# **Defining a General-Purpose On-Chip Bus Architecture for Heterogeneous Multi-Processor Systems: Supporting PowerPC, Arm, TriCore, and RISC-V ISAs**

## **1. Executive Summary**

The design of modern System-on-Chips (SoCs) is increasingly characterized by the integration of heterogeneous processing elements, including diverse Instruction Set Architectures (ISAs) such as PowerPC, Arm, TriCore, and RISC-V. This heterogeneity allows for the optimization of performance, power, and area for specific tasks, but it presents significant challenges for on-chip communication. A robust and general-purpose bus architecture is paramount to ensure efficient data exchange, maintain cache coherency, and facilitate IP reuse across these disparate processor ecosystems.

This report provides an in-depth analysis of the requirements and potential solutions for defining such a general-purpose on-chip bus architecture. It begins by examining the evolving SoC landscape and the critical role of the interconnect. It then delves into the native or common bus ecosystems associated with PowerPC (CoreConnect, CoreNet), Arm (AMBA AXI, AHB, APB, CHI), TriCore (SRI, FPI/SPB/BBB), and RISC-V (TileLink, AXI).

A comprehensive survey of prominent on-chip bus standards—namely AMBA AXI, AHB, APB, CHI, Wishbone, and TileLink—is conducted, evaluating their respective features, transaction types, configurability, coherency mechanisms, and ecosystem maturity. This is followed by a comparative analysis of these standards against key metrics such as performance, complexity, scalability, coherency support, IP availability, and licensing models.

Based on this analysis, several strategies for implementing a general-purpose heterogeneous bus architecture are explored. These include adopting a single dominant standard (AXI/CHI or TileLink) as the backbone, or implementing a hierarchical and bridged architecture. The complexities of bus bridging, including protocol translation, clock domain crossing, data width adaptation, and maintaining cache coherency across bridges, are discussed in detail. Network-on-Chip (NoC) approaches are also considered for highly complex systems.

The report culminates in a set of recommendations. A hierarchical and bridged architecture, leveraging a mature, high-performance, and coherent backbone such as AMBA AXI (specifically AXI5 with CHI for coherent domains), is identified as the most pragmatic and versatile approach. This strategy balances the need for a robust, well-supported primary interconnect with the flexibility to integrate diverse IP cores through specialized, high-quality bridges. The critical importance of meticulous bridge design and verification, robust cache coherency management across different domains, and a clear strategy for IP integration are emphasized as key success factors. The development of such an architecture is presented as an evolutionary process, vital for enabling future innovation and maximizing IP reuse in complex heterogeneous SoCs.

## **2. Introduction: The Challenge of Heterogeneous SoC Interconnects**

### **The Evolving SoC Landscape**

The trajectory of System-on-Chip (SoC) design is marked by an accelerating trend towards functional specialization through the integration of diverse processor Instruction Set Architectures (ISAs). Modern SoCs frequently combine PowerPC cores known for their robust performance in networking and embedded control, Arm processors dominating mobile and increasingly server/automotive spaces, TriCore microcontrollers tailored for real-time automotive and industrial applications, and the rapidly emerging RISC-V ISA offering open-source flexibility for a wide array of custom and general-purpose computing.1 This amalgamation of ISAs on a single die allows designers to optimize for specific workload requirements, ranging from hard real-time control and digital signal processing to general-purpose compute, high-performance graphics, and dedicated AI/ML acceleration. The proliferation of specialized cores—CPUs, DSPs, GPUs, NPUs, and various hardware accelerators—is a direct response to the multifaceted demands of contemporary applications.1 This inherent heterogeneity implies that a simplistic, monolithic bus architecture is unlikely to suffice; the interconnect must be inherently adaptable or structured hierarchically to cater to this diversity.

### **The Interconnect Imperative**

The on-chip communication fabric, or interconnect, serves as the central nervous system of an SoC. It dictates how effectively these diverse processing elements and other Intellectual Property (IP) blocks can collaborate. Consequently, the interconnect architecture is a critical determinant of overall system performance, power consumption, silicon area, and design complexity.6 An inefficient interconnect can lead to performance bottlenecks, excessive power draw, and increased die size, negating the benefits of specialized processing cores. Conversely, a well-designed interconnect can unlock the full potential of a heterogeneous SoC.

The open-source movement, particularly exemplified by the RISC-V ISA and open bus standards like Wishbone and TileLink, introduces both opportunities and challenges.1 While offering unprecedented flexibility and freedom from royalties, these open standards necessitate careful consideration regarding their integration with established proprietary ecosystems and the associated verification complexities. For a general-purpose bus architecture to be truly effective, it must either be an open standard itself or provide robust, well-verified bridging mechanisms to these emerging open standards to leverage the expanding pool of RISC-V IP.

### **Defining "General-Purpose"**

In the context of this report, a "general-purpose" on-chip bus architecture is one that exhibits sufficient flexibility and capability to efficiently connect and manage communication between IP cores based on PowerPC, Arm, TriCore, and RISC-V ISAs. This implies:

* **Support for Diverse Transaction Types:** Accommodating simple register accesses, block data transfers, burst transactions, and atomic operations.
* **Scalable Performance Levels:** Catering to a wide spectrum of bandwidth and latency requirements, from low-speed peripheral control to high-throughput memory and inter-processor communication.
* **Comprehensive Coherency Management:** Providing mechanisms for hardware-managed cache coherency across multiple processor cores and clusters, which may belong to different ISAs.
* **Facilitation of IP Reuse:** Enabling the integration of existing and future IP blocks with minimal interface modification, regardless of their native bus protocol.
* **Configurability and Extensibility:** Allowing adaptation to specific SoC requirements through parameterization and well-defined extension points.

The pursuit of such a general-purpose architecture is not merely a technical exercise; it reflects a long-term strategic objective for enhancing IP reuse, streamlining SoC design methodologies, and enabling platform scalability for future product generations. The architectural choices made will profoundly influence future SoC development roadmaps and the ability to rapidly respond to evolving market demands.

### **Report Objectives and Scope**

This report aims to analyze the requirements and propose viable strategies for defining a general-purpose on-chip bus architecture suitable for heterogeneous SoCs incorporating PowerPC, Arm, TriCore, and RISC-V processors. The scope encompasses:

* A review of the native or commonly adopted bus interfaces for each target ISA.
* A detailed survey of prominent on-chip bus standards, including AMBA AXI, AHB, APB, CHI, Wishbone, and TileLink.
* A comparative analysis of these standards based on critical architectural attributes.
* An exploration of architectural strategies, including single-standard backbones, hierarchical/bridged systems, and Network-on-Chip (NoC) approaches.
* A discussion of key technical considerations such as data width, endianness, arbitration, QoS, error handling, security, and clock domain crossing.
* The formulation of actionable recommendations for a robust and versatile bus architecture.

## **3. Processor Architectures and Their Native/Common Bus Ecosystems**

Understanding the typical on-chip communication interfaces associated with each target processor ISA is fundamental to defining a general-purpose bus architecture. Processor vendors often develop or adopt bus systems tailored to their core's capabilities and the demands of their primary markets. This specialization means that a "universal" bus might struggle to be optimally efficient for all ISAs simultaneously, thereby highlighting the importance of configurability or effective bridging strategies in a heterogeneous environment. Furthermore, the level and mechanisms of cache coherency support vary significantly across these ecosystems, posing a central challenge for any architecture aiming to enable coherent memory sharing between different processor types.

### **3.1. PowerPC Family**

PowerPC processors, with a long history in embedded systems, networking, and automotive applications, have utilized several on-chip bus architectures, primarily driven by IBM and later NXP (formerly Freescale).

CoreConnect Architecture (IBM/NXP):

IBM's CoreConnect bus architecture has been a mainstay for many PowerPC-based SoCs. It is a hierarchical system designed to segregate traffic based on performance requirements.15

* **Processor Local Bus (PLB):** This is the high-performance bus, typically 64-bit or wider, connecting the PowerPC core(s) to main memory controllers and other high-bandwidth master and slave peripherals.15 The PLB supports features like concurrent read/write operations and burst transfers to maximize throughput.18
* **On-Chip Peripheral Bus (OPB):** A 32-bit bus designed for lower-speed peripherals, the OPB offloads traffic from the PLB, allowing it to be dedicated to performance-critical transfers.15 An OPB bridge typically connects the OPB domain to the PLB.
* **Device Control Register (DCR) Bus:** This is a separate, low-bandwidth bus specifically for accessing configuration and status registers of peripherals.15 Removing these accesses from the memory-mapped PLB or OPB can simplify the address map and reduce loading on those buses.

The CoreConnect architecture provides a structured approach for SoC design, but its specific nature may require bridging when integrating with systems based on other bus standards.

NXP QorIQ CoreNet Fabric:

For more recent and higher-performance PowerPC-based SoCs, particularly in the QorIQ families (e.g., T-series, P-series), NXP employs the CoreNet fabric.19 CoreNet is a more advanced, fabric-like interconnect that supports both coherent and non-coherent transactions across multiple e5500 or e500 Power Architecture cores, Data Path Acceleration Architecture (DPAA) engines, memory controllers, and other peripherals.19 It offers features like transaction prioritization and bandwidth allocation, essential for complex multi-core SoCs. The introduction of CoreNet signifies a shift in PowerPC interconnects towards more scalable and coherent solutions, akin to those seen in other high-performance processor ecosystems.

Key Characteristics:

PowerPC systems have traditionally been strong in embedded control and networking applications. Their bus systems often reflect a hierarchical approach to manage different performance domains. A critical consideration when integrating PowerPC IP is endianness, as PowerPC is typically big-endian, although some later cores offer bi-endian support.21 Any general-purpose bus architecture must provide a clear strategy for handling endianness conversion if PowerPC cores are to share data seamlessly with little-endian cores like most Arm and RISC-V implementations.

