[Part 1 discusses exploring the process technology to learn its capabilities and limitations, including evaluating the technology libraries, determining the implementation tools and flows, and capturing the SoC requirements.]

**Die-Size Estimation**

Die size and power estimations are at the foundation of SoC implementation. The key is how early and how accurately can it be done. These two parameters are the main data point for making some critical decisions early on. Freezing the die size in the early phase of SoC development gives a solid foundation for the physical designer, but it is a challenge to come up with an optimum and realistic estimation.

The top-level physical designer should engage with the RTL designer early on in terms of assessing the growth of the blocks and then binding them into the floorplan. The physical designer should come up with an early floorplan and continue to refine it for optimum and realistic die size. Memories significantly impact the die size, hence the front-end engineer should try to close on number of memory requirements early on. It is wise to keep a margin for the growth of the blocks.

Additional routing requirements due to DFT must be accounted for. It's very important to engage with the top-level DFT architect while estimating the die size. The routing requirement can vary significantly based on the Scan and BIST architecture. For example, if the compressor is sitting within a block, then the routing requirements at the top level will be less; and if the compressor is sitting outside the block, then the compressor grouping and placement will determine the congestion within the vertical and horizontal channels. Similarly the BIST controller within each block, and its communication with the top-level server, will also be important to understand while estimating the die size.

Die-size estimation and early physical architecture must go hand-in-hand. For channel based hierarchical designs, appropriate estimation of vertical and horizontal channel widths is important. While estimating the channel requirements, it is important that the physical designer considers all the crucial factors, such as the routing requirements for critical clocks, critical signal routing, additional routs due to DFT and MBIST and space margins for addressing crosstalk. A designer should understand the metal stack and decide how to allocate the routing resources.

Another aspect that cannot be ignored is the packaging requirement. For example the ball out and the bump plan can have a significant impact on die size. There could be a business decision for a certain kind of package and package design, which might drive the die size. There could be a legacy ball map, which might impact the placement of some physical IP or block and therefore affect the floor plan. So it is important to look into this early on. It's better if the bump plan can be made near
final before finalizing the die size. Package design and die design must go hand-in-hand.

It's paramount to engage with packaging while estimating the die size. A continuous methodical and data driven interaction between the physical designer and package engineer is a must. There are many key parameters that package engineers validate and simulate prior to determining the package requirement, which depend upon data from physical designers.

Some of the factors related to design are die size, pad ring design, special IO constraints, interconnect technology (types of flip-chip, wire bond) and voltage domains. Additionally, there are other package-related analyses such as mechanical performance, electrical performance, thermal performance, reliability and most important the cost.

**Power Requirements, Estimation and Analysis**

Establishing the power requirement and power estimation has to start right from the inception of the ASIC. The power requirement could be driven more from the business logistics, so accurate power estimation must compliment this requirement. Power estimation is a continual refinement process and it should start right from the front-end-architecture stage. Typically every design house has a process for power estimation, which starts with a spreadsheet. However, the key aspect is to upgrade this spreadsheet calculation with the silicon data.

Power requirements and power estimation could become an important driving factor for architecting the ASIC from a power consumption standpoint. These factors can influence major decisions about the direction of the design cycle. These decisions could be as major as redefining the product, changing the technology or changing the complete ASIC development flow. At the architectural level it could change the micro architecture, downgrade/upgrade the performance or shut down some part of the logic, etc. The directions from power requirements and associated modification in architecture could also lead to the introduction of high-level clock gating at the RTL stage.

When it comes to physical implementation for power requirements, it's a new domain of discussion and decisions altogether. One could decide to design for low power using multi-voltage and power shut-off techniques. Low-power design can be yet another time consuming challenge, so the schedule must also be kept in mind. One could go for a simpler low-power technique, like clock gating at synthesis, optimizing for VT, etc. So the power requirement and estimation could drive a lot of parameters toward the backend, such as the power planning strategy and the metal stack. It can also impact the package decision and associated substrate design requirements. The packaging decision is very sensitive toward cost, so it's crucial that appropriate data is made available to freeze the package for the chip early on.

The rail analysis is traditionally considered toward the signoff phase, but it's important to look into this early on to avoid power plan iteration at the later design cycle. It should be planned once the design has run through the PD cycle once. This data could be very crucial in validating the power plan strategy and overall floor plan. Accuracy of analysis is the key. Designers should be careful in choosing the analysis corner, while generating the power analysis data for rail analysis.

Knowing where and how to apply power rail analysis can save a great deal of time in power planning and verification. For today's designs, both static and dynamic analysis should be utilized from initial floor planning (power planning) through sign off.

