User Guide of the PCIe SG DMA Engine on Xilinx ML605 Virtex6 Development Board

V1.6

Wenxue Gao

weng.ziti@gmail.com

Sabatini Simone

sabatini.simone@gmail.com

10 March 2012

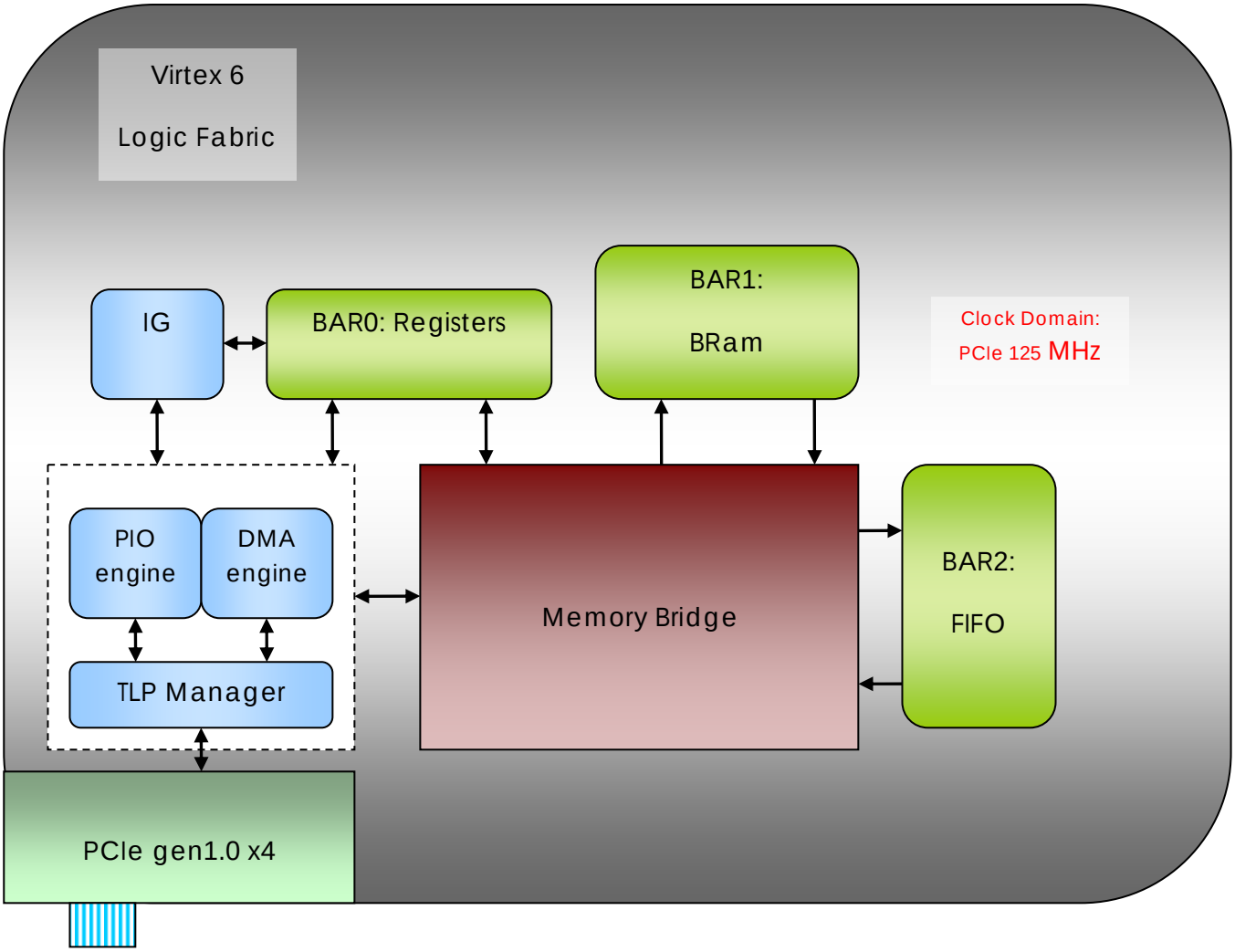
Revision	Date	Comment
1.0	20 Aug 2009	Created.
1.1	26 Nov 2009	Correction of some errors.
1.2	24 Aug 2011	Adapted for OpenCores.org.
1.3	14 Sep 2011	Testbench is added.
1.4	10 Feb 2012	Adapted for Virtex6 ML605
1.5	20 Feb 2012	IP Cores updated to ISE12.3 latest version
1.6	10 Mar 2012	“User Mode” and “LoopBack Mode” added

