Westfälische
Wilhelms-Universität
Münster

› Grafikkarten-Architektur

Parallele Strukturen in der GPU

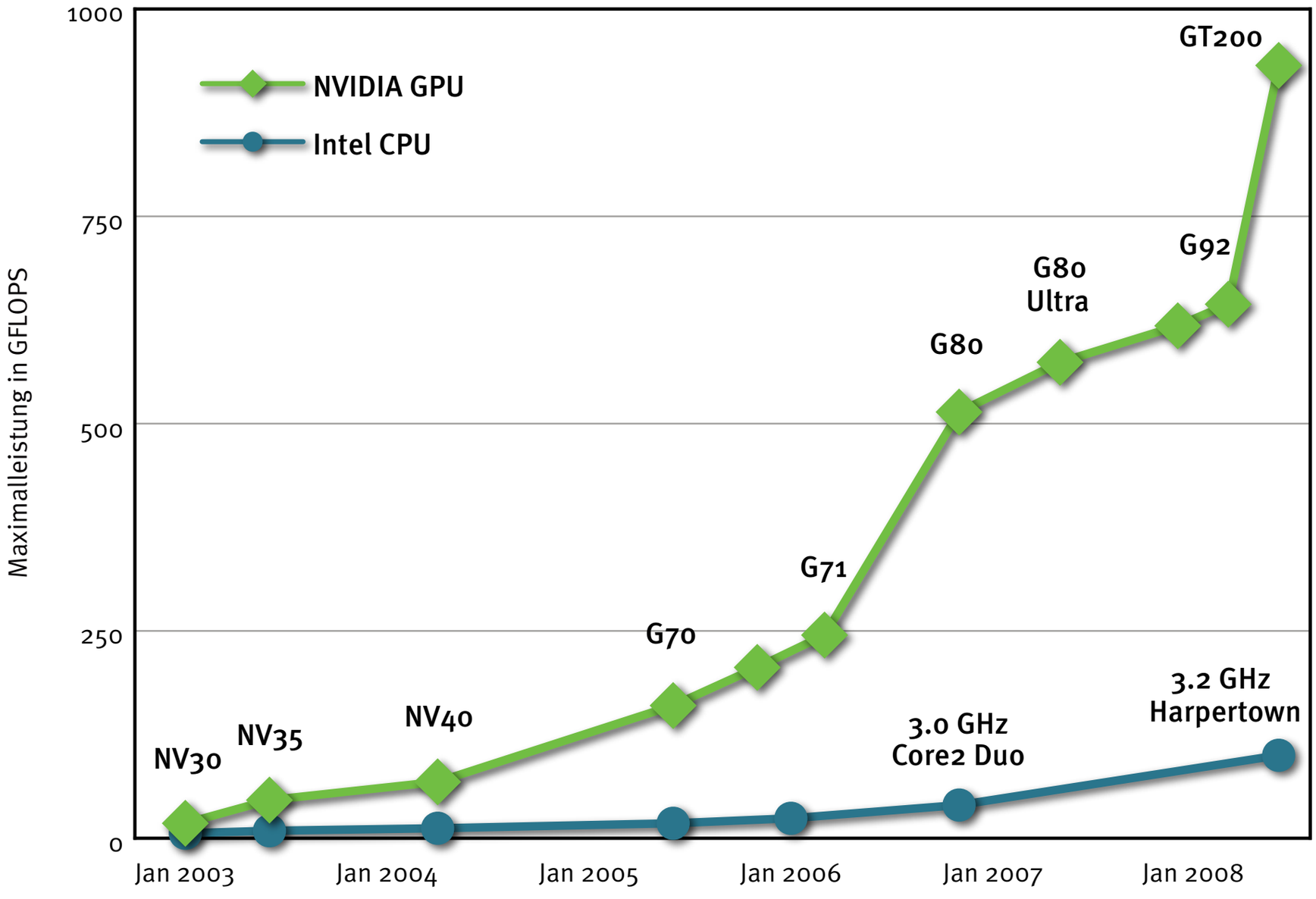
wissen.leben
WWU Münster

Name: Sebastian Albers | E-Mail: s.albers@wwu.de

› Inhalt

- › CPU und GPU im Vergleich
- › Rendering-Pipeline
- › Shader
- › GPGPU
- › Nvidia Tesla-Architektur

› Motivation: Warum auf Grafikkarten rechnen?



GT200 = GeForce GTX 280

G92 = GeForce 9800 GTX

G80 = GeForce 8800 GTX

G71 = GeForce 7900 GTX

G70 = GeForce 7800 GTX

NV40 = GeForce 6800 Ultra

NV35 = GeForce FX 5950 Ultra

NV30 = GeForce FX 5800

Quelle: Nvidia, April 2009

wissen.leben WWU Münster

› CPU und GPU im Vergleich

- › Central Processing Unit (CPU)
  - › Wenige Kerne, hohe Taktung
  - › Cache-Hierarchien
  - › Niedrige Speicherzugriffszeiten (Latenz)
- › Beispiel: Intel Core i7-975
  - › Preis: ca. 900 €
  - › 4 Kerne
  - › Taktung: 3.33 GHz
  - › Bandbreite je nach verwendetem RAM
    - › 3 × 2 GB DDR3-1600: 26.7 GB/s
    - › 3 × 2 GB DDR3-800: 15.5 GB/s

- › Graphics Processing Unit (GPU)
  - › 3D-Rendering hochgradig datenparallel
    ➠ GPU für datenparallelen Code optimiert
  - › Viele Prozessoren, niedrige Taktung
  - › Hoher Datendurchsatz (Speicherbandbreite)
- › Beispiel: Nvidia Geforce GTX 295 (2 GPUs)
  - › Preis: ca. 450 €
  - › 2 × 240 Prozessoren
  - › Taktung: 1.48 GHz
  - › Speicher: 2 × 896 MB GDDR3
  - › Bandbreite: 2 × 111.9 GB/s

› Bildsynthese / Rendering

- › Rendering: Erzeugung eines diskreten 2D-Bildes aus einer 3D-Szene
  - › Programmierer spezifiziert Szene durch Primitive (Dreiecke, Linien, etc.)
    - › Eckpunkt eines Primitivs heißt Vertex
  - › Aus Primitiven werden diskrete Fragmente erzeugt
    - › Fragment enthält Informationen wie 2D-Position, Tiefenwert, Farbe, Alphakanal
  - › Fragmente werden zu Pixeln kombiniert

- › Rendering findet in einer Pipeline statt
  - › Programmierbare Stages der Rendering-Pipeline definiert durch Shader-Funktionen:
    - › In speziellen Shader-Programmiersprachen geschrieben
    - › Definiert als sequentielle Funktion, die für einzelne Dateneinheit berechnet wird

