RapidIO Trade Association

Suite 325, 3925 W. Braker Lane
Austin, TX 78759
512-305-0070 Tel.
512-305-0009 FAX.

TWG First Showing

Item 11-10-00000.000

Subject: Data Streaming (FType 9) Specification Enhancement

Background: Virtualization is becoming commonplace in Servers and Storage. Efficiency in data centers involving server and storage networks can be improved significantly through hardware virtualization using Remote Direct Memory Access (RDMA). Currently, RapidIO logical layer specifications do not map well into RDMA for virtualized environments. This showing proposes an enhancement to data steaming logical specifcation. The proposal supports RDMA using FType 9 through Read/Write semantics to enable hardware virtualization in RapidIO based networks.

Contributor: Mohammad Akhter, IDT

Comment Expiration Date:

Distribution: RapidIO TA Technical Working Group members

1.2 Introduction

RapidIO has gained significant momentum recently in wireless, military and video applications due to its performance (MHz/Watt/Port), ultra low latency and guaranteed delivery attributes. The protocol is now generating lot of interest in data center network for these same attributes in addition to its built-in QoS and reliability features.

Virtualization is one of the key elements in server and storage networks within a Datacenter. Many applications within Datacenter, such as financial and clouding computing, directly benefit from virtualized environment. The applications in this environment are executed on Virtual Machines (VMs) that share a common physical network interface device or other virtualized hardware component. Typically, compute and storage platforms in virtualized environments include serveral key functions:

- Virtual Machine that executes user applications on Guest OS and include virtual hardware such as virtual network interface card (vNIC)
- Hypervisor that includes Virtual Machine Monitor that maps VMs and vNICs to underlying physical hardware and the Kernel
- Physical Hardware Resources such as CPUs, Memories, Storage, I/O Devices

To facilitate secured data transport within virutalized environment each vNIC may support a large number of queues depending on the application requirements. The RapidIO data streaming specification already supports concept of vNICs and queues. For example, it is possible to map some portion of destination ID to vNIC and streamID/CoS to queues. Although these core components in Data Streaming specification establishes baseline for virtualization, data transport efficiency could further benefit from well established RDMA semantics such as read and write.

1.3 RDMA in Virtualized Environment

RDMA allows computing hardware in a network to perform data exchange between memories without involving the CPU or the Operating System to reduce latency and improve CPU utilization. It is common for the queues in a virtualized environment to support data transfer between application memories through RDMA. This capability significantly enhances I/O efficiency for many applications including file system access, storage, and large data transfer.

Direct IO logical specification in RapidIO currently supports RDMA operation using NWRITE/SWRITE/NREAD semantics. With direct IO although it is possible to map destination ID to vNIC, there is no specific field to map the transactions to the queues. It is possible to map part of the memory address field or the srcTID field to queues, however, this method could significantly impacts bandwidth efficiency in direct IO based RDMA (efficiency could drop below 90%).

At present RapidIO Message Passing Logical Specification could also be used to support RDMA operation in virtualized environment by mapping destination ID to vNIC and combination of xmbox, mbox and letter to queues. However, the number of queues would be limited to 64. Therefore, scalability is an issue in large virtual systems with RDMA based on Message passing logical specification.

To stay competitive, RapidIO 10xN must support enhanced RDMA capabilities in virtualized environment. This showing proposes addition of RDMA Read and Write Semantics to the Data Streaming Specification (Part 10, Ftype 9 packets) to reach this goal. This proposal along with the core Data Streaming Specification leads to the following advantages:

- Supports Large Data Transfer (64KByte at a time)
- Improves bandwidth efficiency - Memory address needs to be sent only once
- Supports better QoS and large number of queues and applications (cos: 8-bit, StreamID: 16-bit)
- Supports RDMA Read/Write Semantics (Xtype, xh)
- Supports Superior Traffic Management (Xtype, xh, TM OP)

1.4 Outstanding Topics

Following topics are outstanding and will be addressed in the updated showing.