1. Overview

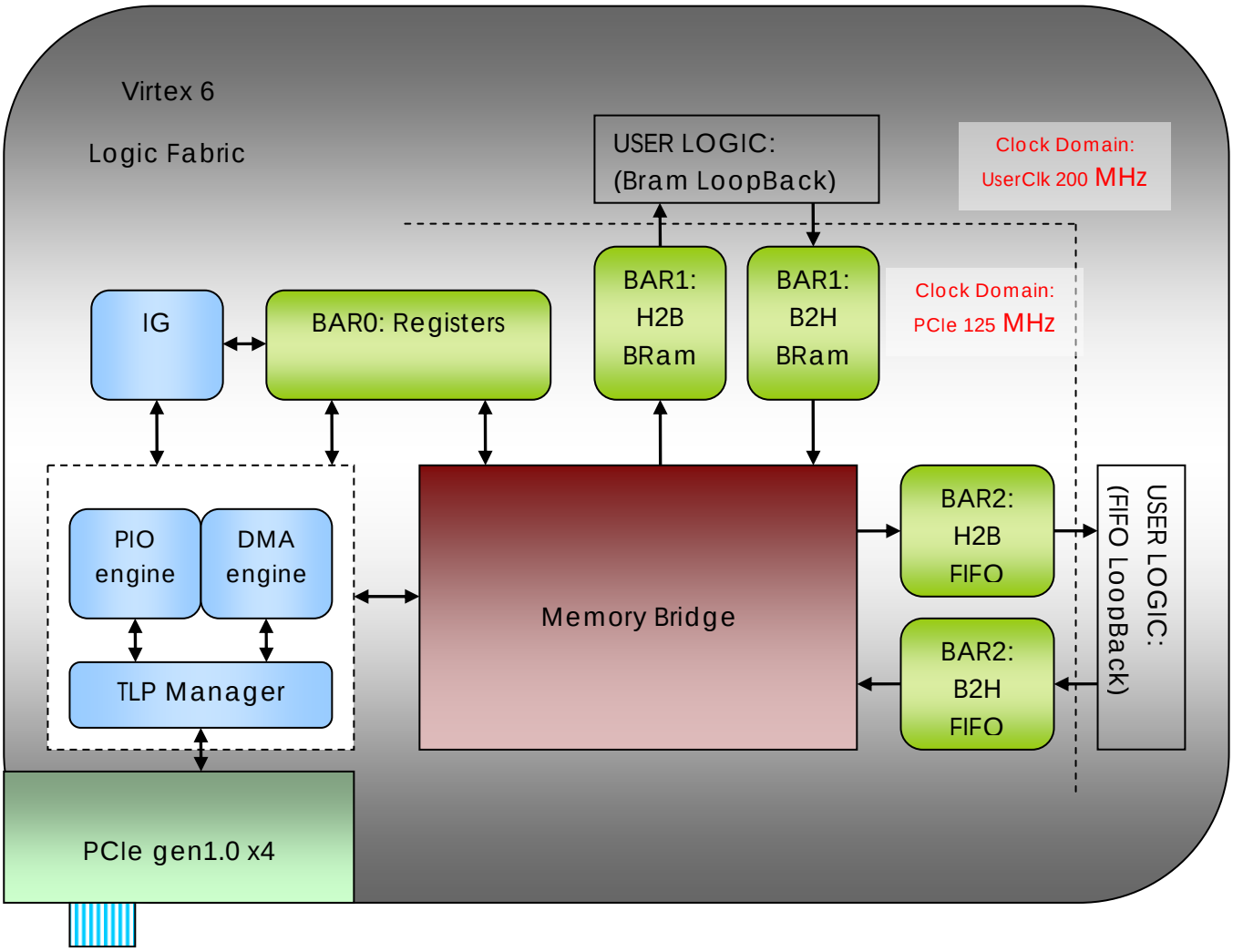
This project for the xc6vlx240t-ff1156-1 FPGA logic fabric implement a PCIe SG DMA Engine.

It can be compiled in two separate modes by changing the value of the constant USE_LOOPBACK_TEST in the “v6abb64Package_efifo_elink.vhd” file:

- *USE_LOOPBACK_TEST* = True → “**LoopBack Mode**”
- *USE_LOOPBACK_TEST* = False → “**User Mode**”



“Figure 1a” is the block diagram for the PCIe SG DMA engine in the “LoopBack Mode”. DMA engine and PIO engine are parallel established. The Memory Bridge is a module like crossbar switch. TLP manager is to manage the virtual channels on the transaction layer of PCIe. Registers space contains the system registers and other register related to DMA and status peeking. FIFO is internally looped back and the Block RAM (BRAM) module emulates the RAM memory space.



In the “User Mode”, “Figure 1b”, another FIFO and another BRAM are added. These copies allow to decouple the Host2Board data flow from the Board2Host data flow. There are two clocks domains also: the PCIe side is clocked with the 125MHz clock derived from the bus, the User side is clocked with the 200MHz clock present on the ML605 board. The UserLogic realize a loopback for Registers, FIFO and BRAM in the 200MHz clock domain, the schematics is reported in Appendix. It was generated with Xilinx SystemGenerator 12.3. The source project is “PCIe_UserLogic_00.mdl”.

In addition the TLP module provides ten user registers and three user IRQ lines. It is very simple add others registers and IRQ lines.

The design is composed by some Xilinx IP Cores. Both the vhdl code and the CoreGen .xco file are provided. To change or upgrade them, a valid license for the cores from Xilinx Inc. should be available. The PCIe core is the 1.6 version. The old 1.3 version is also provided.

The project compile with no problem in ISE 12.3. In ISE 12.3 it is important install the “39430-2_map_123_nt32” patch to downgrades GTX POWER_SAVE errors to warnings (see http://www.xilinx.com/techdocs/39430.htm) and modify the “POWER_SAVE” property from "0000100100" to "0000110100" in “gtx_wrapper_v6.vhd”.

Some problems are present in ISE 13.x. These will be solved in the future update.

Implementing the PCIe v1.6 core at 5Gb/sec (gen2.0) for 4 lines results in some timing error but the design seems to work correctly. The same happens for PCIe v1.6 core at 2.5Gb/sec (gen1.0) for 8 lines.

