Electronic Design

By TAMMY CARTER, Curtiss-Wright Defense Solutions

GPU Trends: The Quest for Performance, Latency, and Flexibility in ISR Systems

Employing strategies such as GPUDirect, PCIe Device Lending, and implementing SISCI API can help system integrators optimize ISR solutions.

For military intelligence, surveillance, and reconnaissance (ISR) applications, such as radar, EO/IR (electro-optic/infrared), or wideband ELINT (electronic intelligence), the ongoing problem is how best to handle the expanding "firehose" of data, fed by an increasing number of wide-bandwidth platform sensors. To handle this massive inflow of data, and the complex algorithms required to process it, state-of-the-art computational engines and data-transport mechanisms are essential.

Deployed High Performance Embedded Computer (HPEC) systems designed to support these applications typically have a heterogeneous architecture of high-performance FPGAs, GPUs, and digital signal processors, or DSPs (today, often Intel Xeon-D based modules). GPUs provide a large number of floating-point cores tuned for complex mathematical algorithms, which makes them ideal for processing the complex algorithms used in ISR applications. In comparison, a single Intel Xeon-D processor can provide a peak throughput of ~600 MFLOPS, while NVIDIA's Pascal P5000 GPU sports 6.4 TFLOPS of peak performance.

Tighter Integration

Today, ISR system integrators have three main goals: minimize latency, maximize system bandwidth, and optimize configuration flexibility within their given SWaP constraints. To address these issues, leading COTS vendors of OpenVPX modules are seeking ways to provide closer integration between the compute elements.

In the beginning, sensor data preprocessed by the FPGA had to be copied to the CPU, which subsequently copied it to the GPU for further processing. Then, NVIDA introduced GPUDirect, which added the capability to move the data directly from an FPGA or network interface, such as Mellanox Infiniband, to a GPU. By eliminating extra copies, both latency and backplane utilization were decreased.

Such an approach works well until the amount of incoming data overwhelms the system, such that one batch of data hasn't completed processing before the next batch of data arrives. This can result either from the transport systems being overwhelmed (I/O bound) or the GPU not completing the calculations in the required time frame (compute bound).

When using GPUs, the limiting factor is often the I/O, and this is usually addressed by employing either a round-robin distribution of the incoming data, and/or pipelining the processing stages. Unfortunately, as sensor data continues to increase, it's become apparent that new techniques are required.

PCI Express

In OpenVPX systems, the standard interface between the FPGAs, GPUs, and CPUs is PCI Express (PCIe)—it offers the fastest path to and from the processor, and by definition, connects to other devices via the expansion plane. Offloading the Ethernet with the PCIe connection reduces latency and increases throughput.

Based on the original PCI parallel bus design, PCIe is controlled by a single "master" host called the root complex that scans the bus to find and enumerate all connected devices. When a PCIe switch is used to connect multiple devices to a root complex, it's called a transparent bridge (TB), and all devices operate in a single address space. With a TB, two root nodes (processors) can't be connected because there will be a memory address conflict.

When a PCIe switch port is configured as non-transparent bridge (NTB), a root node doesn't look to enumerate devices beyond that switch port. So, when either of the two processors enumerates their NTB port, the port requests memory on that processor. The NTB port provides the common memory address translation to either side.

Multicasting

The next step to reducing latency is to multicast the incoming data to multiple GPUs. Until recently, multicasting GPUs in deployed ISR systems has been impractical because previous PCIe implementations were hampered by the traditional design limitations of one-to-one connections. The problem of how to multicast data across multiple GPUs using PCIe links increases the already complex task of programming the PCIe interface and setting up the bridges and root complexes.

Included in Curtiss-Wright's OpenHPEC Accelerator Suite, Dolphin Interconnect Solutions' PCI Express (PCIe) Fabric Communications Library provides the software needed to squeeze every ounce of performance from the PCIe interface. It also eases PCIe programming by abstracting the otherwise time-consuming code to simple APIs.

The Software Infrastructure Shared-Memory Cluster Interconnect (SISCI) is a well-established API and is the fastest way to exchange data. It can be used for Programmed IO (PIO), where a pointer is used to directly access the remote memory with the lowest latency, or as remote direct memory access (RDMA) where the DMA controller of the PCIe-NTB copies data from remote to local memory with the highest bandwidth.

One of the latency-reducing features of SISCI is reflective memory/multicast. The PCIe switch will send out data on all connected ports simultaneously, meaning all nodes will receive data virtually simultaneously when connected to a single switch. When multiple bridges are used, each switch hop will add a tiny delay to the data delay.

