DMA Demo Design for the Arria 10 Demo Board from DreamChip

Thomas Zerrer
Smartlogic GmbH

Version History

Version	Date	Note
0.1	03 / 02 / 2020	Initial version
1.0	03 / 06 / 2020	Added more feedback to pages 2, 4 and 8
1.1	03 / 13 / 2020	Added index, changed bits of GPO Register, added extended tag note, optimized EOF interrupt section, added chapter 13 “Bitstream ”
1.2	03 / 20 / 2020	Changed chapter 4 PCIe IDs
1.3	03 / 23 / 2020	Added several clarifications / corrections
1.4	04 / 03 / 2020	Added more programming sequences, changed FIFO depth description, added interrupt mapping tables, updated tables for the case when only 1 MSI vector is negotiated, added new GPO Bits in chapter 7, added chapter 9 with MSI-X interrupts, changed open issues to NONE
1.5	04 / 06 / 2020	Re-arranged interrupt mapping in chapter 6. Added DMA Read EOF interrupts to table. Updated programming sequence for EOF interrupts
1.6	04 / 24 / 2020	Added clarifications for the loopback mode and the test pattern generator parameters
1.7	05 / 12 / 2020	Added further details
1.8	06 / 03 / 2020	Modified FIFO reset, TPG reset and added chapter 11 and 13
1.9	06 / 09 / 2020	Clarified tdest_width (Table 5-1). Clarified step 10 in Table 6-1, clarified Table 7-1 and Table 6-8. Moved the MSI Interrupt section to the new chapter 8
1.9.1	06 / 17 / 2020	Clarified EOF assertion
1.10	06 / 19 / 2020	Changed Table 6-1 (step 10) because of FPS problem
1.11	06 / 29 / 2020	Changed chapter 12 (removed debugging screenshots) and 13 (added filename of MAX10 Bitstream)
1.12	07 / 08 / 2020	Clarified version 7 in Table 14-1
1.12.1	07 / 08 / 2020	Fixed typos and formatting
1.13	08 / 06 / 2020	Added smooth video and video format switches, optimized Table 6-7 Renamed chapter 11 from “open issues” to “Important Notes” and added further details to this chapter.
1.14	08 / 12 / 2020	Corrected the sequence in Table 6-1 (1D/2D mode moved up)
1.14.1	10 / 26 / 2020	Add note about JTAG programming of Dream Chip A10 development kit
1.14.2	10 / 28 / 2020	Add Revision 9 to version table (Chapter 14)
1.15	12 / 08 / 2020	Changes for Revision 10 : Added TPG 6 and 7 in chapter 1, changed Tables 2-1 and 3-2, changed EOF behaviour from physical interface to logical channel (table 6-5)
1.15.1	12 / 12 / 2020	Chapter 1 : Added TPG 6 and 7 added Tables 1-1 and 1-2, changed Tables 2-1 and 3-2, major changes in chapter 6 in all tables, changed all tables in chapter 7, Changed table 14-1 (added Revision 10)
1.15.2	01 / 16 / 2021	Corrected Tables 2-1, 5-1, 6-8, 9-1 and 14-1
1.16	03 / 18 / 2021	Added version 11 to table 14-1 (added user interrupts to design) and plug and play mechanism (ROM Tables)
1.17	05 / 28 / 2021	Added note to Table 3-1, Added Revision C to Table 14-1

Table of Contents

1 Overview and Structure of the Arria 10 Demo Design

This document describes the functionality of the FPGA demo bitstream on the Intel Arria 10 FPGA evaluation board from DreamChip. The selected hardware platform provides a reliable PCI Express (PCIe) link at Gen3 link speed and x4 link width. The demo design design realizes its main functions with test picture generators and is based on the Smartlogic High Channel Count (HCC) DMA IP Core for PCI-Express. Details about the HCC IP core can be found in the data sheet on the Smartlogic webpage : https://www.smartlogic.de/en/produkt/high-channel-count-dma-ip-core/

The demo bitstream is provided and has the name DMA_Demo_A10.sof

Structure of the Demo Design

The design (Figure 1-1) contains 8 test picture generators that realize UHD and FHD pictures at 60 frames per second. Furthermore, there are 7 header generators (also known as metadata generators) that produce 32 bytes of header information for 7 DMA channels. There is a static option for loopback operation, where the read data can be looped back directly to the DMA-Write engine.

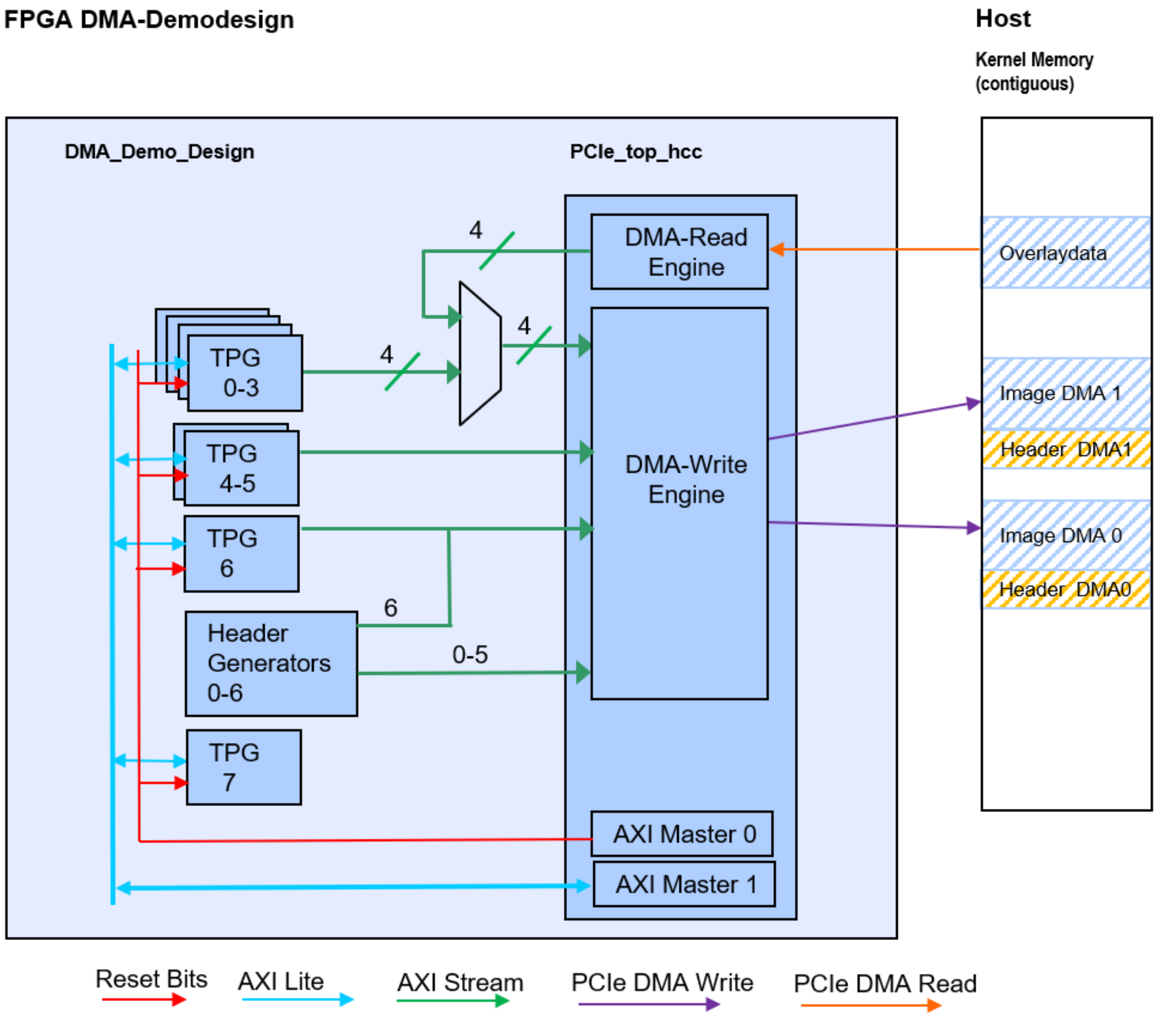


Figure 1-1: Block diagram of the design.