- Deadlock Avoidance: This showing introduced requests and responses. We have to ensure there is not deadlock with the new scheme.
- Reducing Overhead in Continuation and End Segments: It is possible to eliminate transID from the Read Continuation segment assuming transactions are part of the same streamID/cos. It is possible to add “nBytes” field to Write Start segment to support error checking mechanism through “length” field in the Write End segment.

1.5 Proposed Changes to Part 10 Data Streaming Logical Specification

In order to support RDMA in virtualized environment, support for RDMA Read and RDMA Write Semantics is incorporated by enhancing the type 9 Packet format.

1.5.1 New Section 4.4 Type 9 Extended RDMA Packet Format

Create a new section 4.4 as follows:

“

The type 9 extended RDMA packet formats to support Read and Write semantics between two data streaming endpoints are shown below. Packet formats are shown for Read request, Read response start and end segments, Read response continuation segment, Write start and end segments, and Write continuation segment.

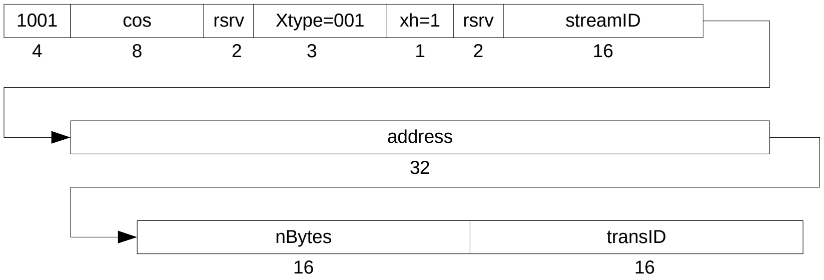

Figure 1-1. RDMA Read Request Type 9 Bit Stream Format

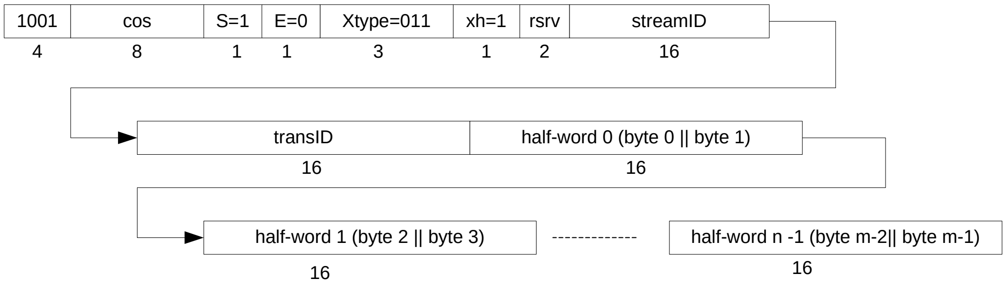
1001
cos
S=1
E=0
Xtype=011
xh=1
rsrv
streamID
4
8
1
1
3
1
2
16
transID
half-word 0 (byte 0 || byte 1)
16
16
half-word 1 (byte 2 || byte 3)
----
half-word n -1 (byte m-2|| byte m-1)
16
16

