Islands of Synchronicity, a Design Methodology for SoC Design

A.P. Niranjan, Paul Wiscombe

Philips Semiconductors
SoC Architecture and Technology
San Jose, California, USA

Abstract

To meet the challenges of faster time to market and growing design complexity, a methodology and supporting infrastructure for advanced System-on-Chip design have been developed and applied to 0.13 micron technology designs.
The Islands of Synchronicity methodology uses locally synchronous islands to produce a timing-closure friendly design style that is widely applicable across different architectures. This approach enables a modular, hierarchical physical design strategy which significantly eases top-level timing closure problems. The resultant design flow is supported by the Skeleton of Reuse, a collection of IP generators and tools that automate many of the steps in SoC implementation.

1. Introduction

Advances in process technology have made it possible to fabricate IC's containing tens of millions of gates. However, the design community is faced with extreme challenges to create System-on-Chip (SoC) designs that can take advantage of the available process capability.

In this paper we describe a hierarchical design methodology known as "Islands of Synchronicity", and the tools that support it, that addresses some of these challenges. We specifically focus on the critical issues of clock distribution, top-level timing closure and hierarchical physical design.

Results are presented from three Philips SoC designs implemented in 130 nanometer CMOS technology, all using the Islands of Synchronicity approach to various degrees.

1.1. Clocking

Clock distribution in large SoC designs has become a major issue as design size and complexity has increased. With multiple clock domains, global clock distribution and clock tree balancing is difficult to achieve at any operating point, and is made even more so by variations in process, voltage and temperature across the die.

Globally synchronous designs with deep buffer trees can suffer from large instantaneous power surges and Electro Magnetic Interference (EMI) coincident with the clock switching activity. In addition, with the clock balanced across the entire SoC, significant energy is expended in the clock tree.

1.2. On-Chip Interconnect

As shown in Figure 1, interconnect delays have increased with each new geometry node [1] to the point that wire delay now dominates logic delay for long traces.

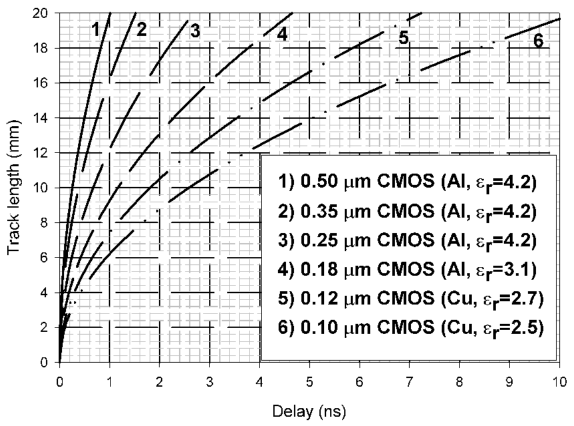

Figure 1. Interconnect Delay

In a globally synchronous environment, the maximum operating frequency of on-chip interconnect is limited by the absolute delay between the source and destination.

Signal buffering strategies, inserting buffers in the interconnect to balance clock trees, to resolve edge rates and to avoid cross-talk problems, further limit the speed at which interconnect can operate.

The combination of global clocking problems and interconnect performance limitations have lead to the requirement for alternative design approaches that alleviate timing closure issues and provide solutions that scale with technology.

1.3. Hierarchical Design

The move to a hierarchical approach is inevitable for most large SoC designs. Designs are just too large and complex to be completed flat. Rather than push EDA tools and compute resources beyond their capabilities, it makes sense to break designs into smaller, manageable pieces.

The design can then be implemented by different design teams in different locations, each with its own area of expertise, in a concurrent engineering environment.

A typical hierarchical design flow has three main steps as shown in Figure 2.