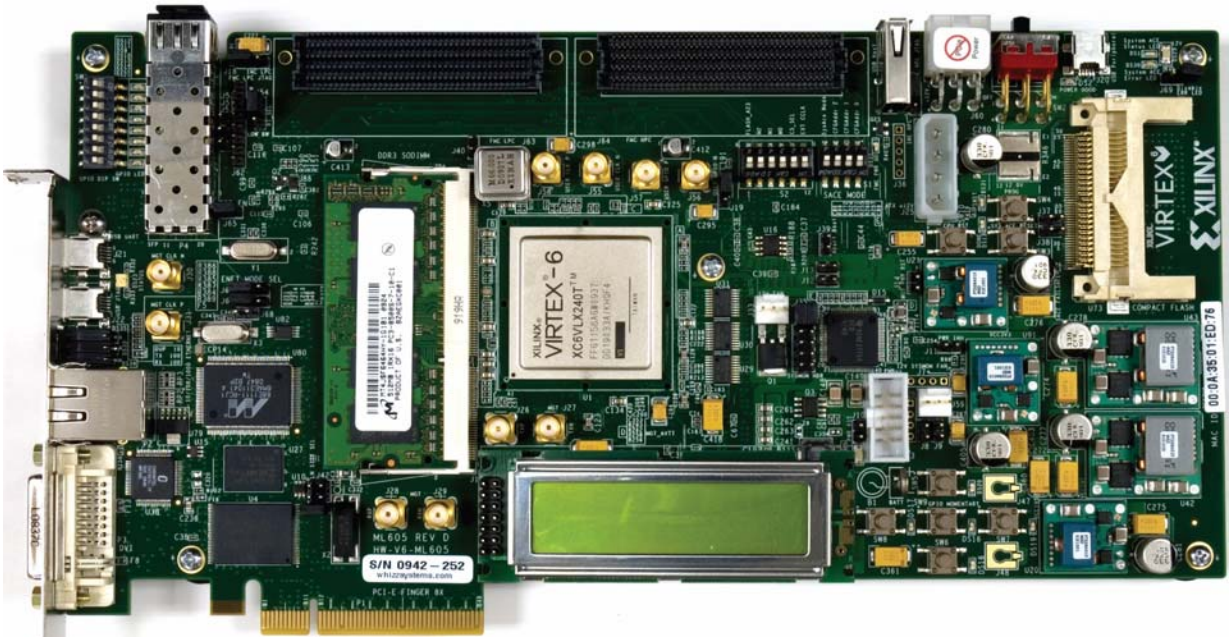
A summary of the tries attempted with several PCIe core implementations and version is in the table:

IP Core Version	Speed	Generation	Lines	Result
V1.3	2.5Gb/sec	Gen 1.0	X4	Success
V1.6	2.5Gb/sec	Gen 1.0	X1	Success
V1.6	2.5Gb/sec	Gen 1.0	X4	Success
V1.6	2.5Gb/sec	Gen 1.0	X8	Compiled but with some timing error
V1.6	5Gb/sec	Gen 2.0	X1	Success
V1.6	5Gb/sec	Gen 2.0	X4	Compiled but with some timing error

In the HDL codes, there are some confusion naming like, DDR_*, Event_*, etc., which are only legacy of a specific project. These will be improved in the future update.

2. Board Characteristics

In terms of hardware layout, the Xilinx ML605 Virtex6 PCIE development board has following major features,



Virtex-6 FPGA

XC6VLX240T-1FFG1156 device

Configuration

Onboard configuration circuitry (USB to JTAG)

16 MB Platform Flash XL

32 MB Parallel (BPI) Flash

System ACE™ CompactFlash (CF) controller

Communication and Networking

10/100/1000 Tri-Speed Ethernet (GMII, RGMII, SGMII, MII)

SFP transceiver connector

GTX port (TX/RX,) with four SMA connectors

USB to UART Bridge

USB host port and USB peripheral port

PCI Express® Gen1 8-lane (x8) and Gen2 4-lane (x4)

Memory

DDR3 SODIMM (512 MB)

Linear BPI Flash (32 MB) (Also available for configuration)

IIC EEPROM (8 Kb)

Clocking

200 MHz oscillator (differential)

66 MHz socketed oscillator (single-ended)

SMA connectors for external clock (differential)

GTX clock port with two SMA connecto

Input/Output and Expansion Ports

16x2 LCD character display

DVI output

System Monitor

User pushbuttons (5), DIP switches (8), LEDs (13)

User GPIO with two SMA connectors

Two FMC expansion ports

High Pin Count (HPC)

- Eight GTX transceivers

- 160 SelectIO™ interface signals

Low Pin Count (LPC)

- One GTX transceiver

- 68 SelectIO interface signals

Power

12V wall adapter or ATX

Voltage and

** FPGA configuration in BPI mode as well as power-up preparation are founded in Appendix.*

3. Memory partition

This design holds 3 BAR's, BAR[0], BAR[1] and BAR[2], as its memory space. Registers are accessed via BAR[0], including the system registers, DMA channel registers and some other control and status registers. Block RAM are assigned to BAR[1], including the 32KB dual-port RAM and the write-only 32KB data generator table RAM. BAR[2] contains the FIFO data port, write and read. FIFO control and status registers reside in BAR[0]. BAR[3] to BAR[6] are reserved. All 3 applied BARs are reachable with PIO operation. DMA can only target on BAR[1] and BAR[2].

Registers are divided into groups, as shown in table 1. BAR[0] and BAR[1] spaces are 4- byte (DW) aligned, i.e. lowest 2 bits of addresses are taken as "00". BAR[2] is 8-byte (QW) aligned and its address offset is arbitrary.

Table 1 Address Assignment

Name	Offset	R/W	BAR
System section			
Design ID Register (DID)	0x000	RO	0
Interrupt Status Register (ISR)	0x000	RO	0
Interrupt Enable Register (IER)	0x001	R + W	0
General Error Register (GER)	0x001	RO	0
General Status Register (GSR)	0x002	RO	0
General Control Register (GCR)	0x002	R + W	0
Upstream DMA channel (Channel#1 in MPRACE)			
Peripheral Address high-DW	0x002	R	0
Peripheral Address low-DW	0x003	R	0
Host address high-DW	0x003	R	0
Host address low-DW	0x003	R	0
Next Buffer Descriptor Address high-DW	0x003	R	0
Next Buffer Descriptor Address low-DW	0x004	R	0
Length in bytes	0x004	R	0
Control	0x004	R	0
Status	0x004	RO	0
Downstream DMA channel (Channel#0 in MPRACE)			
Peripheral Address high-DW	0x005	R	0
Peripheral Address low-DW	0x005	R	0
Host address high-DW	0x005	R	0
Host address low-DW	0x005	R	0
Next Buffer Descriptor Address high-DW	0x006	R	0