The most generic case is that the headers are stored somewhere in host memory. For this application image and header share the same memory, where the header is stored in the first 4kB of the buffer. Other applications that have no need for a header channel should also be considered.

The header generators transfer their 32 bytes of metadata automatically when the associated TPG transmits the last pixel of a frame to the IP core.

Memory Map

BAR 0

BAR 0 – Offset - size	
0x0 – 32 kB	DMA IP Core specific registers*
0x8000 – 4B	Reset Flag register (see Table 6-7)
0x8004 – 32764 Bytes	reserved

Table 1-1

*Details on the HCC DMA IP Core specific registers can be found in the HCC IP Core User Guide. It is available from Smartlogic under NDA.

BAR 1

BAR 1 – Offset – Size	**Peripheral**
0x0000 – 4kB	TPG0
0x1000– 4kB	TPG1
0x2000– 4kB	TPG2
0x3000– 4kB	TPG3
0x4000– 4kB	TPG4
0x5000– 4kB	TPG5
0x6000– 4kB	TPG6
0x7000– 4kB	TPG7
0x8000– 4B	Demo Design Specific register

Table 1-2

2 Test Pattern Generators

Types of TPGs (Table 2-1)

TPG #	TPG Type	Data Throughput	BAR 1 mapping – Offset
0	UHD, 60 fps	2 GB/s	0x0000
1*	UHD, 61 fps	2 GB/s	0x1000
2	FHD, 60 fps	500 MB/s	0x2000
3	FHD, 60 fps	500 MB/s	0x3000
4	FHD, 60 fps	500 MB/s	0x4000
5*	FHD, 61 fps	500 MB/s	0x5000
6	UHD, 120 fps	4 GB/s	0x6000
7	FHD, 60 fps	500 MB/s	0x7000

Table 2-1

* These TPGs have no vertical blanking zone in order to see how the throughput behaves, with a continuous density of data packets. Therefore, the framerate is slightly higher.

Note: The demo evaluation board has only a throughput of approximately 3 GB/s. In case that the sum of generated data per second is greater than 3 GB/s no data loss will occur, but the framerate will decrease.

Register Map for each TPG (Table 2-2)

Offset	Register Name	Description	Comment
0	Trigger	Bit 0: trigger as master Bit 1: enable trigger Bit 2: ‘1’ = single picture, ‘0’ = continuous picture Bit 3: master (1), slave (0)	Single master trigger: 0xF Continuous master trigger: 0xB Slave enable: 0x2 slave behaves like master
4	Grid Width	Bits 15:0 define the grid width of the grid	Recommended value: 0x80 or higher 0 = no grid
8	Grid Height	Bits 15:0 define the grid height of the grid	Recommended value: 0x80 or higher 0 = no grid
12	Line Width	Bits 7:0 define the grid thickness	Recommended value: 0x4
16	Line Grey Value	Bits 7:0 define the Grey value of the grid	Recommended value: 0x80

Table 2-2

All registers are read/write.
The default test picture is a static 32-bit RGB test picture (8 bits for red, 8 bits for green and 8 bits for blue, 8 Bits don’t care) with a programmable grid. In order to provide some dynamic behaviour, the even frames are inverted, and the odd frames contain non-inverted data.

It is possible to configure a 30-bit pixel colour format with 10 bits for each colour and it is possible to change the frame sequence to a colour changing sequence for better visualization. See Table 7-1 for details where the 2 configuration bits are located.

Test Pictures

Below one finds two samples of test pictures (Figure 2-1: Test picture UHD and Figure 2-2: Test picture FHD).

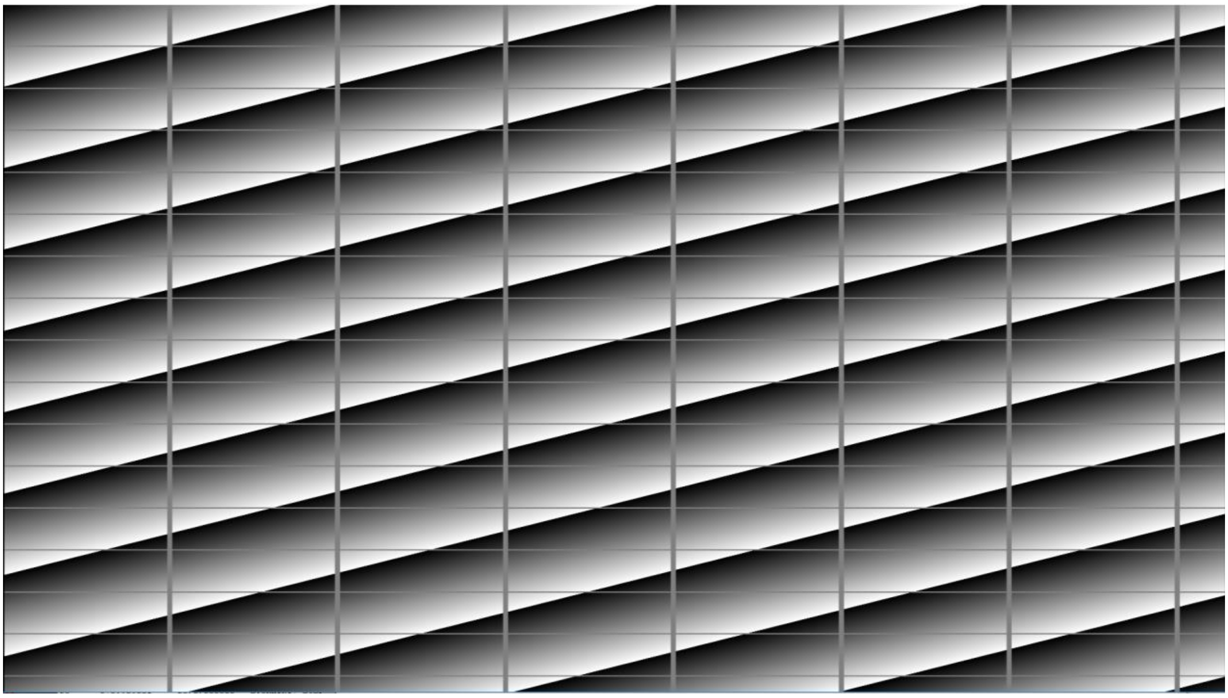


Figure 2-1: Test picture UHD

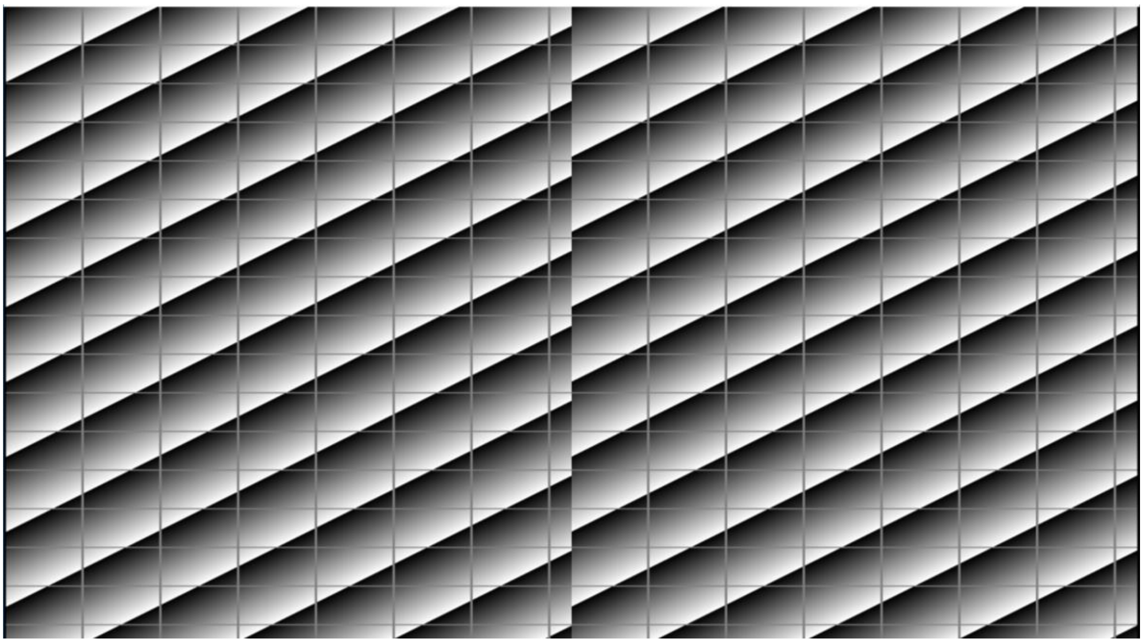


Figure 2-2: Test picture FHD

The TPG produces 32-bit pixels with the following content:
Bits 7:0 = Bits 15:8 = Bits 23:16 = counter, Bits 31:24 = 0x0

3 Header Channel Content

Each DMA channel has its own header channel (also known as metadata) with the following format (Table 3-1).