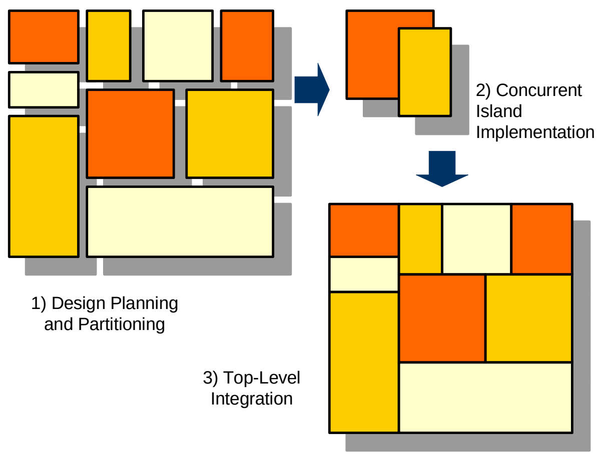
2) Concurrent Island Implementation
1) Design Planning and Partitioning
3) Top-Level Integration

Figure 2. Hierarchical Design Flow

Critical to the success of hierarchical design is the ability to bring the pieces of the design back together again in the final chip assembly stage and have them work together. Timing convergence and efficient physical design are two of the main challenges to resolve during chip assembly.

1.4. Islands of Synchronicity

The IoS methodology provides unique solutions for the hierarchical design process.

Firstly, it places more emphasis on the Design Planning phase of the design. Although the ultimate partitioning task still relies on the expertise of the chip architect, when using IoS, partitioning is heavily influenced by timing considerations and the nature of the on-chip communication between islands.

Secondly, during the Island Implementation phase, islands are kept updated of changes to the floorplan. During implementation a typical SoC design is being continually updated, and changes to an island or its neighbors may affect the details of the on-chip interconnect. By regenerating the interconnect and pin placements each island can be optimized to the new chip topology so that islands are preconditioned for top-level integration while they are being implemented.

At the same time, a skew-tolerant inter-island communication strategy keeps the timing of the island independent of the top-level design. It removes the need for global synchronous clocking and allows clock trees to be implemented locally within islands, making them easier to balance and uses fewer levels of buffering. This results in lower clock tree insertion delay and reduced power dissipation.

Finally, Top-Level Integration is simplified by the use of Asynchronous Communication Links (ACLs) for all inter-island communication. There are minimal timing dependencies between islands, greatly easing chip-level timing closure. The incremental interconnect regeneration that has occurred throughout the island implementation phase and the associated addition of feedthrough channels within islands, makes it possible to assembly the chip by abutment to minimize die area.

Additional benefit is gained because the islands define natural boundaries for implementing block-based Design for Test (DfT) and Design for Debug (DfD). The islands can also be readily extended to create “Islands of Power”, allowing different blocks to operate at different voltages, or in a power-down mode.

2. The Skeleton of Reuse

The supporting infrastructure for the IoS methodology, that we call the “Skeleton of Reuse“ (SoR), includes a configurable clock generation unit, on-chip interconnect “builders” and a design manipulation tool. They have been developed to automate creation of the top-level interconnect between islands, allowing the design infrastructure to evolve in step with the physical implementation of the design

Although this paper focuses on the timing and physical design aspects of IoS, the SoR is not limited to these aspects, it supports extensive SoC design solutions that include DfT, DfD and power management plus traffic generators and bus monitors to provide Quality of Service metrics.

2.1. On-Chip Interconnect

Traditional busses are inherently synchronous, and a synchronous bus that travels to several islands imposes many top level timing relationships on the SoC. IoS enables bus communication to proceed without a globally synchronous clock, removing these timing constraints and allowing islands to be implemented independently.

The SoR provides several types of skew-tolerant ACLs which are either source synchronous or asynchronous in nature, and which target specific data types. These are delivered to design teams as reconfigurable generator tools.

