ASIC Design Flow And Methodology – An Overview

Ashish A Shetty
RV College Of Engineering-Bengaluru-59
Wipro Limited- STA/DFT Engineer

***Abstract** — ASICs are complex. Can contain more than millions of transistors which makes it impossible to do the entire design at one level of abstraction - say through the schematic entry or through custom layout! The individual engineers will be having limited roles with in one sub domain of a VLSI Design and not aware of how their part of work is relevant. A very few people will be having over view of the entire sub domains of VLSI Design – from RTL to GDS. This paper attempts to connect each subdomain with other VLSI Design subdomains, explain how they interact each other. Though GPPs and ASPPs designed and manufactured with similar Methodologies, this paper will focus more on ASIC Design Methodologies.*

***Keywords** — VLSI, ASIC Design, ASIC Design Methodology, Semicustom Design Methodology, Full Custom Design Methodology, FPGA Design Methodology.*

I. INTRODUCTION

An application-specific integrated circuit (ASIC) is an integrated circuit (IC) customized for a particular use, rather than intended for general-purpose use. ASICs are complex. Can contain more than millions of transistors which makes it impossible to do the entire design through the schematic entry (and then custom layout) alone. Example of non-ASICs which include standard parts like memory chips like RAM, ROM, SRAM or microprocessors. These come under the category of ASSP (Application Specific Standard Products). Example of ASICs are the chips of a satellite, chips which contain microprocessor as a block with other logic combined or a chip which controls the interface between memory and the CPU unit in a workstation.

ASICs design has 2 types of flow which include the Full-custom flow and the Semi-custom flow. Full custom flow includes all the custom made logic cells and the mask layers. Manufacturing lead time is approx. 8 week which does not include design time. Full-custom makes sense when there is no suitable existing library present or existing libraries are not fast enough or there may be issues with the power or the size. Advantages of full custom design include the performance and the cost. Advantages come with certain disadvantages of design time, high risk and complexity. Semi-custom flow uses standard cells, full- custom blocks, Functional standard blocks cores, etc. The present-day ASICs are designed through "Cell- based design flow" methodology. Reuse of standard cells and macro cells helps save time and money. Only time overhead is to design standard library or the cells and time to fabricate all the ASIC layers for the new design.

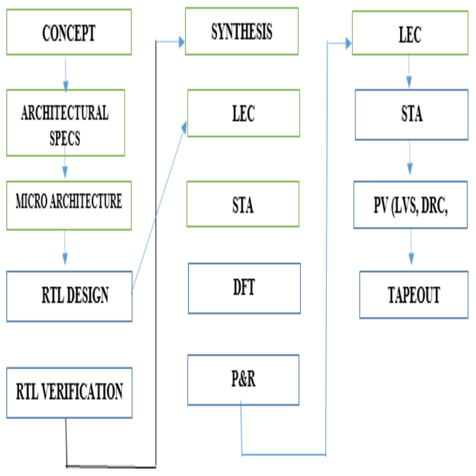
In general, the ASIC design happens at the following stages.

1. RTL Design.
2. Logic Synthesis.
3. Physical Design.

Anything which design has to be verified. All the above design abstractions will be verified. The corresponding verification steps are

1. RTL Verification.
2. Logic Equivalence Check.
3. Physical Verification.

The 'Design for Testability' is one more step which can start from RTL stage and gets integrated at the Logic Synthesis stage. Timing Analysis (STA) will be done on Physical Design as well as Gate-Level Netlist from the Synthesis step.



II. ASIC SPECIFICATION AND RTL CODING

The initial phase in ASIC design flow is characterizing the specifications. This progression includes showcase reviews with potential clients to make sense of the necessities and conversing with the innovation specialists to measure future patterns. The last is especially significant in light of the fact that ASIC design cycle might be anyplace between a half year to two years. It is in this manner essential to anticipate and foresee what patterns would be significant 1-2 years down the line on the off chance that one needs to pitch their item to a wide group of onlookers.

This advertising research converts into high-level item specifications like top-level usefulness of what we mean to do with your ASIC, explicit calculation that you need to execute, clock frequencies that would make the item speaking to the clients, bundle type-Ball Grid Array (BGA) or CSP (Chip Scale Package) and so forth., control supply, correspondence conventions that will assist interface with the outside world, temperature range that you would need your item to work in. Building up a careful and right detail, as a rule, sets a strong establishment for the ASIC design. The specialized specifications need refinement of the specialized necessities after some time, yet it's critical to cover the data in an unambiguous way. Verilog can be utilized at various levels of deliberation. Be that as it may, how valuable are these different levels of deliberation with regards to utilizing a Verilog involves incredible intrigue which fluctuates each new design. Verilog isn't in a perfect world appropriate for conceptual framework level recreation, preceding the equipment programming split. This is somewhat tended to by System Verilog. Not at all like VHDL, which has support for client characterized types and over-burden administrators which enable the designer to extract his work into the space of the issue, Verilog confines the designer to work with pre-characterized framework capacities and undertakings for stochastic recreation and can be utilized for demonstrating execution, throughput and queueing yet just to the extent that those inherent language highlights permit. Designers every so often utilizes the stochastic level of deliberation for this period of the design procedure.