Figure 1-2. RDMA Read Response Start Segment Type 9 Bit Stream Format

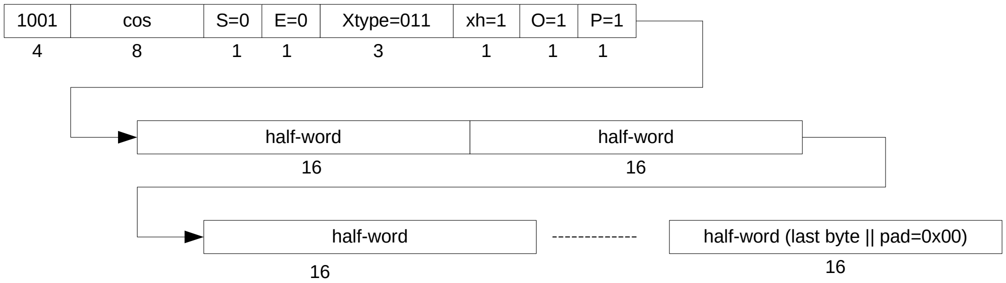

Figure 1-3. RDMA Read Response Continuation Segment Type 9 Bit Stream Format

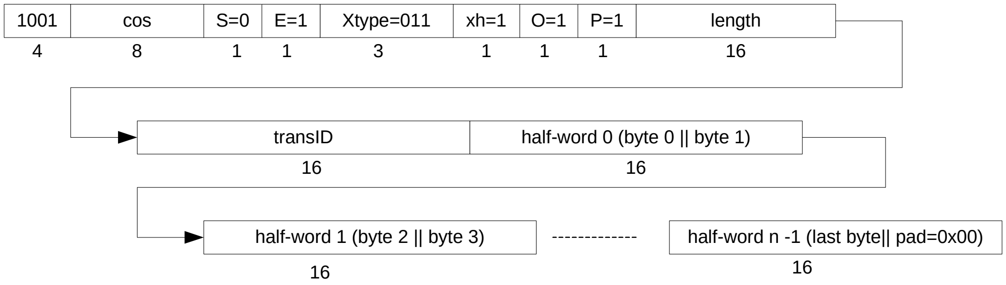
1001
cos
S=0
E=1
Xtype=011
xh=1
O=1
P=1
length
4
8
1
1
3
1
1
1
16
transID
half-word 0 (byte 0 || byte 1)
16
16
half-word 1 (byte 2 || byte 3)
----
half-word n -1 (last byte|| pad=0x00)
16
16

Figure 1-4. RDMA Read Response End Segment Type 9 Bit Stream Format

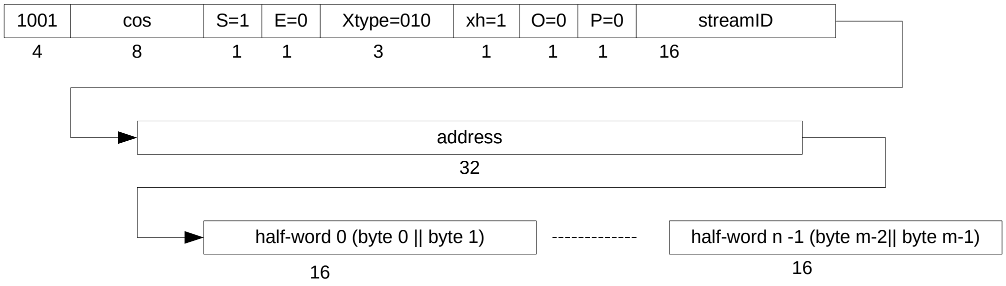

Figure 1-5. RDMA Write Start Segment Type 9 Bit Stream Format

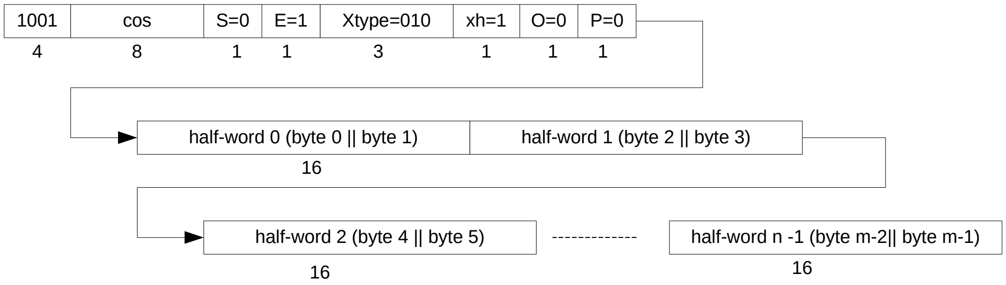
1001
cos
S=0
E=1
Xtype=010
xh=1
O=0
P=0
4
8
1
1
3
1
1
1
half-word 0 (byte 0 || byte 1)
half-word 1 (byte 2 || byte 3)
16
half-word 2 (byte 4 || byte 5)
----
half-word n -1 (byte m-2|| byte m-1)
16
16