### **3.2. Arm Architecture Family**

The Arm architecture is ubiquitous, spanning from low-power microcontrollers to high-performance application processors and servers. The Advanced Microcontroller Bus Architecture (AMBA) is the de facto on-chip interconnect standard for Arm-based SoCs, offering a suite of protocols catering to different performance and complexity needs.22

* **AXI (Advanced eXtensible Interface):** AXI is Arm's flagship protocol for high-performance, high-frequency system designs. It is widely used for connecting Cortex-A series application processors, Neoverse infrastructure processors, Mali GPUs, memory controllers, and other high-bandwidth components.22 Key AXI features include separate address/control and data phases, burst-based transactions (FIXED, INCR, WRAP types), support for multiple outstanding addresses, out-of-order transaction completion, and unaligned data transfers facilitated by byte strobes.22 These features enable high utilization of the bus and efficient data movement.
* **ACE (AXI Coherency Extensions) & ACE-Lite:** To support multi-core processors and heterogeneous compute systems requiring shared memory, ACE extends AXI with additional signals and transaction types to enable system-wide hardware cache coherency.1 ACE allows multiple masters (e.g., CPU clusters) to maintain coherent caches. ACE-Lite provides a lighter-weight, one-way (I/O) coherency, typically used by DMA engines or network interfaces that need to read coherently from CPU caches but do not have their own coherent caches.
* **CHI (Coherent Hub Interface):** As part of AMBA 5, CHI represents Arm's latest generation high-performance coherent interconnect protocol.22 It is designed for even greater scalability and performance in complex SoCs, often superseding ACE in high-end multi-cluster designs. CHI introduces a layered architecture (Protocol, Network, Link) and specific node types (Requester, Home Node, Slave Node) to manage coherent transactions and Distributed Virtual Memory (DVM) operations effectively.1 It aims to reduce congestion and provide a highly efficient transport layer for coherent traffic.
* **AHB (Advanced High-performance Bus) / AHB-Lite:** AHB is a simpler, single clock-edge protocol commonly used for connecting Arm Cortex-M microcontrollers, on-chip memories, and system-level peripherals that require higher bandwidth than APB but not the full complexity of AXI.22 It features centralized arbitration and supports burst transfers. AHB-Lite is a subset of AHB, further simplified for single-master systems.22
* **APB (Advanced Peripheral Bus):** APB is designed for low-bandwidth control accesses, such as register interfaces on system peripherals like UARTs, timers, and GPIOs.22 It has a very simple, non-pipelined interface with minimal signal complexity, making it ideal for low-power and low-area peripheral connections.

The comprehensive and evolving nature of the AMBA specification means that any general-purpose bus architecture will inevitably need to interface efficiently with a multitude of AMBA-compliant IPs.

Key Characteristics:

Arm's bus ecosystem is highly scalable, catering to a vast range of applications. A strong emphasis is placed on hardware-managed cache coherency in multi-core and multi-cluster systems through ACE and CHI. Arm processors predominantly support little-endian byte ordering, though some configurations allow for byte-invariant big-endian mode.21

### **3.3. Infineon TriCore Architecture**

Infineon's TriCore architecture is specifically designed for demanding real-time embedded systems, particularly in the automotive and industrial sectors. It uniquely combines RISC processing, DSP capabilities, and microcontroller features within a single core.40 The on-chip bus system is architected to support these real-time and safety-critical requirements.

* **System Resource Interconnect (SRI) Fabric:** The SRI fabric serves as the high-bandwidth backbone in TriCore MCUs.40 It is typically a 64-bit crossbar-based interconnect that connects the TriCore CPUs, DMA modules, and other high-bandwidth resources like tightly coupled memories and program flash. The SRI supports parallel transactions between multiple masters and slaves and allows for pipelined requests to optimize data flow.40 Detailed access latencies for various resources connected via SRI are often documented, which is crucial for real-time analysis.40
* **System Peripheral Bus (SPB):** The SPB is designed to connect the TriCore CPUs and DMA to medium and low-bandwidth peripherals.40 Masters on the SPB typically access SRI-attached resources via an SFI\_F2S (SRI Fabric Interface to SPB) bridge.
* **Back Bone Bus (BBB):** The BBB provides connectivity for CPUs, DMA, and SPB masters to specialized resources, particularly those related to Advanced Driver-Assistance Systems (ADAS).40 Access to BBB resources from SRI or SPB domains is also managed through bridges.
* **Flexible Peripheral Interconnect (FPI) Bus:** The FPI bus is a multi-master, typically 32-bit, interconnect protocol that serves as the foundation for peripheral buses like the SPB within the TriCore architecture.42 It is optimized for quick bus acquisition and high transfer rates, supporting 8-bit, 16-bit, and 32-bit data transfers, as well as larger 64-bit, 128-bit, and 256-bit block transfers and atomic read-modify-write (RMW) operations.45 FPI includes features like slave-controlled wait state insertion, timeout detection, and various arbitration schemes (priority-based, round-robin, starvation prevention) to ensure reliable and deterministic peripheral communication.45 The documentation 45 clarifies that system-on-chip communication is based on SRI and FPI protocols, with FPI connecting high-speed units like CPUs and DMA to medium and low-bandwidth peripherals.

The TriCore bus system, with its hierarchical structure (SRI for high-speed, FPI-based SPB/BBB for peripherals) and features supporting determinism, is crucial for its target applications. Integrating TriCore IP into a broader heterogeneous system requires careful consideration of these bus interfaces and their specific real-time characteristics.

Key Characteristics:

TriCore systems prioritize determinism, low interrupt latency, and functional safety. Their bus architecture reflects these priorities with features like predictable access times and robust error handling. Mixed-signal capabilities are also common in TriCore MCUs.

### **3.4. RISC-V Architecture**

The RISC-V ISA is distinguished by its open and modular nature, which extends to its on-chip interconnect strategy: RISC-V International does not mandate a single, specific bus standard.11 This openness provides flexibility but also means that designers must choose from existing standards or develop proprietary solutions.

* **Commonly Used Protocols:**
  + **TileLink:** This open standard, originating from SiFive and now fostered by the CHIPS Alliance, has gained significant traction within the RISC-V ecosystem.10 TileLink is designed to be scalable, supports robust cache coherency (TL-C conformance level), and aims for verifiable deadlock freedom.10 Its different conformance levels (TL-UL for Uncached Lightweight, TL-UH for Uncached Heavyweight, and TL-C for Coherent) allow it to cater to a range of IP blocks, from simple peripherals to complex, coherent processor clusters.10
  + **AMBA AXI/AHB:** Due to the mature and extensive IP ecosystem surrounding Arm's AMBA protocols, many RISC-V core providers and SoC designers opt to equip their RISC-V components with AXI or AHB interfaces.1 This facilitates integration with a vast array of third-party IPs and leverages existing verification tools and methodologies.
  + **Proprietary Buses:** For communication within tightly coupled RISC-V processor clusters, some designs may employ proprietary bus protocols optimized for specific microarchitectural features or performance targets.1 These are typically not exposed externally.
* **Inter-Cluster Coherency:** For larger SoCs featuring multiple RISC-V processor clusters that require cache coherency, Arm's AMBA CHI protocol is increasingly being adopted as the interconnect fabric, often implemented as part of a Network-on-Chip (NoC).1 AMBA ACE has also been used for this purpose. This trend underscores the need for high-performance, scalable coherent interconnects in advanced RISC-V systems.
* **RISC-V International Efforts:** RISC-V International hosts various technical committees and working groups focused on SoC infrastructure aspects beyond the core ISA.54 These groups work on standardizing interfaces and specifications for components like IOMMUs, debug modules, interrupt controllers, and Quality of Service (QoS) register interfaces.55 While these efforts are crucial for building complete RISC-V SoCs, a RISC-V-branded on-chip data bus standard intended to replace established protocols like AXI or TileLink is not yet prominent. The current strategy appears to be enabling the use of existing open or de facto industry standards.

The flexibility inherent in the RISC-V ecosystem means that a general-purpose bus architecture must be readily compatible with widely adopted open standards like TileLink or established industry standards like AMBA AXI/CHI to effectively integrate RISC-V based IP.

Key Characteristics:

The RISC-V ecosystem is characterized by its openness, modularity, and scalability. Processors are typically little-endian, though endianness can be configurable in some implementations.12 The lack of a single mandated bus encourages innovation but requires careful consideration of interface compatibility during SoC integration.

### **Table 1: Overview of Processor ISAs and Native/Common Bus Interfaces**

| **Processor ISA** | **Typical Native/Common Bus Interface(s)** | **Key Bus Characteristics (Data Width, Coherency, Typical Use)** | **Endianness Support** |
| --- | --- | --- | --- |
| PowerPC | CoreConnect (PLB, OPB, DCR) 15, CoreNet 19 | PLB: 64-bit+, high-perf CPU/Mem. OPB: 32-bit, peripherals. DCR: Config. CoreNet: Coherent fabric for multi-core. | Typically Big-Endian, some bi-endian support 21 |
| Arm | AMBA AXI, ACE, CHI, AHB, APB 22 | AXI: 32-1024bit, high-perf, optional coherency (ACE). CHI: High-perf coherent fabric. AHB: 32-1024bit, MCU/system perf. APB: 32-bit, low-bw peripherals. | Little-Endian, some Big-Endian (BE8) support 21 |
| TriCore | SRI Fabric, FPI (SPB, BBB) 40 | SRI: 64-bit crossbar, high-perf CPU/Mem/DMA. FPI/SPB: 32-bit, medium/low-bw peripherals, atomic RMW. Deterministic, real-time focus. Coherency typically localized or managed by specific implementations. | Implementation-specific (often Big-Endian) |
| RISC-V | TileLink (TL-UL, TL-UH, TL-C) 10, AMBA AXI/CHI 1 | TileLink: Configurable width, TL-C for coherency, open standard. AXI/CHI: Adopted for IP ecosystem compatibility and high-performance coherency. No single mandated bus; varies by implementation. Intra-cluster often proprietary, inter-cluster uses AXI/CHI or TileLink. | Typically Little-Endian, configurable in some cores 12 |

This table provides a concise summary of the diverse bus landscapes associated with each target processor. It immediately highlights the heterogeneity challenge: different processors "speak" different bus languages natively. This underscores the need for either a highly adaptable single bus standard or effective bridging strategies. The choice of bus architecture will heavily influence the ease of IP integration. If a standard widely supported by third-party IP vendors is chosen as the backbone, it can significantly reduce integration effort. Conversely, a less common or new standard would require more custom wrapper/bridge development.

## **4. Survey of Prominent On-Chip Bus Standards**

To establish a general-purpose bus architecture for heterogeneous SoCs, a thorough understanding of existing, well-established on-chip bus standards is essential. These standards provide the foundational protocols and methodologies for inter-component communication. This section details the features, capabilities, and typical applications of leading candidates: the Arm AMBA family (AXI, AHB, APB, CHI), Wishbone, and TileLink. The evolution of these standards, particularly AMBA, reflects the growing demands of SoCs for higher performance, increased complexity, and robust hardware-managed cache coherency, all of which are critical considerations for a modern general-purpose architecture.

### **4.1. AMBA (Advanced Microcontroller Bus Architecture)**

Arm's AMBA specifications have become the de facto industry standard for on-chip communication, offering a hierarchical set of protocols to meet diverse performance and complexity requirements.

