

1 Axi

Packages

```
import Axi :: * ;
```

Description

The AXI library includes interface, transactor, module and function definitions to implement the Advanced eXtensible Interface (AXI) protocol with Bluespec SystemVerilog. The BSV AXI library groups the AXI data and protocols into reusable, parameterized interfaces, which interact with TLM interfaces. An AXI bus is implemented using AXI transactors to connect TLM interfaces on one side with AXI interfaces on the other side. The TLM interfaces used by the **Axi** package are defined in the TLM2 package.

The AXI library supports the following AXI Bus protocol features:

- Basic and Burst Transfers
- Aligned and Unaligned Transfers

The AXI library does not support the following AXI Bus protocol features:

- Exclusive/Locked Access
- Low Power Interface
- Cache Transaction Attributes

The basic structure of an AXI write bus is show in figure 1. The structure of a read bus is similar. (Note that the nature of the AXI protocol is such that the read and write buses operate totally independently of each other).

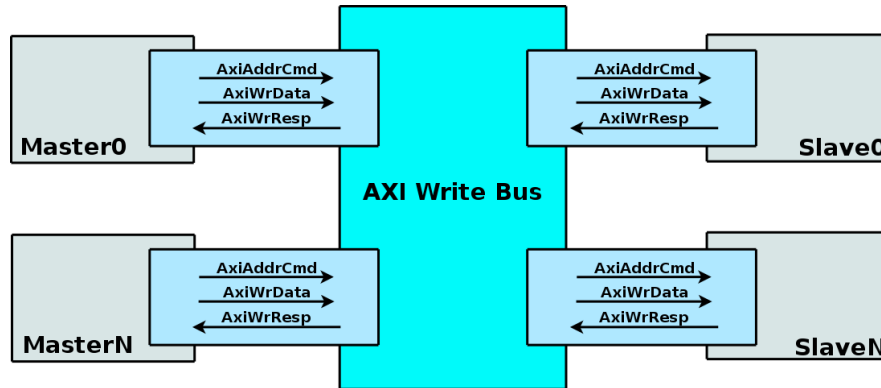


Figure 1: AXI Write Bus Example

The corresponding BSV AXI implementation is shown in figure 2. TLM Write requests are received via the **TLMRecvIFC** interfaces of the master transactors. The request is then transmitted via the **AxiWrMaster** interface out onto the AXI bus and on to the appropriate slave transactor. The slave transactor receives the request via the **AxiWrSlave** interface, translates the request back into a stream of TLM objects, and then transmits those objects via the **TLMSendIFC** interface. The TLM response from the write operation follows the same path in reverse.

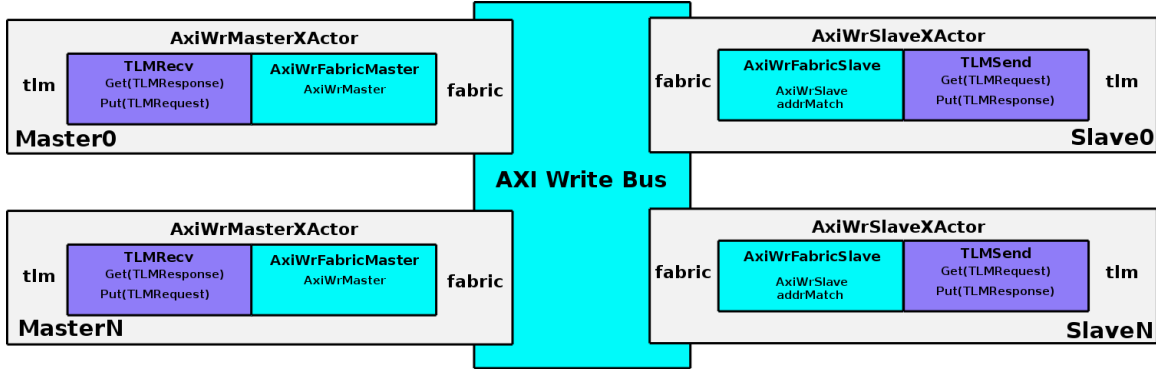


Figure 2: BSV AXI Write Bus Implementation Using TLM Transactors

Data Structures

Inside the transactor modules, the AXI data is organized into the following data structures: the address data is defined by **AxiAddrCmd**, the read response is defined by **AxiRdResp**, the write data is defined by **AxiWrData** and the write response is defined by **AxiWrResp**.

AxiAddrCmd The AXI Address Bus is defined by a structure, **AxiAddrCmd**, the components of which are described in the following table.

