Sensor Processing with Rugged Kepler GPUs

GE Intelligent Platforms
Mil/Aero Embedded Computing

Dustin Franklin, GPGPU Applications Engineer
dustin.franklin@ge.com
443.310.9812 (Washington, DC)
Scalable HPEC

Deployable COTS supercomputing:

• Intel x86 CPUs
• NVIDIA GPUs
• Infiniband (QDR)
• 10GigE
• Switches
• Software stack
  • Linux + drivers
  • CUDA toolkit
  • MPI/OFED/ect

SWAP Reduction + Ruggedization
HPEC Portfolio

Systems

Displays and Panel Computers

Networking, Communications, & I/O

High Performance DSP + GPGPU

Sensor and Signal Processing

Single Board Computers

Applied Image Processing
Prepackaged Intel + NVIDIA Rugged Systems

- **MAGIC1** 3U VPX
  - Intel i7 Sandybridge
  - NVIDIA GPU

- **IPS5100** 3U VPX
  - Intel i7 Sandybridge
  - Dual NVIDIA GPUs
  - Tilera TilePro64
  - Dual port 10GigE-SR

- **IVD2010/15** 6U VPX
  - Intelligent Display (10” or 15”)
  - Intel SBC + NVIDIA GPU
  - Dual port 10GigE-SR

Environmentals: -40°C to 85°C, 40g’s shock, 0.1g²/Hz vibration, 70000 ft. altitude
Rugged CUDA-enabled NVIDIA GPUs

<table>
<thead>
<tr>
<th>GRA series</th>
<th>3U VPX</th>
<th>NVIDIA GPU (BGA)</th>
<th>45W</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPN series</td>
<td>6U VPX</td>
<td>Intel SBC + NV GPU (BGA)</td>
<td>120W</td>
</tr>
<tr>
<td>NPN series</td>
<td>6U VPX</td>
<td>dual NVIDIA GPUs (BGA)</td>
<td>100W</td>
</tr>
<tr>
<td>MXM series</td>
<td>6U VPX</td>
<td>dual MXM3.0A/B carrier</td>
<td>150-200W</td>
</tr>
</tbody>
</table>
HUNTSVILLE, AL – MAY 15, 2012 - GE today announced at GTC (NVIDIA GPU Technology Conference - San Jose, CA: May 14–17) the high performance GRA112 3U VPX rugged graphics board. Designed for demanding graphical applications as well as GPGPU (general purpose computing on a graphics processing unit) applications such as ISR (intelligence, surveillance, reconnaissance), radar and sonar, it features a 384-core ‘Kepler’ GPU from NVIDIA. Also provided is 2GBytes of GDDR5 memory which, together with the latest NVIDIA GPU, makes the GRA112 significantly more powerful than its predecessor, the GRA111 – for which it provides a simple, cost-effective technology insertion upgrade.

GE will also launch upgrades to the NVIDIA 96-core 6U VPX NPN240 dual-GPU platform and 6U VPX IPN250 Intel®/NVIDIA platform later in the year.
GRA112 – EXK107
Chip-down 384-core Kepler GPU

1. CUDA Engine
   - Cores: 384
   - Cores per SM: 192
   - SMs: 2
   - Compute Cap.: 3.0
   - FLOPS: 622 GFLOPS

2. Memory Subsystem
   - Global SDRAM: 2GB GDDR5
     + Bus width: 128-bit
     + Bandwidth: 64 GB/s
   - L1 Bandwidth: 220 GB/s
   - L2 Bandwidth: 96 GB/s

3. I/O
   - PCIe endpoint
     + Bandwidth: 16 GB/s
     + Lanes: x16/x8/x4/x2/x1
   - Video Out
     + 2x Digital: DVI
     + 2x Analog: VGA, RS170, custom, STANAG 3350
   - Video In
     + 1x Analog: RS170, NTSC, PAL

4. Environmentals
   - Chip Mount
     - BGA (BALL GRID ARRAY)
     - Conduction
   - Cooling
     - Passive
     - Operatioal (AMBIENT)
   - Storage
     - -40°C to 85°C
   - Shock
     - -50°C to 100°C
     - 40g peak, 11ms
   - Vibration
     - [MIL-STD-810E]
     - 0.1g^2/Hz
   - Humidity
     - 95%
GRA112 – EXK107

