

Montego™ Design Philosophy, Methodology, and Verification

The design and development of Bay Microsystems' complex, highly integrated network processor, Montego™, posed special challenges to Bay's engineering staff. Montego is the flagship member of Bay's Internetworking Processor™ (InP) family of network processors, and its success will establish Bay's position in this competitive market.

Designed for the OC192c/10G networking market, Montego combines scalability, intelligence processing, and high performance in a single-chip solution. To meet this level of sophistication, Bay's engineering staff faced the challenge of creating a deterministic, pipelined, super-scalar architecture that represents the state of the art in system-on-chip (SoC) design. With the "Access to Long Haul" scalability of this device and sustained line rate performance without compromise, every module and every pipe was a separate design challenge.

A challenge of this magnitude requires an aggressive but deliberate approach to maximize the probability of success. There was also a need to meet very aggressive market goals and timelines. In the very early stages of the project, Bay's engineering staff adopted a six-pronged approach to meet this challenge:

1. Establish a *design philosophy* that maximizes the probability of success.
2. Establish design *scalability and re-usability* such that the intellectual property set can be quickly adopted and proliferated across multiple products.
3. Identify major *risk factors* and create a plan for risk management.
4. Implement a *design methodology* that leads from initial concept to finished product in an orderly manner.
5. Implement *verification* procedures to ensure, every step of the way, that the product meets specifications.
6. Implement a comprehensive *backend methodology* where critical control can be maintained in house.

This paper discusses this approach and how Bay's engineers implemented it.

Design philosophy

Bay adopted a design philosophy that maximized the probability of success. Key elements of this philosophy were:

- All critical functions must be performed in house—We must control our own destiny
- Team members must have vertical design knowledge and experience
- We will use only advanced, stable, and proven CAD tools
- We will execute check lists and reviews at every stage of the design process

To see how this philosophy contributed to a successful design effort, let's look at each element in more detail.

Keep critical functions in-house -- Control our own destiny

Controlling our own destiny means controlling the whole design and development process, from concept through tape-out. The typical ASIC design flow turns a design over to an ASIC vendor at some point in the design process, thus putting quality and scheduling control in the hands of the ASIC vendor.

To maintain complete control of our own design process, we chose a hybrid ASIC/COT (customer-owned tooling) design flow. The COT flow keeps all functions, from initial concept to tape-out, under Bay's control. A tightly coupled development partnership was established between Bay and its foundry. This partnership enabled Bay to leverage the foundry's expertise and resources for circuit design, place-and-route and layout, thereby augmenting Bay's own expertise and resources in the same areas. However, a development partnership of this nature placed additional demands upon the Bay team. Specifically, Bay was required to appreciate the depth of design capability, design style and working style of the foundry's resources. In addition, Bay had to develop a deep understanding of the foundry's process technology.

Consummation of the foundry relationship allowed all critical functions including circuit design, floorplanning, place-and-route, and test development, in addition to front-end logic development to remain in-house at Bay.

Vertical design knowledge

Designers cannot work in a vacuum. Every designer must understand and be able to implement every phase of the design process. Bay's Montego design team used a multi-tiered design methodology that spanned many disciplines.

Some designers worked primarily in different areas during various stages, such as RTL descriptions or gate-level implementation, but every designer was expected to be familiar with all other phases in the design process. There are no artificial walls separating the various design phases.

Advanced, stable design tools

The Montego design team needed advanced design tools at its disposal—but not so advanced that they were not yet silicon-proven. This relatively small design team worked on a tight development schedule, and needed to meticulously follow the design process. The designers could not afford to spend time waiting for bug fixes in the latest offerings from the EDA industry, no matter how many new features the new tool claimed.

Every CAD tool that Bay Microsystems used had previously proven itself in successful design projects. The Bay design team pooled its decades of experience in high-performance design projects and created a design flow that was supported by advanced but proven CAD tools.

