Trivial DMA Architecture

The slides that follow illustrate the high level functionality and data path interconnect for the Trivial DMA Engine which is distributed in the Trivial DMA example provided with the Modular PCIe SOPC Builder Bridge examples..

Here is an overview of what each slide contains:

Page 2 – Implementation Strategy – this slide shows the high level strategy for integrating the Modular PCIe SOPC Builder Bridge with the trivial DMA engine.

Page 3 – High Level Architecture – this slide illustrates the architecture of the trivial DMA engine.

Page 4 – High Level Architecture, How it works – this is the same slide as the High Level Architecture with additional text added to describe the functionality of the structure.

General Description

The Trivial DMA engine interfaces with the PCIe host thru a very basic 64-bit DMA descriptor format. The format is defined as such:

```
descriptor 64-bit: { desc_type[ 0 : 0 ], reserved[ 14 : 0 ], channel[ 7 : 0 ], word_count[ 7 : 0 ], address[ 31 : 0 ] }

     desc_type : 1 bit field : specifies descriptor type
                             : 1'b0 = write descriptor
                             : 1'b1 = read descriptor

     reserved : 15 bit field : value does not matter
                             : set to ZERO

       channel : 8 bit field : specifies what Avalon ST channel the returning read data shall occupy, meaningless for write descriptors
                             : 8'bxxxx_xxx0 = write data
                             : 8'bxxxx_xxx1 = descriptor data

    word_count : 8 bit field : specifies the number of 64-bit words to be read or written
                             : 8'h01 = minimum valid word count
                             : 8'hFF = maximum valid word count

      address : 32-bit field : specifies the byte address for the transaction
                             : must be a 64-bit word aligned address, i.e. the 3 lsb of the address must be ZERO
```

These descriptors may be manually pushed into the descriptor slave interface provided by the Trivial DMA engine, or a the Trivial DMA engine can fetch the descriptors itself using it's read master interface. The engine must be started by pushing at least one descriptor into it's descriptor slave, but that descriptor can instruct the engine to then fetch more and more descriptors as it continues to operate. Here's a simple example of what a potential descriptor pattern might look like:

Create an initial descriptor to push into the descriptor slave:

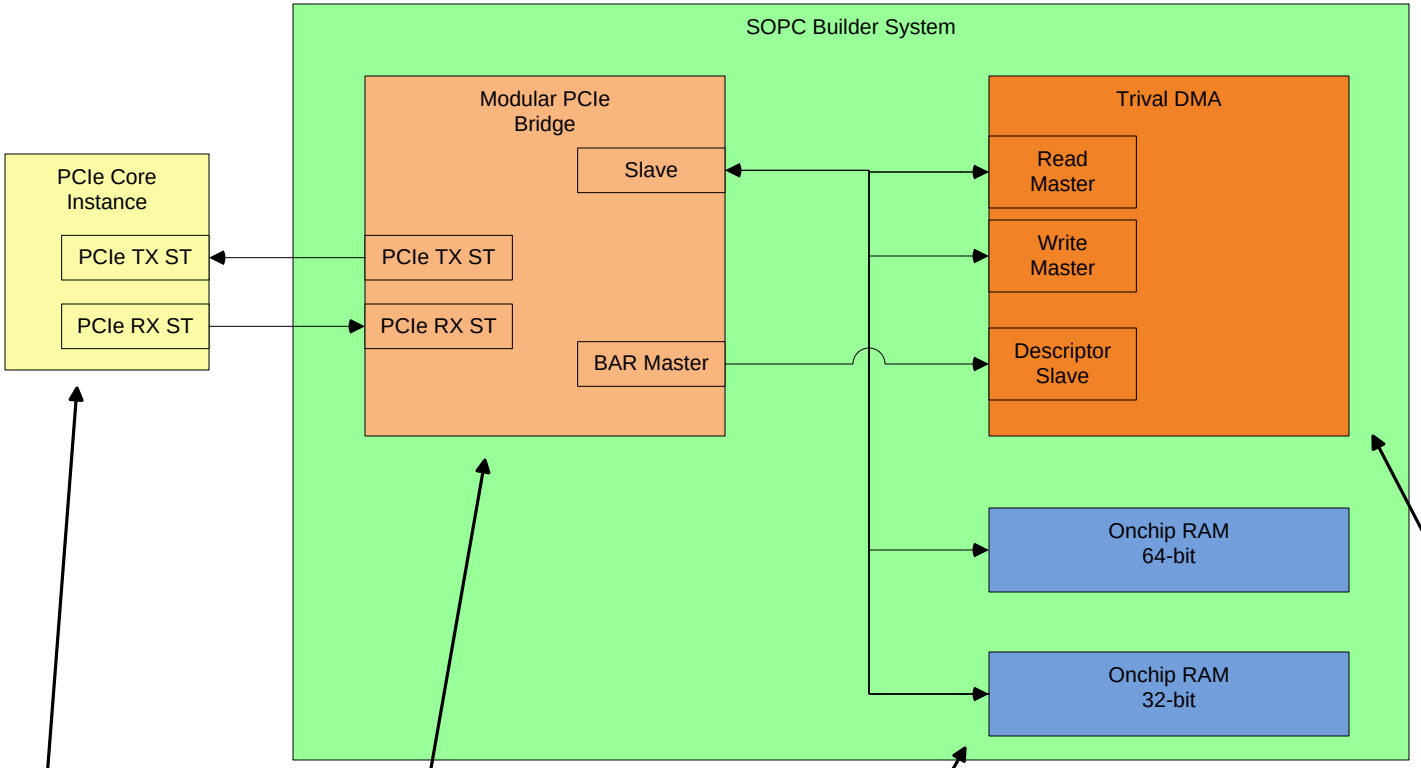
```
{ RD_DESC, DESC, 7, PCIE + 0 } – Read descriptor, reading descriptors, read 7 words, starting at PCIe slave base + 0
```

