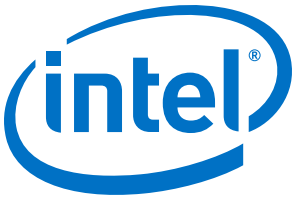


AN 829: PCI Express* Avalon®-MM DMA Reference Design

Updated for Intel® Quartus® Prime Design Suite: **18.0**

Subscribe
Send Feedback

AN-829 | 2018.06.11
Latest document on the web: **PDF** | **HTML**

Contents

1. AN 829: PCI Express* Avalon®-MM DMA Reference Design 3
1.1. Introduction.....3
1.1.1. DMA Reference Design Hardware and Software Requirements.....4
1.1.2. Avalon-MM DMA Bridge Module Descriptions.....5
1.2. Working with the Reference Design.....6
1.2.1. Project Hierarchy.....6
1.2.2. Parameter Settings for PCI Express Hard IP Variations.....6
1.2.3. PCIe Avalon-MM DMA Reference Design Platform Designer Systems.....8
1.2.4. DMA Procedure Steps.....11
1.2.5. Setting Up the Hardware.....11
1.2.6. Programming the Intel Cyclone 10 GX FPGA Oscillator.....12
1.2.7. Installing the DMA Test Driver and Running the Linux DMA Software.....13
1.3. Understanding PCI Express Throughput.....17
1.3.1. Throughput for Posted Writes.....18
1.3.2. Throughput for Reads.....20
1.3.3. Throughput Differences for On-Chip and External Memory.....21
1.4. Document Revision History for AN 829: PCI Express Avalon-MM DMA Reference Design.....22

1. AN 829: PCI Express* Avalon®-MM DMA Reference Design

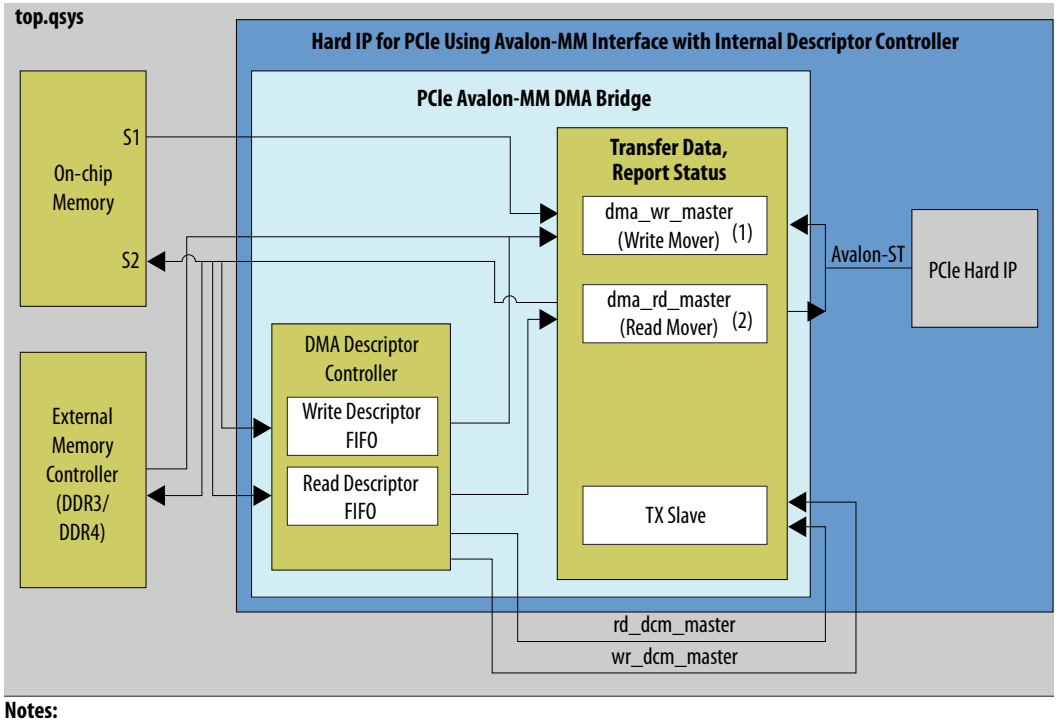
1.1. Introduction

The PCI Express* Avalon® Memory-Mapped (Avalon-MM) Direct Memory Access (DMA) Reference Design demonstrates the performance of the Intel® Arria® 10, Intel Cyclone® 10 GX, and Intel Stratix® 10 Hard IP for PCIe* using an Avalon-MM interface and an embedded, high-performance DMA controller.

The design includes a Linux software driver to set up the DMA transfers. The read DMA moves data from the system memory to the on-chip or external memory. The write DMA moves data from the on-chip or external memory to the system memory. The Linux software driver also measures the system performance. This reference design allows you to evaluate the performance of the PCIe protocol in using the Avalon-MM interface with an embedded, high-performance DMA.

Figure 1. PCIe Avalon-MM DMA Reference Design Block Diagram

This bock diagram shows both the on-chip memory and external memory options.



Notes:

(1) Write Mover transfers data from local domain to the host domain

(2) Read Mover transfers data from the host domain to local domain

Related Information

- Intel Stratix 10 Avalon-MM Interface for PCIe Solutions User Guide
- Intel Arria 10 or Intel Cyclone 10 Avalon-MM DMA Interface for PCIe Solutions User Guide
- PCI Express Base Specification Revision 3.0

1.1.1. DMA Reference Design Hardware and Software Requirements

Hardware Requirements

The reference design runs on the following development kits:

- Intel Arria 10 GX FPGA Development Kit
- Intel Cyclone 10 GX FPGA Development Kit
- Intel Stratix 10 FPGA Development Kit

The reference design requires two computers:

- A computer with a PCIe Gen3 x8 or x16 slot running Linux. This computer is computer number 1.
- A second computer with the Intel Quartus® Prime software version 18.0 installed. This computer downloads the FPGA SRAM Object File (`.sof`) to the FPGA on the development kit. This computer is computer number 2.

