Coherency: The New Normal in SoCs

by Don Dingee

Beyond today's smartphone lies new devices that will test the boundaries of human experience. Compelling new applications include augmented and virtual reality, connected cars, driver assistance and autonomous vehicle systems, Internet of Things gateways and infrastructure, embedded vision, deep learning, and more. All of these applications will depend on innovative system-on-chips (SOCs) designed to deliver more performance in less power.

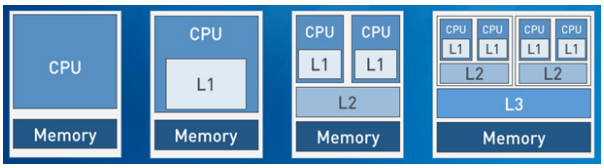
Performance is no longer simply about raw processing. Devices will still compute, but they will also handle display, networking, and storage requirements which are exploding rapidly. We are not far from devices each handling 100 teraflops of compute, billions of pixels of display, hundreds of gigabits of connectivity, and terabytes of storage. Compared with current state-of-the-art mobile SoCs, these are increases of one or two orders of magnitude – at similar or preferably lower power consumption.

SoC design is changing to meet this challenge. Multicore architecture is already the norm in mobile SoCs, a trend accelerating as more designs incorporate the approach. A determining factor in how well these cores work together is cache coherency. We now look at how cores, caching, and coherency have evolved, why existing design practices often run into trouble at scale, and how automation can help even the most experienced SoC design teams handle these new, rich, complex usage models.

Shifting the Burden of Coherency

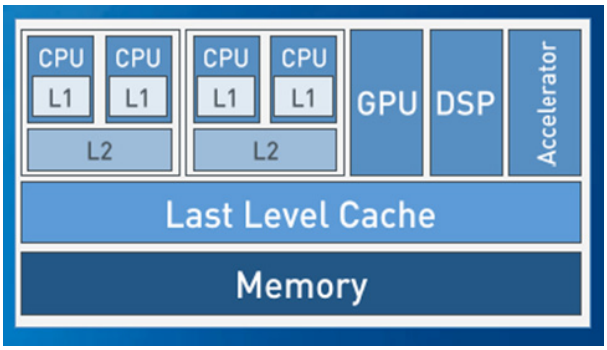
Cache developed in a processor-centric context, a way to keep central processing unit (CPU) pipelines filled with data and instructions faster than possible with relatively slow DRAM. In single processor systems, the only major concern around cache was a read miss causing a lengthy pause while items were reloaded from memory. More sophisticated fetching and replacement algorithms developed to improve cache hit rates and raise performance as CPU execution pipelines deepened.

Multiprocessor systems with shared memory emerged to support multiuser, multitasking operating systems. Demands on I/O subsystems increased; intelligent peripherals began taking advantage of direct memory access (DMA) to reduce CPU time spent simply moving data. For the first time, issues of coherency developed. With CPUs and peripherals each modifying objects in shared memory other CPUs had cached, hit rates plummeted. In response, layers of cache appeared, with a smaller, faster L1 closely coupled to each CPU, and a larger L2 shared between CPUs. Further steps to help performance included marking some memory areas as non-cacheable, while dedicating other regions of memory to a specific CPU, reducing how often expensive cache misses occurred.



System scalability was at first a function of adding more CPUs, improving cache and memory, and moving to a multitasking software model. More processor cores drove clustering and another layer of L3 cache. Soon, the limiting factor became I/O, especially as the graphical user interface and high-performance disk storage became the norm. Peripheral chips with their own embedded cores debuted, changing the role of intelligent peripherals from simply gathering data to processing it. The transition from homogenous to heterogeneous processing was underway.

Improvements in embedded core technology and semiconductor processes combined with demand for smaller yet powerful mobile devices drove system implementation from a single board computer to a system-on-chip (SoC). These SoCs combined heterogeneous processing elements including CPUs, graphics processing unit (GPU), digital signal processing (DSP), multimedia accelerator cores, and more on a single part. Memory, however, was still external to the SoC.



In heterogeneous SoCs, the limiting factors became the efficiency of the interconnect between elements – processing and I/O – and the available memory bandwidth. Multimedia use cases placed more stress on architectures. Simply adding more cores might seem like a solution, but care must be taken with respect to power consumption and interconnect. Memory subsystems serve demands from all the compute engines, so revisiting caching strategies and cache coherency became a priority.

