Optimierungen Fraktalberechnung

Zur Analyse der verschiedenen Versionen wurde die Versionsangabe mittels Kommandozeilenparameter implementiert.
Wobei meherere Versionsnummern durch ein Leerzeichen getrennt mitgegeben werden können und somit verschiedene Versionen nacheinander ausgeführt werden können.

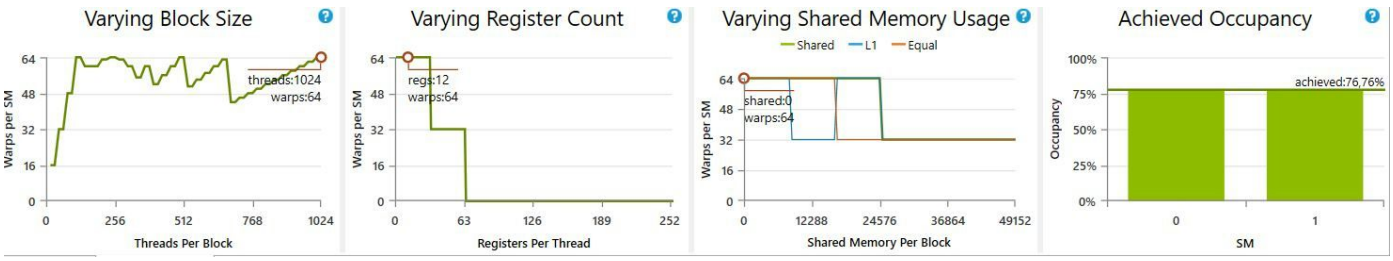
Mttels dem Nsight Debugger ist beim ausführen von verschiedenen Versionen das direkte Vergleichen der Optimierungen möglich.

Das Programm wurde auf einem Gerät mit einer Nvidia GeForce GT 730M ausgeführt und aufgrund der resultierenden Ergebnisse optimiert, die Eigenschaften dieser Grafikkarte können der Datei device_info.xm entnommen werden.

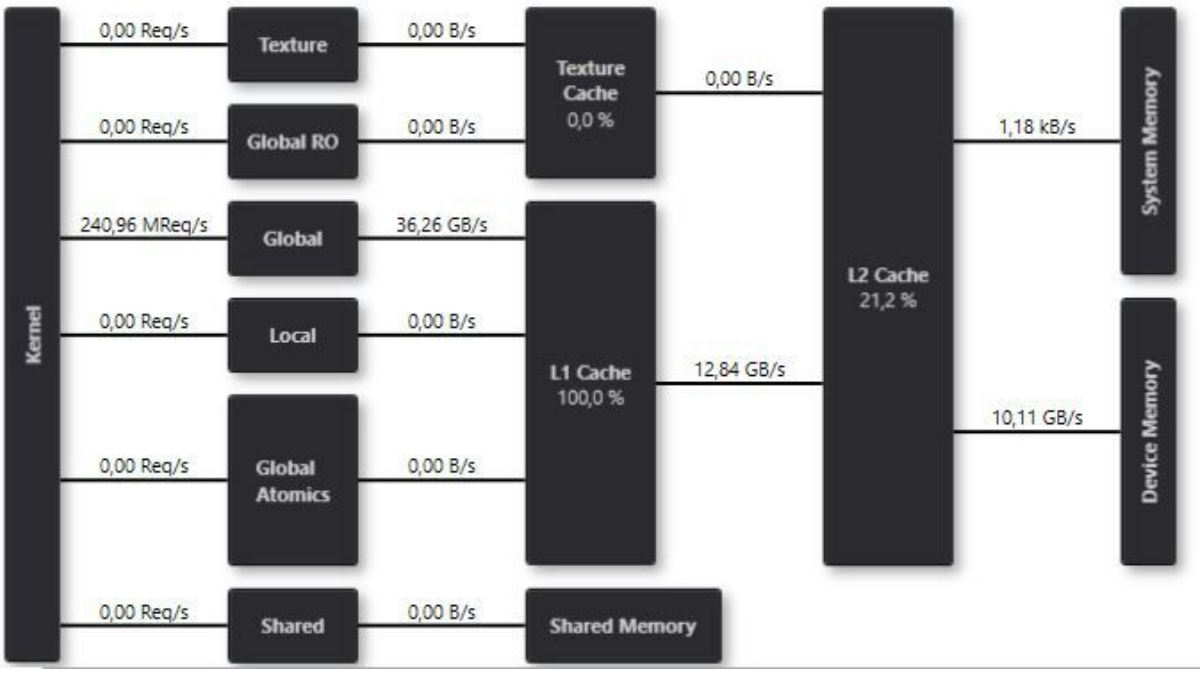
Analyse von Version 0

Analysiert wird das Fraktal das in der größten Zeit (0,0) berechnet werden kann.
Ausgeführte Analyseschritte:

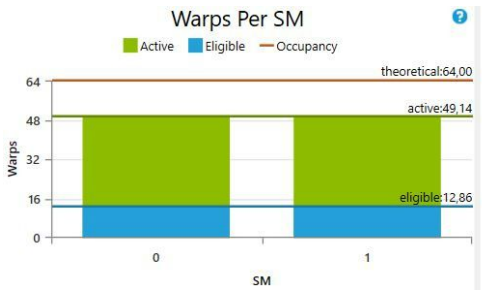
- Occupancy
    - Die momentane Auslastung liegt bei 76,76 %
    - Die Grahen der Block Size, Register Count und Shared Memory Usage zeigen dass die aktuelle Konfiguration optimal hinsichtlich möglicher Warps pro Streaming Multiprocessor (SM) ist.

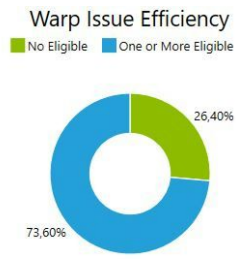


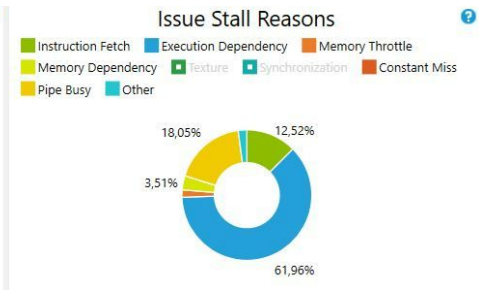
- Memory Analyse:
    - Es wird die maximale Bandbreite nicht erreicht, daher ist der verwendete Kernel nicht memory bound.



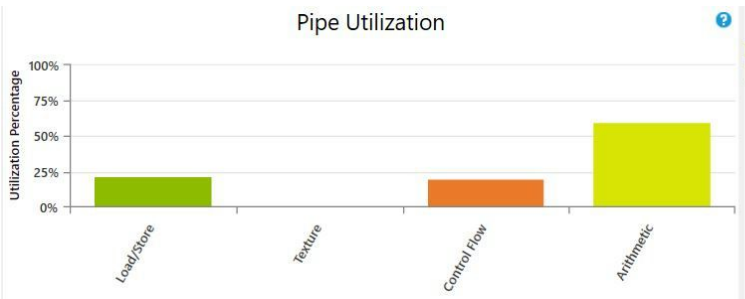
- Warp Issue Efficiency
  - In 26,40% der Zeit sind keine Instruktionen verfügbar, daher stellt der Scheduler kein Problem dar.

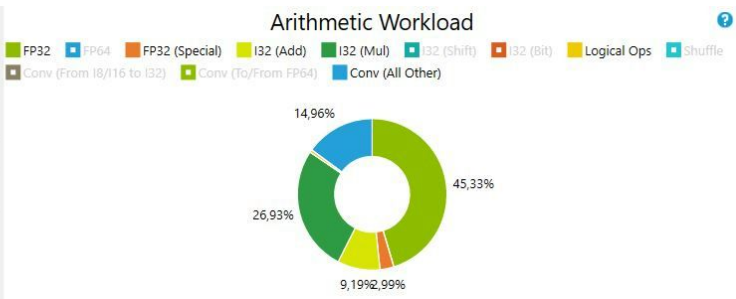






- Pipe Utilization
  - Keine Pipe ist vollkommen asugelastet, daher stellen diese auch kein Problem dar





Optimierung 1

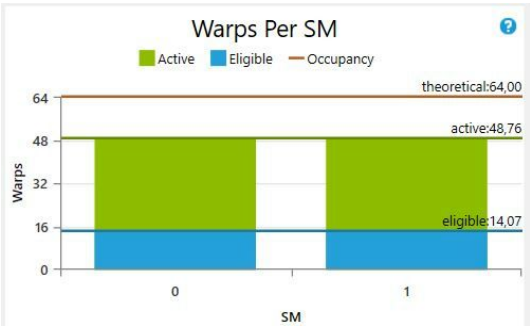
Die Analysen der Version 0 ergaben keine eindeutigen Schwachstellen, daher wird als erste Optimmierung die Verschiebung der Colortable in den constant memory implementiert.