Offset	Length [bytes]	Value	Comment
0	4	0xa2b70b3b	32-bit magic value. MSB must always be ‘1’ because of Driver concept.
4	4	0x0	Version
8	8	Frame Counter	Increments with every end of frame
16	8	Clock Counter	Timestamp with 145.833 MHz clock since boot time
24	4	0x12345678	Static placeholder for the frame CRC
28	4	0x0	Padding zeros

Table 3-1

Mapping of Header and Data Channels to FPGA logical Channels

In case for applications that use the header channel, the FPGA designer must take care to realize the following mapping (Table 3-2).

FPGA logical Channel (TDEST)	Description	Physical Interface
0	Image for DMA channel 0	0
1	Header for DMA channel 0	6
2	Image for DMA channel 1	1
3	Header for DMA channel 1	7
4	Image for DMA channel 2	2
5	Header for DMA channel 2	8
6	Image for DMA channel 3	3
7	Header for DMA channel 3	9
8	Image for DMA channel 4	4
9	Header for DMA channel 4	10
10	Image for DMA channel 5	5
11	Header for DMA channel 5	11
12	Image for DMA Channel 6	12
13	Header for DMA channel 6	
14	Image for DMA Channel 7 (no meta)	13

Table 3-2

In case for applications that do not use the header channel, the FPGA designer must take care to realize the following mapping (Table 3-3).

Logical Channel (TDEST)	Description	Physical Interface
0	Image for DMA channel 0	0
1	Image for DMA channel 1	1
2	Image for DMA channel 2	2
3	Image for DMA channel 3	3
4	Image for DMA channel 4	4
5	Image for DMA channel 5	5
…	…	…

Table 3-3

4 PCI Express IDs

In order to identify the functionality of the bitstream and to map the correct device driver the following PCIe IDs are defined.

VENDOR_ID_C	Customer Vendor ID*
DEVICE_ID_C	0x0CDA
REVISION_ID_C	0x00
CLASS_CODE_C	0x0B4000
SUBSYS_VENDOR_ID_C	0x0
SUBSYS_DEVICE_ID_C	0x0

*For product development, customers should use their own Vendor ID.

5 Checks at Boot Time

Check for a valid Version Register of the IP Core

It is important to check, if BAR0, offset 0x0 does not report 0xFFFF_FFFF. In this case the driver must not be loaded due to a severe PCIe problem.

Check for Link Speed and Link Width

At boot time, it is important to check the negotiated link width and link speed of the endpoint.

The user should provide a minimum link speed / link width combination, that is checked against the negotiated value. It is possible that for soldering problems the endpoint runs at a slower speed than intended, that will not be enough for a correct operation. In this case the driver must detect this error state and disallow the DMA operations.

The negotiated link status is provided in the PCI Status Register located on BAR0 with offset 0x20.

The IP core's version in the FPGA can be read from BAR0, offset 0x0. This information might be useful for reporting.

In order to detect the various important system parameters, Table 5-1 shows the mapping of important registers.

BAR / Offset	Register Name	Description	Comment
0 0x0	Version Register	Bits 7:0: Core revision Bits 15:8: Year information, e.g. 0x14 for 2014 Bits 23:16: Month information, e.g. 0x01 for January Bits 31:24: Day information	Read-only
0 0x20	PCI Status	Bits 7:5: Maximum Payload Size (MPS) Bit 8: Extended Tag : '1' = supported Bits 14:12 : Maximum Request Size (MRS) Bits 19:16: Link speed : 0001 = Gen1, 0010 = Gen2, 0011 = Gen3, 0100 = Gen4 Bits 25:20: Link width 00001 = X1, 00010 = X2, 00100 = X4, 01000 = X8 Bit 31: MSI / MSI-X enabled	Read-only. All values reflect the negotiated values. Encoding for MPS and MRS: 000 = 128 Byte (typical for MPS) 001 = 256 Byte 010 = 512 Byte (typical for MRS) 011 = 1024 Byte 100 = 2048 Byte 101 = 4096 Byte
0 0x808	DMA Status	Bit 0-7: Empty of data FIFOs of AXISS 0-7 Bit 8-15: Empty of length FIFOs of AXISS 0-7 Bits 24-26: log2(tdest_width_g) Bit 27: '1' to identify HCC IP core Bit 28: '1' when 64-bit mode '0' when 32-bit mode Bit 29: '1' when single 64-bit access, '0' when dual 32-Bit access for 64-bit addressing is needed Bit 30: '1' Bit 31: '1' SG activated, '0' linear mode	Read-only, for debug purposes The number of implemented logical channels is 2^(bits 26:24).
0 0x884	General Parameter Info (DMA Write)	Bits 4:0: Number of used AXIS interfaces for DMA Write Bit 5: '1' = SG mode, '0' = linear mode Bit 6: '1' = BRAMs used for SG FIFOs, '0' distributed RAMs used Bit 14:7: log2(SG_FIFO_DEPTH)	Read-only
0 0x1084	General Parameter Info (DMA Read)	Bits 4:0: data channels in use for DMA Read (host to FPGA) Bit 5: '1' = SG Mode, '0' = linear mode Bit 6: '1' = use BRAM for SG FIFOs; '0' use distributed RAM Bit 14:7: log2(SG_FIFO_DEPTH) Bit 26:15: completion timeout (in 100us units) Bit 27 : '1' = Length FIFO enabled Bit 31:28 = log2(Length FIFO Depth)	

Table 5-1

Check for scatter/gather Mode

For correct operation the user must select the scatter/gather mode with the DMA driver (bit 5 of the General Parameter Info must be ‚1‘). If the core does not operate in scatter/gather mode, the driver must inform the user and must refuse operation.

Detecting the Type of the implemented Core

Bit 27 of the DMA status register identifies the Flex core (‚0‘) or the high channel count core (‚1‘).

Detecting the Number of implemented logical Channels

In case of the Flex core, bits 4:0 at offset 0x884 tell how many physical interfaces are implemented.

In case of the HCC core, bits 24-26 of the DMA status register are the log2 of the number of the implemented channels.
Example: If bits 26:24 are 0b100 the number of implemented channels is 2^4 = 16.

In case the user requests an operation with an unimplemented channel, the driver must refuse the operation.

Detecting the Address FIFO Depth

The address FIFO depth can be detected with bits 14:7 of the General Parameter Info register. The number is valid for the Flex core and the HCC core. This information is useful, since it informs the driver over the maximum number of address entries that can be stored within the address FIFO.

Enabling extended Tags

The IP core’s DMA Read performance depends, if it possible to use more than 32 tags. Each tag represents 1 outstanding read request. To reach the highest throughput it is necessary to enable the extended tag mode, where up to 256 tags are supported.

Important

The driver must find out, if the hardware platform on which it is running, supports extended tags. There are platforms that do not support extended tags. In this case turning on the extended tag feature will lead to DMA errors.

