VICTOR WOLLESEN, CEO and Co-founder,
and REZA MOHAMMADI, Network Security,
Per Vices Corp., https://pervices.com/

Design Considerations for High-Throughput SDR Systems

The host and radios must be matched for optimum performance in an software-defined-radio environment.

Software -defined radio (SDR) is a radio communication technology that implements conventional hardware components using software. The motivation behind SDR-based systems is reprogrammability and ease of maintenance. This technology increases the lifespan of radio communication infrastructure by allowing new protocols to be supported through software updates. Another advantage of SDRs is reduced development time and cost.

One key performance parameter of SDRs is their high throughput, which is the rate of data that can flow through the system. The high throughput is enabled by the wide-band connections that SDRs can support, as well as the FPGA used to interface with the host system.

High-throughput SDRs find their applications in time-critical applications such as telecommunications, military, and public safety. In telecom applications, a high instantaneous bandwidth means more users and more data can be transferred over the links. In military applications, radar systems require complex signal processing to resolve geographical positions. Analysis of the signals is a time-critical task that must be done concurrently with data collection, hence requiring higher throughput to support both tasks.

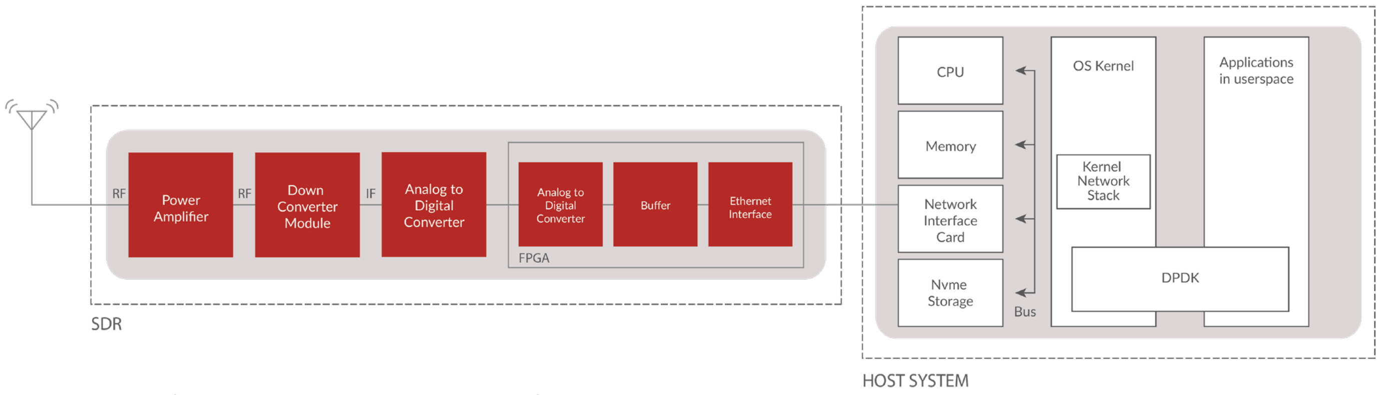
And in public safety applications, interoperability is important for all medical devices in hospitals. The simultaneous data transfer between many devices requires high bandwidth to support this function.

These interesting applications have led to the development of wide-band SDRs. However, effectively streaming large volumes of data on a host system poses several challenges.

SDR System Overview

To better understand these challenges, let's consider a simple scenario in which we aim to capture received transmission data on an SDR system. In *Figure 1*, we see the RX chain of the system, where the SDR receives a weak RF signal from the antenna, which is fed to a power amplifier. Then the signal is shifted to baseband by the downconversion module, and, finally, it goes through the analog-to-digital converter (ADC) to be sent to the FPGA.

In the FPGA, the signal is processed and buffered so that



1. A typical SDR receiver requires a device like an FPGA to handle the digitized data stream.

it can be sent to the host system over Ethernet. The host system then interfaces the SDR with a network interface card (NIC). User-level applications can directly access this data, bypassing the kernel network stack, by utilizing network interface controllers such as the Data Plane Development Kit (DPDK).

