




Crossfield Technology Nuclear Physics SBIR/STTR

Software-Driven Network Architecture for Synchronous Data Acquisition
DE-SC0015151

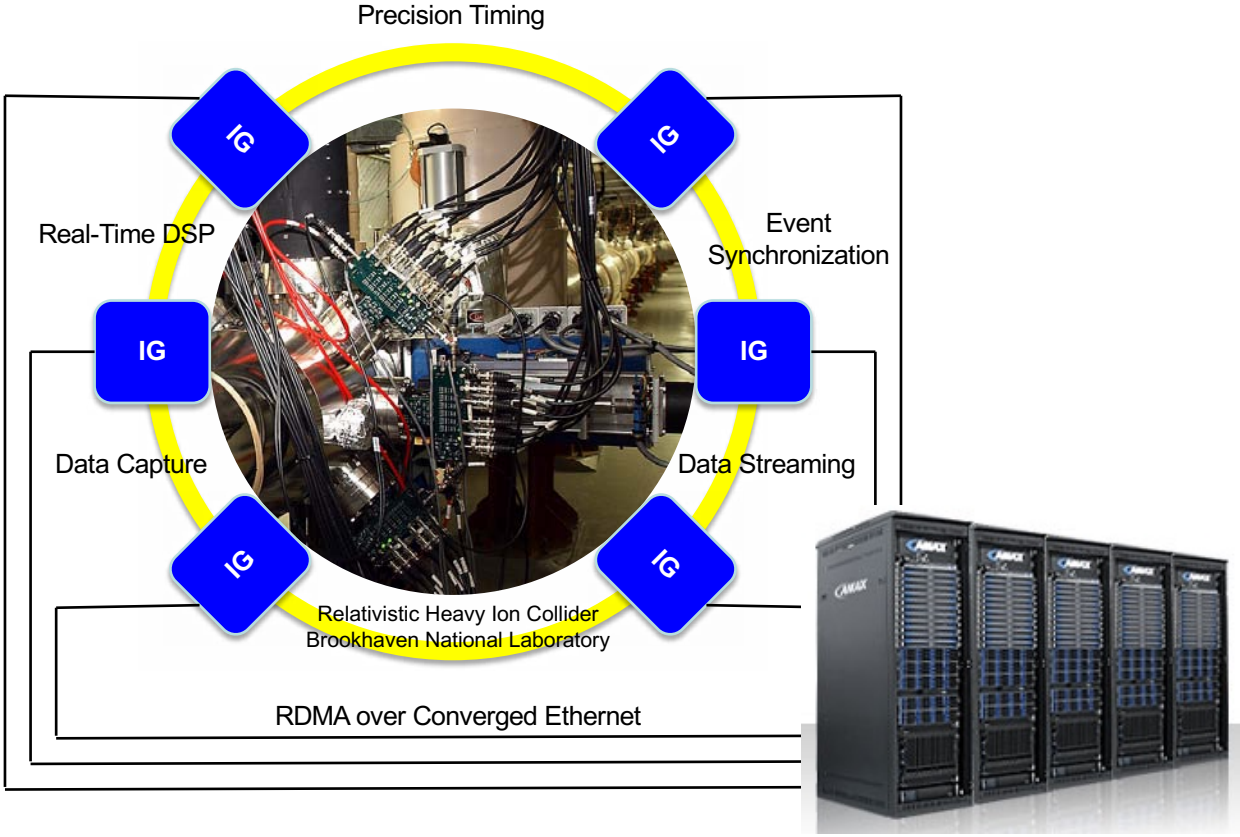
Presentation Outline

- Program Overview
- Company Overview
- Instrumentation Gateways
- RDMA over Converged Ethernet (RoCE)
- FPGA SoC Design
- Status & Results



APPLICATION HIGH ENERGY PHYSICS SYNCHRONOUS DATA ACQUISITION

High Energy Physics Synchronous Data Acquisition



Software-Driven Network for Synchronous Data Acquisition

- DoE SBIR Phase II Program
 - Software-driven instrumentation for event building in high energy physics
 - FPGA-based instruments capture sensor data from multi-GSPS ADCs
 - Sensor data processed in real-time, providing pulse-height and pulse-width analysis, pulse time-of-arrival measurement, and other instrumentation functions traditionally performed in fixed, dedicated instruments
 - Processed event data streamed to HPC for software-based Event Building
- Products Being Developed
 - RDMA over Converged Ethernet (RoCE) IP Core for SoC FPGAs
 - Precision Timing (PTP/SynchE) IP Core for SoC FPGAs

Company Overview

- Started in 2003 as a technology company bringing together a broad range of expertise in high-speed networking, integrated circuit design and wireless system design
- Crossfield has developed multiple generations of Instrumentation Gateways, non-volatile storage, and adapter modules
- Provide software services to capture, analyze and display sensor data streamed from instrumentation gateways

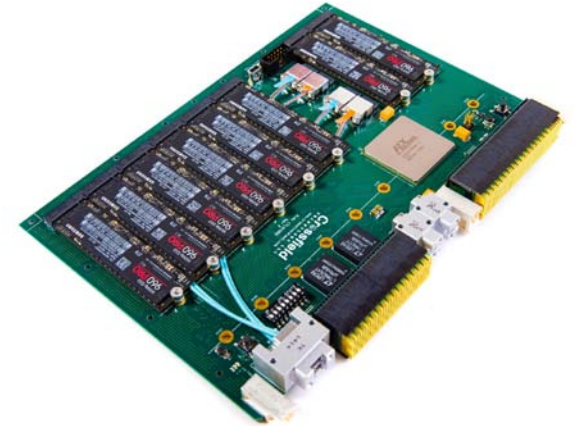
Broad Technology Portfolio



Instrumentation Gateways (IG)



Chassis Management Solutions
SmartFusion2 Initial Focus



Solid-state Storage Modules (SSMs)
CMOSS, HOST & SOSA



OpenVPX IGs
CMOSS, HOST, SOSA

Real-Time Embedded Systems & Software
Chassis Management
High-speed Networking
Synchronous Data Acquisition
Wireless Instrumentation Systems
Modular Open System Architecture (MOSA)