Software Requirements

- The reference design software installed on computer number 1. The reference designs are available in the Intel FPGA Design Store. The Intel Quartus Prime Pro Edition Platform Archive File (`.par`) includes the recommended synthesis, fitter, and timing analysis settings for the parameters specified in the reference designs.
- The Intel Quartus Prime software installed on computer number 2. You can download this software from Intel Quartus Prime Pro Edition Software Features/ Download web page.
- The Linux driver configured specifically for these reference designs.

Related Information

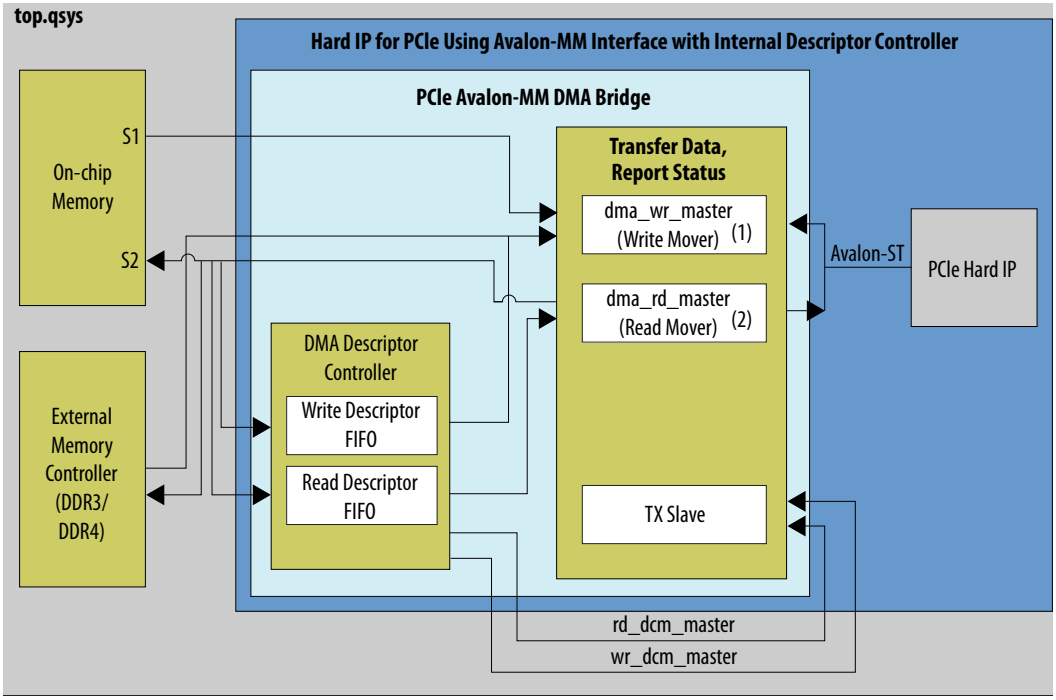
- Intel Arria 10 Reference Design
  To download the reference design and the design software from the Design Store.
- Intel Cyclone 10 GX Reference Design
  To download the reference design and the design software from the Design Store.
- Stratix 10 Reference Design
  To download the reference design and the design software from the Design Store.
- Intel Quartus Prime Pro Edition Download Center

1.1.2. Avalon-MM DMA Bridge Module Descriptions

The Avalon-MM interface with DMA includes the following modules:

Figure 2. PCIe Avalon-MM DMA Reference Design Block Diagram

This bock diagram shows both the on-chip memory and external memory options.



Notes:
(1) Write Mover transfers data from local domain to the host domain
(2) Read Mover transfers data from the host domain to local domain

Read Data Mover

The Read Data Mover sends memory read Transaction Layer Packet (TLPs) upstream. After the Read Data Mover receives the Completion, the Read Data Mover writes the received data to the on-chip or external memory.

Write Data Mover

The Write Data Mover reads data from the on-chip or external memory and sends the data upstream using memory write TLPs on the PCIe link.

DMA Descriptor Controller

The Descriptor Controller module manages the DMA read and write operations.

Host software programs internal registers in the Descriptor Controller with the location and size of the descriptor table residing in the host system memory through the Avalon-MM RX master port. Based on this information, the Descriptor Controller directs the Read Data Mover to copy the entire table to local FIFOs for execution. The Descriptor Controller sends completion status upstream via the Avalon TX slave (TXS) port.

You can also use your own external descriptor controller to manage the Read and Write Data Movers. However, you cannot change the interface between your own external controller and the Read and Write Data Movers embedded in the reference design.

TX Slave

The TX Slave module propagates Avalon-MM reads and writes upstream. External Avalon-MM masters, including the DMA control master, can access system memory using the TX Slave. The DMA Controller uses this path to update the DMA status upstream, using Message Signaled Interrupt (MSI) TLPs.

RX Master (Internal Port for BAR0 Control)

The RX Master module propagates single dword read and write TLPs from the Root Port to the Avalon-MM domain via a 32-bit Avalon-MM master port. Software instructs the RX Master to send control, status, and descriptor information to Avalon-MM slaves, including the DMA control slave. The RX Master port is an internal port that not visible in Platform Designer.

1.2. Working with the Reference Design

1.2.1. Project Hierarchy

The reference design uses the following directory structure:

- `top`—The top-level module.
- `top_hw`— Platform Designer top-level files. If you modify the design using Platform Designer, you must regenerate the system for the changes to take effect.

1.2.2. Parameter Settings for PCI Express Hard IP Variations

This reference design supports a 256-byte maximum payload size. The following tables list the values for all the parameters.

