AES on Graphics Processing Units

AES Encryption Implementation and Analysis on Commodity Graphics Processing Units

Trinity College Dublin

Ireland

Owen Harrison, John Waldron

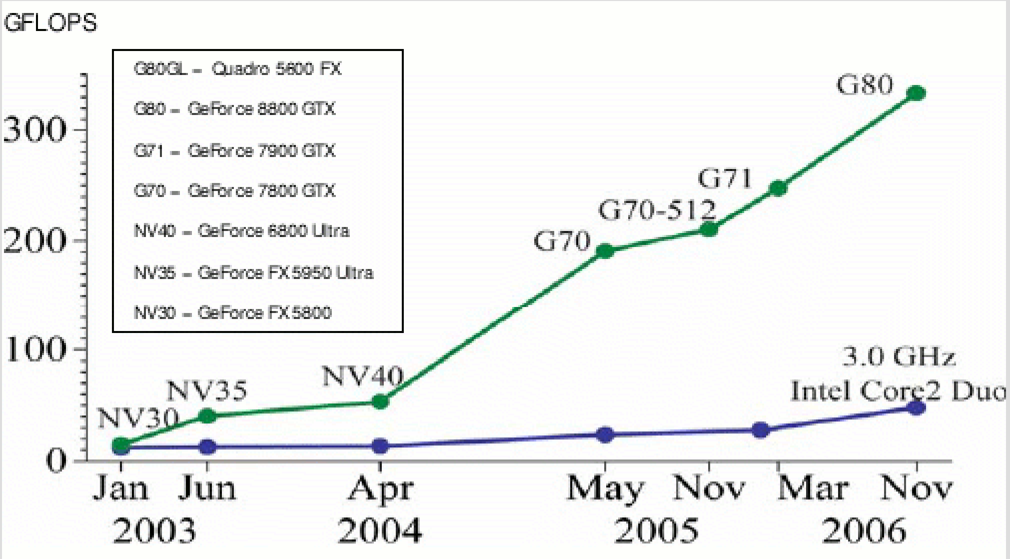
Presentation

- Motivational Background.
- GPU and AES Motivation.
- GPU Programming Interface.
- AES and GPU.
- Encryption Throughput on GPUs.
- GPU as Co-processor.
- Latest GPUs.

Research Background

- Reducing SW Dev Overhead on Highly Parallel Heterogeneous Compute Resources
  - Example Architectures
    - CPU + GPU (GPGPU)
    - CPU + Cell, or Cell alone.
    - FPGA (PCIe Boards / Opteron Socket)
    - Intel TeraScale
    - AMD Fusion
- Focal Applications for research:
  - AES
  - Biotech - docking.

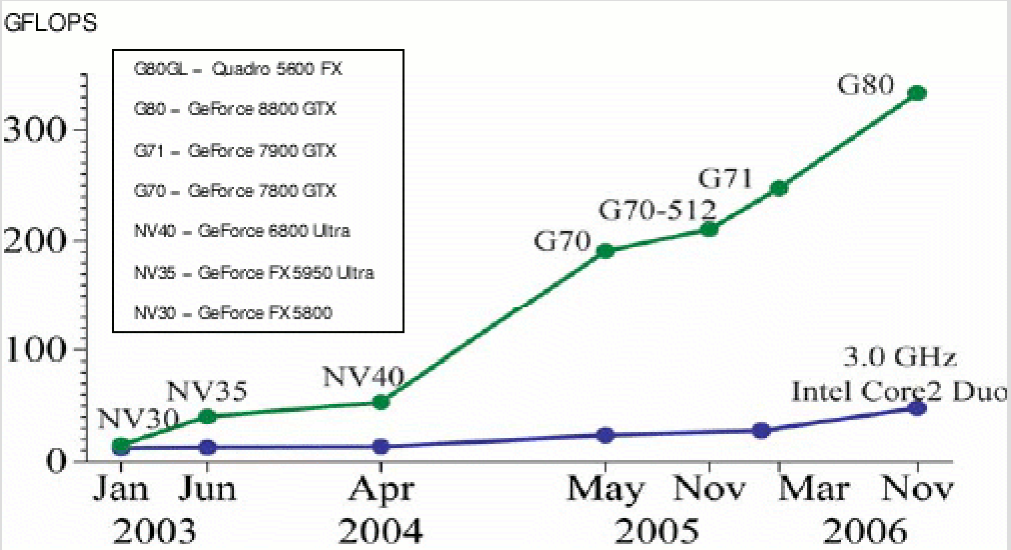
CPU vs GPU



Floating-Point Operations per Second for the CPU and GPU

- Reasons For Highly Parallel Approach:
  - Reduced returns from pipeline deepening.
  - Power/heat considerations with increased clock speeds.
  - Difficultly in ILP.
  - Highly parallel design moves these problems to the developer.