› Shader

- › Shader-Arten:
  - › Vertex-Shader: Verschiebung von Vertices
    - › 1 Vertex → 1 Vertex
  - › Geometry-Shader: Manipulation von Primitiven
    - › 1 Primitiv → beliebig viele Primitive
  - › Fragment-Shader / Pixel-Shader: Änderung der Fragment-Farbe
    - › 1 Fragment → 1 Fragment
- › Viel Geometrie, hohe Auflösungen
  - ➠ Pro Bild werden sehr viele Shader-Funktionen aufgerufen
- › Shader-Berechnungen voneinander unabhängig
  - ➠ Parallele Berechnung der Shader-Funktionen für verschiedene Daten auf so genannten Shader-Einheiten der Grafikkarte

› General Purpose Computing On GPUs (GPGPU)

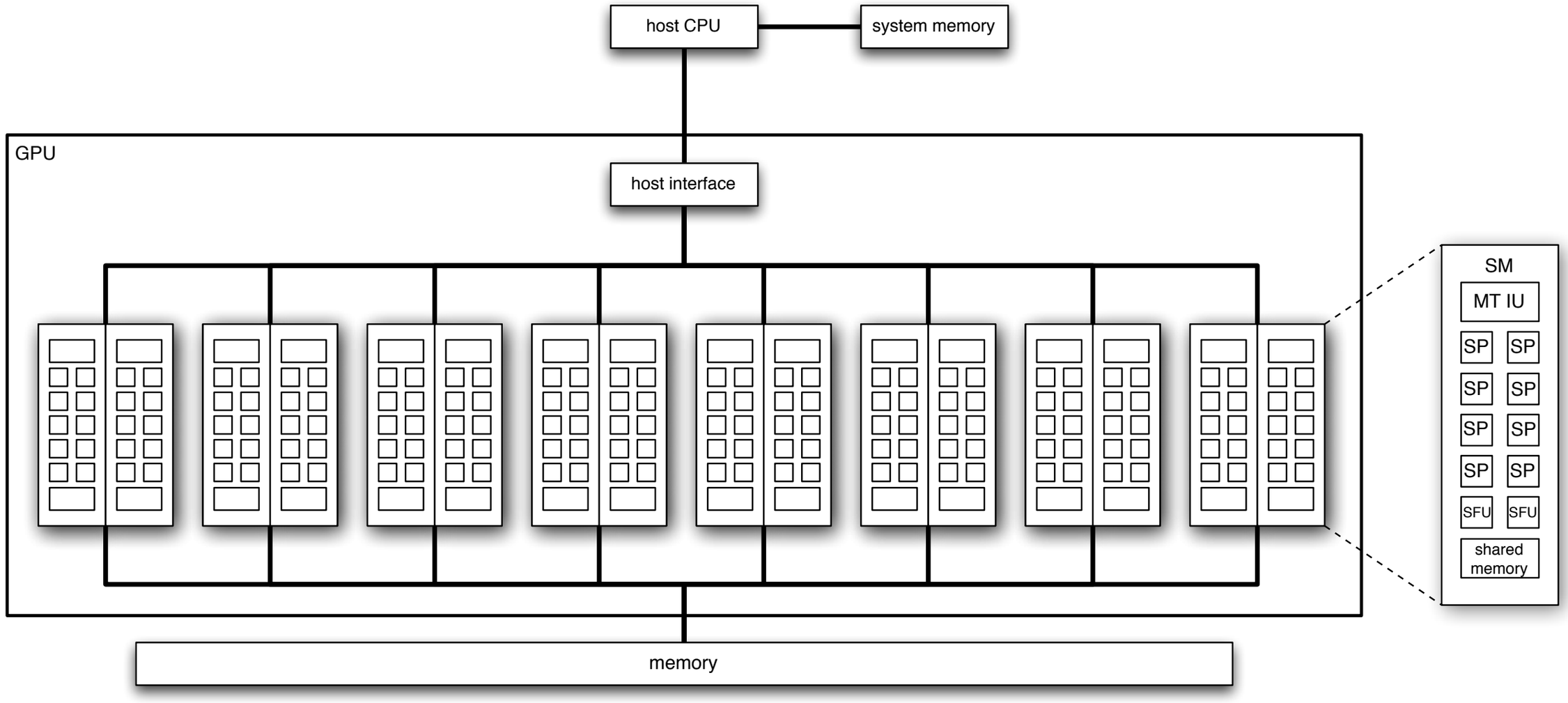
- › GPGPU
  - › Nicht-grafikspezifische Berechnungen auf GPUs
  - › Effizient bei hoher Datenparallelität
  - › Anfänge: Verwendung von OpenGL und Shader-Programmiersprachen
- › Nvidia Tesla-Architektur (ab Geforce 8, Erscheinungsjahr: 2007)
  - › Unified-Shader-Architektur
    - › Jede Shader-Einheit kann sämtliche Shader-Berechnungen durchführen
      - ➠ Bessere Auslastung der GPU
  - › Graphics Mode / Compute Mode
    - › Graphics Mode: 3D-Rendering
    - › Compute Mode: Teile der Grafikpipeline deaktiviert

› Beispiel: Geforce 8800 GTX

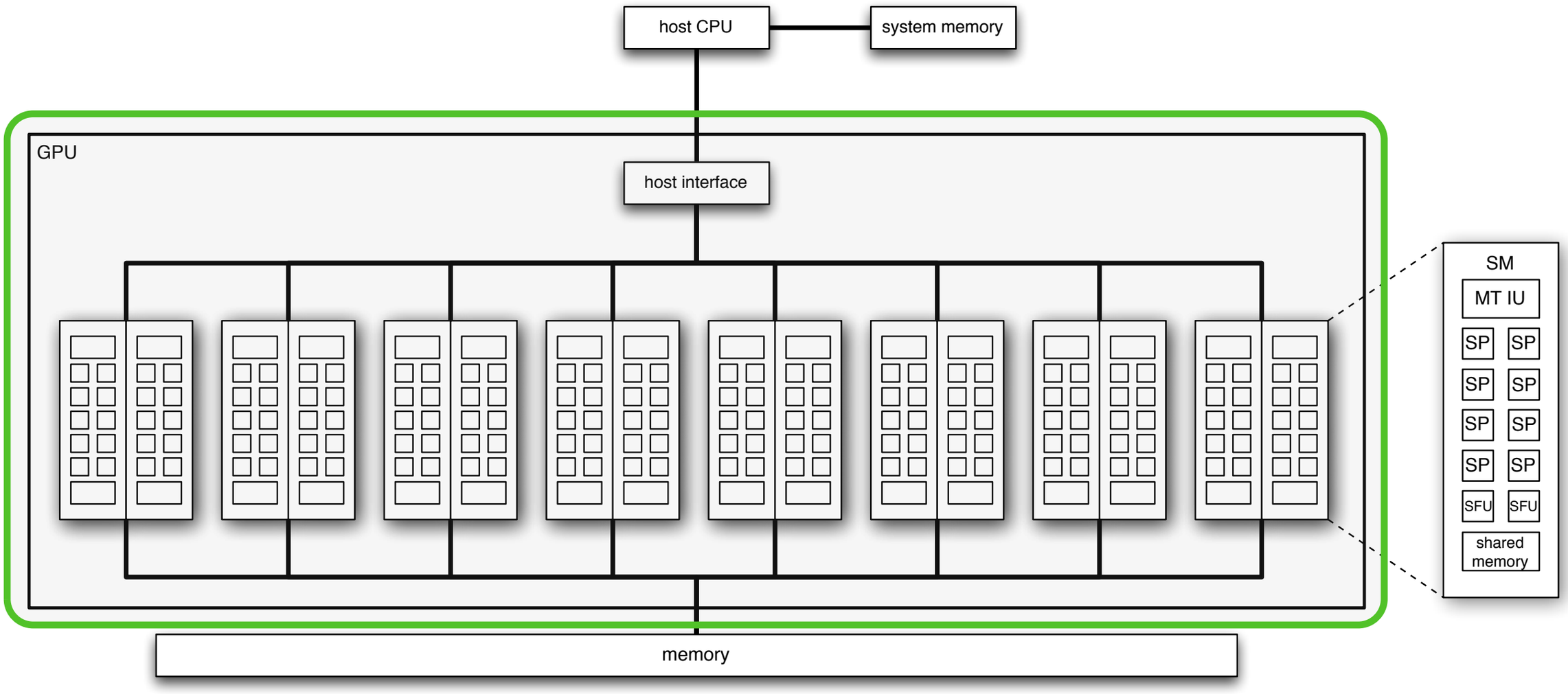
- 128 Streaming Processors (Taktung: 1.35 GHz)
  - Organisiert in 16 Streaming Mulitprocessors
- 768 MB GDDR3 (maximale Bandbreite: 86,4 GB/s)
- Theoretische Maximalleistung: 518 GFLOPS

