Mentor®
A Siemens Business

Catapult® Interface Components Toolkit

AXI4 Streaming Interfaces

November 2018

Mentor Graphics Corporation
8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777.
Telephone: 503.685.7000
Toll-Free Telephone: 800.592.2210
Website: www.mentor.com
SupportNet: supportnet.mentor.com/

Send Feedback on Documentation: supportnet.mentor.com /doc_feedback_form

Introduction

This document and the accompanying example design illustrate how you can use the AXI4 Streaming Interfaces (**ccs_axi4stream_in**, **ccs_axi4stream_out** and **ccs_axi4stream_pipe**) in a design requiring processing at the end of a stream of data.

Toolkit Contents

This toolkit contains the following files:

AXI4Streams.pdf – This document
src/crc2_types.h – C++ header file defining the data types used in the design
src/crc2_gen.cpp – C++ source defining a "fake" CRC generator
src/crc2_check.cpp – C++ source defining a "fake" CRC checker
src/crc2_top.cpp – C++ source creating a hierarchical design instantiating a generator and checker
src/crc2_tb.cpp – C++ testbench file that exercises the generator and checker

The AMBA AXI4-Stream Protocol

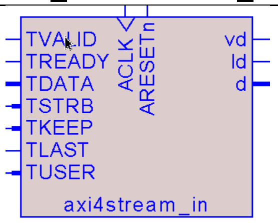
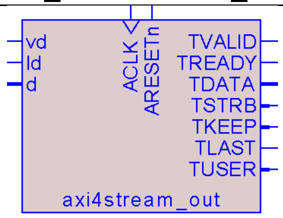
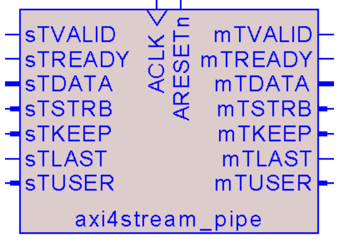
The AXI4-Stream protocol is described in the document "AMBA 4 AXI4-Stream Protocol" version 1.0 available from the ARM website: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0051a/index.html .
Quoting from the specification:

> The AXI4-Stream protocol is used as a standard interface to connect components that wish to exchange data. The interface can be used to connect a single master that generates data, to a single slave, that receives data. The protocol can also be used when connecting larger numbers of master and slave components. The protocol supports multiple data streams using the same set of shared wires, allowing a generic interconnect to be constructed that can perform upsizing, downsizing and routing operations.
> The AXI4-Stream interface also supports a wide variety of different stream types. The stream protocol defines the association between Transfers and Packets.

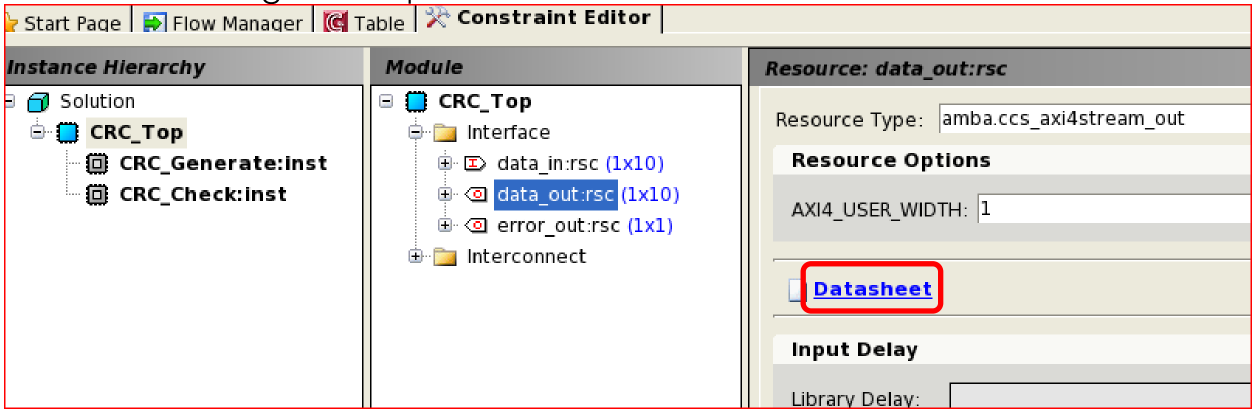
Catapult AXI4 Streaming Interface Components

The AXI4-Stream interfaces provided with Catapult implement the basic AXI4-Stream protocol as single peer-to-peer connections (Master to Slave) in the Continuous Aligned Stream mode. No position or null bytes are allowed. Interface signals include the primary data signal TDATA, the sideband signal TUSER (configurable width), the handshake signals TREADY/TVALID and the end-of-packet signal TLAST. The auxiliary signals TID and TDEST are not used. The configuration signals TSTRB and TKEEP are driven/assert that the Continuous Aligned Stream mode are used.