Table 1. Communication Types

Data Type	Communication Link
Streaming Data Bus	Clock Domain Crossing FIFO
Memory Data Bus	Source Synchronous MTL or DTL Transport Pipe
IO Bus	Device Control and Status Network

The communication links are built upon proprietary Philips IP reuse standards that use Device Transaction Layer (DTL) and Memory Transaction Layer (MTL) interconnect protocols. DTL and MTL are similar to, but precede, the VSI Alliance Virtual Component Interconnect standard [2], and a large portfolio of Philips IP is available that uses them. Thin adapters are available to connect blocks with other interfaces (such as AMBA) to DTL and MTL.

2.2. Clock Generation

An important component of the SoR is the central Clock Generation Unit (CGU). This configurable IP module generates and routes the appropriate clocks to the islands. Frequency synthesis, by means of PLLs and digital divider circuits, creates the necessary clock frequencies for the application. A connection fabric in the CGU allows clock frequencies to connect to different sources in a glitch-free manner, and includes start/stop functionality for power management purposes.

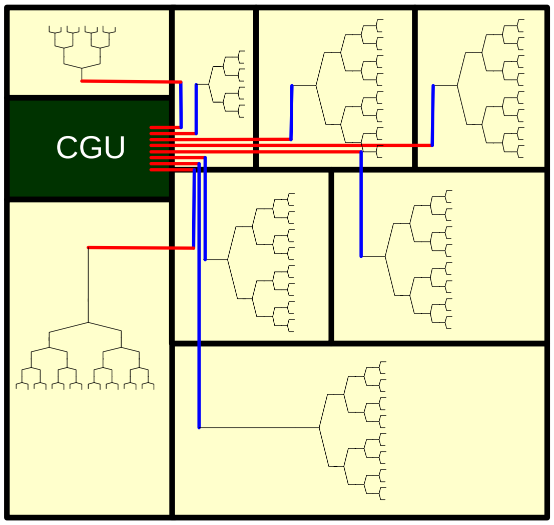

Figure 3. IoS Clock Distribution

In addition to application clocks, the CGU supports on-chip generation of clocks for delay fault testing and provides the precision break point capability and clock manipulation required for silicon debug.

Within the IoS methodology, clock nets from the CGU to the islands do not have skew constraints. Therefore, clock tree balancing is necessary only within the islands, not at the top level as indicated in Figure 3. This reduces clock tree logic depth and eases timing closure.

2.3. Design for Test

The SoR provides a DfT flow that maintains a relaxed test clock, autonomous islands for stand-alone vector generation, simple embedded core test shell insertion and delay fault test support.

All communication between islands is performed using asynchronous or source synchronous techniques. In functional mode, the non-deterministic nature of the asynchronous communication is not a concern. However, in test mode, DfT tools must force these nets to behave in a deterministic fashion.

Philips uses an embedded core technique [3] from its Core Test Action Group which is similar to the IEEE P1500 standard [4]. However, in the IoS methodology, test shells are added at island boundaries, rather than at IP boundaries, to create stand-alone test islands. Aligning test shells with physical islands simplifies the DfT flow. It allows scan chains to be reordered based on placement and Automated Test Pattern Generation can be run stand-alone on each island.

Open defects dominate DSM technologies. We rely on delay fault testing to detect them and to maintain product quality levels. The SoR CGU provides the double system clock required by sequential test pattern generation tools to execute scan-based delay-fault testing. On-chip generation of at-speed clocks for delay fault testing reduces the requirements of the production testers and can reduce test cost.

2.4. Design for Debug

The ability to quickly debug SoC designs in the application environment has become an essential capability for product development. Less than 40% of large SoC designs are bug-free first time [5], and an increasing proportion of development effort is spent isolating errors.

The SoR uses a low-overhead and scalable DfD architecture [6], providing the hardware to implement precision break points, data matching blocks and scan chain reconfiguration.

As with the DfT approach, debug modules are aligned with the island boundaries. They represent a small incremental overhead since they share much of their logic with the DfT circuitry. DfD modules are built entirely within the island, assembled at the top level with skew tolerant interconnect and accessed through the IEEE 1149.1 Test Access Port.