Due to certain drawbacks, designers migrate towards Register Transfer Level (RTL). In simple terms, RTL or Register transfer level is the transfer of data from one register to another. It is an abstraction of the digital portion of a design. RTL is usually captured using Hardware descriptive languages or HDL like Verilog or VHDL. RTL is based on synchronous logic and contains three primary pieces namely, registers which hold the state information, combinatorial logic which defines the next state inputs and clocks that control when the state changes.

III. LINTING

RTL has to follow the specified coding guidelines. Reviewing the code manually against each of the coding guidelines is a tough task. This is where the linting place an important role. The process of analyzing and purifying the code is called Linting. The RTL lint tool analyzes a RTL code before it is simulated or synthesized. Checks for sanity and common mistakes. It catches common errors both simulation wise and synthesis. A linting tool has a set of rules and guidelines, the tool will analyze against these guidelines. It reports wherever there is a violation of rules. The rules are customized to the requirements and the specification of the project. Standard tools used in the industry are SpyGlass Linting tool –From arena, LEDA Linting tool- From Synopsis, Cadence Conformal LEC and explorer.

IV. TOP LEVEL INTEGRATION AND VERIFICATION

The digital circuit design needs to be tested for its exact functionality. Chip designing and manufacturing is an expensive industry. The results of getting it wrong are not only expensive but also disastrous to the design house, to say the least. RTL verification remains one of the most challenging aspects of any digital system design development. There is no one straight forward method and instead, there is a lot of dependency on the design team to come up with a lot of verification or test designs in conjunction with the verification team. A lot of ad hoc test scripts, tools for verifications. There is as yet an absence of a standard or formally dressed worthy methodology. The regular approach in the business is practical approval. The useful model of the advanced plan is simulated with important input stimuli and the output is seen is checked for the normal behavior. The model utilized for recreation is the RTL description. The reproduction includes applying examples of test information at the contributions of the model, at that point utilizing the recreation programming or machine to process the simulated qualities at the yields lastly checking the accuracy of the qualities got. Validation is carried out at the module level and chip level.

At the module level, each module of the design is verified independently. It involves producing an entire set of tests, each of which makes sure it checks the proper behavior of one specific aspect or functionality of that module. Each test includes a set of input patterns to stimulate the module and a portion that verifies that the output of the module corresponds to what is expected. The design of these tests is generally very time consuming since each of them has to be handcrafted by the verification engineering team.

Recently, a few CAD tools have become available to support functional validation: they mainly provide more powerful and compact language primitives to describe the test patterns and to check the outputs of the module, thus saving some test development time. Functional validation can provide only partial coverage because of its approach; the objective is thus to maximize coverage for the design under test. Various measures of coverage are in use: for instance, line coverage counts the lines of the RTL description that have been activated during the simulation.

Another type is state coverage which measures the number of all the possible configurations of a design that have been simulated, that is, validated. This measure is particularly valuable when an estimate of the size of the total state space of the design is available: in this situation, the designer can use state coverage to quantify the fraction of the design that he/she has verified.

Due to the limitations of functional validation, new alternative techniques have received increasing interest. The common trait of these techniques is the attempt to provide some type of mathematical proof that design is correct, thus guaranteeing that some aspect or property of the circuit behavior holds under every circumstance, and thus its validity is not limited only to the set of test patterns that have been checked. These techniques go under the name of formal verification and have been studied mostly in academic research settings for the past 25 years. The most common approach to functional validation involves the use of a logic simulator software. A commonly deployed architecture is based on the leveled compiled code logic simulator.

Another common validation methodology approach in the industry is the pseudo-random simulation. Pseudo-random simulation is mostly used to provide chip level validation and to complement Stand Alone Testing validation at the module level. This approach involves running a logic simulation with stimulus generated randomly but within specific constraints. For instance, a constraint could specify that the reset sequence is only initiated 1% of the time. Or it could specify some high-level flow of the randomly generated test while leaving the specific vectors to be randomly determined. The major advantage of pseudo-random simulation is that the burden on the engineering team for test development is greatly reduced. However, since there is very limited control on the direction of the design state exploration, it is hard to achieve high coverage with this approach and to avoid producing just many similar redundant tests that have limited incremental usefulness.

