Simplifying FPGA Design with A Novel Network-on-Chip Architecture

John Malsbury
Ettus Research
1043 N Shoreline Blvd
Suite 100
+1 (650) 967-2870
john.malsbury@ettus.com

Matt Ettus
Ettus Research
1043 N Shoreline Blvd
Suite 100
+1 (650) 967-2870
matt@ettus.com

ABSTRACT

As wireless communications continue to evolve, complex new standards force researchers to adopt heterogeneous design approaches that include general purpose processors (GPP) and field-programmable gate arrays (FPGA). This combination leads to increased processing throughput, decreased latency, and increased development complexity. Compared to GPP implementations, custom FPGA designs are time-consuming. The Third-Generation Ettus Research USRP™ (Universal Software Radio Peripheral) has been developed with a unique processing and routing architecture based on VITA-49, which can drastically reduce FPGA development time. This architecture allows users to easily integrate custom IP, such as channelizes, modulators, demodulators, processors or protocol stacks. The architecture will also permit scalable designs that can distribute processing across many nodes. This paper will examine this architecture and how it will reduce development time for researchers. A practical example will also be provided for reference.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design]:** Wireless Communication

General Terms

Algorithms, Performance, Design, Experimentation,

Keywords

Software-defined radio, FPGA, heterogeneous systems, VITA-49, VRLP, SDR, GNU Radio, high-performance computing

1. Introduction

The Ettus Research USRP™ (Universal Software Radio Peripheral) has seen widespread adoption as a wireless prototyping platform. It has been used in a large number of research areas with resurgent interest, such as:

- High-Bandwidth Cognitive Systems'
- New standards such as 802.11ac and LTE-A
- Massive MIMO Systems

While the USRP is often paired with GNU Radio and a general-purpose processor (GPP) for research, these new applications and standards are driving requirements for higher throughput, lower latency processing. In many cases, a pure GPP implementation does not scale gracefully to growing processing demands.

Developers addressing the latest challenges in wireless research must adopt a heterogeneous design approach. DSP and networking functions must be allocated to Field-programmable gate arrays (FPGA), Graphics Processing Units (GPU), and GPPs in an efficient manner. This presents a variety of challenges.

One of those challenges relates to the transport of data throughout the system – internal and external to the FPGA. An FPGA gives infinite latitude for bus and interface design. The designer can choose to implement data transport mechanisms how he/she chooses. This flexibility often leads to re-invention, and results in solutions that are not compatible.

Radio transport protocols (RTP) may offer a way to address this challenge. Developing a standard and consistent way to transport data throughout complex heterogeneous systems will maximize design re-use and reduce development time.

2. Overview of Radio Transport Protocols

When defining a radio transport protocol, it is worth investigating existing protocols, such as VITA-49.

2.1 Introduction to VITA-49

The VITA-49 Radio Transport (VRT) standard was established to provide a consistent way to format sampled IF or I/Q data in software-defined radio systems. Prior to VRT, every vendor developed proprietary protocols to transport samples across the processing elements of an SDR. The VITA-49 frame format can be seen in Figure 1. Vendors can choose to conform to a basic frame, which includes a single header word and the payload, or elect to include information that expresses the type and use of the payload data. Optional timestamps also provide a mechanism for precise temporal alignment within a processing system.

SRIF'13, August 12, 2013, Hong Kong, China.

ACM 978-1-4503-2181-5/13/08…$15.00.

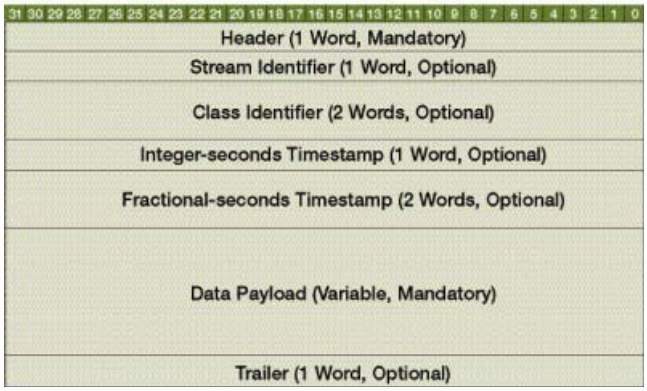

Figure 1 - VITA-49 Frame Format

2.2 VITA-49 Deployment With Second-Generation USRP Devices

VITA-49 was chosen as the underlying transport protocol for UHD-enabled devices. In this case, VITA-49 is used to transport samples and control messages to and from the host-computer via USB 2.0, Ethernet, or external memory interfaces.