2.5. Design Manipulation Tool

Throughout the IoS design flow, an iterative approach is used to incrementally align global infrastructure resources, such as ACLs, to the physical topology. In the absence of a commercial tool to adequately support these processes, an internal design manipulation tool has been developed.

This automates and manages many of the steps of the design flow such as: converting logical and physical hierarchy (assigning ACL components across island boundaries); infrastructure regeneration; adding feedthroughs and buffers within islands; propagating timing constraints across hierarchy; creating island netlist abstractions and assisting with DfT logic integration.

3. IoS Hierarchical Design Flow

The IoS methodology enables an efficient hierarchical physical design flow. It does this using by applying two primary concepts.

Firstly, early consideration of the physical chip implementation in the design flow. Accurate prediction of the island size and form factor produces more accurate timing estimations. Knowledge of the adjacent islands and of the top-level chip routing requirements enables appropriate feedthroughs and buffering to be built into the island during implementation, rather than at chip assembly.

Secondly, recognizing that the IP features and functionality are the critical elements of the design, whereas interconnect is a reconfigurable resource that is closely tied to the physical implementation, and must be regenerated as the floorplan evolves. ACL communication strategies decouple the island from the details of the interconnect, so that regeneration can occur without impacting the timing of the island. They also make it possible to start island implementation ahead of finalizing the floorplan. Figure 4 shows a typical IoS design flow.

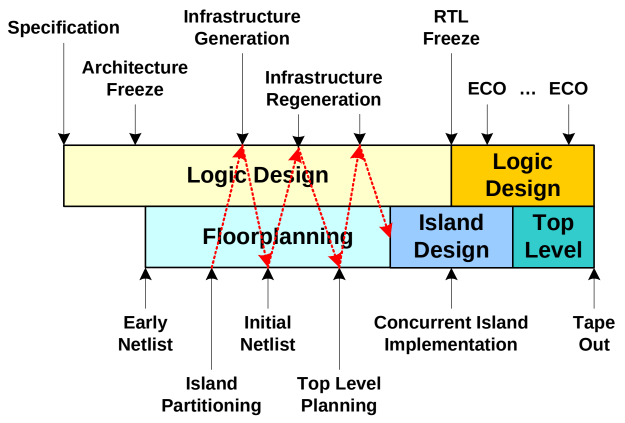

Figure 4. Islands of Synchronicity Design Flow

An early version of the infrastructure is used with the initial netlist and island partitioning to make an early start on floorplanning the design. As the physical design and island partitioning are refined, the infrastructure is regenerated to optimize the island interconnect to the new topology.

With a refined floorplan, engineering teams can work independently on each island to complete detailed routing and timing closure. At the end of the concurrent island physical implementation the islands are re-assembled into the final SoC design prior to tape out.

The SoR, with its automated infrastructure generation, decouples the front-end design process from chip integration allowing front-end designers to concentrate on the feature set of their design, rather than on details of the top-level infrastructure.

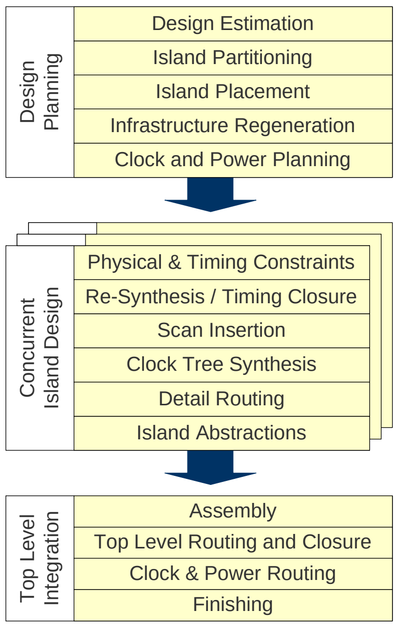

Figure 5. IoS Physical Design Flow

The IoS design flow is an evolution of a traditional hierarchical design flow. It consists of the three main stages shown in Figure 5.

