# Mandelbrot Properties

* IMAGE WIDTH = 8982
* IMAGE HEIGHT = 4602
* NUM ITERATIONS = 128
* ZOOM FACTOR = 0.95
* NUM PICTURES = 200

# GPU Porperties

* NVIDIA GT 650m
* Clock rates
  + Core: 850 MHz
  + Memory: 900 MHz
  + DDR: 7200 MHz
* SM: 2
* Memory size: 2048 MB DDR3

# Mandelbrot Version Speedup Comparison

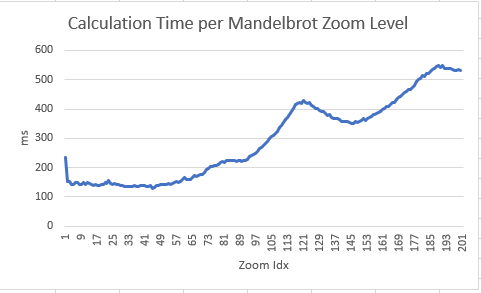
|  |  |  |
| --- | --- | --- |
| Version | ms | Speedup |
| 0 | 58847 |  |
| 1 | 33868 | 1.74 |
| 2 | 27732 | 1.22 |
| 3 | 17593 | 1.58 |
| 4 | 16174 | 1.09 |
| 5 | 15760 | 1.03 |
| 6 | 13946 | 1.13 |

# Mandelbrot Version (Unoptimized)

## Properties

Allocate iterations array on host and allocate iterations array on device. Device sets iteration to iteration array for each pixel. Device iteration array is copied back to host iteration array after completion. After copying CPU colors Mandelbrot by iterating through iterations array, lookup color match and assigns it to the according bmp pixel. BiG & TiB are set to (8x8).

## Results



# Mandelbrot Version 1 (incr. occupancy)

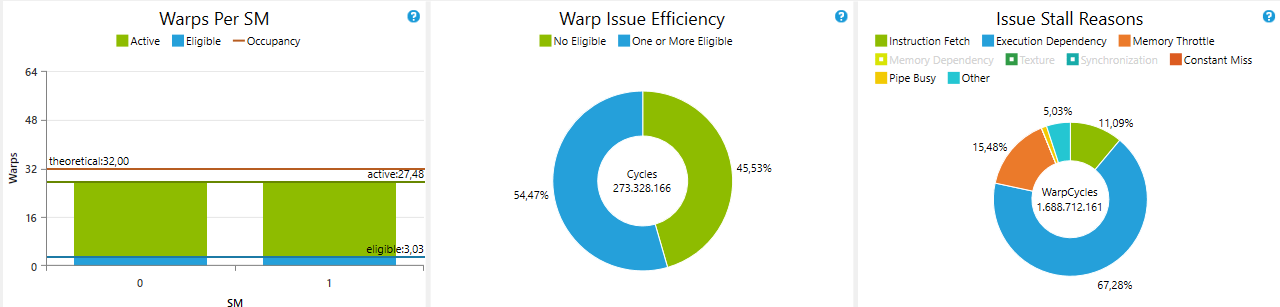
## Properties

Optimize blocks in grid & threads in block to increase occupancy. (8x8,32x4,16x16)