Wireless Gateways

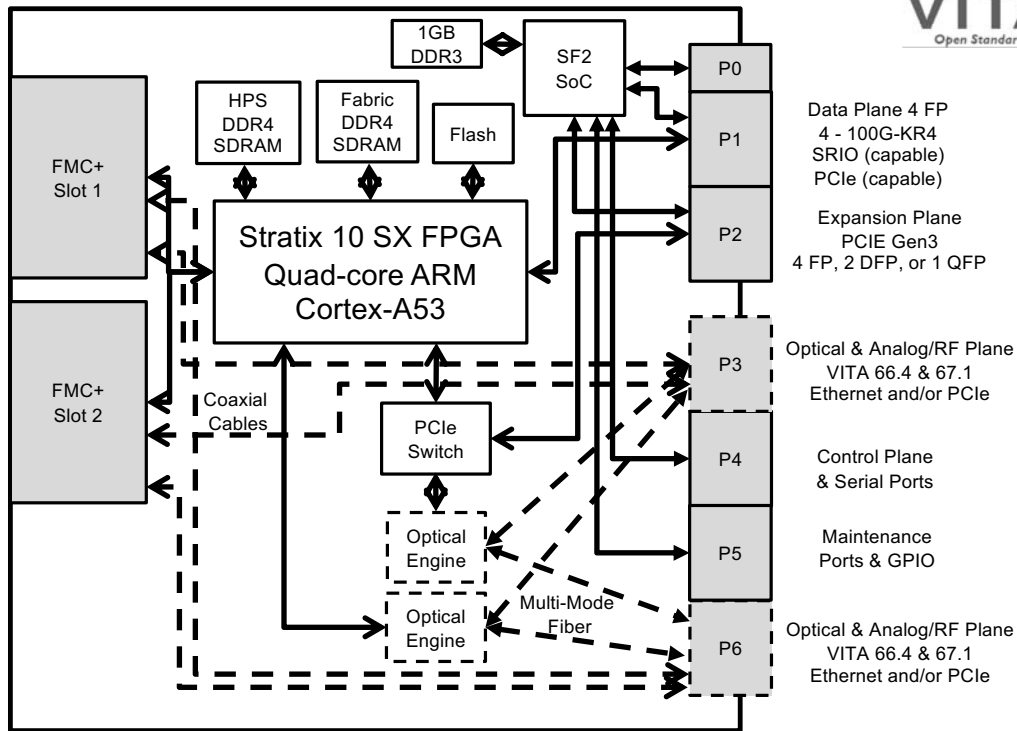


INSTRUMENTATION GATEWAY (IG)

What are Instrumentation Gateways?

- IGs turn simple devices into smart networked edge devices that allows data to be gathered from or distributed to many time-synchronized points
- Key Features:
 - Instrumentation gateways provide advanced networking capabilities, such as Remote Direct Memory Access (RDMA) over InfiniBand or Ethernet
 - IEEE 1588v2 Precision Time Protocol and Synchronous Ethernet provide network time synchronization
- The gateways are provided in desktop, rackmount and OpenVPX form factors

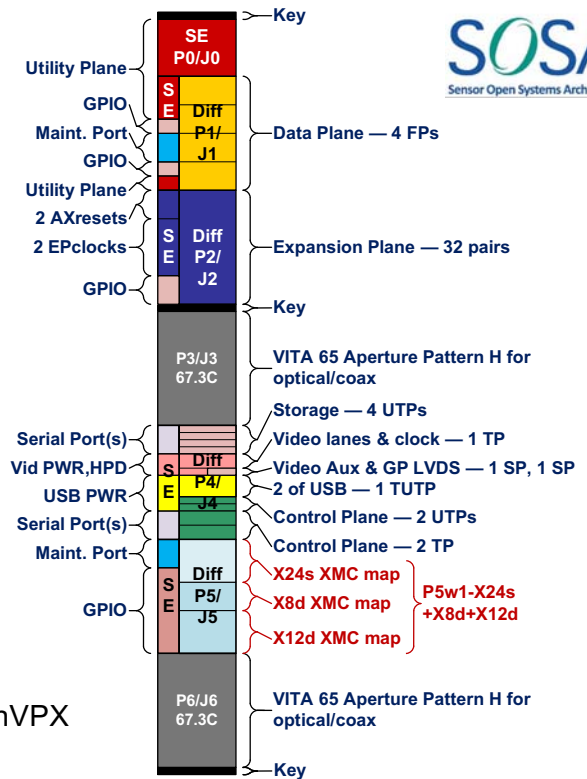
6U OpenVPX Stratix 10 SX Instrumentation Gateway



Dashes denote optional placements



- Data Plane 4 FP
4 - 100G-KR4
SRIO (capable)
PCIe (capable)
- Expansion Plane
PCIe Gen3
4 FP, 2 DFP, or 1 QFP
- Optical & Analog/RF Plane
VITA 66.4 & 67.1
Ethernet and/or PCIe
- Control Plane
& Serial Ports
- Maintenance
Ports & GPIO
- Optical & Analog/RF Plane
VITA 66.4 & 67.1
Ethernet and/or PCIe