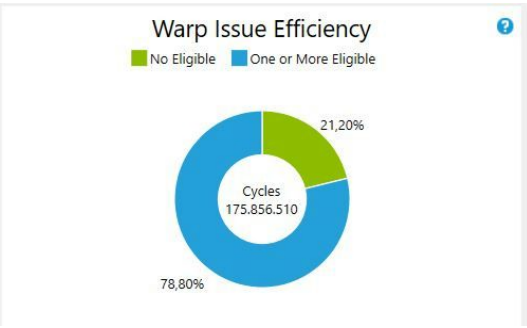
Maßnahmen

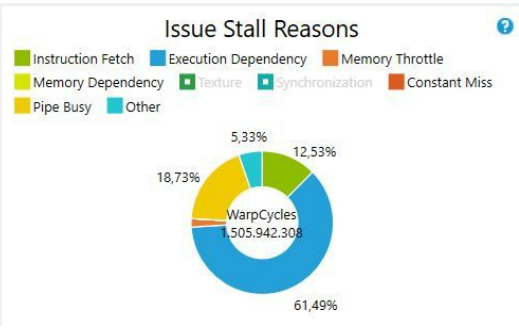
- Verschieben der colortable in den constant memory

Analyse

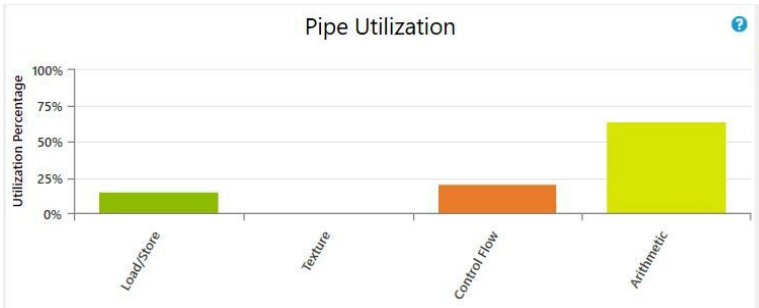
- Warp Issue Efficiency
    - Der Prozentsatz der Zeit zu dem keine Instruktionen verfügbar sind wurde auf 20,2% verringert.

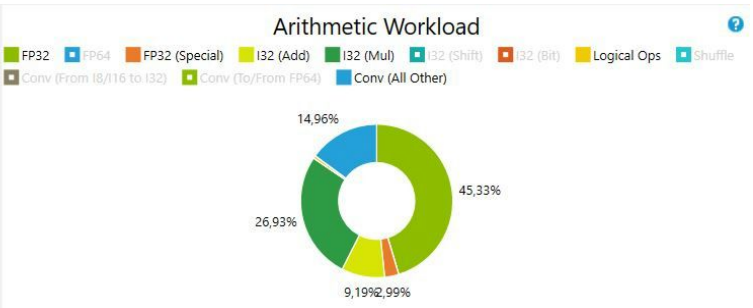




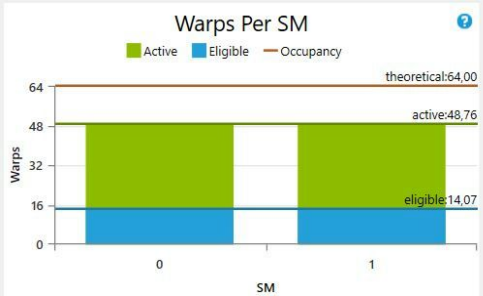


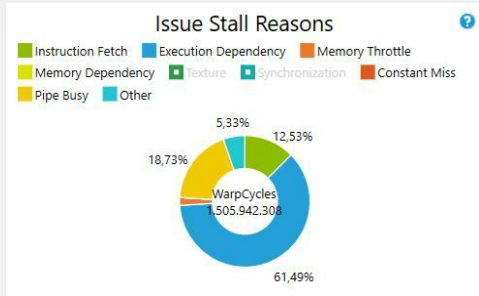
- Pipe Utilization
    - Der Anteil der Load/Store befehle hat sich verringert und der Anteil von arithmetischen Operationen wurde erhöht.





