High Performance GPGPU Computer for Embedded Systems

Author: Dan Mor, Aitech Product Manager

September 2015
Contents

1. Introduction ........................................................................................................................................ 3
2. Existing Challenges in Modern Embedded Systems ................................................................. 3
   2.1. Not Enough Calculation Power ........................................................................................... 3
   2.2. Power Consumption, Form Factor and Heat Dissipation .................................................. 3
   2.3. Generic Approach .............................................................................................................. 4
3. GPGPU Benefits in Embedded Systems .................................................................................. 4
   3.1. GPU Accelerated Computing .............................................................................................. 4
   3.2. CUDA and OpenCL Frameworks - Definitions .................................................................... 5
   3.3. CUDA Architecture and Processing Flow ........................................................................ 5
   3.4. CUDA Benchmark on Embedded System ......................................................................... 7
       3.4.1. “Nbody” - CUDA N-Body Simulation ................................................................. 7
       3.4.2. SobolQRNG - Sobol Quasirandom Number Generator ........................................... 9
   3.5. Switchable Graphics ............................................................................................................. 10
       3.5.1. Why we need switchable graphics in embedded systems? ...................................... 10
       3.5.2. NVIDIA® Optimus™ Technology ............................................................................. 10
       3.5.3. AMD Enduro™ Technology ..................................................................................... 10
4. Use Cases ........................................................................................................................................ 11
5. Aitech RediBuilt™ Approach ................................................................................................. 11
   5.1. Aitech A191 GPGPU System .............................................................................................. 12
      CPU ........................................................................................................................................ 12
      Choice of GPU ..................................................................................................................... 12
1. Introduction

Providing High Performance Embedded Computer (HPEC) using General Purpose Computation on Graphic Processor Units (GPGPU).

In today’s fast growing computer industry, embedded systems must advance side by side with the latest computing technology. The software applications used by embedded systems (civilian or military) are becoming more complex every day and demand much more computation power from the hardware.

Therefore, a deeper understanding of the difficulties in modern embedded systems and better knowledge of existing technical solutions will help in choosing the best available product which will deliver the best performance for civilian and military markets.

2. Existing Challenges in Modern Embedded Systems

Below are the most common challenges which we (Architects, Product Managers and Engineers) are faced with on a daily basis.

2.1. Not Enough Calculation Power

While trying to reuse existing software applications, we are constantly adding new features and implementing solutions to new requirements. The code is becoming more and more complex; the application is becoming CPU “hungry”, so eventually we have to deal with:

- Complex CPU Load Balancing – we are skating on thin ice in order to satisfy our SW application demands
- CPU choking – eventually the OS’ response is so slow that we need to change the entire SW architecture and find a very thin line between acceptable response and getting the job done
- Upgrading and Over-clocking – Another solution for adding additional computation power

The latest trend for the software companies is to use gaming libraries to reduce development time and “squeeze” the maximum performance per watt from the computation engine.

All the reasons above are pushing embedded systems providers to consider using a GPU instead of a CPU as a main computing engine.

2.2. Power Consumption, Form Factor and Heat Dissipation

As described in the previous section, once we get desperate for processing power, we end up buying more powerful hardware or over-clocking existing equipment and this is leads to increased power consumption.
In the mobile computer industry, we can see that the form factor plays a very important role in a product’s success. The footprint is constantly decreasing (although lately we see that the mobile market has increased form factor, as with the latest phablets for example; anyway we know it behaves as a wave form, so once increased it will be decreased again). The embedded industry is not an exception; the developers and manufacturers of the embedded market are trying to keep up with the consumer market by decreasing the footprint of new systems.

Increased power consumption and decreased footprint create a new problem – heat dissipation. Imagine yourself taking a powerful gaming PC and putting it in a small factor “toaster” enclosure, you will end up with toast instead of doing high performance calculations job!

2.3. Generic Approach
While chasing after the customer and trying to meet tight deadlines and milestones, the embedded systems manufacturer often finds himself compromising a generic approach for an ad-hoc solution. How many times have we gotten a system for integration only to find that one or more modules of this system have a proprietary interface? Now, what if the opposite occurs? How smooth do you think the integration process would be if all modules supported a known generic interface? You’re right; it would work like a charm.