In the case of the USRP N-series, the samples are transported over a Gigabit Ethernet (GigE) interface. The VITA-49 frames are encapsulated with UDP. This combination of VITA-49 and UDP facilitates distributed processing design. When initializing the device, a user can specify an arbitrary streaming destination (IP address). A processing device, such as a GPU or FPGA would be assigned to that address, receive the samples, and pass them along for further processing by other elements. An example of such a use case is shown in Figure 2.

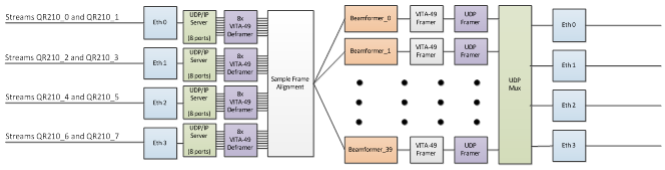

Figure 2 - External FPGA Stream Processing with VITA-49

This reference design highlights the capability of a transport protocol. In this case, 32 separate streams are produced by a number of USRP devices. The streams are muxed onto four 10 GigE ports, and the data is fed to the FPGA for processing. A block examines the time-stamp in the VITA-49 header and aligns samples for coherent processing by 40-beamformers.

In this example, VITA-49 provides a way to distribute data to a number of components connected with an interface such as 10 GigE. This allows designers to build modular systems, and provides a consistent 'language' for every device to communicate with. This concept can also facilitate FPGA design.

3. Developing a Third-Generation Architecture

3.1 Goals of a Third-Generation Architecture

Use of the VITA-49 standard in the second-generation SDR permitted many interesting applications. As FPGAs become more capable, and there is a greater need to leverage FPGAs for low-latency, high-bandwidth processing, there are benefits to leveraging a radio transport protocol both internally, and externally.

Internal transport implementation offers several benefits. Just as VITA-49 permitted modular processing solutions, internal transport designs would allow the user to develop and integrate modular IP inside the FPGA, without drastic impacts to other components in the design. Also, conforming to a standard improves design re-use, and reduces development time. For these reasons, third-generation USRP devices will use a radio transport structure that facilitates FPGA integration as well as external, distributed processing.

It is worth establishing some general characteristics of the FPGA routing architecture. The intent is to develop a framework that meets the following requirements:

1. Revolves around the concept of data streams, in the same manner as VITA-49.
2. There will be producers and consumers of streams.
3. Each producer will be able to route a stream of data to an arbitrary consumer within the FPGA.
4. Each consumer will be able to consume a stream from any producer.
5. Routing fabric will use a header that is present at the beginning of a stream-write to route the stream accordingly.

3.2 Modern Bus Architecture – AXI

When designing a radio transport protocol for use within the FPGA, a designer must consider characteristics and features of modern bus architectures that are available.

Modern FPGAs such as the Xilinx 6 or 7 series FPGAs offer clean support for the Advanced eXtensible Interface (AXI). This bus is part of the ARM AMBA standard. The latest definition for this bus is AXI4, of which there are three types:

1. AXI4 – high-speed memory mapped interface
2. AXI4-Lite – simple, low throughput memory-mapped communication
3. AXI4-Stream – high-speed streaming data

The AXI4-Stream was chosen for this implementation, since it is focused on a data-flow paradigm. Compared to AXI4, which is intended for memory-mapped operations, the AXI4-Stream eliminates the address cycle. It can also support data bursts of arbitrary size. In essence, it is an ideal standard for "plumbing" of data within and FPGA because it can provide high-performance data transport with relatively simple logic.