- Runtime
    - Verbesserung der Laufzeit
- Issue Stall Reasons
    - Der häufigste Grund für stalls ist exection dependency





Optimierung 2

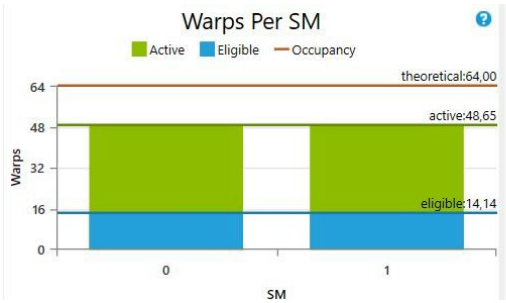
Zur verbesserung der execution dependency wurde Loop-unrolling verwendet.
Die einzige Schleife in der Berechnung der Fraktale ist in der Methode julia::JuliaPixelCalculation::Calc zu finden, daher wurde versucht diese Schleife zu optimieren.
Das Pragma #unroll wurde nicht erkannt, somit war automatisches Loop-unrolling nicht möglich.
Daher wurde ein manuelles Loopunrolling mit dem Faktor 2 implementiert.

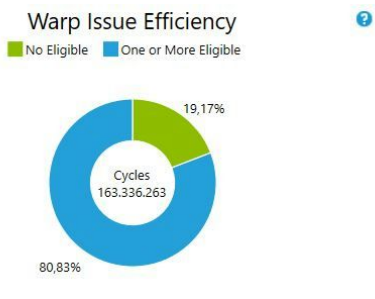
Maßnahmen

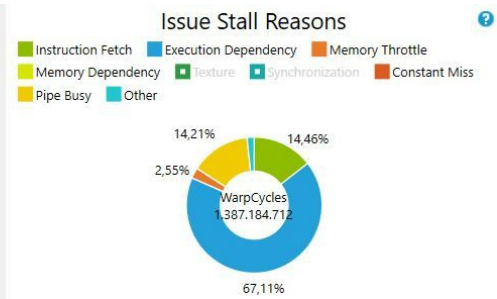
- Manuelles 2-faches loop unrolling in der JuliaPixelCalculation::CalcV2

Analyse

- Runtime:
    - Verbesserung der Laufzeit
- Issue Stall Reasons
    - Der prozentsatz der execution dependency überwiegt immer noch andere







Gründe

- FLOPS
    - Die anzahl der floating point perations per second ist gestiegen

Optimierung 3

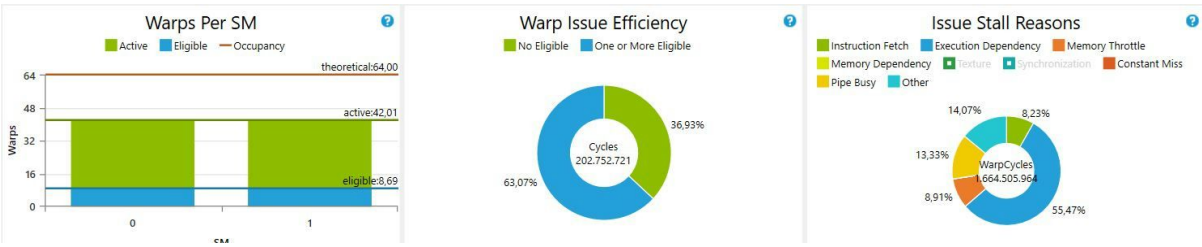
Um die execution dependency weiters zu verbessern werden mehrere Pixel pro Thread berechnet. Dies wurde mittels verschiedener Implementierungen getestet, die manuelle Implementierung von mehreren Pixeln resultierte mit dem besten Ergebniss.