The TX chain is shown in *Figure 2*. It uses the same components in reverse order apart from the digital-to-analog converter (DAC) and the upconversion module, which are required to create a transmittable signal.

Because the sending and receiving of data from the Ethernet link is performed by the FPGA, it's nominally guaranteed to be deterministic and synchronous within its clock domain. However, as the data moves between the FPGA-based SDR to the host system, it not only crosses clock domains, but it also becomes non-deterministic. As a result, synchronization is a major requirement in this system, and it's achieved by defining a proper power-on sequence.

Data-Flow Overview and Challenges

Another important consideration to make is the data throughput of the host system. *Figure 3* shows an example of the various elements and interfaces required to stream data to or from a storage device. Particular care needs to be taken to ensure sufficient PCI Express (PCIe) bandwidth between storage devices, CPUs, and NICs. For host machines with multiple CPUs, this often requires reading the motherboard manual to determine the distribution of PCIe root complexes and ensuring that they're correctly distributed to the various processors.

When considering the components presented in *Figure 3*, several challenges arise in terms of selecting components that enable the creation of a high-throughput SDR system. The selection of the FPGA and SDR platform is crucial because it handles the entire analog portion of the RX and TX chain, and is the limiting factor in how much data can be streamed between the host machine and CPU. For the host system, the CPU and memory, the storage devices, the NIC, and the software interface are crucial to handle the data being sent by the SDR.

SDR and FPGA Selection

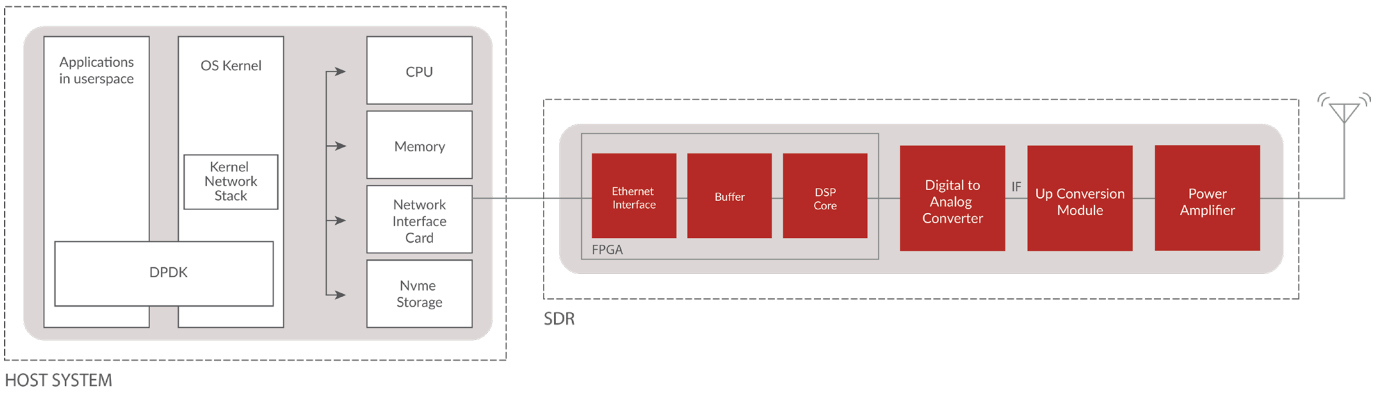
The SDR platform use is crucial in the implementation of a high throughput SDR system. Not only is the SDR entirely responsible for the analog portion of the system, but also the signal processing onboard the FPGA. In the examples shown in Figures 1 and 2, the processing architectures that are common to the Per Vices Cyan and Crimson Software Defined Radios are examined. The Per Vices Crimson SDR supports up to 20Gbps throughput, while the Cyan SDR supports between 160-400Gbps of throughput.

The inclusion of high speed, high resolution ADCs and DACs is critical to maximizing the RF bandwidth associated with the data being transferred between the SDR and host machine. In particular, high-throughput applications greatly benefit from high-sample-rate converter devices.