Next Buffer Descriptor Address low-DW	0x006	R	0
Length in bytes	0x006	R	0
Control	0x006	R	0
Status	0x007	RO	0
PIO Path Controls			
MRd channel control	0x007	W	0
PCIe transaction layer Tx module control	0x007	W	0
ICAP (*reserved*)			
ICAP port write	0x007	W	0
ICAP port read	0x007	R	0
Interrupt Generation (*See Interrupt Generator chapter*)			
Interrupt Generation Control (IGC)	0x008	R +	0
Interrupt Generation Latency (IGL)	0x008	R +	0
Interrupt Generation On Statistic (IGN_ON)	0x008	RO	0
Interrupt Generation Off Statistic (IGN_OFF)	0x008	RO	0
FIFO Control and Status			
Control	0x009	W	0
Hybrid FIFO Status (Used by PCIe)	0x009	R	0
Host2Board FIFO Status (only in “User Mode”)	0x00		0
Board2 Host FIFO Status (only in “User Mode”)	0x00	R	0
DMA Actual Transferred			
Upstream transferred byte count	0x009	RO	0
Downstream transferred byte count	0x009	RO	0
Large memory (1 MB)			
Block RAM (32 KB)	0x0800 0	R +	1
Other regions reserved			*1*
Event FIFO Data Interface			
Read	0x000	R	2
Write	0x000	W	2

Notes for table 1, R:

Readable.
W: Writeable.
R+W: Readable and writeable. RO:
Read-only. Write no effect. WO:
Write-only. Read as zero.

3.1. Register definition — BAR[0]

Following are some registers definition. Greyed bits are reserved, which are read as zero and should avoid writes with non-zero values.

3.1.1 Design ID (+0x0000)

31 ~ 24	23 ~ 16	15 ~ 12	11 ~ 0
Design version	Design major revision	Author Code	Design minor revision

3.1.2 Interrupt Status Register (+0x0008)

31 ~ 9	8	7	6	5	4	3	2	1	0
	"DLM" User IRQ	"CTL" User IRQ	"DAQ" User IRQ	Down stream Time-out	Up stream Time-out		Interrupt Generator	Down Stream DMA Done	Up stream DMA Done

** In MPRACE library, upstream is channel #1 and downstream #0.*

3.1.3 Interrupt Enable Register (+0x0010)

31 ~ 9	8	7	6	5	4	3	2	1	0
	"DLM" User IRQ	"CTL" User IRQ	"DAQ" User IRQ	Down stream Time-out	Up stream Time-out		Interrupt Generator	Down Stream DMA Done	Up stream DMA Done

** In MPRACE library, upstream is channel #1 and downstream #0.*

3.1.4 General Error Register (+0x0018)

31 ~ 20	20	19	18	17 ~ 0
	Event Buffer Overflow	Event Buffer Time-out	Tx Time-out	

3.1.5 General Status Register (+0x0020)

31 ~ 16	15 ~ 10	9 ~ 8	7~6	5	4	3 ~ 0
	PCIe Link Width		DCB Link Active[1:0]	DG Available	ICAP Busy	

** PCIe maximum link width ——*

'B000000 : *Reserved*
'B000001 : x1
'B000010 : x2
'B000100: x4
'B001000: x8
'B001100 : x12
'B010000 : x16
'B100000 : x32

3.1.6 General Control Register (+0x0028)

31 ~ 14	13 ~ 12	11	10 ~ 8	7 ~ 3	2 ~ 0
	Upstream MWr Attr.		Upstream MWr TC		Msg. Routing

** It is highly recommended that no write be made to General Control Register*.

3.1.7 MRd Channel Control Register (+0x0074)

31 ~ 8	7 ~ 0
	Reset (0x0A)

Write with 0x0A resets Mrd channel (also clears Event Buffer overflow).

3.1.8 PCIe transaction layer Tx module Control Register (+0x0078)

31 ~ 8	7 ~ 0
	Reset (0x0A)

Write with 0x0A resets PCIe transaction layer Tx module.

3.1.9 Event Buffer Control Register (+0x0090, write)

31 ~ 8	7 ~ 0
	Reset (0x0A)

Write with 0x0A resets the Event Buffer (FIFO/FIFOs). During reset, the Event Buffer Status Register is read as zero.

3.1.10 Event Buffer (FIFO) Status Register (+0x0090, read)

31 ~ 26	25 ~ 3	2	1	0
	Data count in QW		Almost full	Empty

** A zero value read from this register is possible for the FIFO and means "unavailable", most probably it is being reset.*

In "LoopBack Mode" this register is the monitor of the FIFO.

In “User Mode” this is an hybrid register that monitor the FIFOs Status and is used by PCIe interface and Linux Driver.

3.1.11 Event Buffer H2B (FIFO) Status Register (+0x00D8, read)

31 ~ 26	25 ~ 3	2	1	0
	Data count in QW		Almost full	Full

** A zero value read from this register is possible for the FIFO and means “unavailable”, most probably it is being reset.*

In “LoopBack Mode” this register is empty.

In “User Mode” this register is the monitor of the Host2Board FIFO Status.

3.1.12 Event Buffer B2H (FIFO) Status Register (+0x00D9, read)

31 ~ 26	25 ~ 3	2	1	0
	Data count in QW	Valid	Almost Empty	Empty

** A zero value read from this register is possible for the FIFO and means “unavailable”, most probably it is being reset.*

In “LoopBack Mode” this register is empty.

In “User Mode” this register is the monitor of the Board2Host FIFO Status.

3.1.13 User Registers (from +0x002C to +0x00D7, write / read)

31 ~ 0
User Register Value

In “LoopBack Mode” these registers are undriven.

In “User Mode” these registers are active. These registers are in the PCIe clock domain. Be careful while routing the corresponding signals, host2board reg data and valid (regXX_td , regXX_tv) and board2host data and valid (regXX_rd, regXX_rv) across multiple timing domains.