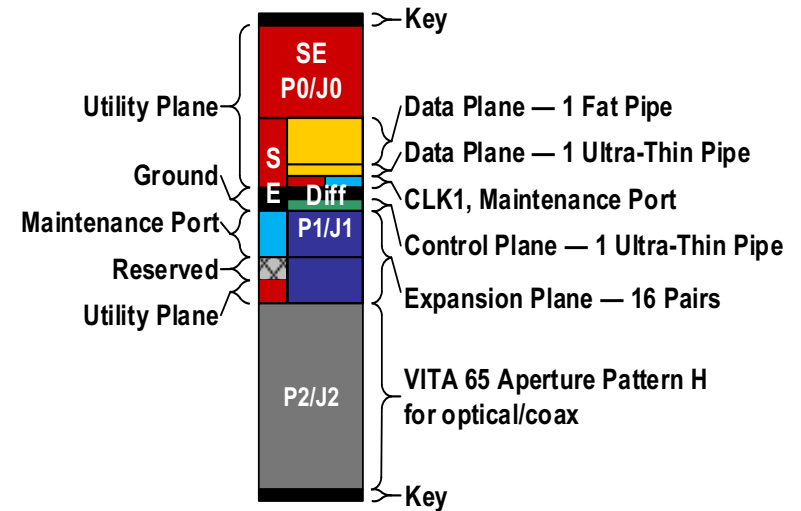
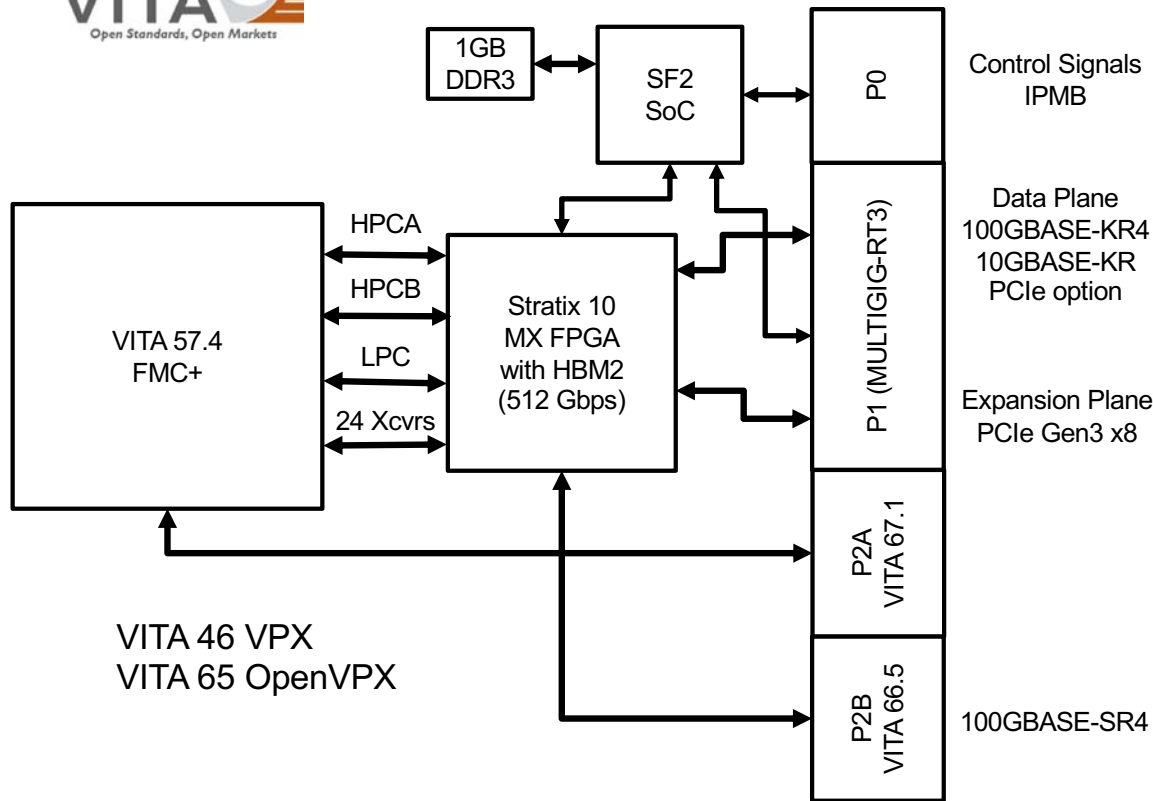
VITA 46 VPX
VITA 65 OpenVPX

OpenVPX Compatible SBC Profile
MOD6-PAY-4F1Q1H4U1T1S1S1TU2U2T1H-12.6.3-n

6U OpenVPX Module with Stratix 10 SX FPGA SoC



3U OpenVPX Stratix 10 MX Instrumentation Gateway



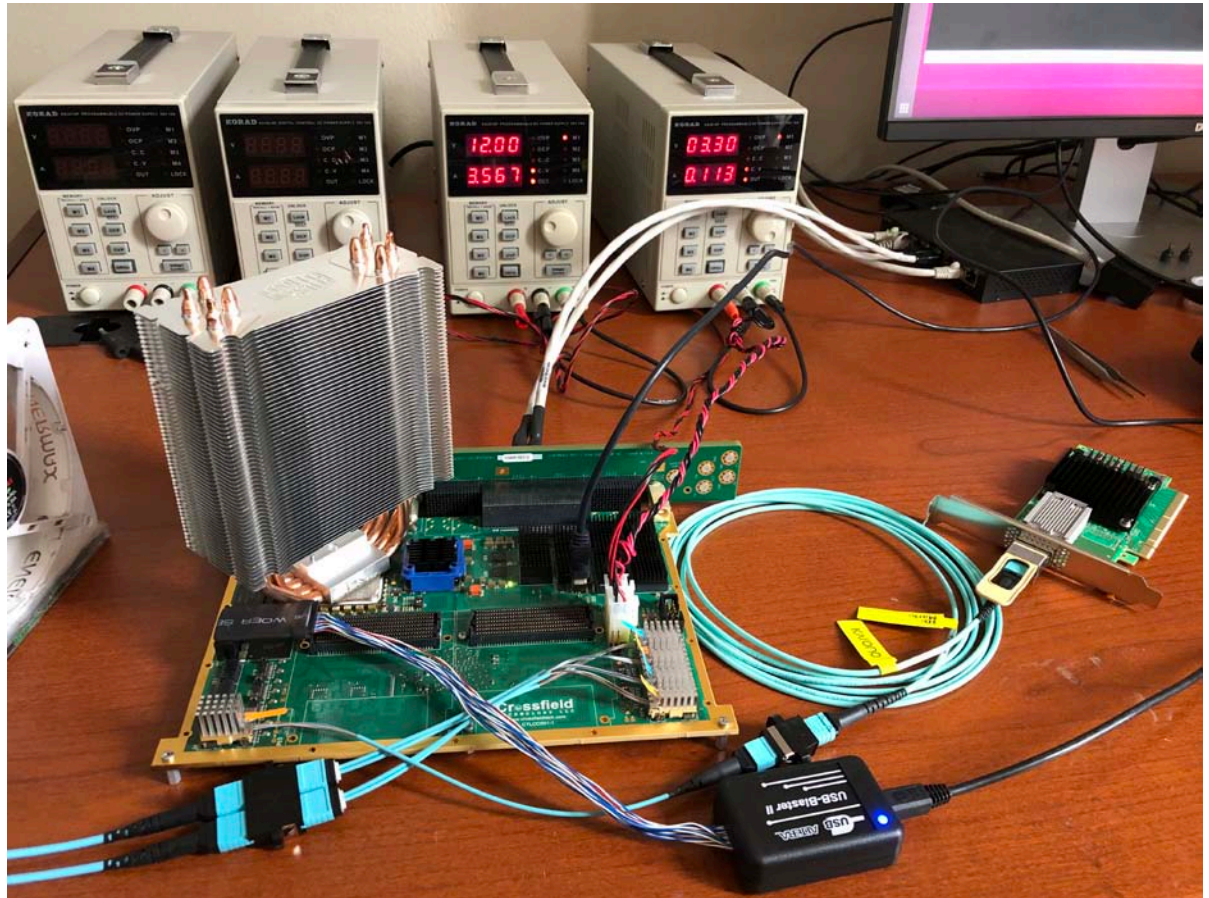
OpenVPX Compatible SBC Profile
MOD3-PAY-1F1U1S1S1U1U2F1H-16.6.11-n