Execute check lists and reviews

Planning and execution were continuing processes throughout the design cycle, and regular design reviews were a part of those processes. Design reviews helped identify problem areas and allow engineering management to formulate plans to correct them.

Scalability and re-usability

Bay's management demanded that Montego's underlying architecture be scalable. In addition, the design's intellectual property set must be reusable.

Bay's management team envisioned Montego as the initial product within a family of devices that would span the network processing marketplace vertically with regards to performance and horizontally with regards to functional solutions served. As such, Montego had to achieve both scalability of performance upward and cost effectiveness downward. In future family members, reuse of internal structures would be leveraged to expedite new product releases.

During the architecture specification phase, Bay's engineering team considered both current and next generation capabilities as they relate to process technology, external memory technology and packaging technology. Their findings were used in determining all of the internal structures of the Montego architecture, including the superscalar pipeline structure itself. The effect was to empower next generation products to be easily designed with nominal processor frequency speed up and corresponding widening of memory buses.

Moreover, each functional module within the architecture was partitioned with re-use in mind. A functional module could then be extracted and used as the basis for a module within a future project or it could be expanded and produced as a stand-alone product to address various market segments. The concepts of scalability and re-usability are all-encompassing issues as they directly affect the micro architecture definition, logic implementation, physical implementation and verification processes.

Risk factors

Bay's technical staff identified six significant risk factors that could produce significant downstream problems:

1. Montego's design includes over a hundred memory macros, some of them quite large.
2. Montego's complex, high-speed, wide-bus I/O architecture supports various standards.
3. The new device would be packaged in a 1600-pin flip-chip BGA package.
4. Place-and-route of the finished design could be very complex, particularly since it must support multiple power supplies and over 2000 flip-chip bumps.
5. Montego's complex functionality and rich feature set presents a significant verification challenge.
6. Complexities associated with system level deployment must be anticipated and addressed as part of the overall design from the start.

Such an aggressive design requires equally aggressive risk management techniques. Engineers often implement designs in FPGAs to test functionality, but for the Montego design it was clear that the FPGA approach would not provide sufficient feedback. Bay Microsystems saw that the only way to adequately validate these risk areas was to actually fabricate test chips and evaluate the results. Bay worked with its foundry partner to produce four test chips to address the first four risk areas:

1. I/O test chip—verify I/O design, performance, ESD, and latch-up.

2. Embedded memories test chip—verify yield and performance.
3. Dummy 1600-pin package before Montego tape-out—verify assemblies.
4. Montego reduced feature test chip—verify various interface protocols, memory architecture, design flow and process performance.

Characterization of the test chip vehicles identified potential pitfalls that were quickly alleviated prior to tape out of the Montego device. The test vehicles also provided verification “proof in silicon” that various design techniques implemented to address high-risk and other areas were successful. However, test vehicles alone are not adequate to fully verify a design of this complexity. The following describes the complete methodology applied by the Bay team.

Design Methodology

Bay’s design methodology follows the guidance of the design philosophy. The *Montego Design and Verification Flow* diagram that follows shows the sequence of steps in the design flow, along with an overview of verification steps and software tools. This section discusses major elements in Bay’s methodology in more detail.

Specification stage

In Montego’s specification stage, Bay’s corporate management, marketing management, and engineering management defined the product’s functionality and performance. The final product specification, therefore, included management direction, marketing requirements, and engineering input. It was formalized in the Architectural Specification.