Table 1. System Settings

Parameter	Value
Number of lanes	Intel Cyclone 10 GX: x4 Intel Arria 10, Intel Stratix 10: x8
Lane rate	Intel Cyclone 10 GX: Gen2 (5.0 Gbps) Intel Arria 10 Intel Stratix 10: Gen3 (8.0 Gbps)
RX buffer credit allocation – performance for received request	Intel Arria 10, Intel Cyclone 10 GX: Low Intel Stratix 10: Not available

Table 2. Base Address Register (BAR) Settings

Parameter	Value
BAR0	64-bit prefetchable memory
BAR1	Disabled
BAR2	64-bit prefetchable memory BAR2 is disabled for Intel Stratix 10

continued...

Parameter	Value
BAR3	Disabled
BAR4	64-bit prefetchable memory BAR4 is disabled for Intel Arria 10 and Intel Cyclone 10 GX
BAR5	Disabled

Table 3. Device Identification Register Settings

Parameter	Value
Vendor ID	0x00001172
Device ID	0x0000E003
Revision ID	0x00000001
Class Code	0x00000000
Subsystem Vendor ID	0x00000000
Subsystem Device ID	0x00000000

Table 4. PCI Express/PCI* Capabilities

Parameter	Value
Maximum payload size	256 Bytes
Completion timeout range	None
Implement Completion Timeout Disable	Enabled

Table 5. Error Reporting Settings

Parameter	Value
Advanced Error Reporting (AER)	Enabled
ECRC checking	Disabled
ECRC generation	Disabled

Table 6. Link Settings

Parameter	Value
Link port number	1
Slot clock configuration	Enabled

Table 7. Message Signaled Interrupts (MSI) and MSI-XSettings

Parameter	Value
Number of MSI messages requested	4
Implement MSI-X	Disabled
Table size	0
Table offset	0x0000000000000000

continued...

Parameter	Value
Table BAR indicator	0
Pending bit array (PBA) offset	0x0000000000000000
PBA BAR Indicator	0

Table 8. Power Management

Parameter	Value
Endpoint L0s acceptable latency	Maximum of 64 ns
Endpoint L1 acceptable latency	Maximum of 1 us

Table 9. PCIe Address Space Setting

Parameter	Value
Address width of accessible PCIe memory space	40

1.2.3. PCIe Avalon-MM DMA Reference Design Platform Designer Systems

The following images show the Platform Designer systems for Intel Arria 10, Intel Cyclone 10 GX, and Intel Stratix 10 devices.

Figure 3. Intel Arria 10 GX DMA Reference Design Platform Designer System

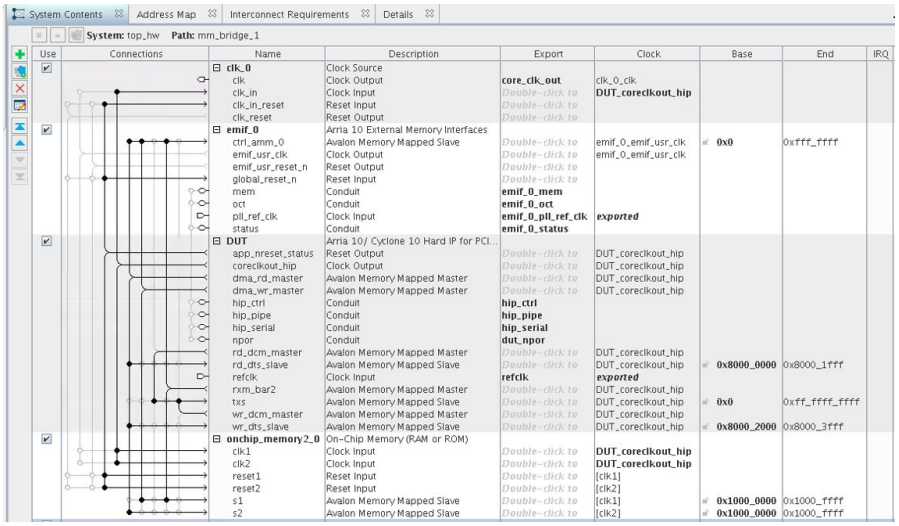


Figure 4. Intel Cyclone 10 GX GX DMA Reference Design Platform Designer System

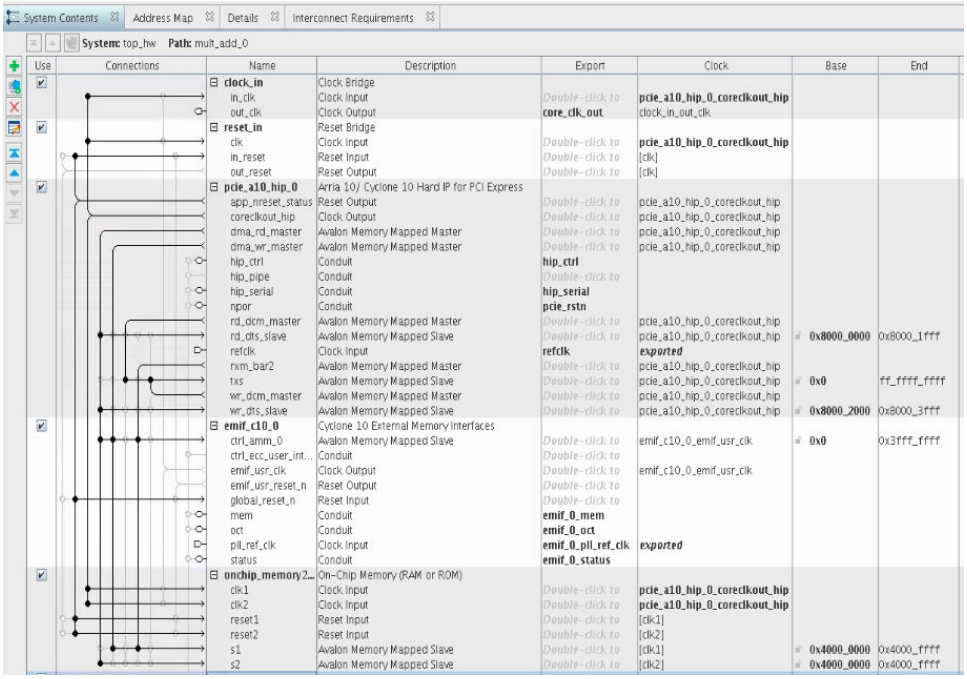


Figure 5. Intel Stratix 10 GX DMA Reference Design Platform Designer System

The Intel Stratix 10 design includes pipeline components and clock-crossing logic that are not present in the other devices.