3U OpenVPX Module with Stratix 10 MX FPGA



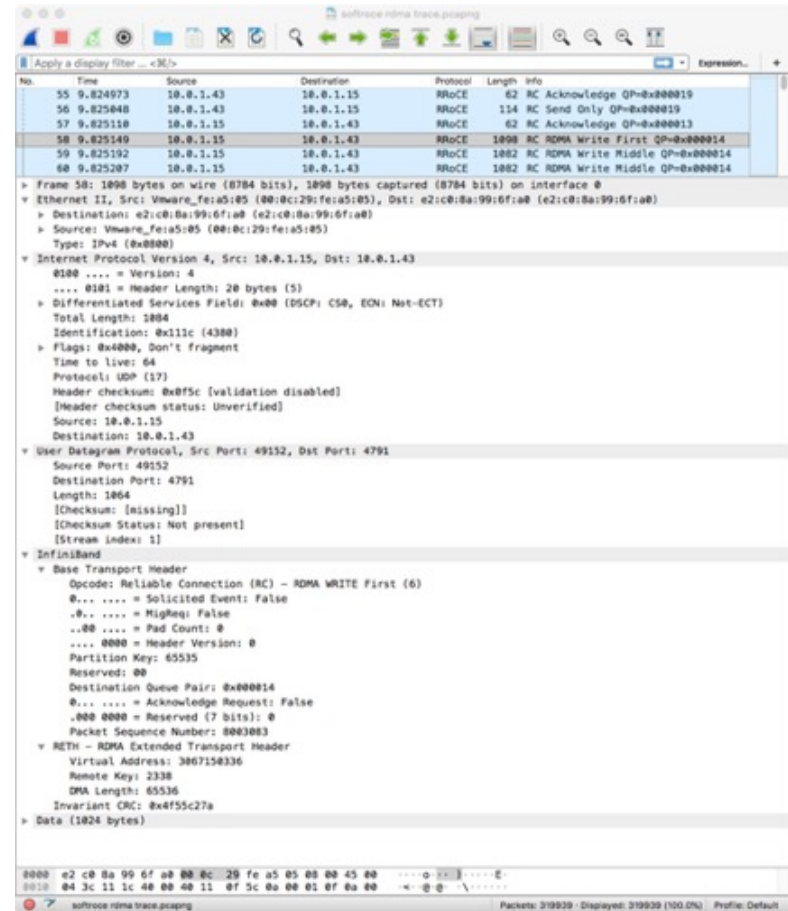
Development Hardware

- 6U OpenVPX IG with a Stratix 10 SX 2800 FPGA
- Linux server with Mellanox ConnectX-5 network adapter and SR4 QSFP28 optical transceiver
- 3-m 12-fiber MPO/MTP cable between systems
- USB-JTAG adapter used to upload programming bitstreams to Stratix 10 FPGA



Wireshark Traffic Analysis

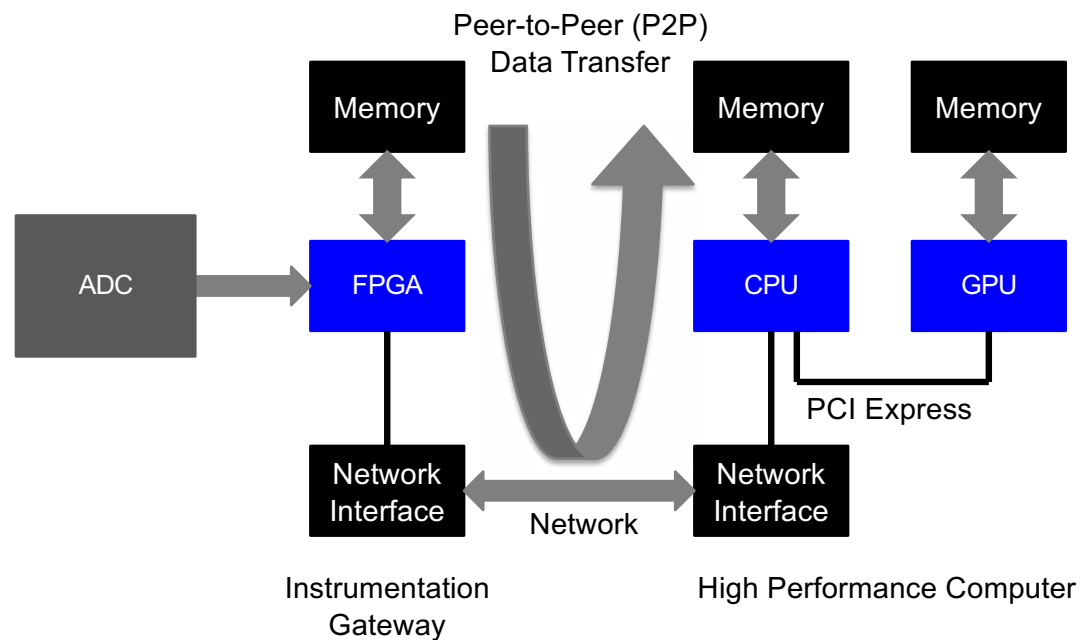
- Wireshark is used to capture and analyze traffic over Ethernet link between the Linux server and the Stratix 10 SX FPGA
- The traffic analysis shows the contents of the packet header and payload
- WireShark capture of SoftRoCE RDMA WRITE with Immediate data transfer over 1GE shown at right



