SonicsSX On-Chip Network

DATASHEET

The SonicsSX On-chip Network combines advanced fabric features and comprehensive data flow services architected for high performance multicore and multi-subsystem cloud-scale SoCs:

High Performance:

High frequency Interconnect Fabric

Hybrid topologies for best-of-breed fit for subsystem and system-wide on-chip networks

Advanced fabric features and data flow services

Optimized for performance – best balance of frequency, width, and efficiency available

Interleaved Multichannel Technology (IMT) allows easy scaling to 2/4/8 channel DRAM

Low-power optimized:

Low idle and efficient active power

Robust power management capabilities – power down, power restore + retention flops

Efficient implementation reduces wire congestion and leakage

Unparalleled Support

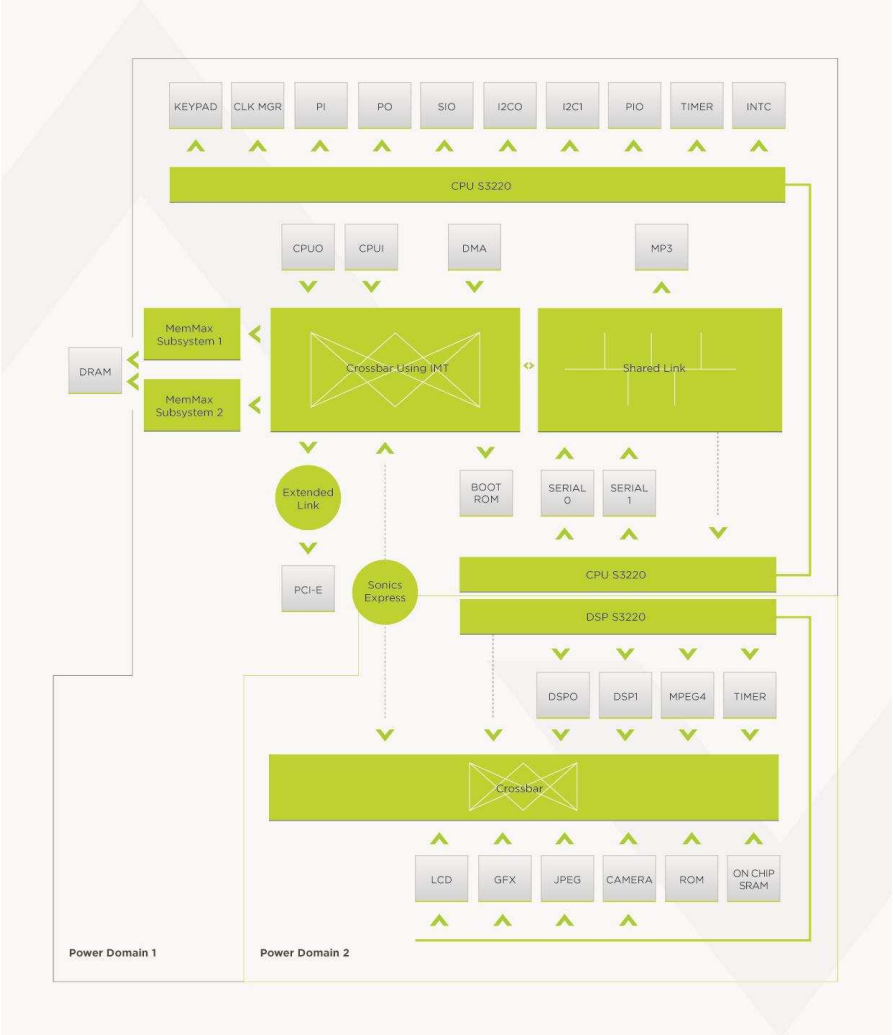
Unique design methodology eases timing closure and speeds design iteration cycles

Integrated performance analysis tools rapidly uncover design hotspots for quick recovery

World-class engineering support for first-time-right silicon

State-of-the-art development tools reduce design time from months to days

The SonicsSX® On-chip Network contains a high performance, advanced fabric and a comprehensive set of data flow services for the development of complex, multicore and multi-subsystem cloud-scale SoCs. By utilizing state-of-the-art physical structure design and advanced protocol management, SonicsSX can act as a local subsystem or global interconnect solution. Ideal for complex video processing and graphics subsystem clusters, SonicsSX provides designs high performance throughput while maintaining impressive power management capabilities. As a global interconnect solution, SonicsSX also supports system-wide quality-of-service mechanisms, access security, hardware monitoring/debug ports, and error handling features that facilitate higher design predictability and decrease chip development time. In addition, SonicsSX allows SoC developers to easily transition from single channel to multiple channel external DRAM memory architectures seamlessly and transparently to software and hardwired initiators.