- **Architectural Specification.** The architectural specification defines Montego’s functions and performance targets, but does *not* define how these targets will be achieved. That came later.
- **High-level C reference model.** The high-level, data-accurate model, written in C, defines Montego’s input and output functions, but does not define timing or internal structure. This model is used to simulate Montego’s functionality in application environments.
- **Micro Architecture Specification.** The micro architecture specification is a highly detailed document that adds hardware structure to the Architectural Specification. It defines registers and pipes. It allows first order performance analysis to ensure that Montego meets its goals. It identifies risk areas for timing and critical areas, and for corner cases with respect to performance.
- **Cycle-based, cycle-accurate C reference model.** A more detailed Montego model is the cycle-based, cycle-accurate C reference model. This model, also written in C, defines Montego’s cycle-by-cycle functionality. This C reference model performed a number of functions in the design cycle:
 - Verified Montego’s performance
 - Verified internal pipe structure
 - Generated test cases for module/chip functional verification

This model's cycle-accurate performance is identical to the Montego chip in a simulation environment, therefore it can be used by potential customers to integrate Montego in their system environment and to verify its functionality in early design stages.

The creation of these models will usually identify problem areas in the original architectural specification that need modification. In Montego's case, those areas were modified and the changes propagated down through the cycle-based C reference model and the micro architecture. In Bay's design methodology, the Architectural Specification and the elements derived from it must be in sync at all times, and no other design steps may be performed until these elements are completed and reviewed. Montego's RTL development started only after the Micro Architecture Specification was completed, reviewed, and approved.

System Feasibility – can it be deployed?

Two major risk factors that cannot be overlooked when developing a complex VLSI device is system deployment and software programmability. Falling into the trap of a 'pins-in' mentality can lead to very discouraging results once a customer attempts to implement a device into a system. Bay addressed these issues as part of our overall design methodology.

To address software programmability we set a goal that our Software API, known as NEXtware™ had to be fully functional and fully tested prior to receipt of silicon. We were able to accomplish this task via utilization of the cycle accurate C-Reference Model. The model was designed to operate exactly as the resulting silicon. Therefore, our software team could develop the APIs directly on the model. This enabled the software design team to complete the API development in advance of the silicon and also test the API as well. The robustness of this approach allowed the Bay team to port the API to the actual silicon in a matter of hours.

In addition, Bay management regarded system level deployment of Montego as a critical requirement. To address this requirement Bay assembled a complete system engineering team. The system team was tasked with developing a full system design based on Montego. The system, known as the Internetworking Development System (IDS), would later be provided to customers as a complete reference platform, emulator and software development environment. Acting as our first Alpha customer well in advance of Montego's completion, the system team was immediately set to task to determine whether Montego was deployable.

The system design team analyzed Montego based on how it would influence an overall system platform architecture. They also analyzed how the physical characteristics of the device would affect a system. From a system architecture point of view, interplay of the fast path, control plane, and forwarding information base (FIB) was examined. In addition, issues surrounding component interoperability, statistics collection, and diagnostics were examined. As for physical deployment the system design team looked at: high-speed busing/interconnect along with associated tuning, power requirements, mechanical layout, thermal dissipation, environmental constraints, EMI, and overall operating characteristics. All of these items were fed back to the VLSI design team in order to insure that they were appropriately handled within the final Montego design.

