The documentation for this test module:
https://github.com/Xilinx/linux-xlnx/blob/master/Documentation/driver-api/dmaengine/dmatest.rst
The source code for this test module:
https://github.com/Xilinx/linux-xlnx/blob/master/drivers/dma/dmatest.c

The instructions to build this test driver are included in the link for documentation.

This test driver allows you to test a variety of parameters, but no default tests are created; all tests you want to perform have to be created yourself.
The parameters you can specify are as follows (can view this with modinfo dmatest after the driver is available to use):

```
root@testpetalinux:~# modinfo dmatest
name:           dmatest
filename:       (builtin)
license:        GPL v2
file:           drivers/dma/dmatest
author:         Haavard Skinnemoen (Atmel)
parm:           test_buf_size:Size of the memcpy test buffer (uint)
parm:           device:Bus ID of the DMA Engine to test (default: any) (string)
parm:           threads_per_chan:Number of threads to start per channel (default:
1) (uint)
parm:           max_channels:Maximum number of channels to use (default: all)
(uint)
parm:           iterations:Iterations before stopping test (default: infinite) (uint)
parm:           dmatest:dmatest 0-memcpy 1-memset (default: 0) (uint)
parm:           xor_sources:Number of xor source buffers (default: 3) (uint)
parm:           pq_sources:Number of p+q source buffers (default: 3) (uint)
parm:           timeout:Transfer Timeout in msec (default: 3000), Pass -1 for
infinite timeout (int)
parm:           noverify:Disable data verification (default: verify) (bool)
parm:           norandom:Disable random offset setup (default: random) (bool)
parm:           verbose:Enable "success" result messages (default: off) (bool)
parm:           alignment:Custom data address alignment taken as 2^(alignment)
(default: not used (-1)) (int)
parm:           transfer_size:Optional custom transfer size in bytes (default: not
used (0)) (uint)
parm:           polled:Use polling for completion instead of interrupts (bool)
parm:           run:Run the test (default: false)
parm:           channel:Bus ID of the channel to test (default: any)
parm:           test_list:Print current test list
parm:           wait:Wait for tests to complete (default: false)
```

Possible DMA operations to test w/ explanations:

- Memcpy - transferring a block of bytes from one location to another
- Memset - setting the value of entire block of bytes to a specified value
- XOR - xoring blocks of bytes specified (used for generating checksums, etc)
- RAID6 P+Q - special ops related to ensuring data integrity for certain data storage mediums

These parameters are set through writing to these different files:

```
root@testpetalinux:~# ls /sys/module/dmatest/parameters/*
/sys/module/dmatest/parameters/alignment
/sys/module/dmatest/parameters/channel
/sys/module/dmatest/parameters/device
/sys/module/dmatest/parameters/dmatest
/sys/module/dmatest/parameters/iterations
/sys/module/dmatest/parameters/max_channels
/sys/module/dmatest/parameters/norandom
/sys/module/dmatest/parameters/noverify
/sys/module/dmatest/parameters/polled
/sys/module/dmatest/parameters/pq_sources
/sys/module/dmatest/parameters/run
/sys/module/dmatest/parameters/test_buf_size
/sys/module/dmatest/parameters/test_list
/sys/module/dmatest/parameters/threads_per_chan
/sys/module/dmatest/parameters/timeout
/sys/module/dmatest/parameters/transfer_size
/sys/module/dmatest/parameters/verbose
/sys/module/dmatest/parameters/wait
/sys/module/dmatest/parameters/xor_sources
```

I used the following script for the initialization of parameters for the test of both device trees:

```
modprobe dmatest
echo 2000 > /sys/module/dmatest/parameters/timeout
echo 1 > /sys/module/dmatest/parameters/iterations
echo "" > /sys/module/dmatest/parameters/channel
echo 1 >/sys/module/dmatest/parameters/run
```

In my regression test for the FPD DMA, I choose to only verify that it works with a single test. This is for the sake of time
and because I only know so much about DMAs, so I can't be certain that any tests I write are actually thorough enough. However,
the test I do perform for each channel of the DMA ensures that it is at least somewhat functional.

Verification that the appropriate DMA has been removed is shown by looking at the device tree listing in both

the baseline and after FPD DMA removed directories (the specific devices to remove can be checked in the
idenfiying devices to remove section). After verification, you can see that the parameter channels,
when set with "", only includes a single DMA this time after the LPD DMA has been removed. This double checks
that the LPD DMA has been successfully removed.

Additionally, the FPD DMA can still be seen in the test output of after_LPD_DMA_removed to be fully functional,
as each channel passes a test successfully. From this, I conclude that the LPD DMA has been successfully removed
and that the FPD DMA has remained functional.