WHITE PAPER

Rapid Prototyping a High-Performance Embedded Computing System

abaco SYSTEMS

Rapid Prototyping a High-Performance System for UAV SAR Processing

A gating factor in the development of sophisticated, high performance applications – especially those to be deployed in highly SWaP-constrained environments – is often the availability of appropriate hardware. However: there is a solution…

Introduction

Your next OpenVPX system design will likely need a customized backplane to provide the appropriate fabric interconnect between general purpose processors (GPP), graphics processing units (GPU), fabric switches, imaging sensors, and input/output to other systems. But: customization is rarely synonymous with quick-turn design, and today's development cycles are feeling the squeeze. While hardware and software development can begin concurrently, hardware availability is the ultimate barrier to a qualified design.

So: having representative hardware at an early stage in the development process aids in the timely achievement of both hardware and software milestones.

The lab-to-rugged transition

Clearly, the sequential process of specifying a backplane design, waiting for fabrication, and then testing the system is not deadline-friendly, and does not provide flexibility when (and they always do) design parameters change midstream. A common way around this is to develop with desktop PCs or rack-mounted computers, graphics cards, and commercial network switches – and then make the change to rugged embedded hardware as it becomes available. Further, the use of open source operating systems like Linux, widely-supported programming APIs like OpenCL, Open VX, and OpenCV, and industry-standard math libraries and inter-processor communication (IPC) protocols – such as those found in Abaco's AXIS software suite – aid in making the lab-to-rugged transition one of relative elegance.

However: Are software boot-up times accurately represented? Are devices responding properly to reset signals and enumerating properly? How to account for custom BIOS settings? Can secure boot, trusted execution, or built-in test features be exercised? Are control- and data-plane fabrics true to the rugged implementation? Did you get used to more memory, larger caches, and higher clock speeds than will be available in the SWaP-constrained system? What about all the features included in the single board computer (SBC) software development kit (SDK) that couldn't be exercised on a PC? And how will that pesky watchdog timer be handled? That's just what comes up on day one.

Open CL and Open VX were created by the Khronos Group, a collaboration of industry partners. OpenCL (Open Computing Language) is an open standard for cross-platform parallel programming of diverse processors. OpenVX (Open Visual Acceleration) is an open standard for cross platform acceleration of computer vision applications. Open CV was an Intel Research initiative and is maintained by OpenCV.org. OpenCV (Open Computer Vision) provides acceleration for vision applications by leveraging Intel IPP, OpenCL and CUDA as appropriate for the hardware.

The solution

An ideal development cycle will incorporate something close to the actual hardware as soon in the development cycle as possible. On the one hand, waiting to develop on the actual hardware can be ideal, but doesn't allow much time for software de-risking activity or adaptation. On the other hand, developing on PCs is a great way to get a head start, but defers a mountain of details until late in the game.

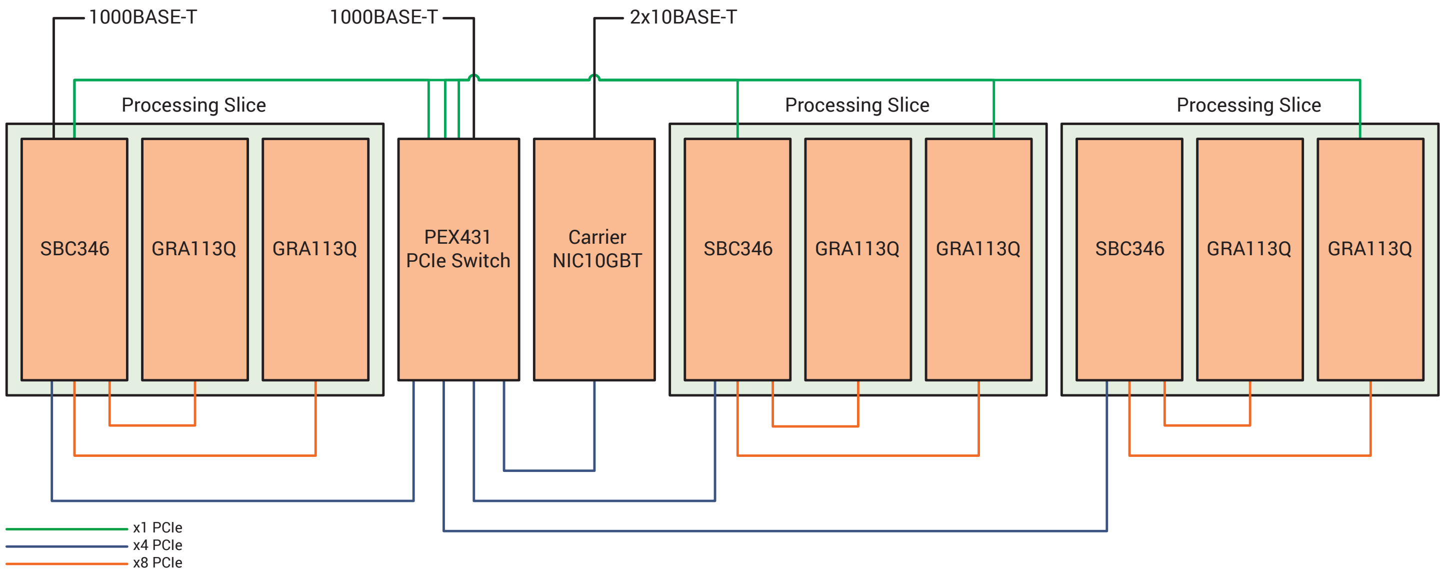
The solution? An 'uncommitted' OpenVPX backplane would provide several slots whose I/O is routed through the backplane from the front VPX connectors (designated by connectors J0, J1, and J2) to the rear VPX connectors (designated by connectors RJ0, RJ1 and RJ2), at which point rear transition modules (RTMs) pick up the I/O and provide industry standard connectors. RTMs are then cabled together as needed. Uncommitted backplanes are readily available because they are not custom, and are flexible because fabric connectivity and switching topology can be modified by simply making cabling changes.