#### **4.1.1. AXI (Advanced eXtensible Interface) - (IHI0022J** 22**)**

AXI is designed for high-performance, high-frequency, and high-bandwidth communication, typically connecting processors, memory controllers, and other performance-critical IP blocks.22

* **Channels:** AXI employs five independent channels for read and write operations: Write Address (AW), Write Data (W), Write Response (B), Read Address (AR), and Read Data (R).22 This separation allows for concurrent processing of different transaction phases.
* **Transaction Types:** AXI is a burst-based protocol. Masters issue a starting address and control information, and a burst of data transfers follows. Supported burst types are FIXED (address remains constant), INCR (address increments), and WRAP (address wraps within a defined boundary).22 It supports unaligned data transfers using byte strobes (WSTRB) and allows for multiple outstanding transactions, enabling slaves to return data out of order, identified by transaction IDs (AxID, WID, RID, BID).22 Atomic operations are supported through Exclusive Access (using AxLOCK) and Locked Access mechanisms.24
* **Key Signals:** Each channel has its own set of signals. Core signals include AVALID (master indicates valid address/control), AREADY (slave indicates ready to accept), AADDR (address), ALEN (burst length), ASIZE (burst size, bytes per transfer), ABURST (burst type), ALOCK (lock type for exclusive/locked access), ACACHE (memory attributes like bufferable, cacheable, allocate policies), APROT (protection attributes like privileged, secure, instruction/data), AID (transaction ID). Write data channel includes WDATA, WSTRB (byte strobes), WLAST (last transfer in burst). Read data channel includes RDATA, RRESP (response status: OKAY, EXOKAY, SLVERR, DECERR), RLAST. Write response channel includes BRESP.24
* **Configurability:** AXI is highly configurable. Data bus widths can range from 8 bits to 1024 bits (and beyond in some versions). Address width is also configurable, determining the addressable memory space. Transaction ID widths are configurable to support varying numbers of outstanding transactions.25
* **Coherency:** The base AXI protocol does not enforce cache coherency. AXI Coherency Extensions (ACE) and ACE-Lite were introduced to add system-wide hardware coherency.1 The ARCACHE and AWCACHE signals provide memory attribute information crucial for caching and coherency management.25
* **Use Cases:** AXI is the backbone for high-performance communication in Arm Cortex-A based SoCs, connecting CPUs to memory subsystems, GPUs, DMA controllers, and high-bandwidth peripherals. It's also frequently used as the interface for IP blocks in Network-on-Chip (NoC) designs.22

#### **4.1.2. AHB (Advanced High-performance Bus) - (IHI0033C** 22**)**

AHB is a simpler bus protocol than AXI, targeted at connecting embedded processors (like Arm Cortex-M series), on-chip memories, and medium-to-high-bandwidth peripherals.22

* **Characteristics:** AHB is a single clock-edge protocol. It uses a centralized arbiter and a decoder to select one master and one slave for each transfer.22 Transactions consist of an address phase followed by one or more data phases.
* **Transaction Types:** AHB supports single transfers and burst transfers. Burst types include undefined length incrementing (INCR), fixed-length incrementing (INCR4, INCR8, INCR16), and fixed-length wrapping (WRAP4, WRAP8, WRAP16).22 The HTRANS signal indicates transfer types: IDLE (no transfer), BUSY (master inserting wait states in a burst), NONSEQ (first transfer of a burst or a single transfer), and SEQ (subsequent transfers in a burst).34 Base AHB does not support out-of-order completion or split transactions.
* **Key Signals:** Core signals include HCLK (bus clock), HRESETn (reset), HADDR (address), HWDATA (write data), HRDATA (read data), HWRITE (transfer direction), HSIZE (transfer size), HBURST (burst type), HPROT (protection attributes), HTRANS (transfer type), HMASTLOCK (locked transfers), HREADY (slave/interconnect indicates ready, allows wait states), HRESP (transfer response: OKAY, ERROR).30 AHB5 adds HNONSEC (non-secure transfer) and HEXCL/HEXOKAY (exclusive access support).34
* **Configurability:** Data bus width is configurable from 8 bits to 1024 bits.34 Endianness is supported, with AHB5 clarifying BE8 (byte-invariant) and BE32 (word-invariant) big-endian modes.34
* **Use Cases:** Widely used in microcontroller-based SoCs (e.g., Arm Cortex-M series 22), connecting the processor to on-chip SRAM, Flash controllers, and peripherals like DMA controllers or communication interfaces. AHB-Lite is a subset for single-master systems, simplifying the design.22

#### **4.1.3. APB (Advanced Peripheral Bus) - (IHI0024C** 22**)**

APB is designed as a low-power, low-bandwidth, and simple interface for accessing configuration registers and connecting slow peripherals.22

* **Characteristics:** APB is a non-pipelined protocol. Every transfer takes at least two clock cycles: a SETUP cycle and an ACCESS cycle.37 It has a minimal signal list, reducing complexity and power consumption.
* **Transaction Types:** APB supports only single read and single write operations.37
* **Key Signals:** PCLK (clock), PRESETn (reset), PADDR (address), PWDATA (write data), PRDATA (read data), PWRITE (transfer direction), PSELx (slave select), PENABLE (indicates ACCESS phase), PREADY (slave ready, allows wait states), PSLVERR (slave error response).37 APB version 2.0 (as per IHI0024C) adds PPROT for protection attributes and PSTRB for write strobes, enabling sparse byte writes.39
* **Use Cases:** Connecting low-bandwidth peripherals such as UARTs, timers, GPIO controllers, and interrupt controllers. Typically, an APB bridge connects an APB domain to a higher-performance bus like AHB or AXI.22

#### **4.1.4. CHI (Coherent Hub Interface) - (IHI0050E** 1**)**

CHI is Arm's most advanced on-chip interconnect protocol, part of AMBA 5, designed for high-performance, highly scalable, and coherent multi-core/multi-cluster systems.1

* **Layers:** CHI has a layered architecture: a Protocol Layer (transaction-level semantics, cache state transitions, flow control), a Network Layer (packetization, source/target ID assignment for routing), and a Link Layer (flit-level flow control, deadlock-free channel management).27
* **Node Types:** Defines various node types: Request Nodes (RN-F for fully coherent, RN-D/RN-I for I/O coherent), Interconnect Nodes (Home Nodes HN-F/HN-I for managing coherence and serialization, Miscellaneous Node MN for DVM), and Slave Nodes (SN-F/SN-I for memory/peripherals).27
* **Channels:** Communication occurs over dedicated point-to-point channels: Request (REQ), Response (RSP - further divided into Completer Response CRSP and Snoop Response SRSP), Data (DAT - further divided into Read Data RDAT and Write Data WDAT), and Snoop (SNP).27
* **Transaction Types:** Supports an extensive set of transactions for both coherent operations (e.g., ReadShared, ReadUnique, WriteBack, CleanInvalid, AtomicCompare) and non-coherent operations (e.g., ReadNoSnp, WriteNoSnp). It also includes Distributed Virtual Memory (DVM) operations for TLB maintenance and other system-level synchronization tasks.27
* **Cache State Model:** Employs a detailed cache state model (e.g., a seven-state model including Unique Clean/Dirty, Shared Clean/Dirty, Invalid) to manage coherency across multiple caching agents.27
* **Scalability & Topologies:** CHI is designed for high scalability, supporting various interconnect topologies like crossbars, rings, and meshes. It includes mechanisms like snoop filters and directories to optimize coherency management in large systems.22
* **Use Cases:** Targeted at high-end SoCs for servers, networking, automotive, and AI, connecting multiple processor clusters, coherent accelerators, and memory subsystems where maintaining data consistency at high performance is paramount.1

### **4.2. Wishbone (B4 Specification** 9**)**

Wishbone is an open-source hardware computer bus specification intended for IP core interconnection within an SoC. It emphasizes simplicity and flexibility.

* **Nature:** Wishbone is defined as a "logic bus," specifying interfaces in terms of signals, clock cycles, and logic levels, rather than electrical characteristics or a fixed bus topology.9 This makes it adaptable to various implementation technologies.
* **Transaction Types:** Supports basic single READ and WRITE cycles, BLOCK transfer cycles for moving multiple data words, and Read-Modify-Write (RMW) cycles for indivisible atomic operations.9
* **Key Signals:** The interface uses a synchronous, single-clock handshake. Core signals include CLK\_I (clock input), RST\_I (reset input), ADR\_O/I (address bus), DAT\_O/I (data bus), WE\_O/I (write enable), SEL\_O/I (byte/word select strobes), STB\_O/I (strobe/chip select), ACK\_O/I (acknowledge), and CYC\_O/I (cycle in progress).61 Optional signals include LOCK\_O/I (locked cycle), ERR\_O/I (error termination), RTY\_O/I (retry termination), and user-defined tag signals (TGA\_O/I for address tags, TGD\_O/I for data tags, TGC\_O/I for cycle tags).61
* **Configurability:** Highly configurable features include data bus widths (8, 16, 32, 64-bit, and extensible beyond), operand sizes, address bus widths (up to 64-bit), and data ordering (both big-endian and little-endian are supported).9 The optional tag bus allows for additional user-defined information to be passed alongside address, data, or cycle signals.
* **Topologies:** Wishbone is adaptable to various interconnection topologies, including point-to-point, shared bus (requiring an arbiter for multiple masters), crossbar switch, and even switched fabric or data flow interconnections.9
* **Use Cases:** Popular for IP core integration in FPGAs and ASICs, especially within the open-source hardware community (e.g., OpenCores projects).9 Its simplicity and flexibility make it suitable for a wide range of custom logic and peripheral integration.
* **Cycle Termination:** Wishbone bus cycles can terminate in one of three ways, signaled by the slave: normal termination (ACK), retry termination (RTY), or error termination (ERR).63

### **4.3. TileLink (Specification 1.8.0** 10**)**

TileLink is an open standard, chip-scale interconnect protocol, initially developed by SiFive and now part of the CHIPS Alliance portfolio. It is designed for connecting processors, caches, accelerators, DMA engines, and peripherals within an SoC, with a strong emphasis on cache coherency and scalability.10