This example SoC architecture utilizes the flexible topology and multi-channel features of SonicsSX, while employing Sonics3220™ to partition slow speed peripherals, the Sonics MemMax® memory subsystem solutions to achieve high performance DRAM memory management and SonicsExpress™ to support clock and voltage domain isolation. This example utilizes IMT mutlichannel memory management capabilities for two channels of external DRAM which enables throughputs up to 16 Gbytes/s per-port to be achieved while implementing quality of service guarantees across the entire SoC. Utilizing SonicSX in conjunction with Sonics' full line of complementary System IP provides designers with a complete SoC infrastructure on which to build the latest generation of consumer-friendly devices.

The SonicsSX interconnect system IP is configured using a suite of development tools from Sonics called StudioXE. StudioXE automates the configuration and modeling of Sonics complete line of system IP products and, specifically, facilitates the configuration of SonicsSX to exactly match the application needs of each particular user. In addition, it provides system modeling capabilities using both SystemC and Verilog output options to ensure rapid design flow integration, high predictability, and lower system verification risks.

Gain Access to Advanced Technology and Make Better Products

SonicsSX incorporates a comprehensive set of technology innovations designed for practical application for today's multicore SoCs. SonicsSX is also highly configurable so that each instantiation can exactly match specific SoC design requirements while also maintaining an interconnect centric, or platform style, architecture across product lines:

- Low latency Interconnect – SonicsSX combines key networking concepts, such as protocol layering, packetized internal transports, end-to-end Quality of Service mechanisms, and advanced error management, with flexible decoupling and topology choices to enable instances that offer both the low latency of computer structures and the scalability of networks. Intelligent point to point protocol management ensures that high performance data flows are achieved while maintaining low latencies and minimal power consumption.
- Multithreaded non-blocking flow control – A key design challenge of multicore SoCs is the ability to design and verify complex interconnect fabric architectures that guarantee IP cores can gain and maintain access to the shared internal and external memories when needed. The advanced technology innovations built into SonicsSX enables interconnect solutions that exactly match each SoC application's demands while minimizing traffic interaction across the high volume of transactions that occur in Multicore SoCs. End to end QoS adds another layer of predictability and scalability by guaranteeing bandwidth for each of the system elements while optimizing for low latency.
- Runtime software programmability – SonicsSX contains software visible registers for access by software drivers and applications to provide real time control of the interconnect operations.

Utilize Unprecedented Architecture Design Flexibility to drive Rapid Development with High Predictability

New product innovations built into SonicsSX provide architects, chip developers, and software developers with configuration options that ease system level design, IP core connectivity and fabric topology decisions.

- Advanced System Partitioning –SonicsSX uses socket based protocol management which decouples the IP cores from each other. This decoupling introduces a great level of modularity for any SoC design, facilitating late design changes with minimal re-design and verification.
- Universal IP core connectivity – Ensures any IP core supporting an OCP, AHB, or AXI interface can be attached to the interconnect seamlessly. Intelligent transport protocol management incorporated in SonicsSX guarantees high performance even when IP cores of different types are connected.
- Flexible Fabric Topologies – SonicsSX supports both cross bar and shared link topologies, with

the ability to create hybrid topologies, enabling an optimal balance of latency and bandwidth and wiring density to be achieved within a given architecture.

Advanced Configuration and Modeling Automation Accelerates Time-to-Market and Lowers System Verification Risks

Access to technology innovations supported in SonicsSX through the automated configuring and modeling capabilities of the StudioXE development suite facilitates rapid exploration and decision making with respect to the trade-offs between architecture choices:

- High Performance – The ability to rapidly configure an array of topology choices and then model and verify protocols and data flow paths at the system level enables SoC developers to match the interconnect behavior exactly to the SoC requirements and achieve the highest performance possible.
- Low Latency – Combining intelligent point to point protocol management ensures that high performance data flows also maintain low latencies.
- Low power – The advanced design optimizations as well as coarse and fine grained clock gating that are part of SonicsSX offers orders of magnitude lower power than conventional computer bus designed interconnects.
- Minimal Area - Automated configuration and modeling enables SoC developers to utilize the high configurability of SonicsSX to rapidly iterate system level instantiations and minimize area while preserving performance and power.
- Rapid Performance Verification – As part of the advanced automation provided within Sonics suite of development tools, SystemC models and RTL netlists are generated from the same database for use in performance validation. By employing statistical modeling within StudioXE, SoC developers can analyze and verify performance of the interconnect configuration before the SoC design begins.
- Concurrent hardware and software development – Pre-verified data flows ensure performance is predictable, enabling design teams to focus on hardware execution, and gives software teams the ability to rely on predictable performance early in the design cycle to streamline software development.