Abaco did exactly this with the SCVPX3U-12 – a 3U Open VPX bench-top or rack-mount chassis with a 12-slot uncommitted backplane. Each payload slot provides differential I/O wafers that feed through to the rear, and accepts a 6U-height RTM for run-of-the-mill I/O like Gigabit Ethernet, video, USB, and SATA, plus data plane fabric like x16 PCIe and 10 Gigabit Ethernet.

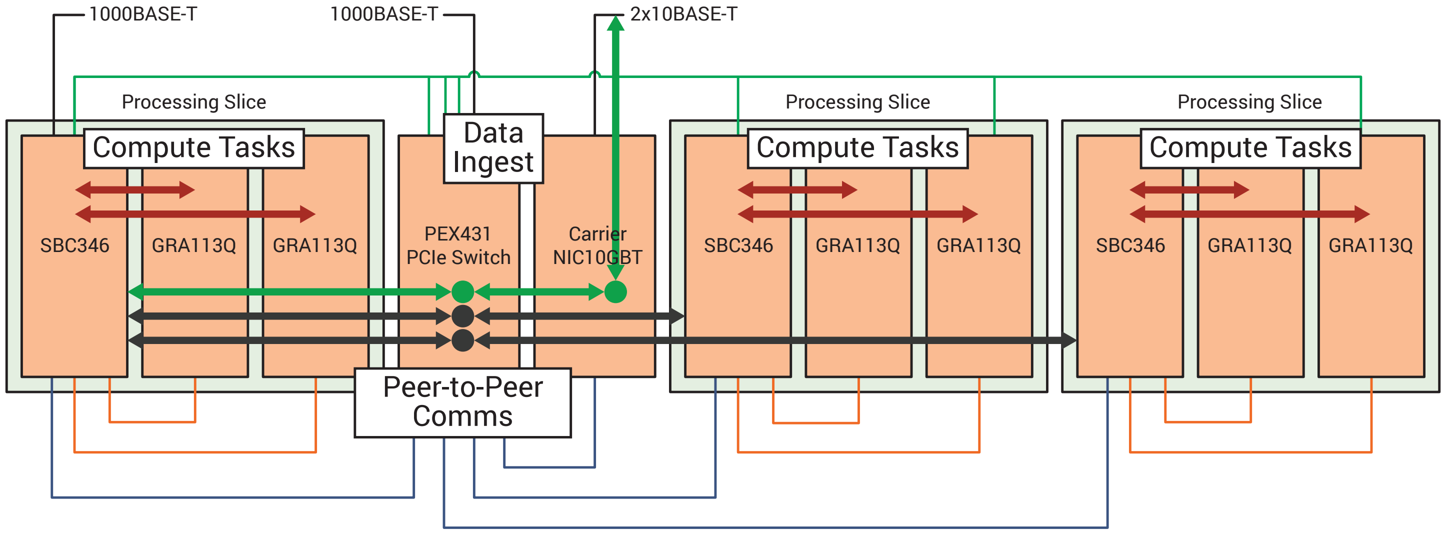
A real world example

Let's take a look at an example customer's system architecture. The application is a synthetic aperture radar back-end processor payload for a medium-altitude endurance UAV. The system provides a hybrid platform for general-purpose processing (GPP) on CPU and general-purpose processing on GPU (GPGPU), with a tightly-coupled peer-to-peer communications path and a 10 Gigabit pipe to the radar front-end's receiver-exciter (REX) and beam steering. It contains three pairs of CPU-GPU processors, each pair known as a slice. The members of each slice communicate via PCIe and Gigabit Ethernet, and each slice is linked to other slices through a PCIe/1GbE switch module. One CPU of the three slices is responsible for managing the system's PCIe and 10GbE data flow – sending tasks to other peer slices for processing, collecting results, and passing data to the front-end.

SCVPX3U-12: Twelve-slot 3U Open VPX chassis that provides for 6U rear transition modules for fabric and I/O connectivity



UAV Pod Processing System Block Diagram



System Data Flow Diagram

The Abaco system hardware components are as follows:

Single-Board Computer	SBC346	Intel 4th gen Core i7 (Haswell)	3U OpenVPX
Graphics Processor	GRA113Q	NVidia GM107 (Maxwell)	3U OpenVPX
PCIe switch/1GbE switch/ XMC carrier	PEX431	PLX and Vitesse	3U OpenVPX
10G network interface	NIC10GBT	Intel X540 dual-port 10GBASE-T	XMC mezzanine

Careful consideration is given to achieve a PCIe architecture with the most favorable tradeoffs. The SBC346 managing the peer-to-peer comms data flow is configured as a PCIe root complex, while SBC346s in other slices are configured as non-transparent ports. This type of port allows the root complex to map the other slices in the topology. PCIe switch bandwidth is shared between the 10GbE