CPU-centric cache coherency strategies no longer make sense in this modern heterogeneous SoC environment. A simplified SoC is pictured, and it only hints at the problem. High-end SoCs now have as many as 150 different intellectual property (IP) blocks, interacting in a complex dance trying to process multimedia data quickly with the fewest resources possible. These IP blocks often behave very differently from traditional CPU cores and place different demands on interconnect and caching. In many cases, the CPU cores are oblivious to other transactions occurring in the SoC. The burden of cache coherency has shifted from the CPU cores

into the heterogeneous SoC interconnect, with each heterogeneous IP core an agent in system-centric cache coherency services.

Pitfalls of "Divide and Conquer"

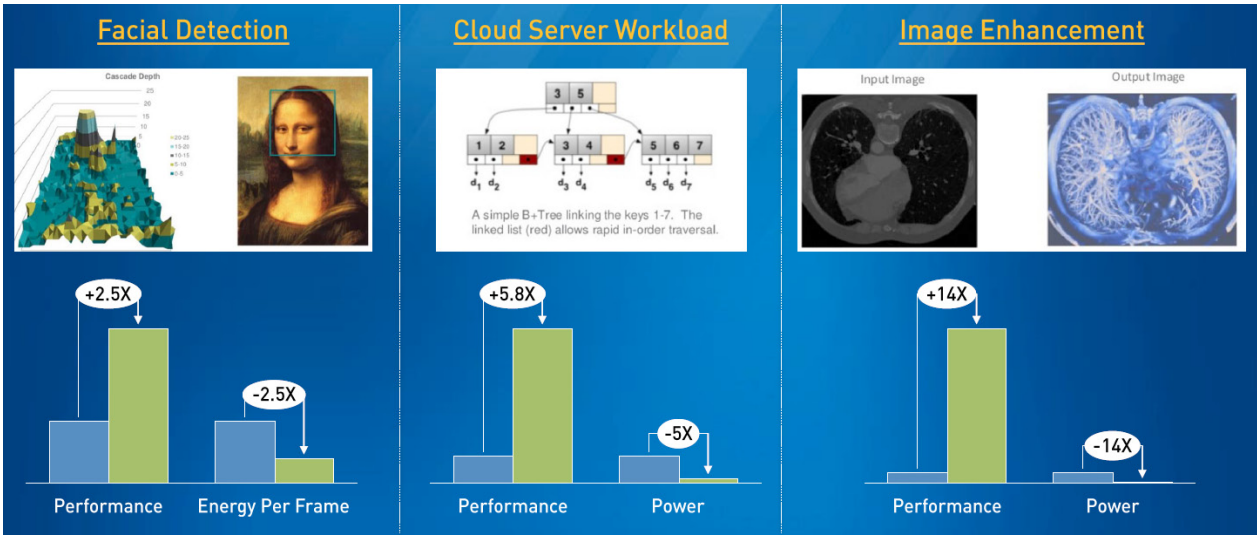
The science of caching and coherency needs to evolve to handle this shift. In the traditional CPU-centric approach to caching, homogeneity served designers well. Caches were patterned identically between equally configured CPUs. Coherency was a matter of informing the other CPUs, with the same knowledge of caching algorithms and timing, of a change that may affect their cached elements.

Once the CPU cluster was optimized, the natural thought was to optimize every other IP block in the SoC for the best power/performance/area (PPA), then integrate all of them over what was presumably a high-speed interconnect. This "divide and conquer" strategy works when the IP block count and core variety is relatively low and the design is still CPU-centric. For instance, a GPU core handling only display-oriented tasks can be handled as a special case.

Complex use cases and a dramatic increase in heterogeneity broke those nice, clean design techniques. SoCs are not only taking on advanced network transports and data rates, they are moving into augmented and virtual reality, automotive applications with vision, radar, and lidar, and gesture and voice interfaces for consumer electronics. Performance demands are doubling and tripling or more, within the same or lower power consumption expectations.

Industry data from the HSA Foundation and other sources shows heterogeneous processing provides a PPA advantage, often with both higher performance and lower power consumption. Tasks such as facial detection can be handled by GPU cores, DSP cores, convolutional neural networks (CNNs), or other dedicated hardware. Security accelerator cores handle encryption in hardware. Always-on voice processing cores wait for inputs while the rest of the SoC is in a power save mode. Each different core presents optimization choices in caching and coherency.