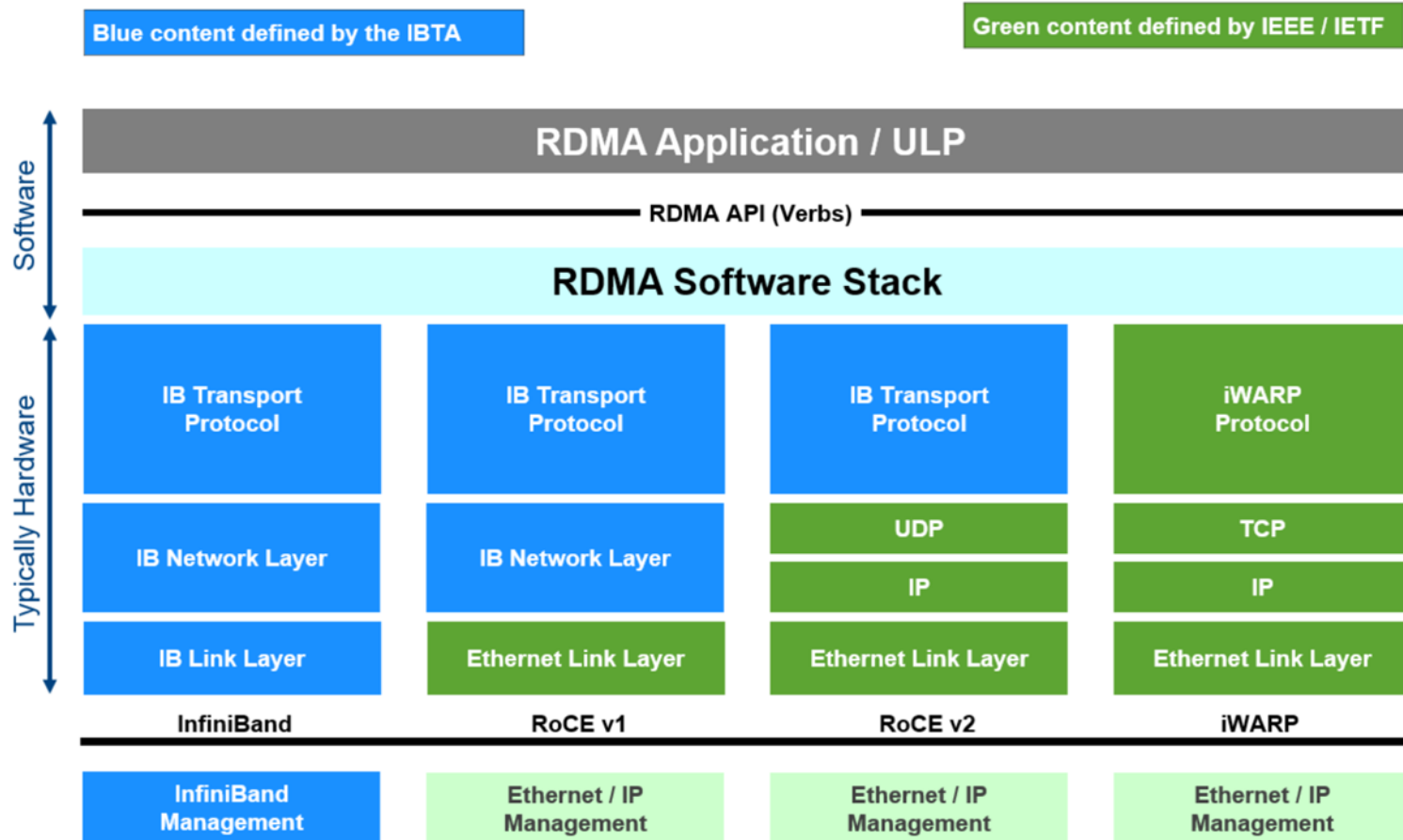


REMOTE DIRECT MEMORY ACCESS OVER CONVERGED ETHERNET

Peer-to-Peer Data Transfer (IG to HPC)

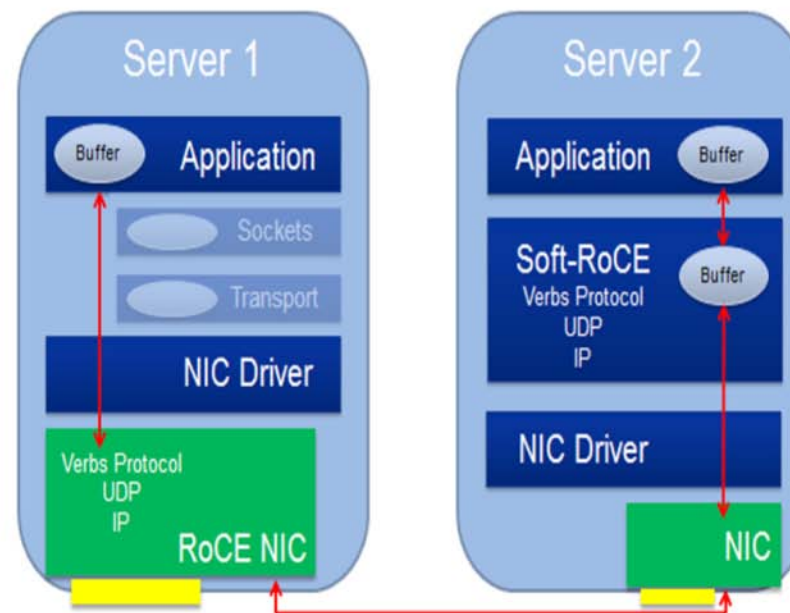


RDMA Networking Choices



Soft RoCE

- Full software implementation of RoCEv2
- Support in Linux readily available
 - On Github --
<https://github.com/SoftRoCE>
- Developed & Verified for x86
 - Main contributors
 - IBM
 - Mellanox
 - System Fabrics Works



Soft-RoCE implements the packet processing otherwise managed by the RoCE NIC.

http://www.roceinitiative.org/wp-content/uploads/2016/11/SoftRoCE_Paper_FINAL.pdf

Soft RoCE Acceleration Provides Solution

- Soft RoCE port to Hard Processor System (HPS) in Intel Stratix 10 SX
 - Board support package for real-time embedded Linux
 - Network Interface Controller (NIC) driver supporting DMA engines and MAC/PHY
- Quad-core ARM Cortex-A53 cannot saturate 100GE link without hardware acceleration
- Initial focus is on RDMA WRITE acceleration capable of streaming raw multi-GSPS ADC data to HPC
 - Retain session and transaction initiation in software
 - Perform ADC data capture and packet encapsulation in FPGA fabric