More emphasis is placed on the Design Planning phase than in traditional design flows. This early planning, in conjunction with the IoS methodology, allows islands to be created independently and greatly simplifies top-level design integration.

3.1. Design Planning

With IoS, early design planning is the key to quickly achieving design convergence. This is enabled by iterative refinement of the floorplan using intermediate design estimates. Some special considerations are imposed by IoS above those for a standard hierarchical design flow.

- **Island Partitioning.** There is no automated partitioning solution. It relies on the expertise of the design architects who must consider factors such as clock domain crossings, size, functionality and inter-island communication links when partitioning the design.
- **Netlist Partitioning.** This must ensure that all inter-island links are skew-tolerant. For example, ACL IP blocks are split into separate islands in the physical hierarchy as shown in Figure 6. This process is supported by the design manipulation tool, and RTL code for the ACLs is structured to reflect the clock domain hierarchy of the block.

  At each iteration, the design's functional and timing integrity are confirmed using Formal Verification and Static Timing Analysis techniques respectively

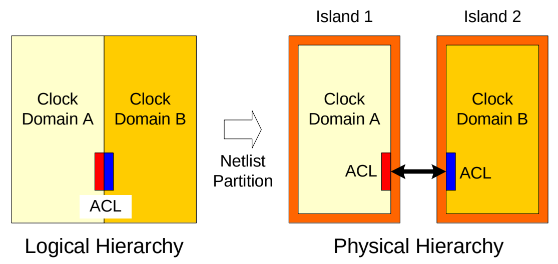

Figure 6. Netlist Partitioning

- **Infrastructure Regeneration**. As the floorplan is refined, the interconnect infrastructure is regenerated to match the new physical topology. There is no direct impact on the timing of the islands, but interconnect regeneration may affect the position of feedthrough areas and the pin placement.

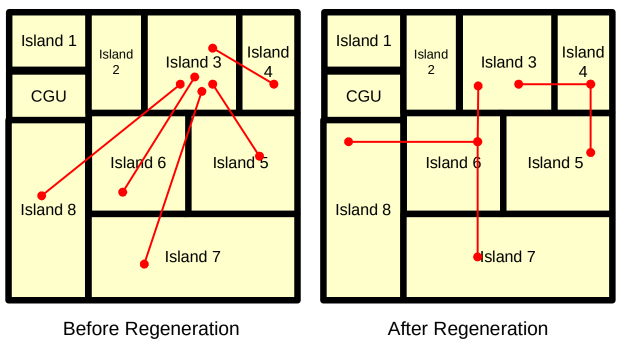

Figure 7. Routing Optimization

Figure 7 shows the reduction in routing complexity that is achieved after regenerating the infrastructure and pin placements.

- **Feedthroughs**. IoS uses edge-to-edge abutment of islands, so dedicated routing resources (feedthroughs) must be added within each island to accommodate the top-level interconnect. The design manipulation tool is used to insert buffers in the feedthrough areas to maintain signal skew within each ACL. This ensures that the timing of the island remains independent of the top-level timing and that the islands contain the routing areas needed for top-level integration.
- **Clock Planning**. The clock timing requirements are relaxed in IoS since there is no clock skew relationship to maintain between different islands. In fact, skew may be deliberately introduced between mesochronous clocks in different islands in order to temporally disperse the switching energy and reduce EMI.

3.2. Island Design

By the end of the design planning phase most of the key island attributes have been defined: physical and timing constraints, the position of feedthroughs and buffers, pin placements, clock planning and power planning.

Placement, routing and timing closure of each island can proceed in isolation, using traditional synthesis or physical synthesis techniques, without any interactions between islands or with the top-level design.

Because clock trees are confined within each island, rather than at a chip level, the capacitive loading is considerably lower. This results in fewer levels of buffering which incurs less insertion delay, lower power dissipation and clocks are easier to balance.

