Modular PCIe SOPC Builder Bridge Example

This Modular PCIe SOPC Buider Bridge example has been created to illustrate how one might go about implementing a bridge from the Altera PCIe core's Avalon ST interfaces into the Avalon MM domain of SOPC Builder. The Altera PCIe core implements a bridge similar to this in the standard component, however for some application requirements the capabilities of the standard PCIe SOPC Builder bridge are inadequate. Some of the limitations of the standard PCIe SOPC Builder bridge that this Modular PCIe SOPC Builder Bridge example addresses are higher performance modes of the PCIe core, root port capability, and multiple MSI support. The current implementation of the standard PCIe SOPC Builder bridge can only support PCIe modes that operate up to 125MHz over a 64-bit data path, so Gen1x4 represents the highest performance that the data path of the standard bridge will support. The current standard bridge has no support for root port operation and it only supports 1 MSI and only 16 TAGs. All of these limitations are addressed in the Modular PCIe SOPC Builder Example and in theory the only limitations of the Modular bridge are those limitations that are imposed by the Altera PCIe core's Avalon ST capabilities. The current Modular PCIe SOPC Builder Bridge components are limited to PCIe modes that run over the 64-bit Avalon ST data path, but they can run up to 250MHz, which means that Gen1x8 and Gen2x4 architectures can be implemented. The modular bridge example also supports root port operation, 32 MSIs, and 64 TAGs, which are the limitations imposed by the Altera PCIe Avalon ST implementation.

The slides that follow illustrate the high level functionality and data path interconnect for the Modular PCIe SOPC Builder Bridge example. There is actually significant detail that is not shown in the diagrams, and the best reference for these low level details would be to study an actual example bridge design in SOPC Builder, many of which are provided with this documentation.

Here is an overview of what each slide contains

Overview

This documentation is not intended to provide basic PCIe instruction, nor Altera PCIe core instruction, it is only intended to provide insight to the Modular PCIe Bridge example architecture. It assumes that the user already understands the basics of PCIe and how to configure and implement the Altera PCIe core. The essential capability of this Modular PCIe Bridge is to provide an interface to the Altera PCIe core from within SOPC Builder, mapping the memory mapped domain of SOPC Builder into the PCIe domain and vice versa. To that end the intentions of this bridge are to allow Native PCIe Endpoints or Root Ports to be easily architected and implemented within this context. While you could contrive an architecture that allowed Legacy Endpoint construction, that is not of primary interest in this bridge architecture. This essentially means is that as an endpoint, you could reasonably expect to create a requester, and/or completer architecture that can handle PCIe memory read and write transactions and leverage the MSI facility in the PCIe domain. As a root port you could expect the same memory read/write requester/completer capabilities that an endpoint has with the addition of to being able to issue configuration requests and receiving messages from the PCIe domain.

The custom hardware blocks which make up the Modular PCIe Bridge example handle the translation of PCIe TLP packets into Avalon MM transactions and vice versa. They can receive requests and return completions as well as post requests and receive completions.

The request generation logic ensures that the dynamic system settings for Maximum Payload Size (MPS) and Maximum Read Request Size (MRRS) are observed and applied to the request TLP generation rules.   It also ensures that 4K boundary crossing are properly translated into valid TLP request sequences.

The completion generation logic ensures that the dynamic system settings for MPS and Read Completion Boundary (RCB) are observed and applied to the completion TLP generation rules.

Byte Enable Rules – The byte enable rules for PCIe and Avalon MM are not quite aligned with today's Avalon Interface Specification. PCIe allows for read and write burst to begin and end with misaligned but contiguous byte enable assertions. Avalon MM requires all byte enables to be asserted during read bursts greater than 1 and it allows discontinuous assertions on each word of a write burst. Both PCIe and Avalon MM allow for discontinuous byte enable assertions for a single word transaction, or 2DW for PCIe. So in order to perform compatible transactions between the two spaces you must ensure the following byte enable assertion rules are followed:

- Any 1DW or 2DW PCIe read or write transaction can map into a valid Avalon MM 1 word burst.
- Any PCIe write burst can map into a legal Avalon MM write burst.
- Any PCIe read burst with misaligned byte enables will map into an Avalon MM transaction with all byte enables asserted. While this is not a direct mapping, it shouldn't cause any problems.
- Any Avalon MM write transaction must assert PCIe compatible byte enables, they can be misaligned, but must be contiguous thru the burst. The bridge will force continuity thru the burst regardless.
- Any Avalon MM read transaction can map directly into a PCIe transaction.

TAG Loss – Within the domain of PCIe the possibility of a request TAG timeout can occur. In a properly functioning and configured PCIe environment, the only reason that TAG loss should occur is due to a programming error in the form of sending requests to invalid addresses within the PCIe domain. Other than programming errors there is no reason why TAG loss should occur in a properly functioning and configured PCIe environment. Because of this, the TLP request generation logic in this bridge makes no attempt to deal with the possibility of loosing a TAG. If the hardware state machines of the request generation logic are commanded to perform illegitimate read requests into the PCIe domain they will simply lock up and wait indefinitely for the TAG sequences to return. Recovering dynamically from a lost TAG is a very difficult proposition for the state machines in the request generation logic, and the ramifications of unwinding the state machines in the SOPC system which issued these requests becomes even more daunting. For this reason, the current implementation of this bridge logic does not attempt to recover dynamically from this possibility.

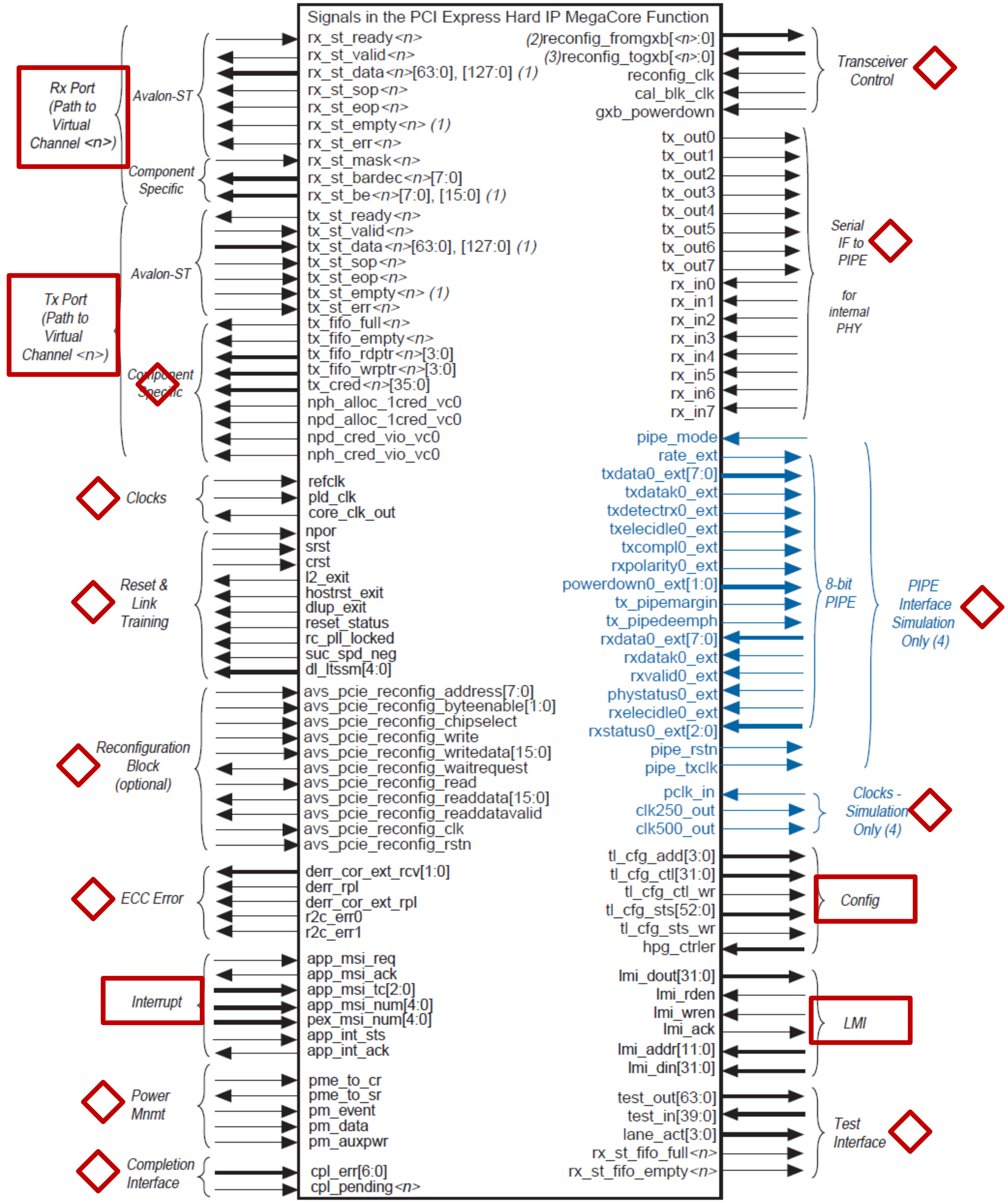
The RAW TLP FIFO paths that can be implemented in this bridge architecture are not so tightly confined by the issue of TAG loss. In fact since they are most likely driven and controlled by software processes, they can recover from TAG loss quite painlessly. Because of this, for root port configuration requests and other memory probing requirements that cannot guarantee proper TAG return it's recommended that those transactions should flow thru the RAW TLP FIFO interfaces.