* **Nature:** TileLink is a packet-based protocol that provides physically addressed, shared-memory access. It is designed for verifiable deadlock freedom when conforming to its rules.10
* **Channels:** It defines five logically independent, unidirectional channels between a master (Client) and a slave (Manager/Agent). These channels have a strict priority order (A << B << C << D << E, from lowest to highest priority) to prevent deadlocks 10:
  + Channel A (Acquire/Arithmetic/Logical/Hint/Get/Put): Master to Slave requests.
  + Channel B (Probe): Slave to Master snoops or permission revocations.
  + Channel C (Release/ProbeAck): Master to Slave responses to Probes or voluntary data writebacks/permission downgrades.
  + Channel D (Grant/AccessAck/HintAck): Slave to Master responses to Channel A requests or data.
  + Channel E (GrantAck): Master to Slave final acknowledgment for certain transactions.
* **Conformance Levels & Transaction Types:** TileLink specifies three main conformance levels 10:
  + **TL-UL (Uncached Lightweight):** Supports basic uncached memory operations like Get (read), PutFullData (full-width write), PutPartialData (partial write with byte mask), atomic operations (ArithmeticData, LogicalData), and Hint operations (e.g., prefetch).
  + **TL-UH (Uncached Heavyweight):** Extends TL-UL with additional features, though specific additional transactions over UL are detailed in the full specification.
  + **TL-C (Cached/Coherent):** Adds full cache coherency support using a MOESI-equivalent protocol. Introduces transactions like AcquireBlock (request data with intent to cache), AcquirePerm (request permissions upgrade), Probe (snoop from manager), ProbeAck/ProbeAckData (response to probe, possibly with data), Release/ReleaseData (voluntary writeback of dirty data or permission downgrade), Grant/GrantData (response from manager to acquire, possibly with data), and GrantAck (final acknowledgment from client).10
* **Key Features:** Packet-based communication where messages are composed of beats. Supports out-of-order completion of concurrent operations. Employs a VALID/READY handshake on each channel. Designed for hierarchical composability and verifiable deadlock freedom based on DAG topology and channel prioritization.10
* **Configurability:** TileLink is highly configurable. Parameters include address width, data width, source ID width (identifying master-side transaction sources), and sink ID width (used for routing responses in some configurations). The presence and data-carrying capability of coherency channels (B, C, E and data on A, D) can be configured.64 A notable feature is the use of "Diplomacy," a Scala-based framework in the Rocket Chip generator, for negotiating and propagating these parameters throughout the interconnect graph, ensuring consistency and enabling specialization of interfaces.47
* **Use Cases:** TileLink is predominantly used with RISC-V cores and SoCs generated by frameworks like Rocket Chip and Chipyard. It is suitable for connecting a wide range of components, from simple peripherals (using TL-UL) to complex, coherent multi-core processor systems (using TL-C).10

The choice of a bus standard involves weighing performance and features against complexity and ecosystem maturity. Open standards like Wishbone and TileLink offer flexibility and cost advantages, particularly attractive for the RISC-V community. However, they may present a steeper learning curve or have a less extensive pre-verified IP and tool ecosystem compared to the de facto industry standard, AMBA AXI, which is backed by Arm's vast market presence. TileLink's "Diplomacy" for parameter negotiation represents an advanced approach to managing the configuration complexity of modern, highly parameterized interconnects, potentially offering benefits in terms of correctness by construction and optimized specialization for complex heterogeneous SoCs. This feature is particularly relevant when aiming for a "general-purpose" architecture that must adapt to many different IP blocks.

## **5. Comparative Analysis of Bus Standards for Heterogeneous Integration**

Selecting an appropriate on-chip bus standard, or a combination thereof, is a pivotal decision in designing a general-purpose architecture for heterogeneous SoCs. This section provides a critical comparison of AMBA AXI/CHI, Wishbone, and TileLink across several key architectural criteria. The aim is to identify the strengths and weaknesses of each standard in the context of integrating PowerPC, Arm, TriCore, and RISC-V processors. A clear trade-off emerges between the feature-richness and performance of protocols like AMBA CHI and AXI, and their inherent complexity, versus the simplicity or novel configurability of open standards like Wishbone and TileLink.

### **5.1. Performance, Bandwidth, and Latency Characteristics**

* **AMBA AXI/CHI:** AXI is engineered for high throughput, featuring separate address and data phases, deep pipelining, support for multiple outstanding transactions, and out-of-order completion, which collectively help in maximizing bus utilization and reducing effective latency.22 AMBA CHI builds upon this, specifically targeting highly scalable, low-latency coherent systems with advanced mechanisms for congestion management and optimized data pathways.22 Both support wide data buses.
* **Wishbone:** Wishbone's performance is highly dependent on the implemented topology (e.g., point-to-point, shared bus, crossbar) and the specific master/slave interface logic.9 Its simpler handshake protocol can be very efficient for direct connections. However, in shared bus configurations with multiple masters, arbitration can become a bottleneck, and it lacks the sophisticated pipelining and out-of-order capabilities of AXI, potentially limiting peak bandwidth and increasing latency under heavy load.
* **TileLink:** TileLink is also designed for high throughput and low latency, incorporating features like out-of-order completion and packet-based communication.10 Its five-channel structure allows for concurrent operations. Performance characteristics vary with the conformance level; TL-C, with its coherency traffic, will have different performance dynamics than the simpler TL-UL. The explicit channel prioritization is designed to prevent deadlocks and ensure forward progress.

In terms of raw potential, AXI and particularly CHI are designed to push the envelope for performance in complex SoCs. TileLink aims for comparable performance with a different architectural approach. Wishbone can be performant in specific configurations but may not scale as effectively for very high-bandwidth, multi-master scenarios without a sophisticated interconnect fabric built around it.

### **5.2. Complexity, Area Overhead, and Power Implications**

* **AMBA AXI/CHI:** These are the most complex protocols among those considered. AXI's five channels, numerous signals, and support for advanced features like out-of-order responses and burst types contribute to significant logic complexity in both master and slave interfaces, leading to larger area and potentially higher power consumption.22 CHI, with its comprehensive coherency management and layered architecture, is even more complex.27 Simpler variants like AXI-Lite and AMBA APB exist for less demanding components, with APB being the lowest in complexity, area, and power.37 AHB offers a middle ground.22
* **Wishbone:** Generally, Wishbone is considered simpler, especially for basic point-to-point or shared bus implementations, resulting in a lower gate count and potentially lower power for the interface logic itself.61 However, implementing complex topologies like crossbars or managing advanced features like user-defined tags can add to its complexity.
* **TileLink:** The complexity of TileLink varies with its conformance level. TL-UL is designed to be lightweight and relatively simple to implement.10 TL-C, which includes full cache coherency, is significantly more complex due to the state machines and logic required to manage coherent transactions across its five channels.

A general-purpose architecture might need to offer different "classes" of interface or adopt a hierarchical approach to avoid imposing the overhead of a high-end bus protocol on simpler components that do not require its full feature set.

### **5.3. Scalability and Modularity**

* **AMBA AXI/CHI:** AXI is inherently scalable when used as the protocol for links within a Network-on-Chip (NoC). AMBA CHI is explicitly designed for building large, scalable coherent systems with many tens or hundreds of agents.22 The AMBA family promotes modular design by providing standardized interfaces for IP blocks.
* **Wishbone:** Wishbone supports various scalable topologies, including shared buses (with arbitration), crossbar switches, and more complex switched fabrics, allowing designers to choose an appropriate structure based on system size and performance needs.9 Its modularity stems from the IP core concept with a standardized Wishbone interface.
* **TileLink:** TileLink is designed with scalability in mind, particularly through its support for Directed Acyclic Graph (DAG) topologies and hierarchical composition facilitated by frameworks like Diplomacy.10 This allows for the construction of complex systems from smaller, composable TileLink agents and interconnects.

All three standards offer paths to scalability, though the mechanisms and optimal use cases differ. CHI and TileLink (with Diplomacy) offer more explicit architectural support for constructing large, complex, and coherent systems.

### **5.4. Cache Coherency Mechanisms and Support**

* **AMBA AXI/CHI:** Base AXI does not provide cache coherency. Arm introduced AXI Coherency Extensions (ACE) and ACE-Lite to enable hardware-managed cache coherency for AXI-based systems.22 AMBA CHI is a protocol fundamentally designed for high-performance cache coherency, supporting sophisticated cache state models and transaction flows to ensure data consistency across multiple processors and caching agents.1
* **Wishbone:** The base Wishbone specification does not define an integrated cache coherency protocol. Achieving cache coherency in a Wishbone-based multi-master system would require implementing a higher-level protocol on top of Wishbone or using custom extensions, which could impact interoperability.
* **TileLink:** The TL-C conformance level of TileLink provides a full, hardware-managed cache coherency protocol, often described as MOESI-equivalent.10 It uses its five channels to manage cache states, snoop transactions, and data transfers necessary for maintaining coherency.

For any general-purpose architecture intended for modern multi-processor SoCs, robust and hardware-managed cache coherency is non-negotiable. AMBA CHI and TileLink-C offer comprehensive solutions in this regard, while AXI relies on ACE. Wishbone would require significant additional effort to support system-wide coherency.

### **5.5. Ecosystem, IP Availability, and Industry Adoption**

* **AMBA AXI/AHB/APB:** The AMBA suite, particularly AXI, boasts the largest and most mature ecosystem. There is vast availability of third-party IP cores with AMBA interfaces, extensive tool support from EDA vendors, and widespread industry adoption across numerous market segments.22 This significantly reduces integration risk and development time. Verification IP for AMBA protocols is also widely available.
* **Wishbone:** Wishbone has a strong presence in the open-source hardware community, with many free cores available via platforms like OpenCores.9 It is also popular in FPGA-based designs. However, the availability of commercially supported and verified Wishbone IP is less extensive compared to AMBA.
* **TileLink:** The TileLink ecosystem is rapidly growing, primarily driven by the RISC-V movement and companies like SiFive, as well as initiatives like the CHIPS Alliance.13 SiFive provides RISC-V cores with native TileLink interfaces.52 Verification IP for TileLink is also becoming available.49 While promising, its ecosystem is not yet as broad or mature as AMBA AXI's.

The maturity and breadth of the ecosystem are critical factors. A rich ecosystem translates to readily available IP, proven verification solutions, and a larger pool of experienced engineers.

### **5.6. Licensing Models and Openness**

* **AMBA:** AMBA specifications are developed and controlled by Arm. While the specifications themselves are available royalty-free for implementation, Arm retains ownership and dictates their evolution.26 This provides stability and a single point of reference but less community-driven flexibility.
* **Wishbone:** Wishbone is a truly open-source specification, effectively in the public domain.9 This allows for maximum freedom in terms of use, modification, and redistribution without any licensing fees or restrictions.
* **TileLink:** TileLink is an open standard, with its specification originally from SiFive and now managed under the CHIPS Alliance.10 The hardware IP generators provided by CHIPS Alliance are typically under permissive open-source licenses like Apache 2.0.48 This model encourages collaborative development and broad adoption.

The definition of "open" varies. AMBA's openness pertains to the ability to implement the specification, whereas Wishbone and TileLink are open in terms of their development and governance model (to varying degrees). This has implications for long-term evolution, community support, and the potential for vendor influence or fragmentation. A more community-driven standard might offer greater adaptability to the future needs of diverse processor ISAs.