data ingest and peer-to-peer comms; the advantage is that each slice has dedicated x8 PCIe lanes between CPU and GPU for highest throughput and lowest latency. Other arrangements could be used to distribute switching resources to provide exclusive bandwidth for data ingest and peer-to-peer comms, for example, but at a sacrifice to compute task throughput within slices.

Easily configured

The SBC346 managing the peer-to-peer comms data flow is also a PCIe synchronous clock source that drives the OpenVPX REFCLK signal to a fan-out buffer on the backplane to distribute it to other slots. The SBC346 boards in other slices are REFCLK receivers. This is easily configured in each board's UEFI/BIOS setup screen. With this arrangement, the root complex and other PCIe switches are in the same clock domain. Although an asynchronous clocking scheme is suitable for PCIe Gen 2, this common clock arrangement makes the system upgradeable to Gen 3 data rates. REFCLK is a differential 100MHz signal, and with a Gen 3 data rate of 8GT/s (Gigatransfers per second), a common clock is an effective way to minimize system jitter. Note that the final backplane design must take into account the appropriate design rules to support Gen 3.

A separate PCIe clock domain, synchronous to the CPUs on the SBC346s, is used between members of each slice. Peer-to-peer comms data flow is asynchronous to compute tasks but there are no ill effects for the system because these two functions can be loosely coupled with each other.

Abaco's peer-to-peer interconnect SDK is utilized to provide a virtual network between slices. This abstraction layer presents an IP socket-based interface and enables standard network protocols to be run over the PCIe interconnect. Hardware switch configuration is also required.

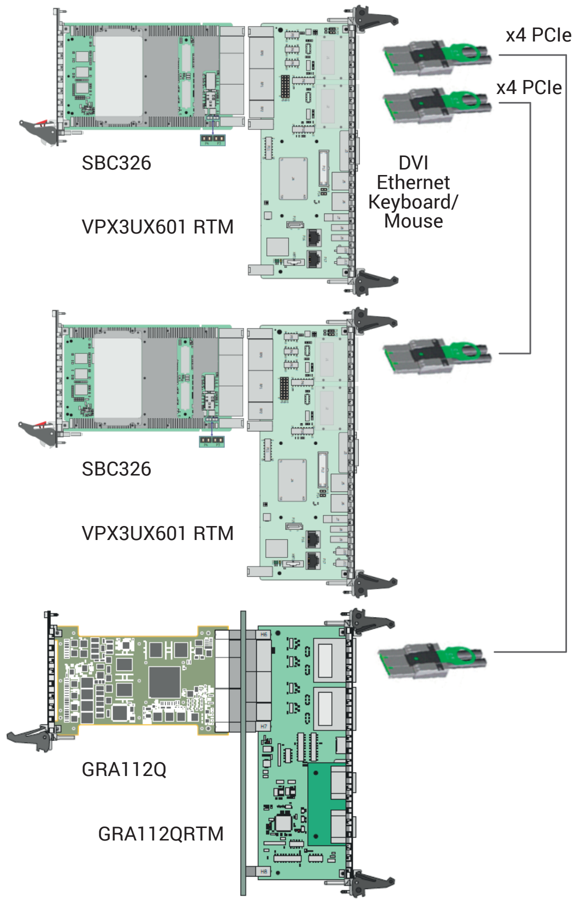
A useful measure for high performance embedded computing systems is floating-point operations per watt, or FLOPS/Watt. The Intel Core i7's dual 256-bit fused multiply-accumulate (FMA) instructions provide 32 single-precision (SP) FLOPS per cycle. The NVIDIA Maxwell's 64-bit FMA provides 2 SP FLOPS per cycle. The theoretical, or peak, FLOPS calculations are shown in the following table.