Figure 1-6. RDMA Write Continuation Segment Type 9 Bit Stream Format

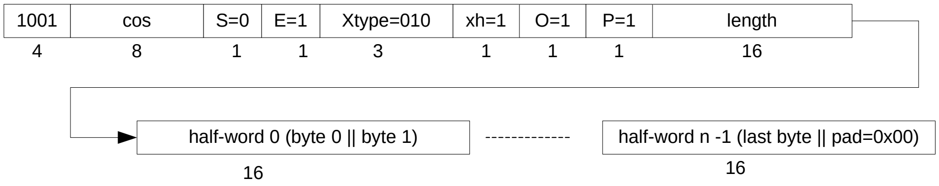
1001
cos
S=0
E=1
Xtype=010
xh=1
O=1
P=1
length
4
8
1
1
3
1
1
1
16
half-word 0 (byte 0 || byte 1)
----
half-word n -1 (last byte || pad=0x00)
16
16

Figure 1-7. RDMA Write End Segment Type 9 Bit Stream Format

The extended Type 9 bit stream fields in the RDMA Read and Write semantics shown above are defined below.

Table 1-1. RDMA Read/Write Extended Header Fields

Type 9 Fields	Encoding	Definition
Xtype	0b001	RDMA Read Request Message.
	0b011	RDMA Read Response Messages. S and E fields are used to denote start, end, and continuation segments.
	0b010	RDMA Write Messages. S and E fields are used to denote start, end, and continuation segments.
	0b100-0x111	Reserved
xh	0b1	Extended Header: There is an extended header on this packet. Currently the extended header is used for stream management and RDMA messages. It is always assigned to 0b1 for type 9 extended packets.
streamID	—	Traffic Stream Identifier: This is an end to end (producer to consumer) traffic stream identifier that is being managed with this message. See Section 4.3.1.
address	—	Address (64-bit): Read or Write Address for RDMA Read or Write Semantics respectively.
nBytes	—	Number of Bytes (16 bit): For RDMA Read operation indicates amount of data to be read.
transID	—	Transaction ID (16-bit): Identifies transaction ID for either Read transactions within a stream.
length	—	Length (16-bit): This is the length in bytes of the segmented PDU. 0x0000 - 64 kbytes 0x0001 - 1 byte 0x0002 - 2 bytes 0x0003 - 3 bytes ... 0xFFFF - 64kbytes - 1

“

1.5.2 Changes to Section 5.5.1 Source Operations CAR (Configuration Space Offset 0x18)

The description of the register fields changes from

“

Table 1-2. Bit Settings for Source Operations CAR

Bit	Field Name	Description
0–11	—	Reserved
12	Data streaming traffic management	PE can support data streaming traffic management
13	Data streaming	PE can support a data streaming operation

Table 1-2. Bit Settings for Source Operations CAR (Continued)

Bit	Field Name	Description
14-15	Implementation defined	Defined by the device implementation
16-29	—	Reserved
30–31	Implementation defined	Defined by the device implementation

“

to

“

Table 1-3. Bit Settings for Source Operations CAR

Bit	Field Name	Description
0–10	—	Reserved
11	Data Streaming RDMA	PE can support RDMA using data streaming
12	Data streaming traffic management	PE can support data streaming traffic management
13	Data streaming	PE can support a data streaming operation
14-15	Implementation defined	Defined by the device implementation
16-29	—	Reserved
30–31	Implementation defined	Defined by the device implementation

“

1.5.3 Changes to Section 5.5.2 Destination Operations CAR (Configuration Space Offset 0x1C)

The description of the register fields changes from

“

Table 1-4. Bit Settings for Destination Operations CAR

Bit	Field Name	Description
0-11	—	Reserved
12	Data streaming traffic management	PE can support data streaming traffic management
13	Data streaming	PE can support a data streaming operation
14-15	Implementation defined	Defined by the device implementation
16-29	—	Reserved
30-31	Implementation defined	Defined by the device implementation