Important points to consider when determining a methodology for power rail analysis include:
- Use static analysis to generate robust power rails (widths, vias, etc.).
- Use dynamic analysis to optimize the insertion of de-coupling capacitance.
- In the case of power gate design, use static analysis to optimize power switch sizes to minimize IR drop. Use dynamic power-up analysis to optimize power switch sizes to control power-up ramp time.
- Use both static and dynamic analysis early and late in the design flow.
- Establish IR drop limits based on understanding how IR drop can affect timing.
- Try to optimize for decoupling capacitors early in the flow, since late optimization for de-caps can lead to major re-work.

Another important part of rail analysis is EM analysis. Designers must ensure that the design meets the DC current density limits of the power mesh and the power rails before finalizing the power plan. It becomes more critical if the design is packed with memories. The power tapping to the memories needs to be sufficient to sustain the current densities.

Signal EM is also becoming crucial for high-frequency designs at technology nodes below 40nm. Signal EM can impact the clock network. However, if proper care is taken, such as NDR for clock network, then chances of signal EM violation can be reduced. Before analyzing the EM violation, the designer must validate the DC and AC current limits in the tech file and validate it through DRM.

The designer should also be aware of the operating conditions while generating the current density models for EM analysis. Typically EM is signed off at 110C, and tech file and DRM provide the table for 110C. One need not signoff EM at the timing corner (i.e., 125C).

In summary, there is a lot at stake when the power requirement is established and power is estimated. Hence, it's extremely important to have as much accurate data as possible early on in the process in order to make the decisions. Because there is so much at stake in terms of providing the direction for ASIC development, it's absolutely necessary to make the decision in the very early stages of ASIC development. It is also advisable to do a complete rail analysis (i.e., static, dynamic IR and EM analysis) well before the tape-out phase. If touching a lot of metal layers to fix rail violations post timing closure occurs, it could cause schedule overhead.

**Entry and Exit Criteria and QoR**

**Entry and Exit Criteria and QoR**

It is a common practice to divide the entire ASIC development cycle into different, well-defined phases, and to take advantage of the learning process throughout the phases. It is crucial to establish a clear-cut entry and exit criteria through the phases, and define the appropriate hand-off criteria across various functions.

Often, back-end activities begin in parallel with various front-end components that are under preliminary development stages. Some blocks' RTL will be very stable while some will continue changing until very late in the cycle. This is different from a one-shot full-chip RTL/netlist hand-off, and leads to the customization of the back-end implementation approach.

It's important to establish QoR criteria, keeping the entire back-end implementation in mind, so that the RTL designer can judge and track the quality of the block being developed. This is one of the key aspects in netlist hand-off criteria, which should be pushed up from the back end. A few important guidelines for the netlist hand-off are:

a. Have a guideline for the max logic depth (without buffers) based upon the operating frequency.
b. The no-load synthesis or pre-layout Static Timing Analysis (STA) uncertainty (i.e. what percentage of the clock period should be left as a margin for physical design). For example, a PD and timing engineer would want to see a block meeting pre-layout timing with an uncertainty of 40% of the clock period. Though it's design dependent, typically 35 to 45% of the clock period for a design running at around 750 MHz should be kept as a margin for the physical design.

c. Physical synthesis is a must approach for synthesis from quality of the netlist as well as the back-end to front-end correlation standpoint. However, care must be taken while accounting for the PD margin through uncertainty. The following parameters (to account for PD margins) should be considered for synthesis of a block in a hierarchical design:
   I. Apply derate while running physical synthesis: Derate = account for OCV + account for miscorrelation between physical synthesis and sign-off
   II. Account for the following through uncertainty during synthesis: Uncertainty = account for PLL jitter + account for skew + account for crosstalk induced from the top level.

**Physical Architecture**

Physical architecture is an integral part of the SoC development process. Physical architecture has become crucial for large and complex chips with hierarchical design flows because they are more likely to have long inter-block paths whose delays make timing closure difficult, leading to time-consuming and unpredictable tape-out schedules. In effect, the complex hierarchical SoC development moves the physical architecture into the front-end space and should begin in parallel with RTL development. It must be treated as a front-end activity. The outcome of the physical architecture process is a seed for actual chip floor planning.

The physical architecture process should address all the issues related to design partitioning, a broad level placement of logical blocks, understanding inter-block communication on the chip, etc. A thorough physical architecture of the ASIC can prevent major design problems and it is certainly one real motivator for improving floor plans and the power plan process. Also, the knowledge gained through the physical architecture process could provide crucial data for an early high-level business decision, such as financial viability of the design.