Processor	Cores	Clock	SP FLOPS/ cycle	Peak SP GFLOPS
Intel 4th gen core i7	4	2400 MHz	32	307.2
NVIDIA Maxwell	640	940 MHz	2	1203

Each CPU-GPU slice can provide a theoretical 1510 GFLOPS single-precision. The total throughput for three slices is approximately 4.5 TFLOPS single-precision, capable of hundreds of backscatter projections per second. The power consumption can be found by summing the typical power of each board in the system. We can also assume a power supply of 500W with an efficiency of 90%.

SBC346	3 x 55W
GRA113Q	3 x 40W
PEX431	25W
NIC10G	25W
500W Power Supply @90% Eff	50W
System Power	385W

So we've achieved 4.5 TFLOPS in 385 Watts, or 11.7 GFLOPS for every Watt of power consumed.



Prototyping the real world example

Now let's look at an actual development system Abaco assembled to speed the customer along the development path. The critical communication paths are between members of the slice and between slices, so we can provide a suitable system using just one CPU-GPU pair, along with a second CPU. We knew software development would progress in stages, so it was decided that the PEX431 switch and the NIC10G network interface were not critical for the development system. These would be integrated later once the backplane was fabricated and other hardware was received. To meet tight timelines, we chose representative hardware in the SBC326 and GRA112Q, instead of SBC346 and GRA113Q.

Conclusion

When you tackle your next custom VPX design, take advantage of Abaco's OpenVPX backplanes and system components to enable a phased development approach and minimize development risk. By prototyping the critical compute and data paths, bottlenecks can be uncovered early in the cycle. Using the latest Intel Core i7 and Xeon-D processors, NVIDIA GPUs and fabric connectivity can provide multi-TeraFLOPS performance.

Appendix: Building your own development system

The instructions that follow show how to configure the hardware and install the software for this development system.

Hardware Setup

The development system consists of an SCVPX3U chassis, two SBC326, two VPX3UX601 RTM, one GRA112Q, and one GRA112QRTM. Two 0.5 meter PCIe cables are used.

On each VPX3UX601 RTM, populate jumpers between P7 and P8 with 8 links to enable the Ethernet signals. Set NVMRO by fitting P5 1-2, since the 3U backplane does not have NVMRO daisy chained to all slots.

Set one VPX3UX601 RTM as SYSCON by fitting a jumper on P5 3-4.

DVI monitors and USB keyboard and mouse can be used, or Ethernet can be used to SSH into each node from a remote computer.

Software

CentOS 7 is installed. The following packages are also installed:

a. Abaco SBC326 SDK, part number 162-SLIO-SBC326-01UC
b. Abaco SBC326 P2P Package, part number PLIA-SBC326-P1E1UC
c. NVIDIA CUDA 7.5- https://developer.nvidia.com/cuda-downloads
d. Netperf - netperf-2.7.0-1.el7.lux.x86_64.rpm

Software Installation Steps

The SBC326 P2P driver requires RHEL7. The CentOS 7 kernel is 3.10.0-229.7.2.el7.x86_64.