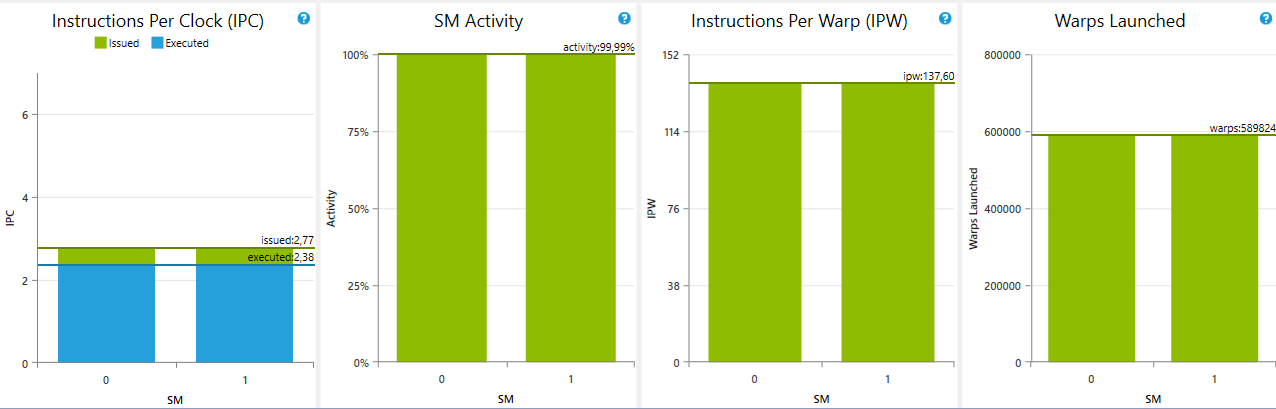
## 8x8

### Occupancy

### Issue Efficiency

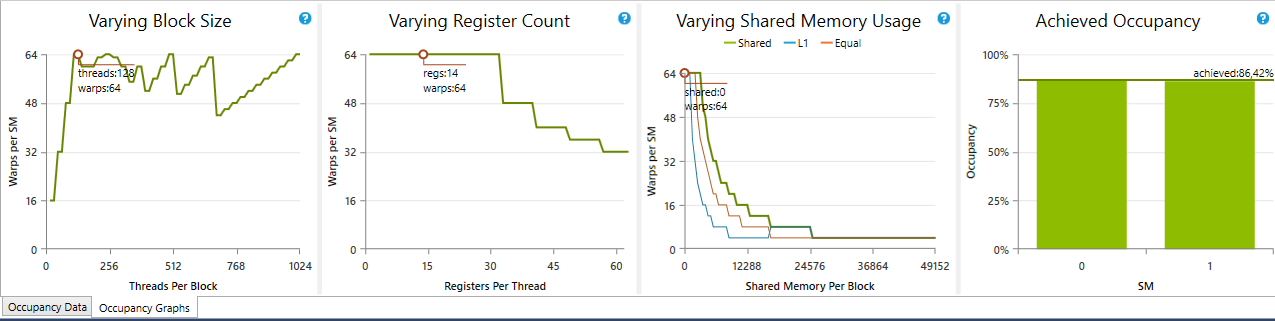


### Instruction Statistics

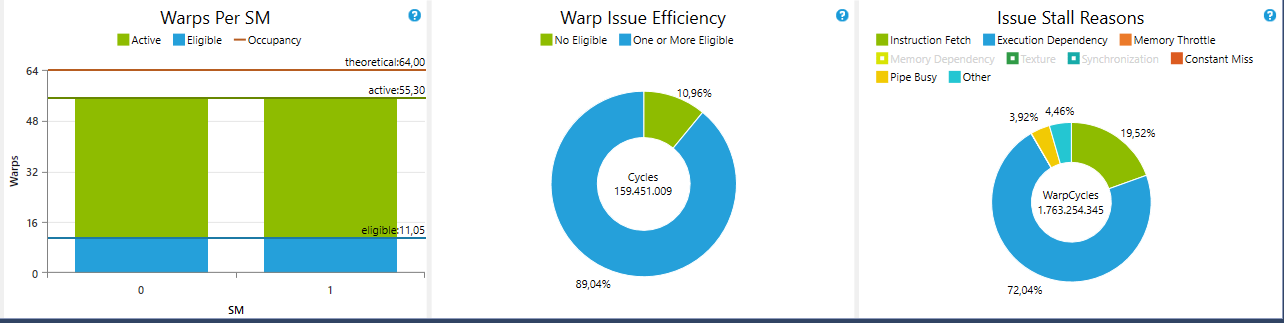
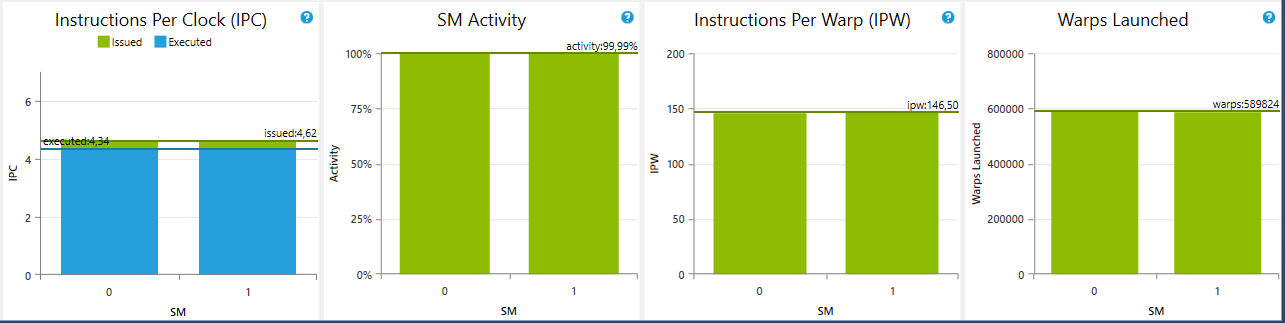