While this document does not go into the subject of multiple TC/VC architectures, the Modular PCIe Bridge could be composed in such a way to allow it to function in multiple VC and multiple TC environments.

There are also some non-PCIe applications that can be implemented with the Modular PCIe SOPC Builder Bridge architecture. This is not discussed here, but an illustration is provided in these slides to show the concept.

Modular PCIe SOPC Builder Bridge HIP Interfaces

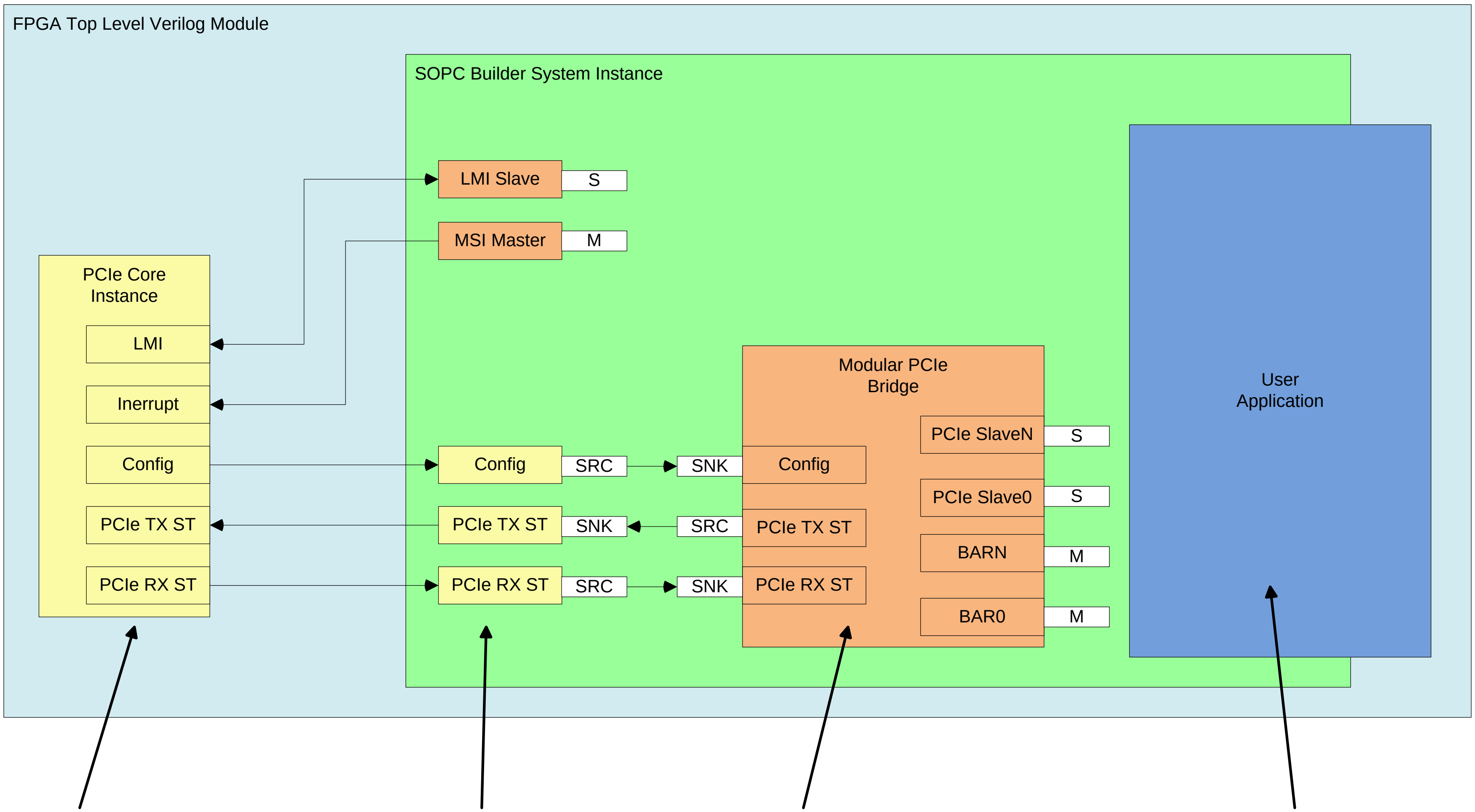
Figure 5–2. Signals in the Hard IP Implementation Endpoint with Avalon-ST Interface



□ = Interface developed in Modular PCIe SOPC Builder Bridge.

◇ = Interface NOT developed in Modular PCIe SOPC Builder Bridge.

Modular PCIe SOPC Builder Bridge Implementation Strategy



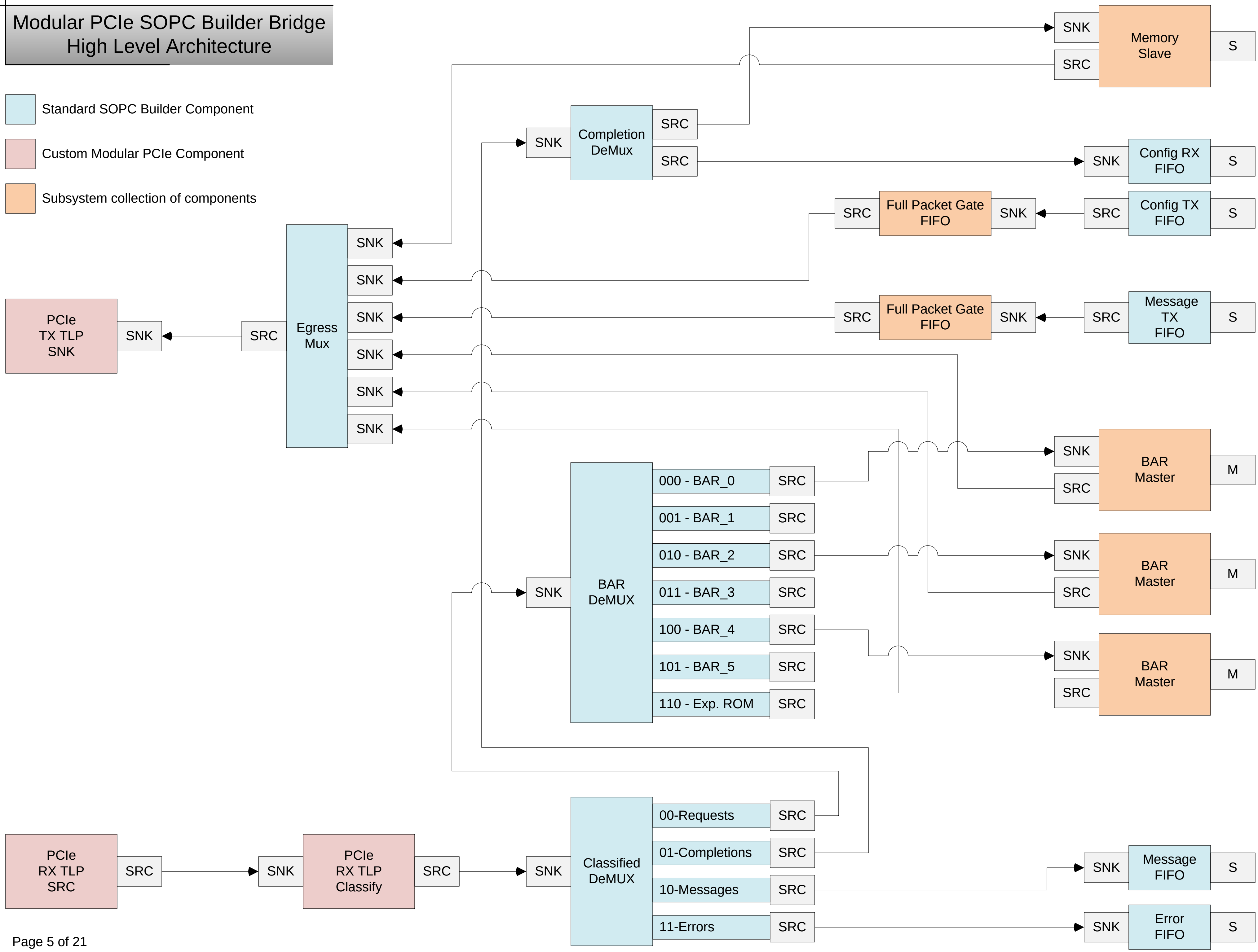
Altera PCIe core is configured and instantiated outside of the SOPC Builder system.

Interfaces to the external PCIe core are created inside the SOPC Builder system.