After timing closure an abstracted timing view is created to reduce the size of the design database for top-level integration. This “shell” or “donut” view contains only the timing information that is needed at the top level.

3.3. Top Level Integration

Assembly of the islands at the top level is straightforward. Designs implemented with the IoS methodology have only a small number of short top-level routes. These are ACLs that are skew-tolerant by construction, and scan chain connections that use data hold latches to eliminate any timing dependencies between the islands.

The most critical top-level timing requirement is to control the skew between the data and clock signals within each source synchronous link. Although the absolute latency of the link is not important, the skew within the link determines its maximum operating frequency (see Figure 8). The source synchronous clock is launched at the mid-point of the valid data eye, and the allowable data skew relative to the source synchronous clock is approximately 50% of the clock cycle, less setup and hold time. Some adjustment in the skew tolerance is also needed to compensate for the logic in the source and destination sides of the link.

For the fully asynchronous data links, skew management is less critical. Timing skew within each link is referenced to handshake signals rather than a transmitted clock, and typically needs to be controlled to a multiple of the receiver clock period rather than a fraction of the transmit clock.

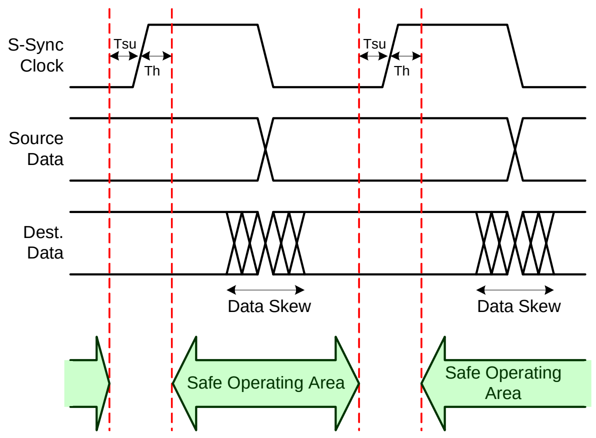

Figure 8. Source Synchronous Link Timing

4. Design Applications

The primary validation vehicle for the IoS methodology has been a 0.13 micron design nicknamed “SoRcery”. This design is approximately 7.5 million logic gate equivalents and has 25 different clock domains. It was implemented in eight Islands of Synchronicity and a separate block for the Clock Generation Unit.

Island partitioning was performed based on functionality, timing and gate count considerations. The islands ranged from 50,000 to 800,000 gates in size and combined between 2 and 14 separate IP blocks. The SoRcery block diagram and floorplan are shown in Figure 9 and Figure 10 respectively.

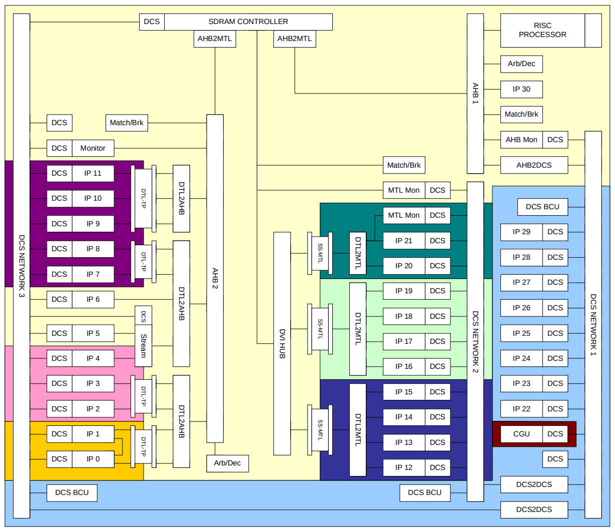

Figure 9. SoRcery Block Diagram

The longest and slowest ACL in this design was a source synchronous MTL link between island 7 and island 3 (the memory controller), passing through island 6. With no optimization at all, this link operated at a frequency of just over 400MHz. Current work on enhancing the on-chip interconnect approach is expected to raise this to 1GHz.