AXI4 and AXI4-Stream can support 32, 64, 128, or 256-bit data widths. While AXI4-Stream does not include an address cycle, the data within streams will include headers, which provide

addressing and other information for data routing. Routing logic must inspect this data for multiplexing. 128-bit and 256-bit addressing logic is more challenging to route in the FPGA while meeting timing constraints. A data width of 64-bits is provides a balanced trade of throughput versus FPGA routing complexity.

Xilinx offers branded cores that conform to the AXI4 standard. There are also a large number of third-party IP vendors that use AXI4. There are IP blocks called "datamovers" that can easily adapt from AXI4 to AXI-Stream for implementations that require memory-mapped interfaces. For example, an FFT block or soft-core processor is more suited for the memory mapped AXI4 interfaces.

3.3 Compressed Header (CHDR)

The compressed header format (CHDR) considers characteristics of existing protocols, as well as the benefits and limitations imposed by a 64-bit AXI4-Stream. Looking at implementations in second-generation USRPs, it is evident that approximately 64-bits are required to meet basic routing requirements. A single, 64-bit header word can express the stream source and destinations, and provide a frame count.

Table 1- Compressed Header - 64-bits

Bits	Function
31-0	Stream ID
47-32	Packet Size (words)
59-33	Sequence Number
60	1 = End-of-Burst
61	1 = Has Time
62	1 = Has Trailer
63	1 = Context?

Expressing this information with single, 64-bit words allows muxes, demuxes, and other elements to easily parse and produce data without complex state machines. This improves throughput, and facilitates timing closure in the FPGA build process.

If necessary, the time can be included. As previously established in second-generation implementations, 64-bits is sufficient for time stamping with sample-clock resolution.

3.4 CHDR Transport Over Media

While CHDR provides a convenient method to standardize internal FPGA interfaces, it also provides a way to design scalable systems. Combining the internal routing capability of CHDR with the external distribution capability of VITA-49 will allow for more straightforward designs and distributed processing.

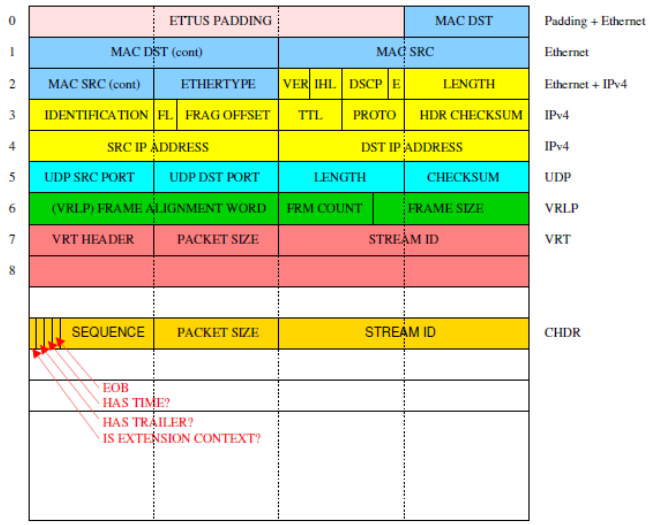

Figure 3 - Encapsulation of CHDR for Transmission over Ethernet

Figure 3 shows the frame formats involved when transmitting data over 10 Gigabit Ethernet. It is worth noting that each of the routing points in this system are configurable. For example, the internal radio transport router can move streams between components internally. Alternatively, the router can pass a stream to a host-interface, which includes a VITA-49 framer and UDP framer. The VITA-49 and UDP framers can be configured to stream to another USRP of a specific IP address. That USRPs transport router will receive the stream, parse the header's Stream ID, and route the stream to another block such as a computational engine or radio block.

Examples of this sort of routing will be shown in latter sections of this paper.