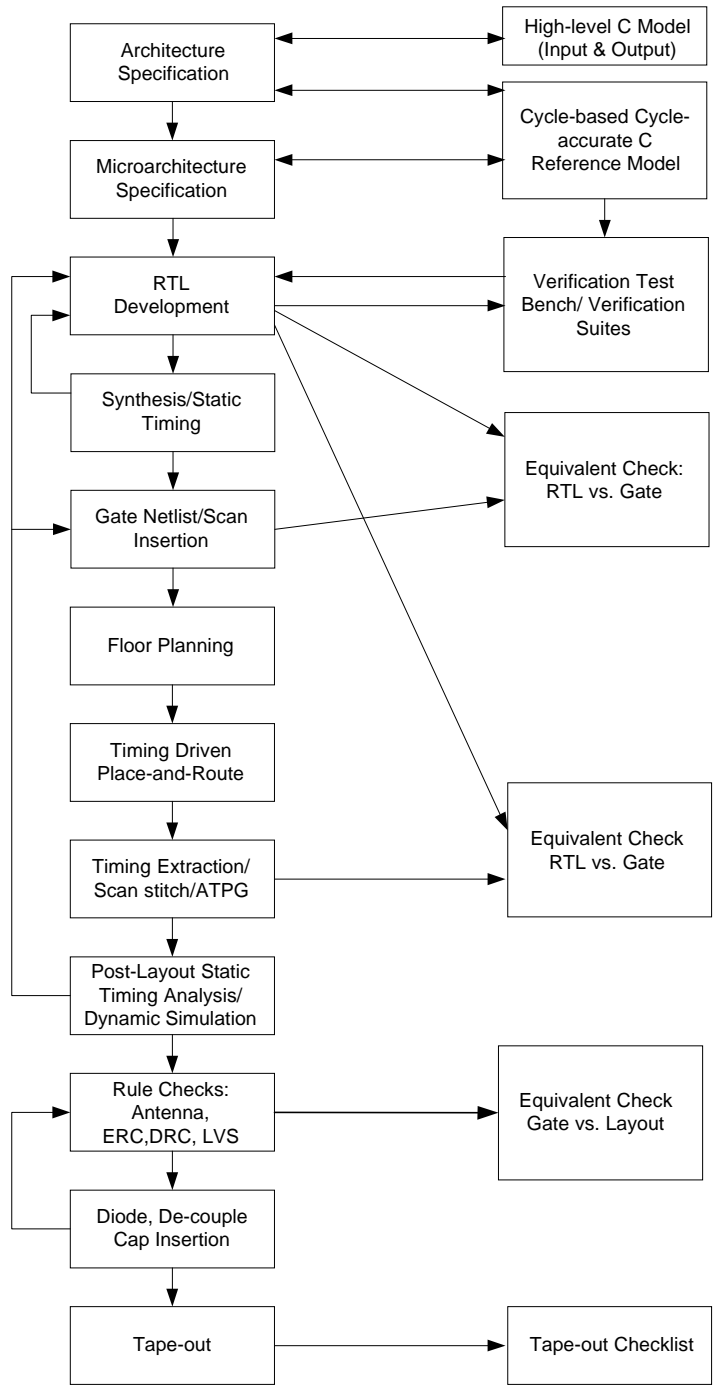
VLSI design flow

The diagram on the next page identifies the remaining steps in the design flow.

Note that this design flow carries the design through tape-out, with all functions being controlled by the Bay design team.

Montego Design and Verification Flow

Design Process



Bay's design methodology begins with a formal Architectural Specification and ends with tape-out of the finished design. Bay's engineers control every step of the process.

Design Verification

One can easily argue that, following the original specification, design verification is the most important phase of the design process. For a design as complex as Montego, that argument is particularly valid. Verification took up approximately sixty percent of the Montego design effort.

Verification is a rigorous process, because it is a well-known truth that the earlier an error is found in a design, the easier it is to correct. Conversely, if an error is not detected early, it can destroy months of design effort, and possibly kill the entire design project. Rigorous verification is the only way to ensure early detection.

In the Montego design flow, verification assumed two forms: functional and formal. The simple definitions of these two forms are:

- *Functional* verification ensures that the RTL representation of the design correctly implements the architectural specification.
- *Formal* verification ensures that the gate and transistor-level implementations of the design are exactly equivalent to the RTL code.

Functional verification

The Specification stage of the Montego design methodology resulted in an Architectural Specification, a high-level C Reference model, a cycle-based/cycle-accurate C Reference model, and a Micro Architecture Specification. Of these elements, the cycle-based/cycle-accurate C Reference model exactly matches the functionality of the target chip. Designers began creating an RTL model of the Montego design following the approval of the specifications and models.

Since the cycle-based C Reference model exactly defines Montego's cycle-by-cycle functionality, Bay's functional verification procedures compared the RTL code with the C Reference model to ensure that they produced identical results.