3.2. Block RAM — BAR[1]

Block RAM size is 32 KB, 64-bit data bus times 4096 items. It is accessed in 32-bit DW units. It can be used to test DMA functions targeted on conventional RAM memory, with address incremented.

We have the Xilinx PCIe core, which manages the physical layer and the data link layer of PCIe, so we have only to care about the transaction layer. The logic analyzes the TLPs from the host and reacts also in TLPs, both via the transaction layer interface.

"_wr_" is write side signal to the BRAM module and "_rdc_" is the read commands to the BRAM. Borrowing PCIe terminology, "_wr_" is posted and "_rdc_" is non-posted. For the "_rdc_" commands, the BRAM module prepares proposed packets and writes them into a FIFO, then the user logic can read out these packets, building them into TLP and sends back to the host. DDR_FIFO_RdEn , DDR_FIFO_Empty , DDR_FIFO_RdQout are the read side signals of this FIFO.

The "ports" of the interface are described below:

```
DDR_wr_sof        -- start of frame
DDR_wr_eof        -- end of frame
DDR_wr_v          -- write valid
DDR_wr_FA         -- deprecated, you can delete
DDR_wr_Shift      -- in some cases, the 64-bit data should be re-aligned to fit in the TLP
DDR_wr_Mask       -- If valid, the corresponding 32-bit should not be written into the
BRAM
DDR_wr_din        -- the write data bus
DDR_wr_full       -- flow control signal. If asserted, the write should be paused

DDR_rdc_sof       -- read command start
DDR_rdc_eof       -- read command end
DDR_rdc_v         -- read command valid
DDR_rdc_FA        -- deprecated.
DDR_rdc_Shift     -- similar to DDR_wr_Shift, for 64-bit / 32-bit realignment
DDR_rdc_din       -- read command data
DDR_rdc_full      -- read command buffer is full. Flow control signal.

DDR_FIFO_RdEn     -- read FIFO read-enable
DDR_FIFO_Empty    -- read FIFO empty
DDR_FIFO_RdQout   -- read FIFO data out

DDR_Ready         -- deprecated
DDR_Blinker       -- debug signal, you can ignore
DMA_ds_Start      -- Downstream DMA (DMA write) start signal
```

3.3. FIFO — BAR[2]

The current version FIFO is built on a built-in FIFO primitive inside the FPGA, 64-bit data bus width for 16384 elements. Externally this buffer is treated exactly as an asynchronous dual-port FIFO. Its empty and almost full flags are connected to the Event Buffer Status Register (+0x0090). If an empty FIFO is read, the time-out might happen. Time-out can be cleared by a reset (+0x0A) written to the Event Buffer Control register (+0x0090). Overwrite is monitored in GER[20] and is cleared either by an Event Buffer reset or an MRd channel reset.

Event Buffer can be accessed with DMA or PIO. DMA size is exactly what the descriptor defines. However, the PIO TLP (Transaction Layer Packet) payload is always in 32-bit wide in our system, the higher 32 bits in the Event Buffer will get lost via PIO. In this sense, PIO and DMA transactions shall not be mixed for the same section of data transfer.

In the “User Mode” the FIFO are two of the same length as the one used in the “LoopBack Mode” and the H2B FIFO Status register and B2H Status register are available to monitor the each FIFO.

4. DMA Functions

4.1. Scatter-Gather DMA

A DMA transfer is initiated via writing the first descriptor to corresponding DMA channel registers. The DMA engine supports multiple descriptors, which provides flexibility to applications in USER memory mode.

In multiple-descriptor DMA, the next descriptor (a group of parameters) is requested according to the Next Buffer Descriptor Address in current DMA descriptor after the current descriptor is executed. Then the DMA engine will execute that new descriptor. This process will continue until a descriptor with the END bit asserted in the Control parameter is done.

4.2. Descriptor description

Table 2 lists the descriptor definition for a DMA channel.

Table 2 DMA channel registers / Buffer descriptor

Name	R/W	Address Offset	+3								+2								+1								+0							
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Peri_Addr_H	R+W	+0x00	Higher 32-bit of DMA Peripheral address																															
Peri_Addr_L	R+W	+0x04	Lower 32-bit of DMA Peripheral address																															
Host_Addr_H	R+W	+0x08	Higher 32-bit of DMA Host address																															
Host_Addr_L	R+W	+0x0C	Lower 32-bit of DMA Host address																															
Next_BDA_H	R+W	+0x10	Higher 32-bit of next descriptor address																															
Next_BDA_L	R+W	+0x14	Lower 32-bit of next descriptor address																															
Length	R+W	+0x18	DMA transaction byte count																															
Control	R+W	+0x1C							V	Last				UPA		BN2	BN1	BN0	Anc							End	Rst (read as 0)							
Status	R	+0x20																												TO			Busy	Done

BN[2:0]: Encoded BAR Number. Default is 000.

V: Valid Control. A special bit for DMA channels. If set, the DMA engine uses this Control word; else, the DMA engine takes the previous Control word of that channel for current transaction. Even the reset command must also set this bit. Default value is '0'. Note: for the first descriptor this bit must be asserted.

Last: If set, the current descriptor is the last one for the current DMA chain. After the descriptor is processed, the DMA engine stops. Default value is '0'.

UPA: Use PA (Peripheral Address) from each descriptor, otherwise, use calculated PA.

Default value is ‘0’. But for the initial descriptor, PA is always used, therefore this bit can be of any value for the first descriptor.

AInc: Address increments. If set, the address of the endpoint side increases by four for every double word transfer. Default value is ‘1’.

End: If set, the DMA engine is paused after this descriptor is processed. This means, the engine can be resumed afterwards. Default value is ‘0’.

Rst: Channel can be reset by write “0x0200000A” to the corresponding Channel Control register. These bits return always 0x0 by reading.