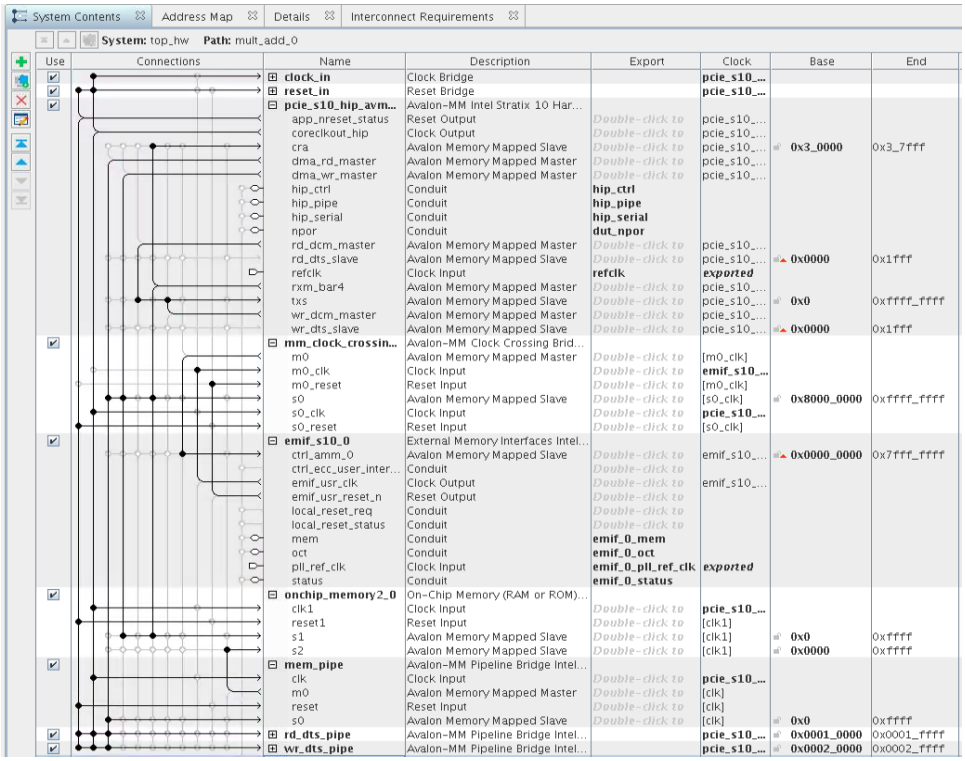


Table 10. Platform Designer Port Descriptions

Port	Function	Description

`rxm_bar2`
or
`rxm_bar4`
	Avalon-MM port	This is an Avalon-MM master port. The PCIe host accesses the memory through PCIe BAR2 for Intel Arria 10 and Intel Cyclone 10 GX devices. The host accesses the memory through PCIe BAR4 for Intel Stratix 10 devices. These BARs connect to both on-chip and external memory.
		In a typical application, system software controls this BAR to initialize random data in the external memory. Software also reads the data back to verify correct operation.

`txs`	TX Avalon-MM Slave	This is an Avalon-MM slave port. In a typical application, an Avalon-MM master controls this port to send memory reads and writes to the PCIe domain. When the DMA completes operation, the Descriptor Controller uses this port to write DMA status back to the descriptor table in the PCIe domain. The Descriptor Controller also uses this port to send MSI interrupts upstream.
`dma_rd_master`	Read Data Mover	This is an Avalon-MM master port. The Read Data Mover uses this Avalon-MM master to move data from the PCIe domain to either the on-chip or external memory. The Read Data Mover also uses this port to fetch descriptors from the PCIe domain and write them to the FIFO in the Descriptor Controller. The design includes separate descriptor tables for read and write descriptors. Consequently, the `dma_rd_master` port connects to `wr_dts_slave` for the write DMA descriptor FIFO and `rd_dts_slave` for the read DMA descriptor FIFO.
`dma_wr_master`	Write Data Mover	This is an Avalon-MM master port. The Write Data Mover uses this Avalon-MM master to read data from either the on-chip or external memory and then write data to the PCIe domain. The external memory controller is a single-port RAM. Consequently, the Write Data Mover and the Read Data Mover must share this port to assess external memory.
`wr_dts_slave` `rd_dts_slave`	FIFO in Descriptor Controller	These are Avalon-MM slave ports for the FIFOs in the Descriptor Controller. When the Read Data Mover fetches the descriptors from system memory, the Read Data Mover writes the descriptors to the FIFO using the `wr_dts_slave` and `rd_dts_slave` ports.
`wr_dcm_master` `rd_dcm_master`	Control module in Descriptor Controller	The Descriptor Controller control module includes one transmit and one receive port for the read and write DMAs. The receive port connects to RXM_BAR0. The transmit port connects to the `txs`. The receive path from the RXM_BAR0 connects internally. RXM_BAR0 is not shown in the Platform Designer connections panel. For the transmit path, both read and write DMA ports connect to the `txs` externally. These ports are visible in the Platform Designer connections panel.
Internal connection, not shown	Avalon-MM port	This Avalon-MM master port passes the memory access from the PCIe host to PCIe BAR0. The host uses this port to program the Descriptor Controller. Because this reference design includes the Descriptor Controller as an internal module, Platform Designer does not display this port on the top-level connections panel.
`onchip_memory2_0`	64 KB Dual Port RAM	This is a 64-KB dual-port on-chip memory. The address range is 0x0800_0000-0x0800_FFFF on the Avalon-MM bus. This address is the source address for write DMAs or destination address for read DMAs. To prevent data corruption, software divides the memory into separate regions for reads and writes. The regions do not overlap.
Intel DDR3 or DDR4 controller	DDR3 or DDR4 Controller	This is a single-port DDR3 or DDR4 controller.