RoCE Version 2 Packet Encapsulation

Ethernet Header

Ethernet Destination Address (6 bytes)
 Ethernet Source Address (6 bytes)
 Ethernet Payload Type: IPv4 (0x0800)

UDP Header (8 bytes)

UDP Source Port (2 bytes)
 UDP Destination Port (2 bytes)
 UDP Payload Length (2 bytes) (1064 First, 1048 Middle & Last)
 UDP Checksum (0x0000 = not used)

ICRC

Computed for each packet or = 0x0000

FCS

Computed by MAC



IP Header (20 bytes)

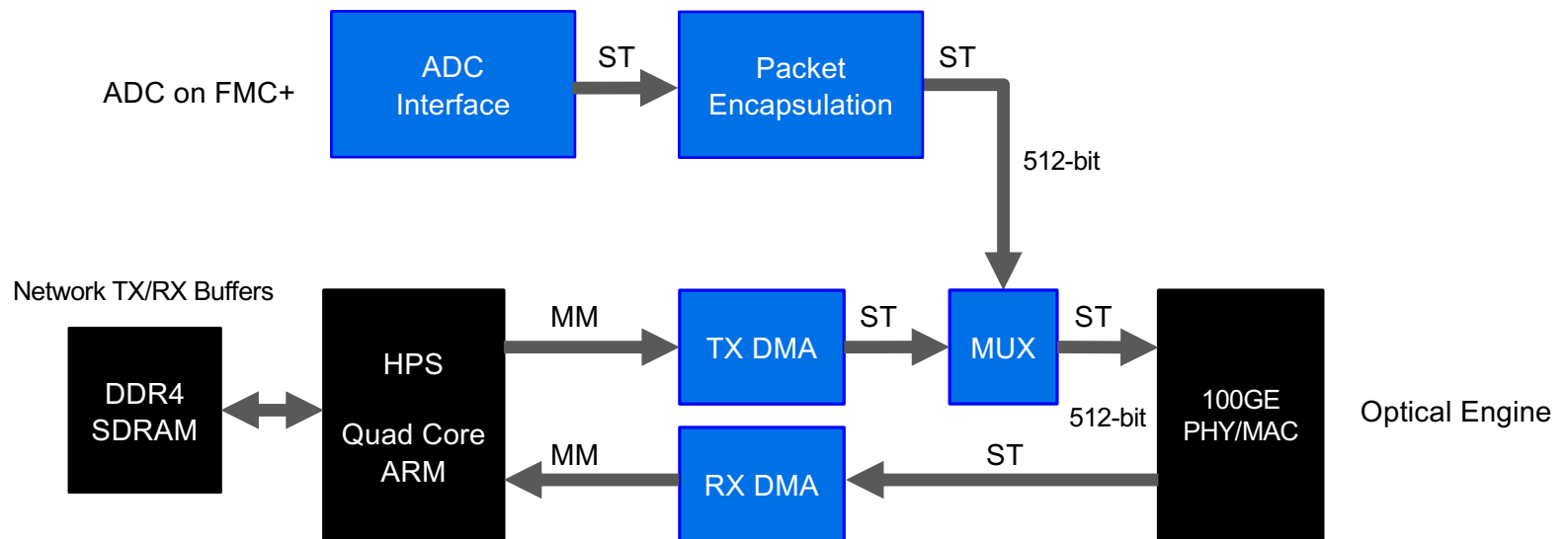
IPv4 Version (4)
 IPv4 Header Length (5) (= 20 bytes)
 IPv4 Packet Length (2 bytes) (1084 First, 1064 Middle & Last)
 IPv4 Identification (2 bytes) (starting ID, increment per packet)
 IPv4 Flags & Fragment Offset (0x4000) (don't fragment)
 IPv4 Time to Live (1 bytes) (64)
 IPv4 Protocol (1 byte) (17 = UDP)
 IPv4 Source Address (4 bytes)
 IPv4 Destination Address (4 bytes)

IB BTH+ Header (12 or 28 bytes)

IB BTH Opcode (1 byte) (06 First, 07 Middle, 08 Last = RC RDMA WRITE)
 IB BTH Partition Key (2 bytes)
 IB BTH Destination Queue Pair (3 bytes)
 IB BTH Acknowledge Request (0 = False First & Middle, 1 = True Last)
 IB BTH Packet Sequence Number (3 bytes) (PSN, increment per packet)
 IB RETH Virtual Address (8 bytes) First packet only
 IB RETH Remote Key (4 bytes) First packet only
 IB RETH DMA Length (4 bytes) First packet only

