Summary - Time Demux

Name	time_demux
Version	1.3.0
Release Date	Feb 2018
Component Library	training.components
Verification Model	None
Implementations	RCC
Tested Platforms	linux-zynq-arm, c7-x86_64

Functionality

The Time Demux component acts as a demultiplexer/router by parsing an `iqstream_with_sync` protocol and routing timestamps and data to separate output ports.

The incoming `iqstream_with_sync` supports three opcodes: `iq`, `Sync`, and `Time`. The output ports use the `iqstream` and `iqstream_with_sync` protocols, the former using a single opcode. Data within the `iqstream_with_sync`'s `Time` opcode (a single 64-bit value) is passed directly through, while data within the `iq` opcode is converted to `iqstream`'s `iq` opcode's data (which, conveniently, is the same structure). The `iqstream_with_sync`'s `Sync` opcode is currently ignored.

Block Diagrams

Top level

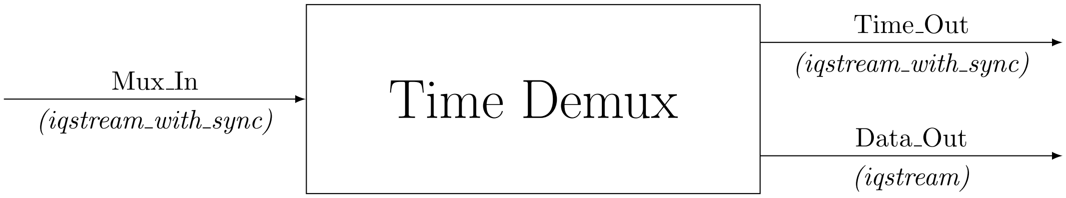

Figure 1: Top-level Block Diagram

RCC source dependency

- training/components/time_demux.rcc/time_demux.cc

Properties

Name	OCS	OWD RCC	OWD HDL	Type	Length	Accessibility	Valid Range	Default	Usage
Current_Second	Property	N/A	N/A	ULong	N/A	Volatile	Default	N/A	Last seen “second” timestamp
Messages_Read	Property	N/A	N/A	ULongLong	N/A	Volatile	Default	N/A	Number of messages seen
Bytes_Read	Property	N/A	N/A	ULongLong	N/A	Volatile	Default	N/A	Total number of bytes read

Ports

Input / Consumer

Port Name	Protocol	Optional	Notes
Mux_In	iqstream_with_sync_protocol	False	Time-stamped 16-bit I/Q data samples (32 bits per sample); `Sync` opcode ignored

Output / Producer

Port Name	Protocol	Optional	Notes
Time_Out	iqstream_with_sync_protocol	False	Only uses 64-bit timestamp in `Time` opcode; `iq` and `Sync` opcodes guaranteed not to be present
Data_Out	iqstream_protocol	False	16-bit I/Q data samples (32 bits per sample)

Control Timing and Signals

N/A; this is an RCC-only component.

Performance and Resource Utilization

HDL

N/A; this is an RCC-only component.

RCC

TBD.

Processor Type	Processor Frequency	Run Function Time
linux-c6-x86_64 Intel(R) Xeon(R) CPU E5-1607	3.00 GHz	TBD
linux-c7-x86_64 Intel(R) Core(TM) i7-3630QM	2.40 GHz	TBD
linux-zynq-arm ARMv7 Processor rev 0 (v7l)	666 MHz	TBD

Test and Verification

This component uses the standard OpenCPI test process. It is one of the few that use multiple ports.

The only currently known issue is that the specfile must define the "Data_Out" port *first* to have the test application use it to signal "done."

Advanced / Detailed Theory of Operation

This section is not essential to understand to perform the training lab.

Testing of the Time Demux component consists of a C++ program (*test_data_generator.cxx*) used to generate input data and expected "golden" outputs (Figure 2).

Fake timestamps and sample data are interleaved into an input file using the "message mode" format required to have *ocpi.file_read* playback opcodes with data. The C++ generator takes input arguments of: input file name, starting timestamp, number of samples to push each "second," filename for the interleaved file, filename for the golden timestamps, and filename for the golden output file. These parameters are all handled by the OpenCPI test XML, with the test application shown in Figure 3.