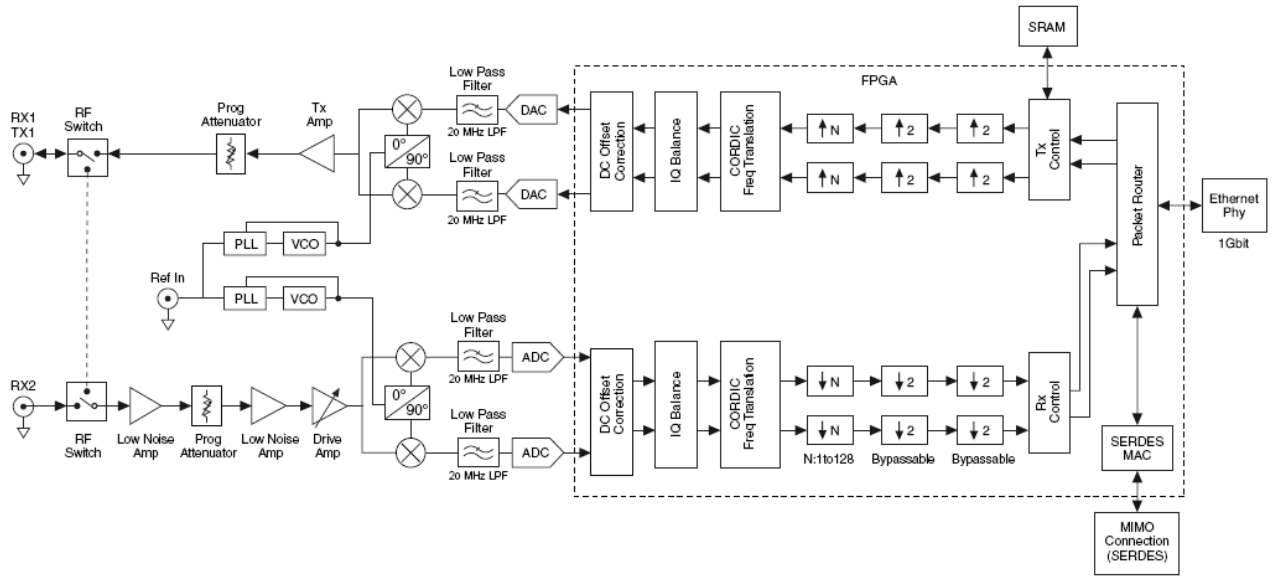

Figure 4 - Second-Generation USRP Architecture

4. Implementation

4.1 FPGA Architecture of Second-Generation USRPs

The second-generation FPGA architecture, which is shown in Figure 4, made the process of integrating custom DSP functions easier. Readme files and makefiles could be readily modified to splice DSP blocks anywhere in the DDC and DUC chains. These are referred to "rx_dsp_chain" and "tx_dsp_chain", respectively.

Considering the receive chain, a researcher can integrate a custom block in several points. A block can interface directly with the ADC frontend and process baseband samples at the full sample rate – 100 MS/s in the case of the USRP N210. A block can also be integrated after the decimation stages, just before the VITA-49 framer, which is labeled as "Rx Control" in the diagram above. The same is basically true in the transmit directions. This architecture facilitate in-line DSP operations.

The VITA-49 framers and de-framers also offer a modest amount of flexibility for operations such as sample size truncation.

4.2 Third-Generation FPGA Architecture

The third-generation USRP architecture leverages radio transport protocols to a greater extent. The transport protocol is used for data routing inside the FPGA, as well as data transport to and from the host computer. The architecture includes several components, and is illustrated in Figure 5.

Any component that is connected to the Radio Transport Router communicates using the CHDR format. The router is able to pass data between any component in a non-blocking fashion. For example, a received sample stream generated by Radio0 can be routed to a computation engine while a stream from Radio1 can be routed directed to the UDP framer and Eth0 interface. The router will support this at the full data rate.