6 Various Programming Sequences (init, interrupt, rmmod, reset)

Programming Sequence after Boot or after `modprobe` (Table 6-1)

Step #	Description	Command
1	Basic IP core checks of chapter 5	
2	Ensure that DMA engines for read and write are disabled (in case another application has been using them before)	WR32: BAR0, offset 0x800 = 0x0 WR32: BAR0, offset 0x1000 = 0x0
3	Reset data and address FIFO for DMA Write engine	WR32: BAR0, offset 0x828 = 0xFFFF_FFFF RD32: BAR0, offset 0x0 (important!) WR32: BAR0, offset 0x828 = 0x0000_0000
4	Reset data and address FIFO for DMA Read engine	WR32: BAR0, offset 0x1028 = 0xFFFF_FFFF RD32: BAR0, offset 0x0 (important!) WR32: BAR0, offset 0x1028 = 0x0000_0000
5	Selection of 1D / 2D mode for DMA Read	WR32: BAR0, offset 0x1024 = 1D mode
6	Load of the Reload Budget Registers of all DMA Read channels with MRS. The number of existing channels can be read via BAR0, offset 0x1084 (Bits 4:0).	WR32: BAR0, offset 0x1200 + 4*ch = MRS
7	Set the channel specific page size registers for DMA Read	WR32: BAR0, offset 0x1080 = 0x1
8	Set the page size for each DMA Read channel Attention: Page size - 1 will be programmed	WR32: BAR0, offset 0x1280 + 4*ch
9	Set image format registers for DMA Read	WR32: BAR0, offset 0x1380 + 4*ch length of the overlay (bytes) for each channel
10	Set IncrementLineOffset registers of all logical channels to 0x200 (number of logical channels can be determined with BAR0, offset 0x808 (Bits 26:24). See Table 5-1 for details, note that this is the log2 of implemented logical channels	WR32: BAR0, offset 0xB00 + 4*if = 0x200 for all implemented interfaces
11	Set the individual page size registers for DMA Write	WR32: BAR0, offset 0x880 = 0x1
12	Set page size for each DMA Write channel Attention: Page size - 1 will be programmed	WR32: BAR0, offset 0xC00 + 4*ch for meta data channels 0xFFF will be written (4k page size)
13	Activate DMA Write engine	WR32: BAR0, offset 0x800 = 0x1
14	Activate DMA Read engine	WR32: BAR0, offset 0x1000 = 0x1

Table 6-1

Programming Sequence to deactivate the IP Core (rmmod or System Shutdown)

In case a `rmmod` is done by the user the sequence of Table 6-2 should be obeyed before the driver is unloaded.

Step #	Description	Command
1	Disable all interrupts	WR32: BAR0, offset 0x4 = 0x0
2	Disable DMA engine (Read and Write)	WR32: BAR0, offset 0x800 and 0x1000 = 0x0
3	Clear interrupt status (Register is clear on read)	RD32: BAR0 offset 0x8
4	Issue DMA Write FIFO Reset sequence (Table 6-7)	
5	FIFO reset DMA Read	WR32: BAR0, offset 0x1028 = 0xFFFF_FFFF RD32: BAR0, offset 0x0 (important !) WR32: BAR0, offset 0x1028 = 0x0000_0000
6	Disable MSI / MSI-X Interrupts within Linux	
7		

Table 6-2

Programming Sequence in case of a powerdown

In case of a powerdown the sequence of Table 6-2 should be obeyed before switching to the powerdown mode:

Step #	Description	Command
1	Disable all interrupts	WR32: BAR0, offset 0x4 = 0x0
2	Disable DMA engine (Read and Write)	WR32: BAR0, offset 0x800 and 0x1000 = 0x0
3	Clear interrupt status (Register is clear on read)	RD32: BAR0 offset 0x8
4	Issue DMA Write FIFO Reset Sequence (Table 6-7)	
5	FIFO reset DMA Read	WR32: BAR0, offset 0x1028 = 0xFFFF_FFFF RD32: BAR0, offset 0x0 (important !) WR32: BAR0, offset 0x1028 = 0x0000_0000
6	Enter powerdown	

Table 6-3

Programming Sequence in case of surprisal removal

In case of a surprisal removal, the driver will be unloaded without issueing any commands to the card because the card is not available anymore.

Programming Sequence to activate a DMA Channel (Table 6-4)

Step #	Description	Command
1	Enable physical interface (the number of implemented physical interfaces (S-AXIS) can be obtained from 0x884 (see chapter 5) See also chapter 3 which channel is mapped to which physical interface.	DMA Write: Read modify write Enable bit on BAR0, offset 0x804 Important for write: Activate all interfaces in streaming mode! DMA Read: Read modify write enable bit on BAR0, offset 0x1004
2	Load physical base address to address FIFO for the logical channel	DMA Write: WR64: BAR0, offset 0x900 + 8* channel DMA Read: WR64: BAR0, offset 0x1100 + 8* channel

Table 6-4

Important

In case that there are several independent threads that modify the DMA config registers (0x804 for write or 0x1004 for read) the driver must ensure, that the read modify write operations are atomic for each thread.

Once the DMA Write engine is correctly setup and a channel is enabled, it is enough to only load a physical address to the address FIFO.

Programming Sequence to activate End-of-Frame Interrupts

The end of frame interrupt is the key interrupt to inform the host, that a specific data buffer has been filled and is ready to be processed.

There are 3 operating modes that must be handled:

- Data channel operates without header or footer
- Data channel operates with additional data channel that is written *after* the frame (footer)
- Data channel operates with additional side information that is written *before* the frame data (header)

In this definition header/footer is derived from the write sequence in respect to the time. A footer is written *after* the data and a header is written before the data.

The demo application uses a footer (even if the footer is written before the data regarding memory addresses).

Table 6-5 shows which EOF triggers an interrupt in a particular use case.

Case	Metadata type	EOF Format	Comment
1	No metadata	EOF from data channel	
2	Written AFTER frame	EOF from metadata channel	
3	Written before frame	EOF from data channel	

Table 6-5

Activating the EOF Interrupt (Table 6-6)

Step #	Description	Command
1	Activation of the Global Interrupt Enable register for EOF not necessary.	BAR0, offset 0x82C: Bit 2 stays ‘0’
2	Activate the EOFs of the respective logical channel (FPGA to host)	WR32: BAR0, offset 0x83C
3	Activate the EOFs of the respective interface (host to FPGA)	WR32: BAR0, offset 0x103C
4	Enable EOF interrupt for DMA Write (FPGA to host) and enable EOF interrupt for DMA Read (host to FPGA)	WR32: BAR0, offset 0x4 = 0xFFFF0000

Table 6-6

Programming Sequence to reset the Address FIFO for DMA Write (Table 6-7)

Special care must be taken, when resetting the address FIFO, in order to avoid unintended critical system damages due to memory write packets to wrong addresses.

Step #	Description	Command
1	Disable AXI Stream interfaces (data and metadata) to force tready to go low	WR32: BAR0, offset 0x804: specific interface
2	Set Reset Bit for this TPG to ‘1’ to inform user logic in FPGA about address FIFO reset sequence	WR32: BAR0, offset 0x8000: specific bit = ‘1’
3	Put associated data FIFO and address FIFO of data and metadata into reset	WR32: BAR0, offset 0x828
4	Dummy read from the FPGA to insure, that the interfaces has finished it’s current packet	RD32: BAR0, offset 0x0
5	Clear both reset bits (data and address) for data and metadata	WR32: BAR0, offset 0x828
6	Set Reset Bit for the data source to ‘0’	WR32: BAR0, offset 0x8000: specific bit = ‘0’

Table 6-7

* Resetting the address FIFO on its own will not drive the tready of the corresponding axi_stream to low. If step 1 is not done and the user logic does not care about the GPO bit, the frame sync gets lost. Therefore, it is necessary to do step 1 in order to drive tready to low. Once the channel is opened again, make sure, that the channel is enabled later again via 0x804.

Definition and mapping of the Reset Flag Register

BAR / Offset	Register Name	Description	Comment
0 0x8000	Reset Flag register	Bit 0: DMA data and metadata are in reset sequence Data source 0 Bit 1: DMA data and metadata are in reset sequence Data source 1 … Bit 7: DMA data and metadata are in reset sequence Data source 7 … Bit 8:31: reserved for further data sources	Reset value is 0x0 When the reset sequence is entered this toggles from 0 to 1. When the reset sequence has finished, the bit toggles from 1 to 0. In the demo design bit 0 is connected to TPG0 reset, bit 1 to TPG1 reset, etc.