Then at PCIe + 0 the Trivial DMA Engine would fetch these descriptors:

```
1 - { WR_DESC, xxxx, 128, OCRAM + 0 } – Write descriptor, write 128 words, starting at Onchip RAM base + 0
2 - { RD_DESC, DATA, 128, PCIE + 0x1000 } – Read descriptor, reading data, read 128 words, starting at PCIe slave base + 0x1000
3 - { WR_DESC, xxxx, 128, PCIE + 0x2000 } – Write descriptor, write 128 words, starting at PCIe slave base + 0x2000
4 - { RD_DESC, DATA, 128, OCRAM + 0 } – Read descriptor, reading data, read 128 words, starting at Onchip RAM base + 0
5 - { WR_DESC, xxxx, 1, PCIE + 0x0 } – Write descriptor, write 1 word, starting at PCIe slave base + 0x0
6 - { RD_DESC, DATA, 1, PCIE + 0x8 } – Read descriptor, reading data, read 1 words, starting at PCIe slave base + 0x8
7 - { RD_DESC, DESC, 7, PCIE + 0x100 } – Read descriptor, reading descriptors, read 7 words, starting at PCIe slave base + 0x100
```

So in this example descriptors 1 and 2 coordinate moving data from PCIe memory to onchip RAM, then descriptors 3 and 4 coordinate moving data from onchip RAM to PCIe memory. Descriptors 5 and 6 coordinate reading descriptor 2 and writing it back over descriptor 1. And finally descriptor 7 coordinates fetching 7 more descriptors.