V. SYNTHESIS

Synthesis converts HDL description or RTL to gate-level implementation using standard gates, cell library, and memory blocks. It contains translation, optimization, and mapping. The translation is the conversion of the HDL format code to gate level Boolean equation. Optimization step involves reducing the Boolean equation. Then these optimized Boolean equations are mapped to library files. The resultant netlist is fed as input to all the back end design tools. A synthesis engineer has to take area, timing, power, and wiring to the consideration. A synthesis engineer will get the inputs from the verification team i.e. a verified RTL design and constraints and also the library files. The output would be a synthesized netlist and reports. Synthesis is a broad term often used to describe very different tools. Synthesis can include silicon compilers and function generators used by ASIC vendors to produce regular RAM and ROM type structures. Synthesis in the context of this tutorial refers to generating random logic structures from Verilog descriptions.

The meaning of Verilog for simulation is thrown in stone and revered in the Language Reference Manual. Different tools which use Verilog, for example, synthesis, will make their own elucidation of the Verilog

language. There is an IEEE standard for Verilog synthesis (IEEE Std. 1364.1-2002) however no vendor clings carefully to it.

It is not sufficient that the Verilog is functionally correct; it must be written in such a way that it directs the synthesis tool to generate good hardware, and moreover, the Verilog must be matched to the idiosyncrasies of the particular synthesis tool being used. There are currently three kinds of synthesis:

- Behavioral synthesis
- High-level synthesis
- RTL synthesis

There is some overlap between these three synthesis domains. We will concentrate on RTL synthesis, which is by far the most common. The essence of the RTL code is that operations described in Verilog are tied to particular clock cycles. The synthesized netlist exhibits the same clock-by-clock cycle behavior, allowing the RTL test bench to be easily re-used for gate-level simulation.

VI. LOGIC EQUIVALENCE CHECK (LEC)

LEC is also is known as formal verification. Verification through simulation applies countless vectors to the circuit and afterward thinks about the subsequent output vectors to anticipated values. As designs become bigger and increasingly mind-boggling and require more simulation vectors, regression testing with customary simulation tools turns into a bottleneck in the design flow. Formal verification utilizes mathematical methods to contrast the logic with being checked against either a logical detail or a reference structure. In contrast to verification through reproduction, formal verification does not require input vectors. As formal verification thinks about just logical functions amid correlations, it is autonomous of the plan's physical properties, for example, layout and timing.

VII. PHYSICAL DESIGN

A fully functional RTL which is synthesized undergoes a design partition or physical hierarchy which is further taken to block level physical design and synthesis activity. This phase of ASIC flow will decide the number of chips per wafer and ultimately the cost. Initial power estimation will help the die design and also help choose the correct package. Early and accurate selection of package and technology node, lead to a quicker solution, cost saving and fast time to market. The physical hierarchy of the blocks will undergo block level placement and route, STA and DFM activities. A layout of chip is realized using Automatic placement and Routing (APR) tools. This is used to decide the floor plan and the power plan, Clock tree synthesis, and placement of standard cells and routing them. Some of the industry level standard tools are ICC from the synopsis, Olympus from mentor graphics, Silicon ensemble from cadence, etc.

Technological library, Timing constraints (SDC), gate level netlist and interconnection models and also the timing scenarios and scan chain information are given as inputs to an APR tool. As a result layout, gate level netlist with clock tree, reports, and scan chain information are given as output. Once the APR flow is complete there should be zero DRC, LVS and timing violations.

Design timing setup is done by loading the standard libraries and technological files. Then come Floor planning and power planning where the placement of standard cells, IOs and macros are decided and power planning is connecting of metal wires from the power source to macros and standard cells to meet the IR goals. Then the process of automatic placing of standard cells during floor planning such that it meets the timing constraints as given in the Standard Delay Constraints (SDC) is called as placement. Then the clock pins of all sequential elements are connected to clock nets so that the skew is minimal, this process is called Clock Tree Synthesis (CTS). The culmination of the process is the routing where the creation of metal contact between pins of all the cells in accordance with the gate level netlist such that timing, DRC and LVS are met.

VIII. PHYSICAL VERIFICATION

Physical verification consists of the following sub steps:

A) Design Rule check (DRC) to achieve a high overall yield and reliability for the design.

B) Layout versus Schematic checks (LVS) to ensure that the layout really represents the circuit you desire to fabricate.

C) Antenna Check to avoid gate oxide damage due to the collection of charge due to antenna effect.