Name: Sebastian Albers | E-Mail: s.albers@wwu.de | Datum: 03.07.2009

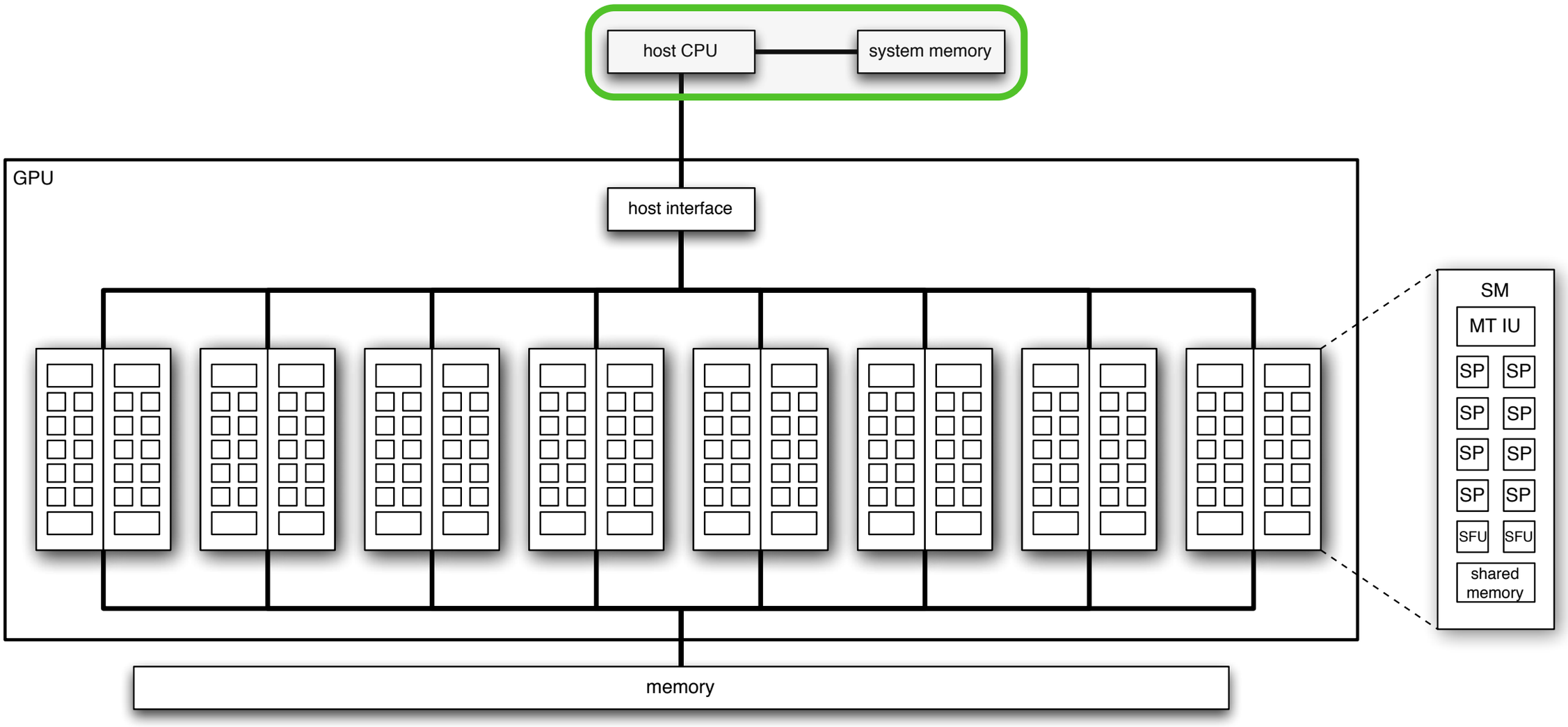
› Geforce 8800 GTX



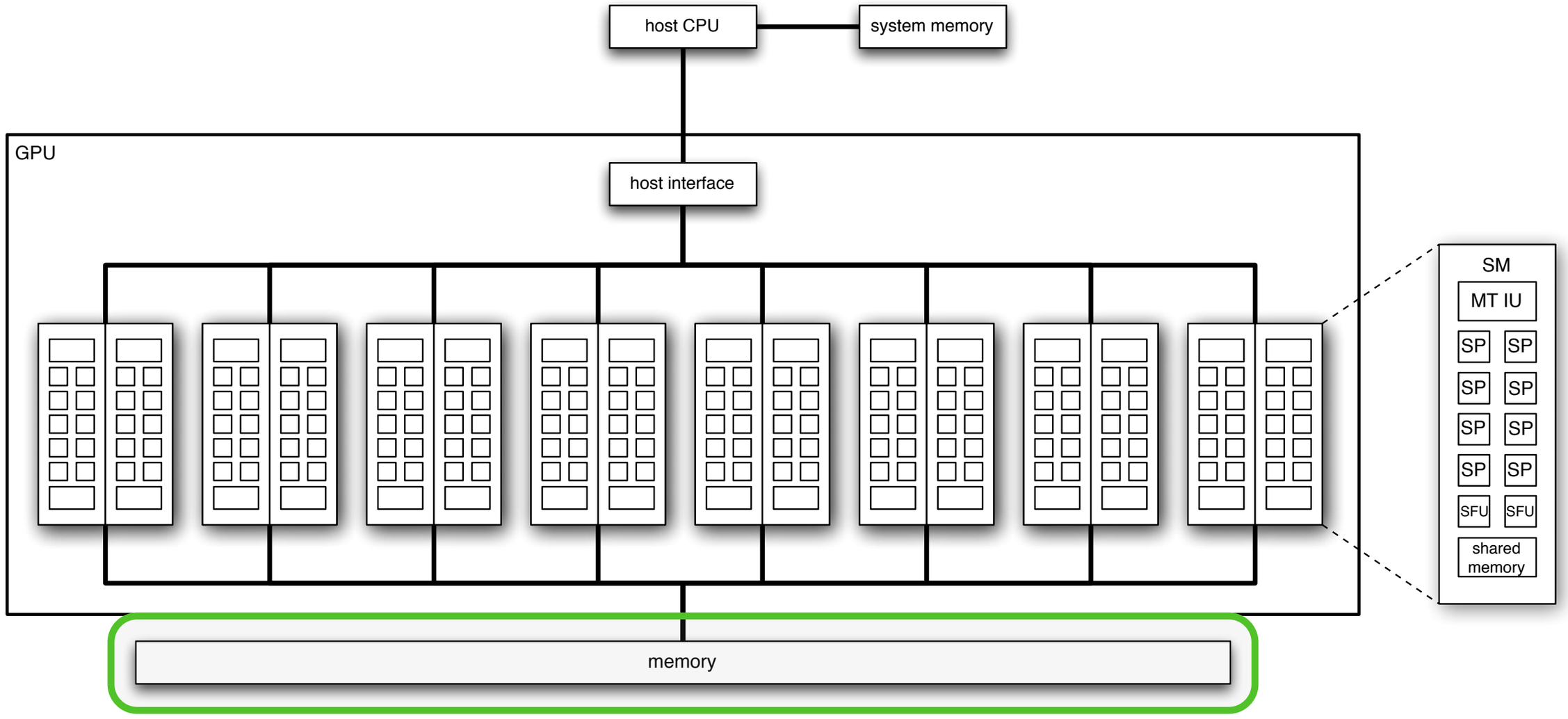
› Geforce 8800 GTX



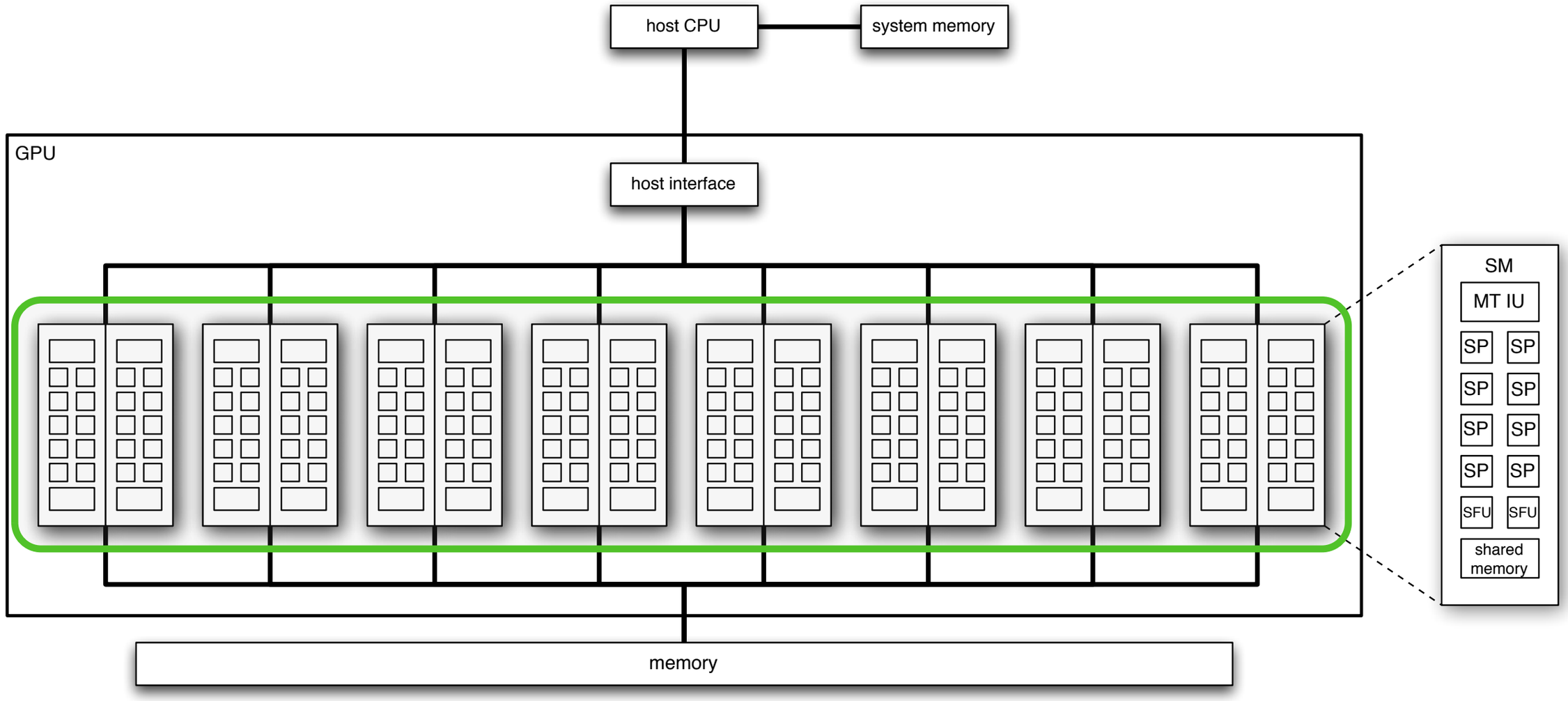
› Geforce 8800 GTX