AxiAddrCmd		
Member Name	DataType	Valid Values
id	AxiId#('TLM_PRM)	Bit#(id_size)
len	AxiLen	Bit#(4)
size	AxiSize	Bit#(3)
burst	AxiBurst	FIXED, INCR, WRAP
lock	AxiLock	NORMAL, EXCLUSIVE, LOCKED
cache	AxiCache	Bit#(4)
prot	AxiProt	Bit#(3)
addr	AxiAddr#('TLM_PRM)	Bit#(addr_size)

```
typedef struct {
    AxiId#('TLM_PRM) id;
    AxiLen len;
    AxiSize size;
    AxiBurst burst;
    AxiLock lock;
    AxiCache cache;
    AxiProt prot;
    AxiAddr#('TLM_PRM) addr;
} AxiAddrCmd#('TLM_PRM_DCL) deriving(Bits,Eq);
```

AxiRdResp The AXI Read Bus is defined by the **AxiRdResp** structure, the components of which are described in the following table.

AxiRdResp		
Member Name	DataType	Valid Values
id	AxiId#('TLM_PRM)	Bit#(id_size)
data	AxiData#('TLM_PRM)	Bit#(data_size)
resp	AxiResp	OKAY, EXOKAY, SLVERR, DECERR
last	Bool	True, False

```
typedef struct {
    AxiId#('TLM_PRM)    id;
    AxiData#('TLM_PRM) data;
    AxiResp              resp;
    Bool                 last;
} AxiRdResp#('TLM_PRM_DCL) deriving(Bits,Eq);
```

The AXI Write Bus is defined by two structures, **AxiWrData** and **AxiWrResp**.

AxiWrData The components of **AxiWrData** are described in the following table.

AxiWrData		
Member Name	DataType	Valid Values
id	AxiId#('TLM_PRM)	Bit#(id_size)
data	AxiData#('TLM_PRM)	Bit#(data_size)
strb	AxiByteEn#('TLM_PRM)	Bit#(TDiv#(data_size, 8))
last	Bool	True, False

```
typedef struct {
    AxiId#('TLM_PRM)    id;
    AxiData#('TLM_PRM) data;
    AxiByteEn#('TLM_PRM) strb;
    Bool                 last;
} AxiWrData#('TLM_PRM_DCL) deriving(Bits,Eq);
```

AxiWrResp The components of **AxiWrResp** are described in the following table.

AxiWrResp		
Member Name	DataType	Valid Values
id	AxiId#('TLM_PRM)	Bit#(id_size)
resp	AxiResp	OKAY, EXOKAY, SLVERR, DECERR

```
typedef struct {
    AxiId#('TLM_PRM)    id;
    AxiResp              resp;
} AxiWrResp#('TLM_PRM_DCL) deriving(Bits,Eq);
```

Bus Interfaces

This section describes the AXI bus master and slave interfaces used by the AXI transactor modules. Since the AXI protocol supports read and write operations on separate buses, two flavors of each interface exist, one for reads and one for writes.

AxiRdMaster The AxiRdMaster interface issues AXI read requests and receives AXI read responses.

```
interface AxiRdMaster#('TLM_PRM_DCL);
    // Address Outputs
    method AxiId#('TLM_PRM) arID;
    method AxiAddr#('TLM_PRM) arADDR;
    method AxiLen arLEN;
    method AxiSize arSIZE;
    method AxiBurst arBURST;
    method AxiLock arLOCK;
    method AxiCache arCACHE;
    method AxiProt arPROT;
    method Bool arVALID;

    // Address Inputs
    method Action arREADY(Bool value);

    // Response Outputs
    method Bool rREADY;

    // Response Inputs
    method Action rID (AxiId#('TLM_PRM) value);
    method Action rDATA (AxiData#('TLM_PRM) value);
    method Action rRESP (AxiResp value);
    method Action rLAST (Bool value);
    method Action rVALID(Bool value);
endinterface
```