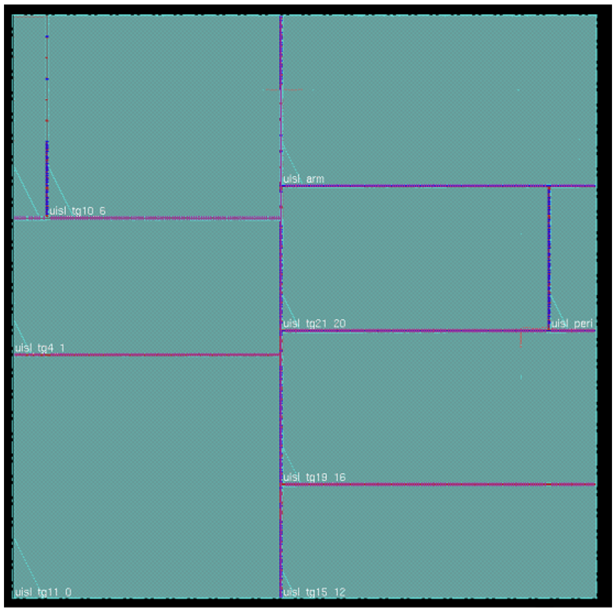

Figure 10. SoRcery Floorplan

Two other Philips products have used various components of the SoR: the Nexperia PNX1500 (connected media processing IC) and the Nexperia PNX8550 (home entertainment engine). Data from these designs and from SoRcery shows that the on-chip interconnect logic adds a gate count overhead of less than 2%.

These designs also confirmed the positive impact of infrastructure regeneration on routing complexity and the ease of top-level timing closure.

5. Summary

Islands of Synchronicity represents an evolutionary approach to SoC design that is tool independent, scales with technology and retains the advantages of contemporary design techniques such as physical synthesis and hierarchical physical design.

It is built upon a solid foundation of IP reuse and IP interconnect standards. Islands are designed in a modular fashion using existing practices with minimal impact on IP development. The approach is fully compatible with current DfT and DfD techniques, and typically makes these easier to apply at the IC level.

IoS introduces design partitioning based on timing and skew-tolerant on-chip communication. The elimination of global clocks significantly eases top-level timing closure, enables hierarchical physical design and reduces time to market for complex SoC designs.

The associated Skeleton of Reuse provides a high level of automation in chip integration by applying central clock generation, bus builders and design manipulation capabilities.

Application of this methodology to multi-million gate advanced technology designs has demonstrated its low area overhead, ease of timing closure, the decoupling of island development from top-level integration, fast iteration of design changes and the ability to create high-performance on-chip interconnect.

IoS is a continually evolving methodology. Work is now in progress to further optimize the on-chip interconnect performance to achieve operation at 1GHz and beyond, and to complete the extension of the methodology to fully support Islands of Power.

6. References

[1] “Reduction of Interconnect Delay by Exploiting Crosstalk“, Van Dijk, S.; Hely, D; ESSCIRC, Sept. 2001. Page 316.

[2] “On-Chip Bus Attributes Specification Version 1”, On-Chip Bus DWG (OCB 1 2.0). September 2001. VSI Alliance WEB site, http://www.vsi.org/.

[3] “A structured and scalable mechanism for test access to embedded reusable cores”, Marinissen, E.J.; Arendsen, R.; Bos, G.; Dingemanse, H.; Lousberg, M.; Wouters, C.; IEEE International Test Conference, 1998. Oct. 1998. Page(s): 284-293.

[4] IEEE P1500 WEB site, http://grouper.ieee.org/groups/1500.

[5] Collett Research International Inc., Cupertino, CA.

[6] “Core-Based Scan Architecture for Silicon Debug”, Vermeulen, B.; Waayers, T.; Goel, S.K.; IEEE International Test Conference, 2002. Oct. 2002. Page (s): 638-647.