Take Advantage of Comprehensive Data Flow Services to Improve Time to Market and Lower Design and System Verification Risks

Comprehensive data flow services supported by SonicsSX enables engineering teams to conquer the system design challenges present in today's multicore and multi-subsystem SoC design in rapid time and with less project risk:

- Power management - Available to address both active and standby power management. When operating the interconnect in conjunction with power management controllers the complete design or specific portions of it can be shut down as needed. Internal coarse and fine grained clock gating are used to minimize the active and idle power.
- Quality of Service (QoS) – The ability to guarantee data flow in a complex heterogeneous distributed processing environment is paramount to the success of any multicore SoC design. The QoS data flow service built into SonicsSX ensures that SoC developers can manage any level of complexity.
- Security management- SonicsSX employs the latest innovations in security management, including firewalls, and regional isolation of portions of the SoC for conditional access. Security data flow services build on classical processor based security to provide SoC developers with a comprehensive scheme for addressing even the most stringent security requirements.
- Exception handling – SonicsSX employs advanced data collection techniques that enables the interconnect to monitor and report error conditions for streamlined error recovery. Key data collected during the data flow

management aides greatly in identifying and formulating error recovery routines minimizing the need to conduct a complete system reset.

- Side band signaling – SonicsSX provides designers with a level of logic flexibility that ensures synchronization can be rapidly achieved and verified across many components within the system. Having side band signaling as part of the interconnect guarantees this synchronization is handled in a consistent and logical manner across the entire system design.
- Data width conversion – The ability to seamlessly connect any IP core to SonicsSX ensures that SoC design and verification is greatly accelerated. Data width conversion is a key data flow management service which further ensures that the connection of IP cores across the system is handled seamlessly and with optimal efficiency.

Improve Product Line Management Through Advanced Architectural Automation

The combination of a highly configurable interconnect and the high degree of automation and technology innovation delivered as part of StudioXE ushers in a new methodology for managing product lines. By transitioning to an interconnect-centric, platform style architecture, entire product lines can be built from a single architecture. Taking advantage of the wealth of innovation provided by Sonics, many customers today have realized the compounding benefits associated with utilizing this methodology, including:

- Progressively declining development costs and risks as the product derivatives are re-engineered
- Transition of IP libraries at the pace most comfortable for the architecture and design teams
- Accelerating time to market with each derivative developed

Description

SonicsSX can act as a global or local subsystem interconnect. The high degree of configurability built into SonicsSX ensures that optimal performance, low power, and minimal area are all achieved for either use model. The high degree of automation delivered through StudioXE also ensures that configuring SonicsSX can be accomplished rapidly with a completely modeled result that is fully optimized.

As the local or global interconnect solution, SonicsSX handles communications functions such as system address decoding, data routing, buffering, flow control and arbitration. Full or partial crossbars assure low-latency paths between specific pairs of initiator and target SoC cores. Connections that do not require dedicated high bandwidth or low latency paths can use a shared bus, saving valuable die area without compromising performance.

Up to four crossbar and shared link exchanges can be joined to form a single SonicsSX instance that supports up to 64 cores. Multiple SonicsSX instances can be used in a single SoC design. SonicsSX supports SoC cores operating at different clock rates. Local clocks can be scaled (frequency divided) from a common source, and the scaling factor can change dynamically to minimize power demands. SonicsSX operates with power management controllers (external to the interconnect) to change voltage levels and clock rates for the SoC cores. SonicsSX interfaces manage rapid power down transitions while preventing data loss. SonicsSX provides a fully configurable SoC interconnect fabric that supports transport, routing, and arbitration and translation functions. To adapt to targets with long or unpredictable latency such as DRAM, SonicsSX establishes independent request and response networks that are configured with the overall topology.

SonicsSX is a member of the Sonics family of On-chip Network system IP products. Complementary products include MemMax™, an advanced memory subsystem solution, and Sonics3220™, a peripheral interconnect that off-loads slow transfers from the system interconnect. Key features include:

- Practical GALs support allows voltage and clock domain isolation configurable on a per port basis

- IMT multichannel memory management for high memory bandwidth and DDR3 burst width conversion
- 64 sockets per SonicsSX instance / unlimited instances
- Configurable to include full or partial crossbars and/or shared links
- A highly flexible structure to accommodate latency sensitive portions of the system
- User-specified connection map, address map, and command map
- Permits internally shared paths
- Natively supports OCP 3.x and 2.x sockets, and provides interfaces to AXI 3/4, and AMBA AHB-Lite 2 sockets
- Implements dynamic endianness aware width conversions
- Configurable address widths up to 64 bits
- Target core address decoding down to a granularity of 1 KB
- Configurable protected regions within address space
- Supports BLCK burst sequences on OCP2 sockets
- Configurable data path widths (sockets and internal) 16, 32, 64, 128, or 256 bits; four to one range of widths allowed within any instance
- Peak bandwidth limited only by target limitations
- Accommodates synchronous and synchronous-divide clock rates for each socket
- Flexible internal pipelining makes frequency independent of span
- Supports 0 cycle minimum latency paths
- Supports differential quality of service (low latency, allocated bandwidth, best efforts)
- Supports sideband signaling for interrupts, errors, and power controls
- Scalable frequency, trading off interconnect span against latency by adjusting the depth of the interconnect pipelines
- Minimized active power consumption using streamlined internal protocols and a physically-aware structure
- Power management interface coordinates voltage or clock removal
- Implements fine and coarse grained clock gating for low idle and active power
- Monitors for software error conditions (unsupported commands, addressing errors) and protection violations
- Monitors for failing cores (timeouts)
- Performs error logging and recovery support
- Optional debug ports to probe operation of internal paths
- Quality-of-service management assuring predictable performance for real-time data flows
- Dynamically configured access protection (firewall) for cores or memory regions against access by specific initiating cores or processes
- Separate request and response networks that adapt easily to target agents with long or unpredictable latency (such as DRAM systems)