CPU vs GPU



Floating-Point Operations per Second for the CPU and GPU

- Heterogeneous = better transistor expenditure for tasks.

- Reasons For Highly Parallel Approach:
  - Reduced returns from pipeline deepening.
  - Power/heat considerations with increased clock speeds.
  - Difficultly in ILP.
  - Highly parallel design moves these problems to the developer.

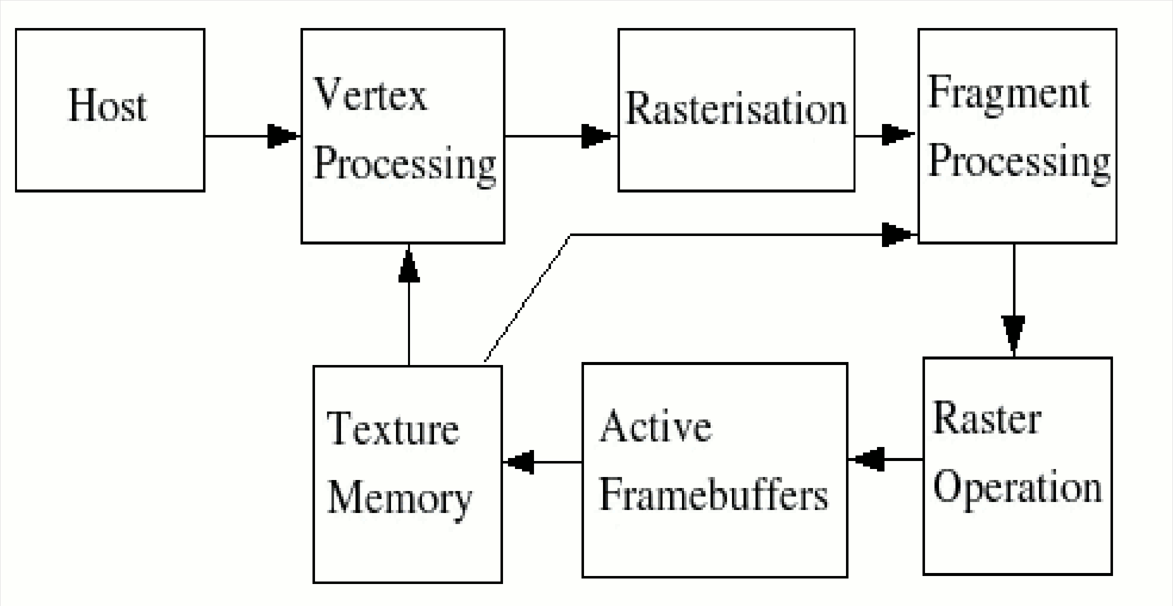
GPU & AES Motivation

- CPU and GPU model converging into some form of heterogeneous architecture. Good to research on likely future compute resources.
- GPU normally highly underutilised, co-processor.
- Investigate if cheaper per byte enc/dec for encryption/streaming farms.
- Reduced trusted computing base for encrypted visual applications.
- Personal reasons – good example parallelisable unexplored application for main research focal point.

GPU Programming Interface

- OpenGL. Advantages: only cross OS, cross graphics card vendors, cross gpu generations, vendor support. Disadvantages: api requires graphical knowledge.
- API used in presented work, though CUDA and CTM are aiming to make GPU programming more mainstream.

OpenGL Pipeline



GPGPU basic idea is to create a 2D quadrilateral and an equivalently sized 2D texture which acts as the input data. The output data is written to the active framebuffer after computation by the fragment processors.

DX9 GPUs and AES

- Data Throughput – PCIe, transfer tool.
- Texture Lookups (memory footprint minimisation) – ie. restricted and non uniform memory layout.
- Gather and Scatter.
- XOR operator - ROP only - restrictive.
- Free Swizzle (useful for free ROTs).
- Parallel Modes of Operation only.
- Floating point only fragment processor.
- OpenGL/DirectX graphics API only.