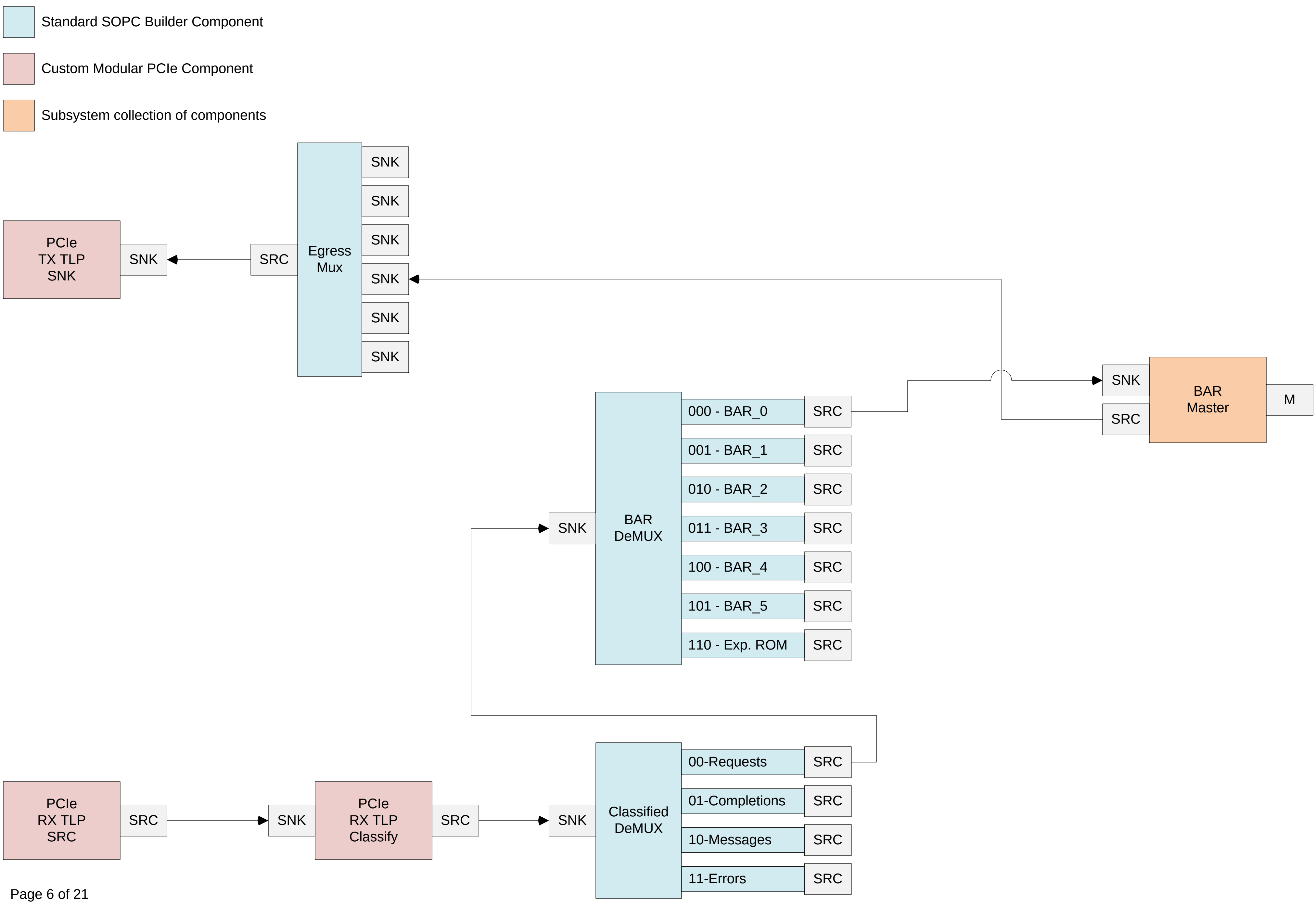
The Modular PCIe SOPC Builder Bridge architecture translates PCIe transactions into Avalon MM transactions and vice versa.

The user application can reside inside the SOPC Buidler system or outside the SOPC Builder system, however the Avalon MM interfaces are the access point required to interact with the bridge.

