Challenges in the Design of a Scalable Data-Acquisition and Processing System-on-Silicon

Karanth S, Soujanna Sarkar, R. Venkatraman, Shyam S. Jagini,
Venkatesh N, Jagdish C. Rao, Udayakumar H, Manohar S., K.P Sheshadri
Somsubhra Talapatra, Parag Mhatre, Jais Abraham, Rubin Parekhji
{karanth,souj,rvenkat,shyam,n-venkatesh,j-rao,uday,s-manohar,sheshadri,s-talapatra1,parag,jais,parekhji}@ti.com

Texas Instruments India Ltd.,
Golf View Homes, Murugeshpalya, Bangalore – 560 017, India.

Abstract—*Increasing complexity of the functionalities and the resultant growth in number of gates integrated in a chip coupled with shrinking geometries and short cycle time requirements bring in several challenges into the design of present day VLSI chips. In this paper we present the challenges faced and the approaches successfully adopted in the design of a complex 2.5 million gate high bandwidth data acquisition and processing VLSI chip (a trace-receiver chip, code-named Drishti) in a deep sub-micron technology at Texas Instruments India. The very high design complexity arises due to the rich architecture of the trace-receiver chip, the aggressive timing and performance requirements and its large size. The trace-receiver chip is highly configurable and scalable, thereby catering to both low-end systems, which are cost sensitive, and high-end applications, which demand performance. We present the innovative approaches that were applied to address the challenges encountered in meeting the aggressive design goals (which include functionality, timing, testability, manufacturability, reliability, system issues etc) and to bring the product early to market. Efficient logical and physical partitioning, design reuse and DFT strategies are a few of the techniques that were applied in this design. We present these along with details on the various analyses carried out as part of the design, including signal-integrity, reliability and system-level analyses, which were very critical in ensuring design-closure.*

Index terms—*deep sub-micron, timing closure, functional verification, electrical analysis, design-closure*

I. INTRODUCTION

Today's VLSI designs are characterized by large gate counts, high complexity and aggressive time-to-market requirements. Reuse of pre-existing building blocks reduces the design cycle time. Optimal logical partitioning is a critical first step in the design process. Functional and timing verification are one of the biggest challenges. To ensure correct behavior of the system requires extensive simulations of various modes. Physical design of a large chip in a deep submicron technology requires optimal physical partitioning; and a placement that is routable, meets timing goals and addresses reliability and signal integrity issues. Achieving timing closure is difficult because of the interconnect delays and the logic depth. Design for testability (DFT) is an important careabout that needs to be taken into consideration. Board-level performance needs to be kept in mind while optimizing the chip interfaces. In this paper we highlight the challenges faced in the design of a scalable data-acquisition and processing system, a trace-receiver chip code-named Drishti.

The rest of the paper is organised as follows. We give an overview of the design in Section II and describe the motivation and challenges in Section III. Section IV presents the design techniques to address the challenges in functional verification, synthesis, physical design, timing closure, test, electrical/reliability analysis and package optimization. Observations and future work are stated in Section V. Conclusions are given in Section VI.

II. DESIGN OVERVIEW

Drishti is a trace receiver that receives trace data from a real time system consisting of one or more digital signal processors (DSP) and/or micro-controllers. Trace data is used to enable debugging of real time application running on the DSP/micro-controller. The architecture of the Drishti chip is generic and scalable and caters to a variety of trace requirements like different clocking modes, high-speed trace, wide trace widths, very long traces or combinations of one or more of them. Drishti also provides for 3 identical and independently configurable I/O trace interfaces. Drishti can receive trace data on only one trace port at any point of time. Each of the 24 bits on the trace port can be configured to belong to any one of the 4 channels. The data streaming out of trace port feeds into 4 channels. Drishti also provides for exception-analysis capability and real-time sampling of trace data and real-time upload of trace data in specific cases. Fig. 1 shows the trace system comprising of one or more Drishti chips, a host PC and a DSP/micro-controller based application. The received data is transferred to internal and/or external memory (trace memory). Host software running on a personal computer (PC) will retrieve the trace data from the trace memory.