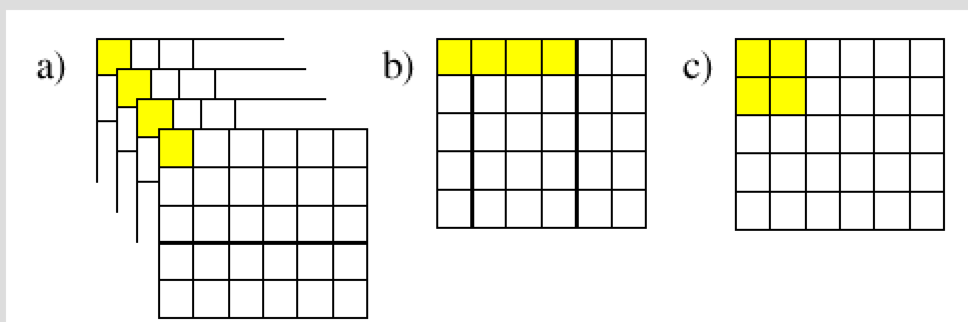
DX9 Cards - XOR

- 8 bit simulated using table lookups.
- 4 bit table lookups with wrapping + multiplies.
- ROP xor with render pass per xor.
- Results in MBytes/s.

	GeForce 6600GT			GeForce 7900GT			CPU	
	8-bit	4-bit	Native	8-bit	4-bit	Native	8-bit	32-bit
W/O Round Trip	181.26	1068.0	4160	672.0	3510	12249	118.29	437.18
With Round Trip	79.61	126.7	141.0	334.83	472.7	475.4		

DX9 AES

- Input: Each column represented as an RGBA 4 8 bit component texel. Output: 4 texture (MRT - lack of scatter)
- 3 Gather techniques:
  - Multi Texture Input, Single Texture H & S gather.



- noROT vs ROT (5 table vs 2 table + rots).

$$
\begin{aligned}
e_j &= T_0[a_{(0,j)}] \oplus T_1[a_{(1,j-c1)}] \oplus \\
&\quad T_2[a_{(2,j-c2)}] \oplus T_3[a_{(3,j-c3)}] \oplus k_j \; .
\end{aligned}
$$

$$
\begin{aligned}
e_j &= k_j \oplus T_0[a_{(0,j)}] \oplus Rot(T_0[a_{(1,j-c1)}] \oplus \\
&\quad Rot(T_0[a_{(2,j-c2)}] \oplus Rot(T_0[a_{(3,j-c3)}]))) \; .
\end{aligned}
$$

DX9 AES

- AES Approach 1: 8 bit simulated xor, 3 gathers approaches, noROT, ROT.
- AES approach 2: 4 bit simulated xor, same as approach 1.
- AES approach 3: ROP xor. Multi input gather only(no scatter/multi passes per round thus output and input textures as same type). Memory read footprint reduction:

$$e_0 = k_0 \oplus T_0[a_{(0,0)}] \oplus Rot(T_0[a_{(1,1)}]) \oplus Rot2(T_0[a_{(2,2)}]) \oplus Rot3(T_0[a_{(3,3)}])$$

$$e_1 = k_1 \oplus Rot3(T_0[a_{(3,0)}]) \oplus T_0[a_{(0,1)}] \oplus Rot(T_0[a_{(1,2)}]) \oplus Rot2(T_0[a_{(2,3)}])$$

DX9 AES Results

- Results of AES implementations in Mbytes/s

Gather Technique		GeForce 6600GT			GeForce 7900GT		
		8-bit	4-bit	Native	8-bit	4-bit	Native
Multi Input	ROT	6.24	11.47	45.15	25.86	39.23	108.86
	noROT	6.11	11.19	44.89	25.71	39.01	108.55
Single Input Sgather	ROT	6.22	11.40	N/A	26.06	39.18	N/A
	noROT	6.11	11.22	N/A	25.92	39.12	N/A
Single Input Hgather	ROT	6.20	11.41	N/A	25.99	39.16	N/A
	noROT	6.15	11.30	N/A	25.69	39.08	N/A