A good physical architecture helps ensure timing closure in many ways: by arranging blocks such that critical paths are short, by preventing routing congestion that can lead to longer paths, by integrating hard IP in an efficient way, by eliminating the need for over-the-top routing for noise-sensitive blocks, etc. The challenge is to create a floor plan with good area efficiency, to conserve silicon real estate and leave sufficient routing area, and to do it as early in the design process as possible.

Critical inter-block connections and on-chip buses require special allowances to ensure they will be able to meet timing constraints. Signal integrity is also important, especially for buses, where crosstalk from simultaneously switching drivers is more likely to cause signal glitches. Analog IP raises additional signal integrity issues due to noise sensitivity. Failure to deal adequately with any of these issues during floor planning can lead to numerous design iterations and/or chip re-spins.

**DFT Planning**

**DFT Planning**

Designing with a thorough understanding of Design for Test (DFT) can greatly increase DFT effectiveness and greatly reduce the DFT effort and timing needed to get the design through implementation. The first step toward this is to put the requirements in the DFT specs and this should happen as early as possible in the device specification and architecture stage.
With the advancements in manufacturing technology, and the increases in design complexity, it has become necessary to cover the entire functionality of an ASIC under the test. The SoC developer wants to control and observe every node of the design before shipping it for functional regression.

The complexity includes a large number of flops, a large number of memories and multiple instances of IPs from different IP vendors. The large number of flops causes a large number of scan chains and lengthy scan chains. This causes an increase in test time and cost. In some cases it might become unmanageable in implementation and at test. This requires DFT to find the solution through scan chain compression and hierarchical scan chain insertion.

The large numbers of memory bits are even more complex from the DFT prospective. They cause yield issues, since at smaller geometry nodes, the Soft Error Rate increases. The TSMC recommends repair implementation if the memory bit counts are beyond 16Mbit. So, the huge memory count in the large chip requires DFT to architect hierarchical memory BIST and added reparability.

Repair implementation is another complex part of DFT, especially its verification. Another issue with a huge memory count chip is the power consumption during memory BIST mode, which may become unmanageable. Care must be taken while implementing BIST for low power designs; memory repair needs to be aware of the power mode of the chip and power consumption capacity of the package.

In the case of many IPs, the SoC designer must establish clear communication with the IP vendor in terms of test capability of their IPs, and factor that into the DFT implementation of the SoC. However, debugging IP failure with the rest of the logic becomes difficult, so the best practice is to isolate the IPs for the test. For the I/O critical path, care must be taken while implanting the DFT muxes on the I/O paths.

SoC developers are increasingly making the ASIC more generic (i.e., for multiple applications and to have control over hardware, firmware or software). DFT plays a key role in enabling such requirements. From the DFT prospective, it translates to binning requirements, and primarily it's MBIST and/or scan-based binning. These very crucial requirements need to be finalized before DFT architecture is implemented.

**Design Rules and RTL Coding Guidelines**

Most designs require some up front preparation to achieve high coverage from DFT techniques. This preparation can start from the RTL, which eventually generates the final logic gates; the gate levels rules that affect the circuit's testability and suggest how RTL code can be written to implement these guidelines.

One of the simplest and biggest implications in the RTL coding style is that the test signal 'Scan_Mode', which is only present in gate-level netlists, must be incorporated into the RTL. This is an added step that most designers are not familiar with, but it will save a tremendous amount of work in the back end when DFT is being inserted. The DFT engineer must guide and incorporate comprehensive design rule checks at the RTL that will reflect these rules, enabling the designers to check for them up front.

The most critical aspect of most physical design and DFT requirements is the clocking methodology. The ideal requirement is to have a clean, un-gated, single clock and no asynchronous resets used in the entire ASIC. However, this does not exist, so the DFT engineer must try to make some adjustments to bring the design as close to that as possible. The functional requirement dictates this variation on the design, and DFT should be flexible to adjust to that. Some of the basic points to be taken care of while coding the RTL are:
In the case of internally-generated clocks, the roots of the new clocks should have a new MUX added to them, and the original clock should be fed through it. The MUX's select pin should be controlled by Scan_Mode.

Gated clocks must be avoided unless specifically required for low-power applications. If gated clocks are unavoidable then a Scan_Mode disabling gate should be used.

All FFs should be controlled by the same edge type clock. It is undesirable from the DFT perspective. If unavoidable, special care must be taken during DFT implementation.

During scan, no clock will feed in to the data input of an FF - Clocks cannot influence the data content of any FF during scan.

All sets and resets should be controlled by a primary I/O pin.