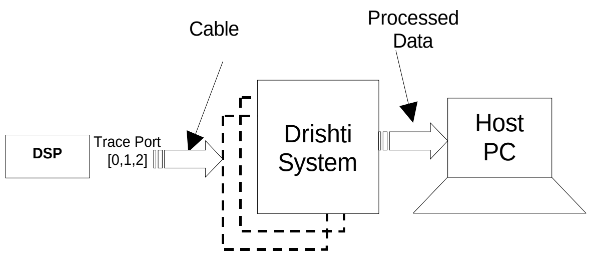

Figure 1. Drishti Trace System

The trace data received on the input port through a cable is subjected to significant duty cycle and skew distortions. The chip has a programmable skew adjust feature by which the data bits can be aligned with the clock (Fig. 2). Drishti can extract data received from input port at a high bandwidth of 7.992 Gbits/sec maximum rate (333 Mbits/sec/pin with a 24-bit trace width). Drishti can be programmed to receive data on a maximum of 4 channels, which can act either in independent mode, or in locked manner. It can retrieve data clocked on single edge or dual-edges. Interleaving trace data across multiple Drishti chips can expand trace memory. Ganging multiple Drishti chips can expand trace width. Multiple chips maintain synchronization between each other through exchange of handshaking signals. Each Drishti recorder stores data in its own trace memory. The different modes of Drishti can be programmed through an asynchronous host interface port.

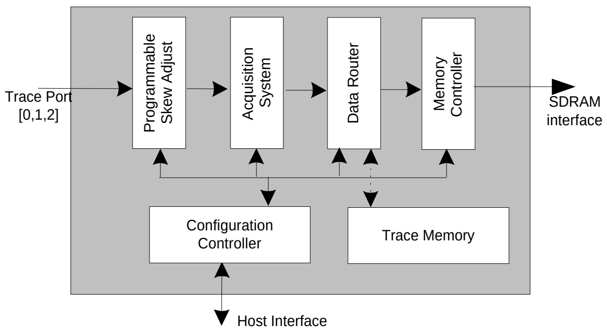

Figure 2. Drishti block diagram

Salient Drishti features are captured in Table 1. 1.5V I/Os are used for Drishti to Drishti communication and 3.3V I/Os are used for interfacing with external SDRAMs, application system. Host interface port also uses 3.3V I/Os.

Equivalent Gate count	2.5 Million
Memory	2.1 Mbits
Logic Gate Count	1.7 Million
Clock Frequency	167 MHz
Maximum Data & I/O Rates	333 MHz
Die Size	16 mm x 16 mm
Power Supply	Core Logic - 1.5 V I/O – 1.5V & 3.3 V
Estimated Power dissipation	3.9 W
Minimum feature size	0.13 µm
Minimum metal pitch	0.5 µm
Number of metal layers	5
Package	600 pin BGA
Signal Pins	360

Table 1. Features Summary

III. MOTIVATION

The design of a complex, multi-million-gate Drishti offers huge technical challenges. Challenges span almost every domain – design implementation and methodology and deep sub-micron process issues. Ensuring first-pass silicon success in the presence of very aggressive performance and time-to-market goals adds to the challenge.

As described in the above section, the Drishti trace chip required a high-bandwidth data interface. Considering the high-performance nature of several IO buses, a proper selection of the IO buffers and optimal pin assignment was a huge challenge [1]. In addition, the presence of dual edge timing causes higher toggle rates resulting in drawing considerable power supply current and ground bounce, forcing attention to signal integrity issues. The chip also has a programmable skew adjust capability for the received trace signals. Any of the 24 trace inputs can be programmed to a clock pin due to which every trace signal needs to be carefully handled. The design of the programmable skew adjust module posed several challenges in the areas of verification, logic synthesis and physical design. The need for tight control over the skew and the asynchronous nature of logic did not allow the use of conventional logic synthesis for this part of the design. The additional requirement to obtain uniform delays on the delay chain that implements the skew adjustment meant that conventional timing-driven physical design flow and a random-logic placement approach would not be acceptable. Additional verification challenges resulted due to the need to debug multiple Drishti. The requirement for three data ports supported within the chip to exhibit exactly the same behavior and presence of both on-chip synchronous and asynchronous logic demanded a very optimal, up-front partitioning of the chip from a logic and physical design implementation perspective.