Zero Time Porting – Develop the entire solution on a Notebook or PC based station with the same GPU architecture and then just copy it to embedded system.

3. GPGPU Benefits in Embedded Systems

3.1. GPU Accelerated Computing
GPU-accelerated computing is the use of a graphics processing unit (GPU) together with a central processing unit (CPU) to accelerate applications.

As presented earlier, the difficulties facing modern embedded systems lead to one conclusion: if we are using only a CPU as a main computing engine, eventually it will choke. What if we could offload some of the computation intensive portion of application to the GPU while the rest of the application runs on the CPU?

This is exactly what GPU Accelerated Computing does—it offloads some of the computation-intensive portion of the application.
So, how come the GPU is capable of doing it faster than the CPU?

The GPU has evolved into an extremely flexible and powerful processor because of:

- Programmability
- Precision (Floating Point)
- Performance - thousands of cores to process parallel workloads
- GPUs are getting faster because the giant gaming industry is pushing it

NVIDIA® explains the difference very well:

*A simple way to understand the difference between a CPU and GPU is to compare how they process tasks. A CPU consists of a few cores optimized for sequential serial processing while a GPU has a massively parallel architecture consisting of thousands of smaller, more efficient cores designed for handling multiple tasks simultaneously.*

See more at: [http://www.nvidia.com/object/what-is-gpu-computing.html#sthash.TxH6NhBD.dpuf](http://www.nvidia.com/object/what-is-gpu-computing.html#sthash.TxH6NhBD.dpuf)

### 3.2. CUDA and OpenCL Frameworks - Definitions

**CUDA** (as defined by Wikipedia), which stands for Compute Unified Device Architecture, is a parallel computing platform and application programming interface (API) model created by NVIDIA®. It allows software developers to use a CUDA-enabled graphics processing unit (GPU) for general purpose processing – an approach known as GPGPU.

**OpenCL** (as defined by Wikipedia) - Open Computing Language (OpenCL) is a framework for writing programs that execute across heterogeneous platforms consisting of central processing units (CPUs), graphics processing units (GPUs), digital signal processors (DSPs), field-programmable gate arrays (FPGAs) and other processors.

### 3.3. CUDA Architecture and Processing Flow

Unlike CPUs, GPUs have a parallel throughput architecture that executes many concurrent threads, rather than only a single thread.
In the computer industry, GPUs are used not only for graphics rendering but also in game physics calculations (physical effects such as debris, smoke, fire and fluids).

CUDA has also been used to accelerate non-graphical applications in computational applications which use mathematical algorithms.

Example of CUDA processing flow

- Copy data from main mem to GPU mem
- CPU instructs the GPU to process the data
- GPU executes in parallel using each core
- Copy the result from GPU mem to main mem


Since we can see in the processing flow that both CPU and GPU are involved, and working simultaneously, we can say that:

The faster this tandem (CPU and GPU) will be the better computing performance we will get.
3.4. CUDA Benchmark on Embedded System
In order to compare computing performance between CPU and GPU, we will use NVIDIA® CUDA SDK 6.5. We will run the same algorithm, once on the CPU and then on the GPU and then we will compare the results.

We want to do a “fair” comparison, so we picked up one of the latest Intel platforms – Haswell together with Nvidia GTX 770M graphic cards.

Both boards and HPEC GPGPU system were taken from Aitech Systems for the purpose of the test.

Aitech Systems web site: [www.rugged.com](http://www.rugged.com)

Here is the hardware involved in this test:

- CPU - Intel Haswell i7 2.4GHz SBC (C873) (Manufacturer Aitech Systems)
- GPU – Nvidia GTX 770M GPGPU board (C530) (Manufacturer Aitech Systems)
- HPEC GPGPU System – A191 (Manufacturer Aitech Systems)

3.4.1. “Nbody” - CUDA N-Body Simulation
This example demonstrates efficient all-pairs simulation of a gravitational n-body simulation in CUDA.