A Verilog simulation tool was used to simulate the RTL model. To adequately determine functionality, the Montego designers had to simulate over 200 Million Montego cycles, the equivalent of one second of 10Gb/s data. Simulation time, therefore, was a bottleneck. The following table illustrates the problem.

Simulation Mode	Simulation Cycles/Sec	Time to Simulate 200 Million cycles
Cycle-based C Reference Model	10K	5.6 hours
Verilog-Module	300	185 hours (~7.7 days)
Verilog-Chip	60	925 hours (~38 days)
Tharas Accel-Chip	600	92.5 hours (~4 days)

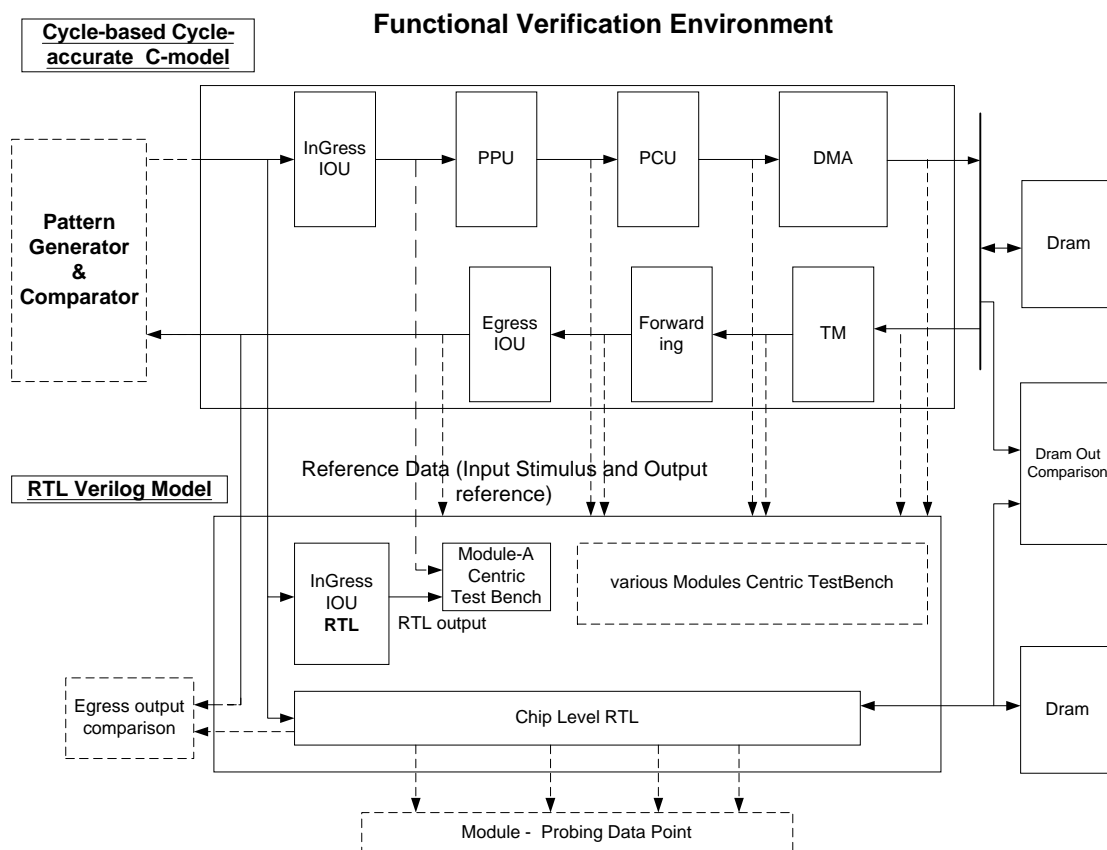
Simulation time is a bottleneck in functional verification. The C Reference Model executes much faster than the VCS simulator in either module or chip simulation mode. A

Tharas software accelerator in chip mode executes an order of magnitude faster than the VCS simulator alone, but is still far slower than the C Reference Model.

