BEAM

Dune NVMe Storage

Test Software

Project	DuneNvme
Date	2020-06-16
Reference	DuneNvmeStorageTestSoftware
Version	1.0.0
Author	Dr Terry Barnaby

Table of Contents

1. Introduction

This document covers the Dune NvmeStorage test software. This is basic test and development software that is able to exercise the NvmeStorage FPGA core. There is also doxygen generated documentation for this test software.

The software has been written for Fedora31 Linux in C++ and includes a simple Linux kernel driver that provides access to the DuneNvmeTest FPGA design.

2. Usage

The test software is in the DuneNvme/test directory in the Git source tree. The test_nvme program is a simple command line program. It offers the following command line interface:

Usage: test_nvme [options] <testname>

Option	Default	Description
-help,-h		Print out usage information
-v		Verbose. More than one “-v” can be supplied for greater verbosity
-no-reset or -nr		Normally the program will reset the NvmeStorage and NVMe’s on startup. Using this option disable this reset and Nvme configuration

		stage.
-no-validate or -nv		Disable data validation. Used for more accurate performance testing
-l		List tests
-d <nvmeNum>	2	Nvme to operate on: 0: Nvme0, 1: Nvme1, 2: Both Nvme's
-s <block>	0	The starting 4k block number
-n <num>	1	The number of blocks to read/write or trim
-rs <block>	0	The starting 4k block number for reads in captureAndRead
-rn <num>	2	The number of blocks for reads in captureAndRead
-o <filename>	None	The filename for output data.

The main commands include:

Command	Description
capture	Performs data input from FPGA TestData source writing the data into the Nvme's.
captureRepeat	Performs continuous captures into alternate areas of the NVMe's. Note that this does not wait for Trim/Deallocate processes to finish so capture rates slow up.
read	Read data from Nvme's validating it against the test pattern. The data can also be stored in a file if wanted. Partial block contents will be printed out if the "-v" option is provided and full block contents if two or mode "-v"'s are given.
captureAndRead	Perform data input from FPGA TestData source into Nvme's and read data at the same time.
write	Write data to Nvme's. This manually writes data blocks, containing an incrementing 32bit value, to the Nvme devices.
trim	Trim/deallocate blocks on Nvme's using the NVMe's Dataset management functionality.
trim1	Trim/deallocate blocks on Nvme's using the NVMe's Write 0's functionality.
regs	Display the NvmeStorage's registers
info	Display information on the NVMe device
test*	Collection of miscellaneous tests. See source code for details.

The “-d <n>” options selects which Nvme to communicate with. The default, 2, means both Nvme devices. Note some tests only work with a single Nvme or both Nvme’s.

Typical commands are:

1. **Device Driver:** Load Linux device driver resetting FPGA: “make driver_load”
2. **Capture:** “test_nvme -d 2 -s 0 -n 100000 capture”: This will capture and write 100000 4k Blocks of TestData to both Nvme’s starting at block 0.
3. **Read:** “test_nvme -d 2 -s 0 -n 1000 read”: This will read 1000 4k Blocks of TestData from both Nvme’s starting at block 0. The bocks contents will be validated against the expected TestData pattern (Incrementing 32bit value).

Any errors when running the command will be printed out and performance figures given. The “-v” verbose option will print out more information.

There is a **test.sh** shell script utilising the **test_nvme** command that provides some usage scenario’s.

Some example output from **test.sh** commands include:

```
nvmeCapture: Write FPGA data stream to Nvme devices. nvme: 2 startBlock: 0x00000000 numBlocks: 52428800
20:38:38.282: StartBlock:        0 ErrorStatus: 0x0, DataRate: 4530.139 MBytes/s, PeakLatancy:    10535 us
20:39:41.226: StartBlock: 52428800 ErrorStatus: 0x0, DataRate: 4582.337 MBytes/s, PeakLatancy:     3764 us
20:41:53.715: StartBlock:        0 ErrorStatus: 0x0, DataRate: 1782.111 MBytes/s, PeakLatancy:    25342 us
20:43:00.634: StartBlock: 52428800 ErrorStatus: 0x0, DataRate: 4390.204 MBytes/s, PeakLatancy:    11379 us
20:45:14.781: StartBlock:        0 ErrorStatus: 0x0, DataRate: 1761.116 MBytes/s, PeakLatancy:   216792 us
20:46:19.346: StartBlock: 52428800 ErrorStatus: 0x0, DataRate: 4425.614 MBytes/s, PeakLatancy:    12385 us
```

3. Building the Software

the software uses a simple Make based build system. To build the software:

1. Change to the test directory: “cd test”
2. Build the Linux FPGA driver: “make driver”
3. Build the test program: “make”

4. Bfpga driver Notes

The Beam bfpga Linux device driver has been developed from one of Beam’s internal device drivers. It is a simplistic driver that provides access to an FPGA based module for testing purposes.

It expects the FPGA design to use a Xilinx XDMA IP core for host communications and is designed to aid testing of the host to FPGA interface and FPGA designs. Being simple it is easy to debug the communications.

It supports a simple memory mapped register interface that can be mapped to the user space applications memory area. The Xilinx XDMA IP core provides an AXI4-Lite interface for this on the FPGA.

It also supports up to 8 DMA channels. These unidirectional DMA streams can be configured to function in either direction. The Xilinx XDMA IP provides up to 8 AXI4 streams on the FPGA. The driver creates a physically contiguous memory region for each DMA channel of a fixed size. This simplifies the DMA as multiple scatter/gather regions are not needed. For simplicity data is coped from/to the applications user space memory from these regions by the driver. It would be possible to map these regions into the applications memory if wanted for greater performance.

The Xilinx XDMA PCIe IP core manual, PG195, should be looked at for information on the hardware interface that this driver uses.

As stated it is simple, lots of improvements could be had, but it is relatively easy to use and to debug because of its simplicity.

It has been built and tested on the Linux kernel 5.6.11-200.fc31.x86_64 and some later kernels as used in Fedora31.

5. Software Notes

The software consists of the following files:

BeamLibBasic.h, BeamLibBasic.cpp	Miscellaneous utility classes and functions based on the BeamLib system.
NvmeAccess.h, NvmeAccess.cpp	Nvme access class which interfaces to the NvmeStorage FPGA module over the bfpga Linux driver and Xilinx PCIe XDMA module.
test_nvme.cpp	The main test program providing test functionality
test.sh	Simple BASH script showing usage of the test_nvme program
test_deallocate.sh	Simple BASH script with some performance tests with deallocate

The test_nvme program memory maps in the NvmeStorage registers for simple direct access. It uses two XDMA DMA channels connected to the NvmeStorage AXI4 streams for bi-directional communications directly to the Nvme devices. Over these streams PCIe request and reply packets are sent matching the Xilinx PCIe gen3 modules packet structures with some extensions for the NvmeStorage system. See the DuneNvmeStorageDesign manual for details.

The NvmeAccess class supports requests and responses from the Nvme devices by running a thread to process them.