Inputs to run DRC are GDSII data, rule deck (provided by foundry), layer map information defined by foundry and DRC documentation. This should give a DRC clean layout which does not guarantee functional silicon. Running LVS is a large part of verifying that we have actually assembled the intended design. GDSII data, Verilog netlist, macro, IOs, standard cell source spice netlist, rule deck, layer map information from foundry and design information in IO ports, power-ground are taken as input for LVS flow. This flow will give a netlist which is ready for tape out which may undergo some changes to meet the timing constraints in Engineering order change (ECO).

IX. STATIC TIMING ANALYSIS (STA)

Required specification of a design can be defined by the functional specification and the timing specifications. Functional verification and Logic equivalence check (LEC) will take care of functional specification. Gate-level simulation (GLS) and STA take care of timing implementation. Static timing analysis comes in the gate level domain. There are different stages a STA engineer plays important roles. A netlist is obtained from the synthesis team to check for setup time and hold time violation. Different elements behave in a different manner. And each of the element has their limitations but are bound to different conditions by varying operation environment. Different corners are set i.e. worst, best and typical corner based on voltage, temperature and process variation (PVT). Best case or the fast corner checks for hold which has fast process, low temperature, and very high supply voltage. Worst case/slow corner which checks for setup violation with high temperature, slow process, and low voltage. A typical corner is for both setup and hold violations. STA has three major steps: 1) Break the design into different Fan-in cones to the destination flop.2) Delay for each is calculated mathematically instead of patterns or vectors. 3) All path delays are checked against time constraints. Inputs for a STA engineer are Netlist which is a gate level description, Timing constraints i.e. standard delay constraints (SDC), Standard cell library, net delay and the parasitic extracted from the physical design tool. The output is the reports which will point out all kind of violations and timing related details. Certain Violations are resolved in the phase of ECO where extra logics are added to meet the timing cutoff. Standard Tools used are Prime time, Formality etc.

X. DESIGN FOR TESTABILITY (DFT)

Design for Testability is a method which results in testable design. DFT helps to verify the functionality of the design. DFT is a process of inserting a test logic to make the Design under Test (DUT) controllable and observable. DFT provides a structured way of testing chip manufacturing defects. It provides faster ramp- to-volume production, quantifiable test metrics, and reduction in debug time. DFT mainly concentrates on manufacturing defects like short ground or power, open circuits, resistive paths or metal bridges.

A DFT engineer needs the netlist which is generated by the synthesis team which has undergone the verification process. A DFT engineer inserts the memory built-in test first to isolate and test the memories. Memory tested design is inserted with the scan logic where all the normal flops are converted into scan flops. Scan insertion process converts all the flops to scan flops and then they are stitched into a single chain. Due to overhead tester time Embedded Deterministic Test (EDT) logic is inserted. EDT logic reduces the tester time by dividing the long chains into shorter chains. A netlist is generated which is scan inserted. Netlist written out by scan insertion process is given to a Static Timing Analysis engineer to determine the critical path, parasitic and multicycle paths are detected. These serve as input to the Automatic Test Pattern Generation (ATPG). ATPG process feeds input response and also generates the expected response. A test procedure file will have the whole description of the process. The generated patterns are simulated to check the correctness and compared with the expected response. Then the patterns are validated pass/fail. DFT engineer plays an important role in post-silicon testing. The patterns generated in ATPG is tested on Automatic Test Equipment. Common Tools used are Tetra max by synopsis, Tessent by mentor graphics, VCS by synopsis, Questa Sim by mentor graphics etc.

XI. CONCLUSION

In summary, the ASIC design is verified against Functional requirements, Timing requirements and foundry requirements (at one or more abstraction level) and taped out. This can vary between different companies but this paper will give a supporting backbone for all kind of flows.

REFERENCE

[1] Bertacco, Valeria. (2006). Design and Verification of Digital Systems. 10.1007/0-387-29906-8_2.

[2] Reyserhove, Hans & Dehaene, Wim. (2019). Efficient VLSI Design Flow. 10.1007/978-3-030-12485-4_3.

[3] Serpanos, Dimitrios & Wolf, Marilyn. (2019). Challenges and Opportunities in VLSI IoT Devices and Systems. IEEE Design & Test. PP. 1-1. 10.1109/MDAT.2019.2917178.

[4] Wang, Laung-Terng. (2009). Design for Testability. 10.1016/B978-0-12-374364-0.50010-2.

[5] Kukimoto, Yuji & Berkelaar, Michel & Sakallah, Karem. (2011). Static Timing Analysis. 10.1007/978-1-4615-0817-5_14.