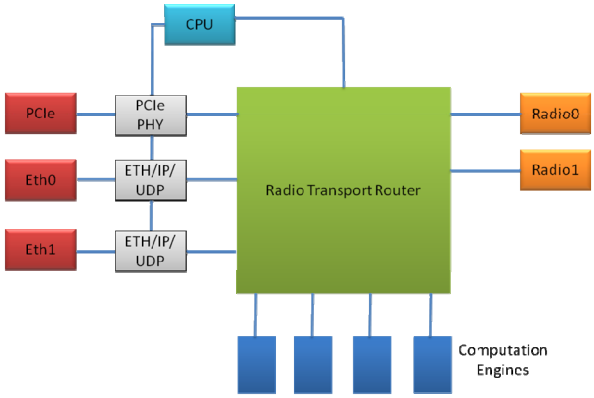

Figure 5 - Radio Transport Protocol Implementation

The radio blocks, "Radio0" and "Radio1" include a DUC chain, DDC chain, and control functions. These blocks provide approximately the same functionality of the rx_dsp and tx_dsp chains of the second-generation architecture. However, unlike the second-generation architecture, these radio modules will be connected to the RTR instead of to the host-interface infrastructure. This allows data to be routed to any other block in the system – computation engines or a host-interface.

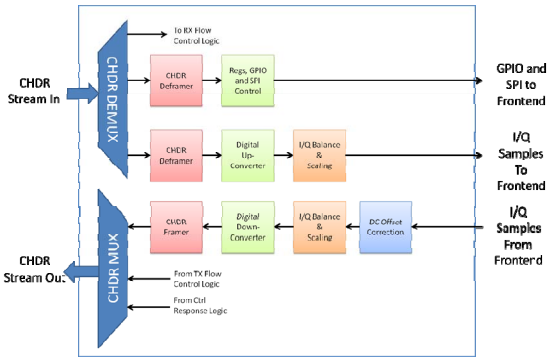

Figure 6 - Radio Block - Rx/Tx DSP Implementation

The blue components shown in Figure 5 are computation engines. These can provide a wide array of functionality - channelization, modulation, demodulation, FFTs. Computation engines can also take on more complicated, frame-based functions. For example, the computation engine might include a soft-core processor and implement an entire protocol stack, or provide intelligence for control plane activities in a cognitive radio.

Of course, the router also connects to key components of the host interface, which is used to exchange data and control-plane information with the host. In this case, the USRP provides several host-interface options – two 1/10 Gigabit Ethernet ports, and PCIe. There are numerous ways these host-interfaces can be connected for large, scalable systems.

A soft-core CPU will provide basic management and autonomous configuration capability, but will not be directly involved with the high-speed processing associated with components connected to the radio transport router. The host will be able to communicate with the CPU directly.

4.3 Router and Stream Configuration

The RTR functions as a non-blocking cross-switch. Figure 5 shows an RTR with 9 ports. The number of ports in the RTR will be configurable at build-time, and will typically be set for 8-ports. For the sake of simple illustration, a cross-switch architecture for a 4-port RTR is shown in Figure 7.

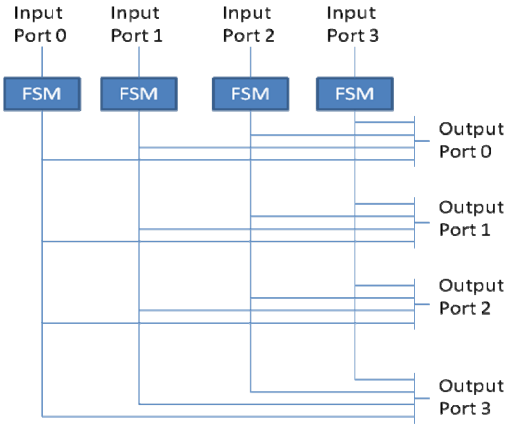

Figure 7 - RTR Cross-Switch

Router logic at the input of each port will examine the first word of a data burst – the CHDR – to determine where it must route the data. This decision will be based on run-time configuration of routing tables and the destination source. In other words, the data source must be configured to assign a stream id, and the router must be configured to direct a data-burst with a given stream id to the appropriate destination. These parameters will be easily settable through the host-driver API.

4.4 Example - Heterogeneous PHY/MAC Implementation

In this architecture, there are several host-interface options. PHY/MAC development will often require low-latency in the system. For this reason, the PCIe is an ideal choice.