CUDA API:

```c
cudaGLSetGLDevice, cudaGraphicsMapResources,
cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer,
```

```c
cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer,
cudaGraphicsUnregisterResource
```

Key Concepts:

*Graphics Interop, Data Parallel Algorithms, Physically-Based Simulation*

![Figure 1: Running “nbody” on CPU](image1)

![Figure 2: Running “nbody” on GPU](image2)
Imagine offloading heavy duty calculations to the GPU and freeing the CPU for other mission critical tasks.

Now, we will increase the numbers of simulated n-bodies and compare the results between CPU and GPU.

<table>
<thead>
<tr>
<th>n-body number</th>
<th>CPU</th>
<th>GPU</th>
<th>FACTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>4096</td>
<td>2080.907</td>
<td>6.852</td>
<td>X 300</td>
</tr>
<tr>
<td>Interactions per second [billion ips]</td>
<td>0.081</td>
<td>24.486</td>
<td></td>
</tr>
<tr>
<td>Single-precision GFLOP/s at 20 flops per interaction</td>
<td>1.612</td>
<td>489.728</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n-body number</th>
<th>CPU</th>
<th>GPU</th>
<th>FACTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>8192</td>
<td>8318.286</td>
<td>27.261</td>
<td>X 300</td>
</tr>
<tr>
<td>Interactions per second [billion ips]</td>
<td>0.081</td>
<td>24.617</td>
<td></td>
</tr>
<tr>
<td>Single-precision GFLOP/s at 20 flops per interaction</td>
<td>1.614</td>
<td>492.342</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n-body number</th>
<th>CPU</th>
<th>GPU</th>
<th>FACTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>16384</td>
<td>33301.543</td>
<td>92.930</td>
<td>X 350</td>
</tr>
<tr>
<td>Interactions per second [billion ips]</td>
<td>0.081</td>
<td>28.886</td>
<td></td>
</tr>
<tr>
<td>Single-precision GFLOP/s at 20 flops per interaction</td>
<td>1.612</td>
<td>577.716</td>
<td></td>
</tr>
</tbody>
</table>
The graph below shows iteration latency of simulated defined numbers of N-Body between CPU and GPU.

![Iteration Latency Graph](image)

**Figure 3: Iteration Latency between CPU and GPU**

*It takes the GPU 93 msec and the CPU 16 sec to perform a simulation on 16384 numbers.*

### 3.4.2. SobolQRNG - Sobol Quasirandom Number Generator

This example implements the Sobol Quasirandom Sequence Generator. Comparing this algorithm between the CPU and GPU, how many samples may be involved per second during the calculation process?

Number of vectors = 100000

Number of dimensions = 100

<table>
<thead>
<tr>
<th>Gsamples/s</th>
<th>CPU</th>
<th>GPU</th>
<th>FACTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.230959</td>
<td>9.13413</td>
<td>X 40</td>
</tr>
</tbody>
</table>
3.5. Switchable Graphics

3.5.1. Why we need switchable graphics in embedded systems?
Using switchable graphics in embedded systems, we can actually enjoy both worlds: decreased power consumption and performance. When needed, we can offload heavy duty tasks to the powerful GPU. On the other hand, when not needed, we can benefit from internal IGP and save a lot of power improving system power consumption and heat dissipation. So, what we have is:

- Full performance benefits of a discrete GPU
- Low power consumption of an integrated graphics solution
- Automatically optimizes embedded systems to offer the best performance or power consumption

3.5.2. NVIDIA® Optimus™ Technology
NVIDIA® Optimus™ technology intelligently optimizes embedded systems, providing the outstanding graphics performance you need, when you need it, all the while optimizing power consumption.