The following interface components are provided with Catapult:

ccs_axi4stream_in	ccs_axi4stream_out	ccs_axi4stream_pipe
 TVALID TREADY TDATA TSTRB TKEEP TLAST TUSER ACLK ARESETn vd ld d axi4stream_in	 vd ld d ACLK ARESETn TVALID TREADY TDATA TSTRB TKEEP TLAST TUSER axi4stream_out	 sTVALID sTREADY sTDATA sTSTRB sTKEEP sTLAST sTUSER ACLK ARESETn mTVALID mTREADY mTDATA mTSTRB mTKEEP mTLAST mTUSER axi4stream_pipe

Individual datasheets for each component can be found on the Architecture Constraints dialog in Catapult:



Resource Type: amba.ccs_axi4stream_out

Resource Options

AXI4_USER_WIDTH: 1

Library Delay:

Figure 1 Datasheet Links in Architecture Constraints

The interface components assume that the MSB of the data is the TLAST signal. So a natural C++ representation of the AXI4 data is a struct. If the data width is one byte, then the following struct could be used:

```
#define AXI_BUS_WIDTH 8
typedef ac_int<AXI_BUS_WIDTH,false> Data_t;

// Type definition for the datastream I/O
struct Stream_t {
  Data_t    TDATA;
  Bool      TUSER;
  bool      TLAST;
};
```

Since the AXI4 bus width must be an even number of bytes, if the C++ struct defines an odd width then Catapult will instantiate the AXI4 interface with the correct size and zero-fill the unused bits.

A Sample Application

This toolkits provides a simple CRC-like application written in C++. The CRC computation is simply the sum of all of the data values plus the size of the received packet. The design is composed of two blocks in hierarchy, shown in ***Figure 2***:

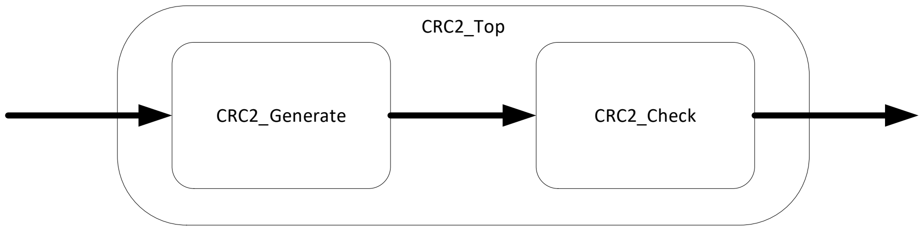


Figure 2 Sample CRC Design Hierarchy

The first block **`CRC_Generate`** takes in a stream of data using an ac_channel using the **`Stream_t`** struct definition presented earlier. A representative stream is shown in ***Figure 3***:

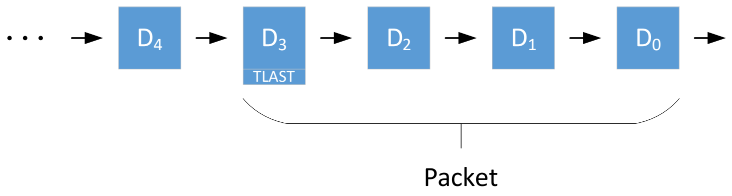


Figure 3 Sample Input Stream

Each data value **D_n** read from the channel is used in the calculation of the CRC value and also passed thru to the output of the block. When the last data is detected (indicated by the TLAST field in the C++ struct Stream_t) then the fake CRC value is sent as the true TLAST. The output stream associated with the input stream of ***Figure 4*** would be:

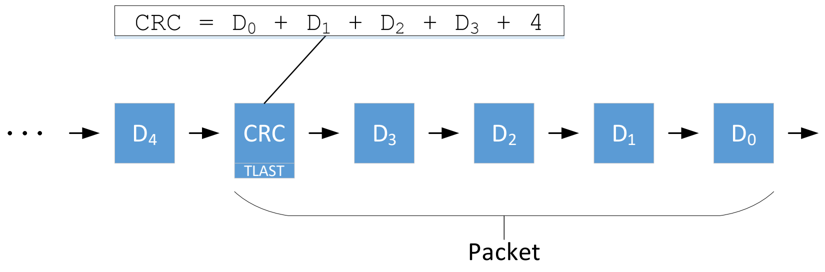
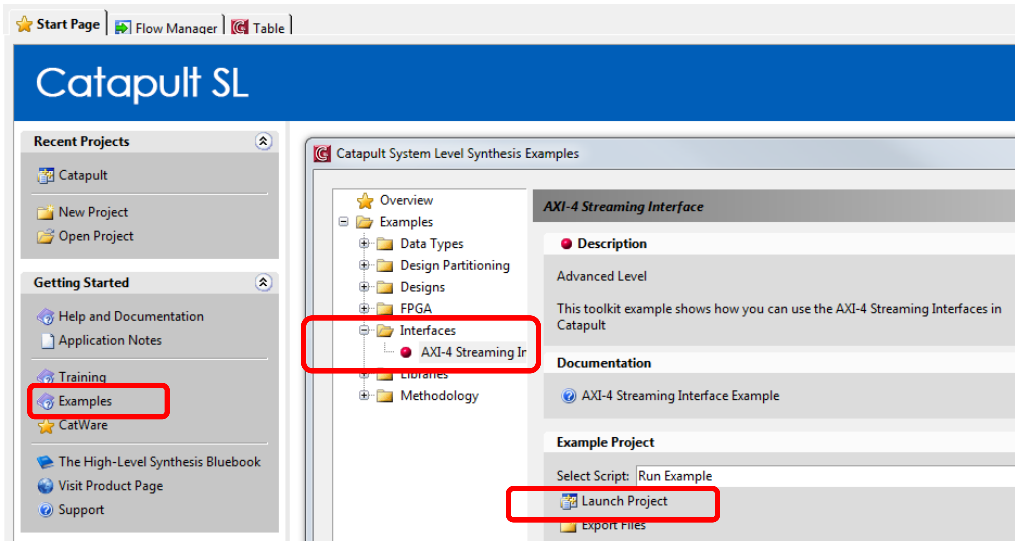