The complexity of Drishti verification is mainly posed from the generic and programmable nature of the architecture and its scalability. Simulation speed is a big issue due to the large size of Drishti design. Several key features like interleaving and ganging are defined with multiple Drishti chips functioning in tandem, which adds to the simulation time complexity and pushes the computing requirements and tool capabilities to the limits. Many configurations in both single and multi-Drishti topology generate innumerable functional boundaries that need to be verified. Significant proportion of the logic (around 40%) is asynchronous. Static Timing Analysis (STA) tools do not fully comprehend the asynchronous logic. Gate level timing simulations are therefore necessary to ensure proper operation of the asynchronous logic. Design "reuse" [2] became a reality on the Drishti design, hence inheriting nearly 300K gates of reusable logic. Comprehending boundary conditions with reuse logic in context of Drishti brings its own nuances into the functional verification.

Achieving rapid timing closure is a key challenge, as we look at large, complex chip designs at process technologies below 0.15um. Increased parasitics on the circuit interconnect, additional coupling capacitance on routes due to aggressive metal pitches, have all attributed to interconnect delays dominating over the path delays. Designers, therefore, need accurate measurements of these interconnect delays in order to predict system timing and enable timing signoff. The increased complexity in solving this problem on the Drishti design was due to the presence of several (maximum 30) levels of logic between pipeline stages. Design reuse required a seamless chip-integration and timing closure methodology that ensured meeting aggressive chip-level timing goals. Traditional approaches to achieving timing closure using wire-load models have been very iterative and deficient in providing good quality of results (QOR) especially on large designs [3]. Bridging the gap between front-end design and the physical implementation using physical synthesis approaches is key to, not only achieving rapid timing closure, but also enabling higher QOR and performance entitlement from the given process technology. Physical synthesis was widely used on the Drishti design to enable timing closure. An aggressive performance target also necessitates a tight control over power IR drops on a large chip like Drishti [4]. Early analysis and review of the chip-level power distribution and coming up with innovative techniques to derive the power grid while ensuring routability is a key challenge. Traditionally, chip designers have given a lot of attention to achieving timing closure [5]. However, given today's aggressive time-to-market goals, achieving overall design closure is more important and critical, particularly because of increased challenges in dealing with reliability issues like electro-migration, crosstalk noise, charge-collecting antennas and channel hot carriers. Given all this, limiting chip-level power consumption and ensuring chip routability are also essential. Focussing on design closure on Drishti enabled us to efficiently deal with all the above issues without significantly impacting the cycle-time.

To achieve high test-coverage, Design for Test (DFT) needed to address large chunks of asynchronous logic, a number of embedded memories of various sizes, and the presence of several clock domains.

Finally, the design and implementation of a complex, performance-critical device like Drishti with all the challenges described above and the need to meet tight market windows requires an efficient concurrent design process rather than a traditional sequential design approach [6,7]. It is essential that verification be performed in concurrence with design-space explorations involving area, delay, power, testability, reliability, pinout and package.

IV. DESIGN TECHNIQUES

A. *Verification*

The need for concurrent exploration at the architecture level, longer simulation run times and large size of the design necessitated the adoption of a bottom-up approach towards the complete design-verification of Drishti.