Maßnahmen

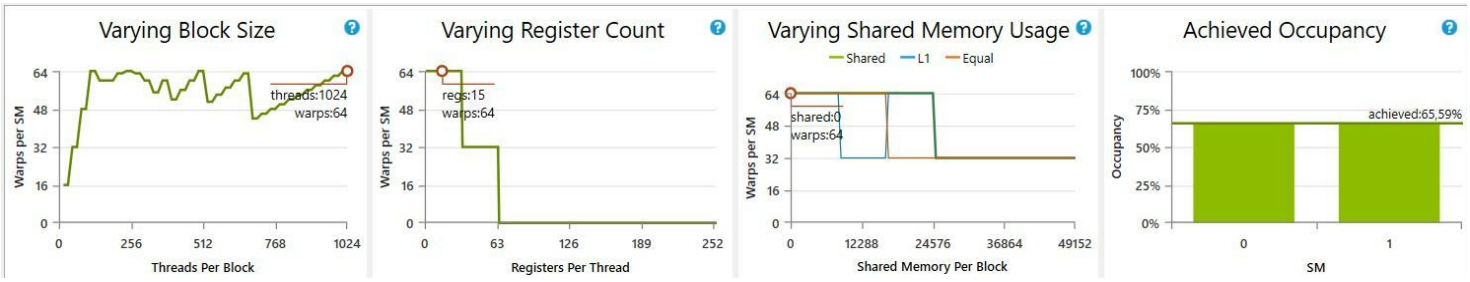
- Berechnung von mehreren Pixeln pro Thread
    - manuelle Berechnung von 2 Pixeln pro Thread

Analyse

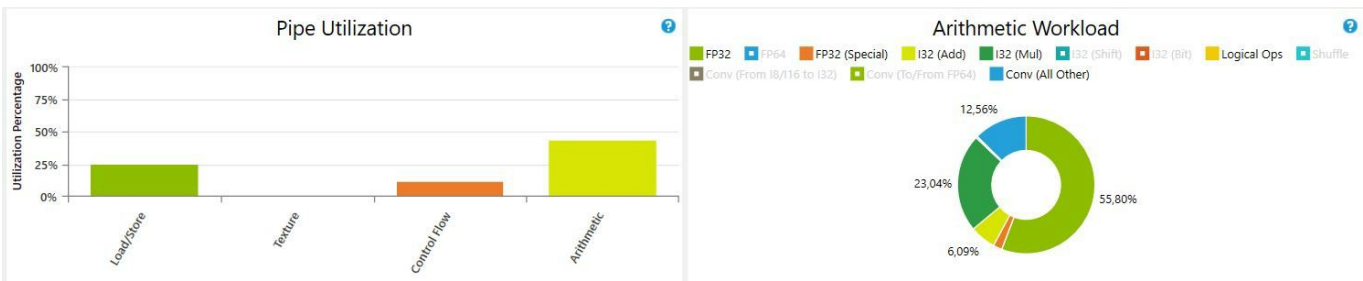
- Runtime:
    - verschlechterte Laufzeit
- Issue Stall Reasons
    - Der Prozentsatz der Zeit zu dem keine Instruktionen verfügbar sind wurde deutlich erhöht



- Occupancy
    - Verschlechterung der gesamten Occupancy auf 65,59 %



- Pipe Utilization
    - Auslastung der arithmetischen und Kontroll Pipe wurde verrigert, dazu steigt die Auslastung von Load/Store



Optimierung 4

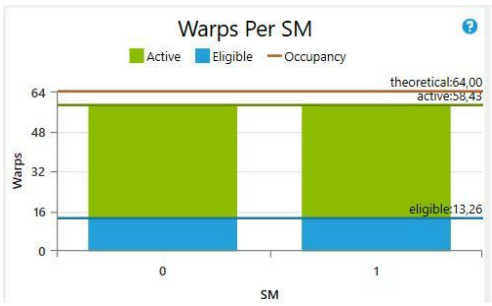
Um eine Verbesserung der Occupancy zu erreichen wurde die Blockgröße verändert und eine fixe Blockgröße von 32 x 32 Threads implementiert. Diese Blockgröße wurde gewählt da die vorhandene Graikkarte 1024 Threads pro Block ermöglicht und mit 32 Threads eine quadratische Abarbeitung möglich ist.