### **Table 2: Comparative Summary of On-Chip Bus Standards**

| **Feature** | **AMBA AXI** | **AMBA CHI** | **Wishbone** | **TileLink** |
| --- | --- | --- | --- | --- |
| **Max Performance** | Very High | Extremely High | Medium to High (Topology Dependent) | High to Very High |
| **Coherency Support** | Via ACE/ACE-Lite 22 | Native, Full Hardware Coherency 22 | None Natively (Requires Custom Extensions) | TL-C for Full Hardware Coherency (MOESI) 10 |
| **Complexity** | High | Very High | Low to Medium | Medium (TL-UL) to High (TL-C) |
| **Licensing** | Arm Specification (Implementation Royalty-Free) 56 | Arm Specification (Implementation Royalty-Free) 27 | Open Source/Public Domain 9 | Open Standard (CHIPS Alliance/SiFive) 48 |
| **Key Strengths** | Ecosystem, Perf., Wide Adoption | Coherency, Scalability, Perf. | Simplicity, Flexibility, Openness | Coherency, Scalability, Openness, Diplomacy 65 |
| **Key Weaknesses** | Complexity, Arm Controlled | Highest Complexity, Arm Controlled | Limited Native Coherency, Smaller Comm. Ecosystem | Newer Ecosystem vs. AXI |
| **Typical Data Widths (bits)** | 8-1024+ 25 | Configurable (typically 64-512+) | 8-64+ (Extensible) 9 | Configurable (e.g., 32-1024) 49 |
| **Typical Use Cases** | CPU-Mem, High-BW Peripherals, NoC links 22 | High-End Coherent Multi-Core Systems 1 | FPGA IP, Open-Source SoC, Custom Logic 9 | RISC-V SoCs, Coherent Systems, Accelerators 10 |
| **Ecosystem Maturity** | Very Mature | Growing (High-End) | Moderate (Open-Source Focus) | Growing Rapidly (RISC-V Focus) |

This comparative analysis underscores that the choice of a bus standard is a multi-faceted decision, balancing technical capabilities with strategic considerations like IP availability and development resources. For a general-purpose architecture targeting diverse ISAs, a protocol's inherent support for coherency, its scalability, and the ease with which it can be bridged to other standards are paramount.

## **6. Strategies for a General-Purpose Heterogeneous Bus Architecture**

Defining a truly general-purpose bus architecture to seamlessly integrate PowerPC, Arm, TriCore, and RISC-V processors requires a strategic approach. Given the diversity in their native bus interfaces and specific requirements (e.g., real-time for TriCore, high performance for Arm/PowerPC, flexibility for RISC-V), a single, monolithic bus standard might not be optimal for all components. This section explores practical architectural strategies, including adopting a dominant standard as a backbone, employing a hierarchical and bridged architecture, and leveraging Network-on-Chip (NoC) concepts. The "best" strategy will likely depend on the anticipated mix of processor types and the performance demands of future SoCs. A critical factor in any approach will be the verification effort, especially when bridging protocols and managing coherency across disparate domains.14

### **6.1. Option 1: Adopting a Single, Dominant Standard as the Backbone**

This strategy involves selecting one powerful and flexible bus standard to serve as the primary interconnect for the entire SoC. All major processing elements and high-bandwidth peripherals would interface directly with this backbone, while other components might connect via simpler, bridged sub-domains.

#### **6.1.1. AMBA AXI/CHI as the Primary Interconnect**

* **Rationale:** The AMBA AXI protocol, particularly when augmented with AXI Coherency Extensions (ACE) or superseded by the Coherent Hub Interface (CHI), offers a compelling case as a backbone.1 Its strengths include a vast and mature IP ecosystem, proven high performance, robust and well-verified coherency mechanisms (ACE for AXI, native in CHI), and increasing adoption or interface availability even for non-Arm ISAs like RISC-V and some modern PowerPC implementations.
* **Integration Approach:** Arm processors and many RISC-V cores could connect natively to an AXI/CHI backbone. PowerPC cores using CoreConnect or CoreNet, and TriCore processors with their SRI/FPI interfaces, would necessitate the development or sourcing of high-quality bridges to AXI/CHI. Peripherals could connect to AXI directly, or to AHB/APB sub-domains bridged from the AXI backbone.
* **Challenges:** The primary challenge is the inherent complexity of AXI and especially CHI, which might be an overkill for simpler processing elements or peripherals if they are forced to implement a full AXI/CHI interface. While ACE/CHI offer strong coherency, their licensing and use in deeply heterogeneous systems with non-Arm masters participating in full coherency need careful consideration, although CHI is generally positioned as more open for such heterogeneous coherent systems.1

#### **6.1.2. TileLink as the Primary Interconnect**

* **Rationale:** TileLink, particularly its coherent variant TL-C, presents an attractive open standard alternative.10 Its key advantages include its royalty-free nature, strong built-in support for cache coherency, design for deadlock freedom, and the sophisticated parameter negotiation capabilities offered by frameworks like Diplomacy (used in Rocket Chip/Chipyard). This can lead to highly specialized and optimized interconnects.
* **Integration Approach:** RISC-V cores designed with TileLink interfaces would connect natively. Processors from Arm, PowerPC, and TriCore ecosystems, which typically do not have native TileLink interfaces, would require bridges. The availability and maturity of such bridges (e.g., AXI-to-TileLink, PLB-to-TileLink) are critical. Existing open-source efforts include TileLink-to-AHB bridges 46 and discussions around TileLink-to-AXI4 bridges.65
* **Challenges:** The TileLink IP ecosystem, while growing rapidly alongside RISC-V, is not yet as extensive as AMBA AXI's. The development and verification of robust bridges from established proprietary bus protocols to TileLink would be a significant undertaking. Ensuring broad industry support and availability of verification IP for a TileLink-centric heterogeneous architecture is also a consideration.

### **6.2. Option 2: Hierarchical and Bridged Architecture**

This approach acknowledges the diversity of native interfaces and performance requirements by employing a high-performance backbone bus for critical inter-processor and memory communication, while allowing other components or sub-systems to reside on their native or simpler buses connected via bridges.6

* **Concept:** A high-performance coherent bus like AMBA CHI, AXI with ACE, or TileLink-C would serve as the main SoC backbone. Slower or specialized bus domains, such as AMBA AHB/APB for Arm-centric peripherals, CoreConnect OPB for legacy PowerPC peripherals, TriCore FPI/SPB for its specific peripherals, or Wishbone for certain open-source IPs, would be bridged to this backbone.
* **Advantages:** This strategy optimizes the bus choice for specific components, preventing the performance or complexity overhead of the main backbone from being imposed on simpler IPs. It maximizes the reuse of existing IP cores with their native interfaces, potentially reducing porting efforts. It also allows for clear separation of clock and power domains.

#### **6.2.1. Bus Bridging: Key Techniques and Challenges**

Bus bridges are critical components in a hierarchical architecture, acting as a slave on one bus segment and a master on another.6 Their design involves several complex considerations:

* **Protocol Translation:** This is the core function, involving the conversion of transaction semantics, signal mappings, and handshake mechanisms between the two connected bus protocols.6 For example, an AXI master's burst write must be correctly translated into a sequence of Wishbone single writes or a block write, managing signals like AWVALID/AWREADY, WVALID/WREADY, BVALID/BREADY on the AXI side and CYC, STB, WE, ACK on the Wishbone side.57 Similar translations are needed for TileLink-to-AHB 46 or OCP-to-AHB.59
* **Handling Clock Domain Crossing (CDC):** If the bridged buses operate in different or asynchronous clock domains, robust CDC mechanisms are essential. These typically involve asynchronous FIFOs for data buffering and multi-flop synchronizers for control signals to prevent metastability, data loss, or incoherency.69 CDC logic is a major source of design complexity and a primary focus for verification.
* **Data Width Adaptation:** Bridges must handle discrepancies in data bus widths (e.g., a 64-bit AXI backbone to a 32-bit AHB peripheral bus). This requires logic for multiplexing, de-multiplexing, and potentially buffering data to pack/unpack transactions correctly.58
* **Burst-to-Single/Block Transaction Conversion:** High-performance protocols like AXI and TileLink heavily rely on burst transactions. When bridging to protocols that primarily support single transfers (or simpler block transfers like Wishbone), the bridge must decompose bursts from the master side and potentially aggregate single transfers from the slave side if the master expects a burst response.57
* **Managing Transaction Ordering and IDs:** Protocols like AXI and TileLink support multiple outstanding transactions and out-of-order responses using transaction IDs. Simpler protocols like AHB or Wishbone often assume in-order processing. Bridges must manage these differences, potentially by enforcing in-order processing on the simpler bus, buffering and reordering responses, or limiting the number of outstanding transactions allowed through the bridge.57
* **Maintaining Cache Coherency Across Bridges:** This is arguably the most complex aspect of bridging in a heterogeneous, multi-processor SoC.
  + *Non-Coherent Bridges:* These are simpler to design as they do not participate in coherency protocols. However, they effectively create separate coherency domains. Data shared across such a bridge must be managed by software (e.g., cache flushing/invalidation), or the memory regions accessed through the bridge must be marked as non-cacheable or I/O coherent, restricting true hardware-managed sharing.4
  + *Coherent Bridges:* These bridges must understand and participate in the coherency protocols of one or both connected buses. This might involve snooping transactions, translating coherency messages (e.g., between AMBA ACE/CHI and TileLink-C), forwarding snoop requests, and managing cache state updates. Proxy caches within the bridge can sometimes be used to allow non-coherent masters to participate in I/O coherency with a coherent domain.5 Designing and verifying coherent bridges is a significant challenge due to the complexity of modern coherency protocols and the potential for subtle race conditions or deadlocks.74

### **6.3. Option 3: Network-on-Chip (NoC) Considerations**

For highly complex SoCs with a large number of diverse processing elements, memory resources, and peripherals, a Network-on-Chip (NoC) architecture may offer the most scalable and flexible solution.1

* **Principles:** NoCs replace traditional shared buses or simple crossbars with a packet-switched network consisting of routers, links, and network interfaces (NIs). This allows for concurrent communication paths and can provide better aggregate bandwidth and latency characteristics for large systems. Common topologies include mesh, torus, and ring.
* **Protocols over NoC:** The links and routers within a NoC typically implement a link-level flow control protocol. The transactions carried over the NoC often adhere to standard bus protocols like AXI, CHI, or TileLink at the NI level. For instance, Arteris Ncore uses AMBA ACE/CHI as part of its coherent NoC solution 4, and the Chipyard framework often employs TileLink-based NoCs.2
* **Advantages for Heterogeneity:** NoCs can naturally accommodate a diverse set of IP blocks, each with its own NI tailored to its specific interface protocol and performance requirements. They allow for the creation of distinct coherent and non-coherent domains within the same fabric, potentially routing traffic through different virtual channels or sub-networks.4
* **Challenges:** NoC design introduces its own complexities, including topology selection, routing algorithm design, flow control, and ensuring Quality of Service (QoS). Latency for communication between distant nodes can be higher than in a direct-connect or small crossbar system. The power consumption of the NoC fabric itself (routers and links) can also be significant.

A hierarchical/bridged architecture, while potentially complex to design and verify initially, offers significant adaptability. It allows for the integration of legacy IP with minimal changes and provides a pathway for incorporating new processor types or evolving bus standards in the future by developing new bridge components. The success of this strategy hinges on the availability or development of high-quality, configurable, and thoroughly verified bridge IPs.