TO: Time-out signal. If asserted, this bit indicates that a time-out event has occurred during the DMA operation. Default value is ‘0’.

Busy: DMA engine is running. Default value is ‘0’.

Done: DMA engine is already finished. Default value is ‘0’. Once asserted, this bit never changes until a channel reset command comes.

4.3. **DMA Commands**

A write to a DMA channel control register with the valid bit asserted is a DMA command. Bits like Last, UPA and AInc are the parameters of the command. Rst bits and End bit are the type of the command. Concerning the bits V, End and Rst, possible types of commands are listed in table 3.

Table 3 DMA commands

Command	V	End	Rst[7:0]	Description
Reset	1	x	0x0A	Reset the DMA channel state.
Start	1	0	0	Start/Resume DMA transaction.
Stop	1	1	0	Pause the current running DMA transaction, if any.
Redo	0	x	x	Repeat the last DMA command, which had the V bit set.

x: don’t care.

4.4. **DMA Status**

Table 4 explains the DMA status.

Table 4 DMA statuses

State	Busy	Done	Description
Idle	0	0	DMA channel is idle, ready to start new DMA transaction.
Busy	1	0	DMA is running.
Done	0	1	A DMA is already finished.
Unreset	1	1	Abnormal state. A previous DMA is finished but state is not reset before a second DMA is started.

4.5. **DMA Chain**

For a DMA chain consisting of multiple descriptors, subsequent descriptors are in the same order and format as the initial one, yet resident in host memory. To do a chained DMA, the upper-level application should get the next descriptors prepared in the host memory before it initiates the first DMA by writing the first descriptor in the peripheral. The DMA engine goes along the chain until the one with Last=1 is executed, after which the DMA finishes.

Figure 3 illustrates a non-loop DMA consisting of multiple descriptors and figure 4 illustrates a looped DMA consisting of multiple descriptors. If the descriptor #0 in figure 3 is labeled with `Last=1`, it will turn out to be a single-descriptor DMA transaction.

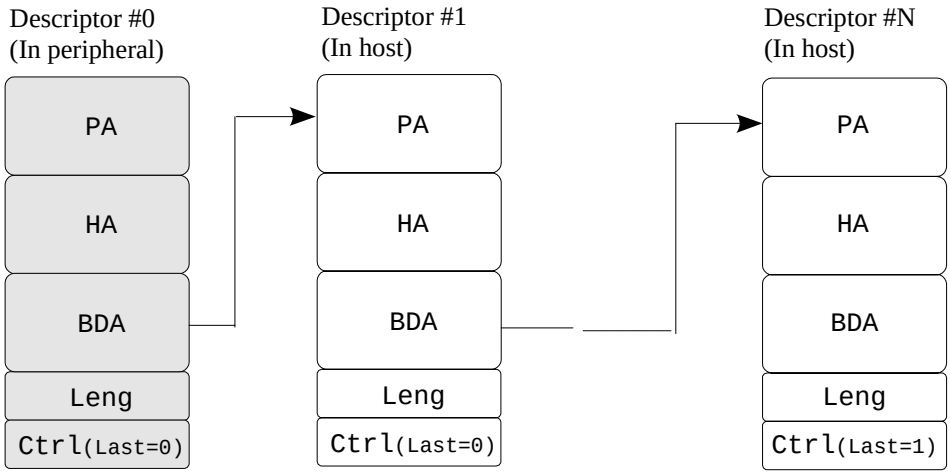


Figure 3 DMA Chain illustration – not looped

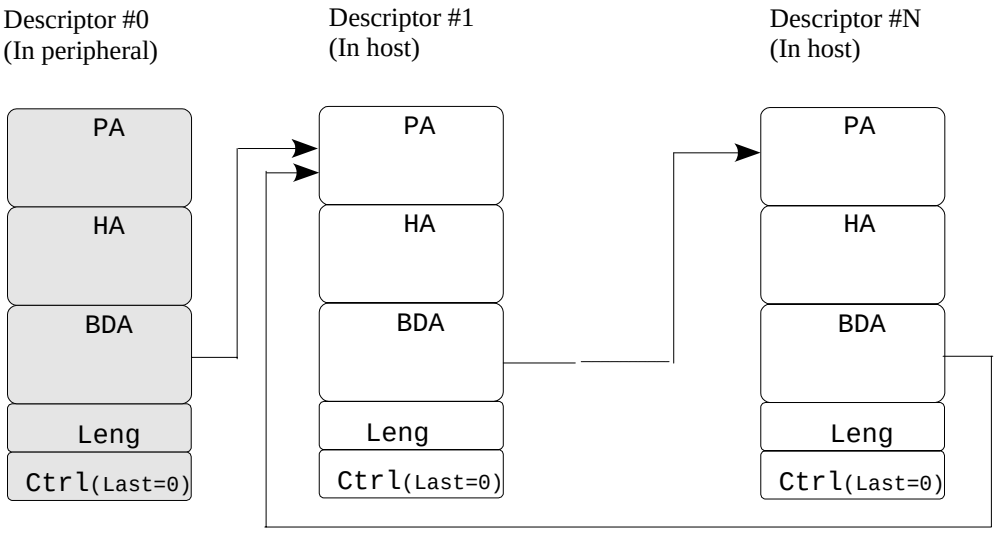


Figure 4 DMA Chain illustration – looped

5. Interrupt Generator

IGC (Interrupt Generator Control) and IGL (Interrupt Generator Latency) are used for the moment as the Interrupt Generator control and parameter registers. The corresponding bit in the Interrupt Status register (ISR) is bit 2.

To enable the IGC you have to compile the project with the IMP_INT_GENERATOR constant at True in the "v6abb64Package_efifo_elink.vhd" file.