1. Install CentOS 7 from DVD ISO or Net-Install ISO. The 16GB SSD is used as the destination. Select Gnome Desktop with Development Tools.
2. Perform a yum update. Sometimes another process has yum in use. Kill that process to free-up yum.

```
ps aux | grep yum
kill <insert PID number>
yum update
```

3. Install DKMS prior to install SDK.

```
http://rpmfind.net/linux/RPM/epel/7/x86 _ 64/d/
dkms-2.2.0.3-30.git.7c3e7c5.el7.noarch.html
yum localinstall dkms-2.2.0.3-25.el7.noarch.rpm
–nogpgcheck
```

4. Install SBC326 SDK

```
tar -xzf 162-SLIO-SBC326-01UC.tar.gz
cd 162-SLIO-SBC326-01UC
./install _ sdk.sh --all
```

5. Install SBC326 P2P Package

```
tar -xzf PLIA-SBC326-P1E1UC-R04 _ 00.tar.gz
cd PLIA-SBC326-P1E1UC-R04 _ 00
cd kernel-driver
make install
```

The P2P-DRV-LINUX script is installed in /etc/init.d. This script can be used to manually start and stop the P2P service:

```
/etc/init.d/P2P-DRV-LINUX <start/stop>
```

Put the driver in rc.local for automatic startup at boot, and make rc.local executable:

```
echo modprobe P2P-DRV-LINUX >> /etc/rc.local
chmod +x /etc/rc.local
```

During initialization, the P2P nodes poll to check the availability of other nodes. The polling time is controlled in two files and defaults to 30 seconds. Modify the wait time parameters to 90 seconds.

For manual driver start-up, edit

```
/etc/init.d/P2P-DRV-LINUX and change to WAIT _
TIME=90.
```

For boot-time driver start-up, edit

```
/etc/modprobe.d/p2p _ module.conf and change to
p2p _ wait _ time=90.
```

To verify that the driver is loaded, look for pcie_p2p_net, p2p_plx, and p2p_setup modules:

```
lsmod|grep p2p
```

To verify that the driver was successful in making a peer to peer connection, check dmesg:

```
dmesg|grep p2p
```

You should something similar to:

```
eth0: p2p Ethernet over PCIE P2P Version 1.01,
MAC 00:01:01:01:00:00
```

Use ifcfg-eth0 script or the Network Manger GUI to configure the IP addresses for the P2P network, for example 192.168.200.2 and 192.168.200.3. Use ifconfig to view P2P Ethernet details.

Check that BIOS write protect for EEPROM is off before programming the PCIe EEPROM.

- Fit SYSCON and NVMRO jumper on VPX3UX601 RTM. Alternatively, fit SYSCON jumper and go to Chipset -> FPGA -> NVMRO Override.
- In BIOS: Chipset -> DIP Switch -> Config EEPROM Write Protection -> NVMRO Controlled, Program DIP Switch
- In BIOS: Chipset -> PLX Switch -> Unlock EEPROM
- Save and Exit and power cycle

Program the EEPROM:

```
cd <install-dir>/PLIA-SBC326-P1E1UC-R04 _ 00/
eeprom _ tools/plx
```

```
./plx-eeprom-prog-pci –m program –f ../../
eeproms/sbc326/<eeprom-file>
```

6. Install CUDA 7.5 which can be found at https://developer.nvidia.com/cuda-downloads.

```
sudo rpm –i cuda-repo-rhel7-7-5-local-7.5-18.
x86 _ 64.rpm
sudo yum clean all
sudo yum install cuda
```

7. Since Linux and NVidia graphics drivers are not compatible, remove the default Linux noveau graphics driver and install the NVidia driver. This is a multi-step process, and there are various methods. One can be found here: https://www.linkedin.com/pulse/20140808222919-219659043-rhel-centos-7-and-nvidia-drivers
8. Install netperf
   Download https://pkgs.org/centos-7/lux/netperf-2.7.0-1.el7.lux.x86_64.rpm.html

```
sudo yum localinstall netperf-2.7.0-1.el7.lux.
x86 _ 64.rpm
```

On both nodes:

```
systemctl stop firewalld (service may take a
few minutes to stop)
netserver –p 12865
```

On one node (example parameters):

```
netperf -fM -H 192.168.x.x -c -C -l10 -- -m
64000
```

Other Tips

1. To check PCIe bus speed:

```
lspci –vvv, find PLX bridge for Port 1, and
check LnkSta parameter. Look for 5GT/s x4.
For example, lspci –vvv –s 2:01.0
```

2. To rescan PCIe bus after making changes:

```
echo “1” > /sys/bus/pci/rescan or
echo 1 | sudo tee /sys/bus/pci/rescan
```

WE INNOVATE. WE DELIVER. YOU SUCCEED.

Americas: 866-OK-ABACO or +1-866-652-2226 **Asia & Oceania:** +81-3-5544-3973
Europe, Africa, & Middle East: +44 (0) 1327-359444
Locate an Abaco Systems Sales Representative visit: abaco.com/products/sales

abaco.com | @AbacoSys

abaco SYSTEMS