Modular PCIe SOPC Builder Bridge High Level Architecture



Modular PCIe SOPC Builder Bridge
Simple Completer Only

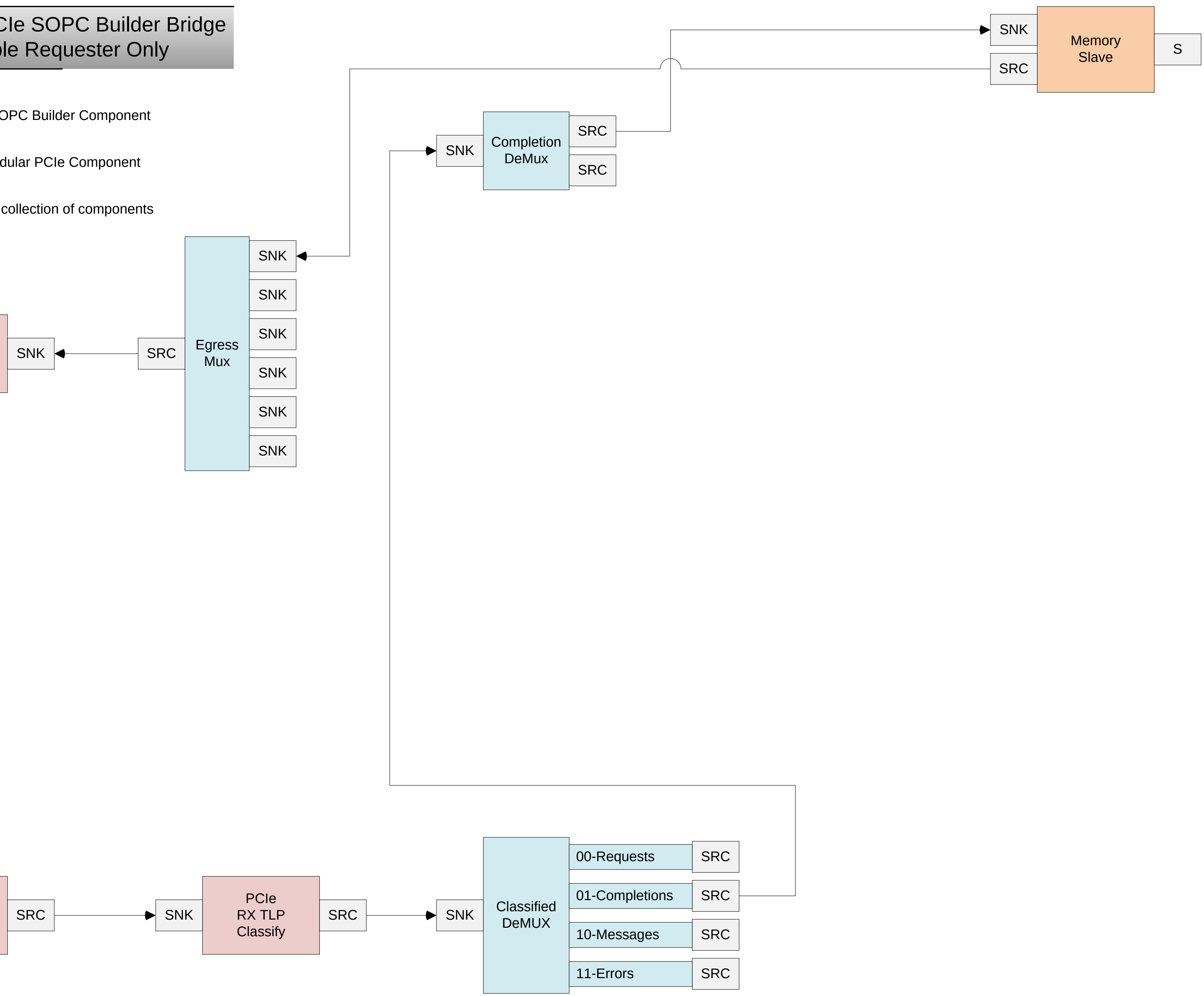


Modular PCIe SOPC Builder Bridge
Simple Requester Only

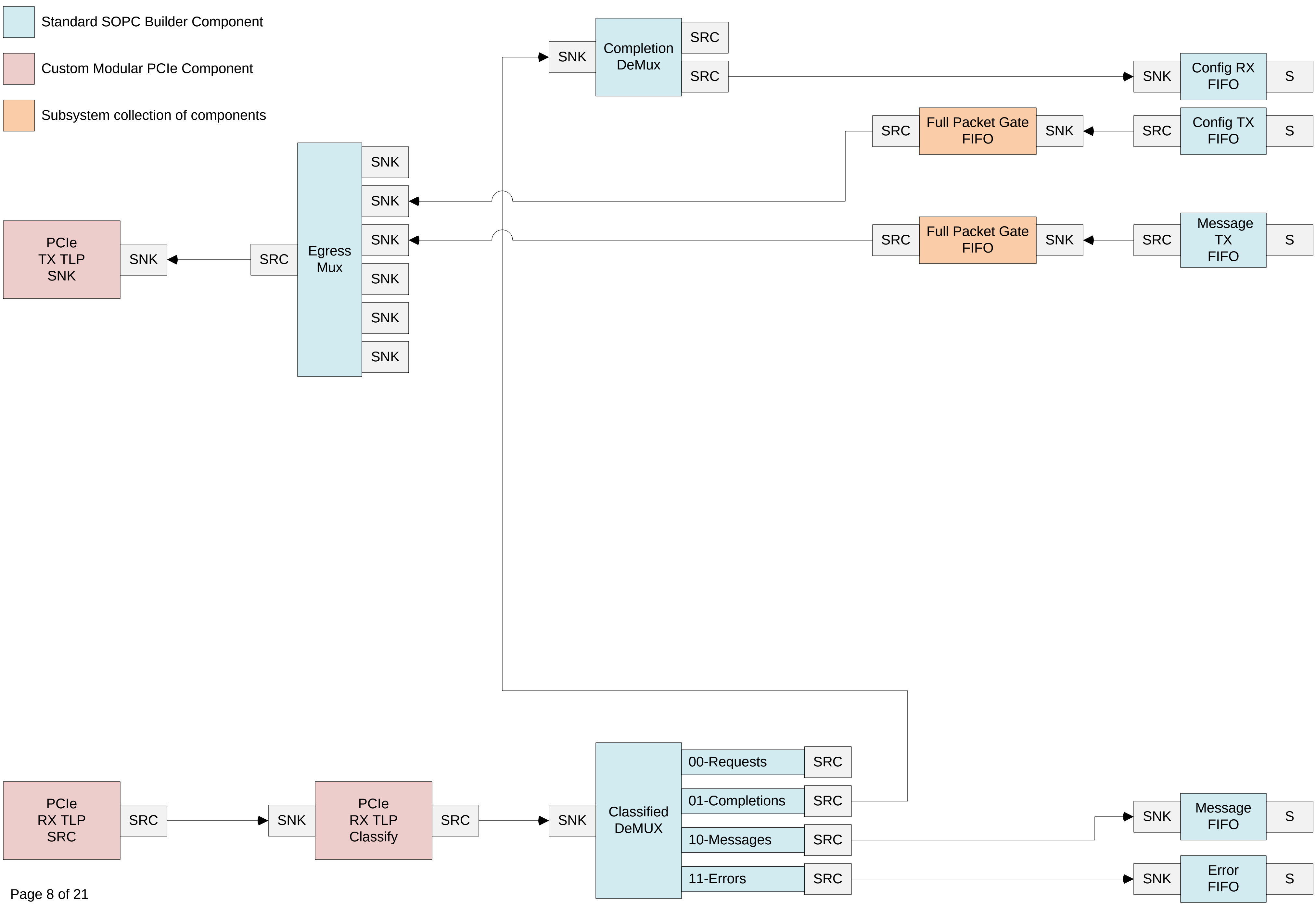
Standard SOPC Builder Component

Custom Modular PCIe Component

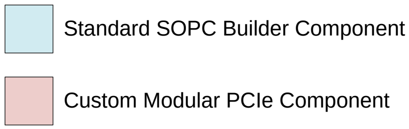
Subsystem collection of components

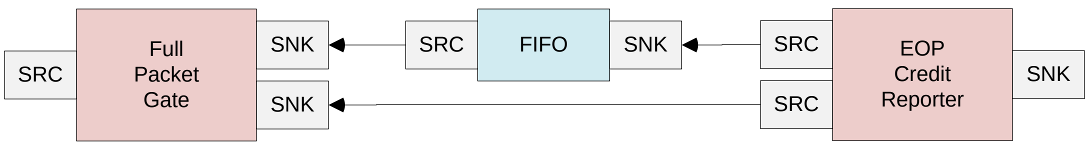


Modular PCIe SOPC Builder Bridge Optional RAW TLP Interfaces



Modular PCIe SOPC Builder Bridge
Full Packet Gate



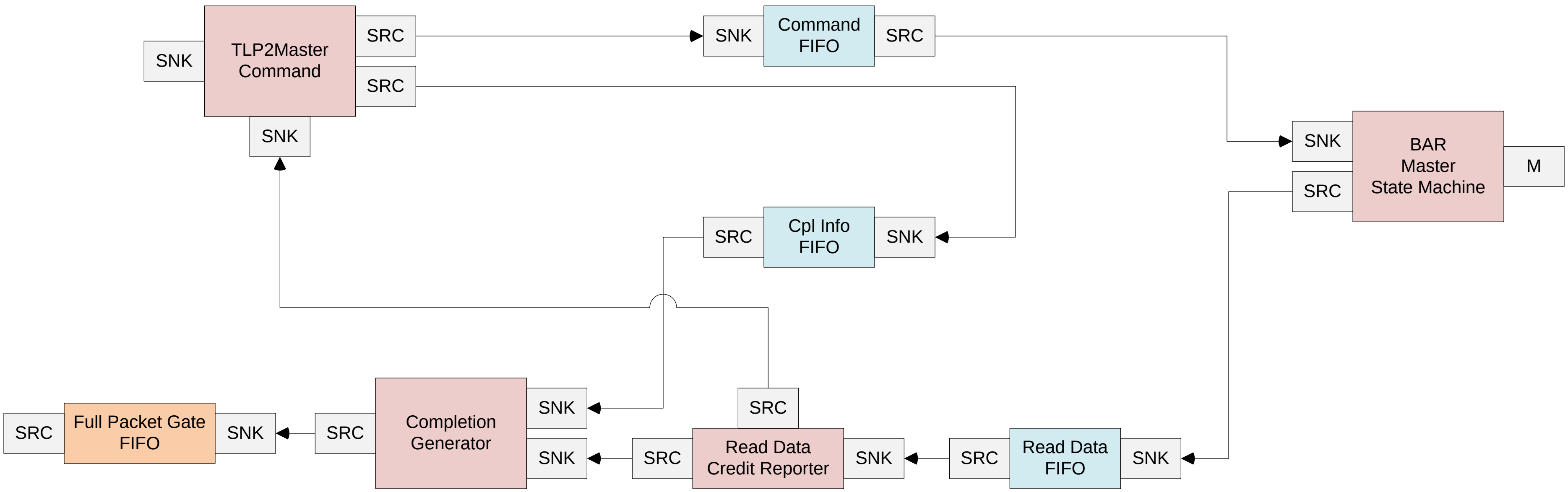


Modular PCIe SOPC Builder Bridge
BAR Master

Standard SOPC Builder Component

Custom Modular PCIe Component

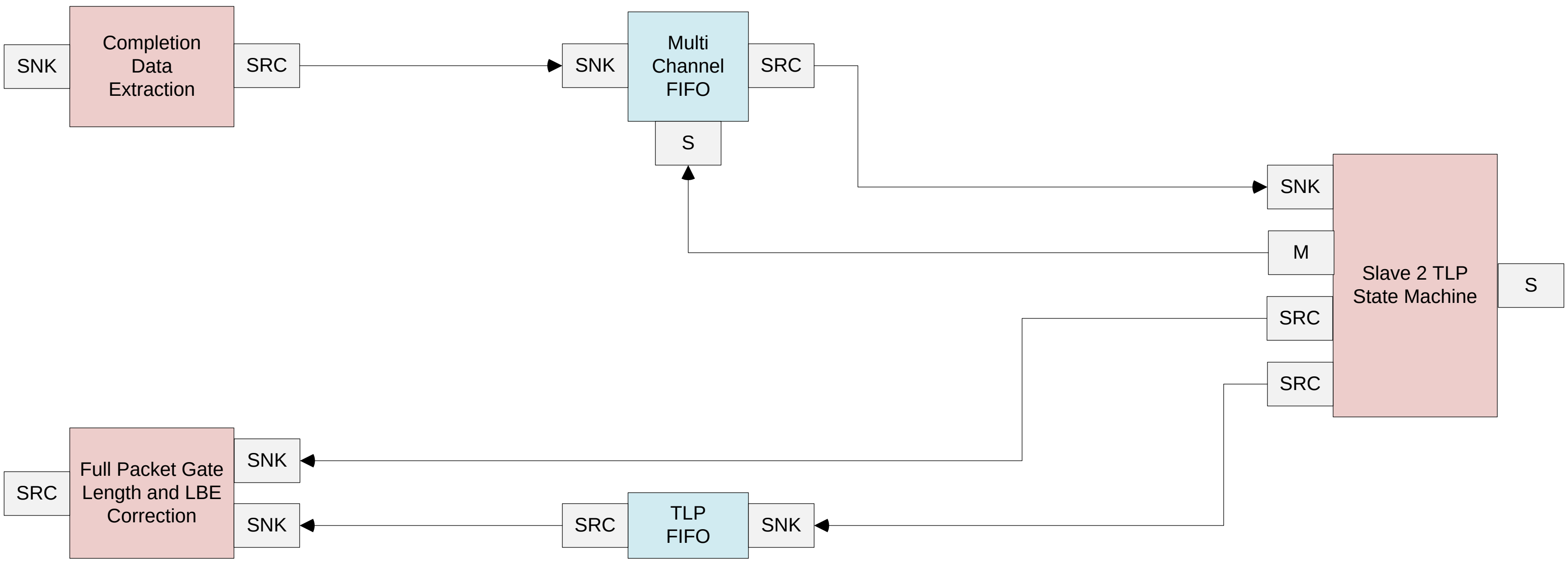
Subsystem collection of components



Modular PCIe SOPC Builder Bridge Memory Slave

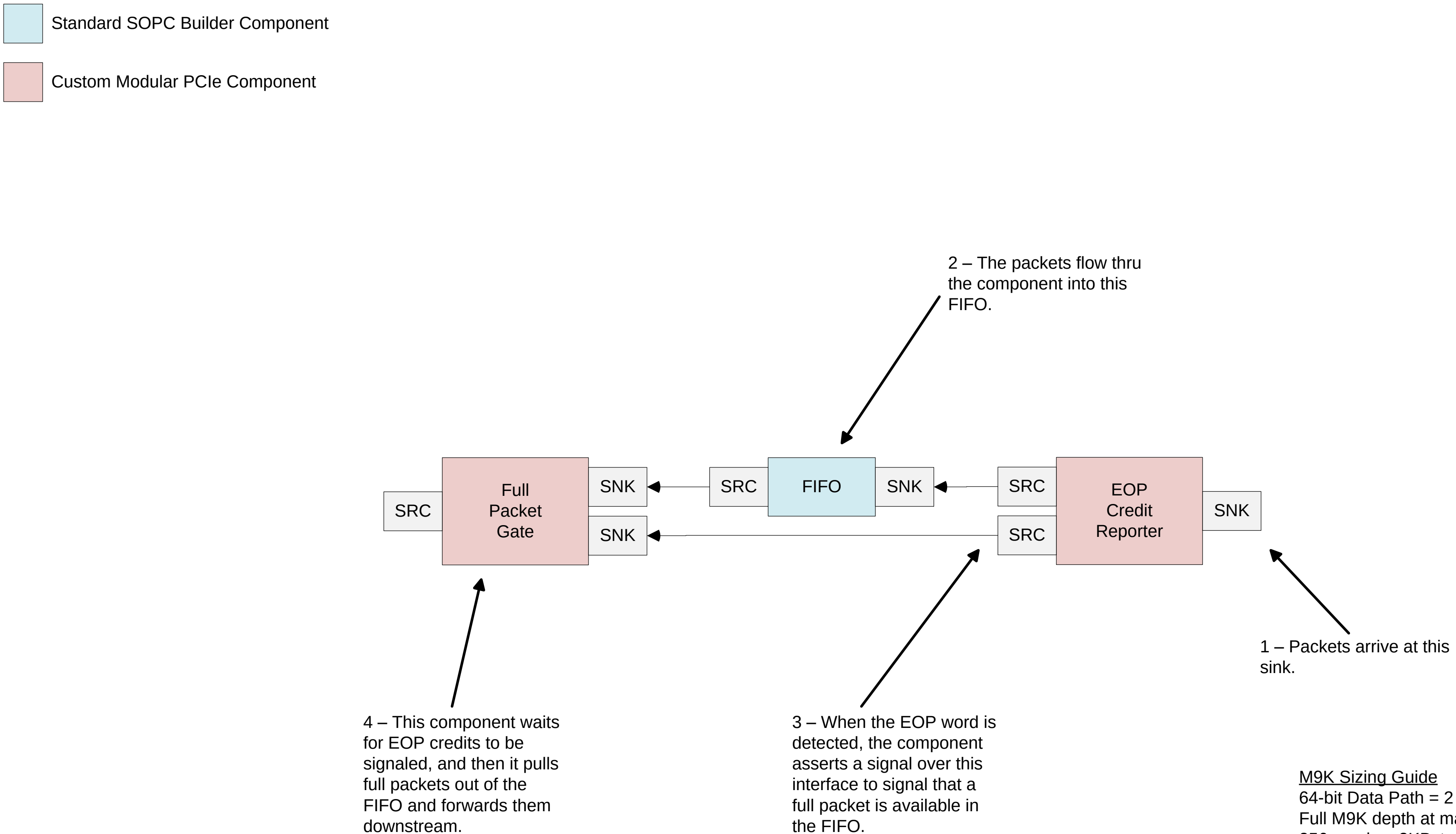
Standard SOPC Builder Component

Custom Modular PCIe Component



Modular PCIe SOPC Builder Bridge
Full Packet Gate

Why does this structure exist? The PCIe HIP core has a requirement that once a TLP Packet has begun transmission into the core, it must not pause or stop until the end of the packet. This structure ensures that a full TLP packet is available to transmit into the PCIe HIP core.



<u>M9K Sizing Guide</u>
64-bit Data Path = 2 M9K RAMs