### **Table 3: Evaluation of General-Purpose Architecture Strategies**

| **Strategy** | **Key Pros** | **Key Cons** | **Suitability for PowerPC Integration** | **Suitability for Arm Integration** | **Suitability for TriCore Integration** | **Suitability for RISC-V Integration** | **Coherency Management Complexity** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Single Backbone: AMBA AXI/CHI** | Mature ecosystem, strong coherency (CHI/ACE), high performance, wide Arm & RISC-V adoption. | High complexity for simple IPs, Arm-centric coherency (licensing/use concerns for non-Arm might arise). | Medium (Requires PLB/CoreNet to AXI/CHI bridge) | High (Native) | Medium (Requires SRI/FPI to AXI/CHI bridge) | High (Many IPs offer AXI) | Medium (ACE) to High (CHI) |
| **Single Backbone: TileLink** | Open standard, strong coherency (TL-C), RISC-V native, advanced parameterization (Diplomacy). | Smaller ecosystem than AXI, bridge maturity for non-RISC-V ISAs is a concern. | Low (Requires PLB/CoreNet to TL bridge) | Medium (Requires AXI to TL bridge) | Low (Requires SRI/FPI to TL bridge) | High (Native for many) | High (TL-C) |
| **Hierarchical/Bridged (AXI/CHI Backbone)** | Balances performance & IP reuse, leverages AXI/CHI ecosystem, optimizes bus for component needs. | Bridge complexity (design & verification), potential bridge latency, coherency across bridges is challenging. | High (Bridge to PLB/OPB/CoreNet) | High (Native backbone, bridges to AHB/APB) | High (Bridge to SRI/FPI) | High (Bridge to TileLink/Wishbone) | High (Backbone + Coherent Bridges) |
| **Hierarchical/Bridged (TileLink Backbone)** | Leverages TileLink openness & coherency, good for RISC-V heavy systems, optimizes bus for components. | Bridge complexity (especially AXI to TL), smaller backbone ecosystem than AXI. | Medium (Bridge to PLB/OPB/CoreNet) | Medium (Bridge to AXI) | Medium (Bridge to SRI/FPI) | High (Native backbone) | High (Backbone + Coherent Bridges) |
| **Full Network-on-Chip (NoC) Approach** | Max scalability, flexible topology, can create distinct coherent/non-coherent zones, good for many IPs. | Highest design complexity, latency for distant nodes, NoC fabric power, NI design per IP. | Medium to High (via NoC NI) | Medium to High (via NoC NI) | Medium to High (via NoC NI) | Medium to High (via NoC NI) | Very High (Coherent NoC fabric) |

This evaluation suggests that for a system aiming for broad generality across these four ISAs, a hierarchical architecture with a well-supported and coherent backbone (like AMBA AXI/CHI) offers a pragmatic balance. It leverages a mature ecosystem while providing pathways to integrate diverse IP through specialized bridges. The complexity and verification of these bridges, especially those handling coherency, remain the most significant challenges.

## **7. Key Architectural Considerations for the General-Purpose Bus**

Beyond selecting a primary bus standard or a hierarchical strategy, several specific technical aspects must be meticulously addressed to ensure the functionality, performance, and reliability of a general-purpose bus architecture designed for heterogeneous SoCs. These considerations often involve managing differences inherent in the connected ISAs and their peripherals.

### **7.1. Data Width and Endianness Management**

* **Data Width:** The bus architecture must accommodate a range of data widths, as different processors and peripherals operate on varying data sizes (e.g., 8-bit, 16-bit, 32-bit, 64-bit, and potentially wider for specialized units like GPUs or AI accelerators). Standard protocols like AXI, AHB, and Wishbone support configurable data widths.58 Bridges connecting buses of different widths must perform appropriate data packing, unpacking, and alignment. For instance, a bridge connecting a 64-bit AXI master to a 32-bit AHB slave needs to split 64-bit AXI transactions into two 32-bit AHB transactions and merge data for reads.58
* **Endianness:** Processors like PowerPC are traditionally big-endian, while RISC-V and most Arm implementations are little-endian.12 TriCore endianness can be implementation-specific. When these processors share data structures in memory, a consistent view of byte ordering is critical. The bus architecture or the bridges must manage endianness conversion if masters and slaves (or different masters sharing memory) have different endian formats.21 This can be handled by configurable endpoints that can adapt to the connected device's endianness or by bridges that perform byte-swapping. If not managed at the hardware level, this becomes a significant software burden and a common source of subtle bugs. A clear architectural strategy for endianness, defining a "system endianness" or providing robust conversion mechanisms, is paramount.

### **7.2. Arbitration and Quality of Service (QoS)**

* **Arbitration:** In any system with multiple bus masters contending for shared resources (slaves or bus segments), an arbitration mechanism is essential. Common schemes include fixed-priority (where masters have predefined priorities) and round-robin (which provides fairer access).34 The choice of arbitration scheme can significantly impact system performance and fairness. Centralized arbiters are common in simpler bus structures, while distributed arbitration might be used in NoCs or more complex fabrics.
* **Quality of Service (QoS):** For systems with real-time constraints (e.g., involving TriCore processors) or diverse performance requirements, simple priority-based arbitration may be insufficient. QoS mechanisms aim to provide differentiated service levels, potentially guaranteeing minimum bandwidth or maximum latency for critical transactions. Protocols like AXI include QoS signaling (e.g., AxQOS signals, though not detailed in all provided snippets, the AXI specification 24 mentions them, and the RISC-V community is defining specifications like the Capacity and Bandwidth QoS Register Interface (CBQRI) 55). Effective QoS in a heterogeneous system with multiple bus domains and bridges is complex. Bridges must be able to propagate or translate QoS requirements, or the overall system QoS guarantees may be compromised. Advanced NoCs often incorporate more sophisticated QoS features within their routing and arbitration logic.

### **7.3. Error Handling, Debug, and Trace**

* **Error Reporting:** The bus architecture must provide a consistent way for slaves to report errors during transactions (e.g., unsupported access, parity error). Protocols like AXI (xRESP signals), AHB (HRESP), APB (PSLVERR), and Wishbone (ERR\_I) have defined error responses.22 Bridges must correctly propagate or translate these error conditions to ensure the originating master is appropriately notified. The system's response to such errors (e.g., generating an interrupt, retrying the transaction) also needs to be defined.
* **Debug & Trace:** Effective debug and trace capabilities are crucial for SoC development and post-silicon validation. The bus architecture should facilitate visibility into bus transactions. This includes compatibility with standard on-chip debug architectures like Arm CoreSight, which uses the Advanced Trace Bus (ATB) to convey trace information.22 TriCore MCUs often feature a Multi-Core Debug Solution (MCDS) capable of tracing on-chip bus transfers.41 RISC-V also has a defined debug specification.55 A general-purpose architecture needs to accommodate or provide pathways for these diverse debug and trace streams, potentially requiring a unified debug fabric or intelligent trace funnels.

### **7.4. Security Considerations**

Modern SoCs often incorporate hardware security features to protect assets and ensure system integrity. The bus architecture plays a role in enforcing security policies.

* **Privilege and Secure Access Control:** Many bus protocols include signals to indicate the privilege level (e.g., user/supervisor) and security state (e.g., secure/non-secure) of a transaction. Examples include AXI AxPROT, AHB HPROT, and APB PPROT signals.25 The bus fabric and peripherals must use this information to enforce access permissions.
* **Memory Protection:** The bus architecture must interoperate correctly with Memory Protection Units (MPUs) or Memory Management Units (MMUs) associated with the processors. Access permissions defined by the MPU/MMU should be respected by the bus fabric.
* **Isolation:** The architecture may need to support the isolation of transactions from different security domains, ensuring that non-secure masters cannot access secure resources, for example. This can be achieved through hardware firewalls or specific routing rules within the interconnect.

### **7.5. Clocking Strategy and Clock Domain Crossing (CDC)**

SoCs typically contain multiple clock domains, with different processors and peripherals operating at various frequencies to optimize power and performance.

* **Synchronous vs. Asynchronous Operation:** While individual bus segments are often synchronous, communication between segments operating in different clock domains must be handled asynchronously.
* **Clock Domain Crossing (CDC) Mechanisms:** Bridges connecting different clock domains are critical points that require robust CDC logic.6 Improper CDC handling is a common source of functional failures in SoCs, leading to issues like metastability, data loss, or data incoherency.69 Standard techniques include using dual-ported asynchronous FIFOs for data paths and carefully designed handshake synchronizers (e.g., multi-flop synchronizers) for control signals. The design and verification of CDC logic demand meticulous attention.

Addressing these architectural considerations comprehensively is vital for creating a general-purpose bus architecture that is not only functional but also performant, reliable, secure, and debuggable across the diverse range of targeted processor ISAs. The complexity introduced by heterogeneity, particularly in areas like endianness management, QoS, and coherent bridging across clock domains, requires careful architectural planning and rigorous verification.

## **8. Recommendations for a General-Purpose Bus Architecture**

Based on the comprehensive analysis of processor ecosystems, prominent on-chip bus standards, and key architectural considerations, this section provides concrete recommendations for defining a general-purpose bus architecture capable of supporting PowerPC, Arm, TriCore, and RISC-V based SoCs. The primary goal is to achieve a balance between performance, scalability, coherency, IP reuse, and manageable complexity.

### **8.1. Recommended Architectural Approach(es)**

The most pragmatic and versatile approach for a general-purpose heterogeneous bus architecture is a **Hierarchical and Bridged Architecture**.

* **Primary Recommendation: Hierarchical Architecture with AMBA AXI/CHI Backbone**
  + **Backbone Choice:** **AMBA AXI (specifically AXI5) augmented with AMBA CHI (Coherent Hub Interface) for coherent domains** is recommended as the primary SoC backbone.
    - **Rationale:** This choice is driven by AXI's mature and extensive ecosystem, widespread IP availability (including from Arm, RISC-V, and some PowerPC vendors), proven high performance, and robust, well-verified coherency solutions provided by CHI (or ACE for legacy AXI4 coherent domains).1 CHI, in particular, is designed for heterogeneous coherent systems and offers excellent scalability and performance.1
  + **Alternative Backbone (Context-Dependent): TileLink (TL-C)**
    - **Rationale:** If the SoC roadmap heavily emphasizes RISC-V based components, or if the advanced parameter negotiation capabilities of TileLink (via Diplomacy) are deemed highly beneficial for managing complexity and ensuring correctness-by-construction, then TileLink (specifically TL-C for coherency) could serve as the backbone.10
    - **Considerations:** This choice would necessitate a greater investment in developing or sourcing high-quality bridges to AMBA (for Arm IP) and proprietary interfaces (for PowerPC, TriCore), and a careful assessment of the TileLink IP and verification ecosystem maturity relative to project needs.
* **Bridging Strategy:** Regardless of the backbone choice, a robust bridging strategy is essential. This involves:
  + **Standardized Bridge Interfaces:** Define standardized interfaces on the backbone side of bridges to simplify their integration.
  + **High-Quality, Configurable Bridges:** Invest in the development or procurement of high-quality, configurable, and thoroughly verified bus bridges from the chosen backbone to:
    - PowerPC native interfaces (e.g., AXI-to-PLB, AXI-to-CoreNet wrapper).
    - TriCore native interfaces (e.g., AXI-to-SRI, AXI-to-FPI).
    - Other relevant standards if specific IPs require them (e.g., AXI-to-Wishbone for open-source IPs, or AXI-to-TileLink if both are used in different domains).
  + **Coherent Bridging:** For connecting coherent processor clusters or IPs that use different coherency protocols to the backbone, coherent bridges are necessary. These bridges must correctly translate or participate in coherency transactions. This is a complex area requiring significant design and verification effort. If CHI is the backbone, its inherent support for heterogeneous coherency can simplify this.
  + **Non-Coherent Bridging:** For peripherals or non-coherent masters, simpler non-coherent bridges (e.g., AXI-to-AHB, AXI-to-APB, AXI-to-OPB) should be used to connect to lower-performance bus domains.

