Chessboard Domination on Programmable Graphics Hardware [CDGPU2006]

"First algorithm to determine the minimum domination number of a chessboard graph using the GPU"

Rick van der Zwet

LIACS - Leiden University

Seminar Combinatorial Algorithms, 2010

Outline

1. Minimum domination set
   - Domination set
   - GPU Inner Workings
2. Algoritm
   - Computing the piece configuration
   - Rendered in Framebuffer
   - Determine Domination (e.g. Mark solution)
3. GPU Optimalizations
4. Results
   - Main Results
   - Discussion

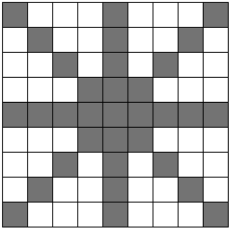
Domination set

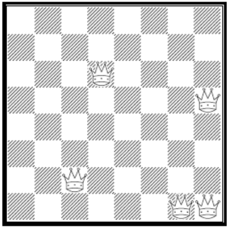
Capture them all

- Use the least amount of items to cover a whole board
- Item based characteristics made whole set

Queen lower bound

- $y(Q_n) \geq \frac{n-1}{2}$, $n \geq 1$[1]
- Every square either contains a queen, or can be reached by a queen (e.g. least amount of pieces required)



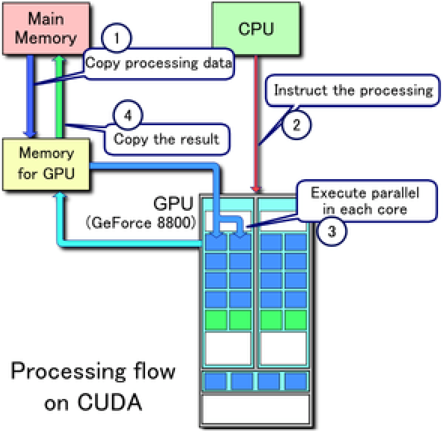


Board Layout

- *streams:* pipelines available on the GPU - a collection of records requiring similar computation.
- *kernel:* function that is applied to each element of a stream.

In the GPU streaming model, textures, geometry, and the framebuffer are seen as streams while vertex and fragment programs are seen as kernels.

Outlined Figure



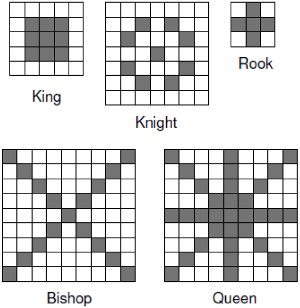
Basic Algoritm

```
finished=false
do
..computes a piece configuration which may be a minimally dominating set
..Rendered in the framebuffer
..if (All pixels are marked)
....finished=true
..fi
while (finished=false)
```

Method

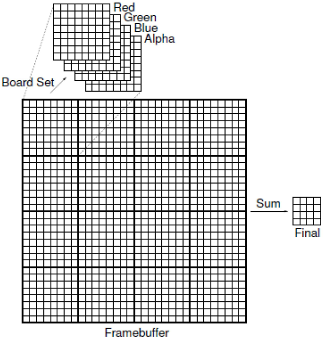
- Exhaustive manner
- Piece configuration stored on the CPU as linked links
- Lower bound and Upper bound is respected

- GPU supports textures, every piece is a texture
- Render points on the CPU and offload to the GPU to map texture on specific place



- Simple approch
- Sum all pixels of $n * n$ board and match if $sum = n * n$

Colour Channels



- GPU is able to process all colours at the times

Grid Framebuffer

- GPU has many CPU's called kernels
- Each kernel can process it's own little block of information
- Putting multiple possible solutions in one bloc

Conclusions and Future Work

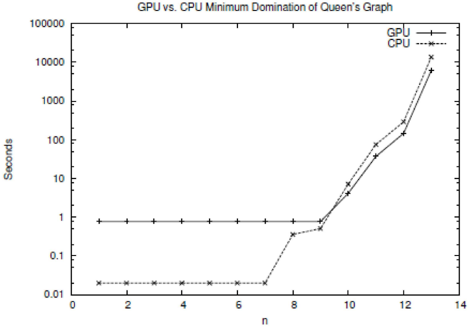


Figure: Execution times (log scale) of CPU and GPU based minimum domination implementations computing $y(Q_n)$. As n increases, the GPU's speed advantage over the CPU become more evident.

Conclusions and Future Work [2]

- Domination texture good mapping between CPU world and GPU world
- Flexible texture definition without any impact

Discussion

- No significant speedup, claim that $n \geq 13$ GPU is *'much'* faster
- No scaleable

Summary

- First GPU algoritm for solving minimum domination described at the time
- Using texture mapping to build bridges between the CPU world and GPU world

- Outlook
  - Make it scale so its decision algoritms is much smarter
  - Build a framework to allow easy and proper testing for various combinations

- E. J. Cockayne
  Chessboard domination problems
  Discrete Math., 86:1320, 1990.

- Nathan Courni
  Chessboard Domination on Programmable Graphics Hardware
  ACM SE'06 March 10-ï¿œ12, 2006. Melbourne, Florida, USA