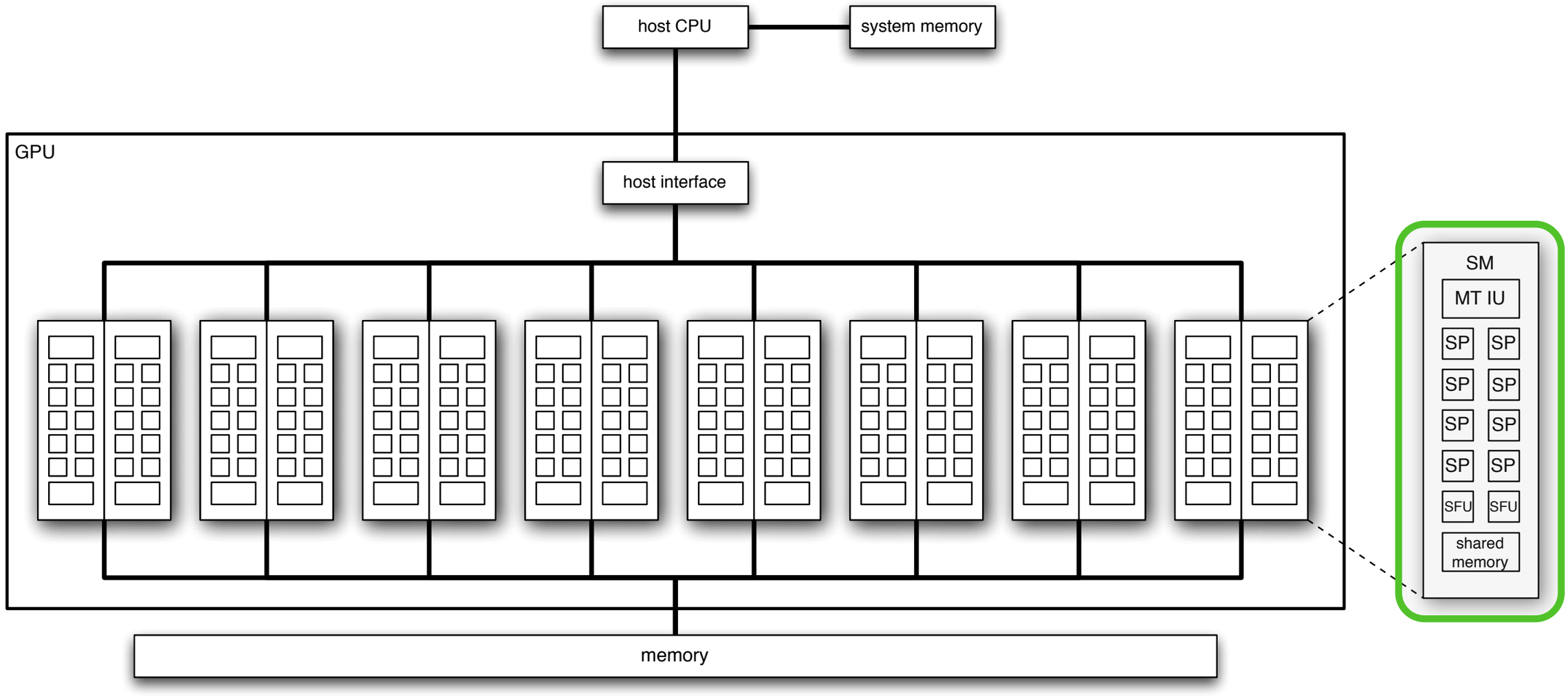
› Geforce 8800 GTX



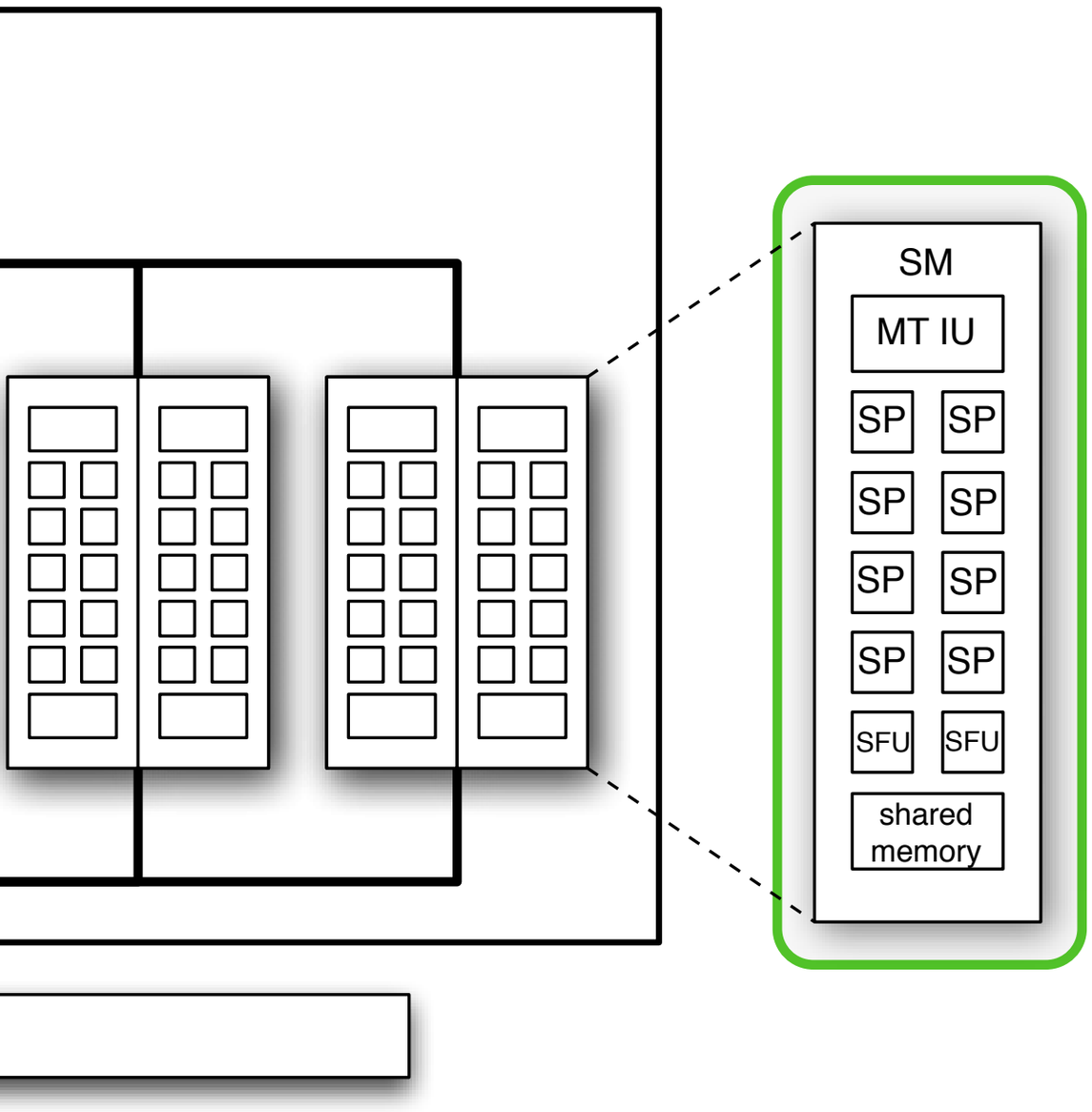
› Geforce 8800 GTX



› Geforce 8800 GTX



› Streaming Multiprocessor



› Streaming Multiprocessor (SM)

› Multithreaded Instruction Unit (MT IU)