“

to

“

Table 1-5. Bit Settings for Destination Operations CAR

Bit	Field Name	Description
0-10	—	Reserved
11	Data streaming RDMA	PE can support RDMA using data streaming
12	Data streaming traffic management	PE can support data streaming traffic management
13	Data streaming	PE can support a data streaming operation
14-15	Implementation defined	Defined by the device implementation
16-29	—	Reserved
30-31	Implementation defined	Defined by the device implementation

“

1.5.4 Changes to Section 5.6.1 Data Streaming Logical Control CSR (Configuration Space Offset 0x48)

The description of the register fields changes from

“

Table 1-6. Bit Settings for Data Streaming Logical Layer Control CSR

Bit	Field Name	Description
0-3	TM Types Supported (read only)	Bit 0 = 1, Basic Type Supported Bit 1 = 1, Rate Type Supported Bit 2 = 1, Credit Type Supported Bit 3 = Reserved Valid Combinations: 0b1000, 0b1100, 0b1010, 0b1110. All others invalid
4 - 7	TM Mode	Traffic Management Mode of operation 0b0000 = TM Disabled 0b0001 = Basic 0b0010 = Rate 0b0011 = Credit 0b0100 = Credit + Rate 0b0101 - 0b0111 = Reserved 0b1000 - 0b1111 = allowed for user defined modes

Table 1-6. Bit Settings for Data Streaming Logical Layer Control CSR (Continued)

Bit	Field Name	Description
8 - 23		Reserved
24-31	MTU	Maximum Transmission Unit - controls the data payload size for segments of an encapsulated PDU. Only single segment PDUs and end segments are permitted to have a data payload that is less than this value. The MTU can be specified in increments of 4 bytes. Support for the entire range is required. 0b0000_0000 - reserved ... 0b0000_0111 - reserved 0b0000_1000 - 32 byte block size 0b0000_1001 - 36 byte block size 0b0000_1010 - 40 byte block size ... 0b0100_0000 - 256 byte block size 0b0100_0001 - Reserved ... 0b1111_1111 - Reserved All other encodings reserved

“

to

“

Table 1-7. Bit Settings for Data Streaming Logical Layer Control CSR

Bit	Field Name	Description
0-3	TM Types Supported (read only)	Bit 0 = 1, Basic Type Supported Bit 1 = 1, Rate Type Supported Bit 2 = 1, Credit Type Supported Bit 3 = Reserved Valid Combinations: 0b1000, 0b1100, 0b1010, 0b1110. All others invalid
4 - 7	TM Mode	Traffic Management Mode of operation 0b0000 = TM Disabled 0b0001 = Basic 0b0010 = Rate 0b0011 = Credit 0b0100 = Credit + Rate 0b0101 - 0b0111 = Reserved 0b1000 - 0b1111 = allowed for user defined modes
8 - 9	RDMA Types Supported	Following RDMA Operations are supported 0b00 = RDMA Read Supported 0b01 = RDMA Write Supported 0b10 = Reserved 0b11 = Reserved

Table 1-7. Bit Settings for Data Streaming Logical Layer Control CSR (Continued)

Bit	Field Name	Description
10 - 23		Reserved
24-31	MTU	Maximum Transmission Unit - controls the data payload size for segments of an encapsulated PDU. Only single segment PDUs and end segments are permitted to have a data payload that is less than this value. The MTU can be specified in increments of 4 bytes. Support for the entire range is required. 0b0000_0000 - reserved ... 0b0000_0111 - reserved 0b0000_1000 - 32 byte block size 0b0000_1001 - 36 byte block size 0b0000_1010 - 40 byte block size ... 0b0100_0000 - 256 byte block size 0b0100_0001 - Reserved ... 0b1111_1111 - Reserved All other encodings reserved

“