Full M9K depth at max width = 256 words
256 words = 2KBytes
2Kbytes = 2 MPS @ 1K
2Kbytes = 4 MPS @ 512
2Kbytes = 8 MPS @ 256
2Kbytes = 16 MPS @ 128

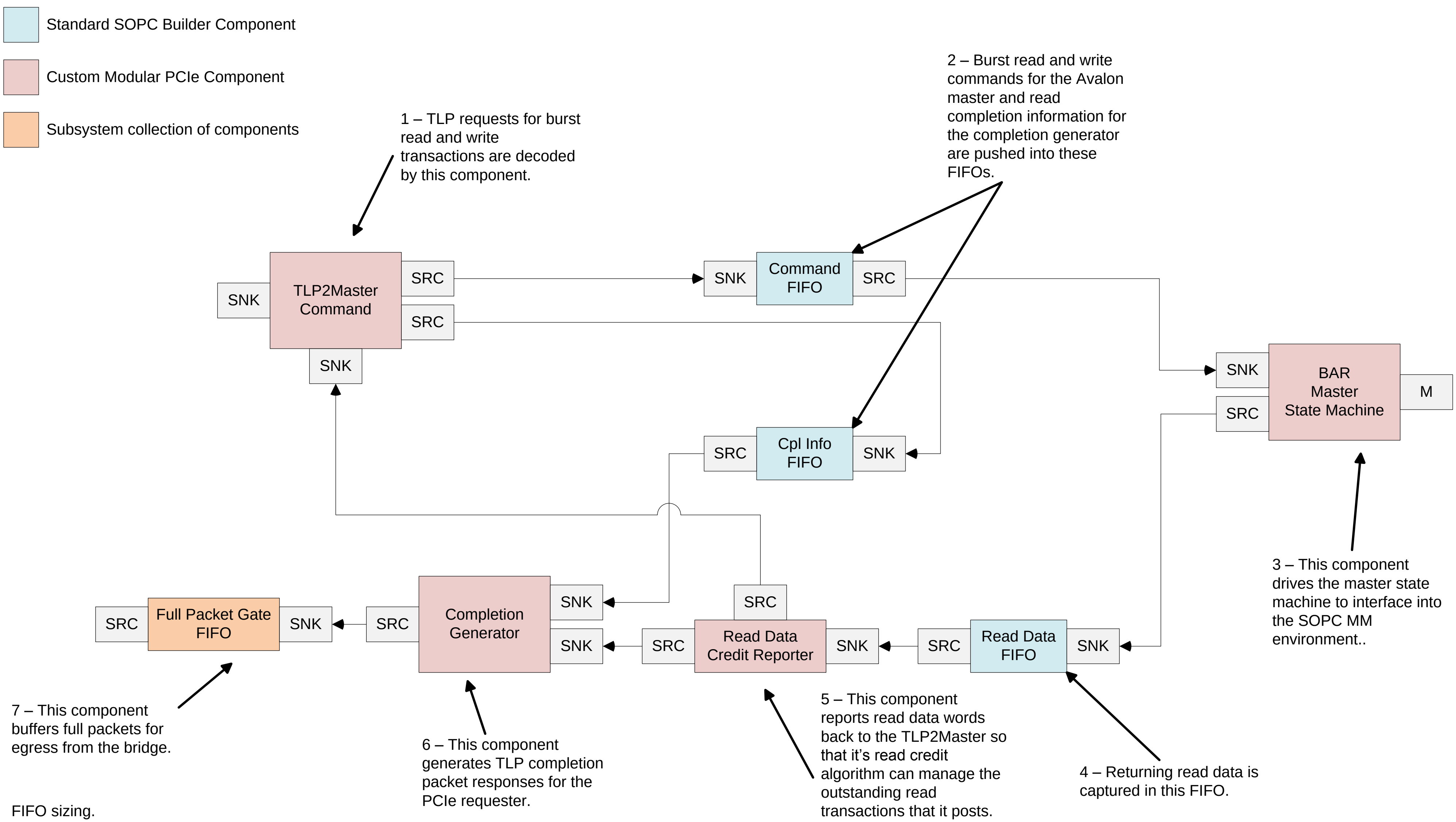
FIFO sizing.

The size of this FIFO is primarily constrained by the Max Payload Size of the PCIe system in which this bridge will be operating. At a minimum this FIFO should be capable of holding 1 Max Payload Size TLP packet, but for performance reasons it would be better if it could hold 2 or more. Depending on the exact topology of the bridge, a 2 packet capacity may be sufficient for the best performance, but in many cases more than 2 packet capacity would be better.

If you consider that this data path will likely be 64-bits wide, it would require 2 M9K RAMs to create a FIFO wide enough to buffer this data, and if you use the maximum depth of the M9K RAM you'll get 256 words of storage which is 2KBytes of space. Since most PCIe systems operate with MPS of 512 bytes or less, this minimal M9K FIFO construction would generally meet the needs of most systems.

Modular PCIe SOPC Builder Bridge BAR Master

Why does this structure exist? This structure translates PCIe memory burst read and write requests into Avalon MM transactions. The BAR Master can be connected to any Avalon Slave interfaces within the SOPC system.



FIFO sizing.