384-core Kepler GPU

3U VPX

384-core NVIDIA "Kepler" EXK107 GPU

GDDR5
GDDR5
GDDR5
GDDR5
GDDR5
GDDR5
GDDR5

Power

P0

x16 PCIe

P1

Ch1 - DVI
Ch2 - DVI
Ch3 - VGA
Ch4 - VGA

P2

FPGA Pass thru Scan Conv SOG
DP-DVI

TV decoders

TV in

x1 PCIe
## GRA112 – EXK107

384-core Kepler GPU

<table>
<thead>
<tr>
<th></th>
<th>GRA111</th>
<th>GRA112</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Process</td>
<td>40nm</td>
<td>28nm</td>
<td></td>
</tr>
<tr>
<td>Chip</td>
<td>GT215</td>
<td>EXK107</td>
<td></td>
</tr>
<tr>
<td>Cores</td>
<td>96</td>
<td>384</td>
<td>$\uparrow4.00\times$</td>
</tr>
<tr>
<td>FLOPS</td>
<td>290 GFLOPS</td>
<td>622 GFLOPS</td>
<td>$\uparrow2.15\times$</td>
</tr>
<tr>
<td>Global Memory</td>
<td>1GB DDR3</td>
<td>2GB GDDR5</td>
<td>$\uparrow2.00\times$</td>
</tr>
<tr>
<td>Global Bandwidth</td>
<td>22.5 GB/s</td>
<td>64 GB/s</td>
<td>$\uparrow2.85\times$</td>
</tr>
<tr>
<td>L1/Shared Memory</td>
<td>16KB</td>
<td>64KB</td>
<td>$\uparrow4.00\times$</td>
</tr>
<tr>
<td>L1/Shared Bandwidth</td>
<td>60 GB/s</td>
<td>220 GB/s</td>
<td>$\uparrow3.67\times$</td>
</tr>
<tr>
<td>L2/Texture Bandwidth</td>
<td>38 GB/s</td>
<td>96 GB/s</td>
<td>$\uparrow2.52\times$</td>
</tr>
<tr>
<td>PCIe endpoint</td>
<td>Gen 2</td>
<td>Gen 3</td>
<td></td>
</tr>
<tr>
<td>RS-170 video out</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>H.264 hw decode</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>H.264 hw encode</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

pin-compatible with GRA111

3U VPX
Kepler Streaming Multiprocessor
CUDA Compute Capability 3.0

- 192 cores
- 256KB register file
- 64KB shared memory / L1 configurable cache
- 32-bit multicast shared memory read access
- Increased atomics performance
- Quad-warp scheduling
- 2048 resident threads per SM
- $2^{31}-1$ blocks per grid (x dimension)
- 256 texture samplers
- 4096x4096x4096 3D textures

CUDA Toolkit 4.2

CUDA Toolkit 5.0 Alpha
Registered Developer Network

source: NVIDIA GeForce GTX680 whitepaper
__global__ void histogramPart1(uint *d_PartialHistograms, uint *d_Data, uint dataCount)
{
    //Per-warp subhistogram storage
    __shared__ uint s_Hist[HISTOGRAM256_THREADBLOCK_MEMORY];
    uint *s_WarpHist = s_Hist + (threadIdx.x >> LOG2_WARP_SIZE) * HISTOGRAM256_BIN_COUNT;

    //Clear shared memory storage for current threadblock before processing
    #pragma unroll
    for (uint i = 0; i < (HISTOGRAM256_THREADBLOCK_MEMORY / HISTOGRAM256_THREADBLOCK_SIZE);
        s_Hist[threadIdx.x + i * HISTOGRAM256_THREADBLOCK_SIZE] = 0;

    //Cycle through the entire data set, update subhistograms for each warp
    const uint tag = threadIdx.x << (UINT_BITS - LOG2_WARP_SIZE);

    __syncthreads();
    for (uint pos = UMAD(blockIdx.x, blockDim.x, threadIdx.x); pos < dataCount;
        pos += UMUL(blockDim.x, gridDim.x))
    {
        uint data = d_Data[pos];
        addWord(s_WarpHist, data, tag);
    }