The table compares the number of Montego operating cycles that various simulation modes can simulate in one second. The C Reference Model can simulate 10,000 Montego cycles in one second, while the Verilog simulator, simulating the full chip, can simulate 60 Montego cycles in one second. Therefore, it takes the Verilog simulator 38 days to simulate one second of 10Gb/s data stream. Bay used a Tharas software accelerator to bring simulation times down to a reasonable level—days rather than weeks of simulation time.

Most functional simulation, however, was done in a module-by-module mode. Each RTL module was compared to its C Reference model equivalent, as shown in the Functional Verification Environment diagram. Simulating a single module was five times faster than simulating the whole chip. When all modules indicate correct functionality, they were integrated and then simulated as a complete chip in the Verilog simulator or Tharas.

Bay used the Verilog simulator to verify various chip functions with short and random patterns. The Tharas software accelerator was used for long patterns with real traffic.

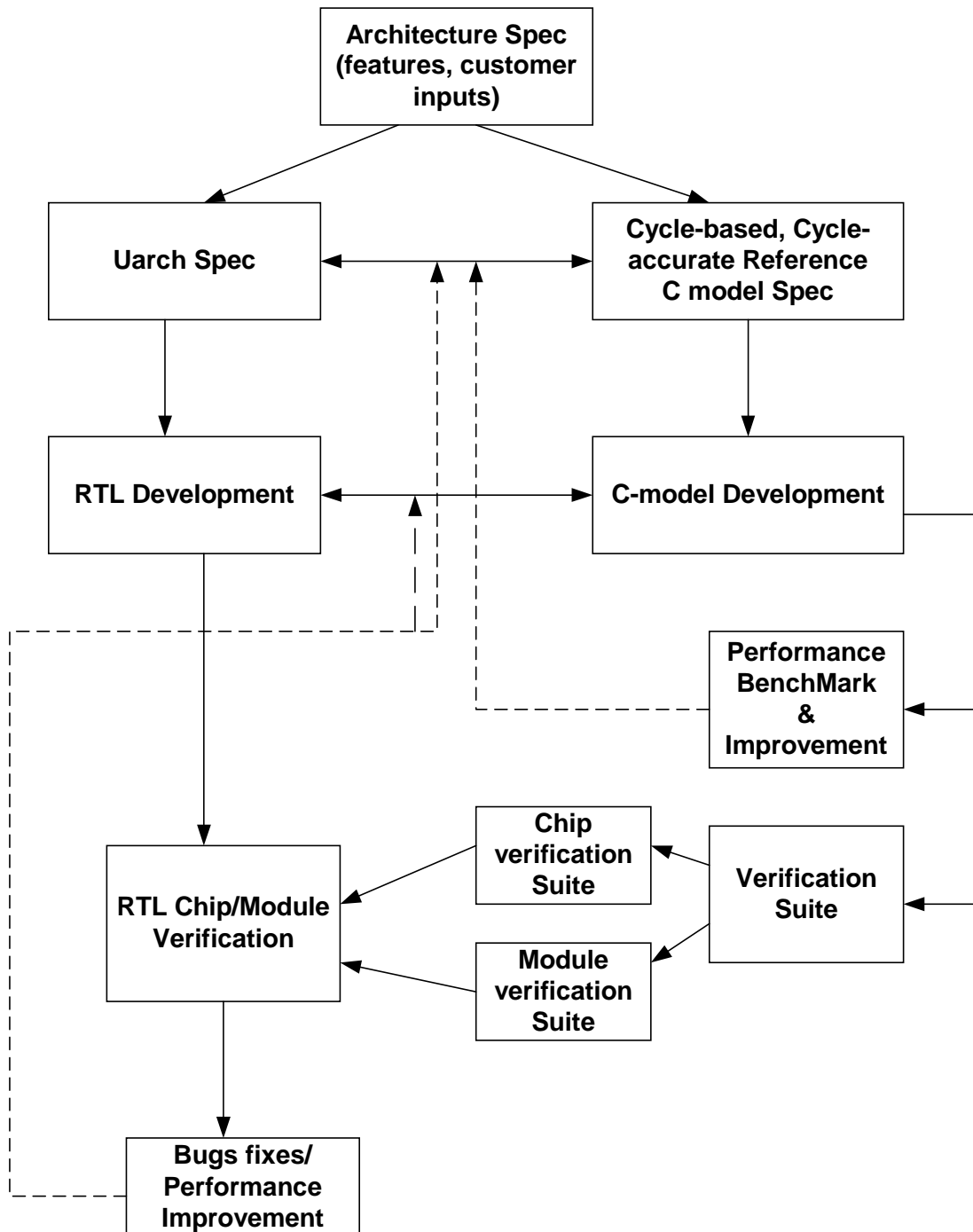


The Montego functional verification environment compares the RTL Verilog Model with the Montego cycle-based C Reference Model. One RTL module at a time is verified in this manner, then all modules are integrated.

The following flow chart shows the verification flow.

Ensuring that the RTL model is identical to the C Reference module is a critical step in the design flow, because all formal verification steps that follow use the RTL model as the master reference.

Montego Functional Verification Methodology



Formal verification

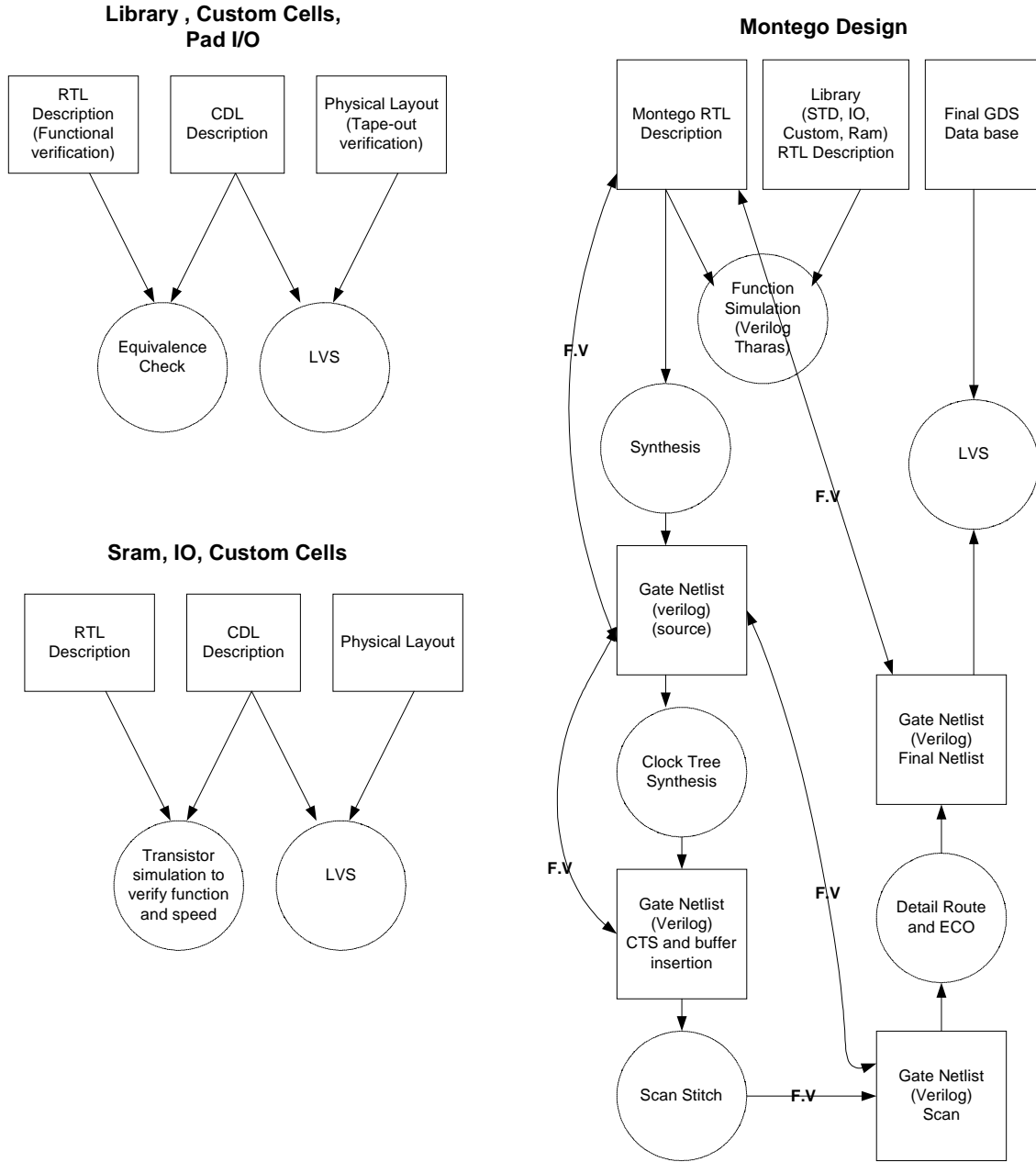
Once the Montego RTL model was proven to be correct, subsequent design steps converted the RTL code into transistors on a chip. The Design Flow diagram earlier in this paper shows the steps that lead from RTL code to tape-out. Formal verification ensured that each step in the process maintained the RTL code functionality.