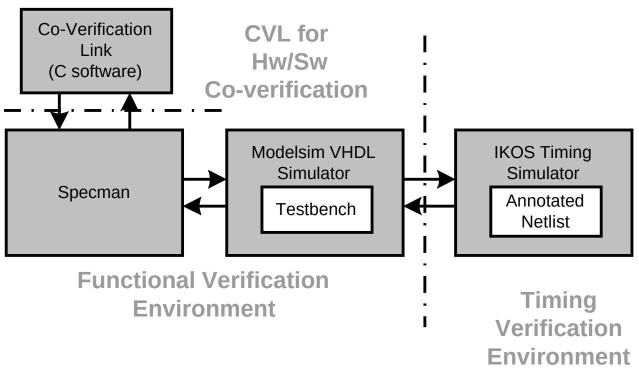
Co-Verification Link (C software)
CVL for Hw/Sw Co-verification
Specman
Modelsim VHDL Simulator
Testbench
IKOS Timing Simulator
Annotated Netlist
Functional Verification Environment
Timing Verification Environment

Figure 3. Drishti Verification Environment

Focused effort was given to the various aspects of timing-simulation due to the presence of large asynchronous logic. Emulation of the architecture was also done to validate some of the system level aspects.

Verification was first performed at the leaf-level modules (40K gates). The modules were thereafter verified then at the subsystem level (500K gates). Final verification was carried out on the full chip, which was followed by a complete system-level verification. Boundary conditions between interacting modules were extensively covered in module- and subsystem-level verification.

The following verification checklist was used to cover all aspects of Drishti functionality and thereby its conformance to specification:

1) 100% function point coverage in a multi-device configuration. Many of the functions are achieved by using a system comprising of 9 Drishti chips working together.
2) 100% statement, branch coverage and 80% condition coverage. Toggle coverage at the top level was set to 100%
3) Simulation of the Drishti in multiple clock frequency ratios to cover throughput and deadlock situations
4) Use of hardware accelerators for accelerated simulation at gate level with timings.
5) Simulating the netlist with post layout timings to uncover any timing related issues in the asynchronous logic as well as to extract information on duty cycle degradation using IKOS^{TM} logic simulation accelerator.
6) Use of transaction based test bench approach to test system at a higher functional domain. The pin level interactions with the device were done using transaction macros in 'e' language of Specman^{TM} verification tool.
7) High level software test cases exercised through a co-verification link built onto the Specman^{TM} transaction macros and verified on RTL as well as gate level netlist with timings both on Modelsim^{TM} simulator and IKOS^{TM} hardware accelerator.
8) Exercising the skew adjust algorithm in post layout timing simulations to verify for non-existence of unwanted glitches and check timing of asynchronous logic.
9) Use of hardware emulators for system regressions
10) Use of equivalence checkers to check integrity between RTL versus netlist and netlist versus netlist

The transactions through the chip interfaces were captured as simple high level functions such as reading a register, writing to a register and waiting for an interrupt. Each of these high-level transactions was mapped to pin level transitions on each pin with their temporal relations. Low-level transaction functions were first verified before using them in chip level test cases. This made it easy to comprehend and build tests at a higher level, which were independent of the underlying simulator or the environment.

Reviewing the test plans against functional specifications ensures functional coverage of Drishti. Exercising multiple test conditions across functions ensured cross-functional coverage, i.e. the correctness or failure of a functional behavior in conjunction with another functional behavior.

Implementation boundary conditions were addressed using RTL code coverage. RTL code coverage is a necessary but not a sufficient condition to ensure proper functionality.

The module and subsystems functionality was verified with tests built using the third party EDA simulation tools. At the device and subsystem level it was easy to build tests using the control software. The control software configures the device for various operational modes as well as does a variety of host data read operations. The software was interfaced to the transaction layer built into the third party simulation tool. The software was used to exercise test cases at the device and system level with the underlying implementation transparent in this setup. The conceptual flow of the software interaction with the hardware simulator is shown in Fig 1. The hardware simulator supports timing-simulations using delay information back-annotated from layout-extraction. The asynchronous data extraction logic and the skew adjustment functions required extensive host data read operations, which was effectively done by simulating the control software with the actual hardware implementation.

FPGA based emulation was used for the synchronous logic due to the high simulation run times. In FPGA based emulation, it is possible to exercise many system specific tests in a shorter time compared to a logic simulator environment. Finally hardware acceleration of logic simulation using IKOS^{TM} helped speed up the gate level timing simulations. These simulations were primarily aimed to verify asynchronous logic timing and clock duty cycle degradations, which are difficult to check using Static Timing Analysis tools.