A unit in IGL register is calculated as 8 nS in ABB2 4-lane version. The value in IGL is the delay between two interrupts generated. To start the Interrupt Generator, a non-zero value is necessary to the IGL register. Thus, a zero written to the IGL register serves as a pause command, which does not reset the statistics registers but makes the Interrupt Generator stop generating interrupts. Resuming the paused interrupt issuing is done by writing a non-zero value to the IGL register. To emulate the interrupt process service, another feature word ("0x00F0") is needed to be written into the IGC register, which clear one interrupt every time.

An example procedure is given below.

```
*(0xED000080) = 0x0A;         // Reset Interrupt Generator
*(0xED000010) = 0x0004;       // Enable interrupt generation
*(0xED000084) = 0x3000;       // Set Interrupt Generator Latency to 98304 ns,
                              //   and trigger it run
```

```
// Interrupt service program
void Int_Service ( )
{
 int Int_Index = *(0xED000008);
 if (Int_Index & 0x0004)         // Interrupt(s) from the Int Generator come
   {
    ... ...                      // Servicing

    *(0xED000080) = 0x00F0;      // Clear one interrupt
   }
}
```

Afterwards, the number of assert interrupts and that of the deassert interrupts can be obtained from the IGN_ON (+0x0088) and IGN_OFF (+0x008C), respectively. Their values give information of the status of the interrupt service performance. These two statistic registers and the Interrupt Generation register are reset by a reset word (0x0A) written to the IGC register (+0x0080).

6. Testbench

Simulation is provided in Verilog HDL, `tf64_pcie_trn.v` and works only for the "LoopBack Mode".

In this simulation environment, 3 modules are instantiated and connected, (1) tlpControl, (2) bram_Control and (3) FIFO_wrapper.

Figure 5 shows the connection inside the testbench.

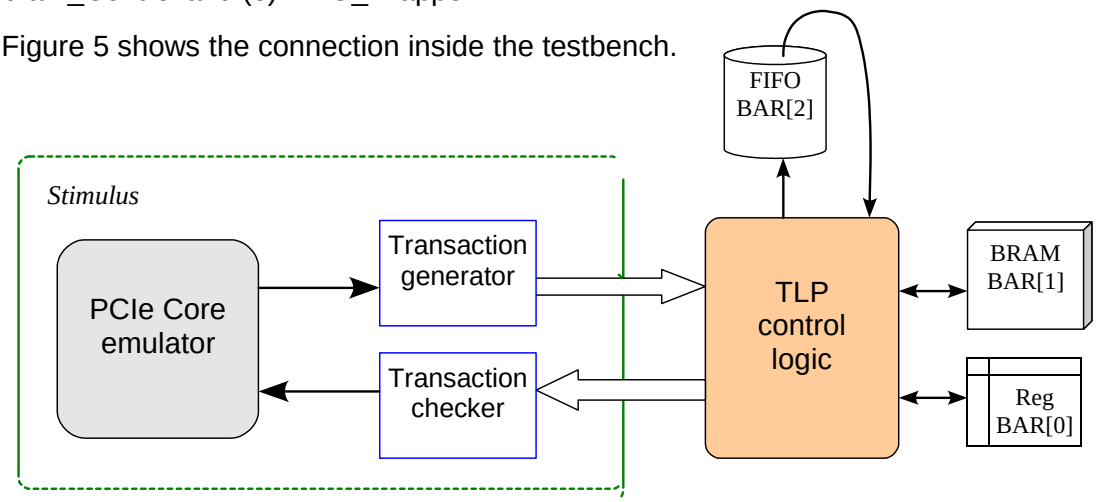


Figure 5 Testbench block diagram

5 types of transactions are simulated, as listed in table 5. Actually the testbench is trying to emulate the behaviour of the PCIe Core in sending and receiving TLPs. In terms of BAR[2] FIFO PIO transactions, note that only the lower 32 bits are accessible. To demonstrate the SG DMA in multiple-descriptor mode, the BAR[2] DMA simulation is done in two descriptors.

Table 5 DMA simulation coverage

	PIO	DMA
BAR[0]	Yes	NO
BAR[1]	Yes	Yes
BAR[2]	Yes *	Yes *

Interrupt is enabled and DMA status polling is simulated. Throttling at Rx and Tx ports can be activated by initializing `Rx_No_Flow_Control` and `Tx_No_Flow_Control` as 0. The simulation runs about 10 µs.

The TLP sending at Rx is made in task `TLP_Feed_Rx`, which is modular for further expansion of the testbench. Rx and Tx TLP format checking is also integrated, so that the upgrade upon the testbench brings no fatal errors.

7. Resource report from ISE 12.3 MAP

Design Summary:

Number of errors:	0
Number of warnings:	6
Slice Logic Utilization:	
Number of Slice Registers:	9,018 out of 301,440 2%
Number used as Flip Flops:	9,017
Number used as Latches:	1
Number used as Latch-thrus:	0
Number used as AND/OR logics:	0
Number of Slice LUTs:	9,315 out of 150,720 6%
Number used as logic:	8,632 out of 150,720 5%
Number using O6 output only:	5,701
Number using O5 output only:	513
Number using O5 and O6:	2,418
Number used as ROM:	0
Number used as Memory:	449 out of 58,400 1%
Number used as Dual Port RAM:	0
Number used as Single Port RAM:	0
Number used as Shift Register:	449
Number using O6 output only:	449
Number using O5 output only:	0
Number using O5 and O6:	0
Number used exclusively as route-thrus:	234
Number with same-slice register load:	197
Number with same-slice carry load:	37
Number with other load:	0

Slice Logic Distribution:

Number of occupied Slices:	3,384 out of 37,680 8%
Number of LUT Flip Flop pairs used:	11,356
Number with an unused Flip Flop:	3,273 out of 11,356 28%
Number with an unused LUT:	2,041 out of 11,356 17%
Number of fully used LUT-FF pairs:	6,042 out of 11,356 53%
Number of unique control sets:	216
Number of slice register sites lost to control set restrictions:	549 out of 301,440 1%

A LUT Flip Flop pair for this architecture represents one LUT paired with one Flip Flop within a slice. A control set is a unique combination of clock, reset, set, and enable signals for a registered element.
The Slice Logic Distribution report is not meaningful if the design is over-mapped for a non-slice resource or if Placement fails.
OVERMAPPING of BRAM resources should be ignored if the design is over-mapped for a non-BRAM resource or if placement fails.