1.2.4. DMA Procedure Steps

Software running on the host completes the following steps to initiate the DMA and verify the results:

1. Software allocates system memory for the descriptor table.
2. Software allocates system memory for the DMA data transfers.
3. Software writes the descriptors to the descriptor table in the system memory. The DMA supports up to 128 read and 128 write descriptors. The descriptor table records the following information:
   - Descriptor ID, ranging from 0-127
   - Source address
   - Destination address
   - Size
4. For the read DMA, the software initializes the system memory space with random data. The Read Data Mover moves this data from the system memory to either the on-chip or external memory. For the write DMA, the software initializes the on-chip or external memory with random data. The Write Data Mover moves the data from the on-chip or external memory to the system memory.
5. Software programs the registers in the Descriptor Controller's control module through BAR0. Programming specifies the base address of the descriptor table in system memory and the base address of the FIFO that stores the descriptors in the FPGA.
6. To initiate the DMA, software writes the ID of the last descriptor to the Descriptor Controller's control logic. The DMA begins fetching descriptors. The DMA starts with descriptor ID 0 and finishes with the ID of the last descriptor.
7. After data transfers for the last descriptor complete, the Descriptor Controller writes 1'b1 to the `Done` bit in the descriptor table entry corresponding to the last descriptor in the PCIe domain using the `txs` port.
8. Software polls the `Done` bit in the descriptor table entry corresponding to the last descriptor. After the DMA Controller writes the `Done` bit, the DMA Controller calculates throughput. Software compares the data in the system memory to the on-chip or external memory. The test passes if there are no errors.
9. For simultaneous read and writes, the software begins the read DMA operation before the write DMA operation. The DMA completes when all the read and write DMAs finish.

1.2.5. Setting Up the Hardware

1. Power down computer number 1.
2. Plug the FPGA Development Kit card into a PCIe slot that supports Gen2 x4 or Gen3 x8.
3. For the Intel Stratix 10 10 FPGA Development Kit, connectors J26 and J27 power the card. After inserting the card into an available PCIe slot, connect 2x4- and 2x3-pin PCIe power cables from the power supply of computer number 1 to the J26 and J27 of the PCIe card, respectively.
4. Connect a USB cable from computer number 2 to the FPGA Development Kit.

The Development Kit includes an on-board Intel FPGA Download Cable for FPGA programming.

5. To power up FPGA Development Kit via the PCIe slot, power on computer number 1. Alternatively, you can power up FPGA Development Kit using the external power adapter that ships with the kit.
6. For the Intel Cyclone 10 GX FPGA Development Kit, an on-board programmable oscillator is the clock source for hardware components. Follow the instructions in *Setting Up the Intel Cyclone 10 GX FPGA Programmable Oscillator* to program this oscillator.
7. On computer number 2, bring up the Intel Quartus Prime programmer and configure the FPGA through an Intel FPGA Download Cable.

   Note: You must reconfigure the FPGA whenever the FPGA Development Kit loses power.
8. To force system enumeration to discover the PCIe device, restart computer 1. If you are using the Intel Stratix 10 GX FPGA Development Kit, you might get the following error message during BIOS initialization if the memory mapped I/O is only 4 GB: *Insufficient PCI Resources Detected*. To work around this issue, enable `Above 4G Decoding` in the BIOS Boot menu.

Related Information

Programming the Intel Cyclone 10 GX FPGA Oscillator on page 12

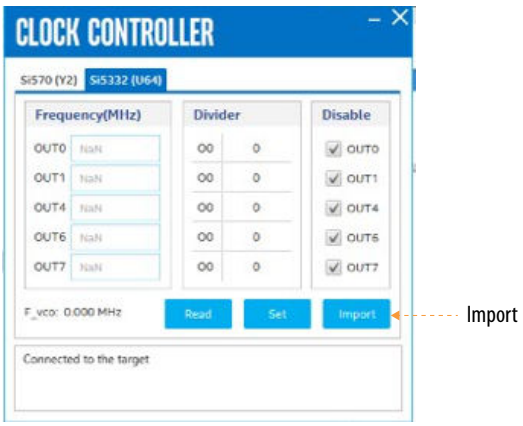
1.2.6. Programming the Intel Cyclone 10 GX FPGA Oscillator

The Intel Cyclone 10 GX Development Kit includes a programmable oscillator that you must set up before you can run the reference design for Intel Cyclone 10 GX devices. A `ClockController` GUI allows you to import the correct settings.

1. Locate the Kit Collateral (zip) link in the **Documentation** area of the Intel Cyclone 10 GX FPGA Development Kit web page.
2. Use this link to download `cyclone-10-gx-kit-collateral.zip`
3. Unzip `cyclone-10-gx-kit-collateral.zip` to a working directory on computer number 2.
4. To bring up the **Clock Controller** dialog box, type the following commands:

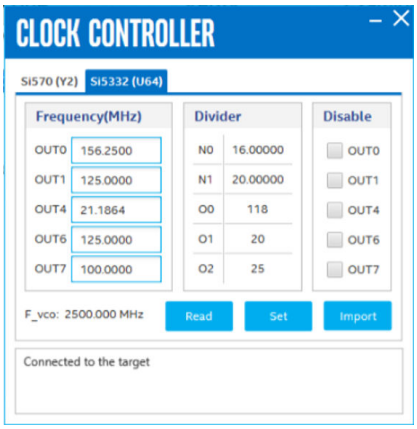
```
% cd <install_dir>/cyclone-10-gx-collateral/examples/board_test_system/
% ./ClockController.sh
```

Figure 6. Clock Controller GUI in Initial State



5. In the **Clock Controller** GUI, click **Import**.
6. Browse to the `<install_dir>/cyclone-10-gx-collateral/examples/board_test_system/` directory and select `U64-Registers.txt`.
7. To import the register settings, click **Open**.
   The message, *Si5332 Register Map is imported successfully* displays. You should see the clock settings shown below.