Who makes those optimization choices, and how? Without a system-centric design



How do you multiply performance 5x, 10x, 100x while reducing power requirements? Heterogeneous architectures combine different specialized compute engines —CPU, GPU, DSP, and accelerators— in an SoC. In the use cases above, heteogeneous architectures, shown in green, show striking improvments in performance and power compared to equivalent multi-core platforms, shown in blue. In the case of facial detection, after normalizing results for the same energy per frame, performance is increased ~6x (2.5 x 2.5). The cloud server example shows almost 30x performance, while image enhancement shows almost 200x.

perspective, optimizing individual IP blocks presents a conundrum in quality of service (QoS). An optimized block can consume system resources and starve other blocks. Optimizing blocks tested alone can prove disappointing when those blocks are integrated into an SoC, yielding performance far less than expected. Characterizing complex subsystem interactions is difficult even for highly experienced architects, and tuning performance manually is time-consuming at best and ineffective at worst.

Problems worsen around third-party IP blocks. Gaining a thorough understanding of how those blocks have been designed and optimized takes valuable time. When integrated into a system, functionality may be perfect, but optimization choices made by those IP block designers may prove to be incorrect for the use case or interactions in the end system. Caching and coherency needs are particularly difficult to anticipate in a stand-alone context.

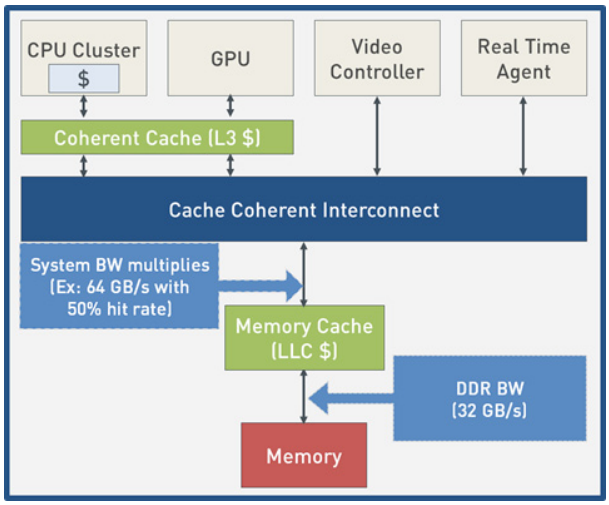
Architects often try to enforce rules to manage the problems. Coherent and non-coherent networks are segregated. Coherent networks are hammered into the same kinds of constructs used for CPU-centric design, with tiled structures and regular connection patterns. Non-coherent networks are relegated to second-class SoC citizens, in the hope that use cases never develop requiring a core on the non-coherent side to participate with a peer on the coherent side.

"Divide and conquer" design practices often result in heterogeneous cores in an SoC that compute, flush, compute, flush, compute, and flush, ad infinitum. QoS approaches are often an afterthought, patched together and dropped on top of an SoC infrastructure to meter traffic. Functional safety is an increasing concern for designers, and again is difficult to add-on after integration of IP blocks.

System-Centric Design Takes Over

Turning high-powered system architects and a team of designers loose on a large heterogeneous SoC is no guarantee of success. Smaller, less experienced teams stand a greater chance of running into problems, even on what would be considered a mid-size SoC design. With a new crop of SoC design starts nearing, how should teams change their approach to shift the odds in favor of success?

Heterogeneous architectures are here to stay. They provide a significant PPA advantage, mainly through precise targeting of workloads and employing hardware acceleration IP blocks dedicated to and optimized for specific tasks. The differentiator for most SoC designs will not be in the functional IP blocks themselves, especially with third-party IP and reuse rising. Rather, differentiation will be found in the system-centric interconnect and how it handles caching and coherency.



The first major opportunity in system-centric SoC design is scalability. In a requirements-driven architectural approach, functionality is quickly decomposed into a collection of IP blocks. Today, most of those blocks interface via a standardized interconnect. AMBA® (Advanced Microcontroller Bus Architecture) was created by ARM as a way to ease system-level integration – and it works well for small- to mid-range CPU-centric designs. Many non-ARM IP block providers also adopted AMBA as its popularity grew. Design practices and EDA tools took shape around AMBA, and evolutions to the specification helped increase performance.