Formal verification steps include:

- *Gate vs. RTL.* A gate-level model of the Montego design was synthesized from the RTL code, but this did not guarantee equivalence. A gate-to-RTL equivalence check ensured that the gate-level model was equivalent to the RTL model.
- *Transistor vs. RTL.* Equivalence checking between the transistor-level chip layout and the RTL model ensured that the layout accurately reflected RTL functionality. This area applied in Memory macros, IO pads, and special custom cells.
- *Transistor vs. Gate.* Equivalence checks between the transistor layout and gate-level model ensured that functionality was maintained.

Rigorous formal verification ensured that the proven Montego RTL design was carried through gate-level implementation, to layout, and finally to the GDSII file from which the Montego chip was fabricated.

RTL, Gate, and Layout Formal Verification Flow



Backend Methodology

In traditional ASIC methodologies, the ASIC vendor performs place-and-route functions using the vendor's generic methodology at the ASIC vendor's facility. The ASIC vendor generally prefers this methodology because specialized tools and specialized engineering training is required. Montego demanded more than an ASIC vendor's routine performance. Every critical function had to be customized to meet Bay's higher standards.

Bay, therefore, chose a hybrid ASIC/COT (customer-owned tooling) design flow. The COT flow differs dramatically from the traditional ASIC approach in that it keeps all functions, from initial concept to tape-out, under the design team's control. Bay's foundry assisted Bay by providing tools and place-and-route specialists to work at Bay's facility, co-working with Bay's engineering staff. The specialists effectively became part of Bay's Montego design team, working closely with Bay's own in-house experts.

The tool chain for the backend is included in the *Montego Design and Verification Flow* diagram.

Results

A meticulous design philosophy and design methodology coupled with rigorous verification gives an engineering team an excellent chance of producing a design that meets a company's goals. Shortcuts, on the other hand, invite downstream disasters. When the Montego team taped out the finished design and delivered it to our silicon vendor, it did so with a high level of confidence that the silicon would work.

The first silicon received at Bay following tape-out proved the value of this methodology. Bay Microsystems first silicon works, and is of a quality level that can be used by customers to verify Montego's functionality in their systems.