The "general-purpose" characteristic is best realized not by a single protocol that natively fits all ISAs (which is impractical), but by a highly *configurable and bridgeable* backbone. The configurability of AXI/CHI (data/address widths, ID widths, QoS attributes, cacheability/security attributes) 25 and TileLink (via Diplomacy for extensive parameter negotiation) 64 is crucial for adapting the backbone itself. The efficiency and correctness of the bridges then determine how well other ISAs are integrated.

### **8.2. Justification**

* **Generality and Flexibility:** A hierarchical/bridged architecture offers the highest degree of generality. It allows each major processor subsystem or IP block to potentially reside on its optimal bus (native or otherwise), while still enabling system-wide communication via the backbone and bridges. This approach accommodates both legacy IP and new designs.
* **Performance Optimization:** A high-speed, coherent backbone (AXI/CHI or TL-C) can be dedicated to performance-critical communication paths, such as between main processors and memory, or between coherent clusters. Slower peripherals are relegated to dedicated, lower-speed bus segments, preventing them from loading down the main backbone.
* **Coherency Management:** Leveraging mature and robust coherency protocols like AMBA CHI or TileLink-C on the backbone is essential for modern multi-processor SoCs. The primary challenge then shifts to managing coherency across bridges connecting to other coherent domains or to I/O coherent masters.
* **Ecosystem Leverage and Risk Mitigation:** Choosing AMBA AXI/CHI as the primary backbone leverages the industry's largest IP and tool ecosystem, reducing development risk and time-to-market. Many RISC-V IPs are also available with AXI interfaces. If TileLink is chosen, it aligns well with the open-source RISC-V movement but requires a more focused effort on ecosystem development and bridging to non-RISC-V components.
* **IP Reuse:** This strategy explicitly supports the reuse of existing IP cores with their native interfaces by providing appropriate bridge solutions, rather than requiring a costly and time-consuming redesign of all IP to a single new standard.

### **8.3. Addressing Key Trade-offs**

The recommended approach comes with inherent trade-offs that must be managed:

* **Bridge Complexity:** Designing and verifying bridges, especially those that are coherent or cross significant clock domain boundaries, is a complex task.46 This requires specialized expertise and significant verification resources. The success of the entire architecture heavily depends on the quality of these bridges.
* **Performance Overhead of Bridges:** Bridges inevitably introduce some latency and potential bandwidth limitations compared to a direct native connection. This overhead must be carefully analyzed and minimized for critical communication paths.
* **Coherency Across Bridges:** Maintaining cache coherency across bridges connecting different coherency protocols or extending coherency to non-native domains is a major architectural challenge.4 Solutions range from software-managed coherency for simpler cases to complex hardware coherent bridges or proxy mechanisms.
* **Standardization Effort:** While reusing native IP interfaces is an advantage, there is still a need to standardize the interfaces on the backbone side of the bridges and to establish clear architectural rules for aspects like endianness, QoS propagation, and error handling across domains.

### **8.4. Phased Approach (Optional)**

For organizations embarking on such an architectural definition, a phased approach can be beneficial:

1. **Phase 1: Backbone Selection and Core Integration:** Select the primary backbone protocol (e.g., AXI5/CHI). Develop or acquire initial bridges for connecting the most critical processor ISAs planned for immediate use. Focus on robust non-coherent bridging first.
2. **Phase 2: Peripheral Integration and Basic Coherency:** Develop a comprehensive suite of bridges for common peripheral bus standards (e.g., AHB, APB, OPB, FPI). Implement basic I/O coherency for DMA masters and other key peripherals.
3. **Phase 3: Advanced Coherency and Optimization:** Develop and verify full coherent bridging solutions for connecting multiple, potentially heterogeneous, coherent processor clusters. Optimize bridge performance and explore advanced features like system-wide QoS.

This phased approach allows for incremental development, verification, and deployment, reducing risk and allowing the architecture to evolve with project needs.

## **9. Conclusion**

The endeavor to define a general-purpose on-chip bus architecture capable of supporting diverse processor ISAs such as PowerPC, Arm, TriCore, and RISC-V is a complex but essential undertaking for modern SoC design. The increasing heterogeneity of SoCs, driven by the need for specialized processing, demands an interconnect strategy that is both performant and flexible. This report has analyzed the native bus ecosystems of these ISAs, surveyed prominent on-chip bus standards, and evaluated strategies for their integration.

The key challenge lies in reconciling the varied native interface protocols, performance requirements, and coherency mechanisms of these distinct processor families. No single existing bus standard perfectly and natively caters to all four ISAs. Therefore, a "one-size-fits-all" bus protocol is impractical.

The recommended architectural strategy is a **Hierarchical and Bridged Architecture**, with **AMBA AXI5 and CHI** serving as the primary high-performance, coherent backbone. This approach leverages the maturity, extensive ecosystem, and robust coherency features of the AMBA standards. For RISC-V-centric designs or where advanced parameter negotiation is paramount, **TileLink-C** presents a viable open-standard alternative for the backbone. Crucially, this hierarchical strategy relies on the development and deployment of high-quality, configurable, and thoroughly verified **bus bridges** to connect processors and IP blocks with other native interfaces (e.g., PowerPC CoreConnect, TriCore SRI/FPI, or other standards like Wishbone) to the chosen backbone.

The success of such an architecture hinges on several critical factors:

* **Robust Coherency Management:** Ensuring data consistency across potentially different coherency domains connected via bridges is paramount and represents a significant technical challenge.
* **Bridge Design and Verification:** The bridges themselves are complex SoCs components. Their correctness, performance, and ability to handle aspects like clock domain crossing, data width adaptation, and protocol-specific transaction semantics are vital. Investing in their rigorous verification is non-negotiable.
* **Clear IP Integration Strategy:** A well-defined methodology for integrating IP cores, including guidelines for interface selection, endianness handling, QoS propagation, and error reporting, is necessary.

The path to a truly general-purpose on-chip bus architecture is an evolutionary one. It will likely involve ongoing refinement of bridge components, adaptation to new versions of bus standards, and continuous optimization based on the specific needs of emerging SoC designs. However, by adopting a well-reasoned hierarchical and bridged approach, SoC designers can achieve a versatile communication fabric. This provides a significant competitive advantage, enabling faster development of diverse and powerful SoCs, maximizing IP reuse, and allowing the flexibility to select the optimal processor ISA for any given task without being unduly constrained by bus interface limitations. Such an architecture will be a cornerstone for future innovation in the era of heterogeneous computing.