For mid- to high-end heterogeneous SoC designs, where almost every IP block talks to everything,

caching and interconnect issues begin to bound performance. AMBA is still used as the low-layer protocol, but higher system-centric thinking is needed to achieve the desired performance. Architects first tried crossbar switches, but found they quickly were overwhelmed as well. Modern approaches now use network-on-chip (NoC) as the interconnect framework, with concepts of packet switching, dynamic priority, and virtual channels to achieve better interconnect bandwidth and latency.

NoC implementations addressed interconnect performance except for one area: memory. Without improvements in caching, memory latency and bandwidth suffer as ultimately, most of the transactions in a heterogeneous SoC are headed for the memory controller. Robust cache configuration eases the load on system memory.

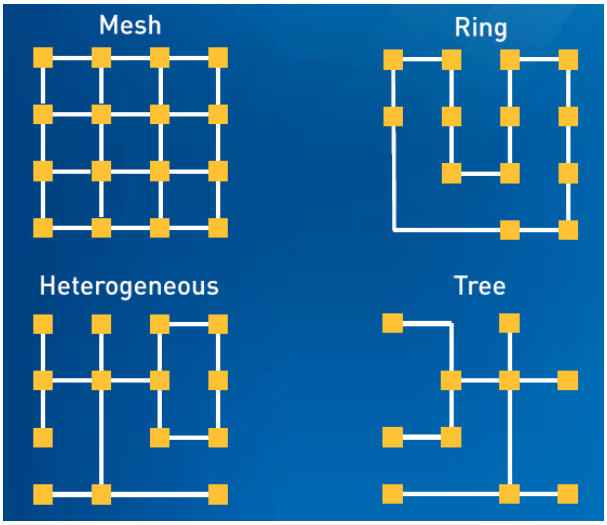
In a system-centric design concept with "end-to-end" coherency, both cache-coherent and non-cache-coherent agents are embraced. Two types of caching are employed to solve two different problems:

- Coherent cache reduces system latency. The focus is to treat every compute core – CPU, GPU, DSP, hardware accelerator – as a peer with equal access to all caches and update policies. Broadcast and multicast support, derived from an underlying routing mechanism inspired by large-scale computer networks, provide updates for all cache-coherent agents.
- Last level cache, or LLC, sits directly in front of memory and effectively doubles DRAM bandwidth. LLC can serve any agent, including ones not in the coherency domain. A configurable design allows for cache, scratchpad, and RAM modes, with pre-allocation policies for better system performance.

Augmenting Architectural Intuition

Scalable features address end-to-end coherency for the heterogeneous SoC environment. Next come choices in the interconnect architecture.

How does a team know that it has created and implemented the best possible cache-coherent interconnect architecture for the job? This raises the second major opportunity in system-centric SoC design: automation.



The idea is not to replace SoC architects or their years of experience, but rather to augment them. Two tendencies get in the way of the best intuition of architects and designers when implementing large heterogeneous SoCs. First is the effect discussed earlier: sub-optimization of IP blocks may inadvertently create blocks that work against each other in delivering performance for system-centric use cases. Most teams don't figure that out until integration, and then have to somehow redo or undo optimizations or perform incremental tweaks hoping to sort the situation out.

The other tendency is to overdesign based on limited data. A working assumption is that each functional IP block is already validated and performs as expected, so any limitations must be in the interconnect. Simulation points to issues, so routers, links, and clocks are overdesigned to remove bottlenecks – but at what cost? The interconnect may deliver the needed performance, but may also consume more power and area than needed to accomplish that goal. Optimizing interconnects is a bigger challenge.

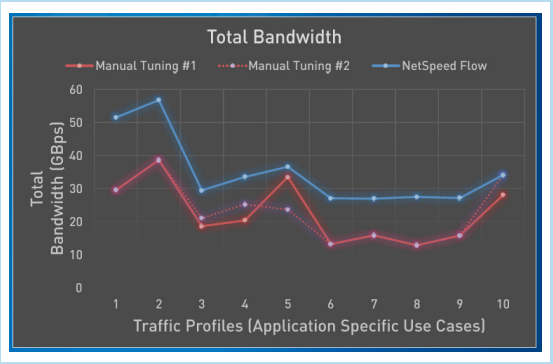
A system-centric design approach should not enforce interconnect topology. As the old saying goes, allowing any topology to be chosen as long as it is a ring may simplify life for the NoC vendor, but it may not solve the problem in some heterogeneous SoC use cases. By applying machine learning techniques along with advanced computer-network router constructs to create auto-generated links, interconnect results can be vastly improved over "hard" tiled solutions.