Agents

Cores connect to SonicsSX through adaptive “Agents”, using standard socket interfaces that isolate the cores from one another. .Both socket and internal data path widths can be configured as 16, 32, 64, 128, or 256 bits. Agents within SonicsSX with IMT fully decouple the functionality of each SoC core from the interconnect communications required among the cores, allowing the cores to be re-used from system to system without rework.

Agents support OCP, AXI or AHB cores. Agents handle any mismatches in data width, clock frequency, or protocol among the various core interfaces while balancing the requirements of latency, physical span, clock frequency, die area and power consumption. SonicsSX ensures that data transfers are correctly handled. For targets and initiators, the state machines handle buffer and thread management, where a thread can create multiple paths on the same set of wires. To support ongoing (and outstanding) transfers for each thread, SonicsSX handles arbitration for all initiator cores and address decoding and routing for all targets. Each initiator or target agent decouples its associated core from the switching fabric.

Agents are embedded in an exchange and provide sockets for the attached cores. The agents use “register points” (small FIFOs) as needed to break long combinatorial timing paths and maintain the required frequency over distance. A “pipeline point,” consisting of a configurable number of register points, connects two exchanges. A pipeline point is responsible for staging data transfers and flow control between the exchanges it connects, adjusting the data width as required, and supporting the crossing of clock domain boundaries. A special agent, the register target is embedded in an exchange and provides an access point to the internal configuration registers. The register target is used to support the needs of the interconnect to support runtime configuration, sideband signaling, or power management.

Configurable register options are used as needed to adapt agent timing to the core. Agents automatically generate interfaces for OCP, AHB or AXI interfaces. Configuration options for each target agent include whether a debug port configuration registers are present and whether embedded register points are used. There are many other configuration options, primarily related to the particular core socket to be supported.

Register Points

Within an agent, embedded register points serve to cross synchronous clock domains, achieve a high clock rate, or bridge extended distances when timing is a factor. The depth of a register point is configurable, so that register points can be used to provide a queuing function. When a register point is embedded within a multi-threaded agent, the depth is configurable on a per-thread basis. Register points are directional, requiring one register point for the request network and one for the response network.

Exchanges

Agents are embedded in an exchange and provide sockets for the attached cores. The agents use “register points” (small FIFOs) as needed to break long combinatorial timing paths and maintain the required frequency over distance. A “pipeline point,” consisting of a configurable number of register points, connects two exchanges. A pipeline point is responsible for staging data transfers and flow control between the exchanges it connects, adjusting the data width as required, and supporting the crossing of clock domain boundaries. A special agent, the register target is embedded in an exchange and provides an access point to the internal configuration registers. The register target is used to support the needs of the interconnect to support dynamic configuration, sideband signaling, or power management. SonicsSX supports dedicated and shared links, using three types of exchanges which can be chained using pipeline points to span greater distances:

Crossbar Exchange (XB)
An XB exchange provides dedicated links connecting multiple initiators to multiple targets with minimum latency. Each XB exchange can connect up to 16 cores in an 8x8 configuration, with no more than 8 initiators (including pipeline points) connected to any one target and no more than 8 targets (including pipeline points) connected to any one initiator. Full crossbar connectivity is not required. The XB exchange provides arbitration for access to targets on the request side and initiators on the response side, while performing any data width conversion needed between initiators and targets.

Shared Link Exchange (SL)
A shared link (SL) is the primary component for wide connectivity between initiators and targets while spanning distance and using minimal wiring resources. A shared link provides a single time multiplexed but non-blocking data path for connections between the initiator and target threads. Shared links can be chained using pipeline points to span even greater distances. Each SL exchange can connect up to 32 cores, with up to 16 initiators (including pipeline points) connected to up to 16 targets (including pipeline points). Configuration options for the shared link include setting the data width, the addition of a debug port, and the physical topology of the modules comprising the link (mux bus), as derived from the floorplan.