    //Merge per-warp histograms into per-block and write to global memory
    __syncthreads();
    for (uint bin = threadIdx.x; bin < HISTOGRAM256_BIN_COUNT; bin += HISTOGRAM256_THREADBLOCK_SIZE)
    {
        uint sum = 0;
        for (uint i = 0; i < WARP_COUNT; i++)
            sum += s_Hist[bin + i * HISTOGRAM256_BIN_COUNT] & TAG_MASK;

        d_PartialHistograms[blockIdx.x * HISTOGRAM256_BIN_COUNT + bin] = sum;
    }
}
With Atomics

```c
__global__ void histogram( uint16* img, uint* hist, int imgWidth )
{
    const int x = blockIdx.x * blockDim.x + threadIdx.x;
    const int y = blockIdx.y * blockDim.y + threadIdx.y;

    atomicAdd( hist + img[y * imgWidth + x], 1 );
}
```

atomicCAS
atomicAdd
atomicSub
atomicExch
atomicMin
atomicMax
atomicInc
atomicDec
atomicAnd
atomicOr
atomicXor
Atomics Performance

16-bit histogram (64k bins)

megapixels / second

<table>
<thead>
<tr>
<th></th>
<th>Tesla (GT215)</th>
<th>Fermi (GF104)</th>
<th>Kepler (EXK107)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>194</td>
<td>265</td>
<td>3,226</td>
</tr>
</tbody>
</table>

Kepler: >10x atomics performance
Kepler Tier-0 Features
Focus on Performance-Per-Watt

- PCIe gen 3 endpoint (16GB/s @ x16)
- 512KB L2 cache (512B/clk bandwidth)
- Fine-grained dynamic clocking (GPUBOost)
- H.264 encoder in HW (encode 1080p @ 240 fps)
- Quad displays (4x DVI-DL/HDMI/DisplayPort)

Source: NVIDIA GeForce GTX680 whitepaper
Kepler Development Systems

- GeForce 640 (128-bit GDDR5)
- Acer Aspire Timeline M3-581TG
  - Sandybridge + GT 640M
- HP Pavilion DV6-7017tx
  - Ivybridge + GT 650M
- Not all 600M’s are created equal...

<table>
<thead>
<tr>
<th>Model</th>
<th>Chip</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTX 675M</td>
<td>GF114</td>
</tr>
<tr>
<td>GTX 670M</td>
<td>GF114</td>
</tr>
<tr>
<td>GTX 660M</td>
<td>GK107</td>
</tr>
<tr>
<td>GTX 650M</td>
<td>GK107</td>
</tr>
<tr>
<td>GTX 640M</td>
<td>GK107</td>
</tr>
<tr>
<td>GT 640M LE</td>
<td>GF108 or GK107</td>
</tr>
<tr>
<td>GT 635M</td>
<td>GF108</td>
</tr>
<tr>
<td>GT 630M</td>
<td>GF108</td>
</tr>
<tr>
<td>GT 620M</td>
<td>GF117</td>
</tr>
<tr>
<td>GT 610M</td>
<td>GF119</td>
</tr>
</tbody>
</table>
GPUDirect Peer-to-Peer with FPGA
Without GPUDirect

- In a standard plug & play OS, the two drivers have separate DMA buffers in the host memory
- Three transfers to move data between I/O endpoint and GPU
GPUDirect Peer-to-Peer

- I/O endpoint and GPU communicate directly, only one transfer required.
- Traffic limited to PCIe switch, no CPU involvement in DMA
- x86 CPU is still necessary to have in the system, to run NVIDIA driver
GPUDirect P2P - Endpoints

- Peer-to-Peer DMA is open technology and works with a wide range of existing devices

GPUDirect Peer-to-Peer Checklist

- PCIe endpoint [gen 1,2,3]
- DMA engine

- It’s free.
  - Supported in CUDA 5.0
  - Users can leverage APIs to implement P2P with 3rd-party endpoints in their system