IO Utilization:

Number of bonded IOBs:	10 out of	600	1%
Number of LOCed IOBs:	10 out of	10	100%
Number of bonded IPADs:	10		
Number of bonded OPADs:	8		

Specific Feature Utilization:

Number of RAMB36E1/FIFO36E1s:	77 out of	416	18%
Number using RAMB36E1 only:	73		
Number using FIFO36E1 only:	4		
Number of RAMB18E1/FIFO18E1s:	0 out of	832	0%
Number of BUFG/BUFGCTRLs:	5 out of	32	15%
Number used as BUFGs:	5		
Number used as BUFGCTRLs:	0		
Number of ILOGICE1/ISERDESE1s:	0 out of	720	0%
Number of OLOGICE1/OSERDESE1s:	0 out of	720	0%
Number of BSCANs:	0 out of	4	0%
Number of BUFHCEs:	0 out of	144	0%
Number of BUFOs:	0 out of	36	0%
Number of BUFIODQSs:	0 out of	72	0%
Number of BUFRs:	0 out of	36	0%
Number of CAPTUREs:	0 out of	1	0%
Number of DSP48E1s:	0 out of	768	0%
Number of EFUSE_USRs:	0 out of	1	0%
Number of FRAME_ECCs:	0 out of	1	0%
Number of GTXE1s:	4 out of	20	20%
Number of LOCed GTXE1s:	4 out of	4	100%
Number of IBUFDS_GTXE1s:	1 out of	12	8%
Number of LOCed IBUFDS_GTXE1s:	1 out of	1	100%
Number of ICAPs:	0 out of	2	0%
Number of IDELAYCTRLs:	0 out of	18	0%
Number of IODELAYE1s:	0 out of	720	0%
Number of MMCM_ADVs:	1 out of	12	8%
Number of LOCed MMCM_ADVs:	1 out of	1	100%
Number of PCIE_2_0s:	1 out of	2	50%
Number of LOCed PCIE_2_0s:	1 out of	1	100%
Number of STARTUPs:	1 out of	1	100%
Number of SYSMONs:	0 out of	1	0%
Number of TEMAC_SINGLEs:	0 out of	4	0%

8. Speed Result

Platform: DELL Precision T5500, Linux Debian 2.6.32 64bit.

CORE: PCIe v1.6 **gen2.0 x1**		No Timing Errors	
Testing DMA Write performance:		Testing DMA Read performance:	
4 kB	: 258.305 MBps	4 kB	: 277.967 MBps
8 kB	: 319.732 MBps	8 kB	: 330.418 MBps
16 kB	: 365.444 MBps	16 kB	: 366.937 MBps
32 kB	: 391.862 MBps	32 kB	: 390.687 MBps
64 kB	: 406.587 MBps	64 kB	: 400.945 MBps
128 kB	: 414.446 MBps	128 kB	: 410.371 MBps
256 kB	: 418.421 MBps	256 kB	: 413.871 MBps
512 kB	: 420.769 MBps	512 kB	: 415.373 MBps
1024 kB	: 421.967 MBps	1024 kB	: 416.503 MBps
2048 kB	: 422.674 MBps	2048 kB	: 416.934 MBps
4096 kB	: 426.149 MBps	4096 kB	: 417.142 MBps

CORE: PCIe v1.6 **gen1.0 x4**		No timing Errors	
Testing DMA Write performance:		Testing DMA Read performance:	
4 kB	: 355.568 MBps	4 kB	: 293.596 MBps
8 kB	: 475.379 MBps	8 kB	: 377.965 MBps
16 kB	: 633.597 MBps	16 kB	: 436.384 MBps
32 kB	: 715.627 MBps	32 kB	: 471.844 MBps
64 kB	: 765.788 MBps	64 kB	: 493.319 MBps
128 kB	: 800.855 MBps	128 kB	: 504.566 MBps
256 kB	: 818.678 MBps	256 kB	: 510.197 MBps
512 kB	: 827.609 MBps	512 kB	: 513.116 MBps
1024 kB	: 832.016 MBps	1024 kB	: 514.594 MBps
2048 kB	: 834.342 MBps	2048 kB	: 515.345 MBps
4096 kB	: 828.808 MBps	4096 kB	: 524.622 MBps

CORE: PCIe v1.6 **gen1.0 x8** or **gen2.0 x4**		Timing Errors	
Testing DMA Write performance:		Testing DMA Read performance:	
4 kB	: 528.582 MBps	4 kB	: 508.326 MBps
8 kB	: 664.743 MBps	8 kB	: 619.924 MBps
16 kB	: 917.387 MBps	16 kB	: 827.973 MBps
32 kB	: 1087.32 MBps	32 kB	: 895.018 MBps
64 kB	: 1223.95 MBps	64 kB	: 981.568 MBps
128 kB	: 1281.89 MBps	128 kB	: 1007.13 MBps
256 kB	: 1314.02 MBps	256 kB	: 1032.15 MBps
512 kB	: 1333.11 MBps	512 kB	: 1039.83 MBps
1024 kB	: 1342.96 MBps	1024 kB	: 1045.13 MBps
2048 kB	: 1347.37 MBps	2048 kB	: 1047.17 MBps
4096 kB	: 1343.45 MBps	4096 kB	: 1043.61 MBps

9. Appendix: SystemGenerator LoopBack UserLogic Schematic

200MHz UserClock Domain