OpenCPI
SDP Buffer Workaround Notes

OpenCPI Release: v2.4.8

Revision History

Revision	Description of Change	Date
1.0	Creation	2023-09-22
1.1	Update rcc platform versions	2025-02-17

1 Overview

This document describes a workaround for increasing buffer size and count for Scalable Data Plane (SDP) implementation. SDP provides a common transport for interfacing OpenCPI HDL workers to external FPGA elements using interconnects such as AXI, PCI Express and Ethernet. The current framework limits the overall memory size allocated for buffers to 64 Kbytes and maximum buffer count to 8. A user may desire the flexibility to either increase or decrease buffer size or count to meet specific throughout and latency requirements.

The intent of these notes is to provide a quick reference to those who may want to increase buffer size and count outside of the default properties currently supported by the framework. Specifically, transporting data from RCC to or from HDL components. The suggestions, as described in these notes, are currently only valid for the Zed platform.

The primary device workers that establish SDP buffers are the ‘sdp_send.hdl’ and ‘sdp_receive.hdl’ workers. The ‘sdp_send.hdl’ worker currently supports a maximum buffer size of 32kB and the ‘sdp_receive.hdl’ worker a maximum buffer size of 16kB. These restrictions appear to be imposed by runtime and opcigen functionality. More specifically attributed to the transport and HDL container implementations. The OpenCPI team is currently investigating the premise of these restrictions and working to devise a more configurable solution.

Summary

The table below shows the achievable buffer size and counts for each SDP device worker when the suggested workaround is applied for Zed platform operation.

		Zed Platform Only	
Path	**Device Worker**	**Max Buffer Size (KB)**	**Max Buffer Count**
HDL → RCC	sdp_send.hdl	32	16
RCC → HDL	sdp_receive.hdl	16	15

2 Verification Method

Verification was conducted using the ‘test_source’ and ‘test_sink’ RCC and HDL component workers found in the OpenCPI assets project ‘util_comps’ library.

~/opencpi/projects/assets/components/util_comps/test_source.rcc

~/opencpi/projects/assets/components/util_comps/test_source.hdl

~/opencpi/projects/assets/components/util_comps/test_sink.rcc

~/opencpi/projects/assets/components/util_comps/test_sink.hdl

The components are instantiated in the ‘test_source_sink.xml’ application and executed running the ‘run_source_sink’ script both found in the assets project application directory.

~/opencpi/projects/assets/applications/test_source_sink.xml

~/opencpi/projects/assets/applications/run_source_sink

The ‘run_source_sink’ script provides arguments to set source and destination platform, buffer size, and source and destination buffer count.

run_source_sink <from-platform> <to-platform> <message-size> <from-buffer-count> <to-buffer-count> <suppress-write> <cache-mode> <clock-divisor>

The ‘suppress-write’ and ‘cache-mode’ arguments are set to 0 for all cases. The ‘clock-divisor’ contributes to reducing the number of overruns and adjusted to the throughput capacity of the platform.

The assemblies for the HDL components are:

~/opencpi/projects/assets/hdl/assemblies/test_source_assy/
test_source_assy.xml

~/opencpi/projects/assets/hdl/assemblies/test_source_assy/test_sink_assy.xml

RCC Platforms:

- ubuntu22_04
- xilinx24_1_aarch32

HDL Platforms:

- zed

3 Workaround

At this time, the suggestions in this documentation are applicable only to applications executed on the ZedBoard, the OpenCPI ‘zed’ platform, and requires manual modification to framework code and component specification properties. Because HDL→ RCC and RCC→HDL transport exhibit different characteristics, each will be addressed separately.

Available memory for buffers is defined by ‘sdp-properties.xml’. By default, the total available memory is set to 64 KBytes. Total memory may be increase by modifying the ‘memory_bytes’ property in this specification. Actual available memory will be limited to the BRAM resources of the FPGA. The Zedboard hosts a Xilinx Zynq-7020 with 140 36Kb RAM blocks. A developer will need to consider the required memory for any components and component count for their specific application which may restrict the available memory for transport buffering.

The following shows the BRAM memory used for the execution of the verification application described in section 2.

- Memory bytes = 256k, BRAM = 46% (65 blocks)
- Memory bytes = 512k, BRAM = 92% (129 blocks)

The allocated memory for SDP buffer size and count is set in the following specification using 256k as an example:

~/opencpi/projects/core/hdl/devices/specs/sdp-properties.xml

- *<property name='memory_bytes' parameter='1' readback='1' default='256k'/>*

To support the increased buffer size, the SMB size defined by the Zynq ‘system.xml’ file must also increase accordingly. This file may be modified directly on the Zynq host or more in the default platform file so that it will persist when loading platform. For case above, change file as shown below:

~/opencpi/platforms/zynq/zynq_system.xml

- *<transfer smbsize=’512k’>*

The OpenCPI framework also imposes a restriction on the number of buffers to a maximum number of 8 that may be used for SDP transport. The OpenCPI team has identified the ‘MAX_READS_OUTSTANDING’ variable defined in the runtime hdl include folder that is hard set to the value of 8 and used as a limit condition to test number of ports. If port number is greater than 8, then an exception is thrown and the application faults. The ‘MAX_READS_OUTSTANDING’ is defined by the ‘XID_WIDTH’ variable of 3 also defined in the ‘HdlSdp.hh’ include file. The OpenCPI team speculates that this is hard coded to 3 to coincide with the 3 least significant bits (LSB) of the AXI Transaction Identity signal. The 3 most significant bits (MSB) appear to be allocated for SDP node identification. As a “workaround only”, at this time, the limit condition must be manually changed to 16 to increase the use of buffer count greater than 8 without

generating an application fault. More investigation is required into the runtime functionality of OpenCPI to resolve this issue.