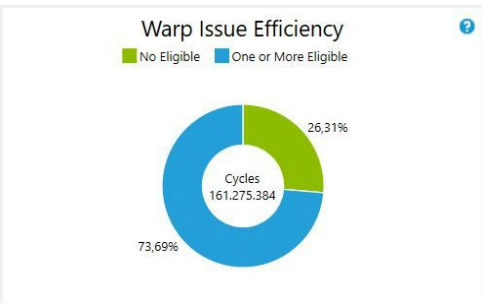
Maßnahmen

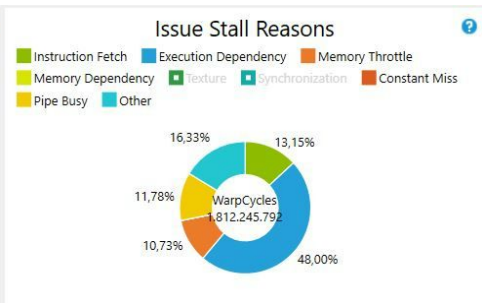
- Änderung der Blockgröße
    - fixe Anzahl von 32 X-Threads und 32 Y-Threads in einem Block

Analyse

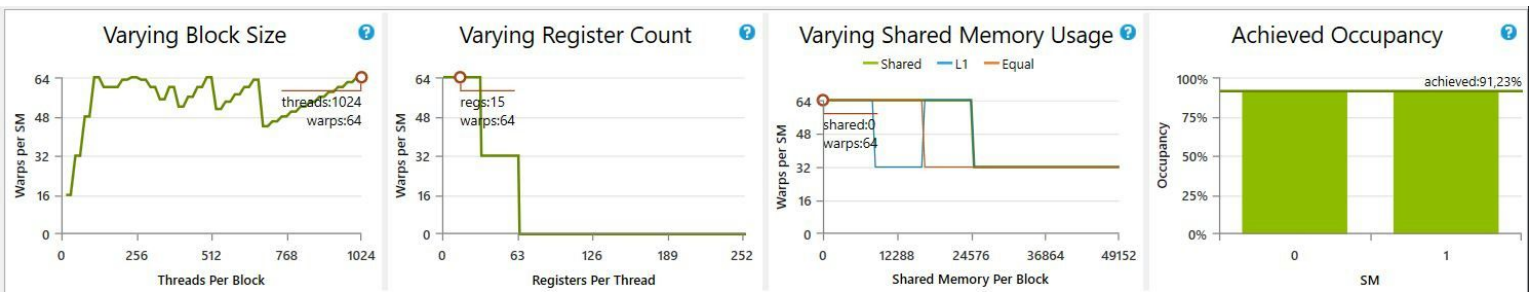
- Runtime
    - verbesserung der Laufzeit
- Issue Efficiency
    - active Warps per SM sind deutlich gestiegen und nähern sich theoretischer Grenze von 64 an
    - Der Anteil der keine zuteilbaren (no eligible) warp zeitpunkte sank
    - Der Prozentsatz der execution dependency sank





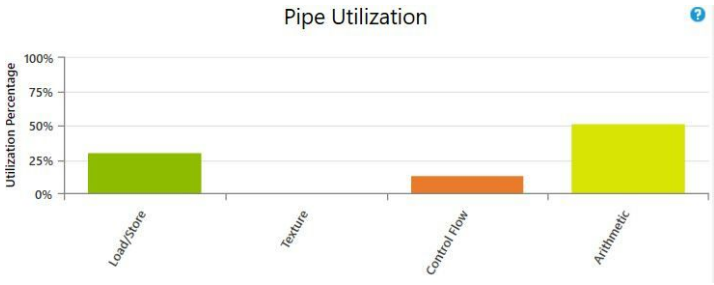


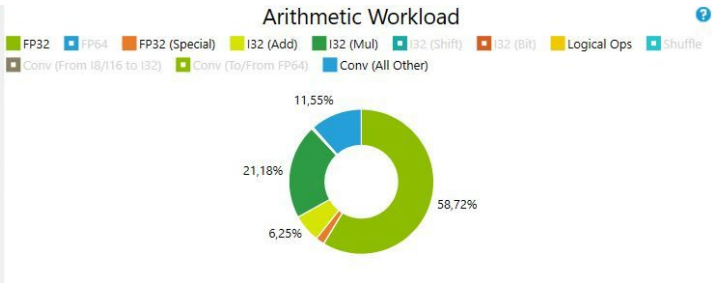
- Occupancy
    - Achieved Occupancy steigt auf 91,23 %



- Pipe Utilization

- Auslastung der arithmetischen und Kontroll Pipe wurde verrigert, dazu steigt die Auslastung von Load/Store