#### Works cited

1. Soft Tiling RISC-V Processor Clusters Speed Design and Reduce Risk, accessed May 19, 2025, <https://riscv.org/blog/2025/03/soft-tiling-risc-v-processor-clusters-speed-design-and-reduce-risk/>
2. NeCTAr: A Heterogeneous RISC-V SoC for Language Model Inference in Intel 16 - arXiv, accessed May 19, 2025, <https://arxiv.org/html/2503.14708v1>
3. arxiv.org, accessed May 19, 2025, <https://arxiv.org/pdf/2401.15639>
4. SoC design: When a network-on-chip meets cache coherency - EDN, accessed May 19, 2025, <https://www.edn.com/soc-design-when-a-network-on-chip-meets-cache-coherency/>
5. Cache Coherent Interconnect - Arteris, accessed May 19, 2025, <https://www.arteris.com/learn/cache-coherent-interconnect/>
6. EE382V: System-on-a-Chip (SoC) Design Lecture 12 - University of Texas at Austin, accessed May 19, 2025, <https://users.ece.utexas.edu/~gerstl/ee382v_f14/lectures/lecture_12.pdf>
7. A Survey of System on Chip and Network on Chip Architectures ..., accessed May 19, 2025, <https://www.researchgate.net/publication/275028360_A_Survey_of_System_on_Chip_and_Network_on_Chip_Architectures>
8. On-Chip Communication Architectures - ResearchGate, accessed May 19, 2025, <https://www.researchgate.net/publication/281654653_On-Chip_Communication_Architectures>
9. Wishbone (computer bus) - Wikipedia, accessed May 19, 2025, <https://en.wikipedia.org/wiki/Wishbone_(computer_bus)>
10. Deciphering the New TileLink Standard | Synopsys Blog, accessed May 19, 2025, <https://www.synopsys.com/blogs/chip-design/understanding-new-tilelink-standard.html>
11. RISC-V Architecture: A Comprehensive Guide to the Open-Source ISA - Wevolver, accessed May 19, 2025, <https://www.wevolver.com/article/risc-v-architecture>
12. RISC-V - Wikipedia, accessed May 19, 2025, <https://en.wikipedia.org/wiki/RISC-V>
13. CHIPS Alliance aims to ease RISC-V design and deployment - TechRepublic, accessed May 19, 2025, <https://www.techrepublic.com/article/chips-alliance-aims-to-ease-risc-v-design-and-deployment/>
14. Assuring the integrity of RISC-V cores and SoCs - Siemens Digital Industries Software, accessed May 19, 2025, <https://resources.sw.siemens.com/en-US/white-paper-assuring-the-integrity-of-risc-v-cores-and-socs/>
15. CoreConnect Bus and AMBA ... - ASIC-System on Chip-VLSI Design, accessed May 19, 2025, <https://asic-soc.blogspot.com/2009/04/coreconnect-bus-and-amba-bus.html>
16. espace.library.uq.edu.au, accessed May 19, 2025, <https://espace.library.uq.edu.au/view/UQ:100705/UQ100705_OA.pdf>
17. bitsavers.computerhistory.org, accessed May 19, 2025, <https://bitsavers.computerhistory.org/components/amcc/PPC405/PPC405EP_PB2004_v1_03.pdf>
18. Core Connect - Mirabilis Design, accessed May 19, 2025, <https://www.mirabilisdesign.com/core-connect/>
19. QorIQ® T1040 and T1020 Multicore Communications Processors ..., accessed May 19, 2025, <https://www.nxp.com/products/T1040>
20. QorIQ® P1020 | NXP Semiconductors, accessed May 19, 2025, <https://www.nxp.com/products/P1020>
21. PowerPC - Wikipedia, accessed May 19, 2025, <https://en.wikipedia.org/wiki/PowerPC>
22. Advanced Microcontroller Bus Architecture - Wikipedia, accessed May 19, 2025, <https://en.wikipedia.org/wiki/Advanced_Microcontroller_Bus_Architecture>
23. On-Chip Communication | SoC Labs, accessed May 19, 2025, <https://www.soclabs.org/technology/chip-communication>
24. AMBA AXI Protocol Specification - Arm, accessed May 19, 2025, <https://documentation-service.arm.com/static/5f915920f86e16515cdc3342>
25. 31409944.s21i.faiusr.com, accessed May 19, 2025, <https://31409944.s21i.faiusr.com/61/ABUIABA9GAAggIvWpwYohsKq9wc.pdf>
26. AMBA Specifications – Arm®, accessed May 19, 2025, <https://www.arm.com/architecture/system-architectures/amba/amba-specifications>
27. kolegite.com, accessed May 19, 2025, <https://kolegite.com/EE_library/datasheets_and_manuals/FPGA/AMBA/IHI0050E_a_amba_5_chi_architecture_spec.pdf>
28. Arm Cortex-M4 Processor Technical Reference Manual Revision r0p1, accessed May 19, 2025, <https://developer.arm.com/documentation/100166/latest/Functional-Description/Interfaces/Bus-interfaces>
29. AMBA AHB Protocol Overview | PDF | Computer Data - Scribd, accessed May 19, 2025, <https://www.scribd.com/document/692099731/AMBA-AHB-Protocol-Overview>
30. Ahb protocol pdf - GM Binder, accessed May 19, 2025, <https://www.gmbinder.com/share/-OMYAqO1a7UO4WRoUxJl>
31. Cortex-M4 Technical Reference Manual r0p0 - Arm Developer, accessed May 19, 2025, <https://developer.arm.com/documentation/ddi0439/b/Functional-Description/Interfaces/Bus-interfaces>
32. Documentation – Arm Developer, accessed May 19, 2025, <https://developer.arm.com/documentation/ddi0337/e/Introduction/Components--hierarchy--and-implementation/Bus-Matrix>
33. accessed December 31, 1969, <https://developer.arm.com/documentation/100166/latest/System-Level-Architecture/System-Level-Interfaces/Bus-interfaces>
34. documentation-service.arm.com, accessed May 19, 2025, [https://documentation-service.arm.com/static/6141bf0d674a052ae36ca811?token=](https://documentation-service.arm.com/static/6141bf0d674a052ae36ca811?token)
35. www.arm.com, accessed May 19, 2025, <https://www.arm.com/-/media/Arm%20Developer%20Community/PDF/Processor%20Datasheets/Arm%20Cortex-M3%20Processor%20Datasheet.pdf>
36. documentation-service.arm.com, accessed May 19, 2025, <https://documentation-service.arm.com/static/5f19da2a20b7cf4bc524d99a>
37. Amba-Apb Protocol | PDF | Electrical Engineering - Scribd, accessed May 19, 2025, <https://www.scribd.com/document/724023868/AMBA-APB-PROTOCOL>
38. Amba 2.0 specification pdf - GM Binder, accessed May 19, 2025, <https://www.gmbinder.com/share/-OIR8PxcFa4c_-qJoYIy>
39. www.eecs.umich.edu, accessed May 19, 2025, <https://www.eecs.umich.edu/courses/eecs373/readings/IHI0024C_amba_apb_protocol_spec.pdf>
40. Infineon AURIX TC3xx Family – Deep Dive - emmtrix Technologies, accessed May 19, 2025, <https://www.emmtrix.com/wiki/Infineon_AURIX_TC3xx>
41. TriCore - Lauterbach TRACE32 Debugger and Trace Solutions, accessed May 19, 2025, <https://www.lauterbach.com/supported-platforms/architectures/tricore>
42. TriCore - Infineon Technologies, accessed May 19, 2025, <https://www.infineon.com/dgdl/Tricore_prodbrief.pdf?fileId=db3a304312bae05f0112be88306e0114>
43. AURIX™ MCU: What is SRI crossbar - Infineon Developer Community, accessed May 19, 2025, <https://community.infineon.com/t5/Knowledge-Base-Articles/AURIX-MCU-What-is-SRI-crossbar/ta-p/336365>
44. www.infineon.com, accessed May 19, 2025, <https://www.infineon.com/dgdl/Infineon-AURIX_TC3xx_Architecture_vol1-UserManual-v01_00-EN.pdf?fileId=5546d46276fb756a01771bc4c2e33bdd>
45. Flexible Peripheral Interconnect (FPI) | Aurix TC3xx Documentation, accessed May 19, 2025, <https://documentation.infineon.com/aurixtc3xx/docs/tvz1715673354645>
46. Antmicro · Open source TileLink to AHB bridges with dedicated ..., accessed May 19, 2025, <https://antmicro.com/blog/2022/10/open-source-tl-to-ahb-bridges-with-cocotb/>
47. Chipyard Intro and Fundamentals - FireSim, accessed May 19, 2025, <https://fires.im/isca21-slides-pdf/02_chipyard_basics.pdf>
48. chipsalliance/tilelink - GitHub, accessed May 19, 2025, <https://github.com/chipsalliance/tilelink>
49. Accelerate RISC-V SoC Verification with TileLink IP | Synopsys Blog, accessed May 19, 2025, <https://www.synopsys.com/blogs/chip-design/tilelink-verification-ip-riscv-socs.html>
50. TileLink - Truechip Verification IP, accessed May 19, 2025, <https://www.truechip.net/details/tilelink/340013>
51. TileLink-1.8.0.pdf - GitHub, accessed May 19, 2025, <https://github.com/chipsalliance/omnixtend/blob/master/OmniXtend-1.0.3/spec/TileLink-1.8.0.pdf>
52. SiFive RISC-V Core IP Products - HubSpot, accessed May 19, 2025, <https://cdn2.hubspot.net/hubfs/3020607/SiFive-RISCVCoreIP.pdf>
53. SiFive RISC-V Core IP, accessed May 19, 2025, <https://sifive-china.oss-cn-zhangjiakou.aliyuncs.com/%E5%85%AC%E5%8F%B8%E5%AE%A3%E4%BC%A0%E5%86%8C/company%20brochure%202019%20v2.pdf>
54. RISC-V Technical Committes, accessed May 19, 2025, <https://riscv.org/developers/technical-committees/>
55. ISA Specifications - Home - RISC-V Tech Hub, accessed May 19, 2025, <https://lf-riscv.atlassian.net/wiki/display/HOME/RISC-V+Technical+Specifications>
56. AMBA® AXI Protocol Specification - Arm, accessed May 19, 2025, [https://documentation-service.arm.com/static/63ff0ebd56ea36189d4e7ee7?token=](https://documentation-service.arm.com/static/63ff0ebd56ea36189d4e7ee7?token)
57. Axi\_WB\_bridge | PDF | Pointer (Computer Programming) - Scribd, accessed May 19, 2025, <https://www.scribd.com/document/803725316/Axi-WB-bridge>
58. Lecture 12 - The On-chip Bus environment (2), accessed May 19, 2025, <https://schaumont.dyn.wpi.edu/ece4530f19/lectures/lecture12-notes.html>
59. ijcrt.org, accessed May 19, 2025, <https://ijcrt.org/papers/IJCRT2006392.pdf>
60. AMBA® CHI Issue F Errata - Arm, accessed May 19, 2025, [https://documentation-service.arm.com/static/6486e01b16f0f201aa6b99c7?token=](https://documentation-service.arm.com/static/6486e01b16f0f201aa6b99c7?token)
61. indico.ictp.it, accessed May 19, 2025, <https://indico.ictp.it/event/a11204/session/35/contribution/22/material/0/0.pdf>
62. Wishbone bus - HackMD, accessed May 19, 2025, <https://hackmd.io/@ruei7916/BkWhrOrt6?utm_source=preview-mode&utm_medium=rec>
63. cdn.opencores.org, accessed May 19, 2025, <https://cdn.opencores.org/downloads/wbspec_b4.pdf>
64. Tilelink — SpinalHDL documentation, accessed May 19, 2025, <https://spinalhdl.github.io/SpinalDoc-RTD/master/SpinalHDL/Libraries/Bus/tilelink/tilelink.html>
65. carrv.github.io, accessed May 19, 2025, <https://carrv.github.io/2017/papers/cook-diplomacy-carrv2017.pdf>
66. eescholars.iitm.ac.in, accessed May 19, 2025, <https://eescholars.iitm.ac.in/sites/default/files/eethesis/ee11b084.pdf>
67. docs/resources · d94b6f93af93758e3d4d2c7db269958d85f5c9de · Juan Nicolas Pardo Martin / RISC-V Lab · GitLab - at https://git.tu - TU Berlin, accessed May 19, 2025, <https://git.tu-berlin.de/kreijstal/risc-v-lab/-/tree/d94b6f93af93758e3d4d2c7db269958d85f5c9de/docs/resources>
68. AMBA AXI Protocol Specification, accessed May 19, 2025, <https://archive.alvb.in/bsc/TCC/correlatos/amba_axi4.pdf>
69. Understanding Clock Domain Crossing (CDC) Checks and Techniques - EE Times, accessed May 19, 2025, <https://www.eetimes.com/understanding-clock-domain-crossing-issues/>
70. Clock Domain Crossing: Managing Setup Time Across Different Clock Zones - FasterCapital, accessed May 19, 2025, <https://www.fastercapital.com/content/Clock-Domain-Crossing--Clock-Domain-Crossing--Managing-Setup-Time-Across-Different-Clock-Zones.html>
71. RISC-V Designs - Arteris, accessed May 19, 2025, <https://www.arteris.com/solutions/risc-v/>
72. An Overview of SoC Buses - SciSpace, accessed May 19, 2025, <https://scispace.com/pdf/an-overview-of-soc-buses-2rnxo5rrma.pdf>
73. AXI bus to Wishbone Wrapper - Stack Overflow, accessed May 19, 2025, <https://stackoverflow.com/questions/27051372/axi-bus-to-wishbone-wrapper>
74. inis.iaea.org, accessed May 19, 2025, <https://inis.iaea.org/records/7q6dy-srz11/files/23012780.pdf?download=1>
75. scispace.com, accessed May 19, 2025, <https://scispace.com/pdf/evaluation-of-interconnect-fabrics-for-an-embedded-mpsoc-in-5fl8jytoaj.pdf>