Latches should be driven in to a transparent state during Scan_Mode.

All tri-state bus drivers need to be driven from a fully decoded set of FFs. All tri-state buffers feeding into a single node need to have one, and only one, output enabled during the scan shift and capture phases.

No combinational logic loops in the design - if there are any combinational logic loops in the design, they should be broken by a Scan_Mode signal.

**Scan Architecture**

Scan architecture is the basic foundation of DFT implementation. Prior to scan architecture, the DFT engineer must have a full understanding of the DFT requirements. Scan architecture can significantly influence the floor plan and timing closure, and the floor plan requirement can also significantly alter the scan architecture. For example, the number of I/O constraints can push for a scan compression implementation, and more stringent requirements might call for implementing the serializer.

Scan compression can also be implemented hierarchically or flat. This decision can also be influenced by the physical architecture. Identifying the number of scan I/Os and scan clocks is a very crucial calculation for DFT implementation as well as the floor plan.

Some of the important points to be kept in mind while architecting scan are:

- Plan chip-level scan issues before you start the block-level design.
- Know the requirements and limitations of your chip testers.
- Avoid design practices that lead to non-scanable elements. Handle all non-scan elements with care.
- Break all combinational logic feedback loops.
- Handle internal tri-state busses with care. Avoid bus contention by design.
- All clocks and asynchronous resets must come from chip pins during the scan mode.
- Handle mixing flip flops triggered off different edges of the clock with care.
- Handle multiple clock domains with care to avoid potential timing problems.
- All scan elements on a scan chain should be in the same clock domain. Otherwise care must be taken in lockup latch introduction while mixing the clocks.

**Scan Compression**

With the cost of test increasing exponentially higher in 40nm geometries and below, a solution to this problem is needed for large designs with huge flops counts. Scan compression offers the best tradeoff for being able to reduce both test time and test volume, the two key components driving test cost.
There are different technical combinations and logistics which can drive the way scan compression can be done. For a perfectly balanced scan chain flow, the best bet is to adopt a flat flow where compression logic must be inserted at the top level, but be prepared for more layout congestion due to more cells and routes at the top level.

However, if you still need to do hierarchal scan insertion, you'll need all blocks available at the time of scan insertion. For the hierarchal scan compression flow, there could be a restriction on how many scan chain I/Os are available externally and on the numerous physical blocks in the design. This directly results in limited compression capabilities even though test/fault coverage did not suffer. Even though the compression hardware was physically at the top level, there could still be significantly less congestion than in the flat flow.

A hierarchal grouped design flow offers the best trade-off because it has the least amount of congestion, and allows a very good compression ratio, even though the scan chains may not be as well balanced as in the flat flow. At the same time, test/fault coverage does not suffer. Another advantage is that the compression logic can be inserted as soon as the block is made available.

After implementing the compression, if there are still limitations on scan I/O, then the serializer technique can be adopted. Serializer is a new technique where the serializer component sits between the compressor and the scan I/Os. It basically pushes the pattern at higher frequency serially to the internal scan chains, thereby reducing the number of scan I/O requirements. So, for a 4:1 serializer, you need one scan I/O for 4 compressor output scan chains. However, this could impose some challenges in scan I/O timing closure.

**MBIST Repair Implementation**

Yield becomes an issue with huge memory counts designs. Memory repair techniques have been developed to improve the yield. This requires basic memory to support repair. Different memory vendors provide different ways of repairing. One or more of following features must be present:

- One or more redundant columns
- One or more redundant rows
- A combination of redundant rows/columns
- A way to enable redundant row/column selection

Following are some of the considerations prior to repair implementations:

- Row Vs column repair: Which ones to choose?
- General rule:
  - Smaller size: column repair
  - Bigger size: memory row repair
  - Look into memory physical architecture
- Consult foundry/DRM
- Sometimes cost associated might make the repair a **"no-solution"**
- Important calculations in cost of repair:
  - Analyze the additional logic added for repair
  - Calculate no-repair vs. post-repair yield

Basically there are two kinds of repair solutions:

- Hard repair:
- The repair signature is stored permanently on-device
  - E-fuse or laser fuse
- Soft repair
  - Signature is stored in flip-flops
  - Device needs to re-load upon power up
- Hard repair is solid
  - Soft repair can add on top of hard repair for in-field usage

Repair Flow

Summary
The complex designs at lower geometry require a lot of emphasis to be given to estimation and analysis. It requires a comprehensive planning on execution. The above analysis will lay down the
foundation for final physical implementation.

In Part 3: Floorplanning and PnR

More about Bhupesh Dasila