To determine which interconnect topology is best, most architects would naturally turn to simulation, as have many NoC vendors. Without cycle-accurate models for all IP blocks,

How machine learning helps interconnect design

Experienced SoC architects might bristle at the notion that algorithms can do a better job of creating an optimized interconnect for a heterogeneous SoC. For its NoC implementations, NetSpeed Systems uses a combination of advanced machine learning techniques including Random Forest and TensorFlow to help in rapid exploration of multiple use cases. The result is implementations proven to outperform manual tuning across multiple use cases. For the effort to manually tune a handful of scenarios, automation can deliver more optimized scenarios more quickly than even skilled human architects.

When designing an SoC, the conventional approach is to architect the system with its IP blocks and interconnect and simulate it later in the development process when most of the design decisions have been made. In contrast, NetSpeed's machine learning capability explores a multitude of models rapidly and provides the system architect with various architecture options and accurate performance predictions early in development before integration begins. NetSpeed's platform models SoC behavior in application-specific use cases, looking at the system as a whole to optimize performance



and power efficiency. This avoids the pitfall of optimizing individual subsystems in isolation and then integrating them, which tends to create SoC bottlenecks and overdesigned links to handle worst-case conditions.

A side-by-side comparison shows the superior performance of an SoC created with NetSpeed's tools versus experienced architects using a typical tool chain which entails considerable manual tuning. While the architects do well in a few use cases, NetSpeed automation does equally well in those, and automation does better in many others that presumably got less attention from the architects. It is not a question of competence, but of speed and the ability to explore many system-centric variables and use cases simultaneously through machine learning.

NetSpeed's machine learning algorithms are built into its advanced Gemini NoC platform, with the industry's first and only interconnect synthesis engine to optimize PPA in a system-centric approach for heterogeneous SoC design.

simulation may not even uncover the real issues – and it says little about power or performance margin. Asking a team to develop all those SystemC models can take weeks or months, and in black-box IP cases may not even be possible.

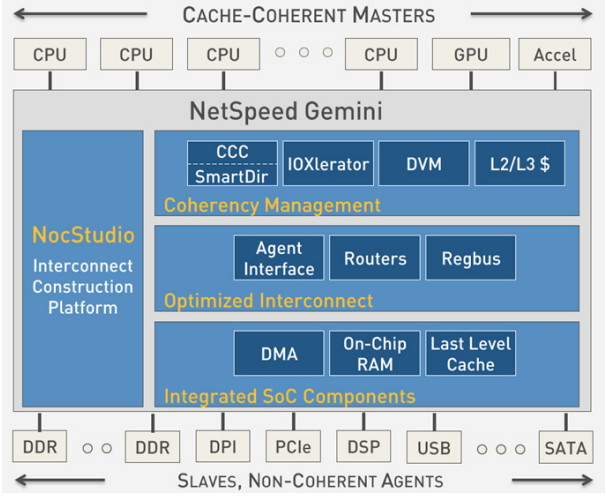
Machine learning provides a better alternative, where algorithms explore patterns in data and iterate solutions quickly. A requirements-driven design flow specifies the IP blocks and their basic connectivity, performance requirements, and system-centric use cases. Instead of architecting the interconnect up front using a best-guess approach, automated architectural exploration uses machine learning techniques to find optimum solutions in power, performance, and area. Each use case is evaluated, and a NoC implementation is suggested. Architects can take a look at those automatically generated results and fine tune their configuration accordingly.

An automated approach helps both experienced and less-experienced SoC design teams. For experienced teams, it allows architects to spend more energy on critical parts of the functional IP while worrying less about interconnects, and enables even larger designs to be tackled with success. For less-experienced teams, automation can save enormous amounts of time and effort otherwise wasted in what can become numerous iterations to fix unanticipated system-centric problems.