In a typical backplane-based ISR system, the CPU is tied directly to a GPU that resides on its XMC mezzanine card, or in a separate slot in the chassis. For example, dual Xeon-Ds on a 6U OpenVPX board are connected via a PCIe switch. Likewise, a 6U GPU card might have two GPUs connected via a PCIe switch to each other and the backplane. Using this two-board 6U DSP and GPU system "slice," one Xeon-D can control both GPUs, or each of the Xeon-Ds can control one of the GPUs. With a three-board combination of one DSP board and two GPU boards, one Xeon-D controls the upstream GPU card while the second Xeon-D controls the downstream GPU board.

If the system is based on the smaller 3U OpenVPX form factor *(see figure)*, the latency issue becomes even more acute because there might only be one x4 PCIe link available to the CPU board. The solution becomes even trickier if the 3U system doesn't include a central switch.

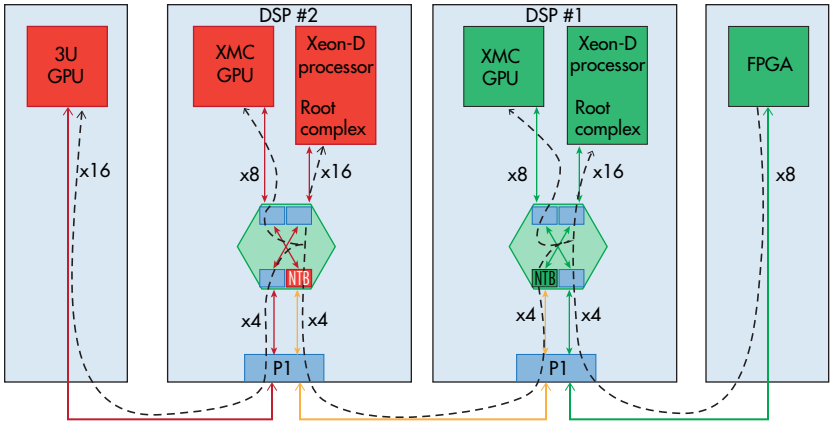
Creating Flexibility in Combined Systems

ISR systems can run multiple modes, but now system integrators also want the ability to combine systems, such as radar and EW, into a single processing box. In such a configuration, the data flows and the computation requirements will vary greatly between each system's modes. Since the GPUs and FPGAs are attached with cables, or connected via the backplane, flexibility can be tricky. Some flexibility might be achieved by reconfiguring the PCIe bridge ports, but what if a CPU could borrow a GPU, FPGA, or even non-volatile memory attached on the PCIe bus?

With PCIe Device Lending by Dolphin, devices can be borrowed over the PCIe network without any software overhead. Device lending is one simple way to reconfigure systems and reallocate resources. The lending function makes the devices available on the network for temporary access. The borrowing function searches the available devices, and then the selected devices can be borrowed temporarily as required. When the work is complete, the devices will be released.

At the GTC 2019 conference, Dolphin demonstrated its proof-of-concept software library for creating GPU-oriented applications with GPU Direct using capable GPUs and commodity NVMe disks. The memory-mapping capabilities of PCIe NTBs are used to set up efficient I/O data paths between GPUs and disks that are attached to different root complexes, allowing multiple GPUs to access a remote disk.

While GPUs have clear advantages for a variety of high-performance ISR applications, some designers are concerned with the lack of determinism exhibited by these



Here's an example of a 3U configuration.

devices. The good news here is that NVIDIA is now working to improve determinism on their GPUs. In CUDA, NVIDIA will expose APIs to describe tasks and the dependencies between tasks, thus providing more control over them. At the system level, NVIDIA is working on a timing-triggered scheduler.

What About Legacy Code?

While new systems can readily benefit from these recent advances in GPU architectures, there is, of course, lots of legacy software still in use. Helping to protect the vast investments in this legacy code is OpenACC. It's a standard programming language that makes optimizing and porting older existing serial code to a multicore processor, like the Xeon-D or a GPU, a very simple process.

All that's required with OpenACC is a hashtag and a definition of the platform that the code will run on. In turn, the code will be able to run on the targeted board. This makes it possible to have optimization pragmas that enable code to be parallelized in small incremental stages (like one loop at a time), essentially allowing the system to take "baby steps" when optimizing/porting the existing code. While not as efficient as handwriting the code in CUDA, the results are impressive, and over the course of a few weeks can deliver great results for optimizing and porting years-old legacy code to a GPU.

When searching for the best possible signal-processing performance from a heterogeneous system, it's good to consider using GPUDirect to cut out the "middle man" CPU in order to reduce latency. The next step is to explore multicasting from the front end to multiple GPUs and CPUs, even if the data must pass through several bridges.

For maximum system flexibility, integrators should check out a library system of devices that can be borrowed using PCIe Device Lending. Keep software on time and under budget by using the SISCI API instead of writing all the PCIe code from scratch. For fast-turn porting of legacy, play with pragmas in OpenACC. These strategies can be helpful guides for any ISR system integrator seeking improved performance, latency, and flexibility.