The current recommendation is to modify the following file:

~/opencpi/runtime/hdl/src/HdlContainer.cc

Comment out line 590:

- *// limit = std::min(maxBuffers, SDP::MAX_READS_OUTSTANDING);*

Replace with:

- *limit = 16;*

This bypasses the conditional check and will allow an application to continue to execute.

To apply change for either the Ubuntu or Zynq host, the framework must be rebuilt. This can be achieved by executing the following commands at the opencpi directory level:

- ~/opencpi$ make framework Platforms=ubuntu22_04
- ~/opencpi$ make framework Platforms=xilinx24_1_aarch32

The ‘sdp_send.hdl’ device work requires that the local buffer count constant value in the VHDL implementation code to be modified. This value is also associated with the ‘MAX_READS_OUTSTANDING’ defined in ‘HdlSdp.hh’.

The current recommendation is to modify the following file:

~/opencpi/projects/core/hdl/devices/sdp_send.hdl/sdp_send.vhd

Comment out line 42:

- *constant max_lcl_buffers_c : natural := ocpi.util.min(max_rem_buffers_c, max_reads_outstanding);*

Replace with:

- *constant max_lcl_buffers_c : natural := 16;*

CAVEAT 1: It is assumed that someone implementing this workaround is already familiar with building device workers, platforms, components, and assemblies. Also, the OpenCPI user is familiar with executing OpenCPI application and in the various available modes.

CAVEAT 2: It is stressed that the suggestions provided by this documentation are a workaround to the OpenCPI framework to increase buffer size and count. It is not intended to be a long term solution and is temporary until further understanding is attained and permanent solution implemented.

4 Findings

Verification was accomplished using the application described in Section 2. More specifically, verification of the SDP transport was limited to the OpenCPI Zed, Ubuntu22_04, and Xilinx24_1_aarch32 platforms. The application was executed in the remote server mode to verify maximum achievable buffer size and buffer count. Results indicate that implementation of the 'sdp_receiver' and 'sdp_send' device workers each exhibit different performance characteristics as described below.

HDL → RCC

When executing the verification application to transport data from the Zed platform to either the Ubuntu22_04 or Xilinx24_1_aarch32 platform, the maximum settable buffer size is 32kB and maximum buffer count is 16. Improvement in number of overruns was seen with increase buffer count.

[zed → ubuntu22_04]

- zed (8 buffers) => ubuntu22_04 (8 buffers), buffer size 32k, notouch 0: MB/s 30.94 MS/s 7.73 Overrun: 131085 Source Msps 7.69230747 (Clock divisor 13)
- zed (16 buffers) => ubuntu22_04 (16 buffers), buffer size 32k, notouch 0: MB/s 30.79 MS/s 7.70 Overrun: None Source Msps 7.69230747 (Clock divisor 13)

[zed → xilinx24_1_aarch32]

- zed (8 buffers) => xilinx24_1_aarch32 (8 buffers), buffer size 32k, notouch 0: MB/s 30.92 MS/s 7.73 Overrun: 131085 Source Msps 7.69230747 (Clock divisor 13)
- zed (16 buffers) => xilinx24_1_aarch32 (16 buffers), buffer size 32k, notouch 0:MB/s 30.77 MS/s 7.69 Overrun: None Source Msps 7.69230747 (Clock divisor 13)

RCC→HDL

When executing the verification application from either the Ubuntu22_04 or Xilinx24_1_aarch32 platforms to the Zed Platform, the maximum settable buffer size is 16kB and maximum buffer count is 15. In the case of transporting from a host platform to Zed, throughput is improved with less buffers. The OpenCPI team will continue to investigate increasing buffer size for this case.

[ubuntu24_04 → zed]

- ubuntu24_04 (4 buffers) => zed (4 buffers), buffer size 16k, notouch 0: MB/s 34.82 MS/s 8.71 Overrun: None Source Msps 100 (Clock divisor 1)
- ubuntu24_04 (8 buffers) => zed (8 buffers), buffer size 16k, notouch 0: MB/s 33.09 MS/s 8.27 Overrun: None Source Msps 100 (Clock divisor 1)
- ubuntu24_04 (15 buffers) => zed (15 buffers), buffer size 16k, notouch 0: MB/s 29.18 MS/s 7.30 Overrun: None Source Msps 100 (Clock divisor 1)

[xilinx24_1_aarch32 → zed]

- xilinx24_1_aarch32 (4 buffers) => zed (4 buffers), buffer size 16k, notouch 0: MB/s 54.11 MS/s 13.53 Overrun: None Source Msps 100 (Clock divisor 1)
- xilinx24_1_aarch32 (8 buffers) => zed (8 buffers), buffer size 16k, notouch 0: MB/s 54.02 MS/s 13.50 Overrun: None Source Msps 100 (Clock divisor 1)
- xilinx24_1_aarch32 (15 buffers) => zed (15 buffers), buffer size 16k, notouch 0: MB/s 53.91 MS/s 13.48 Overrun: None Source Msps 100 (Clock divisor 1)