Table 6-8: Reset Flag Register

For the FPGA Designer:

When the condition is observed, that the Reset Flag Bit shows a high level, while tready is low, the FPGA Designer is informed, that when tready goes back to high, the first dataphase must contain a frame start pixel. If this is not obeyed, frames get chopped.

7 Demo Design specific Programming Sequence to activate Loopback or the TPGs

The demo design has a demo design specific register located on BAR1, offset 0x8000. With this register a global TPG reset and a loopback can be initiated (Table 7-1 and Table 6-8).

BAR / Offset	Register Name	Description	Comment
1 0x8000	Demo_design register	Bit 27: ‘1’: Select test picture with changing colour ‘0’: Sequence with non-inverted frame, inverted frame, non-inverted frame Bit 28: Video format: ‘0’: 24-Bit RGB; ‘1’: 30-Bit RGB Bits 30:29: “00” Normal TPG Mode, “10” Loopback Bits 31: ‘1’ = Reset all TPGs	Reset value is 0x0 Bit 31 resets all TPGs including their parameters

Table 7-1: Demo design project specific bits

Starting the TPGs

The following setup sequence (Table 7-2) must be done before the DMA channel is activated (e.g. programming register BAR0, offset 0x804)

Step #	Description	Command
1	Setup the TPGs as „master” and program grid	WR32: BAR1, offset 0x0000 + k * 0x1000 : 0x8 k=0 equals to TPG1, k=1 => TPG2, a.s.o. WR32: BAR1,0x0004+k*0x1000 : 0x80 WR32: BAR1,0x0008+k*0x1000 : 0x80 WR32: BAR1,0x000C+k*0x1000 : 0x04 WR32: BAR1,0x0010+k*0x1000 : 0x80
2	Activate the TPG, continuous, without trigger output	WR32: BAR1, offset 0x0000 + 0x1000*ch = 0x9

Table 7-2

Stopping the TPGs (Table 7-3)

Step #	Description	Command
1	Stopping the TPG	WR32: BAR1, offset 0x0000 + 0x1000*ch = 0x0

Table 7-3

Note: Stopping the TPG will stop the TPG *after* the current frame is finished.
Make sure to have the DMA channel enabled during stopping (HCC User Guide chapter 7.3.1).

Enabling Loopback (Table 7-4)

Step #	Description	Command
1	Make sure, that DMA engines for writing and reading are not running	WR32: BAR0, offset 0x800 and 0x1000 = 0x0
2	Set the FPGA to loopback by resetting the TPGs	WR32: BAR1, offset 0x8000: 0xC000_0000 WR32: BAR1, offset 0x8000: 0x4000_0000
3	Follow Table 6-1 without step 1	

Table 7-4

Note: Loopback can only be activated for all channels. Due to a Arria 10 device limitation, only a maximum of 4 TPGs can be activated simultaneously for IP Core version before 24032300.

8 MSI Interrupts

The demo design can either signal MSI *or* MSI-X interrupts.

This chapter describes the MSI case. If MSI-X is used, please refer to chapter 9.

For optimized interrupt behaviour MSI messages according to Table 8-1 are sent (8 negotiated MSI messages assumed).

MSI Message #	Interrupt Type	Comment
0	EOF interrupt (metadata or data) from TPG0	Attention: Messages 1- N-1 will only show up, if the requested number of MSI messages was negotiated with the host. If the host allows only 1 message, then all EOFs are mapped to message #0
1	EOF interrupt (metadata or data) from TPG1	
2	EOF interrupt (metadata or data) from TPG2	
…		
5	EOF interrupt (metadata or data) from TPG 5	
3	Test interrupt (for debugging)	This is true, when 8 MSI vectors are negotiated.
6-15	User interrupts	

Table 8-1

In case there are no other interrupt sources, the MSI message is unique and there is no need to read into the FPGA to distinguish between different interrupt sources.

Special Case

If the target hardware platform negotiates *only* 1 MSI message per add-in card, then all EOF interrupts will be sent with message 0. The device driver can detect this situation after system boot.

In order to detect the channel that caused the interrupt in this case, follow this procedure (Table 8-2).

Step #	Description	Command
1	MSI interrupt arrives	
2	Detect interrupt source by reading the Interrupt Status Register Note: This register is auto clear on read, so it is not necessary to clear this register with a write command.	RD32: BAR0, offset 0x8

Table 8-2

Table 8-3 illustrates how to determine the source of an interrupt in case of 1 MSI with the Interrupt Status Register.

Status Register Bit	Interrupt Type	Comment
0	DMA Write interrupt	DMA Write module triggered an interrupt
1	DMA-Read interrupt	DMA Read module triggered an interrupt
2	All EOF seen interrupt	Is asserted, when all activated physical interfaces showed an EOF condition. Main cause is for debugging.
3	Test interrupt	Can be activated with interrupt_control2 register (see HCC user guide)
4-13	User interrupts	User logic in the FPGA can trigger application specific events to inform the host on project specific things.
14:15	Reserved for future features	
16-23	EOF interrupts (FPGA to host) 16: data EOF or metadata EOF of TPG 0 17: data EOF or metadata EOF of TPG 1 18: data EOF or metadata EOF of TPG 2 .. 21: data EOF or with metadata EOF of TPG 5 22: reserved for a future data source 23: reserved for a future data source	Which interrupt triggers the interrupt is defined by 0x83C. It can be metadata or data or both.
24-31	EOF interrupts (host to FPGA) 24: MAXIS 0 25: MAXIS 1 26: MAXIS 2 27: MAXIS 3 28: MAXIS 4 29: MAXIS 5 30: reserved for MAXIS6 31: reserved for MAXIS7	Which interrupt triggers the interrupt is defined by 0x103C For this direction, there are only 16 channels available. Channels are equal to interfaces.

Table 8-3

In case an EOF interrupt occurs, the Interrupt Status Register will show the TPG number of the interrupt.

9 MSI-X Interrupts

PCI Express supports MSI and MSI-X interrupts. MSI interrupts have the disadvantage that the negotiated number of vectors is platform dependent. On one platform there might be 32 messages available for use, on others only 1 MSI message might be available for use. This forces the driver developer to consider both cases, which is awkward and in the case of only 1 message suboptimal regarding performance.

The advantage of MSI-X is, that all messages are available on all platforms. The PCI spec allows the support of up to 2048 separate MSI-X Messages. However due to the internal IP core architecture only 32 MSI-X vectors can be supported. Currently this is no limitation, since the core internal interrupt sources are less than 32.

It is assumed, that for MSI-X every attempt to allocate 32 messages should succeed, if MSI-X is enabled on the FPGA. Therefore, the driver should inform the user in case less than 32 MSI-X vectors were negotiated.

Table 9-1 lists the possible MSI-X interrupts sources.

MSI-X IRQ #	Interrupt Type	Comment
0	DMA Write interrupt	DMA Write module triggered an interrupt
1	DMA-Read interrupt	DMA Read module triggered an interrupt
2	All EOF seen interrupt	Is asserted, when all activated physical interfaces showed an EOF condition. Main cause is for debugging.
3	Test interrupt	Can be activated with interrupt_control2 register (see UG)
4-13	User interrupts	User logic in the FPGA can trigger application specific events to inform the host on project specific things.
14:15	Reserved for future features	
16-23	EOF interrupts (FPGA to host) 16 : data EOF or metadata EOF of TPG 0 17 : data EOF or metadata EOF of TPG 1 18 : data EOF or metadata EOF of TPG 2 .. 21 : data EOF or metadata EOF of TPG 5 22 : data EOF or metadata EOF of TPG 6 23 : data EOF or metadata EOF of TPG 7	Which interrupt triggers the interrupt is defined by 0x83C. It can be meta-data or data or both.
24-31	EOF interrupts (host to FPGA) 24 : MAXIS 0 25 : MAXIS 1 26 : MAXIS 2 27 : MAXIS 3 28 : reserved for MAXIS 4 29 : reserved for MAXIS 5 30 : reserved for MAXIS 6 31 : reserved for MAXIS 7	Which interrupt triggers the interrupt is defined by 0x103C For this direction, there are only 16 channels available. Channels are equal to interfaces.