Optimierung 5

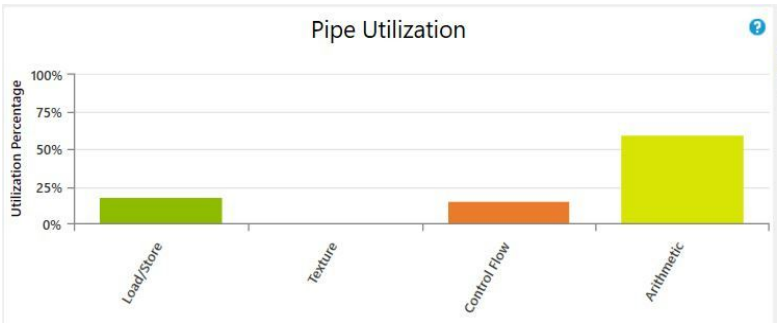
Der Optimierungschritt 3 führte lediglich zu Verschlechterungen, daher wurde die Berechnung von mehreren Pixeln in dieser Optimierung wieder entfernt.

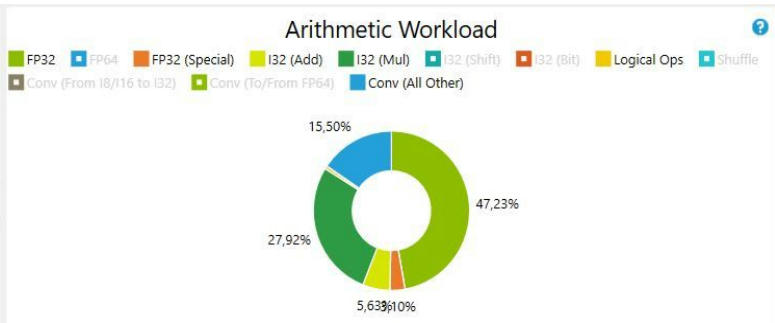
Maßnahmen

- Berechnung von 1 Pixel per Thread

Analyse

- Runtime
  - Leichte Verbesserung der Laufzeit
- Pipe Utilization
  - Auslastung der arithmetischen und Kontroll Pipe wurde vergrößert, dazu sinkt die Auslastung von Load/Store





Zusammenfassung

Die effizienteste Version ist die Version 5, darin werden folgenden Optimierungen verwendet:

- Speicherung der Colortable im constant memory
- Manuelles 2-faches Loop-Unrolling der innersten Berechnungsschleife
- Fixe Blockgröße von 32 x 32 Pixel pro Block

Ergebnisse der Laufzeit

	Function Name	Grid Dimensions	Block Dimensions	Start Time (µs)	Duration (µs)	Occupancy	Registers per Thread	Static Shared Memory per Block (bytes)	Dynamic Shared Memory per Block (bytes)	Cache Configuration Executed	Global Caching Requested	Global Caching Executed	Local Memory per Thread (bytes)	Device Name
1	FractalCalculationV0	{5, 5000, 1}	{1024, 1, 1}	8,576,242.837	26,560.000	100.00 %	12	0	0	PREFER_SHARED	N/A	N/A	0	GeForce GT 730M
2	FractalCalculationV1	{5, 5000, 1}	{1024, 1, 1}	16,843,352.629	25,279.200	100.00 %	12	0	0	PREFER_SHARED	N/A	N/A	0	GeForce GT 730M
3	FractalCalculationV2	{5, 5000, 1}	{1024, 1, 1}	24,361,785.909	23,431.232	100.00 %	13	0	0	PREFER_SHARED	N/A	N/A	0	GeForce GT 730M
4	FractalCalculationV3	{3, 5000, 1}	{1024, 1, 1}	32,641,470.389	29,191.360	100.00 %	15	0	0	PREFER_SHARED	N/A	N/A	0	GeForce GT 730M
5	FractalCalculationV4	{79, 157, 1}	{32, 32, 1}	39,896,946.261	23,043.968	100.00 %	15	0	0	PREFER_SHARED	N/A	N/A	0	GeForce GT 730M
6	FractalCalculationV5	{157, 157, 1}	{32, 32, 1}	47,198,320.021	21,652.608	100.00 %	13	0	0	PREFER_SHARED	N/A	N/A	0	GeForce GT 730M

Die Laufzeit der Version 0 beträgt 25.560 µs, diese wurde mit der Version 5 auf 21.652 µs verringert und ergibt somit eine Verbesserung der Laufzeit um 15,3 %.