AxiWrMaster The AxiWrMaster interface issues AXI write requests and receives AXI write responses.

```
interface AxiWrMaster#('TLM_PRM_DCL);
    // Address Outputs
    method AxiId#('TLM_PRM) awID;
    method AxiAddr#('TLM_PRM) awADDR;
    method AxiLen awLEN;
    method AxiSize awSIZE;
    method AxiBurst awBURST;
    method AxiLock awLOCK;
    method AxiCache awCACHE;
    method AxiProt awPROT;
    method Bool awVALID;

    // Address Inputs
    method Action awREADY(Bool value);

    // Data Outputs
    method AxiId#('TLM_PRM) wID;
    method AxiData#('TLM_PRM) wDATA;
    method AxiByteEn#('TLM_PRM) wSTRE;
    method Bool wLAST;
    method Bool wVALID;
```

```

// Data Inputs
method Action wREADY(Bool value);

// Response Outputs
method Bool bREADY;

// Response Inputs
method Action bID (AxiId#('TLM_PRM) value);
method Action bRESP (AxiResp value);
method Action bVALID(Bool value);
endinterface

```

AxiRdSlave The AxiRdSlave interface receives AXI read requests and returns AXI read responses.

```

interface AxiRdSlave#('TLM_PRM_DCL);
// Address Inputs
method Action arID (AxiId#('TLM_PRM) value);
method Action arADDR (AxiAddr#('TLM_PRM) value);
method Action arLEN (AxiLen value);
method Action arSIZE (AxiSize value);
method Action arBURST(AxiBurst value);
method Action arLOCK (AxiLock value);
method Action arCACHE(AxiCache value);
method Action arPROT (AxiProt value);
method Action arVALID(Bool value);

// Address Outputs
method Bool arREADY;

// Response Inputs
method Action rREADY(Bool value);

// Response Outputs
method AxiId#('TLM_PRM) rID;
method AxiData#('TLM_PRM) rDATA;
method AxiResp rRESP;
method Bool rLAST;
method Bool rVALID;
endinterface

```

AxiWrSlave The AxiWrSlave interface receives AXI write requests and returns AXI write responses.

```

interface AxiWrSlave#('TLM_PRM_DCL);
// Address Inputs
method Action awID (AxiId#('TLM_PRM) value);
method Action awADDR (AxiAddr#('TLM_PRM) value);
method Action awLEN (AxiLen value);
method Action awSIZE (AxiSize value);
method Action awBURST(AxiBurst value);

```

```

method Action awLOCK (AxiLock          value);
method Action awCACHE(AxiCache         value);
method Action awPROT (AxiProt          value);
method Action awVALID(Bool             value);

// Address Outputs
method Bool awREADY;

// Data Inputs
method Action wID   (AxiId#('TLM_PRM)  value);
method Action wDATA (AxiData#('TLM_PRM) value);
method Action wSTRB (AxiByteEn#('TLM_PRM) value);
method Action wLAST (Bool               value);
method Action wVALID(Bool              value);

// Data Ouptuts
method Bool wREADY;

// Response Inputs
method Action bREADY(Bool value);

// Response Outputs
method AxiId#('TLM_PRM) bID;
method AxiResp          bRESP;
method Bool              bVALID;
endinterface

```

The `AxiRdMaster` and `AxiRdSlave` interfaces as well as the `AxiWrMaster` and `AxiWrSlave` interfaces are connectable.

```

instance Connectable#(AxiRdMaster#('TLM_PRM), AxiRdSlave#('TLM_PRM));

instance Connectable#(AxiWrMaster#('TLM_PRM), AxiWrSlave#('TLM_PRM));

```

Fabric Interfaces

When used in the context of a bus or switch, AXI transactor modules must communicate with address decoding logic. As with the BSV implementation of the AHB bus, bus fabric interfaces are provided to support this communication. Unlike the AHB protocol however, with the AXI bus protocol no explicit communication between the arbiter and the master transactor modules is required. Thus the `AxiRdFabricMaster` and `AxiWrFabricMaster` interfaces are simply wrappers around the bus interfaces themselves.

```

interface AxiRdFabricMaster#('TLM_PRM_DCL);
  (* prefix = "" *)
  interface AxiRdMaster#('TLM_PRM) bus;
endinterface

interface AxiWrFabricMaster#('TLM_PRM_DCL);
  (* prefix = "" *)
  interface AxiWrMaster#('TLM_PRM) bus;
endinterface

```

The `AxiRdFabricSlave` and `AxiWrFabricSlave` interfaces each provide an `addrMatch` method which given an AXI address returns an Boolean value indicating whether the given address maps to the associated slave. By polling this method for each slave on the bus, the decoding logic can determine the appropriate destination for each bus transaction.

```
interface AxiRdFabricSlave#('TLM_PRM_DCL);
    (* prefix = "" *)
    interface AxiRdSlave#('TLM_PRM) bus;
    method Bool addrMatch(AxiAddr#('TLM_PRM) value);
endinterface

interface AxiWrFabricSlave#('TLM_PRM_DCL);
    (* prefix = "" *)
    interface AxiWrSlave#('TLM_PRM) bus;
    method Bool addrMatch(AxiAddr#('TLM_PRM) value);
endinterface
```