Output data is compared to the golden file(s) by the Makefile (Figure 4).

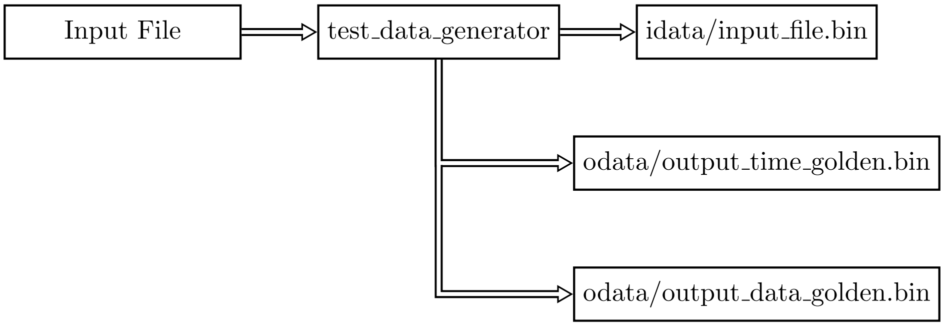

Figure 2: C++ Generator Usage

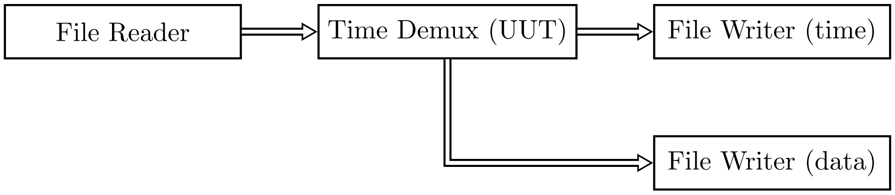

Figure 3: Test Application Layout (*app_time_demux*)

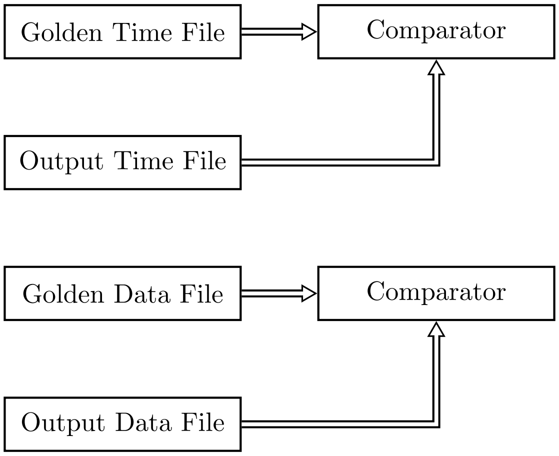

Figure 4: Testing

The default XML provides reasonable values for each of the data generator's required parameters:

Test Property	Default	Notes
IFILE	(Running kernel image) *e.g.* `/boot/vmlinuz-3.10.0-327.4.4.el7.x86_64`	The input file is truncated to a 32-bit boundary to simulate two 16-bit data samples. This may cause verification issues if using random files.
START	0	Decimal only
SAMPLES	256	Should not exceed 2048

For test purposes, the "timestamp" is a one-up counter starting at START placed in the upper 32-bits and then incremented and placed into the lower 32-bits:

```
$ od -t x8 odata/output_time_golden.bin | head
0000000 0000000000000001 0000000100000002
0000020 0000000200000003 0000000300000004
0000040 0000000400000005 0000000500000006
0000060 0000000600000007 0000000700000008
0000100 0000000800000009 000000090000000a
0000120 0000000a0000000b 0000000b0000000c
0000140 0000000c0000000d 0000000d0000000e
0000160 0000000e0000000f 0000000f00000010
0000200 0000001000000011 0000001100000012
0000220 0000001200000013 0000001300000014
```

By default, output golden data is written as *INPUTFILE_gold_data* and *INPUTFILE_gold_time* in the same path as *INPUTFILE*. The *INPUTFILE* named here is the interleaved file whose name is provided by the test framework and is the **output** of *test_data_generator*, based on the IFILE Property.