Several incremental changes to the design netlist take place during full chip timing optimizations and layout iterations. It is extremely time consuming to rerun the entire suite of tests on modified netlist. Equivalence checking between the original regressed netlist and incremental netlist saved several weeks of simulation time and enables shorter design cycle [8].

Use of a variety of different simulation environments to verify different aspects of the design with a common set of test suites was a key aspect of Drishti verification methodology.

B. Package and IO Planning

The input-output (I/O) buffers, pinout and package affect the system performance significantly. The I/O buffers were chosen based on the results of extensive SPICE simulations. The dual-edge high-speed interfaces required the usage of high-speed transceiver logic (HSTL) I/O buffers while the relatively low speed interfaces used 3.3V low-voltage CMOS (LVCMOS) I/Os. As the HSTL I/Os operated at 1.5V, this led to ~4.8 times power savings compared to LVCMOS.

Pinout planning involved the estimation of power and ground pins and their optimal placement. Estimates of power dissipation of the chip helped in the calculation of the power and ground pin counts. Consequently, a 600-pin ball grid array (BGA) package was chosen. The placement of the power, ground and signal pins was done to ensure that simultaneous switching resulting in ground bounce and cross-talk are within acceptable limits. To prevent the high switching of the I/Os from affecting the core logic operation, core and the I/O ground were isolated. SPICE simulations of the I/O buffers were performed using a package SPICE deck with a transmission line model for the off-chip interfaces to sign-off on the pinout.

This analysis ensured that the right package and input-output buffers were chosen for the chip so as not to impact signal integrity adversely.

C. Test Methodology

Full scan methodology was used for the synchronous logic and functional fault grading is considered on asynchronous logic to increase the test-coverage. In scan mode, a single clock tree feeds all the scan flip-flops.

Embedded memories take up a significant area of the die. Built In Self-Test (BIST) was used on embedded memories, which have poor controllability. Parallel Module Test (PMT) was additionally used for large memories to detect word line transition faults, not covered by Memory BIST. Parallel Signature Analyzer (PSA) was added to improve observe-ability of memory outputs.

D. Synthesis and Timing Closure

Very aggressive timing goals and the large size of the design were the key challenges towards achieving timing closure on Drishti. The limited solution space resulting due to the many design constraints posed a big challenge.

To achieve identical and aggressive performance across the three I/O trace interfaces, a bottom-up methodology was adopted for the IO trace-port design. All trace signals were needed to have identical timing characteristics. The trace-port subsystem was designed as an independent module and was instantiated thrice at the chip-level to achieve this purpose. The trace port was further partitioned into three modules (the programmable skew adjust, the acquisition unit and the IO module) based on the different timing characteristics required, and also the nature of the logic. The programmable skew adjust was completely asynchronous and required very tight control of skew across the 24 trace signals. The acquisition unit operated on dual edges of the clock and was designed to tolerate up to 25%-75% duty cycle on the input trace signals. Programmability of any of the 24 input signals as a clock posed a further challenge within the acquisition design and required all the trace signals to be designed as clocks. The IO module was designed to output all the trace signals on both the edges of the clock with minimum distortion and skew.