To meet the demands of high-bandwidth protocols such as LTE-A or 802.11ac, the researcher will implement a portion of PHY and lower MAC layers in the FPGA. In order to maintain a greater level of flexibility in the upper MAC layers, the researcher will implement this on the host using GNU Radio, LabVIEW, IRIS, or use the UHD C++ API directly.

The flow of information using the RTR is shown in Figure 8. In this case, packet data units(PDUs) are passed from the host-computer to an OFDM modulator. Baseband samples are passed from the OFDM modulator to Radio0. Conversely, received baseband samples generated by Radio0 are passed to the OFDM demodulator. Received PDUs are passed to the host-PC through the PCIe interface. Proper design, and intelligent use of timed-features will ensure proper timing and alignment of transmission operations.

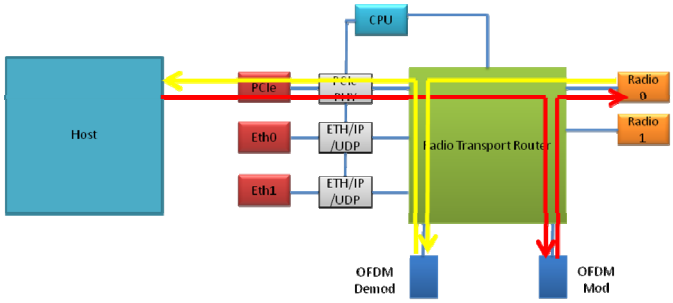

Figure 8 - Heterogeneous PHY/MAC Implementation

This example provides several benefits.

4.5 Example – Distributed Processing

In the example shown in Figure 9, a single host-interface is used, but signal processing functions are split across multiple SDRs. Such a system might be useful for complex spectrum monitoring operations, multi-carrier receivers, or similar applications.

A received sample stream is produced by the first USRP. Before passing along for further processing, the data (red) is routed to an impairments correction block. This block might apply adaptive corrections to minimize I/Q imbalance. Next, the corrected I/Q data (purple) is passed through an Ethernet connection a second device. The second device does not make use any radio frontends. It only provides additional DSP resources. In this device, corrected I/Q data is passed to a channelizer, a multi-channel power detector, and a demodulator/deframer(light purple). The fully demodulated and frame synchronized data is passed to the host through the cascade 10 GigE interface.

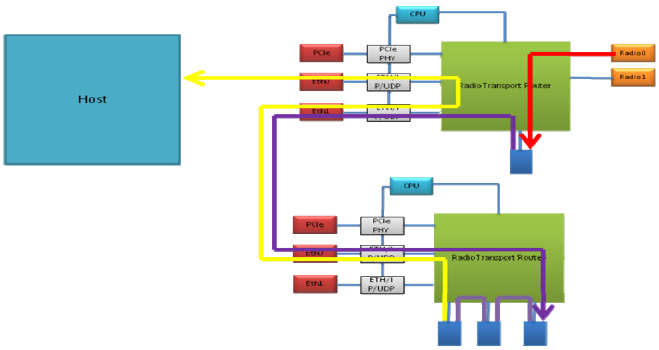

Figure 9 - Scalable System Design with RTR

A system of this architecture can be expanded to include an arbitrary number of processing nodes. Each additional node will use the cascaded Ethernet interfaces of the previous nodes to pass data back to the host.

4.6 Example –Massive MIMO Systems

The distributed processing architecture shown in the previous section be modified slightly to build massive MIMO configuration. This is an area of strong research interest. Multiple USRPs can be time and frequency synchronized. This allows coherent processing for MIMO systems.

In this case shown in Figure 10, each USRP device would provide 1-2 frontends. Processing can be performed on each USRP unit, or distributed throughout the system.