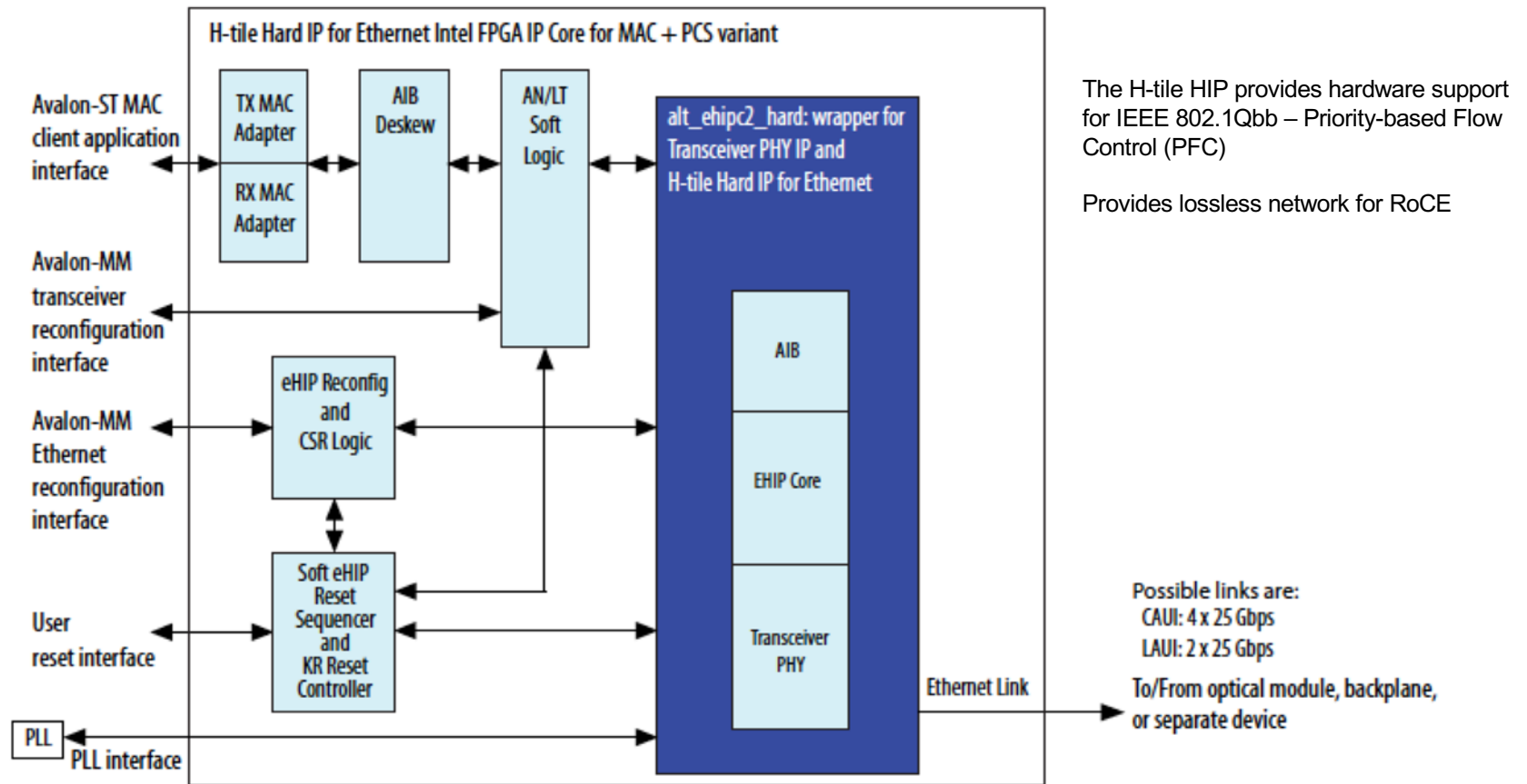
Stratix 10 SX FPGA RoCE Accelerator Datapath

HPS – Hard Processor System
DMA – Direct Memory Access
MM – Memory Mapped Interface
ST – Stream Interface



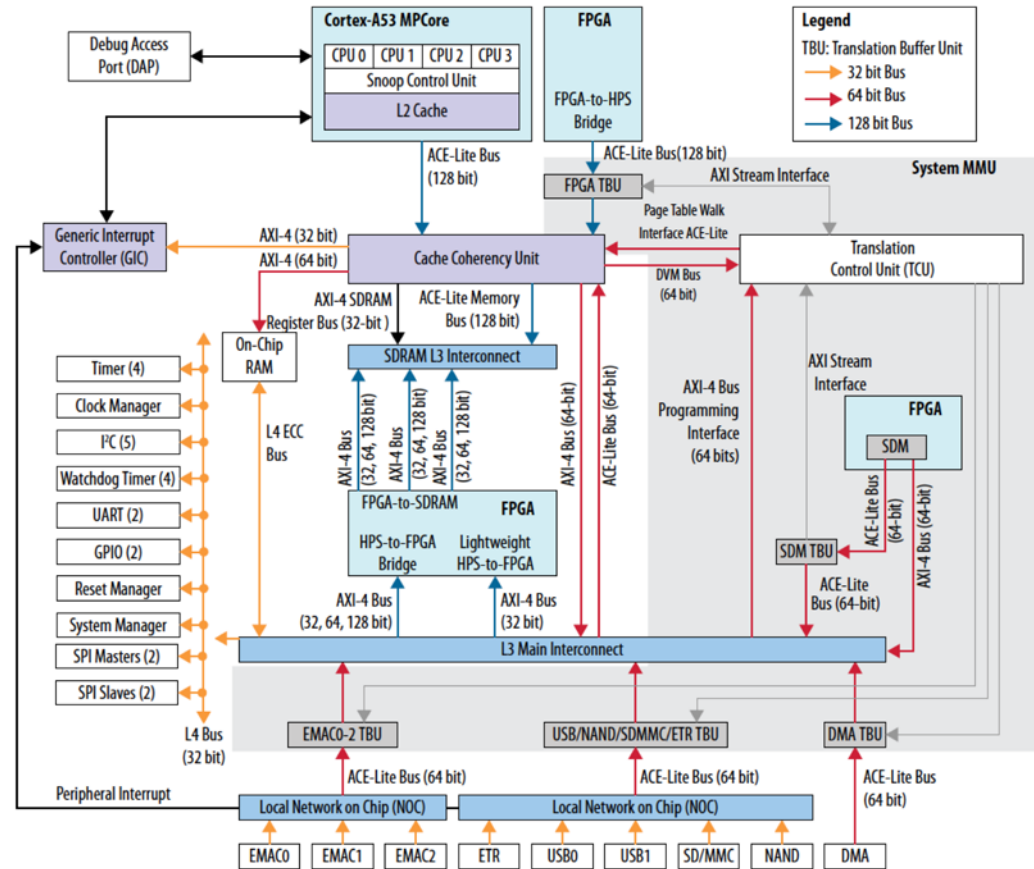
RDMA WRITE first/middle/last packets multiplexed with traffic from network TX/RX buffers in HPS Memory

H-tile Hard IP for Ethernet MAC+PCS IP Core



Hard Processor System (HPS) – Quad-core ARM

- Quad-core 64-bit ARM processor
- HPS-FPGA memory-mapped interfaces provide the major communication channels between the FPGA fabric and the HPS
 - 128-bit FPGA-to-HPS bridge allows the FPGA fabric to have full visibility into the HPS address space – primary data interface used for TX & RX DMA transfers to HPS memory
 - 32-bit Lightweight HPS-to-FPGA bridge used for descriptors and control and status register accesses



Crossfield's Linux eHIP and DMA Device Driver

- Linux device drivers in FPGA SoCs utilize address mappings described in the device tree source to remap hardware addresses into IO memory (iomem) regions
- Crossfield developed a “device tree binding” that mapped the bridge addresses into the kernel and assigned different address regions to named address spaces
- The eHIP device driver implements a platform device, which in turn creates a network device (netdev), phylink, and DMA device
- During hardware probing memory regions defined in the device tree are mapped to iomem regions in the driver for use by the Linux network subsystem via the netdev structure
- When the network subsystem enables the 100G Ethernet interface the netdev is opened and both RX and TX ring buffers are created, RX DMA regions are pre-allocated, and DMA interrupts are enabled. The driver then fills the RX DMA descriptor FIFO with descriptors containing the DMA write addresses of the RX ring buffer.