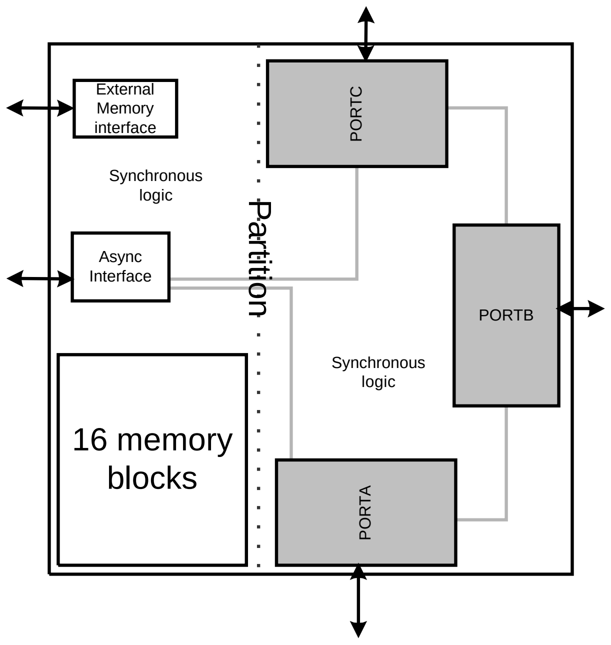

Figure 4. Drishti Partitions

The above-mentioned design hierarchy coupled with the presence of asynchronous design elements at the lowest level of design posed considerable challenge in timing-closure at each level. The main reason for this was the use of conventional synchronous design tools for logic synthesis and physical design, where it was impossible to create models of the asynchronous elements for optimization. This necessitated manual optimization of many timing-paths, the goal being to get the maximum performance possible, so that design constraints during block-use can be met with relative ease. Another challenge was to introduce as much concurrency in the design activities as possible without compromising on accuracy. This was particularly important in the light of multiple levels of nesting in the data acquisition subsystem.

At the chip-level, Drishti was partitioned into two blocks. Timing budgets were generated for these blocks based on the chip floorplan. Each block was then taken through physical synthesis separately and integrated at the top level. Programmation memories and registers, which were needed to be written by and read from an asynchronous external configuration bus, were spread across many modules all over the chip. In the worst scenario, the address, data and control signals needed to be fed to all the memories and registers. This design scenario, which could have resulted in an exponential increase in the number of on-chip busses and thereby increased routing congestion, was handled by connecting all the memories and registers in a daisy chain.

Sufficient margins were provided in the timing budgets for each of the blocks to account for the interconnect delays between them at the top-level. Since the design was finally flattened before routing, optimizations for timing were performed at the chip-level rather than the block-level.

E. Physical Design

The main challenge in the design of Drishti was handling the design size. Ensuring good QoR with respect to timing and routability without impacting cycle-time was a critical requirement. To meet these challenges, a two-phased design approach was adopted. The first phase was to design the trace-port sub-system bottom-up so that this module could be instantiated thrice at the top-level with identical performance characteristics across all the instantiations. The focus of the second phase was the chip-level, where a pseudo-hierarchical methodology was adopted.

i) Trace-Port Design

The trace-port, being very performance-critical in terms of signal-delays and skews, was multiply partitioned, so that each of the blocks could be designed independent of others. This module was designed bottom-up, and had six different sub-modules internally, with two levels of hierarchy. The partitioning criteria were the presence of asynchronous logic and special design requirements like delay matching across many signal lines.

In order that the asynchronous logic, notably the programmable delay adjust elements, be handled with adequate care during placement and routing, these were designed as individual delay blocks. This ensured that through multiple instantiations, identical performance characteristics were possible. The programmable delay element, being one of the most important components of the design, was optimized for the best performance. Rigorous SPICE analyses were performed on this block to ensure that all the design requirements were met. Block aspect ratio was one of the most important factors considered during the performance optimization of this block. This block was about 1K equivalent gates in size. Handling such small sizes in a 2.5 million-gate design was another challenge, with the bottom-up design methodology requiring these blocks to be designed completely before they can be used at the top-level. Furthermore, with a large number of these small blocks being used in the design, routing congestion was one of the main issues faced at various stages of the block hierarchy. This was addressed by optimizing the floorplan at every stage, as well as by designing lower-level blocks using lower layers of metal routes only, so that the higher layers were available for use at the top-level. Managing the iterations in a bottom-up physical design methodology, considering the number of the blocks involved, and the levels of nesting was another challenge.

Skews on various signal lines were minimized by constructing skew-balanced trees on these nets, like those usually constructed on clocks. Furthermore these nets were all delay-balanced using sufficient delays on the faster nets during the construction of the balanced trees.