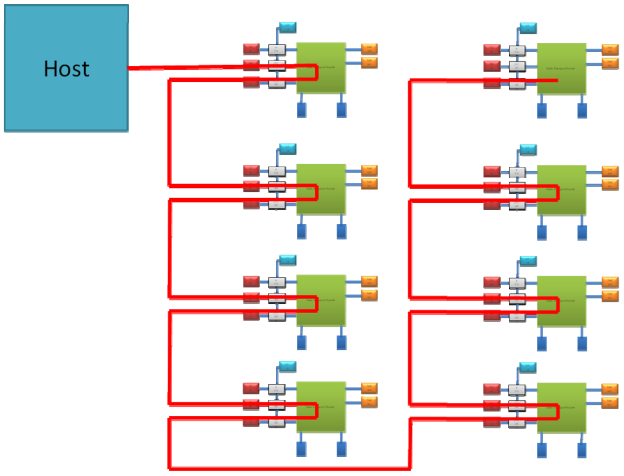

Figure 10 - Massive MIMO 16x16

Figure 10 shows a 16x16 systems. In this case, data is shared across multiple USRPs through a daisy-chained 10 GigE connection. Of course, careful consideration needs to be given to total bandwidth and where various portions of processing will occur. It is also possible to add multiple host-computers A system like this might be used to apply progressive beam-forming on a large array in an incremental fashion.

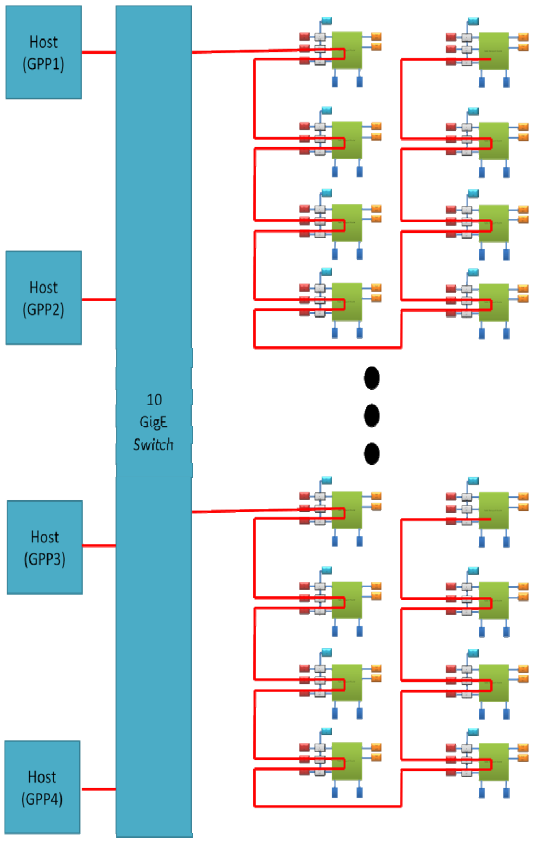

Figure 11 - Massive MIMO System

While the example shown in Figure 10 show a system that relies on cascading of the 10 GigE interface to achieve 16x16 MIMO, it is also possible to connect each device to a 10 GigE switch, and share information across multiple hosts. Or, a hybrid approach can be used. Strands of cascaded USRPs can be connected to a switch in a star configuration. Such system architectures will form the basis of massive-MIMO prototyping systems.

5. Conclusion

Examination or radio transport protocols shows that they canbe used to build complex processing systems, improve design re-use, and potentially decrease development time for heterogeneous systems. The compressed header format and routing architecture discussed in this paper have been integrated into third-generation USRPs, and will allow researchers to meet demands for increased performance in SDRs.

6. ACKNOWLEDGMENTS

We would like to give thanks to all USRP users and the open source SDR community.

7. REFERENCES

[1] Bieberly, F.B., Heterogeneous Processing in Software Defined Radio: Flexible Implementation and Optimal Resource Mapping, February 8, 2012, Blacksburg, Virginia

[2] Fayez, A., Designing a Software Defined Radio to Run on a Heterogenous Process, April 15, 2011, Blacksburg Virginia

[3] Normoyle, R., VITA49 Enhances Capabilities and Interopability for Transporting SDR Data, April 2008, Retrieved on from: http://www.pentek.com/VITA%2049.pdf?Filename=VITA%2049.pdf