Transactor Interfaces

Each AXI transactor module provides AXI and TLM interfaces to implement a translation between a stream of TLM operations and the AXI bus protocol. Each transactor has two subinterfaces: a subinterface for the connection with the AXI bus and a subinterface to send and receive TLM objects. The AXI library package includes two master transactor interfaces and two slave transactor interfaces; The `AXIRdMasterXActor` and `AXIWrMasterXActor` interfaces for masters and the `AXIRdSlaveXActor` and `AXIWrSlaveXActor` interfaces for slaves. Since the AXI protocol supports read and write transaction on separate buses, two transactor implementations are required for masters and two implementations for slaves. The AXI subinterface definitions can be found in [section 1](#).

AXIRdMasterXActorIFC The `AXIRdMasterXActorIFC` has two subinterfaces: an `AxiRdFabricMaster` subinterface and a `TLMRecvIFC` subinterface. The associated transactor converts TLM read requests into the AXI protocol, and converts the AXI response back into TLM.

```
interface AXIRdMasterXActorIFC#('TLM_RR_DCL, 'TLM_PRM_DCL);
    interface TLMRecvIFC#('TLM_RR) tlm;
    (* prefix = "" *)
    interface AxiRdFabricMaster#('TLM_PRM) fabric;
endinterface
```

AXIWrMasterXActorIFC The `AXIWrMasterXActorIFC` has two subinterfaces: an `AxiWrFabricMaster` subinterface and a `TLMRecvIFC` subinterface. The associated transactor converts TLM write requests into the AXI protocol, and converts the AXI response back into TLM.

```
interface AXIWrMasterXActorIFC#('TLM_RR_DCL, 'TLM_PRM_DCL);
    interface TLMRecvIFC#('TLM_RR) tlm;
    (* prefix = "" *)
    interface AxiWrFabricMaster#('TLM_PRM) fabric;
endinterface
```

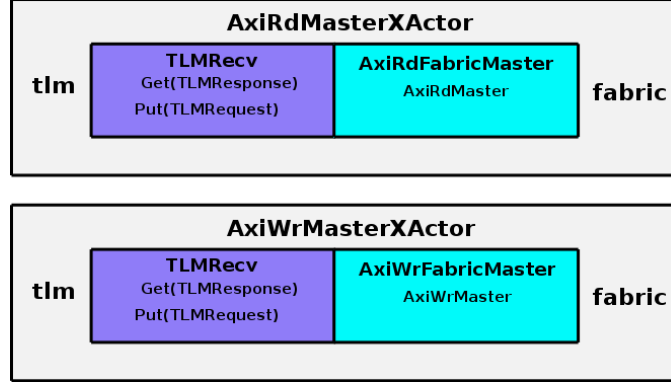


Figure 3: AXIMasterXActor Interfaces (Read and Write Versions)

AxiRdSlaveXActorIFC The **AxiRdSlaveXActorIFC** has two subinterfaces: an **AxiRdFabricSlave** subinterface and a **TLMSeIFC** subinterface. The associated transactor converts an AXI read request into TLM and the TLM response back into the AXI protocol.

```
interface AxiRdSlaveXActorIFC#('TLM_RR_DCL, 'TLM_PRM_DCL);
    interface TLMSeIFC#('TLM_RR)      tlm;
    (* prefix = "" *)
    interface AxiRdFabricSlave#('TLM_PRM) fabric;
endinterface
```

AxiWrSlaveXActorIFC The **AxiWrSlaveXActorIFC** has two subinterfaces: an **AxiWrFabricSlave** subinterface and a **TLMSeIFC** subinterface. The associated transactor converts an AXI write request into TLM and the TLM response back into the AXI protocol.

```
interface AxiWrSlaveXActorIFC#('TLM_RR_DCL, 'TLM_PRM_DCL);
    interface TLMSeIFC#('TLM_RR)      tlm;
    (* prefix = "" *)
    interface AxiWrFabricSlave#('TLM_PRM) fabric;
endinterface
```