- Practically no integration required
  - No changes to device HW
  - No changes to CUDA algorithms
  - I/O device drivers need to use DMA addresses of GPU instead of SYSRAM pages
GPUDirect P2P - Topology

- Peer-to-Peer DMA is flexible and works in many system topologies
  - Single I/O endpoint and single GPU
  - Single I/O endpoint and multiple GPUs
  - Multiple I/O endpoints and single GPU
  - Multiple I/O endpoints and multiple GPUs
  - I/O endpoint(s) and GPU(s) in different PCI domains via non-transparent bridging

![Diagram of GPUDirect P2P Topology]

- PCI domain A
  - I/O endpoint
  - PCIe switch
  - CPU

- PCI domain B
  - I/O endpoint
  - PCIe switch
  - CPU
  - GPU
  - GPU memory

with or without PCIe switch downstream of CPU
GPUDirect P2P - Symmetry

- Peer-to-Peer DMA is bi-directional and bi-masterable
  - I/O endpoint(s) streaming into GPU(s)
  - I/O endpoint(s) streaming from GPU(s)
  - GPU(s) streaming into I/O endpoint(s)
  - GPU(s) streaming from I/O endpoint(s)
GPUDirect P2P - Performance

- **25X decrease in latency (baseline)**
  - Measuring FPGA↔GPU datapath vs. FPGA↔CPU↔GPU datapath
  - The more data transferred, the greater the performance gain

- **CPU Usage – 0%**
  - CPU is no longer burning cycles shuffling data around for the GPU
  - System RAM is no longer being thrashed by DMA engines on FPGA + GPU
  - All PCIe traffic stays downstream of PCIe switch, root complex is not involved
  - Asynchronous co-processing model

- **Increased PCIe efficiency + bandwidth**
  - bypass SYSRAM, MMU, root complex: limiting factors for GPU DMA transfers
SIGINT – beamforming latency

![Graph showing beamforming latency.

- X-axis: beams per sweep
- Y-axis: μs

Lines indicate latency with and without GPUDirect and GPUDirect P2P.]
SIGINT - beamforming latency

- Without GPUDirect
- GPUDirect P2P

Latency values for different numbers of beams per sweep:
- 128 beams: 2902 μs
- 256 beams: 5619 μs
- 512 beams: 11055 μs
- 768 beams: 16456 μs
- 1024 beams: 21924 μs
- 1536 beams: 32816 μs
- 2048 beams: 43784 μs

Graph shows the latency increase with the number of beams per sweep.
GPU – accessing SYSRAM \(^{(DDR^3)}\) vs. FPGA \(^{(DDR^2)}\)

GPU PCIe latency, uint32

<table>
<thead>
<tr>
<th></th>
<th>READ</th>
<th>WRITE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSRAM</td>
<td>1,280</td>
<td>342</td>
</tr>
<tr>
<td>FPGA</td>
<td>2,472</td>
<td>441</td>
</tr>
</tbody>
</table>
Backend Interconnects
32-byte Latency

- 10Gbe
- 10Gbe RDMA
- Infiniband (DDR)
- Infiniband (QDR)
- SRIO (x4)

Latency in microseconds:
Throughput

512K message throughput

<table>
<thead>
<tr>
<th>Interface</th>
<th>Throughput (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10Gbe</td>
<td>500</td>
</tr>
<tr>
<td>10Gbe RDMA</td>
<td>1000</td>
</tr>
<tr>
<td>Infiniband (DDR)</td>
<td>1500</td>
</tr>
<tr>
<td>Infiniband (QDR)</td>
<td>3500</td>
</tr>
<tr>
<td>SRIO (x4)</td>
<td>1000</td>
</tr>
</tbody>
</table>
CPU Overhead

Max Sustained CPU Overhead (in %)

- 10Gbe
- 10Gbe RDMA
- Infiniband (DDR)
- Infiniband (QDR)
- SRIO (x4)
Riding the HPC wave of collaboration between NVIDIA and Mellanox for **GPUDirect** enables low latency, high throughput inter-process communication (IPC) and system scaling to multi-board GPGPU clusters.
MLNX_OFED doesn’t support GPUDirect Peer-to-Peer (yet)