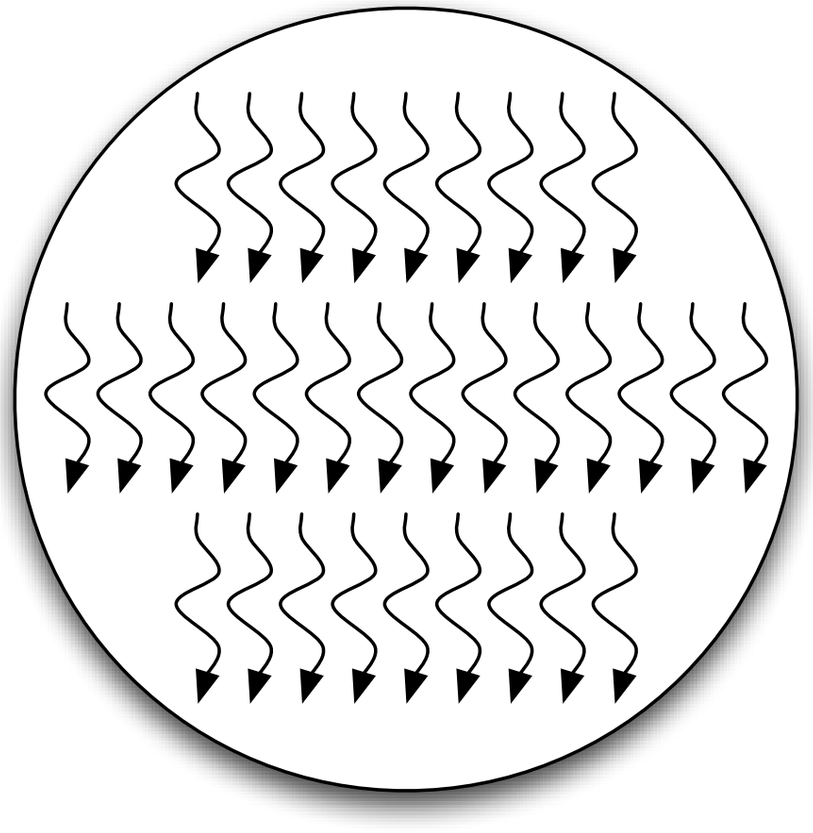
› Streaming Processor (SP)

› Special Function Unit (SFU)

› Shared Memory

› Multithreaded Instruction Unit (MI UT)

- › MI UT verwaltet Ausführung von Threads auf einem SM
- › 32 Threads werden zu einem Warp zusammengefasst
- › Bis zu 24 Warps werden von einem SM verwaltet