Extender Link Exchange (EL)
Each EL exchange connects a single outlying initiator or target with an SL or XB exchange and is optimized to

span distance. The EL exchange contains an embedded initiator or target agent and uses a pipeline point to connect the EL exchange to the XB or SL exchange.

Configurable register options are used as needed to adapt agent timing to the core Agents automatically generate interfaces for OCP, AHB or AXI interfaces.

Configuration options for each target agent include whether a debug port configuration registers are present and whether embedded register points are used. There are many other configuration options, primarily related to the particular core socket to be supported.

Register Points

Within an agent, embedded register points serve to cross synchronous clock domains, achieve a high clock rate, or bridge extended distances when timing is a factor. The depth of a register point is configurable, so that register points can be used to provide a queuing function. When a register point is embedded within a multi-threaded agent, the depth is configurable on a per-thread basis. Register points are directional, requiring one register point for the request network and one for the response network.

Pipeline Points

Pipeline points (PP) are required between exchanges and are used to stage transfers between exchanges, adjust data widths or cross clock boundaries. Pipeline points are composed of register points (the same as used in the agents). Pipeline points permit the physical span from initiator to target to be increased without lowering the operating frequency of the connection. As needed, pipeline points permit frequency conversion (1:N or N:1), establishing distinct clock domains operating at different frequencies. SonicsSX with IMT supports chains of pipeline points to maintain maximum frequency when the exchanges.

Clock Domains

SonicsSX supports multiple clock domains. Agents and the register target may be assigned to specific clock domains. An exchange can reside in only one clock domain, with clock domain crossings occurring in the pipeline points between exchanges or in the register points embedded in the agents. This allows the agent to reside in the same clock domain as the core it connects to. Clock domain crossings require the specification of a separate clock for each domain. Clock domain crossings may be used to control the frequency of the clock domain, dividing a fast base clock by an integer divisor. Either the exchange or one of the embedded cores can have the fastest clock. The ability to dynamically change divide ratios provides an additional level of power management control.

Topologies

Up to 4 XB and SL exchanges and any number of EL exchanges can be joined to form a single SonicsSX instance, supporting up to 64 sockets. Each socket can connect to an initiator or target core, or another instance of a Sonics interconnect (either a SonicsSX or the Sonics3220 interconnect or the SonicsMX, SonicsLX, Stingray, or the Sonics MemMax, or the SonicsExpress). The user determines the overall topology of a SonicsSX instance, and can choose to use more than one instance for a particular SoC design. For example, a design might be partitioned into separate instances to more efficiently manage power consumption. Alternatively, it might be convenient to group a subset of the SoC's cores related to a specific function as a single SonicsSX with instance in anticipation of future reuse.

A SonicsSX instance can occupy a portion of the SoC address space from 2^10 to 2^64 bytes. Some of that addressable space may reside in targets that are directly connected to the SonicsSX instance, while some may reside in targets connected to other instances to which this instance is chained.

Multiple SonicsSX instances are linked using a pair of bridging OCP interfaces, one for each direction of request flow. If independent connections are required between initiators and targets on different SonicsSX instances, the OCP interfaces need to be configured for multiple threads. The internal thread map must avoid sharing threads on the bridging OCP among connections that should not interfere with one another. In addition, the address maps of the individual SonicsSX instances

must be set to avoid forwarding cycles. Partitioning a single SonicsSX instance into multiple instances can result in interface and performance changes. System level design activities, particularly configuration, are different if multiple SonicsSX instances are used to support a set of cores. OCP bridging from one instance to another introduces associated propagation delay, while system level performance may be slightly worse on some paths if multiple instances are traversed.

Performance

SonicsSX provides a number of performance advantages associated with threading and thread mapping, bandwidth, latency and QoS arbitration. The flexibility inherent in the architecture easily allows trade-offs between the selected features through their associated parameters, and gate count or performance. A variety of parameters can be easily modified showing initiator agent performance parameters.

Threading and Thread Mapping

All requests on a connection start on a thread of the initiator core and end on a thread of the target core. Within the SonicsSX interconnect, initiator threads are mapped to target threads. If the target core is single-threaded, every thread from every connected initiator maps to the single target thread. If the target is multi-threaded, the SoC designer must define the mapping between each initiator thread and one of the target's threads. Reverse mapping occurs in the response network with the response on the same initiator thread on which the corresponding request was launched. Thread mapping can take place at agents and within pipeline points.

Pipeline points are subdivided into register points so that each register point represents a single initiator or target core. The breakdown of pipeline points into register points and the placement of the actual mapping point from initiator to target threads and back is produced automatically and is designed to reduce overall buffering (gate count) without lowering performance.