With these blocks designed stand-alone, all electrical and reliability analyses were done at the block-level, so that all the required changes could be made at the block-level itself. This is required because changes in the lower-level block late in the design cycle would prove costly in terms of turn-around time in a bottom-up design methodology. Appropriate models were created for electro-migration, power drop and timing for the lower-level blocks for analysis in the block-use phase. However, the analysis was carried out flat for crosstalk noise estimation using the full sub-block design detail.

ii) Chip-Level Design

Physical synthesis approaches were used for initial timing-closure at the chip-level. In order that the physical synthesis tools handle the large design size, the entire design was partitioned into two large blocks. The placement of standard cells in these blocks was then imported into the chip-level flat environment for routing. Minimal interaction between the two large blocks was ensured through careful partitioning, and this ensured that top-level optimization required on timing-paths across these two blocks was minimal.

At the chip-level, all high-frequency inputs and outputs were grouped. The chip, although being IO limited, had pockets of high routing congestion. The issue was with physical synthesis approaches not handling the presence of the large number hard-blocks optimally with respect to placement and routing obstruction definitions. This was addressed by performing a selective placement adjustment of the standard cells, so that routability was ensured. The resultant impact on timing was handled through multiple incremental optimization iterations.

Chip-level power-distribution was one of the other major challenges, particularly with the large die-size (256mm^2). Initial power-dissipation estimates were used to optimally distribute the power-ports around the periphery. Care was taken to optimally provide power inputs to power-critical portions of the chip, with both horizontal and vertical straps creating a distribution grid. Via locations and sizes on the power grid were optimized so that congestion was not impacted, as also the resistive drops on the power structures. The overall resistive drops on the power-lines were analyzed and the power-network was adjusted to address issues of high resistive drops in certain areas of the die.

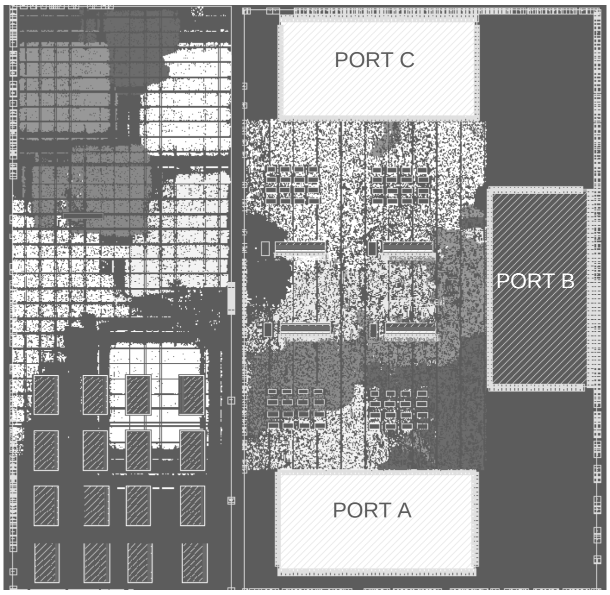

Figure 5 Drishti Chip with the Cell Placements

Placement filler-cells, which could be metal-programmed to function as logic cells were extensively used in this design. Using these metal programmable cells as filler-cells ensured an optimal distribution of these cells on the die. This helped in implementing logic changes required late in the design-cycle, with minimal impact on the turn-around time. Such changes were necessitated by the results of timing-verification of the design, which went concurrently with the physical design.

As mentioned earlier, the individual modules within the trace-port subsystem, as well as the chip were rigorously analyzed for crosstalk noise, charge-collecting antennas on the signal lines and electromigration issues on the metal structures. Adequate care was taken to ensure a correct-by-construction design scenario, so that the final layout changes required to address violations were under control [9]. The design methodology that we had adopted [10] helped us meet the challenge of design closure on a design of this complexity.