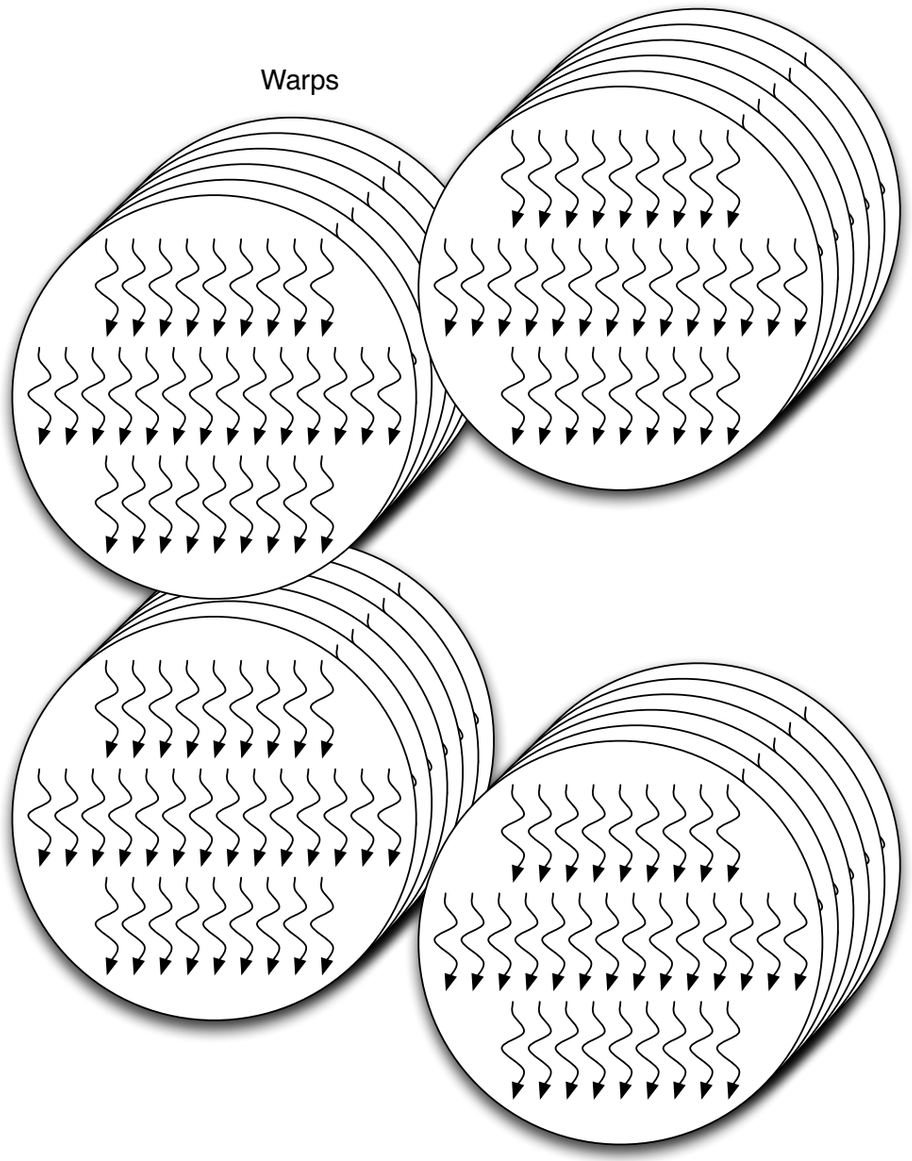


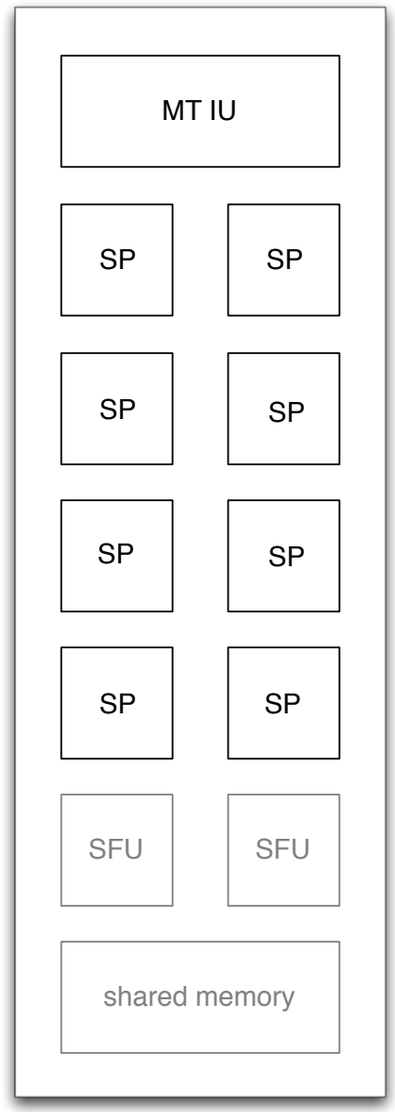
Warp

› Hardware-Multithreading

- › Single Instruction Multiple Thread (SIMT)
  - › Threads eines Warps führen das gleiche Programm aus
  - › Verzweigungen im Programmcode
    → Threads können unterschiedliche Instruktionen ausführen
  - › In Anlehnung an die Flynnsche Klassifikation → SIMD
- › 32 Threads pro Warp
  - › Best case: alle Threads führen identische Instruktionen aus
  - › Worst case: alle Threads führen paarweise verschiedene Instruktionen aus, dann: 1/32 der Maximalleistung
  - › Probleme lassen sich häufig mit wenigen Verzweigungen lösen
    ➠ Meist gute Ausnutzung der Ressourcen möglich