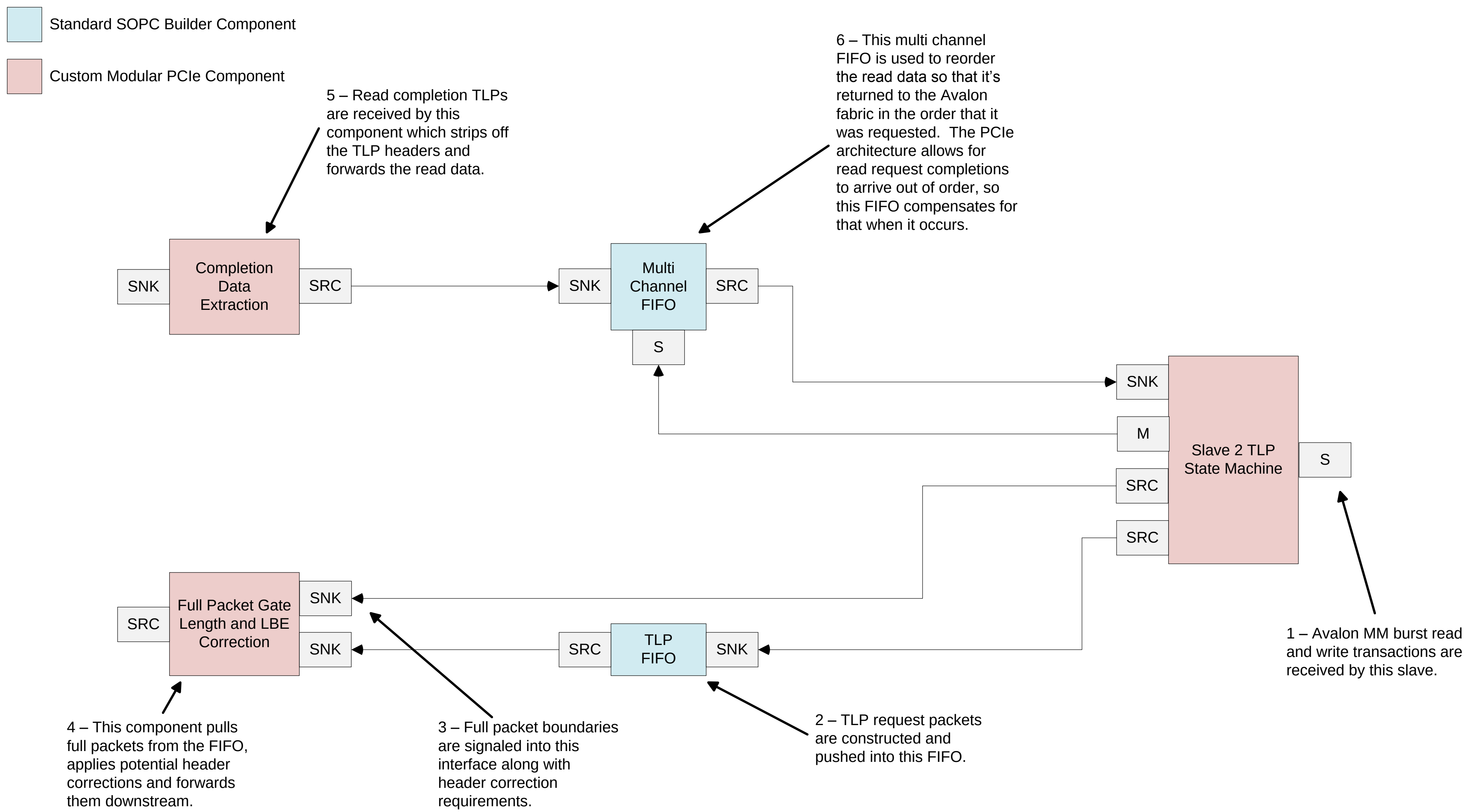
The Command FIFO is primarily constrained by the Max Payload Size of the PCIe system in which this bridge will be operating. At a minimum this FIFO should be capable of holding 1 Max Payload Size TLP packet, but for performance reasons it would be better if it could hold 2 or more. The width of this FIFO is typically greater than 64-bits so it will typically require 3 M9K RAMs to cover the width of this FIFO.

The Read Data FIFO is primarily constrained by the Max Read Request Size of the PCIe system in which this bridge will be operating. At a minimum this FIFO should be capable of holding 1 Max Read Request Size buffer, but for performance reasons it would be better if it could hold 2 or more. The width of this FIFO is 64-bits so it will require 2 M9K RAMs to cover the width of this FIFO.

The Completion Information FIFO is primarily constrained by the number of outstanding read requests you desire to support thru the Avalon system. So practically this is constrained by the depth of the read data FIFO and the size of the read requests that are issued over the PCIe interface. The width of this FIFO is 64-bits so it will require 2 M9K RAMs to cover the width of this FIFO.

Modular PCIe SOPC Builder Bridge Memory Slave

Why does this structure exist? This structure translates Avalon burst read and write requests into PCIe request TLP packets. The Avalon sllave can be connected to any Avalon Master interfaces within the SOPC system.

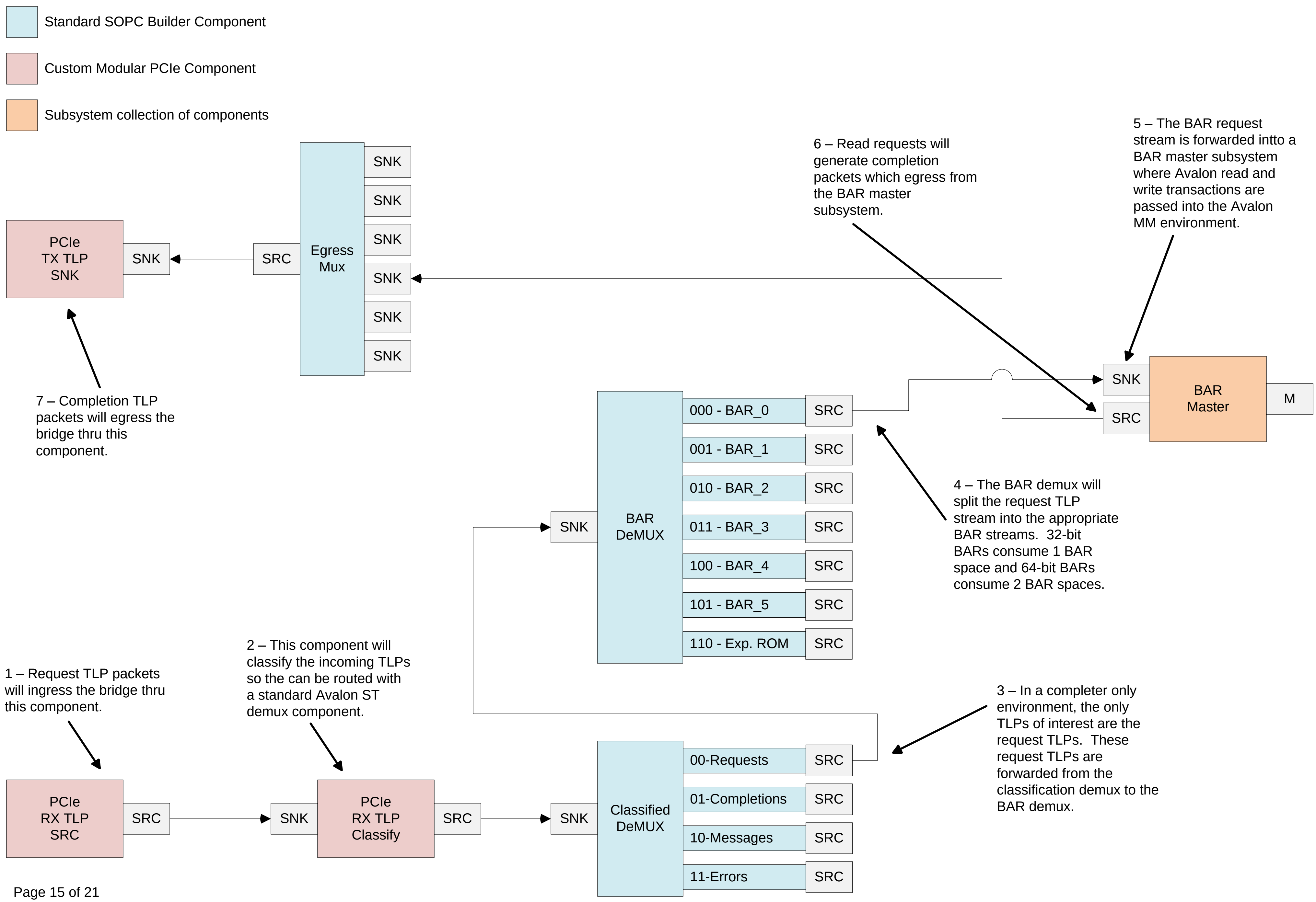


FIFO sizing.