HCA driver & Linux networking stack complexity prohibits experimentation

Instead, utilize Tilera processor

- Tilera TILEPro64 (64-core MIPS)
- Input: dual 10GigE (XAUI)
- Output: dual PCIe x4

**Diagram:**

1. Rx packets
2. Depacketization
3. DMA to GPU
GPUDirect P2P with 10GigE (Tx)

- MLNX_OFED doesn’t support GPUDirect Peer-to-Peer (yet)
- HCA driver & Linux networking stack complexity prohibits experimentation
- Instead, utilize Tilera processor
  - Tilera TILEPro64 (64-core MIPS)
  - Input: dual 10GigE (XAUI)
  - Output: dual PCIe x4
GPUDirect Summary

- ≥ 25X latency reduction
- High bandwidth and PCIe efficiency
- Free’s up CPU and SYSRAM resources for asynchronous co-processing

- Utilize GPUDirect throughout your system
- Make sure chip vendors know DMA interoperability is important to you!

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA</td>
<td>GPUDirect Peer-to-Peer</td>
</tr>
<tr>
<td>Mellanox ConnectX IB/10G</td>
<td>GPUDirect v1.0</td>
</tr>
<tr>
<td>Tilera 10G</td>
<td>GPUDirect Peer-to-Peer</td>
</tr>
<tr>
<td>Other Fermi GPUs</td>
<td>GPUDirect Peer-to-Peer</td>
</tr>
<tr>
<td>Video I/O</td>
<td>GPUDirect for Video</td>
</tr>
</tbody>
</table>
Concurrent Redhawk6
RT Linux extensions for RHEL6

- Linux kernel, drivers, and libs optimized to minimize jitter and process dispatch latency
- Optimized NVIDIA driver for realtime determinism under heavy CUDA processing load
- Convenient tools for CPU Shielding + Affinity of interrupts, daemons, and user processes
- Quickly disable/enable preemption from within applications
- Deterministic, fixed-length context switching
- Priority Inheritance – avoids Priority Inversion in critical sections
- Frequency-based scheduler
- Nightstar – GUI tools for realtime profiling, application + kernel debugging
Concurrent Redhawk6
Without RT Extensions
Concurrent Redhawk6

With RT Extensions
AXISLib-GPU

http://defense.ge-ip.com/products/3547

• CUDA implementation of VSIPL
• Easy migration for users of VSIPL on other platforms
• GE also offers versions of AXISLib/VSIPL optimized for AVX, SSE, and PPC/AltiVec
• Available as a backend for Mentor Graphics VSIPL++

• 200+ elementwise vector functions
• 80+ matrix and vector operations
• 20+ FFT functions
• 20+ filtering, convolution, and correlation functions
• 20+ complex scalar functions
• 15+ linear system solvers
• 5+ random number generators
Rugged CUDA Imaging Applications

Hyperspectral Imaging

- Hyperspectral imagers capture a large number of bands (448) over wide, continuous domains of the EM spectrum (400-2500nm)

- Material classification performed by comparing the spectral signature of each pixel to a library of material signatures.

- Applications include:
  - Explosive particle & residue detection
  - Anti-mine & Anti-IED
  - Tracking under camouflage
  - Aircraft/missile/rocket signature intelligence
  - Drug enforcement

- Large amount of bandwidth to process (> 2 GB/s)
- Acceleration using CUDA: 82x faster than CPU
Rugged CUDA Imaging Applications

Wide-area EO/IR surveillance

- Imaging pods on UAVs: arrays with hundreds of cameras
- GPUs stitch all cameras together to form gigapixel mosaic
- Ground coverage: > 40 km², 0.15m per pixel
- A number of tasks are run on GPU using CUDA:
  - Seamless alignment & mosaicing
  - Moving target indicator (MTI)
  - Vehicle/Human detector & tracking
  - Coherent change detection
  - Geolocation
  - H.264 compression (for downlink to groundstation)

- Acceleration using CUDA: 105x faster than CPU
Rugged CUDA Imaging Applications

Mosaicing Pipeline

This is the process for aligning, warping, and visualizing the mosaic. All of it runs on the GPU, in CUDA and OpenGL.