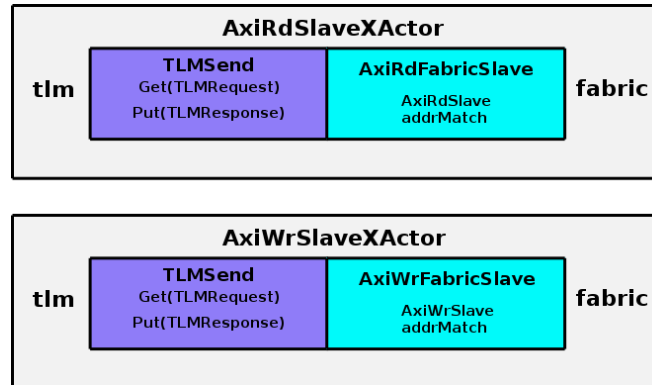


Figure 4: AXISlaveXActor Interfaces (Read and Write Versions)

Modules

The following constructors are used to create AXI transactor modules. Versions with associated synthesis boundaries are also available. These versions are called `mkAxiRdMasterStd`, `mkAxiWrMasterStd`, `mkAxiRdSlaveStd`, and `mkAxiWrSlaveStd`. The specific TLM parameter values for these synthesized versions are as specified by the preprocessor macro `TLM_STD_TYPES`.

mkAxiRdMaster	<p>Creates an AXI master read transactor module. Provides an <code>AxiRdMasterXActorIFC</code> interface.</p> <pre> module mkAxiRdMaster (AxiRdMasterXActorIFC#('TLM_XTR)) provisos(Bits#(req_t, s0), Bits#(resp_t, s1), TLMRequestTC#(req_t, 'TLM_PRM), TLMResponseTC#(resp_t, 'TLM_PRM), DefaultValue#(cstm_type), AxiConvert#(AxiProt, cstm_type) AxiConvert#(AxiCache, cstm_type), AxiConvert#(AxiLock, cstm_type)); </pre>
mkAxiWrMaster	<p>Creates an AXI master write transactor module. Provides an <code>AxiWrMasterXActorIFC</code> interface.</p> <pre> module mkAxiWrMaster (AxiWrMasterXActorIFC#('TLM_XTR)) provisos(Bits#(req_t, s0), Bits#(resp_t, s1), TLMRequestTC#(req_t, 'TLM_PRM), TLMResponseTC#(resp_t, 'TLM_PRM), DefaultValue#(cstm_type), Bits#(cstm_type, s2), AxiConvert#(AxiProt, cstm_type) AxiConvert#(AxiCache, cstm_type), AxiConvert#(AxiLock, cstm_type)); </pre>
mkAxiRdSlave	<p>Creates an AXI slave read transactor module. Provides an <code>AxiRdSlaveXActorIFC</code> interface.</p> <pre> module mkAxiRdSlave#(Integer max_flight, function Bool addr_match(AxiAddr#('TLM_PRM) addr)) (AxiRdSlaveXActorIFC#('TLM_XTR)) provisos (TLMRequestTC#(req_t, 'TLM_PRM), TLMResponseTC#(resp_t, 'TLM_PRM), Bits#(req_t, s0), Bits#(resp_t, s1), Bits#(cstm_type, s2), AxiConvert#(AxiCustom, cstm_type)); </pre>

mkAxiWrSlave	Creates an AXI slave write transactor module. Provides an AxiWrSlaveXactorIFC interface.
	<pre> module mkAxiWrSlave#(Integer max_flight, function Bool addr_match(AxiAddr#('TLM_PRM) addr)) (AxiWrSlaveXactorIFC#('TLM_XTR)) provisos(TLMRequestTC#(req_t, 'TLM_PRM), TLMResponseTC#(resp_t, 'TLM_PRM), Bits#(req_t, s0), Bits#(resp_t, s1), Bits#(cstm_type, s2), AxiConvert#(AxiCustom, cstm_type)); </pre>

The following two module constructors are each used to create an AXI bus fabric. `mkAxiRdBus` is used to create a read bus while `mkAxiWrBus` is used to create a write bus.

mkAxiRdBus	Given a vector of <code>AxiRdFabricMaster</code> interfaces and a vector of <code>AxiRdFabricSlave</code> interfaces, <code>mkAxiRdBus</code> creates an AXI read bus.
	<pre> module mkAxiRdBus#(Vector#(master_count, AxiRdFabricMaster#('TLM_PRM)) masters, Vector#(slave_count, AxiRdFabricSlave#('TLM_PRM))slaves) (Empty) provisos(Log#(master_count, size_m), Add#(slv_count, 1, slave_count), Log#(slave_count, size_s), Add#(ignore0, size_m, id_size), Add#(ignore1, size_s, id_size)); </pre>

mkAxiWrBus	Given a vector of <code>AxiWrFabricMaster</code> interfaces and a vector of <code>AxiWrFabricSlave</code> interfaces, <code>mkAxiWrBus</code> creates an AXI write bus.
	<pre> module mkAxiWrBus#(Vector#(master_count, AxiWrMaster#('TLM_PRM)) masters, Vector#(slave_count, AxiWrSlave#('TLM_PRM)) slaves) (Empty) provisos(Log#(master_count, size_m), Add#(slv_count, 1, slave_count), Log#(slave_count, size_s), Add#(ignore0, size_m, id_size), Add#(ignore1, size_s, id_size)); </pre>