Table 9-1

10 Link analysis features

The design contains the link analysis features described in chapter 8 of the HCC IP Core user guide.

11 Important Notes

FIFO Reset

The FIFO reset issue is now fixed for the direction from FPGA to CPU (DMA Write). The address FIFO can now be reset while the DMA Write engine is running.

The limitation however persists for DMA Read. Here it is not allowed to reset the address FIFO while the DMA Read Engine is running. If the user wants to reset the DMA Read address FIFO, it is important to make sure, that the DMA Read Engine is either disabled globally (Register “DMA Read Control”) or that the specific channel is disabled (Register “DMA Read Config”), before a FIFO Reset for DMA Read is issued.

Releasing DMA Buffers for DMA Read:

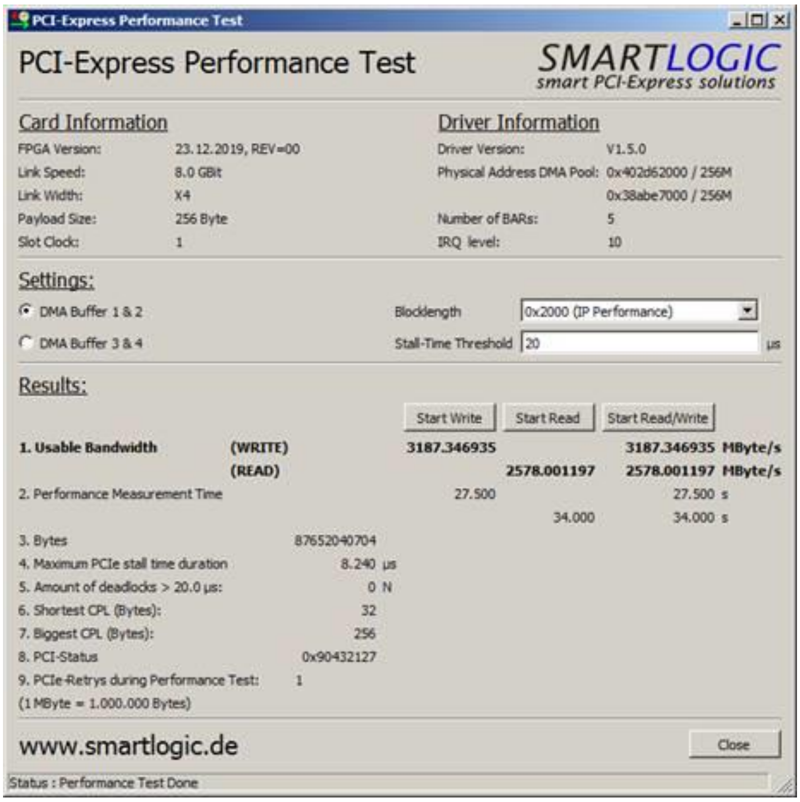
It is important that the User does not release DMA Buffers before the data is transmitted out of the buffer.

It is recommended that the User configures the EOF Interrupts of the DMA Read engine, so that he is informed, when the DMA Buffer data has been transmitted and the buffer itself is ready to be released.

12 Throughput

The asymmetric performance issue for Intel Arria 10 (Performance Demo Design) has been solved.

Intel Arria 10, PCIe Gen3 x4 results (old)


PCI-Express Performance Test
PCI-Express Performance Test
SMARTLOGIC
smart PCI-Express solutions
Start Write
Start Read
Start Read/Write
www.smartlogic.de
Close

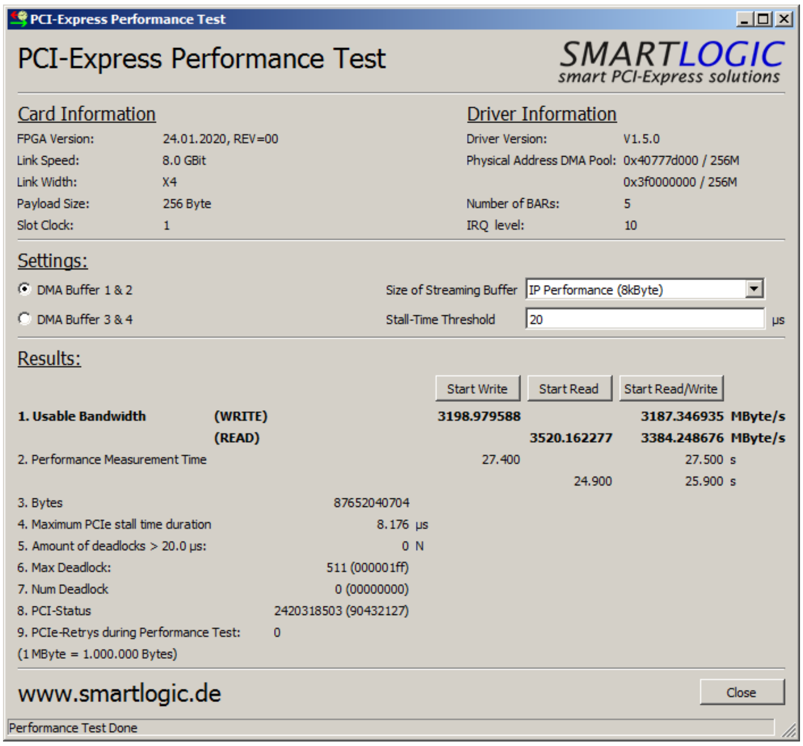
```
Card Information                                Driver Information
FPGA Version:     23.12.2019, REV=00            Driver Version:              V1.5.0
Link Speed:       8.0 GBit                      Physical Address DMA Pool:   0x402d62000 / 256M
Link Width:       X4                                                         0x38abe7000 / 256M
Payload Size:     256 Byte                      Number of BARs:              5
Slot Clock:       1                             IRQ  level:                  10

Settings:
(•) DMA Buffer 1 & 2                  Blocklength            0x2000 (IP Performance)
( ) DMA Buffer 3 & 4                  Stall-Time Threshold   20                        µs

Results:
1. Usable Bandwidth        (WRITE)        3187.346935                  3187.346935  MByte/s
                           (READ)                       2578.001197    2578.001197  MByte/s
2. Performance Measurement Time                27.500                       27.500  s
                                                             34.000         34.000  s
3. Bytes                               87652040704
4. Maximum PCIe stall time duration          8.240  µs
5. Amount of deadlocks > 20.0 µs:                0  N
6. Shortest CPL (Bytes):                        32
7. Biggest CPL (Bytes):                        256
8. PCI-Status                           0x90432127
9. PCIe-Retrys during Performance Test:  1
(1 MByte = 1.000.000 Bytes)

Status : Performance Test Done
```

New results, now show better read performance


PCI-Express Performance Test
PCI-Express Performance Test
SMARTLOGIC
smart PCI-Express solutions
Start Write
Start Read
Start Read/Write
www.smartlogic.de
Close