## 32x4

### Occupancy



### Issue Efficiency

Instruction Statistics

## Results

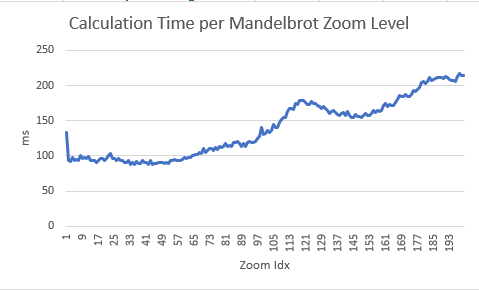
## 

# Mandelbrot Version 2

## Properties

Use of constant memory in device for fast color lookup. Color map is generated on the CPU and copied to constant memory space with cudaMemCpyToSymbol. Allocate pfc::BGR4\_t array on host and allocate pfc::BGR4\_t array on device. Device sets color to pfc::BGR4\_t array for each pixel according to its needed iteration. Device pfc::BGR4\_t array is copied back to host pfc::BGR4\_t array after completion. Host iterates over array and sets each pixel of bmp (with threadpool).

## Results



# Mandelbrot Version 3

## Properties

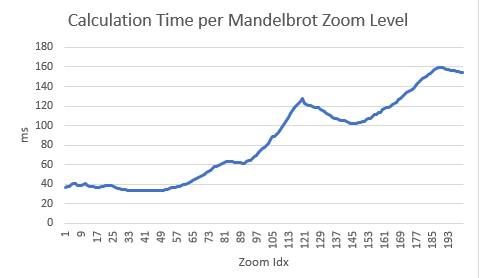
Use of pinned host memory for pfc::BGR4\_t array. The bitmap class is just used as a container for wrapping the pfc::BGR4\_t array. This can be achieved by setting the span of the pinned host memory at bitmap initialization. Bmp(width,height,span(h\_mandelbrot)). This method is significantly faster as it minimizes pressure on CPU as the pfc::BGR4\_t array can be copied directly after finished calculation into the bitmap memory allocation space.

### Memory Copies

### 

### 

## Results

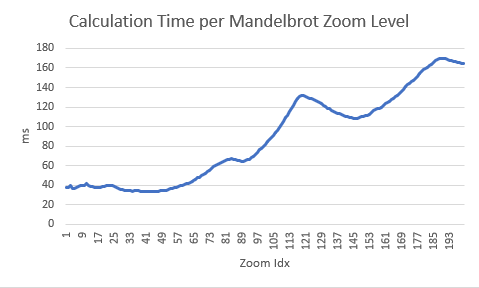


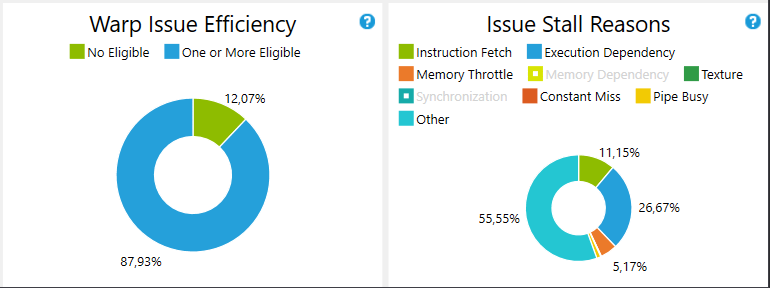
# Mandelbrot Version 4

## Properties

This version unrolls the core Mandelbrot algorithm (#pragma unroll) and minimizes execution dependencies throughout the program.