The following module is used to add probe signals for each of the AXI bus signals. This facilitates debugging and waveform viewing of the created bus fabric.

mkAxiMonitor	Adds a probe module for each of the AXI bus signals. The <code>include_pc</code> value indicates whether or not the monitor module should include an instantiation of an AXI protocol checker module (available from ARM). If the protocol checker is not available, the value of <code>include_pc</code> should be set to False.
	<pre>module mkAxiMonitor#(Bool include_pc, AxiWrMaster#('TLM_PRM) master_wr, AxiWrSlave#('TLM_PRM) slave_wr, AxiRdMaster#('TLM_PRM) master_rd, AxiRdSlave#('TLM_PRM) slave_rd) (AxiMonitor#('TLM_PRM));</pre>

Functions

The following functions convert from TLM to AXI

getAxiAddrCmd	Returns an AxiAddrCmd from a TLM RequestDescriptor
	<pre>function AxiAddrCmd#('TLM_PRM) getAxiAddrCmd (RequestDescriptor#('TLM_PRM) tlm_descriptor) provisos(AxiConvert#(AxiProt, cstm_type), AxiConvert#(AxiCache, cstm_type), AxiConvert#(AxiLock, cstm_type));</pre>
getFirstAxiWrData	Returns the AxiWrData from the TLM RequestDescriptor
	<pre>function AxiWrData#('TLM_PRM) getFirstAxiWrData (RequestDescriptor#('TLM_PRM) tlm_descriptor);</pre>
getAxiByteEn	Returns the AxiByteEn from the TLM RequestDescriptor
	<pre>function AxiByteEn#('TLM_PRM) getAxiByteEn (RequestDescriptor#('TLM_PRM) tlm_descriptor);</pre>
getAxiLen	Returns the AxiLen from the TLM burst_length
	<pre>function AxiLen getAxiLen(TLMUInt#('TLM_PRM) burst_length);</pre>

getAxiSize	Returns the AxiSize from the TLM incr
	<pre>function AxiSize getAxiSize(TLMBurstSize#('TLM_PRM) incr);</pre>

getAxiSize	Returns the AxiSize from the TLM incr
	<pre>function AxiSize getAxiSize(TLMBurstSize#('TLM_PRM) incr);</pre>

getAxiBurst	Returns the AxiBurst from the TLM burst_mode
	<pre>function AxiBurst getAxiBurst(TLMBurstMode burst_mode);</pre>

getAxiId	Returns the AxiId from the TLM transaction_id
	<pre>function AxiId#('TLM_PRM) getAxiId(TLMId#('TLM_PRM) transaction_id);</pre>

The following functions convert from Axi to TLM

fromAxiAddrCmd	Returns the TLM RequestDescriptor from the AXI addr_cmd
	<pre>function RequestDescriptor#('TLM_PRM) fromAxiAddrCmd (AxiAddrCmd#('TLM_PRM) addr_cmd) provisos(Bits#(RequestDescriptor#('TLM_PRM), s0), AxiConvert#(AxiCustom, cstm_type));</pre>

fromAxiLen	Returns the TLM burst_length from the AXI len
	<pre>function TLMUInt#('TLM_PRM) fromAxiLen(AxiLen len);</pre>

fromAxiBurst	Returns the TLM burst_mode from the AXI burst
	<pre>function TLMBurstMode fromAxiBurst(AxiBurst burst);</pre>

fromAxiSize	Returns the TLM burst_size from the AXI size
	<code>function TLMBurstSize#('TLM_PRM) fromAxiSize(AxiSize size);</code>
fromAxiId	Returns a TLM id from the AXI id
	<code>function TLMId#('TLM_PRM) fromAxiId(AxiId#('TLM_PRM) id);</code>
fromAxiResp	Returns the TLM status from the AXI resp
	<code>function TLMStatus fromAxiResp(AxiResp resp);</code>