Because some multi-threaded initiator cores do not need the full capability of independent resources and flow control for each thread, SonicsSX supports thread collapsing at the initiator socket. For example, an initiator core may exhibit more concurrency at its interface (more threads) than an application can exploit in its thread map, or the initiator implementation may fail to ensure that its threads cannot block one another. In such cases, the ability to merge threads immediately at the entry point to the SonicsSX interconnect can reduce the internal overhead needed without degrading performance. SonicsSX allows configuration of the initiator agent to perform input thread collapsing as static N-to-1 mapping. It can also perform output thread collapsing in the response path. These collapsing options are independent of one another. In a similar manner, SonicsSX target agents may be configured independently to perform thread collapsing in the request and/or response paths.

Bandwidth

The maximum bandwidth between an initiator and a target depends on the type of traffic and the components along the path. The type of traffic affects how the request and response networks are utilized, while the type of components along the path and how they are configured determine how many request and response transfers can be processed in a unit of time and the overall bandwidth that can be achieved.

The absolute maximum frequency supported by SonicsSX is specific to the targeted process. SonicsSX may further limit the maximum frequency based upon the number of pipeline stages that its internal logic paths are configured to employ. The logic delay remaining in the slowest of these stages, including flip-flop and clock delays, determines the maximum operating frequency. SonicsSX can be configured with a data width of 16, 32, 64, 128, or 256 bits depending on what is necessary for an application, independent of the data width of any of the connected IP cores.

SonicsSX employs split packetized transactions that allow some concurrent use of the request and response paths. Threading maximizes the use of paths by permitting some connections to operate while other connections are temporarily blocked.

Since the crossbar topology further enables concurrent traffic, the SonicsSX bandwidth scales by the degree of spatial concurrency. The actual bandwidth is the clock rate multiplied by the data path width, multiplied by the ratio of concurrent read/writes resulting from packetized transactions (a factor greater than 1 but less than 2) and the number of concurrent paths. Maximum bandwidth to a particular target may be reduced if there are inadequate outstanding requests to cover round-trip latency or by the need to share a link or target among multiple connections.

Latency

Latency is a function of the distance between initiator and target cores, the clock rates of the cores, and whether clock rate conversion is needed between the cores. Latency is also affected by the amount of translation and arbitration required between the initiator and target. SonicsSX can perform certain simple translations on transactions to maintain semantic equivalence and to permit protocol conversion among the supported interface specifications, including endianess-aware data width conversion and simple address offset translation. SonicsSX arbitration mechanisms then resolve cases where multiple incoming transactions from different initiators address a common target core.

The minimum latency along a request or response path between initiator and target is the sum of the minimum latency of the SonicsSX components along that path. The minimum latency of a component depends on its configuration.

Arbitration

The SonicsSX interconnect performs arbitration at all points of contention to resolve conflicts. The major arbitration points are in the XB and SL exchanges. In the crossbar, there is an independent point of arbitration for each target (requests) and each initiator (responses). In the shared link, there is an arbitration for access to the shared link on the request and another one on the response network side. Arbitration will also occur when an exchange exits to a pipeline point.

On the request side, arbitration is performed at transaction (burst) boundaries. If the target supports limited burst lengths, initiator bursts can be broken into smaller bursts. SonicsSX provides QoS modes that can be configured to govern the arbitration mechanism.

Response arbitration uses a least-recently-serviced policy between the branches of each arbitration point, favoring the branch that has been without service for the longest time. Response arbitration is performed at transfer (word) boundaries. In the SL exchange, only connections that are able to make progress are allowed to win arbitration so that one connection may not block other.

At multi-threaded staging points such as flip-flops where there may be multiple candidates for thread arbitration, additional arbitration points are employed to determine which thread to send next. For example, thread arbitration may exist on the request or response side of multi-threaded initiator agents or on either side of multi-threaded target agents.

Power Management

SonicsSX power management employs fine and coarse-grained mechanisms to reduce idle and active power levels reducing both active and idle power consumption.

Coarse-grained power management uses power control logic external to the SonicsSX interconnects that coordinates power and clock-control state transitions. Each SonicsSX instance provides a single power management interface for the entire interconnect. Coarse-grained power management turns each SonicsSX instance into an explicitly managed power region. Power is managed by removing the clock or supply voltages from the entire interconnect.

While a single instance of the SonicsSX interconnect may contain multiple clock domains, it is a single clock-gating domain with respect to coarse-grained power management. The SonicsSX power management interface notifies the power control logic when voltage

and clocks can be removed or restored without disrupting communications.

The external power control logic and SonicsSX interact through a handshake protocol of power control request and acknowledgement signals. Power management interfaces at each core socket allow each socket's activity status to be observed, indicating whether requests are still pending (for initiator agents) or are waiting to be sent to the target (for target agents). Such activity status signals can be used to help coordinate activity-dependent power management. For example, the activity status signal can be used to let initiator cores know when all requests (including posted writes) are completed or wake up targets (or other SonicsSX instances) that are currently powered down.