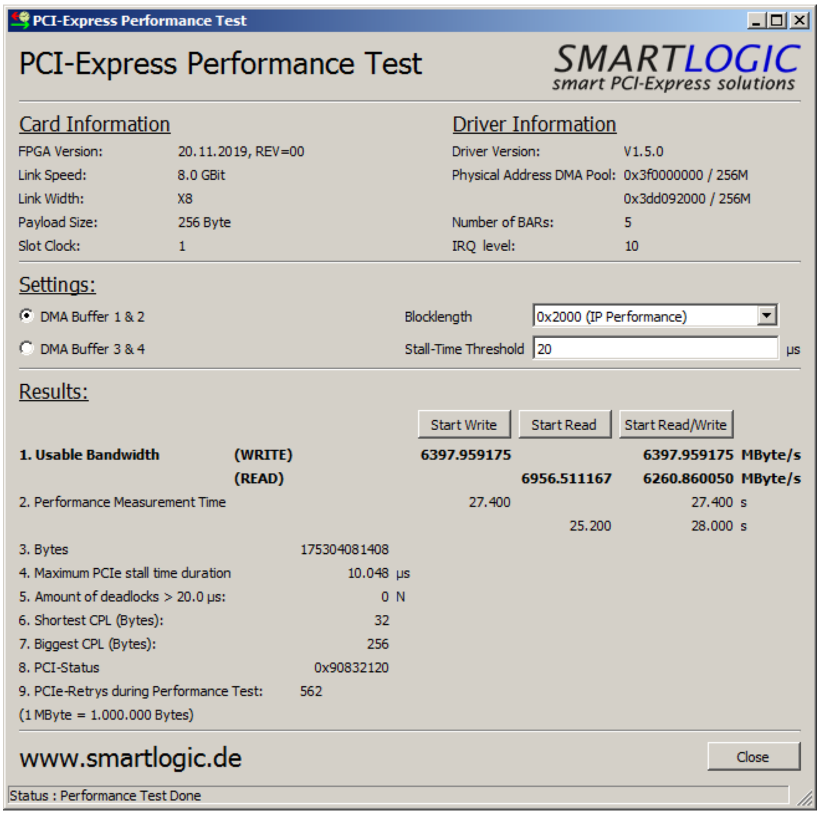
```
Card Information                                Driver Information
FPGA Version:     24.01.2020, REV=00            Driver Version:              V1.5.0
Link Speed:       8.0 GBit                      Physical Address DMA Pool:   0x40777d000 / 256M
Link Width:       X4                                                         0x3f0000000 / 256M
Payload Size:     256 Byte                      Number of BARs:              5
Slot Clock:       1                             IRQ  level:                  10

Settings:
(•) DMA Buffer 1 & 2                  Size of Streaming Buffer   IP Performance (8kByte)
( ) DMA Buffer 3 & 4                  Stall-Time Threshold       20                      µs

Results:
1. Usable Bandwidth        (WRITE)        3198.979588                  3187.346935  MByte/s
                           (READ)                       3520.162277    3384.248676  MByte/s
2. Performance Measurement Time                27.400                       27.500  s
                                                             24.900         25.900  s
3. Bytes                               87652040704
4. Maximum PCIe stall time duration          8.176  µs
5. Amount of deadlocks > 20.0 µs:                0  N
6. Max Deadlock:                      511 (000001ff)
7. Num Deadlock                         0 (00000000)
8. PCI-Status                  2420318503 (90432127)
9. PCIe-Retrys during Performance Test:  0
(1 MByte = 1.000.000 Bytes)

Performance Test Done
```

Stratix 10, PCIe Gen3 x8 results



PCI-Express Performance Test

Card Information

FPGA Version: 20.11.2019, REV=00
Link Speed: 8.0 GBit
Link Width: X8
Payload Size: 256 Byte
Slot Clock: 1

Driver Information

Driver Version: V1.5.0
Physical Address DMA Pool: 0x3f0000000 / 256M
0x3dd092000 / 256M
Number of BARs: 5
IRQ level: 10

Settings:

● DMA Buffer 1 & 2
○ DMA Buffer 3 & 4

Blocklength: 0x2000 (IP Performance)
Stall-Time Threshold: 20 µs

Results:

```
1. Usable Bandwidth            (WRITE)    6397.959175                  6397.959175  MByte/s
                               (READ)                   6956.511167    6260.860050  MByte/s
2. Performance Measurement Time                27.400                        27.400  s
                                                            25.200           28.000  s
3. Bytes                           175304081408
4. Maximum PCIe stall time duration      10.048  µs
5. Amount of deadlocks > 20.0 µs:             0  N
6. Shortest CPL (Bytes):                     32
7. Biggest CPL (Bytes):                     256
8. PCI-Status                        0x90832120
9. PCIe-Retrys during Performance Test:  562
(1 MByte = 1.000.000 Bytes)
```

www.smartlogic.de

Status : Performance Test Done

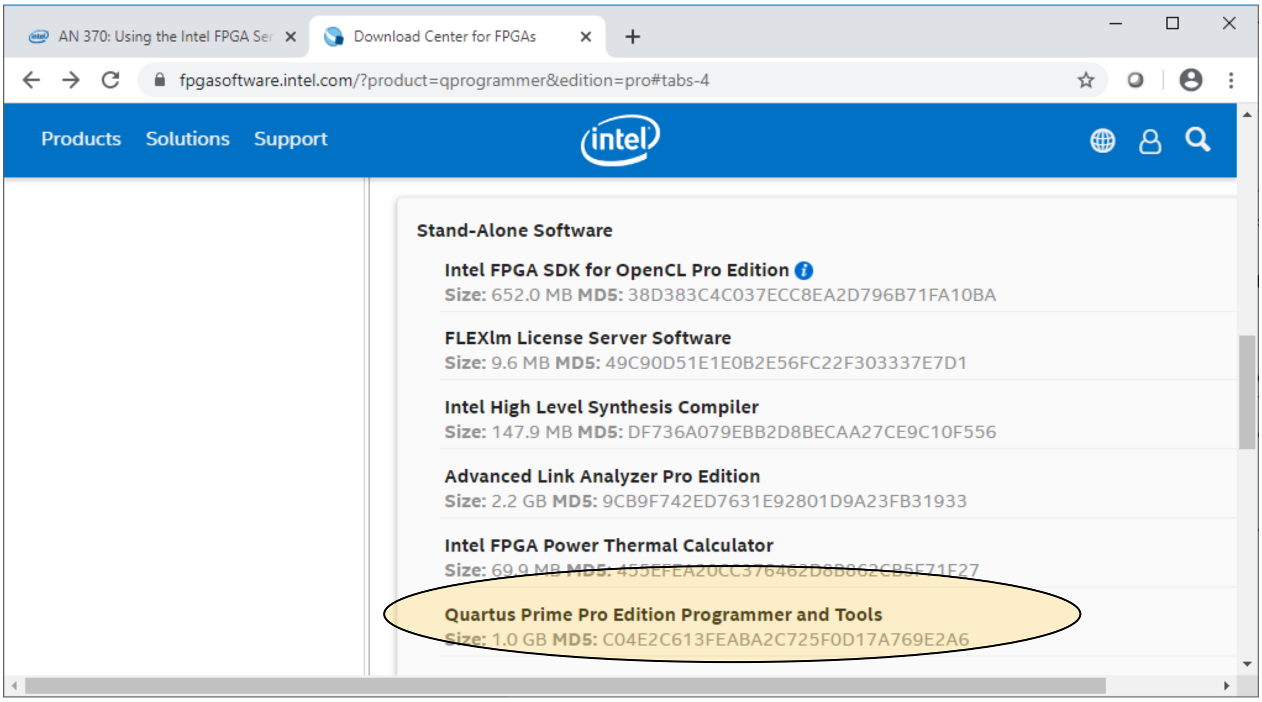
13 Bitstream Programming

This section describes in detail how to update the bitstream on the Dream Chip eval board.