Figure 4 Sample Output from CRC_Generate

The **CRC_Check** block receives the output from **CRC_Generate** and performs the same CRC calculation. When the last data value is received by **CRC_Check**, the calculated CRC value is compared with the received CRC value and a Boolean output is produced if they differ.

Running the Example Design

You can launch Catapult on the example design by selecting the script "Run Example" from the toolkits dialog and clicking "Launch Project" as shown in ***Figure 5***:



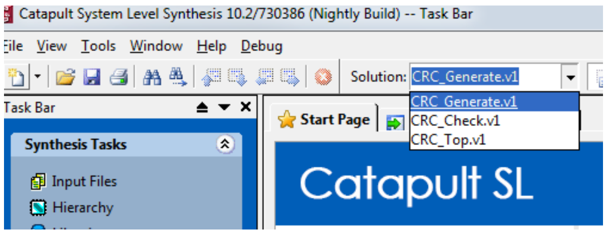
Advanced Level

This toolkit example shows how you can use the AXI-4 Streaming Interfaces in Catapult

Select Script: Run Example

Figure 5 Launching Catapult on the Example Design

This script will generate the CRC example design in a bottom-up fashion. Once Catapult has finished executing the script you can go back and select the different blocks in the Catapult GUI as shown in ***Figure 6***



Catapult System Level Synthesis 10.2/730386 (Nightly Build) -- Task Bar
File View Tools Window Help Debug
Task Bar
Synthesis Tasks
Input Files
Hierarchy
Start Page
Catapult SL

Solution: CRC_Generate.v1

CRC_Generate.v1
CRC_Check.v1
CRC_Top.v1

Figure 6 Selecting Different Blocks

To simulate any of the blocks using SCVerify expand the Verification folder in the Catapult GUI and double-click on the Verify_xxx Makefile. Simulating the CRC_Top block in QuestaSim results in the waveform shown in ***Figure 7***:

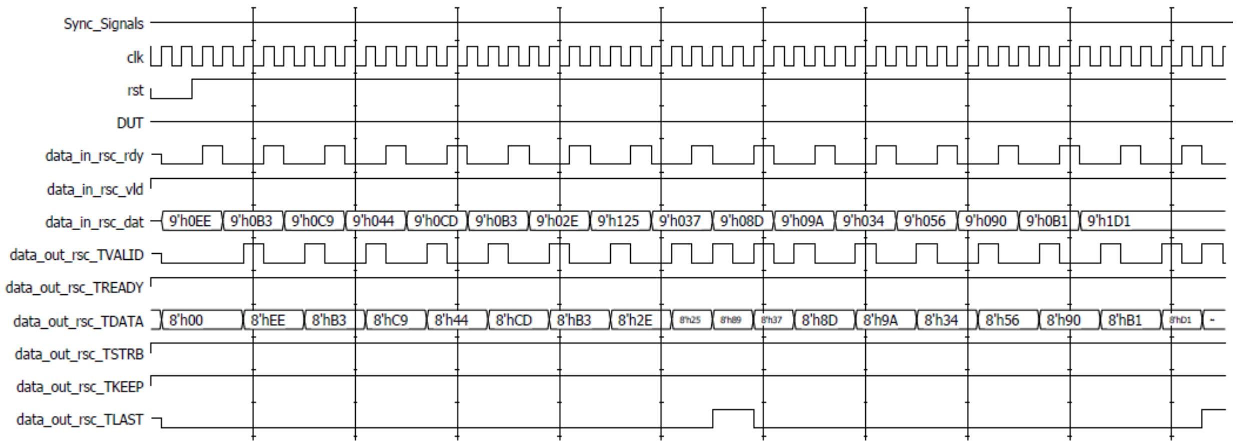


Figure 7 Simulation of CRC_Top

In the simulation you can see 8 data values received on data_in_rsc_dat and the AXI4 stream signals reflecting the same 8 data value plus the CRC checksum value.