For designs employing multiple Sonics interconnects, only some instances may need to be shut down while others continue to operate normally. For these systems, a SonicsSX interconnect can be configured to act as a slave with wake-up capability. Additional active signals will be routed to the external power manager from adjacent SonicsSX interconnect instances.

Quality of Service (QoS)

QoS mechanisms impose selective flow control of multiple incoming threads to the target allowing control of latency and bandwidth on a per-target and per-thread basis. Each connection is configured with a bandwidth weight that is set at the initiator agent. Connections sharing the same target thread are allocated a share of the available target thread bandwidth based on the ratio of the bandwidth weights of each connection.

Weighted, priority or controlled QoS modes govern the arbitration policy of points within XB and SL exchanges and determine the types of thread arbitration priority levels that are available at the target. At a minimum, all initiating threads are non-blocking with respect to other initiating threads accessing targets within SonicsSX. Each target may be configured in one of three QoS modes, and each target thread may be assigned a QoS arbitration priority level as allowed by the target's QoS mode:

Weighted fairness
All threads going into the target are automatically assigned a level of best-effort and are given access to the target according to their assigned weights. In weighted mode, user-specified bandwidth ratios are maintained between the connections sharing a link or a target.

Priority
Arbitration priority levels are either priority or best-effort and are fixed for each thread. Priority level threads share bandwidth with each other but always win in arbitration over best-effort threads. The bandwidth available to best-effort threads depends on the priority thread traffic and may be zero.

Controlled
Controlled mode provides control of latency and bandwidth. In this mode, the target has an additional rate-based QoS control, allocated bandwidth. Controlled mode offers three QoS levels that can be assigned to target threads: best-effort, allocated bandwidth, and priority. Target agent registers are available to set bandwidth allocations for priority and allocated bandwidth threads. Rate-based control is employed to gauge whether priority or allocated bandwidth threads have realized their respective user-specified bandwidth allocations. Only one or two threads can be assigned priority level and share bandwidth with each other until their bandwidth allocation is exceeded. When this occurs the threads temporarily become best-effort threads. Allocated bandwidth threads operate in a similar fashion. The bandwidth made available to best-effort threads depends on the priority and allocated bandwidths thread traffic and may be zero.

Assigning a QoS mode to a target applies the mode to all connections leading to that target, through the request network to the point where thread mapping from the initiator to target thread occurs. Shared link arbitration inherits the properties of the threads sharing the link. If the threads modes are mixed, initiator threads receive the equivalent of a best-effort QoS level,

implying that sufficient bandwidth must be available to satisfy the requirements of the initiators.

Access Protection

SonicsSX provides an access protection mechanism to designate protection regions within the address space of specified targets. The mechanism can be dynamic, with protection region sizes and location programmable at run-time. It can also be role-dependent with permissions defined as a function not only of which initiator is attempting access but also which processing role the initiator is performing at that time. Up to 32 different user-specified roles are permitted. The protection definitions may overlap, and may also be assigned a priority to resolve access rights when the overlap is encountered.

The access protection mechanism allows roles to be defined as an encoding of the in-band attributes that are part of a request as transmitted through MReqInfo. The protection mechanism also allows groups of initiators to be defined. Region access is restricted to initiators that belong to permitted groups with the designated roles.

Interleaved Multichannel Technology

The global address space covered by a SonicsSX address region may be partitioned into a set of channels. The channels are non-overlapping and collectively cover the whole region. The number of channels for a region is a static value derived from the number of individual targets associated with the region, and from the nature of the target. A SonicsSX multichannel address region for a multichannel target is associated with multiple individual targets and can have 2, 4, or 8 channels. The address space assigned to each channel is formed as the union of a set of disjoint sub-regions within the SSX address region called “low channel interleaves”. There can also be “high channel interleaves” that each contain a power of 2 number of low channel interleaves. The address space covered by each low channel interleave is mapped to one active channel target (of a multichannel target).

Example instantiation using IMT The figure above shows the internal structure of a SonicsSX example including 3 IAs and 3 TAs, where TA0 and TA1 are target agents belonging to a multichannel target, DRAM. Only one multichannel target exists in this example.

On the request network, for IA0, the multi-channel path going to the multi-channel target DRAM splits at the IA0’s embedded, request-side RS, Req_rs10. Since there are two channels, the two outgoing single-threaded (ST) DL links (in dashed pink arrow) each goes to a different channel target. The third outgoing ST DL link (in dashed black arrow) is a normal path leading to a normal individual target agent TA2. A request-side channel splitter is color-coded in pink. For the channel target TA0, the MS component, tat00_ms0, upstream to TA0 acts as a channel merger and regulates channel traffic coming from two different initiator agents, IA0 and IA1. The request-side channel merger tat00_ms0 is color-coded in pink.