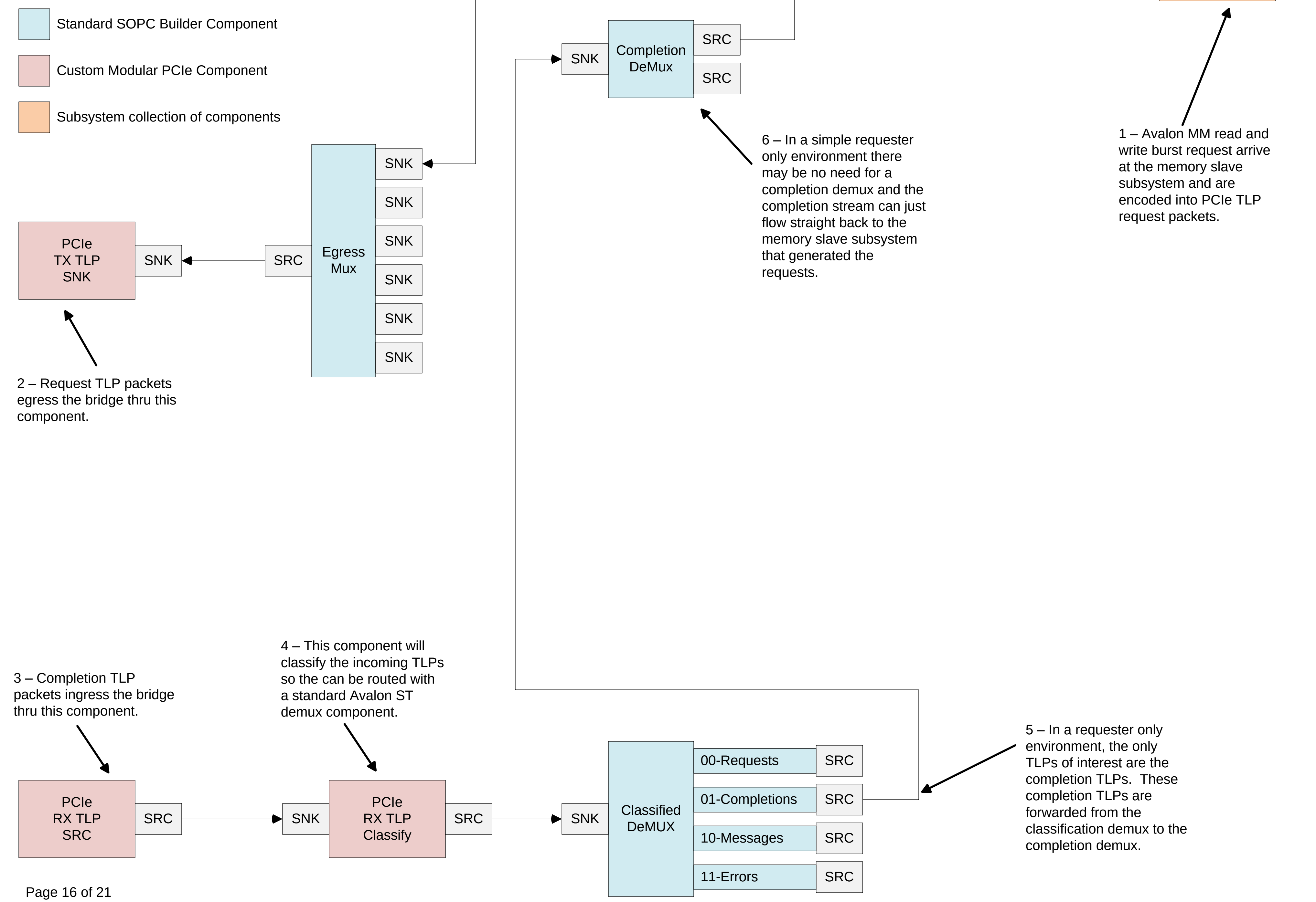
The TLP FIFO is primarily constrained by the Max Payload Size of the PCIe system in which this bridge will be operating. At a minimum this FIFO should be capable of holding 1 Max Payload Size TLP packet, but for performance reasons it would be better if it could hold 2 or more. The width of this FIFO is typically 64-bits so it will typically 2 M9K RAMs to cover the width of this FIFO.

The Multi Channel FIFO is primarily constrained by the Max Read Request Size of the PCIe system in which this bridge will be operating and the number of TAGs allocated to this memory slave structure within the bridge. At a minimum this FIFO should be configured to receive 2 channels, but for performance reasons 8 to 16 channels would be better. The depth of each channel buffer is constrained by the Max Read Request Size that is expected in the system. The width of this FIFO is 64-bits so it will require 2 M9K RAMs to cover the width of this FIFO.

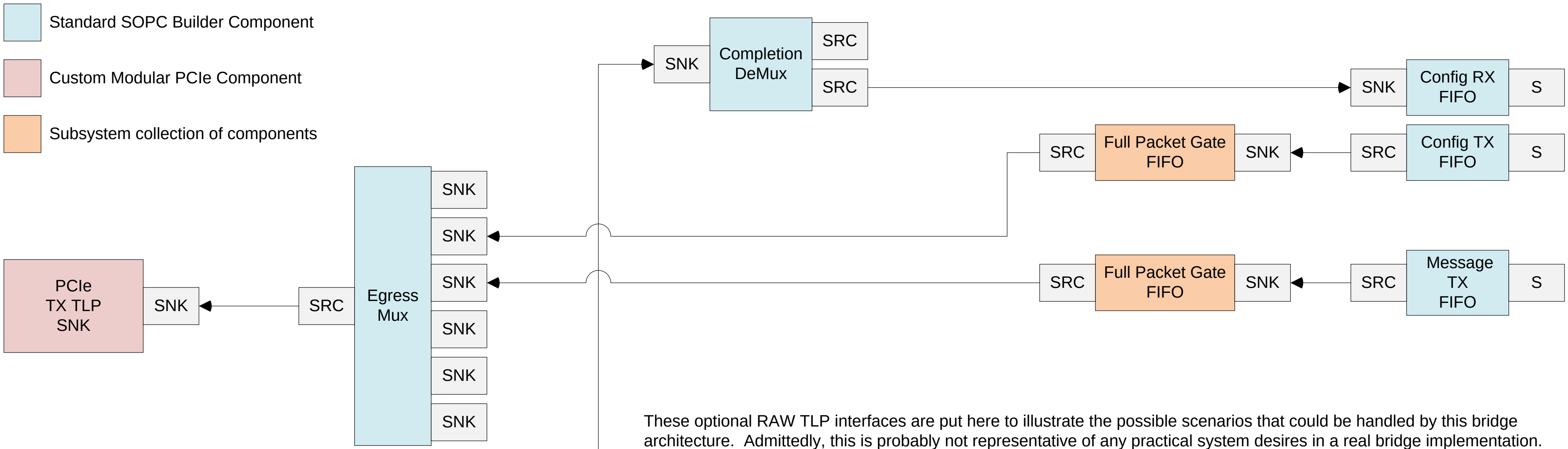
Modular PCIe SOPC Builder Bridge Simple Completer Only



Modular PCIe SOPC Builder Bridge Simple Requester Only



Modular PCIe SOPC Builder Bridge Optional RAW TLP Interfaces

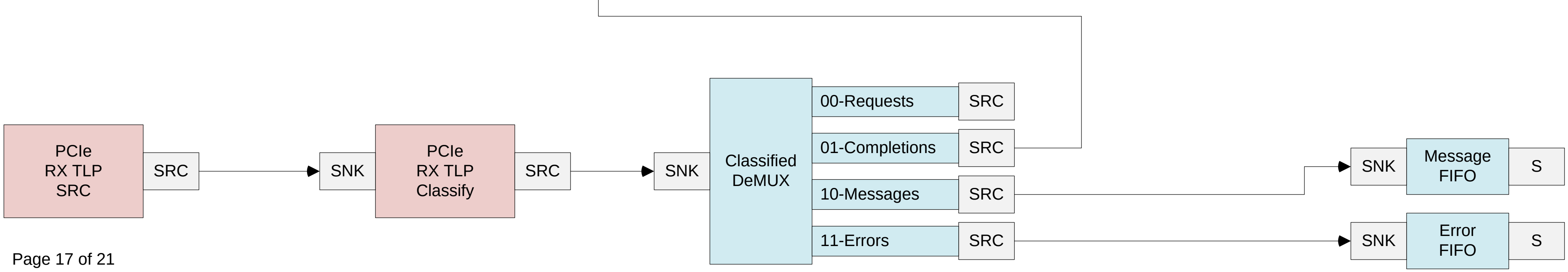


These optional RAW TLP interfaces are put here to illustrate the possible scenarios that could be handled by this bridge architecture. Admittedly, this is probably not representative of any practical system desires in a real bridge implementation.

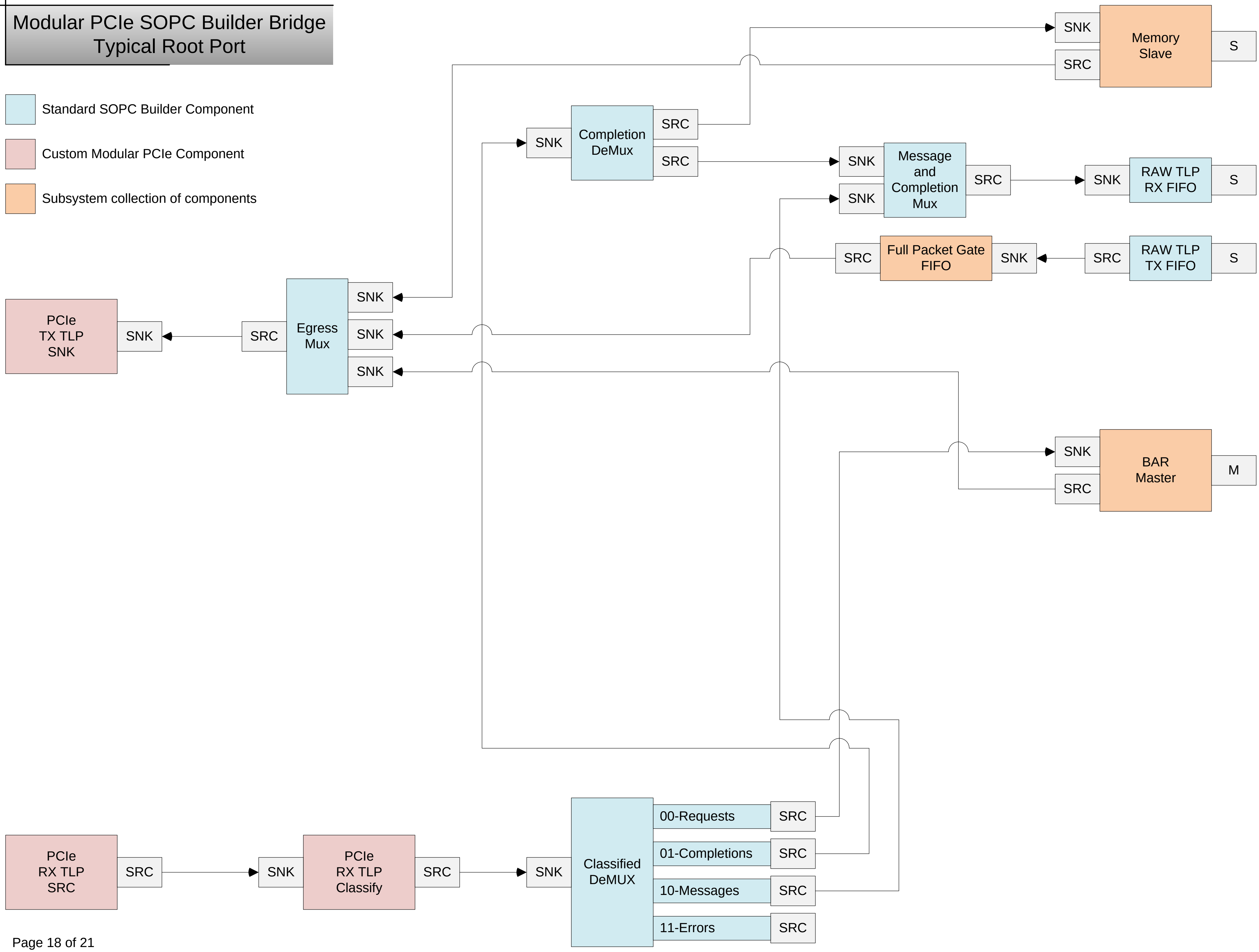
The blue FIFO objects on the right of the diagram represent standard Altera Onchip Memory FIFO components from SOPC Builder. These FIFOs can be configured such that they have an Avalon ST interface on one side for interacting with the data path of the Modular PCIe bridge, and then an Avalon Slave interface on the other side for interacting with something like a CPU. It should be noted that a detail missing from this diagram is the need to swap the endian around for the byte symbols that pass in and out of the Altera Onchip Memory FIFO Avalon ST interface, this component is provided in the Modular PCIe component library but it is not illustrated here.