Trivial DMA Implementation Strategy



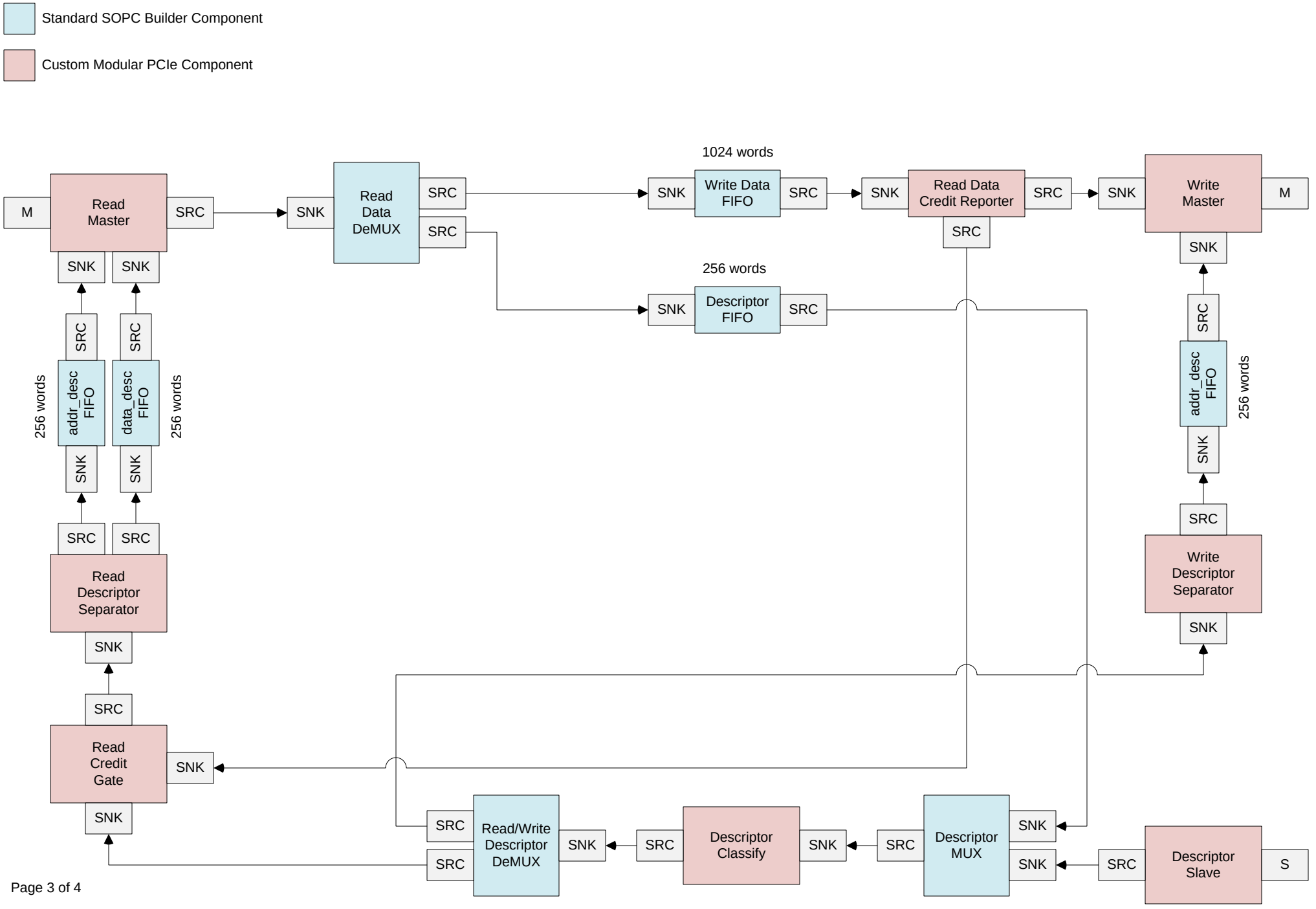
Altera PCIe core is configured and instantiated outside of the SOPC Builder system.

The Modular PCIe SOPC Builder Bridge architecture translates PCIe transactions into Avalon MM transactions and vice versa.

Two onchip RAM targets are provided in the SOPC system, one 64-bit data path and one 32-bit data path. The 64-bit RAM illustrates optimal performance thru the PCIe and DMA system, whereas the 32-bit RAM illustrates a highly restricted performance due to it's narrower datapath.

The Trival DMA structure provides a slave interface that the PCIe host can use to write an initial descriptor into. Then the Read master can be used to fetch additional descriptors from the PCIe memory space. The read and write masters are used to transfer data from either onchip RAM and the PCIe address space.

Trivial DMA
High Level Architecture



Trivial DMA
High Level Architecture

Why does this structure exist? This structure provides a very trivial DMA capability created out of standard SOPC Builder components along with some very trivial custom components to provide the read and write master requirements and descriptor orchestration. The DMA functionality that this structure can provide is full 64-bit word transfers to and from 64-bit word aligned addresses.