Next-Gen Coherent Interconnect

Since first introducing the Gemini 1.0 coherent interconnect solution at the Linley Processor Conference in 2013, NetSpeed Systems has been working with customers worldwide to refine the system-centric approach and prove out its machine learning and interconnect synthesis techniques. The latest incarnation, Gemini 3.0, debuts with enhancements in configurability, routing, and performance.

Unparalleled configurability: Gemini 3.0 offers unparalleled configurability, allowing users to customize every component of the interconnect from interfacing to routers to topology. In addition to providing flexibility in the of number of ports, coherency participation, and



other features, architects can define the cache hierarchy and asymmetric traffic patterns based on system characteristics such as hit rates.

Physically-aware latency optimized design: Gemini 3.0 design tools are physically aware of the layout of the IP blocks in an SoC, producing a customized interconnect topology. This awareness ensures that wiring congestion does not occur late in the design cycle. Appropriate numbers of buffers and pipeline stages in various fabric channels enable smooth backend design. IP components can be partitioned, distributed, and co-located near coherent masters using them, improving latency and power efficiency.

Machine learning-based interconnect construction: Gemini 3.0 is the only SoC interconnect solution that uses machine learning to accurately model the system as a whole to achieve the best application performance. In contrast, conventional approaches tend to optimize individual subsystems in isolation, which can result in bottlenecks and systems that are overdesigned to handle worst case conditions. Gemini 3.0 uses advanced networking algorithms to rapidly create a cache-coherent SoC interconnect that is proven deadlock-free and delivers QoS for all use cases.

Scalable solution: Gemini 3.0 handles cache-coherent, I/O-coherent, and non-coherent traffic in a single SoC interconnect design platform. It supports up to 64 fully cache-coherent

> ***Gemini 3.0 is the only SoC interconnect solution that uses machine learning to accurately model the system as a whole to achieve the best application performance.***

CPU clusters, GPU cores, and other coherent compute blocks, and up a total of 200 I/O coherent and non-coherent agents. Gemini 3.0 is the first and only solution to support both the AMBA 5 CHI (Coherent Hub Interface) and AXI® Coherency Extensions (ACE®) on-chip interconnect standards in a single design, and includes support for broadcast and multicast which can dramatically improve performance.

Other additions to Gemini 3.0 include system-level optimizations:

- An integrated DMA engine with programmable pre-fetch functionality,
- A configurable register bus layer providing for fail-safe system debug,
- Enhanced Pegasus last level cache (LLC) usable as either coherent or memory cache, with scratchpad and RAM modes and programmable pre-allocation policies,
- Architected-in functional safety features for error detection/correction and fault resilience.

Coherency without Compromises

Creating a new wave of heterogeneous SoCs will draw in more designers of both IP blocks and complete chips, as well as more software created for them. Use cases will continue to drive system-centric design principles and provide a path to differentiation for those who understand and apply them wisely. Achieving required performance under all loading conditions in less area and power will be an important metric for SoCs, as will hitting time-to-revenue windows.

NetSpeed Systems has approached cache coherent interconnect from the ground up with a novel machine learning technique for optimization without compromises. It helps cut design time with up front, predictable system-centric modeling, and avoids the pitfalls of a typical divide and conquer design approach that runs into trouble deep into integration. Architects get the best of both worlds, with automation and configurable options for fine tuning. The on-chip features are impressive, and the addition of system level optimizations for hardware coherency takes Gemini 3.0 into new territory.

For more information on NetSpeed Systems and the Gemini 3.0 cache coherent network-on-chip IP, visit:

http://netspeedsystems.com/products/gemini/

Don Dingee is co-author of "Mobile Unleashed", a history of ARM architecture in mobile devices, and is a technologist and strategist for embedded and IoT design. He previously led product marketing efforts for a division of Motorola, and has one patent on the Stinger missile. He blogs regularly at SemiWiki.com. Don holds an MSEE from the University of Southern California, and lives just outside of San Antonio TX.

ABOUT NETSPEED SYSTEMS

NetSpeed Systems provides scalable, coherent, on-chip network IPs to SoC designers for a wide range of markets from mobile to high-performance computing and networking. NetSpeed's on-chip network IPs deliver significant time-to-market advantages through a system-level approach, a high level of user-driven automation and state-of-the-art algorithms.

CONTACT US

NetSpeed Systems Inc.
2870 Zanker Road, Suite 210
San Jose, CA 95134
(408) 914-6929
contact@netspeedsystems.com
www.netspeedsystems.com