process of iterating through synthesis, physical design, and verification to converge on the timing of a digital design.

• **Example:** To achieve timing closure, the design team went through 20 iterations over a one-month period.
Implementation Flow Overview

Logic Synthesis

Gates

Floorplanning

Placement

Clock Tree

Route

GDSII

RTL

Static Timing Analysis

Test

Place and Route

Timing Closure
Timing Closure

- Timing closure is a general term that encompasses many of the processes in chip implementation.
  - As the design goes through its various forms (RTL → gates → placed-gates → CTS-gates → routed-gates → GDSII), the timing for the design becomes more accurate.
  - Changes to the design at each stage may or may not improve the timing.
  - Changes to the design to fix a path (such as physical verification or a hold-time violation) may break another path.
How Does Timing Closure Affect the RTL?

• CTS adds the actual clock buffers, whereas the clocks were “ideal” beforehand.

• Actual clock skew and latency need to be considered for timing.
How Does Timing Closure Affect the RTL? (continued)

• Routing adds the actual wire connections, whereas the routing interconnects were estimated beforehand.
  – Actual delays with crosstalk can affect timing.
  – Congestion can affect routability, which in turn can affect the floorplan, which in turn can affect the overall design constraints greatly.
Routing adds the actual wire connections, whereas the routing interconnects were estimated beforehand.

- Actual delays with crosstalk can affect timing.
- Congestion can affect routability, which in turn can affect the floorplan, which in turn can affect the overall design constraints greatly.
How Does Timing Closure Affect the RTL? (continued)

• CTS, routing, and physical verification can cause changes that affect timing and, therefore, could have an affect on the RTL.
  – Changes to the netlist itself, or to the placement, CTS, or routing can fix the timing violation without having to modify the RTL.
  – In some cases, modifying the RTL is the only choice.
  – Modifying the RTL is usually the last resort, because it has a very significant impact on schedule.
# Implementation on RTL

## Phase in Implementation and Comments

<table>
<thead>
<tr>
<th></th>
<th>Phase in Implementation</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Synthesis</td>
<td>Early</td>
<td>Timing, Area, and Power are Estimated</td>
</tr>
<tr>
<td>Static Timing Analysis</td>
<td>Throughout</td>
<td>Timing is measured throughout the flow</td>
</tr>
<tr>
<td>DFT</td>
<td>Throughout</td>
<td>Timing, Area, Power, DFT Rules are affected</td>
</tr>
<tr>
<td>BIST</td>
<td>Throughout</td>
<td>Timing, Area and Power are affected</td>
</tr>
<tr>
<td>JTAG</td>
<td>Throughout</td>
<td>Design is affected at the chip level</td>
</tr>
<tr>
<td>Floorplanning</td>
<td>Early/Middle</td>
<td>Design is constrained more accurately here</td>
</tr>
<tr>
<td>Placement</td>
<td>Middle</td>
<td>Timing, Area and Power are more accurate</td>
</tr>
<tr>
<td>Timing Closure</td>
<td>Late</td>
<td>Timing can be affected by various Issues</td>
</tr>
</tbody>
</table>
Summary

• The implementation flow starts with RTL and progress through a series of varying representations.
• There are many things to consider when coding RTL, both on the front end (from specification to microarchitecture) and on the back end, through synthesis and place/route.
• As the design progresses through the flow, more accurate representations of the design are created, thus more accurate measures of timing, area, and power.
• RTL changes during this process ensure that the design meets its overall requirements and specifications.
• Changes to the RTL very late in the process impact the schedule significantly.