V. OBSERVATIONS AND FUTURE WORK

The experience gained in designing the trace-receiver chip has been immense in almost all the areas of chip-design. We have been able to make the right changes in the RTL to improve test-coverage on the synchronous blocks, without impacting the cycle-time. We have successfully adopted a physical-synthesis-based methodology for achieving timing-closure in minimum cycle-time. We have also adopted a pseudo-hierarchical physical-design strategy, with the top-level placement happening hierarchically, and the routing being flat. This method has helped us address the issue of design-size without compromising on performance. We have also successfully adopted a physical-design methodology that ensures a correct-by-construction design. This has helped us reduce overall cycle-time by cutting down on layout modifications towards the end of the design cycle. We have successfully achieved 167 MHz performance on silicon, with the various features of the device working as intended.

We are presently working on coming up with a strategy for pattern-generation for the test and fault-simulation of asynchronous blocks. We are also working on adopting a fully hierarchical physical-design methodology, so that the various DSM issues are addressed at smaller design sizes, rather than on a large design. This can help us achieve better cycle-times, with scope for design changes made easier because of the modularity. This is consistent with our concurrent engineering approach, which can necessitate such a change late in the design cycle. We plan to integrate automatic budgeting tools for timing, so that chip-level timing-issues are addressed very early in the design-cycle. We also plan to optimally use decoupling capacitor cells so that the issue of peak IR drop on the power rails can be addressed. We also plan to incorporate the impact of crosstalk noise on signal delays so that a comprehensive timing analysis can be enabled. All these, coupled with our existing methodology for addressing the various reliability issues, should help us achieve faster design-closure.

VI. CONCLUSIONS

This paper highlights the challenges that we had faced in the design of the trace-receiver chip, Drishti. We also detail the way these design challenges were addressed, in terms of verification solutions, the timing-closure methodology, test-strategies and physical-design techniques.

VII. ACKNOWLEGDMENTS

The authors would like to thank Amit Brahme, Bilas Datta, and the rest of the Drishti design team at TI India for their contributions towards the design of this chip. The authors would like to express their gratitude to Ron Lerner, Kim-Williams Smith, Naveen Chava, Larry Hite, Frank Cano for their valuable inputs during the design. We express our gratitude to Mahesh Mehendale for his guidance and encouragement during the course of this design. The authors also acknowledge the help and support received through the course of this design from the TI ASIC group and Synopsys.

VIII. REFERENCES

1. Amit Brahme, Soujanna Sarkar, Ronald L. Lerner, Ramesh Ramamritham, Udayakumar H, “IO buffer selection, pinout optimization and package simulation challenges for a high performance trace chip”, *5th TI India Internal Technical Conference, 2001*.
2. Reinaldo A. Bergamaschi, William R. Lee, “Designing System-on-Chip using Cores”, *Design Automation Conference, 2000*.
3. K.Madathil, Jagdish C. Rao, Bilas Datta, Madhusudhan Rao, “A Links-to-Layout Flow for High-Speed DSP Cores in Deep-Submicron Technologies”, *Design Automation and Test, Europe 2000*.
4. Gregory Steele, David Overhauser, Steffen Rochel, Syed Zakir Hussain, “Full-chip verification methods for DSM power distribution systems”, *Design Automation Conference, 1998*.
5. Massimo Bombana et al, “Design Flow and Synthesis for ASICs: a case study”, *Design Automation Conference, 1995*.
6. Thomas F. Fox, “The Design of High Performance Microprocessors at Digital”, *Design Automation Conference, 1994*.
7. Thomas W. Albrecht, “Concurrent Design Methodology and Configuration Management of the Siemens EWSD-CCS7E Processor System Simulation”, *Design Automation Conference, 1995*.
8. Subash Chander G, Vaideeswaran S, “Addressing Verification Bottlenecks of Fully Synthesized Processor Cores Using Equivalence Checkers”, *Asia-Pacific Design Automation Conference, 2001*
9. Avinash Gautam, Jagdish C Rao, Rohit Rathi, H Udayakumar, “A Design-In Methodology to Ensure First-Time Success of Complex Digital Signal Processors”, *10th Intl. Conf. On VLSI Design, 1999*.
10. Amit Brahme et al., “A Reliable Design Methodology for Rapid Design Closure”, *Avant! AURORA, 2001.*