Install the Quartus Prime Pro programmer from Intel:

https://fpgasoftware.intel.com/?product=qprogrammer&edition=pro#tabs-4

There is the standalone programmer version that runs without a license:


AN 370: Using the Intel FPGA Ser
Download Center for FPGAs
fpgasoftware.intel.com/?product=qprogrammer&edition=pro#tabs-4
Products
Solutions
Support
intel

Stand-Alone Software

Intel FPGA SDK for OpenCL Pro Edition
Size: 652.0 MB **MD5:** 38D383C4C037ECC8EA2D796B71FA10BA

FLEXlm License Server Software
Size: 9.6 MB **MD5:** 49C90D51E1E0B2E56FC22F303337E7D1

Intel High Level Synthesis Compiler
Size: 147.9 MB **MD5:** DF736A079EBB2D8BECAA27CE9C10F556

Advanced Link Analyzer Pro Edition
Size: 2.2 GB **MD5:** 9CB9F742ED7631E92801D9A23FB31933

Intel FPGA Power Thermal Calculator
Size: 69.9 MB **MD5:** 455EFEA20CC376462D8B862CB5F71F27

Quartus Prime Pro Edition Programmer and Tools
Size: 1.0 GB **MD5:** C04E2C613FEABA2C725F0D17A769E2A6

Figure 13-1

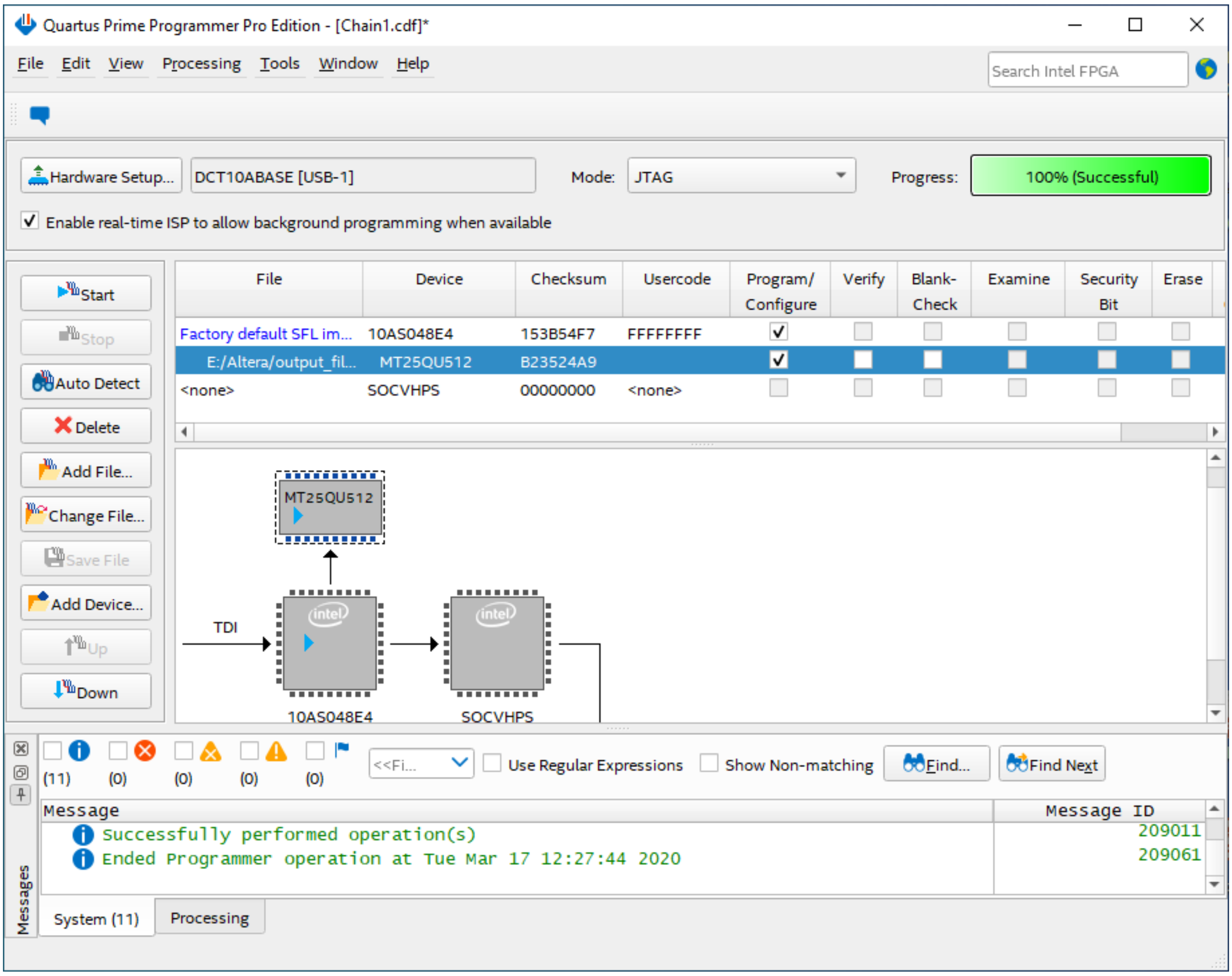
There is no need for special programming hardware. Programming can be done with a USB connection. Simply connect the Mini USB labelled “USB Blaster II” with the PC where the programmer is installed.

It is recommended to use the rear USB ports of the PC since the intern cabling is shorter and considered more robust.

Important

Never update the flash, when the eval board is plugged into a graphics card slot slot.
Either disconnect the eval board from the PCIe slot or plug in the demo board temporarily to a non-graphics card slot. Some operating system versions stop executing when the PCI Express connection is lost.

Open the programmer tool and load the CDF file supplied from Smartlogic.



Hardware Setup: DCT10ABASE [USB-1] Mode: JTAG Progress: 100% (Successful)

☑ Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase
Factory default SFL im...	10AS048E4	153B54F7	FFFFFFFF	☑	☐	☐	☐	☐	☐
E:/Altera/output_fil...	MT25QU512	B23524A9		☑	☐	☐	☐	☐	☐
<none>	SOCVHPS	00000000	<none>	☐	☐	☐	☐	☐	☐

```
Message                                                  Message ID
Successfully performed operation(s)                          209011
Ended Programmer operation at Tue Mar 17 12:27:44 2020       209061
```

Figure 13-2

Simply press “Start” to initiate the programming of the QSPI flash.

Important

The MAX10 of the Arria10 SOM was changed with a special POF file (pwrseq_19_10_02_00.pof) provided by Dream Chip. Only with this POF file the QSPI flash is visible. Both boards were updated (non-volatile) with this POF. In case the user wants to switch back to the factory default, the MAX10 device must be reprogrammed with the factory default POF file (pwrseq_19_00_02_00.pof).

14 Version Table

The 8 LSBs of the version register (BAR0, offset 0x0) specify the revision of the design. Table 14-1 lists the released bitstream revisions including features, limitations and purpose.

Rev #	Description	
0	First release, Error: interrupt mapping	
1	MSI, Smartlogic IDs, bugfixes	
2	MSI-X, Smartlogic IDs, RX buffer setting “low”	
3	MSI-X, IDs as specified in chapter 4. Error: Intel IP RX buffer wrong setting (High), therefore DMA Read error	
4	MSI, Smartlogic IDs, FIFO reset fix, RX buffer setting “low”, throughput optimization, TPG reset changed	
5	Like 4, but with IDs from chapter 4 and MSI-X	
6	Like 4, but optimized dma_pkg.vhd, for better buffering capability	
7	Like 5, but optimized dma_pkg.vhd, for better buffering capability, fixed loopback problem	
8	Like 7, but with selection bits for video format and frame sequence according to Table 7-1	
9	Like 8, but with new release (26102000) of Smartlogic’s HCC IP Core	
10	Changed EOF behaviour of EOF interrupt enable register (BAR0, Offset 0x83C, added TPGs 6 and 7, changed memory mapping, GPO Register is now not used anymore. Based on IP Core Release 13012100	
11	User Interrupts (0:7) now connected to EOF(0:7) user_interrupt(8) connected to 1 ms interrupt, user_interrupt(9) connected to 100 ms interrupt. Based on IP Core Release 13012100	
12	As Revision 11 but based on IP Core Release 03032100. Contains also a protection (Demodesign only), that TPG in Reset does also answer read requests. This prevents Bluescreens due to Completion timeouts	

Table 14-1

Table 14-1 covers the versions for the main Demo design, specified in this document.

The tested Performance Demo Design shows 0x04082000 in version register.

15 References

More user guides for the following topics are available from Smartlogic:

Topic	Smartlogic user guide
Datasheet of the HCC IP Core	pcie_hcc_dma_ip_core.pdf*
User Guide of the HCC IP Core	PCIe_HCC_DMA_IP_Core_UG.pdf**
User Guide of the picture generator	Picture_generator.pdf*
Working with the IP catalog flow for Intel FPGAs	AN_Intel_IP_Catalog_flow.pdf*
Design guide for timing closure	AN_designguides_for_timing_closure.pdf*

* PDF document is available on the Smartlogic Webpage

** PDF document is available from Smartlogic, but requires a NDA

The following Vendor-specific user guide provides additional information:

Technology	User guide name
Arria 10	UG-01145

This user guide can be downloaded from www.intel.com.