NVIDIA® white paper explanation of Optimus™ technology:

*Using NVIDIA’s Optimus technology, when the discrete GPU is handling all the rendering duties, the final image output to the display is still handled by the Intel integrated graphics processor (IGP). In effect, the IGP is only being used as a simple display controller, resulting in a seamless, flicker-free experience with no need to reboot.*

*When less critical or less demanding applications are run, the discrete GPU is powered off and the Intel IGP handles both rendering and display calls to conserve power and provide the highest possible battery life.*


3.5.3. AMD Enduro™ Technology
Optimizes your embedded system to give you an instant boost in graphics performance when you need it, consuming virtually zero watts of power when you don’t.
This is how AMD explains the key benefits:

- **Seamless switch between your APU or integrated graphics and your AMD FirePro™ or AMD Radeon™ graphics based on graphics workload to allow you to get longer battery life and outstanding performance.**

- **Intelligent implementation in AMD Catalyst™ drivers allow the system to find the best graphics option for your needs or enables you to configure it yourself for the best performance possible.**


4. **Use Cases**
There are so many use cases for GPGPU technology. Actually, any application which involves many mathematical calculations can be a very good candidate for this technology.

- Image Processing – enemy detection, vehicle detection, missile guidance, obstacle detection, etc.
- Radar
- Sonar
- Video encoding and decoding (NTSC/PAL to H.264)
- Data encryption/decryption
- Database queries
- Motion Detection
- Video Stabilization

We can benefit from a GPGPU by developing our own application using CUDA and OpenCL high-level languages or we can run industrial GPU accelerated applications which are already optimized.

Hundreds of industry-leading applications are already GPU-accelerated. Find out if the applications you use are GPU-accelerated.


5. **Aitech RediBuilt™ Approach**
After seeing all these benefits from GPGPU and Switchable Graphics, are you wondering if there is an embedded system which embodies all this wonderful technology?
Aitech’s A191 GPGPU subsystem is a true rugged COTS High Performance Embedded Computer (HPEC). Assembled, tested, and qualified, the A191’s integrated 4-th Gen. (Haswell) Corei7 Intel SBC and GPGPU (NVIDIA or AMD) provide an out-of-the-box solution to meet many of today’s military and airborne computing requirements. Designed using proven Aitech technology, this GPGPU subsystem is a fully integrated, ready to use product.

5.1. Aitech A191 GPGPU System

- GPGPU Based High Performance Embedded Computer (HPEC)
- Rugged Computer for Military and other Harsh Environment Applications
- Combination of High Performance Multi-core CPU and GPU
  - 4th Generation Intel® Core™ i7 CPU, Quad Core @ 2.4 GHz
- Choice of GPU
  - NVIDIA® GeForce® GTX 770M
  - AMD Radeon™ HD 7970M
- Fully Integrated and Tested – Ready to Use
- Front Panel I/O Module with D38999 Connectors
- 3U VPX Architecture
- Compact and Lightweight
- Two External Cooling Configurations
  - Convection and Radiation Cooling by Fins
  - Cold Plate-Cooling
- Fully Sealed Faraday Cage and Complete EMI/RFI Filtering
- Environmentally Sealed
- Embedded Frame Grabber
- Up to 1 TB Flash Storage
- Video Outputs
  - 2 x DVI/HDMI
  - 2 x RGBHV
- Video Inputs
  - 2 x SD-SDI + 4 x Composite or
  - 8 x Composite
- Communications I/O
  - 2 x Serial Ports
  - 2 x USB Ports
  - 2 x Gigabit Ethernet Ports
## Benefits

<table>
<thead>
<tr>
<th>Benefits</th>
<th>Aitech A191 System</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPGPU</td>
<td>✓</td>
</tr>
<tr>
<td>Plenty of calculation power</td>
<td>✓</td>
</tr>
<tr>
<td>CUDA</td>
<td>✓</td>
</tr>
<tr>
<td>Switchable Graphic Support</td>
<td>✓</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>✓</td>
</tr>
<tr>
<td>Form Factor and Heat Dissipation</td>
<td>✓</td>
</tr>
<tr>
<td>Generic Approach</td>
<td>✓</td>
</tr>
<tr>
<td>Zero Time Porting</td>
<td>✓</td>
</tr>
</tbody>
</table>

See more:

**Aitech A191 Web Site:**


**Aitech A191 on You Tube:**

[https://www.youtube.com/watch?v=dd_IZDEOkaU](https://www.youtube.com/watch?v=dd_IZDEOkaU)