Design Implementation Strategy – IP Cores

- Baseline design strategy was to use the 100G Ethernet Hard IP Core (eHIP) in combination with Intel soft IP cores for the base RoCE implementation
 - eHIP, modular Scatter-Gather Direct Memory Access (mSGDMA), others
- The eHIP performed as expected once it was provided a stable reference clock, brought out of reset in the proper sequence, and configured correctly
- The mSGDMA was problematic, however, unable to achieve the required 402 MHz clock rate required for the eHIP interface.
- With Intel's assistance it was determined that the IP implementation did not support the new Hyperflex Architecture in Stratix 10, and attempts to update the design eventually proved futile
- An alternative approach was required...

Design Implementation Strategy - HLS

- Crossfield went down a new path using High-Level Synthesis (HLS) and reimplemented the TX and RX DMA engines using the HLS design methodology
- Following the recommended coding styles, Crossfield's designs are now maximally pipelined and fully support re-timing using Hyper-Registers.
- The new HLS designs meet timing with significant margin (~480 MHz)
- Reimplementing the DMA engines in HLS required modification of the embedded Linux device driver
- The HLS-based softRoCE accelerator captures sensor data (e.g. multi-GSPS ADC output), prepends the packet headers shown previously, and the packets are multiplexed into the transmit data stream

High-Level Synthesis (HLS) Design

- HLS designs are implemented in the C++ programming language
 - Components implement the required functionality
 - The main() program implements the testbench
- The testbench and components can be compiled with GCC 9.1.0 and simulated in Mentor ModelSim for design verification
- After design verification, the components are synthesized in Quartus Prime Pro (QPP) along with other HDL designs and IP cores
- For each component, the HLS Compiler creates a corresponding RTL module with top-level ports or interfaces
- You can also explicitly declare Avalon Streaming interfaces (using stream_in and stream_out classes) and Avalon Memory-Mapped interfaces

HLS Code Snippet

- Memory-mapped and stream data is transferred using read() and write() classes
- HLS has built-in support for streaming packets
 - Start-of-packet signal
 - End-of-packet signal
 - Empty bytes in last ST word

```
for (int j=0; j*ST_WIDTH<length; j++) {
    tx_buffer = data_in.read();

    if (j==0)
        start_of_packet = true;
    else
        start_of_packet = false;

    length_counter += ST_WIDTH;

    if (length_counter >= length) {
        end_of_packet = true;
        empty = length_counter - length;
        completions_out.write(1);
    }
    packet_out.write(tx_buffer, start_of_packet, end_of_packet, empty);
}
```

Crossfield HLS Components

- TX DMA Engine Components
 - Avalon MM-to-ST Read
 - Avalon ST FIFO Data Buffer
 - TX Descriptor Dispatcher
 - TX Descriptor FIFO Buffer
 - TX Completion Queue
- RX DMA Engine Components
 - Avalon ST-to-MM Write with FIFO Data Buffer
 - RX Descriptor Dispatcher
 - RX Descriptor FIFO Buffer
 - RX Response Queue
- RoCE RDMA Write Component

Quartus Prime Pro Platform Designer

- The FPGA fabric design is in Intel's Quartus Prime Pro Platform Designer (PD)
- PD is used to interconnect the IP building blocks
 - Hard Processor System (HPS)
 - Ethernet Hard IP (eHIP)
 - TX DMA Engine (HLS)
 - RX DMA Engine (HLS)
 - RoCE Accelerator (HLS)
 - Clock Domain Adapters
 - Clocks (PLLs) & Resets



Status & Results

- Hardware Operational
- Linux Board Support Package
 - Real-time Kernel distro in Yocto
 - SoftRoCE
 - 100GE Device Driver
- Fabric Design Functional
 - eHIP & other IP cores
 - RX DMA Engine
 - TX DMA Engine
 - RoCE RDMA WRITE Accelerator in-progress
- Linux Networking Functional
 - ARP, IP, TCP/UDP, Ping
 - softRoCE

```
root@cc001-0-100ghip:~# rping -c -a 192.168.100.11 -C 3 -v -d
created cm_id 0xaaaaef078c60
cma_event type RDMA_CM_EVENT_ADDR_RESOLVED cma_id 0xaaaaef078c60 (parent)
cma_event type RDMA_CM_EVENT_ROUTE_RESOLVED cma_id 0xaaaaef078c60 (parent)
rdma_resolve_addr - rdma_resolve_route successful
created pd 0xaaaaef07a5f0
created channel 0xaaaaef07a630
created cq 0xaaaaef078fe0
created qp 0xaaaaef07b760
rping_setup_buffers called on cb 0xaaaaef0705f0
allocated & registered buffers...
cq_thread started.
cma_event type RDMA_CM_EVENT_ESTABLISHED cma_id 0xaaaaef078c60 (parent)
ESTABLISHED
rmda_connect successful
RDMA addr aaaaef070fc0 rkey 522 len 64
send completion
recv completion
RDMA addr aaaaef071010 rkey 411 len 64
send completion
recv completion
ping data: rdma-ping-0: ABCDEFGHIJKLMNOPQRSTUVWXYZ[ ]^_`abcdefghijklmnopqr
RDMA addr aaaaef070fc0 rkey 522 len 64
send completion
recv completion
RDMA addr aaaaef071010 rkey 411 len 64
send completion
recv completion
ping data: rdma-ping-1: BCDEFGHIJKLMNOPQRSTUVWXYZ[ ]^_`abcdefghijklmnopqrs
.....

cma_event type RDMA_CM_EVENT_DISCONNECTED cma_id 0xaaaaef078c60 (parent)
client DISCONNECT EVENT...
rping_free_buffers called on cb 0xaaaaef0705f0
destroy cm_id 0xaaaaef078c60
root@cc001-0-100ghip:~#
```


Contact Information

Thank You!

Gary McMillian, Ph.D.

gary.mcmillian@crossfieldtech.com

512-795-0220 x151

Brett McMillan

brett.mcmillian@crossfieldtech.com

512-795-0220 x153

Terry Hulett

terry.Hulett@crossfieldtech.com

512-413-5413