## Results

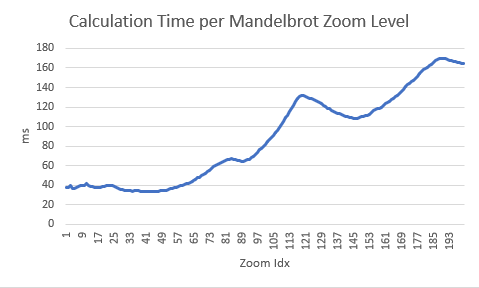




# Mandelbrot Version 5

## Properties

Set compiler flags to ensure max optimization. Use solely L2 cache -Xptax -dlsm=cg. Use -fast-math, maximize speed /O2. Use float4 for boundaries.



# Mandelbrot Version 6

## Properties

Use n streams instead of the default stream. Allocate n times memory on device and host for each stream. Copy to bitmap memory space is performed asynchronous for each stream.

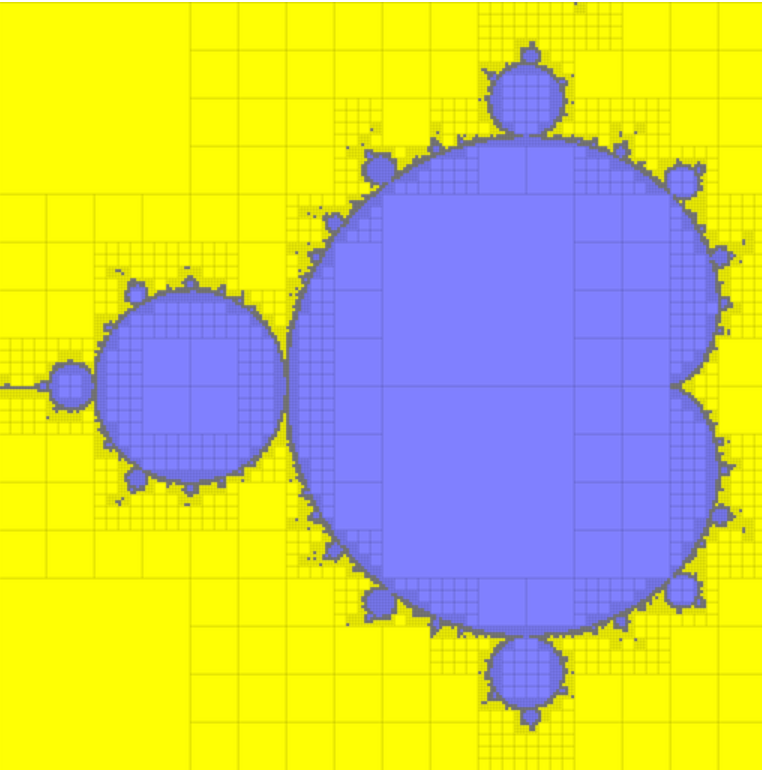
A combination of CPU calculation and GPU calculation with tasks could speed up this even more as we are not fully using CPU.

# Mandelbrot Version 7 (future)

## Properties

As all previous versions do a straight forward per pixel GPU calculation, many threads must wait till other threads in block are finished. Highly inefficient. Dynamic Parallelism with Mariani Silver Algorithm can fix this problem. Recursively call kernel while checking border iterations. If iterations in x and y direction of border are the same, the rectangle in the border can be filled with the same color.

Border Tracing + Mariani Silver Algorithm



## Results

Unfortunately at least compute capability of 3.5 is needed for dynamic parallelism thus I could not test it.