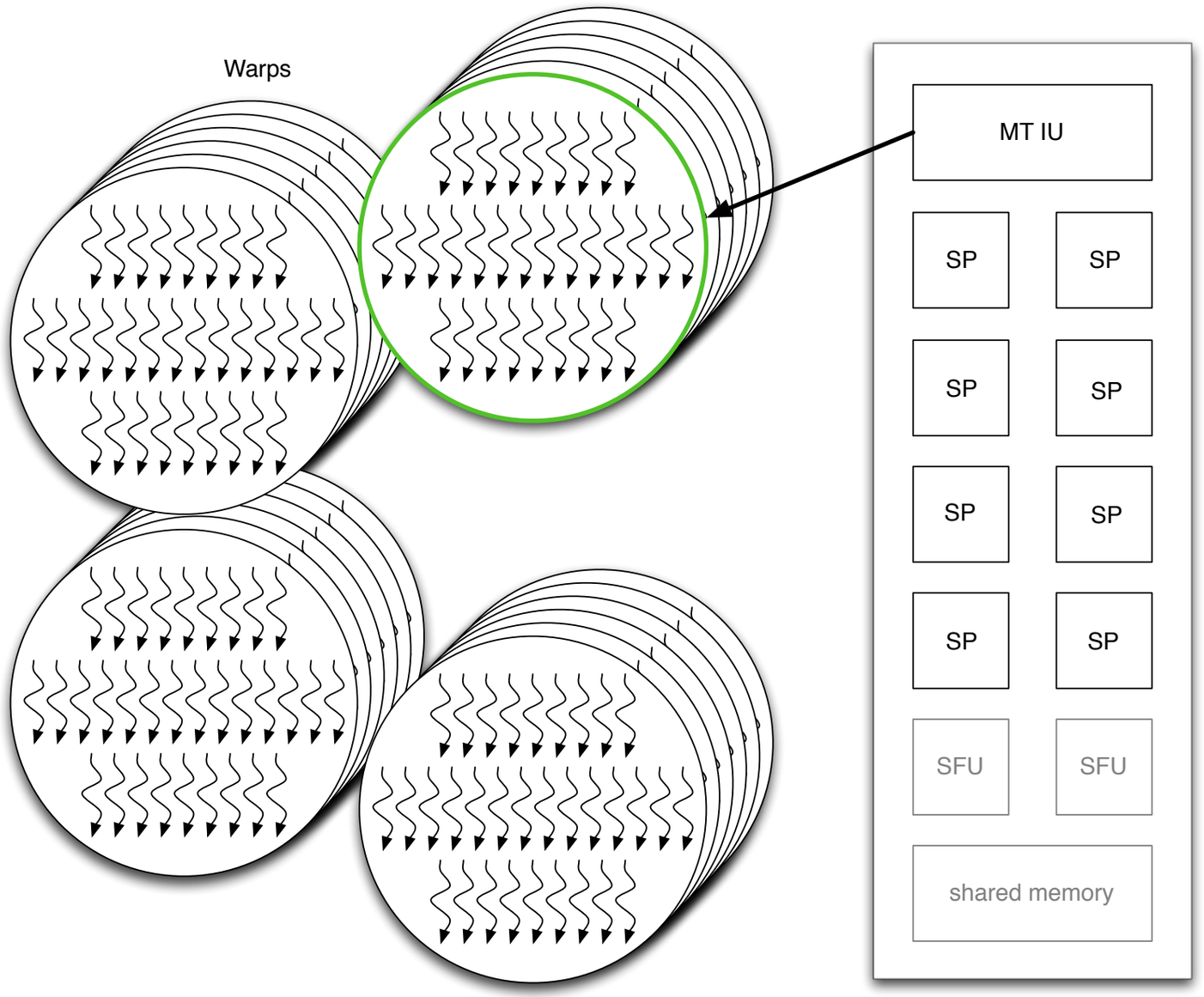
› Hardware Multithreading





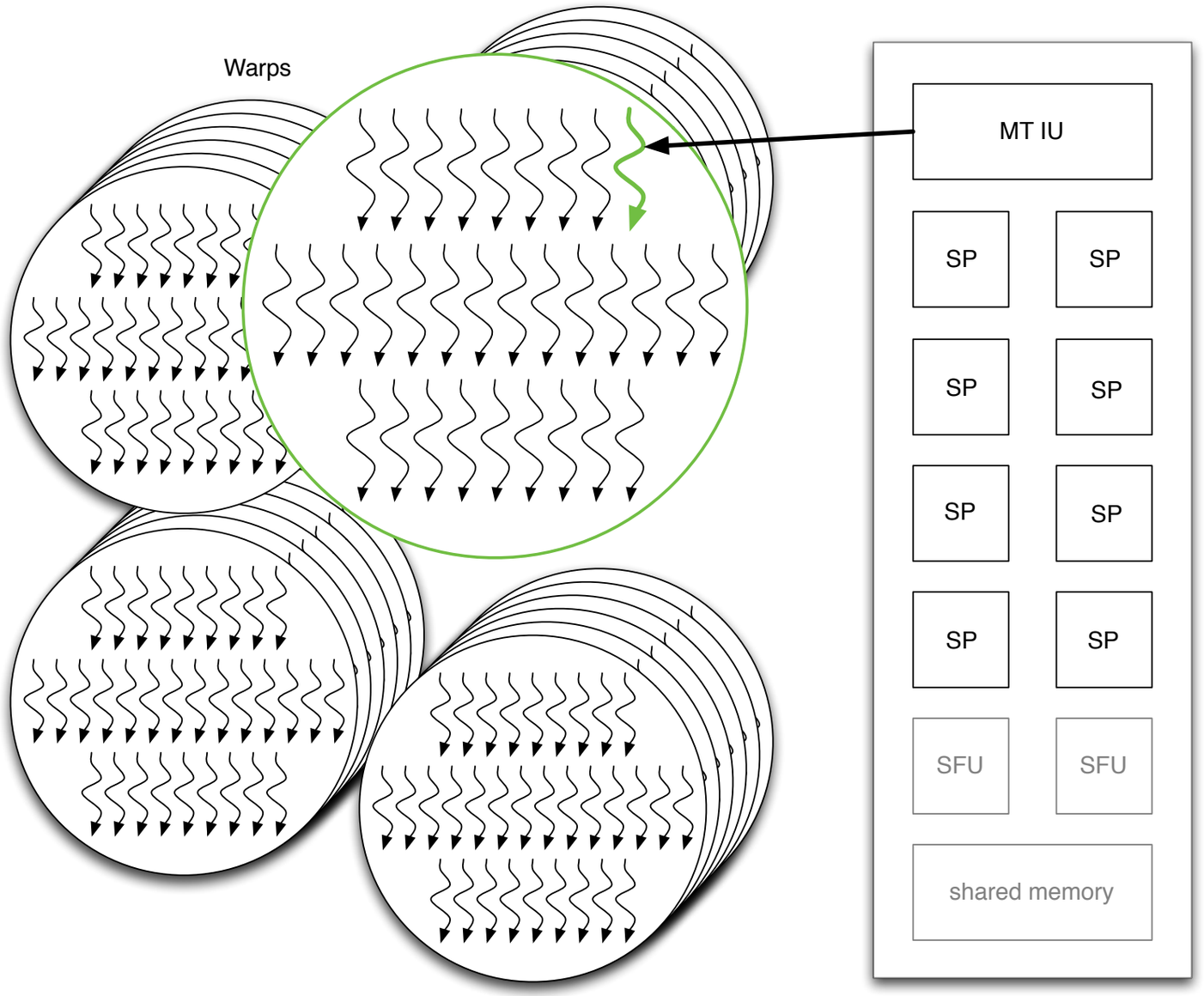
› Hardware Multithreading

1. MT IU wählt Warp zur Ausführung aus



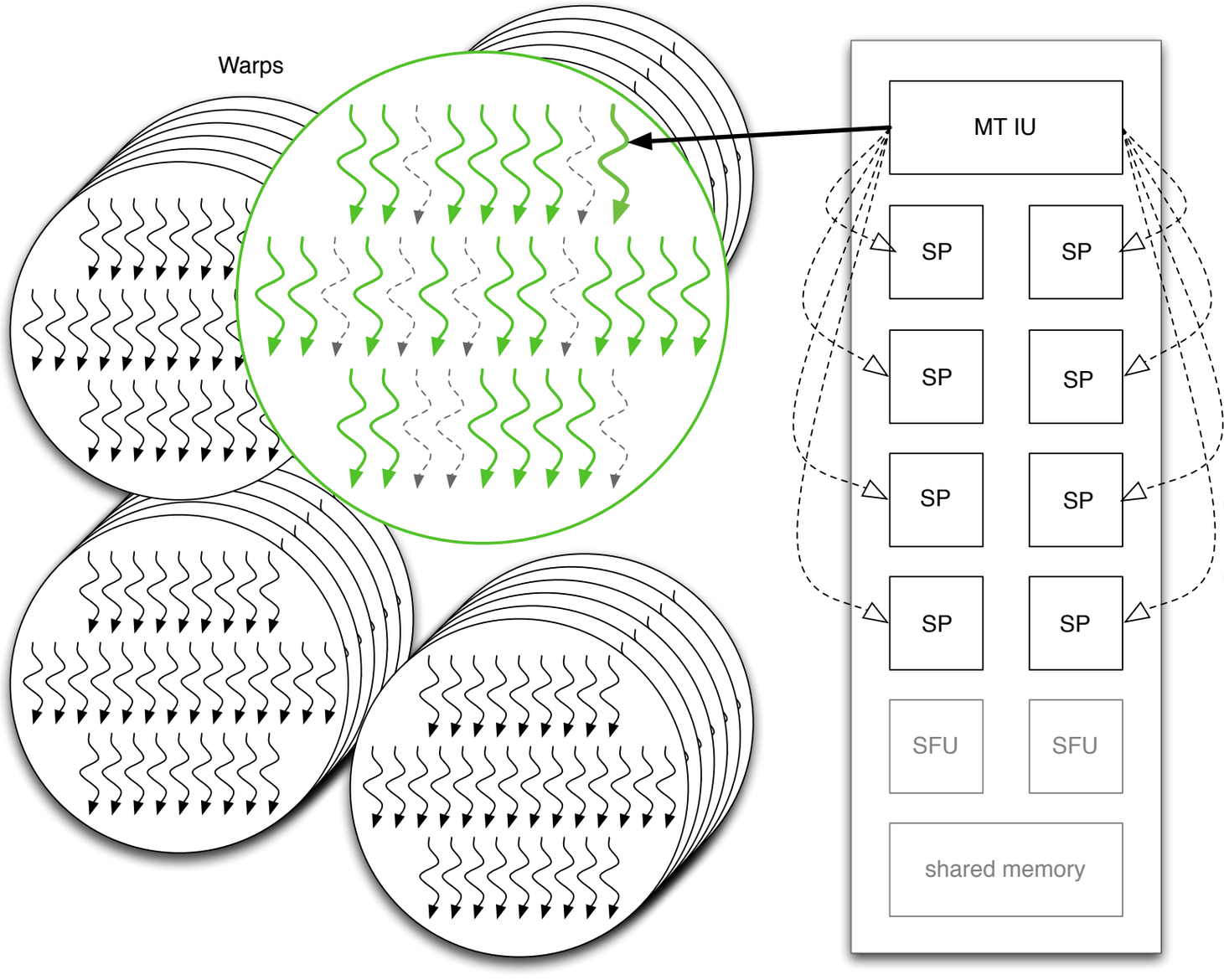
› Hardware Multithreading

1. MT IU wählt Warp zur Ausführung aus
2. MT IU wählt Thread aus dem Warp aus



› Hardware Multithreading

3. Instruktion dieses Threads wird für den gesamten Warp ausgeführt
   - 3.1. Threads, welche die Instruktion ausführen müssen, beteiligen sich
   - 3.2. Threads, welche die Instruktion nicht ausführen müssen, warten



› Zusammenfassung

- Moderne GPUs: viele Prozessoren → hohes Leistungspotential
- Datenparallele Berechnungen lassen sich auf GPUs effizient durchführen
  - Rendering: parallele Ausführung vieler Shader-Funktionen
  - Auf GPUs sind auch nicht-grafikspezifische Berechnungen möglich (GPGPU)
- Tesla-Architektur
  - Beispiel: (stark vereinfachter) Aufbau einer Geforce 8800 GTX
  - Hardware Multithreading
    - 32 Threads werden zu einem Warp zusammengefasst und gemeinsam auf einem Streaming Multiprocessor ausgeführt (SIMT)

Vielen Dank für Ihre Aufmerksamkeit!

Noch Fragen?