SURF keypoints are extracted and matched from different image tiles. It is this correlation between features in the images that provides the alignment. Since the SURF keypoints are invariant to changes in viewpoint, the matches (and hence the alignment) are very accurate.
MAGIC1
Deployable Intel + NVIDIA Prepackaged Rugged System

- Intel SBC + NVIDIA GPU (GRA111/GRA112) + 256GB SSD
- MIL-DTL-38999 connectors (3x GigE, 4x USB, 2x DVI, 3x VGA, video-in, power)
- Line-replace existing GRA111-based units with GRA112

http://defense.ge-ip.com/products/2077
360° Situational Awareness

Distributed Aperture Sensor (DAS)

- Stream & mosaic up to 20 gigabits of EO/IR video
- Display on four crew stations simultaneously (via DVI), all with independent viewing controls (pan, zoom, PiP)
- H.264 compression, metadata tagging, and on-demand VoE streaming for distribution to LAN/WAN and NAS
- Human/vehicle detectors, optical flow, MTI, trackers in CUDA
IPS5100

Deployable 20GigE Ingest + Dual-GPU Rugged System

- 5-slot 3U VPX prepackaged system
  - 2x NVIDIA CUDA-enabled GPU’s (GRA111/GRA112)
  - Intel i7 Sandybridge SBC (SBC324)
  - Tilera 64-core 20GigE network processor (GFG500)
Rugged 10” and 15” CUDA-enabled Intelligent Displays

- Intel SBC + NVIDIA GPU + 256GB SSD (AES-128 encryption)
- Dual 10GBASE-SR ethernet ports
- 4x VIDEO IN (RS-170, NTSC, PAL)
- CANbus / MilCAN and MIL-STD-1553
- MIL-STD-3009 NVIS compatibility
- 5-wire resistive touchscreen
- 30 user-defined bezel keys
360° Situational Awareness

Distributed Aperture Sensor (DAS)

Ingest + Distribution

GBX460 10GigE switch

10GBASE-SR ×N

10GBASE-SR

DVI ×4

WAN

H.264 VoE

NAS
Rugged CUDA Radar Applications

Synthetic Aperture Radar (SAR)

• Utilize Radon transform and tomographic reconstruction to form 2D/3D image of scene from dense set of beams sampled at multiple angles of azimuth and elevation

• Backprojection algorithm incrementally builds the value of each pixel in the output image by integrating the contribution of each beam to that pixel
  • Backprojection has $O(N^{N+1})$ complexity and is the most computationally intensive aspect of SAR processing
  • Exploit inherent parallelism by running on GPU

• Acceleration using CUDA: **220x faster than CPU**
  • Reduced SWaP of SAR processor - deploy onto smaller platforms
  • Increased image resolution and SAR sensor bandwidth
  • Process in realtime, instead of post-collection offload
Rugged CUDA Radar Applications

Ground Moving Target Indicator (GMTI)

• Utilize *Space-Time Adaptive Processing* (STAP) to detect small or slow-moving targets in data with high levels of interference from clutter or jamming.

• STAP estimates the interference’s covariance matrix and filter weights locally for each Doppler/range element.
  
  • Localized adaptivity is very effective at suppressing clutter, but is also very computationally intensive
  
  • Need to estimate covariances, then run Cholesky decomposition to solve filtering weights for each element

• Acceleration using CUDA: **135x faster than CPU**
  
  • Reduced SWaP of SAR processor - deploy onto smaller platforms
  
  • Increased DoF – channels, Doppler pulses, range samples
  
  • Combined SAR/GMTI knowledge-aided modes
6U VPX VITA-48.5 HPEC System

13.6 TERAFLOPS
17 SLOTS
6U VPX VITA-48.5 HPEC System

Top Cover

Air Inlet

I/O Panel

17 VITA 48.5 6U VPX Cards

Air Outlet

Bottom Cover

Power Supply
Conclusion

- **3U VPX GRA112**  rugged 384-core Kepler EXK107 GPU with 2GB GDDR5
- **6U VPX IPN+NPN**  upgrades launched later this year

**GPUDirect Peer-to-Peer**  ≥ 25X latency reduction, 0% CPU. Flexible, free, and open.

Visit GE @ booth 16