On the response network, for TA1, the embedded RS component, Resp_rs01, acts as a response channel splitter – it has three outgoing links for delivering channel responses back to IA0, normal responses back to the normal IA2, and channel responses back to IA1, respectively. A response-side channel splitter is color-coded in blue. For IA1, its upstream MS component, Iah11_ms0, is a channel merger, which not only regulates responses coming back from channel 0 (i.e., TA0) and channel 1 (i.e., TA1) but also handles responses returned by the normal TA2. The response-side channel merger is color-coded in blue.

Since a response-side channel merger MS needs to regulate channel responses but it does not have enough information to act upon, additional re-ordering information needs to be passed to it from the request-side channel splitter of the initiator agent. For instance, the DRL link, in brown, is used to pass response re-ordering information between the request-side channel splitter, Req_rs11, and the response-side channel merger, Iah11_ms0, for IA1.

SonicsSX flow control interlock assures no initiator thread will have transactions outstanding to more than one target at a time. However, it does permit

transactions from one initiator thread to be outstanding to multiple channels at once and therefore to multiple individual targets within a multichannel target at once. This includes a transaction targeted at two different channels, as well as, two transactions (from the same initiator thread) each targeted at a single but different channel, where these two different channels are mapped to two individual targets within a multichannel target.

Since the rate of progress at these individual targets may be different, it is possible that responses will be returned to SonicsSX out of order with respect to how the initiator requests corresponding to those responses were presented by SonicsSX. SonicsSX implements internal mechanisms to manage the request order for each thread, and to use internal flow control in the response network to enforce response ordering restrictions.

SonicsSX is optimized for multichannel targets with non-blocking (i.e. OCP mthreadbusy_exact) response flow control. If the target socket uses blocking flow control or no flow control, then the target agent will interlock requests on each thread with available response buffers for that thread, so that the response path never blocks.

The internal mechanisms on the request path implement multichannel request interlocking to assure deadlock avoidance. This assures that the first transfer of bursts going to a multichannel target thread is delivered to the multichannel target thread strictly in order. Commonly, these interlock delays will be invisible.

The SonicsSX request interlock mechanism's delays should only impact performance for bursts that meet all criteria here simultaneously:

1. The burst is on the same initiator thread as the previous burst.
2. The burst is to a different channel as the previous burst.
3. The burst is short (measured in interconnect clock cycles) compared to the depth of the interconnect.
4. The burst encounters little or no contention on the way to the target.
5. The initiator shares the target thread with some other initiator.

On the response path, the SonicsSX mechanism for ensuring thread ordering using response network flow control impacts performance only to the extent that latency in the flow control loop can slow switching the response stream from one target to another. SonicsSX relies on elasticity buffers either in the target agent, or in the target core to minimize gate count costs. Internal mechanisms are designed to minimize switching delays between response sources. To the extent that these delays are visible, they do not limit the target response bandwidth. They may only limit initiator response bandwidth on a single thread so that it is somewhat less than 100%. Since multiple initiators share a single target, it is already true that initiator bandwidth demands are less than 100%. Therefore, SonicsSX is architected to balance cost and performance.

Protocol Support

SonicsSX offers maximum reuse of SoC cores regardless of their source by providing sockets for cores with Open Core Protocol, AMBA AHB Lite and AMBA 3 AXI bus interfaces. SonicsSX with IMT natively supports the OCP 2.0 protocol and automatically creates low-overhead bridge logic to support cores with OCP 1.0, AHB, and AXI interfaces.

Configuration Options

The configuration options for SonicsSX are either structural (design-time) or run-time options. Structural configuration options must be defined as the interconnect is being created and are then fixed in the hardware. By contrast, run-time configuration options can be changed in the hardware by accessing internal configuration registers.

Structural Configuration Options

The following provides an overview of the various SonicsSX parameters that can be configured as structural configuration options. Specific configurable SonicsSX mechanisms are first identified, followed by

the configurable parameters per each core socket, exchange, register point, initiating or target thread, and connection, and finally the configurable parameters associated with sideband signaling.

Per socket: Option for OCP and AHB configuration, Timeout protection, Error logging, outstanding requests, activity status flags, inband and out of band error conversion, Address fill-ins for initiators, initiator ID fill-ins, flop options, response buffering, thread collapsing, QoS modes, error steering and memconID are all configurable per core socket.

Per exchange: link widths and data paths.

Per register point: register depth (target thread: Arbitration priority level)

Run-Time Configuration Options

The following provides an overview of the various SonicsSX parameters that can be configured as run-time configuration options, thus modifying behavior of the SoC as it operates. Parameters can be configured per core socket, per target thread, or per connection, and sideband signaling can also be enabled.

- Per Connection: Write posting, info mapping, bandwidth weights
- Sideband signaling: straight through signals, composite signaling, output flag enable
- Per core socket: protection regions, address fill-in, agent control
- Per target: bandwidth allocation