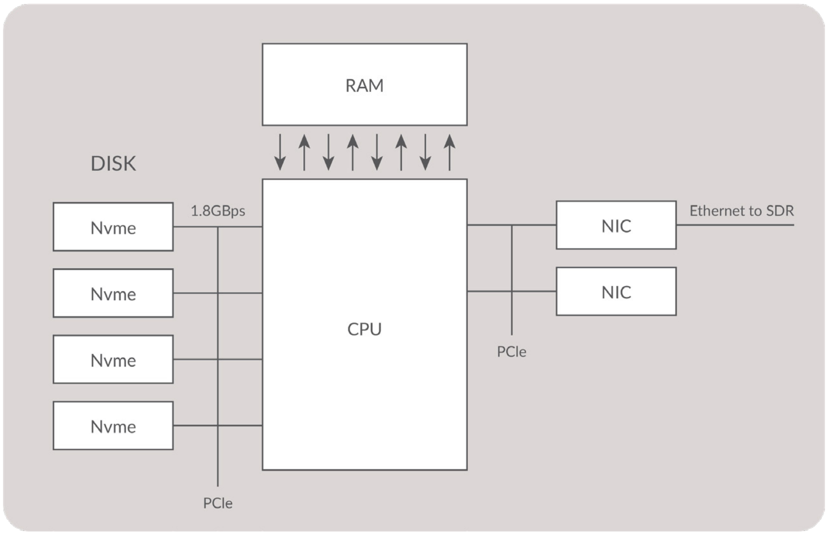
The FPGA enables the core functionality of the SDR and facilitates reprogrammability and high-throughput requirements. FPGAs achieve high-throughput requirements because of their parallel architecture avoiding propagation delays associated with OS-based or bare-metal processors.

The primary functions of the FPGA include performing complex signal-processing tasks and supporting the transfer of processed data between the converter devices and host system in a deterministic fashion and at high rate. Oftentimes, the limiting factor is the digital interface between the SDR and the host machine; multiple high-speed interfaces such as 40-/100-Gb/s Ethernet ensure maximum bandwidth.

For transmit applications, SDRs also benefit from onboard memory to help buffer sample transmission. This enables greater variations between the instantaneous throughput of the system (which varies on non-deterministic or non-real-time platforms), and the average throughput (which must match the rate at which the SDR is sending or receiving samples).



2. An SDR transmitter will need to translate the data stream into its RF components.



3. A host system will require a sufficient number of PCI Express lanes to handle the throughput from the SDR connection.

CPU and Memory Selection

The host system must be able to keep up with the SDR platform to process all the data that's being sent to it. To this end, the primary consideration of the CPU relates to the number of available cores, supported PCIe architecture, and processor speed. To provide some abstract numbers, supporting full-rate capture applications at 100 Gb/s generally requires around six to 12 cores, and a minimum of 32 PCIe v4.0 lanes (x16 lanes to support the NIC, and another 4x4 lanes to support NVMe drives).

For multiprocessor systems, it's critical to ensure proper core affinity between the NIC and storage drives. Care must be taken to specify core and process affinities to avoid unnecessary transfers across PCIe root complexes hosted by different CPUs.

The host system should also possess a memory controller that can support error correcting codes (ECC) to maintain data integrity. Similar to the SDR platform, ensuring ample memory size is useful as is maximizing memory bandwidth by populating all available memory slots. This also ensures efficient utilization of available direct-memory-access (DMA) controller resources and maximizes memory bus size and throughput.

NVM (SSD) and NVMe Controller

For rapid IO of large volumes of data and satisfying the requirement of maximum throughput, PCIe4-based solid-state drives (SSDs) can be utilized. SSDs are non-volatile memories (NVM), and with the added support of PCIe, they're called Non-Volatile Memory Express (NVMe). In addition to considering IO speed, we also need to consider file systems. High-performing file systems include XFS or EXT4. These file systems are reliable and quickly implement common disk operations.

Today's NVMe (PCIe4 SSDs) tend to be four-lane devices with mean throughput around 1.8 to 2.2 Gb/s. Avoid NVMe that uses caching because their mean throughputs are lower when the drive is full. The storage systems must have enough capacity to record the data continuously. This can be challenging, as recording 10 minutes of RF data streaming at 100 Gb/s requires over 7 TB of storage capacity.