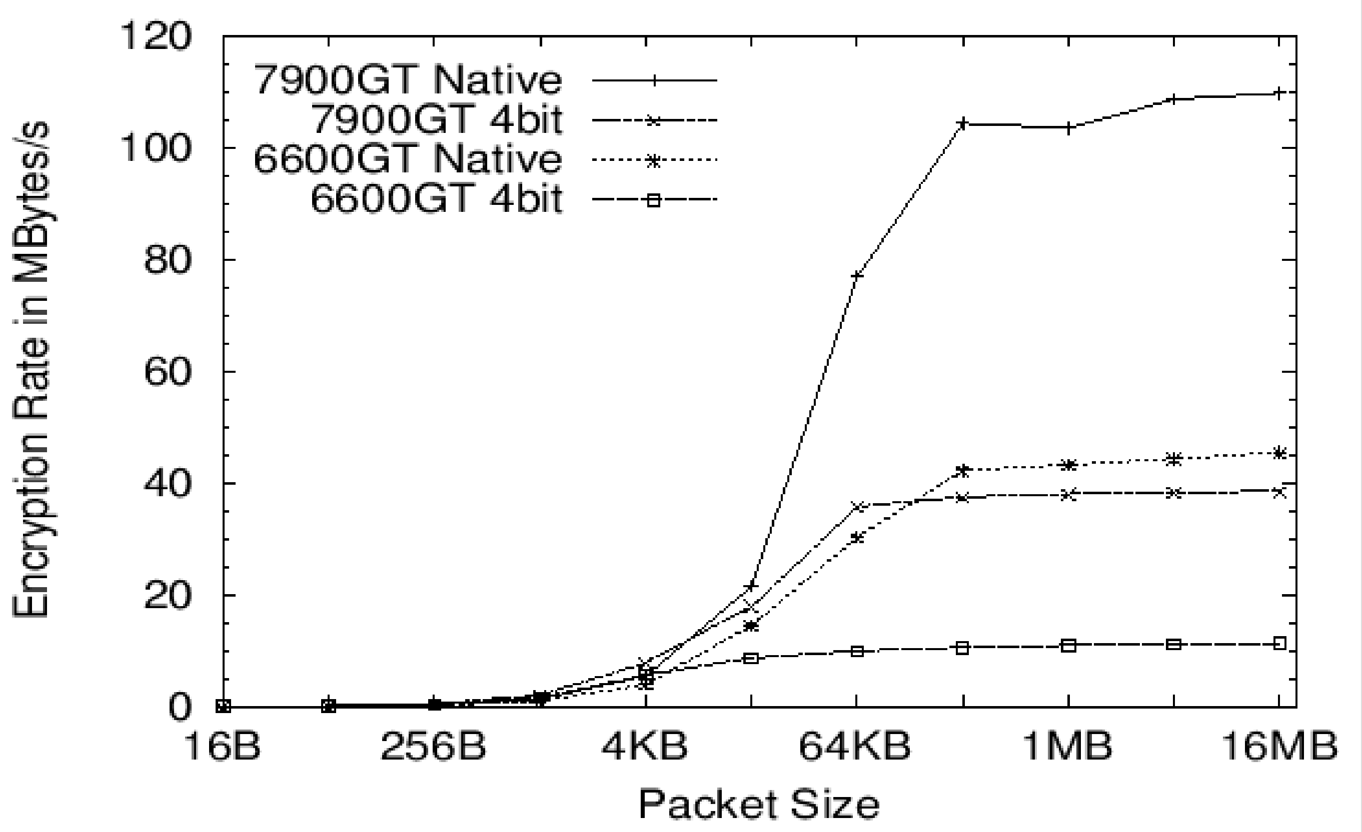
- ROP base XOR proves best performance even though the extra passes overhead. Main bottleneck is non coherent memory access.
- ROT (single table) is slightly better than noROT.

Throughput

- Different work unit sizes and its effect on throughput.
- Small work units = high CPU-GPU interactions = higher inefficiencies. Lack of IO pipelining doesn't help (future gpus).
- Highly parallel systems naturally need enough data to keep processing elements busy.

...Throughput

- Effects of packet size variation on encryption.