Figure 7. Clock Settings for Intel Cyclone 10 GX FPGA Development Kit



Related Information

Setting Up the Hardware on page 11

1.2.7. Installing the DMA Test Driver and Running the Linux DMA Software

1. In a terminal window on computer 1, change to the DMA driver directory and extract `AN829_driver.tar` by typing the following commands:

```
cd % <install_dir>/<device>/_PCIe<GenxN>DMA_<QuartusVer>_project/driver
% tar -xvf AN829_driver.tar
```

2. To install the Linux driver for the appropriate device family, type the command:

```
% sudo./install <device_family>
```

   Valid values for <*device_family*> are `arria10`, `cyclone10`, and `stratix10`.

3. To run the DMA application, type the following command:

```
% ./run
```

   The application prints the commands available to specify the DMA traffic. By default, the software enables DMA reads, DMA writes, and Simultaneous DMA reads and writes. The following table lists the available commands:

Table 11. DMA Test Commands

Command Number	Function
1	Start the DMA.
2	Enable or disable read DMA.
3	Enable or disable write DMA.
4	Enable or disable simultaneous read and write DMA.
5	Set the number of dwords per descriptor. The legal range is 256-4096 dwords.
6	Set the number of descriptors. The legal range is 1-127 descriptors.
7	By default, the reference design selects on-chip memory. If select this command consecutive runs switch between on-chip and external memory.
8	Run a the DMA in a continuous loop.
10	Exit

   For example, type the following commands to specify 4096 dwords per descriptor and 127 descriptors:

```
% 5 4096
% 6 127
% 8
```

   The following figures show the throughput for DMA reads, DMA writes, and simultaneous DMA reads and writes:

Figure 8. Intel Arria 10 DMA Throughput

```
*********************************************
** ALTERA 256b DMA driver                   **
** version 2.02                              **
** 1) start DMA                             **
** 2) enable/disable read dma               **
** 3) enable/disable write dma              **
** 4) enable/disable simul dma              **
** 5) set num dwords (256 - 4096)           **
** 6) set num descriptors (1 - 127)         **
** 7) toggle on-chip or off-chip memory     **
** 8) loop dma                              **
** 10) exit                                 **
*********************************************
Access On Chip RAM      ? 1
Run Read                ? 1
Run Write               ? 1
Run Simultaneous        ? 1
Read Passed             ? 1
Write Passed            ? 1
Simultaneous Passed     ? 1
Read EPLast timeout     ? 0
Write EPLast timeout    ? 0
Number of Dwords/Desc   : 4096
Number of Descriptors   : 128
Length of transfer      : 2048 KB
Rootport address offset : 0
Read Time               : 0 s and 307 us
Read Throughput         : 6.384941 GB/S
Write Time              : 0 s and 308 us
Write Throughput        : 6.343481 GB/S
Simultaneous Time       : 0 s and 334 us
Simultaneous Throughput : 11.699353 GB/S
```

Figure 9. Intel Cyclone 10 GX DMA Throughput

```
** ALTERA 256b DMA driver                    **
** version 2.02                               **
** 1) start DMA                              **
** 2) enable/disable read dma                **
** 3) enable/disable write dma               **
** 4) enable/disable simul dma               **
** 5) set num dwords (256 - 4096)            **
** 6) set num descriptors (1 - 127)          **
** 7) toggle on-chip or off-chip memory      **
** 8) loop dma                               **
** 10) exit                                  **
*********************************************
Access On Chip RAM      ? 1
Run Read                ? 1
Run Write               ? 1
Run Simultaneous        ? 1
Read Passed             ? 1
Write Passed            ? 1
Simultaneous Passed     ? 1
Read EPLast timeout     ? 0
Write EPLast timeout    ? 0
Number of Dwords/Desc   : 4096
Number of Descriptors   : 128
Length of transfer      : 2048 KB
Rootport address offset : 0
Read Time               : 0 s and 1176 us
Read Throughput         : 1.661388 GB/S
Write Time              : 0 s and 1171 us
Write Throughput        : 1.668482 GB/S
Simultaneous Time       : 0 s and 1277 us
Simultaneous Throughput : 3.059972 GB/S
```

Figure 10. Intel Stratix 10 DMA Throughput

```
*******************************************
** ALTERA 256b DMA driver                **
** version 2.02                           **
** 1) start DMA                          **
** 2) enable/disable read dma            **
** 3) enable/disable write dma           **
** 4) enable/disable simul dma           **
** 5) set num dwords (256 - 4096)        **
** 6) set num descriptors (1 - 127)      **
** 7) toggle on-chip or off-chip memory  **
** 8) loop dma                           **
** 10) exit                              **
*******************************************
Access On Chip RAM      ? 1
Run Read                ? 1
Run Write               ? 1
Run Simultaneous        ? 1
Read Passed             ? 1
Write Passed            ? 1
Simultaneous Passed     ? 1
Read EPLast timeout     ? 0
Write EPLast timeout    ? 0
Number of Dwords/Desc   : 512
Number of Descriptors   : 128
Length of transfer      : 256 KB
Rootport address offset : 0
Read Time               : 0 s and 42 us
Read Throughput         : 5.814857 GB/S
Write Time              : 0 s and 40 us
Write Throughput        : 6.105600 GB/S
Simultaneous Time       : 0 s and 47 us
Simultaneous Throughput : 10.392511 GB/S
#
```

1.3. Understanding PCI Express Throughput

The throughput in a PCI Express system depends on the following factors:

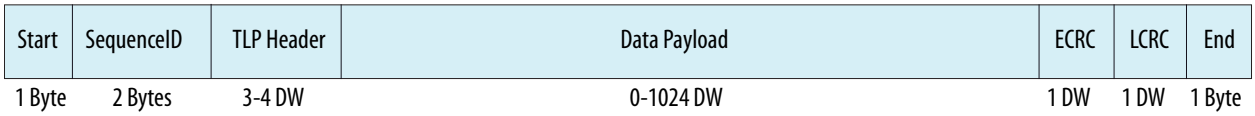
- Protocol overhead
- Payload size
- Completion latency
- Flow control update latency
- Devices forming the link

Protocol Overhead

Protocol overhead includes the following three components:

- 128b/130b Encoding and Decoding—Gen3 links use 128b/130b encoding. This encoding adds two synchronization (sync) bits to each 128-bit data transfer. Consequently, the encoding and decoding overhead is very small at 1.56%. The effective data rate of a Gen3 x8 link is about 8 gigabytes per second (GBps).
- Data Link Layer Packets (DLLPs) and Physical Layer Packets (PLPs)—An active link also transmits DLLPs and PLPs. The PLPs consist of SKP ordered sets which are 16-24 bytes. The DLLPs are two dwords. The DLLPs implement flow control and the ACK/NAK protocol.
- TLP Packet Overhead—The overhead associated with a single TLP ranges from 5-7 dwords if the optional ECRC is not included. The overhead includes the following fields:
  - The Start and End Framing Symbols
  - The Sequence ID
  - A 3- or 4-dword TLP header
  - The Link Cyclic Redundancy Check (LCRC)
  - 0-1024 dwords of data payload

Figure 11. TLP Packet Format



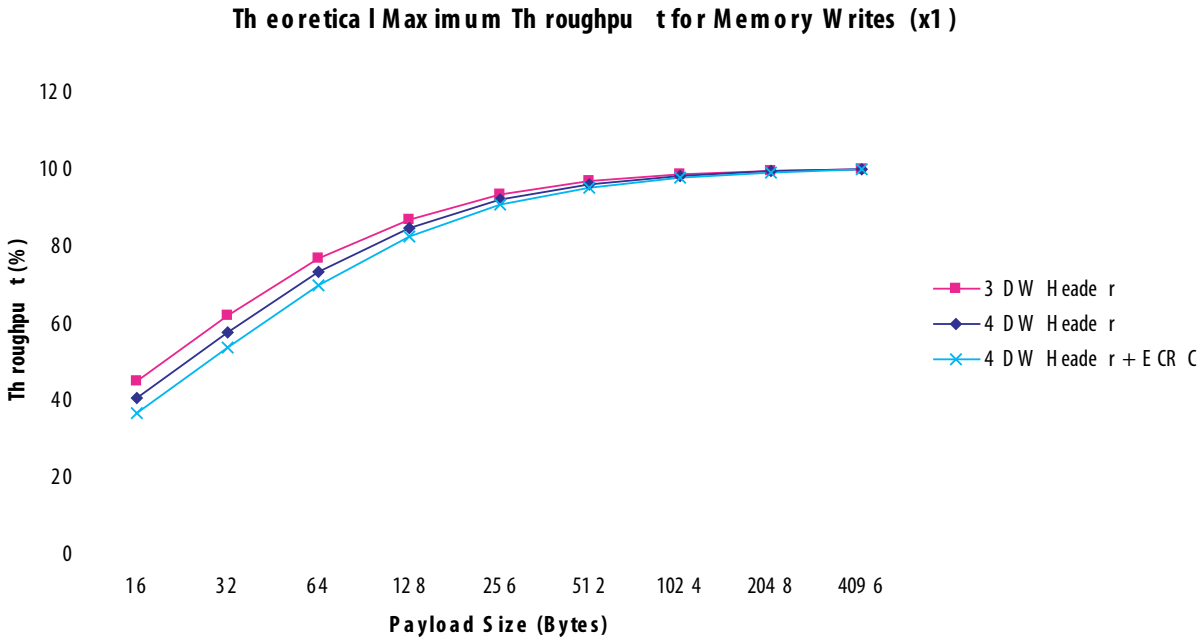
1.3.1. Throughput for Posted Writes

The theoretical maximum throughput calculation uses the following formula:

```
Throughput  = payload size / (payload size + overhead) * link data rate
```

Figure 12. Maximum Throughput for Memory Writes

The graph shows the maximum throughput with different TLP header and payload sizes. The DLLPs and PLPs are excluded from this calculation. For a 256-byte maximum payload size and a 3-dword header the overhead is five dwords. Because the interface is 256 bits, the 5-dword header requires a single bus cycle. The 256-byte payload requires 8 bus cycles.



The following equation shows maximum theoretical throughput:

```
Maximum throughput = 8 cycles/9 cycles  = 88.88% * 8 GBps = 7.2 GBps
```

1.3.1.1. Specifying the Maximum Payload Size

The `Device Control` register, bits [7:5], specifies the maximum TLP payload size of the current system. The `Maximum Payload Size` field of the `Device Capabilities` register, bits [2:0], specifies the maximum permissible value for the payload. You specify this read-only parameter, called **Maximum Payload Size**, using the parameter editor. After determining the maximum TLP payload for the current system, software records that value in the `Device Control` register. This value must be less than the maximum payload specified in the `Maximum Payload Size` field of the `Device Capabilities` register.

Understanding Flow Control for PCI Express

Flow control guarantees that a TLP is not transmitted unless the receiver has enough buffer space to accept the TLP. There are separate credits for headers and payload data. A device needs sufficient header and payload credits before sending a TLP. When the Application Layer in the completer accepts the TLP, it frees up the RX buffer space in the completer's Transaction Layer. The completer sends a flow control update packet (FC Update DLLP) to replenish the consumed credits to the initiator. When a device

consumes all its credits, the rate of FC Update DLLPs to replenish header and payload credits limits throughput. The flow control updates depend on the maximum payload size and the latencies of two connected devices.