Given the importance of matching the throughput of the entire system with each component, it's worth considering that a single NVMe drive supports approximately 16 Gb/s. If the throughput from the NIC is greater than that, as in the case of 100-Gb/s operation, it's necessary to split up writes across various devices.

Particularly for high-speed operation, software RAID configurations are discouraged in favor of direct access to the devices, or, if absolutely required, hardware RAID support. This allows several slower devices to be aggregated together. By carefully measuring the bandwidth associated with each storage device and matching that with the bandwidth of the NICs, you can ensure sufficient capacity to effectively read and write data from the storage devices.

Network Interface Card

The NIC facilitates the connection between the SDR platform and the host system. There are two common NIC solutions: commodity adapters and specialized FPGA adapters.

Though many commodity adapters on the market can handle 100-Gb/s throughput, it's not under every traffic condition, such as small packets. FPGA-based adapters have an edge with the primary benefit being large buffers (in the form of onboard memory) that support 100% packet-capture operations. Some FPGA acceleration also helps reduce protocol overhead as larger chunks of data can be transferred per operation. In addition, FPGA adapters are able to aggregate traffic in hardware at line rates as opposed to commodity adapters. In most cases, there's a substantial benefit to using FPGA-accelerated NICs.

Software-Defined Networking (Kernel vs. DPDK)

As shown in *Figures 1 and 2*, operating-system kernels handle networking through their network stack. This stack is good enough for most applications, but when it comes to high-throughput systems, it's simply too slow. This problem is compounded by the fact that network speeds are increasing while the network stack can't keep up.

To solve this issue, the kernel network stack can be bypassed with other software networking tools such as DPDK. Bypassing the kernel network stack also allows for new technologies to be implemented without needing to change core operating-system kernel code.

Kernel bypass stacks move protocol processing to the user space. This lets applications directly access and handle their own networking packets and increases throughput. It's a very important consideration for the host system, since using the kernel stack would defeat the purpose of designing the hardware components to handle a certain throughput. For this reason, employing a kernel bypass stack is vital in the operation of a high-throughput SDR system.

Analysis

To help illustrate why all of these considerations must be made, we can use *Figure 3* to illustrate an example of a bus-limited system. Let's assume an application requirement to stream data from an SDR to storage has a rate of 40 Gb/s. To begin with, we obviously need to ensure the SDR supports sufficiently high transfer rates over an appropriate interface. In this case, we'll assume a 40-Gb/s link between the SDR and the host computer.

As data is streamed over the NIC, the data is read over the PCIe bus by the CPU, which transfers it to memory, and arranges it to be written to the NVMe storage device. This requires a sufficient number of PCIe lanes (whose number depends the PCIe version supported by the CPU) to accommodate the concurrent reading the data from the NIC, which is writing to the storage devices.

Conclusion

High-throughput SDR systems unlock many applications. And creating such a system is very attainable with current technology. A high-end consumer computer has the required PCIe lanes and memory capacity to support a large influx of data. When paired along with a suitable processor and kernel bypass networking stack, the host system will be able to process all of the data coming in from the SDR platform. An appropriate storage solution can be set up using RAID and multiple SSD drives.

Finally, many SDR platforms exist that can fill the need for the analog front end of the system, such as high-performance SDR solutions developed by Per Vices. Many of these SDR providers also can provide high-performing storage solutions configured to work at the high data rates demanded by these applications. By ensuring proper data flow throughout the application, unnecessary slowdowns are avoided, and success is yours.

Victor Wollesen is the CEO and co-founder of Toronto-based software-defined-radio company Per Vices Corp. Victor has an honour's degree in Physics with a specialization in Astrophysics from the University of Waterloo in Ontario, Canada. He has co-authored several peer-reviewed papers on SDR technology, one of which was presented at IEEE's Radar Conference in 2020. Victor is a member of the Canadian Armed Forces, and in his free time enjoys putting his recreational pilot's license to use.

Reza Mohammadi is a fourth-year engineering student at the University of Toronto. His work at Per Vices focuses on network security.