Co-processor

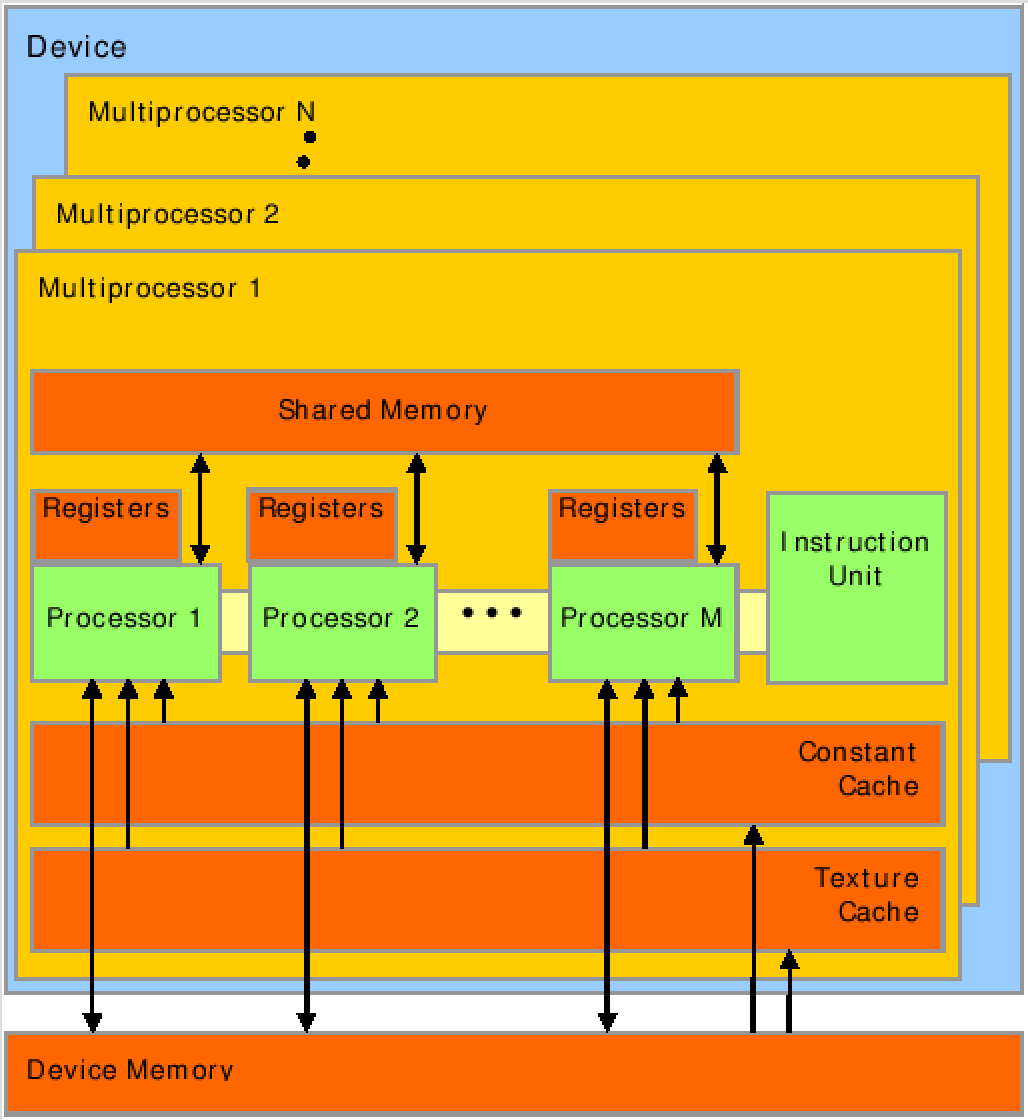
- Linux reports 100% CPU usage during encryption runs. Co-processor?
- Not a true reflection. % CPU Idle Time for GPU enc shown below:

Gather Technique		GeForce 6600GT			GeForce 7900GT		
		8-bit	4-bit	Native	8-bit	4-bit	Native
Multi Input	ROT	96.69%	94.19%	86.75%	87.42%	90.61%	74.84%
	noROT	95.96%	94.10%	85.98%	88.79%	89.79%	74.57%
Single Input SGather	ROT	99.18%	96.75%	N/A	88.06%	93.54%	N/A
	noROT	98.24%	95.32%	N/A	88.65%	92.34%	N/A
Single Input HGather	ROT	98.76%	96.59%	N/A	88.70%	93.02%	N/A
	noROT	98.56%	96.46%	N/A	88.49%	93.34%	N/A

Recent DX10 GPUs

- Massive improvement on previous models in terms of GPGPU.
- Native XORs support.
- Native 32bit Integer support.
- Shaders consolidated in hardware = more processors for general purpose processing.
- API – CUDA, CTI more suited to general purpose processing.
- Throughput and memory footprint still an issue.
- Still only suits applications with high compute intensity vs IO, stream like IO patterns.

Latest GPU Architecture Example



A set of SIMD multiprocessors with on-chip shared memory.

- Nvidia G80 – AES @ > 4Gbps.
- Array of SIMD Processors.
- ~100GB/s Device Memory Bandwidth.
- Peak ~350GFlops.
- Intel QC - 50GFlops.
- IBM Cell - 250GFlops.
- AMD R600 – 450GFlops.
- G92 – 1TeraFlop.
- CPU and GPU are moving towards each other.
- Fusion/Terascale.

El Final...

- Many thanks.