1.3.2. Throughput for Reads

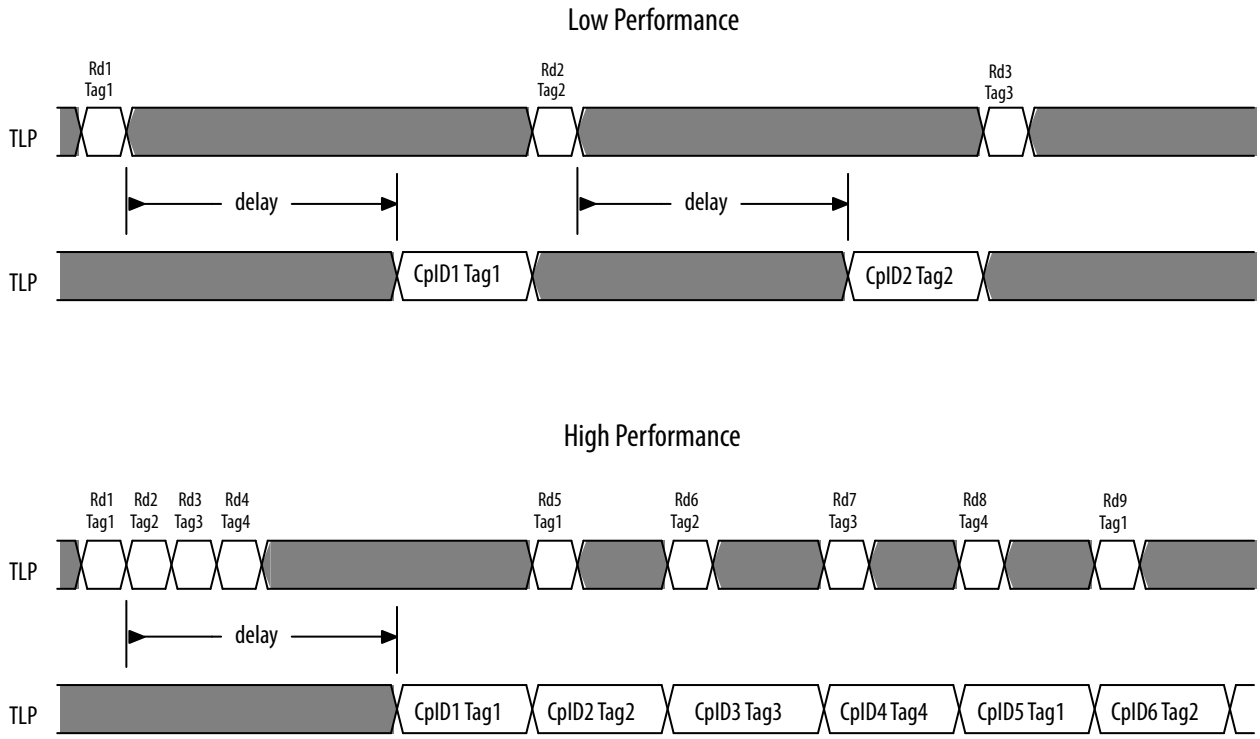
PCI Express uses a split transaction model for reads. The read transaction includes the following steps:

1. The requester sends a Memory Read Request.
2. The completer sends out the ACK DLLP to acknowledge the Memory Read Request.
3. The completer returns a Completion with Data. The completer can split the Completion into multiple completion packets.

Read throughput is typically lower than write throughput because reads require two transactions instead of a single write for the same amount of data. The read throughput also depends on the round trip delay between the time when the Application Layer issues a Memory Read Request and the time when the requested data returns. To maximize the throughput, the application must issue enough outstanding read requests to cover this delay.

Figure 13. Read Request Timing

The figures below show the timing for Memory Read Requests (MRd) and Completions with Data (CplD). The first figure shows the requester waiting for the completion before issuing the subsequent requests. Waiting results in lower throughput. The second figure shows the requester making multiple outstanding read requests to eliminate the delay after the first data returns. Eliminating delays results in higher throughput.



To maintain maximum throughput for the completion data packets, the requester must optimize the following settings:

- The number of completions in the RX buffer
- The rate at which the Application Layer issues read requests and processes the completion data

Read Request Size

Another factor that affects throughput is the read request size. If a requester requires 4 KB data, the requester can issue four, 1 KB read requests or a single 4 KB read request. The 4 KB request results in higher throughput than the four, 1 KB reads. The `Maximum Read Request Size` value in `Device Control` register, bits [14:12], specifies the read request size.

Outstanding Read Requests

A final factor that can affect the throughput is the number of outstanding read requests. If the requester sends multiple read requests to improve throughput, the number of available header tags limits the number of outstanding read requests. To achieve higher performance, Intel Arria 10 and Intel Cyclone 10 GX read DMA can use up to 16 header tags. The Intel Stratix 10 read DMA can use up to 32 header tags.

1.3.2.1. Understanding Throughput Measurement

To measure throughput, the software driver takes two timestamps. Software takes the first timestamp shortly after the you type the `./run` command. Software takes the second timestamp after the DMA completes and returns the required completion status, EPLAST. If read DMA, write DMA and simultaneous read and write DMAs are all enabled, the driver takes six timestamps to make the three measurements.

1.3.3. Throughput Differences for On-Chip and External Memory

This reference design provides a choice between on-chip memory implemented in the FPGA fabric and external memory available on the PCB. The on-chip memory supports separate read and write ports. Consequently, this memory supports simultaneous read and the write DMAs.

The external memory supports a single port. Consequently, the external memory does not support simultaneous read DMA and write DMA accesses. In addition, the latency of external memory is higher than the latency of on-chip memory. These two differences between the on-chip and external memory result in lower throughput for the external memory implementation.

To compare the throughput for on-chip and external memory, select command 7 for consecutive runs to switch between on-chip and external memory.

1.4. Document Revision History for AN 829: PCI Express Avalon-MM DMA Reference Design

Document Version	Intel Quartus Prime Version	Changes
2018.06.11	18.0	Initial release.