For Root Port bridge applications it is assumed that a pair of RAW TLP FIFOs like these would be used to provide a path for the local CPU to push configuration request TLP packets into the bridge and extract configuration completion TLP packets from the bridge. The format of these RAW TLP packets must conform to the TLP packet layout required by the PCIe HIP core, since no additional formatting will be applied by the hardware within the bridge.

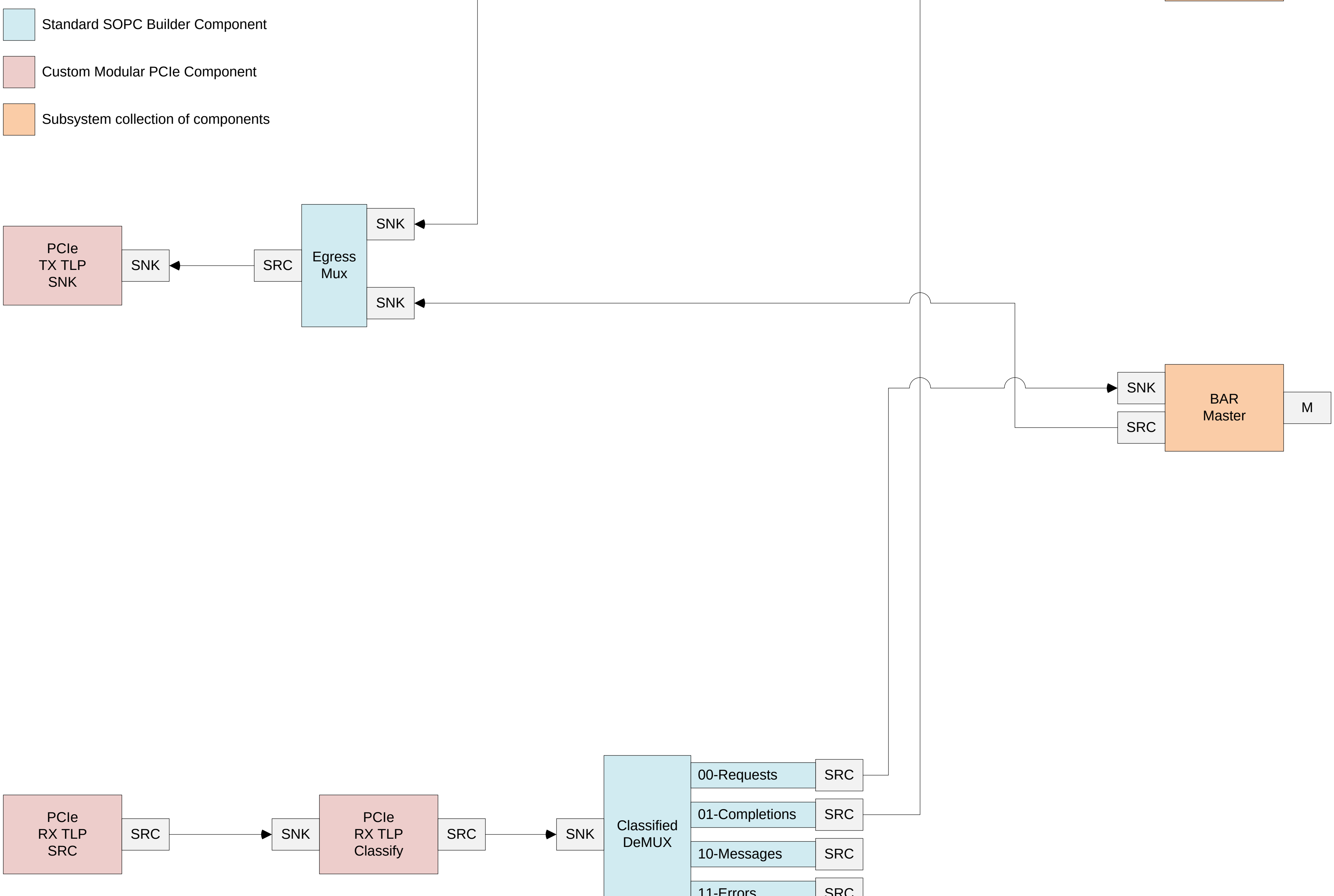
When manually pushing RAW TLP packets thru these FIFO constructs, special care should be taken to manage the TAG allocation plan for the bridge so that completion packets are properly routed back to the RAW TLP FIFO that you expect.



Modular PCIe SOPC Builder Bridge Typical Root Port



Modular PCIe SOPC Builder Bridge Typical Endpoint



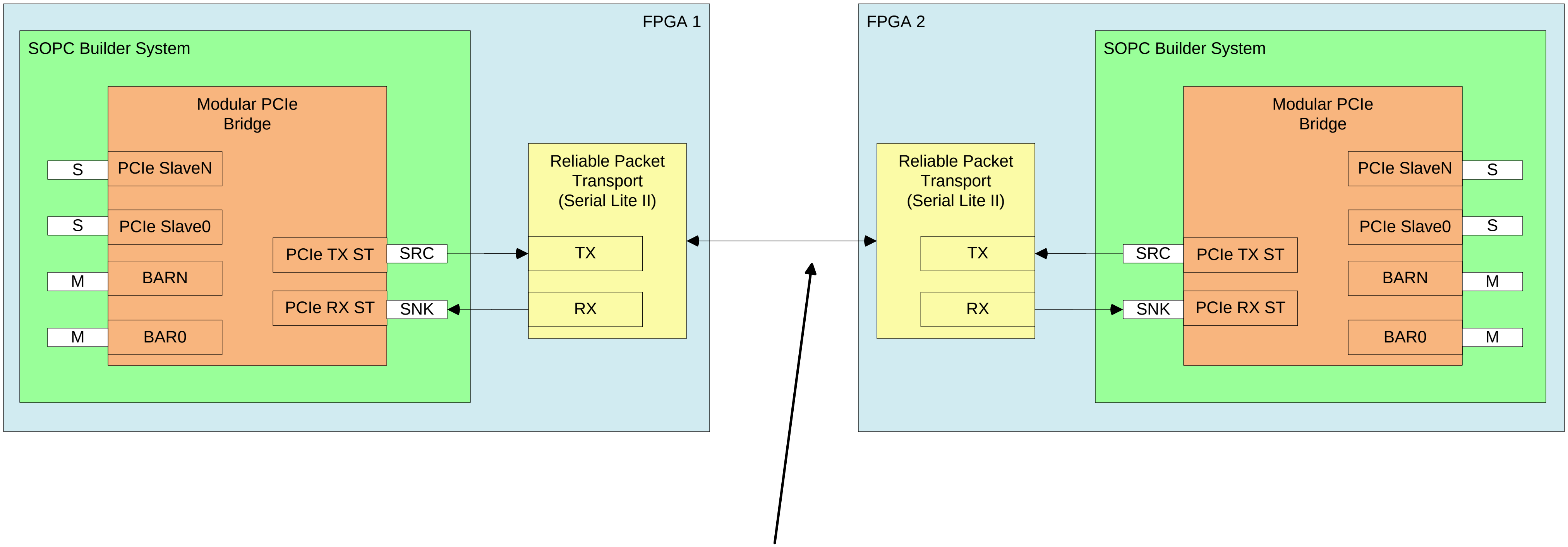
Modular PCIe SOPC Builder Bridge
Potential Low Performance Config

Standard SOPC Builder Component

Custom Modular PCIe Component

Subsystem collection of components

Modular PCIe SOPC Builder Bridge Non-PCIe Application



The Modular PCIe SOPC Builder Bridge could actually be deployed in non-PCIe applications. As long as you have some reliable packet transport which can guarantee packet delivery like PCIe does, then you could actually construct architectures that can connect SOPC systems in different FPGAs over that data link.